RM0090
参考手册

STM32F40xxx、STM32F41xxx、STM32F42xxx、STM32F43xxx
基于 ARM 内核的 32 位高级 MCU

前言

本参考手册面向应用开发人员，提供有关使用 STM32F405xx/07xx、STM32F415xx/17xx、STM32F42xxx 和 STM32F43xxx 微控制器存储器与外设的完整信息。

STM32F405xx/07xx、STM32F415xx/17xx、STM32F42xxx 和 STM32F43xxx 构成一个微控制器系列，各产品具有不同的存储器大小、封装和外设。

有关订购信息以及器件的机械与电气特性，请参见数据手册。

有关 ARM Cortex™-M4F 内核的信息，请参见《Cortex™-M4F 技术参考手册》。

相关文档

意法半导体网站 (http://www.st.com) 提供以下文档：

- *STM32F40x 和 STM32F41x 数据手册*
- *STM32F42x 和 STM32F43x 产品简介*
- 有关带 FPU 的 ARM Cortex™-M4 内核的信息，请参见《STM32F3xx/F4xxx Cortex™-M4 编程手册》*(PM0214)*。

表 1. 适用产品

产品系列	料号和产品类别
微控制器	STM32F405xx、STM32F407xx、STM32F415xx、STM32F417xx、STM32F427xx、STM32F437xx

目录

表格索引

图片索引

1 文档约定

在本文档中，将具有 FPU 的 Cortex-M4 内核称为 Cortex-M4F。

1.1 寄存器相关缩写词列表

寄存器说明中使用以下缩写词：

读/写 (rw)	软件可以读写这些位。
只读 (r)	软件只能读取这些位。
只写 (w)	软件只能写入该位。读取该位时将返回复位值。
读取/清零 (rc_w1)	软件可以读取该位，也可以通过写入 1 将该位清零。写入“0”对该位的值无影响。
读取/清零 (rc_w0)	软件可以读取该位，也可以通过写入 0 将该位清零。写入“1”对该位的值无影响。
读取/读取清零 (rc_r)	软件可以读取该位。读取该位时，将自动清零。写入“0”对该位的值无影响。
读取/置位 (rs)	软件可以读取该位，也可将其置 1。写入“0”对该位的值无影响。
只读写触发 (rt_w)	软件可以读取该位。写入“0”或“1”时，将触发事件，但不会影响该位的值。
切换 (t)	软件只能通过写入“1”来切换该位。写入“0”无影响。
保留 (Res.)	保留位，必须保持复位值。

1.2 词汇表

本节简要介绍本文档中所用首字母缩略词和缩写词的定义：

- 在本文档中，将具有 FPU 的 Cortex-M4 内核称为 Cortex-M4F
- CPU 内核集成了两个调试端口：
  - JTAG 调试端口 (JTAG-DP) 提供基于联合测试工作组 (JTAG) 协议的 5 引脚标准接口。
  - SWD 调试端口 (SWD-DP) 提供基于串行线调试 (SWD) 协议的 2 引脚（时钟和数据）接口。

  有关 JTAG 和 SWD 协议的信息，请参见《Cortex-M4F 技术参考手册》。
- 字：32 位数据/指令。
- 半字：16 位数据/指令。
- 字节：8 位数据。
- 双字：64 位数据。
- IAP（在应用中编程）：IAP 是指可以在用户程序运行期间对微控制器的 Flash 进行重新编程。
- ICP（在线编程）：ICP 是指可以在器件安装于用户应用电路板上时使用 JTAG 协议、SWD 协议或自举程序对微控制器的 Flash 进行编程。
- I-Code：此总线用于将 CPU 内核的指令总线连接到 Flash 指令接口。通过此总线可执行预取操作。
- D-Code：此总线用于将 CPU 的 D-Code 总线（数据加载和调试访问）连接到 Flash 数据接口。
- 选项字节：存储于 Flash 中的产品配置位。
- OBL：选项字节加载器。
- AHB：高级高性能总线。
- CPU：指 Cortex-M4F 内核。

1.3 外设可用性

STM32F405xx/07xx 和 STM32F415xx/17xx 系列中各型号产品的外设可用性及数量信息，请参见 STM32F405xx/07xx 和 STM32F415xx/17xx 数据手册。

STM32F42xxx 和 STM32F43xxx 系列中各型号产品的外设可用性及数量信息，请参见 STM32F42xxx 和 STM32F43xxx 数据手册。

2 存储器和总线架构

2.1 系统架构

主系统由 32 位多层 AHB 总线矩阵构成，可实现以下部分的互连：

- 八条主控总线：
  - Cortex™-M4F 内核 I 总线、D 总线和 S 总线
  - DMA1 存储器总线
  - DMA2 存储器总线
  - DMA2 外设总线
  - 以太网 DMA 总线
  - USB OTG HS DMA 总线
- 七条被控总线：
  - 内部 Flash ICode 总线
  - 内部 Flash DCode 总线
  - 主要内部 SRAM1 (112 KB)
  - 辅助内部 SRAM2 (16 KB)
  - 辅助内部 SRAM3 (64 KB)（仅适用于 STM32F42xxx 和 STM32F43xxx 器件）
  - AHB1 外设（包括 AHB-APB 总线桥和 APB 外设）
  - AHB2 外设
  - FSMC

借助总线矩阵，可以实现主控总线到被控总线的访问，这样即使在多个高速外设同时运行期间，系统也可以实现并发访问和高效运行。此架构如*图 1* 和*图 2* 所示。

注意： *64 KB CCM（内核耦合存储器）数据 RAM 不属于总线矩阵（请参见图 1：STM32F405xx/07xx 和 STM32F415xx/17xx 器件的系统架构和图 2：STM32F42xxx 和 STM32F43xxx 器件的系统架构）。只能通过 CPU 对其进行访问。*

图 1. STM32F405xx/07xx 和 STM32F415xx/17xx 器件的系统架构

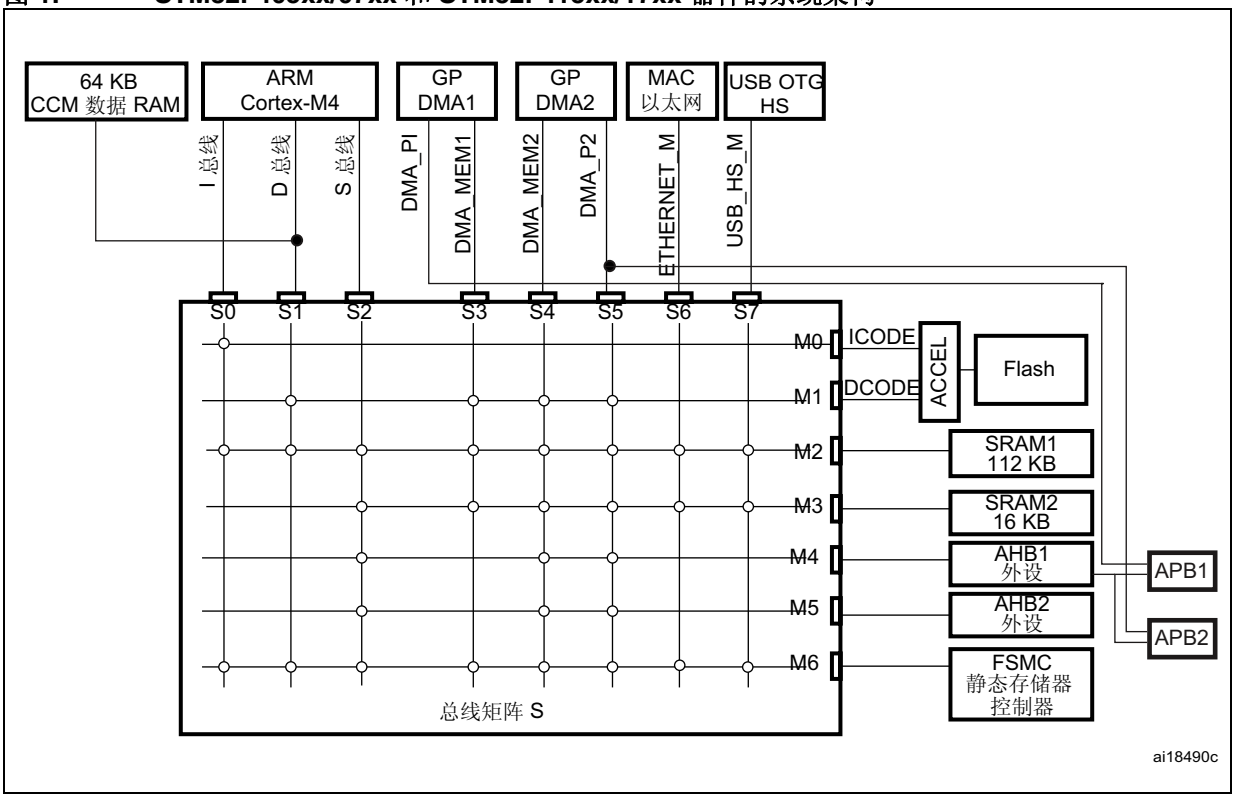
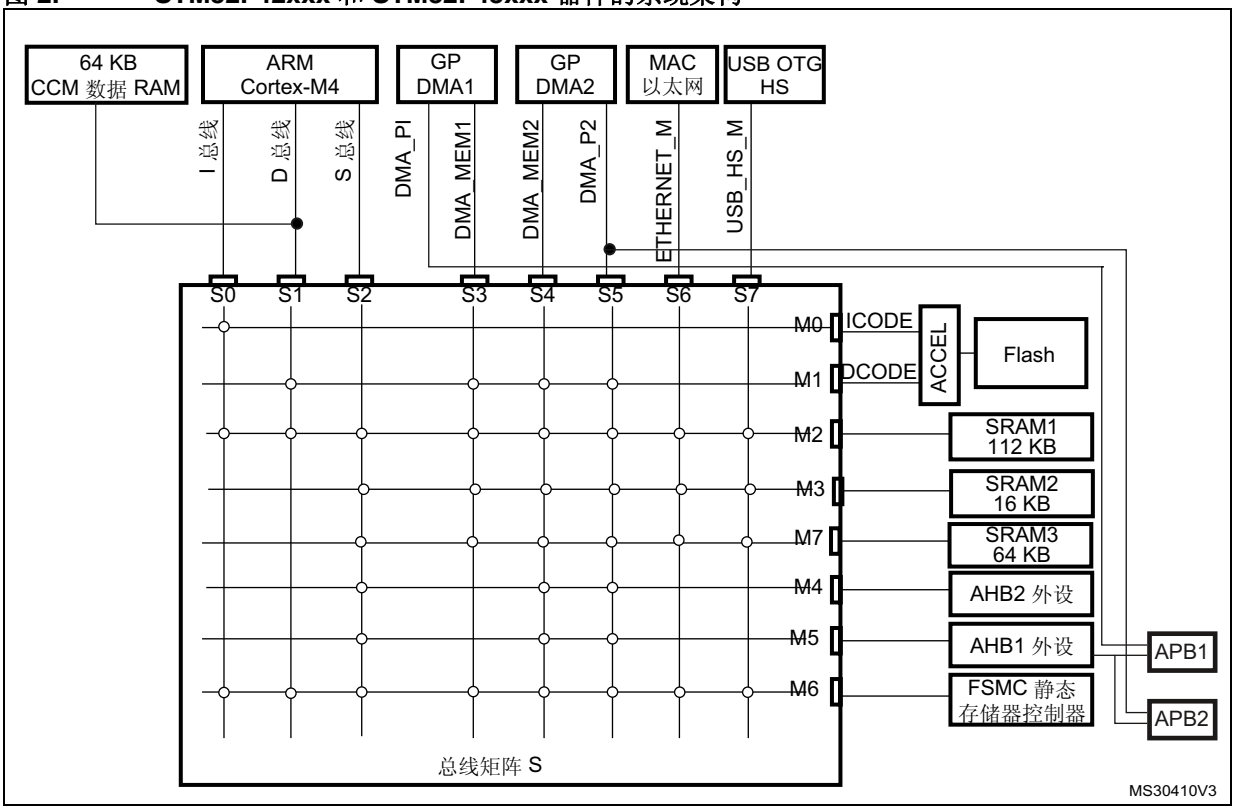


图 2. STM32F42xxx 和 STM32F43xxx 器件的系统架构



2.1.1 S0：I 总线

此总线用于将 Cortex™-M4F 内核的指令总线连接到总线矩阵。内核通过此总线获取指令。此总线访问的对象是包含代码的存储器（内部 Flash/SRAM 或通过 FSMC 的外部存储器）。

2.1.2 S1：D 总线

此总线用于将 Cortex™-M4F 数据总线和 64 KB CCM 数据 RAM 连接到总线矩阵。内核通过此总线进行立即数加载和调试访问。此总线访问的对象是包含代码或数据的存储器（内部 Flash 或通过 FSMC 的外部存储器）。

2.1.3 S2：S 总线

此总线用于将 Cortex™-M4F 内核的系统总线连接到总线矩阵。此总线用于访问位于外设或 SRAM 中的数据。也可通过此总线获取指令（效率低于 ICode）。此总线访问的对象是 112 KB、64 KB 和 16 KB 的内部 SRAM、包括 APB 外设在内的 AHB1 外设、AHB2 外设以及通过 FSMC 的外部存储器。

2.1.4 S3、S4：DMA 存储器总线

此总线用于将 DMA 存储器总线主接口连接到总线矩阵。DMA 通过此总线来执行存储器数据的传入和传出。此总线访问的对象是数据存储器：内部 SRAM（112 KB、64 KB、16 KB）以及通过 FSMC 的外部存储器。

2.1.5 S5：DMA 外设总线

此总线用于将 DMA 外设主总线接口连接到总线矩阵。DMA 通过此总线访问 AHB 外设或执行存储器间的数据传输。此总线访问的对象是 AHB 和 APB 外设以及数据存储器：内部 SRAM 以及通过 FSMC 的外部存储器。

2.1.6 S6：以太网 DMA 总线

此总线用于将以太网 DMA 主接口连接到总线矩阵。以太网 DMA 通过此总线向存储器存取数据。此总线访问的对象是数据存储器：内部 SRAM（112 KB、64 KB 和 16 KB）以及通过 FSMC 的外部存储器。

2.1.7 S7：USB OTG HS DMA 总线

此总线用于将 USB OTG HS DMA 主接口连接到总线矩阵。USB OTG DMA 通过此总线向存储器加载/存储数据。此总线访问的对象是数据存储器：内部 SRAM（112 KB、64 KB 和 16 KB）以及通过 FSMC 的外部存储器。

2.1.8 总线矩阵

总线矩阵用于主控总线之间的访问仲裁管理。仲裁采用循环调度算法。

2.1.9 AHB/APB 总线桥 (APB)

借助两个 AHB/APB 总线桥 APB1 和 APB2，可在 AHB 总线与两个 APB 总线之间实现完全同步的连接，从而灵活选择外设频率。

有关 APB1 和 APB2 最大频率的详细信息，请参见器件数据手册；有关 AHB 和 APB 外设地址映射的信息，请参见*第 52 页的表 2*。

每次芯片复位后，所有外设时钟都被关闭（SRAM 和 Flash 接口除外）。使用外设前，必须在 RCC_AHBxENR 或 RCC_APBxENR 寄存器中使能其时钟。

注意： *对 APB 寄存器执行 16 位或 8 位访问时，该访问将转换为 32 位访问：总线桥将 16 位或 8 位数据复制后提供给 32 位向量。*

2.2 存储器组织结构

程序存储器、数据存储器、寄存器和 I/O 端口排列在同一个顺序的 4 GB 地址空间内。

各字节按小端格式在存储器中编码。字中编号最低的字节被视为该字的最低有效字节，而编号最高的字节被视为最高有效字节。

有关外设寄存器映射的详细信息，请参见相关章节。

可寻址的存储空间分为 8 个主要块，每个块为 512 MB。

未分配给片上存储器和外设的所有存储区域均视为“保留区”。请参见产品数据手册中的存储器映射图。

2.3 存储器映射

有关综合存储器映射图，请参见相应器件的数据手册。*表 2* 提供了所有 STM32F4xx 器件中可用外设的边界地址。

表 2. STM32F4xx 寄存器边界地址

边界地址	外设	总线	寄存器映射
0xA000 0000 - 0xA000 0FFF	FSMC 控制寄存器	AHB3	*第 1241 页的第 32.6.9 节：FSMC 寄存器映射*
0x5006 0800 - 0x5006 0BFF	RNG	AHB2	*第 549 页的第 21.4.4 节：RNG 寄存器映射*
0x5006 0400 - 0x5006 07FF	HASH		*第 569 页的第 22.4.9 节：散列寄存器映射*
0x5006 0000 - 0x5006 03FF	CRYP		*第 543 页的第 20.6.13 节：CRYP 寄存器映射*
0x5005 0000 - 0x5005 03FF	DCMI		*第 327 页的第 13.8.12 节：DCMI 寄存器映射*
0x5000 0000 - 0x5003 FFFF	USB OTG FS		*第 1006 页的第 30.16.6 节：OTG_FS 寄存器映射*

表 2. STM32F4xx 寄存器边界地址（续）

边界地址	外设	总线	寄存器映射
0x4004 0000 - 0x4007 FFFF	USB OTG HS	AHB1	*第 1130 页的第 31.12.6 节：OTG_HS 寄存器映射*
0x4002 9000 - 0x4002 93FF	以太网 MAC		*第 924 页的第 29.8.5 节：以太网寄存器映射*
0x4002 8C00 - 0x4002 8FFF			
0x4002 8800 - 0x4002 8BFF			
0x4002 8400 - 0x4002 87FF			
0x4002 8000 - 0x4002 83FF			
0x4002 6400 - 0x4002 67FF	DMA2		*第 229 页的第 9.5.11 节：DMA 寄存器映射*
0x4002 6000 - 0x4002 63FF	DMA1		
0x4002 4000 - 0x4002 4FFF	BKPSRAM		
0x4002 3C00 - 0x4002 3FFF	Flash 接口寄存器		*第 3.8 节：Flash 接口寄存器*
0x4002 3800 - 0x4002 3BFF	RCC		*第 171 页的第 6.3.32 节：RCC 寄存器映射*
0x4002 3000 - 0x4002 33FF	CRC		*第 85 页的第 4.4.4 节：CRC 寄存器映射*
0x4002 2000 - 0x4002 23FF	GPIOI		*第 192 页的第 7.4.11 节：GPIO 寄存器映射*
0x4002 1C00 - 0x4002 1FFF	GPIOH		
0x4002 1800 - 0x4002 1BFF	GPIOG		
0x4002 1400 - 0x4002 17FF	GPIOF		
0x4002 1000 - 0x4002 13FF	GPIOE		
0x4002 0C00 - 0x4002 0FFF	GPIOD		
0x4002 0800 - 0x4002 0BFF	GPIOC		
0x4002 0400 - 0x4002 07FF	GPIOB		
0x4002 0000 - 0x4002 03FF	GPIOA		
0x4001 5400 - 0x4001 57FF	SPI6	APB2	*第 769 页的第 27.5.10 节：SPI 寄存器映射*
0x4001 5000 - 0x4001 53FF	SPI5		
0x4001 4800 - 0x4001 4BFF	TIM11	APB2	*第 481 页的第 16.6.11 节：TIM10/11/13/14 寄存器映射*
0x4001 4400 - 0x4001 47FF	TIM10		
0x4001 4000 - 0x4001 43FF	TIM9		*第 472 页的第 16.5.14 节：TIM9/12 寄存器映射*
0x4001 3C00 - 0x4001 3FFF	EXTI		*第 247 页的第 10.3.7 节：EXTI 寄存器映射*
0x4001 3800 - 0x4001 3BFF	SYSCFG		*第 199 页的第 8.2.9 节：SYSCFG 寄存器映射*
0x4001 3400 - 0x4001 37FF	SPI4	APB2	*第 769 页的第 27.5.10 节：SPI 寄存器映射*
0x4001 3000 - 0x4001 33FF	SPI1	APB2	*第 769 页的第 27.5.10 节：SPI 寄存器映射*
0x4001 2C00 - 0x4001 2FFF	SDIO		*第 819 页的第 28.9.16 节：SDIO 寄存器映射*
0x4001 2000 - 0x4001 23FF	ADC1 - ADC2 - ADC3		*第 286 页的第 11.13.18 节：ADC 寄存器映射*
0x4001 1400 - 0x4001 17FF	USART6		*第 720 页的第 26.6.8 节：USART 寄存器映射*
0x4001 1000 - 0x4001 13FF	USART1		
0x4001 0400 - 0x4001 07FF	TIM8		*第 390 页的第 14.4.21 节：TIM1 和 TIM8 寄存器映射*
0x4001 0000 - 0x4001 03FF	TIM1		

表 2. STM32F4xx 寄存器边界地址（续）

边界地址	外设	总线	寄存器映射
0x4000 7C00 - 0x4000 7FFF	UART8	APB1	第 720 页的第 26.6.8 节：USART 寄存器映射
0x4000 7800 - 0x4000 7BFF	UART7		
0x4000 7400 - 0x4000 77FF	DAC	APB1	第 306 页的第 12.5.15 节：DAC 寄存器映射
0x4000 7000 - 0x4000 73FF	PWR		第 104 页的第 5.5 节：PWR 寄存器映射
0x4000 6800 - 0x4000 6BFF	CAN2		第 644 页的第 24.9.5 节：bxCAN 寄存器映射
0x4000 6400 - 0x4000 67FF	CAN1		
0x4000 5C00 - 0x4000 5FFF	I2C3		第 675 页的第 25.6.11 节：I2C 寄存器映射
0x4000 5800 - 0x4000 5BFF	I2C2		
0x4000 5400 - 0x4000 57FF	I2C1		
0x4000 5000 - 0x4000 53FF	UART5		第 720 页的第 26.6.8 节：USART 寄存器映射
0x4000 4C00 - 0x4000 4FFF	UART4		
0x4000 4800 - 0x4000 4BFF	USART3		
0x4000 4400 - 0x4000 47FF	USART2		
0x4000 4000 - 0x4000 43FF	I2S3ext		第 769 页的第 27.5.10 节：SPI 寄存器映射
0x4000 3C00 - 0x4000 3FFF	SPI3 / I2S3		
0x4000 3800 - 0x4000 3BFF	SPI2 / I2S2		
0x4000 3400 - 0x4000 37FF	I2S2ext		
0x4000 3000 - 0x4000 33FF	IWDG		第 498 页的第 18.4.5 节：IWDG 寄存器映射
0x4000 2C00 - 0x4000 2FFF	WWDG		第 504 页的第 19.6.4 节：WWDG 寄存器映射
0x4000 2800 - 0x4000 2BFF	RTC & BKP 寄存器		第 605 页的第 23.6.21 节：RTC 寄存器映射
0x4000 2000 - 0x4000 23FF	TIM14		第 481 页的第 16.6.11 节：TIM10/11/13/14 寄存器映射
0x4000 1C00 - 0x4000 1FFF	TIM13		
0x4000 1800 - 0x4000 1BFF	TIM12		第 472 页的第 16.5.14 节：TIM9/12 寄存器映射
0x4000 1400 - 0x4000 17FF	TIM7		第 493 页的第 17.4.9 节：TIM6 和 TIM7 寄存器映射
0x4000 1000 - 0x4000 13FF	TIM6		
0x4000 0C00 - 0x4000 0FFF	TIM5		第 443 页的第 15.4.21 节：TIMx 寄存器映射
0x4000 0800 - 0x4000 0BFF	TIM4		
0x4000 0400 - 0x4000 07FF	TIM3		
0x4000 0000 - 0x4000 03FF	TIM2		

2.3.1 嵌入式 SRAM

STM32F405xx/07xx 和 STM32F415xx/17xx 带有 4 KB 备份 SRAM（请参见*第 5.1.2 节：电池备份域*）和 192 KB 系统 SRAM。

STM32F42xxx 和 STM32F43xxx 带有 4 KB 备份 SRAM（请参见*第 5.1.2 节：电池备份域*）和 256 KB 系统 SRAM。

系统 SRAM 可按字节、半字（16 位）或全字（32 位）访问。读写操作以 CPU 速度执行，且等待周期为 0。系统 SRAM 分为三个块：

- 映射在地址 0x2000 0000 的 112 KB 和 16 KB 块，可供所有 AHB 主控总线访问。
- 映射在地址 0x2002 0000 的 64 KB 块，可供所有 AHB 主控总线访问。（适用于 STM32F42xxx 和 STM32F43xxx）。AHB 主总线支持并发 SRAM 访问（通过以太网或 USB OTG HS）：例如，当 CPU 对 112 KB 或 64 KB SRAM 进行读/写操作时，以太网 MAC 可以同时对 16 KB SRAM 进行读/写操作。
- 在地址 0x1000 0000 映射的 64 KB 块，只能供 CPU 通过数据总线访问。

如果选择从 SRAM 自举或选择物理重映射（SYSCFG 控制器中的*第 8.2.1 节：SYSCFG 存储器重映射寄存器 (SYSCFG_MEMRMP)*），则 CPU 可通过系统总线或 I-Code/D-Code 总线访问系统 SRAM。要在 SRAM 执行期间获得最佳的性能，应选择物理重映射（通过自举管脚及软件配置来选择）。

2.3.2 Flash 概述

Flash 接口可管理 CPU 通过 AHB I-Code 和 D-Code 对 Flash 进行的访问。该接口可针对 Flash 执行擦除和编程操作，并实施读写保护机制。Flash 接口通过指令预取和缓存机制加速代码执行。

Flash 结构如下：

- 主存储器块分为多个扇区。
- 系统存储器，器件在系统存储器自举模式下从该存储器启动
- 512 OTP（一次性可编程）字节，用于存储用户数据。
- 选项字节，用于配置读写保护、BOR 级别、软件/硬件看门狗以及器件处于待机或停止模式下的复位。

更多详细信息，请参见*第 3 节：嵌入式 Flash 接口*。

2.3.3 位段

Cortex™-M4F 存储器映射包括两个位段区域。这些区域将存储器别名区域中的每个字映射到存储器位段区域中的相应位。在别名区域写入字时，相当于对位段区域的目标位执行读-修改-写操作。

在 STM32F4xx 器件中，外设寄存器和 SRAM 均映射到一个位段区域，这样可实现单个位段的读写操作。这些操作仅适用于 Cortex™-M4F 访问，对于其它总线主接口（如 DMA）无效。

可通过一个映射公式说明别名区域中的每个字与位段区域中各个位之间的对应关系。映射公式为：

$$bit\_word\_addr = bit\_band\_base + (byte\_offset \times 32) + (bit\_number \times 4)$$

其中：

- *bit_word_addr* 代表别名区域中将映射到目标位的字的地址
- *bit_band_base* 代表别名区域的起始地址
- *byte_offset* 代表目标位所在位段区域中的字节编号
- *bit_number* 代表目标位的位位置 (0-7)

示例

下例说明如何将 SRAM 地址 0x20000300 处字节的位 2 映射到别名区域：

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

对地址 0x22006008 执行写操作相当于在 SRAM 地址 0x20000300 处字节的位 2 执行读-修改-写操作。

对地址 0x22006008 执行读操作将返回 SRAM 地址 0x20000300 处字节的位 2 的值（0x01 表示位置位，0x00 表示位复位）。

有关位段的详细信息，请参见 *Cortex™-M4F 编程手册*（请参见*第 1 页的相关文档*）。

2.4 自举配置

存储器采用固定的存储器映射，代码区域起始地址为 0x0000 0000（通过 ICode/DCode 总线访问），而数据区域起始地址为 0x2000 0000（通过系统总线访问）。Cortex™-M4F CPU 始终通过 ICode 总线获取复位向量，这意味着只有代码区域（通常为 Flash）可以提供自举空间。STM32F4xx 微控制器实施一种特殊机制，可以从其它存储器（如内部 SRAM）进行自举。

在 STM32F4xx 中，可通过 BOOT[1:0] 引脚选择三种不同的自举模式，如*表 3* 所示。

表 3. 自举模式

自举模式选择引脚		自举模式	自举空间
BOOT1	**BOOT0**		
x	0	主 Flash	选择主 Flash 作为自举空间
0	1	系统存储器	选择系统存储器作为自举空间
1	1	嵌入式 SRAM	选择嵌入式 SRAM 作为自举空间

复位后，在 SYSCLK 的第四个上升沿锁存 BOOT 引脚的值。复位后，用户可以通过设置 BOOT1 和 BOOT0 引脚来选择需要的自举模式。

BOOT0 为专用引脚，而 BOOT1 则与 GPIO 引脚共用。一旦完成对 BOOT1 的采样，相应 GPIO 引脚即进入空闲状态，可用于其它用途。

器件退出待机模式时，还会对 BOOT 引脚重新采样。因此，当器件处于待机模式时，这些引脚必须保持所需的自举模式配置。这样的启动延迟结束后，CPU 将从地址 0x0000 0000 获取栈顶值，然后从始于 0x0000 0004 的自举存储器开始执行代码。

注意： *如果器件从 SRAM 自举，在应用程序初始化代码中，需要使用 NVIC 异常及中断向量表和偏移寄存器来重新分配 SRAM 中的向量表。*

嵌入式自举程序

嵌入式自举程序模式用于通过以下串行接口重新编程 Flash：

- USART1(PA9/PA10)
- USART3（PB10/11 和 PC10/11）
- CAN2(PB5/13)
- USB OTG FS(PA11/12) 从设备模式（DFU：器件固件升级）。

USART 外设以内部 16 MHz 振荡器 (HSI) 频率运行，而 CAN 和 USB OTG FS 则需要相当于 1 MHz 数倍（4 MHz 到 26 MHz 之间）的外部时钟 (HSE) 频率。

嵌入式自举程序代码位于系统存储器中，在芯片生产期间由 ST 编程。有关详细信息，请参见应用笔记 AN2606。

物理重映射

选择自举引脚后，应用程序软件可以将某些存储器设定为从代码空间进行访问（这样，可通过 ICode 总线而非系统总线执行代码）。这样的修改通过在 SYSCFG 控制器中编程*第 8.2.1 节：SYSCFG 存储器重映射寄存器 (SYSCFG_MEMRMP)* 来实现。

因此可重映射以下存储器：

- 主 Flash
- 系统存储器
- 嵌入式 SRAM1 (112 KB)
- FSMC 块 1（NOR/PSRAM 1 和 2）

表 4. 存储器映射与自举模式/物理重映射

地址	主 Flash 中的自举/重映射	嵌入式 SRAM 中的自举/重映射	系统存储器中的自举/重映射	FSMC 中的重映射
0x2002 0000 - 0x2002 FFFF(1)	SRAM3 (64 KB)	SRAM3 (64 KB)	SRAM3 (64 KB)	SRAM3 (64 KB)
0x2001 C000 - 0x2001 FFFF	SRAM2 (16 KB)	SRAM2 (16 KB)	SRAM2 (16 KB)	SRAM2 (16 KB)
0x2000 0000 - 0x2001 BFFF	SRAM1 (112 KB)	SRAM1 (112 KB)	SRAM1 (112 KB)	SRAM1 (112 KB)
0x1FFF 0000 - 0x1FFF 77FF	系统存储器	系统存储器	系统存储器	系统存储器
0x0810 0000 - 0x0FFF FFFF	保留	保留	保留	保留
0x0800 0000 - 0x080F FFFF	Flash	Flash	Flash	Flash
0x0400 0000 - 0x07FF FFFF	保留	保留	保留	FSMC 块 1 NOR/PSRAM 2（使用别名）
0x0000 0000 - 0x03FF FFFF(2)(3)	Flash (1 MB) 使用别名	SRAM1 (112 KB) 使用别名	系统存储器 (30 KB) 使用别名	FSMC 块 1 NOR/PSRAM 1（使用别名）

1. SRAM3 仅适用于 STM32F42xxx 和 STM32F43xxx。
2. 在地址 0x0000 0000 重映射 FSMC 时，只能重映射块 1 存储器控制器的前两个区域（块 1 NOR/PSRAM 1 和 NOR/PSRAM 2）。在重映射模式下，CPU 可以通过 ICode 总线（而不是 System 总线）访问外部存储器来提高性能。
3. 即使在自举存储空间中使用别名，相关存储器仍可通过其原始存储空间进行访问。

3 嵌入式 Flash 接口

3.1 前言

Flash 接口可管理 CPU 通过 AHB I-Code 和 D-Code 对 Flash 进行的访问。该接口可针对 Flash 执行擦除和编程操作，并实施读写保护机制。

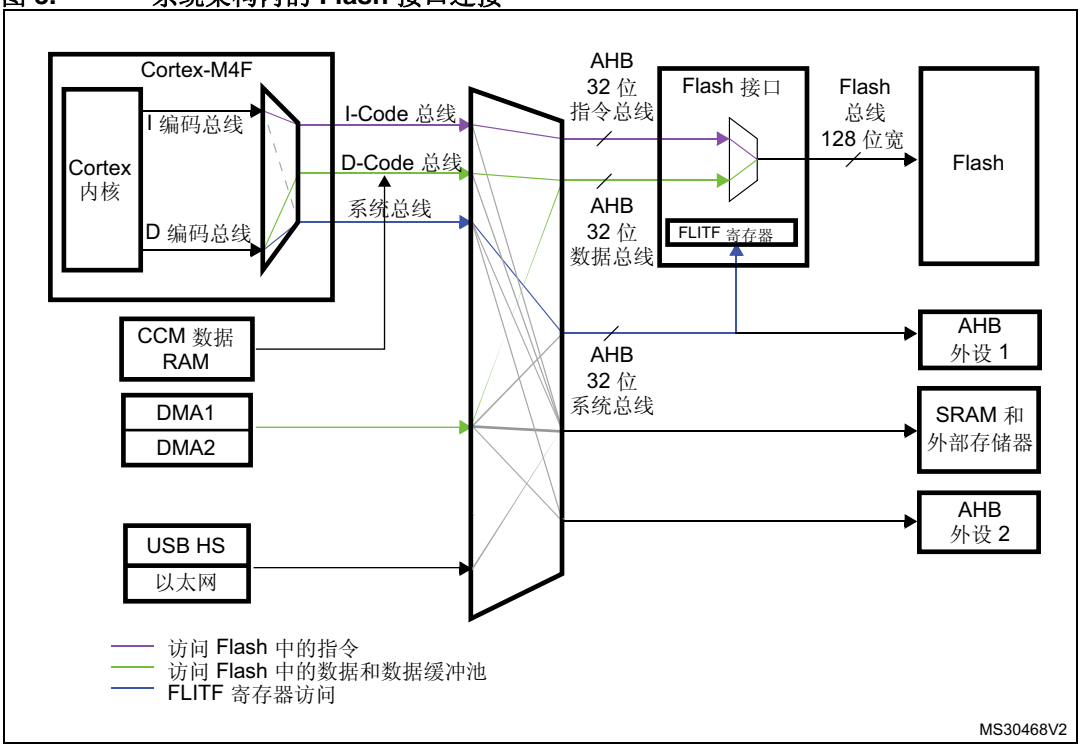
Flash 接口通过指令预取和缓存机制加速代码执行。

3.2 主要特性

- Flash 读操作
- Flash 编程/擦除操作
- 读/写保护
- I-Code 上的预取操作
- I-Code 上的 64 个缓存（128 位宽）
- D-Code 上的 8 个缓存（128 位宽）

图 3 所示为系统架构内的 Flash 接口连接。

图 3. 系统架构内的 Flash 接口连接



3.3 嵌入式 Flash

Flash 具有以下主要特性：

- 对于 STM32F40x 和 STM32F41x，容量高达 1 MB；对于 STM32F42x 和 STM32F43x，容量高达 2 MB
- 128 位宽数据读取
- 字节、半字、字和双字数据写入
- 扇区擦除与全部擦除
- 存储器组织结构

  Flash 结构如下：
  - 主存储器块，分为 4 个 16 KB 扇区、1 个 64 KB 扇区和 7 个 128 KB 扇区
  - 系统存储器，器件在系统存储器自举模式下从该存储器启动
  - 512 字节 OTP（一次性可编程），用于存储用户数据

    OTP 区域还有 16 个额外字节，用于锁定对应的 OTP 数据块。
  - 选项字节，用于配置读写保护、BOR 级别、软件/硬件看门狗以及器件处于待机或停止模式下的复位。
- 低功耗模式（有关详细信息，请参见参考手册的“电源控制 (PWR)”部分）

表 5. Flash 模块构成（STM32F40x 和 STM32F41x）

块	名称	块基址	大小
主存储器	扇区 0	0x0800 0000 - 0x0800 3FFF	16 KB
	扇区 1	0x0800 4000 - 0x0800 7FFF	16 KB
	扇区 2	0x0800 8000 - 0x0800 BFFF	16 KB
	扇区 3	0x0800 C000 - 0x0800 FFFF	16 KB
	扇区 4	0x0801 0000 - 0x0801 FFFF	64 KB
	扇区 5	0x0802 0000 - 0x0803 FFFF	128 KB
	扇区 6	0x0804 0000 - 0x0805 FFFF	128 KB
	. . .	. . .	. . .
	扇区 11	0x080E 0000 - 0x080F FFFF	128 KB
系统存储器		0x1FFF 0000 - 0x1FFF 77FF	30 KB
OTP 区域		0x1FFF 7800 - 0x1FFF 7A0F	528 字节
选项字节		0x1FFF C000 - 0x1FFF C00F	16 字节

表 6. Flash 构成（STM32F42x 和 STM32F43x）

块	名称	块基址	大小
主存储器	扇区 0	0x0800 0000 - 0x0800 3FFF	16 KB
	扇区 1	0x0800 4000 - 0x0800 7FFF	16 KB
	扇区 2	0x0800 8000 - 0x0800 BFFF	16 KB
	扇区 3	0x0800 C000 - 0x0800 FFFF	16 KB
	扇区 4	0x0801 0000 - 0x0801 FFFF	64 KB
	扇区 5	0x0802 0000 - 0x0803 FFFF	128 KB
	扇区 6	0x0804 0000 - 0x0805 FFFF	128 KB
	-	-	-
	-	-	-
	-	-	-
	扇区 11	0x080E 0000 - 0x080F FFFF	128 KB
	扇区 12	0x0810 0000 - 0x0810 3FFF	16 KB
	扇区 13	0x0810 4000 - 0x0810 7FFF	16 KB
	扇区 14	0x0810 8000 - 0x0810 BFFF	16 KB
	扇区 15	0x0810 C000 - 0x0810 FFFF	16 KB
	扇区 16	0x0811 0000 - 0x0811 FFFF	64 KB
	扇区 17	0x0812 0000 - 0x0813 FFFF	128 KB
	扇区 18	0x0814 0000 - 0x0815 FFFF	128 KB
		-	-
		-	-
		-	-
	扇区 23	0x081E 0000 - 0x081F FFFF	128 KB
系统存储器		0x1FFF 0000 - 0x1FFF 77FF	30 KB
OTP		0x1FFF 7800 - 0x1FFF 7A0F	528 字节
选项字节		0x1FFF C000 - 0x1FFF C00F	16 字节
		0x1FFE C000 - 0x1FFE C007	16 字节

3.4 读接口

3.4.1 CPU 时钟频率与 Flash 读取时间之间的关系

为了准确读取 Flash 数据，必须根据 CPU 时钟 (HCLK) 频率和器件电源电压在 Flash 存取控制寄存器 (FLASH_ACR) 中正确地编程等待周期数 (LATENCY)。

当电源电压低于 2.1 V 时，必须关闭预取缓冲器。*表 7* 所示为 Flash 等待周期与 CPU 时钟频率之间的对应关系。

注意： *STM32F405xx/07xx 和 STM32F415xx/17xx 器件：*

- 当 VOS = “0” 时，f_{HCLK} 最大值 = 144 MHz。

- 当 VOS = “1” 时，f_{HCLK} 最大值 = 168 MHz。

STM32F42xxx 和 STM32F43xxx 器件：

- 当 VOS[1:0] = “0x01” 时，f_{HCLK} 最大值为 120 MHz。

- 当 VOS[1:0] = “0x10” 时，f_{HCLK} 最大值为 144 MHz。

- 当 VOS[1:0] = “0x11” 时，f_{HCLK} 最大值为 168 MHz。

表 7. CPU 时钟 (HCLK) 频率对应的等待周期数

等待周期 (WS) (LATENCY)	HCLK (MHz)			
	电压范围 2.7 V - 3.6 V	电压范围 2.4 V - 2.7 V	电压范围 2.1 V - 2.4 V	电压范围 1.8 V - 2.1 V 预取关闭
0 WS （1 个 CPU 周期）	0 < HCLK ≤ 30	0 < HCLK ≤ 24	0 < HCLK ≤ 22	0 < HCLK ≤ 20
1 WS （2 个 CPU 周期）	30 < HCLK ≤ 60	24 < HCLK ≤ 48	22 < HCLK ≤ 44	20 < HCLK ≤ 40
2 WS （3 个 CPU 周期）	60 < HCLK ≤ 90	48 < HCLK ≤ 72	44 < HCLK ≤ 66	40 < HCLK ≤ 60
3 WS （4 个 CPU 周期）	90 < HCLK ≤ 120	72 < HCLK ≤ 96	66 < HCLK ≤ 88	60 < HCLK ≤ 80
4 WS （5 个 CPU 周期）	120 < HCLK ≤ 150	96 < HCLK ≤ 120	88 < HCLK ≤ 110	80 < HCLK ≤ 100
5 WS （6 个 CPU 周期）	150 < HCLK ≤ 168	120 < HCLK ≤ 144	110 < HCLK ≤ 132	100 < HCLK ≤ 120
6 WS （7 个 CPU 周期）		144 < HCLK ≤ 168	132 < HCLK ≤ 154	120 < HCLK ≤ 140
7 WS （8 个 CPU 周期）			154 < HCLK ≤ 168	140 < HCLK ≤ 160

复位后，CPU 时钟频率为 16 MHz，FLASH_ACR 寄存器中的等待周期 (WS) 为 0。

强烈建议通过以下软件序列来根据 CPU 频率调整在访问 Flash 时所需的等待周期数。

需要提高 CPU 频率时的操作步骤

1. 将新的等待周期数编程到 FLASH_ACR 寄存器中的 LATENCY 位
2. 通过读取 FLASH_ACR 寄存器，检查新的等待周期是否设置成功
3. 通过改写 RCC_CFGR 寄存器中的 SW 位来修改 CPU 时钟源
4. 如有需要，可通过改写 RCC_CFGR 中的 HPRE 位来修改 CPU 时钟预分频器
5. 通过读取 RCC_CFGR 寄存器中相应的时钟源状态（SWS 位）和/或 AHB 预分频值（HPRE 位），检查新的 CPU 时钟源和/或新的 CPU 时钟预分频值是否设置成功

需要降低 CPU 频率时的操作步骤

1. 通过改写 RCC_CFGR 寄存器中的 SW 位来修改 CPU 时钟源
2. 如有需要，可通过改写 RCC_CFGR 中的 HPRE 位来修改 CPU 时钟预分频器
3. 通过读取 RCC_CFGR 寄存器中相应的时钟源状态（SWS 位）和/或 AHB 预分频值（HPRE 位），检查新的 CPU 时钟源和/或新的 CPU 时钟预分频值是否设置成功
4. 将新的等待周期数编程到 FLASH_ACR 中的 LATENCY 位
5. 通过读取 FLASH_ACR 寄存器，检查新的等待周期是否设置成功

注意： *CPU 时钟配置或等待周期 (WS) 配置的更改不会立即生效。为了确保当前的 CPU 时钟频率即为所配置的频率，用户可检查 AHB 预分频系数和时钟源状态值。为了确保所编程的 WS 数生效，可读取 FLASH_ACR 寄存器的内容来确认。*

3.4.2 自适应实时存储器加速器 (ART Accelerator™)

专有的自适应实时 (ART) 存储器加速器面向 STM32 工业标准 ARM® Cortex™-M4F 处理器进行了优化。该加速器很好地体现了 ARM Cortex M4F 的固有性能优势，克服了通常条件下，高速的处理器在运行中需要经常等待 FLASH 读取的情况。

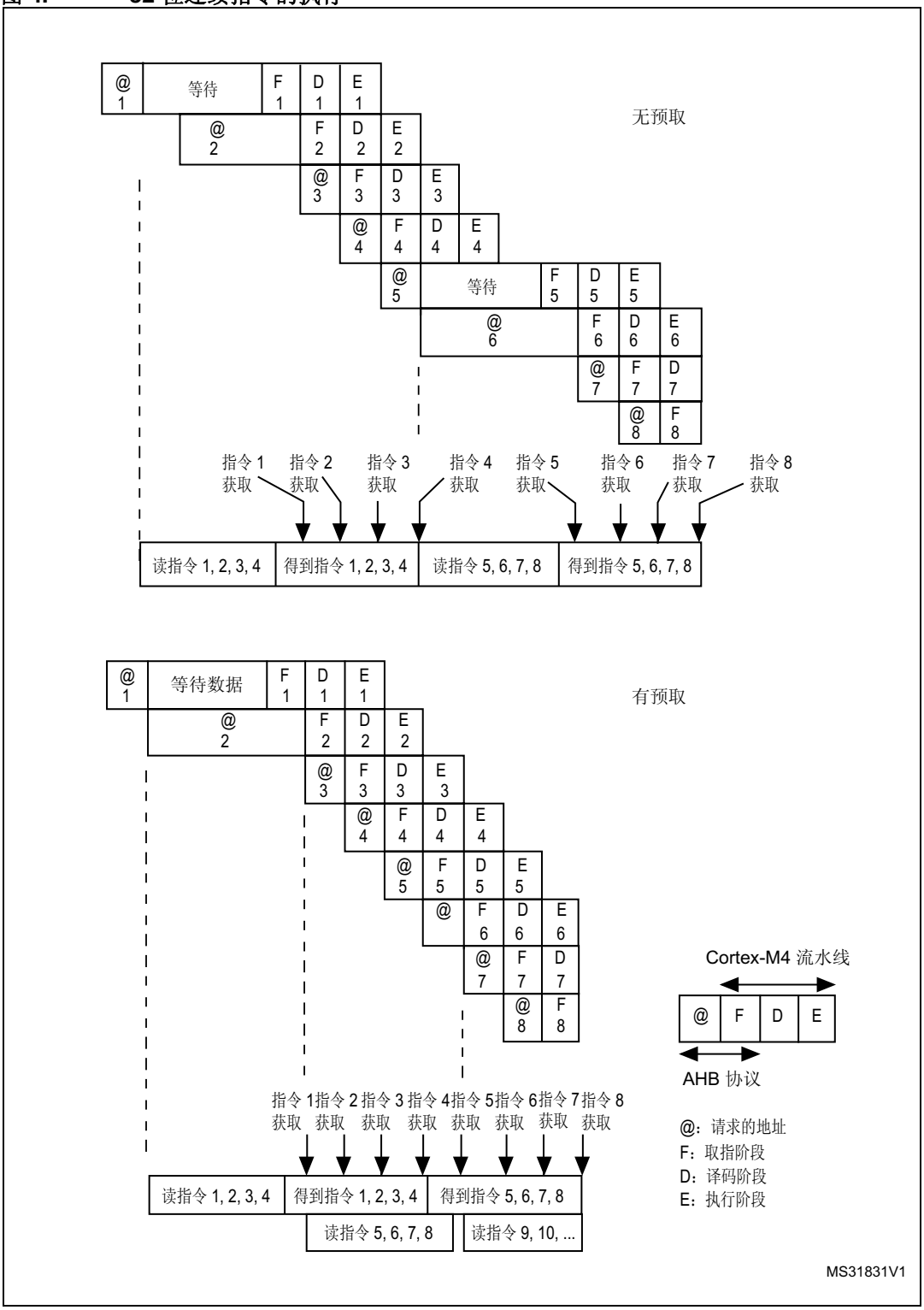
为了发挥处理器的全部性能，该加速器将实施指令预取队列和分支缓存，从而提高了 128 位 Flash 的程序执行速度。根据 CoreMark 基准测试，凭借 ART 加速器所获得的性能相当于 Flash 在 CPU 频率高达 168 MHz 时以 0 个等待周期执行程序。

指令预取

每个 Flash 读操作可读取 128 位，可以是 4 行 32 位指令，也可以是 8 行 16 位指令，具体取决于烧写在 Flash 中的程序。因此对于顺序执行的代码，至少需要 4 个 CPU 周期来执行前一次读取的 128 位指令行。在 CPU 请求当前指令行时，可使用 I-Code 总线的预取操作读取 Flash 中的下一个连续存放的 128 位指令行。可将 FLASH_ACR 寄存器中的 PRFTEN 位置 1，来使能预取功能。当访问 Flash 至少需要一个等待周期时，此功能非常有用。

图 4 所示为需要 3 WS（3 个等待周期）访问 Flash 时连续 32 位指令的执行过程，图中分别介绍了使用和不使用预取操作两种情况。

图 4.　　32 位连续指令的执行



处理非顺序执行的代码（有分支）时，指令可能并不存在于当前使用的或预取的指令行中。这种情况下，CPU 等待时间至少等于等待周期数。

指令缓存存储器

为了减少因指令跳转而产生的时间损耗，可将 64 行 128 位的指令保存到指令缓存存储器中。可将 FLASH_ACR 寄存器中的指令缓存使能 (ICEN) 位置 1，来使能这一特性。每当出现指令缺失（即请求的指令未存在于当前使用的指令行、预取指令行或指令缓存存储器中）时，系统会将新读取的行复制到指令缓存存储器中。如果 CPU 请求的指令已存在于指令缓存区中，则无需任何延时即可立即获取。指令缓存存储器存满后，可采用 LRU（最近最少使用）策略确定指令缓存存储器中待替换的指令行。此特性非常适用于包含循环的代码。

数据管理

在 CPU 流水线执行阶段，将通过 D-Code 总线访问 Flash 中的数据缓冲池。因此，直到提供了请求的数据后，CPU 流水线才会继续执行。为了减少因此而产生的时间损耗，通过 AHB 数据总线 D-Code 进行的访问优先于通过 AHB 指令总线 I-Code 进行的访问。

如果频繁使用某些数据，可将 FLASH_ACR 寄存器中的数据缓存使能 (DCEN) 位置 1，来使能数据缓存存储器。此特性的工作原理与指令缓存存储器类似，但保留的数据大小限制在 8 行 128 位/行以内。

注意： *用户配置扇区中的数据无法缓存。*

3.5 擦除和编程操作

执行任何 Flash 编程操作（擦除或编程）时，CPU 时钟频率 (HCLK) 不能低于 1 MHz。如果在 Flash 操作期间发生器件复位，无法保证 Flash 中的内容。

在对 STM32F4xx 的 Flash 执行写入或擦除操作期间，任何读取 Flash 的尝试都会导致总线阻塞。只有在完成编程操作后，才能正确处理读操作。这意味着，写/擦除操作进行期间不能从 Flash 中执行代码或数据获取操作。

3.5.1 Flash 控制寄存器解锁

复位后，Flash 控制寄存器 (FLASH_CR) 不允许执行写操作，以防因电气干扰等原因出现对 Flash 的意外操作。此寄存器的解锁顺序如下：

1. 在 Flash 密钥寄存器 (FLASH_KEYR) 中写入 KEY1 = 0x45670123
2. 在 Flash 密钥寄存器 (FLASH_KEYR) 中写入 KEY2 = 0xCDEF89AB

如果顺序出现错误，将返回总线错误并锁定 FLASH_CR 寄存器，直到下一次复位。

也可通过软件将 FLASH_CR 寄存器中的 LOCK 位置为 1 来锁定 FLASH_CR 寄存器。

注意： *当 FLASH_SR 寄存器中的 BSY 位为 1 时，将不能在写模式下访问 FLASH_CR 寄存器。BSY 位为 1 时，对该寄存器的任何写操作尝试都会导致 AHB 总线阻塞，直到 BSY 位清零。*

3.5.2 编程/擦除并行位数

通过 FLASH_CR 寄存器中的 PSIZE 字段配置并行位数。并行位数表示每次对 Flash 进行写操作时将编程的字节数。PSIZE 受限于电源电压以及是否使用外部 V_{PP} 电源。因此，在进行任何编程/擦除操作前，必须在 FLASH_CR 寄存器中对其进行正确配置。

Flash 擦除操作只能针对扇区或整个 Flash（批量擦除）执行。擦除时间取决于 PSIZE 编程值。有关擦除时间的详细信息，请参见器件数据手册的电气特性部分。

表 8 提供了正确的 PSIZE 值。

表 8. 编程/擦除并行位数

	电压范围 2.7 - 3.6 V（使用外部 V_{PP}）	电压范围 2.7 - 3.6 V	电压范围 2.4 - 2.7 V	电压范围 2.1 - 2.4 V	电压范围 1.8 V - 2.1 V
并行位数	x64	x32	x16		x8
PSIZE(1:0)	11	10	01		00

注意： *如果在编程并行位数/电压范围设置不一致的情况下启动任何编程或擦除操作，可能会导致出现意外结果。即使后续的读操作指示逻辑值已有效写入存储器中，也无法确定写入操作确实成功。*

要使用 V_{PP}，必须在 V_{PP} 引脚施加一个外部高压电源（8 V 到 9 V 之间）。该外部电源必须在直流电耗超过 10 mA 时也能维持该电压范围。建议仅在工厂生产线上进行初始编程时使用 VPP。V_{PP} 电源的供电时间不得超过一小时，否则 Flash 可能会损坏。

3.5.3 擦除

Flash 擦除操作可针对扇区或整个 Flash（批量擦除）执行。执行批量擦除时，不会影响 OTP 扇区或配置扇区。

扇区擦除

扇区擦除的具体步骤如下：

1. 检查 FLASH_SR 寄存器中的 BSY 位，以确认当前未执行任何 Flash 操作
2. 在 FLASH_CR 寄存器中，将 SER 位置 1，并从主存储块的 12 个 (STM32F405xx/07xx 和 STM32F415xx/17xx) 或 24 个 (STM32F42xxx 和 STM32F43xxx) 扇区中选择要擦除的扇区 (SNB)
3. 将 FLASH_CR 寄存器中的 STRT 位置 1
4. 等待 BSY 位清零

批量擦除

要执行批量擦除，建议采用以下步骤：

1. 检查 FLASH_SR 寄存器中的 BSY 位，以确认当前未执行任何 Flash 操作
2. 将 FLASH_CR 寄存器中的 MER 位置 1(STM32F405xx/07xx 和 STM32F415xx/17xx 器件）
3. 将 FLASH_CR 寄存器中的 MER 和 MER1 位置 1（STM32F42xxx 和 STM32F43xxx 器件）
4. 将 FLASH_CR 寄存器中的 STRT 位置 1
5. 等待 BSY 位清零

注意： *如果 FLASH_CR 寄存器中的 MERx 位和 SER 位均置为 1，则无法执行扇区擦除和批量擦除。*

3.5.4 编程

标准编程

Flash 编程顺序如下：

1. 检查 FLASH_SR 中的 BSY 位，以确认当前未执行任何主要 Flash 操作。
2. 将 FLASH_CR 寄存器中的 PG 位置 1。
3. 针对所需存储器地址（主存储器块或 OTP 区域内）执行数据写入操作：
   - 并行位数为 x8 时按字节写入
   - 并行位数为 x16 时按半字写入
   - 并行位数为 x32 时按字写入
   - 并行位数为 x64 时按双字写入
4. 等待 BSY 位清零。

注意： *把 Flash 的单元从“1”写为“0”时，无需执行擦除操作即可进行连续写操作。把 Flash 的单元从“0”写为“1”时，则需要执行 Flash 擦除操作。*

如果同时发出擦除和编程操作请求，首先执行擦除操作。

编程错误

不允许针对 Flash 执行跨越 128 位行界限的数据编程操作。如果出现这种情况，写操作将不会执行，并且 FLASH_SR 寄存器中的编程对齐错误标志位 (PGAERR) 将置 1。

写访问宽度（字节、半字、字或双字）必须与所选并行位数类型（x8、x16、x32 或 x64）相符。否则，写操作将不会执行，并且 FLASH_SR 寄存器中的编程并行位数错误标志位 (PGPERR) 将置 1。

如果未遵循标准的编程顺序（例如，在 PG 位未置 1 时尝试向 Flash 地址写入数据），则操作将中止并且 FLASH_SR 寄存器中的编程顺序错误标志位 (PGSERR) 将置 1。

编程与缓存

如果 Flash 写访问涉及数据缓存中的某些数据，Flash 写访问将修改 Flash 中的数据和缓存中的数据。

如果 Flash 中的擦除操作也涉及数据或指令缓存中的数据，则必须确保在代码执行期间访问这些数据之前将它们重新写入缓存。如果无法可靠执行这一操作，建议将 FLASH_CR 寄存器中的 DCRST 和 ICRST 位置 1，以刷新缓存。

注意： *I/D 缓存只有在被禁止 (I/DCEN = 0) 的情况下才能刷新。*

3.5.5 中断

如果将 FLASH_CR 寄存器中的操作结束中断使能位 (EOPIE) 置 1，则在擦除或编程操作结束时，即 FLASH_SR 寄存器中的繁忙位 (BSY) 清零（操作正确或非正确完成）时，将产生中断。此时，FLASH_SR 寄存器中的操作结束 (EOP) 位置 1。

如果在请求编程、擦除或读操作期间出现错误，则 FLASH_SR 寄存器中的以下错误标志位之一将置 1：

- PGAERR、PGPERR、PGSERR（编程错误标志）
- WRPERR（保护错误标志）

这种情况下，FLASH_SR 寄存器中的操作错误位 (OPERR) 置 1，并且如果 FLASH_SR 寄存器中的错误中断使能位 (ERRIE) 置 1，则将产生一个中断。

注意： *如果检测到多个连续错误（例如，在对 Flash 进行 DMA 传输期间），则直到连续写操作请求结束，这些错误标志才会清零。*

表 9. Flash 中断请求

中断事件	事件标志	使能控制位
操作结束	EOP	EOPIE
写保护错误	WRPERR	ERRIE
编程错误	PGAERR、PGPERR、PGSERR	ERRIE

3.6 选项字节

3.6.1 关于用户选项字节的说明

选项字节由最终用户根据具体的应用要求进行配置。*表 10* 介绍了这些字节在用户配置扇区内的构成。

表 10. 选项字节构成

地址	[63:16]	[15:0]
0x1FFF C000	保留	ROP 和用户选项字节 (**RDP** & **USER**)
0x1FFF C008	保留	扇区 0 到 11 的写保护 **nWRP** 位
0x1FFE C000	保留	保留
0x1FFE C008	保留	扇区 12 到 23 的写保护 **nWRP** 位

表 11. 关于选项字节的说明 (STM32F405xx/07xx 和 STM32F415xx/17xx)

选项字节（字，地址 0x1FFF C000）	
RDP：*读保护选项字节。* 读保护用于保护 Flash 中存储的软件代码。	
位 15:8	0xAA：级别 0，无保护 0xCC：级别 2，芯片保护（禁止调试和从 RAM 启动） 其它值：级别 1，存储器读保护（调试功能受限）
USER：用户选项字节 此字节用于配置以下功能： – 选择看门狗事件：硬件或软件 – 进入停止模式时产生复位事件 – 进入待机模式时产生复位事件	
位 7	**nRST_STDBY** 0：进入待机模式时产生复位 1：不产生复位

表 11. 关于选项字节的说明 (STM32F405xx/07xx 和 STM32F415xx/17xx)（续）

位 6	**nRST_STOP** 0：进入停止模式时产生复位 1：不产生复位
位 5	**WDG_SW** 0：硬件独立看门狗 1：软件独立看门狗
位 4	0x1：未使用
位 3:2	**BOR_LEV**：BOR 复位级别 这些位包含释放复位信号所需达到的供电电压阈值。通过对这些位执行写操作，可将新的 BOR 级别值编程到 Flash。 00：BOR 级别 3 (VBOR3)，复位阈值电压为 2.70 V 到 3.60 V 01：BOR 级别 2 (VBOR2)，复位阈值电压为 2.40 V 到 2.70 V 10：BOR 级别 1 (VBOR1)，复位阈值电压为 2.10 V 到 2.40 V 11：BOR 关闭 (VBOR0)，复位阈值电压为 1.8 V 到 2.10 V
位 1:0	0x1：未使用
选项字节（字，地址 0x1FFF C008）	
位 15:12	0xF：未使用
nWRP：*Flash 写保护选项字节* 扇区 0 到 11 可采用写保护。	
位 11:0	**nWRPi** 0：开启所选扇区的写保护 1：关闭所选扇区的写保护

表 12. 关于选项字节的说明 (STM32F42xxx 和 STM32F43xxx)

选项字节（字，地址 0x1FFF C000）	
RDP：*读保护选项字节。* 读保护用于保护 Flash 中存储的软件代码。	
位 15:8	0xAA：级别 0，无保护 0xCC：级别 2，芯片保护（禁止调试和从 RAM 启动） 其它值：级别 1，存储器读保护（调试功能受限）
USER：用户选项字节 此字节用于配置以下功能： 选择看门狗事件：硬件或软件 进入停止模式时产生复位事件 进入待机模式时产生复位事件	
位 7	**nRST_STDBY** 0：进入待机模式时产生复位 1：不产生复位
位 6	**nRST_STOP** 0：进入停止模式时产生复位 1：不产生复位

表 12. 关于选项字节的说明 (STM32F42xxx 和 STM32F43xxx)（续）

位 5	**WDG_SW** 0：硬件看门狗 1：软件看门狗
位 4	0x1：未使用
位 3:2	**BOR_LEV**：BOR 复位级别 这些位包含释放复位信号所需达到的供电电压阈值。 通过对这些位执行写操作，可将新的 BOR 级别值编程到 Flash。 00：BOR 级别 3 (VBOR3)，复位阈值电压为 2.70 V 到 3.60 V 01：BOR 级别 2 (VBOR2)，复位阈值电压为 2.40 V 到 2.70 V 10：BOR 级别 1 (VBOR1)，复位阈值电压为 2.10 V 到 2.40 V 11：BOR 关闭 (VBOR0)，复位阈值电压为 1.8 V 到 2.10 V
位 1:0	0x1：未使用
选项字节（字，地址 0x1FFF C008）	
位 15:12	0xF：未使用
nWRP：Flash 写保护选项字节。 扇区 0 到 11 可采用写保护。	
位 11:0	**nWRPi**： 0：开启所选扇区的写保护 1：关闭所选扇区的写保护
选项字节（字，地址 0x1FFE C000）	
位 15:0	0xFF：未使用
选项字节（字，地址 0x1FFE C008）	
位 15:12	0xF：未使用
nWRP：Flash 写保护选项字节。 扇区 12 到 23 可采用写保护。	
位 11:0	**nWRPi**： 0：开启扇区 i 的写保护。 1：关闭扇区 i 的写保护。

3.6.2 用户选项字节编程

要针对此扇区执行任何操作，Flash 选项控制寄存器 (FLASH_OPTCR) 中的选项锁定位 (OPTLOCK) 必须清零。为了能够将该位清零，用户需要顺序执行以下步骤：

1. 在 Flash 选项密钥寄存器 (FLASH_OPTKEYR) 中写入 OPTKEY1 = 0x0819 2A3B
2. 在 Flash 选项密钥寄存器 (FLASH_OPTKEYR) 中写入 OPTKEY2 = 0x4C5D 6E7F

通过软件将 OPTLOCK 位置 1 后，可防止用户选项字节发生意外的擦除/编程操作。

修改 STM32F405xx/07xx 和 STM32F415xx/17xx 上的用户选项字节

要修改用户选项值，请顺序执行以下步骤：

1. 检查 FLASH_SR 寄存器中的 BSY 位，以确认当前未执行任何 Flash 操作
2. 在 FLASH_OPTCR 寄存器中写入所需的选项值。
3. 将 FLASH_OPTCR 寄存器中的选项启动位 (OPTSTRT) 置 1。
4. 等待 BSY 位清零。

注意： *硬件会自动先擦除用户配置扇区，然后以 FLASH_OPTCR 寄存器中包含的值对所有选项字节进行编程。*

修改 STM32F42xxx 和 STM32F43xxx 上的用户选项字节

要修改用户选项字节值，请顺序执行以下步骤：

1. 检查 FLASH_SR 寄存器中的 BSY 位，以确认当前未执行任何 Flash 操作。
2. 在 FLASH_OPTCR 和/或 FLASH_OPTCR1 寄存器中写入选项字节值。
3. 将 FLASH_OPTCR 寄存器中的选项启动位 (OPTSTRT) 置 1。
4. 等待 BSY 位清零。

注意： *硬件会自动先擦除用户配置扇区，然后以 FLASH_OPTCR 和 FLASH_OPTCR1 寄存器中包含的值对所有选项字节进行编程。*

3.6.3 读保护 (RDP)

可对 Flash 中的用户区域实施读保护，以防不受信任的代码读取其中的数据。读保护分三个级别，具体定义如下：

- **级别 0：无读保护**

  将 0xAA 写入读保护选项字节 (RDP) 时，读保护级别即设为 0。此时，在所有自举配置（用户 Flash 自举、调试或从 RAM 自举）中，均可执行对 Flash 或备份 SRAM 的读/写操作（如果未设置写保护）。

- **级别 1：使能读保护**

  这是擦除选项字节后的默认读保护级别。将任意值（分别用于设置级别 0 和级别 2 的 0xAA 和 0xCC 除外）写入 RDP 选项字节时，即激活读保护级别 1。设置读保护级别 1 后：

  – 在连接调试功能或从 RAM 或系统存储器自举时，不能对 Flash 或备份 SRAM 进行访问（读取、擦除、编程）。读请求将导致总线错误。

  – 从 Flash 自举时，允许通过用户代码对 Flash 和备份 SRAM 进行访问（读取、擦除、编程）。

  激活级别 1 后，如果将保护选项字节 (RDP) 编程为级别 0，则将对 Flash 和备份 SRAM 执行全部擦除。因此，在取消读保护之前，用户代码区域会清零。批量擦除操作仅擦除用户代码区域。包括写保护在内的其它选项字节将不受影响。OTP 区域不受批量擦除操作的影响，同样保持不变。只有在已激活级别 1 并请求级别 0 时，才会执行批量擦除。当提高保护级别 (0->1、1->2、0->2) 时，不会执行批量擦除。

- **级别 2：禁止调试/芯片读保护**

  将 0xCC 写入 RDP 选项字节时，可激活读保护级别 2。设置读保护级别 2 后：

  – 级别 1 提供的所有保护均有效。

  – 不再允许从 RAM 或系统存储器自举。

  – JTAG、SWV（单线查看器）、ETM 和边界扫描处于禁止状态。

  – 用户选项字节不能再进行更改。

  – 从 Flash 自举时，允许通过用户代码对 Flash 和备份 SRAM 进行访问（读取、擦除、编程）。

  存储器读保护级别 2 是不可更改的。激活级别 2 后，保护级别不能再降回级别 0 或级别 1。

注意： *激活级别 2 后，将永久性禁止 JTAG 端口（相当于 JTAG 熔断）。这样，将无法执行边界扫描。意法半导体无法对设为保护级别 2 的器件做失效分析。*

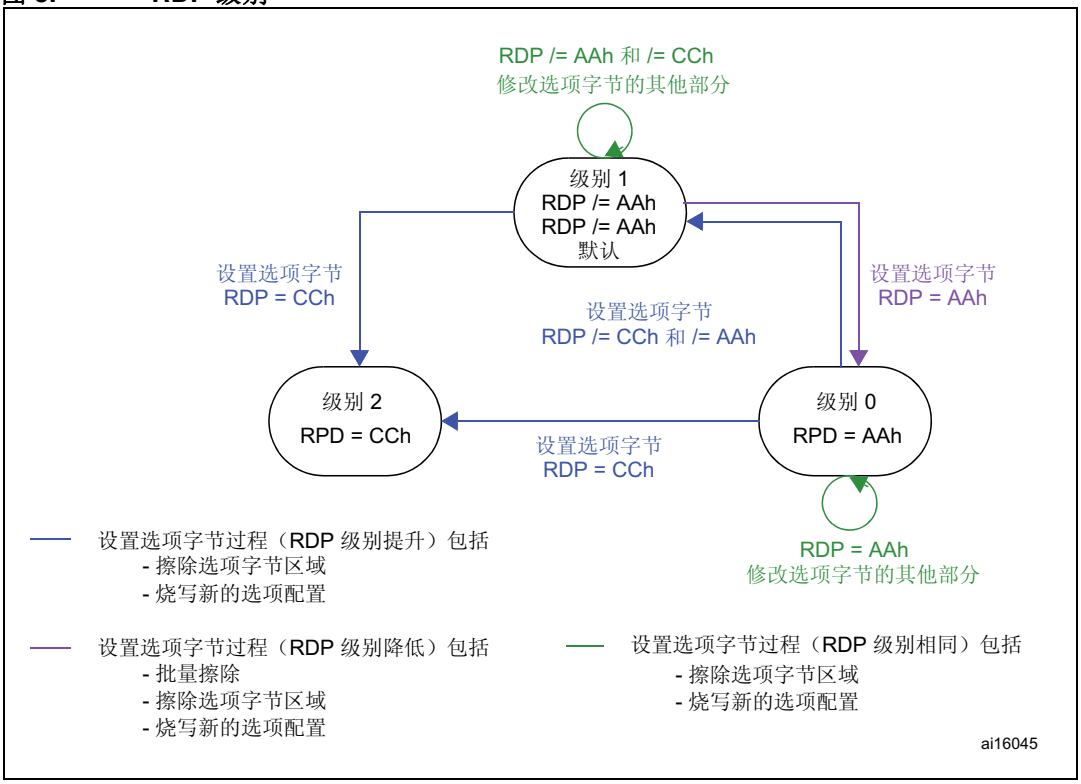
表 13. 不同读保护级别下的访问限制

存储区	保护级别	调试功能，从 RAM 或系统存储器自举			从 Flash 自举		
		读	写	擦除	读	写	擦除
主 Flash 和备份 SRAM	级别 1	否		否(1)	是		
	级别 2	否			是		
选项字节	级别 1	是			是		
	级别 2	否			否		
OTP	级别 1	否		NA	是		NA
	级别 2	否		NA	是		NA

1. 只有在 RDP 从级别 1 更改为级别 0 时，才会擦除主 Flash 和备份 SRAM。OTP 区域保持不变。

图 5 所示为 RDP 级别切换的过程。

图 5. RDP 级别



3.6.4 写保护

Flash 中有多达 24 个用户扇区具备写保护功能，可防止因程序指针错乱而发生意外的写操作。当 FLASH_OPTCR 或 FLASH_OPTCR1 寄存器中的低有效写保护位 nWRPi (0 ≤ i ≤ 11) 为低电平时，无法对相应的扇区执行擦除或编程操作。因此，如果某个扇区处于写保护状态，则无法对整个器件执行全部擦除。

如果尝试对 Flash 中处于写保护状态的区域执行擦除/编程操作（由写保护位保护的扇区、锁定的 OTP 区域或永远不能执行写操作的 Flash 区域，例如 ICP），则 FLASH_SR 寄存器中的写保护错误标志位 (WRPERR) 将置 1。

注意： *选择存储器读保护级别（RDP 级别 = 1）后，如果已连接 CPU 调试功能（JTAG 调试或单线调试）或者正在从 RAM 执行启动代码，则即使 nWRPi = 1，也无法对 Flash 扇区 i 执行编程或擦除操作。*

写保护错误标志

如果对 Flash 的写保护区域执行擦除/编程操作，则 FLASH_SR 寄存器中的写保护错误标志位 (WRPERR) 将置 1。

如果请求执行擦除操作，则以下情况下 WRPERR 位置 1：

- 配置全部擦除、块擦除、扇区擦除（MER 或 MER/MER1 和 SER = 1）
- 请求执行扇区擦除但扇区编号 SNB 字段无效
- 请求执行全部擦除，但至少一个用户扇区通过选项位实施了写保护（FLASH_OPTCRx 寄存器中的 MER 或 MER/MER1 = 1 且 nWRPi = 0，0 ≤ i ≤ 11 位）
- 请求针对写保护扇区执行扇区擦除。（FLASH_OPTCRx 寄存器中 SER = 1、SNB = i 且 nWRPi = 0，0 ≤ i ≤ 11 位）
- Flash 处于读保护状态，但检测到擦除操作企图。

如果请求执行编程操作，则以下情况下 WRPERR 位置 1：

- 针对系统存储器或用户特定扇区的保留区域执行写操作
- 针对用户配置扇区执行写操作
- 针对通过选项位实施写保护的扇区执行写操作
- 请求针对已锁定的 OTP 区域执行写操作
- Flash 处于读保护状态，但检测到写操作企图

3.7 一次性可编程字节

表 14 所示为 OTP 区域中一次性可编程 (OTP) 部分的构成。

表 14. OTP 区域构成

块	[128:96]	[95:64]	[63:32]	[31:0]	地址字节 0
0	OTP0	OTP0	OTP0	OTP0	0x1FFF 7800
	OTP0	OTP0	OTP0	OTP0	0x1FFF 7810
1	OTP1	OTP1	OTP1	OTP1	0x1FFF 7820
	OTP1	OTP1	OTP1	OTP1	0x1FFF 7830

表 14. OTP 区域构成（续）

块	[128:96]	[95:64]	[63:32]	[31:0]	地址字节 0
. . .		. . .			. . .
15	OTP15	OTP15	OTP15	OTP15	0x1FFF 79E0
	OTP15	OTP15	OTP15	OTP15	0x1FFF 79F0
锁定块	LOCKB15 ... LOCKB12	LOCKB11 ... LOCKB8	LOCKB7 ... LOCKB4	LOCKB3 ... LOCKB0	0x1FFF 7A00

OTP 区域划分为 16 个 32 字节的 OTP 数据块和 1 个 16 字节的 OTP 锁定块。OTP 数据块和锁定块均无法擦除。锁定块中包含 16 字节的 LOCKBi (0 ≤ i ≤ 15)，用于锁定相应的 OTP 数据块（块 0 到 15）。每个 OTP 数据块均可编程，除非相应的 OTP 锁定字节编程为 0x00。锁定字节的值只能是 0x00 和 0xFF，否则这些 OTP 字节无法正确使用。

3.8 Flash 接口寄存器

3.8.1 Flash 访问控制寄存器 (FLASH_ACR)

Flash access control register

Flash 访问控制寄存器用于使能/关闭加速功能，并且可根据 CPU 频率控制 Flash 访问时间。

偏移地址：0x00
复位值：0x0000 0000

访问：无等待周期，按字、半字和字节访问

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			DCRST	ICRST	DCEN	ICEN	PRFTEN	Reserved					LATENCY		
			rw	w	rw	rw	rw						rw	rw	rw

位 31:11 保留，必须保持清零。

位 12 **DCRST**：数据缓存复位 (Data cache reset)
0：数据缓存不复位
1：数据缓存复位
只有在关闭数据缓存时才能在该位中写入值。

位 11 **ICRST**：指令缓存复位 (Instruction cache reset)
0：指令缓存不复位
1：指令缓存复位
只有在关闭指令缓存时才能在该位中写入值。

位 10 **DCEN**：数据缓存使能 (Data cache enable)
0：关闭数据缓存
1：使能数据缓存

位 9 **ICEN**：指令缓存使能 (Instruction cache enable)
0：关闭指令缓存
1：使能指令缓存

位 8 **PRFTEN**：预取使能 (Prefetch enable)
0：关闭预取
1：使能预取

位 7:3 保留，必须保持清零。

位 2:0 **LATENCY**：延迟 (Latency)
这些位表示 CPU 时钟周期与 Flash 访问时间之比。
000：零等待周期
001：一个等待周期
010：两个等待周期
011：三个等待周期
100：四个等待周期
101：五个等待周期
110：六个等待周期
111：七个等待周期

3.8.2 Flash 密钥寄存器 (FLASH_KEYR)

Flash key register

借助 Flash 密钥寄存器，可允许对 Flash 控制寄存器的访问，进而允许执行编程和擦除操作。

偏移地址：0x04
复位值：0x0000 0000

访问：无等待周期，按字访问

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[31:16]															
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
KEY[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

位 31:0 **FKEYR**：FPEC 密钥 (FPEC key)
要将 FLASH_CR 寄存器解锁并允许对其执行编程/擦除操作，必须顺序编程以下值：
a) KEY1 = 0x45670123
b) KEY2 = 0xCDEF89AB

3.8.3 Flash 选项密钥寄存器 (FLASH_OPTKEYR)

Flash option key register

借助 Flash 选项密钥寄存器，可允许在用户配置扇区中执行编程和擦除操作。

偏移地址：0x08
复位值：0x0000 0000

访问：无等待周期，按字访问

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OPTKEYR[31:16]															
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
OPTKEYR[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

位 31:0 **OPTKEYR**：选项字节密钥 (Option byte key)

要将 FLASH_OPTCR 寄存器解锁并允许对其编程，必须顺序编程以下值：

a) OPTKEY1 = 0x08192A3B

b) OPTKEY2 = 0x4C5D6E7F

3.8.4 Flash 状态寄存器 (FLASH_SR)

Flash status register

Flash 状态寄存器提供正在执行的编程和擦除操作的相关信息。

偏移地址：0x0C
复位值：0x0000 0000

访问：无等待周期，按字、半字和字节访问

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															BSY
															r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								PGSERR	PGPERR	PGAERR	WRPERR	Reserved		OPERR	EOP
								rc_w1	rc_w1	rc_w1	rc_w1			rc_w1	rc_w1

位 31:17 保留，必须保持清零。

位 16 **BSY**：繁忙 (Busy)

该位指示 Flash 操作正在进行。该位在 Flash 操作开始时置 1，在操作结束或出现错误时清零。

0：当前未执行任何 Flash 操作
1：正在执行 Flash 操作

位 15:8 保留，必须保持清零。

位 7 **PGSERR**：编程顺序错误 (Programming sequence error)

如果代码在控制寄存器未正确配置的情况下对 Flash 执行写访问，将由硬件为该位置 1。
写入 1 即可将该位清零。

位 6 **PGPERR**：编程并行位数错误 (Programming parallelism error)

如果在编程期间数据访问类型（字节、半字、字和双字）与配置的并行位数 PSIZE (x8, x16, x32, x64) 不符，将由硬件为该位置 1。
写入 1 即可将该位清零。

位 5 **PGAERR**：编程对齐错误 (Programming alignment error)

如果要编程的数据不能包含在同一个 128 位 Flash 行中，将由硬件为该位置 1。
写入 1 即可将该位清零。

位 4 **WRPERR**：写保护错误 (Write protection error)

如果要擦除/编程的地址属于 Flash 中处于写保护状态的区域，将由硬件为该位置 1。
写入 1 即可将该位清零。

位 3:2 保留，必须保持清零。

位 1 **OPERR**：操作错误 (Operation error)

如果检测到 Flash 操作（编程/擦除/读取）请求，但由于存在并行位数错误、对齐错误或写保护错误而无法运行，将由硬件对该位置 1。只有在使能错误中断 (ERRIE = 1) 后，该位才会置 1。

位 0 **EOP**：操作结束 (End of operation)

当成功完成一个或多个 Flash 操作（编程/擦除）时，由硬件将该位置 1。只有在使能操作结束中断 (EOPIE = 1) 后，该位才会置 1。
写入 1 即可将该位清零。

3.8.5 用于 STM32F405xx/07xx 和 STM32F415xx/17xx 的 Flash 控制寄存器 (FLASH_CR)

Flash control register

Flash 控制寄存器用于配置和启动 Flash 操作。

偏移地址：0x10

复位值：0x8000 0000

访问：当前未执行任何 Flash 操作时无等待周期，按字、半字和字节访问。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LOCK	Reserved					ERRIE	EOPIE	Reserved							STRT
rs						rw	rw								rs

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						PSIZE[1:0]		Res.	SNB[3:0]				MER	SER	PG
						rw	rw		rw	rw	rw	rw	rw	rw	rw

位 31 **LOCK**：锁定 (Lock)

该位只能写入 1。该位置 1 时，表示 FLASH_CR 寄存器已锁定。当检测到解锁序列时，由硬件将该位清零。
如果解锁操作失败，该位仍保持置 1，直到下一次复位。

位 31:26 保留，必须保持清零。

位 25 **ERRIE**：错误中断使能 (Error interrupt enable)

当 FLASH_SR 寄存器中的 OPERR 位置 1 后，可通过该位使能中断产生功能。
0：禁止错误中断
1：使能错误中断

位 24 **EOPIE**：操作结束中断使能 (End of operation interrupt enable)

当 FLASH_SR 寄存器中的 EOP 位置 1 后，可通过该位使能中断产生功能。
0：禁止中断
1：使能中断

位 23:17 保留，必须保持清零。

位 16 **STRT**：启动 (Start)

该位置 1 后可触发擦除操作。该位只能通过软件置 1，并在 BSY 位清零后随之清零。

位 15:10 保留，必须保持清零。

位 9:8 **PSIZE**：编程大小 (Program size)
这些位用于选择编程并行位数。
00 x8 编程
01 x16 编程
10 x32 编程
11 x64 编程

位 6:3 **SNB**：扇区编号 (Sector number)
这些位用于选择要擦除的扇区。
0000 扇区 0
0001 扇区 1
...
1011 扇区 11
不允许使用其它值

位 2 **MER**：批量擦除 (Mass Erase)
针对所有用户扇区激活擦除操作。

位 1 **SER**：扇区擦除 (Sector Erase)
激活扇区擦除。

位 0 **PG**：编程 (Programming)
激活 Flash 编程。

3.8.6 用于 STM32F42xxx 和 STM32F43xxx 的 Flash 控制寄存器 (FLASH_CR)

Flash control register

Flash 控制寄存器用于配置和启动 Flash 操作。

偏移地址：0x10

复位值：0x8000 0000

访问：当前未执行任何 Flash 操作时无等待周期，按字、半字和字节访问。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LOCK	Reserved					ERRIE	EOPIE	Reserved							STRT
rs						rw	rw								rs

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MER1	Reserved					PSIZE[1:0]		SNB[4:0]					MER	SER	PG
rw						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31 **LOCK**：锁定 (Lock)

该位只能写入 1。该位置 1 时，表示 FLASH_CR 寄存器已锁定。当检测到解锁序列时，由硬件将该位清零。

如果解锁操作失败，该位仍保持置 1，直到下一次复位。

位 31:26 保留，必须保持清零。

位 25 **ERRIE**：错误中断使能 (Error interrupt enable)

当 FLASH_SR 寄存器中的 OPERR 位置 1 后，可通过该位使能中断产生功能。

0：禁止错误中断

1：使能错误中断

位 24 **EOPIE**：操作结束中断使能 (End of operation interrupt enable)

当 FLASH_SR 寄存器中的 EOP 位置 1 后，可通过该位使能中断产生功能。

0：禁止中断

1：使能中断

位 23:17 保留，必须保持清零。

位 16 **STRT**：启动 (Start)

该位置 1 后可触发擦除操作。该位只能通过软件置 1，并在 BSY 位清零后随之清零。

位 15 **MER1**：批量擦除 12 到 23 扇区 (Mass Erase of sectors 12 to 23)

激活 12 到 23 扇区的擦除操作。

位 14:10 保留，必须保持清零。

位 9:8 **PSIZE**：编程大小 (Program size)

这些位用于选择编程并行位数。

00 x8 编程

01 x16 编程

10 x32 编程

11 x64 编程

位 7:3 **SNB**：扇区编号 (Sector number)

这些位用于选择要擦除的扇区。

0000：扇区 0

0001：扇区 1

...

01011：扇区 11

01100：不允许

01101：不允许

01110：不允许

01111：不允许

10000：扇区 12

10001：扇区 13

...

11011 扇区 23

11100：不允许

11101：不允许

11110：不允许

11111：不允许

位 2 **MER**：擦除块 1 里的所有扇区 (Mass Erase of bank 1 sectors)
激活对块 1 里所有扇区的擦除功能

位 1 **SER**：扇区擦除 (Sector Erase)
激活扇区擦除。

位 0 **PG**：编程 (Programming)
激活 Flash 编程。

3.8.7 Flash 选项控制寄存器 (FLASH_OPTCR)

Flash option control register

FLASH_OPTCR 寄存器用于修改用户选项字节。

偏移地址：0x14

复位值：0x0FFF AAED。复位信号释放时，硬件将 Flash 中的值加载到这些选项位。

访问：当前未执行任何 Flash 操作时无等待周期，按字、半字和字节访问。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved				nWRP[11:0]											
				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
RDP[7:0]								nRST_STDBY	nRST_STOP	WDG_SW	Reserved	BOR_LEV		OPTSTRT	OPTLOCK
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rs	rs

位 31:28 保留，必须保持清零。

位 27:16 **nWRP**：无写保护 (Not write protect)
这些位包含复位后扇区写保护选项字节的值。通过对这些位执行写操作，可将新的写保护值编程到 Flash。
0：开启所选扇区的写保护
1：关闭所选扇区的写保护

位 15:8 **RDP**：读保护 (Read protect)
这些位包含复位后读保护选项字节的值。通过对这些位执行写操作，可将新的读保护值编程到 Flash。
0xAA：级别 0，未激活读保护
0xCC：级别 2，激活芯片读保护
其它值：级别 1，激活存储器读保护

位 7:5 **USER**：用户选项字节 (User option bytes)
这些位包含复位后用户选项字节的值。通过对这些位执行写操作，可将新的用户选项字节值编程到 Flash。
位 7：nRST_STDBY
位 6：nRST_STOP
位 5：WDG_SW

注意：当 WDG 模式从硬件切换到软件或从软件切换到硬件时，需要执行系统复位才能更改生效。

位 4 保留，必须保持清零。

位 3:2 **BOR_LEV**：BOR 复位级别 (BOR reset Level)

这些位包含释放复位信号所需达到的供电电压阈值。可通过对这些位执行写操作，来编程新的 BOR 级别。BOR 默认为关闭。当电源电压 (V_{DD}) 降至所选 BOR 级别以下时，将产生器件复位。

00：BOR 级别 3 (VBOR3)，复位阈值电压为 2.70 V 到 3.60 V

01：BOR 级别 2 (VBOR2)，复位阈值电压为 2.40 V 到 2.70 V

10：BOR 级别 1 (VBOR1)，复位阈值电压为 2.10 V 到 2.40 V

11：BOR 关闭 (VBOR0)，复位阈值电压为 1.80 V 到 2.10 V

注意：有关 BOR 特性的完整详情，请参见器件数据手册中的“电气特性”部分。

位 1 **OPTSTRT**：启动选项 (Option start)

该位置 1 后可触发用户选项操作。该位只能通过软件置 1，并在 BSY 位清零后随之清零。

位 0 **OPTLOCK**：锁定选项 (Option lock)

该位只能写入 1。该位置 1 时，表示 FLASH_OPTCR 寄存器已锁定。当检测到解锁序列时，由硬件将该位清零。

如果解锁操作失败，该位仍保持置 1，直到下一次复位。

3.8.8 用于 STM32F42xxx 和 STM32F43xxx 的 Flash 选项控制寄存器 (FLASH_OPTCR1)

Flash option control register

此寄存器仅适用于 STM32F42xxx 和 STM32F43xxx。

FLASH_OPTCR1 寄存器中包含的用户选项设置作用于扇区 12 到 23。

偏移地址：0x18

复位值：0x0FFF 0000。复位信号释放时，硬件将 Flash 中的值加载到这些选项位。

访问：当前未执行任何 Flash 操作时无等待周期，按字、半字和字节访问。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved				nWRP[11:0]											
				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
Reserved															

位 31:28 保留，必须保持清零。

位 27:16 **nWRP**：无写保护 (Not write protect)

这些位包含复位后扇区 12 到 23 的写保护选项字节值。通过对这些位执行写操作，可将新的写保护值编程到 Flash。

0：已激活写保护

1：未激活写保护

位 15:0 保留，必须保持清零。

3.8.9 Flash 接口寄存器映射

表 15. Flash 寄存器映射与复位值 (STM32F405xx/07xx 和 STM32F415xx/17xx)

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	FLASH_ACR	Reserved																			DCRST	ICRST	DCEN	ICEN	PRFTEN	Reserved						LATENCY		
	Reset value																				0	0	0	0	0							0	0	0
0x04	FLASH_KEYR	KEY[31:16]																KEY[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08	FLASH_OPTKEYR	OPTKEYR[31:16]																OPTKEYR[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	FLASH_SR	Reserved															BSY	Reserved								PGSERR	PGPERR	PGAERR	WRPERR	Reserved		OPERR	EOP
	Reset value																0									0	0	0	0			0	0
0x10	FLASH_CR	LOCK	Reserved							EOPIE	Reserved						STRT	Reserved						PSIZE[1:0]		Reserved	SNB[3:0]				MER	SER	PG
	Reset value	1								0							0							0	0		0	0	0	0		0	0
0x14	FLASH_OPTCR	Reserved	nWRP[11:0]															RDP[7:0]								nRST_STDBY	nRST_STOP	WDG_SW	Reserved	BOR_LEV		OPTSTRT	OPTLOCK
	Reset value		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	0	1	0	1	0	1	1	1		1	1	0	1

表 16. Flash 寄存器映射与复位值 (STM32F42xxx 和 STM32F43xxx)

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	FLASH_ACR	Reserved																			DCRST	ICRST	DCEN	ICEN	PRFTEN	Reserved						LATENCY		
	Reset value																				0	0	0	0	0							0	0	0
0x04	FLASH_KEYR	KEY[31:16]																KEY[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08	FLASH_OPTKEYR	OPTKEYR[31:16]																OPTKEYR[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	FLASH_SR	Reserved															BSY	Reserved								PGSERR	PGPERR	PGAERR	WRPERR	Reserved		OPERR	EOP
	Reset value																0									0	0	0	0			0	0
0x10	FLASH_CR	LOCK	Reserved							EOPIE	Reserved						STRT	MER1	Reserved					PSIZE[1:0]		SNB[4:0]					MER	SER	PG
	Reset value	1								0							0	0						0	0	0	0	0	0	0	0	0	0

表 16. Flash 寄存器映射与复位值 (STM32F42xxx 和 STM32F43xxx)（续）

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x14	FLASH_OPTCR	Reserved				nWRP[11:0]												RDP[7:0]								nRST_STDBY	nRST_STOP	WDG_SW	Reserved	BOR_LEV		OPTSTRT	OPTLOCK
	Reset value					1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	0	1	0	1	0	1	1	1		1	1	0	1
0x18	FLASH_ OPTCR1	Reserved				nWRP[11:0]												Reserved															
	Reset value					1	1	1	1	1	1	1	1	1	1	1	1																

4 CRC 计算单元

除非特别说明，否则本部分适用于整个 STM32F4xx 系列。

4.1 CRC 简介

CRC （循环冗余校验）计算单元使用一个固定的多项式发生器从一个 32 位的数据字中产生 CRC 码。

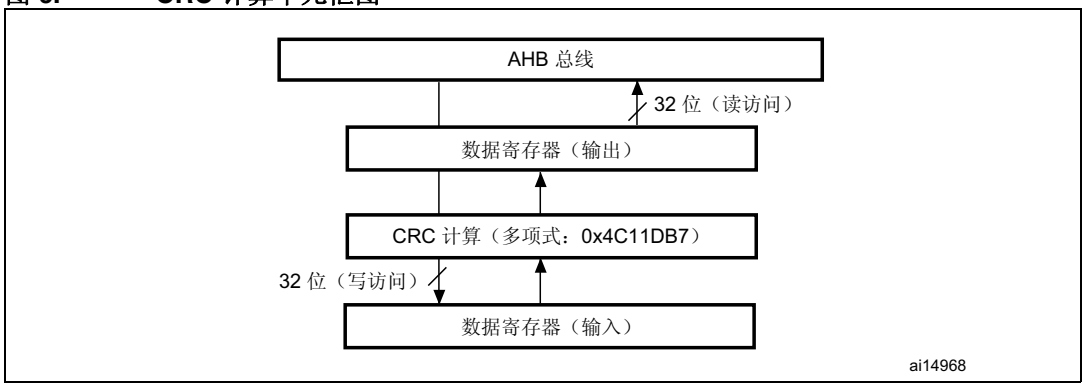
在众多的应用中，基于 CRC 的技术还常用来验证数据传输或存储的完整性。根据 EN/IEC 60335-1 标准的规定，这些技术提供了验证 Flash 完整性的方法。CRC 计算单元有助于在运行期间计算软件的签名，并将该签名与链接时生成并存储在指定存储单元的参考签名加以比较。

4.2 CRC 主要特性

- 使用 CRC-32 （以太网）多项式：0x4C11DB7
  – $X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + X^8 + X^7 + X^5 + X^4 + X^2 + X + 1$
- 单输入/输出 32 位数据寄存器
- CRC 计算在 4 个 AHB 时钟周期 (HCLK) 内完成
- 8 位通用寄存器 （可用于临时存储）

框图如*图 6* 所示。

图 6. CRC 计算单元框图



4.3 CRC 功能说明

CRC 计算单元主要由单个 32 位数据寄存器组成，该寄存器：

- 用作输入寄存器，向 CRC 计算器中输入新数据 （向寄存器写入数据时）
- 可保存之前的 CRC 计算结果 （读取寄存器时）

对数据寄存器的每个写操作都会把当前新输入的数值和之前生成在数据寄存器中的 CRC 值再做一次 CRC 计算 （CRC 计算针对整个 32 位数据字完成，而非逐字节进行）。

CRC 计算的时候，写操作被阻塞，因此允许执行背靠背写访问或连续的写读访问。

使用 CRC_CR 寄存器中的 RESET 控制位即可将 CRC 计算器复位为 0xFFFF FFFF。此操作不影响 CRC_IDR 寄存器的内容。

4.4 CRC 寄存器

CRC 计算单元包含两个数据寄存器和一个控制寄存器。CRC 寄存器必须按字（32 位）访问。

4.4.1 数据寄存器 (CRC_DR)

Data register

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	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
DR [15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:0 **数据寄存器位**

向 CRC 计算器写入新数据时用作输入寄存器。
读取寄存器时可读出之前的 CRC 计算结果。

4.4.2 独立数据寄存器 (CRC_IDR)

Independent data register

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

位 31:8 保留，必须保持复位值。

位 7:0 **通用的 8 位数据寄存器位**

可用作一个字节的临时存储单元。
此寄存器不受 CRC_CR 寄存器中 RESET 位产生的 CRC 复位影响。

4.4.3 控制寄存器 (CRC_CR)

Control register

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

位 31:1 保留，必须保持复位值。

位 0 **RESET 位**

复位 CRC 计算单元并将数据寄存器设为 0xFFFF FFFF。

此位只能置 1，将由硬件自动进行清零。

4.4.4 CRC 寄存器映射

下表提供了 CRC 寄存器映射和复位值。

表 17. CRC 计算单元寄存器映射和复位值

偏移	寄存器	31-24	23-16	15-8	7	6	5	4	3	2	1	0
0x00	**CRC_DR** Reset value	Data register 0xFFFF FFFF										
0x04	**CRC_IDR** Reset value	Reserved			Independent data register 0x00							
0x08	**CRC_CR** Reset value	Reserved										RESET 0

5 电源控制器 (PWR)

除非特别说明，否则本部分适用于整个 STM32F4xx 系列。

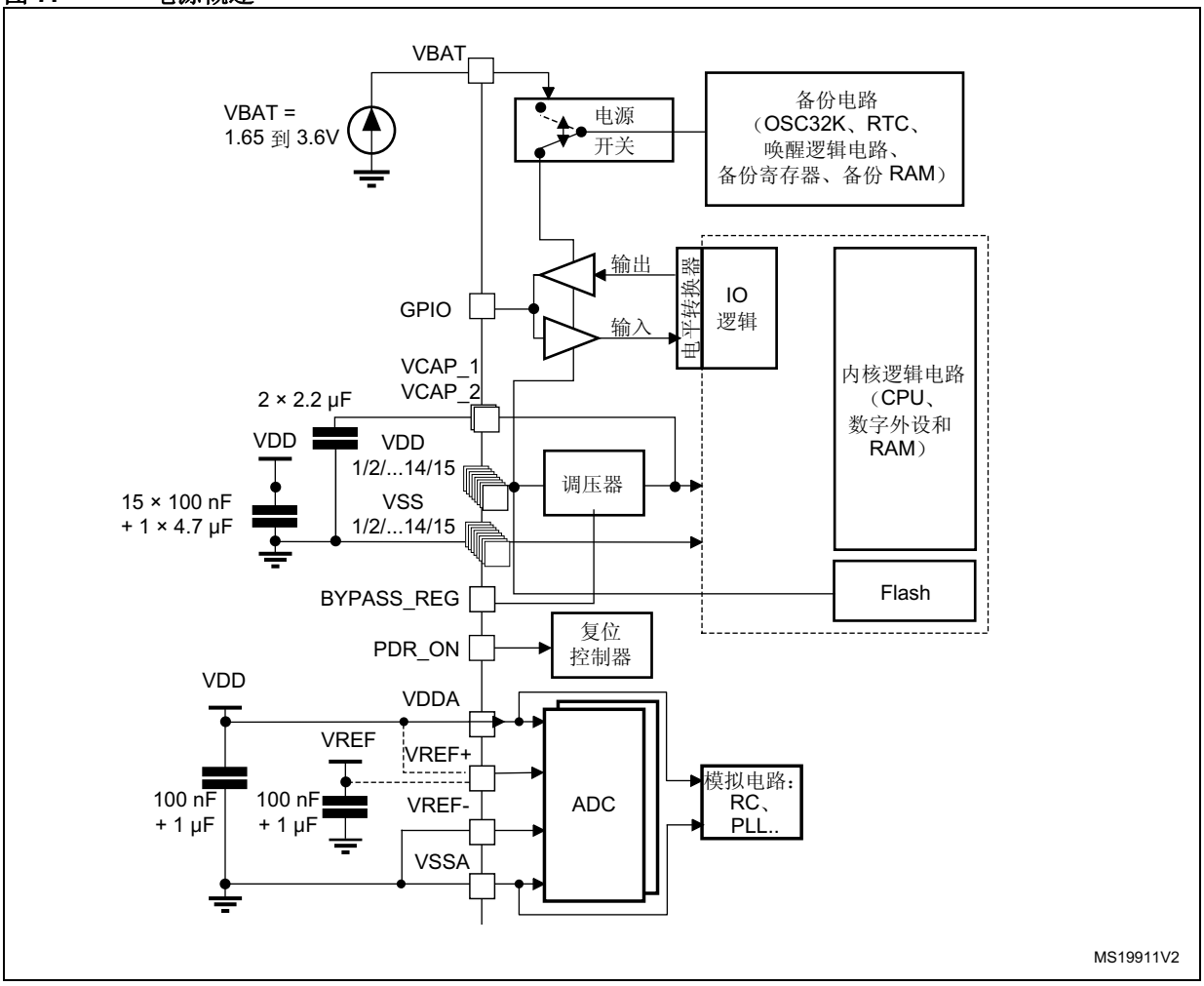
5.1 电源

器件的工作电压 (V_{DD}) 要求介于 1.8 V 到 3.6 V 之间。嵌入式线性调压器用于提供内部 1.2 V 数字电源。

当主电源 V_{DD} 断电时，可通过 V_{BAT} 电压为实时时钟 (RTC)、RTC 备份寄存器和备份 SRAM (BKP SRAM) 供电。

注意： *根据工作期间供电电压的不同，某些外设可能只提供有限的功能和性能。有关详细信息，请参见 STM32F4xx 数据手册中“通用工作条件”一节。*

图 7. 电源概述



1. V_{DDA} 和 V_{SSA} 必须分别连接到 V_{DD} 和 V_{SS}。

5.1.1 独立 A/D 转换器电源和参考电压

为了提高转换精度，ADC 配有独立电源，可以单独滤波并屏蔽 PCB 上的噪声。

- ADC 电源电压从单独的 V_{DDA} 引脚输入。
- V_{SSA} 引脚提供了独立的电源接地连接。

为了确保测量低电压时具有更高的精度，用户可以在 V_{REF} 上连接单独的 ADC 外部参考电压输入。V_{REF} 电压介于 1.8 V 到 V_{DDA} 之间。

5.1.2 电池备份域

备份域说明

要在 V_{DD} 关闭后保留 RTC 备份寄存器和备份 SRAM 的内容并为 RTC 供电，可以将 V_{BAT} 引脚连接到通过电池或其它电源供电的可选备用电压。

要使 RTC 即使在主数字电源 (V_{DD}) 关闭后仍然工作，V_{BAT} 引脚需为以下各模块供电：

- RTC
- LSE 振荡器
- 备份 SRAM（使能低功耗备份调压器时）
- PC13 到 PC15 I/O，以及 PI8 I/O（如果封装有该引脚）

V_{BAT} 电源的开关由复位模块中内置的掉电复位电路进行控制。

警告： **在 $t_{RSTTEMPO}$（V_{DD} 启动后的一段延迟）期间或检测到 PDR 后，V_{BAT} 与 V_{DD} 之间的电源开关仍连接到 V_{BAT}。在启动阶段，如果 V_{DD} 的建立时间小于 $t_{RSTTEMPO}$（有关 $t_{RSTTEMPO}$ 的值，请参见数据手册）且 $V_{DD} > V_{BAT} + 0.6$ V，会有电流经由 V_{DD} 和电源开关 (V_{BAT}) 之间连接的内部二极管注入 V_{BAT} 引脚。如果连接到 V_{BAT} 引脚的电源/电池无法承受此注入电流，则强烈建议在该电源与 V_{BAT} 引脚之间连接一个低压降二极管。**

如果应用中未使用任何外部电池，建议将 V_{BAT} 引脚连接到并联了 100 nF 外部去耦陶瓷电容的 V_{DD}。

通过 V_{DD} 对备份域供电时（模拟开关连接到 V_{DD}），可实现以下功能：

- PC14 和 PC15 可用作 GPIO 或 LSE 引脚
- PC13 可用作 GPIO 或 RTC_AF1 引脚（有关此引脚配置的详细信息，请参见*表 30：RTC_AF1 引脚*）

注意： *由于该开关的灌电流能力有限 (3 mA)，因此使用 GPIO PI8 和 PC13 到 PC15 时存在以下限制：每次只能有一个 I/O 用作输出，最大负载为 30 pF 时速率不得超过 2 MHz，并且这些 I/O 不能用作电流源（如用于驱动 LED）。*

通过 V_{BAT} 对备份域供电时（由于不存在 V_{DD}，模拟开关连接到 V_{BAT}），可实现以下功能：

- PC14 和 PC15 只能用作 LSE 引脚
- PC13 可用作 RTC_AF1 引脚（有关此引脚配置的详细信息，请参见*表 30：RTC_AF1 引脚*）
- PI8 可用作 RTC_AF2

备份域访问

复位后，备份域（RTC 寄存器、RTC 备份寄存器和备份 SRAM）将受到保护，以防止意外的写访问。要使能对备份域的访问，请按以下步骤进行操作：

- 访问 RTC 和 RTC 备份寄存器

1. 将 RCC_APB1ENR 寄存器中的 PWREN 位置 1，使能电源接口时钟（分别参见*第 6.3.15 节*和*第 6.3.16 节*了解 STM32F405xx/07xx 和 STM32F415xx/17xx 和 STM32F42xxx 和 STM32F43xxx）
2. 将*用于 STM32F405xx/07xx 和 STM32F415xx/17xx 的 PWR 电源控制寄存器 (PWR_CR)* 和 *用于STM32F42xxx 和 STM32F43xxx 的 PWR 电源控制寄存器 (PWR_CR)* 中的 DBP 位置 1，使能对备份域的访问
3. 选择 RTC 时钟源：参见*第 6.2.8 节：RTC/AWU 时钟*
4. 通过对 *RCC 备份域控制寄存器 (RCC_BDCR)* 中的 RTCEN [15] 位进行编程，使能 RTC 时钟

- 访问备份 SRAM

1. 将 RCC_APB1ENR 寄存器中的 PWREN 位置 1，使能电源接口时钟（分别参见*第 6.3.15 节*和*第 6.3.16 节*了解 STM32F405xx/07xx 和 STM32F415xx/17xx 和 STM32F42xxx 和 STM32F43xxx）。
2. 将*用于 STM32F405xx/07xx 和 STM32F415xx/17xx 的 PWR 电源控制寄存器 (PWR_CR)* 和*用于STM32F42xxx 和 STM32F43xxx 的 PWR 电源控制寄存器 (PWR_CR)* 中的 DBP 位置 1，使能对备份域的访问。
3. 通过将 *RCC AHB1 外设时钟使能寄存器 (RCC_AHB1ENR)* 中的 BKPSRAMEN 位置 1，使能备份 SRAM 时钟。

RTC 和 RTC 备份寄存器

实时时钟 (RTC) 是一个独立的 BCD 定时器/计数器。RTC 提供一个日历时钟、两个可编程闹钟中断，以及一个具有中断功能的可编程的周期唤醒标志。RTC 包含 20 个备份数据寄存器（80 字节），在检测到入侵事件时将复位。有关详细信息，请参见*第 23 节：实时时钟 (RTC)*。

备份 SRAM

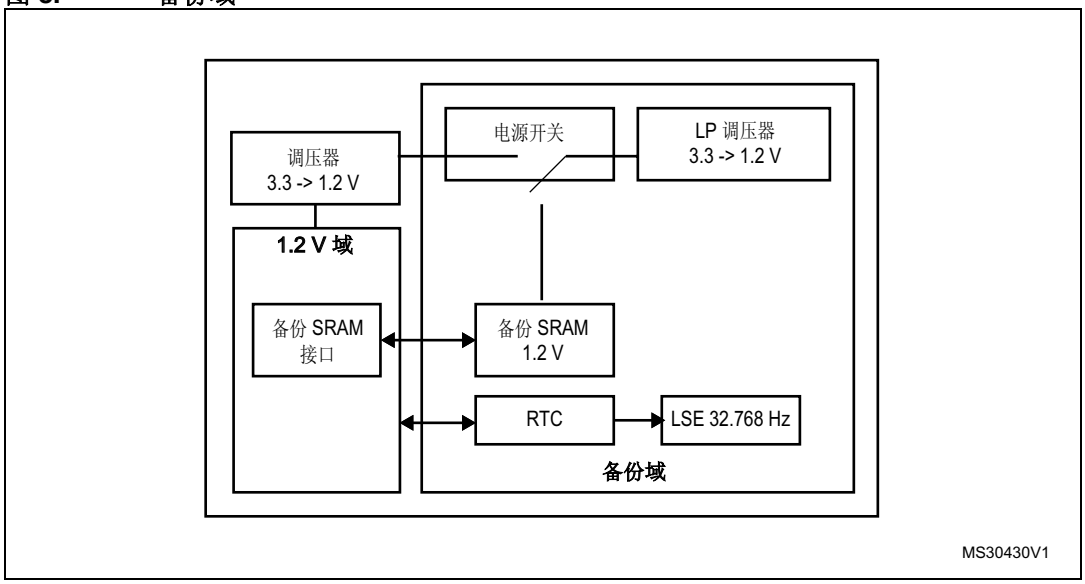
备份域还包括仅可由 CPU 访问的 4 KB 备份 SRAM，可被 32 位、16 位、8 位访问。使能低功耗备份调压器时，即使处于待机或 V_{BAT} 模式，备份 SRAM 的内容也能保留。一直存在 V_{BAT} 时，可以将此备份 SRAM 视为内部 EEPROM。

通过 V_{DD}（模拟开关连接到 V_{DD}）对备份域供电时，备份 SRAM 将从 V_{DD} 而非 V_{BAT} 获取电能，以此延长电池寿命。

通过 V_{BAT}（由于不存在 V_{DD}，模拟开关连接到 V_{BAT}）对备份域供电时，备份 SRAM 通过专用的低功耗调压器供电。此调压器既可以处于开启状态，也可以处于关闭状态，具体取决于应用在待机模式或 V_{BAT} 模式是否需要备份 SRAM 功能。此调压器的掉电由专用位控制，即 PWR_CSR 寄存器的 BRE 控制位（参见*第 5.4.3 节：PWR 电源控制/状态寄存器 (PWR_CSR)*）。

入侵事件不会擦除备份 SRAM。备份 SRAM 设置了读保护，可防止用户对加密私钥等机密数据进行访问。擦除备份 SRAM 的唯一方法是在请求将保护级别从级别 1 更改为级别 0 时通过 Flash 接口实现。请参见 Flash 编程手册中的读保护 (RDP) 说明。

图 8. 备份域



5.1.3 调压器

嵌入式线性调压器为备份域和待机电路以外的所有数字电路供电。调压器输出电压约为 1.2 V。

此调压器需要将两个外部电容连接到专用引脚 V_{CAP_1} 和 V_{CAP_2}，所有封装都配有这两个引脚。为激活或停用调压器，必须将特定引脚连接到 V_{SS} 或 V_{DD}。具体引脚与封装有关。

通过软件激活时，调压器在复位后始终处于使能状态。根据应用模式的不同，可采用三种不同的模式工作。

- **运行模式**，调压器为 1.2 V 域（内核、存储器和数字外设）提供全功率。在此模式下，调压器的输出电压（约 1.2 V）可通过软件调整为不同的电压值：
  - 对于 STM32F405xx/07xx 和 STM32F415xx/17xx

    可通过 VOS（PWR_CR 寄存器的位 15）动态配置成级别 1 或级别 2。
  - 对于 STM32F42xxx 和 STM32F43xxx

    可通过 PWR_CR 寄存器的 VOS[1:0] 位配置成级别 1、级别 2 或级别 3。仅当关闭 PLL 且选择 HSI 或 HSE 时钟源作为系统时钟源时，才能修改输出级别。新的编程值只在 PLL 开启后才生效。PLL 关闭后将自动选择电压输出级别 3（参见*第 5.4.2 节：用于STM32F42xxx 和 STM32F43xxx 的 PWR 电源控制寄存器 (PWR_CR)*）。

  器件运行在最高工作频率时，电压缩放特性可使功耗得到优化。
- **停止模式**，调压器为 1.2 V 域提供低功率，保留寄存器和内部 SRAM 中的内容。
  - 对于 STM32F405xx/07xx 和 STM32F415xx/17xx

    在停止模式下，设置的电压输出级别保持不变（参见*第 5.4.1 节：用于 STM32F405xx/07xx 和 STM32F415xx/17xx 的 PWR 电源控制寄存器 (PWR_CR)*）。
  - 对于 STM32F42xxx 和 STM32F43xxx

    微控制器进入停止模式后将自动选择电压输出级别 3（参见*第 5.4.2 节：用于 STM32F42xxx 和 STM32F43xxx 的 PWR 电源控制寄存器 (PWR_CR)*）。
- **待机模式**，调压器掉电。除待机电路和备份域外，寄存器和 SRAM 的内容都将丢失。

注意： *有关详细信息，请参见 STM32F4xx 数据手册的调压器部分。*

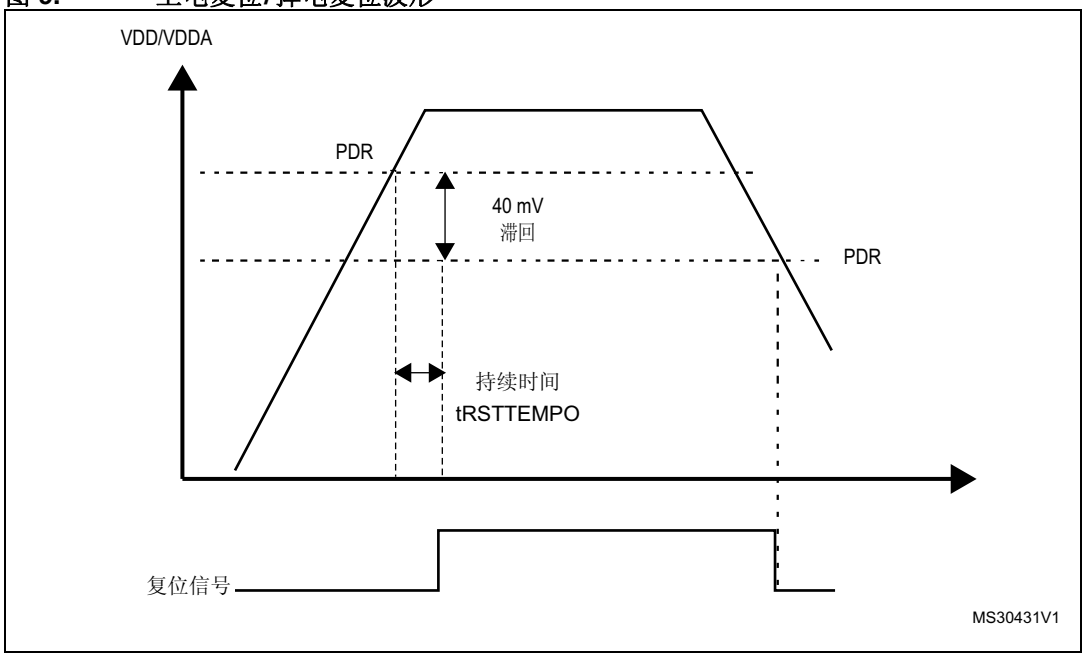
5.2 电源监控器

5.2.1 上电复位 (POR)/掉电复位 (PDR)

本器件内部集成有 POR/PDR 电路，可以从 1.8 V 开始正常工作。

当 V_{DD}/V_{DDA} 低于指定阈值 $V_{POR/PDR}$ 时，器件无需外部复位电路便会保持复位状态。有关上电/掉电复位阈值的相关详细信息，请参见数据手册的电气特性部分。

图 9. 上电复位/掉电复位波形



5.2.2 欠压复位 (BOR)

上电期间，欠压复位 (BOR) 将使器件保持复位状态，直到电源电压达到指定的 V_{BOR} 阈值。

V_{BOR} 通过器件选项字节进行配置。BOR 默认为关闭。可以选择 4 个 V_{BOR} 阈值。

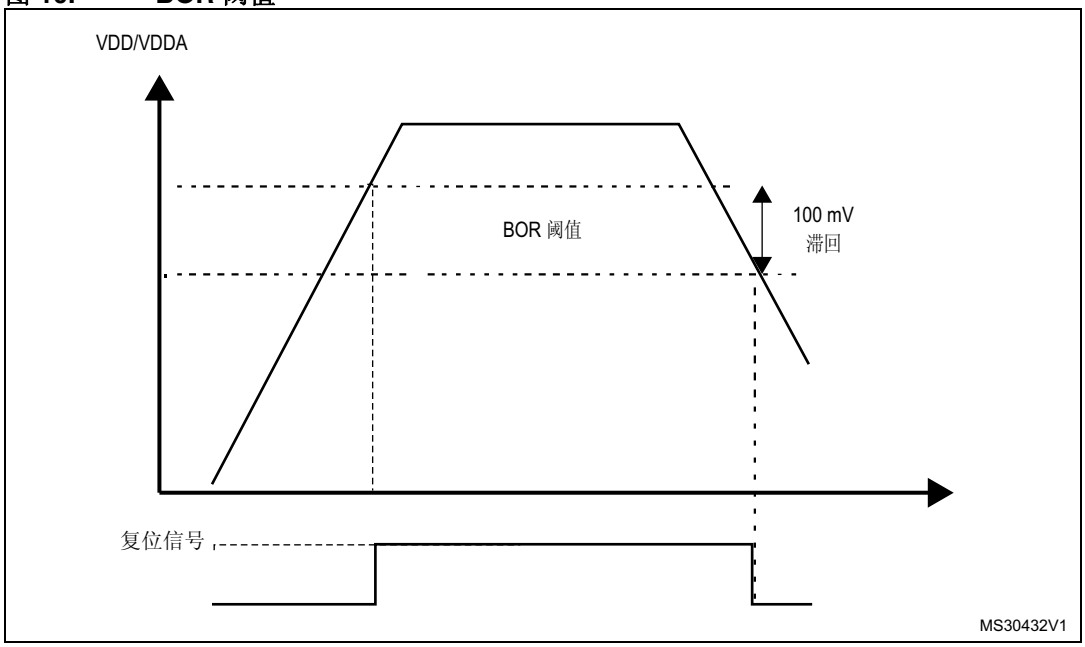
- BOR 关闭 (V_{BOR0})：1.80 V 到 2.10 V 电压范围的复位阈值级别
- BOR 级别 1 (V_{BOR1})：2.10 V 到 2.40 V 电压范围的复位阈值级别
- BOR 级别 2 (V_{BOR2})：2.40 V 到 2.70 V 电压范围的复位阈值级别
- BOR 级别 3 (V_{BOR3})：2.70 V 到 3.60 V 电压范围的复位阈值级别

当电源电压 (V_{DD}) 降至所选 V_{BOR} 阈值以下时，将使器件复位。

通过对器件选项字节进行编程可以禁止 BOR。要禁止 BOR 功能，V_{DD} 必须高于 V_{BOR0}，以启动器件选项字节编程序列。那么就只能由 PDR 监测掉电过程（参见*第 5.2.1 节：上电复位 (POR)/掉电复位 (PDR)*）

BOR 阈值滞回电压约为 100 mV（电源电压的上升沿与下降沿之间）。

图 10. BOR 阈值



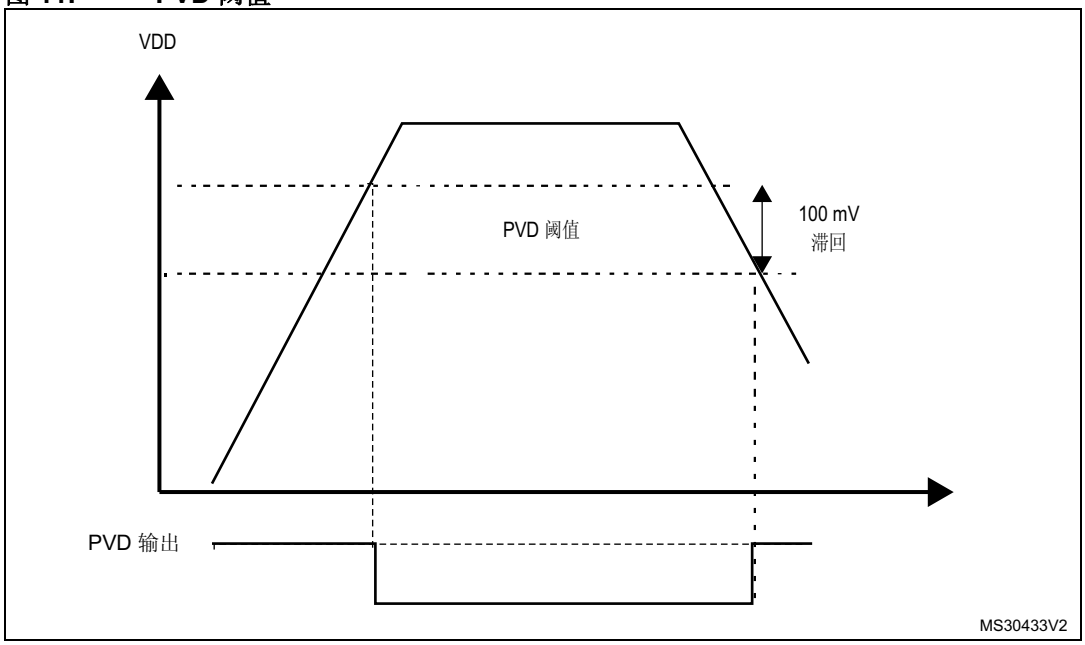
5.2.3 可编程电压检测器 (PVD)

可以使用 PVD 监视 V_{DD} 电源，将其与*用于 STM32F405xx/07xx 和 STM32F415xx/17xx 的 PWR 电源控制寄存器 (PWR_CR)* 和*用于STM32F42xxx 和 STM32F43xxx 的 PWR 电源控制寄存器 (PWR_CR)* 中 PLS[2:0] 位所选的阈值进行比较。

通过设置 PVDE 位来使能 PVD。

PWR 电源控制/状态寄存器 (PWR_CSR) 中提供了 PVDO 标志，用于指示 V_{DD} 是大于还是小于 PVD 阈值。该事件内部连接到 EXTI 线 16，如果通过 EXTI 寄存器使能，则可以产生中断。当 V_{DD} 降至 PVD 阈值以下以及/或者当 V_{DD} 升至 PVD 阈值以上时，可以产生 PVD 输出中断，具体取决于 EXTI 线 16 上升沿/下降沿的配置。该功能的用处之一就是可以在中断服务程序中执行紧急关闭系统的任务。

图 11. PVD 阈值



5.3 低功耗模式

默认情况下，系统复位或上电复位后，微控制器进入运行模式。在运行模式下，CPU 通过 HCLK 提供时钟，并执行程序代码。系统提供了多个低功耗模式，可在 CPU 不需要运行时（例如等待外部事件时）节省功耗。由用户根据应用选择具体的低功耗模式，以在低功耗、短启动时间和可用唤醒源之间寻求最佳平衡。

器件有三个低功耗模式：

- 睡眠模式（Cortex™-M4F 内核停止，外设保持运行）
- 停止模式（所有时钟都停止）
- 待机模式（1.2 V 域断电）

此外，可通过下列方法之一降低运行模式的功耗：

- 降低系统时钟速度
- 不使用 APBx 和 AHBx 外设时，将对应的外设时钟关闭

表 18. 低功耗模式汇总

模式名称	进入	唤醒	对 1.2 V 域时钟的影响	对 V_{DD} 域时钟的影响	调压器
睡眠（立即休眠或退出时休眠）	WFI	任意中断	CPU CLK 关闭对其它时钟或模拟时钟源无影响	无	开启
	WFE	唤醒事件			
停止	PDDS 和 LPDS 位 + SLEEPDEEP 位 + WFI 或 WFE	任意 EXTI 线（在 EXTI 寄存器中配置，内部线和外部线）	所有 1.2 V 域时钟都关闭	HSI 和 HSE 振荡器关闭	开启或处于低功耗模式（取决于 *用于 STM32F405xx/07xx 和 STM32F415xx/17xx 的 PWR 电源控制寄存器 (PWR_CR)* 和 *用于STM32F42xxx 和 STM32F43xxx 的 PWR 电源控制寄存器 (PWR_CR)*）
待机	PDDS 位 + SLEEPDEEP 位 + WFI 或 WFE	WKUP 引脚上升沿、RTC 闹钟（闹钟 A 或闹钟 B）、RTC 唤醒事件、RTC 入侵事件、RTC 时间戳事件、NRST 引脚外部复位、IWDG 复位	所有 1.2 V 域时钟都关闭	HSI 和 HSE 振荡器关闭	关闭

5.3.1 降低系统时钟速度

在运行模式下，可通过对预分频寄存器编程来降低系统时钟（SYSCLK、HCLK、PCLK1 和 PCLK2）速度。进入睡眠模式之前，也可以使用这些预分频器降低外设速度。

有关详细信息，请参见*第 6.3.3 节：RCC 时钟配置寄存器 (RCC_CFGR)*。

5.3.2 外设时钟门控

在运行模式下，可随时停止各外设和存储器的 HCLKx 和 PCLKx 以降低功耗。

要进一步降低睡眠模式的功耗，可在执行 WFI 或 WFE 指令之前禁止外设时钟。

外设时钟门控由 AHB1 外设时钟使能寄存器 (RCC_AHB1ENR)、AHB2 外设时钟使能寄存器 (RCC_AHB2ENR) 和 AHB3 外设时钟使能寄存器 (RCC_AHB3ENR) 进行控制（参见*第 6.3.12 节：RCC AHB1 外设时钟使能寄存器 (RCC_AHB1ENR)*、*第 6.3.13 节：RCC AHB2 外设时钟使能寄存器 (RCC_AHB2ENR)*、*第 6.3.14 节：RCC AHB3 外设时钟使能寄存器 (RCC_AHB3ENR)*）。

在睡眠模式下，复位 RCC_AHBxLPENR 和 RCC_APBxLPENR 寄存器中的对应位可以自动禁止外设时钟。

5.3.3 睡眠模式

进入睡眠模式

执行 WFI（等待中断）或 WFE（等待事件）指令即可进入睡眠模式。根据 Cortex™-M4F 系统控制寄存器中 SLEEPONEXIT 位的设置，可以通过两种方案选择睡眠模式进入机制：

- 立即休眠：如果 SLEEPONEXIT 位清零，MCU 将在执行 WFI 或 WFE 指令时立即进入睡眠模式。
- 退出时休眠：如果 SLEEPONEXIT 位置 1，MCU 将在退出优先级最低的 ISR 时立即进入睡眠模式。

有关如何进入睡眠模式的详细信息，请参见*表 19* 和*表 20*。

退出睡眠模式

如果使用 WFI 指令进入睡眠模式，则嵌套向量中断控制器 (NVIC) 确认的任意外设中断都会将器件从睡眠模式唤醒。

如果使用 WFE 指令进入睡眠模式，MCU 将在有事件发生时立即退出睡眠模式。唤醒事件可通过以下方式产生：

- 在外设的控制寄存器使能一个中断，但不在 NVIC 中使能，同时使能 Cortex™-M4F 系统控制寄存器中的 SEVONPEND 位。当 MCU 从 WFE 恢复时，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 恢复时，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。

由于没有在进入/退出中断时浪费时间，此模式下的唤醒时间最短。

有关如何退出睡眠模式的详细信息，请参见*表 19* 和*表 20*。

表 19. 进入和退出立即休眠

立即休眠模式	说明
进入模式	WFI（等待中断）或 WFE（等待事件），且： – SLEEPDEEP = 0 及 – SLEEPONEXIT = 0 请参见 Cortex™-M4F 系统控制寄存器。
退出模式	如果使用 WFI 进入： 中断：请参见*表 45：STM32F405xx/07xx 和 STM32F415xx/17xx 的向量表*和*表 46：STM32F42xxx 和 STM32F43xxx 的向量表* 如果使用 WFE 进入： 唤醒事件：请参见*第 10.2.3 节：唤醒事件管理*
唤醒延迟	无

表 20. 进入和退出退出时休眠

退出时休眠	说明
进入模式	WFI（等待中断），且： – SLEEPDEEP = 0 及 – SLEEPONEXIT = 1 请参见 Cortex™-M4F 系统控制寄存器。
退出模式	中断：请参见*表 45：STM32F405xx/07xx 和 STM32F415xx/17xx 的向量表*和*表 46：STM32F42xxx 和 STM32F43xxx 的向量表*
唤醒延迟	无

5.3.4 停止模式

停止模式基于 Cortex™-M4F 深度睡眠模式与外设时钟门控。调压器既可以配置为正常模式，也可以配置为低功耗模式。在停止模式下，1.2 V 域中的所有时钟都会停止，PLL、HSI 和 HSE RC 振荡器也被禁止。内部 SRAM 和寄存器内容将保留。

将 PWR_CR 寄存器中的 FPDS 位置 1 后，Flash 还会在器件进入停止模式时进入掉电状态。Flash 处于掉电模式时，将器件从停止模式唤醒将需要额外的启动延时（参见*表 21：停止工作模式*）。

表 21. 停止工作模式

停止模式	LPDS 位	FPDS 位	唤醒延迟
STOP MR （主调压器）	0	0	HSI RC 启动时间
STOP MR-FPD	0	1	HSI RC 启动时间 + Flash 从掉电模式唤醒的时间
STOP LP	1	0	HSI RC 启动时间 + 调压器从 LP 模式唤醒的时间
STOP LP-FPD	1	1	HSI RC 启动时间 + Flash 从掉电模式唤醒的时间 + 调压器从 LP 模式唤醒的时间

进入停止模式

有关如何进入停止模式的详细信息，请参见*表 22*。

要进一步降低停止模式的功耗，可将内部调压器设置为低功耗模式。通过*用于 STM32F405xx/07xx 和 STM32F415xx/17xx 的 PWR 电源控制寄存器 (PWR_CR)* 和*用于 STM32F42xxx 和 STM32F43xxx 的 PWR 电源控制寄存器 (PWR_CR)* 的 LPDS 位进行配置。

如果正在执行 Flash 编程，停止模式的进入将延迟到存储器访问结束后执行。

如果正在访问 APB 域，停止模式的进入则延迟到 APB 访问结束后执行。

在停止模式下，可以通过对各控制位进行编程来选择以下功能：

- 独立的看门狗 (IWDG)：IWDG 通过写入其密钥寄存器或使用硬件选项来启动。而且一旦启动便无法停止，除非复位。请参见*第 18 节：独立看门狗 (IWDG)* 中的*第 18.3 节*。
- 实时时钟 (RTC)：通过 *RCC 备份域控制寄存器 (RCC_BDCR)* 中的 RTCEN 位进行配置。
- 内部 RC 振荡器 (LSI RC)：通过 *RCC 时钟控制和状态寄存器 (RCC_CSR)* 中的 LSION 位进行配置。
- 外部 32.768 kHz 振荡器 (LSE OSC)：通过 *RCC 备份域控制寄存器 (RCC_BDCR)* 中的 LSEON 位进行配置。

在停止模式下，ADC 或 DAC 也会产生功耗，除非在进入停止模式前将其禁止。要禁止这些转换器，必须将 ADC_CR2 寄存器中的 ADON 位和 DAC_CR 寄存器中的 ENx 位都清零。

退出停止模式

有关如何退出停止模式的详细信息，请参见*表 22*。

通过发出中断或唤醒事件退出停止模式时，将选择 HSI RC 振荡器作为系统时钟。

当调压器在低功耗模式下工作时，将器件从停止模式唤醒将需要额外的延时。在停止模式下一直开启内部调压器虽然可以缩短启动时间，但功耗却增大。

表 22. 进入和退出停止模式

停止模式	说明
进入模式	WFI（等待中断）或 WFE（等待事件），且： – 将 Cortex™-M4F 系统控制寄存器中的 SLEEPDEEP 位置 1 – 将电源控制寄存器 (PWR_CR) 中的 PDDS 位清零 – 通过配置 PWR_CR 中的 LPDS 位选择调压器模式。 **注意：**要进入停止模式，所有 EXTI 线挂起位（在*挂起寄存器 (EXTI_PR)* 中）、RTC 闹钟（闹钟 A 和闹钟 B）、RTC 唤醒、 RTC 入侵和 RTC 时间戳标志必须复位。否则将忽略进入停止模式这一过程，继续执行程序。
退出模式	如果使用 WFI 进入： 所有配置为中断模式的 EXTI 线（必须在 NVIC 中使能对应的 EXTI 中断向量）。请参见*表 45：第 234 页的STM32F405xx/07xx 和 STM32F415xx/17xx 的向量表*和*表 46：STM32F42xxx 和 STM32F43xxx 的向量表*。 如果使用 WFE 进入： 所有配置为事件模式的 EXTI 线。请参见*第 10.2.3 节：第 241 页的唤醒事件管理*。
唤醒延迟	*表 21：停止工作模式*

5.3.5 待机模式

待机模式下可达到最低功耗。待机模式基于 Cortex™-M4F 深度睡眠模式，其中调压器被禁止。因此 1.2 V 域断电。PLL、HSI 振荡器和 HSE 振荡器也将关闭。除备份域（RTC 寄存器、RTC 备份寄存器和备份 SRAM）和待机电路中的寄存器外，SRAM 和寄存器内容都将丢失（参见*图 7*）。

进入待机模式

有关如何进入待机模式的详细信息，请参见*表 23*。

在待机模式下，可以通过对各控制位进行编程来选择以下功能：

- 独立的看门狗 (IWDG)：IWDG 通过写入其密钥寄存器或使用硬件选项来启动。而且一旦启动便无法停止，除非复位。请参见*第 18 节：独立看门狗 (IWDG)* 中的*第 18.3 节*。
- 实时时钟 (RTC)：通过备份域控制寄存器 (RCC_BDCR) 中的 RTCEN 位进行配置。
- 内部 RC 振荡器 (LSI RC)：通过控制/状态寄存器 (RCC_CSR) 中的 LSION 位进行配置。
- 外部 32.768 kHz 振荡器 (LSE OSC)：通过备份域控制寄存器 (RCC_BDCR) 中的 LSEON 位进行配置。

退出待机模式

检测到外部复位（NRST 引脚）、IWDG 复位、WKUP 引脚上升沿、RTC 闹钟、入侵事件或时间戳时间时，微控制器退出待机模式。从待机模式唤醒后，除 *PWR 电源控制/状态寄存器 (PWR_CSR)* 外，所有寄存器都将复位。

从待机模式唤醒后，程序将按照复位（启动引脚采样、复位向量已获取等）后的方式重新执行。*PWR 电源控制/状态寄存器 (PWR_CSR)* 中的 SBF 状态标志指示 MCU 已处于待机模式。

有关如何退出待机模式的详细信息，请参见*表 23*。

表 23. 进入和退出待机模式

待机模式	说明
进入模式	WFI（等待中断）或 WFE（等待事件），且： – 将 Cortex™-M4F 系统控制寄存器中的 SLEEPDEEP 位置 1 – 将电源控制寄存器 (PWR_CR) 中的 PDDS 位置 1 – 将电源控制/状态寄存器 (PWR_CSR) 中的 WUF 位清零 – 将与所选唤醒源（RTC 闹钟 A、RTC 闹钟 B、RTC 唤醒、RTC 入侵或 RTC 时间戳标志）对应的 RTC 标志清零
退出模式	WKUP 引脚上升沿、RTC 闹钟（闹钟 A 和闹钟 B）、RTC 唤醒事件、RTC 入侵事件、RTC 时间戳事件、NRST 引脚外部复位 和 IWDG 复位。
唤醒延迟	复位阶段。

待机模式下的 I/O 状态

在待机模式下，除以下各部分以外，所有 I/O 引脚都处于高阻态：

- 复位引脚（仍可用）
- RTC_AF1 引脚 (PC13)（如果针对入侵、时间戳、RTC 闹钟输出或 RTC 时钟校准输出进行了配置）
- WKUP 引脚 (PA0)（如果使能）

调试模式

默认情况下，如果使用调试功能时应用程序将 MCU 置于停止模式或待机模式，调试连接将中断。这是因为 Cortex™-M4F 内核时钟停止了。

不过，通过设置 DBGMCU_CR 寄存器中的一些配置位，即使 MCU 进入低功耗模式，仍可使用软件对其进行调试。有关详细信息，请参见*第 33.16.1 节：对低功耗模式的调试支持*。

5.3.6 对 RTC 复用功能进行编程以从停止模式和待机模式唤醒器件

RTC 复用功能可以从低功耗模式唤醒 MCU。

RTC 复用功能包括 RTC 闹钟（闹钟 A 和闹钟 B）、RTC 唤醒事件、RTC 入侵事件和 RTC 时间戳事件。

这些 RTC 复用功能可将系统从停止和待机低功耗模式唤醒。

通过使用 RTC 闹钟或 RTC 唤醒事件，无需依赖外部中断即可将系统从低功耗模式唤醒（自动唤醒模式）。

RTC 提供了可编程时基，便于定期从停止或待机模式唤醒器件。

为此，通过对 *RCC 备份域控制寄存器 (RCC_BDCR)* 中的 RTCSEL[1:0] 位进行编程，可以选择三个复用 RTC 时钟源中的其中两个：

- 低功耗 32.768 kHz 外部晶振 (LSE OSC)
  此时钟源提供的时基非常精确，功耗也非常低（典型条件下功耗小于 1 µA）
- 低功耗内部 RC 振荡器 (LSI RC)
  此时钟源的优势在于可以节省 32.768 kHz 晶振的成本。此内部 RC 振荡器非常省电。

通过 RTC 复用功能从停止模式唤醒器件

- 要通过 RTC 闹钟事件从停止模式唤醒器件，必须：
  a) 将 EXTI 线 17 配置为检测外部信号的上升沿（中断或事件模式）
  b) 使能 RTC_CR 寄存器中的 RTC 闹钟中断
  c) 配置 RTC 以生成 RTC 闹钟
- 要通过 RTC 入侵事件或时间戳事件从停止模式唤醒器件，必须：
  a) 将 EXTI 线 21 配置为检测外部信号的上升沿（中断或事件模式）
  b) 使能 RTC_CR 寄存器中的 RTC 时间戳中断，或者使能 RTC_TAFCR 寄存器中的 RTC 入侵中断
  c) 配置 RTC 以检测入侵事件或时间戳事件
- 要通过 RTC 唤醒事件从停止模式唤醒器件，必须：
  a) 将 EXTI 线 22 配置为检测外部信号的上升沿（中断或事件模式）
  b) 使能 RTC_CR 寄存器中的 RTC 唤醒中断
  c) 配置 RTC 以生成 RTC 唤醒事件

通过 RTC 复用功能从待机模式唤醒器件

- 要通过 RTC 闹钟事件从待机模式唤醒器件，必须：
  a) 使能 RTC_CR 寄存器中的 RTC 闹钟中断
  b) 配置 RTC 以生成 RTC 闹钟
- 要通过 RTC 入侵事件或时间戳事件从待机模式唤醒器件，必须：
  a) 使能 RTC_CR 寄存器中的 RTC 时间戳中断，或者使能 RTC_TAFCR 寄存器中的 RTC 入侵中断
  b) 配置 RTC 以检测入侵事件或时间戳事件
- 要通过 RTC 唤醒事件从待机模式唤醒器件，必须：
  a) 使能 RTC_CR 寄存器中的 RTC 唤醒中断
  b) 配置 RTC 以生成 RTC 唤醒事件

RTC 复用功能唤醒标志安全清零顺序

如果在 PWR 唤醒标志 (WUTF) 清零之前将所选 RTC 复用功能置 1，则出现下一事件时无法检测到相关功能，因为检测操作只在信号上升沿到来时执行一次。

为了避免 RTC 复用功能所映射到的引脚发生跳变，并确保器件从停止模式和待机模式正常退出，建议在进入待机模式之前按照以下顺序进行操作：

- 使用 RTC 闹钟从低功耗模式唤醒器件时：
  a) 禁止 RTC 闹钟中断（RTC_CR 寄存器中的 ALRAIE 或 ALRBIE 位）
  b) 将 RTC 闹钟 (ALRAF/ALRBF) 标志清零
  c) 将 PWR 唤醒 (WUF) 标志清零
  d) 使能 RTC 闹钟中断
  e) 重新进入低功耗模式
- 使用 RTC 唤醒从低功耗模式唤醒器件时：
  a) 禁止 RTC 唤醒中断（RTC_CR 寄存器中的 WUTIE 位）
  b) 将 RTC 唤醒 (WUTF) 标志清零
  c) 将 PWR 唤醒 (WUF) 标志清零
  d) 使能 RTC 唤醒中断
  e) 重新进入低功耗模式
- 使用 RTC 入侵从低功耗模式唤醒器件时：
  a) 禁止 RTC 入侵中断（RTC_TAFCR 寄存器中的 TAMPIE 位）
  b) 将入侵 (TAMP1F/TSF) 标志清零
  c) 将 PWR 唤醒 (WUF) 标志清零
  d) 使能 RTC 入侵中断
  e) 重新进入低功耗模式
- 使用 RTC 时间戳从低功耗模式唤醒器件时：
  a) 禁止 RTC 时间戳中断（RTC_CR 寄存器中的 TSIE 位）
  b) 将 RTC 时间戳 (TSF) 标志清零
  c) 将 PWR 唤醒 (WUF) 标志清零
  d) 使能 RTC 时间戳中断
  e) 重新进入低功耗模式

5.4 电源控制寄存器

5.4.1 用于 STM32F405xx/07xx 和 STM32F415xx/17xx 的 PWR 电源控制寄存器 (PWR_CR)

PWR power control register

偏移地址：0x00

复位值：0x0000 4000（通过从待机模式唤醒进行复位）

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	VOS	Reserved				FPDS	DBP	PLS[2:0]			PVDE	CSBF	CWUF	PDDS	LPDS
	rw					rw	rw	rw	rw	rw	rw	rc_w1	rc_w1	rw	rw

位 31:15 保留，必须保持复位值。

位 14 **VOS**：调压器输出电压级别选择 (Regulator voltage scaling output selection)

此位用来控制内部主调压器的输出电压，以便在器件未以最大频率工作时使性能与功耗实现平衡。

0：级别 2 模式

1：级别 1 模式（复位时的默认值）

位 13:10 保留，必须保持复位值。

位 9 **FPDS**：停止模式下 Flash 掉电 (Flash power-down in Stop mode)

将此位置 1 时，Flash 将在器件进入停止模式后掉电。这样可以降低停止模式的功耗，但会延长重新启动时间。

0：器件进入停止模式时 Flash 不掉电

1：器件进入停止模式时 Flash 掉电

位 8 **DBP**：禁止备份域写保护 (Disable backup domain write protection)

在复位状态下，RCC_BDCR 寄存器、RTC 寄存器（包括备份寄存器）以及 PWR_CSR 寄存器的 BRE 位均受到写访问保护。必须将此位置 1 才能使能对这些寄存器的写访问。

0：禁止对 RTC、RTC 备份寄存器和备份 SRAM 的访问

1：使能对 RTC、RTC 备份寄存器和备份 SRAM 的访问

位 7:5 **PLS[2:0]**：PVD 级别选择 (PVD level selection)

这些位由软件写入，用于选择电压检测器检测的电压阈值

000：2.0 V

001：2.1 V

010：2.3 V

011：2.5 V

100：2.6 V

101：2.7 V

110：2.8 V

111：2.9 V

注意：有关详细信息，请参见数据手册的电气特性。

位 4 **PVDE**：使能电源电压检测器 (Power voltage detector enable)
此位由软件置 1 和清零。
0：禁止 PVD
1：使能 PVD

位 3 **CSBF**：将待机标志清零 (Clear standby flag)
此位始终读为 0。
0：无操作
1：写 1 将 SBF 待机标志清零。

位 2 **CWUF**：将唤醒标志清零 (Clear wakeup flag)
此位始终读为 0。
0：无操作
1：**写 1 操作 2 个系统时钟周期后**将 WUF 唤醒标志清零

位 1 **PDDS**：深度睡眠掉电 (Power-down deepsleep)
此位由软件置 1 和清零。与 LPDS 位结合使用。
0：器件在 CPU 进入深度睡眠时进入停止模式。调压器状态取决于 LPDS 位。
1：器件在 CPU 进入深度睡眠时进入待机模式。

位 0 **LPDS**：深度睡眠低功耗 (Low-power deepsleep)
此位由软件置 1 和清零。与 PDDS 位结合使用。
0：停止模式下调压器开启
1：停止模式下调压器进入低功耗模式

5.4.2 用于STM32F42xxx 和 STM32F43xxx 的 PWR 电源控制寄存器 (PWR_CR)

PWR power control register

偏移地址：0x00

复位值：0x0000 C000（通过从待机模式唤醒进行复位）

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VOS		ADCDC1	Reserved			FPDS	DBP	PLS[2:0]			PVDE	CSBF	CWUF	PDDS	LPDS
rw	rw	rw				rw	rw	rw	rw	rw	rw	rc_w1	rc_w1	rw	rw

位 31:16 保留，必须保持复位值。

位 15:14 **VOS[1:0]**：调压器输出电压级别选择 (Regulator voltage scaling output selection)
这些位用来控制内部主调压器的输出电压，以便在器件未以最大频率工作时使性能与功耗实现平衡（有关详细信息，请参见 STM32F42xx 和 STM32F43xx 数据手册）。
只有在关闭 PLL 时才可以修改这些位。新的编程值只在 PLL 开启后才生效。PLL 关闭后将自动选择电压级别 3。
00：保留（选择级别 3 模式）
01：级别 3 模式
10：级别 2 模式
11：级别 1 模式（复位值）

位 13 **ADCDC1**：

0：无操作。

1：有关如何使用此位的详细信息，请参见 AN4073。

注意：仅当在 2.7 V 到 3.6 V 之间的电源电压范围内工作且关闭 Flash 预取指功能时，才可以设置此位。

位 12:10 保留，必须保持复位值。

位 9 **FPDS**：停止模式下 Flash 掉电 (Flash power-down in Stop mode)

将此位置 1 时，Flash 将在器件进入停止模式后掉电。这样可以降低停止模式的功耗，但会延长重新启动时间。

0：器件进入停止模式时 Flash 不掉电

1：器件进入停止模式时 Flash 掉电

位 8 **DBP**：禁止备份域写保护 (Disable backup domain write protection)

在复位状态下，RCC_BDCR 寄存器、RTC 寄存器（包括备份寄存器）以及 PWR_CSR 寄存器的 BRE 位均受到写访问保护。必须将此位置 1 才能使能对这些寄存器的写访问。

0：禁止对 RTC、RTC 备份寄存器和备份 SRAM 的访问

1：使能对 RTC、RTC 备份寄存器和备份 SRAM 的访问

位 7:5 **PLS[2:0]**：PVD 级别选择 (PVD level selection)

这些位由软件写入，用于选择电压检测器检测的电压阈值

000：2.0 V

001：2.1 V

010：2.3 V

011：2.5 V

100：2.6 V

101：2.7 V

110：2.8 V

111：2.9 V

注意：有关详细信息，请参见数据手册的电气特性。

位 4 **PVDE**：使能电源电压检测器 (Power voltage detector enable)

此位由软件置 1 和清零。

0：禁止 PVD

1：使能 PVD

位 3 **CSBF**：将待机标志清零 (Clear standby flag)

此位始终读为 0。

0：无操作

1：写 1 将 SBF 待机标志清零。

位 2 **CWUF**：将唤醒标志清零 (Clear wakeup flag)

此位始终读为 0。

0：无操作

1：**写 1 操作 2 个系统时钟周期后**将 WUF 唤醒标志清零

位 1 **PDDS**：深度睡眠掉电 (Power-down deepsleep)

此位由软件置 1 和清零。与 LPDS 位结合使用。

0：器件在 CPU 进入深度睡眠时进入停止模式。调压器状态取决于 LPDS 位。

1：器件在 CPU 进入深度睡眠时进入待机模式。

位 0 **LPDS**：深度睡眠低功耗 (Low-power deepsleep)

此位由软件置 1 和清零。与 PDDS 位结合使用。

0：停止模式下调压器开启

1：停止模式下调压器进入低功耗模式

5.4.3 PWR 电源控制/状态寄存器 (PWR_CSR)

PWR power control/status register

偏移地址：0x04

复位值：0x0000 0000（不通过从待机模式唤醒进行复位）

与标准的 APB 读操作相比，读取此寄存器需要更多的 APB 周期。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
Res.															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	VOS RDY	Reserved				BRE	EWUP	Reserved Res.				BRR	PVDO	SBF	WUF
	r					rw	rw					r	r	r	r

位 31:15 保留，必须保持复位值。

位 14 **VOSRDY**：调压器输出分级电压就绪标志 (Regulator voltage scaling output selection ready bit)

0：未就绪

1：就绪

位 13:10 保留，必须保持复位值。

位 9 **BRE**：使能备份调压器 (Backup regulator enable)

将此位置 1 时，使能备份调压器（用于在待机模式和 V_{BAT} 模式下保持备份 SRAM 内容）。如果 BRE 复位，备份调压器关闭。仍可使用备份 SRAM 但在待机模式和 V_{BAT} 模式中其内容将丢失。将此位置 1 后，应用程序必须等待备份调压器就绪标志 (BRR) 置 1，指示在待机模式和 V_{BAT} 模式下会保持写入 RAM 中的数据。

0：禁止备份调压器

1：使能备份调压器

注意：此位不会在器件从待机模式唤醒时复位，也不会通过系统复位或电源复位进行复位。

位 8 **EWUP**：使能 WKUP 引脚 (Enable WKUP pin)

此位由软件置 1 和清零。

0：WKUP 引脚用作通用 I/O。WKUP 引脚上的事件不会把器件从待机模式唤醒。

1：WKUP 用于从待机模式唤醒器件并被强制配置成输入下拉（WKUP 引脚出现上升沿时从待机模式唤醒系统）。

注意：此位通过系统复位进行复位。

位 7:4 保留，必须保持复位值。

位 3 **BRR**：备份调压器就绪 (Backup regulator ready)

由硬件置 1，用以指示备份调压器已就绪。

0：备份调压器未就绪

1：备份调压器就绪

注意：此位不会在器件从待机模式唤醒时复位，也不会通过系统复位或电源复位进行复位。

位 2 **PVDO**：PVD 输出 (PVD output)

此位通过硬件置 1 和清零。仅当通过 PVDE 位使能 PVD 时此位才有效。

0：V_{DD} 高于 PLS[2:0] 位选择的 PVD 阈值。

1：V_{DD} 低于 PLS[2:0] 位选择的 PVD 阈值。

注意：PVD 在进入待机模式时停止。因此，进入待机模式或执行复位后，此位等于 0，直到 PVDE 位置 1。

位 1 **SBF**：待机标志 (Standby flag)

此位由硬件置 1，清零则只能通过 POR/PDR（上电复位/掉电复位）或将 PWR_CR 寄存器中的 CSBF 位置 1 来实现。

0：器件未进入待机模式

1：器件在此次复位前进入待机模式

位 0 **WUF**：唤醒标志 (Wakeup flag)

此位由硬件置 1，清零则只能通过 POR/PDR（上电复位/掉电复位）或将 PWR_CR 寄存器中的 CWUF 位置 1 来实现。

0：未发生唤醒事件

1：收到唤醒事件，可能来自 WKUP 引脚、RTC 闹钟（闹钟 A 和闹钟 B）、RTC 入侵事件、RTC 时间戳事件或 RTC 唤醒事件。

注意：如果使能 WKUP 引脚（将 EWUP 位置 1）时 WKUP 引脚已为高电平，系统将检测到另一唤醒事件。

5.5 PWR 寄存器映射

下表对 PWR 寄存器进行了汇总。

表 24. STM32F405xx/07xx 和 STM32F415xx/17xx PWR——寄存器映射和复位值

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	**PWR_CR**	Reserved																	VOS	Reserved				FPDS	DBP	PLS[2:0]			PVDE	CSBF	CWUF	PDDS	LPDS
	Reset value																		1					0	0	0	0	0	0	0	0	0	0
0x004	**PWR_CSR**	Reserved																	VOSRDY	Reserved				BRE	EWUP	Reserved				BRR	PVDO	SBF	WUF
	Reset value																		0					0	0					0	0	0	0

表 25. STM32F42xxx 和 STM32F43xxx PWR——寄存器映射和复位值

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	**PWR_CR**	Reserved																VOS[1:0]		ADCDC1	Reserved			FPDS	DBP	PLS[2:0]			PVDE	CSBF	CWUF	PDDS	LPDS
	Reset value																	1	1	0				0	0	0	0	0	0	0	0	0	0
0x004	**PWR_CSR**	Reserved																	VOSRDY	Reserved				BRE	EWUP	Reserved				BRR	PVDO	SBF	WUF
	Reset value																		0					0	0					0	0	0	0

有关寄存器边界地址的信息，请参见*第 52 页的表 2*。

6 复位和时钟控制 (RCC)

6.1 复位

共有三种类型的复位，分别为系统复位、电源复位和备份域复位。

6.1.1 系统复位

除了时钟控制寄存器 CSR 中的复位标志和备份域中的寄存器外，系统复位会将其它全部寄存器都复位为复位值（请参见*图 4*）。

只要发生以下事件之一，就会产生系统复位：

1. NRST 引脚低电平（外部复位）
2. 窗口看门狗计数结束（WWDG 复位）
3. 独立看门狗计数结束（IWDG 复位）
4. 软件复位（SW 复位）（请参见*软件复位*）
5. 低功耗管理复位（请参见*低功耗管理复位*）

软件复位

可通过查看 *RCC 时钟控制和状态寄存器 (RCC_CSR)* 中的复位标志确定。

要对器件进行软件复位，必须将 Cortex™-M4F 应用中断和复位控制寄存器中的 SYSRESETREQ 位置 1。有关详细信息，请参见 Cortex™-M4F 技术参考手册。

低功耗管理复位

引发低功耗管理复位的方式有两种：

1. 进入待机模式时产生复位：

   此复位的使能方式是清零用户选项字节中的 nRST_STDBY 位。使能后，只要成功执行进入待机模式序列，器件就将复位，而非进入待机模式。
2. 进入停止模式时产生复位：

   此复位的使能方式是清零用户选项字节中的 nRST_STOP 位。使能后，只要成功执行进入停止模式序列，器件就将复位，而非进入停止模式。

有关用户选项字节的详细信息，请参见 STM32F40x 和 STM32F41x Flash 编程手册，该手册可从 ST 销售办事处获取，也可以从 ST 网站 www.st.com 获得文档信息。

6.1.2 电源复位

只要发生以下事件之一，就会产生电源复位：

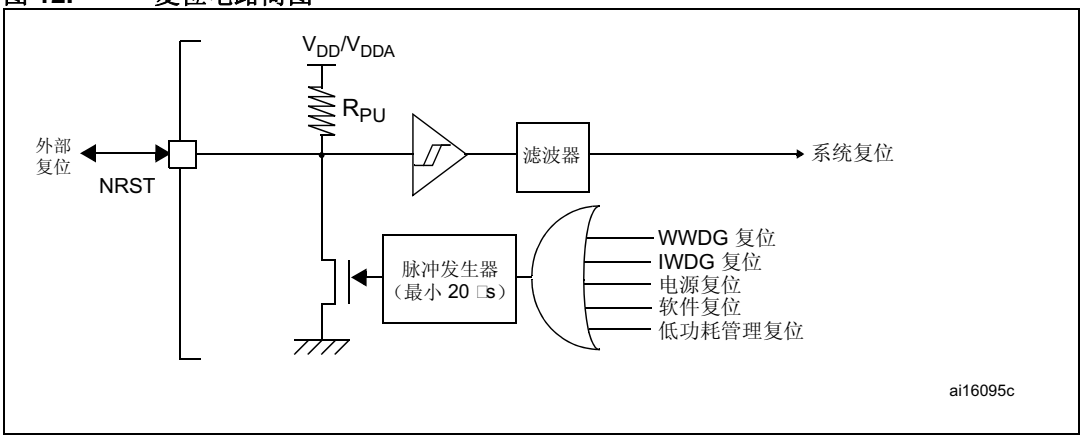
1. 上电/掉电复位（POR/PDR 复位）或欠压 (BOR) 复位
2. 在退出待机模式时

除备份域内的寄存器以外，电源复位会将其它全部寄存器设置为复位值（请参见*图 4*）

这些源均作用于 NRST 引脚，该引脚在复位过程中始终保持低电平。RESET 复位入口向量在存储器映射中固定在地址 0x0000_0004。

芯片内部的复位信号会在 NRST 引脚上输出。脉冲发生器用于保证最短复位脉冲持续时间，可确保每个内部复位源的复位脉冲都至少持续 20 µs。对于外部复位，在 NRST 引脚处于低电平时产生复位脉冲。

图 12. 复位电路简图



备份域具有两个特定的复位，这两个复位仅作用于备份域本身（请参见*图 4*）。

6.1.3 备份域复位

备份域复位会将所有 RTC 寄存器和 RCC_BDCR 寄存器复位为各自的复位值。BKPSRAM 不受此复位影响。BKPSRAM 的唯一复位方式是通过 Flash 接口将 Flash 保护等级从 1 切换到 0。

只要发生以下事件之一，就会产生备份域复位：

1. 软件复位，通过将 *RCC 备份域控制寄存器 (RCC_BDCR)* 中的 BDRST 位置 1 触发。
2. 在电源 V_{DD} 和 V_{BAT} 都已掉电后，其中任何一个又再上电。

6.2 时钟

可以使用三种不同的时钟源来驱动系统时钟 (SYSCLK)：

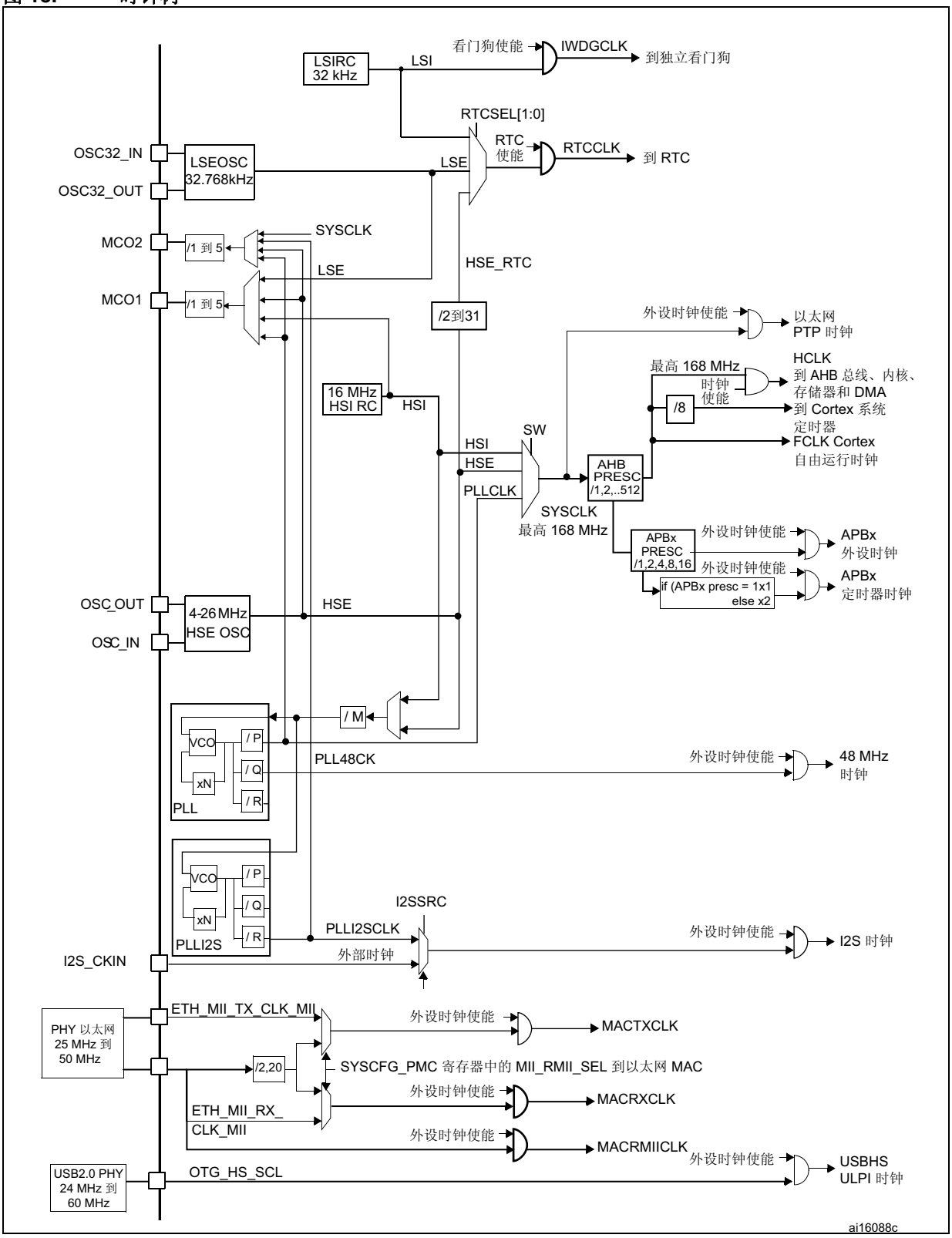
- HSI 振荡器时钟
- HSE 振荡器时钟
- 主 PLL (PLL) 时钟

器件具有以下两个次级时钟源：

- 32 kHz 低速内部 RC (LSI RC)，该 RC 用于驱动独立看门狗，也可选择提供给 RTC 用于停机/待机模式下的自动唤醒。
- 32.768 kHz 低速外部晶振（LSE 晶振），用于驱动 RTC 时钟 (RTCCLK)

对于每个时钟源来说，在未使用时都可单独打开或者关闭，以降低功耗。

图 13. 时钟树



1. 有关内部和外部时钟源特性的所有详细信息，请参见器件数据手册的电气特性部分。

时钟控制器为应用带来了高度的灵活性，用户在运行内核和外设时可选择使用外部晶振或者使用振荡器，既可采用最高的频率，也可为以太网、USB OTG FS 以及 HS、I2S 和 SDIO 等需要特定时钟的外设保证合适的频率。

可通过多个预分频器配置 AHB 频率、高速 APB (APB2) 和低速 APB (APB1)。AHB 域的最大频率为 168 MHz。高速 APB2 域的最大允许频率为 84 MHz。低速 APB1 域的最大允许频率为 42 MHz。

除以下时钟外，所有外设时钟均由系统时钟 (SYSCLK) 提供：

- 来自于特定 PLL 输出 (PLL48CLK) 的 USB OTG FS 时钟 (48 MHz)、基于模拟技术的随机数发生器 (RNG) 时钟 (≤ 48 MHz) 和 SDIO 时钟 (≤ 48 MHz)。
- I2S 时钟

  要实现高品质的音频性能，可通过特定的 PLL (PLLI2S) 或映射到 I2S_CKIN 引脚的外部时钟提供 I2S 时钟。有关 I2S 时钟频率和精度的详细信息，请参见*第 27.4.4 节：时钟发生器*。
- 由外部 PHY 提供的 USB OTG HS (60 MHz) 时钟
- 由外部 PHY 提供的以太网 MAC 时钟（TX、RX 和 RMII）。有关以太网配置的更多信息，请参见以太网外设说明中的*第 29.4.4 节：MII/RMII 选择*。当使用以太网时，AHB 时钟频率至少应为 25 MHz。

RCC 向 Cortex 系统定时器 (SysTick) 馈送 8 分频的 AHB 时钟 (HCLK)。SysTick 可使用此时钟作为时钟源，也可使用 HCLK 作为时钟源，具体可在 SysTick 控制和状态寄存器中配置。

STM32F405xx/07xx 和 STM32F415xx/17xx 的定时器时钟频率由硬件自动设置。分为两种情况：

1. 如果 APB 预分频器为 1，定时器时钟频率等于 APB 域的频率。
2. 否则，等于 APB 域的频率的两倍 (x2)。

STM32F42xxx 和 STM32F43xxx 的定时器时钟频率由硬件自动设置。根据 RCC_CFGR 寄存器中 TIMPRE 位的取值，共分为两种情况：

- 如果 RCC_DKCFGR 寄存器的 TIMPRE 位清 0：

  如果 APB 预分频器分频系数是 1，则定时器时钟频率 (TIMxCLK) 为 PCLKx。否则，定时器时钟频率将为 APB 域的频率的两倍：TIMxCLK = 2xPCLKx。
- 如果 RCC_DKCFGR 寄存器的 TIMPRE 位置 1：

  如果 APB 预分频器配置分频系数是 1、2 或 4，则定时器时钟频率 (TIMxCLK) 将设置为 HCLK。否则，定时器时钟频率将为 APB 域的频率的四倍：TIMxCLK = 4xPCLKx。

FCLK 充当 Cortex™-M4F 的自由运行时钟。有关详细信息，请参见 Cortex™-M4F 技术参考手册。

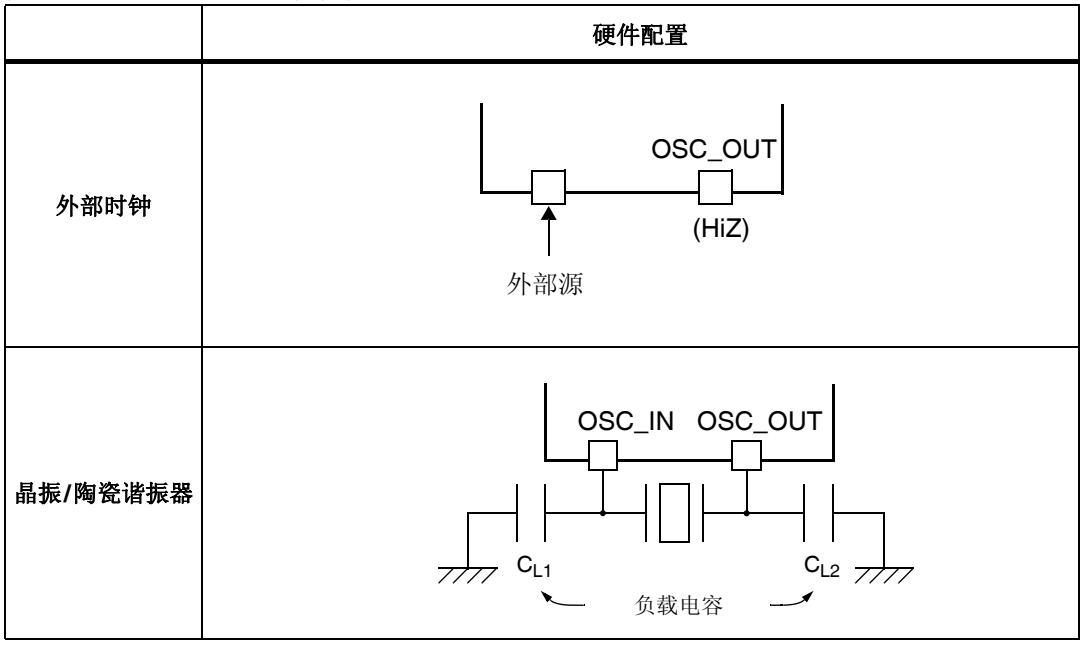
6.2.1 HSE 时钟

高速外部时钟信号 (HSE) 有 2 个时钟源：

- HSE 外部晶振/陶瓷谐振器
- HSE 外部用户时钟

谐振器和负载电容必须尽可能地靠近振荡器的引脚，以尽量减小输出失真和起振稳定时间。负载电容值必须根据所选振荡器的不同做适当调整。

图 14. HSE/LSE 时钟源



外部源（HSE 旁路）

在此模式下，必须提供外部时钟源。此模式通过将 *RCC 时钟控制寄存器 (RCC_CR)* 中的 HSEBYP 和 HSEON 位置 1 进行选择。必须使用占空比约为 50% 的外部时钟信号（方波、正弦波或三角波）来驱动 OSC_IN 引脚，同时 OSC_OUT 引脚应保持为高阻态 (hi-Z)。请参见*图 14*。

外部晶振/陶瓷谐振器（HSE 晶振）

HSE 的特点是精度非常高。

相关的硬件配置如*图 14* 所示。有关详细信息，请参见*数据手册*的电气特性部分。

RCC 时钟控制寄存器 (RCC_CR) 中的 HSERDY 标志指示高速外部振荡器是否稳定。在启动时，硬件将此位置 1 后，此时钟才可以使用。如在 *RCC 时钟中断寄存器 (RCC_CIR)* 中使能中断，则可产生中断。

HSE 晶振可通过 *RCC 时钟控制寄存器 (RCC_CR)* 中的 HSEON 位打开或关闭。

6.2.2 HSI 时钟

HSI 时钟信号由内部 16 MHz RC 振荡器生成，可直接用作系统时钟，或者用作 PLL 输入。

HSI RC 振荡器的优点是成本较低（无需使用外部组件）。此外，其启动速度也要比 HSE 晶振块，但即使校准后，其精度也不及外部晶振或陶瓷谐振器。

校准

因为生产工艺不同，不同芯片的 RC 振荡器频率也不同，因此 ST 会对每个器件进行出厂校准，达到 T_A= 25 °C 时 1% 的精度。

复位后，工厂校准值将加载到 *RCC 时钟控制寄存器 (RCC_CR)* 的 HSICAL[7:0] 位中。

如果应用受到电压或温度变化影响，则这可能也会影响到 RC 振荡器的速度。用户可通过 *RCC 时钟控制寄存器 (RCC_CR)* 中的 HSITRIM[4:0] 位对 HSI 频率进行微调。

RCC 时钟控制寄存器 (RCC_CR) 中的 HSIRDY 标志指示 HSI RC 是否稳定。在启动时，硬件将此位置 1 后，HSI 才可以使用。

HSI RC 可通过 *RCC 时钟控制寄存器 (RCC_CR)* 中的 HSION 位打开或关闭。

HSI 信号还可作为备份时钟源（辅助时钟）使用，以防 HSE 晶振发生故障。请参见*第 6.2.7 节：第 111 页的时钟安全系统 (CSS)*。

6.2.3 PLL 配置

STM32F4xx 器件具有两个 PLL：

- 主 PLL (PLL) 由 HSE 或 HSI 振荡器提供时钟信号，并具有两个不同的输出时钟：
  - 第一个输出用于生成高速系统时钟（最高达 168 MHz）
  - 第二个输出用于生成 USB OTG FS 的时钟 (48 MHz)、随机数发生器的时钟 (≤48 MHz) 和 SDIO 时钟 (≤ 48 MHz)。
- 专用 PLL (PLLI2S) 用于生成精确时钟，从而在 I2S 接口实现高品质音频性能。

由于在 PLL 使能后主 PLL 配置参数便不可更改，所以建议先对 PLL 进行配置，然后再使能（选择 HSI 或 HSE 振荡器作为 PLL 时钟源，并配置分频系数 M、N、P 和 Q）。

PLLI2S 使用与 PLL 相同的输入时钟（PLLM[5:0] 和 PLLSRC 位为两个 PLL 所共用）。但是，PLLI2S 具有专门的使能/禁止和分频系数（N 和 R）配置位。在 PLLI2S 使能后，配置参数便不能更改。

当进入停机和待机模式后，两个 PLL 将由硬件禁止；如将 HSE 或 PLL（由 HSE 提供时钟信号）用作系统时钟，则在 HSE 发生故障时，两个 PLL 也将由硬件禁止。*RCC PLL 配置寄存器 (RCC_PLLCFGR)* 和 *RCC 时钟配置寄存器 (RCC_CFGR)* 可分别用于配置 PLL 和 PLLI2S。

6.2.4 LSE 时钟

LSE 晶振是 32.768 kHz 低速外部 (LSE) 晶振或陶瓷谐振器，可作为实时时钟外设 (RTC) 的时钟源来提供时钟/日历或其它定时功能，具有功耗低且精度高的优点。

LSE 晶振通过 *RCC 备份域控制寄存器 (RCC_BDCR)* 中的 LSEON 位打开和关闭。

RCC 备份域控制寄存器 (RCC_BDCR) 中的 LSERDY 标志指示 LSE 晶振是否稳定。在启动时，硬件将此位置 1 后，LSE 晶振输出时钟信号才可以使用。如在 *RCC 时钟中断寄存器 (RCC_CIR)* 中使能中断，则可产生中断。

外部源（LSE 旁路）

在此模式下，必须提供外部时钟源，最高频率不超过 1 MHz。此模式通过将 *RCC 备份域控制寄存器 (RCC_BDCR)* 中的 LSEBYP 和 LSEON 位置 1 进行选择。必须使用占空比约为 50% 的外部时钟信号（方波、正弦波或三角波）来驱动 OSC32_IN 引脚，同时 OSC32_OUT 引脚应保持为高阻态 (Hi-Z)。请参见*图 14*。

6.2.5 LSI 时钟

LSI RC 可作为低功耗时钟源在停机和待机模式下保持运行，供独立看门狗 (IWDG) 和自动唤醒单元 (AWU) 使用。时钟频率在 32 kHz 左右。有关详细信息，请参见数据手册的电气特性部分。

LSI RC 可通过 *RCC 时钟控制和状态寄存器 (RCC_CSR)* 中的 LSION 位打开或关闭。

RCC 时钟控制和状态寄存器 (RCC_CSR) 中的 LSIRDY 标志指示低速内部振荡器是否稳定。在启动时，硬件将此位置 1 后，此时钟才可以使用。如在 *RCC 时钟中断寄存器 (RCC_CIR)* 中使能中断，则可产生中断。

6.2.6 系统时钟 (SYSCLK) 选择

在系统复位后，默认系统时钟为 HSI。在直接使用 HSI 或者通过 PLL 使用时钟源来作为系统时钟时，该时钟源无法停止。

只有在目标时钟源已就绪时（时钟在启动延迟或 PLL 锁相后稳定时），才可从一个时钟源切换到另一个。如果选择尚未就绪的时钟源，则切换在该时钟源就绪时才会进行。*RCC 时钟控制寄存器 (RCC_CR)* 中的状态位指示哪个（些）时钟已就绪，以及当前哪个时钟正充当系统时钟。

6.2.7 时钟安全系统 (CSS)

时钟安全系统可通过软件激活。激活后，时钟监测器将在 HSE 振荡器启动延迟后使能，并在此振荡器停止时被关闭。

如果 HSE 时钟发生故障，此振荡器将自动禁止，一个时钟故障事件将发送到高级控制定时器 TIM1 和 TIM8 的断路输入，并且同时还将生成一个中断来向软件通知此故障（时钟安全系统中断，CSSI），以使 MCU 能够执行救援操作。CSSI 与 Cortex™-M4F NMI（不可屏蔽中断）异常向量相链接。

注意： *当 CSS 使能后，如果 HSE 时钟偶发故障，则 CSS 将生成一个中断，进而促使 NMI 自动生成。NMI 将无限期执行，除非将 CSS 中断挂起位清零。因此，应用程序必须在 NMI ISR 中将 CSS 中断清零，具体方式为在时钟中断寄存器 (RCC_CIR) 中将 CSSC 位置 1。*

如果直接或间接使用 HSE 振荡器作为系统时钟（间接是指该振荡器直接用作 PLL 的输入时钟，并且该 PLL 时钟为系统时钟）并且检出故障，则系统时钟将切换到 HSI 振荡器并且 HSE 振荡器将被禁止。

如果 HSE 振荡器时钟是充当系统时钟的 PPL 的时钟源，则在发生故障时，PLL 也会被禁止。在此情况下，如果 PLLI2S 已使能，则在 HSE 发生故障时也会将其禁止。

6.2.8 RTC/AWU 时钟

一旦选定 RTCCLK 时钟源后，要想修改所做选择，只能复位电源域。

RTCCLK 时钟源可以是 HSE 1 MHz（HSE 由一个可编程的预分频器分频）、LSE 或者 LSI 时钟。选择方式是编程 *RCC 备份域控制寄存器 (RCC_BDCR)* 中的 RTCSEL[1:0] 位和 *RCC 时钟配置寄存器 (RCC_CFGR)* 中的 RTCPRE[4:0] 位。所做的选择只能通过复位备份域的方式修改。

如果选择 LSE 作为 RTC 时钟，则系统电源丢失时 RTC 仍将正常工作。如果选择 LSI 作为 AWU 时钟，则在系统电源丢失时将无法保证 AWU 的状态。如果 HSE 振荡器通过一个介于 2 和 31 之间的值进行分频，则在备用或系统电源丢失时将无法保证 RTC 的状态。

LSE 时钟位于备份域中，而 HSE 和 LSI 时钟则不是。因此：

- 如果选择 LSE 作为 RTC 时钟：
  - 只要 V_{BAT} 电源保持工作，即使 V_{DD} 电源关闭，RTC 仍可继续工作。
- 如果选择 LSI 作为自动唤醒单元 (AWU) 时钟：
  - 在 V_{DD} 电源掉电时，AWU 的状态将不能保证。有关 LSI 校准的详细信息，请参见 *第 6.2.5 节：第 111 页的LSI 时钟*。
- 如果使用 HSE 时钟作为 RTC 时钟：
  - 如果 V_{DD} 电源掉电或者内部调压器关闭（切断 1.2 V 域的供电），则 RTC 的状态将不能保证。

注意： *要在 APB1 时钟频率低于 RTC 时钟频率的七倍时 ($f_{APB1} < 7xf_{RTCLCK}$) 读取 RTC 日历寄存器，软件必须两次读取日历时间寄存器和日期寄存器。如果对 RTC_TR 的第二次读取访问得到的结果与第一次相同，则表明数据正确无误。否则必须执行第三次读取访问。*

6.2.9 看门狗时钟

如果独立看门狗 (IWDG) 已通过硬件选项字节或软件设置的方式启动，则 LSI 振荡器将强制打开且不可禁止。在 LSI 振荡器稳定后，时钟将提供给 IWDG。

6.2.10 时钟输出功能

共有两个微控制器时钟输出 (MCO) 引脚：

- MCO1

  用户可通过可配置的预分配器（从 1 到 5）向 MCO1 引脚 (PA8) 输出四个不同的时钟源：
  - HSI 时钟
  - LSE 时钟
  - HSE 时钟
  - PLL 时钟

  所需的时钟源通过 *RCC 时钟配置寄存器 (RCC_CFGR)* 中的 MCO1PRE[2:0] 和 MCO1[1:0] 位选择。
- MCO2

  用户可通过可配置的预分配器（从 1 到 5）向 MCO2 引脚 (PC9) 输出四个不同的时钟源：
  - HSE 时钟
  - PLL 时钟
  - 系统时钟 (SYSCLK)
  - PLLI2S 时钟

  所需的时钟源通过 *RCC 时钟配置寄存器 (RCC_CFGR)* 中的 MCO2PRE[2:0] 和 MCO2 位选择。

对于不同的 MCO 引脚，必须将相应的 GPIO 端口在复用功能模式下进行设置。

MCO 输出时钟不得超过 100 MHz（最大 I/O 速度）。

6.2.11 基于 TIM5/TIM11 的内部/外部时钟测量

所有时钟源的频率都可通过对 TIM5 channel4 和 TIM11 channel1 的输入捕获进行间接测量，如*图 15* 和*图 15* 所示。

基于 TIM5 channel4 的内部/外部时钟测量

TIM5 具有一个输入复用器，可选择输入捕获是由 I/O 触发还是由内部时钟触发。此选择通过 TIM5_OR 寄存器的 TI4_RMP [1:0] 位执行。

将 LSE 连接到 channel4 输入捕获的主要目的是为了精确测量 HSI（这需要将 HSI 用作系统时钟源）。借助 LSE 信号连续边沿之间的 HSI 时钟计数数量，即可对内部时钟周期进行测量。利用 LSE 的高精度（通常为几十 ppm），用户能以同一分辨率测定时钟频率，并可通过对时钟源进行微调来补偿由生产工艺造成的和/或与温度和电压相关的频率偏差。

HSI 振荡器设有针对此目的的专用校准位，且支持用户访问。

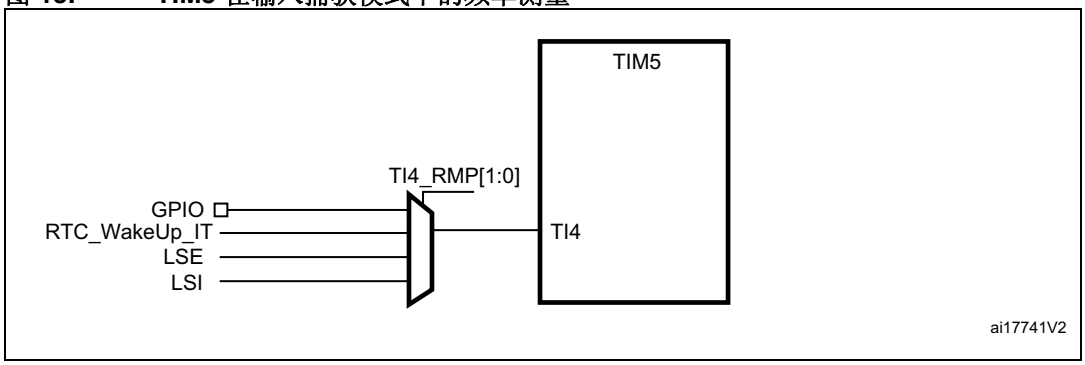
其基本原理是基于相对的测量（例如，HSI/LSE 比）：因此，精度与两个时钟源之比紧密相关。比率越大，测量效果越好。

同时也可测量 LSI 频率：这对于没有晶振的应用场合非常实用。超低功耗 LSI 振荡器具有较大的生产工艺偏差：通过测量该振荡器与 HSI 时钟源的比率，可以借助 HSI 精度为其测定频率。可通过此测量值实现更加精确的 RTC 时基超时（当使用 LSI 作为 RTC 时钟源时）和/或实现精度水平可以接受的 IWDG 超时。

LSI 频率通过以下步骤测量：

1. 使能 TIM5 定时器并将 channel4 配置为输入捕获模式。
2. 将 TIM5_OR 寄存器中的 TI4_RMP 位设置为 0x01，以在内部将 LSI 时钟连接到 TIM5 channel4 输入捕获来实现校准。
3. 通过 TIM5 捕获/比较 4 事件或中断测量 LSI 时钟频率。
4. 使用测得的 LSI 频率来按照所需的时基更新 RTC 的预分频器，并且/或者用其来计算 IWDG 超时。

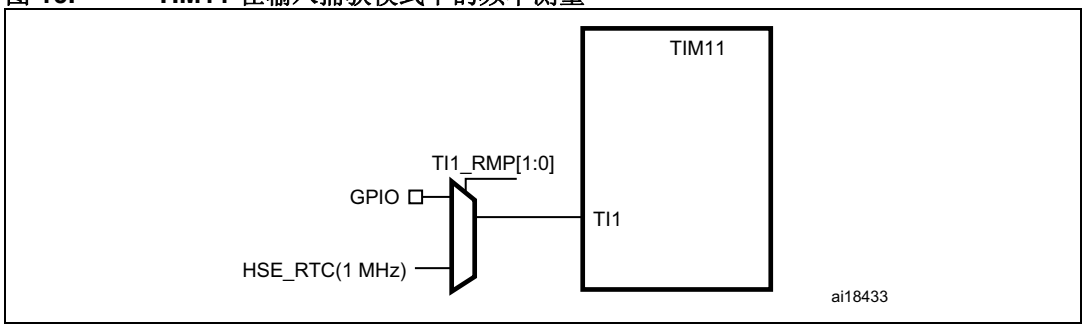
图 15. TIM5 在输入捕获模式下的频率测量



基于 TIM11 channel1 的内部/外部时钟测量

TIM11 具有一个输入复用器，可选择输入捕获是由 I/O 触发还是由内部时钟触发。此选择通过 TIM11_OR 寄存器的 TI1_RMP [1:0] 位执行。HSE_RTC 时钟（由一个可编程预分频器分频的 HSE）连接到通道 1 输入捕获，以粗略指示外部晶振频率。这要求 HSI 为系统时钟源。此功能非常实用，例如可借此测定谐波频率或分谐波频率（–50/+100% 偏差），进而确保符合 IEC 60730/IEC 61335 标准。

图 16. TIM11 在输入捕获模式下的频率测量



6.3 RCC 寄存器

有关寄存器说明中使用的缩写，请参见*第 1.1 节：寄存器相关缩写词列表*。

6.3.1 RCC 时钟控制寄存器 (RCC_CR)

RCC clock control register

偏移地址：0x00

复位值：0x0000 XX83，其中 X 未定义。

访问：无等待周期，按字、半字和字节访问

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved				PLLI2S RDY	PLLI2S ON	PLLRDY	PLLON	Reserved				CSS ON	HSE BYP	HSE RDY	HSE ON
				r	rw	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.	HSI RDY	HSION
r	r	r	r	r	r	r	r	rw	rw	rw	rw	rw		r	rw

位 31:28 保留，必须保持复位值。

位 27 **PLLI2SRDY**：PLLI2S 时钟就绪标志 (PLLI2S clock ready flag)
由硬件置 1，用以指示 PLLI2S 已锁定。
0：PLLI2S 未锁定
1：PLLI2S 已锁定

位 26 **PLLI2SON**：PLLI2S 使能 (PLLI2S enable)
由软件置 1 和清零，用于使能 PLLI2S。
当进入停机或待机模式时由硬件清零。
0：PLLI2S 关闭
1：PLLI2S 开启

位 25 **PLLRDY**：主 PLL (PLL) 时钟就绪标志 (Main PLL (PLL) clock ready flag)
由硬件置 1，用以指示 PLL 已锁定。
0：PLL 未锁定
1：PLL 已锁定

位 24 **PLLON**：主 PLL (PLL) 使能 (Main PLL (PLL) enable)

由软件置 1 和清零，用于使能 PLL。

当进入停机或待机模式时由硬件清零。如果 PLL 时钟用作系统时钟，则此位不可清零。

0：PLL 关闭

1：PLL 开启

位 23:20 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 和清零，用于使能时钟安全系统。当 CSSON 置位时，时钟监测器将在 HSE 振荡器就绪时由硬件使能，并在检出振荡器故障时由硬件禁止。

0：时钟安全系统关闭（时钟监测器关闭）

1：时钟安全系统打开（如果 HSE 振荡器稳定，则时钟监测器打开；如果不稳定，则关闭）

位 18 **HSEBYP**：HSE 时钟旁路 (HSE clock bypass)

由软件置 1 和清零，用于用外部时钟旁路振荡器。外部时钟必须通过 HSEON 位使能才能为器件使用。

HSEBYP 只有在 HSE 振荡器已禁止的情况下才可写入。

0：不旁路 HSE 振荡器

1：外部时钟旁路 HSE 振荡器

位 17 **HSERDY**：HSE 时钟就绪标志 (HSE clock ready flag)

由硬件置 1，用以指示 HSE 振荡器已稳定。在将 HSEON 位清零后，HSERDY 将在 6 个 HSE 振荡器时钟周期后转为低电平。

0：HSE 振荡器未就绪

1：HSE 振荡器已就绪

位 16 **HSEON**：HSE 时钟使能 (HSE clock enable)

由软件置 1 和清零。

由硬件清零，用于在进入停机或待机模式时停止 HSE 振荡器。如果 HSE 振荡器直接或间接用于作为系统时钟，则此位不可复位。

0：HSE 振荡器关闭

1：HSE 振荡器打开

位 15:8 **HSICAL[7:0]**：内部高速时钟校准 (Internal high-speed clock calibration)

这些位在启动时自动初始化。

位 7:3 **HSITRIM[4:0]**：内部高速时钟微调 (Internal high-speed clock trimming)

通过这些位，可在 HSICAL[7:0] 位基础上实现可由用户编程的微调值。可通过编程使其适应电压和温度的差异，使内部 HSI RC 的频率更为准确。

位 2 保留，必须保持复位值。

位 1 **HSIRDY**：内部高速时钟就绪标志 (Internal high-speed clock ready flag)

由硬件置 1，用以指示 HSI 振荡器已稳定。在将 HSION 位清零后，HSIRDY 将在 6 个 HSI 时钟周期后转为低电平。

0：HSI 振荡器未就绪

1：HSI 振荡器已就绪

位 0 **HSION**：内部高速时钟使能 (Internal high-speed clock enable)

由软件置 1 和清零。

由硬件置 1，用于在脱离停机或待机模式时或者在直接或间接用作系统时钟的 HSE 振荡器发生故障时强制 HSI 振荡器打开。如果 HSI 直接或间接用于作为系统时钟，则此位不可清零。

0：HSI 振荡器关闭

1：HSI 振荡器打开

6.3.2 RCC PLL 配置寄存器 (RCC_PLLCFGR)

RCC PLL configuration register

偏移地址：0x04

复位值：0x2400 3010

访问：无等待周期，按字、半字和字节访问。

此寄存器用于根据公式配置 PLL 时钟输出：

- $f_{(VCO\ 时钟)} = f_{(PLL\ 时钟输入)} \times (PLLN / PLLM)$
- $f_{(PLL\ 常规时钟输出)} = f_{(VCO\ 时钟)} / PLLP$
- $f_{(USB\ OTG\ FS, SDIO, RNG\ 时钟输出)} = f_{(VCO\ 时钟)} / PLLQ$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved				PLLQ3	PLLQ2	PLLQ1	PLLQ0	Reserved	PLLSRC	Reserved				PLLP1	PLLP0
				rw	rw	rw	rw		rw					rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	PLLN									PLLM5	PLLM4	PLLM3	PLLM2	PLLM1	PLLM0
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:28 保留，必须保持复位值。

位 27:24 **PLLQ**：主 PLL (PLL) 分频系数，适用于 USB OTG FS、SDIO 和随机数发生器时钟 (Main PLL (PLL) division factor for USB OTG FS, SDIO and random number generator clocks)

由软件置 1 或清零，用于控制 USB OTG FS 时钟、随机数发生器时钟和 SDIO 时钟的频率。这些位应仅在 PLL 已禁止时写入。

小心： 为使 USB OTG FS 能够正常工作，需要 48 MHz 的时钟。对于 SDIO 和随即数生成器，频率需要低于或等于 48 MHz 才可正常工作。

USB OTG FS 时钟频率 = VCO 频率 / PLLQ，并且 2 ≤ PLLQ ≤ 15

0000：PLLQ = 0，错误配置

0001：PLLQ = 1，错误配置

0010：PLLQ = 2

0011：PLLQ = 3

0100：PLLQ = 4

...

1111：PLLQ = 15

位 23 保留，必须保持复位值。

位 22 **PLLSRC**：主 PLL(PLL) 和音频 PLL (PLLI2S) 输入时钟源 (Main PLL(PLL) and audio PLL (PLLI2S) entry clock source)

由软件置 1 和清零，用于选择 PLL 和 PLLI2S 时钟源。此位只有在 PLL 和 PLLI2S 已禁止时才可写入。

0：选择 HSI 时钟作为 PLL 和 PLLI2S 时钟输入

1：选择 HSE 振荡器时钟作为 PLL 和 PLLI2S 时钟输入

位 21:18 保留，必须保持复位值。

位 17:16 **PLLP**：适用于主系统时钟的主 PLL (PLL) 分频系数 (Main PLL (PLL) division factor for main system clock)

由软件置 1 和清零，用于控制常规 PLL 输出时钟的频率。这些位只能在 PLL 已禁止时写入。

小心： 软件必须正确设置这些位，使其在此域中不超过 168 MHz。

PLL 输出时钟频率 = VCO 频率 / PLLP 并且 PLLP = 2、4、6 或 8

00：PLLP = 2
01：PLLP = 4
10：PLLP = 6
11：PLLP = 8

位 14:6 **PLLN**：适用于 VCO 的主 PLL (PLL) 倍频系数 (Main PLL (PLL) multiplication factor for VCO)

由软件置 1 和清零，用于控制 VCO 的倍频系数。这些位只能在 PLL 已禁止时写入。写入这些位时只允许使用半字和字访问。

小心： 软件必须正确设置这些位，确保 VCO 输出频率介于 192 和 432 MHz 之间。

VCO 输出频率 = VCO 输入频率 × PLLN 并且 192 ≤ PLLN ≤ 432

000000000：PLLN = 0，错误配置
000000001：PLLN = 1，错误配置
...
011000000：PLLN = 192
...
110110000：PLLN = 432
110110001：PLLN = 433，错误配置
...
111111111：PLLN = 511，错误配置

位 5:0 **PLLM**：主 PLL (PLL) 和音频 PLL (PLLI2S) 输入时钟的分频系数 (Division factor for the main PLL (PLL) and audio PLL (PLLI2S) input clock)

由软件置 1 和清零，用于在 VCO 之前对 PLL 和 PLLI2S 输入时钟进行分频。这些位只有在 PLL 和 PLLI2S 已禁止时才可写入。

小心： 软件必须正确设置这些位，确保 VCO 输入频率介于 1 和 2 MHz 之间。建议选择 2 MHz 的频率，以便限制 PLL 抖动。

VCO 输入频率 = PLL 输入时钟频率 / PLLM 并且 2 ≤ PLLM ≤ 63

000000：PLLM = 0，错误配置
000001：PLLM = 1，错误配置
000010：PLLM = 2
000011：PLLM = 3
000100：PLLM = 4
...
111110：PLLM = 62
111111：PLLM = 63

6.3.3 RCC 时钟配置寄存器 (RCC_CFGR)

RCC clock configuration register

偏移地址：0x08

复位值：0x0000 0000

访问：0 ≤ 等待周期 ≤ 2，按字、半字和字节访问

只有在时钟源切换期间进行访问时才会插入 1 或 2 个等待周期。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MCO2		MCO2 PRE[2:0]			MCO1 PRE[2:0]			I2SSCR	MCO1		RTCPRE[4:0]				
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
PPRE2[2:0]			PPRE1[2:0]			Reserved		HPRE[3:0]				SWS1	SWS0	SW1	SW0
rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	r	r	rw	rw

位 31:30 **MCO2[1:0]**：微控制器时钟输出 2 (Microcontroller clock output 2)

由软件置 1 和清零。时钟源选择可能会造成对 MCO2 的干扰。强烈建议仅在复位后但在使能外部振荡器和 PLL 之前来配置这些位。

00：选择系统时钟 (SYSCLK) 输出到 MCO2 引脚

01:选择 PLLI2S 时钟输出到 MCO2 引脚

10：选择 HSE 振荡器时钟输出到 MCO2 引脚

11：选择 PLL 时钟输出到 MCO2 引脚

位 27:29 **MCO2PRE**：MCO2 预分频器 (MCO2 prescaler)

由软件置 1 和清零，用于配置 MCO2 的预分频器。对此预分频器进行修改可能会对 MCO2 造成干扰。强烈建议仅在复位后且在使能外部振荡器和 PLL 之前进行此分频器的更改。

0xx：无分频

100：2 分频

101：3 分频

110：4 分频

111：5 分频

位 24:26 **MCO1PRE**：MCO1 预分频器 (MCO1 prescaler)

由软件置 1 和清零，用于配置 MCO1 的预分频器。对此预分频器进行修改可能会对 MCO1 造成干扰。强烈建议仅在复位后且在使能外部振荡器和 PLL 之前进行此分频器的更改。

0xx：无分频

100：2 分频

101：3 分频

110：4 分频

111：5 分频

位 23 **I2SSRC**：I2S 时钟选择 (I2S clock selection)

由软件置 1 和清零。通过此位可在 PLLI2S 时钟和外部时钟之间选择 I2S 时钟源。强烈建议仅在复位之后使能 I2S 模块之前对此位进行更改。

0：PLLI2S 时钟用作 I2S 时钟源

1：在 I2S_CKIN 引脚上映射的外部时钟用作 I2S 时钟源

位 22:21 **MCO1**：微控制器时钟输出 1 (Microcontroller clock output 1)

由软件置 1 和清零。时钟源选择可能会造成对 MCO1 的干扰。强烈建议仅在复位后且在使能外部振荡器和 PLL 之前来配置这些位。

00：选择 HSI 时钟输出到 MCO1 引脚
01：选择 LSE 振荡器输出到 MCO1 引脚
10：选择 HSE 振荡器时钟输出到 MCO1 引脚
11：选择 PLL 时钟输出到 MCO1 引脚

位 20:16 **RTCPRE**：适用于 RTC 时钟的 HSE 分频系数

由软件置 1 和清零，用于对 HSE 时钟输入时钟进行分频，进而为 RTC 生成 1 MHz 的时钟。

小心： 软件必须正确设置这些位，确保提供给 RTC 的时钟为 1 MHz。在选择 RTC 时钟源之前必须配置这些位。

00000：无时钟
00001：无时钟
00010：HSE/2
00011：HSE/3
00100：HSE/4
...
11110：HSE/30
11111：HSE/31

位 15:13 **PPRE2**：APB 高速预分频器 (APB2) (APB high-speed prescaler (APB2))

由软件置位和清零，用于控制 APB 高速时钟分频系数。

小心： 软件必须正确设置这些位，使其在此域中不超过 84 MHz。在 PPRE2 写入后，时钟将通过 1 AHB 到 16 AHB 周期新预分频系数进行分频。

0xx：AHB 时钟不分频
100：AHB 时钟 2 分频
101：AHB 时钟 4 分频
110：AHB 时钟 8 分频
111：AHB 时钟 16 分频

位 12:10 **PPRE1**：APB 低速预分频器 (APB1) (APB Low speed prescaler (APB1))

由软件置位和清零，用于控制 APB 低速时钟分频系数。

小心： 软件必须正确设置这些位，使其在此域中不超过 42 MHz。在 PPRE1 写入后，时钟将通过 1 AHB 到 16 AHB 周期新预分频系数进行分频。

0xx：AHB 时钟不分频
100：AHB 时钟 2 分频
101：AHB 时钟 4 分频
110：AHB 时钟 8 分频
111：AHB 时钟 16 分频

位 9:8 保留，必须保持复位值。

位 7:4 **HPRE**：AHB 预分频器 (AHB prescaler)

由软件置位和清零，用于控制 AHB 时钟分频系数。

小心： 在 HPRE 写入后，时钟将通过 1 AHB 到 16 AHB 周期新预分频系数进行分频。

小心： 当使用以太网时，AHB 时钟频率必须至少为 25 MHz。

0xxx：系统时钟不分频
1000：系统时钟 2 分频
1001：系统时钟 4 分频
1010：系统时钟 8 分频
1011：系统时钟 16 分频
1100：系统时钟 64 分频
1101：系统时钟 128 分频
1110：系统时钟 256 分频
1111：系统时钟 512 分频

位 3:2 **SWS**：系统时钟切换状态 (System clock switch status)

由硬件置 1 和清零，用于指示用作系统时钟的时钟源。
00：HSI 振荡器用作系统时钟
01：HSE 振荡器用作系统时钟
10：PLL 用作系统时钟
11：不适用

位 1:0 **SW**：系统时钟切换 (System clock switch)

由软件置 1 和清零，用于选择系统时钟源。
由硬件置 1，用于在退出停机或待机模式时或者在直接或间接用作系统时钟的 HSE 振荡器发生故障时强制 HSI 的选择。
00：选择 HSI 振荡器作为系统时钟
01：选择 HSE 振荡器作为系统时钟
10：选择 PLL 作为系统时钟
11：不允许

6.3.4 RCC 时钟中断寄存器 (RCC_CIR)

RCC clock interrupt register

偏移地址：0x0C

复位值：0x0000 0000

访问：无等待周期，按字、半字和字节访问

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved								CSSC	Reserved	PLLI2SRDYC	PLLRDYC	HSERDYC	HSIRDYC	LSERDYC	LSIRDYC
								w		w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		PLLI2SRDYIE	PLLRDYIE	HSERDYIE	HSIRDYIE	LSERDYIE	LSIRDYIE	CSSF	Reserved	PLLI2SRDYF	PLLRDYF	HSERDYF	HSIRDYF	LSERDYF	LSIRDYF
		rw	rw	rw	rw	rw	rw	r		r	r	r	r	r	r

位 31:24 保留，必须保持复位值。

位 23 **CSSC**：时钟安全系统中断清零 (Clock security system interrupt clear)
此位由软件置 1，用于将 CSSF 标志清零。
0：无操作
1：将 CSSF 标志清零

位 22 保留，必须保持复位值。

位 21 **PLLI2SRDYC**：PLLI2S 就绪中断清零 (PLLI2S ready interrupt clear)
此位由软件置 1，用于将 PLLI2SRDYF 标志清零。
0：无操作
1：清零 PLLI2SRDYF

位 20 **PLLRDYC**：主 PLL(PLL) 就绪中断清零 (Main PLL(PLL) ready interrupt clear)
此位由软件置 1，用于将 PLLRDYF 标志清零。
0：无操作
1：清零 PLLRDYF

位 19 **HSERDYC**：HSE 就绪中断清零 (HSE ready interrupt clear)
此位由软件置 1，用于将 HSERDYF 标志清零。
0：无操作
1：清零 HSERDYF

位 18 **HSIRDYC**：HSI 就绪中断清零 (HSI ready interrupt clear)
此位由软件置 1，用于将 HSIRDYF 标志清零。
0：无操作
1：清零 HSIRDYF

位 17 **LSERDYC**：LSE 就绪中断清零 (LSE ready interrupt clear)
此位由软件置 1，用于将 LSERDYF 标志清零。
0：无操作
1：清零 LSERDYF

位 16 **LSIRDYC**：LSI 就绪中断清零 (LSI ready interrupt clear)
此位由软件置 1，用于将 LSIRDYF 标志清零。
0：无操作
1：清零 LSIRDYF

位 15:12 保留，必须保持复位值。

位 13 **PLLI2SRDYIE**：PLLI2S 就绪中断使能 (PLLI2S ready interrupt enable)
由软件置 1 和清零，用于使能/禁止由 PLLI2S 锁定引起的中断。
0：禁止 PLLI2S 锁定中断
1：使能 PLLI2S 锁定中断

位 12 **PLLRDYIE**：主 PLL (PLL) 就绪中断使能 (Main PLL (PLL) ready interrupt enable)
由软件置 1 和清零，用于使能/禁止由 PLL 锁定引起的中断。
0：禁止 PLL 锁定中断
1：使能 PLL 锁定中断

位 11 **HSERDYIE**：HSE 就绪中断使能 (HSE ready interrupt enable)
由软件置 1 和清零，用于使能/禁止由 HSE 振荡器稳定所引起的中断。
0：禁止 HSE 就绪中断
1：使能 HSE 就绪中断

位 10 **HSIRDYIE**：HSI 就绪中断使能 (HSI ready interrupt enable)
由软件置 1 和清零，用于使能/禁止由 HSI 振荡器稳定所引起的中断。
0：禁止 HSI 就绪中断
1：使能 HSI 就绪中断

位 9 **LSERDYIE**：LSE 就绪中断使能 (LSE ready interrupt enable)
由软件置 1 和清零，用于使能/禁止由 LSE 振荡器稳定所引起的中断。
0：禁止 LSE 就绪中断
1：使能 LSE 就绪中断

位 8 **LSIRDYIE**：LSI 就绪中断使能 (LSI ready interrupt enable)
由软件置 1 和清零，用于使能/禁止由 LSI 振荡器稳定所引起的中断。
0：禁止 LSI 就绪中断
1：使能 LSI 就绪中断

位 7 **CSSF**：时钟安全系统中断标志 (Clock security system interrupt flag)
当在 HSE 振荡器中检出故障时由硬件置 1。
CSSC 位置 1 时由软件清零。
0：当前未因 HSE 时钟故障而引起时钟安全中断
1：因 HSE 时钟故障而引起时钟安全中断

位 6 保留，必须保持复位值。

位 5 **PLLI2SRDYF**：PLLI2S 就绪中断标志 (PLLI2S ready interrupt flag)
当 PLLI2S 锁定并且 PLLI2SRDYDIE 置 1 时由硬件置 1。
PLLRI2SDYC 位置 1 时由软件清零。
0：当前未因 PLLI2S 锁定而引起时钟就绪中断
1：因 PLLI2S 锁定而引起时钟就绪中断

位 4 **PLLRDYF**：主 PLL (PLL) 就绪中断标志 (Main PLL (PLL) ready interrupt flag)
当 PLL 锁定并且 PLLRDYDIE 置 1 时由硬件置 1。
PLLRDYC 位置 1 时由软件清零。
0：当前未因 PLL 锁定而引起时钟就绪中断
1：因 PLL 锁定而引起时钟就绪中断

位 3 **HSERDYF**：HSE 就绪中断标志 (HSE ready interrupt flag)
当外部高速时钟变为稳定且 HSERDYDIE 置 1 时由硬件置 1。
HSERDYC 位置 1 时由软件清零。
0：当前未因 HSE 振荡器引起时钟就绪中断
1：因 HSE 振荡器引起时钟就绪中断

位 2 **HSIRDYF**：HSI 就绪中断标志 (HSI ready interrupt flag)
当内部高速时钟变为稳定且 HSIRDYDIE 置 1 时由硬件置 1。
HSIRDYC 位置 1 时由软件清零。
0：当前未因 HSI 振荡器引起时钟就绪中断
1：因 HSI 振荡器引起时钟就绪中断

位 1 **LSERDYF**：LSE 就绪中断标志 (LSE ready interrupt flag)
当外部低速时钟变为稳定且 LSERDYDIE 置 1 时由硬件置 1。
LSERDYC 位置 1 时由软件清零。
0：当前未因 LSE 振荡器引起时钟就绪中断
1：因 LSE 振荡器引起时钟就绪中断

位 0 **LSIRDYF**：LSI 就绪中断标志 (LSI ready interrupt flag)
当内部低速时钟变为稳定且 LSIRDYDIE 置 1 时由硬件置 1。
LSIRDYC 位置 1 时由软件清零。
0：当前未因 LSI 振荡器引起时钟就绪中断
1：因 LSI 振荡器引起时钟就绪中断

6.3.5 RCC AHB1 外设复位寄存器 (RCC_AHB1RSTR)

RCC AHB1 peripheral reset register

偏移地址：0x10

复位值：0x0000 0000

访问：无等待周期，按字、半字和字节访问。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved		OTGHSRST	Reserved			ETHMACRST	Reserved		DMA2RST	DMA1RST	Reserved				
		rw				rw			rw	rw					

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			CRCRST	Reserved			GPIOIRST	GPIOHRST	GPIOGGRST	GPIOFRST	GPIOERST	GPIODRST	GPIOCRST	GPIOBRST	GPIOARST
			rw				rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:30 保留，必须保持复位值。

位 29 **OTGHSRST**：USB OTG HS 模块复位 (USB OTG HS module reset)
由软件置 1 和清零。
0：不复位 USB OTG HS 模块
1：复位 USB OTG HS 模块

位 28:26 保留，必须保持复位值。

位 25 **ETHMACRST**：以太网 MAC 复位 (Ethernet MAC reset)
由软件置 1 和清零。
0：不复位以太网 MAC
1：复位以太网 MAC

位 24:23 保留，必须保持复位值。

位 22 **DMA2RST**：DMA2 复位 (DMA2 reset)
由软件置 1 和清零。
0：不复位 DMA2
1：复位 DMA2

位 21 **DMA1RST**：DMA1 复位 (DMA1 reset)
由软件置 1 和清零。
0：不复位 DMA1
1：复位 DMA1

位 20:13 保留，必须保持复位值。

位 12 **CRCRST**：CRC 复位 (CRC reset)
由软件置 1 和清零。
0：不复位 CRC
1：复位 CRC

位 11:9 保留，必须保持复位值。

位 8 **GPIOIRST**：IO 端口 I 复位 (IO port I reset)
由软件置 1 和清零。
0：不复位 IO 端口 I
1：复位 IO 端口 I

位 7 **GPIOHRST**：IO 端口 H 复位 (IO port H reset)
由软件置 1 和清零。
0：不复位 IO 端口 H
1：复位 IO 端口 H

位 6 **GPIOGRST**：IO 端口 G 复位 (IO port G reset)
由软件置 1 和清零。
0：不复位 IO 端口 G
1：复位 IO 端口 G

位 5 **GPIOFRST**：IO 端口 F 复位 (IO port F reset)
由软件置 1 和清零。
0：不复位 IO 端口 F
1：复位 IO 端口 F

位 4 **GPIOERST**：IO 端口 E 复位 (IO port E reset)
由软件置 1 和清零。
0：不复位 IO 端口 E
1：复位 IO 端口 E

位 3 **GPIODRST**：IO 端口 D 复位 (IO port D reset)
由软件置 1 和清零。
0：不复位 IO 端口 D
1：复位 IO 端口 D

位 2 **GPIOCRST**：IO 端口 C 复位 (IO port C reset)
由软件置 1 和清零。
0：不复位 IO 端口 C
1：复位 IO 端口 C

位 1 **GPIOBRST**：IO 端口 B 复位 (IO port B reset)
由软件置 1 和清零。
0：不复位 IO 端口 B
1：复位 IO 端口 B

位 0 **GPIOARST**：IO 端口 A 复位 (IO port A reset)
由软件置 1 和清零。
0：不复位 IO 端口 A
1：复位 IO 端口 A

6.3.6 RCC AHB2 外设复位寄存器 (RCC_AHB2RSTR)

RCC AHB2 peripheral reset register

偏移地址：0x14

复位值：0x0000 0000

访问：无等待周期，按字、半字和字节访问

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								OTGFS RST	RNG RST	HASH RST	CRYP RST	Reserved			DCMI RST
								rw	rw	rw	rw				rw

位 31:8 保留，必须保持复位值。

位 7 **OTGFSRST**：USB OTG FS 模块复位 (USB OTG FS module reset)
由软件置 1 和清零。
0：不复位 USB OTG FS 模块
1：复位 USB OTG FS 模块

位 6 **RNGRST**：随机数发生器模块复位 (Random number generator module reset)
由软件置 1 和清零。
0：不复位随机数发生器模块
1：复位随机数发生器模块

位 5 **HASHRST**：散列模块复位 (Hash module reset)
由软件置 1 和清零。
0：不复位散列模块
1：复位散列模块

位 4 **CRYPRST**：加密模块复位 (Cryptographic module reset)
由软件置 1 和清零。
0：不复位加密模块
1：复位加密模块

位 3:1 保留，必须保持复位值。

位 0 **DCMIRST**：摄像头接口复位 (Camera interface reset)
由软件置 1 和清零。
0：不复位摄像头接口
1：复位摄像头接口

6.3.7 RCC AHB3 外设复位寄存器 (RCC_AHB3RSTR)

RCC AHB3 peripheral reset register

偏移地址：0x18

复位值：0x0000 0000

访问：无等待周期，按字、半字和字节访问。

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

位 31:1 保留，必须保持复位值。

位 0 **FSMCRST**：灵活的静态存储控制器模块复位 (Flexible static memory controller module reset)
由软件置 1 和清零。
0：不复位 FSMC 模块
1：复位 FSMC 模块

6.3.8 用于 STM32F405xx/07xx 和 STM32F415xx/17xx 的 RCC APB1 外设复位寄存器 (RCC_APB1RSTR)

RCC APB1 peripheral reset register

偏移地址：0x20

复位值：0x0000 0000

访问：无等待周期，按字、半字和字节访问。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved		DACRST	PWR RST	Reser-ved	CAN2 RST	CAN1 RST	Reser-ved	I2C3 RST	I2C2 RST	I2C1 RST	UART5 RST	UART4 RST	UART3 RST	UART2 RST	Reser-ved
		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
SPI3 RST	SPI2 RST	Reserved		WWDG RST	Reserved		TIM14 RST	TIM13 RST	TIM12 RST	TIM7 RST	TIM6 RST	TIM5 RST	TIM4 RST	TIM3 RST	TIM2 RST
rw	rw			rw			rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:30 保留，必须保持复位值。

位 29 **DACRST**：DAC 复位 (DAC reset)
由软件置 1 和清零。
0：不复位 DAC 接口
1：复位 DAC 接口

位 28 **PWRRST**：电源接口复位 (Power interface reset)
由软件置 1 和清零。
0：不复位电源接口
1：复位电源接口

位 27 保留，必须保持复位值。

位 26 **CAN2RST**：CAN2 复位 (CAN2 reset)
由软件置 1 和清零。
0：不复位 CAN2
1：复位 CAN2

位 25 **CAN1RST**：CAN1 复位 (CAN1 reset)
由软件置 1 和清零。
0：不复位 CAN1
1：复位 CAN1

位 24 保留，必须保持复位值。

位 23 **I2C3RST**：I2C3 复位 (I2C3 reset)
由软件置 1 和清零。
0：不复位 I2C3
1：复位 I2C3

位 22 **I2C2RST**：I2C2 复位 (I2C2 reset)
由软件置 1 和清零。
0：不复位 I2C2
1：复位 I2C2

位 21 **I2C1RST**：I2C1 复位 (I2C1 reset)
由软件置 1 和清零。
0：不复位 I2C1
1：复位 I2C1

位 20 **UART5RST**：UART5 复位 (UART5 reset)
由软件置 1 和清零。
0：不复位 UART5
1：复位 UART5

位 19 **UART4RST**：USART4 复位 (USART4 reset)
由软件置 1 和清零。
0：不复位 UART4
1：复位 UART4

位 18 **USART3RST**：USART3 复位 (USART3 reset)
由软件置 1 和清零。
0：不复位 USART3
1：复位 USART3

位 17 **USART2RST**：USART2 复位 (USART2 reset)
由软件置 1 和清零。
0：不复位 USART2
1：复位 USART2

位 16 保留，必须保持复位值。

位 15 **SPI3RST**：SPI3 复位 (SPI3 reset)
由软件置 1 和清零。
0：不复位 SPI3
1：复位 SPI3

位 14 **SPI2RST**：SPI2 复位 (SPI2 reset)
由软件置 1 和清零。
0：不复位 SPI2
1：复位 SPI2

位 13:12 保留，必须保持复位值。

位 11 **WWDGRST**：窗口看门狗复位 (Window watchdog reset)
由软件置 1 和清零。
0：不复位窗口看门狗
1：复位窗口看门狗

位 10:9 保留，必须保持复位值。

位 8 **TIM14RST**：TIM14 复位 (TIM14 reset)
由软件置 1 和清零。
0：不复位 TIM14
1：复位 TIM14

位 7 **TIM13RST**：TIM13 复位 (TIM13 reset)
由软件置 1 和清零。
0：不复位 TIM13
1：复位 TIM13

位 6 **TIM12RST**：TIM12 复位 (TIM12 reset)
由软件置 1 和清零。
0：不复位 TIM12
1：复位 TIM12

位 5 **TIM7RST**：TIM7 复位 (TIM7 reset)
由软件置 1 和清零。
0：不复位 TIM7
1：复位 TIM7

位 4 **TIM6RST**：TIM6 复位 (TIM6 reset)
由软件置 1 和清零。
0：不复位 TIM6
1：复位 TIM6

位 3 **TIM5RST**：TIM5 复位 (TIM5 reset)
由软件置 1 和清零。
0：不复位 TIM5
1：复位 TIM5

位 2 **TIM4RST**：TIM4 复位 (TIM4 reset)
由软件置 1 和清零。
0：不复位 TIM4
1：复位 TIM4

位 1 **TIM3RST**：TIM3 复位 (TIM3 reset)
由软件置 1 和清零。
0：不复位 TIM3
1：复位 TIM3

位 0 **TIM2RST**：TIM2 复位 (TIM2 reset)
由软件置 1 和清零。
0：不复位 TIM2
1：复位 TIM2

6.3.9 用于 STM32F42xxx 和 STM32F43xxx 的 RCC APB1 外设复位寄存器 (RCC_APB1RSTR)

RCC APB1 peripheral reset register

偏移地址：0x20

复位值：0x0000 0000

访问：无等待周期，按字、半字和字节访问。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UART8RST	UART7RST	DACRST	PWRRST	Reserved	CAN2RST	CAN1RST	Reserved	I2C3RST	I2C2RST	I2C1RST	UART5RST	UART4RST	UART3RST	UART2RST	Reserved
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	Reserved		WWDGRST	Reserved		TIM14RST	TIM13RST	TIM12RST	TIM7RST	TIM6RST	TIM5RST	TIM4RST	TIM3RST	TIM2RST
rw	rw			rw			rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31 **UART8RST**：UART8 复位 (UART8 reset)
由软件置 1 和清零。
0：不复位 UART8
1：复位 UART8

位 30 **UART7RST**：UART7 复位 (UART7 reset)
由软件置 1 和清零。
0：不复位 UART7
1：复位 UART7

位 29 **DACRST**：DAC 复位 (DAC reset)
由软件置 1 和清零。
0：不复位 DAC 接口
1：复位 DAC 接口

位 28 **PWRRST**：电源接口复位 (Power interface reset)
由软件置 1 和清零。
0：不复位电源接口
1：复位电源接口

位 27 保留，必须保持复位值。

位 26 **CAN2RST**：CAN2 复位 (CAN2 reset)
由软件置 1 和清零。
0：不复位 CAN2
1：复位 CAN2

位 25 **CAN1RST**：CAN1 复位 (CAN1 reset)
由软件置 1 和清零。
0：不复位 CAN1
1：复位 CAN1

位 24 保留，必须保持复位值。

位 23 **I2C3RST**：I2C3 复位 (I2C3 reset)
由软件置 1 和清零。
0：不复位 I2C3
1：复位 I2C3

位 22 **I2C2RST**：I2C2 复位 (I2C2 reset)
由软件置 1 和清零。
0：不复位 I2C2
1：复位 I2C2

位 21 **I2C1RST**：I2C1 复位 (I2C1 reset)
由软件置 1 和清零。
0：不复位 I2C1
1：复位 I2C1

位 20 **UART5RST**：UART5 复位 (UART5 reset)
由软件置 1 和清零。
0：不复位 UART5
1：复位 UART5

位 19 **UART4RST**：USART4 复位 (USART4 reset)
由软件置 1 和清零。
0：不复位 UART4
1：复位 UART4

位 18 **USART3RST**：USART3 复位 (USART3 reset)
由软件置 1 和清零。
0：不复位 USART3
1：复位 USART3

位 17 **USART2RST**：USART2 复位 (USART2 reset)
由软件置 1 和清零。
0：不复位 USART2
1：复位 USART2

位 16 保留，必须保持复位值。

位 15 **SPI3RST**：SPI3 复位 (SPI3 reset)
由软件置 1 和清零。
0：不复位 SPI3
1：复位 SPI3

位 14 **SPI2RST**：SPI2 复位 (SPI2 reset)
由软件置 1 和清零。
0：不复位 SPI2
1：复位 SPI2

位 13:12 保留，必须保持复位值。

位 11 **WWDGRST**：窗口看门狗复位 (Window watchdog reset)
由软件置 1 和清零。
0：不复位窗口看门狗
1：复位窗口看门狗

位 10:9 保留，必须保持复位值。

位 8 **TIM14RST**：TIM14 复位 (TIM14 reset)
由软件置 1 和清零。
0：不复位 TIM14
1：复位 TIM14

位 7 **TIM13RST**：TIM13 复位 (TIM13 reset)
由软件置 1 和清零。
0：不复位 TIM13
1：复位 TIM13

位 6 **TIM12RST**：TIM12 复位 (TIM12 reset)
由软件置 1 和清零。
0：不复位 TIM12
1：复位 TIM12

位 5 **TIM7RST**：TIM7 复位 (TIM7 reset)
由软件置 1 和清零。
0：不复位 TIM7
1：复位 TIM7

位 4 **TIM6RST**：TIM6 复位 (TIM6 reset)
由软件置 1 和清零。
0：不复位 TIM6
1：复位 TIM6

位 3 **TIM5RST**：TIM5 复位 (TIM5 reset)
由软件置 1 和清零。
0：不复位 TIM5
1：复位 TIM5

位 2 **TIM4RST**：TIM4 复位 (TIM4 reset)
由软件置 1 和清零。
0：不复位 TIM4
1：复位 TIM4

位 1 **TIM3RST**：TIM3 复位 (TIM3 reset)
由软件置 1 和清零。
0：不复位 TIM3
1：复位 TIM3

位 0 **TIM2RST**：TIM2 复位 (TIM2 reset)
由软件置 1 和清零。
0：不复位 TIM2
1：复位 TIM2

6.3.10 用于 STM32F405xx/07xx 和 STM32F415xx/17xx 的 RCC APB2 外设复位寄存器 (RCC_APB2RSTR)

RCC APB2 peripheral reset register

偏移地址：0x24

复位值：0x0000 0000

访问：无等待周期，按字、半字和字节访问。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved													TIM11 RST	TIM10 RST	TIM9 RST
													rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reser-ved	SYSCFG RST	Reser-ved	SPI1 RST	SDIO RST	Reserved		ADC RST	Reserved		USART6 RST	USART1 RST	Reserved		TIM8 RST	TIM1 RST
	rw		rw	rw			rw			rw	rw			rw	rw

位 31:19 保留，必须保持复位值。

位 18 **TIM11RST**：TIM11 复位 (TIM11 reset)
由软件置 1 和清零。
0：不复位 TIM11
1：复位 TIM14

位 17 **TIM10RST**：TIM10 复位 (TIM10 reset)
由软件置 1 和清零。
0：不复位 TIM10
1：复位 TIM10

位 16 **TIM9RST**：TIM9 复位 (TIM9 reset)
由软件置 1 和清零。
0：不复位 TIM9
1：复位 TIM9

位 15 保留，必须保持复位值。

位 14 **SYSCFGRST**：系统配置控制器复位 (System configuration controller reset)
由软件置 1 和清零。
0：不复位系统配置控制器
1：复位系统配置控制器

位 13 保留，必须保持复位值。

位 12 **SPI1RST**：SPI1 复位 (SPI1 reset)
由软件置 1 和清零。
0：不复位 SPI1
1：复位 SPI1

位 11 **SDIORST**：SDIO 复位 (SDIO reset)
由软件置 1 和清零。
0：不复位 SDIO 模块
1：复位 SDIO 模块

位 10:9 保留，必须保持复位值。

位 8 **ADCRST**：ADC 接口复位（所有 ADC 共用）(ADC interface reset (common to all ADCs))
由软件置 1 和清零。
0：不复位 ADC 接口
1：复位 ADC 接口

位 7:6 保留，必须保持复位值。

位 5 **USART6RST**：USART6 复位 (USART6 reset)
由软件置 1 和清零。
0：不复位 USART6
1：复位 USART6

位 4 **USART1RST**：USART1 复位 (USART1 reset)
由软件置 1 和清零。
0：不复位 USART1
1：复位 USART1

位 3:2 保留，必须保持复位值。

位 1 **TIM8RST**：TIM8 复位 (TIM8 reset)
由软件置 1 和清零。
0：不复位 TIM8
1：复位 TIM8

位 0 **TIM1RST**：TIM1 复位 (TIM1 reset)
由软件置 1 和清零。
0：不复位 TIM1
1：复位 TIM1

6.3.11 用于 STM32F42xxx 和 STM32F43xxx 的 RCC APB2 外设复位寄存器 (RCC_APB2RSTR)

RCC APB2 peripheral reset register

偏移地址：0x24

复位值：0x0000 0000

访问：无等待周期，按字、半字和字节访问。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved										SPI6 RST	SPI5 RST	Res.	TIM11 RST	TIM10 RST	TIM9 RST
										rw	rw		rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reser-ved	SYSCFG RST	SPI4 RST	SPI1 RST	SDIO RST	Reserved		ADC RST	Reserved		USART6 RST	USART1 RST	Reserved		TIM8 RST	TIM1 RST
	rw	rw	rw	rw			rw			rw	rw			rw	rw

位 31:22 保留，必须保持复位值。

位 21 **SPI6RST**：SPI6 复位 (SPI6 reset)
由软件置 1 和清零。
0：不复位 SPI6
1：复位 SPI6

位 20 **SPI5RST**：SPI5 复位 (SPI5 reset)
由软件置 1 和清零。
0：不复位 SPI5
1：复位 SPI5

位 19 保留，必须保持复位值。

位 18 **TIM11RST**：TIM11 复位 (TIM11 reset)
由软件置 1 和清零。
0：不复位 TIM11
1：复位 TIM14

位 17 **TIM10RST**：TIM10 复位 (TIM10 reset)
由软件置 1 和清零。
0：不复位 TIM10
1：复位 TIM10

位 16 **TIM9RST**：TIM9 复位 (TIM9 reset)
由软件置 1 和清零。
0：不复位 TIM9
1：复位 TIM9

位 15 保留，必须保持复位值。

位 14 **SYSCFGRST**：系统配置控制器复位 (System configuration controller reset)
由软件置 1 和清零。
0：不复位系统配置控制器
1：复位系统配置控制器

位 13 **SPI4RST**：SPI4 复位 (SPI4 reset)
由软件置 1 和清零。
0：不复位 SPI4
1：复位 SPI4

位 12 **SPI1RST**：SPI1 复位 (SPI1 reset)
由软件置 1 和清零。
0：不复位 SPI1
1：复位 SPI1

位 11 **SDIORST**：SDIO 复位 (SDIO reset)
由软件置 1 和清零。
0：不复位 SDIO 模块
1：复位 SDIO 模块

位 10:9 保留，必须保持复位值。

位 8 **ADCRST**：ADC 接口复位（所有 ADC 共用）(ADC interface reset (common to all ADCs))
由软件置 1 和清零。
0：不复位 ADC 接口
1：复位 ADC 接口

位 7:6 保留，必须保持复位值。

位 5 **USART6RST**：USART6 复位 (USART6 reset)
由软件置 1 和清零。
0：不复位 USART6
1：复位 USART6

位 4 **USART1RST**：USART1 复位 (USART1 reset)
由软件置 1 和清零。
0：不复位 USART1
1：复位 USART1

位 3:2 保留，必须保持复位值。

位 1 **TIM8RST**：TIM8 复位 (TIM8 reset)
由软件置 1 和清零。
0：不复位 TIM8
1：复位 TIM8

位 0 **TIM1RST**：TIM1 复位 (TIM1 reset)
由软件置 1 和清零。
0：不复位 TIM1
1：复位 TIM1

6.3.12 RCC AHB1 外设时钟使能寄存器 (RCC_AHB1ENR)

RCC AHB1 peripheral clock enable register

偏移地址：0x30

复位值：0x0010 0000

访问：无等待周期，按字、半字和字节访问。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reser-ved	OTGHS ULPIEN	OTGHS EN	ETHMA CPTPE N	ETHMA CRXEN	ETHMA CTXEN	ETHMA CEN	Reserved		DMA2EN	DMA1EN	CCMDATA RAMEN	Res.	BKPSR AMEN	Reserved	
	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
Reserved			CRCEN	Reserved			GPIOIE N	GPIOH EN	GPIOGE N	GPIOFE N	GPIOEEN	GPIOD EN	GPIOC EN	GPIOB EN	GPIOA EN
			rw				rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31 保留，必须保持复位值。

位 30 **OTGHSULPIEN**：USB OTG HSULPI 时钟使能 (USB OTG HSULPI clock enable)
由软件置 1 和清零。
0：禁止 USB OTG HS ULPI 时钟
1：使能 USB OTG HS ULPI 时钟

位 29 **OTGHSEN**：USB OTG HS 时钟使能 (USB OTG HS clock enable)
由软件置 1 和清零。
0：禁止 USB OTG HS 时钟
1：使能 USB OTG HS 时钟

位 28 **ETHMACPTPEN**：以太网 PTP 时钟使能
由软件置 1 和清零。
0：禁止以太网 PTP 时钟
1：使能以太网 PTP 时钟

位 27 **ETHMACRXEN**：以太网接收时钟使能 (Ethernet Reception clock enable)
由软件置 1 和清零。
0：禁止以太网接收时钟
1：使能以太网接收时钟

位 26 **ETHMACTXEN**：以太网发送时钟使能 (Ethernet Transmission clock enable)
由软件置 1 和清零。
0：禁止以太网发送时钟
1：使能以太网发送时钟

位 25 **ETHMACEN**：以太网 MAC 时钟使能 (Ethernet MAC clock enable)
由软件置 1 和清零。
0：禁止以太网 MAC 时钟
1：使能以太网 MAC 时钟

位 24:23 保留，必须保持复位值。

位 22 **DMA2EN**：DMA2 时钟使能 (DMA2 clock enable)
由软件置 1 和清零。
0：禁止 DMA2 时钟
1：使能 DMA2 时钟

位 21 **DMA1EN**：DMA1 时钟使能 (DMA1 clock enable)
由软件置 1 和清零。
0：禁止 DMA1 时钟
1：使能 DMA1 时钟

位 20 **CCMDATARAMEN**：CCM 数据 RAM 时钟使能 (CCM data RAM clock enable)
由软件置 1 和清零。
0：禁止 CCM 数据 RAM 时钟
1：使能 CCM 数据 RAM 时钟

位 19 保留，必须保持复位值。

位 18 **BKPSRAMEN**：备份 SRAM 接口时钟使能 (Backup SRAM interface clock enable)
由软件置 1 和清零。
0：禁止备份 SRAM 接口时钟
1：使能备份 SRAM 接口时钟

位 17:13 保留，必须保持复位值。

位 12 **CRCEN**：CRC 时钟使能 (CRC clock enable)
由软件置 1 和清零。
0：禁止 CRC 时钟
1：使能 CRC 时钟

位 11:9 保留，必须保持复位值。

位 8 **GPIOIEN**：IO 端口 I 时钟使能 (IO port I clock enable)
由软件置 1 和清零。
0：禁止 IO 端口 I 时钟
1：使能 IO 端口 I 时钟

位 7 **GPIOHEN**：IO 端口 H 时钟使能 (IO port H clock enable)
由软件置 1 和清零。
0：禁止 IO 端口 H 时钟
1：使能 IO 端口 H 时钟

位 6 **GPIOGEN**：IO 端口 G 时钟使能 (IO port G clock enable)
由软件置 1 和清零。
0：禁止 IO 端口 G 时钟
1：使能 IO 端口 G 时钟

位 5 **GPIOFEN**：IO 端口 F 时钟使能 (IO port F clock enable)
由软件置 1 和清零。
0：禁止 IO 端口 F 时钟
1：使能 IO 端口 F 时钟

位 4 **GPIOEEN**：IO 端口 E 时钟使能 (IO port E clock enable)
由软件置 1 和清零。
0：禁止 IO 端口 E 时钟
1：使能 IO 端口 E 时钟

位 3 **GPIODEN**：IO 端口 D 时钟使能 (IO port D clock enable)
由软件置 1 和清零。
0：禁止 IO 端口 D 时钟
1：使能 IO 端口 D 时钟

位 2 **GPIOCEN**：IO 端口 C 时钟使能 (IO port C clock enable)
由软件置 1 和清零。
0：禁止 IO 端口 C 时钟
1：使能 IO 端口 C 时钟

位 1 **GPIOBEN**：IO 端口 B 时钟使能 (IO port B clock enable)
由软件置 1 和清零。
0：禁止 IO 端口 B 时钟
1：使能 IO 端口 B 时钟

位 0 **GPIOAEN**：IO 端口 A 时钟使能 (IO port A clock enable)
由软件置 1 和清零。
0：禁止 IO 端口 A 时钟
1：使能 IO 端口 A 时钟

6.3.13 RCC AHB2 外设时钟使能寄存器 (RCC_AHB2ENR)

RCC AHB2 peripheral clock enable register

偏移地址：0x34

复位值：0x0000 0000

访问：无等待周期，按字、半字和字节访问。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								OTGFSEN	RNGEN	HASHEN	CRYPEN	Reserved			DCMIEN
								rw	rw	rw	rw				rw

位 31:8 保留，必须保持复位值。

位 7 **OTGFSEN**：USB OTG FS 时钟使能 (USB OTG FS clock enable)
由软件置 1 和清零。
0：USB OTG FS 时钟禁用
1：使能 USB OTG FS 时钟

位 6 **RNGEN**：随机数发生器时钟使能 (Random number generator clock enable)
由软件置 1 和清零。
0：禁止随机数发生器时钟
1：使能随机数发生器时钟

位 5 **HASHEN**：散列模块时钟使能 (Hash modules clock enable)
由软件置 1 和清零。
0：禁止散列模块时钟
1：使能散列模块时钟

位 4 **CRYPEN**：加密模块时钟使能 (Cryptographic modules clock enable)
由软件置 1 和清零。
0：禁止加密模块时钟
1：使能加密模块时钟

位 3:1 保留，必须保持复位值。

位 0 **DCMIEN**：摄像头接口使能 (Camera interface enable)
由软件置 1 和清零。
0：禁止摄像头接口时钟
1：使能摄像头接口时钟

6.3.14 RCC AHB3 外设时钟使能寄存器 (RCC_AHB3ENR)

RCC AHB3 peripheral clock enable register

偏移地址：0x38

复位值：0x0000 0000

访问：无等待周期，按字、半字和字节访问。

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

位 31:1 保留，必须保持复位值。

位 0 **FSMCEN**：灵活的静态存储控制器模块时钟使能 (Flexible static memory controller clock enable)
由软件置 1 和清零。
0：禁止 FSMC 模块时钟
1：使能 FSMC 模块时钟

6.3.15 用于 STM32F405xx/07xx 和 STM32F415xx/17xx 的 RCC APB1 外设时钟使能寄存器 (RCC_APB1ENR)

RCC APB1 peripheral clock enable register

偏移地址：0x40

复位值：0x0000 0000

访问：无等待周期，按字、半字和字节访问。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved		DAC EN	PWR EN	Reser-ved	CAN2 EN	CAN1 EN	Reser-ved	I2C3 EN	I2C2 EN	I2C1 EN	UART5 EN	UART4 EN	USART3 EN	USART2 EN	Reser-ved
		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
SPI3 EN	SPI2 EN	Reserved		WWDG EN	Reserved		TIM14 EN	TIM13 EN	TIM12 EN	TIM7 EN	TIM6 EN	TIM5 EN	TIM4 EN	TIM3 EN	TIM2 EN
rw	rw			rw			rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:30 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 和清零。
0：禁止 DAC 接口时钟
1：使能 DAC 接口时钟

位 28 **PWREN**：电源接口时钟使能 (Power interface clock enable)
由软件置 1 和清零。
0：禁止电源接口时钟
1：使能电源接口时钟

位 27 保留，必须保持复位值。

位 26 **CAN2EN**：CAN 2 时钟使能 (CAN 2 clock enable)
由软件置 1 和清零。
0：禁止 CAN 2 时钟
1：使能 CAN 2 时钟

位 25 **CAN1EN**：CAN 1 时钟使能 (CAN 1 clock enable)
由软件置 1 和清零。
0：禁止 CAN 1 时钟
1：使能 CAN 1 时钟

位 24 保留，必须保持复位值。

位 23 **I2C3EN**：I2C3 时钟使能 (I2C3 clock enable)
由软件置 1 和清零。
0：禁止 I2C3 时钟
1：使能 I2C3 时钟

位 22 **I2C2EN**：I2C2 时钟使能 (I2C2 clock enable)
由软件置 1 和清零。
0：禁止 I2C2 时钟
1：使能 I2C2 时钟

位 21 **I2C1EN**：I2C1 时钟使能 (I2C1 clock enable)
由软件置 1 和清零。
0：禁止 I2C1 时钟
1：使能 I2C1 时钟

位 20 **UART5EN**：UART5 时钟使能 (UART5 clock enable)
由软件置 1 和清零。
0：禁止 UART5 时钟
1：使能 UART5 时钟

位 19 **UART4EN**：UART4 时钟使能 (UART4 clock enable)
由软件置 1 和清零。
0：禁止 UART4 时钟
1：使能 UART4 时钟

位 18 **USART3EN**：USART3 时钟使能 (USART3 clock enable)
由软件置 1 和清零。
0：禁止 USART3 时钟
1：使能 USART3 时钟

位 17 **USART2EN**：USART2 时钟使能 (USART2 clock enable)
由软件置 1 和清零。
0：禁止 USART2 时钟
1：使能 USART2 时钟

位 16 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 和清零。
0：禁止 SPI3 时钟
1：使能 SPI3 时钟

位 14 **SPI2EN**：SPI2 时钟使能 (SPI2 clock enable)
由软件置 1 和清零。
0：禁止 SPI2 时钟
1：使能 SPI2 时钟

位 13:12 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 和清零。
0：禁止窗口看门狗时钟
1：使能窗口看门狗时钟

位 10:9 保留，必须保持复位值。

位 8 **TIM14EN**：TIM14 时钟使能 (TIM14 clock enable)
由软件置 1 和清零。
0：禁止 TIM14 时钟
1：使能 TIM14 时钟

位 7 **TIM13EN**：TIM13 时钟使能 (TIM13 clock enable)
由软件置 1 和清零。
0：禁止 TIM13 时钟
1：使能 TIM13 时钟

位 6 **TIM12EN**：TIM12 时钟使能 (TIM12 clock enable)
由软件置 1 和清零。
0：禁止 TIM12 时钟
1：使能 TIM12 时钟

位 5 **TIM7EN**：TIM7 时钟使能 (TIM7 clock enable)
由软件置 1 和清零。
0：禁止 TIM7 时钟
1：使能 TIM7 时钟

位 4 **TIM6EN**：TIM6 时钟使能 (TIM6 clock enable)
由软件置 1 和清零。
0：禁止 TIM6 时钟
1：使能 TIM6 时钟

位 3 **TIM5EN**：TIM5 时钟使能 (TIM5 clock enable)
由软件置 1 和清零。
0：禁止 TIM5 时钟
1：使能 TIM5 时钟

位 2 **TIM4EN**：TIM4 时钟使能 (TIM4 clock enable)
由软件置 1 和清零。
0：禁止 TIM4 时钟
1：使能 TIM4 时钟

位 1 **TIM3EN**：TIM3 时钟使能 (TIM3 clock enable)
由软件置 1 和清零。
0：禁止 TIM3 时钟
1：使能 TIM3 时钟

位 0 **TIM2EN**：TIM2 时钟使能 (TIM2 clock enable)
由软件置 1 和清零。
0：禁止 TIM2 时钟
1：使能 TIM2 时钟

6.3.16 用于 STM32F42xxx 和 STM32F43xxx 的 RCC APB1 外设时钟使能寄存器 (RCC_APB1ENR)

RCC APB1 peripheral clock enable register

偏移地址：0x40

复位值：0x0000 0000

访问：无等待周期，按字、半字和字节访问。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UART8 EN	UART7 EN	DAC EN	PWR EN	Reser-ved	CAN2 EN	CAN1 EN	Reser-ved	I2C3 EN	I2C2 EN	I2C1 EN	UART5 EN	UART4 EN	USART3 EN	USART2 EN	Reser-ved
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
SPI3 EN	SPI2 EN	Reserved		WWDG EN	Reserved		TIM14 EN	TIM13 EN	TIM12 EN	TIM7 EN	TIM6 EN	TIM5 EN	TIM4 EN	TIM3 EN	TIM2 EN
rw	rw			rw			rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31 **UART8EN**：UART8 时钟使能 (UART8 clock enable)
由软件置 1 和清零。
0：禁止 UART8 时钟
1：使能 UART8 时钟

位 30 **UART7EN**：UART7 时钟使能 (UART7 clock enable)
由软件置 1 和清零。
0：禁止 UART7 时钟
1：使能 UART7 时钟

位 29 **DACEN**：DAC 接口时钟使能 (DAC interface clock enable)
由软件置 1 和清零。
0：禁止 DAC 接口时钟
1：使能 DAC 接口时钟

位 28 **PWREN**：电源接口时钟使能 (Power interface clock enable)
由软件置 1 和清零。
0：禁止电源接口时钟
1：使能电源接口时钟

位 27 保留，必须保持复位值。

位 26 **CAN2EN**：CAN 2 时钟使能 (CAN 2 clock enable)
由软件置 1 和清零。
0：禁止 CAN 2 时钟
1：使能 CAN 2 时钟

位 25 **CAN1EN**：CAN 1 时钟使能 (CAN 1 clock enable)
由软件置 1 和清零。
0：禁止 CAN 1 时钟
1：使能 CAN 1 时钟

位 24 保留，必须保持复位值。

位 23 **I2C3EN**：I2C3 时钟使能 (I2C3 clock enable)
由软件置 1 和清零。
0：禁止 I2C3 时钟
1：使能 I2C3 时钟

位 22 **I2C2EN**：I2C2 时钟使能 (I2C2 clock enable)
由软件置 1 和清零。
0：禁止 I2C2 时钟
1：使能 I2C2 时钟

位 21 **I2C1EN**：I2C1 时钟使能 (I2C1 clock enable)
由软件置 1 和清零。
0：禁止 I2C1 时钟
1：使能 I2C1 时钟

位 20 **UART5EN**：UART5 时钟使能 (UART5 clock enable)
由软件置 1 和清零。
0：禁止 UART5 时钟
1：使能 UART5 时钟

位 19 **UART4EN**：UART4 时钟使能 (UART4 clock enable)
由软件置 1 和清零。
0：禁止 UART4 时钟
1：使能 UART4 时钟

位 18 **USART3EN**：USART3 时钟使能 (USART3 clock enable)
由软件置 1 和清零。
0：禁止 USART3 时钟
1：使能 USART3 时钟

位 17 **USART2EN**：USART2 时钟使能 (USART2 clock enable)
由软件置 1 和清零。
0：禁止 USART2 时钟
1：使能 USART2 时钟

位 16 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 和清零。
0：禁止 SPI3 时钟
1：使能 SPI3 时钟

位 14 **SPI2EN**：SPI2 时钟使能 (SPI2 clock enable)
由软件置 1 和清零。
0：禁止 SPI2 时钟
1：使能 SPI2 时钟

位 13:12 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 和清零。
0：禁止窗口看门狗时钟
1：使能窗口看门狗时钟

位 10:9 保留，必须保持复位值。

位 8 **TIM14EN**：TIM14 时钟使能 (TIM14 clock enable)
由软件置 1 和清零。
0：禁止 TIM14 时钟
1：使能 TIM14 时钟

位 7 **TIM13EN**：TIM13 时钟使能 (TIM13 clock enable)
由软件置 1 和清零。
0：禁止 TIM13 时钟
1：使能 TIM13 时钟

位 6 **TIM12EN**：TIM12 时钟使能 (TIM12 clock enable)
由软件置 1 和清零。
0：禁止 TIM12 时钟
1：使能 TIM12 时钟

位 5 **TIM7EN**：TIM7 时钟使能 (TIM7 clock enable)
由软件置 1 和清零。
0：禁止 TIM7 时钟
1：使能 TIM7 时钟

位 4 **TIM6EN**：TIM6 时钟使能 (TIM6 clock enable)
由软件置 1 和清零。
0：禁止 TIM6 时钟
1：使能 TIM6 时钟

位 3 **TIM5EN**：TIM5 时钟使能 (TIM5 clock enable)
由软件置 1 和清零。
0：禁止 TIM5 时钟
1：使能 TIM5 时钟

位 2 **TIM4EN**：TIM4 时钟使能 (TIM4 clock enable)
由软件置 1 和清零。
0：禁止 TIM4 时钟
1：使能 TIM4 时钟

位 1 **TIM3EN**：TIM3 时钟使能 (TIM3 clock enable)
由软件置 1 和清零。
0：禁止 TIM3 时钟
1：使能 TIM3 时钟

位 0 **TIM2EN**：TIM2 时钟使能 (TIM2 clock enable)
由软件置 1 和清零。
0：禁止 TIM2 时钟
1：使能 TIM2 时钟

6.3.17 用于 STM32F405xx/07xx 和 STM32F415xx/17xx 的 RCC APB2 外设时钟使能寄存器 (RCC_APB2ENR)

RCC APB2 peripheral clock enable register

偏移地址：0x44

复位值：0x0000 0000

访问：无等待周期，按字、半字和字节访问。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved													TIM11 EN	TIM10 EN	TIM9 EN
													rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reser-ved	SYSCFG EN	Reser-ved	SPI1 EN	SDIO EN	ADC3 EN	ADC2 EN	ADC1 EN	Reserved		USART6 EN	USART1 EN	Reserved		TIM8 EN	TIM1 EN
	rw		rw	rw	rw	rw	rw			rw	rw			rw	rw

位 31:19 保留，必须保持复位值。

位 18 **TIM11EN**：TIM11 时钟使能 (TIM11 clock enable)
由软件置 1 和清零。
0：禁止 TIM11 时钟
1：使能 TIM11 时钟

位 17 **TIM10EN**：TIM10 时钟使能 (TIM10 clock enable)
由软件置 1 和清零。
0：禁止 TIM10 时钟
1：使能 TIM10 时钟

位 16 **TIM9EN**：TIM9 时钟使能 (TIM9 clock enable)
由软件置 1 和清零。
0：禁止 TIM9 时钟
1：使能 TIM9 时钟

位 15 保留，必须保持复位值。

位 14 **SYSCFGEN**：系统配置控制器时钟使能 (System configuration controller clock enable)
由软件置 1 和清零。
0：禁止系统配置控制器时钟
1：使能系统配置控制器时钟

位 13 保留，必须保持复位值。

位 12 **SPI1EN**：SPI1 时钟使能 (SPI1 clock enable)
由软件置 1 和清零。
0：禁止 SPI1 时钟
1：使能 SPI1 时钟

位 11 **SDIOEN**：SDIO 时钟使能 (SDIO clock enable)
由软件置 1 和清零。
0：禁止 SDIO 模块时钟
1：使能 SDIO 模块时钟

位 10 **ADC3EN**：ADC3 时钟使能 (ADC3 clock enable)
由软件置 1 和清零。
0：禁止 ADC3 时钟
1：使能 ADC3 时钟

位 9 **ADC2EN**：ADC2 时钟使能 (ADC2 clock enable)
由软件置 1 和清零。
0：禁止 ADC2 时钟
1：使能 ADC2 时钟

位 8 **ADC1EN**：ADC1 时钟使能 (ADC1 clock enable)
由软件置 1 和清零。
0：禁止 ADC1 时钟
1：使能 ADC1 时钟

位 7:6 保留，必须保持复位值。

位 5 **USART6EN**：USART6 时钟使能 (USART6 clock enable)
由软件置 1 和清零。
0：禁止 USART6 时钟
1：使能 USART6 时钟

位 4 **USART1EN**：USART1 时钟使能 (USART1 clock enable)
由软件置 1 和清零。
0：禁止 USART1 时钟
1：使能 USART1 时钟

位 3:2 保留，必须保持复位值。

位 1 **TIM8EN**：TIM8 时钟使能 (TIM8 clock enable)
由软件置 1 和清零。
0：禁止 TIM8 时钟
1：使能 TIM8 时钟

位 0 **TIM1EN**：TIM1 时钟使能 (TIM1 clock enable)
由软件置 1 和清零。
0：禁止 TIM1 时钟
1：使能 TIM1 时钟

6.3.18 用于 STM32F42xxx 和 STM32F43xxx 的 RCC APB2 外设时钟使能寄存器 (RCC_APB2ENR)

RCC APB2 peripheral clock enable register

偏移地址：0x44

复位值：0x0000 0000

访问：无等待周期，按字、半字和字节访问。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved										SPI6EN	SPI5EN	Res.	TIM11 EN	TIM10 EN	TIM9 EN
										rw	rw		rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reser-ved	SYSCFG EN	SPI4EN	SPI1 EN	SDIO EN	ADC3 EN	ADC2 EN	ADC1 EN	Reserved		USART6 EN	USART1 EN	Reserved		TIM8 EN	TIM1 EN
	rw	rw	rw	rw	rw	rw	rw			rw	rw			rw	rw

位 31:22 保留，必须保持复位值。

位 21 **SPI6EN**：SPI6 时钟使能 (SPI6 clock enable)
由软件置 1 和清零。
0：禁止 SPI6 时钟
1：使能 SPI6 时钟

位 20 **SPI5EN**：SPI5 时钟使能 (SPI5 clock enable)
由软件置 1 和清零。
0：禁止 SPI5 时钟
1：使能 SPI5 时钟

位 19 保留，必须保持复位值。

位 18 **TIM11EN**：TIM11 时钟使能 (TIM11 clock enable)
由软件置 1 和清零。
0：禁止 TIM11 时钟
1：使能 TIM11 时钟

位 17 **TIM10EN**：TIM10 时钟使能 (TIM10 clock enable)
由软件置 1 和清零。
0：禁止 TIM10 时钟
1：使能 TIM10 时钟

位 16 **TIM9EN**：TIM9 时钟使能 (TIM9 clock enable)
由软件置 1 和清零。
0：禁止 TIM9 时钟
1：使能 TIM9 时钟

位 15 保留，必须保持复位值。

位 14 **SYSCFGEN**：系统配置控制器时钟使能 (System configuration controller clock enable)
由软件置 1 和清零。
0：禁止系统配置控制器时钟
1：使能系统配置控制器时钟

位 13 **SPI4EN**：SPI4 时钟使能 (SPI4 clock enable)
由软件置 1 和清零。
0：禁止 SPI4 时钟
1：使能 SPI4 时钟

位 12 **SPI1EN**：SPI1 时钟使能 (SPI1 clock enable)
由软件置 1 和清零。
0：禁止 SPI1 时钟
1：使能 SPI1 时钟

位 11 **SDIOEN**：SDIO 时钟使能 (SDIO clock enable)
由软件置 1 和清零。
0：禁止 SDIO 模块时钟
1：使能 SDIO 模块时钟

位 10 **ADC3EN**：ADC3 时钟使能 (ADC3 clock enable)
由软件置 1 和清零。
0：禁止 ADC3 时钟
1：使能 ADC3 时钟

位 9 **ADC2EN**：ADC2 时钟使能 (ADC2 clock enable)
由软件置 1 和清零。
0：禁止 ADC2 时钟
1：使能 ADC2 时钟

位 8 **ADC1EN**：ADC1 时钟使能 (ADC1 clock enable)
由软件置 1 和清零。
0：禁止 ADC1 时钟
1：使能 ADC1 时钟

位 7:6 保留，必须保持复位值。

位 5 **USART6EN**：USART6 时钟使能 (USART6 clock enable)
由软件置 1 和清零。
0：禁止 USART6 时钟
1：使能 USART6 时钟

位 4 **USART1EN**：USART1 时钟使能 (USART1 clock enable)
由软件置 1 和清零。
0：禁止 USART1 时钟
1：使能 USART1 时钟

位 3:2 保留，必须保持复位值。

位 1 **TIM8EN**：TIM8 时钟使能 (TIM8 clock enable)
由软件置 1 和清零。
0：禁止 TIM8 时钟
1：使能 TIM8 时钟

位 0 **TIM1EN**：TIM1 时钟使能 (TIM1 clock enable)
由软件置 1 和清零。
0：禁止 TIM1 时钟
1：使能 TIM1 时钟

6.3.19 用于 STM32F405xx/07xx 和 STM32F415xx/17xx 的低功耗模式寄存器中的 RCC AHB1 外设时钟使能 (RCC_AHB1LPENR)

RCC AHB1 peripheral clock enable in low power mode register

偏移地址：0x50

复位值：0x7E67 91FF

访问：无等待周期，按字、半字和字节访问。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reser-ved	OTGHS ULPILPEN	OTGHS LPEN	ETHPTP LPEN	ETHRX LPEN	ETHTX LPEN	ETHMAC LPEN	Reserved		DMA2 LPEN	DMA1 LPEN	Reserved		BKPSRAM LPEN	SRAM2 LPEN	SRAM1 LPEN
	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
FLITF LPEN	Reserved		CRC LPEN	Reserved			GPIOI LPEN	GPIOH LPEN	GPIOGG LPEN	GPIOF LPEN	GPIOE LPEN	GPIOD LPEN	GPIOC LPEN	GPIOB LPEN	GPIOA LPEN
rw			rw				rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31 保留，必须保持复位值。

位 30 **OTGHSULPILPEN**：睡眠模式期间的 USB OTG HS ULPI 时钟使能 (USB OTG HS ULPI clock enable during Sleep mode)

由软件置 1 和清零。
0：睡眠模式期间禁止 USB OTG HS ULPI 时钟
1：睡眠模式期间使能 USB OTG HS ULPI 时钟

位 29 **OTGHSLPEN**：睡眠模式期间的 USB OTG HS 时钟使能 (USB OTG HS clock enable during Sleep mode)

由软件置 1 和清零。
0：睡眠模式期间禁止 USB OTG HS 时钟
1：睡眠模式期间使能 USB OTG HS 时钟

位 28 **ETHMACPTPLPEN**：睡眠模式期间的以太网 PTP 时钟使能 (Ethernet PTP clock enable during Sleep mode)

由软件置 1 和清零。
0：睡眠模式期间禁止以太网 PTP 时钟
1：睡眠模式期间使能 Ethernet PTP 时钟

位 27 **ETHMACRXLPEN**：睡眠模式期间的以太网接收时钟使能 (Ethernet reception clock enable during Sleep mode)

由软件置 1 和清零。
0：睡眠模式期间禁止以太网接收时钟
1：睡眠模式期间使能以太网接收时钟

位 26 **ETHMACTXLPEN**：睡眠模式期间的以太网发送时钟使能 (Ethernet transmission clock enable during Sleep mode)

由软件置 1 和清零。
0：睡眠模式期间禁止以太网发送时钟
1：睡眠模式期间使能以太网发送时钟

位 25 **ETHMACLPEN**：睡眠模式期间的以太网 MAC 时钟使能 (Ethernet MAC clock enable during Sleep mode)

由软件置 1 和清零。
0：睡眠模式期间禁止以太网 MAC 时钟
1：睡眠模式期间使能以太网 MAC 时钟

位 24:23 保留，必须保持复位值。

位 22 **DMA2LPEN**：睡眠模式期间的 DMA2 时钟使能 (DMA2 clock enable during Sleep mode)

由软件置 1 和清零。
0：睡眠模式期间禁止 DMA2 时钟
1：睡眠模式期间使能 DMA2 时钟

位 21 **DMA1LPEN**：睡眠模式期间的 DMA1 时钟使能 (DMA1 clock enable during Sleep mode)

由软件置 1 和清零。
0：睡眠模式期间禁止 DMA1 时钟
1：睡眠模式期间使能 DMA1 时钟

位 20:19 保留，必须保持复位值。

位 18 **BKPSRAMLPEN**：睡眠模式期间的备份 SRAM 接口时钟使能 (Backup SRAM interface clock enable during Sleep mode)

由软件置 1 和清零。
0：睡眠模式期间禁止备份 SRAM 接口时钟
1：睡眠模式期间使能备份 SRAM 接口时钟

位 17 **SRAM2LPEN**：睡眠模式期间的 SRAM 2 接口时钟使能 (SRAM 2 interface clock enable during Sleep mode)

由软件置 1 和清零。
0：睡眠模式期间禁止 SRAM 2 接口时钟
1：睡眠模式期间使能 SRAM 2 接口时钟

位 16 **SRAM1LPEN**：睡眠模式期间的 SRAM 1 接口时钟使能 (SRAM 1interface clock enable during Sleep mode)

由软件置 1 和清零。
0：睡眠模式期间禁止 SRAM 1 接口时钟
1：睡眠模式期间使能 SRAM 1 接口时钟

位 15 **FLITFLPEN**：睡眠模式期间的 Flash 接口时钟使能 (Flash interface clock enable during Sleep mode)

由软件置 1 和清零。
0：睡眠模式期间禁止 Flash 接口时钟
1：睡眠模式期间使能 Flash 接口时钟

位 14:13 保留，必须保持复位值。

位 12 **CRCLPEN**：睡眠模式期间的 CRC 时钟使能 (CRC clock enable during Sleep mode)

由软件置 1 和清零。
0：睡眠模式期间禁止 CRC 时钟
1：睡眠模式期间使能 CRC 时钟

位 11:9 保留，必须保持复位值。

位 8 **GPIOILPEN**：睡眠模式期间的 IO 端口 I 时钟使能 (IO port I clock enable during Sleep mode)

由软件置 1 和清零。
0：睡眠模式期间禁止 IO 端口 I 时钟
1：睡眠模式期间使能 IO 端口 I 时钟

位 7 **GPIOHLPEN**：睡眠模式期间的 IO 端口 H 时钟使能 (IO port H clock enable during Sleep mode)

由软件置 1 和清零。

0：睡眠模式期间禁止 IO 端口 H 时钟

1：睡眠模式期间使能 IO 端口 H 时钟

位 6 **GPIOGLPEN**：睡眠模式期间的 IO 端口 G 时钟使能 (IO port G clock enable during Sleep mode)

由软件置 1 和清零。

0：睡眠模式期间禁止 IO 端口 G 时钟

1：睡眠模式期间使能 IO 端口 G 时钟

位 5 **GPIOFLPEN**：睡眠模式期间的 IO 端口 F 时钟使能 (IO port F clock enable during Sleep mode)

由软件置 1 和清零。

0：睡眠模式期间禁止 IO 端口 F 时钟

1：睡眠模式期间使能 IO 端口 F 时钟

位 4 **GPIOELPEN**：睡眠模式期间的 IO 端口 E 时钟使能 (IO port E clock enable during Sleep mode)

由软件置 1 和清零。

0：睡眠模式期间禁止 IO 端口 E 时钟

1：睡眠模式期间使能 IO 端口 E 时钟

位 3 **GPIODLPEN**：睡眠模式期间的 IO 端口 D 时钟使能 (IO port D clock enable during Sleep mode)

由软件置 1 和清零。

0：睡眠模式期间禁止 IO 端口 D 时钟

1：睡眠模式期间使能 IO 端口 D 时钟

位 2 **GPIOCLPEN:**睡眠模式期间的 IO 端口 C 时钟使能 (IO port C clock enable during Sleep mode)

由软件置 1 和清零。

0：睡眠模式期间禁止 IO 端口 C 时钟

1：睡眠模式期间使能 IO 端口 C 时钟

位 1 **GPIOBLPEN**：睡眠模式期间的 IO 端口 B 时钟使能 (IO port B clock enable during Sleep mode)

由软件置 1 和清零。

0：睡眠模式期间禁止 IO 端口 B 时钟

1：睡眠模式期间使能 IO 端口 B 时钟

位 0 **GPIOALPEN**：睡眠模式期间的 IO 端口 A 时钟使能 (IO port A clock enable during sleep mode)

由软件置 1 和清零。

0：睡眠模式期间禁止 IO 端口 A 时钟

1：睡眠模式期间使能 IO 端口 A 时钟

6.3.20 用于 STM32F42xxx 和 STM32F43xxx的低功耗模式寄存器中的 RCC AHB1 外设时钟使能 (RCC_AHB1LPENR)

RCC AHB1 peripheral clock enable in low power mode register

偏移地址：0x50

复位值：0x7E67 91FF

访问：无等待周期，按字、半字和字节访问。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reser-ved	OTGHS ULPILPEN	OTGHS LPEN	ETHPTP LPEN	ETHRX LPEN	ETHTX LPEN	ETHMAC LPEN	Reserved		DMA2 LPEN	DMA1 LPEN	Res.	SRAM3 LPEN	BKPSRAM LPEN	SRAM2 LPEN	SRAM1 LPEN
	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
FLITF LPEN	Reserved		CRC LPEN	Reserved			GPIOI LPEN	GPIOH LPEN	GPIOGG LPEN	GPIOF LPEN	GPIOE LPEN	GPIOD LPEN	GPIOC LPEN	GPIOB LPEN	GPIOA LPEN
rw			rw				rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31 保留，必须保持复位值。

位 30 **OTGHSULPILPEN**：睡眠模式期间的 USB OTG HS ULPI 时钟使能 (USB OTG HS ULPI clock enable during Sleep mode)

由软件置 1 和清零。
0：睡眠模式期间禁止 USB OTG HS ULPI 时钟
1：睡眠模式期间使能 USB OTG HS ULPI 时钟

位 29 **OTGHSLPEN**：睡眠模式期间的 USB OTG HS 时钟使能 (USB OTG HS clock enable during Sleep mode)

由软件置 1 和清零。
0：睡眠模式期间禁止 USB OTG HS 时钟
1：睡眠模式期间使能 USB OTG HS 时钟

位 28 **ETHMACPTPLPEN**：睡眠模式期间的以太网 PTP 时钟使能 (Ethernet PTP clock enable during Sleep mode)

由软件置 1 和清零。
0：睡眠模式期间禁止以太网 PTP 时钟
1：睡眠模式期间使能 Ethernet PTP 时钟

位 27 **ETHMACRXLPEN**：睡眠模式期间的以太网接收时钟使能 (Ethernet reception clock enable during Sleep mode)

由软件置 1 和清零。
0：睡眠模式期间禁止以太网接收时钟
1：睡眠模式期间使能以太网接收时钟

位 26 **ETHMACTXLPEN**：睡眠模式期间的以太网发送时钟使能 (Ethernet transmission clock enable during Sleep mode)

由软件置 1 和清零。
0：睡眠模式期间禁止以太网发送时钟
1：睡眠模式期间使能以太网发送时钟

位 25 **ETHMACLPEN**：睡眠模式期间的以太网 MAC 时钟使能 (Ethernet MAC clock enable during Sleep mode)

由软件置 1 和清零。
0：睡眠模式期间禁止以太网 MAC 时钟
1：睡眠模式期间使能以太网 MAC 时钟

位 24:23 保留，必须保持复位值。

位 22 **DMA2LPEN**：睡眠模式期间的 DMA2 时钟使能 (DMA2 clock enable during Sleep mode)

由软件置 1 和清零。
0：睡眠模式期间禁止 DMA2 时钟
1：睡眠模式期间使能 DMA2 时钟

位 21 **DMA1LPEN**：睡眠模式期间的 DMA1 时钟使能 (DMA1 clock enable during Sleep mode)

由软件置 1 和清零。
0：睡眠模式期间禁止 DMA1 时钟
1：睡眠模式期间使能 DMA1 时钟

位 20 保留，必须保持复位值。

位 19 **SRAM3LPEN**：睡眠模式期间的 SRAM3 接口时钟使能 (SRAM3 interface clock enable during Sleep mode)

由软件置 1 和清零。
0：睡眠模式期间禁止 SRAM3 接口时钟
1：睡眠模式期间使能 SRAM3 接口时钟

位 18 **BKPSRAMLPEN**：睡眠模式期间的备份 SRAM 接口时钟使能 (Backup SRAM interface clock enable during Sleep mode)

由软件置 1 和清零。
0：睡眠模式期间禁止备份 SRAM 接口时钟
1：睡眠模式期间使能备份 SRAM 接口时钟

位 17 **SRAM2LPEN**：睡眠模式期间的 SRAM2 接口时钟使能 (SRAM2 interface clock enable during Sleep mode)

由软件置 1 和清零。
0：睡眠模式期间禁止 SRAM2 接口时钟
1：睡眠模式期间使能 SRAM2 接口时钟

位 16 **SRAM1LPEN**：睡眠模式期间的 SRAM1 接口时钟使能 (SRAM1 interface clock enable during Sleep mode)

由软件置 1 和清零。
0：睡眠模式期间禁止 SRAM1 接口时钟
1：睡眠模式期间使能 SRAM1 接口时钟

位 15 **FLITFLPEN**：睡眠模式期间的 Flash 接口时钟使能 (Flash interface clock enable during Sleep mode)

由软件置 1 和清零。
0：睡眠模式期间禁止 Flash 接口时钟
1：睡眠模式期间使能 Flash 接口时钟

位 14:13 保留，必须保持复位值。

位 12 **CRCLPEN**：睡眠模式期间的 CRC 时钟使能 (CRC clock enable during Sleep mode)

由软件置 1 和清零。
0：睡眠模式期间禁止 CRC 时钟
1：睡眠模式期间使能 CRC 时钟

位 11:9 保留，必须保持复位值。

位 8 **GPIOILPEN**：睡眠模式期间的 IO 端口 I 时钟使能 (IO port I clock enable during Sleep mode)
由软件置 1 和清零。
0：睡眠模式期间禁止 IO 端口 I 时钟
1：睡眠模式期间使能 IO 端口 I 时钟

位 7 **GPIOHLPEN**：睡眠模式期间的 IO 端口 H 时钟使能 (IO port H clock enable during Sleep mode)
由软件置 1 和清零。
0：睡眠模式期间禁止 IO 端口 H 时钟
1：睡眠模式期间使能 IO 端口 H 时钟

位 6 **GPIOGLPEN**：睡眠模式期间的 IO 端口 G 时钟使能 (IO port G clock enable during Sleep mode)
由软件置 1 和清零。
0：睡眠模式期间禁止 IO 端口 G 时钟
1：睡眠模式期间使能 IO 端口 G 时钟

位 5 **GPIOFLPEN**：睡眠模式期间的 IO 端口 F 时钟使能 (IO port F clock enable during Sleep mode)
由软件置 1 和清零。
0：睡眠模式期间禁止 IO 端口 F 时钟
1：睡眠模式期间使能 IO 端口 F 时钟

位 4 **GPIOELPEN**：睡眠模式期间的 IO 端口 E 时钟使能 (IO port E clock enable during Sleep mode)
由软件置 1 和清零。
0：睡眠模式期间禁止 IO 端口 E 时钟
1：睡眠模式期间使能 IO 端口 E 时钟

位 3 **GPIODLPEN**：睡眠模式期间的 IO 端口 D 时钟使能 (IO port D clock enable during Sleep mode)
由软件置 1 和清零。
0：睡眠模式期间禁止 IO 端口 D 时钟
1：睡眠模式期间使能 IO 端口 D 时钟

位 2 **GPIOCLPEN:**睡眠模式期间的 IO 端口 C 时钟使能 (IO port C clock enable during Sleep mode)
由软件置 1 和清零。
0：睡眠模式期间禁止 IO 端口 C 时钟
1：睡眠模式期间使能 IO 端口 C 时钟

位 1 **GPIOBLPEN**：睡眠模式期间的 IO 端口 B 时钟使能 (IO port B clock enable during Sleep mode)
由软件置 1 和清零。
0：睡眠模式期间禁止 IO 端口 B 时钟
1：睡眠模式期间使能 IO 端口 B 时钟

位 0 **GPIOALPEN**：睡眠模式期间的 IO 端口 A 时钟使能 (IO port A clock enable during sleep mode)
由软件置 1 和清零。
0：睡眠模式期间禁止 IO 端口 A 时钟
1：睡眠模式期间使能 IO 端口 A 时钟

6.3.21 用于低功耗模式寄存器中的 RCC AHB2 外设时钟使能 (RCC_AHB2LPENR)

RCC AHB2 peripheral clock enable in low power mode register

偏移地址：0x54

复位值：0x0000 00F1

访问：无等待周期，按字、半字和字节访问。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								OTGFS LPEN	RNG LPEN	HASH LPEN	CRYP LPEN	Reserved			DCMI LPEN
								rw	rw	rw	rw				rw

位 31:8 保留，必须保持复位值。

位 7 **OTGFSLPEN**：睡眠模式期间的 USB OTG FS 时钟使能 (USB OTG FS clock enable during Sleep mode)

由软件置 1 和清零。
0：睡眠模式期间禁止 USB OTG FS 时钟
1：睡眠模式期间使能 USB OTG FS 时钟

位 6 **RNGLPEN**：睡眠模式期间的随机数发生器时钟使能 (Random number generator clock enable during Sleep mode)

由软件置 1 和清零。
0：睡眠模式期间禁止随机数发生器时钟
1：睡眠模式期间使能随机数发生器时钟

位 5 **HASHLPEN**：睡眠模式期间的 Hash 模块时钟使能 (Hash modules clock enable during Sleep mode)

由软件置 1 和清零。
0：睡眠模式期间禁止 Hash 模块时钟
1：睡眠模式期间使能 Hash 模块时钟

位 4 **CRYPLPEN**：睡眠模式期间的加密模块时钟使能 (Cryptography modules clock enable during Sleep mode)

由软件置 1 和清零。
0：睡眠模式期间禁止加密模块时钟
1：睡眠模式期间使能加密模块时钟

位 3:1 保留，必须保持复位值。

位 0 **DCMILPEN**：睡眠模式期间的摄像头接口时钟使能 (Camera interface enable during Sleep mode)

由软件置 1 和清零。
0：睡眠模式期间禁止摄像头接口时钟
1：睡眠模式期间使能摄像头接口时钟

6.3.22 低功耗模式寄存器中的 RCC AHB3 外设时钟使能 (RCC_AHB3LPENR)

RCC AHB3 peripheral clock enable in low power mode register

偏移地址：0x58

复位值：0x0000 0001

访问：无等待周期，按字、半字和字节访问。

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

位 31:1保留，必须保持复位值。

FSMCLPEN：睡眠模式期间的灵活的静态存储控制器模块使能 (Flexible static memory controller module clock enable during Sleep mode)

位 0 由软件置 1 和清零。

0：睡眠模式期间禁止灵活的静态存储控制器模块时钟

1：睡眠模式期间使能灵活的静态存储控制器模块时钟

6.3.23 用于 STM32F405xx/07xx 和 STM32F415xx/17xx 的低功耗模式寄存器中的 RCC APB1 外设时钟使能 (RCC_APB1LPENR)

RCC APB1 peripheral clock enable in low power mode register

偏移地址：0x60

复位值：0x36FE C9FF

访问：无等待周期，按字、半字和字节访问。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved		DAC LPEN	PWR LPEN	RESERVED	CAN2 LPEN	CAN1 LPEN	Reserved	I2C3 LPEN	I2C2 LPEN	I2C1 LPEN	UART5 LPEN	UART4 LPEN	USART3 LPEN	USART2 LPEN	Reserved
		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
SPI3 LPEN	SPI2 LPEN	Reserved		WWDG LPEN	Reserved		TIM14 LPEN	TIM13 LPEN	TIM12 LPEN	TIM7 LPEN	TIM6 LPEN	TIM5 LPEN	TIM4 LPEN	TIM3 LPEN	TIM2 LPEN
rw	rw			rw			rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:30 保留，必须保持复位值。

位 29 **DACLPEN**：睡眠模式期间的 DAC 接口时钟使能 (DAC interface clock enable during Sleep mode)

由软件置 1 和清零。

0：睡眠模式期间禁止 DAC 接口时钟

1：睡眠模式期间使能 DAC 接口时钟

位 28 **PWRLPEN**：睡眠模式期间的电源接口时钟使能 (Power interface clock enable during Sleep mode)

由软件置 1 和清零。

0：睡眠模式期间禁止电源接口时钟

1：睡眠模式期间使能电源接口时钟

位 27 保留，必须保持复位值。

位 26 **CAN2LPEN**：睡眠模式期间的 CAN 2 时钟使能 (CAN 2 clock enable during Sleep mode)

由软件置 1 和清零。

0：睡眠模式期间禁止 CAN 2 时钟

1：睡眠模式期间使能 CAN 2 时钟

位 25 **CAN1LPEN**：睡眠模式期间的 CAN 1 时钟使能 (CAN 1 clock enable during Sleep mode)

由软件置 1 和清零。

0：睡眠模式期间禁止 CAN 1 时钟

1：睡眠模式期间使能 CAN 1 时钟

位 24 保留，必须保持复位值。

位 23 **I2C3LPEN**：睡眠模式期间的 I2C3 时钟使能 (I2C3 clock enable during Sleep mode)

由软件置 1 和清零。

0：睡眠模式期间禁止 I2C3 时钟

1：睡眠模式期间使能 I2C3 时钟

位 22 **I2C2LPEN**：睡眠模式期间的 I2C2 时钟使能 (I2C2 clock enable during Sleep mode)

由软件置 1 和清零。

0：睡眠模式期间禁止 I2C2 时钟

1：睡眠模式期间使能 I2C2 时钟

位 21 **I2C1LPEN**：睡眠模式期间的 I2C1 时钟使能 (I2C1 clock enable during Sleep mode)

由软件置 1 和清零。

0：睡眠模式期间禁止 I2C1 时钟

1：睡眠模式期间使能 I2C1 时钟

位 20 **UART5LPEN**：睡眠模式期间的 UART5 时钟使能 (UART5 clock enable during Sleep mode)

由软件置 1 和清零。

0：睡眠模式期间禁止 UART5 时钟

1：睡眠模式期间使能 UART5 时钟

位 19 **UART4LPEN**：睡眠模式期间的 UART4 时钟使能 (UART4 clock enable during Sleep mode)

由软件置 1 和清零。

0：睡眠模式期间禁止 UART4 时钟

1：睡眠模式期间使能 UART4 时钟

位 18 **USART3LPEN**：睡眠模式期间的 UART3 时钟使能 (USART3 clock enable during Sleep mode)

由软件置 1 和清零。

0：睡眠模式期间禁止 UART3 时钟

1：睡眠模式期间使能 UART3 时钟

位 17 **USART2LPEN**：睡眠模式期间的 UART2 时钟使能 (USART2 clock enable during Sleep mode)

由软件置 1 和清零。

0：睡眠模式期间禁止 UART2 时钟

1：睡眠模式期间使能 UART2 时钟

位 16 保留，必须保持复位值。

位 15 **SPI3LPEN**：睡眠模式期间的 SPI3 时钟使能 (SPI3 clock enable during Sleep mode)
由软件置 1 和清零。
0：睡眠模式期间禁止 SPI3 时钟
1：睡眠模式期间使能 SPI3 时钟

位 14 **SPI2LPEN**：睡眠模式期间的 SPI2 时钟使能 (SPI2 clock enable during Sleep mode)
由软件置 1 和清零。
0：睡眠模式期间禁止 SPI2 时钟
1：睡眠模式期间使能 SPI2 时钟

位 13:12 保留，必须保持复位值。

位 11 **WWDGLPEN**：睡眠模式期间的窗口看门狗时钟使能 (Window watchdog clock enable during Sleep mode)
由软件置 1 和清零。
0：睡眠模式期间禁止窗口看门狗时钟
1：睡眠模式期间使能窗口看门狗时钟

位 10:9 保留，必须保持复位值。

位 8 **TIM14LPEN**：睡眠模式期间的 TIM14 时钟使能 (TIM14 clock enable during Sleep mode)
由软件置 1 和清零。
0：睡眠模式期间禁止 TIM14 时钟
1：睡眠模式期间使能 TIM14 时钟

位 7 **TIM13LPEN**：睡眠模式期间的 TIM13 时钟使能 (TIM13 clock enable during Sleep mode)
由软件置 1 和清零。
0：睡眠模式期间禁止 TIM13 时钟
1：睡眠模式期间使能 TIM13 时钟

位 6 **TIM12LPEN**：睡眠模式期间的 TIM12 时钟使能 (TIM12 clock enable during Sleep mode)
由软件置 1 和清零。
0：睡眠模式期间禁止 TIM12 时钟
1：睡眠模式期间使能 TIM12 时钟

位 5 **TIM7LPEN**：睡眠模式期间的 TIM7 时钟使能 (TIM7 clock enable during Sleep mode)
由软件置 1 和清零。
0：睡眠模式期间禁止 TIM7 时钟
1：睡眠模式期间使能 TIM7 时钟

位 4 **TIM6LPEN**：睡眠模式期间的 TIM6 时钟使能 (TIM6 clock enable during Sleep mode)
由软件置 1 和清零。
0：睡眠模式期间禁止 TIM6 时钟
1：睡眠模式期间使能 TIM6 时钟

位 3 **TIM5LPEN**：睡眠模式期间的 TIM5 时钟使能 (TIM5 clock enable during Sleep mode)
由软件置 1 和清零。
0：睡眠模式期间禁止 TIM5 时钟
1：睡眠模式期间使能 TIM5 时钟

位 2 **TIM4LPEN**：睡眠模式期间的 TIM4 时钟使能 (TIM4 clock enable during Sleep mode)
由软件置 1 和清零。
0：睡眠模式期间禁止 TIM4 时钟
1：睡眠模式期间使能 TIM4 时钟

位 1 **TIM3LPEN**：睡眠模式期间的 TIM3 时钟使能 (TIM3 clock enable during Sleep mode)
由软件置 1 和清零。
0：睡眠模式期间禁止 TIM3 时钟
1：睡眠模式期间使能 TIM3 时钟

位 0 **TIM2LPEN**：睡眠模式期间的 TIM2 时钟使能 (TIM2 clock enable during Sleep mode)
由软件置 1 和清零。
0：睡眠模式期间禁止 TIM2 时钟
1：睡眠模式期间使能 TIM2 时钟

6.3.24 用于 STM32F42xxx 和 STM32F43xxx 的低功耗模式寄存器中的 RCC APB1 外设时钟使能 (RCC_APB1LPENR)

RCC APB1 peripheral clock enable in low power mode register

偏移地址：0x60

复位值：0x36FE C9FF

访问：无等待周期，按字、半字和字节访问。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UART8 LPEN	UART7 LPEN	DAC LPEN	PWR LPEN	RESERVED	CAN2 LPEN	CAN1 LPEN	Reserved	I2C3 LPEN	I2C2 LPEN	I2C1 LPEN	UART5 LPEN	UART4 LPEN	USART3 LPEN	USART2 LPEN	Reserved
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
SPI3 LPEN	SPI2 LPEN	Reserved		WWDG LPEN	Reserved		TIM14 LPEN	TIM13 LPEN	TIM12 LPEN	TIM7 LPEN	TIM6 LPEN	TIM5 LPEN	TIM4 LPEN	TIM3 LPEN	TIM2 LPEN
rw	rw			rw			rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31 **UART8LPEN**：睡眠模式期间的 UART8 时钟使能 (UART8 clock enable during Sleep mode)
由软件置 1 和清零。
0：睡眠模式期间禁止 UART8 时钟
1：睡眠模式期间使能 UART8 时钟

位 30 **UART7LPEN**：睡眠模式期间的 UART7 时钟使能 (UART7 clock enable during Sleep mode)
由软件置 1 和清零。
0：睡眠模式期间禁止 UART7 时钟
1：睡眠模式期间使能 UART7 时钟

位 29 **DACLPEN**：睡眠模式期间的 DAC 接口时钟使能 (DAC interface clock enable during Sleep mode)
由软件置 1 和清零。
0：睡眠模式期间禁止 DAC 接口时钟
1：睡眠模式期间使能 DAC 接口时钟

位 28 **PWRLPEN**：睡眠模式期间的电源接口时钟使能 (Power interface clock enable during Sleep mode)
由软件置 1 和清零。
0：睡眠模式期间禁止电源接口时钟
1：睡眠模式期间使能电源接口时钟

位 27 保留，必须保持复位值。

位 26 **CAN2LPEN**：睡眠模式期间的 CAN 2 时钟使能 (CAN 2 clock enable during Sleep mode)
由软件置 1 和清零。
0：睡眠模式期间禁止 CAN 2 时钟
1：睡眠模式期间使能 CAN 2 时钟

位 25 **CAN1LPEN**：睡眠模式期间的 CAN 1 时钟使能 (CAN 1 clock enable during Sleep mode)
由软件置 1 和清零。
0：睡眠模式期间禁止 CAN 1 时钟
1：睡眠模式期间使能 CAN 1 时钟

位 24 保留，必须保持复位值。

位 23 **I2C3LPEN**：睡眠模式期间的 I2C3 时钟使能 (I2C3 clock enable during Sleep mode)
由软件置 1 和清零。
0：睡眠模式期间禁止 I2C3 时钟
1：睡眠模式期间使能 I2C3 时钟

位 22 **I2C2LPEN**：睡眠模式期间的 I2C2 时钟使能 (I2C2 clock enable during Sleep mode)
由软件置 1 和清零。
0：睡眠模式期间禁止 I2C2 时钟
1：睡眠模式期间使能 I2C2 时钟

位 21 **I2C1LPEN**：睡眠模式期间的 I2C1 时钟使能 (I2C1 clock enable during Sleep mode)
由软件置 1 和清零。
0：睡眠模式期间禁止 I2C1 时钟
1：睡眠模式期间使能 I2C1 时钟

位 20 **UART5LPEN**：睡眠模式期间的 UART5 时钟使能 (UART5 clock enable during Sleep mode)
由软件置 1 和清零。
0：睡眠模式期间禁止 UART5 时钟
1：睡眠模式期间使能 UART5 时钟

位 19 **UART4LPEN**：睡眠模式期间的 UART4 时钟使能 (UART4 clock enable during Sleep mode)
由软件置 1 和清零。
0：睡眠模式期间禁止 UART4 时钟
1：睡眠模式期间使能 UART4 时钟

位 18 **USART3LPEN**：睡眠模式期间的 UART3 时钟使能 (USART3 clock enable during Sleep mode)
由软件置 1 和清零。
0：睡眠模式期间禁止 UART3 时钟
1：睡眠模式期间使能 UART3 时钟

位 17 **USART2LPEN**：睡眠模式期间的 UART2 时钟使能 (USART2 clock enable during Sleep mode)
由软件置 1 和清零。
0：睡眠模式期间禁止 UART2 时钟
1：睡眠模式期间使能 UART2 时钟

位 16 保留，必须保持复位值。

位 15 **SPI3LPEN**：睡眠模式期间的 SPI3 时钟使能 (SPI3 clock enable during Sleep mode)
由软件置 1 和清零。
0：睡眠模式期间禁止 SPI3 时钟
1：睡眠模式期间使能 SPI3 时钟

位 14 **SPI2LPEN**：睡眠模式期间的 SPI2 时钟使能 (SPI2 clock enable during Sleep mode)
由软件置 1 和清零。
0：睡眠模式期间禁止 SPI2 时钟
1：睡眠模式期间使能 SPI2 时钟

位 13:12 保留，必须保持复位值。

位 11 **WWDGLPEN**：睡眠模式期间的窗口看门狗时钟使能 (Window watchdog clock enable during Sleep mode)

由软件置 1 和清零。
0：睡眠模式期间禁止窗口看门狗时钟
1：睡眠模式期间使能窗口看门狗时钟

位 10:9 保留，必须保持复位值。

位 8 **TIM14LPEN**：睡眠模式期间的 TIM14 时钟使能 (TIM14 clock enable during Sleep mode)

由软件置 1 和清零。
0：睡眠模式期间禁止 TIM14 时钟
1：睡眠模式期间使能 TIM14 时钟

位 7 **TIM13LPEN**：睡眠模式期间的 TIM13 时钟使能 (TIM13 clock enable during Sleep mode)

由软件置 1 和清零。
0：睡眠模式期间禁止 TIM13 时钟
1：睡眠模式期间使能 TIM13 时钟

位 6 **TIM12LPEN**：睡眠模式期间的 TIM12 时钟使能 (TIM12 clock enable during Sleep mode)

由软件置 1 和清零。
0：睡眠模式期间禁止 TIM12 时钟
1：睡眠模式期间使能 TIM12 时钟

位 5 **TIM7LPEN**：睡眠模式期间的 TIM7 时钟使能 (TIM7 clock enable during Sleep mode)

由软件置 1 和清零。
0：睡眠模式期间禁止 TIM7 时钟
1：睡眠模式期间使能 TIM7 时钟

位 4 **TIM6LPEN**：睡眠模式期间的 TIM6 时钟使能 (TIM6 clock enable during Sleep mode)

由软件置 1 和清零。
0：睡眠模式期间禁止 TIM6 时钟
1：睡眠模式期间使能 TIM6 时钟

位 3 **TIM5LPEN**：睡眠模式期间的 TIM5 时钟使能 (TIM5 clock enable during Sleep mode)

由软件置 1 和清零。
0：睡眠模式期间禁止 TIM5 时钟
1：睡眠模式期间使能 TIM5 时钟

位 2 **TIM4LPEN**：睡眠模式期间的 TIM4 时钟使能 (TIM4 clock enable during Sleep mode)

由软件置 1 和清零。
0：睡眠模式期间禁止 TIM4 时钟
1：睡眠模式期间使能 TIM4 时钟

位 1 **TIM3LPEN**：睡眠模式期间的 TIM3 时钟使能 (TIM3 clock enable during Sleep mode)

由软件置 1 和清零。
0：睡眠模式期间禁止 TIM3 时钟
1：睡眠模式期间使能 TIM3 时钟

位 0 **TIM2LPEN**：睡眠模式期间的 TIM2 时钟使能 (TIM2 clock enable during Sleep mode)

由软件置 1 和清零。
0：睡眠模式期间禁止 TIM2 时钟
1：睡眠模式期间使能 TIM2 时钟

6.3.25 用于 STM32F405xx/07xx 和 STM32F415xx/17xx 的低功耗模式寄存器中的 RCC APB2 外设时钟使能 (RCC_APB2LPENR)

RCC APB2 peripheral clock enabled in low power mode register

偏移地址：0x64

复位值：0x0007 5F33

访问：无等待周期，按字、半字和字节访问。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved													TIM11 LPEN	TIM10 LPEN	TIM9 LPEN
													rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reser-ved	SYSCFG LPEN	Reser-ved	SPI1 LPEN	SDIO LPEN	ADC3 LPEN	ADC2 LPEN	ADC1 LPEN	Reserved		USART6 LPEN	USART1 LPEN	Reserved		TIM8 LPEN	TIM1 LPEN
	rw		rw	rw	rw	rw	rw			rw	rw			rw	rw

位 31:19 保留，必须保持复位值。

位 18 **TIM11LPEN**：睡眠模式期间的 TIM11 时钟使能 (TIM11 clock enable during Sleep mode)
由软件置 1 和清零。
0：睡眠模式期间禁止 TIM11 时钟
1：睡眠模式期间使能 TIM11 时钟

位 17 **TIM10LPEN**：睡眠模式期间的 TIM10 时钟使能 (TIM10 clock enable during Sleep mode)
由软件置 1 和清零。
0：睡眠模式期间禁止 TIM10 时钟
1：睡眠模式期间使能 TIM10 时钟

位 16 **TIM9LPEN**：睡眠模式期间的 TIM9 时钟使能 (TIM9 clock enable during sleep mode)
由软件置 1 和清零。
0：睡眠模式期间禁止 TIM9 时钟
1：睡眠模式期间使能 TIM9 时钟

位 15 保留，必须保持复位值。

位 14 **SYSCFGLPEN**：睡眠模式期间的系统配置控制器时钟使能 (System configuration controller clock enable during Sleep mode)
由软件置 1 和清零。
0：睡眠模式期间禁止系统配置控制器时钟
1：睡眠模式期间使能系统配置控制器时钟

位 13 保留，必须保持复位值。

位 12 **SPI1LPEN**：睡眠模式期间的 SPI1 时钟使能 (SPI1 clock enable during Sleep mode)
由软件置 1 和清零。
0：睡眠模式期间禁止 SPI1 时钟
1：睡眠模式期间使能 SPI1 时钟

位 11 **SDIOLPEN**：睡眠模式期间的 SDIO 时钟使能 (SDIO clock enable during Sleep mode)

由软件置 1 和清零。

0：睡眠模式期间禁止 SDIO 模块时钟

1：睡眠模式期间使能 SDIO 模块时钟

位 10 **ADC3LPEN**：睡眠模式期间的 ADC 3 时钟使能 (ADC 3 clock enable during Sleep mode)

由软件置 1 和清零。

0：睡眠模式期间禁止 ADC 3 时钟

1：睡眠模式期间使能 ADC 3 时钟

位 9 **ADC2LPEN**：睡眠模式期间的 ADC2 时钟使能 (ADC2 clock enable during Sleep mode)

由软件置 1 和清零。

0：睡眠模式期间禁止 ADC2 时钟

1：睡眠模式期间使能 ADC2 时钟

位 8 **ADC1LPEN**：睡眠模式期间的 ADC1 时钟使能 (ADC1 clock enable during Sleep mode)

由软件置 1 和清零。

0：睡眠模式期间禁止 ADC1 时钟

1：睡眠模式期间使能 ADC1 时钟

位 7:6 保留，必须保持复位值。

位 5 **USART6LPEN**：睡眠模式期间的 USART6 时钟使能 (USART6 clock enable during Sleep mode)

由软件置 1 和清零。

0：睡眠模式期间禁止 USART6 时钟

1：睡眠模式期间使能 USART6 时钟

位 4 **USART1LPEN**：睡眠模式期间的 USART1 时钟使能 (USART1 clock enable during Sleep mode)

由软件置 1 和清零。

0：睡眠模式期间禁止 USART1 时钟

1：睡眠模式期间使能 USART1 时钟

位 3:2 保留，必须保持复位值。

位 1 **TIM8LPEN**：睡眠模式期间的 TIM8 时钟使能 (TIM8 clock enable during Sleep mode)

由软件置 1 和清零。

0：睡眠模式期间禁止 TIM8 时钟

1：睡眠模式期间使能 TIM8 时钟

位 0 **TIM1LPEN**：睡眠模式期间的 TIM1 时钟使能 (TIM1 clock enable during Sleep mode)

由软件置 1 和清零。

0：睡眠模式期间禁止 TIM1 时钟

1：睡眠模式期间使能 TIM1 时钟

6.3.26 用于 STM32F42xxx 和 STM32F43xxx 的低功耗模式寄存器中的 RCC APB2 外设时钟 (RCC_APB2LPENR)

RCC APB2 peripheral clock enabled in low power mode register

偏移地址：0x64

复位值：0x0007 5F33

访问：无等待周期，按字、半字和字节访问。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved										SPI6 LPEN	SPI5 LPEN	Reser-ved	TIM11 LPEN	TIM10 LPEN	TIM9 LPEN
										rw	rw		rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reser-ved	SYSCFG LPEN	SPI4 LPEN	SPI1 LPEN	SDIO LPEN	ADC3 LPEN	ADC2 LPEN	ADC1 LPEN	Reserved		USART6 LPEN	USART1 LPEN	Reserved		TIM8 LPEN	TIM1 LPEN
	rw	rw	rw	rw	rw	rw	rw			rw	rw			rw	rw

位 31:22 保留，必须保持复位值。

位 21 **SPI6LPEN**：睡眠模式期间的 SPI6 时钟使能 (SPI6 clock enable during Sleep mode)
由软件置 1 和清零。
0：睡眠模式期间禁止 SPI6 时钟
1：睡眠模式期间使能 SPI6 时钟

位 20 **SPI5LPEN**：睡眠模式期间的 SPI5 时钟使能
由软件置 1 和清零。
0：睡眠模式期间禁止 SPI5 时钟
1：睡眠模式期间使能 SPI5 时钟

位 19 保留，必须保持复位值。

位 18 **TIM11LPEN**：睡眠模式期间的 TIM11 时钟使能 (TIM11 clock enable during Sleep mode)
由软件置 1 和清零。
0：睡眠模式期间禁止 TIM11 时钟
1：睡眠模式期间使能 TIM11 时钟

位 17 **TIM10LPEN**：睡眠模式期间的 TIM10 时钟使能 (TIM10 clock enable during Sleep mode)
由软件置 1 和清零。
0：睡眠模式期间禁止 TIM10 时钟
1：睡眠模式期间使能 TIM10 时钟

位 16 **TIM9LPEN**：睡眠模式期间的 TIM9 时钟使能 (TIM9 clock enable during sleep mode)
由软件置 1 和清零。
0：睡眠模式期间禁止 TIM9 时钟
1：睡眠模式期间使能 TIM9 时钟

位 15 保留，必须保持复位值。

位 14 **SYSCFGLPEN**：睡眠模式期间的系统配置控制器时钟使能 (System configuration controller clock enable during Sleep mode)
由软件置 1 和清零。
0：睡眠模式期间禁止系统配置控制器时钟
1：睡眠模式期间使能系统配置控制器时钟

位 13 **SPI4LPEN**：睡眠模式期间的 SPI4 时钟使能 (SPI4 clock enable during Sleep mode)
由软件置 1 和清零。
0：睡眠模式期间禁止 SPI4 时钟
1：睡眠模式期间使能 SPI4 时钟

位 12 **SPI1LPEN**：睡眠模式期间的 SPI1 时钟使能 (SPI1 clock enable during Sleep mode)
由软件置 1 和清零。
0：睡眠模式期间禁止 SPI1 时钟
1：睡眠模式期间使能 SPI1 时钟

位 11 **SDIOLPEN**：睡眠模式期间的 SDIO 时钟使能 (SDIO clock enable during Sleep mode)
由软件置 1 和清零。
0：睡眠模式期间禁止 SDIO 模块时钟
1：睡眠模式期间使能 SDIO 模块时钟

位 10 **ADC3LPEN**：睡眠模式期间的 ADC 3 时钟使能 (ADC 3 clock enable during Sleep mode)
由软件置 1 和清零。
0：睡眠模式期间禁止 ADC 3 时钟
1：睡眠模式期间使能 ADC 3 时钟

位 9 **ADC2LPEN**：睡眠模式期间的 ADC2 时钟使能 (ADC2 clock enable during Sleep mode)
由软件置 1 和清零。
0：睡眠模式期间禁止 ADC2 时钟
1：睡眠模式期间使能 ADC2 时钟

位 8 **ADC1LPEN**：睡眠模式期间的 ADC1 时钟使能 (ADC1 clock enable during Sleep mode)
由软件置 1 和清零。
0：睡眠模式期间禁止 ADC1 时钟
1：睡眠模式期间使能 ADC1 时钟

位 7:6 保留，必须保持复位值。

位 5 **USART6LPEN**：睡眠模式期间的 USART6 时钟使能 (USART6 clock enable during Sleep mode)
由软件置 1 和清零。
0：睡眠模式期间禁止 USART6 时钟
1：睡眠模式期间使能 USART6 时钟

位 4 **USART1LPEN**：睡眠模式期间的 USART1 时钟使能 (USART1 clock enable during Sleep mode)
由软件置 1 和清零。
0：睡眠模式期间禁止 USART1 时钟
1：睡眠模式期间使能 USART1 时钟

位 3:2 保留，必须保持复位值。

位 1 **TIM8LPEN**：睡眠模式期间的 TIM8 时钟使能 (TIM8 clock enable during Sleep mode)
由软件置 1 和清零。
0：睡眠模式期间禁止 TIM8 时钟
1：睡眠模式期间使能 TIM8 时钟

位 0 **TIM1LPEN**：睡眠模式期间的 TIM1 时钟使能 (TIM1 clock enable during Sleep mode)
由软件置 1 和清零。
0：睡眠模式期间禁止 TIM1 时钟
1：睡眠模式期间使能 TIM1 时钟

6.3.27 RCC 备份域控制寄存器 (RCC_BDCR)

RCC Backup domain control register

偏移地址：0x70

复位值：0x0000 0000，通过备份域复位进行复位。
访问：0 ≤ 等待周期 ≤ 3，按字、半字和字节访问连续访问该寄存器时，则插入等待周期。

RCC 备份域控制寄存器 (RCC_BDCR) 中的 LSEON 位、LSEBYP 位、RTCSEL 位和 RTCEN 位在备份域中。因而复位后，这些位受写保护，必须在这些位可修改前将 *Power control register (PWR_CR)* 中的 DBP 位置 1。有关详细信息，请参见 *第87 页的第 5.1.2 节*。只有备份域复位后，这些位才将复位（请参见 *第 6.1.3 节：备份域复位*）。内部复位和外部复位对这些位不会有任何影响。

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

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

位 31:17 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 和清零。
0：复位未激活
1：复位整个备份域
注意：BKPSRAM 不受此复位影响，只能在 Flash 保护级别从级别 1 更改为级别 0 时复位 BKPSRAM。

位 15 **RTCEN**：RTC 时钟使能 (RTC clock enable)
由软件置 1 和清零。
0：RTC 时钟禁止
1：RTC 时钟使能

位 14:10 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)
由软件置 1，用于选择 RTC 的时钟源。选择 RTC 时钟源后，除非备份域复位，否则其不可再更改。可使用 BDRST 位对其进行复位。
00：无时钟
01：LSE 振荡器时钟用作 RTC 时钟
10：LSI 振荡器时钟用作 RTC 时钟
11：由可编程预分频器分频的 HSE 振荡器时钟（通过 RCC 时钟配置寄存器 (RCC_CFGR) 中的 RTCPRE[4:0] 位选择）用作 RTC 时钟

位 7:3 保留，必须保持复位值。

位 2 **LSEBYP**：外部低速振荡器旁路 (External low-speed oscillator bypass)
由软件置 1 和清零，用于旁路调试模式下的振荡器。只有在禁止 LSE 时钟后才能写入该位。
0：不旁路 LSE 振荡器
1：旁路 LSE 振荡器

位 1 **LSERDY**：外部低速振荡器就绪 (External low-speed oscillator ready)

由硬件置 1 和清零，用于指示外部 32 kHz 振荡器已稳定。在 LSEON 位被清零后，LSERDY 将在 6 个外部低速振荡器时钟周期后转为低电平。

0：LSE 时钟未就绪

1：LSE 时钟就绪

位 0 **LSEON**：外部低速振荡器使能 (External low-speed oscillator enable)

由软件置 1 和清零。

0：LSE 时钟关闭

1：LSE 时钟开启

6.3.28 RCC 时钟控制和状态寄存器 (RCC_CSR)

RCC clock control & status register

偏移地址：0x74

复位值：0x0E00 0000，除复位标志只能通过电源复位进行复位以外，其它通过系统复位进行复位。

访问：0 ≤ 等待周期 ≤ 3，按字、半字和字节访问

连续访问该寄存器时，则插入等待周期。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LPWR RSTF	WWDG RSTF	IWDG RSTF	SFT RSTF	POR RSTF	PIN RSTF	BORRS TF	RMVF	Reserved							
rw	rw	rw	rw	rw	rw	rw	rw								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														LSIRDY	LSION
														r	rw

位 31 **LPWRRSTF**：低功耗复位标志 (Low-power reset flag)

发生低功耗管理复位时，由硬件置 1。

通过写入 RMVF 位清零。

0：未发生低功耗管理复位

1：发生低功耗管理复位

有关低功耗管理复位的详细信息，请参见*低功耗管理复位*。

位 30 **WWDGRSTF**：窗口看门狗复位标志 (Window watchdog reset flag)

发生窗口看门狗复位时，由硬件置 1。

通过写入 RMVF 位清零。

0：未发生窗口看门狗复位

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)

发生来自 V_{DD} 域的独立看门狗复位时，由硬件置 1。

通过写入 RMVF 位清零。

0：未发生看门狗复位

1：发生看门狗复位

位 28 **SFTRSTF**：软件复位标志 (Software reset flag)

发生软件复位时，由硬件置 1。

通过写入 RMVF 位清零。

0：未发生软件复位

1：发生软件复位

位 27 **PORRSTF**：POR/PDR 复位标志 (POR/PDR reset flag)

发生 POR/PDR 复位时，由硬件置 1。

通过写入 RMVF 位清零。

0：未发生 POR/PDR 复位

1：发生 POR/PDR 复位

位 26 **PINRSTF**：引脚复位标志 (PIN reset flag)

发生来自 NRST 引脚的复位时，由硬件置 1。

通过写入 RMVF 位清零。

0：未发生来自 NRST 引脚的复位

1：发生来自 NRST 引脚的复位

位 25 **BORRSTF**：BOR 复位标志 (BOR reset flag)

通过软件写入 RMVF 位清零。

发生 POR/PDR 复位或 BOR 复位时，由硬件置 1。

0：未发生 POR/PDR 复位或 BOR 复位

1：发生 POR/PDR 复位或 BOR 复位

位 24 **RMVF**：清除复位标志 (Remove reset flag)

由软件置 1，用于将复位标志清零。

0：无操作

1：清零复位标志

位 23:2 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 和清零，用于指示内部 RC 40 kHz 振荡器已稳定。在 LSION 位被清零后，LSIRDY 将在 3 个 LSI 时钟周期后转为低电平。

0：LSI RC 振荡器未就绪

1：LSI RC 振荡器就绪

位 0 **LSION**：内部低速振荡器使能 (Internal low-speed oscillator enable)

由软件置 1 和清零。

0：LSI RC 振荡器关闭

1：LSI RC 振荡器开启

6.3.29 RCC 扩频时钟生成寄存器 (RCC_SSCGR)

RCC spread spectrum clock generation register

偏移地址：0x80

复位值：0x0000 0000

访问：无等待周期，按字、半字和字节访问。

扩频时钟生成仅适用于主 PLL。

RCC_SSCGR 的写操作必须在使能主 PLL 之前或禁止主 PLL 之后进行。

注意： *有关 PLL 扩频时钟生成 (SSCG) 特性的所有详细信息，请参见器件数据表中的“电气特性”部分。*

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SSCGEN	SPREADSEL	Reserved		INCSTEP											
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
INCSTEP			MODPER												
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31 **SSCGEN**：扩频调制使能 (Spread spectrum modulation enable)

由软件置 1 和清零。

0：扩频调制禁止。（CR[24]=PLLON 位清零后写入）

1：扩频调制使能。（CR[24]=PLLON 位置 1 前写入）

位 30 **SPREADSEL**：扩频选择 (Spread Select)

由软件置 1 和清零。

CR[24]=PLLON 位置 1 前写入。

0：中心扩频

1：向下扩频

位 29:28 保留，必须保持复位值。

位 27:13 **INCSTEP**：增量步长 (Incrementation step)

由软件置 1 和清零。CR[24]=PLLON 位置 1 前写入。

设置调制曲线振幅输入。

位 12:0 **MODPER**：调制周期 (Modulation period)

由软件置 1 和清零。CR[24]=PLLON 位置 1 前写入。

设置调制曲线周期输入。

6.3.30 RCC PLLI2S 配置寄存器 (RCC_PLLI2SCFGR)

RCC PLLI2S configuration register

偏移地址：0x84

复位值：0x2000 3000

访问：无等待周期，按字、半字和字节访问。

此寄存器用于根据公式式配置 PLLI2S 时钟输出：

- $f_{(VCO\ 时钟)} = f_{(PLLI2S\ 时钟输入)} \times (PLLI2SN / PLLM)$
- $f_{(PLL\ I2S\ 时钟输出)} = f_{(VCO\ 时钟)} / PLLI2SR$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved	PLLI2SR2	PLLI2SR1	PLLI2SR0	Reserved											
	rw	rw	rw												

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	PLLI2SN8	PLLI2SN7	PLLI2SN6	PLLI2SN5	PLLI2SN4	PLLI2SN3	PLLI2SN2	PLLI2SN1	PLLI2SN0	Reserved					
	rw	rw	rw	rw	rw	rw	rw	rw	rw						

位 31 保留，必须保持复位值。

位 30:28 **PLLI2SR**：适用于 I2S 时钟的 PLLI2S 分频系数 (PLLI2S division factor for I2S clocks)

由软件置 1 和清零，用于控制 I2S 时钟频率。这些位应仅在 PLLI2S 已禁止时写入。选择的系数必须与 I2S 外设中的预分频值一致，以达到使用标准晶振时误差不超过 0.3%，使用音频晶振时误差为 0%。有关 I2S 时钟频率和精度的详细信息，请参见 I2S 一章的*第 27.4.4 节：时钟发生器*。

小心： I2S 的正常工作频率要求低于或等于 192 MHz。

I2S 时钟频率 = VCO 频率 × PLLR 并且 2 ≤ PLLR ≤ 7

000：PLLR = 0，错误配置
001：PLLR = 1，错误配置
010：PLLR = 2
...
111：PLLR = 7

位 27:15 保留，必须保持复位值。

位 14:6 **PLLI2SN**：用于 VCO 的 PLLI2S 倍频系数 (PLLI2S multiplication factor for VCO)

由软件置 1 和清零，用于控制 VCO 的倍频系数。这些位只能在 PLLI2S 已禁止时写入。写入这些位时只允许使用半字和字访问。

小心： 软件必须正确设置这些位，确保 VCO 输出频率介于 192 和 432 MHz 之间。

VCO 输出频率 = VCO 输入频率 × PLLI2SN 并且 192 ≤ PLLI2SN ≤ 432

000000000：PLLI2SN = 0，错误配置
000000001：PLLI2SN = 1，错误配置
...
011000000：PLLI2SN = 192
011000001：PLLI2SN = 193
011000010：PLLI2SN = 194
...
110110000：PLLI2SN = 432
110110000：PLLI2SN = 433，错误配置
...
111111111：PLLI2SN = 511，错误配置

位 5:0 保留，必须保持复位值。

6.3.31 RCC 专用时钟配置寄存器 (RCC_DCKCFGR)

RCC Dedicated Clocks Configuration Register

此寄存器仅适用于 STM32F42xxx 和 STM32F43xxx 设备。

偏移地址：0x8C

复位值：0x0000 0000

访问：无等待周期，按字、半字和字节访问。

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

位 31:25 保留，必须保持复位值。

位 24 **TIMPRE**：定时器时钟预分频器选择 (Timers clocks prescalers selection)

由软件置 1 和复位，用于控制连接到 APB1 和 APB2 域的所有定时器的时钟频率。

0：如果 APB 预分频器（RCC_CFGR 寄存器中的 PPRE1、PPRE2）的分频系数配置为 1，则 TIMxCLK = PCLKx。否则，定时器时钟频率将设置为与定时器相连的 APB 域的频率的两倍：

TIMxCLK = 2xPCLKx。

如果 APB 预分频器（RCC_CFGR 寄存器中的 PPRE1、PPRE2）的分频系数配置为 1、2 或 4，则 TIMxCLK = HCLK。否则，定时器时钟频率将设置为与定时器相连的 APB 域的频率的四倍：

TIMxCLK = 4xPCLKx。

位 23：0 保留，必须保持复位值。

6.3.32 RCC 寄存器映射

表 26 给出了寄存器映射和复位值。

表 26. RCC 寄存器映射和复位值用于 STM32F405xx/07xx 和 STM32F415xx/17xx

偏移地址	寄存器名称	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	RCC_CR	Reserved				PLL I2SRDY	PLL I2SON	PLL RDY	PLL ON	Reserved				CSSON	HSEBYP	HSERDY	HSEON	HSICAL 7	HSICAL 6	HSICAL 5	HSICAL 4	HSICAL 3	HSICAL 2	HSICAL 1	HSICAL 0	HSITRIM 4	HSITRIM 3	HSITRIM 2	HSITRIM 1	HSITRIM 0	Reserved	HSIRDY	HSION
0x04	RCC_PLLCFGR	Reserved				PLLQ 3	PLLQ 2	PLLQ 1	PLLQ 0	Reserved	PLLSRC	Reserved				PLLP 1	PLLP 0	Reserved	PLLN 8	PLLN 7	PLLN 6	PLLN 5	PLLN 4	PLLN 3	PLLN 2	PLLN 1	PLLN 0	PLLM 5	PLLM 4	PLLM 3	PLLM 2	PLLM 1	PLLM 0
0x08	RCC_CFGR	MCO2 1	MCO2 0	MCO2PRE2	MCO2PRE1	MCO2PRE0	MCO1PRE2	MCO1PRE1	MCO1PRE0	I2SSRC	MCO1 1	MCO1 0	RTCPRE 4	RTCPRE 3	RTCPRE 2	RTCPRE 1	RTCPRE 0	PPRE2 2	PPRE2 1	PPRE2 0	PPRE1 2	PPRE1 1	PPRE1 0	Reserved		HPRE 3	HPRE 2	HPRE 1	HPRE 0	SWS 1	SWS 0	SW 1	SW 0	
0x0C	RCC_CIR	Reserved								CSSC	Reserved	PLLI2SRDYC	PLLRDYC	HSERDYC	HSIRDYC	LSERDYC	LSIRDYC	Reserved		PLLI2SRDYIE	PLLRDYIE	HSERDYIE	HSIRDYIE	LSERDYIE	LSIRDYIE	CSSF	Reserved	PLLI2SRDYF	PLLRDYF	HSERDYF	HSIRDYF	LSERDYF	LSIRDYF
0x10	RCC_AHB1RSTR	Reserved		OTGHSRST	Reserved			ETHMACRST	Reserved		DMA2RST	DMA1RST	Reserved								CRCRST	Reserved			GPIOIRST	GPIOHRST	GPIOGRST	GPIOFRST	GPIOERST	GPIODRST	GPIOCRST	GPIOBRST	GPIOARST
0x14	RCC_AHB2RSTR	Reserved																								OTGFSRST	RNGRST	HSAHRST	CRYPRST	Reserved			DCMIRST
0x18	RCC_AHB3RSTR	Reserved																															FSMCRST
0x1C	Reserved	Reserved																															
0x20	RCC_APB1RSTR	Reserved		DACRST	PWRRST	Reserved	CAN2RST	CAN1RST	Reserved	I2C3RST	I2C2RST	I2C1RST	UART5RST	UART4RST	UART3RST	UART2RST	Reserved	SPI3RST	SPI2RST	Reserved		WWDGRST	Reserved		TIM14RST	TIM13RST	TIM12RST	TIM7RST	TIM6RST	TIM5RST	TIM4RST	TIM3RST	TIM2RST
0x24	RCC_APB2RSTR	Reserved												TIM11RST	TIM10RST	TIM9RST	Reserved	SYSCFGRST	Reserved	SPI1RST	SDIORST	Reserved		ADCRST	Reserved		USART6RST	USART1RST	Reserved		TIM8RST	TIM1RST
0x28	Reserved	Reserved																															
0x2C	Reserved	Reserved																															
0x30	RCC_AHB1ENR	Reserved	OTGHSULPIEN	OTGHSEN	ETHMACPTPEN	ETHMACRXEN	ETHMACTXEN	ETHMACEN	Reserved		DMA2EN	DMA1EN	CCMDATARAMEN	Reserved	BKPSRAMEN	Reserved		CRCEN	Reserved			GPIOIEN		GPIOHEN	GPIOGEN	GPIOFEN	GPIOEEN	GPIODEN	GPIOCEN	GPIOBEN	GPIOAEN		
0x34	RCC_AHB2ENR	Reserved																								OTGFSEN	RNGEN	HASHEN	CRYPEN	Reserved			DCMIEN
0x38	RCC_AHB3ENR	Reserved																															FSMCEN
0x3C	Reserved	Reserved																															

表 26. RCC 寄存器映射和复位值用于 STM32F405xx/07xx 和 STM32F415xx/17xx（续）

偏移地址	寄存器名称	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x40	RCC_APB1ENR	Reserved		DACEN	PWREN	Reserved	CAN2EN	CAN1EN	Reserved	I2C3EN	I2C2EN	I2C1EN	UART5EN	UART4EN	USART3EN	USART2EN	Reserved	SPI3EN	SPI2EN	Reserved		WWDGEN	Reserved		TIM14EN	TIM13EN	TIM12EN	TIM7EN	TIM6EN	TIM5EN	TIM4EN	TIM3EN	TIM2EN
0x44	RCC_APB2ENR	Reserved													TIM11EN	TIM10EN	TIM9EN	Reserved	SYSCFGEN	Reserved	SPI1EN	SDIOEN	ADC3EN	ADC2EN	ADC1EN	Reserved		USART6EN	USART1EN	Reserved		TIM8EN	TIM1EN
0x48	Reserved	Reserved																															
0x4C	Reserved	Reserved																															
0x50	RCC_AHB1LPENR	Reserved	OTGHSULPILPEN	OTGHSLPEN	ETHMACPTPLPEN	ETHMACRXLPEN	ETHMACTXLPEN	ETHMACLPEN	Reserved	DMA2LPEN	DMA1LPEN	Reserved			BKPSRAMLPEN	SRAM2LPEN	SRAM1LPEN	FLITFLPEN	Reserved	CRCLPEN	Reserved				GPIOILPEN	GPIOHLPEN	GPIOGLPEN	GPIOFLPEN	GPIOELPEN	GPIODLPEN	GPIOCLPEN	GPIOBLPEN	GPIOALPEN
0x54	RCC_AHB2LPENR	Reserved																							OTGFSLPEN	RNGLPEN	HASHLPEN	CRYPLPEN	Reserved				DCMILPEN
0x58	RCC_AHB3LPENR	Reserved																															FSMCLPEN
0x5C	Reserved	Reserved																															
0x60	RCC_APB1LPENR	Reserved		DACLPEN	PWRLPEN	Reserved	CAN2LPEN	CAN1LPEN	Reserved	I2C3LPEN	I2C2LPEN	I2C1LPEN	UART5LPEN	UART4LPEN	USART3LPEN	USART2LPEN	Reserved	SPI3LPEN	SPI2LPEN	Reserved		WWDGLPEN	Reserved		TIM14LPEN	TIM13LPEN	TIM12LPEN	TIM7LPEN	TIM6LPEN	TIM5LPEN	TIM4LPEN	TIM3LPEN	TIM2LPEN
0x64	RCC_APB2LPENR	Reserved													TIM11LPEN	TIM10LPEN	TIM9LPEN	Reserved	SYSCFGLPEN	Reserved	SPI1LPEN	SDIOLPEN	ADC3LPEN	ADC2LPEN	ADC1LPEN	Reserved		USART6LPEN	USART1LPEN	Reserved		TIM8LPEN	TIM1LPEN
0x68	Reserved	Reserved																															
0x6C	Reserved	Reserved																															
0x70	RCC_BDCR	Reserved															BDRST	RTCEN	Reserved				RTCSEL 1	RTCSEL 0	Reserved							LSEBYP	LSERDY	LSEON
0x74	RCC_CSR	LPWRRSTF	WWDGRSTF	WDGRSTF	SFTRSTF	PORRSTF	PADRSTF	BORRSTF	RMVF	Reserved																						LSIRDY	LSION
0x78	Reserved	Reserved																															
0x7C	Reserved	Reserved																															
0x80	RCC_SSCGR	SSCGEN	SPREADSEL	Reserved					INCSTEP														MODPER										
0x84	RCC_PLLI2SCFGR	Reserved	PLLI2SRx						Reserved										PLLI2SNx											Reserved				

表 27. RCC 寄存器映射和复位值用于 STM32F42xxx 和 STM32F43xxx

偏移地址	寄存器名称	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	RCC_CR	Reserved				PLL I2SRDY	PLL I2SON	PLL RDY	PLL ON	Reserved				CSSON	HSEBYP	HSERDY	HSEON	HSICAL 7	HSICAL 6	HSICAL 5	HSICAL 4	HSICAL 3	HSICAL 2	HSICAL 1	HSICAL 0	HSITRIM 4	HSITRIM 3	HSITRIM 2	HSITRIM 1	HSITRIM 0	Reserved	HSIRDY	HSION
0x04	RCC_PLLCFGR	Reserved				PLLQ 3	PLLQ 2	PLLQ 1	PLLQ 0	Reserved	PLLSRC	Reserved				PLLP 1	PLLP 0	Reserved	PLLN 8	PLLN 7	PLLN 6	PLLN 5	PLLN 4	PLLN 3	PLLN 2	PLLN 1	PLLN 0	PLLM 5	PLLM 4	PLLM 3	PLLM 2	PLLM 1	PLLM 0
0x08	RCC_CFGR	MCO2 1	MCO2 0	MCO2PRE2	MCO2PRE1	MCO2PRE0	MCO1PRE2	MCO1PRE1	MCO1PRE0	I2SSRC	MCO1 1	MCO1 0	RTCPRE 4	RTCPRE 3	RTCPRE 2	RTCPRE 1	RTCPRE 0	PPRE2 2	PPRE2 1	PPRE2 0	PPRE1 2	PPRE1 1	PPRE1 0	Reserved		HPRE 3	HPRE 2	HPRE 1	HPRE 0	SWS 1	SWS 0	SW 1	SW 0	
0x0C	RCC_CIR	Reserved								CSSC	Reserved	PLLI2SRDYC	PLLRDYC	HSERDYC	HSIRDYC	LSERDYC	LSIRDYC	Reserved		PLLI2SRDYIE	PLLRDYIE	HSERDYIE	HSIRDYIE	LSERDYIE	LSIRDYIE	CSSF	Reserved	PLLI2SRDYF	PLLRDYF	HSERDYF	HSIRDYF	LSERDYF	LSIRDYF
0x10	RCC_AHB1RSTR	Reserved		OTGHSRST	Reserved			ETHMACRST	Reserved	DMA2RST	DMA1RST	Reserved									CRCRST	Reserved			GPIOIRST	GPIOHRST	GPIOGRST	GPIOFRST	GPIOERST	GPIODRST	GPIOCRST	GPIOBRST	GPIOARST
0x14	RCC_AHB2RSTR	Reserved																							OTGFSRST	RNGRST	HSAHRST	CRYPRST	Reserved				DCMIRST
0x18	RCC_AHB3RSTR	Reserved																															FSMCRST
0x1C	Reserved	Reserved																															
0x20	RCC_APB1RSTR	UART8RST	UART7RST	DACRST	PWRRST	Reserved	CAN2RST	CAN1RST	Reserved	I2C3RST	I2C2RST	I2C1RST	UART5RST	UART4RST	UART3RST	UART2RST	Reserved	SPI3RST	SPI2RST	Reserved		WWDGRST	Reserved		TIM14RST	TIM13RST	TIM12RST	TIM7RST	TIM6RST	TIM5RST	TIM4RST	TIM3RST	TIM2RST
0x24	RCC_APB2RSTR	Reserved										SPI6RST	SPI5RST	Reserved	TIM11RST	TIM10RST	TIM9RST	Reserved	SYSCFGRST	SP45RST	SPI1RST	SDIORST	Reserved		ADCRST	Reserved		USART6RST	USART1RST	Reserved		TIM8RST	TIM1RST
0x28	Reserved	Reserved																															
0x2C	Reserved	Reserved																															
0x30	RCC_AHB1ENR	Reserved	OTGHSULPIEN	OTGHSEN	ETHMACPTPEN	ETHMACRXEN	ETHMACTXEN	ETHMACEN	Reserved	DMA2EN	DMA1EN	CCMDATARAMEN	Reserved	BKPSRAMEN	Reserved						CRCEN	Reserved			GPIOIEN	GPIOHEN	GPIOGEN	GPIOFEN	GPIOEEN	GPIODEN	GPIOCEN	GPIOBEN	GPIOAEN
0x34	RCC_AHB2ENR	Reserved																							OTGFSEN	RNGEN	HASHEN	CRYPEN	Reserved				DCMIEN
0x38	RCC_AHB3ENR	Reserved																															FSMCEN
0x3C	Reserved	Reserved																															
0x40	RCC_APB1ENR	SPI8EN	SPI7EN	DACEN	PWREN	Reserved	CAN2EN	CAN1EN	Reserved	I2C3EN	I2C2EN	I2C1EN	UART5EN	UART4EN	USART3EN	USART2EN	Reserved	SPI3EN	SPI2EN	Reserved		WWDGEN	Reserved		TIM14EN	TIM13EN	TIM12EN	TIM7EN	TIM6EN	TIM5EN	TIM4EN	TIM3EN	TIM2EN

表 27. RCC 寄存器映射和复位值用于 STM32F42xxx 和 STM32F43xxx（续）

偏移地址	寄存器名称	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x44	RCC_APB2ENR	Reserved										SPI6EN	SPI5EN	Reserved	TIM11EN	TIM10EN	TIM9EN	Reserved	SYSCFGEN	SPI4EN	SPI1EN	SDIOEN	ADC3EN	ADC2EN	ADC1EN	Reserved		USART6EN	USART1EN	Reserved		TIM8EN	TIM1EN
0x48	Reserved	Reserved																															
0x4C	Reserved	Reserved																															
0x50	RCC_AHB1LPENR	Reserved	OTGHSULPILPEN	OTGHSLPEN	ETHMACPTPLPEN	ETHMACRXLPEN	ETHMACTXLPEN	ETHMACLPEN	Reserved	DMA2LPEN	DMA1LPEN	Reserved	SRAM3LPEN	BKPSRAMLPEN	SRAM2LPEN	SRAM1LPEN	FLITFLPEN	Reserved	CRCLPEN	Reserved					GPIOILPEN	GPIOHLPEN	GPIOGLPEN	GPIOFLPEN	GPIOELPEN	GPIODLPEN	GPIOCLPEN	GPIOBLPEN	GPIOALPEN
0x54	RCC_AHB2LPENR	Reserved																								OTGFSLPEN	RNGLPEN	HASHLPEN	CRYPLPEN	Reserved			DCMILPEN
0x58	RCC_AHB3LPENR	Reserved																															FSMCLPEN
0x5C	Reserved	Reserved																															
0x60	RCC_APB1LPENR	USART8LPEN	USART7LPEN	DACLPEN	PWRLPEN	Reserved	CAN2LPEN	CAN1LPEN	Reserved	I2C3LPEN	I2C2LPEN	I2C1LPEN	UART5LPEN	UART4LPEN	USART3LPEN	USART2LPEN	Reserved	SPI3LPEN	SPI2LPEN	Reserved		WWDGLPEN	Reserved		TIM14LPEN	TIM13LPEN	TIM12LPEN	TIM7LPEN	TIM6LPEN	TIM5LPEN	TIM4LPEN	TIM3LPEN	TIM2LPEN
0x64	RCC_APB2LPENR	Reserved										SPI6LPEN	SPI5LPEN	Reserved	TIM11LPEN	TIM10LPEN	TIM9LPEN	Reserved	SYSCFGLPEN	SPI4LPEN	SPI1LPEN	SDIOLPEN	ADC3LPEN	ADC2LPEN	ADC1LPEN	Reserved		USART6LPEN	USART1LPEN	Reserved		TIM8LPEN	TIM1LPEN
0x68	Reserved	Reserved																															
0x6C	Reserved	Reserved																															
0x70	RCC_BDCR	Reserved															BDRST	RTCEN	Reserved					RTCSEL 1	RTCSEL 0	Reserved					LSEBYP	LSERDY	LSEON
0x74	RCC_CSR	LPWRRSTF	WWDGRSTF	WDGRSTF	SFTRSTF	PORRSTF	PADRSTF	BORRSTF	RMVF	Reserved																						LSIRDY	LSION
0x78	Reserved	Reserved																															
0x7C	Reserved	Reserved																															
0x80	RCC_SSCGR	SSCGEN	SPREADSEL	Reserved											INCSTEP																			
																							MODPER										
0x84	RCC_PLLI2SCFGR	Reserved	PLLI2SRx						Reserved																	PLLI2SNx					Reserved		
0x88	Reserved	Reserved																															
0x8C	RCC_DCKCFGR	Reserved																															TIMPRE
		Reserved																															

有关寄存器边界地址的信息，请参见*第 52 页的表 2*。

7 通用 I/O (GPIO)

除非特别说明，否则本部分适用于整个 STM32F4xx 系列。

7.1 GPIO 简介

每个通用 I/O 端口包括 4 个 32 位配置寄存器（GPIOx_MODER、GPIOx_OTYPER、GPIOx_OSPEEDR 和 GPIOx_PUPDR）、2 个 32 位数据寄存器（GPIOx_IDR 和 GPIOx_ODR）、1 个 32 位置位/复位寄存器 (GPIOx_BSRR)、1 个 32 位锁定寄存器 (GPIOx_LCKR) 和 2 个 32 位复用功能选择寄存器（GPIOx_AFRH 和 GPIOx_AFRL）。

7.2 GPIO 主要特性

- 受控 I/O 多达 16 个
- 输出状态：推挽或开漏 + 上拉/下拉
- 从输出数据寄存器 (GPIOx_ODR) 或外设（复用功能输出）输出数据
- 可为每个 I/O 选择不同的速度
- 输入状态：浮空、上拉/下拉、模拟
- 将数据输入到输入数据寄存器 (GPIOx_IDR) 或外设（复用功能输入）
- 置位和复位寄存器 (GPIOx_BSRR)，对 GPIOx_ODR 具有按位写权限
- 锁定机制 (GPIOx_LCKR)，可冻结 I/O 配置
- 模拟功能
- 复用功能输入/输出选择寄存器（一个 I/O 最多可具有 16 个复用功能）
- 快速翻转，每次翻转最快只需要两个时钟周期
- 引脚复用非常灵活，允许将 I/O 引脚用作 GPIO 或多种外设功能中的一种

7.3 GPIO 功能描述

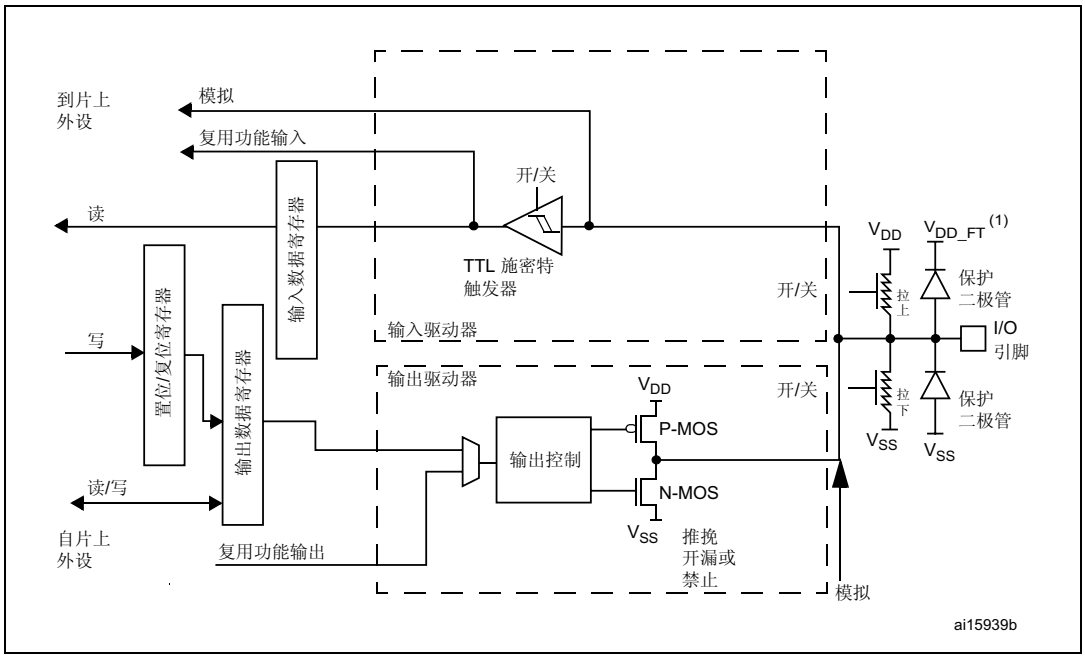
根据数据手册中列出的每个 I/O 端口的特性，可通过软件将通用 I/O (GPIO) 端口的各个端口位分别配置为多种模式：

- 输入浮空
- 输入上拉
- 输入下拉
- 模拟功能
- 具有上拉或下拉功能的开漏输出
- 具有上拉或下拉功能的推挽输出
- 具有上拉或下拉功能的复用功能推挽
- 具有上拉或下拉功能的复用功能开漏

每个 I/O 端口位均可自由编程，但 I/O 端口寄存器必须按 32 位字、半字或字节进行访问。GPIOx_BSRR 寄存器旨在实现对 GPIO ODR 寄存器进行原子读取/修改访问。这样便可确保在读取和修改访问之间发生中断请求也不会有问题。

图 17 显示了 5 V 容忍 I/O 端口位的基本结构。*表 32* 给出了可能的端口位配置方案。

图 17. 5 V 容忍 I/O 端口位的基本结构



1. V_{DD_FT} 是和 5 V 容忍 I/O 相关的电位，与 V_{DD} 不同。

表 28. 端口位配置表(1)

MODER(i) [1:0]	OTYPER(i)	OSPEEDR(i) [B:A]	PUPDR(i) [1:0]		I/O 配置	
01	0	SPEED [B:A]	0	0	GP 输出	PP
	0		0	1	GP 输出	PP + PU
	0		1	0	GP 输出	PP + PD
	0		1	1	保留	
	1		0	0	GP 输出	OD
	1		0	1	GP 输出	OD + PU
	1		1	0	GP 输出	OD + PD
	1		1	1	保留（GP 输出 OD）	
10	0	SPEED [B:A]	0	0	AF	PP
	0		0	1	AF	PP + PU
	0		1	0	AF	PP + PD
	0		1	1	保留	
	1		0	0	AF	OD
	1		0	1	AF	OD + PU
	1		1	0	AF	OD + PD
	1		1	1	保留	

表 28. 端口位配置表[1]（续）

MODER(i) [1:0]	OTYPER(i)	OSPEEDR(i) [B:A]		PUPDR(i) [1:0]		I/O 配置	
00	x	x	x	0	0	输入	浮空
	x	x	x	0	1	输入	PU
	x	x	x	1	0	输入	PD
	x	x	x	1	1	保留（输入浮空）	
11	x	x	x	0	0	输入/输出	模拟
	x	x	x	0	1	保留	
	x	x	x	1	0		
	x	x	x	1	1		

1. GP = 通用、PP = 推挽、PU = 上拉、PD = 下拉、OD = 开漏、AF = 复用功能。

7.3.1 通用 I/O (GPIO)

在复位期间及复位刚刚完成后，复用功能尚未激活，I/O 端口被配置为输入浮空模式。

复位后，调试引脚处于复用功能上拉/下拉状态：

- PA15：JTDI 处于上拉状态
- PA14：JTCK/SWCLK 处于下拉状态
- PA13：JTMS/SWDAT 处于下拉状态
- PB4：NJTRST 处于上拉状态
- PB3：JTDO 处于浮空状态

当引脚配置为输出后，写入到输出数据寄存器 (GPIOx_ODR) 的值将在 I/O 引脚上输出。可以在推挽模式下或开漏模式下使用输出驱动器（输出 0 时仅激活 N-MOS）。

输入数据寄存器 (GPIOx_IDR) 每隔 1 个 AHB1 时钟周期捕获一次 I/O 引脚的数据。

所有 GPIO 引脚都具有内部弱上拉及下拉电阻，可根据 GPIOx_PUPDR 寄存器中的值来打开/关闭。

7.3.2 I/O 引脚复用器和映射

微控制器 I/O 引脚通过一个复用器连接到板载外设/模块，该复用器一次仅允许一个外设的复用功能 (AF) 连接到 I/O 引脚。这可以确保共用同一个 I/O 引脚的外设之间不会发生冲突。

每个 I/O 引脚都有一个复用器，该复用器采用 16 路复用功能输入（AF0 到 AF15），可通过 GPIOx_AFRL（针对引脚 0 到 7）和 GPIOx_AFRH（针对引脚 8 到 15）寄存器对这些输入进行配置：

- 完成复位后，所有 I/O 都会连接到系统的复用功能 0 (AF0)。
- 外设的复用功能映射到 AF1 至 AF13。
- Cortex™-M4F EVENTOUT 映射到 AF15。

下面的*图 18* 对此结构进行了说明。

除了这种灵活的 I/O 复用架构之外，各外设还可以将复用功能映射到不同 I/O 引脚，这可以优化小型封装中可用外设的数量。

要将 I/O 配制成所需功能，请按照以下步骤操作：

1. **系统功能**

   将 I/O 连接到 AF0，然后根据所用功能进行配置：

   – JTAG/SWD：在各器件复位后，会将这些引脚指定为专用引脚，可供片上调试模块立即使用（不受 GPIO 控制器控制）。

   – RTC_REFIN：此引脚应配置为输入浮空模式。

   – MCO1 和 MCO2：这些引脚必须配置为复用功能模式。

注意： *可禁止部分或全部 JTAG/SWD 引脚，以释放相关联的引脚供 GPIO 使用。*

有关详细信息，请参见*第 6.2.10 节：时钟输出功能*。

表 29. 灵活的 SWJ-DP 引脚分配

可用的调试端口	用到的 SWJ I/O 引脚				
	PA13/ JTMS/ SWDIO	PA14/ JTCK/ SWCLK	PA15/ JTDI	PB3/ JTDO	PB4/ NJTRST
全部 SWJ (JTAG-DP + SW-DP) - 复位状态	X	X	X	X	X
全部 SWJ (JTAG-DP + SW-DP)，但不包括 NJTRST	X	X	X	X	
禁止 JTAG-DP 和使能 SW-DP	X	X			
禁止 JTAG-DP 和禁止 SW-DP	已释放				

2. **GPIO**

   在 GPIOx_MODER 寄存器中将所需 I/O 配置为输出或输入。

3. **外设复用功能**

   对于 ADC 和 DAC，在 GPIOx_MODER 寄存器中将所需 I/O 配置为模拟通道。

   对于其它外设：

   – 在 GPIOx_MODER 寄存器中将所需 I/O 配置为复用功能

   – 通过 GPIOx_OTYPER、GPIOx_PUPDR 和 GPIOx_OSPEEDER 寄存器，分别选择类型、上拉/下拉以及输出速度

   – 在 GPIOx_AFRL 或 GPIOx_AFRH 寄存器中，将 I/O 连接到所需 AFx

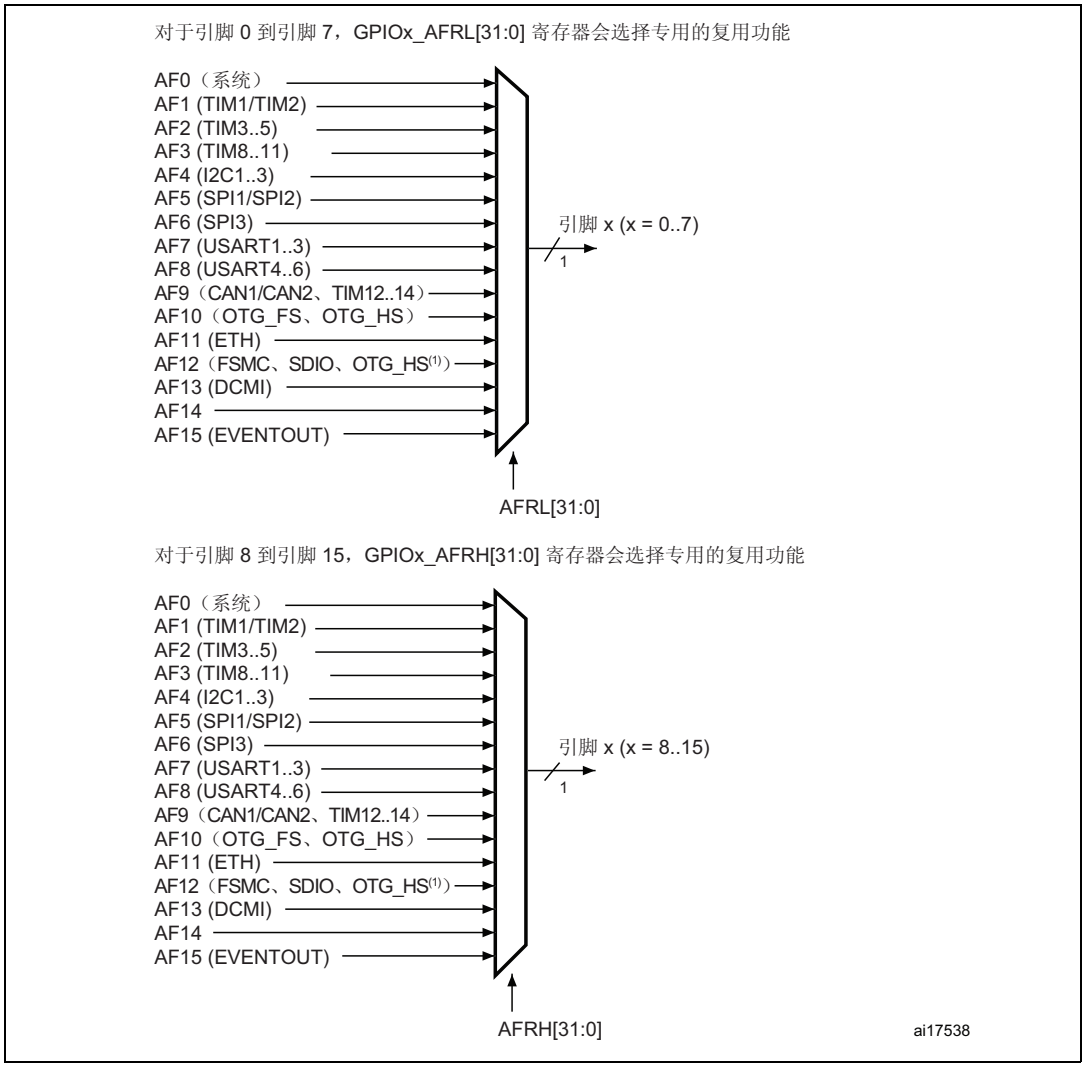
4. **EVENTOUT**

   配置用于输出 Cortex™-M4F EVENTOUT 信号的 I/O 引脚（通过将其连接到 AF15）

注意： *EVENTOUT 不会映射到以下 I/O 引脚：PC13、PC14、PC15、PH0、PH1 和 PI8。*

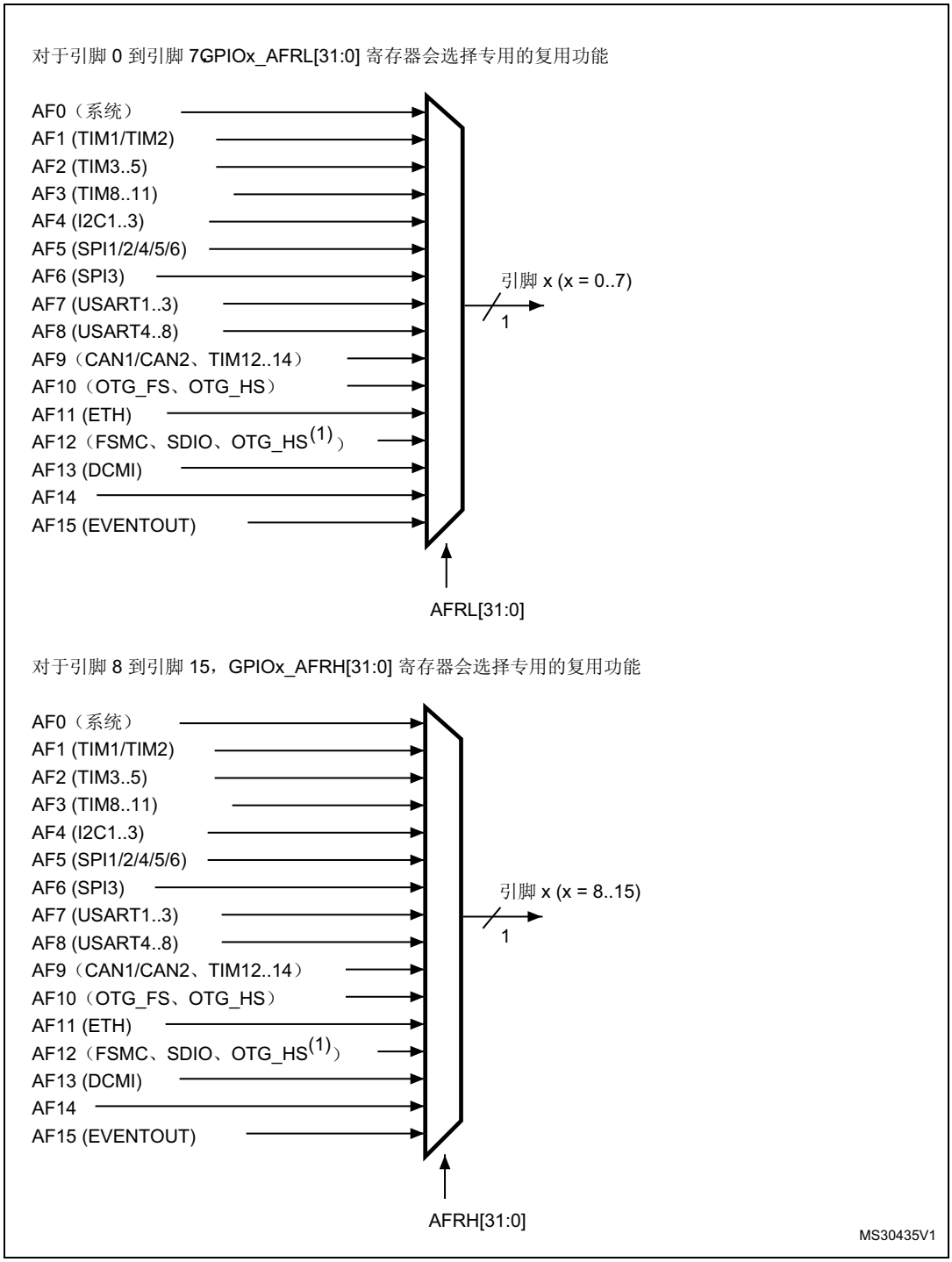
有关系统和外设的复用功能 I/O 引脚映射的详细信息，请参见数据手册中的“复用功能映射”表。

图 18. 在 STM32F405xx/07xx 和 STM32F415xx/17xx 上选择复用功能



1. 在 FS 模式下配置。

图 19.　在 STM32F42xxx 和 STM32F43xxx 上选择复用功能



1. 在 FS 模式下配置。

7.3.3 I/O 端口控制寄存器

每个 GPIO 有 4 个 32 位存储器映射的控制寄存器（GPIOx_MODER、GPIOx_OTYPER、GPIOx_OSPEEDR、GPIOx_PUPDR），可配置多达 16 个 I/O。GPIOx_MODER 寄存器用于选择 I/O 方向（输入、输出、AF、模拟）。GPIOx_OTYPER 和 GPIOx_OSPEEDR 寄存器分别用于选择输出类型（推挽或开漏）和速度 (无论采用哪种 I/O 方向，都会直接将 I/O 速度引脚连接到相应的 GPIOx_OSPEEDR 寄存器位）。无论采用哪种 I/O 方向，GPIOx_PUPDR 寄存器都用于选择上拉/下拉。

7.3.4 I/O 端口数据寄存器

每个 GPIO 都具有 2 个 16 位数据寄存器：输入和输出数据寄存器（GPIOx_IDR 和 GPIOx_ODR）。GPIOx_ODR 用于存储待输出数据，可对其进行读/写访问。通过 I/O 输入的数据存储到输入数据寄存器 (GPIOx_IDR) 中，它是一个只读寄存器。

有关寄存器说明的详细信息，请参见*第 7.4.5 节：GPIO 端口输入数据寄存器 (GPIOx_IDR) (x = A..I)* 和*第 7.4.6 节：GPIO 端口输出数据寄存器 (GPIOx_ODR) (x = A..I)*。

7.3.5 I/O 数据位操作

置位复位寄存器 (GPIOx_BSRR) 是一个 32 位寄存器，它允许应用程序在输出数据寄存器 (GPIOx_ODR) 中对各个单独的数据位执行置位和复位操作。置位复位寄存器的大小是 GPIOx_ODR 的二倍。

GPIOx_ODR 中的每个数据位对应于 GPIOx_BSRR 中的两个控制位：BSRR(i) 和 BSRR(i+SIZE)。当写入 1 时，BSRR(i) 位会**置位**对应的 ODR(i) 位。当写入 1 时，BSRR(i+SIZE) 位会**清零** ODR(i) 对应的位。

在 GPIOx_BSRR 中向任何位写入 0 都不会对 GPIOx_ODR 中的对应位产生任何影响。如果在 GPIOx_BSRR 中同时尝试对某个位执行置位和清零操作，则置位操作优先。

使用 GPIOx_BSRR 寄存器更改 GPIOx_ODR 中各个位的值是一个“单次”操作，不会锁定 GPIOx_ODR 位。随时都可以直接访问 GPIOx_ODR 位。GPIOx_BSRR 寄存器提供了一种执行原子按位处理的方法。

在对 GPIOx_ODR 进行位操作时，软件无需禁止中断：在一次原子 AHB1 写访问中，可以修改一个或多个位。

7.3.6 GPIO 锁定机制

通过将特定的写序列应用到 GPIOx_LCKR 寄存器，可以冻结 GPIO 控制寄存器。冻结的寄存器包括 GPIOx_MODER、GPIOx_OTYPER、GPIOx_OSPEEDR、GPIOx_PUPDR、GPIOx_AFRL 和 GPIOx_AFRH。

要对 GPIOx_LCKR 寄存器执行写操作，必须应用特定的写/读序列。当正确的 LOCK 序列应用到此寄存器的第 16 位后，会使用 LCKR[15:0] 的值来锁定 I/O 的配置（在写序列期间，LCKR[15:0] 的值必须相同）。将 LOCK 序列应用到某个端口位后，在执行下一次复位之前，将无法对该端口位的值进行修改。每个 GPIOx_LCKR 位都会冻结控制寄存器（GPIOx_MODER、GPIOx_OTYPER、GPIOx_OSPEEDR、GPIOx_PUPDR、GPIOx_AFRL 和 GPIOx_AFRH）中的对应位。

LOCK 序列（参见*第 7.4.8 节：GPIO 端口配置锁定寄存器 (GPIOx_LCKR) (x = A..I)*）只能通过对 GPIOx_LCKR 寄存器进行字（32 位长）访问的方式来执行，因为 GPIOx_LCKR 的第 16 位必须与 [15:0] 位同时置位。

有关详细信息，请参见*第 7.4.8 节：GPIO 端口配置锁定寄存器 (GPIOx_LCKR) (x = A..I)* 中的 LCKR 寄存器说明。

7.3.7 I/O 复用功能输入/输出

有两个寄存器可用来从每个 I/O 可用的 16 个复用功能输入/输出中进行选择。借助这些寄存器，可根据应用程序的要求将某个复用功能连接到其它某个引脚。
这意味着可使用 GPIOx_AFRL 和 GPIOx_AFRH 复用功能寄存器在每个 GPIO 上复用多个可用的外设功能。这样一来，应用程序可为每个 I/O 选择任何一个可用功能。由于 AF 选择信号由复用功能输入和复用功能输出共用，所以只需为每个 I/O 的复用功能输入/输出选择一个通道即可。

要了解在每个 GPIO 引脚上复用了哪些功能，请参见数据手册。

注意： *对于每个 I/O 而言，应用程序一次只能为其选择一个可用的外设功能。*

7.3.8 外部中断线/唤醒线

所有端口都具有外部中断功能。要使用外部中断线，必须将端口配置为输入模式，请参见*第 10.2 节：外部中断/事件控制器 (EXTI)*和*第 10.2.3 节：唤醒事件管理*。

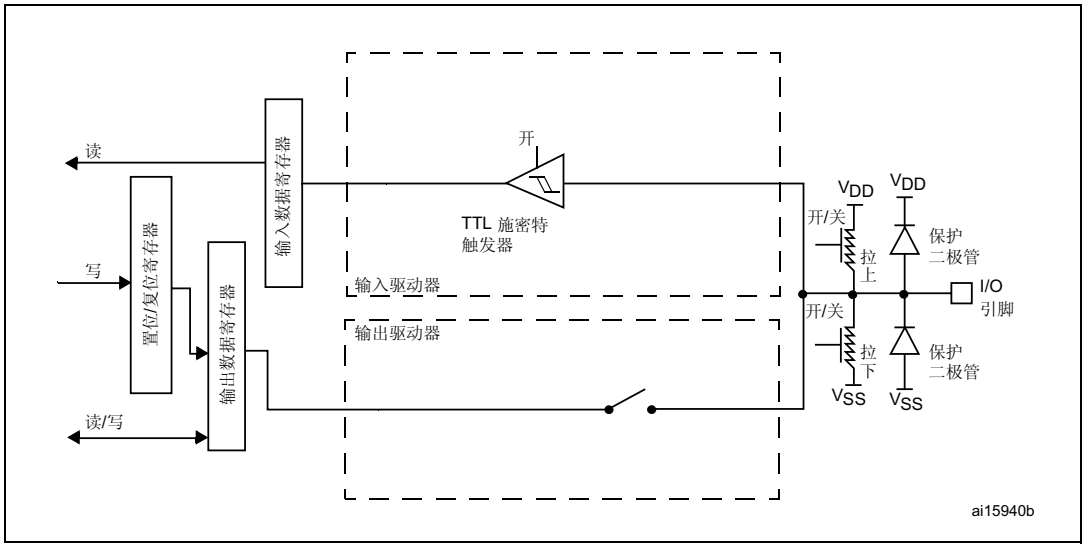
7.3.9 输入配置

对 I/O 端口进行编程作为输入时：

- 输出缓冲器被关闭
- 施密特触发器输入被打开
- 根据 GPIOx_PUPDR 寄存器中的值决定是否打开上拉和下拉电阻
- 输入数据寄存器每隔 1 个 AHB1 时钟周期对 I/O 引脚上的数据进行一次采样
- 对输入数据寄存器的读访问可获取 I/O 状态

图 20 说明了 I/O 端口位的输入配置。

图 20. 输入浮空/上拉/下拉配置



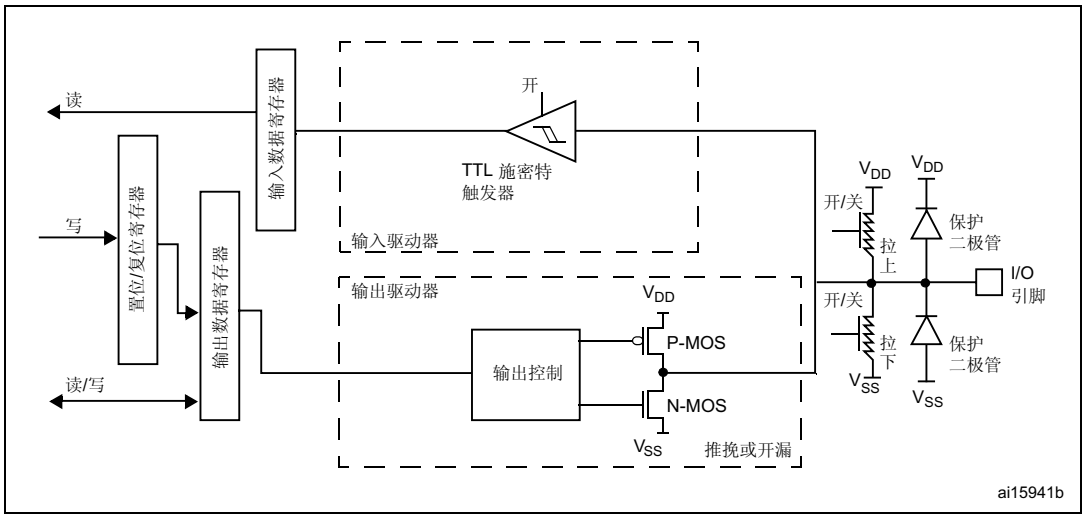
7.3.10 输出配置

对 I/O 端口进行编程作为输出时：

- 输出缓冲器被打开：
  - 开漏模式：输出寄存器中的“0”可激活 N-MOS，而输出寄存器中的“1”会使端口保持高组态 (Hi-Z)（P-MOS 始终不激活）。
  - 推挽模式：输出寄存器中的“0”可激活 N-MOS，而输出寄存器中的“1”可激活 P-MOS。
- 施密特触发器输入被打开
- 根据 GPIOx_PUPDR 寄存器中的值决定是否打开弱上拉电阻和下拉电阻
- 输入数据寄存器每隔 1 个 AHB1 时钟周期对 I/O 引脚上的数据进行一次采样
- 对输入数据寄存器的读访问可获取 I/O 状态
- 对输出数据寄存器的读访问可获取最后的写入值

图 21 说明了 I/O 端口位的输出配置。

图 21. 输出配置



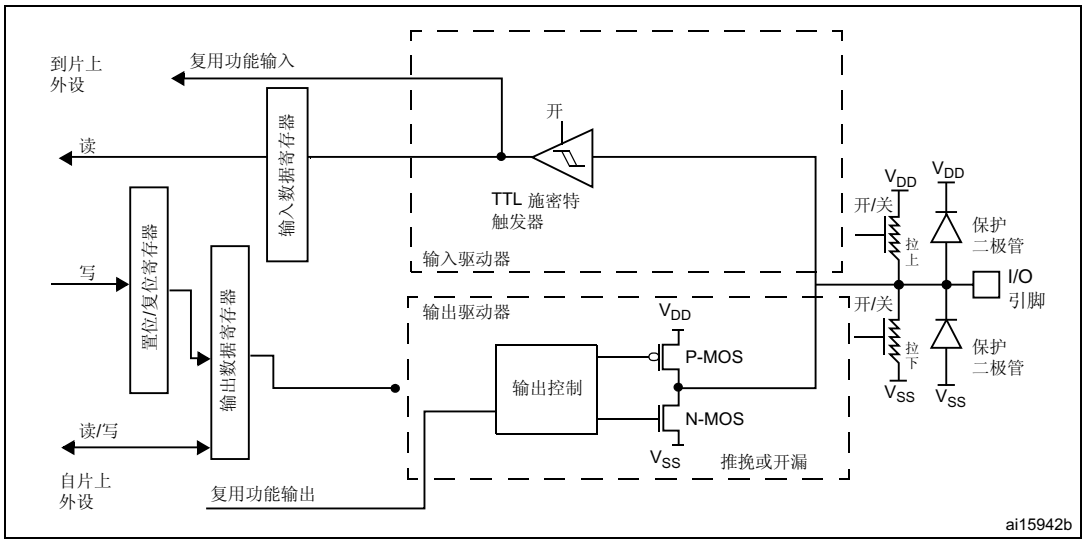
7.3.11 复用功能配置

对 I/O 端口进行编程作为复用功能时：

- 可将输出缓冲器配置为开漏或推挽
- 输出缓冲器由来自外设的信号驱动（发送器使能和数据）
- 施密特触发器输入被打开
- 根据 GPIOx_PUPDR 寄存器中的值决定是否打开弱上拉电阻和下拉电阻
- 输入数据寄存器每隔 1 个 AHB1 时钟周期对 I/O 引脚上的数据进行一次采样
- 对输入数据寄存器的读访问可获取 I/O 状态

图 22 说明了 I/O 端口位的复用功能配置。

图 22. 复用功能配置



7.3.12 模拟配置

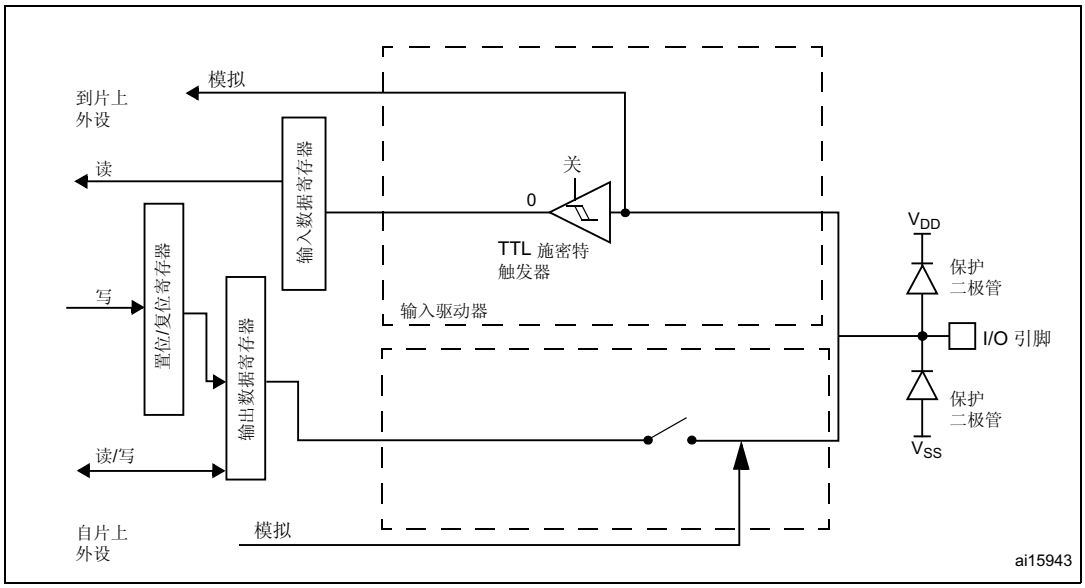
对 I/O 端口进行编程作为模拟配置时：

- 输出缓冲器被禁止。
- 施密特触发器输入停用，I/O 引脚的每个模拟输入的功耗变为零。施密特触发器的输出被强制处理为恒定值 (0)。
- 弱上拉和下拉电阻被关闭。
- 对输入数据寄存器的读访问值为“0”。

注意： *在模拟配置中，I/O 引脚不能为 5 V 容忍。*

图 23 说明了 I/O 端口位的高阻态模拟输入配置。

图 23. 高组态模拟配置



7.3.13 将 OSC32_IN/OSC32_OUT 引脚用作 GPIO PC14/PC15 端口引脚

当 LSE 振荡器处于关闭状态时，可分别将 LSE 振荡器引脚 OSC32_IN 和 OSC32_OUT 用作通用 PC14 I/O 和 PC15 I/O。当 LSE 振荡器处于开启状态时，PC14 I/O 和 PC15 I/O 只能被配置为 LSE 振荡器引脚 OSC32_IN 和 OSC32_OUT。可通过在 RCC_BDCR 寄存器中将 LSEON 位置 1 来完成此操作。LSE 的优先级高于 GPIO 功能。

注意： *当 1.2 V 域掉电时（因器件进入待机模式），或者当备份域由 V_{BAT} 供电时（V_{DD} 不再供电），PC14/PC15 GPIO 功能将丢失。在此情况下，I/O 被设置为模拟输入模式。*

7.3.14 将 OSC_IN/OSC_OUT 引脚用作 GPIO PH0/PH1 端口引脚

当 HSE 振荡器处于关闭状态时，可分别将 HSE 振荡器引脚 OSC_IN 和 OSC_OUT 用作通用 PH0 I/O 和 PH1 I/O。（完成复位后，HSE 振荡器处于关闭状态）。当 HSE 振荡器处于开启状态时，PH0/PH1 I/O 只能被配置为 OSC_IN/OSC_OUT HSE 振荡器引脚。可通过在 RCC_CR 寄存器中将 HSEON 位置 1 来完成此操作。HSE 的优先级高于 GPIO 功能。

7.3.15 选择 RTC_AF1 和 RTC_AF2 复用功能

STM32F4xx 具有两个 GPIO 引脚：RTC_AF1 和 RTC_AF2，可使用它们来检测入侵或时间戳事件、RTC_ALARM 或 RTC_CALIB RTC 输出。

- RTC_AF1 (PC13) 可用于以下目的：

RTC_ALARM 输出：此输出可以是 RTC 闹钟 A、RTC 闹钟 B 或 RTC 唤醒，具体取决于 RTC_CR 寄存器中的 OSEL[1:0] 位

- RTC_CALIB 输出：可通过在 RTC_CR 寄存器中将 COE[23] 置 1 来使能此功能
- RTC_TAMP1：入侵事件检测
- RTC_TS：时间戳事件检测

RTC_AF2 (PI8) 可用于以下目的：

- RTC_TAMP1：入侵事件检测
- RTC_TAMP2：入侵事件检测
- RTC_TS：时间戳事件检测

可如下所示通过 RTC_TAFCR 寄存器来选择相应的引脚：

- TAMP1INSEL 用于选择要用作 RTC_TAMP1 入侵输入的引脚
- TSINSEL 用于选择要用作 RTC_TS 时间戳输入的引脚
- ALARMOUTTYPE 用于选择是以推挽模式还是开漏模式输出 RTC_ALARM

输出机制遵循*表 30* 和*表 31* 中所列的优先级顺序。

表 30. RTC_AF1 引脚(1)

引脚配置和函数	RTC_ALARM 使能	RTC_CALIB 使能	入侵使能	时间戳使能	TAMP1INSEL TAMPER1 引脚选择	TSINSEL TIMESTAMP 引脚选择	ALARMOUTTYPE RTC_ALARM 配置
闹钟输出 输出 OD	1	无关	无关	无关	无关	无关	0
闹钟输出 输出 PP	1	无关	无关	无关	无关	无关	1
校准输出 输出 PP	0	1	无关	无关	无关	无关	无关
TAMPER1 输入浮空	0	0	1	0	0	无关	无关
TIMESTAMP 和 TAMPER1 输入浮空	0	0	1	1	0	0	无关
TIMESTAMP 输入浮空	0	0	0	1	无关	0	无关
标准 GPIO	0	0	0	0	无关	无关	无关

1. OD：开漏；PP：推挽。

表 31. RTC_AF2 引脚

引脚配置和函数	入侵使能	时间戳使能	TAMP1INSEL TAMPER1 引脚选择	TSINSEL TIMESTAMP 引脚选择	ALARMOUTTYPE RTC_ALARM 配置
TAMPER1 输入浮空	1	0	1	无关	无关
TIMESTAMP 和 TAMPER1 输入浮空	1	1	1	1	无关
TIMESTAMP 输入浮空	0	1	无关	1	无关
标准 GPIO	0	0	无关	无关	无关

7.4 GPIO 寄存器

本节对 GPIO 寄存器进行了详细介绍。
有关寄存器位、寄存器偏移地址和复位值的汇总，请参见*表 32*。

可通过字节（8 位）、半字（16 位）或字（32 位）对 GPIO 寄存器进行访问。

7.4.1 GPIO 端口模式寄存器 (GPIOx_MODER) (x = A..I)

GPIO port mode register

偏移地址：0x00

复位值：

- 0xA800 0000（端口 A）
- 0x0000 0280（端口 B）
- 0x0000 0000（其它端口）

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MODER15[1:0]		MODER14[1:0]		MODER13[1:0]		MODER12[1:0]		MODER11[1:0]		MODER10[1:0]		MODER9[1:0]		MODER8[1:0]	
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
MODER7[1:0]		MODER6[1:0]		MODER5[1:0]		MODER4[1:0]		MODER3[1:0]		MODER2[1:0]		MODER1[1:0]		MODER0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 2y:2y+1 **MODERy[1:0]**：端口 x 配置位 (Port x configuration bits) (y = 0..15)
这些位通过软件写入，用于配置 I/O 方向模式。
00：输入（复位状态）
01：通用输出模式
10：复用功能模式
11：模拟模式

7.4.2 GPIO 端口输出类型寄存器 (GPIOx_OTYPER) (x = A..I)

GPIO port output type register

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
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OT15	OT14	OT13	OT12	OT11	OT10	OT9	OT8	OT7	OT6	OT5	OT4	OT3	OT2	OT1	OT0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:16 保留，必须保持复位值。

位 15:0 **OTy[1:0]**：端口 x 配置位 (Port x configuration bits) (y = 0..15)
这些位通过软件写入，用于配置 I/O 端口的输出类型。
0：输出推挽（复位状态）
1：输出开漏

7.4.3 GPIO 端口输出速度寄存器 (GPIOx_OSPEEDR) (x = A..I/)

GPIO port output speed register

偏移地址：0x08

复位值：

- 0x0000 00C0（端口 B）
- 0x0000 0000（其它端口）

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OSPEEDR15[1:0]		OSPEEDR14[1:0]		OSPEEDR13[1:0]		OSPEEDR12[1:0]		OSPEEDR11[1:0]		OSPEEDR10[1:0]		OSPEEDR9[1:0]		OSPEEDR8[1:0]	
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
OSPEEDR7[1:0]		OSPEEDR6[1:0]		OSPEEDR5[1:0]		OSPEEDR4[1:0]		OSPEEDR3[1:0]		OSPEEDR2[1:0]		OSPEEDR1[1:0]		OSPEEDR0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 2y:2y+1 **OSPEEDRy[1:0]**：端口 x 配置位 (Port x configuration bits) (y = 0..15)
这些位通过软件写入，用于配置 I/O 输出速度。
00：2 MHz（低速）
01：25 MHz（中速）
10：50 MHz（快速）
11：30 pF 时为 100 MHz（高速）（15 pF 时为 80 MHz 输出（最大速度））

7.4.4 GPIO 端口上拉/下拉寄存器 (GPIOx_PUPDR) (x = A..I/)

GPIO port pull-up/pull-down register

偏移地址：0x0C

复位值：

- 0x6400 0000（端口 A）
- 0x0000 0100（端口 B）
- 0x0000 0000（其它端口）

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PUPDR15[1:0]		PUPDR14[1:0]		PUPDR13[1:0]		PUPDR12[1:0]		PUPDR11[1:0]		PUPDR10[1:0]		PUPDR9[1:0]		PUPDR8[1:0]	
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
PUPDR7[1:0]		PUPDR6[1:0]		PUPDR5[1:0]		PUPDR4[1:0]		PUPDR3[1:0]		PUPDR2[1:0]		PUPDR1[1:0]		PUPDR0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 2y:2y+1 **PUPDRy[1:0]**：端口 x 配置位 (Port x configuration bits) (y = 0..15)
这些位通过软件写入，用于配置 I/O 上拉或下拉。
00：无上拉或下拉
01：上拉
10：下拉
11：保留

7.4.5 GPIO 端口输入数据寄存器 (GPIOx_IDR) (x = A..I)

GPIO port input data register

偏移地址：0x10

复位值：0x0000 XXXX（其中 X 表示未定义）

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
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 保留，必须保持复位值。

位 15:0 **IDRy[15:0]**：端口输入数据 (Port input data) (y = 0..15)
这些位为只读形式，只能在字模式下访问。它们包含相应 I/O 端口的输入值。

7.4.6 GPIO 端口输出数据寄存器 (GPIOx_ODR) (x = A..I)

GPIO port output data register

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
Reserved															
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 保留，必须保持复位值。

位 15:0 **ODRy[15:0]**：端口输出数据 (Port output data) (y = 0..15)
这些位可通过软件读取和写入。
注意：对于原子置位/复位，通过写入 GPIOx_BSRR 寄存器，可分别对 ODR 位进行置位和复位 (x = A..I)。

7.4.7 GPIO 端口置位/复位寄存器 (GPIOx_BSRR) (x = A..I)

GPIO port bit set/reset register

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
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 31:16 **BRy**：端口 x 复位位 y (Port x reset bit y) (y = 0..15)

这些位为只写形式，只能在字、半字或字节模式下访问。读取这些位可返回值 0x0000。

0：不会对相应的 ODRx 位执行任何操作

1：对相应的 ODRx 位进行复位

注意：如果同时对 BSx 和 BRx 置位，则 BSx 的优先级更高。

位 15:0 **BSy**：端口 x 置位位 y (Port x set bit y) (y= 0..15)

这些位为只写形式，只能在字、半字或字节模式下访问。读取这些位可返回值 0x0000。

0：不会对相应的 ODRx 位执行任何操作

1：对相应的 ODRx 位进行置位

7.4.8 GPIO 端口配置锁定寄存器 (GPIOx_LCKR) (x = A..I)

GPIO port configuration lock register

当正确的写序列应用到第 16 位 (LCKK) 时，此寄存器将用于锁定端口位的配置。位 [15:0] 的值用于锁定 GPIO 的配置。在写序列期间，不能更改 LCKR[15:0] 的值。将 LOCK 序列应用到某个端口位后，在执行下一次复位之前，将无法对该端口位的值进行修改。

注意：可使用特定的写序列对 GPIOx_LCKR 寄存器执行写操作。在此写序列期间只允许使用字访问（32 位长）。

每个锁定位冻结一个特定的配置寄存器（控制寄存器和复用功能寄存器）。

偏移地址：0x1C

复位值：0x0000 0000

访问：仅 32 位字，读/写寄存器

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															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 保留，必须保持复位值。

位 16 **LCKK[16]**：锁定键 (Lock key)

可随时读取此位。可使用锁定键写序列对其进行修改。

0：端口配置锁定键未激活。

1：端口配置锁定键已激活。直到 MCU 复位时，才锁定 GPIOx_LCKR 寄存器。

锁定键写序列：

WR LCKR[16] = '1' + LCKR[15:0]

WR LCKR[16] = '0' + LCKR[15:0]

WR LCKR[16] = '1' + LCKR[15:0]

RD LCKR

RD LCKR[16] = '1'（此读操作为可选操作，但它可确认锁定已激活）

注意：在锁定键写序列期间，不能更改 LCK[15:0] 的值。

锁定序列中的任何错误都将中止锁定操作。

在任一端口位上的第一个锁定序列之后，对 LCKK 位的任何读访问都将返回“1”，直到下一次 CPU 复位为止。

位 15:0 **LCKy**：端口 x 锁定位 y (Port x lock bit y) (y= 0..15)

这些位都是读/写位，但只能在 LCKK 位等于“0”时执行写操作。

0：端口配置未锁定

1：端口配置已锁定

7.4.9 GPIO 复用功能低位寄存器 (GPIOx_AFRL) (x = A..I)

GPIO alternate function low register

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
AFRL7[3:0]				AFRL6[3:0]				AFRL5[3:0]				AFRL4[3:0]			
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
AFRL3[3:0]				AFRL2[3:0]				AFRL1[3:0]				AFRL0[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:0 **AFRLy**：端口 x 位 y 的复用功能选择 (Alternate function selection for port x bit y) (y = 0..7)

这些位通过软件写入，用于配置复用功能 I/O。

AFRLy 选择：

0000：AF0
0001：AF1
0010：AF2
0011：AF3
0100：AF4
0101：AF5
0110：AF6
0111：AF7
1000：AF8
1001：AF9
1010：AF10
1011：AF11
1100：AF12
1101：AF13
1110：AF14
1111：AF15

7.4.10 GPIO 复用功能高位寄存器 (GPIOx_AFRH) (x = A..I)

GPIO alternate function high register

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
AFRH15[3:0]				AFRH14[3:0]				AFRH13[3:0]				AFRH12[3:0]			
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
AFRH11[3:0]				AFRH10[3:0]				AFRH9[3:0]				AFRH8[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:0 **AFRHy**：端口 x 位 y 的复用功能选择 (Alternate function selection for port x bit y) (y = 8.0.15)
这些位通过软件写入，用于配置复用功能 I/O。

AFRHy 选择：

0000：AF0
0001：AF1
0010：AF2
0011：AF3
0100：AF4
0101：AF5
0110：AF6
0111：AF7
1000：AF8
1001：AF9
1010：AF10
1011：AF11
1100：AF12
1101：AF13
1110：AF14
1111：AF15

7.4.11 GPIO 寄存器映射

下表列出了 GPIO 寄存器映射和复位值。

表 32. GPIO 寄存器映射和复位值

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	GPIOA_MODER	MODER15[1:0]		MODER14[1:0]		MODER13[1:0]		MODER12[1:0]		MODER11[1:0]		MODER10[1:0]		MODER9[1:0]		MODER8[1:0]		MODER7[1:0]		MODER6[1:0]		MODER5[1:0]		MODER4[1:0]		MODER3[1:0]		MODER2[1:0]		MODER1[1:0]		MODER0[1:0]	
	Reset value	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00	GPIOB_MODER	MODER15[1:0]		MODER14[1:0]		MODER13[1:0]		MODER12[1:0]		MODER11[1:0]		MODER10[1:0]		MODER9[1:0]		MODER8[1:0]		MODER7[1:0]		MODER6[1:0]		MODER5[1:0]		MODER4[1:0]		MODER3[1:0]		MODER2[1:0]		MODER1[1:0]		MODER0[1:0]	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0
0x00	GPIOx_MODER (where x = C..I/)	MODER15[1:0]		MODER14[1:0]		MODER13[1:0]		MODER12[1:0]		MODER11[1:0]		MODER10[1:0]		MODER9[1:0]		MODER8[1:0]		MODER7[1:0]		MODER6[1:0]		MODER5[1:0]		MODER4[1:0]		MODER3[1:0]		MODER2[1:0]		MODER1[1:0]		MODER0[1:0]	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

表 32. GPIO 寄存器映射和复位值（续）

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x04	GPIOx_OTYPER (where x = A..I/)	Reserved																OT15	OT14	OT13	OT12	OT11	OT10	OT9	OT8	OT7	OT6	OT5	OT4	OT3	OT2	OT1	OT0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08	GPIOx_OSPEEDER (where x = A..**I/** except B)	OSPEEDR15[1:0]		OSPEEDR14[1:0]		OSPEEDR13[1:0]		OSPEEDR12[1:0]		OSPEEDR11[1:0]		OSPEEDR10[1:0]		OSPEEDR9[1:0]		OSPEEDR8[1:0]		OSPEEDR7[1:0]		OSPEEDR6[1:0]		OSPEEDR5[1:0]		OSPEEDR4[1:0]		OSPEEDR3[1:0]		OSPEEDR2[1:0]		OSPEEDR1[1:0]		OSPEEDR0[1:0]	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08	GPIOB_OSPEEDER	OSPEEDR15[1:0]		OSPEEDR14[1:0]		OSPEEDR13[1:0]		OSPEEDR12[1:0]		OSPEEDR11[1:0]		OSPEEDR10[1:0]		OSPEEDR9[1:0]		OSPEEDR8[1:0]		OSPEEDR7[1:0]		OSPEEDR6[1:0]		OSPEEDR5[1:0]		OSPEEDR4[1:0]		OSPEEDR3[1:0]		OSPEEDR2[1:0]		OSPEEDR1[1:0]		OSPEEDR0[1:0]	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0
0x0C	GPIOA_PUPDR	PUPDR15[1:0]		PUPDR14[1:0]		PUPDR13[1:0]		PUPDR12[1:0]		PUPDR11[1:0]		PUPDR10[1:0]		PUPDR9[1:0]		PUPDR8[1:0]		PUPDR7[1:0]		PUPDR6[1:0]		PUPDR5[1:0]		PUPDR4[1:0]		PUPDR3[1:0]		PUPDR2[1:0]		PUPDR1[1:0]		PUPDR0[1:0]	
	Reset value	0	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	GPIOB_PUPDR	PUPDR15[1:0]		PUPDR14[1:0]		PUPDR13[1:0]		PUPDR12[1:0]		PUPDR11[1:0]		PUPDR10[1:0]		PUPDR9[1:0]		PUPDR8[1:0]		PUPDR7[1:0]		PUPDR6[1:0]		PUPDR5[1:0]		PUPDR4[1:0]		PUPDR3[1:0]		PUPDR2[1:0]		PUPDR1[1:0]		PUPDR0[1:0]	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
0x0C	GPIOx_PUPDR (where x = C..**I/**)	PUPDR15[1:0]		PUPDR14[1:0]		PUPDR13[1:0]		PUPDR12[1:0]		PUPDR11[1:0]		PUPDR10[1:0]		PUPDR9[1:0]		PUPDR8[1:0]		PUPDR7[1:0]		PUPDR6[1:0]		PUPDR5[1:0]		PUPDR4[1:0]		PUPDR3[1:0]		PUPDR2[1:0]		PUPDR1[1:0]		PUPDR0[1:0]	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	GPIOx_IDR (where x = A..**I/**)	Reserved																IDR15	IDR14	IDR13	IDR12	IDR11	IDR10	IDR9	IDR8	IDR7	IDR6	IDR5	IDR4	IDR3	IDR2	IDR1	IDR0
	Reset value																	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0x14	GPIOx_ODR (where x = A..**I/**)	Reserved																ODR15	ODR14	ODR13	ODR12	ODR11	ODR10	ODR9	ODR8	ODR7	ODR6	ODR5	ODR4	ODR3	ODR2	ODR1	ODR0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	GPIOx_BSRR (where x = A..**I/**)	BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0	BS15	BS14	BS13	BS12	BS11	BS10	BS9	BS8	BS7	BS6	BS5	BS4	BS3	BS2	BS1	BS0
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1C	GPIOx_LCKR (where x = A..**I/**)	Reserved															LCKK	LCK15	LCK14	LCK13	LCK12	LCK11	LCK10	LCK9	LCK8	LCK7	LCK6	LCK5	LCK4	LCK3	LCK2	LCK1	LCK0
	Reset value																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20	GPIOx_AFRL (where x = A..**I/**)	AFRL7[3:0]				AFRL6[3:0]				AFRL5[3:0]				AFRL4[3:0]				AFRL3[3:0]				AFRL2[3:0]				AFRL1[3:0]				AFRL0[3:0]	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x24	GPIOx_AFRH (where x = A..**I/**)	AFRH15[3:0]				AFRH14[3:0]				AFRH13[3:0]				AFRH12[3:0]				AFRH11[3:0]				AFRH10[3:0]				AFRH9[3:0]				AFRH8[3:0]	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

有关寄存器边界地址的信息，请参见*第 52 页的表 2*。

8 系统配置控制器 (SYSCFG)

系统配置控制器主要用于管理对可执行代码的存储区域的地址重映射、选择以太网 PHY 接口以及管理 GPIO 的外部中断线连接。

除非特别说明，否则本部分适用于整个 STM32F4xx 系列。

8.1 I/O 补偿单元

默认情况下不使用 I/O 补偿单元。但是，当以 50 MHz 或 100 MHz 模式配置 I/O 输出缓冲区速度时，建议使用补偿单元对 I/O $t_{f(IO)out}$/$t_{r(IO)out}$ 进行斜率控制，从而降低 I/O 端口噪声对电源的影响。

补偿单元使能后，会设置一个“就绪”标志，指示补偿单元已就绪，可供使用。只有电源电压范围为 2.4 到 3.6 V 时，才可以使用 I/O 补偿单元。

8.2 SYSCFG 寄存器

8.2.1 SYSCFG 存储器重映射寄存器 (SYSCFG_MEMRMP)

SYSCFG memory remap register

此寄存器用于对存储器重映射进行配置：

- 使用两个位来配置可在地址 0x0000 0000 访问的存储器区域。从而通过软件选择物理重映射，而旁路 BOOT 引脚。
- 这两个位的复位值和复位时 BOOT 引脚的设置相同。当 BOOT 引脚设为 10 [(BOOT1,BOOT0) = (1,0)] 从主 Flash 中自举时，寄存器值为 0x00。

当把 FSMC 重映射到地址 0x0000 0000 时，只有 FSMC 的 Bank1 的前两个区域（NOR/PSRAM 1 和 NOR/PSRAM 2）可被重映射到低端地址。在重映射模式下，CPU 可以通过 ICode 总线（而不是 System 总线）访问外部存储器来提高性能。

偏移地址：0x00

复位值：0x0000 000X（X 和 BOOT 引脚的设置相同）

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

位 31:2 保留，必须保持复位值。

位 1:0 **MEM_MODE**: 存储器映射选择 (Memory mapping selection)

由软件置 1 和清零。此位控制哪块存储区域被映射到地址 0x0000 0000。复位后，这些位将采用 Boot 引脚所选择的值（FSMC 除外）。

00：把主 Flash 映射到地址 0x0000 0000
01：把系统 Flash 映射到地址 0x0000 0000
10：把 FSMC Bank1（NOR/PSRAM 1 和 2）映射到地址 0x0000 0000
11：把 SRAM (112kB) 映射到地址 0x0000 0000

8.2.2 用于 STM32F405xx/07xx 和 STM32F415xx/17xx 的 SYSCFG 外设模式配置寄存器 (SYSCFG_PMC)

SYSCFG peripheral mode configuration register

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

位 31:24 保留，必须保持复位值。

位 23 **MII_RMII_SEL**：以太网 PHY 接口选择 (Ethernet PHY interface selection)
由软件置 1 和清零。这些位控制以太网 MAC 的 PHY 接口。
0：选择 MII 接口
1：选择 RMII Why 接口
注意：必须在 MAC 处于复位状态且在使能 MAC 时钟之前完成此配置。

位 22:0 保留，必须保持复位值。

8.2.3 用于 STM32F42xxx 和 STM32F43xxx 的 SYSCFG 外设模式配置寄存器 (SYSCFG_PMC)

SYSCFG peripheral mode configuration register

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
Reserved								MII_RMII_SEL	Reserved				ADC3DC2	ADC2DC2	ADC1DC2
								rw					rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															

位 31:24 保留，必须保持复位值。

位 23 **MII_RMII_SEL**：以太网 PHY 接口选择 (Ethernet PHY interface selection)
由软件置 1 和清零。这些位控制以太网 MAC 的 PHY 接口。
0：选择 MII 接口
1：选择 RMII Why 接口
注意：必须在 MAC 处于复位状态且在使能 MAC 时钟之前完成此配置。

位 22:19 保留，必须保持复位值。

位 18:16 **ADCxDC2**：

0：无操作。

1：请参见 AN4073 了解如何使用此位。

注意：只有满足以下条件时才能将这些位置 1：

- ADC 时钟大于或等于 30 MHz。

- 如果多个 ADC 的转换启动时间不同且采样时间不同，则必须仅选择一个 ADCxDC2 位。

- 当 PWR_CR 寄存器中的 ADCDC1 位置 1 时，这些位不能置 1。

位 15:0 保留，必须保持复位值。

8.2.4 SYSCFG 外部中断配置寄存器 1 (SYSCFG_EXTICR1)

SYSCFG external interrupt configuration register 1

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
Reserved															
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	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:16 保留，必须保持复位值。

位 15:0 **EXTIx[3:0]**：EXTI x 配置（x = 0 到 3）(EXTI x configuration (x = 0 to 3))

这些位通过软件写入，以选择 EXTIx 外部中断的源输入。

0000：PA[x] 引脚
0001：PB[x] 引脚
0010：PC[x] 引脚
0011：PD[x] 引脚
0100：PE[x] 引脚
0101：PF[C] 引脚
0110：PG[x] 引脚
0111：PH[x] 引脚
1000：PI[x] 引脚

8.2.5 SYSCFG 外部中断配置寄存器 2 (SYSCFG_EXTICR2)

SYSCFG external interrupt configuration register 2

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
Reserved															
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	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:16 保留，必须保持复位值。

位 15:0 **EXTIx[3:0]**：EXTI x 配置（x = 4 到 7）(EXTI x configuration (x = 4 to 7))
这些位通过软件写入，以选择 EXTIx 外部中断的源输入。
0000：PA[x] 引脚
0001：PB[x] 引脚
0010：PC[x] 引脚
0011：PD[x] 引脚
0100：PE[x] 引脚
0101：PF[x] 引脚
0110：PG[x] 引脚
0111：PH[x] 引脚
1000：PI[x] 引脚

8.2.6 SYSCFG 外部中断配置寄存器 3 (SYSCFG_EXTICR3)

SYSCFG external interrupt configuration register 3

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
Reserved															
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	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:16 保留，必须保持复位值。

位 15:0 **EXTIx[3:0]**：EXTI x 配置（x = 8 到 11）(EXTI x configuration (x = 8 to 11))
这些位通过软件写入，以选择 EXTIx 外部中断的源输入。
0000：PA[x] 引脚
0001：PB[x] 引脚
0010：PC[x] 引脚
0011：PD[x] 引脚
0100：PE[x] 引脚
0101：PF[x] 引脚
0110：PG[x] 引脚
0111：PH[x] 引脚
1000：PI[x] 引脚

8.2.7 SYSCFG 外部中断配置寄存器 4 (SYSCFG_EXTICR4)

SYSCFG external interrupt configuration register 4

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
Reserved															
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	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:16 保留，必须保持复位值。

位 15:0 **EXTIx[3:0]**：EXTI x 配置（x = 12 到 15）(EXTI x configuration (x = 12 to 15))
这些位通过软件写入，以选择 EXTIx 外部中断的源输入。
0000：PA[x] 引脚
0001：PB[x] 引脚
0010：PC[x] 引脚
0011：PD[x] 引脚
0100：PE[x] 引脚
0101：PF[x] 引脚
0110：PG[x] 引脚
0111：PH[x] 引脚
注意：PI[15:12] 未使用。

8.2.8 补偿单元控制寄存器 (SYSCFG_CMPCR)

Compensation cell control register

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
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved							READY	Reserved							CMP_PD
							r								rw

位 31:9 保留，必须保持复位值。

位 8 **READY**：补偿单元就绪标志 (Compensation cell ready flag)
0：I/O 补偿单元未就绪
1：I/O 补偿单元就绪

位 7:2 保留，必须保持复位值。

位 0 **CMP_PD**：补偿单元掉电 (Compensation cell power-down)
0：I/O 补偿单元掉电（关闭）
1：I/O 补偿单元上电（使能）

8.2.9 SYSCFG 寄存器映射

下表列出了 SYSCFG 寄存器映射和复位值。

表 33. SYSCFG 寄存器映射和复位值 STM32F405xx/07xx 和 STM32F415xx/17xx

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	SYSCFG_MEMRM	Reserved																														MEM_MODE	
	Reset value																															x	x
0x04	SYSCFG_PMC	Reserved								MII_RMII_SEL	Reserved							Reserved															
	Reset value									0																							
0x08	SYSCFG_EXTICR1	Reserved																EXTI3[3:0]				EXTI2[3:0]				EXTI1[3:0]				EXTI0[3:0]			
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	SYSCFG_EXTICR2	Reserved																EXTI7[3:0]				EXTI6[3:0]				EXTI5[3:0]				EXTI4[3:0]			
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	SYSCFG_EXTICR3	Reserved																EXTI11[3:0]				EXTI10[3:0]				EXTI9[3:0]				EXTI8[3:0]			
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14	SYSCFG_EXTICR4	Reserved																EXTI15[3:0]				EXTI14[3:0]				EXTI13[3:0]				EXTI12[3:0]			
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20	SYSCFG_CMPCR	Reserved																								READY	Reserved						CMP_PD
	Reset value																									0							0

表 34. SYSCFG 寄存器映射和复位值（STM32F42xxx 和 STM32F43xxx）

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	SYSCFG_MEMRM	Reserved																														MEM_MODE	
	Reset value																															x	x
0x04	SYSCFG_PMC	Reserved								MII_RMII_SEL	Reserved				ADC3DC2	ADC2DC2	ADC1DC2	Reserved															
	Reset value									0					0	0	0																
0x08	SYSCFG_EXTICR1	Reserved																EXTI3[3:0]				EXTI2[3:0]				EXTI1[3:0]				EXTI0[3:0]			
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	SYSCFG_EXTICR2	Reserved																EXTI7[3:0]				EXTI6[3:0]				EXTI5[3:0]				EXTI4[3:0]			
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	SYSCFG_EXTICR3	Reserved																EXTI11[3:0]				EXTI10[3:0]				EXTI9[3:0]				EXTI8[3:0]			
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14	SYSCFG_EXTICR4	Reserved																EXTI15[3:0]				EXTI14[3:0]				EXTI13[3:0]				EXTI12[3:0]			
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

表 34. SYSCFG 寄存器映射和复位值（STM32F42xxx 和 STM32F43xxx）（续）

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x20	SYSCFG_CMPCR	Reserved																								READY	Reserved						CMP_PD
	Reset value																									0							0

有关寄存器边界地址的信息，请参见*第 52 页的表 2*。

9 DMA 控制器 (DMA)

除非特别说明，否则本部分适用于整个 STM32F4xx 系列。

9.1 DMA 简介

直接存储器访问 (DMA) 用于在外设与存储器之间以及存储器与存储器之间提供高速数据传输。可以在无需任何 CPU 操作的情况下通过 DMA 快速移动数据。这样节省的 CPU 资源可供其它操作使用。

DMA 控制器基于复杂的总线矩阵架构，将功能强大的双 AHB 主总线架构与独立的 FIFO 结合在一起，优化了系统带宽。

两个 DMA 控制器总共有 16 个数据流（每个控制器 8 个），每一个 DMA 控制器都用于管理一个或多个外设的存储器访问请求。每个数据流总共可以有多达 8 个通道（或称请求）。每个通道都有一个仲裁器，用于处理 DMA 请求间的优先级。

9.2 DMA 主要特性

DMA 主要特性是：

- 双 AHB 主总线架构，一个用于存储器访问，另一个用于外设访问
- 仅支持 32 位访问的 AHB 从编程接口
- 每个 DMA 控制器有 8 个数据流，每个数据流有多达 8 个通道（或称请求）
- 每个数据流有单独的四级 32 位先进先出存储器缓冲区 (FIFO)，可用于 FIFO 模式或直接模式：
  - FIFO 模式：可通过软件将阈值级别选取为 FIFO 大小的 1/4、1/2 或 3/4
  - 直接模式

    每个 DMA 请求会立即启动对存储器的传输。当在直接模式（禁止 FIFO）下将 DMA 请求配置为以存储器到外设模式传输数据时，DMA 仅会将一个数据从存储器预加载到内部 FIFO，从而确保一旦外设触发 DMA 请求时则立即传输数据。
- 通过硬件可以将每个数据流配置为：
  - 支持外设到存储器、存储器到外设和存储器到存储器传输的常规通道
  - 也支持在存储器方双缓冲的双缓冲区通道
- 8 个数据流中的每一个都连接到专用硬件 DMA 通道（请求）
- DMA 数据流请求之间的优先级可用软件编程（4 个级别：非常高、高、中、低），在软件优先级相同的情况下可以通过硬件决定优先级（例如，请求 0 的优先级高于请求 1）
- 每个数据流也支持通过软件触发存储器到存储器的传输（仅限 DMA2 控制器）
- 可供每个数据流选择的通道请求多达 8 个。此选择可由软件配置，允许几个外设启动 DMA 请求
- 要传输的数据项的数目可以由 DMA 控制器或外设管理：
  - DMA 流控制器：要传输的数据项的数目是 1 到 65535，可用软件编程
  - 外设流控制器：要传输的数据项的数目未知并由源或目标外设控制，这些外设通过硬件发出传输结束的信号
- 独立的源和目标传输宽度（字节、半字、字）：源和目标的数据宽度不相等时，DMA 自动封装/解封必要的传输数据来优化带宽。这个特性仅在 FIFO 模式下可用

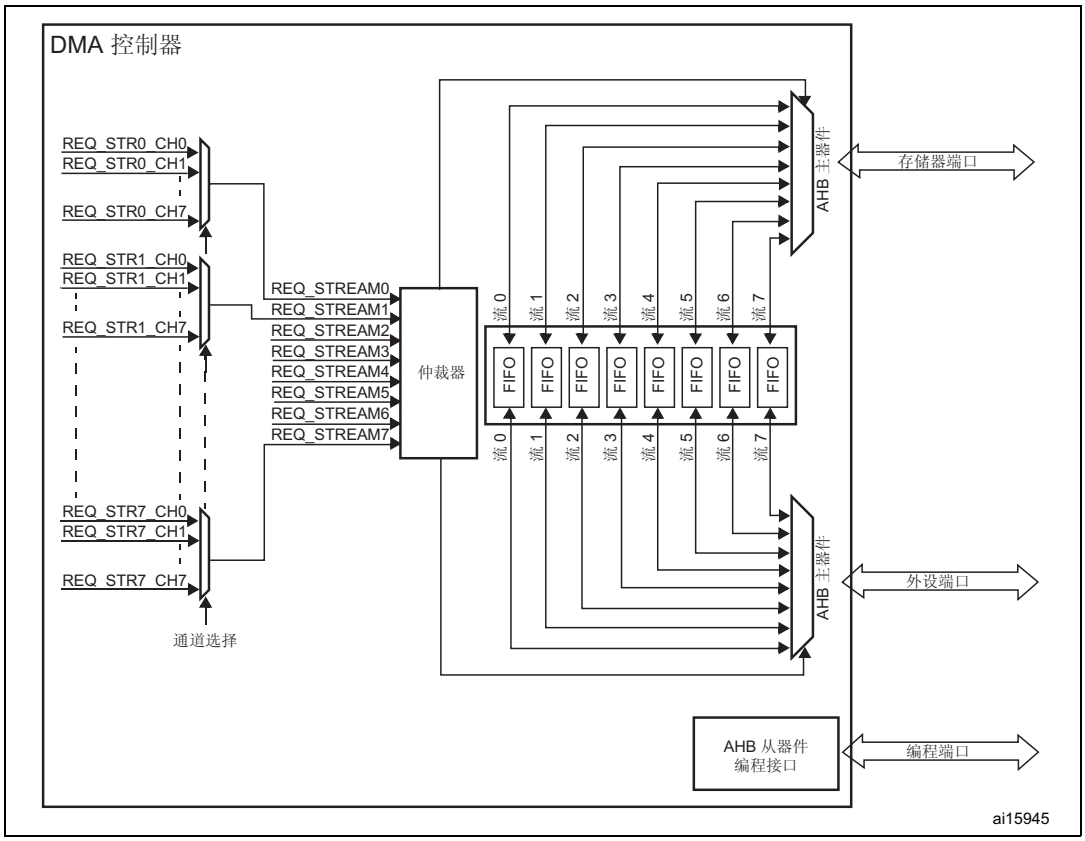
- 对源和目标的增量或非增量寻址
- 支持 4 个、8 个和 16 个节拍的增量突发传输。突发增量的大小可由软件配置，通常等于外设 FIFO 大小的一半
- 每个数据流都支持循环缓冲区管理
- 5 个事件标志（DMA 半传输、DMA 传输完成、DMA 传输错误、DMA FIFO 错误、直接模式错误），进行逻辑或运算，从而产生每个数据流的单个中断请求

9.3 DMA 功能说明

9.3.1 一般说明

图 24 显示了 DMA 的框图。

图 24. DMA 框图



DMA 控制器执行直接存储器传输：因为采用 AHB 主总线，它可以控制 AHB 总线矩阵来启动 AHB 事务。

它可以执行下列事务：

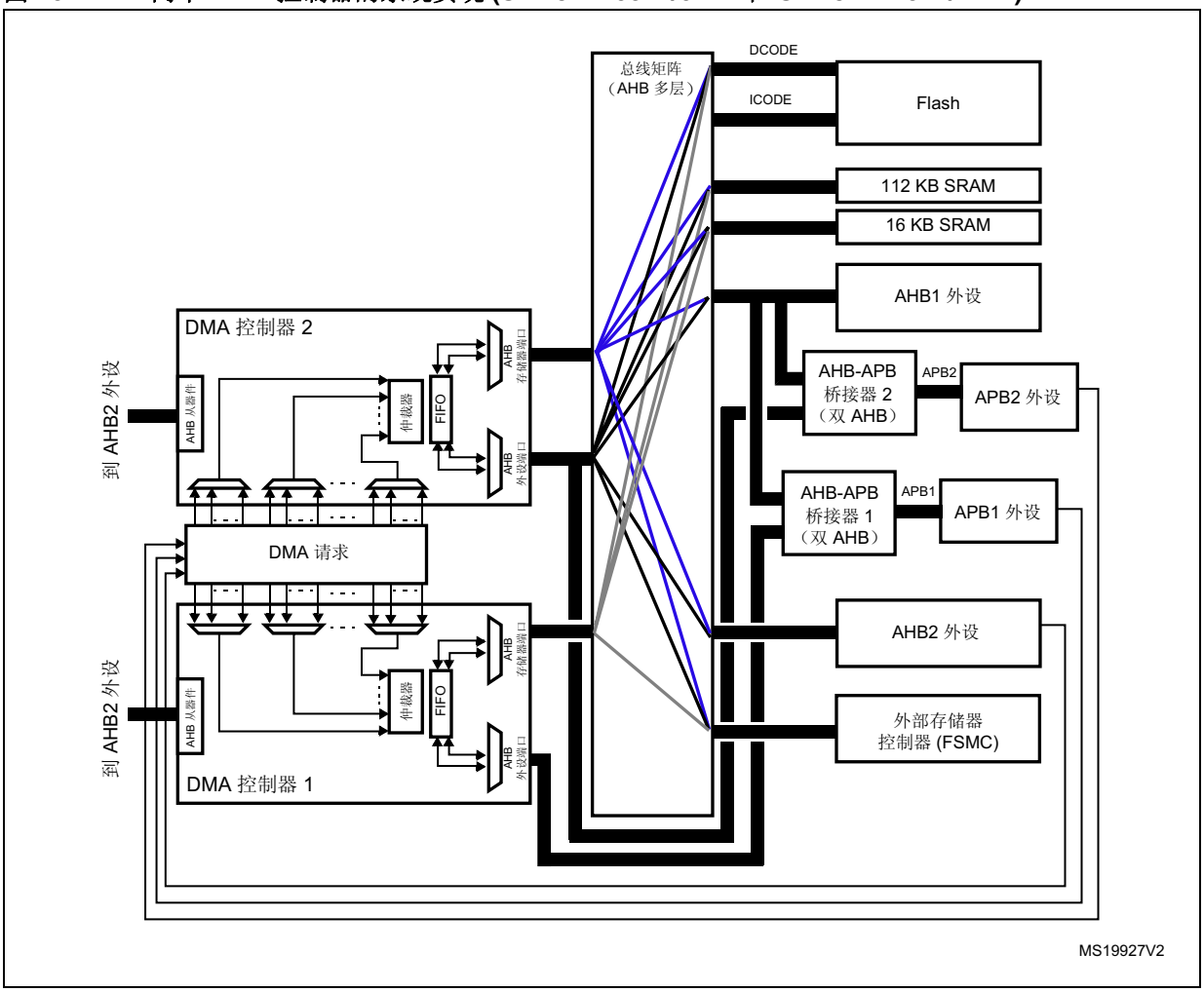
- 外设到存储器的传输
- 存储器到外设的传输
- 存储器到存储器的传输

DMA 控制器提供两个 AHB 主端口：*AHB 存储器端口*（用于连接存储器）和 *AHB 外设端口*（用于连接外设）。但是，要执行存储器到存储器的传输，*AHB 外设端口*必须也能访问存储器。

AHB 从端口用于对 DMA 控制器进行编程（它仅支持 32 位访问）。

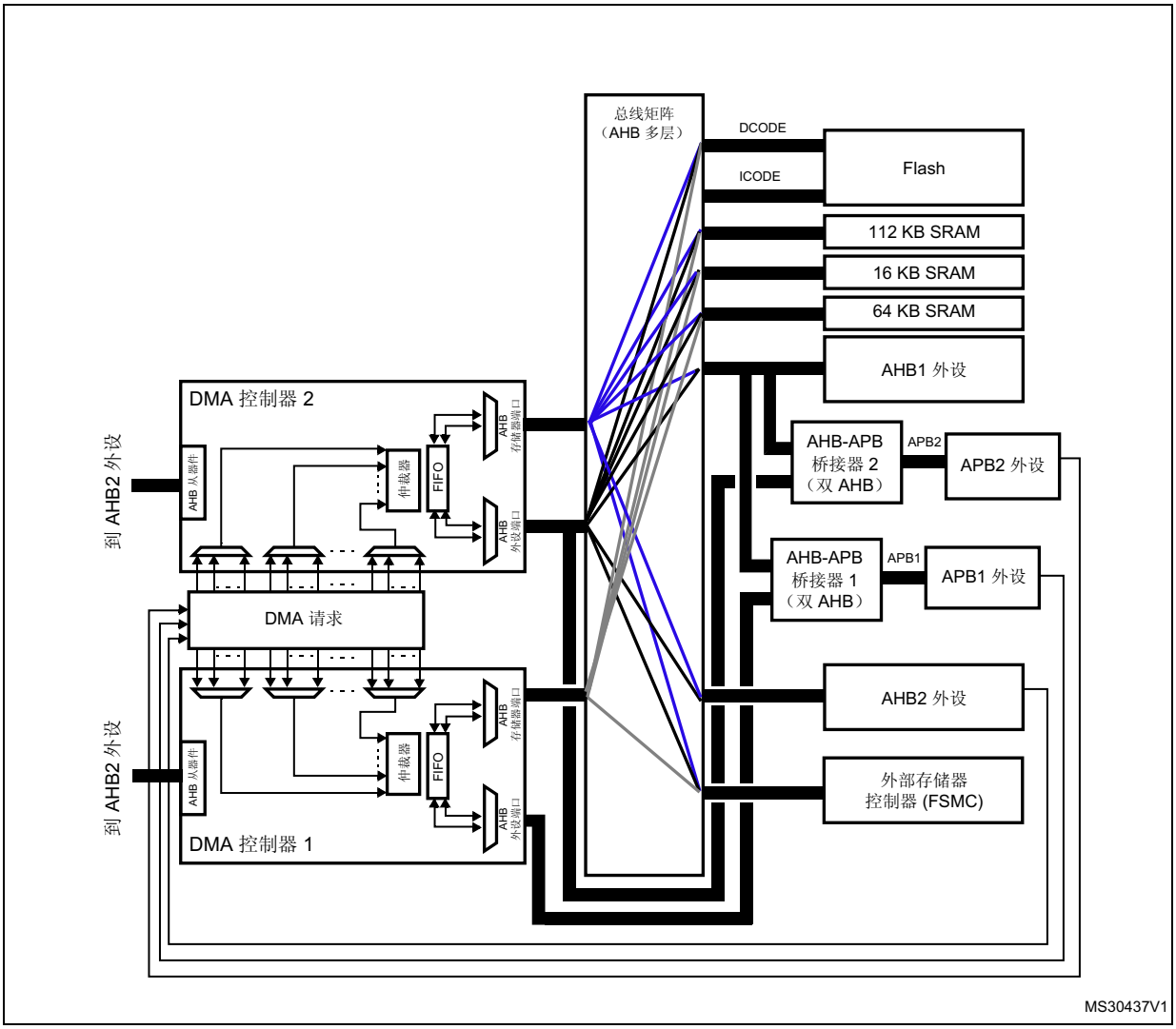
有关两个 DMA 控制器的系统实现，请参见*图 25* 和*图 26*。

图 25. 两个 DMA 控制器的系统实现 (STM32F405xx/07xx 和 STM32F415xx/17xx)



1. DMA1 控制器 AHB 外设端口与 DMA2 控制器的情况不同，不连接到总线矩阵，因此，仅 DMA2 数据流能够执行存储器到存储器的传输。

图 26. 两个 DMA 控制器的系统实现（STM32F42xxx 和 STM32F43xxx）



1. DMA1 控制器 AHB 外设端口与 DMA2 控制器的情况不同，不连接到总线矩阵，因此，仅 DMA2 数据流能够执行存储器到存储器的传输。

9.3.2 DMA 事务

DMA 事务由给定数目的数据传输序列组成。要传输的数据项的数目及其宽度（8 位、16 位或 32 位）可用软件编程。

每个 DMA 传输包含三项操作：

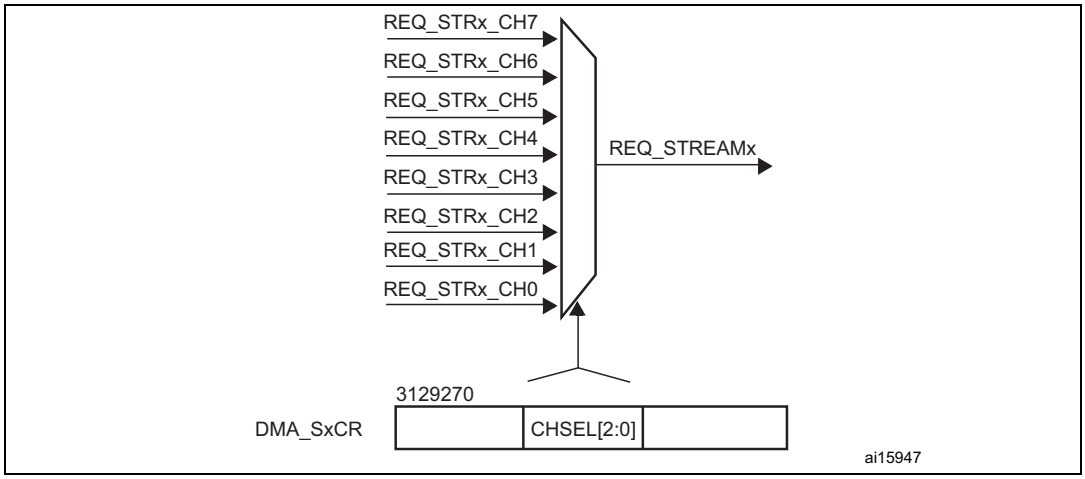
- 通过 DMA_SxPAR 或 DMA_SxM0AR 寄存器寻址，从外设数据寄存器或存储器单元中加载数据。
- 通过 DMA_SxPAR 或 DMA_SxM0AR 寄存器寻址，将加载的数据存储到外设数据寄存器或存储器单元。
- DMA_SxNDTR 计数器在数据存储结束后递减，该计数器中包含仍需执行的事务数。

在产生事件后，外设会向 DMA 控制器发送请求信号。DMA 控制器根据通道优先级处理该请求。只要 DMA 控制器访问外设，DMA 控制器就会向外设发送确认信号。外设获得 DMA 控制器的确认信号后，便会立即释放其请求。一旦外设使请求失效，DMA 控制器就会释放确认信号。如果有更多请求，外设可以启动下一个事务。

9.3.3 通道选择

每个数据流都与一个 DMA 请求相关联，此 DMA 请求可以从 8 个可能的通道请求中选出。此选择由 DMA_SxCR 寄存器中的 CHSEL[2:0] 位控制。

图 27. 通道选择



来自外设的 8 个请求（TIM、ADC、SPI、I2C 等）独立连接到每个通道，具体的连接取决于产品实现情况。

表 35 和 *表 36* 给出了 DMA 请求映射的示例。

表 35. DMA1 请求映射

外设请求	数据流 0	数据流 1	数据流 2	数据流 3	数据流 4	数据流 5	数据流 6	数据流 7
通道 0	SPI3_RX		SPI3_RX	SPI2_RX	SPI2_TX	SPI3_TX		SPI3_TX
通道 1	I2C1_RX		TIM7_UP		TIM7_UP	I2C1_RX	I2C1_TX	I2C1_TX
通道 2	TIM4_CH1		I2S3_EXT_RX	TIM4_CH2	I2S2_EXT_TX	I2S3_EXT_TX	TIM4_UP	TIM4_CH3
通道 3	I2S3_EXT_RX	TIM2_UP TIM2_CH3	I2C3_RX	I2S2_EXT_RX	I2C3_TX	TIM2_CH1	TIM2_CH2 TIM2_CH4	TIM2_UP TIM2_CH4
通道 4	UART5_RX	USART3_RX	UART4_RX	USART3_TX	UART4_TX	USART2_RX	USART2_TX	UART5_TX
通道 5	UART8_TX(1)	UART7_TX(1)	TIM3_CH4 TIM3_UP	UART7_RX(1)	TIM3_CH1 TIM3_TRIG	TIM3_CH2	UART8_RX(1)	TIM3_CH3
通道 6	TIM5_CH3 TIM5_UP	TIM5_CH4 TIM5_TRIG	TIM5_CH1	TIM5_CH4 TIM5_TRIG	TIM5_CH2		TIM5_UP	
通道 7		TIM6_UP	I2C2_RX	I2C2_RX	USART3_TX	DAC1	DAC2	I2C2_TX

1. 这些请求仅在 STM32F42xxx 和 STM32F43xxx 上可用。

表 36. DMA2 请求映射

外设请求	数据流 0	数据流 1	数据流 2	数据流 3	数据流 4	数据流 5	数据流 6	数据流 7
通道 0	ADC1		TIM8_CH1 TIM8_CH2 TIM8_CH3		ADC1		TIM1_CH1 TIM1_CH2 TIM1_CH3	
通道 1		DCMI	ADC2	ADC2		SPI6_TX(1)	SPI6_RX(1)	DCMI
通道 2	ADC3	ADC3		SPI5_RX(1)	SPI5_TX(1)	CRYP_OUT	CRYP_IN	HASH_IN
通道 3	SPI1_RX		SPI1_RX	SPI1_TX		SPI1_TX		
通道 4	SPI4_RX(1)	SPI4_TX(1)	USART1_RX	SDIO		USART1_RX	SDIO	USART1_TX
通道 5		USART6_RX	USART6_RX	SPI4_RX(1)	SPI4_TX(1)		USART6_TX	USART6_TX
通道 6	TIM1_TRIG	TIM1_CH1	TIM1_CH2	TIM1_CH1	TIM1_CH4 TIM1_TRIG TIM1_COM	TIM1_UP	TIM1_CH3	
通道 7		TIM8_UP	TIM8_CH1	TIM8_CH2	TIM8_CH3	SPI5_RX(1)	SPI5_TX(1)	TIM8_CH4 TIM8_TRIG TIM8_COM

1. 这些请求在 STM32F42xxx 和 STM32F43xxx 上可用。

9.3.4 仲裁器

仲裁器为两个 AHB 主端口（存储器和外设端口）提供基于请求优先级的 8 个 DMA 数据流请求管理，并启动外设/存储器访问序列。

优先级管理分为两个阶段：

- 软件：每个数据流优先级都可以在 DMA_SxCR 寄存器中配置。分为四个级别：
  - 非常高优先级
  - 高优先级
  - 中优先级
  - 低优先级
- 硬件：如果两个请求具有相同的软件优先级，则编号低的数据流优先于编号高的数据流。例如，数据流 2 的优先级高于数据流 4。

9.3.5 DMA 数据流

8 个 DMA 控制器数据流都能够提供源和目标之间的单向传输链路。

每个数据流配置后都可以执行：

- 常规类型事务：存储器到外设、外设到存储器或存储器到存储器的传输。
- 双缓冲区类型事务：使用存储器的两个存储器指针的双缓冲区传输（当 DMA 正在进行自/至缓冲区的读/写操作时，应用程序可以进行至/自其它缓冲区的写/读操作）。

要传输的数据量（多达 65535）可以编程，并与连接到外设 AHB 端口的外设（请求 DMA 传输）的源宽度相关。每个事务完成后，包含要传输的数据项总量的寄存器都会递减。

9.3.6 源、目标和传输模式

源传输和目标传输在整个 4 GB 区域（地址在 0x0000 0000 和 0xFFFF FFFF 之间）都可以寻址外设和存储器。

传输方向使用 DMA_SxCR 寄存器中的 DIR[1:0] 位进行配置，有三种可能的传输方向：存储器到外设、外设到存储器或存储器到存储器。*表 37* 介绍了相应的源和目标地址。

表 37. 源和目标地址

DMA_SxCR 寄存器的位 DIR[1:0]	方向	源地址	目标地址
00	外设到存储器	DMA_SxPAR	DMA_SxM0AR
01	存储器到外设	DMA_SxM0AR	DMA_SxPAR
10	存储器到存储器	DMA_SxPAR	DMA_SxM0AR
11	保留	-	-

当数据宽度（在 DMA_SxCR 寄存器的 PSIZE 或 MSIZE 位中编程）分别是半字或字时，写入 DMA_SxPAR 或 DMA_SxM0AR/M1AR 寄存器的外设或存储器地址必须分别在字或半字地址的边界对齐。

外设到存储器模式

图 28 介绍了这种模式。

使能这种模式（将 DMA_SxCR 寄存器中的位 EN 置 1）时，每次产生外设请求，数据流都会启动数据源到 FIFO 的传输。

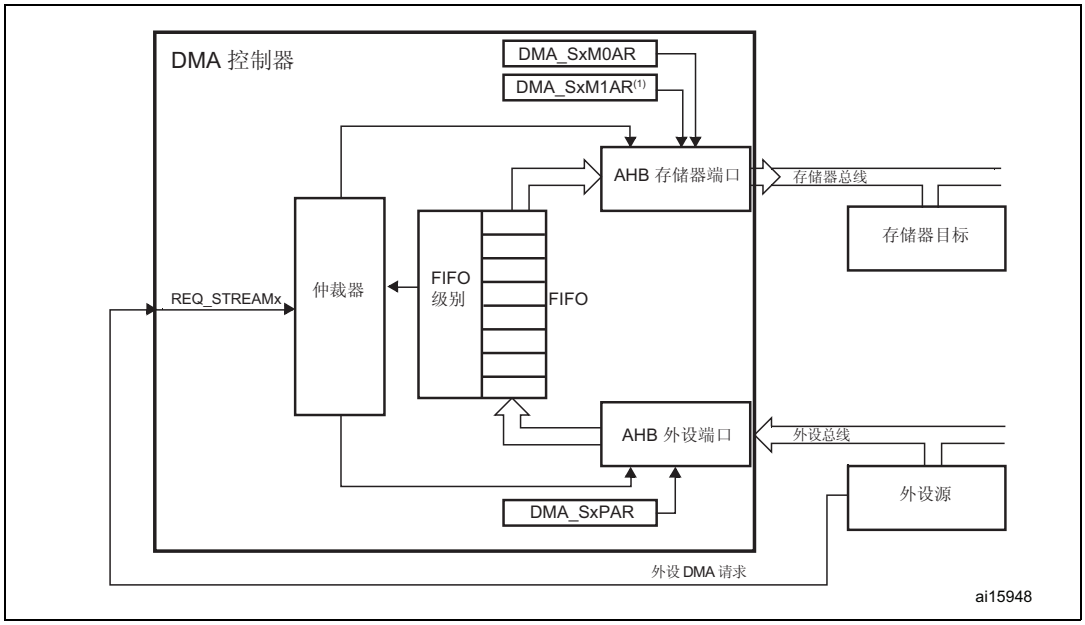
达到 FIFO 的阈值级别时，FIFO 的内容移出并存储到目标中。

如果 DMA_SxNDTR 寄存器达到零、外设请求传输终止（在使用外设流控制器的情况下）或 DMA_SxCR 寄存器中的 EN 位由软件清零，传输即会停止。

在直接模式下（当 DMA_SxFCR 寄存器中的 DMDIS 值为“0”时），不使用 FIFO 的阈值级别控制：每完成一次从外设到 FIFO 的数据传输后，相应的数据立即就会移出并存储到目标中。

只有赢得了数据流的仲裁后，相应数据流才有权访问 AHB 源或目标端口。系统使用在 DMA_SxCR 寄存器 PL[1:0] 位中为每个数据流定义的优先级执行仲裁。

图 28. 外设到存储器模式



1. 用于双缓冲区模式。

存储器到外设模式

图 29 介绍了这种模式。

使能这种模式（将 DMA_SxCR 寄存器中的 EN 位置 1）时，数据流会立即启动传输，从源完全填充 FIFO。

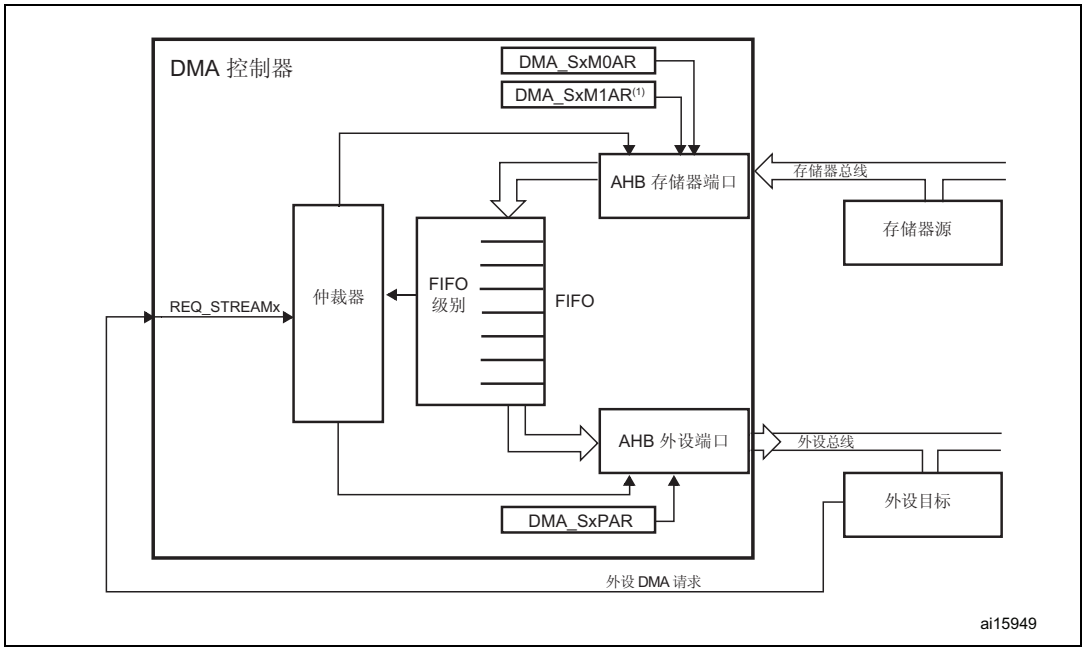
每次发生外设请求，FIFO 的内容都会移出并存储到目标中。当 FIFO 的级别小于或等于预定义的阈值级别时，将使用存储器中的数据完全重载 FIFO。

如果 DMA_SxNDTR 寄存器达到零、外设请求传输终止（在使用外设流控制器的情况下）或 DMA_SxCR 寄存器中的 EN 位由软件清零，传输即会停止。

在直接模式下（当 DMA_SxFCR 寄存器中的 DMDIS 值为“0”时），不使用 FIFO 的阈值级别。一旦使能了数据流，DMA 便会预装载第一个数据，将其传输到内部 FIFO。一旦外设请求数据传输，DMA 便会将预装载的值传输到配置的目标。然后，它会使用要传输的下一个数据再次重载内部空 FIFO。预装载的数据大小为 DMA_SxCR 寄存器中 PSIZE 位字段的值。

只有赢得了数据流的仲裁后，相应数据流才有权访问 AHB 源或目标端口。系统使用在 DMA_SxCR 寄存器 PL[1:0] 位中为每个数据流定义的优先级执行仲裁。

图 29. 存储器到外设模式



1. 用于双缓冲区模式。

存储器到存储器模式

DMA 通道在没有外设请求触发的情况下同样可以工作。此为*图 30* 中介绍的存储器到存储器模式。

通过将 DMA_SxCR 寄存器中的使能位 (EN) 置 1 来使能数据流时，数据流会立即开始填充 FIFO，直至达到阈值级别。达到阈值级别后，FIFO 的内容便会移出，并存储到目标中。

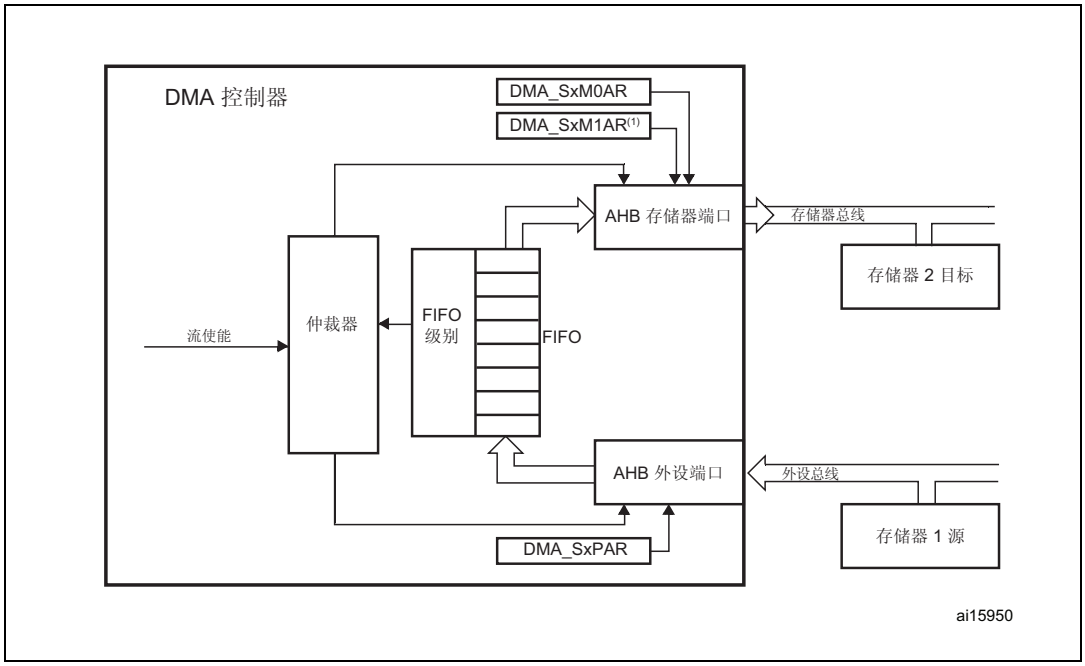
如果 DMA_SxNDTR 寄存器达到零或 DMA_SxCR 寄存器中的 EN 位由软件清零，传输即会停止。

只有赢得了数据流的仲裁后，相应数据流才有权访问 AHB 源或目标端口。系统使用在 DMA_SxCR 寄存器 PL[1:0] 位中为每个数据流定义的优先级执行仲裁。

注意： *使用存储器到存储器模式时，不允许循环模式和直接模式。*

只有 DMA2 控制器能够执行存储器到存储器的传输。

图 30. 存储器到存储器模式



1. 用于双缓冲区模式。

9.3.7 指针递增

根据 DMA_SxCR 寄存器中 PINC 和 MINC 位的状态，外设和存储器指针在每次传输后可以自动向后递增或保持常量。

通过单个寄存器访问外设源或目标数据时，禁止递增模式十分有用。

如果使能了递增模式，则根据在 DMA_SxCR 寄存器 PSIZE 或 MSIZE 位中编程的数据宽度，下一次传输的地址将是前一次传输的地址递增 1（对于字节）、2（对于半字）或 4（对于字）。

为了优化封装操作，可以不管 AHB 外设端口上传输的数据的大小，将外设地址的增量偏移大小固定下来。DMA_SxCR 寄存器中的 PINCOS 位用于将增量偏移大小与外设 AHB 端口或 32 位地址（此时地址递增 4）上的数据大小对齐。PINCOS 位仅对 AHB 外设端口有影响。

如果将 PINCOS 位置 1，则不论 PSIZE 值是多少，下一次传输的地址总是前一次传输的地址递增 4（自动与 32 位地址对齐）。但是，AHB 存储器端口不受此操作影响。

如果 AHB 外设端口或 AHB 存储器端口分别请求突发事务，为了满足 AMBA 协议（在固定地址模式下不允许突发事务），则需要将 PINC 或 MINC 位置 1。

9.3.8 循环模式

循环模式可用于处理循环缓冲区和连续数据流（例如 ADC 扫描模式）。可以使用 DMA_SxCR 寄存器中的 CIRC 位使能此特性。

当激活循环模式时，要传输的数据项的数目在数据流配置阶段自动用设置的初始值进行加载，并继续响应 DMA 请求。

注意： *在循环模式下，如果为存储器配置了突发模式，必须遵循下列规则：*

DMA_SxNDTR = ((Mburst 节拍) × (Msize)/(Psize)) 的倍数，其中：

- *(Mburst 节拍) = 4、8 或 16（取决于 DMA_SxCR 寄存器中的 MBURST 位）*
- *((Msize)/(Psize)) = 1、2、4、1/2 或 1/4（Msize 和 Psize 表示 DMA_SxCR 寄存器中的 MSIZE 和 PSIZE 位。它们与字节相关）*
- *DMA_SxNDTR = AHB 外设端口上要传输的数据项的数目*

例如：Mburst 节拍 = 8 (INCR8)，MSIZE = “00”（字节）和 PSIZE = “01”（半字），在此例中：DMA_SxNDTR 必须是 (8 × 1/2 = 4) 的倍数。

如果不遵循此公式，则 DMA 行为和数据完整性得不到保证。

NDTR 还必须是外设突发大小与外设数据大小乘积的倍数，否则会导致错误的 DMA 行为。

9.3.9 双缓冲区模式

此模式可用于所有 DMA1 和 DMA2 数据流。

通过将 DMA_SxCR 寄存器中的 DBM 位置 1，即可使能双缓冲区模式。

除了有两个存储器指针之外，双缓冲区数据流的工作方式与常规（单缓冲区）数据流的一样。使能双缓冲区模式时，将自动使能循环模式（DMA_SxCR 中的 CIRC 位的状态是“无关”），并在每次事务结束时交换存储器指针。

在此模式下，每次事务结束时，DMA 控制器都从一个存储器目标交换为另一个存储器目标。这样，软件在处理一个存储器区域的同时，DMA 传输还可以填充/使用第二个存储器区域。如*表 38：双缓冲区模式下的源和目标地址寄存器 (DBM=1)* 所述，双缓冲区数据流可以双向工作（存储器既可以是源也可以是目标）。

注意： *在双缓冲区模式下使能数据流时，可遵循下列条件，实时更新 AHB 存储器的基址（DMA_SxM0AR 或 DMA_SxM1AR）：*

- *当 DMA_SxCR 寄存器中的 CT 位为“0”时，可以写入 DMA_SxM1AR 寄存器。当 CT =“1”时，试图写入此寄存器会将错误标志位 (TEIF) 置 1，并自动禁止数据流。*
- *当 DMA_SxCR 寄存器中的 CT 位为“1”时，可以写入 DMA_SxM0AR 寄存器。当 CT =“0”时，试图写入此寄存器会将错误标志位 (TEIF) 置 1，并自动禁止数据流。*

为避免出现任何错误状态，建议在 TCIF 标志位置位时立即更改基址。因为此时根据上述两个条件之一，目标存储器依据 DMA_SxCR 寄存器中 CT 值的情况，一定已从存储器 0 更改为存储器 1（或从存储器 1 更改为存储器 0）。

对于所有其它模式（双缓冲区模式除外），一旦使能数据流，存储器地址寄存器即被写保护。

表 38. 双缓冲区模式下的源和目标地址寄存器 (DBM=1)

DMA_SxCR 寄存器的位 DIR[1:0]	方向	源地址	目标地址
00	外设到存储器	DMA_SxPAR	DMA_SxM0AR/DMA_SxM1AR
01	存储器到外设	DMA_SxM0AR/DMA_SxM1AR	DMA_SxPAR
10	不允许(1)		
11	保留	-	-

1. 使能双缓冲区模式时，自动使能循环模式。由于存储器到存储器模式与循环模式不兼容，所以当使能双缓冲区模式时，不允许配置存储器到存储器模式。

9.3.10 可编程数据宽度、封装/解封、字节序

要传输的数据项数目必须在使能数据流之前编程到 DMA_SxNDTR（要传输数据项数目位，NDT）中，当流控制器是外设且 DMA_SxCR 中的 PFCTRL 位置为 1 时除外。

当使用内部 FIFO 时，源和目标数据的数据宽度可以通过 DMA_SxCR 寄存器的 PSIZE 和 MSIZE 位（可以是 8、16 或32 位）编程。

当 PSIZE 和 MSIZE 不相等时：

- 在 DMA_SxNDTR 寄存器中配置的要传输的数据项数目的数据宽度等于外设总线的宽度（由 DMA_SxCR 寄存器中的 PSIZE 位配置）。例如，在外设到存储器、存储器到外设或存储器到存储器传输的情况下，如果将 PSIZE[1:0] 位配置为半字，则要传输的字节数等于 2 × NDT。
- DMA 控制器仅按小字节序寻址源和目标。相关内容在*表 39：封装/解封和字节序行为（位 PINC = MINC = 1）*中介绍。

在封装/解封数据的过程中，如果在数据完全封装/解封前中断操作，则有数据损坏的危险。因此，为了确保数据一致性，可将数据流配置成生成突发传输：在这种情况下，属于一个突发的每组传输不可分割（请参见*第 9.3.11 节：单次传输和突发传输*）。

在直接模式下（DMA_SxFCR 寄存器中的 DMDIS = 0），不能进行数据封装/解封。这种情况下，不允许源与目标的传输数据宽度不同，二者必须相等，并由 DMA_SxCR 中的 PSIZE 位定义，MSIZE 位的状态是“无关”。

表 39. 封装/解封和字节序行为（位 PINC = MINC = 1）

AHB 存储器端口宽度	AHB 外设端口宽度	要传输的数据项的数目 (NDT)	存储器传输数目	存储器端口地址/字节通道	外设传输数目	外设端口地址/字节通道	
						PINCOS = 1	PINCOS = 0
8	8	4	1 2 3 4	0x0/B0[7:0] 0x1/B1[7:0] 0x2/B2[7:0] 0x3/B3[7:0]	1 2 3 4	0x0/B0[7:0] 0x4/B1[7:0] 0x8/B2[7:0] 0xC/B3[7:0]	0x0/B0[7:0] 0x1/B1[7:0] 0x2/B2[7:0] 0x3/B3[7:0]
8	16	2	1 2 3 4	0x0/B0[7:0] 0x1/B1[7:0] 0x2/B2[7:0] 0x3/B3[7:0]	1 2	0x0/B1|B0[15:0] 0x4/B3|B2[15:0]	0x0/B1|B0[15:0] 0x2/B3|B2[15:0]
8	32	1	1 2 3 4	0x0/B0[7:0] 0x1/B1[7:0] 0x2/B2[7:0] 0x3/B3[7:0]	1	0x0/B3|B2|B1|B0[31:0]	0x0/B3|B2|B1|B0[31:0]
16	8	4	1 2	0x0/B1|B0[15:0] 0x2/B3|B2[15:0]	1 2 3 4	0x0/B0[7:0] 0x4/B1[7:0] 0x8/B2[7:0] 0xC/B3[7:0]	0x0/B0[7:0] 0x1/B1[7:0] 0x2/B2[7:0] 0x3/B3[7:0]
16	16	2	1 2	0x0/B1|B0[15:0] 0x2/B1|B0[15:0]	1 2	0x0/B1|B0[15:0] 0x4/B3|B2[15:0]	0x0/B1|B0[15:0] 0x2/B3|B2[15:0]
16	32	1	1 2	0x0/B1|B0[15:0] 0x2/B3|B2[15:0]	1	0x0/B3|B2|B1|B0[31:0]	0x0/B3|B2|B1|B0[31:0]
32	8	4	1	0x0/B3|B2|B1|B0[31:0]	1 2 3 4	0x0/B0[7:0] 0x4/B1[7:0] 0x8/B2[7:0] 0xC/B3[7:0]	0x0/B0[7:0] 0x1/B1[7:0] 0x2/B2[7:0] 0x3/B3[7:0]

表 39. 封装/解封和字节序行为（位 PINC = MINC = 1）（续）

AHB 存储器端口宽度	AHB 外设端口宽度	要传输的数据项的数目 (NDT)	存储器传输数目	存储器端口地址/字节通道	外设传输数目	外设端口地址/字节通道	
						PINCOS = 1	PINCOS = 0
32	16	2	1	0x0/B3|B2|B1|B0[31:0]	1 2	0x0/B1|B0[15:0] 0x4/B3|B2[15:0]	0x0/B1|B0[15:0] 0x2/B3|B2[15:0]
32	32	1	1	0x0/B3|B2|B1|B0[31:0]	1	0x0/B3|B2|B1|B0[31:0]	0x0/B3|B2|B1|B0[31:0]

注意： *外设端口可以是源或目标（在存储器到存储器传输的情况下，也可能是存储器源）。*

必须配置 PSIZE、MSIZE 和 NDT[15:0]，以确保最后一次传输的完整性。当外设端口的数据宽度（PSIZE 位）小于存储器端口的数据宽度（MSIZE 位）时，可能会发生数据传输不完整的情况。此限制条件汇总在*表 40* 中。

表 40. PSIZE 与 MSIZE 确定时对 NDT 的限制条件

DMA_SxCR 的 PSIZE[1:0]	DMA_SxCR 的 MSIZE[1:0]	DMA_SxNDTR 的 NDT[15:0]
00（8 位）	01（16 位）	必须是 2 的倍数
00（8 位）	10（32 位）	必须是 4 的倍数
01（16 位）	10（32 位）	必须是 2 的倍数

9.3.11 单次传输和突发传输

DMA 控制器可以产生单次传输或 4 个、8 个和 16 个节拍的增量突发传输。

突发大小通过软件针对两个 AHB 端口独立配置，配置时使用 DMA_SxCR 寄存器中的 MBURST[1:0] 和 PBURST[1:0] 位。

突发大小指示突发中的节拍数，而不是传输的字节数。

为确保数据一致性，形成突发的每一组传输都不可分割：在突发传输序列期间，AHB 传输会锁定，并且 AHB 总线矩阵的仲裁器不解除对 DMA 主总线的授权。

根据单次或突发配置的情况，每个 DMA 请求在 AHB 外设端口上相应地启动不同数量的传输。

- 当 AHB 外设端口被配置为单次传输时，根据 DMA_SxCR 寄存器 PSIZE[1:0] 位的值，每个 DMA 请求产生一次字节、半字或字的数据传输。
- 当 AHB 外设端口被配置为突发传输时，根据 DMA_SxCR 寄存器 PBURST[1:0] 和 PSIZE[1:0] 位的值，每个 DMA 请求相应地生成 4 个、8 个或 16 个节拍的字节、半字或字的传输。

对于需要配置 MBURST 和 MSIZE 位的 AHB 存储器端口，必须考虑与上述相同的内容。

在直接模式下，数据流只能生成单次传输，而 MBURST[1:0] 和 PBURST[1:0] 位由硬件强制配置。

必须选择地址指针（DMA_SxPAR 或 DMA_SxM0AR 寄存器），以确保一个突发块内的所有传输在等于传输大小的地址边界对齐。

选择突发配置必须要遵守 AHB 协议，即突发传输*不得*越过 1 KB 地址边界，因为可以分配给单个从设备的最小地址空间是 1 KB。这意味着突发块传输不应越过 1 KB 地址边界，否则就会产生一个 AHB 错误，并且 DMA 寄存器不会报告这个错误。

注意： *仅在使能指针递增模式时允许突发模式：*
– 当 PINC 位为“0”时，也应将 PBURST 位清为“00”
– 当 MINC 位为“0”时，也应将 MBURST 位清为“00”

9.3.12 FIFO

FIFO 结构

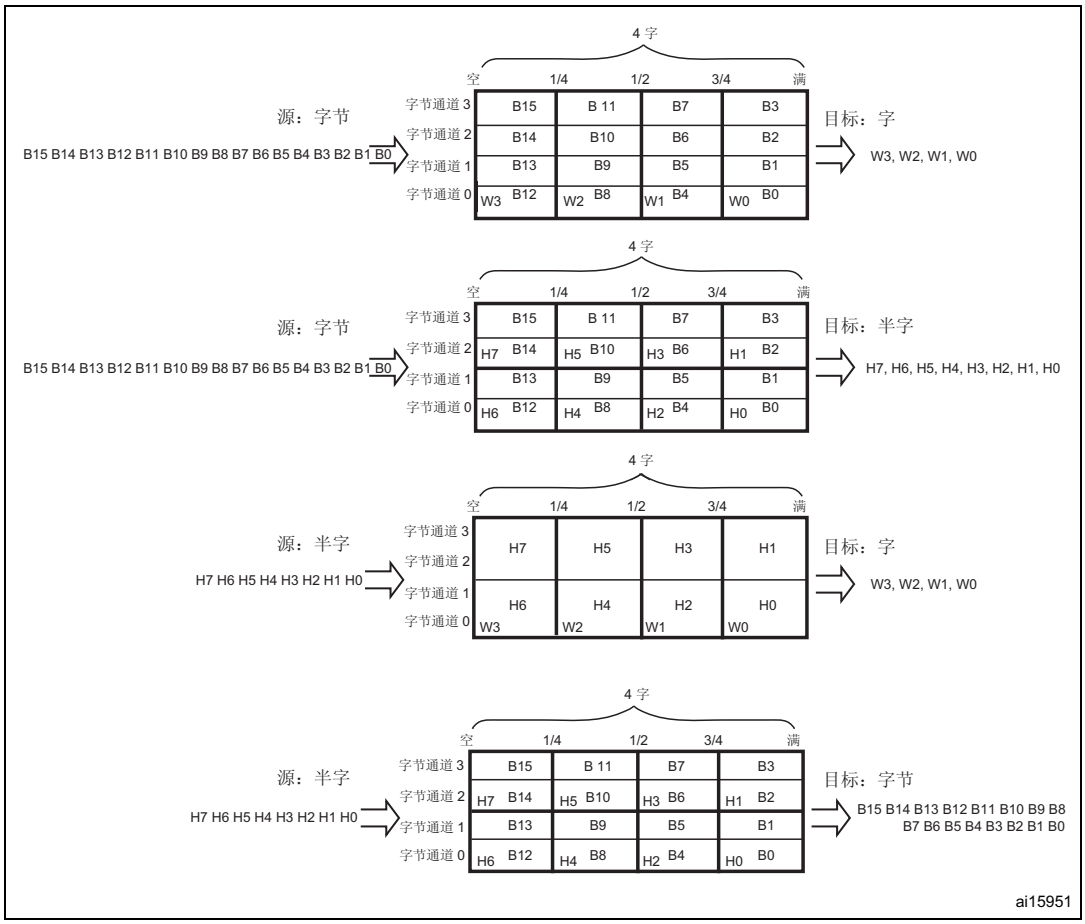
FIFO 用于在源数据传输到目标之前临时存储这些数据。

每个数据流都有一个独立的 4 字 FIFO，阈值级别可由软件配置为 1/4、1/2、3/4 或满。

为了使能 FIFO 阈值级别，必须通过将 DMA_SxFCR 寄存器中的 DMDIS 位置 1 来禁止直接模式。

FIFO 的结构随源与目标数据宽度而不同，相关内容在*图 31：FIFO 结构*中介绍。

图 31. FIFO 结构



FIFO 阈值与突发配置

选择 FIFO 阈值（DMA_SxFCR 寄存器的位 FTH[1:0]）和存储器突发大小（DMA_SxCR 寄存器的 MBURST[1:0] 位）时需要小心：FIFO 阈值指向的内容必须与整数个存储器突发传输完全匹配。如果不是这样，当使能数据流时将生成一个 FIFO 错误（DMA_HISR 或 DMA_LISR 寄存器的标志 FEIFx），然后将自动禁止数据流。允许的和禁止的配置在*表 41：FIFO 阈值配置*中介绍。

表 41. FIFO 阈值配置

MSIZE	FIFO 级别	MBURST = INCR4	MBURST = INCR8	MBURST = INCR16
字节	1/4	4 个节拍的 1 次突发	禁止	禁止
	1/2	4 个节拍的 2 次突发	8 个节拍的 1 次突发	
	3/4	4 个节拍的 3 次突发	禁止	
	满	4 个节拍的 4 次突发	8 个节拍的 2 次突发	16 个节拍的 1 次突发
半字	1/4	禁止	禁止	禁止
	1/2	4 个节拍的 1 次突发		
	3/4	禁止		
	满	4 个节拍的 2 次突发	8 个节拍的 1 次突发	
字	1/4	禁止	禁止	
	1/2			
	3/4			
	满	4 个节拍的 1 次突发		

所有这些情况下，突发大小与数据大小的乘积不得超过 FIFO 大小（数据大小可以为：1（字节）、2（半字）或 4（字））。

如果发生下列一种情况，会导致 DMA 传输结束时出现不完整的突发传输：

- 对于 AHB 外设端口配置：数据项总数（在 DMA_SxNDTR 寄存器中设置）不是突发大小与数据大小乘积的倍数。
- 对于 AHB 存储器端口配置：要传输到存储器的 FIFO 中的剩余数据项的数目不是突发大小与数据大小乘积的倍数。

在这些情况下，即使在 DMA 数据流配置期间请求突发事务，要传输的剩余数据也将由 DMA 在单独模式下管理。

注意： *当在外设 AHB 端口上请求突发传输并且使用 FIFO（DMA_SxCR 寄存器中 DMDIS = 1）时，必须根据 DMA 数据流方向遵守下列规则来避免出现上溢或下溢情况：*

如果 (PBURST × PSIZE) = FIFO_SIZE（4 字），则当 PSIZE = 1、2 或 4，PBURST = 4、8 或 16 时，禁止 FIFO_Threshold = 3/4。

此规则将确保一次释放足够的 FIFO 空间来处理外设的请求。

FIFO 刷新

当复位 DMA_SxCR 寄存器中的 EN 位来禁止数据流，以及配置数据流来管理外设到存储器或存储器到存储器的传输时，可以刷新 FIFO。因为如果禁止数据流时仍有某些数据存留在 FIFO 中，DMA 控制器会将剩余的数据继续传输到目标（即使已经有效禁止了数据流）。刷新完成时，会将 DMA_LISR 或 DMA_HISR 寄存器中的传输完成状态位 (TCIFx) 置 1。

在这种情况下，剩余数据计数器 DMA_SxNDTR 保持的值指示在目标存储器现有多少可用数据项。

请注意在 FIFO 刷新操作期间，如果 FIFO 中要传输到存储器的剩余数据项的数目（以字节为单位）小于存储器数据宽度（例如在 MSIZE 配置为字时 FIFO 中为 2 个字节），则会使用在 DMA_SxCR 寄存器中的 MSIZE 位设置的数据宽度发送数据。这意味着将使用非预期值写入存储器。软件可以读取 DMA_SxNDTR 寄存器来确定包含良好数据的存储器区域（起始地址和最后地址）。

如果 FIFO 中剩余数据项的数目小于突发大小，并且数据流通过将 DMA_SxCR 寄存器 MBURST 位置 1 进行配置来管理在 AHB 存储器端口上的突发传输，则使用单次传输来完成 FIFO 的刷新。

直接模式

默认情况下，FIFO 以直接模式操作（将 DMA_SxFCR 中的 DMDIS 位置 1），不使用 FIFO 阈值级别。如果在每次 DMA 请求之后，系统需要至/自存储器的立即和单独传输，这种模式非常有用。

当在直接模式（禁止 FIFO）下将 DMA 配置为以存储器到外设模式传输数据时，DMA 会将一个数据从存储器预加载到内部 FIFO，从而确保一旦外设触发 DMA 请求时则立即传输数据。

为了避免 FIFO 饱和，建议使用高优先级配置相应的数据流。

该模式仅限以下方式的传输：

- 源和目标传输宽度相等，并均由 DMA_SxCR 中的 PSIZE[1:0] 位定义（MSIZE[1:0] 位的状态是“无关”）
- 不可能进行突发传输（DMA_SxCR 中的 PBURST[1:0] 和 MBURST[1:0] 位的状态是“无关”）

当实现存储器到存储器传输时不得使用直接模式。

9.3.13 DMA 传输完成

以下各种事件均可以结束传输过程，并将 DMA_LISR 或 DMA_HISR 状态寄存器中的 TCIFx 位置 1：

- 在 DMA 流控制器模式下：
  - 在存储器到外设模式下，DMA_SxNDTR 计数器已达到零
  - 传输结束前禁止了数据流（通过将 DMA_SxCR 寄存器中的 EN 位清零），并在传输是外设到存储器或存储器到存储器的模式时，所有的剩余数据均已从 FIFO 刷新到存储器
- 在外设流控制器模式下：
  - 已从外设生成最后的外部突发请求或单独请求，并当 DMA 在外设到存储器模式下工作时，剩余数据已从 FIFO 传输到存储器
  - 数据流由软件禁止，并当 DMA 在外设到存储器模式下工作时，剩余数据已从 FIFO 传输到存储器

注意： *仅在外设到存储器模式下，传输的完成取决于 FIFO 中要传输到存储器的剩余数据。这种情况不适用于存储器到外设模式。*

如果是在非循环模式下配置数据流，传输结束后（即要传输的数据数目达到零），除非软件重新对数据流编程并重新使能数据流（通过将 DMA_SxCR 寄存器中的 EN 位置 1），否则 DMA 即会停止传输（通过硬件将 DMA_SxCR 寄存器中的 EN 位清零）并且不再响应任何 DMA 请求。

9.3.14 DMA 传输暂停

可以随时暂停 DMA 传输以供稍后重新开始；也可以在 DMA 传输结束前明确禁止暂停功能。

分为两种情况：

- 数据流禁止传输，以后不从停止点重新开始暂停。这种情况下，只需将 DMA_SxCR 寄存器中的 EN 位清零来禁止数据流，除此之外不需要任何其他操作。禁止数据流可能要花费一些时间（需要首先完成正在进行的传输）。需要将传输完成中断标志（DMA_LISR 或 DMA_HISR 寄存器中的 TCIF）置 1 来指示传输结束。现在 DMA_SxCR 中的 EN 位的值是“0”，借此确认数据流已经终止传输。DMA_SxNDTR 寄存器包含数据流停止时剩余数据项的数目，这样软件便可以确定数据流中断前已传输了多少数据项。
- 数据流在 DMA_SxNDTR 寄存器中要传输的剩余数据项数目达到 0 之前暂停传输。目的是以后通过重新使能数据流重新开始传输。为了在传输停止点重新开始传输，软件必须在通过写入 DMA_SxCR 寄存器中的 EN 位（然后检查确认该位为‘0’）禁止数据流之后，首先读取 DMA_SxNDTR 寄存器来了解已经收集的数据项的数目。然后：
  - 必须更新外设和/或存储器地址以调整地址指针
  - 必须使用要传输的剩余数据项的数目（禁止数据流时读取的值）更新 SxNDTR 寄存器
  - 然后可以重新使能数据流，从停止点重新开始传输

注意： *请注意，传输完成中断标志（DMA_LISR 或 DMA_HISR 中的 TCIF）置 1 将指示因数据流中断而结束传输。*

9.3.15 流控制器

控制要传输的数据数目的实体称为流控制器。此流控制器使用 DMA_SxCR 寄存器中的 PFCTRL 位针对每个数据流独立配置。

流控制器可以是：

- DMA 控制器：在这种情况下，要传输的数据项的数目在使能 DMA 数据流之前由软件编程到 DMA_SxNDTR 寄存器。
- 外设源或目标：当要传输的数据项的数目未知时属于这种情况。当所传输的是最后的数据时，外设通过硬件向 DMA 控制器发出指示。仅限能够发出传输结束信号的外设支持此功能，也就是：
  - SDIO

当外设流控制器用于给定数据流时，写入 DMA_SxNDTR 的值对 DMA 传输没有作用。实际上，不论写入什么值，一旦使能数据流，硬件即会将该值强制置为 0xFFFF 来执行下列方案：

- 预期的数据流中断：DMA_SxCR 寄存器中的 EN 位由软件重置为 0，以在外设发送最后的数据硬件信号（单独或突发）之前停止数据流。这样，在外设到存储器 DMA 传输的情况下，数据流即会关闭并触发 FIFO 刷新。状态寄存器中相应数据流的 TCIFx 标志置 1 以指示 DMA 完成传输。要了解 DMA 传输期间传输的数据项的数目，请读取 DMA_SxNDTR 寄存器并应用下列公式：
  - 传输的数据数目 = 0xFFFF – DMA_SxNDTR
- 因接收到最后的数据硬件信号而引起的正常数据流中断：当外设请求最后的传输（单独或突发）并当此传输完成时，自动中断数据流。相应流的 TCIFx 标志在状态寄存器中置 1 以指示 DMA 传输完成。要了解传输的数据项的数目，请读取 DMA_SxNDTR 寄存器并应用与上面相同的公式。

- DMA_SxNDTR 寄存器达到 0：状态寄存器中相应数据流的 TCIFx 标志置 1 以指示强制的 DMA 传输完成。即使尚未置位最后的数据硬件信号（单独或突发），也会自动关闭数据流。已传输的数据不会丢失。这意味着即使在外设流控制模式下，DMA 在单独的事务中最多处理 65535 个数据项。

注意： *当在存储器到存储器模式下配置时，DMA 始终是流控制器，而 PFCTRL 位由硬件强制置为 0。*

在外设流控制器模式下禁止循环模式。

9.3.16 可能的 DMA 配置汇总

表 42 汇总了各种可能的 DMA 配置。

表 42. 可能的 DMA 配置

DMA 传输模式	源	目标	流控制器	循环模式	传输类型	直接模式	双缓冲区模式
外设到存储器	AHB 外设端口	AHB 存储器端口	DMA	允许	单独	允许	允许
					突发	禁止	
			外设	禁止	单独	允许	禁止
					突发	禁止	
存储器到外设	AHB 存储器端口	AHB 外设端口	DMA	允许	单独	允许	允许
					突发	禁止	
			外设	禁止	单独	允许	禁止
					突发	禁止	
存储器到存储器	AHB 外设端口	AHB 存储器端口	仅 DMA	禁止	单独	禁止	禁止
					突发		

9.3.17 流配置过程

配置 DMA 数据流 x（其中 x 是数据流编号）时应遵守下面的顺序：

1. 如果使能了数据流，通过重置 DMA_SxCR 寄存器中的 EN 位将其禁止，然后读取此位以确认没有正在进行的数据流操作。将此位写为 0 不会立即生效，因为实际上只有所有当前传输都已完成时才会将其写为 0。当所读取 EN 位的值为 0 时，才表示可以配置数据流。因此在开始任何数据流配置之前，需要等待 EN 位置 0。应将先前的数据块 DMA 传输中在状态寄存器（DMA_LISR 和 DMA_HISR）中置 1 的所有数据流专用的位置 0，然后才可重新使能数据流。
2. 在 DMA_SxPAR 寄存器中设置外设端口寄存器地址。外设事件发生后，数据会从此地址移动到外设端口或从外设端口移动到此地址。
3. 在 DMA_SxMA0R 寄存器（在双缓冲区模式的情况下还有 DMA_SxMA1R 寄存器）中设置存储器地址。外设事件发生后，将从此存储器读取数据或将数据写入此存储器。
4. 在 DMA_SxNDTR 寄存器中配置要传输的数据项的总数。每出现一次外设事件或每出现一个节拍的突发传输，该值都会递减。
5. 使用 DMA_SxCR 寄存器中的 CHSEL[2:0] 选择 DMA 通道（请求）。
6. 如果外设用作流控制器而且支持此功能，请将 DMA_SxCR 寄存器中的 PFCTRL 位置 1。
7. 使用 DMA_SxCR 寄存器中的 PL[1:0] 位配置数据流优先级。
8. 配置 FIFO 的使用情况（使能或禁止，发送和接收阈值）。

9. 配置数据传输方向、外设和存储器增量 / 固定模式、单独或突发事务、外设和存储器数据宽度、循环模式、双缓冲区模式和传输完成一半和/或全部完成，和/或 DMA_SxCR 寄存器中错误的中断。
10. 通过将 DMA_SxCR 寄存器中的 EN 位置 1 激活数据流。

一旦使能了流，即可响应连接到数据流的外设发出的任何 DMA 请求。

一旦在 AHB 目标端口上传输了一半数据，传输一半标志 (HTIF) 便会置 1，如果传输一半中断使能位 (HTIE) 置 1，还会生成中断。传输结束时，传输完成标志 (TCIF) 便会置 1，如果传输完成中断使能位 (TCIE) 置 1，还会生成中断。

警告： **要关闭连接到 DMA 数据流请求的外设，必须首先关闭外设连接的 DMA 数据流，然后等待 EN 位 = 0。只有这样才能安全地禁止外设。**

9.3.18 错误管理

DMA 控制器可以检测到以下错误：

- **传输错误**：当发生下列情况时，传输错误中断标志 (TEIFx) 将置 1：
  - DMA 读或写访问期间发生总线错误
  - 软件请求在双缓冲区模式下写访问存储器地址寄存器，但是，已使能数据流，并且当前目标存储器是受写入存储器地址寄存器操作影响的存储器（请参见*第 9.3.9 节：双缓冲区模式*）
- **FIFO 错误**：如果发生下列情况，FIFO 错误中断标志 (FEIFx) 将置 1：
  - 检测到 FIFO 下溢情况
  - 检测到 FIFO 上溢情况（在存储器到存储器模式下由 DMA 内部管理请求和传输，所以在此模式下不检测溢出情况）
  - 当 FIFO 阈值级别与存储器突发大小不兼容时使能流（请参见*表 41：FIFO 阈值配置*）
- **直接模式错误**：只有当在直接模式下工作并且已将 DMA_SxCR 寄存器中的 MINC 位清零时，才能在外设到存储器模式下将直接模式错误中断标志 (DMEIFx) 置 1。当在先前数据未完全传输到存储器（因为存储器总线未得到授权）的情况下发生 DMA 请求时，该标志将置 1。在这种情况下，该标志指示有两个数据项相继传输到相同的目标地址，如果目标不能管理这种情况，会发生问题。

在直接模式下，如果出现下列条件，FIFO 错误标志也会置 1：

- 在外设到存储器模式下，如果未对存储器总线授权支持多个外设请求，FIFO 可能饱和（上溢）
- 在存储器到外设模式下，如果在外设请求发生前存储器总线未得到授权，则可能发生下溢的情况

如果由于突发大小与 FIFO 阈值级别之间的不兼容而引起 TEIFx 或 FEIFx 标志置 1，则硬件自动将相应数据流配置寄存器 (DMA_SxCR) 中的 EN 位清零，从而禁止错误的数据流。

如果由于上溢或下溢情况引起 DMEIFx 或 FEIFx 标志置 1，则不会自动禁止错误的数据流，而是由软件决定是否通过重置 DMA_SxCR 寄存器中的 EN 位禁止数据流。这是因为当发生这种错误时没有数据丢失。

当 DMA_LISR 或 DMA_HISR 寄存器中数据流的错误中断标志（TEIF、FEIF、DMEIF）置 1 时，如果 DMA_SxCR 或 DMA_SxFCR 寄存器中相应的中断使能位（TEIE、FEIE、DMIE）也置 1，则会生成一个中断。

注意： *当 FIFO 上溢或下溢的情况发生时，因为直到上溢或下溢的情况被清除，数据流才会确认外设请求，所以数据不会丢失。如果此确认过程花费过多时间，则外设本身会检测到其内部缓冲器上溢或下溢的情况，数据可能丢失。*

9.4 DMA 中断

对于每个 DMA 数据流，可在发生以下事件时产生中断：

- 达到半传输
- 传输完成
- 传输错误
- FIFO 错误（上溢、下溢或 FIFO 级别错误）
- 直接模式错误

可以使用单独的中断使能位以实现灵活性，如*表 43* 所示。

表 43. DMA 中断请求

中断事件	事件标志	使能控制位
半传输	HTIF	HTIE
传输完成	TCIF	TCIE
传输错误	TEIF	TEIE
FIFO 上溢/下溢	FEIF	FEIE
直接模式错误	DMEIF	DMEIE

注意： *在将使能控制位置‘1’前，应将相应的事件标志清零，否则会立即产生中断。*

9.5 DMA 寄存器

DMA 寄存器可按字（32 位）进行访问。

9.5.1 DMA 低中断状态寄存器 (DMA_LISR)

DMA low interrupt status register

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
Reserved				TCIF3	HTIF3	TEIF3	DMEIF3	Reserved	FEIF3	TCIF2	HTIF2	TEIF2	DMEIF2	Reserved	FEIF2
r	r	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
Reserved				TCIF1	HTIF1	TEIF1	DMEIF1	Reserved	FEIF1	TCIF0	HTIF0	TEIF0	DMEIF0	Reserved	FEIF0
r	r	r	r	r	r	r	r		r	r	r	r	r		r

位 31:28、15:12 保留，必须保持复位值。

位 27、21、11、5 **TCIFx**：数据流 x 传输完成中断标志 (Stream x transfer complete interrupt flag) (x = 3..0)
此位将由硬件置 1，由软件清零，软件只需将 1 写入 DMA_LIFCR 寄存器的相应位。
0：数据流 x 上无传输完成事件
1：数据流 x 上发生传输完成事件

位 26、20、10、4 **HTIFx**：数据流 x 半传输中断标志 (Stream x half transfer interrupt flag) (x=3..0)
此位将由硬件置 1，由软件清零，软件只需将 1 写入 DMA_LIFCR 寄存器的相应位。
0：数据流 x 上无半传输事件
1：数据流 x 上发生半传输事件

位 25、19、9、3 **TEIFx**：数据流 x 传输错误中断标志 (Stream x transfer error interrupt flag) (x=3..0)
此位将由硬件置 1，由软件清零，软件只需将 1 写入 DMA_LIFCR 寄存器的相应位。
0：数据流 x 上无传输错误
1：数据流 x 上发生传输错误

位 24、18、8、2 **DMEIFx**：数据流 x 直接模式错误中断标志 (Stream x direct mode error interrupt flag) (x=3..0)
此位将由硬件置 1，由软件清零，软件只需将 1 写入 DMA_LIFCR 寄存器的相应位。
0：数据流 x 上无直接模式错误
1：数据流 x 上发生直接模式错误

位 23、17、7、1 保留，必须保持复位值。

位 22、16、6、0 **FEIFx**：数据流 x FIFO 错误中断标志 (Stream x FIFO error interrupt flag) (x=3..0)
此位将由硬件置 1，由软件清零，软件只需将 1 写入 DMA_LIFCR 寄存器的相应位。
0：数据流 x 上无 FIFO 错误事件
1：数据流 x 上发生 FIFO 错误事件

9.5.2 DMA 高中断状态寄存器 (DMA_HISR)

DMA high interrupt status register

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
Reserved				TCIF7	HTIF7	TEIF7	DMEIF7	Reserved	FEIF7	TCIF6	HTIF6	TEIF6	DMEIF6	Reserved	FEIF6
				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
Reserved				TCIF5	HTIF5	TEIF5	DMEIF5	Reserved	FEIF5	TCIF4	HTIF4	TEIF4	DMEIF4	Reserved	FEIF4
				r	r	r	r		r	r	r	r	r		r

位 31:28、15:12 保留，必须保持复位值。

位 27、21、11、5 **TCIFx**：数据流 x 传输完成中断标志 (Stream x transfer complete interrupt flag) (x=7..4)
此位将由硬件置 1，由软件清零，软件只需将 1 写入 DMA_HIFCR 寄存器的相应位。
0：数据流 x 上无传输完成事件
1：数据流 x 上发生传输完成事件

位 26、20、10、4 **HTIFx**：数据流 x 半传输中断标志 (Stream x half transfer interrupt flag) (x=7..4)
此位将由硬件置 1，由软件清零，软件只需将 1 写入 DMA_HIFCR 寄存器的相应位。
0：数据流 x 上无半传输事件
1：数据流 x 上发生半传输事件

位 25、19、9、3 **TEIFx**：数据流 x 传输错误中断标志 (Stream x transfer error interrupt flag) (x=7..4)
此位将由硬件置 1，由软件清零，软件只需将 1 写入 DMA_HIFCR 寄存器的相应位。
0：数据流 x 上无传输错误
1：数据流 x 上发生传输错误

位 24、18、8、2 **DMEIFx**：数据流 x 直接模式错误中断标志 (Stream x direct mode error interrupt flag) (x=7..4)
此位将由硬件置 1，由软件清零，软件只需将 1 写入 DMA_HIFCR 寄存器的相应位。
0：数据流 x 上无直接模式错误
1：数据流 x 上发生直接模式错误

位 23、17、7、1 保留，必须保持复位值。

位 22、16、6、0 **FEIFx**：数据流 x FIFO 错误中断标志 (Stream x FIFO error interrupt flag) (x=7..4)
此位将由硬件置 1，由软件清零，软件只需将 1 写入 DMA_HIFCR 寄存器的相应位。
0：数据流 x 上无 FIFO 错误事件
1：数据流 x 上发生 FIFO 错误事件

9.5.3 DMA 低中断标志清零寄存器 (DMA_LIFCR)

DMA low interrupt flag clear register

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
Reserved				CTCIF3	CHTIF3	CTEIF3	CDMEIF3	Reserved	CFEIF3	CTCIF2	CHTIF2	CTEIF2	CDMEIF2	Reserved	CFEIF2
				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
Reserved				CTCIF1	CHTIF1	CTEIF1	CDMEIF1	Reserved	CFEIF1	CTCIF0	CHTIF0	CTEIF0	CDMEIF0	Reserved	CFEIF0
				w	w	w	w		w	w	w	w	w		w

位 31:28、15:12 保留，必须保持复位值。

位 27、21、11、5 **CTCIFx**：数据流 x 传输完成中断标志清零 (Stream x clear transfer complete interrupt flag) (x = 3..0)
将 1 写入此位时，DMA_LISR 寄存器中相应的 TCIFx 标志将清零

位 26、20、10、4 **CHTIFx**：数据流 x 半传输中断标志清零 (Stream x clear half transfer interrupt flag) (x = 3..0)
将 1 写入此位时，DMA_LISR 寄存器中相应的 HTIFx 标志将清零

位 25、19、9、3 **CTEIFx**：数据流 x 传输错误中断标志清零 (Stream x clear transfer error interrupt flag) (x = 3..0)
将 1 写入此位时，DMA_LISR 寄存器中相应的 TEIFx 标志将清零

位 24、18、8、2 **CDMEIFx**：数据流 x 直接模式错误中断标志清零 (Stream x clear direct mode error interrupt flag) (x = 3..0)
将 1 写入此位时，DMA_LISR 寄存器中相应的 DMEIFx 标志将清零

位 23、17、7、1 保留，必须保持复位值。

位 22、16、6、0 **CFEIFx**：数据流 x FIFO 错误中断标志清零 (Stream x clear FIFO error interrupt flag) (x = 3..0)
将 1 写入此位时，DMA_LISR 寄存器中相应的 CFEIFx 标志将清零

9.5.4 DMA 高中断标志清零寄存器 (DMA_HIFCR)

DMA high interrupt flag clear register

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
Reserved				CTCIF7	CHTIF7	CTEIF7	CDMEIF7	Reserved	CFEIF7	CTCIF6	CHTIF6	CTEIF6	CDMEIF6	Reserved	CFEIF6
				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
Reserved				CTCIF5	CHTIF5	CTEIF5	CDMEIF5	Reserved	CFEIF5	CTCIF4	CHTIF4	CTEIF4	CDMEIF4	Reserved	CFEIF4
				w	w	w	w		w	w	w	w	w		w

位 31:28、15:12 保留，必须保持复位值。

位 27、21、11、5 **CTCIFx**：数据流 x 传输完成中断标志清零 (Stream x clear transfer complete interrupt flag) (x = 7.0.4)

将 1 写入此位时，DMA_HISR 寄存器中相应的 TCIFx 标志将清零

位 26、20、10、4 **CHTIFx**：数据流 x 半传输中断标志清零 (Stream x clear half transfer interrupt flag) (x = 7.0.4)

将 1 写入此位时，DMA_HISR 寄存器中相应的 HTIFx 标志将清零

位 25、19、9、3 **CTEIFx**：数据流 x 传输错误中断标志清零 (Stream x clear transfer error interrupt flag) (x = 7.0.4)

将 1 写入此位时，DMA_HISR 寄存器中相应的 TEIFx 标志将清零

位 24、18、8、2 **CDMEIFx**：数据流 x 直接模式错误中断标志清零 (Stream x clear direct mode error interrupt flag) (x = 7.0.4)

将 1 写入此位时，DMA_HISR 寄存器中相应的 DMEIFx 标志将清零

位 23、17、7、1 保留，必须保持复位值。

位 22、16、6、0 **CFEIFx**：数据流 x FIFO 错误中断标志清零 (Stream x clear FIFO error interrupt flag) (x = 7.0.4)

将 1 写入此位时，DMA_HISR 寄存器中相应的 CFEIFx 标志将清零

9.5.5 DMA 数据流 x 配置寄存器 (DMA_SxCR) (x = 0..7)

DMA stream x configuration register

此寄存器用于配置相关数据流。

偏移地址：0x10 + 0x18 × *数据流编号*

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved				CHSEL[3:0]			MBURST [1:0]		PBURST[1:0]		Reserved	CT	DBM or reserved	PL[1:0]	
				rw	rw	rw	rw	rw	rw	rw		rw	rw or r	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PINCOS	MSIZE[1:0]		PSIZE[1:0]		MINC	PINC	CIRC	DIR[1:0]		PFCTRL	TCIE	HTIE	TEIE	DMEIE	EN
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:28 保留，必须保持复位值。

位 27:25 **CHSEL[2:0]**：通道选择 (Channel selection)
这些位将由软件置 1 和清零。
000：选择通道 0
001：选择通道 1
010：选择通道 2
011：选择通道 3
100：选择通道 4
101：选择通道 5
110：选择通道 6
111：选择通道 7
这些位受到保护，只有 EN 为“0”时才可以写入

位 24:23 **MBURST**：存储器突发传输配置 (Memory burst transfer configuration)
这些位将由软件置 1 和清零。
00：单次传输
01：INCR4（4 个节拍的增量突发传输）
10：INCR8（8 个节拍的增量突发传输）
11：INCR16（16 个节拍的增量突发传输）
这些位受到保护，只有 EN 为“0”时才可以写入
在直接模式中，当位 EN = “1”时，这些位由硬件强制置为 0x0。

位 22:21 **PBURST[1:0]**：外设突发传输配置 (Peripheral burst transfer configuration)
这些位将由软件置 1 和清零。
00：单次传输
01：INCR4（4 个节拍的增量突发传输）
10：INCR8（8 个节拍的增量突发传输）
11：INCR16（16 个节拍的增量突发传输）
这些位受到保护，只有 EN 为“0”时才可以写入
在直接模式下，这些位由硬件强制置为 0x0。

位 20 保留，必须保持复位值。

位 19 **CT**：当前目标（仅在双缓冲区模式下）(Current target (only in double buffer mode))
此位由硬件置 1 和清零，也可由软件写入。
0：当前目标存储器为存储器 0（使用 DMA_SxM0AR 指针寻址）
1：当前目标存储器为存储器 1（使用 DMA_SxM1AR 指针寻址）
只有 EN 为“0”时，此位才可以写入，以指示第一次传输的目标存储区。在使能数据流后，此位相当于一个状态标志，用于指示作为当前目标的存储区。

位 18 **DBM**：双缓冲区模式 (Double buffer mode)
此位由软件置 1 和清零。
0：传输结束时不切换缓冲区
1：DMA 传输结束时切换目标存储区
此位受到保护，只有 EN 为“0”时才可以写入。

位 17:16 **PL[1:0]**：优先级 (Priority level)
这些位将由软件置 1 和清零。
00：低
01：中
10：高
11：非常高
这些位受到保护，只有 EN 为“0”时才可以写入。

位 15 **PINCOS**：外设增量偏移量 (Peripheral increment offset size)
此位由软件置 1 和清零
0：用于计算外设地址的偏移量与 PSIZE 相关
1：用于计算外设地址的偏移量固定为 4（32 位对齐）。
如果位 PINC =“0”，则此位没有意义。
此位受到保护，只有 EN 为“0”时才可以写入。
如果选择直接模式或者 PBURST 不等于“00”，则当使能数据流（位 EN =“1”）时，此位由硬件强制置为低电平。

位 14:13 **MSIZE[1:0]**：存储器数据大小 (Memory data size)
这些位将由软件置 1 和清零。
00：字节（8 位）
01：半字（16 位）
10：字（32 位）
11：保留
这些位受到保护，只有 EN 为“0”时才可以写入。
在直接模式下，当位 EN =“1”时，MSIZE 位由硬件强制置为与 PSIZE 相同的值。

位 12:11 **PSIZE[1:0]**：外设数据大小 (Peripheral data size)
这些位将由软件置 1 和清零。
00：字节（8 位）
01：半字（16 位）
10：字（32 位）
11：保留
这些位受到保护，只有 EN 为“0”时才可以写入

位 10 **MINC**：存储器递增模式 (Memory increment mode)
此位由软件置 1 和清零。
0：存储器地址指针固定
1：每次数据传输后，存储器地址指针递增（增量为 MSIZE 值）
此位受到保护，只有 EN 为“0”时才可以写入。

位 9 **PINC**：外设递增模式 (Peripheral increment mode)
此位由软件置 1 和清零。
0：外设地址指针固定
1：每次数据传输后，外设地址指针递增（增量为 PSIZE 值）
此位受到保护，只有 EN 为“0”时才可以写入。

位 8 **CIRC**：循环模式 (Circular mode)
此位由软件置 1 和清零，并可由硬件清零。
0：禁止循环模式
1：使能循环模式
如果外设为流控制器（位 PFCTRL=1）且使能数据流（位 EN=1），此位由硬件自动强制清零。
如果 DBM 位置 1，当使能数据流（位 EN =“1”）时，此位由硬件自动强制置 1。

位 7:6 **DIR[1:0]**：数据传输方向 (Data transfer direction)
这些位将由软件置 1 和清零。
00：外设到存储器
01：存储器到外设
10：存储器到存储器
11：保留
这些位受到保护，只有 EN 为“0”时才可以写入。

位 5 **PFCTRL**：外设流控制器 (Peripheral flow controller)

此位由软件置 1 和清零。

0：DMA 是流控制器

1：外设是流控制器

此位受到保护，只有 EN 为“0”时才可以写入。

选择存储器到存储器模式（位 DIR[1:0]=10）后，此位由硬件自动强制清零。

位 4 **TCIE**：传输完成中断使能 (Transfer complete interrupt enable)

此位由软件置 1 和清零。

0：禁止 TC 中断

1：使能 TC 中断

位 3 **HTIE**：半传输中断使能 (Half transfer interrupt enable)

此位由软件置 1 和清零。

0：禁止 HT 中断

1：使能 HT 中断

位 2 **TEIE**：传输错误中断使能 (Transfer error interrupt enable)

此位由软件置 1 和清零。

0：禁止 TE 中断

1：使能 TE 中断

位 1 **DMEIE**：直接模式错误中断使能 (Direct mode error interrupt enable)

此位由软件置 1 和清零。

0：禁止 DME 中断

1：使能 DME 中断

位 0 **EN**：数据流使能/读作低电平时数据流就绪标志 (Stream enable / flag stream ready when read low)

此位由软件置 1 和清零。

0：禁止数据流

1：使能数据流

以下情况下，此位可由硬件清零：

– DMA 传输结束时（准备好配置数据流）

– AHB 主总线出现传输错误时

– 存储器 AHB 端口上的 FIFO 阈值与突发大小不兼容时

此位读作 0 时，软件可以对配置和 FIFO 位寄存器编程。EN 位读作 1 时，禁止向这些寄存器执行写操作。

注意：将 EN 位置“1”以启动新传输之前，DMA_LISR 或 DMA_HISR 寄存器中与数据流相对应的事件标志必须清零。

9.5.6 DMA 数据流 x 数据项数寄存器 (DMA_SxNDTR) (x = 0..7)

DMA stream x number of data register

偏移地址：0x14 + 0x18 × *数据流编号*

复位值：0x0000 0000

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

位 31:16 保留，必须保持复位值。

位 15:0 **NDT[15:0]**：要传输的数据项数目 (Number of data items to transfer)

要传输的数据项数目（0 到 65535）。只有在禁止数据流时，才能向此寄存器执行写操作。使能数据流后，此寄存器为只读，用于指示要传输的剩余数据项数。每次 DMA 传输后，此寄存器将递减。

传输完成后，此寄存器保持为零（数据流处于正常模式时），或者在以下情况下自动以先前编程的值重载：

– 以循环模式配置数据流时。

– 通过将 EN 位置“1”来重新使能数据流时

如果该寄存器的值为零，则即使使能数据流，也无法完成任何事务。

9.5.7 DMA 数据流 x 外设地址寄存器 (DMA_SxPAR) (x = 0..7)

DMA stream x peripheral address register

偏移地址：0x18 + 0x18 × *数据流编号*

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PAR[31:16]															
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
PAR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:0 **PAR[31:0]**：外设地址 (Peripheral address)

读/写数据的外设数据寄存器的基址。

这些位受到写保护，只有 DMA_SxCR 寄存器中的 EN 为“0”时才可以写入。

9.5.8 DMA 数据流 x 存储器 0 地址寄存器 (DMA_SxM0AR) (x = 0..7)

DMA stream x memory 0 address register

偏移地址：0x1C + 0x18 × *数据流编号*

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
M0A[31:16]															
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
M0A[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:0 **M0A[31:0]**：存储器 0 地址 (Memory 0 address)

读/写数据的存储区 0 的基址。

这些位受到写保护，只有在以下情况下才可以写入：

– 禁止数据流（DMA_SxCR 寄存器中的位 EN=“0”）或

– 使能数据流（DMA_SxCR 寄存器中的 EN=“1”）并且 DMA_SxCR 寄存器中的位 CT =“1”（在双缓冲区模式下）。

9.5.9 DMA 数据流 x 存储器 1 地址寄存器 (DMA_SxM1AR) (x = 0..7)

DMA stream x memory 1 address register

偏移地址：0x20 + 0x18 × *数据流编号*

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
M1A[31:16]															
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
M1A[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:0 **M1A[31:0]**：存储器 1 地址（用于双缓冲区模式）(Memory 1 address (used in case of Double buffer mode))

读/写数据的存储区 1 的基址。

此寄存器仅用于双缓冲区模式。

这些位受到写保护，只有在以下情况下才可以写入：

– 禁止数据流（DMA_SxCR 寄存器中的位 EN=“0”）或

– 使能数据流（DMA_SxCR 寄存器中的 EN=“1”）并且 DMA_SxCR 寄存器中的位 CT =“0”。

9.5.10 DMA 数据流 x FIFO 控制寄存器 (DMA_SxFCR) (x = 0..7)

DMA stream x FIFO control register

偏移地址：0x24 + 0x24 × *数据流编号*

复位值：0x0000 0021

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								FEIE	Reserved	FS[2:0]			DMDIS	FTH[1:0]	
								rw		r	r	r	rw	rw	rw

位 31:8 保留，必须保持复位值。

位 7 **FEIE**：FIFO 错误中断使能 (FIFO error interrupt enable)

此位由软件置 1 和清零。

0：禁止 FE 中断

1：使能 FE 中断

位 6 保留，必须保持复位值。

位 5:3 **FS[2:0]**：FIFO 状态 (FIFO status)

这些位为只读。

000：0 < fifo_level < 1/4
001：1/4 ≤ fifo_level < 1/2
010：1/2 ≤ fifo_level < 3/4
011：3/4 ≤ fifo_level < 满
100：FIFO 为空
101：FIFO 已满
其它：无意义
在直接模式（DMDIS 位为零）下，这些位无意义。

位 2 **DMDIS**：直接模式禁止 (Direct mode disable)

此位由软件置 1 和清零。它可由硬件置 1。
0：使能直接模式
1：禁止直接模式
此位受到保护，只有 EN 为“0”时才可以写入。
如果选择存储器到存储器模式（DMA_SxCR 中的 DIR 位为“10”），并且 DMA_SxCR 寄存器中的 EN 位为“1”，则此位由硬件置 1，因为在存储器到存储器配置不能使用直接模式。

位 1:0 **FTH[1:0]**：FIFO 阈值选择 (FIFO threshold selection)

这些位将由软件置 1 和清零。
00：FIFO 容量的 1/4
01：FIFO 容量的 1/2
10：FIFO 容量的 3/4
11：FIFO 完整容量
在直接模式（DMDIS 值为零）下，不使用这些位。
这些位受到保护，只有 EN 为“1”时才可以写入。

9.5.11 DMA 寄存器映射

表 44 汇总了 DMA 寄存器。

表 44. DMA 寄存器映射和复位值

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0000	DMA_LISR	Reserved				TCIF3	HTIF3	TEIF3	DMEIF3	Reserved	FEIF3	TCIF2	HTIF2	TEIF2	DMEIF2	Reserved	FEIF2	Reserved				TCIF1	HTIF1	TEIF1	DMEIF1	Reserved	FEIF1	TCIF0	HTIF0	TEIF0	DMEIF0	Reserved	FEIF0
	Reset value					0	0	0	0		0	0	0	0	0		0					0	0	0	0		0	0	0	0	0		0
0x0004	DMA_HISR	Reserved				TCIF7	HTIF7	TEIF7	DMEIF7	Reserved	FEIF7	TCIF6	HTIF6	TEIF6	DMEIF6	Reserved	FEIF6	Reserved				TCIF5	HTIF5	TEIF5	DMEIF5	Reserved	FEIF5	TCIF4	HTIF4	TEIF4	DMEIF4	Reserved	FEIF4
	Reset value					0	0	0	0		0	0	0	0	0		0					0	0	0	0		0	0	0	0	0		0
0x0008	DMA_LIFCR	Reserved				CTCIF3	CHTIF3	TEIF3	CDMEIF3	Reserved	CFEIF3	CTCIF2	CHTIF2	CTEIF2	CDMEIF2	Reserved	CFEIF2	Reserved				CTCIF1	CHTIF1	CTEIF1	CDMEIF1	Reserved	CFEIF1	CTCIF0	CHTIF0	CTEIF0	CDMEIF0	Reserved	CFEIF0
	Reset value					0	0	0	0		0	0	0	0	0		0					0	0	0	0		0	0	0	0	0		0
0x000C	DMA_HIFCR	Reserved				CTCIF7	CHTIF7	CTEIF7	CDMEIF7	Reserved	CFEIF7	CTCIF6	CHTIF6	CTEIF6	CDMEIF6	Reserved	CFEIF6	Reserved				CTCIF5	CHTIF5	CTEIF5	CDMEIF5	Reserved	CFEIF5	CTCIF4	CHTIF4	CTEIF4	CDMEIF4	Reserved	CFEIF4
	Reset value					0	0	0	0		0	0	0	0	0		0					0	0	0	0		0	0	0	0	0		0

表 44. DMA 寄存器映射和复位值 （续）

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0010	DMA_S0CR	Reserved				CHSEL[2:0]			MBURST[1:0]		PBURST[1:0]		Reserved	CT	DBM	PL[1:0]		PINCOS	MSIZE[1:0]		PSIZE[1:0]		MINC	PINC	CIRC	DIR[1:0]		PFCTRL	TCIE	HTIE	TEIE	DMEIE	EN
	Reset value					0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0014	DMA_S0NDTR	Reserved																NDT[15:.]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0018	DMA_S0PAR	PA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x001C	DMA_S0M0AR	M0A[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0020	DMA_S0M1AR	M1A[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0024	DMA_S0FCR	Reserved																								FEIE	Reserved	FS[2:0]			DMDIS	FTH [1:0]	
	Reset value																									0		1	0	0	0	0	1
0x0028	DMA_S1CR	Reserved				CHSEL [2:0]			MBURST[1:]		PBURST[1:0]		ACK	CT	DBM	PL[1:0]		PINCOS	MSIZE[1:0]		PSIZE[1:0]		MINC	PINC	CIRC	DIR[1:0]		PFCTRL	TCIE	HTIE	TEIE	DMEIE	EN
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x002C	DMA_S1NDTR	Reserved																NDT[15:.]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0030	DMA_S1PAR	PA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0034	DMA_S1M0AR	M0A[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0038	DMA_S1M1AR	M1A[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x003C	DMA_S1FCR	Reserved																								FEIE	Reserved	FS[2:0]			DMDIS	FTH [1:0]	
	Reset value																									0		1	0	0	0	0	1
0x0040	DMA_S2CR	Reserved				CHSEL [2:0]			MBURST[1:0]		PBURST[1:0]		ACK	CT	DBM	PL[1:0]		PINCOS	MSIZE[1:0]		PSIZE[1:0]		MINC	PINC	CIRC	DIR [1:0]		PFCTRL	TCIE	HTIE	TEIE	DMEIE	EN
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0044	DMA_S2NDTR	Reserved																NDT[15:.]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0048	DMA_S2PAR	PA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

表 44. DMA 寄存器映射和复位值 （续）

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x004C	DMA_S2M0AR	M0A[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0050	DMA_S2M1AR	M1A[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0054	DMA_S2FCR	Reserved																								FEIE	Reserved	FS[2:0]			DMDIS	FTH [1:0]	
	Reset value																									0		1	0	0	0	0	1
0x0058	DMA_S3CR	Reserved				CHSEL[2:0]			MBURST[1:0]		PBURST[1:0]		ACK	CT	DBM	PL[1:0]		PINCOS	MSIZE[1:0]		PSIZE[1:0]		MINC	PINC	CIRC	DIR[1:0]		PFCTRL	TCIE	HTIE	TEIE	DMEIE	EN
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x005C	DMA_S3NDTR	Reserved																NDT[15:..]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0060	DMA_S3PAR	PA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0064	DMA_S3M0AR	M0A[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0068	DMA_S3M1AR	M1A[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x006C	DMA_S3FCR	Reserved																								FEIE	Reserved	FS[2:0]			DMDIS	FTH [1:0]	
	Reset value																									0		1	0	0	0	0	1
0x0070	DMA_S4CR	Reserved				CHSEL[2:0]			MBURST[1:0]		PBURST[1:0]		ACK	CT	DBM	PL[1:0]		PINCOS	MSIZE[1:0]		PSIZE[1:0]		MINC	PINC	CIRC	DIR [1:0]		PFCTRL	TCIE	HTIE	TEIE	DMEIE	EN
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0074	DMA_S4NDTR	Reserved																NDT[15:..]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0078	DMA_S4PAR	PA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x007C	DMA_S4M0AR	M0A[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0080	DMA_S4M1AR	M1A[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0084	DMA_S4FCR	Reserved																								FEIE	Reserved	FS[2:0]			DMDIS	FTH [1:0]	
	Reset value																									0		1	0	0	0	0	1
0x0088	DMA_S5CR	Reserved				CHSEL[2:0]			MBURST[1:0]		PBURST[1:0]		ACK	CT	DBM	PL[1:0]		PINCOS	MSIZE[1:0]		PSIZE[1:0]		MINC	PINC	CIRC	DIR[1:0]		PFCTRL	TCIE	HTIE	TEIE	DMEIE	EN
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

表 44. DMA 寄存器映射和复位值 （续）

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x008C	DMA_S5NDTR	Reserved																NDT[15:.]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0090	DMA_S5PAR	PA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0094	DMA_S5M0AR	M0A[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0098	DMA_S5M1AR	M1A[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x009C	DMA_S5FCR	Reserved																								FEIE	Reserved	FS[2:0]			DMDIS	FTH [1:0]	
	Reset value																									0		1	0	0	0	0	1
0x00A0	DMA_S6CR	Reserved				CHSEL[2:0]			MBURST[1:0]		PBURST[1:0]		ACK	CT	DBM	PL[1:0]		PINCOS	MSIZE[1:0]		PSIZE[1:0]		MINC	PINC	CIRC	DIR[1:0]		PFCTRL	TCIE	HTIE	TEIE	DMEIE	EN
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00A4	DMA_S6NDTR	Reserved																NDT[15:.]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00A8	DMA_S6PAR	PA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00AC	DMA_S6M0AR	M0A[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00B0	DMA_S6M1AR	M1A[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00B4	DMA_S6FCR	Reserved																								FEIE	Reserved	FS[2:0]			DMDIS	FTH [1:0]	
	Reset value																									0		1	0	0	0	0	1
0x00B8	DMA_S7CR	Reserved				CHSEL[2:0]			MBURST[1:0]		PBURST[1:0]		ACK	CT	DBM	PL[1:0]		PINCOS	MSIZE[1:0]		PSIZE[1:0]		MINC	PINC	CIRC	DIR[1:0]		PFCTRL	TCIE	HTIE	TEIE	DMEIE	EN
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00BC	DMA_S7NDTR	Reserved																NDT[15:.]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00C0	DMA_S7PAR	PA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00C4	DMA_S7M0AR	M0A[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00C8	DMA_S7M1AR	M1A[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00CC	DMA_S7FCR	Reserved																								FEIE	Reserved	FS[2:0]			DMDIS	FTH [1:0]	
	Reset value																									0		1	0	0	0	0	1

有关寄存器边界地址的信息，请参见*第 52 页的表 2*。

10 中断和事件

除非特别说明，否则本部分适用于整个 STM32F4xx 系列。

10.1 嵌套向量中断控制器 (NVIC)

10.1.1 NVIC 特性

嵌套向量中断控制器 NVIC 包含以下特性：

- STM32F405xx/07xx 和 STM32F415xx/17xx 具有 82 个可屏蔽中断通道，STM32F42xxx 和 STM32F43xxx 具有多达 86 个可屏蔽中断通道（不包括 Cortex™-M4F 的 16 根中断线）
- 16 个可编程优先级（使用了 4 位中断优先级）
- 低延迟异常和中断处理
- 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) 和处理器内核接口紧密配合，可以实现低延迟的中断处理和晚到中断的高效处理。

包括内核异常在内的所有中断均通过 NVIC 进行管理。更多关于异常和 NVIC 编程的说明，请参考《ARM Cortex™-M4F 技术参考手册》中的*第 5 章：异常和第 8 章：嵌套向量中断控制器*。

10.1.2 SysTick 校准值寄存器

SysTick 校准值设置为 18750。当 SysTick 时钟设置为 18.75 MHz（HCLK/8，HCLK 设为 150 MHz），会产生 1 ms 时间基准。

10.1.3 中断和异常向量

请参见*表 45* 和*表 46*，了解 STM32F405xx/07xx 和 STM32F415xx/17xx 和 STM32F42xxx 和 STM32F43xxx 器件的向量表。

10.2 外部中断/事件控制器 (EXTI)

外部中断/事件控制器包含多达 23 个用于产生事件/中断请求的边沿检测器。每根输入线都可单独进行配置，以选择类型（中断或事件）和相应的触发事件（上升沿触发、下降沿触发或边沿触发）。每根输入线还可单独屏蔽。挂起寄存器用于保持中断请求的状态线。

表 45. STM32F405xx/07xx 和 STM32F415xx/17xx 的向量表

位置	优先级	优先级类型	名称	说明	地址
	-	-	-	保留	0x0000 0000
	-3	固定	Reset	复位	0x0000 0004
	-2	固定	NMI	不可屏蔽中断。RCC 时钟安全系统 (CSS) 连接到 NMI 向量。	0x0000 0008
	-1	固定	HardFault	所有类型的错误	0x0000 000C
	0	可设置	MemManage	存储器管理	0x0000 0010
	1	可设置	BusFault	预取指失败，存储器访问失败	0x0000 0014
	2	可设置	UsageFault	未定义的指令或非法状态	0x0000 0018
	-	-	-	保留	0x0000 001C - 0x0000 002B
	3	可设置	SVCall	通过 SWI 指令调用的系统服务	0x0000 002C
	4	可设置	Debug Monitor	调试监控器	0x0000 0030
	-	-	-	保留	0x0000 0034
	5	可设置	PendSV	可挂起的系统服务	0x0000 0038
	6	可设置	SysTick	系统嘀嗒定时器	0x0000 003C
0	7	可设置	WWDG	窗口看门狗中断	0x0000 0040
1	8	可设置	PVD	连接到 EXTI 线的可编程电压检测 (PVD) 中断	0x0000 0044
2	9	可设置	TAMP_STAMP	连接到 EXTI 线的入侵和时间戳中断	0x0000 0048
3	10	可设置	RTC_WKUP	连接到 EXTI 线的 RTC 唤醒中断	0x0000 004C
4	11	可设置	FLASH	Flash 全局中断	0x0000 0050
5	12	可设置	RCC	RCC 全局中断	0x0000 0054
6	13	可设置	EXTI0	EXTI 线 0 中断	0x0000 0058
7	14	可设置	EXTI1	EXTI 线 1 中断	0x0000 005C
8	15	可设置	EXTI2	EXTI 线 2 中断	0x0000 0060
9	16	可设置	EXTI3	EXTI 线 3 中断	0x0000 0064
10	17	可设置	EXTI4	EXTI 线 4 中断	0x0000 0068
11	18	可设置	DMA1_Stream0	DMA1 流 0 全局中断	0x0000 006C
12	19	可设置	DMA1_Stream1	DMA1 流 1 全局中断	0x0000 0070
13	20	可设置	DMA1_Stream2	DMA1 流 2 全局中断	0x0000 0074
14	21	可设置	DMA1_Stream3	DMA1 流 3 全局中断	0x0000 0078
15	22	可设置	DMA1_Stream4	DMA1 流 4 全局中断	0x0000 007C
16	23	可设置	DMA1_Stream5	DMA1 流 5 全局中断	0x0000 0080
17	24	可设置	DMA1_Stream6	DMA1 流 6 全局中断	0x0000 0084
18	25	可设置	ADC	ADC1、ADC2 和 ADC3 全局中断	0x0000 0088

表 45. STM32F405xx/07xx 和 STM32F415xx/17xx 的向量表 （续）

位置	优先级	优先级类型	名称	说明	地址
19	26	可设置	CAN1_TX	CAN1 TX 中断	0x0000 008C
20	27	可设置	CAN1_RX0	CAN1 RX0 中断	0x0000 0090
21	28	可设置	CAN1_RX1	CAN1 RX1 中断	0x0000 0094
22	29	可设置	CAN1_SCE	CAN1 SCE 中断	0x0000 0098
23	30	可设置	EXTI9_5	EXTI 线 [9:5] 中断	0x0000 009C
24	31	可设置	TIM1_BRK_TIM9	TIM1 刹车中断和 TIM9 全局中断	0x0000 00A0
25	32	可设置	TIM1_UP_TIM10	TIM1 更新中断和 TIM10 全局中断	0x0000 00A4
26	33	可设置	TIM1_TRG_COM_TIM11	TIM1 触发和换相中断与 TIM11 全局中断	0x0000 00A8
27	34	可设置	TIM1_CC	TIM1 捕获比较中断	0x0000 00AC
28	35	可设置	TIM2	TIM2 全局中断	0x0000 00B0
29	36	可设置	TIM3	TIM3 全局中断	0x0000 00B4
30	37	可设置	TIM4	TIM4 全局中断	0x0000 00B8
31	38	可设置	I2C1_EV	I^2C1 事件中断	0x0000 00BC
32	39	可设置	I2C1_ER	I^2C1 错误中断	0x0000 00C0
33	40	可设置	I2C2_EV	I^2C2 事件中断	0x0000 00C4
34	41	可设置	I2C2_ER	I^2C2 错误中断	0x0000 00C8
35	42	可设置	SPI1	SPI1 全局中断	0x0000 00CC
36	43	可设置	SPI2	SPI2 全局中断	0x0000 00D0
37	44	可设置	USART1	USART1 全局中断	0x0000 00D4
38	45	可设置	USART2	USART2 全局中断	0x0000 00D8
39	46	可设置	USART3	USART3 全局中断	0x0000 00DC
40	47	可设置	EXTI15_10	EXTI 线 [15:10] 中断	0x0000 00E0
41	48	可设置	RTC_Alarm	连接到 EXTI 线的 RTC 闹钟（A 和 B）中断	0x0000 00E4
42	49	可设置	OTG_FS WKUP	连接到 EXTI 线的 USB On-The-Go FS 唤醒中断	0x0000 00E8
43	50	可设置	TIM8_BRK_TIM12	TIM8 刹车中断和 TIM12 全局中断	0x0000 00EC
44	51	可设置	TIM8_UP_TIM13	TIM8 更新中断和 TIM13 全局中断	0x0000 00F0
45	52	可设置	TIM8_TRG_COM_TIM14	TIM8 触发和换相中断与 TIM14 全局中断	0x0000 00F4
46	53	可设置	TIM8_CC	TIM8 捕捉比较中断	0x0000 00F8
47	54	可设置	DMA1_Stream7	DMA1 流 7 全局中断	0x0000 00FC
48	55	可设置	FSMC	FSMC 全局中断	0x0000 0100
49	56	可设置	SDIO	SDIO 全局中断	0x0000 0104

表 45. STM32F405xx/07xx 和 STM32F415xx/17xx 的向量表 （续）

位置	优先级	优先级类型	名称	说明	地址
50	57	可设置	TIM5	TIM5 全局中断	0x0000 0108
51	58	可设置	SPI3	SPI3 全局中断	0x0000 010C
52	59	可设置	UART4	UART4 全局中断	0x0000 0110
53	60	可设置	UART5	UART5 全局中断	0x0000 0114
54	61	可设置	TIM6_DAC	TIM6 全局中断， DAC1 和 DAC2 下溢错误中断	0x0000 0118
55	62	可设置	TIM7	TIM7 全局中断	0x0000 011C
56	63	可设置	DMA2_Stream0	DMA2 流 0 全局中断	0x0000 0120
57	64	可设置	DMA2_Stream1	DMA2 流 1 全局中断	0x0000 0124
58	65	可设置	DMA2_Stream2	DMA2 流 2 全局中断	0x0000 0128
59	66	可设置	DMA2_Stream3	DMA2 流 3 全局中断	0x0000 012C
60	67	可设置	DMA2_Stream4	DMA2 流 4 全局中断	0x0000 0130
61	68	可设置	ETH	以太网全局中断	0x0000 0134
62	69	可设置	ETH_WKUP	连接到 EXTI 线的以太网唤醒中断	0x0000 0138
63	70	可设置	CAN2_TX	CAN2 TX 中断	0x0000 013C
64	71	可设置	CAN2_RX0	CAN2 RX0 中断	0x0000 0140
65	72	可设置	CAN2_RX1	CAN2 RX1 中断	0x0000 0144
66	73	可设置	CAN2_SCE	CAN2 SCE 中断	0x0000 0148
67	74	可设置	OTG_FS	USB On The Go FS 全局中断	0x0000 014C
68	75	可设置	DMA2_Stream5	DMA2 流 5 全局中断	0x0000 0150
69	76	可设置	DMA2_Stream6	DMA2 流 6 全局中断	0x0000 0154
70	77	可设置	DMA2_Stream7	DMA2 流 7 全局中断	0x0000 0158
71	78	可设置	USART6	USART6 全局中断	0x0000 015C
72	79	可设置	I2C3_EV	I^2C3 事件中断	0x0000 0160
73	80	可设置	I2C3_ER	I^2C3 错误中断	0x0000 0164
74	81	可设置	OTG_HS_EP1_OUT	USB On The Go HS 端点 1 输出全局中断	0x0000 0168
75	82	可设置	OTG_HS_EP1_IN	USB On The Go HS 端点 1 输入全局中断	0x0000 016C
76	83	可设置	OTG_HS_WKUP	连接到 EXTI 线的 USB On The Go HS 唤醒中断	0x0000 0170
77	84	可设置	OTG_HS	USB On The Go HS 全局中断	0x0000 0174
78	85	可设置	DCMI	DCMI 全局中断	0x0000 0178
79	86	可设置	CRYP	CRYP 加密全局中断	0x0000 017C

表 45. STM32F405xx/07xx 和 STM32F415xx/17xx 的向量表 （续）

位置	优先级	优先级类型	名称	说明	地址
80	87	可设置	HASH_RNG	哈希和随机数发生器全局中断	0x0000 0180
81	88	可设置	FPU	FPU 全局中断	0x0000 0184

表 46. STM32F42xxx 和 STM32F43xxx 的向量表

位置	优先级	优先级类型	名称	说明	地址
	-	-	-	保留	0x0000 0000
	-3	固定	Reset	复位	0x0000 0004
	-2	固定	NMI	不可屏蔽中断，时钟安全系统	0x0000 0008
	-1	固定	HardFault	所有类型的错误	0x0000 000C
	0	固定	MemManage	MPU 不匹配	0x0000 0010
	1	可设置	BusFault	预取指失败，存储器访问失败	0x0000 0014
	2	可设置	UsageFault	未定义的指令或非法状态	0x0000 0018
	-	-	-	保留	0x0000 001C - 0x0000 002B
	3	可设置	SVCall	通过 SWI 指令调用的系统服务	0x0000 002C
	4	可设置	Debug Monitor	调试监控器	0x0000 0030
		-	-	保留	0x0000 0034
	5	可设置	PendSV	可挂起的系统服务	0x0000 0038
	6	可设置	Systick	系统嘀嗒定时器	0x0000 003C
0	7	可设置	WWDG	窗口看门狗中断	0x0000 0040
1	8	可设置	PVD	连接到 EXTI 线的可编程电压检测 (PVD) 中断	0x0000 0044
2	9	可设置	TAMP_STAMP	连接到 EXTI 线的入侵和时间戳中断	0x0000 0048
3	10	可设置	RTC_WKUP	连接到 EXTI 线的 RTC 唤醒中断	0x0000 004C
4	11	可设置	FLASH	Flash 全局中断	0x0000 0050
5	12	可设置	RCC	RCC 全局中断	0x0000 0054
6	13	可设置	EXTI0	EXTI 线 0 中断	0x0000 0058
7	14	可设置	EXTI1	EXTI 线 1 中断	0x0000 005C
8	15	可设置	EXTI2	EXTI 线 2 中断	0x0000 0060
9	16	可设置	EXTI3	EXTI 线 3 中断	0x0000 0064
10	17	可设置	EXTI4	EXTI 线 4 中断	0x0000 0068
11	18	可设置	DMA1_Stream0	DMA1 流 0 全局中断	0x0000 006C
12	19	可设置	DMA1_Stream1	DMA1 流 1 全局中断	0x0000 0070
13	20	可设置	DMA1_Stream2	DMA1 流 2 全局中断	0x0000 0074

表 46. STM32F42xxx 和 STM32F43xxx 的向量表 （续）

位置	优先级	优先级类型	名称	说明	地址
14	21	可设置	DMA1_Stream3	DMA1 流 3 全局中断	0x0000 0078
15	22	可设置	DMA1_Stream4	DMA1 流 4 全局中断	0x0000 007C
16	23	可设置	DMA1_Stream5	DMA1 流 5 全局中断	0x0000 0080
17	24	可设置	DMA1_Stream6	DMA1 流 6 全局中断	0x0000 0084
18	25	可设置	ADC	ADC1、ADC2 和 ADC3 全局中断	0x0000 0088
19	26	可设置	CAN1_TX	CAN1 TX 中断	0x0000 008C
20	27	可设置	CAN1_RX0	CAN1 RX0 中断	0x0000 0090
21	28	可设置	CAN1_RX1	CAN1 RX1 中断	0x0000 0094
22	29	可设置	CAN1_SCE	CAN1 SCE 中断	0x0000 0098
23	30	可设置	EXTI9_5	EXTI 线 [9:5] 中断	0x0000 009C
24	31	可设置	TIM1_BRK_TIM9	TIM1 刹车中断和 TIM9 全局中断	0x0000 00A0
25	32	可设置	TIM1_UP_TIM10	TIM1 更新中断和 TIM10 全局中断	0x0000 00A4
26	33	可设置	TIM1_TRG_COM_TIM11	TIM1 触发和换相中断与 TIM11 全局中断	0x0000 00A8
27	34	可设置	TIM1_CC	TIM1 捕获比较中断	0x0000 00AC
28	35	可设置	TIM2	TIM2 全局中断	0x0000 00B0
29	36	可设置	TIM3	TIM3 全局中断	0x0000 00B4
30	37	可设置	TIM4	TIM4 全局中断	0x0000 00B8
31	38	可设置	I2C1_EV	I^2C1 事件中断	0x0000 00BC
32	39	可设置	I2C1_ER	I^2C1 错误中断	0x0000 00C0
33	40	可设置	I2C2_EV	I^2C2 事件中断	0x0000 00C4
34	41	可设置	I2C2_ER	I^2C2 错误中断	0x0000 00C8
35	42	可设置	SPI1	SPI1 全局中断	0x0000 00CC
36	43	可设置	SPI2	SPI2 全局中断	0x0000 00D0
37	44	可设置	USART1	USART1 全局中断	0x0000 00D4
38	45	可设置	USART2	USART2 全局中断	0x0000 00D8
39	46	可设置	USART3	USART3 全局中断	0x0000 00DC
40	47	可设置	EXTI15_10	EXTI 线 [15:10] 中断	0x0000 00E0
41	48	可设置	RTC_Alarm	连接到 EXTI 线的 RTC 闹钟（A 和 B）中断	0x0000 00E4
42	49	可设置	OTG_FS WKUP	连接到 EXTI 线的 USB On-The-Go FS 唤醒中断	0x0000 00E8
43	50	可设置	TIM8_BRK_TIM12	TIM8 刹车中断和 TIM12 全局中断	0x0000 00EC
44	51	可设置	TIM8_UP_TIM13	TIM8 更新中断和 TIM13 全局中断	0x0000 00F0

表 46. STM32F42xxx 和 STM32F43xxx 的向量表 （续）

位置	优先级	优先级类型	名称	说明	地址
45	52	可设置	TIM8_TRG_COM_TIM14	TIM8 触发和换相中断与 TIM14 全局中断	0x0000 00F4
46	53	可设置	TIM8_CC	TIM8 捕获比较中断	0x0000 00F8
47	54	可设置	DMA1_Stream7	DMA1 流 7 全局中断	0x0000 00FC
48	55	可设置	FSMC	FSMC 全局中断	0x0000 0100
49	56	可设置	SDIO	SDIO 全局中断	0x0000 0104
50	57	可设置	TIM5	TIM5 全局中断	0x0000 0108
51	58	可设置	SPI3	SPI3 全局中断	0x0000 010C
52	59	可设置	UART4	UART4 全局中断	0x0000 0110
53	60	可设置	UART5	UART5 全局中断	0x0000 0114
54	61	可设置	TIM6_DAC	TIM6 全局中断， DAC1 和 DAC2 下溢错误中断	0x0000 0118
55	62	可设置	TIM7	TIM7 全局中断	0x0000 011C
56	63	可设置	DMA2_Stream0	DMA2 流 0 全局中断	0x0000 0120
57	64	可设置	DMA2_Stream1	DMA2 流 1 全局中断	0x0000 0124
58	65	可设置	DMA2_Stream2	DMA2 流 2 全局中断	0x0000 0128
59	66	可设置	DMA2_Stream3	DMA2 流 3 全局中断	0x0000 012C
60	67	可设置	DMA2_Stream4	DMA2 流 4 全局中断	0x0000 0130
61	68	可设置	ETH	以太网全局中断	0x0000 0134
62	69	可设置	ETH_WKUP	连接到 EXTI 线的以太网唤醒中断	0x0000 0138
63	70	可设置	CAN2_TX	CAN2 TX 中断	0x0000 013C
64	71	可设置	CAN2_RX0	CAN2 RX0 中断	0x0000 0140
65	72	可设置	CAN2_RX1	CAN2 RX1 中断	0x0000 0144
66	73	可设置	CAN2_SCE	CAN2 SCE 中断	0x0000 0148
67	74	可设置	OTG_FS	USB On The Go FS 全局中断	0x0000 014C
68	75	可设置	DMA2_Stream5	DMA2 流 5 全局中断	0x0000 0150
69	76	可设置	DMA2_Stream6	DMA2 流 6 全局中断	0x0000 0154
70	77	可设置	DMA2_Stream7	DMA2 流 7 全局中断	0x0000 0158
71	78	可设置	USART6	USART6 全局中断	0x0000 015C
72	79	可设置	I2C3_EV	I^2C3 事件中断	0x0000 0160
73	80	可设置	I2C3_ER	I^2C3 错误中断	0x0000 0164
74	81	可设置	OTG_HS_EP1_OUT	USB On The Go HS 端点 1 输出全局中断	0x0000 0168
75	82	可设置	OTG_HS_EP1_IN	USB On The Go HS 端点 1 输入全局中断	0x0000 016C

表 46. STM32F42xxx 和 STM32F43xxx 的向量表 （续）

位置	优先级	优先级类型	名称	说明	地址
76	83	可设置	OTG_HS_WKUP	连接到 EXTI 的 USB On The Go HS 唤醒中断	0x0000 0170
77	84	可设置	OTG_HS	USB On The Go HS 全局中断	0x0000 0174
78	85	可设置	DCMI	DCMI 全局中断	0x0000 0178
79	86	可设置	CRYP	CRYP 加密全局中断	0x0000 017C
80	87	可设置	HASH_RNG	哈希和随机数发生器全局中断	0x0000 0180
81	88	可设置	FPU	FPU 全局中断	0x0000 0184
82	89	可设置	UART7	UART 7 全局中断	0x0000 0188
83	90	可设置	UART8	UART 8 全局中断	0x0000 018C
84	91	可设置	SPI4	SPI 4 全局中断	0x0000 0190
85	92	可设置	SPI5	SPI 5 全局中断	0x0000 0194
86	93	可设置	SPI6	SPI 6 全局中断	0x0000 0198

10.2.1 EXTI 主要特性

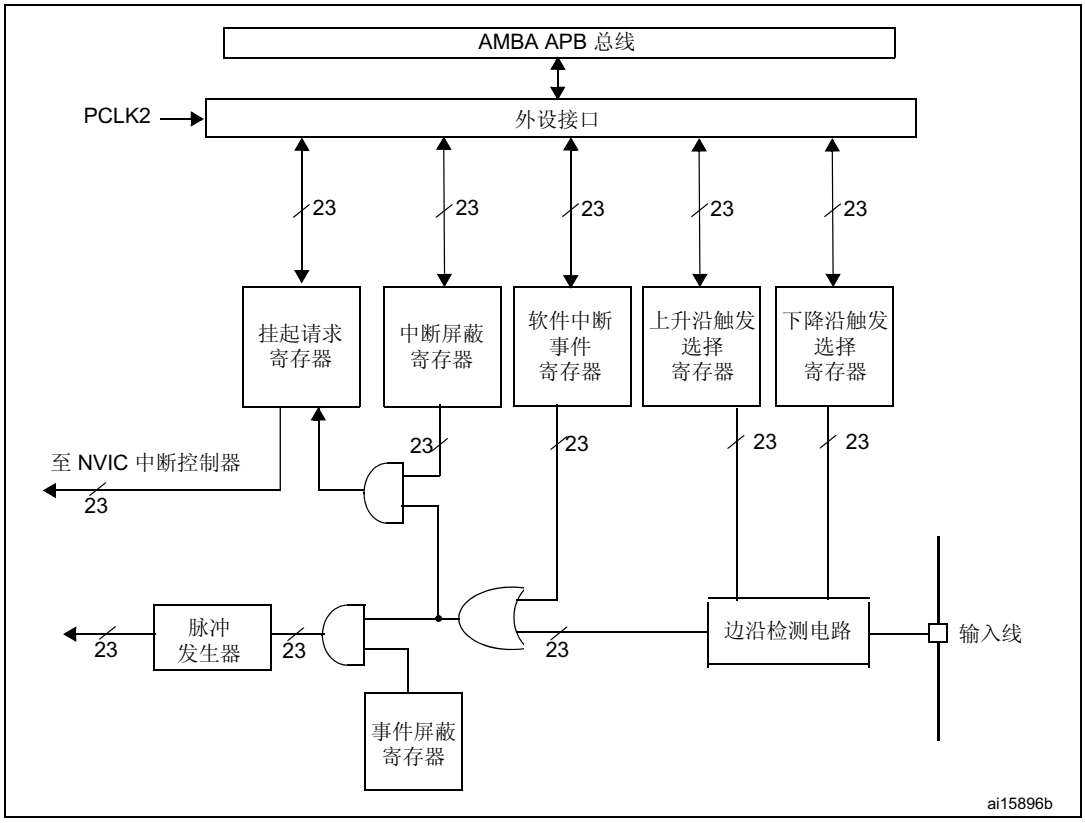
EXTI 控制器的主要特性如下：

- 每个中断/事件线上都具有独立的触发和屏蔽
- 每个中断线都具有专用的状态位
- 支持多达 23 个软件事件/中断请求
- 检测脉冲宽度低于 APB2 时钟宽度的外部信号。有关此参数的详细信息，请参见 STM32F4xx 数据手册的电气特性部分。

10.2.2 EXTI 框图

图 32 显示了框图。

图 32. 外部中断/事件控制器框图



10.2.3 唤醒事件管理

STM32F4xx 能够处理外部或内部事件来唤醒内核 (WFE)。唤醒事件可通过以下方式产生：

- 在外设的控制寄存器使能一个中断，但不在 NVIC 中使能，同时使能 Cortex™-M4F 系统控制寄存器中的 SEVONPEND 位。当 MCU 从 WFE 恢复时，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 恢复时，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。

使用外部线作为唤醒事件，请参见*第 10.2.4 节：功能说明*。

10.2.4 功能说明

要产生中断，必须先配置好并使能中断线。根据需要的边沿检测设置 2 个触发寄存器，同时在中断屏蔽寄存器的相应位写“1”使能中断请求。当外部中断线上出现选定信号沿时，便会产生中断请求，对应的挂起位也会置 1。在挂起寄存器的对应位写“1”，将清除该中断请求。

要产生事件，必须先配置好并使能事件线。根据需要的边沿检测设置 2 个触发寄存器，同时在事件屏蔽寄存器的相应位写“1”允许事件请求。当事件线上出现选定信号沿时，便会产生事件脉冲，对应的挂起位不会置 1。

通过在软件中对软件中断/事件寄存器写“1”，也可以产生中断/事件请求。

硬件中断选择

要配置 23 根线作为中断源，请执行以下步骤：

- 配置 23 根中断线的屏蔽位 (EXTI_IMR)
- 配置中断线的触发选择位（EXTI_RTSR 和 EXTI_FTSR）
- 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，使得 23 个中断线中的请求可以被正确地响应。

硬件事件选择

要配置 23 根线作为事件源，请执行以下步骤：

- 配置 23 根事件线的屏蔽位 (EXTI_EMR)
- 配置事件线的触发选择位（EXTI_RTSR 和 EXTI_FTSR）

软件中断/事件选择

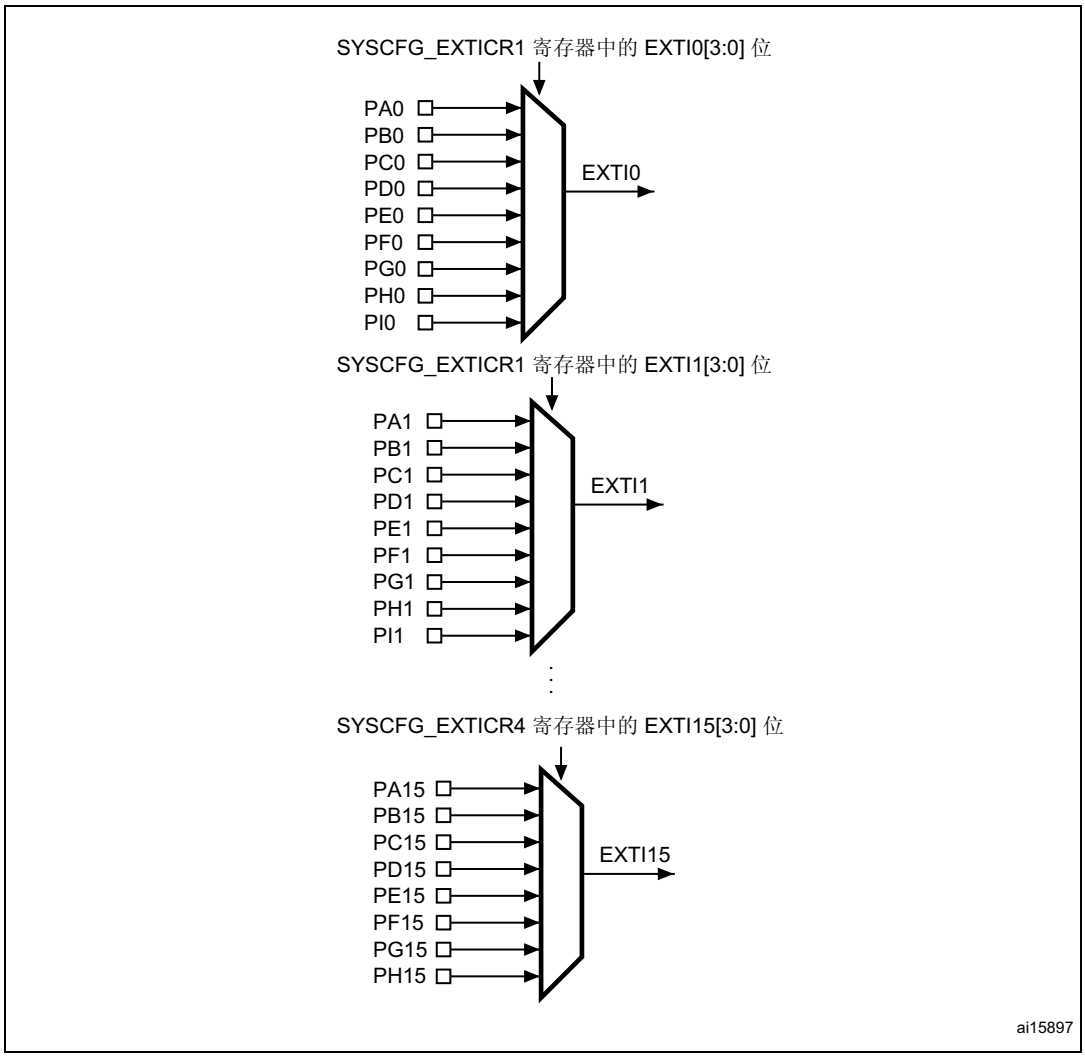
可将这 23 根线配置为软件中断/事件线。以下为产生软件中断的步骤。

- 配置 23 根中断/事件线的屏蔽位（EXTI_IMR、EXTI_EMR）
- 在软件中断寄存器设置相应的请求位 (EXTI_SWIER)

10.2.5 外部中断/事件线映射

多达 140 个 GPIO（STM32F405xx/07xx 和 STM32F415xx/17xx）通过以下方式连接到 16 个外部中断/事件线：

图 33. 外部中断/事件 GPIO 映射



另外七根 EXTI 线连接方式如下：

- EXTI 线 16 连接到 PVD 输出
- EXTI 线 17 连接到 RTC 闹钟事件
- EXTI 线 18 连接到 USB OTG FS 唤醒事件
- EXTI 线 19 连接到以太网唤醒事件
- EXTI 线 20 连接到 USB OTG HS（在 FS 中配置）唤醒事件
- EXTI 线 21 连接到 RTC 入侵和时间戳事件
- EXTI 线 22 连接到 RTC 唤醒事件

10.3 EXTI 寄存器

有关寄存器说明中使用的缩写，请参见*第 47 页的第 1.1 节*。

10.3.1 中断屏蔽寄存器 (EXTI_IMR)

Interrupt mask register

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
Reserved									MR22	MR21	MR20	MR19	MR18	MR17	MR16
									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 31:23 保留，必须保持复位值。

位 22:0 **MRx**：x 线上的中断屏蔽 (Interrupt mask on line x)

0：屏蔽来自 x 线的中断请求
1：开放来自 x 线的中断请求

10.3.2 事件屏蔽寄存器 (EXTI_EMR)

Event mask register

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
Reserved									MR22	MR21	MR20	MR19	MR18	MR17	MR16
									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 31:23 保留，必须保持复位值。

位 22:0 **MRx**：x 线上的事件屏蔽 (Event mask on line x)

0：屏蔽来自 x 线的事件请求
1：开放来自 x 线的事件请求

10.3.3 上升沿触发选择寄存器 (EXTI_RTSR)

Rising trigger selection register

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
Reserved									TR22	TR21	TR20	TR19	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 31:23 保留，必须保持复位值。

位 22:0 **TRx**：线 x 的上升沿触发事件配置位 (Rising trigger event configuration bit of line x)
0：禁止输入线上升沿触发（事件和中断）
1：允许输入线上升沿触发（事件和中断）

注意： *外部唤醒线配置为边沿触发时，在这些线上不能出现毛刺信号。*
如果在向 EXTI_RTSR 寄存器写入值的同时外部中断线上产生上升沿，挂起位将被置位。
在同一中断线上，可以同时设置上升沿和下降沿触发。即任一边沿都可触发中断。

10.3.4 下降沿触发选择寄存器 (EXTI_FTSR)

Falling trigger selection register

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
Reserved									TR22	TR21	TR20	TR19	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 31:23 保留，必须保持复位值。

位 22:0 **TRx**：线 x 的下降沿触发事件配置位 (Falling trigger event configuration bit of line x)
0：禁止输入线下降沿触发（事件和中断）
1：允许输入线下降沿触发（事件和中断）

注意： *外部唤醒线配置为边沿触发时，在这些线上不能出现毛刺信号。*
如果在向 EXTI_FTSR 寄存器写入值的同时外部中断线上产生下降沿，挂起位不会被置位。
在同一中断线上，可以同时设置上升沿和下降沿触发。即任一边沿都可触发中断。

10.3.5 软件中断事件寄存器 (EXTI_SWIER)

Software interrupt event register

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
Reserved									SWIER 22	SWIER 21	SWIER 20	SWIER 19	SWIER 18	SWIER 17	SWIER 16
									rw	rw	rw	rw	rw	rw	rw

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

位 31:23 保留，必须保持复位值。

位 22:0 **SWIERx**：线 x 上的软件中断 (Software Interrupt on line x)

当该位为“0”时，写“1”将设置 EXTI_PR 中相应的挂起位。如果在 EXTI_IMR 和 EXTI_EMR 中允许产生该中断，则产生中断请求。

通过清除 EXTI_PR 的对应位（写入“1”），可以清除该位为“0”。

10.3.6 挂起寄存器 (EXTI_PR)

Pending register

偏移地址：0x14
复位值：未定义

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved									PR22	PR21	PR20	PR19	PR18	PR17	PR16
									rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

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
rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

位 31:23 保留，必须保持复位值。

位 22:0 **PRx**：挂起位 (Pending bit)

0：没有发生触发请求

1：发生了选择的触发请求

当在外部中断线上发生了选择的边沿事件，该位被置“1”。在此位中写入“1”可以清除它，也可以通过改变边沿检测的极性清除。

10.3.7 EXTI 寄存器映射

表 47 给出了 EXTI 寄存器映射和复位值。

表 47. 外部中断/事件控制器寄存器映射和复位值

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	EXTI_IMR	Reserved									MR[22:0]																						
	Reset value										0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	EXTI_EMR	Reserved									MR[22:0]																						
	Reset value										0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08	EXTI_RTSR	Reserved									TR[22:0]																						
	Reset value										0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	EXTI_FTSR	Reserved									TR[22:0]																						
	Reset value										0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	EXTI_SWIER	Reserved									SWIER[22:0]																						
	Reset value										0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14	EXTI_PR	Reserved									PR[22:0]																						
	Reset value										0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

有关寄存器边界地址的信息，请参见*第 52 页的表 2*。

11 模数转换器 (ADC)

除非特别说明，否则本部分适用于整个 STM32F4xx 系列。

11.1 ADC 简介

12 位 ADC 是逐次趋近型模数转换器。它具有多达 19 个复用通道，可测量来自 16 个外部源、两个内部源和 V_{BAT} 通道的信号。这些通道的 A/D 转换可在单次、连续、扫描或不连续采样模式下进行。ADC 的结果存储在一个左对齐或右对齐的 16 位数据寄存器中。

ADC 具有模拟看门狗特性，允许应用检测输入电压是否超过了用户自定义的阈值上限或下限。

11.2 ADC 主要特性

- 可配置 12 位、10 位、8 位或 6 位分辨率
- 在转换结束、注入转换结束以及发生模拟看门狗或溢出事件时产生中断
- 单次和连续转换模式
- 用于自动将通道 0 转换为通道“n”的扫描模式
- 数据对齐以保持内置数据一致性
- 可独立设置各通道采样时间
- 外部触发器选项，可为规则转换和注入转换配置极性
- 不连续采样模式
- 双重/三重模式（具有 2 个或更多 ADC 的器件提供）
- 双重/三重 ADC 模式下可配置的 DMA 数据存储
- 双重/三重交替模式下可配置的转换间延迟
- ADC 转换类型（参见数据手册）
- ADC 电源要求：全速运行时为 2.4 V 到 3.6 V，慢速运行时为 1.8 V
- ADC 输入范围：$V_{REF-} \leq V_{IN} \leq V_{REF+}$
- 规则通道转换期间可产生 DMA 请求

图 34 显示了 ADC 的框图。

注意： *V_{REF-} 如果可用（取决于封装），则必须将其连接到 V_{SSA}。*

11.3 ADC 功能说明

图 34 给出了单个 ADC 的框图，*表 48* 列出了 ADC 的引脚说明。

图 34. 单个 ADC 框图

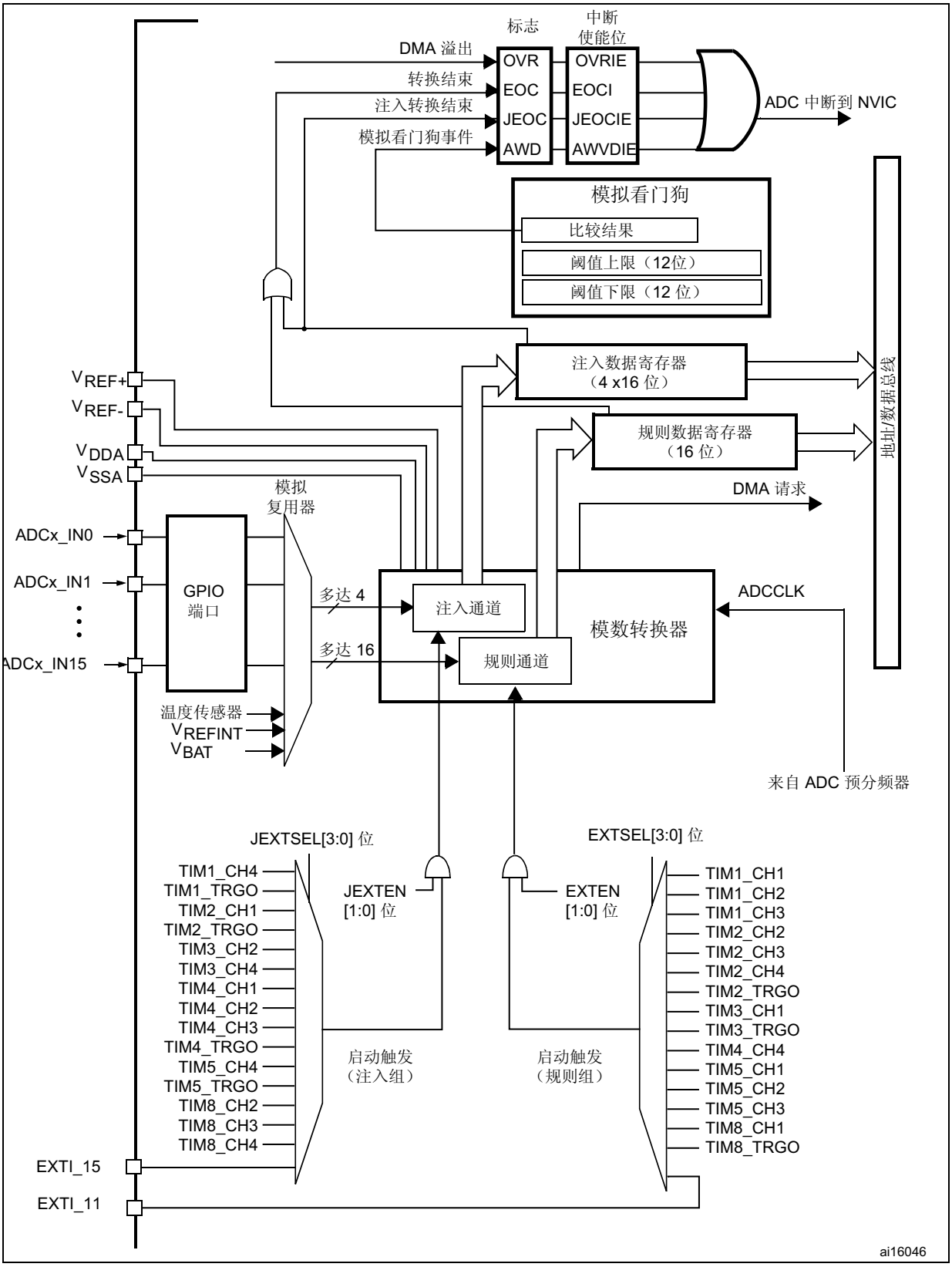

标志
中断
使能位
DMA 溢出
转换结束
注入转换结束
模拟看门狗事件
OVR
EOC
JEOC
AWD
OVRIE
EOCI
JEOCIE
AWVDIE
ADC 中断到 NVIC
模拟看门狗
比较结果
阈值上限（12位）
阈值下限（12 位）
注入数据寄存器
（4 x16 位）
规则数据寄存器
（16 位）
地址/数据总线
DMA 请求
VREF+
VREF-
VDDA
VSSA
模拟
复用器
ADCx_IN0
ADCx_IN1
ADCx_IN15
GPIO
端口
多达 4
多达 16
注入通道
规则通道
模数转换器
ADCCLK
温度传感器
VREFINT
VBAT
来自 ADC 预分频器
JEXTSEL[3:0] 位
EXTSEL[3:0] 位
TIM1_CH4
TIM1_TRGO
TIM2_CH1
TIM2_TRGO
TIM3_CH2
TIM3_CH4
TIM4_CH1
TIM4_CH2
TIM4_CH3
TIM4_TRGO
TIM5_CH4
TIM5_TRGO
TIM8_CH2
TIM8_CH3
TIM8_CH4
JEXTEN
[1:0] 位
EXTEN
[1:0] 位
TIM1_CH1
TIM1_CH2
TIM1_CH3
TIM2_CH2
TIM2_CH3
TIM2_CH4
TIM2_TRGO
TIM3_CH1
TIM3_TRGO
TIM4_CH4
TIM5_CH1
TIM5_CH2
TIM5_CH3
TIM8_CH1
TIM8_TRGO
启动触发
（注入组）
启动触发
（规则组）
EXTI_15
EXTI_11
ai16046

表 48. ADC 引脚

名称	信号类型	备注
V_{REF+}	正模拟参考电压输入	ADC 高/正参考电压，1.8 V ≤ V_{REF+} ≤ V_{DDA}
V_{DDA}	模拟电源输入	模拟电源电压等于 V_{DD}， 全速运行时，2.4 V ≤ V_{DDA} ≤ V_{DD} (3.6 V) 低速运行时，1.8 V ≤ V_{DDA} ≤ V_{DD} (3.6 V)
V_{REF-}	负模拟参考电压输入	ADC 低/负参考电压，V_{REF-} = V_{SSA}
V_{SSA}	模拟电源接地输入	模拟电源接地电压等于 V_{SS}
ADCx_IN[15:0]	模拟输入信号	16 个模拟输入通道

11.3.1 ADC 开关控制

可通过将 ADC_CR2 寄存器中的 ADON 位置 1 来为 ADC 供电。首次将 ADON 位置 1 时，会将 ADC 从掉电模式中唤醒。

SWSTART 或 JSWSTART 位置 1 时，启动 AD 转换。

可通过将 ADON 位清零来停止转换并使 ADC 进入掉电模式。在此模式下，ADC 几乎不耗电（只有几 µA）。

11.3.2 ADC 时钟

ADC 具有两个时钟方案：

- 用于模拟电路的时钟：ADCCLK，所有 ADC 共用

  此时钟来自于经可编程预分频器分频的 APB2 时钟，该预分频器允许 ADC 在 f_{PCLK2}/2、/4、/6 或 /8 下工作。有关 ADCCLK 的最大值，请参见数据手册。
- 用于数字接口的时钟（用于寄存器读/写访问）

  此时钟等效于 APB2 时钟。可以通过 RCC APB2 外设时钟使能寄存器 (RCC_APB2ENR) 分别为每个 ADC 使能/禁止数字接口时钟。

11.3.3 通道选择

有 16 条复用通道。可以将转换分为两组：规则转换和注入转换。每个组包含一个转换序列，该序列可按任意顺序在任意通道上完成。例如，可按以下顺序对序列进行转换：ADC_IN3、ADC_IN8、ADC_IN2、ADC_IN2、ADC_IN0、ADC_IN2、ADC_IN2、ADC_IN15。

- 一个**规则转换组**最多由 16 个转换构成。必须在 ADC_SQRx 寄存器中选择转换序列的规则通道及其顺序。规则转换组中的转换总数必须写入 ADC_SQR1 寄存器中的 L[3:0] 位。
- 一个**注入转换组**最多由 4 个转换构成。必须在 ADC_JSQR 寄存器中选择转换序列的注入通道及其顺序。注入转换组中的转换总数必须写入 ADC_JSQR 寄存器中的 L[1:0] 位。

如果在转换期间修改 ADC_SQRx 或 ADC_JSQR 寄存器，将复位当前转换并向 ADC 发送一个新的启动脉冲，以转换新选择的组。

温度传感器、V_{REFINT} 和 V_{BAT} 内部通道

- 对于 STM32F40x 和 STM32F41x 器件，温度传感器内部连接到通道 ADC1_IN16。
  内部参考电压 VREFINT 连接到 ADC1_IN17。
- 对于 STM23F42x 和 STM32F43x 器件，温度传感器内部连接到与 VBAT 共用的通道 ADC1_IN18。一次只能选择一个转换（温度传感器或 VBAT）。同时设置了温度传感器和 VBAT 转换时，将只进行 VBAT 转换。
  内部参考电压 VREFINT 连接到 ADC1_IN17。

V_{BAT} 通道连接到通道 ADC1_IN18。该通道也可转换为注入通道或规则通道。

注意： *温度传感器、V_{REFINT} 和 V_{BAT} 通道只在主 ADC1 外设上可用。*

11.3.4 单次转换模式

在单次转换模式下，ADC 执行一次转换。CONT 位为 0 时，可通过以下方式启动此模式：

- 将 ADC_CR2 寄存器中的 SWSTART 位置 1（仅适用于规则通道）
- 将 JSWSTART 位置 1（适用于注入通道）
- 外部触发（适用于规则通道或注入通道）

完成所选通道的转换之后：

- 如果转换了规则通道：
  - 转换数据存储在 16 位 ADC_DR 寄存器中
  - EOC（转换结束）标志置 1
  - EOCIE 位置 1 时将产生中断
- 如果转换了注入通道：
  - 转换数据存储在 16 位 ADC_JDR1 寄存器中
  - JEOC（注入转换结束）标志置 1
  - JEOCIE 位置 1 时将产生中断

然后，ADC 停止。

11.3.5 连续转换模式

在连续转换模式下，ADC 结束一个转换后立即启动一个新的转换。CONT 位为 1 时，可通过外部触发或将 ADC_CR2 寄存器中的 SWSTRT 位置 1 来启动此模式（仅适用于规则通道）。

每次转换之后：

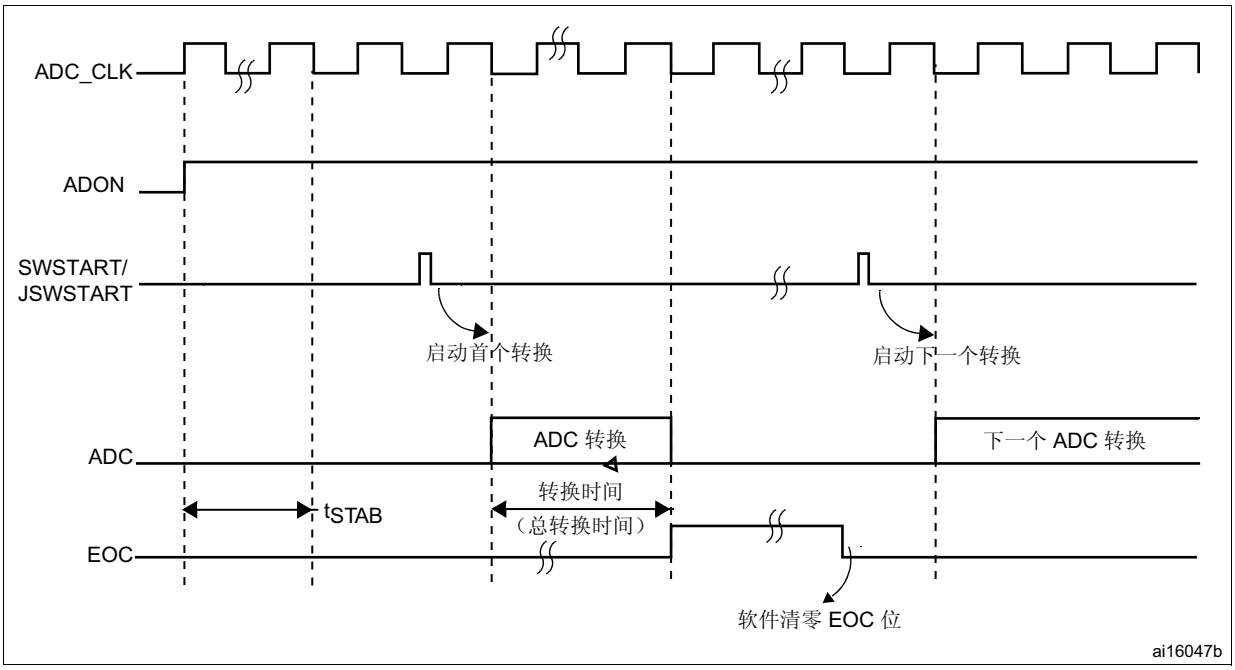
- 如果转换了规则通道组：
  - 上次转换的数据存储在 16 位 ADC_DR 寄存器中
  - EOC（转换结束）标志置 1
  - EOCIE 位置 1 时将产生中断

注意： *无法连续转换注入通道。连续模式下唯一的例外情况是，注入通道配置为在规则通道之后自动转换（使用 JAUTO 位），请参见自动注入一节。*

11.3.6 时序图

如*图 35* 所示，ADC 在开始精确转换之前需要一段稳定时间 t_{STAB}。ADC 开始转换并经过 15 个时钟周期后，EOC 标志置 1，转换结果存放在 16 位 ADC 数据寄存器中。

图 35. 时序图



11.3.7 模拟看门狗

如果 ADC 转换的模拟电压低于阈值下限或高于阈值上限，则 AWD 模拟看门狗状态位会置 1。这些阈值在 ADC_HTR 和 ADC_LTR 16 位寄存器的 12 个最低有效位中进行编程。可以使用 ADC_CR1 寄存器中的 AWDIE 位使能中断。

阈值与 ADC_CR2 寄存器中的 ALIGN 位的所选对齐方式无关。在对齐之前，会将模拟电压与阈值上限和下限进行比较。

表 49 介绍了应如何配置 ADC_CR1 寄存器才能在一个或多个通道上使能模拟看门狗。

图 36. 模拟看门狗的保护区域

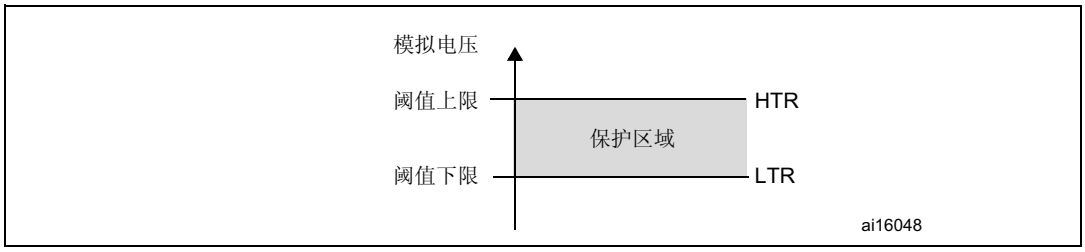


表 49. 模拟看门狗通道选择

模拟看门狗保护的通道	ADC_CR1 寄存器控制位（x = 无关）		
	AWDSGL 位	AWDEN 位	JAWDEN 位
无	x	0	0
所有注入通道	0	0	1
所有规则通道	0	1	0
所有规则通道和注入通道	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(1)规则通道或注入通道	1	1	1

1. 通过 AWDCH[4:0] 位选择

11.3.8 扫描模式

此模式用于扫描一组模拟通道。

通过将 ADC_CR1 寄存器中的 SCAN 位置 1 来选择扫描模式。将此位置 1 后，ADC 会扫描在 ADC_SQRx 寄存器（对于规则通道）或 ADC_JSQR 寄存器（对于注入通道）中选择的所有通道。为组中的每个通道都执行一次转换。每次转换结束后，会自动转换该组中的下一个通道。如果将 CONT 位置 1，规则通道转换不会在组中最后一个所选通道处停止，而是再次从第一个所选通道继续转换。

如果将 DMA 位置 1，则在每次规则通道转换之后，均使用直接存储器访问 (DMA) 控制器将转换自规则通道组的数据（存储在 ADC_DR 寄存器中）传输到 SRAM。

在以下情况下，ADC_SR 寄存器中的 EOC 位置 1：

- 如果 EOCS 位清零，在每个规则组序列转换结束时
- 如果 EOCS 位置 1，在每个规则通道转换结束时

从注入通道转换的数据始终存储在 ADC_JDRx 寄存器中。

11.3.9 注入通道管理

触发注入

要使用触发注入，必须将 ADC_CR1 寄存器中的 JAUTO 位清零。

1. 通过外部触发或将 ADC_CR2 寄存器中的 SWSTART 位置 1 来启动规则通道组转换。
2. 如果在规则通道组转换期间出现外部注入触发或者 JSWSTART 位置 1，则当前的转换会复位，并且注入通道序列会切换为单次扫描模式。
3. 然后，规则通道组的规则转换会从上次中断的规则转换处恢复。
   如果在注入转换期间出现规则事件，注入转换不会中断，但在注入序列结束时会执行规则序列。*图 37* 显示了相应的时序图。

注意： *使用触发注入时，必须确保触发事件之间的间隔长于注入序列。例如，如果序列长度为 30 个 ADC 时钟周期（即，采样时间为 3 个时钟周期的两次转换），则触发事件的最小间隔不能小于 31 个 ADC 时钟周期。*

自动注入

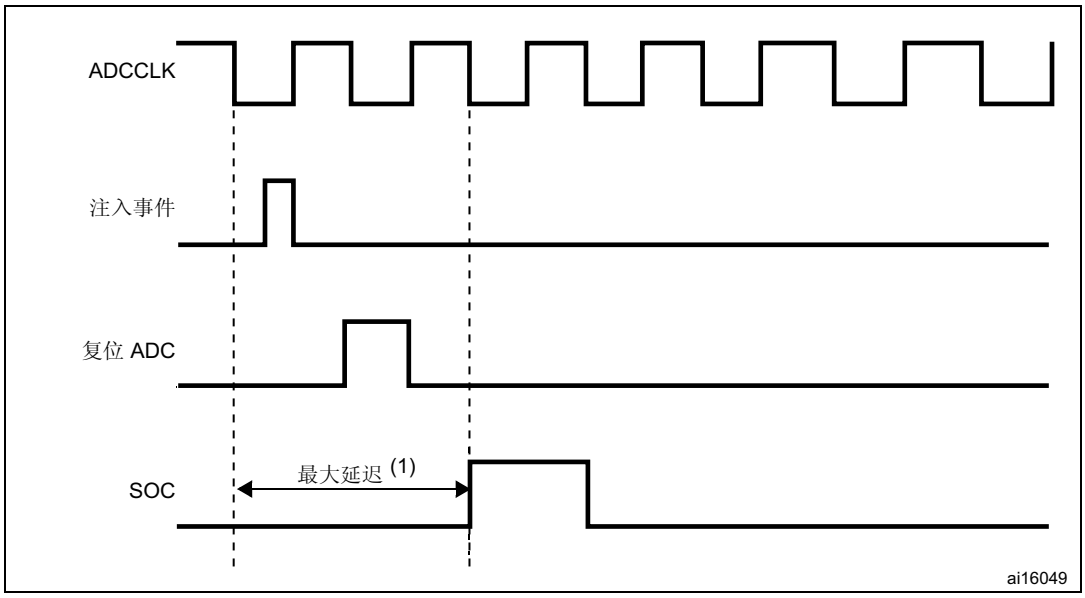
如果将 JAUTO 位置 1，则注入组中的通道会在规则组通道之后自动转换。这可用于转换最多由 20 个转换构成的序列，这些转换在 ADC_SQRx 和 ADC_JSQR 寄存器中编程。

在此模式下，必须禁止注入通道上的外部触发。

如果 CONT 位和 JAUTO 位均已置 1，则在转换规则通道之后会继续转换注入通道。

注意： *不能同时使用自动注入和不连续采样模式。*

图 37. 注入转换延迟



1. 有关延迟的最大值，请参见 STM32F40x 和 STM32F41x 数据手册中的电气特性部分。

11.3.10 不连续采样模式

规则组

可将 ADC_CR1 寄存器中的 DISCEN 位置 1 来使能此模式。该模式可用于转换含有 n (n ≤ 8) 个转换的短序列，该短序列是在 ADC_SQRx 寄存器中选择的转换序列的一部分。可通过写入 ADC_CR1 寄存器中的 DISCNUM[2:0] 位来指定 n 的值。

出现外部触发时，将启动在 ADC_SQRx 寄存器中选择的接下来 n 个转换，直到序列中的所有转换均完成为止。通过 ADC_SQR1 寄存器中的 L[3:0] 位定义总序列长度。

示例：

n = 3，要转换的通道 = 0、1、2、3、6、7、9、10
第 1 次触发：转换序列 0、1、2
第 2 次触发：转换序列 3、6、7
第 3 次触发：转换序列 9、10 并生成 EOC 事件
第 4 次触发：转换序列 0、1、2

注意： *在不连续采样模式下转换规则组时，不会出现翻转。*

转换完所有子组后，下一个触发信号将启动第一个子组的转换。在上述示例中，第 4 次触发重新转换了第 1 个子组中的通道 0、1 和 2。

注入组

可将 ADC_CR1 寄存器中的 JDISCEN 位置 1 来使能此模式。在出现外部触发事件之后，可使用该模式逐通道转换在 ADC_JSQR 寄存器中选择的序列。

出现外部触发时，将启动在 ADC_JSQR 寄存器中选择的下一个通道转换，直到序列中的所有转换均完成为止。通过 ADC_JSQR 寄存器中的 JL[1:0] 位定义总序列长度。

示例：

n = 1，要转换的通道 = 1、2、3
第 1 次触发：转换通道 1
第 2 次触发：转换通道 2
第 3 次触发：转换通道 3 并生成 EOC 和 JEOC 事件
第 4 次触发：通道 1

注意： *转换完所有注入通道后，下一个触发信号将启动第一个注入通道的转换。在上述示例中，第 4 次触发重新转换了第 1 个注入通道。*

不能同时使用自动注入和不连续采样模式。

不得同时为规则组和注入组设置不连续采样模式。只能针对一个组使能不连续采样模式。

11.4 数据对齐

ADC_CR2 寄存器中的 ALIGN 位用于选择转换后存储的数据的对齐方式。可选择左对齐和右对齐两种方式，如*图 38* 和*图 39* 所示。

注入通道组的转换数据将减去 ADC_JOFRx 寄存器中写入的用户自定义偏移量，因此结果可以是一个负值。SEXT 位表示扩展的符号值。

对于规则组中的通道，不会减去任何偏移量，因此只有十二个位有效。

图 38. 12 位数据的右对齐

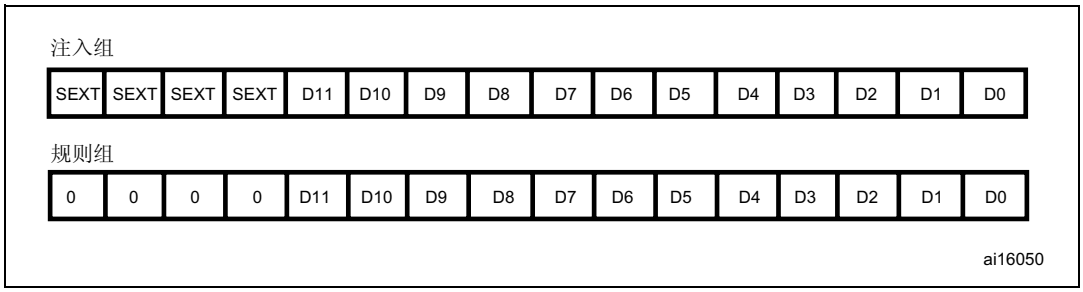
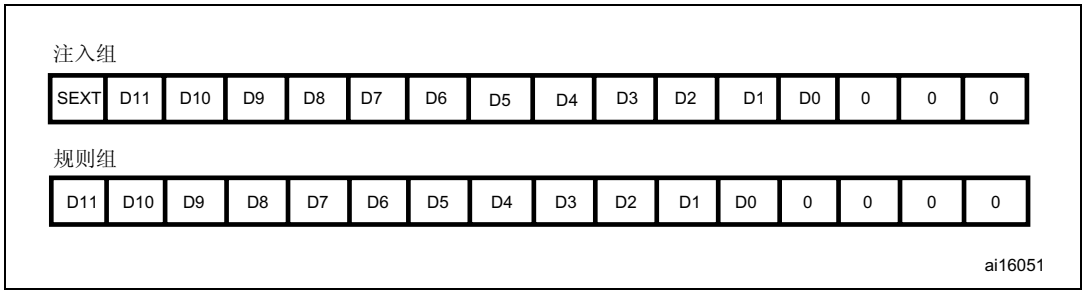
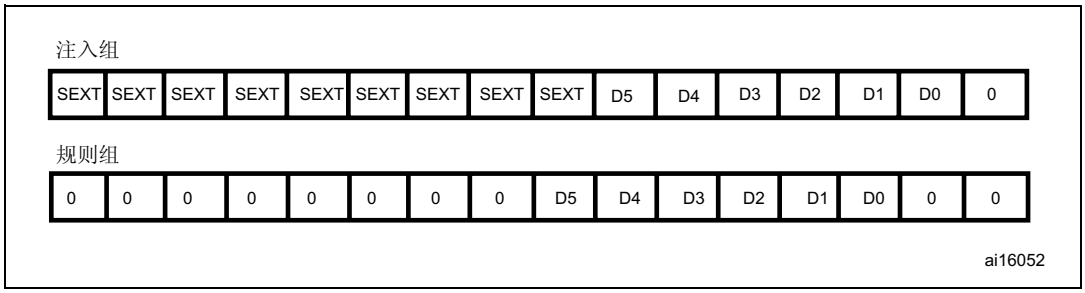


图 39. 12 位数据的左对齐



特例：采用左对齐时，数据基于半字进行对齐，除了分辨率设置为 6 位时。分辨率设置为 6 位时，数据基于字节进行对齐，如*图 40* 所示。

图 40. 6 位数据的左对齐



11.5 可独立设置各通道采样时间

ADC 会在数个 ADCCLK 周期内对输入电压进行采样，可使用 ADC_SMPR1 和 ADC_SMPR2 寄存器中的 SMP[2:0] 位修改周期数。每个通道均可以使用不同的采样时间进行采样。

总转换时间的计算公式如下：

T_{conv} = 采样时间 + 12 个周期

示例：

ADCCLK = 30 MHz 且采样时间 = 3 个周期时：

T_{conv} = 3 + 12 = 15 个周期 = 0.5 µs（APB2 为 60 MHz 时）

11.6 外部触发转换和触发极性

可以通过外部事件（例如，定时器捕获、EXTI 中断线）触发转换。如果 EXTEN[1:0] 控制位（对于行规转换）或 JEXTEN[1:0] 位（对于注入转换）不等于“0b00”，则外部事件能够以所选极性触发转换。*表 50* 提供了 EXTEN[1:0] 和 JEXTEN[1:0] 值与触发极性之间的对应关系。

表 50. 配置触发极性

源	EXTEN[1:0]/JEXTEN[1:0]
禁止触发检测	00
在上升沿时检测	01
在下降沿时检测	10
在上升沿和下降沿均检测	11

注意： *可以实时更改外部触发的极性。*

EXTSEL[3:0] 和 JEXTSEL[3:0] 控制位用于从 16 个可能事件中选择可触发规则组转换和注入组转换的事件。

表 51 给出了可用于规则转换的外部触发。

表 51. 规则通道的外部触发

源	类型	EXTSEL[3:0]
TIM1_CH1 事件	片上定时器的内部信号	0000
TIM1_CH2 事件		0001
TIM1_CH3 事件		0010
TIM2_CH2 事件		0011
TIM2_CH3 事件		0100
TIM2_CH4 事件		0101
TIM2_TRGO 事件		0110
TIM3_CH1 事件		0111
TIM3_TRGO 事件		1000
TIM4_CH4 事件		1001
TIM5_CH1 事件		1010
TIM5_CH2 事件		1011
TIM5_CH3 事件		1100
TIM8_CH1 事件		1101
TIM8_TRGO 事件		1110
EXTI 线 11	外部引脚	1111

表 52 给出了可用于注入转换的外部触发。

表 52. 注入通道的外部触发

源	连接类型	JEXTSEL[3:0]
TIM1_CH4 事件	片上定时器的内部信号	0000
TIM1_TRGO 事件		0001
TIM2_CH1 事件		0010
TIM2_TRGO 事件		0011
TIM3_CH2 事件		0100
TIM3_CH4 事件		0101
TIM4_CH1 事件		0110
TIM4_CH2 事件		0111
TIM4_CH3 事件		1000
TIM4_TRGO 事件		1001
TIM5_CH4 事件		1010
TIM5_TRGO 事件		1011
TIM8_CH2 事件		1100
TIM8_CH3 事件		1101
TIM8_CH4 事件		1110
EXTI 线 15	外部引脚	1111

可通过将 ADC_CR2 寄存器中的 SWSTART（对于规则转换）或 JSWSTART（对于注入转换）位置 1 来产生软件源触发事件。

可通过注入触发中断规则组转换。

注意： *可以实时更改触发选择。不过，当更改触发选择时，会在 1 个 APB 时钟周期的时间范围内禁止触发检测。这是为了避免在转换期间出现意外检测。*

11.7 快速转换模式

可通过降低 ADC 分辨率来执行快速转换。RES 位用于选择数据寄存器中可用的位数。每种分辨率的最小转换时间如下：

- 12 位：3 + 12 = 15 ADCCLK 周期
- 10 位：3 + 10 = 13 ADCCLK 周期
- 8 位：3 + 8 = 11 ADCCLK 周期
- 6 位：3 + 6 = 9 ADCCLK 周期

11.8 数据管理

11.8.1 使用 DMA

由于规则通道组只有一个数据寄存器，因此，对于多个规则通道的转换，使用 DMA 非常有帮助。这样可以避免丢失在下一次写入之前还未被读出的 ADC_DR 寄存器中的数据。

在使能 DMA 模式的情况下（ADC_CR2 寄存器中的 DMA 位置 1），每完成规则通道组中的一个通道转换后，都会生成一个 DMA 请求。这样便可将转换的数据从 ADC_DR 寄存器传输到用软件选择的目标位置。

尽管如此，如果数据丢失（溢出），则会将 ADC_SR 寄存器中的 OVR 位置 1 并生成一个中断（如果 OVRIE 使能位已置 1）。随后会禁止 DMA 传输并且不再接受 DMA 请求。在这种情况下，如果生成 DMA 请求，则会中止正在进行的规则转换并忽略之后的规则触发。随后需要将所使用的 DMA 流中的 OVR 标志和 DMAEN 位清零，并重新初始化 DMA 和 ADC，以将需要的转换通道数据传输到正确的存储器单元。只有这样，才能恢复转换并再次使能数据传输。注入通道转换不会受到溢出错误的影响。

在 DMA 模式下，当 OVR = 1 时，传送完最后一个有效数据后会阻止 DMA 请求，这意味着传输到 RAM 的所有数据均被视为有效。

在最后一次 DMA 传输（DMA 控制器的 DMA_SxRTR 寄存器中配置的传输次数）结束时：

- 如果将 ADC_CR2 寄存器中的 DDS 位清零，则不会向 DMA 控制器发出新的 DMA 请求（这可避免产生溢出错误）。不过，硬件不会将 DMA 位清零。必须将该位写入 0 然后写入 1 才能启动新的传输。
- 如果将 DDS 位置 1，则可继续生成请求。从而允许在双缓冲区循环模式下配置 DMA。

要在使用 DMA 时将 ADC 从 OVR 状态中恢复，请按以下步骤操作：

1. 重新初始化 DMA（调整目标地址和 NDTR 计数器）
2. 将 ADC_SR 寄存器中的 ADC OVR 位清零
3. 触发 ADC 以开始转换。

11.8.2 在不使用 DMA 的情况下管理转换序列

如果转换过程足够慢，则可使用软件来处理转换序列。在这种情况下，必须将 ADC_CR2 寄存器中的 EOCS 位置 1，才能使 EOC 状态位在每次转换结束时置 1，而不仅是在序列结束时置 1。当 EOCS = 1 时，会自动使能溢出检测。因此，每当转换结束时，EOC 都会置 1，并且可以读取 ADC_DR 寄存器。溢出管理与使用 DMA 时的管理相同。

要在 EOCS 位置 1 时将 ADC 从 OVR 状态中恢复，请按以下步骤操作：

1. 将 ADC_SR 寄存器中的 ADC OVR 位清零
2. 触发 ADC 以开始转换。

11.8.3 在不使用 DMA 和溢出检测的情况下进行转换

ADC 在转换一个或多个通道时不是每次都读取数据的情况下，这可能会很有用（例如，存在模拟看门狗时）。为此，必须禁止 DMA (DMA = 0) 并且仅在序列结束 (EOCS = 0) 时才将 EOC 位置 1。在此配置中，溢出检测已禁止。

11.9 多重 ADC 模式

在具有两个或更多 ADC 的器件中，可使用双重（具有两个 ADC）和三重（具有三个 ADC）ADC 模式（参见*图 41*）。

在多重 ADC 模式下，通过 ADC1 主器件到 ADC2 和 ADC3 从器件的交替触发或同时触发来启动转换，具体取决于 ADC_CCR 寄存器中的 MULTI[4:0] 位所选的模式。

注意： *在多重 ADC 模式下，配置外部事件触发转换时，应用必须设置为仅主器件触发而禁止从器件触发，以防止出现意外触发而启动不需要的从转换。*

可实现以下四种模式：

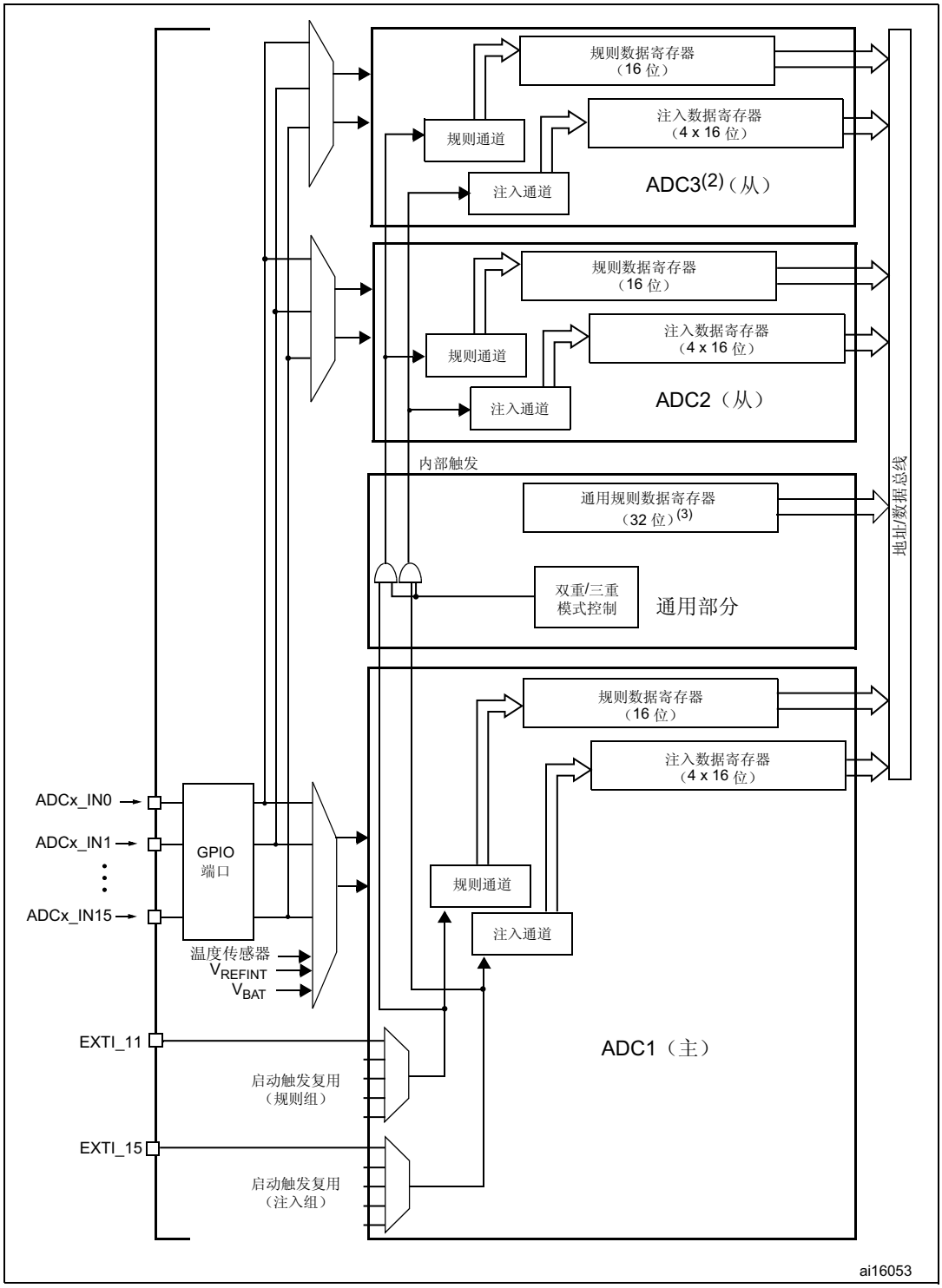
- 注入同时模式
- 规则同时模式
- 交替模式
- 交替触发模式

也可按以下方式组合使用上述模式：

- 注入同时模式 + 规则同时模式
- 规则同时模式 + 交替触发模式

注意： *在多重 ADC 模式下，可在多模式数据寄存器 (ADC_CDR) 中读取转换的数据。可在多模式状态寄存器 (ADC_CSR) 中读取状态位。*

图 41. 多重 ADC 框图(1)



1. 尽管 ADC2 和 ADC3 上存在外部触发，但它们并未显示在此图中。
2. 在双重 ADC 模式下，不存在 ADC3 从器件部分。
3. 在三重 ADC 模式下，ADC 通用数据寄存器 (ADC_CDR) 包含 ADC1、ADC2 和 ADC3 的规则转换数据。按照所选的存储顺序使用全部 32 个寄存器位。
在双重 ADC 模式下，ADC 通用数据寄存器 (ADC_CDR) 包含 ADC1 和 ADC2 的规则转换数据。使用全部 32 个寄存器位。

- 多重 ADC 模式下的 DMA 请求：

  在多重 ADC 模式下，可将 DMA 配置为使用三种不同的模式来传输转换的数据。在所有情况下，要使用的 DMA 流均连接到 ADC：

  – **DMA 模式 1**：每发出一个 DMA 请求（一个数据项可用），就会传输一个表示 ADC 转换的数据项的半字。

    在双重 ADC 模式下，发出第一个请求时传输 ADC1 的数据，发出第二个请求时传输 ADC2 的数据，依次类推。

    在三重 ADC 模式下，发出第一个请求时传输 ADC1 的数据，发出第二个请求时传输 ADC2 的数据，发出第三个请求时传输 ADC3 的数据；重复此序列。因此 DMA 首先传输 ADC1 的数据，随后传输 ADC2 的数据，再传输 ADC3 的数据，依次类推。

    DMA 模式 1 用于三重规则同时模式。

    示例：

    三重规则同时模式：生成 3 个连续的 DMA 请求（每个请求对应一个转换数据项）

    第 1 个请求：ADC_CDR[31:0] = ADC1_DR[15:0]

    第 2 个请求：ADC_CDR[31:0] = ADC2_DR[15:0]

    第 3 个请求：ADC_CDR[31:0] = ADC3_DR[15:0]

    第 4 个请求：ADC_CDR[31:0] = ADC1_DR[15:0]

  – **DMA 模式 2**：每发送一个 DMA 请求（两个数据项可用），就会以字的形式传输表示两个 ADC 转换数据项的两个半字。

    在双重 ADC 模式下，发出第一个请求时会传输 ADC2 和 ADC1 的数据（ADC2 数据占用高位半字，ADC1 数据占用低位半字），依此类推。

    在三重 ADC 模式下，将生成三个 DMA 请求：发出第一个请求时，会传输 ADC2 和 ADC1 的数据（ADC2 数据占用高位半字，ADC1 数据占用低位半字）。发出第二个请求时，会传输 ADC1 和 ADC3 的数据（ADC1 数据占用高位半字，ADC3 数据占用低位半字）。发出第三个请求时，会传输 ADC3 和 ADC2 的数据（ADC3 数据占用高位半字，ADC2 数据占用低位半字），依此类推。

    DMA 模式 2 用于交替模式和规则同时模式（仅适用于双重 ADC 模式）。

    示例：

    a) 双重交替模式：每当有 2 个数据项可用时，就会生成一个 DMA 请求：

    第 1 个请求：ADC_CDR[31:0] = ADC2_DR[15:0] | ADC1_DR[15:0]

    第 2 个请求：ADC_CDR[31:0] = ADC2_DR[15:0] | ADC1_DR[15:0]

    b) 三重交替模式：每当有 2 个数据项可用时，就会生成一个 DMA 请求

    第 1 个请求：ADC_CDR[31:0] = ADC2_DR[15:0] | ADC1_DR[15:0]

    第 2 个请求：ADC_CDR[31:0] = ADC1_DR[15:0] | ADC3_DR[15:0]

    第 3 个请求：ADC_CDR[31:0] = ADC3_DR[15:0] | ADC2_DR[15:0]

    第 4 个请求：ADC_CDR[31:0] = ADC2_DR[15:0] | ADC1_DR[15:0]

  – **DMA 模式 3**：此模式与 DMA 模式 2 相似。唯一的区别是：在这种模式下，每发送一个 DMA 请求（两个数据项可用），就会以半字的形式传输表示两个 ADC 转换数据项的两个字节。此模式下的数据传输顺序与 DMA 模式 2 相似。

    DMA 模式 3 用于分辨率为 6 位和 8 位时的交替模式。

示例：

a) 双重交替模式：每当有 2 个数据项可用时，就会生成一个 DMA 请求
第 1 个请求：ADC_CDR[15:0] = ADC2_DR[7:0] | ADC1_DR[7:0]
第 2 个请求：ADC_CDR[15:0] = ADC2_DR[7:0] | ADC1_DR[7:0]

b) 三重交替模式：每当有 2 个数据项可用时，就会生成一个 DMA 请求
第 1 个请求：ADC_CDR[15:0] = ADC2_DR[7:0] | ADC1_DR7:0]
第 2 个请求：ADC_CDR[15:0] = ADC1_DR[7:0] | ADC3_DR[15:0]
第 3 个请求：ADC_CDR[15:0] = ADC3_DR[7:0] | ADC2_DR[7:0]
第 4 个请求：ADC_CDR[15:0] = ADC2_DR[7:0] | ADC1_DR7:0]

溢出检测：如果在其中一个相关的 ADC（双重和三重模式下的 ADC1 和 ADC2，仅有三重模式时的 ADC3）上检测到溢出，则不再发出 DMA 请求，以确保传输到 RAM 的所有数据都有效。对于与某个 ADC 对应的 EOC 位，有时可能会因为此 ADC 的数据寄存器包含有效数据而保持置 1。

11.9.1 注入同时模式

此模式可转换注入通道组。外部触发源来自 ADC1 的注入组多路复用器（通过 ADC1_CR2 寄存器中的 JEXTSEL[3:0] 位进行选择）。同时触发可用于 ADC2 和 ADC3。

注意： *不要在两个/三个 ADC 上转换同一通道（转换同一通道时，不允许两个/三个 ADC 采样时间重叠）。*

在同时模式下，必须使用同一长度来转换序列，或必须确保触发之间的间隔长于 2 个序列（双重 ADC 模式）/3 个序列（三重 ADC 模式）中的较长时间。否则，当序列较长的 ADC 完成上一次转换时，序列较短的 ADC 可能重新开始转换。

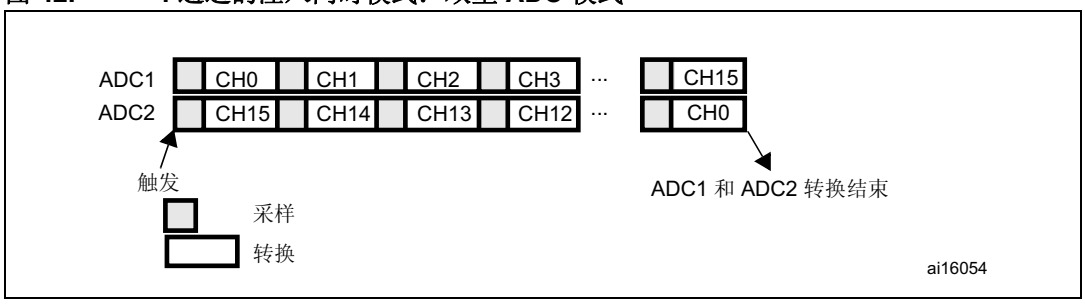
规则转换可在一个或所有 ADC 上执行。这种情况下，它们彼此之间都是独立的，而且会在出现注入事件时中断。它们会在注入转换组结束时恢复转换。

双重 ADC 模式

在 ADC1 或 ADC2 转换事件结束时：

- 转换的数据会存储在各个 ADC 接口的 ADC_JDRx 寄存器中。
- 当 ADC1/ADC2 的注入通道全部完成转换后，会生成一个 JEOC 中断（如果已在两个 ADC 接口中的一个接口上使能）。

图 42. 4 通道的注入同时模式：双重 ADC 模式

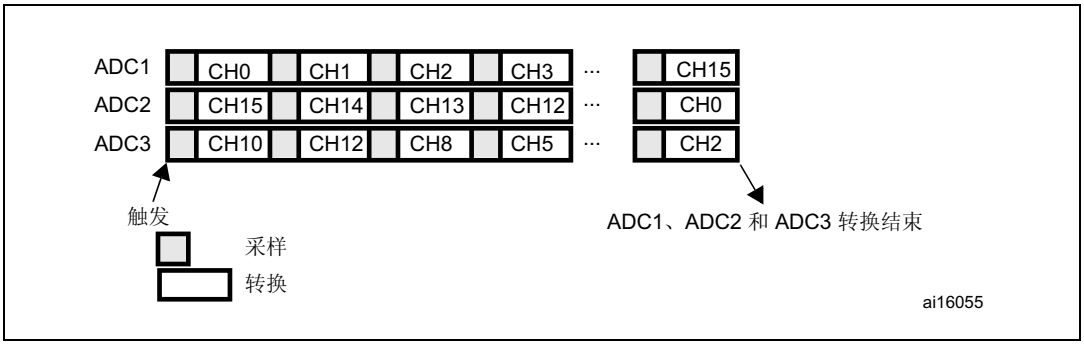


三重 ADC 模式

在 ADC1、ADC2 或 ADC3 转换事件结束时：

- 转换的数据会存储在各个 ADC 接口的 ADC_JDRx 寄存器中。
- 当 ADC1/ADC2/ADC3 的注入通道全部完成转换后，会生成一个 JEOC 中断（如果已在三个 ADC 接口中的一个接口上使能）。

图 43. 4 通道的注入同时模式：三重 ADC 模式



11.9.2 规则同时模式

此模式可用于规则通道组。外部触发源来自 ADC1 的规则组多路复用器（通过 ADC1_CR2 寄存器中的 EXTSEL[3:0] 位进行选择）。同时触发可用于 ADC2 和 ADC3。

注意： *不要在两个/三个 ADC 上转换同一通道（转换同一通道时，不允许两个/三个 ADC 采样时间重叠）。*

在规则同时模式下，必须使用同一长度来转换序列，或必须确保触发之间的间隔长于 2 个序列（双重 ADC 模式）/3 个序列（三重 ADC 模式）中的较长转换时间。否则，当序列较长的 ADC 完成上一次转换时，序列较短的 ADC 可能重新开始转换。

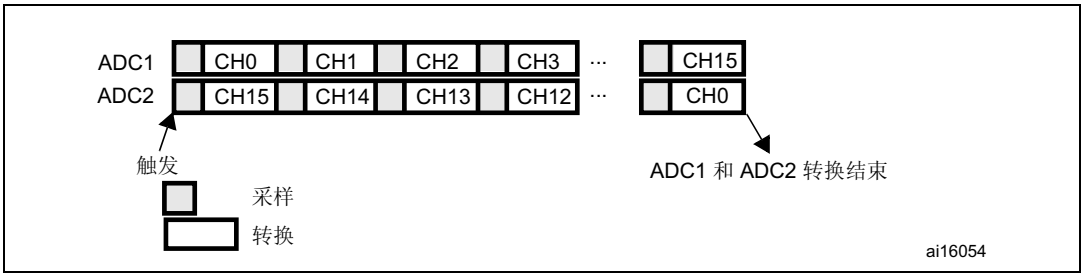
必须禁止注入转换。

双重 ADC 模式

在 ADC1 或 ADC2 转换事件结束时：

- 会生成一个 32 位 DMA 传输请求（如果 ADC_CCR 寄存器中的 DMA[1:0] 位等于 0b10）。此请求会将存储在 ADC_CDR 32 位寄存器高位半字中的 ADC2 转换数据传输到 SRAM，然后将存储在 ADC_CCR 低位半字中的 ADC1 转换数据传输到 SRAM。
- 当 ADC1/ADC2 的规则通道全部完成转换后，会生成一个 EOC 中断（如果已在两个 ADC 接口中的一个接口上使能）。

图 44. 16 通道的规则同时模式：双重 ADC 模式

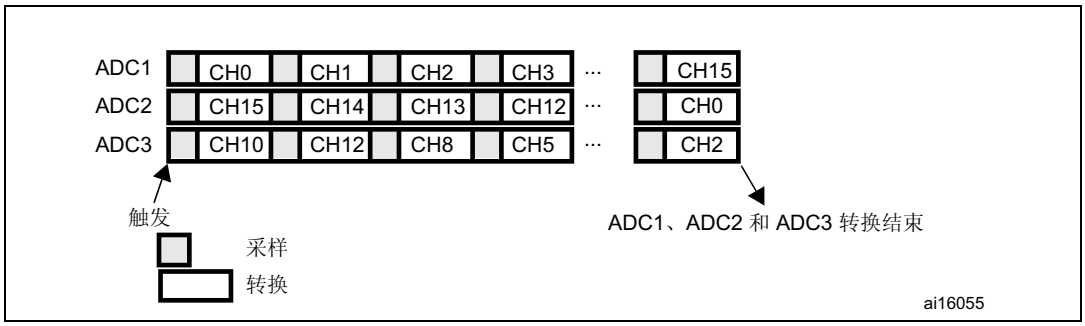


三重 ADC 模式

在 ADC1、ADC2 或 ADC3 转换事件结束时：

- 会生成三个 32 位 DMA 传输请求（如果 ADC_CCR 寄存器中的 DMA[1:0] 位等于 0b01）。之后会发生三次从 ADC_CDR 32 位寄存器到 SRAM 的传输：首先传输 ADC1 转换数据，然后是 ADC2 转换数据，最后是 ADC3 转换数据。每次出现新的三阶段转换时都会重复这一过程。
- 当 ADC1/ADC2/ADC3 的规则通道全部完成转换后，会生成一个 EOC 中断（如果已在三个 ADC 接口中的一个接口上使能）。

图 45. 16 通道的规则同时模式：三重 ADC 模式



11.9.3 交替模式

此模式只能用于规则组（通常为一个通道）。外部触发源来自 ADC1 的规则通道多路复用器。

双重 ADC 模式

出现外部触发之后：

- ADC1 立即启动
- 经过几个 ADC 时钟周期延迟后 ADC2 启动

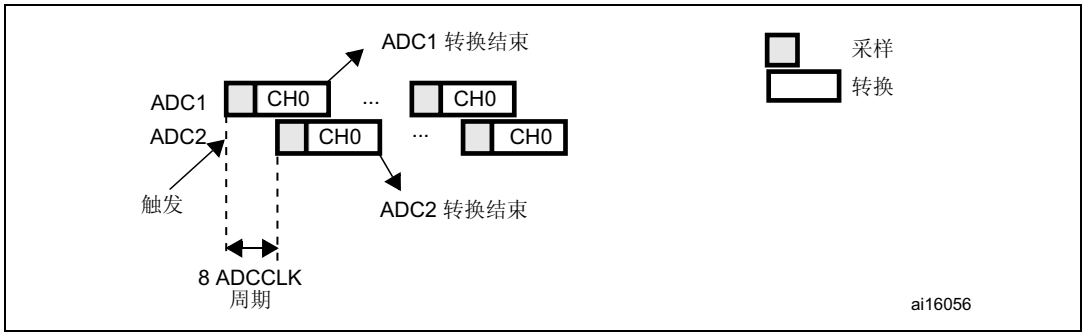
交替模式下 2 个转换之间的最小延迟通过 ADC_CCR 寄存器中的 DELAY 位进行配置。但是，如果某个 ADC 的互补 ADC 仍在对其输入进行采样，则该 ADC 无法启动转换（在给定时间内，只有一个 ADC 能够对输入信号采样）。在这种情况下，延迟时间为采样时间 + 2 个 ADC 时钟周期。例如，如果两个 ADC 的 DELAY = 5 个时钟周期，且采样时间为 15 个时钟周期，则 ADC1 和 ADC2 之间的转换延迟为 17 个时钟周期。

如果 ADC1 和 ADC2 上的 CONT 位均置 1，则这两个 ADC 所选规则通道会连续进行转换。

注意： *如果转换序列中断（例如 DMA 传输结束时），则必须首先通过在独立模式下进行配置来将多重 ADC 定序器复位（位 DUAL[4:0] = 00000），然后才可以对交替模式进行编程。*

ADC2 生成一个 EOC 中断之后（如果已通过 EOCIE 位使能），会生成一个 32 位 DMA 传输请求（如果 ADC_CCR 寄存器中的 DMA[1:0] 位等于 0b10）。此请求首先会将存储在 ADC_CDR 32 位寄存器高位半字中的 ADC2 转换数据传输到 SRAM，然后将存储在寄存器低位半字中的 ADC1 转换数据传输到 SRAM。

图 46. 连续转换模式下 1 通道的交替模式：双重 ADC 模式



三重 ADC 模式

出现外部触发之后：

- ADC1 立即启动
- 经过几个 ADC 时钟周期延迟后 ADC2 启动
- 在 ADC2 转换经过几个 ADC 时钟周期的延迟后 ADC3 启动

交替模式下 2 个转换之间的最小延迟通过 ADC_CCR 寄存器中的 DELAY 位进行配置。但是，如果某个 ADC 的互补 ADC 仍在对其输入进行采样，则该 ADC 无法启动转换（在给定时间内，只有一个 ADC 能够对输入信号采样）。在这种情况下，延迟时间为采样时间 + 2 个 ADC 时钟周期。例如，如果这三个 ADC 的 DELAY = 5 个时钟周期，且采样时间为 15 个时钟周期，则 ADC1、ADC2 和 ADC3 之间的转换延迟为 17 个时钟周期。

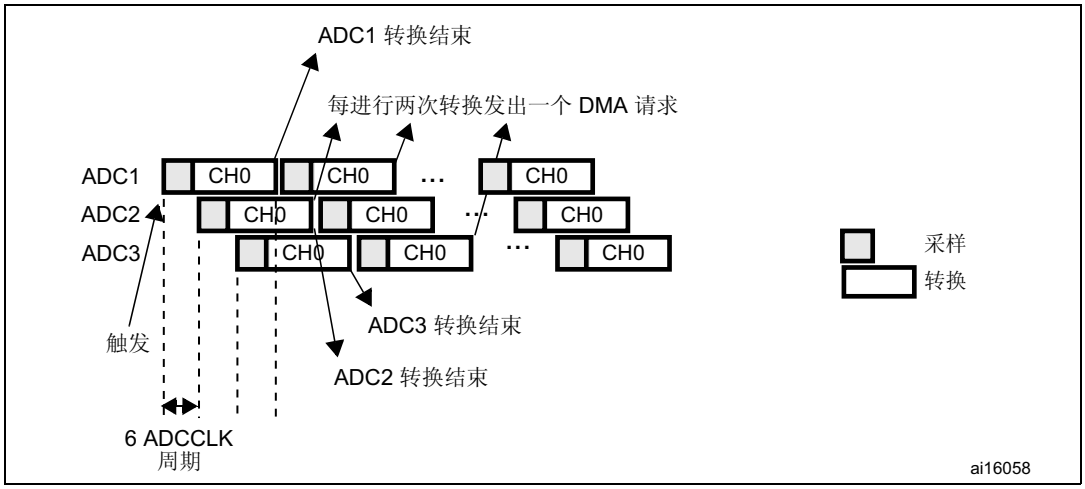
如果 ADC1、ADC2 和 ADC3 上的 CONT 位均置 1，则这些 ADC 所选规则通道会连续进行转换。

注意： *如果转换序列中断（例如 DMA 传输结束时），则必须首先通过在独立模式下进行配置来将多重 ADC 定序器复位（位 DUAL[4:0] = 00000），然后才可以对交替模式进行编程。*

在此模式下，每当出现 2 个可用数据项时，就会生成一个 DMA 传输请求（如果 ADC_CCR 寄存器中的 DMA[1:0] 位等于 0b10）。此请求首先会将存储在 ADC_CDR 32 位寄存器低位半字中的第一批转换数据传输到 SRAM，然后将存储在 ADC_CDR 高位半字中的第二批转换数据传输到 SRAM。具体顺序如下：

- 第 1 个请求：ADC_CDR[31:0] = ADC2_DR[15:0] | ADC1_DR[15:0]
- 第 2 个请求：ADC_CDR[31:0] = ADC1_DR[15:0] | ADC3_DR[15:0]
- 第 3 个请求：ADC_CDR[31:0] = ADC3_DR[15:0] | ADC2_DR[15:0]
- 第 4 个请求：ADC_CDR[31:0] = ADC2_DR[15:0] | ADC1_DR[15:0], ...

图 47. 连续转换模式下 1 通道的交替模式：三重 ADC 模式



11.9.4 交替触发模式

此模式只能用于注入组。外部触发源来自 ADC1 的注入组多路复用器。

注意： *规则转换可在一个或所有 ADC 上使能。在这种情况下，规则转换彼此之间是独立的。当 ADC 必须执行注入转换时，会中断规则转换。它会在注入转换完成后恢复。*

如果转换序列中断（例如 DMA 传输结束时），则必须首先通过在独立模式下进行配置来将多重 ADC 定序器复位（位 DUAL[4:0] = 00000），然后才可以对交替模式进行编程。

两次触发事件之间的时间间隔必须大于或等于 1 个 ADC 时钟周期。同一个 ADC 上可启动转换的两个触发事件之间的最小时间间隔与单个 ADC 模式下相同。

双重 ADC 模式

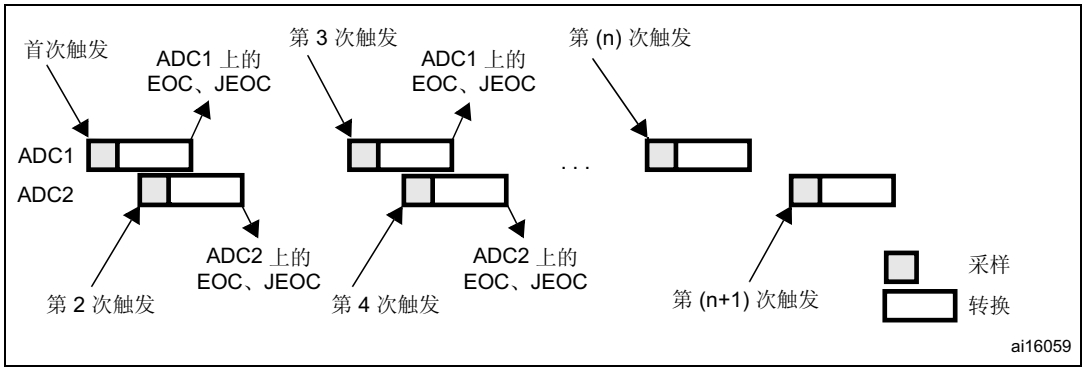
- 发生第一次触发时，将转换 ADC1 中注入组的所有通道
- 发生第二次触发时，将转换 ADC2 中注入组的所有通道
- 以此类推

当组中的所有注入 ADC1 通道都转换完成后，会生成一个 JEOC 中断（如果已使能）。

当组中的所有注入 ADC2 通道都转换完成后，会生成一个 JEOC 中断（如果已使能）。

如果在组中的所有注入通道都完成转换后出现另一个外部触发，则可通过转换组中的注入 ADC1 通道来重新启动交替触发过程。

图 48. 交替触发：各个 ADC 的注入组



如果使能 ADC1 和 ADC2 的注入不连续采样模式：

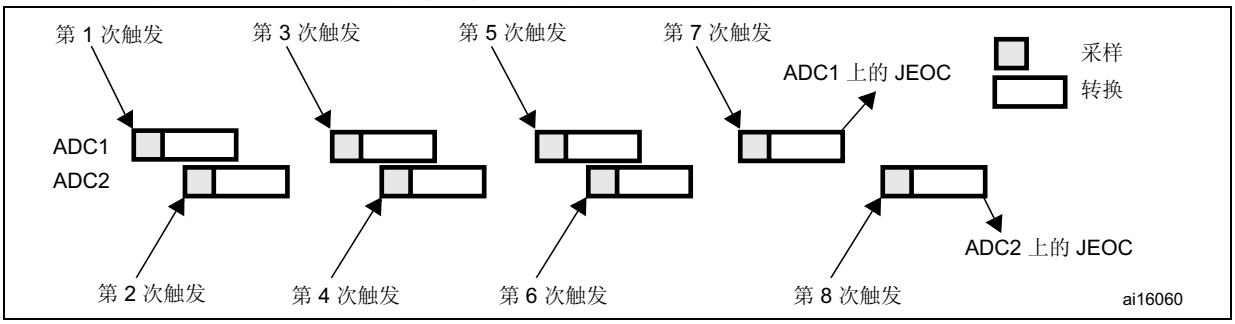
- 发生第一次触发时，将转换第一个注入 ADC1 通道
- 发生第二次触发时，将转换第一个注入 ADC2 通道
- 以此类推

当组中的所有注入 ADC1 通道都转换完成后，会生成一个 JEOC 中断（如果已使能）。

当组中的所有注入 ADC2 通道都转换完成后，会生成一个 JEOC 中断（如果已使能）。

如果注入组中的所有通道都完成转换后出现另一个外部触发，则会重新启动交替触发过程。

图 49. 交替触发：不连续采样模式下的 4 个注入通道（各个 ADC）



三重 ADC 模式

- 发生第一次触发时，将转换 ADC1 中注入组的所有通道。
- 发生第二次触发时，将转换 ADC2 中注入组的所有通道。
- 发生第三次触发时，将转换 ADC3 中注入组的所有通道。
- 以此类推

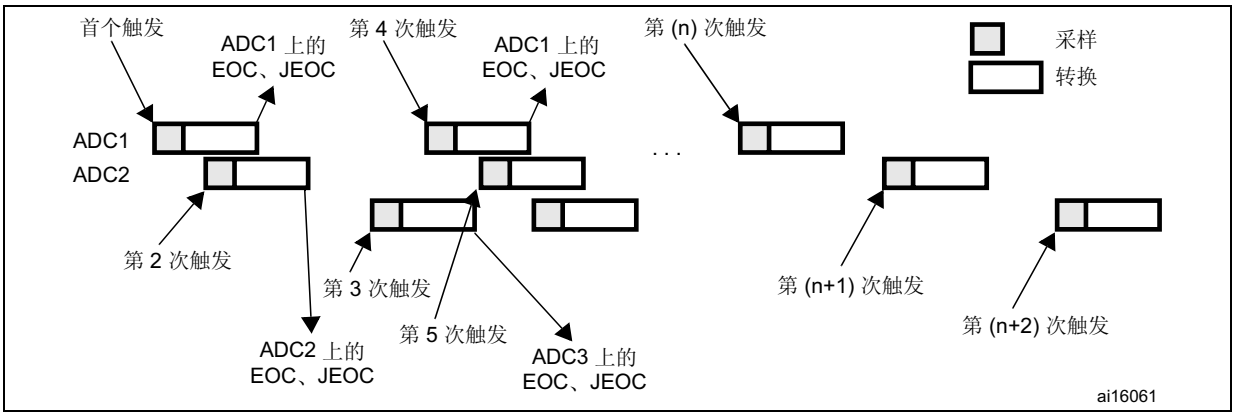
当组中的所有注入 ADC1 通道都转换完成后，会生成一个 JEOC 中断（如果已使能）。

当组中的所有注入 ADC2 通道都转换完成后，会生成一个 JEOC 中断（如果已使能）。

当组中的所有注入 ADC3 通道都转换完成后，会生成一个 JEOC 中断（如果已使能）。

如果在组中的所有注入通道都完成转换后出现另一个外部触发，则可通过转换组中的注入 ADC1 通道来重新启动交替触发过程。

图 50. 交替触发：各个 ADC 的注入组



11.9.5 混合型规则/注入同时模式

可以中断规则组的同时转换，然后开始注入组的同时转换。

注意： *在混合型规则/注入同时模式下，必须使用同一长度来转换序列，或必须确保触发之间的间隔长于 2 个序列（双重 ADC 模式）/3 个序列（三重 ADC 模式）中的较长转换时间。否则，当序列较长的 ADC 完成上一次转换时，序列较短的 ADC 可能重新开始转换。*

11.9.6 规则同时 + 交替触发组合模式

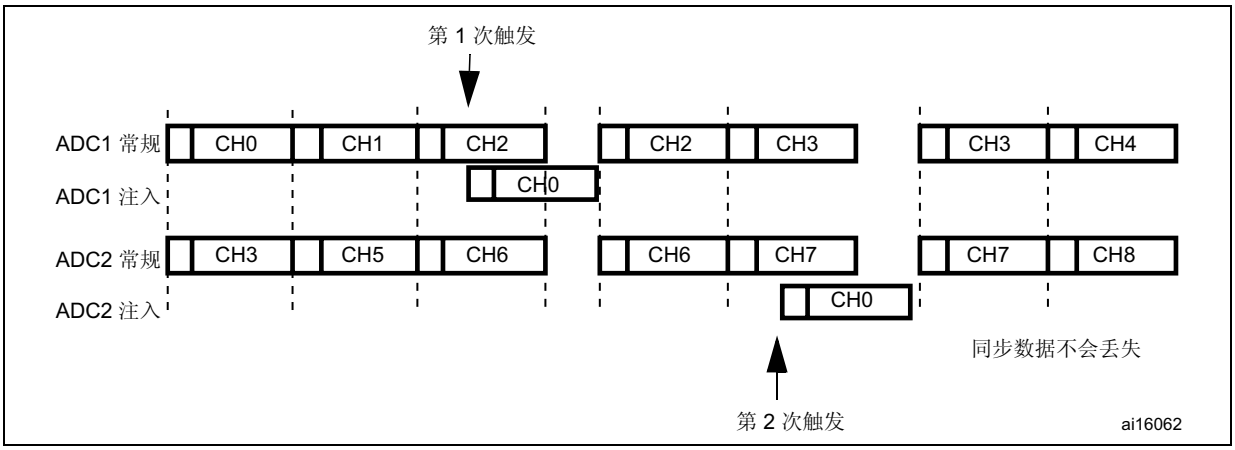
可以中断规则组的同时转换，然后开始注入组的交替触发转换。*图 51* 说明了交替触发模式中断同时规则转换的行为。

注入事件后立即开始注入交替转换。当规则转换处于运行状态时，为确保在注入转换后实现同步，所有的（主/从）ADC 规则转换均将停止，并会在注入转换结束时得以恢复运行。

注意： *在规则同时 + 交替触发组合模式下，必须使用同一长度来转换序列，或必须确保触发之间的间隔长于 2 个序列（双重 ADC 模式）/3 个序列（三重 ADC 模式）的长转换时间。否则，当序列较长的 ADC 完成上一次转换时，序列较短的 ADC 可能重新开始转换。*

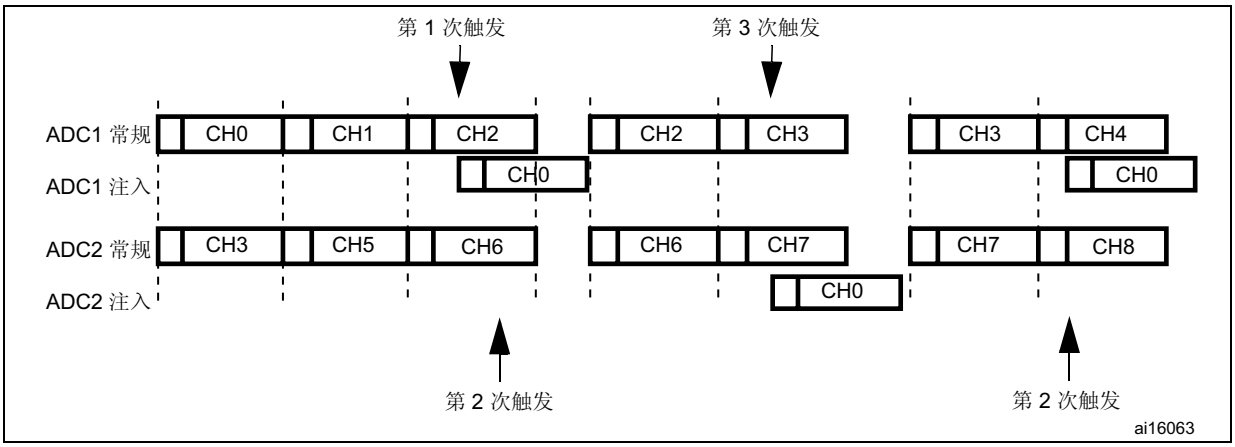
如果转换序列中断（例如 DMA 传输结束时），则必须首先通过在独立模式下进行配置来将多重 ADC 定序器复位（位 DUAL[4:0] = 00000），然后才可以对交替模式进行编程。

图 51. 交替 + 规则同时



在已导致规则转换中断的注入转换期间出现的触发将被忽略。*图 52* 说明了这种情况下的行为（第 2 次触发被忽略）。

图 52. 在注入转换期间出现的触发事件



11.10 温度传感器

温度传感器可用于测量器件的环境温度 (T_A)。

- 对于 STM32F40x 和 STM32F41x 器件，温度传感器内部连接到 ADC1_IN16 通道，而 ADC1 用于将传感器输出电压转换为数字值。
- 对于 STM32F42x 和 STM32F43x 器件，温度传感器内部连接到与 VBAT 共用的输入通道 ADC1_IN18：ADC1_IN18 用于将传感器输出电压或 VBAT 转换为数字值。一次只能选择一个转换（温度传感器或 VBAT）。同时设置了温度传感器和 VBAT 转换时，将只进行 VBAT 转换。

图 53 显示了温度传感器框图。

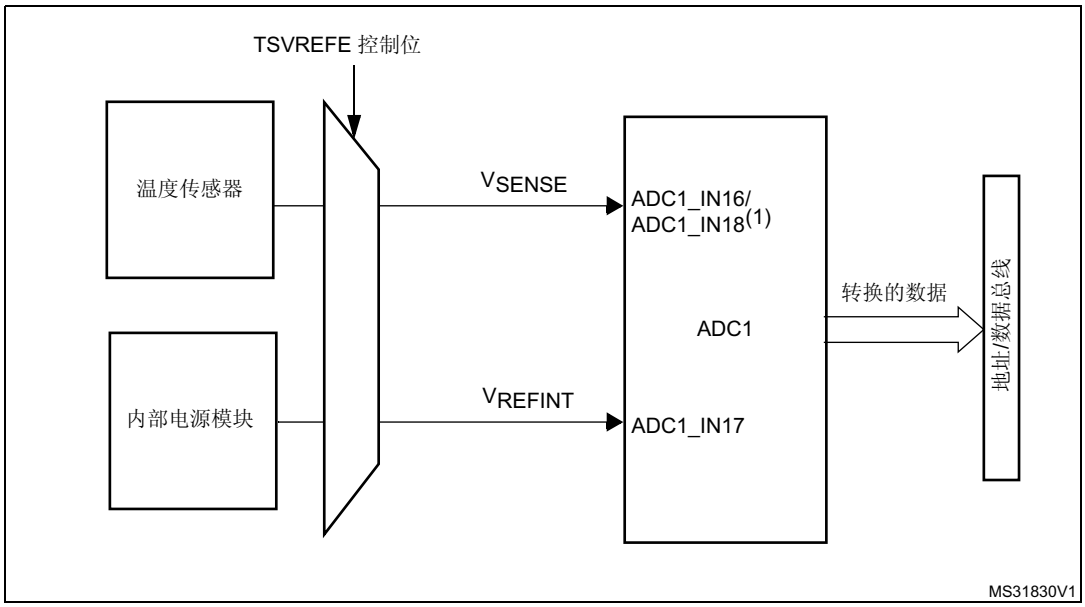
不使用时可将传感器置于掉电模式。

注意： *必须将 TSVREFE 位置 1 才能同时对两个通道进行转换。ADC1_IN16 或 ADC1_IN18（温度传感器）和 ADC1_IN17 (VREFINT)。*

主要特性

- 支持的温度范围：–40 °C 到 125 °C
- 精度：±1.5 °C

图 53. 温度传感器和 V_{REFINT} 通道框图



1. V_{SENSE} 是至 ADC1_IN16 的输入（对于 STM32F40x 和 STM32F41x 器件），也是 ADC1_IN18 的输入（对于 STM32F42x 和 STM32F43x 器件）。

读取温度

要使用传感器，请执行以下操作：

3. 选择 ADC1_IN16 或 ADC1_IN18 输入通道。
4. 选择一个采样时间，该采样时间要大于数据手册中所指定的最低采样时间。
5. 在 ADC_CCR 寄存器中将 TSVREFE 位置 1，以便将温度传感器从掉电模式中唤醒。
6. 通过将 SWSTART 位置 1（或通过外部触发）开始 ADC 转换
7. 读取 ADC 数据寄存器中生成的 V_{SENSE} 数据
8. 使用以下公式计算温度：

   温度（单位为 °C）= {(V_{SENSE} – V_{25}) / Avg_Slope} + 25

   其中：

   – V_{25} = 25 °C 时的 V_{SENSE} 值
   – Avg_Slope = 温度与 V_{SENSE} 曲线的平均斜率（以 mV/°C 或 µV/°C 表示）

   有关 V_{25} 和 Avg_Slope 实际值的相关信息，请参见数据手册中的电气特性一节。

注意： *传感器从掉电模式中唤醒需要一个启动时间，启动时间过后其才能正确输出 V_{SENSE}。ADC 在上电后同样需要一个启动时间，因此，为尽可能减少延迟间，应同时将 ADON 和 TSVREFE 位置 1。*

温度传感器的输出电压随温度线性变化。由于工艺不同，该线性函数的偏移量取决于各个芯片（芯片之间的温度变化可达 45 °C）。

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情况。如果需要读取精确温度，则应使用外部温度传感器。

11.11 电池充电监视

ADC_CCR 寄存器中的 VBATE 位用于切换到电池电压。由于 V_{BAT} 电压可能高于 V_{DDA}，因此 V_{BAT} 引脚需要内部连接到桥接分配器，以确保 ADC 正确运行。

设置 VBATE 后，桥接器会自动使能以进行以下连接：

- 在 STM32F40xx 和 STM32F41xx 器件上将 VBAT/2 连接到 ADC1_IN18 输入通道
- 在 STM32F42xx 和 STM32F43xx 器件上将 VBAT/4 连接到 ADC1_IN18 输入通道

注意： *VBAT 和温度传感器连接到同一 ADC 内部通道 (ADC1_IN18)。一次只能选择一个转换（温度传感器或 VBAT）。同时使能两个转换时，将只进行 VBAT 转换。*

11.12 ADC 中断

当模拟看门狗状态位和溢出状态位分别置 1 时，规则组和注入组在转换结束时可能会产生中断。可以使用单独的中断使能位以实现灵活性。

ADC_SR 寄存器中存在另外两个标志，但这两个标志不存在中断相关性：

- JSTRT（开始转换注入组的通道）
- STRT（开始转换规则组的通道）

表 53. ADC 中断

中断事件	事件标志	使能控制位
结束规则组的转换	EOC	EOCIE
结束注入组的转换	JEOC	JEOCIE
模拟看门狗状态位置 1	AWD	AWDIE
溢出 (Overrun)	OVR	OVRIE

11.13 ADC 寄存器

有关寄存器说明中使用的缩写，请参见*第 47 页的第 1.1 节*。

必须在字级别（32 位）对外设寄存器执行写入操作。而读访问可支持字节（8 位）、半字（16 位）或字（32 位）。

11.13.1 ADC 状态寄存器 (ADC_SR)

ADC status register

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
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved										OVR	STRT	JSTRT	JEOC	EOC	AWD
										rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0

位 31:6 保留，必须保持复位值。

位 5 **OVR**：溢出 (Overrun)

数据丢失时，硬件将该位置 1（在单一模式或双重/三重模式下）。但需要通过软件清零。溢出检测仅在 DMA = 1 或 EOCS = 1 时使能。

0：未发生溢出

1：发生溢出

位 4 **STRT**：规则通道开始标志 (Regular channel start flag)

规则通道转换开始时，硬件将该位置 1。但需要通过软件清零。

0：未开始规则通道转换

1：已开始规则通道转换

位 3 **JSTRT**：注入通道开始标志 (Injected channel start flag)

注入组转换开始时，硬件将该位置 1。但需要通过软件清零。

0：未开始注入组转换

1：已开始注入组转换

位 2 **JEOC**：注入通道转换结束 (Injected channel end of conversion)

组内所有注入通道转换结束时，硬件将该位置 1。但需要通过软件清零。

0：转换未完成

1：转换已完成

位 1 **EOC**：规则通道转换结束 (Regular channel end of conversion)

规则组通道转换结束后，硬件将该位置 1。通过软件或通过读取 ADC_DR 寄存器将该位清零。

0：转换未完成 (EOCS=0) 或转换序列未完成 (EOCS=1)

1：转换已完成 (EOCS=0) 或转换序列已完成 (EOCS=1)

位 0 **AWD**：模拟看门狗标志 (Analog watchdog flag)

当转换电压超过在 ADC_LTR 和 ADC_HTR 寄存器中编程的值时，硬件将该位置 1。但需要通过软件清零。

0：未发生模拟看门狗事件

1：发生模拟看门狗事件

11.13.2 ADC 控制寄存器 1 (ADC_CR1)

ADC control register 1

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
Reserved					OVRIE	RES		AWDEN	JAWDEN	Reserved					
					rw	rw	rw	rw	rw						

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DISCNUM[2:0]			JDISCEN	DISC EN	JAUTO	AWDSGL	SCAN	JEOCIE	AWDIE	EOCIE	AWDCH[4:0]				
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:27 保留，必须保持复位值。

位 26 **OVRIE**：溢出中断使能 (Overrun interrupt enable)
通过软件将该位置 1 和清零可使能/禁止溢出中断。
0：禁止溢出中断
1：使能溢出中断。OVR 位置 1 时产生中断。

位 25:24 **RES[1:0]**：分辨率 (Resolution)
通过软件写入这些位可选择转换的分辨率。
00：12 位（15 ADCCLK 周期）
01：10 位（13 ADCCLK 周期）
10：8 位（11 ADCCLK 周期）
11：6 位（9 ADCCLK 周期）

位 23 **AWDEN**：规则通道上的模拟看门狗使能 (Analog watchdog enable on regular channels)
此位由软件置 1 和清零。
0：在规则通道上禁止模拟看门狗
1：在规则通道上使能模拟看门狗

位 22 **JAWDEN**：注入通道上的模拟看门狗使能 (Analog watchdog enable on injected channels)
此位由软件置 1 和清零。
0：在注入通道上禁止模拟看门狗
1：在注入通道上使能模拟看门狗

位 21:16 保留，必须保持复位值。

位 15:13 **DISCNUM[2:0]**：不连续采样模式通道计数 (Discontinuous mode channel count)
软件将写入这些位，用于定义在接收到外部触发后于不连续采样模式下转换的规则通道数。
000：1 个通道
001：2 个通道
...
111：8 个通道

位 12 **JDISCEN**：注入通道的不连续采样模式 (Discontinuous mode on injected channels)
通过软件将该位置 1 和清零可使能/禁止注入通道的不连续采样模式。
0：禁止注入通道的不连续采样模式
1：使能注入通道的不连续采样模式

位 11 **DISCEN**：规则通道的不连续采样模式 (Discontinuous mode on regular channels)
通过软件将该位置 1 和清零可使能/禁止规则通道的不连续采样模式。
0：禁止规则通道的不连续采样模式
1：使能规则通道的不连续采样模式

位 10 **JAUTO**：注入组自动转换 (Automatic injected group conversion)
通过软件将该位置 1 和清零可在规则组转换后分别使能/禁止注入组自动转换。
0：禁止注入组自动转换
1：使能注入组自动转换

位 9 **AWDSGL**：在扫描模式下使能单一通道上的看门狗 (Enable the watchdog on a single channel in scan mode)
通过软件将该位置 1 和清零可分别使能/禁止通过 AWDCH[4:0] 位确定的通道上的模拟看门狗。
0：在所有通道上使能模拟看门狗
1：在单一通道上使能模拟看门狗

位 8 **SCAN**：扫描模式 (Scan mode)

通过软件将该位置 1 和清零可使能/禁止扫描模式。在扫描模式下，转换通过 ADC_SQRx 或 ADC_JSQRx 寄存器选择的输入。

0：禁止扫描模式

1：使能扫描模式

注意：EOCIE 位置 1 时将生成 EOC 中断：

- *如果 EOCS 位清零，在每个规则组序列转换结束时*
- *如果 EOCS 位置 1，在每个规则通道转换结束时*

注意：JEOCIE 位置 1 时，JEOC 中断仅在最后一个通道转换结束时生成。

位 7 **JEOCIE**：注入通道的中断使能 (Interrupt enable for injected channels)

通过软件将该位置 1 和清零可使能/禁止注入通道的转换结束中断。

0：禁止 JEOC 中断

1：使能 JEOC 中断。JEOC 位置 1 时产生中断。

位 6 **AWDIE**：模拟看门狗中断使能 (Analog watchdog interrupt enable)

通过软件将该位置 1 和清零可使能/禁止模拟看门狗中断。

0：禁止模拟看门狗中断

1：使能模拟看门狗中断

位 5 **EOCIE**：EOC 中断使能 (Interrupt enable for EOC)

通过软件将该位置 1 和清零可使能/禁止转换结束中断。

0：禁止 EOC 中断

1：使能 EOC 中断EOC 位置 1 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)

这些位将由软件置 1 和清零。它们用于选择由模拟看门狗监控的输入通道。

*注意：*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
10010：ADC 模拟输入通道 18
保留其它值

11.13.3 ADC 控制寄存器 2 (ADC_CR2)

ADC control register 2

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
reserved	SWSTART	EXTEN		EXTSEL[3:0]				reserved	JSWSTART	JEXTEN		JEXTSEL[3:0]			
	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
reserved				ALIGN	EOCS	DDS	DMA	Reserved						CONT	ADON
				rw	rw	rw	rw							rw	rw

位 31 保留，必须保持复位值。

位 30 **SWSTART**：开始转换规则通道 (Start conversion of regular channels)

通过软件将该位置 1 可开始转换，而硬件会在转换开始后将该位清零。

0：复位状态

1：开始转换规则通道

注意：该位只能在 ADON = 1 时置 1，否则不会启动转换。

位 29:28 **EXTEN**：规则通道的外部触发使能 (External trigger enable for regular channels)

通过软件将这些位置 1 和清零可选择外部触发极性和使能规则组的触发。

00：禁止触发检测

01：上升沿上的触发检测

10：下降沿上的触发检测

11：上升沿和下降沿上的触发检测

位 27:24 **EXTSEL[3:0]**：为规则组选择外部事件 (External event select for regular group)

这些位可选择用于触发规则组转换的外部事件。

0000：定时器 1 CC1 事件

0001：定时器 1 CC2 事件

0010：定时器 1 CC3 事件

0011：定时器 2 CC2 事件

0100：定时器 2 CC3 事件

0101：定时器 2 CC4 事件

0110：定时器 2 TRGO 事件

0111：定时器 3 CC1 事件

1000：定时器 3 TRGO 事件

1001：定时器 4 CC4 事件

1010：定时器 5 CC1 事件

1011：定时器 5 CC2 事件

1100：定时器 5 CC3 事件

1101：定时器 8 CC1 事件

1110：定时器 8 TRGO 事件

1111：EXTI 线 11

位 23 保留，必须保持复位值。

位 22 **JSWSTART**：开始转换注入通道 (Start conversion of injected channels)

转换开始后，软件将该位置 1，而硬件将该位清零。

0：复位状态

1：开始转换注入通道

注意：该位只能在 ADON = 1 时置 1，否则不会启动转换。

位 21:20 **JEXTEN**：注入通道的外部触发使能 (External trigger enable for injected channels)

通过软件将这些位置 1 和清零可选择外部触发极性和使能注入组的触发。

00：禁止触发检测

01：上升沿上的触发检测

10：下降沿上的触发检测

11：上升沿和下降沿上的触发检测

位 19:16 **JEXTSEL[3:0]**：为注入组选择外部事件 (External event select for injected group)

这些位可选择用于触发注入组转换的外部事件。

0000：定时器 1 CC4 事件
0001：定时器 1 TRGO 事件
0010：定时器 2 CC1 事件
0011：定时器 2 TRGO 事件
0100：定时器 3 CC2 事件
0101：定时器 3 CC4 事件
0110：定时器 4 CC1 事件
0111：定时器 4 CC2 事件
1000：定时器 4 CC3 事件
1001：定时器 4 TRGO 事件
1010：定时器 5 CC4 事件
1011：定时器 5 TRGO 事件
1100：定时器 8 CC2 事件
1101：定时器 8 CC3 事件
1110：定时器 8 CC4 事件
1111：EXTI 线 15

位 15:12 保留，必须保持复位值。

位 11 **ALIGN**：数据对齐 (Data alignment)

此位由软件置 1 和清零。请参见*图 38* 和*图 39*。

0：右对齐
1：左对齐

位 10 **EOCS**：结束转换选择 (End of conversion selection)

此位由软件置 1 和清零。

0：在每个规则转换序列结束时将 EOC 位置 1。溢出检测仅在 DMA=1 时使能。
1：在每个规则转换结束时将 EOC 位置 1。使能溢出检测。

位 9 **DDS**：DMA 禁止选择（对于单一 ADC 模式） (DMA disable selection (for single ADC mode))

此位由软件置 1 和清零。

0：最后一次传输后不发出新的 DMA 请求（在 DMA 控制器中进行配置）
1：只要发生数据转换且 DMA = 1，便会发出 DAM 请求

位 8 **DMA**：直接存储器访问模式（对于单一 ADC 模式）(Direct memory access mode (for single ADC mode))

此位由软件置 1 和清零。有关详细信息，请参见 DMA 控制器一章。

0：禁止 DMA 模式
1：使能 DMA 模式

位 7:2 保留，必须保持复位值。

位 1 **CONT**：连续转换 (Continuous conversion)

此位由软件置 1 和清零。该位置 1 时，转换将持续进行，直到该位清零。

0：单次转换模式
1：连续转换模式

位 0 **ADON**：A/D 转换器开启 / 关闭 (A/D Converter ON / OFF)

此位由软件置 1 和清零。

注意： 0：禁止 ADC 转换并转至掉电模式
1：使能 ADC

11.13.4 ADC 采样时间寄存器 1 (ADC_SMPR1)

ADC sample time register 1

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
Reserved					SMP18[2:0]			SMP17[2:0]			SMP16[2:0]			SMP15[2:1]	
					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
SMP15_0	SMP14[2:0]			SMP13[2:0]			SMP12[2:0]			SMP11[2:0]			SMP10[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:27 保留，必须保持复位值。

位 26:0 **SMPx[2:0]：**通道 X 采样时间选择 (Channel x sampling time selection)

通过软件写入这些位可分别为各个通道选择采样时间。在采样周期期间，通道选择位必须保持不变。

*注意：*000：3 个周期
001：15 个周期
010：28 个周期
011：56 个周期
100：84 个周期
101：112 个周期
110：144 个周期
111：480 个周期

11.13.5 ADC 采样时间寄存器 2 (ADC_SMPR2)

ADC sample time register 2

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
Reserved		SMP9[2:0]			SMP8[2:0]			SMP7[2:0]			SMP6[2:0]			SMP5[2:1]	
		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
SMP 5_0	SMP4[2:0]			SMP3[2:0]			SMP2[2:0]			SMP1[2:0]			SMP0[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:30 保留，必须保持复位值。

位 29:0 **SMPx[2:0]**：通道 X 采样时间选择 (Channel x sampling time selection)
通过软件写入这些位可分别为各个通道选择采样时间。在采样周期期间，通道选择位必须保持不变。

注意：000：3 个周期
001：15 个周期
010：28 个周期
011：56 个周期
100：84 个周期
101：112 个周期
110：144 个周期
111：480 个周期

11.13.6 ADC 注入通道数据偏移寄存器 X (ADC_JOFRx)(x=1..4)

ADC injected channel data offset register x

偏移地址：0x14-0x20

复位值：0x0000 0000

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

位 31:12 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)
通过软件写入这些位可定义在转换注入通道时从原始转换数据中减去的偏移量。可从 ADC_JDRx 寄存器中读取转换结果。

11.13.7 ADC 看门狗高阈值寄存器 (ADC_HTR)

ADC watchdog higher threshold register

偏移地址：0x24

复位值：0x0000 0FFF

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

位 31:12 保留，必须保持复位值。

位 11:0 **HT[11:0]**：模拟看门狗高阈值 (Analog watchdog higher threshold)
通过软件写入这些位可为模拟看门狗定义高阈值。

11.13.8 ADC 看门狗低阈值寄存器 (ADC_LTR)

ADC watchdog lower threshold register

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

位 31:12 保留，必须保持复位值。

位 11:0 **LT[11:0]**：模拟看门狗低阈值 (Analog watchdog lower threshold)
通过软件写入这些位可为模拟看门狗定义低阈值。

11.13.9 ADC 规则序列寄存器 1 (ADC_SQR1)

ADC regular sequence register 1

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
Reserved								L[3:0]				SQ16[4:1]			
								rw	rw	rw	rw	rw	rw	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	rw	rw	rw	rw				rw	rw	rw	rw	rw

位 31:24 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)
通过软件写入这些位可定义规则通道转换序列中的转换总数。
0000：1 次转换
0001：2 次转换
...
1111：16 次转换

位 19:15 **SQ16[4:0]**：规则序列中的第十六次转换 (16th conversion in regular sequence)
通过软件写入这些位，并将通道编号 (0..18) 分配为转换序列中的第十六次转换。

位 14:10 **SQ15[4:0]**：规则序列中的第十五次转换 (15th conversion in regular sequence)

位 9:5 **SQ14[4:0]**：规则序列中的第十四次转换 (14th conversion in regular sequence)

位 4:0 **SQ13[4:0]**：规则序列中的第十三次转换 (13th conversion in regular sequence)

11.13.10 ADC 规则序列寄存器 2 (ADC_SQR2)

ADC regular sequence register 2

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
Reserved		SQ12[4:0]					SQ11[4:0]					SQ10[4:1]			
		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
SQ10_0	SQ9[4:0]						SQ8[4:0]					SQ7[4:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:30 保留，必须保持复位值。

位 29:26 **SQ12[4:0]：**规则序列中的第十二次转换 (12th conversion in regular sequence)
通过软件写入这些位，并将通道编号 (0..18) 分配为序列中的第十二次转换。

位 24:20 **SQ11[4:0]：**规则序列中的第十一次转换 (11th conversion in regular sequence)

位 19:15 **SQ10[4:0]：**规则序列中的第十次转换 (10th conversion in regular sequence)

位 14:10 **SQ9[4:0]：**规则序列中的第九次转换 (9th conversion in regular sequence)

位 9:5 **SQ8[4:0]：**规则序列中的第八次转换 (8th conversion in regular sequence)

位 4:0 **SQ7[4:0]：**规则序列中的第七次转换 (7th conversion in regular sequence)

11.13.11 ADC 规则序列寄存器 3 (ADC_SQR3)

ADC regular sequence register 3

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
Reserved		SQ6[4:0]					SQ5[4:0]					SQ4[4:1]			
		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
SQ4_0	SQ3[4:0]						SQ2[4:0]					SQ1[4:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:30 保留，必须保持复位值。

位 29:25 **SQ6[4:0]：**规则序列中的第六次转换 (6th conversion in regular sequence)
通过软件写入这些位，并将通道编号 (0..18) 分配为序列中的第六次转换。

位 24:20 **SQ5[4:0]：**规则序列中的第五次转换 (5th conversion in regular sequence)

位 19:15 **SQ4[4:0]：**规则序列中的第四次转换 (4th conversion in regular sequence)

位 14:10 **SQ3[4:0]：**规则序列中的第三次转换 (3rd conversion in regular sequence)

位 9:5 **SQ2[4:0]：**规则序列中的第二次转换 (2nd conversion in regular sequence)

位 4:0 **SQ1[4:0]：**规则序列中的第一次转换 (1st conversion in regular sequence)

11.13.12 ADC 注入序列寄存器 (ADC_JSQR)

ADC injected sequence register

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
Reserved										JL[1:0]		JSQ4[4:1]			
										rw	rw	rw	rw	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	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:22 保留，必须保持复位值。

位 21:20 **JL[1:0]**：注入序列长度 (Injected sequence length)
通过软件写入这些位可定义注入通道转换序列中的转换总数。
00：1 次转换
01：2 次转换
10：3 次转换
11：4 次转换

位 19:15 **JSQ4[4:0]**：注入序列中的第四次转换 (4th conversion in injected sequence)（当 JL[1:0] = 3 时，请参见下方的注释）
通过软件写入这些位，并将通道编号 (0..18) 分配为序列中的第四次转换。

位 14:10 **JSQ3[4:0]**：注入序列中的第三次转换 (3rd conversion in injected sequence)（当 JL[1:0] = 3 时，请参见下方的注释）

位 9:5 **JSQ2[4:0]**：注入序列中的第二次转换 (2nd conversion in injected sequence)（当 JL[1:0] = 3 时，请参见下方的注释）

位 4:0 **JSQ1[4:0]**：注入序列中的第一次转换 (1st conversion in injected sequence)（当 JL[1:0] = 3 时，请参见下方的注释）

注意： *当 JL[1:0] = 3（定序器中有 4 次注入转换）时，ADC 将按以下顺序转换通道：JSQ1[4:0]、JSQ2[4:0]、JSQ3[4:0] 和 JSQ4[4:0]。*

当 JL = 2（定序器中有 3 次注入转换）时，ADC 将按以下顺序转换通道：JSQ2[4:0]、JSQ3[4:0] 和 JSQ4[4:0]。

当 JL = 1（定序器中有 2 次注入转换）时，ADC 转换通道的顺序为：先是 JSQ3[4:0]，而后是 JSQ4[4:0]。

当 JL = 0（定序器中有 1 次注入转换）时，ADC 将仅转换 JSQ4[4:0] 通道。

11.13.13 ADC 注入数据寄存器 x (ADC_JDRx) (x= 1..4)

ADC injected data register x

偏移地址：0x3C - 0x48

复位值：0x0000 0000

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

位 31:16 保留，必须保持复位值。

位 15:0 **JDATA[15:0]**：注入数据 (Injected data)

这些位为只读。它们包括来自注入通道 X 的转换结果。数据有左对齐和右对齐两种方式，如*图 38* 和*图 39* 所示。

11.13.14 ADC 规则数据寄存器 (ADC_DR)

ADC regular data register

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

位 31:16 保留，必须保持复位值。

位 15:0 **DATA[15:0]**：规则数据 (Regular data)

这些位为只读。它们包括来自规则通道的转换结果。数据有左对齐和右对齐两种方式，如*图 38* 和*图 39* 所示。

11.13.15 ADC 通用状态寄存器 (ADC_CSR)

ADC Common status register

偏移地址：0x00（该偏移地址与 ADC1 基地址 + 0x300 相关）

复位值：0x0000 0000

该寄存器可提供不同 ADC 的状态位图像。但是，它为只读形式且不允许将不同的状态位清零。必须在对应的 ADC_SR 寄存器中将其写为 0，才能将各个状态位清零。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved										OVR3	STRT3	JSTRT3	JEOC 3	EOC3	AWD3
										ADC3					
										r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		OVR2	STRT2	JSTRT 2	JEOC2	EOC2	AWD2	Reserved		OVR1	STRT1	JSTRT1	JEOC 1	EOC1	AWD1
		ADC2								ADC1					
		r	r	r	r	r	r			r	r	r	r	r	r

位 31:22 保留，必须保持复位值。

位 21 **OVR3**：ADC3 的溢出标志 (Overrun flag of ADC3)
该位是 ADC3_SR 寄存器中 OVR 位的副本。

位 20 **STRT3**：ADC3 的规则通道开始标志 (Regular channel Start flag of ADC3)
该位是 ADC3_SR 寄存器中 STRT 位的副本。

位 19 **JSTRT3**：ADC3 的注入通道开始标志 (Injected channel Start flag of ADC3)
该位是 ADC3_SR 寄存器中 JSTRT 位的副本。

位 18 **JEOC3**：ADC3 的注入通道转换结束 (Injected channel end of conversion of ADC3)
该位是 ADC3_SR 寄存器中 JEOC 位的副本。

位 17 **EOC3**：ADC3 的转换结束 (End of conversion of ADC3)
该位是 ADC3_SR 寄存器中 EOC 位的副本。

位 16 **AWD3**：ADC3 的模拟看门狗标志 (Analog watchdog flag of ADC3)
该位是 ADC3_SR 寄存器中 AWD 位的副本。

位 15:14 保留，必须保持复位值。

位 13 **OVR2**：ADC2 的溢出标志 (Overrun flag of ADC2)
该位是 ADC2_SR 寄存器中 OVR 位的副本。

位 12 **STRT2**：ADC2 的规则通道开始标志 (Regular channel Start flag of ADC2)
该位是 ADC2_SR 寄存器中 STRT 位的副本。

位 11 **JSTRT2**：ADC2 的注入通道开始标志 (Injected channel Start flag of ADC2)
该位是 ADC2_SR 寄存器中 JSTRT 位的副本。

位 10 **JEOC2**：ADC2 的注入通道转换结束 (Injected channel end of conversion of ADC2)
该位是 ADC2_SR 寄存器中 JEOC 位的副本。

位 9 **EOC2**：ADC2 转换结束 (End of conversion of ADC2)
该位是 ADC2_SR 寄存器中 EOC 位的副本。

位 8 **AWD2**：ADC2 的模拟看门狗标志 (Analog watchdog flag of ADC2)
该位是 ADC2_SR 寄存器中 AWD 位的副本。

位 7:6 保留，必须保持复位值。

位 5 **OVR1**：ADC1 的溢出标志 (Overrun flag of ADC1)
该位是 ADC1_SR 寄存器中 OVR 位的副本。

位 4 **STRT1**：ADC1 的规则通道开始标志 (Regular channel Start flag of ADC1)
该位是 ADC1_SR 寄存器中 STRT 位的副本。

位 3 **JSTRT1**：ADC1 的注入通道开始标志 (Injected channel Start flag of ADC1)
该位是 ADC1_SR 寄存器中 JSTRT 位的副本。

位 2 **JEOC1**：ADC1 的注入通道转换结束 (Injected channel end of conversion of ADC1)
该位是 ADC1_SR 寄存器中 JEOC 位的副本。

位 1 **EOC1**：ADC1 的转换结束 (End of conversion of ADC1)
该位是 ADC1_SR 寄存器中 EOC 位的副本。

位 0 **AWD1**：ADC1 的模拟看门狗标志 (Analog watchdog flag of ADC1)
该位是 ADC1_SR 寄存器中 AWD 位的副本。

11.13.16 ADC 通用控制寄存器 (ADC_CCR)

ADC common control register

偏移地址：0x04（该偏移地址与 ADC1 基地址 + 0x300 相关）

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved								TSVREFE	VBATE	Reserved				ADCPRE	
								rw	rw					rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DMA[1:0]		DDS	Res.	DELAY[3:0]				Reserved			MULTI[4:0]				
rw	rw	rw		rw	rw	rw	rw				rw	rw	rw	rw	rw

位 31:24 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)

通过软件将该位置 1 和清零可使能/禁止温度传感器和 V_{REFINT} 通道。

0：禁止温度传感器和 V_{REFINT} 通道

1：使能温度传感器和 V_{REFINT} 通道

注意：对于 STM32F42x 和 STM32F43x 器件，当 TSVREFE 位置 1 时 必须禁止 VBATE。两个位同时置 1 时，仅进行 VBAT 转换。

位 22 **VBATE**：V_{BAT} 使能 (V_{BAT} enable)

通过软件将该位置 1 和清零可使能/禁止 V_{BAT} 通道。

0：禁止 V_{BAT} 通道

1：使能 V_{BAT} 通道

位 21:18 保留，必须保持复位值。

位 17:16 **ADCPRE**：ADC 预分频器 (ADC prescaler)

由软件置 1 和清零，以选择 ADC 的时钟频率。该时钟为所有 ADC 所共用。

注意： 00：PCLK2 2 分频

01：PCLK2 4 分频

10：PCLK2 6 分频

11：PCLK2 8 分频

位 15:14 **DMA**：直接存储器访问模式（对于多个 ADC 模式）(Direct memory access mode for multi ADC mode)

此位由软件置 1 和清零。有关详细信息，请参见 DMA 控制器一节。

00：禁止 DMA 模式

01：使能 DMA 模式 1（依次 2/3 半字 - 1、2、3 依次进行）

10：使能 DMA 模式 2（成对 2/3 半字 - 2 和 1、1 和 3、3 和 2 依次进行）

11：使能 DMA 模式 3（成对 2/3 字节 - 2 和 1、1 和 3、3 和 2 依次进行）

位 13 **DDS**：DMA 禁止选择（对于多个 ADC 模式）(DMA disable selection (for multi-ADC mode))

此位由软件置 1 和清零。

0：最后一次传输后不发出新的 DMA 请求（在 DMA 控制器中进行配置）DMA 位不通过硬件清零，但必须在生成新的 DMA 请求前，通过软件清零并设置为需要的模式。

1：只要数据发生转换且 DMA = 01、10 或 11，便会发出 DMA 请求。

位 12 保留，必须保持复位值。

位 11:8 **DELAY**：2 个采样阶段之间的延迟 (Delay between 2 sampling phases)
由软件置 1 和清零。这些位在双重或三重交错模式下使用。
0000：5 * T_{ADCCLK}
0001：6 * T_{ADCCLK}
0010：7 * T_{ADCCLK}
...
1111：20 * T_{ADCCLK}

位 7:5 保留，必须保持复位值。

位 4:0 **MULTI[4:0]**：多重 ADC 模式选择 (Multi ADC mode selection)
通过软件写入这些位可选择操作模式。
– 所有 ADC 均独立：
00000：独立模式
– 00001 到 01001：双重模式，ADC1 和 ADC2 一起工作，ADC3 独立
00001：规则同时 + 注入同时组合模式
00010：规则同时 + 交替触发组合模式
00011：Reserved
00101：仅注入同时模式
00110：仅规则同时模式
仅交错模式
01001：仅交替触发模式
– 10001 到 11001：三重模式：ADC1、ADC2 和 ADC3 一起工作
10001：规则同时 + 注入同时组合模式
10010：规则同时 + 交替触发组合模式
10011：Reserved
10101：仅注入同时模式
10110：仅规则同时模式
仅交错模式
11001：仅交替触发模式
其它所有组合均需保留且不允许编程

注意：在多重模式下，更改通道配置会生成中止，进而导致同步丢失。建议在更改配置前禁用多重 ADC 模式。

11.13.17 适用于双重和三重模式的 ADC 通用规则数据寄存器（ADC_CDR)

ADC common regular data register for dual and triple modes

偏移地址：0x08（该偏移地址与 ADC1 基地址 + 0x300 相关）

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATA2[15:0]															
r	r	r	r	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
DATA1[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

位 31:16 **DATA2[15:0]**：规则转换对中的第二个数据项 (2nd data item of a pair of regular conversions)
– 在双重模式下，这些位包含 ADC2 的规则数据。请参见*双重 ADC 模式*。
– 在三重模式下，这些位还可以包含 ADC2、ADC1 和 ADC3 的规则数据。请参见*三重 ADC 模式*。

位 15:0 **DATA1[15:0]**：规则转换对中的第一个数据项 (1st data item of a pair of regular conversions)
– 在双重模式下，这些位包含 ADC1 的规则数据。请参见*双重 ADC 模式*。
– 在三重模式下，这些位还可以包含 ADC1、ADC3 和 ADC2 的规则数据。请参见*三重 ADC 模式*。

11.13.18 ADC 寄存器映射

下表对 ADC 寄存器进行了汇总。

表 54. ADC 全局寄存器映射

偏移	寄存器
0x000 - 0x04C	ADC1
0x050 - 0x0FC	Reserved
0x100 - 0x14C	ADC2
0x118 - 0x1FC	Reserved
0x200 - 0x24C	ADC3
0x250 - 0x2FC	Reserved
0x300 - 0x308	通用寄存器

表 55. 每个 ADC 的 ADC 寄存器映射和复位值

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	ADC_SR	Reserved																										OVR	STRT	JSTRT	JEOC	EOC	AWD
	Reset value																											0	0	0	0	0	0
0x04	ADC_CR1	Reserved					OVRIE	RES[1:0]		AWDEN	JAWDEN	Reserved						DISC NUM [2:0]			JDISCEN	DISCEN	JAUTO	AWD SGL	SCAN	JEOCIE	AWDIE	EOCIE	AWDCH[4:0]					
	Reset value						0	0	0	0	0							0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08	ADC_CR2	Reserved	SWSTART	EXTEN[1:0]		EXTSEL[3:0]				Reserved	JSWSTART	JEXTEN[1:0]		JEXTSEL[3:0]				Reserved				ALIGN	EOCS	DDS	DMA	Reserved						CONT	ADON
	Reset value		0	0	0	0	0	0	0		0	0	0	0	0	0	0					0	0		0							0	0
0x0C	ADC_SMPR1	Sample time bits SMPx_x																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	ADC_SMPR2	Sample time bits SMPx_x																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14	ADC_JOFR1	Reserved																				JOFFSET1[11:0]											
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0
0x18	ADC_JOFR2	Reserved																				JOFFSET2[11:0]											
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0
0x1C	ADC_JOFR3	Reserved																				JOFFSET3[11:0]											
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0

表 55. 每个 ADC 的 ADC 寄存器映射和复位值 （续）

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x20	ADC_JOFR4	Reserved																				JOFFSET4[11:0]											
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0
0x24	ADC_HTR	Reserved																				HT[11:0]											
	Reset value																					1	1	1	1	1	1	1	1	1	1	1	1
0x28	ADC_LTR	Reserved																				LT[11:0]											
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0
0x2C	ADC_SQR1	Reserved								L[3:0]				Regular channel sequence SQx_x bits																			
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x30	ADC_SQR2	Reserved	Regular channel sequence SQx_x bits																														
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x34	ADC_SQR3	Reserved	Regular channel sequence SQx_x bits																														
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x38	ADC_JSQR	Reserved										JL[1:0]		Injected channel sequence JSQx_x bits																			
	Reset value											0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x3C	ADC_JDR1	Reserved																JDATA[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x40	ADC_JDR2	Reserved																JDATA[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x44	ADC_JDR3	Reserved																JDATA[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x48	ADC_JDR4	Reserved																JDATA[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x4C	ADC_DR	Reserved																Regular DATA[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

表 56. ADC 寄存器映射和复位值（通用 DAC 寄存器）

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	ADC_CSR	Reserved										OVR	STRT	JSTRT	JEOC	EOC	AWD	Reserved		OVR	STRT	JSTRT	JEOC	EOC	AWD	Reserved		OVR	STRT	JSTRT	JEOC	EOC	AWD
	Reset value											0	0	0	0	0	0			0	0	0	0	0	0			0	0	0	0	0	0
												ADC3								ADC2								ADC1					
0x04	ADC_CCR	Reserved								TSVREFE	VBATE	Reserved				ADCPRE[1:0]		DMA[1:0]		DDS	Reserved	DELAY[3:0]				Reserved		MULTI[4:0]			
	Reset value									0	0					0	0	0	0	0		0	0	0	0			0	0	0	0	0	0
0x08	ADC_CDR	Regular DATA2[15:0]																Regular DATA1[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

有关寄存器边界地址的信息，请参见*第 52 页的表 2*。

12 数模转换器 (DAC)

除非特别说明，否则本部分适用于整个 STM32F4xx 系列。

12.1 DAC 简介

DAC 模块是 12 位电压输出数模转换器。DAC 可以按 8 位或 12 位模式进行配置，并且可与 DMA 控制器配合使用。在 12 位模式下，数据可以采用左对齐或右对齐。DAC 有两个输出通道，每个通道各有一个转换器。在 DAC 双通道模式下，每个通道可以单独进行转换；当两个通道组合在一起同步执行更新操作时，也可以同时进行转换。可通过一个输入参考电压引脚 V_{REF+}（与 ADC 共享）来提高分辨率。

12.2 DAC 主要特性

- 两个 DAC 转换器：各对应一个输出通道
- 12 位模式下数据采用左对齐或右对齐
- 同步更新功能
- 生成噪声波
- 生成三角波
- DAC 双通道单独或同时转换
- 每个通道都具有 DMA 功能
- DMA 下溢错误检测
- 通过外部触发信号进行转换
- 输入参考电压 V_{REF+}

图 54 所示为 DAC 通道的框图，*表 57* 则给出了引脚说明。

图 54. DAC 通道框图

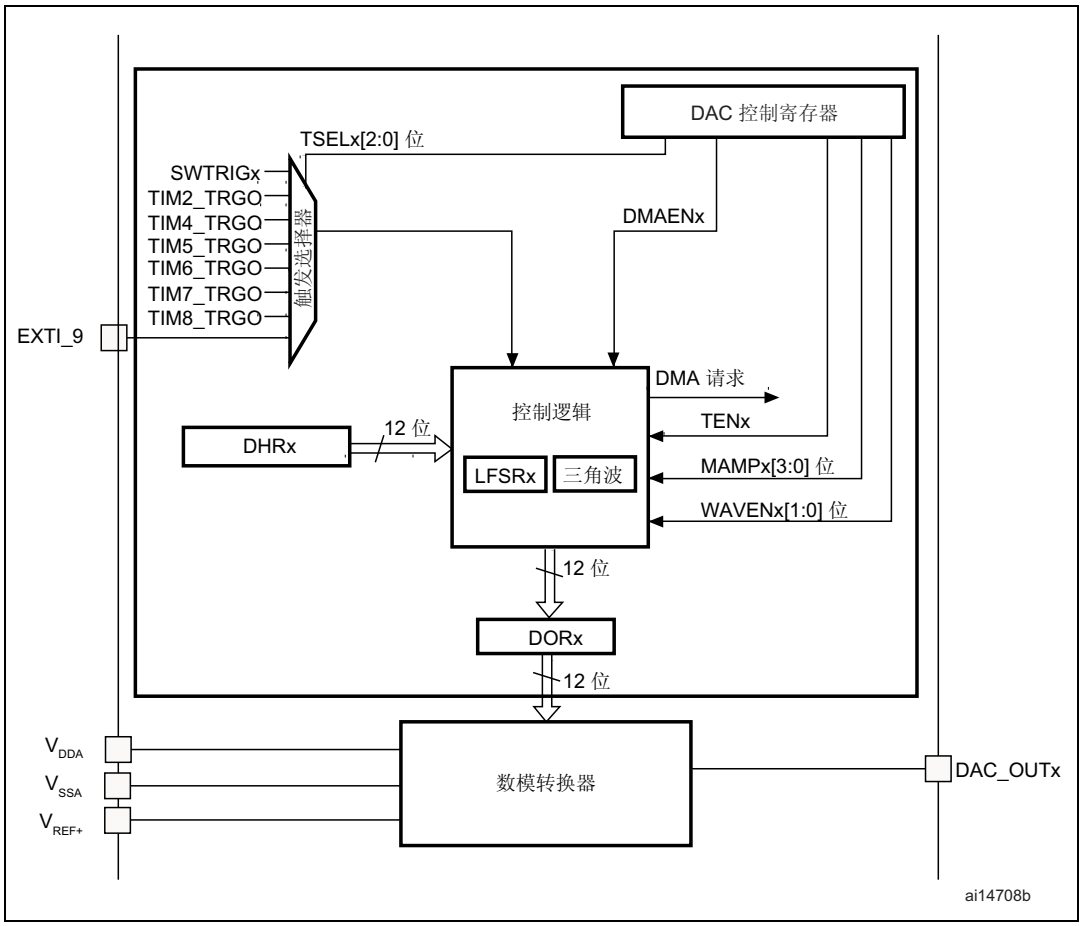


表 57. DAC 引脚

名称	信号类型	备注
V_{REF+}	正模拟参考电压输入	DAC 高/正参考电压，1.8 V ≤ V_{REF+} ≤ V_{DDA}
V_{DDA}	模拟电源输入	模拟电源
V_{SSA}	模拟电源接地输入	模拟电源接地
DAC_OUTx	模拟输出信号	DAC 通道 x 模拟输出

注意： *使能 DAC 通道 x 后，相应 GPIO 引脚（PA4 或 PA5）将自动连接到模拟转换器输出 (DAC_OUTx)。为了避免寄生电流消耗，应首先将 PA4 或 PA5 引脚配置为模拟模式 (AIN)。*

12.3 DAC 功能说明

12.3.1 DAC 通道使能

将 DAC_CR 寄存器中的相应 ENx 位置 1，即可接通对应 DAC 通道。经过一段启动时间 t_{WAKEUP} 后，DAC 通道被真正使能。

注意： *ENx 位只会使能模拟 DAC Channelx 宏单元。即使 ENx 位复位，DAC Channelx 数字接口仍处于使能状态。*

12.3.2 DAC 输出缓冲器使能

DAC 集成了两个输出缓冲器，可用来降低输出阻抗并在不增加外部运算放大器的情况下直接驱动外部负载。通过 DAC_CR 寄存器中的相应 BOFFx 位，可使能或禁止各 DAC 通道输出缓冲器。

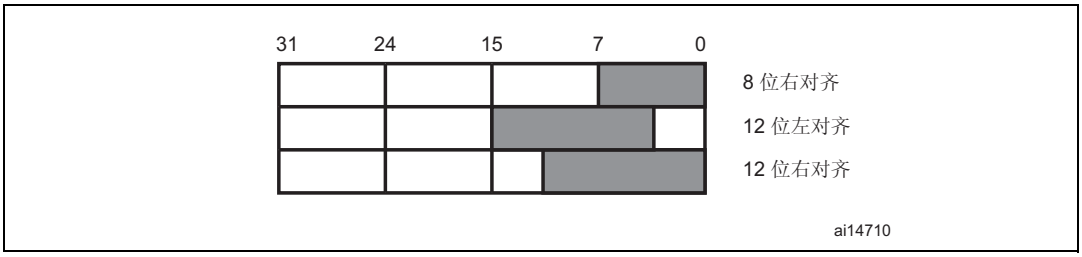
12.3.3 DAC 数据格式

根据所选配置模式，数据必须按如下方式写入指定寄存器：

- 对于 DAC 单通道 x，有三种可能的方式：
  - 8 位右对齐：软件必须将数据加载到 DAC_DHR8Rx [7:0] 位（存储到 DHRx[11:4] 位）。
  - 12 位左对齐：软件必须将数据加载到 DAC_DHR12Lx [15:4] 位（存储到 DHRx[11:0] 位）。
  - 12 位右对齐：软件必须将数据加载到 DAC_DHR12Rx [11:0] 位（存储到 DHRx[11:0] 位）。

根据加载的 DAC_DHRyyyx 寄存器，用户写入的数据将移位并存储到相应的 DHRx（数据保持寄存器 x，即内部非存储器映射寄存器）。之后，DHRx 寄存器将被自动加载，或者通过软件或外部事件触发加载到 DORx 寄存器。

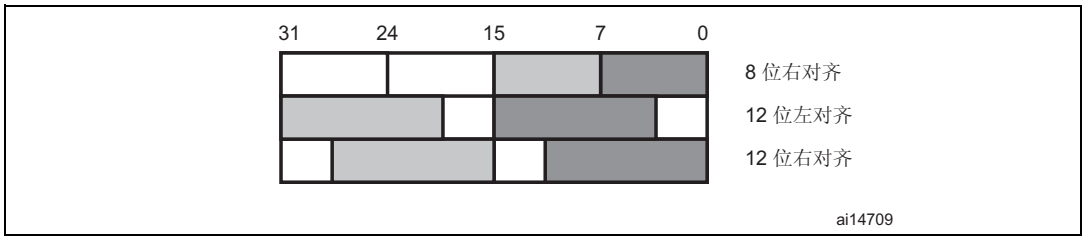
图 55. DAC 单通道模式下的数据寄存器



- 对于 DAC 双通道，有三种可能的方式：
  - 8 位右对齐：将 DAC 1 通道的数据加载到 DAC_DHR8RD [7:0] 位（存储到 DHR1[11:4] 位），将 DAC 2 通道的数据加载到 DAC_DHR8RD [15:8] 位（存储到 DHR2[11:4] 位）
  - 12 位左对齐：将 DAC 1 通道的数据加载到 DAC_DHR12RD [15:4] 位（存储到 DHR1[11:0] 位），将 DAC 2 通道的数据加载到 DAC_DHR12RD [31:20] 位（存储到 DHR2[11:0] 位）
  - 12 位右对齐：将 DAC 1 通道的数据加载到 DAC_DHR12RD [11:0] 位（存储到 DHR1[11:0] 位），将 DAC 2 通道的数据加载到 DAC_DHR12RD [27:16] 位（存储到 DHR2[11:0] 位）

根据加载的 DAC_DHRyyyD 寄存器，用户写入的数据将移位并存储到 DHR1 和 DHR2（数据保持寄存器，即内部非存储器映射寄存器）。之后，DHR1 和 DHR2 寄存器将被自动加载，或者通过软件或外部事件触发分别被加载到 DOR1 和 DOR2 寄存器。

图 56. DAC 双通道模式下的数据寄存器



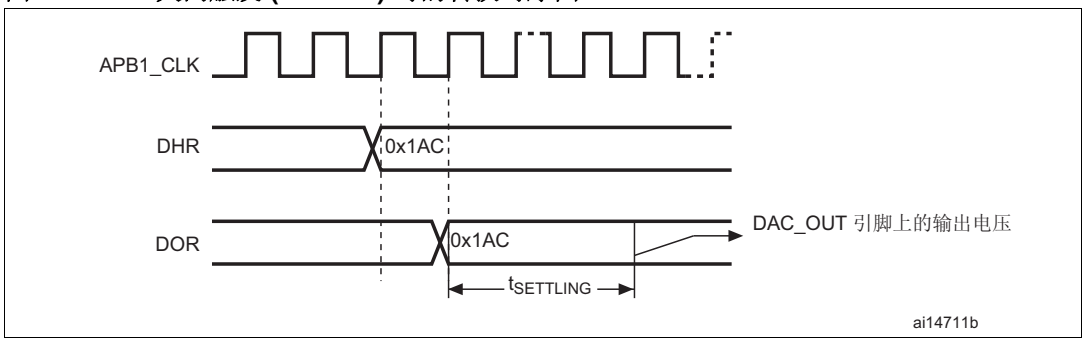
12.3.4 DAC 转换

DAC_DORx 无法直接写入，任何数据都必须通过加载 DAC_DHRx 寄存器（写入 DAC_DHR8Rx、DAC_DHR12Lx、DAC_DHR12Rx、DAC_DHR8RD、DAC_DHR12LD 或 DAC_DHR12LD）才能传输到 DAC 通道 x。

如果未选择硬件触发（DAC_CR 寄存器中的 TENx 位复位），那么经过一个 APB1 时钟周期后，DAC_DHRx 寄存器中存储的数据将自动转移到 DAC_DORx 寄存器。但是，如果选择硬件触发（置位 DAC_CR 寄存器中的 TENx 位）且触发条件到来，将在三个 APB1 时钟周期后进行转移。

当 DAC_DORx 加载了 DAC_DHRx 内容时，模拟输出电压将在一段时间 $t_{SETTLING}$ 后可用，具体时间取决于电源电压和模拟输出负载。

图 57. 关闭触发 (TEN = 0) 时的转换时序图 TEN = 0



12.3.5 DAC 输出电压

经过线性转换后，数字输入会转换为 0 到 V_{REF+} 之间的输出电压。

各 DAC 通道引脚的模拟输出电压通过以下公式确定：

$$DACoutput = V_{REF} \times \frac{DOR}{4095}$$

12.3.6 DAC 触发选择

如果 TENx 控制位置 1，可通过外部事件（定时计数器、外部中断线）触发转换。TSELx[2:0] 控制位将决定通过 8 个可能事件中的哪一个来触发转换，如*表 58* 所示。

表 58. 外部触发器

源	类型	TSEL[2:0]
Timer 6 TRGO event	片上定时器的内部信号	000
Timer 8 TRGO event		001
Timer 7 TRGO event		010
Timer 5 TRGO event		011
Timer 2 TRGO event		100
Timer 4 TRGO event		101
EXTI line9	外部引脚	110
SWTRIG	软件控制位	111

每当 DAC 接口在所选定时器 TRGO 输出或所选外部中断线 9 上检测到上升沿时，DAC_DHRx 寄存器中存储的最后一个数据即会转移到 DAC_DORx 寄存器中。发生触发后再经过三个 APB1 周期，DAC_DORx 寄存器将会得到更新。

如果选择软件触发，一旦 SWTRIG 位置 1，转换即会开始。DAC_DHRx 寄存器内容加载到 DAC_DORx 寄存器中后，SWTRIG 即由硬件复位。

注意： *ENx 位置 1 时，无法更改 TSELx[2:0] 位。*

如果选择软件触发，DAC_DHRx 寄存器的内容只需一个 APB1 时钟周期即可转移到 DAC_DORx 寄存器。

12.3.7 DMA 请求

每个 DAC 通道都具有 DMA 功能。两个 DMA 通道用于处理 DAC 通道的 DMA 请求。

当 DMAENx 位置 1 时，如果发生外部触发（而不是软件触发），则将产生 DAC DMA 请求。DAC_DHRx 寄存器的值随后转移到 DAC_DORx 寄存器。
在双通道模式下，如果两个 DMAENx 位均置 1，则将产生两个 DMA 请求。如果只需要一个 DMA 请求，应仅将相应 DMAENx 位置 1。这样，应用程序可以在双通道模式下通过一个 DMA 请求和一个特定 DMA 通道来管理两个 DAC 通道。

DMA 下溢

DAC DMA 请求没有缓冲队列。这样，如果第二个外部触发到达时尚未收到第一个外部触发的确认，将不会发出新的请求，并且 DAC_SR 寄存器中的 DAM 通道下溢标志 DMAUDRx 将置 1，以报告这一错误状况。DMA 数据传输随即禁止，并且不再处理其他 DMA 请求。DAC 通道仍将继续转换旧有数据。

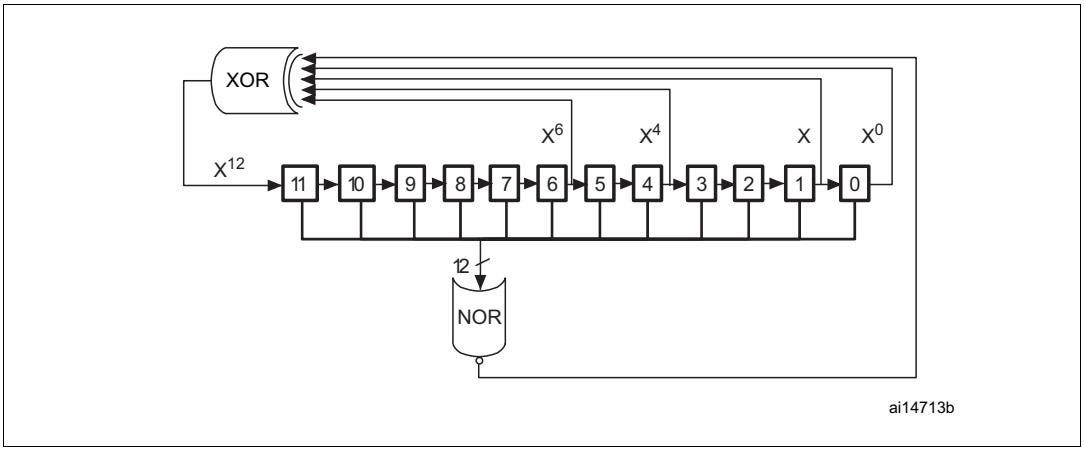
软件应通过写入“1”来将 DMAUDRx 标志清零，将所用 DMA 数据流的 DMAEN 位清零，并重新初始化 DMA 和 DAC 通道，以便正确地重新开始 DMA 传输。软件应修改 DAC 触发转换频率或减轻 DMA 工作负载，以避免再次发生 DMA 下溢。最后，可通过使能 DMA 数据传输和转换触发来继续完成 DAC 转换。

对于各 DAC 通道，如果使能 DAC_CR 寄存器中相应的 DMAUDRIEx 位，还将产生中断。

12.3.8 生成噪声

为了生成可变振幅的伪噪声，可使用 LFSR（线性反馈移位寄存器）。将 WAVEx[1:0] 置为“01”即可选择生成噪声。LFSR 中的预加载值为 0xAAA。在每次发生触发事件后，经过三个 APB1 时钟周期，该寄存器会依照特定的计算算法完成更新。

图 58. DAC LFSR 寄存器计算算法

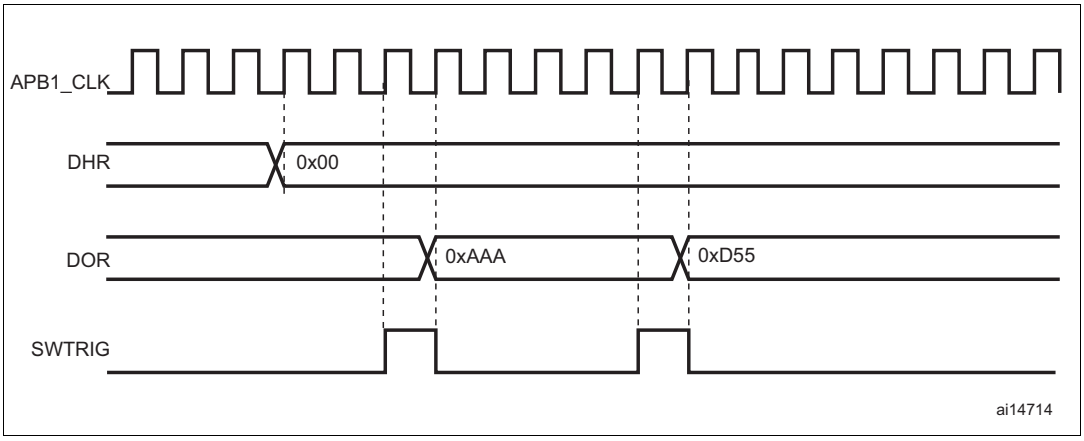


LFSR 值可以通过 DAC_CR 寄存器中的 MAMPx[3:0] 位来部分或完全屏蔽，在不发生溢出的情况下，该值将与 DAC_DHRx 的内容相加，然后存储到 DAC_DORx 寄存器中。

如果 LFSR 为 0x0000，将向其注入“1”（防锁定机制）。

可以通过复位 WAVEx[1:0] 位来将 LFSR 波形产生功能关闭。

图 59. LFSR 产生波形的 DAC 转换（使能软件触发）



注意： *要生成噪声，必须通过将 DAC_CR 寄存器中的 TENx 位置 1 来使能 DAC 触发。*

12.3.9 生成三角波

可以在直流电流或慢变信号上叠加一个小幅三角波。将 WAVEx[1:0] 置为“10”即可选择 DAC 生成三角波。振幅通过 DAC_CR 寄存器中的 MAMPx[3:0] 位进行配置。每次发生触发事件后，经过三个 APB1 时钟周期，内部三角波计数器将会递增。在不发生溢出的情况下，该计数器的值将与 DAC_DHRx 寄存器内容相加，所得总和将存储到 DAC_DORx 寄存器中。只要小于 MAMPx[3:0] 位定义的最大振幅，三角波计数器就会一直递增。一旦达到配置的振幅，计数器将递减至零，然后再递增，以此类推。

可以通过复位 WAVEx[1:0] 位来将三角波产生功能关闭。

图 60. 生成 DAC 三角波

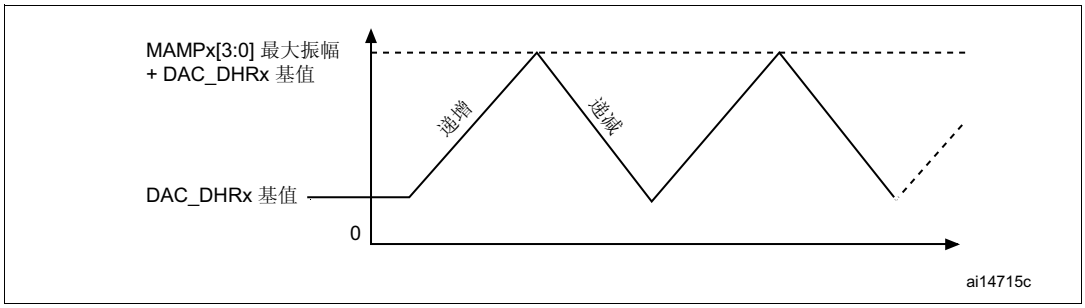
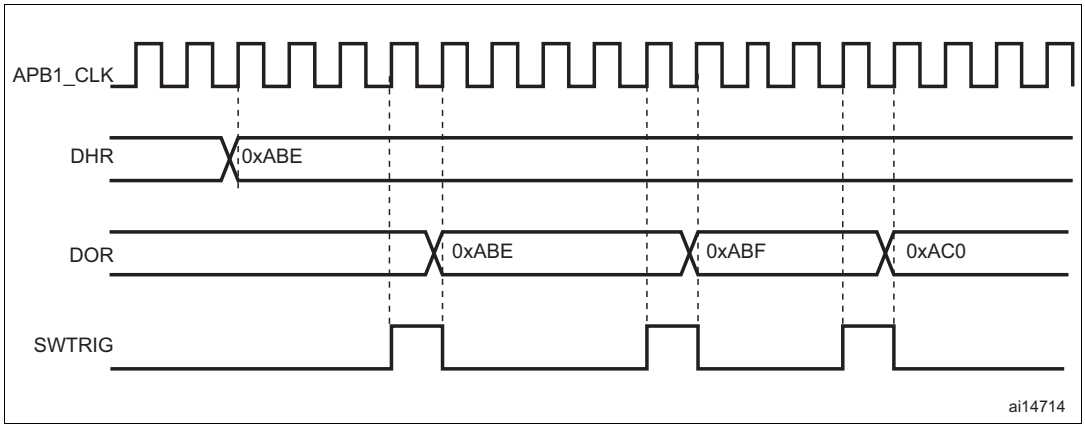


图 61. 生成三角波波形的 DAC 转换（使能软件触发）



注意： *要生成三角波，必须通过将 DAC_CR 寄存器中的 TENx 位置 1 来使能 DAC 触发。MAMPx[3:0] 位必须在使能 DAC 之前进行配置，否则将无法更改。*

12.4 DAC 双通道转换

为了在同时需要两个 DAC 通道的应用中有效利用总线带宽，DAC 模块实现了三个双寄存器：DHR8RD、DHR12RD 和 DHR12LD。这样，只需一个寄存器访问即可同时驱动两个 DAC 通道。

通过两个 DAC 通道和这三个双寄存器可以实现 11 种转换模式。但如果需要，所有这些转换模式也都可以通过单独的 DHRx 寄存器来实现。

下面几段内容将介绍所有这些模式。

12.4.1 独立触发（不产生波形）

要将 DAC 配置为此转换模式，需要遵循以下顺序：

- 将两个 DAC 通道触发使能位 TEN1 和 TEN2 置 1
- 将 TSEL1[2:0] 和 TSEL2[2:0] 设置为不同的值，以配置不同的触发源
- 将 DAC 双通道数据加载到所需 DHR 寄存器（DAC_DHR12RD、DAC_DHR12LD 或 DAC_DHR8RD）

DAC 1 通道触发信号到达时，DHR1 寄存器的内容转移到 DAC_DOR1（三个 APB1 时钟周期之后）。

DAC 2 通道触发信号到达时，DHR2 寄存器的内容转移到 DAC_DOR2（三个 APB1 时钟周期之后）。

12.4.2 独立触发（生成单个 LFSR）

要将 DAC 配置为此转换模式，需要遵循以下顺序：

- 将两个 DAC 通道触发使能位 TEN1 和 TEN2 置 1
- 将 TSEL1[2:0] 和 TSEL2[2:0] 设置为不同的值，以配置不同的触发源
- 将两个 DAC 通道的 WAVEx[1:0] 设置为“01”，并在 MAMPx[3:0] 位中配置相同的 LFSR 掩码值
- 将 DAC 双通道数据加载到所需 DHR 寄存器（DHR12RD、DHR12LD 或 DHR8RD）

DAC 通道 1 触发信号到达时，LFSR1 计数器内容（使用相同的掩码）与 DHR1 寄存器内容相加，所得总和转移到 DAC_DOR1 中（三个 APB1 时钟周期之后）。LFSR1 计数器随即更新。

DAC 通道 2 触发信号到达时，LFSR2 计数器内容（使用相同的掩码）与 DHR2 寄存器内容相加，所得总和转移到 DAC_DOR2 中（三个 APB1 时钟周期之后）。LFSR2 计数器随即更新。

12.4.3 独立触发（生成不同 LFSR）

要将 DAC 配置为此转换模式，需要遵循以下顺序：

- 将两个 DAC 通道触发使能位 TEN1 和 TEN2 置 1
- 将 TSEL1[2:0] 和 TSEL2[2:0] 设置为不同的值，以配置不同的触发源
- 将两个 DAC 通道的 WAVEx[1:0] 设置为“01”，并在 MAMP1[3:0] 和 MAMP2[3:0] 位中设置不同的 LFSR 掩码值
- 将 DAC 双通道数据加载到所需 DHR 寄存器（DAC_DHR12RD、DAC_DHR12LD 或 DAC_DHR8RD）

DAC 通道 1 触发信号到达时，LFSR1 计数器内容（使用 MAMP1[3:0] 配置的掩码）与 DHR1 寄存器内容相加，所得总和转移到 DAC_DOR1 中（三个 APB1 时钟周期之后）。LFSR1 计数器随即更新。

DAC 通道 2 触发信号到达时，LFSR2 计数器内容（使用 MAMP2[3:0] 配置的掩码）与 DHR2 寄存器内容相加，所得总和转移到 DAC_DOR2 中（三个 APB1 时钟周期之后）。LFSR2 计数器随即更新。

12.4.4 独立触发（生成单个三角波）

要将 DAC 配置为此转换模式，需要遵循以下顺序：

- 将两个 DAC 通道触发使能位 TEN1 和 TEN2 置 1
- 将 TSEL1[2:0] 和 TSEL2[2:0] 设置为不同的值，以配置不同的触发源

- 将两个 DAC 通道的 WAVEx[1:0] 设置为“1x”，并在 MAMPx[3:0] 位中配置相同的最大振幅值
- 将 DAC 双通道数据加载到所需 DHR 寄存器（DAC_DHR12RD、DAC_DHR12LD 或 DAC_DHR8RD）

DAC 通道 1 触发信号到达时，DAC 1 通道三角波计数器内容（使用相同的三角波振幅）与 DHR1 寄存器内容相加，所得总和转移到 DAC_DOR1 中（三个 APB1 时钟周期之后）。DAC 1 通道三角波计数器随即更新。

DAC 2 通道触发信号到达时，DAC 2 通道三角波计数器内容（使用相同的三角波振幅）与 DHR2 寄存器内容相加，所得总和转移到 DAC_DOR2 中（三个 APB1 时钟周期之后）。DAC 2 通道三角波计数器随即更新。

12.4.5 独立触发（生成不同三角波）

要将 DAC 配置为此转换模式，需要遵循以下顺序：

- 将两个 DAC 通道触发使能位 TEN1 和 TEN2 置 1
- 将 TSEL1[2:0] 和 TSEL2[2:0] 设置为不同的值，以配置不同的触发源
- 将两个 DAC 通道的 WAVEx[1:0] 设置为“1x”，并在 MAMP1[3:0] 和 MAMP2[3:0] 位中设置不同的最大振幅值
- 将 DAC 双通道数据加载到所需 DHR 寄存器（DAC_DHR12RD、DAC_DHR12LD 或 DAC_DHR8RD）

DAC 1 通道触发信号到达时，DAC 1 通道三角波计数器内容（使用 MAMP1[3:0] 配置的三角波振幅）与 DHR1 寄存器内容相加，所得总和转移到 DAC_DOR1 中（三个 APB1 时钟周期之后）。DAC 1 通道三角波计数器随即更新。

DAC 2 通道触发信号到达时，DAC 2 通道三角波计数器内容（使用 MAMP2[3:0] 配置的三角波振幅）与 DHR2 寄存器内容相加，所得总和转移到 DAC_DOR2 中（三个 APB1 时钟周期之后）。DAC 2 通道三角波计数器随即更新。

12.4.6 同步软件启动

要将 DAC 配置为此转换模式，需要遵循以下顺序：

- 将 DAC 双通道数据加载到所需 DHR 寄存器（DAC_DHR12RD、DAC_DHR12LD 或 DAC_DHR8RD）

在此配置中，DHR1 和 DHR2 寄存器内容会在一个 APB1 时钟周期后分别转移到 DAC_DOR1 和 DAC_DOR2 中。

12.4.7 同步触发（不产生波形）

要将 DAC 配置为此转换模式，需要遵循以下顺序：

- 将两个 DAC 通道触发使能位 TEN1 和 TEN2 置 1
- 将 TSEL1[2:0] 和 TSEL2[2:0] 设置为相同的值，以便为两个 DAC 通道配置相同的触发源
- 将 DAC 双通道数据加载到所需 DHR 寄存器（DAC_DHR12RD、DAC_DHR12LD 或 DAC_DHR8RD）

当触发信号到达时，DHR1 和 DHR2 寄存器内容将分别转移到 DAC_DOR1 和 DAC_DOR2 中（三个 APB1 时钟周期之后）。

12.4.8 同步触发（生成单个 LFSR）

要将 DAC 配置为此转换模式，需要遵循以下顺序：

- 将两个 DAC 通道触发使能位 TEN1 和 TEN2 置 1
- 将 TSEL1[2:0] 和 TSEL2[2:0] 设置为相同的值，以便为两个 DAC 通道配置相同的触发源
- 将两个 DAC 通道的 WAVEx[1:0] 设置为“01”，并在 MAMPx[3:0] 位中配置相同的 LFSR 掩码值
- 将 DAC 双通道数据加载到所需 DHR 寄存器（DHR12RD、DHR12LD 或 DHR8RD）

触发信号到达时，LFSR1 计数器内容（使用相同的掩码）与 DHR1 寄存器内容相加，所得总和转移到 DAC_DOR1 中（三个 APB1 时钟周期之后）。LFSR1 计数器随即更新。同时，LFSR2 计数器内容（使用相同的掩码）与 DHR2 寄存器内容相加，所得总和转移到 DAC_DOR2 中（三个 APB1 时钟周期之后）。LFSR2 计数器随即更新。

12.4.9 同步触发（生成不同 LFSR）

要将 DAC 配置为此转换模式，需要遵循以下顺序：

- 将两个 DAC 通道触发使能位 TEN1 和 TEN2 置 1
- 将 TSEL1[2:0] 和 TSEL2[2:0] 设置为相同的值，以便为两个 DAC 通道配置相同的触发源
- 将两个 DAC 通道的 WAVEx[1:0] 设置为“01”，并在 MAMP1[3:0] 和 MAMP2[3:0] 位中设置不同的 LFSR 掩码值
- 将 DAC 双通道数据加载到所需 DHR 寄存器（DAC_DHR12RD、DAC_DHR12LD 或 DAC_DHR8RD）

触发信号到达时，LFSR1 计数器内容（使用 MAMP1[3:0] 配置的掩码）与 DHR1 寄存器内容相加，所得总和转移到 DAC_DOR1 中（三个 APB1 时钟周期之后）。LFSR1 计数器随即更新。
同时，LFSR2 计数器内容（使用 MAMP2[3:0] 配置的掩码）与 DHR2 寄存器内容相加，所得总和转移到 DAC_DOR2 中（三个 APB1 时钟周期之后）。LFSR2 计数器随即更新。

12.4.10 同步触发（生成单个三角波）

要将 DAC 配置为此转换模式，需要遵循以下顺序：

- 将两个 DAC 通道触发使能位 TEN1 和 TEN2 置 1
- 将 TSEL1[2:0] 和 TSEL2[2:0] 设置为相同的值，以便为两个 DAC 通道配置相同的触发源
- 将两个 DAC 通道的 WAVEx[1:0] 设置为“1x”，并在 MAMPx[3:0] 位中配置相同的最大振幅值
- 将 DAC 双通道数据加载到所需 DHR 寄存器（DAC_DHR12RD、DAC_DHR12LD 或 DAC_DHR8RD）

触发信号到达时，DAC 1 通道三角波计数器内容（使用相同的三角波振幅）与 DHR1 寄存器内容相加，所得总和转移到 DAC_DOR1 中（三个 APB1 时钟周期之后）。DAC 1 通道三角波计数器随即更新。
同时，DAC 2 通道三角波计数器内容（使用相同的三角波振幅）与 DHR2 寄存器内容相加，所得总和转移到 DAC_DOR2 中（三个 APB1 时钟周期之后）。DAC 2 通道三角波计数器随即更新。

12.4.11 同步触发（生成不同三角波）

要将 DAC 配置为此转换模式，需要遵循以下顺序：

- 将两个 DAC 通道触发使能位 TEN1 和 TEN2 置 1
- 将 TSEL1[2:0] 和 TSEL2[2:0] 设置为相同的值，以便为两个 DAC 通道配置相同的触发源
- 将两个 DAC 通道的 WAVEx[1:0] 设置为“1x”，并在 MAMP1[3:0] 和 MAMP2[3:0] 位中设置不同的最大振幅值
- 将 DAC 双通道数据加载到所需 DHR 寄存器（DAC_DHR12RD、DAC_DHR12LD 或 DAC_DHR8RD）

触发信号到达时，DAC 通道 1 三角波计数器内容（使用 MAMP1[3:0] 配置的三角波振幅）与 DHR1 寄存器内容相加，所得总和转移到 DAC_DOR1 中（三个 APB1 时钟周期之后）。DAC 通道 1 三角波计数器随即更新。
同时，DAC 通道 2 三角波计数器内容（使用 MAMP2[3:0] 配置的三角波振幅）与 DHR2 寄存器内容相加，所得总和转移到 DAC_DOR2 中（三个 APB1 时钟周期之后）。DAC 通道 2 三角波计数器随即更新。

12.5 DAC 寄存器

有关寄存器说明中使用的缩写，请参见*第 47 页的第 1.1 节*。

外设寄存器必须按字（32 位）进行访问。

12.5.1 DAC 控制寄存器 (DAC_CR)

DAC control register

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
Reserved		DMAUDRIE2	DMAEN2	MAMP2[3:0]				WAVE2[1:0]		TSEL2[2:0]			TEN2	BOFF2	EN2
		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
Reserved		DMAUDRIE1	DMAEN1	MAMP1[3:0]				WAVE1[1:0]		TSEL1[2:0]			TEN1	BOFF1	EN1
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:30 保留，必须保持复位值。

位 29 **DMAUDRIE2**：DAC 2 通道 DMA 下溢中断使能 (DAC channel2 DMA underrun interrupt enable)
此位由软件置 1 和清零。
0：禁止 DAC 2 通道 DMA 下溢中断
1：使能 DAC 2 通道 DMA 下溢中断

位 28 **DMAEN2**：DAC 2 通道 DMA 使能 (DAC channel2 DMA enable)
此位由软件置 1 和清零。
0：禁止 DAC 2 通道 DMA 模式
1：使能 DAC 2 通道 DMA 模式

位 27:24 **MAMP2[3:0]**：DAC 2 通道掩码/振幅选择器 (DAC channel2 mask/amplitude selector)

这些位由软件写入，用于在生成噪声波模式下选择掩码，或者在生成三角波模式下选择振幅。

0000：不屏蔽 LFSR 的位 0/三角波振幅等于 1
0001：不屏蔽 LFSR 的位 [1:0]/三角波振幅等于 3
0010：不屏蔽 LFSR 的位 [2:0]/三角波振幅等于 7
0011：不屏蔽 LFSR 的位 [3:0]/三角波振幅等于 15
0100：不屏蔽 LFSR 的位 [4:0]/三角波振幅等于 31
0101：不屏蔽 LFSR 的位 [5:0]/三角波振幅等于 63
0110：不屏蔽 LFSR 的位 [6:0]/三角波振幅等于 127
0111：不屏蔽 LFSR 的位 [7:0]/三角波振幅等于 255
1000：不屏蔽 LFSR 的位 [8:0]/三角波振幅等于 511
1001：不屏蔽 LFSR 的位 [9:0]/三角波振幅等于 1023
1010：不屏蔽 LFSR 的位 [10:0]/三角波振幅等于 2047
≥ 1011：不屏蔽 LFSR 的位 [11:0]/三角波振幅等于 4095

位 23:22 **WAVE2[1:0]**：DAC 2 通道噪声/三角波生成使能 (DAC channel2 noise/triangle wave generation enable)

这些位由软件置 1 或清零。
00：禁止生成波
01：使能生成噪声波
1x：使能生成三角波
注意：只在位 TEN2 = 1（使能 DAC 2 通道触发）时使用

位 21:19 **TSEL2[2:0]**：DAC 2 通道触发器选择 (DAC channel2 trigger selection)

这些位用于选择 DAC 2 通道的外部触发事件
000：定时器 6 TRGO 事件
001：定时器 8 TRGO 事件
010：定时器 7 TRGO 事件
011：定时器 5 TRGO 事件
100：定时器 2 TRGO 事件
101：定时器 4 TRGO 事件
110：外部中断线 9
111：软件触发
注意：只在位 TEN2 = 1（使能 DAC 2 通道触发）时使用。

位 18 **TEN2**：DAC 2 通道触发使能 (DAC channel2 trigger enable)

此位由软件置 1 和清零，以使能/禁止 DAC 2 通道触发
0：禁止 DAC 2 通道触发，写入 DAC_DHRx 寄存器的数据在一个 APB1 时钟周期之后转移到 DAC_DOR2 寄存器
1：使能 DAC 2 通道触发，DAC_DHRx 寄存器的数据在三个 APB1 时钟周期之后转移到 DAC_DOR2 寄存器
注意：如果选择软件触发，DAC_DHRx 寄存器的内容只需一个 APB1 时钟周期即可转移到 DAC_DOR2 寄存器。

位 17 **BOFF2**：DAC 2 通道输出缓冲器禁止 (DAC channel2 output buffer disable)

此位由软件置 1 和清零，以使能/禁止 DAC 2 通道输出缓冲器。
0：使能 DAC 2 通道输出缓冲器
1：禁止 DAC 2 通道输出缓冲器

位 16 **EN2**：DAC 2 通道使能 (DAC channel2 enable)

此位由软件置 1 和清零，以使能/禁止 DAC 2 通道。
0：禁止 DAC 2 通道
1：使能 DAC 2 通道

位 15:14 保留，必须保持复位值。

位 13 **DMAUDRIE1**：DAC 1 通道 DMA 下溢中断使能 (DAC channel1 DMA Underrun Interrupt enable)

此位由软件置 1 和清零。

0：禁止 DAC 1 通道 DMA 下溢中断

1：使能 DAC 1 通道 DMA 下溢中断

位 12 **DMAEN1**：DAC 1 通道 DMA 使能 (DAC channel1 DMA enable)

此位由软件置 1 和清零。

0：禁止 DAC 1 通道 DMA 模式

1：使能 DAC 1 通道 DMA 模式

位 11:8 **MAMP1[3:0]**：DAC 1 通道掩码/振幅选择器 (DAC channel1 mask/amplitude selector)

这些位由软件写入，用于在生成噪声波模式下选择掩码，或者在生成三角波模式下选择振幅。

0000：不屏蔽 LFSR 的位 0/三角波振幅等于 1

0001：不屏蔽 LFSR 的位 [1:0]/三角波振幅等于 3

0010：不屏蔽 LFSR 的位 [2:0]/三角波振幅等于 7

0011：不屏蔽 LFSR 的位 [3:0]/三角波振幅等于 15

0100：不屏蔽 LFSR 的位 [4:0]/三角波振幅等于 31

0101：不屏蔽 LFSR 的位 [5:0]/三角波振幅等于 63

0110：不屏蔽 LFSR 的位 [6:0]/三角波振幅等于 127

0111：不屏蔽 LFSR 的位 [7:0]/三角波振幅等于 255

1000：不屏蔽 LFSR 的位 [8:0]/三角波振幅等于 511

1001：不屏蔽 LFSR 的位 [9:0]/三角波振幅等于 1023

1010：不屏蔽 LFSR 的位 [10:0]/三角波振幅等于 2047

≥ 1011：不屏蔽 LFSR 的位 [11:0]/三角波振幅等于 4095

位 7:6 **WAVE1[1:0]**：DAC 1 通道噪声/三角波生成使能 (DAC channel1 noise/triangle wave generation enable)

这些位将由软件置 1 和清零。

00：禁止生成波

01：使能生成噪声波

1x：使能生成三角波

注意：只在位 TEN1 = 1（使能 DAC 1 通道触发）时使用。

位 5:3 **TSEL1[2:0]**：DAC 1 通道触发器选择 (DAC channel1 trigger selection)

这些位用于选择 DAC 1 通道的外部触发事件。

000：定时器 6 TRGO 事件

001：定时器 8 TRGO 事件

010：定时器 7 TRGO 事件

011：定时器 5 TRGO 事件

100：定时器 2 TRGO 事件

101：定时器 4 TRGO 事件

110：外部中断线 9

111：软件触发

注意：只在位 TEN1 = 1（使能 DAC 1 通道触发）时使用。

位 2 **TEN1**：DAC 1 通道触发使能 (DAC channel1 trigger enable)

此位由软件置 1 和清零，以使能/禁止 DAC 1 通道触发。

0：禁止 DAC 1 通道触发，写入 DAC_DHRx 寄存器的数据在一个 APB1 时钟周期之后转移到 DAC_DOR1 寄存器

1：使能 DAC 1 通道触发，DAC_DHRx 寄存器的数据在三个 APB1 时钟周期之后转移到 DAC_DOR1 寄存器

注意：如果选择软件触发，DAC_DHRx 寄存器的内容只需一个 APB1 时钟周期即可转移到 DAC_DOR1 寄存器。

位 1 **BOFF1**：DAC 1 通道输出缓冲器禁止 (DAC channel1 output buffer disable)
此位由软件置 1 和清零，以使能/禁止 DAC 1 通道输出缓冲器。
0：使能 DAC 1 通道输出缓冲器
1：禁止 DAC 1 通道输出缓冲器

位 0 **EN1**：DAC 1 通道使能 (DAC channel1 enable)
此位由软件置 1 和清零，以使能/禁止 DAC 1 通道。
0：禁止 DAC 1 通道
1：使能 DAC 1 通道

12.5.2 DAC 软件触发寄存器 (DAC_SWTRIGR)

DAC software trigger register

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
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														SWTRIG2	SWTRIG1
														w	w

位 31:2 保留，必须保持复位值。

位 1 **SWTRIG2**：DAC 2 通道软件触发 (DAC channel2 software trigger)
此位由软件置 1 和清零，以使能/禁止软件触发。
0：禁止软件触发
1：使能软件触发
注意：一旦 DAC_DHR2 寄存器值加载到 DAC_DOR2 寄存器中，该位即会由硬件清零（一个 APB1 时钟周期之后）。

位 0 **SWTRIG1**：DAC 1 通道软件触发 (DAC channel1 software trigger)
此位由软件置 1 和清零，以使能/禁止软件触发。
0：禁止软件触发
1：使能软件触发
注意：一旦 DAC_DHR1 寄存器值加载到 DAC_DOR1 寄存器中，该位即会由硬件清零（一个 APB1 时钟周期之后）。

12.5.3 DAC 1 通道 12 位右对齐数据保持寄存器 (DAC_DHR12R1)

DAC channel1 12-bit right-aligned data holding register

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
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				DACC1DHR[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:12 保留，必须保持复位值。

位 11:0 **DACC1DHR[11:0]**：DAC 1 通道 12 位右对齐数据 (DAC channel1 12-bit right-aligned data)
这些位由软件写入，用于为 DAC 1 通道指定 12 位数据。

12.5.4 DAC 1 通道 12 位左对齐数据保持寄存器 (DAC_DHR12L1)

DAC channel1 12-bit left aligned data holding register

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
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DACC1DHR[11:0]												Reserved			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw				

位 31:16 保留，必须保持复位值。

位 15:4 **DACC1DHR[11:0]**：DAC 1 通道 12 位左对齐数据 (DAC channel1 12-bit left-aligned data)
这些位由软件写入，用于为 DAC 1 通道指定 12 位数据。

位 3:0 保留，必须保持复位值。

12.5.5 DAC 1 通道 8 位右对齐数据保持寄存器 (DAC_DHR8R1)

DAC channel1 8-bit right aligned data holding register

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
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								DACC1DHR[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

位 31:8 保留，必须保持复位值。

位 7:0 **DACC1DHR[7:0]**：DAC 1 通道 8 位右对齐数据 (DAC channel1 8-bit right-aligned data)
这些位由软件写入，用于为 DAC 1 通道指定 8 位数据。

12.5.6 DAC 2 通道 12 位右对齐数据保持寄存器 (DAC_DHR12R2)

DAC channel2 12-bit right aligned data holding register

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
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				DACC2DHR[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:12 保留，必须保持复位值。

位 11:0 **DACC2DHR[11:0]**：DAC 2 通道 12 位右对齐数据 (DAC channel2 12-bit right-aligned data)
这些位由软件写入，用于为 DAC 2 通道指定 12 位数据。

12.5.7 DAC 2 通道 12 位左对齐数据保持寄存器 (DAC_DHR12L2)

DAC channel2 12-bit left aligned data holding register

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
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DACC2DHR[11:0]												Reserved			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw				

位 31:16 保留，必须保持复位值。

位 15:4 **DACC2DHR[11:0]**：DAC 2 通道 12 位左对齐数据 (DAC channel2 12-bit left-aligned data)
这些位由软件写入，用于为 DAC 2 通道指定 12 位数据。

位 3:0 保留，必须保持复位值。

12.5.8 DAC 2 通道 8 位右对齐数据保持寄存器 (DAC_DHR8R2)

DAC channel2 8-bit right-aligned data holding register

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
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								DACC2DHR[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

位 31:8 保留，必须保持复位值。

位 7:0 **DACC2DHR[7:0]**：DAC 2 通道 8 位右对齐数据 (DAC channel2 8-bit right-aligned data)
这些位由软件写入，用于为 DAC 2 通道指定 8 位数据。

12.5.9 双 DAC 12 位右对齐数据保持寄存器 (DAC_DHR12RD)

Dual DAC 12-bit right-aligned data holding register

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
Reserved				DACC2DHR[11:0]											
				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
Reserved				DACC1DHR[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:28 保留，必须保持复位值。

位 27:16 **DACC2DHR[11:0]**：DAC 2 通道 12 位右对齐数据 (DAC channel2 12-bit right-aligned data)
这些位由软件写入，用于为 DAC 2 通道指定 12 位数据。

位 15:12 保留，必须保持复位值。

位 11:0 **DACC1DHR[11:0]**：DAC 1 通道 12 位右对齐数据 (DAC channel1 12-bit right-aligned data)
这些位由软件写入，用于为 DAC 1 通道指定 12 位数据。

12.5.10 双 DAC 12 位左对齐数据保持寄存器 (DAC_DHR12LD)

DUAL DAC 12-bit left aligned data holding register

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]												Reserved			
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
DACC1DHR[11:0]												Reserved			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw				

位 31:20 **DACC2DHR[11:0]**：DAC 2 通道 12 位左对齐数据 (DAC channel2 12-bit left-aligned data)
这些位由软件写入，用于为 DAC 2 通道指定 12 位数据。

位 19:16 保留，必须保持复位值。

位 15:4 **DACC1DHR[11:0]**：DAC 1 通道 12 位左对齐数据 (DAC channel1 12-bit left-aligned data)
这些位由软件写入，用于为 DAC 1 通道指定 12 位数据。

位 3:0 保留，必须保持复位值。

12.5.11 双 DAC 8 位右对齐数据保持寄存器 (DAC_DHR8RD)

DUAL DAC 8-bit right aligned data holding register

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
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DACC2DHR[7:0]								DACC1DHR[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:16 保留，必须保持复位值。

位 15:8 **DACC2DHR[7:0]**：DAC 2 通道 8 位右对齐数据 (DAC channel2 8-bit right-aligned data)
这些位由软件写入，用于为 DAC 2 通道指定 8 位数据。

位 7:0 **DACC1DHR[7:0]**：DAC 1 通道 8 位右对齐数据 (DAC channel1 8-bit right-aligned data)
这些位由软件写入，用于为 DAC 1 通道指定 8 位数据。

12.5.12 DAC 1 通道数据输出寄存器 (DAC_DOR1)

DAC channel1 data output register

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
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				DACC1DOR[11:0]											
				r	r	r	r	r	r	r	r	r	r	r	r

位 31:12 保留，必须保持复位值。

位 11:0 **DACC1DOR[11:0]**：DAC 1 通道数据输出 (DAC channel1 data output)
这些位为只读，其中包含 DAC 1 通道的数据输出。

12.5.13 DAC 2 通道数据输出寄存器 (DAC_DOR2)

DAC channel2 data output register

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
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				DACC2DOR[11:0]											
				r	r	r	r	r	r	r	r	r	r	r	r

位 31:12 保留，必须保持复位值。

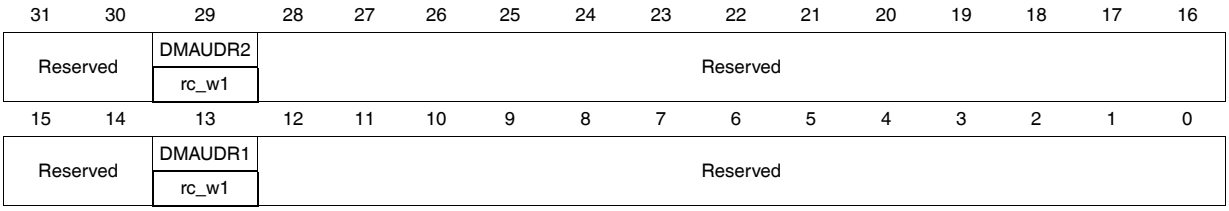
位 11:0 **DACC2DOR[11:0]**：DAC 2 通道数据输出 (DAC channel2 data output)
这些位为只读，其中包含 DAC 2 通道的数据输出。

12.5.14 DAC 状态寄存器 (DAC_SR)

DAC status register

偏移地址：0x34

复位值：0x0000 0000



位 31:30 保留，必须保持复位值。

位 29 **DMAUDR2**：DAC 2 通道 DMA 下溢标志 (DAC channel2 DMA underrun flag)
此位由硬件置 1，由软件清零（写入 1）。
0：DAC 2 通道未发生 DMA 下溢错误状况
1：DAC 2 通道发生 DMA 下溢错误状况（当前所选触发源以高于 DMA 服务能力的频率驱动 DAC 2 通道转换）

位 28:14 保留，必须保持复位值。

位 13 **DMAUDR1**：DAC 1 通道 DMA 下溢标志 (DAC channel1 DMA underrun flag)
此位由硬件置 1，由软件清零（写入 1）。
0：DAC 1 通道未发生 DMA 下溢错误状况
1：DAC 1 通道发生 DMA 下溢错误状况（当前所选触发源以高于 DMA 服务能力的频率驱动 DAC 1 通道转换）

位 12:0 保留，必须保持复位值。

12.5.15 DAC 寄存器映射

表 59 汇总了 DAC 寄存器。

表 59. DAC 寄存器映射

偏移地址	寄存器名称	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	DAC_CR	Reserved		DMAUDRIE2	DMAEN2	MAMP2[3:0]				WAVE2[2:0]		TSEL2[2:0]			TEN2	BOFF2	EN2	Reserved		DMAUDRIE1	DMAEN1	MAMP1[3:0]				WAVE1[2:0]			TSEL1[2:0]			TEN1	BOFF1	EN1
0x04	DAC_SWTRIGR	Reserved																														SWTRIG2	SWTRIG1
0x08	DAC_DHR12R1	Reserved																				DACC1DHR[11:0]											
0x0C	DAC_DHR12L1	Reserved																DACC1DHR[11:0]												Reserved			

表 59. DAC 寄存器映射 （续）

偏移地址	寄存器名称	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x10	**DAC_DHR8R1**	Reserved																								DACC1DHR[7:0]							
0x14	**DAC_DHR12R2**	Reserved																				DACC2DHR[11:0]											
0x18	**DAC_DHR12L2**	Reserved																DACC2DHR[11:0]												Reserved			
0x1C	**DAC_DHR8R2**	Reserved																								DACC2DHR[7:0]							
0x20	**DAC_DHR12RD**	Reserved				DACC2DHR[11:0]												Reserved				DACC1DHR[11:0]											
0x24	**DAC_DHR12LD**	DACC2DHR[11:0]												Reserved				DACC1DHR[11:0]												Reserved			
0x28	**DAC_DHR8RD**	Reserved																DACC2DHR[7:0]								DACC1DHR[7:0]							
0x2C	**DAC_DOR1**	Reserved																				DACC1DOR[11:0]											
0x30	**DAC_DOR2**	Reserved																				DACC2DOR[11:0]											
0x34	**DAC_SR**	Reserved		DMAUDR2		Reserved														DMAUDR1		Reserved											

有关寄存器边界地址的信息，请参见*第 52 页的表 2*。

13 数字摄像头接口 (DCMI)

除非特别说明，否则本部分适用于整个 STM32F4xx 系列。

13.1 DCMI 简介

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。

13.2 DCMI 主要特性

- 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
  - YCbCr 4:2:2 逐行视频
  - RGB 565 逐行视频
  - 压缩数据：JPEG

13.3 DCMI 引脚

表 60 显示了 DCMI 引脚。

表 60. 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
PIXCLX	像素时钟输入

13.4 DCMI 时钟

数字摄像头接口使用 PIXCLK 和 HCLK 这两个时钟域。伴随 PIXCLK 产生的信号稳定之后，将在 HCLK 上升沿时对这些信号采样。HCLK 域中的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。

13.5 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 MB/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。

图 62 显示了 DCMI 框图。

图 62. DCMI 框图

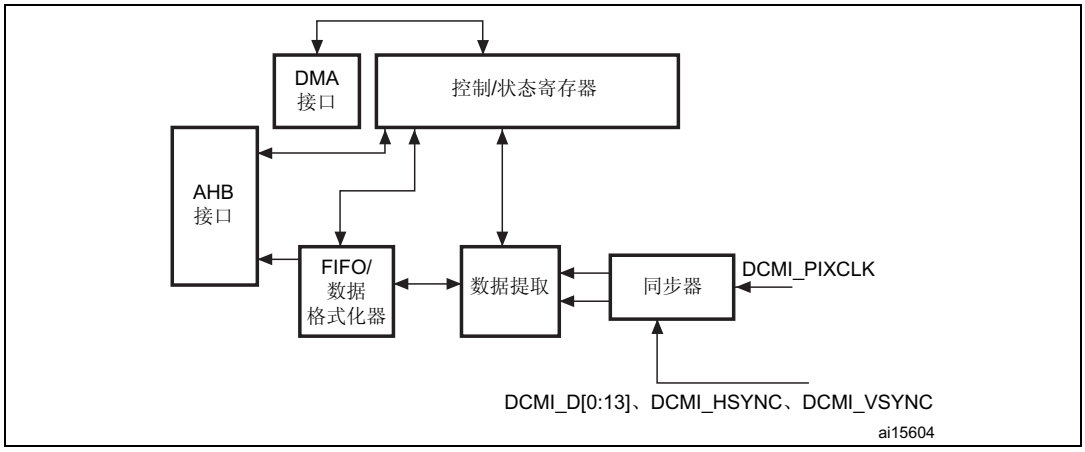
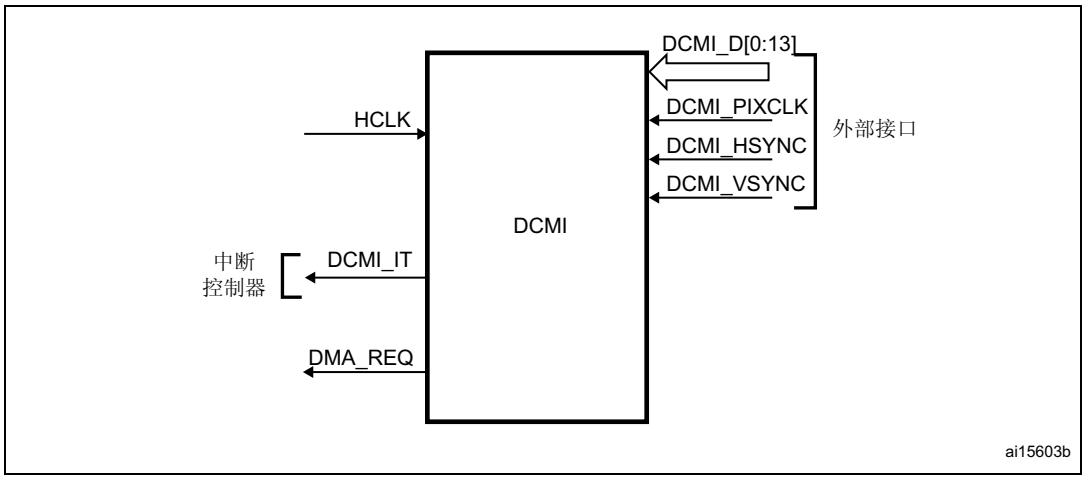


图 63. 顶级框图



13.5.1 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。

13.5.2 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 61. DCMI 信号

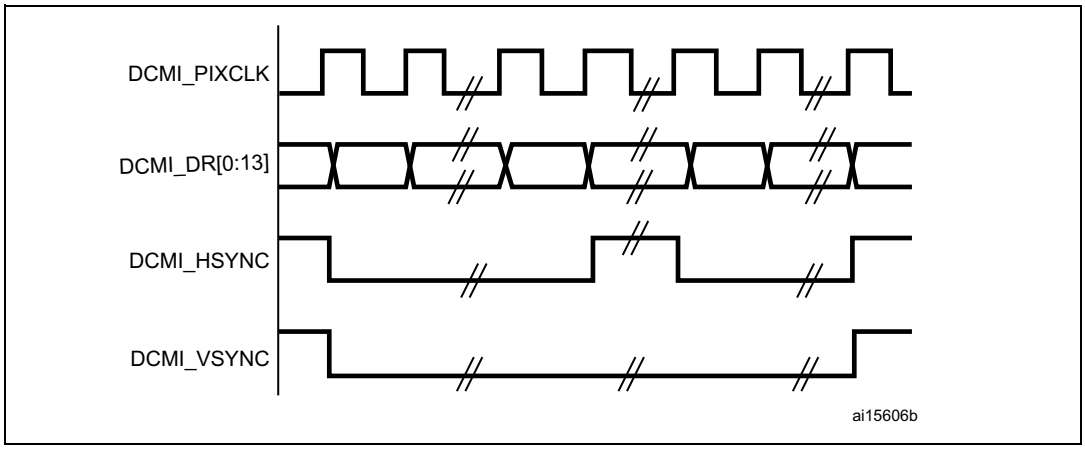
信号名称		信号说明
8 位 10 位 12 位 14 位	D[0..7] D[0..9] D[0..11] D[0..13]	数据
PIXCLK		像素时钟
HSYNC		水平同步/数据有效
VSYNC		垂直同步

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 64. DCMI 信号波形



1. DCMI_PIXCLK 的捕获沿为下降沿，DCMI_HSYNC 和 DCMI_VSYNC 的有效状态为 1。
2. 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，第四个数据字节放置在 32 位字的 MSB 位置。*表 62* 列举了捕获到的数据字节在两个 32 位字中的位置排布。

表 62. 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*表 63* 中所示。

表 63. 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*表 64* 中所示。

表 64. 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*表 65* 中所示。

表 65. 捕获的数据字节在 32 位字（宽 14 位）中的位置排布

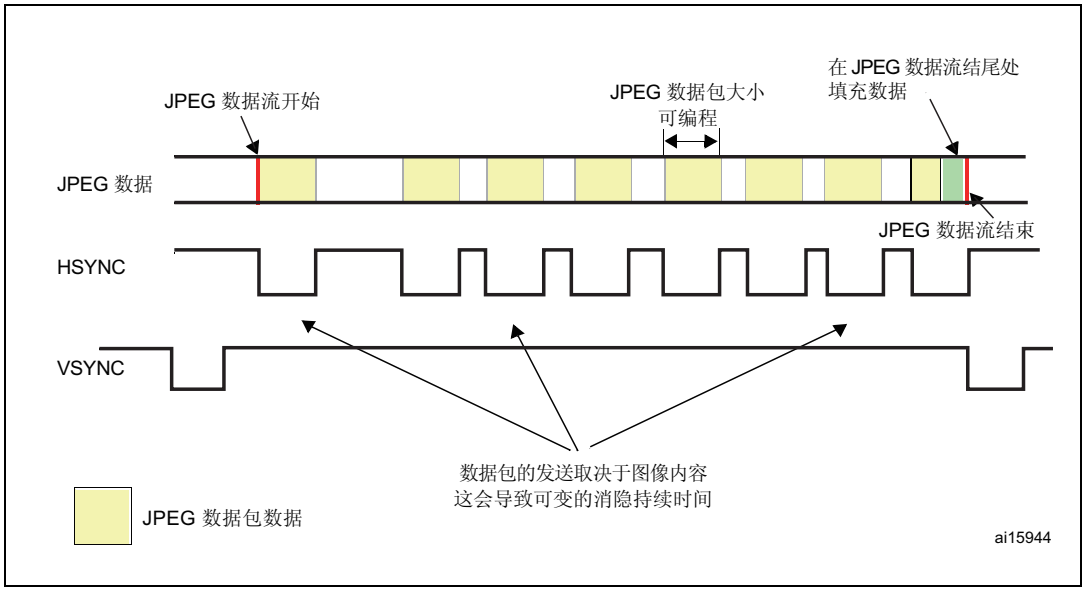
字节地址	31:30	29:16	15:14	13:0
0	0	D_{n+1}[13:0]	0	D_n[13:0]
4	0	D_{n+3}[13:0]	0	D_{n+2}[13:0]

13.5.3 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。*图 65* 显示了相应的时序图。

图 65. 时序图



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）。

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*第 13.8.7 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。

13.5.4 捕获模式

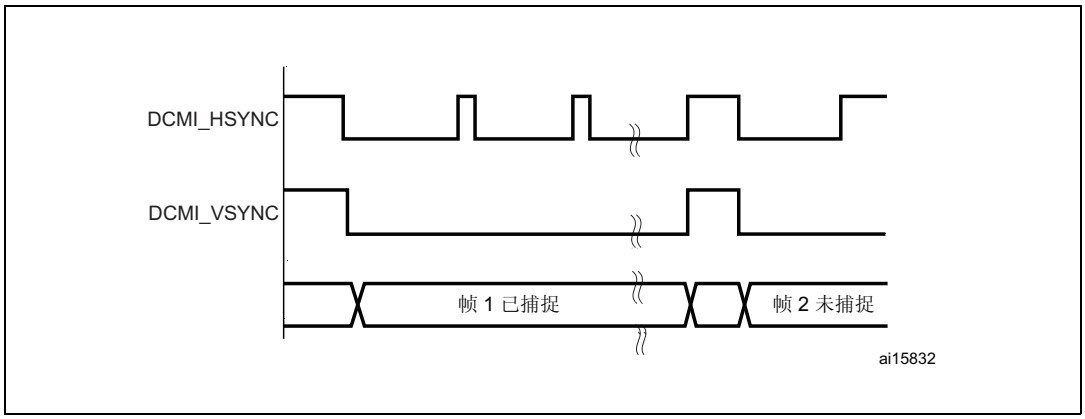
此接口支持两种类型的捕获：快照（单帧）和连续采集。

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，将生成中断 (IT_FRAME)。

如果发生溢出，这帧将丢失并且 CAPTURE 位清零。

图 66. 快照模式下的帧捕获波形

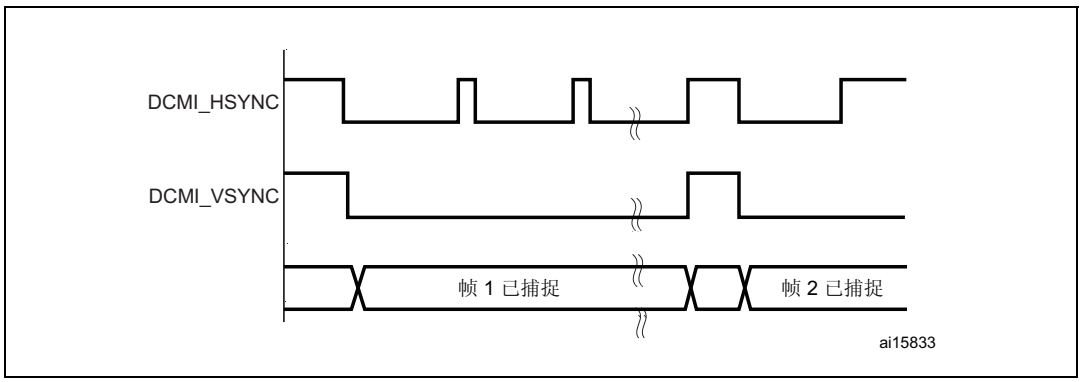


1. 此例中，DCMI_HSYNC 和 DCMI_VSYNC 的有效状态为 1。
2. DCMI_HSYNC 和 DCMI_VSYNC 的状态可同时发生更改。

连续采集模式

在此模式下（DCMI_CR 中的 CM 位 = “0”），一旦 DCMI_CR 中的 CAPTURE 位置 1，将在下一个 VSYNC 或内嵌同步码帧起始同步码时启动采集过程，具体取决于同步模式。该过程一直持续到 DCMI_CR 中的 CAPTURE 位清零。CAPTURE 位清零后，采集过程将持续到当前帧结束。

图 67. 连续采集模式下的帧捕获波形



1. 此例中，DCMI_HSYNC 和 DCMI_VSYNC 的有效状态为 1。
2. DCMI_HSYNC 和 DCMI_VSYNC 的状态可同时发生更改。

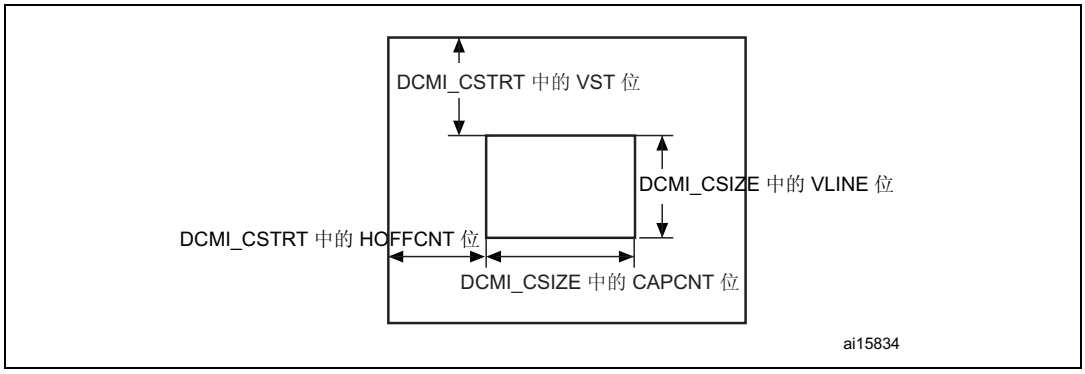
在连续采集模式下，可以通过配置 DCMI_CR 中的 FCRC 位来选择采集所有图片，或每两帧采集一次图片，或每四帧采集一张图片，以此降低帧捕获率。

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）。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。*

13.5.5 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。

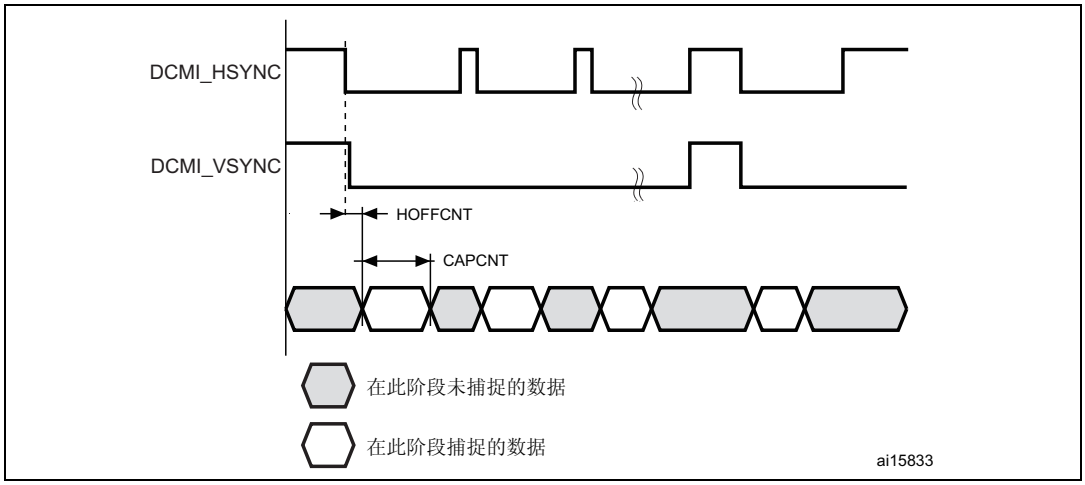
图 68. 修剪后窗口的坐标和大小



这些寄存器将捕获窗口的起点坐标指定为某一行号（从 0 开始）和像素时钟数（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 69. 数据捕获波形



1. 此例中，DCMI_HSYNC 和 DCMI_VSYNC 的有效状态为 1。
2. DCMI_HSYNC 和 DCMI_VSYNC 的状态可同时发生更改。

13.5.6 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。检测到帧结束并且尚未凑成 32 位字时，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。

13.5.7 FIFO

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。

13.6 数据格式说明

13.6.1 数据格式

支持三种类型的数据：

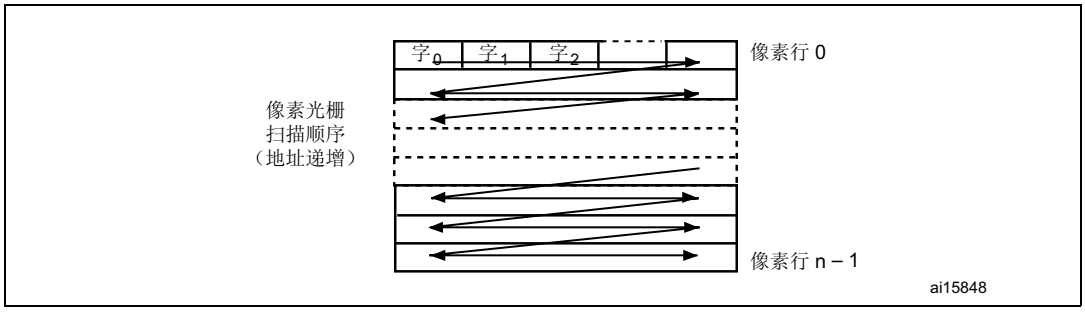
- 8 位逐行视频：单色或原始拜尔格式
- YCbCr 4:2:2 逐行视频
- RGB565 逐行视频。采用 16 位（5 位表示蓝色，5 位表示红色，6 位表示绿色）编码的像素需要两个时钟周期传输。

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 70. 像素光栅扫描顺序



13.6.2 单色格式

特性：

- 光栅格式
- 每个像素 8 位

表 66 显示了数据的存储方式。

表 66. 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

13.6.3 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 16 BPP（每个像素 16 位）：RGB565（每 32 位表示 2 个像素）。
不支持 24 BPP（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。

表 67 显示了数据的存储方式。

表 67. 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	红色 n + 1	绿色 n + 1	蓝色 n + 1	红色 n	绿色 n	蓝色 n
4	红色 n + 4	绿色 n + 3	蓝色 n + 3	红色 n + 2	绿色 n + 2	蓝色 n + 2

13.6.4 YCbCr 格式

特性：

- 光栅格式
- YCbCr 4:2:2
- 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*表 68* 中所示。

表 68. 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

13.7 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。*表 69* 列出了所有中断。

表 69. 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
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

13.8 DCMI 寄存器说明

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。

13.8.1 DCMI 控制寄存器 1 (DCMI_CR)

DCMI control register 1

偏移地址：0x00

复位值：0x0000 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																	ENABLE	Reserved		EDM		FCRC		VSPOL	HSPOL	PCKPOL	ESS	JPEG	CROP	CM	CAPTURE
																	rw			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:15 保留，必须保持复位值。

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 保留，必须保持复位值。

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

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 传输单帧。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。

13.8.2 DCMI 状态寄存器 (DCMI_SR)

DCMI status register

偏移地址：0x04

复位值：0x0000 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																													FNE	VSYNC	HSYNC
																													r	r	r

位 31:3 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

1：FIFO 包含有效数据

0：FIFO 为空

位 1 **VSYNC**

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**

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。

13.8.3 DCMI 原始中断状态寄存器 (DCMI_SR)

DCMI raw interrupt status register

偏移地址：0x08

复位值：0x0000 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																											LINE_RIS	VSYNC_RIS	ERR_RIS	OVR_RIS	FRAME_RIS
																											r	r	r	r	r

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。

位 31:5 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。

13.8.4 DCMI 中断使能寄存器 (DCMI_IER)

DCMI interrupt enable register

偏移地址：0x0C

复位值：0x0000 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																											LINE_IE	VSYNC_IE	ERR_IE	OVR_IE	FRAME_IE
																											rw	rw	rw	rw	rw

DCMI_IER 寄存器用于使能中断。当其中任一位置 1 时，将使能相应中断。此寄存器可读写。

位 31:5 保留，必须保持复位值。

位 4 **LINE_IE**：行中断使能 (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，生成一个中断

13.8.5 DCMI 屏蔽中断状态寄存器 (DCMI_MIS)

DCMI masked interrupt status register

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。

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	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																											LINE_MIS	VSYNC_MIS	ERR_MIS	OVR_MIS	FRAME_MIS
																											r	r	r	r	r

位 31:5 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，且 DCMI_IER 中的 FRAME_IE 位置 1 时生成一个中断。

13.8.6 DCMI 中断清零寄存器 (DCMI_ICR)

DCMI interrupt clear register

偏移地址：0x14

复位值：0x0000 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																											LINE_ISC	VSYNC_ISC	ERR_ISC	OVR_ISC	FRAME_ISC
																											w	w	w	w	w

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。

位 15:5 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

13.8.7 DCMI 内嵌同步码寄存器 (DCMI_ESCR)

DCMI embedded synchronization code register

偏移地址：0x18

复位值：0x0000 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FEC								LEC								LSC								FSC							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	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 0xFF 0x00 0x00 FEC。
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 0xFF 0x00 0x00 LEC。

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 0xFF 0x00 0x00 LSC。

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 0xFF 0x00 0x00 FSC。
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。

13.8.8 DCMI 内嵌码同步取消屏蔽寄存器 (DCMI_ESUR)

DCMI embedded synchronization unmask register

偏移地址：0x1C

复位值：0x0000 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FEU								LEU								LSU								FSU							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	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

13.8.9 DCMI 裁剪窗口起点 (DCMI_CWSTRT)

DCMI crop window start

偏移地址：0x20

复位值：0x0000 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			VST[12:0]													Reserved		HOFFCNT[13:0]													
			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:29 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 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。

13.8.10 DCMI 裁剪窗口大小 (DCMI_CWSIZE)

DCMI crop window size

偏移地址：0x24

复位值：0x0000 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		VLINE13:0]														Reserved		CAPCNT[13:0]													
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:30 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 => 1 行
0x0001 => 2 行
0x0002 => 3 行
....

位 15:14 保留，必须保持复位值。

位 13:0 **CAPCNT[13:0]**：捕获计数 (Capture count)
窗口内要捕捉的像素时钟数。
0x0000 => 1 个像素
0x0001 => 2 个像素
0x0002 => 3 个像素
....

13.8.11 DCMI 数据寄存器 (DCMI_DR)

DCMI data register

偏移地址：0x28

复位值：0x0000 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Byte3								Byte2								Byte1								Byte0							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

位 31:24 数据字节 3

位 23:16 数据字节 2

位 15:8 数据字节 1

位 7:0 数据字节 0

数字摄像头接口每收到 32 位数据，才触发一次 DMA 请求。4 字深度的 FIFO 可为 DMA 传输留出足够时间并避免出现 DMA 溢出情况。

13.8.12 DCMI 寄存器映射

表 70 汇总了 DCMI 寄存器。

表 70. DCMI 寄存器地址映射和复位值

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	DCMI_CR	Reserved																	ENABLE	Reserved		EDM		FCRC		VSPOL	HSPOL	PCKPOL	ESS	JPEG	CROP	CM	CAPTURE
	Reset value																		0			0	0	0	0	0	0	0	0	0	0	0	0
0x04	DCMI_SR	Reserved																													FNE	VSYNC	HSYNC
	Reset value																														0	0	0
0x08	DCMI_RIS	Reserved																											LINE_RIS	VSYNC_RIS	ERR_RIS	OVR_RIS	FRAME_RIS
	Reset value																												0	0	0	0	0
0x0C	DCMI_IER	Reserved																											LINE_IE	VSYNC_IE	ERR_IE	OVR_IE	FRAME_IE
	Reset value																												0	0	0	0	0
0x10	DCMI_MIS	Reserved																											LINE_MIS	VSYNC_MIS	ERR_MIS	OVR_MIS	FRAME_MIS
	Reset value																												0	0	0	0	0
0x14	DCMI_ICR	Reserved																											LINE_ISC	VSYNC_ISC	ERR_ISC	OVR_ISC	FRAME_ISC
	Reset value																												0	0	0	0	0
0x18	DCMI_ESCR	FEC								LEC								LSC								FSC							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1C	DCMI_ESUR	FEU								LEU								LSU								FSU							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

表 70. DCMI 寄存器地址映射和复位值 （续）

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x20	DCMI_CWSTRT	Reserved			VST[12:0]													Reserved		HOFFCNT[13:0]													
	Reset value				0	0	0	0	0	0	0	0	0	0	0	0	0			0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x24	DCMI_CWSIZE	Reserved			VLINE13:0]													Reserved		CAPCNT[13:0]													
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x28	DCMI_DR	Byte3								Byte2								Byte1								Byte0							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

有关寄存器边界地址的信息，请参见*第 52 页的表 2*。

14 高级控制定时器（TIM1 和 TIM8）

除非特别说明，否则本部分适用于整个 STM32F4xx 系列。

14.1 TIM1 和 TIM8 简介

高级控制定时器（TIM1 和 TIM8）包含一个 16 位自动重载计数器，该计数器由可编程预分频器驱动。

此类定时器可用于各种用途，包括测量输入信号的脉冲宽度（输入捕获），或者生成输出波形（输出比较、PWM 和带死区插入的互补 PWM）。

使用定时器预分频器和 RCC 时钟控制器预分频器，可将脉冲宽度和波形周期从几微秒调制到几毫秒。

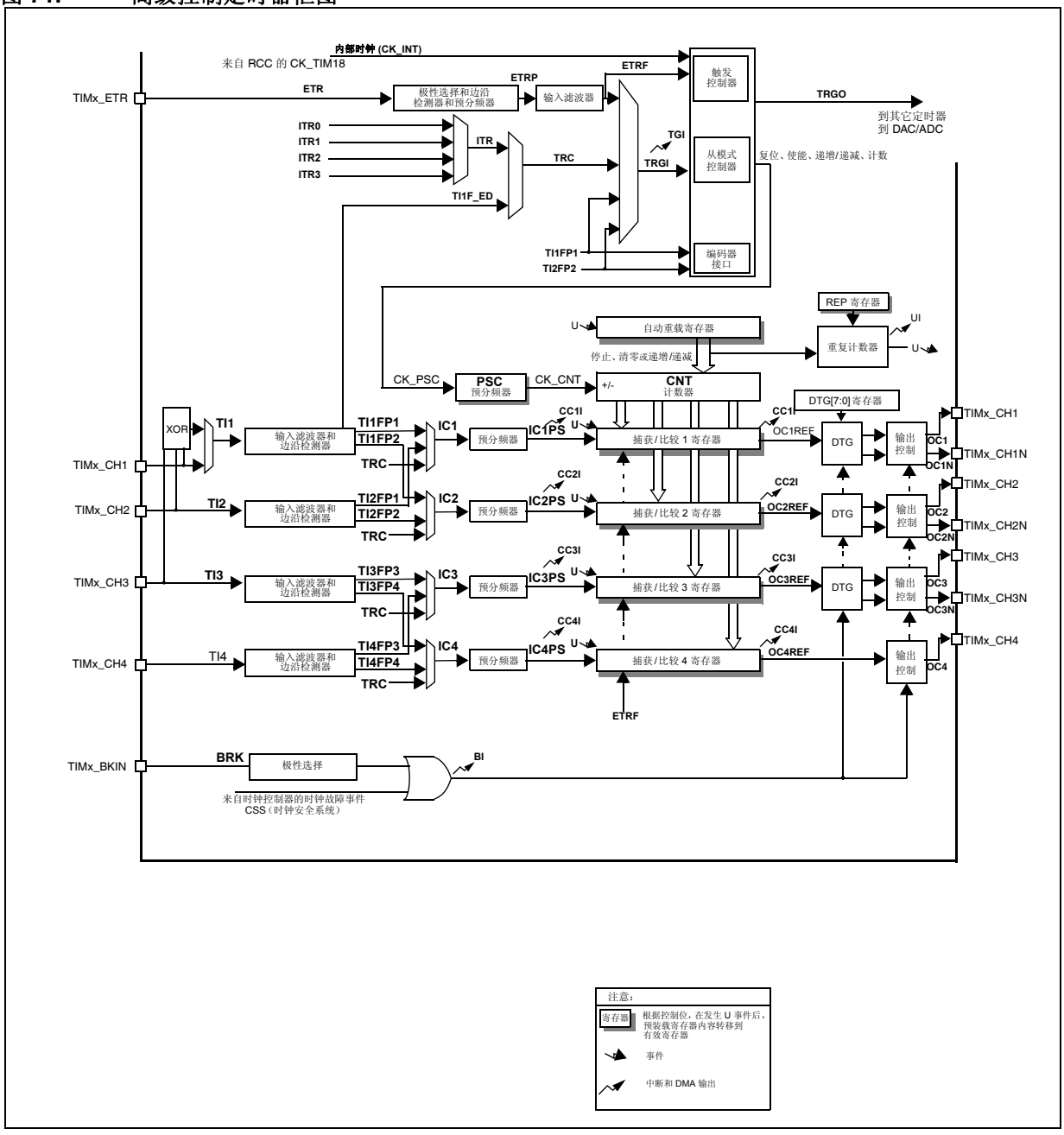
高级控制定时器（TIM1 和 TIM8）和通用 (TIMx) 定时器彼此完全独立，不共享任何资源。如*第 14.3.20 节*中所述，它们可以实现同步。

14.2 TIM1 和 TIM8 主要特性

TIM1 和 TIM8 定时器具有以下特性：

- 16 位递增、递减、递增/递减自动重载计数器。
- 16 位可编程预分频器，用于对计数器时钟频率进行分频（即运行时修改），分频系数介于 1 到 65536 之间。
- 多达 4 个独立通道，可用于：
  - 输入捕获
  - 输出比较
  - PWM 生成（边沿和中心对齐模式）
  - 单脉冲模式输出
- 带可编程死区的互补输出。
- 使用外部信号控制定时器且可实现多个定时器互连的同步电路。
- 重复计数器，用于仅在给定数目的计数器周期后更新定时器寄存器。
- 用于将定时器的输出信号置于复位状态或已知状态的断路输入。
- 发生如下事件时生成中断/DMA 请求：
  - 更新：计数器上溢/下溢、计数器初始化（通过软件或内部/外部触发）
  - 触发事件（计数器启动、停止、初始化或通过内部/外部触发计数）
  - 输入捕获
  - 输出比较
  - 断路输入
- 支持定位用增量（正交）编码器和霍尔传感器电路。
- 外部时钟触发输入或逐周期电流管理。

图 71. 高级控制定时器框图



14.3 TIM1 和 TIM8 功能说明

14.3.1 时基单元

可编程高级控制定时器的主要模块是一个 16 位计数器及其相关的自动重载寄存器。计数器可递增计数、递减计数或交替进行递增和递减计数。计数器的时钟可通过预分频器进行分频。

计数器、自动重载寄存器和预分频器寄存器可通过软件进行读写。即使在计数器运行时也可执行读写操作。

时基单元包括：

- 计数器寄存器 (TIMx_CNT)
- 预分频器寄存器 (TIMx_PSC)
- 自动重载寄存器 (TIMx_ARR)
- 重复计数器寄存器 (TIMx_RCR)

自动重载寄存器是预装载的。对自动重载寄存器执行写入或读取操作时会访问预装载寄存器。预装载寄存器的内容既可以直接传送到影子寄存器，也可以在每次发生更新事件 (UEV) 时传送到影子寄存器，这取决于 TIMx_CR1 寄存器中的自动重载预装载使能位 (ARPE)。当计数器达到上溢值（或者在递减计数时达到下溢值）并且 TIMx_CR1 寄存器中的 UDIS 位为 0 时，将发送更新事件。该更新事件也可由软件产生。下文将针对各配置的更新事件的产生进行详细介绍。

计数器由预分频器输出 CK_CNT 提供时钟，仅当 TIMx_CR1 寄存器中的计数器启动位 (CEN) 置 1 时，才会启动计数器（有关计数器使能的更多详细信息，另请参见从模式控制器的相关说明）。

注意，计数器将在 TIMx_CR1 寄存器的 CEN 位置 1 时刻的一个时钟周期后开始计数。

预分频器说明

预分频器可对计数器时钟频率进行分频，分频系数介于 1 和 65536 之间。该预分频器基于 TIMx_PSC 寄存器中的 16 位寄存器所控制的 16 位计数器。由于该控制寄存器具有缓冲功能，因此可对预分频器进行实时更改。而新的预分频比将在下一更新事件发生时被采用。

图 72 和*图 73* 以一些示例说明在预分频比实时变化时计数器的行为：

图 72. 预分频器分频由 1 变为 2 时的计数器时序图

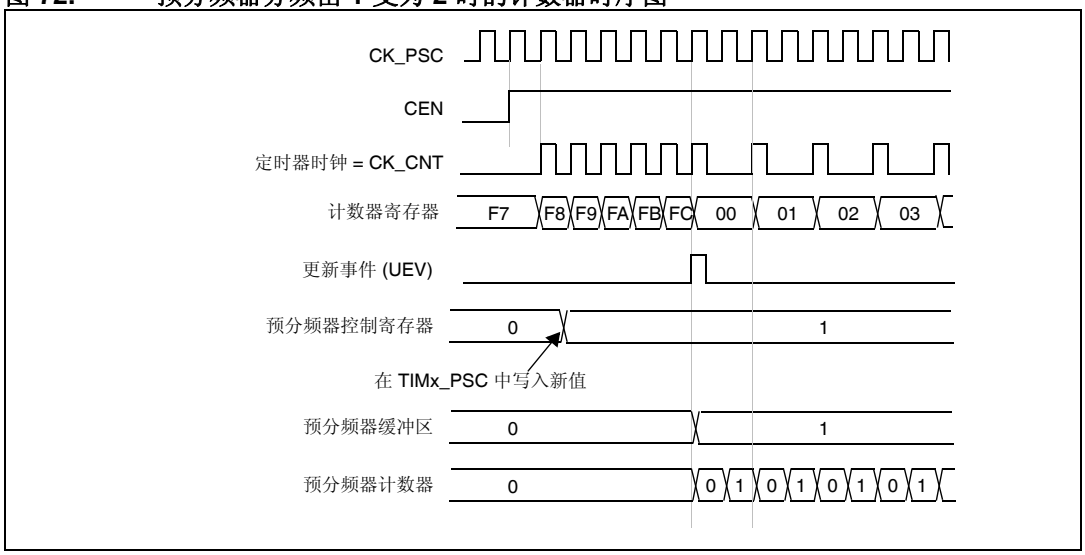
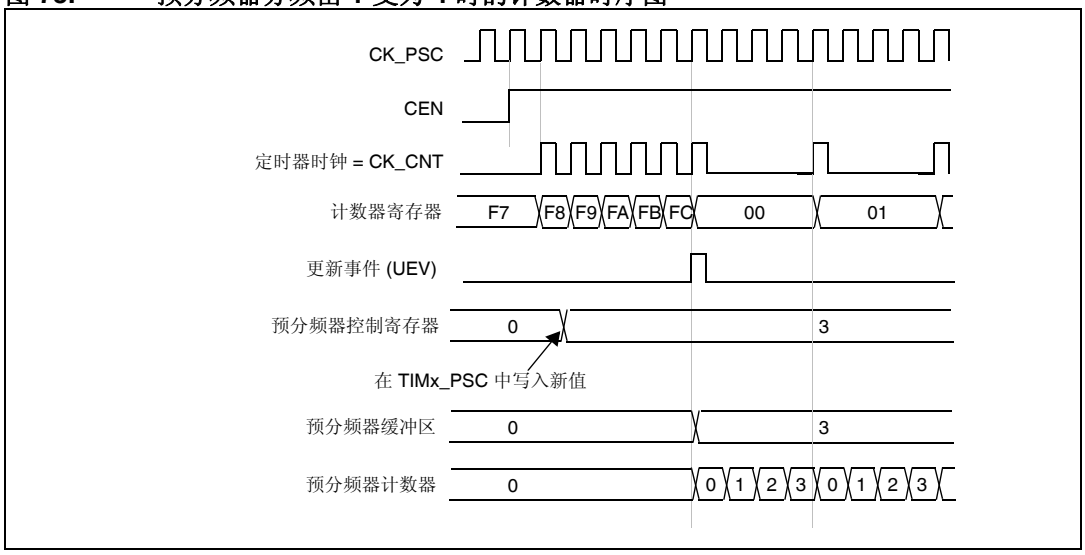


图 73. 预分频器分频由 1 变为 4 时的计数器时序图



14.3.2 计数器模式

递增计数模式

在递增计数模式下，计数器从 0 计数到自动重载值（TIMx_ARR 寄存器的内容），然后重新从 0 开始计数并生成计数器上溢事件。

如果使用重复计数器，则当递增计数的重复次数达到重复计数器寄存器中编程的次数加一次 (TIMx_RCR+1) 后，将生成更新事件 (UEV)。否则，将在每次计数器上溢时产生更新事件。

将 TIMx_EGR 寄存器的 UG 位置 1（通过软件或使用从模式控制器）时，也将产生更新事件。

通过软件将 TIMx_CR1 寄存器中的 UDIS 位置 1 可禁止 UEV 事件。这可避免向预装载寄存器写入新值时更新影子寄存器。在 UDIS 位写入 0 之前不会产生任何更新事件。不过，计数器和预分频器计数器都会重新从 0 开始计数（而预分频比保持不变）。此外，如果 TIMx_CR1 寄存器中的 URS 位（更新请求选择）已置 1，则将 UG 位置 1 会生成更新事件 UEV，但不会将 UIF 标志置 1（因此，不会发送任何中断或 DMA 请求）。这样一来，如果在发生捕获事件时将计数器清零，将不会同时产生更新中断和捕获中断。

发生更新事件时，将更新所有寄存器且将更新标志（TIMx_SR 寄存器中的 UIF 位）置 1（取决于 URS 位）：

- 重复计数器中将重新装载 TIMx_RCR 寄存器的内容
- 自动重载影子寄存器将以预装载值 (TIMx_ARR) 进行更新
- 预分频器的缓冲区中将重新装载预装载值（TIMx_PSC 寄存器的内容）

以下各图以一些示例说明当 TIMx_ARR=0x36 时不同时钟频率下计数器的行为。

图 74. 计数器时序图，1 分频内部时钟

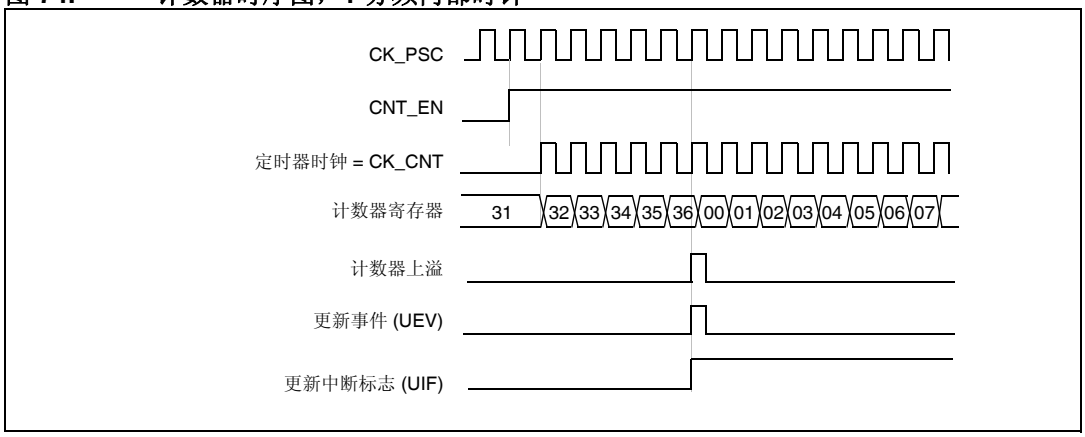


图 75. 计数器时序图，2 分频内部时钟

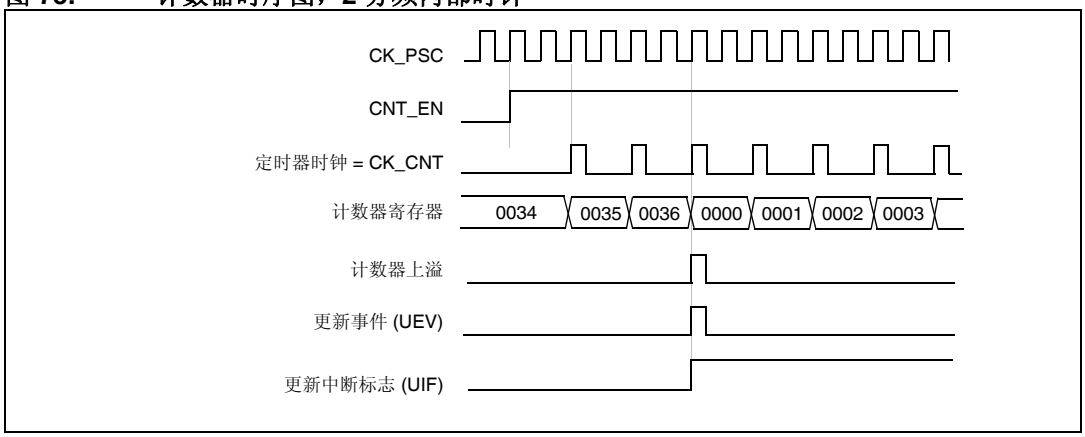


图 76. 计数器时序图，4 分频内部时钟

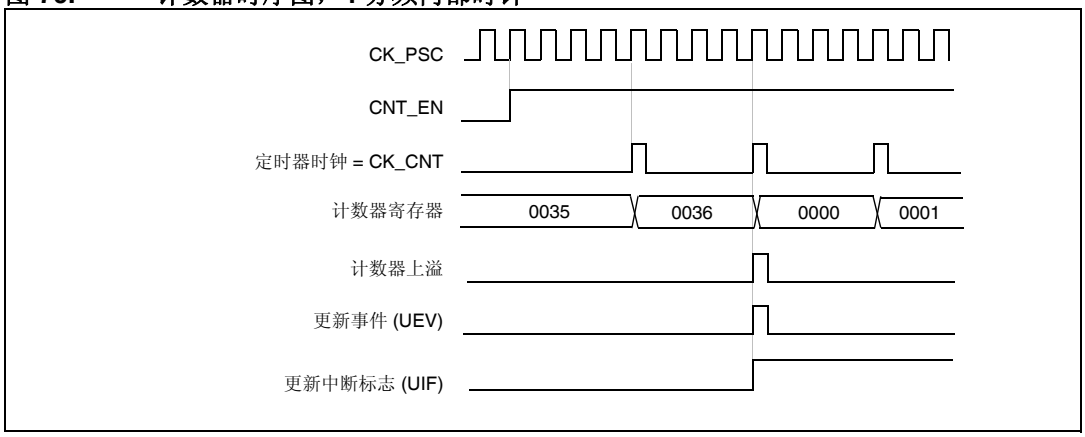


图 77. 计数器时序图，N 分频内部时钟

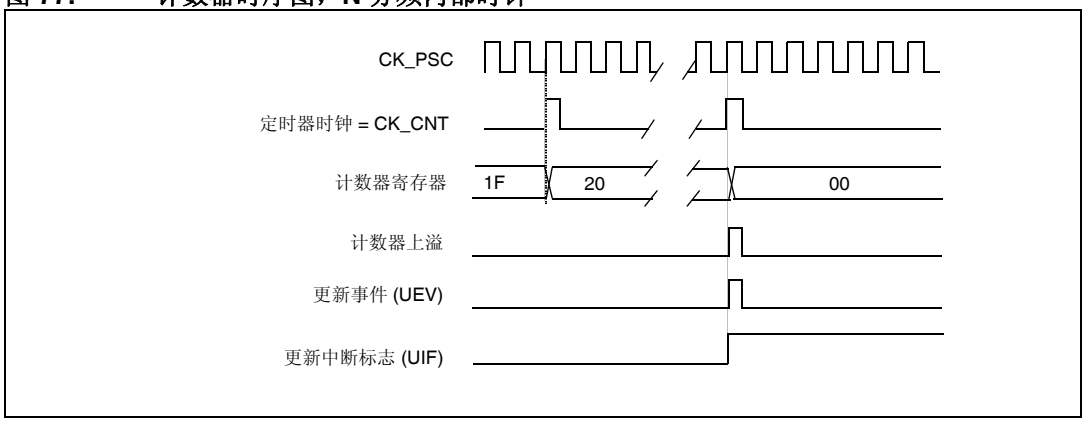


图 78. 计数器时序图，ARPE=0 时更新事件（TIMx_ARR 未预装载）

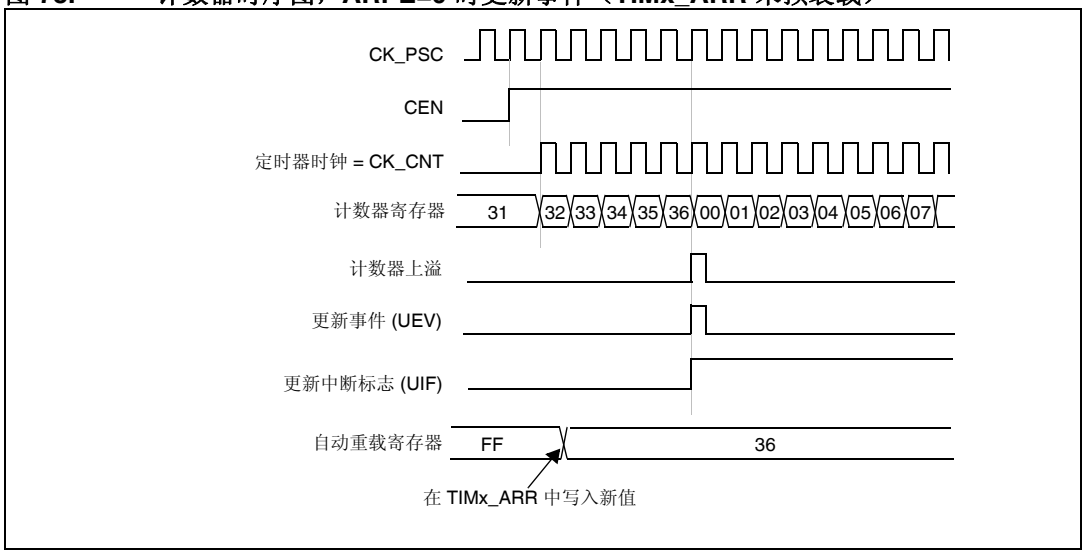
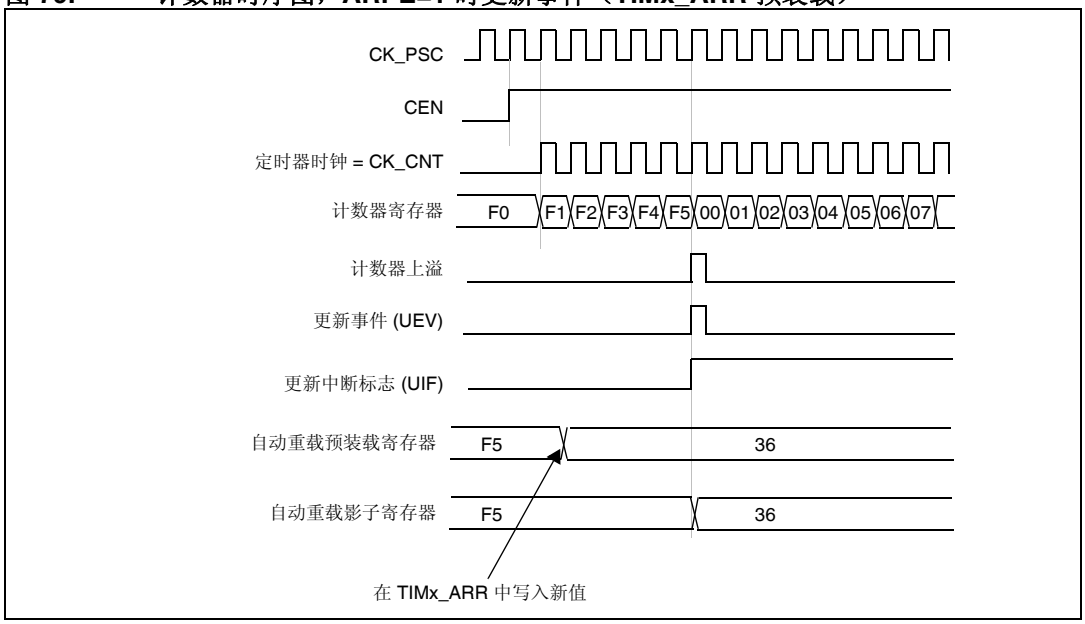


图 79. 计数器时序图，ARPE=1 时更新事件（TIMx_ARR 预装载）



递减计数模式

在递减计数模式下，计数器从自动重载值（TIMx_ARR 寄存器的内容）开始递减计数到 0，然后重新从自动重载值开始计数并生成计数器下溢事件。

如果使用重复计数器，则当递减计数的重复次数达到重复计数器寄存器中编程的次数加一次 (TIMx_RCR+1) 后，将生成更新事件 (UEV)。否则，将在每次计数器下溢时产生更新事件。

将 TIMx_EGR 寄存器的 UG 位置 1（通过软件或使用从模式控制器）时，也将产生更新事件。

通过软件将 TIMx_CR1 寄存器中的 UDIS 位置 1 可禁止 UEV 更新事件。这可避免向预装载寄存器写入新值时更新影子寄存器。在 UDIS 位写入 0 之前不会产生任何更新事件。不过，计数器会重新从当前自动重载值开始计数，而预分频器计数器则重新从 0 开始计数（但预分频比保持不变）。

此外，如果 TIMx_CR1 寄存器中的 URS 位（更新请求选择）已置 1，则将 UG 位置 1 会生成更新事件 UEV，但不会将 UIF 标志置 1（因此，不会发送任何中断或 DMA 请求）。这样一来，如果在发生捕获事件时将计数器清零，将不会同时产生更新中断和捕获中断。

发生更新事件时，将更新所有寄存器且将更新标志（TIMx_SR 寄存器中的 UIF 位）置 1（取决于 URS 位）：

- 重复计数器中将重新装载 TIMx_RCR 寄存器的内容
- 预分频器的缓冲区中将重新装载预装载值（TIMx_PSC 寄存器的内容）
- 自动重载活动寄存器将以预装载值（TIMx_ARR 寄存器的内容）进行更新。注意，自动重载寄存器会在计数器重载之前得到更新，因此，下一个计数周期就是我们所希望的新的周期长度

以下各图以一些示例说明当 TIMx_ARR=0x36 时不同时钟频率下计数器的行为。

图 80. 计数器时序图，1 分频内部时钟

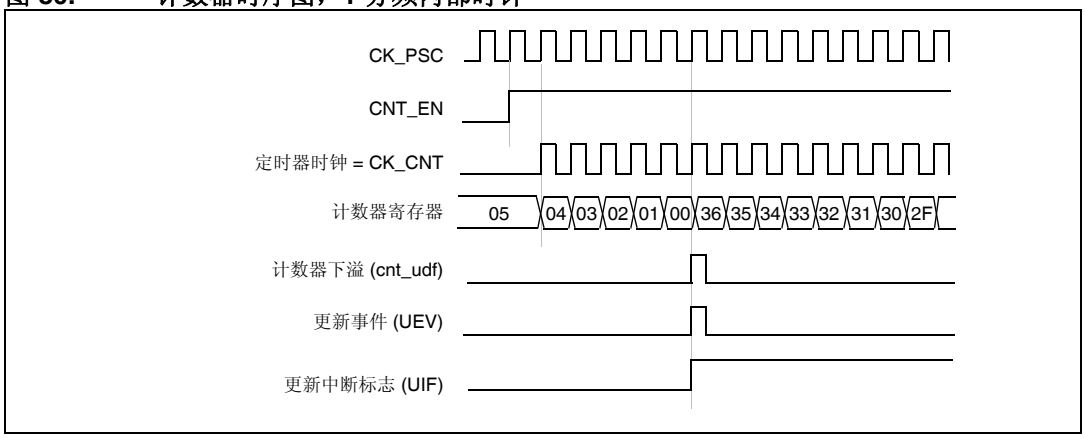


图 81. 计数器时序图，2 分频内部时钟

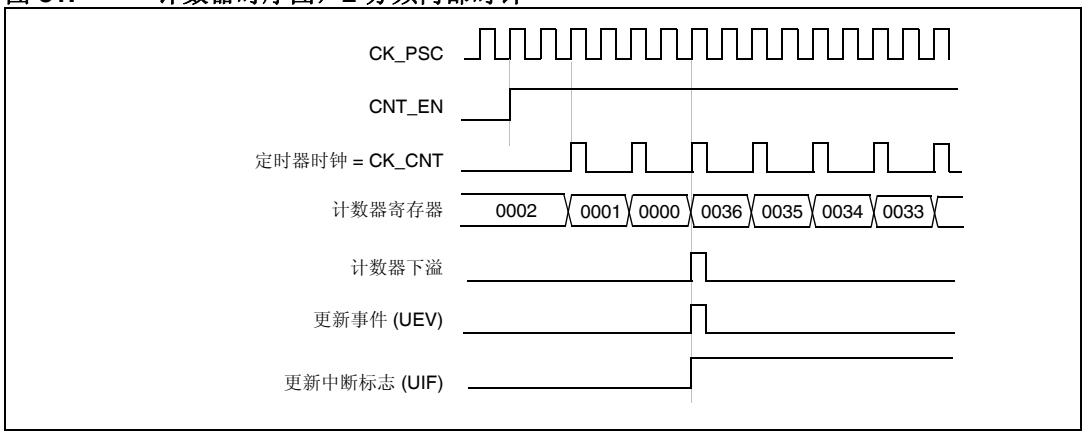


图 82. 计数器时序图，4 分频内部时钟

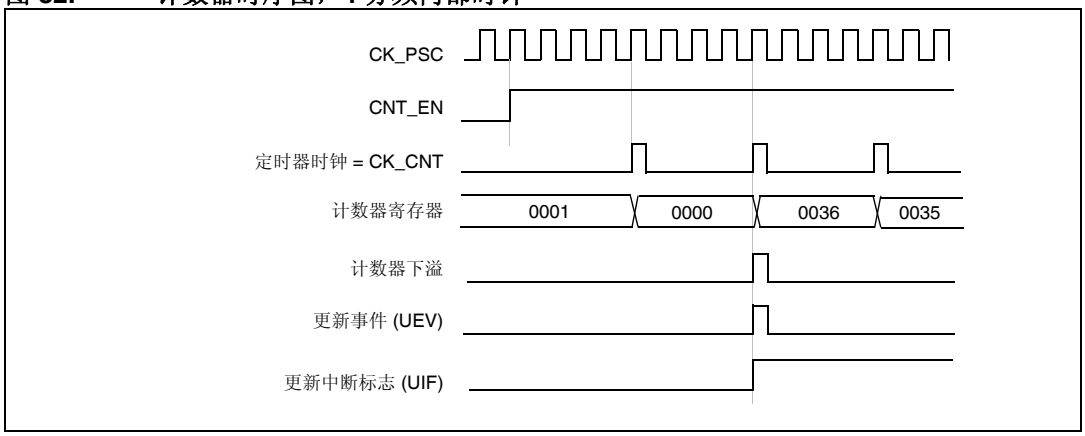


图 83. 计数器时序图，N 分频内部时钟

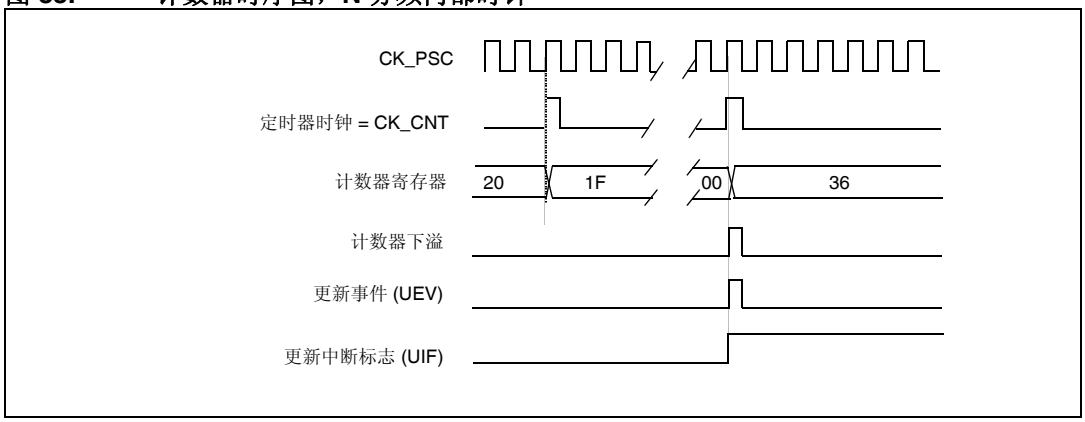
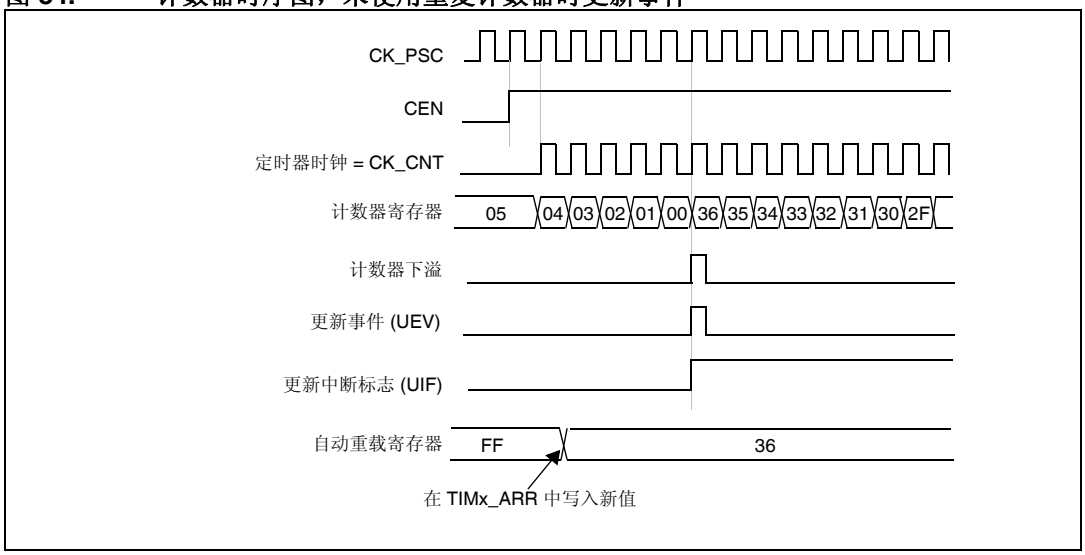


图 84. 计数器时序图，未使用重复计数器时更新事件



中心对齐模式（递增/递减计数）

在中心对齐模式下，计数器从 0 开始计数到自动重载值（TIMx_ARR 寄存器的内容）– 1，生成计数器上溢事件；然后从自动重载值开始向下计数到 1 并生成计数器下溢事件。之后从 0 开始重新计数。

当 TIMx_CR1 寄存器中的 CMS 位不为“00”时，中心对齐模式有效。将通道配置为输出模式时，其输出比较中断标志将在以下模式下置 1，即：计数器递减计数（中心对齐模式 1，CMS = “01”）、计数器递增计数（中心对齐模式 2，CMS = “10”）以及计数器递增/递减计数（中心对齐模式 3，CMS = “11”）。

在此模式下，TIMx_CR1 寄存器的 DIR 方向位不可写入值，而是由硬件更新并指示当前计数器方向。

每次发生计数器上溢和下溢时都会生成更新事件，或将 TIMx_EGR 寄存器中的 UG 位置 1（通过软件或使用从模式控制器）也可以生成更新事件。这种情况下，计数器以及预分频器计数器将重新从 0 开始计数。

通过软件将 TIMx_CR1 寄存器中的 UDIS 位置 1 可禁止 UEV 更新事件。这可避免向预装载寄存器写入新值时更新影子寄存器。在 UDIS 位写入 0 之前不会产生任何更新事件。不过，计数器仍会根据当前自动重载值进行递增和递减计数。

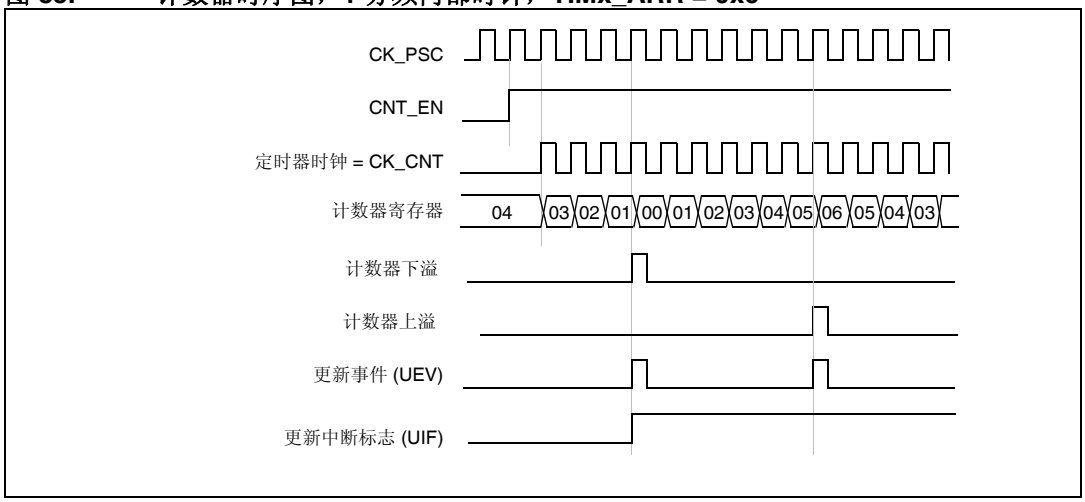
此外，如果 TIMx_CR1 寄存器中的 URS 位（更新请求选择）已置 1，则将 UG 位置 1 会生成 UEV 更新事件，但不会将 UIF 标志置 1（因此，不会发送任何中断或 DMA 请求）。这样一来，如果在发生捕获事件时将计数器清零，将不会同时产生更新中断和捕获中断。

发生更新事件时，将更新所有寄存器且将更新标志（TIMx_SR 寄存器中的 UIF 位）置 1（取决于 URS 位）：

- 重复计数器中将重新装载 TIMx_RCR 寄存器的内容
- 预分频器的缓冲区中将重新装载预装载值（TIMx_PSC 寄存器的内容）
- 自动重载活动寄存器将以预装载值（TIMx_ARR 寄存器的内容）进行更新。注意，如果更新操作是由计数器上溢触发的，则自动重载寄存器在重载计数器之前更新，因此，下一个计数周期就是我们所希望的新的周期长度（计数器被重载新的值）。

以下各图以一些示例说明不同时钟频率下计数器的行为。

图 85. 计数器时序图，1 分频内部时钟，TIMx_ARR = 0x6



1. 此处使用中心对齐模式 1（有关详细信息，请参见*第 365 页的第 14.4 节：TIM1 和 TIM8 寄存器*）。

图 86. 计数器时序图，2 分频内部时钟

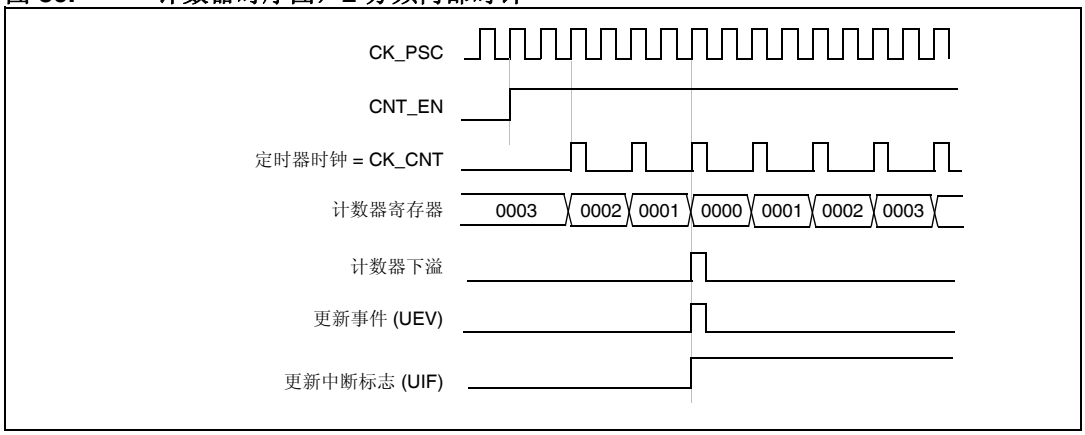
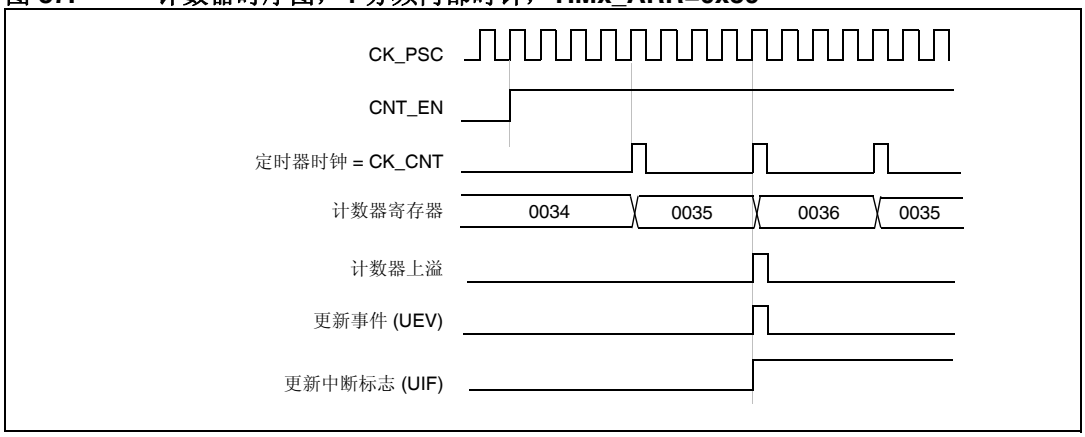


图 87. 计数器时序图，4 分频内部时钟，TIMx_ARR=0x36



1. 中心对齐模式 2 或模式 3 与上溢 UIF 结合使用。

图 88. 计数器时序图，N 分频内部时钟

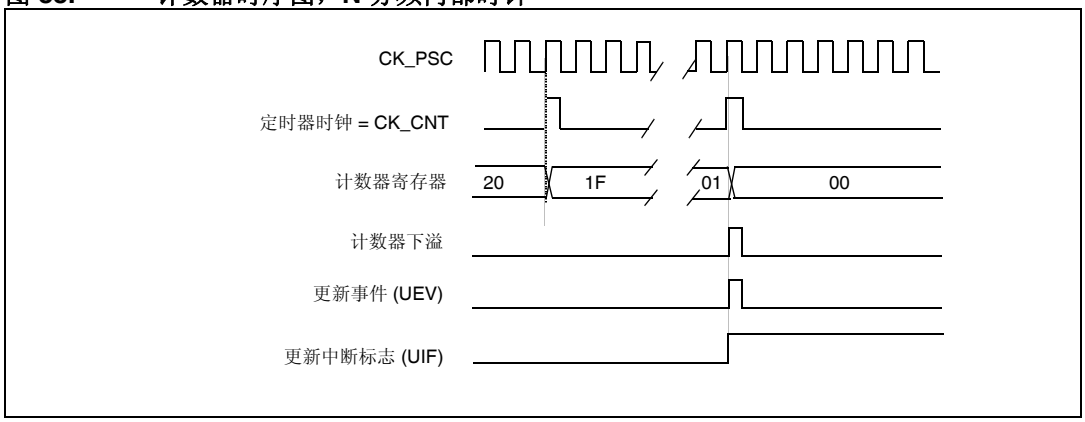


图 89. 计数器时序图，ARPE=1 时的更新事件（计数器下溢）

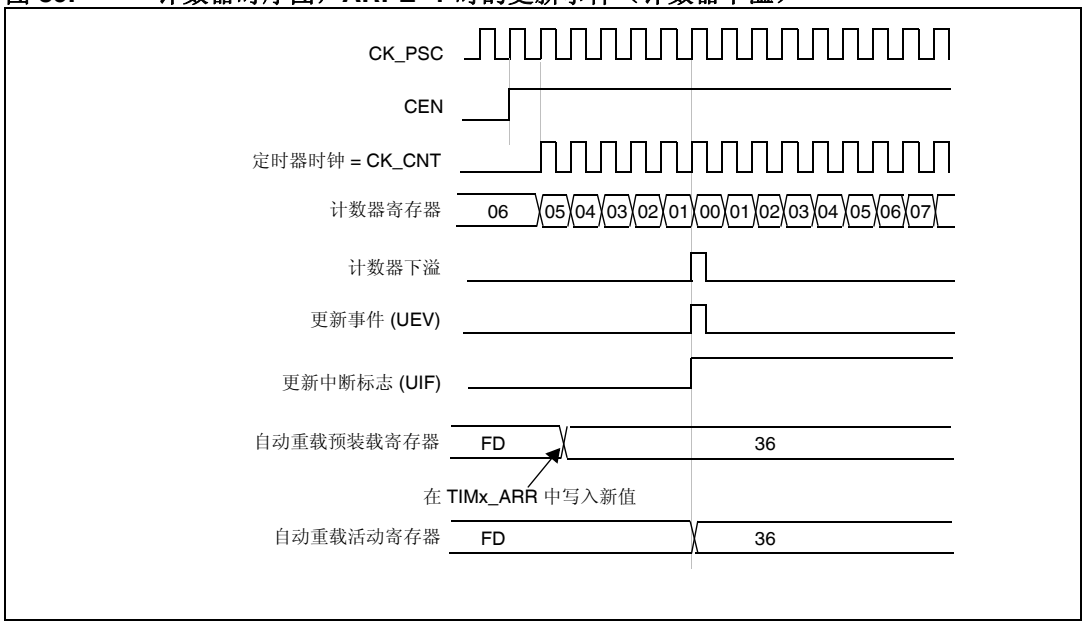
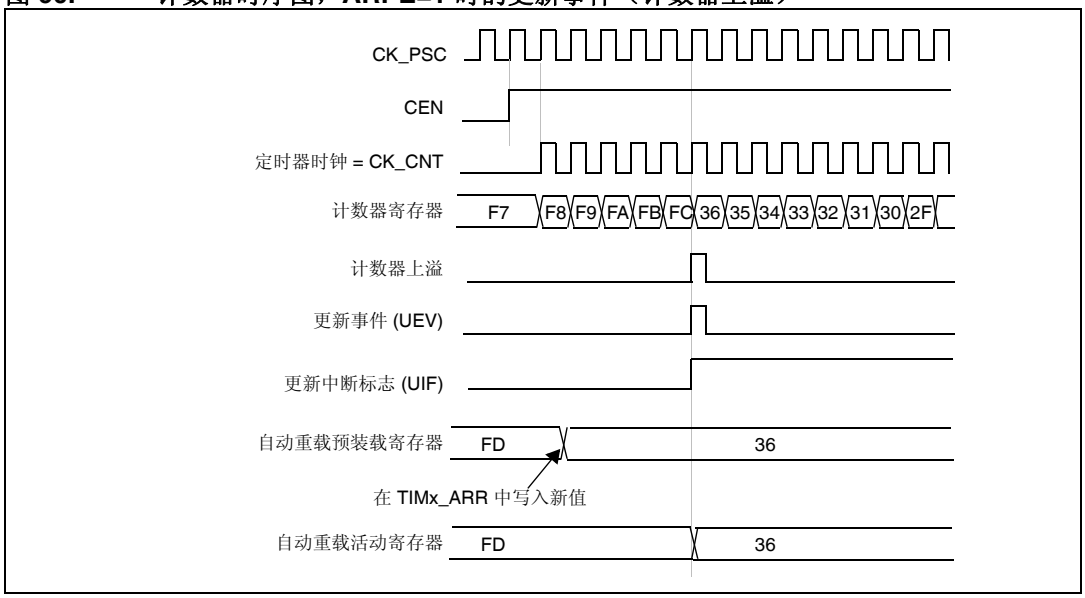


图 90. 计数器时序图，ARPE=1 时的更新事件（计数器上溢）



14.3.3 重复计数器

*第 14.3.1 节：时基单元*介绍如何因计数器上溢/下溢而生成更新事件 (UEV)。实际上，只有当重复计数器达到零时，才会生成更新事件。这在生成 PWM 信号时很有用。

这意味着，每当发生 N+1 个计数器上溢或下溢（其中，N 是 TIMx_RCR 重复计数器寄存器中的值），数据就将从预装载寄存器转移到影子寄存器（TIMx_ARR 自动重载寄存器、TIMx_PSC 预分频器寄存器以及比较模式下的 TIMx_CCRx 捕获/比较寄存器）。

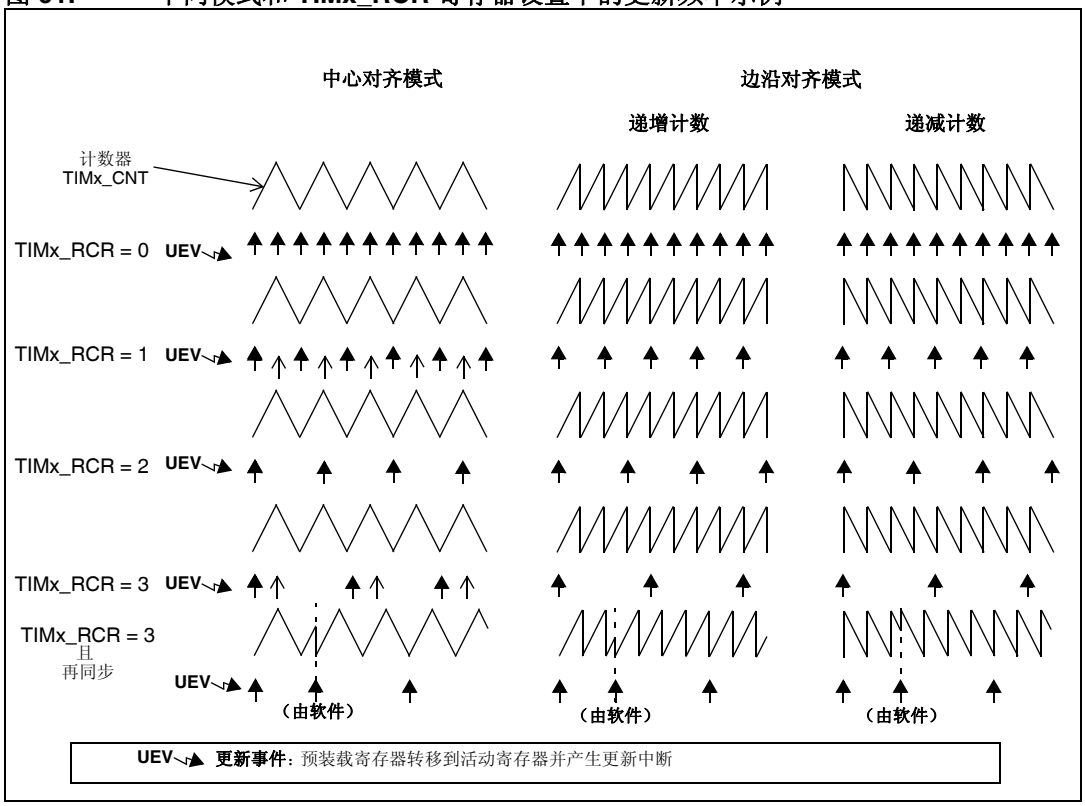
重复计数器在下列情况下递减：

- 递增计数模式下的每个计数器上溢。
- 递减计数模式下的每个计数器下溢。
- 中心对齐模式下每个计数器上溢和计数器下溢。尽管这使得最大重复次数不超过 128 个 PWM 周期，但在每个 PWM 周期内可更新占空比两次。当在中心对齐模式下，每个 PWM 周期仅刷新一次比较寄存器时，由于模式的对称性，最大分辨率为 $2xT_{ck}$。

重复计数器是自动重载类型；其重复率为 TIMx_RCR 寄存器所定义的值（请参见*图 91*）。当更新事件由软件（通过将 TIMx_EGR 寄存器的 UG 位置 1）或硬件（通过从模式控制器）生成时，无论重复计数器的值为多少，更新事件都将立即发生，并且在重复计数器中重新装载 TIMx_RCR 寄存器的内容。

在中心对齐模式下，如果 RCR 值为奇数，更新事件将在上溢或下溢时发生，这取决于何时写入 RCR 寄存器以及何时启动计数器。如果在启动计数器前写入 RCR，则 UEV 在上溢时发生。如果在启动计数器后写入 RCR，则 UEV 在下溢时发生。例如，如果 RCR = 3，UEV 将在每个周期的第四个上溢或下溢事件时生成（取决于何时写入 RCR）。

图 91. 不同模式和 TIMx_RCR 寄存器设置下的更新频率示例



14.3.4 时钟选择

计数器时钟可由下列时钟源提供：

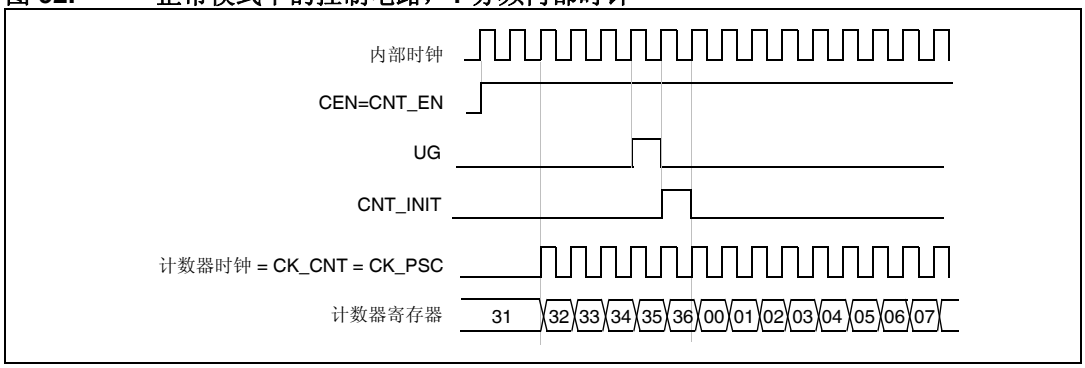
- 内部时钟 (CK_INT)
- 外部时钟模式 1：外部输入引脚
- 外部时钟模式 2：外部触发输入 ETR
- 外部触发输入 (ITRx)：使用一个定时器作为另一定时器的预分频器，例如，可将定时器 1 配置为定时器 2 的预分频器。更多详细信息，请参见*将一个定时器用作另一个定时器的预分频器*。

内部时钟源 (CK_INT)

如果禁止从模式控制器 (SMS=000)，则 CEN 位、DIR 位（TIMx_CR1 寄存器中）和 UG 位（TIMx_EGR 寄存器中）为实际控制位，并且只能通过软件进行更改（UG 除外，仍保持自动清零）。当对 CEN 位写入 1 时，预分频器的时钟就由内部时钟 CK_INT 提供。

图 92 显示了正常模式下控制电路与递增计数器的行为（没有预分频的情况下）。

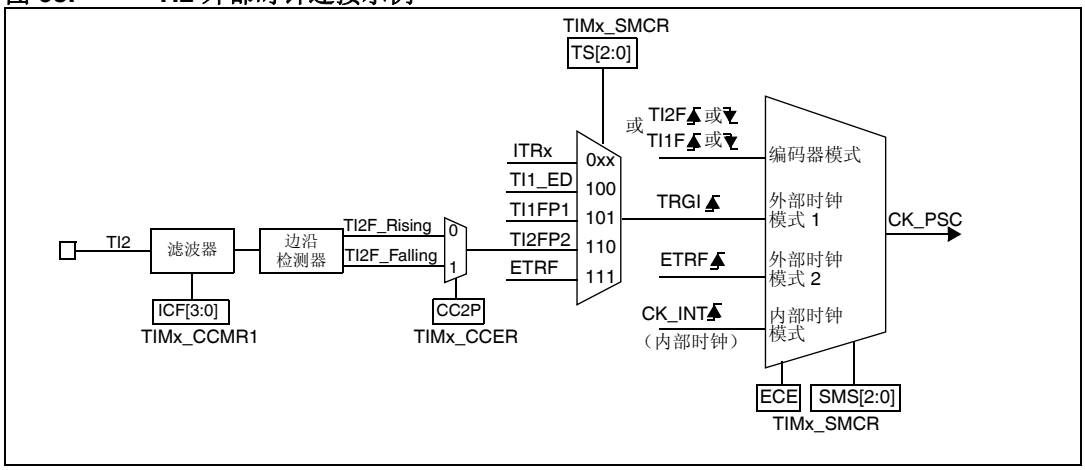
图 92. 正常模式下的控制电路，1 分频内部时钟



外部时钟源模式 1

当 TIMx_SMCR 寄存器中的 SMS=111 时，可选择此模式。计数器可在选定的输入信号上出现上升沿或下降沿时计数。

图 93. TI2 外部时钟连接示例



例如，要使递增计数器在 TI2 输入出现上升沿时计数，请执行以下步骤：

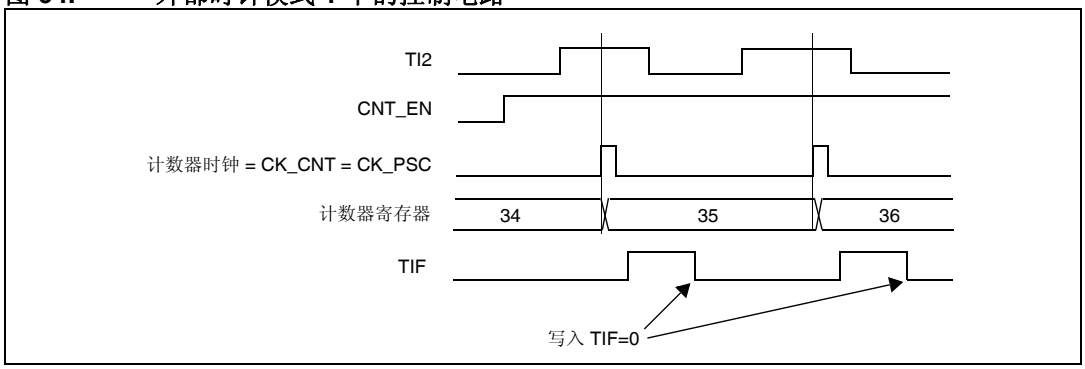
1. 通过在 TIMx_CCMR1 寄存器中写入 CC2S = “01” 来配置通道 2，使其能够检测 TI2 输入的上升沿。
2. 通过在 TIMx_CCMR1 寄存器中写入 IC2F[3:0] 位来配置输入滤波时间（如果不需要任何滤波，请保持 IC2F=0000）。
3. 通过在 TIMx_CCER 寄存器中写入 CC2P=0 和 CC2NP=0 来选择上升沿极性。
4. 通过在 TIMx_SMCR 寄存器中写入 SMS=111，使定时器在外部时钟模式 1 下工作。
5. 通过在 TIMx_SMCR 寄存器中写入 TS=110 来选择 TI2 作为触发输入源。
6. 通过在 TIMx_CR1 寄存器中写入 CEN=1 来使能计数器。

注意： *由于捕获预分频器不用于触发操作，因此无需对其进行配置。*

当 TI2 出现上升沿时，计数器便会计数一次并且 TIF 标志置 1。

TI2 的上升沿与实际计数器时钟之间的延迟是由于 TI2 输入的重新同步电路引起的。

图 94. 外部时钟模式 1 下的控制电路



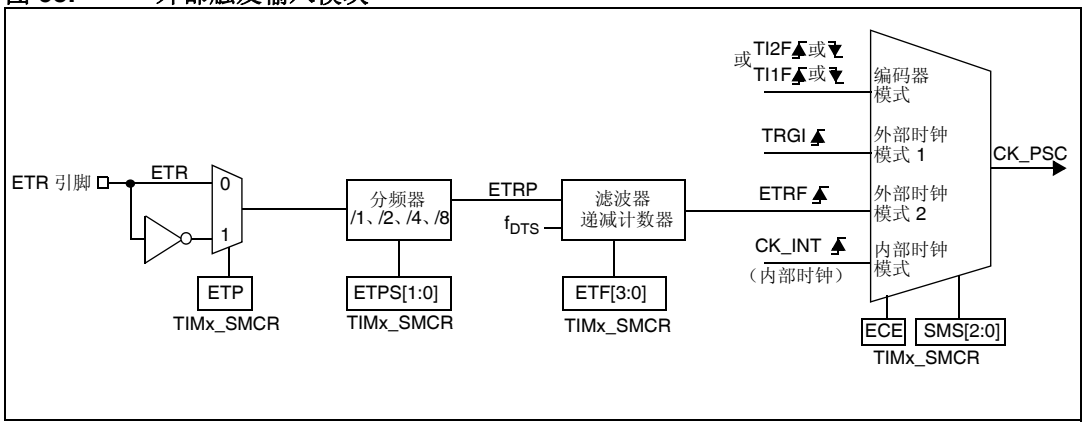
外部时钟源模式 2

通过在 TIMx_SMCR 寄存器中写入 ECE=1 可选择此模式。

计数器可在外部触发输入 ETR 出现上升沿或下降沿时计数。

图 95 简要介绍了外部触发输入模块。

图 95. 外部触发输入模块



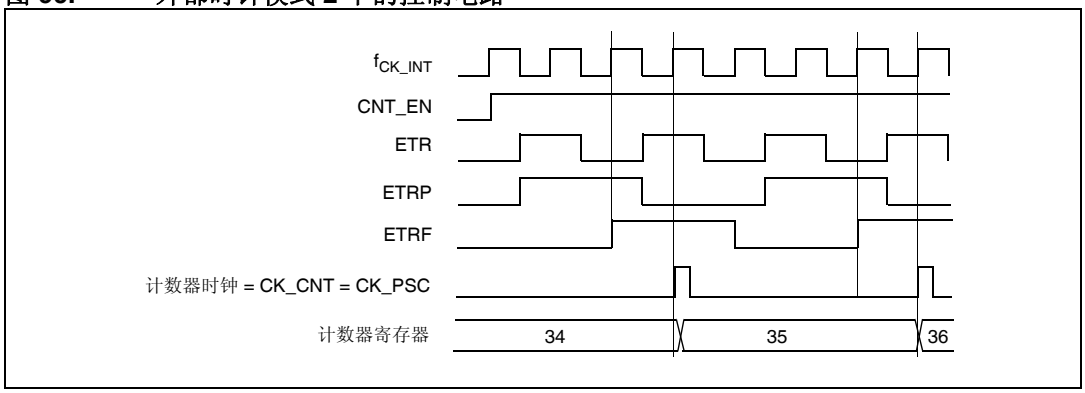
例如，要使递增计数器在 ETR 每出现 2 个上升沿时计数，请执行以下步骤：

1. 由于此例中不需滤波器，因此在 TIMx_SMCR 寄存器中写入 ETF[3:0]=0000。
2. 通过在 TIMx_SMCR 寄存器中写入 ETPS[1:0]=01 来设置预分频器。
3. 通过在 TIMx_SMCR 寄存器中写入 ETP=0 来选择 ETR 引脚的上升沿检测。
4. 通过在 TIMx_SMCR 寄存器中写入 ECE=1 来使能外部时钟模式 2。
5. 通过在 TIMx_CR1 寄存器中写入 CEN=1 来使能计数器。

ETR 每出现 2 个上升沿，计数器计数一次。

ETR 的上升沿与实际计数器时钟之间的延迟是由于 ETRP 信号的重新同步电路引起的。

图 96. 外部时钟模式 2 下的控制电路



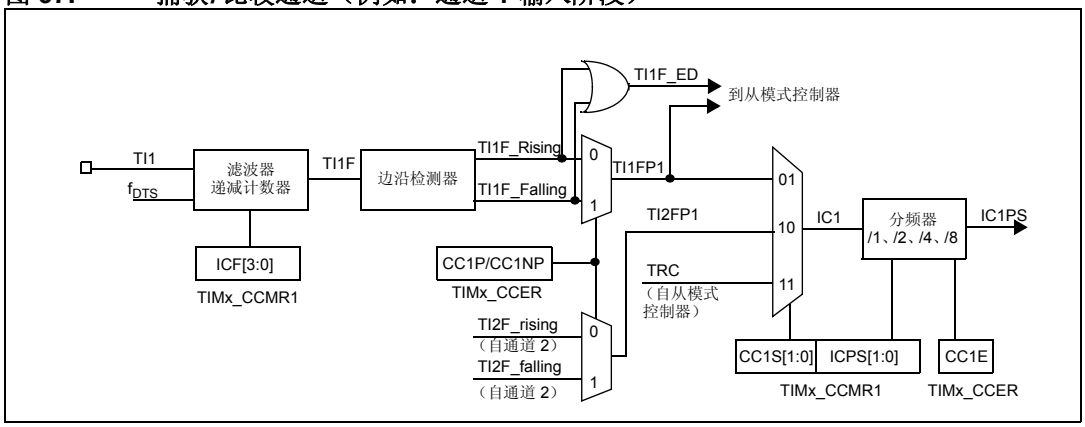
14.3.5 捕获/比较通道

每个捕获/比较通道均围绕一个捕获/比较寄存器（包括一个影子寄存器）、一个捕获输入阶段（数字滤波、多路复用和预分频器）和一个输出阶段（比较器和输出控制）构建而成。

图 97 到*图 100* 概括介绍了一个捕获/比较通道。

输入阶段对相应的 TIx 输入进行采样，生成一个滤波后的信号 TIxF。然后，带有极性选择功能的边沿检测器生成一个信号 (TIxFPx)，该信号可用作从模式控制器的触发输入，也可用作捕获命令。该信号先进行预分频 (ICxPS)，而后再进入捕获寄存器。

图 97. 捕获/比较通道（例如：通道 1 输入阶段）



输出阶段生成一个中间波形作为基准：OCxRef（高电平有效）。链的末端决定最终输出信号的极性。

图 98. 捕获/比较通道 1 主电路

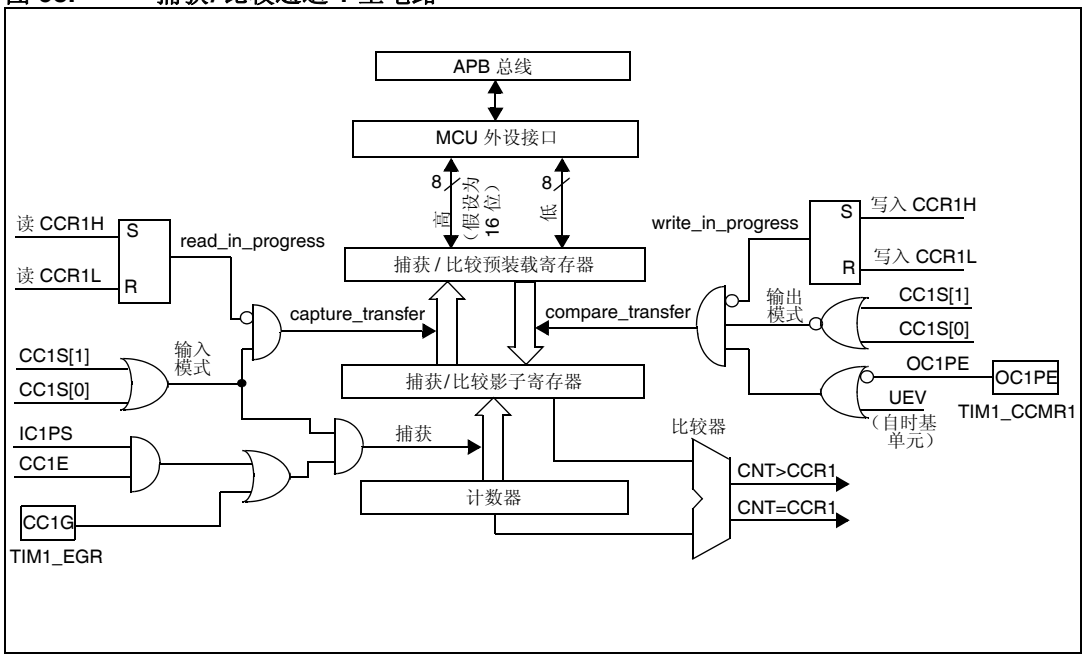


图 99. 捕获/比较通道的输出阶段（通道 1 到 3）

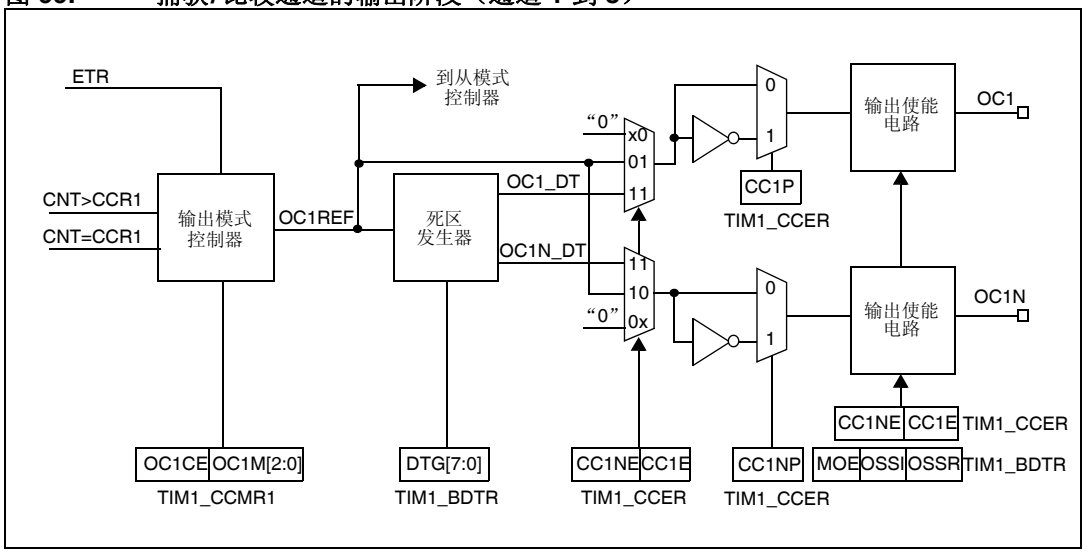
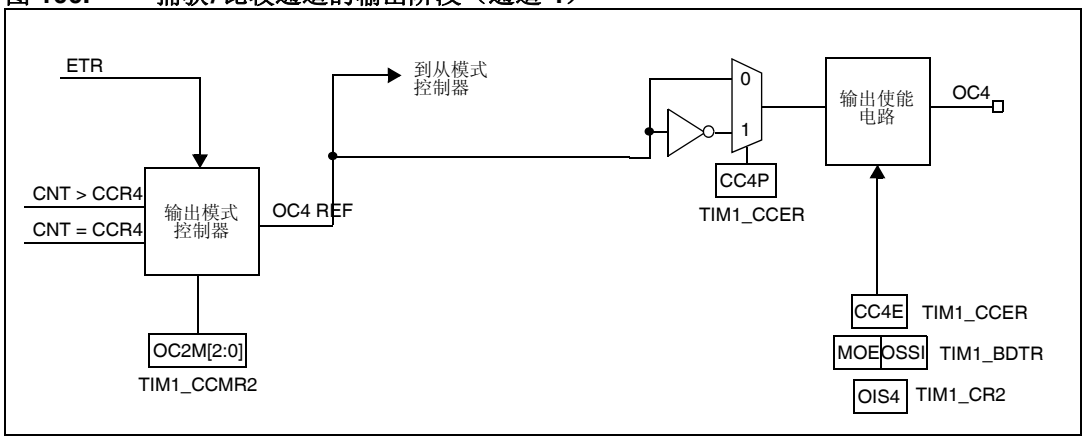


图 100. 捕获/比较通道的输出阶段（通道 4）



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。始终可通过读写操作访问预装载寄存器。

在捕获模式下，捕获实际发生在影子寄存器中，然后将影子寄存器的内容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将被复制到影子寄存器中，然后将影子寄存器的内容与计数器进行比较。

14.3.6 输入捕获模式

在输入捕获模式下，当相应的 ICx 信号检测到跳变沿后，将使用捕获/比较寄存器 (TIMx_CCRx) 来锁存计数器的值。发生捕获事件时，会将相应的 CCXIF 标志（TIMx_SR 寄存器）置 1，并可发送中断或 DMA 请求（如果已使能）。如果发生捕获事件时 CCxIF 标志已处于高位，则会将重复捕获标志 CCxOF（TIMx_SR 寄存器）置 1。可通过软件向 CCxIF 写入 0 来给 CCxIF 清零，或读取存储在 TIMx_CCRx 寄存器中的已捕获数据。向 CCxOF 写入“0”后会将其清零。

以下示例说明了如何在 TI1 输入出现上升沿时将计数器的值捕获到 TIMx_CCR1 中。具体操作步骤如下：

- 选择有效输入：TIMx_CCR1 必须连接到 TI1 输入，因此向 TIMx_CCMR1 寄存器中的 CC1S 位写入 01。只要 CC1S 不等于 00，就会将通道配置为输入模式，并且 TIMx_CCR1 寄存器将处于只读状态。
- 根据连接到定时器的信号，对所需的输入滤波时间进行编程（如果输入为 TIx 输入，则对 TIMx_CCMRx 寄存器中的 ICxF 位进行编程）。假设信号边沿变化时，输入信号最多在 5 个内部时钟周期内发生抖动。因此，我们必须将滤波时间设置为大于 5 个内部时钟周期。在检测到 8 个具有新电平的连续采样（以 f_{DTS} 频率采样）后，可以确认 TI1 上的跳变沿。然后向 TIMx_CCMR1 寄存器中的 IC1F 位写入 0011。
- 通过在 TIMx_CCER 寄存器中将 CC1P 位和 CC1NP 位写入 0，选择 TI1 上的有效转换边沿（本例中为上升沿）。
- 对输入预分频器进行编程。在本例中，我们希望每次有效转换时都执行捕获操作，因此需要禁止预分频器（向 TIMx_CCMR1 寄存器中的 IC1PS 位写入“00”）。
- 通过将 TIMx_CCER 寄存器中的 CC1E 位置 1，允许将计数器的值捕获到捕获寄存器中。
- 如果需要，可通过将 TIMx_DIER 寄存器中的 CC1IE 位置 1 来使能相关中断请求，并且/或者通过将该寄存器中的 CC1DE 位置 1 来使能 DMA 请求。

发生输入捕获时：

- 发生有效跳变沿时，TIMx_CCR1 寄存器会获取计数器的值。
- 将 CC1IF 标志置 1（中断标志）。如果至少发生了两次连续捕获，但 CC1IF 标志未被清零，这样 CC1OF 捕获溢出标志会被置 1。
- 根据 CC1IE 位生成中断。
- 根据 CC1DE 位生成 DMA 请求。

要处理重复捕获，建议在读出捕获溢出标志之前读取数据。这样可避免丢失在读取捕获溢出标志之后与读取数据之前可能出现的重复捕获信息。

注意： *通过软件将 TIMx_EGR 寄存器中的相应 CCxG 位置 1 可生成 IC 中断和/或 DMA 请求。*

14.3.7 PWM 输入模式

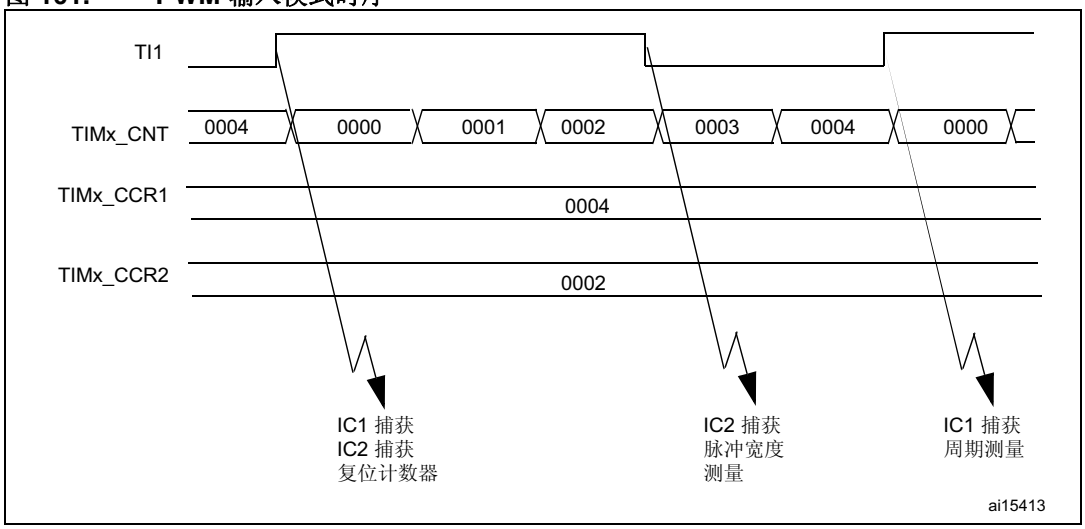
此模式是输入捕获模式的一个特例。其实现步骤与输入捕获模式基本相同，仅存在以下不同之处：

- 两个 ICx 信号被映射至同一个 TIx 输入。
- 这两个 ICx 信号在边沿处有效，但极性相反。
- 选择两个 TIxFP 信号之一作为触发输入，并将从模式控制器配置为复位模式。

例如，可通过以下步骤对应用于 TI1 的 PWM 的周期（位于 TIMx_CCR1 寄存器中）和占空比（位于 TIMx_CCR2 寄存器中）进行测量（取决于 CK_INT 频率和预分频器的值）：

- 选择 TIMx_CCR1 的有效输入：向 TIMx_CCMR1 寄存器中的 CC1S 位写入 01（选择 TI1）。
- 选择 TI1FP1 的有效极性（用于在 TIMx_CCR1 中捕获和计数器清零）：向 CC1P 位和 CC1NP 位写入“0”（上升沿有效）。
- 选择 TIMx_CCR2 的有效输入：向 TIMx_CCMR1 寄存器中的 CC2S 写入 10（选择 TI1）。
- 选择 TI1FP2 的有效极性（用于在 TIMx_CCR2 中捕获）：向 CC2P 位和 CC2NP 位写入“1”（下降沿有效）。
- 选择有效触发输入：向 TIMx_SMCR 寄存器中的 TS 位写入 101（选择 TI1FP1）。
- 将从模式控制器配置为复位模式：向 TIMx_SMCR 寄存器中的 SMS 位写入 100。
- 使能捕获：向 TIMx_CCER 寄存器中的 CC1E 位和 CC2E 位写入“1”。

图 101. PWM 输入模式时序



14.3.8 强制输出模式

在输出模式（TIMx_CCMRx 寄存器中的 CCxS 位 = 00）下，可直接由软件将每个输出比较信号（OCxREF 和 OCx/OCxN）强制设置为有效电平或无效电平，而无需考虑输出比较寄存器和计数器之间的任何比较结果。

要将输出比较信号 (OCXREF/OCx) 强制设置为有效电平，只需向相应 TIMx_CCMRx 寄存器中的 OCxM 位写入 101。OCXREF 进而强制设置为高电平（OCxREF 始终为高电平有效），同时 OCx 获取 CCxP 极性位的相反值。

例如：CCxP=0（OCx 高电平有效）=> OCx 强制设置为高电平。

通过向 TIMx_CCMRx 寄存器中的 OCxM 位写入 100，可将 OCxREF 信号强制设置为低电平。

无论如何，TIMx_CCRx 影子寄存器与计数器之间的比较仍会执行，而且允许将标志置 1。因此可发送相应的中断和 DMA 请求。下面的输出比较模式一节对此进行了介绍。

14.3.9 输出比较模式

此功能用于控制输出波形，或指示已经过某一时间段。

当捕获/比较寄存器与计数器之间相匹配时，输出比较功能：

- 将为相应的输出引脚分配一个可编程值，该值由输出比较模式（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。匹配时，输出引脚既可保持其电平 (OCXM=000)，也可设置为有效电平 (OCXM=001)、无效电平 (OCXM=010) 或进行翻转 (OCxM=011)。
- 将中断状态寄存器中的标志置 1（TIMx_SR 寄存器中的 CCxIF 位）。
- 如果相应中断使能位（TIMx_DIER 寄存器中的 CCXIE 位）置 1，将生成中断。
- 如果相应 DMA 使能位（TIMx_DIER 寄存器的 CCxDE 位，TIMx_CR2 寄存器的 CCDS 位，用来选择 DMA 请求）置 1，将发送 DMA 请求。

使用 TIMx_CCMRx 寄存器中的 OCxPE 位，可将 TIMx_CCRx 寄存器配置为带或不带预装载寄存器。

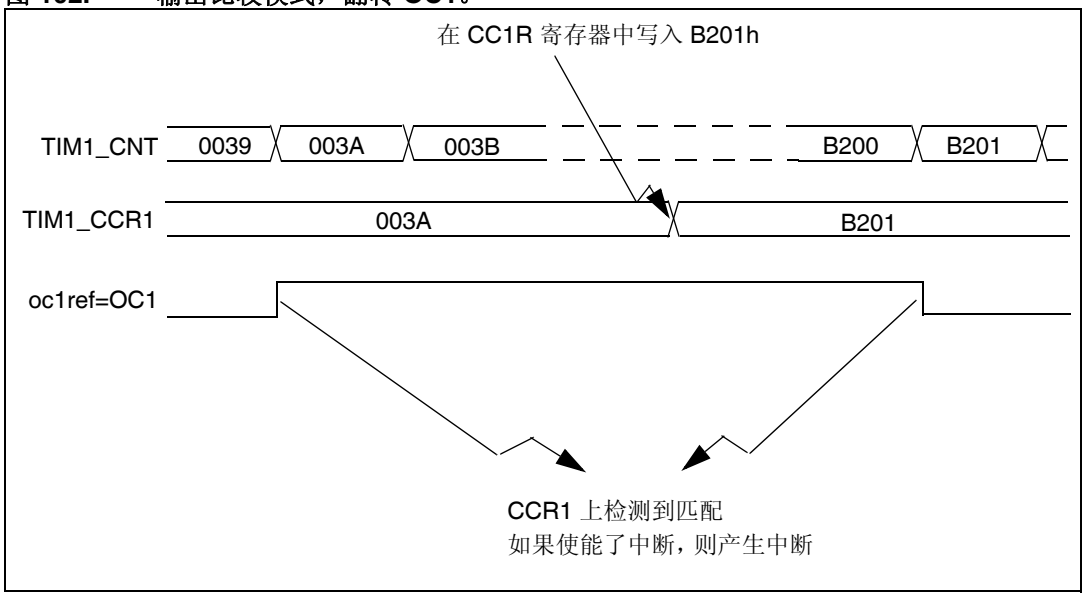
在输出比较模式下，更新事件 UEV 对 OCxREF 和 OCx 输出毫无影响。同步的精度可以达到计数器的一个计数周期。输出比较模式也可用于输出单脉冲（在单脉冲模式下）。

步骤：

1. 选择计数器时钟（内部、外部、预分频器）。
2. 在 TIMx_ARR 和 TIMx_CCRx 寄存器中写入所需数据。
3. 如果要生成中断请求，则需将 CCxIE 位置 1。
4. 选择输出模式。例如：
   - 当 CNT 与 CCRx 匹配时，写入 OCxM = 011 以翻转 OCx 输出引脚
   - 写入 OCxPE = 0 以禁止预装载寄存器
   - 写入 CCxP = 0 以选择高电平有效极性
   - 写入 CCxE = 1 以使能输出
5. 通过将 TIMx_CR1 寄存器中的 CEN 位置 1 来使能计数器。

可通过软件随时更新 TIMx_CCRx 寄存器以控制输出波形，前提是未使能预加载寄存器（OCxPE=“0”，否则仅当发生下一个更新事件 UEV 时，才会更新 TIMx_CCRx 影子寄存器）。*图 102* 给出了一个示例。

图 102. 输出比较模式，翻转 OC1。



14.3.10 PWM 模式

脉冲宽度调制模式可以生成一个信号，该信号频率由 TIMx_ARR 寄存器值决定，其占空比则由 TIMx_CCRx 寄存器值决定。

通过向 TIMx_CCMRx 寄存器中的 OCxM 位写入 110 （PWM 模式 1）或 111 （PWM 模式 2），可以独立选择各通道 （每个 OCx 输出对应一个 PWM）的 PWM 模式。必须通过将 TIMx_CCMRx 寄存器中的 OCxPE 位置 1 使能相应预装载寄存器，最后通过将 TIMx_CR1 寄存器中的 ARPE 位置 1 使能自动重载预装载寄存器（在递增计数或中心对齐模式下）。

由于只有在发生更新事件时预装载寄存器才会传送到影子寄存器，因此启动计数器之前，必须通过将 TIMx_EGR 寄存器中的 UG 位置 1 来初始化所有寄存器。

OCx 极性可使用 TIMx_CCER 寄存器的 CCxP 位来编程。既可以设为高电平有效，也可以设为低电平有效。通过 CCxE、CCxNE、MOE、OSSI 和 OSSR 位（TIMx_CCER 和 TIMx_BDTR 寄存器）的组合使能 OCx 输出。有关详细信息，请参见 TIMx_CCER 寄存器说明。

在 PWM 模式（1 或 2）下，TIMx_CNT 总是与 TIMx_CCRx 进行比较，以确定是 TIMx_CCRx≤ TIMx_CNT 还是 TIMx_CNT≤ TIMx_CCRx（取决于计数器计数方向）。

根据 TIMx_CR1 寄存器中的 CMS 位状态，定时器能够产生边沿对齐模式或中心对齐模式的 PWM 信号。

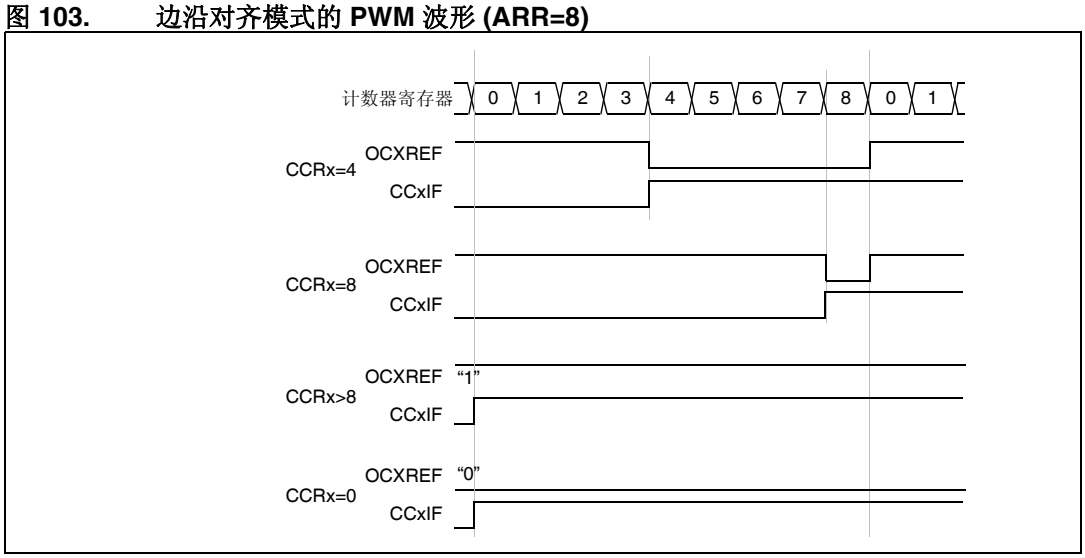
PWM 边沿对齐模式

- 递增计数配置

  当 TIMx_CR1 寄存器中的 DIR 位为低时执行递增计数。请参见*第 332 页的递增计数模式*一节。

  以下以 PWM 模式 1 为例。只要 TIMx_CNT < TIMx_CCRx，PWM 参考信号 OCxREF 便为高电平，否则为低电平。如果 TIMx_CCRx 中的比较值大于自动重载值（TIMx_ARR 中），则 OCxREF 保持为“1”。如果比较值为 0，则 OCxRef 保持为“0”。*图 103* 举例介绍边沿对齐模式的一些 PWM 波形 (TIMx_ARR=8)。

图 103. 边沿对齐模式的 PWM 波形 (ARR=8)



- 递减计数配置

  当 TIMx_CR1 寄存器中的 DIR 位为高时执行递减计数。请参见*第 335 页的递减计数模式*一节。

  在 PWM 模式 1 下，只要 TIMx_CNT > TIMx_CCRx，参考信号 OCxRef 即为低电平，否则其为高电平。如果 TIMx_CCRx 中的比较值大于 TIMx_ARR 中的自动重载值，则 OCxREF 保持为“1”。此模式下不可能产生 0% 的 PWM 波形。

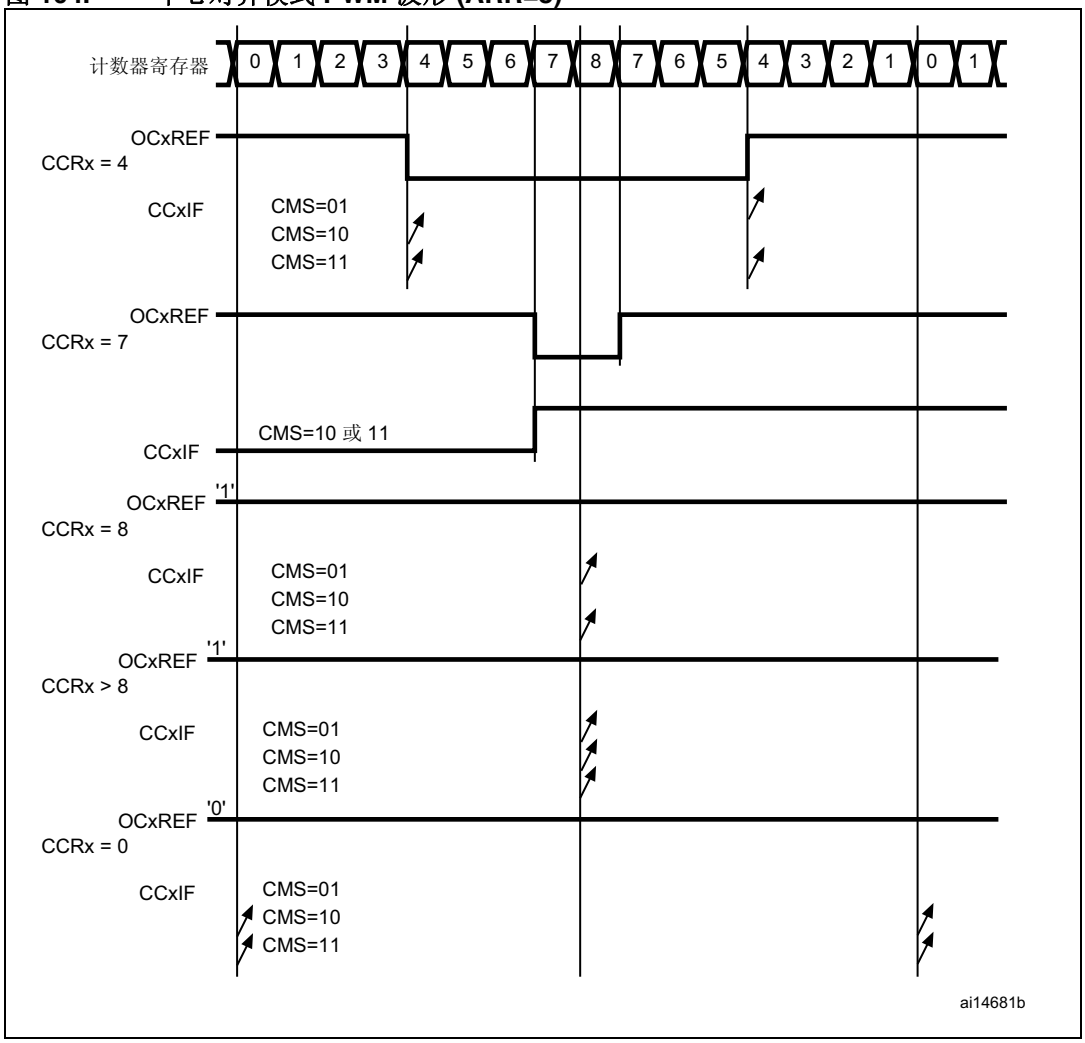
PWM 中心对齐模式

当 TIMx_CR1 寄存器中的 CMS 位不为“00”（其余所有配置对 OCxRef/OCx 信号具有相同的作用），中心对齐模式生效。根据 CMS 位的配置，可以在计数器递增计数、递减计数或同时递增和递减计数时将比较标志置 1。TIMx_CR1 寄存器中的方向位 (DIR) 由硬件更新，不得通过软件更改。请参见*第 337 页的中心对齐模式（递增/递减计数）*。

图 104 显示了中心对齐模式的 PWM 波形，在此例中：

- TIMx_ARR=8，
- PWM 模式为 PWM 模式 1，
- 在根据 TIMx_CR1 寄存器中 CMS=01 而选择的中心对齐模式 1 下，当计数器递减计数时，比较标志置 1。

图 104. 中心对齐模式 PWM 波形 (ARR=8)



中心对齐模式使用建议：

- 启动中心对齐模式时将使用当前的递增/递减计数配置。这意味着计数器将根据写入 TIMx_CR1 寄存器中 DIR 位的值进行递增或递减计数。此外，不得同时通过软件修改 DIR 和 CMS 位。
- 不建议在运行中心对齐模式时对计数器执行写操作，否则将发生意想不到的结果。尤其是：
  - 如果写入计数器中的值大于自动重载值 (TIMx_CNT>TIMx_ARR)，计数方向不会更新。例如，如果计数器之前递增计数，则继续递增计数。
  - 如果向计数器写入 0 或 TIMx_ARR 的值，计数方向会更新，但不生成更新事件 UEV。
- 使用中心对齐模式最为保险的方法是：在启动计数器前通过软件生成更新（将 TIMx_EGR 寄存器中的 UG 位置 1），并且不要在计数器运行过程中对其执行写操作。

14.3.11 互补输出和死区插入

高级控制定时器（TIM1 和 TIM8）可以输出两路互补信号，并管理输出的关断与接通瞬间。

这段时间通常称为死区，用户必须根据与输出相连接的器件及其特性（电平转换器的固有延迟、开关器件产生的延迟...）来调整死区时间

每路输出可以独立选择输出极性（主输出 OCx 或互补输出 OCxN）。可通过对 TIMx_CCER 寄存器中的 CCxP 和 CCxNP 位执行写操作来完成极性选择。

互补信号 OCx 和 OCxN 通过以下多个控制位的组合进行激活：TIMx_CCER 寄存器中的 CCxE 和 CCxNE 位以及 TIMx_BDTR 和 TIMx_CR2 寄存器中的 MOE、OISx、OISxN、OSSI 和 OSSR 位。更多详细信息，请参见*第 382 页的表 73：具有断路功能的互补通道 OCx 和 OCxN 的输出控制位*。应当注意，切换至 IDLE（MOE 下降到 0）的时刻，死区仍然有效。

CCxE 和 CCxNE 位同时置 1 并且 MOE 位置 1（如果存在断路）时，将使能死区插入。TIMx_BDTR 寄存器中的 DTG[7:0] 位用于控制所有通道的死区生成。将基于参考波形 OCxREF 生成 2 个输出 OCx 和 OCxN。如果 OCx 和 OCxN 为高电平有效：

- 输出信号 OCx 与参考信号相同，只是其上升沿相对参考上升沿存在延迟。
- 输出信号 OCxN 与参考信号相反，并且其上升沿相对参考下降沿存在延迟。

如果延迟时间大于有效输出（OCx 或 OCxN）的宽度，则不会产生相应的脉冲。

下图所示为死区发生器的输出信号与参考信号 OCxREF 之间的关系。（在这些示例中，假定 CCxP=0、CCxNP=0、MOE=1、CCxE=1 并且 CCxNE=1）

图 105. 带死区插入的互补输出。

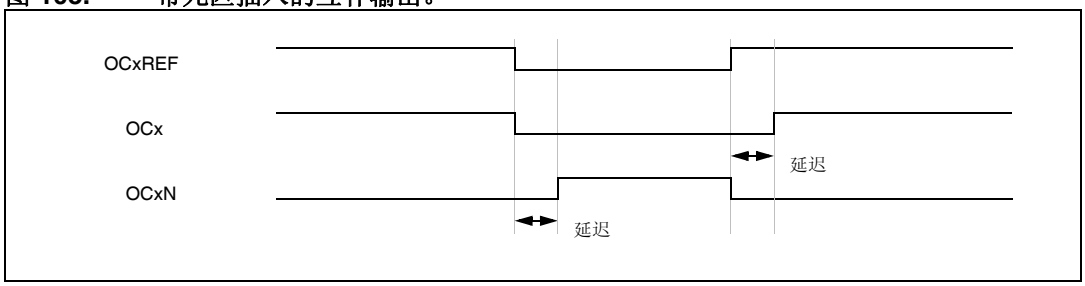


图 106. 延迟时间大于负脉冲宽度的死区波形

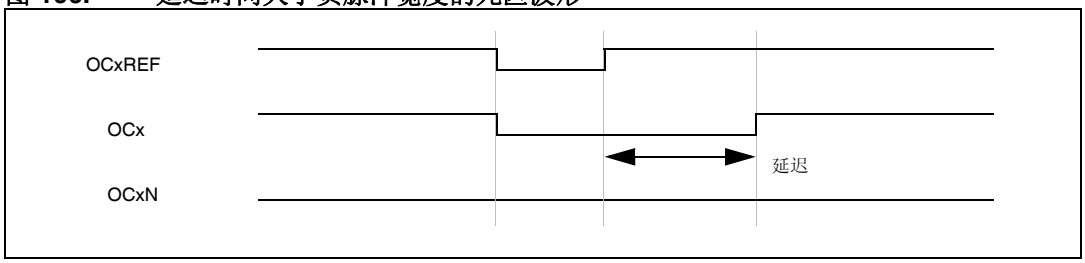
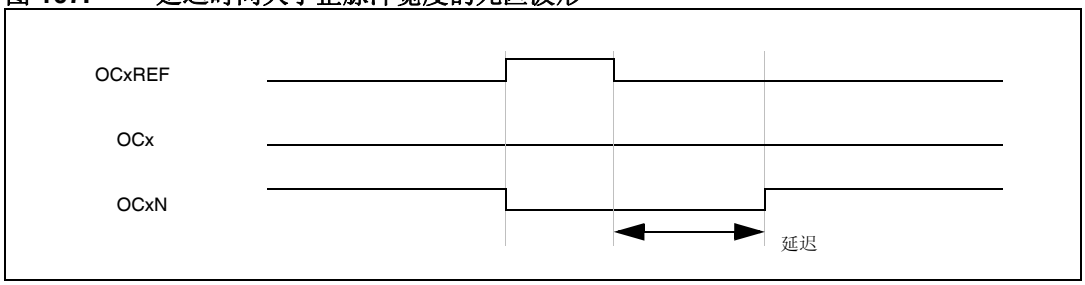


图 107. 延迟时间大于正脉冲宽度的死区波形



死区延迟对于所有通道均相同，可通过 TIMx_BDTR 寄存器中的 DTG 位进行编程。有关延迟时间计算的信息，请参见*第 386 页的第 14.4.18 节：TIM1 和 TIM8 断路和死区寄存器 (TIMx_BDTR)*。

将 OCxREF 重定向到 OCx 或 OCxN

在输出模式（强制输出模式、输出比较模式或 PWM 模式）下，通过配置 TIMx_CCER 寄存器中的 CCxE 和 CCxNE 位，可将 OCxREF 重定向到 OCx 输出或 OCxN 输出。

通过此功能，可以在一个输出上发送特定波形（如 PWM 或静态有效电平），而同时使互补输出保持其无效电平。或者，使两个输出同时保持无效电平，或者两个输出同时处于有效电平，两者互补并且带死区。

注意： *如果仅使能 OCxN (CCxE=0, CCxNE=1)，两者不互补，一旦 OCxREF 为高电平，OCxN 即变为有效。例如，如果 CCxNP=0，则 OCxN=OCxRef。另一方面，如果同时使能 OCx 和 OCxN (CCxE=CCxNE=1)，OCx 在 OCxREF 为高电平时变为有效，而 OCxN 则与之互补，在 OCxREF 为低电平时变为有效。*

14.3.12 使用断路功能

使用断路功能时，根据其它控制位（TIMx_BDTR 寄存器中的 MOE、OSSI 和 OSSR 位以及 TIMx_CR2 寄存器中的 OISx 和 OISxN 位）修改输出使能信号和无效电平。任何情况下，OCx 和 OCxN 输出都不能同时置为有效电平。更多详细信息，请参见*第 382 页的表 73：具有断路功能的互补通道 OCx 和 OCxN 的输出控制位*。

断路源可以是断路输入引脚，也可以是时钟故障事件，后者由复位时钟控制器中的时钟安全系统 (CSS) 生成。有关时钟安全系统的详细信息，请参见*第 6.2.7 节：时钟安全系统 (CSS)*。

退出复位状态后，断路功能处于禁止状态，MOE 位处于低电平。将 TIMx_BDTR 寄存器中的 BKE 位置 1，可使能断路功能。断路输入的极性可通过该寄存器中的 BKP 位来选择。BKE 和 BKP 位可同时修改。对 BKE 和 BKP 位执行写操作时，写操作会在 1 个 APB 时钟周期的延迟后生效。因此，执行写操作后，需要等待 1 个 APB 时钟周期，才能准确回读该位。

由于 MOE 下降沿可能是异步信号，因此在实际信号（作用于输出）与同步控制位（位于 TIMx_BDTR 寄存器中）之间插入了再同步电路，从而在异步信号与同步信号之间产生延迟。具体而言，如果在 MOE 处于低电平时向其写入 1，则必须首先插入延迟（空指令），才能准确进行读取。这是因为写入的是异步信号，而读取的却是同步信号。

发生断路（断路输入上出现所选电平）时：

- MOE 位异步清零，使输出处于无效状态、空闲状态或复位状态（通过 OSSI 位进行选择）。即使 MCU 振荡器关闭，该功能仍然有效。
- MOE=0 时，将以 TIMx_CR2 寄存器 OISx 位中编程的电平驱动每个输出通道。如果 OSSI=0，则定时器将释放使能输出，否则使能输出始终保持高电平。

- 使用互补输出时：
  - 输出首先置于复位状态或无效状态（取决于极性）。这是异步操作，因此即使没有为定时器提供时钟，该操作仍有效。
  - 如果定时器时钟仍存在，则将重新激活死区发生器，进而在死区后以 OISx 和 OISxN 位中编程的电平驱动输出。即使在这种情况下，也不能同时将 OCx 和 OCxN 驱动至其有效电平。请注意，MOE 进行再同步，因此死区的持续时间会比通常情况长一些（约 2 个 ck_tim 时钟周期）。
  - 如果 OSSI=0，则定时器会释放使能输出，否则只要 CCxE 位或 CCxNE 位处于高电平，使能输出就会保持或变为高电平。
- 将断路状态标志（TIMx_SR 寄存器中的 BIF 位）置 1。如果 TIMx_DIER 寄存器中的 BIE 位置 1，可产生中断。如果 TIMx_DIER 寄存器中的 BDE 位置 1，可发送 DMA 请求。
- 如果 TIMx_BDTR 寄存器中的 AOE 位置 1，则 MOE 位会在发生下一更新事件 (UEV) 时自动再次置 1。这一特性有许多用处，比如，可用于实现调节器的功能。否则，MOE 将始终保持低电平，直到再次向该位写入“1”。这种情况下，这一特性可用于确保安全。可以将断路输入连接到功率驱动器的警报、温度传感器或任何安全元件。

注意： *断路输入为电平有效。因此，当断路输入有效电平时，不能将 MOE 位置 1（自动或通过软件）。同时，不能将状态标志 BIF 清零。*

断路可由 BRK 输入生成，该输入具有可编程极性，其使能位 BKE 位于 TIMx_BDTR 寄存器中。

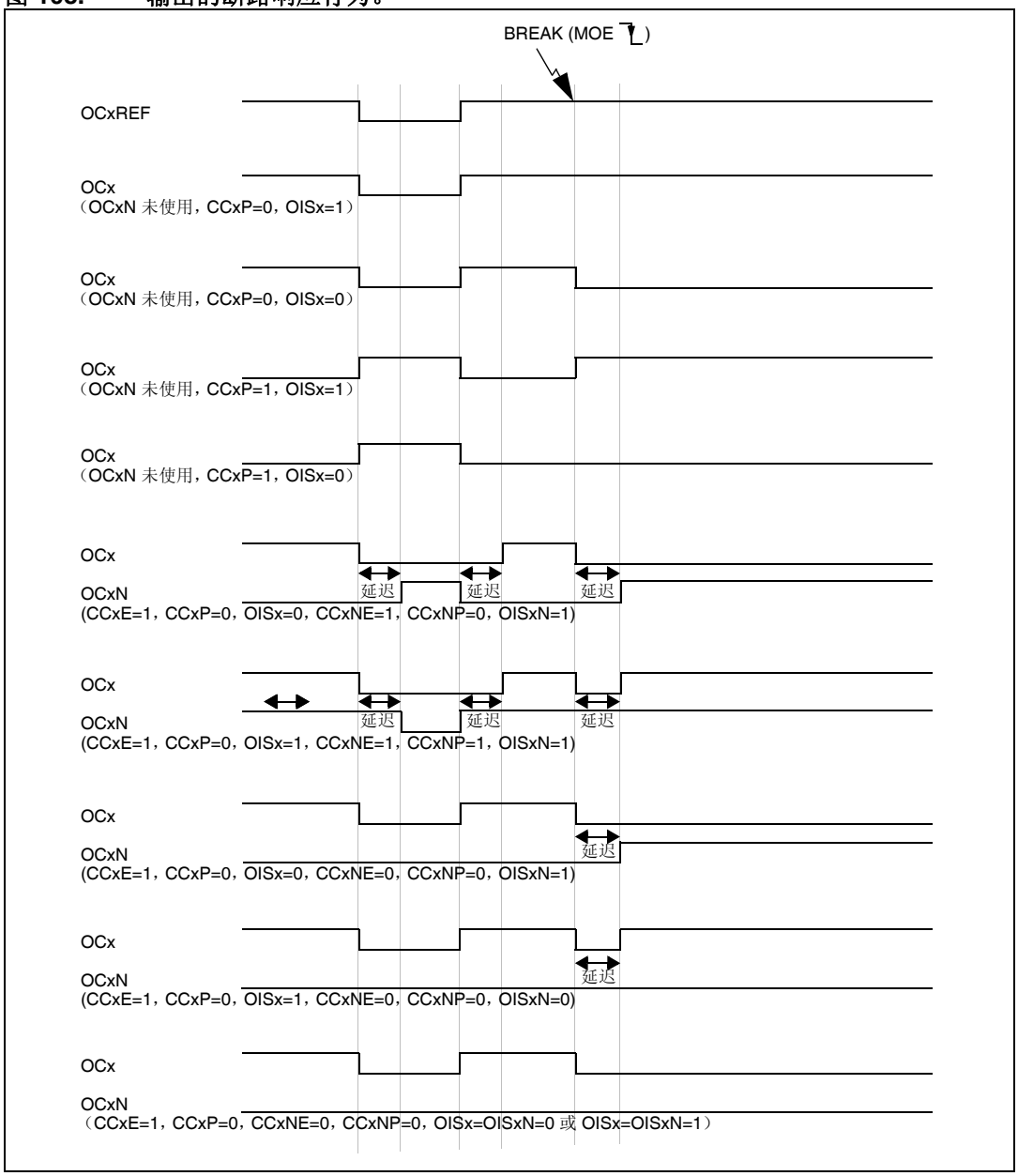
断路有以下两种生成方案：

- 使用 BRK 输入生成，该输入具有可编程极性，其使能位 BKE 位于 TIMx_BDTR 寄存器中。
- 由软件通过 TIMx_EGR 寄存器中的 BG 位生成。

除断路输入和输出管理外，断路电路内部还实施了写保护，用以保护应用的安全。通过该功能，用户可冻结多个参数配置（死区持续时间、OCx/OCxN 极性和禁止时的状态、OCxM 配置、断路使能和极性）。可以通过 TIMx_BDTR 寄存器中的 LOCK 位，从 3 种保护级别中进行选择。请参见*第 386 页的第 14.4.18 节：TIM1 和 TIM8 断路和死区寄存器 (TIMx_BDTR)*。MCU 复位后只能对 LOCK 位执行一次写操作。

图 108 所示为输出对断路响应行为的示例。

图 108. 输出的断路响应行为。



14.3.13 发生外部事件时清除 OCxREF 信号

对于给定通道，在 ETRF 输入施加高电平（相应 TIMx_CCMRx 寄存器中的 OCxCE 使能位置“1”），可使 OCxREF 信号变为低电平。OCxREF 信号将保持低电平，直到发生下一更新事件 (UEV)。

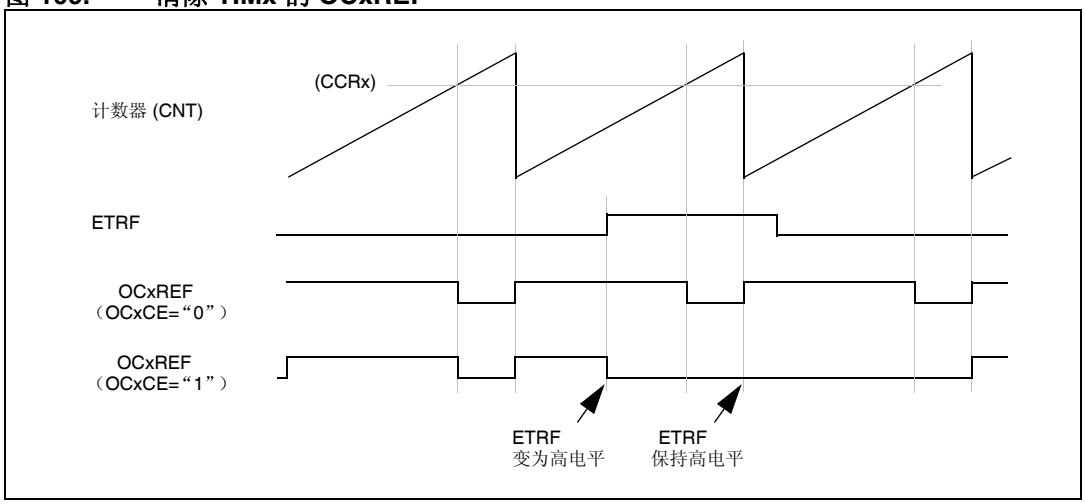
此功能仅能用于输出比较模式和 PWM 模式，而不适用于强制输出模式。

例如，ETR 信号可以连接到比较器的输出，用于控制电流。此时，ETR 必须如下配置：

1. 必须关闭外部触发预分频器：TIMx_SMCR 寄存器中的 ETPS[1:0] 位置“00”。
2. 必须禁止外部时钟模式 2：TIMx_SMCR 寄存器中的 ECE 位置“0”。
3. 外部触发极性 (ETP) 和外部触发滤波器 (ETF) 可根据用户需要进行配置。

图 109 对比了使能位 OCxCE 在不同值下的情况，显示了当 ETRF 输入变为高电平时 OCxREF 信号的行为。在本例中，定时器 TIMx 编程为 PWM 模式。

图 109. 清除 TIMx 的 OCxREF



注意： *如果 PWM 的占空比为 100% (CCRx>ARR)，则下次计数器溢出时会再次使能 OCxREF。*

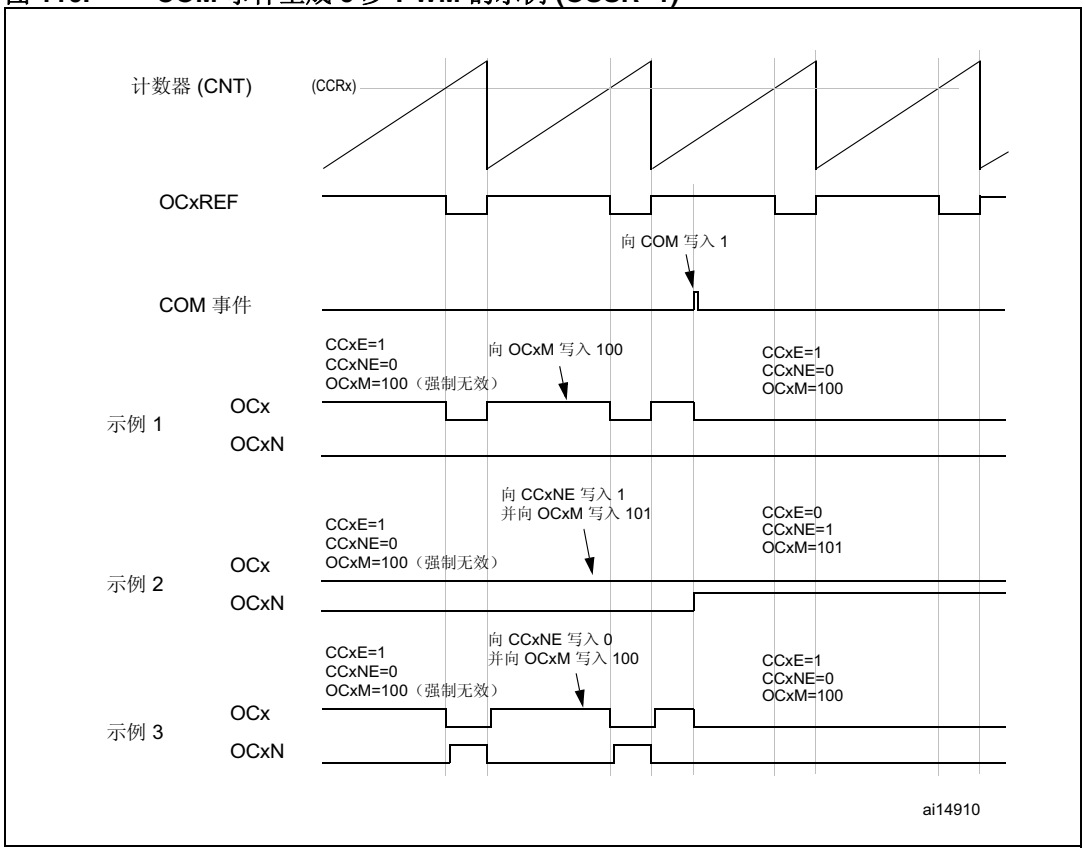
14.3.14 生成 6 步 PWM

当通道使用互补输出时，OCxM、CCxE 和 CCxNE 位上提供预装载位。发生 COM 换向事件时，这些预装载位将传输到影子位。因此，用户可以预先编程下一步骤的配置，并同时更改所有通道的配置。COM 可由软件通过将 TIMx_EGR 寄存器中的 COM 位置 1 而生成，也可以由硬件在 TRGI 上升沿生成。

发生 COM 事件时，某个标志位（TIMx_SR 寄存器中的 COMIF 位）将会置 1。这时，如果 TIMx_DIER 寄存器中的 COMIE 位置 1，将产生中断；如果 TIMx_DIER 寄存器中的 COMDE 位置 1，则将产生 DMA 请求。

图 110 以 3 种不同的编程配置为例，显示了发生 COM 事件时 OCx 和 OCxN 输出的行为。

图 110. COM 事件生成 6 步 PWM 的示例 (OSSR=1)



14.3.15 单脉冲模式

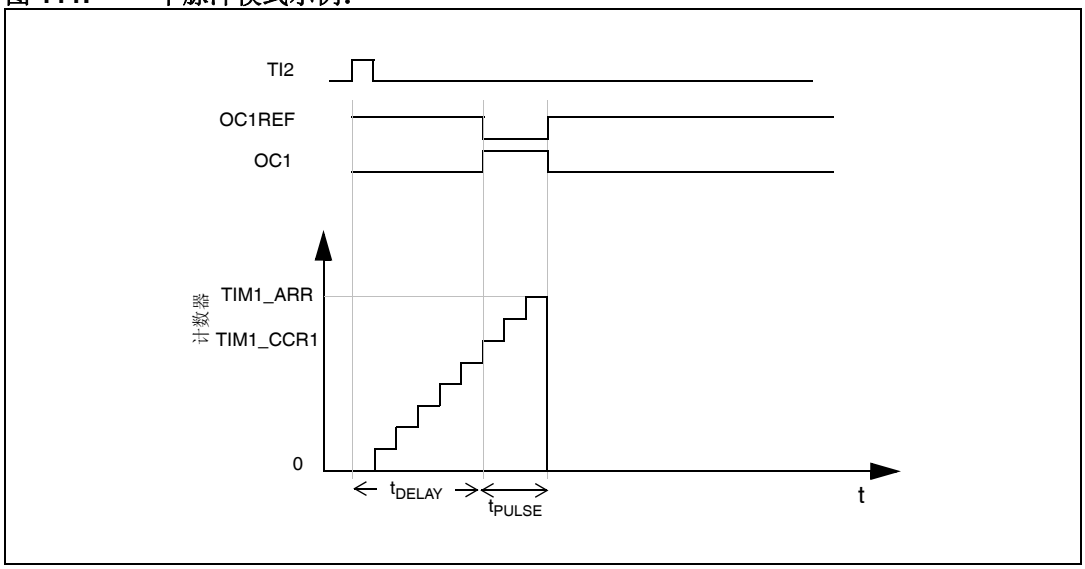
单脉冲模式 (OPM) 是上述模式的一个特例。在这种模式下，计数器可以在一个激励信号的触发下启动，并可在一段可编程的延时后产生一个脉宽可编程的脉冲。

可以通过从模式控制器启动计数器。可以在输出比较模式或 PWM 模式下生成波形。将 TIMx_CR1 寄存器中的 OPM 位置 1，即可选择单脉冲模式。这样，发生下一更新事件 UEV 时，计数器将自动停止。

只有当比较值与计数器初始值不同时，才能正确产生一个脉冲。启动前（定时器等待触发时），必须进行如下配置：

- 递增计数模式下：CNT < CCRx ≤ ARR（特别注意，0 < CCRx）
- 递减计数模式下：CNT > CCRx

图 111. 单脉冲模式示例：



例如，用户希望达到这样的效果：在 TI2 输入引脚检测到正沿时，经过 t_{DELAY} 的延迟，在 OC1 上产生一个长度为 t_{PULSE} 的正脉冲。

使用 TI2FP2 作为触发 1：

- 在 TIMx_CCMR1 寄存器中写入 CC2S=“01”，以将 TI2FP2 映射到 TI2。
- 在 TIMx_CCER 寄存器中写入 CC2P=“0”和 CC2NP=“0”，使 TI2FP2 能够检测上升沿。
- 在 TIMx_SMCR 寄存器中写入 TS=“110”，以将 TI2FP2 配置为从模式控制器的触发 (TRGI)。
- 在 TIMx_SMCR 寄存器中写入 SMS=“110”（触发模式），以使用 TI2FP2 启动计数器。

OPM 波形通过对比较寄存器执行写操作来定义（考虑时钟频率和计数器预分频器）。

- t_{DELAY} 由写入 TIMx_CCR1 寄存器的值定义。
- t_{PULSE} 由自动重载值与比较值 (TIMx_ARR - TIMx_CCR1) 之差来定义。
- 假设希望产生这样的波形：信号在发生比较匹配时从“0”变为“1”，在计数器达到自动重载值时由“1”变为“0”。为此，应在 TIMx_CCMR1 寄存器中写入 OC1M=111，以使能 PWM 模式 2。如果需要，可选择在 TIMx_CCMR1 寄存器的 OC1PE 和 TIMx_CR1 寄存器的 ARPE 中写入“1”，以使能预装载寄存器。这种情况下，必须在 TIMx_CCR1 寄存器中写入比较值并在 TIMx_ARR 寄存器中写入自动重载值，通过将 UG 位置 1 来产生更新，然后等待 TI2 上的外部触发事件。本例中，CC1P 的值为“0”。

在本例中，TIMx_CR1 寄存器中的 DIR 和 CMS 位应为低。

由于仅需要 1 个脉冲（单脉冲模式），因此应向 TIMx_CR1 寄存器的 OPM 位写入“1”，以便在发生下一更新事件（计数器从自动重载值返回到 0）时使计数器停止计数。TIMx_CR1 寄存器中的 OPM 位置“0”时，即选择重复模式。

特例：OCx 快速使能：

在单脉冲模式下，TIx 输入的边沿检测会将 CEN 位置 1，表示使能计数器。然后，在计数器值与比较值之间发生比较时，将切换输出。但是，完成这些操作需要多个时钟周期，这会限制可能的最小延迟（t_{DELAY} 最小值）。

如果要输出延迟时间最短的波形，可以将 TIMx_CCMRx 寄存器中的 OCxFE 位置 1。这样会强制 OCxRef（和 OCx）对激励信号做出响应，而不再考虑比较的结果。其新电平与发生比较匹配时相同。仅当通道配置为 PWM1 或 PWM2 模式时，OCxFE 才会起作用。

14.3.16 编码器接口模式

选择编码器接口模式时，如果计数器仅在 TI2 边沿处计数，在 TIMx_SMCR 寄存器中写入 SMS=“001”；如果计数器仅在 TI1 边沿处计数，写入 SMS=“010”；如果计数器在 TI1 和 TI2 边沿处均计数，则写入 SMS=“011”。

通过编程 TIMx_CCER 寄存器的 CC1P 和 CC2P 位，选择 TI1 和 TI2 极性。如果需要，还可对输入滤波器进行编程。CC1NP 和 CC2NP 必须保持低电平。

TI1 和 TI2 两个输入用于连接增量编码器。请参见 *表 71*。如果使能计数器（在 TIMx_CR1 寄存器的 CEN 位中写入“1”），则计数器的时钟由 TI1FP1 或 TI2FP2 上的每次有效信号转换提供。TI1FP1 和 TI2FP2 是进行输入滤波器和极性选择后 TI1 和 TI2 的信号，如果不进行滤波和反相，则 TI1FP1=TI1，TI2FP2=TI2。将根据两个输入的信号转换序列，产生计数脉冲和方向信号。根据该信号转换序列，计数器相应递增或递减计数，同时硬件对 TIMx_CR1 寄存器的 DIR 位进行相应修改。任何输入（TI1 或 TI2）发生信号转换时，都会计算 DIR 位，无论计数器是仅在 TI1 或 TI2 边沿处计数，还是同时在 TI1 和 TI2 处计数。

编码器接口模式就相当于带有方向选择的外部时钟。这意味着，计数器仅在 0 到 TIMx_ARR 寄存器中的自动重载值之间进行连续计数（根据具体方向，从 0 递增计数到 ARR，或从 ARR 递减计数到 0）。因此，在启动前必须先配置 TIMx_ARR。同样，捕获、比较、预分频器、重复计数器及触发输出功能继续正常工作。编码器模式和外部时钟模式 2 不兼容，因此不能同时选择。

在此模式下，计数器会根据增量编码器的速度和方向自动进行修改，因此，其内容始终表示编码器的位置。计数方向对应于所连传感器的旋转方向。下表汇总了可能的组合（假设 TI1 和 TI2 不同时切换）。

表 71. 计数方向与编码器信号的关系

有效边沿	相反信号的电平（TI1FP1 对应 TI2，TI2FP2 对应 TI1）	TI1FP1 信号		TI2FP2 信号	
		上升	下降	上升	下降
仅在 TI1 处计数	高	递减	递增	不计数	不计数
	低	递增	递减	不计数	不计数
仅在 TI2 处计数	高	不计数	不计数	递增	递减
	低	不计数	不计数	递减	递增
在 TI1 和 TI2 处均计数	高	递减	递增	递增	递减
	低	递增	递减	递减	递增

外部增量编码器可直接与 MCU 相连，无需外部接口逻辑。不过，通常使用比较器将编码器的差分输出转换为数字信号。这样大幅提高了抗噪声性能。用于指示机械零位的第三个编码器输出可与外部中断输入相连，用以触发计数器复位。

图 112 以计数器工作为例，说明了计数信号的生成和方向控制。同时也说明了选择双边沿时如何对输入抖动进行补偿。将传感器靠近其中一个切换点放置时可能出现这种情况。本例中假设配置如下：

- CC1S=“01”（TIMx_CCMR1 寄存器，TI1FP1 映射到 TI1 上）。
- CC2S=“01”（TIMx_CCMR2 寄存器，TI1FP2 映射到 TI2 上）。
- CC1P=“0”，CC1NP=“0”，且 IC1F =“0000”（TIMx_CCER 寄存器，TI1FP1 未反相，TI1FP1=TI1）。
- CC2P=“0”，CC2NP=“0”，且 IC2F =“0000”（TIMx_CCER 寄存器，TI1FP2 未反相，TI1FP2= TI2）。
- SMS=“011”（TIMx_SMCR 寄存器，两个输入在上升沿和下降沿均有效）。
- CEN=“1”（TIMx_CR1 寄存器，使能计数器）。

图 112. 编码器接口模式下的计数器工作示例。

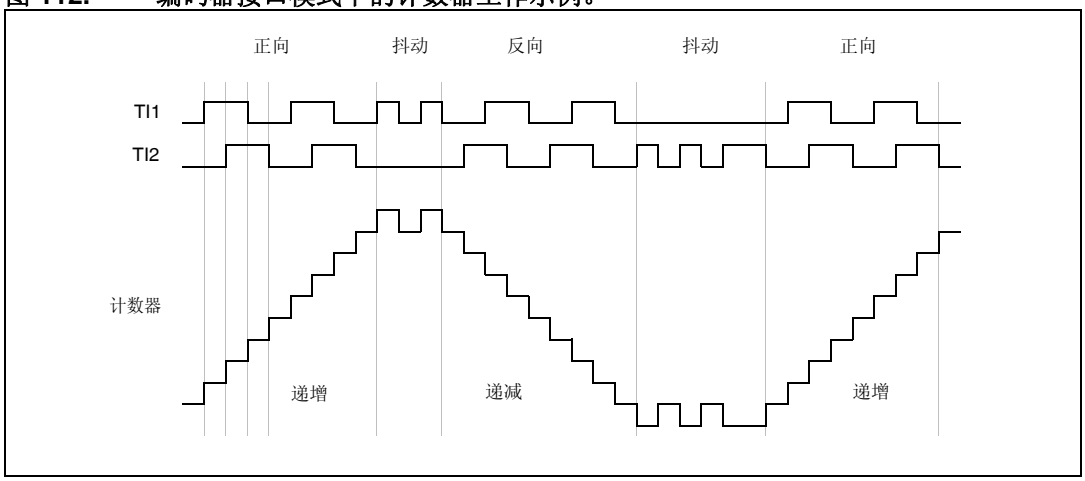
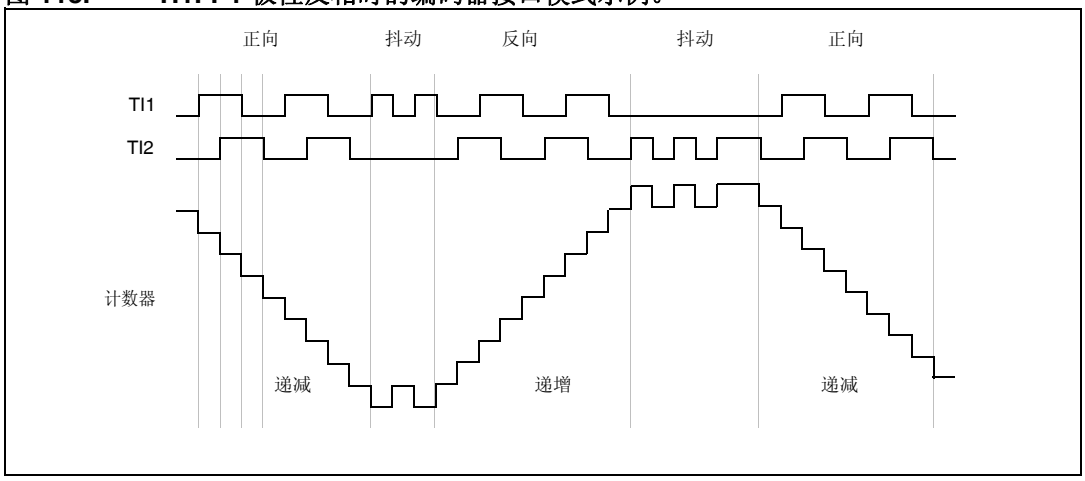


图 113 举例说明 TI1FP1 极性反相时计数器的行为（除 CC1P=“1” 外，其它配置与上例相同）。

图 113. TI1FP1 极性反相时的编码器接口模式示例。



定时器配置为编码器接口模式时，会提供传感器当前位置的相关信息。使用另一个配置为捕获模式的定时器测量两个编码器事件之间的周期，可获得动态信息（速度、加速度和减速度）。指示机械零位的编码器输出即可用于此目的。根据两个事件之间的时间间隔，还可定期读取计数器。如果可能，可以将计数器值锁存到第三个输入捕获寄存器来实现此目的（捕获信号必须为周期性信号，可以由另一个定时器产生）；还可以通过由实时时钟生成的 DMA 请求读取计数器值。

14.3.17 定时器输入异或功能

通过 TIMx_CR2 寄存器中的 TI1S 位，可将通道 1 的输入滤波器连接到异或门的输出，从而将 TIMx_CH1 到 TIMx_CH3 这三个输入引脚组合在一起。

异或输出可与触发或输入捕获等所有定时器输入功能配合使用。下面的*第 14.3.18 节*以连接霍尔传感器为例介绍了此功能。

14.3.18 连接霍尔传感器

可通过用于生成电机驱动 PWM 信号的高级控制定时器（TIM1 或 TIM8）以及*图 114* 中称为“接口定时器”的另一个定时器 TIMx（TIM2、TIM3、TIM4 或 TIM5），实现与霍尔传感器的连接。3 个定时器输入引脚（TIMx_CH1、TIMx_CH2 和 TIMx_CH3）通过异或门连接到 TI1 输入通道（通过将 TIMx_CR2 寄存器中的 TI1S 位置 1 来选择），并由“接口定时器”进行捕获。

从模式控制器配置为复位模式；从输入为 TI1F_ED。这样，每当 3 个输入中有一个输入发生切换时，计数器会从 0 开始重新计数。这样将产生由霍尔输入的任何变化而触发的时基。

在“接口定时器”上，捕获/比较通道 1 配置为捕获模式，捕获信号为 TRC（请参见*第 344 页的图 97：捕获/比较通道（例如：通道 1 输入阶段）*）。捕获值对应于输入上两次变化的间隔时间，可提供与电机转速相关的信息。

“接口定时器”可用于在输出模式下产生脉冲，以通过触发 COM 事件更改高级控制定时器（TIM1 或 TIM8）各个通道的配置。TIM1 定时器用于生成电机驱动 PWM 信号。为此，必须对接口定时器通道进行编程，以便在编程的延迟过后产生正脉冲（在输出比较或 PWM 模式中）。该脉冲通过 TRGO 输出发送到高级控制定时器（TIM1 或 TIM8）。

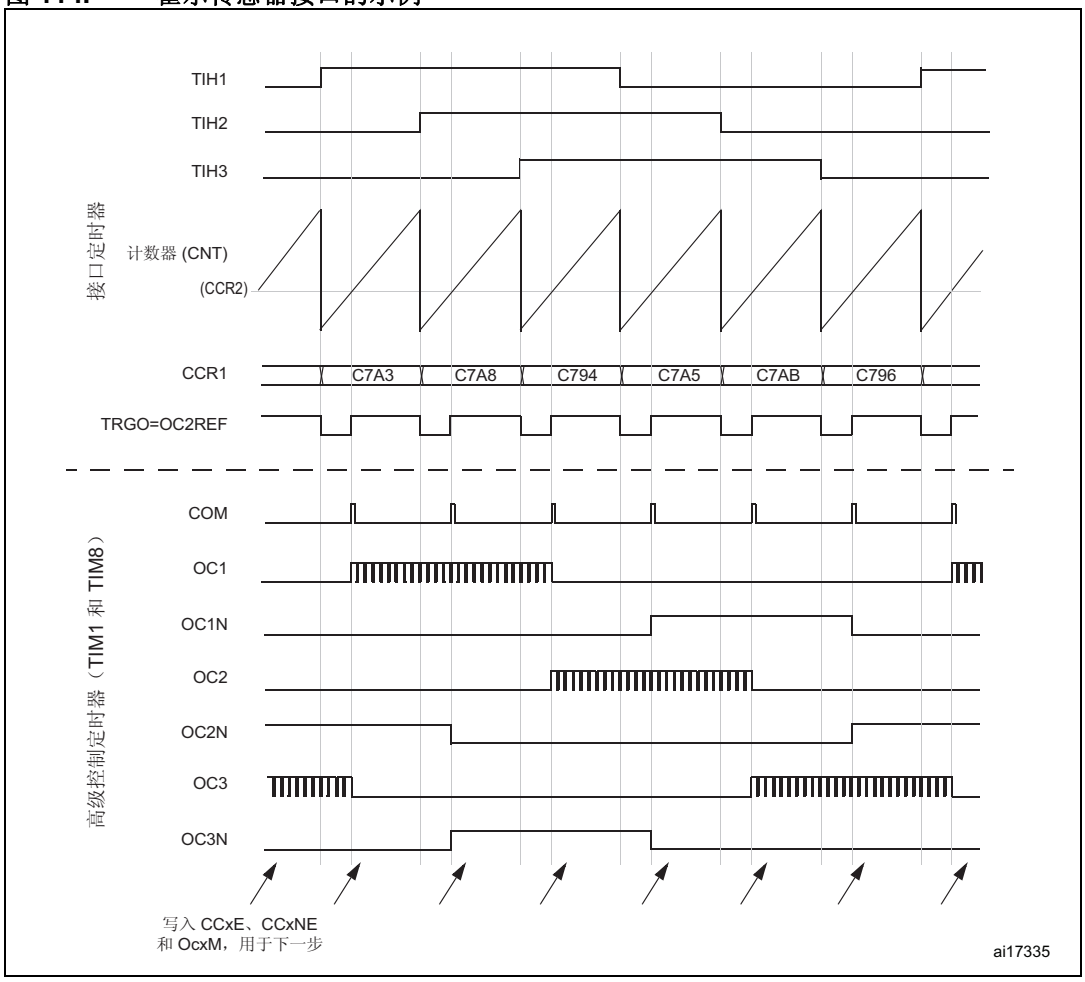
示例：霍尔输入与一个 TIMx 定时器相连接，每当霍尔输入发生更改，需要在所编程的延迟过后更改高级控制定时器 TIM1 的 PWM 配置。

- 向 TIMx_CR2 寄存器的 TI1S 位写入“1”，使 3 个定时器输入经过异或运算后进入 TI1 输入通道。
- 时基编程：向 TIMx_ARR 写入其最大值（计数器必须通过 TI1 的变化清零）。设置预分频器，以得到最大计数器周期，该周期长于传感器上两次变化的间隔时间。
- 将通道 1 编程为捕获模式（选择 TRC）：向 TIMx_CCMR1 寄存器的 CC1S 位写入“11”。如果需要，还可以编程数字滤波器。
- 将通道 2 编程为 PWM 2 模式，并具有所需延迟：向 TIMx_CCMR1 寄存器的 OC2M 位写入“111”，CC2S 位写入“00”。
- 选择 OC2REF 作为 TRGO 上的触发输出：向 TIMx_CR2 寄存器的 MMS 位写入“101”。

在高级控制定时器 TIM1 中，必须选择正确的 ITR 输入作为触发输入，定时器编程为可产生 PWM 信号，捕获/比较控制信号进行预装载（TIMx_CR2 寄存器的 CCPC=1），并且 COM 事件由触发输入控制（TIMx_CR2 寄存器中 CCUS=1）。发生 COM 事件后，在 PWM 控制位（CCxE、OCxM）中写入下一步的配置，此操作可在由 OC2REF 上升沿产生的中断子程序中完成。

图 114 为本示例的示意图。

图 114. 霍尔传感器接口的示例



14.3.19 TIMx 与外部触发同步

TIMx 定时器可与外部触发以下列模式实现同步：复位模式、门控模式和触发模式。

从模式：复位模式

当触发输入信号产生变化时，计数器及其预分频器可重新初始化。此外，如果 TIMx_CR1 寄存器中的 URS 位处于低电平，则会生成更新事件 UEV。然后，所有预装载寄存器（TIMx_ARR 和 TIMx_CCRx）都将更新。

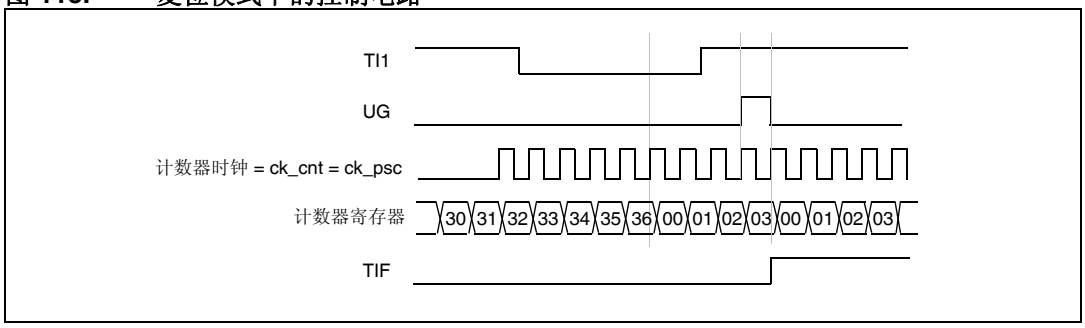
在以下示例中，TI1 输入上出现上升沿时，递增计数器清零：

- 将通道 1 配置为检测 TI1 的上升沿。配置输入滤波时间（本例中不需要任何滤波，因此保持 IC1F=0000）。由于捕获预分频器不用于触发操作，因此无需对其进行配置。CC1S 位只选择输入捕获源，即 TIMx_CCMR1 寄存器中的 CC1S = 01。在 TIMx_CCER 寄存器中写入 CC1P=0 和 CC1NP='0'，验证极性（仅检测上升沿）。
- 在 TIMx_SMCR 寄存器中写入 SMS=100，将定时器配置为复位模式。在 TIMx_SMCR 寄存器中写入 TS=101，选择 TI1 作为输入源。
- 在 TIMx_CR1 寄存器中写入 CEN=1，启动计数器。

计数器使用内部时钟计数，然后正常运转，直到出现 TI1 上升沿。当 TI1 出现上升沿时，计数器清零，然后重新从 0 开始计数。同时，触发标志（TIMx_SR 寄存器中的 TIF 位）置 1，使能中断或 DMA 后，还可发送中断或 DMA 请求（取决于 TIMx_DIER 寄存器中的 TIE 和 TDE 位）。

下图显示了自动重载寄存器 TIMx_ARR=0x36 时的相关行为。TI1 的上升沿与实际计数器复位之间的延迟是由于 TI1 输入的重新同步电路引起的。

图 115. 复位模式下的控制电路



从模式：门控模式

输入信号的电平可用来使能计数器。

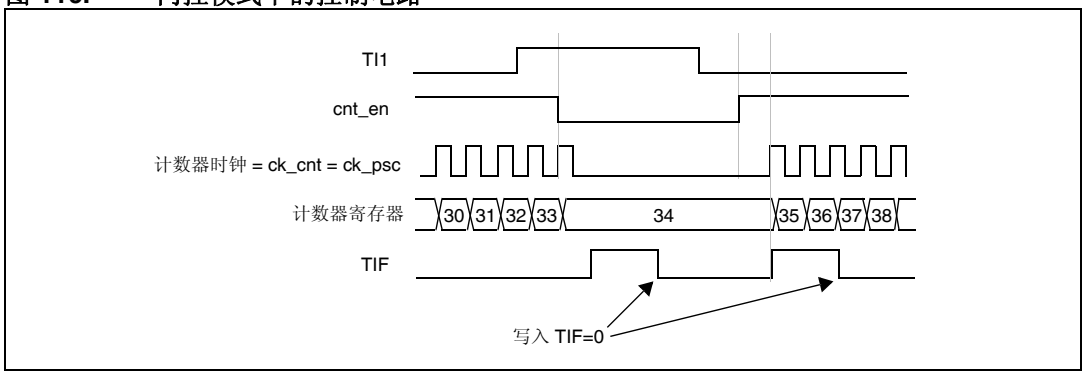
在以下示例中，递增计数器仅在 TI1 输入为低电平时计数：

- 将通道 1 配置为检测 TI1 上的低电平。配置输入滤波时间（本例中不需要任何滤波，因此保持 IC1F=0000）。由于捕获预分频器不用于触发操作，因此无需对其进行配置。CC1S 位只选择输入捕获源，即 TIMx_CCMR1 寄存器中的 CC1S=01。在 TIMx_CCER 寄存器中写入 CC1P=1 和 CC1NP=“0”，以确定极性（仅检测低电平）。
- 在 TIMx_SMCR 寄存器中写入 SMS=101，将定时器配置为门控模式。在 TIMx_SMCR 寄存器中写入 TS=101，选择 TI1 作为输入源。
- 在 TIMx_CR1 寄存器中写入 CEN=1，使能计数器（在门控模式下，如果 CEN=0，则无论触发输入电平如何，计数器都不启动）。

只要 TI1 为低电平，计数器就开始根据内部时钟计数，直到 TI1 变为高电平时停止计数。计数器启动或停止时，TIMx_SR 寄存器中的 TIF 标志都会置 1。

TI1 的上升沿与实际计数器停止之间的延迟是由于 TI1 输入的重新同步电路引起的。

图 116. 门控模式下的控制电路



从模式：触发模式

所选输入上发生某一事件时可以启动计数器。

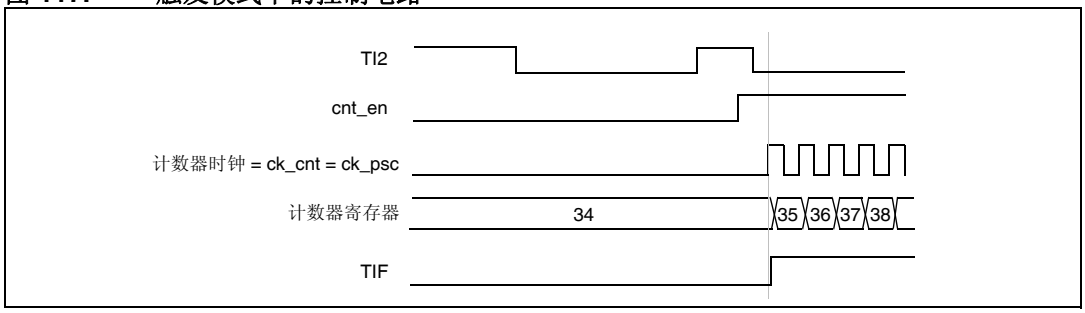
在以下示例中，TI2 输入上出现上升沿时，递增计数器启动：

- 将通道 2 配置为检测 TI2 上的上升沿。配置输入滤波时间（本例中不需要任何滤波，因此保持 IC2F=0000）。由于捕获预分频器不用于触发操作，因此无需对其进行配置。CC2S 位只选择输入捕获源，即 TIMx_CCMR1 寄存器中的 CC2S=01。在 TIMx_CCER 寄存器中写入 CC2P=1 和 CC2NP=0，以确定极性（仅检测低电平）。
- 在 TIMx_SMCR 寄存器中写入 SMS=110，将定时器配置为触发模式。在 TIMx_SMCR 寄存器中写入 TS=110，选择 TI2 作为输入源。

当 TI2 出现上升沿时，计数器开始根据内部时钟计数，并且 TIF 标志置 1。

TI2 的上升沿与实际计数器启动之间的延迟是由于 TI2 输入的重新同步电路引起的。

图 117. 触发模式下的控制电路



从模式：外部时钟模式 2 + 触发模式

外部时钟模式 2 可与另一种从模式（外部时钟模式 1 和编码器模式除外）结合使用。这种情况下，ETR 信号用作外部时钟输入，在复位模式、门控模式或触发模式下工作时，可选择另一个输入作为触发输入。不建议通过 TIMx_SMCR 寄存器中的 TS 位来选择 ETR 作为 TRGI。

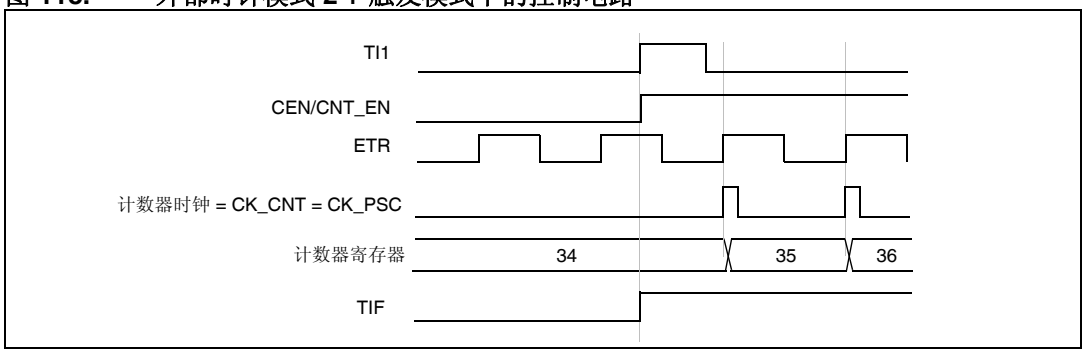
在以下示例中，只要 TI1 出现上升沿，递增计数器即会在 ETR 信号的每个上升沿处递增：

1. 通过对 TIMx_SMCR 寄存器进行如下编程，配置外部触发输入电路：
   - ETF = 0000：无滤波器
   - ETPS = 00：禁止预分频器
   - ETP = 0：检测 ETR 的上升沿，并写入 ECE=1，以使能外部时钟模式 2。
2. 如下配置通道 1，以检测 TI 的上升沿：
   - IC1F=0000：无滤波器。
   - 由于捕获预分频器不用于触发操作，因此无需对其进行配置。
   - TIMx_CCMR1 寄存器中 CC1S=01，只选择输入捕获源。
   - TIMx_CCER 寄存器中 CC1P=0 且 CC1NP=“0”，以确定极性（仅检测上升沿）。
3. 在 TIMx_SMCR 寄存器中写入 SMS=110，将定时器配置为触发模式。在 TIMx_SMCR 寄存器中写入 TS=101，选择 TI1 作为输入源。

TI1 出现上升沿时将使能计数器并且 TIF 标志置 1。然后计数器在 ETR 出现上升沿时计数。

ETR 信号的上升沿与实际计数器复位之间的延迟是由于 ETRP 输入的重新同步电路引起的。

图 118. 外部时钟模式 2 + 触发模式下的控制电路



14.3.20 定时器同步

TIM 定时器从内部链接在一起，以实现定时器同步或级联。有关详细信息，请参见*第 419 页的第 15.3.15 节：定时器同步*。

14.3.21 调试模式

当微控制器进入调试模式（Cortex™-M4F 内核停止）时，TIMx 计数器会根据 DBG 模块中的 DBG_TIMx_STOP 配置位选择继续正常工作或者停止工作。有关详细信息，请参见*第 33.16.2 节：对定时器、看门狗、bxCAN 和 I^2C 的调试支持*。

14.4 TIM1 和 TIM8 寄存器

有关寄存器说明中使用的缩写，请参见*第 47 页的第 1.1 节*。

外设寄存器必须按半字（16 位）或字（32 位）进行写访问。而读访问则可按字节（8 位）、半字（16 位）或字（32 位）进行。

14.4.1 TIM1 和 TIM8 控制寄存器 1 (TIMx_CR1)

TIM1&TIM8 control register 1

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
Reserved						CKD[1:0]		ARPE	CMS[1:0]		DIR	OPM	URS	UDIS	CEN
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 15:10 保留，必须保持复位值。

位 9:8 **CKD[1:0]**：时钟分频 (Clock division)

此位域指示定时器时钟 (CK_INT) 频率与死区发生器以及数字滤波器（ETR、TIx）所使用的死区及采样时钟 (t_{DTS}) 之间的分频比，

00：$t_{DTS} = t_{CK\_INT}$
01：$t_{DTS} = 2 * t_{CK\_INT}$
10：$t_{DTS} = 4 * t_{CK\_INT}$
11：保留，不要设置成此值

位 7 **ARPE**：自动重载预装载使能 (Auto-reload preload enable)

0：TIMx_ARR 寄存器不进行缓冲
1：TIMx_ARR 寄存器进行缓冲

位 6:5 **CMS[1:0]**：中心对齐模式选择 (Center-aligned mode selection)

00：边沿对齐模式。计数器根据方向位 (DIR) 递增计数或递减计数。
01：中心对齐模式 1。计数器交替进行递增计数和递减计数。仅当计数器递减计数时，配置为输出的通道（TIMx_CCMRx 寄存器中的 CxS=00）的输出比较中断标志才置 1。
10：中心对齐模式 2。计数器交替进行递增计数和递减计数。仅当计数器递增计数时，配置为输出的通道（TIMx_CCMRx 寄存器中的 CxS=00）的输出比较中断标志才置 1。
11：中心对齐模式 3。计数器交替进行递增计数和递减计数。当计数器递增计数或递减计数时，配置为输出的通道（TIMx_CCMRx 寄存器中的 CxS=00）的输出比较中断标志都会置 1。

注意：只要计数器处于使能状态 (CEN=1)，就不得从边沿对齐模式切换为中心对齐模式。

位 4 **DIR**：方向 (Direction)

0：计数器递增计数
1：计数器递减计数

注意：当定时器配置为中心对齐模式或编码器模式时，该位为只读状态。

位 3 **OPM**：单脉冲模式 (One pulse mode)

0：计数器在发生更新事件时不会停止计数
1：计数器在发生下一更新事件时停止计数（将 CEN 位清零）

位 2 **URS**：更新请求源 (Update request source)

此位由软件置 1 和清零，用以选择 UEV 事件源。

0：使能时，所有以下事件都会生成更新中断或 DMA 请求。此类事件包括：
- 计数器上溢/下溢
- 将 UG 位置 1
- 通过从模式控制器生成的更新事件

1：使能时，只有计数器上溢/下溢会生成更新中断或 DMA 请求。

位 1 **UDIS**：更新禁止 (Update disable)

此位由软件置 1 和清零，用以使能/禁止 UEV 事件生成。

0：使能 UEV。更新 (UEV) 事件可通过以下事件之一生成：

- 计数器上溢/下溢
- 将 UG 位置 1
- 通过从模式控制器生成的更新事件

然后更新影子寄存器的值。

1：禁止 UEV。不会生成更新事件，各影子寄存器的值（ARR、PSC 和 CCRx）保持不变。但如果将 UG 位置 1，或者从从模式控制器接收到硬件复位，则会重新初始化计数器和预分频器。

位 0 **CEN**：计数器使能 (Counter enable)

0：禁止计数器

1：使能计数器

注意：只有事先通过软件将 CEN 位置 1，才可以使用外部时钟、门控模式和编码器模式。而触发模式可通过硬件自动将 CEN 位置 1。

14.4.2 TIM1 和 TIM8 控制寄存器 2 (TIMx_CR2)

TIM1&TIM8 control register 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.	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	rw		rw

位 15 保留，必须保持复位值。

位 14 **OIS4**：输出空闲状态 4（OC4 输出）(Output Idle state 4 (OC4 output))

参见 OIS1 位

位 13 **OIS3N**：输出空闲状态 3（OC3N 输出）(Output Idle state 3 (OC3N output))

参见 OIS1N 位

位 12 **OIS3**：输出空闲状态 3（OC3 输出）(Output Idle state 3 (OC3 output))

参见 OIS1 位

位 11 **OIS2N**：输出空闲状态 2（OC2N 输出）(Output Idle state 2 (OC2N output))

参见 OIS1N 位

位 10 **OIS2**：输出空闲状态 2（OC2 输出）(Output Idle state 2 (OC2 output))

参见 OIS1 位

位 9 **OIS1N**：输出空闲状态 1（OC1N 输出）(Output Idle state 1 (OC1N output))

0：当 MOE=0 时，经过死区时间后 OC1N=0

1：当 MOE=0 时，经过死区时间后 OC1N=1

注意：只要编程了 LOCK（TIMx_BDTR 寄存器中的 LOCK 位）级别 1、2 或 3，此位即无法修改。

位 8 **OIS1**：输出空闲状态 1（OC1 输出）(Output Idle state 1 (OC1 output))

0：当 MOE=0 时，（如果 OC1N 有效，则经过死区时间之后）OC1=0
1：当 MOE=0 时，（如果 OC1N 有效，则经过死区时间之后）OC1=1

注意：只要编程了 LOCK（TIMx_BDTR 寄存器中的 LOCK 位）级别 1、2 或 3，此位即无法修改。

位 7 **TI1S**：TI1 选择 (TI1 selection)

0：TIMx_CH1 引脚连接到 TI1 输入
1：TIMx_CH1、CH2 和 CH3 引脚连接到 TI1 输入（异或组合）

位 6:4 **MMS[1:0]**：主模式选择 (Master mode selection)

这些位可选择主模式下将要发送到从定时器以实现同步的信息 (TRGO)。这些位的组合如下：
000：**复位**——TIMx_EGR 寄存器中的 UG 位用作触发输出 (TRGO)。如果复位由触发输入生成（从模式控制器配置为复位模式），则 TRGO 上的信号相比实际复位会有延迟。
001：**使能**——计数器使能信号 CNT_EN 用作触发输出 (TRGO)。该触发输出可用于同时启动多个定时器，或者控制在一段时间内使能从定时器。计数器使能信号可由CEN控制位产生。当配置为门控模式时，也可由触发输入产生。当计数器使能信号由触发输入控制时，TRGO 上会存在延迟，选择主/从模式时除外（请参见 TIMx_SMCR 寄存器中 MSM 位的说明）。
010：**更新**——选择更新事件作为触发输出 (TRGO)。例如，主定时器可用作从定时器的预分频器。
011：**比较脉冲**——一旦发生输入捕获或比较匹配事件，当 CC1IF 被置 1 时（即使已为高电平），触发输出都会发送一个正脉冲。(TRGO)。
100：**比较**——OC1REF 信号用作触发输出 (TRGO)
101：**比较**——OC2REF 信号用作触发输出 (TRGO)
110：**比较**——OC3REF 信号用作触发输出 (TRGO)
111：**比较**——OC4REF 信号用作触发输出 (TRGO)

位 3 **CCDS**：捕获/比较 DMA 选择 (Capture/compare DMA selection)

0：发生 CCx 事件时发送 CCx DMA 请求
1：发生更新事件时发送 CCx 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 (CCPC=1) 进行预装载，仅通过将 COMG 位置 1 来对这些位进行更新
1：如果捕获/比较控制位 (CCPC=1) 进行预装载，可通过将 COMG 位置 1 或 TRGI 的上升沿对这些位进行更新。

注意：此位仅对具有互补输出的通道有效。

位 1 保留，必须保持复位值。

位 0 **CCPC**：捕获/比较预装载控制 (Capture/compare preloaded control)

0：CCxE、CCxNE 和 OCxM 位未进行预装载
1：CCxE、CCxNE 和 OCxM 位进行了预装载，写入这些位后，仅当发生换向事件 (COM)（COMG 位置 1 或在 TRGI 上检测到上升沿，取决于 CCUS 位）时才会对这些位进行更新。

注意：此位仅对具有互补输出的通道有效。

14.4.3 TIM1 和 TIM8 从模式控制寄存器 (TIMx_SMCR)

TIM1&TIM8 slave mode control register

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	rw	rw	rw	rw	rw	Res.	rw	rw	rw

位 15 **ETP**：外部触发极性 (External trigger polarity)

此位可选择将 ETR 还是 $\overline{ETR}$ 用于触发操作

0：ETR 未反相，高电平或上升沿有效。

1：ETR 反相，低电平或下降沿有效。

位 14 **ECE**：外部时钟使能 (External clock enable)

此位可使能外部时钟模式 2。

0：禁止外部时钟模式 2

1：使能外部时钟模式 2。计数器时钟由 ETRF 信号的任意有效边沿提供。

*注意：**1**：将 ECE 位置 1 与选择外部时钟模式 1 并将 TRGI 连接到 ETRF（SMS=111 且 TS=111）具有相同效果。*

***2**：外部时钟模式 2 可以和以下从模式同时使用：复位模式、门控模式和触发模式。不过此类情况下 TRGI 不得连接 ETRF（TS 位不得为 111）。*

***3**：如果同时使能外部时钟模式 1 和外部时钟模式 2，则外部时钟输入为 ETRF。*

位 13:12 **ETPS[1:0]**：外部触发预分频器 (External trigger prescaler)

外部触发信号 ETRP 频率不得超过 TIMxCLK 频率的 1/4。可通过使能预分频器来降低 ETRP 频率。这种方法在输入快速外部时钟时非常有用。

00：预分频器关闭

01：2 分频 ETRP 频率

10：4 分频 ETRP 频率

11：8 分频 ETRP 频率

位 11:8 **ETF[3:0]**：外部触发滤波器 (External trigger filter)

此位域可定义 ETRP 信号的采样频率和适用于 ETRP 的数字滤波时间。数字滤波器由事件计数器组成，每 N 个事件才视为一个有效边沿：

0000：无滤波器，按 f_{DTS} 频率进行采样

0001：$f_{SAMPLING}=f_{CK\_INT}$，N=2

0010：$f_{SAMPLING}=f_{CK\_INT}$，N=4

0011：$f_{SAMPLING}=f_{CK\_INT}$，N=8

0100：$f_{SAMPLING}=f_{DTS}/2$，N=6

0101：$f_{SAMPLING}=f_{DTS}/2$，N=8

0110：$f_{SAMPLING}=f_{DTS}/4$，N=6

0111：$f_{SAMPLING}=f_{DTS}/4$，N=8

1000：$f_{SAMPLING}=f_{DTS}/8$，N=6

1001：$f_{SAMPLING}=f_{DTS}/8$，N=8

1010：$f_{SAMPLING}=f_{DTS}/16$，N=5

1011：$f_{SAMPLING}=f_{DTS}/16$，N=6

1100：$f_{SAMPLING}=f_{DTS}/16$，N=8

1101：$f_{SAMPLING}=f_{DTS}/32$，N=5

1110：$f_{SAMPLING}=f_{DTS}/32$，N=6

1111：$f_{SAMPLING}=f_{DTS}/32$，N=8

位 7 **MSM：**主/从模式 (Master/slave mode)

0：不执行任何操作

1：当前定时器的触发输入事件 (TRGI) 的动作被推迟，以使当前定时器与其从定时器实现完美同步（通过 TRGO）。此设置适用于由单个外部事件对多个定时器进行同步的情况。

位 6:4 **TS[2:0]：**触发选择 (Trigger selection)

此位域可选择将要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 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)

有关各定时器 ITRx 含义的详细信息，请参见*第 370 页的表 72：TIMx 内部触发连接*。

注意：这些位只能在未使用的情况下（例如，SMS=000 时）进行更改，以避免转换时出现错误的边沿检测。

位 3 保留，必须保持复位值。

位 2:0 **SMS：**从模式选择 (Slave mode selection)

选择外部信号时，触发信号 (TRGI) 的有效边沿与外部输入上所选的极性相关（请参见输入控制寄存器和控制寄存器说明）。

000：禁止从模式——如果 CEN = “1”，预分频器时钟直接由内部时钟提供。

001：编码器模式 1——计数器根据 TI1FP1 电平在 TI2FP2 边沿递增/递减计数。

010：编码器模式 2——计数器根据 TI2FP2 电平在 TI1FP1 边沿递增/递减计数。

011：编码器模式 3——计数器在 TI1FP1 和 TI2FP2 的边沿计数，计数的方向取决于另外一个信号的电平。

100：复位模式——在出现所选触发输入 (TRGI) 上升沿时，重新初始化计数器并生成一个寄存器更新事件。

101：门控模式——触发输入 (TRGI) 为高电平时使能计数器时钟。只要触发输入变为低电平，计数器立即停止计数（但不复位）。计数器的启动和停止都是受控的。

110：触发模式——触发信号 TRGI 出现上升沿时启动计数器（但不复位）。只控制计数器的启动。

111：外部时钟模式 1——由所选触发信号 (TRGI) 的上升沿提供计数器时钟。

注意：如果将 TI1F_ED 选作触发输入（TS= “100”），则不得使用门控模式。实际上，TI1F 每次转换时，TI1F_ED 都输出 1 个脉冲，而门控模式检查的则是触发信号的电平。

表 72. TIMx 内部触发连接

从 TIM	ITR0 (TS = 000)	ITR1 (TS = 001)	ITR2 (TS = 010)	ITR3 (TS = 011)
TIM1	TIM5	TIM2	TIM3	TIM4
TIM8	TIM1	TIM2	TIM4	TIM5

14.4.4 TIM1 和 TIM8 DMA/中断使能寄存器 (TIMx_DIER)

TIM1&TIM8 DMA/interrupt enable register

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 保留，必须保持复位值。

位 14 **TDE**：触发 DMA 请求使能 (Trigger DMA request enable)

0：禁止 DMA 请求
1：使能 DMA 请求

位 13 **COMDE**：COM DMA 请求使能 (COM DMA request enable)

0：禁止 COM DMA 请求
1：使能 COM DMA 请求

位 12 **CC4DE**：捕获/比较 4 DMA 请求使能 (Capture/Compare 4 DMA request enable)

0：禁止 CC4 DMA 请求
1：使能 CC4 DMA 请求

位 11 **CC3DE**：捕获/比较 3 DMA 请求使能 (Capture/Compare 3 DMA request enable)

0：禁止 CC3 DMA 请求
1：使能 CC3 DMA 请求

位 10 **CC2DE**：捕获/比较 2 DMA 请求使能 (Capture/Compare 2 DMA request enable)

0：禁止 CC2 DMA 请求
1：使能 CC2 DMA 请求

位 9 **CC1DE**：捕获/比较 1 DMA 请求使能 (Capture/Compare 1 DMA request enable)

0：禁止 CC1 DMA 请求
1：使能 CC1 DMA 请求

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 **BIE**：断路中断使能 (Break interrupt enable)

0：禁止断路中断
1：使能断路中断

位 6 **TIE**：触发信号 (TGRI) 中断使能 (Trigger interrupt enable)

0：禁止触发信号 (TGRI) 中断
1：使能触发信号 (TGRI) 中断

位 5 **COMIE**：COM 中断使能 (COM interrupt enable)

0：禁止 COM 中断
1：使能 COM 中断

位 4 **CC4IE**：捕获/比较 4 中断使能 (Capture/Compare 4 interrupt enable)

0：禁止 CC4 中断
1：使能 CC4 中断

位 3 **CC3IE**：捕获/比较 3 中断使能 (Capture/Compare 3 interrupt enable)

0：禁止 CC3 中断
1：使能 CC3 中断

位 2 **CC2IE**：捕获/比较 2 中断使能 (Capture/Compare 2 interrupt enable)

0：禁止 CC2 中断
1：使能 CC2 中断

位 1 **CC1IE**：捕获/比较 1 中断使能 (Capture/Compare 1 interrupt enable)

0：禁止 CC1 中断
1：使能 CC1 中断

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

14.4.5 TIM1 和 TIM8 状态寄存器 (TIMx_SR)

TIM1&TIM8 status register

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
Reserved			CC4OF	CC3OF	CC2OF	CC1OF	Res.	BIF	TIF	COMIF	CC4IF	CC3IF	CC2IF	CC1IF	UIF
			rc_w0	rc_w0	rc_w0	rc_w0	Res.	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0

位 15:13 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)

请参见 CC1OF 说明

位 11 **CC3OF**：捕获/比较 3 重复捕获标志 (Capture/Compare 3 overcapture flag)

请参见 CC1OF 说明

位 10 **CC2OF**：捕获/比较 2 重复捕获标志 (Capture/Compare 2 overcapture flag)

请参见 CC1OF 说明

位 9 **CC1OF**：捕获/比较 1 重复捕获标志 (Capture/Compare 1 overcapture flag)

仅当将相应通道配置为输入捕获模式时，此标志位才会由硬件置 1。通过软件写入“0”可将该位清零。
0：未检测到重复捕获。
1：TIMx_CCR1 寄存器中已捕获到计数器值且 CC1IF 标志已置 1。

位 8 保留，必须保持复位值。

位 7 **BIF**：断路中断标志 (Break interrupt flag)

只要断路输入变为有效状态，此标志便由硬件置 1。断路输入无效后可通过软件对其清零。
0：未发生断路事件。
1：在断路输入上检测到有效电平。

位 6 **TIF**：触发中断标志 (Trigger interrupt flag)

在除门控模式以外的所有模式下，当使能从模式控制器后在 TRGI 输入上检测到有效边沿时，该标志将由硬件置 1。选择门控模式时，该标志将在计数器启动或停止时置 1。但需要通过软件清零。

0：未发生触发事件。

1：触发中断挂起。

位 5 **COMIF**：COM 中断标志 (COM interrupt flag)

此标志在发生 COM 事件时（捕获/比较控制位 CCxE、CCxNE 和 OCxM 已更新时）由硬件置 1。但需要通过软件清零。

0：未发生 COM 事件。

1：COM 中断挂起。

位 4 **CC4IF**：捕获/比较 4 中断标志 (Capture/Compare 4 interrupt flag)

请参见 CC1IF 说明

位 3 **CC3IF**：捕获/比较 3 中断标志 (Capture/Compare 3 interrupt flag)

请参见 CC1IF 说明

位 2 **CC2IF**：捕获/比较 2 中断标志 (Capture/Compare 2 interrupt flag)

请参见 CC1IF 说明

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：

当计数器与比较值匹配时，此标志由硬件置 1，中心对齐模式下除外（请参见 TIMx_CR1 寄存器中的 CMS 位说明）。但需要通过软件清零。

0：不匹配。

1：TIMx_CNT 计数器的值与 TIMx_CCR1 寄存器的值匹配。当 TIMx_CCR1 的值大于 TIMx_ARR 的值时，CC1IF 位将在计数器发生上溢（递增计数模式和增减计数模式下）或下溢（递减计数模式下）时变为高电平。

如果通道 CC1 配置为输入：

此位将在发生捕获事件时由硬件置 1。通过软件或读取 TIMx_CCR1 寄存器将该位清零。

0：未发生输入捕获事件

1：TIMx_CCR1 寄存器中已捕获到计数器值（IC1 上已检测到与所选极性匹配的边沿）

位 0 **UIF**：更新中断标志 (Update interrupt flag)

该位在发生更新事件时通过硬件置 1。但需要通过软件清零。

0：未发生更新。

1：更新中断挂起。该位在以下情况下更新寄存器时由硬件置 1：

– TIMx_CR1 寄存器中的 UDIS=0，并且重复计数器值上溢或下溢时（重复计数器 = 0 时更新）。

– TIMx_CR1 寄存器中的 URS = 0 且 UDIS = 0，并且由软件使用 TIMx_EGR 寄存器中的 UG 位重新初始化 CNT 时。

– TIMx_CR1 寄存器中的 URS=0 且 UDIS=0，并且 CNT 由触发事件重新初始化时（请参见 *第 14.4.3 节：TIM1 和 TIM8 从模式控制寄存器 (TIMx_SMCR)*）。

14.4.6 TIM1 和 TIM8 事件生成寄存器 (TIMx_EGR)

TIM1&TIM8 event generation register

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
Reserved								BG	TG	COMG	CC4G	CC3G	CC2G	CC1G	UG
								w	w	w	w	w	w	w	w

位 15:8 保留，必须保持复位值。

位 7 **BG**：断路生成 (Break generation)

此位由软件置 1 以生成事件，并由硬件自动清零。
0：不执行任何操作
1：生成断路事件。MOE 位清零且 BIF 标志置 1。使能后可发生相关中断或 DMA 传输事件。

位 6 **TG**：生成触发信号 (Trigger generation)

此位由软件置 1 以生成事件，并由硬件自动清零。
0：不执行任何操作
1：TIMx_SR 寄存器中的 TIF 标志置 1。使能后可发生相关中断或 DMA 传输事件。

位 5 **COMG**：捕获/比较控制更新生成 (Capture/Compare control update generation)

该位可通过软件置 1，并由硬件自动清零
0：不执行任何操作
1：CCPC 位置 1 时，可更新 CCxE、CCxNE 和 OCxM 位
注意：此位仅对具有互补输出的通道有效。

位 4 **CC4G**：捕获/比较 4 生成 (Capture/Compare 4 generation)

请参见 CC1G 说明

位 3 **CC3G**：捕获/比较 3 生成 (Capture/Compare 3 generation)

请参见 CC1G 说明

位 2 **CC2G**：捕获/比较 2 生成 (Capture/Compare 2 generation)

请参见 CC1G 说明

位 1 **CC1G**：捕获/比较 1 生成 (Capture/Compare 1 generation)

此位由软件置 1 以生成事件，并由硬件自动清零。
0：不执行任何操作
1：通道 1 上生成捕获/比较事件：
如果通道 CC1 配置为输出：
使能时，CC1IF 标志置 1 并发送相应的中断或 DMA 请求。
如果通道 CC1 配置为输入：
TIMx_CCR1 寄存器中将捕获到计数器当前值。使能时，CC1IF 标志置 1 并发送相应的中断或 DMA 请求。如果 CC1IF 标志已为高电平，CC1OF 标志将置 1。

位 0 **UG**：更新生成 (Update generation)

该位可通过软件置 1，并由硬件自动清零。
0：不执行任何操作
1：重新初始化计数器并生成一个寄存器更新事件。请注意，预分频器计数器也将清零（但预分频比不受影响）。如果选择中心对齐模式或 DIR=0（递增计数），计数器将清零；如果 DIR=1（递减计数），计数器将使用自动重载值 (TIMx_ARR)。

14.4.7 TIM1 和 TIM8 捕获/比较模式寄存器 1 (TIMx_CCMR1)

TIM1&TIM8 capture/compare mode register 1

偏移地址：0x18

复位值：0x0000

这些通道可用于输入（捕获模式）或输出（比较模式）模式。通道方向通过配置相应的 CCxS 位进行定义。此寄存器的所有其它位在输入模式和输出模式下的功能均不同。对于任一给定位，OCxx 用于说明通道配置为输出时该位对应的功能，ICxx 则用于说明通道配置为输入时该位对应的功能。因此，必须注意同一个位在输入阶段和输出阶段具有不同的含义。

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC2 CE	OC2M[2:0]			OC2 PE	OC2 FE	CC2S[1:0]		OC1 CE	OC1M[2:0]			OC1 PE	OC1 FE	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	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)

此位域定义通道方向（输入/输出）以及所使用的输入。

00：CC2 通道配置为输出

01：CC2 通道配置为输入，IC2 映射到 TI2 上

10：CC2 通道配置为输入，IC2 映射到 TI1 上

11：CC2 通道配置为输入，IC2 映射到 TRC 上。此模式仅在通过 TS 位（TIMx_SMCR 寄存器）选择内部触发输入时有效

注意：仅当通道关闭时（TIMx_CCER 中的 CC2E = “0”），才可向 CC2S 位写入数据。

位 7 **OC1CE**：输出比较 1 清零使能 (Output Compare 1 clear enable)

OC1CE：输出比较 1 清零使能 (Output Compare 1 Clear Enable)

0：OC1Ref 不受 ETRF 输入影响

1：ETRF 输入上检测到高电平时，OC1Ref 立即清零。

位 6:4 **OC1M**：输出比较 1 模式 (Output Compare 1 mode)

这些位定义提供 OC1 和 OC1N 的输出参考信号 OC1REF 的行为。OC1REF 为高电平有效，而 OC1 和 OC1N 的有效电平则取决于 CC1P 位和 CC1NP 位。

000：冻结——输出比较寄存器 TIMx_CCR1 与计数器 TIMx_CNT 进行比较不会对输出造成任何影响。（该模式用于生成时基）。

001：将通道 1 设置为匹配时输出有效电平。当计数器 TIMx_CNT 与捕获/比较寄存器 1 (TIMx_CCR1) 匹配时，OC1REF 信号强制变为高电平。

010：将通道 1 设置为匹配时输出无效电平。当计数器 TIMx_CNT 与捕获/比较寄存器 1 (TIMx_CCR1) 匹配时，OC1REF 信号强制变为低电平。

011：翻转——TIMx_CNT=TIMx_CCR1 时，OC1REF 发生翻转。

100：强制变为无效电平——OC1REF 强制变为低电平。

101：强制变为有效电平——OC1REF 强制变为高电平。

110：PWM 模式 1——在递增计数模式下，只要 TIMx_CNT<TIMx_CCR1，通道 1 便为有效状态，否则为无效状态。在递减计数模式下，只要 TIMx_CNT>TIMx_CCR1，通道 1 便为无效状态（OC1REF=“0”），否则为有效状态（OC1REF=“1”）。

111：PWM 模式 2——在递增计数模式下，只要 TIMx_CNT<TIMx_CCR1，通道 1 便为无效状态，否则为有效状态。在递减计数模式下，只要 TIMx_CNT>TIMx_CCR1，通道 1 便为有效状态，否则为无效状态。

*注意：**1：**只要编程了 LOCK（TIMx_BDTR 寄存器中的 LOCK 位）级别 3 且 CC1S=“00”（通道配置为输出），这些位即无法修改。*

***2：**在 PWM 模式 1 或 PWM 模式 2 下，仅当比较结果发生改变或输出比较模式由“冻结”模式切换到“PWM”模式时，OCREF 电平才会发生更改。*

***3：**此位域将在具有互补输出的通道上进行预装载。如果 TIMx_CR2 寄存器中的 CCPC 位置 1，则仅当生成 COM 事件时，OC1M 有效位才会从预装载位获取新值。*

位 3 **OC1PE**：输出比较 1 预装载使能 (Output Compare 1 preload enable)

0:禁止与 TIMx_CCR1 相关的预装载寄存器。可随时向 TIMx_CCR1 写入数据，写入后将立即使用新值。

1：使能与 TIMx_CCR1 相关的预装载寄存器。可读/写访问预装载寄存器。TIMx_CCR1 预装载值在每次生成更新事件时都会装载到活动寄存器中。

*注意：**1：**只要编程了 LOCK（TIMx_BDTR 寄存器中的 LOCK 位）级别 3 且 CC1S=“00”（通道配置为输出），这些位即无法修改。*

***2：**只有单脉冲模式下才可在未验证预装载寄存器的情况下使用 PWM 模式（TIMx_CR1 寄存器中的 OPM 位置 1）。其它情况下则无法保证该行为。*

位 2 **OC1FE**：输出比较 1 快速使能 (Output Compare 1 fast enable)

此位用于加快触发输入事件对 CC 输出的影响。

0：即使触发开启，CC1 也将根据计数器和 CCR1 值正常工作。触发输入出现边沿时，激活 CC1 输出的最短延迟时间为 5 个时钟周期。

1：触发输入上出现有效边沿相当于 CC1 输出上的比较匹配。随后，无论比较结果如何，OC 都设置为比较电平。采样触发输入和激活 CC1 输出的延迟时间缩短为 3 个时钟周期。仅当通道配置为 PWM1 或 PWM2 模式时，OCFE 才会起作用。

位 1:0 **CC1S**：捕获/比较 1 选择 (Capture/Compare 1 selection)

此位域定义通道方向（输入/输出）以及所使用的输入。

00：CC1 通道配置为输出

01：CC1 通道配置为输入，IC1 映射到 TI1 上

10：CC1 通道配置为输入，IC1 映射到 TI2 上

11：CC1 通道配置为输入，IC1 映射到 TRC 上。此模式仅在通过 TS 位（TIMx_SMCR 寄存器）选择内部触发输入时有效

注意：仅当通道关闭时（TIMx_CCER 中的 CC1E = “0”），才可向 CC1S 位写入数据。

输入捕获模式

位 15:12 **IC2F**：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**：捕获/比较 2 选择 (Capture/Compare 2 selection)

此位域定义通道方向（输入/输出）以及所使用的输入。
00：CC2 通道配置为输出
01：CC2 通道配置为输入，IC2 映射到 TI2 上
10：CC2 通道配置为输入，IC2 映射到 TI1 上
11：CC2 通道配置为输入，IC2 映射到 TRC 上。此模式仅在通过 TS 位（TIMx_SMCR 寄存器）选择内部触发输入时有效
注意：仅当通道关闭时（TIMx_CCER 中的 CC2E = “0”），才可向 CC2S 位写入数据。

位 7:4 **IC1F[3:0]**：输入捕获 1 滤波器 (Input capture 1 filter)

此位域可定义 TI1 输入的采样频率和适用于 TI1 的数字滤波器带宽。数字滤波器由事件计数器组成，每 N 个事件才视为一个有效边沿：
0000：无滤波器，按 f_{DTS} 频率进行采样
0001：$f_{SAMPLING}=f_{CK\_INT}$，N=2
0010：$f_{SAMPLING}=f_{CK\_INT}$，N=4
0011：$f_{SAMPLING}=f_{CK\_INT}$，N=8
0100：$f_{SAMPLING}=f_{DTS}/2$，N=6
0101：$f_{SAMPLING}=f_{DTS}/2$，N=8
0110：$f_{SAMPLING}=f_{DTS}/4$，N=6
0111：$f_{SAMPLING}=f_{DTS}/4$，N=8
1000：$f_{SAMPLING}=f_{DTS}/8$，N=6
1001：$f_{SAMPLING}=f_{DTS}/8$，N=8
1010：$f_{SAMPLING}=f_{DTS}/16$，N=5
1011：$f_{SAMPLING}=f_{DTS}/16$，N=6
1100：$f_{SAMPLING}=f_{DTS}/16$，N=8
1101：$f_{SAMPLING}=f_{DTS}/32$，N=5
1110：$f_{SAMPLING}=f_{DTS}/32$，N=6
1111：$f_{SAMPLING}=f_{DTS}/32$，N=8

位 3:2 **IC1PSC**：输入捕获 1 预分频器 (Input capture 1 prescaler)

此位域定义 CC1 输入 (IC1) 的预分频比。
只要 CC1E=“0”（TIMx_CCER 寄存器），预分频器便立即复位。
00：无预分频器，捕获输入上每检测到一个边沿便执行捕获
01：每发生 2 个事件便执行一次捕获
10：每发生 4 个事件便执行一次捕获
11：每发生 8 个事件便执行一次捕获

位 1:0 **CC1S**：捕获/比较 1 选择 (Capture/Compare 1 Selection)

此位域定义通道方向（输入/输出）以及所使用的输入。
00：CC1 通道配置为输出
01：CC1 通道配置为输入，IC1 映射到 TI1 上
10：CC1 通道配置为输入，IC1 映射到 TI2 上
11：CC1 通道配置为输入，IC1 映射到 TRC 上。此模式仅在通过 TS 位（TIMx_SMCR 寄存器）选择内部触发输入时有效
注意：仅当通道关闭时（TIMx_CCER 中的 CC1E = “0”），才可向 CC1S 位写入数据。

14.4.8 TIM1 和 TIM8 捕获/比较模式寄存器 2 (TIMx_CCMR2)

TIM1&TIM8 capture/compare mode register 2

偏移地址：0x1C

复位值：0x0000

请参见上述 CCMR1 寄存器说明。

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC4 CE	OC4M[2:0]			OC4 PE	OC4 FE	CC4S[1:0]		OC3 CE.	OC3M[2:0]			OC3 PE	OC3 FE	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	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**：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**：捕获/比较 4 选择 (Capture/Compare 4 selection)

此位域定义通道方向（输入/输出）以及所使用的输入。
00：CC4 通道配置为输出
01：CC4 通道配置为输入，IC4 映射到 TI4 上
10：CC4 通道配置为输入，IC4 映射到 TI3 上
11：CC4 通道配置为输入，IC4 映射到 TRC 上。此模式仅在通过 TS 位（TIMx_SMCR 寄存器）选择内部触发输入时有效

注意：仅当通道关闭时（TIMx_CCER 中的 CC4E = “0”），才可向 CC4S 位写入数据。

位 7 **OC3CE**：输出比较 3 清零使能 (Output compare 3 clear enable)

位 6:4 **OC3M**：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**：捕获/比较 3 选择 (Capture/Compare 3 selection)

此位域定义通道方向（输入/输出）以及所使用的输入。
00：CC3 通道配置为输出
01：CC3 通道配置为输入，IC3 映射到 TI3 上
10：CC3 通道配置为输入，IC3 映射到 TI4 上
11：CC3 通道配置为输入，IC3 映射到 TRC 上。此模式仅在通过 TS 位（TIMx_SMCR 寄存器）选择内部触发输入时有效

注意：仅当通道关闭时（TIMx_CCER 中的 CC3E = “0”），才可向 CC3S 位写入数据。

输入捕获模式

位 15:12 **IC4F**：输入捕获 4 滤波器 (Input capture 4 filter)

位 11:10 **IC4PSC**：输入捕获 4 预分频器 (Input capture 4 prescaler)

位 9:8 **CC4S**：捕获/比较 4 选择 (Capture/Compare 4 selection)

此位域定义通道方向（输入/输出）以及所使用的输入。
00：CC4 通道配置为输出
01：CC4 通道配置为输入，IC4 映射到 TI4 上
10：CC4 通道配置为输入，IC4 映射到 TI3 上
11：CC4 通道配置为输入，IC4 映射到 TRC 上。此模式仅在通过 TS 位（TIMx_SMCR 寄存器）选择内部触发输入时有效

注意：仅当通道关闭时（TIMx_CCER 中的 CC4E = “0”），才可向 CC4S 位写入数据。

位 7:4 **IC3F**：输入捕获 3 滤波器 (Input capture 3 filter)

位 3:2 **IC3PSC**：输入捕获 3 预分频器 (Input capture 3 prescaler)

位 1:0 **CC3S**：捕获/比较 3 选择 (Capture/compare 3 selection)

此位域定义通道方向（输入/输出）以及所使用的输入。
00：CC3 通道配置为输出
01：CC3 通道配置为输入，IC3 映射到 TI3 上
10：CC3 通道配置为输入，IC3 映射到 TI4 上
11：CC3 通道配置为输入，IC3 映射到 TRC 上。此模式仅在通过 TS 位（TIMx_SMCR 寄存器）选择内部触发输入时有效

注意：仅当通道关闭时（TIMx_CCER 中的 CC3E = “0”），才可向 CC3S 位写入数据。

14.4.9 TIM1 和 TIM8 捕获/比较使能寄存器 (TIMx_CCER)

TIM1&TIM8 capture/compare enable register

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
Reserved		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

位 15:14 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)

请参见 CC1P 说明

位 12 **CC4E**：捕获/比较 4 输出使能 (Capture/Compare 4 output enable)

请参见 CC1E 说明

位 11 **CC3NP**：捕获/比较 3 互补输出极性 (Capture/Compare 3 complementary output polarity)

请参见 CC1NP 说明

位 10 **CC3NE**：捕获/比较 3 互补输出使能 (Capture/Compare 3 complementary output enable)

请参见 CC1NE 说明

位 9 **CC3P**：捕获/比较 3 输出极性 (Capture/Compare 3 output polarity)

请参见 CC1P 说明

位 8 **CC3E**：捕获/比较 3 输出使能 (Capture/Compare 3 output enable)

请参见 CC1E 说明

位 7 **CC2NP**：捕获/比较 2 互补输出极性 (Capture/Compare 2 complementary output polarity)

请参见 CC1NP 说明

位 6 **CC2NE**：捕获/比较 2 互补输出使能 (Capture/Compare 2 complementary output enable)

请参见 CC1NE 说明

位 5 **CC2P**：捕获/比较 2 输出极性 (Capture/Compare 2 output polarity)

请参见 CC1P 说明

位 4 **CC2E**：捕获/比较 2 输出使能 (Capture/Compare 2 output enable)

请参见 CC1E 说明

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：

0：OC1N 高电平有效。

1：OC1N 低电平有效。

CC1 通道配置为输入：

此位与 CC1P 配合使用，用以定义 TI1FP1 和 TI2FP1 的极性。请参见 CC1P 说明。

注意：此位将在具有互补输出的通道上进行预装载。如果 TIMx_CR2 寄存器中的 CCPC 位置 1，则仅当生成换向事件时，CC1NP 有效位才会从预装载位获取新值。

注意：只要编程了 LOCK（TIMx_BDTR 寄存器中的 LOCK 位）级别 2 或 3 且 CC1S=“00”（通道配置为输出），此位立即变为不可写状态。

位 2 **CC1NE**：捕获 / 比较 1 互补输出使能 (Capture/Compare 1 complementary output enable)

0：关闭 ——OC1N 未激活。OC1N 电平是 MOE、OSSI、OSSR、OIS1、OIS1N 和 CC1E 位的函数。

1：开启 —— 在相应输出引脚上输出 OC1N 信号，具体取决于 MOE、OSSI、OSSR、OIS1、OIS1N 和 CC1E 位。

注意：此位将在具有互补输出的通道上进行预装载。如果 TIMx_CR2 寄存器中的 CCPC 位置 1，则仅当生成换向事件时， CC1NE 有效位才会从预装载位获取新值。

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：

0：OC1 高电平有效

1：OC1 低电平有效

CC1 通道配置为输入：

CC1NP/CC1P 位可针对触发或捕获操作选择 TI1FP1 和 TI2FP1 的有效极性。

00：非反相/上升沿触发

电路对 TIxFP1 上升沿敏感（在复位模式、外部时钟模式或触发模式下执行捕获或触发操作），TIxFP1 未反相（在门控模式或编码器模式下执行触发操作）。

01：反相/下降沿触发

电路对 TIxFP1 下升沿敏感（在复位模式、外部时钟模式或触发模式下执行捕获或触发操作），TIxFP1 反相（在门控模式或编码器模式下执行触发操作）。

10：保留，不使用此配置。

11：未反相/边沿触发。

电路对 TIxFP1 上升沿和下降沿都敏感（在复位模式、外部时钟模式或触发模式下执行捕获或触发操作），TIxFP1 未反相（在门控模式下执行触发操作）。编码器模式下不得使用此配置。

注意：此位将在具有互补输出的通道上进行预装载。如果 TIMx_CR2 寄存器中的 CCPC 位置 1，则仅当生成换向事件时，CC1P 有效位才会从预装载位获取新值。

注意：只要编程了 LOCK（TIMx_BDTR 寄存器中的 LOCK 位）级别 2 或 3，此位立即变为不可写状态。

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 未激活。OC1 电平是 MOE、OSSI、OSSR、OIS1、OIS1N 和 CC1NE 位的函数。

1：开启 ——OC1 信号是相应输出引脚上的输出，具体取决于 MOE、OSSI、OSSR、OIS1、OIS1N 和 CC1NE 位。

CC1 通道配置为输入：

此位决定了是否可以实际将计数器值捕获到输入捕获 / 比较寄存器 1 (TIMx_CCR1) 中。

0：禁止捕获。

1：使能捕获。

注意：此位将在具有互补输出的通道上进行预装载。如果 TIMx_CR2 寄存器中的 CCPC 位置 1，则仅当生成换向事件时， CC1E 有效位才会从预装载位获取新值。

表 73. 具有断路功能的互补通道 OCx 和 OCxN 的输出控制位

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	禁止输出（不由定时器驱动） OCx=0、OCx_EN=0	禁止输出（不由定时器驱动） OCxN=0、OCxN_EN=0
		0	0	1	禁止输出（不由定时器驱动） OCx=0、OCx_EN=0	OCxREF + 极性 OCxN=OCxREF 异或 CCxNP、OCxN_EN=1
		0	1	0	OCxREF + 极性 OCx=OCxREF 异或 CCxP、OCx_EN=1	禁止输出（不由定时器驱动） OCxN=0、OCxN_EN=0
		0	1	1	OCREF + 极性 + 死区 OCx_EN=1	OCREF 互补项（而非 OCREF）+ 极性 + 死区 OCxN_EN=1
		1	0	0	禁止输出（不由定时器驱动） OCx=CCxP、OCx_EN=0	禁止输出（不由定时器驱动） OCxN=CCxNP、OCxN_EN=0
		1	0	1	关闭状态（输出使能为无效状态） OCx=CCxP、OCx_EN=1	OCxREF + 极性 OCxN=OCxREF 异或 CCxNP、OCxN_EN=1
		1	1	0	OCxREF + 极性 OCx=OCxREF 异或 CCxP、OCx_EN=1	关闭状态（输出使能为无效状态） OCxN=CCxNP、OCxN_EN=1
		1	1	1	OCREF + 极性 + 死区 OCx_EN=1	OCREF 互补项（而非 OCREF）+ 极性 + 死区 OCxN_EN=1
0	0	X	0	0	禁止输出（不由定时器驱动） OCx=CCxP、OCx_EN=0	禁止输出（不由定时器驱动） OCxN=CCxNP、OCxN_EN=0
	0		0	1	禁止输出 （不由定时器驱动） 异步：OCx=CCxP、 OCx_EN=0、 OCxN=CCxNP、 OCxN_EN=0 如果存在时钟：在死区后 OCx=OISx 且 OCxN=OISxN，从而假定 OISx 和 OISxN 在有效状态下与 OCX 和 OCxN 不对应。	
	0		1	0		
	0		1	1		
	1		0	0	禁止输出（不由定时器驱动） OCx=CCxP、OCx_EN=0	禁止输出（不由定时器驱动） OCxN=CCxNP、OCxN_EN=0
	1		0	1	关闭状态 （输出使能为无效状态） 异步：OCx=CCxP、 OCx_EN=1、 OCxN=CCxNP、 OCxN_EN=1 如果存在时钟：在死区后 OCx=OISx 且 OCxN=OISxN，从而假定 OISx 和 OISxN 在有效状态下与 OCX 和 OCxN 不对应	
	1		1	0		
	1		1	1		

1. 如果一个通道的两个输出均未使用 (CCxE = CCxNE = 0)，则 OISx、OISxN、CCxP 和 CCxNP 位必须保持清零状态。

注意： 与互补通道 OCx 和 OCxN 相连的外部 I/O 引脚的状态取决于通道 OCx 和 OCxN 的状态以及 GPIO 寄存器。

14.4.10 TIM1 和 TIM8 计数器 (TIMx_CNT)

TIM1&TIM8 counter

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	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 15:0 **CNT[15:0]**：计数器值 (Counter value)

14.4.11 TIM1 和 TIM8 预分频器 (TIMx_PSC)

TIM1&TIM8 prescaler

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	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 15:0 **PSC[15:0]**：预分频器值 (Prescaler value)

计数器时钟频率 (CK_CNT) 等于 f_{CK_PSC} / (PSC[15:0] + 1)。
PSC 包含每次发生更新事件（包括计数器通过 TIMx_EGR 寄存器中的 UG 位清零时，或在配置为“复位模式”时通过触发控制器清零时）时要装载到活动预分频器寄存器的值。

14.4.12 TIM1 和 TIM8 自动重载寄存器 (TIMx_ARR)

TIM1&TIM8 auto-reload register

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	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 15:0 **ARR[15:0]**：自动重载值 (Auto-reload value)

ARR 为要装载到实际自动重载寄存器的值。
有关 ARR 更新和行为的详细信息，请参见*第 331 页的第 14.3.1 节：时基单元*。
当自动重载值为空时，计数器不工作。

14.4.13 TIM1 和 TIM8 重复计数器寄存器 (TIMx_RCR)

TIM1&TIM8 repetition counter register

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
Reserved								REP[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

位 15:8 保留，必须保持复位值。

位 7:0 **REP[7:0]**：重复计数器值 (Repetition counter value)

使能预装载寄存器时，用户可通过这些位设置比较寄存器的更新频率（即，从预装载寄存器向活动寄存器周期性传输数据）；使能更新中断时，也可设置更新中断的生成速率。

与 REP_CNT 相关的减计数器每次计数到 0 时，都将生成一个更新事件并且计数器从 REP 值重新开始计数。由于只有生成重复更新事件 U_RC 时，REP_CNT 才会重载 REP 值，因此在生成下一重复更新事件之前，无论向 TIMx_RCR 寄存器写入何值都无影响。

这意味着 PWM 模式下 (REP+1) 相当于：

- 边沿对齐模式下的 PWM 周期数
- 中心对齐模式下的 PWM 半周期数

14.4.14 TIM1 和 TIM8 捕获/比较寄存器 1 (TIMx_CCR1)

TIM1&TIM8 capture/compare register 1

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	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 15:0 **CCR1[15:0]**：捕获/比较 1 值 (Capture/Compare 1 value)

如果通道 CC1 配置为输出：

CCR1 是捕获/比较寄存器 1 的预装载值。

如果没有通过 TIMx_CCMR 寄存器中的 OC1PE 位来使能预装载功能，写入的数值会被直接传输至当前寄存器中。否则只在发生更新事件时生效（拷贝到实际起作用的捕获/比较寄存器1）。

实际捕获/比较寄存器中包含要与计数器 TIMx_CNT 进行比较并在 OC1 输出上发出信号的值。

如果通道 CC1 配置为输入：

CCR1 为上一个输入捕获 1 事件 (IC1) 发生时的计数器值。

14.4.15 TIM1 和 TIM8 捕获/比较寄存器 2 (TIMx_CCR2)

TIM1&TIM8 capture/compare register 2

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	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 15:0 **CCR2[15:0]**：捕获/比较 2 值 (Capture/Compare 2 value)

如果通道 CC2 配置为输出：
CCR2 为要装载到实际捕获/比较 2 寄存器的值（预装载值）。
如果没有通过 TIMx_CCMR2 寄存器中的 OC2PE 位来使能预装载功能，写入的数值会被直接传输至当前寄存器中。否则只有发生更新事件时，预装载值才会复制到活动捕获/比较 2 寄存器中。
实际捕获/比较寄存器中包含要与计数器 TIMx_CNT 进行比较并在 OC2 输出上发出信号的值。
如果通道 CC2 配置为输入：
CCR2 为上一个输入捕获 2 事件 (IC2) 发生时的计数器值。

14.4.16 TIM1 和 TIM8 捕获/比较寄存器 3 (TIMx_CCR3)

TIM1&TIM8 capture/compare register 3

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	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 15:0 **CCR3[15:0]**：捕获/比较值 (Capture/Compare value)

如果通道 CC3 配置为输出：
CCR3 为要装载到实际捕获/比较 3 寄存器的值（预装载值）。
如果没有通过 TIMx_CCMR3 寄存器中的 OC3PE 位来使能预装载功能，写入的数值会被直接传输至当前寄存器中。否则只有发生更新事件时，预装载值才会复制到活动捕获/比较 3 寄存器中。
实际捕获/比较寄存器中包含要与计数器 TIMx_CNT 进行比较并在 OC3 输出上发出信号的值。

如果通道 CC3 配置为输入：
CCR3 为上一个输入捕获 3 事件 (IC3) 发生时的计数器值。

14.4.17 TIM1 和 TIM8 捕获/比较寄存器 4 (TIMx_CCR4)

TIM1&TIM8 capture/compare register 4

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	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 15:0 **CCR4[15:0]**：捕获/比较值 (Capture/Compare value)

如果通道 CC4 配置为输出：
CCR4 为要装载到实际捕获/比较 4 寄存器的值（预装载值）。
如果没有通过 TIMx_CCMR4 寄存器中的 OC4PE 位来使能预装载功能，写入的数值会被直接传输至当前寄存器中。否则只有发生更新事件时，预装载值才会复制到活动捕获/比较 4 寄存器中。
实际捕获/比较寄存器中包含要与计数器 TIMx_CNT 进行比较并在 OC4 输出上发出信号的值。
如果通道 CC4 配置为输入：
CCR4 为上一个输入捕获 4 事件 (IC4) 发生时的计数器值。

14.4.18 TIM1 和 TIM8 断路和死区寄存器 (TIMx_BDTR)

TIM1&TIM8 break and dead-time register

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
MOE	AOE	BKP	BKE	OSSR	OSSI	LOCK[1:0]		DTG[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

注意： *由于可以根据 LOCK 配置锁定位 AOE、BKP、BKE、OSSI、OSSR 和 DTG[7:0] 的写操作，因此必须在第一次对 TIMx_BDTR 寄存器执行写访问时对这些位进行配置。*

位 15 **MOE**：主输出使能 (Main output enable)

只要断路输入变为有效状态，此位便由硬件异步清零。此位由软件置 1，也可根据 AOE 位状态自动置 1。此位仅对配置为输出的通道有效。
0：OC 和 OCN 输出禁止或被强制为空闲状态。
1：如果 OC 和 OCN 输出的相应使能位（TIMx_CCER 寄存器中的 CCxE 和 CCxNE 位）均置 1，则使能 OC 和 OCN 输出。
有关详细信息，请参见 OC/OCN 使能说明（*第 379 页的第 14.4.9 节：TIM1 和 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 只能由软件置 1
1：MOE 可由软件置 1，也可在发生下一更新事件时自动置 1(如果断路输入无效）
注意：只要编程了 LOCK（TIMx_BDTR 寄存器中的 LOCK 位）级别 1，此位即无法修改。

位 13 **BKP**：断路极性 (Break polarity)

0：断路输入 BRK 为低电平有效

1：断路输入 BRK 为高电平有效

注意：只要编程了 LOCK（TIMx_BDTR 寄存器中的 LOCK 位）级别 1，此位即无法修改。

注意：对该位执行任何写操作后，都需要经过 1 个 APB 时钟周期的延迟才生效。

位 12 **BKE**：断路使能 (Break enable)

0：禁止断路输入（BRK 和 CSS 时钟故障事件）

1：使能断路输入（BRK 和 CSS 时钟故障事件）

注意：编程了 LOCK（TIMx_BDTR 寄存器中的 LOCK 位）级别 1 后，此位即无法修改。

注意：对该位执行任何写操作后，都需要经过 1 个 APB 时钟周期的延迟才生效。

位 11 **OSSR**：运行模式下的关闭状态选择 (Off-state selection for Run mode)

此位在 MOE=1 时作用于配置为输出模式且具有互补输出的通道。如果定时器中没有互补输出，则不存在 OSSR。

有关详细信息，请参见 OC/OCN 使能说明（*第 379 页的第 14.4.9 节：TIM1 和 TIM8 捕获/比较使能寄存器 (TIMx_CCER)*）。

0：处于无效状态时，禁止 OC/OCN 输出（OC/OCN 使能输出信号=0）。

1：处于无效状态时，一旦 CCxE=1 或 CCxNE=1，便使能 OC/OCN 输出并将其设为无效电平。然后设置 OC/OCN 使能输出信号=1

注意：编程了 LOCK（TIMx_BDTR 寄存器中的 LOCK 位）级别 2 后，此位即无法修改。

位 10 **OSSI**：空闲模式下的关闭状态选择 (Off-state selection for Idle mode)

此位在 MOE=0 时作用于配置为输出的通道。

有关详细信息，请参见 OC/OCN 使能说明（*第 379 页的第 14.4.9 节：TIM1 和 TIM8 捕获/比较使能寄存器 (TIMx_CCER)*）。

0：处于无效状态时，禁止 OC/OCN 输出（OC/OCN 使能输出信号=0）。

1：处于无效状态时，一旦 CCxE=1 或 CCxNE=1，便将 OC/OCN 输出首先强制为其空闲电平。然后设置 OC/OCN 使能输出信号=1

注意：编程了 LOCK（TIMx_BDTR 寄存器中的 LOCK 位）级别 2 后，此位即无法修改。

位 9:8 **LOCK[1:0]**：锁定配置 (Lock configuration)

这些位用于针对软件错误提供写保护。

00：关闭锁定——不对任何位提供写保护。

01：锁定级别 1，此时无法对 TIMx_BDTR 寄存器中的 DTG 位、TIMx_CR2 寄存器中的 OISx 和 OISxN 位以及 TIMx_BDTR 寄存器中的 BKE/BKP/AOE 位执行写操作。

10：锁定级别 2，此时无法对锁定级别 1 中适用的各位、CC 极性位（TIMx_CCER 寄存器中的 CCxP/CCxNP 位，只要通过 CCxS 位将相关通道配置为输出）以及 OSSR 和 OSSI 位执行写操作。

11：锁定级别 3，此时无法对锁定级别 2 中适用的各位、CC 控制位（TIMx_CCMRx 寄存器中的 OCxM 和 OCxPE 位，只要通过 CCxS 位将相关通道配置为输出）执行写操作。

注意：复位后只能对 LOCK 位执行一次写操作。对 TIMx_BDTR 寄存器执行写操作后其中的内容将冻结，直到下一次复位。

位 7:0 **DTG[7:0]**：配置死区发生器 (Dead-time generator setup)

此位域定义插入到互补输出之间的死区持续时间。DT 与该持续时间相对应。

DTG[7:5]=0xx => DT=DTG[7:0]x t_{dtg}，其中 $t_{dtg}=t_{DTS}$。

DTG[7:5]=10x => DT=(64+DTG[5:0])xt_{dtg}，其中 $T_{dtg}=2xt_{DTS}$。

DTG[7:5]=110 => DT=(32+DTG[4:0])xt_{dtg}，其中 $T_{dtg}=8xt_{DTS}$。

DTG[7:5]=111 => DT=(32+DTG[4:0])xt_{dtg}，其中 $T_{dtg}=16xt_{DTS}$。

示例：如果 T_{DTS}=125ns (8MHz)，则可能的死区值为：

0 到 15875 ns（步长为 125 ns），

16 us 到 31750 ns（步长为 250 ns），

32 us 到 63us（步长为 1 us），

64 us 到 126 us（步长为 2 us）

注意：只要编程了 LOCK（TIMx_BDTR 寄存器中的 LOCK 位）级别 1、2 或 3，此位域即无法修改。

14.4.19 TIM1 和 TIM8 DMA 控制寄存器 (TIMx_DCR)

TIM1&TIM8 DMA control register

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
Reserved			DBL[4:0]					Reserved			DBA[4:0]				
			rw	rw	rw	rw	rw				rw	rw	rw	rw	rw

位 15:13 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 5 位定义了 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）。

TIMx_DMAR 地址）

00000：1 次传输

00001：2 次传输

00010：3 次传输

...

10001：18 次传输

位 7:5 保留，必须保持复位值。

位 4:0 **DBA[4:0]**：DMA 基址 (DMA base address)

该 5 位向量定义 DMA 传输的基址（通过 TIMx_DMAR 地址进行读/写访问时）。DBA 定义为从 TIMx_CR1 寄存器地址开始计算的偏移量。

示例：

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。

14.4.20 TIM1 和 TIM8 全传输 DMA 地址 (TIMx_DMAR)

TIM1&TIM8 DMA address for full transfer

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	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 15:0 **DMAB[15:0]**：DMA 连续传送寄存器 (DMA register for burst accesses)

对 DMAR 寄存器执行读或写操作将访问位于如下地址的寄存器

(TIMx_CR1 地址) + (DBA + DMA 索引) x 4

其中 TIMx_CR1 地址为控制寄存器 1 的地址，DBA 为 TIMx_DCR 寄存器中配置的 DMA 基址，DMA 索引由 DMA 传输自动控制，其范围介于 0 到 DBL（TIMx_DCR 寄存器中配置的 DBL）之间。

DMA 连续传送功能使用方法示例

本例中，定时器 DMA 连续传送功能用于将 CCRx 寄存器（x = 2、3、4）的内容更新为通过 DMA 传输到 CCRx 寄存器中的多个半字。

具体操作步骤如下：

1. 将相应的 DMA 通道配置如下：
   - DMA 通道外设地址为 DMAR 寄存器地址
   - DMA 通道存储器地址为包含要通过 DMA 传输到 CCRx 寄存器的数据的 RAM 缓冲区地址。
   - 要传输的数据量 = 3（参见下文注释）。
   - 禁止循环模式。
2. 通过将 DBA 和 DBL 位域配置如下来配置 DCR 寄存器：
   DBL = 3 次传输，DBA = 0xE。
3. 使能 TIMx 更新 DMA 请求（DIER 寄存器中的 UDE 位置 1）。
4. 使能 TIMx
5. 使能 DMA 通道

注意： *本例适用于每个 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。数据将按照如下方式传输到 CCRx 寄存器：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；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。*

14.4.21 TIM1 和 TIM8 寄存器映射

TIM1 和 TIM8 寄存器可映射为 16 位可寻址寄存器，如下表所述：

表 74. TIM1 和 TIM8 寄存器映射和复位值

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	TIMx_CR1	Reserved																						CKD [1:0]		ARPE	CMS [1:0]		DIR	OPM	URS	UDIS	CEN
	Reset value																							0	0	0	0	0	0	0	0	0	0
0x04	TIMx_CR2	Reserved																	OIS4	OIS3N	OIS3	OIS2N	OIS2	OIS1N	OIS1	TI1S	MMS[2:0]			CCDS	CCUS	Reserved	CCPC
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0		0
0x08	TIMx_SMCR	Reserved																ETP	ECE	ETPS [1:0]		ETF[3:0]				MSM	TS[2:0]			Reserved	SMS[2:0]		
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0
0x0C	TIMx_DIER	Reserved																	TDE	COMDE	CC4DE	CC3DE	CC2DE	CC1DE	UDE	BIE	TIE	COMIE	CC4IE	CC3IE	CC2IE	CC1IE	UIE
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	TIMx_SR	Reserved																			CC4OF	CC3OF	CC2OF	CC1OF	Reserved	BIF	TIF	COMIF	CC4IF	CC3IF	CC2IF	CC1IF	UIF
	Reset value																				0	0	0	0		0	0	0	0	0	0	0	0
0x14	TIMx_EGR	Reserved																								BG	TG	COM	CC4G	CC3G	CC2G	CC1G	UG
	Reset value																									0	0	0	0	0	0	0	0
0x18	TIMx_CCMR1 Output Compare mode	Reserved																OC2CE	OC2M [2:0]			OC2PE	OC2FE	CC2S [1:0]		OC1CE	OC1M [2:0]			OC1PE	OC1FE	CC1S [1:0]	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	TIMx_CCMR1 Input Capture mode	Reserved																IC2F[3:0]				IC2 PSC [1:0]		CC2S [1:0]		IC1F[3:0]				IC1 PSC [1:0]		CC1S [1:0]	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1C	TIMx_CCMR2 Output Compare mode	Reserved																O24CE	OC4M [2:0]			OC4PE	OC4FE	CC4S [1:0]		OC3CE	OC3M [2:0]			OC3PE	OC3FE	CC3S [1:0]	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	TIMx_CCMR2 Input Capture mode	Reserved																IC4F[3:0]				IC4 PSC [1:0]		CC4S [1:0]		IC3F[3:0]				IC3 PSC [1:0]		CC3S [1:0]	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20	TIMx_CCER	Reserved																		CC4P	CC4E	CC3NP	CC3NE	CC3P	CC3E	CC2NP	CC2NE	CC2P	CC2E	CC1NP	CC1NE	CC1P	CC1E
	Reset value																			0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x24	TIMx_CNT	Reserved																CNT[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x28	TIMx_PSC	Reserved																PSC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x2C	TIMx_ARR	Reserved																ARR[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x30	TIMx_RCR	Reserved																								REP[7:0]							
	Reset value																									0	0	0	0	0	0	0	0

表 74. TIM1 和 TIM8 寄存器映射和复位值 （续）

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x34	TIMx_CCR1	Reserved																CCR1[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x38	TIMx_CCR2	Reserved																CCR2[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x3C	TIMx_CCR3	Reserved																CCR3[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x40	TIMx_CCR4	Reserved																CCR4[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x44	TIMx_BDTR	Reserved																MOE	AOE	BKP	BKE	OSSR	OSSI	LOCK [1:0]		DT[7:0]							
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x48	TIMx_DCR	Reserved																			DBL[4:0]					Reserved	DBA[4:0]						
	Reset value																				0	0	0	0	0		0	0	0	0	0	0	0
0x4C	TIMx_DMAR	Reserved																DMAB[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

有关寄存器边界地址的信息，请参见*第 52 页的表 2*。

15 通用定时器（TIM2 到 TIM5）

除非特别说明，否则本部分适用于整个 STM32F4xx 系列。

15.1 TIM2 到 TIM5 简介

通用定时器包含一个 16 位或 32 位自动重载计数器，该计数器由可编程预分频器驱动。

它们可用于多种用途，包括测量输入信号的脉冲宽度（*输入捕获*）或生成输出波形（*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，可将脉冲宽度和波形周期从几微秒调制到几毫秒。

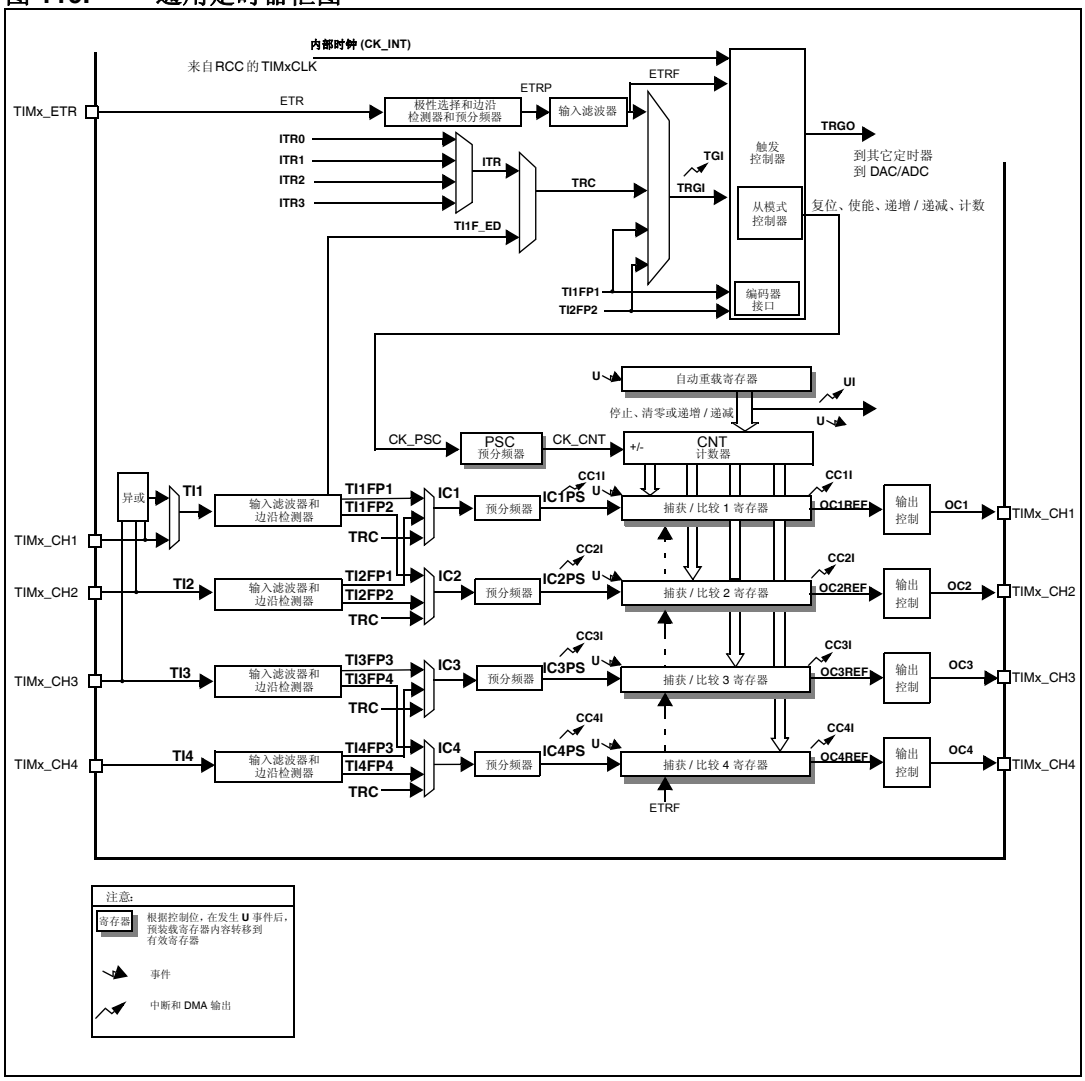
这些定时器彼此完全独立，不共享任何资源。如*第 15.3.15 节*中所述，它们可以同步操作。

15.2 TIM2 到 TIM5 主要特性

通用 TIMx 定时器具有以下特性：

- 16 位（TIM3 和 TIM4）或 32 位（TIM2 和 TIM5） 递增、递减和递增/递减自动重载计数器。
- 16 位可编程预分频器，用于对计数器时钟频率进行分频 （即运行时修改），分频系数介于 1 到 65536 之间。
- 多达 4 个独立通道，可用于：
  - 输入捕获
  - 输出比较
  - PWM 生成（边沿和中心对齐模式）
  - 单脉冲模式输出
- 使用外部信号控制定时器且可实现多个定时器互连的同步电路。
- 发生如下事件时生成中断/DMA 请求：
  - 更新：计数器上溢/下溢、计数器初始化（通过软件或内部/外部触发）
  - 触发事件（计数器启动、停止、初始化或通过内部/外部触发计数）
  - 输入捕获
  - 输出比较
- 支持定位用增量（正交）编码器和霍尔传感器电路
- 外部时钟触发输入或逐周期电流管理

图 119.　通用定时器框图



15.3　TIM2 到 TIM5 功能说明

15.3.1　时基单元

可编程定时器的主要模块由一个 16 位/32 位计数器及其相关的自动重装寄存器组成。此计数器可采用递增方式计数。计数器的时钟可通过预分频器进行分频。

计数器、自动重载寄存器和预分频器寄存器可通过软件进行读写。即使在计数器运行时也可执行读写操作。

时基单元包括：

- 计数器寄存器 (TIMx_CNT)
- 预分频器寄存器 (TIMx_PSC)
- 自动重载寄存器 (TIMx_ARR)

自动重载寄存器是预装载的。对自动重载寄存器执行写入或读取操作时会访问预装载寄存器。预装载寄存器的内容既可以直接传送到影子寄存器，也可以在每次发生更新事件 (UEV) 时传送到影子寄存器，这取决于 TIMx_CR1 寄存器中的自动重载预装载使能位 (ARPE)。当计数器达到上溢值（或者在递减计数时达到下溢值）并且 TIMx_CR1 寄存器中的 UDIS 位为 0 时，将发送更新事件。该更新事件也可由软件产生。下文将针对各配置的更新事件的产生进行详细介绍。

计数器由预分频器输出 CK_CNT 提供时钟，仅当 TIMx_CR1 寄存器中的计数器启动位 (CEN) 置 1 时，才会启动计数器（有关计数器使能的更多详细信息，另请参见从模式控制器的相关说明）。

请注意，真正的计数器使能信号 CNT_EN 在 CEN 置 1 的一个时钟周期后被置 1。

预分频器说明

预分频器可对计数器时钟频率进行分频，分频系数介于 1 到 65536 之间。该预分频器基于 16 位/32 位寄存器（TIMx_PSC 寄存器）所控制的 16 位计数器。由于该控制寄存器具有缓冲功能，因此预分频器可实现实时更改。而新的预分频比将在下一更新事件发生时被采用。

图 120 和 *图 121* 给出了在预分频比发生实时变化时一些计数器行为的示例：

图 120. 预分频器分频由 1 变为 2 时的计数器时序图

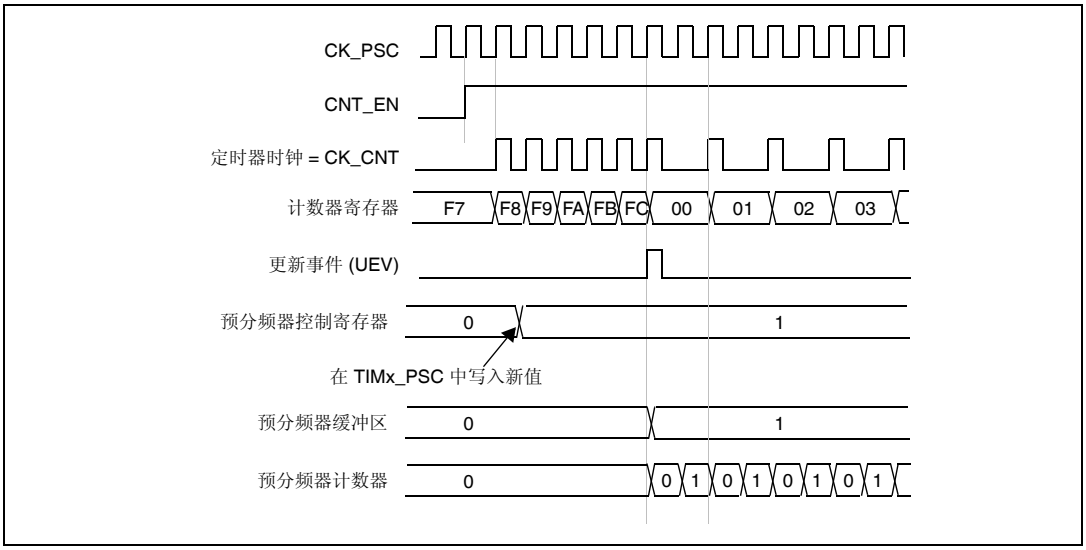
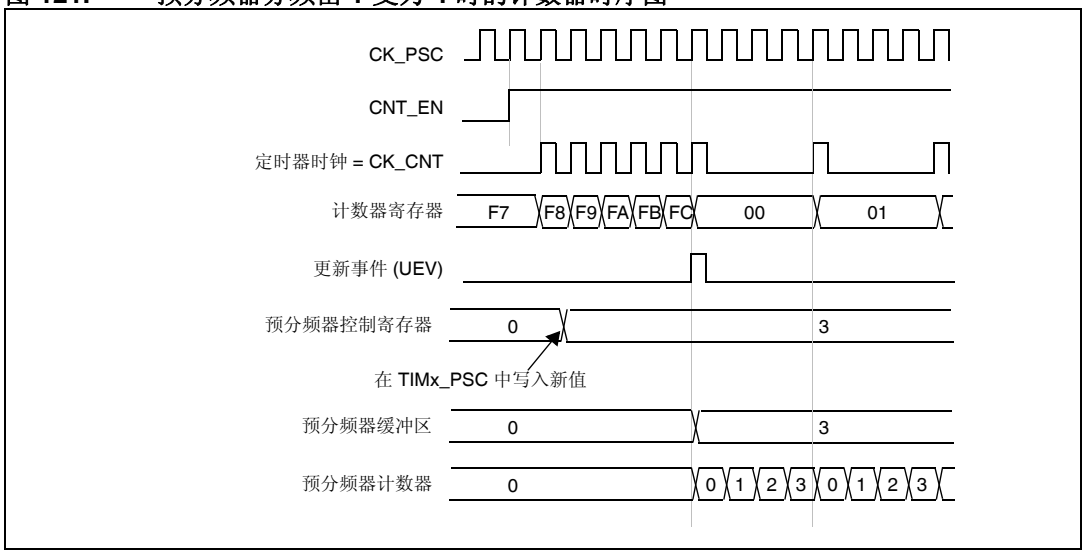


图 121. 预分频器分频由 1 变为 4 时的计数器时序图



15.3.2 计数器模式

递增计数模式

在递增计数模式下，计数器从 0 计数到自动重载值（TIMx_ARR 寄存器的内容），然后重新从 0 开始计数并生成计数器上溢事件。

每次发生计数器上溢时会生成更新事件，或将 TIMx_EGR 寄存器中的 UG 位置 1（通过软件或使用从模式控制器）也可以生成更新事件。

通过软件将 TIMx_CR1 寄存器中的 UDIS 位置 1 可禁止 UEV 事件。这可避免向预装载寄存器写入新值时更新影子寄存器。在 UDIS 位写入 0 之前不会产生任何更新事件。不过，计数器和预分频器计数器都会重新从 0 开始计数（而预分频比保持不变）。此外，如果 TIMx_CR1 寄存器中的 URS 位（更新请求选择）已置 1，则将 UG 位置 1 会生成更新事件 UEV，但不会将 UIF 标志置 1（因此，不会发送任何中断或 DMA 请求）。这样一来，如果在发生捕获事件时将计数器清零，将不会同时产生更新中断和捕获中断。

发生更新事件时，将更新所有寄存器且将更新标志（TIMx_SR 寄存器中的 UIF 位）置 1（取决于 URS 位）：

- 预分频器的缓冲区中将重新装载预装载值（TIMx_PSC 寄存器的内容）
- 自动重载影子寄存器将以预装载值进行更新

以下各图以一些示例说明当 TIMx_ARR=0x36 时不同时钟频率下计数器的行为。

图 122.　计数器时序图，1 分频内部时钟

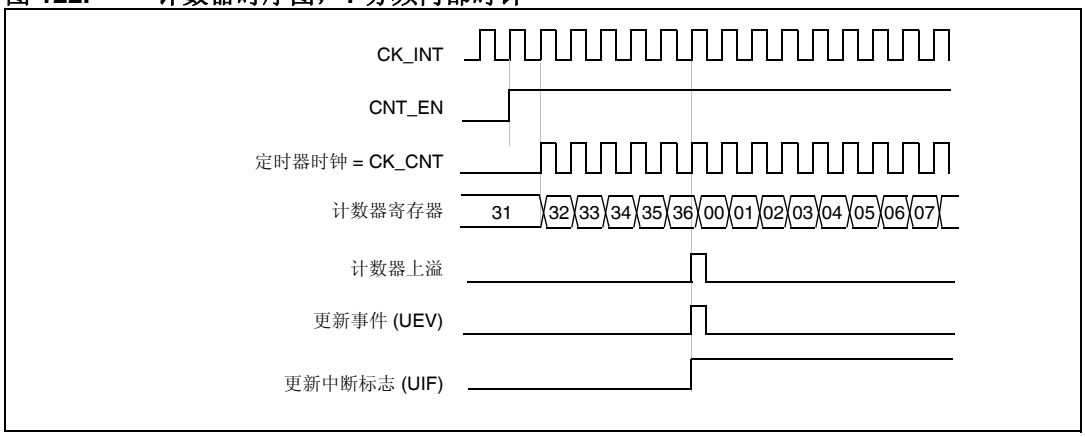


图 123.　计数器时序图，2 分频内部时钟

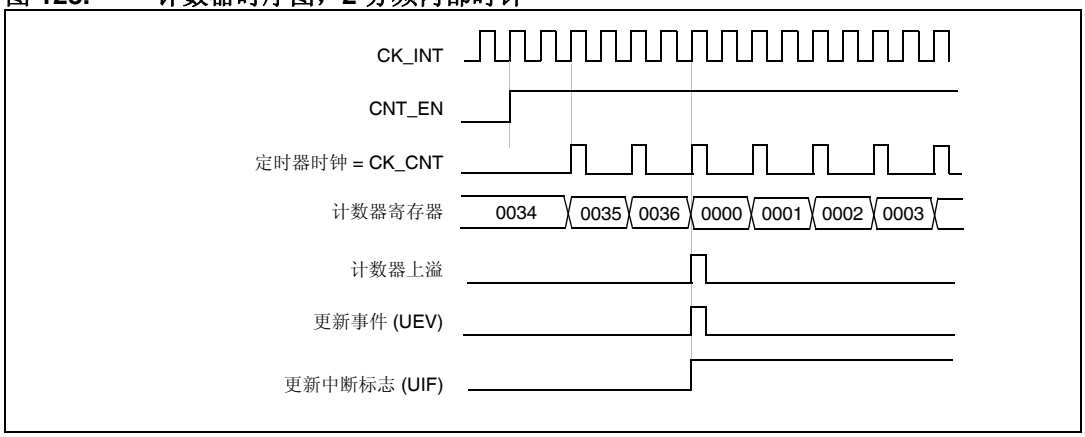


图 124.　计数器时序图，4 分频内部时钟

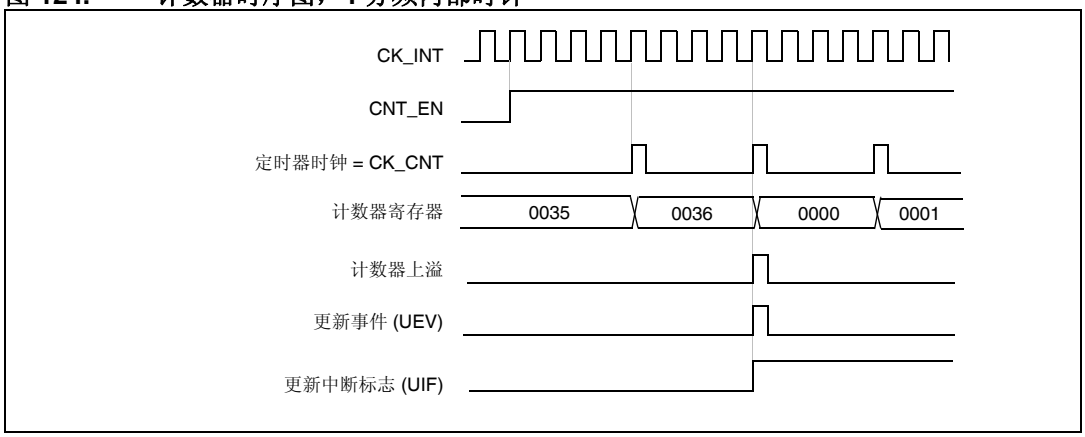


图 125.　计数器时序图，N 分频内部时钟

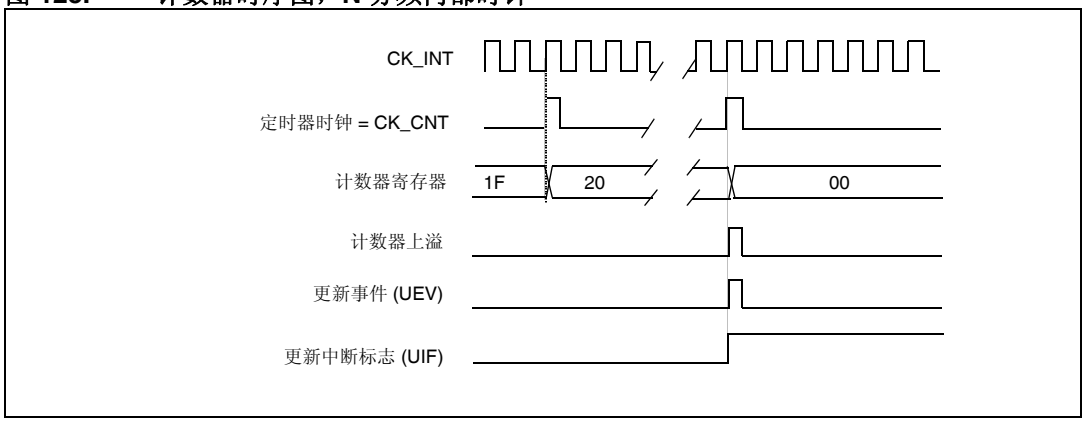


图 126.　计数器时序图，ARPE=0 时更新事件（TIMx_ARR 未预装载）

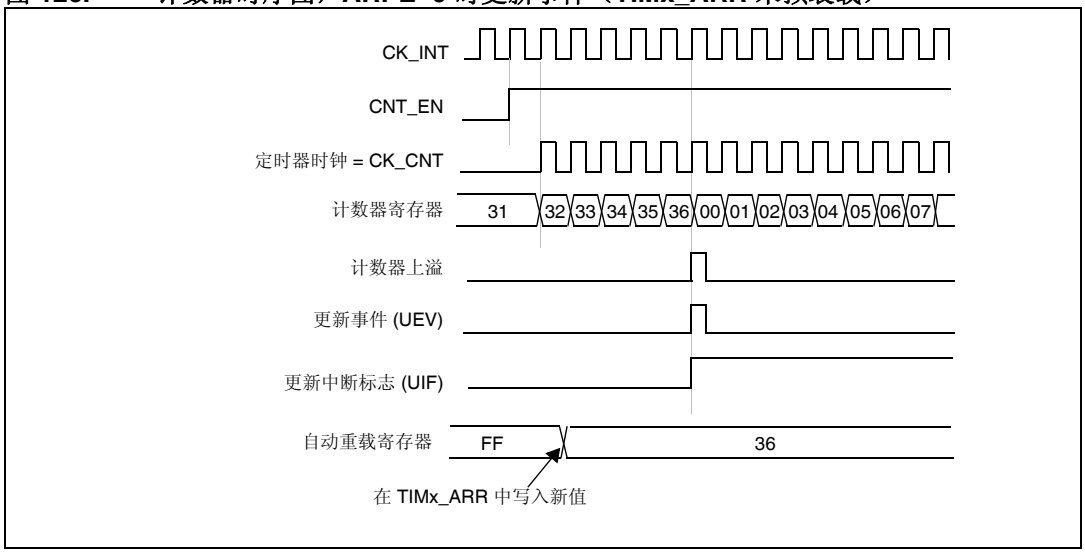
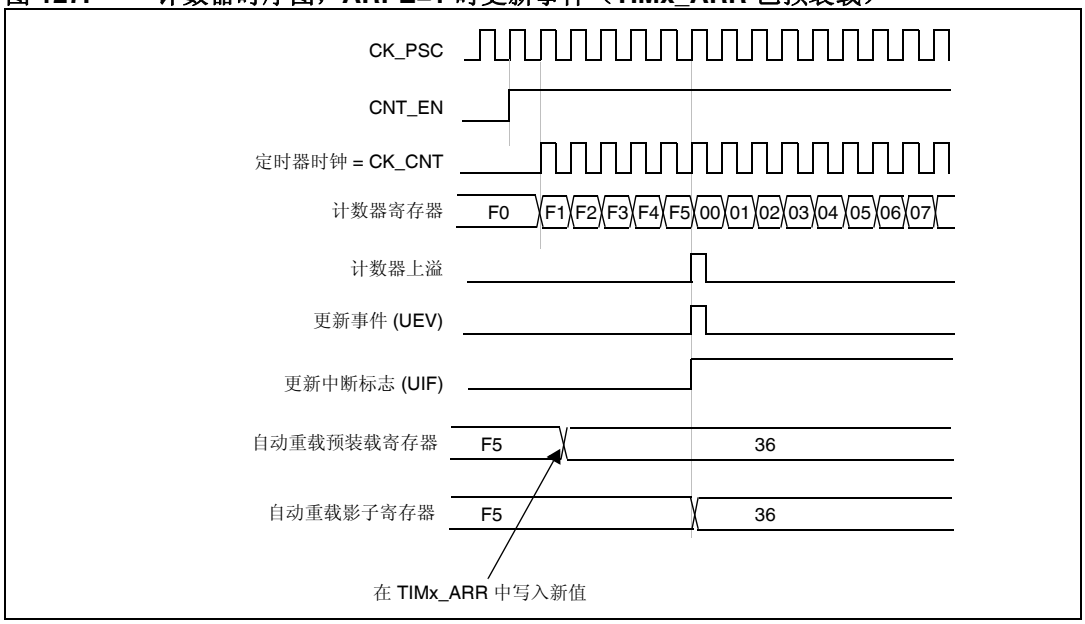


图 127.　　计数器时序图，ARPE=1 时更新事件（TIMx_ARR 已预装载）



递减计数模式

在递减计数模式下，计数器从自动重载值（TIMx_ARR 寄存器的内容）开始递减计数到 0，然后重新从自动重载值开始计数并生成计数器下溢事件。

每次发生计数器下溢时会生成更新事件，或将 TIMx_EGR 寄存器中的 UG 位置 1（通过软件或使用从模式控制器）也可以生成更新事件

通过软件将 TIMx_CR1 寄存器中的 UDIS 位置 1 可禁止 UEV 更新事件。这可避免向预装载寄存器写入新值时更新影子寄存器。在 UDIS 位写入 0 之前不会产生任何更新事件。不过，计数器会重新从当前自动重载值开始计数，而预分频器计数器则重新从 0 开始计数（但预分频比保持不变）。

此外，如果 TIMx_CR1 寄存器中的 URS 位（更新请求选择）已置 1，则将 UG 位置 1 会生成更新事件 UEV，但不会将 UIF 标志置 1（因此，不会发送任何中断或 DMA 请求）。这样一来，如果在发生捕获事件时将计数器清零，将不会同时产生更新中断和捕获中断。

发生更新事件时，将更新所有寄存器且将更新标志（TIMx_SR 寄存器中的 UIF 位）置 1（取决于 URS 位）：

- 预分频器的缓冲区中将重新装载预装载值（TIMx_PSC 寄存器的内容）。
- 自动重载活动寄存器将以预装载值（TIMx_ARR 寄存器的内容）进行更新。注意，自动重载寄存器会在计数器重载之前得到更新，因此，下一个计数周期就是我们所希望的新的周期长度。

以下各图以一些示例说明当 TIMx_ARR=0x36 时不同时钟频率下计数器的行为。。

图 128.　　计数器时序图，1 分频内部时钟

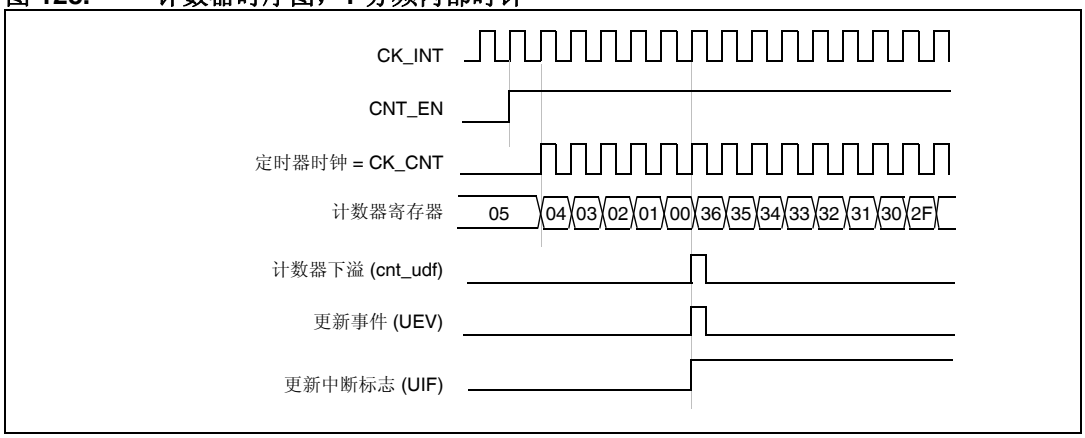


图 129.　　计数器时序图，2 分频内部时钟

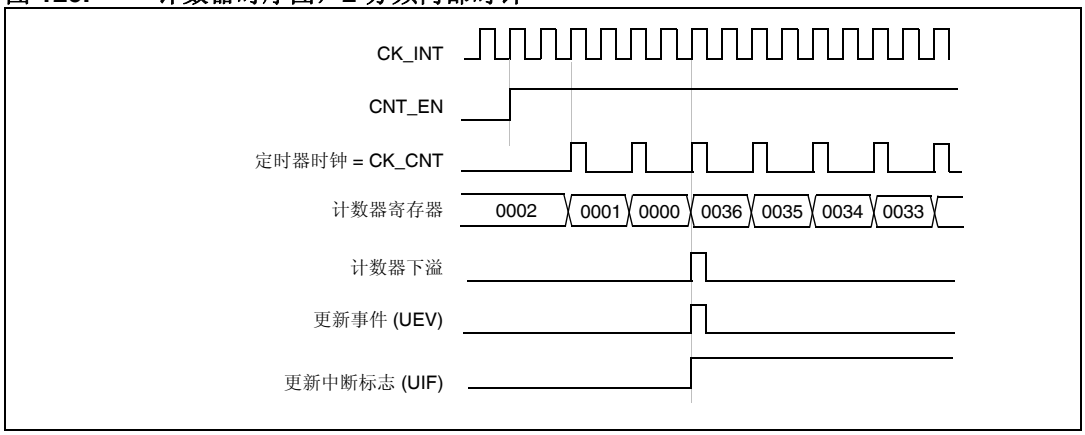


图 130.　　计数器时序图，4 分频内部时钟

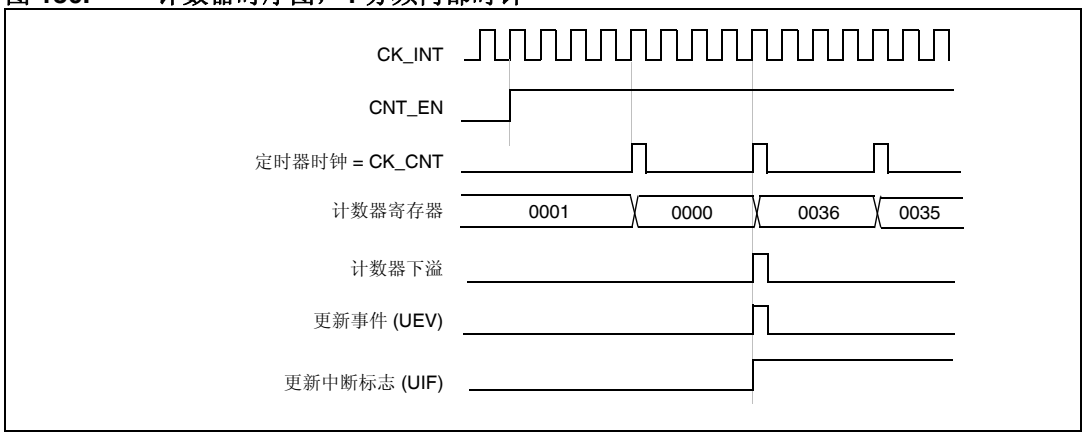


图 131. 计数器时序图，N 分频内部时钟

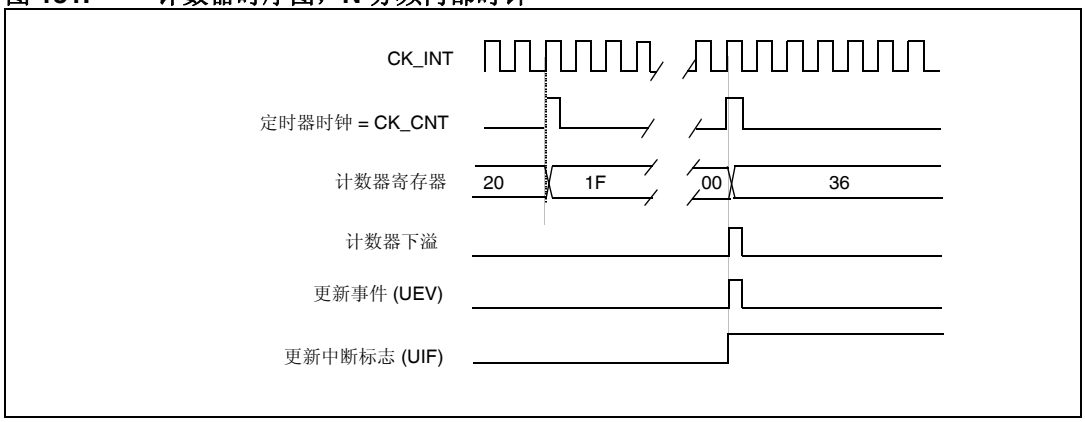
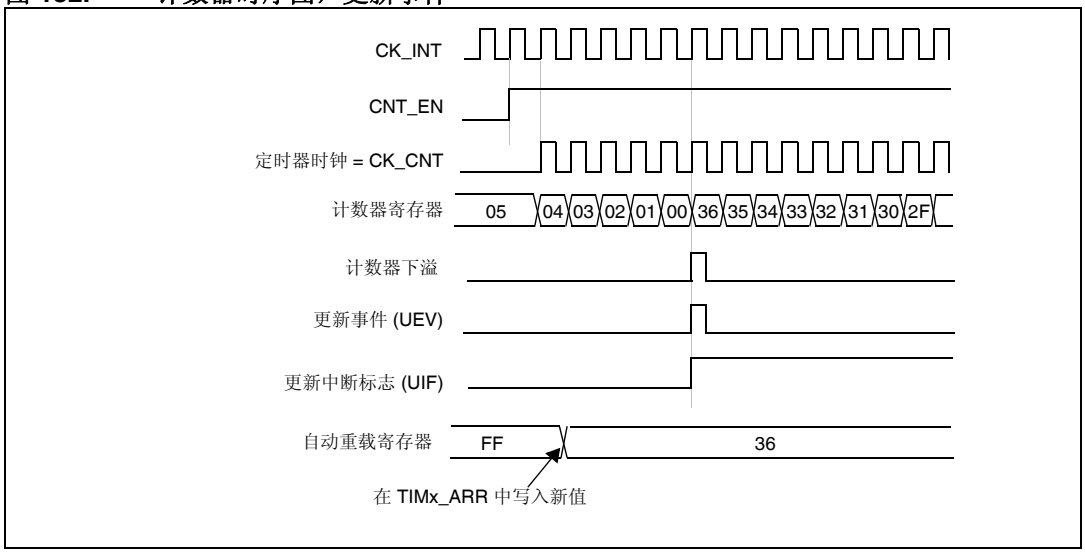


图 132. 计数器时序图，更新事件



中心对齐模式（递增/递减计数）

在中心对齐模式下，计数器从 0 开始计数到自动重载值（TIMx_ARR 寄存器的内容）– 1，生成计数器上溢事件；然后从自动重载值开始向下计数到 1 并生成计数器下溢事件。之后从 0 开始重新计数。

当 TIMx_CR1 寄存器中的 CMS 位不为“00”时，中心对齐模式有效。将通道配置为输出模式时，其输出比较中断标志将在以下模式下置 1，即：计数器递减计数（中心对齐模式 1，CMS = “01”）、计数器递增计数（中心对齐模式 2，CMS = “10”）以及计数器递增/递减计数（中心对齐模式 3，CMS = “11”）。

此模式下无法写入方向位（TIMx_CR1 寄存器中的 DIR 位）。而是由硬件更新并指示当前计数器方向。

每次发生计数器上溢和下溢时都会生成更新事件，或将 TIMx_EGR 寄存器中的 UG 位置 1（通过软件或使用从模式控制器）也可以生成更新事件。这种情况下，计数器以及预分频器计数器将重新从 0 开始计数。

通过软件将 TIMx_CR1 寄存器中的 UDIS 位置 1 可禁止 UEV 更新事件。这可避免向预装载寄存器写入新值时更新影子寄存器。在 UDIS 位写入 0 之前不会产生任何更新事件。不过，计数器仍会根据当前自动重载值进行递增和递减计数。

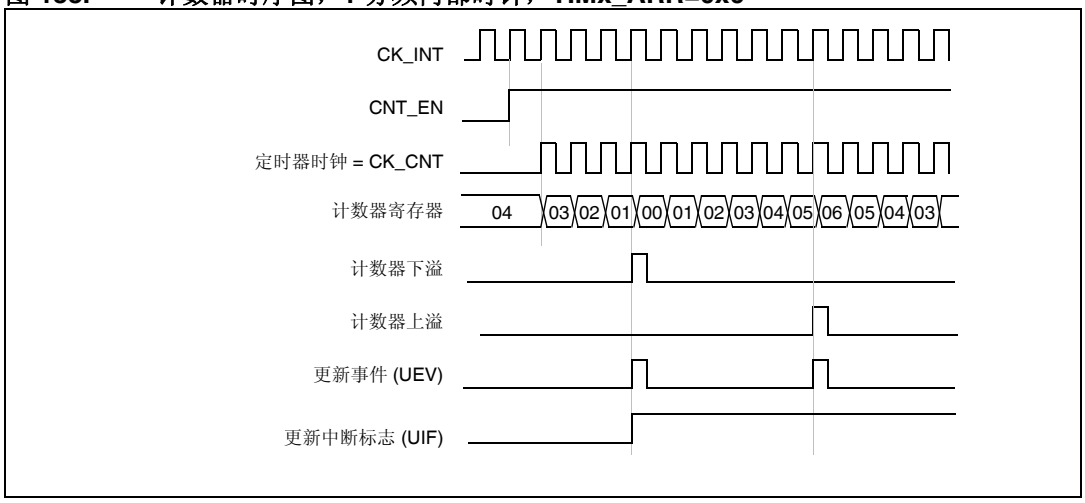
此外，如果 TIMx_CR1 寄存器中的 URS 位（更新请求选择）已置 1，则将 UG 位置 1 会生成更新事件 UEV，但不会将 UIF 标志置 1（因此，不会发送任何中断或 DMA 请求）。这样一来，如果在发生捕获事件时将计数器清零，将不会同时产生更新中断和捕获中断。

发生更新事件时，将更新所有寄存器且将更新标志（TIMx_SR 寄存器中的 UIF 位）置 1（取决于 URS 位）：

- 预分频器的缓冲区中将重新装载预装载值（TIMx_PSC 寄存器的内容）。
- 自动重载活动寄存器将以预装载值 （TIMx_ARR 寄存器的内容）进行更新。注意，如果更新操作是由计数器上溢触发的，则自动重载寄存器在重载计数器之前更新，因此，下一个计数周期就是我们所希望的新的周期长度（计数器被重载新的值）。

以下各图以一些示例说明不同时钟频率下计数器的行为。

图 133. 计数器时序图，1 分频内部时钟，TIMx_ARR=0x6



1. 此处使用了中心对齐模式 1（有关详细信息，请参见*第 424 页的第 15.4.1 节：TIMx 控制寄存器 1 (TIMx_CR1)*）。

图 134. 计数器时序图，2 分频内部时钟

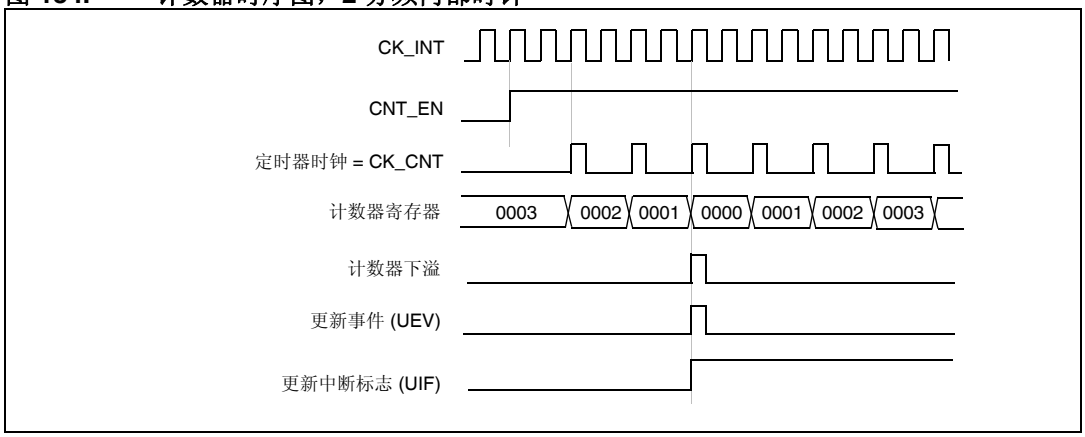
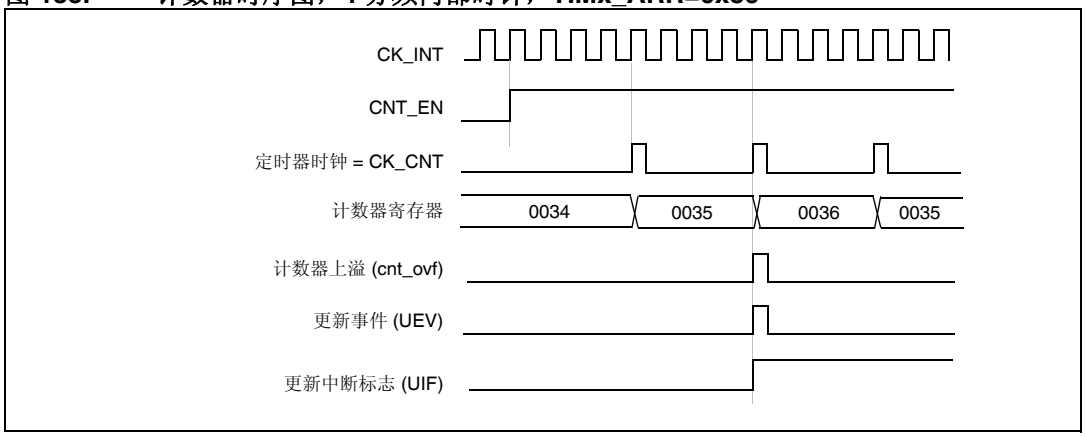


图 135. 计数器时序图，4 分频内部时钟，TIMx_ARR=0x36



1. 中心对齐模式 2 或模式 3 与上溢 UIF 结合使用。

图 136. 计数器时序图，N 分频内部时钟

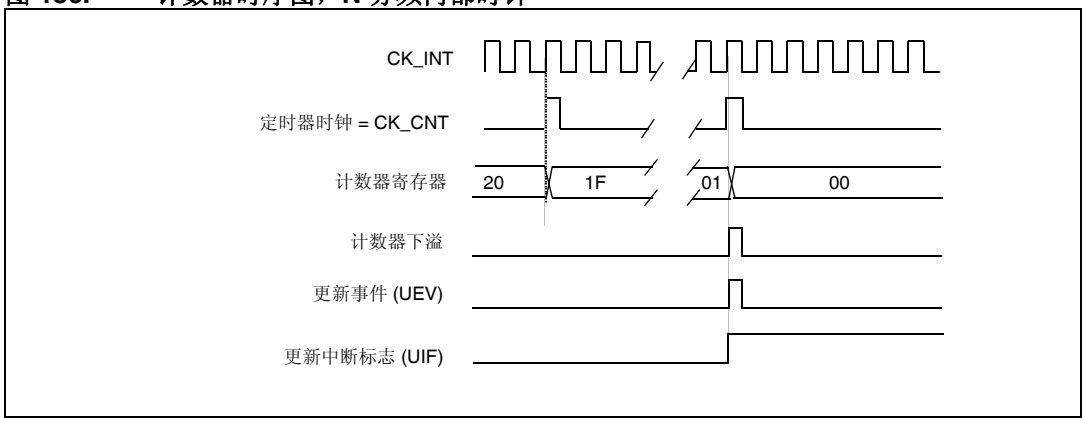


图 137. 计数器时序图，ARPE=1 时的更新事件（计数器下溢）

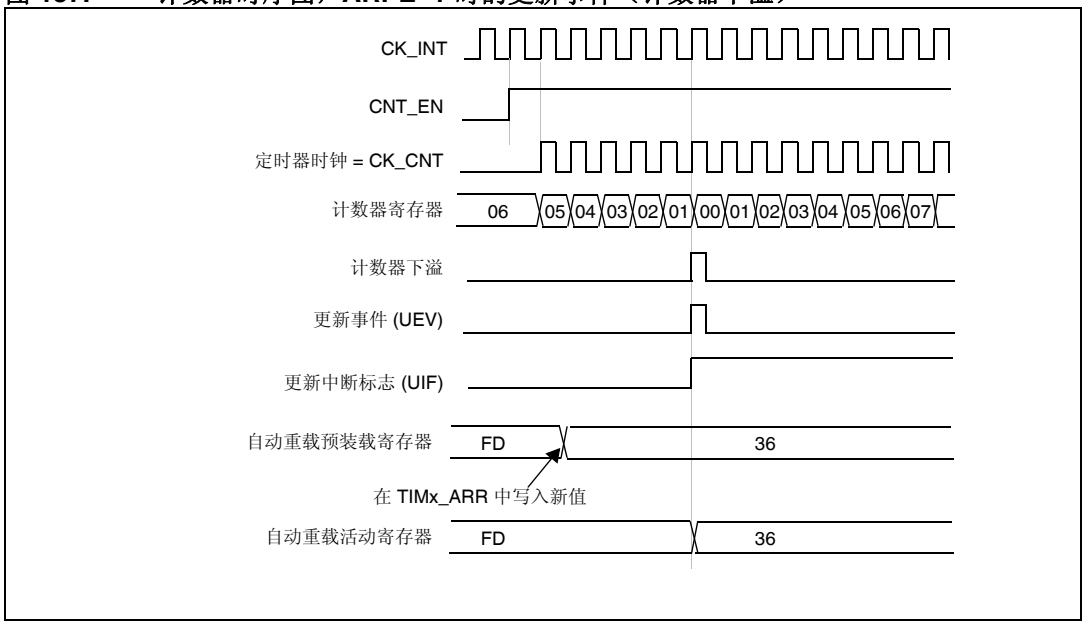
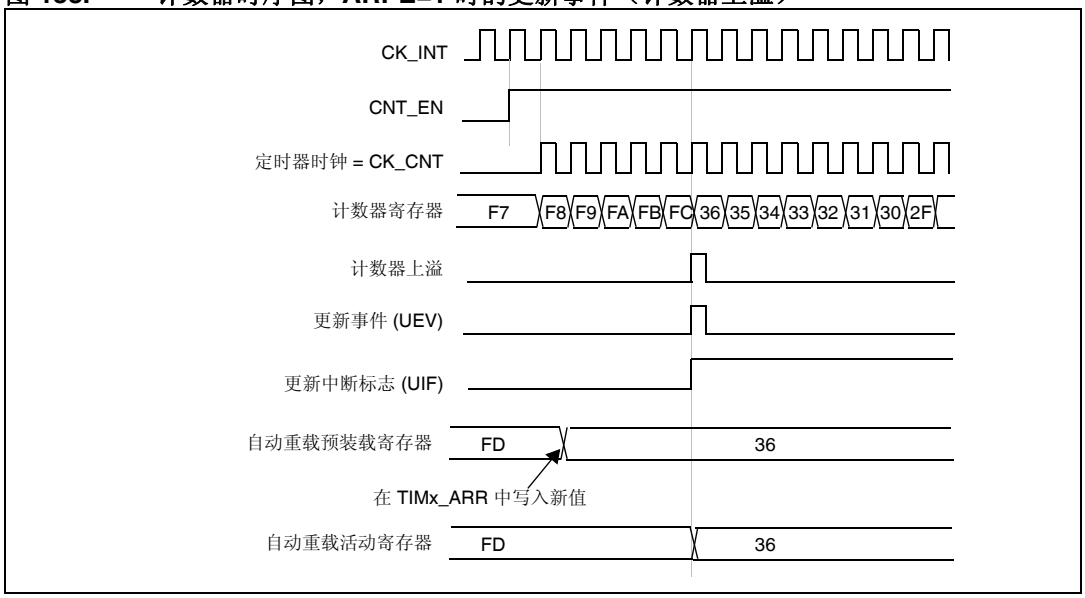


图 138. 计数器时序图，ARPE=1 时的更新事件（计数器上溢）



15.3.3 时钟选择

计数器时钟可由下列时钟源提供：

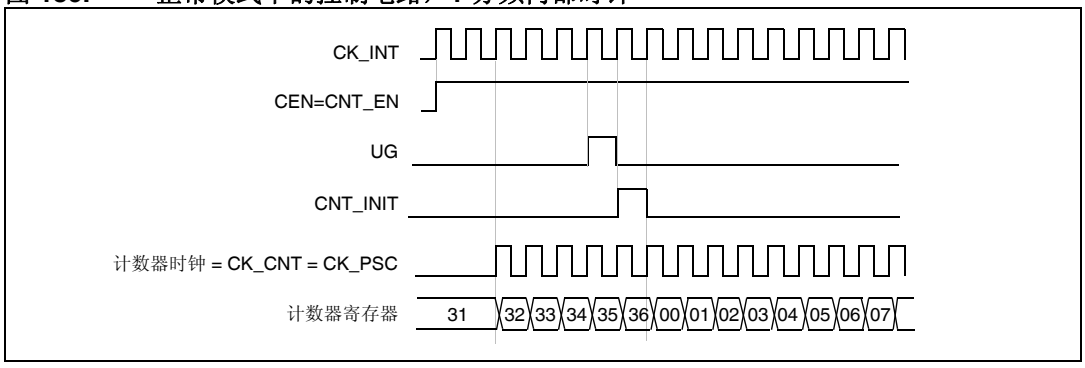
- 内部时钟 (CK_INT)
- 外部时钟模式 1：外部输入引脚 (TIx)
- 外部时钟模式 2：外部触发输入 (ETR)，仅适用于 TIM2、TIM3 和 TIM4。
- 内部触发输入 (ITRx)：使用一个定时器作为另一个定时器的预分频器，例如可以将定时器配置为定时器 2 的预分频器。有关详细信息，请参见*第 419 页的将一个定时器用作另一个定时器的预分频器*。

内部时钟源 (CK_INT)

如果禁止从模式控制器（TIMx_SMCR 寄存器中 SMS=000），则 CEN 位、DIR 位（TIMx_CR1 寄存器中）和 UG 位（TIMx_EGR 寄存器中）为实际控制位，并且只能通过软件进行更改（UG 除外，仍自动清零）。当对 CEN 位写入 1 时，预分频器的时钟就由内部时钟 CK_INT 提供。

图 139 显示了正常模式下控制电路与递增计数器的行为（没有预分频的情况下）。

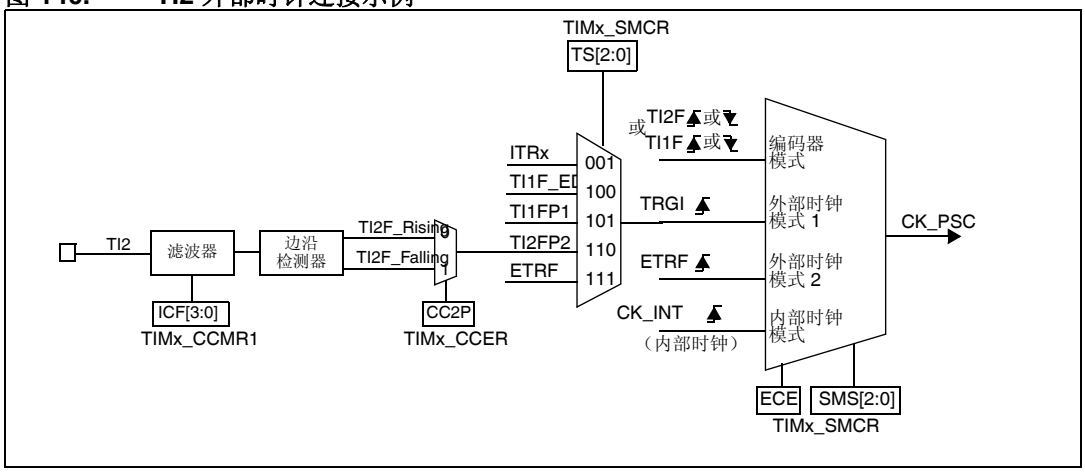
图 139. 正常模式下的控制电路，1 分频内部时钟



外部时钟源模式 1

当 TIMx_SMCR 寄存器中的 SMS=111 时，可选择此模式。计数器可在选定的输入信号上出现上升沿或下降沿时计数。

图 140. TI2 外部时钟连接示例



例如，要使递增计数器在 TI2 输入出现上升沿时计数，请执行以下步骤：

1. 通过在 TIMx_CCMR1 寄存器中写入 CC2S=“01”来配置通道 2，使其能够检测 TI2 输入的上升沿。
2. 通过在 TIMx_CCMR1 寄存器中写入 IC2F[3:0] 位来配置输入滤波时间（如果不需要任何滤波，请保持 IC2F=0000）。

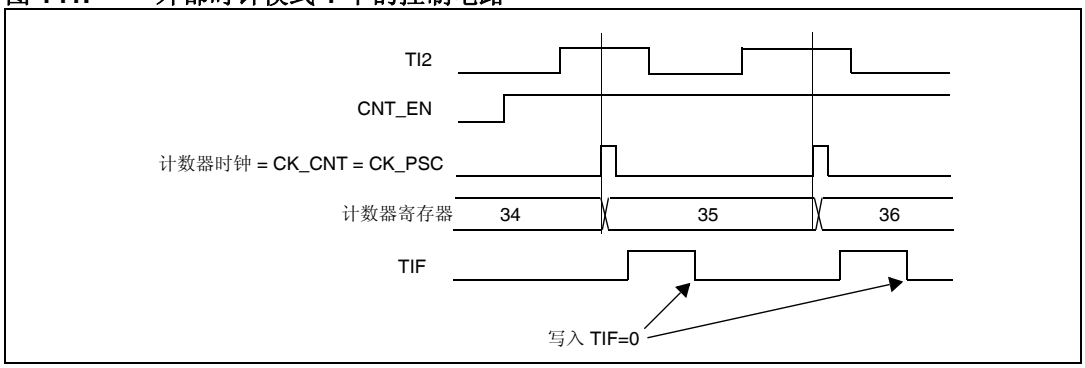
注意： *由于捕获预分频器不用于触发操作，因此无需对其进行配置。*

3. 通过在 TIMx_CCER 寄存器中写入 CC2P=0 和 CC2NP=0 来选择上升沿极性。
4. 通过在 TIMx_SMCR 寄存器中写入 SMS=111，使定时器在外部时钟模式 1 下工作。
5. 通过在 TIMx_SMCR 寄存器中写入 TS=110 来选择 TI2 作为输入源。
6. 通过在 TIMx_CR1 寄存器中写入 CEN=1 来使能计数器。

当 TI2 出现上升沿时，计数器便会计数一次并且 TIF 标志置 1。

TI2 的上升沿与实际计数器时钟之间的延迟是由于 TI2 输入的重新同步电路引起的。

图 141. 外部时钟模式 1 下的控制电路



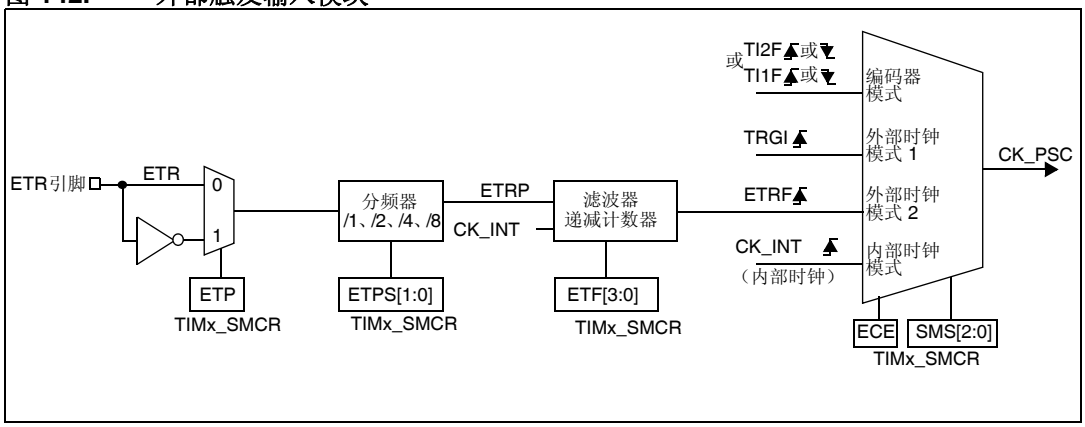
外部时钟源模式 2

通过在 TIMx_SMCR 寄存器中写入 ECE=1 可选择此模式。

计数器可在外部触发输入 ETR 出现上升沿或下降沿时计数。

图 142 简要介绍了外部触发输入模块。

图 142.　　外部触发输入模块



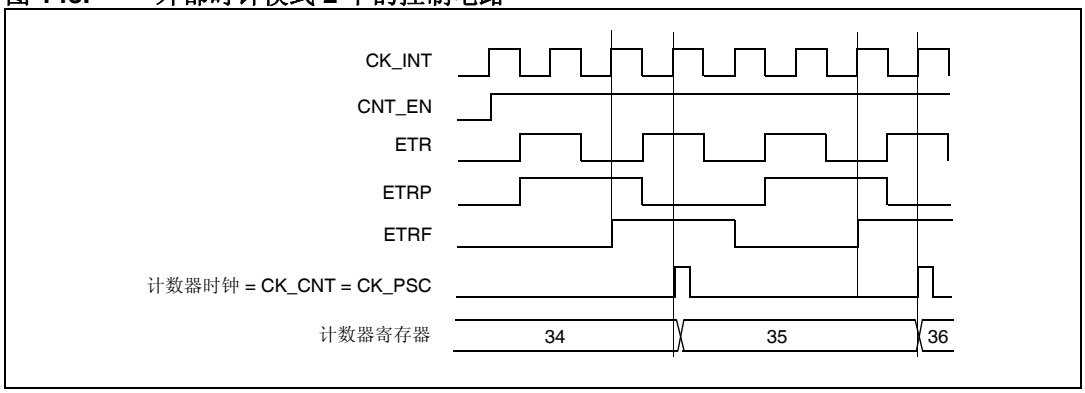
例如，要使递增计数器在 ETR 每出现 2 个上升沿时计数，请执行以下步骤：

1. 由于此例中不需滤波器，因此在 TIMx_SMCR 寄存器中写入 ETF[3:0]=0000。
2. 通过在 TIMx_SMCR 寄存器中写入 ETPS[1:0]=01 来设置预分频器
3. 通过在 TIMx_SMCR 寄存器中写入 ETP=0 来选择 ETR 引脚的上升沿检测
4. 通过在 TIMx_SMCR 寄存器中写入 ECE=1 来使能外部时钟模式 2。
5. 通过在 TIMx_CR1 寄存器中写入 CEN=1 来使能计数器。

ETR 每出现 2 个上升沿，计数器计数一次。

ETR 的上升沿与实际计数器时钟之间的延迟是由于 ETRP 信号的重新同步电路引起的。

图 143.　　外部时钟模式 2 下的控制电路



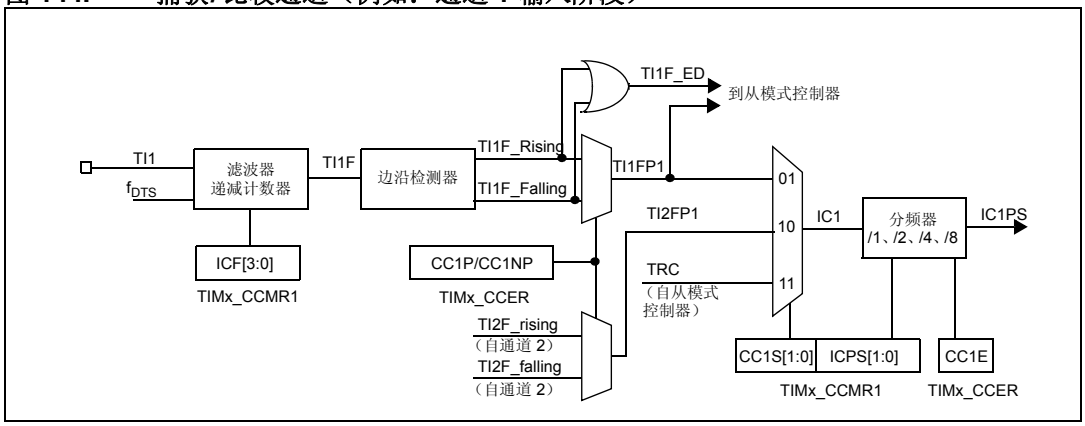
15.3.4 捕获/比较通道

每个捕获/比较通道均围绕一个捕获/比较寄存器（包括一个影子寄存器）、一个捕获输入阶段（数字滤波、多路复用和预分频器）和一个输出阶段（比较器和输出控制）构建而成。

下图概括介绍了一个捕获/比较通道。

输入阶段对相应的 TIx 输入进行采样，生成一个滤波后的信号 TIxF。然后，带有极性选择功能的边沿检测器生成一个信号 (TIxFPx)，该信号可用作从模式控制器的触发输入，也可用作捕获命令。该信号先进行预分频 (ICxPS)，而后再进入捕获寄存器。

图 144. 捕获/比较通道（例如：通道 1 输入阶段）



输出阶段生成一个中间波形作为基准：OCxRef（高电平有效）。链的末端决定最终输出信号的极性。

图 145. 捕获/比较通道 1 主电路

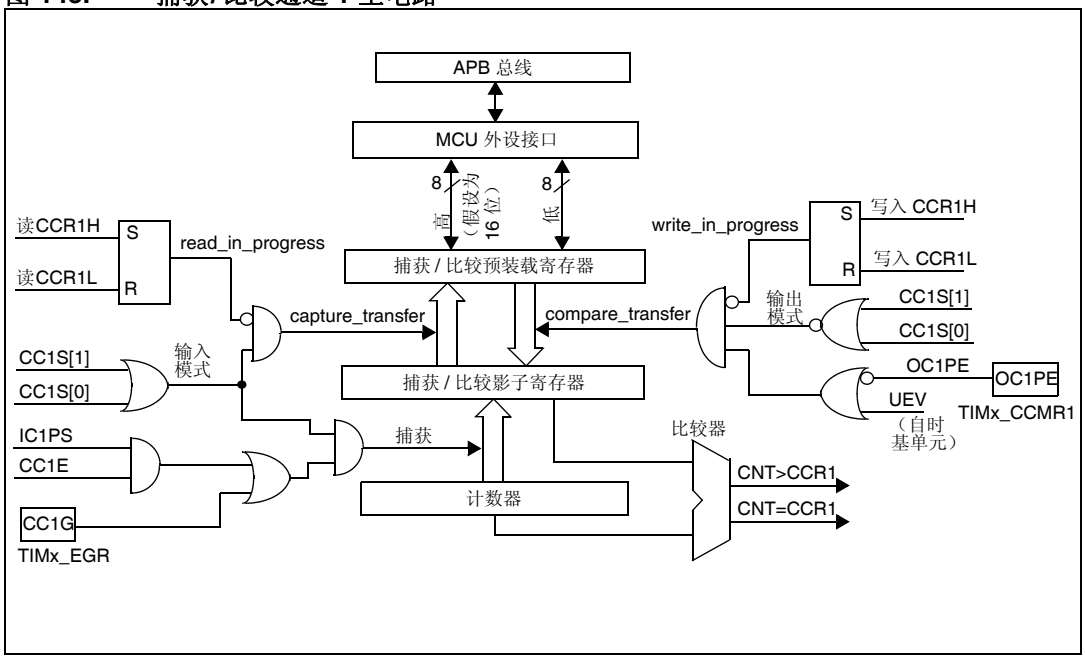
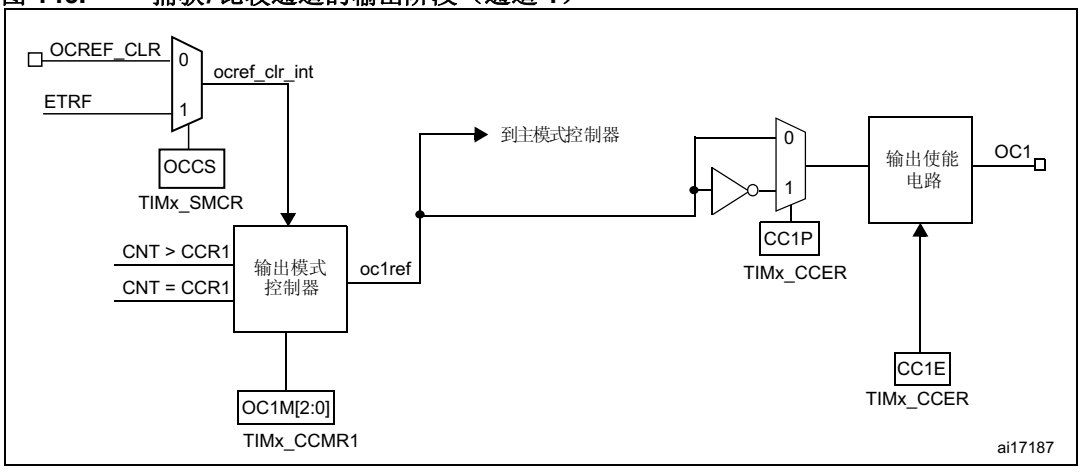


图 146. 捕获/比较通道的输出阶段（通道 1）



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。始终可通过读写操作访问预装载寄存器。

在捕获模式下，捕获实际发生在影子寄存器中，然后将影子寄存器的内容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将复制到影子寄存器中，然后将影子寄存器的内容与计数器进行比较。

15.3.5 输入捕获模式

在输入捕获模式下，当相应的 ICx 信号检测到跳变沿后，将使用捕获/比较寄存器(TIMx_CCRx)来锁存计数器的值。发生捕获事件时，会将相应的 CCXIF 标志（TIMx_SR 寄存器）置 1，并可发送中断或 DMA 请求（如果已使能）。如果发生捕获事件时 CCxIF 标志已处于高位，则会将重复捕获标志 CCxOF（TIMx_SR 寄存器）置 1。可通过软件向 CCxIF 写入 0 来给 CCxIF 清零，或读取存储在 TIMx_CCRx 寄存器中的已捕获数据。向 CCxOF 写入 0 后会将其清零。

以下示例说明了如何在 TI1 输入出现上升沿时将计数器的值捕获到 TIMx_CCR1 中。具体操作步骤如下：

- 选择有效输入：TIMx_CCR1 必须连接到 TI1 输入，因此向 TIMx_CCMR1 寄存器中的 CC1S 位写入 01。只要 CC1S 不等于 00，就会将通道配置为输入模式，并且 TIMx_CCR1 寄存器将处于只读状态。
- 根据连接到定时器的信号，对所需的输入滤波时间进行编程（如果输入为 TIx 输入之一，则对 TIMx_CCMRx 寄存器中的 ICxF 位进行编程）。假设信号变化时，输入信号最多在 5 个内部时钟周期内发生抖动。因此，我们必须将滤波时间设置为大于 5 个内部时钟周期。在检测到 8 个具有新电平的连续采样（以 f_{DTS} 频率采样）后，可以确认 TI1 上的跳变沿。然后向 TIMx_CCMR1 寄存器中的 IC1F 位写入 0011。
- 通过向 TIMx_CCER 寄存器中的 CC1P 位和 CC1NP 位写入 0，选择 TI1 通道的有效转换边沿（本例中为上升沿）。
- 对输入预分频器进行编程。在本例中，我们希望每次有效转换时都执行捕获操作，因此需要禁止预分频器（向 TIMx_CCMR1 寄存器中的 IC1PS 位写入 00）。
- 通过将 TIMx_CCER 寄存器中的 CC1E 位置 1，允许将计数器的值捕获到捕获寄存器中。
- 如果需要，可通过将 TIMx_DIER 寄存器中的 CC1IE 位置 1 来使能相关中断请求，并且/或者通过将该寄存器中的 CC1DE 位置 1 来使能 DMA 请求。

发生输入捕获时：

- 发生有效跳变沿时，TIMx_CCR1 寄存器会获取计数器的值。
- 将 CC1IF 标志置 1（中断标志）。如果至少发生了两次连续捕获，但 CC1IF 标志未被清零，这样CC1OF捕获溢出标志会被置 1。
- 根据 CC1IE 位生成中断。
- 根据 CC1DE 位生成 DMA 请求。

要处理重复捕获，建议在读出捕获溢出标志之前读取数据。这样可避免丢失在读取捕获溢出标志之后与读取数据之前可能出现的重复捕获信息。

注意： *通过软件将 TIMx_EGR 寄存器中的相应 CCxG 位置 1 可生成 IC 中断和/或 DMA 请求。*

15.3.6 PWM 输入模式

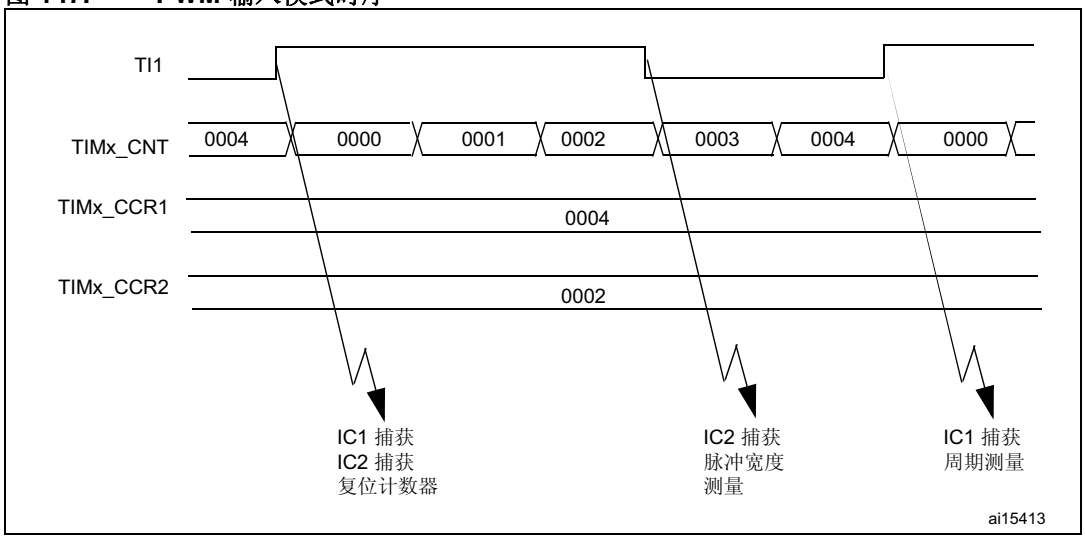
此模式是输入捕获模式的一个特例。其实现步骤与输入捕获模式基本相同，仅存在以下不同之处：

- 两个 ICx 信号被映射至同一个 TIx 输入。
- 这两个 ICx 信号在边沿处有效，但极性相反。
- 选择两个 TIxFP 信号之一作为触发输入，并将从模式控制器配置为复位模式。

例如，可通过以下步骤对应用于 TI1 的 PWM 的周期（位于 TIMx_CCR1 寄存器中）和占空比（位于 TIMx_CCR2 寄存器中）进行测量（取决于 CK_INT 频率和预分频器的值）：

- 选择 TIMx_CCR1 的有效输入：向 TIMx_CCMR1 寄存器中的 CC1S 位写入 01（选择 TI1）。
- 选择 TI1FP1 的有效极性（用于 TIMx_CCR1 中的捕获和计数器清零）：向 CC1P 位和 CC1NP 位写入“0”（上升沿有效）。
- 选择 TIMx_CCR2 的有效输入：向 TIMx_CCMR1 寄存器中的 CC2S 写入 10（选择 TI1）。
- 选择 TI1FP2 的有效极性（用于 TIMx_CCR2 中的捕获）：向 CC2P 位和 CC2NP 位写入“1”（下降沿有效）。
- 选择有效触发输入：向 TIMx_SMCR 寄存器中的 TS 位写入 101（选择 TI1FP1）。
- 将从模式控制器配置为复位模式：向 TIMx_SMCR 寄存器中的 SMS 位写入 100。
- 使能捕获：向 TIMx_CCER 寄存器中的 CC1E 位和 CC2E 位写入“1”。

图 147. PWM 输入模式时序



15.3.7 强制输出模式

在输出模式（TIMx_CCMRx 寄存器中的 CCxS 位 = 00）下，可直接由软件将每个输出比较信号（OCxREF 和 OCx）强制设置为有效电平或无效电平，而无需考虑输出比较寄存器和计数器之间的任何比较结果。

要将输出比较信号 (OCXREF/OCx) 强制设置为有效电平，只需向相应 TIMx_CCMRx 寄存器中的 OCxM 位写入 101。ocxref 进而强制设置为高电平（OCxREF 始终为高电平有效），同时 OCx 获取 CCxP 极性位的相反值。

例如：CCxP=0（OCx 高电平有效）=> OCx 强制设置为高电平。

通过向 TIMx_CCMRx 寄存器中的 OCxM 位写入 100，可将 ocxref 信号强制设置为低电平。

无论如何，TIMx_CCRx 影子寄存器与计数器之间的比较仍会执行，而且允许将标志置 1。因此可发送相应的中断和 DMA 请求。输出比较模式一节对此进行了介绍。

15.3.8 输出比较模式

此功能用于控制输出波形，或指示已经过某一时间段。

当捕获/比较寄存器与计数器之间相匹配时，输出比较功能：

- 将为相应的输出引脚分配一个可编程值，该值由输出比较模式（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。匹配时，输出引脚既可保持其电平 (OCXM=000)，也可设置为有效电平 (OCXM=001)、无效电平 (OCXM=010) 或进行翻转 (OCxM=011)。
- 将中断状态寄存器中的标志置 1（TIMx_SR 寄存器中的 CCxIF 位）。
- 如果相应中断使能位（TIMx_DIER 寄存器中的 CCXIE 位）置 1，将生成中断。
- 如果相应DMA使能位（TIMx_DIER 寄存器的 CCxDE 位，TIMx_CR2 寄存器的 CCDS 位，用来选择 DMA 请求）置 1，将发送 DMA 请求。

使用 TIMx_CCMRx 寄存器中的 OCxPE 位，可将 TIMx_CCRx 寄存器配置为带或不带预装载寄存器。

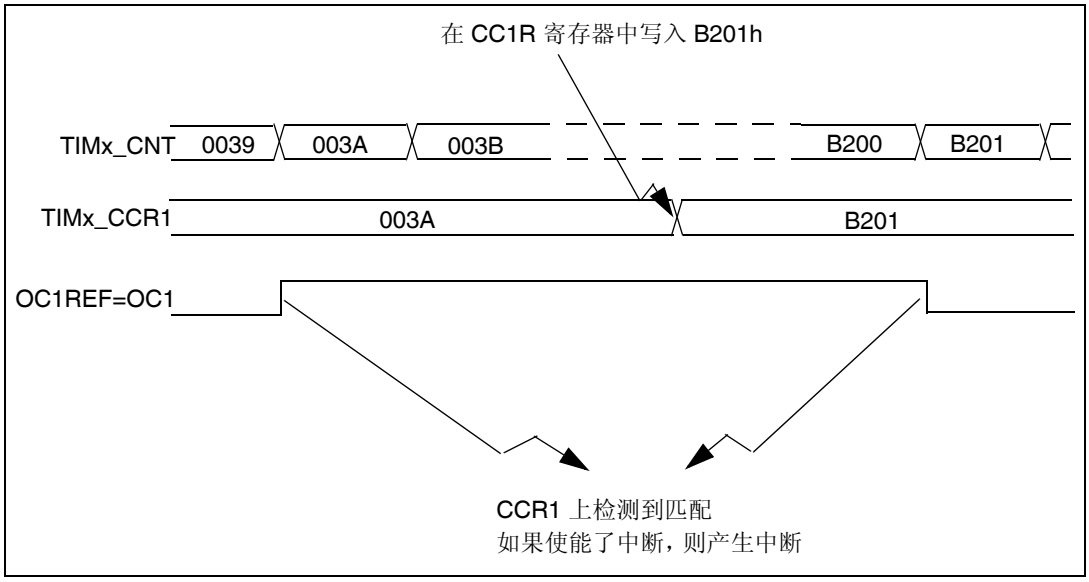
在输出比较模式下，更新事件 UEV 对 ocxref 和 OCx 输出毫无影响。同步的精度可以达到计数器的一个计数周期。输出比较模式也可用于输出单脉冲（在单脉冲模式下）。

步骤：

1. 选择计数器时钟（内部、外部、预分频器）。
2. 在 TIMx_ARR 和 TIMx_CCRx 寄存器中写入所需数据。
3. 如果要生成中断和/或 DMA 请求，将 CCxIE 位和/或 CCxDE 位置 1。
4. 选择输出模式。例如，当 CNT 与 CCRx 匹配、未使用预装载 CCRx 并且 OCx 使能且为高电平有效时，必须写入 OCxM=011、OCxPE=0、CCxP=0 和 CCxE=1 来翻转 OCx 输出引脚。
5. 通过将 TIMx_CR1 寄存器中的 CEN 位置 1 来使能计数器。

可随时通过软件更新 TIMx_CCRx 寄存器以控制输出波形，前提是未使能预装载寄存器（OCxPE=0，否则仅当发生下一个更新事件 UEV 时，才会更新 TIMx_CCRx 影子寄存器）。*图 148* 列出了相关示例。

图 148. 输出比较模式，翻转 OC1



15.3.9 PWM 模式

脉冲宽度调制模式可以生成一个信号，该信号频率由 TIMx_ARR 寄存器值决定，其占空比则由 TIMx_CCRx 寄存器值决定。

通过向 TIMx_CCMRx 寄存器中的 OCxM 位写入 110（PWM 模式 1）或 111（PWM 模式 2），可以独立选择各通道（每个 OCx 输出对应一个 PWM）的 PWM 模式。必须通过将 TIMx_CCMRx 寄存器中的 OCxPE 位置 1 使能相应预装载寄存器，最后通过将 TIMx_CR1 寄存器中的 ARPE 位置 1 使能自动重载预装载寄存器。

由于只有在发生更新事件时预装载寄存器才会传送到影子寄存器，因此启动计数器之前，必须通过将 TIMx_EGR 寄存器中的 UG 位置 1 来初始化所有寄存器。

OCx 极性可使用 TIMx_CCER 寄存器的 CCxP 位来编程。既可以设为高电平有效，也可以设为低电平有效。OCx 输出通过将 TIMx_CCER 寄存器中的 CCxE 位置 1 来使能。有关详细信息，请参见 TIMx_CCERx 寄存器说明。

在 PWM 模式（1 或 2）下，TIMx_CNT 始终与 TIMx_CCRx 进行比较，以确定是 TIMx_CCRx ≤ TIMx_CNT 还是 TIMx_CNT≤ TIMx_CCRx（取决于计数器计数方向）。不过，为了与 ETRF 相符（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），OCREF 信号仅在以下情况下变为有效状态：

- 比较结果发生改变，或
- 输出比较模式（TIMx_CCMRx 寄存器中的 OCxM 位）从“冻结”配置（不进行比较，OCxM=“000”）切换为任一 PWM 模式（OCxM=“110”或“111”）。

定时器运行期间，可以通过软件强制 PWM 输出。

根据 TIMx_CR1 寄存器中的 CMS 位状态，定时器能够产生边沿对齐模式或中心对齐模式的 PWM 信号。

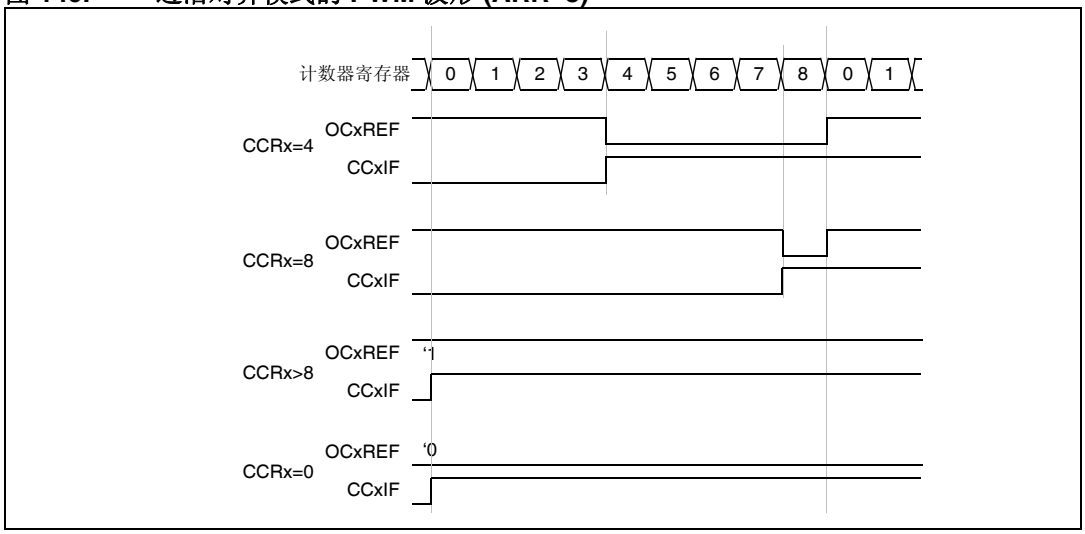
PWM 边沿对齐模式

递增计数配置

当 TIMx_CR1 寄存器中的 DIR 位为低时执行递增计数。请参见*第 395 页的递增计数模式*一节。

以下以 PWM 模式 1 为例。只要 TIMx_CNT < TIMx_CCRx，PWM 参考信号 OCxREF 便为高电平，否则为低电平。如果 TIMx_CCRx 中的比较值大于自动重载值（TIMx_ARR 中），则 OCxREF 保持为“1”。如果比较值为 0，则 OCxREF 保持为“0”。*图 149* 举例介绍边沿对齐模式的一些 PWM 波形 (TIMx_ARR=8)。

图 149. 边沿对齐模式的 PWM 波形 (ARR=8)



递减计数配置

当 TIMx_CR1 寄存器中的 DIR 位为高时执行递减计数。请参见*第 398 页的递减计数模式*一节。

在 PWM 模式 1 下，只要 TIMx_CNT>TIMx_CCRx，参考信号 ocxref 便为低电平，否则为高电平。如果 TIMx_CCRx 中的比较值大于 TIMx_ARR 中的自动重载值，则 ocxref 保持为“1”。此模式下不可能产生 0% 的 PWM 波形。

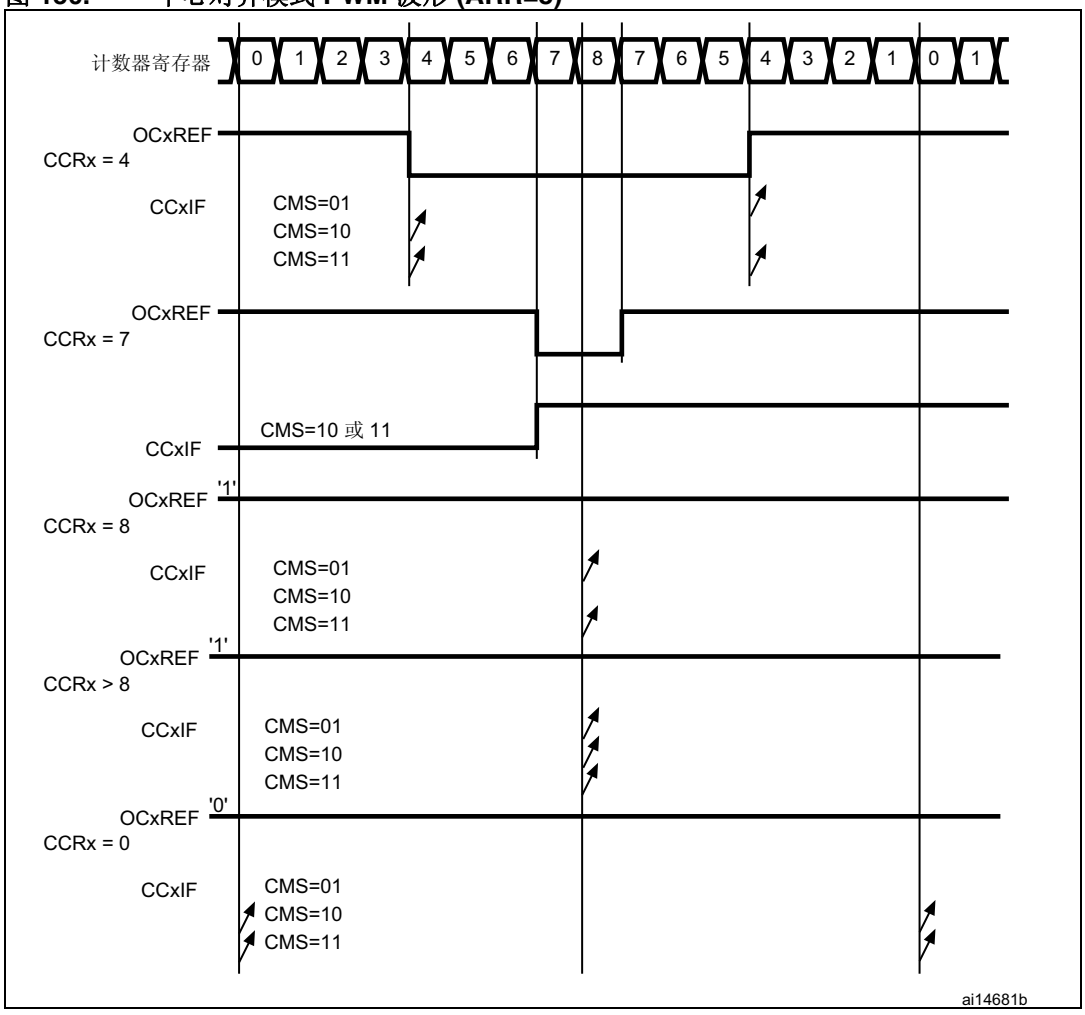
PWM 中心对齐模式

当 TIMx_CR1 寄存器中的 CMS 位不为“00”时（其余所有配置对 ocxref/OCx 信号具有相同的作用），中心对齐模式生效。根据 CMS 位的配置，可以在计数器递增计数、递减计数或同时递增和递减计数时将比较标志置 1。TIMx_CR1 寄存器中的方向位 (DIR) 由硬件更新，不得通过软件更改。请参见*第 400 页的中心对齐模式（递增/递减计数）*一节。

图 150 显示了中心对齐模式的 PWM 波形，在此例中：

- TIMx_ARR=8，
- PWM 模式为 PWM 模式 1，
- 在根据 TIMx_CR1 寄存器中 CMS=01 而选择的中心对齐模式 1 下，当计数器递减计数时，比较标志置 1。

图 150. 中心对齐模式 PWM 波形 (ARR=8)



中心对齐模式使用建议：

- 启动中心对齐模式时将使用当前的递增/递减计数配置。这意味着计数器将根据写入 TIMx_CR1 寄存器中 DIR 位的值进行递增或递减计数。此外，不得同时通过软件修改 DIR 和 CMS 位。

- 不建议在运行中心对齐模式时对计数器执行写操作，否则将发生意想不到的结果。尤其是：
  - 如果写入计数器中的值大于自动重载值 (TIMx_CNT>TIMx_ARR)，计数方向不会更新。例如，如果计数器之前递增计数，则继续递增计数。
  - 如果向计数器写入 0 或 TIMx_ARR 的值，计数方向会更新，但不生成更新事件 UEV。
- 使用中心对齐模式最为保险的方法是：在启动计数器前通过软件生成更新（将 TIMx_EGR 寄存器中的 UG 位置 1），并且不要在计数器运行过程中对其执行写操作。

15.3.10 单脉冲模式

单脉冲模式 (OPM) 是上述模式的一个特例。在这种模式下，计数器可以在一个激励信号的触发下启动，并可在一段可编程的延时后产生一个脉宽可编程的脉冲。

可以通过从模式控制器启动计数器。可以在输出比较模式或 PWM 模式下生成波形。将 TIMx_CR1 寄存器中的 OPM 位置 1，即可选择单脉冲模式。这样，发生下一更新事件 UEV 时，计数器将自动停止。

只有当比较值与计数器初始值不同时，才能正确产生一个脉冲。启动前（定时器等待触发时），必须进行如下配置：

- 递增计数模式下：CNT<CCRx≤ ARR（特别注意，0<CCRx）
- 递减计数模式下：CNT>CCRx

图 151. 单脉冲模式示例

例如，用户希望达到这样的效果：在 TI2 输入引脚检测到正沿时，经过 t_{DELAY} 的延迟，在 OC1 上产生一个长度为 t_{PULSE} 的正脉冲。

使用 TI2FP2 作为触发 1：

- 在 TIMx_CCMR1 寄存器中写入 CC2S=01，将 TI2FP2 映射到 TI2。
- 在 TIMx_CCER 寄存器中写入 CC2P=0 和 CC2NP=“0”，使 TI2FP2 能够检测上升沿。
- 在 TIMx_SMCR 寄存器中写入 TS=110，将 TI2FP2 配置为从模式控制器的触发 (TRGI)。
- 在 TIMx_SMCR 寄存器中写入 SMS=“110”（触发模式），使用 TI2FP2 启动计数器。

OPM 波形通过对比较寄存器执行写操作来定义（考虑时钟频率和计数器预分频器）。

- t_{DELAY} 由写入 TIMx_CCR1 寄存器的值定义。
- t_{PULSE} 由自动重载值与比较值 (TIMx_ARR - TIMx_CCR1) 之差来定义。
- 假设希望产生这样的波形：信号在发生比较匹配时从“0”变为“1”，在计数器达到自动重载值时由“1”变为“0”。为此，应在 TIMx_CCMR1 寄存器中写入 OC1M=111，以使能 PWM 模式 2。如果需要，可选择在 TIMx_CCMR1 寄存器的 OC1PE 和 TIMx_CR1 寄存器的 ARPE 中写入“1”，以使能预装载寄存器。这种情况下，必须在 TIMx_CCR1 寄存器中写入比较值并在 TIMx_ARR 寄存器中写入自动重载值，通过将 UG 位置 1 来产生更新，然后等待 TI2 上的外部触发事件。本例中，CC1P 的值为“0”。

在本例中，TIMx_CR1 寄存器中的 DIR 和 CMS 位应为低。

由于仅需要 1 个脉冲（单脉冲模式），因此应向 TIMx_CR1 寄存器的 OPM 位写入“1”，以便在发生下一更新事件（计数器从自动重载值返回到 0）时使计数器停止计数。TIMx_CR1 寄存器中的 OPM 位置“0”时，即选择重复模式。

特例：OCx 快速使能：

在单脉冲模式下，TIx 输入的边沿检测会将 CEN 位置 1，表示使能计数器。然后，在计数器值与比较值之间发生比较时，将切换输出。但是，完成这些操作需要多个时钟周期，这会限制可能的最小延迟（t_{DELAY} 最小值）。

如果要输出延迟时间最短的波形，可以将 TIMx_CCMRx 寄存器中的 OCxFE 位置 1。这样会强制 OCxRef（和 OCx）对激励信号做出响应，而不再考虑比较的结果。其新电平与发生比较匹配时相同。仅当通道配置为 PWM1 或 PWM2 模式时，OCxFE 才会起作用。

15.3.11 发生外部事件时清除 OCxREF 信号

对于给定通道，在 ETRF 输入施加高电平（相应 TIMx_CCMRx 寄存器中的 OCxCE 使能位置“1”），可使 OCxREF 信号变为低电平。OCxREF 信号将保持低电平，直到发生下一更新事件 (UEV)。

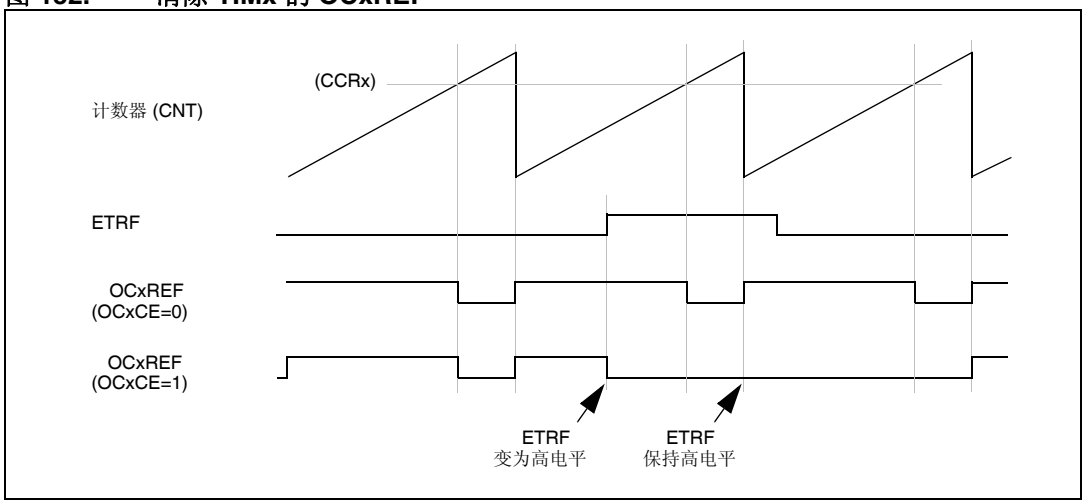
此功能仅能用于输出比较模式和 PWM 模式，而不适用于强制输出模式。

例如，ETR 信号可以连接到比较器的输出，用于控制电流。此时，ETR 必须如下配置：

1. 必须关闭外部触发预分频器：TIMx_SMCR 寄存器中的 ETPS[1:0] 位置 “00”。
2. 必须禁止外部时钟模式 2：TIMx_SMCR 寄存器中的 ECE 位置 “0”。
3. 外部触发极性 (ETP) 和外部触发滤波器 (ETF) 可根据应用需要进行配置。

图 152 对比了使能位OCxCE在不同值下的情况，显示了当 ETRF 输入变为高电平时 OCxREF 信号的行为。在本例中，定时器 TIMx 编程为 PWM 模式。

图 152. 清除 TIMx 的 OCxREF



1. 如果 PWM 的占空比为 100% (CCRx>ARR)，则下次计数器上溢时会再次使能 OCxREF。

15.3.12 编码器接口模式

选择编码器接口模式时，如果计数器仅在 TI2 边沿处计数，在 TIMx_SMCR 寄存器中写入 SMS=001；如果计数器仅在 TI1 边沿处计数，写入 SMS=010；如果计数器在 TI1 和 TI2 边沿处均计数，则写入 SMS=011。

通过编程 TIMx_CCER 寄存器的 CC1P 和 CC2P 位，选择 TI1 和 TI2 极性。如果需要，还可对输入滤波器进行编程。

TI1 和 TI2 两个输入用于连接增量编码器。请参见*表 75*。如果使能计数器（在 TIMx_CR1 寄存器的 CEN 位中写入“1”），则计数器的时钟由 TI1FP1 或 TI2FP2 上的每次有效信号转换提供。TI1FP1 和 TI2FP2 是进行输入滤波器和极性选择后 TI1 和 TI2 的信号，如果不进行滤波和反相，则 TI1FP1=TI1，TI2FP2=TI2。将根据两个输入的信号转换序列，产生计数脉冲和方向信号。根据该信号转换序列，计数器相应递增或递减计数，同时硬件对 TIMx_CR1 寄存器的 DIR 位进行相应修改。任何输入（TI1 或 TI2）发生信号转换时，都会计算 DIR 位，无论计数器是仅在 TI1 或 TI2 边沿处计数，还是同时在 TI1 和 TI2 处计数。

编码器接口模式就相当于带有方向选择的外部时钟。这意味着，计数器仅在 0 到 TIMx_ARR 寄存器中的自动重载值之间进行连续计数（根据具体方向，从 0 递增计数到 ARR，或从 ARR 递减计数到 0）。因此，在启动前必须先配置 TIMx_ARR。同样，捕获、比较、预分频器、触发输出功能继续正常工作。

在此模式下，计数器会根据增量编码器的速度和方向自动进行修改，因此，其内容始终表示编码器的位置。计数方向对应于所连传感器的旋转方向。下表汇总了可能的组合（假设 TI1 和 TI2 不同时切换）。

表 75. 计数方向与编码器信号的关系

有效边沿	相反信号的电平（TI1FP1 对应 TI2，TI2FP2 对应 TI1）	TI1FP1 信号		TI2FP2 信号	
		上升	下降	上升	下降
仅在 TI1 处计数	高	递减	递增	不计数	不计数
	低	递增	递减	不计数	不计数
仅在 TI2 处计数	高	不计数	不计数	递增	递减
	低	不计数	不计数	递减	递增
在 TI1 和 TI2 处均计数	高	递减	递增	递增	递减
	低	递增	递减	递减	递增

外部增量编码器可直接与 MCU 相连，无需外部接口逻辑。不过，通常使用比较器将编码器的差分输出转换为数字信号。这样大幅提高了抗噪声性能。用于指示机械零位的第三个编码器输出可与外部中断输入相连，用以触发计数器复位。

图 153 以计数器工作为例，说明了计数信号的生成和方向控制。同时也说明了选择双边沿时如何对输入抖动进行补偿。将传感器靠近其中一个切换点放置时可能出现这种情况。本例中假设配置如下：

- CC1S=“01”（TIMx_CCMR1 寄存器，TI1FP1 映射到 TI1 上）
- CC2S=“01”（TIMx_CCMR2 寄存器，TI1FP2 映射到 TI2 上）
- CC1P=“0”，CC1NP=“0”，IC1F=“0000”（TIMx_CCER 寄存器，TI1FP1 未反相，TI1FP1=TI1）
- CC2P=“0”，CC2NP=“0”，IC2F=“0000”（TIMx_CCER 寄存器，TI2FP2 未反相，TI1FP2=TI2）
- SMS=“011”（TIMx_SMCR 寄存器，两个输入在上升沿和下降沿均有效）
- CEN = 1（TIMx_CR1 寄存器，使能计数器）

图 153.　编码器接口模式下的计数器工作示例

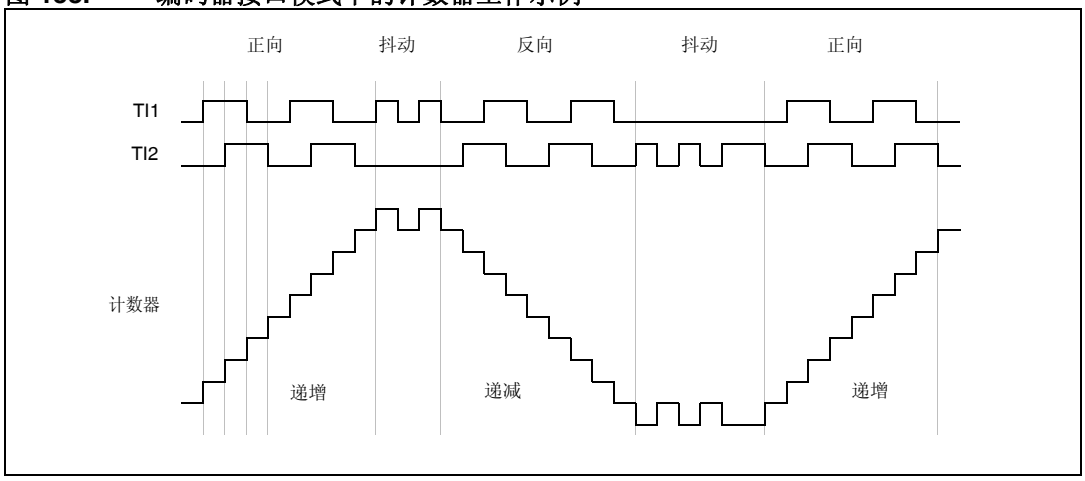
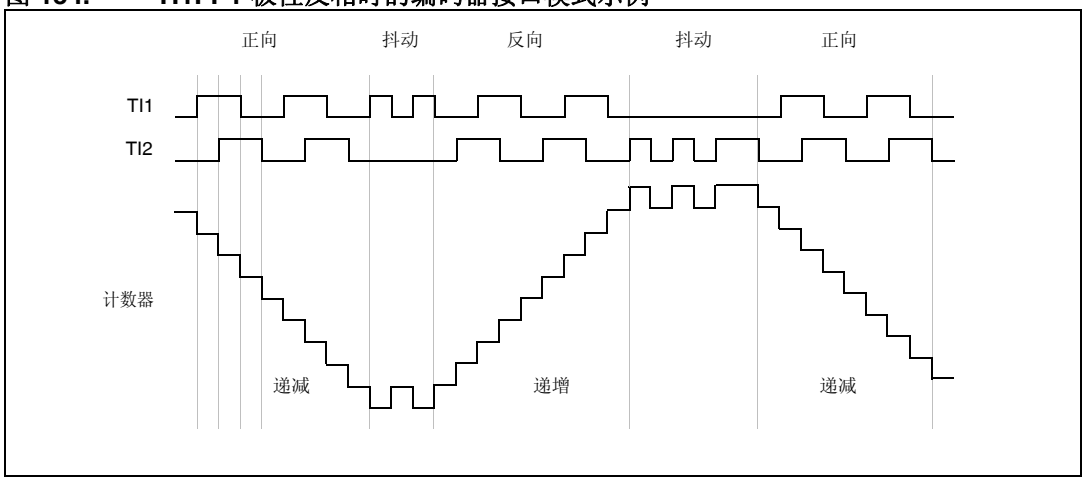


图 154 说明了 TI1FP1 极性反相时的计数器行为示例（除 CC1P=1 外，其它配置与上例相同）。

图 154.　TI1FP1 极性反相时的编码器接口模式示例



定时器配置为编码器接口模式时，会提供传感器当前位置的相关信息。使用另一个配置为捕获模式的定时器测量两个编码器事件之间的周期，可获得动态信息（速度、加速度和减速度）。指示机械零位的编码器输出即可用于此目的。根据两个事件之间的时间间隔，还可定期读取计数器。如果可能，可以将计数器值锁存到第三个输入捕获寄存器来实现此目的（捕获信号必须为周期性信号，可以由另一个定时器产生）；还可以通过由实时时钟生成的 DMA 请求读取计数器值。

15.3.13　定时器输入异或功能

借助 TIMx_CR2 寄存器中的 TI1S 位，可将通道 1 的输入滤波器连接到异或门的输出，从而将 TIMx_CH1 到 TIMx_CH3 这三个输入引脚组合在一起。

异或输出可与触发或输入捕获等所有定时器输入功能配合使用。

15.3.14 定时器与外部触发同步

TIMx 定时器以下列模式与外部触发实现同步：复位模式、门控模式和触发模式。

从模式：复位模式

当触发输入信号发生变化时，计数器及其预分频器可重新初始化。此外，如果 TIMx_CR1 寄存器中的 URS 位处于低电平，则会生成更新事件 UEV。然后，所有预装载寄存器（TIMx_ARR 和 TIMx_CCRx）都将更新。

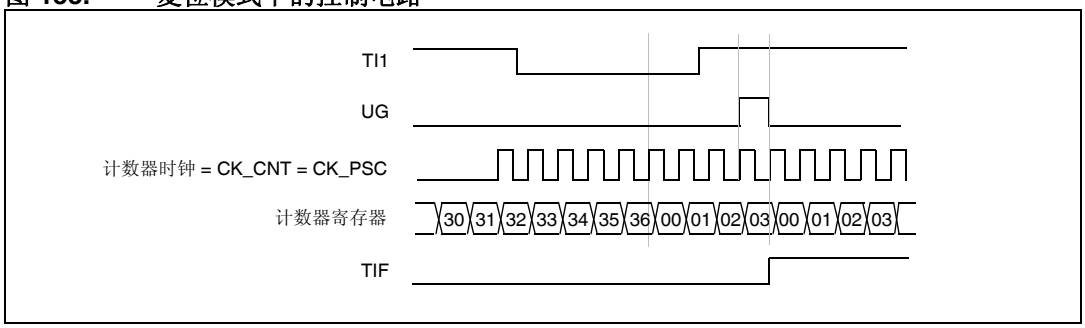
在以下示例中，TI1 输入上出现上升沿时，递增计数器清零：

- 将通道 1 配置为检测 TI1 的上升沿。配置输入滤波时间（本例中不需要任何滤波，因此保持 IC1F=0000）。由于捕获预分频器不用于触发操作，因此无需对其进行配置。CC1S 位只选择输入捕获源，即 TIMx_CCMR1 寄存器中的 CC1S = 01。在 TIMx_CCER 寄存器中写入 CC1P=0 和 CC1NP=0，验证极性（仅检测上升沿）。
- 在 TIMx_SMCR 寄存器中写入 SMS=100，将定时器配置为复位模式。在 TIMx_SMCR 寄存器中写入 TS=101，选择 TI1 作为输入源。
- 在 TIMx_CR1 寄存器中写入 CEN=1，启动计数器。

计数器使用内部时钟计数，然后正常运转，直到出现 TI1 上升沿。当 TI1 出现上升沿时，计数器清零，然后重新从 0 开始计数。同时，触发标志（TIMx_SR 寄存器中的 TIF 位）置 1，使能中断或 DMA 后，还可发送中断或 DMA 请求（取决于 TIMx_DIER 寄存器中的 TIE 和 TDE 位）。

下图显示了自动重载寄存器 TIMx_ARR=0x36 时的相关行为。TI1 的上升沿与实际计数器复位之间的延迟是由于 TI1 输入的重新同步电路引起的。

图 155. 复位模式下的控制电路



从模式：门控模式

输入信号的电平可用来使能计数器。

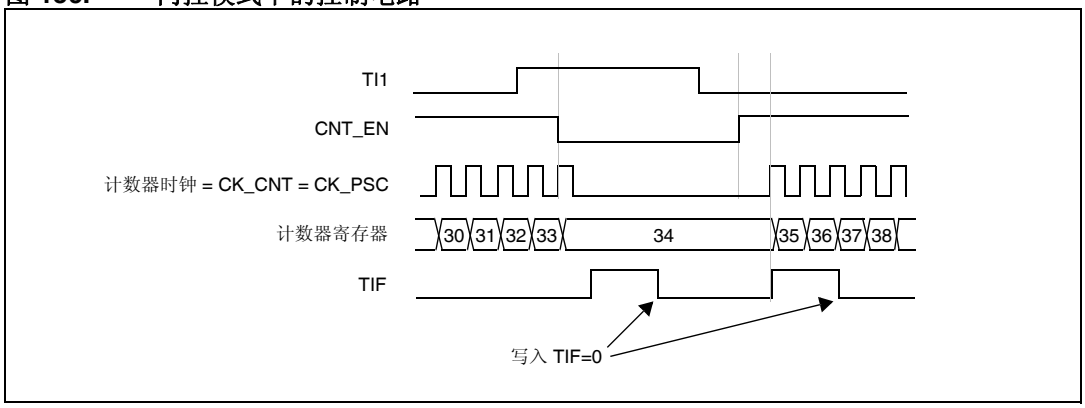
在以下示例中，递增计数器仅在 TI1 输入为低电平时计数：

- 将通道 1 配置为检测 TI1 上的低电平。配置输入滤波时间（本例中不需要任何滤波，因此保持 IC1F=0000）。由于捕获预分频器不用于触发操作，因此无需对其进行配置。CC1S 位只选择输入捕获源，即 TIMx_CCMR1 寄存器中的 CC1S=01。在 TIMx_CCER 寄存器中写入 CC1P=1，以确定极性（仅检测低电平）。
- 在 TIMx_SMCR 寄存器中写入 SMS=101，将定时器配置为门控模式。在 TIMx_SMCR 寄存器中写入 TS=101，选择 TI1 作为输入源。
- 在 TIMx_CR1 寄存器中写入 CEN=1，使能计数器（在门控模式下，如果 CEN=0，则无论触发输入电平如何，计数器都不启动）。

只要 TI1 为低电平，计数器就开始根据内部时钟计数，直到 TI1 变为高电平时停止计数。计数器启动或停止时，TIMx_SR 寄存器中的 TIF 标志都会置 1。

TI1 的上升沿与实际计数器停止之间的延迟是由于 TI1 输入的重新同步电路引起的。

图 156. 门控模式下的控制电路



1. 由于门控模式作用于电平而非边沿，因此在门控模式下，“CCxP=CCxNP=1”（同时检测上升沿和下降沿）不发挥任何作用。

从模式：触发模式

所选输入上发生某一事件时可以启动计数器。

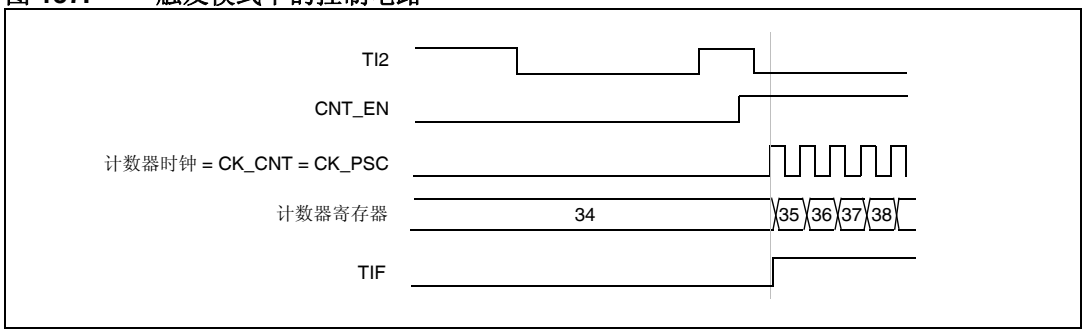
在以下示例中，TI2 输入上出现上升沿时，递增计数器启动：

- 将通道 2 配置为检测 TI2 上的上升沿。配置输入滤波时间（本例中不需要任何滤波，因此保持 IC2F=0000）。由于捕获预分频器不用于触发操作，因此无需对其进行配置。CC2S 位只选择输入捕获源，即 TIMx_CCMR1 寄存器中的 CC2S=01。在 TIMx_CCER 寄存器中写入 CC2P=1，以确定极性（仅检测低电平）。
- 在 TIMx_SMCR 寄存器中写入 SMS=110，将定时器配置为触发模式。在 TIMx_SMCR 寄存器中写入 TS=110，选择 TI2 作为输入源。

当 TI2 出现上升沿时，计数器开始根据内部时钟计数，并且 TIF 标志置 1。

TI2 的上升沿与实际计数器启动之间的延迟是由于 TI2 输入的重新同步电路引起的。

图 157. 触发模式下的控制电路



从模式：外部时钟模式 2 + 触发模式

外部时钟模式 2 可与另一种从模式（外部时钟模式 1 和编码器模式除外）结合使用。这种情况下，ETR 信号用作外部时钟输入，在复位模式、门控模式或触发模式下工作时，可选择另一个输入作为触发输入。不建议通过 TIMx_SMCR 寄存器中的 TS 位来选择 ETR 作为 TRGI。

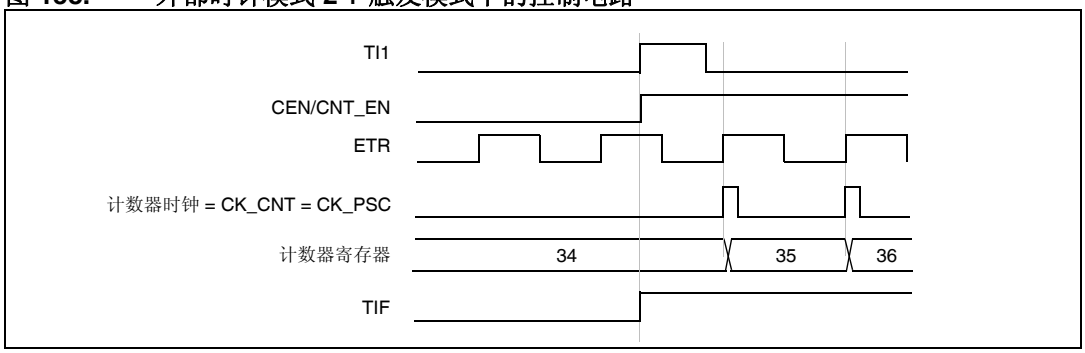
在以下示例中，只要 TI1 出现上升沿，递增计数器即会在 ETR 信号的每个上升沿处递增：

1. 通过对 TIMx_SMCR 寄存器进行如下编程，配置外部触发输入电路：
   - ETF = 0000：无滤波器
   - ETPS = 00：禁止预分频器
   - ETP = 0：检测 ETR 的上升沿，并写入 ECE=1，以使能外部时钟模式 2。
2. 如下配置通道 1，以检测 TI 的上升沿：
   - IC1F = 0000：无滤波器
   - 由于捕获预分频器不用于触发操作，因此无需对其进行配置。
   - TIMx_CCMR1 寄存器中的 CC1S = 01，只选择输入捕获源
   - TIMx_CCER 寄存器中的 CC1P = 0，以确定极性（仅检测上升沿）。
3. 在 TIMx_SMCR 寄存器中写入 SMS=110，将定时器配置为触发模式。在 TIMx_SMCR 寄存器中写入 TS=101，选择 TI1 作为输入源。

TI1 出现上升沿时将使能计数器并且 TIF 标志置 1。然后计数器在 ETR 出现上升沿时计数。

ETR 信号的上升沿与实际计数器复位之间的延迟是由于 ETRP 输入的重新同步电路引起的。

图 158.　外部时钟模式 2 + 触发模式下的控制电路



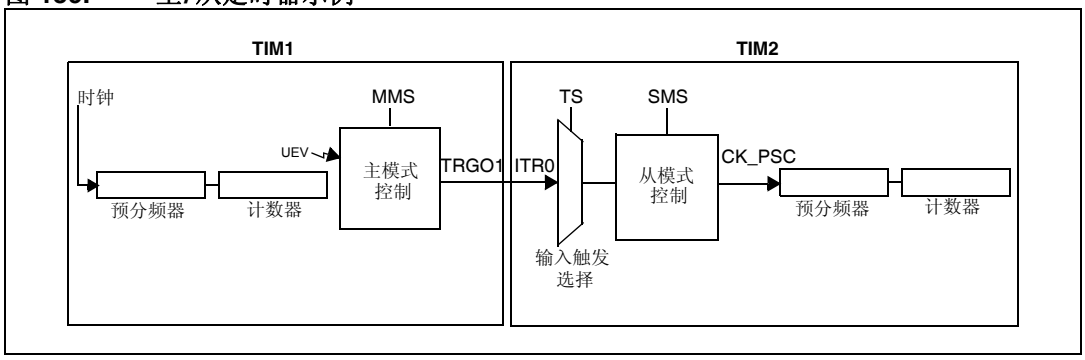
15.3.15　定时器同步

TIMx 定时器从内部连接在一起，以实现定时器同步或级联。当某个定时器配置为主模式时，可对另一个配置为从模式的定时器的计数器执行复位、启动、停止操作或为其提供时钟。

*图 159：主/从定时器示例*简要介绍了触发选择和主模式选择框图。

将一个定时器用作另一个定时器的预分频器

图 159.　主/从定时器示例



例如，可以将定时器 1 配置为定时器 2 的预分频器。请参见*图 159*。为此：

- 将定时器 1 配置为主模式，以便每次发生更新事件 UEV 时都输出一个周期性触发信号。如果在 TIM1_CR2 寄存器中写入 MMS=010，则每次生成更新事件时，TRGO1 都会输出一个上升沿。
- 要将定时器 1 的 TRGO1 输出连接到定时器 2，必须将定时器 2 配置为从模式，使用 ITR0 作为内部触发。通过 TIM2_SMCR 寄存器中的 TS 位（写入 TS=000）可对此进行选择。
- 然后将从模式控制器设为外部时钟模式 1（在 TIM2_SMCR 寄存器中写入 SMS=111）。这样一来，定时器 2 的时钟将由定时器 1 周期性触发信号的上升沿（与定时器 1 的计数器上溢对应）提供。
- 最后必须通过将这两个定时器的相应 CEN 位（TIMx_CR1 寄存器）置 1 同时使能二者。

注意： *如果选择定时器 1 的 OCx 信号作为触发输出 (MMS=1xx)，该信号的上升沿将用于驱动定时器 2 的计数器。*

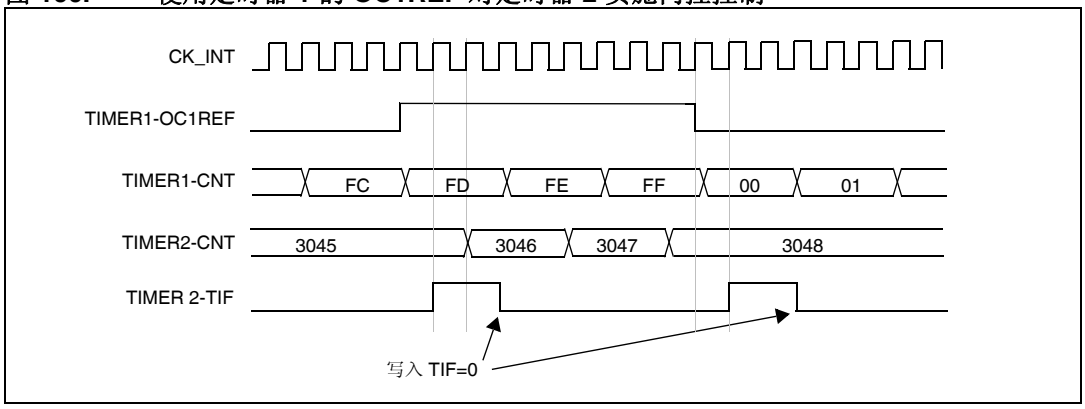
使用一个定时器使能另一个定时器

本例中通过定时器 1 的输出比较 1 来使能定时器 2。相关连接图，请参见*图 159*。仅当定时器 1 的 OC1REF 为高电平时，定时器 2 才根据分频后的内部时钟进行计数。两个计数器的时钟频率都基于 CK_INT 通过预分频器执行 3 分频 ($f_{CK\_CNT} = f_{CK\_INT}/3$)。

- 将定时器 1 配置为主模式，发送其输出比较 1 参考信号 (OC1REF) 作为触发输出（TIM1_CR2 寄存器中的 MMS=100）。
- 配置定时器 1 的 OC1REF 波形（TIM1_CCMR1 寄存器）。
- 配置定时器 2 以接收来自定时器 1 的输入触发（TIM2_SMCR 寄存器中的 TS=000）。
- 将定时器 2 配置为门控模式（TIM2_SMCR 寄存器中的 SMS=101）。
- 通过向 CEN 位（TIM2_CR1 寄存器）写入“1”使能定时器 2。
- 通过向 CEN 位（TIM1_CR1 寄存器）写入“1”启动定时器 1。

注意： *计数器 2 的时钟与计数器 1 不同步，此模式仅影响定时器 2 的计数器使能信号。*

图 160. 使用定时器 1 的 OC1REF 对定时器 2 实施门控控制

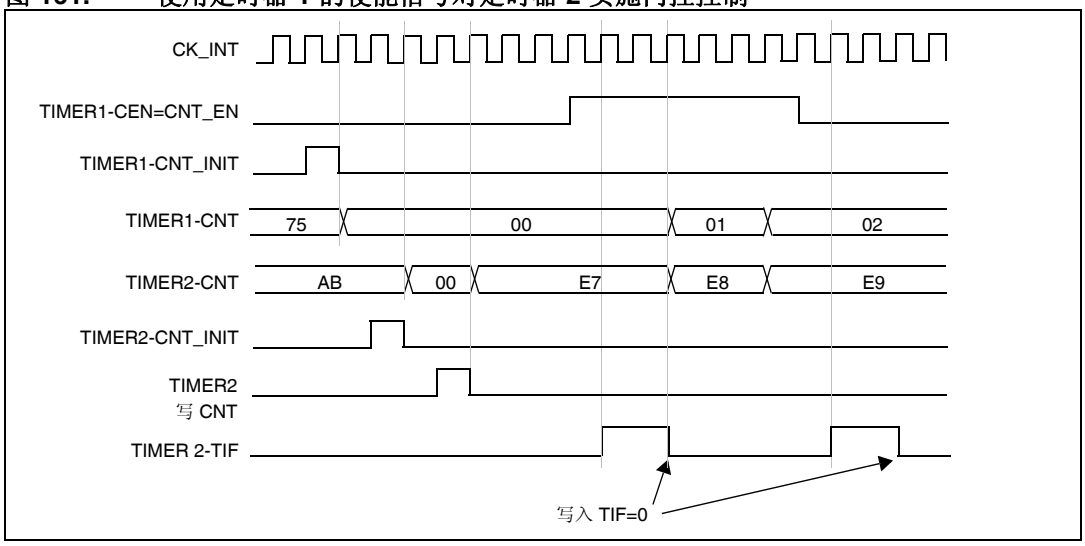


在*图 160* 的示例中，定时器 2 的计数器和预分频器在启动前未进行初始化。因此从各自的当前值开始计数。启动定时器 1 之前，通过复位这两个定时器可以从指定值开始计数。这样便可以在定时器计数器中写入所需的任意值。两个定时器都可通过软件使用 TIMx_EGR 寄存器中的 UG 位轻松复位。

在下一示例中，定时器 1 与定时器 2 同步。定时器 1 为主模式，从 0 开始计数。定时器 2 为从模式，从 0xE7 开始计数。两个定时器的预分频比相同。在 TIM1_CR1 寄存器中通过向 CEN 位写入“0”来禁止定时器 1 时，定时器 2 将停止：

- 将定时器 1 配置为主模式，发送其输出比较 1 参考信号 (OC1REF) 作为触发输出（TIM1_CR2 寄存器中的 MMS=100）。
- 配置定时器 1 的 OC1REF 波形（TIM1_CCMR1 寄存器）。
- 配置定时器 2 以接收来自定时器 1 的输入触发（TIM2_SMCR 寄存器中的 TS=000）。
- 将定时器 2 配置为门控模式（TIM2_SMCR 寄存器中的 SMS=101）。
- 通过向 UG 位（TIM1_EGR 寄存器）写入“1”复位定时器 1。
- 通过向 UG 位（TIM2_EGR 寄存器）写入“1”复位定时器 2。
- 通过在定时器 2 的计数器 (TIM2_CNTL) 中写入“0xE7”使定时器 2 初始化为 0xE7。
- 通过向 CEN 位（TIM2_CR1 寄存器）写入“1”使能定时器 2。
- 通过向 CEN 位（TIM1_CR1 寄存器）写入“1”启动定时器 1。
- 通过向 CEN 位（TIM1_CR1 寄存器）写入“0”停止定时器 1。

图 161. 使用定时器 1 的使能信号对定时器 2 实施门控控制

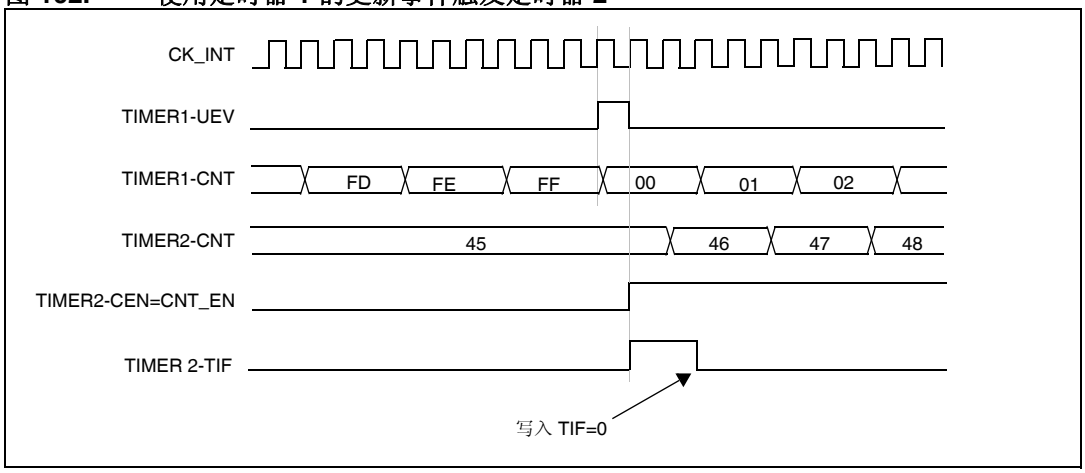


使用一个定时器启动另一个定时器

本例中使用定时器 1 的更新事件使能定时器 2。相关连接图，请参见*图 159*。只要定时器 1 生成更新事件，定时器 2 便根据分频后的内部时钟从当前值（可以不为 0）开始计数。定时器 2 收到触发信号时，其 CEN 位自动置 1，并且计数器开始计数，直到向 TIM2_CR1 寄存器的 CEN 位写入“0”后停止计数。两个计数器的时钟频率都基于 CK_INT 通过预分频器执行 3 分频 ($f_{CK\_CNT} = f_{CK\_INT}/3$)。

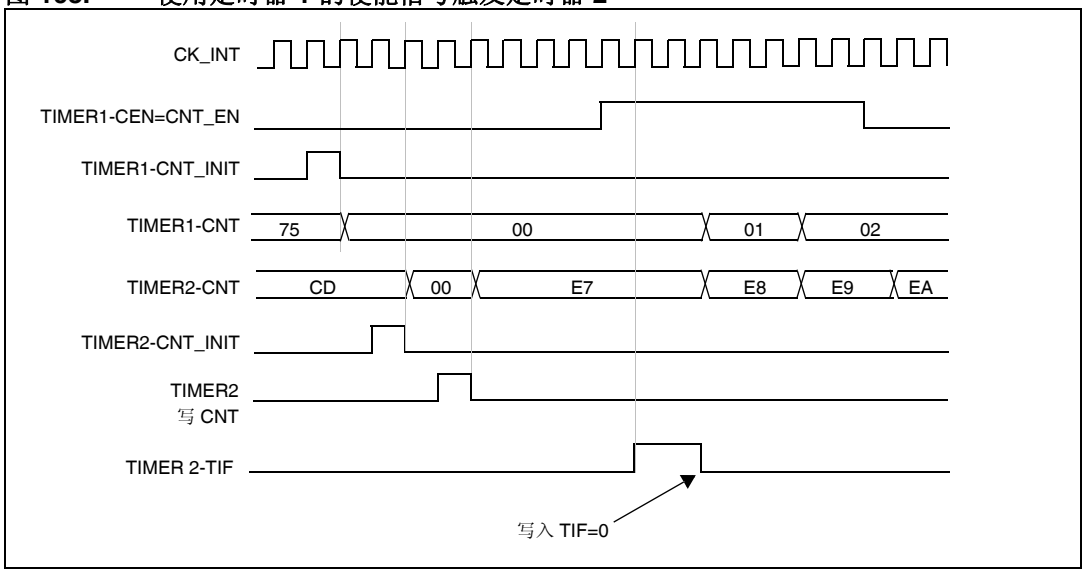
- 将定时器 1 配置为主模式，发送其更新事件 (UEV) 作为触发输出（TIM1_CR2 寄存器中的 MMS=010）。
- 配置定时器 1 的周期（TIM1_ARR 寄存器）。
- 配置定时器 2 以接收来自定时器 1 的输入触发（TIM2_SMCR 寄存器中的 TS=000）。
- 将定时器 2 配置为触发模式（TIM2_SMCR 寄存器中的 SMS=110）。
- 通过向 CEN 位（TIM1_CR1 寄存器）写入“1”启动定时器 1。

图 162.　使用定时器 1 的更新事件触发定时器 2



如上述示例所示，用户可以在开始计数之前初始化两个计数器。*图 163* 显示了与*图 162* 具有相同配置，只不过处于触发模式（TIM2_SMCR 寄存器中的 SMS=110）而非门控模式的计数行为。

图 163.　使用定时器 1 的使能信号触发定时器 2



将一个定时器用作另一个定时器的预分频器

例如，可以将定时器 1 配置为定时器 2 的预分频器。相关连接图，请参见*图 159*。为此：

- 将定时器 1 配置为主模式，发送其更新事件 (UEV) 作为触发输出（TIM1_CR2 寄存器中的 MMS=010）。这样便会在计数器每次发生上溢时输出一个周期性信号。
- 配置定时器 1 的周期（TIM1_ARR 寄存器）。
- 配置定时器 2 以接收来自定时器 1 的输入触发（TIM2_SMCR 寄存器中的 TS=000）。
- 将定时器 2 配置为外部时钟模式（TIM2_SMCR 寄存器中的 SMS=111）。
- 通过向 CEN 位（TIM2_CR1 寄存器）写入“1”启动定时器 2。
- 通过向 CEN 位（TIM1_CR1 寄存器）写入“1”启动定时器 1。

使用一个外部触发同步的启动 2 个定时器

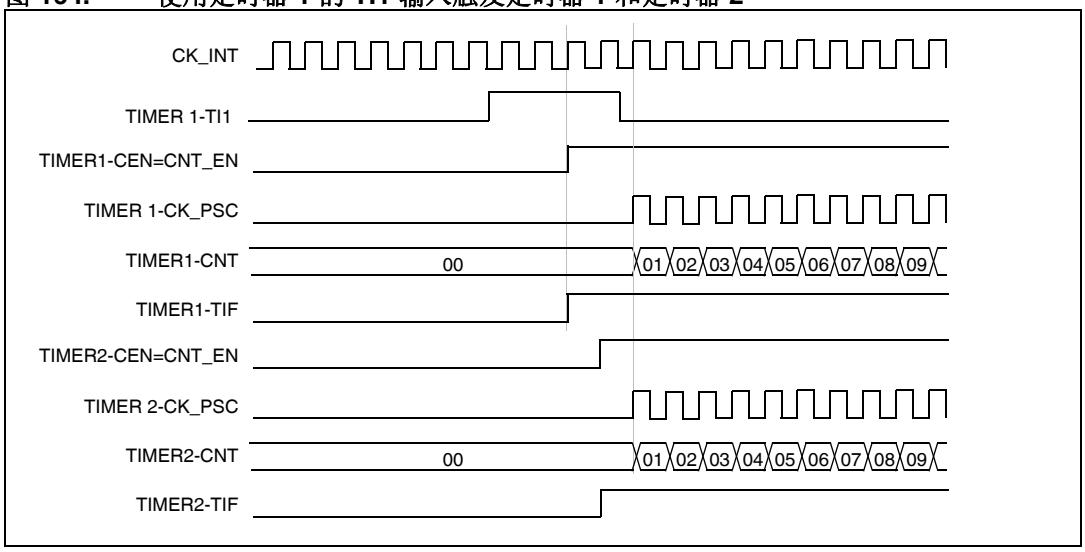
本例中，定时器 1 的 TI1 输入出现上升沿时使能定时器 1，使能定时器 1 的同时使能定时器 2。相关连接图，请参见*图 159*。要确保两个计数器对齐，定时器 1 必须配置为主/从模式（对应的 TI1 为从，对应定时器 2 为主）：

- 将定时器 1 配置为主模式，发送其使能信号作为触发输出（TIM1_CR2 寄存器中的 MMS=001）。
- 将定时器 1 配置为从模式以接收来自 TI1 的输入触发（TIM1_SMCR 寄存器中的 TS=100）。
- 将定时器 1 配置为触发模式（TIM1_SMCR 寄存器中的 SMS=110）。
- 通过写入 MSM=1（TIMx_SMCR 寄存器）将定时器 1 配置为主/从模式。
- 配置定时器 2 以接收来自定时器 1 的输入触发（TIM2_SMCR 寄存器中的 TS=000）。
- 将定时器 2 配置为触发模式（TIM2_SMCR 寄存器中的 SMS=110）。

当 TI1（定时器 1）出现上升沿时，两个计数器开始根据内部时钟同步计数，并且两个 TIF 标志都置 1。

注意： *本例中，两个定时器都在启动之前进行了初始化（通过将各自的 UG 位置 1）。两个计数器都从 0 开始计数，但可以通过对任意一个计数器寄存器 (TIMx_CNT) 进行写操作，在二者之间轻松插入一个偏移量。可注意到主/从模式在定时器 1 的 CNT_EN 与 CK_PSC 之间产生了延迟。*

图 164. 使用定时器 1 的 TI1 输入触发定时器 1 和定时器 2



15.3.16 调试模式

当微控制器进入调试模式时（Cortex™-M4F 内核停止），TIMx 计数器会根据 DBGMCU 模块中的 DBG_TIMx_STOP 配置位选择继续正常工作或者停止工作。有关详细信息，请参见*第 33.16.2 节：对定时器、看门狗、bxCAN 和 I^2C 的调试支持*。

15.4 TIM2 到 TIM5 寄存器

有关寄存器说明中使用的缩写，请参见*第 47 页的第 1.1 节*。

32 位外设寄存器必须按字（32 位）写入数据。所有其它外设寄存器则必须按半字（16 位）或字（32 位）写入数据。而读访问可支持字节（8 位）、半字（16 位）或字（32 位）。

15.4.1 TIMx 控制寄存器 1 (TIMx_CR1)

TIMx control register 1

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
Reserved						CKD[1:0]		ARPE	CMS		DIR	OPM	URS	UDIS	CEN
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 15:10 保留，必须保持复位值。

位 9:8 **CKD**：时钟分频 (Clock division)

此位域指示定时器时钟 (CK_INT) 频率与数字滤波器所使用的采样时钟（ETR、TIx）之间的分频比，

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**：自动重载预装载使能 (Auto-reload preload enable)

0：TIMx_ARR 寄存器不进行缓冲
1：TIMx_ARR 寄存器进行缓冲

位 6:5 **CMS**：中心对齐模式选择 (Center-aligned mode selection)

00：边沿对齐模式。计数器根据方向位 (DIR) 递增计数或递减计数。
01：中心对齐模式 1。计数器交替进行递增计数和递减计数。仅当计数器递减计数时，配置为输出的通道（TIMx_CCMRx 寄存器中的 CxS=00）的输出比较中断标志才置 1。
10：中心对齐模式 2。计数器交替进行递增计数和递减计数。仅当计数器递增计数时，配置为输出的通道（TIMx_CCMRx 寄存器中的 CxS=00）的输出比较中断标志才置 1。
11：中心对齐模式 3。计数器交替进行递增计数和递减计数。当计数器递增计数或递减计数时，配置为输出的通道（TIMx_CCMRx 寄存器中的 CxS=00）的输出比较中断标志都会置 1。

注意：只要计数器处于使能状态 (CEN=1)，就不得从边沿对齐模式切换为中心对齐模式。

位 4 **DIR**：方向 (Direction)

0：计数器递增计数
1：计数器递减计数

注意：当定时器配置为中心对齐模式或编码器模式时，该位为只读状态。

位 3 **OPM**：单脉冲模式 (One-pulse mode)

0：计数器在发生更新事件时不会停止计数
1：计数器在发生下一更新事件时停止计数（将 CEN 位清零）

位 2 **URS**：更新请求源 (Update request source)

此位由软件置 1 和清零，用以选择 UEV 事件源。

0：使能时，所有以下事件都会生成更新中断或 DMA 请求。此类事件包括：

– 计数器上溢/下溢

– 将 UG 位置 1

– 通过从模式控制器生成的更新事件

1：使能时，只有计数器上溢/下溢会生成更新中断或 DMA 请求。

位 1 **UDIS**：更新禁止 (Update disable)

此位由软件置 1 和清零，用以使能/禁止 UEV 事件生成。

0：使能 UEV。更新 (UEV) 事件可通过以下事件之一生成：

– 计数器上溢/下溢

– 将 UG 位置 1

– 通过从模式控制器生成的更新事件

然后缓冲的寄存器将加载预装载值。

1：禁止 UEV。不会生成更新事件，各影子寄存器的值（ARR、PSC 和 CCRx）保持不变。但如果将 UG 位置 1，或者从从模式控制器接收到硬件复位，则会重新初始化计数器和预分频器。

位 0 **CEN**：计数器使能 (Counter enable)

0：禁止计数器

1：使能计数器

注意：只有事先通过软件将 CEN 位置 1，才可以使用外部时钟、门控模式和编码器模式。而触发模式可通过硬件自动将 CEN 位置 1。

在单脉冲模式下，当发生更新事件时会自动将 CEN 位清零。

15.4.2 TIMx 控制寄存器 2 (TIMx_CR2)

TIMx control register 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
Reserved								TI1S	MMS[2:0]			CCDS	Reserved		
								rw	rw	rw	rw	rw			

位 15:8 保留，必须保持复位值。

位 7 **TI1S**：TI1 选择 (TI1 selection)

0：TIMx_CH1 引脚连接到 TI1 输入

1：TIMx_CH1、TIMx_CH2 和 TIMx_CH3 引脚连接到 TI1 输入（异或组合）

位 6:4 **MMS**：主模式选择 (Master mode selection)

这些位可选择主模式下将要发送到从定时器以实现同步的信息 (TRGO)。这些位的组合如下：
000：**复位**——TIMx_EGR 寄存器中的 UG 位用作触发输出 (TRGO)。如果复位由触发输入生成（从模式控制器配置为复位模式），则 TRGO 上的信号相比实际复位会有延迟。
001：**使能**——计数器使能信号 (CNT_EN) 用作触发输出 (TRGO)。该触发输出可用于同时启动多个定时器，或者控制在一段时间内使能从定时器。计数器使能信号可由CEN控制位产生。当配置为门控模式时，也可由触发输入产生。
当计数器使能信号由触发输入控制时，TRGO 上会存在延迟，选择主/从模式时除外（请参见 TIMx_SMCR 寄存器中 MSM 位的说明）。
010：**更新**——选择更新事件作为触发输出 (TRGO)。例如，主定时器可用作从定时器的预分频器。
011：**比较脉冲**——一旦发生输入捕获或比较匹配事件，当 CC1IF 被置 1 时（即使已为高电平），触发输出都会发送一个正脉冲 (TRGO)。(TRGO)
100：**比较**——OC1REF 信号用作触发输出 (TRGO)
101：**比较**——OC2REF 信号用作触发输出 (TRGO)
110：**比较**——OC3REF 信号用作触发输出 (TRGO)
111：**比较**——OC4REF 信号用作触发输出 (TRGO)

位 3 **CCDS**：捕获/比较 DMA 选择 (Capture/compare DMA selection)

0：发生 CCx 事件时发送 CCx DMA 请求
1：发生更新事件时发送 CCx DMA 请求

位 2:0 保留，必须保持复位值。

15.4.3 TIMx 从模式控制寄存器 (TIMx_SMCR)

TIMx slave mode control register

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	rw	rw	rw	rw	rw		rw	rw	rw

位 15 **ETP**：外部触发极性 (External trigger polarity)

此位可选择将 ETR 还是 $\overline{ETR}$ 用于触发操作
0：ETR 未反相，高电平或上升沿有效
1：ETR 反相，低电平或下降沿有效

位 14 **ECE**：外部时钟使能 (External clock enable)

此位可使能外部时钟模式 2。
0：禁止外部时钟模式 2
1：使能外部时钟模式 2。计数器时钟由 ETRF 信号的任意有效边沿提供。
***1：**将 ECE 位置 1 与选择外部时钟模式 1 并将 TRGI 连接到 ETRF（SMS=111 且 TS=111）具有相同效果。*
***2：**外部时钟模式 2 可以和以下从模式同时使用：复位模式、门控模式和触发模式。不过此类情况下 TRGI 不得连接 ETRF （TS 位不得为 111）。*
***3：**如果同时使能外部时钟模式 1 和外部时钟模式 2，则外部时钟输入为 ETRF。*

位 13:12 **ETPS**：外部触发预分频器 (External trigger prescaler)

外部触发信号 ETRP 频率不得超过 CK_INT 频率的 1/4。可通过使能预分频器来降低 ETRP 频率。这种方法在输入快速外部时钟时非常有用。

00：预分频器关闭
01：2 分频 ETRP 频率
10：4 分频 ETRP 频率
11：8 分频 ETRP 频率

位 11:8 **ETF[3:0]**：外部触发滤波器 (External trigger filter)

此位域可定义 ETRP 信号的采样频率和适用于 ETRP 的数字滤波器滤波时间。数字滤波器由事件计数器组成，每 N 个事件才视为一个有效边沿：

0000：无滤波器，按 f_{DTS} 频率进行采样
0001： $f_{SAMPLING}=f_{CK\_INT}$， N=2
0010： $f_{SAMPLING}=f_{CK\_INT}$， N=4
0011： $f_{SAMPLING}=f_{CK\_INT}$， N=8
0100： $f_{SAMPLING}=f_{DTS}/2$， N=6
0101： $f_{SAMPLING}=f_{DTS}/2$， N=8
0110： $f_{SAMPLING}=f_{DTS}/4$， N=6
0111： $f_{SAMPLING}=f_{DTS}/4$， N=8
1000： $f_{SAMPLING}=f_{DTS}/8$， N=6
1001： $f_{SAMPLING}=f_{DTS}/8$， N=8
1010： $f_{SAMPLING}=f_{DTS}/16$， N=5
1011： $f_{SAMPLING}=f_{DTS}/16$， N=6
1100： $f_{SAMPLING}=f_{DTS}/16$， N=8
1101： $f_{SAMPLING}=f_{DTS}/32$， N=5
1110： $f_{SAMPLING}=f_{DTS}/32$， N=6
1111： $f_{SAMPLING}=f_{DTS}/32$， N=8

位 7 **MSM**：主/从模式 (Master/Slave mode)

0：不执行任何操作
1：当前定时器的触发输入事件（TRGI）的动作被推迟，以使当前定时器与其从定时器实现完美同步（通过 TRGO）。此设置适用于单个外部事件对多个定时器进行同步的情况。

位 6:4 **TS**：触发选择 (Trigger selection)

此位域可选择将要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 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)

有关各定时器 ITRx 含义的详细信息，请参见*第 428 页的表 76：TIMx 内部触发连接*。

注意：这些位只能在未使用的情况下（例如，SMS=000 时）进行更改，以避免转换时出现错误的边沿检测。

位 3 保留，必须保持复位值。

位 2:0 **SMS**：从模式选择 (Slave mode selection)

选择外部信号时，触发信号 (TRGI) 的有效边沿与外部输入上所选的极性相关（请参见输入控制寄存器和控制寄存器说明）。
000：禁止从模式——如果 CEN = “1”，预分频器时钟直接由内部时钟提供。
001：编码器模式 1——计数器根据 TI1FP1 电平在 TI2FP2 边沿递增/递减计数。
010：编码器模式 2——计数器根据 TI2FP2 电平在 TI1FP1 边沿递增/递减计数。
011：编码器模式 3——计数器在 TI1FP1 和 TI2FP2 的边沿计数，计数的方向取决于另外一个信号的电平。
100：复位模式——在出现所选触发输入 (TRGI) 上升沿时，重新初始化计数器并生成一个寄存器更新事件。
101：门控模式——触发输入 (TRGI) 为高电平时使能计数器时钟。只要触发输入变为低电平，计数器立即停止计数（但不复位）。计数器的启动和停止都是受控的。
110：触发模式——触发信号 TRGI 出现上升沿时启动计数器（但不复位）。只控制计数器的启动。
111：外部时钟模式 1——由所选触发信号 (TRGI) 的上升沿提供计数器时钟。

注意：如果将 TI1F_ED 选作触发输入 (TS=100)，则不得使用门控模式。实际上，TI1F 每次转换时，TI1F_ED 都输出 1 个脉冲，而门控模式检查的则是触发信号的电平。

表 76. TIMx 内部触发连接

从 TIM	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

15.4.4 TIMx DMA/中断使能寄存器 (TIMx_DIER)

TIMx DMA/Interrupt enable register

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 保留，必须保持复位值。

位 14 **TDE**：触发 DMA 请求使能 (Trigger DMA request enable)

0：禁止触发 DMA 请求。
1：使能触发 DMA 请求。

位 13 保留，始终读为 0

位 12 **CC4DE**：捕获/比较 4 DMA 请求使能 (Capture/Compare 1 DMA request enable)

0：禁止 CC4 DMA 请求。
1：使能 CC4 DMA 请求。

位 11 **CC3DE**：捕获/比较 3 DMA 请求使能 (Capture/Compare 1 DMA request enable)
0：禁止 CC3 DMA 请求。
1：使能 CC3 DMA 请求。

位 10 **CC2DE**：捕获/比较 2 DMA 请求使能 (Capture/Compare 1 DMA request enable)
0：禁止 CC2 DMA 请求。
1：使能 CC2 DMA 请求。

位 9 **CC1DE**：捕获/比较 1 DMA 请求使能 (Capture/Compare 1 DMA request enable)
0：禁止 CC1 DMA 请求。
1：使能 CC1 DMA 请求。

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 保留，必须保持复位值。

位 6 **TIE**：触发信号(TRGI)中断使能 (Trigger interrupt enable)
0：禁止触发信号 (TRGI) 中断。
1：使能触发信号 (TRGI) 中断。

位 5 保留，必须保持复位值。

位 4 **CC4IE**：捕获/比较 4 中断使能 (Capture/Compare 1 interrupt enable)
0：禁止 CC4 中断。
1：使能 CC4 中断。

位 3 **CC3IE**：捕获/比较 3 中断使能 (Capture/Compare 1 interrupt enable)
0：禁止 CC3 中断
1：使能 CC3 中断

位 2 **CC2IE**：捕获/比较 2 中断使能 (Capture/Compare 1 interrupt enable)
0：禁止 CC2 中断
1：使能 CC2 中断

位 1 **CC1IE**：捕获/比较 1 中断使能 (Capture/Compare 1 interrupt enable)
0：禁止 CC1 中断
1：使能 CC1 中断

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

15.4.5 TIMx 状态寄存器 (TIMx_SR)

TIMx status register

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
Reserved			CC4OF	CC3OF	CC2OF	CC1OF	Reserved		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　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 1 overcapture flag)
请参见 CC1OF 说明

位 11 **CC3OF**：捕获/比较 3 重复捕获标志 (Capture/Compare 1 overcapture flag)
请参见 CC1OF 说明

位 10 **CC2OF**：捕获/比较 2 重复捕获标志 (Capture/compare 2 overcapture flag)
请参见 CC1OF 说明

位 9 **CC1OF**：捕获/比较 1 重复捕获标志 (Capture/Compare 1 overcapture flag)
仅当将相应通道配置为输入捕获模式时，此标志位才会由硬件置 1。通过软件写入“0”可将该位清零。
0：未检测到重复捕获
1：TIMx_CCR1 寄存器中已捕获到计数器值且 CC1IF 标志已置 1。

位 8:7　保留，必须保持复位值。

位 6 **TIF**：触发中断标志 (Trigger interrupt flag)
在除门控模式以外的所有模式下，当使能从模式控制器后在 TRGI 输入上检测到有效边沿时，该标志将由硬件置 1。选择门控模式时，该标志将在计数器启动或停止时置 1。但需要通过软件清零。
0：未发生触发信号 (TRGI) 事件
1：触发信号 (TRGI) 中断挂起

位 5　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)
请参见 CC1IF 说明

位 3 **CC3IF**：捕获/比较 3 中断标志 (Capture/Compare 3 interrupt flag)
请参见 CC1IF 说明

位 2 **CC2IF**：捕获/比较 2 中断标志 (Capture/Compare 2 interrupt flag)
请参见 CC1IF 说明

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：
当计数器与比较值匹配时，此标志由硬件置 1，中心对齐模式下除外（请参见 TIMx_CR1 寄存器中的 CMS 位说明）。但需要通过软件清零。
0：不匹配
1：TIMx_CNT 计数器的值与 TIMx_CCR1 寄存器的值匹配。当 TIMx_CCR1 的值大于 TIMx_ARR 的值时，CC1IF 位将在计数器发生上溢（递增计数模式和增减计数模式下）或下溢（递减计数模式下）时变为高电平。
如果通道 CC1 配置为输入：
此位将在发生捕获事件时由硬件置 1。通过软件或读取 TIMx_CCR1 寄存器将该位清零。
0：未发生输入捕获事件
1：TIMx_CCR1 寄存器中已捕获到计数器值（IC1 上已检测到与所选极性匹配的边沿）

位 0 **UIF**：更新中断标志 (Update interrupt flag)

- 该位在发生更新事件时通过硬件置 1。但需要通过软件清零。
  0：未发生更新。
  1：更新中断挂起。该位在以下情况下更新寄存器时由硬件置 1：
- 上溢或下溢（对于 TIM2 到 TIM5）以及当 TIMx_CR1 寄存器中 UDIS = 0 时。
- TIMx_CR1 寄存器中的 URS = 0 且 UDIS = 0，并且由软件使用 TIMx_EGR 寄存器中的 UG 位重新初始化 CNT 时。

TIMx_CR1 寄存器中的 URS=0 且 UDIS=0，并且 CNT 由触发事件重新初始化时（参见同步控制寄存器说明）。

15.4.6 TIMx 事件生成寄存器 (TIMx_EGR)

TIMx event generation register

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
Reserved									TG	Res.	CC4G	CC3G	CC2G	CC1G	UG
									w		w	w	w	w	w

位 15:7 保留，必须保持复位值。

位 6 **TG**：产生触发信号 (Trigger generation)

此位由软件置 1 以生成事件，并由硬件自动清零。
0：不执行任何操作
1：TIMx_SR 寄存器中的 TIF 标志置 1。使能后可发生相关中断或 DMA 传输事件。

位 5 保留，必须保持复位值。

位 4 **CC4G**：捕获/比较 4 生成 (Capture/compare 1 generation)

请参见 CC1G 说明

位 3 **CC3G**：捕获/比较 3 生成 (Capture/compare 1 generation)

请参见 CC1G 说明

位 2 **CC2G**：捕获/比较 2 生成 (Capture/compare 1 generation)

请参见 CC1G 说明

位 1 **CC1G**：捕获/比较 1 生成 (Capture/compare 1 generation)

此位由软件置 1 以生成事件，并由硬件自动清零。
0：不执行任何操作
1：通道 1 上生成捕获/比较事件：
如果通道 CC1 配置为输出：
使能时，CC1IF 标志置 1 并发送相应的中断或 DMA 请求。
如果通道 CC1 配置为输入：
TIMx_CCR1 寄存器中将捕获到计数器当前值。使能时，CC1IF 标志置 1 并发送相应的中断或 DMA 请求。如果 CC1IF 标志已为高电平，CC1OF 标志将置 1。

位 0 **UG**：更新生成 (Update generation)

该位可通过软件置 1，并由硬件自动清零。
0：不执行任何操作
1：重新初始化计数器并生成寄存器更新事件。请注意，预分频器计数器也将清零（但预分频比不受影响）。如果选择中心对齐模式或 DIR=0（递增计数），计数器将清零；如果 DIR=1（递减计数），计数器将使用自动重载值 (TIMx_ARR)。

15.4.7 TIMx 捕获/比较模式寄存器 1 (TIMx_CCMR1)

TIMx capture/compare mode register 1

偏移地址：0x18

复位值：0x0000

这些通道可用于输入（捕获模式）或输出（比较模式）模式。通道方向通过配置相应的 CCxS 位进行定义。此寄存器的所有其它位在输入模式和输出模式下的功能均不同。对于任一给定位，OCxx 用于说明通道配置为输出时该位对应的功能，ICxx 则用于说明通道配置为输入时该位对应的功能。因此，必须注意同一个位在输入阶段和输出阶段具有不同的含义。

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	rw	rw

输出比较模式

位 15 **OC2CE**：输出比较 2 清零使能 (Output compare 3 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)

此位域定义通道方向（输入/输出）以及所使用的输入。
00：CC2 通道配置为输出
01：CC2 通道配置为输入，IC2 映射到 TI2 上
10：CC2 通道配置为输入，IC2 映射到 TI1 上
11：CC2 通道配置为输入，IC2 映射到 TRC 上。此模式仅在通过 TS 位（TIMx_SMCR 寄存器）选择内部触发输入时有效

注意：仅当通道关闭时（TIMx_CCER 中的 CC2E = 0），才可向 CC2S 位写入数据。

位 7 **OC1CE**：输出比较 1 清零使能 (Output compare 3 clear enable)

OC1CE：输出比较 1 清零使能 (Output Compare 1 Clear Enable)
0：OC1Ref 不受 ETRF 输入影响
1：ETRF 输入上检测到高电平时， OC1Ref 立即清零。

位 6:4 **OC1M**：输出比较 1 模式 (Output compare 1 mode)

这些位定义提供 OC1 和 OC1N 的输出参考信号 OC1REF 的行为。OC1REF 为高电平有效，而 OC1 和 OC1N 的有效电平则取决于 CC1P 位和 CC1NP 位。
000：冻结——输出比较寄存器 TIMx_CCR1 与计数器 TIMx_CNT 进行比较不会对输出造成任何影响。（该模式用于生成时基）。
001：将通道 1 设置为匹配时输出有效电平。当计数器 TIMx_CNT 与捕获/比较寄存器 1 (TIMx_CCR1) 匹配时，OC1REF 信号强制变为高电平。
010：将通道 1 设置为匹配时输出无效电平。当计数器 TIMx_CNT 与捕获/比较寄存器 1 (TIMx_CCR1) 匹配时，OC1REF 信号强制变为低电平。
011：翻转——TIMx_CNT=TIMx_CCR1 时，OC1REF 发生翻转。
100：强制变为无效电平——OC1REF 强制变为低电平。
101：强制变为有效电平——OC1REF 强制变为高电平。
110：PWM 模式 1——在递增计数模式下，只要 TIMx_CNT<TIMx_CCR1，通道 1 便为有效状态，否则为无效状态。在递减计数模式下，只要 TIMx_CNT>TIMx_CCR1，通道 1 便为无效状态 (OC1REF=0)，否则为有效状态 (OC1REF=1)。
111：PWM 模式 2——在递增计数模式下，只要 TIMx_CNT<TIMx_CCR1，通道 1 便为无效状态，否则为有效状态。在递减计数模式下，只要 TIMx_CNT>TIMx_CCR1，通道 1 便为有效状态，否则为无效状态。

注意：在 PWM 模式 1 或 PWM 模式 2 下，仅当比较结果发生改变或输出比较模式由“冻结”模式切换到“PWM”模式时，OCREF 电平才会发生更改。

位 3 **OC1PE**：输出比较 1 预装载使能 (Output compare 1 preload enable)

0：禁止与 TIMx_CCR1 相关的预装载寄存器。可随时向 TIMx_CCR1 写入数据，写入后将立即使用新值。
1：使能与 TIMx_CCR1 相关的预装载寄存器。可读/写访问预装载寄存器。TIMx_CCR1 预装载值在每次生成更新事件时都会装载到活动寄存器中。

*注意：**1**：只要编程了 LOCK 级别 3（TIMx_BDTR 寄存器中的 LOCK 位）且 CC1S=00（通道配置为输出），便无法修改这些位。*

***2**：只有单脉冲模式下才可在未验证预装载寄存器的情况下使用 PWM 模式（TIMx_CR1 寄存器中的 OPM 位置 1）。其它情况下则无法保证该行为。*

位 2 **OC1FE**：输出比较 1 快速使能 (Output compare 1 fast enable)

此位用于加快触发输入事件对 CC 输出的影响。
0：即使触发开启，CC1 也将根据计数器和 CCR1 值正常工作。触发输入出现边沿时，激活 CC1 输出的最短延迟时间为 5 个时钟周期。
1：触发输入上出现有效边沿相当于 CC1 输出上的比较匹配。随后，无论比较结果如何，OC 都设置为比较电平。采样触发输入和激活 CC1 输出的延迟时间缩短为 3 个时钟周期。仅当通道配置为 PWM1 或 PWM2 模式时，OCFE 才会起作用。

位 1:0 **CC1S**：捕获/比较 1 选择 (Capture/Compare 1 selection)

此位域定义通道方向（输入/输出）以及所使用的输入。
00：CC1 通道配置为输出。
01：CC1 通道配置为输入，IC1 映射到 TI1 上。
10：CC1 通道配置为输入，IC1 映射到 TI2 上。
11：CC1 通道配置为输入，IC1 映射到 TRC 上。此模式仅在通过 TS 位（TIMx_SMCR 寄存器）选择内部触发输入时有效

注意：仅当通道关闭时（TIMx_CCER 中的 CC1E = 0），才可向 CC1S 位写入数据。

输入捕获模式

位 15:12 **IC2F**：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**：捕获/比较 2 选择 (Capture/compare 2 selection)

此位域定义通道方向（输入/输出）以及所使用的输入。
00：CC2 通道配置为输出。
01：CC2 通道配置为输入，IC2 映射到 TI2 上。
10：CC2 通道配置为输入，IC2 映射到 TI1 上。
11：CC2 通道配置为输入，IC2 映射到 TRC 上。此模式仅在通过 TS 位（TIMx_SMCR 寄存器）选择内部触发输入时有效

注意：仅当通道关闭时（TIMx_CCER 中的 CC2E = 0），才可向 CC2S 位写入数据。

位 7:4 **IC1F**：输入捕获 1 滤波器 (Input capture 1 filter)

此位域可定义 TI1 输入的采样频率和适用于 TI1 的数字滤波器带宽。数字滤波器由事件计数器组成，每 N 个事件才视为一个有效边沿：

0000：无滤波器，按 f_{DTS} 频率进行采样
0001：$f_{SAMPLING}=f_{CK\_INT}$，N=2
0010：$f_{SAMPLING}=f_{CK\_INT}$，N=4
0011：$f_{SAMPLING}=f_{CK\_INT}$，N=8
0100：$f_{SAMPLING}=f_{DTS}/2$，N=6
0101：$f_{SAMPLING}=f_{DTS}/2$，N=8
0110：$f_{SAMPLING}=f_{DTS}/4$，N=6
0111：$f_{SAMPLING}=f_{DTS}/4$，N=8
1000：$f_{SAMPLING}=f_{DTS}/8$，N=6
1001：$f_{SAMPLING}=f_{DTS}/8$，N=8
1010：$f_{SAMPLING}=f_{DTS}/16$，N=5
1011：$f_{SAMPLING}=f_{DTS}/16$，N=6
1100：$f_{SAMPLING}=f_{DTS}/16$，N=8
1101：$f_{SAMPLING}=f_{DTS}/32$，N=5
1110：$f_{SAMPLING}=f_{DTS}/32$，N=6
1111：$f_{SAMPLING}=f_{DTS}/32$，N=8

注意：在当前硅版本中，当 ICxF[3:0]= 1、2 或 3 时，将用 CK_INT 代替公式中的 f_{DTS}。

位 3:2 **IC1PSC**：输入捕获 1 预分频器 (Input capture 1 prescaler)

此位域定义 CC1 输入 (IC1) 的预分频比。
只要 CC1E=0（TIMx_CCER 寄存器），预分频器便立即复位。
00：无预分频器，捕获输入上每检测到一个边沿便执行捕获
01：每发生 2 个事件便执行一次捕获
10：每发生 4 个事件便执行一次捕获
11：每发生 8 个事件便执行一次捕获

位 1:0 **CC1S**：捕获/比较 1 选择 (Capture/Compare 1 selection)

此位域定义通道方向（输入/输出）以及所使用的输入。
00：CC1 通道配置为输出
01：CC1 通道配置为输入，IC1 映射到 TI1 上
10：CC1 通道配置为输入，IC1 映射到 TI2 上
11：CC1 通道配置为输入，IC1 映射到 TRC 上。此模式仅在通过 TS 位（TIMx_SMCR 寄存器）选择内部触发输入时有效

注意：仅当通道关闭时（TIMx_CCER 中的 CC1E = 0），才可向 CC1S 位写入数据。

15.4.8 TIMx 捕获/比较模式寄存器 2 (TIMx_CCMR2)

TIMx capture/compare mode register 2

偏移地址：0x1C

复位值：0x0000

请参见上述 CCMR1 寄存器说明。

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	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**：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**：捕获/比较 4 选择 (Capture/Compare 4 selection)

此位域定义通道方向（输入/输出）以及所使用的输入。
00：CC4 通道配置为输出
01：CC4 通道配置为输入，IC4 映射到 TI4 上
10：CC4 通道配置为输入，IC4 映射到 TI3 上
11：CC4 通道配置为输入，IC4 映射到 TRC 上。此模式仅在通过 TS 位（TIMx_SMCR 寄存器）选择内部触发输入时有效

注意：仅当通道关闭时（TIMx_CCER 中的 CC4E = 0），才可向 CC4S 位写入数据。

位 7 **OC3CE**：输出比较 3 清零使能 (Output compare 3 clear enable)

位 6:4 **OC3M**：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**：捕获/比较 3 选择 (Capture/Compare 3 selection)

此位域定义通道方向（输入/输出）以及所使用的输入。
00：CC3 通道配置为输出
01：CC3 通道配置为输入，IC3 映射到 TI3 上
10：CC3 通道配置为输入，IC3 映射到 TI4 上
11：CC3 通道配置为输入，IC3 映射到 TRC 上。此模式仅在通过 TS 位（TIMx_SMCR 寄存器）选择内部触发输入时有效

注意：仅当通道关闭时（TIMx_CCER 中的 CC3E = 0），才可向 CC3S 位写入数据。

输入捕获模式

位 15:12 **IC4F**：输入捕获 4 滤波器 (Input capture 4 filter)

位 11:10 **IC4PSC**：输入捕获 4 预分频器 (Input capture 4 prescaler)

位 9:8 **CC4S**：捕获/比较 4 选择 (Capture/Compare 4 selection)

此位域定义通道方向（输入/输出）以及所使用的输入。
00：CC4 通道配置为输出
01：CC4 通道配置为输入，IC4 映射到 TI4 上
10：CC4 通道配置为输入，IC4 映射到 TI3 上
11：CC4 通道配置为输入，IC4 映射到 TRC 上。此模式仅在通过 TS 位（TIMx_SMCR 寄存器）选择内部触发输入时有效

注意：仅当通道关闭时（TIMx_CCER 中的 CC4E = 0），才可向 CC4S 位写入数据。

位 7:4 **IC3F**：输入捕获 3 滤波器 (Input capture 3 filter)

位 3:2 **IC3PSC**：输入捕获 3 预分频器 (Input capture 3 prescaler)

位 1:0 **CC3S**：捕获/比较 3 选择 (Capture/Compare 3 selection)

此位域定义通道方向（输入/输出）以及所使用的输入。
00：CC3 通道配置为输出
01：CC3 通道配置为输入，IC3 映射到 TI3 上
10：CC3 通道配置为输入，IC3 映射到 TI4 上
11：CC3 通道配置为输入，IC3 映射到 TRC 上。此模式仅在通过 TS 位（TIMx_SMCR 寄存器）选择内部触发输入时有效

注意：仅当通道关闭时（TIMx_CCER 中的 CC3E = 0），才可向 CC3S 位写入数据。

15.4.9 TIMx 捕获/比较使能寄存器 (TIMx_CCER)

TIMx capture/compare enable register

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 输出极性 (Capture/Compare 1 output Polarity)*。
请参见 CC1NP 说明

位 14 保留，必须保持复位值。

位 13 **CC4P**：*捕获/比较 4 输出极性 (Capture/Compare 1 output Polarity)*。
请参见 CC1P 说明

位 12 **CC4E**：*捕获/比较 4 输出使能 (Capture/Compare 2 output enable)*。
请参见 CC1E 说明

位 11 **CC3NP**：*捕获/比较 3 输出极性 (Capture/Compare 1 output Polarity)*。
请参见 CC1NP 说明

位 10 保留，必须保持复位值。

位 9 **CC3P**：*捕获/比较 3 输出极性 (Capture/Compare 1 output Polarity)*。
请参见 CC1P 说明

位 8 **CC3E**：*捕获/比较 3 输出使能 (Capture/Compare 2 output enable)*。
请参见 CC1E 说明

位 7 **CC2NP**：*捕获/比较 2 输出极性 (Capture/Compare 1 output Polarity)*。
请参见 CC1NP 说明

位 6 保留，必须保持复位值。

位 5 **CC2P**：*捕获/比较 2 输出极性 (Capture/Compare 1 output Polarity)*。
请参见 CC1P 说明

位 4 **CC2E**：*捕获/比较 2 输出使能 (Capture/Compare 2 output enable)*。
请参见 CC1E 说明

位 3 **CC1NP**：*捕获/比较 1 输出极性 (Capture/Compare 1 output Polarity)*。
CC1 通道配置为输出：
在这种情况下， CC1NP 必须保持清零。
CC1 通道配置为输入：
此位与 CC1P 配合使用，用以定义 TI1FP1/TI2FP1 的极性。请参见 CC1P 说明。

位 2 保留，必须保持复位值。

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：
0：OC1 高电平有效
1：OC1 低电平有效
CC1 通道配置为输入：
CC1NP/CC1P 位可针对触发或捕获操作选择 TI1FP1 和 TI2FP1 的极性。
00：非反相/上升沿触发
电路对 TIxFP1 上升沿敏感 （在复位模式、外部时钟模式或触发模式下执行捕获或触发操作）， TIxFP1 未反相 （在门控模式或编码器模式下执行触发操作）。
01：反相/下降沿触发
电路对 TIxFP1 下降沿敏感 （在复位模式、外部时钟模式或触发模式下执行捕获或触发操作）， TIxFP1 反相 （在门控模式或编码器模式下执行触发操作）。
10：保留，不使用此配置。
11：非反相/上升沿和下降沿均触发
电路对 TIxFP1 上升沿和下降沿都敏感（在复位模式、外部时钟模式或触发模式下执行捕获或触发操作），TIxFP1 未反相（在门控模式下执行触发操作）。编码器模式下不得使用此配置。

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 未激活
1：开启——在相应输出引脚上输出 OC1 信号
CC1 通道配置为输入：
此位决定了是否可以实际将计数器值捕获到输入捕获/比较寄存器 1 (TIMx_CCR1) 中。
0：禁止捕获
1：使能捕获

表 77. 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： *与标准 OCx 通道相连的外部 IO 引脚的状态取决于通道 OCx 的状态以及 GPIO 寄存器。*

15.4.10 TIMx 计数器 (TIMx_CNT)

TIMx counter

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	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 15:0 **CNT[15:0]**：计数器值 (Counter value)

15.4.11 TIMx 预分频器 (TIMx_PSC)

TIMx prescaler

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	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 15:0 **PSC[15:0]**：预分频器值 (Prescaler value)

计数器时钟频率 CK_CNT 等于 f_{CK_PSC} / (PSC[15:0] + 1)。

PSC 包含在每次发生更新事件时要装载到实际预分频器寄存器的值。

15.4.12 TIMx 自动重载寄存器 (TIMx_ARR)

TIMx auto-reload register

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	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 15:0 **ARR[15:0]**：自动重载值 (Auto-reload value)

ARR 为要装载到实际自动重载寄存器的值。

有关 ARR 更新和行为的更多详细信息，请参见*第 393 页的第 15.3.1 节：时基单元*。

当自动重载值为空时，计数器不工作。

15.4.13 TIMx 捕获/比较寄存器 1 (TIMx_CCR1)

TIMx capture/compare register 1

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
CCR1[31:16] (depending on timers)															
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
CCR1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:16 **CCR1[31:16]**：捕获/比较 1 的高 16 位（对于 TIM2 和 TIM5）。

位 15:0 **CCR1[15:0]**：捕获/比较 1 的低 16 位 (Low Capture/Compare 1 value)

如果通道 CC1 配置为输出：
CCR1 是捕获/比较寄存器 1 的预装载值。
如果没有通过 TIMx_CCMR 寄存器中的 OC1PE 位来使能预装载功能，写入的数值会被直接传输至当前寄存器中。否则只在发生更新事件时生效（拷贝到实际起作用的捕获/比较寄存器1）。
实际捕获/比较寄存器中包含要与计数器 TIMx_CNT 进行比较并在 OC1 输出上发出信号的值。

如果通道 CC1 配置为输入：
CCR1 为上一个输入捕获 1 事件 (IC1) 发生时的计数器值。

15.4.14 TIMx 捕获/比较寄存器 2 (TIMx_CCR2)

TIMx capture/compare register 2

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
CCR2[31:16] (depending on timers)															
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
CCR2[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:16 **CCR2[31:16]**：捕获/比较 2 的高 16 位（对于 TIM2 和 TIM5）。

位 15:0 **CCR2[15:0]**：捕获/比较 2 的低 16 位 (Low Capture/Compare 2 value)

如果通道 CC2 配置为输出：
CCR2 为要装载到实际捕获/比较 2 寄存器的值（预装载值）。
如果没有通过 TIMx_CCMR 寄存器中的 OC2PE 位来使能预装载功能，写入的数值会被直接传输至当前寄存器中。否则只有发生更新事件时，预装载值才会复制到活动捕获/比较 2 寄存器中。
实际捕获/比较寄存器中包含要与计数器 TIMx_CNT 进行比较并在 OC2 输出上发出信号的值。
如果通道 CC2 配置为输入：
CCR2 为上一个输入捕获 2 事件 (IC2) 发生时的计数器值。

15.4.15 TIMx 捕获/比较寄存器 3 (TIMx_CCR3)

TIMx capture/compare register 3

偏移地址：0x3C

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CCR3[31:16] (depending on timers)															
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
CCR3[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:16 **CCR3[31:16]**：捕获/比较 3 的高 16 位（对于 TIM2 和 TIM5）。

位 15:0 **CCR3[15:0]**：捕获/比较 3 的低 16 位 (Low Capture/Compare value)

如果通道 CC3 配置为输出：

CCR3 为要装载到实际捕获/比较 3 寄存器的值（预装载值）。

如果没有通过 TIMx_CCMR 寄存器中的 OC3PE 位来使能预装载功能，写入的数值会被直接传输至当前寄存器中。否则只有发生更新事件时，预装载值才会复制到活动捕获/比较 3 寄存器中。

实际捕获/比较寄存器中包含要与计数器 TIMx_CNT 进行比较并在 OC3 输出上发出信号的值。

如果通道 CC3 配置为输入：

CCR3 为上一个输入捕获 3 事件 (IC3) 发生时的计数器值。

15.4.16 TIMx 捕获/比较寄存器 4 (TIMx_CCR4)

TIMx capture/compare register 4

偏移地址：0x40

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CCR4[31:16] (depending on timers)															
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
CCR4[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:16 **CCR4[31:16]**：捕获/比较 4 的高 16 位（对于 TIM2 和 TIM5）。

位 15:0 **CCR4[15:0]**：捕获/比较 4 的低 16 位 (Low Capture/Compare value)

1. 如果 CC4 通道配置为输出（CC4S 位）：
   CCR4 为要装载到实际捕获/比较 4 寄存器的值（预装载值）。
   如果没有通过 TIMx_CCMR 寄存器中的 OC4PE 位来使能预装载功能，写入的数值会被直接传输至当前寄存器中。否则只有发生更新事件时，预装载值才会复制到活动捕获/比较 4 寄存器中。
   实际捕获/比较寄存器中包含要与计数器 TIMx_CNT 进行比较并在 OC4 输出上发出信号的值。
2. 如果 CC4 通道配置为输入（TIMx_CCMR4 寄存器中的 CC4S 位）：
   CCR4 为上一个输入捕获 4 事件 (IC4) 发生时的计数器值。

15.4.17 TIMx DMA 控制寄存器 (TIMx_DCR)

TIMx DMA control register

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
Reserved			DBL[4:0]					Reserved			DBA[4:0]				
			rw	rw	rw	rw	rw				rw	rw	rw	rw	rw

位 15:13 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 5 位定义了 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）。
00000：1 次传输，
00001：2 次传输，
00010：3 次传输，
...
10001：18 次传输。

位 7:5 保留，必须保持复位值。

位 4:0 **DBA[4:0]**：DMA 基址 (DMA base address)

该 5 位向量定义 DMA 传输的基址（通过 TIMx_DMAR 地址进行读/写访问时）。DBA 定义为从 TIMx_CR1 寄存器地址开始计算的偏移量。
示例：
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。

15.4.18 TIMx 全传输 DMA 地址 (TIMx_DMAR)

TIMx DMA address for full transfer

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	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 15:0 **DMAB[15:0]**：DMA 连续传送寄存器 (DMA register for burst accesses)

对 DMAR 寄存器执行读或写操作将访问位于如下地址的寄存器

(TIMx_CR1 地址) + (DBA + DMA 索引) x 4

其中 TIMx_CR1 地址为控制寄存器 1 的地址，DBA 为 TIMx_DCR 寄存器中配置的 DMA 基址，DMA 索引由 DMA 传输自动控制，其范围介于 0 到 DBL（TIMx_DCR 寄存器中配置的 DBL）之间。

DMA 连续传送功能使用方法示例

本例中，定时器 DMA 连续传送功能用于将 CCRx 寄存器（x = 2、3、4）的内容更新为通过 DMA 传输到 CCRx 寄存器中的多个半字。

具体操作步骤如下：

1. 将相应的 DMA 通道配置如下：
   - DMA 通道外设地址为 DMAR 寄存器地址
   - DMA 通道存储器地址为包含要通过 DMA 传输到 CCRx 寄存器的数据的 RAM 缓冲区地址。
   - 要传输的数据量 = 3（参见下文注释）。
   - 禁止循环模式。
2. 通过将 DBA 和 DBL 位域配置如下来配置 DCR 寄存器：
   DBL = 3 次传输，DBA = 0xE。
3. 使能 TIMx 更新 DMA 请求（DIER 寄存器中的 UDE 位置 1）。
4. 使能 TIMx
5. 使能 DMA 通道

注意： *本例适用于每个 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。数据将按照如下方式传输到 CCRx 寄存器：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；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。*

15.4.19 TIM2 选项寄存器 (TIM2_OR)

TIM2 option register

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
Reserved				ITR1_RMP		Reserved									
				rw	rw										

位 15:12 保留，必须保持复位值。

位 11:10 **ITR1_RMP**：内部触发 1 重映射 (Internal trigger 1 remap)
由软件置 1 和清零。
00：TIM8_TRGOUT
01：PTP 触发输出连接到 TIM2_ITR1
10：OTG FS SOF 连接到 TIM2_ITR1 输入
11：OTG HS SOF 连接到 TIM2_ITR1 输入

位 9:0 保留，必须保持复位值。

15.4.20 TIM5 选项寄存器 (TIM5_OR)

TIM5 option register

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
Reserved								TI4_RMP		Reserved					
								rw	rw						

位 15:8 保留，必须保持复位值。

位 7:6 **TI4_RMP**：定时器输入 4 重映射 (Timer Input 4 remap)

由软件置 1 和清零。

00：TIM5 的通道 4 连接到 GPIO：请参见数据手册中的复用功能映射表。

01：为进行校准，LSI 内部时钟连接到 TIM5_CH4 输入。

10：为进行校准，LSE 内部时钟连接到 TIM5_CH4 输入。

11：为进行校准，RTC 唤醒中断连接到 TIM5_CH4 输入。应使能唤醒中断。

位 5:0 保留，必须保持复位值。

15.4.21 TIMx 寄存器映射

TIMx 寄存器按照下表所述方式映射：

表 78. TIM2 到 TIM5 寄存器映射和复位值

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**TIMx_CR1**	Reserved																						CKD [1:0]		ARPE	CMS [1:0]		DIR	OPM	URS	UDIS	CEN
	Reset value																							0	0	0	0	0	0	0	0	0	0
0x04	**TIMx_CR2**	Reserved																								TI1S	MMS[2:0]			CCDS	Reserved		
	Reset value																									0	0	0	0	0			
0x08	**TIMx_SMCR**	Reserved																ETP	ECE	ETPS [1:0]		ETF[3:0]				MSM	TS[2:0]			Reserved	SMS[2:0]		
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0
0x0C	**TIMx_DIER**	Reserved																	TDE	COMDE	CC4DE	CC3DE	CC2DE	CC1DE	UDE	Reserved	TIE	Reserved	CC4IE	CC3IE	CC2IE	CC1IE	UIE
	Reset value																		0	0	0	0	0	0	0		0		0	0	0	0	0
0x10	**TIMx_SR**	Reserved																			CC4OF	CC3OF	CC2OF	CC1OF	Reserved		TIF	Reserved	CC4IF	CC3IF	CC2IF	CC1IF	UIF
	Reset value																				0	0	0	0			0		0	0	0	0	0
0x14	**TIMx_EGR**	Reserved																								TG	Reserved		CC4G	CC3G	CC2G	CC1G	UG
	Reset value																									0			0	0	0	0	0
0x18	**TIMx_CCMR1** *Output Compare mode*	Reserved																OC2CE	OC2M [2:0]			OC2PE	OC2FE	CC2S [1:0]		OC1CE	OC1M [2:0]			OC1PE	OC1FE	CC1S [1:0]	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	TIMx_CCMR1 *Input Capture mode*	Reserved																IC2F[3:0]				IC2 PSC [1:0]		CC2S [1:0]		IC1F[3:0]				IC1 PSC [1:0]		CC1S [1:0]	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

表 78. TIM2 到 TIM5 寄存器映射和复位值（续）

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x1C	**TIMx_CCMR2** *Output Compare mode*	Reserved																O24CE	OC4M [2:0]			OC4PE	OC4FE	CC4S [1:0]		OC3CE	OC3M [2:0]			OC3PE	OC3FE	CC3S [1:0]	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	TIMx_CCMR2 *Input Capture mode*	Reserved																IC4F[3:0]				IC4 PSC [1:0]		CC4S [1:0]		IC3F[3:0]				IC3 PSC [1:0]		CC3S [1:0]	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20	**TIMx_CCER**	Reserved																CC4NP	Reserved	CC4P	CC4E	CC3NP	Reserved	CC3P	CC3E	CC2NP	Reserved	CC2P	CC2E	CC1NP	Reserved	CC1P	CC1E
	Reset value																	0		0	0	0		0	0	0		0	0	0		0	0
0x24	**TIMx_CNT**	CNT[31:16] (TIM2 and TIM5 only, reserved on the other timers)																CNT[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x28	**TIMx_PSC**	Reserved																PSC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x2C	**TIMx_ARR**	ARR[31:16] (TIM2 and TIM5 only, reserved on the other timers)																ARR[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x30	Reserved																																
0x34	**TIMx_CCR1**	CCR1[31:16] (TIM2 and TIM5 only, reserved on the other timers)																CCR1[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x38	**TIMx_CCR2**	CCR2[31:16] (TIM2 and TIM5 only, reserved on the other timers)																CCR2[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x3C	**TIMx_CCR3**	CCR3[31:16] (TIM2 and TIM5 only, reserved on the other timers)																CCR3[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x40	**TIMx_CCR4**	CCR4[31:16] (TIM2 and TIM5 only, reserved on the other timers)																CCR4[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x44	Reserved																																
0x48	**TIMx_DCR**	Reserved																			DBL[4:0]			Reserved		DBA[4:0]							
	Reset value																				0	0	0	0	0	0	0	0	0	0	0	0	0
0x4C	**TIMx_DMAR**	Reserved																DMAB[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x50	TIM2_OR	Reserved																Reserved				ITR1_ RMP		Reserved									
	Reset value																					0	0										
0x50	TIM5_OR	Reserved																Reserved								IT4_R MP		Reserved					
	Reset value																									0	0						

有关寄存器边界地址的信息，请参见*第 52 页的表 2*。

16 通用定时器（TIM9 到 TIM14）

除非特别说明，否则本部分适用于整个 STM32F4xx 系列。

16.1 TIM9 到 TIM14 简介

TIM9 到 TIM14 通用定时器包含一个 16 位自动重载计数器，该计数器由可编程预分频器驱动。

它们可用于多种用途，包括测量输入信号的脉冲宽度（输入捕获），或者生成输出波形（输出比较、PWM）。

使用定时器预分频器和 RCC 时钟控制器预分频器，可将脉冲宽度和波形周期从几微秒调制到几毫秒。

TIM9 到 TIM14 定时器彼此完全独立，不共享任何资源。如*第 16.4.12 节*中所述，它们可以同步操作。

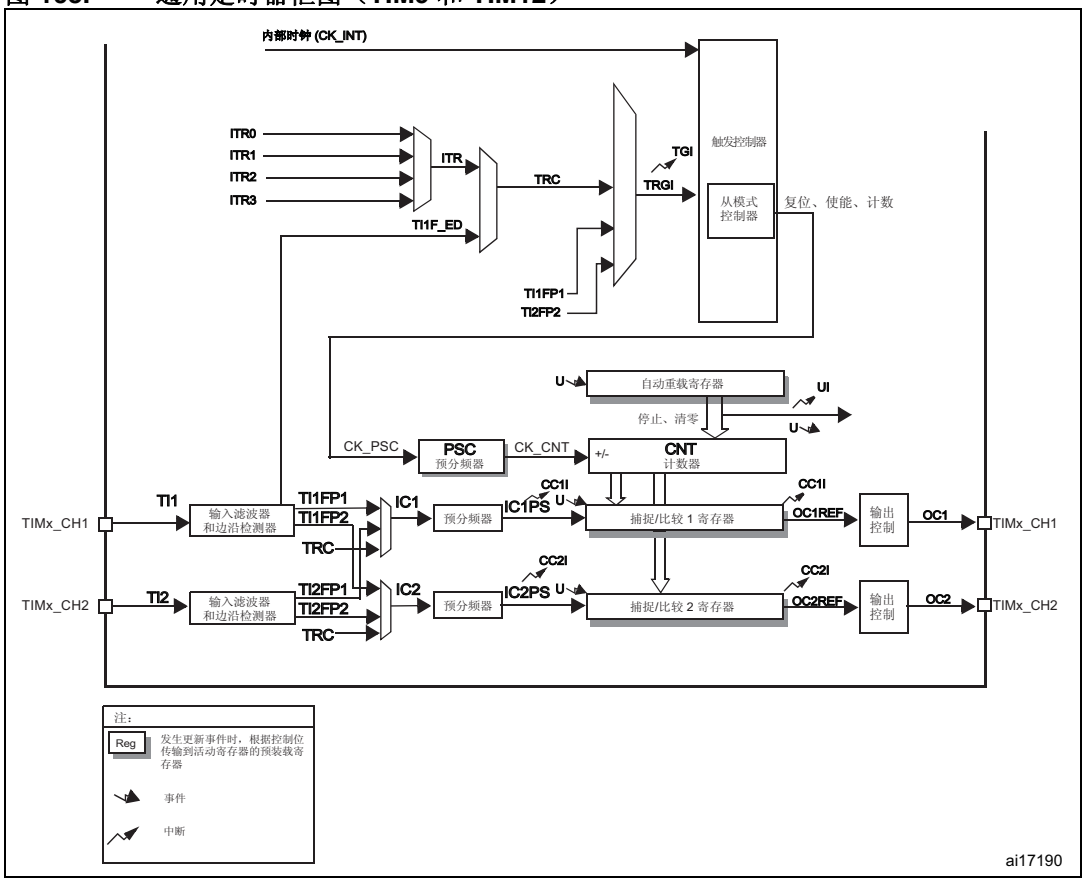
16.2 TIM9 到 TIM14 主要特性

16.2.1 TIM9/TIM12 主要特性

TIM9 到 TIM14 通用定时器具有以下特性：

- 16 位自动重载递增计数器（属于中等容量器件）
- 16 位可编程预分频器，用于对计数器时钟频率进行分频（即运行时修改），分频系数介于 1 和 65536 之间
- 多达 2 个独立通道，可用于：
  - 输入捕获
  - 输出比较
  - PWM 生成（边沿对齐模式）
  - 单脉冲模式输出
- 使用外部信号控制定时器且可实现多个定时器互连的同步电路
- 发生如下事件时生成中断：
  - 更新：计数器上溢、计数器初始化（通过软件或内部触发）
  - 触发事件（计数器启动、停止、初始化或者由内部触发计数）
  - 输入捕获
  - 输出比较

图 165. 通用定时器框图（TIM9 和 TIM12）

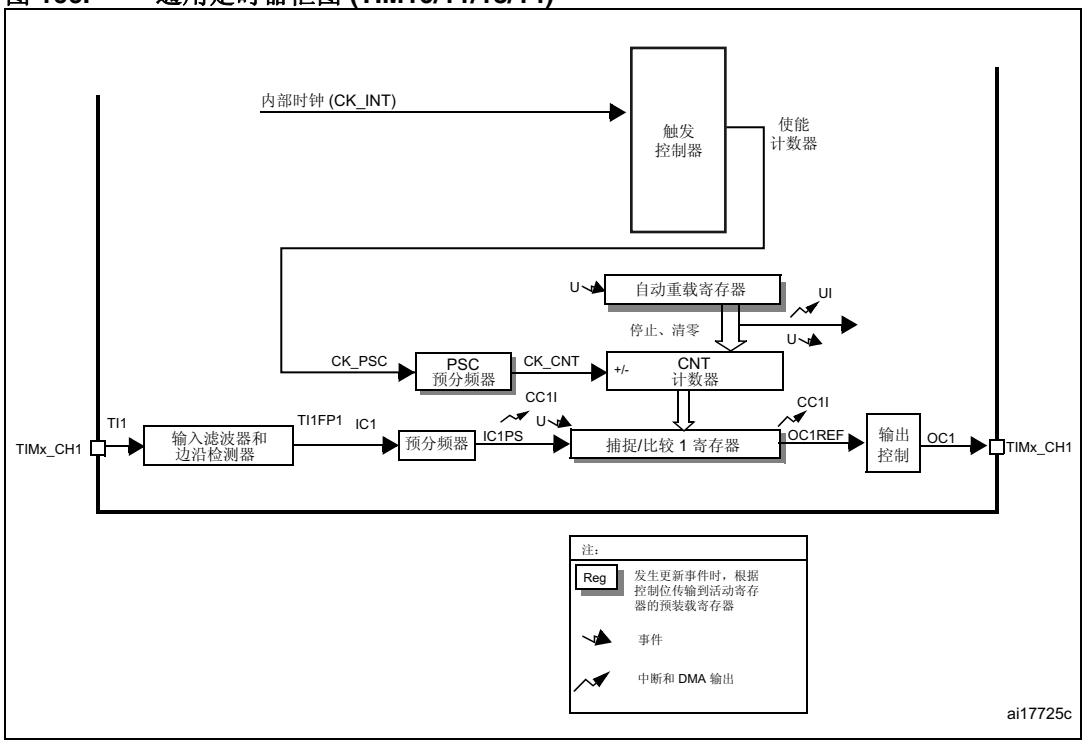


16.3 TIM10/TIM11 和 TIM13/TIM14 主要特性

通用定时器 TIM10/TIM11 和 TIM13/TIM14 具有以下特性：

- 16 位自动重载递增计数器
- 16 位可编程预分频器，用于对计数器时钟频率进行分频（即运行时修改），分频系数介于 1 和 65536 之间
- 独立通道，可用于：
  - 输入捕获
  - 输出比较
  - PWM 生成（边沿对齐模式）
  - 单脉冲模式输出
- 发生如下事件时生成中断：
  - 更新：计数器上溢、计数器初始化（通过软件）
  - 输入捕获
  - 输出比较

图 166. 通用定时器框图 (TIM10/11/13/14)



16.4 TIM9 到 TIM14 功能说明

16.4.1 时基单元

定时器的主要模块由一个 16 位计数器及其相关的自动重载寄存器组成。计数器采用递增方式计数。

计数器的时钟可通过预分频器进行分频。

计数器、自动重载寄存器和预分频器寄存器可通过软件进行读写。即使在计数器运行时也可执行读写操作。

时基单元包括：

- 计数器寄存器 (TIMx_CNT)
- 预分频器寄存器 (TIMx_PSC)
- 自动重载寄存器 (TIMx_ARR)

自动重载寄存器是预装载的。对自动重载寄存器执行写入或读取操作时会访问预装载寄存器。预装载寄存器的内容既可以直接传送到影子寄存器，也可以在每次发生更新事件 (UEV) 时传送到影子寄存器，这取决于 TIMx_CR1 寄存器中的自动重载预装载使能位 (ARPE)。当计数器达到上溢值并且 TIMx_CR1 寄存器中的 UDIS 位为 0 时，将发送更新事件。该更新事件也可由软件产生。下文将针对各配置的更新事件的产生进行详细介绍。

计数器由预分频器输出 CK_CNT 提供时钟，仅当 TIMx_CR1 寄存器中的计数器启动位 (CEN) 置 1 时，才会启动计数器（有关计数器使能的更多详细信息，另请参见从模式控制器的相关说明）。

注意，计数器将在 TIMx_CR1 寄存器的 CEN 位置 1 时刻的一个时钟周期后开始计数。

预分频器说明

预分频器可对计数器时钟频率进行分频，分频系数介于 1 和 65536 之间。该预分频器基于 TIMx_PSC 寄存器中的 16 位寄存器所控制的 16 位计数器。由于该控制寄存器具有缓冲功能，因此预分频器可实现实时更改。而新的预分频比将在下一更新事件发生时被采用。

图 167 和*图 168* 以一些示例说明在预分频比实时变化时计数器的行为：

图 167. 预分频器分频由 1 变为 2 时的计数器时序图

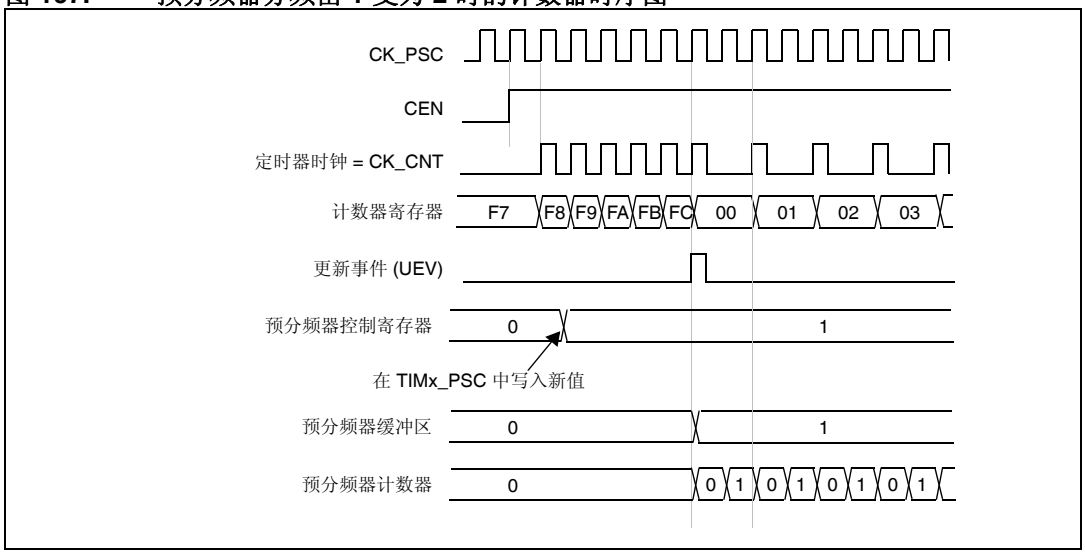
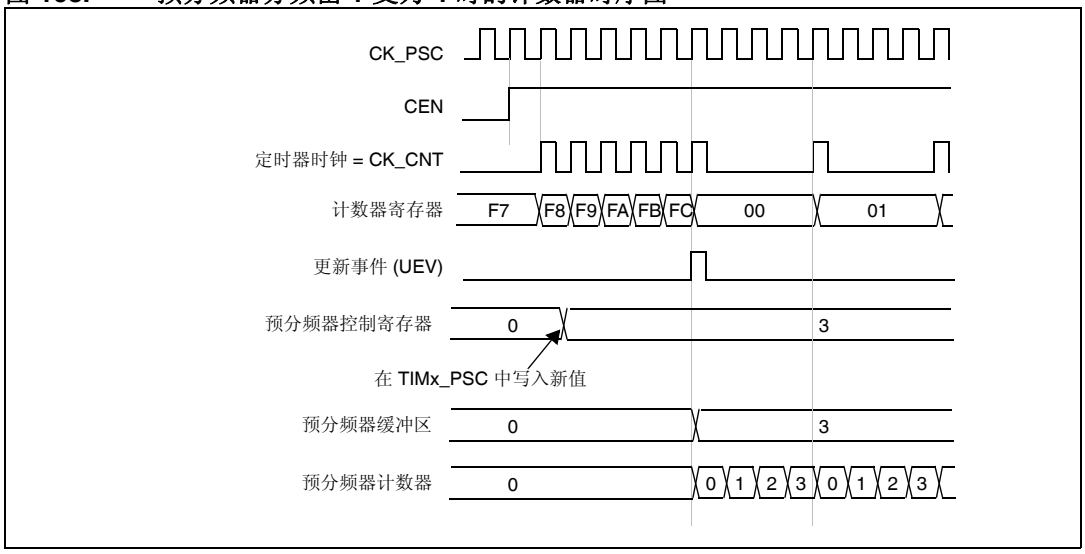


图 168. 预分频器分频由 1 变为 4 时的计数器时序图



16.4.2 计数器模式

递增计数模式

在递增计数模式下，计数器从 0 计数到自动重载值（TIMx_ARR 寄存器的内容），然后重新从 0 开始计数并生成计数器上溢事件。

将 TIMx_EGR 寄存器中的 UG 位置 1（通过软件或使用 TIM9 和 TIM12 上的从模式控制器）也会生成更新事件。

通过软件将 TIMx_CR1 寄存器中的 UDIS 位置 1 可禁止 UEV 事件。这可避免向预装载寄存器写入新值时更新影子寄存器。在 UDIS 位写入 0 之前不会产生任何更新事件。不过，计数器和预分频器计数器都会重新从 0 开始计数（而预分频比保持不变）。此外，如果 TIMx_CR1 寄存器中的 URS 位（更新请求选择）已置 1，则将 UG 位置 1 会生成更新事件 UEV，但不会将 UIF 标志置 1（因此，不会发送任何中断）。这样一来，如果在发生捕获事件时将计数器清零，将不会同时产生更新中断和捕获中断。

发生更新事件时，将更新所有寄存器且将更新标志（TIMx_SR 寄存器中的 UIF 位）置 1（取决于 URS 位）：

- 自动重载影子寄存器将以预装载值 (TIMx_ARR) 进行更新
- 预分频器的缓冲区中将重新装载预装载值（TIMx_PSC 寄存器的内容）

以下各图以一些示例说明当 TIMx_ARR=0x36 时不同时钟频率下计数器的行为。

图 169. 计数器时序图，1 分频内部时钟

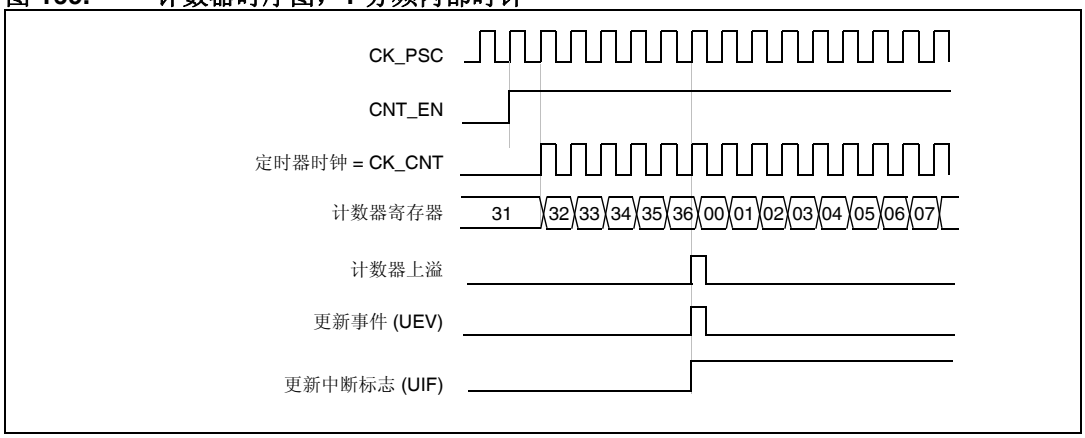


图 170. 计数器时序图，2 分频内部时钟

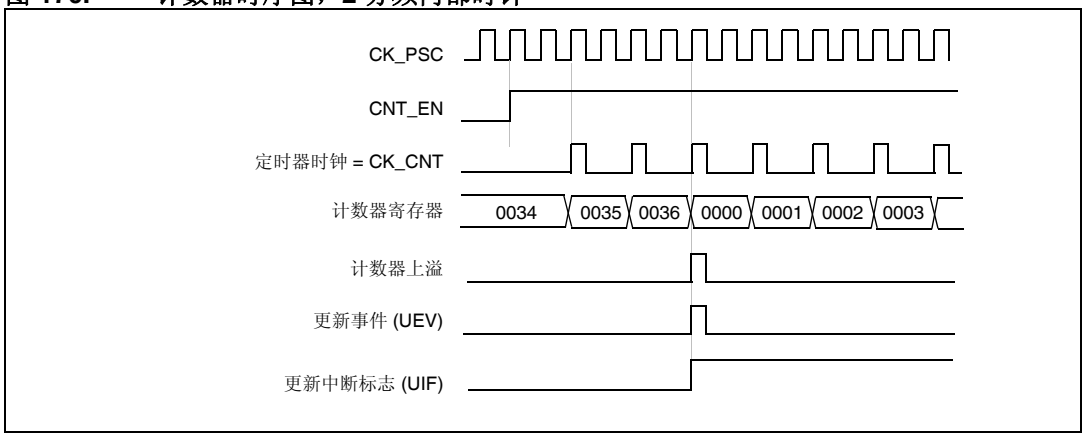


图 171.　计数器时序图，4 分频内部时钟

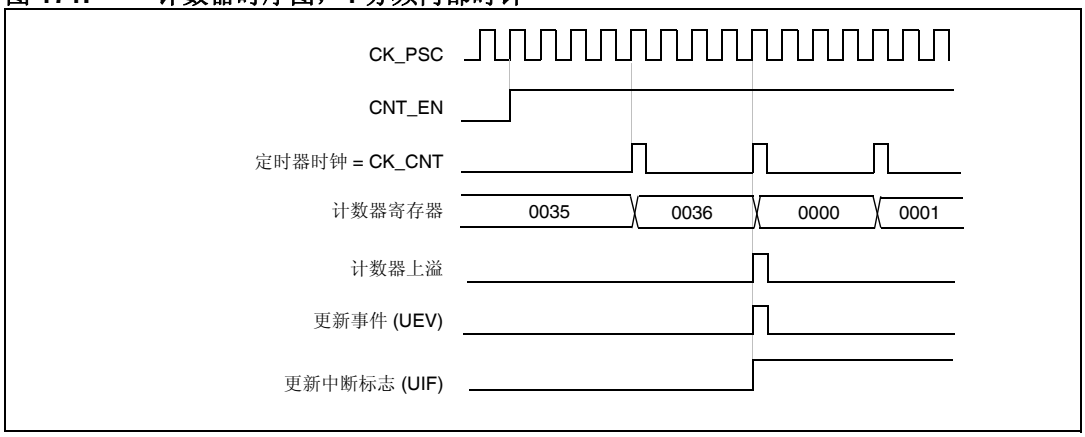


图 172.　计数器时序图，N 分频内部时钟

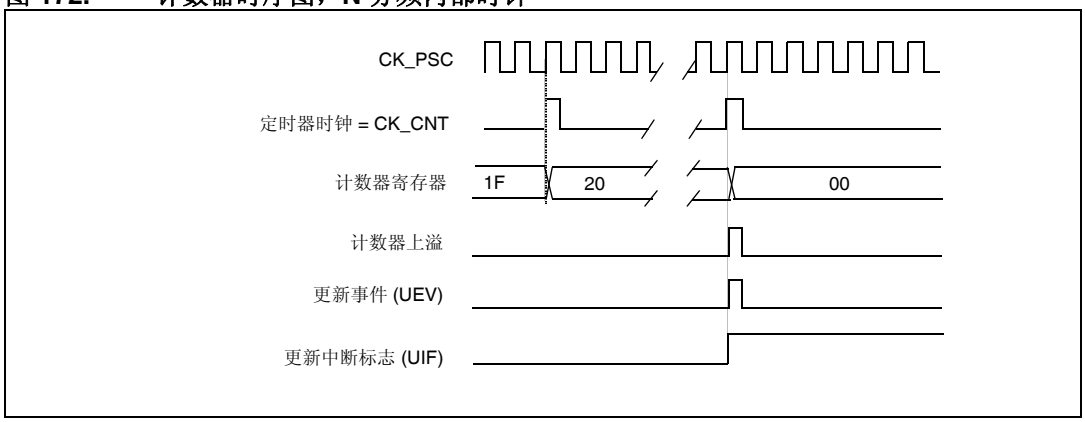


图 173.　计数器时序图，ARPE=0 时更新事件（TIMx_ARR 未预装载）

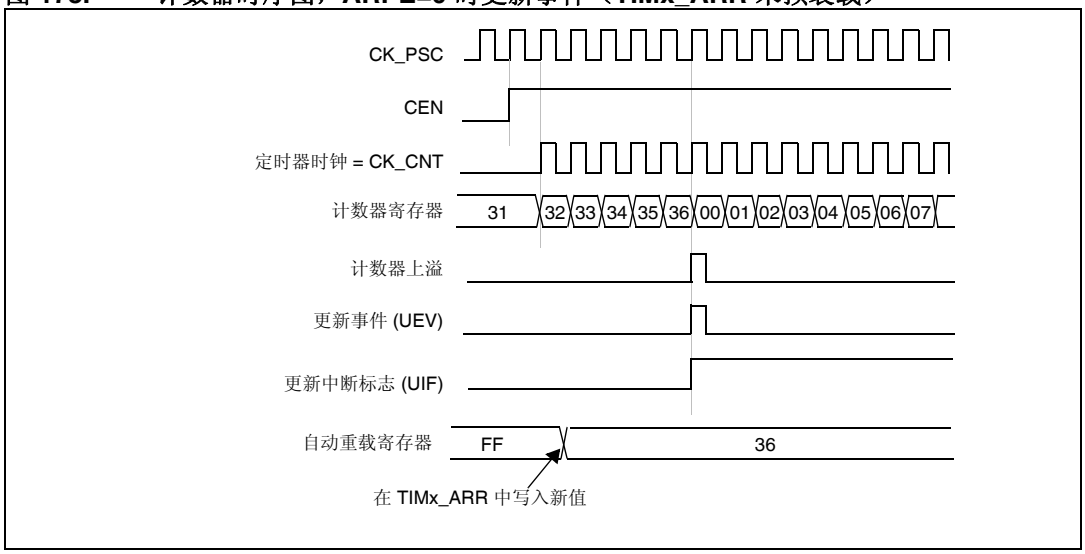
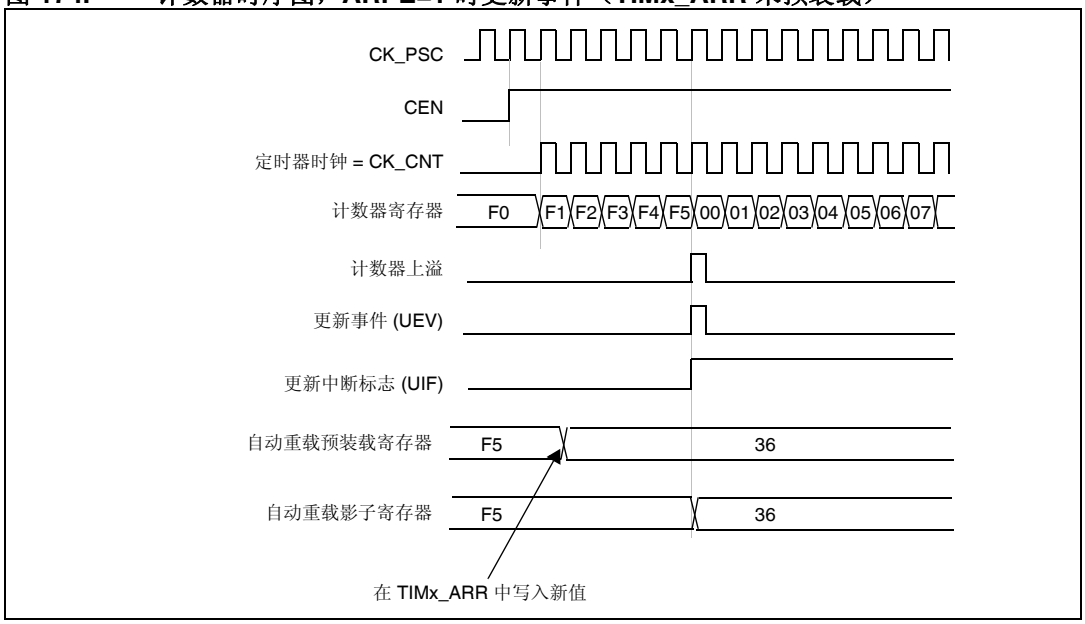


图 174. 计数器时序图，ARPE=1 时更新事件（TIMx_ARR 未预装载）



16.4.3 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（针对 **TIM9 和 TIM12**）：外部输入引脚 (TIx)
- 内部触发输入 (ITRx)（针对 **TIM9 和 TIM12**）：连接来自其它计数器的触发输出。更多详细信息，请参见*将一个定时器用作另一个定时器的预分频器*一节

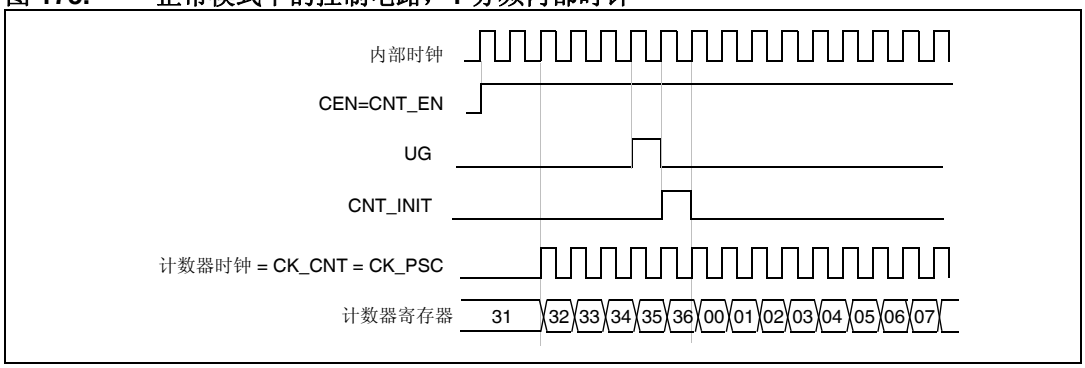
内部时钟源 (CK_INT)

内部时钟源是 TIM10/TIM11 和 TIM13/TIM14 的默认时钟源。

对于 TIM9 和 TIM12，在禁止从模式控制器后选择内部时钟源（SMS=“000”）。然后，将 TIMx_CR1 寄存器中的 CEN 位和 TIMx_EGR 寄存器中的 UG 位用作控制位，并且只能通过软件对其进行更改（保持清零状态的 UG 除外）。当对 CEN 位写入 1 时，预分频器的时钟就由内部时钟 CK_INT 提供。

图 175 显示了正常模式下控制电路与递增计数器的行为（没有预分频的情况下）。

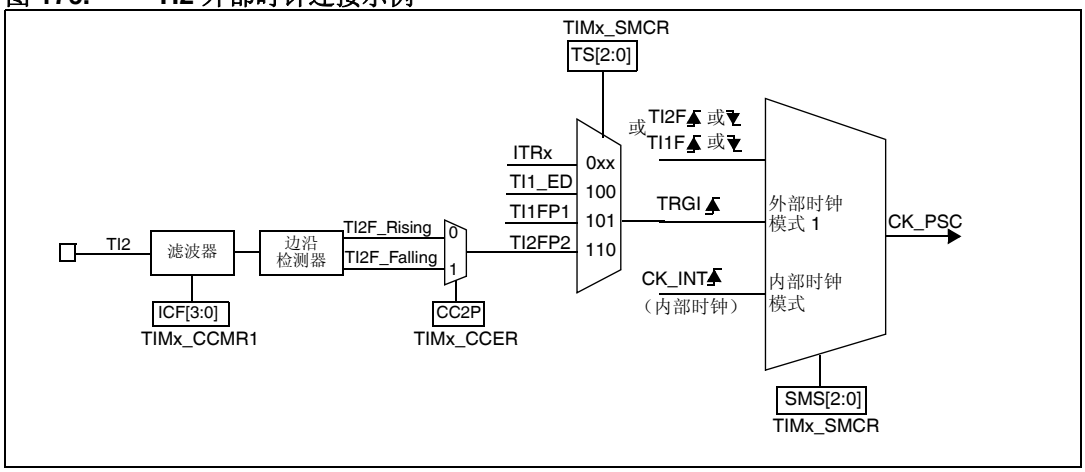
图 175. 正常模式下的控制电路，1 分频内部时钟



外部时钟源模式 1（TIM9 和 TIM12）

当 TIMx_SMCR 寄存器中的 SMS=“111”时，可选择此模式。计数器可在选定的输入信号上出现上升沿或下降沿时计数。

图 176. TI2 外部时钟连接示例



例如，要使递增计数器在 TI2 输入出现上升沿时计数，请执行以下步骤：

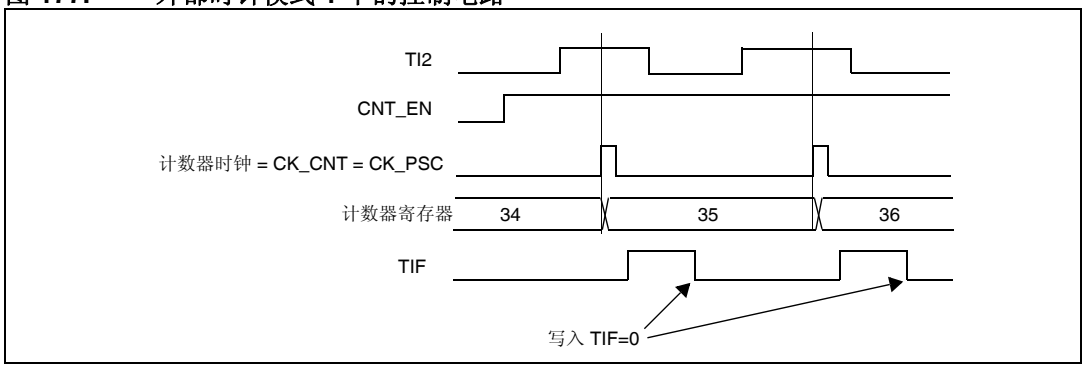
1. 通过在 TIMx_CCMR1 寄存器中写入 CC2S = “01”来配置通道 2，使其能够检测 TI2 输入的上升沿。
2. 通过在 TIMx_CCMR1 寄存器中写入 IC2F[3:0] 位来配置输入滤波时间（如果不需要任何滤波器，请保持 IC2F=“0000”）。
3. 通过在 TIMx_CCER 寄存器中写入 CC2P=“0”和 CC2NP=“0”来选择上升沿极性。
4. 通过在 TIMx_SMCR 寄存器中写入 SMS=“111”，使定时器在外部时钟模式 1 下工作。
5. 通过在 TIMx_SMCR 寄存器中写入 TS=“110”来选择 TI2 作为触发输入源。
6. 通过在 TIMx_CR1 寄存器中写入 CEN=“1”来使能计数器。

注意： *由于捕获预分频器不用于触发操作，因此无需对其进行配置。*

当 TI2 出现上升沿时，计数器便会计数一次并且 TIF 标志置 1。

TI2 的上升沿与实际计数器时钟之间的延迟是由于 TI2 输入的重新同步电路引起的。

图 177. 外部时钟模式 1 下的控制电路



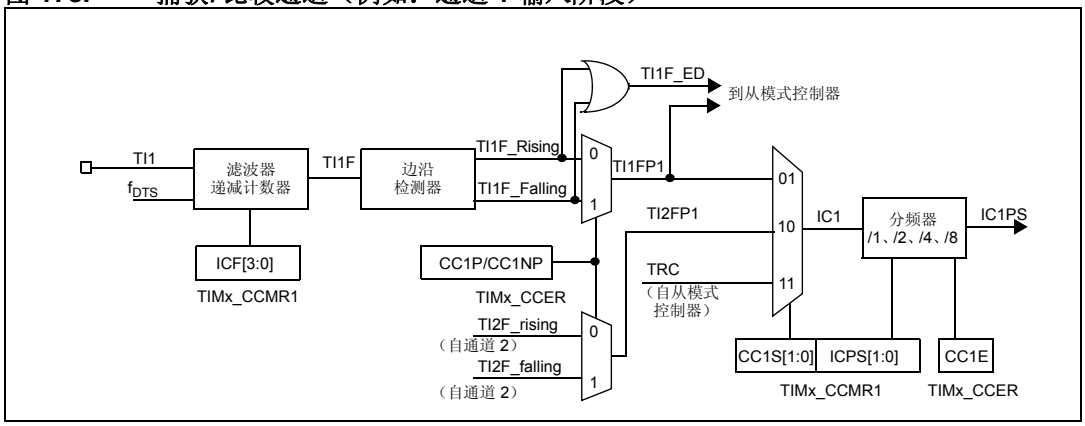
16.4.4 捕获/比较通道

每个捕获/比较通道均围绕一个捕获/比较寄存器（包括一个影子寄存器）、一个捕获输入阶段（数字滤波、多路复用和预分频器）和一个输出阶段（比较器和输出控制）构建而成。

图 178 到*图 180* 对捕获/比较通道做了简要说明。

输入阶段对相应的 TIx 输入进行采样，生成一个滤波后的信号 TIxF。然后，带有极性选择功能的边沿检测器生成一个信号 (TIxFPx)，该信号可用作从模式控制器的触发输入，也可用作捕获命令。该信号先进行预分频 (ICxPS)，而后再进入捕获寄存器。

图 178. 捕获/比较通道（例如：通道 1 输入阶段）



输出阶段生成一个中间波形作为基准：OCxRef（高电平有效）。链的末端决定最终输出信号的极性。

图 179. 捕获/比较通道 1 主电路

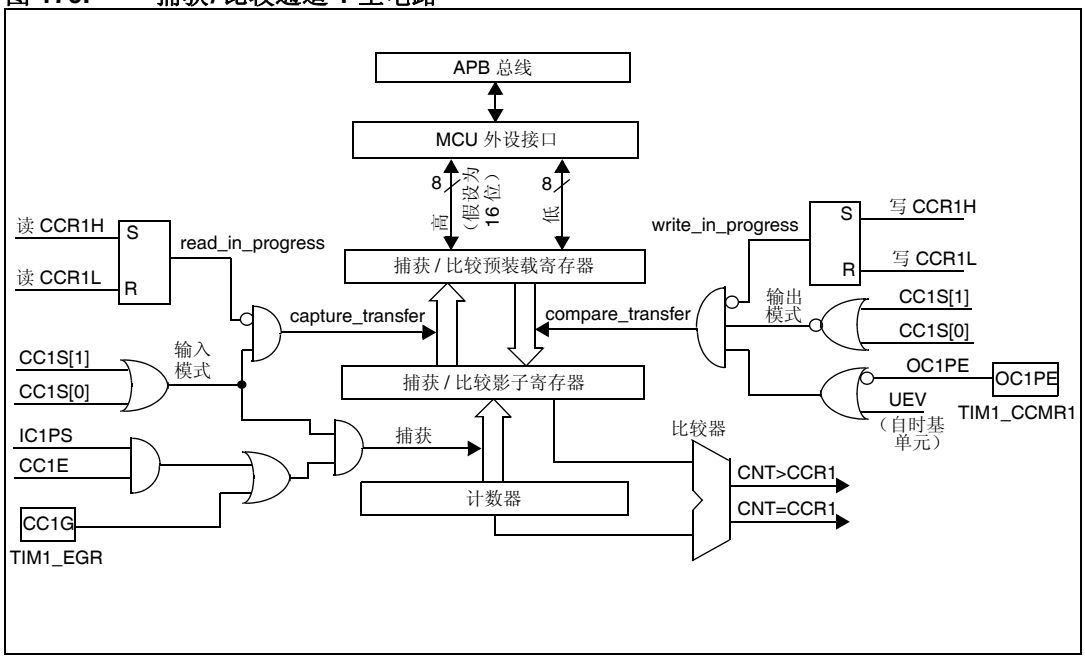
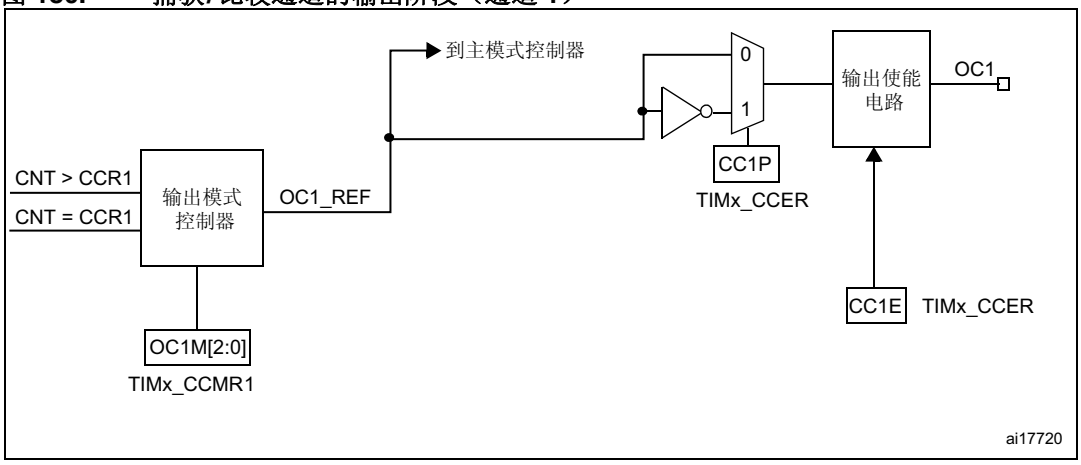


图 180. 捕获/比较通道的输出阶段（通道 1）



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。始终可通过读写操作访问预装载寄存器。

在捕获模式下，捕获实际发生在影子寄存器中，然后将影子寄存器的内容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将复制到影子寄存器中，然后将影子寄存器的内容与计数器进行比较。

16.4.5 输入捕获模式

在输入捕获模式下，当相应的 ICx 信号检测到跳变沿后，将使用捕获/比较寄存器 (TIMx_CCRx) 来锁存计数器的值。发生捕获事件时，会将相应的 CCXIF 标志（TIMx_SR 寄存器）置 1，并可发送中断或 DMA 请求（如果已使能）。如果发生捕获事件时 CCxIF 标志已处于高位，则会将重复捕获标志 CCxOF（TIMx_SR 寄存器）置 1。可通过软件向 CCxIF 写入 0 来给 CCxIF 清零，或读取存储在 TIMx_CCRx 寄存器中的已捕获数据。向 CCxOF 写入“0”后会将其清零。

以下示例说明了如何在 TI1 输入出现上升沿时将计数器的值捕获到 TIMx_CCR1 中。具体操作步骤如下：

1. 选择有效输入：TIMx_CCR1 必须连接到 TI1 输入，因此向 TIMx_CCMR1 寄存器中的 CC1S 位写入“01”。只要 CC1S 不等于“00”，就会将通道配置为输入模式，并且 TIMx_CCR1 寄存器将处于只读状态。
2. 根据连接到定时器的信号，对所需的输入滤波时间进行编程（如果输入为 TIx 输入之一，则对 TIMx_CCMRx 寄存器中的 ICxF 位进行编程）。信号变化时，输入信号最多在 5 个内部时钟周期内发生抖动。因此，我们必须将滤波时间设置为大于 5 个内部时钟周期。在检测到 8 个具有新电平的连续采样（以 f_{DTS} 频率采样）后，可以确认 TI1 上的跳变沿。然后向 TIMx_CCMR1 寄存器中的 IC1F 位写入“0011”。
3. 通过在 TIMx_CCER 寄存器中将 CC1P 位和 CC1NP 位编程为“00”，选择 TI1 上的有效转换边沿（本例中为上升沿）。
4. 对输入预分频器进行编程。在本例中，我们希望每次有效转换时都执行捕获操作，因此需要禁止预分频器（向 TIMx_CCMR1 寄存器中的 IC1PS 位写入“00”）。
5. 通过将 TIMx_CCER 寄存器中的 CC1E 位置 1，允许将计数器的值捕获到捕获寄存器中。
6. 如果需要，可通过将 TIMx_DIER 寄存器中的 CC1IE 位置 1 来使能相关中断请求。

发生输入捕获时：

- 发生有效跳变沿时，TIMx_CCR1 寄存器会获取计数器的值。
- 将 CC1IF 标志置 1（中断标志）。如果至少发生了两次连续捕获，但 CC1IF 标志未被清零，这样 CC1OF 捕获溢出标志会被置 1。
- 根据 CC1IE 位生成中断。

要处理重复捕获，建议在读出捕获溢出标志之前读取数据。这样可避免丢失在读取捕获溢出标志之后与读取数据之前可能出现的重复捕获信息。

注意： *通过软件将 TIMx_EGR 寄存器中的相应 CCxG 位置 1 可生成 IC 中断请求。*

16.4.6 PWM 输入模式（仅适用于 TIM9/12）

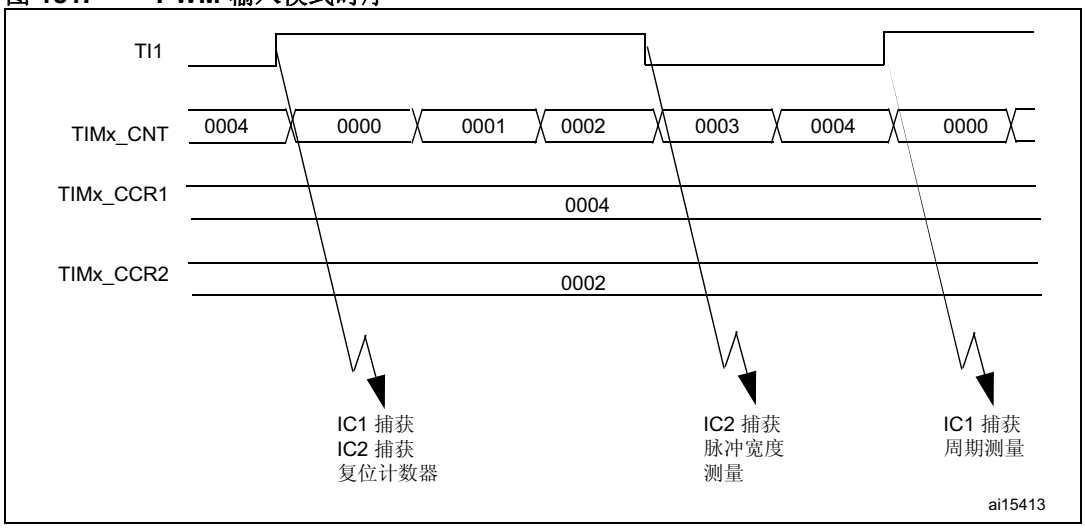
此模式是输入捕获模式的一个特例。其实现步骤与输入捕获模式基本相同，仅存在以下不同之处：

- 两个 ICx 信号被映射至同一个 TIx 输入。
- 这两个 ICx 信号在边沿处有效，但极性相反。
- 选择两个 TIxFP 信号之一作为触发输入，并将从模式控制器配置为复位模式。

例如，可通过以下步骤对应用于 TI1 的 PWM 的周期（位于 TIMx_CCR1 寄存器中）和占空比（位于 TIMx_CCR2 寄存器中）进行测量（取决于 CK_INT 频率和预分频器的值）：

1. 选择 TIMx_CCR1 的有效输入：向 TIMx_CCMR1 寄存器中的 CC1S 位写入“01”（选择 TI1）。
2. 选择 TI1FP1 的有效极性（同时用于 TIMx_CCR1 中的捕获和计数器清零）：将 CC1P 位和 CC1NP 位编程为“00”（上升沿有效）。
3. 选择 TIMx_CCR2 的有效输入：向 TIMx_CCMR1 寄存器中的 CC2S 位写入“10”（选择 TI1）。
4. 选择 TI1FP2 的有效极性（用于 TIMx_CCR2 中的捕获）：将 CC2P 位和 CC2NP 位编程为“11”（下降沿有效）。
5. 选择有效触发输入：向 TIMx_SMCR 寄存器中的 TS 位写入“101”（选择 TI1FP1）。
6. 将从模式控制器配置为复位模式：向 TIMx_SMCR 寄存器中的 SMS 位写入“100”。
7. 使能捕获：向 TIMx_CCER 寄存器中的 CC1E 位和 CC2E 位写入“1”。

图 181. PWM 输入模式时序



1. PWM 输入模式只能与 TIMx_CH1/TIMx_CH2 信号配合使用，因为只有 TI1FP1 和 TI2FP2 与从模式控制器相连。

16.4.7 强制输出模式

在输出模式（TIMx_CCMRx 寄存器中的 CCxS 位 =“00”）下，可直接由软件将每个输出比较信号（OCxREF 和 OCx）强制设置为有效电平或无效电平，而无需考虑输出比较寄存器和计数器之间的任何比较结果。

要将输出比较信号 (OCXREF/OCx) 强制设置为有效电平，只需向相应 TIMx_CCMRx 寄存器中的 OCxM 位写入“101”。OCXREF 进而强制设置为高电平（OCxREF 始终为高电平有效），同时 OCx 获取 CCxP 极性位的相反值。

例如：CCxP=“0”（OCx 高电平有效）=> 将 OCx 强制设置为高电平。

通过向 TIMx_CCMRx 寄存器中的 OCxM 位写入“100”，可将 OCxREF 信号强制设置为低电平。

无论如何，TIMx_CCRx 影子寄存器与计数器之间的比较仍会执行，而且允许将标志置 1。因此可相应发送中断请求。下面的输出比较模式一节对此进行了介绍。

16.4.8 输出比较模式

此功能用于控制输出波形，或指示已经过某一时间段。

当捕获/比较寄存器与计数器之间相匹配时，输出比较功能：

1. 将为相应的输出引脚分配一个可编程值，该值由输出比较模式（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。匹配时，输出引脚既可保持其电平（OCXM=“000”），也可设置为有效电平（OCxM=“001”）、无效电平（OCxM=“010”）或进行翻转（OCxM=“011”）。
2. 将中断状态寄存器中的标志置 1（TIMx_SR 寄存器中的 CCxIF 位）。
3. 如果相应中断使能位（TIMx_DIER 寄存器中的 CCXIE 位）置 1，将生成中断。

使用 TIMx_CCMRx 寄存器中的 OCxPE 位，可将 TIMx_CCRx 寄存器配置为带或不带预装载寄存器。

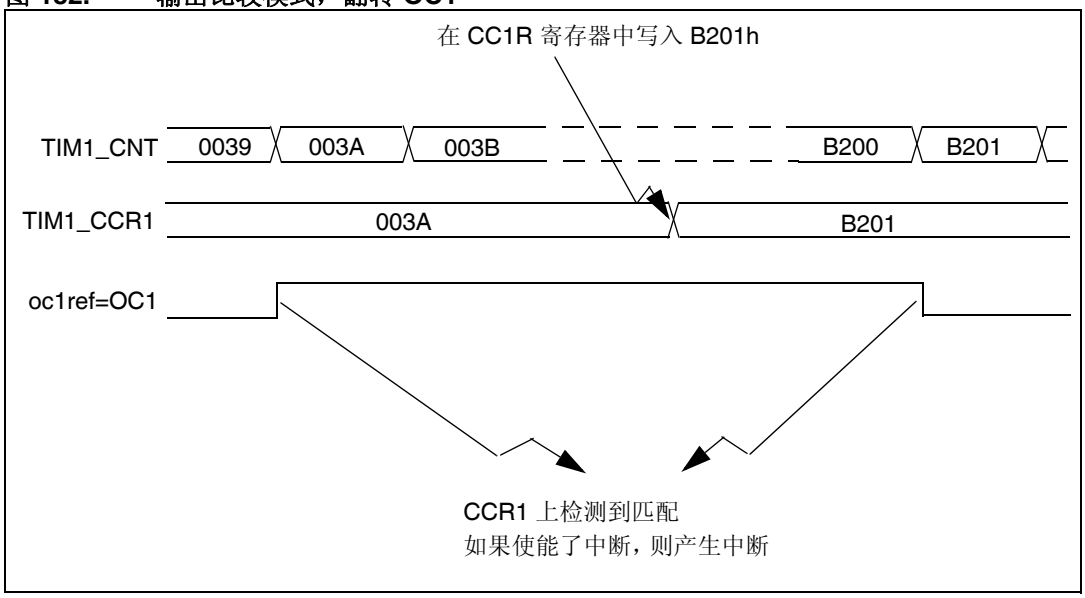
在输出比较模式下，更新事件 UEV 对 OCxREF 和 OCx 输出毫无影响。同步的精度可以达到计数器的一个计数周期。输出比较模式也可用于输出单脉冲（在单脉冲模式下）。

步骤：

1. 选择计数器时钟（内部、外部、预分频器）。
2. 在 TIMx_ARR 和 TIMx_CCRx 寄存器中写入所需数据。
3. 如果要生成中断请求，则需将 CCxIE 位置 1。
4. 选择输出模式。例如：
   - 当 CNT 与 CCRx 匹配时，写入 OCxM = “011” 以翻转 OCx 输出引脚
   - 写入 OCxPE = “0” 以禁止预装载寄存器
   - 写入 CCxP = “0” 以选择高电平有效极性
   - 写入 CCxE = “1” 以使能输出
5. 通过将 TIMx_CR1 寄存器中的 CEN 位置 1 来使能计数器。

可通过软件随时更新 TIMx_CCRx 寄存器以控制输出波形，前提是未使能预加载寄存器（OCxPE= “0”，否则仅当发生下一个更新事件 UEV 时，才会更新 TIMx_CCRx 影子寄存器）。*图 182* 给出了一个示例。

图 182. 输出比较模式，翻转 OC1



16.4.9 PWM 模式

脉冲宽度调制模式可以生成一个信号，该信号频率由 TIMx_ARR 寄存器值决定，其占空比则由 TIMx_CCRx 寄存器值决定。

通过向 TIMx_CCMRx 寄存器中的 OCxM 位写入 110（PWM 模式 1）或 111（PWM 模式 2），可以独立选择各通道 （每个 OCx 输出对应一个 PWM）的 PWM 模式。必须通过将 TIMx_CCMRx 寄存器中的 OCxPE 位置 1 使能相应预装载寄存器，最后通过将 TIMx_CR1 寄存器中的 ARPE 位置 1 使能自动重载预装载寄存器（在递增计数或中心对齐模式下）。

由于只有在发生更新事件时预装载寄存器才会传送到影子寄存器，因此启动计数器之前，必须通过将 TIMx_EGR 寄存器中的 UG 位置 1 来初始化所有寄存器。

OCx 极性可使用 TIMx_CCER 寄存器的 CCxP 位来编程。既可以设为高电平有效，也可以设为低电平有效。OCx 输出通过将 TIMx_CCER 寄存器中的 CCxE 位置 1 来使能。有关详细信息，请参见 TIMx_CCERx 寄存器说明。

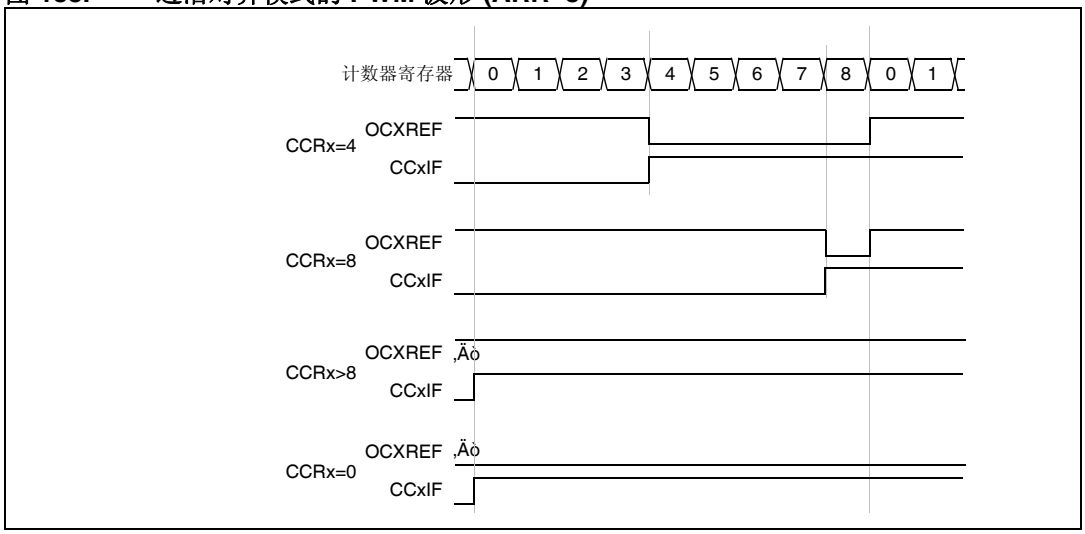
在 PWM 模式（1 或 2）下，TIMx_CNT 总是与 TIMx_CCRx 进行比较，以确定是 TIMx_CNT ≤ TIMx_CCRx。

因为计数器采用递增方式计数，所以定时器能够在边沿对齐模式下生成 PWM。

PWM 边沿对齐模式

以下以 PWM 模式 1 为例。只要 TIMx_CNT < TIMx_CCRx，PWM 参考信号 OCxREF 便为高电平，否则为低电平。如果 TIMx_CCRx 中的比较值大于自动重载值（TIMx_ARR 中），则 OCxREF 保持为“1”。如果比较值为 0，则 OCxRef 保持为“0”。*图 183* 举例介绍边沿对齐模式的一些 PWM 波形 (TIMx_ARR=8)。

图 183. 边沿对齐模式的 PWM 波形 (ARR=8)



16.4.10 单脉冲模式

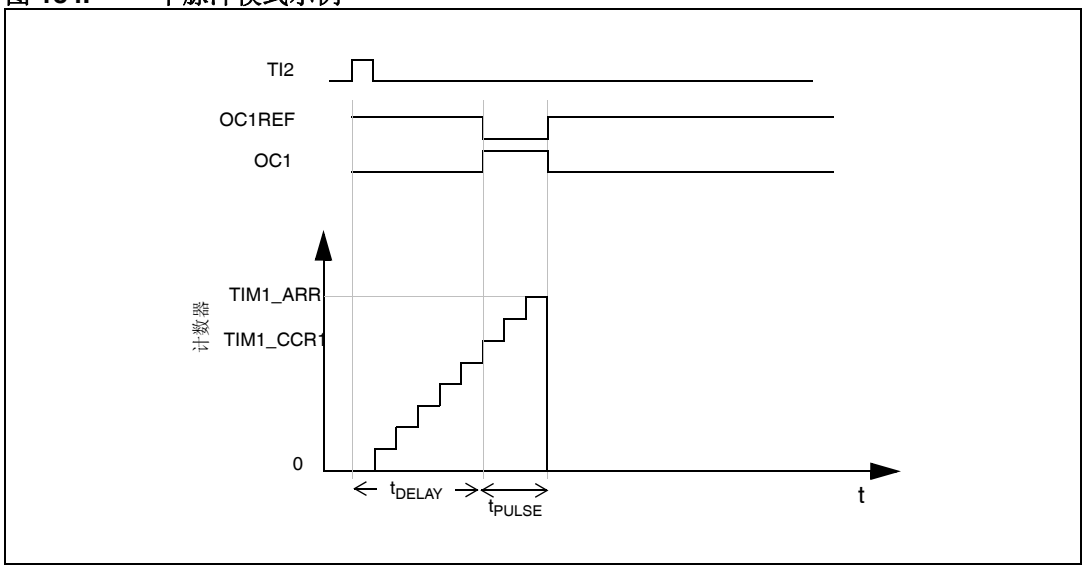
单脉冲模式 (OPM) 是上述模式的一个特例。在这种模式下，计数器可以在一个激励信号的触发下启动，并可在一段可编程的延时后产生一个脉宽可编程的脉冲。

可以通过从模式控制器启动计数器。可以在输出比较模式或 PWM 模式下生成波形。将 TIMx_CR1 寄存器中的 OPM 位置 1，即可选择单脉冲模式。这样，发生下一更新事件 UEV 时，计数器将自动停止。

只有当比较值与计数器初始值不同时，才能正确产生一个脉冲。启动前（定时器等待触发时），必须进行如下配置：

CNT < CCRx ≤ ARR（特别注意，0 < CCRx）

图 184. 单脉冲模式示例



例如，用户希望达到这样的效果：在 TI2 输入引脚检测到正沿时，经过 t_{DELAY} 的延迟，在 OC1 上产生一个长度为 t_{PULSE} 的正脉冲。

使用 TI2FP2 作为触发 1：

1. 在 TIMx_CCMR1 寄存器中写入 CC2S=“01”，以将 TI2FP2 映射到 TI2。
2. 在 TIMx_CCER 寄存器中写入 CC2P=“0”和 CC2NP=“0”，使 TI2FP2 能够检测上升沿。
3. 在 TIMx_SMCR 寄存器中写入 TS=“110”，以将 TI2FP2 配置为从模式控制器的触发 (TRGI)。
4. 在 TIMx_SMCR 寄存器中写入 SMS=“110”（触发模式），以使用 TI2FP2 启动计数器。

OPM 波形通过对比较寄存器执行写操作来定义（考虑时钟频率和计数器预分频器）。

- t_{DELAY} 由写入 TIMx_CCR1 寄存器的值定义。
- t_{PULSE} 由自动重载值与比较值 (TIMx_ARR - TIMx_CCR1) 之差来定义。
- 假设希望产生这样的波形：信号在发生比较匹配时从“0”变为“1”，在计数器达到自动重载值时由“1”变为“0”。为此，应在 TIMx_CCMR1 寄存器中写入 OC1M=“111”，以使能 PWM 模式 2。如果需要，可选择在 TIMx_CCMR1 寄存器的 OC1PE 和 TIMx_CR1 寄存器的 ARPE 中写入“1”，以使能预装载寄存器。这种情况下，必须在 TIMx_CCR1 寄存器中写入比较值并在 TIMx_ARR 寄存器中写入自动重载值，通过将 UG 位置 1 来产生更新，然后等待 TI2 上的外部触发事件。本例中，CC1P 的值为“0”。

由于仅需要 1 个脉冲（单脉冲模式），因此应向 TIMx_CR1 寄存器的 OPM 位写入“1”，以便在发生下一更新事件（计数器从自动重载值返回到 0）时使计数器停止计数。TIMx_CR1 寄存器中的 OPM 位置“0”时，即选择重复模式。

特例：OCx 快速使能

在单脉冲模式下，TIx 输入的边沿检测会将 CEN 位置 1，表示使能计数器。然后，在计数器值与比较值之间发生比较时，将切换输出。但是，完成这些操作需要多个时钟周期，这会限制可能的最小延迟（t_{DELAY} 最小值）。

如果要输出延迟时间最短的波形，可以将 TIMx_CCMRx 寄存器中的 OCxFE 位置 1。这样会强制 OCxRef（和 OCx）对激励信号做出响应，而不再考虑比较的结果。其新电平与发生比较匹配时相同。仅当通道配置为 PWM1 或 PWM2 模式时，OCxFE 才会起作用。

16.4.11 TIM9/12 外部触发同步

TIM9/12 定时器可以下列模式与外部触发实现同步：复位模式、门控模式和触发模式。

从模式：复位模式

当触发输入信号产生变化时，计数器及其预分频器可重新初始化。此外，如果 TIMx_CR1 寄存器中的 URS 位处于低电平，则会生成更新事件 UEV。然后，所有预装载寄存器（TIMx_ARR 和 TIMx_CCRx）都将更新。

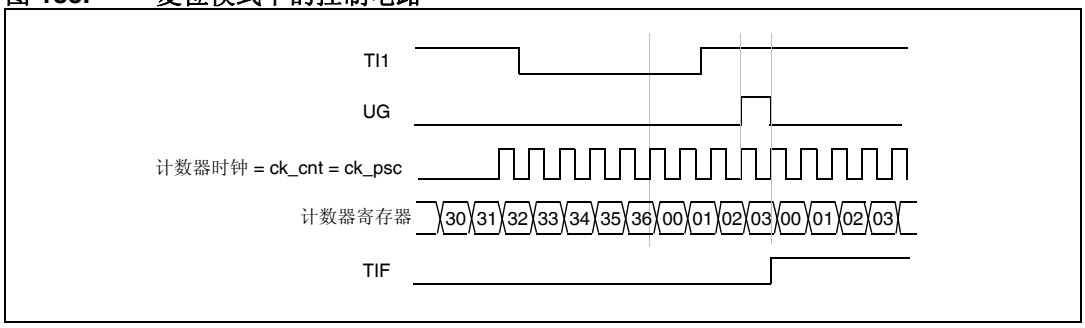
在以下示例中，TI1 输入上出现上升沿时，递增计数器清零：

1. 将通道 1 配置为检测 TI1 的上升沿。配置输入滤波时间（本例中不需要任何滤波器，因此保持 IC1F=“0000”）。由于捕获预分频器不用于触发操作，因此无需对其进行配置。CC1S 位只选择输入捕获源，即 TIMx_CCMR1 寄存器中的 CC1S = “01”。将 TIMx_CCER 寄存器中的 CC1P 和 CC1NP 编程为“00”，以验证极性（仅检测上升沿）。
2. 在 TIMx_SMCR 寄存器中写入 SMS=“100”，将定时器配置为复位模式。在 TIMx_SMCR 寄存器中写入 TS=“101”，选择 TI1 作为输入源。
3. 通过在 TIMx_CR1 寄存器中写入 CEN=“1”来启动动计数器。

计数器使用内部时钟计数，然后正常运转，直到出现 TI1 上升沿。当 TI1 出现上升沿时，计数器清零，然后重新从 0 开始计数。同时，触发标志（TIMx_SR 寄存器中的 TIF 位）置 1，使能中断，还可发送中断请求（取决于 TIMx_DIER 寄存器中的 TIE 位）。

下图显示了自动重载寄存器 TIMx_ARR=0x36 时的相关行为。TI1 的上升沿与实际计数器复位之间的延迟是由于 TI1 输入的重新同步电路引起的。

图 185. 复位模式下的控制电路



从模式：门控模式

输入信号的电平可用来使能计数器。

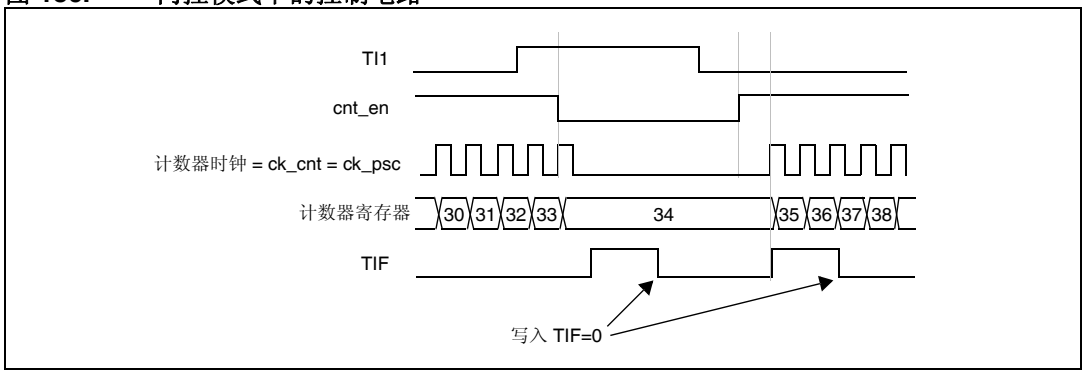
在以下示例中，递增计数器仅在 TI1 输入为低电平时计数：

1. 将通道 1 配置为检测 TI1 上的低电平。配置输入滤波时间（本例中不需要任何滤波器，因此保持 IC1F=“0000”）。由于捕获预分频器不用于触发操作，因此无需对其进行配置。CC1S 位只选择输入捕获源，即 TIMx_CCMR1 寄存器中的 CC1S = “01”。将 TIMx_CCER 寄存器中的 CC1P 和 CC1NP 分别编程为“1”和“0”，以确定极性（仅检测低电平）。
2. 在 TIMx_SMCR 寄存器中写入 SMS=“101”，将定时器配置为门控模式。在 TIMx_SMCR 寄存器中写入 TS=“101”，选择 TI1 作为输入源。
3. 在 TIMx_CR1 寄存器中写入 CEN=“1”，使能计数器（在门控模式下，如果 CEN=“0”，则不论触发输入电平如何，计数器都不启动）。

只要 TI1 为低电平，计数器就开始根据内部时钟计数，直到 TI1 变为高电平时停止计数。计数器启动或停止时，TIMx_SR 寄存器中的 TIF 标志都会置 1。

TI1 的上升沿与实际计数器停止之间的延迟是由于 TI1 输入的重新同步电路引起的。

图 186. 门控模式下的控制电路



从模式：触发模式

所选输入上发生某一事件时可以启动计数器。

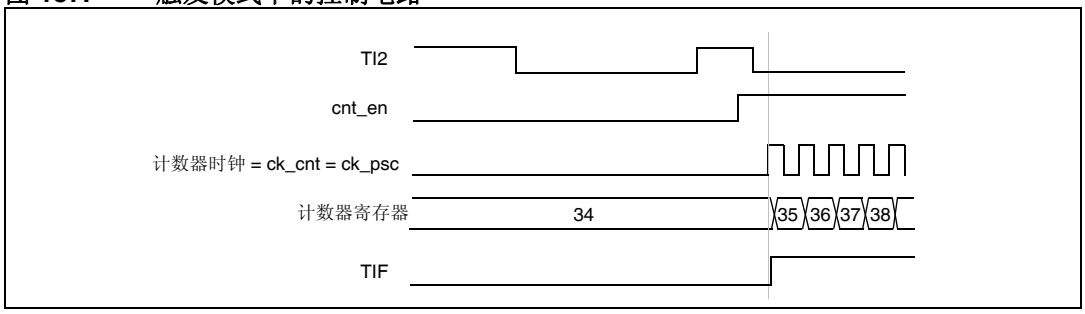
在以下示例中，TI2 输入上出现上升沿时，递增计数器启动：

1. 将通道 2 配置为检测 TI2 上的上升沿。配置输入滤波时间（本例中不需要任何滤波器，因此保持 IC2F=“0000”）。由于捕获预分频器不用于触发操作，因此无需对其进行配置。CC2S 位只选择输入捕获源，即 TIMx_CCMR1 寄存器中的 CC2S=“01”。将 TIMx_CCER 寄存器中的 CC2P 和 CC2NP 分别编程为“1”和“0”，以确定极性（仅检测低电平）。
2. 在 TIMx_SMCR 寄存器中写入 SMS=“110”，将定时器配置为触发模式。在 TIMx_SMCR 寄存器中写入 TS=“110”，选择 TI2 作为输入源。

当 TI2 出现上升沿时，计数器开始根据内部时钟计数，并且 TIF 标志置 1。

TI2 的上升沿与实际计数器启动之间的延迟是由于 TI2 输入的重新同步电路引起的。

图 187. 触发模式下的控制电路



16.4.12 定时器同步 (TIM9/12)

TIM 定时器从内部链接在一起，以实现定时器同步或级联。有关详细信息，请参见*第 419 页的第 15.3.15 节：定时器同步*。

16.4.13 调试模式

当微控制器进入调试模式（Cortex™-M4F 内核停止）时，TIMx 计数器会根据 DBG 模块中的 DBG_TIMx_STOP 配置位选择继续正常工作或者停止工作。有关详细信息，请参见*第 33.16.2 节：对定时器、看门狗、bxCAN 和 I^2C 的调试支持*。

16.5 TIM9 和 TIM12 寄存器

有关寄存器说明中使用的缩写，请参见*第 1.1 节*。

外设寄存器的写访问仅支持半字（16 位）或字（32 位）。而读访问可支持字节（8 位）、半字（16 位）或字（32 位）。

16.5.1 TIM9/12 控制寄存器 1 (TIMx_CR1)

TIM9/12 control register 1

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
Reserved						CKD[1:0]		ARPE	Reserved			OPM	URS	UDIS	CEN
						rw	rw	rw				rw	rw	rw	rw

位 15:10 保留，必须保持复位值。

位 9:8 **CKD**：时钟分频 (Clock division)

此位域指示定时器时钟 (CK_INT) 频率与数字滤波器所使用的采样时钟 (TIx) 之间的分频比，

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**：自动重载预装载使能 (Auto-reload preload enable)

0：TIMx_ARR 寄存器不进行缓冲。

1：TIMx_ARR 寄存器进行缓冲。

位 6:4 保留，必须保持复位值。

位 3 **OPM**：单脉冲模式 (One-pulse mode)

0：计数器在发生更新事件时不会停止计数

1：计数器在发生下一更新事件时停止计数（将 CEN 位清零）

位 2 **URS**：更新请求源 (Update request source)

此位由软件置 1 和清零，用以选择 UEV 事件源。

0：使能后，所有以下事件都会生成更新中断：

– 计数器上溢

– 将 UG 位置 1

1：使能后，只有计数器上溢会生成更新中断。

位 1 **UDIS**：更新禁止 (Update disable)

此位由软件置 1 和清零，用以使能/禁止更新事件 (UEV) 生成。

0：使能 UEV。UEV 可通过以下事件之一生成：

– 计数器上溢

– 将 UG 位置 1

然后更新影子寄存器的值。

1：禁止 UEV。不会生成 UEV，各影子寄存器的值（ARR、PSC 和 CCRx）保持不变。如果 UG 位置 1，则会重新初始化计数器和预分频器。

位 0 **CEN**：计数器使能 (Counter enable)

0：禁止计数器
1：使能计数器

在单脉冲模式下，当发生更新事件时会自动将 CEN 位清零。

16.5.2 TIM9/12 控制寄存器 2 (TIMx_CR2)

TIM9/12 control register 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
Reserved									MMS[2:0]			Reserved			
									rw	rw	rw				

位 15:7 保留，必须保持复位值。

位 6:4 **MMS**：主模式选择 (Master mode selection)

这些位用于选择主模式下将要发送到从定时器以实现同步的信息 (TRGO)。这些位的组合如下：
000：**复位**——TIMx_EGR 寄存器中的 UG 位用作触发输出 (TRGO)。如果复位由触发输入生成（从模式控制器配置为复位模式），则 TRGO 上的信号相比实际复位会有延迟。
001：**使能**——计数器使能信号 (CNT_EN) 用作触发输出 (TRGO)。该触发输出可用于同时启动多个定时器，或者控制在一段时间内使能从定时器。计数器使能信号由 CEN 控制位与门控模式下的触发输入的逻辑或运算组合而成。
当计数器使能信号由触发输入控制时，TRGO 上会存在延迟，选择主/从模式时除外（请参见 TIMx_SMCR 寄存器中对 MSM 位的说明）。
010：**更新**——选择更新事件作为触发输出 (TRGO)。例如，主定时器可用作从定时器的预分频器。
011：**比较脉冲**——一旦发生输入捕获或比较匹配事件，当CC1IF被置1时（即使已为高电平），触发输出都会发送一个正脉冲。(TRGO)。
100：**比较**——OC1REF 信号用作触发输出 (TRGO)。
101：**比较**——OC2REF 信号用作触发输出 (TRGO)。
110：保留。
111：保留。

位 3:0 保留，必须保持复位值。

16.5.3 TIM9/12 从模式控制寄存器 (TIMx_SMCR)

TIM9/12 slave mode control register

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
Reserved								MSM	TS[2:0]			Res.	SMS[2:0]		
								rw	rw	rw	rw		rw	rw	rw

位 15:8 保留，必须保持复位值。

位 7 **MSM**：主/从模式 (Master/Slave mode)

0：不执行任何操作。

1：延迟事件对触发输入 (TRGI) 的影响，以使当前定时器与其从定时器实现完美同步（通过 TRGO）。此设置适用于要在发生单个外部事件时对多个定时器进行同步的情况。

位 6:4 **TS**：触发选择 (Trigger selection)

此位域可选择将要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 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：保留

有关各定时器 ITRx 含义的详细信息，请参见 *第 464 页的表 79：TIMx 内部触发连接*。

注意：这些位只能在未使用的情况下（例如，SMS=“000”时）进行更改，以避免转换时出现错误的边沿检测。

位 3 保留，必须保持复位值。

位 2:0 **SMS**：从模式选择 (Slave mode selection)

选择外部信号时，触发信号 (TRGI) 的有效边沿与外部输入上所选的极性相关（请参见输入控制寄存器和控制寄存器说明）。

000：禁止从模式——如果 CEN =“1”，预分频器时钟直接由内部时钟提供

001：保留

010：保留

011：保留

100：复位模式——在出现所选触发输入 (TRGI) 上升沿时，重新初始化计数器并生成一个寄存器更新事件

101：门控模式——触发输入 (TRGI) 为高电平时使能计数器时钟。只要触发输入变为低电平，计数器立即停止计数（但不复位）。计数器的启动和停止都是受控的。

110：触发模式——触发信号 TRGI 出现上升沿时启动计数器（但不复位）。只控制计数器的启动

111：外部时钟模式 1——在所选触发 (TRGI) 的上升沿驱动计数器

注意：如果将 TI1F_ED 选作触发输入（TS=“100”），则不得使用门控模式。实际上，TI1F 每次转换时，TI1F_ED 都输出 1 个脉冲，而门控模式检查的则是触发信号的电平。

表 79. TIMx 内部触发连接

从 TIM	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
TIM9	TIM2	TIM3	TIM10	TIM11
TIM12	TIM4	TIM5	TIM13	TIM14

16.5.4 TIM9/12 中断使能寄存器 (TIMx_DIER)

TIM9/12 Interrupt enable register

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
Reserved									TIE	Res			CC2IE	CC1IE	UIE
									rw				rw	rw	rw

位 15:7　保留，必须保持复位值。

位 6 **TIE**：触发信号 (TGRI) 中断使能 (Trigger interrupt enable)

0：禁止触发信号 (TGRI) 中断。
1：使能触发信号 (TGRI) 中断。

位 5:3　保留，必须保持复位值。

位 2 **CC2IE**：捕获/比较 2 中断使能 (Capture/Compare 2 interrupt enable)

0：禁止 CC2 中断。
1：使能 CC2 中断。

位 1 **CC1IE**：捕获/比较 1 中断使能 (Capture/Compare 1 interrupt enable)

0：禁止 CC1 中断。
1：使能 CC1 中断。

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。

16.5.5 TIM9/12 状态寄存器 (TIMx_SR)

TIM9/12 status register

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
Reserved					CC2OF	CC1OF	Reserved		TIF	Reserved			CC2IF	CC1IF	UIF
					rc_w0	rc_w0			rc_w0				rc_w0	rc_w0	rc_w0

位 15:11　保留，必须保持复位值。

位 10 **CC2OF**：捕获/比较 2 重复捕获标志 (Capture/compare 2 overcapture flag)

请参见 CC1OF 说明

位 9 **CC1OF**：捕获/比较 1 重复捕获标志 (Capture/Compare 1 overcapture flag)

仅当将相应通道配置为输入捕获模式时，此标志位才会由硬件置 1。通过软件写入“0”可将该位清零。

0：未检测到重复捕获。
1：TIMx_CCR1 寄存器中已捕获到计数器值且 CC1IF 标志已置 1。

位 8:7　保留，必须保持复位值。

位 6 **TIF**：触发中断标志 (Trigger interrupt flag)

在除门控模式以外的所有模式下，当使能从模式控制器后在 TRGI 输入上检测到有效边沿时，该标志将由硬件置 1。选择门控模式时，该标志将在计数器启动或停止时置 1。但需要通过软件清零。

0：未发生触发事件。

1：触发中断挂起。

位 5:3 保留，必须保持复位值。

位 2 **CC2IF**：捕获/比较 2 中断标志 (Capture/Compare 2 interrupt flag)

请参见 CC1IF 说明

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：

当计数器与比较值匹配时，此标志由硬件置 1。但需要通过软件清零。

0：不匹配。

1：TIMx_CNT 计数器的值与 TIMx_CCR1 寄存器的值匹配。当 TIMx_CCR1 的值大于 TIMx_ARR 的值时，CC1IF 位将在计数器发生上溢时变为高电平。

如果通道 CC1 配置为输入：

此位将在发生捕获事件时由硬件置 1。通过软件或读取 TIMx_CCR1 寄存器将该位清零。

0：未发生输入捕获事件。

1：TIMx_CCR1 寄存器中已捕获到计数器值（IC1 上已检测到与所选极性匹配的边沿）。

位 0 **UIF**：更新中断标志 (Update interrupt flag)

该位在发生更新事件时通过硬件置 1。但需要通过软件清零。

0：未发生更新。

1：更新中断挂起。该位在以下情况下更新寄存器时由硬件置 1：

– 上溢且当 TIMx_CR1 寄存器中 UDIS = “0” 时。

– 当由于 TIMx_CR1 寄存器中 URS = “0” 且 UDIS = “0” 而通过软件使用 TIMx_EGR 寄存器中的 UG 位重新初始化 CNT 时。

– 当由于 TIMx_CR1 寄存器中 URS = “0” 且 UDIS = “0” 而通过触发事件（请参见同步控制寄存器说明）重新初始化 CNT 时。

16.5.6 TIM9/12 事件生成寄存器 (TIMx_EGR)

TIM9/12 event generation register

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
Reserved									TG	Reserved			CC2G	CC1G	UG
									w				w	w	w

位 15:7 保留，必须保持复位值。

位 6 **TG**：生成触发信号 (Trigger generation)

此位由软件置 1 以生成事件，并由硬件自动清零。

0：不执行任何操作

1：TIMx_SR 寄存器中的 TIF 标志置 1。使能后可发生相关中断

位 5:3 保留，必须保持复位值。

位 2 **CC2G**：捕获/比较 2 生成 (Capture/compare 2 generation)

请参见 CC1G 说明

位 1 **CC1G**：捕获/比较 1 生成 (Capture/compare 1 generation)

此位由软件置 1 以生成事件，并由硬件自动清零。
0：不执行任何操作
1：通道 1 上生成捕获/比较事件：
如果通道 CC1 配置为输出：
使能后，CC1IF 标志置 1 并发送相应中断。
如果通道 CC1 配置为输入：
TIMx_CCR1 寄存器中将捕获到计数器的当前值。使能后，CC1IF 标志置 1 并发送相应中断。如果 CC1IF 标志已为高电平，CC1OF 标志将置 1。

位 0 **UG**：更新生成 (Update generation)

该位可通过软件置 1，并由硬件自动清零。
0：不执行任何操作
1：重新初始化计数器并生成寄存器更新事件。预分频器计数器也将清零，但预分频比不受影响。计数器清零

16.5.7 TIM9/12 捕获/比较模式寄存器 1 (TIMx_CCMR1)

TIM9/12 capture/compare mode register 1

偏移地址：0x18

复位值：0x0000

这些通道可用于输入（捕获模式）或输出（比较模式）模式。通道方向通过配置相应的 CCxS 位进行定义。该寄存器的所有其它位在输入模式和输出模式下的功能不同。对于任一给定位，OCxx 用于说明通道配置为输出模式时该位对应的功能，ICxx 则用于说明通道配置为输入模式时该位对应的功能。因此，必须注意同一个位在输入阶段和输出阶段具有不同的含义。

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	OC2M[2:0]			OC2PE	OC2FE	CC2S[1:0]		Res.	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	rw	rw

输出比较模式

位 15 保留，必须保持复位值。

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)

此位域定义通道方向（输入/输出）以及所使用的输入。
00：CC2 通道配置为输出
01：CC2 通道配置为输入，IC2 映射到 TI2 上
10：CC2 通道配置为输入，IC2 映射到 TI1 上
11：CC2 通道配置为输入，IC2 映射到 TRC 上。此模式仅在通过 TS 位（TIMx_SMCR 寄存器）选择内部触发输入时有效

注意：仅当通道关闭时（TIMx_CCER 中的 CC2E = 0），才可向 CC2S 位写入数据。

位 7 保留，必须保持复位值。

位 6:4 **OC1M**：输出比较 1 模式 (Output compare 1 mode)

这些位定义提供 OC1 和 OC1N 的输出参考信号 OC1REF 的行为。OC1REF 为高电平有效，而 OC1 和 OC1N 的有效电平则分别取决于 CC1P 位和 CC1NP 位。
000：冻结——输出比较寄存器 TIMx_CCR1 与计数器 TIMx_CNT 进行比较不会对输出造成任何影响。（该模式用于生成时基）。
001：将通道 1 设置为匹配时输出有效电平。当计数器 TIMx_CNT 与捕获/比较寄存器 1 (TIMx_CCR1) 匹配时，OC1REF 信号强制变为高电平。
010：将通道 1 设置为匹配时输出无效电平。当计数器 TIMx_CNT 与捕获/比较寄存器 1 (TIMx_CCR1) 匹配时，OC1REF 信号强制变为低电平。
011：翻转——TIMx_CNT=TIMx_CCR1 时，OC1REF 发生翻转。
100：强制变为无效电平——OC1REF 强制变为低电平。
101：强制变为有效电平——OC1REF 强制变为高电平。
110：PWM 模式 1——在递增计数模式下，只要 TIMx_CNT<TIMx_CCR1，通道 1 便为有效状态，否则为无效状态。在递减计数模式下，只要 TIMx_CNT>TIMx_CCR1，通道 1 便为无效状态（OC1REF=“0”），否则为有效状态（OC1REF=“1”）。
111：PWM 模式 2——在递增计数模式下，只要 TIMx_CNT<TIMx_CCR1，通道 1 便为无效状态，否则为有效状态。在递减计数模式下，只要 TIMx_CNT>TIMx_CCR1，通道 1 便为有效状态，否则为无效状态。

注意：在 PWM 模式 1 或 PWM 模式 2 下，仅当比较结果发生改变或输出比较模式由“冻结”模式切换到“PWM”模式时，OCREF 电平才会发生更改。

位 3 **OC1PE**：输出比较 1 预装载使能 (Output compare 1 preload enable)

0：禁止与 TIMx_CCR1 相关的预装载寄存器。可随时向 TIMx_CCR1 写入数据，写入后将立即使用新值。
1：使能与 TIMx_CCR1 相关的预装载寄存器。可读/写访问预装载寄存器。TIMx_CCR1 预装载值在每次生成更新事件时都会装载到活动寄存器中。

注意：只有单脉冲模式下才可在未验证预装载寄存器的情况下使用 PWM 模式（TIMx_CR1 寄存器中的 OPM 位置 1）。其它情况下则无法保证该行为。

位 2 **OC1FE**：输出比较 1 快速使能 (Output compare 1 fast enable)

此位用于加快触发输入事件对 CC 输出的影响。
0：即使触发开启，CC1 也将根据计数器和 CCR1 值正常工作。触发输入出现边沿时，激活 CC1 输出的最短延迟时间为 5 个时钟周期。
1：触发输入上出现有效边沿相当于 CC1 输出上的比较匹配。随后，无论比较结果如何，OC 都设置为比较电平。采样触发输入和激活 CC1 输出的延迟时间缩短为 3 个时钟周期。仅当通道配置为 PWM1 或 PWM2 模式时，OC1FE 才会起作用。

位 1:0 **CC1S**：捕获/比较 1 选择 (Capture/Compare 1 selection)

此位域定义通道方向（输入/输出）以及所使用的输入。
00：CC1 通道配置为输出
01：CC1 通道配置为输入，IC1 映射到 TI1 上
10：CC1 通道配置为输入，IC1 映射到 TI2 上
11：CC1 通道配置为输入，IC1 映射到 TRC 上。此模式仅在通过 TS 位（TIMx_SMCR 寄存器）选择内部触发输入时有效

注意：仅当通道关闭时（TIMx_CCER 中的 CC1E = 0），才可向 CC1S 位写入数据。

输入捕获模式

位 15:12 **IC2F**：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**：捕获/比较 2 选择 (Capture/compare 2 selection)

此位域定义通道方向（输入/输出）以及所使用的输入。
00：CC2 通道配置为输出
01：CC2 通道配置为输入，IC2 映射到 TI2 上
10：CC2 通道配置为输入，IC2 映射到 TI1 上
11：CC2 通道配置为输入，IC2 映射到 TRC 上。此模式仅在通过 TS 位（TIMx_SMCR 寄存器）选择内部触发输入时有效

注意：仅当通道关闭时（TIMx_CCER 中的 CC2E = 0），才可向 CC2S 位写入数据。

位 7:4 **IC1F**：输入捕获 1 滤波器 (Input capture 1 filter)

此位域可定义 TI1 输入的采样频率和适用于 TI1 的数字滤波器带宽。数字滤波器由事件计数器组成，每 N 个事件才视为一个有效边沿：

0000：无滤波器，以 f_{DTS} 频率进行采样
0001：$f_{SAMPLING}=f_{CK\_INT}$，N=2
0010：$f_{SAMPLING}=f_{CK\_INT}$，N=4
0011：$f_{SAMPLING}=f_{CK\_INT}$，N=8
0100：$f_{SAMPLING}=f_{DTS}/2$，N=6
0101：$f_{SAMPLING}=f_{DTS}/2$，N=8
0110：$f_{SAMPLING}=f_{DTS}/4$，N=6
0111：$f_{SAMPLING}=f_{DTS}/4$，N=8
1000：$f_{SAMPLING}=f_{DTS}/8$，N=6
1001：$f_{SAMPLING}=f_{DTS}/8$，N=8
1010：$f_{SAMPLING}=f_{DTS}/16$，N=5
1011：$f_{SAMPLING}=f_{DTS}/16$，N=6
1100：$f_{SAMPLING}=f_{DTS}/16$，N=8
1101：$f_{SAMPLING}=f_{DTS}/32$，N=5
1110：$f_{SAMPLING}=f_{DTS}/32$，N=6
1111：$f_{SAMPLING}=f_{DTS}/32$，N=8

注意：在当前硅版本中，当 ICxF[3:0]= 1、2 或 3 时，将用 CK_INT 代替公式中的 f_{DTS}。

位 3:2 **IC1PSC**：输入捕获 1 预分频器 (Input capture 1 prescaler)

此位域定义 CC1 输入 (IC1) 的预分频比。
只要 CC1E= “0” （TIMx_CCER 寄存器），预分频器便立即复位。
00：无预分频器，捕获输入上每检测到一个边沿便执行捕获
01：每发生 2 个事件便执行一次捕获
10：每发生 4 个事件便执行一次捕获
11：每发生 8 个事件便执行一次捕获

位 1:0 **CC1S**：捕获/比较 1 选择 (Capture/Compare 1 selection)

此位域定义通道方向（输入/输出）以及所使用的输入。
00：CC1 通道配置为输出
01：CC1 通道配置为输入，IC1 映射到 TI1 上
10：CC1 通道配置为输入，IC1 映射到 TI2 上
11：CC1 通道配置为输入，IC1 映射到 TRC 上。此模式仅在通过 TS 位（TIMx_SMCR 寄存器）选择内部触发输入时有效

注意：仅当通道关闭时（TIMx_CCER 中的 CC1E = 0），才可向 CC1S 位写入数据。

16.5.8 TIM9/12 捕获/比较使能寄存器 (TIMx_CCER)

TIM9/12 capture/compare enable register

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
Reserved								CC2NP	Res.	CC2P	CC2E	CC1NP	Res.	CC1P	CC1E
								rw		rw	rw	rw		rw	rw

位 15:8 保留，必须保持复位值。

位 7 **CC2NP**：捕获/比较 2 输出极性 (Capture/Compare 2 output polarity)

请参见 CC1NP 说明

位 6 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)

请参见 CC1P 说明

位 4 **CC2E**：捕获/比较 2 输出使能 (Capture/Compare 2 output enable)

请参见 CC1E 说明

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：CC1NP 必须保持清零

CC1 通道配置为输入：CC1NP 与 CC1P 配合使用可定义 TI1FP1/TI2FP1 的极性（请参见 CC1P 说明）。

位 2 保留，必须保持复位值。

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：

0：OC1 高电平有效。

1：OC1 低电平有效。

CC1 通道配置为输入：

CC1NP/CC1P 位可针对触发或捕获操作选择 TI1FP1 和 TI2FP1 的极性。

00：未反相/上升沿触发

电路对 TIxFP1 上升沿敏感（在复位模式、外部时钟模式或触发模式下执行捕获或触发操作），TIxFP1 未反相（在门控模式或编码器模式下执行触发操作）。

01：反相/下降沿触发

电路对 TIxFP1 下降沿敏感（在复位模式、外部时钟模式或触发模式下执行捕获或触发操作），TIxFP1 反相（在门控模式或编码器模式下执行触发操作）。

10：保留，不使用此配置。

注意：11：未反相/上升沿和下降沿均触发

电路对 TIxFP1 上升沿和下降沿都敏感（在复位模式、外部时钟模式或触发模式下执行捕获或触发操作），TIxFP1 未反相（在门控模式下执行触发操作）。编码器模式下不得使用此配置。

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 未激活。

1：开启——在相应输出引脚上输出 OC1 信号。

CC1 通道配置为输入：

该位决定了输入捕获/比较寄存器 1 (TIMx_CCR1) 是否能够实际捕获到计数器的值。

0：禁止捕获。

1：使能捕获。

表 80. 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： *与标准 OCx 通道相连的外部 I/O 引脚的状态取决于通道 OCx 的状态以及 GPIO 寄存器。*

16.5.9 TIM9/12 计数器 (TIMx_CNT)

偏移地址：0x24

复位值：0x0000 0000

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

位 15:0 **CNT[15:0]**：计数器值 (Counter value)

16.5.10 TIM9/12 预分频器 (TIMx_PSC)

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	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 15:0 **PSC[15:0]**：预分频器值 (Prescaler value)

计数器时钟频率 CK_CNT 等于 f_{CK_PSC} / (PSC[15:0] + 1)。
PSC 包含在每次发生更新事件时要装载到实际预分频器寄存器的值。

16.5.11 TIM9/12 自动重载寄存器 (TIMx_ARR)

TIM9/12 auto-reload register

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
ARR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 15:0 **ARR[15:0]**：自动重载值 (Auto-reload value)

ARR 为要装载到实际自动重载寄存器的值。
有关 ARR 更新和行为的更多详细信息，请参见*第 447 页的第 16.4.1 节：时基单元*。
当自动重载值为空时，计数器不工作。

16.5.12 TIM9/12 捕获/比较寄存器 1 (TIMx_CCR1)

TIM9/12 capture/compare register 1

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	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 15:0 **CCR1[15:0]**：捕获/比较 1 值 (Capture/Compare 1 value)

如果通道 CC1 配置为输出：
CCR1 是捕获/比较寄存器 1 的预装载值。
如果没有通过 TIMx_CCMR 寄存器中的 OC1PE 位来使能预装载功能，写入的数值会被直接传输至当前寄存器中。否则只在发生更新事件时生效（拷贝到实际起作用的捕获/比较寄存器1）。
实际捕获/比较寄存器中包含要与计数器 TIMx_CNT 进行比较并在 OC1 输出上发出信号的值。
如果通道 CC1 配置为输入：
CCR1 为上一个输入捕获 1 事件 (IC1) 发生时的计数器值。

16.5.13 TIM9/12 捕获/比较寄存器 2 (TIMx_CCR2)

TIM9/12 capture/compare register 2

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	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 15:0 **CCR2[15:0]**：捕获/比较 2 值 (Capture/Compare 2 value)

如果通道 CC2 配置为输出：
CCR2 为要装载到实际捕获/比较 2 寄存器的值（预装载值）。
如果没有通过 TIMx_CCMR2 寄存器中的 OC2PE 位来使能预装载功能，写入的数值会被直接传输至当前寄存器中。否则只有发生更新事件时，预装载值才会复制到活动捕获/比较 2 寄存器中。
活动捕获/比较寄存器中包含要与计数器 TIMx_CNT 进行比较并在 OC2 输出上发出信号的值。
如果通道 CC2 配置为输入：
CCR2 为上一个输入捕获 2 事件 (IC2) 发生时的计数器值。

16.5.14 TIM9/12 寄存器映射

TIM9/12 寄存器可映射为 16 位可寻址寄存器，如下表所述：

表 81. TIM9/12 寄存器映射和复位值

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**TIMx_CR1**	Reserved																						CKD [1:0]		ARPE	Reserved			OPM	URS	UDIS	CEN
	Reset value																							0	0	0				0	0	0	0
0x04	**TIMx_CR2**	Reserved																									MMS[2:0]			Reserved			
	Reset value																										0	0	0				

表 81. TIM9/12 寄存器映射和复位值 （续）

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x08	**TIMx_SMCR**	Reserved																								MSM	TS[2:0]			Reserved	SMS[2:0]		
	Reset value																									0	0	0	0		0	0	0
0x0C	**TIMx_DIER**	Reserved																									TIE	Reserved			CC2IE	CC1IE	UIE
	Reset value																										0				0	0	0
0x10	**TIMx_SR**	Reserved																					CC2OF	CC1OF	Reserved		TIF	Reserved			CC2IF	CC1IF	UIF
	Reset value																						0	0			0				0	0	0
0x14	**TIMx_EGR**	Reserved																									TG	Reserved			CC2G	CC1G	UG
	Reset value																										0				0	0	0
0x18	**TIMx_CCMR1** *Output Compare mode*	Reserved																	OC2M [2:0]			OC2PE	OC2FE	CC2S [1:0]		Reserved	OC1M [2:0]			OC1PE	OC1FE	CC1S [1:0]	
	Reset value																		0	0	0	0	0	0	0		0	0	0	0	0	0	0
	TIMx_CCMR1 *Input Capture mode*	Reserved																IC2F[3:0]				IC2 PSC [1:0]		CC2S [1:0]		IC1F[3:0]				IC1 PSC [1:0]		CC1S [1:0]	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1C	Reserved																																
0x20	**TIMx_CCER**	Reserved																								CC2NP	Reserved	CC2P	CC2E	CC1NP	Reserved	CC1P	CC1E
	Reset value																									0		0	0	0		0	0
0x24	**TIMx_CNT**	Reserved																CNT[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x28	**TIMx_PSC**	Reserved																PSC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x2C	**TIMx_ARR**	Reserved																ARR[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x30	Reserved																																
0x34	**TIMx_CCR1**	Reserved																CCR1[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x38	**TIMx_CCR2**	Reserved																CCR2[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x3C to 0x4C	Reserved																																

有关寄存器边界地址的信息，请参见*第 52 页的表 2*。

16.6 TIM10/11/13/14 寄存器

外设寄存器的写访问仅支持半字（16 位）或字（32 位）。而读访问可支持字节（8 位）、半字（16 位）或字（32 位）。

16.6.1 TIM10/11/13/14 控制寄存器 1 (TIMx_CR1)

TIM10/11/13/14 control register 1

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
Reserved						CKD[1:0]		ARPE	Reserved				URS	UDIS	CEN
						rw	rw	rw					rw	rw	rw

位 15:10 保留，必须保持复位值。

位 9:8 **CKD**：时钟分频 (Clock division)

此位域指示定时器时钟 (CK_INT) 频率与数字滤波器所使用的采样时钟（ETR、TIx）之间的分频比，

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**：自动重载预装载使能 (Auto-reload preload enable)

0：TIMx_ARR 寄存器不进行缓冲
1：TIMx_ARR 寄存器进行缓冲

位 6:3 保留，必须保持复位值。

位 2 **URS**：更新请求源 (Update request source)

此位由软件置 1 和清零，用以选择更新中断 (UEV) 源。
0：使能后，所有以下事件都会生成 UEV：
– 计数器上溢
– 将 UG 位置 1
1：使能后，只有计数器上溢会生成 UEV。

位 1 **UDIS**：更新禁止 (Update disable)

此位由软件置 1 和清零，用以使能/禁止更新中断 (UEV) 事件生成。
0：使能 UEV。UEV 可通过以下事件之一生成：
– 计数器上溢
– 将 UG 位置 1
然后更新影子寄存器的值。
1：禁止 UEV。不会生成 UEV，各影子寄存器的值（ARR、PSC 和 CCRx）保持不变。如果 UG 位置 1，则会重新初始化计数器和预分频器。

位 0 **CEN**：计数器使能 (Counter enable)

0：禁止计数器
1：使能计数器

16.6.2 TIM10/11/13/14 状态寄存器 (TIMx_SR)

TIM10/11/13/14 status register

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
Reserved						CC1OF	Reserved							CC1IF	UIF
						rc_w0								rc_w0	rc_w0

位 15:10 保留，必须保持复位值。

位 9 **CC1OF**：捕获/比较 1 重复捕获标志 (Capture/Compare 1 overcapture flag)

仅当将相应通道配置为输入捕获模式时，此标志位才会由硬件置 1。通过软件写入“0”可将该位清零。

0：未检测到重复捕获。

1：TIMx_CCR1 寄存器中已捕获到计数器值且 CC1IF 标志已置 1。

位 8:2 保留，必须保持复位值。

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：

当计数器与比较值匹配时，此标志由硬件置 1。但需要通过软件清零。

0：不匹配。

1：TIMx_CNT 计数器的值与 TIMx_CCR1 寄存器的值匹配。当 TIMx_CCR1 的值大于 TIMx_ARR 的值时，CC1IF 位将在计数器发生上溢时变为高电平。

如果通道 CC1 配置为输入：

此位将在发生捕获事件时由硬件置 1。通过软件或读取 TIMx_CCR1 寄存器将该位清零。

0：未发生输入捕获事件。

1：TIMx_CCR1 寄存器中已捕获到计数器值（IC1 上已检测到与所选极性匹配的边沿）。

位 0 **UIF**：更新中断标志 (Update interrupt flag)

该位在发生更新事件时通过硬件置 1。但需要通过软件清零。

0：未发生更新。

1：更新中断挂起。该位在以下情况下更新寄存器时由硬件置 1：

- 上溢且当 TIMx_CR1 寄存器中 UDIS = “0”时。
- 当由于 TIMx_CR1 寄存器中 URS = “0”且 UDIS = “0”而通过软件使用 TIMx_EGR 寄存器中的 UG 位重新初始化 CNT 时。

16.6.3 TIM10/11/13/14 事件生成寄存器 (TIMx_EGR)

TIM10/11/13/14 event generation register

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
Reserved														CC1G	UG
														w	w

位 15:2 保留，必须保持复位值。

位 1 **CC1G**：捕获/比较 1 生成 (Capture/compare 1 generation)

此位由软件置 1 以生成事件，并由硬件自动清零。
0：不执行任何操作。
1：通道 1 上生成捕获/比较事件：
如果通道 CC1 配置为输出：
使能后，CC1IF 标志置 1 并发送相应的中断。
如果通道 CC1 配置为输入：
TIMx_CCR1 寄存器中将捕获到计数器当前值。使能后，CC1IF 标志置 1 并发送相应中断。
如果 CC1IF 标志已为高电平，CC1OF 标志将置 1。

位 0 **UG**：更新生成 (Update generation)

该位可通过软件置 1，并由硬件自动清零。
0：不执行任何操作。
1：重新初始化计数器并生成寄存器更新事件。请注意，预分频器计数器也将清零（但预分频比不受影响）。计数器清零。

16.6.4 TIM10/11/13/14 捕获/比较模式寄存器 1 (TIMx_CCMR1)

TIM10/11/13/14 capture/compare mode register 1

偏移地址：0x18

复位值：0x0000

这些通道可用于输入（捕获模式）或输出（比较模式）模式。通道方向通过配置相应的 CCxS 位进行定义。此寄存器的所有其它位在输入模式和输出模式下的功能均不同。对于任一给定位，OCxx 用于说明通道配置为输出时该位对应的功能，ICxx 则用于说明通道配置为输入时该位对应的功能。因此，必须注意同一个位在输入阶段和输出阶段具有不同的含义。

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved									OC1M[2:0]			OC1PE	OC1FE	CC1S[1:0]	
Reserved								IC1F[3:0]				IC1PSC[1:0]			
								rw	rw	rw	rw	rw	rw	rw	rw

输出比较模式

位 15:7　保留，必须保持复位值。

位 6:4 **OC1M**：输出比较 1 模式 (Output compare 1 mode)

这些位定义提供 OC1 的输出参考信号 OC1REF 的行为。OC1REF 为高电平有效，而 OC1 的有效电平则取决于 CC1P 位。

000：冻结。输出比较寄存器 TIMx_CCR1 与计数器 TIMx_CNT 进行比较不会对输出造成任何影响。

001：将通道 1 设置为匹配时输出有效电平。当计数器 TIMx_CNT 与捕获/比较寄存器 1 (TIMx_CCR1) 匹配时，OC1REF 信号强制变为高电平。

010：将通道 1 设置为匹配时输出无效电平。当计数器 TIMx_CNT 与捕获/比较寄存器 1 (TIMx_CCR1) 匹配时，OC1REF 信号强制变为低电平。

011：翻转——TIMx_CNT = TIMx_CCR1 时，OC1REF 发生翻转。

100：强制变为无效电平——OC1REF 强制变为低电平。

101：强制变为有效电平——OC1REF 强制变为高电平。

110：PWM 模式 1——只要 TIMx_CNT < TIMx_CCR1，通道 1 便为有效状态，否则为无效状态。

111：PWM 模式 2——只要 TIMx_CNT < TIMx_CCR1，通道 1 便为无效状态，否则为有效状态。

注意：在 PWM 模式 1 或 PWM 模式 2 下，仅当比较结果发生改变或输出比较模式由“冻结”模式切换到“PWM”模式时，OCREF 电平才会发生更改。

位 3 **OC1PE**：输出比较 1 预装载使能 (Output compare 1 preload enable)

0：禁止与 TIMx_CCR1 相关的预装载寄存器。可随时向 TIMx_CCR1 写入数据，写入后将立即使用新值。

1：使能与 TIMx_CCR1 相关的预装载寄存器。可读/写访问预装载寄存器。TIMx_CCR1 预装载值在每次生成更新事件时都会装载到活动寄存器中。

注意：只有单脉冲模式下才可在未验证预装载寄存器的情况下使用 PWM 模式（TIMx_CR1 寄存器中的 OPM 位置 1）。其它情况下则无法保证该行为。

位 2 **OC1FE**：输出比较 1 快速使能 (Output compare 1 fast enable)

此位用于加快触发输入事件对 CC 输出的影响。

0：即使触发开启，CC1 也将根据计数器和 CCR1 值正常工作。触发输入出现边沿时，激活 CC1 输出的最短延迟时间为 5 个时钟周期。

1：触发输入上出现有效边沿相当于 CC1 输出上的比较匹配。随后，无论比较结果如何，OC 都设置为比较电平。采样触发输入和激活 CC1 输出的延迟时间缩短为 3 个时钟周期。仅当通道配置为 PWM1 或 PWM2 模式时，OC1FE 才会起作用。

位 1:0 **CC1S**：捕获/比较 1 选择 (Capture/Compare 1 selection)

此位域定义通道方向（输入/输出）以及所使用的输入。

00：CC1 通道配置为输出。

01：CC1 通道配置为输入，IC1 映射到 TI1 上。

10：

11：

注意：仅当通道关闭时（TIMx_CCER 中的 CC1E = 0），才可向 CC1S 位写入数据。

输入捕获模式

位 15:8　保留，必须保持复位值。

位 7:4 **IC1F**：输入捕获 1 滤波器 (Input capture 1 filter)

此位域可定义 TI1 输入的采样频率和适用于 TI1 的数字滤波器带宽。数字滤波器由事件计数器组成，每 N 个事件才视为一个有效边沿：

0000：无滤波器，以 f_{DTS} 频率进行采样
0001：$f_{SAMPLING}=f_{CK\_INT}$，N=2
0010：$f_{SAMPLING}=f_{CK\_INT}$，N=4
0011：$f_{SAMPLING}=f_{CK\_INT}$，N=8
0100：$f_{SAMPLING}=f_{DTS}/2$，N=6
0101：$f_{SAMPLING}=f_{DTS}/2$，N=8
0110：$f_{SAMPLING}=f_{DTS}/4$，N=6
0111：$f_{SAMPLING}=f_{DTS}/4$，N=8
1000：$f_{SAMPLING}=f_{DTS}/8$，N=6
1001：$f_{SAMPLING}=f_{DTS}/8$，N=8
1010：$f_{SAMPLING}=f_{DTS}/16$，N=5
1011：$f_{SAMPLING}=f_{DTS}/16$，N=6
1100：$f_{SAMPLING}=f_{DTS}/16$，N=8
1101：$f_{SAMPLING}=f_{DTS}/32$，N=5
1110：$f_{SAMPLING}=f_{DTS}/32$，N=6
1111：$f_{SAMPLING}=f_{DTS}/32$，N=8

注意：在当前硅版本中，当 ICxF[3:0]= 1、2 或 3 时，将用 CK_INT 代替公式中的 f_{DTS}。

位 3:2 **IC1PSC**：输入捕获 1 预分频器 (Input capture 1 prescaler)

此位域定义 CC1 输入 (IC1) 的预分频比。
只要 CC1E=“0”（TIMx_CCER 寄存器），预分频器便立即复位。
00：无预分频器，捕获输入上每检测到一个边沿便执行捕获
01：每发生 2 个事件便执行一次捕获
10：每发生 4 个事件便执行一次捕获
11：每发生 8 个事件便执行一次捕获

位 1:0 **CC1S**：捕获/比较 1 选择 (Capture/Compare 1 selection)

此位域定义通道方向（输入/输出）以及所使用的输入。
00：CC1 通道配置为输出
01：CC1 通道配置为输入，IC1 映射到 TI1 上
10：保留
11：保留

注意：仅当通道关闭时（TIMx_CCER 中的 CC1E = 0），才可向 CC1S 位写入数据。

16.6.5 TIM10/11/13/14 捕获/比较使能寄存器 (TIMx_CCER)

TIM10/11/13/14 capture/compare enable register

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
Reserved												CC1NP	Res.	CC1P	CC1E
												rw		rw	rw

位 15:4　保留，必须保持复位值。

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：CC1NP 必须保持清零。
CC1 通道配置为输入：CC1NP 位与 CC1P 配合使用可定义 TI1FP1 的极性（请参见 CC1P 说明）。

位 2　保留，必须保持复位值。

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：

0：OC1 高电平有效

1：OC1 低电平有效

CC1 通道配置为输入：

CC1P 位可针对触发或捕获操作选择 TI1FP1 和 TI2FP1 的极性。

00：未反相/上升沿触发

电路对 TI1FP1 上升沿敏感（捕获模式），TI1FP1 未反相。

01：反相/下降沿触发

电路对 TI1FP1 下降沿敏感（捕获模式），TI1FP1 反相。

10：保留，不使用此配置

11：未反相/上升沿和下降沿均触发

电路对 TI1FP1 上升沿和下降沿均敏感（捕获模式），TI1FP1 未反相。

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 未激活

1：开启——在相应输出引脚上输出 OC1 信号

CC1 通道配置为输入：

该位决定了输入捕获/比较寄存器 1 (TIMx_CCR1) 是否能够实际捕获到计数器的值。

0：禁止捕获

1：使能捕获

表 82. 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： *与标准 OCx 通道相连的外部 I/O 引脚的状态取决于通道 OCx 的状态以及 GPIO 寄存器。*

16.6.6 TIM10/11/13/14 计数器 (TIMx_CNT)

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	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 15:0 **CNT[15:0]**：计数器值 (Counter value)

16.6.7 TIM10/11/13/14 预分频器 (TIMx_PSC)

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	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 15:0 **PSC[15:0]**：预分频器值 (Prescaler value)

计数器时钟频率 CK_CNT 等于 f_{CK_PSC} / (PSC[15:0] + 1)。

PSC 包含在每次发生更新事件时要装载到实际预分频器寄存器的值。

16.6.8 TIM10/11/13/14 自动重载寄存器 (TIMx_ARR)

TIM10/11/13/14 auto-reload register

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	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 15:0 **ARR[15:0]**：自动重载值 (Auto-reload value)

ARR 为要装载到实际自动重载寄存器的值。

有关 ARR 更新和行为的详细信息，请参见*第 447 页的第 16.4.1 节：时基单元*。

当自动重载值为空时，计数器不工作。

16.6.9 TIM10/11/13/14 捕获/比较寄存器 1 (TIMx_CCR1)

TIM10/11/13/14 capture/compare register 1

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	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 15:0 **CCR1[15:0]**：捕获/比较 1 值 (Capture/Compare 1 value)

如果通道 CC1 配置为输出：

CCR1 为要装载到实际捕获/比较 1 寄存器的值（预装载值）。

如果没有通过 TIMx_CCMR 寄存器中的 OC1PE 位来使能预装载功能，写入的数值会被直接传输至当前寄存器中。否则只在发生更新事件时生效（拷贝到实际起作用的捕获/比较寄存器 1）

实际捕获/比较寄存器中包含要与计数器 TIMx_CNT 进行比较并在 OC1 输出上发出信号的值。

如果通道 CC1 配置为输入：

CCR1 为上一个输入捕获 1 事件 (IC1) 发生时的计数器值。

16.6.10 TIM11 选项寄存器 1 (TIM11_OR)

TIM11 option register 1

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
Reserved														TI1_RMP[1:0]	
														rw	

位 15:2 保留，必须保持复位值。

位 1:0 **TI1_RMP[1:0]**：TIM11 输入 1 重映射功能 (TIM11 Input 1 remapping capability)
由软件置 1 和清零。
00,01,11：TIM11 通道 1 连接到 GPIO （请参见数据手册中的复用功能映射表）。
10：HSE_RTC 时钟（由可编程预分频器分频的 HSE）连接到 TIM11_CH1 输入，用于测量用途

16.6.11 TIM10/11/13/14 寄存器映射

TIMx 寄存器可映射为 16 位可寻址寄存器，如下表所述：

表 83. TIM10/11/13/14 寄存器映射和复位值

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**TIMx_CR1**	Reserved																						CKD [1:0]		ARPE	Reserved				URS	UDIS	CEN
	Reset value																							0	0	0					0	0	0
0x08	**TIMx_SMCR**	Reserved																															
	Reset value																																
0x0C	**TIMx_DIER**	Reserved																														CC1IE	UIE
	Reset value																															0	0
0x10	**TIMx_SR**	Reserved																						CC1OF	Reserved							CC1IF	UIF
	Reset value																							0								0	0
0x14	**TIMx_EGR**	Reserved																														CC1G	UG
	Reset value																															0	0
0x18	**TIMx_CCMR1** *Output compare mode*	Reserved																									OC1M [2:0]			OC1PE	OC1FE	CC1S [1:0]	
	Reset value																										0	0	0	0	0	0	0
	TIMx_CCMR1 *Input capture mode*	Reserved																								IC1F[3:0]				IC1 PSC [1:0]		CC1S [1:0]	
	Reset value																									0	0	0	0	0	0	0	0
0x1C	Reserved																																
0x20	**TIMx_CCER**	Reserved																												CC1NP	Reserved	CC1P	CC1E
	Reset value																													0		0	0
0x24	**TIMx_CNT**	Reserved																CNT[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

表 83. TIM10/11/13/14 寄存器映射和复位值 （续）

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x28	**TIMx_PSC**	Reserved																PSC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x2C	**TIMx_ARR**	Reserved																ARR[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x30	Reserved																																
0x34	**TIMx_CCR1**	Reserved																CCR1[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x38 to 0x4C	Reserved																																
0x50	**TIMx_OR**	Reserved																														TI1_RMP	
	Reset value																															0	0

有关寄存器边界地址的信息，请参见*第 52 页的表 2*。

17 基本定时器（TIM6 和 TIM7）

除非特别说明，否则本节适用于整个 STM32F4xx 系列器件。

17.1 TIM6 和 TIM7 简介

基本定时器 TIM6 和 TIM7 包含一个 16 位自动重载计数器，该计数器由可编程预分频器驱动。

此类定时器不仅可用作通用定时器以生成时基，还可以专门用于驱动数模转换器 (DAC)。实际上，此类定时器内部连接到 DAC 并能够通过其触发输出驱动 DAC。

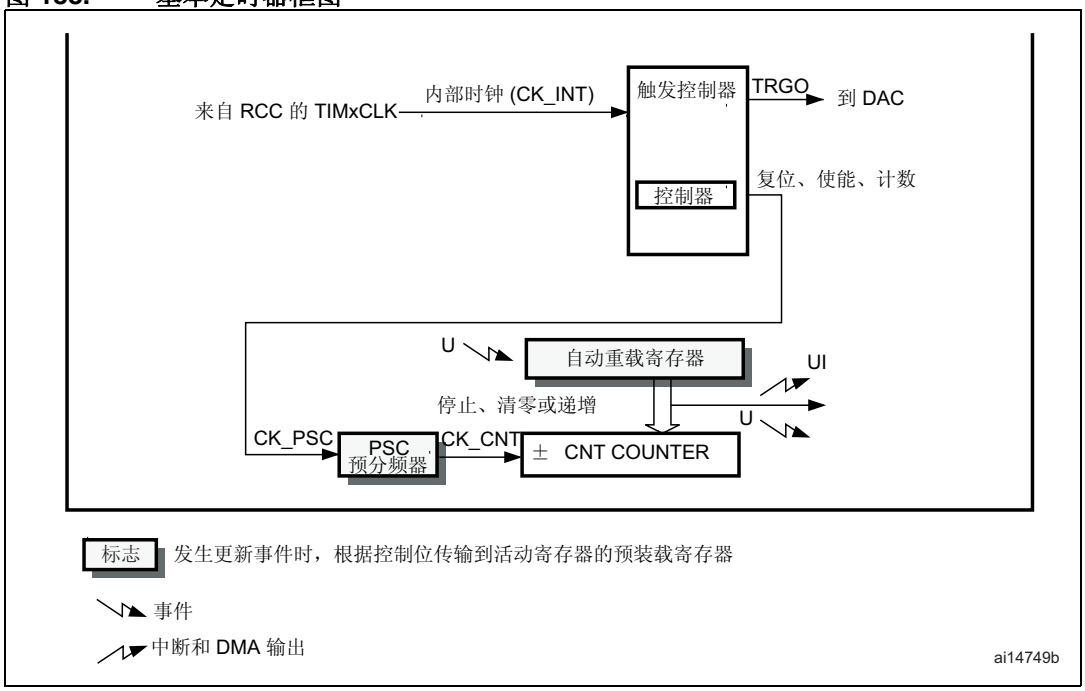
这些定时器彼此完全独立，不共享任何资源。

17.2 TIM6 和 TIM7 的主要特性

基本定时器（TIM6 和 TIM7）的特性包括：

- 16 位自动重载递增计数器
- 16 位可编程预分频器，用于对计数器时钟频率进行分频（即运行时修改），分频系数介于 1 和 65536 之间
- 用于触发 DAC 的同步电路
- 发生如下更新事件时会生成中断/DMA 请求：计数器上溢

图 188. 基本定时器框图



17.3 TIM6 和 TIM7 功能说明

17.3.1 时基单元

可编程定时器的主要模块由一个 16 位递增计数器及其相关的自动重载寄存器组成。计数器的时钟可通过预分频器进行分频。

计数器、自动重载寄存器和预分频器寄存器可通过软件进行读写。即使在计数器运行时也可执行读写操作。

时基单元包括：

- 计数器寄存器 (TIMx_CNT)
- 预分频器寄存器 (TIMx_PSC)
- 自动重载寄存器 (TIMx_ARR)

自动重载寄存器是预装载的。每次尝试对自动重载寄存器执行读写操作时，都会访问预装载寄存器。预装载寄存器的内容既可以直接传送到影子寄存器，也可以在每次发生更新事件 UEV 时传送到影子寄存器，这取决于 TIMx_CR1 寄存器中的自动重载预装载使能位 (ARPE)。当计数器达到上溢值并且 TIMx_CR1 寄存器中的 UDIS 位为 0 时，将发送更新事件。该更新事件也可由软件产生。下文将针对各配置的更新事件的产生进行详细介绍。

计数器由预分频器输出 CK_CNT 提供时钟，仅当 TIMx_CR1 寄存器中的计数器启动位 (CEN) 置 1 时，才会启动计数器。

请注意，实际的计数器使能信号 CNT_EN 在 CEN 置 1 的一个时钟周期后被置 1。

预分频器说明

预分频器可对计数器时钟频率进行分频，分频系数介于 1 和 65536 之间。该预分频器基于 TIMx_PSC 寄存器中的 16 位寄存器所控制的 16 位计数器。由于 TIMx_PSC 控制寄存器有缓冲，因此可对预分频器进行实时更改。而新的预分频比将在下一更新事件发生时被采用。

图 189 和 *图 190* 给出了在预分频比发生实时变化时一些计数器行为的示例。

图 189. 预分频器分频由 1 变为 2 时的计数器时序图

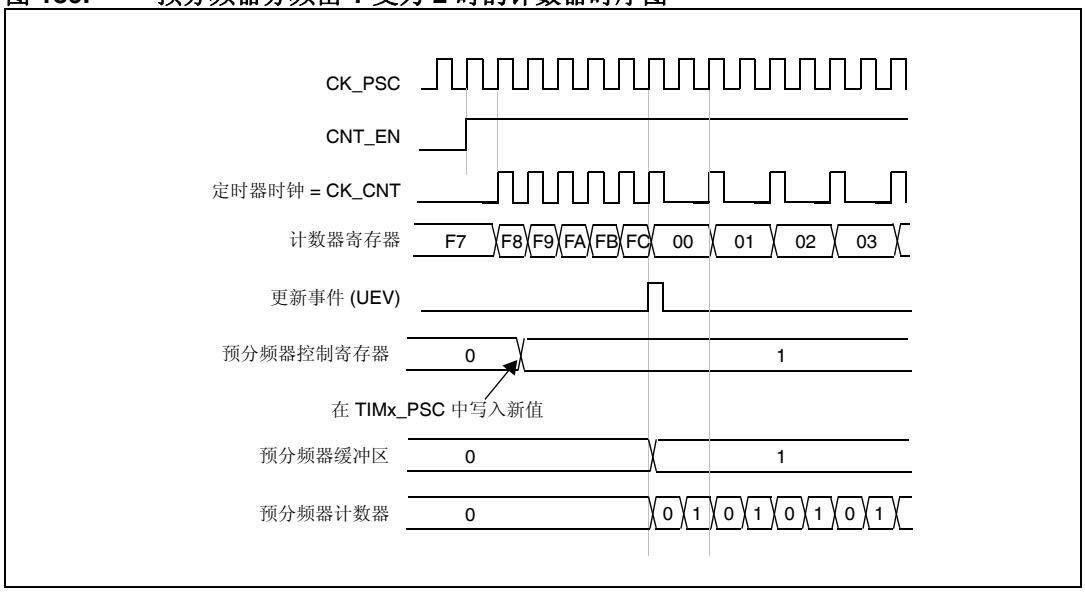
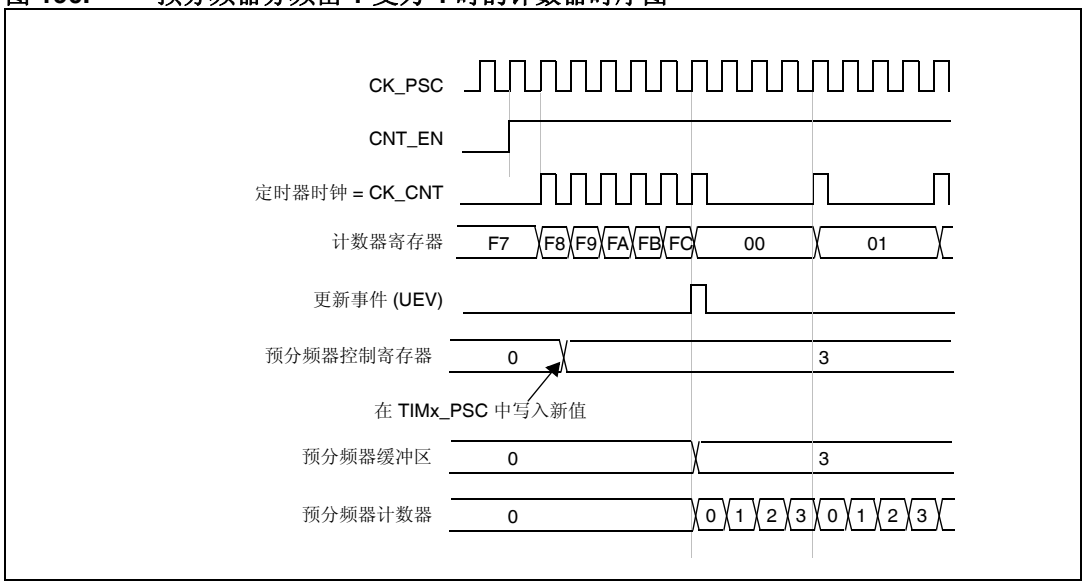


图 190. 预分频器分频由 1 变为 4 时的计数器时序图



17.3.2 计数模式

计数器从 0 计数到自动重载值（TIMx_ARR 寄存器的内容），然后重新从 0 开始计数并生成计数器上溢事件。

每次发生计数器上溢时会生成更新事件，或将 TIMx_EGR 寄存器中的 UG 位置 1（通过软件或使用从模式控制器）也可以生成更新事件。

通过软件将 TIMx_CR1 寄存器中的 UDIS 位置 1 可禁止 UEV 事件。这可避免向预装载寄存器写入新值时更新影子寄存器。这样，直到 UDIS 位中写入 0 前便不会生成任何更新事件，但计数器和预分频器计数器都会重新从 0 开始计数（而预分频比保持不变）。此外，如果 TIMx_CR1 寄存器中的 URS 位（更新请求选择）已置 1，则将 UG 位置 1 会生成更新事件 UEV，但不会将 UIF 标志置 1（因此，不会发送任何中断或 DMA 请求）。

发生更新事件时，将更新所有寄存器且将更新标志（TIMx_SR 寄存器中的 UIF 位）置 1（取决于 URS 位）：

- 使用预装载值（TIMx_PSC 寄存器的内容）重新装载预分频器的缓冲区
- 使用预装载值 (TIMx_ARR) 更新自动重载影子寄存器

以下各图显示了当 TIMx_ARR=0x36 时不同时钟频率下计数器行为的示例。

图 191. 计数器时序图，1 分频内部时钟

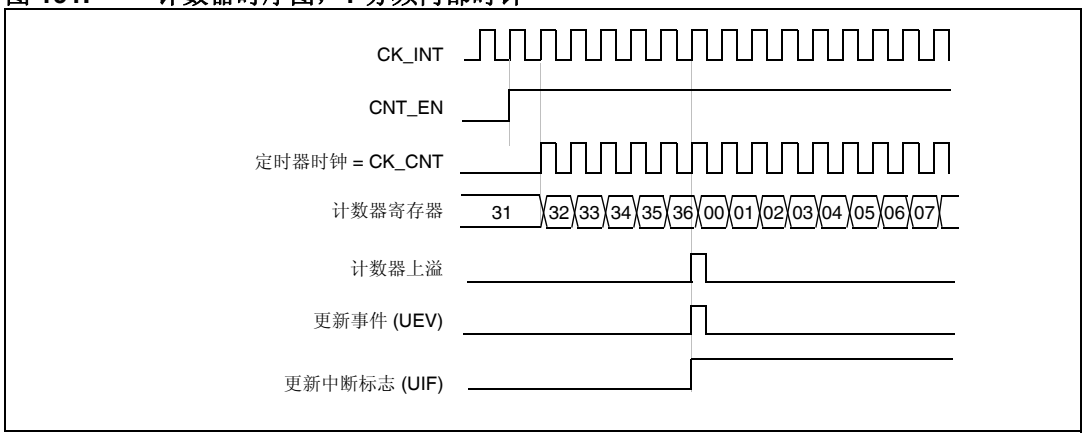


图 192. 计数器时序图，2 分频内部时钟

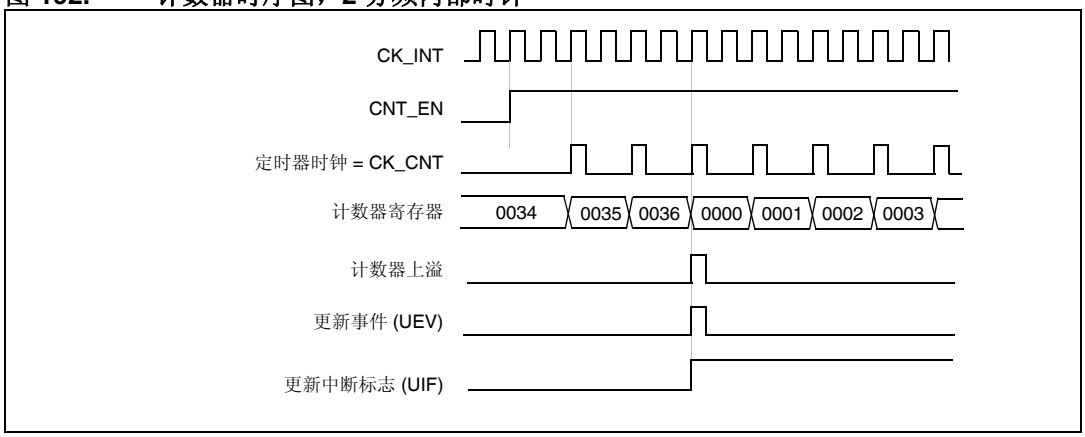


图 193. 计数器时序图，4 分频内部时钟

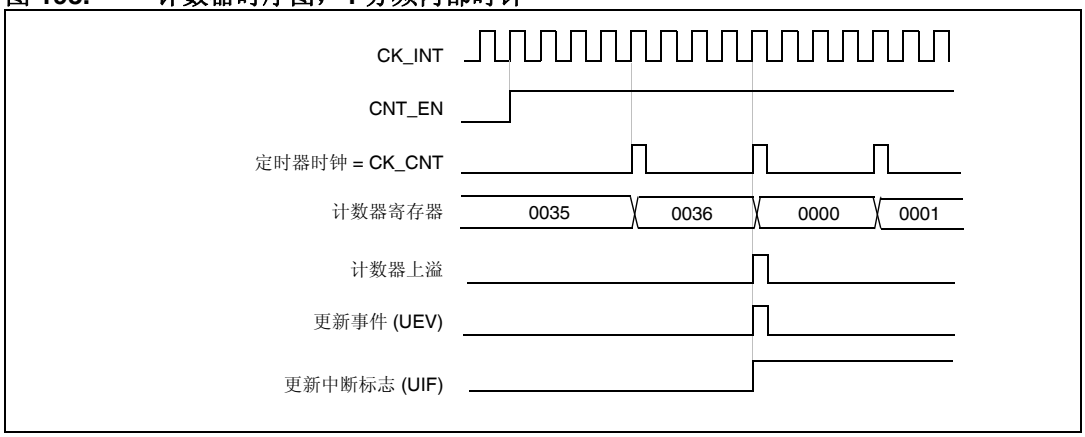


图 194.　计数器时序图，N 分频内部时钟

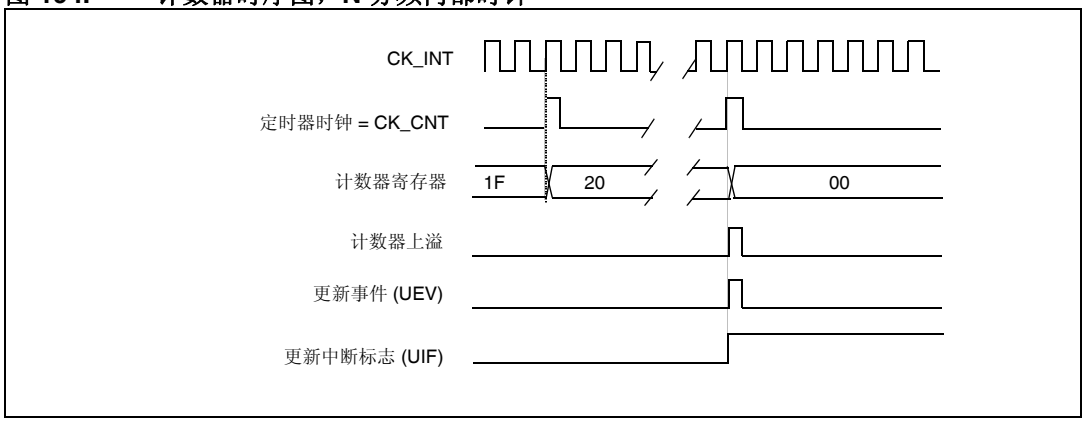


图 195.　计数器时序图，ARPE = 0 时更新事件（TIMx_ARR 未预装载）

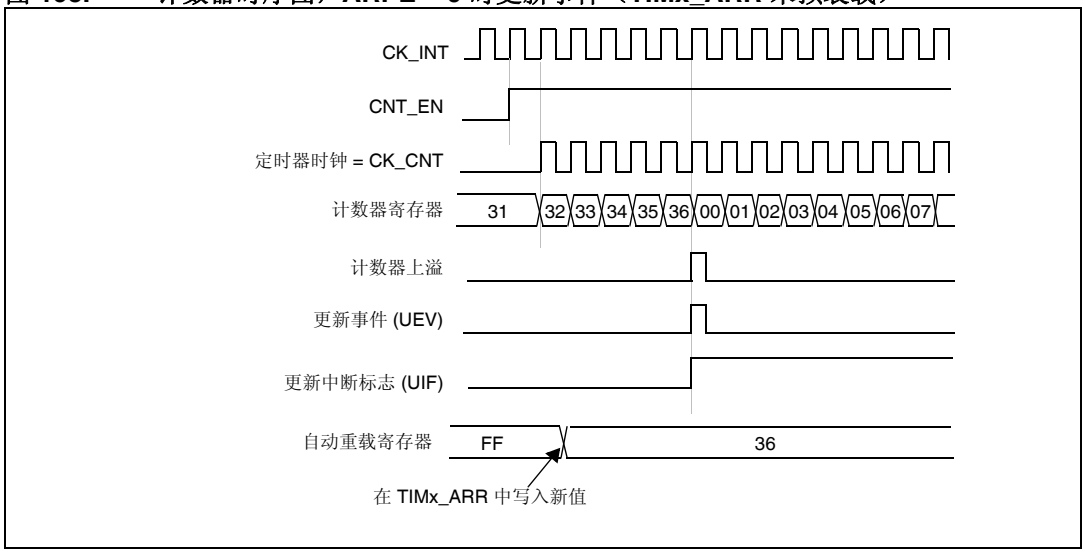
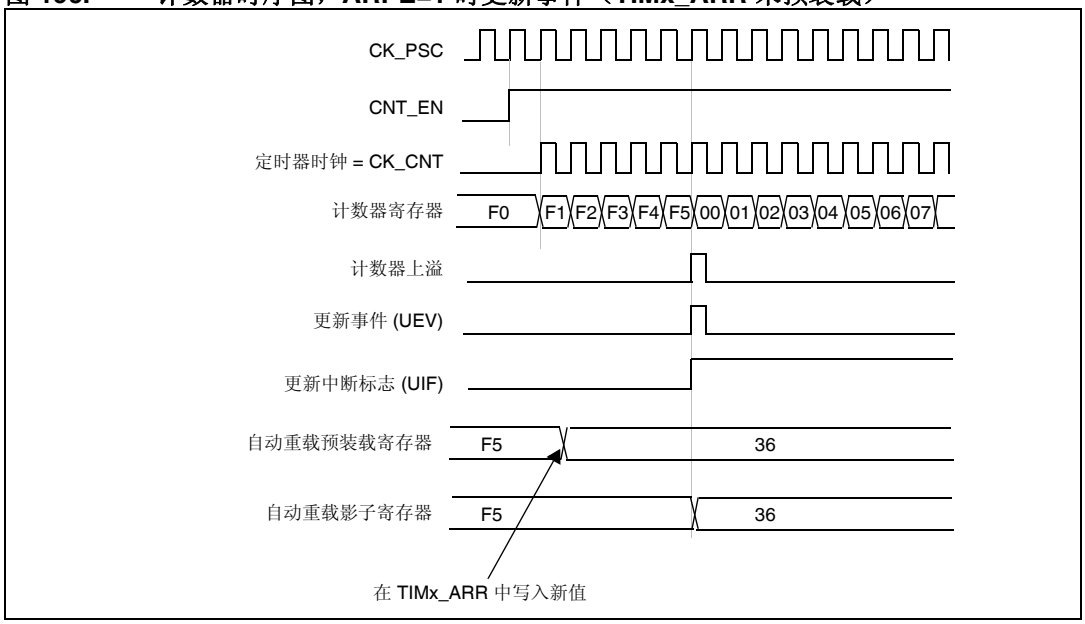


图 196. 计数器时序图，ARPE=1 时更新事件（TIMx_ARR 未预装载）



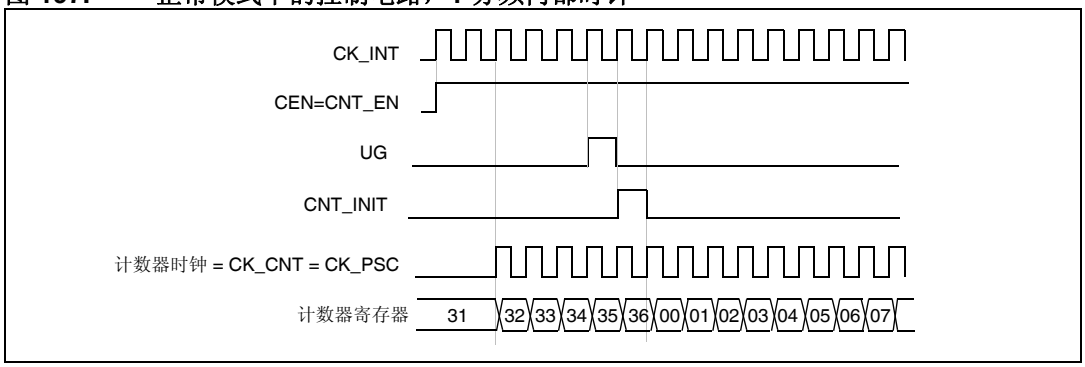
17.3.3 时钟源

计数器时钟由内部时钟 (CK_INT) 源提供。

CEN（TIMx_CR1 寄存器中）和 UG 位（TIMx_EGR 寄存器中）为实际控制位，并且只能通过软件进行更改（保持自动清零的 UG 除外）。当对 CEN 位写入 1 时，预分频器的时钟就由内部时钟 CK_INT 提供。

图 197 显示了正常模式下控制电路与递增计数器的行为（没有预分频的情况下）。

图 197. 正常模式下的控制电路，1 分频内部时钟



17.3.4 调试模式

当微控制器进入调试模式时（Cortex™-M4F 内核停止），TIMx 计数器会根据 DBG 模块中的 DBG_TIMx_STOP 配置位选择继续正常工作或者停止工作。有关详细信息，请参见*第 33.16.2 节： 对定时器、看门狗、bxCAN 和 I^2C 的调试支持*。

17.4 TIM6 和 TIM7 寄存器

有关寄存器说明中使用的缩写，请参见*第 47 页的第 1.1 节*。

外设寄存器的写访问仅支持半字（16 位）或字（32 位）。而读访问可支持字节（8 位）、半字（16 位）或字（32 位）。

17.4.1 TIM6 和 TIM7 控制寄存器 1 (TIMx_CR1)

TIM6&TIM7 control register 1

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
Reserved								ARPE	Reserved			OPM	URS	UDIS	CEN
								rw				rw	rw	rw	rw

位 15:8 保留，必须保持复位值。

位 7 **ARPE**：自动重载预装载使能 (Auto-reload preload enable)

0：TIMx_ARR 寄存器不进行缓冲。

1：TIMx_ARR 寄存器进行缓冲。

位 6:4 保留，必须保持复位值。

位 3 **OPM**：单脉冲模式 (One-pulse mode)

0：计数器在发生更新事件时不会停止计数

1：计数器在发生下一更新事件时停止计数（将 CEN 位清零）。

位 2 **URS**：更新请求源 (Update request source)

此位由软件置 1 和清零，用以选择 UEV 事件源。

0：使能时，所有以下事件都会生成更新中断或 DMA 请求。此类事件包括：

– 计数器上溢/下溢

– 将 UG 位置 1

– 通过从模式控制器生成的更新事件

1：使能时，只有计数器上溢/下溢会生成更新中断或 DMA 请求。

位 1 **UDIS**：更新禁止 (Update disable)

此位由软件置 1 和清零，用以使能/禁止 UEV 事件生成。

0：使能 UEV。更新 (UEV) 事件可通过以下事件之一生成：

– 计数器上溢/下溢

– 将 UG 位置 1

– 通过从模式控制器生成的更新事件

然后更新影子寄存器的值。

1：禁止 UEV。不会生成更新事件，各影子寄存器的值（ARR 和 PSC）保持不变。但如果将 UG 位置 1，或者从从模式控制器接收到硬件复位，则会重新初始化计数器和预分频器。

位 0 **CEN**：计数器使能 (Counter enable)

0：禁止计数器
1：使能计数器

注意：只有事先通过软件将 CEN 位置 1，才可以使用门控模式。而触发模式可通过硬件自动将 CEN 位置 1。

在单脉冲模式下，当发生更新事件时会自动将 CEN 位清零。

17.4.2 TIM6 和 TIM7 控制寄存器 2 (TIMx_CR2)

TIM6&TIM7 control register 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
Reserved									MMS[2:0]			Reserved			
									rw	rw	rw				

位 15:7 保留，必须保持复位值。

位 6:4 **MMS**：主模式选择 (Master mode selection)

这些位用于选择主模式下将要发送到从定时器以实现同步的信息 (TRGO)。这些位的组合如下：
000：**复位**——TIMx_EGR 寄存器中的 UG 位用作触发输出 (TRGO)。如果复位由触发输入生成（从模式控制器配置为复位模式），则 TRGO 上的信号相比实际复位会有延迟。

001：**使能**——计数器使能信号 (CNT_EN) 用作触发输出 (TRGO)。该触发输出可用于同时启动多个定时器，或者控制在一段时间内使能从定时器。计数器使能信号由 CEN 控制位与门控模式下的触发输入的逻辑或运算组合而成。

当计数器使能信号由触发输入控制时，TRGO 上会存在延迟，选择主/从模式时除外（请参见 TIMx_SMCR 寄存器中对 MSM 位的说明）。

010：**更新**——选择更新事件作为触发输出 (TRGO)。例如，主定时器可用作从定时器的预分频器。

位 3:0 保留，必须保持复位值。

17.4.3 TIM6 和 TIM7 DMA/中断使能寄存器 (TIMx_DIER)

TIM6&TIM7 DMA/Interrupt enable register

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
Reserved							UDE	Reserved							UIE
							rw								rw

位 15:9 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 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。

17.4.4 TIM6 和 TIM7 状态寄存器 (TIMx_SR)

TIM6&TIM7 status register

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
Reserved															UIF
															rc_w0

位 15:1 保留，必须保持复位值。

位 0 **UIF**：更新中断标志 (Update interrupt flag)

该位在发生更新事件时通过硬件置 1。但需要通过软件清零。

0：未发生更新。

1：更新中断挂起。该位在以下情况下更新寄存器时由硬件置 1：

– 上溢或下溢并且当 TIMx_CR1 寄存器中 UDIS = 0 时。

– 当由于 TIMx_CR1 寄存器中 URS = 0 且 UDIS = 0 而通过软件使用 TIMx_EGR 寄存器中的 UG 位重新初始化 CNT 时。

17.4.5 TIM6 和 TIM7 事件生成寄存器 (TIMx_EGR)

TIM6&TIM7 event generation register

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
Reserved															UG
															w

位 15:1 保留，必须保持复位值。

位 0 **UG**：更新生成 (Update generation)

该位可通过软件置 1，并由硬件自动清零。

0：不执行任何操作。

1：重新初始化定时器计数器并生成寄存器更新事件。请注意，预分频器计数器也将清零（但预分频比不受影响）。

17.4.6 TIM6 和 TIM7 计数器 (TIMx_CNT)

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	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 15:0 **CNT[15:0]**：计数器值 (Counter value)

17.4.7 TIM6 和 TIM7 预分频器 (TIMx_PSC)

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	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 15:0 **PSC[15:0]**：预分频器值 (Prescaler value)

计数器时钟频率 CK_CNT 等于 f_{CK_PSC} / (PSC[15:0] + 1)。

PSC 包含在每次发生更新事件时要装载到实际预分频器寄存器的值。

17.4.8 TIM6 和 TIM7 自动重载寄存器 (TIMx_ARR)

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	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 15:0 **ARR[15:0]**：自动重载值 (Auto-reload value)

ARR 为要装载到实际自动重载寄存器的值。

有关 ARR 更新和行为的详细信息，请参见*第 17.3.1 节：第 484 页的时基单元*。

当自动重载值为空时，计数器不工作。

17.4.9 TIM6 和 TIM7 寄存器映射

TIMx 寄存器可映射为 16 位可寻址寄存器，如下表所述：

表 84. TIM6 和 TIM7 寄存器映射和复位值

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**TIMx_CR1**	Reserved																								ARPE	Reserved			OPM	URS	UDIS	CEN
	Reset value																									0				0	0	0	0
0x04	**TIMx_CR2**	Reserved																									MMS[2:0]			Reserved				
	Reset value																										0	0	0				
0x08	Reserved																																
0x0C	**TIMx_DIER**	Reserved																							UDE	Reserved							UIE
	Reset value																								0								0
0x10	**TIMx_SR**	Reserved																															UIF
	Reset value																																0
0x14	**TIMx_EGR**	Reserved																															UG
	Reset value																																0
0x18	Reserved																																
0x1C	Reserved																																
0x20	Reserved																																
0x24	**TIMx_CNT**	Reserved																CNT[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x28	**TIMx_PSC**	Reserved																PSC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x2C	**TIMx_ARR**	Reserved																ARR[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

有关寄存器边界地址的信息，请参见*第 52 页的表 2*。

18 独立看门狗 (IWDG)

除非特别说明，否则本部分适用于整个 STM32F4xx 系列。

18.1 IWDG 简介

此器件具有两个嵌入式看门狗外设，具有安全性高、定时准确及使用灵活的优点。两个看门狗外设（独立和窗口）均可用于检测并解决由软件错误导致的故障；当计数器达到给定的超时值时，触发一个中断（仅适用于窗口型看门狗）或产生系统复位。

独立看门狗 (IWDG) 由其专用低速时钟 (LSI) 驱动，因此即便在主时钟发生故障时仍然保持工作状态。窗口看门狗 (WWDG) 时钟由 APB1 时钟经预分频后提供，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。

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。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。有关窗口看门狗的详细信息，请参见*第 499 页的第 19 节*。

18.2 IWDG 主要特性

- 自由运行递减计数器
- 时钟由独立 RC 振荡器提供（可在待机和停止模式下运行）
- 当递减计数器值达到 0x000 时产生复位（如果看门狗已激活）

18.3 IWDG 功能说明

图 198 给出了独立看门狗模块的功能框图。

当通过对关键字寄存器 (IWDG_KR) 写入值 0xCCCC 启动独立看门狗时，计数器开始从复位值 0xFFF 递减计数。当计数器计数到终值 (0x000) 时会产生一个复位信号（IWDG 复位）。

任何时候将关键字 0xAAAA 写到 IWWDG_KR 寄存器中，IWDG_RLR 的值就会被重载到计数器，从而避免产生看门狗复位。

18.3.1 硬件看门狗

如果通过器件选项位使能“硬件看门狗”功能，上电时将自动使能看门狗；如果在计数器计数结束前，若软件没有向关键字寄存器写入相应的值，则系统会产生复位。

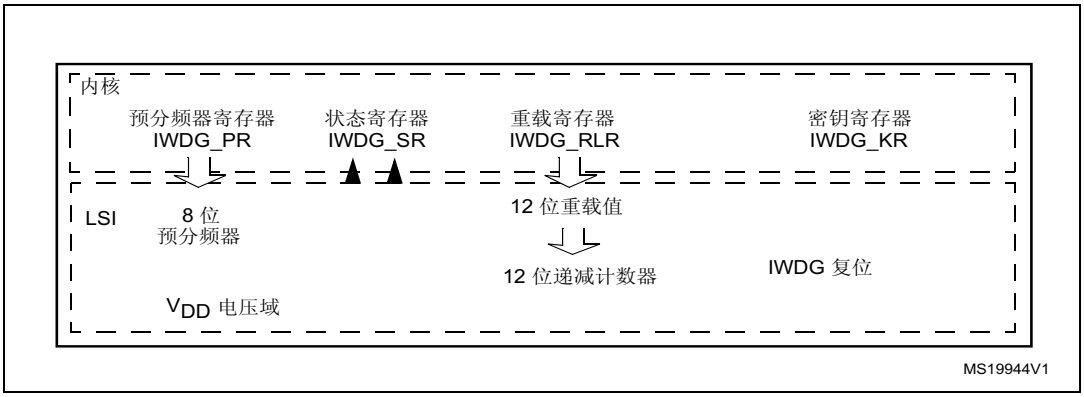
18.3.2 寄存器访问保护

IWDG_PR 和 IWDG_RLR 寄存器具有写访问保护。若要修改寄存器，必须首先对 IWDG_KR 寄存器写入代码 0x5555。而写入其他值则会破坏该序列，从而使寄存器访问保护再次生效。这意味着重装载操作（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。

18.3.3 调试模式

当微控制器进入调试模式时（Cortex™-M4F 内核停止），IWDG 计数器会根据 DBG 模块中的 DBG_IWDG_STOP 配置位选择继续正常工作或者停止工作。有关详细信息，请参见*第 33.16.2 节：对定时器、看门狗、bxCAN 和 I^2C 的调试支持*。

图 198. 独立看门狗框图



注意：看门狗功能由 V_{DD} 电压域供电，在停止模式和待机模式下仍能工作。

表 85. 32 kHz (LSI) 频率条件下 IWDG 超时周期的最小值/最大值 (1)

预分频器	PR[2:0] 位	最短超时 (ms) RL[11:0]= 0x000	最长超时 (ms) RL[11:0]= 0xFFF
/4	0	0.125	512
/8	1	0.25	1024
/16	2	0.5	2048
/32	3	1	4096
/64	4	2	8192
/128	5	4	16384
/256	6	8	32768

1. 这些时间均针对 32 kHz 时钟给出。实际上，MCU 内部的 RC 频率会在 30kHz 到 60kHz 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。

18.4 IWDG 寄存器

有关寄存器说明中使用的缩写，请参见*第 47 页的第 1.1 节*。

外设寄存器可支持半字（16 位）或字（32 位）访问。

18.4.1 关键字寄存器 (IWDG_KR)

Key register

偏移地址：0x00

复位值：0x0000 0000（通过待机模式复位）

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

位 31:16 保留，必须保持复位值。

位 15:0 **KEY[15:0]**：键值 (Key value)（只写位，读为 0000h）

必须每隔一段时间便通过软件对这些位写入键值 AAAAh，否则当计数器计数到 0 时，看门狗会产生复位。

写入键值 5555h 可使能对 IWDG_PR 和 IWDG_RLR 寄存器的访问（请参见*第 18.3.2 节*）

写入键值 CCCCh 可启动看门狗（选中硬件看门狗选项的情况除外）

18.4.2 预分频器寄存器 (IWDG_PR)

Prescaler register

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

位 31:3 保留，必须保持复位值。

位 2:0 **PR[2:0]**：预分频器 (Prescaler divider)

这些位受写访问保护，请参见*第 18.3.2 节*。通过软件设置这些位来选择计数器时钟的预分频因子。若要更改预分频器的分频系数，IWDG_SR 的 PVU 位必须为 0。

000：4 分频
001：8 分频
010：16 分频
011：32 分频
100：64 分频
101：128 分频
110：256 分频
111：256 分频

注意：读取该寄存器会返回 VDD 电压域的预分频器值。如果正在对该寄存器执行写操作，则读取的值可能不是最新的/有效的。因此，只有在 IWDG_SR 寄存器中的 PVU 位为 0 时，从寄存器读取的值才有效。

18.4.3 重载寄存器 (IWDG_RLR)

Reload register

偏移地址：0x08

复位值：0x0000 0FFF（待机模式时复位）

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

位 31:12 保留，必须保持复位值。

位 11:0 **RL[11:0]**：看门狗计数器重载值 (Watchdog counter reload value)

这些位受写访问保护，请参见*第 18.3.2 节*。这个值由软件设置，每次对 IWDR_KR 寄存器写入值 AAAAh 时，这个值就会重装载到看门狗计数器中。之后，看门狗计数器便从该装载的值开始递减计数。超时周期由该值和时钟预分频器共同决定。请参见*表 85*。

若要更改重载值，IWDG_SR 中的 RVU 位必须为 0。

注意：读取该寄存器会返回 VDD 电压域的重载值。如果正在对该寄存器执行写操作，则读取的值可能不是最新的/有效的。因此，只有在 IWDG_SR 寄存器中的 RVU 位为 0 时，从寄存器读取的值才有效。

18.4.4 状态寄存器 (IWDG_SR)

Status register

偏移地址：0x0C

复位值：0x0000 0000（待机模式时不复位）

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

位 31:2 保留，必须保持复位值。

位 1 **RVU**：看门狗计数器重载值更新 (Watchdog counter reload value update)

可通过硬件将该位置 1 以指示重载值正在更新。当在 V_{DD} 电压域下完成重载值更新操作后（需要多达 5 个 RC 40 kHz 周期），会通过硬件将该位复位。

重载值只有在 RVU 位为 0 时才可更新。

位 0 **PVU**：看门狗预分频器值更新 (Watchdog prescaler value update)

可通过硬件将该位置 1 以指示预分频器值正在更新。当在 V_{DD} 电压域下完成预分频器值更新操作后（需要多达 5 个 RC 40 kHz 周期），会通过硬件将该位复位。

预分频器值只有在 PVU 位为 0 时才可更新。

注意：如果应用使用多个重载值或预分频器值，则必须等到 RVU 位被清零后才能更改重载值，而且必须等到 PVU 位被清零后才能更改预分频器值。但是，在更新预分频器和/或重载值之后，则无需等到 RVU 或 PVU 复位后再继续执行代码（即便进入低功耗模式，也会继续执行写操作至完成）

18.4.5 IWDG 寄存器映射

下表提供了 IWDG 寄存器映射和复位值。

表 86. IWDG 寄存器映射和复位值

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**IWDG_KR**	Reserved																KEY[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	**IWDG_PR**	Reserved																													PR[2:0]		
	Reset value																														0	0	0
0x08	**IWDG_RLR**	Reserved																RL[11:0]															
	Reset value																	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x0C	**IWDG_SR**	Reserved																														RVU	PVU
	Reset value																															0	0

有关寄存器边界地址的信息，请参见*第 52 页的表 2*。

19 窗口看门狗 (WWDG)

除非特别说明，否则本部分适用于整个 STM32F4xx 系列。

19.1 WWDG 简介

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造成的应用程序背离正常的运行序列而产生的软件故障。除非递减计数器的值在 T6 位变成 0 前被刷新，看门狗电路在达到预置的时间周期时，会产生一个 MCU 复位。如果在递减计数器达到窗口寄存器值之前刷新控制寄存器中的 7 位递减计数器值，也会产生 MCU 复位。这意味着必须在限定的时间窗口内刷新计数器。

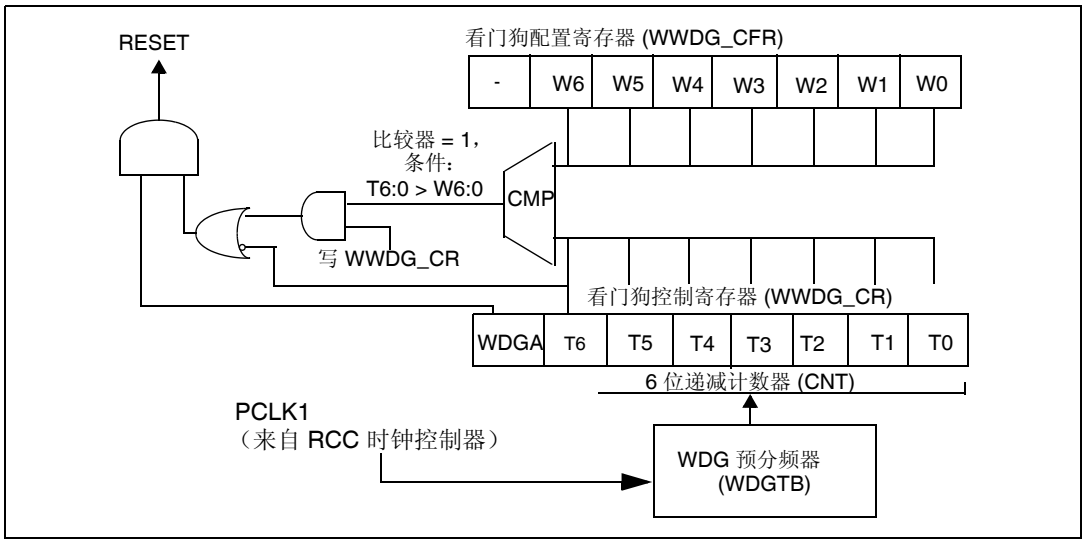
19.2 WWDG 主要特性

- 可编程的自由运行递减计数器
- 复位条件
  - 当递减计数器值小于 0x40 时复位（如果看门狗已激活）
  - 在窗口之外重载递减计数器时复位（如果看门狗已激活）（请参见*图 200*）
- 提前唤醒中断 (EWI)：当递减计数器等于 0x40 时触发（如果已使能且看门狗已激活）

19.3 WWDG 功能说明

如果激活看门狗（WWDG_CR 寄存器中的 WDGA 位置 1），则当 7 位递减计数器（T[6:0] 位）从 0x40 滚动到 0x3F（T6 已清零）时会引发复位。当计数器值大于窗口寄存器中所存储的值时，如果软件重载计数器，则会产生复位。

图 199. 看门狗框图



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。只有当计数器值低于窗口寄存器值时，才能执行此操作。存储在 WWDG_CR 寄存器中的值必须介于 0xFF 和 0xC0 之间：

使能看门狗

在系统复位后，看门狗总是处于关闭状态。可通过设置 WWDG_CR 寄存器中的 WDGA 位来使能看门狗，之后除非执行复位操作，否则不能再次关闭。

控制递减计数器

递减计数器处于自由运行状态：即使禁止看门狗，递减计数器仍继续递减计数。当使能看门狗时，必须将 T6 位置 1，以防止立即复位。

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（请参见*图 200*）。配置寄存器 (WWDG_CFR) 包含窗口的上限：为防止发生复位，当递减计数器的值低于窗口寄存器值且大于 0x3F 时必须重载。*图 200* 介绍了窗口看门狗的工作过程。

注意： *可使用 T6 位产生软件复位（将 WDGA 位置 1 并将 T6 位清零）。*

看门狗中断高级特性

如果在产生实际复位之前必须执行特定的安全操作或数据记录，则可使用提前唤醒中断 (EWI)。通过设置 WWDG_CFR 寄存器中的 EWI 位使能 EWI 中断。当递减计数器的值为 0x40 时，将生成 EWI 中断。在复位器件之前，可以使用相应的中断服务程序 (ISR) 来触发特定操作（例如通信或数据记录）。

在某些应用中，可以使用 EWI 中断来管理软件系统检查和/或系统恢复/功能退化，而不会生成 WWDG 复位。在这种情况下，相应的中断服务程序 (ISR) 可用来重载 WWDG 计数器以避免 WWDG 复位，然后再触发所需操作。

通过将 0 写入 WWDG_SR 寄存器中的 EWIF 位来清除 EWI 中断。

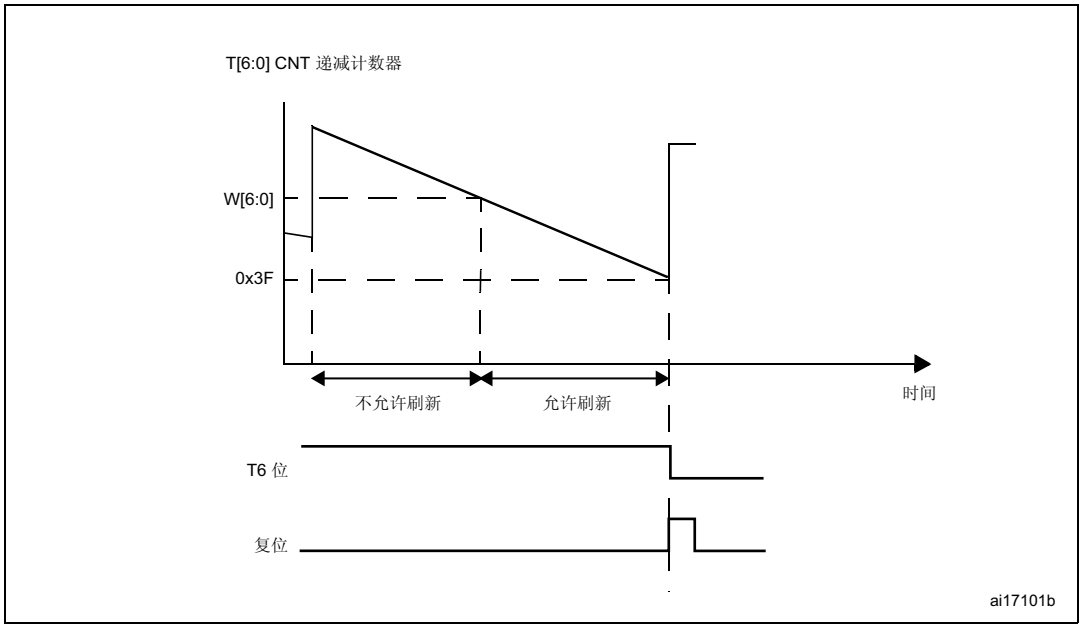
注意： *当由于在更高优先级任务中有系统锁定而无法使用 EWI 中断时，最终会产生 WWDG 复位。*

19.4 如何设置看门狗超时

可以使用*图 200* 中的公式来计算 WWDG 超时。

警告： **写入 WWDG_CR 寄存器时，始终将 1 写入 T6 位，以避免生成立即复位。**

图 200. 窗口看门狗时序图



超时值的计算公式如下：

$$t_{WWDG} = t_{PCLK1} \times 4096 \times 2^{WDGTB} \times (t[5:0] + 1) \quad (ms)$$

其中：

t_{WWDG}：WWDG 超时

t_{PCLK1}：APB1 时钟周期，以 ms 为测量单位

有关 T_{WWDG} 的最小值和最大值，请参见下表。

表 87. 30 MHz (f_{PCLK1}) 时的超时值

预分频器	WDGTB	最小超时 (µs) T[5:0] = 0x00	最大超时 (ms) T[5:0] = 0x3F
1	0	136.53	8.74
2	1	273.07	17.48
4	2	546.13	34.95
8	3	1092.27	69.91

19.5 调试模式

当微控制器进入调试模式时（Cortex™-M4F 内核停止），WWDG 计数器会根据 DBG 模块中的 DBG_WWDG_STOP 配置位选择继续正常工作或者停止工作。有关详细信息，请参见 *第 33.16.2 节：对定时器、看门狗、bxCAN 和 I^2C 的调试支持*。

19.6 WWDG 寄存器

有关寄存器说明中使用的缩写，请参见*第 47 页的第 1.1 节*。

外设寄存器可支持半字（16 位）或字（32 位）访问。

19.6.1 控制寄存器 (WWDG_CR)

Status register

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

位 31:8 保留，必须保持复位值。

位 7 **WDGA**：激活位 (Activation bit)

此位由软件置 1，只有复位后才由硬件清零。当 WDGA = 1 时，看门狗可产生复位。

0: 禁止看门狗

1: 使能看门狗

位 6:0 **T[6:0]**：7 位计数器 (MSB 到 LSB)

这些位用来存储看门狗计数器的值。它每隔 (4096 x 2^{WDGTB}) PCLK1 个周期递减一次。当它从 0x40 滚动到 0x3F（T6 清零）时会产生复位。

19.6.2 配置寄存器 (WWDG_CFR)

Configuration register

偏移地址：0x04

复位值：0x0000 007F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						EWI	WDGTB[1:0]		W[6:0]						
						rs	rw		rw						

位 31:10 保留，必须保持复位值。

位 9 **EWI**：提前唤醒中断 (Early wakeup interrupt)
置 1 后，只要计数器值达到 0x40 就会产生中断。此中断只有在复位后才由硬件清零。

位 8:7 **WDGTB[1:0]**：定时器时基 (Timer base)
可按如下方式修改预分频器的时基：
00：CK 计数器时钟 (PCLK1 div 4096) 分频器 1
01：CK 计数器时钟 (PCLK1 div 4096) 分频器 2
10:CK 计数器时钟 (PCLK1 div 4096) 分频器 4
11:CK 计数器时钟 (PCLK1 div 4096) 分频器 8

位 6:0 **W[6:0]**：7 位窗口值 (7-bit window value)
这些位包含用于与递减计数器进行比较的窗口值。

19.6.3 状态寄存器 (WWDG_SR)

Status register

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
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															EWIF
															rc_w0

位 31:1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。它必须由软件通过写入 0 来清零。写入 1 不起作用。如果不使能中断，此位也会被置 1。

19.6.4 WWDG 寄存器映射

下表提供了 WWDG 寄存器映射和复位值。

表 88. WWDG 寄存器映射和复位值

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**WWDG_CR**	Reserved																								WDGA	T[6:0]						
	Reset value																									0	1	1	1	1	1	1	1
0x04	**WWDG_CFR**	Reserved																						EWI	WDGTB1	WDGTB0	W[6:0]						
	Reset value																							0	0	0	1	1	1	1	1	1	1
0x08	**WWDG_SR**	Reserved																															EWIF
	Reset value																																0

有关寄存器边界地址的信息，请参见*第 52 页的表 2*。

20 加密处理器 (CRYP)

除非特别说明，否则本部分适用于整个 STM32F4xx 系列器件。

20.1 CRYP 简介

借助加密处理器，可使用 DES、三重 DES 或 AES（128、192 或 256）算法对数据进行加密或解密。加密处理器完全兼容下列标准：

- 联邦信息处理标准出版物“FIPS PUB 46-3，1999 年 10 月 25 日”规定的数据加密标准 (DES) 和三重 DES (TDES)。它遵循美国国家标准协会 (ANSI) X9.52 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)。

CRYP 处理器可在电子密码本 (ECB) 模式或加密分组链接 (CBC) 模式下使用 DES 和 TDES 算法执行数据加密和解密。

CRYP 外设为 32 位 AHB2 外设。它支持传入数据和已处理数据的 DMA 传输，并具有输入和输出 FIFO（分别为 8 个字深）。

20.2 CRYP 主要特性

- 适用于 AES、DES 和 TDES 加密和解密操作
- AES
  - 支持 ECB、CBC、CTR、CCM 和 GCM 链接算法（CCM 和 GCM 仅适用于 STM32F42xxx 和 STM32F43xxx）
  - 支持 128 位、192 位和 256 位密钥
  - 支持在 CBC、CTR、CCM 和 GCM 模式下使用的 4 × 32 位初始化向量 (IV)

表 89. 处理 128 位块所需的周期数（STM32F405xx/07xx 和 STM32F415xx/17xx）

算法/密钥大小	ECB	CBC	CTR
128b	14	14	14
192b	16	16	16
256b	18	18	18

表 90. 处理 128 位块所需的周期数（STM32F42xxx 和 STM32F43xxx）

算法/密钥大小	ECB	CBC	CTR	GCM				CCM			
				初始化	标头	有效负载	标签	初始化	标头	有效负载	标签
128b	14	14	14	24	10	14	14	12	14	25	14
192b	16	16	16	28	10	16	16	14	16	29	16
256b	18	18	18	32	10	18	18	16	18	33	18

- DES/TDES
  - 直接执行简单 DES 算法（使用单一密钥 K1）
  - 支持 ECB 和 CBC 链接算法
  - 支持 64 位、128 位和 192 位密钥（包括奇偶校验）
  - 支持在 CBC 模式下使用的 2 × 32 位初始化向量 (IV)
  - 使用 DES 处理一个 64 位块需要 16 个 HCLK 周期
  - 使用 TDES 处理一个 64 位块需要 48 个 HCLK 周期
- DES/TDES 和 AES 的共同点
  - 采用 IN 和 OUT FIFO（各具有 8 个字深和 32 位宽，与 4 个 DES 块或 2 个 AES 块相对应）
  - 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已处理数据）
  - 采用数据交换逻辑，支持 1 位、8 位、16 位或 32 位数据

20.3 CRYP 功能说明

加密处理器可实现三重 DES（即 TDES，同样支持 DES）内核和 AES 加密内核。*第 20.3.1 节*和*第 20.3.2 节*提供有关内核的详细信息。

由于 TDES 和 AES 算法使用块密码，因此，加密前需对不完整的输入数据块进行填充（应将额外的位附加到数据串的尾端）。解密后需要丢弃填充项。硬件不处理填充操作，需要通过软件进行处理。

图 201 显示了加密处理器的框图。

图 201. 框图（STM32F405xx/07xx 和 STM32F415xx/17xx）

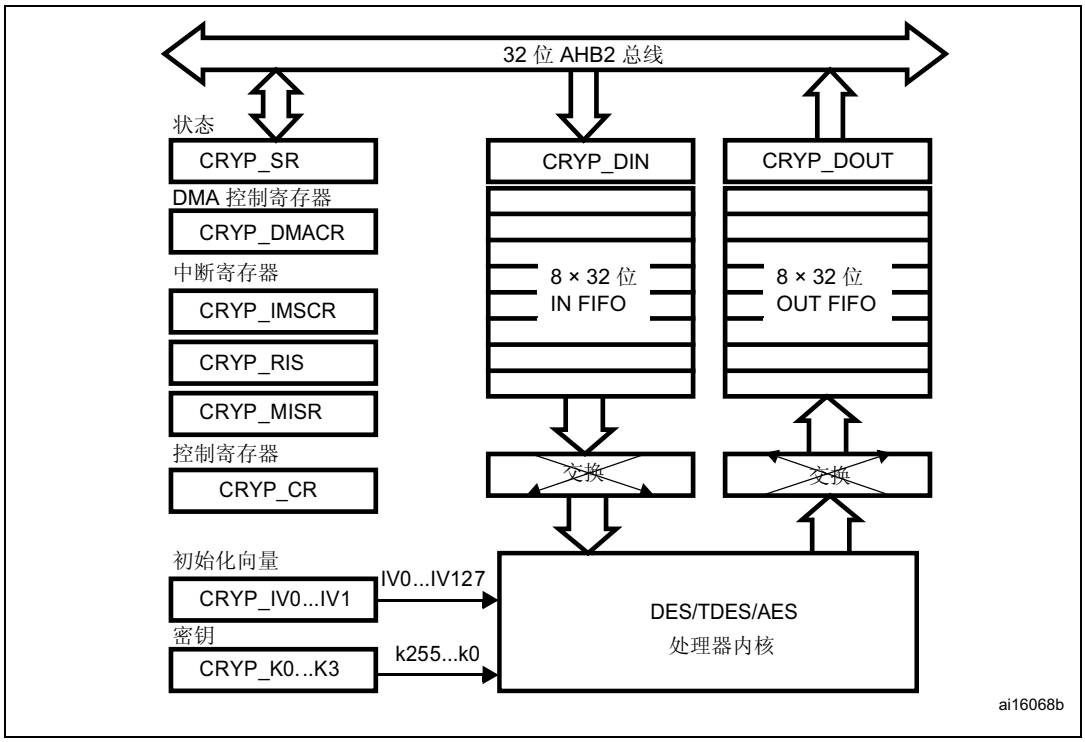
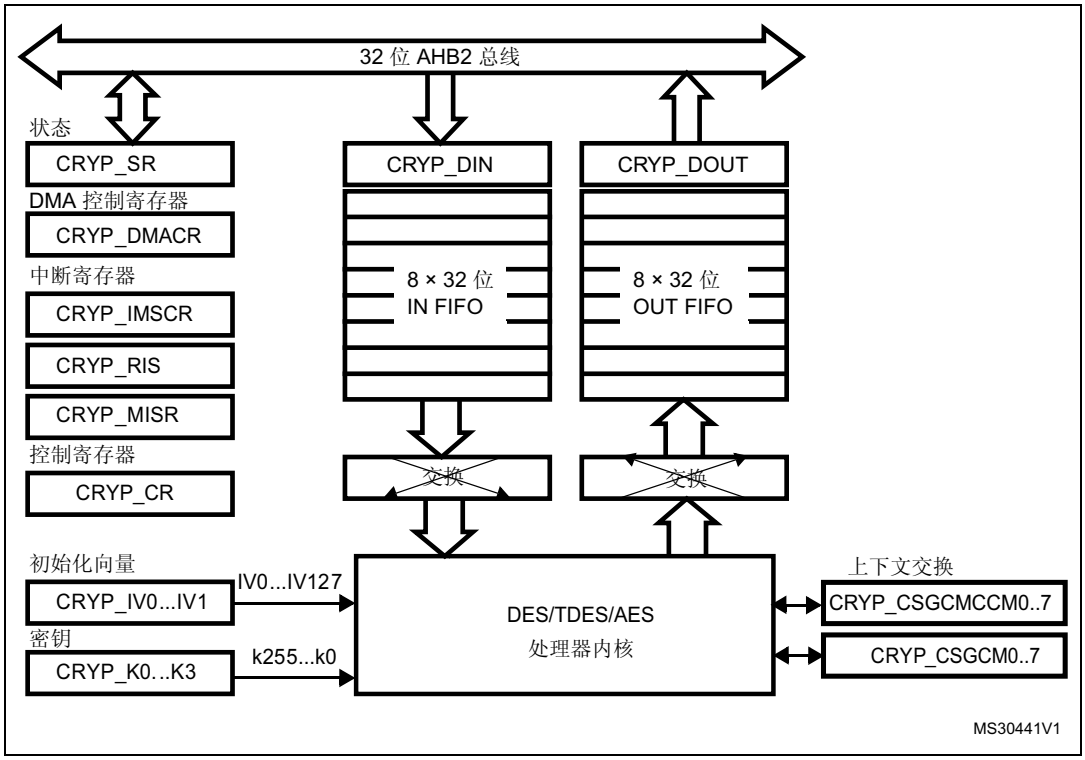


图 202. 框图（STM32F42xxx 和 STM32F43xxx）



20.3.1 DES/TDES 加密内核

DES/三重 DES 加密内核由三部分组成：

- DES 算法 (DEA)
- 多个密钥（DES 算法使用 1 个密钥，TDES 算法使用 1 到 3 个密钥）
- 初始化向量（在 CBC 模式中使用）

TDES 中涉及的基本处理如下：输入块通过 DEA 读取，并使用第一个密钥 K1 加密（TDES 模式中不使用 K0）。然后，输出使用第二个密钥 K2 解密，再使用第三个密钥 K3 加密。密钥由使用的算法决定：

- DES 模式：密钥 = [K1]
- TDES 模式：密钥 = [K3 K2 K1]

其中 Kx = [KxR KxL]，R = 右，L = 左

生成的输出块用于计算密文，具体取决于使用的模式。

注意，中间 DEA 阶段的输出始终不会在加密边界外显示。

TDES 允许使用三种不同的密钥选项：

- 三个独立的密钥

  第一个选项指定所有的密钥均是独立的，即 K1、K2 和 K3 均是独立的。FIPS PUB 46-3 – 1999（以及 ANSI X9.52 – 1998）将该选项称为“密钥选项 1”，并将这种 TDES 称为“3 密钥 TDES”。

- 两个独立的密钥

  第二个选项指定 K1 和 K2 是独立的，而 K3 等于 K1，即 K1 和 K2 独立，而 K3 = K1。FIPS PUB 46-3 – 1999（以及 ANSI X9.52 – 1998）将该选项称为“密钥选项 2”，并将这种 TDES 称为“2 密钥 TDES”。

- 三个相同的密钥

  第三个选项指定 K1、K2 和 K3 是相同的，即 K1 = K2 = K3。FIPS PUB 46-3 – 1999（以及 ANSI X9.52 – 1998）将第三个选项称为“密钥选项 3”。这种“1 密钥”TDES 与单独 DES 相同。

FIPS PUB 46-3 – 1999（以及 ANSI X9.52-1998）对 TDEA（TDES 算法）所提供的四种操作模式中涉及的处理过程进行了详尽的解释，这四种模式分别为：TDES-ECB 加密、TDES-ECB 解密、TDES-CBC 加密和 TDES-CBC 解密。
本参考手册仅对每种模式进行了简要解释。

DES 和 TDES 电子密码本 (DES/TDES-ECB) 模式

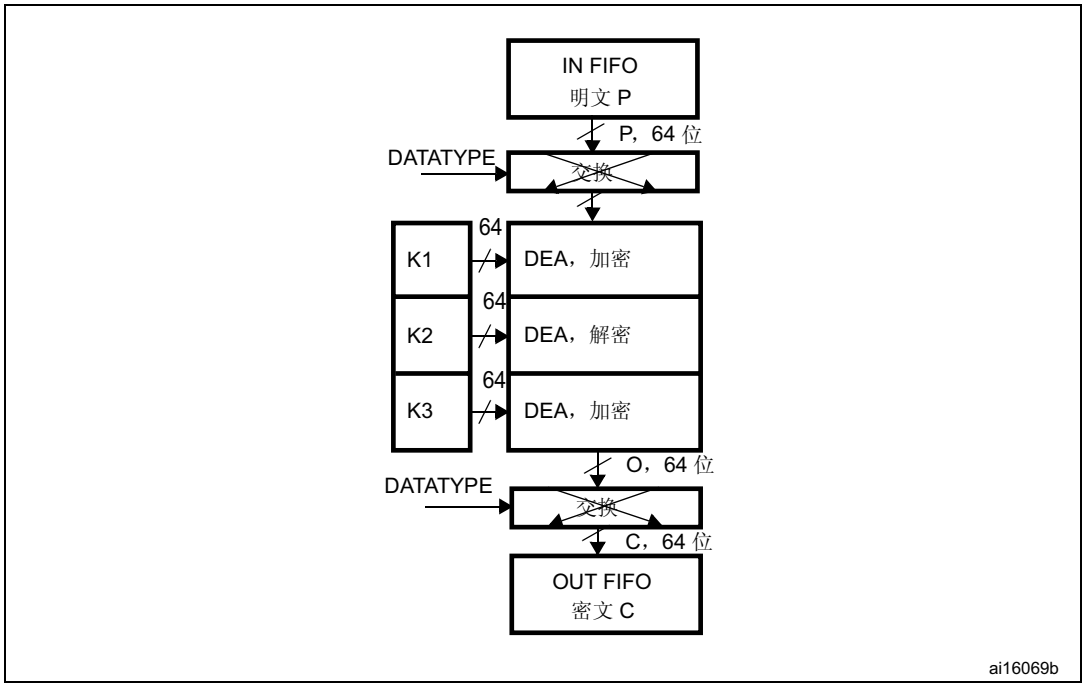
- DES/TDES-ECB 模式加密

  图 203 介绍了 DES 和 TDES 电子密码本 (DES/TDES-ECB) 模式中的加密。64 位明文数据块 (P) 经过位/字节/半字交换（请参见*第 522 页的第 20.3.3 节：数据类型*）后作为输入块 (I)。输入块通过 DEA 在加密状态下使用 K1 进行加密处理。上述处理过程的输出会直接反馈到 DEA 的输入，在解密状态下使用 K2 执行 DES。上述处理过程的输出会直接反馈到 DEA 的输入，在加密状态下使用 K3 执行 DES。生成的 64 位输出块 (O) 在执行位/字节/半字交换之后，以密文 (C) 形式推入 OUT FIFO。

- DES/TDES-ECB 模式解密

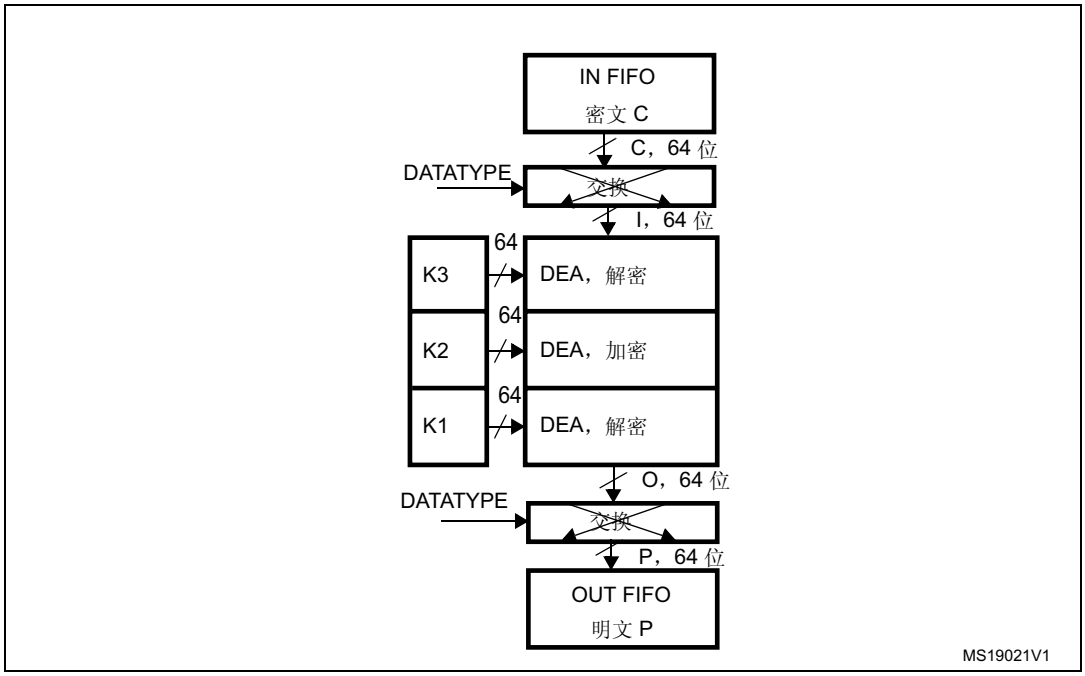
  图 204 介绍了 DES/TDES-ECB 解密。64 位官方块 (C) 经过位/字节/半字交换后，作为输入块 (I)。该密钥序列与加密过程中使用的密钥序列相反。输入块通过 DEA 在解密状态下使用 K3 进行解密处理。上述处理过程的输出会直接反馈到 DEA 的输入，在加密状态下使用 K2 执行 DES。新结果会直接反馈到 DEA 的输入，在解密状态下使用 K1 执行 DES。生成的 64 位输出块 (O) 在进行位/字节/半字交换后，产生明文 (P)。

图 203. DES/TDES-ECB 模式加密



1. K：密钥；C：密文；I：输入块；O：输出块；P：明文。

图 204. DES/TDES-ECB 模式解密



1. K：密钥；C：密文；I：输入块；O：输出块；P：明文。

DES 和 TDES 密码块链接 (DES/TDES-CBC) 模式

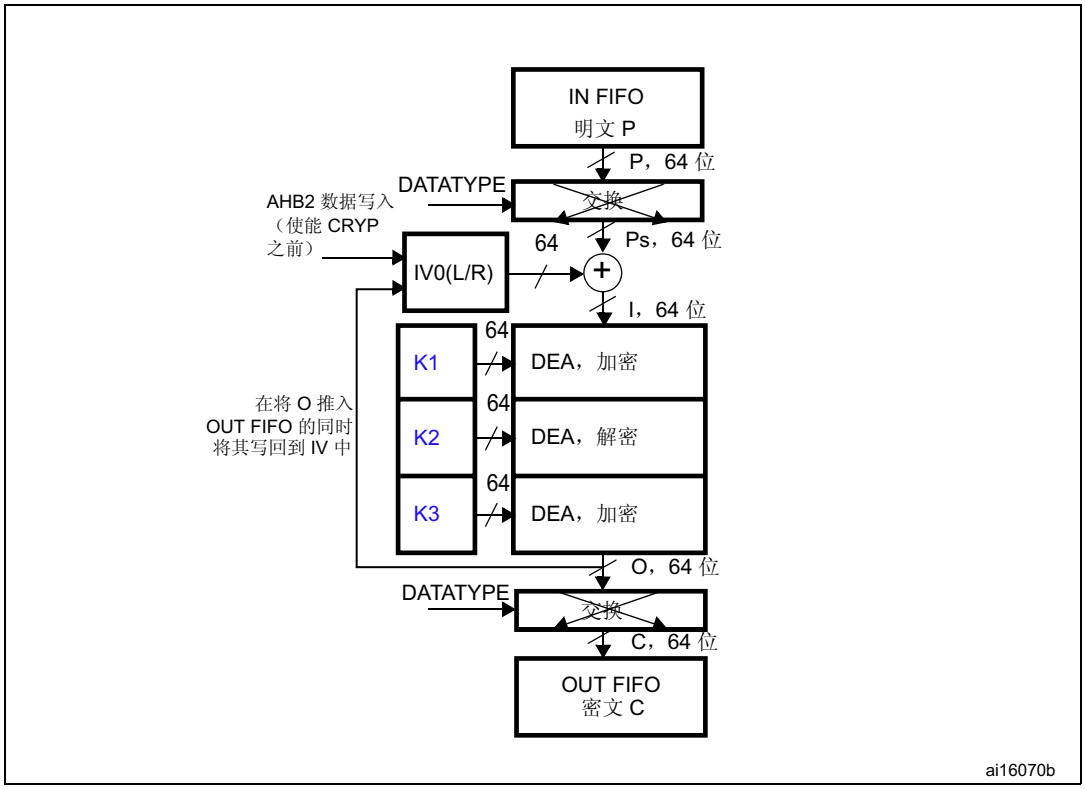
- DES/TDES-CBC 模式加密

  图 205 介绍了 DES 和三重 TDES 密码分组链接 (DES/TDES-CBC) 模式加密。该模式首先将明文消息分成多个 64 位数据块。在 TCBC 加密中，执行位/字节/半字交换（请参见*第 522 页的第 20.3.3 节：数据类型*）后获得的第一个输入块 (I_1) 是通过一个 64 位初始化向量 IV 与第一个明文数据块 (P_1) 进行异或运算形成的 ($I_1 = IV \oplus P_1$)。输入块通过 DEA 在加密状态下使用 K1 进行加密处理。上述处理过程的输出会直接反馈到 DEA 的输入，在解密状态下使用 K2 执行 DES。上述处理过程的输出会直接反馈到 DEA 的输入，在加密状态下使用 K3 执行 DES。生成的 64 位输出块 (O_1) 将直接用作密文 (C_1)，即 $C_1 = O_1$。然后，第一个密文块与第二个明文数据块进行异或运算，从而生成第二个输入块 (I_2) = ($C_1 \oplus P_2$)。注意，此时的 I_2 和 P_2 指的是第二个块。第二个输入块通过 TDEA 处理而生成第二个密文块。此加密处理会不断将后续密文块和明文块链接到一起，直到消息中最后一个明文块得到加密为止。如果消息中包含的数据块数不是整数，则应按照应用程序指定的方式对最后的不完整数据块进行加密。

- DES/TDES-CBC 模式解密

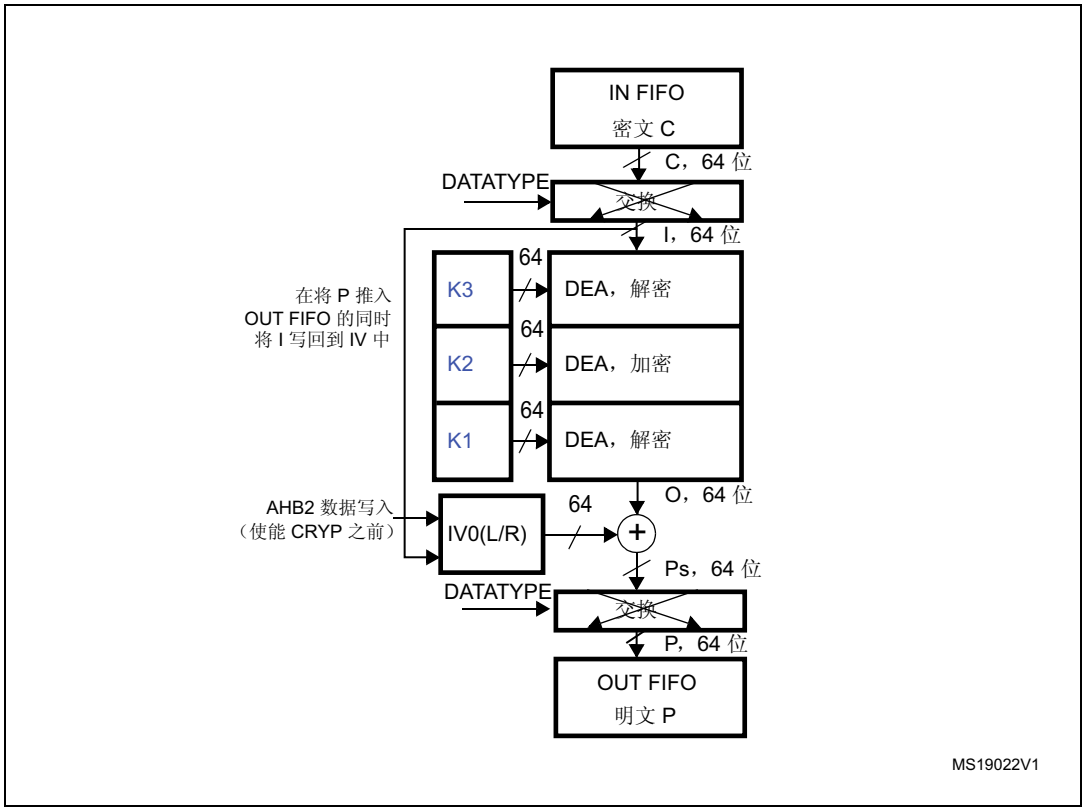
  在 DES/TDES-CBC 解密（请参见*图 206*）中，第一个密文块 (C_1) 将直接用作输入块 (I_1)。该密钥序列与加密过程中使用的密钥序列相反。输入块通过 DEA 在解密状态下使用 K3 进行解密处理。上述处理过程的输出会直接反馈到 DEA 的输入，在加密状态下使用 K2 执行 DES。生成值会直接反馈到 DEA 的输入，在解密状态下使用 K1 处理 DES。生成的输出块与 IV（必须与加密期间使用的相同）进行异或运算，从而生成第一个明文块 ($P_1 = O_1 \oplus IV$)。然后，第二个密文块将用作下一个输入块，并由 TDEA 进行处理。生成的输出块与第一个密文块进行异或运算，从而生成第二个明文数据块 ($P_2 = O_2 \oplus C_1$)。（注意，P_2 和 O_2 指的是第二个数据块。）TCBC 解密过程将以此方式继续进行，直到最后一个完整密文块得到解密为止。必须按照应用程序指定的方式对不完整数据块密文进行解密。

图 205. DES/TDES-CBC 模式加密



1. K：密钥；C：密文；I：输入块；O：输出块；Ps：交换前（解码时）或交换后（编码时）的明文；P：明文；IV：初始化向量。

图 206. DES/TDES-CBC 模式解密



1. K：密钥；C：密文；I：输入块；O：输出块；Ps：交换前（解码时）或交换后（编码时）的明文；P：明文；IV：初始化向量。

20.3.2 AES 加密内核

AES 加密内核由以下三部分组成：

- AES 算法（AEA：高级加密算法）
- 多个密钥
- 初始化向量或随机值

AES 可使用以下三种长度的密钥：128 位密钥、192 位密钥或 256 位密钥，并且根据使用的操作模式，不使用初始化向量 (IV) 或使用一个 128 位的初始化向量 (IV)。

AES 中涉及的基本处理如下：128 位的输入块通过输入 FIFO 读取，然后发送至 AEA 使用密钥 (K0...3) 进行加密。密钥格式取决于密钥大小：

- 如果密钥大小 = 128：密钥 = [K3 K2]
- 如果密钥大小 = 192：密钥 = [K3 K2 K1]
- 如果密钥大小 = 256：密钥 = [K3 K2 K1 K0]

其中 Kx = [KxR KxL]，R = 右，L = 左

生成的输出块用于计算密文，具体取决于使用的模式。

FIPS PUB 197（2001 年 11 月 26 日）对 AES 内核提供的四种操作模式中涉及的处理过程进行了详尽的解释，这四种模式为：AES-ECB 加密、AES-ECB 解密、AES-CBC 加密和 AES-CBC 解密。本参考手册仅对每种模式进行了简要解释。

AES 电子密码本 (AES-ECB) 模式

- AES-ECB 模式加密

  图 207 介绍了 AES 电子密码本 (AES-ECB) 模式加密。

  在 AES-ECB 加密中，执行位/字节/半字交换（请参见*第 522 页的第 20.3.3 节：数据类型*）后，128 位明文数据块 (P) 将用作输入块 (I)。输入块通过 AEA 在加密状态下使用 128 位、192 位或 256 位密钥进行处理。执行位/字节/半字交换后，生成的 128 位输出块 (O) 将用作密文 (C)。该密文随后推入 OUT FIFO。

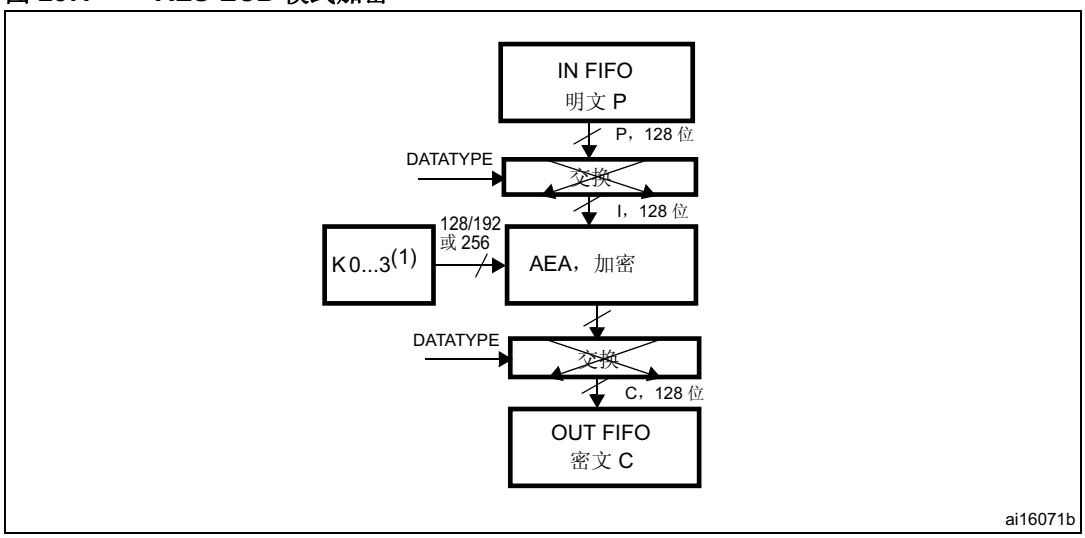
- AES-ECB 模式解密

  图 208 介绍了 AES 电子密码本 (AES-ECB) 模式加密。

  要在 ECB 模式下执行 AES 解密，需要准备密钥（需要针对加密执行完整的密钥计划），方法为：收集最后一个轮密钥并将其用作解密密文的第一个轮密钥。该准备过程通过 AES 内核计算完成。有关如何准备密钥的详细信息，请参见*第 20.3.6 节：加密或解密执行步骤*。

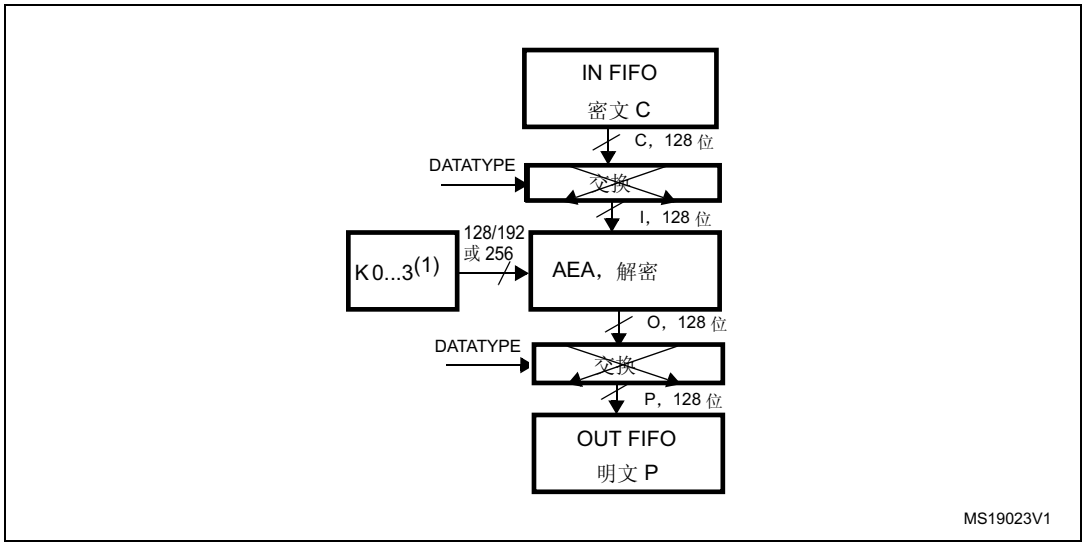
  在 AES-ECB 解密中，执行位/字节/半字交换后，128 位密文块 (C) 将用作输入块 (I)。该密钥序列与加密处理中的密钥序列相反。执行位/字节/半字交换，生成的 128 位输出块 (O) 将产生明文 (P)。

图 207. AES-ECB 模式加密



1. K：密钥；C：密文；I：输入块；O：输出块；P：明文。
2. 如果密钥大小 = 128：密钥 = [K3 K2]。
   如果密钥大小 = 192：密钥 = [K3 K2 K1]
   如果密钥大小 = 256：密钥 = [K3 K2 K1 K0]。

图 208. AES-ECB 模式解密



1. K：密钥；C：密文；I：输入块；O：输出块；P：明文。
2. 如果密钥大小 = 128 => 密钥 = [K3 K2]。
   如果密钥大小 = 192 => 密钥 = [K3 K2 K1]
   如果密钥大小 = 256 => 密钥 = [K3 K2 K1 K0]。

AES 加密分组链接 (AES-CBC) 模式

- AES-CBC 模式加密

  AES 加密分组链接 (AES-CBC) 模式解密如*图 209* 所示。

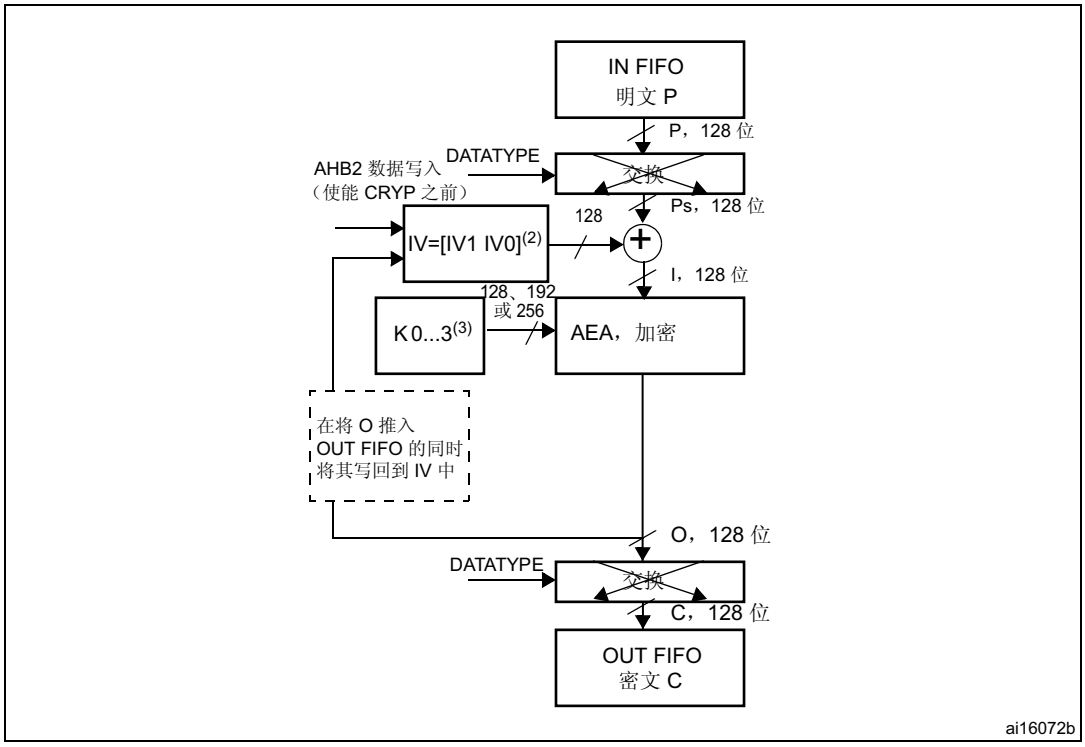
  在 AES-CBC 加密中，执行位/字节/半字交换（请参见*第 522 页的第 20.3.3 节：数据类型*）后所获得的第一个输入块 (I_1) 是通过一个 128 位初始化向量 IV 与第一个明文数据块 (P_1) 进行异或运算形成的 ($I_1 = IV \oplus P_1$)。输入块通过 AEA 在加密状态下使用 128 位、192 位或 256 位密钥 (K0...K3) 进行处理。生成的 128 位输出块 (O_1) 将直接用作密文 (C_1)，即 $C_1 = O_1$。然后，第一个密文块与第二个明文数据块进行异或运算，从而生成第二个输入块 (I_2) = ($C_1 \oplus P_2$)。注意，此时的 I_2 和 P_2 指的是第二个块。第二个输入块通过 AEA 处理而生成第二个密文块。此加密处理会不断将后续密文块和明文块链接到一起，直到消息中最后一个明文块得到加密为止。如果消息中包含的数据块数不是整数，则应按照应用程序指定的方式对最后的不完整数据块进行加密。

  在 CBC 模式下（如 ECB 模式），必须准备密钥才可以执行 AES 解密。有关如何准备密钥的详细信息，请参见*第 526 页的第 20.3.6 节：加密或解密执行步骤*。

- AES-CBC 模式解密

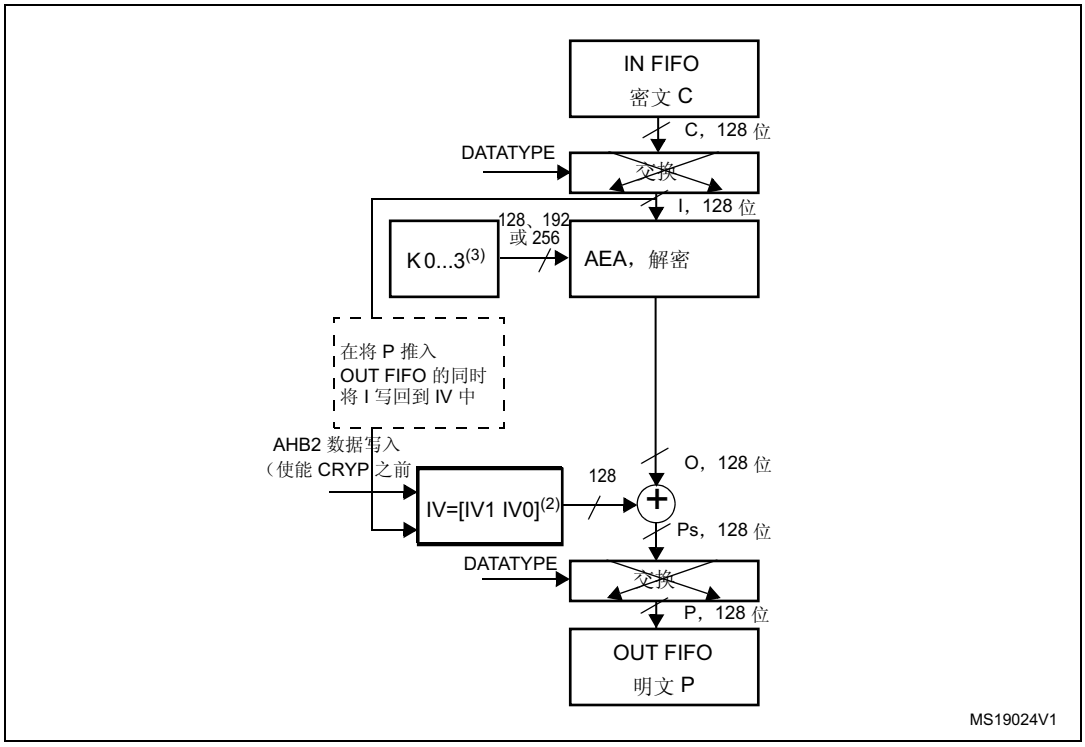
  在 AES-CBC 解密（请参见*图 210*）中，第一个 128 位密文块 (C_1) 将直接用作输入块 (I_1)。输入块通过 AEA 在解密状态下使用 128 位、192 位或 256 位密钥进行处理。生成的输出块与 128 位初始化向量 IV（必须与加密期间使用的相同）进行异或运算，从而生成第一个明文块 ($P_1 = O_1 \oplus IV$)。然后，第二个密文块将用作下一个输入块，并由 AEA 进行处理。生成的输出块与第一个密文块进行异或运算，从而生成第二个明文数据块 ($P_2 = O_2 \oplus C_1$)。（注意，P_2 和 O_2 指的是第二个数据块。）AES-CBC 解密过程将以此方式继续进行，直到最后一个完整密文块得到解密为止。必须按照应用程序指定的方式对不完整数据块密文进行解密。

图 209. AES-CBC 模式加密



1. K：密钥；C：密文；I：输入块；O：输出块；Ps：交换前（解码时）或交换后（编码时）的明文；P：明文；IV：初始化向量。
2. IVx = [IVxR IVxL]，R = 右，L = 左。
3. 如果密钥大小 = 128 => 密钥 = [K3 K2]。
   如果密钥大小 = 192 => 密钥 = [K3 K2 K1]
   如果密钥大小 = 256 => 密钥 = [K3 K2 K1 K0]。

图 210. AES-CBC 模式解密



1. K：密钥；C：密文；I：输入块；O：输出块；Ps：交换前（解码时）或交换后（编码时）的明文；P：明文；IV：初始化向量。
2. IVx = [IVxR IVxL]，R = 右，L = 左。
3. 如果密钥大小 = 128 => 密钥 = [K3 K2]。
   如果密钥大小 = 192 => 密钥 = [K3 K2 K1]
   如果密钥大小 = 256 => 密钥 = [K3 K2 K1 K0]。

AES 计数器模式 (AES-CTR) 模式

AES 计数器模式使用 AES 块作为密钥流生成器。然后，生成的密钥与明文进行异或运算后获得密文。因此，认为 CTR 加密和解密有差异是不合理的，因为这两种操作完全相同。

事实上，给定：

- 明文：P[0]、P[1]、...、P[n]（每个均为 128 位）
- 要使用的密钥 K（大小不重要）
- 初始计数器块（称为 ICB，但功能与 CBC 的 IV 相同）

密文将按如下公式计算：

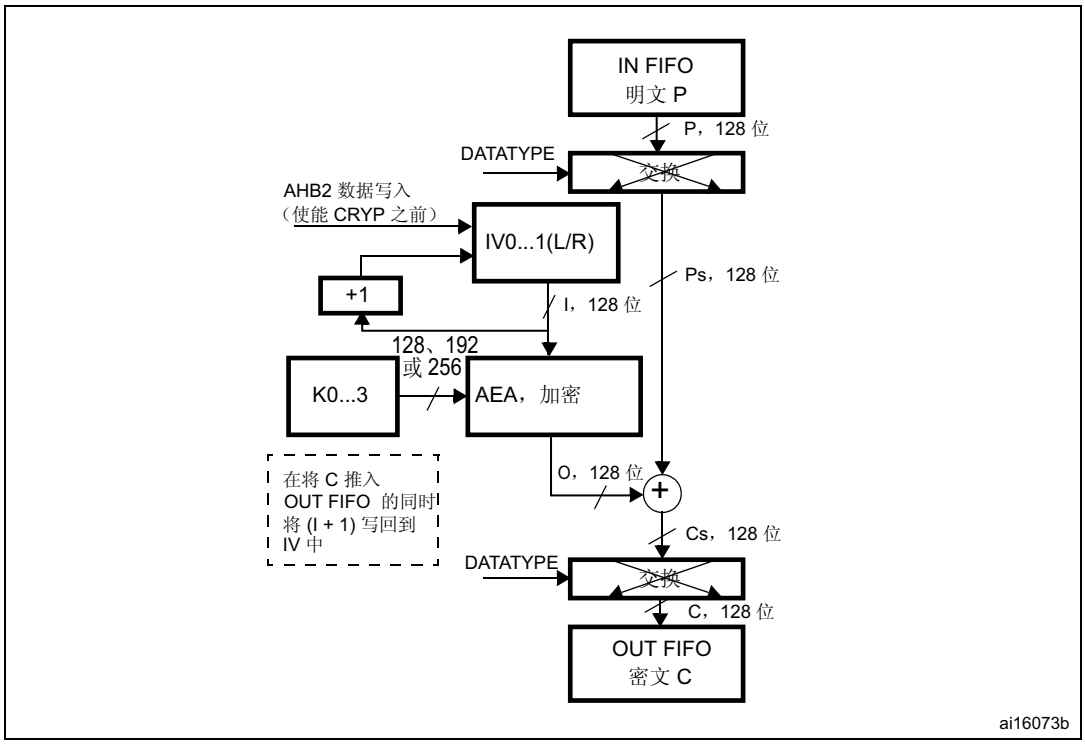
`C[i] = enck(iv[i]) xor P[i]`，其中：

`iv[0]` = ICB，而 `iv[i+1] = func(iv[i])`，其中 `func` 是应用于先前 iv 块的更新函数；`func` 从本质上而言是组成 iv 块的字段之一的增量。

假定解密所用的 ICB 与加密所用的 ICB 相同，则在解密期间生成的密钥流便与加密期间生成的密钥流相同。然后，密文会与密钥流进行异或运算，从而得到原始明文。因此，解密操作与加密操作的操作方式完全相同。

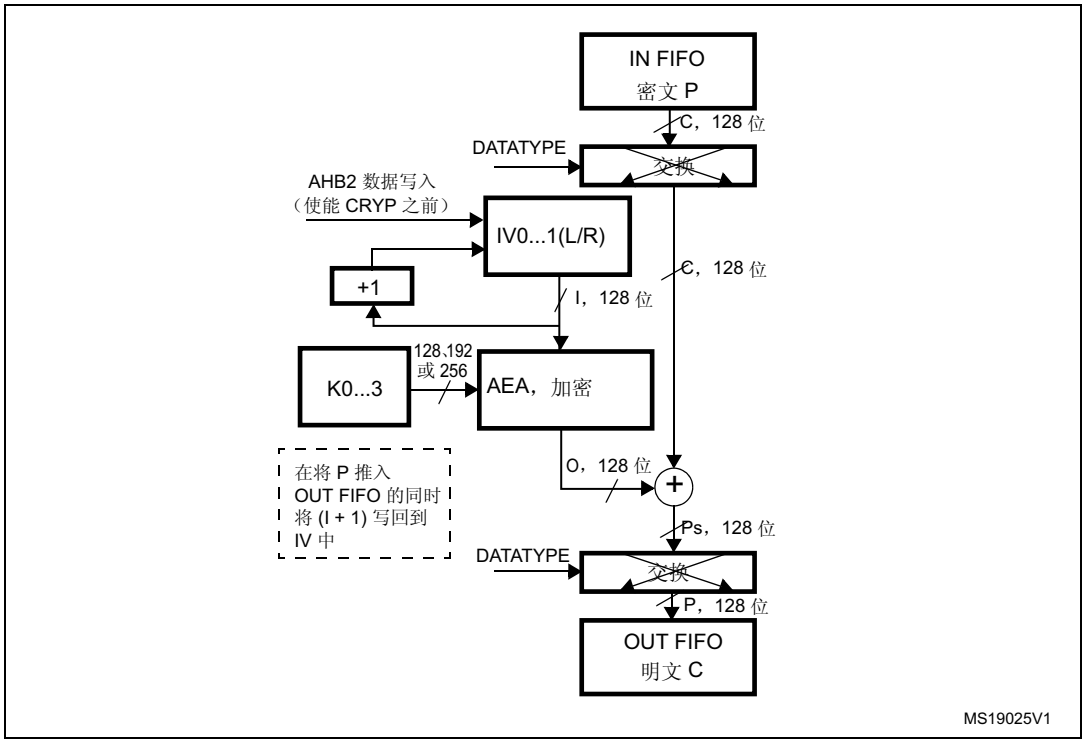
图 211 和*图 212* 分别介绍了 AES-CTR 加密和解密。

图 211. AES-CTR 模式加密



1. K：密钥；C：密文；I：输入块；O：输出块；Ps：交换前（解码时）或交换后（编码时）的明文；Cs：交换后（解码时）或交换前（编码时）的密文；P：明文；IV：初始化向量。

图 212. AES-CTR 模式解密



1. K：密钥；C：密文；I：输入块；O：输出块；Ps：交换前（解码时）或交换后（编码时）的明文；Cs：交换后（解码时）或交换前（编码时）的密文；P：明文；IV：初始化向量。

图 213 显示了由标准 [2] 定义的 IV 块结构。它由三个不同字段组成。

图 213. 计数器模式下的初始计数器块结构

- 随机值为 32 位一次性值。应将新的随机值分配至每个不同的通信。
- 初始化向量 (IV) 是一个 64 位值，而标准指定加密器必须选择 IV，以确保一个给定值只用于一个给定的密钥。
- 计数器是一个 32 位的大端模式的整数，随着块的加密次数而递增。应将计数器的初始值设置为“1”。

块将使用最低有效的 32 位进行增量计数，而保持其它最高有效的 96 位不变。

AES Galois/计数器模式 (GCM)

AES Galois/计数器模式 (GCM) 允许加密和验证明文，以及生成对应的密文和标记（也称为消息验证码或消息完整性检查）。该算法基于 AES 计数器模式，可保证保密性。它针对固定有限的字段使用乘法器来生成标记。开始执行算法时需要使用初始化向量。

要处理的消息分为 2 个部分：

- 标头（又称附加认证数据）：需要验证但不受保护的数据（如用于路由数据包的信息）
- 有效负载（又称明文或密文）：经验证和加密的消息本身。

注意： *标头必须在有效负载的前面，并且两部分不能混合。*

根据 GCM 标准，在消息结束时，必须传递完由标头大小（64 位）和有效负载大小（64 位）构成的特定 128 位块。在计算过程中，必须将标头块与有效负载块区分开。

在 GCM 模式下，执行加密/解密需要以下四个步骤：

1. GCM 初始化阶段

在此步骤中，将在内部计算和保存 HASH 密钥，供处理所有块使用。建议遵守下列顺序：

a) 确保通过将 CRYP_CR 寄存器中的 CRYPEN 位清零来禁止加密处理器。

b) 通过将 CRYP_CR 中的 ALGOMODE 位编程为“01000”来选择 GCM 链接模式。

c) 将 CRYP_CR 中的 GCM_CCMPH 位配置为“00”来开始 GCM 初始化阶段。

d) 初始化 CRYP_KEYRx 中的密钥寄存器（128 位、192 位和 256 位）以及初始化向量 (IV)。

e) 将 CRYPEN 位置 1 以开始计算 HASH 密钥。

f) 等待 CRYPEN 位清零后进入下一个阶段。

g) 将 CRYPEN 位置“1”。

2. GCM 标头阶段

必须在 GCM 初始化阶段后方能执行该步骤：

h) 将 CRYP_CR 中的 GCM_CCMPH 位置“01”来指示标头阶段已开始。

i) 写入标头数据。可以使用以下三种方法：

– 按 32 位块将数据写入 CRYP_DIN 寄存器，并使用 IFNF 标志来确定输入 FIFO 是否能够接收数据。标头大小必须为 128 位（4 个字）的倍数。

– 按 8 个字块将数据写入 CRYP_DIN 寄存器，并使用 IFEM 标志来确定输入 FIFO 是否能够接收数据（IFEM = “1”）。标头大小必须为 128 位（4 个字）的倍数。

– 使用 DMA。

j) 提供完所有的标头数据后，等待 CRYP_SR 寄存器中的 BUSY 位清零。

3. GCM 有效负载阶段（加密/解密）

必须在 GCM 标头阶段后方能执行该步骤：

k) 将 CRYP_CR 寄存器中的 GCM_CCMPH 配置为“10”。

l) 通过使用 CRYP_CR 中的 ALGODIR 位来选择算法方向（加密或解密）。

m) 将有效负载消息写入 CRYP_DIN 寄存器，并使用 IFNF 标志来确定输入 FIFO 是否能够接收数据。也可以按 8 个字块将数据写入 CRYP_DIN 寄存器，并使用 IFEM 标志来确定输入 FIFO 是否能够接收数据（IFEM = “1”）。同时，可对 CRYP_DOUT 寄存器的 OFNE/OFFU 标志进行监控，以检查输出 FIFO 是否为空。

n) 重复之前的步骤，直到所有的有效负载块均得到加密或解密。也可以使用 DMA。

4. GCM 最后阶段

该步骤将生成验证标记：

o) 将 CRYP_CR 中的 GCM_CCMPH[1:0] 配置为“11”。

p) 将输入写入 CRYP_DIN 寄存器 4 次。输入中必须包含标头中的位数（64 位），并连接着有效负载中的位数（64 位）。

q) 等待 CRYP_SR 寄存器中的 OFNE 标志（FIFO 输出非空）置“1”。

r) 读取 CRYP_DOUT 寄存器 4 次：此输出即为验证标记。

s) 禁止加密处理器（CRYP_CR 中的 CRYPEN 位 =“0”）

注意： *执行解密时，开始时无需计算密钥。解密结束时，应将生成的标记与通过消息传递的预期标记进行比较。此外，必须将 ALGODIR 位（算法方向）置“1”。*

从标头阶段到标记阶段时，无需禁止/使能 CRYP 处理器。

AES Galois 消息认证代码 (GMAC)

加密处理器还支持使用 GMAC 算法对明文进行验证。它针对固定有限的字段使用 GCM 算法和乘法器来生成相应标记。

开始执行算法时需要使用初始化向量。

实际上，GMAC 算法相当于应用在仅由标头构成的消息上的 GCM 算法。因此，不需要有效负载阶段。

AES 组合式密码机 (CCM)

CCM 算法允许加密和验证明文，以及生成对应的密文和标记（又称消息验证码或消息完整性检查）。该算法基于 AES 计数器模式，可保证保密性。它使用 AES CBC 模式生成 128 位标记。

该 CCM 标准（RFC 3610 Counter with CBC-MAC (CCM) 标准，2003 年 9 月发布）可为首个验证块（在此标准中称为 B0）定义特定的编码规则。具体而言，首个块包括标志、随机值和有效负载长度（字节数）。CCM 标准可为加密/解密指定另外的格式（称为 A 或计数器）。计数器在有效负载阶段会递增计数，其 32 个低有效位在生成标记期间会初始化为“1”（在 CCM 标准中称为 A0 数据包）。

注意： *硬件不执行 B0 数据包的格式化操作，该操作由软件处理。*

对于 GCM 算法，要处理的消息会分为 2 个部分：

- 标头（又称附加认证数据）：需要验证但不受保护的数据（如用于路由数据包的信息）
- 有效负载（又称明文或密文）：经验证和加密的消息本身。

注意： *标头必须在有效负载的前面，并且两部分不能混合。*

在 CCM 模式下，执行加密或解密需要以下四个步骤：

1. **CCM 初始化阶段**

   第一步，将 CCM 消息的 B0 数据包（第一个数据包）会写入 CRYP_DIN 寄存器。在此阶段期间，CRYP_DOUT 寄存器不包含任何输出数据。

   必须遵守以下顺序：

   a) 确保通过将 CRYP_CR 寄存器中的 CRYPEN 位清零来禁止加密处理器。

   b) 通过将 CRYP_CR 寄存器中的 ALGOMODE 位置为“01001”来选择 CCM 链接模式。

   c) 将 CRYP_CR 中的 GCM_CCMPH 位配置为“00”来开始 CCM 初始化阶段。

   d) 初始化 CRYP_KEYRx 中的密钥寄存器（128 位、192 位和 256 位）以及初始化向量 (IV)。

   e) 将 CRYP_CR 中的 CRYPEN 位置“1”。

   f) 将 B0 数据包写入输入数据寄存器。

   g) 等待 CRYPEN 位清零后进入下一个阶段。

   h) 将 CRYPEN 位置“1”。

2. CCM 标头阶段

必须在 CCM 初始化阶段后方能执行该步骤。加密与解密的顺序完全相同。

在此阶段期间，CRYP_DOUT 寄存器不包含任何输出数据。

如无附加验证数据，则可以跳过该阶段。

必须遵守以下顺序：

i) 将 CRYP_CR 中的 GCM_CCMPH 位置“01”来指示标头阶段已开始。

j) 可以使用以下三种方法：

– 按 32 位块将标头数据写入 CRYP_DIN 寄存器，并使用 IFNF 标志来确定输入 FIFO 是否能够接收到数据。标头大小必须为 128 位（4 个字）的倍数。

– 按 8 个字块将标头数据写入 CRYP_DIN 寄存器，并使用 IFEM 标志来确定输入 FIFO 是否能够接收到数据（IFEM = “1”）。标头大小必须为 128 位（4 个字）的倍数。

– 使用 DMA。

注意： *必须使用标头长度格式化首个块 B1。应通过软件执行该任务。*

k) 提供完所有的标头数据后，等待 BUSY 标志清零。

3. CCM 有效负载阶段（加密/解密）

必须在 CCM 标头阶段后方能执行该步骤。在此阶段期间，加密/解密的有效负载将存储在 CRYP_DOUT 寄存器中。

必须遵守以下顺序：

l) 将 CRYP_CR 中的 GCM_CCMPH 位配置为“10”。

m) 通过使用 CRYP_CR 中的 ALGODIR 位来选择算法方向（加密或解密）。

n) 将有效负载消息写入 CRYP_DIN 寄存器，并使用 IFNF 标志来确定输入 FIFO 是否能够接收数据。也可以按 8 个字块将数据写入 CRYP_DIN 寄存器，并使用 IFEM 标志来确定输入 FIFO 是否能够接收数据（IFEM = “1”）。同时，可对 CRYP_DOUT 寄存器的 OFNE/OFFU 标志进行监控，以检查输出 FIFO 是否为空。

o) 重复之前的步骤，直到所有的有效负载块均得到加密或解密。也可以使用 DMA。

4. CCM 最后阶段

该步骤将生成认证标记。在此阶段期间，将生成消息的认证标记，该标记存储在 CRYP_DOUT 寄存器中。

p) 将 CRYP_CR 中的 GCM_CCMPH[1:0] 位配置为“11”。

q) 加载 A0 初始化计数器，并通过将 32 位数据写入 CRYP_DIN 寄存器 4 次来编程 128 位 A0 值。

r) 等待 CRYP_SR 寄存器中的 OFNE 标志（FIFO 输出非空）置“1”。

s) 读取 CRYP_DOUT 寄存器 4 次：此输出即为加密的验证标记。

t) 禁止加密处理器（CRYP_CR 中的 CRYPEN 位 = “0”）

注意： *硬件不执行原始 B0 和 B1 数据包的格式化操作，也不执行加密与解密之间的标记比较操作。这些操作通过软件处理。*

*从标头阶段到标记阶段，无需禁止/使能*加密处理器。

AES 密文消息验证代码 (CMAC)

CMAC 算法允许验证明文并生成相应的标记。CMAC 顺序与 CCM 顺序相同，但其跳过有效负载阶段。

20.3.3 数据类型

将数据写入 CRYP_DIN 寄存器时，一次会向 CRYP 处理器输入 32 位（字）数据。DES 的原则是每隔 64 位对数据流进行处理。针对每个 64 位块，将从 M1 到 M64 对位进行编号，其中 M1 为块最左侧的位，M64 为块最右侧的位。AES 使用相同的原则，但其块大小为 128 位。

系统存储器结构采用小端模式：无论使用何种数据类型（位、字节、16 位半字、32 位字），最低有效数据均占用最低地址位置。因此，对于从 IN FIFO 中读取的数据，在其进入 CRYP 处理器之前，必须对这些数据执行位、字节或半字交换操作（取决于要加密的数据类型）。在 CRYP 数据写入 OUT FIFO 之前，需要对其执行同样的交换操作。例如，对 ASCII 文本流执行字节交换操作。

要处理的数据类型通过 CRYP 控制寄存器 (CRYP_CR) 中的 DATATYPE 位字段进行配置。

表 91. 数据类型

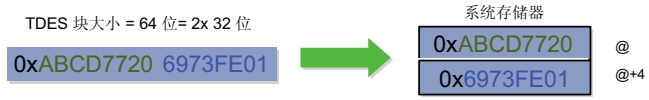
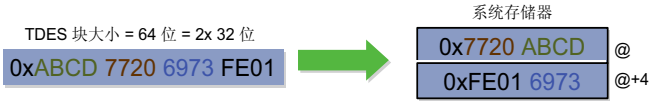
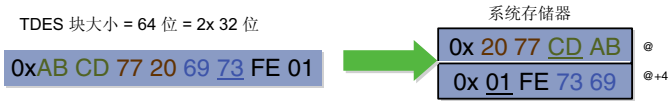
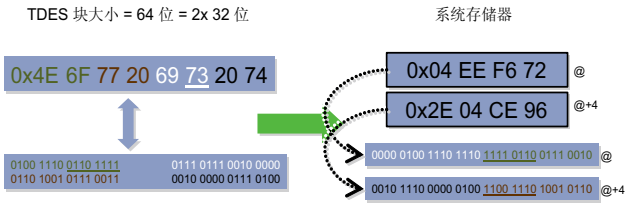
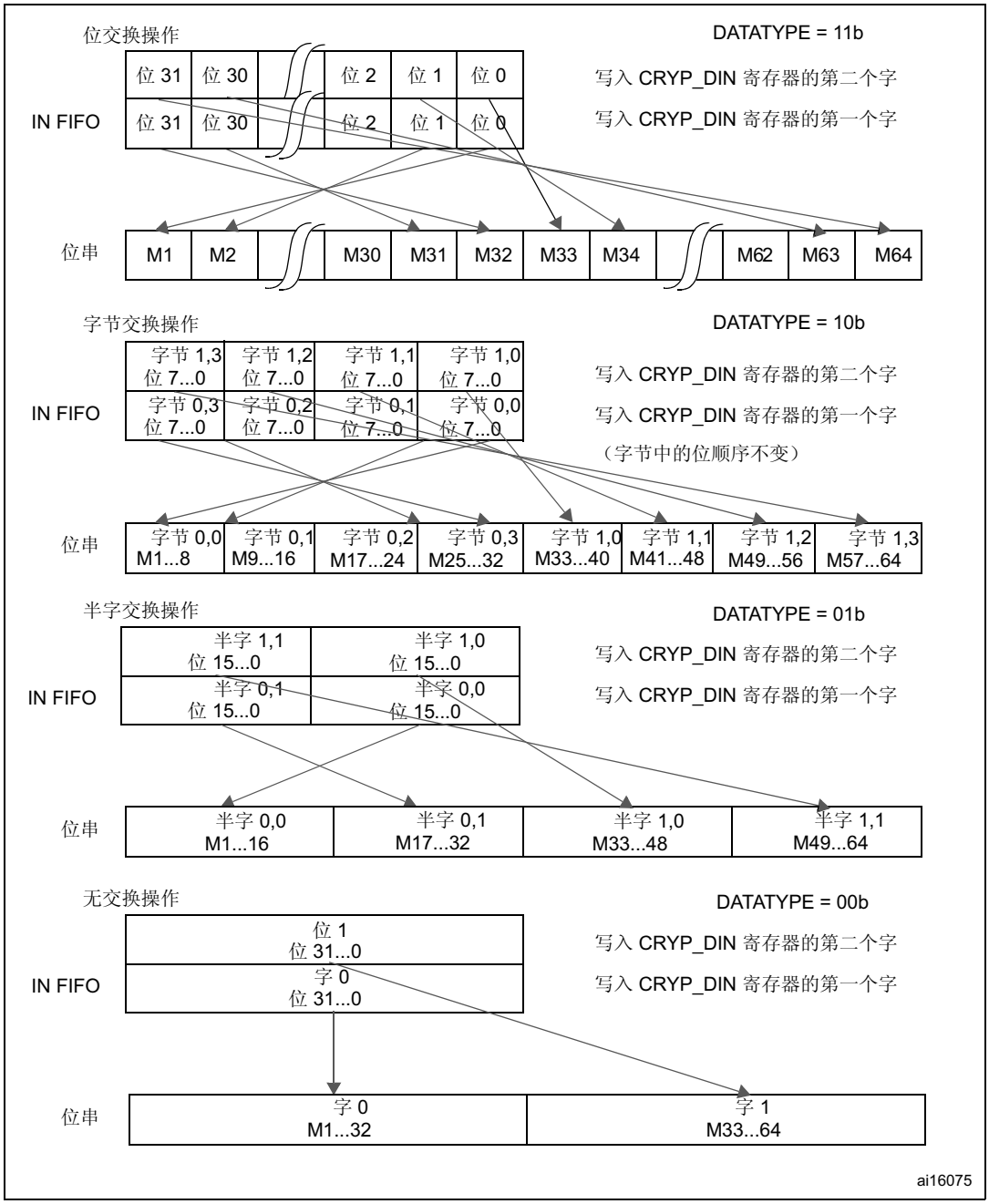
CRYP_CR 中的 DATATYPE	要执行的交换	系统存储器数据（明文或密码）
00b	无交换	示例：TDES 块值 **0xABCD77206973FE01** 在系统存储器中表示为： 
01b	半字（16 位）交换	示例：TDES 块值 **0xABCD77206973FE01** 在系统存储器中表示为： 
10b	字节（8位）交换	示例：TDES 块值 **0xABCD77206973FE01** 在系统存储器中表示为： 
11b	位交换	TDES 块值 **0x4E6F772069732074** 在系统存储器中表示为： 

图 214 展示了根据不同的 DATATYPE 值，64 位数据块 M1...64 是如何由 CRYP 处理器从 IN FIFO 中弹出的两个连续的 32 位字来构建的。相同的原理，可以轻松扩展构建用于 AES 加密算法的 128 位数据块（对于 AES，块长度为四个 32 位字，但由于交换仅发生在字级别，因此它与此处描述的 TDES 的交换过程相同）。

注意： *IN FIFO 和 CRYP 数据块之间，以及 CRYP 数据块和 OUT FIFO 之间执行相同的交换操作。*

图 214. 根据 DATATYPE 构建 64 位块



20.3.4 初始化向量 —— CRYP_IV0...1(L/R)

可将初始化向量视为两个 64 位数据项。因此，初始化向量在系统存储器中的数据格式和表示形式均不同于明文或密码数据，而且它们不受 DATATYPE 值的影响。

初始化向量采用两个连续的 32 位字进行定义，即 CRYP_IVL（左部分，记为位 IV1...32）和 CRYP_IVR（右部分，记为位 IV33...64）。

在 DES 或 TDES CBC 加密期间，CRYP_IV0(L/R) 位会与一个 64 位数据块（即，数据块的 M1...64 位）进行异或运算，这里的 64 位数据块是根据 DATATYPE 值交换后从 IN FIFO 弹出的。当 DEA3 块的输出可用时，该输出值会复制到 CRYP_IV0(L/R) 向量，之后，这个新值与 IN FIFO 弹出的下一个 64 位数据块进行异或运算，以此类推。

在 DES 或 TDES CBC 解密期间，CRYP_IV0(L/R) 位首先会与 TDEA1 块输出的 64 位数据块（即，M1...64 位）进行异或运算，然后异或运算后的结果根据 DATATYPE 值进行交换并压入 OUT FIFO。异或运算后的结果进行交换并压入 OUT FIFO 后，IN FIFO 的输出会取代 CRYP_IV0(L/R) 值，然后 IN FIFO 执行弹出操作，随后可对新的 64 位数据块进行处理。

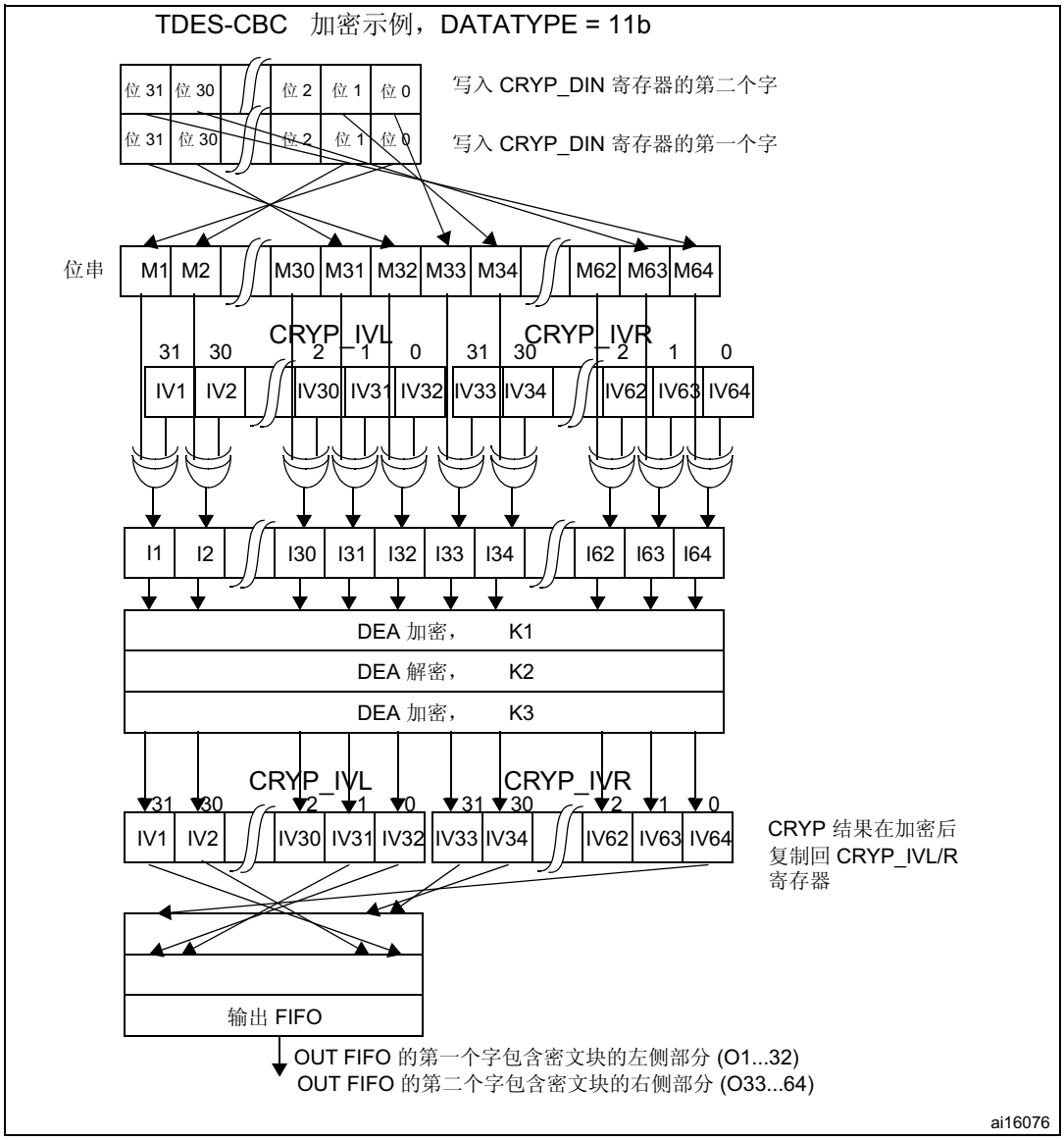
在 AES CBC 加密期间，CRYP_IV0...1(L/R) 位会与一个 128 位数据块进行异或运算，这里的 128 位数据块是根据 DATATYPE 值交换后从 IN FIFO 中弹出的。当 AES 内核的输出可用时，该输出值会复制到 CRYP_IV0...1(L/R) 向量，之后，这个新值与 IN FIFO 弹出的下一个 128 位数据块进行异或运算，以此类推。

在 AES CBC 解密期间，CRYP_IV0...1(L/R) 位首先会与 AES 内核输出的 128 位数据块进行异或运算，然后异或运算后的结果根据 DATATYPE 值进行交换并压入 OUT FIFO。异或运算后的结果进行交换并压入 OUT FIFO 后，IN FIFO 的输出会取代 CRYP_IV0...1(L/R) 值，然后 IN FIFO 执行弹出操作，随后可对新的 128 位数据块进行处理。

在 AES CTR 加密或解密期间，AES 内核会将 CRYP_IV0...1(L/R) 位加密。之后，加密的结果会与一个 128 位数据块进行异或运算，这里的 128 位数据块是根据 DATATYPE 值交换后从 IN FIFO 中弹出的。异或运算后的结果进行交换并压入 OUT FIFO 后，CRYP_IV0...1(L/R) 值 (32 LSB) 的计数器部分会递增。

当 CRYP_SR 寄存器的 BUSY 位 = 1b 时，针对 CRYP_IV0...1(L/R) 寄存器的任何写操作都会被忽略（CRYP_IV0...1(L/R) 寄存器内容不会被修改）。因此，在修改初始化向量之前，必须检查 BUSY 位 = 0b。

图 215. TDES-CBC 加密过程中使用的初始化向量



20.3.5 CRYP 忙碌状态

当输入 FIFO 中有充足的数据（至少有 2 个字可用于 DES 或 TDES 算法模式，至少有 4 个字可用于 AES 算法模式）、输出 FIFO 中有充足的自由空间（至少有 2 个 (DES/TDES) 或 4 个 (AES) 字位置），以及 CRYP_CR 寄存器中的位 CRYPEN = 1 时，加密处理器会自动开始加密或解密过程（根据 CRYP_CR 寄存器中 ALGODIR 位的值）。

在此过程中，执行三重 DES 算法需要 48 个 AHB2 时钟周期，执行简易 DES 算法需 16 个 AHB2 时钟周期，而执行密钥长度为 128、192 或 256 位的 AES 算法则分别需要 14、16 或 18 个 AHB2 时钟周期。在整个过程中，CRYP_SR 寄存器的 BUSY 位始终置“1”。完成此过程后，CRYP 内核会将两个 (DES/TDES) 或四个 (AES) 字写入输出 FIFO，并将 BUSY 位清零。在 CBC、CTR 模式下，还会更新初始化向量 CRYP_IVx(L/R)R (x = 0..3)。

当加密处理器繁忙时（CRYP_SR 寄存器中的位 BUSY = 1b），会忽略针对密钥寄存器（CRYP_Kx(L/R)R，x = 0..3）、初始化寄存器 （CRYP_IVx(L/R)R, x = 0..3）或 CRYP_CR 寄存器的位 [9:2] 的写操作，且不会修改这些寄存器。因此，不能在加密处理器处理数据块时修改其配置。不过，可以在 BUSY = 1 时将 CRYPEN 位清零，这种情况下，只有完成正在进行的 DES、TDES 或 AES 处理过程并将两个/四个字的结果写入输出 FIFO 后才能将 BUSY 位清零。

注意： *在 DES 或 TDES 模式下处理某个块时，如果输出 FIFO 已满并且输入 FIFO 至少含一个新块，则输入 FIFO 会弹出新块且 BUSY 位保持置 1，直到有足够的空间可将这个新块存储到输出 FIFO。*

20.3.6 加密或解密执行步骤

初始化

1. 初始化外设（操作顺序并不重要，除非密钥准备用于 AES-ECB 或 AES-CBC 解密。在准备密钥之前必须输入密钥大小和密钥值，准备好密钥后，必须立即配置相应算法）：
   a) 使用 CRYP_CR 寄存器中的 KEYSIZE 位配置密钥大小（129 位、192 位或 256 位，仅限 AES）
   b) 将对称密钥写入 CRYP_KxL/R 寄存器（需写入 2 到 8 个寄存器，具体取决于算法）
   c) 使用 CRYP_CR 寄存器中的 DATATYPE 位配置数据类型（1 位、8 位、16 位或 32 位）
   d) 在 AES-ECB 或 AES-CBC 解密的情况下，必须准备以下密钥：将 CRYP_CR 寄存器中的 ALGOMODE 位置为“111”来配置密钥准备模式。随后向 CRYPEN 位写入“1”：BUSY 位随即置 1。等待 BUSY 位返回 0（CRYPEN 位也会自动清零）：已准备好密钥供解密使用
   e) 使用 CRYP_CR 寄存器中的 ALGOMODE 位配置算法和链接（在 ECB/CBC 中为 DES/TDES，在 ECB/CBC/CTR/GCM/CCM 中为 AES）
   f) 使用 CRYP_CR 寄存器中的 ALGODIR 位配置方向（加密/解密）
   g) 将初始化向量写入 CRYP_IVxL/R 寄存器（仅在 CBC 或 CTR 模式下）
2. 向 CRYP_CR 寄存器中的 FFLUSH 位写入 1，刷新 IN 和 OUT FIFO

DMA 用于存储器数据传入和传出时的处理过程

1. 将 DMA 控制器配置为传输存储器中的输入数据。传输长度为消息的长度。当消息填充并非由外设进行管理时，消息长度必须为整个数量的数据块。数据传输均在突发模式下进行。AES 中的突发长度为 4 个字，而 DES/TDES 中的突发长度为 2 个字或 4 个字。应将 DMA 配置为在完成输出数据传输时设置一个中断，以指示处理过程已结束。
2. 通过向 CRYPEN 位写入 1 来使能加密处理器。将 CRYP_DMACR 寄存器中的 DIEN 和 DOEN 位置 1，以使能 DMA 请求。
3. 所有传输和处理过程均由 DMA 和加密处理器管理。DMA 中断表示处理过程已完成。两个 FIFO 通常均为空，且 BUSY = 0。

在中断期间通过 CPU 传输数据的处理过程

1. 将 CRYP_IMSCR 寄存器中的 INIM 和 OUTIM 位置 1，以使能中断。
2. 将 CRYP_CR 寄存器中的 CRYPEN 位置 1，以使能加密处理器。
3. 在中断中管理输入数据：将输入消息加载到 IN FIFO。一次性可加载 2 个字或 4 个字，或者加载数据直到 FIFO 已满。当消息的最后一个字进入 FIFO 时，可通过将 INIM 位清零来禁止中断。

4. 在中断中管理输出数据：读取 OUT FIFO 中的输出消息。一次可读取 1 个块（2 个字或 4 个字），或者读取数据直到 FIFO 为空。读取最后一个字后，INIM=0、BUSY=0 且两个 FIFO 均为空（IFEM=1 且 OFNE=0）。将 OUTIM 位清零可禁止中断，而将 CRYPEN 位清零可禁止外设。

不使用 DMA 也不使用中断时的处理过程

1. 将 CRYP_CR 寄存器中的 CRYPEN 位置 1，以使能加密处理器。
2. 将首个块写入输入 FIFO（2 到 8 个字）。
3. 重复以下步骤，直到处理完整个信息：
   a) 等待 OFNE=1，然后读取输出 FIFO（读取 1 个块，或 FIFO 为空为止）
   b) 等待 IFNF=1，然后写入 IN FIFO（写入 1 个块，或 FIFO 已满为止）
4. 处理过程结束时，BUSY=0 且两个 FIFO 均为空（IFEM=1 且 OFNE=0）。将 CRYPEN 位清零可禁止外设。

20.3.7 上下文交换

如果因 OS 发起的一项新任务需要上下文资源而需要进行上下文交换，则须执行以下任务来恢复完整的上下文（以使用 DMA 为例）：

在 AES 和 DES 的情况下

1. 保存上下文
   a) 将 CRYP_DMACR 寄存器中的 DIEN 位清零，以停止 DMA 对 IN FIFO 的数据传输。
   b) 等待 IN 和 OUT FIFO 均为空（CRYP_SR 寄存器中的 IFEM=1 且 OFNE=0），并且 BUSY 位清零。
   c) 将 CRYP_DMACR 寄存器中的 DOEN 位清零并将 CRYPEN 位清零，以停止 DMA 对 OUT FIFO 的数据传输。
   d) 保存当前配置（CRYP_CR 寄存器中的位 [9:2] 和位 19）和初始化向量（如果不是在 ECB 模式下）。密钥的值必须已经存在于存储器中。根据需要保存 DMA 状态（IN 和 OUT 消息的指针、保留字节数等）。

   使用 GCM/GMAC 或 CCM/CMAC 算法时应保存其它位：

   – CRYP_CR 寄存器中的位 [17:16]
   – 上下文交换寄存器：

     使用 GCM/GMAC 或 CCM/CMAC 算法时为 CRYP_CSGCMCCM0..7

     使用 GCM/GMAC 算法时为 CRYP_CSGCM0..7。
2. 配置和执行其它处理过程。
3. 上下文恢复
   a) 使用保存的配置按*第 526 页的第 20.3.6 节：加密或解密执行步骤*的*初始化*中所述配置处理器。针对 AES-ECB 或 AES-CBC 解密，必须再次准备密钥。
   b) 根据需要重新配置 DMA 控制器，以便传输余下的消息。
   c) 将 CRYPEN 位置 1 以使能处理器，将 DIEN 和 DOEN 位置 1，以使能 DMA 请求。

在 TDES 的情况下

可采用与 AES 相同的方式进行 TDES 下的上下文交换。不过，由于这种情况下的输入 FIFO 可包含多达 4 个未处理的块，而且每个块的处理时间很长，因此，在某些情况下，不用等待 IN FIFO 为空就可中断处理过程，这样能够节省时间。

1. 保存上下文
   a) 将 CRYP_DMACR 寄存器中的 DIEN 位清零，以停止 DMA 对 IN FIFO 的数据传输。
   b) 将 CRYPEN 位清零，以禁止处理器（处理过程会在当前块的末尾停止）。
   c) 等待 OUT FIFO 为空（CRYP_SR 寄存器中的 OFNE=0），并且 BUSY 位清零。
   d) 向 CRYP_DMACR 寄存器中的 DOEN 位写入 0，以停止 DMA 对 OUT FIFO 的数据传输。
   e) 保存当前配置（CRYP_CR 寄存器中的位 [9:2] 和位 19）和初始化向量（如果不是在 ECB 模式下）。密钥的值必须已经存在于存储器中。根据需要保存 DMA 状态（IN 和 OUT 消息的指针、保留字节数等）。读回 IN FIFO 中加载的尚未处理的数据，并将其保存到存储器中，直到 FIFO 为空。

注意： *在 GCM/GMAC 或 CCM/CMAC 模式下，还应保存 CRYP_CR 寄存器的位 [17:16]。*

2. 配置和执行其它处理过程。
3. 上下文恢复
   a) 使用保存的配置按*第 526 页的第 20.3.6 节：加密或解密执行步骤*的*初始化*中所述配置处理器。针对 AES-ECB 或 AES-CBC 解密，必须再次准备密钥。
   b) 将保存上下文期间保存的数据写入 IN FIFO。
   c) 根据需要重新配置 DMA 控制器，以便传输余下的消息。
   d) 将 CRYPEN 位置 1 以使能处理器，将 DIEN 和 DOEN 位置 1，以使能 DMA 请求。

20.4 CRYP 中断

CRYP 可产生两个可单独屏蔽的中断源。这两个中断源合为同一个中断信号，而该中断信号是 CRYP 发出的唯一中断信号，用于驱动 NVIC（嵌套向量中断控制器）。这一组合中断是两个单独屏蔽的中断源的或运算结果。如果下面列出的各个中断中的任何一个中断产生，则此组合中断即会产生。

通过更改 CRYP_IMSCR 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。

关于各个中断源的状态，通过 CRYP_RISR 寄存器可以读取原始中断状态，通过 CRYP_MISR 寄存器可以读取屏蔽中断状态。

输出 FIFO 服务中断 - OUTMIS

当输出 FIFO 中存在一个或多个（32 位字）数据项时，即会产生输出 FIFO 服务中断。通过读取输出 FIFO 的数据，直到读完所有有效（32 位）字，即可将此中断清除（即，该中断的状态与 OFNE（输出 FIFO 非空）标志一致）。

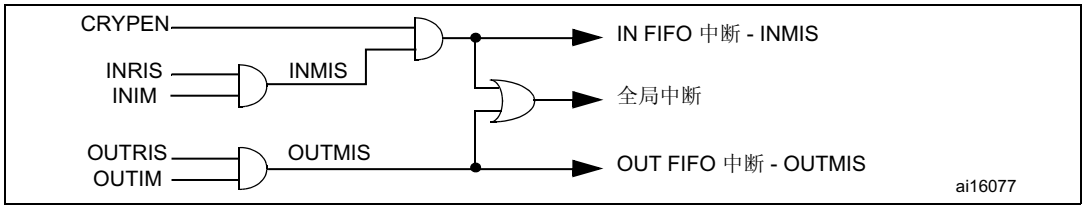
输出 FIFO 服务中断 OUTMIS 不是通过 CRYP 使能位使能。因此，如果输出 FIFO 非空，即使禁止 CRYP 之后也不会强制 OUTMIS 信号为低电平。

输入 FIFO 服务中断 - INMIS

当输入 FIFO 中少于四个字时，会产生输入 FIFO 服务中断。对输入 FIFO 执行写操作直到其中所含字不小于四字，这样即可将此中断清除。

输入 FIFO 服务中断 INMIS 通过 CRYP 使能位使能。因此，禁止 CRYP 之后，即使输入 FIFO 为空，INMIS 信号也为低（无效）。

图 216. CRYP 中断映射图表



20.5 CRYP DMA 接口

加密处理器可使用一个接口连接 DMA 控制器。DMA 操作通过 CRYP DMA 控制寄存器 CRYP_DMACR 进行控制。

突发传输请求信号和单次传输请求信号并不相互排斥。这两种信号可同时产生。例如，当 OUT FIFO 中存在 6 个字时，会产生突发传输请求和单次传输请求。突发传输 4 个字之后，将只产生单次传输请求来传输余下的 2 个字。当数据流中待接收的剩余字数少于突发传输字数时，这一特性非常有用。

在产生相关的 DMA 清除信号之前，仍会产生各个请求信号。禁止请求清除信号之后，可再次激活某个请求信号，具体取决于上述条件。如果已禁止 CRYP 外设并且已将 DMA 使能位清零（CRYP_DMACR 寄存器中的 DIEN 位用于 IN FIFO，DOEN 位用于 OUT FIFO），则会禁止所有请求信号。

注意： *DMA 控制器必须配置为执行不多余 4 字的突发传输。否则可能会丢失一些数据。*

为了在装满 IN FIFO 之前让 DMA 控制器清空 OUT FIFO，OUTDMA 通道的优先级应高于 INDMA 通道。

20.6 CRYP 寄存器

加密内核连接若干控制和状态寄存器、八个密钥寄存器和四个初始化向量寄存器。

20.6.1 用于 STM32F405xx/07xx 和 STM32F415xx/17xx 的 CRYP 控制寄存器 (CRYP_CR)

CRYP control register

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
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CRYPEN	FFLUSH	Reserved				KEYSIZE		DATATYPE		ALGOMODE[2:0]			ALGODIR	Res.	Res.
rw	w					rw	rw	rw	rw	rw	rw	rw	rw		

位 31:18 保留，必须保持复位值

位 17:16 保留，必须保持复位值

位 15 **CRYPEN**：加密处理器使能 (Cryptographic processor enable)

0：CRYP 处理器禁止
1：CRYP 处理器使能

注意：当密钥准备过程结束 (ALGOMODE=111b) 或执行 GCM_CCM *初始化阶段，硬件会自动将 CRYPEN 位清零*

位 14 **FFLUSH**：FIFO 刷新 (FIFO flush)

CRYPEN = 0 时，在此位中写入 1 可刷新 IN 和 OUT FIFO（即，将复位 FIFO 的读指针和写指针）。若在此位中写入 0 则不会产生影响。
CRYPEN = 1 时，在该位中写入 0 或 1 都不会产生影响。
读取该位时，始终返回 0。

位 13:10 保留，必须保持复位值

位 9:8 **KEYSIZE[1:0]**：密钥大小选择 (Key size selection)（仅限 AES 模式）

此位字段定义了 AES 加密内核使用的密钥位长度。此位字段在 DES 或 TDES 模式中可忽略。
00：128 位密钥长度
01：192 位密钥长度
10：256 位密钥长度
11：保留，不使用该值

位 7:6 **DATATYPE[1:0]**：数据类型选择 (Data type selection)

此位字段定义了在 CRYP_DIN 寄存器中输入的数据格式（请参见*第 20.3.3 节：数据类型*）。
00：32 位数据。不交换各个字。压入 IN FIFO（或 OUT FIFO 弹出）的首个字形成数据块的位 1...32，第二个字形成数据块的位 33...64。
01：16 位数据或半字。可将压入 IN FIFO（或 OUT FIFO 弹出）的每个字视为 2 个已相互交换的半字。
10：8 位数据或字节。可将压入 IN FIFO（或 OUT FIFO 弹出）的每个字视为 4 个已相互交换的字。
11：位数据或位串。可将压入 IN FIFO（或 OUT FIFO 弹出）的每个字视为 32 个已相互交换的位（字符串的首个位在位置 0）。

位 5:3 **ALGOMODE[2:0]**：算法模式 (Algorithm mode)

000：TDES-ECB（三重 DES 电子密码本）：数据块之间无反馈。不使用初始化向量 (CRYP_IV0(L/R))，而是使用三个密钥向量（K1、K2 和 K3）（不使用 K0）。

001：TDES-CBC（三重 DES 加密分组链接）：输出块会与后续输入块先进行异或运算，然后再执行相应算法。必须对初始化向量 (CRYP_IV0L/R) 进行初始化，使用三个密钥向量（K1、K2 和 K3）（不使用 K0）。

010：DES-ECB（简易 DES 电子密码本）：数据块之间无反馈。不使用初始化向量 (CRYP_IV0L/R)，仅使用一个密钥向量 (K1)（不使用 K0、K2 和 K3）。

011：DES-CBC（简易 DES 加密分组链接）：输出块会与后续输入块先进行异或运算，然后再执行相应算法。必须对初始化向量 (CRYP_IV0L/R) 进行初始化。仅使用一个密钥向量 (K1)（不使用 K0、K2 和 K3）。

100：AES-ECB（AES 电子密码本）：数据块之间无反馈。不使用初始化向量 (CRYP_IV0L/R...1L/R)。使用所有四个密钥向量 (K0...K3)。

101：AES-CBC（AES 加密分组链接）：输出块会与后续输入块先进行异或运算，然后再执行相应算法。必须对初始化向量 (CRYP_IV0L/R...1L/R) 进行初始化。使用所有四个密钥向量 (K0...K3)。

110：AES-CTR（AES 计数器模式）：输出块会与后续输入块先进行异或运算，然后再执行相应算法。必须对初始化向量 (CRYP_IV0L/R...1L/R) 进行初始化。使用所有四个密钥向量 (K0...K3)。CTR 解密与 CTR 加密并无不同，因为内核始终会对当前的计数器块加密以产生密钥数据流，该密钥数据流与输入中的明文或密码进行异或运算。因此，ALGOMODE = 110b 时 ALGODIR 可忽略，而且不得提供（准备）解密密钥。

111：针对解密模式准备 AES 密钥。CRYPEN = 1 时写入该值会立即启动 AES 轮来准备密钥。秘密密钥必须已预先加载到 K0...K3 寄存器。准备密钥期间会将 CRYP_SR 寄存器的 BUSY 位置 1。完成密钥处理之后，得到的密钥会复制到 K0...K3 寄存器，并且 BUSY 位会清零。

位 2 **ALGODIR**：算法方向 (Algorithm direction)

0：加密

1：解密

位 1:0 保留，必须保持复位值

注意： *当 BUSY=1 时写入 KEYSIZE、DATATYPE、ALGOMODE 和 ALGODIR 位无效。这些位只能在 BUSY=0 时进行配置。*

FFLUSH 位必须在 BUSY=0 时才能置 1。如果在其他情况下将 FFLUSH 位置 1，虽然 FIFO 会进行刷新，但在刷新操作后可能会将所处理的数据块压入输出 FIFO，从而产生 FIFO 非空的情况。

20.6.2 用于 STM32F42xxx 和 STM32F43xxx 的 CRYP 控制寄存器 (CRYP_CR)

CRYP control register

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
Reserved												ALGO MODE [3]	Res.	GCM_CCMPH	
												rw		rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CRYPEN	FFLUSH	Reserved				KEYSIZE		DATATYPE		ALGOMODE[2:0]			ALGODIR	Reserved	
rw	w					rw	rw	rw	rw	rw	rw	rw	rw		

位 31:20 保留，由硬件强制清零。

位 18 保留，由硬件强制清零。

位 17:16 **GCM_CCMPH[1:0]**：如果未设置“GCM 或 CCM 算法”则该位无影响

00：GCM_CCM 初始化阶段
01：GCM_CCM 标头阶段
10：GCM_CCM 有效负载阶段
11：GCM_CCM 最后阶段

位 15 **CRYPEN**：加密处理器使能 (Cryptographic processor enable)

0：CRYP 处理器禁止
1：CRYP 处理器使能

注意：当密钥准备过程结束 (ALGOMODE=111b) 或执行 GCM_CCM *初始化阶段，硬件会自动将 CRYPEN 位清零*

位 14 **FFLUSH**：FIFO 刷新 (FIFO flush)

CRYPEN = 0 时，在此位中写入 1 可刷新 IN 和 OUT FIFO（即，将复位 FIFO 的读指针和写指针）。若在此位中写入 0 则不会产生影响。
CRYPEN = 1 时，在该位中写入 0 或 1 都不会产生影响。
读取该位时，始终返回 0。

位 13:10 保留，由硬件强制清零。

位 9:8 **KEYSIZE[1:0]**：密钥大小选择 (Key size selection)（仅限 AES 模式）

此位字段定义了 AES 加密内核使用的密钥位长度。此位字段在 DES 或 TDES 模式中可忽略。
00：128 位密钥长度
01：192 位密钥长度
10：256 位密钥长度
11：保留，不使用该值

位 7:6 **DATATYPE[1:0]**：数据类型选择 (Data type selection)

此位字段定义了在 CRYP_DIN 寄存器中输入的数据格式（请参见*第 20.3.3 节：数据类型*）。
00：32 位数据。不交换各个字。压入 IN FIFO（或 OUT FIFO 弹出）的首个字形成数据块的位 1...32，第二个字形成数据块的位 33...64。
01：16 位数据或半字。可将压入 IN FIFO（或 OUT FIFO 弹出）的每个字视为 2 个已相互交换的半字。
10：8 位数据或字节。可将压入 IN FIFO（或 OUT FIFO 弹出）的每个字视为 4 个已相互交换的字。
11：位数据或位串。可将压入 IN FIFO（或 OUT FIFO 弹出）的每个字视为 32 个已相互交换的位（字符串的首个位在位置 0）。

位 19 和 5:3 **ALGOMODE[3:0]**：算法模式 (Algorithm mode)

0000：TDES-ECB（三重 DES 电子密码本）：数据块之间无反馈。不使用初始化向量 (CRYP_IV0(L/R))，而是使用三个密钥向量（K1、K2 和 K3）（不使用 K0）。

0001：TDES-CBC（三重 DES 加密分组链接）：输出块会与后续输入块先进行异或运算，然后再执行相应算法。必须对初始化向量 (CRYP_IV0L/R) 进行初始化，使用三个密钥向量（K1、K2 和 K3）（不使用 K0）。

0010：DES-ECB（简易 DES 电子密码本）：数据块之间无反馈。不使用初始化向量 (CRYP_IV0L/R)，仅使用一个密钥向量 (K1)（不使用 K0、K2 和 K3）。

0011：DES-CBC（简易 DES 加密分组链接）：输出块会与后续输入块先进行异或运算，然后再执行相应算法。必须对初始化向量 (CRYP_IV0L/R) 进行初始化。仅使用一个密钥向量 (K1)（不使用 K0、K2 和 K3）。

0100：AES-ECB（AES 电子密码本）：数据块之间无反馈。不使用初始化向量 (CRYP_IV0L/R...1L/R)。使用所有四个密钥向量 (K0...K3)。

0101：AES-CBC（AES 加密分组链接）：输出块会与后续输入块先进行异或运算，然后再执行相应算法。必须对初始化向量 (CRYP_IV0L/R...1L/R) 进行初始化。使用所有四个密钥向量 (K0...K3)。

0110：AES-CTR（AES 计数器模式）：输出块会与后续输入块先进行异或运算，然后再执行相应算法。必须对初始化向量 (CRYP_IV0L/R...1L/R) 进行初始化。使用所有四个密钥向量 (K0...K3)。CTR 解密与 CTR 加密并无不同，因为内核始终会对当前的计数器块加密以产生密钥数据流，该密钥数据流与输入中的明文或密码进行异或运算。因此，ALGOMODE = 110b 时 ALGODIR 可忽略，而且不得提供（准备）解密密钥。

0111：针对解密模式准备 AES 密钥。CRYPEN = 1 时写入该值会立即启动 AES 轮来准备密钥。秘密密钥必须已预先加载到 K0...K3 寄存器。准备密钥期间会将 CRYP_SR 寄存器的 BUSY 位置 1。完成密钥处理之后，得到的密钥会复制到 K0...K3 寄存器，并且 BUSY 位会清零。

1000：Galois 计数器模式 (GCM)。此算法模式同样适用于 GMAC 算法。

1001：CBC-MAC 计数器模式 (CCM)。此算法模式同样适用于 CMAC 算法。

位 2 **ALGODIR**：算法方向 (Algorithm direction)

0：加密

1：解密

位 1:0 保留，必须保持为零。

注意： *当 BUSY=1 时写入 KEYSIZE、DATATYPE、ALGOMODE 和 ALGODIR 位无效。这些位只能在 BUSY=0 时进行配置。*
FFLUSH 位必须在 BUSY=0 时才能置 1。如果在其他情况下将 FFLUSH 位置 1，虽然 FIFO 会进行刷新，但在刷新操作后可能会将所处理的数据块压入输出 FIFO，从而产生 FIFO 非空的情况。

20.6.3 CRYP 状态寄存器 (CRYP_SR)

CRYP status register

偏移地址：0x04

复位值：0x0000 0003

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved											BUSY	OFFU	OFNE	IFNF	IFEM
											r	r	r	r	r

位 31:5 保留，必须保持复位值

位 4 **BUSY**：忙碌位 (Busy bit)

0：CRYP 内核当前没有处理任何数据。具体原因是：

– CRYP 内核已禁止（CRYP_CR 寄存器的 CRYPEN=0）且最后的处理过程已完成，或者

– CRYP 内核正在等待输入 FIFO 的数据足够多或者输出 FIFO 的自由空间足够大（即，在这两种情况下，DES 中至少有 2 个字，AES 中至少有 4 个字）。

1：CRYP 内核目前正在处理一个数据块或准备密钥（用于 AES 解密）。

位 3 **OFFU**：输出 FIFO 已满 (Output FIFO full)

0：输出 FIFO 未满

1：输出 FIFO 已满

位 2 **OFNE**：输出 FIFO 非空 (Output FIFO not empty)

0：输出 FIFO 为空

1：输出 FIFO 非空

位 1 **IFNF**：输入 FIFO 未满 (Input FIFO not full)

0：输入 FIFO 已满

1：输入 FIFO 未满

位 0 **IFEM**：输入 FIFO 为空 (Input FIFO empty)

0：输入 FIFO 非空

1：输入 FIFO 为空

20.6.4 CRYP 数据输入寄存器 (CRYP_DIN)

CRYP data input register

偏移地址：0x08

复位值：0x0000 0000

CRYP_DIN 寄存器为数据输入寄存器。其宽度为 32 位。最多可将四个 64 位 (TDES) 或两个 128 位 (AES) 明文（加密时）或密文（解密时）块输入到输入 FIFO，一次输入一个 32 位字。

写入 FIFO 的首个字为输入块的 MSB。最后写入的是输入块的 LSB。不论数据交换如何：

- 在 DES/TDES 模式下：块是指从位 1（最左位）到位 64（最右位）的位序列。位 1 对应输入 FIFO 中的首个字的 MSB（位 31），位 64 对应输入 FIFO 中的第二个字的 LSB（位 0）。
- 在 AES 模式下：数据块是指从位 0（最左位）到位 127（最右位）的位序列。位 0 对应写入 FIFO 中的首个字的 MSB（位 31），位 127 对应写入 FIFO 中的第四个字的 LSB（位 0）。

为了满足不同的数据大小，通过配置 CRYP_CR 寄存器中的 DATATYPE 位，写入 CRYP_DIN 寄存器中的数据可在处理之前进行交换。更多详细信息，请参见*第 522 页的第 20.3.3 节：数据类型*。

写入 CRYP_DIN 寄存器时，数据会压入输入 FIFO。如果 DES/TDES 模式下至少有两个 32 位字（或者 AES 模式下至少有四个 32 位字）已压入输入 FIFO，并且输出 FIFO 中至少有 2 个字的自由空间，则 CRYP 引擎会启动加密或解密过程。在 DES/TDES 模式下，此过程会从输入 FIFO 取出两个 32 位字（在 AES 模式下则取出四个 32 位字），并且每一轮处理过程都会将两个 32 位字（在 AES 模式则为 4 个字）传输到输出 FIFO。

读取 CRYP_DIN 寄存器时：

- 如果 CRYPEN = 0，则 FIFO 会弹出数据，之后，返回输入 FIFO 中的数据（从最早的数据（首先读取）到最新的数据（最后读取）。在执行每次读取操作前必须检查 IFEM 标志，以确保 FIFO 非空。
- 如果 CRYPEN = 1，则会返回一个未定义值。

读取 CRYP_DIN 寄存器一次或多次之后，在处理新数据之前必须通过将 FFLUSH 位置 1 来刷新 FIFO。

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

位 31:0 **DATAIN**：数据输入 (Data input)

读取 = 在 CRYPEN = 0 时返回输入 FIFO 内容，否则返回不确定值。

写入 = 写入输入 FIFO。

20.6.5 CRYP 数据输出寄存器 (CRYP_DOUT)

CRYP data output register

偏移地址：0x0C

复位值：0x0000 0000

CRYP_DOUT 寄存器为数据输出寄存器。该寄存器为 32 位宽度的只读寄存器。最多可接收来自输出 FIFO 的四个 64 位（TDES 模式）或两个 128 位（AES 模式）密文（加密时）或明文（解密时）块，一次接收一个 32 位字。

与输入数据相似，输出块的 MSB 是从输出 FIFO 读取的首个字。最后读取的是输出块的 LSB。不论数据交换如何：

- 在 DES/TDES 模式下：位 1（最左位）对应从输出 FIFO 中读出的首个字的 MSB（位 31），位 64（最右位）对应从输出 FIFO 中读出的第二个字的 LSB（位 0）。
- 在 AES 模式下：位 0（最左位）对应从输出 FIFO 中读出的首个字的 MSB（位 31），位 127（最右位）对应从输出 FIFO 中读出的第四个字的 LSB（位 0）。

为了满足不同的数据大小，通过配置 CRYP_CR 寄存器中的 DATATYPE 位，数据可在处理之后进行交换。更多详细信息，请参见*第 522 页的第 20.3.3 节：数据类型*。

读取 CRYP_DOUT 寄存器时，会返回最后输入输出 FIFO 的数据（由读指针指定）。

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

位 31:0 **DATAOUT**：数据输出 (Data output)
读取 = 返回输出 FIFO 中的内容。
写入 = 无操作。

20.6.6 CRYP DMA 控制寄存器 (CRYP_DMACR)

CRYP DMA control register

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

位 31:2 保留，必须保持复位值

位 1 **DOEN**：DMA 输出使能 (DMA output enable)
0：禁止用于传出数据传输的 DMA
1：使能用于传出数据传输的 DMA

位 0 **DIEN**：DMA 输入使能 (DMA input enable)
0：禁止用于传入数据传输的 DMA
1：使能用于传入数据传输的 DMA

20.6.7 CRYP 中断屏蔽置位/清零寄存器 (CRYP_IMSCR)

CRYP interrupt mask set/clear register

偏移地址：0x14

复位值：0x0000 0000

CRYP_IMSCR 寄存器为中断屏蔽置位/清零寄存器。该寄存器为读/写寄存器。执行读取操作时，此寄存器会给出相关中断屏蔽的当前值。向该特定位写入 1 时会设置屏蔽，从而使能要读取的中断。向该位写入 0 时会清除相应的屏蔽。外设复位后，所有位均清零。

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

位 31:2 保留，必须保持复位值

位 1 **OUTIM**：输出 FIFO 服务中断屏蔽 (Output FIFO service interrupt mask)
0：屏蔽输出 FIFO 服务中断
1：不屏蔽输出 FIFO 服务中断

位 0 **INIM**：输入 FIFO 服务中断屏蔽 (Input FIFO service interrupt mask)
0：屏蔽输入 FIFO 服务中断
1：不屏蔽输入 FIFO 服务中断

20.6.8 CRYP 原始中断状态寄存器 (CRYP_RISR)

CRYP raw interrupt status register

偏移地址：0x18

复位值：0x0000 0001

CRYP_RISR 寄存器为原始中断状态寄存器。该寄存器为只读寄存器。执行读取操作时，此寄存器会在屏蔽前给出相关中断的当前原始状态。执行写入操作不起作用。

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

位 31:2 保留，必须保持复位值

位 1 **OUTRIS**：输出 FIFO 服务原始中断状态 (Output FIFO service raw interrupt status)
给出屏蔽输出 FIFO 服务中断之前的原始中断状态。
0：原始中断未挂起
1：原始中断挂起

位 0 **INRIS**：输入 FIFO 服务原始中断状态 (Input FIFO service raw interrupt status)
给出屏蔽输入 FIFO 服务中断之前的原始中断状态。
0：原始中断未挂起
1：原始中断挂起

20.6.9 CRYP 屏蔽中断状态寄存器 (CRYP_MISR)

CRYP masked interrupt status register

偏移地址：0x1C

复位值：0x0000 0000

CRYP_RISR 寄存器为屏蔽中断状态寄存器。该寄存器为只读寄存器。执行读取操作时，此寄存器会在屏蔽前给出相关中断的当前屏蔽状态。执行写入操作不起作用。

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

位 31:2 保留，必须保持复位值

位 1 **OUTMIS**：输出 FIFO 服务屏蔽中断状态 (Output FIFO service masked interrupt status)

给出屏蔽输出 FIFO 服务中断之后的中断状态。
0：中断未挂起
1：中断挂起

位 0 **INMIS**：输入 FIFO 服务屏蔽中断状态 (Input FIFO service masked interrupt status)

给出屏蔽输入 FIFO 服务中断之后的中断状态。
0：中断未挂起
1：CRYPEN = 1 时中断挂起

20.6.10 CRYP 密钥寄存器 (CRYP_K0...3(L/R)R)

CRYP key registers

偏移地址：0x20 到 0x3C

复位值：0x0000 0000

这些寄存器中含有加密密钥。

在 TDES 模式下，密钥为 64 位二进制值（按从左到右编号，最左位为位 1），分别命名为 K1、K2 和 K3（不使用 K0），各个密钥由 56 个信息位和 8 个奇偶校验位组成。奇偶校验位保留用于错误检测，当前块不使用这些位。因此，不使用各个 64 位密钥值 Kx[1:64] 的位 8、16、24、32、40、48、56 和 64。

在 AES 模式下，可将密钥视为一个 128 位、192 位或 256 位长的位序列，$k_0k_1k_2...k_{127/191/255}$（$k_0$ 为最左位）。AES 密钥按如下方式输入寄存器中：

- 对于 AES-128：$k_0..k_{127}$ 对应 $b_{127}..b_0$（不使用 $b_{255}..b_{128}$）
- 对于 AES-192：$k_0..k_{191}$ 对应 $b_{191}..b_0$（不使用 $b_{255}..b_{192}$）
- 对于 AES-256：$k_0..k_{255}$ 对应 $b_{255}..b_0$

在任何情况下，b_0 为最右位。

CRYP_K0LR（偏移地址：0x20）

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
b255	b254	b253	b252	b251	b250	b249	b248	b247	b246	b245	b244	b243	b242	b241	b240
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
b239	b238	b237	b236	b235	b234	b233	b232	b231	b230	b229	b228	b227	b226	b225	b224
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

CRYP_K0RR（偏移地址：0x24）

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
b223	b222	b221	b220	b219	b218	b217	b216	b215	b214	b213	b212	b211	b210	b209	b208
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
b207	b206	b205	b204	b203	b202	b201	b200	b199	b198	b197	b196	b195	b194	b193	b192
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

CRYP_K1LR（偏移地址：0x28）

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
k1.1 b191	k1.2 b190	k1.3 b189	k1.4 b188	k1.5 b187	k1.6 b186	k1.7 b185	k1.8 b184	k1.9 b183	k1.10 b182	k1.11 b181	k1.12 b180	k1.13 b179	k1.14 b178	k1.15 b177	k1.16 b176
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
k1.17 b175	k1.18 b174	k1.19 b173	k1.20 b172	k1.21 b171	k1.22 b170	k1.23 b169	k1.24 b168	k1.25 b167	k1.26 b166	k1.27 b165	k1.28 b164	k1.29 b163	k1.30 b162	k1.31 b161	k1.32 b160
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

CRYP_K1RR（偏移地址：0x2C）

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
k1.33 b159	k1.34 b158	k1.35 b157	k1.36 b156	k1.37 b155	k1.38 b154	k1.39 b153	k1.40 b152	k1.41 b151	k1.42 b150	k1.43 b149	k1.44 b148	k1.45 b147	k1.46 b146	k1.47 b145	k1.48 b144
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
k1.49 b143	k1.50 b142	k1.51 b141	k1.52 b140	k1.53 b139	k1.54 b138	k1.55 b137	k1.56 b136	k1.57 b135	k1.58 b134	k1.59 b133	k1.60 b132	k1.61 b131	k1.62 b130	k1.63 b129	k1.64 b128
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

CRYP_K2LR（偏移地址：0x30）

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
k2.1 b127	k2.2 b126	k2.3 b125	k2.4 b124	k2.5 b123	k2.6 b122	k2.7 b121	k2.8 b120	k2.9 b119	k2.10 b118	k2.11 b117	k2.12 b116	k2.13 b115	k2.14 b114	k2.15 b113	k2.16 b112
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
k2.17 b111	k2.18 b110	k2.19 b109	k2.20 b108	k2.21 b107	k2.22 b106	k2.23 b105	k2.24 b104	k2.25 b103	k2.26 b102	k2.27 b101	k2.28 b100	k2.29 b99	k2.30 b98	k2.31 b97	k2.32 b96
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

CRYP_K2RR（偏移地址：0x34）

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
k2.33 b95	k2.34 b94	k2.35 b93	k2.36 b92	k2.37 b91	k2.38 b90	k2.39 b89	k2.40 b88	k2.41 b87	k2.42 b86	k2.43 b85	k2.44 b84	k2.45 b83	k2.46 b82	k2.47 b81	k2.48 b80
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
k2.49 b79	k2.50 b78	k2.51 b77	k2.52 b76	k2.53 b75	k2.54 b74	k2.55 b73	k2.56 b72	k2.57 b71	k2.58 b70	k2.59 b69	k2.60 b68	k2.61 b67	k2.62 b66	k2.63 b65	k2.64 b64
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

CRYP_K3LR（偏移地址：0x38）

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
k3.1 b63	k3.2 b62	k3.3 b61	k3.4 b60	k3.5 b59	k3.6 b58	k3.7 b57	k3.8 b56	k3.9 b55	k3.10 b54	k3.11 b53	k3.12 b52	k3.13 b51	k3.14 b50	k3.15 b49	k3.16 b48
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
k3.17 b47	k3.18 b46	k3.19 b45	k3.20 b44	k3.21 b43	k3.22 b42	k3.23 b41	k3.24 b40	k3.25 b39	k3.26 b38	k3.27 b37	k3.28 b36	k3.29 b35	k3.30 b34	k3.31 b33	k3.32 b32
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

CRYP_K3RR（偏移地址：0x3C）

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
k3.33 b31	k3.34 b30	k3.35 b29	k3.36 b28	k3.37 b27	k3.38 b26	k3.39 b25	k3.40 b24	k3.41 b23	k3.42 b22	k3.43 b21	k3.44 b20	k3.45 b19	k3.46 b18	k3.47 b17	k3.48 b16
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
k3.49 b15	k3.50 b14	k3.51 b13	k3.52 b12	k3.53 b11	k3.54 b10	k3.55 b9	k3.56 b8	k3.57 b7	k3.58 b6	k3.59 b5	k3.60 b4	k3.61 b3	k3.62 b2	k3.63 b1	k3.64 b0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

注意： *当加密处理器繁忙时（CRYP_SR 寄存器中的位 BUSY = 1），会忽略对这些寄存器的写访问。*

20.6.11 CRYP 初始化向量寄存器 (CRYP_IV0...1(L/R)R)

CRYP initialization vector registers

偏移地址：0x40 到 0x4C

复位值：0x0000 0000

CRYP_IV0...1(L/R)R 是初始化向量专用的左字和右字寄存器（DES/TDES 模式下为 64 位，AES 模式下为 128 位），可在 CBC（加密分组链接）和计数器 (CTR) 模式下使用。完成每一轮的 TDES 或 AES 内核计算后，CRYP_IV0...1(L/R)R 寄存器会按 *第 510 页的DES 和 TDES 密码块链接 (DES/TDES-CBC) 模式*一节、*第 514 页的AES 加密分组链接 (AES-CBC) 模式*一节和 *第 516 页的AES 计数器模式 (AES-CTR) 模式*一节所述进行更新。

IV0 是初始化向量的最左位，而 IV63（DES，TDES）或 IV127 (AES) 是初始化向量的最右位。IV1(L/R)R 只能在 AES 模式中使用。

CRYP_IV0LR（偏移地址：0x40）

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IV0	IV1	IV2	IV3	IV4	IV5	IV6	IV7	IV8	IV9	IV10	IV11	IV12	IV13	IV14	IV15
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
IV16	IV17	IV18	IV19	IV20	IV21	IV22	IV23	IV24	IV25	IV26	IV27	IV28	IV29	IV30	IV31
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

CRYP_IV0RR（偏移地址：0x44）

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IV32	IV33	IV34	IV35	IV36	IV37	IV38	IV39	IV40	IV41	IV42	IV43	IV44	IV45	IV46	IV47
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
IV48	IV49	IV50	IV51	IV52	IV53	IV54	IV55	IV56	IV57	IV58	IV59	IV60	IV61	IV62	IV63
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

CRYP_IV1LR（偏移地址：0x48）

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IV64	IV65	IV66	IV67	IV68	IV69	IV70	IV71	IV72	IV73	IV74	IV75	IV76	IV77	IV78	IV79
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
IV80	IV81	IV82	IV83	IV84	IV85	IV86	IV87	IV88	IV89	IV90	IV91	IV92	IV93	IV94	IV95
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

CRYP_IV1RR（偏移地址：0x4C）

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IV96	IV97	IV98	IV99	IV100	IV101	IV102	IV103	IV104	IV105	IV106	IV107	IV108	IV109	IV110	IV111
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
IV112	IV113	IV114	IV115	IV116	IV117	IV118	IV119	IV120	IV121	IV122	IV123	IV124	IV125	IV126	IV127
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

注意： *在 DES/3DES 模式下，仅使用 CRYP_IV0(L/R)。*

当加密处理器繁忙时（CRYP_SR 寄存器中的位 BUSY = 1），会忽略对这些寄存器的写访问。

20.6.12 用于 STM32F42xxx 和 STM32F43xxx 的 CRYP 上下文交换寄存器（CRYP_CSGCMCCM0..7R 和 CRYP_CSGCM0..7R）

CRYP context swap registers

偏移地址：

- CRYP_CSGCMCCM0..7：0x050 到 0x06C：仅用于 GCM/GMAC 或 CCM/CMAC 算法
- CRYP_CSGCM0..7：0x070 到 0x08C：仅用于 GCM/GMAC 算法

复位值：0x0000 0000

在选择 GCM/GMAC 或 CCM/CMAC 算法后，这些寄存器含有 CRYP 处理器的全部内部寄存器状态。如果高优先级的任务需要使用加密处理器，而该处理器正在执行其他任务，则需要使用上述寄存器执行上下文交换。

出现此类事件时，需要对 CRYP_CSGCMCCM0..7R 和 CRYP_CSGCM0..7R（GCM/GMAC 模式）或 CRYP_CSGCMCCM0..7R（CCM/CMAC 模式）寄存器进行读操作，并且需要将读出的值保存到系统存储器空间。之后，可使用加密处理器执行高优先级的任务，完成加密计算之后，可从存储器中读出保存的上下文并将其写入相应的上下文交换寄存器。

注意： *这些寄存器仅在选择了 GCM/GMAC 或 CCM/CMAC 算法模式时才可使用。*

CRYP_CSGCMCCMxR：其中 x=[7:0]

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

CRYP_CSGCMxR：其中 x=[7:0]

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

20.6.13 CRYP 寄存器映射

表 92. 用于 STM32F405xx/07xx 和 STM32F415xx/17xx 的 CRYP 寄存器映射和复位值

偏移	寄存器名称 复位值	寄存器大小 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00 0x00	CRYP_CR	Reserved																CRYPEN	FFLUSH	Reserved				KEYSIZE		DATATYPE		ALOMODE[2:0]			ALGODIR	Res.	
	Reset value																	0	0					0	0	0	0	0	0	0	0		
0x04	CRYP_SR	Reserved																											BUSY	OFFU	OFNE	IFNF	IFEM
	Reset value																												0	0	0	1	1
0x08	CRYP_DIN	DATAIN																															
	Reset value																				0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	CRYP_DOUT	DATAOUT																															
	Reset value																				0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	CRYP_DMACR	Reserved																														DOEN	DIEN
	Reset value																															0	0
0x14	CRYP_IMSCR	Reserved																														OUTIM	INIM
	Reset value																															0	0
0x18	CRYP_RISR	Reserved																														OUTRIS	INRIS
	Reset value																															0	1
0x1C	CRYP_MISR	Reserved																														OUTMIS	IN%IS
	Reset value																															0	0
0x20	CRYP_K0LR	CRYP_K0LR																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x24	CRYP_K0RR	CRYP_K0RR																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
...																																	
0x38	CRYP_K3LR	CRYP_K3LR																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x3C	CRYP_K3RR	CRYP_K3RR																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x40	CRYP_IV0LR	CRYP_IV0LR																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x44	CRYP_IV0RR	CRYP_IV0RR																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x48	CRYP_IV1LR	CRYP_IV1LR																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x4C	CRYP_IV1RR	CRYP_IV1RR																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

表 93. 用于 STM32F42xxx 和 STM32F43xxx 的 CRYP 寄存器映射和复位值

偏移	寄存器名称 复位值	寄存器大小 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00 0x00	CRYP_CR	Reserved												ALGOMODE[3]	Res.	GCM_CCMPH		CRYPEN	FFLUSH	Reserved				KEYSIZE		DATATYPE		ALOMODE[2:0]			ALGODIR	Res..	
	Reset value															0	0	0	0					0	0	0	0	0	0	0	0		
0x04	CRYP_SR	Reserved																											BUSY	OFFU	OFNE	IFNF	IFEM
	Reset value																												0	0	0	1	1
0x08	CRYP_DIN	DATAIN																															
	Reset value																				0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	CRYP_DOUT	DATAOUT																															
	Reset value																				0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	CRYP_DMACR	Reserved																														DOEN	DIEN
	Reset value																															0	0
0x14	CRYP_IMSCR	Reserved																														OUTIM	INIM
	Reset value																															0	0
0x18	CRYP_RISR	Reserved																														OUTRIS	INRIS
	Reset value																															0	1
0x1C	CRYP_MISR	Reserved																														OUTMIS	IN%IS
	Reset value																															0	0
0x20	CRYP_K0LR	CRYP_K0LR																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x24	CRYP_K0RR	CRYP_K0RR																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
...																																	
0x38	CRYP_K3LR	CRYP_K3LR																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x3C	CRYP_K3RR	CRYP_K3RR																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x40	CRYP_IV0LR	CRYP_IV0LR																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x44	CRYP_IV0RR	CRYP_IV0RR																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x48	CRYP_IV1LR	CRYP_IV1LR																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x4C	CRYP_IV1RR	CRYP_IV1RR																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x50	CRYP_CSGCMCCMR	CRYP_CSGCMCCM0R																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x54	CRYP_CSGCMCCM1R	CRYP_CSGCMCCM1R																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x58	CRYP_CSGCMCCM2R	CRYP_CSGCMCCM2R																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x5C	CRYP_CSGCMCCM3R	CRYP_CSGCMCCM3R																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

表 93. 用于 STM32F42xxx 和 STM32F43xxx 的 CRYP 寄存器映射和复位值 （续）

偏移	寄存器名称 复位值	寄存器大小																													
		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x60	CRYP_CSGCMCCM4R	CRYP_CSGCMCCM4R																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x64	CRYP_CSGCMCCM5R	CRYP_CSGCMCCM5R																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x68	CRYP_CSGCMCCM6R	CRYP_CSGCMCCM6R																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x6C	CRYP_CSGCMCCM7R	CRYP_CSGCMCCM7R																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x70	CRYP_CSGCM0R	CRYP_CSGCM0R																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x74	CRYP_CSGCM1R	CRYP_CSGCM1R																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x78	CRYP_CSGCM2R	CRYP_CSGCM2R																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x7C	CRYP_CSGCM3R	CRYP_CSGCM3R																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x80	CRYP_CSGCM4R	CRYP_CSGCM4R																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x84	CRYP_CSGCM5R	CRYP_CSGCM5R																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x88	CRYP_CSGCM6R	CRYP_CSGCM6R																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x8C	CRYP_CSGCM7R	CRYP_CSGCM7R																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

有关寄存器边界地址的信息，请参见*第 52 页的表 2*。

21 随机数发生器 (RNG)

除非特别说明，否则本部分适用于整个 STM32F4xx 系列器件。

21.1 RNG 简介

RNG 处理器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。

RNG 已通过 FIPS PUB 140-2（2001 年 10 月 10 日）测试，成功率达 99%。

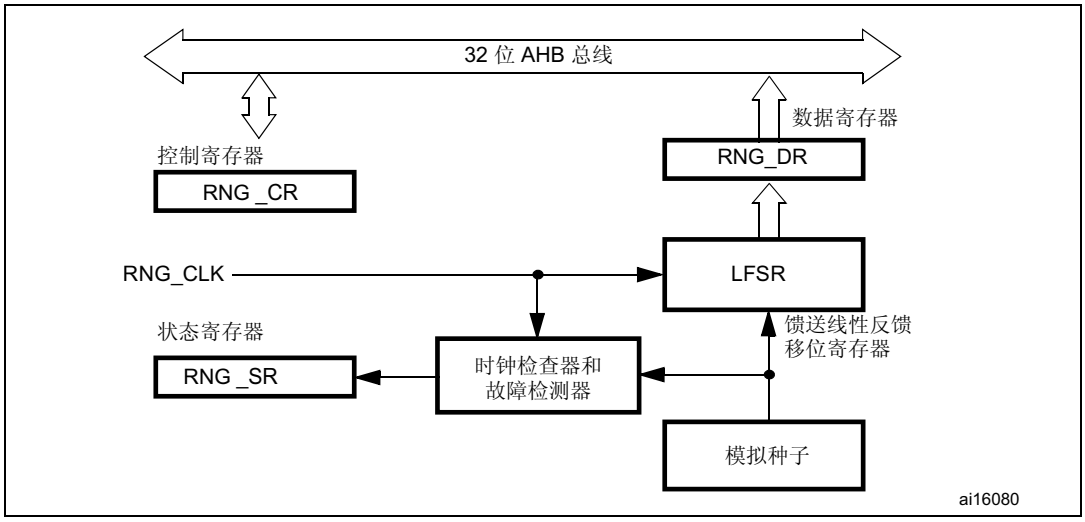
21.2 RNG 主要特性

- 提供由模拟量发生器产生的 32 位随机数
- 两个连续随机数的间隔为 40 个 PLL48CLK 时钟信号周期
- 通过监视 RNG 熵来标识异常行为（产生稳定值，或产生稳定的值序列）
- 可被禁止以降低功耗

21.3 RNG 功能说明

图 217 显示了 RNG 框图。

图 217. 框图



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 (RNG_LFSR) 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。RNG_LFSR 由专用时钟 (PLL48CLK) 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 RNG_LFSR 后，RNG_LFSR 的内容会传入数据寄存器 (RNG_DR)。

同时，系统会监视模拟种子和专用时钟 PLL48CLK。状态位（RNG_SR 寄存器中）指示何时在种子上出现异常序列，或指示何时 PLL48CLK 时钟频率过低。检测到错误时生成中断。

21.3.1 操作

要运行 RNG，请按以下步骤操作：

1. 如果需要，使能中断（为此，将 RNG_CR 寄存器中的 IE 位置 1）。准备好随机数时或出现错误时生成中断。
2. 通过将 RNG_CR 寄存器中的 RNGEN 位置 1 使能随机数产生。这会激活模拟部分、RNG_LFSR 和错误检测器。
3. 每次中断时，检查确认未出现错误（RNG_SR 寄存器中的 SEIS 和 CEIS 位应为 0），并且随机数已准备就绪（RNG_SR 寄存器中的 DRDY 位为 1）。然后即可读取 RNG_DR 寄存器中的内容。

按照 FIPS PUB（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）。

21.3.2 错误管理

如果 CEIS 位的值为 1（时钟错误）

出现时钟错误时，RNG 无法再产生随机数，因为 PLL48CLK 时钟不正确。检查时钟控制器是否正确配置，是否可提供 RNG 时钟，然后将 CEIS 位清零。当 CECS 位为 0 时，RNG 可正常工作。时钟错误对产生的上一个随机数没有影响，因此 RNG_DR 寄存器内容可以使用。

如果 SEIS 位的值为 1（种子错误）

出现种子错误时，只要 SECS 位为 1，就会中断随机数产生。如果 RNG_DR 寄存器中有可用随机数，不能使用该随机数，因为它可能没有足够的熵。

应执行以下操作：将 SEIS 位清零，然后将 RNGEN 位清零并置 1，以便重新初始化和重新启动 RNG。

21.4 RNG 寄存器

RNG 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。

21.4.1 RNG 控制寄存器 (RNG_CR)

RNG control register

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

位 31:4 保留，必须保持复位值

位 3 **IE**：中断使能 (Interrupt enable)

0：禁止 RNG 中断。

1：使能 RNG 中断。只要 RNG_SR 寄存器中 DRDY=1 或 SEIS=1 或 CEIS=1，就会挂起中断。

位 2 **RNGEN**：随机数发生器使能 (Random number generator enable)

0：禁止随机数发生器。

1：使能随机数发生器。

位 1:0 保留，必须保持复位值

21.4.2 RNG 状态寄存器 (RNG_SR)

RNG status register

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
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved									SEIS	CEIS	Reserved		SECS	CECS	DRDY
									rc_w0	rc_w0			r	r	r

位 31:3 保留，必须保持复位值

位 6 **SEIS**：种子错误中断状态 (Seed error interrupt status)

此位与 SECS 同时设置，通过向其写入 0 来清零。

0：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)

如果 RNG_CR 寄存器中 IE = 1，则会挂起中断。

位 5 **CEIS**：时钟错误中断状态 (Clock error interrupt status)

此位与 CECS 同时设置，通过向其写入 0 来清零。

0：正确检测到 PLL48CLK 时钟

1：未正确检测到 PLL48CLK 时钟 ($f_{PLL48CLK} < f_{HCLK}/16$)

如果 RNG_CR 寄存器中 IE = 1，则会挂起中断。

位 4:3 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。如果 SEIS 位置 1，则意味着已检测到错误序列并已恢复正常。

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 **CECS**：时钟错误当前状态 (Clock error current status)

0：正确检测到 PLL48CLK 时钟。如果 CEIS 位置 1，则意味着已检测到时钟错误并已恢复正常。

1：未正确检测到 PLL48CLK 时钟 ($f_{PLL48CLK} < f_{HCLK}/16$)。

位 0 **DRDY**：数据就绪 (Data ready)

0：RNG_DR 寄存器尚未有效，无可用随机数据

1：RNG_DR 寄存器包含有效随机数据

注意：如果 RNG_CR 寄存器中 IE = 1，则会挂起中断。

读取 RNG_DR 寄存器后，此位恢复到 0，直到计算出新的有效值。

21.4.3 RNG 数据寄存器 (RNG_DR)

RNG data register

偏移地址：0x08

复位值：0x0000 0000

RNG_DR 寄存器是只读寄存器，在读取时提供 32 位随机数值。读取后，此寄存器在最多 40 个 PLL48CLK 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 位是否已置 1。

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

位 31:0 **RNDATA**：随机数据 (Random data)

32 位随机数据。

21.4.4 RNG 寄存器映射

表 94 给出了 RNG 寄存器映射和复位值。

表 94. RNG 寄存器映射和复位映射

偏移	寄存器名称 复位值	寄存器大小 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
0x00	RNG_CR 0x0000000	Reserved … IE, RNGEN, Reserved
0x04	RNG_SR 0x0000000	Reserved … SEIS, CEIS, Reserved, SECS, CECS, DRDY
0x08	RNG_DR 0x0000000	RNDATA[31:0]

22 散列处理器 (HASH)

除非特别说明，否则本部分适用于整个 STM32F4xx 系列器件。

22.1 散列简介

散列处理器完全兼容安全散列算法（SHA-1、SHA-224 和 SHA-256）、MD5（消息摘要算法 5）散列算法和适合多种应用的 HMAC（密钥散列消息认证码）散列算法。对长达 ($2^{64} - 1$) 位的消息，散列处理器计算消息摘要（SHA-1 算法为 160 位，SHA-256 算法为 256 位，SHA-224 算法为 224 位，MD5 算法为 128 位），而 HMAC 算法则通过散列函数来对消息进行验证。HMAC 算法存在于两次调用 SHA-1、SHA-224、SHA-256 或 MD5 散列函数。

22.2 散列主要特性

- 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-1、SHA-224 和 SHA-256）
  - IETF RFC 1321（互联网工程任务组征求意见文档编号 1321）规范 (MD5)
- 快速计算 SHA-1、SHA-224、SHA-256 以及 MD5（仅 STM32F42xxx 和 STM32F43xxx 中可使用 SHA-224 和 SHA-256）
- AHB 从外设
- 32 位数据字用于输入数据，支持字、半字、字节和位位串表示法（仅采用小端模式数据表示法）
- 可自动交换，以兼容大端模式 SHA1、SHA-224 和 SHA-256 计算标准（采用小端模式输入位串表示法）
- 可自动填充来完成输入位串，从而适应模数为 512（16 × 32 位）消息摘要计算
- STM32F405xx/07xx 和 STM32F415xx/17xx 上的 5× 32 位字（H0 到 H5）和 STM32F42xxx 和 STM32F43xxx 上的 8 x 32 位字（H0 到 H7）用于输出摘要，重载可继续被打断的消息摘要计算
- 连续消息块中摘要的对应 32 位字添加到彼此之中，以构成整个消息的摘要
- 数据流自动控制，支持直接存储器访问 (DMA)

注意： *SHA-1*、SHA-224 和 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。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。

22.3 散列功能说明

图 218 显示了散列处理器的框图。

图 218. STM32F405xx/07xx 和 STM32F415xx/17xx 的框图

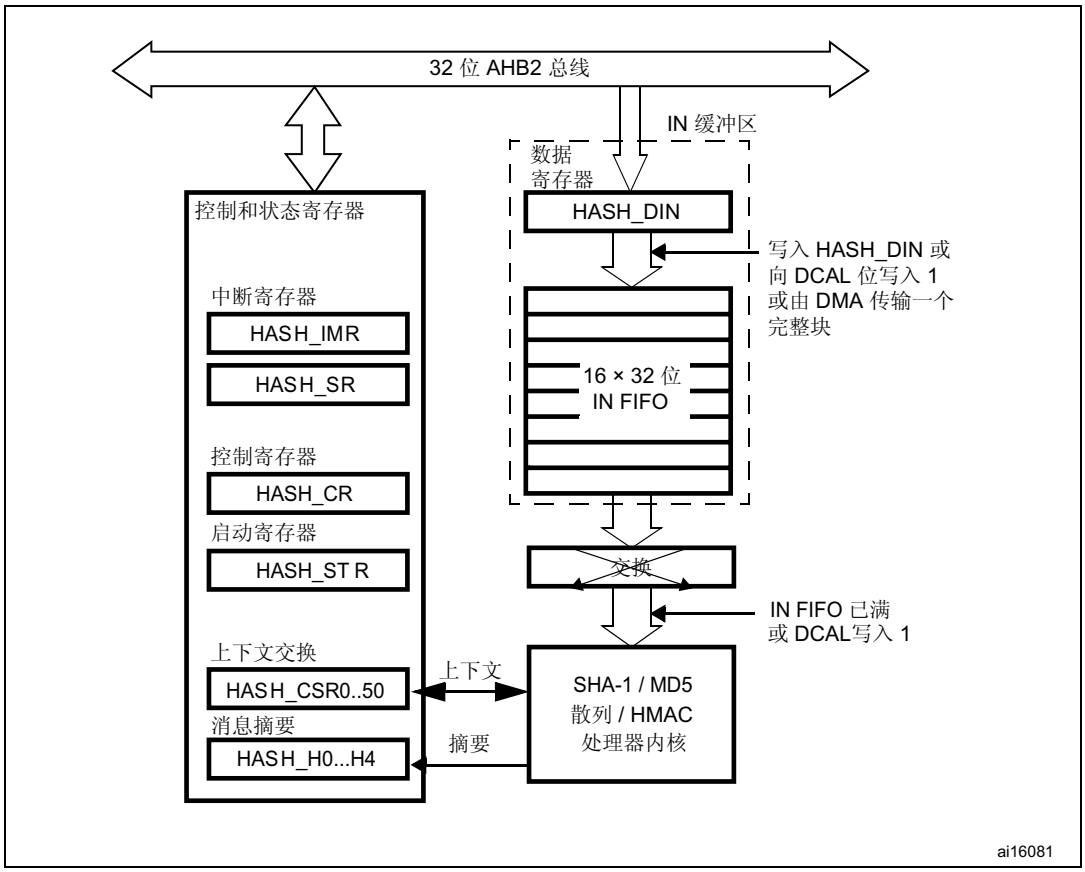
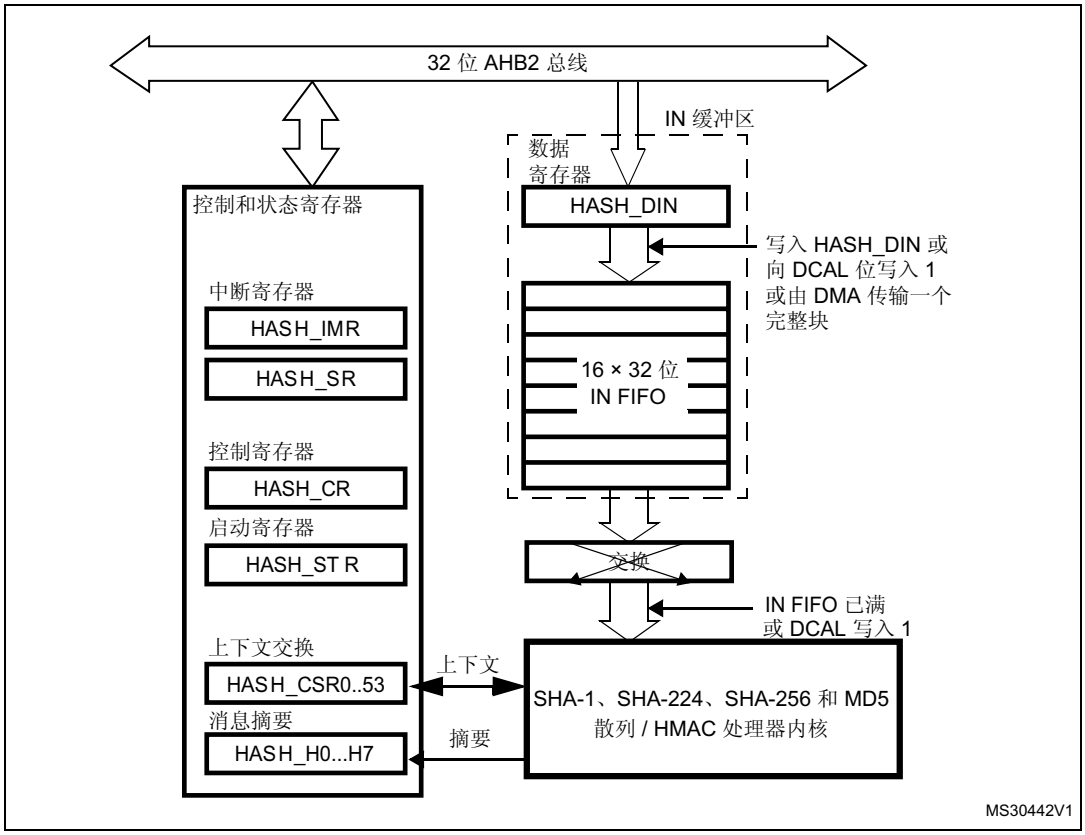


图 219. 框图 STM32F42xxx 和 STM32F43xxx



针对计算消息或数据文件的压缩表示，FIPS PUB 180-2 标准和 IETF RFC 1321 出版物分别详细说明了 SHA-1、SHA-224 和 SHA-256 以及 MD5 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-1、SHA-224 和 SHA-256 以及 MD5 将分别生成一个 160 位、224 位、256 位和 128 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通常可提高流程的效率，因为消息摘要通常比消息要小得多。数字签名的验证程序和数字签名的创建程序必须使用相同散列算法。

SHA-1、SHA-224 和 SHA-256 以及 MD5 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-1 或 SHA-224 和 SHA-256 算法的详细信息，请参见 2002 年 8 月 1 日颁布的 FIPS PUB 180-2（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，FIPS PUB 180-1 标准使用如下约定：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（请参见*第 12 页的第 1.3.4 节：消息填充*）。

22.3.1 处理的持续时间

计算消息的中间块所花费的时间是：

- 在 SHA-1 中，为 66 个 HCLK 时钟周期
- 在 SHA-224 中，为 50 个 HCLK 时钟周期
- 在 SHA-256 中，为 50 个 HCLK 时钟周期
- 在 MD5 中，为 50 个 HCLK 时钟周期

必须将上述时间与将块的 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。

处理消息的最后一个块（或者 HMAC 中某个密钥的最后一个块）所需要的时间可能会更长。此时间取决于最后一个块的长度以及密钥的大小（在 HMAC 模式中）。与处理中间块相比，此时间可能按如下某个系数增加：

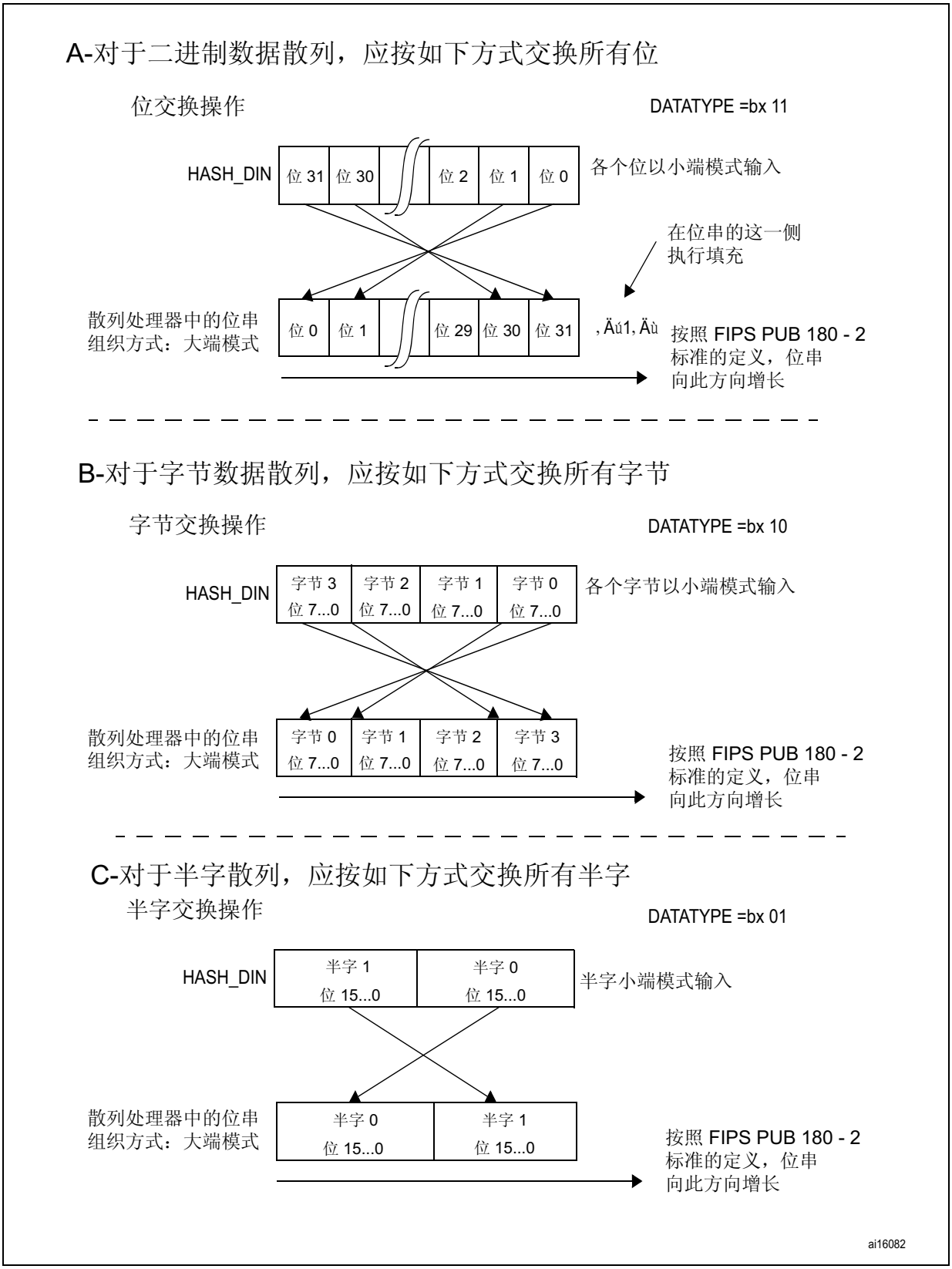
- 对于散列消息，该系数为 1 到 2.5
- 对于 HMAC 输入密钥，该系数在 2.5 左右
- 对于 HMAC 消息，该系数为 1 到 2.5
- 对于 HMAC 输出密钥，如果是短密钥，则该系数在 2.5 左右
- 对于 HMAC 输出密钥，如果是长密钥，则该系数为 3.5 到 5

22.3.2 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 SHA1、SHA-224 和 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 位字段可配置要处理的数据类型。

图 220. 位、字节和半字交换



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置 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置 0，依此类推。

22.3.3 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 16 x 32 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 (IN FIFO)。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 IN FIFO 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。
  - 如果是单个 DMA 传输：在 STM32F42xxx 和 STM32F43xxx 上应将多个 DMA 传输 (MDMAT) 位清零。如果最后一个块已通过 DMA 通道传输到 HASH_DIN 寄存器，则 HASH_STR 寄存器中的 DCAL 位将自动置 1，以启动最终摘要计算。
  - 如果是多个 DMA 传输（仅在 STM32F42xxx 和 STM32F43xxx 中可用）：应该由软件将多个 DMA 传输 (MDMAT) 位置为 1，所以硬件无法将 DCAL 位自动置 1，在这种情况下，散列的最终摘要计算以及 HMAC 的每个阶段的最终摘要计算在 DMA 传输请求结束时将不会启动，因此处理器便可以接收新的 DMA 传输（有关 HMAC 阶段的详细信息，请参见“HMAC 运算”部分）。在最后 DMA 传输期间，应由软件将多个 DMA 传输 (MDMAT) 位清零，以便在最后一个块结束时将 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 突寄存器的 NBLW 位段中，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 已写入到 1 一样。

22.3.4 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，目的是生成一个长度为 512 的填充消息块。“1”将添加到 HASH_DIN 寄存器中写入的最后一个字，位位置由 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
<-- 写入 HASH_DIN 的第一个字 -->
```

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）：

```
01100001 01100010 01100011 1UUUUUUU
```

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 00000000 00000018。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 00000028
```

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 0x0000 0001 插入一个新字。然后将添加一个全零字 (0x0000 0000) 以及双字表示法格式的消息长度，从而形成 16 x 32 位字的块。
4. 将执行散列计算，并且随后在 HASH_Hx 寄存器 (x = 0...4) 中消息摘要可供 SHA-1 算法使用。例如：

```
H0 = 0xA9993E36
H1 = 0x4706816A
H2 = 0xBA3E2571
H3 = 0x7850C26C
H4 = 0x9CD0D89D
```

22.3.5 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-1、SHA-224、SHA-256 和 MD5）。同时（即 INIT 位置 1 时），使用 ALGO 位来选择算法（SHA-1、SHA-224、SHA-256 或 MD5）。

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-1 算法，散列处理器在 66 个周期内保持忙碌，对于 MD5 算法、SHA-224 算法和 SHA-256 算法，为 50 个周期。

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 *DMA 加载数据的过程*一节。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，在 STM32F405xx/07xx 和 STM32F415xx/17xx 上可以从 HASH_H0 到 HASH_H4 寄存器中读取摘要（对于 MD5 算法 HASH_H4 无关），在 STM32F42xxx 和 STM32F43xxx 在上可以从 HASH_H0 到 HASH_H7 寄存器中读取摘要，其中：

选择了 MD5 算法时，HASH_H4 到 HASH_H7 无关。

选择了 SHA-1 算法时，HASH_H5 到 HASH_H7 无关。

选择了 SHA-224 算法时，HASH_H7 无关。

22.3.6 HMAC 运算

HMAC 以不可逆方式将正在处理的消息与用户所选的密钥进行绑定，从而用于消息验证。有关 HMAC 规范，请参见 H. Krawczyk、M. Bellare 和 R. Canetti 于 1997 年 2 月发表的文章“HMAC：密钥散列消息认证，H. Krawczyk，M. Bellare, R. Canetti，1997 年 2 月”。

基本而言，该算法由两个嵌套的散列运算组成：

```
HMAC(message) = Hash[((key | pad) XOR 0x5C)
          | Hash(((key | pad) XOR 0x36) | message)]
```

其中：

- pad 是将密钥扩展到底层散列函数数据块的长度所需要的零序列（对于 SHA-1、SHA224、SHA-256 和 MD5 散列算法是 512 位）。
- | 表示串联运算符。

要计算 HMAC，需要四个不同阶段：

1. 当 MODE 位为 1 且 ALGO 位段已根据所需算法设好对应的值，通过将 INIT 位置 1 对块进行初始化。如果使用的密钥长度超过 64 位，则在此阶段中还必须将 LKEY 位置 1（在此情况下，HMAC 规范规定密钥的散列应代替真实密钥）。
2. 随后将用于内部散列函数的密钥提供给内核。此操作采用的机制即是用于在散列运算中发送消息的机制（也就是通过写入到 HASH_DIN 并最终写入到 HASH_STR）。
3. 输入了最后一个字并且计算启动后，散列处理器便开始生成密钥。然后便可以使用与在散列运算中发送消息相同的机制接受消息文本。
4. 在第一轮散列之后，散列处理器将返回“就绪”以表明其可以接收用于外部散列函数的密钥（通常，此密钥与用于内部散列函数的密钥相同）。在输入了密钥的最后一个字并且计算开始后，HMAC 结果存储于 HASH_H0...HASH_H4 寄存器（STM32F405xx/07xx 和 STM32F415xx/17xx）或 HASH_H0...HASH_H7 寄存器（STM32F42xxx 和 STM32F43xxx）。

注意： 1 HMAC 本原的计算延迟取决于密钥和消息的长度。可以将 HMAC 视为密钥长度相同（长或短）的两个嵌套底层散列函数。

22.3.7 上下文交换

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
  - 在 STM32F42xxx 和 STM32F43xxx 上是 HASH_CSR0 到 HASH_CSR53；在 STM32F405xx/07xx 和 STM32F415xx/17xx 上是 HASH_CSR0 到 HASH_CSR50。
- 恢复上下文：

  在高优先级任务完成后，可以恢复上下文。请遵循以下顺序。

  a) 将存储器中保存的值写入以下寄存器中：HASH_IMR、HASH_STR 和 HASH_CR

  b) 通过将 HASH_CR 寄存器中的 INIT 位置 1 来初始化散列处理器

  c) 在 STM32F42xxx 和 STM32F43xxx 上将存储器中保存的值写入 HASH_CSR0 到 HASH_CSR53 寄存器；在 STM32F405xx/07xx 和 STM32F415xx/17xx 上将存储器保存的值写入 HASH_CSR0 到 HASH_CSR50 寄存器

  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 寄存器中的 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： *1 如果上下文交换不涉及 HMAC 操作，则不必保存和恢复以下寄存器：HASH_CSR38 到 HASH_CSR50（在 STM32F405xx/07xx 和 STM32F415xx/17xx 上）以及 HASH_CSR38 到 HASH_CSR53（在 STM32F42xxx 和 STM32F43xxx 上）。*

2 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 IN FIFO，HASH_CR 寄存器中 NBW = 000），则不必保存和恢复 HASH_CSR22 到 HASH_CSR37 寄存器。

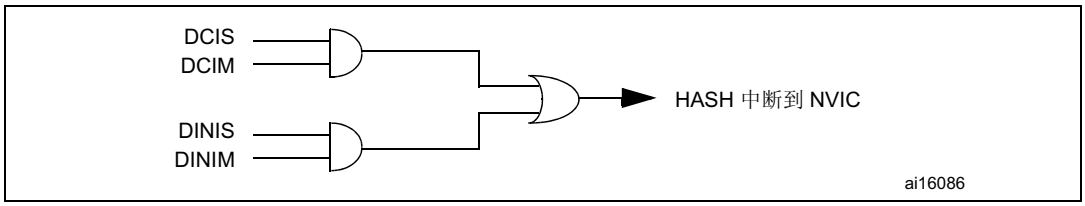
22.3.8 散列中断

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 221. 散列中断映射图



22.4 散列寄存器

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，否则将生成 AHB 错误。

22.4.1 用于 STM32F405xx/07xx 和 STM32F415xx/17xx 的散列控制寄存器 (HASH_CR)

HASH control register

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
Reserved															LKEY
															rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			DINNE	NBW				ALGO[0]	MODE	DATATYPE		DMAE	INIT	Reserved	
			r	r	r	r	r	rw	rw	rw	rw	rw	w		

位 31:17 保留，由硬件强制清零。

位 16 **LKEY**：长密钥选择 (Long key selection)

在 HMAC 模式中，该位在短密钥（≤ 64 字节）或长密钥（> 64 字节）之间进行选择

0：短密钥（≤ 64 字节）

1：长密钥（> 64 字节）

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 15:13 保留，由硬件强制清零。

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，会将此位清零。

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

位 11:8 **NBW**：已推入的字数 (Number of words already pushed)

该位字段反映了消息中已推入到 IN FIFO 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 位写入到 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 IN FIFO 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，IN FIFO 为空）

0001：2 个字已推入到 DIN 缓冲器（HASH_DIN 寄存器和 IN FIFO 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 IN FIFO 的字数。

位 7 **ALGO[1:0]**：算法选择 (Algorithm selection)

这些位用于选择 SHA-1 或 MD5 算法：

0：选择 SHA-1 算法

1：选择 MD5 算法

注意：仅当 INIT 位置 1 时此选择才有效。在计算期间更改该位不起作用。

位 6 **MODE**：模式选择 (Mode selection)

该位针对所选算法选择散列或 HMAC 模式：

0：选择散列模式

1：选择 HMAC 模式。如果正在使用的密钥的长度超过 64 字节，则必须 LKEY 置 1。

注意：仅当 INIT 位置 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**：数据类型选择 (Data type selection)

定义在 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，便发送 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。

2：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。

若在此位中写入 0 则不会产生影响。

读取该位将始终返回 0。

位 1:0 保留，必须保持清零。

22.4.2 用于 STM32F42xxx 和 STM32F43xxx 散列控制寄存器 (HASH_CR)

HASH control register

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
Reserved													ALGO[1]	Reserved	LKEY
													rw		rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		MDMAT	DINNE	NBW				ALGO[0]	MODE	DATATYPE		DMAE	INIT	Reserved	
		rw	r	r	r	r	r	rw	rw	rw	rw	rw	w		

位 31:19 保留，由硬件强制清零。

位 17 保留，由硬件强制清零。

位 16 **LKEY**：长密钥选择 (Long key selection)

在 HMAC 模式中，该位在短密钥（≤ 64 字节）或长密钥（> 64 字节）之间进行选择

0：短密钥（≤ 64 字节）

1：长密钥（> 64 字节）

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 15:14 保留，由硬件强制清零。

位 13 **MDMAT**：多个 DMA 传输 (Multiple DMA Transfers)

如果在需要多个 DMA 传输时对大型文件进行散列运算，则将此位置 1。

0：在 DMA 传输结束时自动将 DCAL 置 1。

1：在 DMA 传输结束时不自动将 DCAL 置 1。

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，会将此位清零。

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

位 11:8 **NBW**：已推入的字数 (Number of words already pushed)

该位字段反映了消息中已推入到 IN FIFO 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 位写入到 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 IN FIFO 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，IN FIFO 为空）

0001：2 个字已推入到 DIN 缓冲器（HASH_DIN 寄存器和 IN FIFO 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 IN FIFO 的字数。

位 18 和位 7 **ALGO[1:0]**：算法选择 (Algorithm selection)

这些位用于选择 SHA-1、SHA-224、SHA256 或 MD5 算法：

00：选择 SHA-1 算法

01：选择 MD5 算法

10：选择 SHA224 算法

11：选择 SHA256 算法

注意：仅当 INIT 位置 1 时此选择才有效。在计算期间更改该位不起作用。

位 6 **MODE**：模式选择 (Mode selection)

该位针对所选算法选择散列或 HMAC 模式：

0：选择散列模式。

1：选择 HMAC 模式。如果正在使用的密钥的长度超过 64 字节，则必须 LKEY 置 1。

注意：仅当 INIT 位置 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**：数据类型选择 (Data type selection)

定义在 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，便发送 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。

2：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。

若在此位中写入 0 则不会产生影响。

读取该位将始终返回 0。

位 1:0 保留，必须保持清零。

22.4.3 散列数据输入寄存器 (HASH_DIN)

HASH data input register

偏移地址：0x04

复位值：0x0000 0000

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															
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
DATAIN															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:0 **DATAIN**：数据输入 (Data input)

读取 = 返回寄存器的当前内容。
写入 = 当前寄存器内容推入到 IN FIFO，并且寄存器获取在 AHB 数据总线上呈现的新值。

22.4.4 散列启动寄存器 (HASH_STR)

HASH start register

偏移地址：0x08

复位值：0x0000 0000

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
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved							DCAL	Reserved			NBLW				
							w				rw	rw	rw	rw	rw

位 31:9 保留，由硬件强制清零。

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 IN FIFO 的所有数据字开始计算最终消息摘要。
读取该位将返回 0。
注意

位 7:5 保留，由硬件强制清零。

位 4:0 **NBLW**：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

如果写入这些位并且 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”时写入这些位，则位字段不会更改。
读取这些位将返回写入到 NBLW 的最后一个值。

注意：必须在设置 DCAL 位之前配置这些位，否则这些位不会有效。特别请注意，配置 NBLW 和设置 DCAL 不能同时进行。

22.4.5 散列摘要寄存器（HASH_HR0 到 4/5/6/7）

HASH digest registers

偏移地址：0x0C 到 0x1C（STM32F405xx/07xx 和 STM32F415xx/17xx）以及 0x310 到 0x32C（STM32F42xxx 和 STM32F43xxx）

复位值：0x0000 0000

这些寄存器包含消息摘要结果，其命名方式为：

1. 在 SHA1 算法说明中分别为 H0、H1、H2、H3 和 H4。
   请注意，这里不使用 HASH_H5 到 HASH_H7 并且读取值为零。
2. 在 MD5 算法说明中分别为 A、B、C 和 D。
   请注意，这里不使用 HASH_H4 到 HASH_H7 并且读取值为零。
3. 在 SHA224 算法说明中分别为 H0 到 H6。
   请注意，这里不使用 HASH_H7 并且读取值为零。
4. 在 SHA256 算法说明中分别为 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： *H0、H1、H2、H3 和 H4 映射到两个区域。*

HASH_HR0

偏移地址：0x0C 和 0x310

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
H0															
r	r	r	r	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
H0															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

HASH_HR1

偏移地址：0x10 和 0x314

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
H1															
r	r	r	r	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
H1															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

HASH_HR2

偏移地址：0x14 和 0x318

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
H2															
r	r	r	r	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
H2															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

HASH_HR3

偏移地址：0x18 和 0x31C

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
H3															
r	r	r	r	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
H3															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

HASH_HR4

偏移地址：0x1C 和 0x320

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
H4															
r	r	r	r	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
H4															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

HASH_HR5

偏移地址：0x324

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
H5															
r	r	r	r	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
H5															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

HASH_HR6

偏移地址：0x328

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
H6															
r	r	r	r	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
H6															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

HASH_HR7

偏移地址：0x32C

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
H7															
r	r	r	r	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
H7															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

注意： *当启动新位流的摘要计算时（通过将 INIT 位写入到 1），这些寄存器将呈现其复位值。*

22.4.6 散列中断使能寄存器 (HASH_IMR)

HASH interrupt enable register

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

位 31:2 保留，由硬件强制清零。

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

22.4.7 散列状态寄存器 (HASH_SR)

HASH status register

偏移地址：0x24

复位值：0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												BUSY	DMAS	DCIS	DINIS
												r	r	rc_w0	rc_w0

位 31:4 保留，由硬件强制清零。

位 3 **BUSY**：忙碌位 (Busy bit)

0：当前未在处理任何块
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，并且在 DMAE=0 且未在进行任何 DMA 传输时将被清零。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 寄存器中的 INIT 位写入 1。

0：HASH_Hx 寄存器中无任何摘要。
1：摘要计算已完成，摘要存储于 HASH_Hx 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。要将该位清零写入 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。

22.4.8 散列上下文交换寄存器 (HASH_CSRx)

HASH context swap registers

偏移地址：0x0F8 到 0x1C0

- 对于 HASH_CSR0 寄存器：复位值为 0x0000 0002。
- 对于其它寄存器：复位值是 0x0000 0000，但是 STM32F42xxx 和 STM32F43xxx 设备（其中的 HASH_CSR2 寄存器复位值是 0x2000 0000）除外。

在 STM32F42xxx 和 STM32F43xxx 上，其它寄存器可在 0x1C1 到 0x1CC 获得

- 复位值：0x0000 0000。

这些寄存器包含散列处理器的完整内部寄存器状态。当高优先级任务必须使用散列处理器，而散列处理器已由其它任务使用时，必须执行上下文交换，这时这些寄存器很有用。

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某个位置。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这些 HASH_CSRx 寄存器中。

HASH_CSRx

偏移地址：在 STM32F405xx/07xx 和 STM32F415xx/17xx 上是 0x0F8 到 0x1C0

偏移地址：在 STM32F42xxx 和 STM32F43xxx 上是 0x0F8 到 0x1CC

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

22.4.9 散列寄存器映射

表 95 汇总了散列寄存器映射和复位值。

表 95. STM32F405xx/07xx 和 STM32F415xx/17xx 上的散列寄存器映射和复位值

偏移	寄存器名称 复位值	寄存器大小																													
		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**HASH_CR**	Reserved															LKEY	Reserved			DINNE	NBW				ALGO[0]	MODE	DATATYPE		DMAE	INIT	Reserved	
	Reset value																0				0	0	0	0	0	0	0	0	0	0	0		
0x04	**HASH_DIN**	DATAIN																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08	**HASH_STR**	Reserved																							DCAL	Reserved				NBLW			
	Reset value																								0					0	0	0	0	0
0x0C	**HASH_HR0**	H0																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	**HASH_HR1**	H1																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14	**HASH_HR2**	H2																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	**HASH_HR3**	H3																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1C	**HASH_HR4**	H4																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20	**HASH_IMR**	Reserved																													DCIE	DINIE	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x24	**HASH_SR**	Reserved																												BUSY	DMAS	DCIS	DINIS
	Reset value																													0	0	0	1

表 95. STM32F405xx/07xx 和 STM32F415xx/17xx 上的散列寄存器映射和复位值 （续）

偏移	寄存器名称 复位值	寄存器大小 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
0xF8	**HASH_CSR0**	CSR0
	Reset value	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0
		... ...
0x1C0	**HASH_CSR50**	CSR50
	Reset value	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
	Reserved	
0x310	**HASH_HR0**	H0
	Reset value	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0x314	**HASH_HR1**	H1
	Reset value	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0x318	**HASH_HR2**	H2
	Reset value	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0x31C	**HASH_HR3**	H3
	Reset value	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0x320	**HASH_HR4**	H4
	Reset value	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

表 96. STM32F42xxx 和 STM32F43xxx 上的散列寄存器映射和复位值

偏移	寄存器名称 复位值	寄存器大小 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
0x00	**HASH_CR**	Reserved | ALGO[1] | Reserved | LKEY | Reserved | MDMAT | DINNE | NBW | ALGO[0] | MODE | DATATYPE | DMAE | INIT | Reserved
	Reset value	0 0 0 0 0 0
0x04	**HASH_DIN**	DATAIN
	Reset value	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0x08	**HASH_STR**	Reserved | DCAL | Reserved | NBLW
	Reset value	0 0 0 0 0 0
0x0C	**HASH_HR0**	H0
	Reset value	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0x10	**HASH_HR1**	H1
	Reset value	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0x14	**HASH_HR2**	H2
	Reset value	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0x18	**HASH_HR3**	H3
	Reset value	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0x1C	**HASH_HR4**	H4
	Reset value	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0x20	**HASH_IMR**	Reserved | DCIE | DINIE
	Reset value	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0x24	**HASH_SR**	Reserved | BUSY | DMAS | DCIS | DINIS
	Reset value	0 0 0 1

表 96. STM32F42xxx 和 STM32F43xxx 上的散列寄存器映射和复位值 （续）

偏移	寄存器名称 复位值	寄存器大小																														
		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xF8	**HASH_CSR0**	CSR0																														
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
...																																
0x1CC	**HASH_CSR53**	CSR53																														
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reserved																																
0x310	**HASH_HR0**	H0																														
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x314	**HASH_HR1**	H1																														
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x318	**HASH_HR2**	H2																														
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x31C	**HASH_HR3**	H3																														
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x320	**HASH_HR4**	H4																														
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x324	**HASH_HR5**	H5																														
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x328	**HASH_HR6**	H6																														
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x32C	**HASH_HR7**	H7																														
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

23 实时时钟 (RTC)

除非特别说明，否则本部分适用于整个 STM32F4xx 系列。

23.1 前言

实时时钟 (RTC) 是一个独立的 BCD 定时器/计数器。RTC 提供一个日历时钟、两个可编程闹钟中断，以及一个具有中断功能的周期性可编程唤醒标志。RTC 还包含用于管理低功耗模式的自动唤醒单元。

两个 32 位寄存器包含二进码十进数格式 (BCD) 的秒、分钟、小时（12 或 24 小时制）、星期几、日期、月份和年份。此外，还可提供二进制格式的亚秒值。

系统可以自动将月份的天数补偿为 28、29（闰年）、30 和 31 天。并且还可以进行夏令时补偿。

其它 32 位寄存器还包含可编程的闹钟亚秒、秒、分钟、小时、星期几和日期。

此外，还可以使用数字校准功能对晶振精度的偏差进行补偿。

上电复位后，所有 RTC 寄存器都会受到保护，以防止可能的非正常写访问。

无论器件状态如何（运行模式、低功耗模式或处于复位状态），只要电源电压保持在工作范围内，RTC 便不会停止工作。

23.2 RTC 主要特性

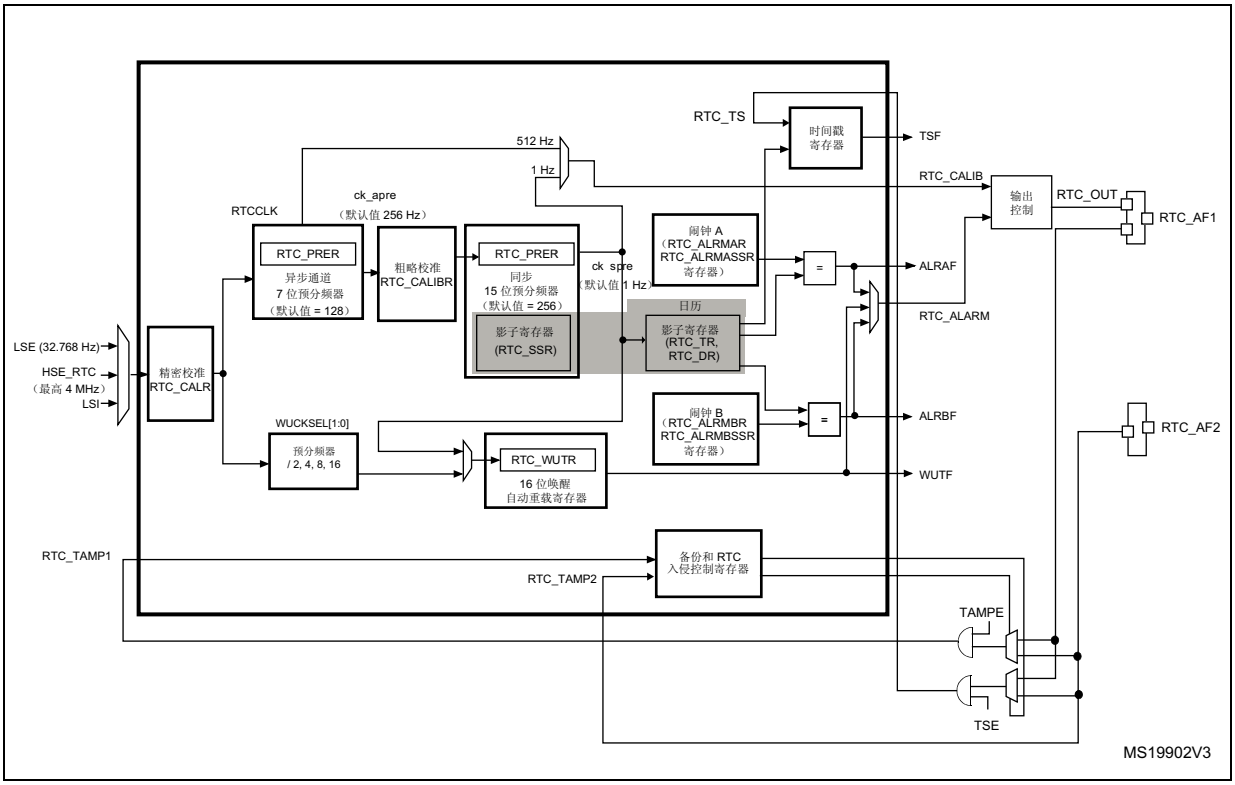
RTC 单元的主要特性如下（参见*图 222：RTC 框图*）：

- 包含亚秒、秒、分钟、小时（12/24 小时制）、星期几、日期、月份和年份的日历。
- 软件可编程的夏令时补偿。
- 两个具有中断功能的可编程闹钟。可通过任意日历字段的组合驱动闹钟。
- 自动唤醒单元，可周期性地生成标志以触发自动唤醒中断。
- 参考时钟检测：可使用更加精确的第二时钟源（50 Hz 或 60 Hz）来提高日历的精确度。
- 利用亚秒级移位特性与外部时钟实现精确同步。
- 可屏蔽中断/事件：
  - 闹钟 A
  - 闹钟 B
  - 唤醒中断
  - 时间戳
  - 入侵检测
- 数字校准电路（周期性计数器调整）
  - 精度为 5 ppm
  - 精度为 0.95 ppm，在数秒钟的校准窗口中获得
- 用于事件保存的时间戳功能（1 个事件）
- 入侵检测：
  - 2 个带可配置过滤器和内部上拉的入侵事件
- 20 个备份寄存器（80 字节）。发生入侵检测事件时，将复位备份寄存器。

- 复用功能输出 (RTC_OUT)，可选择以下两个输出之一：
  - RTC_CALIB：512 Hz 或 1 Hz 时钟输出（LSE 频率为 32.768 kHz)。
    可通过将 RTC_CR 寄存器中的 COE[23] 位置 1 来使能此输出。该输出可连接到器件 RTC_AF1 功能。
  - RTC_ALARM（闹钟 A、闹钟 B 或唤醒）。
    可通过配置 RTC_CR 寄存器的 OSEL[1:0] 位选择此输出。该输出可连接到器件 RTC_AF1 功能。
- RTC 复用功能输入：
  - RTC_TS：时间戳事件检测。该输入可连接到器件 RTC_AF1 和 RTC_AF2 功能。
  - RTC_TAMP1：TAMPER1 事件检测。该输入可连接到器件 RTC_AF1 和 RTC_AF2 功能。
  - RTC_TAMP2：TAMPER2 事件检测。
  - RTC_REFIN：参考时钟输入（通常为市电，50 Hz 或 60 Hz）。

请参见第 7.3.15 节：选择 RTC_AF1 和 RTC_AF2 复用功能。

图 222. RTC 框图



1. 对于 STM32F4xx 器件，RTC_AF1 和 RTC_AF2 复用功能分别与 PC13 和 PI8 相连。

23.3 RTC 功能说明

23.3.1 时钟和预分频器

RTC 时钟源 (RTCCLK) 通过时钟控制器从 LSE 时钟、LSI 振荡器时钟以及 HSE 时钟三者中选择。有关 RTC 时钟源配置的更多信息，请参见*第 6 节：复位和时钟控制 (RCC)*。

可编程的预分频器阶段可生成 1 Hz 的时钟，用于更新日历。为最大程度地降低功耗，预分频器分为 2 个可编程的预分频器（参见*图 222：RTC 框图*）：

- 一个通过 RTC_PRER 寄存器的 PREDIV_A 位配置的 7 位异步预分频器。
- 一个通过 RTC_PRER 寄存器的 PREDIV_S 位配置的 15 位同步预分频器。

注意： *使用两个预分频器时，推荐将异步预分频器配置为较高的值，以最大程度降低功耗。*

要使用频率为 32.768 kHz 的 LSE 获得频率为 1 Hz 的内部时钟 (ck_spre)，需要将异步预分频系数设置为 128，并将同步预分频系数设置为 256。

分频系数的最小值为 1，最大值为 2^{22}。

这对应于约为 4 MHz 的最大输入频率。

f_{ck_apre} 可根据以下公式得出：

$$f_{CK\_APRE} = \frac{f_{RTCCLK}}{PREDIV\_A + 1}$$

ck_apre 时钟用于为二进制 RTC_SSR 亚秒递减计数器提供时钟。当该计数器计数到 0 时，会使用 PREDIV_S 的内容重载 RTC_SSR。

f_{ck_spre} 可根据以下公式得出：

$$f_{CK\_SPRE} = \frac{f_{RTCCLK}}{(PREDIV\_S + 1) \times (PREDIV\_A + 1)}$$

ck_spre 时钟既可以用于更新日历，也可以用作 16 位唤醒自动重载定时器的时基。为获得较短的超时周期，还可以将 16 位唤醒自动重载定时器与经可编程的 4 位异步预分频器分频的 RTCCLK 一同运行（有关详细信息，请参见*第 23.3.4 节：周期性自动唤醒*）。

23.3.2 实时时钟和日历

RTC 日历时间和日期寄存器可通过与 PCLK1（APB1 时钟）同步的影子寄存器来访问。这些时间和日期寄存器也可以直接访问，这样可避免等待同步的持续时间。

- RTC_SSR 对应于亚秒
- RTC_TR 对应于时间
- RTC_DR 对应于日期

每隔两个 RTCCLK 周期，便将当前日历值复制到影子寄存器，并将 RTC_ISR 寄存器的 RSF 位置 1（请参见*第 23.6.4 节*）。在停机和待机模式下不会执行复制操作。退出这两种模式时，影子寄存器会在最长 2 个 RTCCLK 周期后进行更新。

当应用读取日历寄存器时，它会访问影子寄存器的内容。也可以通过将 RTC_CR 寄存器的 BYPSHAD 控制位置 1 来直接访问日历寄存器。默认情况下，该位被清零，用户访问影子寄存器。

在 BYPSHAD=0 模式下读取 RTC_SSR、RTC_TR 或 RTC_DR 寄存器时，APB 时钟频率 (f_{APB}) 必须至少为 RTC 时钟频率 (f_{RTCCLK}) 的 7 倍。

影子寄存器通过系统复位来复位。

23.3.3 可编程闹钟

RTC 单元提供两个可编程闹钟，即闹钟 A 和闹钟 B。

可通过将 RTC_CR 寄存器中的 ALRAE 和 ALRBE 位置 1 来使能可编程闹钟功能。如果日历亚秒、秒、分钟、小时、日期或日分别与闹钟寄存器 RTC_ALRMASSR/RTC_ALRMAR 和 RTC_ALRMBSSR/RTC_ALRMBR 中编程的值相匹配，则 ALRAF 和 ALRBF 标志会被置为 1。可通过 RTC_ALRMAR 和 RTC_ALRMBR 寄存器的 MSKx 位以及 RTC_ALRMASSR 和 RTC_ALRMBSSR 寄存器的 MASKSSx 位单独选择各日历字段。可通过 RTC_CR 寄存器中的 ALRAIE 和 ALRBIE 位使能闹钟中断。

闹钟 A 和闹钟 B（如果已通过 RTC_CR 寄存器中的位 OSEL[0:1] 使能）可连接到 RTC_ALARM 输出。可通过 RTC_CR 寄存器的 POL 位配置 RTC_ALARM 极性。

注意： 如果选择秒字段（RTC_ALRMAR 或 RTC_ALRMBR 中的 MSK0 位复位），则 RTC_PRER 寄存器中设置的同步预分频器分频系数必须至少为 3，才能确保闹钟正确地运行。

23.3.4 周期性自动唤醒

周期性唤醒标志由 16 位可编程自动重载递减计数器生成。唤醒定时器范围可扩展至 17 位。

可通过 RTC_CR 寄存器中的 WUTE 位来使能此唤醒功能。

唤醒定时器的时钟输入可以是：

- 2、4、8 或 16 分频的 RTC 时钟 (RTCCLK)。

  当 RTCCLK 为 LSE (32.768 kHz) 时，可配置的唤醒中断周期介于 122 µs 和 32 s 之间，且分辨率低至 61 µs。
- ck_spre（通常为 1 Hz 内部时钟）

  当 ck_spre 频率为 1 Hz 时，可得到的唤醒时间为 1s 到 36h 左右，分辨率为 1 秒。这一较大的可编程时间范围分为两部分：
  - WUCKSEL [2:1] = 10 时为 1s 到 18h。
  - WUCKSEL [2:1] = 11 时约为 18h 到 36h。在后一种情况下，会将 2^{16} 添加到 16 位计数器当前值。完成初始化序列后（请参见 *第 577 页的编程唤醒定时器*），定时器开始递减计数。在低功耗模式下使能唤醒功能时，递减计数保持有效。此外，当计数器计数到 0 时，RTC_ISR 寄存器的 WUTF 标志会置 1，并且唤醒寄存器会使用其重载值（RTC_WUTR 寄存器值）动重载。

之后必须用软件清零 WUTF 标志。

通过将 RTC_CR2 寄存器中的 WUTIE 位置 1 来使能周期性唤醒中断时，它会使器件退出低功耗模式。

如果已通过 RTC_CR 寄存器中的位 OSEL[0:1] 使能周期性唤醒标志，则该标志可连接到 RTC_ALARM 输出。可通过 RTC_CR 寄存器的 POL 位配置 RTC_ALARM 极性。

系统复位以及低功耗模式（睡眠、停机和待机）对唤醒定时器没有任何影响。

23.3.5 RTC 初始化和配置

RTC 寄存器访问

RTC 寄存器为 32 位寄存器。除了当 BYPSHAD=0 时对日历影子寄存器执行的读访问之外，APB 接口会在访问 RTC 寄存器时引入 2 个等待周期。

RTC 寄存器写保护

系统复位后，可通过 PWR 电源控制寄存器 (PWR_CR) 的 DBP 位保护 RTC 寄存器以防止非正常的写访问。必须将 DBP 位置 1 才能使能 RTC 寄存器的写访问。

上电复位后，所有 RTC 寄存器均受到写保护。通过向写保护寄存器 (RTC_WPR) 写入一个密钥来使能对 RTC 寄存器的写操作。

要解锁所有 RTC 寄存器（RTC_ISR[13:8]、RTC_TAFCR 和 RTC_BKPxR 除外）的写保护，需要执行以下步骤：

1. 将“0xCA”写入 RTC_WPR 寄存器。
2. 将“0x53”写入 RTC_WPR 寄存器。

写入一个错误的关键字会再次激活写保护。

保护机制不受系统复位影响。

日历初始化和配置

要编程包括时间格式和预分频器配置在内的初始时间和日期日历值，需按照以下顺序操作：

1. 将 RTC_ISR 寄存器中的 INIT 位置 1 以进入初始化模式。在此模式下，日历计数器将停止工作并且其值可更新。
2. 轮询 RTC_ISR 寄存器中的 INITF 位。当 INITF 置 1 时进入初始化阶段模式。大约需要 2 个 RTCCLK 时钟周期（由于时钟同步）。
3. 要为日历计数器生成 1 Hz 时钟，应首先编程 RTC_PRER 寄存器中的同步预分频系数，然后编程异步预分频系数。即使只需要更改这两个字段中之一，也必须对 RTC_PRER 寄存器执行两次单独的写访问。
4. 在影子寄存器（RTC_TR 和 RTC_DR）中加载初始时间和日期值，然后通过 RTC_CR 寄存器中的 FMT 位配置时间格式（12 或 24 小时制）。
5. 通过清零 INIT 位退出初始化模式。随后，自动加载实际日历计数器值，在 4 个 RTCCLK 时钟周期后重新开始计数。

当初始化序列完成之后，日历开始计数。

注意： *系统复位后，应用可读取 RTC_ISR 寄存器中的 INITS 标志，以检查日历是否已初始化。如果该标志为 0，表明日历尚未初始化，因为年份字段设置为其上电复位时的默认值 (0x00)。*

要在初始化之后读取日历，必须首先用软件检查 RTC_ISR 寄存器的 RSF 标志是否置 1。

夏令时

可通过 RTC_CR 寄存器的 SUB1H、ADD1H 和 BKP 位管理夏令时。

利用 SUB1H 或 ADD1H，软件只需单次操作便可在日历中减去或增加一个小时，无需执行整个初始化步骤。

此外，软件还可以使用 BKP 位来记录是否曾经执行过此操作。

编程闹钟

要对可编程的闹钟（闹钟 A 或闹钟 B）进行编程或更新，必须执行类似的步骤：

1. 将 RTC_CR 寄存器中的 ALRAE 或 ALRBE 位清零以禁止闹钟 A 或闹钟 B。
2. 轮询 RTC_ISR 寄存器中的 ALRAWF 或 ALRBWF 位，直到其中一个置 1，以确保闹钟寄存器可以访问。大约需要 2 个 RTCCLK 时钟周期（由于时钟同步）。
3. 编程闹钟 A 或闹钟 B 寄存器（RTC_ALRMASSR/RTC_ALRMAR 或 RTC_ALRMBSSR/RTC_ALRMBR）。
4. 将 RTC_CR 寄存器中的 ALRAE 或 ALRBE 位置 1 以再次使能闹钟 A 或闹钟 B。

注意： *约 2 个 RTCCLK 时钟周期（由于时钟同步）后，将执行对 RTC_CR 寄存器的更改。*

编程唤醒定时器

要配置或更改唤醒定时器的自动重载值（RTC_WUTR 中的 WUT[15:0]），需要按照以下顺序操作：

1. 清零 RTC_CR 中的 WUTE 以禁止唤醒定时器。
2. 轮询 RTC_ISR 中的 WUTWF，直到该位置 1，以确保可以访问唤醒自动重载定时器和 WUCKSEL[2:0] 位。大约需要 2 个 RTCCLK 时钟周期（由于时钟同步）。
3. 编程唤醒自动重载值 WUT[15:0]，并选择唤醒时钟（RTC_CR 中的 WUCKSEL[2:0] 位）。将 RTC_CR 寄存器中的 WUTE 位置 1 以再次使能定时器。唤醒定时器重新开始递减计数。

23.3.6 读取日历

当 RTC_CR 寄存器中的 BYPSHAD 控制位清零时

要正确读取 RTC 日历寄存器（RTC_SSR、RTC_TR 和 RTC_DR），APB1 时钟频率 (f_{PCLK1}) 必须等于或大于 f_{RTCCLK} RTC 时钟频率的七倍。这可以确保同步机制行为的安全性。

如果 APB1 时钟频率低于 RTC 时钟频率的七倍，则软件必须分两次读取日历时间寄存器和日期寄存器。这样，当两次读取的 RTC_TR 结果相同时，才能确保数据正确。否则必须执行第三次读访问。任何情况下，APB1 的时钟频率都不能低于 RTC 的时钟频率。

每次将日历寄存器中的值复制到 RTC_SSR、RTC_TR 和 RTC_DR 影子寄存器时，RTC_ISR 寄存器中的 RSF 位都会置 1。每两个 TRCCLK 周期执行一次复制。为确保这 3 个值来自同一时刻点，读取 RTC_SSR 或 RTC_TR 时会锁定高阶日历影子寄存器中的值，直到读取 RTC_DR。为避免软件对日历执行读访问的时间间隔小于 2 个 RTCCLK 周期：第一次读取日历之后必须通过软件将 RSF 清零，并且软件必须等待到 RSF 置 1 之后才可再次读取 RTC_SSR、RTC_TR 和 RTC_DR 寄存器。

从低功耗模式（停机模式或待机模式）唤醒之后，必须通过软件将 RSF 清零。之后，软件必须等待至 RSF 再次置 1 之后才可以读取 RTC_SSR、RTC_TR 和 RTC_DR 寄存器。

RSF 位必须在唤醒之后而不是进入低功耗模式之前进行清零。

注意： *系统复位之后，软件必须等待至 RSF 置 1 之后才可以读取 RTC_SSR、RTC_TR 和 RTC_DR 寄存器。实际上，系统复位会将影子寄存器复位为其默认值。*

初始化之后（请参见第 576 页的日历初始化和配置）：软件必须等待至 RSF 置 1 之后才可以读取 RTC_SSR、RTC_TR 和 RTC_DR 寄存器。

同步之后（请参见第 23.3.8 节：RTC 同步）：软件必须等待至 RSF 置 1 之后才可以读取 RTC_SSR、RTC_TR 和 RTC_DR 寄存器。

当 RTC_CR 寄存器中的 BYPSHAD 控制位置 1 时（旁路影子寄存器）

读取日历寄存器时会直接从日历计数器获取值，这样便无需等待至 RSF 位置 1。这对于从低功耗模式（停机模式或待机模式）退出后的情况特别有用，因为影子寄存器在这些模式下不更新。

当 BYPSHAD 位置 1 时，如果在对寄存器的两次读访问之间出现 RTCCLK 沿，则不同寄存器的结果彼此可能不一致。此外，如果在读操作期间出现 RTCCLK 沿，则可能导致其中一个寄存器的值不正确。软件必须分两次读取所有寄存器，然后将两次结果加以比较来确认数据是否一致和正确。此外，软件也可以只比较两次读取日历寄存器得到的结果的最低位。

注意： *当 BYPSHAD=1 时，读取日历寄存器的指令需要一个额外的 APB 周期才能完成。*

23.3.7 复位 RTC

日历影子寄存器（RTC_SSR、RTC_TR 和 RTC_DR）以及 RTC 状态寄存器 (RTC_ISR) 的某些位通过所有可用的系统复位源复位为各自的默认值。

相反，以下寄存器则通过上电复位来复位为各自的默认值并且不受系统复位的影响：RTC 当前日历寄存器、RTC 控制寄存器 (RTC_CR)、预分频器寄存器 (RTC_PRER)、RTC 校准寄存器（RTC_CALIBR 或 RTC_CALR）、RTC 移位寄存器 (RTC_SHIFTR)、RTC 时间戳寄存器（RTC_TSSSR、RTC_TSTR 和 RTC_TSDR）、RTC 入侵和复用功能配置寄存器 (RTC_TAFCR)、RTC 备份寄存器 (RTC_BKPxR)、唤醒定时器寄存器 (RTC_WUTR) 以及闹钟 A 和闹钟 B 寄存器（RTC_ALRMASSR/RTC_ALRMAR 和 RTC_ALRMBSSR/RTC_ALRMBR）。

此外，如果复位源不是上电复位源，则发生系统复位时，RTC 会继续工作。发生上电复位时，RTC 会停止工作，并且所有 RTC 寄存器都会设置为各自的复位值。

23.3.8 RTC 同步

RTC 可与高精度的远程时钟同步。在读取亚秒字段后（RTC_SSR 或 RTC_TSSSR），即可计算远程时钟的时间与 RTC 之间的精准偏差。之后，可使用 RTC_SHIFTR 对 RTC 的时钟进行零点几秒的“平移”，经过调整后可消除此偏差。

RTC_SSR 包含同步预分频器计数器的值。这样，便可计算分辨率低至 1/(PREDIV_S + 1) 秒的 RTC 的准确时间。因此，可通过增大同步预分频器的值 (PREDIV_S[14:0]) 来提高分辨率。将 PREDIV_S 设置为 0x7FFF 时，可得到允许的最大分辨率（30.52 µs，时钟频率为 32768 Hz）。

但是，提高 PREDIV_S 意味着必须降低 PREDIV_A 才能将同步预分频器的输出维持在 1 Hz。这样，异步预分频器的输出频率会增大，RTC 的动态功耗也会相应增加。

可以使用 RTC 平移控制寄存器 (RTC_SHIFTR) 对 RTC 进行微调。可以用大小为 1/(PREDIV_S + 1) 秒的分辨率对 RTC_SHIFTR 进行写操作，将时钟平移（延迟或提前）最长 1 秒。在这种平移操作中，会将 SUBFS[14:0] 值加到同步预分频器计数器 SS[15:0] 中：这将使时钟产生延迟。如果同时将 ADD1S 位置 1，则会增加一秒，与此同时减去的时间为零点几秒，因此将使时钟提前。

注意： 初始化平移操作前，用户必须检查确认 SS[15] = 0，以确保不会发生上溢。

对 RTC_SHIFTR 寄存器执行写操作以启动平移操作时，硬件会将 SHPF 标志置 1 以指示平移操作挂起。完成平移操作时，硬件会将将该位清零。

注意： 该同步功能与参考时钟检测功能不兼容：当 REFCKON=1 时，固件不能对 RTC_SHIFTR 执行写操作。

23.3.9 RTC 参考时钟检测

RTC 日历更新可与参考时钟 RTC_REFIN（通常为市电，50 Hz 或 60 Hz）同步。RTC_REFIN 参考时钟的精度应高于 32.768 kHz LSE 时钟。使能 RTC_REFIN 检测时（将 RTC_CR 的 REFCKON 位置 1），日历仍由 LSE 提供时钟，而 RTC_REFIN 用于补偿不准确的日历更新频率 (1 Hz)。

每个 1 Hz 时钟边沿都与最近的参考时钟边沿进行比较（如果在给定的时间窗口内发现一个边沿）。在大多数情况下，两个时钟边沿恰好对齐。当 1 Hz 时钟由于 LSE 时钟不精确而发生偏离时，RTC 会稍微偏移 1 Hz 时钟，以便后续的 1 Hz 时钟边沿能够对齐。利用这种机制，可使日历像参考时钟一样精确。

RTC 使用 32.768 kHz 石英产生的 256 Hz 时钟 (ck_apre) 检测是否存在参考时钟源。大约在日历每次更新时（每 1 秒钟），便会在时间窗口期间执行一次检测。检测到第一个参考时钟边沿时，该窗口等于 7 个 ck_apre 周期。随后的日历更新使用长度为 3 个 ck_apre 周期的较小窗口。

每次在窗口中检测到参考时钟时，都会行强制输出 ck_apre 时钟的异步预分频器进行重载。当参考时钟与 1 Hz 时钟对齐时，此操作不起作用，因为预分频器会在同一时刻重载。当时钟不对齐时，重载操作会微调后续的 1 Hz 时钟边沿，使其与参考时钟对齐。

如果参考时钟停止（在 3 个 ck_apre 窗口内未出现参考时钟边沿），日历将仅根据 LSE 时钟进行连续更新。RTC 随后使用 ck_spre 边沿上居中的大检测窗口（7 个 ck_apre 周期）等待参考时钟。

使能参考时钟检测后，必须将 PREDIV_A 和 PREDIV_S 设置为各自的默认值：

- PREDIV_A = 0x007F
- PREVID_S = 0x00FF

注意： *参考时钟检测在待机模式下不可用。*

注意： 参考时钟检测功能不能与粗略数字校准同时使用：当 REFCKON=1 时，必须使 RTC_CALIBR 保持为 0x0000 0000。

23.3.10 RTC 粗略数字校准

可以使用两种数字校准方法：粗略校准和精密校准。要执行粗略校准，请参见*第 23.6.7 节：RTC 校准寄存器 (RTC_CALIBR)*。

这两种校准方法不能一起使用，应用必须从中选择一种。粗略校准出于兼容性原因而提供。要执行精密校准，请参见*第 23.3.11 节：RTC 精密数字校准*和*第 23.6.16 节：RTC 校准寄存器 (RTC_CALR)*。

粗略数字校准功能可通过在异步预分频器 (ck_apre) 的输出端增加（正校准）或减少（负校准）时钟周期来实现晶振误差补偿。

将 RTC_CALIBR 寄存器中的 DCS 位置“0”和“1”可分别选择正校准和负校准。

当使能正校准时 （DCS = “0”），在 2xDC 分钟时间内，每分钟（约 15360 个 ck_apre 周期）增加 2 个 ck_apre 周期。这会提前更新日历，因此可将 RTC 的有效频率调高一些。

当使能负校准时（DCS = “1”），在 2xDC 分钟时间内，每分钟（约 15360 个 ck_apre 周期）减少 1 个 ck_apre 周期。这将推迟更新日历，因此可将 RTC 的有效频率调低一些。

可通过 RTC_CALIBR 寄存器的位 DC[4:0] 配置 DC。其数字范围为 0 到 31，对应的时间间隔 (2xDC) 范围为 0 到 62。

粗略数字校准只能在初始化模式下进行配置，并在 INIT 位清零后启动。整个校准周期将持续 64 分钟。可采用上述方法修改这 64 分钟周期的前 2xDC 分钟。

负校准的分辨率约为 2 ppm，而正校准的分辨率约为 4 ppm。最大校准范围为 –63 ppm 到 126 ppm。

当使用 LSE 或 HSE 作为 RTC 时钟时，可以使用这种校准方法。

注意： 如果 PREDIV_A < 6，数字校准可能无法正常工作。

例如当 RTCCLK=32.768 kHz 且 PREDIV_A+1=128 时

以下描述假定 ck_apre 频率为 256 Hz，由标称频率为 32.768 kHz 的 LSE 时钟提供，并且 PREDIV_A 设置为 127（默认值）。

ck_spre 时钟频率只能在 64 分钟周期的前 2xDC 分钟内被更改。例如，当 DC 等于 1 时，只有前两分钟被更改。这意味着，假定每个 ck_apre 周期表示 128 个 RTCCLK 周期（通过 PREDIV_A+1=128 计算得出），则对于每个 64 分钟周期的前 2xDC 分钟来说，每分钟都会有一秒减少 256 个 RTCCLK 周期或增加 128 个 RTCCLK 周期。

因此，对于每 125829120 个 RTCCLK 周期（64min x 60 s/min x 32768 周期/s），每个校准步骤都会增加 512 个振荡器周期或减少 256 个振荡器周期。这相当于每个校准步骤的分辨率为 +4.069 ppm 或 -2.035 ppm。因此，每个月的校准分辨率为 +10.5 或 –5.27 秒，每个月的总校准范围为 +5.45 到 –2.72 分钟。

为测量时钟偏差，需要输出一个 512 Hz 的时钟用于校准。请参见*第 23.3.14 节：校准时钟输出*。

23.3.11 RTC 精密数字校准

RTC 频率可采用约 0.954 ppm 的分辨率进行数字校准，校准范围为 -487.1 ppm 到 +488.5 ppm。使用一系列微调（增加和/或减少单独的 RTCCLK 脉冲）进行频率校正。这些微调的分布非常均匀，因此 RTC 的校准效果相当好，即使在短时间内持续观察也是如此。

当输入频率为 32768 Hz 时，精密数字校准的周期约为 2^{20} 个 RTCCLK 脉冲或 32 秒。精密数字校准寄存器 (RTC_CALR) 可指定 32 秒周期内要减少的 RTCCLK 时钟周期数：

- 将位 CALM[0] 置 1 时，32 秒周期内将只减少 1 个脉冲。
- 将 CALM[1] 置 1 时，将减少两个周期。
- 将 CALM[2] 置 1 时，将减少 4 个周期。
- 依此类推，将 CALM[8] 置 1 时，将减少 256 个时钟。

使用适当分辨率时，CALM 可使 RTC 频率减少最多 487.1 ppm，而 CALP 可用于使频率增加 488.5 ppm。将 CALP 置“1”，可每隔 2^{11} 个 RTCCLK 周期有效插入一个额外的 RTCCLK 脉冲，这意味着每 32 秒周期可增加 512 个时钟。

与 CALM 和 CALP 配合使用时，可在 32 秒周期内增加一个范围为 -511 到 +512 RTCCLK 周期的偏差，对应的校准范围为 -487.1 ppm 到 +488.5 ppm，分辨率约为 0.954 ppm。

若输入频率 (F_{RTCCLK}) 已知，可通过以下公式计算有效校准频率 (F_{CAL})：

$$F_{CAL} = F_{RTCCLK} \times [\, 1 + (CALP \times 512 - CALM) / (2^{20} + CALM - CALP \times 512) \,]$$

PREDIV_A<3 条件下的校准

当异步预分频器值（RTC_PRER 寄存器中的 PREDIV_A 位）小于 3 时，不能将 CALP 位置 1。如果 CALP 已置 1 并且 PREDIV_A 位的值小于 3，则会忽略 CALP，即假定 CALP 等于 0 而执行校准。

要在 PREDIV_A 小于 3 的条件下执行校准，应降低同步预分频器值 (PREDIV_S) 以便每秒内可加速 8 个 RTCCLK 时钟周期，这意味着每 32 秒可增加 256 个时钟周期。因此，仅使用 CALM 位，可在每 32 秒内有效增加 255 到 256 个时钟脉冲（对应的校准范围为 243.3 ppm 到 244.1 ppm）。

在标称 RTCCLK 频率 32768 Hz 下，当 PREDIV_A 等于 1 时（分频系数为 2），应将 PREDIV_S 设置为 16379 而不是 16383（少 4）。唯一相关的其它情况是，当 PREDIV_A 等于 0 时，应将 PREDIV_S 设置为 32759 而不是 32767（少 8）。

如果以这种方式减少 PREDIV_S，则采用以下公式计算校准输入时钟的有效频率：

$$F_{CAL} = F_{RTCCLK} \times [\ 1 + (256 - CALM) / (2^{20} + CALM - 256)\ ]$$

在这种情况下，如果 RTCCLK 恰好为 32768.00 Hz，则当 CALM[7:0] 等于 0x100 时（CALM 范围的中值），说明设置正确。

验证 RTC 校准

通过测量 TRCCLK 的精确频率，计算正确的 CALM 和 CALP 值以实现 RTC 精度。此外，还为应用提供了一个可选的 1 Hz 输出，用来测量和验证 RTC 精度。

如果在有限的间隔内测量 RTC 的精确频率，则会导致测量期间产生最多 2 个 RTCCLK 时钟周期的测量误差，具体取决于数字校准周期与测量周期的对齐方式。

但是，如果测量周期与校准周期的长度相同，则可以消除此测量误差。在这种情况下，观测到的唯一误差是由数字校准的分辨率导致的误差。

- 默认情况下，校准周期为 32 秒。

  在此模式下，测量整个 32 秒内 1 Hz 输出的精度，可确保测量误差在 0.477 ppm 内（32 秒内为 0.5 个 RTCCLK 周期，受校准分辨率限制）。
- 可将 RTC_CALR 寄存器的 CALW16 位置 1，以强制 16 秒的校准周期。

  此时，可在 16 秒内测量 RTC 精度，产生的最大误差为 0.954 ppm（16 秒内为 0.5 个 RTCCLK 周期）。但是，由于校准分辨率降低，长期的 RTC 精度也会降到 0.954 ppm：将 CALW16 置 1 时，CALM[0] 位将始终保持为 0。
- 可将 RTC_CALR 寄存器的 CALW8 位置 1，以强制 8 秒的校准周期。

  此时，可在 8 秒内测量 RTC 精度，产生的最大误差为 1.907 ppm（8 秒内为 0.5 个 RTCCLK 周期）。长期的 RTC 精度也会降到 1.907 ppm：将 CALW8 置 1 时，CALM[1:0] 位将始终保持为 00。

动态重校准

当 RTC_ISR/INITF=0 时，可动态更新校准寄存器 (RTC_CALR)，具体步骤如下：

1. 轮询 RTC_ISR/RECALPF（重新校准挂起标志）。
2. 如果该标志为 0，则可以根据需要向 RTC_CALR 写入新值。随后 RECALPF 位会被自动置为 1。
3. 新校准设置将在对 RTC_CALR 执行写操作之后的三个 ck_apre 周期内生效。

23.3.12 时间戳功能

将 RTC_CR 寄存器的 TSE 位置 1 可使能时间戳。

当在 TIMESTAMP 备用功能映射到的引脚上检测到时间戳事件时，日历会保存到时间戳寄存器（RTC_TSSSR、RTC_TSTR 和 RTC_TSDR）中。发生时间戳事件时，RTC_ISR 寄存器中的时间戳标志位 (TSF) 将置 1。

通过将 RTC_CR 寄存器中的 TSIE 位置 1，可在发生时间戳事件时生成中断。

如果在时间戳标志 (TSF) 已置 1 的条件下检测到新的时间戳事件，则时间戳上溢标志 (TSOVF) 将置 1，而时间戳寄存器（RTC_TSTR 和 RTC_TSDR）将保持上一事件的结果。

注意： *由于同步过程，TSF 将在时间戳事件后 2 个 ck_apre 周期置 1。*

将 TSOVF 置 1 时不存在延迟。这意味着，如果两个时间戳事件接连发生，则 TSOVF 可能为“1”而 TSF 仍为“0”。因此，建议只在检测到 TSF 为“1”后再轮询 TSOVF。

注意： 如果在 TSF 位清零后紧接着发生时间戳事件，则 TSF 和 TSOVF 位都将置 1。为防止在时间戳事件发生的同时屏蔽该事件，除非已将 TSF 位读取为“1”，否则应用程序不得将“0”写入 TSF 位。

此外，入侵事件可能导致时间戳被记录。有关 TAMPTS 控制位的说明，请参见*第 23.6.17 节：RTC 入侵和复用功能配置寄存器 (RTC_TAFCR)*。如果时间戳事件与配置为过滤模式的入侵事件（TAMPFLT 设置为非零值）使用同一引脚，则必须通过将 RTC_TAFCR 寄存器中的 TAMPTS 置“1”来选择入侵检测事件的时间戳模式。

时间戳复用功能

时间戳复用功能 (RTC_TS) 可映射到 RTC_AF1 或 RTC_AF2，具体取决于 RTC_TAFCR 寄存器中 TSINSEL 位的值（请参见*第 23.6.17 节：RTC 入侵和复用功能配置寄存器 (RTC_TAFCR)*）。如果 RTC_AF1 用作过滤模式的 TAMPER（TAMPFLT 设置为非零值），则不允许将时间戳事件映射到 RTC_AF2 上。

23.3.13 入侵检测

有两个入侵检测输入可用。这两个输入既可配置为边沿检测，也可配置为带过滤的电平检测。

RTC 备份寄存器

备份寄存器 (RTC_BKPxR) 包括20 个 32 位寄存器，用于存储 80 字节的用户应用数据。这些寄存器在备份域中实现，可在 V_{DD} 电源关闭时通过 V_{BAT} 保持上电状态。备份寄存器不会在系统复位或电源复位时复位，也不会在器件从待机模式唤醒时复位。

发生入侵检测事件时，将复位备份寄存器。（请参见*第 23.6.20 节：RTC 备份寄存器 (RTC_BKPxR)* 和*第 582 页的入侵检测初始化*）

入侵检测初始化

两个入侵检测输入分别与 RTC_ISR2 寄存器中的标志 TAMP1F/TAMP2F 相关联。可通过将 RTC_TAFCR 寄存器中相应的 TAMP1E/TAMP2E 位置 1 来使能各输入。

入侵检测事件会复位所有备份寄存器 (RTC_BKPxR)。

通过将 RTC_TAFCR 寄存器中的 TAMPIE 位置 1，可在发生入侵检测事件时生成中断。

入侵事件的时间戳

当 TAMPTS 置“1”时，任何入侵事件都会导致时间戳事件的发生。在这种情况下，如同发生正常时间戳事件一样，RTC_ISR 中的 TSF 位或 TSOVF 位会置 1。在 TSF 或 TSOVF 置 1 的同时，受影响的入侵标志寄存器（TAMP1F、TAMP2F）也会随之置 1。

对入侵输入的边沿检测

如果 TAMPFLT 位为“00”，则在观测到上升沿或下降沿时（根据相应的 TAMPxTRG 位），TAMPER 引脚会生成入侵检测事件 (RTC_TAMP[2:1])。选择边沿检测时，会禁止入侵输入上的内部上拉电阻。

注意： 为避免丢失入侵检测事件，用于边沿检测的信号与 TAMPxE 使用逻辑与操作来检测入侵事件，以避免丢失在使能 TAMPERx 引脚前发生的入侵事件。

- 当 TAMPxTRG = 0 时：如果 TAMPERx 复用功能在使能入侵检测之前已变为高电平（TAMPxE 位置 1），则使能 TAMPERx 后便会立即检测到入侵事件，即使在 TAMPxE 置 1 后 TAMPERx 引脚上未出现上升沿。
- 当 TAMPxTRG = 1 时：如果 TAMPERx 复用功能在使能入侵检测之前已变为低电平，则使能 TAMPERx 后便可立即检测到入侵事件（即使在 TAMPxE 置 1 后 TAMPERx 引脚上未出现下降沿）。

检测到入侵事件并清零后，应当在重新编程备份寄存器 (RTC_BKPxR) 之前禁止 TAMPERx 复用功能，然后再重新使能（TAMPxE 置 1）。这可防止应用在 TAMPERx 值仍指示入侵检测时，对备份寄存器执行写操作。这相当于对 TAMPERx 复用功能的电平检测。

注意： *当 V_{DD} 电源关闭时，入侵检测仍有效。要避免意外复位备份寄存器，应将 TAMPER 复用功能映射到的引脚从外部连接到正确的电平。*

对入侵输入的带过滤电平检测

通过将 TAMPFLT 设置为非零值可执行带过滤的电平检测。在 TAMPxTRG 位 (TAMP1TRG/TAMP2TRG) 指定的电平连续出现 2、4 或 8 个（取决于 TAMPFLT 值）采样时生成入侵检测事件。

除非通过将 TAMPPUDIS 置 1 禁止入侵输入，否则在入侵输入的状态被采样前，将通过 I/O 内部上拉电阻对这些输入预充电。预充电的持续时间由 TAMPPRCH 位确定，允许增大入侵输入上的电容。

可使用 TAMPFREQ 确定用于电平检测的采样频率，以便使入侵检测延迟与上拉电阻功耗之间达到最佳平衡。

注意： *有关上拉电阻的电气特性，请参见数据手册。*

TAMPER 复用功能检测

TAMPER1 备用功能 (RTC_TAMP1) 可映射到 RTC_AF1(PC13) 或 RTC_AF2 (PI8)，具体取决于 RTC_TAFCR 寄存器中 TAMP1INSEL 位的值（请参见*第 23.6.17 节：RTC 入侵和复用功能配置寄存器 (RTC_TAFCR)*）。修改 TAMP1INSEL 后必须将 TAMPE 位清零，以避免将 TAMPF 意外置 1。

TAMPER 2 复用功能对应于 RTC_TAMP2 引脚。

23.3.14 校准时钟输出

将 RTC_CR 寄存器中的 COE 位置 1 时，会在 RTC_CALIB 器件输出上提供一个参考时钟。如果 RTC_CR 寄存器中的 COSEL 位复位且 PREDIV_A = 0x7F，则 RTC_CALIB 频率为 $f_{RTCCLK}/64$。这相当于 RTCCLK 频率为 32.768 kHz 时，512 Hz 的校准输出。

RTC_CALIBR 寄存器中编程的校准值不会影响 RTC_CALIB 输出。RTC_CALIB 占空比是不规则的：下降沿上存在轻微抖动。因此推荐使用上升沿。

如果 COSEL 置 1 且“PREDIV_S+1”为 256 的非零整数倍（即：PREDIV_S[7:0] = 0xFF），则 RTC_CALIB 频率为 $f_{RTCCLK}/(256 * (PREDIV\_A+1))$。这相当于 RTCCLK 频率为 32.768 kHz 时，1 Hz 的校准输出，其中预分频器为默认值（PREDIV_A = Ox7F、PREDIV_S = 0xFF）。

校准复用功能输出

当将 RTC_CR 寄存器中的 COE 位置 1 时，RTC_AF1 上会使能校准复用功能 (RTC_CALIB)。

23.3.15 闹钟输出

闹钟输出有三种功能可供选择：ALRAF、ALRBF 和 WUTF。这些功能可反映 RTC_ISR 寄存器中相应标志的内容。

RTC_CR 寄存器中的 OSEL[1:0] 控制位用于激活 RTC_AF1 中的闹钟复用功能输出 (RTC_ALARM)，以及选择 RTC_ALARM 输出上的功能。

输出的极性由 RTC_CR 中的 POL 控制位确定，这样当 POL 置 1 时会输出选定标志位的相反值。

闹钟复用功能输出

使用 RTC_TAFCR 寄存器中的控制位 ALARMOUTTYPE 可将 RTC_ALARM 配置为输出开漏或输出上拉。

注意： *使能* RTC_ALARM *之后，其优先级高于 RTC_CALIB（COE 位的设置与* RTC_AF1 *无关）。*
*选择 RTC_CALIB 或 RTC_ALARM 时，*RTC_AF1 *会自动配置为输出复用功能。*

23.4 RTC 和低功耗模式

表 97. 低功耗模式对 RTC 的作用

模式	说明
睡眠	无影响 RTC 中断可使器件退出睡眠模式。
停止	当 RTC 时钟源为 LSE 或 LSI 时，RTC 保持工作状态。RTC 闹钟、RTC 入侵事件、RTC 时间戳事件、和 RTC 唤醒会使器件退出停机模式。
待机	当 RTC 时钟源为 LSE 或 LSI 时，RTC 保持工作状态。RTC 闹钟、RTC 入侵事件、RTC 时间戳事件、和 RTC 唤醒会使器件退出待机模式。

23.5 RTC 中断

所有 RTC 中断均与 EXTI 控制器相连。

要使能 RTC 闹钟中断，需按照以下顺序操作：

1. 将 EXTI 线 17 配置为中断模式并将其使能，然后选择上升沿有效。
2. 配置 NVIC 中的 RTC_Alarm IRQ 通道并将其使能。
3. 配置 RTC 以生成 RTC 闹钟（闹钟 A 或闹钟 B）。

要使能 RTC 唤醒中断，需按照以下顺序操作：

1. 将 EXTI 线 22 配置为中断模式并将其使能，然后选择上升沿有效。
2. 配置 NVIC 中的 RTC_WKUP IRQ 通道并将其使能。
3. 配置 RTC 以生成 RTC 唤醒定时器事件。

要使能 RTC 入侵中断，需按照以下顺序操作：

1. 将 EXTI 线 21 配置为中断模式并将其使能，然后选择上升沿有效。
2. 配置 NVIC 中的 TAMP_STAMP IRQ 通道并将其使能。
3. 配置 RTC 以检测 RTC 入侵事件。

要使能 RTC 时间戳中断，需按照以下顺序操作：

1. 将 EXTI 线 21 配置为中断模式并将其使能，然后选择上升沿有效。
2. 配置 NVIC 中的 TAMP_STAMP IRQ 通道并将其使能。
3. 配置 RTC 以检测 RTC 时间戳事件。

表 98. 中断控制位

中断事件	事件标志	使能控制位	退出睡眠模式	退出停止模式	退出待机模式
闹钟 A	ALRAF	ALRAIE	是	是(1)	是(1)
闹钟 B	ALRBF	ALRBIE	是	是(1)	是(1)
Wakeup	WUTF	WUTIE	是	是(1)	是(1)
TimeStamp	TSF	TSIE	是	是(1)	是(1)
Tamper1 检测	TAMP1F	TAMPIE	是	是(1)	是(1)
Tamper2 检测(2)	TAMP2F	TAMPIE	是	是(1)	是(1)

1. 仅当 RTC 时钟源为 LSE 或 LSI 时，才能从停机和待机模式唤醒。
2. 如果存在 RTC_TAMPER2 引脚。请参见器件数据手册的引脚排列。

23.6 RTC 寄存器

有关寄存器说明中使用的缩写，请参见参考手册中的*第 1.1 节*。

外设寄存器可按字（32 位）进行访问。

23.6.1 RTC 时间寄存器 (RTC_TR)

RTC time register

RTC_TR 是日历时间影子寄存器。只能在初始化模式下对该寄存器执行写操作。请参见*第 576 页的日历初始化和配置*和*第 577 页的读取日历*。

偏移地址：0x00

上电复位值：0x0000 0000

系统复位：当 BYPSHAD = 0 时为 0x0000 0000；当 BYPSHAD = 1 时不受影响。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved									PM	HT[1:0]		HU[3:0]			
									rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	MNT[2:0]			MNU[3:0]				Reserved	ST[2:0]			SU[3:0]			
	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw

位 31-24 保留

位 23 保留，必须保持复位值。

位 22 **PM**：AM/PM 符号 (AM/PM notation)
0：AM 或 24 小时制
1：PM

位 21:20 **HT[1:0]**：小时的十位（BCD 格式）(Hour tens in BCD format)

位 16:16 **HU[3:0]**：小时的个位（BCD 格式）(Hour units in BCD format)

位 15 保留，必须保持复位值。

位 14:12 **MNT[2:0]**：分钟的十位（BCD 格式）(Minute tens in BCD format)

位 11:8 **MNU[3:0]**：分钟的个位（BCD 格式）(Minute units in BCD format)

位 7 保留，必须保持复位值。

位 6:4 **ST[2:0]**：秒的十位（BCD 格式）(Second tens in BCD format)

位 3:0 **SU[3:0]**：秒的个位（BCD 格式）(Second units in BCD format)

注意： *此寄存器受写保护。第 576 页的RTC 寄存器写保护中介绍了写访问的过程。*

23.6.2 RTC 日期寄存器 (RTC_DR)

RTC date register

RTC_DR 是日历日期影子寄存器。只能在初始化模式下对该寄存器执行写操作。请参见*第 576 页的日历初始化和配置*和*第 577 页的读取日历*。

偏移地址：0x04

上电复位值：0x0000 2101

系统复位：当 BYPSHAD = 0 时为 0x0000 2101；当 BYPSHAD = 1 时不受影响。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved								YT[3:0]				YU[3:0]			
								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WDU[2:0]			MT	MU[3:0]				Reserved		DT[1:0]		DU[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw

位 31-24 保留

位 23:20 **YT[3:0]**：年份的十位（BCD 格式）(Year tens in BCD format)

位 19:16 **YU[3:0]**：年份的个位（BCD 格式）(Year units in BCD format)

位 15:13 **WDU[2:0]**：星期几的个位 (Week day units)

000：禁止
001：星期一
...
111：星期日

位 12 **MT**：月份的十位（BCD 格式）(Month tens in BCD format)

位 11:8 **MU**：月份的个位（BCD 格式）(Month units in BCD format)

位 7:6 保留，必须保持复位值。

位 5:4 **DT[1:0]**：日期的十位（BCD 格式）(Date tens in BCD format)

位 3:0 **DU[3:0]**：日期的个位（BCD 格式）(Date units in BCD format)

注意： *此寄存器受写保护。第 576 页的RTC 寄存器写保护中介绍了写访问的过程。*

23.6.3 RTC 控制寄存器 (RTC_CR)

RTC control register

偏移地址：0x08

上电复位值：0x0000 0000

系统复位：不受影响

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved								COE	OSEL[1:0]		POL	COSEL	BKP	SUB1H	ADD1H
								rw	rw	rw	rw	rw	rw	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TSIE	WUTIE	ALRBE	ALRAIE	TSE	WUTE	ALRBE	ALRAE	DCE	FMT	BYPS HAD	REFCKON	TSEDGE	WUCKSEL[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:24 保留，必须保持复位值。

位 23 **COE**：校准输出使能 (Calibration output enable)

该位使能 RTC_CALIB 输出
0：禁止校准输出
1：使能校准输出

位 22:21 **OSEL[1:0]**：输出选择 (Output selection)
这些位用于选择要连接到 RTC_ALARM 输出的标志
00：禁止输出
01：使能闹钟 A 输出
10：使能闹钟 B 输出
11：使能唤醒输出

位 20 **POL**：输出极性 (Output polarity)
该位用于配置 RTC_ALARM 输出的极性
0：当 ALRAF/ALRBF/WUTF 置 1 时（取决于 OSEL[1:0]），该引脚为高电平
1：当 ALRAF/ALRBF/WUTF 置 1 时（取决于 OSEL[1:0]），该引脚为低电平

位 19 **COSEL**：校准输出选择 (Calibration output selection)
当 COE=1 时，该位可选择 RTC_CALIB 上输出的信号。
0：校准输出为 512 Hz
1：校准输出为 1 Hz
在 RTCCLK 为 32.768 kHz 且预分频器为其默认值（PREDIV_A=127 且 PREDIV_S=255）的条件下，这些频率有效。请参见*第 23.3.14 节：校准时钟输出*。

位 18 **BKP**：备份 (Backup)
用户可对此位执行写操作以记录是否已对夏令时进行更改。

位 17 **SUB1H**：减少 1 小时（冬季时间更改）(*S*ubtract 1 hour (winter time change))
当该位在初始化模式以外的模式下置 1 时，如果当前小时不是 0，则日历时间将减少 1 小时。此位始终读为 0。
当前小时为 0 时，将此位置 1 没有任何作用。
0：无作用。
1：将当前时间减少 1 小时。这可用于冬季时间更改。

位 16 **ADD1H**：增加 1 小时（夏季时间更改）(Add 1 hour (summer time change))
当该位在初始化模式以外的模式下置 1 时，日历时间将增加 1 小时。此位始终读为 0。
0：无作用。
1：将当前时间增加 1 小时。这可用于夏季时间更改

位 15 **TSIE**：时间戳中断使能 (Timestamp interrupt enable)
0：禁止时间戳中断
1：使能时间戳中断

位 14 **WUTIE**：使能唤醒定时器使能 (Wakeup timer interrupt enable)
0：禁止唤醒定时器中断
1：使能唤醒定时器中断

位 13 **ALRBIE**：*闹钟 B 中断使能 (Alarm B interrupt enable)*
0：闹钟 B 中断禁止
1：闹钟 B 中断使能

位 12 **ALRAIE**：闹钟 A 中断使能 (Alarm A interrupt enable)
0：禁止闹钟 A 中断
1：使能闹钟 A 中断

位 11 **TSE**：时间戳使能 (Time stamp enable)
0：禁止时间戳
1：使能时间戳

位 10 **WUTE**：唤醒定时器使能 (Wakeup timer enable)
0：禁止唤醒定时器
1：使能唤醒定时器

位 9 **ALRBE**：*闹钟 B 使能 (Alarm B enable)*
0：禁止闹钟 B
1：使能闹钟 B

位 8 **ALRAE**：闹钟 A 使能 (Alarm A enable)
0：禁止闹钟 A
1：使能闹钟 A

位 7 **DCE**：粗略数字校准使能 (Coarse digital calibration enable)
0：禁止数字校准
1：使能数字校准
PREDIV_A 必须大于或等于 6

位 6 **FMT**：小时格式 (Hour format)
0：24 小时/天格式
1：AM/PM 小时格式

位 5 **BYPSHAD**：旁路影子寄存器 (Bypass the shadow registers)
0：日历值（从 RTC_SSR、RTC_TR 和 RTC_DR 读取时）取自影子寄存器，该影子寄存器每两个 RTCCLK 周期更新一次。
1：日历值（从 RTC_SSR、RTC_TR 和 RTC_DR 读取时）直接取自日历计数器。
注意：如果 APB1 时钟的频率低于 7 倍的 RTCCLK 频率，则必须将 BYPSHAD 置“1”。

位 4 **REFCKON**：参考时钟检测使能（50 Hz 或 60 Hz）(Reference clock detection enable (50 or 60 Hz))
0：禁止参考时钟检测
1：使能参考时钟检测
注意：PREDIV_S 必须为 0x00FF。

位 3 **TSEDGE**：时间戳事件有效边沿 (Timestamp event active edge)
0：TIMESTAMP 上升沿生成时间戳事件
1：TIMESTAMP 下降沿生成时间戳事件
TSEDGE 发生更改时，必须复位 TSE 以避免将 TSF 意外置 1

位 2:0 **WUCKSEL[2:0]**：唤醒时钟选择 (Wakeup clock selection)
000：选择 RTC/16 时钟
001：选择 RTC/8 时钟
010：选择 RTC/4 时钟
011：选择 RTC/2 时钟
10x：选择 ck_spre 时钟（通常为 1 Hz）
11x：选择 ck_spre 时钟（通常为 1 Hz）并将 WUT 计数器值增加 2^{16}（见下面的注释）

注意： *WUT = 唤醒单元计数器值。当 WUCKSEL[2:1 = 11] 时，WUT = (0x0000 to 0xFFFF) + 0x10000（增加的值）。*

只能在初始化模式下 (RTC_ISR/INITF = 1) 对该寄存器的位 7、6 和 4 执行写操作。

只能在 RTC_CR WUTE 位 = 0 且 RTC_ISR WUTWF 位 = 1 时对该寄存器的位 2 到 0 执行写操作。

建议不要在日历小时递增时更改小时，因为这样做会屏蔽日历小时的增量。

ADD1H 和 SUB1H 的更改在下一秒生效。

此寄存器受写保护。第 576 页的RTC 寄存器写保护中介绍了写访问的过程。

23.6.4 RTC 初始化和状态寄存器 (RTC_ISR)

RTC initialization and status register

偏移地址：0x0C

上电复位值：0x0000 0007

系统复位值：不受影响（INIT、INITF 和 RSF 除外，它们在复位时被清零）。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															RECALPF
															r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TAMP2F	TAMP1F	TSOVF	TSF	WUTF	ALRBF	ALRAF	INIT	INITF	RSF	INITS	SHPF	WUTWF	ALRBWF	ALRAWF
	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rw	r	rc_w0	r	rc_w0	r	r	r

位 31:17 保留

位 16 **RECALPF**：重新校准挂起标志 (Recalibration pending Flag)

当软件对 RTC_CALR 寄存器执行写操作时，RECALPF 状态标志将自动置“1”，指示 RTC_CALR 寄存器已屏蔽。当采用新的校准设置时，该位恢复为“0”。请参见*动态重校准*一节。

位 15 保留，必须保持复位值。

位 14 **TAMP2F**：TAMPER2 检测标志 (TAMPER2 detection flag)

在入侵输入 2 上检测到入侵检测事件时，由硬件将此标志置 1。

该标志由软件写零清除。

位 13 **TAMP1F**：入侵检测标志 (Tamper detection flag)

当检测到入侵检测事件时，由硬件将此标志置 1。

该标志由软件写零清除。

位 12 **TSOVF**：时间戳溢出标志 (Timestamp overflow flag)

当在 TSF 已置 1 的情况下发生时间戳事件时，由硬件将此标志置 1。

该标志由软件写零清除。建议仅在 TSF 位清零之后再检查并清零 TSOVF 位。否则，如果时间戳事件恰好在清零 TSF 位之前刚刚发生，则溢出事件可能会被漏掉。

位 11 **TSF**：时间戳标志 (Timestamp flag)

发生时间戳事件时，由硬件将此标志置 1。

该标志由软件写零清除。

位 10 **WUTF**：唤醒定时器标志 (Wakeup timer flag)

当唤醒自动重载计数器计数到 0 时，由硬件将此标志置 1。

该标志由软件写零清除。

软件必须在 WUTF 再次置 1 的 1.5 个 RTCCLK 周期之前将该标志清零。

位 9 **ALRBF**：闹钟 B 标志 (Alarm B flag)

当时间/日期寄存器（RTC_TR 和 RTC_DR）与闹钟 B 寄存器 (RTC_ALRMBR) 匹配时，由硬件将该标志置 1。

该标志由软件写零清除。

位 8 **ALRAF**：闹钟 A 标志 (Alarm A flag)

当时间/日期寄存器（RTC_TR 和 RTC_DR）与闹钟 A 寄存器 (RTC_ALRMAR) 匹配时，由硬件将该标志置 1。

该标志由软件写零清除。

位 7 **INIT**：初始化模式 (Initialization mode)

0：自由运行模式。

1：初始化模式，用于编程时间和日期寄存器（RTC_TR 和 RTC_DR）以及预分频器寄存器 (RTC_PRER)。计数器停止计数，当 INIT 被复位后，计数器从新值开始计数。

位 6 **INITF**：初始化标志 (Initialization flag)

当此位置 1 时，RTC 处于初始化状态，此时可更新事件、日期和预分频器寄存器。

0：不允许更新日历寄存器。

1：允许更新日历寄存器。

位 5 **RSF**：寄存器同步标志 (Registers synchronization flag)

每次将日历寄存器的值复制到影子寄存器（RTC_SSRx、RTC_TRx 和 RTC_DRx）时，都会由硬件将此位置 1。在初始化模式下、平移操作挂起时 (SHPF=1) 或在旁路影子寄存器模式 (BYPSHAD=1) 下，该位由硬件清零。该位还可由软件清零。

0：日历影子寄存器尚未同步

1：日历影子寄存器已同步

位 4 **INITS**：初始化状态标志 (Initialization status flag)

当日历年份字段不为 0 时（上电复位状态），由硬件将该位置 1。

0：日历尚未初始化

1：日历已经初始化

位 3 **SHPF**：平移操作挂起 (Shift operation pending)

0：没有平移操作挂起

1：某个平移操作挂起

只要通过对 RTC_SHIFTR 寄存器执行写操作来启动平移操作，此标志便由硬件置 1。执行完相应的平移操作后，此标志由硬件清零。对 SHPF 执行写入操作不起作用。

位 2 **WUTWF**：唤醒定时器写标志 (Wakeup timer write flag)

在 RTC_CR 寄存器中的 WUTE 位置 0 后，当唤醒定时器值可更改时，由硬件将该位置 1。

0：不允许更新唤醒定时器配置

1：允许更新唤醒定时器配置

位 1 **ALRBWF**：闹钟 B 写标志 (Alarm B write flag)

在 RTC_CR 寄存器中的 ALRBIE 位置 0 之后，当闹钟 B 的值可更改时，由硬件将该位置 1。该位在初始化模式下由硬件清零。

0：不允许更新闹钟 B

1：允许更新闹钟 B

位 0 **ALRAWF**：闹钟 A 写标志 (Alarm A write flag)

在 RTC_CR 寄存器中的 ALRAE 位置 0 后，当闹钟 A 的值可更改时，由硬件将该位置 1。该位在初始化模式下由硬件清零。

0：不允许更新闹钟 A

1：允许更新闹钟 A

注意： *将 ALRAF、ALRBF、WUTF 和 TSF 位编程为 0 之后 2 个 APB 时钟周期，清零生效。*

此寄存器受写保护（RTC_ISR[13:8] 位除外）。第 576 页的RTC 寄存器写保护中介绍了写访问的过程。

23.6.5 RTC 预分频器寄存器 (RTC_PRER)

RTC prescaler register

偏移地址：0x10

上电复位值：0x007F 00FF

系统复位：不受影响

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved									PREDIV_A[6:0]						
									rw	rw	rw	rw	rw	rw	rw

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

位 31:24 保留

位 23 保留，必须保持复位值。

位 22:16 **PREDIV_A[6:0]**：异步预分频系数 (Asynchronous prescaler factor)

下面是异步分频系数的公式：

ck_apre 频率 = RTCCLK 频率/(PREDIV_A+1)

注意：PREDIV_A [6:0]= 000000 为禁用值。

位 15 保留，必须保持复位值。

位 14:0 **PREDIV_S[14:0]**：同步预分频系数 (Synchronous prescaler factor)

下面是同步分频系数的公式：

ck_spre 频率 = ck_apre 频率/(PREDIV_S+1)

注意： *只能在初始化模式下对该寄存器执行写操作。必须通过两次独立的写访问执行初始化。请参见第 576 页的日历初始化和配置。*

此寄存器受写保护。第 576 页的RTC 寄存器写保护中介绍了写访问的过程。

23.6.6 RTC 唤醒定时器寄存器 (RTC_WUTR)

RTC wakeup timer register

偏移地址：0x14

上电复位值：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
Reserved															

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

位 31:16 保留

位 15:0 **WUT[15:0]**：唤醒自动重载值位 (Wakeup auto-reload value bit)

当使能唤醒定时器时（WUTE 置 1），每 (WUT[15:0] + 1) 个 ck_wut 周期将 WUTF 标志置 1 一次。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。

注意：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。

注意：仅当 RTC_ISR 中的 WUTWF 置 1 时才可对该寄存器执行写操作。

此寄存器受写保护。第 576 页的RTC 寄存器写保护中介绍了写访问的过程。

23.6.7 RTC 校准寄存器 (RTC_CALIBR)

RTC calibration register

偏移地址：0x18

上电复位值：0x0000 0000

系统复位：不受影响

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

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

位 31:8 保留

位 7 **DCS**：数字校准符号 (Digital calibration sign)

0：正校准：增加日历更新频率

1：负校准：降低日历更新频率

位 6:5 保留，必须保持复位值。

位 4:0 **DC[4:0]**：数字校准 (Digital calibration)

DCS = 0（正校准）

00000：+ 0 ppm

00001：+ 4 ppm（舍入值）

00010：+ 8 ppm（舍入值）

..

11111：+ 126 ppm（舍入值）

DCS = 1（负校准）

00000：–0 ppm

00001：–2 ppm（舍入值）

00010：–4 ppm（舍入值）

..

11111：–63 ppm（舍入值）

有关准确的步骤值，请参见*第 580 页的例如当 RTCCLK=32.768 kHz 且 PREDIV_A+1=128 时*。

注意：只能在初始化模式（RTC_ISR/INITF = “1”）下对此寄存器执行写操作。

此寄存器受写保护。第 576 页的RTC 寄存器写保护中介绍了写访问的过程。

23.6.8 RTC 闹钟 A 寄存器 (RTC_ALRMAR)

RTC alarm A register

偏移地址：0x1C

上电复位值：0x0000 0000

系统复位：不受影响

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MSK4	WDSEL	DT[1:0]		DU[3:0]				MSK3	PM	HT[1:0]		HU[3:0]			
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
MSK2	MNT[2:0]			MNU[3:0]				MSK1	ST[2:0]			SU[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31 **MSK4**：闹钟 A 日期掩码 (Alarm A date mask)
0：如果日期/日匹配，则闹钟 A 置 1
1：在闹钟 A 比较中，日期/日无关

位 30 **WDSEL**：星期几选择 (Week day selection)
0：DU[3:0] 代表日期的个位
1：DU[3:0] 代表星期几。DT[1:0] 为无关位。

位 29:28 **DT[1:0]**：日期的十位（BCD 格式）(Date tens in BCD format)。

位 27:24 **DU[3:0]**：日期的个位或日（BCD 格式）(Date units or day in BCD format)。

位 23 **MSK3**：闹钟 A 小时掩码 (Alarm A hours mask)
0：如果小时匹配，则闹钟 A 置 1
1：在闹钟 A 比较中，小时无关

位 22 **PM**：AM/PM 符号 (AM/PM notation)
0：AM 或 24 小时制
1：PM

位 21:20 **HT[1:0]**：小时的十位（BCD 格式）(Hour tens in BCD format)。

位 19:16 **HU[3:0]**：小时的个位（BCD 格式）(Hour units in BCD format)。

位 15 **MSK2**：闹钟 A 分钟掩码 (Alarm A minutes mask)
0：如果分钟匹配，则闹钟 A 置 1
1：在闹钟 A 比较中，分钟无关

位 14:12 **MNT[2:0]**：分钟的十位（BCD 格式）(Minute tens in BCD format)。

位 11:8 **MNU[3:0]**：分钟的个位（BCD 格式）(Minute units in BCD format)。

位 7 **MSK1**：闹钟 A 秒掩码 (Alarm A seconds mask)
0：如果秒匹配，则闹钟 A 置 1
1：在闹钟 A 比较中，秒无关

位 6:4 **ST[2:0]**：秒的十位（BCD 格式）(Second tens in BCD format)。

位 3:0 **SU[3:0]**：秒的个位（BCD 格式）(Second units in BCD format)。

注意：仅当 RTC_ISR 中的 ALRAWF 置 1 时或在初始化模式下，才可以对该寄存器执行写操作。此寄存器受写保护。第 576 页的RTC 寄存器写保护中介绍了写访问的过程。

23.6.9 RTC 闹钟 B 寄存器 (RTC_ALRMBR)

RTC alarm B register

偏移地址：0x20

上电复位值：0x0000 0000

系统复位：不受影响

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MSK4	WDSEL	DT[1:0]		DU[3:0]				MSK3	PM	HT[1:0]		HU[3:0]			
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
MSK2	MNT[2:0]			MNU[3:0]				MSK1	ST[2:0]			SU[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31 **MSK4**：闹钟 B 日期掩码 (Alarm B date mask)
0：如果日期和日匹配，则闹钟 B 置 1
1：在闹钟 B 比较中，日期和日无关

位 30 **WDSEL**：星期几选择 (Week day selection)
0：DU[3:0] 代表日期的个位。
1：DU[3:0] 代表星期几。DT[1:0] 为无关位。

位 29:28 **DT[1:0]**：日期的十位（BCD 格式）(Date tens in BCD format)

位 27:24 **DU[3:0]**：日期个位或日（BCD 格式）(Date units or day in BCD format)

位 23 **MSK3**：闹钟 B 小时掩码 (Alarm B hours mask)
0：如果小时匹配，则闹钟 B 置 1
1：在闹钟 B 比较中，小时无关

位 22 **PM**：AM/PM 符号 (AM/PM notation)
0：AM 或 24 小时制
1：PM

位 21:20 **HT[1:0]**：小时的十位（BCD 格式）(Hour tens in BCD format)

位 19:16 **HU[3:0]**：小时的个位（BCD 格式）(Hour units in BCD format)

位 15 **MSK2**：闹钟 B 分钟掩码 (Alarm B minutes mask)
0：如果分钟匹配，则闹钟 B 置 1
1：在闹钟 B 比较中，分钟无关

位 14:12 **MNT[2:0]**：分钟的十位（BCD 格式）(Minute tens in BCD format)

位 11:8 **MNU[3:0]**：分钟的个位（BCD 格式）(Minute units in BCD format)

位 7 **MSK1**：闹钟 B 秒掩码 (Alarm B seconds mask)
0：如果秒匹配，则闹钟 B 置 1
1：在闹钟 B 比较中，秒无关

位 6:4 **ST[2:0]**：秒的十位（BCD 格式）(Second tens in BCD format)

位 3:0 **SU[3:0]**：秒的个位（BCD 格式）(Second units in BCD format)

注意：仅当 RTC_ISR 中的 ALRBWF 置 1 时或在初始化模式下，才可以对该寄存器执行写操作。此寄存器受写保护。第 576 页的RTC 寄存器写保护中介绍了写访问的过程。

23.6.10 RTC 写保护寄存器 (RTC_WPR)

RTC write protection register

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

位 31:8 保留，必须保持复位值。

位 7:0 **KEY**：写保护关键字 (Write protection key)

可通过软件对该字节执行写操作。

读取该字节时，始终返回 0x00。

有关如何解锁 RTC 寄存器写保护的介绍，请参见 *RTC 寄存器写保护*。

23.6.11 RTC 亚秒寄存器 (RTC_SSR)

RTC sub second register

偏移地址：0x28

上电复位值：0x0000 0000

系统复位：当 BYPSHAD = 0 时为 0x0000 0000；当 BYPSHAD = 1 时不受影响。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
r	r	r	r	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
SS[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

位 31:16 保留

位 15:0 **SS**：亚秒值 (Sub second value)

SS[15:0] 是同步预分频器计数器的值。此亚秒值可根据以下公式得出：

亚秒值 = (PREDIV_S - SS) / (PREDIV_S + 1)

注意：仅当执行平移操作之后，SS 才能大于 PREDIV_S。在这种情况下，正确的时间/日期比 RTC_TR/RTC_DR 所指示的时间/日期慢一秒钟。

23.6.12 RTC 平移控制寄存器 (RTC_SHIFTR)

RTC shift control register

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
ADD1S	Reserved														
w	r	r	r	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
Res.	SUBFS[14:0]														
r	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

位 31 **ADD1S**：增加一秒钟 (Add one second)

0：无作用

1：对时钟/日历增加一秒钟

此位为只写位且始终读为 0。当平移操作挂起（RTC_ISR 中的 SHPF=1）时，对该位执行写操作无作用。

此函数应与 SUBFS 配合使用（请参见下文介绍），以便有效地向原子操作机制的时钟添加亚秒值。

位 31:15 保留

位 14:0 **SUBFS**：减少亚秒值 (Subtract a fraction of a second)

此位为只写位且始终读为 0。当平移操作挂起（RTC_ISR 中的 SHPF=1）时，对该位执行写操作无作用。

写入 SUBFS 的值将加到同步预分频器计数器中。由于该计数器递减计数，此操作可有效地从时钟减去（延迟）以下时间：

延迟（秒）= SUBFS / (PREDIV_S + 1)

当 ADD1S 函数与 SUBFS 结合使用时，可有效地将亚秒值增加到时钟（提前时钟），使时钟提前以下时间：

提前（秒）= (1 - (SUBFS / (PREDIV_S + 1)))。

注意：对 SUBFS 执行写操作将使 RSF 清零。软件随后会等待至 RSF=1 以确定影子寄存器已更新为平移后的时间。

请参见*第 23.3.8 节：RTC 同步*。

注意：此寄存器受写保护。第 576 页的RTC 寄存器写保护中介绍了写访问的过程。

23.6.13 RTC 时间戳时间寄存器 (RTC_TSTR)

RTC time stamp time register

偏移地址：0x30

上电复位值：0x0000 0000

系统复位：不受影响

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved									PM	HT[1:0]		HU[3:0]			
									r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	MNT[2:0]			MNU[3:0]				Reserved	ST[2:0]			SU[3:0]			
	r	r	r	r	r	r	r		r	r	r	r	r	r	r

位 31:23 保留，必须保持复位值。

位 22 **PM**：AM/PM 符号 (AM/PM notation)

0：AM 或 24 小时制

1：PM

位 21:20 **HT[1:0]**：小时的十位（BCD 格式）(Hour tens in BCD format)。

位 19:16 **HU[3:0]**：小时的个位（BCD 格式）(Hour units in BCD format)。

位 15 保留，必须保持复位值。

位 14:12 **MNT[2:0]**：分钟的十位（BCD 格式）(Minute tens in BCD format)。

位 11:8 **MNU[3:0]**：分钟的个位（BCD 格式）(Minute units in BCD format)。

位 7 保留，必须保持复位值。

位 6:4 **ST[2:0]**：秒的十位（BCD 格式）(Second tens in BCD format)。

位 3:0 **SU[3:0]**：秒的个位（BCD 格式）(Second units in BCD format)。

注意： *仅当 RTC_ISR 中的 TSF 置 1 时，该寄存器的内容才有效。当 TSF 位复位时，清零该寄存器。*

23.6.14 RTC 时间戳日期寄存器 (RTC_TSDR)

RTC time stamp date register

偏移地址：0x34

上电复位值：0x0000 0000

系统复位：不受影响

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WDU[1:0]			MT	MU[3:0]				Reserved		DT[1:0]		DU[3:0]			
r	r	r	r	r	r	r	r			r	r	r	r	r	r

位 31:16 保留，必须保持复位值。

位 15:13 **WDU[1:0]**：星期几的个位 (Week day units)

位 12 **MT**：月份的十位（BCD 格式）(Month tens in BCD format)

位 11:8 **MU[3:0]**：月份的个位（BCD 格式）(Month units in BCD format)

位 7:6 保留，必须保持复位值。

位 5:4 **DT[1:0]**：日期的十位（BCD 格式）(Date tens in BCD format)

位 3:0 **DU[3:0]**：日期的个位（BCD 格式）(Date units in BCD format)

注意： *仅当 RTC_ISR 中的 TSF 置 1 时，该寄存器的内容才有效。当 TSF 位复位时，清零该寄存器。*

23.6.15 RTC 时间戳亚秒寄存器 (RTC_TSSSR)

RTC timestamp sub second register

偏移地址：0x38

上电复位值：0x0000 0000

系统复位：不受影响

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
r	r	r	r	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
SS[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

位 31:16 保留

位 15:0 **SS**：亚秒值 (Sub second value)

当发生时间戳事件时，SS[15:0] 是同步预分频器计数器的值。

注意： *仅当 RTC_ISR/TSF 置 1 时，该寄存器的内容才有效。当 RTC_ISR/TSF 位复位时，清零该寄存器。*

23.6.16 RTC 校准寄存器 (RTC_CALR)

RTC calibration register

偏移地址：0x3C

上电复位值：0x0000 0000

系统复位：不受影响

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
r	r	r	r	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
CALP	CALW8	CALW16	Reserved				CALM[8:0]								
rw	rw	rw	r	r	r	r	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:16 保留

位 15 **CALP**：将 RTC 的频率增加 488.5 ppm (Increase frequency of RTC by 488.5 ppm)
0：不增加 RTCCLK 脉冲。
1：每 2^{11} 个脉冲有效插入一个 RTCCLK 脉冲（将频率增加 488.5 ppm）。
此功能应与 CALM 结合使用，后者在高分辨率下会降低日历的频率。如果输入频率为 32768 Hz，则在 32 秒窗口中增加的 RTCCLK 脉冲数按如下公式计算：(512 * CALP) - CALM。
请参见*第 23.3.11 节：RTC 精密数字校准*。

位 14 **CALW8**：使用 8 秒校准周期 (Use an 8-second calibration cycle period)
当 CALW8 置“1”时，选择 8 秒校准周期。
CALW8=“1”时，CALM[1:0] 将始终保持为“00”。
请参见*第 23.3.11 节：RTC 精密数字校准*。

位 13 **CALW16**：使用 16 秒校准周期 (Use a 16-second calibration cycle period)
当 CALW16 置“1”时，选择 16 秒校准周期。如果 CALW8=1，则必须将该位置“1”。
注意：当 CALW16=“1”时，CALM[0] 将始终保持为“0”。
请参见*第 23.3.11 节：RTC 精密数字校准*。

位 12:9 保留

位 8:0 **CALM[8:0]**：负校准 (Calibration minus)
在 2^{20} 个 RTCCLK 脉冲内屏蔽 CALM 个脉冲（如果输入频率为 32768 Hz，则为 32 秒）来降低日历的频率。其分辨率为 0.9537 ppm。
要提高日历的频率，则应将此功能与 CALP 结合使用。
请参见*第 580 页的第 23.3.11 节：RTC 精密数字校准*。

注意：此寄存器受写保护。第 576 页的RTC 寄存器写保护中介绍了写访问的过程。

23.6.17 RTC 入侵和复用功能配置寄存器 (RTC_TAFCR)

RTC tamper and alternate function configuration register

偏移地址：0x40

上电复位值：0x0000 0000

系统复位：不受影响

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved													ALARMOUT TYPE	TSIN SEL	TAMP1 INSEL
													rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TAMP-PUDIS	TAMP-PRCH[1:0]		TAMPFLT[1:0]		TAMPFREQ[2:0]			TAMPTS	Reserved		TAMP2-TRG	TAMP2E	TAMPIE	TAMP1TRG	TAMP1E
rw	rw	rw	rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw

位 31:19 保留。始终读为 0。

位 18 **ALARMOUTTYPE**：RTC_ALARM 输出类型 (RTC_ALARM output type)
0：RTC_ALARM 为开漏输出
1：RTC_ALARM 为推挽输出

位 17 **TSINSEL**：TIMESTAMP 映射 (TIMESTAMP mapping)
0：RTC_AF1 用作 TIMESTAMP
1：RTC_AF2 用作 TIMESTAMP

位 16 **TAMP1INSEL**：TAMPER1 映射 (TAMPER1 mapping)
0：RTC_AF1 用作 TAMPER1
1：RTC_AF2 用作 TAMPER1
注意：更改 TAMP1INSEL 时必须复位 TAMP1E，以避免将 TAMP1F 意外置 1。

位 15 **TAMPPUDIS**：TAMPER 禁止内部上拉 (TAMPER pull-up disable)
该位决定在每次采样之前是否对入侵引脚都进行预充电。
0：采样之前对入侵引脚进行预充电（使能内部上拉）
1：禁止对入侵引脚进行预充电
注意：

位 14:13 **TAMPPRCH[1:0]**：入侵预充电持续时间 (Tamper precharge duration)
这些位决定了在每次采样之前激活上拉的持续时间。TAMPPRCH 对每个入侵输入都有效。
0x0：1 个 RTCCLK 周期
0x1：2 个 RTCCLK 周期
0x2：4 个 RTCCLK 周期
0x3：8 个 RTCCLK 周期

位 12:11 **TAMPFLT[1:0]**：入侵过滤器计数 (Tamper filter count)
这些位决定了为激活入侵事件所需的指定电平 (TAMP*TRG) 上的连续采样次数。TAMPFLT 对每个入侵输入都有效。
0x0：在入侵输入转变为有效电平的边沿激活入侵（入侵输入上无内部上拉）。
0x1：在有效电平上连续执行 2 次采样后激活入侵。
0x2：在有效电平上连续执行 4 次采样后激活入侵。
0x3：在有效电平上连续执行 8 次采样后激活入侵。

位 10:8 **TAMPFREQ[2:0]**：入侵采样频率 (Tamper sampling frequency)
决定对每个入侵输入进行采样时的频率。
0x0：RTCCLK / 32768（RTCCLK = 32768 Hz 时为 1 Hz）
0x1：RTCCLK / 16384（RTCCLK = 32768 Hz 时为 2 Hz）
0x2：RTCCLK / 8192（RTCCLK = 32768 Hz 时为 4 Hz）
0x3：RTCCLK / 4096（RTCCLK = 32768 Hz 时为 8 Hz）
0x4：RTCCLK / 2048（RTCCLK = 32768 Hz 时为 16 Hz）
0x5：RTCCLK / 1024（RTCCLK = 32768 Hz 时为 32 Hz）
0x6：RTCCLK / 512（RTCCLK = 32768 Hz 时为 64 Hz）
0x7：RTCCLK / 256（RTCCLK = 32768 Hz 时为 128 Hz）

位 7 **TAMPTS**：发生入侵检测事件时激活时间戳 (Activate timestamp on tamper detection event)
0：发生入侵检测事件时不保存时间戳
1：发生入侵检测事件时保存时间戳
即便 RTC_CR 寄存器中的 TSE=0，TAMPTS 仍有效。

位 6:5 保留。始终读为 0。

位 4 **TAMP2TRG**：入侵 2 的有效电平 (Active level for tamper 2)

如果 TAMPFLT != 00：

0：TAMPER2 保持低电平会触发入侵检测事件。

1：TAMPER2 保持高电平会触发入侵检测事件。

如果 TAMPFLT = 00：

0：TAMPER2 上升沿会触发入侵检测事件。

1：TAMPER2 下降沿会触发入侵检测事件。

位 3 **TAMP2E**：入侵 2 检测使能 (Tamper 2 detection enable)

0：禁止入侵 2 检测

1：使能入侵 2 检测

位 2 **TAMPIE**：入侵中断使能 (Tamper interrupt enable)

0：禁止入侵中断

1：使能入侵中断

位 1 **TAMP1TRG**：入侵 1 的有效电平 (Active level for tamper 1)

如果 TAMPFLT != 00

0：TAMPER1 保持低电平会触发入侵检测事件。

1：TAMPER1 保持高电平会触发入侵检测事件。

如果 TAMPFLT = 00：

0：TAMPER1 上升沿会触发入侵检测事件。

1：TAMPER1 下降沿会触发入侵检测事件。

注意： 如果 TAMPFLT = 0，则更改 TAMP1TRG 时必须复位 TAMP1E，以避免将 TAMP1F 意外置 1。

位 0 **TAMP1E**：入侵 1 检测使能 (Tamper 1 detection enable)

0：禁止入侵 1 检测

1：使能入侵 1 检测

23.6.18 RTC 闹钟 A 亚秒寄存器 (RTC_ALRMASSR)

RTC alarm A sub second register

偏移地址：0x44

上电复位值：0x0000 0000

系统复位：不受影响

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved				MASKSS[3:0]				Reserved							
r	r	r	r	rw	rw	rw	rw	r	r	r	r	r	r	r	r

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

位 31:28 保留

位 27:24 **MASKSS[3:0]**：屏蔽从此位开始的最高有效位 (Mask the most-significant bits starting at this bit)

0：不对闹钟 A 的亚秒进行比较。当秒单元递增时设置闹钟（假定其余字段均匹配）。
1：在闹钟 A 比较中，SS[14:1] 为无关位。仅比较 SS[0]。
2：在闹钟 A 比较中，SS[14:2] 为无关位。仅比较 SS[1:0]。
3：在闹钟 A 比较中，SS[14:3] 为无关位。仅比较 SS[2:0]。
...
12：在闹钟 A 比较中，SS[14:12] 为无关位。比较 SS[11:0]。
13：在闹钟 A 比较中，SS[14:13] 为无关位。比较 SS[12:0]。
14：在闹钟 A 比较中，SS[14] 为无关位。比较 SS[13:0]。
15：所有 15 个 SS 位均进行比较，并且必须全部匹配才能激活闹钟。
同步计数器的溢出位（位 15）从不进行比较。仅当执行平移操作之后，此位才不为 0。

位 23:15 保留

位 14:0 **SS[14:0]**：亚秒值 (Sub seconds value)

该值与同步预分频器计数器的内容进行比较以确定是否要激活闹钟 A。仅比较位 0 到 MASKSS-1。

注意： *仅当 RTC_CR 中的 ALRAE 复位时或在初始化模式下，才可以对该寄存器执行写操作。此寄存器受写保护。第 576 页的RTC 寄存器写保护中介绍了写访问的过程。*

23.6.19 RTC 闹钟 B 亚秒寄存器 (RTC_ALRMASSR)

RTC alarm B sub second register

偏移地址：0x48

上电复位值：0x0000 0000

系统复位：不受影响

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved				MASKSS[3:0]				Reserved							
r	r	r	r	rw	rw	rw	rw	r	r	r	r	r	r	r	r

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

位 31:28 保留

位 27:24 **MASKSS[3:0]**：屏蔽从此位开始的最高有效位 (Mask the most-significant bits starting at this bit)

0x0：不对闹钟 B 的亚秒进行比较。当秒单元递增时设置闹钟（假定其余字段均匹配）。
0x1：在闹钟 B 比较中，SS[14:1] 为无关位。仅比较 SS[0]。
0x2：在闹钟 B 比较中，SS[14:2] 为无关位。仅比较 SS[1:0]。
0x3：在闹钟 B 比较中，SS[14:3] 为无关位。仅比较 SS[2:0]。
...
0xC：在闹钟 B 比较中，SS[14:12] 为无关位。比较 SS[11:0]。
0xD：在闹钟 B 比较中，SS[14:13] 为无关位。比较 SS[12:0]。
0xE：在闹钟 B 比较中，SS[14] 为无关位。比较 SS[13:0]。
0xF：所有 15 个 SS 位均进行比较，并且必须全部匹配才能激活闹钟。
同步计数器的溢出位（位 15）从不进行比较。仅当执行平移操作之后，此位才不为 0。

位 23:15 保留

位 14:0 **SS[14:0]**：亚秒值 (Sub seconds value)

该值与同步预分频器计数器的内容进行比较以确定是否要激活闹钟 B。仅比较位 0 到 MASKSS-1。

注意： *仅当 RTC_CR 中的 ALRBE 复位时或在初始化模式下，才可以对该寄存器执行写操作。该寄存器受写保护。RTC 寄存器写保护一节中介绍了写访问的过程。*

23.6.20 RTC 备份寄存器 (RTC_BKPxR)

RTC backup registers

偏移地址：0x50 到 0x9C

上电复位值：0x0000 0000

系统复位：不受影响

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BKP[31:16]															
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
BKP[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	w	rw	rw

位 31:0 **BKP[31:0]**

应用可向/从这些寄存器写入/读取数据。
当 V_{DD} 关闭时，这些寄存器由 V_{BAT} 供电，因而系统复位时，这些寄存器不会复位，并且当器件在低功耗模式下工作时，寄存器的内容仍然有效。
发生入侵检测事件时该寄存器会被复位，并且只要 TAMPxF=1，该寄存器就一直保持复位。

23.6.21 RTC 寄存器映射

表 99. RTC 寄存器映射和复位值

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**RTC_TR**	Reserved									PM	HT [1:0]		HU[3:0]				Reserved	MNT[2:0]			MNU[3:0]				Reserved	ST[2:0]			SU[3:0]			
	Reset value										0	0	0	0	0	0	0		0	0	0	0	0	0	0		0	0	0	0	0	0	0
0x04	**RTC_DR**	Reserved								YT[3:0]				YU[3:0]				WDU[2:0]			MT	MU[3:0]				Reserved	DT [1:0]			DU[3:0]			
	Reset value																	0	0	0	1	0	0	0	1		0	0	0	0	0	0	1
0x08	**RTC_CR**	Reserved								COE	OSEL [1:0]		POL	COSEL	BKP	SUB1H	ADD1H	TSIE	WUTIE	ALRBIE	ALRAIE	TSE	WUTE	ALRBE	ALRAE	DCE	FMT	BYPSHAD	REFCKON	TSEDGE	WCKSEL [2:0]		
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	**RTC_ISR**	Reserved																	TAMP2F	TAMP1F	TSOVF	TSF	WUTF	ALRBF	ALRAF	INIT	INITF	RSF	INITS	SHPF	WUTWF	ALRBWF	ALRAWF
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	1	0	1
0x10	**RTC_PRER**	Reserved									PREDIV_A[6:0]							Reserved	PREDIV_S[14:0]														
	Reset value										1	1	1	1	1	1	1		0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0x14	**RTC_WUTR**	Reserved																	WUT[15:0]														
	Reset value																		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x18	**RTC_CALIBR**	Reserved																								DCS	Reserved	DC[4:0]					
	Reset value																									0		0	0	0	0	0	0
0x1C	**RTC_ALRMAR**	MSK4	WDSEL	DT [1:0]		DU[3:0]				MSK3	PM	HT [1:0]		HU[3:0]				MSK2	MNT[2:0]			MNU[3:0]				MSK1	ST[2:0]			SU[3:0]			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20	**RTC_ALRMBR**	MSK4	WDSEL	DT [1:0]		DU[3:0]				MSK3	PM	HT [1:0]		HU[3:0]				MSK2	MNT[2:0]			MNU[3:0]				MSK2	ST[2:0]			SU[3:0]			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x24	**RTC_WPR**	Reserved																								KEY[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x28	**RTC_SSR**	Reserved																	SS[15:0]														
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x2C	**RTC_SHIFTR**	ADD1S	Reserved																SUBFS[14:0]														
	Reset value	0																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x30	**RTC_TSTR**	Reserved									PM	HT[1:0]		HU[3:0]				Reserved	MNT[2:0]			MNU[3:0]				Reserved	ST[2:0]			SU[3:0]			
	Reset value										0	0	0	0	0	0	0		0	0	0	0	0	0	0		0	0	0	0	0	0	0
0x38	**RTC_TSSSR**	Reserved																	SS[15:0]														
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x3C	**RTC_ CALR**	Reserved																CALP	CALW8	CALW16	Reserved					CALM[8:0]							
	Reset value																	0	0	0						0	0	0	0	0	0	0	0

表 99. RTC 寄存器映射和复位值 （续）

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x40	**RTC_TAFCR**	Reserved													ALARMOUTTYPE	TSINSEL	TAMP1INSEL	TAMPPUDIS	TAMPPRCH[1:0]		TAMPFLT[1:0]		TAMPFREQ[2:0]			TAMPTS	Reserved		TAMP2TRG	TAMP2E	TAMPIE	TAMP1ETRG	TAMP1E
	Reset value														0	0	0	0	0	0	0	0	0	0	0	0			0	0	0	0	0
0x44	**RTC_ ALRMASSR**	Reserved				MASKSS[3:0]				Reserved									SS[14:0]														
	Reset value					0	0	0	0										0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x48	**RTC_ ALRMBSSR**	Reserved				MASKSS[3:0]				Reserved									SS[14:0]														
	Reset value					0	0	0	0										0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x50 to 0x9C	**RTC_BKP0R**	BKP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	to RTC_BKP**19**R	BKP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

有关寄存器边界地址的信息，请参见*第 52 页的表 2*。

24 控制器区域网络 (bxCAN)

除非特别说明，否则本节适用于整个 STM32F4xx 系列器件。

24.1 bxCAN 简介

基本扩展 CAN 外设又称 **bxCAN**，可与 CAN 网络进行交互。该外设支持 2.0A 和 B 版本的 CAN 协议，旨在以最少的 CPU 负载高效管理大量的传入消息，并可按需要的优先级实现消息发送。

在攸关安全性的应用中，CAN 控制器提供所有必要的硬件功能来支持 CAN 时间触发通信方案。

24.2 bxCAN 主要特性

- 支持 2.0 A 及 2.0 B Active 版本 CAN 协议
- 比特率高达 1 Mb/s
- 支持时间触发通信方案

发送

- 三个发送邮箱
- 可配置的发送优先级
- SOF 发送时间戳

接收

- 两个具有三级深度的接收 FIFO
- 可调整的筛选器组：
  - CAN1 和 CAN2 之间共享 28 个筛选器组
- 标识符列表功能
- 可配置的 FIFO 上溢
- SOF 接收时间戳

时间触发通信方案

- 禁止自动重发送模式
- 16 位自由运行定时器
- 在最后两个数据字节发送时间戳

管理

- 可屏蔽中断
- 在唯一地址空间通过软件实现高效的邮箱映射

双 CAN

- CAN1：主 bxCAN，用于管理 bxCAN 与 512 字节 SRAM 存储器之间的通信。
- CAN2：从 bxCAN，无法直接访问 SRAM 存储器。
- 两个 bxCAN 单元共享 512 字节 SRAM 存储器（请参见*图 224：双 CAN 框图*）。

24.3 bxCAN 一般说明

在如今的 CAN 应用中，网络节点数量日益增多，经常需要通过网关将数个网络连接在一起。系统中的消息数量（以及各个节点需要处理的消息）也有了显著增加。除应用程序消息外，还引入了网络管理和诊断消息。

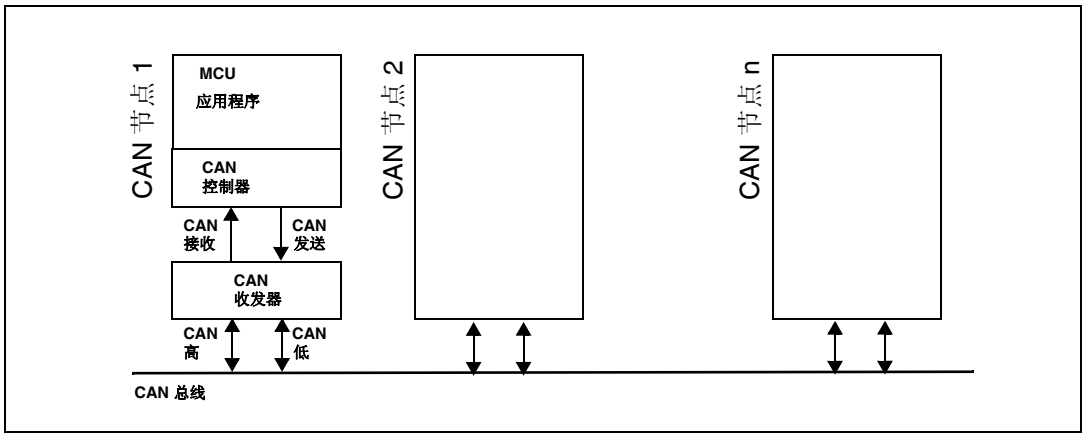
- 各种类型的消息需要一个增强的筛选机制进行处理。

此外，应用程序任务需要更多的 CPU 时间，因此必须减少因消息接收而对实时处理造成的限制。

- 接收 FIFO 方案使 CPU 能够长时间专门处理应用程序任务，而又不致丢失消息。

基于标准 CAN 驱动程序的标准 HLP（更高层协议）需要一个高效接口来与 CAN 控制器连接。

图 223. CAN 网络拓扑结构



24.3.1 CAN 2.0B 主动内核

bxCAN 模块可完全自主地处理 CAN 消息的发送和接收。标准标识符（11 位）和扩展标识符（29 位）完全由硬件支持。

24.3.2 控制、状态和配置寄存器

应用程序使用这些寄存器进行以下操作：

- 配置 CAN 参数，例如波特率
- 请求发送
- 处理接收
- 管理中断
- 获取诊断信息

24.3.3 发送邮箱

软件可通过三个发送邮箱设置消息。发送调度程序负责决定首先发送哪个邮箱的内容。

24.3.4 验收筛选器

bxCAN 提供了 28 个可调整/可配置的标识符筛选器组，用于选择软件所需的传入消息并丢弃其余消息。

bxCAN 提供了 28 个可调整/可配置的标识符筛选器组，用于选择软件所需的传入消息并丢弃其余消息。其他器件中，共有 14 个可调整/可配置的标识符筛选器组。

接收 FIFO

硬件使用两个接收 FIFO 来存储传入消息。每个 FIFO 中可以存储三条完整消息。FIFO 完全由硬件管理。

图 224. 双 CAN 框图

24.4 bxCAN 工作模式

bxCAN 有三种主要的工作模式：**初始化**、**正常**和**睡眠**。硬件复位后，bxCAN 进入睡眠模式以降低功耗，同时 CANTX 上的内部上拉电阻激活。软件将 CAN_MCR 寄存器的 INRQ 或 SLEEP 位置 1，以请求 bxCAN 进入**初始化**或**睡眠**模式。一旦进入该模式，bxCAN 即将 CAN_MSR 寄存器的 INAK 或 SLAK 位置 1，以确认该模式，同时禁止内部上拉电阻。如果 INAK 和 SLAK 均未置 1，则 bxCAN 将处于**正常**模式。进入**正常**模式之前，bxCAN 必须始终在 CAN 总线上实现**同步**。为了进行同步，bxCAN 将等待 CAN 总线空闲（即，已监测到 CANRX 上的 11 个隐性位）。

24.4.1 初始化模式

当硬件处于初始化模式时，可以进行软件初始化。为进入该模式，软件将 CAN_MCR 寄存器的 INRQ 位置 1，并等待硬件通过将 CAN_MCR 寄存器的 INAK 位置 1 来确认请求。

为退出初始化模式，软件将 INQR 位清零。一旦硬件将 INAK 位清零，bxCAN 即退出初始化模式。

在初始化模式下，所有从 CAN 总线传入和传出的消息都将停止，并且 CAN 总线输出 CANTX 的状态为隐性（高）。

进入初始化模式不会更改任何配置寄存器。

为初始化 CAN 控制器，软件必须设置位定时 (CAN_BTR) 和 CAN 选项 (CAN_MCR) 寄存器。

为初始化与 CAN 筛选器组相关的寄存器（模式、尺度、FIFO 分配、激活和筛选器值），软件必须将 FINIT 位 (CAN_FMR) 置 1。筛选器的初始化也可以在初始化模式之外进行。

注意： *FINIT=1 时，CAN 接收停用。*

筛选器值也可通过停用（CAN_FA1R 寄存器的）相关筛选器激活位来修改。

如果某个筛选器组未使用，建议将其保持未激活状态（将相应 FACT 位保持清零）。

24.4.2 正常模式

一旦初始化完成，软件必须向硬件请求进入正常模式，这样才能在 CAN 总线上进行同步，并开始接收和发送。

进入正常模式的请求可通过将 CAN_MCR 寄存器的 INRQ 位清零来发出。bxCAN 进入正常模式，并与 CAN 总线上的数据传输实现同步后，即可参与总线活动。执行这一步时，需要等待出现一个由 11 个连续隐性位（总线空闲状态）组成的序列。硬件通过将 CAN_MSR 寄存器的 INAK 位清零，来确认切换到正常模式。

筛选器值的初始化与初始化模式无关，但必须要在筛选器处于未激活状态（相应 FACTx 位清零）时进行。筛选器尺度和模式配置必须在进入正常模式之前完成。

24.4.3 睡眠模式（低功耗）

为降低能耗功耗，bxCAN 具有低功耗模式，称为睡眠模式。软件通过将 CAN_MCR 寄存器的 SLEEP 位置 1 而发出请求后，即可进入该模式。该模式下，bxCAN 时钟停止，但软件仍可访问 bxCAN 邮箱。

在 bxCAN 处于**睡眠**模式时，如果软件通过将 INRQ 位置 1 来请求进入**初始化**模式，则必须同时将 SLEEP 位清零。

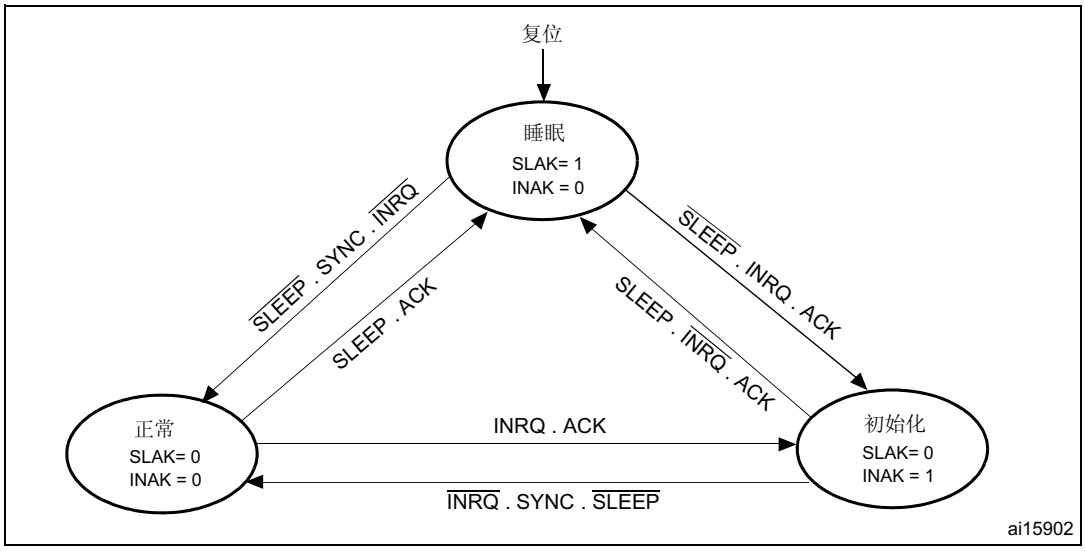
软件将 SLEEP 位清零或是检测到 CAN 总线活动时，bxCAN 即被唤醒（退出睡眠模式）。

检测到 CAN 总线活动后，如果 CAN_MCR 寄存器的 AWUM 位置 1，硬件将通过清零 SLEEP 位来自动执行唤醒序列。如果 AWUM 位清零，在发生唤醒中断时，软件必须将 SLEEP 位清零才能退出睡眠模式。

注意： *如果使能唤醒中断（CAN_IER 寄存器的 WKUIE 位置 1），即使 bxCAN 自动执行唤醒序列，一旦检测到 CAN 总线活动，也会发生唤醒中断。*

SLEEP 位清零后，一旦 bxCAN 与 CAN 总线同步，即会退出睡眠模式，请参见*图 225：bxCAN 工作模式*。一旦硬件将 SLAK 位清零，即会退出睡眠模式。

图 225. bxCAN 工作模式



1. ACK = 硬件通过将 CAN_MSR 寄存器的 INAK 或 SLAK 位置 1 来确认请求的等待状态
2. SYNC = bxCAN 等待 CAN 总线变为空闲（即在 CANRX 上监测到连续 11 个隐性位）的状态

24.5 测试模式

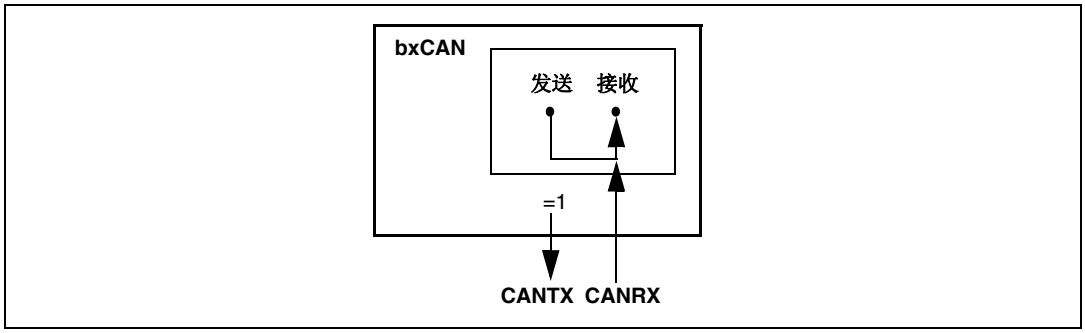
可以通过 CAN_BTR 寄存器中的 SILM 和 LBKM 位来选择测试模式。这些位必须在 bxCAN 处于初始化模式时进行配置。选择测试模式后，必须复位 CAN_MCR 寄存器中的 INRQ 位才能进入正常模式。

24.5.1 静默模式

可以通过将 CAN_BTR 寄存器的 SILM 位置 1，将 bxCAN 置于静默模式。

在静默模式下，bxCAN 可以接收有效数据帧和有效遥控帧，但仅在 CAN 总线上发送隐性位，并且无法启动发送。如果 bxCAN 必须发送一个显性位（ACK 位、溢出标志、活动错误标志），该位将在内部被改道发送，以便 CAN 内核可以监视该显性位，但 CAN 总线可以保持隐性状态。静默模式可用于分析 CAN 总线上的流量，同时又不会因发送显性位（确认位、错误帧）对其造成影响。

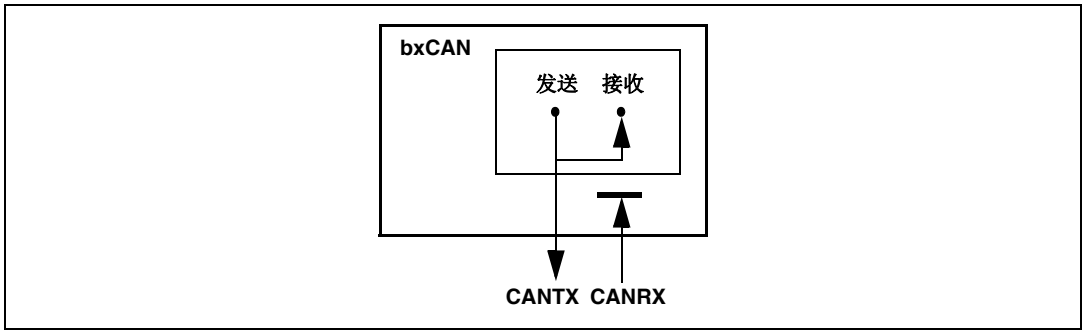
图 226. 静默模式下的 bxCAN



24.5.2 环回模式

可以通过将 CAN_BTR 寄存器的 LBKM 位置 1，将 bxCAN 置于环回模式。在环回模式下，bxCAN 将其自身发送的消息作为接收的消息来处理并存储（如果这些消息通过了验收筛选）在接收邮箱中。

图 227. 环回模式下的 bxCAN

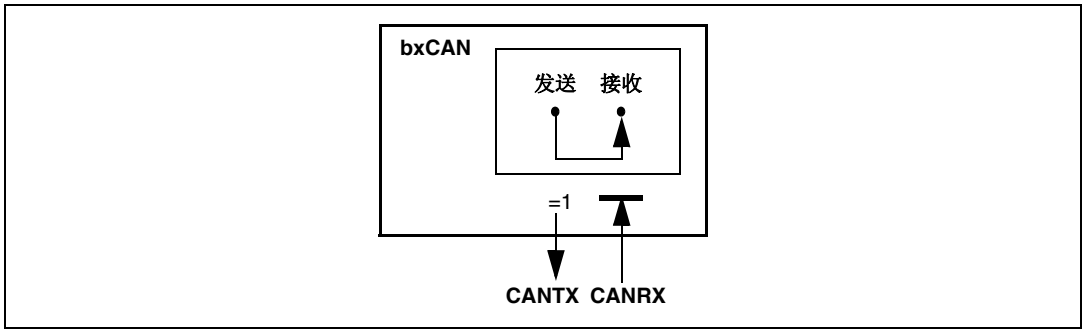


该模式为自检功能提供。为了不受外部事件的影响，CAN 内核在环回模式下将忽略确认错误（在数据/远程帧的确认时隙不对显性位采样）。在此模式下，bxCAN 将执行从发送输出到接收输入的内部反馈。bxCAN 将忽略 CANRX 输入引脚的实际值。从 CANTX 引脚可以监视发送的消息。

24.5.3 环回与静默组合模式

可以通过将 CAN_BTR 寄存器的 LBKM 和 SILM 位置 1，将环回模式和静默模式组合起来。该模式可用于“热自检”，也就是说，bxCAN 可以像在环回模式下一样进行检测，同时又不会影响与 CANTX 和 CANRX 引脚相连接的运行中的 CAN 系统。在此模式下，CANRX 引脚与 bxCAN 断开连接，CANTX 引脚则保持隐性。

图 228. 组合模式下的 bxCAN



24.6 调试模式

当微控制器进入调试模式（Cortex™-M4F 内核停止）时，bxCAN 可以继续正常工作，也可以停止工作，具体取决于如下条件：

- DBG 模块中用于 CAN1 的 DBG_CAN1_STOP 位或者用于 CAN2 的 DBG_CAN2_STOP 位。有关详细信息，请参见*第 33.16.2 节：对定时器、看门狗、bxCAN 和 I^2C 的调试支持*。
- CAN_MCR 中的 DBF 位。有关详细信息，请参见*第 24.9.2 节：CAN 控制和状态寄存器*。

24.7 bxCAN 功能说明

24.7.1 发送处理

为了发送消息，应用程序必须在请求发送前，通过将 CAN_TIxR 寄存器的相应 TXRQ 位置 1，选择一个**空**发送邮箱，并设置标识符、数据长度代码 (DLC) 和数据。一旦邮箱退出**空**状态，软件即不再具有对邮箱寄存器的写访问权限。TXRQ 位置 1 后，邮箱立即进入**挂起**状态，等待成为优先级最高的邮箱，请参见*发送优先级*。一旦邮箱拥有最高优先级，即被**安排**发送。CAN 总线变为空闲后，被安排好的邮箱中的消息即开始发送（进入**发送**状态）。邮箱一旦发送成功，即恢复**空**状态。硬件通过将 CAN_TSR 寄存器的 RQCP 和 TXOK 位置 1，来表示发送成功。

如果发送失败，失败原因将由 CAN_TSR 寄存器的 ALST 位（仲裁丢失）和/或 TERR 位（检测到发送错误）指示。

发送优先级

按标识符

当多个发送邮箱挂起时，发送顺序由邮箱中所存储消息的标识符来确定。根据 CAN 协议的仲裁，标识符值最低的消息具有最高的优先级。如果标识符值相等，则首先安排发送编号较小的邮箱。

按发送请求顺序

可以通过设置 CAN_MCR 寄存器中的 TXFP 位，将发送邮箱配置为发送 FIFO。在此模式下，优先级顺序按照发送请求顺序来确定。

该模式对分段发送非常有用。

中止

可以通过将 CAN_TSR 寄存器的 ABRQ 位置 1，来中止发送请求。在**挂起**或**已安排**状态下，邮箱立即中止。如果在邮箱处于**发送**状态时请求中止，则会出现两种结果。如果邮箱发送成功，将变为**空**状态，同时 CAN_TSR 寄存器的 TXOK 位置 1。如果发送失败，邮箱变为**已安排**状态，发送中止并变为**空**状态，同时 TXOK 位清零。在所有情况下，邮箱至少在当前发送结束时都会恢复**空**状态。

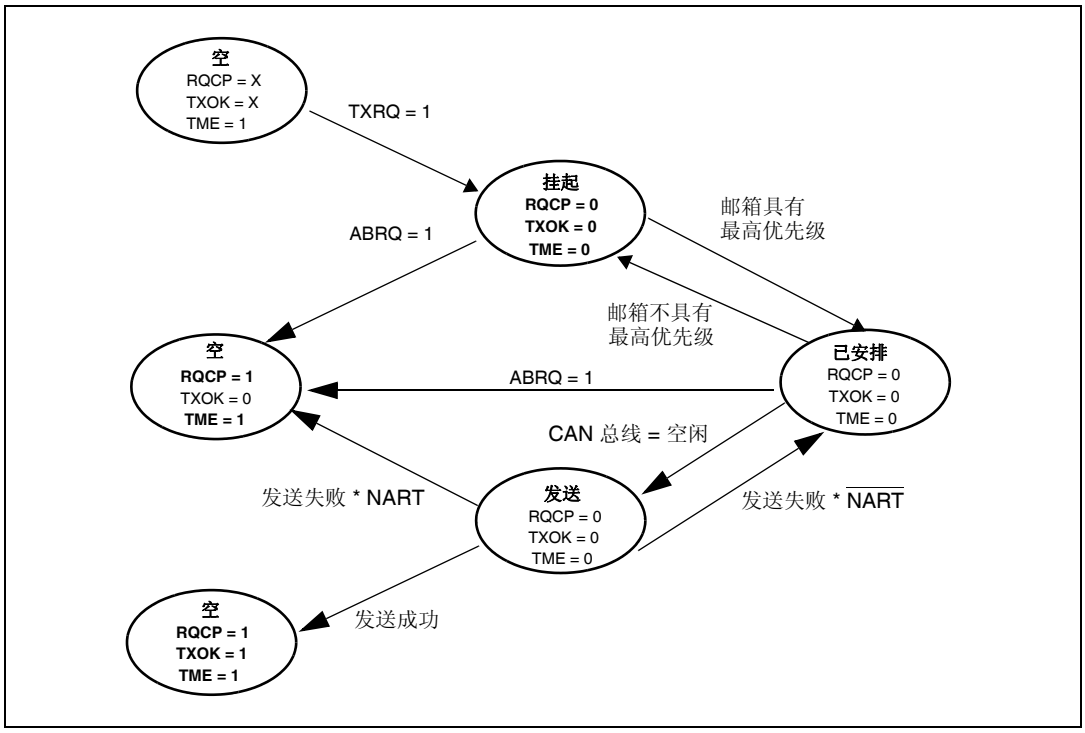
禁止自动重发送模式

该模式旨在满足 CAN 标准的时间触发通信方案的要求。要将硬件配置为此模式，必须将 CAN_MCR 寄存器的 NART 位置 1。

在此模式下，每个发送仅启动一次。如果第一次尝试失败，由于仲裁丢失或错误，硬件将不会自动重新启动消息发送。

第一次发送尝试结束时，硬件将认为请求已完成，并将 CAN_TSR 寄存器的 RQCP 位置 1。发送结果由 CAN_TSR 寄存器的 TXOK、ALST 和 TERR 位来指示。

图 229. 发送邮箱状态



24.7.2 时间触发通信模式

在此模式下，CAN 硬件的内部计数器激活，用于为接收和发送邮箱生成时间戳值，这些值分别存储在 CAN_RDTxR/CAN_TDTxR 寄存器中。内部计数器在每个 CAN 位时间递增（请参见*第 24.7.7 节：位时序*）。在接收和发送时，都会在帧起始位的采样点捕获内部计数器。

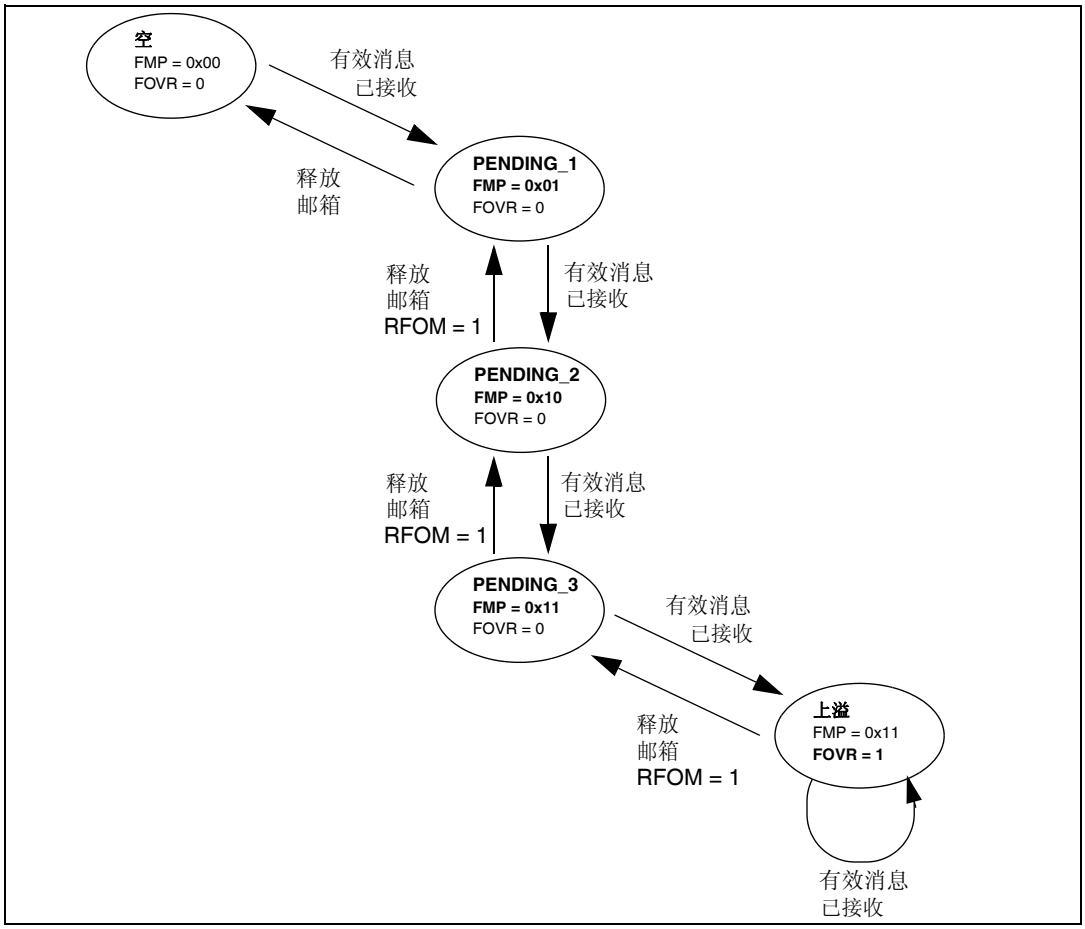
24.7.3 接收处理

为了接收 CAN 消息，提供了构成 FIFO 的三个邮箱。为了节约 CPU 负载，简化软件并保证数据一致性，FIFO 完全由硬件进行管理。应用程序通过 FIFO 输出邮箱访问 FIFO 中所存储的消息。

有效消息

当消息依据 CAN 协议正确接收（直到 EOF 字段的倒数第二位都没有发送错误）并且成功通过标识符筛选后，该消息将视为有效，请参见*第 24.7.4 节：标识符筛选*。

图 230. 接收 FIFO 状态



FIFO 管理

FIFO 开始时处于**空**状态，在接收的第一条有效消息存储在其中后，变为 **Pending_1** 状态。硬件通过将 CAN_RFR 寄存器的 FMP[1:0] 位置为 01b 来指示该事件。消息将在 FIFO 输出邮箱中供读取。软件将读取邮箱内容，并通过将 CAN_RFR 寄存器的 RFOM 位置 1，来将邮箱释放。FIFO 随即恢复**空**状态。如果同时接收到新的有效消息，FIFO 将保持 **Pending_1** 状态，新消息将在输出邮箱中供读取。

如果应用程序未释放邮箱，下一条有效消息将存储在 FIFO 中，使其进入 **Pending_2** 状态 (FMP[1:0] = 10b)。下一条有效消息会重复该存储过程，同时将 FIFO 变为 **Pending_3** 状态 (FMP[1:0] = 11b)。此时，软件必须通过将 RFOM 位置 1 来释放输出邮箱，从而留出一个空邮箱来存储下一条有效消息。否则，下一次接收到有效消息时，将导致消息丢失。

另请参见*第 24.7.5 节：消息存储*。

上溢

一旦 FIFO 处于 **Pending_3** 状态（即三个邮箱均已满），则下一次接收到有效消息时，将导致**上溢**并丢失一条消息。硬件通过将 CAN_RFR 寄存器的 FOVR 位置 1 来指示上溢状况。丢失的消息取决于 FIFO 的配置：

- 如果禁止 FIFO 锁定功能（CAN_MCR 寄存器的 RFLM 位清零），则新传入的消息将覆盖 FIFO 中存储的最后一条消息。在这种情况下，应用程序将始终能访问到最新的消息。
- 如果使能 FIFO 锁定功能（CAN_MCR 寄存器的 RFLM 位置 1），则将丢弃最新的消息，软件将提供 FIFO 中最早的三条消息。

与接收相关的中断

消息存储到 FIFO 中后，FMP[1:0] 位即会更新，如果 CAN_IER 寄存器的 FMPIE 位置 1，将产生中断请求。

FIFO 存满消息（即存储了第三条消息）后，CAN_RFR 寄存器的 FULL 位置 1，如果 CAN_IER 寄存器的 FFIE 位置 1，将产生中断。

出现上溢时，FOVR 位将置 1，如果 CAN_IER 寄存器的 FOVIE 位置 1，将产生中断。

24.7.4 标识符筛选

在 CAN 协议中，消息的标识符与节点地址无关，但与消息内容有关。因此，发送器将消息广播给所有接收器。在接收到消息时，接收器节点会根据标识符的值来确定软件是否需要该消息。如果需要，该消息将复制到 SRAM 中。如果不需要，则必须在无软件干预的情况下丢弃该消息。

为了满足这一要求，bxCAN 控制器为应用程序提供了 28 个可配置且可调整的筛选器组 (27-0)。在其他器件中，bxCAN 控制器为应用程序提供了 14 个可配置且可调整的筛选器组 (13-0)，以便仅接收软件需要的消息。此硬件筛选功能可以节省软件筛选所需的 CPU 资源。每个筛选器组 x 均包含两个 32 位寄存器，分别是 CAN_FxR0 和 CAN_FxR1。

可调整的宽度

为了根据应用程序的需求来优化和调整筛选器，每个筛选器组可分别进行伸缩调整。根据筛选器尺度不同，一个筛选器组可以：

- 为 STDID[10:0]、EXTID[17:0]、IDE 和 RTR 位提供一个 32 位筛选器。
- 为 STDID[10:0]、RTR、IDE 和 EXTID[17:15] 位提供两个 16 位筛选器。

请参见*图 231*。

此外，筛选器还可配置为掩码模式或标识符列表模式。

掩码模式

在**掩码**模式下，标识符寄存器与掩码寄存器关联，用以指示标识符的哪些位“必须匹配”，哪些位“无关”。

标识符列表模式

在**标识符列表**模式下，掩码寄存器用作标识符寄存器。这时，不会定义一个标识符和一个掩码，而是指定两个标识符，从而使单个标识符的数量加倍。传入标识符的所有位都必须与筛选器寄存器中指定的位匹配。

筛选器组尺度和模式配置

筛选器组通过相应的 CAN_FMR 寄存器进行配置。为了配置筛选器组，必须通过将 CAN_FAR 寄存器的 FACT 位清零而将其停用。筛选器尺度通过 CAN_FS1R 寄存器的相应 FSCx 位进行配置，请参见*图 231*。相应掩码/标识符寄存器的**标识符列表**或**标识符掩码**模式通过 CAN_FMR 寄存器的 FBMx 位进行配置。

要筛选一组标识符，应将掩码/标识符寄存器配置为掩码模式。

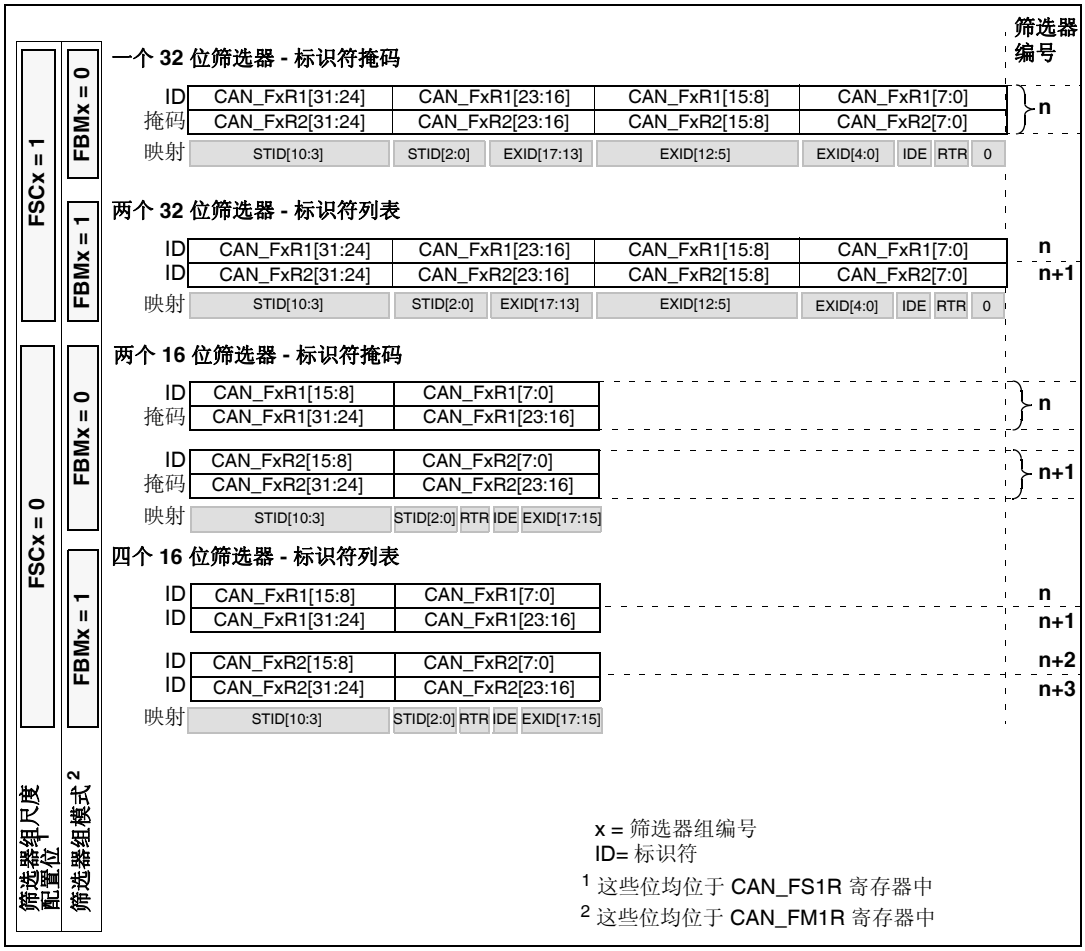
要选择单个标识符，应将掩码/标识符寄存器配置为标识符列表模式。

未由应用程序使用的筛选器应保持停用。

筛选器组中的每个筛选器将按从 0 到最大值的顺序进行编号（称为*筛选器编号*），具体取决于每个筛选器组的模式和尺度。

有关筛选器配置，请参见*图 231*。

图 231. 筛选器组尺度配置 - 寄存器构成



筛选器匹配索引

消息接收到 FIFO 中后，即可供应用程序使用。应用程序数据通常会复制到 SRAM 中的位置。为了将数据复制到正确的位置，应用程序必须通过标识符来识别数据。为了避免这种情况，方便访问 SRAM 位置，CAN 控制器提供了一个筛选器匹配索引。

该索引根据筛选器优先级规则与消息一同存储在邮箱中。因此，每条收到的消息都有相关联的筛选器匹配索引。

筛选器匹配索引的使用方法有两种：

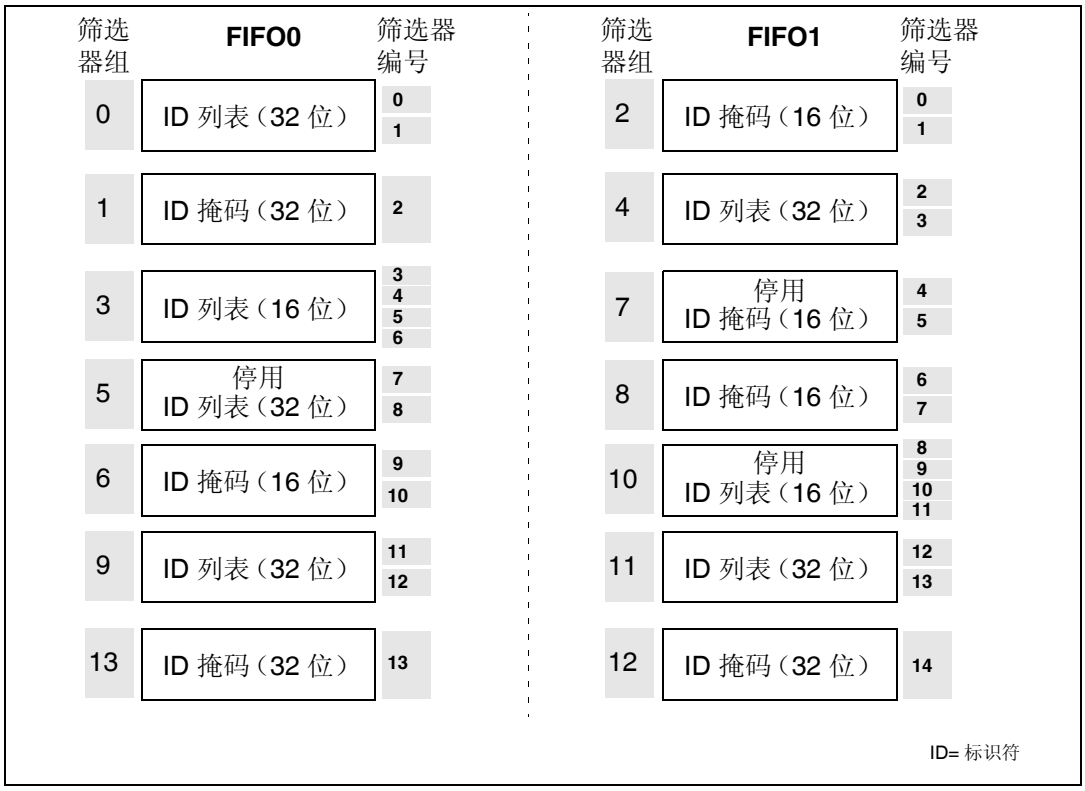
- 将筛选器匹配索引与预期值列表进行比较。
- 将筛选器匹配索引用作阵列索引，以访问数据目标位置。

对于非屏蔽筛选器，软件不再需要比较标识符。

如果筛选器有屏蔽，软件则只需比较屏蔽位。

筛选器编号的索引值与筛选器组的激活状态无关。此外，还将使用两个独立的编号方案，每个 FIFO 各一个。有关示例，请参见*图 232*。

图 232. 筛选器编号示例

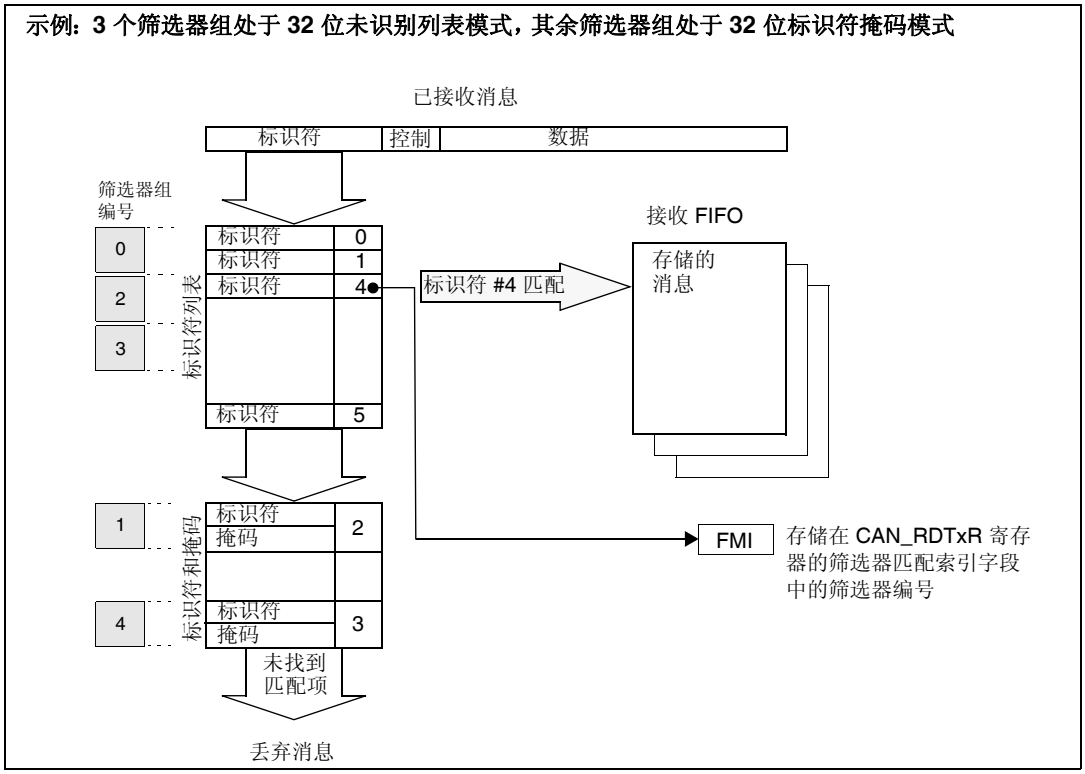


筛选器优先级规则

根据筛选器组合，可能会出现一个标识符成功通过数个筛选器的情况。这种情况下，将根据以下优先级规则选择接收邮箱中存储的筛选器匹配值：

- 32 位筛选器优先于 16 位筛选器。
- 对于尺度相等的筛选器，标识符列表模式优先于标识符掩码模式。
- 对于尺度和模式均相等的筛选器，则按筛选器编号确定优先级（编号越低，优先级越高）。

图 233. 筛选机制 - 示例



以上示例说明了 bxCAN 的筛选原则。接收到消息后，首先将标识符与标识符列表模式中配置的筛选器进行比较。如果匹配，消息将存储在相应 FIFO 中，匹配筛选器的索引则存储在筛选器匹配索引中。如本例所示，标识符与标识符 #2 匹配，因此消息内容和 FMI 2 存储到该 FIFO 中。

如果不匹配，则将传入标识符与掩码模式中配置的筛选器进行比较。

如果标识符与筛选器中配置的任何标识符均不匹配，硬件会在不干扰软件的情况下丢弃该消息。

24.7.5 消息存储

CAN 消息软件与硬件之间的接口通过邮箱实现。邮箱中包含所有与消息相关的信息：标识符、数据、控制、状态和时间戳信息。

发送邮箱

软件在空发送邮箱中设置将要发送的消息。发送状态由硬件在 CAN_TSR 寄存器中进行指示。

表 100. 发送邮箱映射

与发送邮箱基址之间的偏移	寄存器名称
0	CAN_TIxR
4	CAN_TDTxR
8	CAN_TDLxR
12	CAN_TDHxR

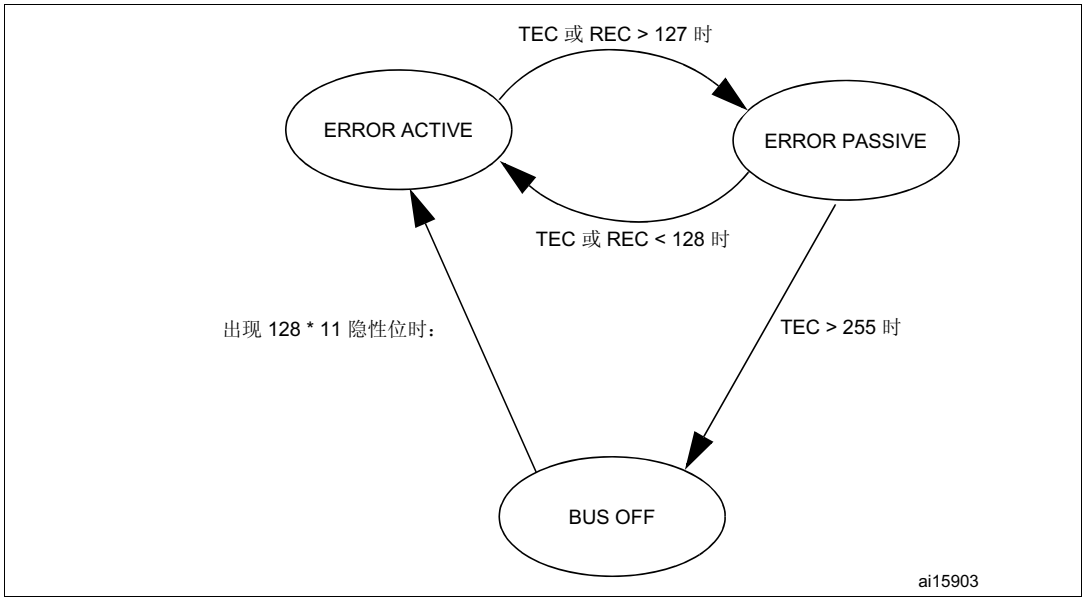
接收邮箱

消息在接收到后，将在 FIFO 输出邮箱中供软件使用。一旦软件对消息进行了处理（例如读取），则必须通过 CAN_RFR 寄存器的 RFOM 位释放 FIFO 输出邮箱，以接收下一条传入消息。筛选器匹配索引存储在 CAN_RDTxR 寄存器的 MFMI 字段中。16 位时间戳值则存储在 CAN_RDTxR 的 TIME[15:0] 字段中。

表 101. 接收邮箱映射

与接收邮箱基址之间的偏移（字节）	寄存器名称
0	CAN_RIxR
4	CAN_RDTxR
8	CAN_RDLxR
12	CAN_RDHxR

图 234. CAN 错误状态图



24.7.6 错误管理

如 CAN 协议所述，错误管理完全由硬件通过发送错误计数器（CAN_ESR 寄存器中的 TEC 值）和接收错误计数器（CAN_ESR 寄存器中的 REC 值）来处理，这两个计数器根据错误状况进行递增或递减。有关 TEC 和 REC 管理的详细信息，请参见 CAN 标准。

两者均可由软件读取，用以确定网络的稳定性。此外，CAN 硬件还将在 CAN_ESR 寄存器中提供当前错误状态的详细信息。通过 CAN_IER 寄存器（ERRIE 位等），软件可以非常灵活地配置在检测到错误时生成的中断。

总线关闭恢复

当 TEC 大于 255 时，达到总线关闭状态，该状态由 CAN_ESR 寄存器的 BOFF 位指示。在总线关闭状态下，bxCAN 不能再发送和接收消息。

bxCAN 可以自动或者应软件请求而从总线关闭状态中恢复（恢复错误主动状态），具体取决于 CAN_MCR 寄存器的 ABOM 位。但在两种情况下，bxCAN 都必须至少等待 CAN 标准中指定的恢复序列完成（在 CANRX 上监测到 128 次 11 个连续隐性位）。

如果 ABOM 位置 1，bxCAN 将在进入总线关闭状态后自动启动恢复序列。

如果 ABOM 位清零，则软件必须请求 bxCAN 先进入再退出初始化模式，从而启动恢复序列。

注意： *在初始化模式下，bxCAN 不会监视 CANRX 信号，因此无法完成恢复序列。**要进行恢复，bxCAN 必须处于正常模式**。*

24.7.7 位时序

位时序逻辑将监视串行总线，执行采样并调整采样点，在调整采样点时，需要在起始位边沿进行同步并后续的边沿进行再同步。

通过将标称位时间划分为以下三段，即可解释其工作过程：

- **同步段 (SYNC_SEG)**：位变化应该在此时间段内发生。它只有一个时间片的固定长度 (1 x t_{CAN})。
- **位段 1 (BS1)**：定义采样点的位置。它包括 CAN 标准的 PROP_SEG 和 PHASE_SEG1。其持续长度可以在 1 到 16 个时间片之间调整，但也可以自动加长，以补偿不同网络节点的频率差异所导致的正相位漂移。
- **位段 2 (BS2)**：定义发送点的位置。它代表 CAN 标准的 PHASE_SEG2。其持续长度可以在 1 到 8 个时间片之间调整，但也可以自动缩短，以补偿负相位漂移。

再同步跳转宽度 (SJW) 定义位段加长或缩短的上限。它可以在 1 到 4 个时间片之间调整。

有效边沿是指一个位时间内总线电平从显性到隐性的第一次转换（前提是控制器本身不发送隐性位）。

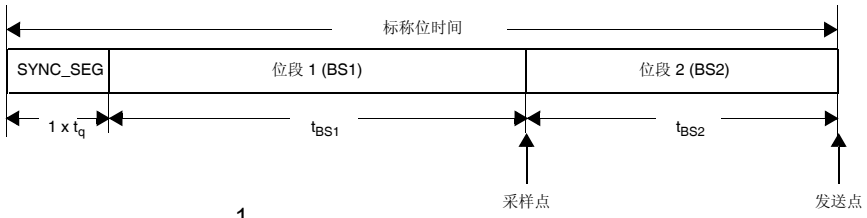
如果在 BS1 而不是 SYNC_SEG 中检测到有效边沿，则 BS1 会延长最多 SJW，以便延迟采样点。

相反地，如果在 BS2 而不是 SYNC_SEG 中检测到有效边沿，则 BS2 会缩短最多 SJW，以便提前发送点。

为了避免编程错误，位时序寄存器 (CAN_BTR) 只能在器件处于待机模式时进行配置。

注意： *有关 CAN 位时序和再同步机制的详细说明，请参见 ISO 11898 标准。*

图 235. 位时序



$$BaudRate = \frac{1}{NominalBitTime}$$

$$NominalBitTime = 1 \times t_q + t_{BS1} + t_{BS2}$$

其中：

$t_{BS1} = t_q$ x (TS1[3:0] + 1)，

$t_{BS2} = t_q$ x (TS2[2:0] + 1)，

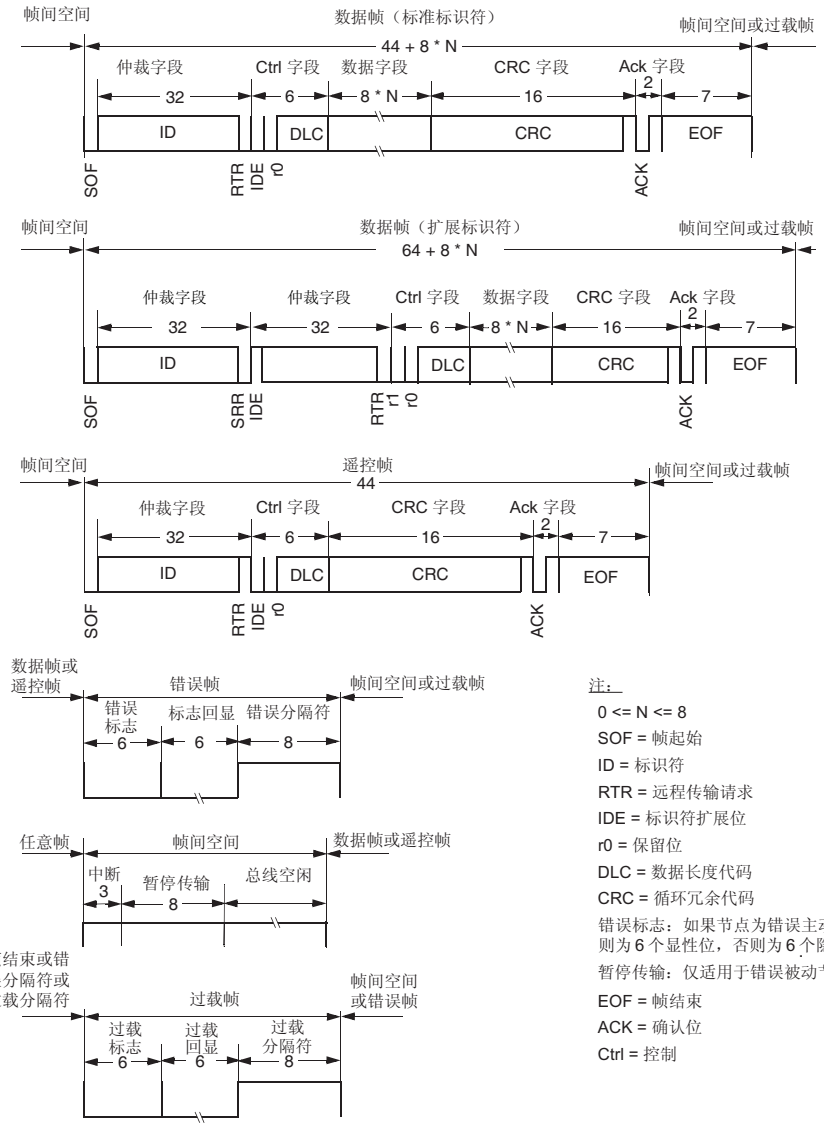
t_q = (BRP[9:0] + 1) x t_{PCLK}

其中 t_q 为时间片

t_{PCLK} = APB 时钟的时间周期，

BRP[9:0]、TS1[3:0] 和 TS2[2:0] 在 CAN_BTR 寄存器中定义。

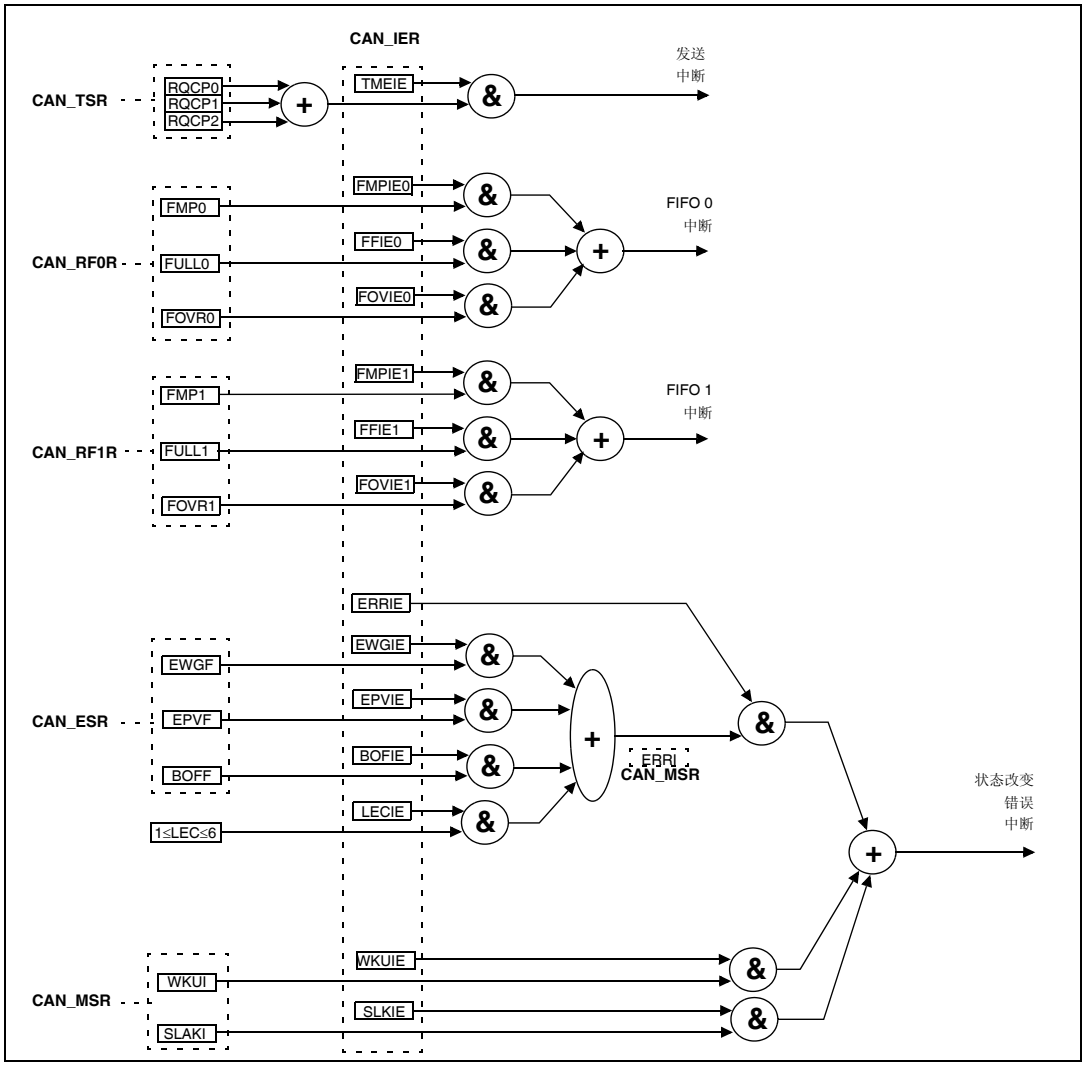
图 236. CAN 帧


帧间空间
数据帧（标准标识符）
帧间空间或过载帧
44 + 8 * N
仲裁字段
Ctrl 字段
数据字段
CRC 字段
Ack 字段
32
6
8 * N
16
2
7
ID
DLC
CRC
EOF
SOF
RTR
IDE
r0
ACK
帧间空间
数据帧（扩展标识符）
帧间空间或过载帧
64 + 8 * N
仲裁字段
仲裁字段
Ctrl 字段
数据字段
CRC 字段
Ack 字段
32
32
6
8 * N
16
2
7
ID
DLC
CRC
EOF
SOF
SRR
IDE
RTR
r1
r0
ACK
帧间空间
遥控帧
帧间空间或过载帧
44
仲裁字段
Ctrl 字段
CRC 字段
Ack 字段
32
6
16
2
7
ID
DLC
CRC
EOF
SOF
RTR
IDE
r0
ACK
数据帧或
遥控帧
错误帧
帧间空间或过载帧
错误
标志
标志回显
错误分隔符
6
6
8
任意帧
帧间空间
数据帧或遥控帧
中断
暂停传输
总线空闲
3
8
帧结束或错
误分隔符或
过载分隔符
过载帧
帧间空间
或错误帧
过载
标志
过载
回显
过载
分隔符
6
6
8
注：
0 <= N <= 8
SOF = 帧起始
ID = 标识符
RTR = 远程传输请求
IDE = 标识符扩展位
r0 = 保留位
DLC = 数据长度代码
CRC = 循环冗余代码
错误标志：如果节点为错误主动节点，
则为 6 个显性位，否则为 6 个隐性位。
暂停传输：仅适用于错误被动节点。
EOF = 帧结束
ACK = 确认位
Ctrl = 控制
ai15154

24.8 bxCAN 中断

bxCAN 共有四个专用的中断向量。每个中断源均可通过 CAN 中断使能寄存器 (CAN_IER) 来单独地使能或禁止。

图 237. 事件标志与中断产生



- **发送中断**可由以下事件产生：
  - 发送邮箱 0 变为空，CAN_TSR 寄存器的 RQCP0 位置 1。
  - 发送邮箱 1 变为空，CAN_TSR 寄存器的 RQCP1 位置 1。
  - 发送邮箱 2 变为空，CAN_TSR 寄存器的 RQCP2 位置 1。
- **FIFO 0 中断**可由以下事件产生：
  - 接收到新消息，CAN_RF0R 寄存器的 FMP0 位不是“00”。
  - FIFO0 满，CAN_RF0R 寄存器的 FULL0 位置 1。
  - FIFO0 上溢，CAN_RF0R 寄存器的 FOVR0 位置 1。

- **FIFO 1 中断**可由以下事件产生：
  - 接收到新消息，CAN_RF1R 寄存器的 FMP1 位不是“00”。
  - FIFO1 满，CAN_RF1R 寄存器的 FULL1 位置 1。
  - FIFO1 上溢，CAN_RF1R 寄存器的 FOVR1 位置 1。
- **错误和状态改变中断**可由以下事件产生：
  - 错误状况，有关错误状况的更多详细信息，请参见 CAN 错误状态寄存器 (CAN_ESR)。
  - 唤醒状况，CAN Rx 信号上监测到 SOF。
  - 进入睡眠模式。

24.9 CAN 寄存器

外设寄存器必须按字（32 位）进行访问。

24.9.1 寄存器访问保护

错误访问某些配置寄存器可能会导致硬件暂时性地干扰整个 CAN 网络。因此，只有在 CAN 硬件处于初始化模式时，才可以通过软件修改 CAN_BTR 寄存器。

尽管传输错误的数据不会引发 CAN 网络级别的故障，但可能会对应用程序造成严重干扰。发送邮箱只能在处于空状态时通过软件进行修改，请参见*图 229：发送邮箱状态*。

可以通过停用相应筛选器组或将 FINIT 位置 1 来修改筛选器值。此外，只有在 CAN_FMR 寄存器的筛选器初始化模式位置 1 (FINIT=1) 时，才能对 CAN_FMxR、CAN_FSxR 和 CAN_FFAR 寄存器中的筛选器配置（大小、模式和 FIFO 分配）进行修改。

24.9.2 CAN 控制和状态寄存器

有关寄存器说明中使用的缩写，请参见*第 1.1 节*。

CAN 主控制寄存器 (CAN_MCR)

CAN master control register

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
Reserved															DBF
															rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESET	Reserved							TTCM	ABOM	AWUM	NART	RFLM	TXFP	SLEEP	INRQ
rs								rw	rw	rw	rw	rw	rw	rw	rw

位 31:17 保留，必须保持复位值。

位 16 **DBF**：调试冻结 (Debug freeze)

0：调试期间 CAN 处于工作状态。

1：调试期间 CAN 处于接收/发送冻结状态。接收 FIFO 仍可正常访问/控制。

位 15 **RESET**：bxCAN 软件主复位 (bxCAN software master reset)

0：正常工作。

1：强制 bxCAN 进行主复位 -> 复位后激活睡眠模式（FMP 位和 CAN_MCR 寄存器初始化为复位值）。此位自动复位为 0。

位 14:8 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：使能时间触发通信模式。

注意：有关时间触发通信模式的更多信息，请参见第 24.7.2 节：时间触发通信模式。

位 6 **ABOM**：自动的总线关闭管理 (Automatic bus-off management)

此位控制 CAN 硬件在退出总线关闭状态时的行为。

0：在软件发出请求后，一旦监测到 128 次连续 11 个隐性位，并且软件将 CAN_MCR 寄存器的 INRQ 位先置 1 再清零，即退出总线关闭状态。

1：一旦监测到 128 次连续 11 个隐性位，即通过硬件自动退出总线关闭状态。

有关总线关闭状态的详细信息，请参见*第 24.7.6 节：错误管理*。

位 5 **AWUM**：自动唤醒模式 (Automatic wakeup mode)

此位控制 CAN 硬件在睡眠模式下接收到消息时的行为。

0：在软件通过将 CAN_MCR 寄存器的 SLEEP 位清零发出请求后，退出睡眠模式。

1：一旦监测到 CAN 消息，即通过硬件自动退出睡眠模式。

CAN_MCR 寄存器的 SLEEP 位和 CAN_MCR 寄存器的 SLAK 位由硬件清零。

位 4 **NART**：禁止自动重发送 (No automatic retransmission)

0：CAN 硬件将自动重发送消息，直到根据 CAN 标准消息发送成功。

1：无论发送结果如何（成功、错误或仲裁丢失），消息均只发送一次。

位 3 **RFLM**：接收 FIFO 锁定模式 (Receive FIFO locked mode)

0：接收 FIFO 上溢后不锁定。接收 FIFO 装满后，下一条传入消息将覆盖前一条消息。

1：接收 FIFO 上溢后锁定。接收 FIFO 装满后，下一条传入消息将被丢弃。

位 2 **TXFP**：发送 FIFO 优先级 (Transmit FIFO priority)

此位用于控制在几个邮箱同时挂起时的发送顺序。

0：优先级由消息标识符确定

1：优先级由请求顺序（时间顺序）确定

位 1 **SLEEP**：睡眠模式请求 (Sleep mode request)

此位由软件置 1，用于请求 CAN 硬件进入睡眠模式。一旦当前 CAN 活动（发送或接收 CAN 帧）结束，即进入睡眠模式。

此位由软件清零时，将退出睡眠模式。

当 AWUM 位置 1 以及在 CAN RX 信号上检测到 SOF 位时，硬件即将此位清零。

复位后，此位将置 1 - CAN 启动睡眠模式。

位 0 **INRQ**：初始化请求 (Initialization request)

软件通过将此位清零，来将硬件切换到正常模式。一旦在 Rx 信号上监测到连续 11 个隐性位，CAN 硬件即完成同步并准备进行发送和接收。硬件通过将 CAN_MSR 寄存器的 INAK 位清零来指示此事件。

软件通过将此位置 1 来请求 CAN 硬件进入初始化模式。一旦软件将 INRQ 位置 1，CAN 硬件将等待当前 CAN 活动（发送或接收）结束，然后进入初始化模式。硬件通过将 CAN_MSR 寄存器的 INAK 位置 1 来指示此事件。

CAN 主状态寄存器 (CAN_MSR)

CAN master status register

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
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved.				RX	SAMP	RXM	TXM	Reserved			SLAKI	WKUI	ERRI	SLAK	INAK
				r	r	r	r				rc_w1	rc_w1	rc_w1	r	r

位 31:12 保留，必须保持复位值。

位 11 **RX**：CAN Rx 信号 (CAN Rx signal)
监视 **CAN_RX** 引脚的实际值。

位 10 **SAMP**：上一个采样点 (Last sample point)
上一个采样点的 RX 值（当前接收的位值）。

位 9 **RXM**：接收模式 (Receive mode)
CAN 硬件当前为接收器。

位 8 **TXM**：发送模式 (Transmit mode)
CAN 硬件当前为发送器。

位 7:5 保留，必须保持复位值。

位 4 **SLAKI**：睡眠确认中断 (Sleep acknowledge interrupt)
当 SLKIE=1 时，硬件将此位置 1，以指示 bxCAN 已经进入睡眠模式。如果 CAN_IER 寄存器的 SLKIE 位置 1，则当此位置 1 后将产生状态改变中断。
SLAK 清零后，此位由软件或硬件清零。
注意：SLKIE=0 时，无法对 SLAKI 进行轮询。在这种情况下，可以轮询 SLAK 位。

位 3 **WKUI**：唤醒中断 (Wakeup interrupt)
此位由硬件置 1，用于指示在 CAN 硬件处于睡眠模式期间检测到一个 SOF 位。如果 CAN_IER 寄存器的 WKUIE 位置 1，则当此位置 1 后将产生状态改变中断。
此位由软件清零。

位 2 **ERRI**：错误中断 (Error interrupt)
如果在检测到错误时 CAN_ESR 的一个位置 1，并且使能 CAN_IER 中的相应中断，则硬件将此位置 1。如果 CAN_IER 寄存器的 ERRIE 位置 1，则当此位置 1 后将产生状态改变中断。
此位由软件清零。

位 1 **SLAK**：睡眠确认 (Sleep acknowledge)
此位由硬件置 1，用于向软件指示 CAN 硬件此时处于睡眠模式。此位可确认软件的睡眠模式请求（CAN_MCR 寄存器的 SLEEP 位置 1）。
CAN 硬件退出睡眠模式（以在 CAN 总线上进行同步）时，此位由硬件清零。要进行同步，硬件必须在 CAN RX 信号上监测到由 11 个连续隐性位组成的一个序列。
注意：CAN_MCR 寄存器的 SLEEP 位清零时，将触发退出睡眠模式程序。有关 SLEEP 位清零的详细信息，请参阅 CAN_MCR 寄存器 AWUM 位的说明。

位 0 **INAK**：初始化确认 (Initialization acknowledge)
此位由硬件置 1，用于向软件指示 CAN 硬件此时处于初始化模式。此位可确认软件的初始化请求（CAN_MCR 寄存器的 INRQ 位置 1）。
CAN 硬件退出初始化模式（以在 CAN 总线上进行同步）时，此位由硬件清零。要进行同步，硬件必须在 CAN RX 信号上监测到由 11 个连续隐性位组成的一个序列。

CAN 发送状态寄存器 (CAN_TSR)

CAN transmit status register

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	Reserved			TERR2	ALST2	TXOK2	RQCP2
r	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	Reserved Res.			TERR1	ALST1	TXOK1	RQCP1	ABRQ0	Reserved			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)
当多个邮箱挂起等待发送且邮箱 2 优先级最低时，此位由硬件置 1。

位 30 **LOW1**：邮箱 1 最低优先级标志 (Lowest priority flag for mailbox 1)
当多个邮箱挂起等待发送且邮箱 1 优先级最低时，此位由硬件置 1。

位 29 **LOW0**：邮箱 0 最低优先级标志 (Lowest priority flag for mailbox 0)
当多个邮箱挂起等待发送且邮箱 0 优先级最低时，此位由硬件置 1。
注意：当只有一个邮箱挂起时，LOW[2:0] 位置为零。

位 28 **TME2**：发送邮箱 2 空 (Transmit mailbox 2 empty)
当邮箱 2 没有挂起的发送请求时，此位由硬件置 1。

位 27 **TME1**：发送邮箱 1 空 (Transmit mailbox 1 empty)
当邮箱 1 没有挂起的发送请求时，此位由硬件置 1。

位 26 **TME0**：发送邮箱 0 空 (Transmit mailbox 0 empty)
当邮箱 0 没有挂起的发送请求时，此位由硬件置 1。

位 25:24 **CODE[1:0]**：邮箱代码 (Mailbox code)
如果至少一个发送邮箱空闲，代码值等于下一个空闲发送邮箱的编号。
如果所有发送邮箱均挂起，则代码值等于优先级最低的发送邮箱的编号。

位 23 **ABRQ2**：邮箱 2 中止请求 (Abort request for mailbox 2)
由软件置 1，用于中止相应邮箱的发送请求。
邮箱变为空后，此位由硬件清零。
邮箱未挂起等待发送时，将此位置 1 没有任何作用。

位 22:20 保留，必须保持复位值。

位 19 **TERR2**：邮箱 2 发送错误 (Transmission error of mailbox 2)
如果上一次发送因错误而失败，此位将置 1。

位 18 **ALST2**：邮箱 2 仲裁丢失 (Arbitration lost for mailbox 2)
如果上一次发送因仲裁丢失而失败，此位将置 1。

位 17 **TXOK2**：邮箱 2 发送成功 (Transmission OK of mailbox 2)
每次发送尝试后，硬件都将更新此位。
0：上一次发送失败
1：上一次发送成功
当邮箱 2 的发送请求成功完成时，此位由硬件置 1。请参见*图 229*。

位 16 **RQCP2**：邮箱 2 请求完成 (Request completed mailbox 2)
最后一个请求（发送或中止）执行完毕时，由硬件置 1。
由软件通过写入“1”清零，或是在发生发送请求（CAN_TMID2R 寄存器中的 TXRQ2 位置 1）时由硬件清零。
如果将此位清零，邮箱 2 的所有状态位（TXOK2、ALST2 和 TERR2）都将清零。

位 15 **ABRQ1**：邮箱 1 中止请求 (Abort request for mailbox 1)
由软件置 1，用于中止相应邮箱的发送请求。
邮箱变为空后，此位由硬件清零。
邮箱未挂起等待发送时，将此位置 1 没有任何作用。

位 14:12 保留，必须保持复位值。

位 11 **TERR1**：邮箱 1 发送错误 (Transmission error of mailbox 1)
如果上一次发送因错误而失败，此位将置 1。

位 10 **ALST1**：邮箱 1 仲裁丢失 (Arbitration lost for mailbox 1)
如果上一次发送因仲裁丢失而失败，此位将置 1。

位 9 **TXOK1**：邮箱 1 发送成功 (Transmission OK of mailbox 1)
每次发送尝试后，硬件都将更新此位。
0：上一次发送失败
1：上一次发送成功
当邮箱 1 的发送请求成功完成时，此位由硬件置 1。请参见*图 229*。

位 8 **RQCP1**：邮箱 1 请求完成 (Request completed mailbox 1)
最后一个请求（发送或中止）执行完毕时，由硬件置 1。
由软件通过写入“1”清零，或是在发生发送请求（CAN_TI1R 寄存器中的 TXRQ1 位）时由硬件清零。
如果将此位清零，邮箱 1 的所有状态位（TXOK1、ALST1 和 TERR1）都将清零。

位 7 **ABRQ0**：邮箱 0 中止请求 (Abort request for mailbox 0)
由软件置 1，用于中止相应邮箱的发送请求。
邮箱变为空后，此位由硬件清零。
邮箱未挂起等待发送时，将此位置 1 没有任何作用。

位 6:4 保留，必须保持复位值。

位 3 **TERR0**：邮箱 0 发送错误 (Transmission error of mailbox 0)
如果上一次发送因错误而失败，此位将置 1。

位 2 **ALST0**：邮箱 0 仲裁丢失 (Arbitration lost for mailbox 0)
如果上一次发送因仲裁丢失而失败，此位将置 1。

位 1 **TXOK0**：邮箱 0 发送成功 (Transmission OK of mailbox 0)
每次发送尝试后，硬件都将更新此位。
0：上一次发送失败
1：上一次发送成功
当邮箱 1 的发送请求成功完成时，此位由硬件置 1。请参见*图 229*。

位 0 **RQCP0**：邮箱 0 请求完成 (Request completed mailbox 0)
最后一个请求（发送或中止）执行完毕时，由硬件置 1。
由软件通过写入“1”清零，或是在发生发送请求（CAN_TI0R 寄存器的 TXRQ0 位置 1）时由硬件清零。
如果将此位清零，邮箱 0 的所有状态位（TXOK0、ALST0 和 TERR0）都将清零。

CAN 接收 FIFO 0 寄存器 (CAN_RF0R)

CAN receive FIFO 0 register

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
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved										RFOM0	FOVR0	FULL0	Res.	FMP0[1:0]	
										rs	rc_w1	rc_w1		r	r

位 31:6 保留，必须保持复位值。

位 5 **RFOM0**：释放 FIFO 0 输出邮箱 (Release FIFO 0 output mailbox)
由软件置 1，用于释放 FIFO 的输出邮箱。FIFO 中至少有一条消息挂起时，才能释放输出邮箱。FIFO 为空时，将此位置 1 没有任何作用。如果 FIFO 中至少有两条消息挂起，软件必须释放输出邮箱，才能访问下一条消息。
输出邮箱释放后，此位由硬件清零。

位 4 **FOVR0**：FIFO 0 上溢 (FIFO 0 overrun)
FIFO 填满时，如果接收到新消息并且通过筛选器，此位将由硬件置 1。
此位由软件清零。

位 3 **FULL0**：FIFO 0 满(FIFO 0 full)
FIFO 中存储三条消息后，由硬件置 1。
此位由软件清零。

位 2 保留，必须保持复位值。

位 1:0 **FMP0[1:0]**：FIFO 0 消息挂起 (FIFO 0 message pending)
这些位用于指示接收 FIFO 中挂起的消息数。
硬件每向 FIFO 存储一条新消息，FMP 即会增加。软件每次通过将 RFOM0 位置 1 来释放输出邮箱，FMP 即会减小。

CAN 接收 FIFO 1 寄存器 (CAN_RF1R)

CAN receive FIFO 1 register

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
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved										RFOM1	FOVR1	FULL1	Res.	FMP1[1:0]	
										rs	rc_w1	rc_w1		r	r

位 31:6 保留，必须保持复位值。

位 5 **RFOM1**：释放 FIFO 1 输出邮箱 (Release FIFO 1 output mailbox)
由软件置 1，用于释放 FIFO 的输出邮箱。FIFO 中至少有一条消息挂起时，才能释放输出邮箱。FIFO 为空时，将此位置 1 没有任何作用。如果 FIFO 中至少有两条消息挂起，软件必须释放输出邮箱，才能访问下一条消息。
输出邮箱释放后，此位由硬件清零。

位 4 **FOVR1**：FIFO 1 上溢 (FIFO 1 overrun)
FIFO 填满时，如果接收到新消息并且通过筛选器，此位将由硬件置 1。
此位由软件清零。

位 3 **FULL1**：FIFO 1 满 (FIFO 1 full)
FIFO 中存储三条消息后，由硬件置 1。
此位由软件清零。

位 2 保留，必须保持复位值。

位 1:0 **FMP1[1:0]**：FIFO 1 消息挂起 (FIFO 1 message pending)
这些位用于指示接收 FIFO1 中挂起的消息数。
硬件每向 FIFO1 存储一条新消息，FMP1 即会增加。软件每次通过将 RFOM1 位置 1 来释放输出邮箱，FMP 即会减小。

CAN 中断使能寄存器 (CAN_IER)

CAN interrupt enable register

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
Reserved														SLKIE	WKUIE
														rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ERRIE	Reserved			LEC IE	BOF IE	EPV IE	EWG IE	Res.	FOV IE1	FF IE1	FMP IE1	FOV IE0	FF IE0	FMP IE0	TME IE
rw				rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw

位 31:18 保留，必须保持复位值。

位 17 **SLKIE**：睡眠中断使能 (Sleep interrupt enable)
0：SLAKI 位置 1 时不产生中断。
1：SLAKI 位置 1 时产生中断。

位 16 **WKUIE**：唤醒中断使能 (Wakeup interrupt enable)
0：WKUI 置 1 时不产生中断。
1：WKUI 位置 1 时产生中断。

位 15 **ERRIE**：错误中断使能 (Error interrupt enable)
0：CAN_ESR 中有挂起的错误状况时，不会产生中断。
1：CAN_ESR 中有挂起的错误状况时，会产生中断。

位 14:12 保留，必须保持复位值。

位 11 **LECIE**：上一个错误代码中断使能 (Last error code interrupt enable)
0：如果在检测到错误后硬件将 LEC[2:0] 中的错误代码置 1，则不会将 ERRI 位置 1。
1：如果在检测到错误后硬件将 LEC[2:0] 中的错误代码置 1，会将 ERRI 位置 1。

位 10 **BOFIE**：总线关闭中断使能 (Bus-off interrupt enable)
0：BOFF 置 1 时，不会将 ERRI 位置 1。
1：BOFF 置 1 时，将 ERRI 位置 1。

位 9 **EPVIE**：错误被动中断使能 (Error passive interrupt enable)
0：EPVF 置 1 时，不会将 ERRI 位置 1。
1：EPVF 置 1 时，将 ERRI 位置 1。

位 8 **EWGIE**：错误警告中断使能 (Error warning interrupt enable)
0：EWGF 置 1 时，不会将 ERRI 位置 1。
1：EWGF 置 1 时，将 ERRI 位置 1。

位 7 保留，必须保持复位值。

位 6 **FOVIE1**：FIFO 上溢中断使能 (FIFO overrun interrupt enable)
0：FOVR 置 1 时不产生中断。
1：FOVR 置 1 时产生中断。

位 5 **FFIE1**：FIFO 满中断使能 (FIFO full interrupt enable)
0：FULL 位置 1 时不产生中断。
1：FULL 位置 1 时产生中断。

位 4 **FMPIE1**：FIFO 消息挂起中断使能 (FIFO message pending interrupt enable)
0：FMP[1:0] 位的状态不是 00b 时，不产生中断。
1：FMP[1:0] 位的状态不是 00b 时，产生中断。

位 3 **FOVIE0**：FIFO 上溢中断使能 (FIFO overrun interrupt enable)
0：FOVR 位置 1 时不产生中断。
1：FOVR 位置 1 时产生中断。

位 2 **FFIE0**：FIFO 满中断使能 (FIFO full interrupt enable)
0：FULL 位置 1 时不产生中断。
1：FULL 位置 1 时产生中断。

位 1 **FMPIE0**:FIFO 消息挂起中断使能 (FIFO message pending interrupt enable)
0：FMP[1:0] 位的状态不是 00b 时，不产生中断。
1：FMP[1:0] 位的状态不是 00b 时，产生中断。

位 0 **TMEIE**：发送邮箱空中断使能 (Transmit mailbox empty interrupt enable)

0：RQCPx 位置 1 时不产生中断。

1：RQCPx 位置 1 时产生中断。

注意：请参见第 24.8 节：bxCAN 中断。

CAN 错误状态寄存器 (CAN_ESR)

CAN error status register

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	r	r	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
Reserved								LEC[2:0]				Res.	BOFF	EPVF	EWGF
								rw	rw	rw			r	r	r

位 31:24 **REC[7:0]**：接收错误计数器 (Receive error counter)

CAN 协议故障隔离机制的实施部分。如果接收期间发生错误，该计数器按 1 或 8 递增，具体取决于 CAN 标准所定义的错误状况。每次成功接收后，该计数器按 1 递减，如果其数值大于 128，则复位为 120。计数器值超过 127 时，CAN 控制器进入错误被动状态。

位 23:16 **TEC[7:0]**：9 位发送错误计数器最低有效字节 (Least significant byte of the 9-bit transmit error counter)

CAN 协议故障隔离机制的实施部分。

位 15:7 保留，必须保持复位值。

位 6:4 **LEC[2:0]**：上一个错误代码 (Last error code)

该字段由硬件置 1，其中的代码指示 CAN 总线上检测到的上一个错误的错误状况。如果消息成功传送（接收或发送）且未发生错误，该字段将清为“0”。

LEC[2:0] 位可由软件置为 0b111 值。这些位由硬件更新，以指示当前通信状态。

000：无错误
001：填充错误
010：格式错误
011：确认错误
100：位隐性错误
101：位显性错误
110：CRC 错误
111：由软件置 1

位 3 保留，必须保持复位值。

位 2 **BOFF**：总线关闭标志 (Bus-off flag)

此位由硬件在进入睡眠状态时置 1。TEC 上溢（超过 255）时，进入总线关闭状态，请参见*第 620 页的第 24.7.6 节*。

位 1 **EPVF**：错误被动标志 (Error passive flag)

达到错误被动极限（接收错误计数器或发送错误计数器 > 127）时，此位由硬件置 1。

位 0 **EWGF**：错误警告标志 (Error warning flag)

达到警告极限时，此位由硬件置 1（接收错误计数器或发送错误计数器≥96）。

CAN 位时序寄存器 (CAN_BTR)

CAN bit timing register

偏移地址：0x1C
复位值：0x0123 0000

只有 CAN 硬件处于初始化模式时，才能由软件访问此寄存器。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SILM	LBKM	Reserved				SJW[1:0]		Res.	TS2[2:0]			TS1[3:0]			
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
Reserved						BRP[9:0]									
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31 **SILM**：静默模式（调试）(Silent mode (debug))
0：正常工作
1：静默模式

位 30 **LBKM**：环回模式（调试）(Loop back mode (debug))
0：禁止环回模式
1：使能环回模式

位 29:26 保留，必须保持复位值。

位 25:24 **SJW[1:0]**：再同步跳转宽度 (Resynchronization jump width)
这些位定义 CAN 硬件在执行再同步时最多可以将位加长或缩短的时间片数目。
$t_{RJW} = t_{CAN} \times (SJW[1:0] + 1)$

位 23 保留，必须保持复位值。

位 22:20 **TS2[2:0]**：时间段 2 (Time segment 2)
这些位定义时间段 2 中的时间片数目。
$t_{BS2} = t_{CAN} \times (TS2[2:0] + 1)$

位 19:16 **TS1[3:0]**：时间段 1 (Time segment 1)
这些位定义时间段 1 中的时间片数目。
$t_{BS1} = t_{CAN} \times (TS1[3:0] + 1)$
有关位时序的详细信息，请参见*第 621 页的第 24.7.7 节：位时序*。

位 15:10 保留，必须保持复位值。

位 9:0 **BRP[9:0]**：波特率预分频器 (Baud rate prescaler)
这些位定义一个时间片的长度。
$t_q = (BRP[9:0]+1) \times t_{PCLK}$

24.9.3 CAN 邮箱寄存器

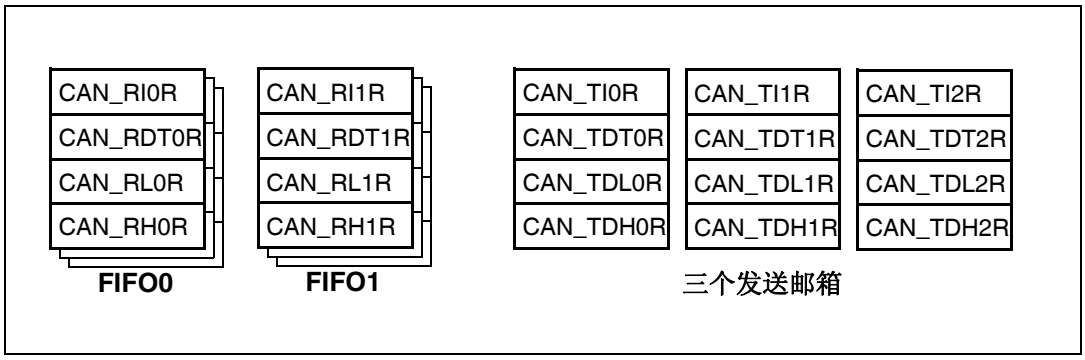
本章介绍发送和接收邮箱的寄存器。有关详细的寄存器映射，请参见*第 619 页的第 24.7.5 节：消息存储*。

除了以下情况外，发送邮箱和接收邮箱使用相同的寄存器：

- CAN_RDTxR 寄存器中的 FMI 字段。
- 接收邮箱始终处于写保护状态。
- 发送邮箱仅在为空时才处于可写状态（CAN_TSR 寄存器中的相应 TME 位置 1）。

发送邮箱共有 3 个，接收邮箱共有 2 个。每个接收邮箱允许访问一个深度为 3 级的 FIFO，访问仅限于 FIFO 中最早接收到的消息。

每个邮箱由 4 个寄存器组成。



CAN 发送邮箱标识符寄存器 (CAN_TIxR) (x=0..2)

CAN TX mailbox identifier register

偏移地址：0x180、0x190、0x1A0
复位值：0xXXXX XXXX（位 0 除外，TXRQ = 0）

当邮箱处于发送挂起状态（TMEx 复位）时，所有发送寄存器均为写保护状态。

该寄存器还会进行发送请求控制（位 0）- 复位值 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	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
EXID[12:0]													IDE	RTR	TXRQ
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	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)
标准标识符或扩展标识符的 MSB（取决于 IDE 位的值）。

位 20:3 **EXID[17:0]**：扩展标识符 (Extended identifier)
扩展标识符的 LSB。

位 2 **IDE**：标识符扩展 (Identifier extension)
此位用于定义邮箱中消息的标识符类型。
0：标准标识符。
1：扩展标识符。

位 1 **RTR**：远程发送请求 (Remote transmission request)
0：数据帧
1：遥控帧

位 0 **TXRQ**：发送邮箱请求 (Transmit mailbox request)
由软件置 1，用于请求发送相应邮箱的内容。
邮箱变为空后，此位由硬件清零。

CAN 邮箱数据长度控制和时间戳寄存器 (CAN_TDTxR) (x=0..2)

CAN mailbox data length control and time stamp register

当邮箱未处于空状态时，该寄存器的所有位均为写保护状态。

偏移地址：0x184、0x194、0x1A4
复位值：0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TIME[15:0]															
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
Reserved							TGT	Reserved				DLC[3:0]			
							rw					rw	rw	rw	rw

位 31:16 **TIME[15:0]**：消息时间戳 (Message time stamp)
此字段包含在进行 SOF 发送时所捕获的 16 位定时器值。

位 15:9 保留，必须保持复位值。

位 8 **TGT**：发送全局时间 (Transmit global time)
只有硬件处于时间触发通信模式（CAN_MCR 寄存器的 TTCM 位置 1）时，此位才会激活。
0：不发送时间戳 TIME[15:0]。
1：在 8 字节消息的最后两个数据字节中发送时间戳 TIME[15:0] 的值。数据字节 7 对应 TIME[7:0]，数据字节 6 对应 TIME[15:8]，该值将替换 CAN_TDHxR[31:16] 寄存器（DATA6[7:0] 和 DATA7[7:0]）中写入的数据。DLC 必须编程为 8，才能通过 CAN 总线发送这两个字节。

位 7:4 保留，必须保持复位值。

位 3:0 **DLC[3:0]**：数据长度代码 (Data length code)
该字段定义数据帧或遥控帧请求中的数据字节数。
一条消息可以包含 0 到 8 个数据字节，具体取决于 DLC 字段的值。

CAN 邮箱数据低位寄存器 (CAN_TDLxR) (x=0..2)

CAN mailbox data low register

当邮箱未处于空状态时，该寄存器的所有位均为写保护状态。

偏移地址：0x188、0x198、0x1A8
复位值：0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATA3[7:0]								DATA2[7:0]							
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
DATA1[7:0]								DATA0[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:24 **DATA3[7:0]**：数据字节 3 (Data byte 3)
消息的数据字节 3。

位 23:16 **DATA2[7:0]**：数据字节 2 (Data byte 2)
消息的数据字节 2。

位 15:8 **DATA1[7:0]**：数据字节 1 (Data byte 1)
消息的数据字节 1。

位 7:0 **DATA0[7:0]**：数据字节 0 (Data byte 0)
消息的数据字节 0。
一条消息可以包含 0 到 8 个数据字节，从字节 0 开始。

CAN 邮箱数据高位寄存器 (CAN_TDHxR) (x=0..2)

CAN mailbox data high register

当邮箱未处于空状态时，该寄存器的所有位均为写保护状态。

偏移地址：0x18C、0x19C、0x1AC
复位值：0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATA7[7:0]								DATA6[7:0]							
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
DATA5[7:0]								DATA4[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:24 **DATA7[7:0]**：数据字节 7 (Data byte 7)
消息的数据字节 7。
注意：如果 TTCG 以及此消息的 TGT 为激活状态，则 DATA7 和 DATA6 将以时间戳值替换。

位 23:16 **DATA6[7:0]**：数据字节 6 (Data byte 6)
消息的数据字节 6。

位 15:8 **DATA5[7:0]**：数据字节 5 (Data byte 5)
消息的数据字节 5。

位 7:0 **DATA4[7:0]**：数据字节 4 (Data byte 4)
消息的数据字节 4。

CAN 接收 FIFO 邮箱标识符寄存器 (CAN_RIxR) (x=0..1)

CAN receive FIFO mailbox identifier register

偏移地址：0x1B0、0x1C0
复位值：0xXXXX XXXX

所有接收寄存器均受到写保护。

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	r	r	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
EXID[12:0]													IDE	RTR	Res.
r	r	r	r	r	r	r	r	r	r	r	r	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)
标准标识符或扩展标识符的 MSB（取决于 IDE 位的值）。

位 20:3 **EXID[17:0]**：扩展标识符 (Extended identifier)
扩展标识符的 LSB。

位 2 **IDE**：标识符扩展 (Identifier extension)
此位用于定义邮箱中消息的标识符类型。
0：标准标识符。
1：扩展标识符。

位 1 **RTR**：远程发送请求 (Remote transmission request)
0：数据帧
1：遥控帧

位 0 保留，必须保持复位值。

CAN 接收 FIFO 邮箱数据长度控制和时间戳寄存器 (CAN_RDTxR) (x=0..1)

CAN receive FIFO mailbox data length control and time stamp register

偏移地址：0x1B4、0x1C4
复位值：0xXXXX XXXX

所有接收寄存器均受到写保护。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TIME[15:0]															
r	r	r	r	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
FMI[7:0]								Reserved				DLC[3:0]			
r	r	r	r	r	r	r	r					r	r	r	r

位 31:16 **TIME[15:0]**：消息时间戳 (Message time stamp)
此字段包含在进行 SOF 检测时所捕获的 16 位定时器值。

位 15:8 **FMI[7:0]**：筛选器匹配索引 (Filter match index)
该寄存器包含筛选器索引，邮箱中存储的消息需要经过该筛选器。有关标识符筛选的更多详细信息，请参见*第 616 页的第 24.7.4 节：标识符筛选*的**筛选器匹配索引**一节。

位 7:4 保留，必须保持复位值。

位 3:0 **DLC[3:0]**：数据长度代码 (Data length code)
该字段定义一个数据帧所包含的数据字节数（0 到 8）。如果是远程帧请求，则为 0。

CAN 接收 FIFO 邮箱数据低位寄存器 (CAN_RDLxR) (x=0..1)

CAN receive FIFO mailbox data low register

当邮箱未处于空状态时，该寄存器的所有位均为写保护状态。

偏移地址：0x1B8、0x1C8
复位值：0xXXXX XXXX

所有接收寄存器均受到写保护。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATA3[7:0]								DATA2[7:0]							
r	r	r	r	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
DATA1[7:0]								DATA0[7:0]							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

位 31:24 **DATA3[7:0]**：数据字节 3 (Data byte 3)
消息的数据字节 3。

位 23:16 **DATA2[7:0]**：数据字节 2 (Data byte 2)
消息的数据字节 2。

位 15:8 **DATA1[7:0]**：数据字节 1 (Data byte 1)
消息的数据字节 1。

位 7:0 **DATA0[7:0]**：数据字节 0 (Data byte 0)
消息的数据字节 0。
一条消息可以包含 0 到 8 个数据字节，从字节 0 开始。

CAN 接收 FIFO 邮箱数据高位寄存器 (CAN_RDHxR) (x=0..1)

CAN receive FIFO mailbox data high register

偏移地址：0x1BC、0x1CC
复位值：0xXXXX XXXX

所有接收寄存器均受到写保护。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATA7[7:0]								DATA6[7:0]							
r	r	r	r	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
DATA5[7:0]								DATA4[7:0]							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

位 31:24 **DATA7[7:0]**：数据字节 7 (Data byte 7)
消息的数据字节 3。

位 23:16 **DATA6[7:0]**：数据字节 6 (Data byte 6)
消息的数据字节 2。

位 15:8 **DATA5[7:0]**：数据字节 5 (Data byte 5)
消息的数据字节 1。

位 7:0 **DATA4[7:0]**：数据字节 4 (Data byte 4)
消息的数据字节 0。

24.9.4 CAN 筛选器寄存器

CAN 筛选器主寄存器 (CAN_FMR)

CAN filter master register

偏移地址：0x200
复位值：0x2A1C 0E01

此寄存器的所有位均由软件置 1 和清零。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		CAN2SB[5:0]						Reserved							FINIT
		rw	rw	rw	rw	rw	rw								rw

位 31:14 保留，必须保持复位值。

位 13:8 **CAN2SB[5:0]**：CAN2 起始存储区 (CAN2 start bank)

这些位将由软件置 1 和清零。它们为处于 0 到 27 范围内的 CAN2 接口（从模式）定义起始存储区。

注意：CAN2SB[5:0] = 28d 时，可以使用 CAN1 的所有筛选器。

CAN2SB[5:0] 设置为 0 时，不会为 CAN1 分配任何筛选器。

位 7:1 保留，必须保持复位值。

位 0 **FINIT**：筛选器初始化模式 (Filter init mode)

筛选器组的初始化模式

0：筛选器工作模式。

1：筛选器初始化模式。

CAN 筛选器模式寄存器 (CAN_FM1R)

CAN filter mode register

偏移地址：0x204

复位值：0x0000 0000

仅当 CAN_FMR 寄存器中设置了筛选器初始化模式 (FINIT=1) 时，才能对此寄存器执行写操作。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved				FBM27	FBM26	FBM25	FBM24	FBM23	FBM22	FBM21	FBM20	FBM19	FBM18	FBM17	FBM16
				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
FBM15	FBM14	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	rw	rw

注意：请参见第 617 页的图 231：筛选器组尺度配置 - 寄存器构成。

位 31:28 保留，必须保持复位值。

位 27:0 **FBM*x***：筛选器模式 (Filter mode)

筛选器 x 的寄存器的模式

0：筛选器存储区 x 的两个 32 位寄存器处于标识符屏蔽模式。

1：筛选器存储区 x 的两个 32 位寄存器处于标识符列表模式。

CAN 筛选器尺度寄存器 (CAN_FS1R)

CAN filter scale register

偏移地址：0x20C
复位值：0x0000 0000

仅当 CAN_FMR 寄存器中设置了筛选器初始化模式 (FINIT=1) 时，才能对此寄存器执行写操作。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved				FSC27	FSC26	FSC25	FSC24	FSC23	FSC22	FSC21	FSC20	FSC19	FSC18	FSC17	FSC16
				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
FSC15	FSC14	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	rw	rw

位 31:28 保留，必须保持复位值。

位 27:0 **FSCx**：筛选器尺度配置 (Filter scale configuration)
这些位定义了筛选器 13-0 的尺度配置。
0：双 16 位尺度配置
1：单 32 位尺度配置

注意： *请参见第 617 页的图 231：筛选器组尺度配置 - 寄存器构成。*

CAN 筛选器 FIFO 分配寄存器 (CAN_FFA1R)

CAN filter FIFO assignment register

偏移地址：0x214
复位值：0x0000 0000

仅当 CAN_FMR 寄存器中设置了筛选器初始化模式 (FINIT=1) 时，才能对此寄存器执行写操作。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved				FFA27	FFA26	FFA25	FFA24	FFA23	FFA22	FFA21	FFA20	FFA19	FFA18	FFA17	FFA16
				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
FFA15	FFA14	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	rw	rw

位 31:28 保留，必须保持复位值。

位 27:0 **FFA*x***：筛选器 x 的筛选器 FIFO 分配 (Filter FIFO assignment for filter x)
通过此筛选器的消息将存储在指定的 FIFO 中。
0：筛选器分配到 FIFO 0
1：筛选器分配到 FIFO 1

CAN 筛选器激活寄存器 (CAN_FA1R)

CAN filter activation register

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
Reserved				FACT27	FACT26	FACT25	FACT24	FACT23	FACT22	FACT21	FACT20	FACT19	FACT18	FACT17	FACT16
				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
FACT15	FACT14	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	rw	rw

位 31:28 保留，必须保持复位值。

位 27:0 **FACT*x***：筛选器激活 (Filter active)

软件将此位置 1 可激活筛选器 x。要修改筛选器 x 的寄存器 (CAN_FxR[0:7])，必须将 FACTx 位清零或将 CAN_FMR 寄存器的 FINIT 位置 1。
0：筛选器 x 未激活
1：筛选器 x 激活

筛选器组 i 寄存器 x (CAN_FiRx)（i=0..27，x=1，2）

Filter bank i register x

偏移地址：0x240..0x31C
复位值：0xXXXX XXXX

共有 28 个筛选器组，i=0 ..27。每个筛选器组 i 由两个 32 位寄存器组成，即 CAN_FiR[2:1]。

仅当 CAN_FAxR 寄存器的 FACTx 位清零，或者 CAN_FMR 寄存器的 FINIT 位置 1 时，才能修改此寄存器。

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

在所有配置中：

位 31:0 **FB[31:0]**：筛选器位 (Filter bits)

标识符
寄存器的每一位用于指定预期标识符相应位的级别。
0：需要显性位
1：需要隐性位
掩码
寄存器的每一位用于指定相关标识符寄存器的位是否必须与预期标识符的相应位匹配。
0：无关，不使用此位进行比较。
1：必须匹配，传入标识符的此位必须与筛选器相应标识符寄存器中指定的级别相同。

注意： *每个寄存器的功能可能因筛选器的尺度和模式配置而异。有关筛选器映射、功能说明和掩码寄存器关联的信息，请参见第 616 页的第 24.7.4 节：标识符筛选。*

掩码模式中，掩码/标识符寄存器的位映射与**标识符列表**模式中的相同。

有关筛选器组的寄存器映射/地址信息，请参见第 644 页的表 102。

24.9.5 bxCAN 寄存器映射

有关寄存器边界地址的信息，请参见*第 52 页的表 2*。寄存器仅在 CAN1 中位于偏移量为 0x200 到 31C 的地址处。

表 102. bxCAN 寄存器映射和复位值

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	**CAN_MCR**	Reserved															DBF	RESET	Reserved							TTCM	ABOM	AWUM	NART	RFLM	TXFP	SLEEP	INRQ
	Reset value																1	0								0	0	0	0	0	0	1	0
0x004	**CAN_MSR**	Reserved																				RX	SAMP	RXM	TXM	Reserved			SLAKI	WKUI	ERRI	SLAK	INAK
	Reset value																					1	1	0	0				0	0	0	1	0
0x008	**CAN_TSR**	LOW[2:0]			TME[2:0]			CODE[1:0]		ABRQ2	Reserved			TERR2	ALST2	TXOK2	RQCP2	ABRQ1	Reserved			TERR1	ALST1	TXOK1	RQCP1	ABRQ0	Reserved			TERR0	ALST0	TXOK0	RQCP0
	Reset value	0	0	0	1	1	1	0	0	0				0	0	0	0	0				0	0	0	0	0				0	0	0	0
0x00C	**CAN_RF0R**	Reserved																										RFOM0	FOVR0	FULL0	Reserved	FMP0[1:0]	
	Reset value																											0	0	0		0	0
0x010	**CAN_RF1R**	Reserved																										RFOM1	FOVR1	FULL1	Reserved	FMP1[1:0]	
	Reset value																											0	0	0		0	0
0x014	**CAN_IER**	Reserved													SLKIE	WKUIE	ERRIE	Reserved				LECIE	BOFIE	EPVIE	EWGIE	Reserved	FOVIE1	FFIE1	FMPIE1	FOVIE0	FFIE0	FMPIE0	TMEIE
	Reset value														0	0	0					0	0	0	0		0	0	0	0	0	0	0
0x018	**CAN_ESR**	REC[7:0]								TEC[7:0]									Reserved							LEC[2:0]			Reserved	BOFF	EPVF	EWGF	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0								0	0	0		0	0	0	
0x01C	**CAN_BTR**	SILM	LBKM	Reserved				SJW[1:0]		Reserved	TS2[2:0]			TS1[3:0]					Reserved							BRP[9:0]							
	Reset value	0	0					0	0		0	1	0	0	0	1	1									0	0	0	0	0	0	0	0	0
0x020-0x17F		Reserved																															
0x180	**CAN_TI0R**	STID[10:0]/EXID[28:18]											EXID[17:0]																		IDE	RTR	TXRQ
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	0
0x184	**CAN_TDT0R**	TIME[15:0]																	Reserved						TGT	Reserved				DLC[3:0]			
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x							x					x	x	x	x
0x188	**CAN_TDL0R**	DATA3[7:0]								DATA2[7:0]									DATA1[7:0]											DATA0[7:0]			
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

表 102. bxCAN 寄存器映射和复位值 （续）

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x18C	**CAN_TDH0R**	DATA7[7:0]								DATA6[7:0]								DATA5[7:0]								DATA4[7:0]							
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0x190	**CAN_TI1R**	STID[10:0]/EXID[28:18]								EXID[17:0]																					IDE	RTR	TXRQ
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	0
0x194	**CAN_TDT1R**	TIME[15:0]																Reserved							TGT	Reserved				DLC[3:0]			
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x								x					x	x	x	x
0x198	**CAN_TDL1R**	DATA3[7:0]								DATA2[7:0]								DATA1[7:0]								DATA0[7:0]							
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0x19C	**CAN_TDH1R**	DATA7[7:0]								DATA6[7:0]								DATA5[7:0]								DATA4[7:0]							
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0x1A0	**CAN_TI2R**	STID[10:0]/EXID[28:18]								EXID[17:0]																					IDE	RTR	TXRQ
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	0
0x1A4	**CAN_TDT2R**	TIME[15:0]																Reserved							TGT	Reserved				DLC[3:0]			
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x								x					x	x	x	x
0x1A8	**CAN_TDL2R**	DATA3[7:0]								DATA2[7:0]								DATA1[7:0]								DATA0[7:0]							
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0x1AC	**CAN_TDH2R**	DATA7[7:0]								DATA6[7:0]								DATA5[7:0]								DATA4[7:0]							
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0x1B0	**CAN_RI0R**	STID[10:0]/EXID[28:18]								EXID[17:0]																					IDE	RTR	Reserved
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	
0x1B4	**CAN_RDT0R**	TIME[15:0]																FMI[7:0]								Reserved				DLC[3:0]			
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x					x	x	x	x
0x1B8	**CAN_RDL0R**	DATA3[7:0]								DATA2[7:0]								DATA1[7:0]								DATA0[7:0]							
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0x1BC	**CAN_RDH0R**	DATA7[7:0]								DATA6[7:0]								DATA5[7:0]								DATA4[7:0]							
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0x1C0	**CAN_RI1R**	STID[10:0]/EXID[28:18]								EXID[17:0]																					IDE	RTR	Reserved
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	
0x1C4	**CAN_RDT1R**	TIME[15:0]																FMI[7:0]								Reserved				DLC[3:0]			
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x					x	x	x	x
0x1C8	**CAN_RDL1R**	DATA3[7:0]								DATA2[7:0]								DATA1[7:0]								DATA0[7:0]							
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

表 102. bxCAN 寄存器映射和复位值 （续）

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x1CC	**CAN_RDH1R**	DATA7[7:0]								DATA6[7:0]								DATA5[7:0]								DATA4[7:0]							
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0x1D0-0x1FF	Reserved																																
0x200	**CAN_FMR**	Reserved																		CAN2SB[5:0]						Reserved							FINIT
	Reset value																			0	0	1	1	1	0								1
0x204	**CAN_FM1R**	Reserved								FBM[27:0]																							
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x208	Reserved																																
0x20C	**CAN_FS1R**	Reserved								FSC[27:0]																							
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x210	Reserved																																
0x214	**CAN_FFA1R**	Reserved								FFA[27:0]																							
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x218	Reserved																																
0x21C	**CAN_FA1R**	Reserved								FACT[27:0]																							
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x220	Reserved																																
0x224-0x23F	Reserved																																
0x240	**CAN_F0R1**	FB[31:0]																															
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0x244	**CAN_F0R2**	FB[31:0]																															
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0x248	**CAN_F1R1**	FB[31:0]																															
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0x24C	**CAN_F1R2**	FB[31:0]																															
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
. . . .	. . . .	.																															
0x318	**CAN_F27R1**	FB[31:0]																															
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0x31C	**CAN_F27R2**	FB[31:0]																															
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

25 内部集成电路 (I²C) 接口

除非特别说明，否则本部分适用于整个 STM32F4xx 系列。

25.1 I²C 简介

I²C（内部集成电路）总线接口用作微控制器和 I²C 串行总线之间的接口。它提供多主模式功能，可以控制所有 I²C 总线特定的序列、协议、仲裁和时序。它支持标准和快速模式。它还与 SMBus 2.0 兼容。

它可以用于多种用途，包括 CRC 生成和验证、SMBus（系统管理总线）以及 PMBus（电源管理总线）。

根据器件的不同，可利用 DMA 功能来减轻 CPU 的工作量。

25.2 I²C 主要特性

- 并行总线/I²C 协议转换器
- 多主模式功能：同一接口既可用作主模式也可用作从模式
- I²C 主模式特性：
  - 时钟生成
  - 起始位和停止位生成
- I²C 从模式特性：
  - 可编程 I²C 地址检测
  - 双寻址模式，可对 2 个从地址应答
  - 停止位检测
- 7 位/10 位寻址以及广播呼叫的生成和检测
- 支持不同的通信速度：
  - 标准速度（高达 100 kHz）
  - 快速速度（高达 400 kHz）
- 适用于 STM32F42xxx 和 STM32F43xxx 的可编程数字噪声滤波器
- 状态标志：
  - 发送/接收模式标志
  - 字节传输结束标志
  - I²C 忙碌标志
- 错误标志：
  - 主模式下的仲裁丢失情况
  - 地址/数据传输完成后的应答失败
  - 检测误放的起始位和停止位
  - 禁止时钟延长后出现的上溢/下溢
- 2 个中断向量：
  - 一个中断由成功的地址/数据字节传输事件触发
  - 一个中断由错误状态触发
- 可选的时钟延长

- 带 DMA 功能的 1 字节缓冲
- 可配置的 PEC（数据包错误校验）生成或验证：
  - 在 Tx 模式下，可将 PEC 值作为最后一个字节进行传送
  - 针对最后接收字节的 PEC 错误校验
- SMBus 2.0 兼容性：
  - 25 ms 时钟低电平超时延迟
  - 10 ms 主器件累计时钟低电平延长时间
  - 25 ms 从器件累计时钟低电平延长时间
  - 具有 ACK 控制的硬件 PEC 生成/验证
  - 支持地址解析协议 (ARP)
- PMBus 兼容性

注意： *上述某些特性可能不适用于某些产品。用户应参照产品数据手册来确定 I²C 接口实现所支持的特定特性。*

25.3 I²C 功能说明

除了接收和发送数据之外，此接口还可以从串行格式转换为并行格式，反之亦然。中断由软件使能或禁止。该接口通过数据引脚 (SDA) 和时钟引脚 (SCL) 连接到 I²C 总线。它可以连接到标准（高达 100 kHz）或快速（高达 400 kHz）I²C 总线。

25.3.1 模式选择

该接口在工作时可选用以下四种模式之一：

- 从发送器
- 从接收器
- 主发送器
- 主接收器

默认情况下，它以从模式工作。接口在生成起始位后会自动由从模式切换为主模式，并在出现仲裁丢失或生成停止位时从主模式切换为从模式，从而实现多主模式功能。

通信流程

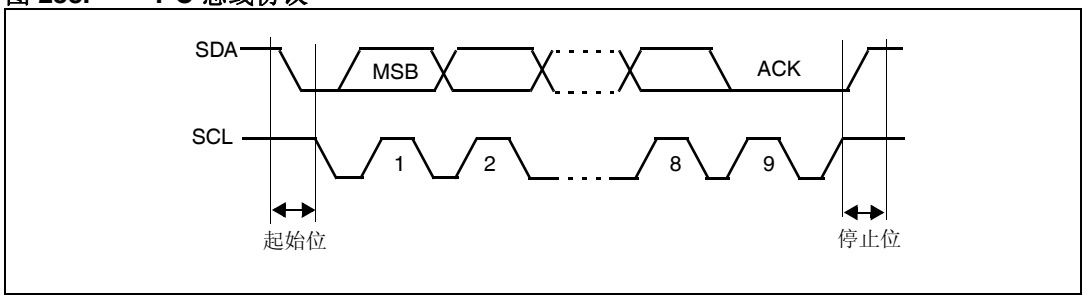
在主模式下，I²C 接口会启动数据传输并生成时钟信号。串行数据传输始终是在出现起始位时开始，在出现停止位时结束。起始位和停止位均在主模式下由软件生成。

在从模式下，该接口能够识别其自身地址（7 或 10 位）以及广播呼叫地址。广播呼叫地址检测可由软件使能或禁止。

数据和地址均以 8 位字节传输，MSB 在前。起始位后紧随地址字节（7 位地址占据一个字节；10 位地址占据两个字节）。地址始终在主模式下传送。

在字节传输 8 个时钟周期后是第 9 个时钟脉冲，在此期间接收器必须向发送器发送一个应答位。请参见*图 238*。

图 238. I²C 总线协议



应答位可由软件使能或禁止。I²C 接口地址（7 位/10 位双寻址模式和/或广播呼叫地址）可通过软件进行选择。

I²C 接口框图如*图 239* 所示。

图 239. STM32F40x/41x 的 I²C 框图

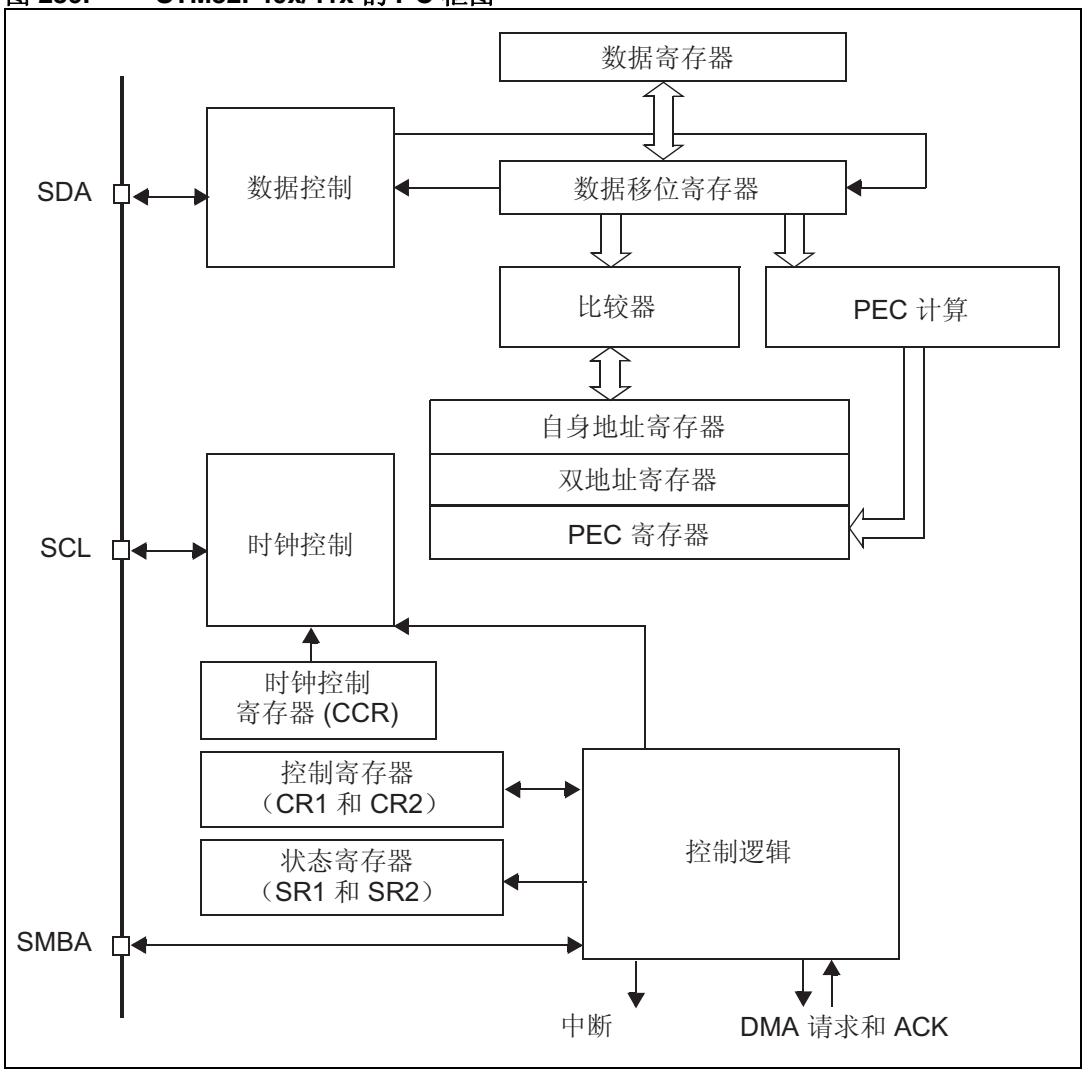
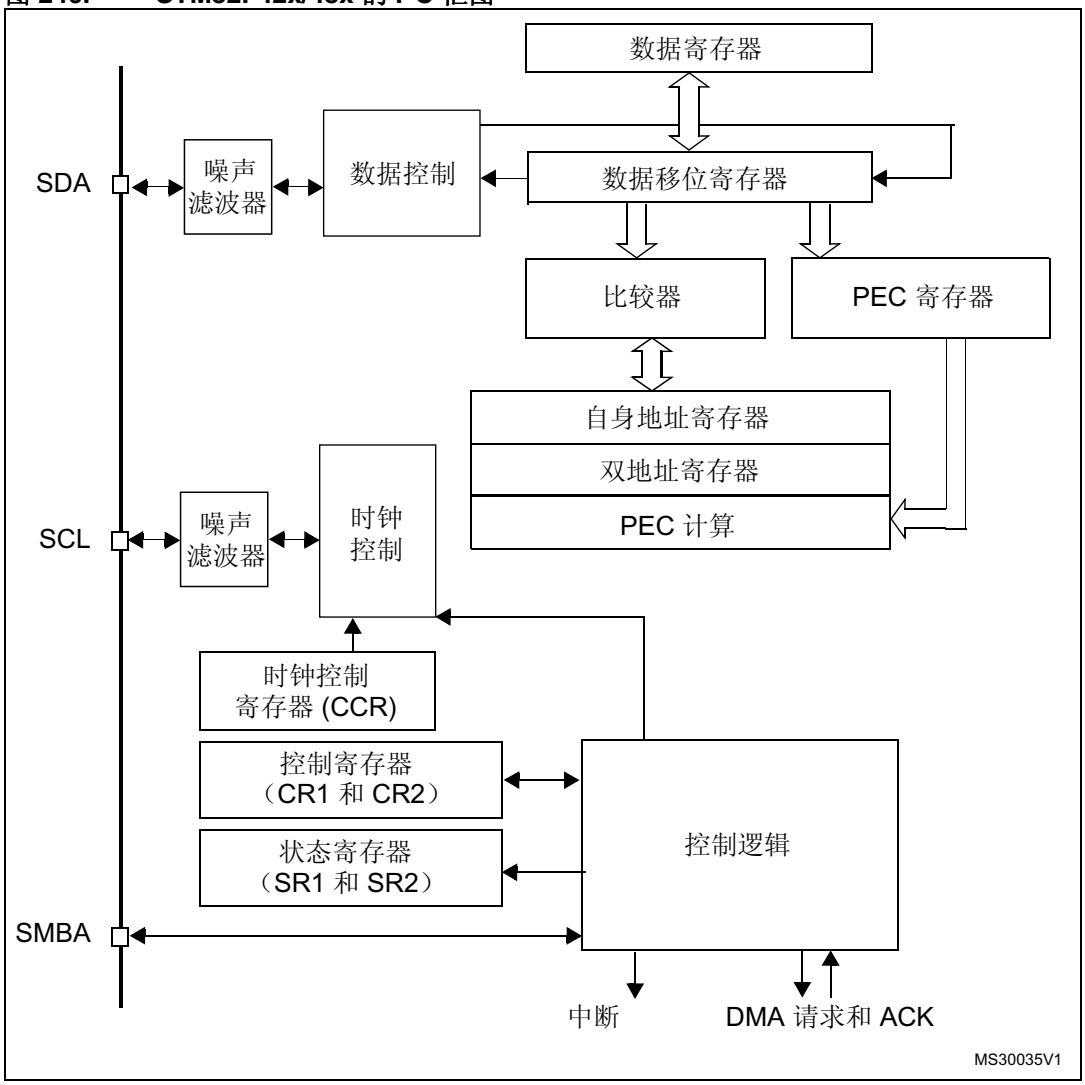


图 240. STM32F42x/43x 的 I²C 框图



1. SMBA 是 SMBus 模式下的一个可选信号。如果 SMBus 已禁止，则此信号不适用。

25.3.2 I²C 从模式

默认情况下，I²C 接口在从模式下工作。要将工作模式由默认的从模式切换为主模式，需要生成一个起始位。

为了生成正确的时序，必须在 I2C_CR2 寄存器中对外设输入时钟进行编程。外设输入时钟频率的下限为：

- 标准模式下 2 MHz
- 快速模式下 4 MHz

检测到起始位后，便会立即接收到来自 SDA 线的地址并将其送到移位寄存器。之后，会将其与接口地址 (OAR1) 和 OAR2（如果 ENDUAL=1）或者广播呼叫地址（如果 ENGC = 1）进行比较。

注意： *在 10 位寻址模式下，比较对象还包括头序列 (11110xx0)，其中，xx 表示该地址的两个最高有效位。*

头或地址不匹配：接口会忽略它并等待下一个起始位。

头匹配（仅针对 10 位模式）：如果 ACK 位置 1，则接口会生成一个应答脉冲并等待 8 位从地址。

地址匹配：接口会依次：

- 发出应答脉冲（如果 ACK 位置 1）
- ADDR 位会由硬件置 1 并在 ITEVFEN 位置 1 时生成一个中断。
- 如果 ENDUAL=1，则软件必须读取 DUALF 位状态来核对哪些从地址进行了应答。

在 10 位模式下，完成地址序列接收后，从模式始终处于接收模式。在接收到重复起始位以及一个匹配地址位和最低有效位均置 1 的头序列 (11110xx1) 后，它会进入发送模式。

TRA 位指示从设备是处于接收模式还是处于发送模式。

从发送器

在接收到地址并将 ADDR 清零后，从设备会通过内部移位寄存器将 DR 寄存器中的字节发送到 SDA 线。

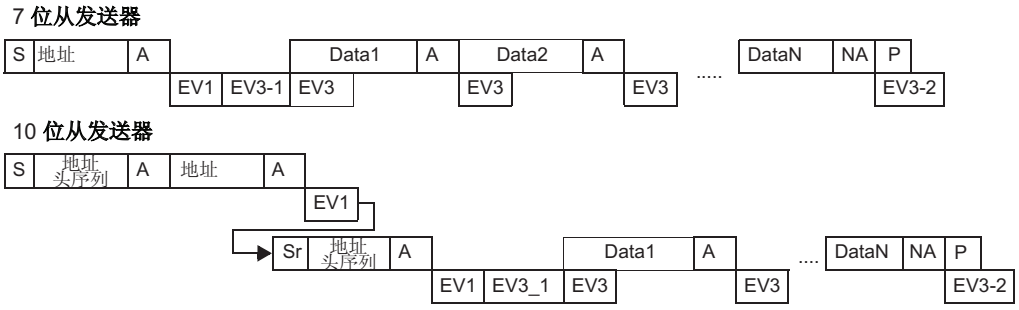
从设备会延长 SCL 低电平时间，直到 ADDR 位清零且 DR 寄存器中填满待发送数据为止（请参见*图 241* 传输序列 EV1 EV3）。

接收到应答脉冲时：

- TxE 位会由硬件置 1 并在 ITEVFEN 和 ITBUFEN 位均置 1 时生成一个中断。

如果在下一次数据传输结束之前 TxE 位已置 1 但某些数据尚未写入 I2C_DR 寄存器，则 BTF 位会置 1，而接口会一直延长 SCL 低电平，直到通过软件对 I2C_SR1 读操作，以及对 I2C_DR 写操作后，把 BTF 清零为止。

图 241.　从发送器的传输序列图



图注：S= 起始位，Sr = 重复起始位，P= 停止位，A= 应答，NA= 非应答，
EVx= 事件（如果 ITEVFEN=1 则发生中断）

EV1：ADDR=1，通过先读取 SR1 再读取 SR2 来清零。
EV3-1：TxE=1，移位寄存器为空，数据寄存器为空，在 DR 中写入 Data1。
EV3：TxE=1，移位寄存器非空，数据寄存器为空，通过对 DR 执行写操作来清零。
EV3-2：AF=1；通过在 SR1 寄存器的 AF 位写入“0”将 AF 清零。

ai18209

1. EV1 和 EV3_1 事件可延长 SCL 低电平时间，直到相应的软件序列结束为止。
2. 如果软件序列在下一个字节传输结束之前未能完成，EV3 事件会延长 SCL 低电平时间。

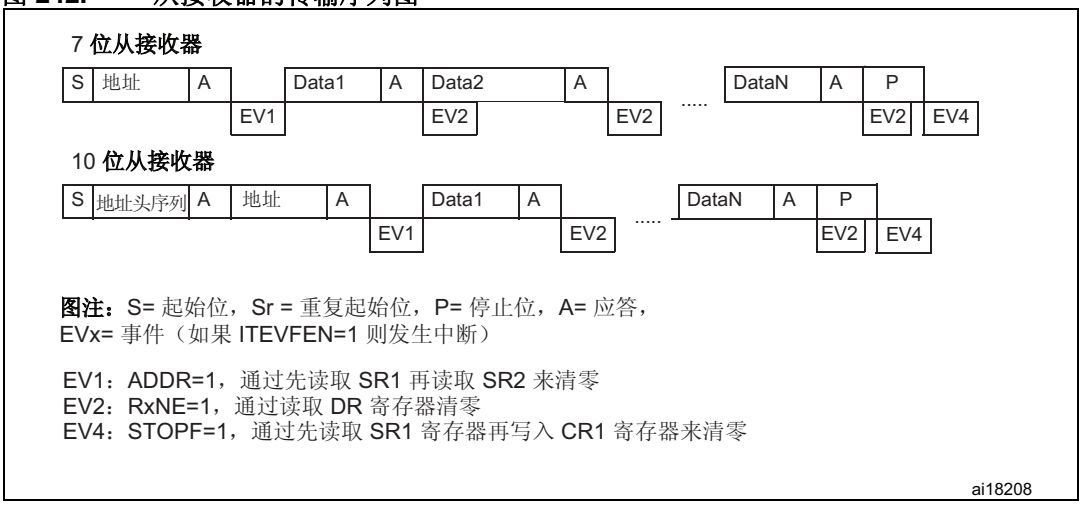
从接收器

在接收到地址并将 ADDR 位清零后，从设备会通过内部移位寄存器接收 SDA 线中的字节并将其保存到 DR 寄存器。在每个字节接收完成后，接口都会依次：

- 发出应答脉冲（如果 ACK 位置 1）
- RxNE 位会由硬件置 1 并在 ITEVFEN 和 ITBUFEN 位均置 1 时生成一个中断。

如果在下一次数据接收结束之前 RxNE 位已置 1 但 DR 寄存器中的数据尚未读取，则 BTF 位会置 1，而接口会一直延长 SCL 低电平，直到软件通过读取 I2C_DR 寄存器来把 BTF 清零（请参见*图 242*传输序列）。

图 242.　从接收器的传输序列图



1. EV1 事件可延长 SCL 低电平时间，直到相应的软件序列结束为止。
2. 如果软件序列在下一个字节接收结束之前未能完成，EV2 事件会延长 SCL 低电平时间。
3. 检查了 SR1 寄存器内容之后，用户应针对各个被置位的标志执行完整的清零序列。
   对于 ADDR 和 STOPF 标志，需要在 I2C 中断程序中执行以下序列：
   对 SR1 执行读操作
   if (ADDR == 1) {对 SR1 执行读操作；对 SR2 执行读操作}
   if (STOPF == 1) {对 SR1 执行读操作；对 CR1 执行写操作}
   这样做的目的是为了确保 ADDR 和 STOPF 这两个标志都清零（如果二者均为被置位）。

关闭从设备通信

传输完最后一个数据字节之后，主设备会生成一个停止位。接口会检测此条件并：

- 将 STOPF 位置 1 并在 ITEVFEN 位置 1 时生成一个中断。

通过先读取 SR1 寄存器然后写入 CR1 寄存器的方式将 STOPF 位清零（请参见*图 242：从接收器的传输序列图* EV4）。

25.3.3　I²C 主模式

在主模式下，I²C 接口会启动数据传输并生成时钟信号。串行数据传输始终是在出现起始位时开始，在出现停止位时结束。只要通过 START 位在总线上生成了起始位，即会选中主模式。

在主模式中要求执行以下序列。

- 在 I2C_CR2 寄存器中对外设输入时钟进行编程，以生成正确的时序
- 配置时钟控制寄存器
- 配置上升时间寄存器
- 对 I2C_CR1 寄存器进行编程，以便使能外设

- 将 I2C_CR1 寄存器的 START 位置 1，以生成起始位

外设输入时钟频率的下限为：

- 标准模式下 2 MHz
- 快速模式下 4 MHz

起始位

START 位置 1 后，接口会在 BUSY 位清零后生成一个起始位并切换到主模式（M/SL 位置 1）。

注意： *在主设备下，START 位置 1 后，接口会在当前字节传输结束后生成一个重复起始位。*

起始位发出之后：

- SB 位会由硬件置 1 并在 ITEVFEN 位置 1 时生成一个中断。

接下来主设备会等待软件对 SR1 执行读操作，然后把从设备地址写入 DR 寄存器（请参见*图 243* 和*图 244* 传输序列 EV5）。

从地址传输

接下来从地址会通过内部移位寄存器发送到 SDA 线。

- 在 10 位寻址模式中，发送头序列会产生以下事件：
  - ADD10 位会由硬件置 1 并在 ITEVFEN 位置 1 时生成一个中断。

    接下来主设备会等待软件读取 SR1 ，然后把第二个地址字节写入 DR 寄存器（请参见*图 243* 和*图 244* 传输序列）。
  - ADDR 位会由硬件置 1 并在 ITEVFEN 位置 1 时生成一个中断。

    接下来主设备会等待对 SR1 寄存器执行读操作，然后对 SR2 寄存器执行读操作（请参见*图 243* 和*图 244* 传输序列）。
- 在 7 位寻址模式下，会发送一个地址字节。

  地址字节被发出后，
  - ADDR 位会由硬件置 1 并在 ITEVFEN 位置 1 时生成一个中断。

    接下来主设备会等待对 SR1 寄存器执行读操作，然后对 SR2 寄存器执行读操作（请参见*图 243* 和*图 244* 传输序列）。

主设备会根据发送的从地址字节 LSB 来决定是进入发送模式还是接收模式。

- 在 7 位寻址模式下，
  - 要进入发送模式，主设备会发送从地址并将 LSB 复位。
  - 要进入接收模式，主设备会发送从地址并将 LSB 置 1。
- 在 10 位寻址模式下，
  - 要进入发送模式，主设备会先发送头序列 (11110xx0)，然后发送从地址（其中 xx 表示该地址的两个最高有效位）。
  - 要进入接收模式，主设备会先发送头序列 (11110xx0)，然后发送从地址。接下来会发送一个重复起始位，然后再发送头序列 (11110xx1)（其中 xx 表示地址的两个最高有效位）。

  TRA 位指示主设备是处于接收模式还是处于发送模式。

主发送器

在发送出地址并将 ADDR 清零后，主设备会通过内部移位寄存器将 DR 寄存器中的字节发送到 SDA 线。

主设备会一直等待，直到首个数据字节被写入 I2C_DR 为止（请参见*图 243* 传输序列 EV8_1）。

接收到应答脉冲后，TxE 位会由硬件置 1 并在 ITEVFEN 和 ITBUFEN 位均置 1 时生成一个中断。

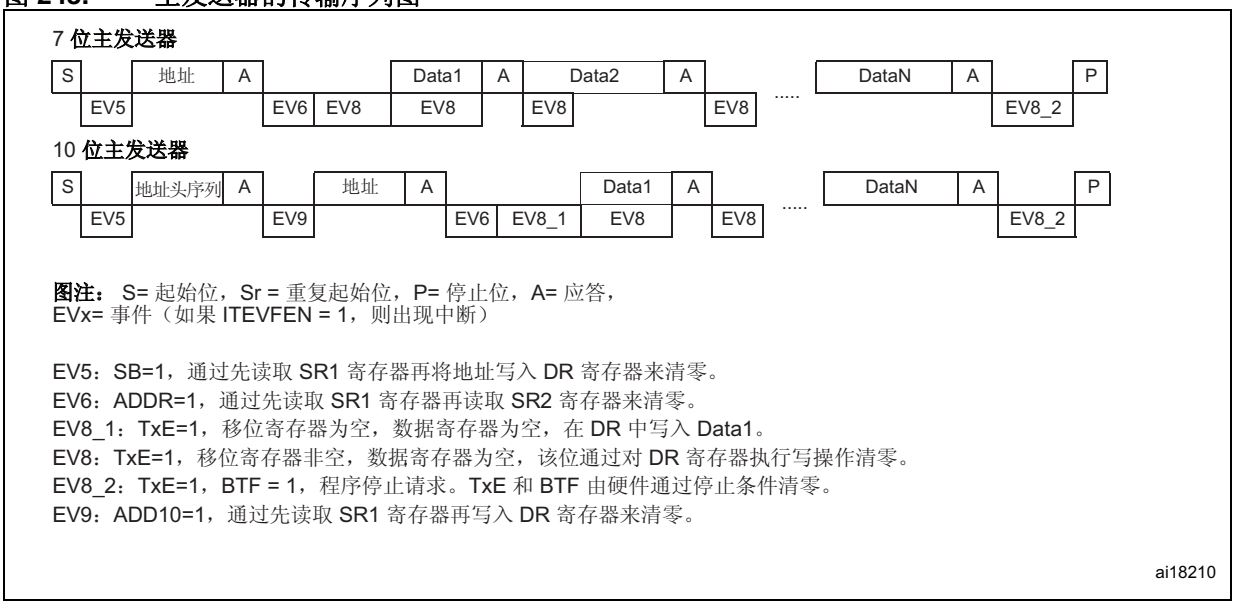
如果在上一次数据传输结束之前 TxE 位已置 1 但数据字节尚未写入 DR 寄存器，则 BTF 位会置 1，而接口会一直延长 SCL 低电平，等待I2C_DR 寄存器被写入，以将 BTF 清零。

结束通信

当最后一个字节写入 DR 寄存器后，软件会将 STOP 位置 1 以生成一个停止位（请参见*图 243* 传输序列 EV8_2）。接口会自动返回从模式（M/SL 位清零）。

注意： *当 TxE 或 BTF 中的任何一个置 1 时，应在 EV8_2 事件期间对停止位进行编程。*

图 243.　主发送器的传输序列图



1. EV5、EV6、EV9、EV8_1 和 EV8_2 事件可延长 SCL 低电平时间，直到相应的软件序列结束为止。
2. 如果软件序列在下一个字节传输结束之前未能完成，EV8 事件会延长 SCL 低电平时间。

主接收器

完成地址传输并将 ADDR 位清零后，I²C 接口会进入主接收模式。在此模式下，接口会通过内部移位寄存器接收 SDA 线中的字节并将其保存到 DR 寄存器。在每个字节传输结束后，接口都会依次：

1. 发出应答脉冲（如果 ACK 位置 1）
2. RxNE 位置 1 并在 ITEVFEN 和 ITBUFEN 位均置 1 时生成一个中断（参见*图 244* 传输序列 EV7）。

如果在上一次数据接收结束之前 RxNE 位已置 1 但 DR 寄存器中的数据尚未读取，则 BTF 位会由硬件置 1，而接口会一直延长 SCL 低电平，等待 I2C_DR 寄存器被写入，以将 BTF 清零。

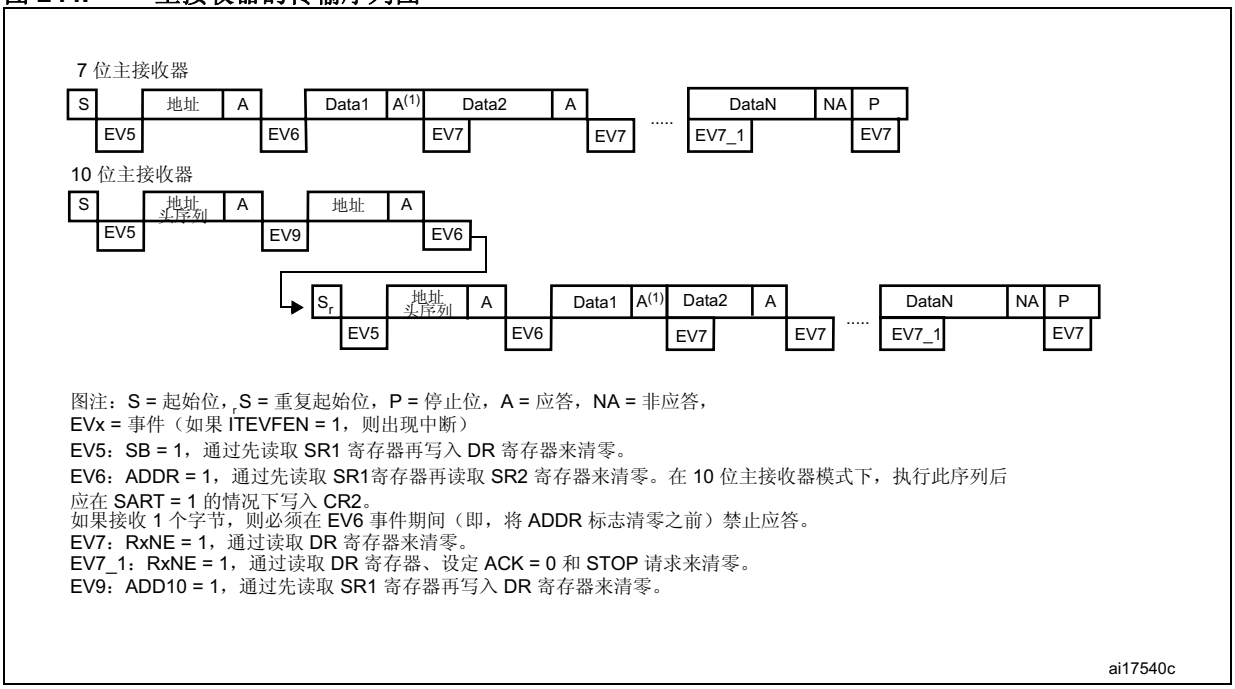
结束通信

主设备会针对自从设备接收的最后一个字节发送 NACK。在接收到此 NACK 之后，从设备会释放对 SCL 和 SDA 线的控制。随后，主设备可发送一个停止位/重复起始位。

1. 为了在最后一个接收数据字节后生成非应答脉冲，必须在读取倒数第二个数据字节后（倒数第二个 RxNE 事件之后）立即将 ACK 位清零。
2. 要生成停止位/重复起始位，软件必须在读取倒数第二个数据字节后（倒数第二个 RxNE 事件之后）将 STOP/START 位置 1。
3. 在只接收单个字节的情况下，会在 EV6 期间（在 ADDR 标志清零之前）禁止应答并在 EV6 之后生成停止位。

生成停止位后，接口会自动返回从模式（M/SL 位清零）。

图 244. 主接收器的传输序列图



1. 如果接收单个字节，则为 NA。
2. EV5、EV6 和 EV9 事件可延长 SCL 低电平时间，直到相应的软件序列结束为止。
3. 如果软件序列在下一个字节接收结束之前未能完成，EV7 事件会延长 SCL 低电平时间。
4. EV7_1 软件序列必须在当前字节传输的 ACK 脉冲之前结束。

如果 EV7_1 软件序列未能在当前字节传输的 ACK 脉冲之前结束，建议采用下列步骤。

必须遵循以下步骤以确保：

- 在结束上一个数据接收之前及时将 ACK 位复位
- 在上一次数据接收后将 STOP 位置位且不接收补充数据。

针对 2 字节的接收：

- 等待 ADDR = 1（SCL 保持低电平，直到 ADDR 标志清零）
- 将 ACK 复位，POS 置位
- 将 ADDR 标志清零
- 等待 BTF = 1（数据 1 在 DR 中，数据 2 在移位寄存器中，SCL 保持低电平，直到读取了数据 1 为止）
- 将 STOP 置位
- 读取数据 1 和数据 2

针对 N > 2 的字节接收，从 N-2 数据接收开始

- 等待 BTF = 1（数据 N-2 在 DR 中，数据 N-1 在移位寄存器中，SCL 保持低电平，直到读取了数据 N-2 为止）
- 将 ACK 复位
- 读取数据 N-2
- 等待 BTF = 1（数据 N-1 在 DR 中，数据 N 在移位寄存器中，SCL 保持低电平，直到读取了数据 N-1 为止）
- 将 STOP 置位
- 读取数据 N-1 和数据 N

25.3.4 错误条件

以下错误条件可能导致通信失败。

总线错误 (BERR)

当 I²C 接口在传输地址或数据期间检测到外部停止位或起始位时，会出现此错误。在这种情况下：

- BERR 位置 1 并在 ITERREN 位置 1 时生成一个中断
- 在从模式下：硬件会丢弃数据并释放数据线：
  - 在误放起始位的情况下，从模式会认为它是重复起始位并会等待一个地址或停止位
  - 在误放停止位的情况下，从模式会按停止位操作且硬件会释放数据线
- 在主模式下：不会释放数据线且不会影响当前传输状态。由软件决定是否中止当前传输

应答失败 (AF)

当接口检测到未应答脉冲会出现此错误。在这种情况下：

- AF 位置 1 并在 ITERREN 位置 1 时生成一个中断
- 接收到 NACK 的发送器必须复位通信：
  - 在从模式下：硬件释放数据线
  - 在主模式下：必须由软件生成一个停止位或重复起始位

仲裁丢失 (ARLO)

当 I²C 接口检测到仲裁丢失时会出现此错误。在这种情况下，

- ARLO 位会由硬件置 1（并在 ITERREN 位置 1 时生成一个中断）
- I²C 接口会自动返回从模式（M/SL 位清零）。I²C 失去仲裁时，将无法在本次传输中对自身从地址进行应答，但是能够在赢得仲裁的主设备发出重复起始位后对自身设备进行应答。
- 硬件释放数据线

上溢/下溢错误 (OVR)

当时钟延长已禁止且 I²C 接口正在接收数据时，从模式中可能出现上溢错误。接口已经收到一个字节 (RxNE=1)，但是收到下一个字节之前 DR 中的数据未被读走。在这种情况下，

- 会丢失接收的最后一个字节。
- 在出现上溢错误时，软件应将 RxNE 位清零，而发送器应重新发送最后一个字节。

当时钟延长已禁止且 I²C 接口正在发送数据时，从模式中可能出现下溢错误。下一个字节的时钟信号到来之前，从设备还未把下一个要发送的数据写进DR（TxE=1）。在这种情况下，

- 会再次发送 DR 寄存器中的字节
- 用户应确保在出现下溢错误期间从接收器端接收到的数据被丢弃，并确保在 I²C 总线标准指定的时钟低电平时间内写入下一个字节。

对于要传送的首个字节，必须在 ADDR 清零之后且出现首个 SCL 上升沿之前将其写入 DR 寄存器。否则，接收器必须丢弃首个数据。

25.3.5 可编程噪声滤波器

可编程噪声滤波器仅适用于 STM32F42xxx 和 STM32F43xxx 器件。

在快速模式下， I²C 标准要求将 SDA 和 SCL 线上尖峰脉宽在 50 ns 以内的噪声都抑止掉。

SDA 和 SCL I/O 中采用了模拟噪声滤波器。此滤波器默认为使能，通过将 I2C_FLTR 寄存器中的 ANOFF 位置 1 可禁止它。

将 DNF[3:0] 位配置为一个非零值可使能数字噪声滤波器。这可以抑制 SDA 和 SCL 输入上脉宽小于 DNF[3:0] * T_{PCLK1} 的噪声。

使能数字噪声滤波器后，SDA 保持时间可增加 (DNF[3:0] +1)* T_{PCLK}。

为了符合 I²C 总线规范版本 2.1 (Thd:dat) 中对最大保持时间的要求，必须使用*表 103* 所示的限制条件对 DNF 位进行编程并假定模拟滤波器已禁止。

注意： *DNF[3:0] 必须在 I²C 已禁止 (PE = 0) 的情况下配置。如果模拟滤波器也已使能，则需将数字滤波器添加到模拟滤波器中。*

表 103. 符合 Thd:dat(max) 的 DNF[3:0] 最大值

PCLK1 频率	DNF 最大值	
	标准模式	快速模式
2 <= F_{PCLK1} <= 5	2	0
5 < F_{PCLK1} <= 10	12	0
10 < F_{PCLK1} <= 20	15	1
20 < F_{PCLK1} <= 30	15	7
30 < F_{PCLK1} <= 40	15	13
40 < F_{PCLK1} <= 50	15	15

注意： *上表根据各个频率范围的最低频率给出了相应的限制条件。如果系统可以支持最大保持时间违例，则可使用更大的 DNF 值。*

25.3.6 SDA/SCL 线控制

- 如果时钟延长已使能：
  - 发送模式：如果 TxE=1 且 BTF=1：接口会在发送数据之前保持时钟线为低电平，以等待微控制器将字节写入数据寄存器（缓冲寄存器和移位寄存器均为空）。
  - 接收模式：如果 RxNE=1 且 BTF=1：接口会在接收数据之后保持时钟线为低电平，以等待微控制器将字节读入数据寄存器（缓冲寄存器和移位寄存器均已满）。
- 如果从模式中的时钟延长已禁止：
  - RxNE=1，且在下一个数据接收完成之前还未读走 DR 中的数据就出现上溢错误。会丢失接收的最后一个字节。
  - TxE=1，且在下一个数据的始终到来之前还未把发送数据写到 DR 中，就出现下溢错误。会再次发送同一字节。
  - 不会对写冲突进行管理。

25.3.7 SMBus

前言

系统管理总线 (SMBus) 是一个双线制接口，各器件可通过它在彼此之间或者与系统的其余部分进行通信。它以 I²C 的工作原理为基础。SMBus 可针对系统和电源管理相关的任务提供控制总线。系统可使用 SMBus 与设备进行消息传递，而无需切换各个控制线。

系统管理总线规范涉及三类器件。*从器件*，用于接收或响应命令。*主器件*，用于发出命令、生成时钟和中止传输。*主机*，专用的主器件，可提供连接系统 CPU 的主接口。主机必须具有主 - 从设备功能，并且必须支持 SMBus 主机通知协议。系统中只允许存在一个主机。

SMBus 与 I²C 的相似之处

- 双线制总线协议（1 个时钟总线，1 个数据总线）+ 可选 SMBus 报警线
- 主从通信，主器件提供时钟
- 多主器件功能
- SMBus 数据格式与 I²C 7 位地址格式相似（*图 238*）。

SMBus 与 I²C 之间的差异

下表介绍了 SMBus 与 I²C 之间的差异。

表 104. SMBus 与 I²C

SMBus	I²C
最大速度 100 kHz	最大速度 400 kHz
最小时钟速度 10 kHz	无最小时钟速度
35 ms 时钟低电平超时	无超时
逻辑电平固定	逻辑电平取决于 V_{DD}
地址类型不同（保留、动态等）	7 位、10 位和广播从模式地址类型
总线协议不同（快速命令、过程调用等）	无总线协议

SMBus 应用用途

通过系统管理总线，器件可以提供制造商信息、告诉系统它的型号/部件号、保存暂停事件的状态、报告不同的错误类型、接受控制参数并返回其状态。SMBus 可针对系统和电源管理相关的任务提供控制总线。

器件标识

系统管理总线中作为从器件的任何器件均具有一个唯一地址，被称为从地址。有关保留的从地址列表的信息，请参见 SMBus 规范版本 2.0 (*http://smbus.org/specs/*)。

总线协议

SMBus 规范支持多达 9 类总线协议。有关这些协议和 SMBus 地址类型的详细信息，请参见 SMBus 规范版本 2.0 (*http://smbus.org/specs/*)。这些协议应通过用户软件实施。

地址解析协议 (ARP)

通过为各个从器件动态分配一个新的唯一地址可解决 SMBus 从地址冲突的问题。地址解析协议 (ARP) 具有以下属性：

- 地址分配采用标准的 SMBus 物理层仲裁机制
- 器件通电时，分配的地址保持不变；器件断电时，也允许保留地址
- 完成地址分配后不会带来额外的 SMBus 数据包开销。（即，对已分配的从地址进行后续访问与访问固定地址的器件所产生的开销相同。）
- 任何 SMBus 主器件都可以遍历总线

唯一的器件标识符 (UDID)

为了提供一种机制来针对地址分配隔离各个器件，各器件必须具有唯一的器件标识符 (UDID)。

有关 128 位 UDID 和 ARP 的详细信息，请参见 SMBus 规范版本 2.0 (*http://smbus.org/specs/*)。

SMBus 报警模式

SMBus 报警是带有中断线的可选信号，主要用于希望扩展它们的控制能力而牺牲一个引脚的器件。SMBA 是与 SCL 和 SDA 信号类似的线与信号。SMBA 与 SMBus 广播地址配合使用。使用 SMBus 调用的消息长度为 2 字节。

将 I2C_CR1 寄存器中的 ALERT 位置 1 后一个只具有从功能的器件可通过 SMBA 向主机发出信号，指示它想要通信。主机会处理该中断并通过 *报警响应地址*（简称 ARA，其值为 0001 100X）同步访问所有 SMBA 器件。只有那些将 SMBA 拉到低电平的器件会确认报警响应地址。通过 I2C_SR1 寄存器中的 SMBALERT 状态标志可确定这一状态。主机会执行修改后的接收字节操作。由从发送器件提供的 7 位器件地址被放置在字节的 7 个最高有效位。第 8 位可以是 0 或 1。

如果有不止一个器件将 SMBA 拉为低电平，则在从地址传输期间，具有最高优先级（最低位地址）的器件会通过标准仲裁获得通信权限。在确认从地址之后，器件必须释放 SMBA。如果消息传输结束后主机检测到 SMBA 仍为低电平，会再次读取 ARA。
未实现 SMBA 信号的主机会定期访问 ARA。

有关 SMBus 报警模式的详细信息，请参见 SMBus 规范版本 2.0 (*http://smbus.org/specs/*)。

超时错误

SMBus 和 I²C 之间存在一些定时规范方面的差异。
SMBus 定义了一个时钟低电平超时，TIMEOUT 为 35 ms。另外，SMBus 还指定 TLOW：SEXT 作为从器件的累积时钟低电平延长时间。SMBus 指定 TLOW：MEXT 作为主器件的累积时钟低电平延长时间。有关这些超时的详细信息，请参见 SMBus 规范版本 2.0 (*http://smbus.org/specs/*)。

I2C_SR1 寄存器中的 Timeout 或 Tlow 错误状态标志表明了此特性的状态。

如何在 SMBus 模式下使用该接口

要从 I²C 模式切换到 SMBus 模式，应执行以下步骤。

- 将 I2C_CR1 寄存器中的 SMBus 位置 1
- 根据应用的要求配置 I2C_CR1 寄存器中的 SMBTYPE 和 ENARP 位

如果要将某个器件配置为主器件，请按照*第 25.3.3 节：I2C 主模式*中介绍的起始位生成步骤进行操作。否则按照*第 25.3.2 节：I2C 从模式*中的顺序操作。

该应用需要通过软件控制各种 SMBus 协议。

- ENARP=1 且 SMBTYPE=0 时使用 SMB 器件默认地址
- ENARP=1 且 SMBTYPE=1 时使用 SMB 主机头字段
- SMBALERT=1 时使用 SMB 报警响应地址

25.3.8 DMA 请求

DMA 请求（在使能后）仅用于数据传输。当发送数据寄存器变空以及接收数据寄存器变满时会生成 DMA 请求。进行 I2C 数据传输之前，必须先初始化并使能 DMA。I2C_CR2 寄存器中的 DMAEN 位必须在 ADDR 事件之前置 1。在主模式或从模式下，如果已使能时钟延长，DMAEN 位也可以在 ADDR 事件期间于 ADDR 标志清零之前置 1。结束当前字节传输之前，必须发出 DMA 请求。当传输的数据量达到相应 DMA 通道编程设定的值时，DMA 控制器会发送一个结束传输 EOT 信号给 I²C 接口，并生成一个传输完成中断（如果已使能）：

- 主发送器：在 EOT 中断后的中断程序中，禁止 DMA 请求，然后在等到 BTF 事件后设置停止位。
- 主接收器
  - 当要接收的字节数等于或大于二时，DMA 控制器会在收到倒数第二个数据字节（第 N - 1 个数据时）发送一个硬件信号 EOT_1。如果 I2C_CR2 寄存器中的 LAST 位置 1，I²C 会在 EOT_1 后的下一个字节之后自动发送一个 NACK。用户可在 DMA 传输完成中断（如果已使能）程序中生成停止位。
  - 当必须接收单个字节时：必须在 EV6 事件期间于 ADDR 标志清零之前对 NACK 进行编程，即当 ADDR=1 时编程设定 ACK=0。接下来，用户可在 ADDR 标志清零之后或者在执行 DMA 传输完成中断程序时编程设定停止位。

使用 DMA 进行发送

将 I2C_CR2 寄存器中的 DMAEN 位置 1 可以使能 DMA 模式进行发送。当 TXE 位置 1 时，数据将由 DMA 从预置的存储区装载进 I2C_DR 寄存器。要映射一个 DMA 通道以便进行 I²C 发送，请按以下步骤操作：其中的 x 表示通道编号。

1. 设置 DMA_CPARx 寄存器中的 I2C_DR 寄存器地址。每次发生 TxE 事件后，数据都会从存储器移动到此地址。
2. 设置 DMA_CMARx 寄存器中的存储器地址。每次发生 TxE 事件后，数据都会从此存储器加载到 I2C_DR。
3. 在 MA_CNDTRx 寄存器中配置要传输的总字节数。在每次 TxE 事件后，此值都会递减。
4. 使用 DMA_CCRx 寄存器中的 PL[0:1] 位来配置通道优先级。
5. 在完成半数传输或全部传输（取决于应用的需求）之后，将 DMA_CCRx 寄存器中的 DIR 位置 1 并配置中断。
6. 将 DMA_CCRx 寄存器中的 EN 位置 1 以激活通道。

当传输的数据量达到 DMA 控制器中编程设定的值时，DMA 控制器会发送一个结束传输 EOT/EOT_1 信号给 I²C 接口，而 DMA 会在 DMA 通道中断向量上生成一个中断（如果已使能）：

注意： *如果使用 DMA 进行传送，请勿使能 I2C_CR2 寄存器中的 ITBUFEN 位。*

使用 DMA 进行接收

将 I2C_CR2 寄存器中的 DMAEN 位置 1 可以使能 DMA 模式进行接收。接收数据字节时，数据会从 I2C_DR 寄存器加载到使用 DMA 外设配置的存储区中（参见 DMA 规范）。要映射一个 DMA 通道以便进行 I²C 接收，请按以下步骤操作：其中的 x 表示通道编号。

1. 设置 DMA_CPARx 寄存器中的 I2C_DR 寄存器地址。每次发生 RxNE 事件后，数据都会从此地址移动到存储器。
2. 设置 DMA_CMARx 寄存器中的存储器地址。每次发生 RXNE 事件后，数据都会从 I2C_DR 寄存器加载到此存储区。
3. 在 MA_CNDTRx 寄存器中配置要传输的总字节数。在每次 RxNE 事件后，此值都会递减。
4. 使用 DMA_CCRx 寄存器中的 PL[0:1] 位来配置通道优先级。
5. 在完成半数传输或全部传输（取决于应用的需求）之后，将 DMA_CCRx 寄存器中的 DIR 位重新置 1 并配置中断。
6. 将 DMA_CCRx 寄存器中的 EN 位置 1 以激活通道。

当传输的数据量达到 DMA 控制器中编程设定的值时，DMA 控制器会发送一个结束传输 EOT/EOT_1 信号给 I²C 接口，而 DMA 会在 DMA 通道中断向量上生成一个中断（如果已使能）：

注意： *如果使用 DMA 进行接收，请勿使能 I2C_CR2 寄存器中的 ITBUFEN 位。*

25.3.9 数据包错误校验

PEC 计算器的应用目的是提高通信的可靠性。PEC 对各个位使用 CRC-8 多项式 $C(x) = x^8 + x^2 + x + 1$ 公式进行串行计算。

- 将 I2C_CR1 寄存器中的 ENPEC 位置 1 即可使能 PEC 计算。PEC 是针对所有消息字节（包括地址和 R/W 位）的 CRC-8 计算。
  - 在发送过程中：在与最后一个字节对应的 TxE 事件发生后，将 I2C_CR1 寄存器中的 PEC 传输位置 1。PEC 会在最后一个传输的字节之后进行传送。

  - 在接收过程中：在与最后一个字节对应的 RxNE 事件发生之后，将 I2C_CR1 寄存器中的 PEC 位置 1，以便接收器在接收到的下一个字节不等于内部计算的 PEC 时发送一个 NACK。在主接收器中，无论校验结果如何，PEC 后都将发送 NACK。在从模式下，必须在 CRC 接收的 ACK 之前设置 PEC。在主模式下，必须在复位 ACK 时设置 PEC 位。
- I2C_SR1 寄存器中还提供 PECERR 错误标志/中断。
- 如果 DMA 和 PEC 计算均已使能：-
  - 在发送过程中：当 I²C 接口接收来自 DMA 控制器的 EOT 信号时，会在最后一个字节之后自动发送 PEC。
  - 在接收过程中：当 I²C 接口接收来自 DMA 控制器的 EOT_1 信号时，会自动将下一个字节视为 PEC 并对其进行校验。在 PEC 接收之后会生成一个 DMA 请求。
- 为了允许进行中间 PEC 传输，可使用 I2C_CR2 寄存器中的一个控制位（LAST 位）来确定它是否真的是最后一个 DMA 传输。如果确实是主接收器的最后一个 DMA 请求，则会在接收最后一个字节后自动发送一个 NACK。
- PEC 计算会因仲裁丢失而失效。

25.4 I²C 中断

下表列出了 I²C 中断请求列表。

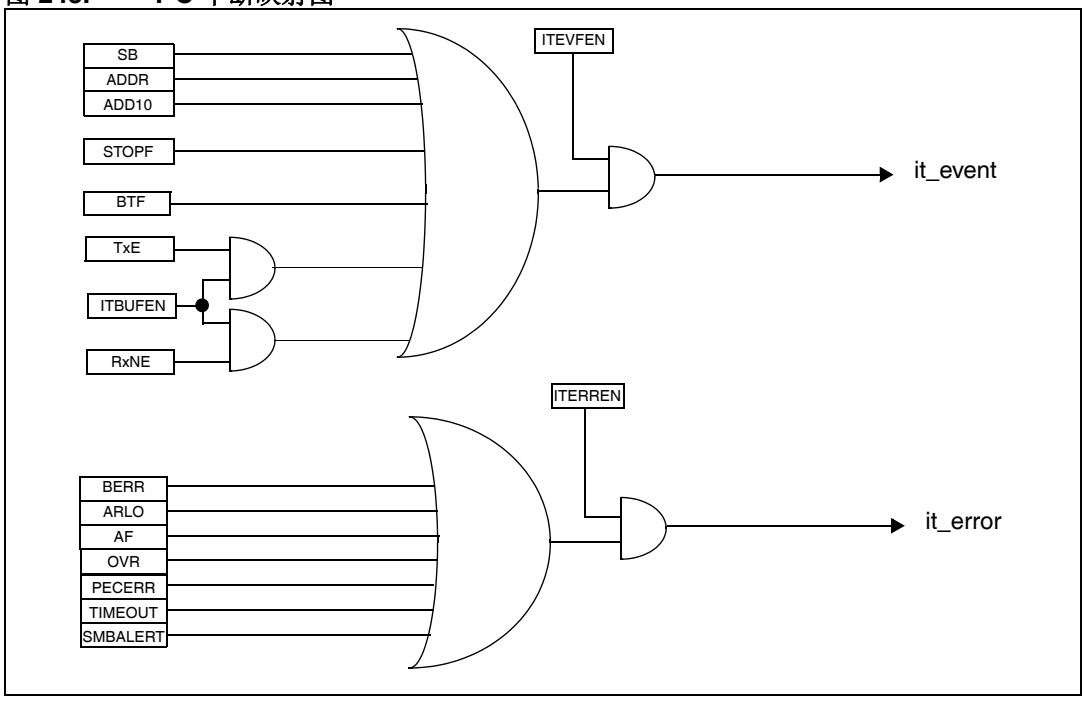
表 105. I²C 中断请求

中断事件	事件标志	使能控制位
发送起始位（主模式）	SB	ITEVFEN
地址已发送（主模式）或地址匹配（从模式）	ADDR	
10 位地址的头段已发送（主模式）	ADD10	
已收到停止位（从模式）	STOPF	
完成数据字节传输	BTF	
接收缓冲区非空	RxNE	ITEVFEN 和 ITBUFEN
发送缓冲区为空	TxE	
总线错误	BERR	ITERREN
仲裁丢失（主模式）	ARLO	
应答失败	AF	
上溢/下溢	OVR	
PEC 错误	PECERR	
超时/Tlow 错误	TIMEOUT	
SMBus 报警	SMBALERT	

注意： *SB、ADDR、ADD10、STOPF、BTF、RxNE 和 TxE 通过逻辑或映射到同一个中断通道上。*

BERR、ARLO、AF、OVR、PECERR、TIMEOUT 和 SMBALERT 通过逻辑或映射到同一个中断通道上。

图 245. I²C 中断映射图



25.5 I²C 调试模式

当微控制器进入调试模式时（Cortex™-M4F 内核停止），SMBUS 超时会根据 DBG 模块中的 DBG_I2Cx_SMBUS_TIMEOUT 配置位选择继续正常工作或者停止工作。有关详细信息，请参见*第 1259 页的第 33.16.2 节：对定时器、看门狗、bxCAN 和 I²C 的调试支持*。

25.6 I²C 寄存器

有关寄存器说明中使用的缩写，请参见*第 47 页的第 1.1 节*。

外设寄存器可支持半字（16 位）或字（32 位）访问。

25.6.1 I²C 控制寄存器 1 (I2C_CR1)

I²C Control register 1
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	NO STRETCH	ENGC	ENPEC	ENARP	SMB TYPE	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)

当置 1 时，I2C 处于复位状态。在复位此位之前，确保 I2C 线已释放且总线空闲。

0：I²C 外设未处于复位状态

1：I²C 外设处于复位状态

注意：当出现错误或锁定状态后，可使用此位重新初始化外设。例如，如果 BUSY 位已置 1 但因母线干扰而不能复位，则可使用 SWRST 位退出此状态。

位 14 保留，必须保持复位值

位 13 **ALERT**：SMBus 报警 (SMBus alert)

此位由软件置 1 和清零，并可在 PE=0 时由硬件清零。

0：释放 SMBA 引脚使其变成高电平。报警响应地址头后跟 NACK。

1：驱动 SMBA 引脚使其变成低电平。报警响应地址头后跟 ACK。

位 12 **PEC**：数据包错误校验 (Packet error checking)

此位由软件置 1 和清零，并可在 PEC 传输完成时由硬件清零，或者在 PE=0 时或在检测到起始位或停止位时由硬件清零。

0：不传输 PEC

1：PEC 传输（在 Tx 或 Rx 模式下）

注意：PEC 计算会因仲裁丢失而失效。

位 11 **POS**：应答/PEC 位置 (Acknowledge/PEC Position)（针对接收数据）

此位由软件置 1 和清零，并可在 PE=0 时由硬件清零。

0：ACK 位控制移位寄存器中当前正在接收的字节的 (N)ACK。PEC 位指示移位寄存器中的当前字节是一个 PEC。

1：ACK 位控制移位寄存器中要接收的下一个字节的 (N)ACK。PEC 位指示移位寄存器的下一个字节是一个 PEC。

注意：POS 位只能用于主设备接收 2 个字节时。它必须在数据开始接收之前进行配置，如第 654 页的主接收器一节中建议的 2 字节接收步骤所述。

位 10 **ACK**：应答使能 (Acknowledge enable)

此位由软件置 1 和清零，并可在 PE=0 时由硬件清零。

0：不返回应答

1：在接收一个字节（匹配地址或数据）之后返回应答

位 9 **STOP**：生成停止位

该位由软件置 1 和清零，也可在检测到停止位时由硬件清零，在检测到超时错误时由硬件置 1。

在主模式下：

0：不生成停止位。

1：在传输当前字节或发送当前起始位后生成停止位。

在从模式下：

0：不生成停止位。

1：完成当前字节传输后释放 SCL 和 SDA 线。

位 8 **START**：生成起始位

此位由软件置 1 和清零，并可在起始位发送完成后或 PE=0 时由硬件清零。

在主模式下：

0：不生成起始位

1：生成重复起始位

在从模式下：

0：不生成起始位

1：在总线空闲时生成起始位

位 7 **NOSTRETCH**：禁止时钟延长 (Clock stretching disable)（从模式）

在从模式下，当 ADDR 或 BTF 标志置 1 时，此位用于禁止时钟延长，直到软件将其复位为止。

0：使能时钟延长

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。不对地址 00h 应答。

1：使能广播呼叫。对地址 00h 应答。

位 5 **ENPEC**：PEC 使能 (PEC enable)

0：禁止 PEC 计算

1：使能 PEC 计算

位 4 **ENARP**：ARP 使能 (ARP enable)

0：禁止 ARP

1：使能 ARP

SMBTYPE=0 时识别 SMBus 器件默认地址

SMBTYPE=1 时识别 SMBus 主机地址

位 3 **SMBTYPE**：SMBus 类型 (SMBus type)

0：SMBus 器件

1：SMBus 主机

位 2 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**：外设使能 (Peripheral enable)

0：禁止外设

1：使能外设

注意：如果此位在通信进行过程中复位，在结束本次通信后会带 IDLE 状态，外设被禁止。由于通信结束时 PE=0，所有位均会复位。

在主模式下，此位不能在通信结束之前复位。

注意：STOP、START 或 PEC 位置 1 之后，在硬件将该位清零之前，软件不能对 I2C_CR1 执行任何写操作。否则，可能会再次发出 STOP、START 或 PEC 置位请求。

25.6.2 I²C 控制寄存器 2 (I2C_CR2)

I²C Control register 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
Reserved			LAST	DMA EN	ITBUF EN	ITEVT EN	ITERR EN	Reserved		FREQ[5:0]					
			rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw

位 15:13 保留，必须保持复位值

位 12 **LAST**：最后一次 DMA 传输 (DMA last transfer)

0：下一个 DMA EOT 不是最后一次传输
1：下一个 DMA EOT 是最后一次传输

注意：此位用于主接收模式，可对最后接收的数据生成 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 时使能 DMA 请求

位 10 **ITBUFEN**：缓冲中断使能 (Buffer interrupt enable)

0：TxE = 1 或 RxNE = 1 时不生成任何中断。
1：TxE = 1 或 RxNE = 1 时生成事件中断（与 DMAEN 状态无关）

位 9 **ITEVTEN**：事件中断使能 (Event interrupt enable)

0：禁止事件中断
1：使能事件中断

满足以下条件时将生成此中断：

–SB = 1（主模式）
–ADDR = 1（主/从模式）
–ADD10= 1（主模式）
–STOPF = 1（从模式）
–BTF = 1，无 TxE 或 RxNE 事件
–ITBUFEN = 1 且 TxE 事件置 1
–ITBUFEN = 1 且 RxNE 事件置 1

位 8 **ITERREN**：错误中断使能 (Error interrupt enable)

0：禁止错误中断
1：使能错误中断

满足以下条件时将生成此中断：

– BERR = 1
– ARLO = 1
– AF = 1
– OVR = 1
– PECERR = 1
– TIMEOUT = 1
– SMBALERT = 1

位 7:6 保留，必须保持复位值

位 5:0 **FREQ[5:0]**：外设时钟频率 (Peripheral clock frequency)

外设时钟频率必须使用 APB 时钟频率进行配置（I2C 外设连接到 APB）。允许的最小频率为 2 MHz，最大频率则受限于 APB 最大频率 (42 MHz) 和固有极限频率 46 MHz。

0b000000：不允许
0b000001：不允许
0b000010：2 MHz
...
0b101010：42 MHz
大于 0b101010：不允许

25.6.3 I²C 自有地址寄存器 1 (I2C_OAR1)

I²C Own address register 1
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
ADD MODE	Reserved					ADD[9:8]		ADD[7:1]							ADD0
rw						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 15 **ADDMODE** 寻址模式 (Addressing mode)（从模式）
0：7 位从地址（无法应答 10 位地址）
1：10 位从地址（无法应答 7 位地址）

位 14 应通过软件始终保持为 1。

位 13:10 保留，必须保持复位值

位 9:8 **ADD[9:8]**：接口地址
7 位寻址模式：无意义
10 位寻址模式：地址的第 9:8 位

位 7:1 **ADD[7:1]**：接口地址
地址的第 7:1 位

位 0 **ADD0**：接口地址
7 位寻址模式：无意义
10 位寻址模式：地址的第 0 位

25.6.4 I²C 自有地址寄存器 2 (I2C_OAR2)

I²C Own address register 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
Reserved								ADD2[7:1]							ENDUAL
								rw	rw	rw	rw	rw	rw	rw	rw

位 15:8 保留，必须保持复位值

位 7:1 **ADD2[7:1]**：接口地址
双寻址模式下的地址第 7:1 位

位 0 **ENDUAL**：双寻址模式使能 (Dual addressing mode enable)
0：7 位寻址模式下仅对 OAR1 地址响应
1：7 位寻址模式下能对 OAR1 和 OAR2 两个地址响应

25.6.5 I²C 数据寄存器 (I2C_DR)

I²C Data register
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
Reserved								DR[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

位 15:8 保留，必须保持复位值

位 7:0 **DR[7:0]** 8 位数据寄存器

接收的字节或者要发送到总线的字节。

–发送模式：在 DR 寄存器中写入第一个字节时自动开始发送字节。如果在启动传送 (TxE=1) 后立即将下一个要传送的数据置于 DR 中，则可以保持连续的传送流

–接收模式：将接收到的字节复制到 DR 中 (RxNE=1)。如果在接收下一个数据字节 (RxNE=1) 之前读取 DR，则可保持连续的传送流。

注意：在从模式下，地址并不会复制到 DR 中。

注意：硬件不对写冲突进行管理（TxE=0 时也可对 DR 执行写操作）。

注意：如果发出 ACK 脉冲时出现 ARLO 事件，则不会将接收到的字节复制到 DR 寄存器，因而也无法读取字节。

25.6.6 I²C 状态寄存器 1 (I2C_SR1)

I²C Status register 1
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
SMB ALERT	TIME OUT	Res.	PEC ERR	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 报警 (SMBus alert)

在 SMBus 主机模式下：
0：无 SMBALERT
1：引脚上发生 SMBALERT 事件
在 SMBus 从模式下：
0：无 SMBALERT 响应地址头
1：接收到指示 SMBALERT 低电平的 SMBALERT 响应地址头
– 由软件写入 0 来清零，或在 PE=0 时由硬件清零。

位 14 **TIMEOUT**：超时或 Tlow 错误 (Timeout or Tlow error)

0：无超时错误
1：SCL 低电平时长持续 25 ms（超时）
或
主器件累计时钟低电平延长时间超过 10 ms (Tlow:mext)
或
从器件累计时钟低电平延长时间超过 25 ms (Tlow:sext)

– 在从模式下置 1 时：从器件复位通信且硬件释放数据线
– 在主模式下置 1 时：由硬件发送停止位
– 由软件写入 0 来清零，或在 PE=0 时由硬件清零。

注意：此功能仅在 SMBus 模式下可用。

位 13 保留，必须保持复位值

位 12 **PECERR**：接收期间 PEC 错误 (PEC Error in reception)

0：无 PEC 错误：接收器在接收 PEC 后返回 ACK（如果 ACK=1）

1：PEC 错误：接收器在接收 PEC 后返回 NACK（无论 ACK 什么值）

–由软件写入 0 来清零，或在 PE=0 时由硬件清零。

–*注意：接收到错误的 CRC 时，如果在结束 CRC 接收之前 PEC 控制位没有置 1，则 PECERR 位在从模式下不会置 1。不过可以通过读取 PEC 值来判定接收到的 CRC 是否正确。*

位 11 **OVR**：上溢/下溢 (Overrun/Underrun)

0：未发生上溢/下溢
1：上溢或下溢

–在从模式下由硬件置 1，前提是满足 NOSTRETCH=1 且：
–接收过程中接收到一个新字节（包括 ACK 脉冲）但尚未读取 DR 寄存器。新接收的字节将丢失。
–发送过程中将发送一个新字节但尚未向 DR 寄存器写入数据。同一字节发送两次。
–由软件写入 0 来清零，或在 PE=0 时由硬件清零。

注意：如果 DR 写操作时间与出现 SCL 上升沿的时间非常接近，则发出的数据不确定，并且出现数据保持时间错误。

位 10 **AF**：应答失败 (Acknowledge failure)

0：未发生应答失败
1：应答失败

–无应答返回时由硬件置 1。
–由软件写入 0 来清零，或在 PE=0 时由硬件清零。

位 9 **ARLO**：仲裁丢失 (Arbitration lost)（主模式）

0：未检测到仲裁丢失
1：检测到仲裁丢失

当接口在竞争总线是输给另一个主设备时，由硬件将该位置 1

–由软件写入 0 来清零，或在 PE=0 时由硬件清零。

发生 ARLO 事件后，接口会自动切换回从模式 (M/SL=0)。

注意：在 SMBUS 中，从模式下的数据仲裁仅发生在数据阶段或发送确认期间（不适用于地址确认）。

位 8 **BERR**：总线错误 (Bus error)

0：无误放的起始或停止位

1：存在误放的起始或停止位

–SCL 为高电平时，若接口在字节传输期间检测到某个无效位置出现 SDA 上升沿或下降沿，则会由硬件将该位置 1。

–由软件写入 0 来清零，或在 PE=0 时由硬件清零。

位 7 **TxE**：数据寄存器为空 (Data register empty)（发送器）

0：数据寄存器非空

1：数据寄存器为空

–发送过程中 DR 为空时该位置 1。TxE 不会在地址阶段置 1。

–由软件写入 DR 寄存器来清零，或在出现起始、停止位或者 PE=0 时由硬件清零。

如果接收到 NACK 或要发送的下一个字节为 PEC (PEC=1)，TxE 将不会置 1

注意：写入第一个要发送的数据或在 BTF 置 1 时写入数据都无法将 TxE 清零，因为这两种情况下数据寄存器仍为空。

位 6 **RxNE**：数据寄存器非空 (Data register not empty)（接收器）

0：数据寄存器为空

1：数据寄存器非空

–接收模式下数据寄存器非空时置 1。RxNE 不会在地址阶段置 1。

–由软件读取或写入 DR 寄存器来清零，或在 PE=0 时由硬件清零。

发生 ARLO 事件时 RxNE 不会置 1。

注意：BTF 置 1 时无法通过读取数据将 RxNE 清零，因为此时数据寄存器仍为满。

位 5 保留，必须保持复位值

位 4 **STOPF**：停止位检测 (Stop detection)（从模式）

0：未检测到停止位

1：检测到停止位

–从设备在应答脉冲后（如果 ACK=1）检测到停止位，由硬件置 1。

–由软件分别对 SR1 寄存器和 CR1 寄存器执行读操作和写操作来清零，或在 PE=0 时由硬件清零。

注意：收到 NACK 后 STOPF 位不会置 1。
建议在 STOPF 置 1 后执行完整的清零序列（首先读取 SR1，然后写入 CR1）。请参见第 652 页的图 242：从接收器的传输序列图。

位 3 **ADD10**：发送 10 位头（主模式）

0：未发生 ADD10 事件。

1：主器件已发送第一个地址字节（头）。

–主器件在 10 位地址模式下已发送第一个字节时由硬件置 1。

–由软件在读取 SR1 寄存器后在 DR 寄存器中写入第二个地址字节来清零，或在 PE=0 时由硬件清零。

注意：收到 NACK 后 ADD10 位不会置 1

位 2 **BTF**：字节传输完成 (Byte transfer finished)

0：数据字节传输未完成

1：数据字节传输成功完成

–由硬件置 1，前提是满足 NOSTRETCH=0 且：

–接收过程中接收到一个新字节（包括 ACK 脉冲）但尚未读取 DR 寄存器 (RxNE=1)。

–发送过程中将发送一个新字节但尚未向 DR 寄存器写入数据 (TxE=1)。

–由软件读或写 DR 寄存器来清零，或在发送过程中出现起始或停止位后由硬件清零，也可以在 PE=0 时由硬件清零。

注意：收到 NACK 后 BTF 位不会置 1

如果下一个要发送的字节为 PEC（I2C_SR2 寄存器中的 TRA=1，I2C_CR1 寄存器中的 PEC=1），则 BTF 位不会置 1

位 1 **ADDR**：地址已发送（主模式）/地址匹配（从模式）

由软件在读取 SR1 寄存器后读取 SR2 寄存器来清零，或在 PE=0 时由硬件清零。

地址匹配（从模式）：

0：地址不匹配或未接收到地址。

1：接收到的地址匹配。

–当接收到的从地址与 OAR 寄存器内容、广播呼叫地址或 SMBus 器件默认地址匹配时，或者识别到 SMBus 主机或 SMBus 报警时，该位由硬件置 1。（根据配置确定何时使能）。

注意：在从模式下，建议在 ADDR 置 1 后执行完整的清零序列（首先读取 SR1，然后写入 SR2）。请参见第 652 页的图 242：从接收器的传输序列图。

地址已发送（主模式）

0：地址发送未结束

1：地址发送结束

–在 10 位寻址模式下，接收到第二个地址字节的 ACK 后该位置 1。

–在 7 位寻址模式下，接收到地址字节的 ACK 后该位置 1。

注意：收到 NACK 后 ADDR 位不会置 1

位 0 **SB**：起始位 (Start bit)（主模式）

0：无起始位

1：起始位已经发送。

–生成启动条件时置 1。

–由软件在读取 SR1 寄存器后写入 DR 寄存器来清零，或在 PE=0 时由硬件清零

25.6.7 I²C 状态寄存器 2 (I2C_SR2)

I²C Status register 2

偏移地址：0x18

复位值：0x0000

注意：读取 I2C_SR1 后再读取 I2C_SR2 可将 ADDR 标志清零，即使 ADDR 标志在读取 I2C_SR1 之后置 1 也如此。因此，必须仅在 I2C_SR1 中的 ADDR 位已置 1 或者 STOPF 位已清零后读取 I2C_SR2。

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PEC[7:0]								DUALF	SMB HOST	SMBDE FAULT	GEN CALL	Res.	TRA	BUSY	MSL
r	r	r	r	r	r	r	r	r	r	r	r		r	r	r

位 15:8 **PEC[7:0]** 数据包错误校验寄存器

ENPEC=1 时，此寄存器包含内部 PEC。

位 7 **DUALF**：双标志 (Dual flag)（从模式）

0：接收到的地址与 OAR1 匹配
1：接收到的地址与 OAR2 匹配
–出现停止位、重复起始位或 PE=0 时由硬件清零。

位 6 **SMBHOST**：SMBus 主机头 (SMBus host header)（从模式）

0：无 SMBus 主机地址
1：SMBTYPE=1 且 ENARP=1 时接收到 SMBus 主机地址。
–出现停止位、重复起始位或 PE=0 时由硬件清零。

位 5 **SMBDEFAULT**：SMBus 器件默认地址 (SMBus device default address)（从模式）

0：无 SMBus 器件默认地址
1：ENARP=1 时接收到 SMBus 器件默认地址
–出现停止位、重复起始位或 PE=0 时由硬件清零。

位 4 **GENCALL**：广播呼叫地址 (General call address)（从模式）

0：无广播呼叫
1：ENGC=1 时接收到广播呼叫地址
–出现停止位、重复起始位或 PE=0 时由硬件清零。

位 3 保留，必须保持复位值

位 2 **TRA**：发送器/接收器 (Transmitter/receiver)

0：接收器
1：发送器

此位在整个地址阶段的结尾处根据地址字节的 R/W 位状态进行置 1。

同样，检测到停止位 (STOPF=1)、重复起始位、总线仲裁丢失 (ARLO=1) 或当 PE=0 时该位也由硬件清零。

位 1 **BUSY**：总线忙碌 (Bus busy)

0：总线上无通信
1：总线正在进行通信
–检测到 SDA 或 SCL 低电平时由硬件置 1
–检测到停止位时由硬件清零。

该位指示总线上是否正在进行通信。即使禁止接口 (PE=0) 后此信息也会更新。

位 0 **MSL**：主/从模式 (Master/slave)

0：从模式
1：主模式
–接口进入主模式时 (SB=1) 由硬件置 1。
–检测到总线上的停止位、仲裁丢失 (ARLO=1) 或当 PE=0 时由硬件清零。

注意： *读取 I2C_SR1 后再读取 I2C_SR2 可将 ADDR 标志清零，即使 ADDR 标志在读取 I2C_SR1 之后置 1 也如此。因此，必须仅在 I2C_SR1 中的 ADDR 位已置 1 或者 STOPF 位已清零后读取 I2C_SR2。*

25.6.8 I²C 时钟控制寄存器 (I2C_CCR)

I²C Clock control register
偏移地址：0x1C
复位值：0x0000

注意： *f_{PCLK1} 必须至少为 2 MHz，才能达到标准模式 I²C 频率。必须至少为 4 MHz 才能达到快速模式 I²C 频率。必须为 10MHz 的倍数才能实现最大 400 kHz 的 I²C 快速模式时钟。*

CCR 寄存器必须仅在禁止 I2C (PE = 0) 的情况下配置。

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
F/S	DUTY	Reserved		CCR[11:0]											
rw	rw			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 15 **F/S**：I2C 主模式选择 (I2C master mode selection)
0：标准模式 I2C
1：快速模式 I2C

位 14 **DUTY**：快速模式占空比 (Fast mode duty cycle)
0：快速模式 t_{low}/t_{high} = 2
1：快速模式 t_{low}/t_{high} = 16/9（参见 CCR）

位 13:12 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寄存器 (Clock control register in Fast/Standard mode)（主模式）

控制主模式下的 SCL 时钟。

标准模式或 SMBus 模式：

$T_{high} = CCR * T_{PCLK1}$
$T_{low} = CCR * T_{PCLK1}$

快速模式：

如果 DUTY = 0：
$T_{high} = CCR * T_{PCLK1}$
$T_{low} = 2 * CCR * T_{PCLK1}$
如果 DUTY = 1：（达到 400 kHz）
$T_{high} = 9 * CCR * T_{PCLK1}$
$T_{low} = 16 * CCR * T_{PCLK1}$
例如：要在标准模式下生成 100 kHz 的 SCL 频率：

如果 FREQR = 08，T_{PCLK1} = 125 ns，则必须将 CCR 编程为 0x28

(0x28 <=> 40d x 125 ns = 5000 ns)。

注意： *1. 允许的最小值为 0x04，但快速占空比模式例外，其最小值为 0x01*
.这些时间均未经过滤波。
.CCR 寄存器必须仅在禁止 I²C (PE = 0) 的情况下配置。

25.6.9 I²C TRISE 寄存器 (I2C_TRISE)

I²C TRISE register
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
Reserved										TRISE[5:0]					
										rw	rw	rw	rw	rw	rw

位 15:6 保留，必须保持复位值

位 5:0 **TRISE[5:0]**：快速/标准模式下的最大上升时间 (Maximum rise time in Fast/Standard mode)（主模式）

这些位必须编程为 I²C 总线规范中给定的最大 SCL 上升时间加 1。

例如：标准模式下允许的最大 SCL 上升时间为 1000 ns。

如果 I2C_CR2 寄存器中 FREQ[5:0] 位的值等于 0x08 且 T_{PCLK1} = 125 ns，则 TRISE[5:0] 位必须编程为 09h。

(1000 ns / 125 ns = 8 + 1)

滤波器值也可以叠加到 TRISE[5:0]。

如果结果不为整数，则 TRISE[5:0] 必须编程为整数部分，以符合 t_{HIGH} 参数要求。

注意：TRISE[5:0] 必须仅在禁止 I2C (PE = 0) 的情况下配置。

25.6.10 I²C FLTR 寄存器 (I2C_FLTR)

I²C FLTR register
偏移地址：0x24
复位值：0x0000

I2C_FLTR 寄存器仅适用于 STM32F42xxx 和 STM32F43xxx。

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

位 15:5 保留，必须保持复位值

位 4 **ANOFF**：模拟噪声滤波器关闭 (Analog noise filter OFF)

0：使能模拟噪声滤波器

1：禁止模拟噪声滤波器

注意：ANOFF 必须仅在禁止 I2C (PE = 0) 的情况下配置。

位 3:0 **DNF[3:0]**：数字噪声滤波器 (Digital noise filter)

这些位用于配置 SDA 和 SCL 输入端的数字噪声滤波器。数字滤波器可抑制脉宽达 DNF[3:0] * TPCLK1 以下的尖峰。

0000：禁止数字噪声滤波器

0001：使能数字噪声滤波器，滤波能力可达 1* TPCLK1。

...

1111：使能数字噪声滤波器，滤波能力可达 15* TPCLK1。

注意：DNF[3:0] 必须仅在禁止 I2C (PE = 0) 的情况下配置。如果模拟滤波器也已使能，则需将数字滤波器添加到模拟滤波器中。

25.6.11 I²C 寄存器映射

下表提供了 I²C 寄存器映射和复位值。

表 106. I²C 寄存器映射和复位值

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**I2C_CR1**	Reserved																SWRST	Reserved	ALERT	PEC	POS	ACK	STOP	START	NOSTRETCH	ENGC	ENPEC	ENARP	SMBTYPE	Reserved	SMBus	PE
	Reset value																	0		0	0	0	0	0	0	0	0	0	0	0		0	0
0x04	**I2C_CR2**	Reserved																			LAST	DMAEN	ITBUFEN	ITEVTEN	ITERREN	Reserved	FREQ[5:0]						
	Reset value																				0	0	0	0	0		0	0	0	0	0	0	0
0x08	**I2C_OAR1**	Reserved																ADDMODE	Reserved					ADD[9:8]		ADD[7:1]							ADD0
	Reset value																	0						0	0	0	0	0	0	0	0	0	0
0x0C	**I2C_OAR2**	Reserved																								ADD2[7:1]							ENDUAL
	Reset value																									0	0	0	0	0	0	0	0
0x10	**I2C_DR**	Reserved																								DR[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x14	**I2C_SR1**	Reserved																SMBALERT	TIMEOUT	Reserved	PECERR	OVR	AF	ARLO	BERR	TxE	RxNE	Reserved	STOPF	ADD10	BTF	ADDR	SB
	Reset value																	0	0		0	0	0	0	0	0	0		0	0	0	0	0
0x18	**I2C_SR2**	Reserved																PEC[7:0]								DUALF	SMBHOST	SMBDEFAULT	GENCALL	Reserved	TRA	BUSY	MSL
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0
0x1C	**I2C_CCR**	Reserved																F/S	DUTY	Reserved	CCR[11:0]												
	Reset value																	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0
0x20	**I2C_TRISE**	Reserved																										TRISE[5:0]					
	Reset value																											0	0	0	0	1	0
0x24	**I2C_FLTR**	Reserved																											ANOFF	DNF[3:0]			
	Reset value																												0	0	0	0	0

有关寄存器边界地址表，请参见*第 52 页的表 2*。

26 通用同步异步收发器 (USART)

除非特别说明，否则本部分适用于整个 STM32F4xx 系列。

26.1 USART 简介

通用同步异步收发器 (USART) 能够灵活地与外部设备进行全双工数据交换，满足外部设备对工业标准 NRZ 异步串行数据格式的要求。USART 通过小数波特率发生器提供了多种波特率。

它支持同步单向通信和半双工单线通信；还支持 LIN（局域互连网络）、智能卡协议与 IrDA（红外线数据协会）SIR ENDEC 规范，以及调制解调器操作 (CTS/RTS)。而且，它还支持多处理器通信。

通过配置多个缓冲区使用 DMA 可实现高速数据通信。

26.2 USART 主要特性

- 全双工异步通信
- NRZ 标准格式（标记/空格）
- 可配置为 16 倍过采样或 8 倍过采样，因而为速度容差与时钟容差的灵活配置提供了可能
- 小数波特率发生器系统
  - 通用可编程收发波特率（有关最大 APB 频率时的波特率值，请参见数据手册）。
- 数据字长度可编程（8 位或 9 位）
- 停止位可配置 - 支持 1 或 2 个停止位
- LIN 主模式同步停止符号发送功能和 LIN 从模式停止符号检测功能
  - 对 USART 进行 LIN 硬件配置时可生成 13 位停止符号和检测 10/11 位停止符号
- 用于同步发送的发送器时钟输出
- IrDA SIR 编码解码器
  - 正常模式下，支持 3/16 位持续时间
- 智能卡仿真功能
  - 智能卡接口支持符合 ISO 7816-3 标准中定义的异步协议智能卡
  - 智能卡工作模式下，支持 0.5 或 1.5 个停止位
- 单线半双工通信
- 使用 DMA（直接存储器访问）实现可配置的多缓冲区通信
  - 使用 DMA 在预留的 SRAM 缓冲区中收/发字节
- 发送器和接收器具有单独使能位
- 传输检测标志：
  - 接收缓冲区已满
  - 发送缓冲区为空
  - 传输结束标志
- 奇偶校验控制：
  - 发送奇偶校验位
  - 检查接收的数据字节的奇偶性

- 四个错误检测标志：
  - 溢出错误
  - 噪声检测
  - 帧错误
  - 奇偶校验错误
- 十个具有标志位的中断源：
  - CTS 变化
  - LIN 停止符号检测
  - 发送数据寄存器为空
  - 发送完成
  - 接收数据寄存器已满
  - 接收到线路空闲
  - 溢出错误
  - 帧错误
  - 噪声错误
  - 奇偶校验错误
- 多处理器通信 - 如果地址不匹配，则进入静默模式
- 从静默模式唤醒（通过线路空闲检测或地址标记检测）
- 两个接收器唤醒模式：地址位（MSB，第 9 位），线路空闲

26.3 USART 功能说明

接口通过三个引脚从外部连接到其它设备（请参见*图 246*）。任何 USART 双向通信均需要至少两个引脚：接收数据输入引脚 (RX) 和发送数据引脚输出 (TX)：

RX：接收数据输入引脚就是串行数据输入引脚。过采样技术可区分有效输入数据和噪声，从而用于恢复数据。

TX：发送数据输出引脚。如果关闭发送器，该输出引脚模式由其 I/O 端口配置决定。如果使能了发送器但没有待发送的数据，则 TX 引脚处于高电平。在单线和智能卡模式下，该 I/O 用于发送和接收数据（USART 电平下，随后在 SW_RX 上接收数据）。

正常 USART 模式下，通过这些引脚以帧的形式发送和接收串行数据：

- 发送或接收前保持空闲线路
- 起始位
- 数据（字长 8 位或 9 位），最低有效位在前
- 用于指示帧传输已完成的 0.5 个、1 个、1.5 个、2 个停止位
- 该接口使用小数波特率发生器 - 带 12 位尾数和 4 位小数
- 状态寄存器 (USART_SR)
- 数据寄存器 (USART_DR)
- 波特率寄存器 (USART_BRR) - 12 位尾数和 4 位小数。
- 智能卡模式下的保护时间寄存器 (USART_GTPR)。

有关各个位的定义，请参见*第 711 页的第 26.6 节：USART 寄存器*。

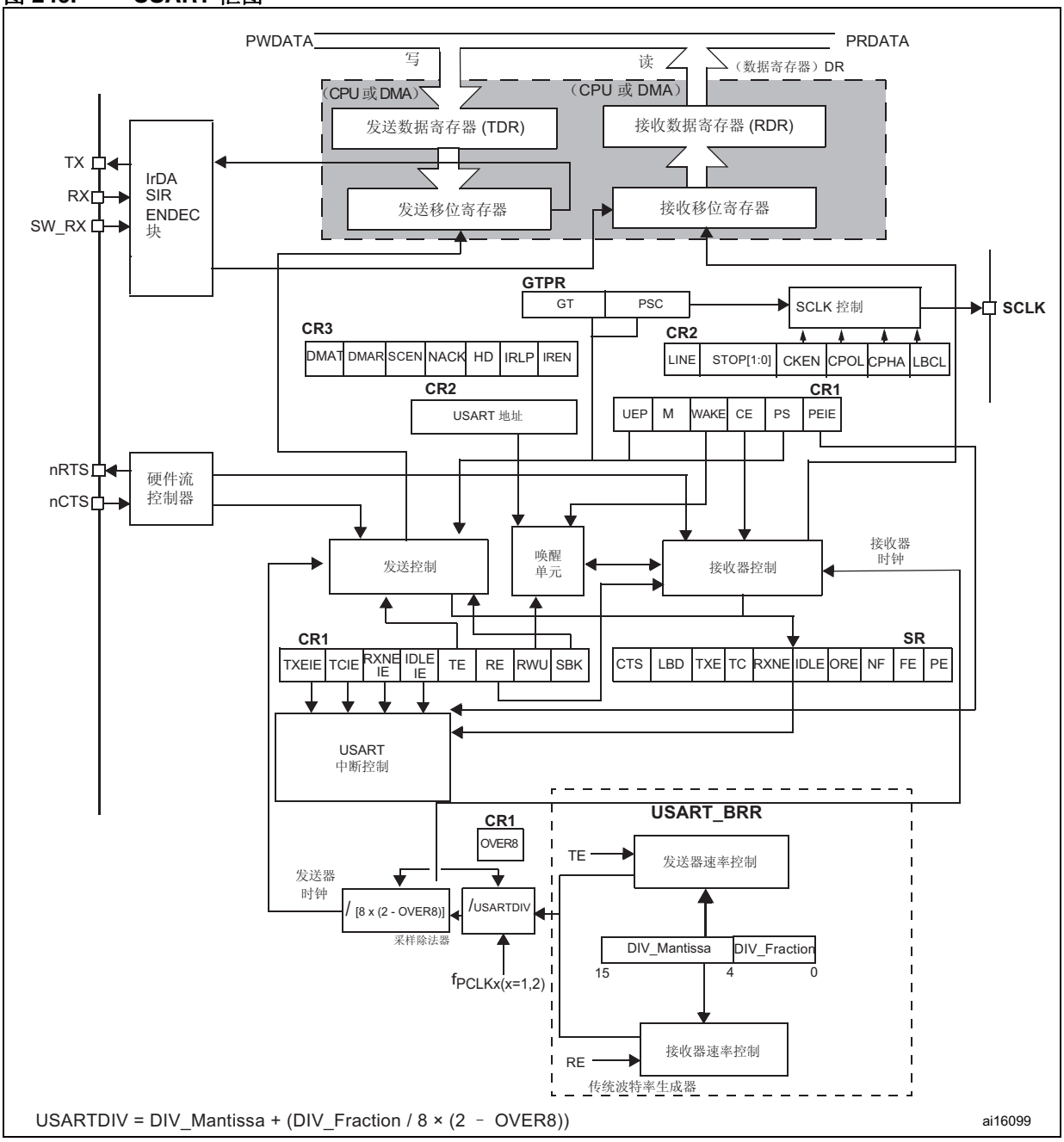
在同步模式下连接时需要以下引脚：

- **SCLK**：发送器时钟输出。该引脚用于输出发送器数据时钟，以便按照 SPI 主模式进行同步发送（起始位和结束位上无时钟脉冲，可通过软件向最后一个数据位发送时钟脉冲）。RX 上可同步接收并行数据。这一点可用于控制带移位寄存器的外设（如 LCD 驱动器）。时钟相位和极性可通过软件编程。在智能卡模式下，SCLK 可向智能卡提供时钟。

在硬件流控制模式下需要以下引脚：

- **nCTS**：“清除以发送”用于在当前传输结束时阻止数据发送（高电平时）
- **nRTS**：“请求以发送”用于指示 USART 已准备好接收数据（低电平时）。

图 246. USART 框图



26.3.1 USART 字符说明

可通过对 USART_CR1 寄存器中的 M 位进行编程来选择 8 位或 9 位的字长（请参见*图 247*）。

TX 引脚在起始位工作期间处于低电平状态。在停止位工作期间处于高电平状态。

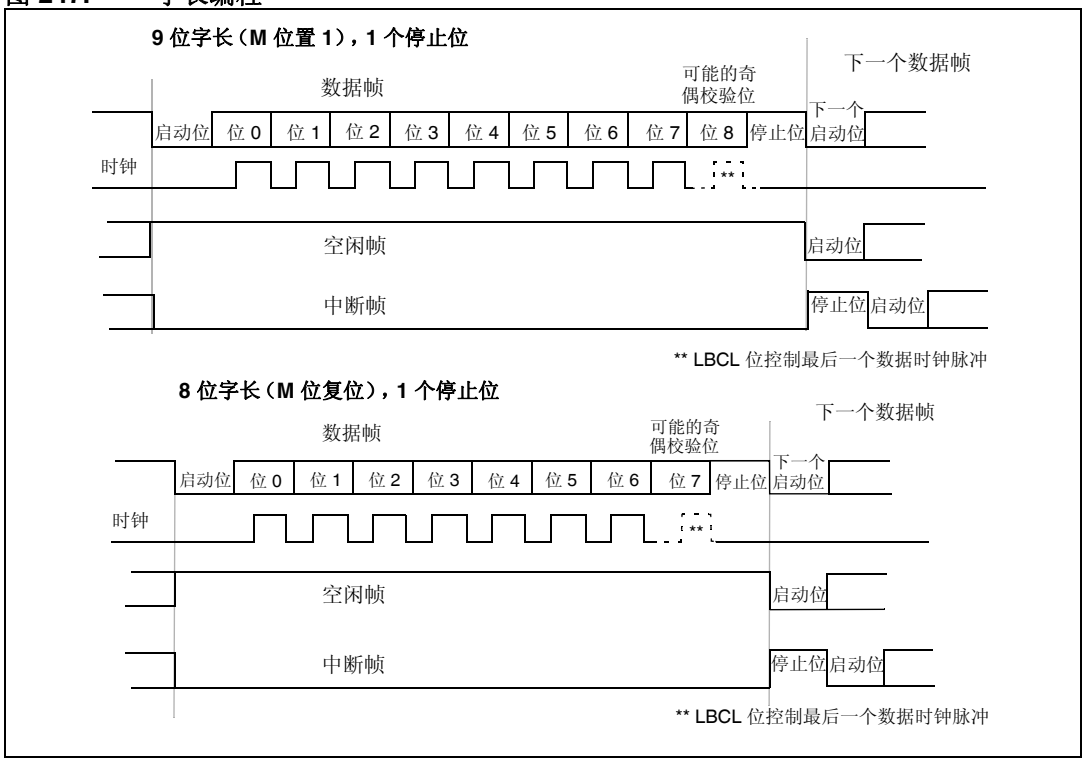
*空闲字符*可理解为整个帧周期内电平均为“1”（停止位的电平也是“1”），该字符后是下一个数据帧的起始位。

*停止字符*可理解为在一个帧周期内接收到的电平均为“0”。发送器在中断帧的末尾插入 1 或 2 个停止位（逻辑“1”位）以确认起始位。

发送和接收由通用波特率发生器驱动，发送器和接收器的使能位分别置 1 时将生成相应的发送时钟和接收时钟。

下面给出了各个块的详细信息。

图 247. 字长编程



26.3.2 发送器

发送器可发送 8 位或 9 位的数据字，具体取决于 M 位的状态。发送使能位 (TE) 置 1 时，发送移位寄存器中的数据在 TX 引脚输出，相应的时钟脉冲在 SCLK 引脚输出。

字符发送

USART 发送期间，首先通过 TX 引脚移出数据的最低有效位。该模式下，USART_DR 寄存器的缓冲区 (TDR) 位于内部总线和发送移位寄存器之间（请参见*图 246*）。

每个字符前面都有一个起始位，其逻辑电平在一个位周期内为低电平。字符由可配置数量的停止位终止。

USART 支持以下停止位：0.5、1、1.5 和 2 个停止位。

注意： *数据发送期间不应复位 TE 位。发送期间复位 TE 位会冻结波特率计数器，从而将损坏 TX 引脚上的数据。当前传输的数据将会丢失。*

使能 TE 位后，将会发送空闲帧。

可配置的停止位

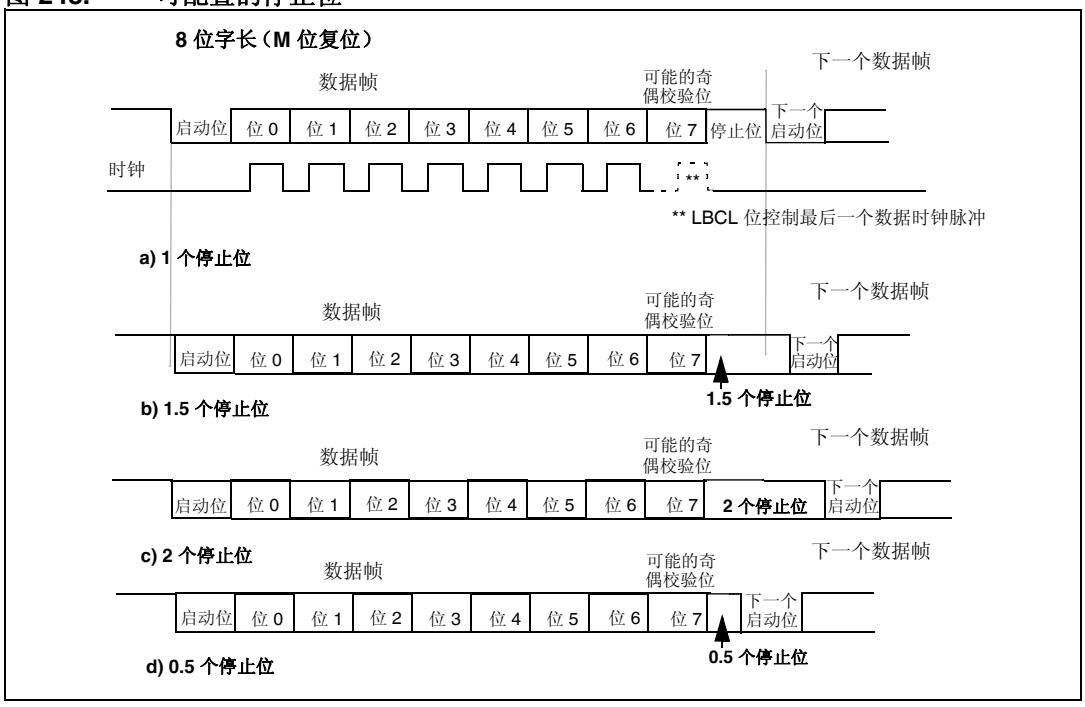
可以在控制寄存器 2 的位 13 和 位 12 中编程将随各个字符发送的停止位的数量。

- ***1 个停止位***：这是停止位数量的默认值。
- ***2 个停止位***：正常 USART 模式、单线模式和调制解调器模式支持该值。
- ***0.5 个停止位***：在智能卡模式下接收数据时使用。
- ***1.5 个停止位***：在智能卡模式下发送和接收数据时使用。

空闲帧发送将包括停止位。

m = 0 时，中断发送是 10 个低电平位，然后是已配置数量的停止位；m = 1 时，中断发送是 11 个低电平位，然后是已配置数量的停止位。无法传送长中断（中断长度大于 10/11 个低电平位）。

图 248.　可配置的停止位



步骤：

1. 通过向 USART_CR1 寄存器中的 UE 位写入 1 使能 USART。
2. 对 USART_CR1 中的 M 位进行编程以定义字长。
3. 对 USART_CR2 中的停止位数量进行编程。
4. 如果将进行多缓冲区通信，请选择 USART_CR3 中的 DMA 使能 (DMAT)。按照多缓冲区通信中的解释说明配置 DMA 寄存器。
5. 使用 USART_BRR 寄存器选择所需波特率。
6. 将 USART_CR1 中的 TE 位置 1 以便在首次发送时发送一个空闲帧。

7. 在 USART_DR 寄存器中写入要发送的数据（该操作将清零 TXE 位）。为每个要在单缓冲区模式下发送的数据重复这一步骤。
8. 向 USART_DR 寄存器写入最后一个数据后，等待至 TC=1。这表明最后一个帧的传送已完成。禁止 USART 或进入暂停模式时需要此步骤，以避免损坏最后一次发送。

单字节通信

始终通过向数据寄存器写入数据来将 TXE 位清零。

TXE 位由硬件置 1，它表示：

- 数据已从 TDR 移到移位寄存器中且数据发送已开始。
- TDR 寄存器为空。
- USART_DR 寄存器中可写入下一个数据，而不会覆盖前一个数据。

TXEIE 位置 1 时该标志位会生成中断。

发送时，要传入 USART_DR 寄存器的写指令中存有 TDR 寄存器中的数据，该数据将在当前发送结束时复制到移位寄存器中。

未发送时，要传入 USART_DR 寄存器的写指令直接将数据置于移位寄存器中，数据发送开始时，TXE 位立即置 1。

如果帧已发送（停止位后）且 TXE 位置 1，TC 位将变为高电平。如果 USART_CR1 寄存器中的 TCIE 位置 1，将生成中断。

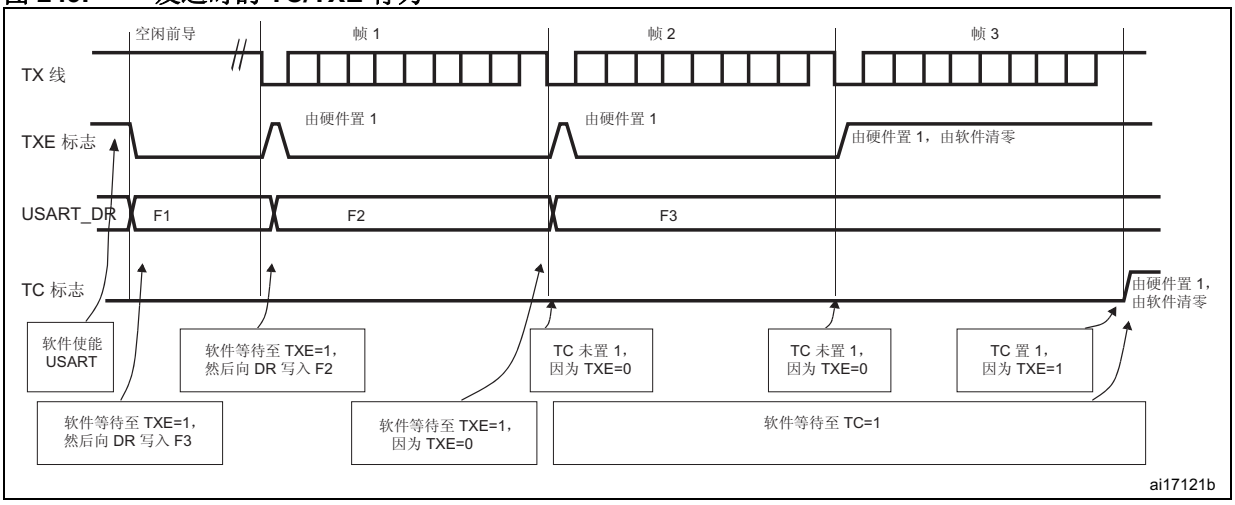
向 USART_DR 寄存器中写入最后一个数据后，必须等待至 TC=1，之后才可禁止 USART 或使微控制器进入低功率模式（请参见*图 249：发送时的 TC/TXE 行为*）。

TC 位通过以下软件序列清零：

1. 从 USART_SR 寄存器读取数据
2. 向 USART_DR 寄存器写入数据

注意： *还可通过向 TC 位写入“0”将其清零。建议仅在多缓冲区通信时使用此清零序列。*

图 249. 发送时的 TC/TXE 行为



中断字符

将 SBK 位置 1 将发送一个中断字符。中断帧的长度取决于 M 位（请参见*图 247*）。

如果 SBK 位置“1”，当前字符发送完成后，将在 TX 线路上发送一个中断字符。中断字符发送完成时（发送中断字符的停止位期间），该位由硬件复位。USART 在上一个中断帧的末尾插入一个逻辑“1”位，以确保识别下个帧的起始位。

注意： *如果软件在中断发送开始前对 SBK 位进行了复位，将不会发送中断字符。对于两个连续的中断，应在上一个中断的停止位发送完成后将 SBK 位置 1。*

空闲字符

将 TE 位置 1 会驱动 USART 在第一个数据帧之前发送一个空闲帧。

26.3.3 接收器

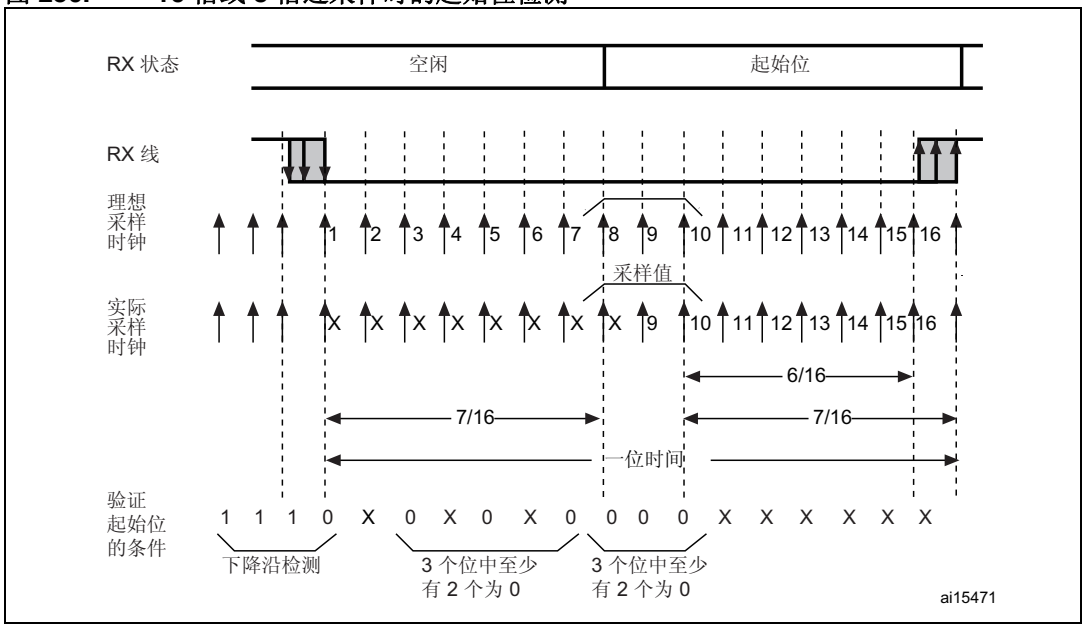
USART 可接收 8 位或 9 位的数据字，具体取决于 USART_CR1 寄存器中的 M 位。

起始位检测

16 倍或 8 倍过采样时，起始位检测序列相同。

在 USART 中，识别出特定序列的采样时会检测起始位。该序列为：1 1 1 0 X 0 X 0 X 0 0 0 0。

图 250. 16 倍或 8 倍过采样时的起始位检测



注意： *如果序列不完整，起始位检测将中止，接收器将返回空闲状态（无标志位置 1）等待下降沿。*

如果 3 个采样位均为 0（针对第 3 位、第 5 位和第 7 位进行首次采样时检测到这 3 位均为 0；针对第 8 位、第 9 位和第 10 位进行第二次采样时检测到这 3 位均为 0），可确认起始位（RXNE 标志位置 1，RXNEIE=1 时生成中断）。

如果两次采样时（对第 3 位、第 5 位和第 7 位进行采样以及对第 8 位、第 9 位和第 10 位进行采样），3 个采样位中至少有 2 个为 0，则可验证起始位（RXNE 标志位置 1，RXNEIE=1 时生成中断）但 NE 噪声标志位置 1。如果不满足此条件，则启动检测中止，接收器返回空闲状态（无标志位置 1）。

如果其中一次采样时（对第 3 位、第 5 位和第 7 位进行采样或对第 8 位、第 9 位和第 10 位进行采样），3 个采样位中有 2 个为 0，则可验证起始位但 NE 噪声标志位置 1。

字符接收

USART 接收期间，首先通过 RX 引脚移入数据的最低有效位。该模式下，USART_DR 寄存器的缓冲区 (RDR) 位于内部总线和接收移位寄存器之间。

步骤：

1. 通过向 USART_CR1 寄存器中的 UE 位写入 1 使能 USART。
2. 对 USART_CR1 中的 M 位进行编程以定义字长。
3. 对 USART_CR2 中的停止位数量进行编程。
4. 如果将进行多缓冲区通信，请选择 USART_CR3 中的 DMA 使能 (DMAR)。按照多缓冲区通信中的解释说明配置 DMA 寄存器。步骤 3
5. 使用波特率寄存器 USART_BRR 选择所需波特率
6. 将 RE 位 USART_CR1 置 1。这一操作将使能接收器开始搜索起始位。

接收到字符时

- RXNE 位置 1。这表明移位寄存器的内容已传送到 RDR。也就是说，已接收到并可读取数据（以及其相应的错误标志）。
- 如果 RXNEIE 位置 1，则会生成中断。
- 如果接收期间已检测到帧错误、噪声错误或上溢错误，错误标志位可置 1。
- 在多缓冲区模式下，每接收到一个字节后 RXNE 均置 1，然后通过 DMA 对数据寄存器执行读操作清零。
- 在单缓冲区模式下，通过软件对 USART_DR 寄存器执行读操作将 RXNE 位清零。RXNE 标志也可以通过向该位写入零来清零。RXNE 位必须在结束接收下一个字符前清零，以避免发生上溢错误。

注意： *接收数据时，不应将 RE 位复位。如果接收期间禁止了 RE 位，则会中止接收当前字节。*

中断字符

接收到中断字符时，USART 将会按照帧错误对其进行处理。

空闲字符

检测到空闲帧时，处理步骤与接收到数据的情况相同；如果 IDLEIE 位为 1，则会产生中断。

上溢错误

如果在 RXNE 未复位时接收到字符，则会发生上溢错误。RXNE 位清零前，数据无法从移位寄存器传送到 RDR 寄存器。

每接收到一个字节后，RXNE 标志位都将置 1。当 RXNE 标志位是 1 时，如果在接收到下一个数据或尚未处理上一个 DMA 请求时，则会发生上溢错误。发生上溢错误时：

- ORE 位置 1。
- RDR 中的内容不会丢失。对 USART_DR 执行读操作时可使用先前的数据。
- 移位寄存器将被覆盖。之后，上溢期间接收到的任何数据都将丢失。
- 如果 RXNEIE 位置 1 或 EIE 与 DMAR 位均为 1，则会生成中断。
- 通过先后对 USART_SR 寄存器和 USART_DR 寄存器执行读操作将 ORE 位清除。

注意： *ORE 位置 1 时表示至少 1 个数据丢失。存在两种可能：*

- *如果 RXNE=1，则最后一个有效数据存储于接收寄存器 RDR 中并且可进行读取；*
- *如果 RXNE=0，则表示最后一个有效数据已被读取，因此 RDR 中没有要读取的数据。接收到新（丢失）数据的同时已读取 RDR 中的最后一个有效数据时，会发生该情况。读取序列期间（在 USART_SR 寄存器读访问与 USART_DR 读访问之间）接收到新数据时也会发生该情况。*

选择合适的过采样方法

接收器采用不同的用户可配置过采样技术（除了同步模式下），可以从噪声中提取有效数据。

可通过编程 USART_CR1 寄存器中的 OVER8 位来选择采样方法，且采样时钟可以是波特率时钟的 16 倍或 8 倍（请参见*图 251* 和*图 252*）。

根据应用：

- 选择 8 倍过采样 (OVER8=1) 以获得更高的速度（高达 $f_{PCLK}/8$）。这种情况下接收器对时钟偏差的最大容差将会降低（请参见*第 695 页的第 26.3.5 节：USART 接收器对时钟偏差的容差*）
- 选择 16 倍过采样 (OVER8=0) 以增加接收器对时钟偏差的容差。这种情况下，最大速度限制为最高 $f_{PCLK}/16$

可通过编程 USART_CR3 寄存器中的 ONEBIT 位选择用于评估逻辑电平的方法。有两种选择：

- 在已接收位的中心进行三次采样，从而进行多数表决。这种情况下，如果用于多数表决的 3 次采样结果不相等，NF 位置 1。
- 在已接收位的中心进行单次采样

  根据应用：
  - 在噪声环境下工作时，请选择三次采样的多数表决法 (ONEBIT=0)；在检测到噪声时请拒绝数据（请参见*图 107*），因为这表示采样过程中产生了干扰。
  - 线路无噪声时请选择单次采样法 (ONEBIT=1) 以增加接收器对时钟偏差的容差（请参见*第 695 页的第 26.3.5 节：USART 接收器对时钟偏差的容差*）。这种情况下 NF 位始终不会置 1。

帧中检测到噪声时：

- 在 RXNE 位的上升沿时 NF 位置 1。
- 无效数据从移位寄存器传送到 USART_DR 寄存器。
- 单字节通信时无中断产生。然而，在 RXNE 位产生中断时，该位出现上升沿。多缓冲区通信时，USART_CR3 寄存器中的 EIE 位置 1 时将发出中断。

通过先后对 USART_SR 寄存器和 USART_DR 寄存器执行读操作将 NF 位清零。

注意： *智能卡、IrDA 和 LIN 模式下不可采用 8 倍过采样。在这些模式下，OVER8 位由硬件强制清零。*

图 251. 16 倍过采样时的数据采样

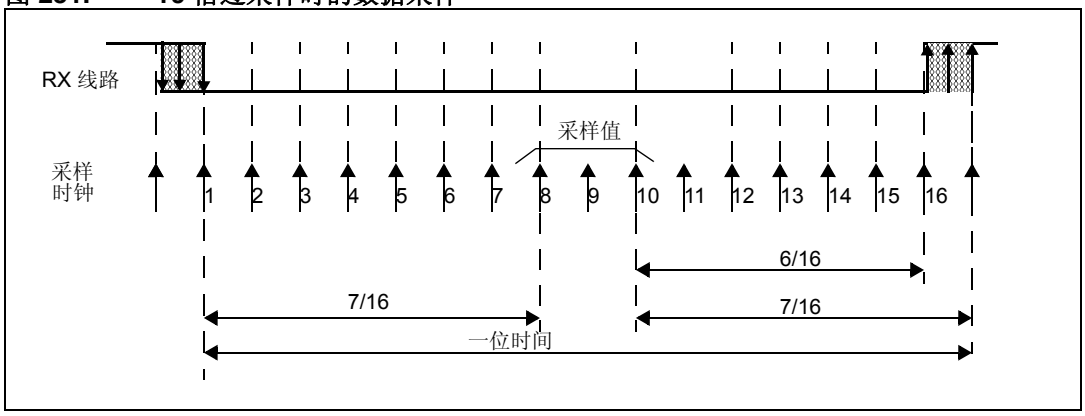


图 252. 8 倍过采样时的数据采样

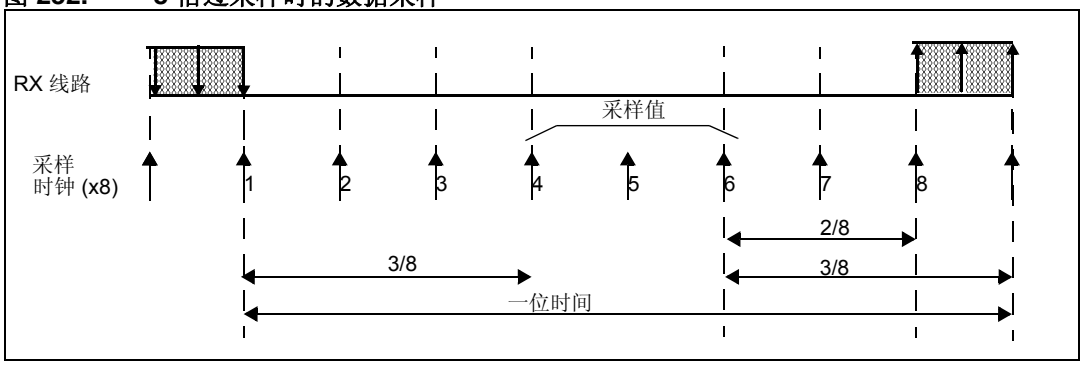


表 107. 通过采样数据进行噪声检测

采样值	NE 状态	接收的位值
000	0	0
001	1	0
010	1	0
011	1	1
100	1	0
101	1	1
110	1	1
111	0	1

帧错误

以下情况下将检测到帧错误：

接收数据时未在预期时间内识别出停止位，从而出现同步失效或过度的噪声。

检测到帧错误时：

- FE 位由硬件置 1
- 无效数据从移位寄存器传送到 USART_DR 寄存器。
- 单字节通信时无中断产生。然而，在 RXNE 位产生中断时，该位出现上升沿。多缓冲区通信时，USART_CR3 寄存器中的 EIE 位置 1 时将发出中断。

通过先后对 USART_SR 寄存器和 USART_DR 寄存器执行读操作将 FE 位清零。

接收期间可配置的停止位

可通过控制寄存器 2 中的控制位配置要接收的停止位的数量 - 可以是 1 或 2 个（正常模式下），也可以是 0.5 或 1.5 个（智能卡模式下）。

1. ***0.5 个停止位（在智能卡模式下接收时）***：不会对 0.5 个停止位进行采样。结果，选择 0.5 个停止位时，无法检测到帧错误和中断帧。
2. ***1 个停止位***：将在第 8、第 9 和第 10 次采样时对 1 个停止位进行采样。
3. ***1.5 个停止位（在智能卡模式下）***：在智能卡模式下发送时，设备必须检查数据是否正确发送。因此必须使能接收器块（USART_CR1 寄存器中的 RE =1）并检查停止位，以测试智能卡是否已检测到奇偶校验错误。发生奇偶校验错误时，智能卡会在采样时将数据信号强制为低电平，即 NACK 信号-，该信号被标记为帧错误。之后，FE 标志在 1.5 个停止位的末尾由 RXNE 置 1。在第 16、第 17 和第 18 次采样时对 1.5 个停止位进行采样（停止位采样开始后维持 1 个波特时钟周期）。1.5 个停止位可分为 2 个部分：0.5 个波特时钟周期（未发生任何动作），然后是 1 个正常的停止位周期（一半时间处进行采样）。有关详细信息，请参见*第 703 页的第 26.3.11 节：智能卡*。
4. ***2 个停止位***：采样 2 个停止位时在第 8、第 9 和第 10 次采样时对第一个停止位进行采样。如果在第一个停止位期间检测到帧错误，则帧错误标志位将会置 1。发生帧错误时不检测第 2 个停止位。RXNE 标志将在第一个停止位末尾时置 1。

26.3.4 小数波特率生成

对 USARTDIV 的尾数值和小数值进行编程时，接收器和发送器（Rx 和 Tx）的波特率均设置为相同值。

公式 1：适用于标准 USART（包括 SPI 模式）的波特率

$$\text{Tx/Rx 波特率} = \frac{f_{CK}}{8 \times (2 - OVER8) \times USARTDIV}$$

公式 2：智能卡、LIN 和 IrDA 模式下的波特率

$$\text{Tx/Rx 波特率} = \frac{f_{CK}}{16 \times USARTDIV}$$

USARTDIV 是一个存放在 USART_BRR 寄存器中的无符号定点数。

- 当 OVER8=0 时，小数部分编码为 4 位并通过 USART_BRR 寄存器中的 DIV_fraction[3:0] 位编程。
- 当 OVER8=1 时，小数部分编码为 3 位并通过 USART_BRR 寄存器中的 DIV_fraction[2:0] 位编程，此时 DIV_fraction[3] 位必须保持清零状态。

注意： *对 USART_BRR 执行写操作后，波特率计数器更新为波特率寄存器中的新值。因此，波特率寄存器的值不应在通信时发生更改。*

OVER8=0 时如何从 USART_BRR 寄存器值中获取 USARTDIV

示例 1：

如果 DIV_Mantissa = 0d27 且 DIV_Fraction = 0d12 (USART_BRR = 0x1BC)，则

尾数 (USARTDIV) = 0d27

小数 (USARTDIV) = 12/16 = 0d0.75

因此 USARTDIV = 0d27.75

示例 2：

要设定 USARTDIV = 0d25.62

这将导致：

DIV_Fraction = 16*0d0.62 = 0d9.92

最接近的实数为 0d10 = 0xA

DIV_Mantissa = 尾数 (0d25.620) = 0d25 = 0x19

则 USART_BRR = 0x19A，因此 USARTDIV = 0d25.625

示例 3：

要设定 USARTDIV = 0d50.99

这将导致：

DIV_Fraction = 16*0d0.99 = 0d15.84

最接近的实数为 0d16 = 0x10 => DIV_frac[3:0] 溢出 => 尾数必须添加进位

DIV_Mantissa = 尾数（0d50.990 + 进位）= 0d51 = 0x33

则 USART_BRR = 0x330，因此 USARTDIV = 0d51.000

OVER8=1 时如何从 USART_BRR 寄存器中获取 USARTDIV

示例 1：

如果 DIV_Mantissa = 0x27 且 DIV_Fraction[2:0]= 0d6 (USART_BRR = 0x1B6)，则

尾数 (USARTDIV) = 0d27

小数 (USARTDIV) = 6/8 = 0d0.75

因此 USARTDIV = 0d27.75

示例 2：

要设定 USARTDIV = 0d25.62

这将导致：

DIV_Fraction = 8*0d0.62 = 0d4.96

最接近的实数为 0d5 = 0x5

DIV_Mantissa = 尾数 (0d25.620) = 0d25 = 0x19

则 USART_BRR = 0x195 => USARTDIV = 0d25.625

示例 3：

要设定 USARTDIV = 0d50.99

这将导致：

DIV_Fraction = 8*0d0.99 = 0d7.92

最接近的实数为 0d8 = 0x8 => DIV_frac[2:0] 溢出 => 尾数必须添加进位

DIV_Mantissa = 尾数（0d50.990 + 进位）= 0d51 = 0x33

则 USART_BRR = 0x0330 => USARTDIV = 0d51.000

表 108.　采用 16 倍过采样时，在 f_{PCLK} = 8 MHz 或 f_{PCLK} = 12 MHz 下编程波特率时的误差计算(1)

16 倍过采样时 (OVER8=0)							
波特率		f_{PCLK} = 8 MHz			f_{PCLK} = 12 MHz		
序号	所需值	实际值	波特率寄存器中编程的值	误差 % = (计算值 - 所需值) / 所需波特率	实际值	波特率寄存器中编程的值	误差 %
1	1.2 KBps	1.2 KBps	416.6875	0	1.2 KBps	625	0
2	2.4 KBps	2.4 KBps	208.3125	0.01	2.4 KBps	312.5	0
3	9.6 KBps	9.604 KBps	52.0625	0.04	9.6 KBps	78.125	0
4	19.2 KBps	19.185 KBps	26.0625	0.08	19.2 KBps	39.0625	0
5	38.4 KBps	38.462 KBps	13	0.16	38.339 KBps	19.5625	0.16
6	57.6 KBps	57.554 KBps	8.6875	0.08	57.692 KBps	13	0.16
7	115.2 KBps	115.942 KBps	4.3125	0.64	115.385 KBps	6.5	0.16
8	230.4 KBps	228.571 KBps	2.1875	0.79	230.769 KBps	3.25	0.16
9	460.8 KBps	470.588 KBps	1.0625	2.12	461.538 KBps	1.625	0.16
10	921.6 KBps	NA	NA	NA	NA	NA	NA
11	2 MBps	NA	NA	NA	NA	NA	NA
12	3 MBps	NA	NA	NA	NA	NA	NA

1. CPU 时钟越低，特定波特率的精度就越低。可达到的波特率的上限可使用这些数据进行确定。

表 109. 采用 8 倍过采样时，在 f_{PCLK} = 8 MHz 或 f_{PCLK} = 12 MHz 下编程波特率时的误差计算(1)

8 倍过采样 (OVER8 = 1)							
波特率		f_{PCLK} = 8 MHz			f_{PCLK} = 12 MHz		
序号	所需值	实际值	波特率寄存器中编程的值	误差 % = (计算值 - 所需值) / 所需波特率	实际值	波特率寄存器中编程的值	误差 %
1	1.2 KBps	1.2 KBps	833.375	0	1.2 KBps	1250	0
2	2.4 KBps	2.4 KBps	416.625	0.01	2.4 KBps	625	0
3	9.6 KBps	9.604 KBps	104.125	0.04	9.6 KBps	156.25	0
4	19.2 KBps	19.185 KBps	52.125	0.08	19.2 KBps	78.125	0
5	38.4 KBps	38.462 KBps	26	0.16	38.339 KBps	39.125	0.16
6	57.6 KBps	57.554 KBps	17.375	0.08	57.692 KBps	26	0.16
7	115.2 KBps	115.942 KBps	8.625	0.64	115.385 KBps	13	0.16
8	230.4 KBps	228.571 KBps	4.375	0.79	230.769 KBps	6.5	0.16
9	460.8 KBps	470.588 KBps	2.125	2.12	461.538 KBps	3.25	0.16
10	921.6 KBps	888.889 KBps	1.125	3.55	923.077 KBps	1.625	0.16
11	2 MBps	NA	NA	NA	NA	NA	NA
12	3 MBps	NA	NA	NA	NA	NA	NA

1. CPU 时钟越低，特定波特率的精度就越低。可达到的波特率的上限可使用这些数据进行确定。

表 110. 采用 16 倍过采样时，在 f_{PCLK} = 16 MHz 或 f_{PCLK} = 24 MHz 下编程波特率时的误差计算(1)

16 倍过采样 (OVER8 = 0)							
波特率		f_{PCLK} = 16 MHz			f_{PCLK} = 24 MHz		
序号	所需值	实际值	波特率寄存器中编程的值	误差 % = (计算值 - 所需值) / 所需波特率	实际值	波特率寄存器中编程的值	误差 %
1	1.2 KBps	1.2 KBps	833.3125	0	1.2	1250	0
2	2.4 KBps	2.4 KBps	416.6875	0	2.4	625	0
3	9.6 KBps	9.598 KBps	104.1875	0.02	9.6	156.25	0
4	19.2 KBps	19.208 KBps	52.0625	0.04	19.2	78.125	0
5	38.4 KBps	38.369 KBps	26.0625	0.08	38.4	39.0625	0
6	57.6 KBps	57.554 KBps	17.375	0.08	57.554	26.0625	0.08
7	115.2 KBps	115.108 KBps	8.6875	0.08	115.385	13	0.16
8	230.4 KBps	231.884 KBps	4.3125	0.64	230.769	6.5	0.16
9	460.8 KBps	457.143 KBps	2.1875	0.79	461.538	3.25	0.16
10	921.6 KBps	941.176 KBps	1.0625	2.12	923.077	1.625	0.16

表 110. 采用 16 倍过采样时，在 f_{PCLK} = 16 MHz 或 f_{PCLK} = 24 MHz 下编程波特率时的误差计算(1)

16 倍过采样 (OVER8 = 0)							
波特率		f_{PCLK} = 16 MHz			f_{PCLK} = 24 MHz		
序号	所需值	实际值	波特率寄存器中编程的值	误差 % = (计算值 - 所需值) / 所需波特率	实际值	波特率寄存器中编程的值	误差 %
11	2 MBps	NA	NA	NA	NA	NA	NA
12	3 MBps	NA	NA	NA	NA	NA	NA

1. CPU 时钟越低，特定波特率的精度就越低。可达到的波特率的上限可使用这些数据进行确定。

表 111. 采用 8 倍过采样时，在 f_{PCLK} = 16 MHz 或 f_{PCLK} = 24 MHz 下编程波特率时的误差计算(1)

8 倍过采样 (OVER8=1)							
波特率		f_{PCLK} = 16 MHz			f_{PCLK} = 24 MHz		
序号	所需值	实际值	波特率寄存器中编程的值	误差 % = (计算值 - 所需值) / 所需波特率	实际值	波特率寄存器中编程的值	误差 %
1	1.2 KBps	1.2 KBps	1666.625	0	1.2 KBps	2500	0
2	2.4 KBps	2.4 KBps	833.375	0	2.4 KBps	1250	0
3	9.6 KBps	9.598 KBps	208.375	0.02	9.6 KBps	312.5	0
4	19.2 KBps	19.208 KBps	104.125	0.04	19.2 KBps	156.25	0
5	38.4 KBps	38.369 KBps	52.125	0.08	38.4 KBps	78.125	0
6	57.6 KBps	57.554 KBps	34.75	0.08	57.554 KBps	52.125	0.08
7	115.2 KBps	115.108 KBps	17.375	0.08	115.385 KBps	26	0.16
8	230.4 KBps	231.884 KBps	8.625	0.64	230.769 KBps	13	0.16
9	460.8 KBps	457.143 KBps	4.375	0.79	461.538 KBps	6.5	0.16
10	921.6 KBps	941.176 KBps	2.125	2.12	923.077 KBps	3.25	0.16
11	2 MBps	2000 KBps	1	0	2000 KBps	1.5	0
12	3 MBps	NA	NA	NA	3000 KBps	1	0

1. CPU 时钟越低，特定波特率的精度就越低。可达到的波特率的上限可使用这些数据进行确定。

表 112. 采用 16 倍过采样时，在 f_{PCLK} = 8 MHz 或 f_{PCLK} = 16 MHz 下编程波特率时的误差计算(1)

16 倍过采样时 (OVER8=0)							
波特率		f_{PCLK} = 8 MHz			f_{PCLK} = 16 MHz		
序号	所需值	实际值	波特率寄存器中编程的值	误差 % =（计算值 - 所需值）/所需波特率	实际值	波特率寄存器中编程的值	误差 %
1.	**2.4 KBps**	2.400 KBps	208.3125	0.00%	2.400 KBps	416.6875	0.00%
2.	**9.6 KBps**	9.604 KBps	52.0625	0.04%	9.598 KBps	104.1875	0.02%
3.	**19.2 KBps**	19.185 KBps	26.0625	0.08%	19.208 KBps	52.0625	0.04%
4.	**57.6 KBps**	57.554 KBps	8.6875	0.08%	57.554 KBps	17.3750	0.08%
5.	**115.2 KBps**	115.942 KBps	4.3125	0.64%	115.108 KBps	8.6875	0.08%
6.	**230.4 KBps**	228.571 KBps	2.1875	0.79%	231.884 KBps	4.3125	0.64%
7.	**460.8 KBps**	470.588 KBps	1.0625	2.12%	457.143 KBps	2.1875	0.79%
8.	**896 KBps**	NA	NA	NA	888.889 KBps	1.1250	0.79%
9.	**921.6 KBps**	NA	NA	NA	941.176 KBps	1.0625	2.12%
10.	**1.792 MBps**	NA	NA	NA	NA	NA	NA
11.	**1.8432 MBps**	NA	NA	NA	NA	NA	NA
12.	**3.584 MBps**	NA	NA	NA	NA	NA	NA
13.	**3.6864 MBps**	NA	NA	NA	NA	NA	NA
14.	**7.168 MBps**	NA	NA	NA	NA	NA	NA
15.	**7.3728 MBps**	NA	NA	NA	NA	NA	NA

1. CPU 时钟越低，特定波特率的精度就越低。可达到的波特率的上限可使用这些数据进行确定。

表 113. 采用 8 倍过采样时，在 f_{PCLK} = 8 MHz 或 f_{PCLK} = 16 MHz 下编程波特率时的误差计算(1)

8 倍过采样 (OVER8=1)							
波特率		f_{PCLK} = 8 MHz			f_{PCLK} = 16 MHz		
序号	所需值	实际值	波特率寄存器中编程的值	误差 % =（计算值 - 所需值）/所需波特率	实际值	波特率寄存器中编程的值	误差 %
1.	**2.4 KBps**	2.400 KBps	416.625	0.01%	2.400 KBps	833.375	0.00%
2.	**9.6 KBps**	9.604 KBps	104.125	0.04%	9.598 KBps	208.375	0.02%
3.	**19.2 KBps**	19.185 KBps	52.125	0.08%	19.208 KBps	104.125	0.04%
4.	**57.6 KBps**	57.557 KBps	17.375	0.08%	57.554 KBps	34.750	0.08%
5.	**115.2 KBps**	115.942 KBps	8.625	0.64%	115.108 KBps	17.375	0.08%
6.	**230.4 KBps**	228.571 KBps	4.375	0.79%	231.884 KBps	8.625	0.64%
7.	**460.8 KBps**	470.588 KBps	2.125	2.12%	457.143 KBps	4.375	0.79%
8.	**896 KBps**	888.889 KBps	1.125	0.79%	888.889 KBps	2.250	0.79%
9.	**921.6 KBps**	888.889 KBps	1.125	3.55%	941.176 KBps	2.125	2.12%
10.	**1.792 MBps**	NA	NA	NA	1.7777 MBps	1.125	0.79%

表 113. 采用 8 倍过采样时，在 f_{PCLK} = 8 MHz 或 f_{PCLK} = 16 MHz 下编程波特率时的误差计算(1)

8 倍过采样 (OVER8=1)							
波特率		f_{PCLK} = 8 MHz			f_{PCLK} = 16 MHz		
序号	所需值	实际值	波特率寄存器中编程的值	误差 % = （计算值 - 所需值）/ 所需波特率	实际值	波特率寄存器中编程的值	误差 %
11.	**1.8432 MBps**	NA	NA	NA	1.7777 MBps	1.125	3.55%
12.	**3.584 MBps**	NA	NA	NA	NA	NA	NA
13.	**3.6864 MBps**	NA	NA	NA	NA	NA	NA
14.	**7.168 MBps**	NA	NA	NA	NA	NA	NA
15.	**7.3728 MBps**	NA	NA	NA	NA	NA	NA

1. CPU 时钟越低，特定波特率的精度就越低。可达到的波特率的上限可使用这些数据进行确定。

表 114. 采用 16 倍过采样时，在 f_{PCLK} = 30 MHz 或 f_{PCLK} = 60 MHz 下编程波特率时的误差计算(1)(2)

16 倍过采样时 (OVER8=0)							
波特率		f_{PCLK} = 30 MHz			f_{PCLK} = 60 MHz		
序号	所需值	实际值	波特率寄存器中编程的值	误差 % =（计算值 - 所需值）/所需波特率	实际值	波特率寄存器中编程的值	误差 %
1.	**2.4 KBps**	2.400 KBps	781.2500	0.00%	2.400 KBps	1562.5000	0.00%
2.	**9.6 KBps**	9.600 KBps	195.3125	0.00%	9.600 KBps	390.6250	0.00%
3.	**19.2 KBps**	19.194 KBps	97.6875	0.03%	19.200 KBps	195.3125	0.00%
4.	**57.6 KBps**	57.582KBps	32.5625	0.03%	57.582 KBps	65.1250	0.03%
5.	**115.2 KBps**	115.385 KBps	16.2500	0.16%	115.163 KBps	32.5625	0.03%
6.	**230.4 KBps**	230.769 KBps	8.1250	0.16%	230.769KBps	16.2500	0.16%
7.	**460.8 KBps**	461.538 KBps	4.0625	0.16%	461.538 KBps	8.1250	0.16%
8.	**896 KBps**	909.091 KBps	2.0625	1.46%	895.522 KBps	4.1875	0.05%
9.	**921.6 KBps**	909.091 KBps	2.0625	1.36%	923.077 KBps	4.0625	0.16%
10.	**1.792 MBps**	1.1764 MBps	1.0625	1.52%	1.8182 MBps	2.0625	1.36%
11.	**1.8432 MBps**	1.8750 MBps	1.0000	1.73%	1.8182 MBps	2.0625	1.52%
12.	**3.584 MBps**	NA	NA	NA	3.2594 MBps	1.0625	1.52%
13.	**3.6864 MBps**	NA	NA	NA	3.7500 MBps	1.0000	1.73%
14.	**7.168 MBps**	NA	NA	NA	NA	NA	NA
15.	**7.3728 MBps**	NA	NA	NA	NA	NA	NA

1. CPU 时钟越低，特定波特率的精度就越低。可达到的波特率的上限可使用这些数据进行确定。
2. 仅 USART1 和 USART6 使用 PCLK2 计时。其它 USART 使用 PCLK1 计时。有关 PCLK1 和 PCLK2 的最大值，请参见器件数据手册。

表 115. 采用 8 倍过采样时，在 f_{PCLK} = 30 MHz 或 f_{PCLK} = 60 MHz 下编程波特率时的误差计算(1) (2)

8 倍过采样 (OVER8=1)							
波特率		f_{PCLK} = 30 MHz			f_{PCLK} = 60 MHz		
序号	所需值	实际值	波特率寄存器中编程的值	误差 % =（计算值 - 所需值）/所需波特率	实际值	波特率寄存器中编程的值	误差 %
1.	**2.4 KBps**	2.400 KBps	1562.5000	0.00%	2.400 KBps	3125.0000	0.00%
2.	**9.6 KBps**	9.600 KBps	390.6250	0.00%	9.600 KBps	781.2500	0.00%
3.	**19.2 KBps**	19.194 KBps	195.3750	0.03%	19.200 KBps	390.6250	0.00%
4.	**57.6 KBps**	57.582 KBps	65.1250	0.16%	57.582 KBps	130.2500	0.03%
5.	**115.2 KBps**	115.385 KBps	32.5000	0.16%	115.163 KBps	65.1250	0.03%
6.	**230.4 KBps**	230.769 KBps	16.2500	0.16%	230.769 KBps	32.5000	0.16%
7.	**460.8 KBps**	461.538 KBps	8.1250	0.16%	461.538 KBps	16.2500	0.16%
8.	**896 KBps**	909.091 KBps	4.1250	1.46%	895.522 KBps	8.3750	0.05%
9.	**921.6 KBps**	909.091 KBps	4.1250	1.36%	923.077 KBps	8.1250	0.16%
10.	**1.792 MBps**	1.7647 MBps	2.1250	1.52%	1.8182 MBps	4.1250	1.46%
11.	**1.8432 MBps**	1.8750 MBps	2.0000	1.73%	1.8182 MBps	4.1250	1.36%
12.	**3.584 MBps**	3.7500 MBps	1.0000	4.63%	3.5294 MBps	2.1250	1.52%
13.	**3.6864 MBps**	3.7500 MBps	1.0000	1.73%	3.7500 MBps	2.0000	1.73%
14.	**7.168 MBps**	NA	NA	NA	7.5000 MBps	1.0000	4.63%
15.	**7.3728 MBps**	NA	NA	NA	7.5000 MBps	1.0000	1.73%

1. CPU 时钟越低，特定波特率的精度就越低。可达到的波特率的上限可使用这些数据进行确定。
2. 仅 USART1 和 USART6 使用 PCLK2 计时。其它 USART 使用 PCLK1 计时。有关 PCLK1 和 PCLK2 的最大值，请参见器件数据手册。

表 116. 采用 16 倍过采样时，在 f_{PCLK} = 42 MHz 或 f_{PCLK} = 84 Hz 下编程波特率时的误差计算(1)(2)

16 倍过采样时 (OVER8=0)							
波特率		f_{PCLK} = 42 MHz			f_{PCLK} = 84 MHz		
序号	所需值	实际值	波特率寄存器中编程的值	误差 % =（计算值 - 所需值）/所需波特率	实际值	波特率寄存器中编程的值	误差 %
1	1.2 KBps	1.2 KBps	2187.5	0	1.2 KBps	4375	0
2	2.4 KBps	2.4 KBps	1093.75	0	2.4 KBps	2187.5	0
3	9.6 KBps	9.6 KBps	273.4375	0	9.6 KBps	546.875	0
4	19.2 KBps	19.195 KBps	136.75	0.02	19.2 KBps	273.4375	0
5	38.4 KBps	38.391 KBps	68.375	0.02	38.391 KBps	136.75	0.02
6	57.6 KBps	57.613 KBps	45.5625	0.02	57.613 KBps	91.125	0.02
7	115.2 KBps	115.068 KBps	22.8125	0.11	115.226 KBps	45.5625	0.02
8	230.4 KBps	230.769 KBps	11.375	0.16	230.137 KBps	22.8125	0.11

表 116. 采用 16 倍过采样时，在 f_{PCLK} = 42 MHz 或 f_{PCLK} = 84 Hz 下编程波特率时的误差计算(1)(2)

16 倍过采样时 (OVER8=0)							
波特率		f_{PCLK} = 42 MHz			f_{PCLK} = 84 MHz		
序号	所需值	实际值	波特率寄存器中编程的值	误差 % =（计算值 - 所需值）/所需波特率	实际值	波特率寄存器中编程的值	误差 %
9	460.8 KBps	461.538 KBps	5.6875	0.16	461.538 KBps	11.375	0.16
10	921.6 KBps	913.043 KBps	2.875	0.93	923.076 KBps	5.6875	0.93
11	1.792 MBps	1.826 MBps	1.4375	1.9	1.787 MBps	2.9375	0.27
12	1.8432 MBps	1.826 MBps	1.4375	0.93	1.826 MBps	2.875	0.93
13	3.584 MBps	N.A	N.A	N.A	3.652 MBps	1.4375	1.9
14	3.6864 MBps	N.A	N.A	N.A	3.652 MBps	1.4375	0.93
15	7.168 MBps	N.A	N.A	N.A	N.A	N.A	N.A
16	7.3728 MBps	N.A	N.A	N.A	N.A	N.A	N.A
18	9 MBps	N.A	N.A	N.A	N.A	N.A	N.A
20	10.5 MBps	N.A	N.A	N.A	N.A	N.A	N.A

1. CPU 时钟越低，特定波特率的精度就越低。可达到的波特率的上限可使用这些数据进行确定。
2. 仅 USART1 和 USART6 使用 PCLK2 计时。其它 USART 使用 PCLK1 计时。有关 PCLK1 和 PCLK2 的最大值，请参见器件数据手册。

表 117. 采用 8 倍过采样时，在 f_{PCLK} = 42 MHz 或 f_{PCLK} = 84 MHz 下编程波特率时的误差计算(1)(2)

8 倍过采样 (OVER8=1)							
波特率		f_{PCLK} = 42 MHz			f_{PCLK} = 84 MHz		
序号	所需值	实际值	波特率寄存器中编程的值	误差 % =（计算值 - 所需值）/ 所需波特率	实际值	波特率寄存器中编程的值	误差 %
1.	1.2 KBps	1.2 KBps	4375	0	1.2 KBps	8750	0
2.	2.4 KBps	2.4 KBps	2187.5	0	2.4 KBps	4375	0
3.	9.6 KBps	9.6 KBps	546.875	0	9.6 KBps	1093.75	0
4.	19.2 KBps	19.195 KBps	273.5	0.02	19.2 KBps	546.875	0
5.	38.4 KBps	38.391 KBps	136.75	0.02	38.391 KBps	273.5	0.02
6.	57.6 KBps	57.613 KBps	91.125	0.02	57.613 KBps	182.25	0.02
7.	115.2 KBps	115.068 KBps	45.625	0.11	115.226 KBps	91.125	0.02
8.	230.4 KBps	230.769 KBps	22.75	0.11	230.137 KBps	45.625	0.11
9.	460.8 KBps	461.538 KBps	11.375	0.16	461.538 KBps	22.75	0.16
10.	921.6 KBps	913.043 KBps	5.75	0.93	923.076 KBps	11.375	0.93
11.	1.792 MBps	1.826 MBps	2.875	1.9	1.787Mbps	5.875	0.27
12.	1.8432 MBps	1.826 MBps	2.875	0.93	1.826 MBps	5.75	0.93
13.	3.584 MBps	3.5 MBps	1.5	2.34	3.652 MBps	2.875	1.9
14.	3.6864 MBps	3.82 MBps	1.375	3.57	3.652 MBps	2.875	0.93

表 117. 采用 8 倍过采样时，在 f_{PCLK} = 42 MHz 或 f_{PCLK} = 84 MHz 下编程波特率时的误差计算(1)(2)

8 倍过采样 (OVER8=1)							
波特率		f_{PCLK} = 42 MHz			f_{PCLK} = 84 MHz		
序号	所需值	实际值	波特率寄存器中编程的值	误差 % =（计算值 - 所需值）/ 所需波特率	实际值	波特率寄存器中编程的值	误差 %
15.	7.168 MBps	N.A	N.A	N.A	7 MBps	1.5	2.34
16.	7.3728 MBps	N.A	N.A	N.A	7.636 MBps	1.375	3.57
18.	9 MBps	N.A	N.A	N.A	9.333 MBps	1.125	3.7
20.	10.5 MBps	N.A	N.A	N.A	10.5 MBps	1	0

1. CPU 时钟越低，特定波特率的精度就越低。可达到的波特率的上限可使用这些数据进行确定。
2. 仅 USART1 和 USART6 使用 PCLK2 计时。其它 USART 使用 PCLK1 计时。有关 PCLK1 和 PCLK2 的最大值，请参见器件数据手册。

26.3.5 USART 接收器对时钟偏差的容差

仅当总时钟系统偏差小于 USART 接收器的容差时，USART 异步接收器器才能正常工作。影响总偏差的因素包括：

- DTRA：发送器误差引起的偏差（其中还包括发送器本地振荡器的偏差）
- DQUANT：接收器的波特率量化引起的误差
- DREC：接收器本地振荡器的偏差
- DTCL：传输线路引起的偏差（通常是由于收发器所引起，它可能会在低电平到高电平转换时序与高电平到低电平转换时序之间引入不对称）

DTRA + DQUANT + DREC + DTCL < USART 接收器的容差

对于正常接收数据，USART 接收器的容差等于所容许的最大偏差，具体取决于以下选择：

- 由 USART_CR1 寄存器中的 M 位定义的 10 位或 11 位字符长度
- 由 USART_CR1 寄存器中的 OVER8 位定义的 8 倍或 16 倍过采样
- 是否使用小数波特率
- 使用 1 位或 3 位对数据进行采样，取决于 USART_CR3 寄存器中 ONEBIT 位的值

表 118. DIV_Fraction 为 0 时的 USART 接收器容差

M 位	OVER8 位 = 0		OVER8 位 = 1	
	ONEBIT=0	ONEBIT=1	ONEBIT=0	ONEBIT=1
0	3.75%	4.375%	2.50%	3.75%
1	3.41%	3.97%	2.27%	3.41%

表 119. DIV_Fraction 不为 0 时的 USART 接收器容差

M 位	OVER8 位 = 0		OVER8 位 = 1	
	ONEBIT=0	ONEBIT=1	ONEBIT=0	ONEBIT=1
0	3.33%	3.88%	2%	3%
1	3.03%	3.53%	1.82%	2.73%

注意：当接收的帧包含 10 个位时间的空闲帧（M=0）或 11 个位时间的空闲帧（M=1）时，表 118 和表 119 中指定的数字可能与特例中的数字略微不同。

26.3.6 多处理器通信

可以与 USART 进行多处理器通信（多个 USART 连接在一个网络中）。例如，其中一个 USART 可以是主 USART，其 TX 输出与其它 USART 的 RX 输入相连接。其它 USART 为从 USART，其各自的 TX 输出在逻辑上通过与运算连在一起，并与主 USART 的 RX 输入相连接。

在多处理器配置中，理想情况下通常只有预期的消息接收方主动接收完整的消息内容，从而减少由所有未被寻址的接收器造成的冗余 USART 服务开销。

可通过静音功能将未被寻址的器件置于静音模式下。在静音模式下：

- 不得将接收状态位置 1。
- 禁止任何接收中断。
- USART_CR1 寄存器中的 RWU 位置 1。RWU 可由硬件自动控制，或在特定条件下由软件写入。

根据 USART_CR1 寄存器中 WAKE 位的设置，USART 可使用以下两种方法进入或退出静音模式：

- 如果 WAKE 位被复位，则进行空闲线路检测，
- 如果 WAKE 位置 1，则进行地址标记检测。

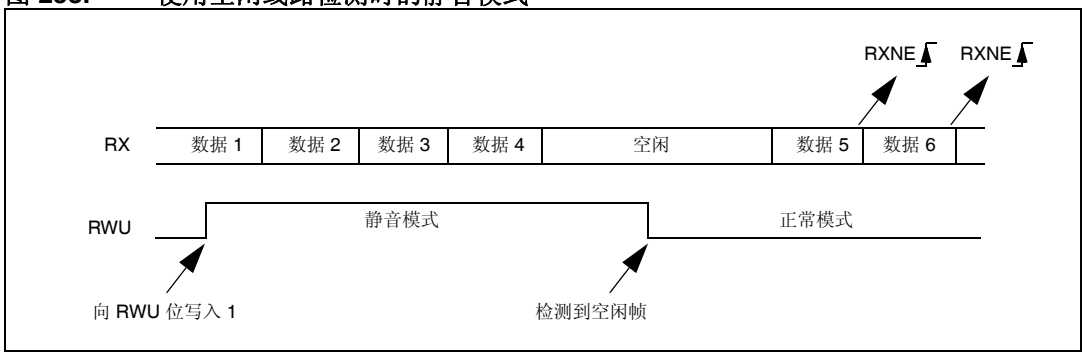
空闲线路检测 (WAKE=0)

当向 RWU 位写入 1 时，USART 进入静音模式。

当检测到空闲帧时，它会被唤醒。此时 RWU 位会由硬件清零，但 USART_SR 寄存器中的 IDLE 位不会置 1。还可通过软件向 RWU 位写入 0。

图 253 中给出了使用空闲线路检测时静音模式行为的示例。

图 253. 使用空闲线路检测时的静音模式



地址标记检测 (WAKE=1)

在此模式下，如果字节的 MSB 为 1，则将这些字节识别为地址，否则将其识别为数据。在地址字节中，目标接收器的地址位于 4 个 LSB 上。接收器会将此 4 位字与其地址进行比较，该接收器的地址在 USART_CR2 寄存器的 ADD 位中进行设置。

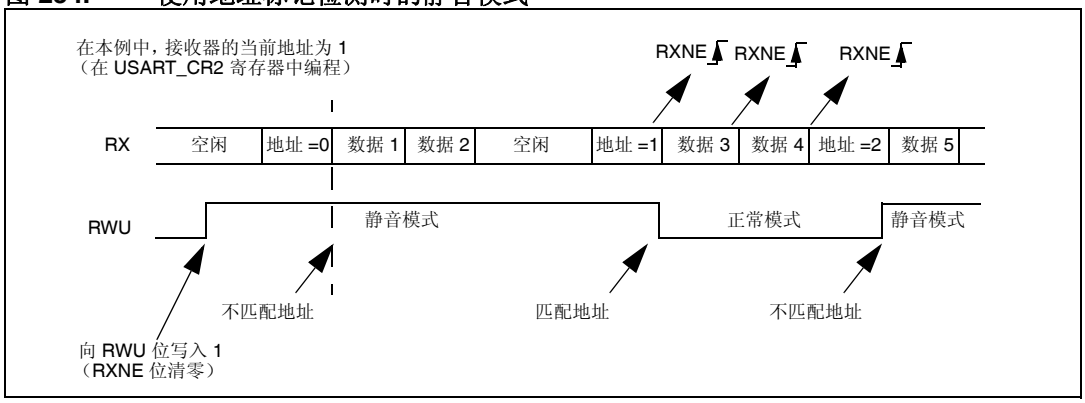
当接收到与其编程地址不匹配的地址字符时，USART 会进入静音模式。此时，RWU 位将由硬件置 1。由于当时 USART 已经进入了静音模式，所以 RXNE 标志不会针对此地址字节置 1，也不会发出中断或 DMA 请求。

当接收到与编程地址匹配的地址字符时，它会退出静音模式。然后 RWU 位被清零，可以开始正常接收后续字节。由于 RWU 位已清零，RXNE 位会针对地址字符 置 1。

当接收器的缓冲区不包含任何数据（USART_SR 寄存器中 RXNE=0）时，可向 RWU 位写入 0 或 1。否则会忽略写尝试。

图 254 中给出了使用地址标记检测时静音模式行为的示例。

图 254. 使用地址标记检测时的静音模式



26.3.7 奇偶校验控制

将 USART_CR1 寄存器中的 PCE 位置 1，可以使能奇偶校验控制（发送时生成奇偶校验位，接收时进行奇偶校验检查）。根据 M 位定义的帧长度，*表 120* 中列出了可能的 USART 帧格式。

表 120. 帧格式

M 位	PCE 位	USART 帧(1)
0	0	起始位 | 8 位数据 | 停止位
0	1	起始位 | 7 位数据 | PB | 停止位
1	0	起始位 | 9 位数据 | 停止位
1	1	起始位 | 8 位数据 PB | 停止位

1. 图注：SB：起始位，STB：停止位，P：奇偶校验位。

偶校验

对奇偶校验位进行计算，使帧和奇偶校验位中“1”的数量为偶数（帧由 7 个或 8 个 LSB 位组成，具体取决于 M 等于 0 还是 1）。

例如：数据=00110101；4 个位置 1 => 如果选择偶校验（USART_CR1 寄存器中的 PS 位 = 0），则校验位是 0。

奇校验

对奇偶校验位进行计算，使帧和奇偶校验位中“1”的数量为奇数（帧由 7 个或 8 个 LSB 位组成，具体取决于 M 等于 0 还是 1）。

例如：数据=00110101；4 个位置 1 => 如果选择奇校验（USART_CR1 寄存器中的 PS 位 = 1），则校验位是 1。

接收时进行奇偶校验检查

如果奇偶校验检查失败，则 USART_SR 寄存器中的 PE 标志置 1；如果 USART_CR1 寄存器中 PEIE 位置 1，则会生成中断。PE 标志由软件序列清零（从状态寄存器中读取，然后对 USART_DR 数据寄存器执行读或写访问）。

注意： *如果被地址标记唤醒：会使用数据的 MSB 位而非奇偶校验位来识别地址。此外，接收器不会对地址数据进行奇偶校验检查（奇偶校验出错时，PE 不置 1）。*

发送时的奇偶校验生成

如果 USART_CR1 寄存器中的 PCE 位置 1，则在数据寄存器中所写入数据的 MSB 位会进行传送，但是会由奇偶校验位进行更改（如果选择偶校验 (PS=0)，则“1”的数量为偶数；如果选择奇校验 (PS=1)，则“1”的数量为奇数）。

注意： *用于管理发送过程的软件程序可以激活软件序列，进而将 PE 标志清零（从状态寄存器中读取，然后对数据寄存器执行读或写访问）。在半双工模式下工作时（具体取决于软件），这可能会导致 PE 标志意外清零。*

26.3.8 LIN（局域互连网络）模式

通过将 USART_CR2 寄存器中的 LINEN 位置 1 来选择 LIN 模式。在 LIN 模式下，必须将以下位清零：

- USART_CR2 寄存器中的 CLKEN 位，
- USART_CR3 寄存器中的 STOP[1:0]、SCEN、HDSEL 和 IREN 位。

LIN 发送

与正常的 USART 发送相比，在 LIN 主器件中发送时必须采用 *第 26.3.2 节* 中介绍的相同步骤，同时还具有以下区别：

- M 位清零以配置 8 位字长度。
- LINEN 位置 1 以进入 LIN 模式。此时，将 SBK 位置 1 会发送 13 个“0”位作为断路字符。然后会发送值为“1”的位以进行下一启动检测。

LIN 接收

断路检测电路在 USART 接口上实现。该检测完全独立于正常的 USART 接收器。在空闲状态或某个帧期间，只要发生断路即可检测出来。

接收器（USART_CR1 寄存器中 RE=1）使能后，电路便开始监测启动信号的 RX 输入。检测起始位的方法与搜索断路字符或数据的方法相同。检测到起始位后，电路会对接下来的位进行采样，方法与数据采样相同（第 8、第 9 和第 10 次采样）。如果 10 个（USART_CR2 寄存器中 LBDL = 0 时）或 11 个（USART_CR2 寄存器中 LBDL=1 时）连续位均检测为“0”，且其后跟随分隔符，则 USART_SR 寄存器中的 LBD 标志将会置 1。如果 LBDIE 位 =1，则会生成中断。在验证断路前，会对分隔符进行检查，因为它表示 RX 线路已恢复到高电平。

如果在第 10 或第 11 次采样前已对“1”采样，则断路检测电路会取消当前检测，并重新搜索起始位。

如果禁止 LIN 模式 (LINEN=0)，接收器会作为正常的 USART 继续工作，不会再进行断路检测。

如果使能 LIN 模式 (LINEN=1)，只要发生帧错误（例如，在“0”处检测到停止位，这种情况可能出现在任何断路帧中），接收器即会停止，直到断路检测电路接收到“1”（断路字不完整时）或接收到分隔符（检测到断路时）为止。

*图 255：第 699 页的LIN 模式下的断路检测（11 位断路长度——LBDL 位置 1）*中显示了断路检测器状态机和断路标志的行为。

*图 256：第 700 页的LIN 模式下的断路检测与帧错误检测*中列出了断路帧的示例。

图 255. LIN 模式下的断路检测（11 位断路长度——LBDL 位置 1）

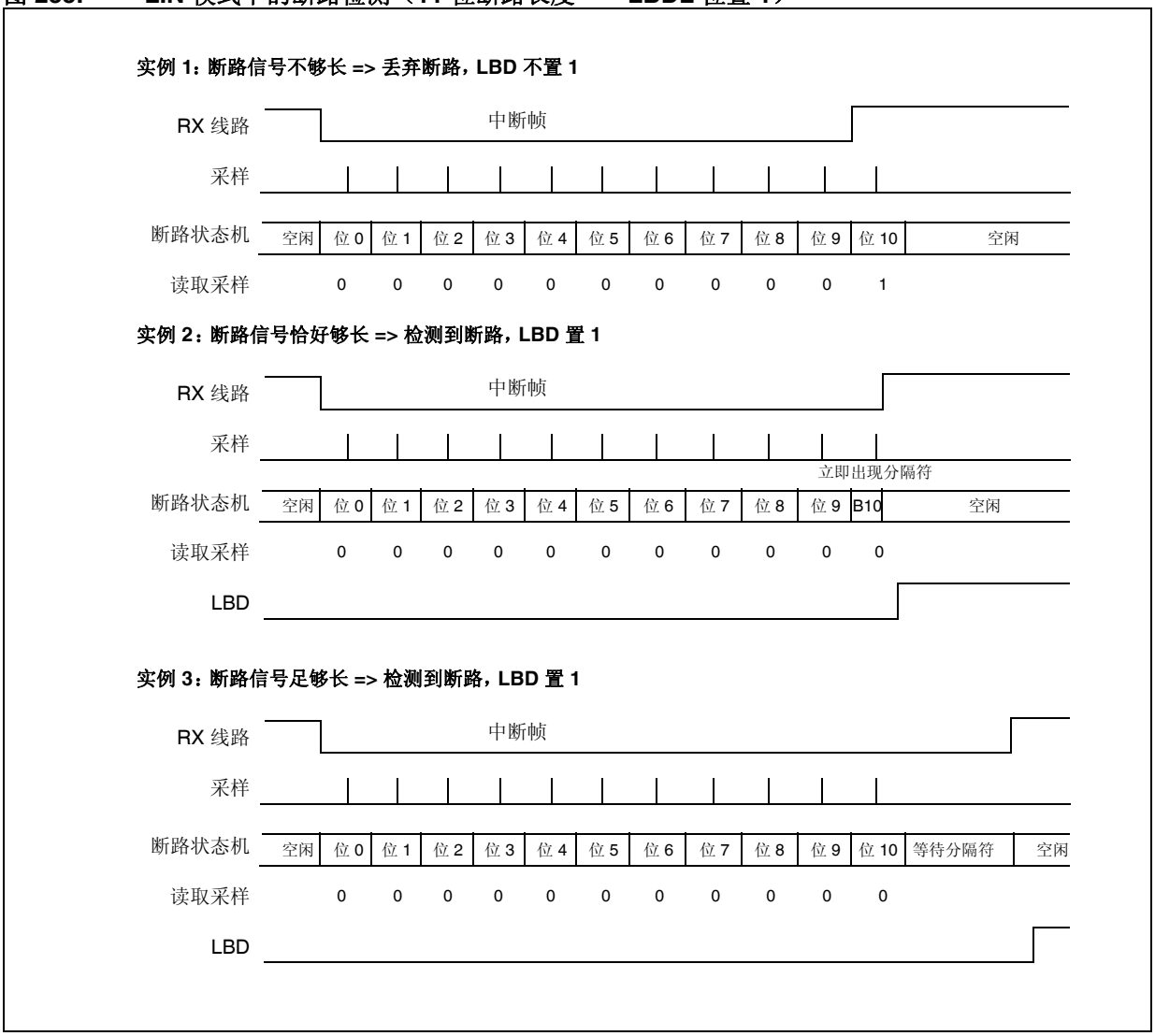
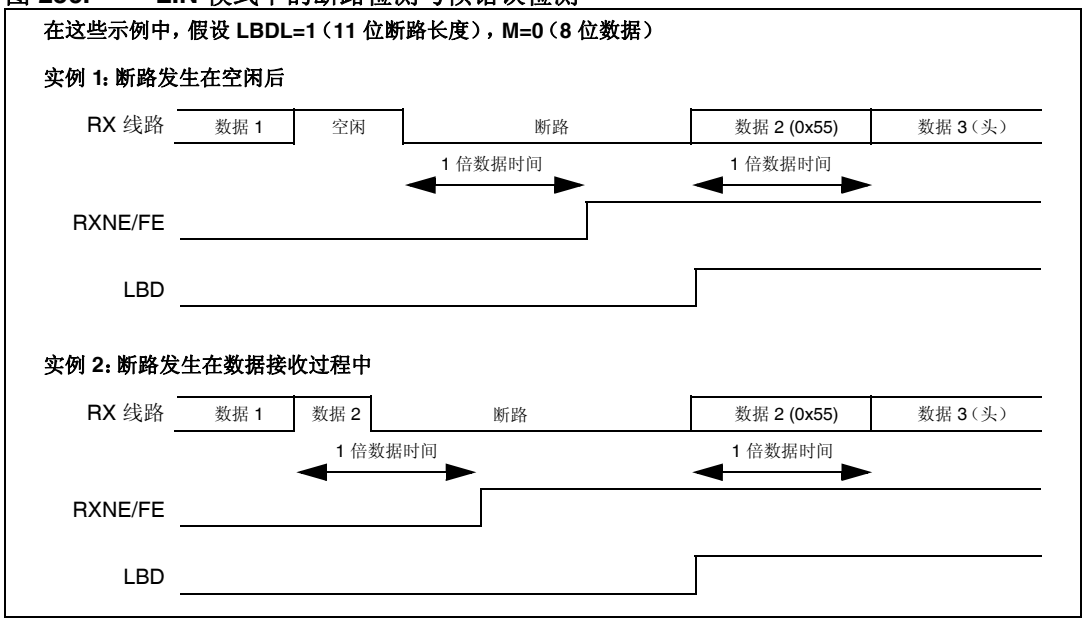


图 256. LIN 模式下的断路检测与帧错误检测



26.3.9 USART 同步模式

通过将 USART_CR2 寄存器中的 CLKEN 位写入 1 来选择同步模式。在同步模式下，必须将以下位清零：

- USART_CR2 寄存器中的 LINEN 位，
- USART_CR3 寄存器中的 SCEN、HDSEL 和 IREN 位。

通过 USART，用户可以在主模式下控制双向同步串行通信。SCLK 引脚是 USART 发送器时钟的输出。在起始位或停止位期间，不会向 SCLK 引脚发送时钟脉冲。在最后一个有效数据位（地址标记）期间，将会（也可能不会）生成时钟脉冲，这取决于 USART_CR2 寄存器中 LBCL 位的状态。通过 USART_CR2 寄存器中的 CPOL 位，用户可以选择时钟极性；通过 USART_CR2 寄存器中的 CPHA 位，用户可以选择外部时钟相位（参见*图 257*、*图 258* 和*图 259*）。

在空闲状态、报头模式和发送断路期间，外部 SCLK 时钟处于未激活状态。

USART 发送器在同步模式下的工作方式与异步模式下完全相同。但是由于 SCLK 与 TX 同步（根据 CPOL 和 CPHA），因此 TX 上的数据是同步的。

在此模式下，USART 接收器的工作方式与异步模式下不同。如果 RE=1，则数据在 SCLK 上采样（上升或下降沿，取决于 CPOL 和 CPHA），而不会进行任何过采样。此时必须确保建立时间和保持时间（取决于波特率：1/16 位时间）符合要求。

注意： *SCLK 引脚可与 TX 引脚结合使用。因此，仅当使能发送器 (TE=1) 且正在发送数据时（对数据寄存器 USART_DR 已被写入），才会提供时钟。这意味着，没有发送数据的情况下无法接收同步数据。*

当发送器和接收器 (TE=RE=0) 都被禁止时，必须选择 LBCL、CPOL 和 CPHA 位，以确保时钟脉冲正常工作。当使能发送器或接收器时，不得对这些位进行更改。

建议按照相同指令将 TE 和 RE 位置 1，以尽量缩短接收器的建立时间和保持时间。

USART 只支持主模式：它不能接收或发送与输入时钟相关的数据（SCLK 始终为输出）。

图 257. USART 同步发送示例

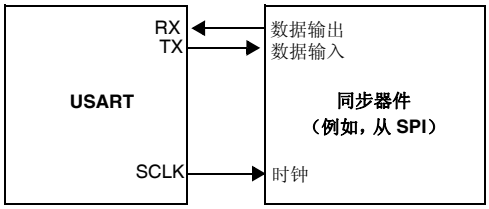


图 258. USART 数据时钟时序图 (M=0)

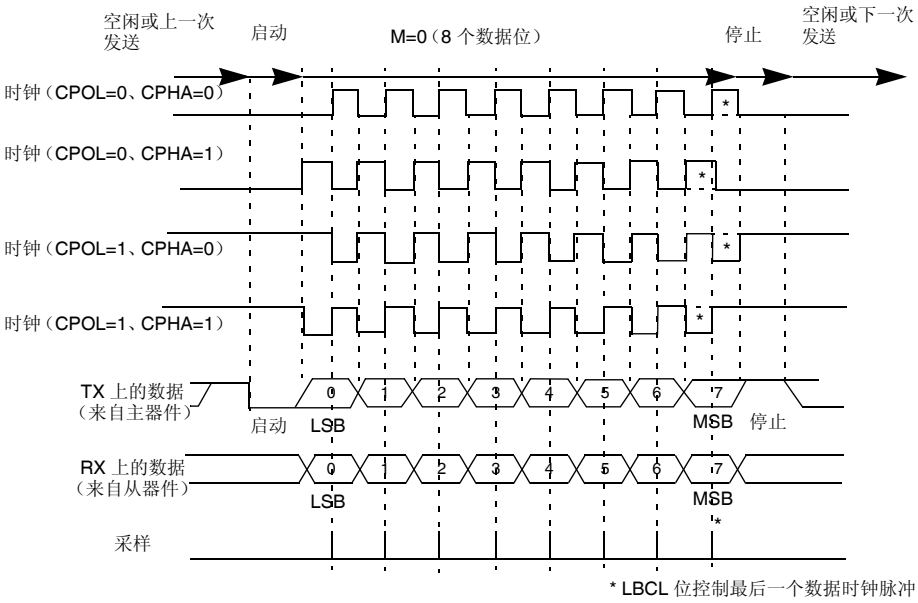


图 259. USART 数据时钟时序图 (M=1)

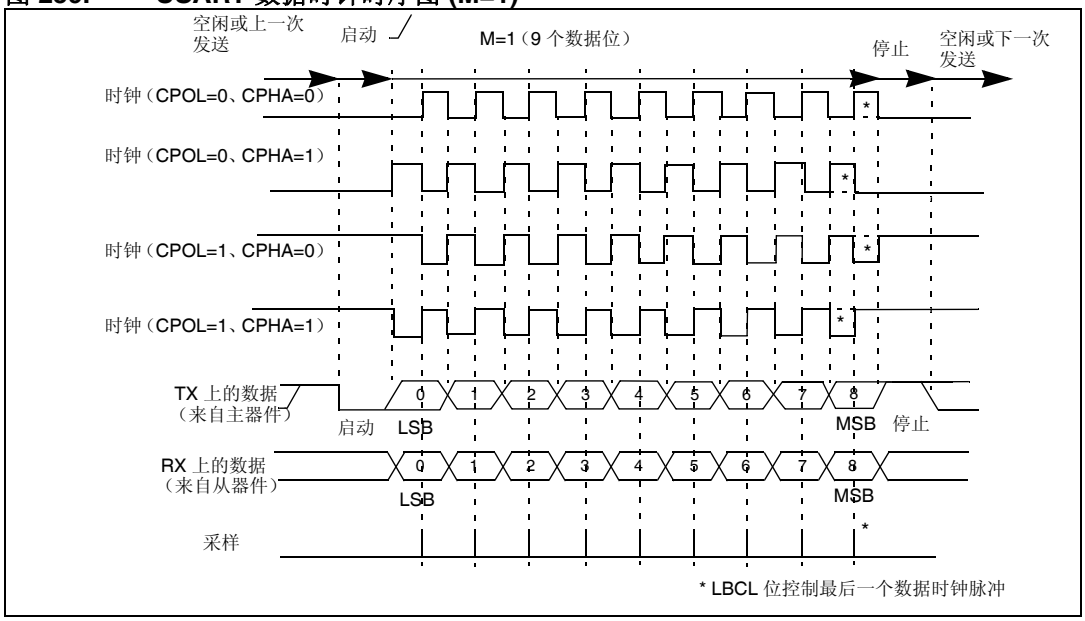
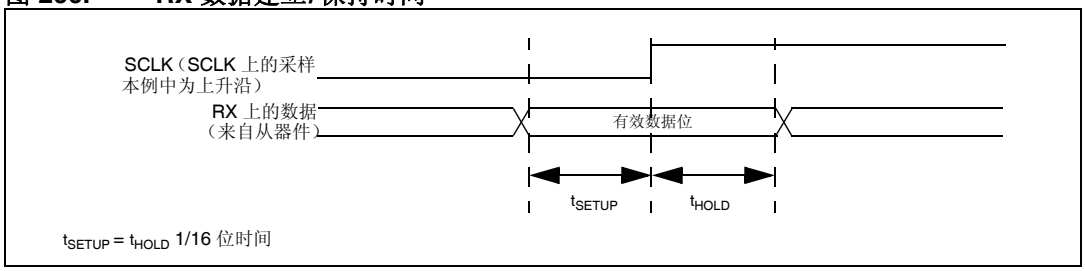


图 260. RX 数据建立/保持时间



注意： *在智能卡模式下，SCLK 的功能有所不同。有关详细信息，请参见智能卡模式一章。*

26.3.10 单线半双工通信

通过将 USART_CR3 寄存器中的 HDSEL 位置 1 来选择单线半双工模式。在此模式下，必须将以下位清零：

- USART_CR2 寄存器中的 LINEN 和 CLKEN 位，
- USART_CR3 寄存器中的 SCEN 和 IREN 位。

USART 可以配置为遵循单线半双工协议，其中 TX 和 RX 线路从内部相连接。使用控制位“HALF DUPLEX SEL”（USART_CR3 寄存器中的 HDSEL 位），可以在半双工通信和全双工通信间进行选择。

一旦向 HDSEL 位写入 1：

- TX 和 RX 线路从内部相连接
- 不能再使用 RX 引脚
- 无数据传输时，TX 引脚始终处于释放状态。因此，它在空闲状态或接收过程中用作标准 I/O。这意味着，必须对 I/O 进行配置，以便在未受 USART 驱动时，使 TX 成为浮空输入（或高电平开漏输出）。

除此之外，通信与正常 USART 模式下的通信相似。此线路上的冲突必须由软件进行管理（例如，使用中央仲裁器）。尤其要注意，发送过程永远不会被硬件封锁，只要数据是在 TE 位置 1 的情况下写入，发送就会持续进行。

26.3.11 智能卡

通过将 USART_CR3 寄存器中的 SCEN 位置 1 来选择智能卡模式。在智能卡模式下，必须将以下位清零：

- USART_CR2 寄存器中的 LINEN 位，
- USART_CR3 寄存器中的 HDSEL 和 IREN 位。

此外，可能需要将 CLKEN 位置 1，以便为智能卡提供时钟。

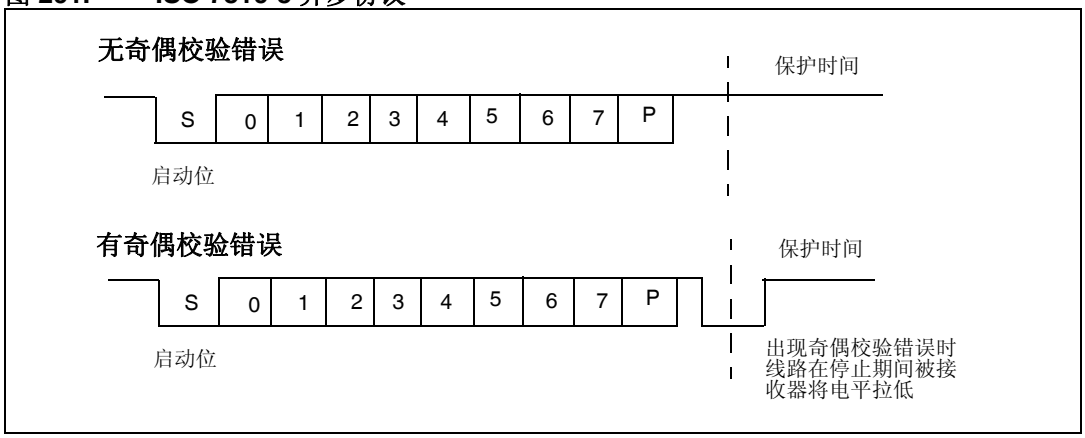
智能卡接口支持符合 ISO 7816-3 标准的异步协议智能卡。USART 应如下所示进行配置：

- 8 个位加奇偶校验：当 USART_CR1 寄存器中 M=1 且 PCE=1 时
- 发送和接收时使用 1.5 个停止位：当 USART_CR2 寄存器中 STOP=11 时。

注意： *接收时也可以选择 0.5 个停止位，但为了避免在两种配置之间切换，建议发送和接收时均使用 1.5 个停止位。*

图 261 显示了有奇偶校验错误和无奇偶校验错误时数据线上情况的示例。

图 261. ISO 7816-3 异步协议



连接到智能卡时，USART 的 TX 输出会驱动一条双向线（它也由智能卡驱动）。必须将 TX 引脚配置为开漏引脚。

智能卡是一个单线半双工通信协议。

- 从发送移位寄存器发送数据会经过至少 1/2 个时钟周期的延迟。正常工作时，已满的发送移位寄存器会在下一个时钟边沿开始移位。在智能卡模式下，此发送过程还会进一步经过 1/2 波特时钟周期的延迟。
- 如果在接收一个使用 0.5 或 1.5 个停止位编程的帧期间检测到奇偶校验错误，则在完成接收帧后，发送线会被拉低一个时钟周期。这是为了向智能卡指出发送到 USART 的数据尚未正确接收。此 NACK 信号（将发送线拉低 1 个时钟周期）会导致发送器端（配置为 1.5 个停止位）出现帧错误。应用程序可根据协议重新发送数据。如果 NACK 控制位置 1，则接收器会向奇偶校验错误发送“NACK”信号；否则不会发送 NACK 信号。

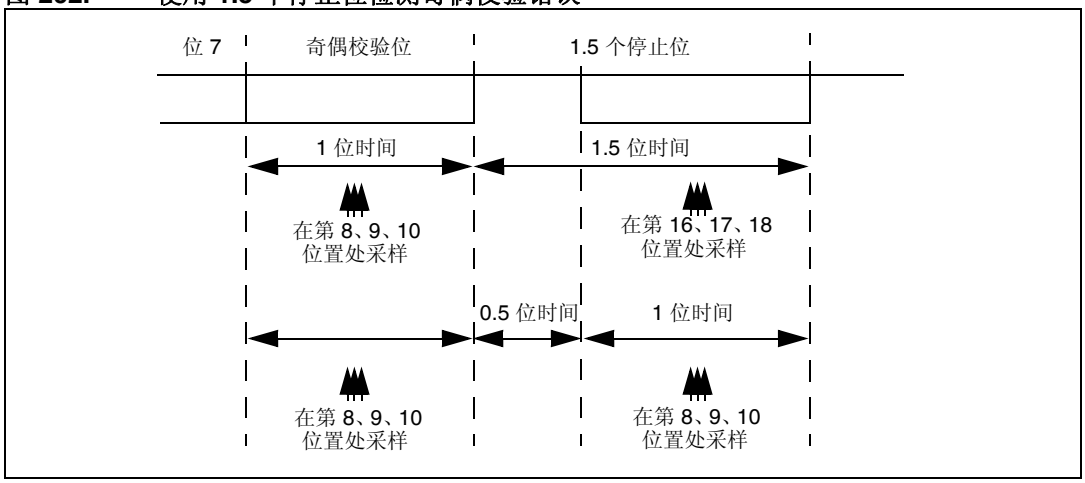
- 通过对保护时间寄存器进行编程，可以延迟 TC 标志的置位。正常工作时，当发送移位寄存器为空且没有新的发送请求出现时，会对 TC 标志进行置位。在智能卡模式下，空的发送移位寄存器会触发保护时间计数器，使其递增计数至保护时间寄存器中的值。在此期间，TC 标志被强制为低电平。当保护时间计数器达到设置值时，TC 置位为高电平。
- TC 标志的释放不受智能卡模式的影响。
- 如果在发送端检测到帧错误（由来自接收器的 NACK 信号引起），则发送端的接收块不会将 NACK 作为起始位进行检测。根据 ISO 协议，接收到的 NACK 信号的持续时间可以是 1 或 2 个时钟周期。
- 在接收端，如果检测到奇偶校验错误并发送了 NACK 信号，则接收端不会将 NACK 作为起始位进行检测。

注意： *中断字符在智能卡模式下无效。带有帧错误的 0x00 数据将被视为数据，而非中断。*

当翻转 TE 位时，不会发送空闲帧。空闲帧（在其它配置中进行了定义）在 ISO 协议中未进行定义。

图 262 详细介绍了 USART 如何对 NACK 信号采样。在本例中，USART 正在发送数据并配置了 1.5 个停止位。USART 的接收部分已被使能，以检查数据的完整性和 NACK 信号。

图 262. 使用 1.5 个停止位检测奇偶校验错误



USART 可以通过 SCLK 输出为智能卡提供时钟。在智能卡模式下，SCLK 仅通过一个 5 位预分频器由内部外设输入时钟提供。分频比在预分频器寄存器 USART_GTPR 中进行配置。SCLK 频率可在 $f_{CK}/2$ 到 $f_{CK}/62$ 之间进行编程，其中 f_{CK} 为外设输入时钟。

26.3.12 IrDA SIR ENDEC 模块

通过将 USART_CR3 寄存器中的 IREN 位置 1 来选择 IrDA 模式。在 IrDA 模式下，必须将以下位清零：

- USART_CR2 寄存器中的 LINEN、STOP 和 CLKEN 位，
- USART_CR3 寄存器中的 SCEN 和 HDSEL 位。

IrDA SIR 物理层规定使用反相归零 (RZI) 调制方案，它以红外光脉冲表示逻辑 0（参见*图 263*）。

SIR 发送编码器用于调制 USART 发出的非归零 (NRZ) 位流。输出脉冲流会发送到外部输出驱动器和红外线 LED。USART 支持的 SIR ENDEC 比特率最高为 115.2Kbps。在正常模式下，所发送的脉冲宽度规定为一个位周期的 3/16。

SIR 接收解码器用于解调由红外探测器发出的归零位流，并将接收到的 NRZ 串行位流输出到 USART。在空闲状态下，解码器输入通常为高电平（标记状态）。发送编码器输出的极性与解码器输入相反。当解码器输入为低电平时，会检测到起始位。

- IrDA 是一个半双工通信协议。如果发送器忙（例如，USART 正在向 IrDA 编码器发送数据），则 IrDA 解码器会忽略 IrDA 接收线上的所有数据；如果接收器忙（例如，USART 正在接收来自 USART 的解码数据），则 IrDA 不会对 USART 发送到 IrDA 的 TX 上的数据进行编码。在接收数据时，应避免同时进行发送，因为这样做可能会破坏要发送的数据。
- “0”作为高电平脉冲发送，而“1”作为“0”发送。在正常模式下，脉冲宽度规定为所选位周期的 3/16（参见*图 264*）。
- SIR 解码器用于将兼容 IrDA 的接收信号转换为 USART 的位流。
- SIR 接收逻辑将高电平状态视为逻辑“1”，将低电平脉冲视为逻辑“0”。
- 发送编码器输出的极性与解码器输入相反。SIR 输出在空闲时处于低电平状态。
- IrDA 规范要求脉冲容忍值要大于 1.41 us。可接受的脉冲宽度可通过寄存器设置。接收器端的干扰检测逻辑会滤除宽度小于 2 个 PSC 周期的脉冲（PSC 是在 IrDA 低功耗波特寄存器 USART_GTPR 中编程的预分频器值）。宽度小于 1 个 PSC 周期的脉冲都将被拒绝，但宽度大于 1 个而小于 2 个周期的脉冲可能被接受也可能被拒绝，而宽度大于 2 个周期的脉冲将被接受作为有效脉冲。当 PSC=0 时，IrDA 编码器/解码器不工作。
- 接收器能够与低功耗发送器进行通信。
- 在 IrDA 模式下，USART_CR2 寄存器中的 STOP 位必须配置为“1 个停止位”。

IrDA 低功耗模式

发送器:

在低功耗模式下，脉冲宽度不再保持为位周期的 3/16。此时的脉冲宽度为低功耗波特率的 3 倍，最小可为 1.42 MHz。通常此值是 1.8432 MHz (1.42 MHz < PSC< 2.12 MHz)。低功耗模式下的可编程分频器会对系统时钟进行分频，以达到此值。

接收器:

在低功耗模式下接收与在正常模式下接收类似。为进行干扰检测，USART 应丢弃持续时间短于 1/PSC 的脉冲。只有当持续时间大于 2 个 IrDA 低功耗波特时钟周期（USART_GTPR 寄存器中的 PSC 值）时，才是有效低电平。

注意： *宽度小于两个但大于一个 PSC 周期的脉冲可能被接受，也可能被拒绝。*

接收器的建立时间应由软件进行管理。IrDA 物理层规范规定发送和接收之间至少要经过 10 ms 的延迟（IrDA 是一个半双工协议）。

图 263. IrDA SIR ENDEC——框图

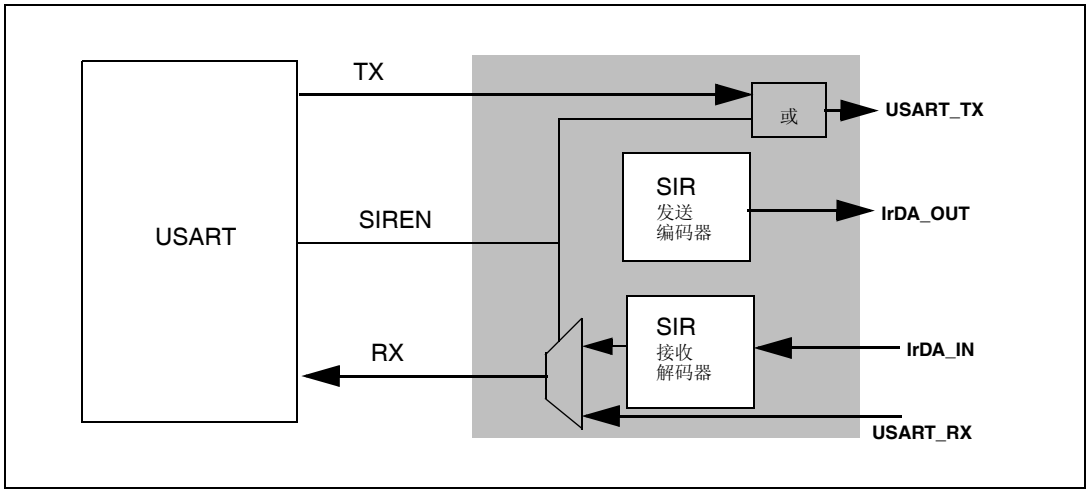
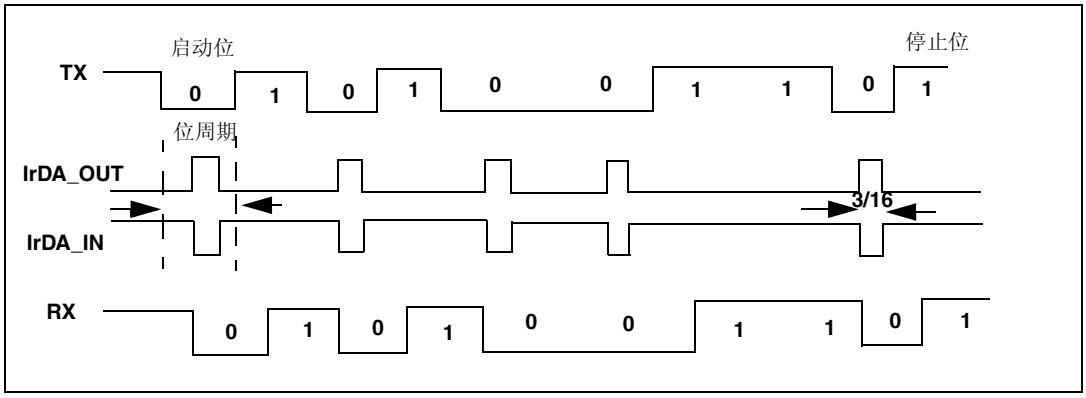


图 264. IrDA 数据调制 (3/16)——正常模式



26.3.13 使用 DMA 进行连续通信

USART 能够使用 DMA 进行连续通信。接收缓冲区和发送缓冲区的 DMA 请求是独立的。

注意： *有关 DMA 控制器可用性的信息，请参见产品规范。如果 DMA 不适用于该产品，应按照第 26.3.2 节或 26.3.3 中的解释说明使用 USART。可以将 USART_SR 寄存器中的 TXE/RXNE 标志清零，从而实现连续通信。*

使用 DMA 进行发送

将 USART_CR3 寄存器中的 DMAT 位置 1 可以使能 DMA 模式进行发送。当 TXE 位置 1 时，可将数据从 SRAM 区（通过 DMA 配置，参见 DMA 部分）加载到 USART_DR 寄存器。要映射一个 DMA 通道以进行 USART 发送，请按以下步骤操作（x 表示通道编号）：

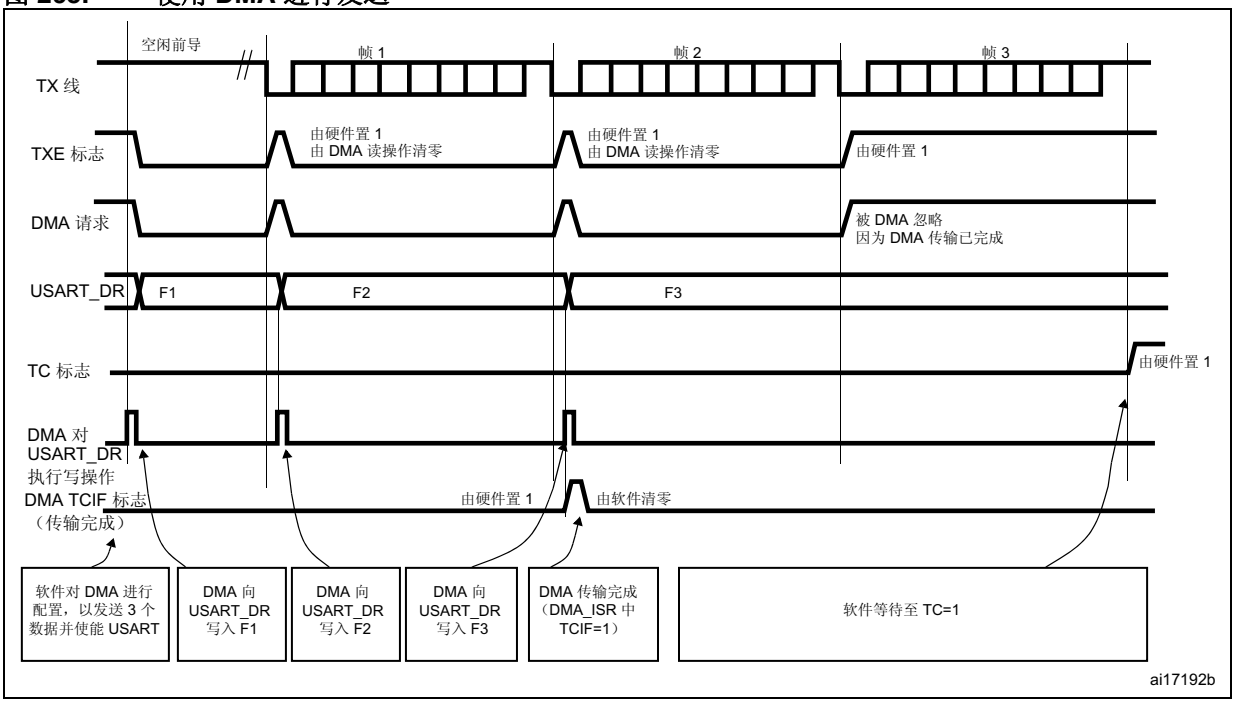
1. 在 DMA 控制寄存器中写入 USART_DR 寄存器地址，将其配置为传输的目标地址。每次发生 TXE 事件后，数据都会从存储器移动到此地址。
2. 在 DMA 控制寄存器中写入存储器地址，将其配置为传输的源地址。每次发生 TXE 事件后，数据都会从这个存储区域加载到 USART_DR 寄存器中。
3. 在 DMA 控制寄存器中配置要传输的总字节数。
4. 在 DMA 寄存器中配置通道优先级
5. 根据应用的需求，在完成一半或全部传输后产生 DMA 中断。

6. 向 SR 寄存器中的 TC 位写入 0，将其清零。
7. 在 DMA 寄存器中激活该通道。

当达到在 DMA 控制器中设置的数据传输量时，DMA 控制器会在 DMA 通道的中断向量上产生一个中断。

在发送模式下，DMA 对所有要发送的数据执行了写操作（DMA_ISR 寄存器中的 TCIF 标志置 1）后，可以对 TC 标志进行监视，以确保 USART 通信已完成。在禁止 USART 或进入停止模式前必须执行此步骤，以避免损坏最后一次发送。软件必须等待直到 TC=1。TC 标志在所有数据发送期间都必须保持清零状态，然后在最后一帧发送结束后由硬件置 1。

图 265. 使用 DMA 进行发送



使用 DMA 进行接收

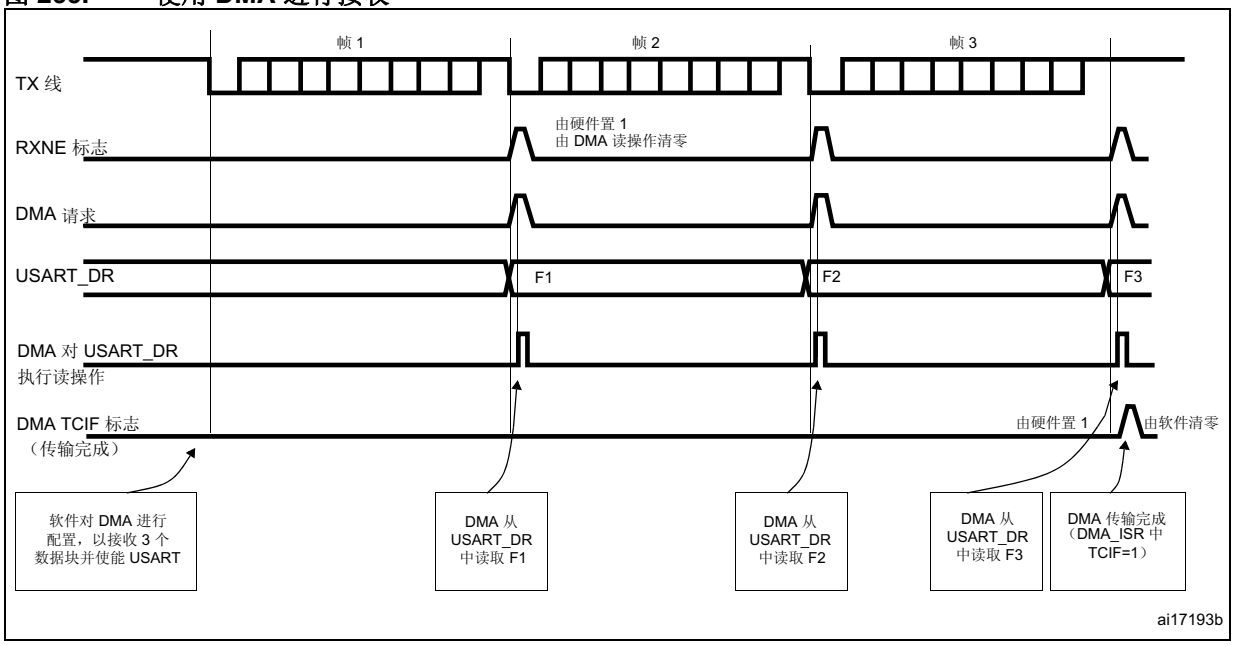
将 USART_CR3 寄存器中的 DMAR 位置 1 可以使能 DMA 模式进行接收。接收数据字节时，数据会从 USART_DR 寄存器加载到 SRAM 区域中（通过 DMA 配置，参见 DMA 规范）。要映射一个 DMA 通道以进行 USART 接收，请按以下步骤操作：

1. 在 DMA 控制寄存器中写入 USART_DR 寄存器地址，将其配置为传输的源地址。每次发生 RXNE 事件后，数据都会从此地址移动到存储器。
2. 在 DMA 控制寄存器中写入存储器地址，将其配置为传输的目标地址。每次发生 RXNE 事件后，数据都会从 USART_DR 寄存器加载到此存储区。
3. 在 DMA 控制寄存器中配置要传输的总字节数。
4. 在 DMA 控制寄存器中配置通道优先级。
5. 根据应用的需求，在完成一半或全部传输后产生中断。
6. 在 DMA 控制寄存器中激活该通道。

当达到在 DMA 控制器中设置的数据传输量时，DMA 控制器会在 DMA 通道的中断向量上产生一个中断。在中断子程序中，USART_CR3 寄存器中的 DMAR 位应由软件清零。

注意： *如果 DMA 用于接收，则不要使能 RXNEIE 位。*

图 266. 使用 DMA 进行接收



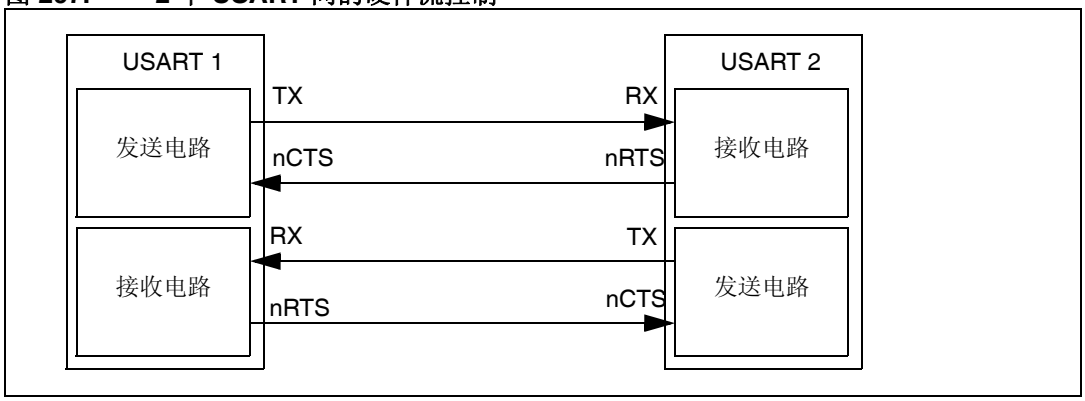
多缓冲区通信中的错误标志和中断生成

在多缓冲区通信中，如果事务中发生任何错误，都会在当前字节后放置错误标志。如果中断使能置 1，则会产生中断。在单字节接收过程中，与 RXNE 一同置位的帧错误、上溢错误和噪声标志具有单独的错误标志中断使能位（USART_CR3 寄存器中的 EIE 位）；如果该位置 1，则会因其中任何一个错误而在当前字节后产生中断。

26.3.14 硬件流控制

使用 nCTS 输入和 nRTS 输出可以控制 2 个器件间的串行数据流。*图 267* 显示了在这种模式下如何连接 2 个器件：

图 267. 2 个 USART 间的硬件流控制

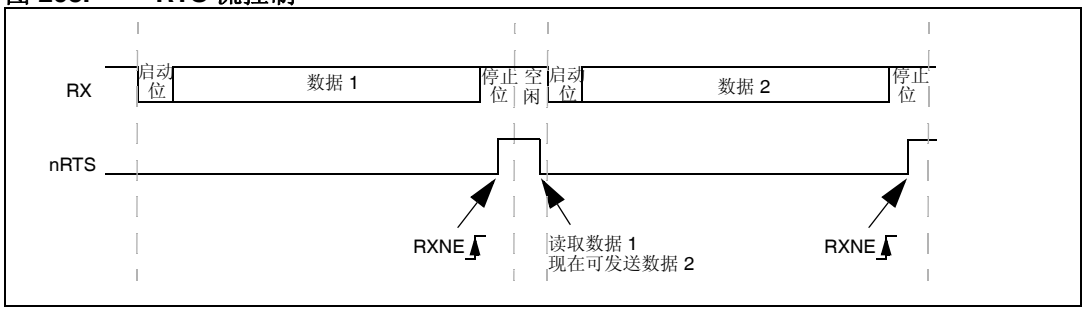


分别向 USART_CR3 寄存器中的 RTSE 位和 CTSE 位写入 1，可以分别使能 RTS 和 CTS 流控制。

RTS 流控制

如果使能 RTS 流控制 (RTSE=1)，只要 USART 接收器准备好接收新数据，便会将 nRTS 变为有效（连接到低电平）。当接收寄存器已满时，会将 nRTS 变为无效，表明发送过程会在当前帧结束后停止。*图 268* 显示了在使能 RTS 流控制的情况下进行通信的示例。

图 268. RTS 流控制

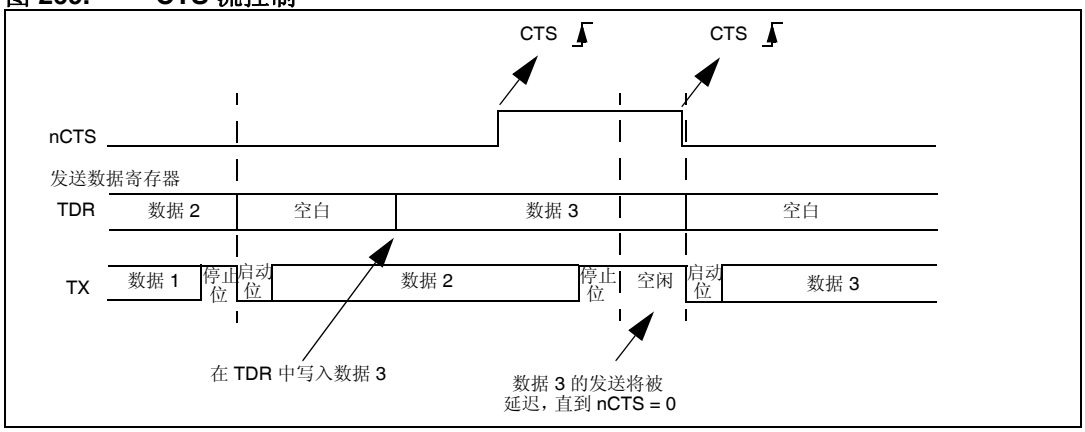


CTS 流控制

如果使能 CTS 流控制 (CTSE=1)，则发送器会在发送下一帧前检查 nCTS。如果 nCTS 有效（连接到低电平），则会发送下一数据（假设数据已准备好发送，即 TXE=0）；否则不会进行发送。如果在发送过程中 nCTS 变为无效，则当前发送完成之后，发送器停止。

当 CTSE=1 时，只要 nCTS 发生变化，CTSIF 状态位便会由硬件自动置 1。这指示接收器是否已准备好进行通信。如果 USART_CR3 寄存器中的 CTSIE 位置 1，则会产生中断。下图显示了在使能 CTS 流控制的情况下进行通信的示例。

图 269. CTS 流控制



注意： ***停止帧的特殊行为：****使能 CTS 流后，发送器发送停止信号时将不检查 nCTS 输入状态。*

26.4 USART 中断

表 121. USART 中断请求

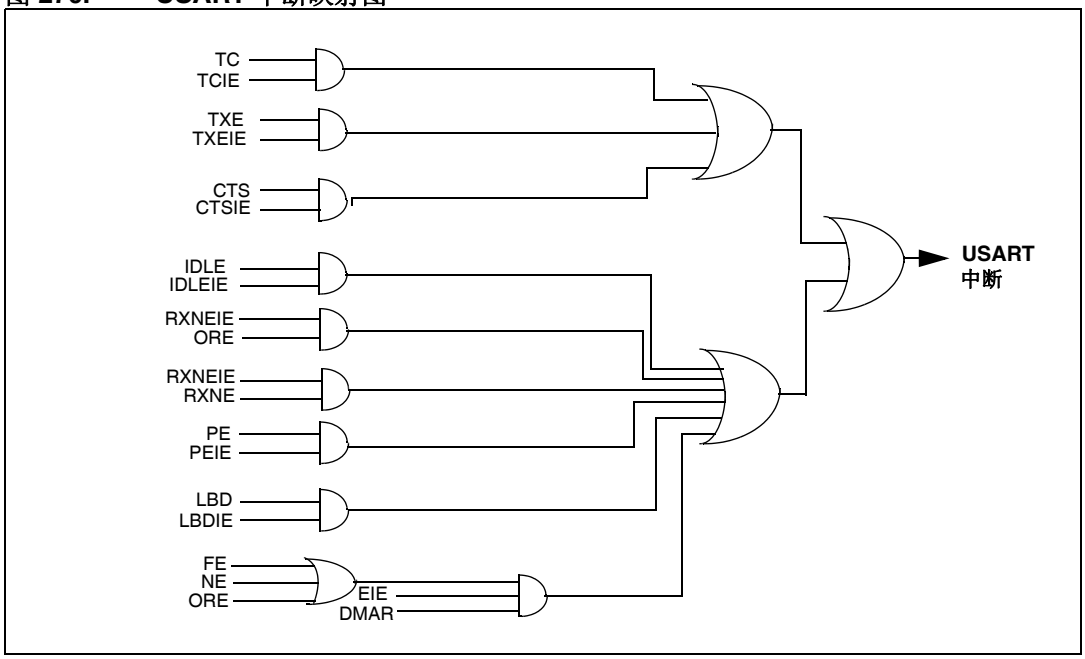
中断事件	事件标志	使能控制位
发送数据寄存器为空	TXE	TXEIE
CTS 标志	CTS	CTSIE
发送完成	TC	TCIE
准备好读取接收到的数据	RXNE	RXNEIE
检测到上溢错误	ORE	
检测到空闲线路	IDLE	IDLEIE
奇偶校验错误	PE	PEIE
断路标志	LBD	LBDIE
多缓冲区通信中的噪声标志、上溢错误和帧错误	NF 或 ORE 或 FE	EIE

USART 中断事件被连接到相同的中断向量（请参见*图 270*）。

- 发送期间：发送完成、清除以发送或发送数据寄存器为空中断。
- 接收期间：空闲线路检测、上溢错误、接收数据寄存器不为空、奇偶校验错误、LIN 断路检测、噪声标志（仅限多缓冲区通信）和帧错误（仅限多缓冲区通信）

如果相应的使能控制位置 1，则这些事件会生成中断。

图 270. USART 中断映射图



26.5 USART 模式配置

表 122. USART 模式配置(1)

USART 模式	USART1	USART2	USART3	UART4	UART5	USART6
异步模式	X	X	X	X	X	X
硬件流控制	X	X	X	NA	NA	X
多缓冲区通信 (DMA)	X	X	X	X	X	X
多处理器通信	X	X	X	X	X	X
同步	X	X	X	NA	NA	X
智能卡	X	X	X	NA	NA	X
半双工（单线模式）	X	X	X	X	X	X
IrDA	X	X	X	X	X	X
LIN	X	X	X	X	X	X

1. X = 支持；NA = 不适用。

26.6 USART 寄存器

有关寄存器说明中使用的缩写，请参见*第 47 页的第 1.1 节*。

外设寄存器可支持半字（16 位）或字（32 位）访问。

26.6.1 状态寄存器 (USART_SR)

Status register

偏移地址：0x00

复位值：0x00C0 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						CTS	LBD	TXE	TC	RXNE	IDLE	ORE	NF	FE	PE
						rc_w0	rc_w0	r	rc_w0	rc_w0	r	r	r	r	r

位 31:10 保留，必须保持复位值

位 9 **CTS**：CTS 标志 (CTS flag)

如果 CTSE 位置 1，当 nCTS 输入变换时，此位由硬件置 1。通过软件将该位清零（通过向该位中写入 0）。如果 USART_CR3 寄存器中 CTSIE=1，则会生成中断。

0：nCTS 状态线上未发生变化

1：nCTS 状态线上发生变化

注意：该位不适用于 UART4 和 UART5。

位 8 **LBD**：LIN 断路检测标志 (LIN break detection flag)

检测到 LIN 断路时，该位由硬件置 1。通过软件将该位清零（通过向该位中写入 0）。如果 USART_CR2 寄存器中 LBDIE = 1，则会生成中断。

0：未检测到 LIN 断路

1：检测到 LIN 断路

注意：如果 LBDIE=1，则当 LBD=1 时生成中断

位 7 **TXE**：发送数据寄存器为空 (Transmit data register empty)

当 TDR 寄存器的内容已传输到移位寄存器时，该位由硬件置 1。如果 USART_CR1 寄存器中 TXEIE 位 = 1，则会生成中断。通过对 USART_DR 寄存器执行写入操作将该位清零。

0：数据未传输到移位寄存器

1：数据传输到移位寄存器

注意：单缓冲区发送期间使用该位。

位 6 **TC**：发送完成 (Transmission complete)

如果已完成对包含数据的帧的发送并且 TXE 置 1，则该位由硬件置 1。如果 USART_CR1 寄存器中 TCIE = 1，则会生成中断。该位由软件序列清零（读取 USART_SR 寄存器，然后写入 USART_DR 寄存器）。TC 位也可以通过向该位写入‘0’来清零。建议仅在多缓冲区通信时使用此清零序列。

0：传送未完成

1：传送已完成

位 5 **RXNE**：读取数据寄存器不为空 (Read data register not empty)

当 RDR 移位寄存器的内容已传输到 USART_DR 寄存器时，该位由硬件置 1。如果 USART_CR1 寄存器中 RXNEIE = 1，则会生成中断。通过对 USART_DR 寄存器执行读入操作将该位清零。RXNE 标志也可以通过向该位写入零来清零。建议仅在多缓冲区通信时使用此清零序列。

0：未接收到数据

1：已准备好读取接收到的数据

位 4 **IDLE**：检测到空闲线路 (IDLE line detected)

检测到空闲线路时，该位由硬件置 1。如果 USART_CR1 寄存器中 IDLEIE = 1，则会生成中断。该位由软件序列清零（读入 USART_SR 寄存器，然后读入 USART_DR 寄存器）。

0：未检测到空闲线路

1：检测到空闲线路

注意：直到 RXNE 位本身已置 1 时（即，当出现新的空闲线路时）IDLE 位才会被再次置 1。

位 3 **ORE**：上溢错误 (Overrun error)

在 RXNE = 1 的情况下，当移位寄存器中当前正在接收的字准备好传输到 RDR 寄存器时，该位由硬件置 1。如果 USART_CR1 寄存器中 RXNEIE = 1，则会生成中断。该位由软件序列清零（读入 USART_SR 寄存器，然后读入 USART_DR 寄存器）。

0：无上溢错误

1：检测到上溢错误

注意：当该位置 1 时，RDR 寄存器的内容不会丢失，但移位寄存器会被覆盖。如果 EIE 位置 1，则在进行多缓冲区通信时会对 ORE 标志生成一个中断。

位 2 **NF**：检测到噪声标志 (Noise detected flag)

当在接收的帧上检测到噪声时，该位由硬件置 1。该位由软件序列清零（读入 USART_SR 寄存器，然后读入 USART_DR 寄存器）。

0：未检测到噪声

1：检测到噪声

注意：如果 EIE 位置 1，则在进行多缓冲区通信时，该位不会生成中断，因为该位出现的时间与本身生成中断的 RXNE 位因 NF 标志而生成的时间相同。

注意：当线路无噪声时，可以通过将 ONEBIT 位编程为 1 提高 USART 对偏差的容差来禁止 NF 标志（请参见第 695 页的第 26.3.5 节：USART 接收器对时钟偏差的容差）。

位 1 **FE**：帧错误 (Framing error)

当检测到去同步化、过度的噪声或中断字符时，该位由硬件置 1。该位由软件序列清零（读入 USART_SR 寄存器，然后读入 USART_DR 寄存器）。

0：未检测到帧错误

1：检测到帧错误或中断字符

注意：该位不会生成中断，因为该位出现的时间与本身生成中断的 RXNE 位出现的时间相同。如果当前正在传输的字同时导致帧错误和上溢错误，则会传输该字，且仅有 ORE 位被置 1。

如果 EIE 位置 1，则在进行多缓冲区通信时会对 FE 标志生成一个中断。

位 0 **PE**：奇偶校验错误 (Parity error)

当在接收器模式下发生奇偶校验错误时，该位由硬件置 1。该位由软件序列清零（读取状态寄存器，然后对 USART_DR 数据寄存器执行读或写访问）。将 PE 位清零前软件必须等待 RXNE 标志被置 1。

如果 USART_CR1 寄存器中 PEIE = 1，则会生成中断。

0：无奇偶校验错误

1：奇偶校验错误

26.6.2 数据寄存器 (USART_DR)

Data register

偏移地址：0x04

复位值：0xXXXX XXXX

位 31:9 保留，必须保持复位值

位 8:0 **DR[8:0]**：数据值

包含接收到数据字符或已发送的数据字符，具体取决于所执行的操作是“读取”操作还是“写入”操作。

因为数据寄存器包含两个寄存器，一个用于发送 (TDR)，一个用于接收 (RDR)，因此它具有双重功能（读和写）。

TDR 寄存器在内部总线和输出移位寄存器之间提供了并行接口（参见图 1）。

RDR 寄存器在输入移位寄存器和内部总线之间提供了并行接口。

在使能奇偶校验位的情况下（USART_CR1 寄存器中的 PCE 位被置 1）进行发送时，由于 MSB 的写入值（位 7 或位 8，具体取决于数据长度）会被奇偶校验位所取代，因此该值不起任何作用。

在使能奇偶校验位的情况下进行接收时，从 MSB 位中读取的值为接收到的奇偶校验位。

26.6.3 波特率寄存器 (USART_BRR)

Baud rate register

注意：如果 TE 或 RE 位分别被禁止，则波特计数器会停止计数。

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
Reserved															
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	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:16 保留，必须保持复位值

位 15:4 **DIV_Mantissa[11:0]**：USARTDIV 的尾数
这 12 个位用于定义 USART 除数 (USARTDIV) 的尾数

位 3:0 **DIV_Fraction[3:0]**：USARTDIV 的小数
这 4 个位用于定义 USART 除数 (USARTDIV) 的小数。当 OVER8 = 1 时，不考虑 DIV_Fraction3 位，且必须将该位保持清零。

26.6.4 控制寄存器 1 (USART_CR1)

Control register 1

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
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OVER8	Reserved	UE	M	WAKE	PCE	PS	PEIE	TXEIE	TCIE	RXNEIE	IDLEIE	TE	RE	RWU	SBK
rw	Res.	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:16 保留，必须保持复位值

位 15 **OVER8**：过采样模式 (Oversampling mode)
0：16 倍过采样
1：8 倍过采样
注意：8 倍过采样在智能卡、IrDA 和 LIN 模式下不可用：当 SCEN=1、IREN=1 或 LINEN=1 时，OVER8 由硬件强制清零。

位 14 保留，必须保持复位值

位 13 **UE**：USART 使能 (USART enable)
该位清零后，USART 预分频器和输出将停止，并会结束当前字节传输以降低功耗。此位由软件置 1 和清零。
0：禁止 USART 预分频器和输出
1：使能 USART

位 12 **M**：字长 (Word length)
该位决定了字长。该位由软件置 1 或清零。
0：1 起始位，8 数据位，n 停止位
1：1 起始位，9 数据位，n 停止位
注意：在数据传输（发送和接收）期间不得更改 M 位

位 11 **WAKE**：唤醒方法 (Wakeup method)
该位决定了 USART 唤醒方法，该位由软件置 1 或清零。
0：空闲线路
1：地址标记

位 10 **PCE**：奇偶校验控制使能 (Parity control enable)
该位选择硬件奇偶校验控制（生成和检测）。使能奇偶校验控制时，计算出的奇偶校验位被插入到 MSB 位置（如果 M=1，则为第 9 位；如果 M=0，则为第 8 位），并对接收到的数据检查奇偶校验位。此位由软件置 1 和清零。一旦该位置 1，PCE 在当前字节的后面处于活动状态（在接收和发送时）。
0：禁止奇偶校验控制
1：使能奇偶校验控制

位 9 **PS**：奇偶校验选择 (Parity selection)

该位用于在使能奇偶校验生成/检测（PCE 位置 1）时选择奇校验或偶校验。该位由软件置 1 和清零。将在当前字节的后面选择奇偶校验。

0：偶校验

1：奇校验

位 8 **PEIE**：PE 中断使能 (PE interrupt enable)

此位由软件置 1 和清零。

0：禁止中断

1：当 USART_SR 寄存器中 PE=1 时，生成 USART 中断

位 7 **TXEIE**：TXE 中断使能 (TXE interrupt enable)

此位由软件置 1 和清零。

0：禁止中断

1：当 USART_SR 寄存器中 TXE=1 时，生成 USART 中断。

位 6 **TCIE**：传送完成中断使能 (Transmission complete interrupt enable)

此位由软件置 1 和清零。

0：禁止中断

1：当 USART_SR 寄存器中 TC=1 时，生成 USART 中断

位 5 **RXNEIE**：RXNE 中断使能 (RXNE interrupt enable)

此位由软件置 1 和清零。

0：禁止中断

1：当 USART_SR 寄存器中 ORE=1 或 RXNE=1 时，生成 USART 中断

位 4 **IDLEIE**：IDLE 中断使能 (IDLE interrupt enable)

此位由软件置 1 和清零。

0：禁止中断

1：当 USART_SR 寄存器中 IDLE=1 时，生成 USART 中断

位 3 **TE**：发送器使能 (Transmitter enable)

该位使能发送器。该位由软件置 1 和清零。

0：禁止发送器

1：使能发送器

注意：1：除了在智能卡模式下以外，传送期间 TE 位上的“0”脉冲（“0”后紧跟的是“1”）会在当前字的后面发送一个报头（空闲线路）。

2：当 TE 置 1 时，在发送开始前存在 1 位的时间延迟。

位 2 **RE**：接收器使能 (Receiver enable)

该位使能接收器。该位由软件置 1 和清零。

0：禁止接收器

1：使能接收器并开始搜索起始位

位 1 **RWU**：接收器唤醒 (Receiver wakeup)

该位决定 USART 是否处于静音模式。该位由软件置 1 和清零，并可在识别出唤醒序列时由硬件清零。

0：接收器处于活动模式

1：接收器处于静音模式

注意：1：选择静音模式前（通过将 RWU 位置 1），USART 必须首先接收一个数据字节，否则当由空闲线路检测到唤醒时，它无法于静音模式下正常工作。

2：在地址标记检测唤醒配置（WAKE 位 = 1）中，RXNE 位置 1 时，RWU 位不能由软件进行修改。

位 0 **SBK**：发送断路 (Send break)

该位用于发送断路字符。该位可由软件置 1 和清零。该位应由软件置 1，并在断路停止位期间由硬件重置。

0：不发送断路字符

1：将发送断路字符

26.6.5 控制寄存器 2 (USART_CR2)

Control register 2

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
Reserved															
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	rw	rw	rw	rw	rw

位 31:15 保留，必须保持复位值

位 14 **LINEN**：LIN 模式使能 (LIN mode enable)

此位由软件置 1 和清零。

0：禁止 LIN 模式

1：使能 LIN 模式

LIN 模式可以使用 USART_CR1 寄存器中的 SBK 位发送 LIN 同步断路（13 个低位），并可检测 LIN 同步断路。

位 13:12 **STOP**：停止位 (STOP bit)

这些位用于编程停止位。

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

注意：0.5 个停止位和 1.5 个停止位不适用于 UART4 和 UART5。

位 11 **CLKEN**：时钟使能 (Clock enable)

该位允许用户使能 SCLK 引脚。

0：禁止 SCLK 引脚

1：使能 SCLK 引脚

该位不适用于 UART4 和 UART5。

位 10 **CPOL**：时钟极性 (Clock polarity)

该位允许用户在同步模式下选择 SCLK 引脚上时钟输出的极性。它与 CPHA 位结合使用可获得所需的时钟/数据关系

0：空闲时 SCLK 引脚为低电平。

1：空闲时 SCLK 引脚为高电平。

该位不适用于 UART4 和 UART5。

位 9 **CPHA**：时钟相位

该位允许用户在同步模式下选择 SCLK 引脚上时钟输出的相位。它与 CPOL 位结合使用可获得所需的时钟/数据关系（请参见图 *258* 至 *259*）

0：在时钟第一个变化沿捕获数据

1：在时钟第二个变化沿捕获数据

注意：该位不适用于 UART4 和 UART5。

位 8 **LBCL**：最后一个位时钟脉冲 (Last bit clock pulse)

该位允许用户在同步模式下选择与发送的最后一个数据位 (MSB) 关联的时钟脉冲是否必须在 SCLK 引脚上输出。

0：最后一个数据位的时钟脉冲不在 SCLK 引脚上输出

1：最后一个数据位的时钟脉冲在 SCLK 引脚上输出

注意：1：最后一位为发送的第 8 或第 9 个数据位，具体取决于 USART_CR1 寄存器中 M 位所选择的 8 位或 9 位格式。

2：该位不适用于 UART4 和 UART5。

位 7 保留，必须保持复位值

位 6 **LBDIE**：LIN 断路检测中断使能 (LIN break detection interrupt enable)

断路中断屏蔽（使用断路分隔符进行断路检测）

0：禁止中断

1：当 USART_SR 寄存器中 LBD = 1 时，生成中断

位 5 **LBDL**：*lin* 断路检测长度 (*lin* break detection length)

该位用于选择 11 位断路检测或 10 位断路检测。

0：10 位断路检测

1：11 位断路检测

位 4 保留，必须保持复位值

位 3:0 **ADD[3:0]**：USART 节点的地址

该位域用于指定 USART 节点的地址。

将在多处理器通信时于静音模式下使用该位域，以通过地址标记检测进行唤醒。

注意：使能发送器时不应对这 3 个位（CPOL、CPHA、LBCL）进行写操作。

26.6.6 控制寄存器 3 (USART_CR3)

Control register 3

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
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				ONEBIT	CTSIE	CTSE	RTSE	DMAT	DMAR	SCEN	NACK	HDSEL	IRLP	IREN	EIE
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:12 保留，必须保持复位值

位 11 **ONEBIT**：一个采样位方法使能 (One sample bit method enable)

该位允许用户选择采样方法。选择一个采样位方法后，将禁止噪声检测标志 (NF)。

0：三个采样位方法

1：一个采样位方法

位 10 **CTSIE**：CTS 中断使能 (CTS interrupt enable)

0：禁止中断

1：当 USART_SR 寄存器中 CTS = 1 时，生成中断

注意：该位不适用于 UART4 和 UART5。

位 9 **CTSE**：CTS 使能 (CTS enable)

0：禁止 CTS 硬件流控制

1：使能 CTS 模式，仅当 nCTS 输入有效（连接到 0）时才发送数据。如果在发送数据时使 nCTS 输入无效，会在停止之前完成发送。如果使 nCTS 有效时数据已写入数据寄存器，则将延迟发送，直到 nCTS 有效。

注意：该位不适用于 UART4 和 UART5。

位 8 **RTSE**：RTS 使能 (RTS enable)

0：禁止 RTS 硬件流控制

1：使能 RTS 中断，仅当接收缓冲区中有空间时才会请求数据。发送完当前字符后应停止发送数据。可以接收数据时使 nRTS 输出有效（连接到 0）。

注意：该位不适用于 UART4 和 UART5。

位 7 **DMAT**：DMA 使能发送器 (DMA enable transmitter)

该位由软件置 1/ 复位。

1：针对发送使能 DMA 模式。

0：针对发送禁止 DMA 模式。

位 6 **DMAR**：DMA 使能接收器 (DMA enable receiver)

该位由软件置 1/ 复位。

1：针对接收使能 DMA 模式

0：针对接收禁止 DMA 模式

位 5 **SCEN**：智能卡模式使能 (Smartcard mode enable)

该位用于使能智能卡模式。

0：禁止智能卡模式

1：使能智能卡模式

注意：该位不适用于 UART4 和 UART5。

位 4 **NACK**：智能卡 NACK 使能 (Smartcard NACK enable)

0：出现奇偶校验错误时禁止 NACK 发送

1：出现奇偶校验错误时使能 NACK 发送

注意：该位不适用于 UART4 和 UART5。

位 3 **HDSEL**：半双工选择 (Half-duplex selection)

选择单线半双工模式

0：未选择半双工模式

1：选择半双工模式

位 2 **IRLP**：IrDA 低功耗 (IrDA low-power)

该位用于选择正常模式和低功耗 IrDA 模式

0：正常模式

1：低功耗模式

位 1 **IREN**：IrDA 模式使能 (IrDA mode enable)

此位由软件置 1 和清零。

0：禁止 IrDA

1：使能 IrDA

位 0 **EIE**：错误中断使能 (Error interrupt enable)

对于多缓冲区通信（USART_CR3 寄存器中 DMAR = 1），如果发生帧错误、上溢错误或出现噪声标志（USART_SR 寄存器中 FE = 1 或 ORE = 1 或 NF = 1），则需要使用错误中断使能位来使能中断生成。

0：禁止中断

1：当 USART_CR3 寄存器中的 DMAR = 1 并且 USART_SR 寄存器中的 FE = 1 或 ORE = 1 或 NF = 1 时，将生成中断。

26.6.7 保护时间和预分频器寄存器 (USART_GTPR)

Guard time and prescaler register

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

位 31:16 保留，必须保持复位值

位 15:8 **GT[7:0]**：保护时间值 (Guard time value)

该位域提供保护时间值 （以波特时钟数为单位）。

该位用于智能卡模式。经过此保护时间后，发送完成标志置 1。

注意：该位不适用于 UART4 和 UART5。

位 7:0 **PSC[7:0]**：预分频器值

– **在 IrDA 低功耗模式下：**

PSC[7:0] = IrDA 低功耗波特率

用于编程预分频器，进行系统时钟分频以获得低功耗频率：

使用寄存器中给出的值（8 个有效位）对源时钟进行分频：

00000000：保留 - 不编程此值

源时钟 1 分频

00000010：源时钟 2 分频

...

– **在正常 IrDA 模式下：**PSC 必须设置为 00000001。

– **在智能卡模式下：**

PSC[4:0]：预分频器值

用于编程预分频器，进行系统时钟分频以提供智能卡时钟。

将寄存器中给出的值（5 个有效位）乘以 2 得出源时钟频率的分频系数：

00000：保留 - 不编程此值

00001：源时钟 2 分频

00010：源时钟 4 分频

00011：源时钟 6 分频

...

注意：1：如果使用智能卡模式，则位 [7:5] 不起作用。

2：该位不适用于 UART4 和 UART5。

26.6.8 USART 寄存器映射

下表提供了 USART 寄存器映射和复位值。

表 123. USART 寄存器映射和复位值

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**USART_SR**	Reserved																						CTS	LBD	TXE	TC	RXNE	IDLE	ORE	NF	FE	PE
	Reset value																							0	0	1	1	0	0	0	0	0	0
0x04	**USART_DR**	Reserved																							DR[8:0]								
	Reset value																								0	0	0	0	0	0	0	0	0
0x08	**USART_BRR**	Reserved																DIV_Mantissa[15:4]												DIV_Fraction [3:0]			
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	**USART_CR1**	Reserved																OVER8	Reserved	UE	M	WAKE	PCE	PS	PEIE	TXEIE	TCIE	RXNEIE	IDLEIE	TE	RE	RWU	SBK
	Reset value																	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	**USART_CR2**	Reserved																	LINEN	STOP [1:0]		CLKEN	CPOL	CPHA	LBCL	Reserved	LBDIE	LBDL	Reserved	ADD[3:0]			
	Reset value																		0	0	0	0	0	0	0		0	0		0	0	0	0
0x14	**USART_CR3**	Reserved																				ONEBIT	CTSIE	CTSE	RTSE	DMAT	DMAR	SCEN	NACK	HDSEL	IRLP	IREN	EIE
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0
0x18	**USART_GTPR**	Reserved																GT[7:0]												PSC[7:0]			
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

有关寄存器边界地址的信息，请参见*第 52 页的表 2*。

27 串行外设接口 (SPI)

除非特别说明，否则本部分适用于整个 STM32F4xx 系列。

27.1 SPI 简介

SPI 接口提供两个主要功能，支持 SPI 协议或 I^2S 音频协议。默认情况下，选择的是 SPI 功能。可通过软件将接口从 SPI 切换到 I^2S。

串行外设接口 (SPI) 可与外部器件进行半双工/全双工的同步串行通信。该接口可配置为主模式，在这种情况下，它可为外部从器件提供通信时钟 (SCK)。该接口还能够在多主模式配置下工作。

它可用于多种用途，包括基于双线的单工同步传输，其中一条可作为双向数据线，或使用 CRC 校验实现可靠通信。

I^2S 也是同步串行通信接口。它可满足四种不同音频标准的要求，包括 I^2S Philips 标准、MSB 和 LSB 对齐标准，以及 PCM 标准。它可在全双工模式（使用 4 引脚）或半双工模式（使用 3 个引脚）下作为从器件或主器件工作。当 I^2S 配置为通信主模式时，该接口可以向外部从器件提供主时钟。

警告： **由于部分 SPI1 和 SPI3/I2S3 引脚可能会映射到 JTAG 接口所使用的部分引脚上（SPI1_NSS 映射到 JTDI、SPI3_NSS/I2S3_WS 映射到 JTDI 以及 SPI3_SCK/I2S3_CK 映射到 JTDO），用户可以：**
– 将 SPI/I2S 映射到其它引脚
–（调试应用时）在配置 SPI 引脚前禁止 JTAG 接口而使能 SWD 接口，或者
– 禁止 JTAG/SWD 接口（脱离调试，应用单独运行时）。
有关配置 JTAG/SWD 接口引脚的更多信息，请参见*第 7.3.2 节：I/O 引脚复用器和映射*。

27.2 SPI 和 I^2S 主要特性

27.2.1 SPI 特性

- 基于三条线的全双工同步传输
- 基于双线的单工同步传输，其中一条可作为双向数据线
- 8 位或 16 位传输帧格式选择
- 主模式或从模式操作
- 多主模式功能
- 8 个主模式波特率预分频器（最大值为 $f_{PCLK}/2$）
- 从模式频率（最大值为 $f_{PCLK}/2$）
- 对于主模式和从模式都可实现更快的通信
- 对于主模式和从模式都可通过硬件或软件进行 NSS 管理：动态切换主/从操作
- 可编程的时钟极性和相位
- 可编程的数据顺序，最先移位 MSB 或 LSB
- 可触发中断的专用发送和接收标志
- SPI 总线忙状态标志
- SPI TI 模式
- 用于确保可靠通信的硬件 CRC 功能：
  - 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、上溢和 CRC 错误标志
- 具有 DMA 功能的 1 字节发送和接收缓冲器：发送和接收请求

27.2.2 I^2S 特性

- 全双工通信
- 半双工通信（仅作为发送器或接收器）
- 主模式或从模式操作
- 8 位可编程线性预分频器，可实现精确的音频采样频率（从 8 kHz 到 192 kHz）
- 数据格式可以是 16 位、24 位或 32 位
- 数据包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）
- 可编程的时钟极性（就绪时的电平状态）
- 从发送模式下的下溢标志、接收模式下的上溢标志（主模式和从模式），以及接收和发送模式下的帧错误标志（仅从模式）
- 发送和接收使用同一个 16 位数据寄存器
- 支持的 I^2S 协议：
  - I^2S Phillps 标准
  - MSB 对齐标准（左对齐）
  - LSB 对齐标准（右对齐）
  - PCM 标准（在 16 位通道帧或扩展为 32 位通道帧的 16 位数据帧上进行短帧和长帧同步）

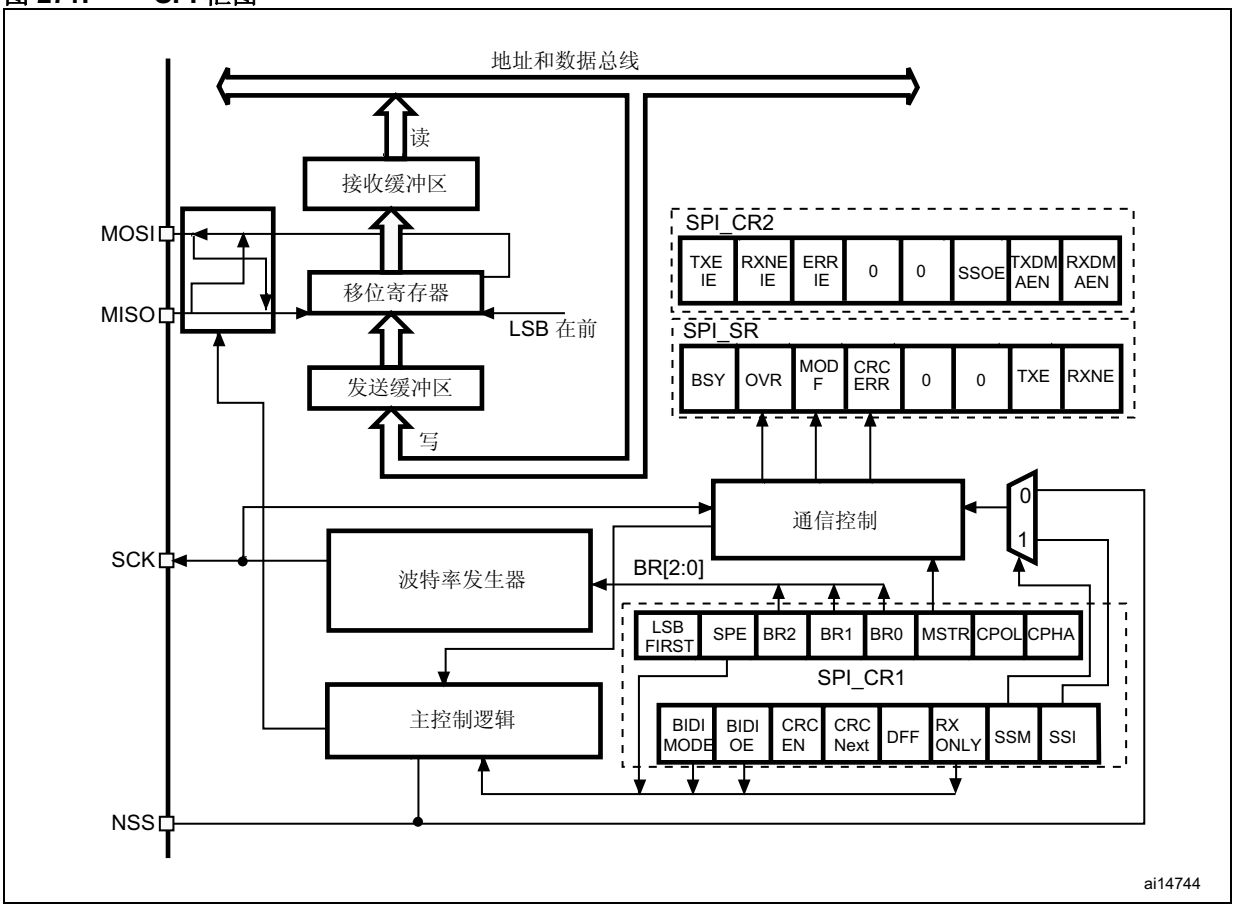
- 数据方向始终为 MSB 在前
- 用于发送和接收的 DMA 功能（16 位宽）
- 可输出主时钟以驱动外部音频元件。比率固定为 256 × F_S（其中 F_S 为音频采样频率）
- 两个 I^2S（I2S2 和 I2S3）均有专用的 PLL (PLLI2S)，可生成更为精确的时钟。
- I^2S（I2S2 和 I2S3）时钟可由 I2S_CKIN 引脚上的外部时钟提供。

27.3 SPI 功能说明

27.3.1 一般说明

SPI 的框图如*图 271* 所示。

图 271. SPI 框图

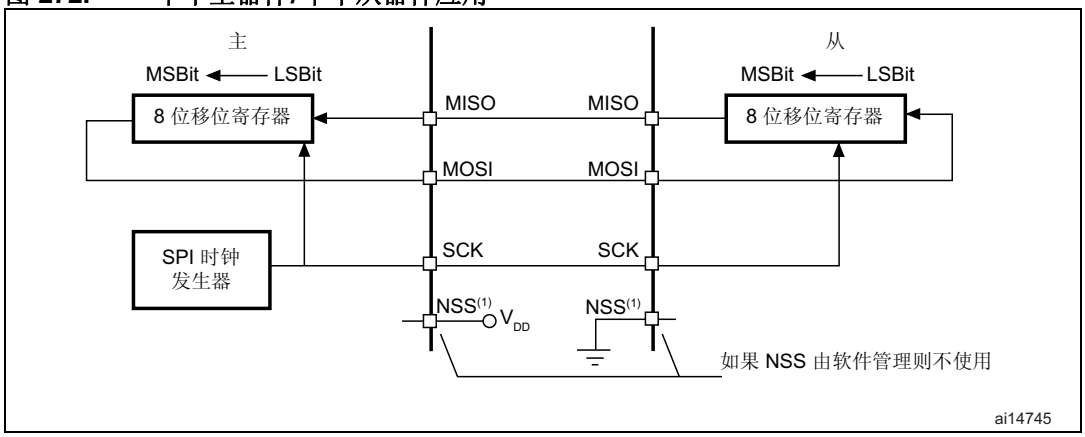


通常，SPI 通过 4 个引脚与外部器件连接：

- MISO：主输入/从输出数据。此引脚可用于在从模式下发送数据和在主模式下接收数据。
- MOSI：主输出/从输入数据。此引脚可用于在主模式下发送数据和在从模式下接收数据。
- SCK：用于 SPI 主器件的串行时钟输出以及 SPI 从器件的串行时钟输入。
- NSS：从器件选择。这是用于选择从器件的可选引脚。此引脚用作“片选”，可让 SPI 主器件与从器件进行单独通信，从而并避免数据线上的竞争。从器件的 NSS 输入可由主器件上的标准 IO 端口驱动。NSS 引脚在使能（SSOE 位）时还可用作输出，并可在 SPI 处于主模式配置时驱动为低电平。通过这种方式，只要器件配置成 NSS 硬件管理模式，所有连接到该主器件 NSS 引脚的其它器件 NSS 引脚都将呈现低电平，并因此而作为从器件。当配置为主模式，且 NSS 配置为输入（MSTR=1 且 SSOE=0）时，如果 NSS 拉至低电平，SPI 将进入主模式故障状态：MSTR 位自动清零，并且器件配置为从模式（参见*第 743 页的第 27.3.10 节：错误标志*）。

图 272 给出了单个主器件和单个从器件之间的互连的基本示例。

图 272. 单个主器件/单个从器件应用



1. 此处，NSS 引脚配置为输入。

MOSI 引脚连接在一起，MISO 引脚连接在一起。通过这种方式，主器件和从器件之间以串行方式传输数据（最高有效位在前）。

通信始终由主器件发起。当主器件通过 MOSI 引脚向从器件发送数据时，从器件同时通过 MISO 引脚做出响应。这是一个数据输出和数据输入都由同一时钟进行同步的 1 全双工通信过程。

从器件选择 (NSS) 引脚管理

可以使用 SPI_CR1 寄存器中的 SSM 位设置硬件或软件管理从器件选择。

- 软件管理 NSS (SSM = 1)

  从器件选择信息在内部由 SPI_CR1 寄存器中的 SSI 位的值驱动。外部 NSS 引脚空闲，可供其它应用使用。

- 硬件管理 NSS (SSM = 0)

  根据 NSS 输出配置（SPI_CR1 寄存器中的 SSOE 位），硬件管理 NSS 有两种模式。

  – NSS 输出使能（SSM = 0，SSOE = 1）

    仅当器件在主模式下工作时才使用此配置。当主器件开始通信时，NSS 信号驱动为低电平，并保持到 SPI 被关闭为止。

  – NSS 输出禁止（SSM = 0，SSOE = 0）

    对于在主模式下工作的器件，此配置允许多主模式功能。对于设置为从模式的器件，NSS 引脚用作传统 NSS 输入：在 NSS 为低电平时片选该从器件，在 NSS 为高电平时取消对它的片选。

时钟相位和时钟极性

通过 SPI_CR1 寄存器中的 CPOL 和 CPHA 位，可以用软件选择四种可能的时序关系。CPOL（时钟极性）位控制不传任何数据时的时钟电平状态。此位对主器件和从 器件都有作用。如果复位 CPOL，SCK 引脚在空闲状态处于低电平。如果将 CPOL 置 1，SCK 引脚在空闲状态处于高电平。

如果将 CPHA（时钟相位）位置 1，则 SCK 引脚上的第二个边沿（如果复位 CPOL 位，则为下降沿；如果将 CPOL 位置 1，则为上升沿）对 MSBit 采样。即，在第二个时钟边沿锁存数据。如果复位 CPHA 位，则 SCK 引脚上的第一个边沿（如果将 CPOL 位置 1，则为下降沿；如果复位 CPOL 位，则为上升沿）对 MSBit 采样。即，在第一个时钟边沿锁存数据。

CPOL（时钟极性）和 CPHA（时钟相位）位的组合用于选择数据捕获时钟边沿。

图 273 显示了在 CPHA 和 CPOL 位的四种组合下的 SPI 传输。可以将该图解释为主器件或从器件时序图，其中 SCK 引脚、MISO 引脚、MOSI 引脚直接连接在主器件和从器件之间。

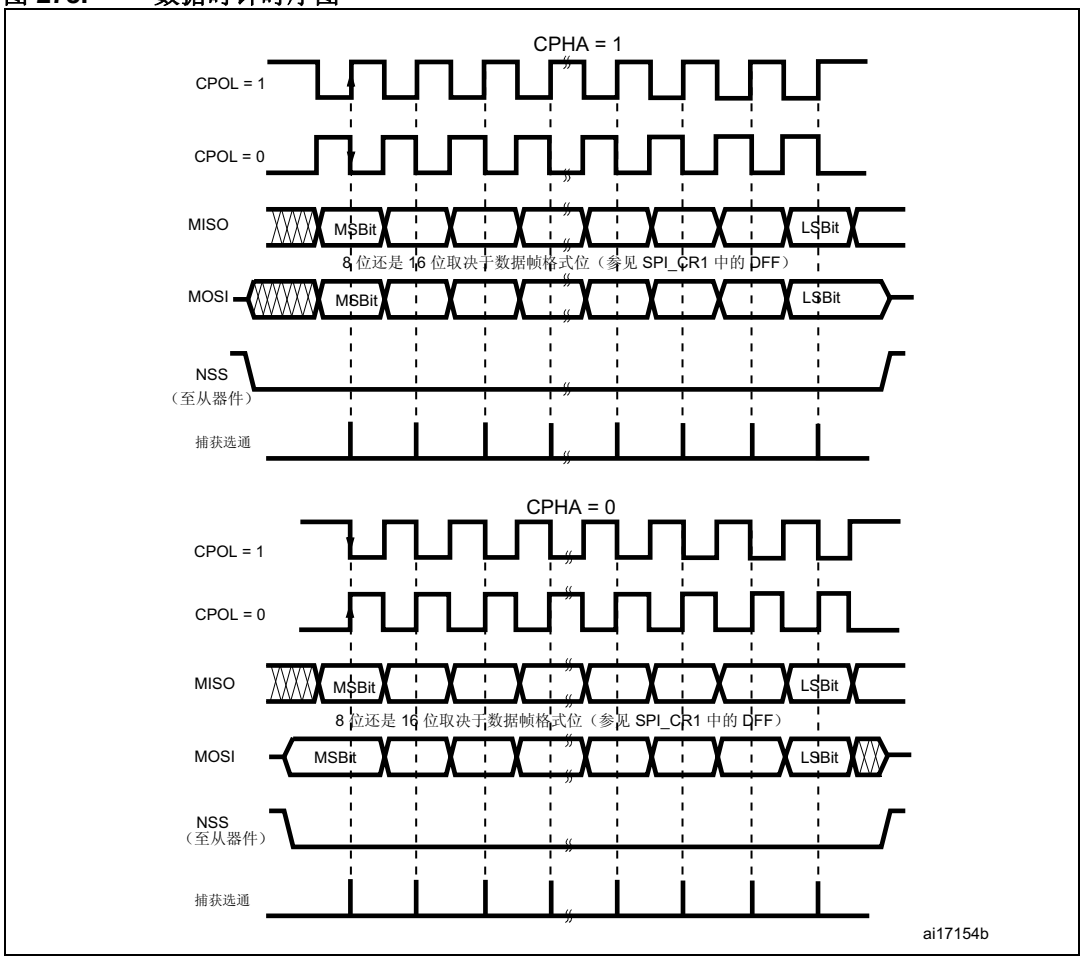
注意： *在切换 CPOL/CPHA 位之前，必须通过复位 SPE 位来关闭 SPI。*

必须以同一时序模式对主器件和从器件进行编程。

SCK 的空闲状态必须与 SPI_CR1 寄存器中选择的极性相对应（如果 CPOL=1，则上拉 SCK；如果 CPOL=0，则下拉 SCK）。

通过 SPI_CR1 寄存器中的 DFF 位选择数据帧格式（8 或 16 位），该格式决定了发送/接收过程中的数据长度。

图 273. 数据时钟时序图



1. 所示的时序为 SPI_CR1 寄存器中的 LSBFIRST 位复位时的时序。

数据帧格式

移出数据时 MSB 在前还是 LSB 在前取决于 SPI_CR1 寄存器中 LSBFIRST 位的值。

每个数据帧的长度均为 8 位或 16 位，具体取决于使用 SPI_CR1 寄存器中的 DFF 位。所选的数据帧格式适用于发送和/或接收。

27.3.2 把 SPI 配置成从器件

在从模式配置中，从 SCK 引脚上接收主器件的串行时钟。SPI_CR1 寄存器的 BR[2:0] 位中设置的值不会影响数据传输率。

注意： *建议在主器件发送时钟前使能 SPI 从器件。否则，数据传输可能会不正常。在主时钟的第一个边沿到来之前或者正在进行的通信结束之前，从器件的数据寄存器就需要准备就绪。在使能从器件和主器件之前，必须将通信时钟的极性设置为空闲时的时钟电平。*

按照下面的步骤把 SPI 模式配置成从模式：

步骤

1. 设置 DFF 位，以定义 8 或 16 位数据帧格式。
2. 选择 CPOL 和 CPHA 位，以定义数据传输和串行时钟之间的关系（四种关系中的一种）（参见*图 273*）。要实现正确的数据传输，必须以相同方式在从器件和主器件中配置 CPOL 和 CPHA 位。如果通过 SPI_CR2 寄存器中的 FRF 位选择 TI 模式，则不需要此步骤。
3. 帧格式（MSB 在前或 LSB 在前取决于 SPI_CR1 寄存器中 LSBFIRST 位的值）必须与主器件的帧格式相同。如果选择 TI 模式，则不需要此步骤。
4. 在硬件模式下（参见*第 725 页的从器件选择 (NSS) 引脚管理*），NSS 引脚在整个字节发送序列期间都必须连接到低电平。在 NSS 软件模式下，将 SPI_CR1 寄存器中的 SSM 位置 1，将 SSI 位清零。如果选择 TI 模式，则不需要此步骤。
5. 将 SPI_CR2 寄存器中的 FRF 位置 1，以选择 TI 模式协议进行串行通信。
6. 将 MSTR 位清零，并将 SPE 位置 1（两个位均在 SPI_CR1 寄存器中）。

在此配置中，MOSI 引脚为数据输入，MISO 引脚为数据输出。

发送序列

数据字节在写周期内被并行加载到发送缓冲区中。

当从器件在其 MOSI 引脚上收到时钟信号和数据的最高有效位时，发送序列开始。其余位（8 位数据帧格式中的 7 个位，16 位数据帧格式中的 15 个位）将加载到移位寄存器中。SPI_SR 寄存器中的 TXE 标志在数据从发送缓冲区传输到移位寄存器时置 1，并且在 SPI_CR2 寄存器中的 TXEIE 位置 1 时将生成中断。

接收序列

对于接收器，在数据传输完成时：

- 移位寄存器中的数据将传输到接收缓冲区，并且 RXNE 标志（SPI_SR 寄存器）置 1
- 如果 SPI_CR2 寄存器中的 RXNEIE 位置 1，则生成中断。

在出现最后一个采样时钟边沿后，RXNE 位置 1，移位寄存器中接收的数据字节被拷贝到接收缓冲区中。当读取 SPI_DR 寄存器时，SPI 外设将返回此缓冲值。

通过读取 SPI_DR 寄存器将 RXNE 位清零。

从模式下的 SPI TI 协议

在从模式下，SPI 接口与 TI 协议兼容。可以使用 SPI_CR2 寄存器的 FRF 位来配置从 SPI 串行通信，以兼容此协议。

时钟极性和相位都被强制为遵循 TI 协议，和 SPI_CR1 中的设置无关。NSS 管理也特定于 TI 协议，这使用户无需通过设置 SPI_CR1 和 SPI_CR2 寄存器（例如 SSM、SSI、SSOE）来对 NSS 管理进行设置。

在从模式下（*图 274：TI 模式——从模式，单次传输*和*图 275：TI 模式——从模式，连续传输*），使用 SPI 波特率预分频器来控制 MISO 引脚状态切换到高阻态的时刻。可以使用任意波特率，因此可以非常灵活地确定此时刻。但是，波特率通常设置为外部主时钟波特率。MISO 信号变为高阻态的时间 ($t_{release}$) 取决于芯片内部电路同步以及通过 SPI_CR1 寄存器的 BR[2:0] 设置的波特率值。具体公式如下：

$$\frac{t_{baud\_rate}}{2} + 4 \times t_{pclk} < t_{release} < \frac{t_{baud\_rate}}{2} + 6 \times t_{pclk}$$

注意： *此特性不适用于 Motorola SPI 通信（FRF 位为 0）。*

要在从器件发送器模式下使用错误中断 (ERRIE = 1) 检测 TI 帧错误，必须通过将 SPI_CR1 寄存器中的 BIDIMODE 和 BIDIOE 置 1 来将 SPI 配置为双线单向模式。当 BIDIMODE 置为 0 时，OVR 将置 1，因为始终不会读取数据寄存器从而始终生成溢出错误中断；而当 BIDIMODE 置 1 时，不会接收数据，也不会将 OVR 置 1。

图 274. TI 模式——从模式，单次传输

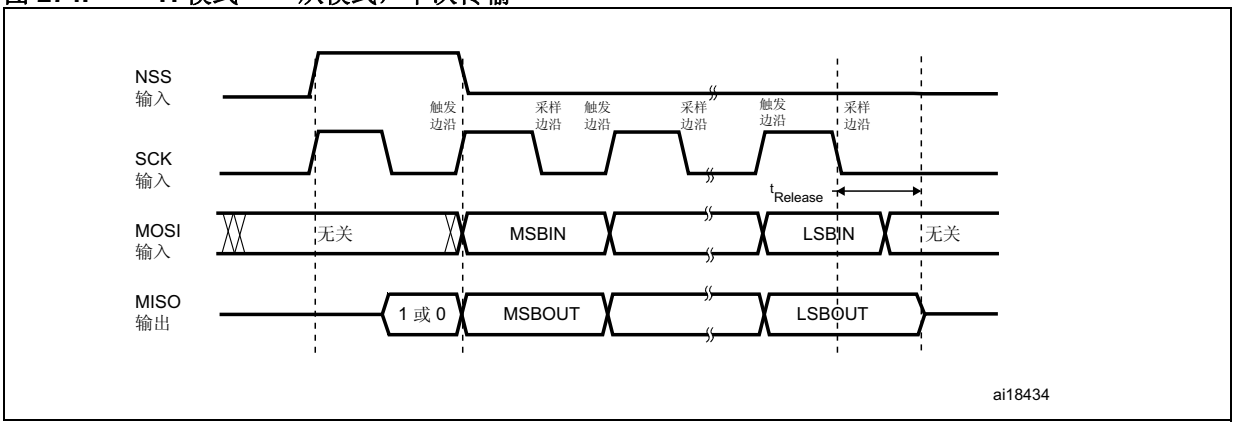
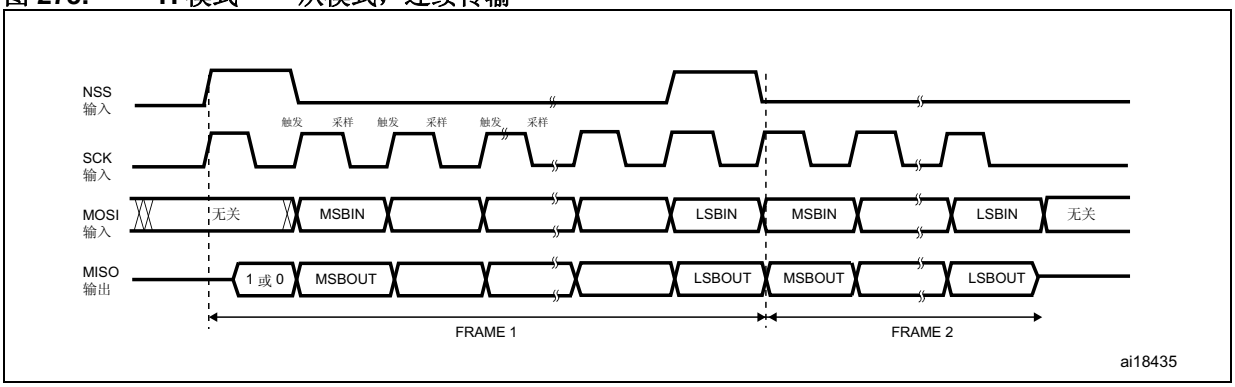


图 275. TI 模式——从模式，连续传输



27.3.3 把 SPI 配置成主器件

在主模式配置下，在 SCK 引脚上输出串行时钟。

步骤

1. 设置 BR[2:0] 位以定义串行时钟波特率（参见 SPI_CR1 寄存器）。
2. 选择 CPOL 和 CPHA 位，以定义数据传输和串行时钟之间的关系（四种关系中的一种）（参见*图 273*）。如果选择 TI 模式，则不需要此步骤。
3. 设置 DFF 位，以定义 8 或 16 位数据帧格式
4. 配置 SPI_CR1 寄存器中的 LSBFIRST 位以定义帧格式。如果选择 TI 模式，则不需要此步骤。
5. 如果 NSS 引脚配置成输入，在 NSS 硬件模式下，NSS 引脚在整个字节发送序列期间都连接到高电平信号；在 NSS 软件模式下，将 SPI_CR1 寄存器中的 SSM 和 SSI 位置 1。如果 NSS 引脚配置成输出，只应将 SSOE 位置 1。如果选择 TI 模式，则不需要此步骤。
6. 将 SPI_CR2 中的 FRF 位置 1，以选择 TI 协议进行串行通信。
7. MSTR 和 SPE 位必须置 1（仅当 NSS 引脚与高电平信号连接时，这两个位才保持置 1）。

在此配置中，MOSI 引脚为数据输出，MISO 引脚为数据输入。

发送序列

在发送缓冲区中写入字节时，发送序列开始。

在第一个位传输期间，数据字节（从内部总线）并行加载到移位寄存器中，然后以串行方式移出到 MOSI 引脚，至于是 MSB 在前还是 LSB 在前则取决于 SPI_CR1 寄存器中的 LSBFIRST 位。TXE 标志在数据从发送缓冲区传输到移位寄存器时置 1，并且在 SPI_CR2 寄存器中的 TXEIE 位置 1 时将生成中断。

接收序列

对于接收器，在数据传输完成时：

- 移位寄存器中的数据将传输到接收缓冲区，并且 RXNE 标志置 1
- 如果 SPI_CR2 寄存器中的 RXNEIE 位置 1，则生成中断

在出现最后一个采样时钟边沿时，RXNE 位置 1，移位寄存器中接收的数据字节被拷贝到接收缓冲区中。当读取 SPI_DR 寄存器时，SPI 外设将返回此缓冲值。

通过读取 SPI_DR 寄存器将 RXNE 位清零。

如果在发送开始后将要发送的下一个数据置于发送缓冲区，则可保持连续的发送流。请注意，仅当 TXE 标志为 1 时，才可以对发送缓冲区执行写操作。

注意： *如果与之通信的从器件需要在每个字节传输之间 拉低片选信号，必须将该主器件的 NSS 配置成 GPIO，或使用另外 GPIO，通过软件控制从器件的片选。*

主模式下的 SPI TI 协议

在主模式下，SPI 接口与 TI 协议兼容。可以使用 SPI_CR2 寄存器的 FRF 位来配置主 SPI 串行通信，以兼容此协议。

时钟极性和相位都被强制为遵循 TI 协议，和 SPI_CR1 中的设置无关。NSS 管理也特定于 TI 协议，这使用户无需通过设置 SPI_CR1 和 SPI_CR2 寄存器（例如 SSM、SSI、SSOE）来对 NSS 管理进行设置。

*图 276：TI 模式——主模式，单次传输*和*图 277：TI 模式——主模式，连续传输*）显示了在主模式下选择 TI 模式时的 SPI 主模式通信波形。

图 276. TI 模式——主模式，单次传输

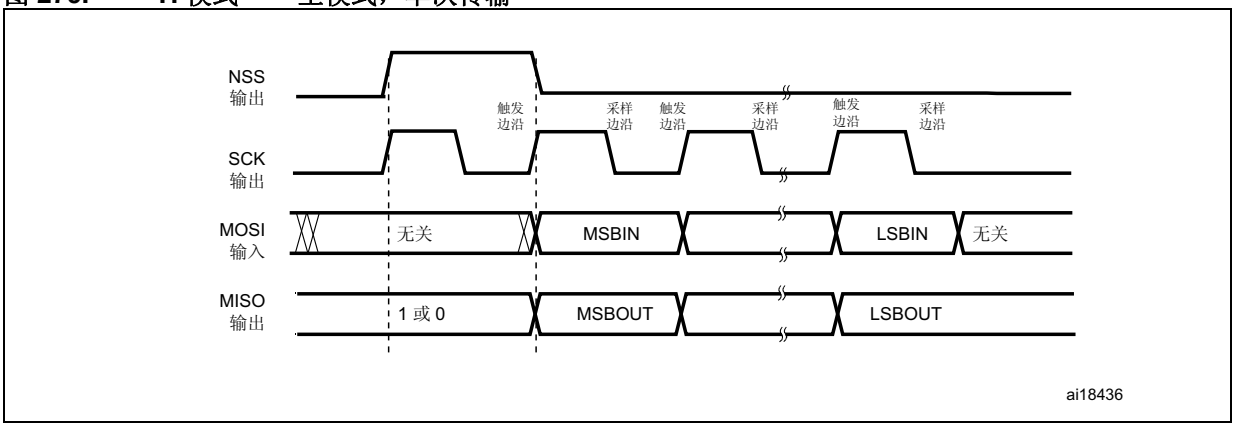
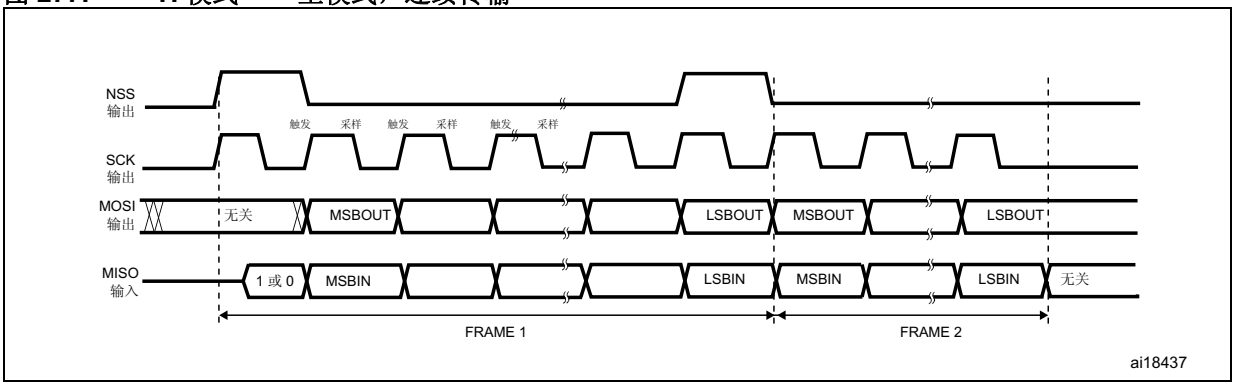


图 277. TI 模式——主模式，连续传输



27.3.4 配置 SPI 进行半双工通信

SPI 能够在以下两种配置中以半双工模式工作。

- 1 个时钟和 1 条双向数据线
- 1 个时钟和 1 条数据线（只接收或只发送）

1 个时钟和 1 条双向数据线 (BIDIMODE=1)

可将 SPI_CR1 寄存器中的 BIDIMODE 位置 1 来使能此模式。在此模式下，SCK 用于时钟，MOSI（主模式下）或 MISO（从模式下）用于数据通信。通过 SPI_CR1 寄存器中的 BIDIOE 位来选择传输方向（输入/输出）。当该位置 1 时，数据线为输出，否则为输入。

1 个时钟和 1 条单向数据线 (BIDIMODE=0)

在此模式下，应用程序可使用 SPI 的只发送或只接收功能。

- 只发送模式类似于全双工模式（BIDIMODE=0、RXONLY=0）：在发送引脚（主模式下的 MOSI 或从模式下的 MISO）上发送数据，接收引脚（主模式下的 MISO 或从模式下的 MOSI）可用作通用 IO。在这种情况下，应用程序只需要忽略接收缓冲区（即使读取数据寄存器，它也不包含接收值）。
- 只接收模式下，应用程序可将 SPI_CR2 寄存器中的 RXONLY 位置 1 来关闭 SPI 输出功能。在这种情况下，发送 IO 引脚（主模式下的 MOSI 或从模式下的 MISO）可用于其它用途。

要在只接收模式下开始通信，配置并使能 SPI：

- 在主模式下，通信会立即开始，并在 SPE 位清零且当前接收结束时停止。在此模式下无需读取 BSY 标志。在进行 SPI 通信时，该标志始终置 1。
- 在从模式下，只要 NSS 被拉低（或在 NSS 软件模式下将 SSI 位清零）并且一直有来自主器件的 SCK 输入，SPI 就会继续接收。

27.3.5 数据发送和接收过程

接收和发送缓冲区

在接收过程中，数据收到后，先存储到内部接收缓冲区中；而在发送过程中，先将数据存储到内部发送缓冲区中，然后发送数据。

对 SPI_DR 寄存器的读访问将返回接收缓冲值，而对 SPI_DR 寄存器的写访问会将写入的数据存储到发送缓冲区中。

在主模式下启动通信序列

- 在全双工模式下（BIDIMODE=0 且 RXONLY=0）
  - 将数据写入到 SPI_DR 寄存器（发送缓冲区）时，通信序列启动。
  - 随后在第一个位的发送期间，将数据从发送缓冲区并行加载到 8 位移位寄存器中，然后以串行方式将其移出到 MOSI 引脚。
  - 同时，将 MISO 引脚上接收的数据以串行方式移入 8 位移位寄存器，然后并行加载到 SPI_DR 寄存器（接收缓冲区）中。
- 在单向只接收模式下（BIDIMODE=0 且 RXONLY=1）
  - 只要 SPE = 1，通信序列就立即开始。
  - 只有接收器激活，并且在 MISO 引脚上接收的数据以串行方式移入 8 位移位寄存器，然后并行加载到 SPI_DR 寄存器（接收缓冲区）中。
- 在双向模式下，进行发送时（BIDIMODE=1 且 BIDIOE=1）
  - 将数据写入到 SPI_DR 寄存器（发送缓冲区）时，通信序列启动。
  - 随后在第一个位的发送期间，将数据从发送缓冲区并行加载到 8 位移位寄存器中，然后以串行方式将其移出到 MOSI 引脚。
  - 不接收任何数据。
- 在双向模式下，进行接收时（BIDIMODE=1 且 BIDIOE=0）
  - 只要 SPE=1 且 BIDIOE=0，通信序列就立即开始。
  - 在 MOSI 引脚上接收的数据以串行方式移入 8 位移位寄存器，然后并行加载到 SPI_DR 寄存器 （接收缓冲区）中。
  - 发送器没有激活，因此不会有数据以串行方式移出 MOSI 引脚。

在从模式下启动通信序列

- 在全双工模式下（BIDIMODE=0 且 RXONLY=0）
  - 当从器件在其 MOSI 引脚上收到时钟信号和数据的第一个位时，通信序列开始。其余 7 个位将加载到移位寄存器中。
  - 同时，在第一个位的发送期间，将数据从发送缓冲区并行加载到 8 位移位寄存器中，然后以串行方式将其移出到 MISO 引脚。在 SPI 主器件启动传输前，软件必须已把要从器件发送的数据写入发送缓冲区。
- 在单向只接收模式下（BIDIMODE=0 且 RXONLY=1）
  - 当从器件在其 MOSI 引脚上收到时钟信号和数据的第一个位时，通信序列开始。其余 7 个位将加载到移位寄存器中。
  - 发送器没有激活，因此不会有数据以串行方式移出 MISO 引脚。
- 在双向模式下，进行发送时（BIDIMODE=1 且 BIDIOE=1）
  - 当从器件收到时钟信号，并将发送缓冲区中的第一个位在 MISO 引脚上发送时，通信序列开始。
  - 随后在第一个位的发送期间，将数据从发送缓冲区并行加载到 8 位移位寄存器中，然后以串行方式将其移出到 MISO 引脚。在 SPI 主器件启动传输前，软件必须已把要从器件发送的数据写入发送缓冲区。
  - 不接收任何数据。
- 在双向模式下，进行接收时（BIDIMODE=1 且 BIDIOE=0）
  - 当从器件在其 MISO 引脚上收到时钟信号和数据的第一个位时，通信序列开始。
  - 在 MISO 引脚上接收的数据以串行方式移入 8 位移位寄存器，然后并行加载到 SPI_DR 寄存器 （接收缓冲区）中。
  - 发送器没有激活，因此不会有数据以串行方式移出 MISO 引脚。

处理数据发送与接收

将数据从发送缓冲区传输到移位寄存器时，TXE 标志（发送缓冲区为空）置 1。该标志表示内部发送缓冲区已准备好加载接下来的数据。如果 SPI_CR2 寄存器中的 TXEIE 位置 1，可产生中断。通过对 SPI_DR 寄存器执行写操作将 TXE 位清零。

注意： *软件必须确保在尝试写入发送缓冲区之前 TXE 标志已置 1。否则，将覆盖之前写入发送缓冲区的数据。*

将数据从移位寄存器传输到接收缓冲区时，RXNE 标志（接收缓冲区非空）会在最后一个采样时钟边沿置 1。它表示已准备好从 SPI_DR 寄存器中读取数据。如果 SPI_CR2 寄存器中的 RXNEIE 位置 1，可产生中断。通过读取 SPI_DR 寄存器将 RXNE 位清零。

对于某些配置，可以在最后一次数据传输期间使用 BSY 标志来等待传输完成。

主模式或从模式下的全双工发送和接收过程（BIDIMODE=0 且 RXONLY=0）

软件必须遵循以下步骤来发送和接收数据（参见*图 278* 和*图 279*）：

1. 通过将 SPE 位置 1 来使能 SPI。
2. 将第一个要发送的数据项写入 SPI_DR 寄存器（此操作会将 TXE 标志清零）。
3. 等待至 TXE=1，然后写入要发送的第二个数据项。然后等待至 RXNE=1，读取 SPI_DR 以获取收到的第一个数据项（此操作会将 RXNE 位清零）。对每个要发送/接收的数据项重复此操作，直到第 n-1 个接收的数据为止。
4. 等待至 RXNE=1，然后读取最后接收的数据。
5. 等待至 TXE=1，然后等待至 BSY=0，再关闭 SPI。

此外，还可以使用在 RXNE 或 TXE 标志所产生的中断对应的各个中断子程序来实现该过程。

图 278. 主/全双工模式（BIDIMODE=0 且 RXONLY=0）下的 TXE/RXNE/BSY 行为（在连续传输的情况下）

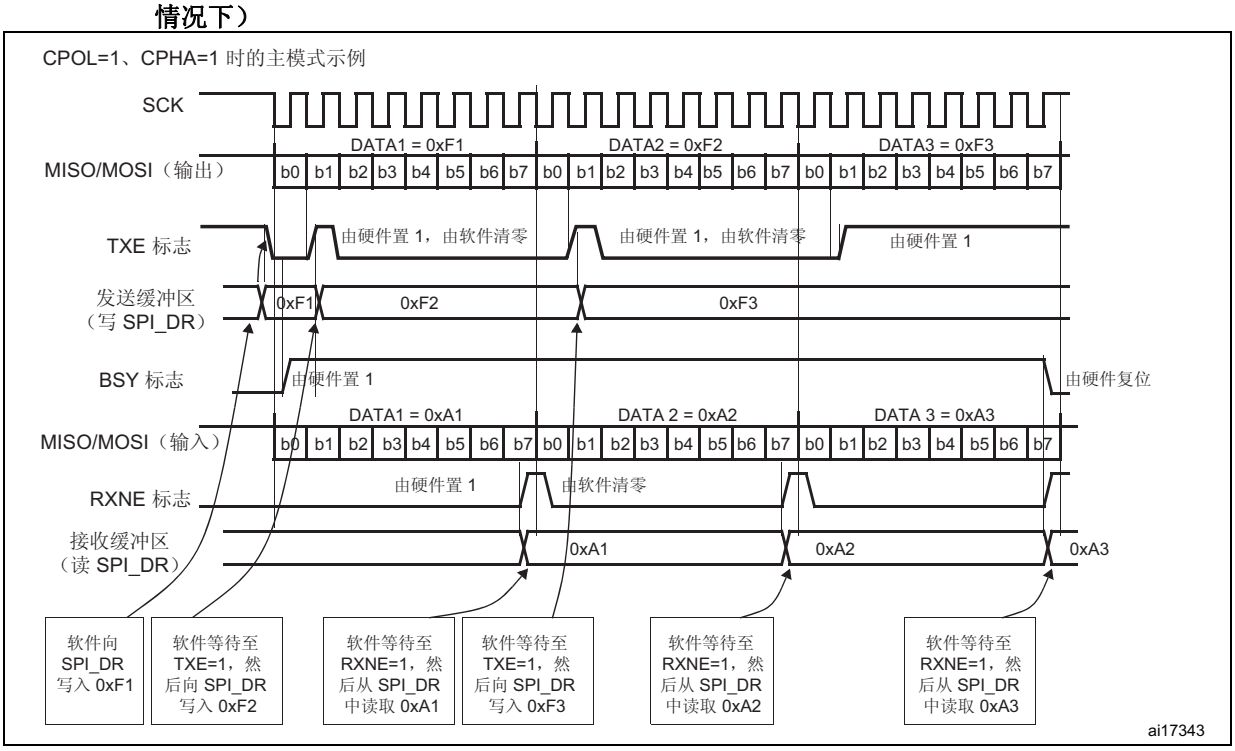
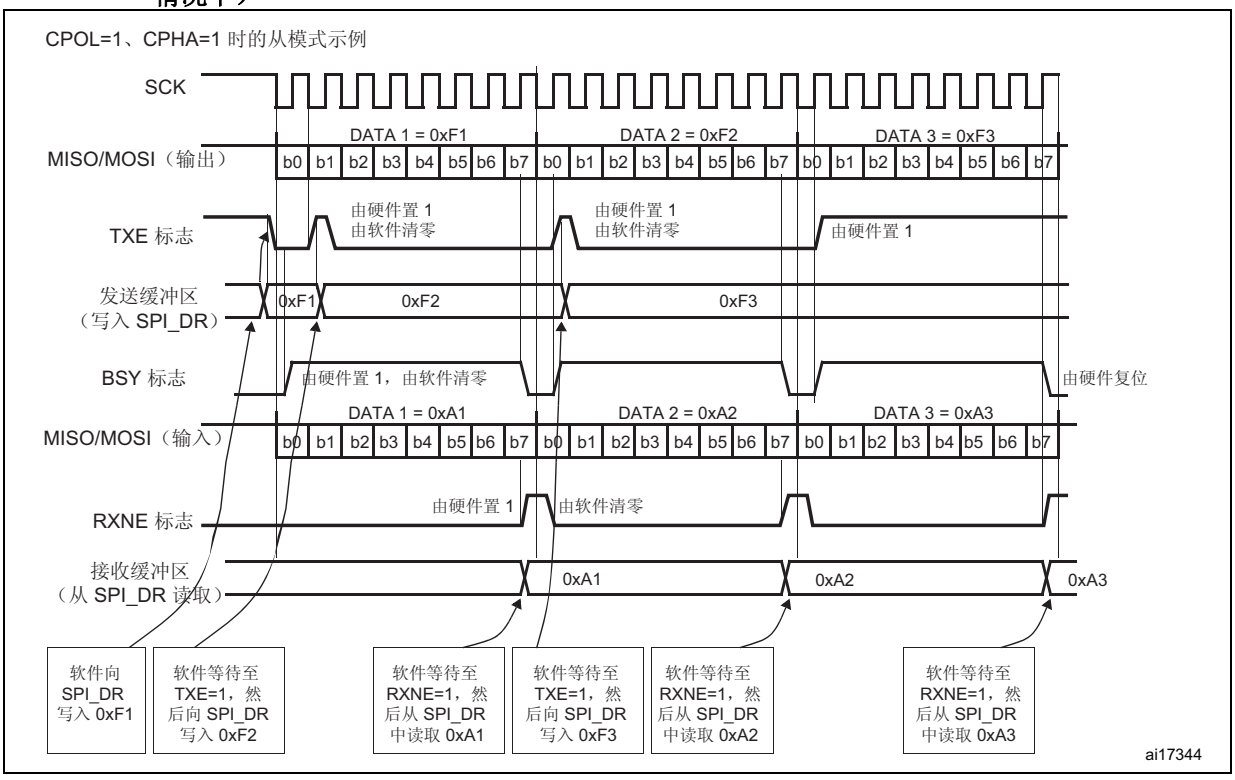


图 279. 主/全双工模式（BIDIMODE=0 且 RXONLY=0）下的 TXE/RXNE/BSY 行为（在连续传输的情况下）



只发送模式下的数据发送过程（BIDIMODE=0、RXONLY=0）

在此模式下，可以按下文所述简化过程，并且可使用 BSY 位等待发送完成（参见*图 280* 和*图 281*）。

1. 通过将 SPE 位置 1 来使能 SPI。
2. 将第一个要发送的数据项写入 SPI_DR 寄存器（此操作会将 TXE 标志清零）。
3. 等待至 TXE=1，然后写入下一个要发送的数据项。对每个要发送的数据项重复此步骤。
4. 将最后一个数据项写入 SPI_DR 寄存器后，等待至 TXE=1，然后等待至 BSY=0，这表示最后的数据发送完成。

此外，还可以使用在 TXE 标志所产生的中断对应的中断子程序来实现该过程。

注意： *在不连续通信期间，在对 SPI_DR 执行写操作与 BSY 位置 1 之间有 2 个 APB 时钟周期的延迟。因此，在只发送模式下，写入最后的数据后，必须先等待 TXE 位置 1，然后等待 BSY 位清零。*

在只发送模式下，发送两个数据项后，SPI_SR 寄存器中的 OVR 标志将置 1，因为始终不会读取接收的数据。

图 280. 主设备只发送模式（BIDIMODE=0 且 RXONLY=0）下的 TXE/BSY 行为（在连续传输的情况下）

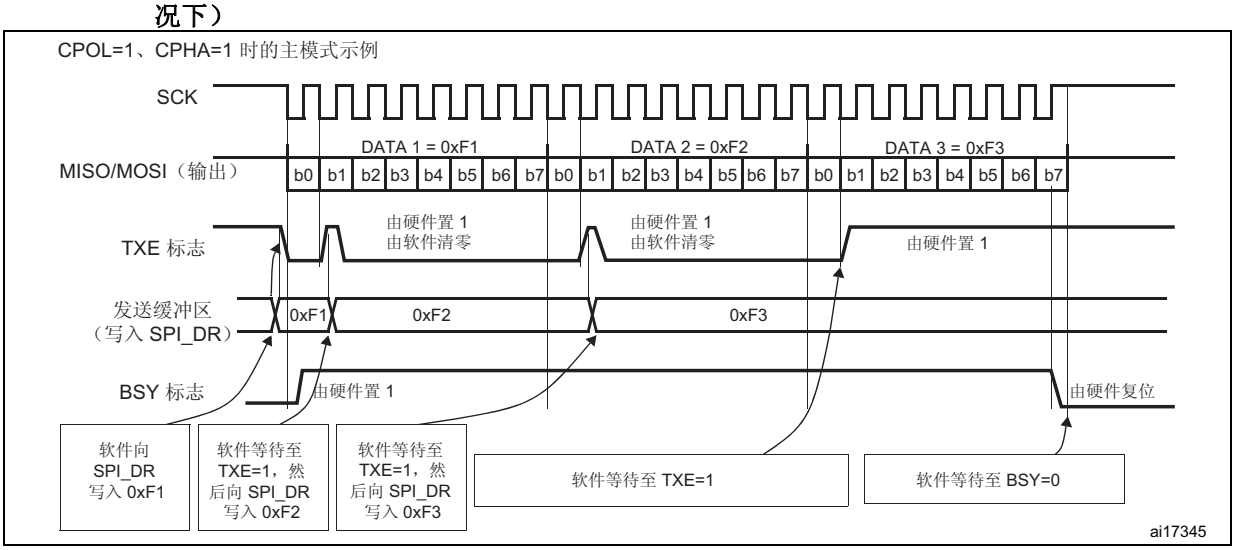
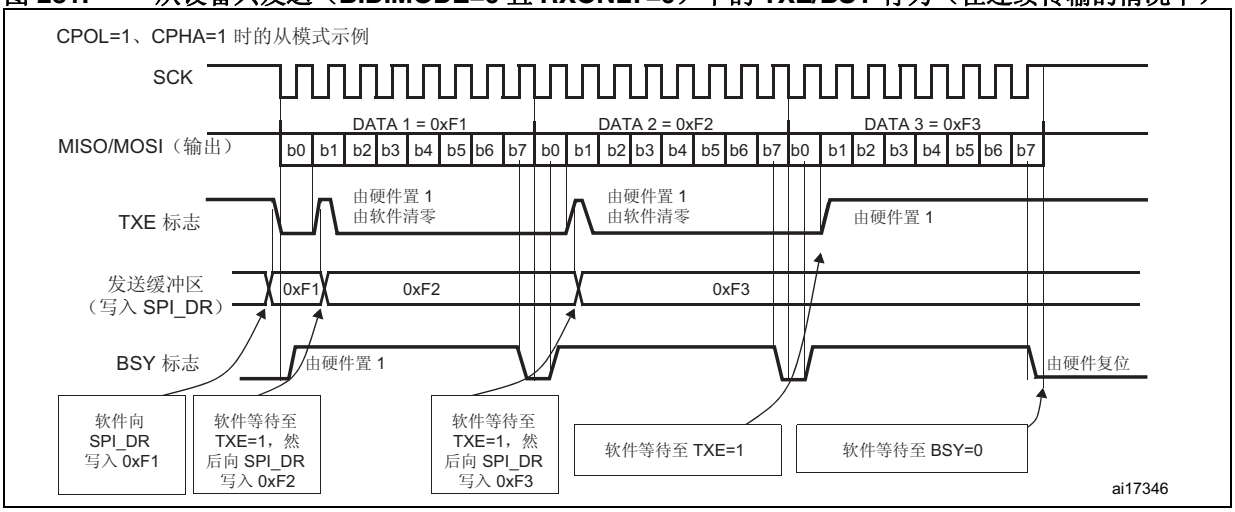


图 281. 从设备只发送（BIDIMODE=0 且 RXONLY=0）下的 TXE/BSY 行为（在连续传输的情况下）



单线双向模式下的发送过程（BIDIMODE=1 且 BIDIOE=1）

在此模式下，过程与只发送模式下的过程相似，除了在使能 SPI 前必须将 SPI_CR2 寄存器中的 BIDIMODE 位和 BIDIOE 位均置 1。

单向只接收过程（BIDIMODE=0 且 RXONLY=1）

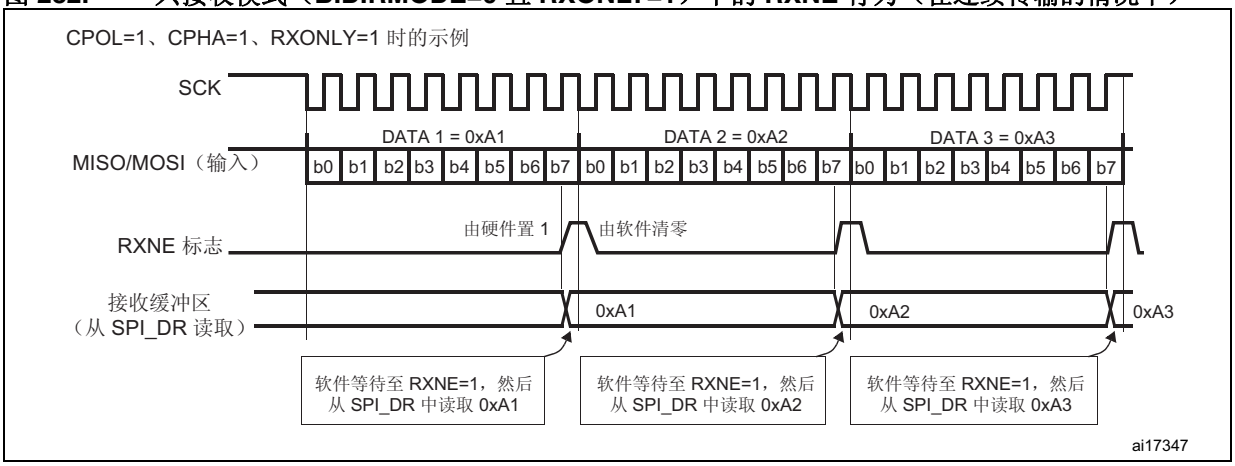
在此模式下，可以按如下所述简化过程（参见*图 282*）：

1. 将 SPI_CR2 寄存器中的 RXONLY 位置 1。
2. 通过将 SPE 位置 1 使能 SPI：
   a) 在主模式下，这会立即激活 SCK 时钟的产生，并以串行方式接收数据，直到关闭 SPI (SPE=0)。
   b) 在从模式下，当 SPI 主器件将该从器件的 NSS 驱动为低电平并输出 SCK 时钟时，接收数据。
3. 等待 RXNE=1，然后读取 SPI_DR 寄存器以获取接收的数据（此操作会将 RXNE 位清零）。对每个要接收的数据项重复此操作。

此外，还可以使用在 RXNE 标志所产生的中断对应的中断子程序来实现该过程。

注意： *如果需要在最后一次传输后关闭 SPI，请采纳第 740 页的第 27.3.8 节：关闭 SPI 中所述的建议。*

图 282. 只接收模式（BIDIRMODE=0 且 RXONLY=1）下的 RXNE 行为（在连续传输的情况下）



单线双向模式下的接收过程（BIDIMODE=1 和 BIDIOE=0）

在此模式下，过程与只接收模式的过程相似，除了在使能 SPI 前必须将 SPI_CR2 寄存器中的 BIDIMODE 位置 1 并将 BIDIOE 清零。

连续传输和间断传输

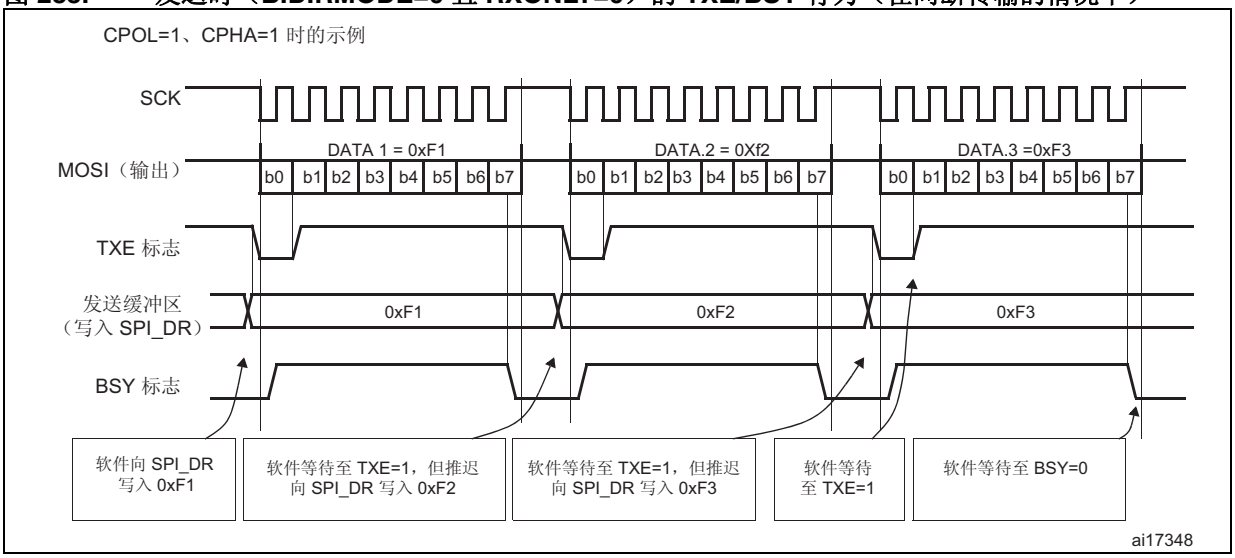
在主模式下发送数据时，如果软件速度快到能在下一个数据传输完成之前响应该数据从数据寄存器传输到移位寄存器所触发的 TXE 中断，并能立即完成再下一个数据的写 SPI_DR 操作，则这种通信称为连续通信。在这种情况下，各数据项之间 SPI 时钟的生成不会间断，并且各数据传输之间不会清零 BSY 位。

相反，如果软件速度不够快，则可能导致通信中断。在这种情况下，各数据传输之间会清零 BSY 位（参见*图 283*）。

在主设备仅接收模式 (RXONLY=1) 下，通信始终是连续的，且 BSY 标志始终读为 1。

在从模式下，通信的连续性由 SPI 主器件决定。任何情况下（即使通信是连续的），在各个数据传输之间 BSY 标志都会短暂变为低电平，持续时间为一个 SPI 时钟周期（参见*图 281*）。

图 283. 发送时（BIDIRMODE=0 且 RXONLY=0）的 TXE/BSY 行为（在间断传输的情况下）



27.3.6 CRC 计算

为确保通信的可靠性，SPI 模块实现了硬件 CRC 功能。针对发送的数据和接收的数据分别实现 CRC 计算。使用可编程的多项式对每个位来计算 CRC。在由 SPI_CR1 寄存器中的 CPHA 位和 CPOL 位定义的采样时钟边沿采样每个位来进行计算。

注意： *该 SPI 模块提供两种 CRC 计算标准，具体取决于为发送和/或接收选择的数据帧格式：8 位数据采用 CR8，16 位数据采用 CRC16。*

通过将 SPI_CR1 寄存器中的 CRCEN 位置 1 来使能 CRC 的计算。此操作将复位 CRC 寄存器（SPI_RXCRCR 和 SPI_TXCRCR）。在全双工或只发送模式下，如果传输由软件（CPU 模式）管理，则在将最后传输的数据写入 SPI_DR 后，必须立即对 CRCNEXT 位执行写操作。最后一次数据传输结束时，将发送 SPI_TXCRCR 值。

在只接收模式下，如果传输由软件（CPU 模式）管理，则在接收到倒数第二个数据后，必须对 CRCNEXT 位执行写操作。在收到最后一个数据后会收到 CRC，然后执行 CRC 校验。

如果传输过程中出现数据损坏，则在数据和 CRC 传输结束时，SPI_SR 寄存器中的 CRCERR 标志将置 1。

如果发送缓冲区中存在数据，则只有在发送数据字节后才会发送 CRC 值。在 CRC 发送期间，CRC 计算器处于关闭状态且寄存器值保持不变。

可通过以下步骤使用 CRC 进行 SPI 通信：

1. 对 CPOL、CPHA、LSBFirst、BR、SSM、SSI 和 MSTR 值进行编程。
2. 对 SPI_CRCPR 寄存器中的多项式进行编程。
3. 通过将 SPI_CR1 寄存器中的 CRCEN 位置 1 来使能 CRC 计算。此操作还会将 SPI_RXCRCR 和 SPI_TXCRCR 寄存器清零。
4. 通过将 SPI_CR1 寄存器中的 SPE 位置 1 使能 SPI。
5. 启动并维持通信，直到只剩下一个字节或半字未发送或接收。
   - 在全双工或只发送模式下，如果传输由软件管理，则在向发送缓冲区写入最后一个字节或半字后，将 SPI_CR1 寄存器中的 CRCNEXT 位置 1，以表示在发送完最后一个字节后将发送 CRC。
   - 在只接收模式下，在接收倒数第二个数据后，立即将 CRCNEXT 位置 1，以便使 SPI 准备好在接收完最后一个数据后进入 CRC 阶段。在 CRC 传输期间，CRC 计算将冻结。
6. 传输完最后一个字节或半字后，SPI 进入 CRC 传输和校验阶段。在全双工模式或只接收模式下，将接收的 CRC 与 SPI_RXCRCR 值进行比较。如果两个值不匹配，则 SPI_SR 中的 CRCERR 标志将置 1，并且在 SPI_CR2 寄存器中的 ERRIE 位置 1 时会产生中断。

注意： *当 SPI 处于从模式时，注意只能在时钟稳定（即，时钟处于空闲电平）时使能 CRC 计算。否则，可能导致 CRC 计算错误。因为，只要 CRCEN 位置 1，无论 SPE 位的值如何，只要有时钟输入，CRC 计算器就开始工作。*

在 SPI 通信时钟频率较高的情况下，发送 CRC 时务必小心。由于在 CRC 传输阶段 CPU 应尽可能空闲，因此禁止在 CRC 发送阶段调用函数，以便避免最后的数据和 CRC 接收出错。实际上，在发送/接收最后的数据之前必须对 CRCNEXT 位执行写操作。

SPI 通信时钟频率较高时，建议使用 DMA 模式来避免由于 CPU 访问影响 SPI 带宽而导致 SPI 速度性能下降。

如果将器件配置为从器件，并且使用 NSS 硬件模式，则需要在数据阶段和 CRC 阶段之间将 NSS 引脚保持为低电平。

当 SPI 配置为从模式并且 CRC 功能已使能时，即使 NSS 引脚上为高电平，也会进行 CRC 计算。例如，在多从模式环境下可能出现这种情况，此时通信主器件会交替寻址从器件。

在对从器件片选的切换期间内，应在主器件和从器件两端同时将 CRC 值清零，以重新同步主从双方的 CRC 计算。

要将 CRC 清零，请按以下步骤操作：

1. 关闭 SPI (SPE = 0)
2. 将 CRCEN 位清零
3. 将 CRCEN 位置 1
4. 使能 SPI (SPE = 1)

27.3.7 状态标志

应用可通过三种状态标志监视 SPI 总线的状态。

发送缓冲区为空 (TXE)

此标志置 1 时，表示发送缓冲区为空，可以将待发送的下一个数据加载到缓冲区中。对 SPI_DR 寄存器执行写操作时，将清零 TXE 标志。

接收缓冲区非空 (RXNE)

此标志置 1 时，表示接收缓冲区中存在有效的已接收数据。读取 SPI_DR 时，将清零该标志。

BUSY

BSY 标志由硬件置 1 和清零（对此标志执行写操作没有任何作用）。BSY 标志用于指示 SPI 通信的状态。

BSY 置 1 时，表示 SPI 正忙于通信。在主模式下的双向通信接收模式（MSTR=1 且 BDM=1 且 BDOE=0）有一个例外情况，BSY 标志在接收过程中保持低电平。

如果软件要关闭 SPI 并进入停止模式（或关闭外设时钟），可使用 BSY 标志检测传输是否结束以避免破坏最后一个数据的传输。为此，必须严格遵循下述步骤。

BSY 标志还可用于避免在多主模式系统中发生写冲突。

传输开始时，BSY 标志将置 1，但在主模式下的双向通信接收模式（MSTR=1 且 BDM=1 且 BDOE=0）下例外。

在以下情况硬件将清零该标志：

- 传输完成时（主模式下的连续通信除外）
- 关闭 SPI 时
- 发生主模式故障时 (MODF=1)

当通信不连续时，BSY 标志在各通信之间处于低电平。

当通信连续时：

- 在主模式下，BSY 标志在所有传输期间均保持高电平
- 在从模式下，BSY 标志在各传输之间的一个 SPI 时钟周期内变为低电平

注意： *请勿使用 BSY 标志处理每次数据发送或接收，最好改用 TXE 标志和 RXNE 标志。*

27.3.8 关闭 SPI

传输终止时，应用可通过关闭 SPI 外设来停止通信。这通过将 SPE 位清零来完成。

对于某些配置，在传输进行时关闭 SPI 并进入停止模式会导致当前传输受损，并且/或者 BSY 标志可能不可靠。

为避免上述后果，建议在关闭 SPI 时按以下步骤操作：

在主模式或全双工从模式（BIDIMODE=0、RXONLY=0）下

1. 等待 RXNE=1 以接收最后的数据
2. 等待 TXE=1
3. 然后等待 BSY=0
4. 关闭 SPI (SPE=0)，最后进入停止模式（或关闭外设时钟）

在主模式或单向只发送从模式（BIDIMODE=0、RXONLY=0）或双向通信发送模式（BIDIMODE=1、BIDIOE=1）下

在最后的数据写入 SPI_DR 寄存器后：

1. 等待 TXE=1
2. 然后等待 BSY=0
3. 关闭 SPI (SPE=0)，最后进入停止模式（或关闭外设时钟）

在单向只接收主模式（MSTR=1、BIDIMODE=0、RXONLY=1）或双向通信接收模式（MSTR=1、BIDIMODE=1、BIDIOE=0）下

必须以特殊方式管理这种情况，以避免多余的SPI数据传输。以下序列仅适用于 SPI Motorola 配置（FRF 位置 0）：

1. 等待倒数第二个数据（第 n-1 个）对应的 RxNE 标志置位
2. 然后等待一个 SPI 时钟周期（使用软件循环），才能关闭 SPI (SPE=0)
3. 再等待最后的 RXNE=1，然后进入停止模式（或关闭外设时钟）

当 SPI 配置为 TI 模式（FRF 位置 1）时，必须按以下步骤操作以避免当关闭 SPI 时在 NSS 上产生不需要的脉冲：

1. 等待倒数第二个数据（第 n-1 个）对应的 RxNE 标志置位
2. 在以下窗口帧中使用软件循环关闭 SPI (SPE = 0)：
   - 至少一个 SPI 时钟周期后，
   - LSB 数据开始传输前。

注意： *在双向通信接收主模式（MSTR=1、BDM=1、BDOE=0）下，BSY 标志在传输期间保持低电平。*

在只接收从模式（MSTR=0、BIDIMODE=0、RXONLY=1）或双向通信接收模式（MSTR=0、BIDIMODE=1、BIDOE=0）下

1. 可以随时关闭 SPI（写入 SPE=0）：当前传输完成后，SPI 才被真正关闭
2. 之后，如果要进入停止模式，则必须首先等待至 BSY = 0，然后才能进入停止模式（或关闭外设时钟）。

27.3.9 使用 DMA（直接存储器寻址）进行 SPI 通信

要以最大速度工作，需要给 SPI 不断提供要发送的数据，并及时读取接收缓冲区中的数据，以避免上溢。为加速传输，SPI 提供了 DMA 功能，以实现简单的请求/应答协议。

当使能 SPI_CR2 寄存器中的使能位时，将请求 DMA 访问。发送缓冲区和接收缓冲区会发出各自的 DMA 请求（参见*图 284* 和*图 285*）：

- 在发送过程中，每次 TXE 位置 1 都会发出 DMA 请求。DMA 随后对 SPI_DR 寄存器执行写操作（此操作会将 TXE 标志清零）。
- 在接收过程中，每次 RXNE 位置 1 都会发出 DMA 请求。DMA 随后对 SPI_DR 寄存器执行读操作（此操作会将 RXNE 标志清零）。

当 SPI 仅用于发送数据时，可以只使能 SPI Tx DMA 通道。在这种情况下，OVE 标志会置 1，因为未读取接收的数据。

当 SPI 仅用于接收数据时，可以只使能 SPI Rx DMA 通道。

在发送模式下，DMA 完成了所有要发送数据的传输（DMA_ISR 寄存器中的 TCIF 标志置 1）后，可以对 BSY 标志进行监视，以确保 SPI 通信已完成。在关闭 SPI 或进入停止模式前必须执行此步骤，以避免损坏最后一次数据的发送。软件必须首先等待 TXE=1，再等待 BSY=0。

注意： *在不连续通信期间，在对 SPI_DR 执行写操作与 BSY 位置 1 之间有 2 个 APB 时钟周期的延迟。因此，必须在写入最后的数据后，先等待 TXE=1，再等待 BSY=0。*

图 284. 使用 DMA 进行发送

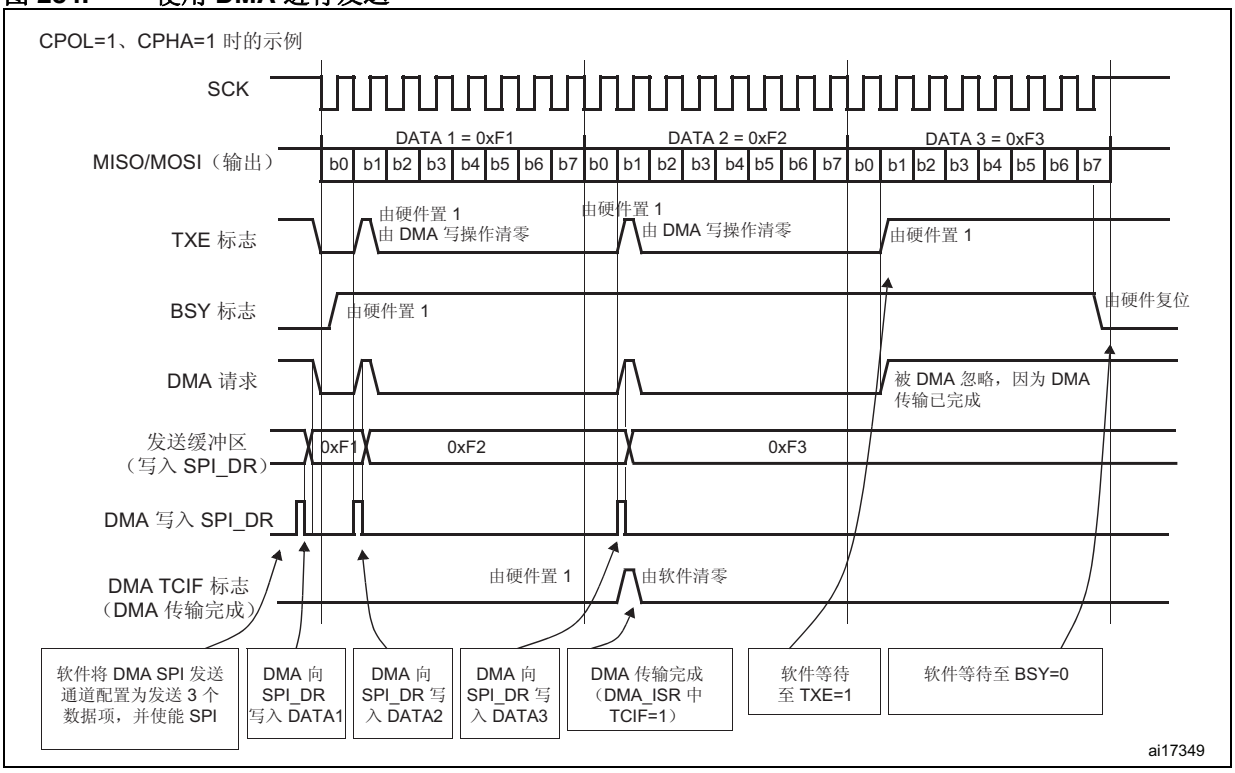
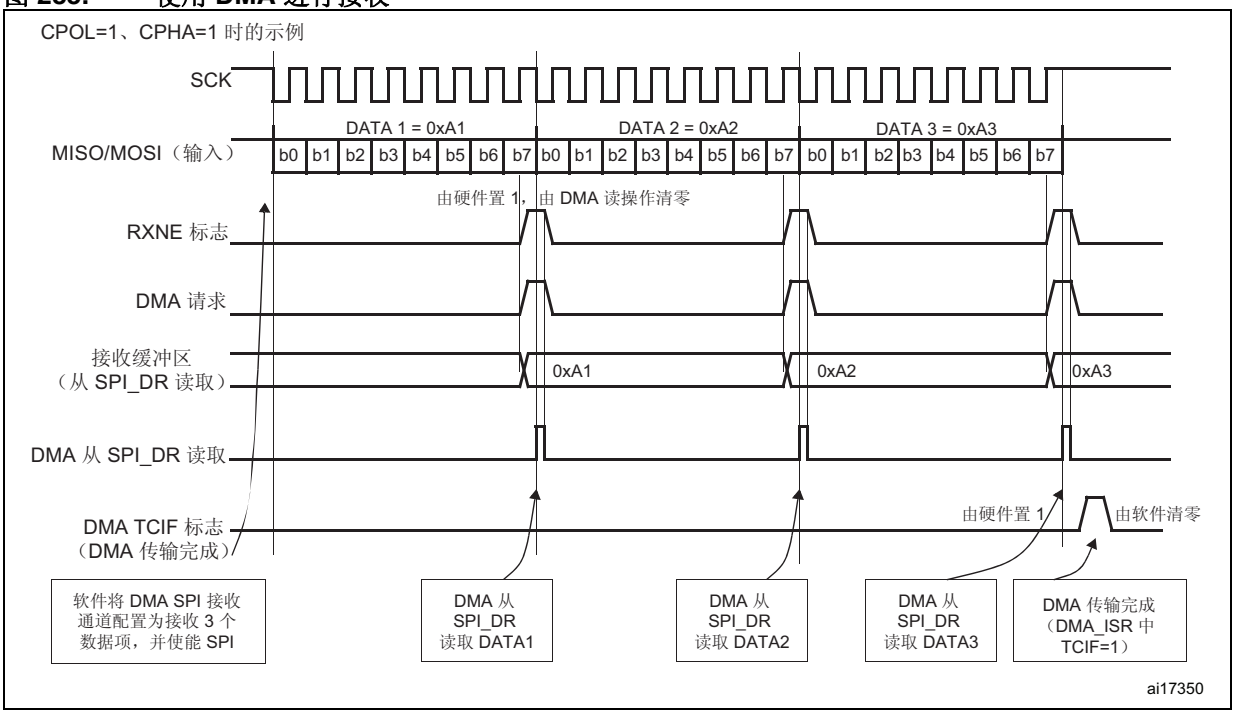


图 285. 使用 DMA 进行接收



DMA 功能与 CRC

当使能的 SPI 通信支持 CRC 通信和 DMA 模式时，在通信结束时会自动发送和接收 CRC，无需使用 CRCNEXT 位。接收 CRC 后，必须在 SPI_DR 寄存器中读取 CRC，以将 RXNE 标志清零。

如果传输过程中出现损坏，则在数据和 CRC 传输结束时，SPI_SR 寄存器中的 CRCERR 标志将置 1。

27.3.10 错误标志

主模式故障 (MODF)

当主器件的 NSS 引脚拉低（NSS 硬件模式下）或 SSI 位为 0（NSS 软件模式下）时，会发生主模式故障，这会自动将 MODF 位置 1。主模式故障会在以下几方面影响 SPI 外设：

- 如果 ERRIE 位置 1，MODF 位将置 1，并生成 SPI 中断。
- SPE 位清零。这将关闭器件的所有输出，并关闭 SPI 接口。
- MSTR 位清零，从而强制器件进入从模式。

使用以下软件序列将 MODF 位清零：

1. 在 MODF 位置 1 时，对 SPI_SR 寄存器执行读或写访问。
2. 然后，对 SPI_CR1 寄存器执行写操作。

为避免包含多个 MCU 的系统中发生多从模式冲突，必须在 MODF 位清零序列期间将 NSS 引脚拉高。在该清零序列后，可以将 SPE 和 MSTR 位恢复到原始状态。

作为安全措施，硬件不允许在 MODF 位置 1 时将 SPE 和 MSTR 位置 1。

在从器件中，不能将 MODF 位置 1。但是，在多主模式配置中，器件可在 MODF 位置 1 时处于从模式。在这种情况下，MODF 位指示系统控制可能存在多主模式冲突。可使用中断程序从此状态完全恢复，方法是执行复位或返回到默认状态。

溢出错误

当主器件发送完数据字节，而从器件尚未将上一个收到的数据所产生的 RXNE 位清零时，将出现溢出情况。出现溢出错误时：

- OVR 位置 1 并在 ERRIE 位置 1 时生成一个中断。

在这种情况下，接收器缓冲区内容不会被来自主器件的新数据更新。读取 SPI_DR 寄存器将返回此字节。主器件后续发送的所有其它字节均将丢失。

依次读取 SPI_DR 寄存器和 SPI_SR 寄存器可将 OVR 清除。

CRC 错误

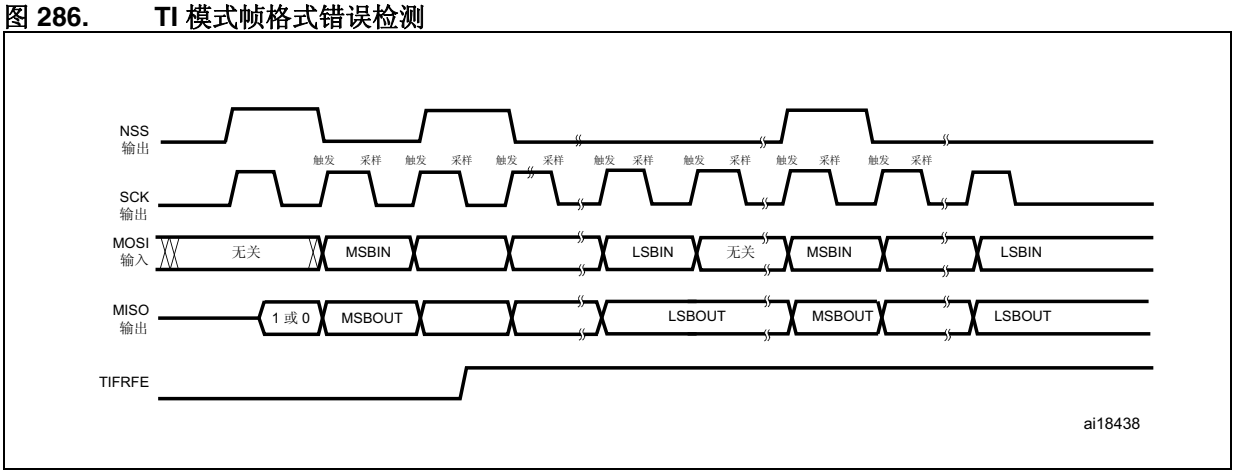
当 SPI_CR1 寄存器中的 CRCEN 位置 1 时，此标志用于验证接收数据的有效性。如果移位寄存器中接收的值与 SPI_RXCRCR 的值不匹配，SPI_SR 寄存器中的 CRCERR 标志将置 1。

TI 模式帧格式错误

如果 SPI 在从模式下工作，并配置为符合 TI 模式协议，则在持续通信期间出现 NSS 脉冲时，将检测到 TI 模式帧格式错误。出现此错误时，SPI_SR 寄存器中的 FRE 标志将置 1。发生错误时不会关闭 SPI，但会忽略 NSS 脉冲，并且 SPI 会等待至下一个 NSS 脉冲，然后再开始新的传输。由于错误检测可能导致丢失两个数据字节，因此数据可能会损坏。

读取 SPI_SR 寄存器时，将清零 FRE 标志。如果 ERRIE 位置 1，则检测到帧格式错误时将产生中断。在这种情况下，由于无法保证数据的连续性，应关闭 SPI，并在重新使能从 SPI 后，由主器件重新发起通信。

图 286. TI 模式帧格式错误检测



27.3.11 SPI 中断

表 124. SPI 中断请求

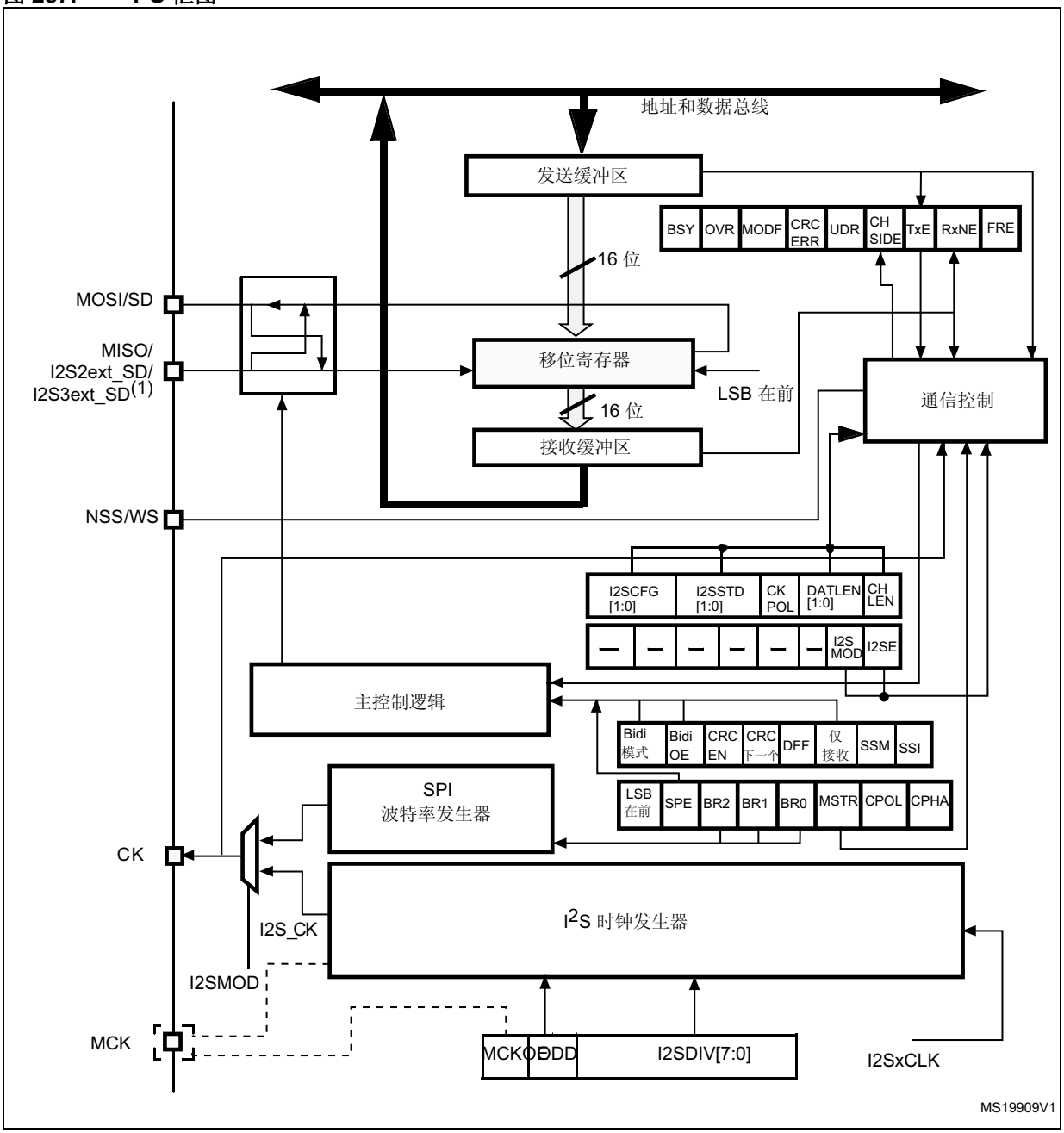
中断事件	事件标志	使能控制位
发送缓冲区为空	TXE	TXEIE
接收缓冲区非空	RXNE	RXNEIE
主模式故障	MODF	ERRIE
溢出错误	OVR	
CRC 错误	CRCERR	
TI 帧格式错误	FRE	ERRIE

27.4 I²S 功能说明

27.4.1 I²S 一般说明

I²S 的框图如*图 287* 所示。

图 287. I²S 框图



1. I2S2ext_SD 和 I2S3ext_SD 为扩展 SD 引脚，用于控制 I²S 全双工模式。

当使能 I²S 功能（将 SPI_I2SCFGR 寄存器中的 I2SMOD 位置 1）时，SPI 可用作音频 I²S 接口。此接口使用几乎与 SPI 相同的引脚、标志和中断。

I^2S 与 SPI 共用以下三个引脚：

- SD：串行数据（映射到 MOSI 引脚），用于发送或接收两个时分复用的数据通道上的数据（仅半双工模式）。
- WS：字选择（映射到 NSS 引脚），是主模式下的数据控制信号输出以及从模式下的数据控制信号输入。
- CK：串行时钟（映射到 SCK 引脚），是主模式下的串行时钟输出以及从模式下的串行时钟输入。
- I2S2ext_SD 和 I2S3ext_SD：用于控制 I^2S 全双工模式的附加引脚 （映射到 MISO 引脚）。

当某些外部音频设备需要使用主时钟输出时，可以使用附加引脚：

- MCK：当 I^2S 配置为主模式（并且 SPI_I2SPR 寄存器中的 MCKOE 位置 1）时，使用主时钟（单独映射）输出此附加时钟，该时钟输出频率 256 × F_S，其中 F_S 为音频信号采样频率。

I^2S 在主模式下使用自身的时钟发生器生成通信时钟。此时钟发生器也是主时钟输出的源。在 I^2S 模式下可以使用两个额外的寄存器。一个是时钟发生器配置寄存器 SPI_I2SPR，另一个是通用 I^2S 配置寄存器 SPI_I2SCFGR（音频标准、从/主模式、数据格式、数据包帧、时钟极性等）。

在 I^2S 模式下不使用 SPI_CR1 寄存器和所有 CRC 寄存器。同样，也不使用 SPI_CR2 寄存器中的 SSOE 位以及 SPI_SR 中的 MODF 和 CRCERR 位。

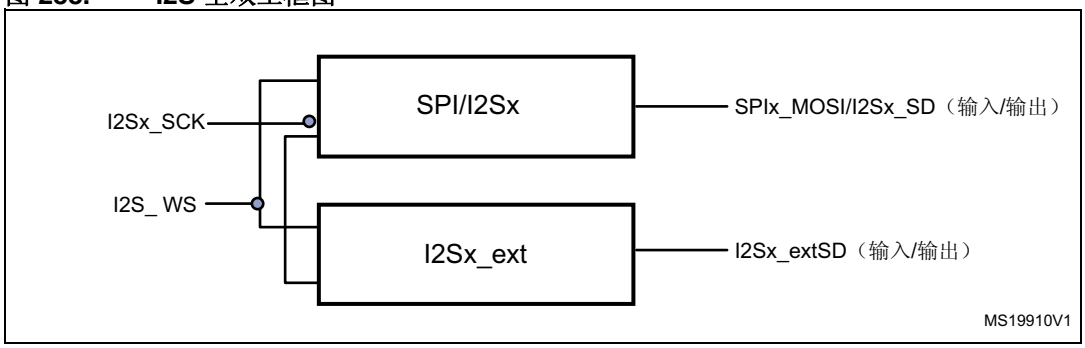
I^2S 使用和 SPI 相同的寄存器 (SPI_DR) 进行 16 位数据传输。

27.4.2 I2S 全双工

为支持 I2S 全双工模式，除了 I2S2 和 I2S3，还可以使用两个额外的 I^2S，它们称为扩展 I2S（I2S2_ext、I2S3_ext）（参见*图 288*）。因此，第一个 I2S 全双工接口基于 I2S2 和 I2S2_ext，第二个基于 I2S3 和 I2S3_ext。

注意： I2S2_ext 和 I2S3_ext 仅用于全双工模式。

图 288. I2S 全双工框图



1. 其中 X 可以是 2 或 3。

I2Sx 可以在主模式下工作。因此：

- 只有 I2Sx 可在半双工模式下输出 SCK 和 WS
- 只有 I2Sx 可在全双工模式下向 I2S2_ext 和 I2S3_ext 提供 SCK 和 WS。

扩展 I2S (I2Sx_ext) 只能用于全双工模式。I2Sx_ext 始终在从模式下工作。

I2Sx 和 I2Sx_ext 均可用于发送和接收。

27.4.3 支持的音频协议

四线总线通常需要处理时分复用的两个通道（右通道和左通道）上的音频数据。但是，只有一个 16 位寄存器进行发送和接收。所以，需由软件来把和通道对应的数据写入数据寄存器，以及从数据寄存器中读取数据，并通过检查 SPI_SR 寄存器中的 CHSIDE 位来识别对应的通道。始终先发送“左通道数据”，而后再发送右通道数据（对于 PCM 协议来说，CHSIDE 没有意义）。

有四种数据和帧格式组合，可采用下列格式发送数据：

- 将 16 位数据封装在 16 位帧中
- 将 16 位数据封装在 32 位帧中
- 将 24 位数据封装在 32 位帧中
- 将 32 位数据封装在 32 位帧中

当使用 32 位数据包中的 16 位数据时，前 16 位 (MSB) 为有效位，16 位 LSB 被强制清零，无需任何软件操作或 DMA 请求（只需一个读/写操作）。

如果应用程序首选 DMA，则 24 位和 32 位数据帧需要对 SPI_DR 执行两次 CPU 读取或写入操作，或者需要两次 DMA 操作。24 位的数据帧，硬件会将 8 位非有效位扩展到带有 0 位的 32 位。

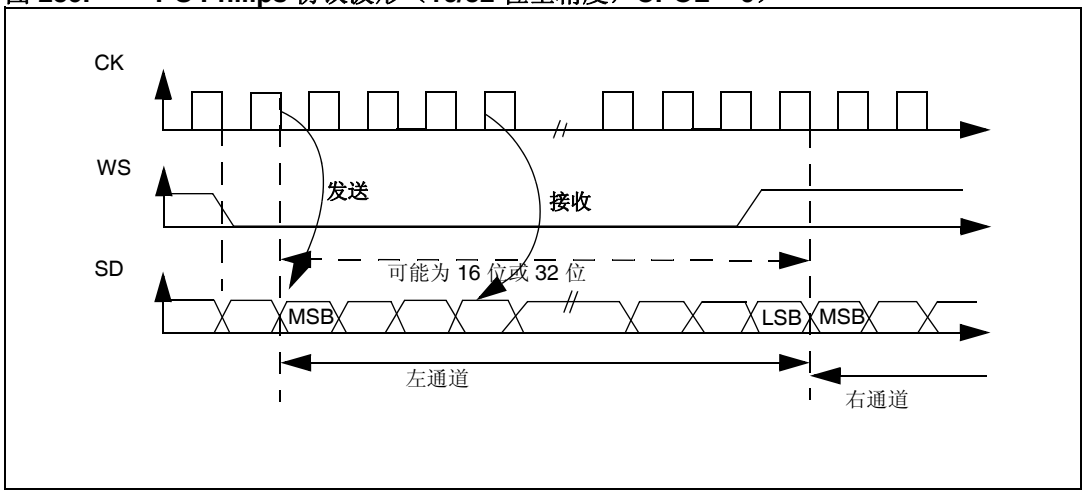
对于所有数据格式和通信标准而言，始终会先发送最高有效位（MSB 优先）。

I^2S 接口支持四种音频标准，可使用 SPI_I2SCFGR 寄存器中的 I2SSTD[1:0] 和 PCMSYNC 位对其进行配置。

I^2S Philips 标准

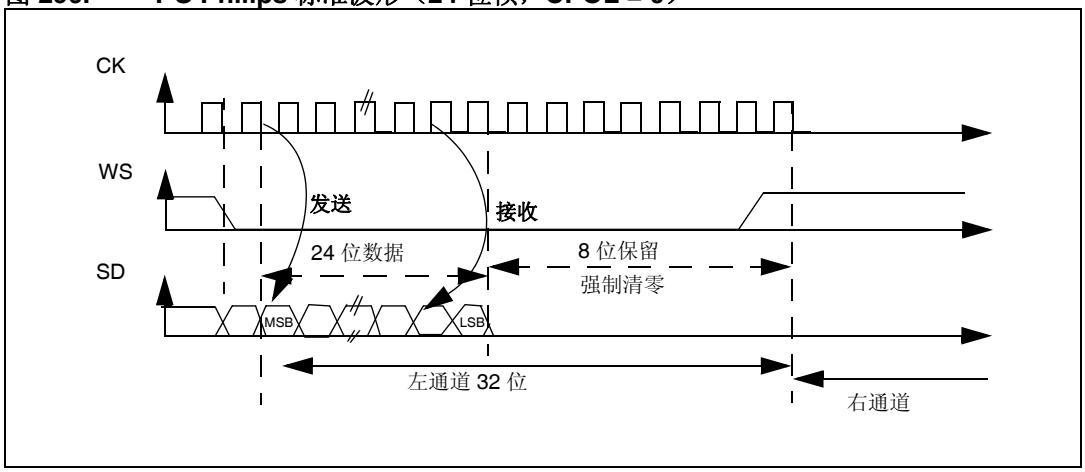
使用 WS 信号来指示当前正在发送的数据所属的通道。该信号从当前通道数据的第一个位 (MSB) 之前的一个时钟开始有效。

图 289. I^2S Philips 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 CK 的下降沿变化。

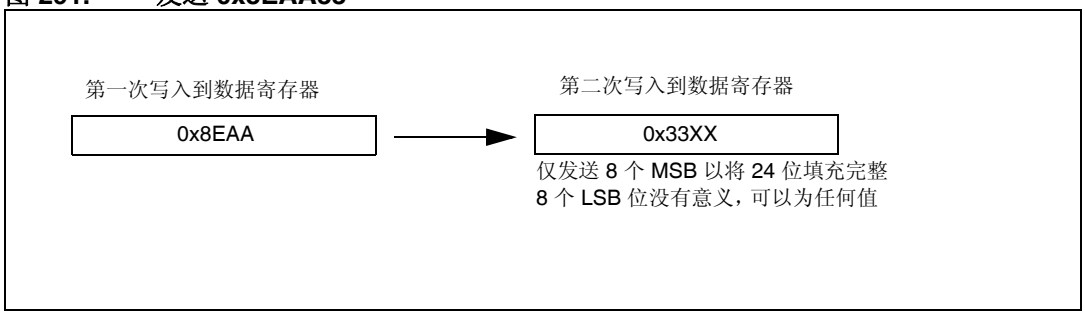
图 290. I²S Philips 标准波形（24 位帧，CPOL = 0）



该模式需要对 SPI_DR 执行两次写入或读取操作。

- 在发送模式下：
  如果需要发送 0x8EAA33（24 位）：

图 291. 发送 0x8EAA33



- 在接收模式下：
  如果接收数据 0x8EAA33：

图 292. 接收 0x8EAA33

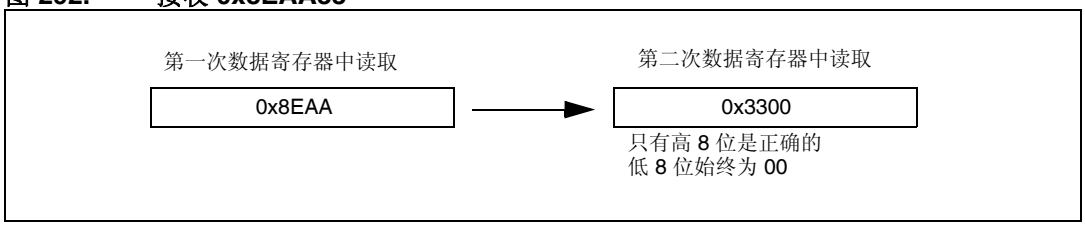
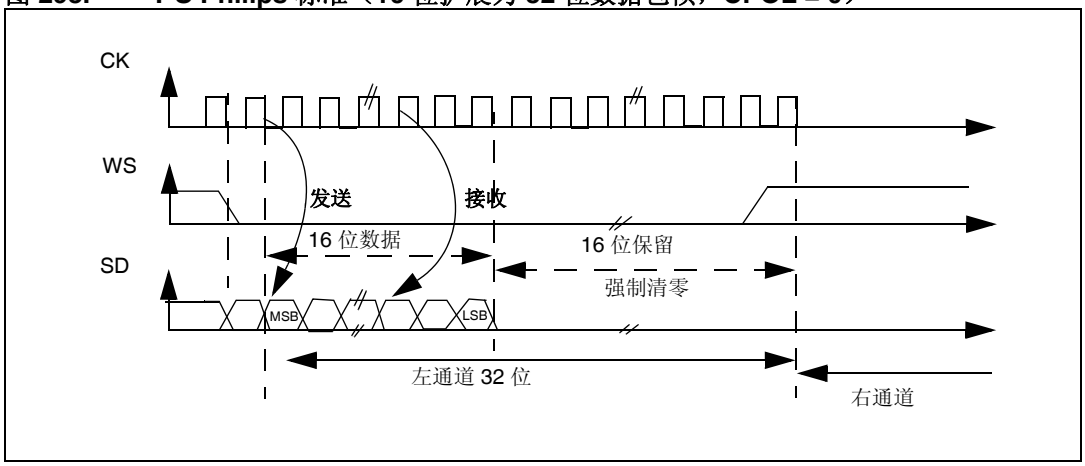


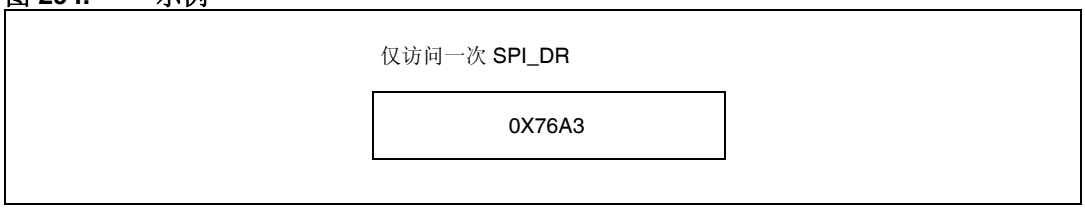
图 293. I^2S Philips 标准（16 位扩展为 32 位数据包帧，CPOL = 0）



如果 I^2S 配置成将 16 位数据扩展到 32 位帧，只需要访问一次寄存器 SPI_DR。硬件会将其余 16 个位强制设置为 0x0000，以便将数据扩展为 32 位格式。

如果要发送的数据或已接收的数据为 0x76A3（0x76A30000 扩展为 32 位），则需要执行*图 294* 中显示的操作。

图 294. 示例



发送时，每次将 16 位数据写入 SPI_DR，硬件就会置位 TXE 标志，并在中断使能的情况下触发中断，以把下一个要发送的数据加载到 SPI_DR。即使硬件填充的低 16 位 0x0000 还未发送，也会如此，因为低 16 位是由硬件发送。

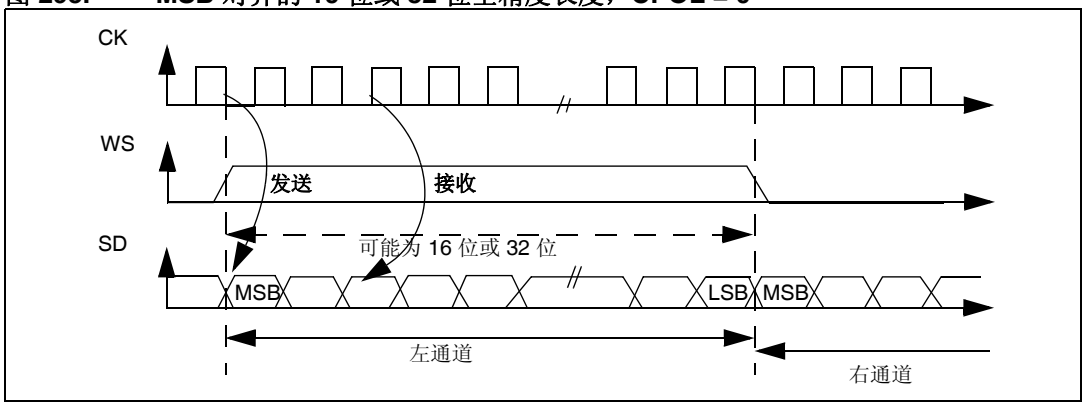
接收时，接收到第一个半字（高 16 位），则硬件将 RXNE 标志置 1，并在中断使能的情况下触发中断。

这样，就延长了两个写入或读取操作之间的时间间隔，从而可防止出现下溢或上溢情况（具体取决于数据传输方向）。

MSB 对齐标准

此标准同时生成 WS 信号和第一个数据位（即 MSBit）。

图 295. MSB 对齐的 16 位或 32 位全精度长度，CPOL = 0



发送方在时钟信号的下降沿改变数据；接收方在上升沿读取数据。

图 296. MSB 对齐的 24 位帧长度，CPOL = 0

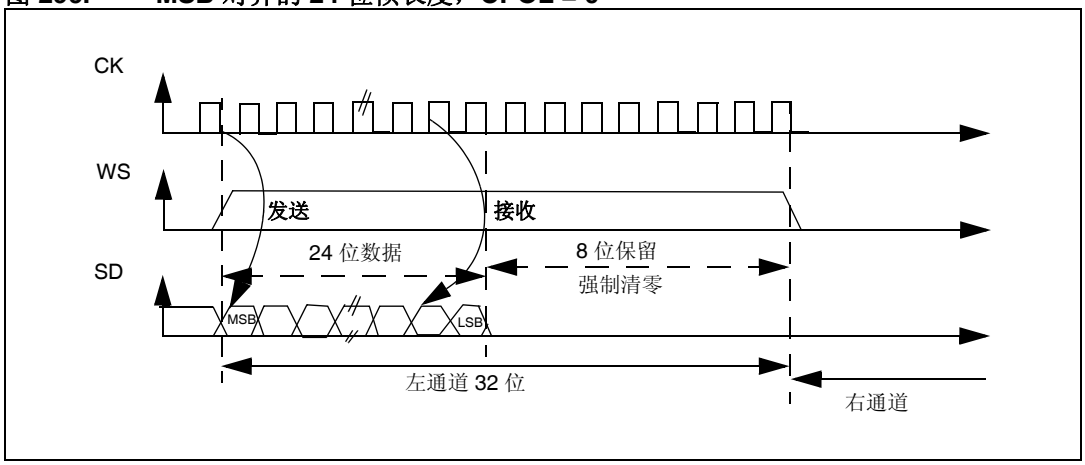
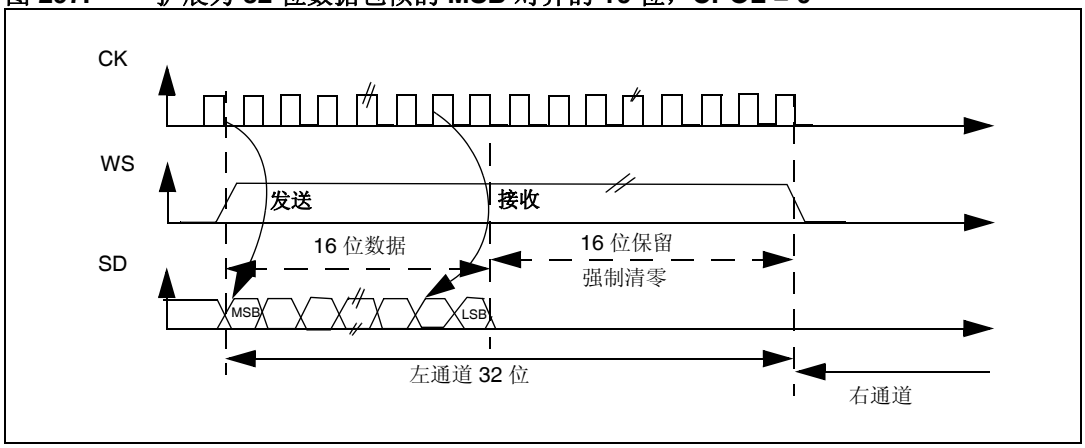


图 297. 扩展为 32 位数据包帧的 MSB 对齐的 16 位，CPOL = 0



LSB 对齐标准

该标准与 MSB 对齐标准类似（对于 16 位和 32 位全精度帧格式，没有任何不同）。

图 298. LSB 对齐的 16 位或 32 位全精度，CPOL = 0

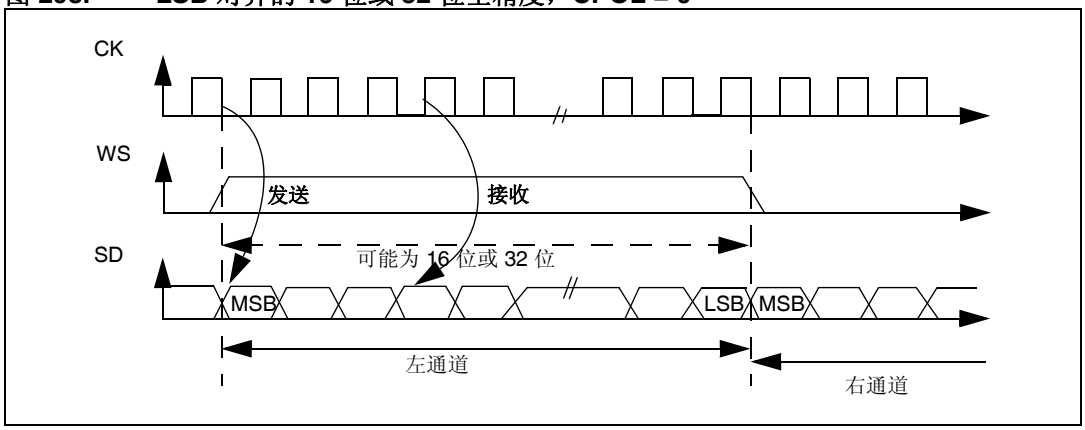
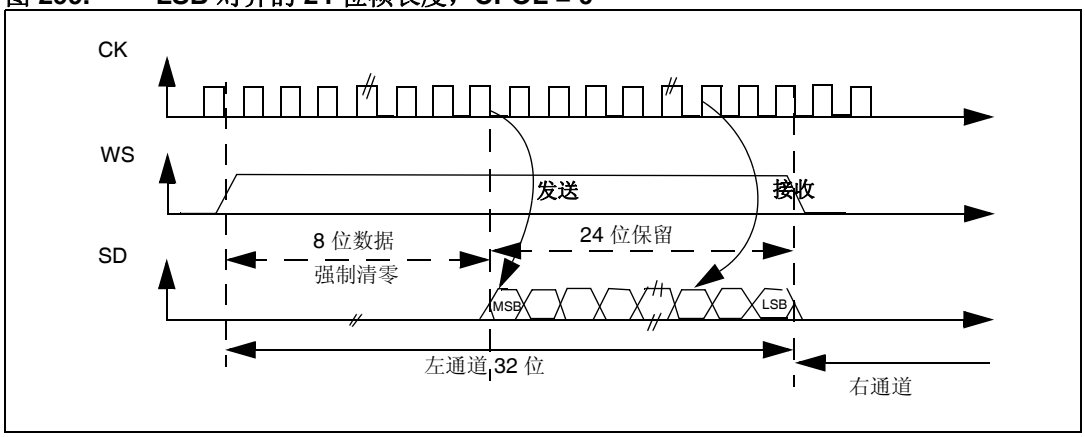


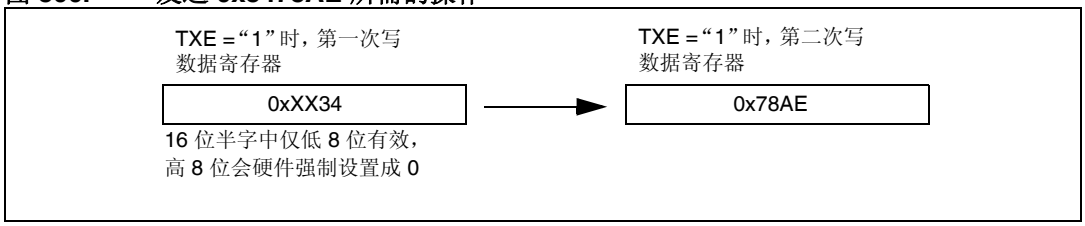
图 299. LSB 对齐的 24 位帧长度，CPOL = 0



- 在发送模式下：

  如果需要发送数据 0x3478A，则需要通过软件或 DMA 对 SPI_DR 寄存器执行两次写入操作。操作如下所示。

图 300. 发送 0x3478AE 所需的操作



- 在接收模式下：
  如果接收到数据 0x3478AE，则在每个RXNE事件时需要对 SPI_DR 执行两次连续的读取操作。

图 301. 接收 0x3478AE 时所需的操作

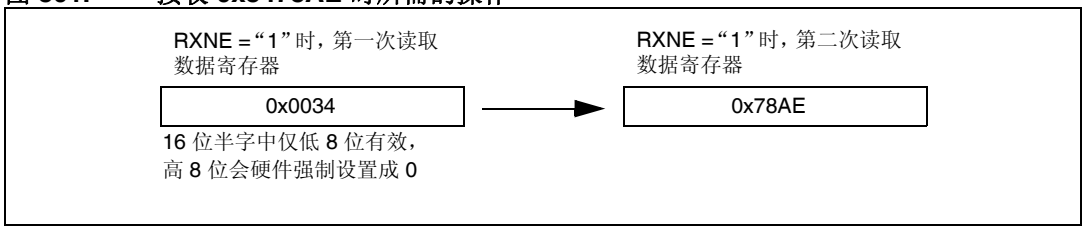
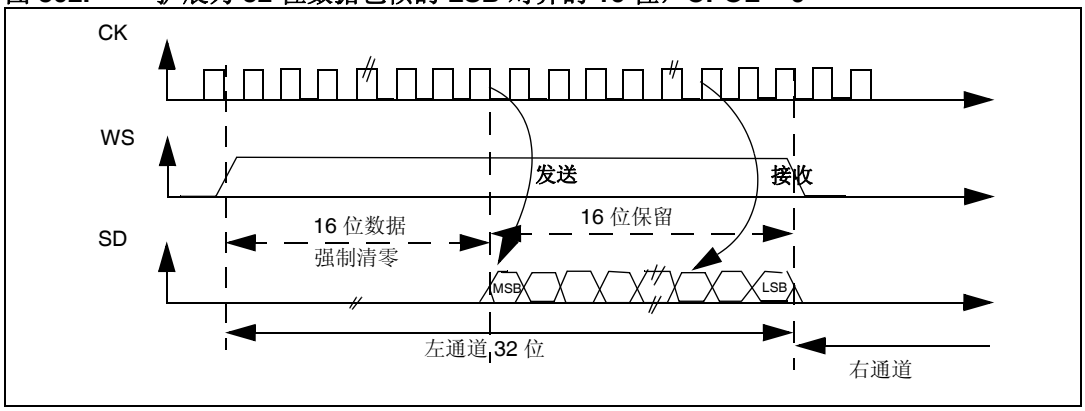


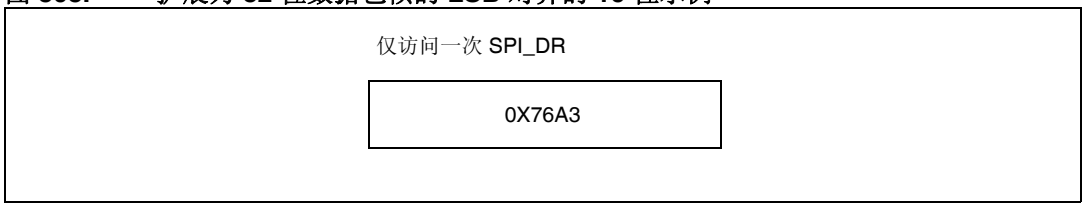
图 302. 扩展为 32 位数据包帧的 LSB 对齐的 16 位，CPOL = 0



如果 I^2S 配置成将 16 位数据扩展到 32 位帧，只需要访问一次寄存器 SPI_DR。硬件会将其余 16 个位强制设置为 0x0000，以便将数据扩展为 32 位格式。在这种情况下，其对应于半字 MSB。

如果要发送的数据或已接收的数据为 0x76A3（0x0000 76A3 扩展为 32 位），则需要执行*图 303* 中显示的操作。

图 303. 扩展为 32 位数据包帧的 LSB 对齐的 16 位示例



在发送模式下，TXE 置位时，应用程序需要写入要发送的数据（此例中，为 0x76A3）。首先发送 0x000 字段（扩展到 32 位）。有效数据 (0x76A3) 发送完毕后，TXE 会被再次置位。

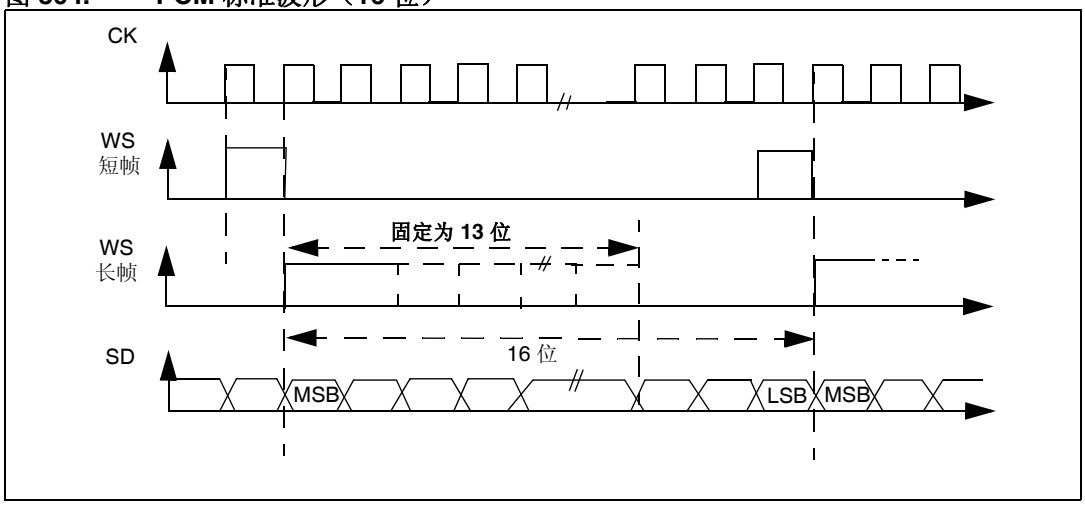
在接收模式下，当接收到有效半字后（而非 0x0000 字段），即会置位 RXNE。

这样，就延长了两个写入或读取操作之间的时间间隔，以防止出现下溢或上溢情况。

PCM 标准

对于 PCM 标准，无需使用通道信息。有两种 PCM 模式（短帧和长帧），并且可使用 SPI_I2SCFGR 中的 PCMSYNC 位来配置。

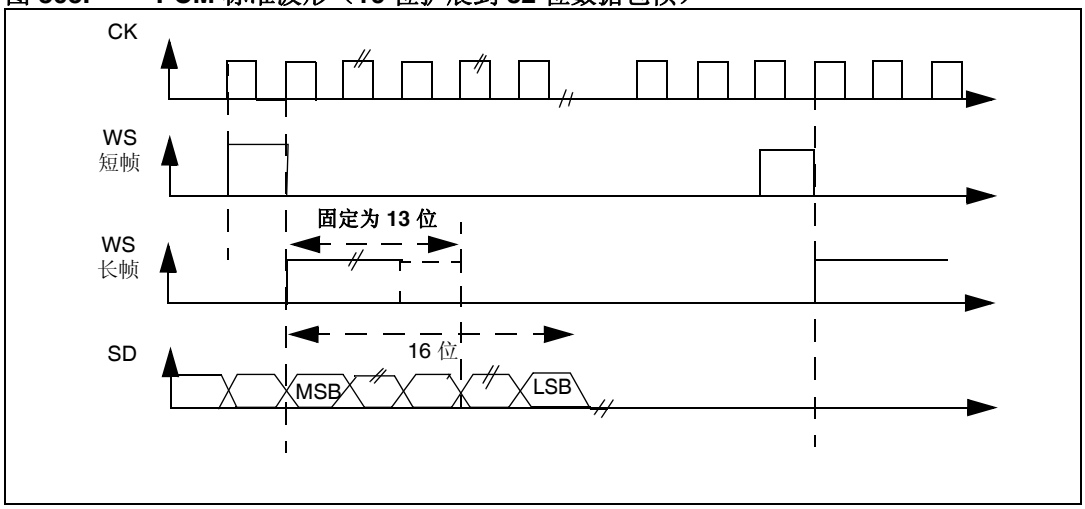
图 304. PCM 标准波形（16 位）



对于长帧同步，在主模式下会将 WS 信号持续 13 个周期。

对于短帧同步，WS 同步信号的持续时间仅为一个周期。

图 305. PCM 标准波形（16 位扩展到 32 位数据包帧）



注意：对于两种模式（主/从模式）和两种同步（短/长同步），即使在从模式下，也需要指定两组连续数据（以及两个同步信号）之间位的个数（SPI_I2SCFGR 寄存器中的 DATLEN 位和 CHLEN 位）。

27.4.4 时钟发生器

I^2S 比特率用来确定 I^2S 数据线上的数据流和 I^2S 时钟信号频率。

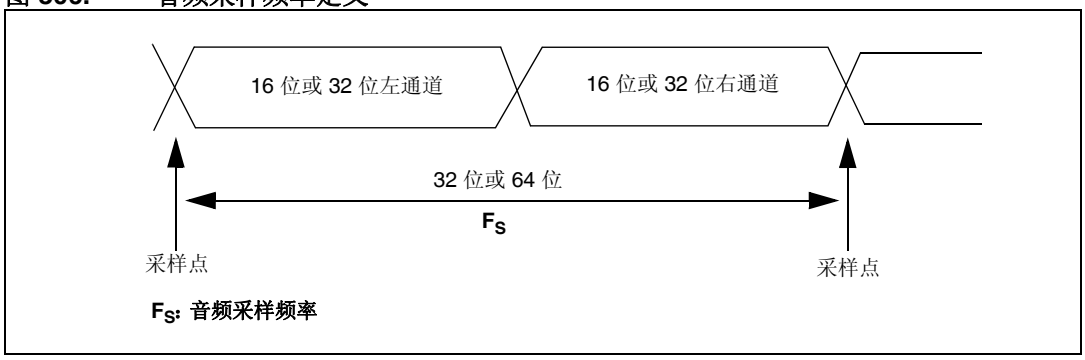
I^2S 比特率 = 每个通道的位数 × 通道数 × 音频采样频率

对于 16 位双通道音频，I^2S 比特率的计算公式如下：

I^2S 比特率 = 16 × 2 × F_S

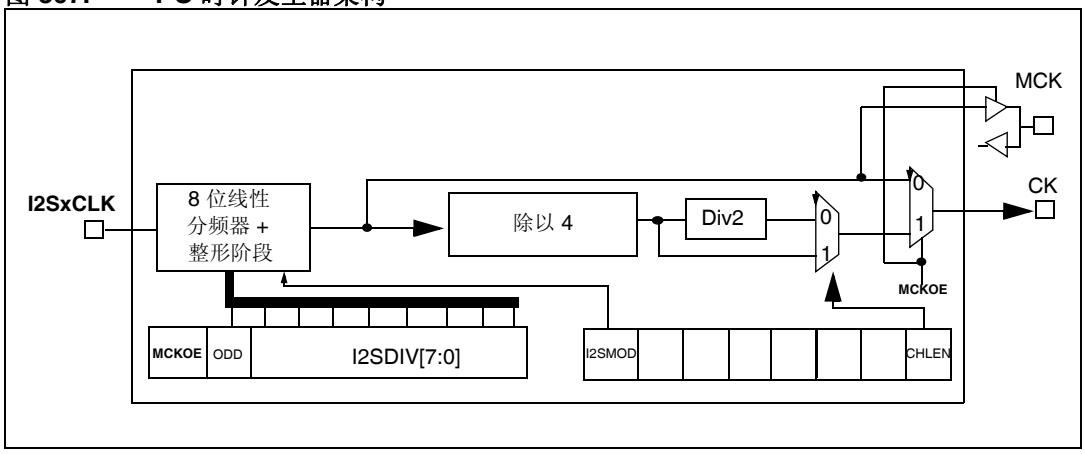
如果数据包为 32 位宽，则 I^2S 比特率 = 32 x 2 x F_S。

图 306. 音频采样频率定义



配置主模式时，需要正确地对线性分频器进行设置，以便采用所需的音频频率进行通信。

图 307. I^2S 时钟发生器架构



1. 其中，X 可以是 2 或 3。

图 306 展示了通信时钟架构。要实现高品质的音频性能，I2SxCLK 时钟源可为 PLLI2S 输出（通过 R 分频系数）或外部时钟（映射到 I2S_CKIN 引脚）。

音频采样频率可能是 192 kHz、96 kHz 或 48 kHz。为达到所需频率，需要根据以下公式对线性分频器进行编程：

输出主时钟（SPI_I2SPR 寄存器中的 MCKOE 置 1)时：

F_S = I2SxCLK / [(16*2)*((2*I2SDIV)+ODD)*8)]（通道帧宽度为 16 位时）

F_S = I2SxCLK / [(32*2)*((2*I2SDIV)+ODD)*4)]（通道帧宽度为 32 位时）

关闭主时钟输出（MCKOE 位清零）时：

F_S = I2SxCLK / [(16*2)*((2*I2SDIV)+ODD))]（通道帧宽度为 16 位时）

F_S = I2SxCLK / [(32*2)*((2*I2SDIV)+ODD))]（通道帧宽度为 32 位时）

表 125 提供了针对不同时钟配置的示例精度值。

注意： *还有其他配置可达到更好的时钟精度。*

表 125. 音频频率精度（针对 PLLM VCO = 1 MHz 或 2 MHz）(1)

主时钟	目标 f_S (Hz)	数据格式	PLLI2SN	PLLI2SR	I2SDIV	I2SODD	实时 f_S (Hz)	误差
输出关闭	8000	16 位	192	2	187	1	8000	0.0000%
		32 位	192	3	62	1	8000	0.0000%
	16000	16 位	192	3	62	1	16000	0.0000%
		32 位	256	2	62	1	16000	0.0000%
	32000	16 位	256	2	62	1	32000	0.0000%
		32 位	256	5	12	1	32000	0.0000%
	48000	16 位	192	5	12	1	48000	0.0000%
		32 位	384	5	12	1	48000	0.0000%
	96000	16 位	384	5	12	1	96000	0.0000%
		32 位	424	3	11	1	96014.49219	0.0151%
	22050	16 位	290	3	68	1	22049.87695	0.0006%
		32 位	302	2	53	1	22050.23438	0.0011%
	44100	16 位	302	2	53	1	44100.46875	0.0011%
		32 位	429	4	19	0	44099.50781	0.0011%
	192000	16 位	424	3	11	1	192028.9844	0.0151%
		32 位	258	3	3	1	191964.2813	0.0186%
输出使能	8000	无关	256	5	12	1	8000	0.0000%
	16000	无关	213	2	13	0	16000.60059	0.0038%
	32000	无关	213	2	6	1	32001.20117	0.0038%
	48000	无关	258	3	3	1	47991.07031	0.0186%
	96000	无关	344	2	3	1	95982.14063	0.0186%
	22050	无关	429	4	9	1	22049.75391	0.0011%
	44100	无关	271	2	6	0	44108.07422	0.0183%

1. 该表格仅给出了不同时钟配置的示例值。还有其他配置可达到更好的时钟精度。

27.4.5 I²S 主模式

I²S 可如下配置：

- 发送主器件或接收主器件（使用 I2Sx 的半双工模式）
- 同时收发的主器件（使用 I2Sx 和 I2Sx_ext 的全双工模式）。

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 SPI_I2SPR 的 MCKOE 位来选择输出或者不输出主时钟 (MCK)。

步骤

1. 设置 SPI_I2SPR 寄存器的 I2SDIV[7:0] 位，以定义串行时钟波特率，从而达到适当的音频采样频率。SPI_I2SPR 寄存器的 ODD 位也需要设置。
2. 设置 CKPOL 位，定义时钟在空闲时的电平状态。如果需要为外部 DAC/ADC 音频组件提供主时钟 MCK，则将 SPI_I2SPR 寄存器的 MCKOE 位置 1（I2SDIV 和 ODD 值应根据 MCK 输出的状态进行计算。有关详细信息，请参见*第 27.4.4 节：时钟发生器*）。
3. 将 SPI_I2SCFGR 中的 I2SMOD 位置 1 以激活 I²S 功能，通过 I2SSTD[1:0] 和 PCMSYNC 位选择 I²S 标准，通过 DATLEN[1:0] 位选择数据长度并通过配置 CHLEN 位选择每个通道的位数。此外，通过 SPI_I2SCFGR 寄存器的 I2SCFG[1:0] 位选择 I²S 主模式和方向（发送器或接收器）。
4. 如果需要，通过对 SPI_CR2 寄存器执行写操作来选择所有可能的中断源和 DMA 功能。
5. SPI_I2SCFGR 寄存器的 I2SE 位必须置 1。

WS 和 CK 配置为输出模式。如果 SPI_I2SPR 的 MCKOE 位置 1，则 MCK 也是输出。

发送序列

将半字写入发送缓冲区后，发送序列随即开始。

通常，写入发送缓冲区的第一个数据对应于左通道数据。数据从发送缓冲区传输到移位寄存器时，TXE 置 1，并且必须将对应于右通道的数据写入发送缓冲区。CHSIDE 标志指示将发送的数据对应的通道。TXE 标志置 1 时，CHSIDE 标志有意义，因为该标志在 TXE 变为高电平时进行更新。

一个完整帧表示先进行左通道数据发送再进行右通道数据发送。不存在仅发送左通道的部分帧。

首位发送期间，数据按半字并行加载到 16 位移位寄存器中，然后以串行方式移位并输出到 MOSI/SD 引脚（MSB 在前）。每次数据从发送缓冲区传输到移位寄存器后，TXE 标志都将置 1，如果 SPI_CR2 寄存器的 TXEIE 位置 1，将产生中断。

有关各种 I²S 标准模式中的写操作的更多详细信息，请参见*第 27.4.3 节：支持的音频协议*。

为确保连续进行音频数据发送，必须在当前数据发送结束前将下一个要发送数据写入 SPI_DR。

要通过将 I2SE 清零来关闭 I²S，必须等待 TXE 置 1 且 BSY 清零。

接收序列

此工作模式与发送模式相同，只有第 3 点存在不同（请参见*第 27.4.5 节：I²S 主模式*所述的步骤），即通过 I2SCFG[1:0] 位设置主器件接收模式。

无论数据或通道长度如何，音频数据始终按 16 位数据包进行接收。这意味着，每当接收缓冲区满时，RXNE 标志即置 1，并且如果 SPI_CR2 寄存器的 RXNEIE 位置 1，还将产生中断。所接收的右通道或左通道的音频值可通过一次或两次接收操作进入接收缓冲区，具体取决于数据和通道长度配置。

读取 SPI_DR 寄存器即会使 RXNE 位清零。

CHSIDE 在每次接收后进行更新。它由 I²S 单元所产生的 WS 信号触发。

有关各种 I²S 标准模式中的读操作的更多详细信息，请参见*第 27.4.3 节：支持的音频协议*。

如果在先前收到的数据尚未读取时又接收到新数据，将产生溢出错误并将 OVR 标志置 1。如果 SPI_CR2 寄存器的 ERRIE 位置 1，将产生中断以指示该错误。

要关闭 I²S，需要执行特定操作来确保 I²S 正确完成传输周期而不启动新的数据传输。该序列取决于数据和通道长度的配置，以及所选的音频协议模式。在以下情况下：

- 32 位通道长度上扩展的 16 位数据长度（DATLEN = 00 且 CHLEN = 1），使用 LSB 对齐模式 (I2SSTD = 10)
  a) 等待倒数第二个 RXNE = 1 (n – 1)
  b) 然后等待 17 个 I²S 时钟周期（使用软件循环）
  c) 关闭 I²S (I2SE = 0)
- 32 位通道长度上扩展的 16 位数据长度（DATLEN = 00 且 CHLEN = 1），使用 MSB 对齐、I²S 或 PCM 模式（即 I2SSTD = 00、I2SSTD = 01 或 I2SSTD = 11）
  a) 等待最后一个 RXNE
  b) 然后等待 1 个 I²S 时钟周期（使用软件循环）
  c) 关闭 I²S (I2SE = 0)
- 对于 DATLEN 和 CHLEN 的所有其它组合，无论通过 I2SSTD 位选择何种音频模式，都将执行以下序列来关闭 I²S：
  a) 等待倒数第二个 RXNE = 1 (n – 1)
  b) 然后等待 1 个 I²S 时钟周期（使用软件循环）
  c) 关闭 I²S (I2SE = 0)

注意： *传输期间，BSY 标志保持低电平。*

27.4.6 I²S 从模式

I²S 可如下配置：

- 发送从器件或接收从器件（使用 I2Sx 的半双工模式）
- 同时收发的从器件（使用 I2Sx 和 I2Sx_ext 的全双工模式）。

此工作模式所遵循的规则与 I²S 主模式基本相同。在从模式下，I²S 接口不产生时钟。时钟和 WS 信号从 I²S 接口所连接的外部主器件输入。这样，用户便不需要配置时钟。

应遵循如下配置步骤：

1. 将 SPI_I2SCFGR 寄存器的 I2SMOD 位置 1 以激活 I²S 功能，通过 I2SSTD[1:0] 位选择 I²S 标准，通过 DATLEN[1:0] 位选择数据长度并通过 CHLEN 位选择帧中每个通道的位数。此外，通过 SPI_I2SCFGR 寄存器的 I2SCFG[1:0] 位选择从器件的模式（发送或接收）。
2. 如果需要，通过对 SPI_CR2 寄存器执行写操作来选择所有可能的中断源和 DMA 功能。
3. SPI_I2SCFGR 寄存器的 I2SE 位必须置 1。

发送序列

当外部主器件发送时钟并且通过 NSS_WS 信号请求传输数据时，发送序列开始。必须首先使能从器件，然后外部主器件才能开始通信。主器件开始通信前，从器件还必须加载 I²S 数据寄存器。

对于 I²S、MSB 对齐和 LSB 对齐模式，要写入数据寄存器的第一个数据项对应于左通道的数据。通信开始时，数据从发送缓冲区传输到移位寄存器。TXE 标志随即置 1，以请求将右通道的数据写入 I²S 数据寄存器。

CHSIDE 标志指示将发送的数据对应的通道。与主发送模式相比，在从模式下，CHSIDE 由来自外部主器件的 WS 信号触发。这意味着，从器件需要首先为发送第一个数据做好准备，然后主器件才能产生时钟。WS 置位意味着首先发送左通道数据。

注意： *必须要在主器件发出的第一个时钟出现在 CK 线上至少 2 个 PCLK 周期之前置位 I2SE。*

首位发送期间，数据按半字从内部总线并行加载到 16 位移位寄存器中，然后以串行方式移位并输出到 MOSI/SD 引脚（MSB 在前）。每次数据从发送缓冲区传输到移位寄存器后，TXE 标志都将置 1，如果 SPI_CR2 寄存器的 TXEIE 位置 1，将产生中断。

请注意，仅当 TXE 标志为 1 时，才可以尝试向发送缓冲区写入数据。

有关各种 I²S 标准模式中的写操作的更多详细信息，请参见*第 27.4.3 节：支持的音频协议*。

为确保连续进行音频数据发送，必须在当前数据发送结束前将下一个要发送数据写入 SPI_DR 寄存器。如果在数据尚未写入 SPI_DR 寄存器时下一个数据通信的首个时钟边沿到来，下溢标志将置 1 并可能产生中断。通过这种方式，软件可以获知所传输的数据不正确。如果 SPI_CR2 寄存器的 ERRIE 位置 1，则当 SPI_SR 寄存器中的 UDR 标志变为 1 时，将产生中断。这种情况下，必须关闭 I²S 并从左通道开始重新启动数据传输。

要通过将 I2SE 位清零来关闭 I²S，必须等待 TXE 置 1 且 BSY 清零。

接收序列

此工作模式与发送模式相同，只有第 1 点存在不同（请参见*第 27.4.6 节：I²S 从模式*所述的步骤），即通过 SPI_I2SCFGR 寄存器的 I2SCFG[1:0] 位设置从器件接收模式。

无论数据长度或通道长度如何，音频数据始终按 16 位数据包进行接收。这意味着，每当接收缓冲区填满时，SPI_SR 寄存器中的 RXNE 标志即置 1，并且如果 SPI_CR2 寄存器的 RXNEIE 位置 1，还将产生中断。所接收的右通道或左通道的音频值可能通过一次或两次接收操作进入接收缓冲区，具体取决于数据长度和通道长度配置。

每次接收要从 SPI_DR 读取的数据时，CHSIDE 标志都将更新。该标志由外部主器件所管理的外部 WS 线路触发。

读取 SPI_DR 寄存器即会使 RXNE 位清零。

有关各种 I²S 标准模式中的读操作的更多详细信息，请参见*第 27.4.3 节：支持的音频协议*。

如果在先前收到的数据尚未读取时又接收到新数据，将产生溢出错误并将 OVR 标志置 1。如果 SPI_CR2 寄存器的 ERRIE 位置 1，将产生中断以指示该错误。

要在接收模式下关闭 I²S，必须在接收到最后一个 RXNE = 1 后立即将 I2SE 清零。

注意： *外部主器件应能够通过音频通道以 16 位或 32 位数据包发送/接收数据。*

27.4.7 状态标志

应用程序可通过三个状态标志来全面监视 I²S 总线的状态。

忙标志 (BSY)

BSY 标志由硬件置 1 和清零（写入此标志没有任何作用）。该标志表示 I²S 通信的状态。

硬件将 BSY 置 1 时，指示 I²S 正在忙于通信。在主接收模式 (I2SCFG = 11) 中，BSY 标志的情况例外，该标志在接收期间仍保持低电平。

如果软件需要关闭 I²S，可使用 BSY 标志检测传输是否结束。以避免破坏最后一个数据的传输。为此，必须严格遵循下述步骤。

在传输开始时（I²S 处于主接收器模式时除外）硬件将 BSY 标志置 1。

出现以下情况时，BSY 标志被硬件清零：

- 传输完成（主发送模式除外，在该模式下通信是连续的）
- 关闭 I²S 时

当通信连续时：

- 在主发送模式下，BSY 标志在所有传输期间均保持高电平
- 在从模式下，BSY 标志在每次传输之间变为低电平并持续一个 I²S 时钟周期

注意： *请勿使用 BSY 标志处理每次数据发送或接收，最好改用 TXE 标志和 RXNE 标志。*

发送缓冲区为空 (TXE)

如果此标志置 1，表示发送缓冲区为空，可将要发送的下一个数据加载到其中。发送缓冲区已包含要发送的数据时，TXE 标志复位。关闭 I²S（I2SE 位复位）时，该标志也会复位。

接收缓冲区非空 (RXNE)

此标志置 1 时，表示接收缓冲区中存在有效的已接收数据。读取 SPI_DR 寄存器时，该标志复位。

通道方向 (CHSIDE)

在发送模式下，此标志将在 TXE 变为高电平时进行刷新。此标志指示 SD 上要传输的数据所属的通道。如果在从发送模式下发生下溢错误事件，此标志将不可靠，在恢复通信前需要关闭并重新开启 I²S。

在接收模式下，此标志将在数据接收到 SPI_DR 中时进行刷新。此标志指示已接收的数据所属的通道。请注意，如果发生错误（例如 OVR），此标志将失去意义，应通过关闭并重新使能 I²S（根据需要更改配置）来将其复位。

此标志在 PCM 标准中没有意义（短帧和长帧模式）。

当 SPI_SR 中的 OVR 或 UDR 标志置 1，并且 SPI_CR2 中的 ERRIE 位也置 1 时，将产生中断。中断源被清除后，可通过读取 SPI_SR 状态寄存器来将此中断清零。

27.4.8 错误标志

I^2S 单元共有三个错误标志。

下溢标志 (UDR)

在从发送模式下，如果在软件尚未将任何值加载到 SPI_DR 之前出现第一个数据发送时钟，此标志将置 1。SPI_I2SCFGR 中的 I2SMOD 位置 1 时，可以使用此标志。如果 SPI_CR2 中的 ERRIE 位置 1，可产生中断。
UDR 位通过 SPI_SR 寄存器上的读操作进行清零。

上溢标志 (OVR)

如果在尚未从 SPI_DR 读取上一个数据时又接收到新数据，此标志将置 1。因此，传入的数据将丢失。如果 SPI_CR2 中的 ERRIE 位置 1，可产生中断。

这种情况下，将不会用接收到的新数据更新接收缓冲区的内容。对 SPI_DR 寄存器执行的读操作将返回先前正确接收的数据。主器件后续发送的所有其它半字都将丢失。

要将 OVR 位清零，应首先对 SPI_DR 寄存器执行读操作，然后再对 SPI_SR 寄存器进行读访问。

帧错误标志 (FRE)

仅当 I2S 配置为从模式时，此标志才可由硬件置 1。如果外部主器件没有按照从器件期望的那样改变 WS 信号，则此标志将置 1。如果失去同步，要从此状态中恢复并将外部主器件与 I2S 从器件重新同步，应遵循如下步骤：

1. 关闭 I2S
2. 在 WS 线上检测到正确的电平时将其重新使能（WS 在 I2S 模式下为高电平，在 MSB 对齐、LSB 对齐或 PCM 模式下为低电平）。

主器件与从器件之间的同步失效可能是由于 SCK 通信时钟或 WS 帧同步信号线上存在噪音干扰。如果 ERRIE 位置 1，可产生错误中断。读取状态寄存器时，同步失效标志 (FRE) 由软件清零。

27.4.9 I^2S 中断

表 126 为 I^2S 中断的列表。

表 126. I^2S 中断请求

中断事件	事件标志	使能控制位
发送缓冲区为空	TXE	TXEIE
接收缓冲区非空	RXNE	RXNEIE
上溢错误	OVR	ERRIE
下溢错误	UDR	
帧错误	FRE	ERRIE

27.4.10 DMA 特性

DMA 的工作方式与 SPI 模式完全相同。对于 I^2S 没有任何区别。只是由于不需要数据传输保护机制，因此 I^2S 模式没有 CRC 功能。

27.5 SPI 和 I²S 寄存器

有关寄存器说明中使用的缩写，请参见*第 47 页的第 1.1 节*。

外设寄存器可支持半字（16 位）或字（32 位）访问。

27.5.1 SPI 控制寄存器 1 (SPI_CR1)（不用于 I²S 模式）

SPI control register 1

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
BIDI MODE	BIDI OE	CRC EN	CRC NEXT	DFF	RX ONLY	SSM	SSI	LSB FIRST	SPE	BR [2:0]			MSTR	CPOL	CPHA
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 15 **BIDIMODE**：双向通信数据模式使能 (Bidirectional data mode enable)
- 0：选择双线单向通信数据模式
- 1：选择单线双向通信数据模式

*注意：**不适用于 I²S 模式***

位 14 **BIDIOE**：双向通信模式下的输出使能 (Output enable in bidirectional mode)

此位结合 BIDIMODE 位，用于选择双向通信模式下的传输方向
- 0：禁止输出（只接收模式）
- 1：使能输出（只发送模式）

注意：在主模式下，使用 MOSI 引脚；在从模式下，使用 MISO 引脚。

不适用于 I²S 模式

位 13 **CRCEN**：硬件 CRC 计算使能 (Hardware CRC calculation enable)
- 0：禁止 CRC 计算
- 1：使能 CRC 计算

注意：为确保正确操作，只应在禁止 SPI（SPE = “0”）时对此位执行写操作

不适用于 I²S 模式

位 12 **CRCNEXT**：下一次传输 CRC (CRC transfer next)
- 0：数据阶段（无 CRC 阶段）
- 1：下一次传输为 CRC（CRC 阶段）

注意：当 SPI 配置为全双工或只发送模式时，只要最后一个数据写入 SPI_DR 寄存器，就必须对 CRCNEXT 执行写操作。

当 SPI 配置为只接收模式时，必须在接收到倒数第二个数据之后将 CRCNEXT 置 1。

当传输由 DMA 管理时，此位应保持清零状态。

不适用于 I²S 模式

位 11 **DFF**：数据帧格式 (Data frame format)
- 0：为发送/接收选择 8 位数据帧格式
- 1：为发送/接收选择 16 位数据帧格式

注意：为确保正确操作，只应在禁止 SPI（SPE = “0”）时对此位执行写操作

适用于 I²S 模式

位 10 **RXONLY**：只接收 (Receive only)

此位结合 BIDIMODE 位，用于选择双线单向模式下的传输方向。此位也适用于多从模式系统，在此类系统中，不会访问特定从器件，也不会损坏访问的从器件的输出。

0：全双工（发送和接收）

1：关闭输出（只接收模式）

*注意：**不适用于 I²S 模式***

位 9 **SSM**：软件从器件管理 (Software slave management)

当 SSM 位置 1 时，NSS 引脚输入替换为 SSI 位的值。

0：禁止软件从器件管理

1：使能软件从器件管理

*注意：**不适用于 I²S 模式和 SPI TI 模式***

位 8 **SSI**：内部从器件选择 (Internal slave select)

仅当 SSM 位置 1 时，此位才有效。此位的值将作用到NSS引脚上，并忽略 NSS 引脚的 IO 值。

*注意：**不适用于 I²S 模式和 SPI TI 模式***

位 7 **LSBFIRST**：帧格式 (Frame format)

0：先发送 MSB

1：先发送 LSB

注意：正在通信时不应更改此位。

不适用于 I²S 模式和 SPI TI 模式

位 6 **SPE**：SPI 使能 (SPI enable)

0：关闭外设

1：使能外设

*注意：**1- 不适用于 I²S 模式。***

注意：2- 关闭 SPI 时，请按照第 27.3.8 节：关闭 SPI 中所述的步骤操作。

位 5:3 **BR[2:0]**：波特率控制 (Baud rate control)

000: $f_{PCLK}/2$ 100: $f_{PCLK}/32$

001: $f_{PCLK}/4$ 101: $f_{PCLK}/64$

010: $f_{PCLK}/8$ 110: $f_{PCLK}/128$

011: $f_{PCLK}/16$ 111: $f_{PCLK}/256$

注意：正在通信时不应更改这些位。

不适用于 I²S 模式

位 2 **MSTR**：主模式选择 (Master selection)

0：从配置

1：主配置

注意：正在通信时不应更改此位。

不适用于 I²S 模式

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：正在通信时不应更改此位。

不适用于 I²S 模式和 SPI TI 模式

位 0 **CPHA**：时钟相位 (Clock phase)

0：从第一个时钟边沿开始采样数据

1：从第二个时钟边沿开始采样数据

注意：正在通信时不应更改此位。

不适用于 I²S 模式和 SPI TI 模式

27.5.2 SPI 控制寄存器 2 (SPI_CR2)

SPI control register 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
Reserved								TXEIE	RXNEIE	ERRIE	FRF	Res.	SSOE	TXDMAEN	RXDMAEN
								rw	rw	rw	rw		rw	rw	rw

位 15:8 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：屏蔽 TXE 中断
1：使能 TXE 中断。TXE 标志置 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：屏蔽 RXNE 中断
1：使能 RXNE 中断。RXNE 标志置 1 时产生中断请求。

位 5 **ERRIE**：错误中断使能 (Error interrupt enable)
此位用于控制在错误状况发生时是否产生中断（SPI 模式中的 CRCERR、OVR、MODF，以及 UDR、OVR 和 FRE）。
0：屏蔽错误中断
1：使能错误中断

位 4 **FRF**：帧格式 (Frame format)
0：SPI Motorola 模式
1：SPI TI 模式
注意：不适用于 I^2S 模式

位 3 保留。由硬件强制为零。

位 2 **SSOE**：SS 输出使能 (SS output enable)
0：在主模式下禁止 SS 输出，可在多主模式配置下工作
1：在主模式下使能 SS 输出，不能在多主模式环境下工作
注意：不适用于 I^2S 模式和 SPI TI 模式

位 1 **TXDMAEN**：发送缓冲区 DMA 使能 (Tx buffer DMA enable)
当此位置 1 时，每当 TXE 标志置 1 时，即产生 DMA 请求。
0：关闭发送缓冲区 DMA
1：使能发送缓冲区 DMA

位 0 **RXDMAEN**：接收缓冲区 DMA 使能 (Rx buffer DMA enable)
当此位置 1 时，每当 RXNE 标志置 1 时，即产生 DMA 请求。
0：关闭接收缓冲区 DMA
1：使能接收缓冲区 DMA

27.5.3 SPI 状态寄存器 (SPI_SR)

SPI status register

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
Reserved							FRE	BSY	OVR	MODF	CRC ERR	UDR	CHSIDE	TXE	RXNE
							r	r	r	r	rc_w0	r	r	r	r

位 15:9 保留。由硬件强制为零。

位 8 **FRE**：帧格式错误 (Frame format error)

0：没有帧格式错误

1：出现帧格式错误

此标志由硬件置 1，当读取 SPIx_SR 寄存器时由软件清零。

注意：当 SPI 在 TI 从模式或 I2S 从模式下工作时，使用此标志（请参见第 27.3.10 节）。

位 7 **BSY**：忙标志 (Busy flag)

0：SPI（或 I2S）不繁忙

1：SPI（或 I2S）忙于通信或者发送缓冲区不为空

此标志由硬件置 1 和清零。

注意：BSY 标志必须谨慎使用：请参见第 27.3.7 节：状态标志和第 27.3.8 节：关闭 SPI。

位 6 **OVR**：上溢标志 (Overrun flag)

0：未发生上溢

1：发生上溢

此标志由硬件置 1，可由软件序列复位。有关软件序列的信息，请参见第 760 页的第 27.4.8 节。

位 5 **MODF**：模式故障 (Mode fault)

0：未发生模式故障

1：发生模式故障

此标志由硬件置 1，可由软件序列复位。有关软件序列，请参见第 743 页的第 27.3.10 节。

注意：不适用于 I^2S 模式

位 4 **CRCERR**：CRC 错误标志 (CRC error flag)

0：接收到的 CRC 值与 SPI_RXCRCR 值匹配

1：接收到的 CRC 值与 SPI_RXCRCR 值不匹配

此标志由硬件置 1，通过软件写入 0 来清零。

注意：不适用于 I^2S 模式

位 3 **UDR**：下溢标志 (Underrun flag)

0：未发生下溢

1：发生下溢

此标志由硬件置 1，可由软件序列复位。有关软件序列，请参见第 760 页的第 27.4.8 节。

注意：不适用于 SPI 模式

位 2 **CHSIDE**：通道信息 (Channel side)

0：发送或接收左通道信息

1：发送或接收右通道信息

注意：不适用于 SPI 模式。在 PCM 模式下没有意义

位 1 **TXE**：发送缓冲区为空 (Transmit buffer empty)

0：发送缓冲区非空

1：发送缓冲区为空

位 0 **RXNE**：接收缓冲区非空 (Receive buffer not empty)

0：接收缓冲区为空

1：接收缓冲区非空

27.5.4 SPI 数据寄存器 (SPI_DR)

SPI data register

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	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 15:0 **DR[15:0]**：数据寄存器 (Data register)

已接收或者要发送的数据。

数据寄存器分为 2 个缓冲区，一个用于写入（发送缓冲区），一个用于读取（接收缓冲区）。对数据寄存器执行写操作时，数据将写入发送缓冲区，从数据寄存器执行读取时，将返回接收缓冲区中的值。

针对 SPI 模式的说明：

发送或接收的数据为 8 位或 16 位，具体取决于数据帧格式选择位（SPI_CR1 寄存器中的 DFF）。必须在使能 SPI 前进行此项选择，以确保操作正确。

对于 8 位数据帧，缓冲区为 8 位，只有寄存器的 LSB (SPI_DR[7:0]) 用于发送/接收。在接收模式下，寄存器的 MSB (SPI_DR[15:8]) 强制为 0。

对于 16 位数据帧，缓冲区为 16 位，整个寄存器 SPI_DR[15:0] 均用于发送/接收。

27.5.5 SPI CRC 多项式寄存器 (SPI_CRCPR)（不用于 I^2S 模式）

SPI CRC polynomial register

偏移地址：0x10

复位值：0x0007

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

位 15:0 **CRCPOLY[15:0]**：CRC 多项式寄存器 (CRC polynomial register)

此寄存器包含用于 CRC 计算的多项式。

CRC 多项式 (0007h) 是此寄存器的复位值。可根据需要配置另一个多项式。

注意：不适用于 I^2S 模式。

27.5.6 SPI RX CRC 寄存器 (SPI_RXCRCR)（不用于 I²S 模式）

SPI RX CRC register

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
RXCRC[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

位 15:0 **RXCRC[15:0]**：接收 CRC 寄存器 (Rx CRC register)

使能 CRC 计算后，RxCRC[15:0] 位将包含后续接收字节在计算后所得到的 CRC 值。当 SPI_CR1 寄存器中的 CRCEN 位写入 1 时，此寄存器复位。CRC 通过 SPI_CRCPR 寄存器中编程的多项式连续计算。

数据帧格式设置为 8 位数据（SPI_CR1 的 DFF 位清零）时，仅考虑 8 个 LSB 位。CRC 计算依据任意 CRC8 标准进行。

选择 16 位数据帧格式（SPI_CR1 寄存器的 DFF 位置 1）时，考虑此寄存器的全部 16 个位。CRC 计算依据任意 CRC16 标准进行。

注意：当 BSY 标志置 1 时，读取此寄存器可能返回一个不正确的值。
不适用于 I²S 模式。

27.5.7 SPI TX CRC 寄存器 (SPI_TXCRCR)（不用于 I²S 模式）

SPI TX CRC register

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	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

位 15:0 **TXCRC[15:0]**：发送 CRC 寄存器 (Tx CRC register)

使能 CRC 计算后，TxCRC[7:0] 位将包含后续发送字节在计算后所得到的 CRC 值。当 SPI_CR1 寄存器中的 CRCEN 位写入 1 时，此寄存器复位。CRC 通过 SPI_CRCPR 寄存器中编程的多项式连续计算。

数据帧格式设置为 8 位数据（SPI_CR1 的 DFF 位清零）时，仅考虑 8 个 LSB 位。CRC 计算依据任意 CRC8 标准进行。

选择 16 位数据帧格式（SPI_CR1 寄存器的 DFF 位置 1）时，考虑此寄存器的全部 16 个位。CRC 计算依据任意 CRC16 标准进行。

注意：当 BSY 标志置 1 时，读取此寄存器可能返回一个不正确的值。
不适用于 I²S 模式。

27.5.8 SPI_I²S 配置寄存器 (SPI_I2SCFGR)

SPI_I²S configuration register

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
Reserved				I2SMOD	I2SE	I2SCFG		PCMSYNC	Reserved	I2SSTD		CKPOL	DATLEN		CHLEN
				rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw

位 15:12 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：应在 SPI 或 I²S 禁止时配置此位。

位 10 **I2SE**：I2S 使能 (I2S Enable)
0：关闭 I²S 外设
1：使能 I²S 外设
注意：不适用于 SPI 模式

位 9:8 **I2SCFG**：I2S 配置模式 (I2S configuration mode)
00：从模式 - 发送
01：从模式 - 接收
10：主模式 - 发送
11：主模式 - 接收
注意：应在 I²S 禁止时配置此位。
不适用于 SPI 模式

位 7 **PCMSYNC**：PCM 帧同步 (PCM frame synchronization)
0：短帧同步
1：长帧同步
注意：只有在 I2SSTD = 11（使用 PCM 标准）时，此位才有意义
不适用于 SPI 模式

位 6 保留：由硬件强制为 0

位 5:4 **I2SSTD**：I2S 标准选择 (I2S standard selection)
00：I²S Philips 标准。
01：MSB 对齐标准（左对齐）
10：LSB 对齐标准（右对齐）
11：PCM 标准
有关 I²S 标准的详细信息，请参见*第 747 页的第 27.4.3 节*。*不适用于 SPI 模式。*
注意：为确保正确运行，应在 I²S 关闭时配置这些位。

位 3 **CKPOL**：空闲状态的时钟电平 (Steady state clock polarity)
0：空闲状态时钟为低电平
1：空闲状态时钟为高电平
注意：为确保正确运行，应在 I²S 关闭时配置此位。
不适用于 SPI 模式

位 2:1 **DATLEN**：传输的数据长度 (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：为确保正确运行，应在 I²S 关闭时配置这些位。

不适用于 SPI 模式。

位 0 **CHLEN**：通道长度（每个音频通道的位数）(Channel length (number of bits per audio channel))

0：16 位

1：32 位

只有在 DATLEN = 00 时，此位的写操作才有意义，否则无论填入何值，通道长度始终由硬件固定为 32 位。*不适用于 SPI 模式。*

注意：为确保正确运行，应在 I²S 关闭时配置此位。

27.5.9 SPI_I²S 预分频器寄存器 (SPI_I2SPR)

SPI_I²S prescaler register

偏移地址：0x20

复位值：0000 0010 (0x0002)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						MCKOE	ODD	I2SDIV							
						rw	rw	rw							

位 15:10 保留，必须保持复位值。

位 9 **MCKOE**：主时钟输出使能 (Master clock output enable)

0：禁止主时钟输出

1：使能主时钟输出

注意：应在 I²S 禁止时配置此位。只有在 I²S 为主模式时，才会使用此位。

不适用于 SPI 模式。

位 8 **ODD**：预分频器的奇数因子 (Odd factor for the prescaler)

0：实际分频值为 = I2SDIV *2

1：实际分频值为 = (I2SDIV * 2)+1

请参见*第 754 页的第 27.4.4 节。不适用于 SPI 模式。*

注意：应在 I²S 禁止时配置此位。只有在 I²S 为主模式时，才会使用此位。

位 7:0 **I2SDIV**：I2S 线性预分频器 (I2S Linear prescaler)

I2SDIV [7:0] = 0 或 I2SDIV [7:0] = 1 为禁用值。

请参见*第 754 页的第 27.4.4 节。不适用于 SPI 模式。*

注意：应在 I²S 禁止时配置这些位。只有在 I²S 为主模式时，才会使用此位。

27.5.10 SPI 寄存器映射

下表显示了 SPI 寄存器映射和复位值。

表 127. SPI 寄存器映射和复位值

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**SPI_CR1**	Reserved																BIDIMODE	BIDIOE	CRCEN	CRCNEXT	DFF	RXONLY	SSM	SSI	LSBFIRST	SPE	BR [2:0]			MSTR	CPOL	CPHA
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	**SPI_CR2**	Reserved																								TXEIE	RXNEIE	ERRIE	FRF	Reserved	SSOE	TXDMAEN	RXDMAEN
	Reset value																									0	0	0	0		0	0	0
0x08	**SPI_SR**	Reserved																							FRE	BSY	OVR	MODF	CRCERR	UDR	CHSIDE	TXE	RXNE
	Reset value																								0	0	0	0	0	0	0	1	0
0x0C	**SPI_DR**	Reserved																DR[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	**SPI_CRCPR**	Reserved																CRCPOLY[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1
0x14	**SPI_RXCRCR**	Reserved																RxCRC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	**SPI_TXCRCR**	Reserved																TxCRC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1C	**SPI_I2SCFGR**	Reserved																				I2SMOD	I2SE	I2SCFG	PCMSYNC	Reserved	I2SSTD		CKPOL	DATLEN		CHLEN
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0
0x20	**SPI_I2SPR**	Reserved																						MCKOE	ODD	I2SDIV							
	Reset value																							0	0	0	0	0	0	0	0	1	0

有关寄存器边界地址的信息，请参见*第 52 页的表 2*。

28 安全数字输入/输出接口 (SDIO)

除非特别说明，否则本节适用于整个 STM32F4xx 系列器件。

28.1 SDIO 主要特性

SD/SDIO MMC 卡主机接口 (SDIO) 提供 APB2 外设总线与多媒体卡 (MMC)、SD 卡、SDIO 卡以及 CE-ATA 设备之间的接口。

多媒体卡协会网站 *www.mmca.org* 中提供了由 MMCA 技术委员会发布的多媒体卡系统规范。

SD 卡协会网站 *www.sdcard.org* 中提供了 SD 存储卡和 SD I/O 卡系统规范。

CE-ATA 工作组网站 *www.ce-ata.org* 中提供了 CE-ATA 系统规范。

SDIO 具有以下特性：

- 完全兼容*多媒体卡系统规范版本 4.2*。卡支持三种不同数据总线模式：1 位（默认）、4 位和 8 位
- 完全兼容先前版本的多媒体卡（向前兼容性）
- 完全兼容 *SD 存储卡规范版本 2.0*
- 完全兼容 *SD I/O 卡规范版本 2.0*：卡支持两种不同数据总线模式：1 位（默认）和 4 位
- 完全支持 CE-ATA 功能（完全符合 *CE-ATA 数字协议版本 1.1*）
- 对于 8 位模式，数据传输高达 48 MHz
- 数据和命令输出使能信号，控制外部双向驱动程序。

注意： *SDIO 不具备兼容 SPI 的通信模式。*

SD 存储卡协议是多媒体卡系统规范版本 2.11 定义的多媒体卡协议的超集。一些 SD 存储卡器件所需的命令，仅 SD-I/O 的卡或组合卡的 I/O 部分都不支持。部分上述命令（如擦除命令）无法在 SD I/O 器件中使用，因此在 SDIO 中不受支持。此外，一些命令在 SD 存储卡和 SD I/O 卡之间有所不同，因此在 SDIO 中不受支持。有关详细信息，请参见 SD I/O 卡规范版本 1.0。CE-ATA 支持通过 MMC 电气接口实现，这种接口使用一种利用现有 MMC 访问原语的协议。接口的电气和信令定义与 MMC 参考中定义的相同。

多媒体卡/SD 总线将卡连接到控制器。

当前版本的 SDIO 每次只支持一个 SD/SDIO/MMC4.2 卡，但支持多个 MMC4.1 或之前版本的卡。

28.2 SDIO 总线拓扑

总线的通信基于命令和数据传输。

多媒体卡/SD/SD I/O 总线上的基本事务是命令/响应事务。这些类型的总线事务直接在命令或响应结构中传输其信息。此外，某些操作具有数据令牌。

与 SD/SDIO 存储卡的数据相互传输在数据块中执行。与 MMC 的数据相互传输在数据块或流中执行。与 CE-ATA 器件的数据相互传输在数据块中执行。

图 308. SDIO“无响应”和“无数据”操作

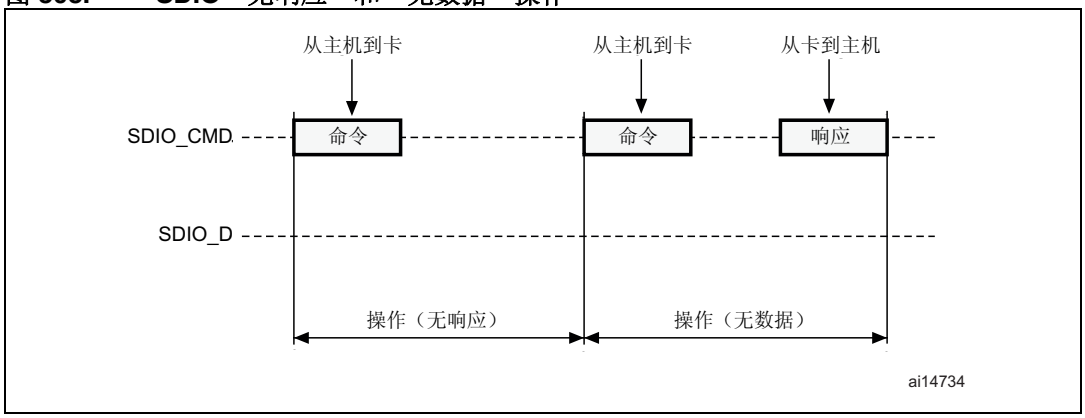


图 309. SDIO（多个）块读取操作

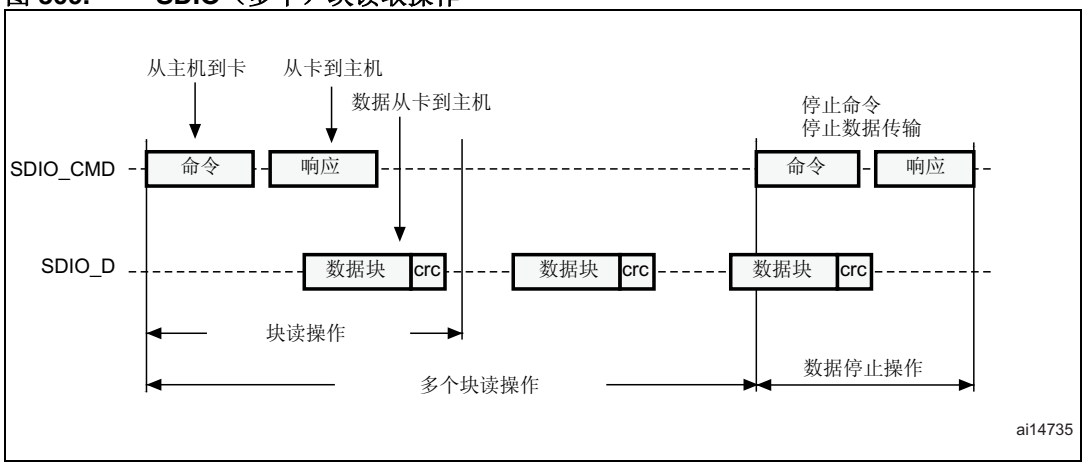
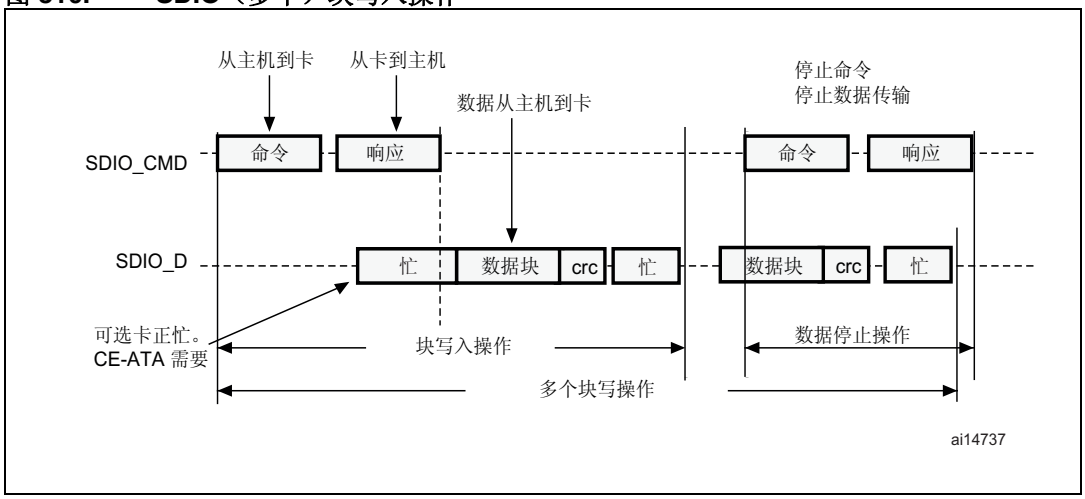


图 310. SDIO（多个）块写入操作



注意： *只要发出“繁忙”信号（SDIO_D0 被拉到低电平），SDIO 便不会发送任何数据。*

图 311. SDIO 连续读取操作

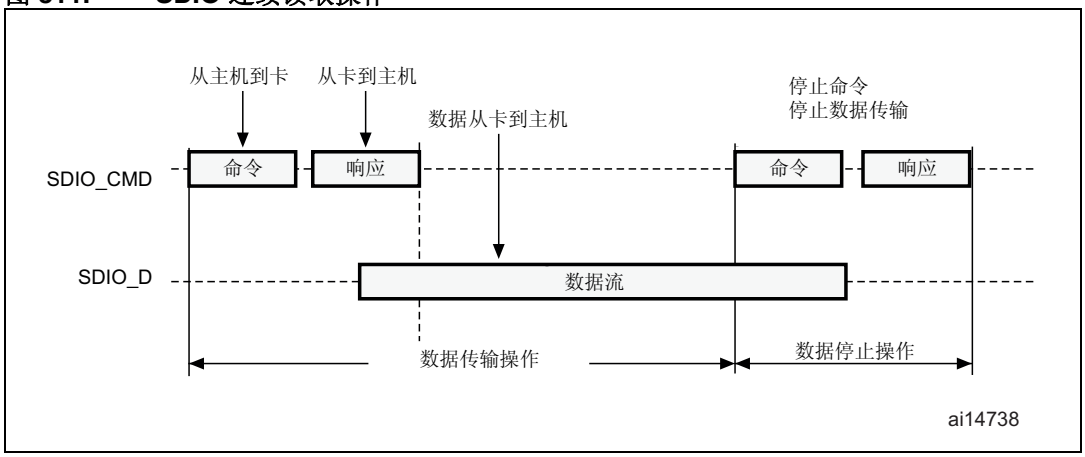
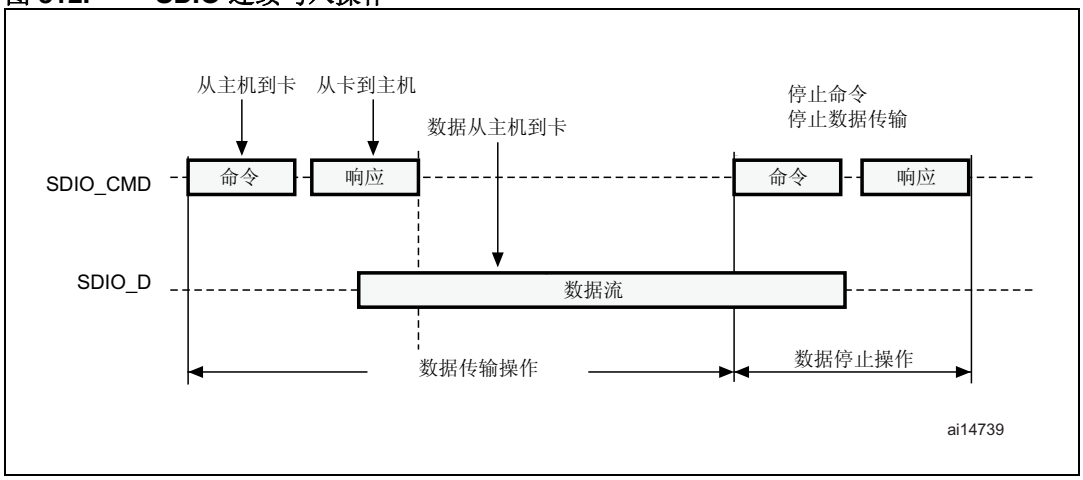


图 312. SDIO 连续写入操作

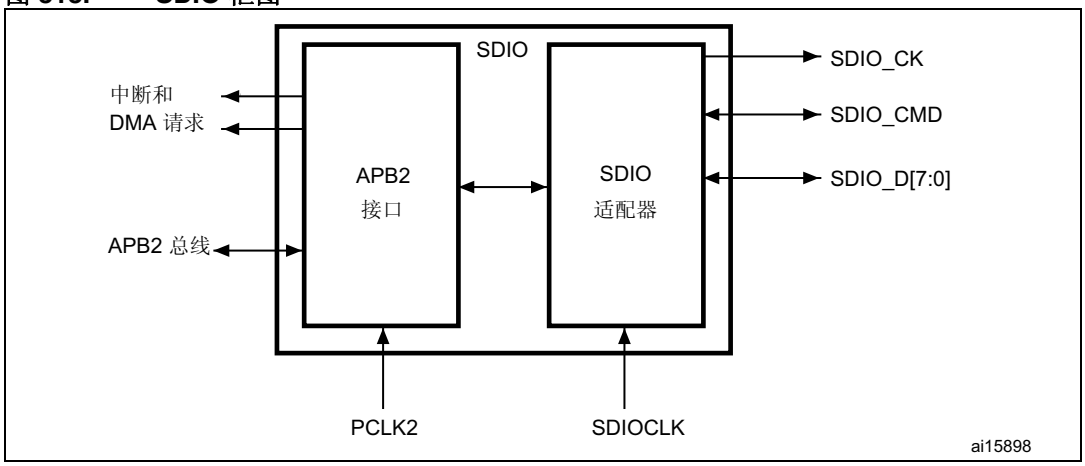


28.3 SDIO 功能说明

SDIO 由两部分组成：

- SDIO 适配器块提供特定于 MMC/SD/SD I/O 卡的所有功能，如时钟生成单元、命令和数据传输。
- APB2 接口访问 SDIO 适配器寄存器，并且生成中断和 DMA 请求信号。

图 313. SDIO 框图



默认情况下，SDIO_D0 用于数据传输。初始化后，主机可以更改数据总线宽度。

如果多媒体卡连接到总线，则 SDIO_D0、SDIO_D[3:0] 或 SDIO_D[7:0] 可以用于数据传输。MMC V3.31 或更低版本仅支持 1 位数据，因此只能使用 SDIO_D0。

如果 SD 或 SD I/O 卡连接到总线，则主机可以将数据传输配置为使用 SDIO_D0 或 SDIO_D[3:0]。所有数据线均以推挽模式运行。

SDIO_CMD 有两种操作模式：

- 开漏引脚，用于初始化（仅限于 MMC V3.31 或更低版本）
- 推挽，用于命令传输（SD/SD I/O 卡 MMC4.2 还将推挽驱动程序用于初始化）

SDIO_CK 是与卡相连的时钟：一个位在每个时钟周期内同时在命令和数据线上传输。对于 MMC V3.31，时钟频率可以在 0 MHz 到 20 MHz 之间变化，对于 MMC V4.0/4.2，可以在 0 到 48 MHz 之间变化，对于 SD/SD I/O 卡，可以在 0 到 25 MHz 之间变化。

SDIO 使用两个时钟信号：

- SDIO 适配器时钟 (SDIOCLK = 48 MHz)
- APB2 总线时钟 (PCLK2)

PCLK2 和 SDIO_CK 时钟频率必须满足以下条件：

$$Frequenc(PCLK2) \geq 3/8 \times Frequency(SDIO\_CK)$$

表 128 中的信号在多媒体卡/SD/SD I/O 卡总线中使用。

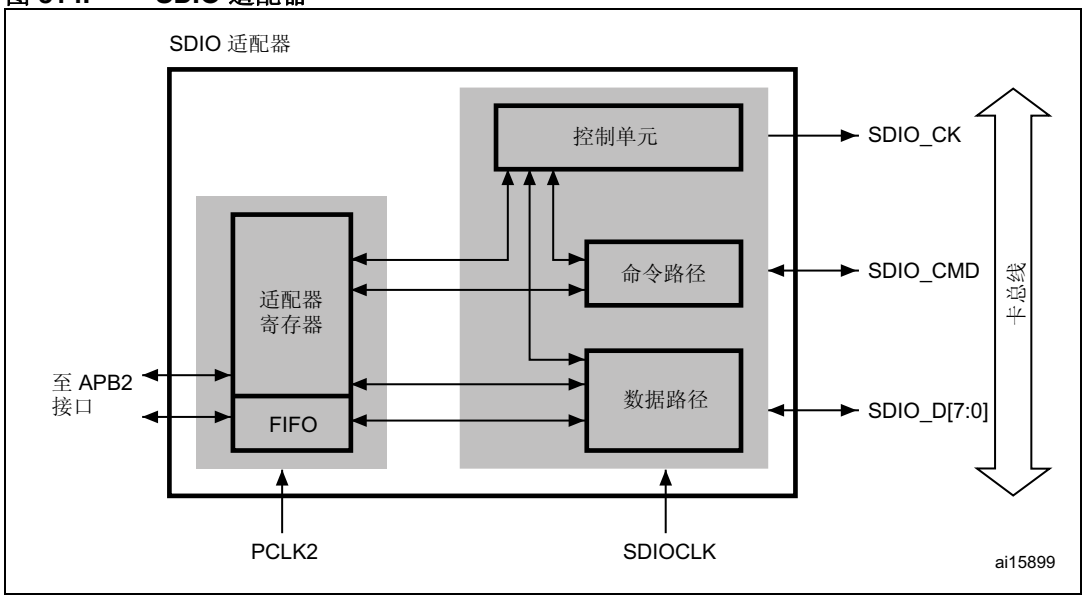
表 128. SDIO I/O 定义

引脚	方向	说明
SDIO_CK	输出	多媒体卡/SD/SDIO 卡时钟。该引脚是从主机到卡之间的时钟。
SDIO_CMD	双向	多媒体卡/SD/SDIO 卡命令。该引脚是双向命令/响应信号。
SDIO_D[7:0]	双向	多媒体卡/SD/SDIO 卡数据。这些引脚是双向数据总线。

28.3.1 SDIO 适配器

图 314 显示了 SDIO 适配器的简化框图。

图 314. SDIO 适配器



SDIO 适配器是一个多媒体卡/安全数字存储卡总线主设备，提供与多媒体卡堆栈或安全数字存储卡的接口。该适配器由五个子单元组成：

- 适配器寄存器块
- 控制单元
- 命令路径
- 数据路径
- 数据 FIFO

注意： *适配器寄存器和 FIFO 使用 APB2 总线时钟域 (PCLK2)。控制单元、命令路径和数据路径使用 SDIO 适配器时钟域 (SDIOCLK)。*

适配器寄存器块

适配器寄存器模块包含所有系统寄存器。该模块还生成将多媒体卡中的静态标志清零的信号。当写入 1 到 SDIO 清零寄存器中对应位的位置时，将生成清零信号。

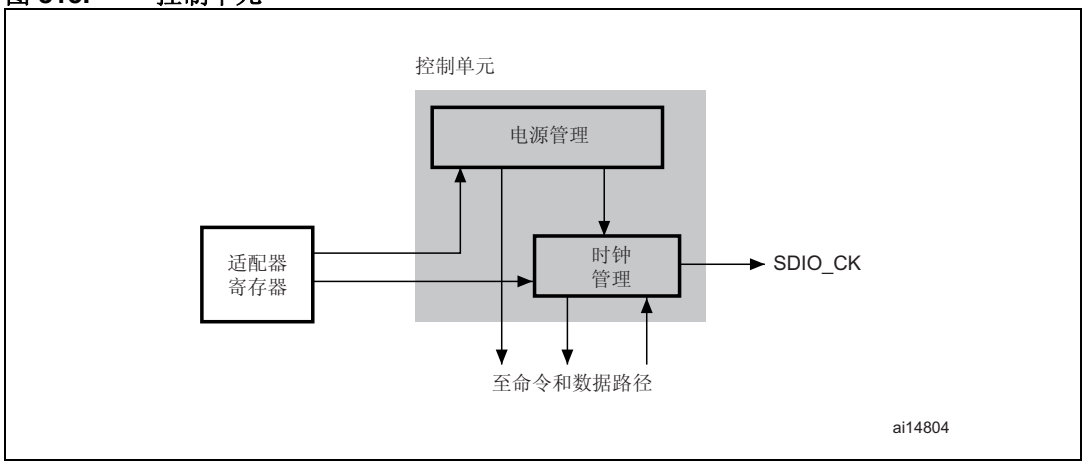
控制单元

控制单元包含电源管理功能和存储卡时钟的时钟分频器。

有三个电源阶段：

- 掉电
- 上电
- 通电

图 315. 控制单元



控制单元如*图 315* 所示。控制单元由电源管理子单元和时钟管理子单元组成。

电源管理子单元会在断电阶段和上电阶段中禁止卡总线输出信号。

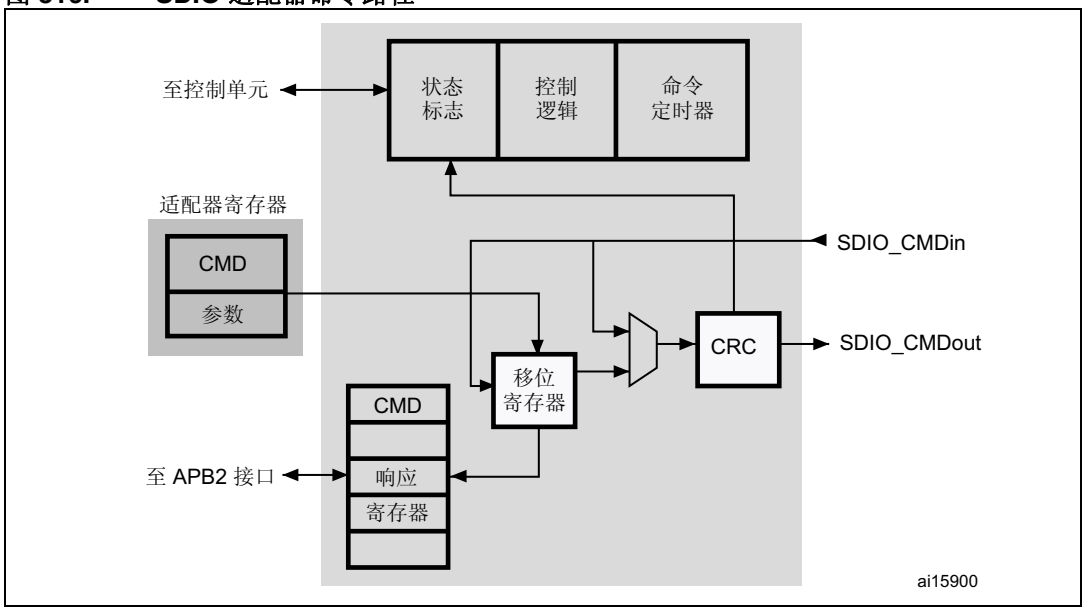
时钟管理子单元负责生成和控制 SDIO_CK 信号。SDIO_CK 输出可以使用时钟分频或时钟旁路模式。时钟输出在以下情况下无效：

- 复位后
- 在断电或上电阶段中
- 在使能了节电模式并且卡总线处于空闲状态的情况下（命令和数据路径子单元进入空闲阶段后的八个时钟周期之后）

命令路径

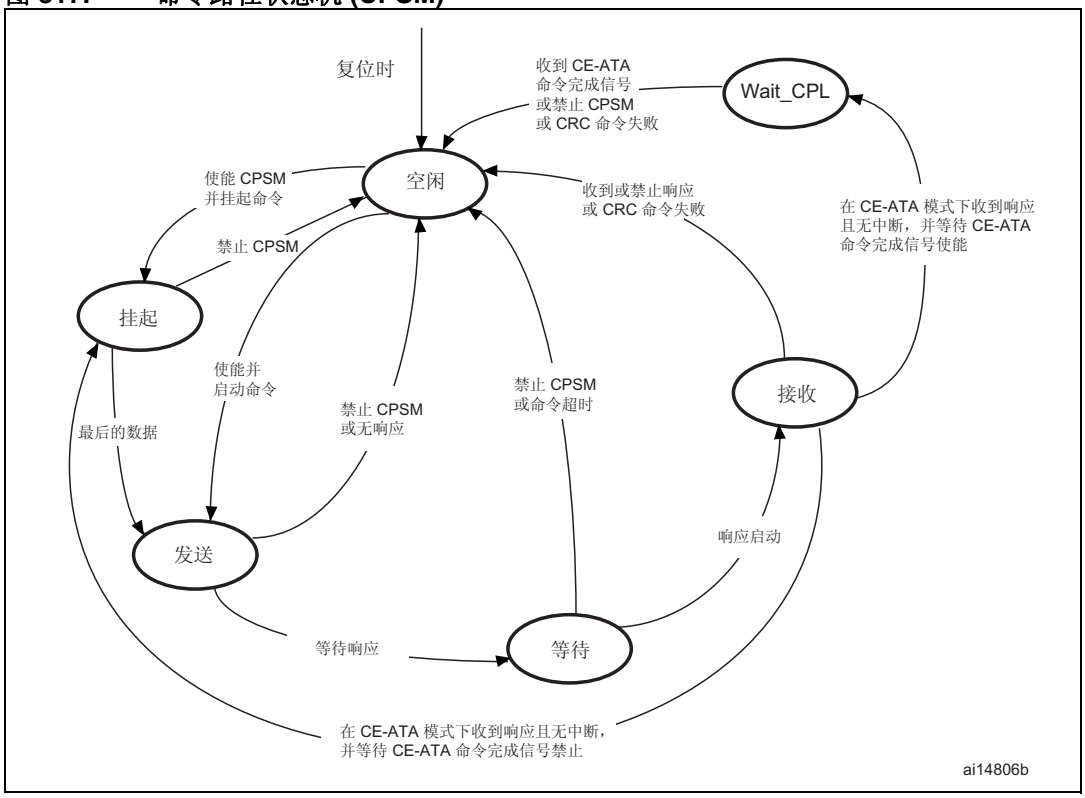
命令路径单元向卡发送命令并从卡接收响应。

图 316. SDIO 适配器命令路径



- 命令路径状态机 (CPSM)
  - 写入命令寄存器并且将使能位置 1 后，命令传输将开始。发送了命令后，命令路径状态机 (CPSM) 将将状态标志置 1，并且在不需要响应时进入空闲状态。如果需要响应，则等待响应（请参见 *第 776 页的图 317*）。收到响应时，将比较生成的 CRC 代码和内部生成代码，并且将相应的状态标志置 1。

图 317. 命令路径状态机 (CPSM)



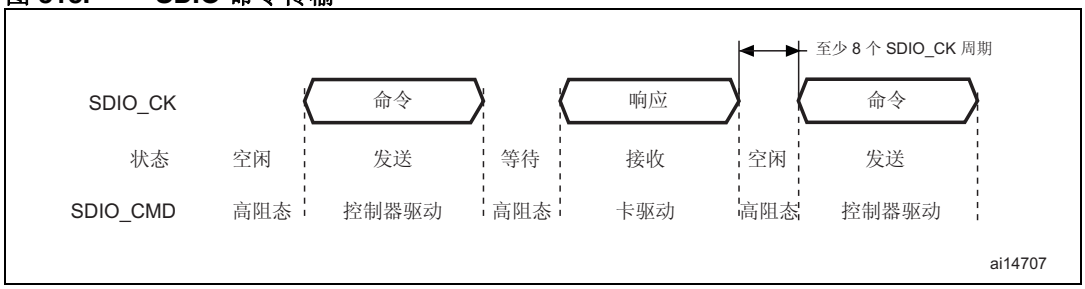
进入等待状态后，命令计时器便开始运行。如果 CPSM 尚未变为接收状态便已达到超时，则将超时标志置 1 并且进入空闲状态。

注意： *命令超时为固定值：64 个 SDIO_CK 时钟周期。*

如果在命令寄存器中将中断位置 1，则禁止计时器运行并且 CPSM 等待来自其中一个卡的中断请求。如果在命令寄存器中将挂起位置 1，则 CPSM 进入挂起状态并等待来自数据路径子单元的 CmdPend 信号。检测到 CmdPend 时，CPSM 将变为发送状态。这会使能数据计数器，用以触发停止命令传输。

注意： *CPSM 保持空闲状态至少八个 SDIO_CK 周期以满足 N_{CC} 和 N_{RC} 时序限制。N_{CC} 是两个主机命令之间的最小延迟，N_{RC} 是主机命令和卡响应之间的最小延迟。*

图 318. SDIO 命令传输



- 命令格式
  - 命令：命令是用于启动操作的令牌。命令从主机发送到单个卡（编址命令），或发送到所有已连接的卡（MMC V3.31 或更低版本可以使用广播命令）。命令在 CMD 线上以串行方式传输。所有命令都为固定长度 48 位。*表 129* 中显示了多媒体卡、SD 存储卡和 SDIO 卡的命令令牌的常规格式。CE-ATA 命令是 MMC 命令 V4.2 的扩展，因此格式相同。

    命令路径以半双工模式运行，因此可以发送或者接收命令和响应。如果 CPSM 未处于发送状态，则 SDIO_CMD 输出处于 Hi-Z 状态，如*第 777 页的图 318* 中所示。SDIO_CMD 上的数据与 SDIO_CK 的上升沿保持同步。*表 129* 显示了命令格式。

表 129. 命令格式

位的位置	宽度	值	说明
47	1	0	起始位
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：响应是一个令牌，它作为对先前接收命令的应答，从编址卡（或者，对于 MMC V3.31 或更低版本，从所有已连接的卡同步）发送到主机。响应在 CMD 线上以串行方式传输。

SDIO 支持两种响应类型。两种类型均使用 CRC 错误校验：

- 48 位短响应
- 136 位长响应

注意： *如果响应不包含 CRC（CMD1 响应），则设备驱动程序必须忽略 CRC 失败状态。*

表 130. 短响应格式

位的位置	宽度	值	说明
47	1	0	起始位
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 131. 长响应格式

位的位置	宽度	值	说明
135	1	0	起始位
134	1	0	传输位
[133:128]	6	111111	保留
[127:1]	127	-	CID 或 CSD（包括内部 CRC7）
0	1	1	结束位

命令寄存器包含命令索引（发送到卡的六位）和命令类型。它们确定命令是否需要响应以及响应的长度是 48 位还是 136 位（请参见*第 807 页的第 28.9.4 节*）。命令路径实施*表 132* 中显示的状态标志：

表 132. 命令路径状态标志

标志	说明
CMDREND	在响应 CRC 正常的情况下将此标志置 1
CCRCFAIL	在响应 CRC 失败的情况下将此标志置 1
CMDSENT	发送了不需要响应的命令时将此标志置 1
CTIMEOUT	响应超时。
CMDACT	正在进行命令传输。

CRC 生成器计算 CRC 代码前面所有位的 CRC 校验和。这包括起始位、传输位、命令索引和命令参数（或卡状态）。对于长响应格式，将为 CID 或 CSD 的前 120 位计算 CRC 校验和。请注意，CRC 计算中不使用起始位、传输位和六个保留位。

CRC 校验和是一个 7 位值：

CRC[6:0] = 余数 [(M(x) * x^7) / G(x)]

G(x) = $x^7 + x^3 + 1$

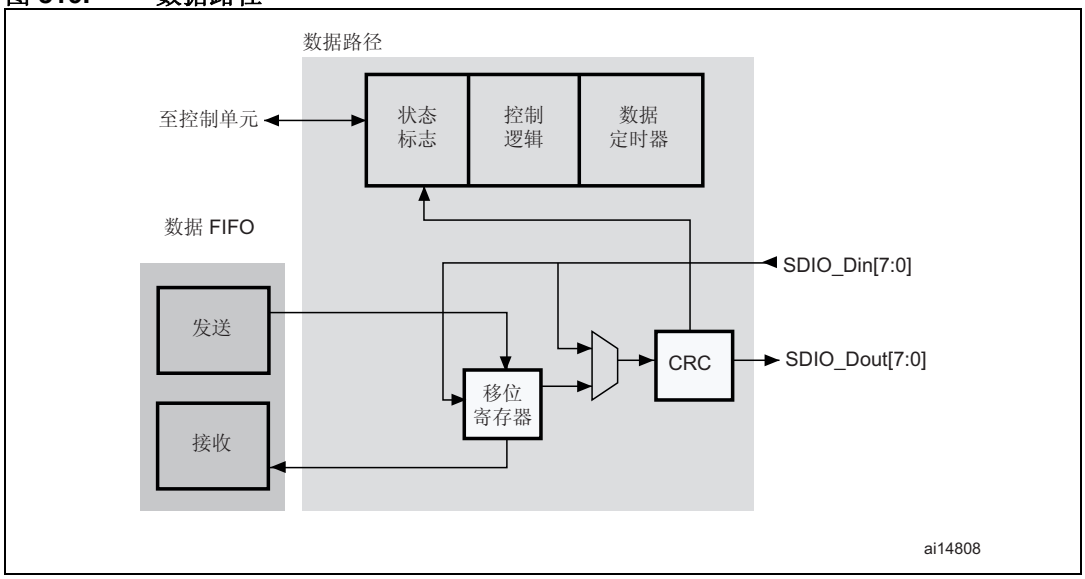
M(x) =（起始位）* x^{39} + ...+（CRC 前的最后一位）* x^0 或者

M(x) =（起始位）* x^{119} + ...+（CRC 前的最后一位）* x^0

数据路径

数据路径子单元负责与卡相互传输数据。*图 319* 显示了数据路径的框图。

图 319. 数据路径



可以使用时钟控制寄存器对卡数据总线宽度进行编程。如果使能了 4 位宽度的总线模式，则使用所有四个数据信号线 (SDIO_D[3:0]) 在每个时钟周期内传输 4 个数据位。如果使能了 8 位宽度的总线模式，则使用所有八个数据信号线 (SDIO_D[7:0]) 在每个时钟周期内传输 8 个数据位。如果未使能宽总线模式，则每个时钟周期仅传输一位，且使用 SDIO_D0。

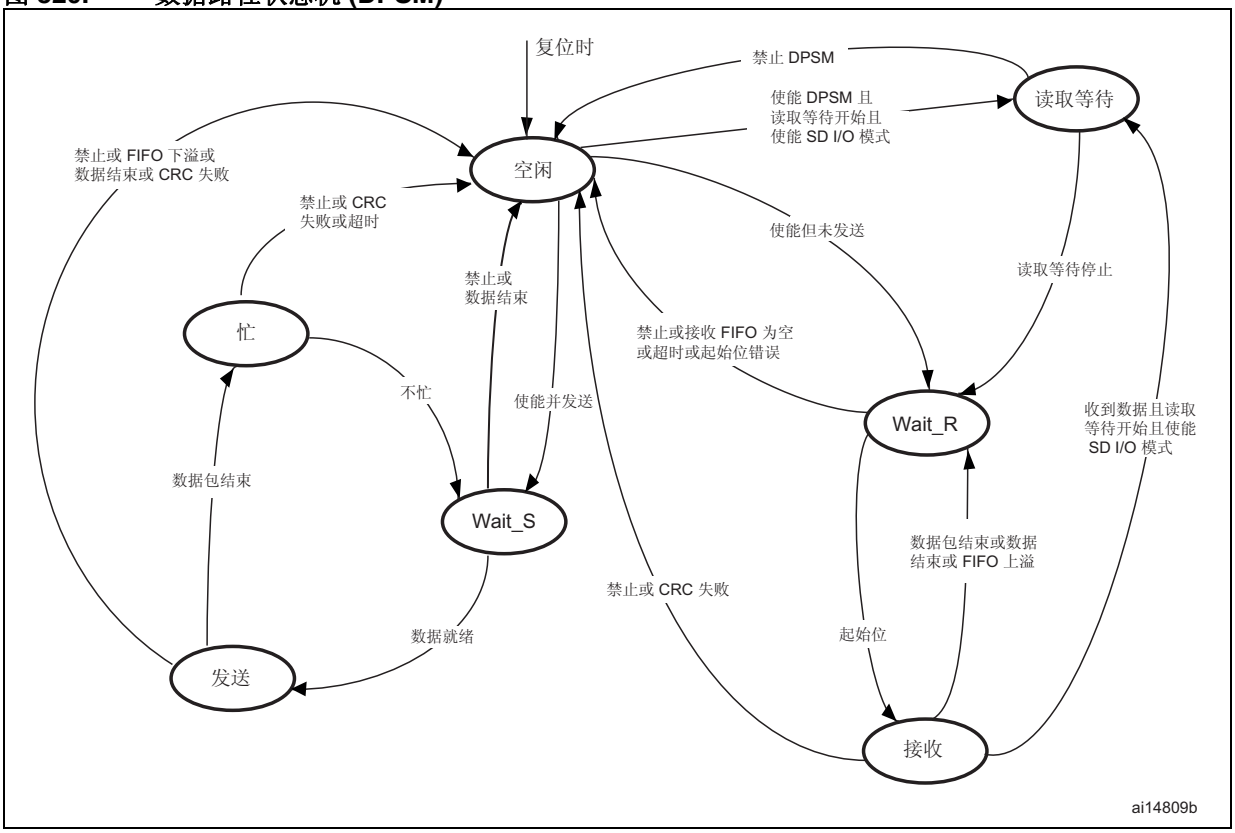
根据传输方向（发送或接收），数据路径状态机 (DPSM) 将变为 Wait_S 或 Wait_R 状态（如果已使能）：

- 发送：DPSM 变为 Wait_S 状态。如果传输 FIFO 中有数据，则 DPSM 变为发送状态并且数据路径子单元开始向卡发送数据。
- 接收：DPSM 变为 Wait_R 状态并等待起始位。在 DPSM 收到起始位时，将变为接收状态，并且数据路径子单元开始从卡中接收数据。

数据路径状态机 (DPSM)

DPSM 以 SDIO_CK 频率运行。卡总线信号上的数据与 SDIO_CK 的上升沿保持同步。DPSM 有六种状态，如*图 320：数据路径状态机 (DPSM)* 中所示。

图 320. 数据路径状态机 (DPSM)



- 空闲：数据路径处于无效状态，SDIO_D[7:0] 输出处于 Hi-Z。当写入数据控制寄存器并将使能位置 1 后，DPSM 将使用新值来加载数据计数器，根据数据方向位，将变为 Wait_S 或 Wait_R 状态。
- Wait_R：如果数据计数器等于零，则 DPSM 会在接收 FIFO 为空时变为空闲状态。如果数据计数器不为零，则 DPSM 等待 SDIO_D 上的起始位。如果 DPSM 在超时之前收到了起始位，则 DPSM 变为接收状态，并加载数据块计数器。如果 DPSM 尚未检测到起始位便已超时或者发生起始位错误，则变为空闲状态并且将超时状态标志置 1。
- 接收：从卡收到的串行数据以字节为单位打包，并写入到数据 FIFO。根据数据控制寄存器中的传输模式，数据传输模式可以是块或流：
  - 在块模式中，当数据块计数器达到零时，DPSM 将等待直至其收到 CRC 代码。如果收到的代码与内部生成的 CRC 代码相匹配，则 DPSM 变为 Wait_R 状态。如果不匹配，则将 CRC 失败状态标志置 1 并且 DPSM 变为空闲状态。
  - 在流模式中，DPSM 在数据计数器不为零时接收数据。当计数器为零时，会将移位寄存器中其余数据写入到数据 FIFO，并且 DPSM 将变为 Wait_R 状态。

  如果发生 FIFO 溢出错误，则 DPSM 将 FIFO 错误标志置 1 并变为空闲状态：
- Wait_S：如果数据计数器为零，则 DPSM 变为空闲状态。如果不为零，则 DPSM 将等待直至停止发出 FIFO 空标志，并变为发送状态。

注意： *DPSM 在 Wait_S 状态下需要至少保持两个时钟周期以满足 N_{WR} 计时要求，N_{WR} 是从收到卡响应到开始从主机传输数据这一期间的时钟周期数。*

- 发送：DPSM 开始向卡发送数据。根据数据控制寄存器中的传输模式，数据传输模式可以是块或流：
  - 在块模式中，当数据块计数器达到零时，DPSM 将发送一个内部生成的 CRC 代码和结束位，并变为繁忙状态。
  - 在流模式中，DPSM 将在使能位处于高电平并且数据计数器不为零时，向卡发送数据。然后它会变为空闲状态。

  如果发生 FIFO 下溢错误，则 DPSM 将 FIFO 错误标志置 1 并变为空闲状态。
- 忙碌：DPSM 等待 CRC 状态标志：
  - 如果未收到正确的 CRC 状态，则变为空闲状态并且将 CRC 失败状态标志置 1。
  - 如果收到正确的 CRC 状态，则在 SDIO_D0 未处于低电平（卡未处于繁忙状态）的情况下变为 Wait_S 状态。

  如果 DPSM 处于繁忙状态时发生超时，则数据超时标志置 1 并变为空闲状态。

  当 DPSM 处于 Wait_R 或繁忙状态时将使能数据计时器，并生成数据超时错误：
  - 发送数据时，如果 DPSM 处于繁忙状态的时间超过了编程的超时周期，则发生超时错误
  - 接收数据时，如果数据的结尾不为“真”，并且 DPSM 处于 Wait_R 状态的时间超过了编程的超时周期，则发生超时。
- **数据**：可以将数据从卡发送到主机，或从主机到卡。数据通过数据线来传输。数据存储在 32 字的 FIFO 中，每个字为 32 位宽。

表 133. 数据令牌格式

说明	起始位	数据	CRC16	结束位
块数据	0	-	是	1
流数据	0	-	否	1

数据 FIFO

数据 FIFO（先进先出）子单元是一个数据缓冲器，带发送和接收单元。

FIFO 包含一个宽度为 32 位且深度为 32 字的数据缓冲器和发送/接收逻辑。由于数据 FIFO 在 APB2 时钟域 (PCLK2) 中运行，因此来自 SDIO 时钟域 (SDIOCLK) 的子单元中的所有信号都将重新同步。

根据 TXACT 和 RXACT 标志，可以将 FIFO 禁止、使能传输或使能接收。TXACT 和 RXACT 是由数据路径子单元驱动，并且两者互斥：

- 发出 TXACT 时，传输 FIFO 引用传输逻辑和数据缓冲器
- 发出 RXACT 时，接收 FIFO 引用接收逻辑和数据缓冲器

- 传输 FIFO：

  如果使能了 SDIO 以用于传输，那么可以通过 APB2 接口将数据写入到传输 FIFO。

  可以通过 32 个连续地址来访问传输 FIFO。传输 FIFO 包含一个数据输出寄存器，其中存储了读取指针所指向的数据字。在数据路径子单元已加载了其移位寄存器之后，该子单元会将读取指针递增并将新数据推出。

  如果禁止了传输 FIFO，则停止发出所有状态标志。数据路径子单元在传输数据时置位 TXACT。

表 134. 传输 FIFO 状态标志

标志	说明
TXFIFOF	当所有 32 个传输 FIFO 字都包含有效数据时，设置为高电平。
TXFIFOE	当传输 FIFO 不包含有效数据时，设置为高电平。
TXFIFOHE	当 8 个或更多个传输 FIFO 字为空时，设置为高电平。该标志可以用作 DMA 请求。
TXDAVL	当传输 FIFO 包含有效数据时，设置为高电平。该标志与 TXFIFOE 标志正好相反。
TXUNDERR	发生下溢错误时，设置为高电平。通过写入到 SDIO 清零寄存器可将该位清零。

- 接收 FIFO

  当数据路径子单元接收到一个字的数据时，该子单元将在写入数据总线上驱动数据。在写入操作完成后，写入指针递增。在读取端，读取指针的当前值指向的 FIFO 字内容被推入读取数据总线中。如果禁止了接收 FIFO，则会停止发出所有状态标志，并且将读取和写入指针复位。数据路径子单元在接收数据时置位 RXACT。*表 135* 列出了接收 FIFO 状态标志。可以通过 32 个连续地址来访问接收 FIFO。

表 135. 接收 FIFO 状态标志

标志	说明
RXFIFOF	当所有 32 个接收 FIFO 字都包含有效数据时，设置为高电平
RXFIFOE	当接收 FIFO 不包含有效数据时，设置为高电平。
RXFIFOHF	当 8 个或更多个接收 FIFO 字包含有效数据时，设置为高电平。该标志可以用作 DMA 请求。
RXDAVL	当接收 FIFO 非空时，设置为高电平。该标志与 RXFIFOE 标志正好相反。
RXOVERR	发生溢出错误时，设置为高电平。通过写入到 SDIO 清零寄存器可将该位清零。

28.3.2 SDIO APB2 接口

APB2 接口生成中断和 DMA 请求，并且访问 SDIO 适配器寄存器和数据 FIFO。该接口由数据路径、寄存器解码器和中断/DMA 逻辑构成。

SDIO 中断

中断逻辑生成一个中断请求信号，当至少其中一个所选状态标志为高电平时，便发出此信号。提供了一个屏蔽寄存器，用于选择将生成中断的条件。如果对应的屏蔽标志置 1，则状态标志将生成中断请求。

SDIO/DMA 接口 - SDIO 和存储器之间的数据传输过程

在显示的示例中，是使用 CMD24 (WRITE_BLOCK)，从 SDIO 主机控制器传输到 MMC（512 字节）。使用 DMA 控制器将存储器中的数据填充到 SDIO FIFO。

1. 执行卡识别过程
2. 增加 SDIO_CK 频率
3. 发送 CMD7 来选择卡
4. 配置 DMA2，如下所示：
   a) 使能 DMA2 控制器并将所有挂起的中断清零。
   b) 使用存储器位置的基址对 DMA2_Stream3 或 DMA2_Stream6 Channel4 源地址寄存器进行编程，并使用 SDIO_FIFO 寄存器地址对 DMA2_Stream3 或 DMA2_Stream6 Channel4 目标地址寄存器进行编程。
   c) 对 DMA2_Stream3 或 DMA2_Stream6 Channel4 控制寄存器进行编程（存储器递增、非外设递增，外设和源宽度是字）。
   d) 对 DMA2_Stream3 或 DMA2_Stream6 Channel4 进行编程以选择外设作为流控制器（在 DMA_S3CR 或 DMA_S6CR 配置寄存器中将 PFCTRL 位置 1）。
   e) 在 DMA2_Stream3 或 DMA2_Stream6 Channel4 中，将递增突发传输配置为 4 个节拍（至少来自于外设端）。
   f) 使能 DMA2_Stream3 或 DMA2_Stream6 Channel4
5. 发送 CMD24 (WRITE_BLOCK)，如下所示：
   a) 对 SDIO 数据长度寄存器进行编程（在卡识别过程之前，SDIO 数据计时器寄存器应该已编程）。
   b) 使用数据传输目标卡的地址位置对 SDIO 参数寄存器进行编程。
   c) 对 SDIO 命令寄存器进行编程：CmdIndex 为 24 (WRITE_BLOCK)；WaitResp 为“1”（SDIO 卡主机等待响应）；CPSMEN 为“1”（使能 SDIO 卡主机，以发送命令）。其它字段是其各自的复位值。
   d) 等待 SDIO_STA[6] = CMDREND 中断，然后对 SDIO 数据控制寄存器进行编程：DTEN 为“1”（使能 SDIO 卡主机以发送数据）；DTDIR 为“0”（从控制器到卡）；DTMODE 为“0”（块数据传输）；DMAEN 为“1”（使能 DMA）；DBLOCKSIZE 为 0x9（512 字节）。其它字段无关。
   e) 等待 SDIO_STA[10] = DBCKEND。
6. 轮询 DMA 通道使能状态寄存器，以确认没有任何通道仍处于使能状态。

28.4 卡功能说明

28.4.1 卡识别模式

在卡识别模式下，主机复位所有卡，验证运行电压范围，识别卡并在总线上为每个卡设置相对卡地址 (RCA)。卡识别模式中的所有数据通信都仅使用命令行 (CMD)。

28.4.2 智能卡复位

GO_IDLE_STATE 命令 (CMD0) 是软件复位命令，它会将多媒体卡和 SD 存储器置于空闲状态。IO_RW_DIRECT 命令 (CMD52) 可复位 SD I/O 卡。在上电或 CMD0 之后，所有卡输出总线驱动程序都处于高阻态状态，并且这些卡将使用默认的相对卡地址 (RCA=0x0001) 和默认的驱动程序阶段寄存器设置（最低速度，最高驱动电流容量）进行初始化。

28.4.3 工作电压范围验证

所有卡都可以使用规范范围内的任何工作电压与 SDIO 卡主机进行通信。卡上的运行条件寄存器 (OCR) 定义了支持的最小和最大 V_{DD} 值。

在有效负载存储器中存储卡标识号 (CID) 和卡特定数据 (CSD) 的卡仅能够在数据传输 V_{DD} 条件下传播此信息。当 SDIO 卡主机模块和卡具有不兼容的 V_{DD} 范围时，卡无法完成识别周期且无法发送 CSD 数据。因此，SEND_OP_COND (CMD1)、SD_APP_OP_COND（用于 SD 存储器的 ACMD41）和 IO_SEND_OP_COND（用于 SD I/O 的 CMD5）等特殊命令旨在提供一种机制，用于识别和拒绝与 SDIO 卡主机要求的 V_{DD} 范围不匹配的卡。SDIO 卡主机会发送所需的 V_{DD} 电压窗口作为这些命令的运算对象。无法在指定范围内执行数据传输的卡将断开与总线的连接，并且变为无效状态。

如果在使用这些命令时不将电压范围作为运算对象包括在内，则 SDIO 卡主机可能会查询每个卡并确定通用电压范围，然后再将超出范围的卡置于无效状态。当 SDIO 卡主机能够选择通用电压范围或者当用户要求就卡无法使用这一情况进行通知时，将会使用此查询。

28.4.4 卡识别过程

多媒体卡和 SD 卡的卡识别过程有所不同。对于多媒体卡，识别过程是以时钟速率 F_{od} 开始。SDIO_CMD 线输出驱动器是开漏引脚，在此识别过程中允许并行的卡操作。对准过程以如下方式完成：

1. 激活总线。
2. SDIO 卡主机广播 SEND_OP_COND (CMD1) 以接收运行条件。
3. 响应是来自所有卡的运行条件寄存器的线与运算。
4. 不兼容的卡将被置于无效状态。
5. SDIO 卡主机向所有有效卡广播 ALL_SEND_CID (CMD2)。
6. 有效卡同时以串行方式发送其 CID 号。如果卡的输出 CID 位与命令线上的位不匹配，则卡将停止传输，必须等待下一个识别周期。一个卡成功地将完整 CID 传输到 SDIO 卡主机并进入识别状态。
7. SDIO 卡主机向该卡发出 SET_RELATIVE_ADDR (CMD3)。这一新地址称为相对卡地址 (RCA)；它比 CID 更短，可对卡进行寻址。分配的卡变为待机状态，它不会对进一步的识别周期进行响应，并且其输出将从开漏切换为推挽。
8. SDIO 卡主机不断重复步骤 5 到 7，直到收到超时条件为止。

对于 SD 卡，识别过程是以时钟速率 F_{od} 开始，并且 SDIO_CMD 线输出驱动是推挽驱动器，而非开漏引脚。对准过程以如下方式完成：

1. 激活总线。
2. SDIO 卡主机广播 SD_APP_OP_COND (ACMD41)。
3. 卡以其运行条件寄存器的内容进行响应。
4. 不兼容的卡将被置于无效状态。
5. SDIO 卡主机向所有有效卡广播 ALL_SEND_CID (CMD2)。
6. 这些卡将发回其唯一的卡识别号 (CID) 并进入识别状态。
7. SDIO 卡主机向某个地址的有效卡发出 SET_RELATIVE_ADDR (CMD3)。这一新地址称为相对卡地址 (RCA)；它比 CID 更短，可对卡进行寻址。分配的卡变为待机状态。SDIO 卡主机可以重新发出此命令以更改 RCA。卡的 RCA 是最后分配的值。
8. SDIO 卡主机对所有有效卡重复步骤 5 到 7。

对于 SD I/O 卡，对准过程以如下方式完成：

1. 激活总线。
2. SDIO 卡主机发送 IO_SEND_OP_COND (CMD5)。
3. 卡以其运行条件寄存器的内容进行响应。
4. 不兼容的卡将被设置无效状态。
5. SDIO 卡主机向某个地址的有效卡发出 SET_RELATIVE_ADDR (CMD3)。这一新地址称为相对卡地址 (RCA)；它比 CID 更短，可对卡进行寻址。分配的卡变为待机状态。SDIO 卡主机可以重新发出此命令以更改 RCA。卡的 RCA 是最后分配的值。

28.4.5 块写入

在块写入 (CMD24 - 27) 期间，一个或多个数据块从主机传输到卡，并且由主机在每个块的结尾追加一个 CRC。支持块写入的卡始终可以接受 WRITE_BL_LEN 所定义的数据块。如果 CRC 失败，卡将在 SDIO_D 线上指示失败，已传输的数据将废弃而不会写入，并且将忽略所有进一步传输的块（在多块写入模式中）。

如果主机使用不完整的块（其累计长度没有进行块对齐）并且不允许块偏离（未设置 CSD 参数 WRITE_BLK_MISALIGN），则卡会在第一个偏离块之前检测块偏离错误。（ADDRESS_ERROR 错误位在状态寄存器中设置）。如果主机尝试在写保护区域进行写入，也将会中止写入操作。但是，在此情况下，卡会将 WP_VIOLATION 位置 1。

对 CID 和 CSD 寄存器的编程不需要先前的块长度设置。传输的数据也受到 CRC 保护。如果 CSD 或 CID 寄存器的某一部分存储在 ROM 中，则这一无法更改的部分必须与接收缓冲器的对应部分相匹配。如果不匹配，则卡将报告错误并且不会更改任何寄存器内容。某些卡可能需要很长且无法预测的时间来写入数据块。收到数据块并完成 CRC 校验后，卡开始进行写入，如果卡的写入缓冲器已满并且无法接受来自新 WRITE_BLOCK 命令的新数据，则会保持 SDIO_D 线的低电平。主机可能随时使用 SEND_STATUS 命令 (CMD13) 轮询卡状态，而卡将使用其状态进行响应。READY_FOR_DATA 状态位指示卡是否可以接受新数据以及写入进程是否仍在进行中。主机可以通过发出 CMD7（用于选择不同的卡）来取消选择卡，这会将卡置于断开连接状态并释放 SDIO_D 线，但不会中断写入操作。再次选择卡时，如果编程仍在进行中并且写入缓冲器不可用，则卡会通过将 SDIO_D 拉到低电平来重新激活忙碌指示。

28.4.6 块读取

在块读取模式中，数据传输的基本单位是块，其最大大小在 CSD (READ_BL_LEN) 中定义。如果将 READ_BL_PARTIAL 置 1，则还可以传输更小的块，这些块的开始和结束地址全部包含在一个物理块中（如 READ_BL_LEN 所定义）。在每个块的结尾会追加一个 CRC，以确保数据传输完整性。CMD17 (READ_SINGLE_BLOCK) 会启动块读取，在完成传输后，卡将返回到传输状态。

CMD18 (READ_MULTIPLE_BLOCK) 会启动多个连续块的传输。

主机在多个块操作中可以随时中止读取，无论其类型如何。通过发送停止传输命令可中止传输。

在多个块读取操作期间（全部两种类型），如果卡检测到错误（例如，超出范围、地址未对齐或内部错误），卡将停止数据传输并保持处于数据状态。然后，主机必须通过发送停止传输命令来中止操作。读取错误将在停止传输命令的响应中报告。

如果主机在卡以预定义的块数量传输了多个块操作的最后一个块之后发送了停止传输命令，则会以非法命令来响应它，因为卡已经不再处于数据状态。如果主机使用不完整的块（其累计长度没有进行块对齐）并且不允许块偏离，则卡会在第一个偏离块的开头检测块偏离错误条件（在状态寄存器中 ADDRESS_ERROR 错误位置 1）。

28.4.7 流访问、流写入和流读取（仅限多媒体卡）

在流模式中，数据以字节进行传输，并且不会在每个块的结尾追加 CRC。

流写入（仅限多媒体卡）

WRITE_DAT_UNTIL_STOP (CMD20) 会启动从 SDIO 卡主机到卡的数据传输，从指定地址开始并继续，直到 SDIO 卡主机发出停止命令为止。如果允许不完整的块（CSD 参数 WRITE_BL_PARTIAL 置 1），则数据库可以在卡地址空间内的任何地址开始和停止，否则只能在时钟边界处开始和停止。由于未提前确定要传输的数据量，因此无法使用 CRC。如果在发送数据时达到了存储器范围的结尾并且 SD 卡主机未发出停止命令，则将丢弃已传输的任何多余的数据。

使用以下特定于卡的数据寄存器等式字段，可确定流写入操作的最大时钟频率：

$$\text{Maximumspeed} = \text{MIN}(\text{TRANSPEED}, \frac{(8 \times 2^{\text{writebllen}})(-\text{NSAC})}{\text{TAAC} \times \text{R2WFACTOR}})$$

- Maximumspeed = 最大写入频率
- TRANSPEED = 最大数据传输速率
- writebllen = 写入数据块最大长度
- NSAC = CLK 周期中数据读取访问时间 2
- TAAC = 数据读取访问时间 1
- R2WFACTOR = 写入速度系数

如果主机尝试使用更高的频率，卡可能无法处理数据并停止编程，在状态寄存器中将 OVERRUN 错误位置 1，同时忽略所有进一步数据传输，并在接收数据状态下等待停止命令。如果主机尝试在写保护区域进行写入，也将会中止写入操作。但是，在此情况下，卡会将 WP_VIOLATION 位置 1。

流读取（仅限多媒体卡）

READ_DAT_UNTIL_STOP (CMD11) 控制面向流的数据传输。

此命令指示卡从指定地址开始发送其数据，直到 SDIO 卡主机发送 STOP_TRANSMISSION (CMD12) 为止。由于命令是串行传输，因此停止命令在执行时会有延迟，数据传输会在停止命令的结束位之后停止。如果在发送数据时达到了存储器范围的结尾并且 SDIO 卡主机未发出停止命令，则发送的任何后续数据都被视为未定义。

使用以下特定于卡的数据寄存器等式字段，可确定流读取操作的最大时钟频率。

$$\text{Maximumspeed} = \text{MIN}(\text{TRANSPEED}, \frac{(8 \times 2^{\text{readbllen}})(-\text{NSAC})}{\text{TAAC} \times \text{R2WFACTOR}})$$

- Maximumspeed = 最大读取频率
- TRANSPEED = 最大数据传输速率
- readbllen = 读取数据块最大长度
- writebllen = 写入数据块最大长度
- NSAC = CLK 周期中数据读取访问时间 2
- TAAC = 数据读取访问时间 1
- R2WFACTOR = 写入速度系数

如果主机尝试使用更高的频率，则卡无法持续进行数据传输。如果发生此情况，卡会在状态寄存器中将 UNDERRUN 错误位置 1，中止传输并在数据状态下等待停止命令。

28.4.8 擦除：组擦除和扇区擦除

多媒体卡的可擦除单元是擦除组。擦除组以写入块来测量，这些写入块是卡的基本可写入单元。擦除组的大小是特定于卡的参数，在 CSD 中定义。

主机可以擦除某一连续范围内的擦除组。启动擦除是一个三步的过程。

首先，主机使用 ERASE_GROUP_START (CMD35) 命令定义范围的起始地址，然后使用 ERASE_GROUP_END (CMD36) 命令定义范围的最终地址，最后通过发出 ERASE (CMD38) 命令来启动擦除过程。擦除命令中的地址字段是以字节为单位的擦除组地址。卡将忽略小于擦除组大小的所有 LSB，从而有效地将地址向下取整为擦除组边界。

如果未按顺序接收擦除命令，则卡会在状态寄存器中将 ERASE_SEQ_ERROR 位置 1 并复位整个序列。

如果收到了一个乱序命令（两个都不是擦除命令，但 SEND_STATUS 除外），则卡会在状态寄存器中将 ERASE_RESET 状态位置 1、复位擦除序列并执行最后的命令。

如果擦除范围包括写保护块，则它们将保留不变，仅会擦除不受保护的块。状态寄存器中的 WP_ERASE_SKIP 状态位将会置 1。

卡通过将 SDIO_D 保持低电平来指示擦除正在进行中。实际擦除时间可能很长，主机可以发出 CMD7 来取消选择卡。

28.4.9 宽总线选择或取消选择

使用 SET_BUS_WIDTH (ACMD6) 来选择或取消选择宽总线（4 位总线宽度）操作模式。加电或 GO_IDLE_STATE (CMD0) 后，默认总线宽度是 1 位。SET_BUS_WIDTH (ACMD6) 仅在传输状态下有效，这意味着只有在通过 SELECT/DESELECT_CARD (CMD7) 选择了卡之后才能更改总线宽度。

28.4.10 保护管理

SDIO 卡主机模块中支持三种针对卡的写保护方法。

1. 卡内部写保护（卡的职责）
2. 机械写保护开关（仅由 SDIO 卡主机模块负责）
3. 密码保护的卡锁定操作

卡内部写保护

可以防止对卡数据进行写入和擦除。通过在 CSD 中设置永久或临时写保护位，制造商或内容供应商可以将整个卡永久进行写保护。如果卡支持对扇区组进行写保护（通过在 CSD 中将 WP_GRP_ENABLE 位置 1），可以保护部分数据，并且写保护可以由应用程序进行更改。写保护是以 CSD 中指定的 WP_GRP_SIZE 扇区为单位。SET_WRITE_PROT 和 CLR_WRITE_PROT 命令控制编址组的保护。SEND_WRITE_PROT 命令类似于单个块读取命令。卡将发送一个数据块，其中包含 32 个写保护位（表示从指定地址开始的 32 个写保护组），后跟 16 个 CRC 位。写保护命令中的地址字段是以字节为单位的组地址。

卡将忽略小于组大小的所有 LSB。

机械写保护开关

一个位于卡侧面的机械拨片，用户可以利用它来设置或清除卡上的写保护。当拨片位于窗口打开的位置时，卡受到写保护，当窗口关闭时，可以更改卡内容。插座一侧的匹配开关向 SDIO 卡主机模块表明，卡受到写保护。SDIO 卡主机模块负责保护卡。卡的内部电路并不知道写保护开关的位置。

密码保护

密码保护功能使 SDIO 卡主机模块可以使用密码来锁定和解锁卡。密码存储在 128 位 PWD 寄存器中，其大小在 8 位 PWD_LEN 寄存器中设置。这些寄存器具有非易失性，因此断电不会擦除它们。锁定的卡将响应和执行某些命令。这意味着 SDIO 卡主机模块可以进行复位、初始化、选择以及查询状态，但是不允许访问卡的数据。设置了密码后（由 PWD_LEN 的非零值来指示），卡将在上电后自动锁定。与 CSD 和 CID 寄存器的写命令一样，仅在传输状态中才可以使用锁定/解锁命令。在此情况下，该命令不包括地址参数，必须首先选择卡然后才能使用。卡锁定/解锁命令具备常规单一块写入命令的结构和总线事务类型。传输的数据块包括命令的所有必需信息（密码设置模式、PWD 本身以及卡锁定/解锁）。命令数据块大小由 SDIO 卡主机模块在发送卡锁定/解锁命令之前定义，命令数据块的结构如*表 149* 中所示。

位设置如下所示：

- ERASE：设置后会强制执行擦除操作。所有其它位都必须为零，并且仅发送命令字节
- LOCK_UNLOCK：设置后会锁定卡。LOCK_UNLOCK 可以与 SET_PWD 一起设置，但不能与 CLR_PWD 一起设置
- CLR_PWD：设置后会清除密码数据
- SET_PWD：设置后会将密码数据保存到存储器
- PWD_LEN：它定义密码的长度（以字节为单位）
- PWD：密码（新的或当前使用的，取决于命令）

以下各节列出了有关设置/复位密码、锁定/解锁卡以及强制擦除的命令。

设置密码

1. 如果未选择任何卡，请选择一个 (SELECT/DESELECT_CARD, CMD7)。
2. 定义要发送的块长度 (SET_BLOCKLEN, CMD16)（由 8 位卡锁定/解锁模式来确定）、8 位 PWD_LEN 以及新密码的字节数。完成了命令替换后，块大小必须考虑到如下情况：无论是旧密码还是新密码都会与命令一起发送。
3. 在数据线上，与相应的数据块大小一起发送 LOCK/UNLOCK (CMD42)，包括 16 位 CRC。数据块指示模式 (SET_PWD = 1)、长度 (PWD_LEN) 和密码自身 (PWD)。完成密码替换后，长度值 (PWD_LEN) 会包括新旧两个密码的长度，PWD 字段包括旧密码（当前使用的）后跟新密码。
4. 如果密码匹配，则新密码及其大小会分别保存到 PWD 和 PWD_LEN 字段。如果发送的旧密码与预期的密码不符（大小和/或内容），会在卡状态寄存器中将 LOCK_UNLOCK_FAILED 错误位置 1，而密码不会更改。

密码长度字段 (PWD_LEN) 指示当前是否设置了密码。如果此字段非零，则表示设置了密码并且卡在上电后锁定自身。通过将 LOCK_UNLOCK 位置 1（在设置密码时）或通过发送用于卡锁定的其它命令，可以在当前上电会话中立即锁定卡。

复位密码

1. 如果未选择任何卡，请选择一个 (SELECT/DESELECT_CARD, CMD7)。
2. 定义要发送的块长度 (SET_BLOCKLEN, CMD16)（由 8 位卡锁定/解锁模式来确定）、8 位 PWD_LEN 以及当前使用的密码的字节数。
3. 在数据线上，与相应的数据块大小一起发送 LOCK/UNLOCK (CMD42)，包括 16 位 CRC。数据块指示模式 (CLR_PWD = 1)、长度 (PWD_LEN) 和密码自身 (PWD)。将忽略 LOCK_UNLOCK 位。
4. 如果密码匹配，会清除 PWD 字段并将 PWD_LEN 置 0。如果发送的密码与预期的密码不符（大小和/或内容），会在卡状态寄存器中将 LOCK_UNLOCK_FAILED 错误位置 1，而密码不会更改。

锁定卡

1. 如果未选择任何卡，请选择一个 (SELECT/DESELECT_CARD, CMD7)。
2. 定义要发送的块长度 (SET_BLOCKLEN, CMD16)（由 8 位卡锁定/解锁模式来确定，请参见*表 149* 中的字节 0）、8 位 PWD_LEN 以及当前密码的字节数。
3. 在数据线上，与相应的数据块大小一起发送 LOCK/UNLOCK (CMD42)，包括 16 位 CRC。数据块指示模式 (LOCK_UNLOCK = 1)、长度 (PWD_LEN) 和密码自身 (PWD)。
4. 如果密码匹配，则卡被锁定并且卡状态寄存器中 CARD_IS_LOCKED 状态位会置 1。如果发送的密码与预期的密码不符（大小和/或内容），会在卡状态寄存器中将 LOCK_UNLOCK_FAILED 错误位置 1，且锁定会失败。

可以按照相同的顺序来设置密码和锁定卡。在此情况下，SDIO 卡主机模块会执行用于设置密码的所有必需步骤（请参见*第 788 页的设置密码*），但是发送新密码命令时，需要在步骤 3 中将 LOCK_UNLOCK 位置 1。

如果之前设置过密码（PWD_LEN 不为 0），则卡在上电复位后会自动锁定。尝试锁定已锁定的卡或者尝试锁定没有密码的卡都会失败，并且会在卡状态寄存器中将 LOCK_UNLOCK_FAILED 错误位置 1。

解锁卡

1. 如果未选择任何卡，请选择一个 (SELECT/DESELECT_CARD, CMD7)。
2. 定义要发送的块长度 (SET_BLOCKLEN, CMD16)（由 8 位卡锁定/解锁模式来确定，请参见*表 149* 中的字节 0）、8 位 PWD_LEN 以及当前密码的字节数。
3. 在数据线上，与相应的数据块大小一起发送 LOCK/UNLOCK (CMD42)，包括 16 位 CRC。数据块指示模式 (LOCK_UNLOCK = 0)、长度 (PWD_LEN) 和密码自身 (PWD)。
4. 如果密码匹配，则卡被解锁并且卡状态寄存器中 CARD_IS_LOCKED 状态位会清零。如果发送的密码在大小和/或内容方面不正确并且与预期的密码不符，会在卡状态寄存器中将 LOCK_UNLOCK_FAILED 错误位置 1，且卡保持锁定。

解锁功能仅对当前上电会话有效。如果 PWD 字段未清零，则卡会在下一次上电时自动锁定。

尝试解锁已解锁的卡会导致失败，并且会在卡状态寄存器中将 LOCK_UNLOCK_FAILED 错误位置 1。

强制擦除

如果用户忘记了密码（PWD 内容），可以在清除卡上所有数据之后访问卡。这种强制擦除操作将擦除所有卡数据和所有密码数据。

1. 如果未选择任何卡，请选择一个 (SELECT/DESELECT_CARD, CMD7)。
2. 将块长度 (SET_BLOCKLEN, CMD16) 设置为 1 个字节。仅发送 8 位卡锁定/解锁字节（*表 149* 中的字节 0）。
3. 在数据线上，与相应的数据字节一起发送 LOCK/UNLOCK (CMD42)，包括 16 位 CRC。数据块指示模式 (ERASE = 1)。所有其它位都必须为零。
4. 如果数据字段中仅设置了 ERASE 位，会擦除所有卡内容，包括 PWD 和 PWD_LEN 字段，并且卡不再锁定。如果设置了任何其它位，则会在卡状态寄存器中将 LOCK_UNLOCK_FAILED 错误位置 1，而卡会保留其所有数据并保持锁定。

尝试在锁定的卡上使用强制擦除会导致失败，并且会在卡状态寄存器中将 LOCK_UNLOCK_FAILED 错误位置 1。

28.4.11 卡状态寄存器

响应格式 R1 包含一个由 32 位字段命名的卡状态。此字段旨在将卡状态信息（可能存储在本地状态寄存器中）传输到主机。除非另有说明，否则状态条目始终与先前发出的命令有关。

表 136 定义了状态的不同条目。表中的类型和清除条件字段以如下方式缩写：

类型：

- E：错误位
- S：状态位
- R：已检测到并已针对实际命令响应进行了设置
- X：已检测到并在命令执行期间进行了设置。SDIO 卡主机要对卡进行轮询必须发出状态命令来读取这些位。

清除条件：

- A：根据卡的当前状态
- B：始终与先前命令有关。收到有效命令后会将其清零（延迟一个命令）
- C：通过读取清零

表 136. 卡状态

位	标识符	类型	值	说明	清除条件
31	ADDRESS_ OUT_OF_RANGE	E R X	“0” = 无错误 “1” = 有错误	命令地址参数超出了此卡允许的范围。 多个块或流读取/写入操作正在尝试进行超出卡容量的读取或写入（尽管是从有效地址开始）。	C
30	ADDRESS_MISALIGN		“0” = 无错误 “1” = 有错误	命令地址参数（与当前设置的块长度一致）会将第一个偏离的数据块定位到卡的物理块。 多个块读取/写入操作正在尝试读取或写入与卡的物理块未对齐的数据块（尽管是以有效的地址/块长度组合开始）。	C
29	BLOCK_LEN_ERROR		“0” = 无错误 “1” = 有错误	可能是 SET_BLOCKLEN 命令的参数超过了卡的最大允许值，或者先前定义的块长度对当前命令而言非法（例如，主机发出了写入命令，而当前块长度小于卡的最大允许值并且不允许写入不完整的块）。	C
28	ERASE_SEQ_ERROR		“0” = 无错误 “1” = 有错误	擦除命令序列中发生错误。	C
27	ERASE_PARAM	E X	“0” = 无错误 “1” = 有错误	为擦除选择的擦除组无效。	C
26	WP_VIOLATION	E X	“0” = 无错误 “1” = 有错误	尝试对写保护块进行编程。	C
25	CARD_IS_LOCKED	S R	“0” = 表示卡已解锁 “1” = 卡已锁定	如果设置，则表示卡已由主机锁定	A
24	LOCK_UNLOCK_ FAILED	E X	“0” = 无错误 “1” = 有错误	如果在锁定/解锁卡命令中检测到序列或密码错误，则会设置	C
23	COM_CRC_ERROR	E R	“0” = 无错误 “1” = 有错误	先前命令的 CRC 校验失败。	B

表 136. 卡状态（续）

位	标识符	类型	值	说明	清除条件
22	ILLEGAL_COMMAND	E R	“0” = 无错误 “1” = 有错误	命令对于卡状态不合法	B
21	CARD_ECC_FAILED	E X	“0” = 成功 “1” = 失败	应用了卡内部 ECC，但未能更正数据。	C
20	CC_ERROR	E R	“0” = 无错误 “1” = 有错误	（标准中未定义）发生了卡错误，该错误与主机命令无关。	C
19	ERROR	E X	“0” = 无错误 “1” = 有错误	（标准中未定义）与最后一个主机命令的执行（以及在此期间的检测）有关的一般卡错误（例如，读取或写入失败）。	C
18	保留				
17	保留				
16	CID/CSD_OVERWRITE	E X	“0” = 无错误， “1” = 有错误	可以是以下错误之一： – CID 寄存器已写入且无法覆盖 – CSD 的只读部分与卡内容不匹配 – 尝试恢复已设置的副本（设置为原始值）或永久 WP（未受保护）位	C
15	WP_ERASE_SKIP	E X	“0” = 未受保护 “1” = 受保护	**仅当由于现有写入而导致部分地址空间被擦除时才会设置**	C
14	CARD_ECC_DISABLED	S X	“0” = 使能 “1” = 禁用	命令已在未使用内部 ECC 的情况下执行。	A
13	ERASE_RESET		“0” = 已清零 “1” = 已置 1	在执行之前已清除了擦除序列，因为收到的不是擦除序列命令（CMD35、CMD36、CMD38 或 CMD13 之外的命令）	C
12:9	CURRENT_STATE	S R	0 = 空闲 1 = 就绪 2 = 识别 3 = 待机 4 = 传输 5 = 数据 6 = 接收 7 = 进行中 8 = 断开 9 = 突发 10-15 = 保留	接收命令时卡的状态。如果命令执行导致了状态变化，主机可以在下一个命令的响应中看到。这四个位将被解释为 0 到 15 之间的二进制数字。	B
8	READY_FOR_DATA	S R	“0” = 未就绪， “1” = 就绪	对应于总线上的缓冲器空信令	
7	SWITCH_ERROR	E X	“0” = 无错误 “1” = 开关错误	如果设置，则卡不会根据 SWITCH 命令的请求切换到预期模式	B
6	保留				
5	APP_CMD	S R	“0” = 禁用 “1” = 使能	卡将预期收到 ACMD，或者命令已解释为 ACMD 的指示	C
4	保留，用于 SD I/O 卡				
3	AKE_SEQ_ERROR	E R	“0” = 无错误 “1” = 有错误	认证过程序列中出现错误	C

表 136. 卡状态（续）

位	标识符	类型	值	说明	清除条件
2	保留，用于应用程序特定的命令				
1	保留，用于制造商测试模式				
0					

28.4.12 SD 状态寄存器

SD 状态包含与 SD 存储卡专有功能相关的状态位，在将来可用于特定于应用的场合中。SD 状态的大小是一个 512 位的数据块。如果发送了 ACMD13（CMD55，后面带有 CMD13），此寄存器的内容将传输到 SDIO 卡主机。ACMD13 只能发送到处于传输状态的卡（卡已被选定）。

表 137 定义了 SD 状态寄存器的不同条目。表中的类型和清除条件字段以如下方式缩写：

类型：

- E：错误位
- S：状态位
- R：已检测到并已针对实际命令响应进行了设置
- X：已检测到并在命令执行期间进行了设置。SDIO 卡主机要对卡进行轮询必须发出状态命令来读取这些位。

清除条件：

- A：根据卡的当前状态
- B：始终与先前命令有关。收到有效命令后会将其清零（延迟一个命令）
- C：通过读取清零

表 137. SD 状态

位	标识符	类型	值	说明	清除条件
511:510	DAT_BUS_WIDTH	S R	“00” = 1（默认值） “01” = 保留 “10” = 4 位宽度 “11” = 保留	显示了当前定义的数据总线宽度（由 SET_BUS_WIDTH 命令定义）	A
509	SECURED_MODE	S R	“0” = 未处于安全模式 “1” = 处于安全模式	卡处于安全操作模式（请参见“SD 安全规范”）。	A
508:496	保留				
495:480	SD_CARD_TYPE	S R	“00xxh” = SD 存储卡，按照物理规范版本 1.01-2.00 中所定义（“x” = 无关）。目前定义了以下卡： “0000” = 常规 SD RD/WR 卡。 “0001” = SD ROM 卡	将来，8 个 LSB 将用于定义 SD 存储卡的不同变体（每个位将定义不同的 SD 类型）。8 个 MSB 将用于定义不符合当前 SD 物理层规范的 SD 卡。	A
479:448	SIZE_OF_PROTECT ED_AREA	S R	受保护区域的大小（请参见以下内容）	（请参见以下内容）	A
447:440	SPEED_CLASS	S R	卡的速度等级（请参见以下内容）	（请参见以下内容）	A

表 137. SD 状态（续）

位	标识符	类型	值	说明	清除条件
439:432	PERFORMANCE_MOVE	S R	按 1 [MB/s] 步长表示的移动性能。（请参见以下内容）	（请参见以下内容）	A
431:428	AU_SIZE	S R	AU 大小（请参见以下内容）	（请参见以下内容）	A
427:424	保留				
423:408	ERASE_SIZE	S R	一次要擦除的 AU 数	（请参见以下内容）	A
407:402	ERASE_TIMEOUT	S R	擦除 UNIT_OF_ERASE_AU 指定的区域时的超时值	（请参见以下内容）	A
401:400	ERASE_OFFSET	S R	为擦除时间增加的固定偏移值。	（请参见以下内容）	A
399:312	保留				
311:0	保留，用于制造商				

SIZE_OF_PROTECTED_AREA

此字段的设置在标准容量卡和高容量卡之间有所不同。如果是标准容量卡，受保护区域的容量计算方式如下：

受保护区域 = SIZE_OF_PROTECTED_AREA_* MULT * BLOCK_LEN。

SIZE_OF_PROTECTED_AREA 是以 MULT*BLOCK_LEN 为单位来指定。

如果是高容量卡，受保护区域的容量在以下字段中指定：

受保护区域 = SIZE_OF_PROTECTED_AREA

SIZE_OF_PROTECTED_AREA 是以字节为单位来指定。

SPEED_CLASS

此 8 位字段指示速度等级，该值可以通过 $P_W/2$ 来计算（其中 P_W 是写入性能）。

表 138. 速度等级代码字段

SPEED_CLASS	值定义
00h	等级 0
01h	等级 2
02h	等级 4
03h	等级 6
04h – FFh	保留

PERFORMANCE_MOVE

此 8 位字段指示 Pm（移动性能），该值可通过 1 [MB/s] 步长进行设置。如果卡不移动已使用的 RU（记录单元），则 Pm 应被视为无穷大。将该字段设置为 FFh 即表示无穷大。

表 139. 移动性能字段

PERFORMANCE_MOVE	值定义
00h	未定义
01h	1 [MB/s]
02h	02h 2 [MB/s]
---------	---------
FEh	254 [MB/s]
FFh	无穷大

AU_SIZE

该 4 位域指示 AU 大小，其取值范围是 2 的 n 次方（起始值为 16KB）。

表 140. AU_SIZE 字段

AU_SIZE	值定义
00h	未定义
01h	16 KB
02h	32 KB
03h	64 KB
04h	128 KB
05h	256 KB
06h	512 KB
07h	1 MB
08h	2 MB
09h	4 MB
Ah – Fh	保留

AU 的最大大小，具体取决于卡的容量（在*表 141* 中进行了定义）。可将卡的容量大小设置为介于 RU 大小和最大 AU 大小之间的任何 AU 大小。

表 141. 最大 AU 大小

容量	16 MB-64 MB	128 MB-256 MB	512 MB	1 GB-32 GB
最大 AU 大小	512 KB	1 MB	2 MB	4 MB

ERASE_SIZE

该 16 位域指示 NERASE。擦除 AU 的 NERASE 数量后，将由 ERASE_TIMEOUT 指定超时值（请参见 *ERASE_TIMEOUT*）。主机应确定要在一个操作中擦除的正确 AU 数量，以便主机可以显示擦除操作的进度。如果将该字段设置为 0，则不支持擦除超时计算。

表 142. 擦除大小字段

ERASE_SIZE	值定义
0000h	不支持擦除超时计算。
0001h	1 个 AU
0002h	2 个 AU
0003h	3 个 AU
---------	---------
FFFFh	65535 个 AU

ERASE_TIMEOUT

该 6 位域指示 TERASE，该值指示当擦除 ERASE_SIZE 所指定的多个 AU 时，与偏移量之间的擦除超时。ERASE_TIMEOUT 的取值范围最大可定义为 63 秒，卡制造商可以选择 ERASE_SIZE 和 ERASE_TIMEOUT 的任意组合，具体取决于实现情况。确定 ERASE_TIMEOUT 即确定了 ERASE_SIZE。

表 143. 擦除超时字段

ERASE_TIMEOUT	值定义
00	不支持擦除超时计算。
01	1 [秒]
02	2 [秒]
03	3 [秒]
---------	---------
63	63 [秒]

ERASE_OFFSET

该 2 位域指示 TOFFSET，可选择四个值之一。如果 ERASE_SIZE 和 ERASE_TIMEOUT 字段设置为 0，则该字段无意义。

表 144. 擦除偏移字段

ERASE_OFFSET	值定义
0h	0 [秒]
1h	1 [秒]
2h	2 [秒]
3h	3 [秒]

28.4.13 SD I/O 模式

SD I/O 中断

为使 SD I/O 卡能够中断多媒体卡/SD 模块，在 SD 接口的一个引脚上提供了一个中断功能。引脚 8（当以 4 位 SD 模式运行时用作 SDIO_D1）可以向多媒体卡/SD 模块发出卡中断信号。每个卡或者卡中的功能都可以使用中断。SD I/O 中断为电平敏感型，这意味着，在多媒体卡/SD 模块识别出中断线并对其采取操作或者由于中断周期结束而使其失效之前，中断线必须保持有效状态（低电平）。在多媒体卡/SD 模块处理完中断后，将通过在 SD I/O 卡内部寄存器中的相应位中进行 I/O 写入来清除中断状态位。所有 SD I/O 卡的中断输出均为低电平有效，并且应用程序必须在所有数据线 (SDIO_D[3:0]) 上提供外部上拉电阻。只有在中断周期内，多媒体卡/SD 模块才会将引脚 8 (SDIO_D/IRQ) 的电平仅采样到中断检测器中。而在所有其他时候，多媒体卡/SD 模块将忽略此值。

中断周期适用于存储器和 I/O 操作。对于具有单个块的操作和多块数据传输，二者的中断周期定义有所不同。

SD I/O 挂起和恢复

在多功能 SD I/O 中或者在同时具备 I/O 和存储器功能的卡中，有多个设备（I/O 和存储器）共享对 MMC/SD 总线的访问权限。要在多个设备之间共享对 MMC/SD 模块的访问权限，SD I/O 和组合卡可以有选择性地实施“挂起/恢复”这一概念。如果卡支持挂起/恢复，则 MMC/SD 模块可以临时停止对某一功能或存储器的数据传输操作（挂起）以释放总线，从而将总线用于其他功能或存储器的更高优先级传输。此更高优先级传输完成后，原始传输将从中断位置恢复（重新开始）。是否支持“挂起/恢复”功能具体取决于卡。要在 MMC/SD 总线上执行挂起/恢复，MMC/SD 模块将执行以下步骤：

1. 确定当前使用 SDIO_D [3:0] 线的功能
2. 请求将优先级较低或速度较慢的事务挂起
3. 等待事务挂起完成
4. 开始更高优先级的事务
5. 等待更高优先级事务完成
6. 恢复挂起的事务

SD I/O 读取等待

可选的读取等待 (RW) 操作仅针对 SD 的 1 位和 4 位模式进行了定义。“读取等待”操作允许 MMC/SD 模块发出如下信号：卡正在读取多个寄存器 (IO_RW_EXTENDED, CMD53) 以临时停止数据传输，同时允许 MMC/SD 模块向 SD I/O 设备中的任何功能发送命令。要确定卡是否支持“读取等待”协议，MMC/SD 模块必须测试内部卡寄存器中的功能位。“读取等待”的计时是基于中断周期的。

28.4.14 命令和响应

应用程序特定的命令和常规命令

SD 卡主机模块系统旨在为各种应用程序类型提供一个标准接口。在此环境中，需要有特定的客户/应用程序功能。为实现这些功能，在标准中定义了两种类型的通用命令：应用程序特定的命令 (ACMD) 和常规命令 (GEN_CMD)。

当卡接收到 APP_CMD (CMD55) 命令时，卡所需的下一个命令是应用程序特定的命令。ACMD 与常规多媒体卡命令的结构相同，且 CMD 编号也可以相同。卡将其识别 ACMD 的依据是：它出现在 APP_CMD (CMD55) 的后面。如果后面紧跟 APP_CMD (CMD55) 的命令不是已定义的应用程序特定的命令，则会使用标准命令。例如，如果卡具有 SD_STATUS (ACMD13) 的定义并且接收到了后跟 APP_CMD (CMD55) 的 CMD13，则此命令被解释为 SD_STATUS (ACMD13)。但是，如果卡接收到后跟 APP_CMD (CMD55) 的 CMD7，但是卡没有 ACMD7 的定义，则此命令被解释为标准的 (SELECT/DESELECT_CARD) CMD7。

要使用制造商特定的 ACMD，SD 卡主机必须执行以下步骤：

1. 发送 APP_CMD (CMD55)
   卡对多媒体卡/SD 模块进行响应，指示已将 APP_CMD 位置 1，并指示当前需要的是 ACMD。
2. 发送所需的 ACMD
   卡对多媒体卡/SD 模块进行响应，指示已将 APP_CMD 位置 1，并指示接受的命令被解释为 ACMD。如果发送非 ACMD，则卡会将其作为普通多媒体卡命令进行处理，卡状态寄存器中的 APP_CMD 位保持清零。

如果发送了无效命令（既不是 ACMD 也不是 CMD），则将作为标准的多媒体卡非法命令错误来处理。

GEN_CMD 的总线事务与单块读/写命令（WRITE_BLOCK, CMD24 或 READ_SINGLE_BLOCK,CMD17）的总线事务相同。在此情况下，参数表示数据传输的方向而非地址，数据块具有供应商特定的格式和含义。

发送 GEN_CMD (CMD56) 之前，必须选择卡（处于传输状态）。数据块大小由 SET_BLOCKLEN (CMD16) 定义。对 GEN_CMD (CMD56) 的响应为 R1b 格式。

命令类型

应用程序特定的命令和常规命令均具有以下四种类型：

- **广播命令 (BC)**：发送到所有卡；不返回任何响应。
- **具有响应的广播命令 (BCR)**：发送到所有卡；同时接收来自所有卡的响应。
- **编址（点到点）命令 (AC)**：发送到选定的卡；不包含 SDIO_D 线上的数据传输。
- **编址（点到点）数据传输命令 (ADTC)**：发送到选定的卡；包含 SDIO_D 线上的数据传输。

命令格式

请参见*第 777 页的表 129* 以了解命令格式。

多媒体卡/SD 模块的命令

表 145. 面向块的写入命令

CMD 索引	类型	参数	响应格式	缩写	说明
CMD23	ac	[31:16] 设置为 0 [15:0] 块数量	R1	SET_BLOCK_COUNT	用于定义将在后面的多块读/写命令中传输的块数量。
CMD24	adtc	[31:0] 数据地址	R1	WRITE_BLOCK	写入一个块，可通过 SET_BLOCKLEN 命令选择其大小。
CMD25	adtc	[31:0] 数据地址	R1	WRITE_MULTIPLE_BLOCK	连续写入数据块，直至出现 STOP_TRANSMISSION，或者直至已接收到所请求的块数量。

表 145. 面向块的写入命令

CMD 索引	类型	参数	响应格式	缩写	说明
CMD26	adtc	[31:0] 填充位	R1	PROGRAM_CID	对卡标识寄存器进行编程。每个卡只能发出一次该命令。卡包含的硬件可防止在首次编程后执行此操作。通常该命令留给供制造商使用。
CMD27	adtc	[31:0] 填充位	R1	PROGRAM_CSD	对 CSD 的可编程位进行编程。

表 146. 面向块的写保护命令

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	如果卡具有写保护功能，此命令会将编址组的写保护位置 1。写保护的属性以卡特定的数据 (WP_GRP_SIZE) 进行编码。
CMD29	ac	[31:0] 数据地址	R1b	CLR_WRITE_PROT	如果卡提供了写保护功能，此命令会将编址组的写保护位清零。
CMD30	adtc	[31:0] 写保护数据地址	R1	SEND_WRITE_PROT	如果卡提供了写保护功能，此命令将要求卡发送写保护位的状态。
CMD31	保留				

表 147. 擦除命令

CMD 索引	类型	参数	响应格式	缩写	说明
CMD32 ... CMD34	保留。不能使用这些命令索引来保持与较旧版本多媒体卡的向后兼容性。				
CMD35	ac	[31:0] 数据地址	R1	ERASE_GROUP_START	在要选择进行擦除的某个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	在要选择进行擦除的某个连续范围内设置最后一个擦除组的地址。
CMD37	保留。不能使用此命令索引来保持与较旧版本多媒体卡的向后兼容性。				
CMD38	ac	[31:0] 填充位	R1	ERASE	擦除所有先前选择的写入块。

表 148. 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 [15:15] 寄存器写入标志 [14:8] 寄存器地址 [7:0] 寄存器数据	R4	FAST_IO	用于写入和读取 8 位（寄存器）数据字段。该命令用于对卡和寄存器寻址，如果将写入标志置 1，则提供要写入的数据。R4 响应包含从寻址寄存器中读取的数据。该命令可访问与应用程序相关、未在多媒体卡标准中定义的寄存器。
CMD40	bcr	[31:0] 填充位	R5	GO_IRQ_STATE	使系统处于中断模式。
CMD41	保留				

表 149. 锁定卡

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/重置密码或锁定/解锁卡。数据块的大小由 SET_BLOCK_LEN 命令设置。
CMD43 ... CMD54	保留				

表 150. 应用程序特定的命令

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	告知卡下一个命令位是应用程序特定的命令，而非标准命令
CMD56	adtc	[31:1] 填充位 [0]：RD/WR			用于将数据块传输到卡，或从卡中获取数据块以用于常规命令/应用程序特定的命令。数据块的大小由 SET_BLOCK_LEN 命令设置。
CMD57 ...CMD59	保留。				
CMD60 ...CMD63	留给制造商使用。				

28.5 响应格式

所有响应都是通过 MCCMD 命令线 SDIO_CMD 来发送。响应传输始终从对应于响应代码字的位字符串的左侧位开始。代码长度取决于响应类型。

响应始终从起始位（始终为 0）开始，后跟用于指示传输方向的位 (card = 0)。下表中标有 x 的值表示变量项。除 R3 响应类型之外的所有响应均受 CRC 保护。每个命令代码字由结束位（始终为 1）来终止。

有五种类型的响应。其格式定义如下：

28.5.1 R1（正常响应命令）

代码长度 = 48 位。45:40 位用于指示待响应命令的索引，该值被解释为二进制编码的数字（介于 0 和 63 之间）。卡的状态采用 32 位进行编码。

表 151. R1 响应

位的位置	宽度（位）	值	说明
47	1	0	起始位
46	1	0	传输位
[45:40]	6	X	命令索引
[39:8]	32	X	卡状态
[7:1]	7	X	CRC7
0	1	1	结束位

28.5.2 R1b

同 R1，并且有一个可选的繁忙信号在数据线上传输。根据卡在接收命令之前所处的状态，卡在接收到这些命令之后可能会变为繁忙状态。

28.5.3 R2（CID 和 CSD 寄存器）

代码长度 = 136 位。发送 CID 寄存器的内容以响应 CMD2 和 CMD10 命令。发送 CSD 寄存器的内容以响应 CMD9。仅传输 CID 和 CSD 的位 [127...1]，这些寄存器的保留位 [0] 将被替换为响应的结束位。卡通过将 MCDAT 保持低电平来指示擦除正在进行中。实际擦除时间可能很长，主机可以发出 CMD7 来取消选择卡。

表 152. R2 响应

位的位置	宽度（位）	值	说明
135	1	0	起始位
134	1	0	传输位
[133:128]	6	“111111”	命令索引
[127:1]	127	X	卡状态
0	1	1	结束位

28.5.4 R3（OCR 寄存器）

代码长度：48 位。发送 OCR 寄存器的内容以响应 CMD1。电平编码如下所示：受限电压窗口 = 低电平，卡繁忙 = 低电平。

表 153. R3 响应

位的位置	宽度（位）	值	说明
47	1	0	起始位
46	1	0	传输位
[45:40]	6	“111111”	保留
[39:8]	32	X	OCR 寄存器
[7:1]	7	“1111111”	保留
0	1	1	结束位

28.5.5 R4（快速 I/O）

代码长度：48 位。参数字段包含寻址卡的 RCA、要读出或写入的寄存器地址及其内容。

表 154. R4 响应

位的位置		宽度（位）	值	说明
47		1	0	起始位
46		1	0	传输位
[45:40]		6	“100111”	CMD39
[39:8] 参数字段	[31:16]	16	X	RCA
	[15:8]	8	X	寄存器地址
	[7:0]	8	X	读取寄存器内容
[7:1]		7	X	CRC7
0		1	1	结束位

28.5.6 R4b

仅限 SD I/O：接收 CMD5 的 SDIO 卡将以唯一的 SDIO 响应 R4 进行响应。格式为：

表 155. R4b 响应

位的位置		宽度（位）	值	说明
47		1	0	起始位
46		1	0	传输位
[45:40]		6	x	保留
[39:8] 参数字段	39	16	X	卡就绪
	[38:36]	3	X	I/O 功能的数量
	35	1	X	存在存储器
	[34:32]	3	X	填充位
	[31:8]	24	X	I/O ORC
[7:1]		7	X	保留
0		1	1	结束位

一旦 SD I/O 卡接收到 CMD5，即会使能该卡的 I/O 部分以正常响应所有后续命令。I/O 卡中的此功能 I/O 使能将始终保持有效，直至卡接收到复位、电源重启或对 I/O 复位执行写入操作的 CMD52。请注意，仅具存储器的 SD 卡可对 CMD5 进行响应。仅具存储器的卡的正确响应将为：*存在存储器* = 1，*I/O 功能的数量* = 0。仅具存储器的卡（其设计标准符合 SD 存储卡规范版本 1.0）会将 CMD5 视为非法命令，并且不予响应。可感知 I/O 的主机将发送 CMD5。如果卡以响应 R4 进行响应，则主机会根据 R4 响应中包含的数据来确定卡的配置。

28.5.7 R5（中断请求）

仅用于多媒体卡。代码长度：48 位。如果响应由主机生成，则参数中的 RCA 字段将为 0x0。

表 156. R5 响应

位的位置		宽度（位）	值	说明
47		1	0	起始位
46		1	0	传输位
[45:40]		6	“101000”	CMD40
[39:8] 参数字段	[31:16]	16	X	胜出的卡或主机的 RCA [31:16]
	[15:0]	16	X	未定义。可用于 IRQ 数据
[7:1]		7	X	CRC7
0		1	1	结束位

28.5.8 R6

仅用于 SD I/O。存储设备对 CMD3 的正常响应。相关内容在*表 157* 中进行了说明。

表 157. R6 响应

位的位置		宽度（位）	值	说明
47		1	0	起始位
46		1	0	传输位
[45:40]		6	“101000”	CMD40
[39:8] 参数字段	[31:16]	16	X	胜出的卡或主机的 RCA [31:16]
	[15:0]	16	X	未定义。可用于 IRQ 数据
[7:1]		7	X	CRC7
0		1	1	结束位

当 CMD3 发送到 I/O 专用卡时，卡的状态位 [23:8] 将更改。在此情况下，响应的 16 位代表仅限 SD I/O 的值：

- 位 [15] COM_CRC_ERROR
- 位 [14] ILLEGAL_COMMAND
- 位 [13] ERROR
- 位 [12:0] 保留

28.6 SDIO I/O 卡专用操作

以下功能是专用于 SD I/O 的操作：

- 通过触发 SDIO_D2 执行的 SDIO 读取等待操作
- 通过停止时钟执行的 SDIO 读取等待操作
- SDIO 挂起/恢复操作（写入和读取挂起）
- SDIO 中断

仅当 SDIO_DCTRL[11] 位置 1 时，SDIO 才支持这些操作，但读取挂起操作除外，该操作不需要专门的硬件实现。

28.6.1 通过触发 SDIO_D2 执行的 SDIO I/O 读取等待操作

可以在收到第一个数据块前启动读取等待间隔：如果使能数据路径（SDIO_DCTRL[0] 位置 1）并使能 SDIO 专用操作（SDIO_DCTRL[11] 位置 1），则当读取等待开始（SDI0_DCTRL[10] =0 且 SDI_DCTRL[8] =1）并且数据方向为从卡到 SDIO (SDIO_DCTRL[1] = 1) 时，DPSM 直接从空闲状态进入读取等待状态。在读取等待状态下，DPSM 在 2 个 SDIO_CK 时钟周期后将 SDIO_D2 驱动为 0。在此状态下，如果将 RWSTOP 位 (SDIO_DCTRL[9]) 置 1，DPSM 保持等待状态的时间将多出两个 SDIO_CK 时钟周期，以将 SDIO_D2 驱动为 1 并保持一个时钟周期（根据 SDIO 规范）。之后，DPSM 重新开始等待，直至从卡中接收到数据。当 DPSM 接收到数据块时，即使读取等待开始位置 1，也不会启动读取等待间隔：读取等待间隔将在接收到 CRC 后启动。必须将 RWSTOP 位清零，才能开始新的读取等待操作。在读取等待间隔内，SDIO 可以在 SDIO_D1 上检测 SDIO 中断。

28.6.2 通过停止 SDIO_CK 执行的 SDIO 读取等待操作

如果 SDIO 卡不支持上述读取等待方法，SDIO 可以通过停止 SDIO_CK 来执行读取等待（SDIO_DCTRL 置 1，这一点与*第 28.6.1 节*所述方法相同，但 SDIO_DCTRL[10] =1）：达到当前接收数据块的结束位后，DSPM 将在两个 SDIO_CK 周期后停止时钟，并在读取等待开始位置 1 后再次启动时钟。

SDIO_CK 停止时，可以向卡发出任何命令。在读取/等待间隔期间，SDIO 可以在 SDIO_D1 上检测 SDIO 中断。

28.6.3 SDIO 挂起/恢复操作

向卡发送数据时，SDIO 可以挂起写操作。SDIO_CMD[11] 位将置 1 并向 CPSM 指示当前命令为挂起命令。CPSM 将分析响应，一旦从卡接收到 ACK（接受挂起）信号，即确认 CPSM 在接收到当前数据块的 CRC 令牌后将变为空闲状态。

硬件不会保存要完成挂起的操作（恢复）还需发送的剩余数据块数量。

写操作可以由软件挂起，只需在从卡接收到挂起命令的 ACK 信号时禁止 DPSM (SDIO_DCTRL[0] =0)。DPSM 随即进入空闲状态。

要挂起读取操作：DPSM 将以 Wait_r 状态进行等待，因为要挂起的功能会在停止数据事务前发送一个完整的数据包。应用程序将继续读取 RxFIFO 直至 FIF0 为空，然后 DPSM 自动进入空闲状态。

28.6.4 SDIO 中断

一旦 SDIO_DCTRL[11] 位置 1，即会在 SDIO_D1 线路上检测到 SDIO 中断。

28.7 CE-ATA 专用操作

以下功能是专用于 CE-ATA 的操作：

- 向 CE-ATA 器件发送命令完成信号禁止
- 从 CE-ATA 器件接收命令完成信号
- 使用状态位和/或中断，向 CPU 发出 CE-ATA 命令完成信号。

SDIO 仅对 CE-ATA CMD61 命令支持这些操作，也就是说，应将 SDIO_CMD[14] 置 1。

28.7.1 命令完成信号禁止

如果“使能 CMD 完成”位 SDIO_CMD[12] 未置 1，而“非中断使能”位 SDIO_CMD[13] 置 1，则在接收到短响应后，再经过 8 个位周期，将发送命令完成信号禁止。

CPSM 进入挂起状态，禁止序列“00001”将加载到命令移位寄存器中，43 将加载到命令计数器中。八个周期后，触发器将 CPSM 变为发送状态。当命令计数器达到 48 时，CPSM 将变为空闲状态，因为不再等待任何响应。

28.7.2 命令完成信号使能

如果“使能 CMD 完成”位 SDIO_CMD[12] 置 1 并且“非中断使能”位 SDIO_CMD[13] 也置 1，则 CPSM 将以 Waitcpl 状态等待命令完成信号。

如果在 CMD 线路上接收到“0”，CPSM 将进入空闲状态。在 7 个位周期内将不能再发送任何新命令。在上述 7 个周期内，CMD 线路会在后 5 个周期以推挽模式驱动为“1”。

28.7.3 CE-ATA 中断

通过状态位 SDIO_STA[23] 向 CPU 发出命令完成信号。此静态位可通过清零位 SDIO_ICR[23] 来清零。

SDIO_STA[23] 状态位可在每个中断线上产生一个中断，具体取决于屏蔽位 SDIO_MASKx[23]。

28.7.4 中止 CMD61

如果尚未发送命令完成禁止信号并且需要中止 CMD61，则必须禁止命令状态机。CMD61 随后进入空闲状态，并且可发送 CMD12 命令。在此操作期间，不会发送任何命令完成禁止信号。

28.8 硬件流控制

硬件流控制功能用于避免 FIFO 下溢（发送模式）和上溢（接收模式）错误。

该功能可停止 SDIO_CK 并冻结 SDIO 状态机。数据传输将停止，而 FIFO 无法发送或接收数据。只有由 SDIOCLK 提供时钟的状态机才会冻结，APB2 接口仍保持活动状态。因此，即使激活流控制，仍可填充或清空 FIFO。

要使能硬件流控制，SDIO_CLKCR[14] 寄存器位必须置 1。复位后，流控制将禁止。

28.9 SDIO 寄存器

器件通过 32 位控制寄存器（可通过 APB2 访问）与系统进行通信。

外设寄存器必须按字（32 位）进行访问。

28.9.1 SDIO 电源控制寄存器 (SDIO_POWER)

SDIO power control register

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

位 31:2 保留，必须保持复位值

位 1:0 **PWRCTRL**：电源控制位 (Power supply control bits)。

这些位用于定义卡时钟的当前功能状态：

00：掉电：停止为卡提供时钟。

01：保留

10：保留，上电

11：通电：为卡提供时钟。

注意： *此寄存器的两次写访问至少需要相隔七个 HCLK 时钟周期。*

注意： *写入数据后，在三个 SDIOCLK (48 MHz) 时钟周期以及两个 PCLK2 时钟周期内无法向此寄存器写入数据。*

28.9.2 SDI 时钟控制寄存器 (SDIO_CLKCR)

SDI clock control register

偏移地址：0x04

复位值：0x0000 0000

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	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																	HWFC_EN	NEGEDGE	WIDBUS		BYPASS	PWRSAV	CLKEN	CLKDIV								
																	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:15 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：禁止硬件流控制
1：使能硬件流控制
如果使能硬件流控制，请参见*第 28.9.11 节*的 SDIO 状态寄存器定义来确定 TXFIFOE 和 RXFIFOE 中断信号的含义。

位 13 **NEGEDGE**：SDIO_CK 移相选择位 (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**：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 位宽总线模式：使用 SDIO_D[3:0]
10：8 位宽总线模式：使用 SDIO_D[7:0]

位 10 **BYPASS**：时钟分频器旁路使能位 (Clock divider bypass enable bit)
0：禁止旁路：在驱动 SDIO_CK 输出信号前，根据 CLKDIV 值对 SDIOCLK 进行分频。
1：使能旁路：SDIOCLK 直接驱动 SDIO_CK 输出信号。

位 9 **PWRSAV**：节能模式配置位 (Power saving configuration bit)
要实现节能模式，可在总线空闲时通过将 PWRSAV 置 1 来禁止 SDIO_CK 时钟输出：
0：始终使能 SDIO_CK 时钟
1：仅在总线激活时使能 SDIO_CK

位 8 **CLKEN**：时钟使能位 (Clock enable bit)
0：禁止 SDIO_CK
1：使能 SDIO_CK

位 7:0 **CLKDIV**：时钟分频系数 (Clock divide factor)
该字段定义输入时钟 (SDIOCLK) 与输出时钟 (SDIO_CK) 之间的分频系数：
SDIO_CK 频率 = SDIOCLK / [CLKDIV + 2]。

注意： *处于 SD/SDIO 卡或 MMC 识别模式时，SDIO_CK 频率必须小于 400 kHz。*

如果为所有卡分配相对卡地址，则可以将时钟频率更改为卡总线最大频率。

写入数据后，在三个 SDIOCLK (48 MHz) 时钟周期以及两个 PCLK2 时钟周期内无法向该寄存器写入数据。在 SD I/O 卡的读取等待间隔期间也可以停止 SDIO_CK：在此情况下，SDIO_CLKCR 寄存器不对 SDIO_CK 进行控制。

28.9.3 SDIO 参数寄存器 (SDIO_ARG)

SDIO argument register

偏移地址：0x08

复位值：0x0000 0000

SDIO_ARG 寄存器包含一个 32 位命令参数，该参数作为命令消息的一部分发送到卡。

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

位 31:0 **CMDARG**：命令参数 (Command argument)

作为命令消息的一部分发送给卡的命令参数。如果命令包含参数，则在将命令写入到命令寄存器之前，必须将参数加载到此寄存器中。

28.9.4 SDIO 命令寄存器 (SDIO_CMD)

SDIO command register

偏移地址：0x0C

复位值：0x0000 0000

SDIO_CMD 寄存器包含命令索引和命令类型位。命令索引作为命令消息的一部分而发送给卡。命令类型位控制命令路径状态机 (CPSM)。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																	CE-ATACMD	nIEN	ENCMDcompl	SDIOSuspend	CPSMEN	WAITPEND	WAITINT	WAITRESP		CMDINDEX						
																	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:15 保留，必须保持复位值

位 14 **ATACMD**：CE-ATA 命令 (CE-ATA command)

如果 ATACMD 置 1，则 CPSM 将传输 CMD61。

位 13 **nIEN**：非中断使能 (not Interrupt Enable)

如果该位为 0，则使能 CE-ATA 设备中的中断。

位 12 **ENCMDcompl**：使能 CMD 完成 (Enable CMD completion)

如果此位置 1，则使能命令完成信号。

位 11 **SDIOSuspend**：SD I/O 挂起命令 (SD I/O suspend command)
如果此位置 1，则要发送的命令为挂起命令（仅用于 SDIO 卡）。

位 10 **CPSMEN**：命令路径状态机 (CPSM) 使能位 (Command path state machine (CPSM) Enable bit)
如果此位置 1，则使能 CPSM。

位 9 **WAITPEND**：CPSM 等待数据传输结束（CmdPend 内部信号）(CPSM Waits for ends of data transfer (CmdPend internal signal))。
如果此位置 1，则 CPSM 将等到数据传输结束后才开始发送命令。

位 8 **WAITINT**：CPSM 等待中断请求 (CPSM waits for interrupt request)
如果此位置 1，则 CPSM 禁止命令超时并等待中断请求。

位 7:6 WAITRESP：等待响应位 (Wait for response bits)
这些位用于配置 CPSM 是否等待响应，如果等待，将等待哪种类型的响应。
00：无响应，但 CMDSENT 标志除外
01：短响应，但 CMDREND 或 CCRCFAIL 标志除外
10：无响应，但 CMDSENT 标志除外
11：长响应，但 CMDREND 或 CCRCFAIL 标志除外

位 5:0 **CMDINDEX**：命令索引 (Command index)
命令索引作为命令消息的一部分发送给卡。

注意： *写入数据后，在三个 SDIOCLK (48 MHz) 时钟周期以及两个 PCLK2 时钟周期内无法向该寄存器写入数据。*

MMC 可以发送两种类型的响应：短响应（48 位）或长响应（136 位）。SD 卡和 SD I/O 卡则只能发送短响应，参数因响应类型而异：软件将根据发送的命令区分响应类型。CE-ATA 设备仅发送短响应。

28.9.5 SDIO 命令响应寄存器 (SDIO_RESPCMD)

SDIO command response register

偏移地址：0x10

复位值：0x0000 0000

SDIO_RESPCMD 寄存器包含接收到的最后一个命令响应的命令索引字段。如果命令响应传输不包含命令索引字段（长响应或 OCR 响应），则 RESPCMD 字段为未知，但该字段必须包含 111111b（响应中保留字段的值）。

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

位 31:6 保留，必须保持复位值

位 5:0 **RESPCMD**：响应命令索引 (Response command index)
只读位域。包含接收到的最后一个命令响应的命令索引。

28.9.6 SDIO 响应 1..4 寄存器 (SDIO_RESPx)

SDIO response 1..4 register

偏移地址：(0x10 + (4 × x)); x = 1..4

复位值：0x0000 0000

SDIO_RESP1/2/3/4 寄存器包含卡的状态，该状态来自接收的响应。

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

位 31:0 **CARDSTATUSx**：请参见*表 158*。

卡状态为 32 位或 127 位，具体取决于响应类型。

表 158. 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	未使用	卡状态 [95:64]
SDIO_RESP3	未使用	卡状态 [63:32]
SDIO_RESP4	未使用	卡状态 [31:1]0b

首先接收卡状态的最高有效位。SDIO_RESP3 寄存器 LSB 始终为 0b。

28.9.7 SDIO 数据定时器寄存器 (SDIO_DTIMER)

SDIO data timer register

偏移地址：0x24

复位值：0x0000 0000

SDIO_DTIMER 寄存器包含以卡总线时钟周期表示的数据超时周期。

计数器加载 SDIO_DTIMER 寄存器的值，并在数据路径状态机进入 Wait_R 或繁忙状态后开始递减。如果定时器达到 0 而 DPSM 处于上述一种状态，则超时状态标志置 1。

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

位 31:0 **DATATIME**：数据超时周期 (Data timeout period)
以卡总线时钟周期表示的数据超时周期。

注意： *数据传输必须首先写入到数据计时器寄存器和数据长度寄存器，然后才写入到数据控制寄存器。*

28.9.8 SDIO 数据长度寄存器 (SDIO_DLEN)

SDIO data length register

偏移地址：0x28

复位值：0x0000 0000

SDIO_DLEN 寄存器包含要传输的数据字节数。当数据传输开始时，值将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	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved							DATALENGTH																								
							rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:25 保留，必须保持复位值

位 24:0 **DATALENGTH：**数据长度值 (Data length value)
要传输的数据字节数量。

注意： *对于块数据传输，数据长度寄存器中的值必须是块大小的倍数（请参见 SDIO_DCTRL）。数据传输必须首先写入到数据计时器寄存器和数据长度寄存器，然后才写入到数据控制寄存器。*

对于 SDIO 多字节传输，数据长度寄存器中的值必须在 1 到 512 之间。

28.9.9 SDIO 数据控制寄存器 (SDIO_DCTRL)

SDIO data control register

偏移地址：0x2C

复位值：0x0000 0000

SDIO_DCTRL 寄存器控制数据路径状态机 (DPSM)。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																				SDIOEN	RWMOD	RWSTOP	RWSTART	DBLOCKSIZE				DMAEN	DTMODE	DTDIR	DTEN
																				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:12 保留，必须保持复位值

位 11 **SDIOEN：**SD I/O 使能功能 (SD I/O enable functions)
如果将该位置 1，则 DPSM 执行特定于 SD I/O 卡的操作。

位 10 **RWMOD：**读取等待模式 (Read wait mode)
0：通过停止 SDIO_D2 进行读取等待控制
1：使用 SDIO_CK 进行读取等待控制

位 9 **RWSTOP：**读取等待停止 (Read wait stop)
0：如果将 RWSTART 位置 1，则读取等待正在进行中
1：如果将 RWSTART 位置 1，则使能读取等待停止

位 8 **RWSTART**：读取等待开始 (Read wait start)
如果将该位置 1，则读取等待操作开始。

位 7:4 **DBLOCKSIZE**：数据块大小 (Data block size)
定义在选择了块数据传输模式时数据块的长度：
0000：（十进制数 0）块长度 = 2^0 = 1 字节
0001：（十进制数 1）块长度 = 2^1 = 2 字节
0010：（十进制数 2）块长度 = 2^2 = 4 字节
0011：（十进制数 3）块长度 = 2^3 = 8 字节
0100：（十进制数 4）块长度 = 2^4 = 16 字节
0101：（十进制数 5）块长度 = 2^5 = 32 字节
0110：（十进制数 6）块长度 = 2^6 = 64 字节
0111：（十进制数 7）块长度 = 2^7 = 128 字节
1000：（十进制数 8）块长度 = 2^8 = 256 字节
1001：（十进制数 9）块长度 = 2^9 = 512 字节
1010：（十进制数 10）块长度 = 2^{10} = 1024 字节
1011：（十进制数 11）块长度 = 2^{11} = 2048 字节
1100：（十进制数 12）块长度 = 2^{12} = 4096 字节
1101：（十进制数 13）块长度 = 2^{13} = 8192 字节
1110：（十进制数 14）块长度 = 2^{14} = 16384 字节
1111：（十进制数 15）保留

位 3 **DMAEN**：DMA 使能位 (DMA enable bit)
0：禁止 DMA。
1：使能 DMA。

位 2 **DTMODE**：数据传输模式选择 (Data transfer mode selection)
0：块数据传输
1：流或 SDIO 多字节数据传输

位 1 **DTDIR**：数据传输方向选择 (Data transfer direction selection)
0：从控制器到卡。
1：从卡到控制器。

位 0 **DTEN**：数据传输使能位 (Data transfer enabled bit)
如果 1 写入到 DTEN 位，则数据传输开始。根据方向位 DTDIR，如果在传输开始时立即将 RW 置 1开始，则 DPSM 变为 Wait_S 状态、Wait_R 状态或读取等待状态。在数据传输结束后不需要将使能位清零，但必须更新 SDIO_DCTRL 以使能新的数据传输

注意： *写入数据后，在三个 SDIOCLK (48 MHz) 时钟周期以及两个 PCLK2 时钟周期内无法向该寄存器写入数据。*

根据 SDIOEN 位的值，DTMODE 位的含义将有所不同。如果 SDIOEN=0 且 DTMODE=1，则使能 MMC 流模式，如果 SDIOEN=1 且 DTMODE=1，则外设使能 SDIO 多字节传输。

28.9.10 SDIO 数据计数器寄存器 (SDIO_DCOUNT)

SDIO data counter register

偏移地址：0x30

复位值：0x0000 0000

当 DPSM 从空闲状态变为 Wait_R 或 Wait_S 状态时，SDIO_DCOUNT 寄存器将从数据长度寄存器中加载值（请参见 SDIO_DLEN）。在传输数据时，计数器将值递减直至计数器达到 0。然后 DPSM 将变为空闲状态，并且将数据状态结束标志 DATAEND 置 1。

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

位 31:25 保留，必须保持复位值

位 24:0 **DATACOUNT**：数据计数值 (Data count value)
读取该位时，将返回要传输的剩余数据字节数量。写入没有任何效果。

注意： *仅当数据传输完成后才应读取该寄存器。*

28.9.11 SDIO 状态寄存器 (SDIO_STA)

SDIO status register

偏移地址：0x34

复位值：0x0000 0000

SDIO_STA 寄存器是一个只读寄存器。它包含两种类型的标志：

- 静态标志（位 [23:22,10:0]）：在通过写入到 SDIO 中断清零寄存器来清零这些位之前，会一直保持这些位（请参见 SDIO_ICR）。
- 动态标志（位 [21:11]）：这些位根据底层逻辑的状态来更改状态（例如，随着数据写入到 FIFO，发出和停止发出 FIFO 满和空标志）

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								CEATAEND	SDIOIT	RXDAVL	TXDAVL	RXFIFOE	TXFIFOE	RXFIFOF	TXFIFOF	RXFIFOHF	TXFIFOHE	RXACT	TXACT	CMDACT	DBCKEND	STBITERR	DATAEND	CMDSENT	CMDREND	RXOVERR	TXUNDERR	DTIMEOUT	CTIMEOUT	DCRCFAIL	CCRCFAIL
Res.								r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

位 31:24 保留，必须保持复位值

位 23 **CEATAEND**：针对 CMD61 收到了 CE-ATA 命令完成信号 (CE-ATA command completion signal received for CMD61)

位 22 **SDIOIT**：收到了 SDIO 中断 (SDIO interrupt received)

位 21 **RXDAVL**：接收 FIFO 中有数据可用 (Data available in receive FIFO)

位 20 **TXDAVL**：传输 FIFO 中有数据可用 (Data available in transmit FIFO)

位 19 **RXFIFOE**：接收 FIFO 为空 (Receive FIFO empty)

位 18 **TXFIFOE**：发送 FIFO 为空 (Transmit FIFO empty)
如果使能了硬件流控制，则 TXFIFOE 信号在 FIFO 包含 2 个字时激活。

位 17 **RXFIFOF**：接收 FIFO 已满 (Receive FIFO full)
如果使能了硬件流控制，则 RXFIFOF 信号在 FIFO 差 2 个字便变满之前激活。

位 16 **TXFIFOF**：传输 FIFO 已满 (Transmit FIFO full)

位 15 **RXFIFOHF**：接收 FIFO 半满：FIFO 中至少有 8 个字 (Receive FIFO half full: there are at least 8 words in the FIFO)

位 14 **TXFIFOHE**：传输 FIFO 半空：至少可以写入 8 个字到 FIFO (Transmit FIFO half empty: at least 8 words can be written into the FIFO)

位 13 **RXACT**：数据接收正在进行中 (Data receive in progress)

位 12 **TXACT**：数据传输正在进行中 (Data transmit in progress)

位 11 **CMDACT**：命令传输正在进行中 (Command transfer in progress)

位 10 **DBCKEND**：已发送 / 接收数据块（CRC 校验通过）(Data block sent/received (CRC check passed))

位 9 **STBITERR**：在宽总线模式下，并非在所有数据信号上都检测到了起始位 (Start bit not detected on all data signals in wide bus mode)

位 8 **DATAEND**：数据结束（数据计数器 SDIDCOUNT 为零）(Data end (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**：已接收命令响应（CRC 校验通过）(Command response received (CRC check passed))

位 5 **RXOVERR**：收到了 FIFO 上溢错误 (Received FIFO overrun error)

位 4 **TXUNDERR**：传输 FIFO 下溢错误 (Transmit FIFO underrun error)

位 3 **DTIMEOUT**：数据超时 (Data timeout)

位 2 **CTIMEOUT**：命令响应超时 (Command response timeout)
命令超时周期为固定值 64 个 SDIO_CK 时钟周期。

位 1 **DCRCFAIL**：已发送 / 接收数据块（CRC 校验失败）(Data block sent/received (CRC check failed))

位 0 **CCRCFAIL**：已接收命令响应（CRC 校验失败）(Command response received (CRC check failed))

28.9.12 SDIO 中断清零寄存器 (SDIO_ICR)

SDIO interrupt clear register

偏移地址：0x38

复位值：0x0000 0000

SDIO_ICR 寄存器是一个只写寄存器。以 1 写入某个位会将 SDIO_STA 状态寄存器中的对应位清零。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								CEATAENDC	SDIOITC	Reserved											DBCKENDC	STBITERRC	DATAENDC	CMDSENTC	CMDRENDC	RXOVERRC	TXUNDERRC	DTIMEOUTC	CTIMEOUTC	DCRCFAILC	CCRCFAILC
								rw	rw												rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:24 保留，必须保持复位值

位 23 **CEATAENDC**：CEATAEND 标志清零位 (CEATAEND flag clear bit)
由软件置 1，用于将 CEATAEND 标志清零。
0：未将 CEATAEND 清零
1：已将 CEATAEND 清零

位 22 **SDIOITC**：SDIOIT 标志清零位 (SDIOIT flag clear bit)
由软件置 1，用于将 SDIOIT 标志清零。
0：未将 SDIOIT 清零
1：已将 SDIOIT 清零

位 21:11 保留，必须保持复位值

位 10 **DBCKENDC**：DBCKEND 标志清零位 (DBCKEND flag clear bit)
由软件置 1，用于将 DBCKEND 标志清零。
0：未将 DBCKEND 清零
1：已将 DBCKEND 清零

位 9 **STBITERRC**：STBITERR 标志清零位 (STBITERR flag clear bit)
由软件置 1，用于将 STBITERR 标志清零。
0：未将 STBITERR 清零
1：已将 STBITERR 清零

位 8 **DATAENDC**：DATAEND 标志清零位 (DATAEND flag clear bit)
由软件置 1，用于将 DATAEND 标志清零。
0：未将 DATAEND 清零
1：已将 DATAEND 清零

位 7 **CMDSENTC**：CMDSENT 标志清零位 (CMDSENT flag clear bit)
由软件置 1，用于将 CMDSENT 标志清零。
0：未将 CMDSENT 清零
1：已将 CMDSENT 清零

位 6 **CMDRENDC**：CMDREND 标志清零位 (CMDREND flag clear bit)
由软件置 1，用于将 CMDREND 标志清零。
0：未将 CMDREND 清零
1：已将 CMDREND 清零

位 5 **RXOVERRC**：RXOVERR 标志清零位 (RXOVERR flag clear bit)
由软件置 1，用于将 RXOVERR 标志清零。
0：未将 RXOVERR 清零
1：已将 RXOVERR 清零

位 4 **TXUNDERRC**：TXUNDERR 标志清零位 (TXUNDERR flag clear bit)
由软件置 1，用于将 TXUNDERR 标志清零。
0：未将 TXUNDERR 清零
1：已将 TXUNDERR 清零

位 3 **DTIMEOUTC**：DTIMEOUT 标志清零位 (DTIMEOUT flag clear bit)
由软件置 1，用于将 DTIMEOUT 标志清零。
0：未将 DTIMEOUT 清零
1：已将 DTIMEOUT 清零

位 2 **CTIMEOUTC**：CTIMEOUT 标志清零位 (CTIMEOUT flag clear bit)
由软件置 1，用于将 CTIMEOUT 标志清零。
0：未将 CTIMEOUT 清零
1：已将 CTIMEOUT 清零

位 1 **DCRCFAILC**：DCRCFAIL 标志清零位 (DCRCFAIL flag clear bit)
由软件置 1，用于将 DCRCFAIL 标志清零。
0：未将 DCRCFAIL 清零
1：已将 DCRCFAIL 清零

位 0 **CCRCFAILC**：CCRCFAIL 标志清零位 (CCRCFAIL flag clear bit)
由软件置 1，用于将 CCRCFAIL 标志清零。
0：未将 CCRCFAIL 清零
1：已将 CCRCFAIL 清零

28.9.13 SDIO 屏蔽寄存器 (SDIO_MASK)

SDIO mask register

偏移地址：0x3C

复位值：0x0000 0000

中断屏蔽寄存器通过将对应的位置 1 来确定哪一个状态标志位可以产生中断。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								CEATAENDIE	SDIOITIE	RXDAVLIE	TXDAVLIE	RXFIFOEIE	TXFIFOEIE	RXFIFOFIE	TXFIFOFIE	RXFIFOHFIE	TXFIFOHEIE	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	rw	rw	rw	rw	rw	rw	rw	rw

位 31:24 保留，必须保持复位值

位 23 **CEATAENDIE**：接收到 CE-ATA 命令完成信号时中断使能 (CE-ATA command completion signal received interrupt enable)
由软件置 1 和清零，用于使能/禁止收到 CE-ATA 命令完成信号时生成的中断。
0：禁止接收到 CE-ATA 命令完成信号时中断
1：使能接收到 CE-ATA 命令完成信号时中断

位 22 **SDIOITIE**：接受到 SDIO 模式中断时中断使能 (SDIO mode interrupt received interrupt enable)
由软件置 1 和清零，用于使能/禁止收到 SDIO 模式中断时生成中断。
0：禁止接收到 SDIO 模式中断时中断
1：使能接收到 SDIO 模式中断时中断

位 21 **RXDAVLIE**：Rx FIFO 中数据可用时中断使能 (Data available in Rx FIFO interrupt enable)
由软件置 1 和清零，用于使能/禁止由于 Rx FIFO 中存在数据而生成的中断。
0：禁止 Rx FIFO 中有数据时中断
1：使能 Rx FIFO 中有数据时中断

位 20 **TXDAVLIE**：Tx FIFO 中数据可用时中断使能 (Data available in Tx FIFO interrupt enable)
由软件置 1 和清零，用于使能/禁止由于 Tx FIFO 中存在数据而生成的中断。
0：禁止 Tx FIFO 中数据可用时中断
1：使能 Tx FIFO 中数据可用时中断

位 19 **RXFIFOEIE**：Rx FIFO 为空时中断使能 (Rx FIFO empty interrupt enable)
由软件置 1 和清零，用于使能/禁止由 Rx FIFO 为空引起的中断。
0：禁止 Rx FIFO 为空时中断
1：使能 Rx FIFO 为空时中断

位 18 **TXFIFOEIE**：Tx FIFO 为空时中断使能 (Tx FIFO empty interrupt enable)
由软件置 1 和清零，用于使能/禁止由 Tx FIFO 为空引起的中断。
0：禁止 Tx FIFO 为空时中断
1：使能 Tx FIFO 为空时中断

位 17 **RXFIFOFIE**：Rx FIFO 变满时中断使能 (Rx FIFO full interrupt enable)
由软件置 1 和清零，用于使能/禁止由 Rx FIFO 变满引起的中断。
0：禁止 Rx FIFO 变满时中断
1：使能 Rx FIFO 变满时中断

位 16 **TXFIFOFIE**：Tx FIFO 变满时中断使能 (Tx FIFO full interrupt enable)
由软件置 1 和清零，用于使能/禁止由 Tx FIFO 变满引起的中断。
0：禁止 Tx FIFO 变满时中断
1：使能 Tx FIFO 变满时中断

位 15 **RXFIFOHFIE**：Rx FIFO 半满时中断使能 (Rx half full interrupt enable)
由软件置 1 和清零，用于使能/禁止由 Rx FIFO 半满引起的中断。
0：禁止 Rx FIFO 半满时中断
1：使能 Rx FIFO 半满时中断

位 14 **TXFIFOHEIE**：Tx FIFO 半空时中断使能 (Tx FIFO half empty interrupt enable)
由软件置 1 和清零，用于使能/禁止由 Tx FIFO 半空引起的中断。
0：禁止 Tx FIFO 半空时中断
1：使能 Tx FIFO 半空时中断

位 13 **RXACTIE**：数据接收操作中断使能 (Data receive acting interrupt enable)
由软件置 1 和清零，用于使能/禁止由正在接收的数据（数据接收操作）导致的中断。
0：禁止数据接收操作中断
1：使能数据接收操作中断

位 12 **TXACTIE**：数据传输操作中断使能 (Data transmit acting interrupt enable)
由软件置 1 和清零，用于使能/禁止由正在传输的数据（数据传输操作）导致的中断。
0：禁止数据传输操作中断
1：使能数据传输操作中断

位 11 **CMDACTIE**：命令操作中断使能 (Command acting interrupt enable)
由软件置 1 和清零，用于使能/禁止由正在传输的命令（命令操作）导致的中断。
0：禁止命令操作中断
1：使能命令操作中断

位 10 **DBCKENDIE**：数据块结束中断使能 (Data block end interrupt enable)
由软件置 1 和清零，用于使能/禁止由数据块结束引起的中断。
0：禁止数据块结束中断
1：使能数据块结束中断

位 9 **STBITERRIE**：起始位错误中断使能 (Start bit error interrupt enable)
由软件置 1 和清零，用于使能/禁止由起始位错误引起的中断。
0：禁止起始位错误中断
1：使能起始位错误中断

位 8 **DATAENDIE**：数据结束中断使能 (Data end interrupt enable)
由软件置 1 和清零，用于使能/禁止由数据结束引起的中断。
0：禁止数据结束中断
1：使能数据结束中断

位 7 **CMDSENTIE**：命令发送中断使能 (Command sent interrupt enable)
由软件置 1 和清零，用于使能/禁止由发送命令引起的中断。
0：禁止命令发送中断
1：使能命令发送中断

位 6 **CMDRENDIE**：命令响应接收中断使能 (Command response received interrupt enable)
由软件置 1 和清零，用于使能/禁止由接收命令响应引起的中断。
0：禁止命令响应接收中断
1：使能命令响应接收中断

位 5 **RXOVERRIE**：Rx FIFO 上溢错误中断使能 (Rx FIFO overrun error interrupt enable)
由软件置 1 和清零，用于使能/禁止由 Rx FIFO 上溢错误引起的中断。
0：禁止 Rx FIFO 上溢错误中断
1：使能 Rx FIFO 上溢错误中断

位 4 **TXUNDERRIE**：Tx FIFO 下溢错误中断使能 (Tx FIFO underrun error interrupt enable)
由软件置 1 和清零，用于使能/禁止由 Tx FIFO 下溢错误引起的中断。
0：禁止 Tx FIFO 下溢错误中断
1：使能 Tx FIFO 下溢错误中断

位 3 **DTIMEOUTIE**：数据超时中断使能 (Data timeout interrupt enable)
由软件置 1 和清零，用于使能/禁止由数据超时引起的中断。
0：禁止数据超时中断
1：使能数据超时中断

位 2 **CTIMEOUTIE**：命令超时中断使能 (Command timeout interrupt enable)
由软件置 1 和清零，用于使能/禁止由命令超时引起的中断。
0：禁止命令超时中断
1：使能命令超时中断

位 1 **DCRCFAILIE**：数据 CRC 失败中断使能 (Data CRC fail interrupt enable)
由软件置 1 和清零，用于使能/禁止由数据 CRC 失败引起的中断。
0：禁止数据 CRC 失败中断
1：使能数据 CRC 失败中断

位 0 **CCRCFAILIE**：命令 CRC 失败中断使能 (Command CRC fail interrupt enable)
由软件置 1 和清零，用于使能/禁止由命令 CRC 失败引起的中断。
0：禁止命令 CRC 失败中断
1：使能命令 CRC 失败中断

28.9.14 SDIO FIFO 计数器寄存器 (SDIO_FIFOCNT)

SDIO FIFO counter register

偏移地址：0x48

复位值：0x0000 0000

SDIO_FIFOCNT 寄存器包含要在 FIFO 中写入或读取的剩余字数。如果数据控制寄存器（SDIO_DCTRL 寄存器）中将数据传输使能位 DTEN 置 1，并且 DPSM 处于空闲状态，则 FIFO 计数器从数据长度寄存器中加载值（请参见 SDIO_DLEN）。如果数据长度未进行字对齐（4 的倍数），则剩余的 1 到 3 个字节被视为一个字。

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

位 31:24 保留，必须保持复位值

位 23:0 **FIFOCOUNT**：要在 FIFO 中写入或读取的剩余字数。

28.9.15 SDIO 数据 FIFO 寄存器 (SDIO_FIFO)

SDIO data FIFO register

偏移地址：0x80

复位值：0x0000 0000

接收和发送 FIFO 可以被当做 32 位宽的寄存器来读出或写入。FIFO 在 32 个连续地址上包含 32 个入口。这使 CPU 可以使用其加载 (load) 和存储 (store) 多操作符命令来读写 FIFO。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIF0Data																															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:0 **FIFOData**：接收和传输 FIFO 数据 (Receive and transmit FIFO data)
FIFO 数据占用 32 位字的 32 个入口，从地址：
SDIO 基址 + 0x080 到 SDIO 基址 + 0xFC。

28.9.16 SDIO 寄存器映射

下表对 SDIO 寄存器进行了汇总。

表 159. SDIO 寄存器映射

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	SDIO_POWER	Reserved																														PWRCTRL	
0x04	SDIO_CLKCR	Reserved																	HWFC_EN	NEGEDGE	WIDBUS		BYPASS	PWRSAV	CLKEN	CLKDIV							
0x08	SDIO_ARG	CMDARG																															
0x0C	SDIO_CMD	Reserved																	CE-ATACMD	nIEN	ENCMDcompl	SDIOSuspend	CPSMEN	WAITPEND	WAITINT	WAITRESP	CMDINDEX					
0x10	SDIO_RESPCMD	Reserved																															
0x14	SDIO_RESP1	CARDSTATUS1																															
0x18	SDIO_RESP2	CARDSTATUS2																															
0x1C	SDIO_RESP3	CARDSTATUS3																															
0x20	SDIO_RESP4	CARDSTATUS4																															
0x24	SDIO_DTIMER	DATATIME																															
0x28	SDIO_DLEN	Reserved																	DATALENGTH														
0x2C	SDIO_DCTRL	Reserved																				SDIOEN	RWMOD	RWSTOP	RWSTART	DBLOCKSIZE				DMAEN	DTMODE	DTDIR	DTEN
0x30	SDIO_DCOUNT	Reserved																	DATACOUNT														
0x34	SDIO_STA	Reserved								CEATAEND	SDIOIT	RXDAVL	TXDAVL	RXFIFOE	TXFIFOE	RXFIFOF	TXFIFOF	RXFIFOHF	TXFIFOHE	RXACT	TXACT	CMDACT	DBCKEND	STBITERR	DATAEND	CMDSENT	CMDREND	RXOVERR	TXUNDERR	DTIMEOUT	CTIMEOUT	DCRCFAIL	CCRCFAIL
0x38	SDIO_ICR	Reserved								CEATAENDC	SDIOITC	Reserved											DBCKENDC	STBITERRC	DATAENDC	CMDSENTC	CMDRENDC	RXOVERRC	TXUNDERRC	DTIMEOUTC	CTIMEOUTC	DCRCFAILC	CCRCFAILC
0x3C	SDIO_MASK	Reserved								CEATAENDIE	SDIOITIE	RXDAVLIE	TXDAVLIE	RXFIFOEIE	TXFIFOEIE	RXFIFOFIE	TXFIFOFIE	RXFIFOHFIE	TXFIFOHEIE	RXACTIE	TXACTIE	CMDACTIE	DBCKENDIE	STBITERRIE	DATAENDIE	CMDSENTIE	CMDRENDIE	RXOVERRIE	TXUNDERRIE	DTIMEOUTIE	CTIMEOUTIE	DCRCFAILIE	CCRCFAILIE
0x48	SDIO_FIFOCNT	Reserved								FIFOCOUNT																							
0x80	SDIO_FIFO	FIF0Data																															

有关寄存器边界地址的信息，请参见*第 52 页的表 2*。

29 以太网 (ETH)：通过 DMA 控制器进行介质访问控制 (MAC)

除非特别说明，否则本节适用于整个 STM32F4xx 系列器件。

29.1 以太网简介

借助以太网外设，STM32F4xx 可以通过以太网按照 IEEE 802.3-2002 标准发送和接收数据。

以太网提供了可配置、灵活的外设，用以满足客户的各种应用需求。它支持与外部物理层 (PHY) 相连的两个工业标准接口：默认情况下使用的介质独立接口 (MII)（在 IEEE 802.3 规范中定义）和简化介质独立接口 (RMII)。它有多种应用领域，例如交换机、网络接口卡等。

以太网遵守以下标准：

- IEEE 802.3-2002，用于以太网 MAC
- IEEE 1588-2008 标准，用于规定联网时钟同步的精度。
- AMBA 2.0，用于 AHB 主/从端口
- RMII 联盟的 RMII 规范

29.2 以太网主要特性

以太网 (ETH) 外设包括以下特性（按类列出）：

29.2.1 MAC 内核特性

- 支持外部 PHY 接口实现 10/100 Mbit/s 数据传输速率
- 通过符合 IEEE 802.3 的 MII 接口与外部快速以太网 PHY 进行通信
- 支持全双工和半双工操作
  - 支持适用于半双工操作的 CSMA/CD 协议
  - 支持适用于全双工操作的 IEEE 802.3x 流量控制
  - 全双工操作时可以将接收的暂停控制帧转发到用户应用程序
  - 半双工操作时提供背压流量控制
  - 全双工操作中如果流量控制输入信号消失，将自动发送零时间片暂停帧
- 报头和帧起始数据 (SFD) 在发送路径中插入、在接收路径中删除
- 可逐帧控制 CRC 和 pad 自动生成
- 接收帧时可自动去除 pad/CRC
- 可编程帧长度，支持高达 16 KB 的巨型帧
- 可编程帧间隔（40-96 位时间，以 8 为步长）

- 支持多种灵活的地址过滤模式：
  - 高达四个 48 位完美 (DA) 地址过滤器，对每个字节进行掩码操作
  - 高达三次 48 位 SA 地址比较检查，对每个字节进行掩码操作
  - 64 位 Hash 滤波器（可选），适用于多播和单播 (DA) 地址
  - 可传送所有多播地址帧
  - 支持混合模式，因此可传送所有帧，无需为网络监视进行过滤
  - 传送所有传入数据包时（每次过滤时）均附有一份状态报告
- 为发送和接收数据包分别返回 32 位状态
- 支持对接收帧进行 IEEE 802.1Q VLAN 变量检测
- 为应用程序提供单独的发送、接收和控制接口
- 支持通过 RMON/MIB 计数器 (RFC2819/RFC2665) 进行强制网络统计
- 使用 MDIO 接口配置和管理 PHY 设备
- 检测 LAN 唤醒帧和 AMD Magic Packet™ 帧
- 在接收功能中支持对接收到的由以太网帧封装的 IPv4 和 TCP 数据包进行校验和卸载
- 在增强型接收功能中支持检查 IPv4 头校验和以及在 IPv4 或 IPv6 数据包中封装的 TCP、UDP 或 ICMP 校验和
- 支持以太网帧时间戳（请参见 IEEE 1588-2008）。每个帧的发送或接收状态下给出 64 位时间戳
- 两组 FIFO：一个具有可编程阈值功能的 2 KB 发送 FIFO 和一个具有可配置阈值（默认为 64 个字节）功能的 2 KB 接收 FIFO
- 接收 FIFO 进行多帧存储时，通过在 EOF 传输后向接收 FIFO 插入接收状态矢量，从而使得接收 FIFO 无需存储这些帧的接收状态
- 在存储转发模式下，可以在接收时过滤所有的错误帧，但不将这些错误帧转发给应用程序
- 可以转发过小的好帧
- 为接收 FIFO 中丢失或损坏的帧（由于溢出）生成脉冲，借此支持数据统计
- 向 MAC 内核发送数据时支持存储转发机制
- 根据接收 FIFO 填充（阈值可配置）级别自动生成要发送至 MAC 内核的暂停帧控制或背压信号
- 发送时处理冲突帧的自动重新发送
- 丢弃延迟冲突、过度冲突、过度延迟和下溢条件下的帧
- 通过软件控制刷新 Tx FIFO
- 计算 IPv4 头校验和与 TCP、UDP 或 ICMP 校验和并将其插入在存储转发模式下发送的帧中
- 支持调试时通过 MII 进行内部回送

29.2.2 DMA 特性

- AHB 从接口中支持所有 AHB 突发类型
- 软件可以在 AHB 主接口中选择 AHB 突发类型（固定或不确定突发）
- 可以从 AHB 主端口选择地址对齐突发
- 通过帧定界符优化以数据包为导向的 DMA 传输
- 支持对数据缓冲区进行字节对齐寻址
- 双缓冲区（环）或链表（链接）描述符链接
- 采用描述符架构可以在 CPU 几乎不干预的情况传输大型数据块。
- 每个描述符可传输高达 8 KB 的数据
- 报告正常工作和传输错误时的综合状态
- 可为发送和接收 DMA 引擎单独编程突发大小，以充分利用主总线
- 可编程中断选项，适用于不同的工作条件
- 按帧控制发送/接收完成中断
- 接收引擎和发送引擎间采用循环调度仲裁或固定优先级仲裁
- 启动/停止模式
- 当前 Tx/Rx 缓冲区指针作为状态寄存器
- 当前 Tx/Rx 描述符指针作为状态寄存器

29.2.3 PTP 特性

- 接收帧和发送帧时间戳
- 粗略校准法和精细校准法
- 系统时间大于目标时间时触发中断
- 输出秒脉冲（产品复用功能输出）

29.3 以太网引脚

表 160 显示了 MAC 信号和相应的 MII/RMII 信号映射。所有 MAC 信号均映射到 AF11，一些信号映射到不同的 I/O 引脚，这些应在复用功能模式下进行配置（有关详细信息，请参见 *第 7.3.2 节：I/O 引脚复用器和映射*）。

表 160. 复用功能映射

端口	AF11
	ETH
PA0-WKUP	ETH_MII_CRS
PA1	ETH_MII _RX_CLK/ETH_RMII _REF_CLK
PA2	ETH _MDIO
PA3	ETH _MII_COL
PA7	ETH_MII _RX_DV/ETH_RMII _CRS_DV
PB0	ETH _MII_RXD2
PB1	ETH _MII_RXD3
PB5	ETH _PPS_OUT
PB8	ETH _MII_TXD3
PB10	ETH_ MII_RX_ER
PB11	ETH _MII_TX_EN/ETH _RMII_TX_EN
PB12	ETH _MII_TXD0/ETH _RMII_TXD0
PB13	ETH _MII_TXD1/ETH _RMII_TXD1
PC1	ETH _MDC
PC2	ETH _MII_TXD2
PC3	ETH _MII_TX_CLK
PC4	ETH_MII_RXD0/ETH_RMII_RXD0
PC5	ETH _MII_RXD1/ETH _RMII_RXD1
PE2	ETH_MII_TXD3
PG8	ETH_PPS_OUT
PG11	ETH _MII_TX_EN/ETH _RMII_TX_EN
PG13	ETH _MII_TXD0/ETH _RMII_TXD0
PG14	ETH _MII_TXD1/ETH _RMII_TXD1
PH2	ETH _MII_CRS
PH3	ETH _MII_COL
PH6	ETH _MII_RXD2
PH7	ETH _MII_RXD3
PI10	ETH _MII_RX_ER

29.4 以太网功能说明：SMI、MII 和 RMII

以太网外设包括带专用 DMA 控制器的 MAC 802.3（介质访问控制）。它支持默认情况下使用的介质独立接口 (MII) 和简化介质独立接口 (RMII)，并通过一个选择位在两个接口间进行切换（请参见 SYSCFG_PMC 寄存器）。

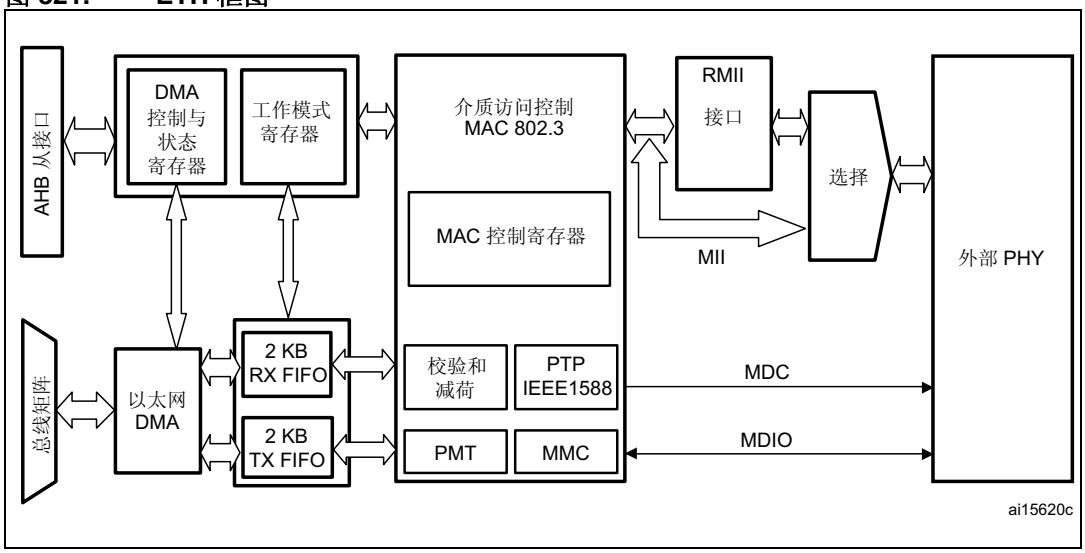
DMA 控制器通过 AHB 主从接口与内核和存储器相连。AHB 主接口用于控制数据传输，而 AHB 从接口则用于访问“控制和状态寄存器”(CSR) 的空间。

在进行数据发送时，首先将数据由系统存储器以 DMA 的方式送至发送 FIFO (Tx FIFO) 进行缓冲，再通过 MAC 内核发送。同样，接收 FIFO (Rx FIFO) 则存储通过线路接收的以太网帧，直到这些帧通过 DMA 传送到系统存储器。

以太网外设还包括用于与外部 PHY 通信的 SMI。通过一组配置寄存器，用户可以为 MAC 控制器和 DMA 控制器选择所需模式和功能。

注意： *当使用以太网时，AHB 时钟频率必须至少为 25 MHz。*

图 321. ETH 框图



1. 有关 AHB 连接的信息，请参见*图 1：STM32F405xx/07xx 和 STM32F415xx/17xx 器件的系统架构*和*图 2：STM32F42xxx 和 STM32F43xxx 器件的系统架构*。

29.4.1 站管理接口：SMI

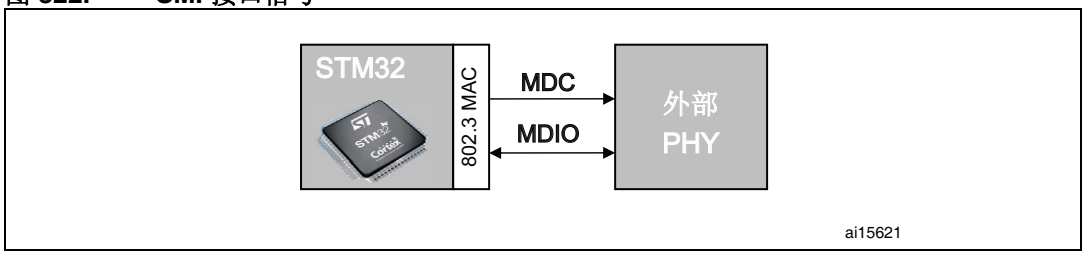
站管理接口 (SMI) 允许应用程序通过 2 线时钟和数据线访问任意 PHY 寄存器。该接口支持访问多达 32 个 PHY。

应用程序可以从 32 个 PHY 中选择一个 PHY，然后从任意 PHY 包含的 32 个寄存器中选择一个寄存器，发送控制数据或接收状态信息。任意给定时间内只能对一个 PHY 中的一个寄存器进行寻址。

MDC 时钟线和 MDIO 数据线在微控制器中均用作复用功能 I/O：

- MDC：周期性时钟，提供以最大频率 2.5 MHz 传输数据时的参考时序。MDC 的最短高电平时间和最短低电平时间必须均为 160 ns。MDC 的最小周期必须为 400 ns。在空闲状态下，SMI 管理接口将 MDC 时钟信号驱动为低电平。
- MDIO：数据输入/输出比特流，用于通过 MDC 时钟信号向/从 PHY 设备同步传输状态信息。

图 322. SMI 接口信号



SMI 帧格式

表 13 中给出了与读操作或写操作有关的帧结构，位传输顺序必须从左到右。

表 161. 管理帧格式

	管理帧字段							
	报头（32 位）	起始	操作	PADDR	RADDR	TA	数据（16 位）	空闲
读取	1...1	01	10	ppppp	rrrrr	Z0	dddddddddddddddd	Z
写入	1...1	01	01	ppppp	rrrrr	10	dddddddddddddddd	Z

管理帧包括八个字段：

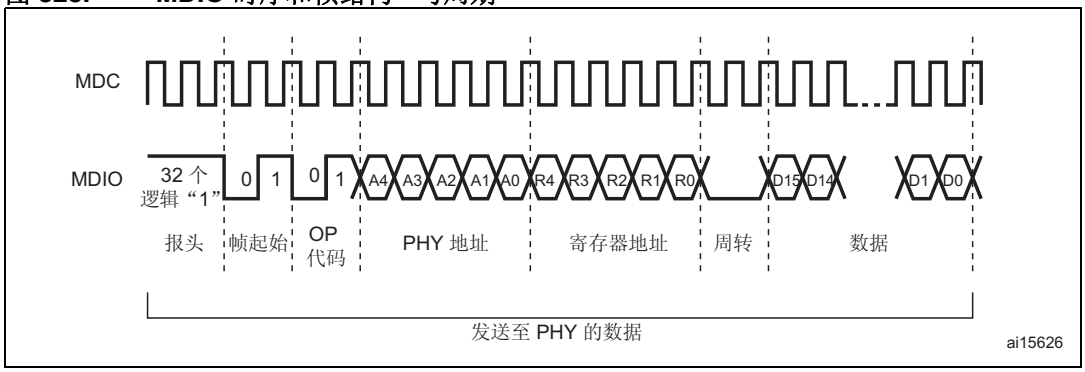
- **报头**：每个事务（读取或写入）均可通过报头字段启动，报头字段对应于 MDIO 线上 32 个连续的逻辑“1”位以及 MDC 上的 32 个周期。该字段用于与 PHY 设备建立同步。
- **起始**：帧起始由 <01> 模式定义，用于验证线路从默认逻辑“1”状态变为逻辑“0”状态，然后再从逻辑“0”状态变为逻辑“1”状态。
- **操作**：定义正在发生的事务（读取或写入）的类型。
- **PADDR**：PHY 地址有 5 位，可构成 32 个唯一 PHY 地址。最先发送和接收地址的 MSB 位。
- **RADDR**：寄存器地址有 5 位，从而可在所选 PHY 设备中对 32 个不同的寄存器进行寻址。最先发送和接收地址的 MSB 位。
- **TA**：周转字段在 RADDR 和 DATA 字段间定义了一个 2 位模式，以避免在读取事务期间出现竞争现象。读取事务时，MAC 控制器将 TA 的 2 个位驱动为 MDIO 线上的高阻态。PHY 设备必须将 TA 的第一位驱动为高阻态，将 TA 的第二位驱动为“0”。
  写入事务时，MAC 控制器针对 TA 字段驱动 <10> 模式。PHY 设备必须将 TA 的 2 个位驱动为高阻态。
- **数据**：数据字段为 16 位。最先发送和接收的位必须为 ETH_MIID 寄存器的位 15。
- **空闲**：MDIO 线驱动为高阻态。三态驱动器必须禁止，PHY 的上拉电阻使线路保持逻辑“1”状态。

SMI 写操作

当应用程序将 MII 写入位和繁忙位（在*以太网 MAC MII 地址寄存器 (ETH_MACMIIAR)* 中）置 1 时，SMI 将通过传输 PHY 地址、PHY 中的寄存器地址以及写入数据（在*以太网 MAC MII 数据寄存器 (ETH_MACMIIDR)* 中）来触发对 PHY 寄存器进行写操作。事务进行期间，应用程序不应更改 MII 地址寄存器的内容或 MII 数据寄存器。在此期间对 MII 地址寄存器或 MII 数据寄存器执行的写操作将会忽略（繁忙位处于高电平状态），事务将无错完成。写操作完成后，SMI 将通过复位繁忙位进行指示。

图 323 显示了写操作的帧格式。

图 323. MDIO 时序和帧结构 - 写周期

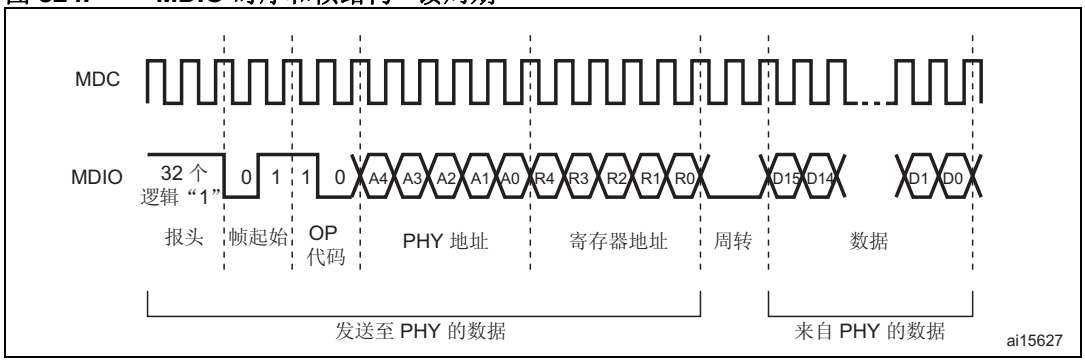


SMI 读操作

当用户将以太网 MAC MII 地址寄存器 (ETH_MACMIIAR) 中的 MII 繁忙位置 1、MII 写入位清零时，SMI 将通过传输 PHY 地址和 PHY 中的寄存器地址在 PHY 寄存器中触发读操作。事务进行期间，应用程序不应更改 MII 地址寄存器的内容或 MII 数据寄存器。在此期间对 MII 地址寄存器或 MII 数据寄存器执行的写操作将会忽略（繁忙位处于高电平状态），事务将无错完成。读操作完成后，SMI 将复位繁忙位，然后用从 PHY 中读取的数据更新 MII 数据寄存器。

图 324 显示了读操作的帧格式。

图 324. MDIO 时序和帧结构 - 读周期



SMI 时钟选择

MAC 启动管理写/读操作。SMI 时钟是一个分频时钟，其时钟源为应用时钟（AHB 时钟）。分频系数取决于 MII 地址寄存器中设置的时钟范围。

表 162 显示了如何设置时钟范围。

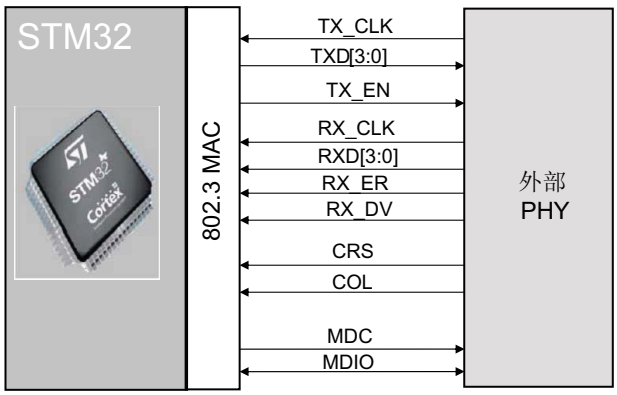
表 162. 时钟范围

选择	HCLK 时钟	MDC 时钟
000	60-100 MHz	AHB 时钟/42
001	100-168 MHz	AHB 时钟/62
010	20-35 MHz	AHB 时钟/16
011	35-60 MHz	AHB 时钟/26
100, 101, 110, 111	保留	-

29.4.2 介质独立接口：MII

介质独立接口 (MII) 定义了 10 Mbit/s 和 100 Mbit/s 的数据传输速率下 MAC 子层与 PHY 之间的互连。

图 325. 介质独立接口信号



- MII_TX_CLK：连续时钟信号。该信号提供进行 TX 数据传输时的参考时序。标称频率为：速率为 10 Mbit/s 时为 2.5 MHz；速率为 100 Mbit/s 时为 25 MHz。
- MII_RX_CLK：连续时钟信号。该信号提供进行 RX 数据传输时的参考时序。标称频率为：速率为 10 Mbit/s 时为 2.5 MHz；速率为 100 Mbit/s 时为 25 MHz。
- MII_TX_EN：发送使能信号。该信号表示 MAC 当前正针对 MII 发送半字节。该信号必须与报头的前半字节进行同步 (MII_TX_CLK)，并在所有待发送的半字节均发送到 MII 时必须保持同步。
- MII_TXD[3:0]：数据发送信号。该信号是 4 个一组的数据信号，由 MAC 子层同步驱动，在 MII_TX_EN 信号有效时才为有效信号（有效数据）。MII_TXD[0] 为最低有效位，MII_TXD[3] 为最高有效位。禁止 MII_TX_EN 时，发送数据不会对 PHY 产生任何影响。

- MII_CRS：载波侦听信号。当发送或接收介质处于非空闲状态时，由 PHY 使能该信号。发送和接收介质均处于空闲状态时，由 PHY 禁止该信号。PHY 必须确保 MII_CS 信号在冲突条件下保持有效状态。该信号无需与 TX 和 RX 时钟保持同步。在全双工模式下，该信号没意义。
- MII_COL：冲突检测信号。检测到介质上存在冲突后，PHY 必须立即使能冲突检测信号，并且只要存在冲突条件，冲突检测信号必须保持有效状态。该信号无需与 TX 和 RX 时钟保持同步。在全双工模式下，该信号没意义。
- MII_RXD[3:0]：数据接收信号。该信号是 4 个一组的数据信号，由 PHY 同步驱动，在 MII_RX_DV 信号有效时才为有效信号（有效数据）。MII_RXD[0] 为最低有效位，MII_RXD[3] 为最高有效位。当 MII_RX_DV 禁止、MII_RX_ER 使能时，特定的 MII_RXD[3:0] 值用于传输来自 PHY 的特定信息（请参见*表 164*）。
- MII_RX_DV：接收数据有效信号。该信号表示 PHY 当前正针对 MII 接收已恢复并解码的半字节。该信号必须与恢复帧的头半字节进行同步 (MII_RX_CLK)，并且一直保持同步到恢复帧的最后半字节。该信号必须在最后半字节随后的第一个时钟周期之前禁止。为了正确地接收帧，MII_RX_DV 信号必须在时间范围上涵盖要接收的帧，其开始时间不得迟于 SFD 字段出现的时间。
- MII_RX_ER：接收错误信号。该信号必须保持一个或多个周期 (MII_RX_CLK)，从而向 MAC 子层指示在帧的某处检测到错误。该错误条件必须通过 MII_RX_DV 验证，如*表 164* 所示。

表 163. TX 接口信号编码

MII_TX_EN	MII_TXD[3:0]	说明
0	0000 到 1111	正常帧间
1	0000 到 1111	正常数据发送

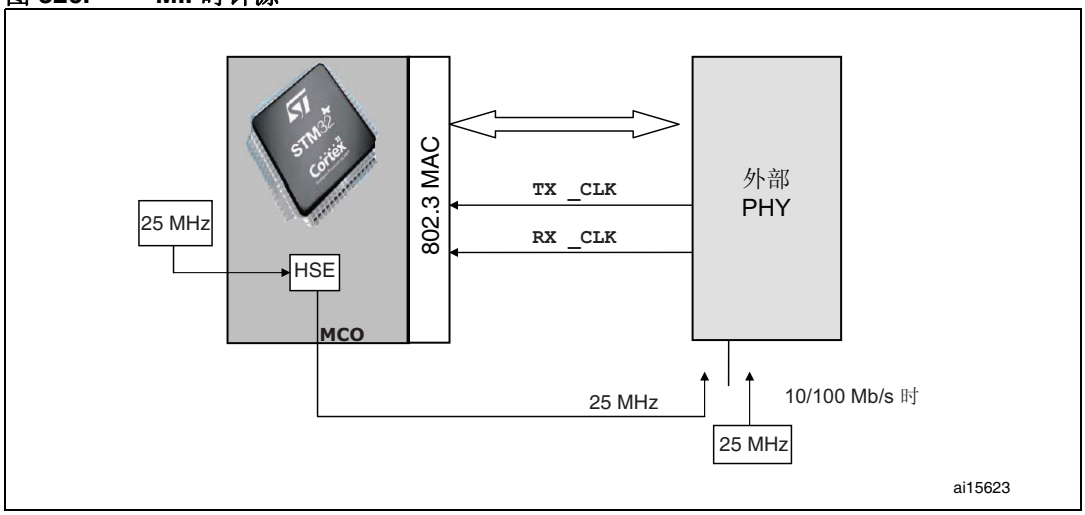
表 164. RX 接口信号编码

MII_RX_DV	MII_RX_ER	MII_RXD[3:0]	说明
0	0	0000 到 1111	正常帧间
0	1	0000	正常帧间
0	1	0001 到 1101	保留
0	1	1110	错误载波检测
0	1	1111	保留
1	0	0000 到 1111	正常数据接收
1	1	0000 到 1111	数据接收出现错误

MII 时钟源

要生成 TX_CLK 和 RX_CLK 时钟信号，必须向外部 PHY 提供 25MHz 时钟，如*图 326* 所示。除了使用外部 25 MHz 石英晶体提供该时钟，还可以通过 STM32F4xx 微控制器的 MCO 引脚输出该信号。这种情况下，必须对 PLL 倍频进行配置，以通过 25 MHz 外部石英晶体在 MCO 引脚上获得所需频率。

图 326. MII 时钟源



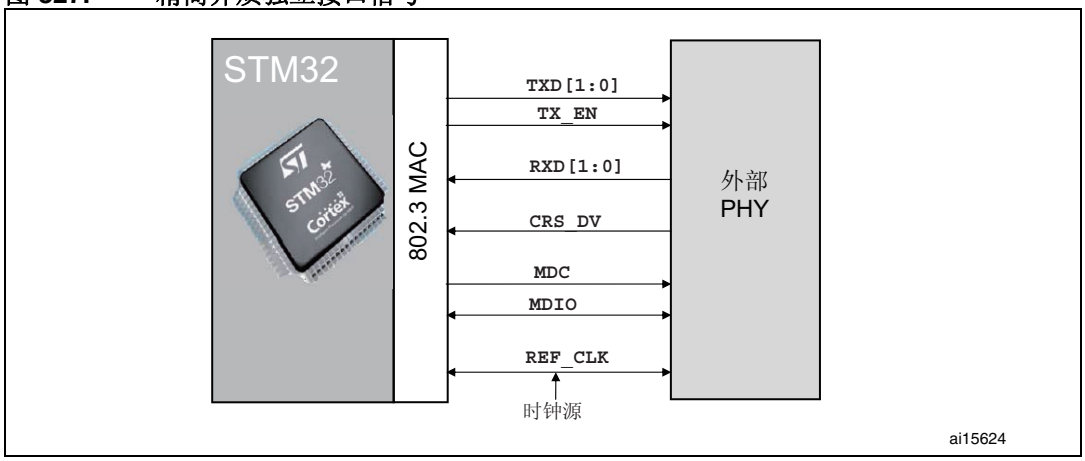
29.4.3 精简介质独立接口：RMII

精简介质独立接口 (RMII) 规范降低了 10/100 Mbit/s 下微控制器以太网外设与外部 PHY 间的引脚数。根据 IEEE 802.3u 标准，MII 包括 16 个数据和控制信号的引脚。RMII 规范将引脚数减少为 7 个（引脚数减少 62.5%）。

RMII 接口是 MAC 和 PHY 之间的实例化对象。这有助于将 MAC 的 MII 转换为 RMII。RMII 具有以下特性：

- 支持 10-Mbit/s 和 100-Mbit/s 的运行速率
- 参考时钟必须是 50 MHz
- 相同的参考时钟必须从外部提供给 MAC 和外部以太网 PHY
- 它提供了独立的 2 位宽（双位）的发送和接收数据路径

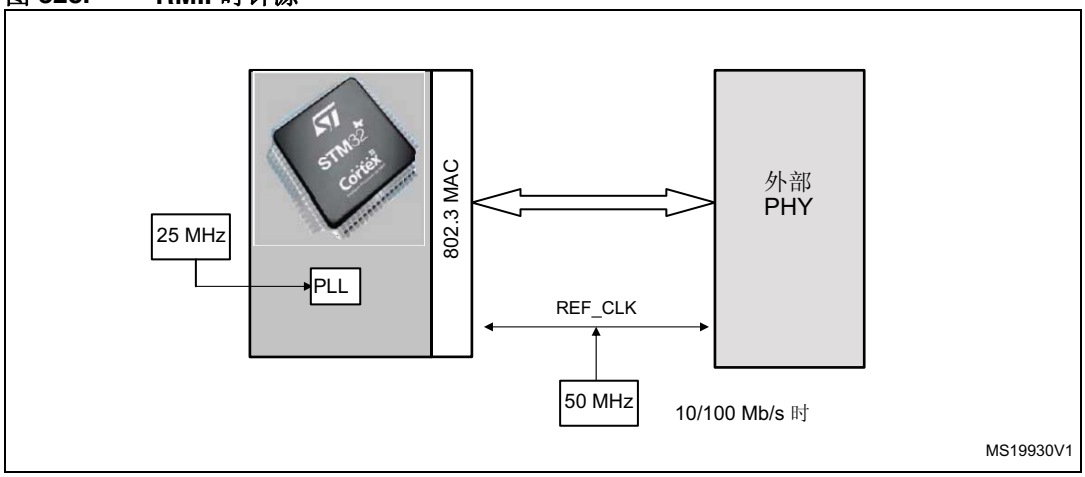
图 327. 精简介质独立接口信号



RMII 时钟源

使用外部 50 MHz 时钟驱动 PHY 或使用嵌入式 PLL 生成 50 MHz 频率信号来驱动 PHY。

图 328. RMII 时钟源



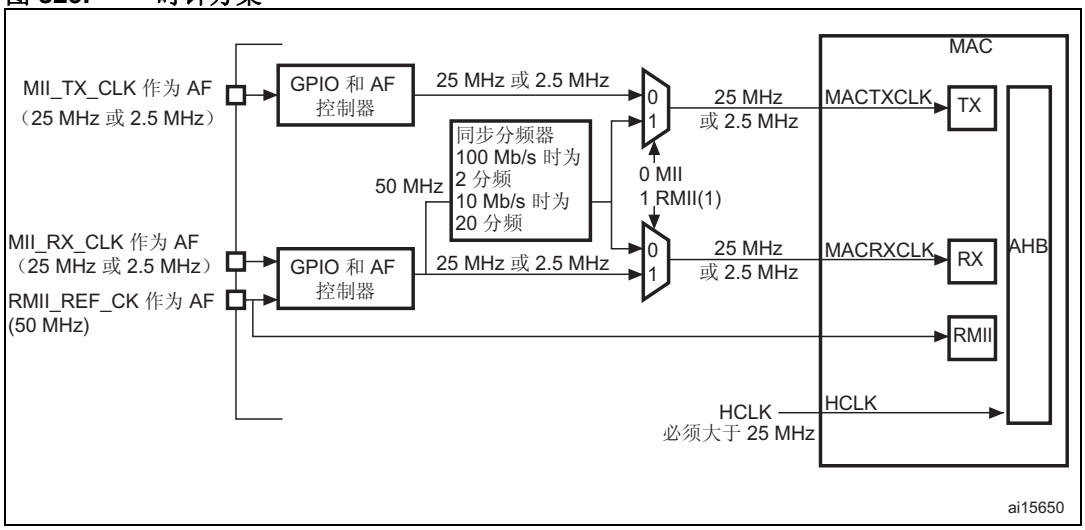
29.4.4 MII/RMII 选择

使用 SYSCFG_PMC 寄存器中的配置位 23 MII_RMII_SEL 选择 MII 或 RMII 模式。以太网控制器处于复位模式或使能时钟前，应用程序必须设置 MII/RMII 模式。

MII/RMII 内部时钟方案

支持 MII 和 RMII 以及 10 和 100 Mbit/s 运行所需的时钟方案如*图 329* 所示。

图 329. 时钟方案



1. 通过 SYSCFG_PMC 寄存器中的位 23 MII_RMII_SEL 控制 MII/RMII 选择。

要节省引脚，需在同一个 GPIO 引脚上复用 RMII_REF_CK 和 MII_RX_CLK 这两个输入时钟信号。

29.5 以太网功能说明：MAC 802.3

适用于局域网 (LAN) 的 IEEE 802.3 国际标准将 CSMA/CD（带有冲突检测的载波侦听多路访问）用作访问方法。

以太网外设包括一个带介质独立接口 (MII) 的 MAC 802.3（介质访问控制）控制器和一个专用 DMA 控制器。

MAC 模块对以下系列的系统使用 LAN CSMA/CD 子层：数据速率为 10 Mbit/s 和 100 Mbit/s 的基带系统和宽带系统。支持半双工和全双工工作模式。冲突检测访问方法仅适用于半双工工作模式。支持 MAC 控制帧子层。

MAC 子层执行以下与数据链路控制步骤相关的功能：

- 数据封装（发送和接收）
  - 组帧（帧边界定界、帧同步）
  - 寻址（处理源地址和目标地址）
  - 错误检测
- 介质访问管理
  - 介质分配（冲突避免）
  - 竞争解决（冲突处理）

MAC 子层主要有两个工作模式：

- 半双工模式：站点使用 CSMA/CD 算法争用物理介质。
- 全双工模式：满足以下条件时，无需解决竞争问题（CSMA/CD 算法不是必需的）便可同时发送和接收数据：
  - 物理介质能够支持同步发送和接收
  - 正好有 2 个站点与 LAN 相连
  - 两个站均配置为全双工工作模式

29.5.1 MAC 802.3 帧格式

正如 IEEE 802.3-2002 标准规定，MAC 块使用 MAC 子层和可选 MAC 控制子层 (10/100 Mbit/s)。

为使用 CSMA/CD MAC 的数据通信系统指定了两个帧格式：

- 基本 MAC 帧格式
- 标记 MAC 帧格式（扩展了基本 MAC 帧格式）

图 331 和*图 332* 介绍了包含以下字段的帧结构（未标记和标记）：

- 报头：7 字节字段，用于进行同步（PLS 电路）

  十六进制值：55-55-55-55-55-55-55

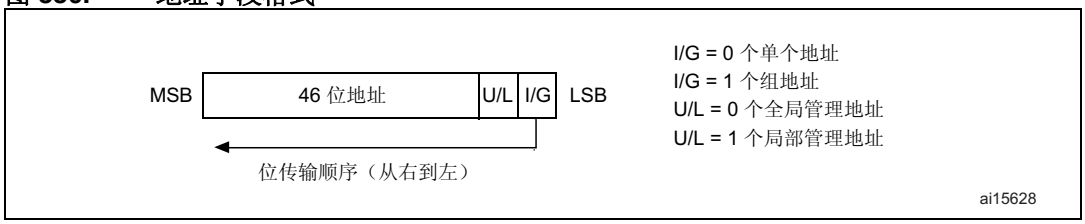
  位模式：01010101 01010101 01010101 01010101 01010101 01010101 01010101（从右到左进行位传输）
- 起始帧定界符 (SFD)：1 字节字段，用于指示帧的开始。十六进制值：D5

  位模式：11010101（从右到左进行位传输）
- 目标地址和源地址字段：MAC 地址字段（6字节），指示目标站和源站地址，具体如下（请参见*图 330*）：
  - 每个地址的长度为 48 位
  - 目标地址字段中的第一个 LSB 位 (I/G) 用于指示单个地址 (I/G = 0) 或组地址 (I/G = 1)。一个组地址可以标识“无”、一个或多个，或所有连接至 LAN 的站。在源地址中，第一位保留并复位为 0。
  - 第二位 (U/L) 用于区分局部 (U/L = 1) 或全局 (U/L = 0) 管理地址。对于广播地址，该位同样为 1。
  - 每个地址字段的各个字节必须最先发送最低有效位。

地址指定基于以下类型：

- 单个地址：与网络中的特殊站关联的物理地址。
- 组地址。与给定网络中一个或多个站关联的多目标地址。有两种多播地址：
  - 多播组地址：与一组逻辑相关站关联的地址。
  - 广播地址：一个特殊的预定义多播地址（目标地址字段中全为“1”），该地址总是表示给定 LAN 上的所有站。

图 330. 地址字段格式



- QTag 前缀：在源地址字段和 MAC 客户端长度/类型字段中插入的 4 字节字段。该字段是对基本帧（未标记）的扩展，用于获得标记的 MAC 帧。未标记的 MAC 帧不包括该字段。扩展的标记如下：
  - 2 字节常量长度/类型字段值（符合“类型”解析，大于 0x0600），等于 802.1Q 变量协议类型的值（十六进制 0x8100）。该常量字段用于区分标记和未标记的 MAC 帧。
  - 包含变量控制信息的 2 字节字段可以再分为：一个 3 位用户优先级、一个 1 位标准格式指示符 (CFI) 和一个 12 位 VLAN 标识符。标记 MAC 帧的长度通过 QTag 前缀扩展 4 个字节。
- MAC 客户端长度/类型：2 字节字段，具有不同含义（互斥），具体取决于其值：
  - 如果该值小于或等于 maxValidFrame (0d1500)，则该字段表示 802.3 帧的后续数据字段中所包含的 MAC 客户端数据字节的数量（长度解析）。
  - 如果该值大于或等于 MinTypeValue（十进制 0d1536，0x0600），则该字段表示与以太网帧相关的 MAC 客户端协议的种类（类型解析）。

  无论长度/类型字段的解析结果为何，如果数据字段的长度小于协议正确运行所需的最小长度，则将在数据字段之后、FCS（帧检查序列）字段之前添加一个 PAD 字段。发送和接收长度/类型字段时，高位字节在前。

  对于在 maxValidLength 到 minTypeValue 范围内（不包括边界）的长度/类型字段值，未指定 MAC 子层的行为：MAC 子层可能传递、也可能不传递这些值。
- 数据和 PAD 字段：n 字节数据字段。其数据完全透明，这意味着数据字段中可能出现任意顺序的字节值。PAD 的大小（如果存在）由数据字段的大小决定。数据和 PAD 字段的最大和最小长度为：
  - 最大长度 = 1500 字节
  - 无标记的 MAC 帧的最小长度 = 46 字节
  - 带标记的 MAC 帧的最小长度 = 42 字节

  当数据字段的长度小于要求的最小长度时，将添加 PAD 字段以匹配最小长度（带标记的帧为 42 字节，无标记的帧为 46 字节）。
- 帧检查序列：包含循环冗余校验 (CRC) 值的 4 字节字段。CRC 计算基于下列字段：源地址、目标地址、QTag 前缀、长度/类型、LLC 数据和 PAD（即，除报头、SFD 字段以外的所有字段）。生成的多项式如下：

$$G(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$$

帧的 CRC 值按如下方式计算：

- 帧的前 2 位是互补位
- 帧的 n 个位是 (n – 1) 次多项式 M(x) 的系数。目标地址的第一位对应于 x^{n-1} 项，数据字段的最后一位对应于 x^0 项
- M(x) 乘以 x^{32} 再除以 G(x)，得到 ≤ 31 次的余数 R(x)
- 将 R(x) 的系数视为一个 32 位序列
- 对位序列进行互补，结果为 CRC
- 32 位的 CRC 值放置在帧检查序列中。首先发送 x^{32} 项，最后发送 x^0 项

图 331. MAC 帧格式

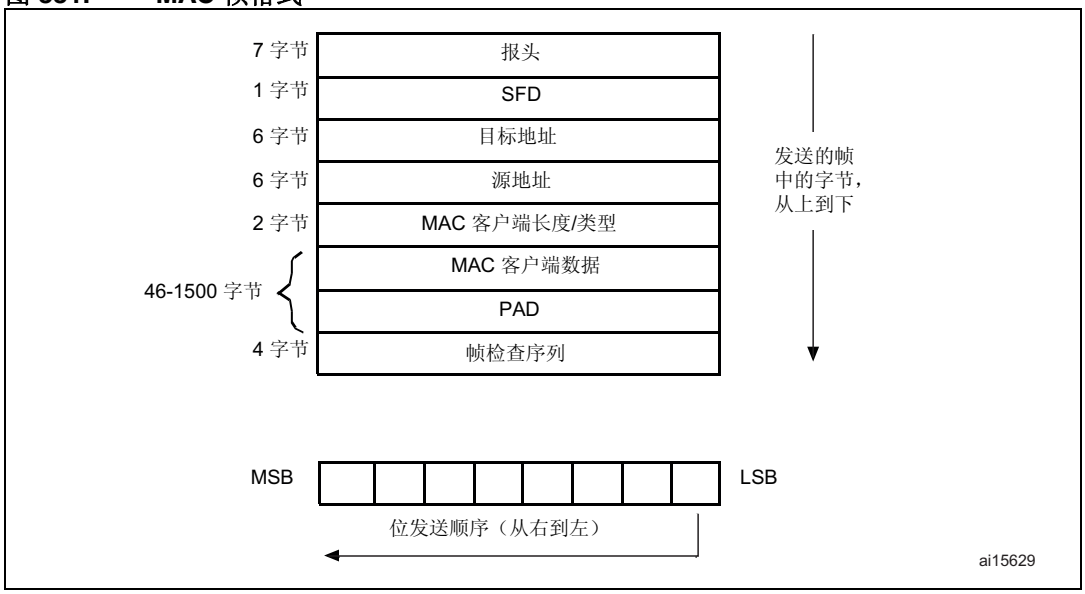
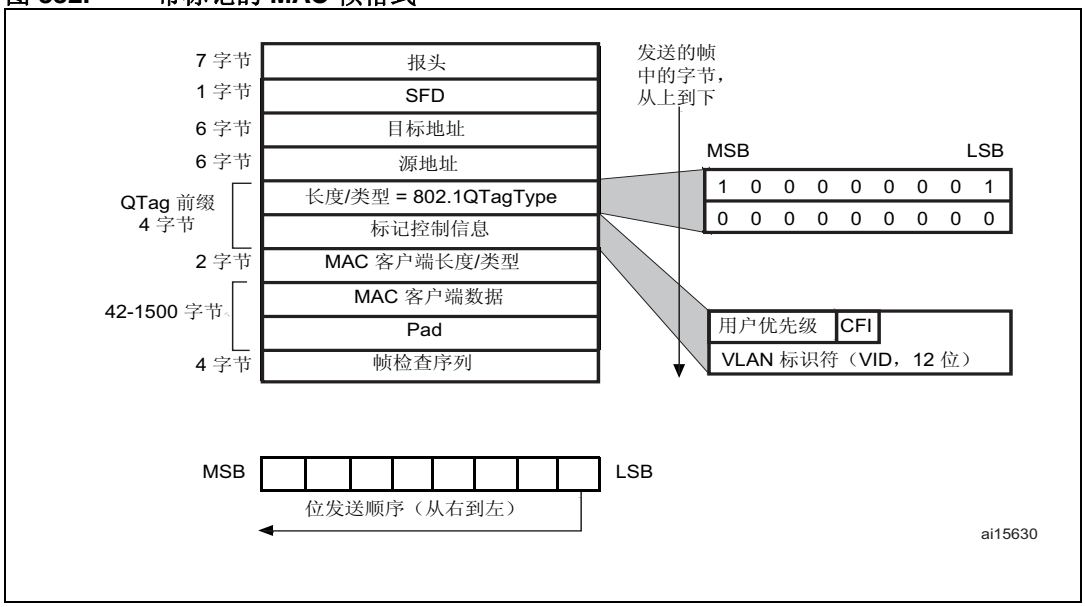


图 332. 带标记的 MAC 帧格式



发送 MAC 帧的每个字节（除 FSC 字段外）时，低位在前。

通过以下条件之一定义无效 MAC 帧：

- 帧长度与长度 / 类型字段指定的预期值不一致。如果长度 / 类型字段包含类型值，则认为帧长度与此字段一致（没有无效帧）
- 帧长度不是字节的整数倍（额外位）
- 根据传入帧计算出的 CRC 值与包含的 FCS 不匹配。

29.5.2 MAC 帧发送

DMA 控制发送路径的所有事务。从系统存储器读取的以太网帧由 DMA 推入 FIFO。然后将帧弹出并传输到 MAC 内核。帧传输结束时，从 MAC 内核获取发送状态并传回 DMA。发送 FIFO 的深度为 2 KB。FIFO 填充级别将指示给 DMA，以便 DMA 可通过 AHB 接口在所需的系统存储器突发中启动数据获取。来自 AHB 主接口的数据将推入 FIFO。

检测到 SOF 时，MAC 接受数据并开始向 MII 发送。在应用程序启动发送后，向 MII 发送帧数据所需的时间是可变的，具体取决于 IFG 延迟、发送报头/SFD 的时间以及半双工模式的任意回退延迟等延迟因素。EOF 传输到 MAC 内核后，内核将完成正常的发送，然后将发送的状态返回给 DMA。如果在发送过程中发生常规冲突（在半双工模式下），MAC 内核将使发送状态有效，然后接受并丢弃所有后续数据，直至收到下一 SOF。检测到来自 MAC 的重试请求（在状态中）时，应从 SOF 重新发送同一帧。如果发送期间未连续提供数据，MAC 将发出下溢状态。在帧的正常传输期间，如果 MAC 在未获得前一帧的 EOF 的情况下接收到 SOF，则将忽略该 SOF 并将新的帧视为前一帧的延续。

向 MAC 内核弹出数据有两种操作模式：

- 在阈值模式下，只要 FIFO 中的字节数超过配置的阈值（或在超过阈值前写入帧结束），数据就准备好弹出并转发到 MAC 内核。使用 ETH_DMABMR 的 TTC 位配置阈值。
- 在存储转发模式下，仅当在 FIFO 中存储完整的帧后，才会向 MAC 内核弹出帧。如果 Tx FIFO 的大小小于要发送的以太网帧，则在 Tx FIFO 接近填满时向 MAC 内核弹出帧。

应用可通过将 FTF 位（ETH_DMAOMR 寄存器 [20]）置 1 来清空发送 FIFO 的所有内容。此位自行清零，并将 FIFO 指针初始化为默认状态。如果向 MAC 内核传输帧时将 FTF 位置 1，则传输将停止，因为此时 FIFO 被视为空。因此，MAC 发送器将出现下溢事件并且相应的状态字将转发给 DMA。

自动 CRC 和 pad 生成

当从应用程序接收的字节数少于 60（DA+SA+LT+数据）时，会向发送帧附加零，使数据长度正好为 46 字节，以满足 IEEE 802.3 的最小数据字段要求。可将 MAC 编程为不附加任何填充值。计算帧检查序列 (FCS) 字段的循环冗余校验 (CRC) 并将其附加到正在发送的数据。如果将 MAC 编程为不将 CRC 值附加到以太网帧的末尾，则不发送计算出的 CRC。此规则有一种例外情况，即，将 MAC 编程为向小于 60 字节（DA+SA+LT+数据）的帧附加填充时，CRC 将附加在填充帧的末尾。

CRC 发生器计算以太网帧的 FCS 字段的 32 位 CRC。编码由下面的多项式定义。

$$G(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$$

发送协议

MAC 控制以太网帧的发送操作。它执行下列功能以满足 IEEE 802.3/802.3z 规范。包括：

- 生成报头和 SFD
- 在半双工模式下生成阻塞信号
- 控制 Jabber 超时
- 控制半双工模式下的流量（背压）
- 生成发送帧状态
- 包含符合 IEEE 1588 的时间戳快照逻辑

当请求发送新的帧时，MAC 将发送报头和 SFD，紧接着发送数据。报头定义为 0b10101010 模式的 7 个字节，SFD 定义为 0b10101011 模式的 1 个字节。冲突窗口定义为 1 个时隙（对于 10/100 Mb/s 以太网，为 512 个位时间）。阻塞信号生成仅适用于半双工模式，不适用于全双工模式。

在 MII 模式下，如果在开始传输帧到 CRC 字段结束之间的任何时间发生冲突，MAC 将在 MII 上发送 0x5555 5555 的 32 位阻塞信号，通知所有其它站已发生冲突。如果在报头发送阶段发生冲突，MAC 将完成报头和 SFD 的发送，然后发送阻塞信号。

系统使用一个 jabber 定时器，用于在传输的字节超过 2048 字节（默认值）时切断以太网帧的发送。在半双工模式下，MAC 使用延迟机制进行流量控制（背压）。当应用程序请求停止接收帧时，如果已使能发送流量控制，则只要 MAC 检测到接收帧，就会发送一个 32 字节的 JAM 信号。这会导致冲突并使远程站回退。应用程序通过将 ETH_MACFCR 寄存器中的 BPA 位（位 0）置 1 来请求流量控制。如果应用程序请求发送帧，则即使激活背压功能，也将按调度计划发送。请注意，如果背压功能长时间保持激活（发生的连续冲突事件超过 16 个），则远程站将由于冲突过多而中止发送。如果针对发送帧使能 IEEE 1588 时间戳功能，则将 SFD 置于发送 MII 总线时，此模块会获取系统时间的快照。

发送调度程序

MAC 负责调度 MII 上的帧发送。它可保持两个发送帧之间的帧间隔，并在半双工模式下遵循截断二进制指数回退算法。在满足 IFG 和回退延迟条件后，MAC 使能发送。它可确保两个发送帧之间的空闲期段，即配置的帧间隔（ETH_MACCR 寄存器中的 IFG 位）。如果要发送的帧在配置的 IFG 时间之前到达，则 MII 会等待来自 MAC 的使能信号，然后再开始发送。只要 MII 的载波信号进入无效状态，MAC 就会启动其 IFG 计数器。在编程的 IFG 值的末尾，MAC 将以全双工模式使能发送。在半双工模式下，当 IFG 配置为 96 个位时间时，MAC 将遵循 IEEE 802.3 规范第 4.2.3.2.1 节中指定的顺从规则。如果在 IFG 间隔的前三分之二时间内（对于所有 IFG 值都为 64 位时间）检测到载波，MAC 将复位其 IFG 计数器。如果在 IFG 间隔的后三分之一时间内检测到载波，MAC 将继续执行 IFG 计数并在 IFG 间隔结束后使能发送器。MAC 在半双工模式下工作时，实施截断二进制指数回退算法。

发送流量控制

在全双工模式下，当发送流量控制使能位（ETH_MACFCR 中的 TFE 位）置 1 时，MAC 将生成暂停帧并根据需要发送暂停帧。暂停帧与计算出的 CRC 附加在一起并发送。可以通过两种方式启动暂停帧的生成。

当应用程序将 ETH_MACFCR 寄存器中的 FCB 位置 1 或接收 FIFO 已满（数据包缓冲区）时，将发送暂停帧。

- 如果应用程序已通过将 ETH_MACFCR 寄存器中的 FCB 位置 1 来请求流量控制，则 MAC 将生成并发送单个暂停帧。生成的帧中的暂停时间值为 ETH_MACFCR 中编程的暂停时间值。要在先前发送的暂停帧中指定的时间之前延长或结束暂停时间，应用程序必须以适当的值编程暂停时间值（ETH_MACFCR 寄存器中的 PT），然后请求另一次暂停帧发送。
- 如果接收 FIFO 填满时应用程序已请求流量控制，则 MAC 将生成并发送暂停帧。生成的帧中的暂停时间值为 ETH_MACFCR 中编程的暂停时间值。如果在此暂停时间结束前，接收 FIFO 在可配置的时隙数（ETH_MACFCR 中的 PLT 位）期间保持填满状态，将发送第二个暂停帧。只要接收 FIFO 保持填满状态，该过程将一直重复下去。如果在采样时间之前不再满足此条件，MAC 将发送一个暂停时间为零的暂停帧，向远程端表明接收缓冲区已准备好接收新数据帧。

单数据包发送操作

发送操作的常规事件序列如下：

1. 如果系统有要传输的数据，则 DMA 控制器将通过 AHB 主接口从存储器中获取这些数据并将其转发给 FIFO。它将继续接收数据直到传输帧结束。
2. 当超过阈值或 FIFO 接收到完整的数据包时，直到将完整的数据包传输到 MAC 为止。DMA 继续从 FIFO 传输数据，数据会被弹出并被传输到 MAC 内核。帧传输完成后，来自 MAC 的状态将通知 DMA 控制器。

发送操作——缓冲区中有两个数据包

1. 由于 DMA 必须先更新描述符状态才能将其释放给主机，因此一个发送 FIFO 内最多只能有两个帧。仅当 OSF（对第二个帧起作用）位置 1 时，DMA 才会获取第二个帧并将其置于 FIFO 中。如果此位未置 1，则仅当 MAC 完全处理该帧且 DMA 已释放描述符后，DMA 才会从存储器获取下一帧。
2. 如果 OSF 已置 1，则 DMA 将第一个帧传输到 FIFO 后将立即开始获取第二个帧。不会等待状态更新。同时，在发送第一个帧时，FIFO 接收到第二个帧。第一帧的数据传输完成后，来自 MAC 的状态会通知 DMA。如果 DMA 已向 FIFO 发送第二个数据包，则第二次发送必须等待第一个数据包的状态，才能继续下一帧。

冲突期间的重新发送

在半双工模式下，向 MAC 传输帧时，可能在 MAC 线接口上发生冲突事件。MAC 甚至会在接收到帧结束之前就给出状态来指示重试。然后将使能重新发送并再次将帧从 FIFO 中弹出。当超过 96 个字节弹向 MAC 内核后，FIFO 控制器将释放该空间，使 DMA 可推入更多数据。这意味着超过阈值后或 MAC 内核指示延迟冲突事件时，无法重新发送。

发送 FIFO 刷新操作

MAC 为软件提供了一个控制权，使其可通过操作模式寄存器中的位 20 来清空发送 FIFO。清空操作是立即操作，即使 Tx FIFO 正在向 MAC 内核传输帧，Tx FIFO 和相应的指针也会清零到初始状态。这将导致 MAC 发送器中生成下溢事件，并且帧发送将中止。此类帧的状态将同时标记下溢和帧清空事件（TDES0 位 13 和 1）。清空操作期间，没有数据从应用程序 (DMA) 传输到 FIFO。根据清空的帧数（包括局部帧），将相应数量的传输发送状态字传输到应用程序。完全清空的帧的帧清空状态位 (TDES0 13) 将置 1。当应用程序 (DMA) 接受所有已清空的帧的状态字后，清空操作完成。发送 FIFO 清空控制寄存器位随后将清零。此时，将接受来自应用程序 (DMA) 的新帧。清空操作完成后，将丢弃所有为发送提交的数据，除非数据以 SOF 标记开头。

发送状态字

在向 MAC 内核传输以太网帧结束时以及内核完成帧的发送后，发送状态将提供给应用程序。发送状态的详细说明与 TDES0 中的位 [23:0] 相同。如果使能 IEEE 1588 时间戳功能，将返回特定帧的 64 位时间戳以及发送状态。

发送校验和减荷

通信协议（例如 TCP 和 UDP）将实施校验和字段，这有助于确定通过网络发送的数据的完整性。由于以太网最广泛的用途是通过 IP 数据报封装 TCP 和 UDP，因此以太网控制器具有发送校验和减荷功能，该功能支持校验和计算、发送路径中的校验和插入以及接收路径中的错误检测。本节将介绍发送帧的校验和减荷功能的操作。

注意： *通过完整的帧来计算 TCP、UDP 或 ICMP 的校验和，然后将其插入相应的报头字段。由于此要求，仅当发送 FIFO 配置为存储转发模式（即，ETH_ETH_DMAOMR 寄存器中的 TSF 位置 1）时，才使能此功能。如果内核配置为阈值（直通）模式，则将绕过发送校验和减荷。*

必须确保发送 FIFO 足够深，使其能存储一个完整的帧，以便将该帧传输到 MAC 内核发送器。如果 FIFO 的深度小于输入以太网帧的太小，则将绕过有效负载 (TCP/UDP/ICMP) 校验和插入功能，并且仅修改帧的 IPv4 报头校验和，即使在存储转发模式下也是如此。

发送校验和减荷支持两种类型的校验和计算和插入。可通过将 CIC 位（TDES1 中的位 28:27，在*第 868 页的TDES1：发送描述符字 1*中介绍）置 1 来控制每个帧的校验和。

有关 IPv4、TCP、UDP、ICMP、IPv6 和 ICMPv6 数据包头规范的信息，请分别参见 IETF 规范 RFC 791、RFC 793、RFC 768、RFC 792、RFC 2460 和 RFC 4443。

- IP 报头校验和

  在 IPv4 数据报中，报头字段的完整性由 16 位头校验和字段（IPv4 数据报的第十一个字节和第十二个字节）指示。当以太网帧的类型字段值为 0x0800 且 IP 数据报的版本字段值为 0x4 时，校验和减荷将检测到 IPv4 数据报。计算期间，将忽略输入帧的校验和字段并将其替换为计算出的值。IPv6 报头没有校验和字段；因此，校验和减荷不修改 IPv6 报头字段。此 IP 报头校验和计算的结果由发送状态中的 IP 报头错误状态位（位 16）指示。只要以太网类型字段的值和 IP 报头版本字段的值不一致，或当以太网帧没有足够的数据（如 IP 报头长度字段所指示）时，此状态位将置 1。换言之，在以下情况下发生 IP 报头错误时，此位将置 1：

  a) 对于 IPv4 数据报：
  – 接收的以太网类型为 0x0800，但 IP 报头的版本字段不等于 0x4
  – IPv4 报头长度字段指示小于 0x5（20 字节）的值
  – 总的帧长度小于 IPv4 报头长度字段给定的值

  b) 对于 IPv6 数据报：
  – 以太网类型为 0x86DD，但 IP 报头版本字段不等于 0x6
  – 帧在 IPv6 报头（40 字节）之前结束，或已完全接收到扩展报头（如扩展报头中相应的报头长度字段中给定）。如果以太网类型字段指示 IPv4 有效负载，即使校验和减荷检测到 IP 报头错误，也会插入 IPv4 报头校验和。

- TCP/UDP/ICMP 校验和

  TCP/UDP/ICMP 校验和对 IPv4 或 IPv6 报头（包括扩展报头）进行处理，并确定封装的有效负载是 TCP、UDP 还是 ICMP。

  请注意：

  a) 对于非 TCP、非 UDP 或非 ICMP/ICMPv6 的有效负载，将绕过此校验和且不会对帧进行修改。

  b) 将绕过分段的 IP 帧（IPv4 或 IPv6）、带安全功能（例如，验证报头或封装的安全有效负载）的 IP 帧和带路由报头的 IPv6 帧，并且校验和不对这些帧进行处理。

计算 TCP、UDP 或 ICMP 有效负载的校验和，然后将其插入报头中相应的字段。它可工作在以下两种模式：

– 在第一种模式下，TCP、UDP 或 ICMPv6 伪报头并未包含在校验和计算中，并假定其存在于输入帧的校验和字段中。校验和字段包含在校验和计算中，然后替换为最终计算出的校验和。

– 在第二种模式下，将忽略校验和字段，TCP、UDP 或 ICMPv6 伪报头数据包含在校验和计算中，并使用最终计算出的值覆盖校验和字段。

请注意：对于 ICMP-over-IPv4 数据包，由于没有为其定义伪报头，因此在这两种模式下 ICMP 数据包中的校验和字段都必须始终为 0x0000。如果不等于 0x0000，可能向数据包插入不正确的校验和。

此操作的结果由发送状态向量中的有效负载校验和错误状态位（位 12）指示。当检测到下列情况之一时，有效负载校验和错误状态位置 1：

– 在存储转发模式下，帧已转发到 MAC 发送器，但帧结束未写入 FIFO

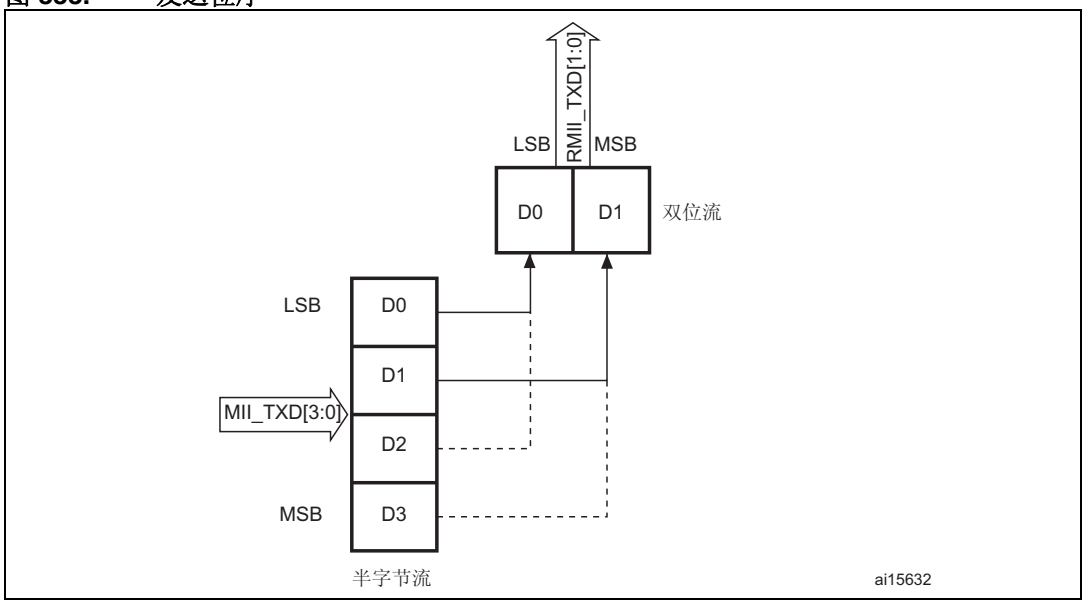
– 在接收到 IP 报头中的有效负载长度字段指示的字节数前，数据包已结束。

当数据包的长度大于指示的有效负载长度时，将字节作为填充字节忽略，并且不报告错误。检测到第一种类型的错误时，不修改 TCP、UDP 或 ICMP 报头。对于第二种错误类型，计算的校验和仍将插入相应的报头字段。

MII/RMII 发送位序

来自 MII 的每个半字节都在 RMII 上发送，一次发送双位，双位的发送顺序如*图 333* 所示。首先发送位序较低的位（D1 和 D0），再发送位序较高的位（D2 和 D3）。

图 333. 发送位序



MII/RMII 发送时序图

图 334. 无冲突发送

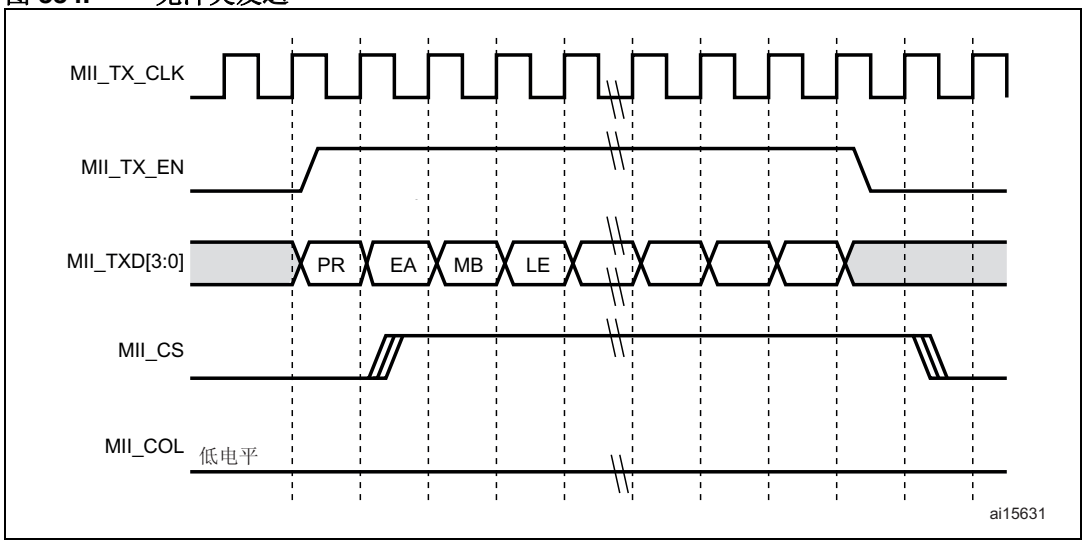


图 335. 有冲突发送

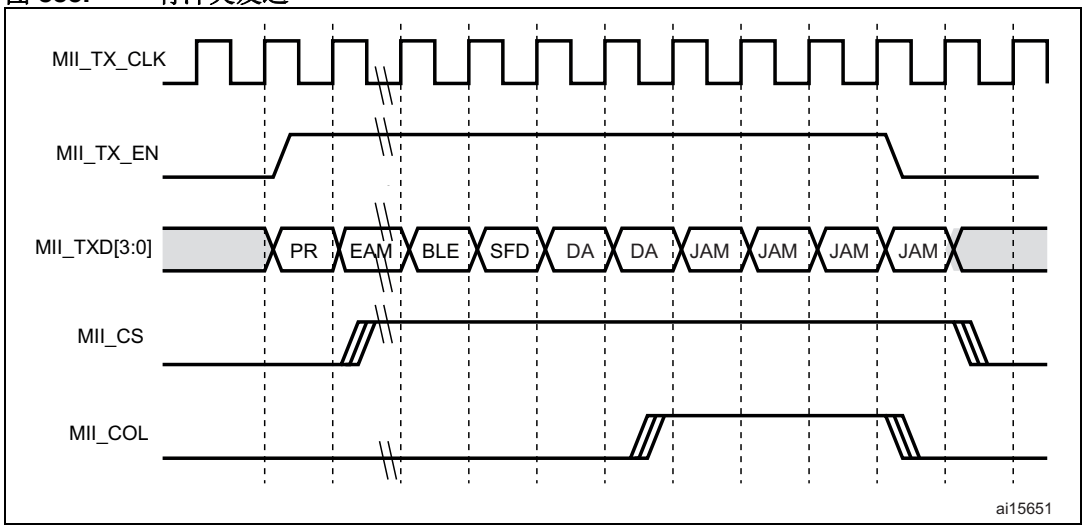
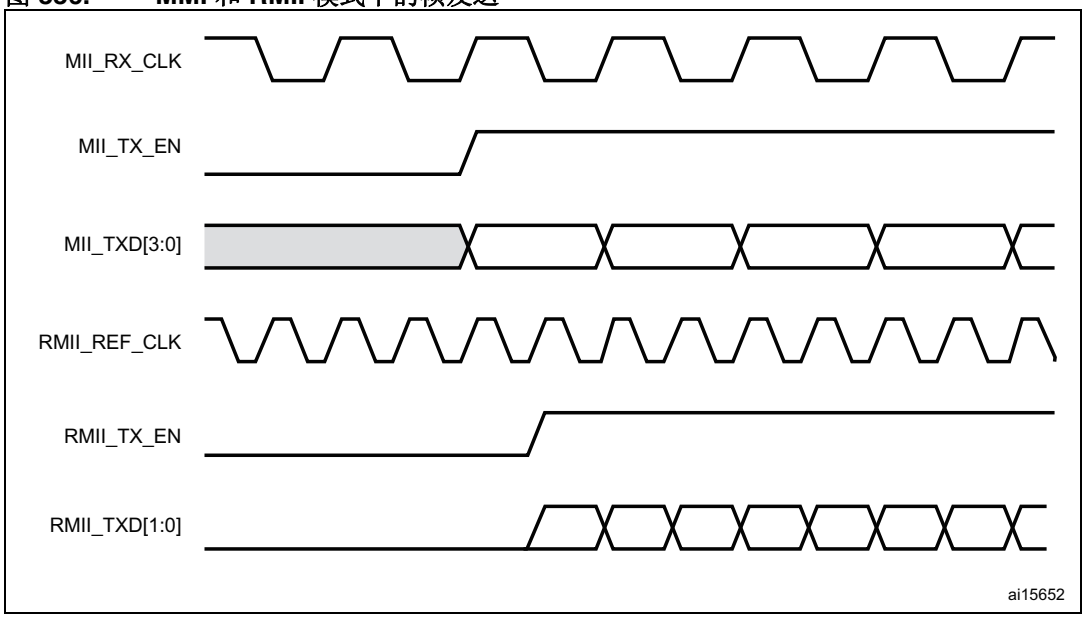


图 336 显示了 MII 和 RMII 中的帧发送。

图 336. MMI 和 RMII 模式下的帧发送



29.5.3 MAC 帧接收

MAC 接收的帧将推入 Rx FIFO。此 FIFO 的状态（填充级别）一旦超过配置的接收阈值（ETH_DMAOMR 寄存器中的 RTC），就会将其指示给 DMA，这样 DMA 可向 AHB 接口发起预配置的突发传输。

在默认直通模式下，当 FIFO 接收到 64 个字节（使用 ETH_DMAOMR 寄存器中的 RTC 位配置）或完整的数据包时，数据将弹出，其可用性将通知给 DMA。DMA 向 AHB 接口发起传输后，数据传输将从 FIFO 持续进行，直到传输完整个数据包。完成 EOF 帧的传输后，状态字将弹出并发送到 DMA 控制器。

在 Rx FIFO 存储转发模式（通过 ETH_DMAOMR 寄存器中的 RSF 位配置）下，仅在帧完全写入接收 FIFO 后才可读出帧。在此模式下，将丢弃所有错误帧（如果内核配置为执行此操作），这样只会读出有效帧并将其转发到应用程序。在直通模式下，某些错误帧不会被丢弃，因为在帧结束时接收到错误状态，而此时已从 FIFO 读出帧开始。

当 MAC 在 MII 上检测到 SFD 时，将启动接收操作。MAC 内核将去除报头和 SFD，然后再继续处理帧。检查报头字段以进行过滤，FCS 字段用于验证帧的 CRC。如果帧未通过地址滤波器，则在内核中丢弃该帧。

接收协议

去除接收的帧的报头和 SFD。检测到 SFD 后，MAC 开始向接收 FIFO 发送以太网帧数据，从 SFD 后面的第一个字节（目标地址）开始发送。如果使能 IEEE 1588 时间戳功能，则在 MII 上检测到任何帧的 SFD 时，都将获取系统时间的快照。除非 MAC 滤出并丢弃帧，否则此时间戳将传递给应用程序。

如果接收的帧长度/类型字段小于 0x600 并且为 MAC 编程了自动去除 CRC/pad 选项，则 MAC 将向接收 FIFO 发送帧数据（数据量不超过长度/类型字段中指定的数量），然后开始丢弃字节（包括 FCS 字段）。如果长度/类型字段大于或等于 0x600，则不管编程的自动 CRC 去除选项的值如何，MAC 都会向 Rx FIFO 发送所有接收到的以太网帧数据。默认情况下，使能 MAC 看门狗定时器，即，超过 2048 个字节（DA + SA + LT + 数据 + pad + FCS）的帧会被切断。可通过对 MAC 配置寄存器中的看门狗禁止 (WD) 位编程来禁止此功能。但是，即使禁止看门狗定时器，仍将切断大于 16 KB 的帧并给出看门狗超时状态。

接收 CRC：自动 CRC 和 pad 去除

MAC 将检查接收帧中的任何 CRC 错误。它将计算接收的帧（包括目标地址字段到 FCS 字段）的 32 位 CRC。编码由下面的多项式定义。

$$G(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^{8} + x^{7} + x^{5} + x^{4} + x^{2} + x + 1$$

不管是否自动去除 pad/CRC，MAC 都将接收整个帧来计算所接收帧的 CRC 校验。

接收校验和减荷

对接收的以太网帧中的 IPv4 和 IPv6 帧进行检测和处理，以确保数据完整性。可通过将 ETH_MACCR 寄存器中的 IPCO 位置 1 来使能接收校验和减荷。MAC 通过检查所接收的以太网帧的类型字段中是否存在值 0x0800 或 0x86DD，来识别 IPv4 或 IPv6 帧。此识别方法也适用于带 VLAN 标记的帧。接收校验和减荷将计算 IPv4 报头校验和，并检查其是否与接收的 IPv4 报头校验和匹配。如果指示的有效负载类型（以太网类型字段）与 IP 报头版本之间有任何不匹配，或接收的帧的字节数小于 IPv4 报头的长度字段中指示的数量（IPv4 或 IPv6 报头中的可用字节数少于 20），IP 报头错误位将置 1。接收校验和减荷还可以识别接收的 IP 数据报（IPv4 或 IPv6）中的 TCP、UDP 或 ICMP 有效负载，并正确计算此类有效负载的校验和，如 TCP、UDP 或 ICMP 规范中所定义。它包括用于校验和计算的 TCP/UDP/ICMPv6 伪报头字节，并检查接收的校验和字段是否与计算的值匹配。此操作的结果由接收状态字中的有效负载校验和错误位给出。如果 TCP、UDP 或 ICMP 有效负载的长度与 IP 报头中给出的预期有效负载长度不匹配，此状态位也将置 1。如 *第 838 页的 TCP/UDP/ICMP 校验和*中所述，接收校验和减荷将绕过分段 IP 数据报的有效负载、带安全功能的 IP 数据报、IPv6 路由报头以及除 TCP、UDP 或 ICMP 以外的有效负载。此信息在接收状态中给出（无论是否绕过校验和），如 *RDES0：接收描述符字 0* 一节所述。在此配置中，内核不会向接收的以太网帧附加任何有效负载校验和字节。

如*第 874 页的RDES0：接收描述符字 0*中所述，某些寄存器位的含义会发生变化，如*表 165* 所示。

表 165. 帧状态

位 18：以太网帧	位 27：报头校验和错误	位 28：有效负载校验和错误	帧状态
0	0	0	帧为 IEEE 802.3 帧（长度字段值小于 0x0600）。
1	0	0	IPv4/IPv6 类型帧，在其中未检测到校验和错误。
1	0	1	IPv4/IPv6 类型帧，在其中检测到有效负载校验和错误（参见 PCE 的说明）。
1	1	0	IPv4/IPv6 类型帧，在其中检测到 IP 报头校验和错误（参见 IPCO HCE 的说明）。
1	1	1	IPv4/IPv6 类型帧，在其中检测到 PCE 和 IPCO HCE。
0	0	1	IPv4/IPv6 类型帧，其中不存在 IP HCE，并且由于不支持的有效负载而绕过有效负载检查。
0	1	1	既不是 IPv4 也不是 IPv6 的类型帧（校验和减荷完全绕过校验和的校验）
0	1	0	保留

接收帧控制器

如果复位 MAC CSR 帧过滤寄存器中的 RA 位，则 MAC 将根据目标/源地址执行帧过滤（如果应用程序决定不接收任何不良帧，如矮帧、CRC 错误帧等，则仍需要执行另一等级的过滤）。检测到过滤失败时，帧将被丢弃且不会传输到应用程序。当过滤参数动态改变时，如果 (DA-SA) 过滤失败，则剩余的帧将被丢弃并且接收状态字将立即更新（零帧长度位、CRC 错误位和矮帧错误位将置 1），指示过滤失败。在以太网掉电模式下，所有接收到的帧都将被丢弃且不会转发给应用程序。

接收流量控制

MAC 将检测接收暂停帧并暂停帧发送，暂停时间为接收的暂停帧内指定的延迟（仅限全双工模式）。可通过 ETH_MACFCR 中的 RFCE 位使能或禁止暂停帧检测功能。使能接收流量控制后，将开始监视接收帧的目标地址是否与控制帧的多播地址 (0x0180 C200 0001) 匹配。如果检测到匹配（接收的帧的目标地址与保留的控制帧的目标地址匹配），MAC 将根据 ETH_MACFFR 中的 PCF 位来决定是否将接收的控制帧传输到应用程序。

MAC 还将对接收控制帧的类型、操作码和暂停定时器字段进行解码。如果状态的字节计数指示 64 个字节，并且不存在任何 CRC 错误，则 MAC 发送器将暂停任何数据帧的发送，暂停时间为解码的暂停时间值乘以时隙（对于 10/100 Mb/s 模式，均为 64 字节时间）。同时，如果检测到另一个零暂停时间值的暂停帧，MAC 将复位暂停时间并管理新的暂停请求。

如果接收的控制帧与类型字段 (0x8808)、操作码 (0x00001) 以及字节长度（64 字节）均不匹配，或存在 CRC 错误，则 MAC 不会生成暂停。

如果暂停帧具有多播目标地址，MAC 将根据地址匹配来过滤帧。

对于具有单播目标地址的暂停帧，MAC 将根据 DA 是否与 MAC 地址 0 寄存器的内容匹配以及 ETH_MACFCR 中的 UPDF 位是否置 1（检测具有单播目标地址的暂停帧）来进行过滤。PCF 寄存器位（ETH_MACFFR 中的位 [7:6]）可对控制帧的过滤以及地址过滤进行控制。

接收操作多帧处理

由于接收数据后状态立即可用，因此只要 FIFO 未满，就可以向其中存储帧。

错误处理

如果在从 MAC 接收 EOF 数据之前 Rx FIFO 已满，则将声明上溢并丢弃整个帧，（ETH_DMAMFBOCR 寄存器）中的上溢计数器将递增。由于上溢，状态将指示这是一个部分帧。如果（使用 ETH_DMAOMR 中的 FEF 和 FUGF 位）使能相应功能，Rx FIFO 可过滤错误帧和过小帧。
如果将接收 FIFO 配置为在存储转发模式下工作，则可过滤并丢弃所有错误帧。
在直通模式下，如果在从 Rx FIFO 读取帧的 SOF 时，该帧的状态和长度可用，则可丢弃整个错误帧。DMA 可通过使能接收帧清空位来清空正在从 FIFO 读取的错误帧。然后到应用 (DMA) 的数据传输将停止，其余帧将从内部读取并丢弃。如果可用，则可以启动下一帧传输。

接收状态字

以太网帧接收结束时，MAC 向应用 (DMA) 输出接收状态。接收状态的详细说明与 RDES0 中的位 [31:0] 相同，请参见*第 874 页的RDES0：接收描述符字 0*。

帧长度接口

对于开关应用，应用和 MAC 之间的数据发送和接收以传输完整帧的形式进行。为了向出站端口传输帧，应用层应知道从入站端口接收的帧的长度。在每次帧接收结束时，MAC 内核在状态内提供每个接收的帧的长度。

注意： *对于由于上溢而写入 Rx FIFO 的部分帧，将指定值为 0 的帧长度。*

MII/RMII 接收位序

每个半字节都从接收自 RMII 的双位发送到 MII，半字节发送顺序如*图 337* 所示。先接收位序较低的位（D0 和 D1），再接收位序较高的位（D2 和 D3）。

图 337. 接收位序

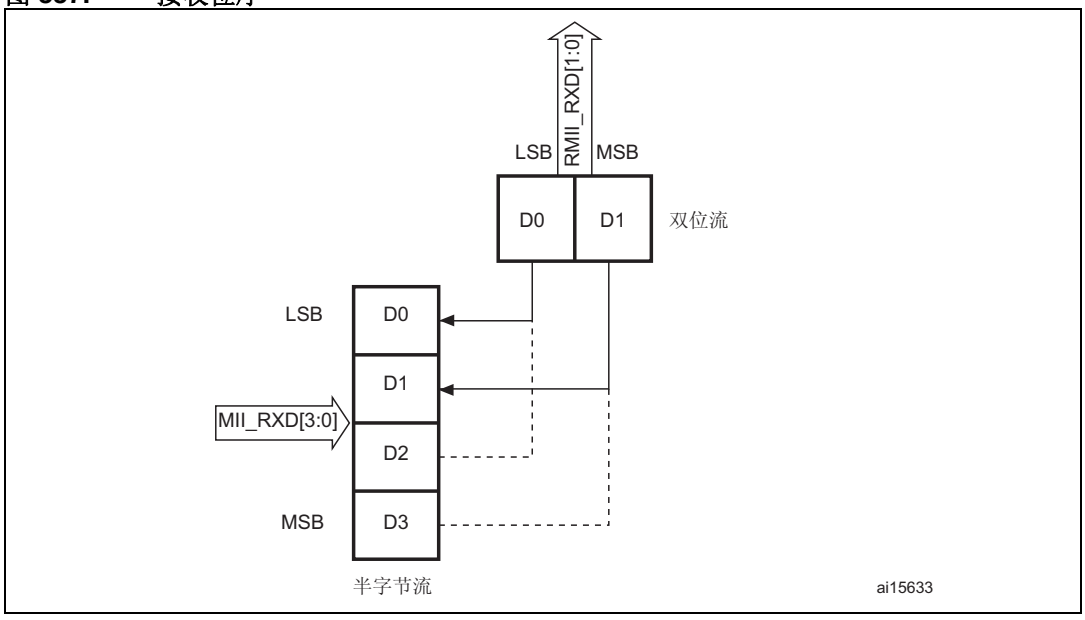


图 338. 无错误接收

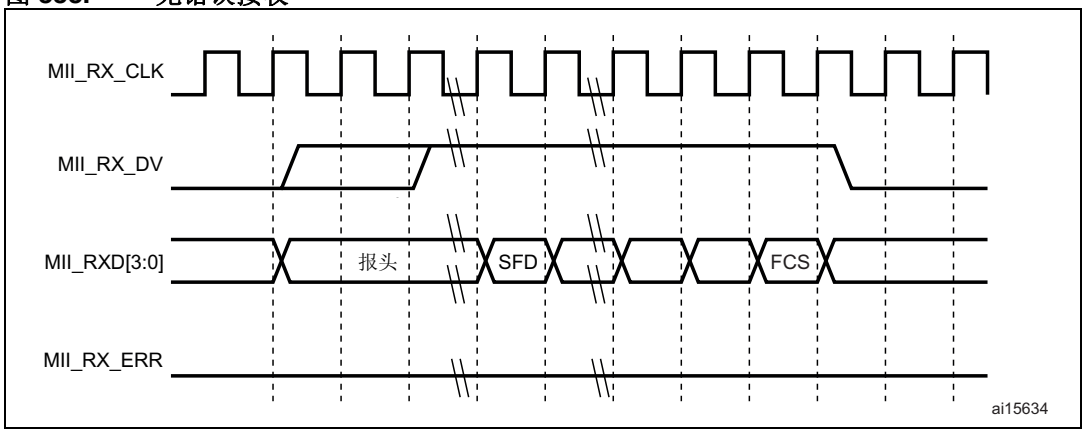


图 339. 有错误接收

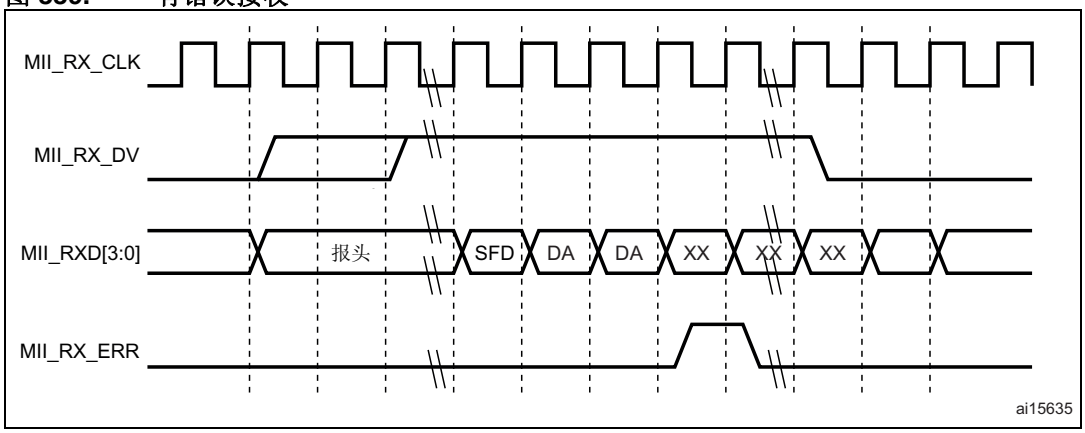
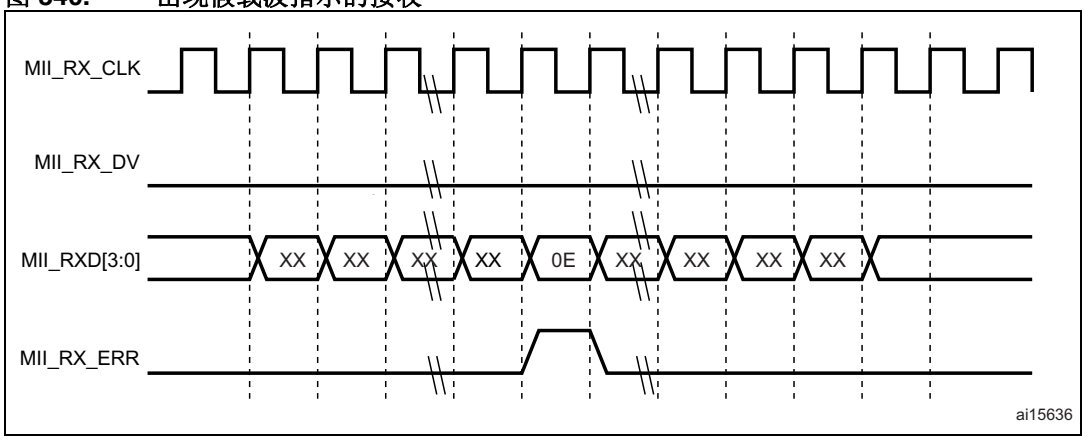


图 340. 出现假载波指示的接收



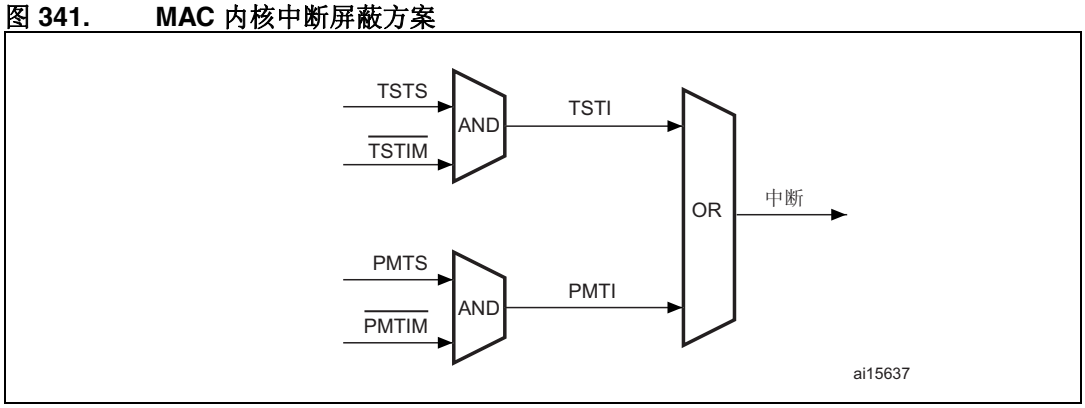
29.5.4 MAC 中断

MAC 内核可由于各种事件而生成中断。

ETH_MACSR 寄存器描述了可导致 MAC 内核生成中断的事件。可通过将中断屏蔽寄存器中相应的屏蔽位置 1 来阻止各事件生成中断。

中断寄存器位仅指示报告事件的模块。必须读取相应的状态寄存器和其它寄存器才能清除中断。例如，如果中断寄存器的位 3 设置为高电平，则指示在掉电模式下接收到魔术数据包或网络唤醒帧。必须读取 ETH_MACPMTCSR 寄存器才能清除此中断事件。

图 341. MAC 内核中断屏蔽方案



29.5.5 MAC 过滤

地址过滤

地址过滤将检查所有接收的帧的目标地址和源地址并相应报告地址过滤状态。地址检查基于应用程序选择的不同参数（帧过滤寄存器）。还可以识别过滤的帧：多播帧或广播帧。

地址过滤使用站的物理 (MAC) 地址和多播散列表进行地址检查。

单播目标地址过滤

MAC 支持多达 4 个用于单播完美过滤的 MAC 地址。如果选择完美过滤（复位帧过滤寄存器中的 HU 位），MAC 会将接收的单播地址的所有 48 位与编程的 MAC 地址进行比较来确定是否匹配。默认情况下，始终使能 MacAddr0，其它地址 MacAddr1–MacAddr3 则通过单独的使能位进行选择。将其它地址 (MacAddr1–MacAddr3) 的各个字节与接收的相应 DA 字节进行比较时，可以将寄存器中相应的屏蔽字节控制位置 1 来屏蔽该字节。这有助于 DA 的组地址过滤。在散列过滤模式（HU 位置 1）下，MAC 将使用 64 位散列表执行对单播地址的不完美过滤。对于散列过滤，MAC 将使用接收的目标地址的 6 个高 CRC（参见下文注释 1）位来索引散列表的内容。值为 000000 时，选取所选寄存器中的位 0；值为 111111 时，选取散列表寄存器中的位 63。如果相应位（由 6 位 CRC 指示）已置 1，将认为单播帧已通过散列过滤，否则认为帧未能通过散列过滤。

注意： *CRC 是使用下列多项式编码的 32 位值（有关详细信息，请参见第 29.5.3 节：MAC 帧接收）：*

$$G(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^{8} + x^{7} + x^{5} + x^{4} + x^{2} + x + 1$$

多播目标地址过滤

可通过将帧过滤寄存器中的 PAM 位置 1，将 MAC 编程为通过所有多播帧。如果 PAM 位复位，MAC 将根据帧过滤寄存器中的 HM 位执行对多播地址的过滤。在完美过滤模式下，将多播地址与编程的 MAC 目标地址寄存器 (1–3) 进行比较。组地址过滤也受到支持。在散列过滤模式下，MAC 将使用 64 位散列表执行不完美过滤。对于散列过滤，MAC 将使用接收的多播地址的 6 个高 CRC（参见下文注释 1）位来索引散列表的内容。值为 000000 时，选取所选寄存器中的位 0；值为 111111 时，选取散列表寄存器中的位 63。如果相应位已置 1，将认为多播帧已通过散列过滤，否则认为帧未能通过散列过滤。

注意： *CRC 是使用下列多项式编码的 32 位值（有关详细信息，请参见第 29.5.3 节：MAC 帧接收）：*

$$G(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$$

散列或完美地址过滤

可通过将帧过滤寄存器中的 HPF 位置 1 并将相应的 HU 或 HM 位置 1，来将 DA 过滤配置为在其 DA 与散列过滤或完美过滤匹配时允许帧通过。此配置适用于单播帧和多播帧。如果 HPF 位复位，则只有一种过滤方式（散列或完美）应用于接收的帧。

广播地址过滤

在默认模式下，MAC 不过滤任何广播帧。但是，如果将帧过滤寄存器中的 BFD 位置 1 来将 MAC 编程为拒绝所有广播帧，则会丢弃任何广播帧。

单播源地址过滤

MAC 还可以根据接收的帧的源地址字段来执行完美过滤。默认情况下，MAC 将 SA 字段与 SA 寄存器中编程的值进行比较。可通过将相应寄存器中的位 30 置 1，来将 MAC 地址寄存器 [1:3] 配置为包含 SA 而不是 DA 进行比较。带 SA 的组地址过滤也受到支持。如果帧过滤寄存器中的 SAF 位置 1，则 MAC 将丢弃未通过 SA 过滤的帧。否则，SA 过滤的结果将通过接收状态字中的状态位给出（参见 *RDES0：接收描述符字 0*）。

SAF 位置 1 时，对 SA 过滤和 DA 过滤的结果进行与运算，以决定是否需要转发帧。这意味着任何一个过滤未通过都将丢弃帧。两个过滤必须都通过，才能将帧转发到应用。

反向过滤操作

对于目标地址和源地址过滤，可在最终输出时选择反转过滤匹配结果。这分别由帧过滤寄存器中的 DAIF 和 SAIF 位控制。DAIF 位同时适用于单播和多播 DA 帧。在此模式下，将反转单播/多播目标地址过滤的结果。类似地，当 SAIF 位置 1 时，将反转单播 SA 过滤的结果。*表 166* 和 *表 167* 按照接收帧的类型汇总了目标地址和源地址过滤。

表 166. 目标地址过滤

帧类型	PM	HPF	HU	DAIF	HM	PAM	DB	DA 过滤操作
广播	1	X	X	X	X	X	X	通过
	0	X	X	X	X	X	0	通过
	0	X	X	X	X	X	1	不通过
单播	1	X	X	X	X	X	X	通过所有帧
	0	X	0	0	X	X	X	完美/组过滤匹配时通过
	0	X	0	1	X	X	X	完美/组过滤匹配时不通过
	0	0	1	0	X	X	X	散列过滤匹配时通过
	0	0	1	1	X	X	X	散列过滤匹配时不通过
	0	1	1	0	X	X	X	散列或完美/组过滤匹配时通过
	0	1	1	1	X	X	X	散列或完美/组过滤匹配时不通过
多播	1	X	X	X	X	X	X	通过所有帧
	X	X	X	X	X	1	X	通过所有帧
	0	X	X	0	0	0	X	完美/组过滤匹配时通过，并在 PCF = 0x 时丢弃暂停控制帧
	0	0	X	0	1	0	X	散列过滤匹配时通过，并在 PCF = 0x 时丢弃暂停控制帧
	0	1	X	0	1	0	X	散列或完美/组过滤匹配时通过，并在 PCF = 0x 时丢弃暂停控制帧
	0	X	X	1	0	0	X	完美/组过滤匹配时不通过，并在 PCF = 0x 时丢弃暂停控制帧
	0	0	X	1	1	0	X	散列过滤匹配时不通过，并在 PCF = 0x 时丢弃暂停控制帧
	0	1	X	1	1	0	X	散列或完美/组过滤匹配时不通过，并在 PCF = 0x 时丢弃暂停控制帧

表 167. 源地址过滤

帧类型	PM	SAIF	SAF	SA 过滤操作
单播	1	X	X	通过所有帧
	0	0	0	完美/组过滤匹配时通过，但不丢弃未通过的帧
	0	1	0	完美/组过滤匹配时不通过，但不丢弃帧
	0	0	1	完美/组过滤器匹配时通过并将不通过的帧丢弃
	0	1	1	完美/组过滤器匹配时不通过并将不通过的帧丢弃

29.5.6 MAC 回送模式

MAC 支持对发送到其接收器的帧进行回送。默认情况下，禁止 MAC 回送功能，可通过编程 MAC ETH_MACCR 寄存器中的 Loopback 位使能该功能。

29.5.7 MAC 管理计数器：MMC

MAC 管理计数器 (MMC) 保持一组寄存器来收集有关已接收帧和已发送帧的统计信息。其中包括一个用于控制各寄存器行为的控制寄存器、两个包含当前中断状态的 32 位寄存器（接收和发送），以及两个包含中断寄存器掩码的 32 位寄存器（接收和发送）。这些寄存器均可从应用程序中访问。每个寄存器均为 32 位宽。

*第 29.8 节：以太网寄存器说明*介绍了各个计数器，并列出了各统计计数器的地址。该地址用于对所需发送/接收计数器进行读/写访问。

接收 MMC 计数器针对通过地址过滤的帧进行更新。已丢弃帧的统计信息不会进行更新，除非丢弃的帧为小于 6 字节的矮帧（DA 字节不能完全接收）。

好的发送帧与接收帧

如果帧成功发送，则将发送的帧视为“好帧”。换句话说，如果帧发送过程没有因为以下错误而中止，则认为发送的帧是好帧：

+ Jabber 超时
+ 无载波/载波丢失
+ 延迟冲突
+ 帧下溢
+ 过度延迟
+ 过度冲突

如果不存在以下错误，则认为接收帧是“好帧”：

+ CRC 错误
+ 矮帧（短于 64 字节）
+ 对齐错误（仅限 10/100 Mb/s）
+ 长度错误（仅限非类型帧）
+ 超出范围（仅限非类型帧，超过最大大小）
+ MII_RXER 输入错误

最大帧大小取决于帧类型，如下：

+ 无标记帧的最大大小 = 1518
+ VLAN 帧的最大大小 = 1522

29.5.8 电源管理：PMT

本部分介绍 MAC 支持的电源管理 (PMT) 机制。PMT 支持接收网络（远程）唤醒帧和魔术数据包帧。PMT 可为 MAC 接收到的唤醒帧和魔术数据包帧生成中断。可通过远程唤醒帧使能位以及魔术数据包使能位使能 PMT 模块。这些使能位（WFE 和 MPE）位于 ETH_MACPMTCSR 寄存器中，可由应用编程。在 PMT 中使能掉电模式时，MAC 将丢弃所有接收到的帧并且不会将这些帧转发给应用。仅当接收到魔术数据包或远程唤醒帧且使能相应检测时，MAC 才会退出掉电模式。

远程唤醒帧过滤寄存器

有八个唤醒帧过滤寄存器。要对每个寄存器执行写操作，需要逐个值加载唤醒帧过滤寄存器。连续加载唤醒帧过滤寄存器八次，便可对唤醒帧过滤寄存器加载所需的值。读操作与写操作相同。要读取八个值，必须读唤醒帧过滤寄存器八次后，才能到达最后一个寄存器。每次读/写操作都会将唤醒帧过滤寄存器指向下一个过滤寄存器。

图 342. 唤醒帧过滤寄存器

- 过滤器 i 字节屏蔽

  该寄存器定义过滤器 i（0、1、2 和 3）检测帧的哪些字节来确定帧是否为唤醒帧。MSB（第 31 位）必须为零。位 j [30:0] 为字节屏蔽。如果将字节屏蔽的位 j（字节数）置 1，则传入帧的过滤器 i 偏移 + j 由 CRC 模块处理；否则将忽略过滤器 i 偏移 + j。

- 过滤器 i 命令

  该 4 位命令控制过滤器 i 操作。位 3 指定地址类型，定义模式的目标地址类型。当该位置 1 时，模式只适用于多播帧。当该位复位时，模式只适用于单播帧。位 2 和位 1 为保留位。位 0 为过滤器 i 的使能位；如果位 0 未置 1，则将禁止过滤器 i。

- 过滤器 i 偏移

  该寄存器定义过滤器 i 要检测的帧的偏移（在帧范围内）。该 8 位模式偏移是要检测的过滤器 i 第一个字节的偏移。允许的最小值为 12，表示帧的第 13 个字节（偏移值 0 表示帧的第一个字节）。

- 过滤器 i CRC-16

  该寄存器包含根据模式计算的 CRC_16 值，以及对唤醒过滤寄存器模块编程的字节屏蔽。

远程唤醒帧检测

当 MAC 处于睡眠模式并已使能 ETH_MACPMTCSR 寄存器中的远程唤醒位时，MAC 可在接收到远程唤醒帧后恢复正常工作。应用可通过连续对唤醒帧过滤寄存器地址执行写操作，来写入全部八个唤醒过滤寄存器。应用可通过向 ETH_MACPMTCSR 寄存器中的位 2 写入 1 来使能远程唤醒。PMT 支持四种可编程过滤器，这些过滤器提供不同的接收帧模式。如果传入帧通过过滤命令的地址过滤，并且过滤器 CRC-16 与传入的检测模式相匹配，则可接收唤醒帧。Filter_offset（最小值 12，表示帧的第 13 个字节）确定要检测的帧的偏移。过滤器字节屏蔽确定必须检测帧的哪些字节。必须将字节屏蔽的第 31 位置 0。只需要检查唤醒帧是否存在长度错误、FCS 错误、帧尾错误、MII 错误、冲突，确保其不是矮帧。即使唤醒帧长度超过 512 字节，但只要帧的 CRC 值有效，帧便有效。ETH_MACPMTCSR 寄存器中的唤醒帧检测会针对每个接收到的远程唤醒帧进行更新。此外，还会生成 PMT 中断（如果已使能）来指示已接收到远程唤醒帧。

魔术数据包检测

魔术数据包帧基于一种特殊的方法，这种方法使用 AMD 公司的魔术数据包技术对网络上处于睡眠模式下的器件上电。MAC 接收称为魔术数据包的特定信息包，此信息包的目标地址是网络上的节点。只对目标地址为器件或多播地址的魔术数据包进行检查，以确定这些数据包是否满足唤醒要求。对通过地址过滤（单播或多播）的魔术数据包进行检测，确定其是否符合远程唤醒数据格式，即 6 个字节的数据所有位为全‘1’的数据包加上重复出现 16 次的 MAC 地址。应用通过向 ETH_MACPMTCSR 寄存器中的位 1 写入 1 来使能魔术数据包唤醒。PMT 模块持续监视目标地址为对应于指定魔术数据包模式的节点的每个帧。检查每个接收帧的目标地址和源地址字段之后是否为 0xFFFF FFFF FFFF 形式。之后，PMT 模块检查帧是否存在重复 16 次且没有任何断开或中断的 MAC 地址。如果重复 16 次的地址中存在中断，则会再次扫描传入帧中是否存在 0xFFFF FFFF FFFF 形式。这 16 次重复可位于帧中的任何位置，但必须在同步数据流后面 (0xFFFF FFFF FFFF)。此外，只要检测到重复 16 次的 MAC 地址，器件就可以接收多播帧。如果节点的 MAC 地址为 0x0011 2233 4455，则 MAC 扫描的数据序列为：

目标地址源地址 ……..FFFF FFFF FFFF

0011 2233 4455 0011 2233 4455 0011 2233 4455 0011 2233 4455

0011 2233 4455 0011 2233 4455 0011 2233 4455 0011 2233 4455

0011 2233 4455 0011 2233 4455 0011 2233 4455 0011 2233 4455

0011 2233 4455 0011 2233 4455 0011 2233 4455 0011 2233 4455

…CRC

ETH_MACPMTCSR 寄存器中的魔术数据包检测会针对接收到的魔术数据包进行更新。此外，还会生成 PMT 中断（如果已使能）来指示已接收到魔术数据包。

掉电期间的系统的注意事项

如果使能 EXTI 中断线 19，以太网 PMT 模块能够在系统处于停止模式时检测帧。

MAC 接收器状态机在掉电模式期间应保持使能状态。这意味着，由于 ETH_MACCR 寄存器中的 RE 位与魔术数据包/网络唤醒帧检测有关，因此必须始终将该位置 1。但是，在掉电模式期间，应通过将 ETH_MACCR 寄存器中的 TE 位清零来关闭发送器状态机。此外，在掉电模式期间，应禁止以太网 DMA，原因是此时无需将魔术数据包/网络唤醒帧复制到 SRAM。要禁止以太网 DMA，需将 ETH_DMAOMR 寄存器中的 ST 位和 SR 位（分别对应发送 DMA 和接收 DMA）清零。

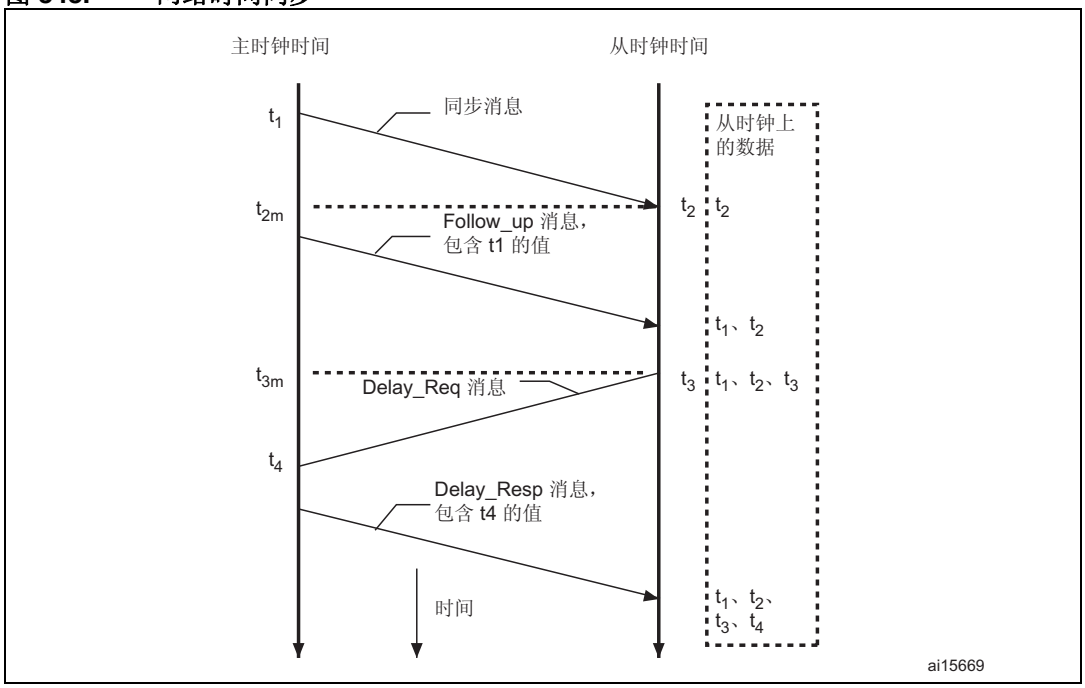
推荐的掉电和唤醒顺序如下：

1. 禁止发送 DMA，并等待所有之前的帧发送完成。接收到发送中断 ETH_DMASR 寄存器 [0] 时，便可检测到这些帧发送。
2. 通过将 ETH_MACCR 配置寄存器中的 RE 位和 TE 位清零来禁止 MAC 发送器和 MAC 接收器。
3. 等待接收 DMA 清空 Rx FIFO 中的所有帧。
4. 禁止接收 DMA。
5. 配置并使能 EXTI 中断线 19，以生成一个事件或中断。
6. 如果将 EXTI 中断线 19 配置为生成中断，还必须正确配置 ETH_WKUP_IRQ 处理程序功能，此功能用于清零 EXTI 中断线 19 的挂起位。
7. 通过将 ETH_MACPMTCSR 寄存器中的 MFE/WFE 位置 1 来使能魔术数据包/网络唤醒帧检测。
8. 通过将 ETH_MACPMTCSR 寄存器中的 PD 位置 1 来使能 MAC 掉电模式。
9. 通过将 ETH_MACCR 寄存器中的 RE 位置 1 来使能 MAC 接收器。
10. 进入系统的停止模式（有关更多详细信息，请参见*第 5.3.4 节：停止模式*）：
11. 接收到有效唤醒帧时，以太网外设会退出掉电模式。
12. 读取 ETH_MACPMTCSR 以清零电源管理事件标志，使能 MAC 发送器状态机并接收和发送 DMA。
13. 配置系统时钟：使能 HSE 并设置时钟。

29.5.9 精密时间协议 (IEEE1588 PTP)

IEEE 1588 标准定义了一种协议，此协议支持使用网络通信、局域计算和分布式对象等技术实现的测量和控制系统中的精密时钟同步。该协议适用于利用支持多播消息传送的局域网（包括但不限于以太网）进行通信的系统。该协议用于对非均匀系统进行同步，这类系统包含固有精度、分辨率和稳定性都不断变化的时钟。该协议支持亚秒范围的系统级同步精度，并且需要极少的网络和本地时钟计算资源。这种基于消息的协议称为精密时间协议 (PTP)，它通过 UDP/IP 传送。系统或网络归类为主节点和从节点，用于分配时序/时钟信息。该协议通过交换 PTP 消息来同步从节点与主节点，*图 343* 中对此有详细说明。

图 343. 网络时间同步



1. 主节点向其所有节点广播 PTP 同步消息。同步消息包含主节点的参考时间信息。消息离开主节点系统的时间为 t_1。对于以太网端口，必须通过 MII 接口来捕获该时间。
2. 从节点接收同步消息并利用其参考时序捕获准确时间 t_2。
3. 主节点随后会向从节点发送一个包含 t_1 信息的 Follow_up 消息以备后用。
4. 从节点向主节点发送一个 Delay_Req 消息，标出该帧离开 MII 的准确时间 t_3。
5. 主节点接收该消息，捕获消息进入其系统的准确时间 t_4。
6. 主节点将 Delay_Resp 消息中的 t_4 信息发送给从节点。
7. 从节点使用 t_1、t_2、t_3 和 t_4 这四个值来同步其本地参考时序与主节点的参考时序。

大多数协议都在 UDP 层之上通过软件实现。但如上所述，要通过 MII 接口来捕获特定 PTP 包进入或离开以太网端口时的准确时间，需要硬件支持。必须捕获该时序信息并将其返回给软件，以实现高精度的 PTP。

参考时序源

根据 IEEE 1588 规范的定义，要获取时间快照，内核需要一个 64 位格式的参考时间（分成两个 32 位通道，高 32 位表示时间的秒数，低 32 位表示时间的纳秒数）。

PTP 参考时钟输入用于在内部生成参考时间（亦称作系统时间）以及捕获时间戳。该参考时钟的频率必须大于等于时间戳计数器的分辨率。主节点与各从节点之间的同步精度目标约为 100 ns。

*系统时间校准方法*一节中介绍了系统时间的生成、更新与修改。

精度取决于 PTP 参考时钟输入周期、振荡器的特性（漂移）以及同步过程的频率。

由于从 Tx 和 Rx 时钟输入域到 PTP 参考时钟域的同步，时间戳锁存值的不确定度为 1 个参考时钟周期。如果加上分辨率导致的不确定度，则会增加一半的时间戳周期。

使用 PTP 功能发送帧

在 MII 上输出帧的 SFD 时，将捕获时间戳。对于需要捕获时间戳的帧，可按帧为单位进行控制。换句话说，可对每个发送帧进行标记以指示是否有必要捕获该帧的时间戳。无需处理发送帧即可识别 PTP 帧。可通过发送描述符中的控制位执行帧控制。捕获到的时间戳返回给应用的方式与提供帧状态的方式相同。时间戳会随帧的发送状态一同发送回相应的发送描述符内，从而自动将时间戳与特定 PTP 帧相连。64 位时间戳信息会写回 TDES2 和 TDES3 字段，其中 TDES2 会保持时间戳的 32 个最低有效位。

使用 PTP 功能接收帧

使能 IEEE 1588 时间戳功能时，以太网 MAC 将捕获 MII 上接收到的所有帧的时间戳。MAC 会在完成帧接收过程时提供时间戳。捕获到的时间戳返回给应用的方式与提供帧状态的方式相同。时间戳会随帧的接收状态一同发送回相应的接收描述符内。64 位时间戳信息会写回 RDES2 和 RDES3 字段，其中 RDES2 会保持时间戳的 32 个最低有效位。

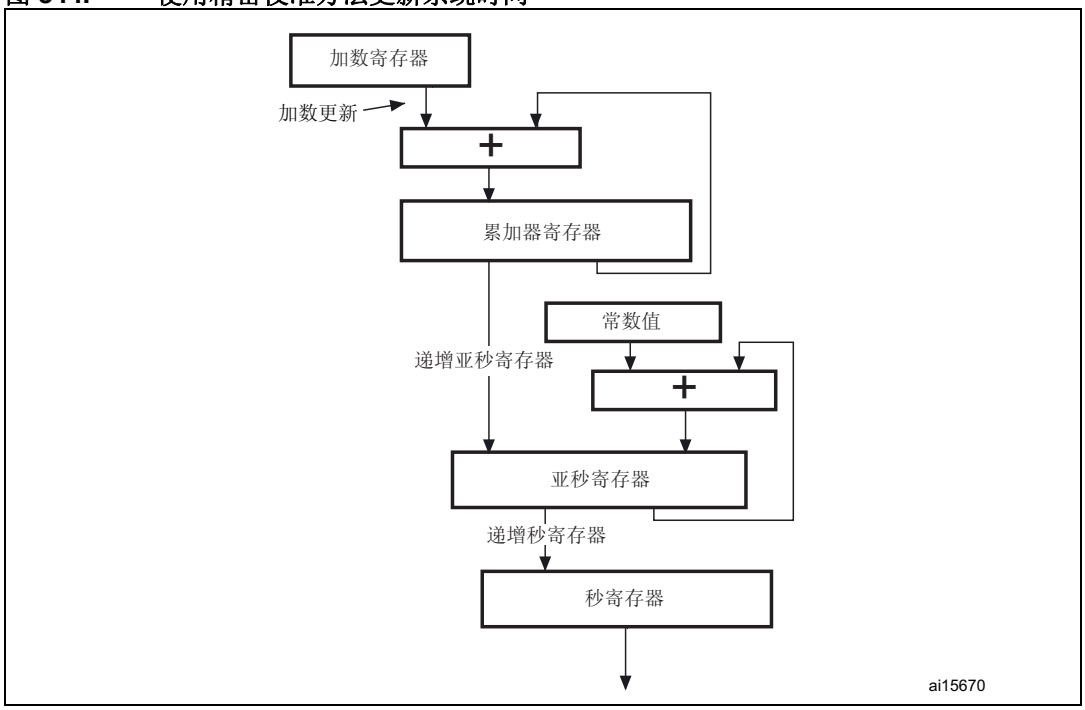
系统时间校准方法

使用 PTP 输入参考时钟 HCLK 更新 64 位 PTP 时间。该 PTP 时间可用作时钟源，以获取 MII 上发送或接收的以太网帧的快照（时间戳）。可使用粗略校准方法或精密校准方法对系统时间定时器进行初始化或校准。

使用粗略校准方法时，初始值或偏移值会写入时间戳更新寄存器（请参见*第 29.8.3 节：第 903 页的IEEE 1588 时间戳寄存器*）。对于初始化，会将时间戳更新寄存器中的值写入系统时间计数器；对于系统时间校准，会将偏移值（时间戳更新寄存器）加到系统时间中或从系统时间中减去。

使用精密校准方法时，从时钟（参考时钟）频率相对于主时钟（如 IEEE 1588 中定义）的偏移会在一段时间内进行校准，而不像粗略校准方法中那样，在单个时钟周期内进行校准。校准时间越长，越有助于保持线性时间，并且不会导致各 PTP 同步消息间隔之间的参考时间发生剧烈变化（或者大型抖动）。在此方法中，会使用一个累加器对加数寄存器中的内容求和，如*图 344* 所示。累加器生成的算数进位将用作使系统时间计数器递增的脉冲。累加器和加数寄存器均为 32 位寄存器。此处，累加器用作高精度频率乘法器或除法器。*图 344* 给出了该算法。

图 344. 使用精密校准方法更新系统时间



系统时间更新逻辑需要使用 50 MHz 的时钟频率，以达到 20 ns 的精度。分频是指参考时钟频率与所需时钟频率的比率。因此，如果我们说参考时钟 (HCLK) 的频率为 66MHz，则通过计算可知分频比为 66 MHz/50 MHz = 1.32。故此，寄存器中设置的默认加数值为 2^{32}/1.32，相当于 0xC1F0 7C1F。

如果参考时钟向下偏移，例如降至 65 MHz，则分频比为 65/50 或 1.3，加数寄存器中要设置的值为 2^{32}/1.30，相当于 0xC4EC 4EC4。如果时钟向上偏移，例如升至 67 MHz，则必须将加数寄存器设置为 0xBF0 B7672。当时钟偏移为零时，应编程的默认加法值为 0xC1F0 7C1F (2^{32}/1.32)。

在*图 344* 中，用于使亚秒寄存器递增的常数值为 0d43。这可使系统时间的精度达到 20 ns（换句话说，增量步长时间为 20 ns）。

软件必须根据同步消息对频率偏移进行计算，并相应更新加数寄存器。首先，使用加数寄存器中的 FreqCompensationValue0 设置从时钟。该值的计算公式如下：

FreqCompensationValue0 = 2^{32} / FreqDivisionRatio

如果起初假定 MasterToSlaveDelay 对于连续的同步消息是相同的，则必须应用下述算法。数个同步周期之后，频率会锁定。从时钟随后可确定 MasterToSlaveDelay 的精确值，并使用新值重新与主时钟同步。

该算法如下：

- 在 MasterSyncTime (n) 时刻，主时钟向从时钟发送同步消息。从时钟在其本地时钟为 SlaveClockTime (n) 时接收到该消息，并用如下公式计算 MasterClockTime (n)：
  MasterClockTime (n) = MasterSyncTime (n) + MasterToSlaveDelay (n)
- 当前同步周期的主时钟计数 MasterClockCount (n) 的计算公式如下：
  MasterClockCount (n) = MasterClockTime (n) – MasterClockTime (n – 1)（假定 MasterToSlaveDelay 对于同步周期 n 和 n – 1 是相同的）
- 当前同步周期的从时钟计数 SlaveClockCount (n) 的计算公式如下：
  SlaveClockCount (n) = SlaveClockTime (n) – SlaveClockTime (n – 1)
- 当前同步周期的主从时钟计数差值 SlaveClockCount (n) 的计算公式如下：
  ClockDiffCount (n) = MasterClockCount (n) – SlaveClockCount (n)
- 从时钟的分频系数 FreqScaleFactor (n) 的计算公式如下：
  FreqScaleFactor (n) = (MasterClockCount (n) + ClockDiffCount (n)) / SlaveClockCount (n)
- 加数寄存器的频率补偿值 FreqCompensationValue (n) 的计算公式如下：
  FreqCompensationValue (n) = FreqScaleFactor (n) × FreqCompensationValue (n – 1)

理论上，该算法可在一个同步周期内实现锁定；但是，由于网络传播延迟和工作条件会不断变化，因此可能需要多个周期。

该算法可进行自校准：如果出于某些原因导致最初通过主时钟设置的从时钟不正确，则该算法会花费更多同步周期将其校准。

系统时间生成初始化的编程步骤

通过将时间戳控制寄存器 (ETH__PTPTSCR) 中的位 0 置 1 来使能时间戳功能。但之后必须对时间戳计数器进行初始化才能启动时间戳操作。适当的顺序如下：

1. 通过将 MACIMR 寄存器中的位 9 置 1 以屏蔽时间戳触发中断。
2. 编程时间戳寄存器位 0 以使能时间戳。
3. 根据 PTP 时钟频率编程亚秒递增计数器。
4. 如果使用精密校准方法，编程时间戳加数寄存器并将时间戳控制寄存器的位 5 置 1（加数寄存器更新）。
5. 轮询时间戳控制寄存器，直到位 5 清零。
6. 要选择精密校准方法（根据需要），编程时间戳控制寄存器位 1。
7. 用适当的时间值编程时间戳高位更新寄存器和时间戳低位更新寄存器。
8. 将时间戳控制寄存器位 2 置 1（时间戳初始化）。
9. 用时间戳更新寄存器中写入的值初始化时间戳计数器后，时间戳计数器便开始运行。
10. 使能 MAC 接收器和发送器以使时间戳功能正常运行。

注意： *如果将 ETH_PTPTSCR 寄存器中的位 0 清零来禁止时间戳操作，必须重复执行以上步骤以重新启动时间戳操作。*

使用粗略校准方法更新系统时间的编程步骤

要在一个过程（粗略校准方法）中同步或更新系统时间，请按照以下步骤执行：

1. 将偏移（正偏移或负偏移）写入时间戳更新高位和低位寄存器。
2. 将时间戳控制寄存器中的位 3 (TSSTU) 置 1。
3. 当 TSSTU 位清零时，时间戳更新寄存器中的值将加到系统时间中或者从系统时间中减去。

使用精密校准方法更新系统时间的编程步骤

要同步或更新系统时间以减少系统时间抖动（精密校准方法），请按照以下步骤执行：

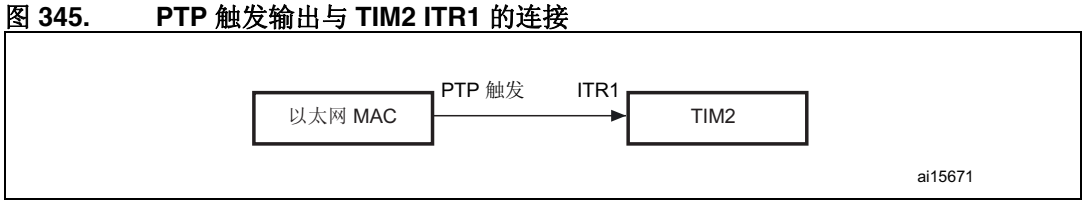
1. 借助*系统时间校准方法*一节中介绍的算法，计算想要增加或减少系统时间增量时采用的速率。
2. 更新时间戳。
3. 等待加数寄存器中的新值生效。这可以通过在系统时间达到目标值后，激活时间戳触发中断来实现。
4. 在目标时间高位和低位寄存器中编程所需目标时间。通过将 ETH_MACIMR 寄存器中的位 9 清零以取消屏蔽时间戳中断。
5. 将时间戳控制寄存器位 4 (TSARU) 置 1。
6. 当该寄存器产生中断时，读取 ETH_MACSR 寄存器。
7. 用旧值重新编程时间戳加数寄存器并再次将 ETH_TPTSCR 位 5 置 1。

与 TIM2 内部相连的 PTP 触发

当系统时间大于目标时间时，MAC 会提供触发中断。使用中断会引起已知延迟，并导致命令执行时间出现不确定性。

为避免这种不确定性，可在系统时间大于目标时间时将 PTP 触发输出信号置为高电平。该信号内部连接到 TIM2 输入触发。通过该信号，由已同步的 PTP 系统时间触发的输入捕获功能、输出比较功能以及定时器的波形均可使用。这时不会引起不确定性，因为定时器的时钟（PCLK1：TIM2 APB1 时钟）以及 PTP 参考时钟 (HCLK) 是同步的。
该 PTP 触发信号会连接到可用软件选择的 TIM2 ITR1 输入。该连接可通过 TIM2 选项寄存器 (TIM2_OR) 中的位 11 和 10 使能。*图 345* 显示了该连接。

图 345. PTP 触发输出与 TIM2 ITR1 的连接



PTP 每秒脉冲数输出信号

该 PTP 脉冲输出用于检查网络中所有节点之间的同步性。要能够测试本地从时钟与主参考时钟之间的差值，可为两个时钟提供每秒脉冲数 (PPS) 输出信号（根据需要，可将该信号连接到示波器）。这样便可测量两个信号间的偏差。PPS 输出的脉冲宽度为 125 ms。

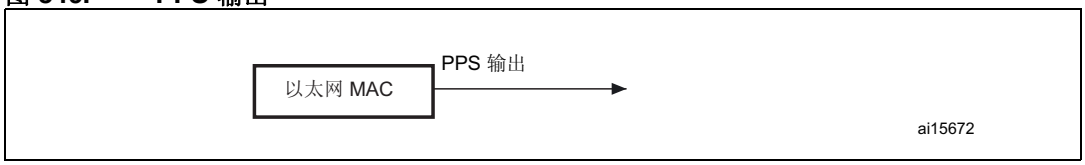
PPS 输出可通过 TIM2 选项寄存器 (TIM2_OR) 中的位 11 和 10 使能。

PPS 输出的默认频率为 1 Hz。可使用 PPSFREQ[3:0]（位于 ETH_PTPPPSCR 中）将 PPS 输出的频率设置为 $2^{PPSFREQ}$ Hz。

如果设置为 1 Hz，则使用二进制翻转（TSSSR=0，ETH_PTPTSCR 寄存器中的位 9）时，PPS 脉冲宽度为 125 ms；使用数字翻转 (TSSSR=1) 时，PPS 脉冲宽度为 100 ms。如果设置为 2 Hz 或更高频率，则使用二进制翻转时，PPS 输出的占空比为 50%。

使用数字翻转 (TSSSR=1) 时，建议不要使用频率为 1 Hz 以外的 PPS 输出，因为波形可能不规则（尽管其平均频率可能在任意一秒窗口期间始终正确）。

图 346. PPS 输出



29.6 以太网功能说明：DMA 控制器操作

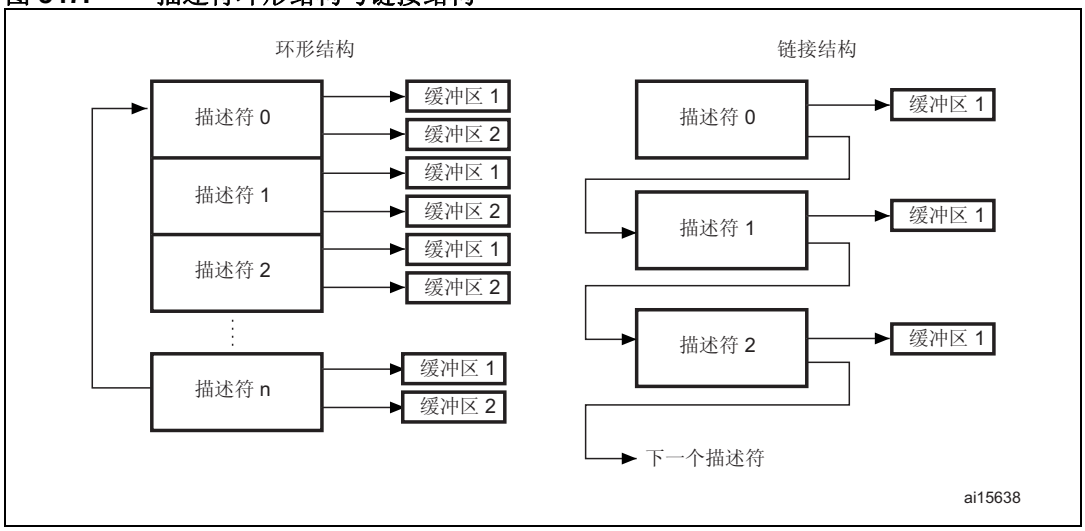
DMA 具有独立的发送和接收引擎以及相应的 CSR（控制和状态寄存器）空间。发送引擎将数据从系统存储器传送到 Tx FIFO，而接收引擎将数据从 Rx FIFO 传送到系统存储器。DMA 可以在 CPU 完全不干预的情况下，通过描述符有效地将数据从源传送到目标。DMA 专为面向包的数据传送（如以太网中的帧）而设计。该控制器经过编程后，可在完成帧发送和接收传送操作时以及其它正常/错误条件下产生 CPU 中断。DMA 和 STM32F4xx 通过以下两种数据结构进行通信：

- 控制和状态寄存器 (CSR)
- 描述符列表和数据缓冲区。

*第 29.8 节*详细介绍了控制和状态寄存器。*还有相应的段落部分*详细介绍了描述符。

DMA 既可将接收到的数据帧传送到 STM32F4xx 存储器中的接收缓冲区，也可以传送 STM32F4xx 存储器的发送缓冲区中的数据帧。位于 STM32F4xx 存储器中的描述符用作指向这些缓冲区的指针。共有两个描述符列表：一个用于接收，一个用于发送。两个列表的基址分别写入 DMA 寄存器 3 和寄存器 4。描述符列表是一种前向链表（无论是隐式还是显式）。最后一个描述符会指回第一个描述符以构成环形结构。可通过配置接收和发送描述符（RDES1[14] 和 TDES0[20]）中链接的第二个地址来完成描述符的显式链接。描述符列表位于主机的物理存储空间。每个描述符最多可指向两个缓冲区。这样便能使用两个物理寻址的缓冲区替代存储器中两个连续的缓冲区。数据缓冲区位于主机的物理存储空间，通常由整个帧或部分帧组成，但不会超过单个帧。缓冲区中仅包含数据。其状态保存在描述符中。数据链接是指跨越多个数据缓冲区的帧。但是单个描述符不能跨越多个帧。检测到帧结束时，DMA 会跳到下一个帧缓冲区。可以使能或禁止数据链接。描述符的环形结构与链接结构如*图 347* 所示。

图 347. 描述符环形结构与链接结构



29.6.1 使用 DMA 进行传送的初始化

MAC 的初始化步骤如下：

1. 对 ETH_DMABMR 执行写操作以设置 STM32F4xx 总线访问参数。
2. 对 ETH_DMAIER 寄存器执行写操作以屏蔽不必要的中断源。
3. 软件驱动器创建发送和接收描述符列表。随后对 ETH_DMARDLAR 和 ETH_DMATDLAR 寄存器执行写操作，为 DMA 提供各列表的起始地址。
4. 对 MAC 寄存器 1、2 和 3 执行写操作以选择所需的过滤选项。
5. 对 MAC ETH_MACCR 寄存器执行写操作以配置和使能发送与接收工作模式。可根据自动协商结果将 PS 位与 DM 位置 1（读取自 PHY）。
6. 对 ETH_DMAOMR 寄存器执行写操作，将位 13 和位 1 置 1 以启动发送和接收。
7. 发送与接收引擎进入运行状态，并尝试从相应描述符列表中获取描述符。这两个引擎随后开始处理接收和发送操作。发送和接收处理过程彼此独立，可单独进行启动或停止。

29.6.2 主机总线突发访问

DMA 会尝试在 AHB 主接口上执行固定长度的突发传送（如果已使用 ETH_DMABMR 中的 FB 位进行相应配置）。突发的最大长度由 PBL 字段 (ETH_DMABMR [13:8]) 指示和限制。对于要读取的 16 个字节，接收和发送描述符始终采用可能的最大突发大小（由 PBL 限制）进行访问。

仅当发送 FIFO 中的空间足以容纳配置的突发或帧结束之前的字节数时（当帧短于配置的突发长度时），发送 DMA 才会启动数据传送。DMA 会向 AHB 主接口指示起始地址和所需的传送次数。当 AHB 接口配置为固定长度突发时，将使用 INCR4、INCR8、INCR16 和单独事务的最佳搭配来传送数据。否则（非固定长度突发）会使用 INCR（未定义长度）与单独事务传送数据。

仅当接收 FIFO 中有足够的数据用于配置的突发时，或者在接收 FIFO 中检测到帧结束时（当帧短于配置的突发长度时），接收 DMA 才会启动数据传送。DMA 会向 AHB 主接口指示起始地址和所需的传送次数。当 AHB 接口配置为固定长度突发时，将使用 INCR4、INCR8、INCR16 和单独事务的最佳搭配来传送数据。如果在 AHB 接口上的固定突发结束前已到达帧结束，将执行空传送以完成固定长度的突发传送。否则（复位 ETH_DMABMR 中的 FB 位）会使用 INCR（未定义长度）和单独事务传送数据。

当 AHB 接口配置为地址对齐的节拍时，两个 DMA 引擎会确保 AHB 启动的第一次突发传送小于或者等于已配置 PBL 的大小。这样，后续的所有节拍都会从与已配置 PBL 对齐的地址开始。由于 AHB 接口多于 INCR16 的传送，DMA 只能对齐节拍最大为 16（对于 PBL > 16）的地址。

29.6.3 主机数据缓冲区对齐

发送和接收数据缓冲区在起始地址对齐方面没有任何限制。在我们的系统（32 位寻址空间）中，缓冲区的起始地址可与四个字节中的任意一个对齐。但是，DMA 始终在地址与总线宽度对齐时启动传输，并且在不需要的字节通道上传输空数据。这通常发生在以太网帧的开始或结束传送期间。

- 缓冲区读操作示例：

  如果发送缓冲区地址为 0x0000 0FF2，并且需要传送 15 个字节，则 DMA 将从地址 0x0000 0FF0 读取 5 个全字，但在将数据传送到发送 FIFO 时，将丢弃或忽略额外的字节（前两个字节）。同样地，还将忽略最后一次传送的最后 3 个字节。DMA 始终可确保向发送 FIFO 传送的是全 32 位数据项，除非是帧结束。

- 缓冲区写操作示例：

  如果接收缓冲区地址为 0x0000 0FF2，并且需要传送接收到的帧的 16 个字节，则 DMA 将从地址 0x0000 0FF0 开始写入 5 个全 32 位数据项。但是，第一次传送的前 2 个字节与第三次传送的最后 2 个字节将包含空数据。

29.6.4 缓冲区大小计算

DMA 不会更新发送和接收描述符中的大小字段。DMA 只更新描述符的状态字段 (xDES0)。必须用驱动程序计算大小。发送 DMA 会向 MAC 内核传送准确的字节数（由 TDES1 中的缓冲区大小字段指示）。如果将描述符标记为第一个描述符（将 TDES0 中的 FS 位置 1），则 DMA 会将缓冲区的第一次传送标记为帧起始。如果将描述符标记为最后一个描述符（TDES0 中的 LS 位），则 DMA 会将数据缓冲区的最后一次传送标记为帧结束。接收 DMA 持续将数据传送至缓冲区，直到缓冲区已满或接收到帧结束为止。当描述符的 FS 位置 1 时，如果未将描述符标记为最后一个描述符（RDES0 中的 LS 位），则该描述符对应的缓冲区会填满，并且缓冲区中的有效数据量将通过缓冲区大小字段减去数据缓冲指针偏移得到的结果表示。当数据缓冲区指针与数据总线宽度对齐时，偏移为零。如果将描述符标记为最后一个描述符，则缓冲区不会填满（由 RDES1 中的缓冲区大小指示）。要计算该缓冲区中最终的有效数据量，驱动程序必须读取帧长度（RDES0[29:16] 中的 FL 位），并减去该帧中先前缓冲区的缓冲区大小之和。接收 DMA 始终使用新的描述符传送下一个帧的帧起始。

注意： *即使当接收缓冲区的起始地址与系统数据总线宽度未对齐时，系统也应分配一个大小与系统总线宽度对齐的接收缓冲区。例如，如果系统分配一个起始地址为 0x1000、大小为 1024 字节 (1 KB) 的接收缓冲区，则软件可将接收描述符中的缓冲区起始地址编程为具有 0x1002 偏移。接收 DMA 将帧写入该缓冲区，其中前两个单元（0x1000 和 0x1001）中为空数据。实际帧从单元 0x1002 开始写入。因此，尽管已将缓冲区大小编程为 1024 字节，但由于存在起始偏移地址，因此该缓冲区的实际有用空间为 1022 字节。*

29.6.5 DMA 仲裁器

DMA 内的仲裁器会在发送和接收通道对 AHB 主接口进行的访问之间进行仲裁。可以使用两类仲裁：循环调度和固定优先级。如果选择循环调度仲裁（复位 ETH_DMABMR 中的 DA 位），仲裁器会在发送和接收 DMA 同时请求访问时，按照 ETH_DMABMR 中的 PM 位设置的比率分配数据总线。当 DA 位置 1 时，接收 DMA 进行数据访问的优先级始终高于发送 DMA。

29.6.6 对 DMA 的错误响应

对于由 DMA 通道发起的任何数据传送，如果从机给出错误响应，则相应 DMA 将停止所有操作并更新状态寄存器（ETH_DMASR 寄存器）中的错误位和致命总线错误位。此外，该 DMA 控制器只能在软复位或硬复位外设以及重新初始化 DMA 之后才能恢复操作。

29.6.7 Tx DMA 配置

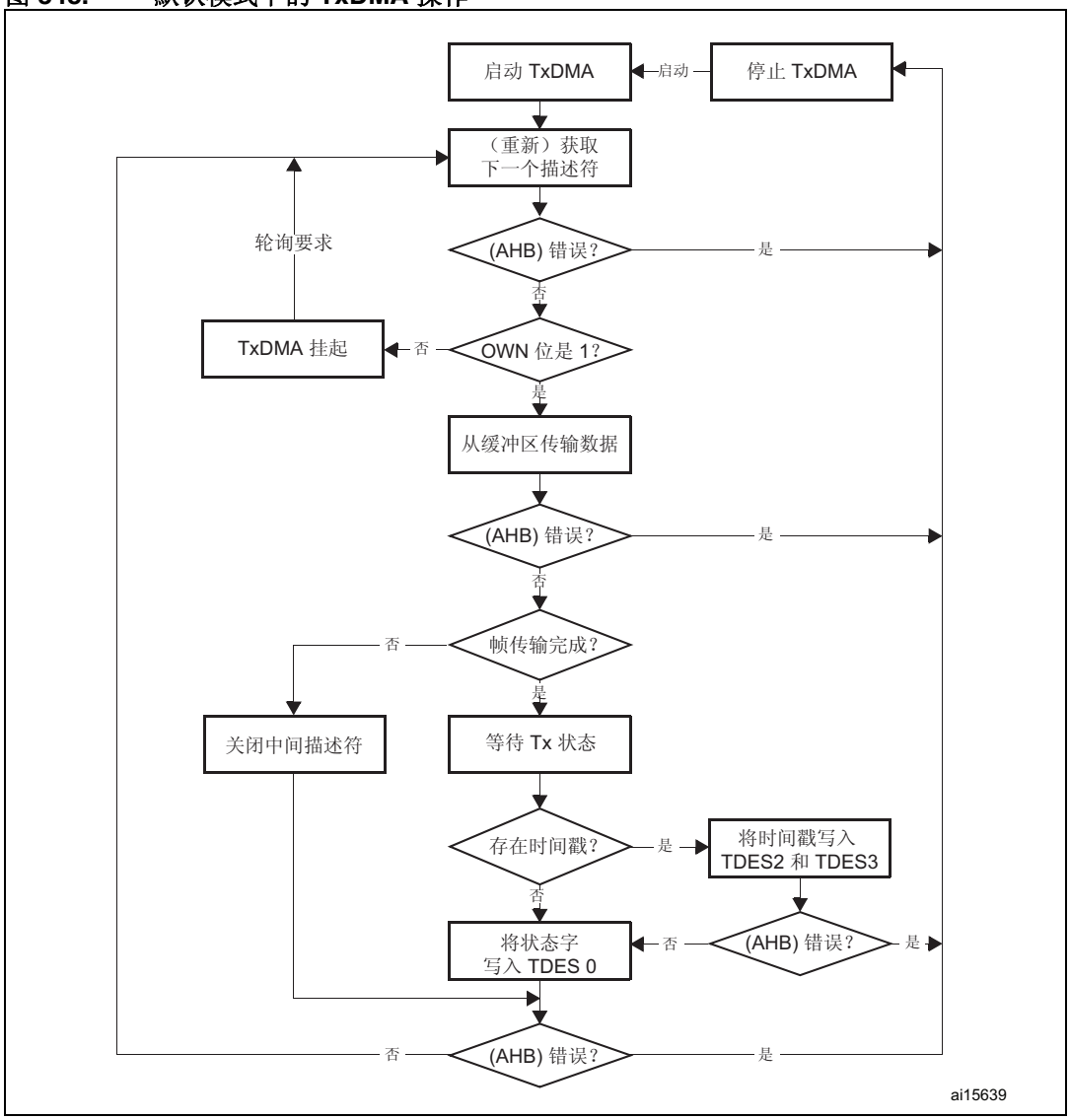
TxDMA 操作：默认（非 OSF）模式

发送 DMA 引擎在默认模式下的操作顺序如下：

1. 在为数据缓冲区设置好相应的以太网帧数据后，用户配置发送描述符 (TDES0-TDES3) 并将 OWN 位 (TDES0[31]) 置 1。
2. ST 位（ETH_DMAOMR 寄存器 [13]）置 1 后，DMA 会立即进入运行状态。
3. 在运行状态下，DMA 会为需要传送的帧轮询发送描述符列表。轮询启动后，DMA 会以连续的描述符环形顺序或链接顺序持续进行。如果 DMA 检测到标记为 CPU 所有的描述符，或者发生错误条件，则会挂起传送并将发送缓冲区不可用位（ETH_DMASR 寄存器 [2]）与正常中断汇总使能位（ETH_DMASR 寄存器 [16]）置 1。发送引擎继续执行步骤 9。
4. 如果获得的描述符标记为 DMA 所有（TDES0[31] 置 1），则 DMA 将根据取得的描述符对发送数据缓冲区地址进行解码。
5. DMA 从 STM32F4xx 存储器中获取发送数据并传送这些数据。
6. 如果以太网帧保存在多个描述符的数据缓冲区中，则 DMA 将关闭中间描述符并获取下一个描述符。重复执行步骤 3、4 和 5，直到完成以太网帧数据的传送。
7. 完成帧传送后，如果已为帧使能 IEEE 1588 时间戳（按发送状态中的指示），则时间戳值将写入包含帧结束缓冲区的发送描述符（TDES2 和 TDES3）随后，状态信息将写入该发送描述符 (TDES0)。由于此步骤中会清零 OWN 位，因此该描述符现在由 CPU 所有。如果没有为该帧使能时间戳，DMA 不会更改 TDES2 和 TDES3 的内容。
8. 发送完在其最后一个描述符中将完成时中断 (TDES1[31]) 置 1 的帧后，发送中断（ETH_DMASR 寄存器 [0]）将置 1。随后，DMA 引擎返回步骤 3。
9. 在挂起状态下，DMA 会在接收到发送轮询要求时尝试重新获取描述符（因此返回步骤 3），并将下溢中断状态位清零。

图 348 给出了默认模式下 TxDMA 发送过程的流程图。

图 348. 默认模式下的 TxDMA 操作



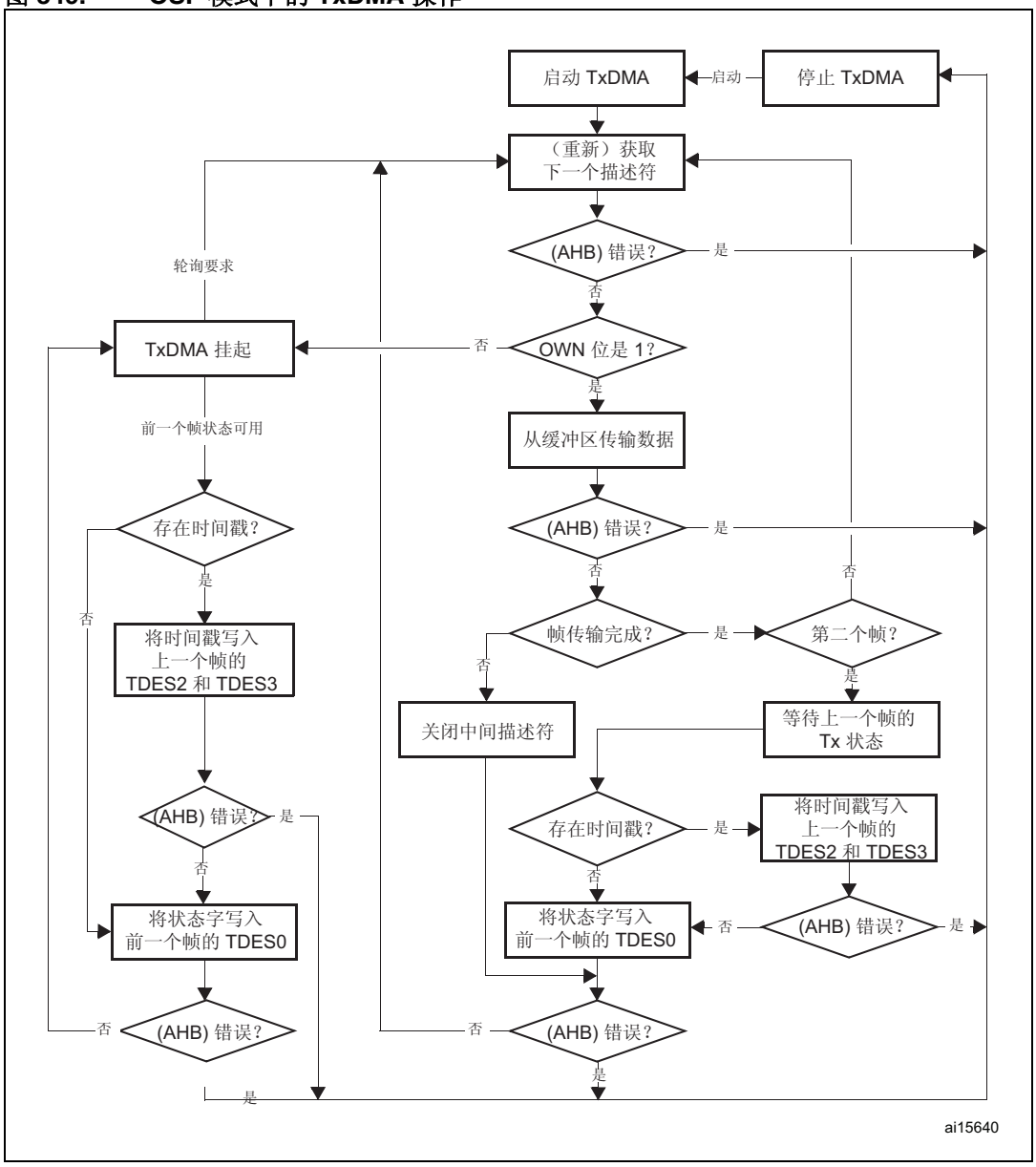
TxDMA 操作：OSF 模式

在运行状态下，发送过程无需关闭第一个帧的状态描述符便可同时获取两个帧（如果 ETH_DMAOMR 寄存器 [2] 中的 OSF 位置 1）。发送过程完成第一个帧的传送后，会立即轮询发第二个帧的发送描述符列表。如果第二个帧有效，发送过程会在写入第一个帧的状态信息前发送此帧。在 OSF 模式下，运行状态发送 DMA 按照以下顺序进行操作：

1. DMA 的操作过程如默认模式下 TxDMA 的步骤 1–6 中所述。
2. 无需关闭前一帧的最后一个描述符，DMA 便可获取下一个描述符。
3. 如果 DMA 拥有所需描述符，便会对该描述符中的发送缓冲区地址进行解码。如果 DMA 未拥有描述符，则会进入挂起模式并跳到步骤 7。
4. DMA 从 STM32F4xx 存储器中取得发送帧并发送该帧，直到帧数据发送结束，如果该帧拆分到多个描述符中，则会同时关闭中间描述符。
5. DMA 等待前一个帧的发送状态和时间戳。当状态可用时，如果捕获到时间戳（由状态位指示），DMA 会将这些时间戳写入 TDES2 和 TDES3。之后，OWN 位清零，DMA 将状态写入相应的 TDES0，进而关闭描述符。如果没有为前一个帧使能时间戳，DMA 不会更改 TDES2 和 TDES3 的内容。
6. 如果已使能时间戳，发送中断将置 1，DMA 获取下一个描述符，然后继续执行步骤 3（状态正常时）。如果上一个发送状态显示下溢错误，则 DMA 会进入挂起模式（步骤 7）。
7. 在挂起模式下，如果 DMA 接收到挂起状态和时间戳，则会将时间戳（如果已为当前帧使能时间戳）写入 TDES2 和 TDES3，随后将状态写入相应的 TDES0。之后，将相关中断置 1 并返回挂起模式。
8. 只有在接收到发送轮询要求（ETH_DMATPDR 寄存器）后，DMA 才会退出挂起模式并进入运行状态（转到步骤 1 或步骤 2，具体取决于挂起状态）。

图 349 显示了 OSF 模式下的基+本流程图。

图 349. OSF 模式下的 TxDMA 操作



发送帧处理

发送 DMA 期望数据缓冲区包含完整的以太网帧，不包括报头、填充字节和 FCS 字段。DA、SA 和类型/长度字段包含有效数据。如果发送描述符指示 MAC 内核必须禁止插入 CRC 或填充字节，则缓冲区必须具有包括 CRC 字节的完整以太网帧（不包括报头）。帧可以采用数据链接形式，也可以跨越多个缓冲区。帧必须由第一个描述符 (TDES0[28]) 和最后一个描述符 (TDES0[29]) 定界。传送启动时，必须将第一个描述符的 TDES0[28] 置 1。之后，帧数据便从存储器缓冲区传送到发送 FIFO。与此同时，如果当前帧的最后一个描述符 (TDES0[29]) 清零，发送过程将尝试获取下一个描述符。发送过程期望 TDES0[28] 在此描述符中清零。如果 TDES0[29] 清零，则指示中间缓冲区。如果 TDES0[29] 置 1，则指示帧的最后一个缓冲区。当帧的最后一个缓冲区完成传送后，DMA 会将最终状态信息写回描述符的发送描述符 0 (TDES0) 字，该描述符在发送描述符 0 (TDES0[29]) 中将末段置 1。此时，如果完成时中断 (TDES0[30]) 置 1，则发送中断（ETH_DMASR 寄存器 [0] 中）将置 1，并获取下一个描述符，然后重复执行该过程。实际的帧传送过程在发送 FIFO 达到可编程的发送阈值时（ETH_DMAOMR 寄存器 [16:14]），或者 FIFO 中包含完整的帧时才会启动。此外，还可以选择存储转发模式（ETH_DMAOMR 寄存器 [21]）。DMA 完成帧的传送后会释放描述符（清零 OWN 位 TDES0[31]）。

发送轮询挂起

可通过以下任意条件暂停发送轮询：

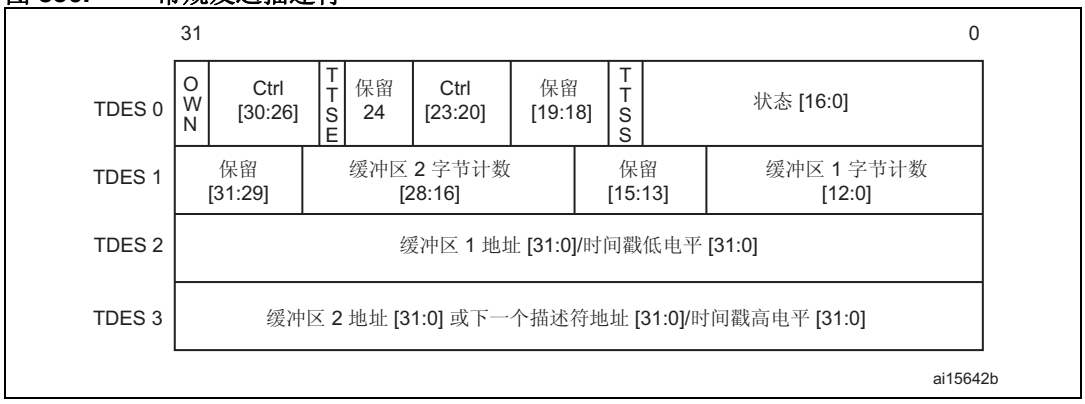
- DMA 检测到 CPU 所有的描述符 (TDES0[31]=0)，并且发送缓冲区不可用标志置 1（ETH_DMASR 寄存器 [2]）。如要恢复，驱动程序必须将描述符的所有权交给 DMA，然后发出轮询查询要求命令。
- 检测到由下溢导致的传送错误时，将中止帧的传送。相应的发送描述符 0 (TDES0) 位将置 1。如果发生第二个条件，异常中断汇总位（ETH_DMASR 寄存器 [15] 中）与发送下溢位（ETH_DMASR 寄存器 [5] 中）都将置 1，并且信息将写入发送描述符 0，从而导致挂起。如果 DMA 由于第一个条件而进入挂起状态，则正常中断汇总位（ETH_DMASR 寄存器 [16]）与发送缓冲区不可用位（ETH_DMASR 寄存器 [2]）均置 1。两种情况下，发送列表中的位置都会保留。保留的位置是 DMA 关闭的最后一个描述符后面的描述符位置。纠正挂起原因后，驱动程序必须明确发出发送轮询要求命令。

常规 Tx DMA 描述符

常规发送描述符结构由四个 32 位字组成，如*图 350* 所示。下文给出了 TDES0、TDES1、TDES2 和 TDES3 的位说明。

请注意，如果时间戳已激活（ETH_PTPTSCR 位 0，TSE=1）或 IPv4 校验和减荷已激活（ETH_MACCR 位 10，IPCO=1），则必须使用增强的描述符。

图 350. 常规发送描述符



- **TDES0：发送描述符字 0**

  应用软件必须在描述符初始化期间编程控制位 [30:26]+[23:20] 以及 OWN 位 [31]。当 DMA 更新描述符（或将其写回）时，将复位所有控制位与 OWN 位，并且只报告状态信息。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OWN	IC	LS	FS	DC	DP	TTSE	Res	CIC		TER	TCH	Res.		TTSS	IHE	ES	JT	FF	IPE	LCA	NC	LCO	EC	VF	CC				ED	UF	DB
rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31 **OWN**：所有关系位 (Own bit)

该位置 1 时，表示描述符为 DMA 所有。该位复位时，表示描述符为 CPU 所有。DMA 可在完成帧发送时或在描述符分配的缓冲区全部读取完成后将该位清零。将属于同一个帧的所有后续描述符置 1 后，必须将帧的第一个描述符的所有位置 1。

位 30 **IC**：完成时中断 (Interrupt on completion)

该位置 1 时，当前帧发送完毕后，发送中断位（寄存器 5[0]）置 1。

位 29 **LS**：末段 (Last segment)

该位置 1 时，指示缓冲区中包含帧的末段。

位 28 **FS**：首段 (First segment)

该位置 1 时，指示缓冲区中包含帧的首段。

位 27 **DC**：禁止 CRC (Disable CRC)

该位置 1 时，MAC 不会将循环冗余校验 (CRC) 附加到所发送帧的末尾。该位仅在首段位 (TDES0[28]) 置 1 的情况下有效。

位 26 **DP**：禁止 pad (Disable pad)

该位置 1 时，MAC 不会自动为不足 64 字节的帧添加补位项。该位复位时，DMA 会自动为不足 64 字节的帧添加补位项和 CRC，是否添加 CRC 字段与 DC (TDES0[27]) 位状态无关。该位仅在首段位 (TDES0[28]) 置 1 的情况下有效。

位 25 **TTSE**：发送时间戳使能 (Transmit time stamp enable)

TTSE 位置 1 且 TSE 置 1（ETH_PTPTSCR 位为 0）时，将针对描述符所描述的发送帧激活 IEEE1588 硬件时间戳功能。该字段仅在首段控制位 (TDES0[28]) 置 1 的情况下有效。

位 24 保留，必须保持复位值。

位 23:22 **CIC**：校验和插入控制 (Checksum insertion control)

这些位控制校验和的计算与插入。位编码如下所示：

00：禁止插入校验和

01：仅使能 IP 报头校验和的计算与插入

10：使能 IP 报头校验和以及有效负载校验和的计算与插入，但不会在硬件中计算伪报头校验和

11：使能 IP 报头校验和以及有效负载校验和的计算与插入，并在硬件中计算伪报头校验和。

位 21 **TER**：环发送结束 (Transmit end of ring)

该位置 1 时，表示描述符列表已到达其最后一个描述符。DMA 会返回描述符列表的基址，并形成一个描述符环。

位 20 **TCH**：链接的第二个地址 (Second address chained)

该位置 1 时，表示描述符中的第二个地址是下一个描述符地址，而非第二个缓冲区地址。TDES0[20] 置 1 时，TBS2 (TDES1[28:16]) 为“无关”值。TDES0[21] 优先级高于 TDES0[20]。

位 19:18 保留，必须保持复位值。

位 17 **TTSS**：发送时间戳状态 (Transmit time stamp status)

此字段用作状态位，指示已为所描述的发送帧捕获到时间戳。该位置 1 时，TDES2 和 TDES3 将保存为发送帧捕获到的时间戳值。该字段仅在描述符末段控制位 (TDES0[29]) 置 1 的情况下有效。

注意，使能增强的描述符时（ETH_DMABMR 中的 EDFE=1），TTSS=1 指示 TDES6 和 TDES7 中存有时间戳值。

位 16 **IHE**：IP 报头错误 (IP header error)

该位置 1 时，指示 MAC 发送器在 IP 数据报头中检测到错误。发送器将对照从应用程序接收的报头字节数检查 IPv4 数据包的报头长度，如果出现不匹配，则指示错误状态。对于 IPv6 帧，如果主报头长度不是 40 个字节，则报告一个报头错误。此外，IPv4 或 IPv6 帧的以太网长度/类型字段值必须与随数据包接收的 IP 报头版本匹配。对于 IPv4 帧，如果报头长度字段的值小于 0x5，也会指示一个错误状态。

位 15 **ES**：错误汇总 (Error summary)

指示以下位的逻辑或运算结果：

TDES0[14]：Jabber 超时 (Jabber timeout)
TDES0[13]：帧刷新 (Frame flush)
TDES0[11]：载波丢失 (Loss of carrier)
TDES0[10]：无载波 (No carrier)
TDES0[9]：延迟冲突 (Late collision)
TDES0[8]：过度冲突 (Excessive collision)
TDES0[2]：过度延迟 (Excessive deferral)
TDES0[1]：下溢错误 (Underflow error)
TDES0[16]：IP 报头错误 (IP header error)
TDES0[12]：IP 有效负载错误 (IP payload error)

位 14 **JT**：Jabber 超时 (Jabber timeout)

该位置 1 时，指示 MAC 发送器经历了 jabber 超时。只有在 MAC 配置寄存器的 JD 位未置 1 时，该位才会置 1。

位 13 **FF**：帧已刷新 (Frame flushed)

该位置 1 时，指示 DMA/MTL 已依照 CPU 发出的软件刷新命令刷新帧。

位 12 **IPE**：IP 有效负载错误 (IP payload error)

该位置 1 时，指示 MAC 发送器在 TCP、UDP 或 ICMP IP 数据报有效负载中检测到错误。发送器将对照从应用程序接收的 TCP、UDP 或 ICMP IP 数据包实际字节数检查 IPv4 或 IPv6 报头中接收的有效负载长度，如果出现不匹配，则指示错误状态。

位 11 **LCA**：载波丢失 (Loss of carrier)

该位置 1 时，指示帧发送期间丢失载波（即，帧发送期间有一个或多个发送时钟周期的 MII_CRS 信号无效）。该位仅对在 MAC 处于半双工模式时实现无冲突发送的帧有效。

位 10 **NC**：无载波 (No carrier)

该位置 1 时，指示在发送期间未由 PHY 触发载波侦听信号。

位 9 **LCO**：延迟冲突 (Late collision)

该位置 1 时，指示帧发送过程因冲突窗口（MII 模式下为 64 个字节时间，含报头）后出现冲突而中止。如果下溢错误位置 1，则该位无效。

位 8 **EC**：过度冲突 (Excessive collision)

该位置 1 时，指示尝试发送当前帧时因出现 16 个连续冲突而中止发送。如果 MAC 配置寄存器中的 RD（禁止重试）位置 1，则此位会在出现首个冲突后置 1 并且帧发送过程将中止。

位 7 **VF**：VLAN 帧 (VLAN frame)

该位置 1 时，指示所发送帧为 VLAN 类型的帧。

位 6:3 **CC**：冲突计数 (VLAN frame)

此 4 位计数器值指示发送帧之前出现的冲突个数。过度冲突位 (TDES0[8]) 置 1 时，该计数无效。

位 2 **ED**：过度延迟 (Excessive deferral)

该位置 1 时，如果 MAC 控制寄存器中的延迟检查 (DC) 位为高电平，则此位指示因出现超过 24288 个位时间的过度延迟而中止发送。

位 1 **UF**：下溢错误 (Underflow error)

该位置 1 时，指示 MAC 因 RAM 存储器的数据未及时到达而中止了帧发送。下溢错误表示 DMA 在帧发送期间遇到发送缓冲区为空的情况。发送过程将进入挂起状态并将发送下溢（寄存器 5[5]）和发送中断（寄存器 5[0]）位置 1。

位 0 **DB**：延迟位 (Deferred bit)

该位置 1 时，指示 MAC 在发送前因存在载波而延迟。该位仅在半双工模式下有效。

- **TDES1：发送描述符字 1**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			TBS2													Reserved			TBS1												
			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

31:29 保留，必须保持复位值。

28:16 **TBS2**：发送缓冲区 2 大小 (Transmit buffer 2 size)

这些位以字节为单位指示第二个数据缓冲区的大小。如果 TDES0[20] 位置 1，则该字段无效。

15:13 保留，必须保持复位值。

12:0 **TBS1**：发送缓冲区 1 大小 (Transmit buffer 1 size)

这些位以字节为单位指示第一个数据缓冲区的大小。如果该字段为 0，DMA 将忽略该缓冲区并使用缓冲区 2 或下一个描述符，具体取决于 TCH (TDES0[20]) 的值。

- **TDES2：发送描述符字 2**

TDES2 包含描述符第一个缓冲区的地址指针，或者包含时间戳数据。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TBAP1/TBAP/TTSL																															
rw																															

位 31:0 **TBAP1**：发送缓冲区 1 地址指针/发送帧时间戳低位 (Transmit buffer 1 address pointer / Transmit frame time stamp low)

这些位有两种不同的功能：这些位向 DMA 指示数据在存储器中的位置，而当所有数据传输完毕后，DMA 可利用这些位传回时间戳数据。

TBAP：当软件为 DMA 提供此描述符（TDES0 中的 OWN 位置 1）时，这些位将指示缓冲区 1 的物理地址。缓冲区地址对齐方式不限。有关缓冲区地址对齐方式的详细信息，请参见 *第 860 页的主机数据缓冲区对齐*。

TTSL：在将 TDES0 中的 OWN 位清零前，DMA 将使用为相应发送帧所捕获的时间戳的 32 个最低有效位更新该字段（覆盖 TBAP1 的值）。只有在激活该帧的时间戳功能（请参见 TDES0 的位 25 TTSE）并且描述符中的末段控制位 (LS) 置 1 时，该字段才包含时间戳。

- **TDES3：发送描述符字 3**

  TDES3 包含描述符中第二个缓冲区或下一个描述符的地址指针，或者包含时间戳数据。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TBAP2/TBAP2/TTSH																															
rw																															

位 31:0 **TBAP2**：发送缓冲区 2 地址指针（下一个描述符地址）/发送帧时间戳高位 (Transmit buffer 2 address pointer (Next descriptor address) / Transmit frame time stamp high)

这些位有两种不同的功能：这些位向 DMA 指示数据在存储器中的位置，而当所有数据传输完毕后，DMA 可利用这些位传回时间戳数据。

TBAP2：当软件为 DMA 提供此描述符（TDES0 中的 OWN 位置 1）并且使用描述符环结构时，这些位将指示缓冲区 2 的物理地址。如果链接第二个地址位 (TDES1 [24]) 置 1，则该地址包含下一个描述符所在物理寄存器的指针。只有在 TDES1 [24] 位置 1 时，缓冲区地址指针才必须与总线宽度相符。（将在内部忽略 LSB。）

TTSH：在将 TDES0 中的 OWN 位清零前，DMA 将使用为相应发送帧所捕获的时间戳的 32 个最高有效位更新该字段（覆盖 TBAP2 的值）。只有在激活该帧的时间戳功能（请参见 TDES0 的位 25 TTSE）并且描述符中的末段控制位 (LS) 置 1 时，该字段才包含时间戳。

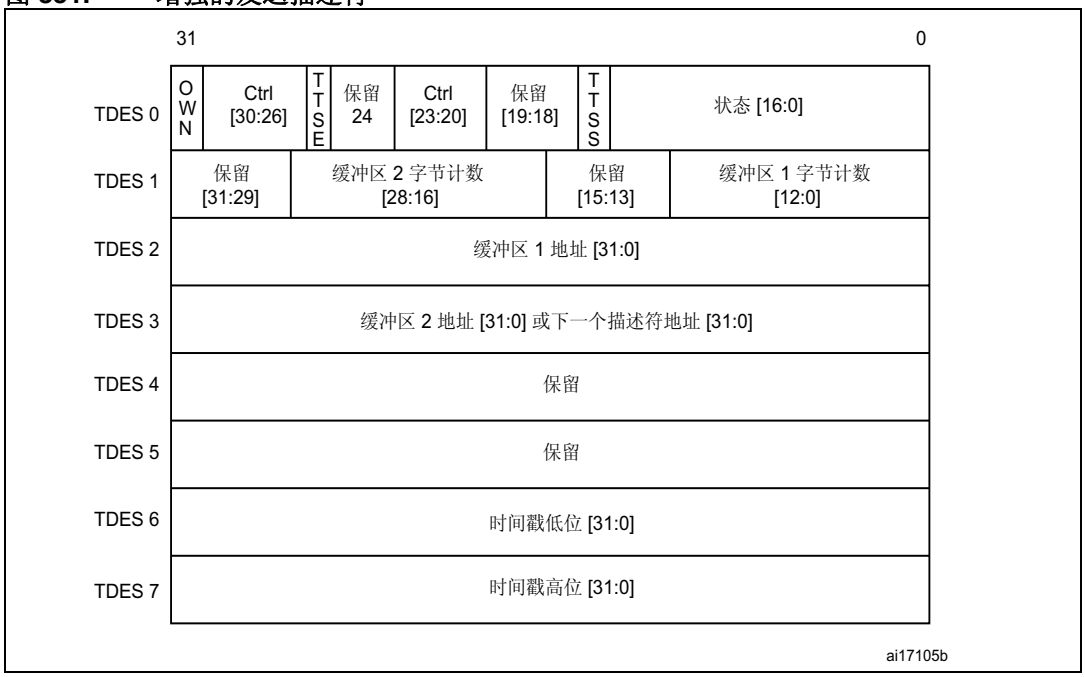
增强 Tx DMA 描述符

如果时间戳已激活（TSE=1，ETH_PTPTSCR 位 0）或 IPv4 校验和减荷已激活（IPCO=1，ETH_MACCR 位 10），则必须使用增强的描述符（通过 ETHDMABMR 位 7 EDFE=1 使能）。

增强的描述符由 8 个 32 位字组成，即描述符正常大小的两倍。TDES0、TDES1、TDES2 和 TDES3 的定义与常规发送描述符相同（请参见 *常规 Tx DMA 描述符*）。TDES6 和 TDES7 包含时间戳。TDES4、TDES5、TDES6 和 TDES7 的定义如下。

选择增强的描述符模式时，需要利用软件为每个描述符分配 32 个字节（8 个双字）的内存。如果未使用时间戳或 IPv4 校验和减荷，则可以禁止增强的描述符格式，而软件可使用默认大小为 16 字节的常规描述符。

图 351. 增强的发送描述符



- **TDES4：发送描述符字 4**

  保留

- **TDES5：发送描述符字 5**

  保留

- **TDES6：发送描述符字 6**

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

位 31:0 **TTSL**：发送帧时间戳低位 (Transmit frame time stamp low)

DMA 使用为相应发送帧所捕获的时间戳的 32 个最低有效位更新该字段。只有在描述符的末段控制位 (LS) 置 1 时，该字段才包含时间戳。

- **TDES7：发送描述符字 7**

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

位 31:0 **TTSH**：发送帧时间戳高位 (Transmit frame time stamp high)

DMA 使用为相应发送帧所捕获的时间戳的 32 个最高有效位更新该字段。只有在描述符的末段控制位 (LS) 置 1 时，该字段才包含时间戳。

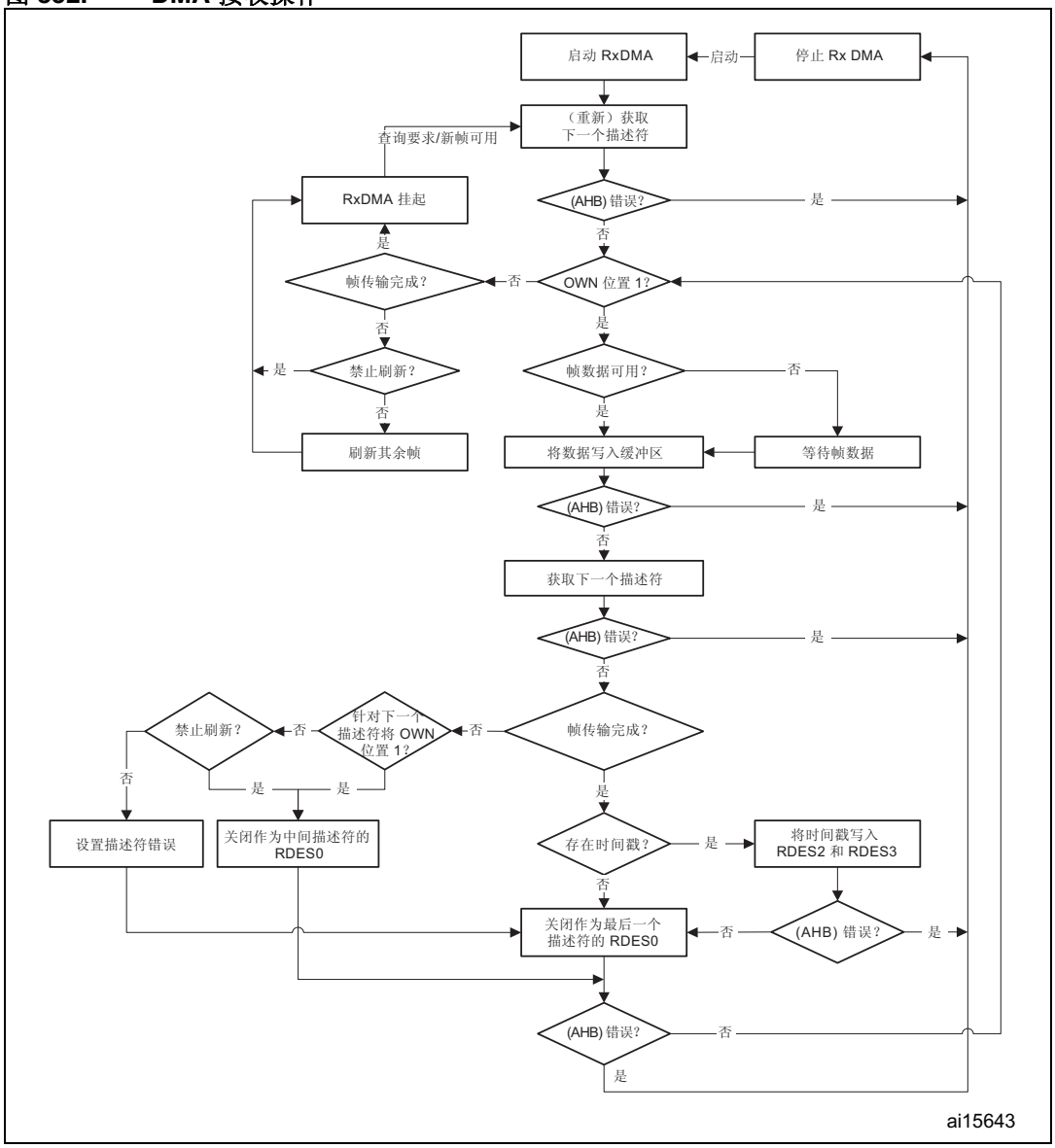
29.6.8 Rx DMA 配置

图 352 所示为接收 DMA 引擎的接收序列，具体说明如下：

1. CPU 设置接收描述符 (RDES0-RDES3) 并将 OWN 位 (RDES0[31]) 置 1。
2. SR 位（ETH_DMAOMR 寄存器 [1]）置 1 后，DMA 即进入运行状态。在运行状态下，DMA 轮询接收描述符列表，尝试获取空闲描述符。如果获得的描述符不空闲（由 CPU 所拥有），则 DMA 进入挂起状态并跳转到第 9 步。
3. DMA 将所获取描述符中的接收数据缓冲区地址解码。
4. 处理传入帧并将其放入所获取描述符的数据缓冲区。
5. 当缓冲区已满或帧传输完成时，接收引擎将获取下一个描述符。
6. 如果当前的帧传输完成，DMA 将继续执行第 7 步。如果 DMA 未拥有下一个获取的描述符且帧传输尚未完成（尚未传输 EOF），则 DMA 会将 RDES0 中的描述符错误位置 1（除非禁止刷新）。DMA 关闭当前描述符（将 OWN 位清零）并通过在 RDES1 值中将末段位 (LS) 清零来将其标记为中间描述符（如果禁止刷新，则将其标记为最后一个描述符），随后继续执行第 8 步。如果 DMA 已拥有下一个描述符，但当前的帧传输尚未完成，则 DMA 将关闭当前描述符作为中间值并返回第 4 步。
7. 如果已使能 IEEE 1588 时间戳功能，DMA 会将时间戳（如果可用）写入当前描述符的 RDES2 和 RDES3。DMA 随后获取所接收帧的状态并将该状态字写入当前描述符的 RDES0，同时 OWN 位清零且末段位置 1。
8. 接收引擎检查最新描述符的 OWN 位。如果 CPU 拥有该描述符（OWN 位为 0），则接收缓冲区不可用位（ETH_DMASR 寄存器 [7]）置 1 且 DMA 接收引擎进入挂起状态（第 9 步）。如果 DMA 拥有该描述符，引擎将返回第 4 步并等待下一个帧。
9. 在接收引擎进入挂起状态前，将从接收 FIFO 中刷新部分帧（可使用 ETH_DMAOMR 寄存器的位 24 控制刷新）。
10. 当收到接收轮询要求命令或者可以从接收 FIFO 获得下一个帧的起点时，接收 DMA 将退出挂起状态。引擎继续执行第 2 步并重新获取下一个描述符。

直到完成时间戳回写并准备将状态回写到描述符时，DMA 才会确认接收状态。如果软件已通过 CSR 使能时间戳功能，则当无法获得帧的有效时间戳值时（例如，在写入时间戳前接收 FIFO 已满），DMA 会向 RDES2 和 RDES3 的所有位写入 1。否则（即，未使能时间戳功能），RDES2 和 RDES3 保持不变。

图 352. DMA 接收操作



接收描述符获取

接收引擎始终会尝试为即将传入的帧获取一个额外的描述符。只要满足以下任一条件，即尝试获取描述符：

- DMA 进入运行状态后，接收启动/停止位（ETH_DMAOMR 寄存器 [1]）立即置 1。
- 在当前传输的帧结束前，当前描述符的数据缓冲区已满。
- 控制器已完成数据帧接收，但当前的接收描述符尚未关闭。
- 接收过程因缓冲区由 CPU 所拥有 (RDES0[31] = 0) 而挂起，并且接收到新帧。
- 已发出接收轮询要求命令。

接收帧处理

只有当帧通过地址过滤且其大小大于或等于为接收 FIFO 所设置的可配置阈值字节数时，或者在存储转发模式下将整个帧写入 FIFO 时，MAC 才会将接收的帧传输到 STM32F4xx 存储器。如果帧未通过地址过滤，将自行进入 MAC 块中（除非接收所有位 ETH_MACFFR [31] 置 1）。不足 64 字节的帧由于会发生冲突或过早结束，因而可从接收 FIFO 中清除。接收 64（可配置阈值）个字节后，DMA 块开始将帧数据传输到当前描述符所指定的接收缓冲区中。DMA AHB 接口准备好接收数据传输后，如果 DMA 当前未从存储器获取发送数据，则将第一个描述符位 (RDES0[9]) 置 1，以分隔该帧。数据缓冲区已满或者帧的末段已传输到接收缓冲区时，OWN (RDES0[31]) 位将复位为 0，此时将释放描述符。如果帧包含在单个描述符中，则最后一个描述符位 (RDES0[8]) 和第一个描述符位 (RDES0[9]) 均置 1。DMA 获取下一个描述符，将最后一个描述符位 (RDES0[8]) 置 1，并释放前一个帧描述符中的 RDES0 状态位。然后，DMA 将接收中断位（ETH_DMASR 寄存器 [6]）置 1。这一过程将重复执行，直至 DMA 遇到被标记为由 CPU 所有的描述符。如果出现这种情况，接收过程将接收缓冲区不可用位（ETH_DMASR 寄存器 [7]）置 1，随后进入挂起状态。但其在接收列表中的位置仍然保留。

接收过程挂起

如果在接收过程处于挂起状态时有新的接收帧到达，则 DMA 将重新获取 STM32F4xx 存储器中的当前描述符。如果该描述符现在由 DMA 所拥有，接收过程将重新进入运行状态并开始接收帧。如果该描述符仍由主机所拥有，则默认情况下，DMA 将丢弃 Rx FIFO 顶部的当前帧并将丢失帧计数器递增。如果 Rx FIFO 中存储了多个帧，将重复执行该过程。将 DMA 工作模式寄存器的位 24 (DFRF) 置 1 后，可避免丢弃或刷新 Rx FIFO 顶部的帧。在这种情况下，接收过程将接收缓冲区不可用状态位置 1 并返回到挂起状态。

常规 Rx DMA 描述符

常规接收描述符结构由四个 32 位字（16 字节）组成，如*图 353* 所示。下文将介绍 RDES0、RDES1、RDES2 和 RDES3 的各个位。

请注意，如果已激活时间戳功能（ETH_PTPTSCR 位 0 TSE=1）或 IPv4 校验和减荷（ETH_MACCR 位 10 IPCO=1），则必须使用增强的描述符。

图 353. 常规 Rx DMA 描述符结构

- **RDES0：接收描述符字 0**

  REDS0 包含接收帧状态、帧长度和描述符所有关系信息。

31	30	29:16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OWN	AFM	FL	ES	DE	SAF	LE	OE	VLAN	FS	LS	IPHCE/TSV	LCO	FT	RWT	RE	DE	CE	PCE/ESA
rw																		

位 31 **OWN**：所有关系位 (Own bit)

该位置 1 时，指示描述符由 MAC 子系统的 DMA 所拥有。该位复位时，指示描述符由主机所拥有。DMA 在帧接收完成或此描述符的关联缓冲区已满时将该位清零。

位 30 **AFM**：目标地址过滤失败 (Destination address filter fail)

该位置 1 时，指示帧未能通过 MAC 内核中的 DA 过滤。

位 29:16 **FL**：帧长度 (Frame length)

这些位指示传输到主机存储器中的接收帧字节长度（含 CRC）。只有在最后一个描述符位 (RDES0[8]) 置 1 且描述符错误位 (RDES0[14]) 复位时，该字段才有效。

在最后一个描述符位 (RDES0[8]) 置 1 时，该字段有效。当最后一个描述符位和错误汇总位均未置 1 时，该字段指示已为当前帧传输的累计字节数。

位 15 **ES**：错误汇总 (Error summary)

指示以下位的逻辑或运算结果：

RDES0[1]：CRC 错误 (CRC error)

RDES0[3]：接收错误 (Receive error)

RDES0[4]：看门狗超时 (Watchdog timeout)

RDES0[6]：延迟冲突 (Late collision)

RDES0[7]：巨帧 (Giant frame)（当 RDES0[7] 指示 IPV4 报头校验和错误时，该位不适用。）

RDES0[11]：上溢错误 (Overflow error)

RDES0[14]：描述符错误 (Descriptor error)。

只有最后一个描述符 (RDES0[8]) 置 1 时，该字段才有效。

位 14 **DE**：描述符错误 (Descriptor error)

该位置 1 时，指示某个帧因超过当前描述符缓冲区的大小而被截断以及 DMA 未拥有下一个描述符。帧被截断。只有最后一个描述符 (RDES0[8]) 置 1 时，该字段才有效。

位 13 **SAF**：源地址过滤失败 (Source address filter fail)

该位置 1 时，指示帧的 SA 字段未能通过 MAC 内核中的 SA 过滤。

位 12 **LE**：长度错误 (Length error)

该位置 1 时，指示接收帧的实际长度与长度/类型字段的值不符。该字段仅在帧类型位 (RDES0[5]) 复位后有效。

位 11 **OE**：上溢错误 (Overflow error)

该位置 1 时，指示接收帧因缓冲区上溢而损坏。

位 10 **VLAN**：VLAN 标记 (VLAN tag)

该位置 1 时，指示描述符所指向的帧是由 MAC 内核标记的 VLAN 帧。

位 9 **FS**：第一个描述符 (First descriptor)

该位置 1 时，指示此描述符包含帧的第一个缓冲区。如果第一个缓冲区的大小为 0，则第二个缓冲区将包含帧的帧头。如果第二个缓冲区的大小为 0，则下一个描述符将包含帧的帧头。

位 8 **LS**：最后一个描述符 (Last descriptor)

该位置 1 时，指示此描述符指向的缓冲区为帧的最后一个缓冲区。

位 7 **IPHCE/TSV**：IPv 报头校验和错误/时间戳有效 (IPv header checksum error/time stamp valid)

IPHCE 位置 1 时，指示 IPv4 或 IPv6 报头中存在错误。导致此错误的原因可能是：以太网类型字段与 IP 报头版本字段值不一致，或者与 IPv4 中报头的校验和不匹配，或者以太网帧缺少所需的 IP 报头字节数。如*表 168* 中所指出的那样，该位可具有特殊的含义。

使能增强的描述符格式（EDFE=1，ETH_DMABMR 的位 7）后，该位具有 TSV 功能（否则其为 IPHCE）。TSV 位置 1 时，表示将在描述符字 6 (RDES6) 和 7 (RDES7) 中写入时间戳快照。只有最后一个描述符位 (RDES0[8]) 置 1 时，TSV 才有效。

位 6 **LCO**：延迟冲突 (Late collision)

该位置 1 时，表示在以半双工模式接收帧时发生了延迟冲突。

位 5 **FT**：帧类型 (Frame type)

该位置 1 时，表示接收帧为以太网类型帧（LT 字段大于或等于 0x0600）。该位复位时，表示接收的帧为 IEEE802.3 帧。该位不适用于少于 14 个字节的矮帧。使用正常描述符格式 (ETH_DMABMR EDFE=0) 时，FT 可具有特殊的含义，如*表 168* 中所指出的那样。

位 4 **RWT**：接收看门狗超时 (Receive watchdog timeout)

该位置 1 时，表示接收看门狗计时器在接收当前帧时超时，且当前帧在看门狗超时后被截断了。

位 3 **RE**：接收错误 (Receive error)

该位置 1 时，表示在帧接收期间，当发出 RX_DV 信号时，会发出 RX_ERR 信号。

位 2 **DE**：Dribble 位错误 (Dribble bit error)

该位置 1 时，表示接收的帧具有非整数倍数的字节（奇数半字节）。该位仅在 MII 模式下有效。

位 1 **CE**：CRC 错误 (CRC error)

该位置 1 时，表示接收的帧发生循环冗余校验 (CRC) 错误。只有最后一个描述符 (RDES0[8]) 置 1 时，该字段才有效。

位 0 **PCE/ESA**：有效负载校验和错误/扩展状态可用

该位置 1 时，表示由内核计算的 TCP、UDP 或 ICMP 校验和与接收到的封装 TCP、UDP 或 ICMP 段校验和字段不匹配。当接收到的有效负载字节数与接收到的以太网帧中封装 IPv4 或 IPv6 数据图的长度字段中所指示的值不匹配时，该位也会被置 1。如*表 168* 中所指出的那样，该位可具有特殊的含义。

使能增强的描述符格式（EDFE=1，ETH_DMABMR 的位 7）后，该位具有 ESA 功能（否则其为 PCE）。ESA 置 1 时，表示描述符字 4 (RDES4) 中存在扩展状态。只有最后一个描述符位 (RDES0[8]) 置 1 时，ESA 才有效。

位 5、7 和 0 用于指示*表 168* 中讨论的情况。

表 168. 接收描述符 0 - 位 7、5 和 0 的编码（仅正常描述符格式，EDFE=0)

位 5：帧类型 (frame type)	位 7：IPC 校验和错误 (IPC checksum error)	位 0：有效负载校验和错误 (payload checksum error)	帧状态
0	0	0	IEEE 802.3 类型帧（长度字段的值小于 0x0600。）
1	0	0	IPv4/IPv6 类型帧，未检测到校验和错误
1	0	1	IPv4/IPv6 类型帧，检测到有效负载校验和错误（请参见 PCE 的相关说明）

表 168. 接收描述符 0 - 位 7、5 和 0 的编码（仅正常描述符格式，EDFE=0)（续）

位 5：帧类型 (frame type)	位 7：IPC 校验和错误 (IPC checksum error)	位 0：有效负载校验和错误 (payload checksum error)	帧状态
1	1	0	IPv4/IPv6 类型帧，检测到 IP 报头校验和错误（请参见 IPC CE 的相关说明）
1	1	1	IPv4/IPv6 类型帧，同时检测到 IP 报头和有效负载校验和错误
0	0	1	IPv4/IPv6 类型帧，无 IP 报头校验和错误，有效负载检查因有效负载不受支持而被绕过
0	1	1	既不是 IPv4 也不是 IPv6 的类型帧（校验和减荷引擎完全绕过校验和。）
0	1	0	保留

- **RDES1：接收描述符字 1**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DIC	RBS2		RBS2													RER	RCH	Reserved	RBS												
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31 **DIC:** 完成时禁止中断 (Disable interrupt on completion)

置 1 时，该位会阻止状态寄存器的 RS 位 (CSR5[6]) 置 1，使得接收的帧在由该描述符所指示的缓冲区中结束，从而会因该帧的 RS 而禁止引发主机的中断。

位 30:29 保留，必须保持复位值。

位 28:16 **RBS2：**接收缓冲区 2 大小 (Receive buffer 2 size)

这些位以字节为单位指示第二个数据缓冲区的大小。即使 RDES3（缓冲区 2 地址指针）的值未与总线宽度对齐，缓冲区大小也必须为 4、8 或 16 的倍数，具体取决于总线宽度（分别为 32、64 或 128）。如果缓冲区大小不是 4、8 或 16 的倍数，则不会定义产生的行为。如果将 RDES1[14] 置 1，则该字段无效。

位 15 **RER：**接收环结束 (Receive end of ring)

该位置 1 时，表示描述符列表已到达其最后一个描述符。DMA 会返回描述符列表的基址，并形成一个描述符环。

位 14 **RCH：**链接的第二个地址 (Second address chained)

该位置 1 时，表示描述符中的第二个地址是下一个描述符地址，而非第二个缓冲区地址。该位置 1 时，RBS2 (RDES1[28:16]) 为“无关”值。RDES1[15] 优先级高于 RDES1[14]。

位 13 保留，必须保持复位值。

位 12:0 **RBS1：**接收缓冲区 1 大小 (Receive buffer 1 size)

以字节为单位表示第一个数据缓冲区的大小。即使 RDES2（缓冲区 1 地址指针）的值未对齐，缓冲区大小也必须为 4、8 或 16 的倍数，具体取决于总线宽度（32、64 或 128）。如果缓冲区大小不是 4、8 或 16 的倍数，则不会定义产生的行为。如果该字段为 0，则 DMA 会忽略该缓冲区并使用缓冲区 2 或下一个描述符，具体取决于 RCH（位 14）的值。

- **RDES2：接收描述符字 2**

  RDES2 包含描述符中第一个数据缓冲区的地址指针，或者包含时间戳数据。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RBP1/RTSL																															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:0 **RBAP1/RTSL**：接收缓冲区 1 地址指针 / 接收帧时间戳低电平 (Receive buffer 1 address pointer / Receive frame time stamp low)

这些位有两种不同的功能：应用程序使用它们向 DMA 指示数据在存储器中的存储位置，而当所有数据均传输完成后，DMA 可以使用这些位来传回时间戳数据。

RBAP1：当软件为 DMA 提供此描述符时（当 RDES0 中的 OWN 位置 1 时），这些位会指示缓冲区 1 的物理地址。在缓冲区地址对齐方面没有任何限制，但以下情况除外：当使用 RDES2 值存储帧起点时，DMA 会使用所配置的值生成其地址。请注意，在传输帧开始期间，DMA 将在 RDES2[3/2/1:0] 位为 0 的情况下执行写操作，但帧数据会根据实际的缓冲区地址指针进行移位。当地址指针指向存储有帧中间部分或最后部分的缓冲区时，DMA 将忽略 RDES2[3/2/1:0]（与总线宽度 128/64/32 相对应）。

RTSL：在将 RDES0 中的 OWN 位清零之前，DMA 会使用为对应接收帧捕获的时间戳的 32 个最低有效位更新该字段（覆盖 RBAP1 的值）。只有此帧的时间戳已激活且描述符中的末段控制位 (LS) 已置 1 时，该字段才具有时间戳。

- **RDES3：接收描述符字 3**

  RDES3 包含描述符中第二个数据缓冲区或下一个描述符的地址指针，或者包含时间戳数据。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RBP2/RTSH																															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:0 **RBAP2/RTSH**：接收缓冲区 2 地址指针（下一个描述符地址）/ 接收帧时间戳高电平 (Receive buffer 2 address pointer (next descriptor address) / Receive frame time stamp high)

这些位有两种不同的功能：应用程序使用它们向 DMA 指示数据在存储器中的存储位置，而当所有数据均传输完成后，DMA 可以使用这些位来传回时间戳数据。

RBAP1：当软件为 DMA 提供此描述符时（当 RDES0 中的 OWN 位置 1 时），这些位会指示当使用描述符环结构时缓冲区 2 的物理地址。如果链接的第二个地址 (RDES1 [24]) 位置 1，则该地址包含指向下一个描述符所在物理寄存器的指针。如果 RDES1 [24] 置 1，则缓冲区（下一个描述符）地址指针必须是总线宽度对齐形式（RDES3[3、2 或 1:0] = 0，与总线宽度 128、64 或 32 相对应，LSB 被内部忽略。）然而，当 RDES1 [24] 复位时，RDES3 值无任何限制，但以下情况除外：当使用 RDES3 值存储帧起点时，DMA 会使用所配置的值生成其缓冲区地址。当地址指针指向存储有帧中间部分或最后部分的缓冲区时，DMA 将忽略 RDES3[3、2 或 1:0]（与总线宽度 128、64 或 32 相对应）。

RTSH：在将 RDES0 中的 OWN 位清零之前，DMA 会使用为对应接收帧捕获的时间戳的 32 个最高有效位更新该字段（覆盖 RBAP2 的值）。只有时间戳已激活且描述符中的末段控制位 (LS) 已置 1 时，该字段才具有时间戳。

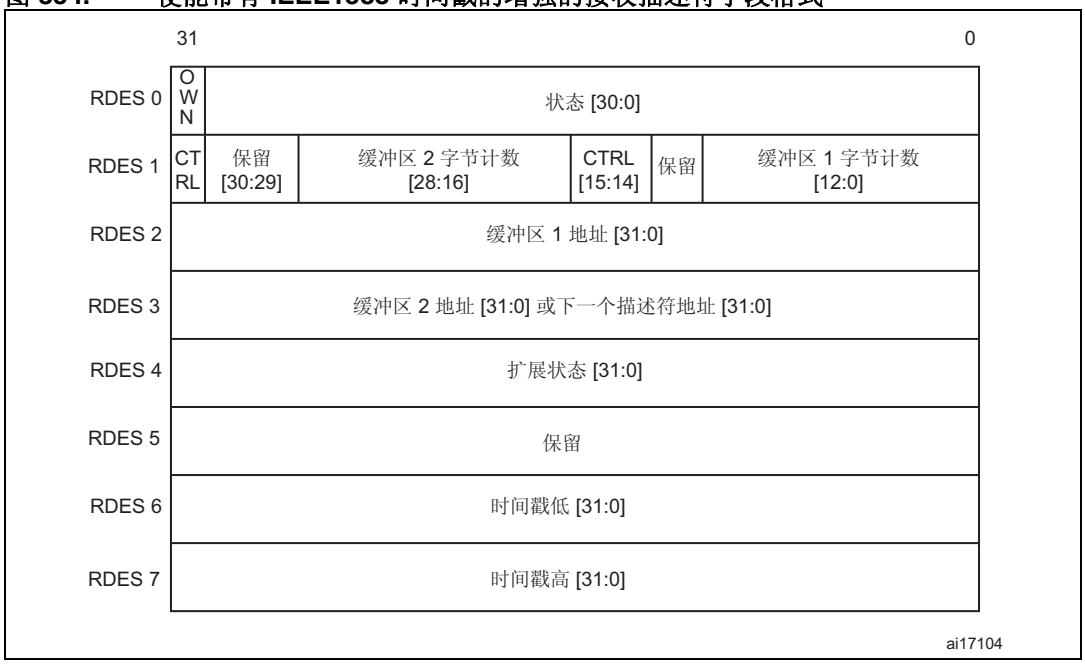
带有 IEEE1588 时间戳的增强的 Rx DMA 描述符

如果时间戳已激活（TSE=1，ETH_PTPTSCR 位 0）或 IPv4 校验和减荷已激活（IPCO=1，ETH_MACCR 位 10），则必须使用增强的描述符（通过 ETHDMABMR 位 7 EDFE=1 使能）。

增强的描述符由 8 个 32 位字组成，即描述符正常大小的两倍。RDES0、RDES1、RDES2 和 RDES3 的定义与常规接收描述符相同（请参见 *常规 Rx DMA 描述符*）。RDES4 包含扩展状态，而 RDES6 和 RDES7 包含时间戳。RDES4、RDES5、RDES6 和 RDES7 的定义如下。

选择增强的描述符模式时，需要利用软件为每个描述符分配 32 个字节（8 个双字）内存。如果未使用时间戳或 IPv4 校验和减荷，则可以禁止增强的描述符格式，而软件可使用默认大小为 16 字节的常规描述符。

图 354. 使能带有 IEEE1588 时间戳的增强的接收描述符字段格式



- **RDES4：接收描述符字 4**

仅当存在与 IPv4 校验和或时间戳（由 RDES0 中的位 0 加以指示）相关的状态时，如下所示的扩展状态才有效。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																		PV	PFT	PMT				IPV6PR	IPV4PR	IPCB	IPPE	IPHE	IPPT		
																		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:14 保留，必须保持复位值。

位 13 **PV**：PTP 版本 (PTP version)

置 1 时，表示接收的 PTP 消息使用 IEEE 1588 的版本 2 格式。清零时，将使用版本 1 格式。仅当消息类型为非零时才有效。

位 12 **PFT**：PTP 帧类型 (PTP frame type)

该位置 1 时，表示 PTP 消息直接通过以太网发送。该位清零且消息类型为非零时，表示 PTP 消息通过 UDP-IPv4 或 UDP-IPv6 发送。可通过位 6 或位 7 获取有关 IPv4 或 IPv6 的信息。

位 11:8 **PMT**：PTP 消息类型 (PTP message type)

将对这些位进行编码，以给出所接收消息的类型。

– 0000：未接收到任何 PTP 消息
– 0001：SYNC（所有时钟类型）
– 0010：Follow_Up（所有时钟类型）
– 0011：Delay_Req（所有时钟类型）
– 0100：Delay_Resp（所有时钟类型）
– 0101：Pdelay_Req（在点对点透明时钟中）或 Announce（在普通时钟或边界时钟中）
– 0110：Pdelay_Resp（在点对点透明时钟中）或 Management（在普通时钟或边界时钟中）
– 0111：Pdelay_Resp_Follow_Up（在点对点透明时钟中）或 Signaling（在普通时钟或边界时钟中）
– 1xxx - 保留

位 7 **IPV6PR**：接收到 IPv6 数据包 (IPv6 packet received)

该位置 1 时，表示接收到的数据包为 IPv6 数据包。

位 6 **IPV4PR**：接收到 IPv4 数据包 (IPv4 packet received)

该位置 1 时，表示接收到的数据包为 IPv4 数据包。

位 5 **IPCB**：绕过 IP 校验和 (IP checksum bypassed)

该位置 1 时，表示绕过校验和减荷引擎。

位 4 **IPPE**：IP 有效负载错误 (IP payload error)

该位置 1 时，表示由内核计算的 16 位 IP 有效负载校验和（即 TCP、UDP 或 ICMP 校验和）与接收段中对应的校验和字段不匹配。当 TCP、UDP 或 ICMP 段长度与 IP 报头字段中的有效负载长度值不匹配时，它同样会置 1。

位 3 **IPHE**：IP 报头错误 (IP header error)

该位置 1 时，表示由内核计算的 16 位 IPv4 报头校验和与接收的校验和字节不匹配，或 IP 数据报版本与以太网类型值不一致。

位 2:0 **IPPT**：IP 有效负载字节 (IP payload type)

如果 IPv4 校验和减荷被激活（IPCO=1，ETH_MACCR 位 10），则这些位表示 IP 数据报中封装的有效负载类型。当 IP 报头出错或存在分段的 IP时，这些位为“00”。

– 000：未知，或未处理 IP 有效负载
– 001：UDP
– 010：TCP
– 011：ICMP
– 1xx：保留

- **RDES5：接收描述符字 5**

  保留。

- **RDES6：接收描述符字 6**

  下表介绍了当接收描述符关闭且使能时间戳时对 RDES6 而言具有不同含义的字段。

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

位 31:0 **RTSL**：接收帧时间戳低电平 (Transmit frame time stamp low)

DMA 使用为对应接收帧捕获的时间戳的 32 最低有效位更新该字段。DMA 仅为最后描述符状态位 (RDES0[8]) 指示的接收帧的最后描述符更新该字段。当该字段和 RDES7 中的 RTSH 字段均显示 1 时，必须视为时间戳已损坏。

- **RDES7：接收描述符字 7**

  下表介绍了当接收描述符关闭且使能时间戳时对 RDES7 而言具有不同含义的字段。

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

位 31:0 **RTSH**：接收帧时间戳高电平 (Receive frame time stamp high)

DMA 使用为对应接收帧捕获的时间戳的 32 最高有效位更新该字段。DMA 仅为最后描述符状态位 (RDES0[8]) 指示的接收帧的最后描述符更新该字段。

当该字段和 RDES7 的 RTSL 字段都显示 1 时，必须将时间戳视为已损坏。

29.6.9 DMA 中断

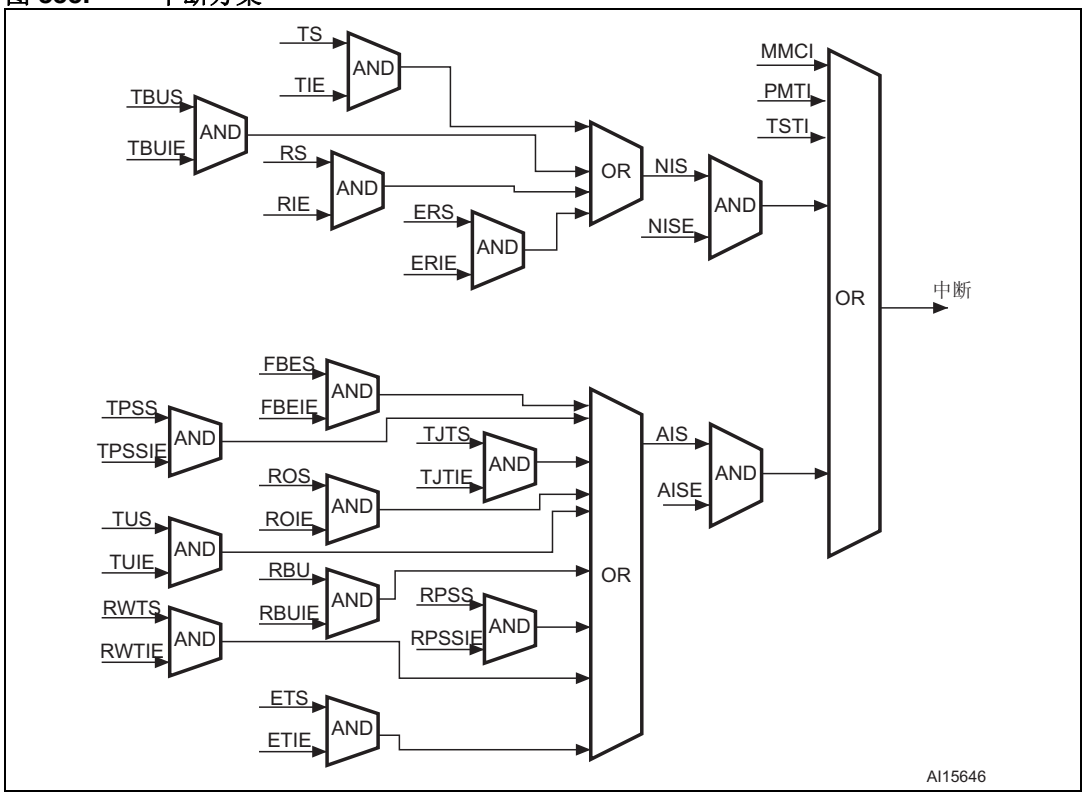
中断可以作为各种事件的结果而生成。ETH_DMASR 寄存器包含可能引起中断的所有位。ETH_DMAIER 寄存器对可能引起中断的每一个事件都包含一个使能位。

如 ETH_DMASR 寄存器中所描述，存在两组中断：正常中断和异常中断。通过向对应位位置写入 1 来清除中断。当清除了组内所有使能的中断时，对应的汇总位亦被清零。如果 MAC 内核是引发中断的原因，则 ETH_DMASR 寄存器中的任何 TSTS 或 PMTS 位都将置为高电平。

中断不会排队积压，如果中断事件在驱动程序对其做出响应之前发生，则不会再生成中断。例如，接收中断位（ETH_DMASR 寄存器 [6]）指示一个或多个帧被传输到 STM32F4xx 缓冲器。驱动程序必须扫描所有描述符，从 DMA 拥有的最后记录的位置到第一个位置。

对于同时发生的多个事件，一个中断只生成一次。驱动程序必须扫描 ETH_DMASR 寄存器，查找中断的原因。不会再次生成中断，除非在驱动程序已将 ETH_DMASR 寄存器中的适当位清零后发生新的中断事件。例如，控制器生成接收中断（ETH_DMASR 寄存器[6]）而驱动程序开始读取 ETH_DMASR 寄存器。然后，发生接收缓冲器不可用（ETH_DMASR 寄存器[7]）事件。驱动程序清除接收中断。即使是这时，也会由于活动或挂起的接收缓冲器不可用中断而生成一个新的中断。

图 355. 中断方案



29.7 以太网中断

以太网控制器有两个中断向量：一个专用于正常以太网操作，另一个仅当映射到 EXTI 线路 19 时用于以太网唤醒事件（带有唤醒帧或魔术数据包检测）。

如 *MAC 中断*和 *DMA 中断*章节所列，第一个以太网向量为 MAC 和 DMA 生成的中断而保留。

第二个向量为发生唤醒事件时 PMT 生成的中断而保留。唤醒事件对 EXTI 线路 19 的映射使 STM32F4xx 退出低功耗模式并生成一个中断。

当映射到 EXTI 线路 19 的以太网唤醒事件发生、使能了 MAC PMT 中断并且还使能了带上升沿检测的 EXTI 线路 19 中断时，会生成这两个中断。

可使用看门狗定时器（请参见 ETH_DMARSWTR 寄存器）灵活控制 RS 位（ETH_DMASR 寄存器）。当此看门狗定时器使用非零值编程时，只要 RxDMA 完成将接收的帧传输到系统存储器并且未触发接收状态（因未在相应接收描述符 (RDES1[31]) 使能接收状态），看门狗定时器即会被激活。当定时器按照编程值到时时，如果使能了 ETH_DMAIER 寄存器中的相应 RIE，则 RS 位会置 1 并会引发中断。当由于为该描述符将其使能而将一帧传输到存储器并且 RS 置 1 时，会在到时前禁止定时器。

注意： *读取 PMT 控制和状态寄存器会自动将接收的唤醒帧和接收的魔术数据包 PMT 中断标志清零。但是，由于用于这些标志的寄存器位于 CLK_RX 域，因此在固件能发现此更新前可能有显著的延迟。当 RX 时钟很慢（在 10 Mbit 模式）和当 AHB 总线为高频时，该延迟会特别长。由于从 PMT 到 CPU 的中断请求基于 CLK_RX 域中的相同寄存器，所以即使在读取 PMT_CSR 之后，CPU 也可能错误地第二次调用中断例程。因此，可能需要固件轮询接收的唤醒帧和接收的魔术数据包位，并仅在发现它们都为‘0’时退出中断服务程序。*

29.8 以太网寄存器说明

外设寄存器可支持字节（8 位）、半字（16 位）或字（32 位）访问。

29.8.1 MAC 寄存器说明

以太网 MAC 配置寄存器 (ETH_MACCR)

Ethernet MAC configuration register

偏移地址：0x0000

复位值：0x0000 8000

MAC 配置寄存器是 MAC 的工作模式寄存器。它建立了接收和发送工作模式。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						CSTF	Reserved	WD	JD	Reserved		IFG			CSD	Reserved	FES	ROD	LM	DM	IPCO	RD	Reserved	APCS	BL		DC	TE	RE	Reserved	
						rw		rw	rw			rw			rw		rw	rw	rw	rw	rw	rw		rw	rw		rw	rw	rw		

位 31:246 保留，必须保持复位值。

位 25 **CSTF**：类型帧的 CRC 去除 (CRC stripping for Type frames)
该位置 1 时，在将帧转发到应用之前，将去除并丢弃所有以太类型（类型字段大于 0x0600）帧的最后 4 个字节 (FCS)。

位 24 保留，必须保持复位值。

位 23 **WD**：禁止看门狗 (Watchdog disable)
当该位置 1 时，MAC 禁止接收器上的看门狗定时器并可接收多达 16384 字节的帧。
当该位复位时，MAC 允许接收的帧不超过 2048 字节并会截断超过此限制接收的任何字节。

位 22 **JD**：禁止 Jabber (Jabber disable)
当该位置 1 时，MAC 禁止发送器上的 jabber 定时器并可发送多达 16384 字节的帧。
当该位复位时，如果发送期间应用程序发送超过 2 048 字节的数据，MAC 会截断发送器。

位 21:20 保留，必须保持复位值。

位 19:17 **IFG**：帧间间隙 (Interframe gap)
这些位控制发送期间帧间的最小间隙。
000：96 位时间
001：88 位时间
010：80 位时间
……
111：40 位时间
注意：在半双工模式下，最小 IFG 仅能配置为 64 位时间 (IFG = 100)。不考虑更低的值。

位 16 **CSD**：禁止载波侦听 (Carrier sense disable)
值设置得很高时，该位会使 MAC 发送器忽略半双工模式下帧发送期间的 MII CRS 信号。不会因在此发送期间载波丢失或无载波而生成错误。
当该位为低电平时，MAC 发送器会生成载波侦听错误，甚至中止发送。

位 15 保留，必须保持复位值。

位 14 **FES**：快速以太网速度 (Fast Ethernet speed)

指示快速以太网 (MII) 模式下的速度：

0：10 Mbit/s

1：100 Mbit/s

位 13 **ROD**：禁止接收自身 (Receive own disable)

当该位置 1 时，MAC 禁止在半双工模式下接收帧。

当该位复位时，MAC 接收发送时 PHY 提供的所有包。

如果 MAC 在全双工模式下工作，该位不适用。

位 12 **LM**：回送模式 (Loopback mode)

当该位置 1 时，MAC 在 MII 回送模式下工作。回送正确工作需要 MII 接收时钟输入 (RX_CLK)，因为发送时钟不是内部回送的。

位 11 **DM**：双工模式 (Duplex mode)

当该位置 1 时，MAC 在全双工模式下工作，此时它可以同时发送和接收。

位 10 **IPCO**：IPv4 校验和减荷 (IPv4 checksum offload)

置 1 时，该位使能接收的帧有效载荷的 TCP/UDP/ICMP 标头的 IPv4 校验和检查。当该位复位时，禁止接收器中的校验和减荷功能并始终将相应的 PCE 和 IP HCE 状态位（请参见*第 843 页的表 165*）清零。

位 9 **RD**：禁止重试 (Retry disable)

当该位置 1 时，MAC 仅尝试一次发送。当在 MII 模式下发生冲突时，MAC 会忽略当前帧发送并以发送帧状态下过度冲突错误报告帧中止。

当该位复位时，MAC 会尝试根据 BL 的设置进行重试。

注意：该位仅在半双工模式下适用。

位 8 保留，必须保持复位值。

位 7 **APCS**：去除自动 pad/CRC (Automatic pad/CRC stripping)

当该位置 1 时，MAC 仅在长度字段值小于或等于 1500 字节时去除传入帧上的 Pad/FCS 字段。传送给应用程序的所有接收的帧，如果长度字段大于或等于 1501 字节，将都不去除 Pad/FCS 字段。

当该位复位时，MAC 会不加修改地传送所有传入的帧。

位 6:5 **BL**：后退限制 (Back-off limit)

后退限制决定发生冲突后重试期间 MAC 在重新安排一次发送之前等待的随机整数 (r) 个时间片延迟（对于 1000 Mbit/s 为 4 096 位时间，对于 10/100 Mbit/s 为 512 位时间）。

注意：该位仅在半双工模式下适用。

00: k = min (*n*, 10)

01: k = min (*n*, 8)

10: k = min (*n*, 4)

11: k = min (*n*, 1),

其中 *n* = 重新发送尝试的次数。随机整数 *r* 的取值范围为 $0 \leq r < 2^k$

位 4 **DC**：检查延迟 (Deferral check)

当该位置 1 时，便在 MAC 中使能了检查延迟功能。当发送状态机在 10/100-Mbit/s 模式下延迟超过 24288 位时间时，MAC 将指示帧中止状态，同时在发送帧状态中将过度延迟错误位置 1。当发送器准备好发送但因 MII 上有有效 CRS（载波侦听）信号而被阻止时延迟开始。延迟时间是非累积性的。如果发送器延迟 10 000 位时间，然后发送、冲突、后退，然后在完成后退后不得不再次延迟，则延迟定时器复位为 0 并重新开始。

当该位复位时，禁止延迟检查功能，MAC 发生延迟，直到 CRS 信号变为无效信号。该位仅在半双工模式下适用。

位 3 **TE**：发送器使能 (Transmitter enable)

当该位置 1 时，针对 MII 上的发送使能 MAC 的发送状态机。当该位复位时，在当前帧的发送完成后禁止 MAC 发送状态机，并且不再发送任何帧。

位 2 **RE**：接收器使能 (Receiver enable)

当该位置 1 时，使能 MAC 的接收器状态机，以便从 MII 接收帧。当该位复位时，在当前帧接收完成后禁止 MAC 接收状态机，并且不再从 MII 接收任何帧。

位 1:0 保留，必须保持复位值。

以太网 MAC 帧过滤寄存器 (ETH_MACFFR)

Ethernet MAC frame filter register

偏移地址：0x0004
复位值：0x0000 0000

MAC 帧过滤寄存器包含用于接收帧的过滤控件。该寄存器的某些控件转到 MAC 的地址检查块，以执行第一级地址过滤。第二级过滤基于其它控件（例如传送不良帧和传送控制帧）对传入帧过滤。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RA	Reserved																				HPF	SAF	SAIF	PCF		BFD	PAM	DAIF	HM	HU	PM
rw																					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31 **RA**：接收所有 (Receive all)

当该位置 1 时，MAC 接收器将所有接收的帧传送到应用程序，不管它们是否已通过地址过滤。SA/DA 过滤的结果在接收状态字的相应位更新（通过或失败）。当该位复位时，MAC 接收器传送给应用程序的只是通过了 SA/DA 地址过滤的那些帧。

位 30:11 保留，必须保持复位值。

位 10 **HPF**：散列或完美过滤器 (Hash or perfect filter)

当该位置 1 时，如果 HM 或 HU 位置 1，则地址过滤器传送与完美过滤或散列过滤匹配的帧。
当该位清零时，如果 HU 或 HM 位置 1，则仅传送与散列过滤器匹配的帧。

位 9 **SAF**：源地址过滤器 (Source address filter)

MAC 内核将接收的帧的 SA 字段与使能的 SA 寄存器中编程的值进行比较。如果比较结果匹配，则将 RxStatus 字中的 SAMatch 位设为高电平。当该位设为高电平且 SA 过滤失败时，MAC 放弃该帧。
当该位复位时，MAC 内核将接收的帧转发给应用程序。它还根据 SA 地址比较结果转发 RxStatus 中更新的 SA 匹配位。

位 8 **SAIF**：源地址反向过滤 (Source address inverse filtering)

当该位置 1 时，地址检查块在反向过滤模式下进行 SA 地址比较。SA 与 SA 寄存器匹配的帧被标记为 SA 地址过滤失败。
当该位复位时，SA 与 SA 寄存器不匹配的帧被标记为 SA 地址过滤失败。

位 7:6 **PCF**：传送控制帧 (Pass control frames)

这些位控制所有控制帧（包括单播和多播暂停帧）的转发。请注意暂停控制帧的处理只取决于流控制寄存器[2] 中的 RFCE。

00：MAC 阻止所有控制帧到达应用程序

01：MAC 将除了暂停控制帧以外的所有控制帧转发到应用程序

10：即使所有控制帧通过地址过滤失败，MAC 仍将它们转发给应用程序

11：MAC 转发通过地址过滤的控制帧。

这些位控制所有控制帧（包括单播和多播暂停帧）的转发。请注意暂停控制帧的处理只取决于流控制寄存器[2] 中的 RFCE。

00 或 01：MAC 阻止所有控制帧到达应用程序

10：即使所有控制帧通过地址过滤失败，MAC 仍将它们转发给应用程序

11：MAC 转发通过地址过滤的控制帧。

位 5 **BFD**：禁止广播帧 (Broadcast frames disable)

当该位置 1 时，地址过滤器对所有传入的广播帧进行过滤。

当该位复位时，地址过滤器传送所有接收到的广播帧。

位 4 **PAM**：传送所有多播 (Pass all multicast)

置 1 时，该位指示传送接收到的带多播目标地址（目标地址字段的第一位是 1）的所有帧。

复位时，多播帧的过滤取决于 HM 位。

位 3 **DAIF**：目标地址反向过滤 (Destination address inverse filtering)

当该位置 1 时，地址检查块在反向过滤模式下对单播和多播帧都进行 DA 地址比较。

复位时，正常过滤帧。

位 2 **HM**：散列多播 (Hash multicast)

置 1 时，MAC 根据散列表对接收到的多播帧执行目标地址过滤。

复位时，MAC 对多播帧执行完美目标地址过滤，即将 DA 字段与 DA 寄存器中编程的值进行比较。

位 1 **HU**：散列单播 (Hash unicast)

置 1 时，MAC 根据散列表对单播帧执行目标地址过滤。

复位时，MAC 对单播帧执行完美目标地址过滤，即将 DA 字段与 DA 寄存器中编程的值进行比较。

位 0 **PM**：混合模式 (Promiscuous mode)

当该位置 1 时，不论其目标或源地址为何，地址过滤器都传送所有传入的帧。PM 置 1 时，始终将接收状态字中 SA/DA 过滤失败状态位清零。

以太网 MAC 散列表高位寄存器 (ETH_MACHTHR)

Ethernet MAC hash table high register

偏移地址：0x0008

复位值：0x0000 0000

64 位散列表用于组地址过滤。对于散列过滤，传入帧中目标地址的内容通过 CRC 逻辑传送，CRC 寄存器中的高 6 位用于对散列表内容进行索引。CRC 是使用下列多项式编码的 32 位值（有关详细信息，请参见*第 29.5.3 节：MAC 帧接收*）：

$$G(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^{8} + x^{7} + x^{5} + x^{4} + x^{2} + x + 1$$

最高有效位决定要使用的寄存器（散列表高位/散列表低位），其它 5 位决定寄存器中的位。散列值 0b0 0000 选取所选寄存器中的位 0，值 0b1 1111 选取所选寄存器中的位 31。

例如，如果传入帧的 DA 接收为 0x1F52 419C B6AF（0x1F 是 MII 接口上接收的第一个字节），则内部计算出的 6 位散列值为 0x2C 且为了过滤要检查 HTH 寄存器位[12]。如果传入帧的 DA 接收为 0xA00A 9800 0045，则计算出的 6 位散列值为 0x07 且为了过滤要检查 HTL 寄存器位[7]。

如果寄存器中对应的位值为 1，则接受该帧。否则拒绝该帧。如果 ETH_MACFFR 寄存器中的 PAM（传送所有多播）位置 1，则不论多播散列值为何都接受所有多播帧。

散列表高位寄存器包含多播散列表的高 32 位。

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

位 31:0 **HTH**：散列表高位 (Hash table high)
该字段包含散列表的高 32 位。

以太网 MAC 散列表低位寄存器 (ETH_MACHTLR)

Ethernet MAC hash table low register

偏移地址：0x000C

复位值：0x0000 0000

散列表低位寄存器包含多播散列表的低 32 位。

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

位 31:0 **HTL**：散列表低位 (Hash table low)
该字段包含散列表的低 32 位。

以太网 MAC MII 地址寄存器 (ETH_MACMIIAR)

Ethernet MAC MII address register

偏移地址：0x0010

复位值：0x0000 0000

MII 地址寄存器通过管理接口控制外部 PHY 的管理周期。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																PA					MR					Reserved	CR			MW	MB
																rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rc_w1

位 31:16 保留，必须保持复位值。

位 15:11 **PA**：PHY 地址 (PHY address)
该字段指示正在访问 32 个可能的 PHY 器件中的哪一个。

位 10:6 **MR**：MII 寄存器 (MII register)
这些位在所选 PHY 器件中选择需要的 MII 寄存器。

位 5 保留，必须保持复位值。

位 4:2 **CR**：时钟范围 (Clock range)

CR 时钟范围选项可确定 HCLK 频率并用于决定 MDC 时钟频率：

选项	HCLK	MDC 时钟
000	60-100 MHz	HCLK/42
001	100-150 MHz	HCLK/62
010	20-35 MHz	HCLK/16
011	35-60 MHz	HCLK/26
100	150-168 MHz	HCLK/102
101、110、111	保留	-

位 1 **MW**：MII 写 (MII write)

此位置 1 是在告知 PHY，将要启动一个使用 MII 数据寄存器的写操作。如果此位未置 1，则表示会启动一个读操作，将数据放入 MII 数据寄存器。

位 0 **MB**：MII 忙碌 (MII busy)

向 ETH_MACMIIAR 和 ETH_MACMIIDR 写入前，此位应读取逻辑 0。向 ETH_MACMIIAR 写入过程中，此位也必须复位为 0。在 PHY 寄存器访问过程中，此位由应用程序设为 0b1，指示读或写访问正在进行中。在对 PHY 进行写操作过程中，ETH_MACMIIDR（MII 数据）应始终保持有效，直到 MAC 将此位清零。在对 PHY 进行读操作过程中，ETH_MACMIIDR 始终无效，直到 MAC 将此位清零。在此位清零后，才可以向 ETH_MACMIIAR（MII 地址）写入。

以太网 MAC MII 数据寄存器 (ETH_MACMIIDR)

Ethernet MAC MII data register

偏移地址：0x0014

复位值：0x0000 0000

MAC MII 数据寄存器存储要写入 PHY 寄存器的数据，该寄存器地址在 ETH_MACMIIAR 中指定。ETH_MACMIIDR 也将存储从 PHY 寄存器读取的数据，该寄存器地址由 ETH_MACMIIAR 指定。

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

位 31:16 保留，必须保持复位值。

位 15:0 **MD**：MII 数据 (MII data)

其中包含在某次管理读操作之后从 PHY 中读取的 16 位数据值，或在某次管理写操作之前要写入 PHY 的 16 位数据值。

以太网 MAC 流控制寄存器 (ETH_MACFCR)

Ethernet MAC flow control register

偏移地址：0x0018

复位值：0x0000 0000

流控制寄存器通过 MAC 来管理控制（暂停命令）帧的生成和接收。忙碌位置“1”时对寄存器的写操作将导致 MAC 生成暂停控制帧。控制帧的字段根据 802.3x 规范中的指定进行选择，此寄存器中的暂停时间值被用在控制帧的暂停时间字段中。忙碌位将保持置 1，直到将控制帧发送到电缆。主机必须确保向该寄存器写入前已将忙碌位清零。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PT																Reserved								ZQPD	Reserved	PLT		UPFD	RFCE	TFCE	FCB/ BPA
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw									rw		rw	rw	rw	rw	rw	rc_w1 /rw

位 31:16 **PT**：暂停时间 (Pause time)

此字段保留发送控制帧中暂停时间字段要使用的值。如果将暂停时间位配置为与 MII 时钟域双同步，则只能在目标时钟域的至少 4 个时钟周期后执行对此寄存器的连续写操作。

位 15:8 保留，必须保持复位值。

位 7 **ZQPD**：零时间片暂停禁止 (Zero-quanta pause disable)

如果置 1，当来自 FIFO 层的流控制信号去断言后，此位会禁止自动生成零时间片暂停控制帧。

此位复位时，会使能自动零时间片暂停控制帧生成的正常操作。

位 6 保留，必须保持复位值。

位 5:4 **PLT**：暂停阈值下限 (Pause low threshold)

此字段配置暂停定时器的阈值，达到该值时，会自动重新传输暂停帧。该阈值应始终小于位 [31:16] 配置的暂停时间。例如，如果 PT = 100H（256 个时隙），而 PLT = 01，则在第一个 PAUSE 帧发送完成后，第二个 PAUSE 帧会在 228 (256 - 28) 个时隙启动时自动发送。

选项	阈值
00	暂停时间减去 4 个时隙
01	暂停时间减去 28 个时隙
10	暂停时间减去 144 个时隙
11	暂停时间减去 256 个时隙

时隙定义为 MII 接口每发送 512 位（64 字节）所需的时间。

位 3 **UPFD**：单播暂停帧检测 (Unicast pause frame detect)

当此位置 1 时，MAC 除了检测具有唯一多播地址的暂停帧外，还会检测具有 ETH_MACA0HR 和 ETH_MACA0LR 寄存器所指定的站单播地址的暂停帧。

当此位复位时，MAC 仅检测具有 802.3x 标准中指定的唯一多播地址的暂停帧。

位 2 **RFCE**：接收流控制使能 (Receive flow control enable)

当此位置 1 时，MAC 对接收到的暂停帧进行解码，并禁止其在指定时间（暂停时间）内发送。

当此位复位时，将禁止暂停帧的解码功能。

位 1 **TFCE**：发送流控制使能 (Transmit flow control enable)

在全双工模式下，当此位置 1 时，MAC 将使能流控制操作来发送暂停帧。当此位复位时，将禁止 MAC 中的流控制操作，MAC 不会传送任何暂停帧。

在半双工模式下，当此位置 1 时，MAC 将使能背压操作。当此位复位时，将禁止背压功能。

位 0 **FCB/BPA**：流控制忙碌/背压激活 (Flow control busy/back pressure activate)

此位在全双工模式下启动暂停控制帧，在半双工模式下，TFCE 位置 1 时，会激活背压功能。

全双工模式下，向流控制寄存器写入数据前此位应读为 0。要启动暂停控制帧，应用程序必须将此位置 1。在控制帧发送过程中，此位保持置 1 以指示帧发送正在进行中。当暂停控制帧发送完成后，MAC 会将此位复位为 0。将此位清零后，才可以对流控制寄存器执行写操作。

在半双工模式下，当此位置 1（TFCE 也置 1）时，MAC 内核将置位背压功能。在背压操作期间，当 MAC 接收到新帧时，发送器会开始发送一个导致冲突的 JAM 模式。当 MAC 配置为全双工模式时，将自动禁止 BPA。

以太网 MAC VLAN 标记寄存器 (ETH_MACVLANTR)

Ethernet MAC VLAN tag register

偏移地址：0x001C

复位值：0x0000 0000

VLAN 标记寄存器包含用于标识 VLAN 帧的 IEEE 802.1Q VLAN 标记。MAC 将所接收帧的第十三及第十四字节（长度/类型）与 0x8100 比较，随后的 2 个字节与 VLAN 标记比较，如果存在匹配，则将所接收帧状态中收到的 VLAN 位置 1。帧的合法长度从 1518 字节增加到 1522 字节。

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

位 31:17 保留，必须保持复位值。

位 16 **VLANTC**：12 位 VLAN 标记比较 (12-bit VLAN tag comparison)

当此位置 1 时，使用 12 位 VLAN 标识符而不是完整的 16 位 VLAN 标记进行比较和过滤。VLAN 标记的位 [11:0] 与收到的 VLAN 标记帧的相应字段进行比较。

当此位复位时，收到的 VLAN 帧的第十五和第十六字节的全部 16 位都要进行比较。

位 15:0 **VLANTI**：VLAN 标记标识符 (VLAN tag identifier)（对应接收帧）

此字段包含用于标识 VLAN 帧的 802.1Q VLAN 标记，并与正在接收的 VLAN 帧的第十五和第十六字节进行比较。位 [15:13] 是用户优先级，位 [12] 是标准格式指示符 (CFI)，位 [11:0] 是 VLAN 标记的 VLAN 标识符 (VID) 字段。VLANTC 位置 1 时，仅使用 VID（位 [11:0]）进行比较。

如果 VLANTI（VLANTI[11:0]，如果 VLANTC 位置 1）全部为零，则 MAC 不会检查用于 VLAN 标记比较的第十五和第十六字节，而是将类型字段值为 0x8100 的所有帧均声明为 VLAN 帧。

以太网 MAC 远程唤醒帧过滤寄存器 (ETH_MACRWUFFR)

Ethernet MAC remote wakeup frame filter register

偏移地址：0x0028

复位值：0x0000 0000

应用程序将通过此地址对远程唤醒帧过滤寄存器进行写/读操作。实际上，唤醒帧过滤寄存器是八个（不透明的）此类唤醒帧过滤寄存器。对偏移 (0x0028) 的这个地址所进行的八个连续写操作会写入所有唤醒帧过滤寄存器。对偏移 (0x0028) 的这个地址所进行的八个连续读操作会读取所有唤醒帧过滤寄存器。此寄存器包含第七个 MAC 地址的高 16 位。有关详细信息，请参见*远程唤醒帧过滤寄存器*。

图 356. 以太网 MAC 远程唤醒帧过滤寄存器 (ETH_MACRWUFFR)

以太网 MAC PMT 控制和状态寄存器 (ETH_MACPMTCSR)

Ethernet MAC PMT control and status register

偏移地址：0x002C

复位值：0x0000 0000

ETH_MACPMTCSR 会配置唤醒事件请求并监视唤醒事件。

31	30:10	9	8:7	6	5	4:3	2	1	0
WFFRPR	Reserved	GU	Reserved	WFR	MPR	Reserved	WFE	MPE	PD
rs	Res.	rw		rc_r	rc_r		rw	rw	rs

位 31 **WFFRPR**：唤醒帧过滤寄存器指针复位 (Wakeup frame filter register pointer reset)
当此位置 1 时，会将远程唤醒帧过滤寄存器指针复位为 0b000。它会在 1 个时钟周期后自动清零。

位 30:10 保留，必须保持复位值。

位 9 **GU**：全局单播 (Global unicast)
当此位置 1 时，它会将 MAC (DAF) 地址确认所过滤的任意单播包使能为唤醒帧。

位 8:7 保留，必须保持复位值。

位 6 **WFR**：接收到唤醒帧 (Wakeup frame received)
当此位置 1 时，表示因接收唤醒帧而生成了电源管理事件。通过对此寄存器执行读操作可将此位清零。

位 5 **MPR**：接收到魔术数据包 (Magic packet received)
当此位置 1 时，表示因接收魔术数据包而生成了电源管理事件。通过对此寄存器执行读操作可将此位清零。

位 4:3 保留，必须保持复位值。

位 2 **WFE**：唤醒帧使能 (Wakeup frame enable)
当此位置 1 时，会因为接收到唤醒帧而使能电源管理事件的生成。

位 1 **MPE**：魔术数据包使能 (Magic Packet enable)
当此位置 1 时，会因为接收到魔术数据包而使能电源管理事件的生成。

位 0 **PD**：掉电 (Power down)
当此位置 1 时，所有接收到的帧都将丢弃。接收到魔术数据包或唤醒帧时，此位会自动清零，同时禁止掉电模式。此位清零后收到的帧会转发到应用程序。只有魔术数据包使能 (Magic Packet Enable) 或唤醒帧使能 (Wakeup Frame Enable) 位置 1 时，才可以将此位置 1。

以太网 MAC 调试寄存器 (ETH_MACDBGR)

Ethernet MAC debug register

偏移地址：0x0034

复位值：0x0000 0000

此调试寄存器给出了发送及接收数据路径和 FIFO 的所有主要模块的状态。全零状态表示 MAC 内核处于空闲状态（FIFO 为空）并且数据路径中没有进行任何活动。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						TFF	TFNE	Reserved	TFWA	TFRS		MTP	MTFCS		MMTEA	Reserved						RFFL		Reserved	RFRCS		RFWRA	Reserved	MSFRWCS		MMRPEA
						ro	ro		ro	ro	ro	ro	ro	ro	ro							ro	ro		ro	ro	ro		ro	ro	ro

位 31:26 保留，必须保持复位值。

位 25 **TFF**：Tx FIFO 已满 (Tx FIFO full)
当它为高电平时，表示 Tx FIFO 已满，因此无法再接受发送的帧。

位 24 **TFNE**：Tx FIFO 非空 (Tx FIFO not empty)
当它为高电平时，表示 TxFIFO 不为空，还留有一些数据要发送。

位 23 保留，必须保持复位值。

位 22 **TFWA**：Tx FIFO 写有效 (Tx FIFO write active)
当它为高电平时，表示 TxFIFO 写控制器有效且正在将数据传输到 TxFIFO。

位 21:20 **TFRS**：Tx FIFO 读状态 (Tx FIFO read status)
此字段指示 TxFIFO 读控制器的状态：
00：空闲状态 (Idle state)
01：读状态（将数据传输到 MAC 发送器）
10：等待来自 MAC 发送器的 TxStatus
11：写入收到的 TxStatus 或刷新 TxFIFO

位 19 **MTP**：MAC 发送器处于暂停 (MAC transmitter in pause)
当它为高电平时，表示 MAC 发送器处于暂停条件（仅在全双工模式下），因此不会安排发送任何帧。

位 18:17 **MTFCS**：MAC 发送帧控制器状态 (MAC transmit frame controller status)
此字段指示 MAC 发送帧控制器的状态：
00：空闲
01：等待前一个帧的状态或 IFG/回退阶段结束
10：生成并发送暂停控制帧（在全双工模式下）
11：传输要发送的输入帧

位 16 **MMTEA**：MAC MII 发送引擎有效 (MAC MII transmit engine active)
当它为高电平时，表示 MAC MII 发送引擎正在主动发送数据而未处于空闲状态。

位 15:10 保留，必须保持复位值。

位 9:8 **RFFL**：Rx FIFO 填充级别 (Rx FIFO fill level)
它指示 Rx FIFO 填充级别的状态：
00：RxFIFO 为空
01：RxFIFO 填充级别低于流控制取消激活阈值
10：RxFIFO 填充级别高于流控制激活阈值
11：RxFIFO 已满

位 7 保留，必须保持复位值。

位 6:5 **RFRCS**：Rx FIFO 读控制器状态 (Rx FIFO read controller status)
它指示 Rx FIFO 读控制器的状态：
00：IDLE 状态
01：读取帧数据
10：读取帧状态（或时间戳）
11：刷新帧数据和状态

位 4 **RFWRA**：Rx FIFO 写控制器有效 (Rx FIFO write controller active)
当它为高电平时，表示 Rx FIFO 写控制器有效并正在将收到的帧传输到 FIFO。

位 3 保留，必须保持复位值。

位 2:1 **MSFRWCS**：MAC 小 FIFO 读/写控制器状态 (MAC small FIFO read / write controllers status)
当这些位为高电平时，表示 MAC 接收帧控制器模块的各个小 FIFO 读和写控制器的有效状态。

位 0 **MMRPEA**：MAC MII 接收协议引擎有效 (MAC MII receive protocol engine active)
当它为高电平时，表示 MAC MII 接收协议引擎正在主动接收数据而未处于空闲状态。

以太网 MAC 中断状态寄存器 (ETH_MACSR)

Ethernet MAC interrupt status register

偏移地址：0x0038

复位值：0x0000 0000

ETH_MACSR 寄存器内容在 MAC 中标识可生成中断的事件。

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						TSTS	Reserved		MMCTS	MMCRS	MMCS	PMTS	Reserved		
						rc_r			r	r	r	r			

位 15:10 保留，必须保持复位值。

位 9 **TSTS**：时间戳触发状态 (Time stamp trigger status)
当系统时间值等于或大于目标时间高位与低位寄存器中指定的值时，此位置 1。对此寄存器执行读操作时此位清零。

位 8:7 保留，必须保持复位值。

位 6 **MMCTS**：MMC 发送状态 (MMC transmit status)
只要 ETH_MMCTIR Register 寄存器中生成中断，此位就为高电平。当此中断寄存器 (ETH_MMCTIR) 中的所有位都清零时，此位清零。

位 5 **MMCRS**：MMC 接收状态 (MMC receive status)
只要 ETH_MMCRIR 寄存器中生成中断，此位就为高电平。当此中断寄存器 (ETH_MMCRIR) 中的所有位都清零时，此位清零。

位 4 **MMCS**：MMC 状态 (MMC status)
只要位 6:5 中的任何一个位被设置为高电平，此位即会被设置为高电平。仅当这两个位均为低电平时，此位才被清零。

位 3 **PMTS**：PMT 状态 (PMT status)
只要在掉电模式下接收到魔术数据包或局域网唤醒帧，此位即被置 1（请参见 ETH_MACPMTCSR 寄存器中的位 5 和 6 *第 890 页的以太网 MAC PMT 控制和状态寄存器 (ETH_MACPMTCSR)*）。当此最后一个寄存器的两个位 [6:5] 因对 ETH_MACPMTCSR 寄存器执行读取操作而被清零时，此位也将会清零。

位 2:0 保留，必须保持复位值。

以太网 MAC 中断屏蔽寄存器 (ETH_MACIMR)

Ethernet MAC interrupt mask register

偏移地址：0x003C

复位值：0x0000 0000

通过使用 ETH_MACIMR 寄存器位，可屏蔽因 ETH_MACSR 寄存器中的相应事件导致的中断信号。

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

位 15:10 保留，必须保持复位值。

位 9 **TSTIM**：时间戳触发中断屏蔽 (Time stamp trigger interrupt mask)
如果此位置 1，则会禁止生成时间戳中断。

位 8:4 保留，必须保持复位值。

位 3 **PMTIM**：PMT 中断屏蔽 (PMT interrupt mask)
如果此位置 1，则会禁止因将 ETH_MACSR 中的 PMT 状态位置 1 而触发中断信号。

位 2:0 保留，必须保持复位值。

以太网 MAC 地址 0 高位寄存器 (ETH_MACA0HR)

Ethernet MAC address 0 high register

偏移地址：0x0040

复位值：0x0010 FFFF

MAC 地址 0 高位寄存器可保存站的第一个 6 字节 MAC 地址的高 16 位。请注意，在 MII 接口上接收到的第一个 DA 字节与 MAC 地址低位寄存器的 LS 字节（位 [7:0]）相对应。例如，如果在 MII 上将 0x1122 3344 5566（0x11 为第一个字节）作为目标地址进行接收，则 MAC 地址 0 寄存器 [47:0] 将与 0x6655 4433 2211 进行比较。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MO	Reserved															MACA0H															
1																rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31 **MO**：始终为 1 (Always 1)。

位 30:16 保留，必须保持复位值。

位 15:0 **MACA0H**：MAC 地址 0 高位 [47:32] (MAC address0 high [47:32])
此字段包含 6 字节 MAC 地址 0 的高 16 位 (47:32)。MAC 使用此字段过滤所接收的帧以及将 MAC 地址插入到发送流控制（暂停）帧中。

以太网 MAC 地址 0 低位寄存器 (ETH_MACA0LR)

Ethernet MAC address 0 low register

偏移地址：0x0044

复位值：0xFFFF FFFF

MAC 地址 0 低位寄存器可保存站的第一个 6 字节 MAC 地址的低 32 位。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MACA0L																															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:0 **MACA0L**：MAC 地址 0 低位 [31:0] (MAC address0 low [31:0])

此字段包含 6 字节 MAC 地址 0 的低 32 位。MAC 使用此字段过滤所接收的帧以及将 MAC 地址插入到发送流控制（暂停）帧中。

以太网 MAC 地址 1 高位寄存器 (ETH_MACA1HR)

Ethernet MAC address 1 high register

偏移地址：0x0048

复位值：0x0000 FFFF

MAC 地址 1 高位寄存器可保存站的第二个 6 字节 MAC 地址的高 16 位。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AE	SA	MBC						Reserved								MACA1H															
rw	rw	rw	rw	rw	rw	rw	rw									rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31 **AE**：地址使能 (Address enable)

此位置 1 时，地址过滤器使用 MAC 地址 1 实现完美过滤。此位清零时，地址过滤器会忽略用于过滤的地址。

位 30 **SA**：源地址 (Source address)

此位置 1 时，将使用 MAC 地址 1 [47:0] 与所接收帧的 SA 字段进行比较。
此位清零时，将使用 MAC 地址 1 [47:0] 与所接收帧的 DA 字段进行比较。

位 29:24 **MBC**：屏蔽字节控制 (Mask byte control)

这些位是用于比较每个 MAC 地址 1 字节的屏蔽控制位。当将它们设为高电平时，MAC 内核不会将所接收到的 DA/SA 的相应字节与 MAC 地址 1 寄存器的内容进行比较。每个位都用于控制字节的屏蔽，如下所示：

– 位 29：ETH_MACA1HR [15:8]
– 位 28：ETH_MACA1HR [7:0]
– 位 27：ETH_MACA1LR [31:24]
…
– 位 24：ETH_MACA1LR [7:0]

位 23:16 保留，必须保持复位值。

位 15:0 **MACA1H**：MAC 地址 1 高位 [47:32] (MAC address1 high [47:32])

此字段包含第二个 6 字节 MAC 地址的高 16 位 (47:32)。

以太网 MAC 地址 1 低位寄存器 (ETH_MACA1LR)

Ethernet MAC address1 low register

偏移地址：0x004C

复位值：0xFFFF FFFF

MAC 地址 1 低位寄存器可保存站的第二个 6 字节 MAC 地址的低 32 位。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MACA1L																															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:0 **MACA1L**：MAC 地址 1 低位 [31:0] (MAC address1 low [31:0])

此字段包含 6 字节 MAC 地址 1 的低 32 位。如果在初始化过程之后，应用程序未加载此字段的内容，则该内容将不会被定义。

以太网 MAC 地址 2 高位寄存器 (ETH_MACA2HR)

Ethernet MAC address 2 high register

偏移地址：0x0050

复位值：0x0000 FFFF

MAC 地址 2 高位寄存器可保存站的第二个 6 字节 MAC 地址的高 16 位。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AE	SA	MBC						Reserved								MACA2H															
rw	rw	rw	rw	rw	rw	rw	rw									rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31 **AE**：地址使能 (Address enable)

此位置 1 时，地址过滤器使用 MAC 地址 2 实现完美过滤。复位后，地址过滤器会忽略用于过滤的地址。

位 30 **SA**：源地址 (Source address)

此位置 1 时，将使用 MAC 地址 2 [47:0] 与所接收帧的 SA 字段进行比较。

此位被复位时，将使用 MAC 地址 2 [47:0] 与所接收帧的 DA 字段进行比较。

位 29:24 **MBC**：屏蔽字节控制 (Mask byte control)

这些位是用于比较每个 MAC 地址 2 字节的屏蔽控制位。当将其设为高电平时，MAC 内核不会将所接收到的 DA/SA 的相应字节与 MAC 地址 2 寄存器的内容进行比较。每个位都用于控制字节的屏蔽，如下所示：

– 位 29：ETH_MACA2HR [15:8]
– 位 28：ETH_MACA2HR [7:0]
– 位 27：ETH_MACA2LR [31:24]
…
– 位 24：ETH_MACA2LR [7:0]

位 23:16 保留，必须保持复位值。

位 15:0 **MACA2H**：MAC 地址 2 高位 [47:32] (MAC address2 high [47:32])

此字段包含 6 字节 MAC 地址 2 的高 16 位 (47:32)。

以太网 MAC 地址 2 低位寄存器 (ETH_MACA2LR)

Ethernet MAC address 2 low register

偏移地址：0x0054

复位值：0xFFFF FFFF

MAC 地址 2 低位寄存器可保存站的第二个 6 字节 MAC 地址的低 32 位。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MACA2L																															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:0 **MACA2L**：MAC 地址 2 低位 [31:0] (MAC address2 low [31:0])

此字段包含第二个 6 字节 MAC 地址 2 的低 32 位。如果在初始化过程之后，应用程序未加载此字段的内容，则该内容将不会被定义。

以太网 MAC 地址 3 高位寄存器 (ETH_MACA3HR)

Ethernet MAC address 3 high register

偏移地址：0x0058

复位值：0x0000 FFFF

MAC 地址 3 高位寄存器可保存站的第二个 6 字节 MAC 地址的高 16 位。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AE	SA	MBC						Reserved								MACA3H															
rw	rw	rw	rw	rw	rw	rw	rw									rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31 **AE**：地址使能 (Address enable)

此位置 1 时，地址过滤器使用 MAC 地址 3 实现完美过滤。此位清零时，地址过滤器会忽略用于过滤的地址。

位 30 **SA**：源地址 (Source address)

此位置 1 时，将使用 MAC 地址 3 [47:0] 与所接收帧的 SA 字段进行比较。

此位被清零时，将使用 MAC 地址 3 [47:0] 与所接收帧的 DA 字段进行比较。

位 29:24 **MBC**：屏蔽字节控制 (Mask byte control)

这些位是用于比较每个 MAC 地址 3 字节的屏蔽控制位。当将这些位设为高电平时，MAC 内核不会将所接收到的 DA/SA 的相应字节与 MAC 地址 3 寄存器的内容进行比较。每个位都用于控制字节的屏蔽，如下所示：

– 位 29：ETH_MACA3HR [15:8]
– 位 28：ETH_MACA3HR [7:0]
– 位 27：ETH_MACA3LR [31:24]
…
– 位 24：ETH_MACA3LR [7:0]

位 23:16 保留，必须保持复位值。

位 15:0 **MACA3H**：MAC 地址 3 高位 [47:32] (MAC address3 high [47:32])

该字段包含 6 字节 MAC 地址 3 的高 16 位 (47:32)。

以太网 MAC 地址 3 低位寄存器 (ETH_MACA3LR)

Ethernet MAC address 3 low register

偏移地址：0x005C

复位值：0xFFFF FFFF

MAC 地址 3 低位寄存器可保存站的第二个 6 字节 MAC 地址的低 32 位。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MACA3L																															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:0 **MACA3L**：MAC 地址 3 低位 [31:0] (MAC address3 low [31:0])

此字段包含第二个 6 字节 MAC 地址 3 的低 32 位。如果在初始化过程之后，应用程序未加载此字段的内容，则该内容将不会被定义。

29.8.2 MMC 寄存器说明

以太网 MMC 控制寄存器 (ETH_MMCCR)

Ethernet MMC control register

偏移地址：0x0100

复位值：0x0000 0000

以太网 MMC 控制寄存器用于建立管理计数器的工作模式。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																										MCFHP	MCP	MCF	ROR	CSR	CR
																										rw	rw	rw	rw	rw	rw

位 31:6 保留，必须保持复位值。

位 5 **MCFHP:** MMC 计数器全-半预设置 (MMC counter Full-Half preset)

当 MCFHP 为低电平且位 4 置 1 时，所有 MMC 计数器均预设为几乎一半值。所有帧计数器均预设为 0x7FFF_FFF0（一半值 - 16）

当 MCFHP 为高电平且位 4 置 1 时，所有 MMC 计数器均预设为几乎全值。所有帧计数器均预设为 0xFFFF_FFF0（全值 - 16）

位 4 **MCP:** MMC 计数器预设 (MMC counter preset)

该位置 1 时，所有计数器都将根据上面的位 5 初始化或预设为几乎全值或几乎一半值。该位将在 1 个时钟周期后自动清零。该位连同位 5 用于调试和测试由于 MMC 计数器变为半全值或全值而导致的中断产生。

位 3 **MCF**：MMC 计数器冻结 (MMC counter freeze)

该位置 1 时，会冻结所有 MMC 计数器，使其保持为当前值。（该位清零后，才会因存在发送帧或接收帧而对 MMC 计数器进行更新。在此模式下，“读取时复位”位置 1 时，如果读取任何 MMC 计数器，则该计数器也会清零。）

位 2 **ROR**：读取时复位 (Reset on read)
该位置 1 时，读取 MMC 计数器后，该计数器会复位为零（复位后自清）。读取最低有效字节通道（位 [7:0]）后，计数器会清零。

位 1 **CSR**：计数器停止翻转 (Counter stop rollover)
该位置 1 时，计数器达到其最大值后，不会返回到零。

位 0 **CR**：计数器复位 (Counter reset)
该位置 1 时，所有计数器都将复位。该位在 1 个时钟周期后自动清零。

以太网 MMC 接收中断寄存器 (ETH_MMCRIR)

Ethernet MMC receive interrupt register

偏移地址：0x0104

复位值：0x0000 0000

以太网 MMC 接收中断寄存器用于保持当接收统计计数器达到其最大值的一半时所生成的中断。（计数器的 MSB 置 1。）该寄存器是一个 32 位宽的寄存器。当读取引发中断的各个 MMC 计数器时，会将中断位清零。必须读取各个计数器的最低有效字节通道（位 [7:0]），才能将中断位清零。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														RGUFS	Reserved										RFAES	RFCES	Reserved				
														rc_r											rc_r	rc_r					

位 31:18 保留，必须保持复位值。

位 17 **RGUFS**：接收的良好单播帧状态 (Received Good Unicast Frames Status)
当接收到的良好单播帧计数器达到其最大值的一半时，该位置 1。

位 16:7 保留，必须保持复位值。

位 6 **RFAES**：接收的帧对齐错误状态 (Received frames alignment error status)
当存在对齐错误的接收帧计数器达到其最大值的一半时，该位置 1。

位 5 **RFCES**：接收的帧 CRC 错误状态 (Received frames CRC error status)
当存在 CRC 错误的接收帧计数器达到其最大值的一半时，该位置 1。

位 4:0 保留，必须保持复位值。

以太网 MMC 发送中断寄存器 (ETH_MMCTIR)

Ethernet MMC transmit interrupt register

偏移地址：0x0108

复位值：0x0000 0000

以太网 MMC 发送中断寄存器用于保持当发送统计计数器达到其最大值的一半时所生成的中断。（计数器的 MSB 置 1。）该寄存器是一个 32 位宽的寄存器。当读取引发中断的各个 MMC 计数器时，会将中断位清零。必须读取各个计数器的最低有效字节通道（位 [7:0]），才能将中断位清零。

31:22	21	20:16	15	14	13:0
Reserved	TGFS	Reserved	TGFMSCS	TGFSCS	Reserved
	rc_r		rc_r	rc_r	

位 31:22 保留，必须保持复位值。

位 21 **TGFS**：发送的良好帧状态 (Transmitted good frames status)
当发送的良好帧计数器达到其最大值的一半时，该位置 1。

位 20:16 保留，必须保持复位值。

位 15 **TGFMSCS**：多个冲突后发送的良好帧状态 (Transmitted good frames more single collision status)
多个冲突后发送的良好帧计数器达到其最大值的一半时，该位置 1。

位 14 **TGFSCS**：单个冲突后发送的良好帧状态 (Transmitted good frames single collision status)
单个冲突后发送的良好帧计数器达到其最大值的一半时，该位置 1。

位 13:0 保留，必须保持复位值。

以太网 MMC 接收中断屏蔽寄存器 (ETH_MMCRIMR)

Ethernet MMC receive interrupt mask register

偏移地址：0x010C

复位值：0x0000 0000

以太网 MMC 接收中断屏蔽寄存器用于保持当接收统计计数器达到其最大值的一半时所生成中断的屏蔽。（计数器的 MSB 置 1。）该寄存器是一个 32 位宽的寄存器。

31:18	17	16:7	6	5	4:0
Reserved	RGUFM	Reserved	RFAEM	RFCEM	Reserved
	rw		rw	rw	

位 31:18 保留，必须保持复位值。

位 17 **RGUFM**：接收的良好单播帧屏蔽 (Received good unicast frames mask)
如果该位置 1，当接收到的良好单播帧计数器达到其最大值的一半时，会屏蔽中断。

位 16:7 保留，必须保持复位值。

位 6 **RFAEM**：接收的帧对齐错误屏蔽 (Received frames alignment error mask)
如果该位置 1，当存在对齐错误的接收帧计数器达到其最大值的一半时，会屏蔽中断。

位 5 **RFCEM**：接收的帧 CRC 错误屏蔽 (Received frame CRC error mask)
如果该位置 1，当存在 CRC 错误的接收帧计数器达到其最大值的一半时，会屏蔽中断。

位 4:0 保留，必须保持复位值。

以太网 MMC 发送中断屏蔽寄存器 (ETH_MMCTIMR)

Ethernet MMC transmit interrupt mask register

偏移地址：0x0110

复位值：0x0000 0000

以太网 MMC 发送中断屏蔽寄存器用于保持当发送统计计数器达到其最大值的一半时所生成中断的屏蔽。（计数器的 MSB 置 1）。该寄存器是一个 32 位宽的寄存器。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved										TGFM	Reserved					TGFMSCM	TGFSCM	Reserved													
										rw						rw	rw														

位 31:22 保留，必须保持复位值。

位 21 **TGFM**：发送的良好帧屏蔽 (Transmitted good frames mask)
如果该位置 1，当发送的良好帧计数器达到其最大值的一半时，会屏蔽中断。

位 20:16 保留，必须保持复位值。

位 15 **TGFMSCM**：多个冲突后发送的良好帧屏蔽 (Transmitted good frames more single collision mask)
如果该位置 1，当多个冲突后发送的良好帧计数器达到其最大值的一半时，会屏蔽中断。

位 14 **TGFSCM**：单个冲突后发送的良好帧屏蔽 (Transmitted good frames single collision mask)
如果该位置 1，当单个冲突后发送的良好帧计数器达到其最大值的一半时，会屏蔽中断。

位 13:0 保留，必须保持复位值。

以太网 MMC 在单个冲突后发送的良好帧计数器寄存器 (ETH_MMCTGFSCCR)

Ethernet MMC transmitted good frames after a single collision counter register

偏移地址：0x014C

复位值：0x0000 0000

该寄存器包含在半双工模式下于单个冲突后成功发送的帧的数量。

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

位 31:0 **TGFSCC**：单个冲突后发送的良好帧计数器 (Transmitted good frames single collision counter)
单个冲突后发送的良好帧计数器。

以太网 MMC 在多个冲突后发送的良好帧计数器寄存器 (ETH_MMCTGFMSCCR)

Ethernet MMC transmitted good frames after more than a single collision counter register

偏移地址：0x0150

复位值：0x0000 0000

该寄存器包含在半双工模式下于多个冲突后成功发送的帧的数量。

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

位 31:0 **TGFMSCC**：多个冲突后发送的良好帧计数器 (Transmitted good frames more single collision counter)

多个冲突后发送的良好帧计数器

以太网 MMC 发送的良好帧计数器寄存器 (ETH_MMCTGFCR)

Ethernet MMC transmitted good frames counter register

偏移地址：0x0168

复位值：0x0000 0000

该寄存器包含发送的良好帧的数量。

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

位 31:0 **TGFC**：发送的良好帧计数器 (Transmitted good frames counter)

带有 CRC 错误计数器寄存器的以太网 MMC 接收帧 (ETH_MMCRFCECR)

Ethernet MMC received frames with CRC error counter register

偏移地址：0x0194

复位值：0x0000 0000

此寄存器包含接收的含 CRC 错误的帧数。

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

位 31:0 **RFCEC**：接收的帧 CRC 错误计数器 (Received frames CRC error counter)

接收的带有 CRC 错误计数器的帧

带有对齐错误计数器寄存器的以太网 MMC 接收帧 (ETH_MMCRFAECR)

Ethernet MMC received frames with alignment error counter register

偏移地址：0x0198

复位值：0x0000 0000

此寄存器包含接收的含对齐 (dribble) 错误的帧数。

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

位 31:0 **RFAEC**：接收的帧对齐错误计数器 (Received frames alignment error counter)
接收的带有对齐错误计数器的帧

MMC 接收的良好单播帧计数器寄存器 (ETH_MMCRGUFCR)

MMC received good unicast frames counter register

偏移地址：0x01C4

复位值：0x0000 0000

此寄存器包含接收的良好单播帧数。

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

位 31:0 **RGUFC**：接收的良好单播帧计数器 (Received good unicast frames counter)

29.8.3 IEEE 1588 时间戳寄存器

本节介绍用于支持符合 IEEE 1588 标准的精确网络时钟同步功能的寄存器。

以太网 PTP 时间戳控制寄存器 (ETH_PTPTSCR)

Ethernet PTP time stamp control register

偏移地址：0x0700

复位值：0x0000 00002000

此寄存器用于控制时间戳生成和更新逻辑。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved													TSPFFMAE	TSCNT		TSSMRME	TSSEME	TSSIPV4FE	TSSIPV6FE	TSSPTPOEFE	TSPTPPSV2E	TSSSR	TSSARFE	Reserved		TTSARU	TSITE	TSSTU	TSSTI	TSFCU	TSE
													rw	rw		rw	rw	rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw

位 31:19 保留，必须保持复位值。

位 18 **TSPFFMAE**：时间戳 PTP 帧过滤 MAC 地址使能 (Time stamp PTP frame filtering MAC address enable)

如果置 1，则当 PTP 直接通过以太网发送时，此位将使用 MAC 地址（除 MAC 地址 0）来过滤 PTP 帧。

位 17:16 **TSCNT**：时间戳时钟节点类型 (Time stamp clock node type)

可用的时钟节点类型如下：

00：普通时钟

01：边界时钟

10：端对端透明时钟

11：点对点透明时钟

位 15 **TSSMRME**：主节点相关消息时间戳快照使能 (Time stamp snapshot for message relevant to master enable)

当此位置 1 时，仅拍摄主节点相关消息的快照。当此位清零时，仅拍摄从节点相关消息的快照。这仅适用于普通时钟和边界时钟节点。

位 14 **TSSEME**：事件消息时间戳快照使能 (Time stamp snapshot for event message enable)

当此位置 1 时，仅拍摄事件消息的时间戳快照（SYNC、Delay_Re、Pdelay_Req 或 Pdelay_Resp）。当此位清零时，将拍摄除 Announce、Management 和 Signaling 以外其它所有消息的快照。

位 13 **TSSIPV4FE**：IPv4 帧时间戳快照使能 (Time stamp snapshot for IPv4 frames enable)

当此位置 1 时，将拍摄 IPv4 帧的时间戳快照。

位 12 **TSSIPV6FE**：IPv6 帧时间戳快照使能 (Time stamp snapshot for IPv6 frames enable)

当此位置 1 时，将拍摄 IPv6 帧的时间戳快照。

位 11 **TSSPTPOEFE**：PTP over ethernet 帧时间戳快照使能 (Time stamp snapshot for PTP over ethernet frames enable)

当此位置 1 时，将拍摄在以太网帧中也含有 PTP 消息 (PTP over Ethernet) 的帧的时间戳快照。默认情况下，将拍摄 UDP-IPEthernet PTP 数据包的快照。

位 10 **TSPTPPSV2E**：监听版本 2 格式时间戳 PTP 数据包使能 (Time stamp PTP packet snooping for version2 format enable)

当此位置 1 时，将使用版本 2 格式对 PTP 数据包进行监听。当此位清零时，将使用版本 1 格式对 PTP 数据包进行监听。

注意：IEEE 1588 版本 1 和版本 2 格式符合 IEEE 标准 1588-2008（版本为 IEEE STD.1588-2002).

位 9 **TSSSR**：时间戳亚秒翻转：数字或二进制翻转控制 (Time stamp subsecond rollover: digital or binary rollover control)

当此位置 1 时，如果亚秒计数器达到值 0x3B9A C9FF（十进制 999 999 999）并递增了时间戳（高位）秒数，则时间戳低位寄存器将翻转。

当此位清零时，亚秒寄存器的翻转值将达到 0x7FFF FFFF。亚秒增量必须根据 PTP 参考时钟频率和此位的值正确进行编程。

位 8 **TSSARFE**：所有接收帧的时间戳快照使能 (Time stamp snapshot for all received frames enable)
当此位置 1 时，时间戳快照会针对内核所接收的全部帧处于使能状态。

位 7:6 保留，必须保持复位值。

位 5 **TSARU**：时间戳加数寄存器更新 (Time stamp addend register update)
当此位置 1 时，时间戳加数寄存器的内容会被更新到 PTP 进行精密校准。更新结束时此位会清零。此寄存器位必须在读为零后，才可以将其置 1。

位 4 **TSITE**：时间戳中断触发使能 (Time stamp interrupt trigger enable)
当此位置 1 时，如果系统时间大于在目标时间寄存器中写入的值，将产生时间戳中断。产生时间戳触发中断时，此位会清零。

位 3 **TSSTU**：时间戳系统时间更新 (Time stamp system time update)
当此位置 1 时，系统时间将通过在时间戳高位更新寄存器和时间戳低位更新寄存器中指定的值进行更新（增至其中或从中减去）。必须将 TSSTU 和 TSSTI 两位均读为零，才可以将此位置 1。硬件中的更新完成后，此位会清零。

位 2 **TSSTI**：时间戳系统时间初始化 (Time stamp system time initialize)
当此位置 1 时，系统时间将通过在时间戳高位更新寄存器和时间戳低位更新寄存器中指定的值进行初始化（覆盖）。此位必须在读为零后，才可以将其置 1。初始化完成后，此位会清零。

位 1 **TSFCU**：时间戳精密更新或粗略更新 (Time stamp fine or coarse update)
置 1 时，此位表示将使用精密更新方法更新系统时间戳。清零时，此位表示将使用粗略方法更新系统时间戳。

位 0 **TSE**：时间戳使能 (Time stamp enable)
当此位置 1 时，时间戳针对发送和接收帧处于使能状态。当此位清零时，时间戳功能将挂起，不会为发送和接收帧添加时间戳。由于保持的系统时间处于挂起状态，所以在将此位置为高电平后，始终都需要初始化时间戳功能（系统时间）。

下表指出了拍摄快照时的各种消息（取决于事件消息寄存器设置的时钟、使能主节点和使能快照）。

Table 169. 时间戳快照对寄存器位的相依性

TSCNT（位 17:16）	TSSMRME（位 15）(1)	TSSEME（位 14）	拍摄快照时的消息
00 或 01	X(2)	0	SYNC、Follow_Up、Delay_Req、Delay_Resp
00 或 01	1	1	Delay_Req
00 或 01	0	1	SYNC
10	N/A	0	SYNC、Follow_Up、Delay_Req、Delay_Resp
10	N/A	1	SYNC、Follow_Up
11	N/A	0	SYNC、Follow_Up、Delay_Req、Delay_Resp、Pdelay_Req、Pdelay_Resp
11	N/A	1	SYNC、Pdelay_Req、Pdelay_Resp

1. N/A = 不适用。
2. X = 无关。

以太网 PTP 亚秒递增寄存器 (ETH_PTPSSIR)

Ethernet PTP subsecond increment register

偏移地址：0x0704

复位值：0x0000 0000

此寄存器包含亚秒寄存器递增时使用的 8 位值。在粗略更新模式下（ETH_PTPTSCR 中的 TSFCU 位），此寄存器中的值在每个 HCLK 时钟周期后都会被添加到系统时间。在精密更新模式下，此寄存器中的值将在累加器溢出时被添加到系统时间。

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

位 31:8 保留，必须保持复位值。

位 7:0 **STSSI**：系统时间亚秒增量 (System time subseconds)

此寄存器中编程的值在每次更新时都会被增加到系统时间亚秒值的内容中。

例如，要得到 20 ns 的精度，该值为：20 / 0.467 = ~ 43（或 0x2A）。

以太网 PTP 时间戳高位寄存器 (ETH_PTPTSHR)

Ethernet PTP time stamp high register

偏移地址：0x0708

复位值：0x0000 0000

此寄存器包含最高有效（高位）32 个时间位。这个只读寄存器包含秒系统时间值。时间戳高位寄存器与时间戳低位寄存器可指示由 MAC 保持的系统时间的当前值，但是，它可以连续进行更新。

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

位 31:0 **STS**：系统时间秒 (System time second)

此字段中的值表示由内核维持的以秒为单位的系统时间当前值。

以太网 PTP 时间戳低位寄存器 (ETH_PTPTSLR)

Ethernet PTP time stamp low register

偏移地址：0x070C

复位值：0x0000 0000

此寄存器包含最低有效（低位）32 个时间位。这个只读寄存器包含亚秒系统时间值。

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

位 31 **STPNS**：系统时间正负号 (System time positive or negative sign)

该位指示正负时间值。置 1 时，该位指示时间表示为负值。清零时，该位指示时间表示为正值。由于系统时间始终应为正值，因此该位通常为零。

位 30:0 **STSS**：系统时间亚秒 (System time subseconds)

该字段中的值包含亚秒时间表示，精度为 0.46 ns。

以太网 PTP 时间戳高位更新寄存器 (ETH_PTPTSHUR)

Ethernet PTP time stamp high update register

偏移地址：0x0710

复位值：0x0000 0000

该寄存器包含要写入、增加到系统时间值或从系统时间值中减去的最高有效（高）32 位时间。时间戳高位更新寄存器与时间戳低位更新寄存器一起用于初始化或更新 MAC 维护的系统时间。在将时间戳控制寄存器中的 TSSTI 或 TSSTU 位置 1 前，必须写入这两个寄存器。

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

位 31:0 **TSUS**：时间戳更新秒 (Time stamp update second)

该字段中的值指示要初始化或增加到系统时间中的时间，单位为秒。

以太网 PTP 时间戳低位更新寄存器 (ETH_PTPTSLUR)

Ethernet PTP time stamp low update register

偏移地址：0x0714

复位值：0x0000 0000

该寄存器包含要写入、增加到系统时间值或从系统时间值中减去的最低有效（低）32 位时间。

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

位 31 **TSUPNS**：时间戳更新正负号 (Time stamp update positive or negative sign)

该位指示正负时间值。置 1 时，该位指示时间表示为负值。清零时，该位指示时间表示为正值。当 TSSTI 置 1（系统时间初始化）时，该位应为零。如果当 TSSTU 置 1 时该位也置 1，则时间戳更新寄存器中的值要从系统时间中减去。否则，它要增加到系统时间中。

位 30:0 **TSUSS**：时间戳更新亚秒 (Time stamp update subseconds)

该字段中的值指示要初始化或增加到系统时间中的亚秒时间。该值的精度为 0.46 ns（换句话说，值 0x0000_0001 即为 0.46 ns）。

以太网 PTP 时间戳加数寄存器 (ETH_PTPTSAR)

Ethernet PTP time stamp addend register

偏移地址：0x0718

复位值：0x0000 0000

该寄存器由软件用来重新线性调节时钟频率，使其与主时钟频率匹配。仅当系统时间配置为微调更新模式（ETH_PTPTSCR 中的 TSFCU 位）时使用该寄存器值。该寄存器的内容在每个时钟周期添加到一个 32 位累加器中，系统时间在该累加器发生上溢时更新。

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

位 31:0 **TSA**：时间戳加数 (Time stamp addend)

该寄存器指示要添加到累加器寄存器以实现时间同步的 32 位时间值。

以太网 PTP 目标时间高位寄存器 (ETH_PTPTTHR)

Ethernet PTP target time high register

偏移地址：0x071C

复位值：0x0000 0000

该寄存器包含要与用于中断事件生成的系统时间进行比较的高 32 位时间。目标时间高位寄存器与目标时间低位寄存器一起，用于在系统时间超过这些寄存器中编程的值时规划中断事件（ETH_PTPTSCR 中的 TSARU 位）。

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

位 31:0 **TTSH：**目标时间戳高位 (Target time stamp high)

该寄存器存储以秒为单位的时间。当时间戳的值匹配或同时超过两个目标时间戳寄存器时，MAC（如果已使能）会生成中断。

以太网 PTP 目标时间低位寄存器 (ETH_PTPTTLR)

Ethernet PTP target time low register

偏移地址：0x0720

复位值：0x0000 0000

该寄存器包含要与用于中断事件生成的系统时间进行比较的低 32 位时间。

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

位 31:0 **TTSL：**目标时间戳低位 (Target time stamp low)

该寄存器存储以纳秒为单位的时间（带符号）。当时间戳的值匹配或同时超过两个目标时间戳寄存器时，MAC（如果已使能）会生成中断。

以太网 PTP 时间戳状态寄存器 (ETH_PTPTSSR)

Ethernet PTP time stamp status register

偏移地址：0x0728

复位值：0x0000 0000

该寄存器包含时间戳状态寄存器。

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

位 31:2 保留，必须保持复位值。

位 1 **TSTTR**：达到时间戳目标时间 (Time stamp target time reached)
置 1 时，该位指示系统时间值大于或等于在目标时间高位和低位寄存器中指定的值。

位 0 **TSSO**：时间戳秒上溢 (Time stamp second overflow)
置 1 时，该位指示时间戳的秒值已上溢，超过 0xFFFF FFFF。

以太网 PTP PPS 控制寄存器 (ETH_PTPPPSCR)

Ethernet PTP PPS control register

偏移地址：0x072C

复位值：0x0000 0000

该寄存器控制 PPS 输出的频率。

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

位 31:4 保留，必须保持复位值。

位 3:0 **PPSFREQ**：PPS 频率选择 (PPS frequency selection)
PPS 输出频率设置为 $2^{PPSFREQ}$ Hz。
0000：1 Hz，使用二进制翻转时，脉冲宽度为 125 ms，使用数字翻转时，脉冲宽度为 100 ms
0001：2 Hz，使用二进制翻转时，占空比为 50%（不建议使用数字翻转）
0010：4 Hz，使用二进制翻转时，占空比为 50%（不建议使用数字翻转）
0011：8 Hz，使用二进制翻转时，占空比为 50%（不建议使用数字翻转）
0100：16 Hz，使用二进制翻转时，占空比为 50%（不建议使用数字翻转）
...
1111：32768 Hz，使用二进制翻转时，占空比为 50%（不建议使用数字翻转）

注意：如果使用数字翻转（TSSSR = 1，ETH_PTPTSCR 中的位 9），建议不要使用频率不是 1 Hz 的 PPS 输出。否则，使用数字翻转时，PPS 输出在更高的频率下的波形会不规则（尽管其平均频率在任何一秒窗口期间始终正确）。

29.8.4 DMA 寄存器说明

本节定义了每个 DMA 寄存器的位。只要地址是字对齐的，就允许非 32 位访问。

以太网 DMA 总线模式寄存器 (ETH_DMABMR)

Ethernet DMA bus mode register

偏移地址：0x1000

复位值：0x0000 2101

总线模式寄存器为 DMA 建立总线工作模式。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved					MB	AAB	FPM	USP	RDP						FB	PM		PBL						EDFE	DSL					DA	SR
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rs

位 31:27 保留，必须保持复位值。

位 26 **MB:** 混合突发 (Mixed burst)

该位置为高电平且 FB 位置为低电平时，AHB 主接口将以 INCR（未定义突发）启动所有长度大于 16 的突发传输。该位清零时，对于长度等于或小于 16 的突发，将恢复为固定突发传输（INCRx 和 SINGLE）。

位 25 **AAB**：地址对齐的节拍 (Address-aligned beats)

当该位设置为高电平并且 FB 位等于 1 时，AHB 接口会生成与起始地址 LS 位对齐的所有突发。如果 FB 位等于 0，则第一个突发（访问数据缓冲器的起始地址）不对齐，但后续的突发与地址对齐。

位 24 **FPM**：4xPBL 模式 (4xPBL mode)

当设置为高电平时，该位会将编程的 PBL 值（位 [22:17] 和位 [13:8]）乘以四倍。因此，DMA 根据 PBL 值以最大 4、8、16、32、64 和 128 个节拍传输数据。

位 23 **USP**：使用单独的 PBL (Use separate PBL)

设置为高电平时，它会配置 RxDMA，将位 [22:17] 中配置的值用作 PBL，而位 [13:8] 中的 PBL 值仅适用于 TxDMA 操作。当该位清零时，位 [13:8] 中的 PBL 值同时适用于两个 DMA 引擎。

位 22:17 **RDP**：Rx DMA PBL

这些位指示要在一个 RxDMA 事务中传输的最大节拍数。这是在单个块读/写操作中使用的最大值。RxDMA 每次在主机总线上开始突发传输时，始终尝试按 RDP 中指定的方式进行突发。允许使用值 1、2、4、8、16 和 32 对 RDP 进行编程。任何其它值都会产生未定义的行为。

仅在 USP 设置为高电平时这些位才有效且适用。

位 16 **FB**：固定突发 (Fixed burst)

该位控制 AHB 主接口是否执行固定突发传输。置 1 时，AHB 在正常突发传输开始期间仅使用 SINGLE、INCR4、INCR8 或 INCR16。复位时，AHB 使用 SINGLE 和 INCR 突发传输操作。

位 15:14 **PM**：Rx Tx 优先级比 (Rx Tx priority ratio)

RxDMA 请求优先于 TxDMA 请求，优先级比如下：

00：1:1
01：2:1
10：3:1
11：4:1
这仅在 DA 位清零时有效。

位 13:8 **PBL**：可编程突发长度 (Programmable burst length)

这些位指示要在一个 DMA 事务中传输的最大节拍数。这是在单个块读/写操作中使用的最大值。DMA 每次在主机总线上开始突发传输时，始终尝试按 PBL 中指定的方式进行突发。允许使用值 1、2、4、8、16 和 32 对 PBL 进行编程。任何其它值都会产生未定义的行为。当 USP 置 1 时，此 PBL 值仅适用于 TxDMA 事务。

PBL 值有以下限制：

– 可能的最大节拍数 (PBL) 受 Tx FIFO 和 Rx FIFO 大小的限制。
– FIFO 有一个限制，即支持的最大节拍数等于 FIFO 深度的一半。
– 如果 PBL 由发送和接收 DMA 共用，则必须考虑 Rx FIFO 和 Tx FIFO 的最小深度。
– 请不要编程超出范围的 PBL 值，因为系统可能发生异常行为。

位 7 **EDFE:** 增强描述符格式使能 (Enhanced descriptor format enable)

该位置 1 时，使能增强描述符格式，并将描述符大小增加至 32 字节（8 个 DWORD）。如果已激活时间戳功能（ETH_PTPTSCR 位 0 TSE=1）或 IPv4 校验和减荷（ETH_MACCR 位 10 IPCO=1），则必须使用此增强描述符。

位 6:2 **DSL**：描述符跳过长度 (Descriptor skip length)

该位指定两个未链接描述符之间跳过的字数。地址从当前描述符结束处开始跳到下一个描述符起始处。当 DSL 值等于零时，在环形模式下，DMA 会将描述符表视为连续的。

位 1 **DA**：DMA 仲裁 (DMA Arbitration)

0：循环调度，Rx:Tx 优先级比在位 [15:14] 中给出
1：Rx 优先于 Tx

位 0 **SR**：软件复位 (Software reset)

当该位置 1 时，MAC DMA 控制器会复位所有 MAC 子系统的内部寄存器和逻辑。在所有内核时钟域完成复位操作后，该位自动清零。重新编程任何内核寄存器之前，在该位中读取 0 值。

以太网 DMA 发送轮询要求寄存器 (ETH_DMATPDR)

Ethernet DMA transmit poll demand register

偏移地址：0x1004

复位值：0x0000 0000

应用程序使用此寄存器来指示 DMA 轮询发送描述符列表。发送轮询要求寄存器使能发送 DMA 来检查当前描述符是否为 DMA 所有。如果 TxDMA 处于挂起模式，则发出发送轮询要求命令将其唤醒。如果发送帧中出现下溢错误或发送 DMA 所拥有的描述符不可用，则 TxDMA 会进入挂起模式。用户可以随时发出此命令，当该命令开始重新获取主机存储器的当前描述符后，TxDMA 即会对其进行复位。

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

位 31:0 TPD：发送轮询要求 (Transmit poll demand)

向这些位写入任何值时，DMA 都会读取 ETH_DMACHTDR 寄存器指向的当前描述符。如果该描述符不可用（由主机所有），则发送会返回到挂起状态，并将 ETH_DMASR 寄存器位 2 进行置位。如果该描述符可用，则发送会继续进行。

以太网 DMA 接收轮询要求寄存器 (ETH_DMARPDR)

EHERNET DMA receive poll demand register

偏移地址：0x1008

复位值：0x0000 0000

应用程序使用此寄存器来指示 DMA 轮询接收描述符列表。接收轮询要求寄存器会使能接收 DMA 来检查新描述符。此命令用于将 RxDMA 从挂起状态唤醒。仅当 RxDMA 所拥有的描述符不可用时，它才会进入挂起状态。

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

位 31:0 **RPD**：接收轮询要求 (Receive poll demand)

向这些位写入任何值时，DMA 都会读取 ETH_DMACHRDR 寄存器指向的当前描述符。如果该描述符不可用（由主机所有），则接收会返回到挂起状态，且不会将 ETH_DMASR 寄存器位 7 进行置位。如果该描述符可用，接收 DMA 将返回到活动状态。

以太网 DMA 接收描述符列表地址寄存器 (ETH_DMARDLAR)

Ethernet DMA receive descriptor list address register

偏移地址：0x100C

复位值：0x0000 0000

接收描述符列表地址寄存器会指向接收描述符列表的起始处。描述符列表位于 STM32F4xx 的物理存储器空间，且必须为字对齐。DMA 会将描述符列表的对应 LS 位置为低电平，进而在内部将其转换为总线宽度对齐地址。仅当接收停止时，才允许对 ETH_DMARDLAR 寄存器执行写操作。停止后，必须先对 ETH_DMARDLAR 寄存器执行写操作，然后才能发出接收启动命令。

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

位 31:0 **SRL**：接收列表的起始处 (Start of receive list)

此字段包含接收描述符列表中的首个描述符的基址。DMA 会在内部将 LSB 位 [1/2/3:0]（对应于 32/64/128 位的总线宽度）忽略，并将其值均视为零。因此，这些 LSB 位为只读。

以太网 DMA 发送描述符列表地址寄存器 (ETH_DMATDLAR)

Ethernet DMA transmit descriptor list address register

偏移地址：0x1010

复位值：0x0000 0000

发送描述符列表地址寄存器会指向发送描述符列表的起始处。描述符列表位于 STM32F4xx 的物理存储器空间，且必须为字对齐。DMA 会将描述符列表的对应 LSB 位置为低电平，进而在内部将其转换为总线宽度对齐地址。仅当发送停止时，才允许对 ETH_DMATDLAR 寄存器执行写操作。发送停止后，可以先对 ETH_DMATDLAR 寄存器执行写操作，然后再发出发送启动命令。

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

位 31:0 **STL**：发送列表的起始处 (Start of transmit list)

此字段包含发送描述符列表中的首个描述符的基址。DMA 会在内部将 LSB 位 [1/2/3:0]（对应于 32/64/128 位的总线宽度）忽略，并将其值均视为零。因此，这些 LSB 位为只读。

以太网 DMA 状态寄存器 (ETH_DMASR)

Ethernet DMA status register

偏移地址：0x1014

复位值：0x0000 0000

状态寄存器包含所有 DMA 向应用程序报告的状态位。软件驱动程序通常在中断服务例程或轮询期间读取 ETH_DMASR 寄存器。此寄存器中的大部分字段会导致主机中断。读取时 ETH_DMASR 寄存器位不会清零。向 ETH_DMASR 寄存器 [16:0] 中的（未保留）位写入 1 会将其清零，写入 0 则不起作用。通过屏蔽 ETH_DMAIER 寄存器中的位，可以屏蔽各个对应字段（位 [16:0]）。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		TSTS	PMTS	MMCS	Reserved	EBS			TPS			RPS			NIS	AIS	ERS	FBES	Reserved		ETS	RWTS	RPSS	RBUS	RS	TUS	ROS	TJTS	TBUS	TPSS	TS
		r	r	r		r	r	r	r	r	r	r	r	r	rc-w1	rc-w1	rc-w1	rc-w1			rc-w1	rc-w1	rc-w1	rc-w1	rc-w1	rc-w1	rc-w1	rc-w1	rc-w1	rc-w1	rc-w1

位 31:30 保留，必须保持复位值。

位 29 **TSTS**：时间戳触发状态 (Time stamp trigger status)

此位指示 MAC 内核的时间戳发生器块中发生的中断事件。软件必须读取 MAC 内核的状态寄存器，将其源（位 9）清零，才能将此位复位为 0。当此位处于高电平时，使能后会产生中断。

位 28 **PMTS**：PMT 状态 (PMT status)

此位指示 MAC 内核的 PMT 中发生的事件。软件必须读取 MAC 内核中对应的寄存器，以获取产生中断的具体原因，然后将其源清零，才能将此位复位为 0。当此位处于高电平时，使能后会产生中断。

位 27 **MMCS**：MMC 状态 (MMC status)

此位反映 MAC 内核的 MMC 中发生的事件。软件必须读取 MAC 内核中对应的寄存器，以获取产生中断的具体原因，然后将中断源清零，才能将此位置 0。当此位处于高电平时，使能后会产生中断。

位 26 保留，必须保持复位值。

位 25:23 **EBS**：错误位状态 (Error bits status)

这些位指示导致总线错误（AHB 接口上的错误响应）的错误类型。仅在致命总线错误位（ETH_DMASR 寄存器 [13]）置 1 时有效。此字段不会产生中断。

位 23
- 1 TxDMA 传输数据时出错 (Error during data transfer by TxDMA)
- 0 RxDMA 传输数据时出错 (Error during data transfer by RxDMA)

位 24
- 1 读取传输时出错 (Error during read transfer)
- 0 写入传输时出错 (Error during write transfer)

位 25
- 1 访问描述符时出错 (Error during descriptor access)
- 0 访问数据缓冲区时出错 (Error during data buffer access)

位 22:20 **TPS**：发送过程状态 (Transmit process state)

这些位指示发送 DMA FSM 的状态。此字段不会产生中断。

000：停止；发出复位或停止发送命令

001：运行中；正在获取发送传输描述符

010：运行中；正在等待状态

011：运行中；正在读取主机存储器缓冲区中的数据并将其加入发送缓冲区 (Tx FIFO) 队列

100、101：预留给将来使用

110：挂起；发送描述符不可用或发送缓冲区下溢

111：运行中；正在关闭发送描述符

位 19:17 **RPS**：接收过程状态 (Receive process state)

这些位指示接收 DMA FSM 的状态。此字段不会产生中断。

000：停止：发出复位或停止接收命令

001：运行中：正在获取接收传输描述符

010：预留给将来使用

011：运行中：正在等待接收数据包

100：挂起：接收描述符不可用

101：运行中：正在关闭接收描述符

110：预留给将来使用

111：运行中：将接收数据包的数据从接收缓冲区传输到主机存储器

位 16 **NIS**：正常中断汇总 (Normal interrupt summary)

当使能 ETH_DMAIER 寄存器中对应的中断位时，正常中断汇总位的值是以下位的逻辑或运算结果：

– ETH_DMASR [0]：发送中断 (Transmit interrupt)
– ETH_DMASR [2]：发送缓冲区不可用 (Transmit buffer unavailable)
– ETH_DMASR [6]：接收中断 (Receive interrupt)
– ETH_DMASR [14]：提前接收中断 (Early receive interrupt)

只有未屏蔽的位会影响正常中断汇总位。

它是黏着位，每当导致 NIS 位置 1 的对应位被清零时，必须同时将它也清零（通过向此位写入 1）。

位 15 **AIS**：异常中断汇总 (Abnormal interrupt summary)

当使能 ETH_DMAIER 寄存器中对应的中断位时，异常中断汇总位的值是以下位的逻辑或运算结果：

– ETH_DMASR [1]：发送过程停止 (Transmit process stopped)
– ETH_DMASR [3]：发送 jabber 超时 (Transmit jabber timeout)
– ETH_DMASR [4]：接收 FIFO 上溢 (Receive FIFO overflow)
– ETH_DMASR [5]：发送下溢 (Transmit underflow)
– ETH_DMASR [7]：接收缓冲区不可用 (Receive buffer unavailable)
– ETH_DMASR [8]：接收过程停止 (Receive process stopped)
– ETH_DMASR [9]：接收看门狗超时 (Receive watchdog timeout)
– ETH_DMASR [10]：提前发送中断 (Early transmit interrupt)
– ETH_DMASR [13]：致命总线错误 (Fatal bus error)

只有未屏蔽的位会影响异常中断汇总位。

它是黏着位，每当导致 AIS 位置 1 的对应位被清零时，必须同时将它也清零。

位 14 **ERS**：提前接收状态 (Early receive status)

此位指示 DMA 已填满数据包的首个数据缓冲区。接收中断 ETH_DMASR [6] 会自动将此位清零。

位 13 **FBES**：致命总线错误状态 (Fatal bus error status)
此位指示发生了总线错误，具体信息参见位 [25:23]。此位置 1 后，对应的 DMA 引擎会禁止其所有的总线访问。

位 12:11 保留，必须保持复位值。

位 10 **ETS**：提前发送状态 (Early transmit status)
此位指示要发送的帧已完全传输到发送 FIFO。

位 9 **RWTS**：接收看门狗超时状态 (Receive watchdog timeout status)
当接收到的帧的长度大于 2048 个字节时，会对此位进行置位。

位 8 **RPSS**：接收过程停止状态 (Receive process stopped status)
当接收过程进入停止状态时，会对此位进行置位。

位 7 **RBUS**：接收缓冲区不可用状态 (Receive buffer unavailable status)
此位指示接收列表中的下一个描述符由主机所拥有，DMA 无法获取。接收过程进入挂起状态。要恢复处理接收描述符，主机应更改描述符的拥有关系，然后发出接收轮询要求命令。如果未发出接收轮询要求命令，则当接收到下一个识别的传入帧时，接收过程会恢复。仅当上一接收描述符由 DMA 所拥有时，才能将 ETH_DMASR [7] 置 1。

位 6 **RS**：接收状态 (Receive status)
此位指示帧接收已完成。特定的帧状态信息已发布在描述符中。接收保持运行状态。

位 5 **TUS**：发送下溢状态 (Transmit underflow status)
此位指示在帧发送期间，发送缓冲区发生下溢。发送会进入挂起状态，且下溢错误 TDES0[1] 置 1。

位 4 **ROS**：接收上溢状态 (Receive overflow status)
此位指示在帧接收期间，接收缓冲区发生上溢。如果部分帧已传输到应用程序，则 RDES0[11] 中的上溢状态位置 1。

位 3 **TJTS**：发送 jabber 超时状态 (Transmit jabber timeout status)
此位指示发送 jabber 定时器已过期，这意味着发送器过度有效。发送过程会中止并将其置于停止状态。这会导致对发送 jabber 超时 TDES0[14] 标志进行置位。

位 2 **TBUS**：发送缓冲区不可用状态 (Transmit buffer unavailable status)
此位指示发送列表中的下一个描述符由主机所拥有，DMA 无法获取。发送会进入挂起状态。位 [22:20] 用于解释发送过程状态转换。要恢复处理发送描述符，主机应更改描述符位的拥有关系，然后发出发送轮询要求命令。

位 1 **TPSS**：发送过程停止状态 (Transmit process stopped status)
当发送停止时，此位置 1。

位 0 **TS**：发送状态 (Transmit status)
此位指示帧发送已完成，且首个描述符中的 TDES1[31] 位已置 1。

以太网 DMA 工作模式寄存器 (ETH_DMAOMR)

Ethernet DMA operation mode register

偏移地址：0x1018

复位值：0x0000 0000

工作模式寄存器将建立发送和接收工作模式及命令。作为 DMA 初始化过程的一部分，应将 ETH_DMAOMR 寄存器作为最后的 CSR 进行写操作。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved					DTCEFD	RSF	DFRF	Reserved		TSF	FTF	Reserved			TTC			ST	Reserved					FEF	FUGF	Reserved	RTC		OSF	SR	Reserved
					rw	rw	rw			rw	rs				rw	rw	rw	rw						rw	rw		rw	rw	rw	rw	

位 31:27 保留，必须保持复位值。

位 26 **DTCEFD**：禁止丢弃 TCP/IP 校验和错误帧 (Dropping of TCP/IP checksum error frames disable)

当此位置 1 时，如果帧中仅存在由接收校验和减荷引擎检测出来的错误，则内核不会丢弃它。这类帧在 MAC 接收到的以太网帧中没有任何错误（包括 FCS 错误），而仅在封装的有效负载中有错误。当此位清零时，如果 FEF 位进行了复位，则会丢弃所有错误帧。

位 25 **RSF**：接收存储并转发 (Receive store and forward)

当此位置 1 时，向 Rx FIFO 写入完整帧后，可从中读取一个帧，同时忽略 RTC 位。当此位清零时，Rx FIFO 在直通模式下工作，具体取决于 RTC 位指定的阈值。

位 24 **DFRF**：禁止刷新接收帧 (Disable flushing of received frames)

当此位置 1 时，RxDMA 不会因为接收描述符/缓冲区不可用而刷新任何帧（通常情况下，当此位清零时它会这样做）。（请参见*第 873 页的接收过程挂起*）

位 23:22 保留，必须保持复位值。

位 21 **TSF**：发送存储并转发 (Transmit store and forward)

当此位置 1 时，如果发送 FIFO 中有一个完整帧，则发送会启动。当此位置 1 时，会忽略由 ETH_DMAOMR 寄存器位 [16:14] 指定的 TTC 值。

该位清零后，ETH_DMAOMR 寄存器位 [16:14] 指定的 TTC 值才会有效。

该位只有在已停止传输时才能更改。

位 20 **FTF**：刷新发送 FIFO (Flush transmit FIFO)

该位置 1 时，发送 FIFO 控制器逻辑会复位为默认值，因此，Tx FIFO 中的所有数据均会丢失/刷新。刷新操作结束时该位在内部清零。此位清零之前不得对工作模式寄存器执行写操作。

位 19:17 保留，必须保持复位值。

位 16:14 **TTC**：发送阈值控制 (Transmit threshold control)

这三个位用于控制发送 FIFO 的阈值级别。当发送 FIFO 中的帧大小大于阈值时启动发送。此外，还会发送长度小于阈值的全帧。这些位只有在 TSF 位（位 21）清零后才能使用。

000：64
001：128
010：192
011：256
100：40
101：32
110：24
111：16

位 13 **ST**：启动/停止发送 (Start/stop transmission)

该位置 1 时，发送过程会进入运行状态，DMA 会检查当前位置的发送列表查找待发送的帧。可通过发送列表中的当前位置，即由 ETH_DMATDLAR 寄存器设定的基址，或者上一次停止发送时的保留位置尝试获取描述符。如果当前描述符不属于 DMA，则发送过程会进入挂起状态，且发送缓冲区不可用位 (ETH_DMASR [2]) 置 1。启动发送命令只有在传输已停止时才有效。如果该命令在设置 DMA ETH_DMATDLAR 寄存器之前发出，则 DMA 行为无法预知。

该位清零时，发送过程会在完成发送当前帧的任务之后进入停止状态，并保存发送列表中的下一个描述符位置，该位置在重启发送后会成为当前位置。停止发送命令只有在当前帧发送过程结束或发送过程处于挂起状态时才有效。

位 12:8 保留，必须保持复位值。

位 7 **FEF**：转发错误帧 (Forward error frame)

该位置 1 时，除短错误帧之外的所有帧都会转发到 DMA。

该位清零时，Rx FIFO 会丢弃带有错误状态（CRC 错误、冲突错误、巨帧、看门狗超时、上溢）的帧。不过，如果某个帧的起始字节（写）指针已传输到读控制器端（阈值模式下），则不会丢弃该帧。如果 ARI 总线上未传输（输出）帧的起始字节，则 Rx FIFO 会丢弃此错误帧。

位 6 **FUGF**：转发过小的好帧 (Forward undersized good frame)

该位置 1 时，Rx FIFO 会转发包含 pad 字节和 CRC 的过小帧（无错误但长度不足 64 字节的帧）。

该位清零时，Rx FIFO 会丢弃所有不足 64 字节的帧，除非因接收阈值下限更低（例如 RTC = 01）而导致此类帧已传输。

位 5 保留，必须保持复位值。

位 4:3 **RTC**：接收阈值控制 (Receive threshold control)

这两个位用于控制接收 FIFO 的阈值级别。当接收 FIFO 中的帧大小大于阈值时启动 DMA 传输（请求）。此外，长度小于阈值的全帧会自动传输。

注意：如果配置的接收 FIFO 大小为 128 字节，则不适合使用 11 作为阈值。

注意：这些位只有在 RSF 位为 0 时才有效，而当 RSF 位置 1 后会忽略这些位。

00：64
01：32
10：96
11：128

位 2 **OSF**：处理第二个帧 (Operate on second frame)

该位置 1 时会命令 DMA 处理第二个发送数据帧，即使尚未获得首个帧的状态。

位 1 **SR**：启动/停止接收 (Start/stop receive)

该位置 1 时，接收过程会进入运行状态。DMA 尝试从接收列表中获取描述符并处理传入帧。可通过列表中的当前位置，由 DMA ETH_DMARDLAR 寄存器设定的地址，或者上一次停止接收时的保留位置尝试获取描述符。如果 DMA 未占据任何描述符，则接收过程会进入挂起状态，且接收缓冲区不可用位 (ETH_DMASR [7]) 置 1。启动接收命令只有在已停止接收时才有效。如果该命令在设置 DMA ETH_DMARDLAR 寄存器之前发出，则 DMA 行为无法预知。

该位清零时，RxDMA 会在传输当前帧之后停止操作，并保存接收列表中的下一个描述符位置，该位置在接收过程重启后会成为当前位置。停止接收命令只有在当前接收过程处于运行（等待接收数据包）或挂起状态时才有效。

位 0 保留，必须保持复位值。

以太网 DMA 中断使能寄存器 (ETH_DMAIER)

Ethernet DMA interrupt enable register

偏移地址：0x101C

复位值：0x0000 0000

中断使能寄存器可使能 ETH_DMASR 报告的中断。将一个位置 1 可使能相应的中断。完成硬件或软件复位之后，会禁止所有中断。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															NISE	AISE	ERIE	FBEIE	Reserved		ETIE	RWTIE	RPSIE	RBUIE	RIE	TUIE	ROIE	TJTIE	TBUIE	TPSIE	TIE
															rw	rw	rw	rw			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:17 保留，必须保持复位值。

位 16 **NISE**：正常中断汇总使能 (Normal interrupt summary enable)

该位置 1 时，会使能一个正常中断。该位清零时，会禁止一个正常中断。该位可使能以下位：

– ETH_DMASR [0]：发送中断
– ETH_DMASR [2]：发送缓冲区不可用
– ETH_DMASR [6]：接收中断
– ETH_DMASR [14]：提前接收中断

位 15 **AISE**：异常中断汇总使能 (Abnormal interrupt summary enable)

该位置 1 时，会使能一个异常中断。该位清零时，会禁止一个异常中断。该位可使能以下位：

– ETH_DMASR [1]：发送过程停止
– ETH_DMASR [3]：发送 jabber 超时
– ETH_DMASR [4]：接收上溢
– ETH_DMASR [5]：发送下溢
– ETH_DMASR [7]：接收缓冲区不可用
– ETH_DMASR [8]：接收过程停止
– ETH_DMASR [9]：接收看门狗超时
– ETH_DMASR [10]：提前发送中断
– ETH_DMASR [13]：致命总线错误

位 14 **ERIE**：提前接收中断使能 (Early receive interrupt enable)

当该位通过正常中断汇总使能位（ETH_DMAIER 寄存器 [16]）置 1 时，可使能提前接收中断。

该位清零时，会禁止提前接收中断。

位 13 **FBEIE**：致命总线错误中断使能 (Fatal bus error interrupt enable)

当该位通过异常中断汇总使能位（ETH_DMAIER 寄存器 [15]）置 1 时，可使能致命总线错误中断。

该位清零时，会禁止致命总线使能中断。

位 12:11 保留，必须保持复位值。

位 10 **ETIE**：提前发送中断使能 (Early transmit interrupt enable)

当该位通过异常中断汇总使能位（ETH_DMAIER 寄存器 [15]）置 1 时，可使能提前发送中断。

该位清零时，会禁止提前发送中断。

位 9 **RWTIE**：接收看门狗超时中断使能 (receive watchdog timeout interrupt enable)
当该位通过异常中断汇总使能位（ETH_DMAIER 寄存器 [15]）置 1 时，可使能接收看门狗超时中断。
该位清零时，会禁止接收看门狗超时中断。

位 8 **RPSIE**：接收过程停止中断使能 (Receive process stopped interrupt enable)
当该位通过异常中断汇总使能位（ETH_DMAIER 寄存器 [15]）置 1 时，可使能接收停止中断。该位清零时，会禁止接收停止中断。

位 7 **RBUIE**：接收缓冲区不可用中断使能 (Receive buffer unavailable interrupt enable)
当该位通过异常中断汇总使能位（ETH_DMAIER 寄存器 [15]）置 1 时，可使能接收缓冲区不可用中断。
该位清零时，会禁止接收缓冲区不可用中断。

位 6 **RIE**：接收中断使能 (Receive interrupt enable)
当该位通过正常中断汇总使能位（ETH_DMAIER 寄存器 [16]）置 1 时，可使能接收中断。
该位清零时，会禁止接收中断。

位 5 **TUIE**：下溢中断使能 (Underflow interrupt enable)
当该位通过异常中断汇总使能位（ETH_DMAIER 寄存器 [15]）置 1 时，可使能发送下溢中断。
该位清零时，会禁止下溢中断。

位 4 **ROIE**：上溢中断使能 (Overflow interrupt enable)
当该位通过异常中断汇总使能位（ETH_DMAIER 寄存器 [15]）置 1 时，可使能接收上溢中断。
该位清零时，会禁止上溢中断。

位 3 **TJTIE**：发送 jabber 超时中断使能 (Transmit jabber timeout interrupt enable)
当该位通过异常中断汇总使能位（ETH_DMAIER 寄存器 [15]）置 1 时，可使能发送 jabber 超时中断。
该位清零时，会禁止发送 jabber 超时中断。

位 2 **TBUIE**：发送缓冲区不可用中断使能 (Transmit buffer unavailable interrupt enable)
当该位通过正常中断汇总使能位（ETH_DMAIER 寄存器 [16]）置 1 时，可使能发送缓冲区不可用中断。
该位清零时，会禁止发送缓冲区不可用中断。

位 1 **TPSIE**：发送过程停止中断使能 (Transmit process stopped interrupt enable)
当该位通过异常中断汇总使能位（ETH_DMAIER 寄存器 [15]）置 1 时，可使能发送停止中断。
该位清零时，会禁止发送停止中断。

位 0 **TIE**：发送中断使能 (Transmit interrupt enable)
当该位通过正常中断汇总使能位（ETH_DMAIER 寄存器 [16]）置 1 时，可使能发送中断。
该位清零时，会禁止发送中断。

只有 DMA 状态寄存器的 TSTS 或 PMTS 位已置位但相应的中断均未带标记时，或者 NIS/AIS 状态位已置位并且相应的中断使能位 (NISE/AISE) 已使能时，才会生成以太网中断。

以太网 DMA 丢失帧和缓冲区上溢计数器寄存器 (ETH_DMAMFBOCR)

Ethernet DMA missed frame and buffer overflow counter register

偏移地址：0x1020

复位值：0x0000 0000

DMA 可维护两个计数器在接收过程中对丢失帧数目进行计数。该寄存器会报告计数器的当前值。计数器用于进行诊断。位 [15:0] 指示因 STM32F4xx 缓冲区不可用（无可用的接收描述符）而丢失的帧。位 [27:17] 指示因 Rx FIFO 存在上溢情况以及帧长度过短（不足 64 字节的好帧）而丢失的帧。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			OFOC	MFA											OMFC	MFC															
			rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r

位 31:29 保留，必须保持复位值。

位 28 **OFOC**：FIFO 上溢计数器的上溢位 (Overflow bit for FIFO overflow counter)

位 27:17 **MFA**：应用程序所丢失的帧 (Missed frames by the application)
指示应用程序所丢失的帧数

位 16 **OMFC**：丢失帧计数器的上溢位 (Overflow bit for missed frame counter)

位 15:0 **MFC**：控制器所丢失的帧 (Missed frames by the controller)
指示控制器因主机接收缓冲区不可用而丢失的帧数。DMA 每丢弃一个传入帧，此计数器值加 1。

以太网 DMA 接收状态看门狗定时器寄存器 (ETH_DMARSWTR)

Ethernet DMA receive status watchdog timer register

偏移地址：0x1024

复位值：0x0000 0000

向该寄存器写入一个非零值时，可使能针对接收状态 (RS, ETH_DMASR[6]) 的看门狗定时器。

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

位 31:8 保留，必须保持复位值。

位 7:0 **RSWTC**：接收状态 (RS) 看门狗定时器计数 (Receive status (RS) watchdog timer count)
指示 HCLK 时钟周期数乘以 256 之后的时间值，即看门狗定时器的设定时间值。看门狗定时器会在 RxDMA 结束帧传输之后由编程设定的值触发，但相应的 RS 状态位因相应描述符中的 RDES1[31] 已置 1 而未置 1。当看门狗定时器计时结束时，RS 位置 1 且定时器停止。由于 RS 根据每个接收帧的 RDES1[31] 进行自动设置而使 RS 位设为高电平时，看门狗定时器复位。

以太网 DMA 当前主机发送描述符寄存器 (ETH_DMACHTDR)

Ethernet DMA current host transmit descriptor register

偏移地址：0x1048

复位值：0x0000 0000

当前主机发送描述符寄存器会指向 DMA 所读取的当前发送描述符的起始地址。

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

位 31:0 **HTDAP**：主机发送描述符地址指针 (Host transmit descriptor address pointer)
复位时清零。该指针在运行期间由 DMA 更新。

以太网 DMA 当前主机接收描述符寄存器 (ETH_DMACHRDR)

Ethernet DMA current host receive descriptor register

偏移地址：0x104C

复位值：0x0000 0000

当前主机接收描述符寄存器会指向 DMA 所读取的当前接收描述符的起始地址。

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

位 31:0 **HRDAP**：主机接收描述符地址指针 (Host receive descriptor address pointer)
复位时清零。该指针在运行期间由 DMA 更新。

以太网 DMA 当前主机发送缓冲区地址寄存器 (ETH_DMACHTBAR)

Ethernet DMA current host transmit buffer address register

偏移地址：0x1050

复位值：0x0000 0000

当前主机发送缓冲区地址寄存器会指向 DMA 所读取的当前发送缓冲区地址。

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

位 31:0 **HTBAP**：主机发送缓冲区地址指针 (Host transmit buffer address pointer)
复位时清零。该指针在运行期间由 DMA 更新。

以太网 DMA 当前主机接收缓冲区地址寄存器 (ETH_DMACHRBAR)

Ethernet DMA current host receive buffer address register

偏移地址：0x1054

复位值：0x0000 0000

当前主机接收缓冲区地址寄存器会指向 DMA 所读取的当前接收缓冲区地址。

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

位 31:0 **HRBAP**：主机接收缓冲区地址指针 (Host receive buffer address pointer)
复位时清零。该指针在运行期间由 DMA 更新。

29.8.5 以太网寄存器映射

表 170 给出了 ETH 寄存器映射和复位值。

表 170. 以太网寄存器映射和复位值

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	ETH_MACCR	Reserved						CSTF	eserved	WD	JD	Reserved		IFG			CSD	Reserved	FES	ROD	LM	DM	IPCO	RD	Reserved	APCS	BL		DC	TE	RE	Reserved	
	Reset value							0		0	0			0	0	0	0		0	0	0	0	0	0		0	0	0	0	0	0		
0x04	ETH_MACFFR	RA	Reserved																				HPF	SAF		PCF		BFD	PAM	DAIF	HM	HU	PM
	Reset value	0																					0	0	0	0	0	0	0	0	0	0	0
0x08	ETH_MACHTHR	HTH[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	ETH_MACHTLR	HTL[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	ETH_MACMIIAR	Reserved																PA					MR				CR				MW	MB
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14	ETH_MACMIIDR	Reserved																MD															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	ETH_MACFCR	PT																Reserved								ZQPD	Reserved	PLT		UPFD	RFCE	TFCE	FCB/BPA
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0									0		0	0	0	0	0	0
0x1C	ETH_MACVLANTR	Reserved															VLANTC	VLANTI															
	Reset value																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x28	ETH_MACRWUFFR	Frame filter reg0\Frame filter reg1\Frame filter reg2\Frame filter reg3\Frame filter reg4\...\Frame filter reg7																															
	Reset value	0																															
0x2C	ETH_MACPMTCSR	WFFRPR	Reserved																					GU	Reserved		WFR	MPR	Reserved			WFE	MPE	PD
	Reset value	0																						0			0	0				0	0	0

表 170. 以太网寄存器映射和复位值（续）

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x34	ETH_MACDBGR	Reserved						TFF	TFNEGU	Reserved	TFWA	TFRS		MTP	MTFCS		MMTEA	Reserved						RFFL		Reserved	RFRCS		RFWRA	Reserved	MSFRWCS	MMRPEA
	Reset value							0	0		0	0	0	0	0	0	0							0	0		0	0	0		0	0	0
0x38	ETH_MACSR																	Reserved						TSTS		Reserved	MMCTS	MMCRS	MMCS	PMTS	Reserved		
	Reset value																							0			0	0	0	0			
0x3C	ETH_MACIMR																	Reserved						TSTIM		Reserved				PMTIM	Reserved		
	Reset value																							0						0			
0x40	ETH_MACA0HR	MO	Reserved															MACA0H															
	Reset value	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x44	ETH_MACA0LR	MACA0L																															
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x48	ETH_MACA1HR	AE	SA	MBC[6:0]						Reserved								MACA1H															
	Reset value	0	0	0	0	0	0	0	0									1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x4C	ETH_MACA1LR	MACA1L																															
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x50	ETH_MACA2HR	AE	SA	MBC						Reserved								MACA2H															
	Reset value	0	0	0	0	0	0	0	0									1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x54	ETH_MACA2LR	MACA2L																															
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x58	ETH_MACA3HR	AE	SA	MBC						Reserved								MACA3H															
	Reset value	0	0	0	0	0	0	0	0									1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x5C	ETH_MACA3LR	MACA3L																															
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x100	ETH_MMCCR	Reserved																									MCFHP	MCP	MCF	ROR	CSR	CR	
	Reset value																										0	0	0	0	0	0	
0x104	ETH_MMCRIR	Reserved														RGUFS		Reserved									RFAES	RFCES	Reserved				
	Reset value															0											0	0					
0x108	ETH_MMCTIR	Reserved										TGFS		Reserved				TGFMSCS	TGFSCS	Reserved													
	Reset value											0						0	0														
0x10C	ETH_MMCRIMR	Reserved														RGUFM		Reserved									RFAEM	RFCEM	Reserved				
	Reset value															0											0	0					
0x110	ETH_MMCTIMR	Reserved										TGFM		Reserved				TGFMSCM	TGFSCM	Reserved													
	Reset value											0						0	0														
0x14C	ETH_MMCTGFSCCR	TGFSCC																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x150	ETH_MMCTGFMSCCR	TGFMSCC																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

表 170. 以太网寄存器映射和复位值（续）

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x168	ETH_MMCTGFCR	TGFC																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x194	ETH_MMCRFCECR	RFCEC																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x198	ETH_MMCRFAECR	RFAEC																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1C4	ETH_MMCRGUFCR	RGUFC																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x700	ETH_PTPTSCR	Reserved													TSPFFMAE	TSCNT	TSSMRME	TSSEME	TSSIPV4FE	TSSIPV6FE	TSSPTPOEFE	TSPTPPSV2E	TSSSR	TSSARFE	Reserved	TTSARU	TSITE	TSSTU	TSSTI	TSFCU	TSE
	Reset value														0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
0x704	ETH_PTPSSIR	Reserved																								STSSI							
	Reset value																									0	0	0	0	0	0	0	0
0x708	ETH_PTPTSHR	STS[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x70C	ETH_PTPTSLR	STPNS	STSS																														
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x710	ETH_PTPTSHUR	TSUS																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x714	ETH_PTPTSLUR	TSUPNS	TSUSS																														
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x718	ETH_PTPTSAR	TSA																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x71C	ETH_PTPTTHR	TTSH																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x720	ETH_PTPTTLR	TTSL																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x728	ETH_PTPTSSR	Reserved																														TSTTR	TSSO
	Reset value																															0	0
0x1000	ETH_DMABMR	Reserved					MB	AAB	FPM	USP	RDP						FB	PM	PBL						EDFE	DSL					DA	SR
	Reset value						0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1
0x1004	ETH_DMATPDR	TPD																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1008	ETH_DMARPDR	RPD																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x100C	ETH_DMARDLAR	SRL																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1010	ETH_DMATDLAR	STL																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1014	ETH_DMASR	Reserved	TSTS	PMTS	MMCS	Reserved	EBS			TPS			RPS			NIS	AIS	ERS	FBES	Reserved	ETS	RWTS	RPSS	RBUS	RS	TUS	ROS	TJTS	TBUS	TPSS	TS		
	Reset value		0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0

表 170. 以太网寄存器映射和复位值（续）

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x1018	ETH_DMAOMR	Reserved					DTCEFD	RSF	DFRF	Reserved		TSF	FTF	Reserved			TTC			ST	Reserved					FEF	FUGF	Reserved	RTC		OSF	SR	Reserved
	Reset value						0	0	0			0	0				0	0	0	0						0	0		0	0	0	0	
0x101C	ETH_DMAIER	Reserved															NISE	AISE	ERIE	FBEIE	Reserved		ETIE	RWTIE	RPSIE	RBUIE	RIE	TUIE	ROIE	TJTIE	TBUIE	TPSIE	TIE
	Reset value																0	0	0	0			0	0	0	0	0	0	0	0	0	0	0
0x1020	ETH_DMAMFBOCR	Reserved			OFOC	MFA											OMFC	MFC															
	Reset value				0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1024	ETH_DMARSWTR	Reserved																								RSWTC							
	Reset value																									0	0	0	0	0	0	0	0
0x1048	ETH_DMACHTDR	HTDAP																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x104C	ETH_DMACHRDR	HRDAP																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1050	ETH_DMACHTBAR	HTBAP																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1054	ETH_DMACHRBAR	HRBAP																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

有关寄存器边界地址的信息，请参见*第 52 页的表 2*。

30 全速 USB on-the-go (OTG_FS)

除非特别说明，否则本节适用于整个 STM32F4xx 系列器件。

30.1 OTG_FS 简介

本节介绍了 OTG_FS 控制器的架构和编程模型。

使用了以下首字母缩略词：

FS	全速
LS	低速
MAC	介质访问控制器
OTG	On-the-go
PFC	数据包 FIFO 控制器
PHY	物理层
USB	通用串行总线
UTMI	USB 2.0 收发器宏单元接口 (UTMI)

参考文档如下：

- USB On-The-Go 补充标准，第 1.3 版
- 通用串行总线规范第 2.0 版

OTG_FS 是一款双角色设备 (DRD) 控制器，同时支持从机功能和主机功能，完全符合 *USB 2.0 规范的 On-The-Go 补充标准*。此外，该控制器也可配置为“仅主机”模式或“仅从机”模式，完全符合 *USB 2.0 规范*。在主机模式下，OTG_FS 支持全速（FS，12 Mb/s）和低速（LS，1.5 Mb/s）收发器，而从机模式下则仅支持全速（FS，12 Mb/s）收发器。OTG_FS 同时支持 HNP 和 SRP。主机模式下需要的唯一外部设备是提供 V_{BUS} 的电荷泵。

30.2 OTG_FS 主要特性

主要特性可分为三类：通用特性、主机模式特性和从机模式特性。

30.2.1 通用特性

OTG_FS 接口的通用特性如下：

- 经 USB-IF 认证，符合通用串行总线规范第 2.0 版
- 模块内嵌的 PHY 还完全支持定义在标准规范 OTG 补充第 1.3 版中的 OTG 协议
  - 支持 A-B 器件识别（ID 线）
  - 支持主机协商协议 (HNP) 和会话请求协议 (SRP)
  - 允许主机关闭 V_{BUS} 以在 OTG 应用中节省电池电量
  - 支持通过内部比较器对 V_{BUS} 电平采取监控
  - 支持主机到从机的角色动态切换
- 可通过软件配置为以下角色：
  - 具有 SRP 功能的 USB FS 从机（B 器件）
  - 具有 SRP 功能的 USB FS/LS 主机（A 器件）
  - USB On-The-Go 全速双角色设备
- 支持 FS SOF 和 LS Keep-alive 令牌
  - SOF 脉冲可通过 PAD 输出
  - SOF 脉冲从内部连接到定时器 2 (TIM2)
  - 可配置的帧周期
  - 可配置的帧结束中断
- 具有省电功能，例如在 USB 挂起期间停止系统、关闭数字模块时钟、对 PHY 和 DFIFO 电源加以管理
- 具有采用高级 FIFO 控制的 1.25 KB 专用 RAM
  - 可将 RAM 空间划分为不同 FIFO，以便灵活有效地使用 RAM
  - 每个 FIFO 可存储多个数据包
  - 动态分配存储区
  - FIFO 大小可配置为非 2 的幂次方值，以便连续使用存储单元
- 一帧之内可以无需要应用程序干预，以达到最大 USB 带宽

30.2.2 主机模式特性

OTG_FS 接口在主机模式下具有以下主要特性和要求：

- 通过外部电荷泵生成 V_{BUS} 电压。
- 多达 8 个主机通道（管道）：每个通道都可以动态实现重新配置，可支持任何类型的 USB 传输。
- 内置硬件调度器可：
  - 在周期性硬件队列中存储多达 8 个中断加同步传输请求
  - 在非周期性硬件队列中存储多达 8 个控制加批量传输请求
- 管理一个共享 RX FIFO、一个周期性 TX FIFO 和一个非周期性 TX FIFO，以有效使用 USB 数据 RAM。

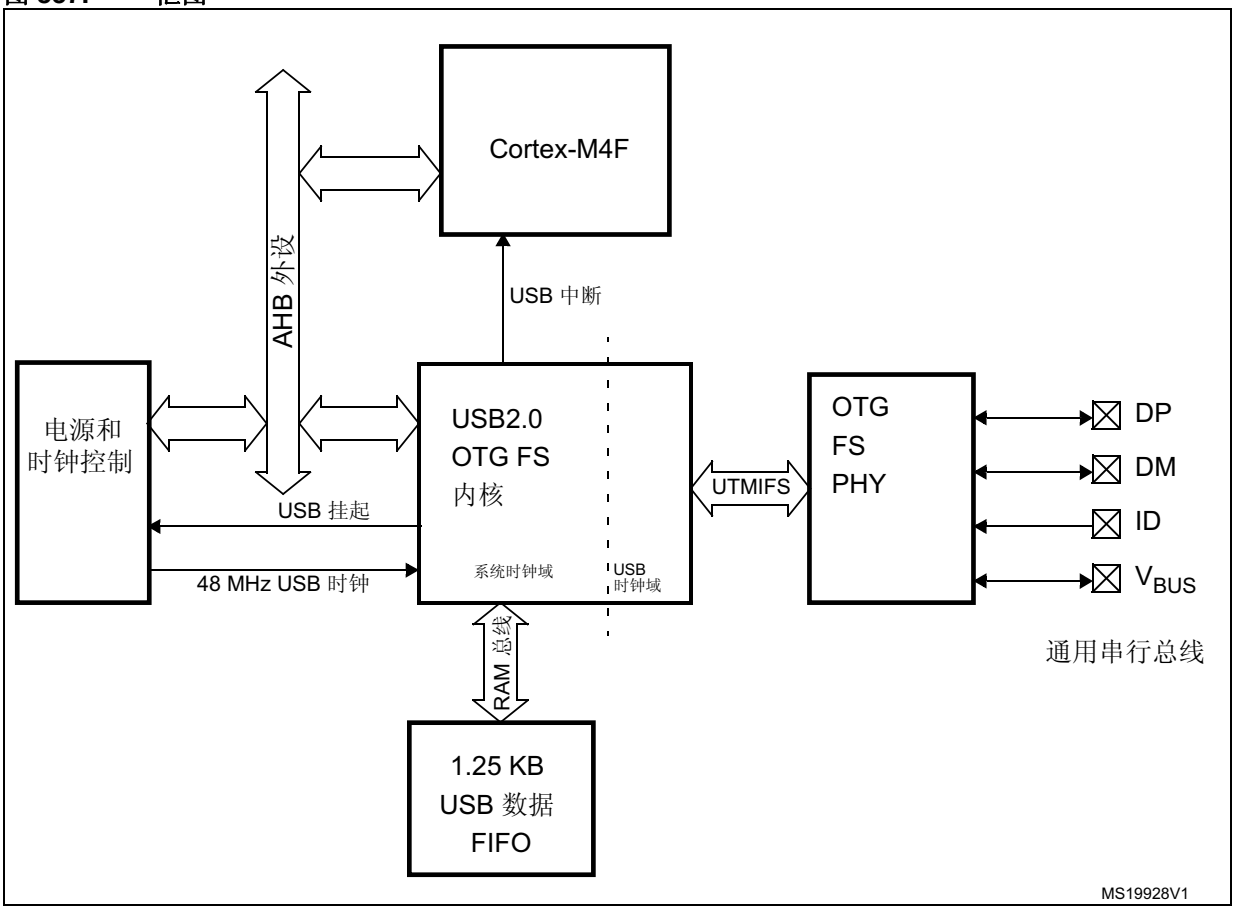
30.2.3 从机模式特性

OTG_FS 接口在从机模式下具有以下特性：

- 1 个双向控制端点 0
- 3 个 IN 端点 (EP)，可配置为支持批量传输、中断传输或同步传输
- 3 个 OUT 端点，可配置为支持批量传输、中断传输或同步传输
- 管理一个共享 Rx FIFO 和一个 Tx-OUT FIFO，以高效使用 USB 数据 RAM
- 管理多达 4 个专用 Tx-IN FIFO（分别用于每个使能的 IN EP），降低应用程序负荷
- 支持软断开功能。

30.3 OTG_FS 功能说明

图 357. 框图



30.3.1 OTG 全速模块

USB OTG FS 通过外部石英时钟从复位和时钟控制器 (RCC) 接收 48 MHz ±0.25% 的时钟。USB 时钟用于全速 (12 Mb/s) 驱动 48 MHz 域，必须在配置 OTG FS 模块前使能。

CPU 通过 AHB 外设总线对 OTG FS 模块寄存器进行读写操作，通过*第 30.15 节：OTG_FS 中断*中所述的 USB OTG 中断线接收 USB 事件通知。

CPU 通过向特定的 OTG_FS 单元（压栈寄存器）写入 32 位字来向 USB 提交数据。数据随即自动存储到 USB 数据 RAM 中配置的数据发送 FIFO 中。每个 IN 端点（从机模式）或 OUT 通道（主机模式）都有一个 Tx-FIFO 压栈寄存器。

CPU 从特定的 OTG_FS 地址（出栈寄存器）读取 32 位字，以接收来自 USB 的数据。数据随即从在 1.25 KB USB 数据 RAM 内配置的共享 Rx-FIFO 中弹出。每个 OUT 端点或 IN 通道都有一个 Rx-FIFO 出栈寄存器。

USB 协议层通过串行接口引擎 (SIE) 驱动，并通过片上物理层 (PHY) 中的全速/低速收发器模块经由 USB 进行数据的串行通信。

30.3.2 全速 OTG PHY

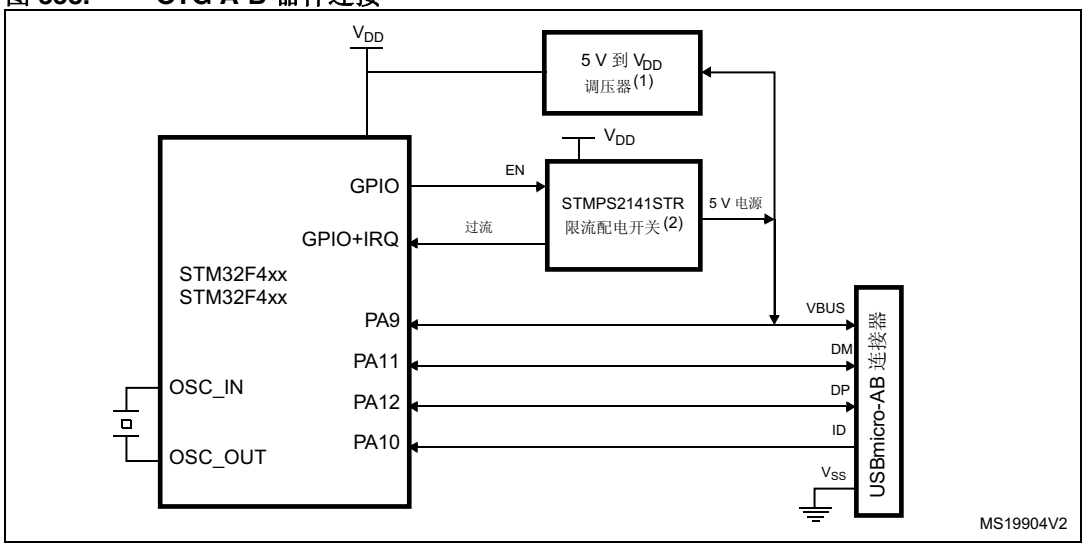
嵌入式全速 OTG PHY 由 OTG FS 模块控制，通过 UTMI+ 总线 (UTMIFS) 的全速子集传送 USB 控制和数据信号。为 USB 连接提供物理支持。
全速 OTG PHY 包括以下组成部分：

- 供主机和设备使用的 FS/LS 收发器模块。直接在单端 USB 线上驱动发送和接收操作。
- 集成 ID 上拉电阻，用于对 ID 线进行采样，以便识别 A/B 器件。
- 由 OTG_FS 模块控制的 DP/DM 集成上拉电阻和下拉电阻，具体使能哪种电阻取决于设备的当前角色。作为从机使用时，只要检测到 V_{BUS} 为有效电平（B 会话有效），立即使能 DP 上拉电阻。主机模式下则使能 DP/DM 上的下拉电阻。通过主机协商协议 (HNP) 更改设备角色时，将在上拉电阻和下拉电阻之间动态切换。
- 上拉/下拉电阻 ECN 电路。根据适用于 USB 2.0 版本的电阻 ECN 规定，DP 上拉电路包括 2 个由 OTG_FS 单独进行控制的电阻。对 DP 上拉阻值的动态调整可以提高噪声抑制能力和 Tx/Rx 信号质量。
- 带滞回功能的 V_{BUS} 感应比较器，用于检测 V_{BUS} 有效、A-B 会话有效和会话端电压阈值。执行 USB 操作期间，这些比较器用于驱动会话请求协议 (SRP)、检测会话的有效启动和结束条件，以及持续监视 V_{BUS} 供电情况。
- V_{BUS} 脉冲电路，用于在 SRP 期间通过电阻对 V_{BUS} 充电/放电（驱动力较弱）。

小心： 为确保 USB OTG FS 模块正常工作，AHB 频率应大于 14.2 MHz。

30.4 OTG 双角色设备 (DRD)

图 358. OTG A-B 器件连接



1. 只有在构建由 V_{BUS} 供电的器件时才需要外部调压器。
2. 只有在应用必须支持由 V_{BUS} 供电的器件时才需要 STMPS2141STR。如果应用电路板提供 5 V 电源，则可以使用基本电源开关。
3. V_{DD} 范围介于 2 V 到 3.6 V 之间。

30.4.1 ID 线检测

采取主机还是从机（默认设置）角色取决于 ID 输入引脚的电平。插入 USB 时即可根据哪一端 USB 电缆连接到 micro-AB 插座来确定 ID 线状态。

- 如果 USB 电缆的 B 端连入，其 ID 线悬空，则由于设备在 ID 线上的集成上拉电阻设备将检测到 ID 高电平并确认采取默认的从机角色。在此配置中，OTG_FS 符合“USB2.0 On-The-Go 规范第 1.3 版补充标准中第 6.8.2 章节 On-The-Go B 器件”中所述的 FSM 标准。
- 如果 USB 电缆的 A 端连入，其 ID 线接地，则 OTG_FS 将发出 ID 线状态更改中断（OTG_FS_GINTSTS 中的 CIDSCHG 位）以初始化主机软件，并自动切换为主机角色。在此配置中，OTG_FS 符合 USB2.0 On-The-Go 规范第 1.3 版补充标准中第 6.8.1 章节 On-The-Go A 器件。

30.4.2 HNP 双角色设备

全局 USB 配置寄存器中的 HNP 使能位（OTG_FS_GUSBCFG 中的 HNPCAP 位）可使 OTG_FS 模块根据主机协商协议 (HNP) 动态切换角色，例如从 A 主机切换为 A 从机（反之亦然），或者从 B 从机切换为 B 主机（反之亦然）。通过全局 OTG 控制和状态寄存器中的连接器 ID 状态位（OTG_FS_GOTGCTL 中的 CIDSTS 位）及全局中断和状态寄存器中的当前工作模式位（OTG_FS_GINTSTS 中的 CMOD 位）二者的组合值可读取设备当前状态。

*第 30.17 节：OTG_FS 编程模型*详细介绍了 HNP 编程模型。

30.4.3 SRP 双角色设备

全局 USB 配置寄存器中的 SRP 使能位（OTG_FS_GUSBCFG 中的 SRPCAP 位）可使 OTG_FS 模块关闭 V_{BUS} 供电，为 A 器件节省电能。注意，无论 OTG_FS 采取主机角色还是从机角色，A 器件将始终负责 V_{BUS} 的提供。

*第 30.17 节：OTG_FS 编程模型*详细介绍了 SRP A/B 器件编程模型。

30.5 USB 设备

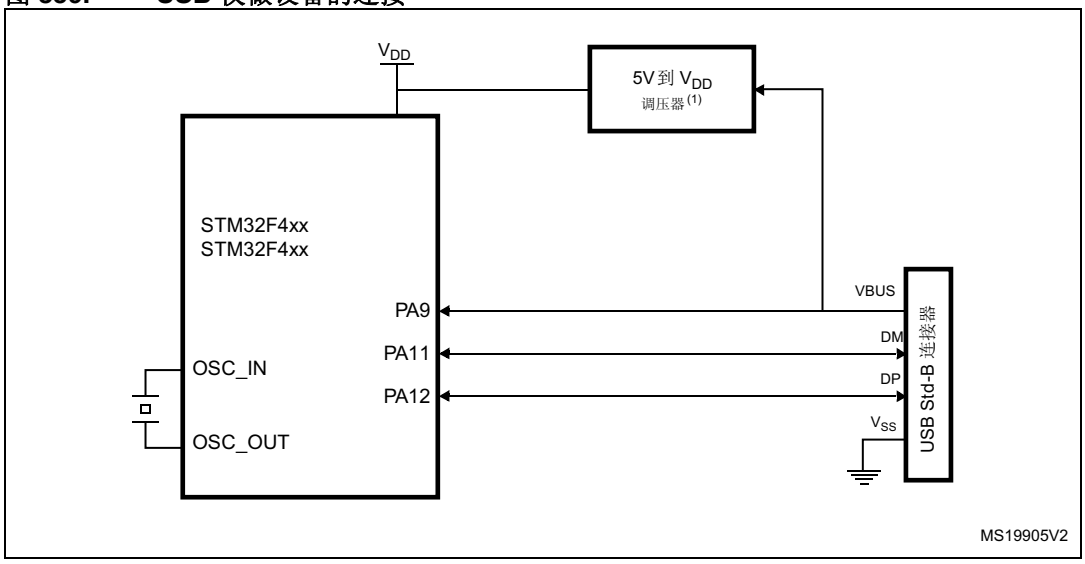
本节介绍 OTG_FS 在 USB 设备模式下所具有的功能。在以下情形下，OTG_FS 用作 USB 设备：

- OTG B 器件
  - OTG B 器件插入 USB 电缆 B 端时的默认状态
- OTG A 器件
  - OTG A 器件被 HNP 切换为设备角色后的状态
- B 器件
  - 如果 ID 线有效，器件与 USB 电缆的 B 端相连，全局 USB 配置寄存器中的 HNP 功能位（OTG_FS_GUSBCFG 中的 HNPCAP 位）清零（请参见 On-The-Go 第 1.3 版的第 6.8.3 节）。
- 仅作设备（请参见*图 359：USB 仅做设备的连接*）
  - 全局 USB 配置寄存器中的强制设备模式位（OTG_FS_GUSBCFG 中的 FDMOD）置 1，强制 OTG_FS 模块仅用作 USB 设备（请参见 On-The-Go 第 1.3 版的第 6.8.3 节）。这种情况下，即使 USB 连接器上存在 ID 线，也会将该 ID 线忽略。

注意： *要在 B 器件或仅作设备配置情形下构建总线供电的设备方案，需要添加一个外部调压器，用于从 V_{BUS} 生成 V_{DD} 芯片电源。*

关闭 V_{BUS} 感应选项可以释放 V_{BUS} 引脚。可通过在 OTG_FS_GCCFG 寄存器中将 NOVBUSSENS 位置 1 来完成此操作。这种情况下，内部将 V_{BUS} 视为始终处于 V_{BUS} 有效电平 (5 V)。

图 359. USB 仅做设备的连接



1. 使用调压器构建总线供电设备。
2. V_{DD} 范围介于 2 V 到 3.6 V 之间。

30.5.1 支持 SRP 功能的设备

全局 USB 配置寄存器中的 SRP 功能位（OTG_FS_GUSBCFG 中的 SRPCAP 位）可使 OTG_FS 支持会话请求协议 (SRP)。这样一来，远程 A 器件便可以在 USB 会话挂起时，通过关闭 V_{BUS} 来节省电能。

*B 器件会话请求协议*一节详细介绍了 SRP 设备的编程模型。

30.5.2 设备状态

供电状态

V_{BUS} 输入检测到 B 会话有效电压，就会使 USB 设备进入供电状态（请参见 USB2.0 第 9.1 节）。然后，OTG_FS 自动连接 DP 上拉电阻，发出全速设备与主机相连的信号并生成会话请求中断（OTG_FS_GINTSTS 中的 SRQINT 位），指示进入供电状态。

此外，V_{BUS} 输入还可确保主机在 USB 操作期间提供有效的 V_{BUS} 电平。如果检测到 V_{BUS} 降至 B 会话有效电压以下（例如，因电源干扰或主机端口关闭引发），OTG_FS 将自动断开连接并生成检测到会话结束中断（OTG_FS_GOTGINT 中的 SEDET 位），指示 OTG_FS 已退出供电状态。

供电状态下，OTG_FS 期望收到来自主机的复位信号。其它 USB 操作则无法执行。收到复位信号后，立即生成检测到复位中断（OTG_FS_GINTSTS 中的 USBRST）。复位信号结束后，将生成枚举完成中断（OTG_FS_GINTSTS 中的 ENUMDNE 位），OTG_FS 随即进入默认状态。

软断开

供电状态可借助软断开功能通过软件退出。将设备控制寄存器中的软断开位（OTG_FS_DCTL 中的 SDIS 位）置 1 即可移除 DP 上拉电阻，此时尽管没有从主机端口实际拔出 USB 电缆，但主机端仍会发生设备断开检测中断。

默认状态

默认状态下，OTG_FS 期望从主机收到 SET_ADDRESS 命令。其它 USB 操作则无法执行。当 USB 上解码出有效 SET_ADDRESS 命令时，应用程序会将相应的地址值写入设备配置寄存器中的设备地址字段（OTG_FS_DCFG 中的 DAD 位）。OTG_FS 随即进入地址状态，并准备好以所配置的 USB 地址对主机事务进行应答。

挂起状态

OTG_FS 设备持续监视 USB 活动。在 USB 空闲时间达到 3 ms 后，将发出早期挂起中断（OTG_FS_GINTSTS 中的 ESUSP 位），并在 3 ms 后由挂起中断（OTG_FS_GINTSTS 中的 USBSUSP 位）确认设备进入挂起状态。然后，设备状态寄存器中的设备挂起位（OTG_FS_DSTS 中的 SUSPSTS 位）自动置 1，OTG_FS 随即进入挂起状态。

可通过设备本身退出挂起状态。这种情况下，应用程序会将设备控制寄存器中的远程唤醒信号位（OTG_FS_DCTL 中的 RWUSIG 位）置 1，并在 1 ms 到 15 ms 后将其清零。

但若设备检测到主机发出的恢复信号，将生成恢复中断（OTG_FS_GINTSTS 中的 WKUPINT 位），设备挂起位自动清零。

30.5.3 设备端点

OTG_FS 模块实现了以下 USB 端点：

- 控制端点 0：
  - 双向且仅处理控制消息
  - 使用一组单独的寄存器来处理 IN 和 OUT 事务
  - 专用控制 (OTG_FS_DIEPCTL0/OTG_FS_DOEPCTL0) 寄存器、传输配置 (OTG_FS_DIEPTSIZ0/OTG_FS_DIEPTSIZ0) 寄存器和状态中断 (OTG_FS_DIEPINTx/OTG_FS_DOEPINT0) 寄存器。控制和传输大小寄存器中可用的位组与其它端点中稍有不同
- 3 个 IN 端点
  - 每个端点都可配置为支持同步传输、批量传输或中断传输类型
  - 每个端点都有专用控制 (OTG_FS_DIEPCTL*x*) 寄存器、传输配置 (OTG_FS_DIEPTSIZ*x*) 寄存器和状态中断 (OTG_FS_DIEPINT*x*) 寄存器
  - 设备 IN 端点通用中断屏蔽寄存器 (OTG_FS_DIEPMSK) 可用于使能/禁止所有 IN 端点（包括 EP0）上的同一类端点中断源
  - 支持未完成的同步 IN 传输中断（OTG_FS_GINTSTS 中的 IISOIXFR 位），该中断将在当前帧中至少有一个同步 IN 端点上的传输未完成时触发。该中断和周期性帧中断 (OTG_FS_GINTSTS/EOPF) 一起触发
- 3 个 OUT 端点
  - 每个端点都可配置为支持同步传输、批量传输或中断传输类型
  - 每个端点都有专用控制 (OTG_FS_DOEPCTL*x*) 寄存器、传输配置 (OTG_FS_DOEPTSIZ*x*) 寄存器和状态中断 (OTG_FS_DOEPINT*x*) 寄存器
  - 设备 OUT 端点通用中断屏蔽寄存器 (OTG_FS_DOEPMSK) 可用于使能/禁止所有 OUT 端点（包括 EP0）上的同一类端点中断源
  - 支持未完成的同步 OUT 传输中断（OTG_FS_GINTSTS 中的 INCOMPISOOUT 位），该中断将在当前帧中至少有一个同步 OUT 端点上的传输未完成时触发。该中断和周期性帧中断 (OTG_FS_GINTSTS/EOPF) 一起触发

端点控制

- 应用程序可通过设备端点 *x* IN/OUT 控制寄存器 (DIEPCTL*x*/DOEPCTL*x*) 对端点采取以下控制：
  - 端点使能/禁止
  - 在当前配置下激活端点
  - 设置 USB 传输类型（同步、批量和中断）
  - 设置支持的数据包大小
  - 设置与 IN 端点相关的 Tx-FIFO 编号
  - 设置希望收到的或发送时要使用到的 data0/data1 PID（仅限批量/中断传输）
  - 设置接收或发送事务时所对应的奇/偶帧（仅限同步传输）
  - 可以设置 NAK 位，从而不论此时 FIFO 的状态如何，都对主机的请求回复 NAK
  - 可以设置 STALL 位，使得主机对该端点的令牌都被硬件回复 STALL
  - 可以将 OUT 端点设置为侦听模式，即对接收到的数据不进行 CRC 检查

端点传输

设备端点 *x* 传输尺寸寄存器 (DIEPTSIZ*x*/DOEPTSIZ*x*) 允许应用程序对传输尺寸参数进行编程并读取传输状态。必须在端点控制寄存器中的端点使能位置 1 之前完成对此寄存器的设置。使能端点后，这些字段立即变为只读状态，同时 OTG FS 模块根据当前传输状态对这些字段进行更新。

可对以下传输参数进行编程：

- 以字节为单位的传输大小
- 构成整个传输的数据包个数

端点状态/中断

设备端点 *x* 中断寄存器 (DIEPINT*x*/DOPEPINT*x*) 指示端点在出现 USB 和 AHB 相关事件时的状态。当模块中断寄存器中的 OUT 端点中断位或 IN 端点中断位（分别为 OTG_FS_GINTSTS 中的 OEPINT 位或 OTG_FS_GINTSTS 中的 IEPINT 位）置 1 时，应用程序必须读取这些寄存器以获得详细信息。在应用程序读取这些寄存器之前，必须先读取设备全体端点中断 (OTG_FS_DAINT) 寄存器，以获取设备端点 *x* 中断寄存器的端点编号。应用程序必须将此寄存器中的相应位清零，才能将 DAINT 和 GINTSTS 寄存器中的相应位清零。

模块提供以下状态检查和中断产生功能：

- 传输完成中断，指示应用程序 (AHB) 和 USB 端均已完成数据传输
- Setup 阶段已完成（仅针对控制传输类型的 OUT 端点）
- 相关的发送 FIFO 为半空或全空状态（IN 端点）
- NAK 应答已发送到主机（仅针对同步传输的 IN 端点）
- Tx-FIFO 为空时接收到 IN 令牌（仅针对批量和中断传输类型的 IN 端点）
- 尚未使能端点时接收到 OUT 令牌
- 检测到 babble 错误
- 应用程序关闭端点生效
- 应用程序对端点设置 NAK 生效（仅针对同步传输类型的 IN 端点）
- 接收到 3 个以上连续 setup 数据包（仅针对控制类型的 OUT 端点）
- 检测到超时状况（仅针对控制传输类型的 IN 端点）
- 同步传输类型的数据包未产生中断而丢失

30.6 USB 主机

本节介绍了 OTG_FS 在 USB 主机模式下所具有的功能。在以下情形下，OTG_FS 用作 USB 主机：

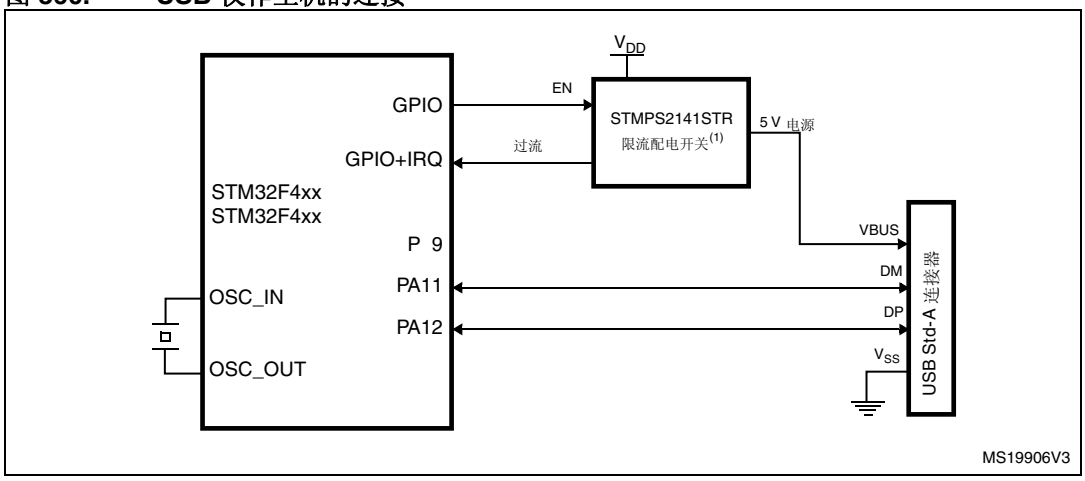
- OTG A 主机
  - OTG A 器件在插入 USB 电缆 A 端时的默认状态
- OTG B 主机
  - OTG B 器件被 HNP 切换为主机角色后的状态
- A 器件
  - 如果 ID 线有效，器件与 USB 电缆的 A 端相连，全局 USB 配置寄存器中的 HNP 功能位（OTG_FS_GUSBCFG 中的 HNPCAP 位）清零。DP/DM 线上的集成下拉电阻自动使能。
- 仅作主机（请参见*图 360：USB 仅作主机的连接*）。
  - 全局 USB 配置寄存器中的强制模式位（OTG_FS_GUSBCFG 中的 FHMOD 位）将强制 OTG_FS 模块作为 USB 仅作主机运行。这种情况下，即使 USB 连接器上存在 ID 线，也会将该 ID 线忽略。DP/DM 线上的集成下拉电阻自动使能。

注意： *微控制器不能输出 5V 以提供 V_{BUS}。为此，必须在微控制器以外添加电荷泵或电源开关（如果应用电路板提供 5 V 电源）来驱动 5 V V_{BUS} 线。外部电荷泵可通过任何 GPIO 输出驱动。OTG A 主机、A 器件和仅作主机配置都需要使用电荷泵。*

V_{BUS} 输入可确保电荷泵在 USB 操作期间提供有效的 V_{BUS} 电平，同时电荷泵过流输出可连接到任意配置成外部中断的 GPIO 引脚。在过流 ISR 中必须立即关闭 V_{BUS}。

关闭 V_{BUS} 感应选项可以释放 V_{BUS} 引脚。可通过在 OTG_FS_GCCFG 寄存器中将 NOVBUSSENS 位置 1 来完成此操作。这种情况下，内部将 V_{BUS} 视为始终处于 V_{BUS} 有效电平 (5 V)。

图 360. USB 仅作主机的连接



1. 只有在应用必须支持由 V_{BUS} 供电的器件时才需要 STMPS2141STR。如果应用电路板提供 5 V 电源，则可以使用基本电源开关。
2. V_{DD} 范围介于 2 V 到 3.6 V 之间。

30.6.1 支持 SRP 功能的主机

全局 USB 配置寄存器中的 SRP（OTG_FS_GUSBCFG 中的 SRPCAP 位）可提供 SRP 支持。使能 SRP 功能后，主机可在 USB 会话挂起时通过关闭 V_{BUS} 电源来节省电能。

*A 器件会话请求协议*一节详细介绍了 SRP 主机的编程模型。

30.6.2 USB 主机状态

给主机端口供电

微控制器不能输出 5V 以提供 V_{BUS}。为此，必须在微控制器以外添加电荷泵或基本电源开关（如果应用电路板提供 5 V 电源）来驱动 5 V V_{BUS} 线。外部电荷泵可通过任何 GPIO 输出驱动。当应用程序确定使用 GPIO 来控制外部器件输出 V_{BUS}，还必须将主机端口控制和状态寄存器中的端口电源位（OTG_FS_HPRT 中的 PPWR 位）置 1。

V_{BUS} 有效

使能 HNP 或 SRP 后，应将 VBUS 感应引脚 (PA9) 连接到 V_{BUS}。V_{BUS} 输入可确保电荷泵在 USB 操作期间提供有效的 V_{BUS} 电平。如果 V_{BUS} 电压意外降至 V_{BUS} 有效阈值 (4.25 V) 以下，将通过会话结束检测位（OTG_FS_GOTGINT 中的 SEDET 位）触发 OTG 中断。之后应用必须断开 V_{BUS} 电源并使端口电源位清零。

同时禁止 HNP 和 SRP 时，应断开 VBUS 感应引脚 (PA9) 与 V_{BUS} 之间的连接。该引脚可用作 GPIO。

电荷泵过流标志也可用来防止电气损坏。将电荷泵的过流标志输出连接到任意 GPIO 输入，然后将其配置为出现有效电平时生成端口中断。过流 ISR 必须立即关闭 V_{BUS} 并清零端口电源位。

主机检测设备连接

如果使能 SRP 或 HNP，即使可以随时连接 USB 外设或 B 器件，但是 OTG_FS 也只有在 V_{BUS} 有效后 (5V) 才能检测到设备的连接。当 V_{BUS} 处于有效电平且已连接远程 B 器件时，OTG_FS 模块将发出主机端口中断信号，该中断由主机端口控制和状态寄存器中的设备连接位（OTG_FS_HPRT 中的 PCDET 位）触发。

在 HNP 和 SRP 同时关闭的情况下，USB 设备或 B 器件将在连接后立即被检测到。OTG_FS 模块将发出主机端口中断信号，该中断由主机端口控制和状态寄存器中的设备连接位（OTG_FS_HPRT 中的 PCDET 位）触发。

主机检测设备断开

设备断开事件将触发断开连接检测中断（OTG_FS_GINTSTS 中的 DISCINT 位）。

主机枚举

检测到设备连接后，若又有新的设备连接进来，主机必须通过向新的设备发送 USB 复位和配置命令来启动枚举过程。

开始驱动 USB 复位前，应用程序必须等待去抖动完成位（OTG_FS_GOTGINT 中的 DBCDNE 位）触发 OTG 中断，这表示由于在 DP (FS) 或 DM (LS) 上连接上拉电阻而发生电气抖动之后，总线恢复稳定状态。

应用程序通过将主机端口控制和状态寄存器中的端口复位位（OTG_FS_HPRT 中的 PRST 位）置 1，使该过程最少持续 10 ms、最多持续 20 ms，以此通过 USB 驱动 USB 复位信号（单端零）。应用程序计算这个过程的持续时间，然后将端口复位位清零。

USB 复位序列完成后，端口使能/禁止更改位（OTG_FS_HPRT 中的 PENCHNG 位）立即触发主机端口中断，进而向应用程序发出通知，指示可从主机端口控制和状态寄存器中的端口速度字段（OTG_FS_HPRT 中的 PSPD）读取枚举的设备速度，以及主机已经开始驱动 SOF (FS) 或 Keep-alive 令牌 (LS)。此时主机已就绪，可通过对设备发送命令来完成对设备的枚举。

主机挂起

应用程序通过将主机端口控制和状态寄存器中的端口挂起位（OTG_FS_HPRT 中的 PSUSP）置 1 来挂起 USB 活动。OTG_FS 模块停止发送 SOF 并进入挂起状态。

可由远程设备的自主活动（远程唤醒）使总线退出挂起状态。这种情况下，远程唤醒信号将触发远程唤醒中断（OTG_FS_GINTSTS 中的 WKUPINT 位），硬件把主机端口控制和状态寄存器中的端口恢复位（OTG_FS_HPRT 中的 PRES 位）自行复位，并通过 USB 自动驱动恢复信号。应用程序必须为恢复窗口定时，然后将端口恢复位清零以退出挂起状态并重新启动 SOF。

如果由主机发起退出挂起状态，则应用程序必须将端口恢复位置 1 以启动主机端口上的恢复信号，为恢复窗口定时并最终将端口恢复位清零。

30.6.3 主机通道

OTG_FS 模块实现了 8 个主机通道。每个主机通道均可用于 USB 主机传输（USB 管道）。主机最多能同时处理 8 个传输请求。如果应用程序有 8 个以上的传输请求挂起，则在通道从之前任务释放后（即，接收到传输完成和通道停止中断后），主机控制器驱动器 (HCD) 必须为未处理的传输请求重新对通道进行分配。

每个主机通道都可配置为支持输入/输出以及周期性/非周期性事务。每个主机通道都使用专用控制 (HCCHAR*x*) 寄存器、传输配置 (HCTSIZ*x*) 寄存器/中断 (HCINT*x*) 寄存器以及和其相关的中断屏蔽寄存器 (HCINTMSK*x*)。

主机通道控制

- 应用程序可通过主机通道 *x* 特性寄存器 (HCCHAR*x*) 对主机通道作以下控制：
  - 通道使能/禁止
  - 设置目标 USB 设备的速度：FS/LS
  - 设置目标 USB 设备的地址
  - 设置与该通道通信的目标 USB 设备上的端点的编号
  - 设置该通道上的传输方向：IN/OUT
  - 设置该通道上的 USB 传输的类型：控制/批量/中断/同步
  - 设置与该通道通信的设备端点的最大包长
  - 设置要进行周期传输的帧：奇帧/偶帧

主机通道传输

主机通道传输大小寄存器 (HCTSIZ*x*) 允许应用程序对传输大小参数进行编程并读取传输状态。必须在主机通道特性寄存器中的通道使能位置 1 之前完成对此寄存器的设置。使能端点后，数据包计数字段立即变为只读状态，同时 OTG FS 模块根据当前传输状态对该字段进行更新。

- 可对以下传输参数进行编程：
  - 以字节为单位的传输大小
  - 构成整个传输大小的数据包个数
  - 初始数据 PID

主机通道状态/中断

主机通道 *x* 中断寄存器 (HCINT*x*) 指示端点在出现 USB 和 AHB 相关事件时的状态。当中断寄存器中的主机通道中断位（OTG_FS_GINTSTS 中的 HCINT 位）置 1 时，应用程序必须读取这些寄存器以获得详细信息。在读取这些寄存器之前，应用程序必须先读取主机全体通道中断 (HCAINT) 寄存器，以获取主机通道 x 中断寄存器的通道编号。应用程序必须将此寄存器中的相应位清零，才能将 HAINT 和 GINTSTS 寄存器中的相应位清零。OTG_FS_HCINTMSK-x 寄存器还提供每个通道各中断源的屏蔽位。

- 主机模块提供以下状态检查和中断产生功能：
  - 传输完成中断，指示应用程序 (AHB) 和 USB 端均已完成数据传输
  - 通道因传输完成、USB 事务错误或应用程序发出禁止命令而停止
  - 相关的发送 FIFO 为半空或全空状态（IN 端点）
  - 接收到 ACK 响应
  - 接收到 NAK 响应
  - 接收到 STALL 响应
  - 由于 CRC 校验失败、超时、位填充错误和错误的 EOP 导致 USB 事务错误
  - 串扰错误
  - 帧上溢
  - 用于数据同步的翻转位出错

30.6.4 主机调度器

主机模块内置硬件调度器，可自主对应用程序发出的 USB 事务请求重新排序和管理。每一帧开始时，主机都先执行周期性（同步和中断）事务，然后执行非周期性（控制和批量）事务，以符合 USB 规范对同步和中断传输高优先级的保证。

主机通过请求队列（一个周期性请求队列和一个非周期请求队列）处理 USB 事务。每个请求队列最多可存储 8 个条目。每个条目代表一个应用程序发起但还未得到响应的 USB 事务请求，并存储了执行该 USB 事务所用到的 IN 或 OUT 通道的编号，以及其它相关信息。USB 事务请求在队列中的写入顺序决定了事务在 USB 接口上的执行顺序。

每一帧开始时，主机都先处理周期性请求队列，然后处理非周期性请求队列。如果当前帧结束时，计划在当前帧执行的同步或中断类型的 USB 传输事务请求仍处于挂起状态，则主机将发出未完成周期性传输中断（OTG_FS_GINTSTS 中的 IPXFR 位）。OTG HS 模块负责对周期性和非周期性请求队列的管理。周期性发送 FIFO 和队列状态寄存器 (HPTXSTS) 与

非周期性发送 FIFO 和队列状态寄存器 (HNPTXSTS) 都为只读寄存器，应用程序可使用它们来读取各请求队列的状态。其中包括：

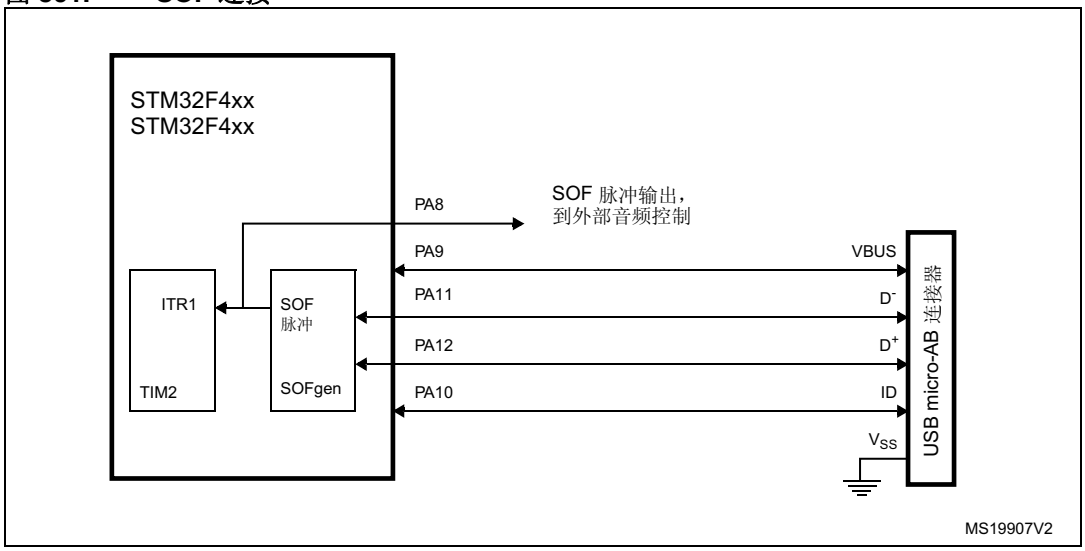
- 周期性（非周期性）请求队列中当前可用的空闲条目数（最多 8 个）
- 周期性（非周期性）Tx-FIFO（OUT 事务）中当前可用的空闲空间
- IN/OUT 令牌、主机通道编号和其它状态信息。

由于每个请求队列最多可存储 8 个 USB 事务请求，因此应用程序可以把主机 USB 事务请求提前发送给调度器；实际的通信最晚会在调度器处理完已挂起的 8 个周期事务和 8 个非周期事务完成之后出现在 USB 总线上。

要向主机调度器（队列）发出事务请求，应用程序必须读取 OTG_FS_HNPTXSTS 寄存器中的 PTXQSAV 位或 OTG_FS_HNPTXSTS 寄存器中的 NPTQXSAV 位，确保周期性（非周期性）请求队列中至少有一个可用空间来存储当前请求。

30.7 SOF 触发

图 361. SOF 连接



OTG FS 在主机和设备模式下都可以监视、跟踪和配置 SOF 帧并且还具备 SOF 脉冲输出功能。

此功能尤其适用于自适应音频时钟生成，其中音频设备需要与 PC 提供的同步音频数据流实现同步，或者主机需要根据音频设备的要求调整数据帧率。

30.7.1 主机 SOF

主机模式下，可以在主机帧间隔寄存器 (HFIR) 中对所产生的两个连续 SOF (FS) 或 keep-alive (LS) 令牌期间所出现的 PHY 时钟数进行编程，进而应用程序可对 SOF 帧周期进行控制。帧开始（OTH_FS_GINTSTS 中的 SOF 位）时都将生成中断。当前帧编号和出现下一个 SOF 前剩余的时间应用程序在主机帧编号寄存器 (HFNUM) 中能够进行跟踪。

使用全局控制和配置寄存器中的 SOFOUTEN 位，可以使任何 SOF 令牌发出的同时产生的、宽度为 12 个系统时钟周期的 SOF 脉冲信号从 SOF 引脚输出。此外，SOF 脉冲还与片上定时器 2 (TIM2) 的输入触发相连，因此可通过 SOF 脉冲触发输入捕获功能、输出比较功能和定时器。SOF 脉冲和 TIM2 的连接通过 TIM2_OR 寄存器的 ITR1_RMP 位使能。

30.7.2 设备 SOF

在设备模式下，USB 每次接收到 SOF 令牌时，都将触发帧开始中断（OTH_FS_GINTSTS 中的 SOF 位）。相应的帧编号可从设备状态寄存器（OTG_FS_DSTS 中的 FNSOF 位）读取。使用全局控制和配置寄存器中的 SOF 输出使能位（OTG_FS_GCCFG 中的 SOFOUTEN）位，还可以生成宽度为 12 个系统时钟周期的 SOF 脉冲信号，并使该信号在 SOF 引脚输出，以实现外部可用。此外，SOF 脉冲信号还在内部与 TIM2 的输入触发相连，因此可通过 SOF 脉冲触发输入捕获功能、输出比较功能和定时器。SOF 脉冲与 TIM2 的连接通过 TIM2 选项寄存器 (TIM2_OR) 中的 ITR1_RMP 位使能。

周期性帧结束中断 (GINTSTS/EOPF) 用于在经过了 80%、85%、90% 或 95% 的帧间隔时间时通知应用程序，具体取决于设备配置寄存器中的周期性帧间隔字段（OTG_FS_DCFG 中的 PFIVL 位）。此功能可用于确定该帧的所有同步通信是否完成。

30.8 电源选项

OTG PHY 的功耗由通用模块配置寄存器中的以下三个位控制：

- PHY 掉电 (GCCFG/PWRDWN)

  用于开启/关闭 PHY 的全速收发器模块。先置位后才允许后续的 USB 操作。

- A-V_{BUS} 感应使能 (GCCFG/VBUSBSEN)

  用于开启/关闭与 A 器件操作关联的 V_{BUS} 比较器。必须在 A 器件（USB 主机）模式和 HNP 期间进行设置。

- B-V_{BUS} 感应使能 (GCCFG/VBUSASEN)

  用于开启/关闭与 B 器件操作关联的 V_{BUS} 比较器。必须在 B 器件（USB 设备）模式和 HNP 期间进行设置。

USB 会话没有开始或设备未连接时，可以在 USB 挂起状态下使用功率降低技术。

- 停止 PHY 时钟（OTG_FS_PCGCCTL 中的 STPPCLK 位）

  将时钟门控控制寄存器中的停止 PHY 时钟位置 1 时，OTG 全速模块的大多数 48 MHz 内部时钟域均由时钟门控关闭。即使应用程序仍提供时钟输入，也会节省掉模块由于时钟信号翻转带来的动态功耗

  还会关掉收发器的大部分单元，只有负责检测异步恢复事件或远程唤醒事件的部分还保持工作状态。

- HCLK 门控（OTG_FS_PCGCCTL 中的 GATEHCLK 位）

  将时钟门控控制寄存器中的 GATEHCLK 位置 1 时，OTG_FS 模块内部的大多数系统时钟域均由时钟门控关闭。只有寄存器读取和写入接口保持活动状态。即使应用程序仍提供时钟输入，也会节省掉模块由于时钟信号翻转带来的动态功耗。

- USB 系统停止

  当 OTG_FS 处于 USB 挂起状态时，应用程序可通过将 USB 系统中的所有时钟源全部关闭来显著降低总功耗。USB 系统停止可通过以下方式激活：首先将停止 PHY 时钟位置 1，然后在电源控制系统模块 (PWR) 中将系统配置为深度睡眠模式。

  OTG_FS 模块通过对 USB 上的远程唤醒（作为主机）或恢复（作为设备）信号进行异步检测，自动重新激活系统时钟和 USB 时钟。

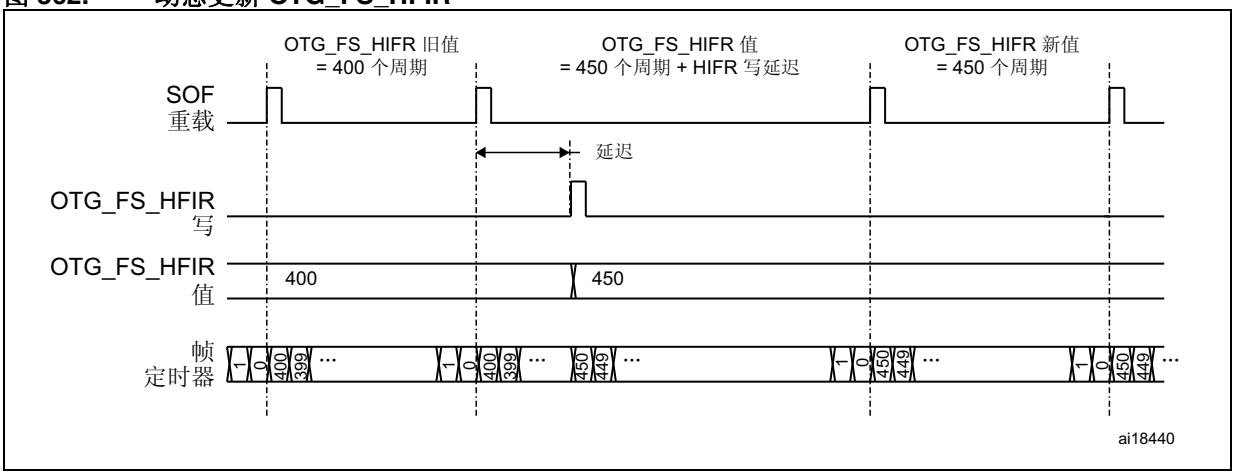
为了节省动态功耗，只在 USB 数据 FIFO 被 OTG_FS 模块访问时为其提供时钟。

30.9 动态更新 OTG_FS_HFIR 寄存器

主机模式下，USB 模块具有对微帧周期进行动态微调的功能，能够将外部设备与 micro-SOF 帧进行同步。

如果 OTG_HS_HFIR 寄存器在当前 micro-SOF 帧内发生更改，则将在下一个帧中对 SOF 周期进行相应修正，具体说明请参见*图 362*。

图 362. 动态更新 OTG_FS_HFIR

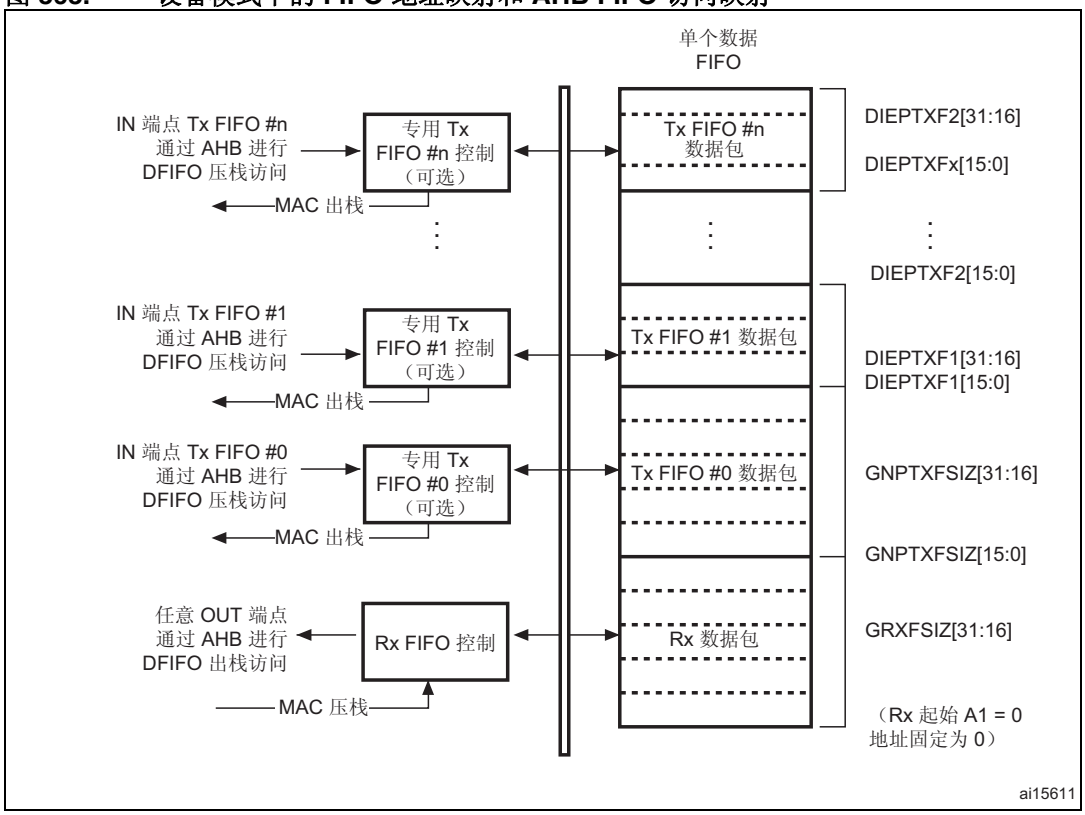


30.10 USB 数据 FIFO

USB 系统具有 1.25 KB 专用 RAM，采用复杂的 FIFO 控制机制。OTG_FS 模块中的数据包 FIFO 控制器模块将 RAM 空间划分为多个 Tx-FIFO（USB 传输前，应用程序将数据压入其中进行短暂存储）和单个 Rx FIFO（从 USB 接收到的数据被应用程序读取之前，在其中进行短暂存储）。RAM 中所构建的 FIFO 的数量与组织方式取决于设备的角色。设备模式下，为每个激活的 IN 端点配置一个 Tx FIFO。FIFO 的大小均由软件配置，以更好地满足应用要求。

30.11 设备 FIFO 架构

图 363. 设备模式下的 FIFO 地址映射和 AHB FIFO 访问映射



30.11.1 设备 Rx FIFO

OTG 设备使用单个接收 FIFO 接收发送到所有 OUT 端点的数据。只要 Rx FIFO 中有空余空间，收到的数据包就挨个填入 Rx FIFO。除了有效数据外，接收到的数据包状态（包含 OUT 端点目标编号、字节数、数据 PID 和对所接收数据的验证）也由模块进行存储。没有可用空间时，设备会回复主机事务 NACK 应答并在被寻址的端点上触发中断。接收 FIFO 的大小在接收 FIFO 大小寄存器 (GRXFSIZ) 中配置。

单个接收 FIFO 架构使得 USB 设备更高效地填充接收 RAM 缓冲区：

- 所有 OUT 端点共享同一个 RAM 缓冲区（共享 FIFO）
- 对于任意主机序列 OUT 令牌，OTG FS 模块可将接收 FIFO 填充至限值

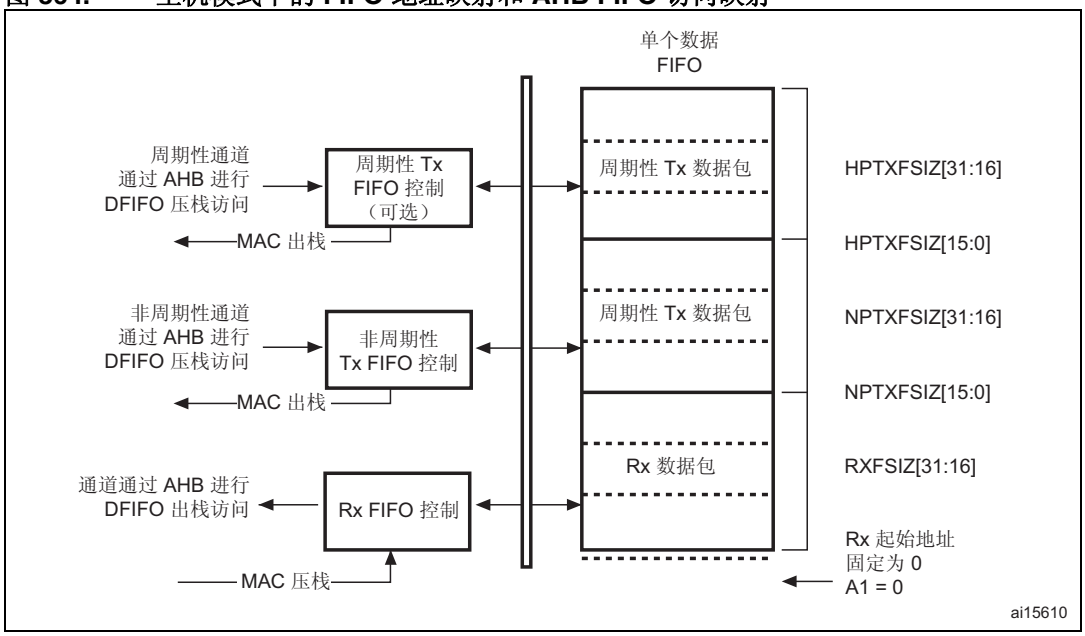
只要至少有一个数据包在 RX FIFO 中可供读取，应用程序就会一直接收 Rx-FIFO 非空中断（OTG_FS_GINTSTS 中的 RXFLVL 位）。应用程序从接收状态读取和出栈寄存器 (GRXSTSP) 中读取数据包信息，最后通过读取与端点相关的出栈地址从接收 FIFO 读出相应数据。

30.11.2 设备 Tx FIFO

模块为各个 IN 端点提供了专用的 FIFO。应用程序通过非周期发送 FIFO 大小寄存器 (OTG_FS_TX0FSIZ) 为 IN 端点 0 配置 FIFO 大小；通过设备 IN 端点发送 FIFOx 寄存器 (DIEPTXFx) 为 IN 端点 x 配置 FIFO 大小。

30.12 主机 FIFO 架构

图 364. 主机模式下的 FIFO 地址映射和 AHB FIFO 访问映射



30.12.1 主机 Rx FIFO

主机使用一个接收 FIFO 处理所有周期和非周期事务。FIFO 用作接收缓冲区以保存从 USB 接收到的数据（接收到的数据包的数据部分），直至这些数据传输到系统存储器。只要 FIFO 中有空间，来自设备 IN 端点的数据包就接收进来并挨个存储。接收到的每个数据包的状态（包含主机目标通道、字节数、数据 PID 和对所接收数据的校验）也存储在 FIFO 中。接收 FIFO 的大小在接收 FIFO 大小寄存器 (GRXFSIZ) 中配置。

单个接收 FIFO 架构使得 USB 主机高效地填充接收数据缓冲区：

- 所有 IN 配置主机通道共享同一个 RAM 缓冲区（共享 FIFO）
- 对于主机软件驱动的任意序列 IN 令牌，OTG FS 模块可将接收 FIFO 填充至限值

只要至少有一个数据包在 RX FIFO 中可供读取，应用程序就会接收 Rx FIFO 非空中断。应用程序从接收状态读取和出栈寄存器中读取数据包信息，最后从接收 FIFO 中读出数据。

30.12.2 主机 Tx FIFO

主机使用一个发送 FIFO 处理所有非周期（控制和批量）OUT 事务，使用另一个发送 FIFO 处理所有周期（同步和中断）OUT 事务。FIFO 用作发送缓冲区以保存要通过 USB 发送的数据（发送数据包）。周期（非周期）Tx FIFO 的大小在主机周期（非周期）发送 FIFO 大小 (HPTXFSIZ/HNPTXFSIZ) 寄存器中配置。

两个 Tx FIFO 按优先级实施操作，周期性通信的优先级较高，因此在 USB 一帧的时间内首先进行周期性通信。帧起始时，内置的主机调度器先处理周期请求队列，再处理非周期请求队列。

两个发送 FIFO 的架构使得 USB 主机能够对周期和非周期发送数据缓冲区分别进行优化管理：

- 配置为支持周期（非周期）OUT 事务的所有主机通道共享同一个 RAM 缓冲区（共享 FIFO）
- 对于主机软件驱动的任意序列 OUT 令牌，OTG FS 模块可将周期性（非周期性）发送 FIFO 填充至限值

只要周期性 Tx-FIFO 为半空或全空，OTG_FS 模块就会发出周期性 Tx FIFO 空中断（OTG_FS_GINTSTS 中的 PTXFE 位），具体取决于 AHB 配置寄存器中的周期性 Tx-FIFO 空等级位（OTG_FS_GAHBCFG 中的 PTXFELVL 位）的值。只要周期性 Tx FIFO 和周期性请求队列中均存在空闲空间，应用程序便可提前写入发送数据。可通过读取主机周期性发送 FIFO 和队列状态寄存器 (HPTXSTS) 来了解二者的可用空间。

只要非周期性 Tx-FIFO 为半空或全空，OTG_FS 模块就会发出非周期性 Tx FIFO 空中断（OTG_FS_GINTSTS 中的 NPTXFE 位），具体取决于 AHB 配置寄存器中的非周期性 Tx FIFO 空等级位（OTG_FS_GAHBCFG 中的 TXFELVL 位）。只要非周期性 Tx FIFO 和非周期性请求队列中均存在空闲空间，应用程序便可写入发送数据。可通过读取主机非周期性发送 FIFO 和队列状态寄存器 (HNPTXSTS) 来了解二者的可用空间。

30.13 FIFO RAM 分配

30.13.1 设备模式

接收 FIFO RAM 分配：应用程序应为 SETUP 数据包分配 RAM：接收 FIFO 中必须保留 10 个位置以在控制端点上接收 SETUP 数据包。OTG 模块不会向这些为 SETUP 数据包保留的位置写入任何其它数据。将会为全局 OUT NAK 分配一个位置。状态信息随各个接收数据包写入 FIFO。因此，必须至少为接收数据包分配（最大数据包大小 / 4）+ 1 的空间。如果使能了多个同步端点，则为接收连续数据包分配的空间必须至少为（最大数据包大小 / 4） 的两倍 + 1。通常，推荐的空间为（最大数据包/4 + 1）的两倍，这样当上一个数据包向 CPU 传送时，USB 可同时接收后续的数据包。

传输完成状态信息和该端点收到的最后一个数据包会一起被推入 FIFO。通常情况下，推荐为每个 OUT 端点分配一个位置。

发送 FIFO RAM 分配：各个 IN 端点发送 FIFO 所需的最小 RAM 空间为该特定 IN 端点的最大数据包大小。

注意： *为发送 IN 端点 FIFO 分配的空间越多，USB 的性能就越高。*

30.13.2 主机模式

接收 FIFO RAM 分配

状态信息随各个接收数据包写入 FIFO。因此，必须至少为接收数据包分配（最大数据包大小 / 4）+ 1 的空间。如果使能了多个同步通道，则为接收连续数据包分配的空间必须至少为（最大数据包大小 / 4） 的两倍 + 1。通常，推荐的空间为（最大数据包/4 + 1）的两倍，这样当上一个数据包向 CPU 传送时，USB 可同时接收后续的数据包。

传输完成状态信息和该端点收到的最后一个数据包会一起被推入 FIFO。所以必须为此分配一个位置。

发送 FIFO RAM 分配

主机非周期性发送 FIFO 所需的最小 RAM 为所支持的所有非周期性 OUT 通道上传输的最大数据包的大小。

通常，推荐的空间为最大数据包大小的两倍，这样当 USB 正在发送当前数据包的同时，AHB 可以往发送 FIFO 填入下一个数据包。

主机周期性发送 FIFO 所需的最小 RAM 为所支持的所有周期性 OUT 通道上传输的最大数据包的大小。如果至少有一个同步 OUT 端点，则空间必须至少为该通道中最大数据包大小的两倍。

注意： *为非周期性发送 FIFO 分配的空间越多，USB 的性能就越高。*

30.14 USB 系统性能

凭借大容量 RAM 缓冲区、高度可配置的 FIFO 大小、通过 AHB 压栈/出栈寄存器进行 32 位 FIFO 快速访问，尤其是高级 FIFO 控制机制可获得最佳 USB 和系统性能。实际上，无论当前 USB 序列如何，OTG_FS 均可通过该机制高效填充可用的 RAM 空间。借助这些特性：

- 应用程序有足够的裕量来计算并校正CPU的负载，从而优化 CPU 带宽利用率：
  - 应用程序可先积累大量发送数据，再通过 USB 发送出去
  - 可带来足够的时间裕量，以从接收 FIFO 读取数据
- USB 模块能够保持全速工作状态，也就是提供最大的全速带宽（尽量多的硬件自动运行，尽量少的软件参与）：
  - USB 模块可提前积累大量发送数据供其支配，从而可对 USB 数据发送进行自主管理
  - 接收缓冲区中有大量空白空间，可通过 USB 中的数据自动填满

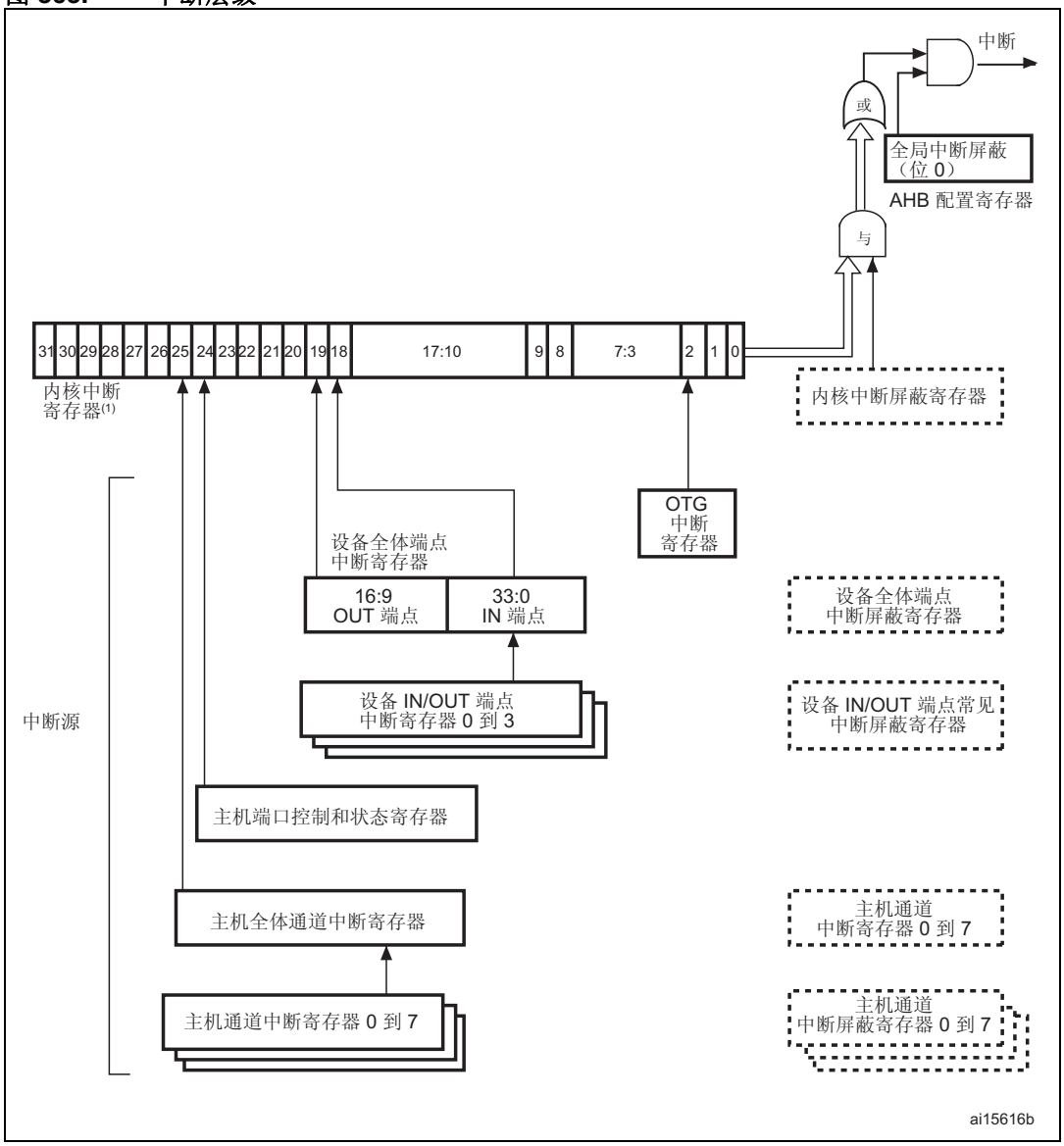
由于 OTG_FS 模块能够高效填充 1.25 KB RAM 缓冲区且 1.25 KB 发送/接收数据足以满足一个全速帧所能容纳的数据量，因此 USB 系统在一帧之内可以无需应用程序干预达到最大 USB 带宽。

30.15 OTG_FS 中断

当 OTG_FS 控制器在一种模式下（设备模式或主机模式）工作时，应用程序不得以另一种角色模式访问寄存器。如果发生了非法访问，将会产生模式不匹配中断并在模块中断寄存器（OTG_FS_GINTSTS 寄存器中的 MMIS 位）中反映。当模块从一种角色模式切换到另一种角色模式时，新工作模式下的寄存器必须重新编程为上电复位后的状态。

图 365 显示了中断层级。

图 365. 中断层级



1. 有关模块中断寄存器位，请参见*第 962 页的OTG_FS 模块中断寄存器 (OTG_FS_GINTSTS)*。

30.16 OTG_FS 控制和状态寄存器

应用程序通过 AHB 从接口对控制和状态寄存器 (CSR) 进行读写操作，以此来控制 OTG_FS 模块。这些都是 32 位寄存器，其地址按 32 位对齐，因此只能以 32 位的方式访问。

CSR 分为以下几类：

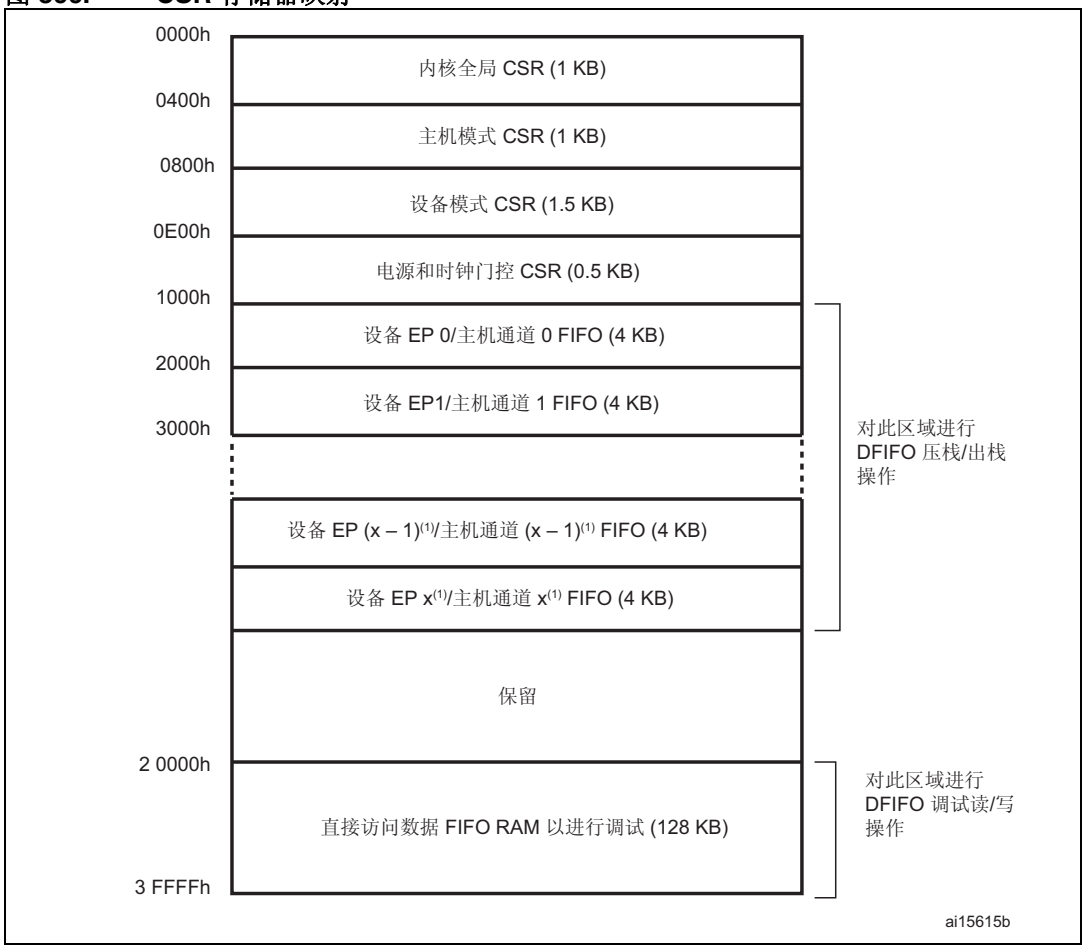
- 模块全局寄存器
- 主机模式寄存器
- 主机全局寄存器
- 主机端口 CSR
- 主机通道相关寄存器
- 设备模式寄存器
- 设备全局寄存器
- 设备端点相关寄存器
- 电源和时钟门控寄存器
- 数据 FIFO (DFIFO) 访问寄存器

只有模块全局寄存器、电源和时钟门控寄存器、数据 FIFO 访问寄存器以及主机端口控制和状态寄存器，可在主机模式和设备模式下进行访问。当 OTG_FS 控制器在一种模式下（设备模式或主机模式）工作时，应用程序不得以另一种角色模式访问寄存器。如果发生了非法访问，将会产生模式不匹配中断并在模块中断寄存器（OTG_FS_GINTSTS 寄存器中的 MMIS 位）中反映。当模块从一种角色模式切换到另一种角色模式时，新工作模式下的寄存器必须重新编程为上电复位后的状态。

30.16.1 CSR 存储器映射

主机模式寄存器和设备模式寄存器占据不同的地址。所有寄存器均在 AHB 时钟域内实现。

图 366. CSR 存储器映射



1. 设备模式下 x = 3；主机模式下 x = 7。

全局 CSR 地址映射

这些寄存器在主机模式和设备模式下均可用。

表 171. 模块全局控制和状态寄存器 (CSR)

缩写	偏移地址	寄存器名称
OTG_FS_GOTGCTL	0x000	*第 954 页的OTG_FS 控制和状态寄存器 (OTG_FS_GOTGCTL)*
OTG_FS_GOTGINT	0x004	*第 956 页的OTG_FS 中断寄存器 (OTG_FS_GOTGINT)*
OTG_FS_GAHBCFG	0x008	*第 957 页的OTG_FS AHB 配置寄存器 (OTG_FS_GAHBCFG)*
OTG_FS_GUSBCFG	0x00C	*第 958 页的OTG_FS USB 配置寄存器 (OTG_FS_GUSBCFG)*
OTG_FS_GRSTCTL	0x010	*第 960 页的OTG_FS 复位寄存器 (OTG_FS_GRSTCTL)*
OTG_FS_GINTSTS	0x014	*第 962 页的OTG_FS 模块中断寄存器 (OTG_FS_GINTSTS)*
OTG_FS_GINTMSK	0x018	*第 965 页的OTG_FS 中断屏蔽寄存器 (OTG_FS_GINTMSK)*

表 171. 模块全局控制和状态寄存器 (CSR)（续）

缩写	偏移地址	寄存器名称
OTG_FS_GRXSTSR	0x01C	*第 968 页的OTG_FS 接收状态调试读取/OTG 状态读取和出栈寄存器 (OTG_FS_GRXSTSR/OTG_FS_GRXSTSP)*
OTG_FS_GRXSTSP	0x020	
OTG_FS_GRXFSIZ	0x024	*第 970 页的OTG_FS 接收 FIFO 大小寄存器 (OTG_FS_GRXFSIZ)*
OTG_FS_HNPTXFSIZ/ OTG_FS_DIEPTXF0(1)	0x028	*OTG_FS 主机非周期性发送 FIFO 大小寄存器 (OTG_FS_HNPTXFSIZ)/端点 0 发送 FIFO 大小 (OTG_FS_DIEPTXF0)*
OTG_FS_HNPTXSTS	0x02C	*第 971 页的OTG_FS 非周期性发送 FIFO/队列状态寄存器 (OTG_FS_HNPTXSTS)*
OTG_FS_GCCFG	0x038	*第 972 页的OTG_FS 通用模块配置寄存器 (OTG_FS_GCCFG)*
OTG_FS_CID	0x03C	*第 973 页的OTG_FS 模块 ID 寄存器 (OTG_FS_CID)*
OTG_FS_HPTXFSIZ	0x100	*第 973 页的OTG_FS 主机周期性发送 FIFO 大小寄存器 (OTG_FS_HPTXFSIZ)*
OTG_FS_DIEPTXFx	0x104 0x124 ... 0x138	*第 974 页的OTG_FS 设备 IN 端点发送 FIFO 大小寄存器 (OTG_FS_DIEPTXFx)（x = 1..3，其中 x 为 FIFO_number）*

1. 通用规则是：主机模式下使用 OTG_FS_HNPTXFSIZ；设备模式下使用 OTG_FS_DIEPTXF0。

主机模式 CSR 地址映射

模块每次切换到主机模式时都必须对这些寄存器进行设置。

表 172. 主机模式控制和状态寄存器 (CSR)

缩写	偏移地址	寄存器名称
OTG_FS_HCFG	0x400	*第 974 页的OTG_FS 主机配置寄存器 (OTG_FS_HCFG)*
OTG_FS_HFIR	0x404	*第 975 页的OTG_FS 主机帧时间间隔寄存器 (OTG_FS_HFIR)*
OTG_FS_HFNUM	0x408	*第 976 页的OTG_FS 主机帧编号/帧剩余时间寄存器 (OTG_FS_HFNUM)*
OTG_FS_HPTXSTS	0x410	*第 976 页的OTG_FS_Host 周期性发送 FIFO/队列状态寄存器 (OTG_FS_HPTXSTS)*
OTG_FS_HAINT	0x414	*第 977 页的OTG_FS 主机全体通道中断寄存器 (OTG_FS_HAINT)*
OTG_FS_HAINTMSK	0x418	*第 978 页的OTG_FS 主机全体通道中断屏蔽寄存器 (OTG_FS_HAINTMSK)*
OTG_FS_HPRT	0x440	*第 978 页的OTG_FS 主机端口控制和状态寄存器 (OTG_FS_HPRT)*
OTG_FS_HCCHARx	0x500 0x520 ... 0x6E0h	*第 981 页的OTG_FS 主机通道 x 特性寄存器 (OTG_FS_HCCHARx)（x = 0..7，其中 x = 通道编号）*
OTG_FS_HCINTx	508h	*第 982 页的OTG_FS 主机通道 x 中断寄存器 (OTG_FS_HCINTx)（x = 0..7，其中 x = 通道编号）*

表 172. 主机模式控制和状态寄存器 (CSR)（续）

缩写	偏移地址	寄存器名称
OTG_FS_HCINTMSKx	50Ch	*第 983 页的OTG_FS 主机通道 x 中断屏蔽寄存器 (OTG_FS_HCINTMSKx)（x = 0..7，此处 x = 通道编号）*
OTG_FS_HCTSIZx	510h	*第 984 页的OTG_FS 主机通道 x 传输大小寄存器 (OTG_FS_HCTSIZx)（x = 0..7，此处 x = 通道编号）*

设备模式 CSR 地址映射

模块每次切换到设备模式时都必须对这些寄存器进行编程。

表 173. 设备模式控制和状态寄存器

缩写	偏移地址	寄存器名称
OTG_FS_DCFG	0x800	*第 985 页的OTG_FS 设备配置寄存器 (OTG_FS_DCFG)*
OTG_FS_DCTL	0x804	*第 986 页的OTG_FS 设备控制寄存器 (OTG_FS_DCTL)*
OTG_FS_DSTS	0x808	*第 987 页的OTG_FS 设备状态寄存器 (OTG_FS_DSTS)*
OTG_FS_DIEPMSK	0x810	*第 988 页的OTG_FS 设备 IN 端点通用中断屏蔽寄存器 (OTG_FS_DIEPMSK)*
OTG_FS_DOEPMSK	0x814	*第 989 页的OTG_FS 设备 OUT 端点通用中断屏蔽寄存器 (OTG_FS_DOEPMSK)*
OTG_FS_DAINT	0x818	*第 990 页的OTG_FS 设备全体端点中断寄存器 (OTG_FS_DAINT)*
OTG_FS_DAINTMSK	0x81C	*第 990 页的OTG_FS 全体端点中断屏蔽寄存器 (OTG_FS_DAINTMSK)*
OTG_FS_DVBUSDIS	0x828	*第 991 页的OTG_FS 设备 V_{BUS} 放电时间寄存器 (OTG_FS_DVBUSDIS)*
OTG_FS_DVBUSPULSE	0x82C	*第 991 页的OTG_FS 设备 V_{BUS} 脉冲时间寄存器 (OTG_FS_DVBUSPULSE)*
OTG_FS_DIEPEMPMSK	0x834	*第 992 页的OTG_FS 设备 IN 端点 FIFO 空中断屏蔽寄存器: OTG_FS_DIEPEMPMSK)*
OTG_FS_DIEPCTL0	0x900	*第 992 页的OTG_FS 设备控制 IN 端点 0 控制寄存器 (OTG_FS_DIEPCTL0)*
OTG_FS_DIEPCTLx	0x920 0x940 ... 0xAE0	*第 994 页的OTG 设备端点 x 控制寄存器 (OTG_FS_DIEPCTLx)（x = 1..3，其中 x = 端点编号）*
OTG_FS_DIEPINTx	0x908	*第 999 页的OTG_FS 设备端点 x 中断寄存器 (OTG_FS_DIEPINTx)（x = 0..3，其中 x = 端点编号）*
OTG_FS_DIEPTSIZ0	0x910	*第 1001 页的OTG_FS 设备 IN 端点 0 传输大小寄存器 (OTG_FS_DIEPTSIZ0)*
OTG_FS_DTXFSTSx	0x918	*第 1004 页的OTG_FS 设备 IN 端点发送 FIFO 状态寄存器 (OTG_FS_DTXFSTSx)（x = 0..3，其中 x = 端点编号）*

表 173. 设备模式控制和状态寄存器（续）

缩写	偏移地址	寄存器名称
OTG_FS_DIEPTSIZx	0x930 0x950 ... 0xAF0	*第 1004 页的OTG_FS 设备 OUT 端点 x 传输大小寄存器 (OTG_FS_DOEPTSIZx)（x = 1..3，其中 x = 端点编号）*
OTG_FS_DOEPCTL0	0xB00	*第 996 页的OTG_FS 设备控制 OUT 端点 0 控制寄存器 (OTG_FS_DOEPCTL0)*
OTG_FS_DOEPCTLx	0xB20 0xB40 ... 0xCC0 0xCE0 0xCFD	*第 994 页的OTG 设备端点 x 控制寄存器 (OTG_FS_DIEPCTLx)（x = 1..3，其中 x = 端点编号）*
OTG_FS_DOEPINTx	0xB08	*第 999 页的OTG_FS 设备端点 x 中断寄存器 (OTG_FS_DIEPINTx)（x = 0..3，其中 x = 端点编号）*
OTG_FS_DOEPTSIZx	0xB10	*第 1004 页的OTG_FS 设备 OUT 端点 x 传输大小寄存器 (OTG_FS_DOEPTSIZx)（x = 1..3，其中 x = 端点编号）*

数据 FIFO (DFIFO) 访问寄存器地址映射

这些寄存器在主机模式和设备模式下均可用，用于对给定方向的特定端点或通道的 FIFO 空间进行读写操作。如果主机通道类型为 IN，则只能对该通道上的 FIFO 进行读操作。同样地，如果主机通道类型为 OUT，则只能对该通道上的 FIFO 进行写操作。

表 174. 数据 FIFO (DFIFO) 访问寄存器地址映射

FIFO 访问寄存器段	地址范围	访问方式
设备 IN 端点 0/主机 OUT 通道 0：DFIFO 写访问 设备 OUT 端点 0/主机 IN 通道 0：DFIFO 读访问	0x1000–0x1FFC	w r
设备 IN 端点 1/主机 OUT 通道 1：DFIFO 写访问 设备 OUT 端点 1/主机 IN 通道 1：DFIFO 读访问	0x2000–0x2FFC	w r
...	...	...
设备 IN 端点 x(1)/主机 OUT 通道 x(1)：DFIFO 写访问 设备 OUT 端点 x(1)/主机 IN 通道 x(1)：DFIFO 读访问	0xX000h–0xXFFCh	w r

1. 其中，设备模式下 x 为 3；主机模式下 x 为 7。

电源和时钟门控 CSR 地址映射

由一个单独寄存器对电源和时钟门控进行管理。在主机模式和设备模式下均可使用。

表 175. 电源和时钟门控控制和状态寄存器

寄存器名称	缩写	偏移地址：0xE00–0xFFF
电源和时钟门控控制寄存器	PCGCR	0xE00-0xE04
保留		0xE05–0xFFF

30.16.2 OTG_FS 全局寄存器

这些寄存器在主机模式和设备模式下都可用，且在这两个模式间切换时无需对其进行重新编程。

除非特别说明，否则寄存器描述中的位值以二进制表示。

OTG_FS 控制和状态寄存器 (OTG_FS_GOTGCTL)

OTG_FS control and status register

偏移地址：0x000

复位值：0x0000 0800

OTG_FS_GOTGCTL 寄存器控制模块的 OTG 功能并反映其状态。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												BSVLD	ASVLD	DBCT	CIDSTS	Reserved				DHNPEN	HSHNPEN	HNPRQ	HNGSCS	Reserved						SRQ	SRQSCS
												r	r	r	r					rw	rw	rw	r							rw	r

位 31:20 保留，必须保持复位值。

位 19 **BSVLD**：B 会话有效 (B-session valid)

指示设备模式下收发器的状态。

0：B 会话无效。

1：B 会话有效。

在 OTG 模式下，该位可用于确定设备处于连接状态还是断开状态。

注意：仅可在设备模式下访问。

位 18 **ASVLD**：A 会话有效 (A-session valid)

指示主机模式下收发器的状态。

0：A 会话无效

1：A 会话有效

注意：仅可在主机模式下访问。

位 17 **DBCT**：长/短去抖动时间 (Long/short debounce time)

指示已检测到连接的去抖动时间。

0：长去抖动时间，用于物理连接 (100 ms + 2.5 µs)

1：短去抖动时间，用于软连接 (2.5 µs)

注意：仅可在主机模式下访问。

位 16 **CIDSTS**：连接器 ID 状态 (Connector ID status)

指示发生连接事件时的连接器 ID 状态。

0：OTG_FS 控制器处于 A 器件模式

1：OTG_FS 控制器处于 B 器件模式

注意：在设备模式和主机模式均可访问。

位 15:12 保留，必须保持复位值。

位 11 **DHNPEN**：使能设备 HNP 特性 (Device HNP enabled)

从所连 USB 主机处成功接收到 SetFeature.SetHNPEnable 命令时，应用程序将该位置 1。

0：未在应用程序中使能 HNP

1：已在应用程序中使能 HNP

注意：仅可在设备模式下访问。

位 10 **HSHNPEN**：主机设置 HNP 使能 (Host set HNP enable)

（使用 SetFeature.SetHNPEnable 命令）在所连接设备上成功使能 HNP 后，应用程序将该位置 1。

0：未使能主机设置 HNP

1：已使能主机设置 HNP

注意：仅可在主机模式下访问。

位 9 **HNPRQ**：HNP 请求 (HNP request)

应用程序将该位置 1 时，将对所连接 USB 主机发起 HNP 请求。当 OTG_FS_GOTGINT 寄存器中的主机协商成功状态更改位（OTG_FS_GOTGINT 中的 HNSSCHG 位）置 1 时，应用程序可通过写 0 将该位清零。HNSSCHG 位清零时，模块会将该位清零。

0：不发送 HNP 请求

1：发送 HNP 请求

注意：仅可在设备模式下访问。

位 8 **HNGSCS**：主机协商成功 (Host negotiation success)

当主机协商成功时，模块会将该位置 1。当该寄存器中的 HNP 请求位 (HNPRQ) 置 1 时，模块会将该位清零。

0：主机协商失败

1：主机协商成功

注意：仅可在设备模式下访问。

位 7:2 保留，必须保持复位值。

位 1 **SRQ**：会话请求 (Session request)

应用程序将该位置 1 时，将在 USB 上发起会话请求。当 OTG_FS_GOTGINT 寄存器中的主机协商成功状态更改位（OTG_FS_GOTGINT 中的 HNSSCHG 位）置 1 时，应用程序可通过写 0 将该位清零。HNSSCHG 位清零时，模块会将该位清零。

如要使用 USB 1.1 全速串行收发器接口来启动会话请求，则应用程序必须在该寄存器中的 B 会话有效位（OTG_FS_GOTGCTL 中的 BSVLD 位）清零后，等待 V_{BUS} 放电至 0.2 V。该放电时间因 PHY 不同而异，可从 PHY 供应商处了解相关信息。

0：无会话请求

1：会话请求

注意：仅可在设备模式下访问。

位 0 **SRQSCS**：会话请求成功 (Session request success)

当会话请求成功时，模块会将该位置 1。

0：会话请求失败

1：会话请求成功

注意：仅可在设备模式下访问。

OTG_FS 中断寄存器 (OTG_FS_GOTGINT)

OTG_FS interrupt register

偏移地址：0x04

复位值：0x0000 0000

应用程序会在出现 OTG 中断时读取该寄存器，并通过将该寄存器中的位清零来清除 OTG 中断。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												DBCDNE	ADTOCHG	HNGDET	Reserved							HNSSCHG	SRSSCHG	Reserved					SEDET	Res.	
												rc_w1	rc_w1	rc_w1								rc_w1	rc_w1						rc_w1		

位 31:20 保留，必须保持复位值。

位 19 **DBCDNE**：去抖动完成 (Debounce done)
模块会在设备连接后且去抖动结束时将该位置 1。应用程序会在看见该中断后，开始驱动 USB 复位。该位仅在 OTG_FS_GUSBCFG 寄存器中的 HNP 功能位或 SRP 功能位（分别为 OTG_FS_GUSBCFG 中的 HNPCAP 位或 SRPCAP 位）置 1 时有效。
注意：仅可在主机模式下访问。

位 18 **ADTOCHG**：A 器件超时更改 (A-device timeout change)
模块将该位置 1 时，指示 A 器件在等待 B 器件连接时超时。
注意：在设备模式和主机模式均可访问。

位 17 **HNGDET**：检测到主机协商 (Host negotiation detected)
当检测到 USB 上的主机协商请求时，模块会将该位置 1。
注意：在设备模式和主机模式均可访问。

位 16:10 保留，必须保持复位值。

位 9 **HNSSCHG**：主机协商成功状态更改 (Host negotiation success status change)
模块将在 USB 主机协商请求成功或失败时将此位置 1。应用程序必须读取 OTG_FS_GOTGCTL 寄存器中的主机协商成功位（OTG_FS_GOTGCTL 中的 HNGSCS）来检查请求成功还是失败。
注意：在设备模式和主机模式均可访问。

位 7:3 保留，必须保持复位值。

位 8 **SRSSCHG**：会话请求成功状态更改 (Session request success status change)
模块将在会话请求成功或失败时将此位置 1。应用程序必须读取 OTG_FS_GOTGCTL 寄存器中的会话请求成功位（OTG_FS_GOTGCTL 中的 SRQSCS 位）来检查请求成功还是失败。
注意：在设备模式和主机模式均可访问。

位 2 **SEDET**：检测到会话结束 (Session end detected)
模块将该位置 1 时，指示 V_{BUS} < 0.8 V，V_{BUS} 上的电压不再适用于 B 器件会话。

位 1:0 保留，必须保持复位值。

OTG_FS AHB 配置寄存器 (OTG_FS_GAHBCFG)

OTG_FS AHB configuration register

偏移地址：0x008

复位值：0x0000 0000

该寄存器可用于在上电后或更改角色模式时对模块进行配置。该寄存器主要包含 AHB 系统相关的配置参数。应用程序必须在开始任何 AHB 或 USB 事务前对该寄存器进行编程。请勿在初始编程后更改该寄存器。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																							PTXFELVL	TXFELVL	Reserved						GINTMSK
																							rw	rw							rw

位 31:20 保留，必须保持复位值。

位 8 **PTXFELVL**：周期性 TxFIFO 空门限 (Periodic TxFIFO empty level)

指示 OTG_FS_GINTSTS 寄存器中的周期性 TxFIFO 空中断位（OTG_FS_GINTSTS 中的 PTXFE 位）已触发。

0：PTXFE（位于 OTG_FS_GINTSTS）中断指示周期性 TxFIFO 为半空状态

1：PTXFE（位于 OTG_FS_GINTSTS）中断指示周期性 TxFIFO 为全空状态

注意：仅可在主机模式下访问。

位 7 **TXFELVL**：TxFIFO 空级别 (TxFIFO empty level)

在设备模式下，该位指示 IN 端点发送 FIFO 空中断（OTG_FS_DIEPINTx 中的 TXFE）已触发。

TXFE（位于 OTG_FS_DIEPINTx）中断指示 IN 端点 TxFIFO 为半空状态

TXFE（位于 OTG_FS_DIEPINTx）中断指示 IN 端点 TxFIFO 为全空状态

在主机模式下，该位指示非周期性 Tx FIFO 空中断（OTG_FS_GINTSTS 中的 NPTXFE 位）已触发。

NPTXFE（位于 OTG_FS_GINTSTS）中断指示非周期性 Tx FIFO 为半空状态

NPTXFE（位于 OTG_FS_GINTSTS）中断指示非周期性 Tx FIFO 为全空状态

位 6:1 保留，必须保持复位值。

位 0 **GINTMSK**：全局中断屏蔽 (Global interrupt mask)

该位用于屏蔽全局中断或对全局中断取消屏蔽。中断状态寄存器由模块进行更新，与此位的设置无关。

0：屏蔽应用程序触发的中断。

1：取消对应用程序触发的中断的屏蔽。

注意：在设备模式和主机模式均可访问。

OTG_FS USB 配置寄存器 (OTG_FS_GUSBCFG)

OTG_FS USB configuration register

偏移地址：0x00C

复位值：0x0000 0A00

该寄存器可用于在上电或更改角色模式后对模块进行配置。其中包含与 USB 和 USB-PHY 相关的配置参数。应用程序必须在开始任何 AHB 或 USB 事务前对该寄存器进行编程。请勿在初始编程后更改该寄存器。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CTXPKT	FDMOD	FHMOD	Reserved															TRDT				HNPCAP	SRPCAP	Res.	PHYSEL	Reserved			TOCAL		
rw	rw	rw																rw				r/rw	r/rw		wo				rw		

位 31:20 保留，必须保持复位值。

位 31 **CTXPKT**：损坏的发送数据包 (Corrupt Tx packet)

该位仅用于调试。切勿将其置 1。

注意：在设备模式和主机模式均可访问。

位 30 **FDMOD**：强制设备模式 (Force device mode)

向该位写入 1 时，可将模块强制为设备模式，而无需考虑 OTG_FS_ID 输入引脚。

0：正常模式

1：强制设备模式

将强制位置 1 后，应用程序必须等待至少 25 ms 后更改方可生效。

注意：在设备模式和主机模式均可访问。

位 29 **FHMOD**：强制主机模式 (Force host mode)

向该位写入 1 时，可将模块强制为主机模式，而无需考虑 OTG_FS_ID 输入引脚。

0：正常模式

1：强制主机模式

将强制位置 1 后，应用程序必须等待至少 25 ms 后更改方可生效。

注意：在设备模式和主机模式均可访问。

位 28:14 保留，必须保持复位值。

位 13:10 **TRDT**：USB 周转时间 (USB turnaround time)

以 PHY 时钟数为单位设置周转时间。

要计算 TRDT 的值，请使用如下公式：

TRDT = 4 × AHB 时钟 + 1 个 PHY 时钟

例如：

1. 如果 AHB 时钟频率 = 72 MHz（PHY 时钟频率 = 48 MHz），则 TRDT 设置为 9。
2. 如果 AHB 时钟频率 = 48 MHz（PHY 时钟频率 = 48 MHz），则 TRDT 设置为 5。

注意：仅可在设备模式下访问。

位 9 **HNPCAP**：HNP 使能 (HNP-capable)

应用程序使用该位控制 OTG_FS 控制器的 HNP 功能。

0：不使能 HNP 功能。

1：使能 HNP 功能。

注意：在设备模式和主机模式均可访问。

位 8 **SRPCAP**：SRP 使能 (SRP-capable)

应用程序使用该位控制 OTG_FS 控制器的 SRP 功能。如果模块作为无 SRP 功能的 B 器件，则无法请求连接的 A 器件（主机）激活 V_{BUS} 并开始会话。

0：不使能 SRP 功能。

1：使能 SRP 功能。

注意：在设备模式和主机模式均可访问。

位 7 保留，必须保持复位值。

位 6 **PHYSEL**：全速系列收发器选择 (Full Speed serial transceiver select)

此位为只写位且始终为 1。

位 5:3 保留，必须保持复位值。

位 2:0 **TOCAL**：FS 超时校准 (FS timeout calibration)

PHY 引入的额外延迟包括应用程序在该字段中设置的 PHY 时钟数，以及模块的全速数据包间超时间隔。不同 PHY 引入的延迟对数据线状态的影响是不同的。

全速操作的 USB 标准超时值为 16 到 18（含）个位时间。应用程序必须根据枚举速度编程该字段。每个 PHY 时钟增加的位时间数为 0.25 个位时间。

OTG_FS 复位寄存器 (OTG_FS_GRSTCTL)

OTG_FS reset register

偏移地址：0x10

复位值：0x2000 0000

应用程序通过此寄存器复位模块中的各项硬件特性。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AHBIDL	Reserved																				TXFNUM					TXFFLSH	RXFFLSH	Reserved	FCRST	HSRST	CSRST
r																					rw					rs	rs		rs	rs	rs

位 31 **AHBIDL**：AHB 主器件空闲 (AHB master idle)

指示 AHB 主器件状态机处于空闲情况。

注意：在设备模式和主机模式均可访问。

位 30:11 保留，必须保持复位值。

位 10:6 **TXFNUM**：TxFIFO 编号 (TxFIFO number)

使用 TxFIFO 刷新位进行 FIFO 刷新的 FIFO 编号。只有在模块将 TxFIFO 刷新位清零后，方可更改此字段。

00000：

– 主机模式下刷新非周期性 TxFIFO

– 设备模式下刷新 Tx FIFO 0

00001：

– 主机模式下刷新周期性 TxFIFO

– 设备模式下刷新 TXFIFO 1

00010：设备模式下刷新 TXFIFO 2

...

00101：设备模式下刷新 TXFIFO 15

10000：在设备模式或主机模式下刷新所有的发送 FIFO。

注意：在设备模式和主机模式均可访问。

位 5 **TXFFLSH**：TxFIFO 刷新 (TxFIFO flush)

此位选择性地刷新一个或所有的发送 FIFO，但当模块处理通信事务时无法执行该操作。

只有在确认模块当前未对 TxFIFO 执行读写操作后，应用程序方可对此位执行写操作。使用以下寄存器进行确认：

读 — NAK 有效中断可确保模块当前未对 FIFO 执行读操作

写 — OTG_HS_GRSTCTL 中的 AHBIDL 位可确保模块当前未对 FIFO 执行任何写操作。

注意：在设备模式和主机模式均可访问。

位 4 **RXFFLSH**：RxFIFO 刷新 (RxFIFO flush)

应用程序可使用此位刷新整个 RxFIFO，但必须首先确保模块当前未在处理通信事务。

只有在确认模块当前未对 RxFIFO 执行读写操作后，应用程序方可对此位执行写操作。

应用程序必须等到此位清零后，方可执行其它操作。通常需要等待 8 个时钟周期（以 PHY 或 AHB 时钟中最慢的为准）。

注意：在设备模式和主机模式均可访问。

位 3 保留，必须保持复位值。

位 2 **FCRST**：主机帧计数器复位 (Host frame counter reset)

应用程序对该位执行写操作时，模块中的帧数计数器复位。帧计数器复位后，由模块发送的下一个 SOF 的帧号为 0。

注意：仅可在主机模式下访问。

位 1 **HSRST**：HCLK 软复位 (HCLK soft reset)

应用程序使用此位来刷新 AHB 时钟域中的控制逻辑。仅复位 AHB 时钟域流水线。

FIFO 不通过此位来刷新。

遵照协议终止 AHB 上的事务后，AHB 时钟域中的所有状态机均复位至空闲状态。

AHB 时钟域状态机所使用的 CSR 控制位清零。

要清零该中断，需要将由 AHB 时钟域状态机生成并用于控制中断状态的状态屏蔽位清零。

由于中断状态位并未清零，因此应用程序可以获取在该位置 1 后所发生的所有模块事件的状态。

此位为自清零位，模块将在其中所有必要逻辑复位后将该位清零。该过程需要若干个时钟的时间，具体取决于模块的当前状态。

注意：在设备模式和主机模式均可访问。

位 0 **CSRST**：模块软复位 (Core soft reset)

按如下所述将 HCLK 和 PCLK 域复位：

除以下各位外，将各个中断和所有 CSR 寄存器位清零：

– OTG_FS_PCGCCTL 中的 RSTPDMODL 位
– OTG_FS_PCGCCTL 中的 GAYEHCLK 位
– OTG_FS_PCGCCTL 中的 PWRCLMP 位
– OTG_FS_PCGCCTL 中的 STPPCLK 位
– OTG_FS_HCFG 中的 FSLSPCS 位
– OTG_FS_DCFG 中的 DSPD 位

将所有模块状态机（AHB 从器件除外）复位至空闲状态，并清空所有发送 FIFO 和接收 FIFO。

在 AHB 传输的最后数据阶段结束后，尽快终止 AHB 主器件上的所有事务。立即终止 USB 上的所有事务。

应用程序可在需要复位模块时随时对该位执行写操作。该位为自清零位，模块将在其中所有必要逻辑复位后将该位清零，该过程需要若干个时钟的时间，具体取决于模块的当前状态。该位一旦清零，软件必须等待至少 3 个 PHY 时钟后才可以访问 PHY 域（同步延迟）。此外，软件还必须在确定该寄存器中的位 31 置 1（AHB 主器件空闲）后方可开始运行。

软件复位通常在两种情况下使用，一是软件开发期间，二是用户动态更改以上所列 USB 配置寄存器中的 PHY 选择位后。用户更改 PHY 时，将为 PHY 选择相应的时钟并用于 PHY 域中。一旦选择了新的时钟，则必须复位 PHY 域，才能保证正常运行。

注意：在设备模式和主机模式均可访问。

OTG_FS 模块中断寄存器 (OTG_FS_GINTSTS)

OTG_FS core interrupt register

偏移地址：0x014

复位值：0x0400 0020

该寄存器用于在当前模式（设备模式或主机模式）下借助系统级别的事件来中断应用程序。

该寄存器中的某些位仅在主机模式下有效，而其它位则仅在设备模式下有效。此外，该寄存器还可指示当前模式。要将 rc_w1 类型中断状态位清零，应用程序必须向该位写入 1。

FIFO 状态中断为只读；如果软件在处理这些中断期间对 FIFO 执行读写操作，则 FIFO 中断标志将自动清零。

使能中断位前，应用程序必须在初始化时将 OTG_FS_GINTSTS 寄存器清零，才可以避免在初始化前产生任何中断。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WKUINT	SRQINT	DISCINT	CIDSCHG	Reserved	PTXFE	HCINT	HPRTINT	Reserved		IPXFR/INCOMPISOOUT	IISOIXFR	OEPINT	IEPINT	Reserved		EOPF	ISOODRP	ENUMDNE	USBRST	USBSUSP	ESUSP	Reserved		GOUTNAKEFF	GINAKEFF	NPTXFE	RXFLVL	SOF	OTGINT	MMIS	CMOD
rc_w1					r	r	r	Res.		rc_w1		r	r			rc_w1								r	r	r	r	rc_w1	r	rc_w1	r

位 31 **WKUPINT**：检测到恢复/远程唤醒中断 (Resume/remote wakeup detected interrupt)

在设备模式下，当 USB 总线上检测到恢复信号时，将触发该中断。在主机模式下，当 USB 上检测到远程唤醒时，将触发该中断。

注意：在设备模式和主机模式均可访问。

位 30 **SRQINT**：检测到会话请求/新会话中断 (Session request/new session detected interrupt)

在主机模式下，当检测到来自设备的会话请求时，将触发该中断。在设备模式下，当 V_{BUS} 在 B 外设设备的有效范围内时，将触发该中断。在设备模式和主机模式均可访问。

位 29 **DISCINT**：检测到断开连接中断 (Disconnect detected interrupt)

当检测到设备断开连接时触发该中断。

注意：仅可在主机模式下访问。

位 28 **CIDSCHG**：连接器 ID 线状态更改 (Connector ID status change)

当连接器 ID 线状态发生更改时，模块将该位置 1。

注意：在设备模式和主机模式均可访问。

位 27 保留，必须保持复位值。

位 26 **PTXFE**：周期性 TxFIFO 空 (Periodic TxFIFO empty)

当周期性发送 FIFO 为半空或全空状态，且周期性请求队列中存在可写入至少一个条目的空间时，将触发该中断。该 FIFO 为半空状态还是全空状态由 OTG_FS_GAHBCFG 寄存器中的周期性 TxFIFO 空级别位（OTG_FS_GAHBCFG 中的 PTXFELVL 位）决定。

注意：仅可在主机模式下访问。

位 25 **HCINT**：主机通道中断 (Host channels interrupt)

模块将该位置 1 时，指示模块中一个通道上存在挂起的中断（在主机模式下）。应用程序必须读取主机 OTG_FS_HAINT 寄存器，以确定发生中断的通道的准确编号，然后读取相应的 OTG_FS_HCINTx 寄存器，以确定引发中断的确切原因。应用程序必须先将 OTG_FS_HCINTx 寄存器的相应状态位清零，之后才能将该位清零。

注意：仅可在主机模式下访问。

位 24 **HPRTINT**：主机端口中断 (Host port interrupt)

模块将该位置 1 时，指示主机模式下 OTG_FS 控制器端口的状态发生变化。应用程序必须读取 OTG_FS_HPRT 寄存器，以确定引发此中断的确切事件。应用程序必须先将 OTG_FS_HPRT 寄存器的相应状态位清零，之后才能将该位清零。

注意：仅可在主机模式下访问。

位 23:22 保留，必须保持复位值。

位 21 **IPXFR**：未完成周期性传输 (Incomplete periodic transfer)

在主机模式下，如果存在仍处于挂起状态的未完成周期性事务，而这些事务计划在当前帧期间完成，则模块会将该中断位置 1。

INCOMPISOOUT：未完成 OUT 同步传输 (Incomplete isochronous OUT transfer)

在设备模式下，模块将该中断置 1 时，指示当前帧中至少有一个同步 OUT 端点上的传输未完成。该中断随该寄存器中的周期性帧结束中断 (EOPF) 位一同触发。

位 20 **IISOIXFR**：未完成 IN 同步传输 (Incomplete isochronous IN transfer)

模块将该中断置 1 时，指示当前帧中至少有一个同步 IN 端点上的传输未完成。该中断随该寄存器中的周期性帧结束中断 (EOPF) 位一同触发。

注意：仅可在设备模式下访问。

位 19 **OEPINT**：OUT 端点中断 (OUT endpoint interrupt)

模块将该位置 1 时，指示模块中一个 OUT 端点上存在挂起的中断（在设备模式下）。应用程序必须读取主机 OTG_FS_DAINT 寄存器，以确定发生中断的 OUT 端点的准确编号，然后读取相应的 OTG_FS_DOEPINTx 寄存器，以确定引发中断的确切原因。应用程序必须先将相应 OTG_FS_DOEPINTx 寄存器的相应状态位清零，之后才能将该位清零。

注意：仅可在设备模式下访问。

位 18 **IEPINT**：IN 端点中断 (IN endpoint interrupt)

模块将该位置 1 时，指示模块中一个 IN 端点上存在挂起的中断（在设备模式下）。应用程序必须读取主机 OTG_FS_DAINT 寄存器，以确定发生中断的 IN 端点的准确编号，然后读取相应的 OTG_FS_DIEPINTx 寄存器，以确定引发中断的确切原因。应用程序必须先将相应 OTG_FS_DIEPINTx 寄存器的相应状态位清零，之后才能将该位清零。

注意：仅可在设备模式下访问。

位 17:16 保留，必须保持复位值。

位 15 **EOPF**：周期性帧结束中断 (End of periodic frame interrupt)

指示当前帧已达到 OTG_FS_DCFG 寄存器中周期性帧间隔字段（OTG_FS_DCFG 中的 PFIVL 位）所指定的周期。

注意：仅可在设备模式下访问。

位 14 **ISOODRP**：丢弃同步 OUT 数据包中断 (Isochronous OUT packet dropped interrupt)

如果由于 RxFIFO 空间不足，无法容纳同步 OUT 端点的最大数据包，从而导致模块无法向 RxFIFO 写入同步 OUT 数据包，模块会将该位置 1。

注意：仅可在设备模式下访问。

位 13 **ENUMDNE**：枚举完成 (Enumeration done)

模块将该位置 1 时，指示速度枚举已完成。应用程序必须读取 OTG_FS_DSTS 寄存器来获取枚举速度。

注意：仅可在设备模式下访问。

位 12 **USBRST**：USB 复位 (USB reset)

模块将该位置 1 时，指示在 USB 上检测到复位信号。

注意：仅可在设备模式下访问。

位 11 **USBSUSP**：USB 挂起 (USB suspend)

模块将该位置 1 时，指示在 USB 上检测到挂起状态。当 USB 总线上的空闲状态保持 3 ms，模块便会进入挂起状态。

注意：仅可在设备模式下访问。

位 10 **ESUSP**：早期挂起 (Early suspend)

模块将该位置 1 时，指示已检测到 USB 处于空闲状态的时间达到 3 ms。

注意：仅可在设备模式下访问。

位 9:8 保留，必须保持复位值。

位 7 **GONAKEFF**：全局 OUT NAK 有效 (Global OUT NAK effective)

指示 OTG_FS_DCTL 寄存器中由应用程序设置的“将全局 OUT NAK 置 1”位（OTG_FS_DCTL 中的 SGONAK 位）已在模块中生效。通过写入 OTG_FS_DCTL 寄存器中的“将全局 OUT NAK 清零”位（OTG_FS_DCTL 中的 CGONAK 位），可将该位清零。

注意：仅可在设备模式下访问。

位 6 **GINAKEFF**：全局非周期性 IN NAK 有效 (Global IN nonperiodic NAK effective)

指示 OTG_FS_DCTL 寄存器中由应用程序设置的“将全局非周期性 IN NAK 置 1”位（OTG_FS_DCTL 中的 SGINAK 位）已在模块中生效。也就是说，模块已对应用程序设置的全局 IN NAK 位进行采样，结果已生效。通过清零 OTG_FS_DCTL 寄存器中的“将全局非周期性 IN NAK 清零”位（OTG_FS_DCTL 中的 CGINAK 位），可将该位清零。

此中断不一定表示 USB 上已发送了一个 NAK 握手信号。STALL 位优先级高于 NAK 位。

注意：仅可在设备模式下访问。

位 5 **NPTXFE**：非周期性 TxFIFO 空 (Non-periodic TxFIFO empty)

当非周期性 TxFIFO 为半空或全空状态，且非周期性发送请求队列中至少存在可写入一个条目的空间时，将触发该中断。该 FIFO 为半空状态还是全空状态由 OTG_FS_GAHBCFG 寄存器中的非周期性 TxFIFO 空级别位（OTG_FS_GAHBCFG 中的 TXFELVL 位）决定。

注意：仅可在主机模式下访问。

位 4 **RXFLVL**：RxFIFO 非空 (RxFIFO non-empty)

指示 RxFIFO 中至少有一个数据包等待读取。

注意：在主机模式和设备模式均可访问。

位 3 **SOF**：帧起始 (Start of frame)

在主机模式下，模块将该位置 1 时，指示 USB 上已发送一个 SOF (FS) 或 Keep-Alive (LS) 信号。应用程序必须将此位置 1 才可清除该中断。

在设备模式下，模块将该位置 1 时，指示 USB 上已接收到一个 SOF 令牌。应用程序可通过读取设备状态寄存器来获得当前的帧编号。只有在模块以 FS 模式运行时，才会出现此中断。

注意：在主机模式和设备模式均可访问。

位 2 **OTGINT**：OTG 中断 (OTG interrupt)

模块将该位置 1 时，指示出现 OTG 协议事件。应用程序必须读取 OTG 中断状态 (OTG_FS_GOTGINT) 寄存器，以确定引发此中断的确切事件。应用程序必须先将 OTG_FS_GOTGINT 寄存器的相应状态位清零，之后才能将该位清零。

注意：在主机模式和设备模式均可访问。

位 1 **MMIS**：模式不匹配中断 (Mode mismatch interrupt)

当应用程序尝试访问以下寄存器时，模块将该位置 1：

– 模块运行在设备模式下访问主机模式寄存器

– 模块运行在主机模式下访问设备模式寄存器

寄存器访问在 AHB 上以 OKAY 响应结束，但该访问在内部被模块忽略并且不会影响模块运行。

注意：在主机模式和设备模式均可访问。

位 0 **CMOD**：当前工作模式 (Current mode of operation)

指示当前模式。

0：设备模式

1：主机模式

注意：在主机模式和设备模式均可访问。

OTG_FS 中断屏蔽寄存器 (OTG_FS_GINTMSK)

OTG_FS interrupt mask register

偏移地址：0x018

复位值：0x0000 0000

该寄存器与模块中断寄存器结合使用，以中断应用程序。如果将某个中断位屏蔽，则不会产生与该位相关的中断。但是，与该中断相对应的模块中断 (OTG_FS_GINTSTS) 寄存器位仍会置 1。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WUIM	SRQIM	DISCINT	CIDSCHGM	Reserved	PTXFEM	HCIM	PRTIM	Reserved		IPXFRM/IISOOXFRM	IISOIXFRM	OEPINT	IEPINT	EPMISM	Reserved	EOPFM	ISOODRPM	ENUMDNEM	USBRST	USBSUSPM	ESUSPM	Reserved		GONAKEFFM	GINAKEFFM	NPTXFEM	RXFLVLM	SOFM	OTGINT	MMISM	Reserved
rw	rw	rw	rw		rw	rw	r			rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw	rw	

位 31 **WUIM**：检测到恢复/远程唤醒中断屏蔽 (Resume/remote wakeup detected interrupt mask)

0：屏蔽中断

1：使能中断

注意：在主机模式和设备模式均可访问。

位 30 **SRQIM**：检测到会话请求/新会话中断屏蔽 (Session request/new session detected interrupt mask)

0：屏蔽中断

1：使能中断

注意：在主机模式和设备模式均可访问。

位 29 **DISCINT**：检测到断开连接中断屏蔽 (Disconnect detected interrupt mask)

0：屏蔽中断

1：使能中断

注意：仅可在设备模式下访问。

位 28 **CIDSCHGM**：连接器 ID 状态更改屏蔽 (Connector ID status change mask)
0：屏蔽中断
1：使能中断
注意：在主机模式和设备模式均可访问。

位 27 保留，必须保持复位值。

位 26 **PTXFEM**：周期性 TxFIFO 空屏蔽 (Periodic TxFIFO empty mask)
0：屏蔽中断
1：使能中断
注意：仅可在主机模式下访问。

位 25 **HCIM**：主机通道中断屏蔽 (Host channels interrupt mask)
0：屏蔽中断
1：使能中断
注意：仅可在主机模式下访问。

位 24 **PRTIM**：主机端口中断屏蔽 (Host port interrupt mask)
0：屏蔽中断
1：使能中断
注意：仅可在主机模式下访问。

位 23:22 保留，必须保持复位值。

位 21 **IPXFRM**：未完成周期性传输中断屏蔽 (Incomplete periodic transfer mask)
0：屏蔽中断
1：使能中断
注意：仅可在主机模式下访问。
IISOOXFRM：未完成 OUT 同步传输中断屏蔽 (Incomplete isochronous OUT transfer mask)
0：屏蔽中断
1：使能中断
注意：仅可在设备模式下访问。

位 20 **IISOIXFRM**：未完成 IN 同步传输中断屏蔽 (Incomplete isochronous IN transfer mask)
0：屏蔽中断
1：使能中断
注意：仅可在设备模式下访问。

位 19 **OEPINT**：OUT 端点中断屏蔽 (OUT endpoints interrupt mask)
0：屏蔽中断
1：使能中断
注意：仅可在设备模式下访问。

位 18 **IEPINT**：IN 端点中断屏蔽 (IN endpoints interrupt mask)
0：屏蔽中断
1：使能中断
注意：仅可在设备模式下访问。

位 17 **EPMISM**：端点不匹配中断屏蔽 (Endpoint mismatch interrupt mask)
0：屏蔽中断
1：使能中断
注意：仅可在设备模式下访问。

位 16 保留，必须保持复位值。

位 15 **EOPFM**：周期性帧结束中断屏蔽 (End of periodic frame interrupt mask)
0：屏蔽中断
1：使能中断
注意：仅可在设备模式下访问。

位 14 **ISOODRPM**：丢弃同步 OUT 数据包中断屏蔽 (Isochronous OUT packet dropped interrupt mask)
0：屏蔽中断
1：使能中断
注意：仅可在设备模式下访问。

位 13 **ENUMDNEM**：枚举完成中断屏蔽 (Enumeration done mask)
0：屏蔽中断
1：使能中断
注意：仅可在设备模式下访问。

位 12 **USBRST**：USB 复位中断屏蔽 (USB reset mask)
0：屏蔽中断
1：使能中断
注意：仅可在设备模式下访问。

位 11 **USBSUSPM**：USB 挂起中断屏蔽 (USB suspend mask)
0：屏蔽中断
1：使能中断
注意：仅可在设备模式下访问。

位 10 **ESUSPM**：早期挂起中断屏蔽 (Early suspend mask)
0：屏蔽中断
1：使能中断
注意：仅可在设备模式下访问。

位 9:8 保留，必须保持复位值。

位 7 **GONAKEFFM**：全局 OUT NAK 生效中断屏蔽 (Global OUT NAK effective mask)
0：屏蔽中断
1：使能中断
注意：仅可在设备模式下访问。

位 6 **GINAKEFFM**：全局非周期性 IN NAK 生效中断屏蔽 (Global non-periodic IN NAK effective mask)
0：屏蔽中断
1：使能中断
注意：仅可在设备模式下访问。

位 5 **NPTXFEM**：非周期性 TxFIFO 空中断屏蔽 (Non-periodic TxFIFO empty mask)
0：屏蔽中断
1：使能中断
注意：仅可在主机模式下访问。

位 4 **RXFLVLM**：接收 FIFO 非空中断屏蔽 (Receive FIFO nonempty mask)
0：屏蔽中断
1：使能中断
注意：在设备模式和主机模式均可访问。

位 3 **SOFM**：帧起始中断屏蔽 (Start of frame mask)
0：屏蔽中断
1：使能中断
注意：在设备模式和主机模式均可访问。

位 2 **OTGINT**：OTG 中断屏蔽 (OTG interrupt mask)
0：屏蔽中断
1：使能中断
注意：在设备模式和主机模式均可访问。

位 1 **MMISM**：模式不匹配中断屏蔽 (Mode mismatch interrupt mask)
0：屏蔽中断
1：使能中断
注意：在设备模式和主机模式均可访问。

位 0 保留，必须保持复位值。

OTG_FS 接收状态调试读取/OTG 状态读取和出栈寄存器 (OTG_FS_GRXSTSR/OTG_FS_GRXSTSP)

OTG_FS Receive status debug read/OTG status read and pop registers

读取的偏移地址：0x01C

出栈的偏移地址：0x020

复位值：0x0000 0000

读取接收状态调试读取寄存器将返回接收 FIFO 顶部的内容。读取接收状态读取和出栈寄存器将额外弹出 RxFIFO 顶部的数据条目。

接收状态内容在主机模式和设备模式下的解释不同。当接收 FIFO 为空时，模块会忽略对该寄存器的读取或出栈操作，并返回值 0x0000 0000。当模块中断寄存器的接收 FIFO 非空位（OTG_FS_GINTSTS 中的 RXFLVL 位）置位时，应用程序必须仅弹出接收状态 FIFO。

主机模式：

31 … 21	20:17	16:15	14:4	3:0
Reserved	PKTSTS	DPID	BCNT	CHNUM
	r	r	r	r

位 31:21 保留，必须保持复位值。

位 20:17 **PKTSTS**：数据包状态 (Packet status)
指示接收的数据包的状态
0010：接收到 IN 数据包
0011：IN 传输完成（触发中断）
0101：数据同步错误（触发中断）
0111：暂停通道（触发中断）
其它值：保留

位 16:15 **DPID**：数据 PID (Data PID)
指示接收的数据包的数据 PID
00：DATA0
10：DATA1
01：DATA2
11：MDATA

位 14:4 **BCNT**：字节计数 (Byte count)
指示接收的 IN 数据包的字节数。

位 3:0 **CHNUM**：通道编号 (Channel number)
指示当前接收的数据包所属的通道编号。

设备模式：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved							FRMNUM				PKTSTS				DPID		BCNT												EPNUM			
							r				r				r		r												r			

位 31:25 保留，必须保持复位值。

位 24:21 **FRMNUM**：帧编号 (Frame number)
这是在 USB 上接收的数据包帧编号的 4 个最低有效位。只有当支持同步 OUT 端点时才支持此字段。

位 20:17 **PKTSTS**：数据包状态 (Packet status)
指示接收的数据包的状态
0001：全局 OUT NAK（触发中断）
0010：接收到 OUT 数据包
0011：OUT 传输完成（触发中断）
0100：SETUP 事务完成（触发中断）
0110：接收到 SETUP 数据包
其它值：保留

位 16:15 **DPID**：数据 PID (Data PID)
指示接收的 OUT 数据包的数据 PID
00：DATA0
10：DATA1
01：DATA2
11：MDATA

位 14:4 **BCNT**：字节计数 (Byte count)
指示接收的数据包的字节数。

位 3:0 **EPNUM**：端点编号 (Endpoint number)
指示当前接收的数据包所属的端点编号。

OTG_FS 接收 FIFO 大小寄存器 (OTG_FS_GRXFSIZ)

OTG_FS Receive FIFO size register

偏移地址：0x024

复位值：0x0000 0200

此应用程序可以对必须分配给 RxFIFO 的 RAM 大小进行编程。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																RXFD															
																r/rw															

位 31:16 保留，必须保持复位值。

位 15:0 **RXFD**：RxFIFO 深度 (RxFIFO depth)
以 32 位字为单位。
最小值为 16
最大值为 256
上电复位值为最大 Rx 数据 FIFO 深度。

OTG_FS 主机非周期性发送 FIFO 大小寄存器 (OTG_FS_HNPTXFSIZ)/端点 0 发送 FIFO 大小 (OTG_FS_DIEPTXF0)

OTG_FS Host non-periodic transmit FIFO size register

偏移地址：0x028

复位值：0x0000 0200

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NPTXFD/TX0FD																NPTXFSA/TX0FSA															
r/rw																r/rw															

主机模式

位 31:16 **NPTXFD**：非周期性 TxFIFO 深度 (Non-periodic TxFIFO depth)
以 32 位字为单位。
最小值为 16
最大值为 256

位 15:0 **NPTXFSA**：非周期性发送 RAM 起始地址 (Non-periodic transmit RAM start address)
此字段包含非周期性发送 FIFO RAM 的存储器起始地址。

设备模式

位 31:16 **TX0FD**：端点 0 TxFIFO 深度 (Endpoint 0 TxFIFO depth)
以 32 位字为单位。
最小值为 16
最大值为 256

位 15:0 **TX0FSA**：端点 0 发送 RAM 起始地址 (Endpoint 0 transmit RAM start address)
此字段包含端点 0 发送 FIFO RAM 的存储器起始地址。

OTG_FS 非周期性发送 FIFO/队列状态寄存器 (OTG_FS_HNPTXSTS)

OTG_FS non-periodic transmit FIFO/queue status register

偏移地址：0x02C

复位值：0x0008 0200

注意： *在设备模式下，此寄存器无效。*

此只读寄存器包含非周期性 TxFIFO 和非周期性发送请求队列的自由空间信息。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	NPTXQTOP							NPTQXSAV								NPTXFSAV															
	r							r								r															

位 31 保留，必须保持复位值。

位 30:24 **NPTXQTOP**：非周期性发送请求队列顶部 (Top of the non-periodic transmit request queue)
非周期性发送请求队列中 MAC 目前正在处理的条目。
位 30:27：通道/端点编号 (Channel/endpoint number)
位 26:25：
- 00：IN/OUT 令牌
- 01：长度为零的发送数据包（设备 IN/主机 OUT）
- 11：通道停止命令

位 24：结束（所选通道/端点的最后一个条目）(Terminate (last entry for selected channel/endpoint))

位 23:16 **NPTQXSAV**：非周期性发送请求队列可用空间 (Non-periodic transmit request queue space available)
指示非周期性发送请求队列中的可用空闲空间大小。在主机模式下，此队列保存 IN 和 OUT 请求。设备模式仅具有 IN 请求。
00：非周期性发送请求队列已满
01：1 个位置可用
10：2 个位置可用
bxn：*n* 个位置可用 ($0 \leq n \leq 8$)
其它值：保留

位 15:0 **NPTXFSAV**：非周期性 TxFIFO 可用空间 (Non-periodic TxFIFO space available)
指示非周期性 TxFIFO 中的可用空闲空间大小。
以 32 位字为单位。
00：非周期性 TxFIFO 已满
01：1 个字可用
10：2 个字可用
0xn：*n* 个字可用（其中，0 ≤ *n* ≤ 256）
其它值：保留

OTG_FS 通用模块配置寄存器 (OTG_FS_GCCFG)

OTG_FS general core configuration register

偏移地址：0x038

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved										NOVBUSSENS	SOFOUTEN	VBUSBSEN	VBUSASEN	Reserved	.PWRDWN	Reserved															
										rw	rw	rw	rw		rw																

位 31:22 保留，必须保持复位值。

位 21 **NOVBUSSENS**：V_{BUS} 感应禁止选项 (V_{BUS} sensing disable option)
该位置 1 时，在内部将 VBUS 视为始终位于 V_{BUS} 有效电平 (5 V)。该选项消除了对 V_{BUS} 引脚的需求，使该引脚可自由用于其它用途，例如该引脚上的其他功能。V_{BUS} 连接可重映射到其它通用输入引脚，并由软件监视。
此选项仅适用于只作主机或只作设备。
0：硬件支持 V_{BUS} 感应
1：硬件不支持 VBUS 感应

位 20 **SOFOUTEN**：SOF 输出使能 (SOF output enable)
0：SOF 脉冲不从引脚输出
1：SOF 脉冲可从引脚输出

位 19 **VBUSBSEN**：使能“B”器件的 V_{BUS} 感应功能 (Enable the V_{BUS} sensing “B” device)
0：禁止“B”器件的 V_{BUS} 感应功能
1：使能“B”器件的 V_{BUS} 感应功能

位 18 **VBUSASEN**：使能“A”器件的 V_{BUS} 感应功能 (Enable the V_{BUS} sensing “A” device)
0：禁止“A”器件的 V_{BUS} 感应功能
1：使能“A”器件的 V_{BUS} 感应功能

位 17 保留，必须保持复位值。

位 16 **PWRDWN**：掉电 (Power down)
用于在发送/接收时激活收发器
0：掉电激活
1：掉电停用（“收发器激活”）

位 15:0 保留，必须保持复位值。

OTG_FS 模块 ID 寄存器 (OTG_FS_CID)

OTG_FS core ID register

偏移地址：0x03C

复位值：0x0000 1100

该寄存器为只读寄存器，包含产品 ID。

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

位 31:0 **PRODUCT_ID**：产品 ID 字段 (Product ID field)
可通过应用程序编程的 ID 字段。

OTG_FS 主机周期性发送 FIFO 大小寄存器 (OTG_FS_HPTXFSIZ)

OTG_FS Host periodic transmit FIFO size register

偏移地址：0x100

复位值：0x0200 0600

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PTXFSIZ																PTXSA															
r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw

位 31:16 **PTXFD**：主机周期性 TxFIFO 深度 (Host periodic TxFIFO depth)
以 32 位字为单位。
最小值为 16

位 15:0 **PTXSA**：主机周期性 TxFIFO 起始地址 (Host periodic TxFIFO start address)
上电复位值是最大 Rx 数据 FIFO 深度与最大非周期性 Tx 数据 FIFO 深度之和。

OTG_FS 设备 IN 端点发送 FIFO 大小寄存器 (OTG_FS_DIEPTXFx)（x = 1..3，其中 x 为 FIFO_number）

OTG_FS device IN endpoint transmit FIFO size register

偏移地址：0x104 + (FIFO_number – 1) × 0x04

复位值：0x02000400

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INEPTXFD																INEPTXSA															
r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw

位 31:16 **INEPTXFD**：IN 端点 TxFIFO 深度

以 32 位字为单位。

最小值为 16

上电复位值为最大 IN 端点 FIFO 深度。

位 15:0 **INEPTXSA**：IN 端点发送 FIFOx RAM 起始地址 (IN endpoint FIFOx transmit RAM start address)

此字段包含 IN 端点发送 FIFOx 的存储器起始地址。该地址必须与 32 位存储器位置对齐。

30.16.3 主机模式寄存器

除非特别说明，否则寄存器描述中的位值以二进制表示。

主机模式寄存器会影响主机模式下的模块操作。在设备模式下不得访问主机模式寄存器，因为产生的结果不明确。主机模式寄存器可进行如下分类：

OTG_FS 主机配置寄存器 (OTG_FS_HCFG)

OTG_FS Host configuration register

偏移地址：0x400

复位值：0x0000 0000

此寄存器将在上电后对模块进行配置。请勿在初始化主机后更改此寄存器。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																													FSLSS	FSLSPCS	
																													r	rw	rw

位 31:3 保留，必须保持复位值。

位 2 **FSLSS**：仅支持 FS 和 LS (FS- and LS-only support)

应用程序使用此位控制模块的枚举速度。使用此位，应用程序可使模块工作为 FS 主机，即使所连接的设备支持 HS 通信也是如此。请勿在初始编程后更改此字段。

1：仅限 FS/LS，即使所连接设备可支持 HS（只读）

位 1:0 **FSLSPCS**：FS/LS PHY 时钟选择 (FS/LS PHY clock select)

当模块处于 FS 主机模式时

01：PHY 时钟以 48 MHz 运行

其它值：保留

当模块处于 LS 主机模式时

00：保留

01：选择 48 MHz PHY 时钟频率

10：选择 6 MHz PHY 时钟频率

11：保留

注意：当设备连上主机时，必须依照所连接设备的速度设置 FSLSPCS（更改此位后，必须进行软件复位）。

OTG_FS 主机帧时间间隔寄存器 (OTG_FS_HFIR)

OTG_FS Host frame interval register

偏移地址：0x404

复位值：0x0000 EA60

此寄存器用于存储 OTG_FS 控制器对已连接设备当前速度所设定的帧间隔信息。

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

位 31:16 保留，必须保持复位值。

位 15:0 **FRIVL**：帧间隔 (Frame interval)

应用程序在此字段编程的值用于指定两个连续 SOF (FS) 或 Keep-Alive 令牌 (LS) 之间的时间间隔。此字段包含构成所需帧间隔的 PHY 时钟数。只有将主机端口控制和状态寄存器的端口使能位（OTG_FS_HPRT 的 PENA 位）置 1 后，应用程序才能向此寄存器中写入值。如果未对值进行编程，模块将根据在主机配置寄存器的 FS/LS PHY 时钟选择字段（OTG_FS_HCFG 中的 FSLSPCS）中指定的 PHY 时钟来计算。请勿在初始配置后更改此字段的值。

1 ms ×（PHY 时钟频率）

OTG_FS 主机帧编号/帧剩余时间寄存器 (OTG_FS_HFNUM)

OTG_FS Host frame number/frame time remaining register

偏移地址：0x408

复位值：0x0000 3FFF

此寄存器用于指示当前帧编号。它还指示当前帧的剩余时间（以 PHY 时钟数为单位）。

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

位 31:16 **FTREM**：帧剩余时间 (Frame time remaining)

指示当前帧的剩余时间（以 PHY 时钟数为单位）。每过去 1 个 PHY 时钟，此字段递减 1。当值达到零时，此字段将重新装载帧间隔寄存器中的值，并由模块在 USB 上发送一个新 SOF。

位 15:0 **FRNUM**：帧编号 (Frame number)

当在 USB 上发送 1 个新 SOF 时此字段的值将递增 1，当达到 0x3FFF 时会清零。

OTG_FS_Host 周期性发送 FIFO/队列状态寄存器 (OTG_FS_HPTXSTS)

OTG_FS_Host periodic transmit FIFO/queue status register

偏移地址：0x410

复位值：0x0008 0100

此只读寄存器包含周期性 TxFIFO 和周期性发送请求队列的空闲空间信息。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PTXQTOP								PTXQSAV								PTXFSAVL															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:24 **PTXQTOP**：周期性发送请求队列顶部 (Top of the periodic transmit request queue)

指示周期性 Tx 请求队列中 MAC 当前正在处理的项。

该寄存器用于调试。

位 31：奇数/偶数帧 (Odd/Even frame)

- 0：以偶数帧发送
- 1：以奇数帧发送

位 30:27：通道/端点编号 (Channel/endpoint number)

位 26:25：类型 (Type)

- 00：输入/输出
- 01：零长度数据包
- 11：禁止通道命令

位 24：结束（所选通道/端点的最后一个条目）(Terminate (last entry for the selected channel/endpoint))

位 23:16 **PTXQSAV**：周期性发送请求队列可用空间 (Periodic transmit request queue space available)
指示可供写入的周期性发送请求队列的空闲位置的数量。该队列既包含 IN 请求，又包含 OUT 请求。
00：周期性发送请求队列已满
01：1 个位置可用
10：2 个位置可用
bxn：n 个位置可用 (0 ≤ n ≤ 8)
其它值：保留

位 15:0 **PTXFSAVL**：周期性发送数据 FIFO 可用空间 (Periodic transmit data FIFO space available)
指示可供写入的周期性 TxFIFO 的空闲位置的数量。
以 32 位字为单位
0000：周期性 TxFIFO 已满
0001：1 个字可用
0010：2 个字可用
bxn：n 个字可用（其中 0 ≤ n ≤ PTXFD）
其它值：保留

OTG_FS 主机全体通道中断寄存器 (OTG_FS_HAINT)

OTG_FS Host all channels interrupt register

偏移地址：0x414

复位值：0x0000 000

当通道上有事件发生时，主机全体通道中断寄存器会使用模块中断寄存器中的主机通道中断位（OTG_FS_GINTSTS 中的 HCINT 位）中断应用程序。相关内容如*图 365* 所示。每个通道对应 1 个中断位，最多有 16 个位。当应用程序通过相应主机通道 x 中断寄存器清零中断时，该寄存器中的位也会清零。

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

位 31:16 保留，必须保持复位值。

位 15:0 **HAINT**：通道中断 (Channel interrupt)
每个通道对应一位：通道 0 对应位 0，通道 15 对应位 15

OTG_FS 主机全体通道中断屏蔽寄存器 (OTG_FS_HAINTMSK)

OTG_FS Host all channels interrupt mask register

偏移地址：0x418

复位值：0x0000 0000

主机全体通道中断屏蔽寄存器与主机全体通道中断寄存器结合使用，进而在通道上发生事件时中断应用程序。每个通道对应 1 个中断屏蔽位，最多有 16 个位。

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

位 31:16 保留，必须保持复位值。

位 15:0 **HAINTM**：通道中断屏蔽 (Channel interrupt mask)

0：屏蔽中断

1：使能中断

每个通道对应一位：通道 0 对应位 0，通道 15 对应位 15

OTG_FS 主机端口控制和状态寄存器 (OTG_FS_HPRT)

OTG_FS Host port control and status register

偏移地址：0x440

复位值：0x0000 0000

该寄存器仅在主机模式下可用。当前，OTG 主机仅支持一个端口。

该寄存器包含与 USB 端口相关的信息，例如 USB 复位、使能、挂起、恢复、连接状态以及测试模式。具体如*图 365* 中所示。该寄存器中的 rc_w1 位可通过模块中断寄存器中的主机端口中断位（OTG_FS_GINTSTS 中的 HPRTINT 位）触发应用程序中断。发生端口中断时，应用程序必须读取该寄存器，并将引起中断的位清零。对于 rc_w1 位，应用程序必须向该位写入 1 以清除该中断。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved													PSPD		PTCTL				PPWR	PLSTS		Reserved	PRST	PSUSP	PRES	POCCHNG	POCA	PENCHNG	PENA	PCDET	PCSTS
													r	r	rw	rw	rw	rw	rw	r	r		rw	rs	rw	rc_w1	r	rc_w1	rc_w0	rc_w1	r

位 31:19 保留，必须保持复位值。

位 18:17 **PSPD**：端口速度 (Port speed)

指示连接到该端口的设备的速度。

01：全速

10：低速

11：保留

位 16:13 **PTCTL**：端口测试控制 (Port test control)

应用程序向该字段写入一个非零值，以将端口置于测试模式，同时端口上会产生对应模式的信号。

0000：测试模式禁止

0001：Test_J 模式

0010：Test_K 模式

0011：Test_SE0_NAK 模式

0100：Test_Packet 模式

0101：Test_Force_Enable

其它值：保留

位 12 **PPWR**：端口电源 (Port power)

应用程序使用该字段控制该端口的电源，且发生过流情况时，模块会将该位清零。

0：掉电

1：通电

位 11:10 **PLSTS**：端口线状态 (Port line status)

指示 USB 数据线的当前逻辑电平

位 10：OTG_FS_FS_DP 的逻辑电平 (Logic level of OTG_FS_FS_DP)

位 11：OTG_FS_FS_DM 的逻辑电平 (Logic level of OTG_FS_FS_DM)

位 9 保留，必须保持复位值。

位 8 **PRST**：端口复位 (Port reset)

应用程序将该位置 1 时，会在该端口上启动复位序列。应用程序必须为复位周期定时，并在复位序列完成后将该位清零。

0：端口未处于复位状态

1：端口处于复位状态

应用程序必须将该位置 1 并最少保持 10 ms，以在端口上启动复位。除所需的最少持续时间之外，在将该位清零前，应用程序可将该位置 1 的状态再保持 10 ms，即使 USB 标准并没有设置最大限制。

位 7 **PSUSP**：端口挂起 (Port suspend)

应用程序将此位置 1 以将此端口置于挂起模式。只有此位置 1 时，模块才会停止发送 SOF。要停止 PHY 时钟，应用程序必须将端口时钟停止位置 1，这会使能 PHY 的挂起输入引脚。

此位的读取值反映该端口的当前挂起状态。检测到远程唤醒信号，或者应用程序将此寄存器中的端口复位位或端口恢复位置 1 后，模块可将此位清零；或应用程序将模块中断寄存器中的恢复/远程唤醒检测中断位或断开连接检测中断位（分别为 OTG_FS_GINTSTS 中的 WKUINT 或 DISCINT）置 1，模块也可将此位清零。

0：端口未处于挂起模式

1：端口处于挂起模式

位 6 **PRES**：端口恢复 (Port resume)

应用程序将此位置 1 以在该端口上驱动恢复信号。模块会持续驱动恢复信号直到应用程序将此位清零。

如模块中断寄存器中的端口恢复/远程唤醒检测中断位（OTG_FS_GINTSTS 中 WKUINT 位）指示，如果模块检测到 USB 远程唤醒序列，则开始驱动恢复信号，而无需应用程序进行干预；如果模块检测到断开连接的情况，则将此位清零。此位的读取值指示当前模块是否正在驱动恢复信号。

0：不驱动恢复信号

1：驱动恢复信号

位 5 **POCCHNG**：端口过流变化 (Port overcurrent change)

该寄存器中端口过流激活位（位 4）状态发生变化时，模块将此位置 1。

位 4 **POCA**：端口过流激活 (Port overcurrent active)

此位指示端口的过流状况。

0：无过流状况

1：有过流状况

位 3 **PENCHNG**：端口使能/禁止变化 (Port enable/disable change)

该寄存器中的端口使能位 2 的状态发生变化时，模块将此位置 1。

位 2 **PENA**：端口使能 (Port enable)

端口执行复位序列后，只能由模块使能，并且可以由过流状况、断开连接状况或应用程序将此位清零来禁止。应用程序无法通过对寄存器执行写操作将此位置 1。只能将此位清零来禁止端口。对此位的操作不会触发应用程序的任何中断。

0：禁止端口

1：使能端口

位 1 **PCDET**：检测到端口连接 (Port connect detected)

当检测到设备连接时，模块将此位置 1，以使用模块中断寄存器中的主机端口中断位（OTG_FS_GINTSTS 中的 HPRTINT 位）触发应用程序的中断。应用程序必须将此位置 1 才可清除该中断。

位 0 **PCSTS**：端口连接状态 (Port connect status)

0：端口未连接设备

1：端口已连接设备

OTG_FS 主机通道 x 特性寄存器 (OTG_FS_HCCHARx)（x = 0..7，其中 x = 通道编号）

OTG_FS Host channel-x characteristics register

偏移地址：0x500 + (通道编号 × 0x20)

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CHENA	CHDIS	ODDFRM	DAD							MCNT		EPTYP		LSDEV	Reserved	EPDIR	EPNUM				MPSIZ										
rs	rs	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31 **CHENA**：通道使能 (Channel enable)

此字段由应用程序软件置 1，并由 OTG 主机硬件清零。

0：禁止通道 (Channel disabled)

1：使能通道

位 30 **CHDIS**：通道禁止 (Channel disable)

应用程序将此位置 1 以停止通过通道发送/接收数据，即使通过该通道的传输还未完成，停止操作仍然生效。应用程序必须等待禁止通道的中断以确认通道已经被禁止。

位 29 **ODDFRM**：奇数帧 (Odd frame)

此字段由应用程序置位或复位，以分别指示 OTG 主机必须传输奇数帧或偶数帧。此字段只适用于周期性（同步和中断）事务。

0：偶数帧

1：奇数帧 (Odd frame)

位 28:22 **DAD**：设备地址 (Device address)

此字段用于指定要与该主机通信的特定设备。

位 21:20 **MCNT**：多重计数 (Multicount)

此字段向主机指示该周期性端点每帧必须执行的事务数。该字段不用于非周期性传输。

00：保留。对该字段的操作会产生不明确的结果

01：1 个事务

10：该端点每帧需要发出 2 个事务

11：该端点每帧需要发出 3 个事务

注意：此字段至少须置为 01。

位 19:18 **EPTYP**：端点类型 (Endpoint type)

指示选择的传输类型。

00：控制

01：同步

10：批量

11：中断

位 17 **LSDEV**：低速设备 (Low-speed device)

此字段由应用程序置 1，表示此通道正在与一个低速设备进行通信。

位 16 保留，必须保持复位值。

位 15 **EPDIR**：端点方向 (Endpoint direction)

指示通信事务的方向是输入还是输出。

0：输出

1：输入

位 14:11 **EPNUM**：端点编号 (Endpoint number)
指示要与该主机通道通信的 USB 设备的端点号。

位 10:0 **MPSIZ**：最大数据包大小 (Maximum packet size)
指示与该主机通道通信的设备端点的最大数据包大小。

OTG_FS 主机通道 x 中断寄存器 (OTG_FS_HCINTx)（x = 0..7，其中 x = 通道编号）

OTG_FS Host channel-x interrupt register

偏移地址：0x508 + (通道编号 × 0x20)

复位值：0x0000 0000

该寄存器指示在出现 USB 和 AHB 相关事件时通道的状态。具体如*图 365* 中所示。当模块中断寄存器中的主机通道中断位（OTG_FS_GINTSTS 中的 HCINT 位）置 1 时，应用程序必须读取该寄存器。在对寄存器执行读操作之前，应用程序必须先读取主机全体通道中断 (OTG_FS_HAINT) 寄存器，以获取主机通道 x 中断寄存器的准确通道编号。应用程序必须将该寄存器中的相应位清零，才能将 OTG_FS_HAINT 和 OTG_FS_GINTSTS 寄存器中的对应位清零。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																					DTERR	FRMOR	BBERR	TXERR	Reserved	ACK	NAK	STALL	Reserved	CHH	XFRC
																					rc_w1	rc_w1	rc_w1	rc_w1		rc_w1	rc_w1	rc_w1		rc_w1	rc_w1

位 31:11 保留，必须保持复位值。

位 10 **DTERR**：数据同步错误 (Data toggle error)

位 9 **FRMOR**：帧溢出错误 (Frame overrun)

位 8 **BBERR**：串扰错误 (Babble error)

位 7 **TXERR**：通信事务错误 (Transaction error)
指示 USB 上发生下列错误之一：
CRC 校验失败
超时
位填充错误
错误的 EOP

位 6 保留，必须保持复位值。

位 5 **ACK**：收到/发出 ACK 响应 (ACK response received/transmitted interrupt)

位 4 **NAK**：收到 NAK 响应 (NAK response received interrupt)

位 3 **STALL**：收到 STALL 响应 (STALL response received interrupt)

位 2 保留，必须保持复位值。

位 1 **CHH**：通道停止 (Channel halted)
因任意 USB 事务错误或为响应应用程序的禁止请求而导致传输非正常结束。

位 0 **XFRC**：传输完成 (Transfer completed)
未出现任何错误，正常完成传输。

OTG_FS 主机通道 x 中断屏蔽寄存器 (OTG_FS_HCINTMSKx)（x = 0..7，此处 x = 通道编号）

OTG_FS Host channel-x interrupt mask register

偏移地址：0x50C + (通道编号 × 0x20)

复位值：0x0000 0000

此寄存器反映了先前部分中介绍的各通道状态的屏蔽情况。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																					DTERRM	FRMORM	BBERRM	TXERRM	NYET	ACKM	NAKM	STALLM	Reserved	CHHM	XFRCM
																					rw	rw	rw	rw	rw	rw	rw	rw		rw	rw

位 31:11 保留，必须保持复位值。

位 10 **DTERRM**：数据同步错误屏蔽 (Data toggle error mask)
0：屏蔽中断
1：使能中断

位 9 **FRMORM**：帧溢出屏蔽 (Frame overrun mask)
0：屏蔽中断
1：使能中断

位 8 **BBERRM**：串扰错误屏蔽 (Babble error mask)
0：屏蔽中断
1：使能中断

位 7 **TXERRM**：通信事务错误屏蔽 (Transaction error mask)
0：屏蔽中断
1：使能中断

位 6 **NYET**：NYET 响应接收中断屏蔽 (response received interrupt mask)
0：屏蔽中断
1：使能中断

位 5 **ACKM**：ACK 响应接收/发送中断屏蔽 (ACK response received/transmitted interrupt mask)
0：屏蔽中断
1：使能中断

位 4 **NAKM**：NAK 响应接收中断屏蔽 (NAK response received interrupt mask)
0：屏蔽中断
1：使能中断

位 3 **STALLM**：STALL 响应接收中断屏蔽 (STALL response received interrupt mask)
0：屏蔽中断
1：使能中断

位 2 保留，必须保持复位值。

位 1 **CHHM**：通道停止中断屏蔽 (Channel halted mask)

0：屏蔽中断

1：使能中断

位 0 **XFRCM**：传输完成中断屏蔽 (Transfer completed mask)

0：屏蔽中断

1：使能中断

OTG_FS 主机通道 x 传输大小寄存器 (OTG_FS_HCTSIZx)（x = 0..7，此处 x = 通道编号）

OTG_FS Host channel-x transfer size register

偏移地址：0x510 + (通道编号 × 0x20)

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	DPID		PKTCNT										XFRSIZ																		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31 保留，必须保持复位值。

位 30:29 **DPID**：数据 PID (Data PID)

应用程序在此字段设置数据通信的初始同步 PID。主机在此次传输事务过程中保留该字段的设置。

00：DATA0

01：DATA2

10：DATA1

11：MDATA（控制传输）/SETUP（非控制传输）

位 28:19 **PKTCNT**：数据包计数 (Packet count)

应用程序在此字段中设置将要发送 (OUT) 或接收 (IN) 的数据包数。

主机每成功发送或接收一个 OUT/IN 数据包便递减一次计数值。此值达到 0 后，将中断应用程序来指示操作正常完成。

位 18:0 **XFRSIZ**：传输大小 (Transfer size)

对于 OUT 操作，此字段为传输期间主机发送的数据字节数。

对于 IN 操作，此字段为应用程序保留给传输的缓冲区大小。对于 IN 事务（周期性和非周期性），应用程序会将此字段编程为最大数据包大小的整数倍。

30.16.4 设备模式寄存器

Device-mode registers

OTG_FS 设备配置寄存器 (OTG_FS_DCFG)

OTG_FS device configuration register

偏移地址：0x800

复位值：0x0220 0000

此寄存器在上电、执行某些控制命令或枚举后，会将模块配置为设备模式。请勿在初始编程后更改该寄存器。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																			PFIVL		DAD							Reserved	NZLSOHSK		DSPD	
																			rw		rw	rw	rw	rw	rw	rw	rw		rw	rw	rw

位 31:13 保留，必须保持复位值。

位 12:11 **PFIVL**：周期性帧间隔 (Periodic frame interval)

指示一帧内必须使用周期性帧中断通知应用程序的时间点。此功能可用于确定该帧的所有同步通信是否完成。

00：80% 帧间隔

01：85% 帧间隔

10：90% 帧间隔

11：95% 帧间隔

位 10:4 **DAD**：设备地址 (Device address)

应用程序必须在执行每个 SetAddress 控制命令后根据命令参数对该字段进行设置。

位 3 保留，必须保持复位值。

位 2 **NZLSOHSK**：非零长度状态 OUT 握手信号 (Non-zero-length status OUT handshake)

在控制传输状态阶段的 OUT 事务期间，当模块收到非零长度数据包后，应用程序可以使用此字段选择要发送的握手信号。

1：收到非零长度状态 OUT 事务时，回复 STALL 握手信号，收到的 OUT 数据包不发送给应用程序。

0：将收到的 OUT 数据包（零长度或非零长度）发送给应用程序，并基于设备端点控制寄存器中端点的 NAK 和 STALL 位回复握手信号。

位 1:0 **DSPD**：设备速度 (Device speed)

指示应用程序要求模块进行枚举所采用的速度，或应用程序支持的最大速度。但是，实际总线速度只有在完成 chirp 序列后才能确定，同时此速度基于与模块连接的 USB 主机的速度。

00：保留

01：保留

10：保留

11：全速（USB 1.1 收发器时钟为 48 MHz）

OTG_FS 设备控制寄存器 (OTG_FS_DCTL)

OTG_FS device control register

偏移地址：0x804

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																				POPRGDNE	CGONAK	SGONAK	CGINAK	SGINAK	TCTL			GONSTS	GINSTS	SDIS	RWUSIG
																				rw	w	w	w	w	rw	rw	rw	r	r	rw	rw

位 31:12 保留，必须保持复位值。

位 11 **POPRGDNE**：上电编程完成 (Power-on programming done)
应用程序使用此位指示寄存器从掉电模式唤醒后已完成编程。

位 10 **CGONAK**：将全局 OUT NAK 清零 (Clear global OUT NAK)
对此位执行写操作会将全局 OUT NAK 清零。

位 9 **SGONAK**：将全局 OUT NAK 置 1 (Set global OUT NAK)
对此位执行写操作会将全局 OUT NAK 置 1。
应用程序使用此位在所有 OUT 端点发送 NAK 握手信号。
应用程序只有确定模块中断寄存器中全局 OUT NAK 有效位（OTG_FS_GINTSTS 中 GONAKEFF 位）已清零时，才可以将此位置 1。

位 8 **CGINAK**：将全局 IN NAK 清零 (Clear global IN NAK)
对此位执行写操作会将全局 IN NAK 清零。

位 7 **SGINAK**：将全局 IN NAK 置 1 (Set global IN NAK)
对此字段执行写操作会将全局非周期性 IN NAK 置 1。应用程序使用此位使所有非周期性 IN 端点发送 NAK 握手信号。
应用程序只有确定模块中断寄存器中全局 IN NAK 有效位（OTG_FS_GINTSTS 中 GINAKEFF 位）已清零时，才可以将此位置 1。

位 6:4 **TCTL**：测试控制 (Test control)
000：测试模式禁止
001：Test_J 模式
010：Test_K 模式
011：Test_SE0_NAK 模式
100：Test_Packet 模式
101：Test_Force_Enable
其它值：保留

位 3 **GONSTS**：全局 OUT NAK 状态 (Global OUT NAK status)
0：将根据 FIFO 状态和 NAK 和 STALL 位设置发送握手信号。
1：无论 RxFIFO 中是否还有空闲空间都不接收数据。除 SETUP 事务之外，对所有收到的数据包回复 NAK 握手信号。所有同步类型的 OUT 数据包都将被丢弃。

位 2 **GINSTS**：全局 IN NAK 状态 (Global IN NAK status)
0：将根据发送 FIFO 中的数据可用性回复握手信号。
1：使所有非周期性 IN 端点回复 NAK 握手信号，无需考虑发送 FIFO 中的数据可用性。

位 1 **SDIS**：软断开 (Soft disconnect)

应用程序使用该位向 USB OTG 模块发出执行软断开的信号。该位置 1 时，主机不会看到设备已连接，且该设备也不会接收 USB 上的信号。在应用程序将此位清零之前，模块会保持断开状态。

0：正常工作。此位在软断开之后清零，会使主机收到设备已连接的事件。重新连接设备之后，USB 主机会重新启动设备枚举。

1：使主机收到设备断开连接的事件。

位 0 **RWUSIG**：发送远程唤醒信号 (Remote wakeup signaling)

应用程序将此位置 1 时，模块会启动远程发送信号，以唤醒 USB 主机。应用程序必须将此位置 1 以使模块退出挂起状态。根据 USB 2.0 规范，应用程序必须在将此位置 1 之后的 1 ms 到 15 ms 内将其清零。

表 176 显示了为使 USB 主机检测到设备断开连接所需的软断开 (SDIS) 位置 1 的最短时间（取决于设备状态）。为了协调时钟抖动，建议应用程序在指定的最小时间基础上再加入一段延迟。

表 176. 软断开的最小时间

运行速度	设备状态	最小时间
全速	挂起	1 ms + 2.5 µs
全速	空闲	2.5 µs
全速	非空闲或挂起（正在通信时）	2.5 µs

OTG_FS 设备状态寄存器 (OTG_FS_DSTS)

OTG_FS device status register

偏移地址：0x808

复位值：0x0000 0010

此寄存器指示模块在出现 USB 相关事件时的状态。发生中断时，必须从设备全体中断 (OTG_FS_DAINT) 寄存器读取发生中断的端点信息。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved										FNSOF														Reserved				EERR	ENUMSPD		SUSPSTS
										r	r	r	r	r	r	r	r	r	r	r	r	r	r					r	r	r	r

位 31:22 保留，必须保持复位值。

位 21:8 **FNSOF**：接收 SOF 的帧编号 (Frame number of the received SOF)

位 7:4 保留，必须保持复位值。

位 3 **EERR**：不定错误 (Erratic error)

模块将该位置 1 以报告任何不定错误。

由于不定错误，OTG_FS 控制器会进入挂起状态，并且会以 OTG_FS_GINTSTS 寄存器的早期挂起位（OTG_FS_GINTSTS 中的 ESUSP 位）生成一个中断。如果早期挂起中断是由不定错误触发，则应用程序只能执行软断开以恢复通信。

位 2:1 **ENUMSPD**：枚举速度 (Enumerated speed)

指示 OTG_FS 控制器通过 chirp 序列检测速度后被枚举成的速度。

01：保留

10：保留

11：全速（PHY 时钟运行频率为 48 MHz）

其它值：保留

位 0 **SUSPSTS**：挂起状态 (Suspend status)

在设备模式下，只要在 USB 上检测到挂起状态，该位就会置 1。当 USB 总线上的空闲状态保持 3ms，模块便会进入挂起状态。出现以下情况时，模块会退出挂起状态：

– USB 数据线上有活动

– 应用程序对 OTG_FS_DCTL 寄存器的远程唤醒信号位（OTG_FS_DCTL 中的 RWUSIG 位）执行写操作。

OTG_FS 设备 IN 端点通用中断屏蔽寄存器 (OTG_FS_DIEPMSK)

OTG_FS device IN endpoint common interrupt mask register

偏移地址：0x810

复位值：0x0000 0000

此寄存器与全体端点的各个 OTG_FS_DIEPINTx 寄存器配合使用，以便在每个 IN 端点上生成中断。通过对此寄存器的相应位执行写操作，可屏蔽 OTG_FS_DIEPINTx 寄存器中的 IN 端点中断。默认情况下，状态中断都被屏蔽。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																									INEPNEM	INEPNMM	ITTXFEMSK	TOM	Reserved	EPDM	XFRCM
																									rw	rw	rw	rw		rw	rw

位 31:7 保留，必须保持复位值。

位 6 **INEPNEM**：IN 端点 NAK 有效中断屏蔽 (IN endpoint NAK effective mask)

0：屏蔽中断

1：使能中断

位 5 **INEPNMM**：EP 不匹配时接收到 IN 令牌中断屏蔽 (IN token received with EP mismatch mask)

0：屏蔽中断

1：使能中断

位 4 **ITTXFEMSK**：TxFIFO 为空时接收到 IN 令牌中断屏蔽 (IN token received when TxFIFO empty mask)

0：屏蔽中断

1：使能中断

位 3 **TOM**：超时中断屏蔽（非同步端点）(Timeout condition mask (Non-isochronous endpoints))

0：屏蔽中断

1：使能中断

位 2 保留，必须保持复位值。

位 1 **EPDM**：端点禁止中断屏蔽 (Endpoint disabled interrupt mask)
0：屏蔽中断
1：使能中断

位 0 **XFRCM**：传输完成中断屏蔽 (Transfer completed interrupt mask)
0：屏蔽中断
1：使能中断

OTG_FS 设备 OUT 端点通用中断屏蔽寄存器 (OTG_FS_DOEPMSK)

OTG_FS device OUT endpoint common interrupt mask register

偏移地址：0x814

复位值：0x0000 0000

此寄存器与所有端点的各个 OTG_FS_DOEPINTx 寄存器配合使用，以便可以在每个 OUT 端点上生成中断。通过对此寄存器的相应位执行写操作，可屏蔽 OTG_FS_DOEPINTx 寄存器中的 OUT 端点中断。默认情况下，状态中断都被屏蔽。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																											OTEPDM	STUPM	Reserved	EPDM	XFRCM
																											rw	rw		rw	rw

位 31:5 保留，必须保持复位值。

位 4 **OTEPDM**：端点禁止时接收到 OUT 令牌中断屏蔽 (OUT token received when endpoint disabled mask)
仅适用于控制 OUT 端点。
0：屏蔽中断
1：使能中断

位 3 **STUPM**：SETUP 阶段完成中断屏蔽 (SETUP phase done mask)
仅适用于控制端点。
0：屏蔽中断
1：使能中断

位 2 保留，必须保持复位值。

位 1 **EPDM**：端点禁止中断屏蔽 (Endpoint disabled interrupt mask)
0：屏蔽中断
1：使能中断

位 0 **XFRCM**：传输完成中断屏蔽 (Transfer completed interrupt mask)
0：屏蔽中断
1：使能中断

OTG_FS 设备全体端点中断寄存器 (OTG_FS_DAINT)

OTG_FS device all endpoints interrupt register

偏移地址：0x818

复位值：0x0000 0000

当端点上发生有效事件时，OTG_FS_DAINT 寄存器将通过 OTG_FS_GINTSTS 寄存器中的设备 OUT 端点中断位或设备 IN 端点中断位（分别为 OTG_FS_GINTSTS 中的 OEPINT 或 IEPINT 位）来中断应用程序。每个端点对应一个中断位，OUT 端点和 IN 端点均最多有 16 个中断位。双向端点将使用相应的 IN 和 OUT 中断位。当应用程序将相应设备端点 x 中断寄存器 (OTG_FS_DIEPINTx/OTG_FS_DOEPINTx) 中的位置 1 和清零时，此寄存器中的相应位也将置 1 和清零。

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

位 31:16 **OEPINT**：OUT 端点中断位 (OUT endpoint interrupt bits)
每个 OUT 端点对应一位：
OUT 端点 0 对应位 16，而 OUT 端点 3 对应位 18。

位 15:0 **IEPINT**：IN 端点中断位 (IN endpoint interrupt bits)
每个 IN 端点对应一位：
IN 端点 0 对应位 0，而 IN 端点 3 对应位 3。

OTG_FS 全体端点中断屏蔽寄存器 (OTG_FS_DAINTMSK)

OTG_FS all endpoints interrupt mask register

偏移地址：0x81C

复位值：0x0000 0000

OTG_FS_DAINTMSK 寄存器与设备端点中断寄存器结合使用，在设备端点上发生事件时中断应用程序。但是，与该中断相对应的 OTG_FS_DAINT 寄存器位仍会置 1。

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

位 31:16 **OEPM**：OUT EP 中断屏蔽位 (OUT EP interrupt mask bits)
每个 OUT 端点对应一位：
OUT EP 0 对应位 16，而 OUT EP 3 对应位 18
0：屏蔽中断
1：使能中断

位 15:0 **IEPM**：IN EP 中断屏蔽位 (IN EP interrupt mask bits)
每个 IN 端点对应一位：
IN EP 0 对应位 0，而 IN EP 3 对应位 3
0：屏蔽中断
1：使能中断

OTG_FS 设备 V_{BUS} 放电时间寄存器 (OTG_FS_DVBUSDIS)

OTG_FS device V_{BUS} discharge time register

偏移地址：0x0828

复位值：0x0000 17D7

该寄存器指定 SRP 期间 V_{BUS} 发出脉冲之后的 V_{BUS} 放电时间。

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

位 31:16 保留，必须保持复位值。

位 15:0 **VBUSDT**：设备 V_{BUS} 放电时间 (Device V_{BUS} discharge time)
指定 SRP 期间 V_{BUS} 发出脉冲之后的 V_{BUS} 放电时间。该时间值等于：
V_{BUS} 放电时间（PHY 时钟数）/1024
该值可基于不同的 V_{BUS} 负载根据需要进行调整。

OTG_FS 设备 V_{BUS} 脉冲时间寄存器 (OTG_FS_DVBUSPULSE)

OTG_FS device V_{BUS} pulsing time register

偏移地址：0x082C

复位值：0x0000 05B8

该寄存器指定 SRP 期间的 V_{BUS} 脉冲时间。

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

位 31:12 保留，必须保持复位值。

位 11:0 **DVBUSP**：设备 V_{BUS} 脉冲时间 (Device V_{BUS} pulsing time)
指定 SRP 期间的 V_{BUS} 脉冲时间。该时间值等于：
V_{BUS} 脉冲时间（PHY 时钟数）/1024

OTG_FS 设备 IN 端点 FIFO 空中断屏蔽寄存器：OTG_FS_DIEPEMPMSK)

OTG_FS device IN endpoint FIFO empty interrupt mask register

偏移地址：0x834

复位值：0x0000 0000

此寄存器用于控制 IN 端点 FIFO 空中断的生成 (TXFE_OTG_FS_DIEPINTx)。

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

位 31:16 保留，必须保持复位值。

位 15:0 **INEPTXFEM**：IN EP Tx FIFO 空中断屏蔽位 (IN EP Tx FIFO empty interrupt mask bits)
这些位用作 OTG_FS_DIEPINTx 的屏蔽位。
每个位对应一个 IN 端点的 TXFE 中断：
IN 端点 0 对应位 0，而 IN 端点 3 对应位 3
0：屏蔽中断
1：使能中断

OTG_FS 设备控制 IN 端点 0 控制寄存器 (OTG_FS_DIEPCTL0)

OTG_FS device control IN endpoint 0 control register

偏移地址：0x900

复位值：0x0000 0000

本节介绍 OTG_FS_DIEPCTL0 寄存器。非零控制端点使用编号 1-3 的寄存器。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EPENA	EPDIS	Reserved		SNAK	CNAK	TXFNUM				STALL	Reserved	EPTYP		NAKSTS	Reserved	USBAEP	Reserved													MPSIZ	
r	r			w	w	rw	rw	rw	rw	rs		r	r	r		r														rw	rw

位 31 **EPENA**：端点使能 (Endpoint enable)
应用程序将此位置 1 以在端点 0 上启动数据发送。
在此端点上触发以下任一中断之前，模块会将此位清零：
– 端点禁止
– 传输完成

位 30 **EPDIS**：端点禁止 (Endpoint disable)
即使在该端点上的传输完成之前，应用程序也可将此位置 1，以停止端点上的数据发送。应用程序必须等到发生端点禁止中断后，才能将端点视为禁止端点。在端点禁止中断位置 1 前，模块会将此位清零。只有在该端点的端点使能位置 1 后，应用程序才可将该位置 1。

位 29:28 保留，必须保持复位值。

位 27 **SNAK**：将 NAK 置 1 (Set NAK)

对此位进行写操作会将端点的 NAK 位置 1。

通过此位，应用程序可以控制端点上 NAK 握手信号的发送。模块也可在端点接收到 SETUP 数据包后将该端点的此位置 1。

位 26 **CNAK**：将 NAK 清零 (Clear NAK)

对此位进行写操作会将端点的 NAK 位清零。

位 25:22 **TXFNUM**：TxFIFO 编号 (TxFIFO number)

该值设置为分配给 IN 端点 0 的 FIFO 编号。

位 21 **STALL**：STALL 握手 (STALL handshake)

应用程序只能将此位置 1，端点接收到 SETUP 令牌时，模块会将此位清零。如果 NAK 位、全局 IN NAK 或全局 OUT NAK 与此位均置 1，则 STALL 位优先。

位 20 保留，必须保持复位值。

位 19:18 **EPTYP**：端点类型 (Endpoint type)

硬件设置为‘00’，表示控制类型的端点

位 17 **NAKSTS**：NAK 状态 (NAK status)

指示以下结果：

0：模块根据 FIFO 状态回复非 NAK 握手。

1：模块在此端点上回复 NAK 握手。

当此位置 1 时（无论是被应用程序还是被模块），即使 TxFIFO 中仍有数据可用，模块也会停止发送数据。无论此位如何设置，模块总是通过 ACK 握手响应 SETUP 数据包。

位 16 保留，必须保持复位值。

位 15 **USBAEP**：USB 活动端点 (USB active endpoint)

此位总是置 1，指示在所有配置和接口中控制端点 0 始终处于激活状态。

位 14:2 保留，必须保持复位值。

位 1:0 **MPSIZ**：最大数据包大小 (Maximum packet size)

应用程序必须将此字段编程为当前逻辑端点的最大数据包大小。

00：64 字节

01：32 字节

10：16 字节

11：8 字节

OTG 设备端点 x 控制寄存器 (OTG_FS_DIEPCTLx)（x = 1..3，其中 x = 端点编号）

OTG device endpoint-x control register

偏移地址：0x900 + (端点编号 × 0x20)

复位值：0x0000 0000

应用程序使用此寄存器控制各个逻辑端点（端点 0 除外）的行为。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EPENA	EPDIS	SODDFRM	SD0PID/SEVNFRM	SNAK	CNAK	TXFNUM				Stall	Reserved	EPTYP		NAKSTS	EONUM/DPID	USBAEP	Reserved				MPSIZ											
rs	rs	w	w	w	w	rw	rw	rw	rw	rw/ rs		rw	rw	r	r	rw					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31 **EPENA**：端点使能 (Endpoint enable)

应用程序将此位置 1 以在端点上启动数据发送。

在此端点上触发以下任一中断之前，模块会将此位清零：

- SETUP 阶段完成
- 端点禁止
- 传输完成

位 30 **EPDIS**：端点禁止 (Endpoint disable)

即使在该端点上的传送完成之前，应用程序也可将此位置 1，以停止端点上的数据发送/接收。应用程序必须等到发生端点禁止中断后，才能将端点视为禁止端点。在端点禁止中断位置 1 前，模块会将此位清零。只有在该端点的端点使能位置 1 后，应用程序才可将该位置 1。

位 29 **SODDFRM**：设置奇数帧 (Set odd frame)

仅适用于同步 IN 和 OUT 端点。

对此字段进行写操作会将偶数/奇数帧 (EONUM) 字段设置为奇数帧。

位 28 **SD0PID**：设置 DATA0 PID (Set DATA0 PID)

仅适用于中断/批量 IN 端点。

对此字段进行写操作会将此寄存器中的端点数据 PID (DPID) 字段设置为 DATA0。

SEVNFRM：设置偶数帧 (Set even frame)

仅适用于同步 IN 端点。

对此字段进行写操作会将偶数/奇数帧 (EONUM) 字段设置为偶数帧。

位 27 **SNAK**：将 NAK 置 1 (Set NAK)

对此位进行写操作会将端点的 NAK 位置 1。

通过此位，应用程序可以控制端点上 NAK 握手信号的发送。发生传输完成中断时或端点上接收到 SETUP 后，模块也可以将 OUT 端点的这个位置 1。

位 26 **CNAK**：将 NAK 清零 (Clear NAK)

对此位进行写操作会将端点的 NAK 位清零。

位 25:22 **TXFNUM**：TxFIFO 编号 (TxFIFO number)

这些位用于指定与此端点相关联的 FIFO 编号。必须为每个有效的 IN 端点设置单独的 FIFO 编号。

此字段仅针对 IN 端点有效。

位 21 **STALL**：STALL 握手 (STALL handshake)

仅适用于非控制、非同步 IN 端点（访问类型为 rw）。

应用程序将此位置 1 使得设备对来自 USB 主机的所有令牌都回复 STALL。如果 NAK 位、全局 IN NAK 或全局 OUT NAK 与此位同时置 1，则 STALL 位优先。只有应用程序能够将此位清零，而模块则不能。

仅适用于控制端点（访问类型为 rs）

此端点接收到 SETUP 令牌时，应用程序只能将此位置 1，而模块会将其清零。如果 NAK 位、全局 IN NAK 或全局 OUT NAK 与此位同时置 1，则 STALL 位优先。无论此位如何设置，模块总是通过 ACK 握手响应 SETUP 数据包。

位 20 保留，必须保持复位值。

位 19:18 **EPTYP**：端点类型 (Endpoint type)

以下是这个逻辑端点支持的传输类型。

00：控制
01：同步
10：批量
11：中断

位 17 **NAKSTS**：NAK 状态 (NAK status)

它指示以下结果：

0：模块根据 FIFO 状态回复非 NAK 握手。

1：模块在此端点上回复 NAK 握手。

当应用程序或模块将此位置 1 时：

对于非同步 IN 端点：即使 TxFIFO 中存在可用数据，模块也会停止通过 IN 端点发送任何数据。

对于同步 IN 端点：即使 TxFIFO 中存在可用数据，模块也会发送长度为零的数据包。

无论此位如何设置，模块总是通过 ACK 握手响应 SETUP 数据包。

位 16 **EONUM**：偶数/奇数帧 (Even/odd frame)

仅适用于同步 IN 端点。

指示模块为此端点发送/接收同步的数据所在的帧的编号。应用程序必须通过此寄存器中的 SEVNFRM 和 SODDFRM 字段对偶数/奇数帧编号进行编程，以便此端点发送/接收同步数据。

0：偶数帧

1：奇数帧

DPID：端点数据 PID (Endpoint data PID)

仅适用于中断/批量 IN 端点。

包含此端点上将要接收或发送的数据包的 PID。端点激活后，应用程序必须对要在此端点上接收或发送的首个数据包的 PID 进行编程。应用程序使用 SD0PID 寄存器字段对 DATA0 或 DATA1 PID 进行编程。

0：DATA0

1：DATA1

位 15 **USBAEP**：USB 活动端点 (USB active endpoint)

指示此端点在当前配置和接口中是否激活。检测到 USB 复位后，模块会为所有端点（端点 0 除外）将此位清零。接收到 SetConfiguration 和 SetInterface 命令后，应用程序必须相应地对端点寄存器进行编程并将此位置 1。

位 14:11 保留，必须保持复位值。

位 10:0 **MPSIZ**：最大数据包大小 (Maximum packet size)

应用程序必须将此字段编程为当前逻辑端点的最大数据包大小。此值以字节为单位。

OTG_FS 设备控制 OUT 端点 0 控制寄存器 (OTG_FS_DOEPCTL0)

OTG_FS device control OUT endpoint 0 control register

偏移地址：0xB00

复位值：0x0000 8000

本节介绍 OTG_FS_DOEPCTL0 寄存器。非零控制端点使用编号 1-3 的寄存器。

31	30	29 28	27	26	25 24 23 22	21	20	19 18	17	16	15	14 13 12 11 10 9 8 7 6 5 4 3 2	1 0
EPENA	EPDIS	Reserved	SNAK	CNAK	Reserved	Stall	SNPM	EPTYP	NAKSTS	Reserved	USBAEP	Reserved	MPSIZ
w	r		w	w		rs	rw	r r	r		r		r r

位 31 **EPENA**：端点使能 (Endpoint enable)

应用程序将此位置 1 以在端点 0 上启动数据发送。

在此端点上触发以下任一中断之前，模块会将此位清零：

- SETUP 阶段完成
- 端点禁止
- 传输完成

位 30 **EPDIS**：端点禁止 (Endpoint disable)

应用程序无法禁止控制 OUT 端点 0。

位 29:28 保留，必须保持复位值。

位 27 **SNAK**：将 NAK 置 1 (Set NAK)

对此位进行写操作会将端点的 NAK 位置 1。

通过此位，应用程序可以控制端点上 NAK 握手信号的发送。发生传输完成中断时或端点上接收到 SETUP 后，模块也可以将此位置 1。

位 26 **CNAK**：将 NAK 清零 (Clear NAK)

对此位进行写操作会将端点的 NAK 位清零。

位 25:22 保留，必须保持复位值。

位 21 **STALL**：STALL 握手 (STALL handshake)

此端点接收到 SETUP 令牌时，应用程序只能将此位置 1，而模块会将其清零。如果 NAK 位、全局 OUT NAK 与此位同时置 1，则 STALL 位优先。无论此位如何设置，模块总是通过 ACK 握手响应 SETUP 数据包。

位 20 **SNPM**：监听模式 (Snoop mode)

此位用于将端点配置为监听模式。在监听模式下，模块不会在将 OUT 数据包传输到应用存储区前检查其是否正确。

位 19:18 **EPTYP**：端点类型 (Endpoint type)

硬件固定为二进制 00，表示端点为控制传输类型。

位 17 **NAKSTS**：NAK 状态 (NAK status)

指示以下结果：

0：模块根据 FIFO 状态回复非 NAK 握手。

1：模块在此端点上回复 NAK 握手。

当应用程序或模块将此位置 1 时，即使 RxFIFO 中存在空间可继续容纳收到的数据包，模块也会停止接收数据。无论此位如何设置，模块总是通过 ACK 握手响应 SETUP 数据包。

位 16 保留，必须保持复位值。

位 15 **USBAEP**：USB 活动端点 (USB active endpoint)

此位总是置 1，指示在所有配置和接口中控制端点 0 始终处于激活状态。

位 14:2 保留，必须保持复位值。

位 1:0 **MPSIZ**：最大数据包大小 (Maximum packet size)

控制 OUT 端点 0 的最大数据包大小与在控制 IN 端点 0 中进行编程的值相同。

00：64 字节

01：32 字节

10：16 字节

11：8 字节

OTG_FS 设备端点 x 控制寄存器 (OTG_FS_DOEPCTLx)（x = 1..3，其中 x = 端点编号）

OTG_FS device endpoint-x control register

OUT 端点的偏移地址：0xB00 + (端点编号 × 0x20)

复位值：0x0000 0000

应用程序使用此寄存器控制各个逻辑端点（端点 0 除外）的行为。

31	30	29	28	27	26	25 24 23 22	21	20	19 18	17	16	15	14 13 12 11	10 9 8 7 6 5 4 3 2 1 0
EPENA	EPDIS	SODDFRM/SD1PID	SD0PID/SEVNFRM	SNAK	CNAK	Reserved	Stall	SNPM	EPTYP	NAKSTS	EONUM/DPID	USBAEP	Reserved	MPSIZ
rs	rs	w	w	w	w		rw/ rs	rw	rw	r	r	rw		rw rw rw rw rw rw rw rw rw rw rw

位 31 **EPENA**：端点使能 (Endpoint enable)

适用于 IN 和 OUT 端点。

应用程序将此位置 1 以在端点上启动数据发送。

在此端点上触发以下任一中断之前，模块会将此位清零：

- SETUP 阶段完成
- 端点禁止
- 传输完成

位 30 **EPDIS**：端点禁止 (Endpoint disable)

即使在该端点上的传送完成之前，应用程序也可将此位置 1，以停止端点上的数据发送/接收。应用程序必须等到发生端点禁止中断后，才能将端点视为禁止端点。在端点禁止中断位置 1 前，模块会将此位清零。只有在该端点的端点使能位置 1 后，应用程序才可将该位置 1。

位 29 **SD1PID**：设置 DATA1 PID (Set DATA1 PID)

仅适用于中断/批量 IN 和 OUT 端点。对此字段进行写操作会将此寄存器中的端点数据 PID (DPID) 字段设置为 DATA1。

SODDFRM：设置奇数帧 (Set odd frame)

仅适用于同步 IN 和 OUT 端点。对此字段进行写操作会将偶数/奇数帧 (EONUM) 字段设置为奇数帧。

位 28 **SD0PID**：设置 DATA0 PID (Set DATA0 PID)

仅适用于中断/批量 OUT 端点。

对此字段进行写操作会将此寄存器中的端点数据 PID (DPID) 字段设置为 DATA0。

SEVNFRM：设置偶数帧 (Set even frame)

仅适用于同步 OUT 端点。

对此字段进行写操作会将偶数/奇数帧 (EONUM) 字段设置为偶数帧。

位 27 **SNAK**：将 NAK 置 1 (Set NAK)

对此位进行写操作会将端点的 NAK 位置 1。

通过此位，应用程序可以控制端点上 NAK 握手信号的发送。发生传输完成中断时或端点上接收到 SETUP 后，模块也可以将 OUT 端点的这个位置 1。

位 26 **CNAK**：将 NAK 清零 (Clear NAK)

对此位进行写操作会将端点的 NAK 位清零。

位 25:22 保留，必须保持复位值。

位 21 **STALL**：STALL 握手 (STALL handshake)

仅适用于非控制、非同步 OUT 端点（访问类型为 rw）。

应用程序将此位置 1 使得设备对来自 USB 主机的所有令牌都回复 STALL。如果 NAK 位、全局 IN NAK 或全局 OUT NAK 与此位同时置 1，则 STALL 位优先。只有应用程序能够将此位清零，而模块则不能。

仅适用于控制端点（访问类型为 rs）

此端点接收到 SETUP 令牌时，应用程序只能将此位置 1，而模块会将其清零。如果 NAK 位、全局 IN NAK 或全局 OUT NAK 与此位同时置 1，则 STALL 位优先。无论此位如何设置，模块总是通过 ACK 握手响应 SETUP 数据包。

位 20 **SNPM**：监听模式 (Snoop mode)

此位用于将端点配置为监听模式。在监听模式下，模块不会在将 OUT 数据包传输到应用存储区前检查其是否正确。

位 19:18 **EPTYP**：端点类型 (Endpoint type)

以下是这个逻辑端点支持的传输类型。

00：控制

01：同步

10：批量

11：中断

位 17 **NAKSTS**：NAK 状态 (NAK status)

指示以下结果：

0：模块根据 FIFO 状态回复非 NAK 握手。

1：模块在此端点上回复 NAK 握手。

当应用程序或模块将此位置 1 时：

即使 RxFIFO 存在空间可容纳传入数据包，模块也会停止在 OUT 端点上接收任何数据。

无论此位如何设置，模块总是通过 ACK 握手响应 SETUP 数据包。

位 16 **EONUM**：偶数/奇数帧 (Even/odd frame)

仅适用于同步 IN 和 OUT 端点。

指示模块为此端点发送/接收同步的数据所在的帧的编号。应用程序必须通过此寄存器中的 SEVNFRM 和 SODDFRM 字段对偶数/奇数帧编号进行编程，以便此端点发送/接收同步数据。

0：偶数帧

1：奇数帧

DPID：端点数据 PID (Endpoint data PID)

仅适用于中断/批量 OUT 端点。

包含此端点上将要接收或发送的数据包的 PID。端点激活后，应用程序必须对要在此端点上接收或发送的首个数据包的 PID 进行编程。应用程序使用 SD0PID 寄存器字段对 DATA0 或 DATA1 PID 进行编程。

0：DATA0

1：DATA1

位 15 **USBAEP**：USB 活动端点 (USB active endpoint)

指示此端点在当前配置和接口中是否激活。检测到 USB 复位后，模块会为所有端点（端点 0 除外）将此位清零。接收到 SetConfiguration 和 SetInterface 命令后，应用程序必须相应地对端点寄存器进行编程并将此位置 1。

位 14:11 保留，必须保持复位值。

位 10:0 **MPSIZ**：最大数据包大小 (Maximum packet size)

应用程序必须将此字段编程为当前逻辑端点的最大数据包大小。此值以字节为单位。

OTG_FS 设备端点 x 中断寄存器 (OTG_FS_DIEPINTx)（x = 0..3，其中 x = 端点编号）

OTG_FS device endpoint-x interrupt register

偏移地址：0x908 + (端点编号 × 0x20)

复位值：0x0000 0080

此寄存器指示端点在出现 USB 和 AHB 相关事件时的状态。具体如*图 365* 中所示。当模块中断寄存器中的 IN 端点中断位（OTG_FS_GINTSTS 中的 IEPINT 位）置 1 时，应用程序必须读取此寄存器。在应用程序能够读取此寄存器之前，必须先读取设备全体端点中断 (OTG_FS_DAINT) 寄存器，以获取设备端点 x 中断寄存器的准确端点编号。应用程序必须将此寄存器中的相应位清零，才能将 OTG_FS_DAINT 和 OTG_FS_GINTSTS 寄存器中的对应位清零。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																								TXFE	INEPNE	Reserved	ITTXFE	TOC	Reserved	EPDISD	XFRC
																								r	rc_w1/rw		rc_w1	rc_w1		rc_w1	rc_w1

位 31:8 保留，必须保持复位值。

位 7 **TXFE**：发送 FIFO 为空 (Transmit FIFO empty)

当此端点的 TxFIFO 为半空或全空时，此中断被置位。TxFIFO 为半空还是全空状态由 OTG_FS_GAHBCFG 寄存器中的 TxFIFO 空白级别位（OTG_FS_GAHBCFG 中的 TXFELVL 位）决定。

位 6 **INEPNE**：IN 端点 NAK 有效 (IN endpoint NAK effective)

当应用程序通过向 OTG_FS_DIEPCTLx 中的 CNAK 位写入数据来将 IN 端点 NAK 清零时，此位可被清零。

该中断指示模块已对（由应用程序或模块）置 1 的 NAK 采样，结果已生效。该中断指示由应用程序置 1 的 IN 端点 NAK 位已在模块中起作用。

此中断不保证在 USB 上发送了 NAK 握手信号。STALL 位的优先级高于 NAK 位。

位 5 保留，必须保持复位值。

位 4 **ITTXFE**：TxFIFO 为空时接收到 IN 令牌 (IN token received when TxFIFO is empty)

仅适用于非周期性 IN 端点。

当和该端点相对应的 TxFIFO（周期性/非周期性）为空时，接收到 IN 令牌，从而产生中断。

位 3 **TOC**：超时条件 (Timeout condition)

仅适用于控制 IN 端点。

指示该端点对最近收到的 IN 令牌响应超时。

位 2 保留，必须保持复位值。

位 1 **EPDISD**：端点禁止中断 (Endpoint disabled interrupt)

此位指示该端点已经由应用程序禁止掉。

位 0 **XFRC**：传输完成中断 (Transfer completed interrupt)

此字段指示在此端点上设置的传输已经在 USB 和 AHB 上传输完成。

OTG_FS 设备端点 x 中断寄存器 (OTG_FS_DOEPINTx)（x = 0..3，其中 x = 端点编号）

OTG_FS device endpoint-x interrupt register

偏移地址：0xB08 + (端点编号 × 0x20)

复位值：0x0000 0080

此寄存器指示端点在出现 USB 和 AHB 相关事件时的状态。具体如*图 365* 中所示。当 OTG_FS_GINTSTS 寄存器中的 OUT 端点中断位（OTG_FS_GINTSTS 中的 OEPINT 位）置 1 时，应用程序必须读取此寄存器。在应用程序能够读取此寄存器之前，必须先读取 OTG_FS_DAINT 寄存器，以获取 OTG_FS_DOEPINTx 寄存器的准确端点编号。应用程序必须将此寄存器中的相应位清零，才能将 OTG_FS_DAINT 和 OTG_FS_GINTSTS 寄存器中的对应位清零。

31:8	7	6	5	4	3	2	1	0
Reserved	Reserved	B2BSTUP	Reserved	OTEPDIS	STUP	Reserved	EPDISD	XFRC
		rc_w1/rw		rc_w1	rc_w1		rc_w1	rc_w1

位 31:7 保留，必须保持复位值。

位 6 **B2BSTUP**：接收到连续的 SETUP 数据包 (Back-to-back SETUP packets received)

仅适用于控制 OUT 端点。

此位指示该端点已接收到三个以上的连续 SETUP 数据包。

位 5 保留，必须保持复位值。

位 4 **OTEPDIS**：端点禁止时接收到 OUT 令牌 (OUT token received when endpoint disabled)
仅适用于控制 OUT 端点。
指示在尚未使能端点时接收到 OUT 令牌，从而产生中断。

位 3 **STUP**：SETUP 阶段完成 (SETUP phase done)
仅适用于控制 OUT 端点。
指示控制端点的 SETUP 阶段已完成，当前控制传输中不再接收到连续的 SETUP 数据包。在此中断上，应用程序可以对接收到的 SETUP 数据包进行解码。

位 2 保留，必须保持复位值。

位 1 **EPDISD**：端点禁止中断 (Endpoint disabled interrupt)
此位指示该端点已经由应用程序禁止掉。

位 0 **XFRC**：传输完成中断 (Transfer completed interrupt)
此字段指示在此端点上设置的传输已经在 USB 和 AHB 上传输完成。

OTG_FS 设备 IN 端点 0 传输大小寄存器 (OTG_FS_DIEPTSIZ0)

OTG_FS device IN endpoint 0 transfer size register

偏移地址：0x910

复位值：0x0000 0000

在使能端点 0 之前，应用程序必须修改此寄存器。通过设备控制端点 0 控制寄存器中的端点使能位（OTG_FS_DIEPCTL0 中的 EPENA）使能端点 0 后，模块对此寄存器进行修改。仅当模块将端点使能位清零后，应用程序才能读取此寄存器。

非零端点使用端点 1–3 的寄存器。

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

位 31:21 保留，必须保持复位值。

位 20:19 **PKTCNT**：数据包计数 (Packet count)
指示端点 0 的一次数据传输包含的数据包个数。
每次从 TxFIFO 读取数据包（最大大小或短数据包）时，此字段将递减。

位 18:7 保留，必须保持复位值。

位 6:0 **XFRSIZ**：传输大小 (Transfer size)
指示端点 0 的一次数据传输包含的数据量，以字节为单位。仅当应用程序传输完这些数据后，模块才会中断该应用程序。传输大小可以设置为端点的最大数据包大小，以在每个数据包结束时中断。
每次向 TxFIFO 写入来自外部存储器的数据包时，模块会使此字段递减。

OTG_FS 设备 OUT 端点 0 传输大小寄存器 (OTG_FS_DOEPTSIZ0)

OTG_FS device OUT endpoint 0 transfer size register

偏移地址：0xB10

复位值：0x0000 0000

在使能端点 0 之前，应用程序必须修改此寄存器。通过 OTG_FS_DOEPCTL0 寄存器中的端点使能位（OTG_FS_DOEPCTL0 中的 EPENA 位）使能端点 0 后，模块对此寄存器进行修改。仅当模块将端点使能位清零后，应用程序才能读取此寄存器。

非零端点使用端点 1–3 的寄存器。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	STUPCNT		Reserved									PKTCNT	Reserved												XFRSIZ						
	rw	rw										rw													rw	rw	rw	rw	rw	rw	rw

位 31 保留，必须保持复位值。

位 30:29 **STUPCNT**：SETUP 数据包计数 (SETUP packet count)

此字段指定端点能连续接收的 SETUP 数据包数量。

01：1 个数据包

10：2 个数据包

11：3 个数据包

位 28:20 保留，必须保持复位值。

位 19 **PKTCNT**：数据包计数 (Packet count)

每向 RxFIFO 写入一个数据包，此字段递减。

位 18:7 保留，必须保持复位值。

位 6:0 **XFRSIZ**：传输大小 (Transfer size)

指示端点 0 的一次数据传输包含的数据量，以字节为单位。仅当应用程序传输完这些数据后，模块才会中断该应用程序。传输大小可以设置为端点的最大数据包大小，以在每个数据包结束时中断。

每次从 RxFIFO 读取数据包并将其写入外部存储器时，模块会使此字段递减。

OTG_FS 设备端点 x 传输大小寄存器 (OTG_FS_DIEPTSIZx)（x = 1..3，其中 x = 端点编号）

OTG_FS device endpoint-x transfer size register

偏移地址：0x910 + (端点编号 × 0x20)

复位值：0x0000 0000

在使能该端点之前，应用程序必须修改此寄存器。通过 OTG_FS_DIEPCTLx 寄存器中的端点使能位（OTG_FS_DIEPCTLx 中的 EPENA 位）使能该端点后，模块对此寄存器进行修改。仅当模块将端点使能位清零后，应用程序才能读取此寄存器。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	MCNT		PKTCNT										XFRSIZ																		
	rw/r/rw	rw/r/rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31 保留，必须保持复位值。

位 30:29 **MCNT**：多重计数 (Multi count)

对于周期性 IN 端点，此字段指示在 USB 上每帧必须发送的数据包数。模块使用此字段计算同步 IN 端点的数据 PID。

01：1 个数据包

10：2 个数据包

11：3 个数据包

位 28:19 **PKTCNT**：数据包计数 (Packet count)

指示该端点上的一次数据传输包含的数据包个数。

每次从 TxFIFO 读取数据包（最大大小或短数据包）时，此字段将递减。

位 18:0 **XFRSIZ**：传输大小 (Transfer size)

此字段包含当前端点的一次数据传输包含的数据量，以字节为单位。仅当应用程序传输完这些数据后，模块才会中断该应用程序。传输大小可以设置为端点的最大数据包大小，以在每个数据包结束时中断。

每次向 TxFIFO 写入来自外部存储器的数据包时，模块会使此字段递减。

OTG_FS 设备 IN 端点发送 FIFO 状态寄存器 (OTG_FS_DTXFSTSx)（x = 0..3，其中 x = 端点编号）

OTG_FS device IN endpoint transmit FIFO status register

IN 端点的偏移地址：0x918 + (端点编号 × 0x20) 此只读寄存器包含设备 IN 端点 TxFIFO 的空闲空间信息。

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

31:16 保留，必须保持复位值。

15:0 **INEPTFSAV**：IN 端点 TxFIFO 可用空间 (IN endpoint TxFIFO space available)
指示端点 TxFIFO 中的可用空闲空间大小。
以 32 位字为单位：
0x0：端点 TxFIFO 已满
0x1：1 个字可用
0x2：2 个字可用
0xn：n 个字可用
其它值：保留

OTG_FS 设备 OUT 端点 x 传输大小寄存器 (OTG_FS_DOEPTSIZx)（x = 1..3，其中 x = 端点编号）

OTG_FS device OUT endpoint-x transfer size register

偏移地址：0xB10 + (端点编号 × 0x20)

复位值：0x0000 0000

在使能该端点之前，应用程序必须修改此寄存器。通过 OTG_FS_DOEPCTLx 寄存器中的端点使能位（OTG_FS_DOEPCTLx 中的 EPENA 位）使能该端点后，模块对此寄存器进行修改。仅当模块将端点使能位清零后，应用程序才能读取此寄存器。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	RXDPID/STUPCNT		PKTCNT										XFRSIZ																		
	rw/r/rw	rw/r/rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31 保留，必须保持复位值。

位 30:29 **RXDPID**：接收到的数据 PID (Received data PID)
仅适用于同步 OUT 端点。
这是此端点收到的上一个数据包的 PID。
00：DATA0
01：DATA2
10：DATA1
11：MDATA

STUPCNT：SETUP 数据包计数 (SETUP packet count)

仅适用于控制 OUT 端点。

此字段指定端点能连续接收的 SETUP 数据包数量。

01：1 个数据包

10：2 个数据包

11：3 个数据包

位 28:19 **PKTCNT**：数据包计数 (Packet count)

指示该端点上的一次数据传输包含的数据包个数。

每次向 RxFIFO 写入数据包（最大大小或短数据包）后，此字段将递减。

位 18:0 **XFRSIZ**：传输大小 (Transfer size)

此字段包含当前端点的一次数据传输包含的数据量，以字节为单位。仅当应用程序传输完这些数据后，模块才会中断该应用程序。传输大小可以设置为端点的最大数据包大小，以在每个数据包结束时中断。

每次从 RxFIFO 读取数据包并将其写入外部存储器时，模块会使此字段递减。

30.16.5 OTG_FS 电源和时钟门控控制寄存器 (OTG_FS_PCGCCTL)

OTG_FS power and clock gating control register

偏移地址：0xE00

复位值：0x0000 0000

此寄存器在主机模式和设备模式下均可用。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																											PHYSUSP	Reserved			GATEHCLK	STPPCLK
																											rw			rw	rw

位 31:5 保留，必须保持复位值。

位 4 **PHYSUSP**：PHY 挂起 (PHY Suspended)

指示 PHY 已挂起。应用程序将 STPPCLK 位（位 0）置 1 后，一旦 PHY 挂起，此位就会更新。

位 3:2 保留，必须保持复位值。

位 1 **GATEHCLK**：门控 HCLK (Gate HCLK)

当 USB 通信挂起或会话无效时，应用程序会将此位置 1，以停止对除 AHB 总线从接口、主接口和唤醒逻辑之外的模块提供时钟。当 USB 恢复通信或新会话启动时，应用程序将此位清零。

位 0 **STPPCLK**：停止 PHY 时钟 (Stop PHY clock)

当 USB 通信挂起、会话无效或设备断开连接时，应用程序将此位置 1 以停止 PHY 时钟。当 USB 恢复通信或新会话启动时，应用程序将此位清零。

30.16.6 OTG_FS 寄存器映射

下表提供了 USB OTG 寄存器映射和复位值。

表 177. OTG_FS 寄存器映射和复位值

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	OTG_FS_GOTGCTL	Reserved												BSVLD	ASVLD	DBCT	CIDSTS	Reserved				DHNPEN	HSHNPEN	HNPRQ	HNGSCS	Reserved						SRQ	SRQSCS
	Reset value													0	0	0	1					0	0	0	0							0	0
0x004	OTG_FS_GOTGINT	Reserved												DBCDNE	ADTOCHG	HNGDET	Reserved							HNSSCHG	SRSSCHG	Reserved					SEDET	Res.	
	Reset value													0	0	0								0	0						0		
0x008	OTG_FS_GAHBCFG	Reserved																							PTXFELVL	TXFELVL	Reserved						GINTMSK
	Reset value																								0	0							0
0x00C	OTG_FS_GUSBCFG	CTXPKT	FDMOD	FHMOD	Reserved								TRDT									HNPCAP	SRPCAP	Reserved	PHYSEL	Reserved				TOCAL			
	Reset value												0	1	0	1						0	0		1					0	0	0
0x010	OTG_FS_GRSTCTL	AHBIDL	Reserved											TXFNUM															RXFFLSH	Reserved	FCRST	HSRST	CSRST
	Reset value	1												0	0	0								0		0	0	0	0	0	0	0	0
0x014	OTG_FS_GINTSTS	WKUINT	SRQINT	DISCINT	CIDSCHG	Reserved	PTXFE	HCINT	HPRTINT	Reserved		IPXFR/INCOMPISOOUT	IISOIXFR	OEPINT	IEPINT	Reserved		EOPF	ISODDRP	ENUMDNE	USBRST	USBSUSP	ESUSP	Reserved	GOUTNAKEFF	GINAKEFF	NPTXFE	RXFLVL	SOF	OTGINT	MMIS	CMOD
	Reset value	0	0	0	0		1	0	0			0	0	0	0			0	0	0	0	0	0		0	0	1	0	0	0	0	0
0x018	OTG_FS_GINTMSK	WUIM	SRQIM	DISCINT	CIDSCHGM	Reserved	PTXFEM	HCIM	PRTIM	Reserved		IPXFRM/IISOOXFRM	IISOIXFRM	OEPINT	IEPINT	EPMISM	Reserved	EOPFM	ISODDRPM	ENUMDNEM	USBRST	USBSUSPM	ESUSPM	Reserved	GONAKEFFM	GINAKEFFM	NPTXFEM	RXFLVLM	SOFM	OTGINT	MMISM	Reserved	
	Reset value	0	0	0	0		0	0	0			0	0	0	0	0		0	0	0	0	0	0		0	0	0	0	0	0	0		
0x01C	OTG_FS_GRXSTSR (host mode)	Reserved											PKTSTS				DPID		BCNT											CHNUM			
	Reset value												0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
	OTG_FS_GRXSTSR (host mode)	Reserved							FRMNUM				PKTSTS				DPID		BCNT											EPNUM			
	Reset value								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x020	OTG_FS_GRXSTSR (host mode)	Reserved											PKTSTS				DPID		BCNT											CHNUM			
	Reset value												0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
	OTG_FS_GRXSTSR (host mode)	Reserved							FRMNUM				PKTSTS				DPID		BCNT											EPNUM			
	Reset value								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

表 177. OTG_FS 寄存器映射和复位值（续）

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x024	OTG_FS_GRXFSIZ	Reserved																RXFD															
	Reset value																	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0x028	OTG_FS_HNPTXFSIZ/ OTG_FS_DIEPTXF0	NPTXFD/TX0FD																NPTXFSA/TX0FSA															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0x02C	OTG_FS_HNPTXSTS	Res.	NPTXQTOP							NPTQXSAV								NPTXFSAV															
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0x038	OTG_FS_GCCFG	Reserved										NOVBUSSENS	SOFOUTEN	VBUSBSEN	VBUSASEN	Reserved	.PWRDWN	Reserved															
	Reset value											0	0	0	0		0															
0x03C	OTG_FS_CID	PRODUCT_ID																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0
0x100	OTG_FS_HPTXFSIZ	PTXFSIZ																PTXSA															
	Reset value	0	0	0	0	0	1	1	1	0	1	1	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	1	0	0
0x104	OTG_FS_DIEPTXF1	INEPTXFD																INEPTXSA															
	Reset value	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
0x108	OTG_FS_DIEPTXF2	INEPTXFD																INEPTXSA															
	Reset value	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
0x10C	OTG_FS_DIEPTXF3	INEPTXFD																INEPTXSA															
	Reset value	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
0x400	OTG_FS_HCFG	Reserved																													FSLSS	FSLSPCS
	Reset value																														0	0	0
0x404	OTG_FS_HFIR	Reserved																FRIVL															
	Reset value																	1	1	1	0	1	0	1	0	0	1	1	0	0	0	0	0
0x408	OTG_FS_HFNUM	FTREM																FRNUM															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x410	OTG_FS_HPTXSTS	PTXQTOP								PTXQSAV								PTXFSAVL															
	Reset value	0	0	0	0	0	0	0	0	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
0x414	OTG_FS_HAINT	Reserved																HAINT															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x418	OTG_FS_HAINTMSK	Reserved																HAINTM															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x440	OTG_FS_HPRT	Reserved													PSPD	PTCTL		PPWR	PLSTS	Reserved	PRST	PSUSP	PRES	POCCHNG	POCA	PENCHNG	PENA	PCDET	PCSTS				
	Reset value														0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
0x500	OTG_FS_HCCHAR0	CHENA	CHDIS	ODDFRM	DAD							MCNT		EPTYP		LSDEV	Reserved	EPDIR	EPNUM					MPSIZ									
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x520	OTG_FS_HCCHAR1	CHENA	CHDIS	ODDFRM	DAD							MCNT		EPTYP		LSDEV	Reserved	EPDIR	EPNUM					MPSIZ									
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

表 177. OTG_FS 寄存器映射和复位值（续）

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x540	**OTG_FS_HCCHAR2**	CHENA	CHDIS	ODDFRM	DAD							MCNT		EPTYP		LSDEV	Reserved	EPDIR	EPNUM				MPSIZ											
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x560	**OTG_FS_HCCHAR3**	CHENA	CHDIS	ODDFRM	DAD							MCNT		EPTYP		LSDEV	Reserved	EPDIR	EPNUM				MPSIZ											
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x580	**OTG_FS_HCCHAR4**	CHENA	CHDIS	ODDFRM	DAD							MCNT		EPTYP		LSDEV	Reserved	EPDIR	EPNUM				MPSIZ											
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x5A0	**OTG_FS_HCCHAR5**	CHENA	CHDIS	ODDFRM	DAD							MCNT		EPTYP		LSDEV	Reserved	EPDIR	EPNUM				MPSIZ											
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x5C0	**OTG_FS_HCCHAR6**	CHENA	CHDIS	ODDFRM	DAD							MCNT		EPTYP		LSDEV	Reserved	EPDIR	EPNUM				MPSIZ											
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x5E0	**OTG_FS_HCCHAR7**	CHENA	CHDIS	ODDFRM	DAD							MCNT		EPTYP		LSDEV	Reserved	EPDIR	EPNUM				MPSIZ											
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x508	**OTG_FS_HCINT0**	Reserved																					DTERR	FRMOR	BBERR	TXERR	Reserved	ACK	NAK	STALL	Reserved	CHH	XFRC
	Reset value																						0	0	0	0		0	0	0		0	0
0x528	**OTG_FS_HCINT1**	Reserved																					DTERR	FRMOR	BBERR	TXERR	Reserved	ACK	NAK	STALL	Reserved	CHH	XFRC
	Reset value																						0	0	0	0		0	0	0		0	0
0x548	**OTG_FS_HCINT2**	Reserved																					DTERR	FRMOR	BBERR	TXERR	Reserved	ACK	NAK	STALL	Reserved	CHH	XFRC
	Reset value																						0	0	0	0		0	0	0		0	0
0x568	**OTG_FS_HCINT3**	Reserved																					DTERR	FRMOR	BBERR	TXERR	Reserved	ACK	NAK	STALL	Reserved	CHH	XFRC
	Reset value																						0	0	0	0		0	0	0		0	0
0x588	**OTG_FS_HCINT4**	Reserved																					DTERR	FRMOR	BBERR	TXERR	Reserved	ACK	NAK	STALL	Reserved	CHH	XFRC
	Reset value																						0	0	0	0		0	0	0		0	0
0x5A8	**OTG_FS_HCINT5**	Reserved																					DTERR	FRMOR	BBERR	TXERR	Reserved	ACK	NAK	STALL	Reserved	CHH	XFRC
	Reset value																						0	0	0	0		0	0	0		0	0
0x5C8	**OTG_FS_HCINT6**	Reserved																					DTERR	FRMOR	BBERR	TXERR	Reserved	ACK	NAK	STALL	Reserved	CHH	XFRC
	Reset value																						0	0	0	0		0	0	0		0	0
0x5E8	**OTG_FS_HCINT7**	Reserved																					DTERR	FRMOR	BBERR	TXERR	Reserved	ACK	NAK	STALL	Reserved	CHH	XFRC
	Reset value																						0	0	0	0		0	0	0		0	0
0x50C	**OTG_FS_HCINTMSK0**	Reserved																					DTERRM	FRMORM	BBERRM	TXERRM	NYET	ACKM	NAKM	STALLM	Reserved	CHHM	XFRCM
	Reset value																						0	0	0	0	0	0	0	0		0	0

表 177. OTG_FS 寄存器映射和复位值（续）

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x52C	OTG_FS_HCINTMSK1	Reserved																					DTERRM	FRMORM	BBERRM	TXERRM	NYET	ACKM	NAKM	STALLM	Reserved	CHHM	XFRCM
	Reset value																						0	0	0	0	0	0	0	0		0	0
0x54C	OTG_FS_HCINTMSK2	Reserved																					DTERRM	FRMORM	BBERRM	TXERRM	NYET	ACKM	NAKM	STALLM	Reserved	CHHM	XFRCM
	Reset value																						0	0	0	0	0	0	0	0		0	0
0x56C	OTG_FS_HCINTMSK3	Reserved																					DTERRM	FRMORM	BBERRM	TXERRM	NYET	ACKM	NAKM	STALLM	Reserved	CHHM	XFRCM
	Reset value																						0	0	0	0	0	0	0	0		0	0
0x58C	OTG_FS_HCINTMSK4	Reserved																					DTERRM	FRMORM	BBERRM	TXERRM	NYET	ACKM	NAKM	STALLM	Reserved	CHHM	XFRCM
	Reset value																						0	0	0	0	0	0	0	0		0	0
0x5AC	OTG_FS_HCINTMSK5	Reserved																					DTERRM	FRMORM	BBERRM	TXERRM	NYET	ACKM	NAKM	STALLM	Reserved	CHHM	XFRCM
	Reset value																						0	0	0	0	0	0	0	0		0	0
0x5CC	OTG_FS_HCINTMSK6	Reserved																					DTERRM	FRMORM	BBERRM	TXERRM	NYET	ACKM	NAKM	STALLM	Reserved	CHHM	XFRCM
	Reset value																						0	0	0	0	0	0	0	0		0	0
0x5EC	OTG_FS_HCINTMSK7	Reserved																					DTERRM	FRMORM	BBERRM	TXERRM	NYET	ACKM	NAKM	STALLM	Reserved	CHHM	XFRCM
	Reset value																						0	0	0	0	0	0	0	0		0	0
0x510	OTG_FS_HCTSIZ0	Reserved	DPID		PKTCNT										XFRSIZ																		
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x530	OTG_FS_HCTSIZ1	Reserved	DPID		PKTCNT										XFRSIZ																		
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x550	OTG_FS_HCTSIZ2	Reserved	DPID		PKTCNT										XFRSIZ																		
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x570	OTG_FS_HCTSIZ3	Reserved	DPID		PKTCNT										XFRSIZ																		
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x590	OTG_FS_HCTSIZ4	Reserved	DPID		PKTCNT										XFRSIZ																		
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x5B0	OTG_FS_HCTSIZ5	Reserved	DPID		PKTCNT										XFRSIZ																		
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x5D0	OTG_FS_HCTSIZ6	Reserved	DPID		PKTCNT										XFRSIZ																		
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x5F0	OTG_FS_HCTSIZ7	Reserved	DPID		PKTCNT										XFRSIZ																		
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

表 177. OTG_FS 寄存器映射和复位值（续）

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x800	OTG_FS_DCFG	Reserved																			PFIVL		DAD							Reserved	NZLSOHSK	DSPD	
	Reset value																				0	0	0	0	0	0	0	0	0	0	0	0	0
0x804	OTG_FS_DCTL	Reserved																				POPRGDNE	CGONAK	SGONAK	CGINAK	SGINAK	TCTL			GONSTS	GINSTS	SDIS	RWUSIG
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0
0x808	OTG_FS_DSTS	Reserved										FNSOF											Reserved							EERR	ENUMSPD		SUSPSTS
	Reset value											0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				0	0	0	0
0x810	OTG_FS_DIEPMSK	Reserved																										INEPNEM	INEPNMM	ITTXFEMSK	TOM	Reserved	EPDM	XFRCM
	Reset value																											0	0	0	0		0	0
0x814	OTG_FS_DOEPMSK	Reserved																												OTEPDM	STUPM	Reserved	EPDM	XFRCM
	Reset value																													0	0		0	0
0x818	OTG_FS_DAINT	OEPINT																IEPINT															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x81C	OTG_FS_DAINTMSK	OEPM																IEPM															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x828	OTG_FS_DVBUSDIS	Reserved																VBUSDT															
	Reset value																	0	0	0	1	0	1	1	1	1	1	0	1	0	1	1	1
0x82C	OTG_FS_DVBUSPULSE	Reserved																				DVBUSP											
	Reset value																					0	1	0	1	1	0	1	1	1	0	0	0
0x834	OTG_FS_DIEPEMPMSK	Reserved																INEPTXFEM															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x900	OTG_FS_DIEPCTL0	EPENA	EPDIS	Reserved			SNAK	CNAK	TXFNUM				Stall	Reserved	EPTYP		NAKSTS	Reserved	USBAEP	Reserved												MPSIZ	
	Reset value	0	0				0	0	0	0	0	0	0		0	0	0		1													0	0
0x918	TG_FS_DTXFSTS0	Reserved																INEPTFSAV															
	Reset value																	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0x920	OTG_FS_DIEPCTL1	EPENA	EPDIS	SODDFRM/SD1PID	SD0PID/SEVNFRM	SNAK	CNAK	TXFNUM					Stall	Reserved	EPTYP		NAKSTS	EONUM/DPID	USBAEP	Reserved												MPSIZ	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0				0	0	0	0	0	0	0	0	0	0	0
0x938	TG_FS_DTXFSTS1	Reserved																INEPTFSAV															
	Reset value																	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0

表 177. OTG_FS 寄存器映射和复位值（续）

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x940	**OTG_FS_DIEPCTL2**	EPENA	EPDIS	SODDFRM	SD0PID/SEVNFRM	SNAK	CNAK	TXFNUM				Stall	Reserved	EPTYP		NAKSTS	EONUM/DPID	USBAEP	Reserved				MPSIZ										
	Reset value	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0
0x958	**TG_FS_DTXFSTS2**	Reserved																INEPTFSAV															
	Reset value																	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0x960	**OTG_FS_DIEPCTL3**	EPENA	EPDIS	SODDFRM	SD0PID/SEVNFRM	SNAK	CNAK	TXFNUM				Stall	Reserved	EPTYP		NAKSTS	EONUM/DPID	USBAEP	Reserved				MPSIZ										
	Reset value	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0
0x978	**TG_FS_DTXFSTS3**	Reserved																INEPTFSAV															
	Reset value																	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0xB00	**OTG_FS_DOEPCTL0**	EPENA	EPDIS	Reserved		SNAK	CNAK	Reserved				Stall	SNPM	EPTYP		NAKSTS	Reserved	USBAEP	Reserved												MPSIZ	
	Reset value	0	0			0	0					0	0	0	0	0		1													0	0
0xB20	**OTG_FS_DOEPCTL1**	EPENA	EPDIS	SODDFRM	SD0PID/SEVNFRM	SNAK	CNAK	Reserved				Stall	SNPM	EPTYP		NAKSTS	EONUM/DPID	USBAEP	Reserved				MPSIZ										
	Reset value	0	0	0	0	0	0					0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0
0xB40	**OTG_FS_DOEPCTL2**	EPENA	EPDIS	SODDFRM	SD0PID/SEVNFRM	SNAK	CNAK	Reserved				Stall	SNPM	EPTYP		NAKSTS	EONUM/DPID	USBAEP	Reserved				MPSIZ										
	Reset value	0	0	0	0	0	0					0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0
0xB60	**OTG_FS_DOEPCTL3**	EPENA	EPDIS	SODDFRM	SD0PID/SEVNFRM	SNAK	CNAK	Reserved				Stall	SNPM	EPTYP		NAKSTS	EONUM/DPID	USBAEP	Reserved				MPSIZ										
	Reset value	0	0	0	0	0	0					0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0
0x908	**OTG_FS_DIEPINT0**	Reserved																								TXFE	INEPNE	Reserved	ITTXFE	TOC	Reserved	EPDISD	XFRC
	Reset value																									1	0		0	0		0	0
0x928	**OTG_FS_DIEPINT1**	Reserved																								TXFE	INEPNE	Reserved	ITTXFE	TOC	Reserved	EPDISD	XFRC
	Reset value																									1	0		0	0		0	0
0x948	**OTG_FS_DIEPINT2**	Reserved																								TXFE	INEPNE	Reserved	ITTXFE	TOC	Reserved	EPDISD	XFRC
	Reset value																									1	0		0	0		0	0

表 177. OTG_FS 寄存器映射和复位值（续）

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x968	OTG_FS_DIEPINT3	Reserved																								TXFE	INEPNE	Reserved	ITTXFE	TOC	Reserved	EPDISD	XFRC
	Reset value																									1	0		0	0		0	0
0xB08	OTG_FS_DOEPINT0	Reserved																								Reserved	B2BSTUP	Reserved	OTEPDIS	STUP	Reserved	EPDISD	XFRC
	Reset value																										0		0	0		0	0
0xB28	OTG_FS_DOEPINT1	Reserved																								Reserved	B2BSTUP	Reserved	OTEPDIS	STUP	Reserved	EPDISD	XFRC
	Reset value																										0		0	0		0	0
0xB48	OTG_FS_DOEPINT2	Reserved																								Reserved	B2BSTUP	Reserved	OTEPDIS	STUP	Reserved	EPDISD	XFRC
	Reset value																										0		0	0		0	0
0xB68	OTG_FS_DOEPINT3	Reserved																								Reserved	B2BSTUP	Reserved	OTEPDIS	STUP	Reserved	EPDISD	XFRC
	Reset value																										0		0	0		0	0
0x910	OTG_FS_DIEPTSIZ0	Reserved												PKTCNT												XFRSIZ							
	Reset value													0	0											0	0	0	0	0	0	0	0
0x930	OTG_FS_DIEPTSIZ1	Reserved	MCNT		PKTCNT																					XFRSIZ							
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x950	OTG_FS_DIEPTSIZ2	Reserved	MCNT		PKTCNT																					XFRSIZ							
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x970	OTG_FS_DIEPTSIZ3	Reserved	MCNT		PKTCNT																					XFRSIZ							
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xB10	OTG_FS_DOEPTSIZ0	Reserved	STUPCNT		Reserved									PKTCNT												XFRSIZ							
	Reset value		0	0										0												0	0	0	0	0	0	0	0
0xB30	OTG_FS_DOEPTSIZ1	Reserved	RXDPID/STUPCNT		PKTCNT																					XFRSIZ							
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xB50	OTG_FS_DOEPTSIZ2	Reserved	RXDPID/STUPCNT		PKTCNT																					XFRSIZ							
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xB70	OTG_FS_DOEPTSIZ3	Reserved	RXDPID/STUPCNT		PKTCNT																					XFRSIZ							
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xE00	OTG_FS_PCGCCTL	Reserved																											PHYSUSP	Reserved		GATEHCLK	STPPCLK
	Reset value																																

有关寄存器边界地址的信息，请参见*第 52 页的表 2*。

30.17 OTG_FS 编程模型

30.17.1 模块初始化

应用程序必须执行模块初始化序列。如果上电期间连接电缆，则 OTG_FS_GINTSTS 中的当前工作模式位（OTG_FS_GINTSTS 中的 CMOD 位）将指示模式。连接“A 型”插头后，OTG_FS 控制器进入主机模式；连接“B 型”插头后，OTG_FS 控制器进入设备模式。

本节介绍了 OTG_FS 控制器上电后的初始化过程。无论是以主机模式还是设备模式工作，应用程序都必须遵循初始化序列。根据模块配置对所有模块全局寄存器进行初始化：

1. 在 OTG_FS_GAHBCFG 寄存器中编程以下字段：
   - 全局中断屏蔽位 GINTMSK = 1
   - RxFIFO 非空（OTG_FS_GINTSTS 中的 RXFLVL 位）
   - 周期性 TxFIFO 空门限
2. 在 OTG_FS_GUSBCFG 寄存器中编程以下字段：
   - HNP 功能位
   - SRP 功能位
   - FS 超时校准字段
   - USB 周转时间字段
3. 软件必须取消对 OTG_FS_GINTMSK 寄存器中以下位的屏蔽：

   OTG 中断屏蔽 (OTG interrupt mask)

   模式不匹配中断屏蔽
4. 通过读取 OTG_FS_GINTSTS 中的 CMOD 位，软件可确定 OTG_FS 控制器是在主机模式还是设备模式下工作。

30.17.2 主机初始化

要将模块作为主机进行初始化，应用程序必须执行以下步骤：

1. 编程 OTG_FS_GINTMSK 寄存器中的 HPRTINT 以将对取消屏蔽。
2. 编程 OTG_FS_HCFG 寄存器以选择全速主机。
3. 将 OTG_FS_HPRT 中的 PPWR 位编程为 1，给 USB 总线提供 V_{BUS}。
4. 等待 OTG_FS_HPRT0 中的 PCDET 中断。这表示某设备已连接到主机端口。
5. 将 OTG_FS_HPRT 中的 PRST 位编程为 1，在 USB 总线上发出复位信号。
6. 至少等待 10 ms，以便完成复位过程。
7. 将 OTG_FS_HPRT 中的 PRST 位编程为 0。
8. 等待 OTG_FS_HPRT 中的 PENCHNG 中断。
9. 读取 OTG_FS_HPRT 中的 PSPD 位以获取枚举速度。
10. 使用所选 PHY 时钟，相应地设置 HFIR 寄存器。
11. 根据步骤 9 中检测到的设备速度编程 OTG_FS_HCFG 寄存器中的 FSLSPCS 字段。如果 FSLSPCS 发生更改，则必须执行端口复位。
12. 编程 OTG_FS_GRXFSIZ 寄存器以选择接收 FIFO 的大小。
13. 编程 OTG_FS_HNPTXFSIZ 寄存器，以选择用于非周期性通信事务的非周期性发送 FIFO 的大小和起始地址。
14. 编程 OTG_FS_HPTXFSIZ 寄存器，以选择用于周期性事务的周期性通信发送 FIFO 的大小和起始地址。

要与设备通信，系统软件必须初始化并使能至少一个通道。

30.17.3 设备初始化

上电期间或者从主机模式切换为设备模式后，应用程序必须执行下列步骤来将模块作为设备进行初始化。

1. 在 OTG_FS_DCFG 寄存器中编程以下字段：
   - 设备速度
   - 非零长度状态 OUT 握手信号
2. 编程 OTG_FS_GINTMSK 寄存器以取消屏蔽以下中断：
   - USB 复位
   - 枚举完成
   - 早期挂起
   - USB 挂起
   - SOF
3. 编程 OTG_FS_GCCFG 寄存器中的 VBUSBSEN 位，以使能“B”设备模式的 V_{BUS} 感应并把DP线上的上拉电阻上拉到5V。
4. 等待 OTG_FS_GINTSTS 中的 USBRST 中断。这表示已在 USB 上检测到复位信号，复位过程自接收到此中断后约持续 10 ms。

等待 OTG_FS_GINTSTS 中的 ENUMDNE 中断。此中断指示 USB 上复位过程结束。接收到此中断时，应用程序必须读取 OTG_FS_DSTS 寄存器以确定枚举速度并执行 *第 1030 页的枚举完成时的端点初始化*中所列的步骤。

此时，设备已准备好接受 SOF 数据包并在控制端点 0 上执行控制传输。

30.17.4 主机编程模型

通道初始化

应用程序必须初始化一个或多个通道，之后才能与所连接的设备通信。要初始化和使能通道，应用程序必须执行以下步骤：

1. 编程 OTG_FS_GINTMSK 寄存器以取消对以下位的中断屏蔽：
2. 通道中断
   - 用于 OUT 事务的非周期性发送 FIFO 为空（在流水线事务级别工作且数据包计数字段编程值大于 1 时适用）。
   - 用于 OUT 事务的非周期性发送 FIFO 为半空（在流水线事务级别工作且数据包计数字段编程值大于 1 时适用）。
3. 编程 OTG_FS_HAINTMSK 寄存器以使能所选通道中断。
4. 编程 OTG_FS_HCINTMSK 寄存器，以使能主机通道中断寄存器中反映的和通信事务有关的中断。
5. 编程所选通道的 OTG_FS_HCTSIZx 寄存器，指定以字节为单位的总传输大小和包括短数据包在内的预期数据包个数。应用程序必须使用初始数据 PID（用于第一个 OUT 事务或预期从第一个 IN 事务获取）编程 PID 字段。
6. 编程所选通道的 OTG_FS_HCCHARx 寄存器，指定设备的端点特性，例如类型、速度、方向等。（仅当应用程序准备好发送或接收数据包时，才能通过将通道使能位置 1 来使能通道）。

通道的停止

应用程序可以通过编程 OTG_FS_HCCHARx 寄存器将 CHDIS 和 CHENA 位置 1 来禁止任何通道。这会使 OTG_FS 主机清空之前在该通道上发出的请求（如果有）并生成通道停止中断。应用程序在将通道重新分配给其它通信事务之前，必须等待 OTG_FS_HCINTx 中的 CHH 中断。OTG_FS 主机不会中断已在 USB 上启动的通信事务。

禁止通道前，应用程序必须确保非周期性请求队列（禁止非周期性通道时）或周期性请求队列（禁止周期性通道时）中至少有一个空闲空间。应用程序可以在请求队列已满时（禁止通道之前），通过编程 OTG_FS_HCCHARx 寄存器将 CHDIS 位置 1 和将 CHENA 位清零，清空请求队列。

出现以下任一情况时，应用程序将禁止通道：

1. IN 或 OUT 通道的 OTG_FS_HCINTx 中接收到 STALL、TXERR、BBERR 或 DTERR 中断。应用程序在接收到通道停止信号之前，必须能够接收相同通道的其它中断（DTERR、Nak、Data、TXERR）。
2. 接收到 OTG_FS_GINTSTS 中的 DISCINT（断开设备连接）中断。（应用程序将禁止所有已使能的通道）。
3. 应用程序在传输正常完成之前将其中止。

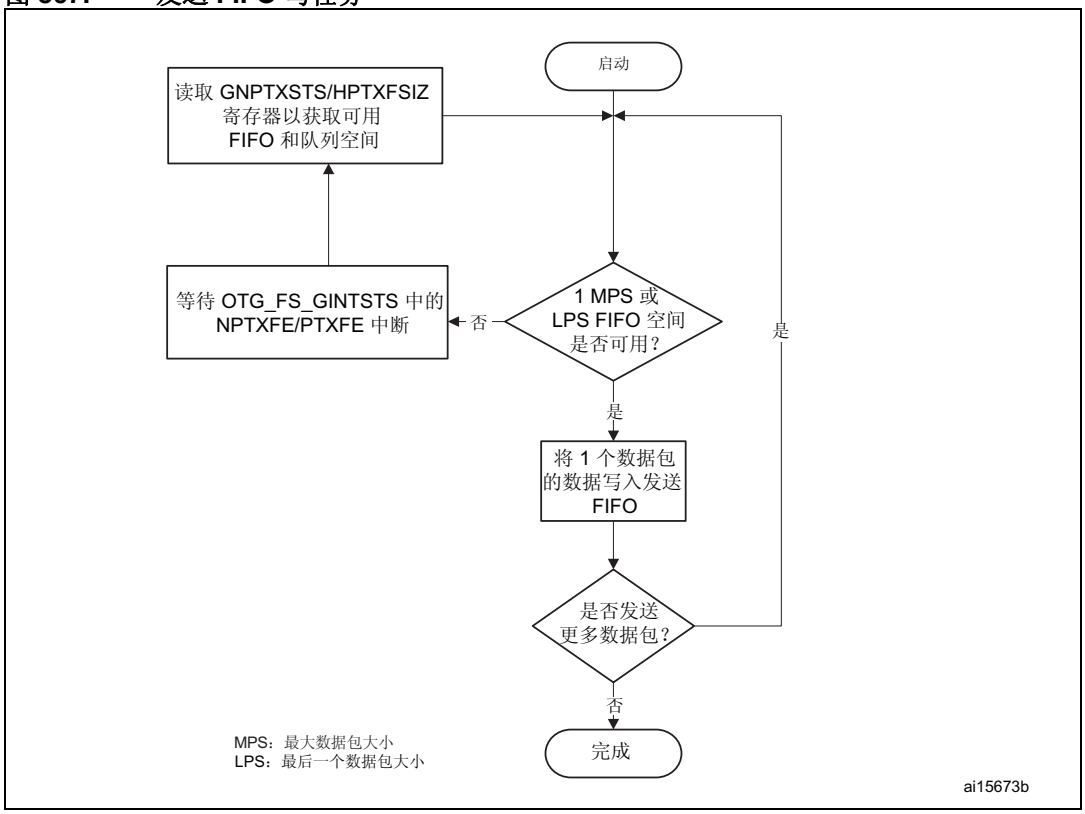
操作模型

应用程序必须初始化一个通道，之后才能与所连接的设备通信。本节介绍了针对不同 USB 事务类型要执行的操作序列。

- **写入发送 FIFO**

  应用程序对数据包执行最后一个 DWORD 写操作的同时，OTG_FS 主机自动向周期性/非周期性请求队列写入一个条目（OUT 请求）。因此开始向发送 FIFO 写入数据之前，应用程序必须确保周期性/非周期性请求队列中至少有一个空闲空间。应用程序必须始终以 DWORD 形式向发送 FIFO 写入数据。如果数据包大小不是 DWORD 的整数倍，则应用程序必须将数据包填充到 DWORD 的整数倍。OTG_FS 主机根据设定的最大数据包大小和传输大小确定实际数据包大小。

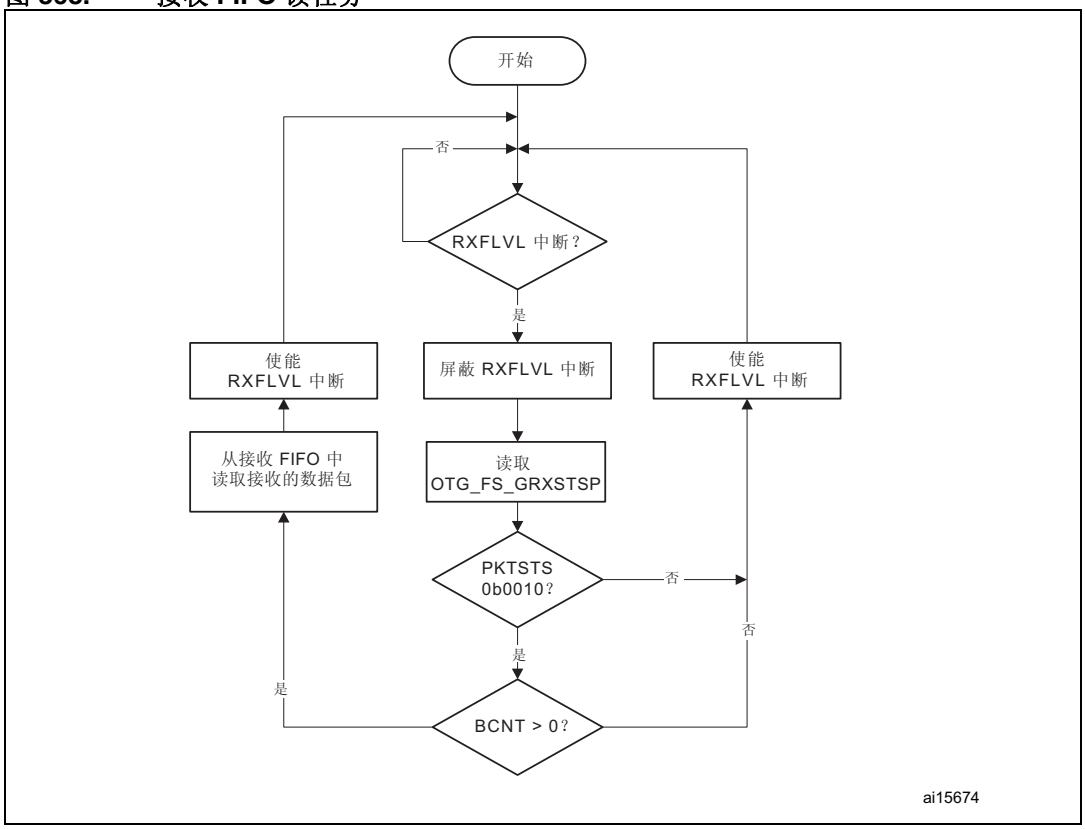
图 367. 发送 FIFO 写任务



- **读取接收 FIFO**

  应用程序必须忽略除 IN 数据包 (bx0010) 以外的所有数据包状态。

图 368. 接收 FIFO 读任务



- **批量和控制传输类型的 OUT/SETUP 通信事务**

  图 369 显示了典型的批量或控制 OUT/SETUP 流水线事务级操作。请参见通道 1 (ch_1)。该通道发送了两个批量 OUT 数据包。控制 SETUP 事务的工作方式相同，只不过只包含一个数据包。假设：

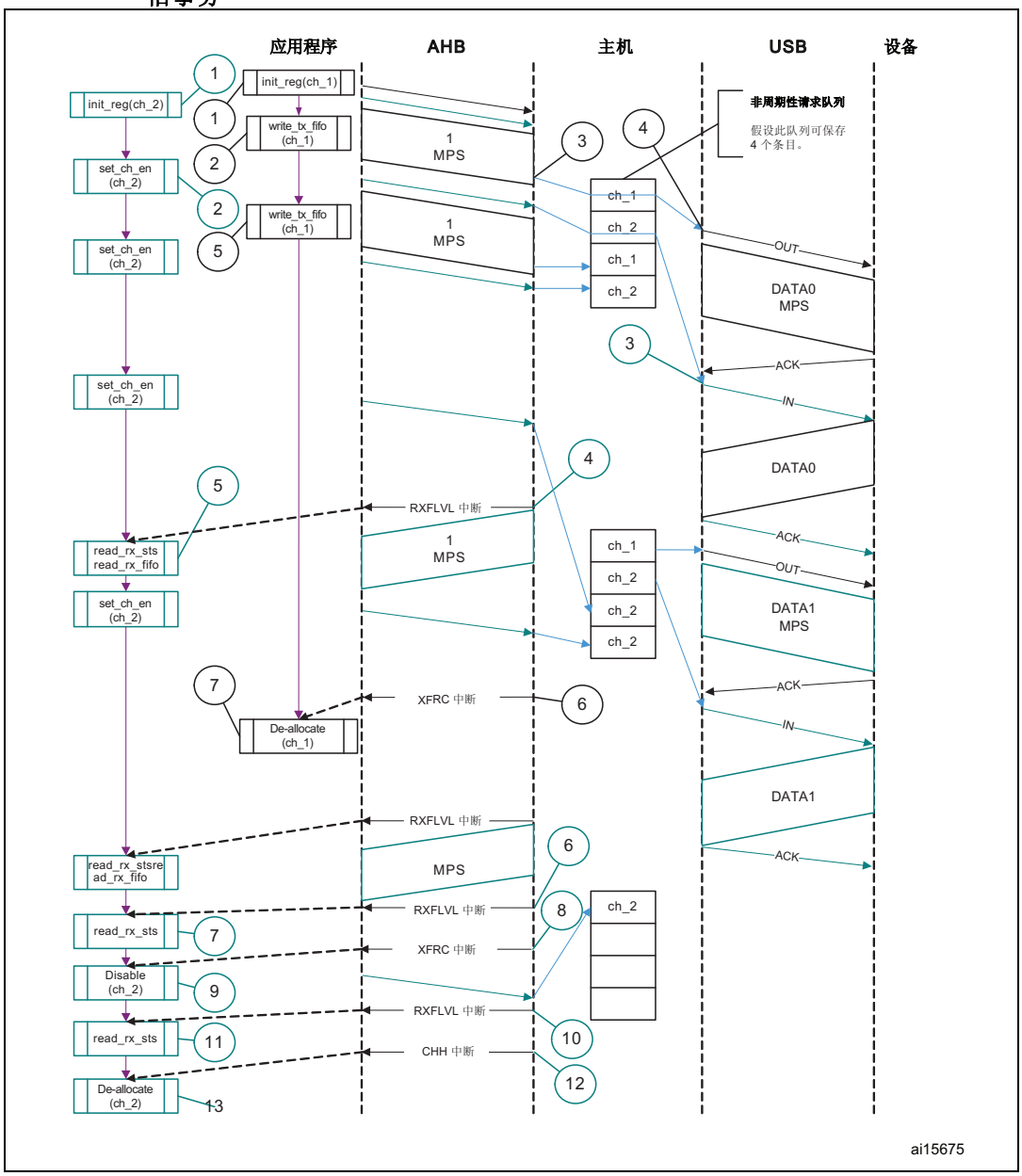
  – 应用程序尝试发送两个最大数据包大小的数据包（传输大小 = 1024 字节）。
  – 非周期性发送 FIFO 可存储两个数据包（FS 对应 128 字节）。
  – 非周期性请求队列深度 = 4。

- **正常批量和控制传输类型的 OUT/SETUP 操作**

  （通道 1）中的操作顺序如下：

  a) 初始化通道 1
  b) 写入通道 1 的第一个数据包
  c) 在应用执行最后一次 DWORD 写操作时，模块将向非周期性请求队列写入一个请求条目
  d) 只要非周期性队列非空，模块即会尝试在当前帧内发送一个 OUT 令牌
  e) 写入通道 1 的第二个（最后一个）数据包
  f) 成功完成最后一个事务后，模块立即生成 XFRC 中断
  g) 为了响应 XFRC 中断，将释放通道以供其它传输操作使用
  h) 处理非 ACK 响应

图 369. 正常批量/控制传输类型的 OUT/SETUP 通信事务和批量/控制传输类型的 IN 通信事务



以下代码示例说明了批量和控制 OUT/SETUP 传输类型的通信事务的通道相关中断服务程序。

- **批量/控制 OUT/SETUP 和批量/控制 IN 事务的中断服务程序**

  a) 批量/控制 OUT/SETUP

```
Unmask (NAK/TXERR/STALL/XFRC)
if (XFRC)
   {
   Reset Error Count
   Mask ACK
```

```
   De-allocate Channel
   }
else if (STALL)
   {
   Transfer Done = 1
   Unmask CHH
   Disable Channel
   }
else if (NAK or TXERR )
   {
   Rewind Buffer Pointers
   Unmask CHH
   Disable Channel
   if (TXERR)
      {
      Increment Error Count
      Unmask ACK
      }
   else
      {
      Reset Error Count
      }
      }
else if (CHH)
   {
   Mask CHH
   if (Transfer Done or (Error_count == 3))
      {
      De-allocate Channel
      }
   else
      {
      Re-initialize Channel
      }
   }
   else if (ACK)
      {
      Reset Error Count
      Mask ACK
      }
```

当发送 FIFO 和请求队列中有可用空间时，应用程序会将数据包写入发送 FIFO。应用程序可利用 OTG_FS_GINTSTS 中的 NPTXFE 中断确定发送 FIFO 空间。

b) 批量/控制 IN

```
Unmask (TXERR/XFRC/BBERR/STALL/DTERR)
if (XFRC)
   {
   Reset Error Count
   Unmask CHH
   Disable Channel
```

```
   Reset Error Count
   Mask ACK
   }
else if (TXERR or BBERR or STALL)
   {
   Unmask CHH
   Disable Channel
   if (TXERR)
      {
      Increment Error Count
      Unmask ACK
      }
   }
else if (CHH)
   {
   Mask CHH
   if (Transfer Done or (Error_count == 3))
      {
      De-allocate Channel
      }
   else
      {
      Re-initialize Channel
      }
   }
else if (ACK)
   {
   Reset Error Count
   Mask ACK
   }
else if (DTERR)
   {
   Reset Error Count
   }
```

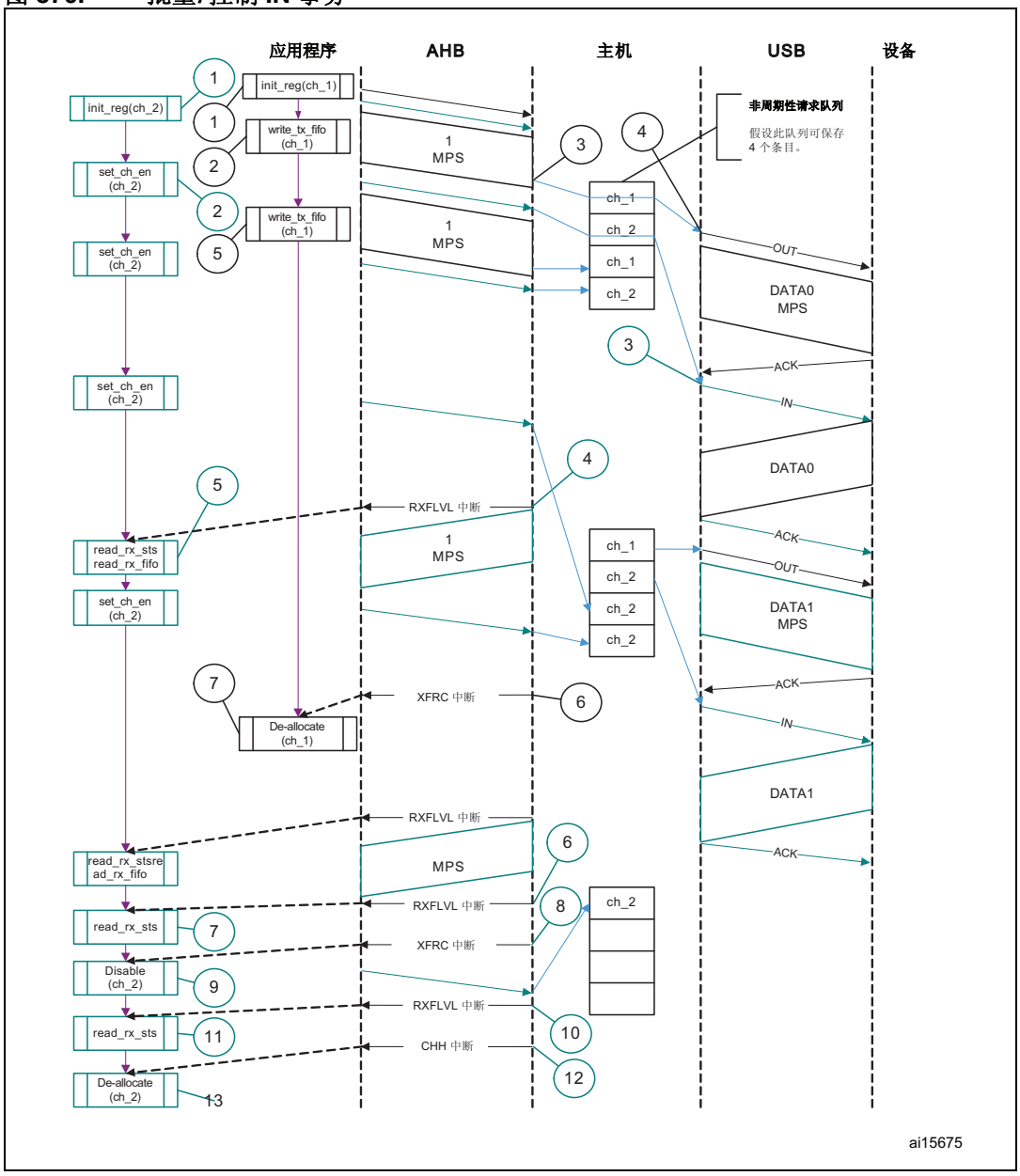
当请求队列空间可用时，应用程序会写入请求，直到接收到 XFRC 中断。

- **批量和控制 IN 事务**

  图 370 显示了典型的批量或控制传输类型的 IN 流水线事务级操作。请参见通道 2 (ch_2)。假设：

  – 应用程序要接收两个最大数据包大小的数据包（传输大小 = 1024 字节）。
  – 接收 FIFO 可以包含至少一个最大数据包大小的数据包和每个数据包的两个状态字（FS 对应 72 字节）。
  – 非周期性请求队列深度 = 4。

图 370. 批量/控制 IN 事务



操作顺序如下：

a) 初始化通道 2。

b) 将 HCCHAR2 中的 CHENA 位置 1，以向非周期性请求队列写入 IN 请求。

c) 模块尝试在完成当前 OUT 事务后发送 IN 令牌。

d) 接收到的数据包写入接收 FIFO 后，模块立即生成 RXFLVL 中断。

e) 为了响应 RXFLVL 中断，将屏蔽 RXFLVL 中断并读取接收到的数据包状态，以此确定接收的字节数，然后相应地读取接收 FIFO。之后取消对 RXFLVL 中断的屏蔽。

f) 模块针对接收 FIFO 中的传输完成状态条目生成 RXFLVL 中断。
g) 应用程序必须读取接收数据包状态，并在不是 IN 数据包（GRXSTSR 中的 PKTSTS ≠ 0b0010）时忽略它。
h) 读取接收数据包状态后，模块立即生成 XFRC 中断。
i) 为了响应 XFRC 中断，将禁止通道并停止针对其它请求向 OTG_FS_HCCHAR2 写入数据。向 OTG_FS_HCCHAR2 寄存器写入数据时，模块立即向非周期性请求队列写入通道禁止请求。
j) 停止状态写入接收 FIFO 后，模块立即生成 RXFLVL 中断。
k) 读取并忽略接收数据包状态。
l) 只要停止状态从接收 FIFO 弹出，模块立即生成 CHH 中断。
m) 为了响应 CHH 中断，将释放通道以供其它传输操作使用。
n) 处理非 ACK 响应。

- **控制事务**

控制传输的建立、数据和状态阶段必须作为三个独立的传输过程来执行。建立、数据和状态阶段的 OUT 事务与上文所述的批量 OUT 事务的执行方式类似。数据或状态阶段的 IN 事务与上文所述的批量 IN 事务的执行方式类似。在所有这三个阶段，应用程序都会将 OTG_FS_HCCHAR1 中的 EPTYP 字段设置为控制传输类型。在建立阶段期间，应用程序会将 OTG_FS_HCTSIZ1 中的 PID 字段设置为 SETUP。

- **中断 OUT 事务**

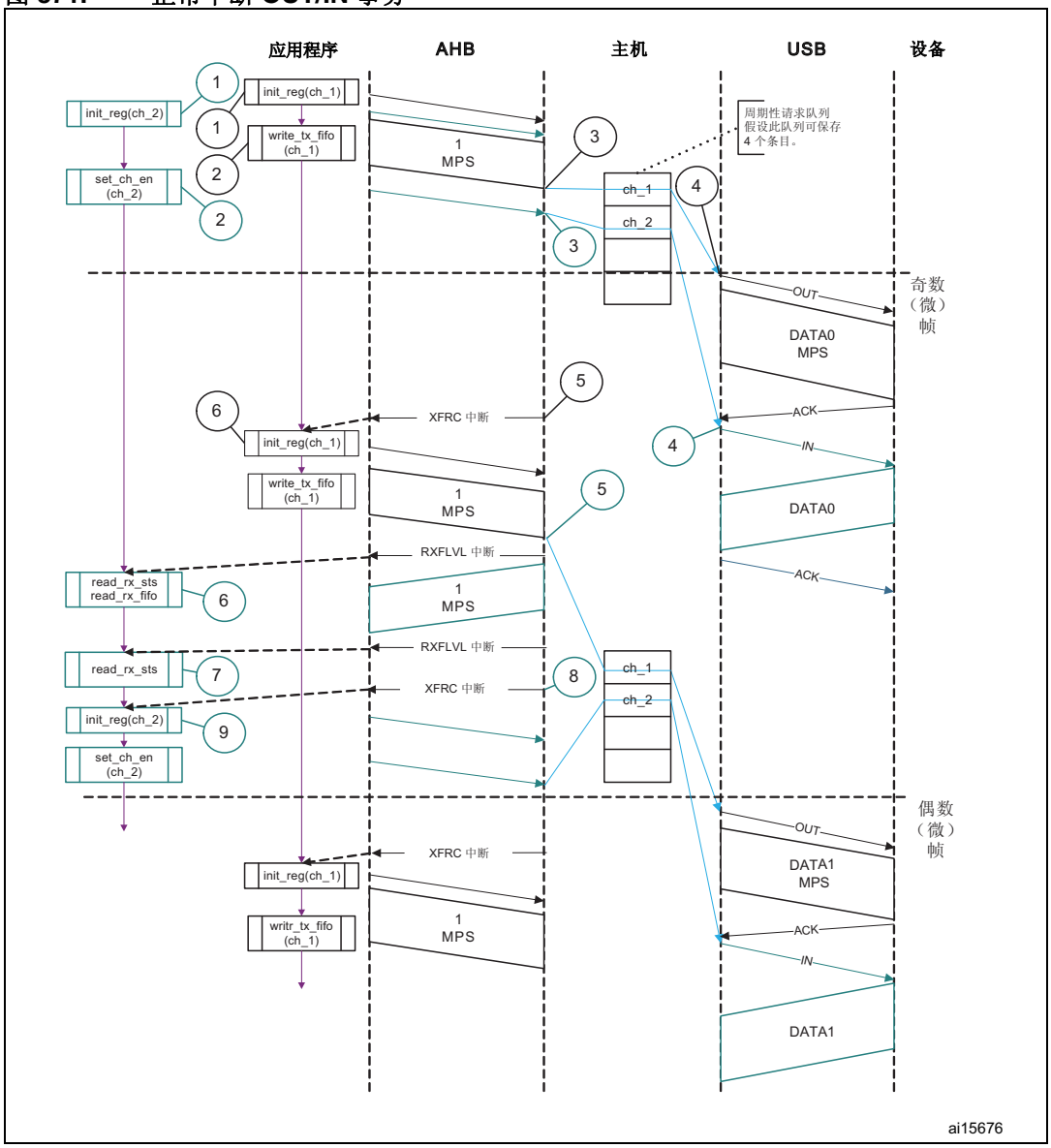
图 371 显示了典型的中断 OUT 操作。假设：

– 应用程序尝试从奇数帧开始，每帧发送一个数据包（高达 1 个最大数据包大小）（传输大小 = 1024 字节）
– 周期性发送 FIFO 可存储一个数据包 (1 KB)
– 周期性请求队列深度 = 4

操作顺序如下：

a) 初始化和使能通道 1。应用程序必须将 OTG_FS_HCCHAR1 中的 ODDFRM 位置 1。
b) 写入通道 1 的第一个数据包。
c) 应用程序在执行每个数据包的最后一次 DWORD 写操作时，OTG_FS 主机向周期性请求队列写入一个请求条目。
d) OTG_FS 主机尝试在下一（奇数）帧发送 OUT 令牌。
e) 成功发送最后一个数据包后，OTG_FS 主机立即生成 XFRC 中断。
f) 为了响应 XFRC 中断，将重新初始化通道以供下一传输操作使用。

图 371. 正常中断 OUT/IN 事务



- **中断 OUT/IN 事务的中断服务程序**

  a) 中断 OUT

```
Unmask (NAK/TXERR/STALL/XFRC/FRMOR)
if (XFRC)
   {
   Reset Error Count
   Mask ACK
   De-allocate Channel
   }
else
   if (STALL or FRMOR)
      {
      Mask ACK
      Unmask CHH
```

```
      Disable Channel
      if (STALL)
        {
        Transfer Done = 1
        }
      }
    else
      if (NAK or TXERR)
        {
        Rewind Buffer Pointers
        Reset Error Count
        Mask ACK
        Unmask CHH
        Disable Channel
        }
      else
        if (CHH)
        {
        Mask CHH
        if (Transfer Done or (Error_count == 3))
          {
          De-allocate Channel
          }
        else
          {
          Re-initialize Channel (in next b_interval - 1 Frame)
          }
        }
      else
        if (ACK)
        {
        Reset Error Count
        Mask ACK
        }
```

应用程序利用 OTG_FS_GINTSTS 中的 NPTXFE 中断确定发送 FIFO 空间。

b) 中断 IN

```
Unmask (NAK/TXERR/XFRC/BBERR/STALL/FRMOR/DTERR)
if (XFRC)
  {
  Reset Error Count
  Mask ACK
  if (OTG_FS_HCTSIZx.PKTCNT == 0)
    {
    De-allocate Channel
    }
  else
    {
    Transfer Done = 1
    Unmask CHH
    Disable Channel
    }
```

```
    }
  else
    if (STALL or FRMOR or NAK or DTERR or BBERR)
      {
      Mask ACK
      Unmask CHH
      Disable Channel
      if (STALL or BBERR)
        {
        Reset Error Count
        Transfer Done = 1
        }
      else
        if (!FRMOR)
        {
        Reset Error Count
        }
      }
  else
    if (TXERR)
      {
      Increment Error Count
      Unmask ACK
      Unmask CHH
      Disable Channel
      }
  else
    if (CHH)
      {
      Mask CHH
      if (Transfer Done or (Error_count == 3))
        {
        De-allocate Channel
        }
      else
        Re-initialize Channel (in next b_interval - 1 /Frame)
      }
    }
  else
    if (ACK)
      {
      Reset Error Count
      Mask ACK
      }
```

- **中断 IN 事务**

  假设：

  – 应用程序要从奇数帧开始，每帧接收一个数据包（最大 1 个最大数据包大小）（传输大小 = 1024 字节）。

  – 接收 FIFO 可以保存至少一个最大数据包大小的数据包和每个数据包的两个状态字（1031 字节）。

  – 周期性请求队列深度 = 4。

- **正常中断 IN 操作**

  操作顺序如下：

  a) 初始化通道 2。应用程序必须将 OTG_FS_HCCHAR2 中的 ODDFRM 位置 1。

  b) 将 OTG_FS_HCCHAR2 中的 CHENA 位置 1，以将 IN 请求写入周期性请求队列。

  c) 只要 CHENA 置位，对于每次 OTG_FS_HCCHAR2 寄存器写操作，OTG_FS 主机都会将一个 IN 请求写入周期性请求队列。

  d) OTG_FS 主机尝试在下一（奇数）帧发送 IN 令牌。

  e) 接收并写入 IN 数据包以接收 FIFO 后，OTG_FS 主机便会立即生成 RXFLVL 中断。

  f) 为响应 RXFLVL 中断，读取接收到的数据包状态以确定接收的字节数，然后相应地读取接收 FIFO。应用程序必须在读取接收 FIFO 前屏蔽 RXFLVL 中断，并且在读取整个数据包后取消屏蔽。

  g) 模块针对接收 FIFO 中的传输完成状态条目生成 RXFLVL 中断。应用程序必须读取接收数据包状态，并在不是 IN 数据包（GRXSTSR 中的 PKTSTS ≠ 0b0010）时忽略它。

  h) 读取接收数据包状态后，模块立即生成 XFRC 中断。

  i) 为响应 XFRC 中断，将读取 OTG_FS_HCTSIZ2 中的 PKTCNT 字段。如果 OTG_FS_HCTSIZ2 中的 PKTCNT 位不等于 0，则在重新初始化通道以进行下次传输前（如果存在），禁止该通道。如果 OTG_FS_HCTSIZ2 中的 PKTCNT 位等于 0，则重新初始化通道以进行下次传输。此时，应用程序必须复位 OTG_FS_HCCHAR2 中的 ODDFRM 位。

- **同步 OUT 事务**

  图 372 中显示了典型的同步 OUT 操作。假设：

  – 应用程序尝试从奇数帧开始，每帧发送一个数据包（最大 1 个最大数据包大小）。（传输大小 = 1024 字节）。

  – 周期性发送 FIFO 可存储一个数据包 (1 KB)。

  – 周期性请求队列深度 = 4。

  操作顺序如下：

  a) 初始化和使能通道 1。应用程序必须将 OTG_FS_HCCHAR1 中的 ODDFRM 位置 1。

  b) 写入通道 1 的第一个数据包。

  c) 应用程序在执行每个数据包的最后一次 DWORD 写操作时，OTG_FS 主机向周期性请求队列写入一个请求条目。

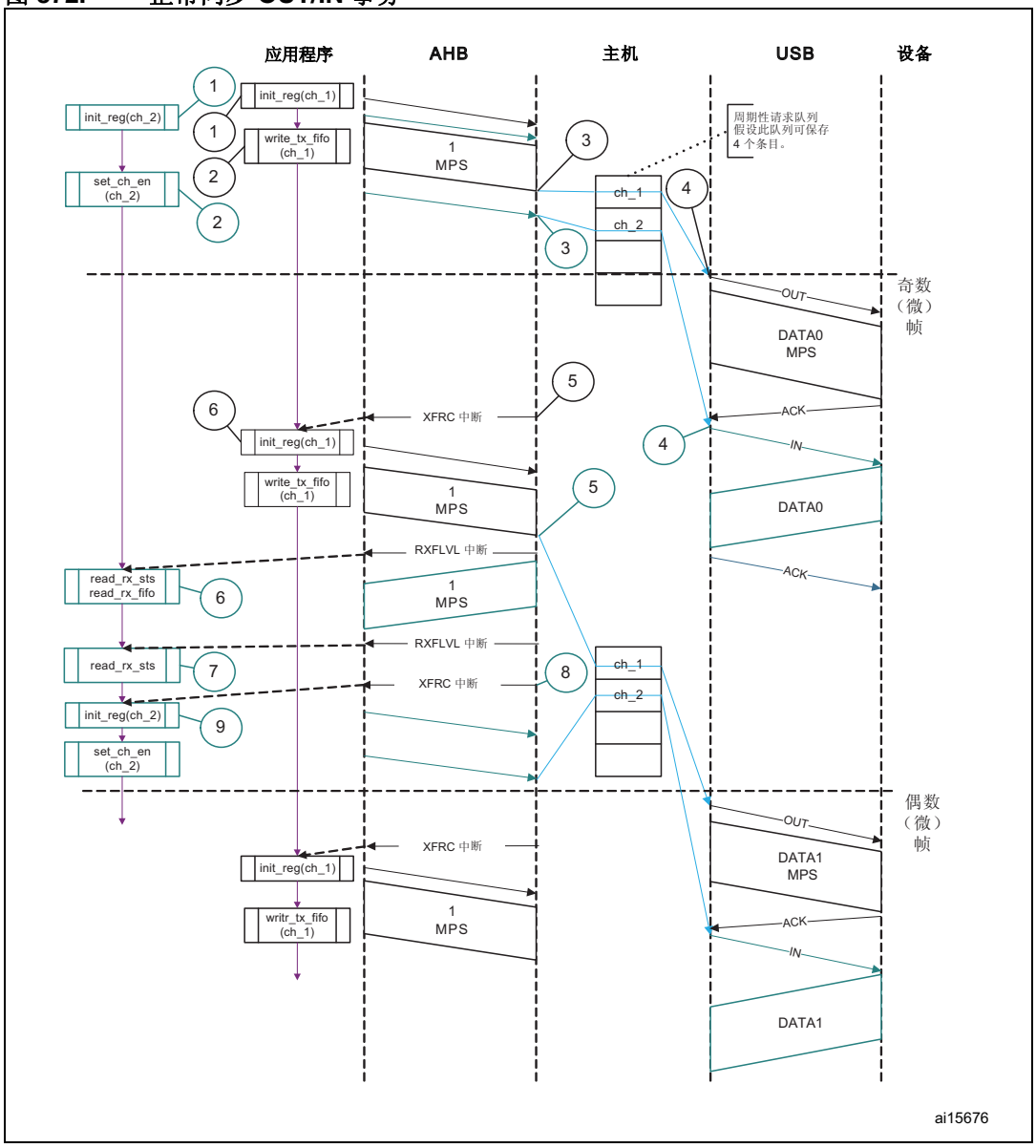
  d) OTG_FS 主机尝试在下一（奇数）帧发送 OUT 令牌。

  e) 成功发送最后一个数据包后，OTG_FS 主机立即生成 XFRC 中断。

  f) 为了响应 XFRC 中断，将重新初始化通道以供下一传输操作使用。

  g) 处理非 ACK 响应。

图 372. 正常同步 OUT/IN 事务



- **同步 OUT/IN 事务的中断服务程序**

  代码示例：同步 OUT

```
Unmask (FRMOR/XFRC)
if (XFRC)
   {
   De-allocate Channel
   }
else
   if (FRMOR)
      {
      Unmask CHH
      Disable Channel
      }
```

```
   else
   if (CHH)
      {
      Mask CHH
      De-allocate Channel
      }
```

代码示例：Isochronous IN

```
Unmask (TXERR/XFRC/FRMOR/BBERR)
if (XFRC or FRMOR)
   {
   if (XFRC and (OTG_FS_HCTSIZx.PKTCNT == 0))
      {
      Reset Error Count
      De-allocate Channel
      }
   else
      {
      Unmask CHH
      Disable Channel
      }
   }
   else
      if (TXERR or BBERR)
         {
         Increment Error Count
         Unmask CHH
         Disable Channel
         }
      else
         if (CHH)
            {
            Mask CHH
            if (Transfer Done or (Error_count == 3))
               {
               De-allocate Channel
               }
            else
               {
               Re-initialize Channel
               }
            }
```

- **同步 IN 事务**

  假设：

  – 应用程序尝试从下一个奇数帧开始，每帧接收一个数据包（最大 1 个最大数据包大小）（传输大小 = 1024 字节）。

  – 接收 FIFO 可以保存至少一个最大数据包大小的数据包和每个数据包的两个状态字（1031 字节）。

  – 周期性请求队列深度 = 4。

  操作顺序如下：

  a) 初始化通道 2。应用程序必须将 OTG_FS_HCCHAR2 中的 ODDFRM 位置 1。

  b) 将 OTG_FS_HCCHAR2 中的 CHENA 位置 1，以将 IN 请求写入周期性请求队列。

  c) 只要 CHENA 置位，对于每次 OTG_FS_HCCHAR2 寄存器写操作，OTG_FS 主机都会将一个 IN 请求写入周期性请求队列。

  d) OTG_FS 主机尝试在下一奇数帧发送 IN 令牌。

  e) 接收并写入 IN 数据包以接收 FIFO 后，OTG_FS 主机便会立即生成 RXFLVL 中断。

  f) 为响应 RXFLVL 中断，读取接收到的数据包状态以确定接收的字节数，然后相应地读取接收 FIFO。应用程序必须在读取接收 FIFO 前屏蔽 RXFLVL 中断，并且在读取整个数据包后取消屏蔽。

  g) 模块针对接收 FIFO 中的传输完成状态条目生成 RXFLVL 中断。应用程序必须读取接收数据包状态，并在不是 IN 数据包（OTG_FS_GRXSTSR 中的 PKTSTS 位 ≠ 0b0010）时忽略它。

  h) 读取接收数据包状态后，模块立即生成 XFRC 中断。

  i) 为响应 XFRC 中断，将读取 OTG_FS_HCTSIZ2 中的 PKTCNT 字段。如果在 OTG_FS_HCTSIZ2 中的 PKTCNT≠ 0，则在重新初始化通道以进行下次传输前（如果存在），禁止该通道。如果 OTG_FS_HCTSIZ2 中的 PKTCNT = 0，则重新初始化通道以进行下次传输。此时，应用程序必须复位 OTG_FS_HCCHAR2 中的 ODDFRM 位。

- **选择队列深度**

  请谨慎选择周期性和非周期性请求队列深度，以与要访问的周期性/非周期性端点数量相匹配。

  非周期性请求队列深度会影响非周期性传输的性能。队列越深（加上足够的 FIFO 大小），模块就更能对非周期性传输进行流水线处理。如果队列大小太小，则仅在队列空间释放时模块才能放入新请求。

  要按调度计划执行周期性传输，模块的周期性请求队列深度至关重要。根据一个微帧内要安排的周期性传输次数，选择周期性队列深度。如果周期性请求队列深度小于一个微帧内要安排的周期性传输数，则会发生帧溢出情况。

- **处理 babble 情况**

  OTG_FS 控制器处理两种情况的 babble：数据包 babble 和端口 babble。如果设备发送的数据量超过通道的最大数据包大小，则会发生数据包 babble。如果模块从 EOF2（非常接近下一帧的 SOF）时刻还在从设备接收数据，则发生端口 babble。

  当 OTG_FS 控制器检测到数据包 babble 时，它停止向 Rx 缓冲区中写入数据并等待数据包结束信号 (EOP)。当该控制器检测到 EOP 时，它会清空 Rx 缓冲区中的已写入数据并对应用程序生成 Babble 中断。

当 OTG_FS 控制器检测到端口 babble 时，它会清空 RxFIFO 并禁止端口。模块随后将生成“端口已禁止”中断（OTG_FS_GINTSTS 中的 HPRTINT 位和 OTG_FS_HPRT 中的 PENCHNG 位）。在收到该中断时，应用程序必须通过检查 OTG_FS_HPRT 中的 POCA 位，来确定该中断并非由过流（“端口已禁止”中断的另一个原因）所致，然后执行软复位。检测到端口 babble 情况后，模块不再发送任何其它令牌。

30.17.5 设备编程模型

USB 复位时的端点初始化

1. 为所有 OUT 端点将 NAK 位置 1
   - OTG_FS_DOEPCTLx 中，SNAK = 1（对于所有 OUT 端点）
2. 取消对以下中断位的屏蔽
   - OTG_FS_DAINTMSK 中，INEP0 = 1（控制 0 IN 端点）
   - OTG_FS_DAINTMSK 中，OUTEP0 = 1（控制 0 OUT 端点）
   - DOEPMSK 中，STUP = 1
   - DOEPMSK 中，XFRC = 1
   - DIEPMSK 中，XFRC = 1
   - DIEPMSK 中，TOC = 1
3. 为每个 FIFO 设置数据 FIFO RAM
   - 对 OTG_FS_GRXFSIZ 寄存器进行编程，以能够接收控制传输的 OUT 数据和设置数据。如果未使能阈值，则该寄存器必须至少等于控制端点 0 的 1 个最大数据包大小 + 2 个字（用于控制 OUT 数据包的状态）+ 10 个字（用于 SETUP 数据包）。
   - 对 OTG_FS_TX0FSIZ 寄存器进行编程（取决于所选的 FIFO 编号），以能够发送控制 IN 数据。该寄存器至少必须等于控制端点 0 的 1 个最大数据包大小。
4. 对端点相关寄存器中的以下字段进行编程，以使控制 OUT 端点 0 接收 SETUP 数据包
   - OTG_FS_DOEPTSIZ0 中的 STUPCNT = 3（接收最多 3 个连续的 SETUP 数据包）

此时，接收 SETUP 数据包所需的所有初始化工作便已完成。

枚举完成时的端点初始化

1. 在枚举完成中断（OTG_FS_GINTSTS 中的 ENUMDNE）中，读取 OTG_FS_DSTS 寄存器以确定设备的枚举速度。
2. 对 OTG_FS_DIEPCTL0 中的 MPSIZ 字段进行编程以设置最大数据包大小。该步骤配置控制端点 0。控制端点的最大数据包大小取决于枚举速度。

此时，设备已准备好接收 SOF 数据包并配置为在控制端点 0 上执行控制传输。

收到 SetAddress 命令时的端点初始化

本节介绍了应用程序在 SETUP 数据包中接收到 SetAddress 命令时必须执行的操作。

1. 使用在 SetAddress 命令中接收到的设备地址来对 OTG_FS_DCFG 寄存器进行编程
2. 对模块进行编程以发出状态阶段的 IN 数据包

收到 SetConfiguration/SetInterface 命令时的端点初始化

本节介绍了应用程序在 SETUP 包中接收 SetConfiguration 或 SetInterface 命令时必须执行的操作。

1. 接收到 SetConfiguration 命令时，应用程序必须对端点寄存器进行编程，以使用新配置中有效端点的特性来配置这些端点寄存器。
2. 接收到 SetInterface 命令时，应用程序必须对命令指定的端点的端点寄存器进行编程。
3. 在先前配置或其它设置中有效的端点在新的配置或其它设置中无效。必须停用这些无效端点。
4. 使用 OTG_FS_DAINTMSK 寄存器使能有效端点的中断，屏蔽无效端点的中断。
5. 为每个 FIFO 设置数据 FIFO RAM。
6. 配置完所有必需的端点后，应用程序必须对模块进行编程以发送状态阶段的 IN 数据包。

此时，设备模块已可以接收和发送任何类型的数据包。

端点激活

本节介绍激活设备端点或者将现有设备端点配置为新类型所需的步骤。

1. 在 OTG_FS_DIEPCTLx 寄存器（对于 IN 或双向端点）或 OTG_FS_DOEPCTLx 寄存器（对于 OUT 或双向端点）的以下字段中，对所需端点的特性进行编程。
   - 最大数据包大小
   - USB 活动端点位置 1
   - 端点初始数据同步位（对于中断和批量端点）
   - 端点类型
   - TxFIFO 编号
2. 激活端点后，模块便开始解码发送到该端点的令牌，并在收到的令牌有效的情况下回复有效握手信号。

端点停用

本节介绍停用现有端点所需的步骤。

1. 在要停用的端点中，将 OTG_FS_DIEPCTLx 寄存器（对于 IN 或双向端点）或 OTG_FS_DOEPCTLx 寄存器（对于 OUT 或双向端点）中的 USB 活动端点位清零。
2. 停用端点后，模块便会忽略发送到该端点的令牌，从而导致 USB 超时。

注意： *应用程序必须满足以下条件才能设置设备模块以处理通信：*
必须将 OTG_FS_GINTMSK 寄存器中的 NPTXFEM 和 RXFLVLM 清零。

30.17.6 操作模型

SETUP 和 OUT 数据传输

本节介绍了数据 OUT 传输和 SETUP 事务期间的内部数据流和应用程序操作步骤。

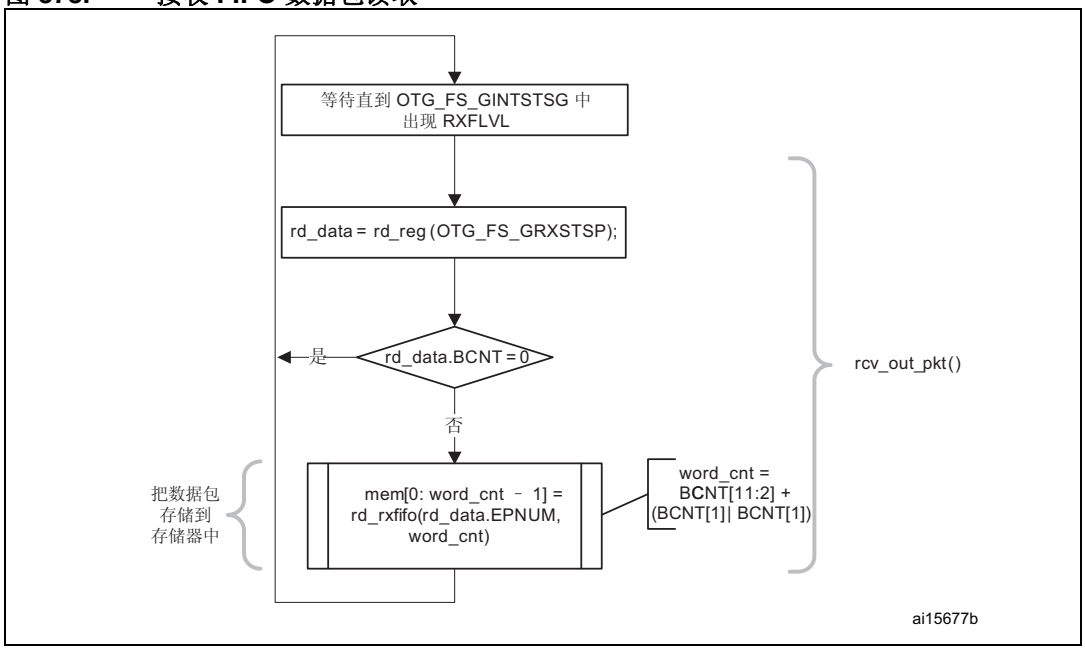
- **数据包读取**

本节介绍如何从接收 FIFO 读取数据包（OUT 数据和 SETUP 数据包）。

1. 捕获到 RXFLVL 中断（OTG_FS_GINTSTS 寄存器）时，应用程序必须读取接收状态弹出寄存器 (OTG_FS_GRXSTSP)。
2. 应用程序可以通过写入 RXFLVL = 0（在 OTG_FS_GINTMSK 中）来屏蔽 RXFLVL 中断（在 OTG_FS_GINTSTS 中），直到它把数据包从接收 FIFO 中读取出来。
3. 如果已接收数据包的字节计数不是 0，则从接收数据 FIFO 中弹出这些数据并存储在存储器中。如果接收到的数据包字节计数为 0，则不会从接收数据 FIFO 中弹出任何数据。
4. 从接收 FIFO 读出的数据包状态有以下几种状态：
   a) 全局 OUT NAK：
   PKTSTS = 全局 OUT NAK，BCNT = 0x000，EPNUM 和 DPID 的值无关紧要。
   这些数据表示全局 OUT NAK 位已生效。
   b) SETUP 数据包：
   PKTSTS = SETUP，BCNT = 0x008，EPNUM = 控制 EP 编号，DPID = D0。这些数据表示指定端点上收到的 SETUP 数据包现在可从接收 FIFO 中读取。
   c) 建立阶段完成：
   PKTSTS = 建立阶段完成，BCNT = 0x0，EPNUM = 控制 EP 编号，DPID 值无关紧要。
   这些数据表示指定端点的建立阶段完成并且数据阶段已启动。在此状态条目从接收 FIFO 中弹出后，模块将在该控制 OUT 端点上产生建立中断。
   d) OUT 数据包：
   PKTSTS = DataOUT，BCNT = 接收的 OUT 数据包的大小 (0 ≤ BCNT ≤ 1024)，EPNUM = 收到数据包的端点编号，DPID = 实际数据 PID。
   e) 数据传输完成：
   PKTSTS = OUT 数据传输完成，BCNT = 0x0，EPNUM = 完成数据传输的 OUT EP 编号，DPID 值无关紧要。
   这些数据表示指定 OUT 端点的 OUT 数据传输完成。在此状态条目从接收 FIFO 中弹出后，模块将在指定的 OUT 端点上引发“传输完成”中断。
5. 从接收 FIFO 中弹出数据后，必须取消对 RXFLVL 中断的屏蔽 (OTG_FS_GINTSTS)。
6. 每次应用程序检测到 OTG_FS_GINTSTS 中的 RXFLVL 中断时，都将重复步骤 1 到 5。读取空的接收 FIFO 可能导致未定义的模块行为。

图 373 提供了上述过程的流程图。

图 373. 接收 FIFO 数据包读取



- **SETUP 事务**

本节介绍了模块处理 SETUP 数据包的方式以及应用程序处理 SETUP 事务的顺序。

- **应用程序要求**

1. 要接收 SETUP 数据包，必须将控制 OUT 端点中的 STUPCNT 字段 (OTG_FS_DOEPTSIZx) 编程为非零值。如果应用程序将 STUPCNT 字段编程为非零值，模块会接收 SETUP 数据包并将其写入接收 FIFO，而不考虑 NAK 状态和 OTG_FS_DOEPCTLx 中的 EPENA 位设置。控制端点每收到一个 SETUP 数据包后， STUPCNT 字段都会递减。如果在接收 SETUP 数据包之前，未将 STUPCNT 字段编程为适当值，模块仍能接收 SETUP 数据包并使 STUPCNT 字段递减，但应用程序可能无法确定在控制传输的建立阶段中接收的 SETUP 数据包正确数量。
   – 在 OTG_FS_DOEPTSIZx 中，STUPCNT = 3
2. 应用程序必须始终在接收数据 FIFO 中分配一些额外空间，以便能够在控制端点上接收连续的最多三个 SETUP 数据包。
   – 预留空间 10 个字。第一个 SETUP 数据包需要 3 个字，“建立阶段完成”状态双字需要 1 个字，还需要 6 个字以存储两个额外的 SETUP 数据包。
   – 每个 SETUP 数据包需要 3 个字以存储 8 个字节的 SETUP 数据和 4 个字节的 SETUP 状态。模块将在接收 FIFO 中保留这些空间。
   – 这段 FIFO 仅用于存储 SETUP 包，绝对不会将该空间用于数据包。
3. 应用程序必须从接收 FIFO 中读取 SETUP 数据包的 2 个字。
4. 应用程序必须从接收 FIFO 中读取并丢弃“建立阶段完成”状态字。

- **内部数据流**

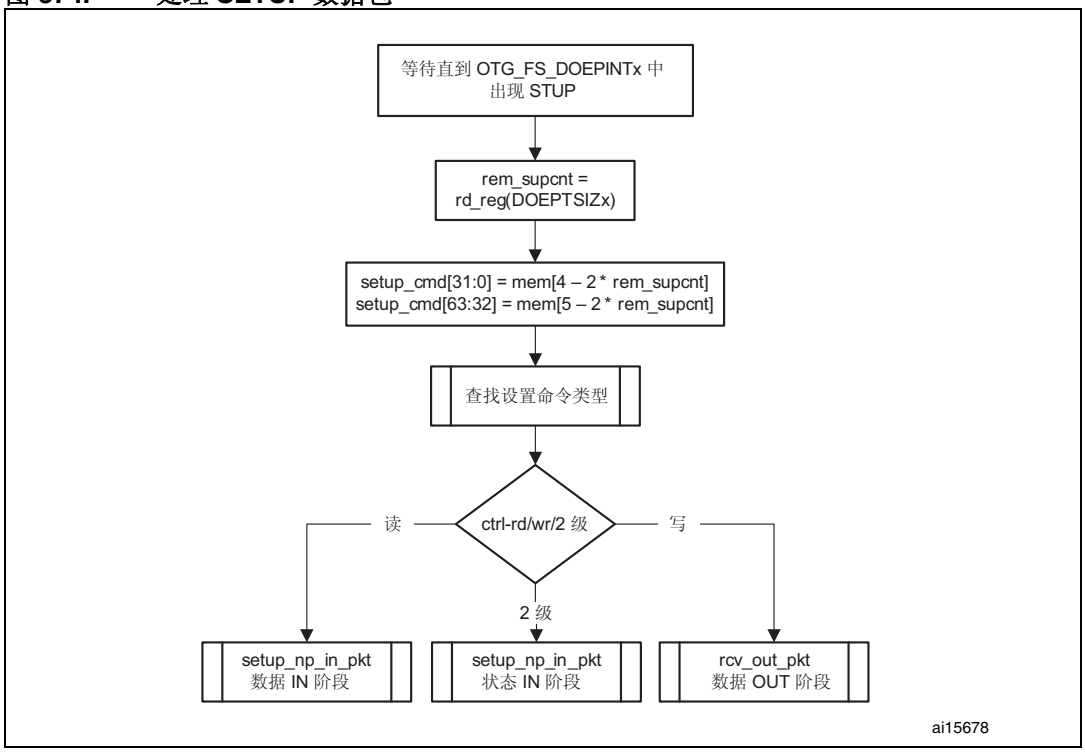
5. 接收到 SETUP 数据包时，模块会将接收到的数据写入接收 FIFO，而不会检查接收 FIFO 中的可用空间，且不考虑端点的 NAK 和 STALL 位设置。
   – 模块会在内部将接收到 SETUP 数据包的控制 IN/OUT 端点的 IN NAK 和 OUT NAK 位置 1。

6. USB 上接收到的每个 SETUP 数据包，模块会将 3 个字的数据写入接收 FIFO，并且将 STUPCNT 字段递减 1。
   – 第一个字包含由模块所使用的内部控制信息
   – 第二个字包含 SETUP 命令的前 4 个字节
   – 第三个字包含 SETUP 命令的最后 4 个字节
7. 当建立阶段结束，数据 IN/OUT 阶段开始时，模块会将一个状态条目（“建立阶段完成”字）写入接收 FIFO，指示建立阶段完成。
8. 在 AHB 端，SETUP 数据包被应用程序读取。
9. 当应用程序从接收 FIFO 中弹出“建立阶段完成”字时，模块将使用 STUP 中断 (OTG_FS_DOEPINTx) 来中断应用程序，指示其可以处理接收到的 SETUP 数据包。
   – 模块会将控制 OUT 端点的端点使能位清零。

- **应用程序编程顺序**

1. 对 OTG_FS_DOEPTSIZx 寄存器进行编程。
   – STUPCNT = 3
2. 等待 RXFLVL 中断 (OTG_FS_GINTSTS) 并且从接收 FIFO 中读取数据包。
3. STUP 中断的触发 (OTG_FS_DOEPINTx) 表示 SETUP 数据传输成功完成。
   – 发生该中断时，应用程序必须读取 OTG_FS_DOEPTSIZx 寄存器以确定接收的 SETUP 数据包数量并处理最后接收的 SETUP 数据包。

图 374. 处理 SETUP 数据包



- **处理三个以上连续的 SETUP 数据包**

根据 USB 2.0 规范，在 SETUP 数据包错误中，主机通常不会向同一个端点发送 3 个以上连续的 SETUP 数据包。但是，USB 2.0 规范并未限制主机可以向同一个端点发送的连续 SETUP 数据包数量。发生这种情况时，OTG_FS 控制器将生成中断（OTG_FS_DOEPINTx 中的 B2BSTUP）。

- **将全局 OUT NAK 置 1**

内部数据流：

1. 如果应用程序将全局 OUT NAK（OTG_FS_DCTL 中的 SGONAK 位）置 1，模块将停止向接收 FIFO 中写入 SETUP 数据包以外的数据。无论接收 FIFO 中可用空间大小如何，设备都会对主机发送的非同步 OUT 令牌回复 NAK，而对同步 OUT 数据包直接予以忽略。
2. 模块将全局 OUT NAK 写入接收 FIFO。应用程序必须为此留出足够空间。
3. 当应用程序从接收 FIFO 中弹出全局 OUT NAK 字时，模块会将 GONAKEFF 中断 (OTG_FS_GINTSTS) 置 1。
4. 应用程序检测到该中断后，会认为模块处于全局 OUT NAK 模式。应用程序可以通过将 OTG_FS_DCTL 中的 SGONAK 位清零来清除该中断。

应用程序编程顺序：

1. 要停止接收任何类型的数据到接收 FIFO 中，应用程序必须通过编程以下字段以将全局 OUT NAK 位置 1。
   – 在 OTG_FS_DCTL 中，SGONAK = 1
2. 等待 OTG_FS_GINTST 中的 GONAKEFF 中断。一旦被触发，该中断表示模块已停止接收 SETUP 数据包以外的任何类型数据。
3. 如果应用程序已将 OTG_FS_DCTL 中的 SGONAK 位置 1，则在模块引发 GONAKEFF 中断 (OTG_FS_GINTSTS) 之前，应用程序可以接收有效 OUT 数据包。
4. 应用程序可通过对 OTG_FS_GINTMSK 寄存器中的 GINAKEFFM 位执行写操作来暂时屏蔽此中断。
   – 在 OTG_FS_GINTMSK 寄存器中，GINAKEFFM = 0
5. 当应用程序准备退出全局 OUT NAK 模式时，必须将 OTG_FS_DCTL 中的 SGONAK 位清零。此操作还会清除 GONAKEFF 中断 (OTG_FS_GINTSTS)。
   – 在 CGONAK 中，OTG_FS_DCTL = 1
6. 如果应用程序在之前已屏蔽此中断，则必须按以下方式取消对该中断的屏蔽：
   – 在 GINTMSK 中，GINAKEFFM = 1

- **禁止 OUT 端点**

应用程序必须使用以下顺序禁止已使能的 OUT 端点。

应用程序编程顺序：

1. 禁止任何 OUT 端点前，应用程序必须在模块中使能全局 OUT NAK 模式。
   – 在 OTG_FS_DCTL 中，SGONAK = 1
2. 等待 GONAKEFF 中断 (OTG_FS_GINTSTS)
3. 通过编程以下字段来禁止 OUT 端点：
   – 在 OTG_FS_DOEPCTLx 中，EPDIS = 1
   – 在 OTG_FS_DOEPCTLx 中，SNAK = 1

4. 等待 EPDISD 中断 (OTG_FS_DOEPINTx)，该中断表示已完全禁止 OUT 端点。引发 EPDISD 中断时，模块还会将以下位清零：
   – 在 OTG_FS_DOEPCTLx 中，EPDIS = 0
   – 在 OTG_FS_DOEPCTLx 中，EPENA = 0
5. 应用程序必须将全局 OUT NAK 位清零，以开始从其它未禁止的 OUT 端点接收数据。
   – 在 OTG_FS_DCTL 中，SGONAK = 0

- **通用非同步 OUT 数据传输**

本节介绍一种常规非同步 OUT 数据传输（控制、批量或中断）。

应用程序要求：

1. 建立 OUT 传输前，应用程序必须在存储器中分配一个缓冲区，以容纳要作为 OUT 传输的一部分而接收的所有数据。
2. 对于 OUT 传输，端点的传输大小寄存器中的传输大小字段必须是端点的最大数据包大小的倍数（且以字对齐）。
   – 传输大小[EPNUM] = $n \times$ (MPSIZ[EPNUM] + 4 – (MPSIZ[EPNUM] mod 4))
   – 数据包计数[EPNUM] = n
   – $n > 0$
3. 发生 OUT 端点中断时，应用程序必须读取端点的传输大小寄存器以计算存储器中有效数据量。接收的有效数据量可能小于编程的传输大小。
   – 存储器中的有效数据量 = 应用程序设置的初始传输量 – 模块更新后的剩余传输量
   – 接收到 USB 数据包数 = 应用程序设置的初始数据包数 – 模块更新后的剩余数据包数

内部数据流：

1. 应用程序必须在端点相关寄存器中设置传输大小和数据包计数字段，将 NAK 位清零，并使能端点来接收数据。
2. NAK 位清零后，模块便开始接收数据并将数据写入接收 FIFO（只要接收 FIFO 中有空间）。对于 USB 上接收的每个数据包，数据包及其状态都会写入接收 FIFO。写入接收 FIFO 的每个数据包（数据量达到最大数据包大小的数据包或短数据包）都会使该端点的数据包计数字段递减 1。
   – 收到的数据包若 CRC 无效，则自动被从接收 FIFO 中清除。
   – 在 USB 上为数据包回复 ACK 后，模块将丢弃主机因无法检测到 ACK 而重新发送的非同步 OUT 数据包。应用程序不会在具有相同数据 PID 的相同端点上检测到多个连续的 OUT 数据包。在这种情况下，数据包计数不会递减。
   – 如果接收 FIFO 中没有空间，则会忽略同步或非同步数据包并且不会将它们写入接收 FIFO。此外，非同步 OUT 令牌将会收到 NAK 握手应答。
   – 在上述所有三种情况中，数据包计数都不会递减，因为没有任何数据写入接收 FIFO。
3. 当数据包计数变为 0 或者在端点上接收到短数据包时，该端点的 NAK 位将置 1。NAK 位置 1 后，将忽略同步或非同步数据包并且不会将它们写入接收 FIFO，同时非同步 OUT 令牌会收到 NAK 握手应答。
4. 在数据写入接收 FIFO 后，应用程序将从接收 FIFO 中读取数据并将数据写入外部存储器，一次一个数据包，逐个端点过来。
5. 在 AHB 上向外部存储器写入完每个数据包后，端点的传输大小都会自动减去该数据包的大小。

6. 在以下情况时，OUT 端点的 OUT 数据传输完成状态将写入接收 FIFO：
   – 传输大小为 0 并且数据包计数为 0
   – 写入接收 FIFO 的最后一个 OUT 数据包是短数据包（0 ≤ 数据包大小 < 最大数据包大小）
7. 当应用程序弹出此状态条目（OUT 数据传输完成），并生成该端点的传输完成中断，同时清零端点使能位。

应用程序编程顺序：

1. 使用传输大小和相应数据包个数对 OTG_FS_DOEPTSIZx 寄存器进行编程。
2. 使用端点特性对 OTG_FS_DOEPCTLx 寄存器进行编程，并将 EPENA 和 CNAK 位置 1。
   – 在 OTG_FS_DOEPCTLx 中，EPENA = 1
   – 在 OTG_FS_DOEPCTLx 中，CNAK = 1
3. 等待 RXFLVL 中断（在 OTG_FS_GINTSTS 中）并且从接收 FIFO 中读走数据包。
   – 此步骤可重复多次，具体取决于传输大小。
4. 触发 XFRC 中断 (OTG_FS_DOEPINTx)，以表示非同步 OUT 数据传输成功完成。
5. 读取 OTG_FS_DOEPTSIZx 寄存器，以确定有效数据量。

- **通用同步 OUT 数据传输**

本节介绍常规的同步 OUT 数据传输。

应用程序要求：

1. 非同步 OUT 数据传输的所有应用程序要求均适用于同步 OUT 数据传输。
2. 对于同步 OUT 数据传输中的传输大小和数据包计数字段，必须始终将其设置为单个帧中可接收的最大数据包大小的数据包数目。同步类型的 OUT 数据传输事务必须在一个帧内完成。
3. 在周期性帧结束（OTG_FS_GINTSTS 中的 EOPF 中断）之前，应用程序必须从接收 FIFO 中读取所有同步 OUT 数据包（数据条目和状态条目）。
4. 要接收下一帧中的数据，必须在 EOPF (OTG_FS_GINTSTS) 之后 SOF (OTG_FS_GINTSTS) 之前使能一个同步 OUT 端点。

内部数据流：

1. 同步 OUT 端点的内部数据流与非同步 OTU 端点的基本相同，但稍有差异。
2. 同步 OUT 端点通过将端点使能位置 1 并将 NAK 位清零来使能时，必须相应地将偶数/奇数帧位置 1。仅当符合以下条件时，模块才会在同步 OUT 端点上接收特定帧中的数据：
   – EONUM（在 OTG_FS_DOEPCTLx 中）= SOFFN[0]（在 OTG_FS_DSTS 中）
3. 当应用程序从接收 FIFO 中完整地读取一个同步 OUT 数据包（数据和状态）时，模块会根据从接收 FIFO 中读取的最后一个同步 OUT 数据包的数据 PID 更新 OTG_FS_DOEPTSIZx 中的 RXDPID 字段。

应用程序编程顺序：

1. 使用传输大小和相应数据包计数对 OTG_FS_DOEPTSIZx 寄存器进行编程
2. 使用端点特性对 OTG_FS_DOEPCTLx 寄存器进行编程，并将端点使能位、清除 NAK 位和奇数/偶数帧位置 1。
   - EPENA = 1
   - CNAK = 1
   - EONUM = （0：偶数/1：奇数）
3. 等待 RXFLVL 中断（在 OTG_FS_GINTSTS 中）并且从接收 FIFO 中读走数据包
   - 此步骤可重复多次，具体取决于传输大小。
4. XFRC 中断（在 OTG_FS_DOEPINTx 中）表示同步 OUT 数据传输完成。该中断不一定意味着存储器中的数据是有效的。
5. 对于同步 OUT 传输，应用程序可能并不总会检测到该中断。相反，应用程序可能检测到 OTG_FS_GINTSTS 中的 IISOOXFRM 中断。
6. 读取 OTG_FS_DOEPTSIZx 寄存器以确定接收的传输大小以及确定帧中接收的数据的有效性。仅当符合以下条件之一时，应用程序才必须将存储器中接收的数据视为有效数据：
   - RXDPID = D0（在 OTG_FS_DOEPTSIZx 中）并且接收该有效数据的 USB 数据包数量 = 1
   - RXDPID = D1（在 OTG_FS_DOEPTSIZx 中）并且接收该有效数据的 USB 数据包数量 = 2
   - RXDPID = D2（在 OTG_FS_DOEPTSIZx 中）并且接收该有效数据的 USB 数据包数量 = 3

     接收该有效数据的 USB 数据包数量 =
     应用程序编程的初始数据包个数 - 模块更新后的剩余数据包个数

   应用程序可将无效数据包丢弃。

- **不完整的同步 OUT 数据传输**

本节介绍了同步 OUT 数据包出现丢包时应用程序编程顺序。

内部数据流：

1. 对于同步 OUT 端点，可能不会始终引发 XFRC 中断（在 OTG_FS_DOEPINTx 中）。如果模块丢弃同步 OUT 数据包，则在以下情况下，应用程序可能无法检测到 XFRC 中断 (OTG_FS_DOEPINTx)：
   - 在接收 FIFO 无法容纳完整的 ISO OUT 数据包时，模块将丢弃接收到的 ISO OUT 数据
   - 接收到的同步 OUT 数据包存在 CRC 错误
   - 模块接收到的同步 OUT 令牌损坏
   - 应用程序从接收 FIFO 中读取数据的速度非常缓慢
2. 如果模块在所有同步 OUT 端点的传输完成前检测到周期性帧结束，将触发未完成同步 OUT 数据中断（OTG_FS_GINTSTS 中的 IISOOXFRM），指示至少有一个同步 OUT 端点上未触发 XFRC 中断（在 OTG_FS_DOEPINTx 中）。此时，未完成传输的端点仍保持使能，但在 USB 的该端点上，没有进行中的有效传输。

应用程序编程顺序：

1. 硬件触发 IISOOXFRM 中断 (OTG_FS_GINTSTS) 表示当前帧中至少有一个同步 OUT 端点具有未完成的传输。
2. 如果因未从端点完全读取同步 OUT 数据而发生这种情况，应用程序必须确保首先从接收 FIFO 读取走所有同步 OUT 数据（包括数据条目和状态条目），然后再继续处理。
   – 从接收 FIFO 读取所有数据后，应用程序即可检测到 XFRC 中断 (OTG_FS_DOEPINTx)。在此情况下，应用程序必须重新使能端点以接收下一个帧中的同步 OUT 数据。
3. 当应用程序接收到 IISOOXFRM 中断（在 OTG_FS_GINTSTS 中）时，应用程序必须读取所有同步 OUT 端点的控制寄存器 (OTG_FS_DOEPCTLx)，以确定哪些端点在当前微帧中具有不完整的传输。同时满足以下两个条件时，表示端点传输未完成：
   – EONUM 位（在 OTG_FS_DOEPCTLx 中）= SOFFN[0]（在 OTG_FS_DSTS 中）
   – EPENA = 1（在 OTG_FS_DOEPCTLx 中）
4. 在检测到 SOF 中断（在 OTG_FS_GINTSTS 中）前，必须执行完成上一步操作，以确保当前帧编号未发生更改。
5. 对于具有不完整传输的同步 OUT 端点，应用程序必须丢弃存储器中的数据，并通过将 OTG_FS_DOEPCTLx 中的 EPDIS 位置 1 来禁止端点。
6. 等待 EPDIS 中断（在 OTG_FS_DOEPINTx 中），并且使能端点以在下一帧中接收新数据。
   – 由于模块可能需要一些时间才能禁止端点，因此应用程序在接收到无效同步数据后，可能无法接收下一个帧中的数据。

- **停止非同步 OUT 端点**

本节介绍应用程序如何才能停止非同步端点。

1. 将模块置于全局 OUT NAK 模式。
2. 禁止所需的端点
   – 禁止端点时，请设置 STALL = 1（在 OTG_FS_DOEPCTL 中），而不是将 OTG_FS_DOEPCTL 中的 SNAK 位置 1。

     STALL 位的优先级始终高于 NAK 位。
3. 当应用程序不再需要端点回复 STALL 握手信号时，必须将 STALL 位（在 OTG_FS_DOEPCTLx 中）清零。
4. 如果应用程序由于收到主机的 SetFeature.Endpoint Halt 或 ClearFeature.Endpoint Halt 命令来设置或清除端点的 STALL 状态，则必须在该控制端点上的状态阶段传输前，将 STALL 位置 1 或清零。

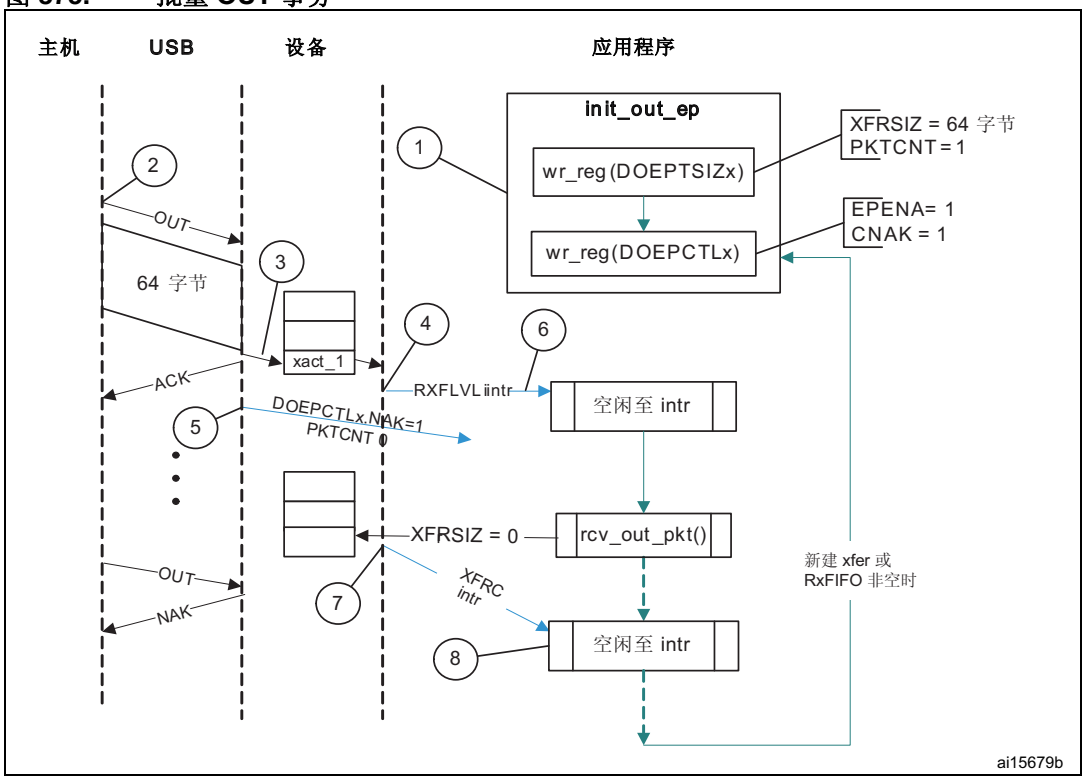
示例

本节介绍并描述了一些基本的传输类型和情景。

- 批量 OUT 事务

图 375 描述了将单个批量 OUT 数据包从 USB 接收到 AHB 中的过程，并介绍了这一过程中涉及到的事件。

图 375. 批量 OUT 事务



接收到 SetConfiguration/SetInterface 命令后，应用程序将初始化所有 OUT 端点：将 CNAK 和 EPENA 置 1（在 OTG_FS_DOEPCTLx 中），并将 OTG_FS_DOEPTSIZx 寄存器中的 XFRSIZ 和 PKTCNT 设置成合适的值。

1. 主机尝试将数据（OUT 令牌）发送到端点。
2. 当模块在 USB 上接收到 OUT 令牌时，模块会将数据包存储在 RxFIFO 中，因为其中有可用空间。
3. 在 RxFIFO 中写入完整数据包后，模块随后会引发 RXFLVL 中断（在 OTG_FS_GINTSTS 中）。
4. 接收到 PKTCNT 所指定数量的 USB 数据包后，模块在内部将该端点的 NAK 位置 1，以避免其再接收任何数据包。
5. 应用程序处理中断并从 RxFIFO 中读取数据。
6. 应用程序读取完所有数据后（相当于 XFRSIZ），模块将生成 XFRC 中断（在 OTG_FS_DOEPINTx 中）。
7. 应用程序处理中断并通过 XFRC 中断的触发得知本次传输完成。

IN 数据传输

- **数据包写入**

本节介绍在已使能专用发送 FIFO 的情况下应用程序如何将数据包写入端点 FIFO。

1. 应用程序可以选择轮询模式或中断模式。
   - 在轮询模式下，应用程序通过读取 OTG_FS_DTXFSTSx 寄存器来监视端点发送数据 FIFO 的状态，从而确定数据 FIFO 中是否有足够空间。
   - 在中断模式下，应用程序等待 TXFE 中断（在 OTG_FS_DIEPINTx 中），然后读取 OTG_FS_DTXFSTSx 寄存器以确定数据 FIFO 中是否有足够空间。
   - 要写入单个非零长度的数据包，数据 FIFO 中必须有足够的空间来容纳整个数据包。
   - 要写入零长度的数据包，应用程序不能查看 FIFO 空间。
2. 如果使用上述方法之一，当应用程序确定有足够空间来写入发送数据包时，应用程序必须首先对端点控制寄存器进行相应写操作，然后再将数据写入数据 FIFO。通常，应用程序必须对 OTG_FS_DIEPCTLx 寄存器执行读-修改-写操作，以避免在将端点使能位置 1 的同时，修改寄存器中的其它内容。

如果有足够空间，应用程序可将同一端点的多个数据包写入发送 FIFO。对于周期性 IN 端点，应用程序只能一次写入一个微帧内的多个数据包。只有先前一个微帧的通信事务传输完成之后，应用程序才会写入下一个微帧内要发送的所有数据包。

- **将 IN 端点 NAK 置 1**

内部数据流：

1. 当应用程序将特定端点的 IN NAK 置 1 时，模块将停止端点上的数据发送，而不考虑端点发送 FIFO 中的数据是否可用。
2. 非同步端点收到 IN 令牌，回复 NAK 握手应答。
   - 同步端点收到 IN 令牌，返回零长度数据包
3. 模块在 OTG_FS_DIEPINTx 中触发 INEPNE（IN 端点 NAK 有效）中断以响应 OTG_FS_DIEPCTLx 中的 SNAK 位。
4. 应用程序检测到该中断后，便会认为端点处于 IN NAK 模式。应用程序可通过将 OTG_FS_DIEPCTLx 中的 CNAK 位置 1 来清除该中断。

应用程序编程顺序：

1. 要在特定 IN 端点上停止发送任何数据，应用程序必须将 IN NAK 位置 1。要将该位置 1，必须编程以下字段。
   - OTG_FS_DIEPCTLx 中的 SNAK = 1
2. 等待 OTG_FS_DIEPINTx 中的 INEPNE 中断触发。该中断表示模块已在端点上停止发送数据。
3. 在应用程序将 NAK 位置 1 但“NAK 有效”中断尚未触发时，模块可以在端点上发送有效 IN 数据。
4. 应用程序可通过写入 DIEPMSK 中的 INEPNEM 位来临时屏蔽该中断。
   - 在 DIEPMSK 中，INEPNEM = 0
5. 要退出端点 NAK 模式，应用程序必须将 OTG_FS_DIEPCTLx 中的 NAK 状态位 (NAKSTS) 清零。此操作还会清除 INEPNE 中断（在 OTG_FS_DIEPINTx 中）。
   - 在 OTG_FS_DIEPCTLx 中，CNAK = 1
6. 如果应用程序已将该中断屏蔽，则必须按以下方式取消屏蔽：
   - 在 DIEPMSK 中，INEPNEM = 1

- **禁止 IN 端点**

使用以下顺序来禁止先前已使能的特定 IN 端点。

应用程序编程顺序：

1. 应用程序必须先停止在 AHB 上写入数据，之后才能禁止 IN 端点。
2. 应用程序必须将端点设置为 NAK 模式。
   – OTG_FS_DIEPCTLx 中的 SNAK = 1
3. 等待 OTG_FS_DIEPINTx 中的 INEPNE 中断。
4. 将必须禁止的端点的 OTG_FS_DIEPCTLx 寄存器中的以下位置 1。
   – OTG_FS_DIEPCTLx 中的 EPDIS = 1
   – OTG_FS_DIEPCTLx 中的 SNAK = 1
5. OTG_FS_DIEPINTx 中的 EPDISD 中断的触发表示模块已完全禁止指定的端点。在触发中断的同时，模块还会将以下位清零：
   – 在 OTG_FS_DIEPCTLx 中，EPENA = 0
   – 在 OTG_FS_DIEPCTLx 中，EPDIS = 0
6. 应用程序必须为周期性 IN EP 读取 OTG_FS_DIEPTSIZx 寄存器，以计算端点上有多少数据是在 USB 上发送的。
7. 应用程序必须通过将 OTG_FS_GRSTCTL 寄存器中的以下字段置 1，来清空端点发送 FIFO 中的数据：
   – TXFNUM（在 OTG_FS_GRSTCTL 中）= 端点发送 FIFO 编号
   – TXFFLSH（在 OTG_FS_GRSTCTL 中）= 1

应用程序必须轮询 OTG_FS_GRSTCTL 寄存器，直至模块将 TXFFLSH 位清零，这表示 FIFO 清空操作结束。要在该端点上发送新数据，应用程序可以在稍后重新使能该端点。

- **通用非周期性 IN 数据传输**

应用程序要求：

1. 建立 IN 传输前，应用程序必须确保组成一次 IN 传输的每个数据包都可以容纳在单个缓冲区中。
2. 对于 IN 传输，端点传输大小寄存器中的传输大小字段表示本次传输的有效数据量，它由多个最大数据包大小和单个短数据包组成。该短数据包在传输结束时发送。
   – 要发送多个最大数据包大小的数据包并在传输结束时外加一个短数据包：
     传输大小[EPNUM] = $x \times$ MPSIZ[EPNUM] + sp
     如果 (sp > 0)，数据包计数[EPNUM] = x + 1。
     否则，数据包计数[EPNUM] = x
   – 要发送单个零长度数据包：
     传输大小[EPNUM] = 0
     数据包计数[EPNUM] = 1
   – 要发送多个最大数据包大小的数据包并在传输结束时外加一个零长度数据包，应用程序必须将传输拆分为两个部分。第一部分发送最大数据包大小的数据包，第二部分仅发送零长度数据包。
     第一次传输：传输大小[EPNUM] = $x \times$ MPSIZ[epnum]；数据包计数 = n；
     第二次传输：传输大小[EPNUM] = 0；数据包计数 = 1；

3. 使能某个端点进行数据传输后，模块会更新传输大小寄存器。在 IN 传输结束时，应用程序必须读取传输大小寄存器，以确定送入发送 FIFO 中的数据已有多少通过 USB 发送出去。
4. 送入发送 FIFO 中的数据量 = 应用程序编程的初始传输大小 - 模块更新后的最终传输大小
   – 通过 USB 已经发送的数据量 = (应用程序编程的初始数据包计数 - 模块更新后的最终数据包计数) × MPSIZ[EPNUM]
   – 要通过 USB 发送的剩余数据量 = (应用程序编程的初始传输大小 - 已通过 USB 发送的数据量)

内部数据流：

1. 应用程序必须在特定端点的寄存器中设置传输大小和数据包计数字段，并使能该端点来发送数据。
2. 应用程序还必须向该端点的发送 FIFO 写入必需的数据。
3. 应用程序每向发送 FIFO 写入一个数据包，该端点的传输大小便会自动减去该数据包大小。应用程序持续从存储器获取数据来写入发送 FIFO，直到该端点的传输大小变为 0。向 FIFO 写入数据后，“FIFO 中的数据包数”计数会递增（这是一个 3 位计数，由模块在内部进行维护，每个 IN 端点发送 FIFO 对应一个。在 IN 端点 FIFO 中，模块所维护的最大数据包数始终为八个）。对于零长度数据包，每个 FIFO 均另有一个单独的标志，FIFO 中没有任何数据。
4. 当数据写入发送 FIFO 后，模块会在接收到 IN 令牌时将这些数据送出。每个数据包发送出去并收到回复的 ACK 握手信号后，该端点的数据包计数都会递减 1，直到数据包计数变 0 为止。发生超时时，数据包计数不会递减。
5. 对于零长度数据包（由内部零长度标志指示），模块会针对 IN 令牌发出一个零长度数据包，并递减数据包计数字段的值。
6. 如果接收到 IN 令牌的端点对应的 FIFO 中无数据，且该端点的数据包计数字段为零，则模块会针对该端点生成一个“TxFIFO 为空时接收到 IN 令牌”(ITTXFE) 中断（前提是该端点的 NAK 位未置 1）。模块在该非同步端点上回复 NAK 握手信号。
7. 模块会在内部使 FIFO 指针重新返回到开头，并且不会生成超时中断。
8. 当传输大小为 0 且数据包计数为 0 时，将生成该端点的传输完成 (XFRC) 中断，同时将端点使能清零。

应用程序编程顺序：

1. 使用传输大小和相应数据包计数对 OTG_FS_DIEPTSIZx 寄存器进行编程。
2. 使用端点特性对 OTG_FS_DIEPCTLx 寄存器进行编程，并将 CNAK 和 EPENA（端点使能）位置 1。
3. 发送非零长度数据包时，应用程序必须轮询 OTG_FS_DTXFSTSx 寄存器（其中 x 为与该端点相关联的 FIFO 编号）以确定数据 FIFO 中是否有足够的空间。写入数据前，应用程序也可选用 TXFE 位（在 OTG_FS_DIEPINTx 中）。

- **通用周期性 IN 数据传输**

本节介绍典型的周期性 IN 数据传输。

应用程序要求：

1. *第 1042 页的通用非周期性 IN 数据传输*的应用程序要求 1、2、3、4 对周期性 IN 数据传输同样适用（只是对要求 2 稍加修改）。
   - 应用程序只能发送若干个最大数据包大小的数据包或若干个最大数据包大小的包，外加传输结束时的一个短数据包。要发送多个最大数据包大小的数据包并在传输结束时外加一个短数据包，必须满足以下条件：

     传输大小[EPNUM] = x × MPSIZ[EPNUM] + sp
     （其中 *x* 是整数 ≥ 0，且 0 ≤ sp < MPSIZ[EPNUM]）

     如果 (sp > 0)，数据包计数[EPNUM] = *x* + 1
     否则，数据包计数[EPNUM] = *x*；

     MCNT[EPNUM] = 数据包计数[EPNUM]
   - 应用程序无法在传输结束时发送零长度数据包。应用程序可以单独发送一个零长度数据包。要发送单个零长度数据包：
   - 传输大小[EPNUM] = 0

     数据包计数[EPNUM] = 1

     MCNT[EPNUM] = 数据包计数[EPNUM]
2. 应用程序一次只能安排一帧的数据传输。
   - (MCNT – 1) × MPSIZ ≤ XFERSIZ ≤ MCNT × MPSIZ
   - PKTCNT = MCNT（在 OTG_FS_DIEPTSIZx 中）
   - 如果 XFERSIZ < MCNT × MPSIZ，则传输的最后一个数据包为短数据包
   - 请注意：MCNT 位于 OTG_FS_DIEPTSIZx 中、MPSIZ 位于 OTG_FS_DIEPCTLx 中、PKTCNT 位于 OTG_FS_DIEPTSIZx 中、XFERSIZ 位于 OTG_FS_DIEPTSIZx 中
3. 接收到 IN 令牌前，应用程序必须将要在帧中发送的完整数据写入到发送 FIFO 中。在接收到 IN 令牌时，即使发送 FIFO 中该帧要发送的数据只差 1 个双字未写进来，模块也会执行 FIFO 为空时的操作。当发送 FIFO 为空时：
   - 同步端点上将回复零长度数据包
   - 中断端点上将回复 NAK 握手信号

内部数据流：

1. 应用程序必须在特定端点的寄存器中设置传输大小和数据包计数字段，并使能该端点来发送数据。
2. 应用程序还必须向与该端点相关联的发送 FIFO 写入必需的数据。
3. 应用程序每向发送 FIFO 写入一个数据包，该端点的传输大小便会自动减去该数据包大小。应用程序持续从存储器获取数据来写入发送 FIFO，直到该端点的传输大小变为 0。
4. 当周期性端点接收到 IN 令牌时，模块将开始发送 FIFO 中的数据（如果 FIFO 中有数据）。如果 FIFO 中没有该帧要发送的数据的完整数据包，则模块将为该端点生成一个“TxFIFO 为空时接收到 IN 令牌”中断。
   - 同步端点上将回复零长度数据包
   - 中断端点上将回复 NAK 握手信号

5. 端点的数据包计数会在下列情况下递减 1：
   – 对于同步端点，发送一个零长度或非零长度的数据包时
   – 对于中断端点，在发送 ACK 握手信号时递减
   – 当传输大小和数据包计数均为 0 时，将生成该端点的传输完成中断，同时将端点使能位清零。
6. 在“周期性帧间隔”（由 OTG_FS_DCFG 中的 PFIVL 位控制）内，当模块发现任何在当前帧内应为空的同步 IN 端点 FIFO 中的数据还未发送完成时，都会在 OTG_FS_GINTSTS 中生成一个 IISOIXFR 中断。

应用程序编程顺序：

1. 使用端点特性对 OTG_FS_DIEPCTLx 寄存器进行编程，并将 CNAK 和 EPENA 位置 1。
2. 将需要在下一帧中发送的数据写入发送 FIFO。
3. 硬件触发 ITTXFE 中断（在 OTG_FS_DIEPINTx 中）表示应用程序尚未将需要发送的全部数据写入发送 FIFO。
4. 如果在检测到中断前已使能中断端点，则将忽略该中断。如果中断端点未使能，则使能此端点，以便数据能够在收到下一次 IN 令牌时发送出去。
5. 硬件触发 XFRC 中断（在 OTG_FS_DIEPINTx 中）时如果 OTG_FS_DIEPINTx 中未产生 ITTXFE 中断，则表示成功完成同步 IN 传输。读取 OTG_FS_DIEPTSIZx 寄存器时始终得到传输大小 = 0 且数据包计数 = 0，则表示所有数据都已通过 USB 发送完毕。
6. 置位 XFRC 中断（在 OTG_FS_DIEPINTx 中）时无论是否产生 ITTXFE 中断（在 OTG_FS_DIEPINTx 中），都表示成功完成中断 IN 传输。读取 OTG_FS_DIEPTSIZx 寄存器时始终得到传输大小 = 0 且数据包计数 = 0，则表示所有数据都已通过 USB 发送完毕。
7. 在 OTG_FS_GINTSTS 中置位未完成的同步 IN 传输 (IISOIXFR) 中断时如果未产生任何前述中断，则表示在当前帧中模块至少未收到 1 个周期性的 IN 令牌。

- **未完成同步 IN 数据传输**

本节介绍应用程序针对未完成同步 IN 数据传输必须执行的操作。

内部数据流：

1. 符合下列条件之一时，即认为同步 IN 传输未完成：
   a) 模块在至少一个同步 IN 端点上接收到损坏的同步 IN 令牌。此时，应用程序检测到未完成同步 IN 传输中断（OTG_FS_GINTSTS 中的 IISOIXFR 位）。
   b) 应用程序向发送 FIFO 写入数据的速度过慢，在将完整数据写入 FIFO 之前便接收到 IN 令牌。此时，应用程序在 OTG_FS_DIEPINTx 中检测到“TxFIFO 为空时接收到 IN 令牌”中断。应用程序可忽略此中断，因为最终这将在周期性帧结束时产生一个未完成同步 IN 传输中断（OTG_FS_GINTSTS 中的 IISOIXFR 位）。

   模块会通过 USB 发送一个零长度数据包来响应接收到的 IN 令牌。
2. 应用程序必须尽快停止向发送 FIFO 写入数据。
3. 应用程序必须将端点的 NAK 位和禁止位置 1。
4. 模块会禁止该端点，将禁止位清零并触发端点的“端点禁止”中断。

应用程序编程顺序：

1. 应用程序可以在任何同步 IN 端点上忽略 OTG_FS_DIEPINTx 中的“TxFIFO 为空时接收到 IN 令牌”中断，因为最终这将产生一个未完成同步 IN 传输中断（在 OTG_FS_GINTSTS 中）。
2. 硬件触发未完成同步 IN 传输中断（在 OTG_FS_GINTSTS 中）表示在至少一个同步 IN 端点上存在未完成的同步 IN 传输。
3. 应用程序必须读取所有同步 IN 端点的“端点控制”寄存器来检测存在未完成 IN 数据传输的端点。
4. 应用程序必须停止向与这些端点相关联的“周期性发送 FIFO”写入数据。
5. 对 OTG_FS_DIEPCTLx 寄存器中的下列字段进行编程以禁止端点：
   - OTG_FS_DIEPCTLx 中的 SNAK = 1
   - OTG_FS_DIEPCTLx 中的 EPDIS = 1
6. 硬件触发 OTG_FS_DIEPINTx 中的“端点禁止”中断表示模块已禁止该端点。
   - 此时，应用程序必须清空相关联的发送 FIFO 中的数据，或者通过在下一微帧中使能新传输的端点来覆盖 FIFO 中的现有数据。要刷新数据，应用程序必须使用 OTG_FS_GRSTCTL 寄存器。

- **停止非同步 IN 端点**

本节介绍应用程序如何才能停止非同步端点。

应用程序编程顺序：

1. 禁止要停止的 IN 端点。同时将 STALL 位置 1。
2. OTG_FS_DIEPCTLx 中的 EPDIS = 1（当端点已使能时）
   - OTG_FS_DIEPCTLx 中的 STALL = 1
   - STALL 位的优先级始终高于 NAK 位
3. 硬件触发“端点禁止”中断（在 OTG_FS_DIEPINTx 中）可以让应用程序知道模块已禁止指定端点。
4. 应用程序必须根据端点类型清空非周期性或周期性发送 FIFO。对于非周期性端点，应用程序必须重新使能另一个无需停止的非周期性端点来发送数据。
5. 当应用程序准备好结束该端点的 STALL 握手信号时，必须将 OTG_FS_DIEPCTLx 的 STALL 位清零。
6. 如果应用程序因收到来自主机的 SetFeature.Endpoint Halt 命令或 ClearFeature.Endpoint Halt 命令来设置或清除端点的 STALL 状态，则必须在该控制端点的状态阶段传输之前将 STALL 位置 1 或清零。

特例：停止控制 OUT 端点

如果在控制传输的数据阶段，主机发送的 IN/OUT 令牌数超过 SETUP 数据包指定的值，则模块必须对这些多余的 IN/OUT 令牌回复 STALL。在这种情况下，应用程序必须在控制传输的数据阶段使能 OTG_FS_DIEPINTx 的 ITTXFE 中断和 OTG_FS_DOEPINTx 的 OTEPDIS 中断（当模块已完成传输 SETUP 数据包指定的数据量后）。随后，当应用程序收到此中断时，必须将相应端点控制寄存器中的 STALL 位置 1 并清除此中断。

30.17.7 最坏情况下的响应时间

当 OTG_FS 控制器作为设备使用时，对于任何跟随在同步 OUT 之后的令牌，都存在一个最坏响应时间。这个最坏情况响应时间随 AHB 时钟频率的不同而异。

模块寄存器位于 AHB 域内，在更新这些寄存器值之前，模块不会接受新令牌。对于任何跟随在同步 OUT 之后的令牌，最坏的情况是：由于同步事务不需要握手信号，所以下一个令牌可能很快就到达。当 AHB 与 PHY 时钟频率相同时，这个最坏情况值为 7 个 PHY 时钟。AHB 时钟越快，此值越小。

如果发生这种最坏情况，模块将以 NAK 响应批量/中断令牌，并丢弃同步和 SETUP 令牌。对于 SETUP，主机会将这种情况视为超时，并尝试重新发送 SETUP 数据包。对于同步传输，会产生未完成同步 IN 传输中断 (IISOIXFR) 和未完成同步 OUT 传输中断 (IISOOXFR)，来通知应用程序同步 IN/OUT 数据包被丢弃。

选择 OTG_FS_GUSBCFG 中 TRDT 的值

TRDT 的值 (OTG_FS_GUSBCFG) 是指在接收到 IN 令牌后以 PHY 时钟计算的时间长度，MAC 将在这段时间内获取 FIFO 状态并从 PFC 模块获取第一个数据。这段时间包含 PHY 和 AHB 时钟之间的同步延迟。最坏情况延迟是当 AHB 时钟与 PHY 时钟相等时的延迟。这种情况下的延迟为 5 个时钟周期。

MAC 接收到 IN 令牌后，此信息（接收到的令牌）将由 PFC（PFC 以 AHB 时钟运行）同步到 AHB 时钟。随后，PFC 从 SPRAM 中读取数据并将它们写入双时钟源缓冲区。MAC 再从源缓冲区读出数据（4 个深度）。

如果 AHB 的运行频率高于 PHY，应用程序可以使用较小的 TRDT 值（在 OTG_FS_GUSBCFG 中）。

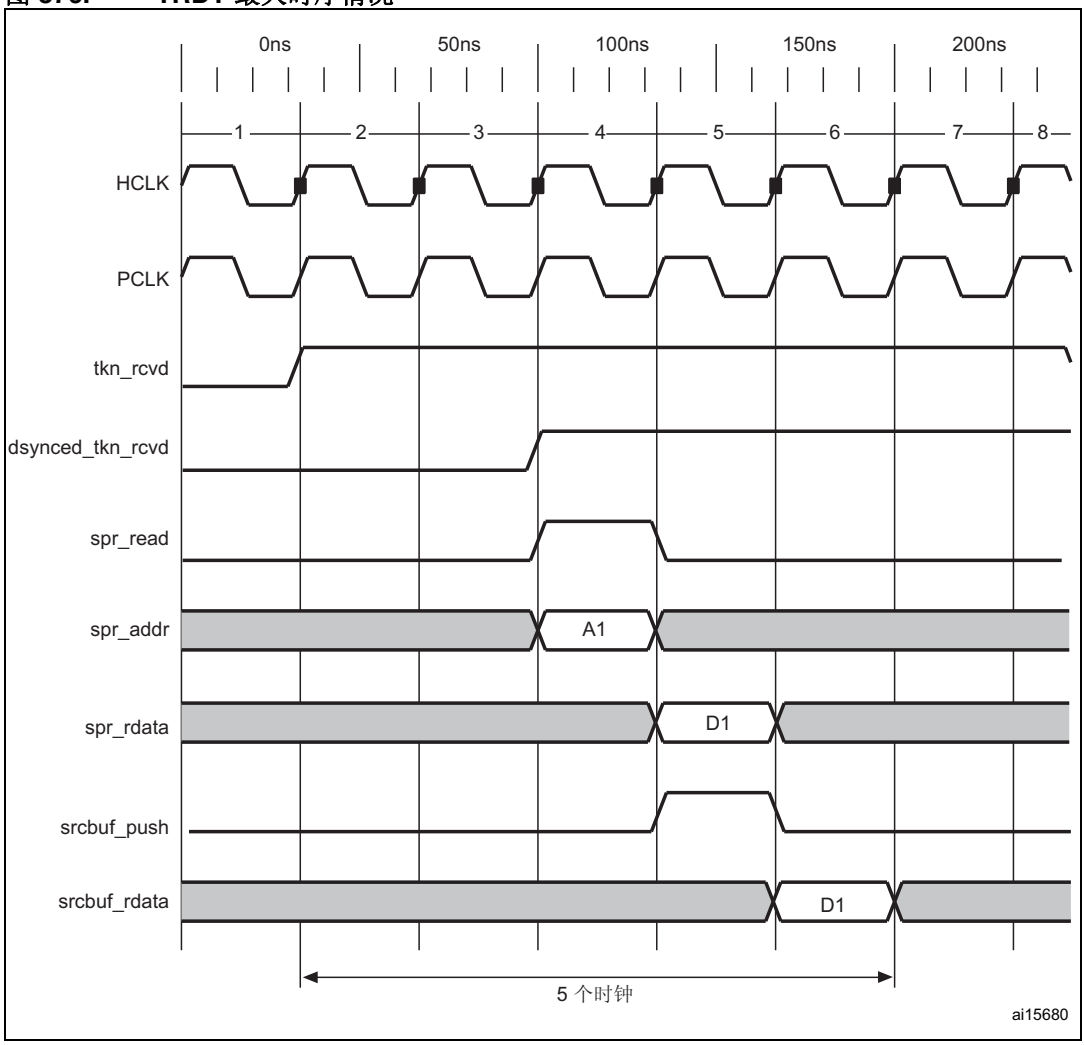
图 376 具有以下信号：

- tkn_rcvd：接收到令牌信息（从 MAC 到 PFC）
- dynced_tkn_rcvd：双倍同步 tkn_rcvd（从 PCLK 到 HCLK 域）
- spr_read：读取到 SPRAM
- spr_addr：寻址到 SPRAM
- spr_rdata：从 SPRAM 中读取数据
- srcbuf_push：压入源缓冲区
- srcbuf_rdata：从源缓冲区读取数据。MAC 看到的数据

应用程序可以使用下列公式来计算 TRDT 的值：

4 × AHB 时钟 + 1 个 PHY 时钟 = (2 个时钟进行同步 + 1 个时钟进行存储器寻址 + 1 个时钟从同步 RAM 读取存储器数据) + 1 个 PHY 时钟（下一个 PHY 时钟 MAC 可以对 2 个时钟 FIFO 输出进行采样）

图 376. TRDT 最大时序情况



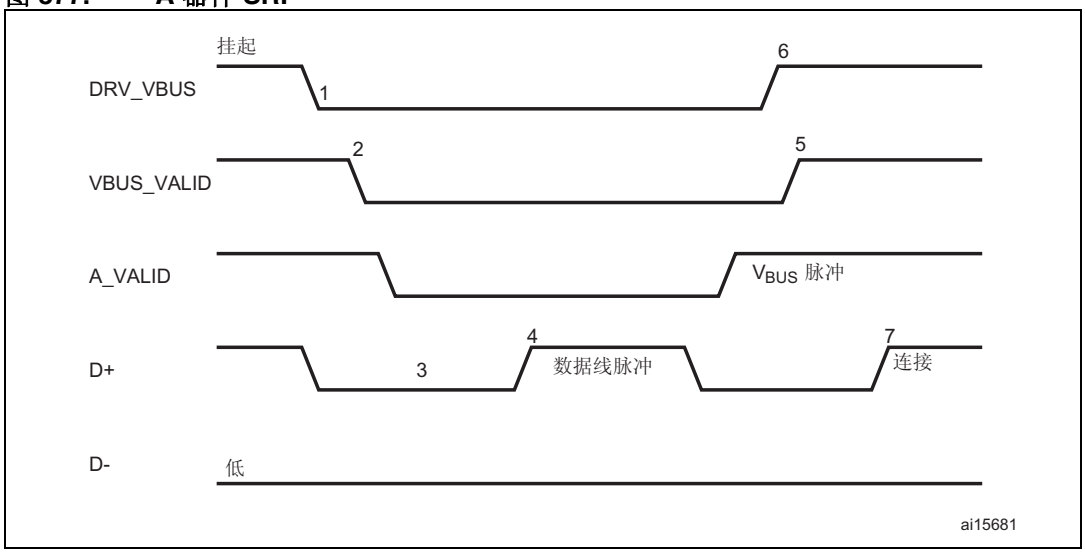
30.17.8 OTG 编程模型

OTG_FS 控制器是一种支持 HNP 和 SRP 的 OTG 设备。该模块接口与“A 型”插头连接时，该模块称为 A 器件。该模块接口与“B 型”插头连接时，该模块称为 B 器件。在主机模式下，OTG_FS 控制器将关闭 V_{BUS} 以节省电能。B 器件可以借助 SRP 请求 A 器件打开 V_{BUS} 电源。设备必须同时执行数据线脉冲和 V_{BUS} 脉冲，但主机可以检测数据线脉冲或者 V_{BUS} 脉冲中的一个用于 SRP。B 器件可以借助 HNP 协商并切换到主机角色。在协商模式下执行 HNP 后，B 器件会挂起总线并恢复到设备角色。

A 器件会话请求协议

应用程序必须将模块 USB 配置寄存器中的 SRP 使能位置 1。这将使能 OTG_FS 控制器在 A 器件模式下检测到 SRP。

图 377. A 器件 SRP



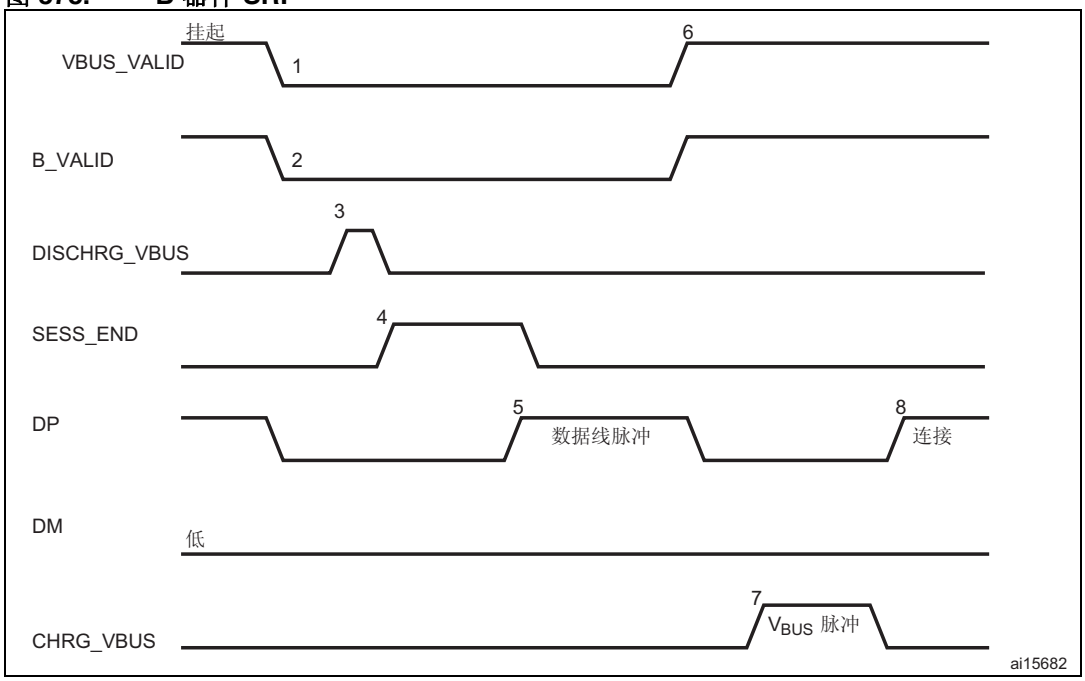
1. DRV_VBUS = 发送到 PHY 的 V_{BUS} 驱动信号
   VBUS_VALID = 来自 PHY 的 V_{BUS} 有效信号
   A_VALID = 发送到 PHY 的 A 器件 V_{BUS} 电平信号
   D+ = 正向数据线
   D- = 负向数据线

1. 为节省电能，在总线空闲时，应用程序会通过在主机端口控制和状态寄存器中写入端口挂起位和端口电源位来挂起设备并关闭端口电源。
2. PHY 通过禁止 VBUS_VALID 信号来指示端口电源已关闭。
3. 当 V_{BUS} 电源关闭时，设备必须检测到至少 2 ms 的 SE0 信号才可以启动 SRP。
4. 要启动 SRP，设备需要打开其数据线的上拉电阻并维持 5 到 10 ms。OTG_FS 控制器会检测到数据线脉冲。
5. 设备会驱动 V_{BUS} 到 A 器件会话有效电平（最小 2.0 V）以上，以产生 V_{BUS} 脉冲。
   OTG_FS 控制器检测到 SRP 时将中断应用程序。在全局中断状态寄存器中，检测到的会话请求位（OTG_FS_GINTSTS 中的 SRQINT 位）将会置 1。
6. 应用程序必须响应“检测到会话请求”中断，并通过在主机端口控制和状态寄存器中写入端口电源位来打开端口电源位。PHY 通过输出 VBUS_VALID 信号来指示端口已通电。
7. 当 USB 通电后，设备将连接，从而完成 SRP 过程。

B 器件会话请求协议

应用程序必须将模块 USB 配置寄存器中的 SRP 使能位置 1。这将使能 OTG_FS 控制器在 B 器件模式下启动 SRP。OTG_FS 控制器可以借助 SRP 向主机请求新会话。

图 378. B 器件 SRP



1. VBUS_VALID = 来自 PHY 的 V_{BUS} 有效信号
   B_VALID = 发送到 PHY 的 B 器件有效会话
   DISCHRG_VBUS = 发送到 PHY 的放电信号
   SESS_END = 发送到 PHY 的会话结束信号
   CHRG_VBUS = 发送到 PHY 的 V_{BUS} 充电信号
   DP = 正向数据线
   DM = 负向数据线

1. 为节省电能，在总线空闲时主机会挂起并关闭端口电源。

   OTG_FS 控制器会在总线空闲 3 ms 后，将模块中断寄存器中的早期挂起位置 1。随后，OTG_FS 控制器会将模块中断寄存器中的 USB 挂起位置 1。

   OTG_FS 控制器会通知 PHY 对 V_{BUS} 进行放电。

2. PHY 指示会话结束。这是 SRP 的初始条件。启动 SRP 之前，OTG_FS 控制器需要 2 ms 的 SE0 信号。

   对于 USB 1.1 全速串行收发器，应用程序必须在 BSVLD 位（位于 OTG_FS_GOTGCTL）被禁止后，等待 V_{BUS} 放电至 0.2 V。此放电时间可从收发器供应商处获取，该值可能因收发器而异。

3. USB OTG 模块通知 PHY 对 V_{BUS} 加速放电。

4. 应用程序通过将 OTG 控制和状态寄存器中的会话请求位置 1 来启动 SRP。OTG_FS 控制器先执行数据线脉冲，随后执行 V_{BUS} 脉冲。

5. 主机会从数据线脉冲或 V_{BUS} 脉冲检测到 SRP，然后打开 V_{BUS}。PHY 指示 V_{BUS} 对设备上电。

6. OTG_FS 控制器执行 V_{BUS} 脉冲。

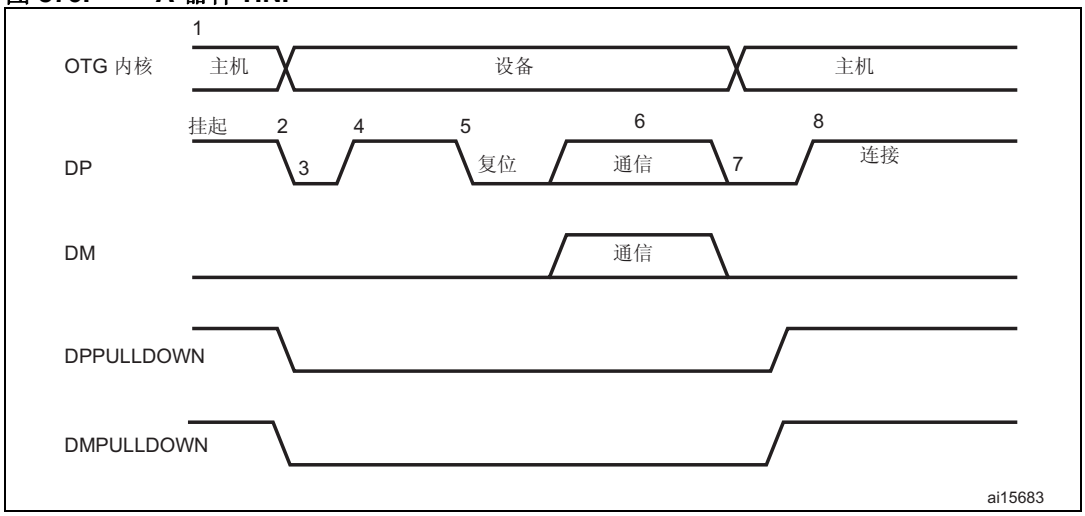
   主机通过打开 V_{BUS} 启动一个新会话，指示 SRP 已成功。OTG_FS 控制器通过将 OTG 中断状态寄存器中的会话请求成功状态变化位置 1 来中断应用程序。应用程序读取 OTG 控制和状态寄存器中的会话请求成功位。

7. 当 USB 上电时，OTG_FS 控制器进行连接，完成 SRP 过程。

A 器件主机协商协议

通过 HNP 可以将 USB 主机角色从 A 器件切换到 B 器件。应用程序必须将模块 USB 配置寄存器中的 HNP 功能位置 1，以使能 OTG_FS 控制器作为 A 器件的 HNP 功能。

图 379. A 器件 HNP



1. DPPULLDOWN = 从模块发送到 PHY 的信号，用于使能/禁止 PHY 内部 DP 线的下拉电阻。
   DMPULLDOWN = 从模块发送到 PHY 的信号，用于使能/禁止 PHY 内部 DM 线的下拉电阻。

1. OTG_FS 控制器发送给 B 器件一个 SetFeature b_hnp_enable 描述符以使能 HNP 支持。B 器件的 ACK 响应表示 B 器件支持 HNP。应用程序必须将 OTG 控制和状态寄存器中的主机设置 HNP 使能位置 1，以向 OTG_FS 控制器指示 B 器件支持 HNP。
2. 应用程序不再使用总线时，会通过在主机端口控制和状态寄存器中写入端口挂起位来挂起总线。
3. B 器件检测到 USB 挂起时，将断开连接，指示 HNP 的初始条件。B 器件只有在切换到主机角色时才会启动 HNP，否则总线会保持挂起状态。

   OTG_FS 控制器将 OTG 中断状态寄存器中检测到的主机协商中断位置 1，指示 HNP 的启动。

   OTG_FS 控制器将禁止 PHY 中 DP 线和 DM 线的下拉，以指示设备角色。PHY 使能 OTG_FS_DP 上拉电阻，以告知 B 器件 A 器件的连接。

   应用程序必须读取 OTG 控制和状态寄存器中的当前模式位，以确定设备工作模式。
4. B 器件检测到连接，发出 USB 复位信号，并枚举 OTG_FS 进行数据通信。
5. B 器件继续担当主机角色，发起数据通信，并在结束时挂起总线。

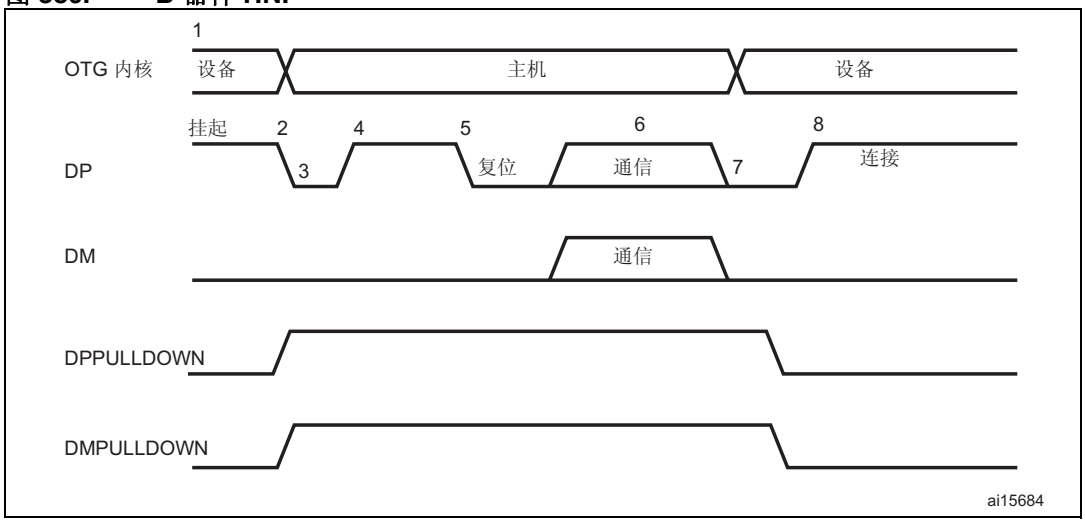
   OTG_FS 控制器会在总线空闲 3 ms 后，将模块中断寄存器中的早期挂起位置 1。随后，OTG_FS 控制器会将模块中断寄存器中的 USB 挂起位置 1。

6. 在协商模式下，OTG_FS 控制器会检测到总线挂起，连接断开，然后切换回主机角色。OTG_FS 控制器将置位 PHY 中 DP 线和 DM 线的下拉，以指示其假设的主机角色。
7. OTG_FS 控制器将 OTG 中断状态寄存器中的连接器 ID 状态变化中断位置 1。应用程序必须读取 OTG 控制和状态寄存器中的连接器 ID 状态位，以确定 OTG_FS 控制器在 A 器件模式下工作。这向应用程序表示 HNP 已经完成。应用程序必须读取 OTG 控制和状态寄存器中的当前模式位，以确定主机工作模式。
8. 连接 B 器件，完成 HNP 过程。

B 器件主机协商协议

通过 HNP 可以将 USB 主机角色从 B 器件切换到 A 器件。应用程序必须将模块 USB 配置寄存器中的 HNP 功能位置 1，以使能 OTG_FS 控制器作为 B 器件的 HNP 功能。

图 380. B 器件 HNP



1. DPPULLDOWN = 从模块发送到 PHY 的信号，用于使能/禁止 PHY 内部 DP 线的下拉电阻。
DMPULLDOWN = 从模块发送到 PHY 的信号，用于使能/禁止 PHY 内部 DM 线的下拉电阻。

1. A 器件发出一个 SetFeature b_hnp_enable 描述符以使能 HNP 支持。OTG_FS 控制器的 ACK 响应表示它支持 HNP。应用程序必须将 OTG 控制和状态寄存器中的设备 HNP 使能位置 1，以指示支持 HNP。

   应用程序将 OTG 控制和状态寄存器中的 HNP 请求位置 1，以向 OTG_FS 控制器指示启动 HNP。
2. A 器件不再使用总线时，会通过在主机端口控制和状态寄存器中写入端口挂起位来挂起总线。

   OTG_FS 控制器会在总线空闲 3 ms 后，将模块中断寄存器中的早期挂起位置 1。随后，OTG_FS 控制器会将模块中断寄存器中的 USB 挂起位置 1。

   OTG_FS 控制器会断开连接，A 器件在总线上检测到 SE0，指示 HNP 即将开始。OTG_FS 控制器将置位 PHY 中 DP 线和 DM 线的下拉，以指示其假设的主机角色。

   在检测到 SE0 3 ms 内，A 器件会通过激活 OTG_FS_DP 上拉电阻进行响应。OTG_FS 控制器检测到此信号时认为有设备接入。

   OTG_FS 控制器将 OTG 中断状态寄存器中的主机协商成功状态变化中断位置 1，指示 HNP 的状态。应用程序必须读取 OTG 控制和状态寄存器中的主机协商成功位，以确定主机协商成功。应用程序必须读取模块中断寄存器 (OTG_FS_GINTSTS) 中的当前模式位，以确定主机工作模式。

3. 应用程序将复位位（OTG_FS_HPRT 中的 PRST 位）置 1，同时 OTG_FS 控制器发出 USB 复位信号，并枚举 A 器件进行数据通信。
4. OTG_FS 控制器继续保持发起通信的主机角色，在通信结束后，会通过将主机端口控制和状态寄存器中的端口挂起位置 1 来挂起总线。
5. 在协商模式下，当 A 器件检测到挂起时，会断开连接，然后切换回主机角色。OTG_FS 控制器将禁止 PHY 中 DP 线和 DM 线的下拉，以指示其假设的设备角色。
6. 应用程序必须读取模块中断寄存器 (OTG_FS_GINTSTS) 中的当前模式位，以确定主机工作模式。
7. OTG_FS 控制器进行连接，完成 HNP 过程。

31 高速 USB on-the-go (OTG_HS)

除非特别说明，否则本部分适用于整个 STM32F4xx 系列器件。

31.1 OTG_HS 简介

本节介绍了 OTG_HS 控制器的架构和编程模型。

本节中使用以下缩写：

FS	全速
HS	高速
LS	低速
USB	通用串行总线
OTG	On-the-go
PHY	物理层
MAC	介质访问控制器
PFC	数据包 FIFO 控制器
UTMI	USB 收发器宏单元接口
ULPI	UTMI+ 引脚数目少的接口

参考文档如下：

- USB On-The-Go 补充标准，第 1.3 版
- 通用串行总线规范第 2.0 版

OTG_HS 是一个双角色设备 (DRD) 控制器，同时支持从机和主机功能，并且完全符合 *USB 2.0 规范的 On-The-Go 补充标准*。此外，该控制器也可配置为仅主机或仅从机控制器，完全符合 *USB 2.0 规范*。在主机模式中，OTG_HS 支持高速（HS，480 Mbits/s）、全速（FS、12 Mbits/s）和低速（LS，1.5 Mbits/s）传输，而在从机模式中，仅支持高速（HS，480 Mbits/s）和全速（FS、12 Mbits/s）传输。OTG_HS 还支持 HNP 和 SRP。OTG 模式下需要的唯一外部设备是提供 VBUS 的电荷泵。

31.2 OTG_HS 主要特性

主要特性可分为三类：通用特性、主机模式特性和从机模式特性。

31.2.1 通用特性

OTG_HS 接口的通用特性如下：

- 经 USB-IF 认证，符合通用串行总线规范 2.0 版本
- 支持 3 个 PHY 接口
  - 片上全速 PHY

  - 连接外部全速 PHY 的 I^2C 接口
  - 连接外部高速 PHY 的 ULPI 接口
- 支持主机协商协议 (HNP) 和会话请求协议 (SRP)
- 在 OTG 应用中允许主机关闭 V_{BUS} 以节省功耗，而不需要外部组件
- 允许使用内部比较器来监视 V_{BUS} 电平
- 支持主机和从机之间的动态角色切换
- 可通过软件配置为以下角色：
  - 支持 SRP 的 USB HS/FS 从机（B 器件）
  - 支持 SRP 的 USB HS/FS/LS 主机（A 器件）
  - USB OTG FS 双角色设备
- 支持 HS/FS SOF 以及低速 (LS)“Keep-alive”令牌并具有如下功能：
  - SOF 脉冲引脚输出功能
  - SOF 脉冲与定时器 2 (TIM2) 的内部连接
  - 可配置的帧周期
  - 可配置的帧结束中断
- 模块内嵌 DMA，并可软件配置 AHB 的突发传输类型
- 具备省电功能，例如在 USB 挂起期间停止系统时钟，关闭数字模块内部时钟域、PHY 和 DFIFO 电源管理
- 具有包含高级 FIFO 管理的专用 4K 字节数据 RAM：
  - 可以将存储区配置为不同 FIFO，以便灵活高效地使用 RAM
  - 每个 FIFO 可包含多个数据包
  - 动态地进行存储器分配
  - FIFO 大小可配置为 2 的幂以外的值，以便连续使用存储区
- 一帧之内可以无需要应用程序干预，以达到最大 USB 带宽

31.2.2 主机模式特性

主机模式下的 OTG_HS 接口特征如下：

- 需要外部电荷泵来生成 V_{BUS}
- 具有多达 12 个主机通道（管道），每个通道可动态地进行重新配置，可支持任何类型的 USB 传输
- 内置硬件调度器：
  - 在周期性硬件队列中存储多达 8 个中断加同步传输请求
  - 在非周期性硬件队列中存储多达 8 个控制加批量传输请求
- 管理一个共享 RX FIFO、一个周期性 TX FIFO 和一个非周期性 TX FIFO，以有效使用 USB 数据 RAM
- 在主机模式下具备对 SOF 帧周期进行动态调校的功能

31.2.3 从机模式特性

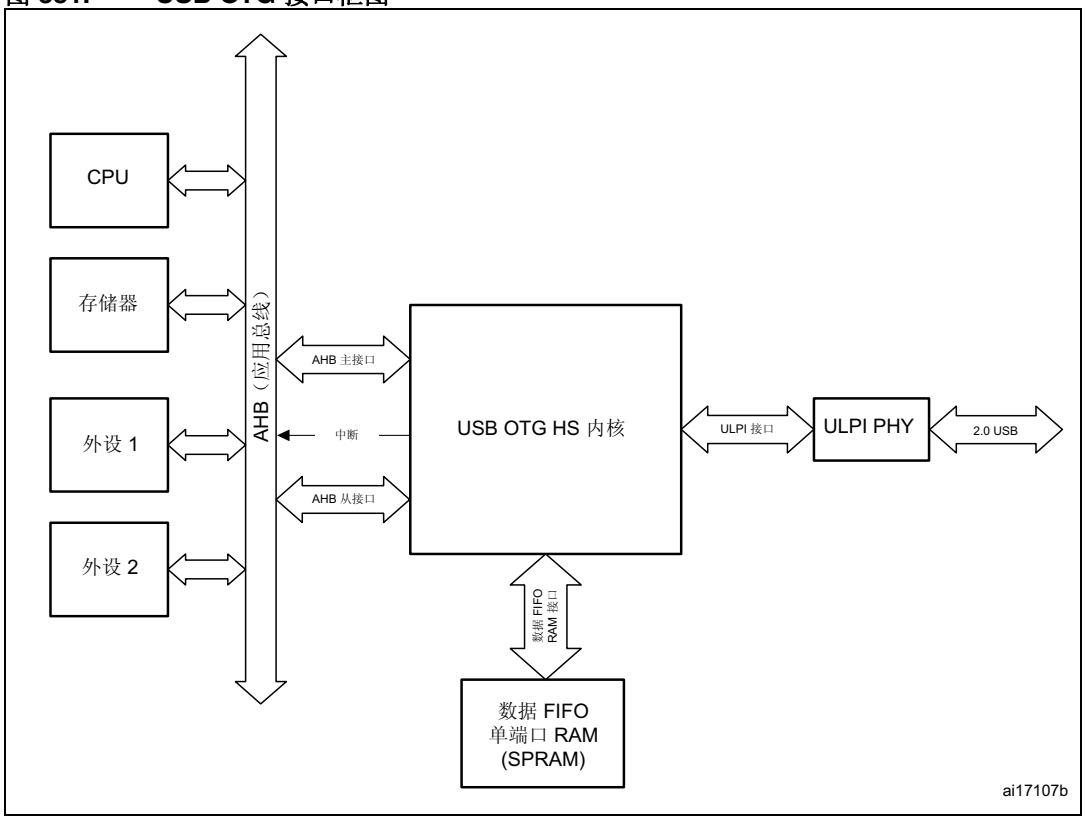
OTG_HS 接口在从机模式下具有以下特性：

- 具有 1 个双向控制端点 0
- 具有 5 个 IN 端点 (EP)，可配置为支持批量、中断或同步传输
- 具有 5 个 OUT 端点，可配置为支持批量、中断或同步传输
- 管理一个共享 Rx FIFO 和一个 Tx-OUT FIFO，可高效使用 USB 数据 RAM
- 管理多达 6 个专用 Tx-IN FIFO（每个 IN 配置的 EP 使用一个）以降低应用程序负载
- 具备软断开功能

31.3 OTG_HS 功能说明

图 381 显示了 OTG_HS 接口框图。

图 381. USB OTG 接口框图



1. USB DMA 无法直接对内部 Flash 进行寻址。

31.3.1 高速 OTG PHY

USB OTG HS 模块嵌入了一个 ULPI 接口以连接外部 HS PHY。

31.3.2 使用 I2C 接口的外部全速 OTG PHY

USB OTG HS 模块嵌入了一个 I^2C 接口，用于连接到外部 FS PHY。

31.3.3 嵌入式全速 OTG PHY

全速 OTG PHY 包括以下组成部分：

- 供主机和设备使用的 FS/LS 收发器模块。直接在单端 USB 线上驱动发送和接收操作。
- 集成 ID 上拉电阻，用于对 ID 线进行采样，以便识别 A/B 器件。
- 由 OTG_HS 模块控制的 DP/DM 集成上拉电阻和下拉电阻，具体使能哪种电阻取决于设备的当前角色。作为外设使用时，只要检测到 V_{BUS} 为有效电平（B 会话有效），立即使能 DP 上拉电阻，以告知全速设备的连接。主机模式下则使能 DP/DM 上的下拉电阻。通过主机协商协议 (HNP) 更改设备角色时，将在上拉电阻和下拉电阻之间动态切换。
- 上拉/下拉电阻 ECN 电路。

  根据适用于 USB 2.0 版的电阻 ECN 规定，DP 上拉电路包括 2 个由 OTG_HS 单独进行控制的电阻。对DP上拉阻值的动态调整可以提高噪声抑制能力和 Tx/Rx 信号质量。
- 带有滞回的 V_{BUS} 感应比较器，用于检测 VBUS_VALID、A-B 会话有效和会话结束电压阈值。执行 USB 操作期间，这些比较器用于驱动会话请求协议 (SRP)、检测会话的有效启动和结束条件，以及持续监视 V_{BUS} 供电情况。
- V_{BUS} 脉冲电路，用于在 SRP 期间通过电阻对 V_{BUS} 充电/放电（驱动力较弱）。

小心： 为确保 USB OTG HS 模块正常工作，AHB 频率应大于 30 MHz。

31.4 OTG 双角色设备

31.4.1 ID 线检测

主机或设备（默认设置）角色取决于 ID 输入线的电平。角色是在插入 USB 电缆时确定，并且取决于 USB 电缆的哪一端连接到 micro-AB 插座：

- 如果 USB 电缆的 B 端连入，其 ID 线悬空，则由于设备在 ID 线上的集成上拉电阻设备将检测到 ID 高电平并确认采取默认的从机角色。在此配置中，OTG_HS 符合USB On-The-Go 补充标准第 1.3 版中第 6.8.2 章节 On-The-Go B 器件中所述的 FSM 标准。
- 如果 USB 电缆的 A 端连入，且 ID 线接地，则 OTG_HS 将发出 ID 线状态更改中断（OTG_HS_GINTSTS 寄存器中的 CIDSCHG 位）以初始化主机软件，且自动切换为主机角色。在此配置中，OTG_HS 符合USB On-The-Go 补充标准第 1.3 版中第 6.8.1 章节 On-The-Go A 器件中所述的 FSM 标准。

31.4.2 HNP 双角色设备

全局 USB 配置寄存器中的 HNP 使能位（OTG_HS_ GUSBCFG 寄存器中的 HNPCAP 位）根据主机协商协议 (HNP)，将 OTG_HS 模块从 A 主机角色动态更改为 A 器件角色（反之亦然），或者从 B 器件角色更改为 B 主机角色（反之亦然）。当前设备状态是由以下两个位的组合来定义：全局 OTG 控制和状态寄存器中的连接器 ID 状态位（OTG_HS_GOTGCTL 中的 CIDSTS 位）与全局中断和状态寄存器中的当前工作模式位（OTG_HS_GINTSTS 中的 CMOD 位）。

*第 31.13 节：OTG_HS 编程模型*中详细介绍了 HNP 编程模型。

31.4.3 SRP 双角色设备

全局 USB 配置寄存器中的 SRP 使能位（OTG_HS_ GUSBCFG 寄存器中的 SRPCAP 位）可使 OTG_HS 模块关闭 V_{BUS}，为 A 器件节省电能。无论 OTG_HS 角色如何（主机或设备），A 器件始终负责提供 V_{BUS}。*第 31.13 节：OTG_HS 编程模型*详细介绍了 SRP A/B 器件编程模型。

31.5 设备模式下的 USB 功能说明

在以下情形下，OTG_HS 作为 USB 设备运行：

- OTG B 器件

  OTG B 器件在插入 USB 电缆 B 端时的默认状态
- OTG A 器件

  OTG A 器件被 HNP 切换为外设角色后的状态
- B 器件

  如果 ID 线有效，设备与 USB 电缆的 B 端相连，而且全局 USB 配置寄存器中的 HNP 功能位（OTG_FS_GUSBCFG 中的 HNPCAP 位）清零（请参见 On-The-Go 规范第 1.3 版的第 6.8.3 节）。
- 仅作设备（请参见*图 359：USB 仅做设备的连接*）

  全局 USB 配置寄存器中的强制外设模式位（OTG_HS_GUSBCFG 中的 FDMOD）置 1，可强制 OTG_HS 模块以 USB 外设模式运行（请参见 On-The-Go 规范第 1.3 版中的第 6.8.3 节）。这种情况下，即使 USB 连接器上存在 ID 线，也会将该 ID 线忽略。

注意： *在B设备模式和仅作外设模式下，若要构建总线供电设备，必须添加一个外部调压器以通过 V_{BUS} 来生成 V_{DD} 电源电压。*

31.5.1 支持 SRP 功能的 USB 设备

全局 USB 配置寄存器中的 SRP 使能位（OTG_HS_GUSBCFG 中的 SRPCAP 位）可使 OTG_HS 支持会话请求协议 (SRP)。因此，远程 A 器件便可在 USB 会话挂起时关闭 V_{BUS} 来节省电能。

*B 器件会话请求协议*一节中详细介绍了 SRP 设备的编程模型。

31.5.2 USB 设备状态

供电状态

V_{BUS} 输入检测到 B 会话有效电压，就会使 USB 设备进入供电状态（请参见 USB 2.0 规范第 9.1 节）。然后，OTG_HS 自动连接 DP 上拉电阻，发出全速设备与主机相连的信号并生成会话请求中断（OTG_HS_GINTSTS 中的 SRQINT 位），指示进入供电状态。此外，V_{BUS} 输入还可确保主机在 USB 操作期间提供有效的 V_{BUS} 电平。如果 V_{BUS} 降至 B 会话有效电压以下（例如，由于发生电源干扰或者主机端口已关闭），则 OTG_HS 自动断开连接，并且生成检测到会话结束中断（OTG_HS_GOTGINT 中的 SEDET 位），指示 OTG_HS 已退出供电状态。

在供电状态下，OTG_HS 会收到来自主机的复位信号。其它 USB 操作则无法执行。收到复位信号，将生成检测到复位中断（OTG_HS_GINTSTS 中的 USBRST）。复位完成时，将生成枚举已完成中断（OTG_HS_GINTSTS 中的 ENUMDNE 位），并且 OTG_HS 进入默认状态。

软断开

供电状态可借助软断开功能通过软件退出。将设备控制寄存器中的软断开位（OTG_HS_DCTL 中的 SDIS 位）置 1 即可移除 DP 上拉电阻，此时尽管没有从主机端口实际拔出 USB 电缆，但主机端仍会生成设备断开检测中断。

默认状态

默认状态下，OTG_HS 会从主机收到 SET_ADDRESS 命令。其它 USB 操作则无法执行。当 USB 上解码出有效 SET_ADDRESS 命令时，应用程序会将相应的数值写入设备配置寄存器中的设备地址字段（OTG_HS_DCFG 中的 DAD 位）。OTG_HS 随即进入地址状态，并准备好以所配置的 USB 地址对主机事务进行应答。

挂起状态

OTG_HS 设备持续监视 USB 活动。如果 USB 处于空闲状态 3 ms，则发出早期挂起中断（OTG_HS_GINTSTS 中的 ESUSP 位）。并在3 ms后通过挂起中断（OTG_HS_GINTSTS 中的 USBSUSP 位）来确认设备进入挂起状态。然后，设备状态寄存器中的设备挂起位（OTG_HS_DSTS 中的 SUSPSTS 位）自动置 1，OTG_HS 随即进入挂起状态。

设备可自行从挂起状态退出。这种情况下，应用程序会将设备控制寄存器中的远程唤醒信号位（OTG_HS_DCTL 中的 RWUSIG 位）置 1，并在 1 ms 到 15 ms 后将其清零。

但若设备检测到主机发出的恢复信号时，将生成恢复中断（OTG_HS_GINTSTS 中的 WKUPINT 位），设备挂起位自动清零。

31.5.3 USB 设备端点

OTG_HS 模块实现了以下 USB 端点：

- 控制端点 0

  该端点是双向的，仅处理控制消息。

  该端点具有单独一组寄存器来处理 IN 和 OUT 事务，还具有专用的控制寄存器 (OTG_HS_DIEPCTL0/OTG_HS_DOEPCTL0)、传输配置寄存器 (OTG_HS_DIEPTSIZ0/OTG_HS_DIEPTSIZ0) 和状态中断寄存器 (OTG_HS_DIEPINTx/OTG_HS_DOEPINT0)。控制和传输大小寄存器中可用的位与其它端点略有不同。

- 5 个 IN 端点
  - 可以将其配置为支持同步、批量或中断传输类型。
  - 它们具备专用的控制寄存器 (OTG_HS_DIEPCTLx)、传输配置寄存器 (OTG_HS_DIEPTSIZx) 和状态中断寄存器 (OTG_HS_DIEPINTx)。
  - 设备 IN 端点通用中断屏蔽寄存器 (OTG_HS_DIEPMSK) 可以在所有 IN 端点（包括 EP0）上使能/禁止单个端点中断源。
  - 支持不完整的同步 IN 传输中断（OTG_HS_GINTSTS 中的 IISOIXFR 位）。只要有一个同步 IN 端点，其传输未在当前帧内完成，则触发该中断。该中断和周期帧结束中断 (OTG_HS_GINTSTS/EOPF) 一起触发。
- 5 个 OUT 端点
  - 可以将其配置为支持同步、批量或中断传输类型。
  - 它们具备专用的控制寄存器 (OTG_HS_DOEPCTLx)、传输配置寄存器 (OTG_HS_DOEPTSIZx) 和状态中断寄存器 (OTG_HS_DOEPINTx)。
  - 设备 OUT 端点通用中断屏蔽寄存器 (OTG_HS_DOEPMSK) 可以在所有 OUT 端点（包括 EP0）上使能/禁止单个端点中断源。
  - 支持不完整的同步 OUT 传输中断（OTG_HS_GINTSTS 中的 INCOMPISOOUT 位）。只要有一个同步 OUT 端点，其传输未在当前帧内完成，则触发该中断。该中断和周期帧结束中断 (OTG_HS_GINTSTS/EOPF) 一起触发。

端点控制

通过设备端点 x IN/OUT 控制寄存器 (DIEPCTLx/DOEPCTLx) 可以对端点进行以下控制：

- 端点使能/禁止
- 在当前配置下激活端点
- 设置 USB 传输类型（同步、批量和中断）
- 设置支持的数据包大小
- 设置与 IN 端点相关的 Tx-FIFO 编号
- 设置希望收到的或发送时要使用到的 data0/data1 PID（仅限批量/中断传输）
- 设置接收或发送事务时所对应的奇/偶帧（仅限同步传输）
- 可以设置 NAK 位，从而不论此时 FIFO 的状态如何，都对主机的请求回复 NAK
- 可以设置 STALL 位，使得主机对该端点的令牌都被硬件回复 STALL
- 可以将 OUT 端点设置为侦听模式，即对接收到的数据不进行 CRC 检查

端点传输

设备端点 x 传输尺寸寄存器 (DIEPTSIZx/DOEPTSIZx) 允许应用程序对传输尺寸参数进行编程并读取传输状态。

必须在端点控制寄存器中的端点使能位置 1 之前完成对此寄存器的设置。

使能端点后，这些字段立即变为只读状态，同时 OTG FS 模块根据当前传输状态对这些字段进行更新。

可对以下传输参数进行编程：

- 以字节为单位的传输大小
- 构成整个传输大小的数据包个数

端点状态/中断

设备端点 x 中断寄存器 (DIEPINTx/DOPEPINTx) 指示端点在出现 USB 和 AHB 相关事件时的状态。当模块中断寄存器中的 OUT 端点中断位或 IN 端点中断位（分别为 OTG_HS_GINTSTS 中的 OEPINT 位或 OTG_HS_GINTSTS 中的 IEPINT 位）置 1 时，应用程序必须读取这些寄存器以获得详细信息。在应用程序读取这些寄存器之前，必须先读取设备全体端点中断 (OTG_HS_DAINT) 寄存器，以获取设备端点 x 中断寄存器的具体端点编号。应用程序必须将此寄存器中的相应位清零，才能将 DAINT 和 GINTSTS 寄存器中的相应位清零。

模块提供以下状态检查和中断产生功能：

- 传输完成中断，指示应用程序 (AHB) 和 USB 端均已完成数据传输
- Setup 阶段已完成（仅针对控制传输类型的 OUT 端点）
- 相关的发送 FIFO 为半空或全空状态（IN 端点）
- NAK 应答已发送到主机（仅针对同步传输的 IN 端点）
- Tx-FIFO 为空时接收到 IN 令牌（仅针对批量和中断传输类型的 IN 端点）
- 尚未使能端点时接收到 OUT 令牌
- 检测到 babble 错误
- 应用程序关闭端点生效
- 应用程序对端点设置 NAK 生效（仅针对同步传输类型的 IN 端点）
- 接收到 3 个以上连续 setup 数据包（仅针对控制类型的 OUT 端点）
- 检测到超时状况（仅针对控制传输类型的 IN 端点）
- 同步传输类型的数据包未产生中断而丢失

31.6 主机模式下的 USB 功能说明

本节介绍了 OTG_HS 在 USB 主机模式下所具有的功能。在以下情形下，OTG_HS 用作 USB 主机：

- OTG A 主机

  OTG A 器件在插入 USB 电缆 A 端时的默认状态
- OTG B 主机

  OTG B 器件被 HNP 切换为主机角色后的状态
- A 器件

  如果 ID 线有效，设备与 USB 电缆的 A 端相连，则全局 USB 配置寄存器中的 HNP 功能位（OTG_HS_GUSBCFG 中的 HNPCAP 位）清零。DP/DM 线上的集成下拉电阻自动使能。
- 仅作主机（*图 360：USB 仅作主机的连接*）。

  全局 USB 配置寄存器中的强制模式位（OTG_HS_GUSBCFG 中的 FHMOD 位）将强制 OTG_HS 模块作为 USB 主机运行。这种情况下，即使 USB 连接器上存在 ID 线，也会将该 ID 线忽略。OTG_HS_FS_DP/OTG_HS_FS_DM 线上的集成下拉电阻自动使能。

注意： *微控制器不能输出 5 V 以提供 V_{BUS}。因此，必须在微控制器以外添加电荷泵或电源开关（如果应用电路板提供 5 V 电源）来驱动 5 V V_{BUS} 线。外部电荷泵可通过任何 GPIO 输出驱动。OTG A 主机、A 器件和仅作主机配置都需要使用电荷泵。*

V_{BUS} 输入可确保电荷泵在 USB 操作期间提供有效的 V_{BUS} 电平，同时电荷泵过流输出可连接到任意配置成外部中断的 GPIO 引脚。在过流 ISR 中必须立即关闭 V_{BUS}。

31.6.1 支持 SRP 功能的 USB 主机

全局 USB 配置寄存器中的 SRP 使能位（OTG_HS_GUSBCFG 中的 SRPCAP 位）可提供 SRP 支持。使能 SRP 功能后，主机可在 USB 会话挂起时通过关闭 V_{BUS} 电源来节省电能。*A 器件会话请求协议*一节中详细介绍了 SRP 主机的编程模型。

31.6.2 USB 主机状态

给主机端口供电

微控制器不能输出 5 V 以提供 V_{BUS}。因此，必须在微控制器以外添加电荷泵或电源开关（如果应用电路板提供 5 V 电源）来驱动 5 V V_{BUS} 线。外部电荷泵可通过任何 GPIO 输出驱动。如果应用通过所选 GPIO 来为 V_{BUS} 供电，则必须将主机端口控制和状态寄存器中的端口电源位（OTG_HS_HPRT 中的 PPWR 位）置 1。

V_{BUS} 有效

在使能 HNP 或 SRP 的情况下，应将 VBUS 感应引脚 (PB13) 连接到 V_{BUS}。V_{BUS} 输入可确保电荷泵在 USB 操作期间提供有效的 V_{BUS} 电平。如果 V_{BUS} 电压意外降至 V_{BUS} 有效阈值 (4.25 V) 以下，将通过会话结束检测位（OTG_HS_GOTGINT 中的 SEDET 位）触发 OTG 中断。在中断处理程序中，应用程序必须关闭 V_{BUS} 电源并将端口电源位清零。

在 HNP 和 SRP 同时关闭的情况下，应断开 VBUS 感应引脚 (PB13) 与 V_{BUS} 之间的连接。该引脚可用作 GPIO。

电荷泵过流标志也可用来防止电气损坏。将电荷泵的过流标志输出连接到任意 GPIO 输入，然后将其配置为出现有效电平时生成端口中断。过流 ISR 必须立即关闭 V_{BUS} 并清零端口电源位。

主机检测到设备连接

如果使能 SRP 或 HNP，即使可以随时连接 USB 外设或 B 器件，但是 OTG_HS 也只有在 V_{BUS} 有效后 (4.75 V) 才能检测到设备的连接。

当 V_{BUS} 处于有效电平且已连接远程 B 器件时，OTG_HS 模块将发出主机端口中断信号，该中断由主机端口控制和状态寄存器中的设备连接位（OTG_HS_HPRT 中的 PCDET 位）触发。

在 HNP 和 SRP 同时关闭的情况下，USB 设备或 B 器件将在连接后立即被检测到。OTG_FS 模块将发出主机端口中断信号，该中断由主机端口控制和状态寄存器中的设备连接位（OTG_FS_HPRT 中的 PCDET 位）触发。

主机检测到设备断开连接

设备断开事件将触发断开连接中断（OTG_HS_GINTSTS 中的 DISCINT 位）。

主机枚举

检测到设备连接后，主机必须通过向新的设备发出 USB 复位和配置命令来启动枚举过程。

发送 USB 复位前，应用程序将等待由“去抖动完成”位（OTG_HS_GOTGINT 中的 DBCDNE 位）触发的 OTG 中断，这表明在 OTG_HS_FS_DP（全速）或 OTG_HS_FS_DM（低速）上连接上拉电阻所导致的电气抖动之后，总线再次稳定。

应用程序将主机端口控制和状态寄存器中的“端口复位”位（OTG_HS_HPRT 中的 PRST 位）置 1，保持 10 ms 到 20 ms，然后将“端口复位”位清零，以发出 USB 复位信号（单端零）。

USB 复位序列完成后，端口使能/禁止更改位（OTG_HS_HPRT 中的 PENCHNG 位）将触发主机端口中断来通知应用程序，指示可以从主机端口控制和状态寄存器中的端口速度字段（OTG_HS_HPRT 中的 PSPD 位）读取所枚举设备的速度，以及主机已经开始驱动 SOF（全速）或“keep-alive”令牌（低速）。此时主机已就绪，可通过对设备发送命令来完成对设备的枚举。

主机挂起

应用程序可通过将主机端口控制和状态寄存器中的端口挂起位（OTG_HS_HPRT 中的 PSUSP）置 1 来挂起 USB 活动。OTG_HS 模块停止发送 SOF 并进入挂起状态。

可由远程设备的自主活动（远程唤醒）使总线退出挂起状态。在此情况下，远程唤醒事件将触发远程唤醒中断（OTG_HS_GINTSTS 中的 WKUPINT 位），把主机端口控制和状态寄存器中的端口恢复位（OTG_HS_HPRT 中的 PRES 位）置 1，并且在 USB 上自动发出恢复信号。应用程序必须监视恢复窗口持续时间，随后必须将端口恢复位清零以退出挂起状态并重新启动 SOF 发送。

如果由主机退出挂起状态，则应用程序必须将端口恢复位置 1 以在主机端口上启动恢复信号，监视恢复窗口持续时间并随后将端口恢复位清零。

31.6.3 主机通道

OTG_HS 模块实现了 12 个主机通道。每个主机通道均可用于 USB 主机传输（USB 管道）。主机最多能同时处理 8 个传输请求。如果应用程序有 8 个以上的传输请求挂起，则通道可用后（也就是在收到“传输完成”和“通道停止”中断后），主机控制器驱动程序 (HCD) 必须为未处理的传输请求重新分配通道。

每个主机通道都可配置为支持输入/输出以及周期性/非周期性事务。每个主机通道都具有专用的控制寄存器 (HCCHARx)、传输配置寄存器 (HCTSIZx) 状态/中断寄存器 (HCINTx) 以及和其相关的中断屏蔽寄存器 (HCINTMSKx)。

主机通道控制

通过主机通道 x 特性寄存器 (HCCHARx) 可以对主机通道作以下控制：

- 通道使能/禁止
- 设置目标 USB 设备的速度：HS/FS/LS
- 设置目标 USB 设备的地址
- 设置与该通道通信的目标 USB 设备上的端点的编号
- 设置该通道上的传输方向：IN/OUT
- 设置该通道上的 USB 传输的类型：控制/批量/中断/同步
- 设置与该通道通信的设备端点的最大包长
- 设置要进行周期传输的帧：奇帧/偶帧

主机通道传输

主机通道传输尺寸寄存器 (HCTSIZx) 允许应用程序对传输尺寸参数进行编程并读取传输状态。

必须在主机通道特性控制寄存器中的通道使能位置 1 之前完成对此寄存器的设置。使能端点后，数据包计数字段立即变为只读状态，同时 OTG HS 模块根据当前传输状态对该字段进行更新。

可对以下传输参数进行编程：

- 以字节为单位的传输大小
- 构成整个传输大小的数据包个数
- 初始数据 PID

主机通道状态/中断

主机通道 x 中断寄存器 (HCINTx) 指示端点在出现 USB 和 AHB 相关事件时的状态。当中断寄存器中的主机通道中断位（OTG_HS_GINTSTS 中的 HCINT 位）置 1 时，应用程序必须读取这些寄存器以获得详细信息。在读取这些寄存器之前，应用程序必须先读取主机全体通道中断 (HCAINT) 寄存器，以获取主机通道 x 中断寄存器的通道编号。应用程序必须将此寄存器中的相应位清零，才能将 HAINT 和 GINTSTS 寄存器中的相应位清零。OTG_HS_HCINTMSK-x 寄存器还提供每个通道各中断源的屏蔽位。

主机模块提供以下状态检查和中断产生功能：

- 传输完成中断，指示应用程序 (AHB) 和 USB 端均已完成数据传输
- 通道因传输完成、USB 事务错误或应用程序发出禁止命令而停止
- 相关的发送 FIFO 为半空或全空状态（IN 端点）
- 接收到 ACK 响应
- 接收到 NAK 响应
- 接收到 STALL 响应
- 由于 CRC 校验失败、超时、位填充错误和错误的 EOP 导致 USB 事务错误
- 串扰错误
- 帧上溢
- 用于数据同步的翻转位出错

31.6.4 主机调度器

主机模块内置硬件调度器，可自主对应用程序发出的 USB 事务请求重新排序和管理。每一帧开始时，主机都先执行周期性（同步和中断）事务，然后执行非周期性（控制和批量）事务，以符合 USB 规范对同步和中断传输高优先级的保证。

主机通过请求队列（一个周期性请求队列和一个非周期请求队列）处理 USB 事务。每个请求队列最多可存储 8 个条目。每个条目代表一个应用程序发起但还未得到响应的 USB 事务请求，并存储了执行该 USB 事务所用到的 IN 或 OUT 通道的编号，以及其它相关信息。USB 事务请求在队列中的写入顺序决定了事务在 USB 接口上的执行顺序。

每一帧开始时，主机都先处理周期性请求队列，然后处理非周期性请求队列。如果当前帧结束时，计划在当前帧执行的同步或中断类型的 USB 传输事务请求仍处于挂起状态，则主机将发出未完成周期性传输中断（OTG_HS_GINTSTS 中的 IPXFR 位）。OTG HS 模块负责对周期性和非周期性请求队列的管理。周期性发送 FIFO 和队列状态寄存器 (HPTXSTS) 与非周期性发送 FIFO 和队列状态寄存器 (HNPTXSTS) 都为只读寄存器，应用程序可使用它们来读取各请求队列的状态。其中包括：

- 周期性（非周期性）请求队列中当前可用的空闲条目数（最多 8 个）
- 周期性（非周期性）Tx-FIFO（OUT 事务）中当前可用的空闲空间
- IN/OUT 令牌、主机通道编号和其它状态信息。

由于每个请求队列最多可存储 8 个 USB 事务请求，因此应用程序可以把主机 USB 事务请求提前发送给调度器；实际的通信最晚会在调度器处理完已挂起的 8 个周期事务和 8 个非周期事务完成之后出现在 USB 总线上。

要向主机调度器（队列）发出事务请求，应用程序必须读取 OTG_HS_HNPTXSTS 寄存器中的 PTXQSAV 位或 OTG_HS_HNPTXSTS 寄存器中的 NPTQXSAV 位，确保周期性（非周期性）请求队列中至少有一个可用空间来存储当前请求。

31.7 SOF 触发

OTG HS 模块在主机和设备模式下都可以监视、跟踪和配置 SOF 帧。并且还具备 SOF 脉冲输出功能。

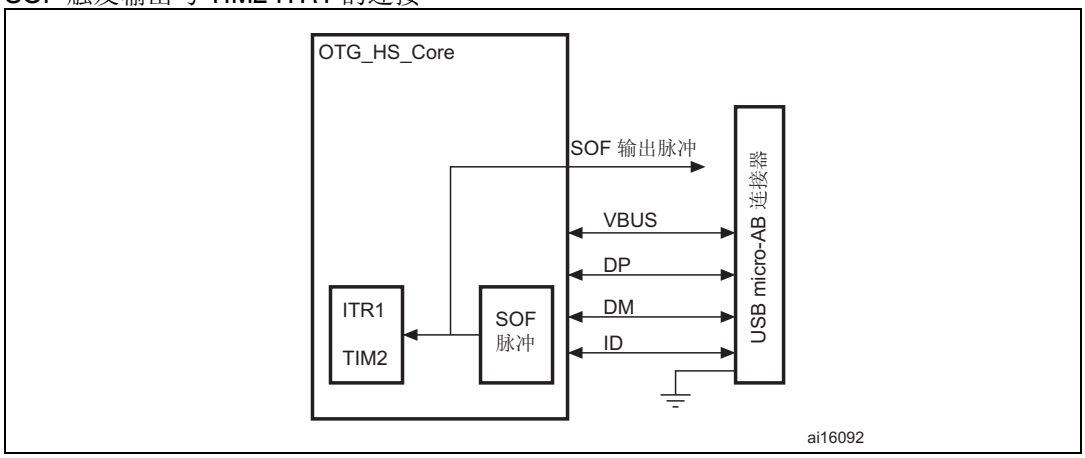
此功能尤其适用于自适应音频时钟生成，其中音频设备需要与 PC 提供的同步音频数据流实现同步，或者主机需要根据音频设备的要求调整数据帧率。

31.7.1 主机 SOF

主机模式下，可以在主机帧间隔寄存器 (OTG_HS_HFIR) 中对所产生的两个连续 SOF (FS) 或 keep-alive (LS) 令牌期间所出现的 PHY 时钟数进行编程，进而应用程序可对 SOF 帧周期进行控制。帧开始（OTH_HS_GINTSTS 中的 SOF 位）时都将生成中断。当前帧编号和出现下一个 SOF 前剩余的时间应用程序在主机帧编号寄存器 (OTG_HS_HFNUM) 中能够进行跟踪。

SOF 令牌发出的同时会产生 SOF 脉冲信号，并且宽度为 12 个系统时钟周期。可以通过全局控制和配置寄存器中的 SOFOUTEN 位，把 SOF 脉冲从 SOF 引脚输出。此外，SOF 脉冲还与片上定时器 2 (TIM2) 的输入触发相连，因此可通过 SOF 脉冲触发输入捕获功能、输出比较功能和定时器。SOF 脉冲与 TIM2 的连接通过 TIM2_OR 寄存器的 ITR1_RMP 位使能。

SOF 触发输出与 TIM2 ITR1 的连接



31.7.2 设备 SOF

在设备模式下，USB 每次接收到 SOF 令牌时，都将触发帧开始中断（OTH_HS_GINTSTS 中的 SOF 位）。相应的帧编号可从设备状态寄存器（OTG_HS_DSTS 中的 FNSOF 位）读取。使用全局控制和配置寄存器中的 SOF 输出使能位（OTG_HS_GCCFG 中的 SOFOUTEN）位，还可以生成宽度为 12 个系统时钟周期的 SOF 脉冲信号，并使该信号在 SOF 引脚输出，以实现外部可用。此外，SOF 脉冲信号还在内部与 TIM2 的输入触发相连，因此可通过 SOF 脉冲触发输入捕获功能、输出比较功能和定时器（请参见*图*）。SOF 脉冲与 TIM2 的连接通过 TIM2_OR 寄存器的 ITR1_RMP 位使能。

周期性帧结束中断 (GINTSTS/EOPF) 用于在经过了 80%、85%、90% 或 95% 的帧间隔时间时通知应用程序，具体取决于设备配置寄存器中的周期性帧间隔字段（OTG_HS_DCFG 中的 PFIVL 位）。

此功能可用于确定该帧的所有同步通信是否完成。

31.8 USB_HS 功耗模式

OTG PHY 的功耗由通用模块配置寄存器中的以下三个位控制：

- PHY 掉电 (GCCFG/PWRDWN)

  该位可开启/关闭 PHY 全速收发器模块。先置位后才允许后续的 USB 操作。

- A-VBUS 感应使能 (GCCFG/VBUSASEN)

  该位可开启/关闭与 A 器件操作相关联的的 V_{BUS} 比较器。必须在 A 器件（USB 主机）模式和 HNP 期间进行设置。

- B-VBUS 感应使能 (GCCFG/VBUSBSEN)

  该位可开启/关闭与 B 器件操作相关联的的 V_{BUS} 比较器。必须在 B 器件（USB 设备）模式和 HNP 期间进行设置。

  USB 会话没有开始或设备未连接时，可以在 USB 挂起状态下使用功率降低技术。

- 停止 PHY 时钟（OTG_HS_PCGCCTL 中的 STPPCLK 位）
  - 将时钟门控控制寄存器中的停止 PHY 时钟位置 1 时，OTG 高速模块的大部分内部时钟域都将由时钟门控关闭。即使应用程序仍提供时钟输入，也会节省掉模块由于时钟信号翻转带来的动态功耗。
  - 还会关掉收发器的大部分单元，只有负责检测异步恢复事件或远程唤醒事件的部分还保持工作状态。

- HCLK 门控（OTG_HS_PCGCCTL 中的 GATEHCLK 位）

  将时钟门控控制寄存器中的 Gate HCLK 位置 1 时，OTG_HS 模块内部的大多数系统时钟域均由时钟门控关闭。只有寄存器读取和写入接口保持活动状态。即使应用程序仍提供时钟输入，也会节省掉模块由于时钟信号翻转带来的动态功耗。

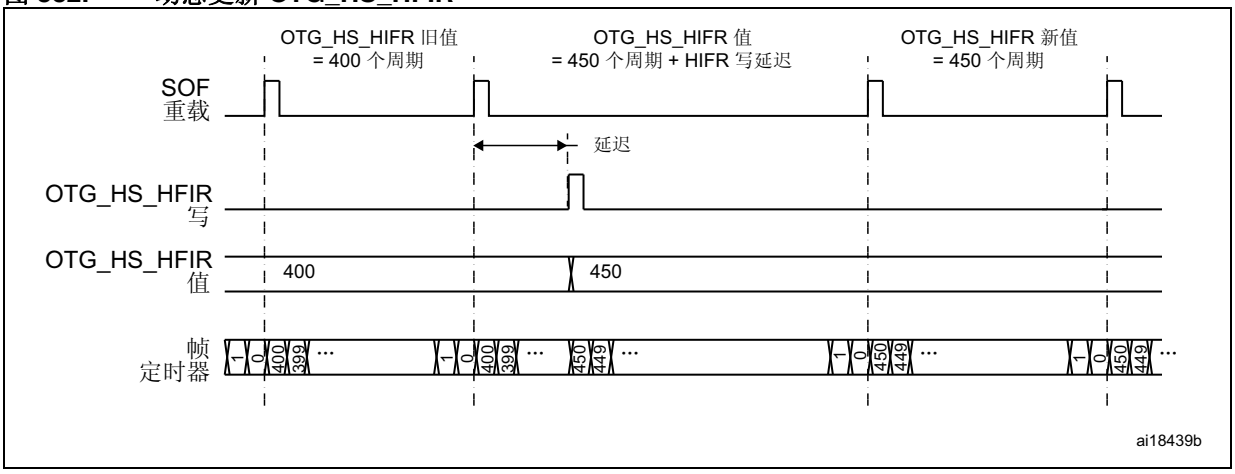
- USB 系统停止
  - 当 OTG_HS 处于 USB 挂起状态时，应用程序可通过关闭 USB 系统中的所有时钟源来大幅降低总体功耗。USB 系统停止可通过以下方式激活：首先将停止 PHY 时钟位置 1，然后在电源控制系统模块 (PWR) 中将系统配置为深度睡眠模式。
  - OTG_HS 模块通过对 USB 上的远程唤醒（作为主机）或恢复（作为设备）信号进行异步检测，自动重新激活系统时钟和 USB 时钟。

31.9 动态更新 OTG_HS_HFIR 寄存器

主机模式下，USB 模块具有对微帧周期进行动态微调的功能，能够将外部设备与 micro-SOF 帧进行同步。

如果 OTG_HS_HFIR 寄存器在当前 micro-SOF 帧内发生更改，则将在下一个帧中对 SOF 周期进行相应修正，具体说明请参见*图 382*。

图 382. 动态更新 OTG_HS_HFIR



31.10 FIFO RAM 分配

31.10.1 设备模式

接收 FIFO RAM

对于接收 FIFO RAM，应用程序应为 SETUP 数据包分配 RAM：接收 FIFO 中必须保留 10 个位置以在控制端点上接收 SETUP 数据包。这些位置将为 SETUP 数据包保留，OTG 模块不会向这些位置来写入任何其它数据。

必须为全局 OUT NAK 分配一个位置。状态信息也将与收到的每个数据包一起写入到 FIFO 中。因此，必须至少为接收数据包分配（最大数据包大小 / 4）+ 1 的空间。如果使能了高带宽端点或多个同步端点，则必须至少分配两个（最大数据包的四分之一）+ 1 的空间，以接收连续数据包。通常，推荐的空间为（最大数据包 / 4 + 1）的两倍，这样当上一个数据包向 AHB 传送时，USB 可同时接收后续的数据包。

传输完成状态信息和该端点收到的最后一个数据包会一起被推入 FIFO。通常情况下，推荐为每个 OUT 端点分配一个位置。

发送 FIFO RAM

对于发送 FIFO RAM，每个 IN 端点发送 FIFO 需要的最小 RAM 空间是该 IN 端点的最大数据包大小。

注意： *为发送 IN 端点 FIFO 分配的空间越多，USB 的性能就越高。*

31.10.2 主机模式

接收 FIFO RAM

对于接收 FIFO RAM 分配，状态信息将与收到的每个数据包一起写入到 FIFO。因此，必须至少为接收数据包分配（最大数据包大小 / 4）+ 1 的空间。如果使能了高带宽通道或多个同步通道，则必须至少分配两个（最大数据包的四分之一）+ 1 的空间，以接收连续数据包。通常，推荐的空间为（最大数据包 / 4 + 1）的两倍，这样当上一个数据包向 AHB 传送时，USB 可同时接收后续的数据包。

传输完成状态信息和该通道收到的最后一个数据包会一起被推入 FIFO。因此，必须分配一个位置以存储此数据。

发送 FIFO RAM

对于发送 FIFO RAM 分配，主机非周期发送 FIFO 所需要的最小 RAM 量是所有非周期性 OUT 通道中的最大数据包大小。通常，推荐的空间为最大数据包大小的两倍，这样当 USB 正在发送当前数据包的同时，AHB 可以往发送 FIFO 填入下一个数据包。

主机周期性发送 FIFO 所需要的最小 RAM 为所有周期性 OUT 通道中的最大数据包大小。如果至少有一个高带宽同步 OUT 端点，则空间必须至少为该通道中最大数据包大小的两倍。

注意： *为非周期性发送 FIFO 分配的空间越多，USB 的性能就越高。*

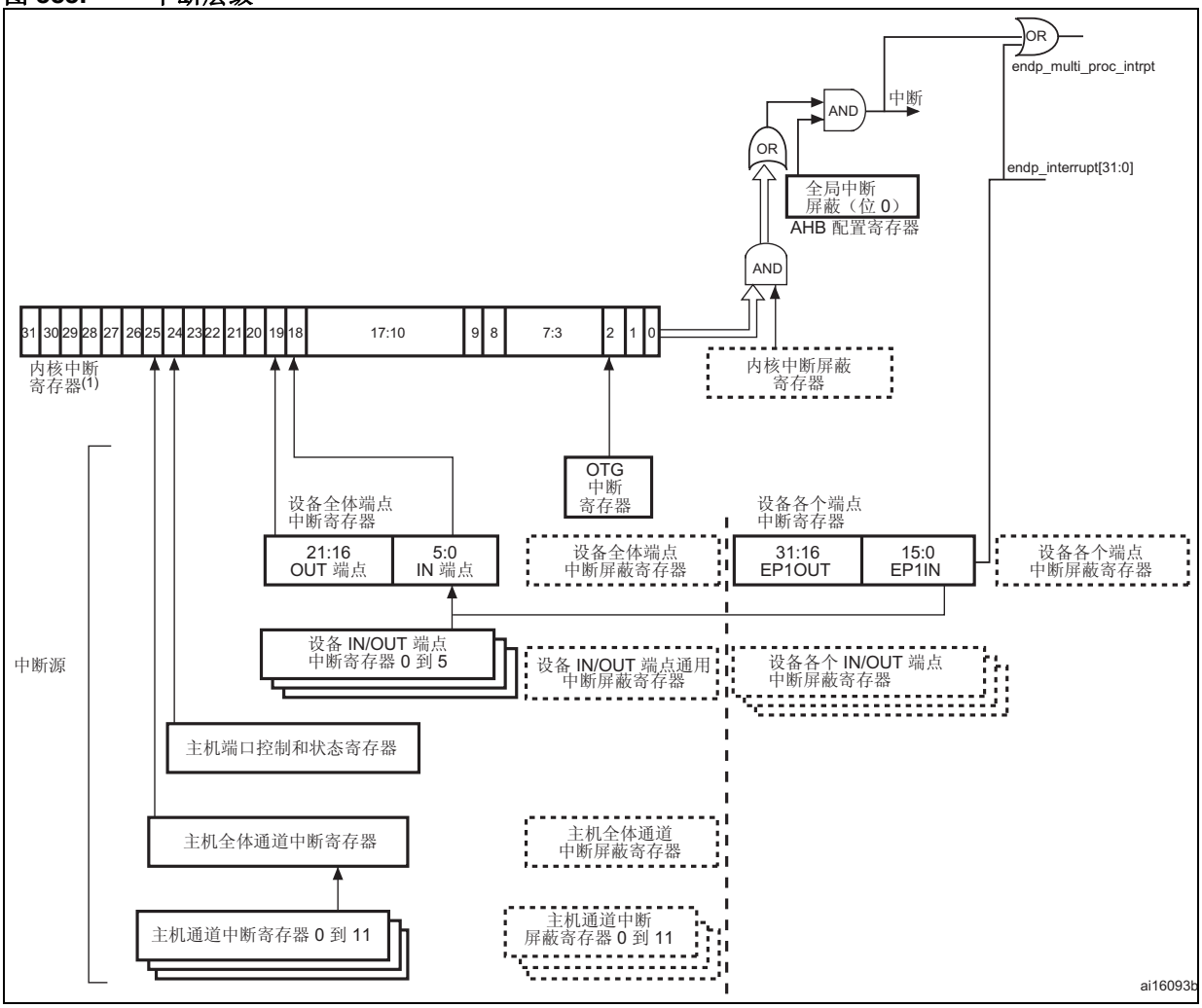
以 DMA 模式运行时，每个主机通道的 DMA 地址寄存器 (HCDMAn) 将存储在 SPRAM (FIFO) 中。为此需要在 FIFO 中为每个通道保留一个位置存放其地址寄存器。

31.11 OTG_HS 中断

当 OTG_HS 控制器在一种模式（设备模式或主机模式）下工作时，应用程序不得以另一种角色模式访问寄存器。如果发生了非法访问，将会产生模式不匹配中断并在模块中断寄存器（OTG_HS_GINTSTS 寄存器中的 MMIS 位）中反映。当模块从一种角色模式切换到另一种角色模式时，新工作模式下的寄存器必须重新编程为上电复位后的状态。

图 383 显示了中断层级。

图 383. 中断层级



1. 有关模块中断寄存器位，请参见*第 1081 页的 OTG_HS 模块中断寄存器 (OTG_HS_GINTSTS)*。

31.12 OTG_HS 控制和状态寄存器

应用程序通过 AHB 从接口对控制和状态寄存器 (CSR) 进行读写操作，以此来控制 OTG_HS 模块。这些都是 32 位寄存器，其地址按 32 位对齐，因此只能以 32 位的方式访问。CSR 分为以下几类：

- 模块全局寄存器
- 主机模式寄存器
- 主机全局寄存器
- 主机端口 CSR
- 主机通道相关寄存器
- 设备模式寄存器
- 设备全局寄存器
- 设备端点相关寄存器

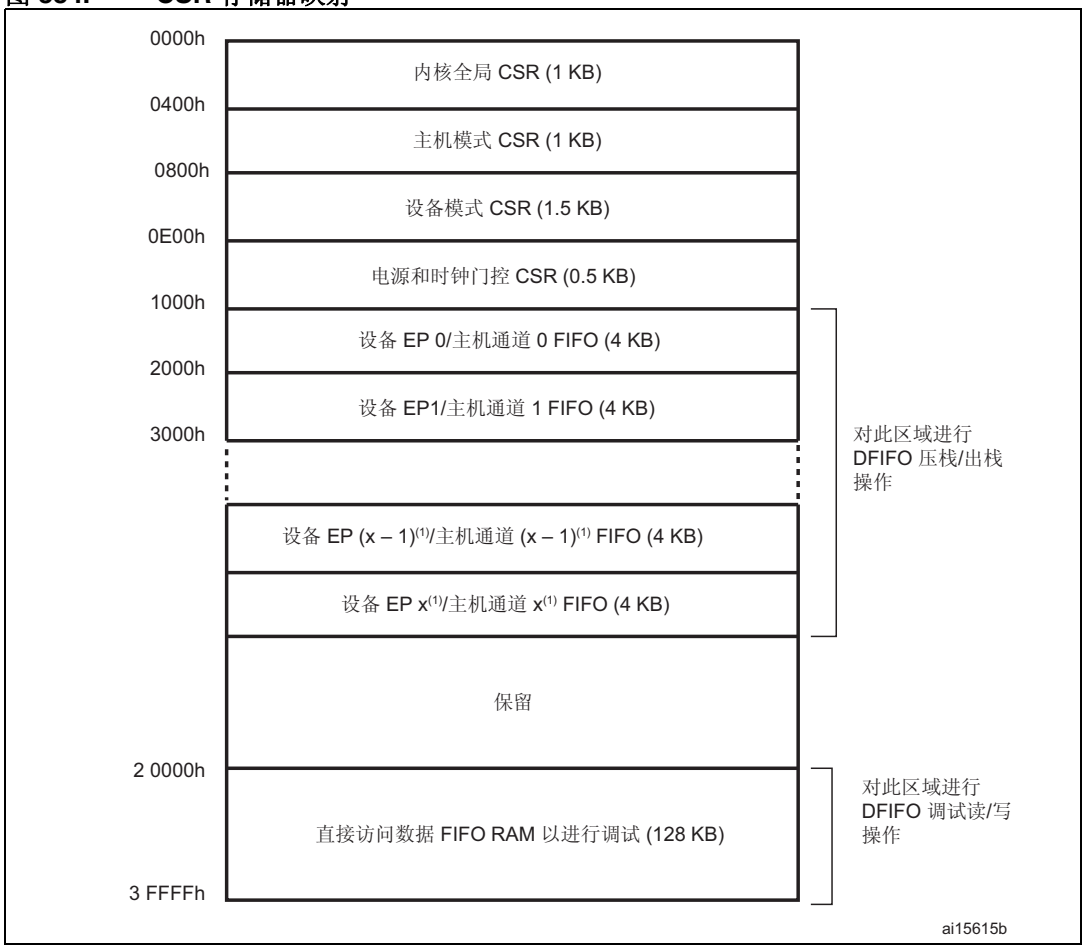
- 电源和时钟门控寄存器
- 数据 FIFO (DFIFO) 访问寄存器

只有模块全局寄存器、电源和时钟门控寄存器、数据 FIFO 访问寄存器以及主机端口控制和状态寄存器，可在主机模式和设备模式下进行访问。当 OTG_HS 控制器在一种模式（设备模式或主机模式）下工作时，应用程序不得以另一种角色模式访问寄存器。如果发生了非法访问，将会产生模式不匹配中断并在模块中断寄存器（OTG_HS_GINTSTS 寄存器中的 MMIS 位）中反映。当模块从一种角色模式切换到另一种角色模式时，新工作模式下的寄存器必须重新编程为上电复位后的状态。

31.12.1 CSR 存储器映射

主机和设备模式寄存器占用不同的地址。所有寄存器均在 AHB 时钟域内实现。

图 384. CSR 存储器映射



1. 设备模式下 x = 5；主机模式下 x = 11。

全局 CSR 地址映射

这些寄存器在主机模式和设备模式下均可用。

表 178. 模块全局控制和状态寄存器 (CSR)

缩写	偏移地址	寄存器名称
OTG_HS_GOTGCTL	0x000	*第 1073 页的OTG_HS 控制和状态寄存器 (OTG_HS_GOTGCTL)*
OTG_HS_GOTGINT	0x004	*第 1075 页的OTG_HS 中断寄存器 (OTG_HS_GOTGINT)*
OTG_HS_GAHBCFG	0x008	*第 1076 页的OTG_HS AHB 配置寄存器 (OTG_HS_GAHBCFG)*
OTG_HS_GUSBCFG	0x00C	*第 1077 页的OTG_HS USB 配置寄存器 (OTG_HS_GUSBCFG)*
OTG_HS_GRSTCTL	0x010	*第 1080 页的OTG_HS 复位寄存器 (OTG_HS_GRSTCTL)*
OTG_HS_GINTSTS	0x014	*第 1081 页的OTG_HS 模块中断寄存器 (OTG_HS_GINTSTS)*
OTG_HS_GINTMSK	0x018	*第 1085 页的OTG_HS 中断屏蔽寄存器 (OTG_HS_GINTMSK)*
OTG_HS_GRXSTSR	0x01C	*第 1088 页的OTG_HS 接收状态调试读取/OTG 状态读取和出栈寄存器 (OTG_HS_GRXSTSR/OTG_HS_GRXSTSP)*
OTG_HS_GRXSTSP	0x020	
OTG_HS_GRXFSIZ	0x024	*第 1090 页的OTG_HS 接收 FIFO 大小寄存器 (OTG_HS_GRXFSIZ)*
OTG_HS_GNPTXFSIZ/ OTG_HS_TX0FSIZ	0x028	*第 1090 页的OTG_HS 非周期性发送 FIFO 大小/端点 0 发送 FIFO 大小寄存器 (OTG_HS_GNPTXFSIZ/OTG_HS_TX0FSIZ)*
OTG_HS_GNPTXSTS	0x02C	*第 1091 页的OTG_HS 非周期性发送 FIFO/队列状态寄存器 (OTG_HS_GNPTXSTS)*
OTG_HS_GCCFG	0x038	*第 1093 页的OTG_HS 通用模块配置寄存器 (OTG_HS_GCCFG)*
OTG_HS_CID	0x03C	*第 1094 页的OTG_HS 模块 ID 寄存器 (OTG_HS_CID)*
OTG_HS_HPTXFSIZ	0x100	*第 1094 页的OTG_HS 主机周期性发送 FIFO 大小寄存器 (OTG_HS_HPTXFSIZ)*
OTG_HS_DIEPTXFx	0x104 0x124 ... 0x13C	*第 1095 页的OTG_HS 设备 IN 端点发送 FIFO 大小寄存器 (OTG_HS_DIEPTXFx)（x = 1..7，其中 x 为 FIFO 编号）*

主机模式 CSR 地址映射

模块每次切换到主机模式时都必须对这些寄存器进行设置。

表 179. 主机模式控制和状态寄存器 (CSR)

缩写	偏移地址	寄存器名称
OTG_HS_HCFG	0x400	*第 1095 页的OTG_HS 主机配置寄存器 (OTG_HS_HCFG)*
OTG_HS_HFIR	0x404	*第 1096 页的OTG_HS 主机帧间隔寄存器 (OTG_HS_HFIR)*
OTG_HS_HFNUM	0x408	*第 1096 页的OTG_HS 主机帧编号/帧剩余时间寄存器 (OTG_HS_HFNUM)*
OTG_HS_HPTXSTS	0x410	*第 1097 页的OTG_HS_Host 周期性发送 FIFO/队列状态寄存器 (OTG_HS_HPTXSTS)*

表 179. 主机模式控制和状态寄存器 (CSR) （续）

缩写	偏移地址	寄存器名称
OTG_HS_HAINT	0x414	*第 1098 页的OTG_HS 主机全体通道中断寄存器 (OTG_HS_HAINT)*
OTG_HS_HAINTMSK	0x418	*第 1098 页的OTG_HS 主机全体通道中断屏蔽寄存器 (OTG_HS_HAINTMSK)*
OTG_HS_HPRT	0x440	*第 1098 页的OTG_HS 主机端口控制和状态寄存器 (OTG_HS_HPRT)*
OTG_HS_HCCHARx	0x500 0x520 ... 0x6E0	*第 1101 页的OTG_HS 主机通道 x 特性寄存器 (OTG_HS_HCCHARx)（x = 0..11，其中 x = 通道编号）*
OTG_HS_HCSPLTx	0x504	*第 1102 页的OTG_HS 主机通道 x 分离控制寄存器 (OTG_HS_HCSPLTx)（x = 0..11，其中 x = 通道编号）*
OTG_HS_HCINTx	0x508	*第 1103 页的OTG_HS 主机通道 x 中断寄存器 (OTG_HS_HCINTx)（x = 0..11，其中 x = 通道 编号）*
OTG_HS_HCINTMSKx	0x50C	*第 1104 页的OTG_HS 主机通道 x 中断屏蔽寄存器 (OTG_HS_HCINTMSKx)（x = 0..11，其中 x = 通道编号）*
OTG_HS_HCTSIZx	0x510	*第 1105 页的OTG_HS 主机通道 x 传输大小寄存器 (OTG_HS_HCTSIZx)（x = 0..11，其中 x = 通道编号）*
OTG_HS_HCDMAx	0x514	*第 1105 页的OTG_HS 主机通道 x DMA 地址寄存器 (OTG_HS_HCDMAx)（x = 0..11，其中 x = 通道编号）*

设备模式 CSR 地址映射

模块每次切换到设备模式时都必须对这些寄存器进行设置。

表 180. 设备模式控制和状态寄存器

缩写	偏移地址	寄存器名称
OTG_HS_DCFG	0x800	*第 1106 页的OTG_HS 设备配置寄存器 (OTG_HS_DCFG)*
OTG_HS_DCTL	0x804	*第 1107 页的OTG_HS 设备控制寄存器 (OTG_HS_DCTL)*
OTG_HS_DSTS	0x808	*第 1109 页的OTG_HS 设备状态寄存器 (OTG_HS_DSTS)*
OTG_HS_DIEPMSK	0x810	*第 1109 页的OTG_HS 设备 IN 端点通用中断屏蔽寄存器 (OTG_HS_DIEPMSK)*
OTG_HS_DOEPMSK	0x814	*第 1110 页的OTG_HS 设备 OUT 端点通用中断屏蔽寄存器 (OTG_HS_DOEPMSK)*
OTG_HS_DAINT	0x818	*第 1112 页的OTG_HS 设备全体端点中断寄存器 (OTG_HS_DAINT)*
OTG_HS_DAINTMSK	0x81C	*第 1112 页的OTG_HS 全体端点中断屏蔽寄存器 (OTG_HS_DAINTMSK)*
OTG_HS_DVBUSDIS	0x828	*第 1113 页的OTG_HS 设备 V_{BUS} 放电时间寄存器 (OTG_HS_DVBUSDIS)*
OTG_HS_DVBUSPULSE	0x82C	*第 1113 页的OTG_HS 设备 V_{BUS} 脉冲时间寄存器 (OTG_HS_DVBUSPULSE)*

表 180. 设备模式控制和状态寄存器 （续）

缩写	偏移地址	寄存器名称
OTG_HS_DIEPEMPMSK	0x834	*第 1115 页的OTG_HS 设备 IN 端点 FIFO 空中断屏蔽寄存器: OTG_HS_DIEPEMPMSK)*
OTG_HS_EACHHINT	0x838	*第 1115 页的OTG_HS 设备单个端点中断寄存器 (OTG_HS_DEACHINT)*
OTG_HS_EACHHINTMSK	0x83C	*第 1116 页的OTG_HS 设备单个端点中断屏蔽寄存器 (OTG_HS_DEACHINTMSK)*
OTG_HS_DIEPEACHMSK1	0x844	*第 1116 页的OTG_HS 设备 IN 端点 1 中断屏蔽寄存器 (OTG_HS_DIEPEACHMSK1)*
OTG_HS_DOEPEACHMSK1	0x884	*第 1117 页的OTG_HS 设备 OUT 端点 1 中断屏蔽寄存器 (OTG_HS_DOEPEACHMSK1)*
OTG_HS_DIEPCTLx	0x920 0x940 ... 0xAE0	*第 1118 页的OTG 设备端点 x 控制寄存器 (OTG_HS_DIEPCTLx) （x = 0..7，其中 x = 端点编号）*
OTG_HS_DIEPINTx	0x908	*第 1124 页的OTG_HS 设备端点 x 中断寄存器 (OTG_HS_DIEPINTx) （x = 0..7，其中 x = 端点 编号）*
OTG_HS_DIEPTSIZ0	0x910	*第 1126 页的OTG_HS 设备 IN 端点 0 传输大小寄存器 (OTG_HS_DIEPTSIZ0)*
OTG_HS_DIEPDMAx	0x914	*第 1129 页的OTG_HS 设备端点 x DMA 地址寄存器 (OTG_HS_DIEPDMAx/OTG_HS_DOEPDMAx)（x = 1..5，其中 x = 端点编号）*
OTG_HS_DTXFSTSx	0x918	*第 1128 页的OTG_HS 设备 IN 端点发送 FIFO 状态寄存器 (OTG_HS_DTXFSTSx)（x = 0..5，其中 x = 端点编号）*
OTG_HS_DIEPTSIZx	0x930 0x950 ... 0xAF0	*第 1128 页的OTG_HS 设备端点 x 传输大小寄存器 (OTG_HS_DOEPTSIZx)（x = 1..5，其中 x = 端点编号）*
OTG_HS_DOEPCTL0	0xB00	*第 1120 页的OTG_HS 设备控制 OUT 端点 0 控制寄存器 (OTG_HS_DOEPCTL0)*
OTG_HS_DOEPCTLx	0xB20 0xB40 ... 0xCC0 0xCE0 0xCFD	*第 1118 页的OTG 设备端点 x 控制寄存器 (OTG_HS_DIEPCTLx) （x = 0..7，其中 x = 端点编号）*
OTG_HS_DOEPINTx	0xB08	*第 1124 页的OTG_HS 设备端点 x 中断寄存器 (OTG_HS_DIEPINTx) （x = 0..7，其中 x = 端点 编号）*
OTG_HS_DOEPTSIZx	0xB10	*第 1128 页的OTG_HS 设备端点 x 传输大小寄存器 (OTG_HS_DOEPTSIZx)（x = 1..5，其中 x = 端点编号）*

数据 FIFO (DFIFO) 访问寄存器地址映射

这些寄存器在主机模式和设备模式下均可用，用于对给定方向的特定端点或通道的 FIFO 空间进行读写操作。如果主机通道类型为 IN，则只能对该通道上的 FIFO 进行读操作。同样地，如果主机通道类型为 OUT，则只能对该通道上的 FIFO 进行写操作。

表 181. 数据 FIFO (DFIFO) 访问寄存器地址映射

FIFO 访问寄存器段	地址范围	访问方式
设备 IN 端点 0/主机 OUT 通道 0：DFIFO 写访问 设备 OUT 端点 0/主机 IN 通道 0：DFIFO 读访问	0x1000–0x1FFC	w r
设备 IN 端点 1/主机 OUT 通道 1：DFIFO 写访问 设备 OUT 端点 1/主机 IN 通道 1：DFIFO 读访问	0x2000–0x2FFC	w r
...	...	...
设备 IN 端点 x(1)/主机 OUT 通道 x(1)：DFIFO 写访问 设备 OUT 端点 x(1)/主机 IN 通道 x(1)：DFIFO 读访问	0xX000h–0xXFFCh	w r

1. 其中，设备模式下 x 为 5；主机模式下为 11。

电源和时钟门控 CSR 地址映射

由一个单独寄存器对电源和时钟门控进行管理。该寄存器在主机模式和设备模式下均可用。

表 182. 电源和时钟门控控制和状态寄存器

寄存器名称	缩写	偏移地址：0xE00–0xFFF
电源和时钟门控控制寄存器	PCGCR	0xE00-0xE04
保留		0xE05–0xFFF

31.12.2 OTG_HS 全局寄存器

这些寄存器在主机模式和设备模式下都可用，且在这两个模式间切换时无需对其进行重新编程。

除非特别说明，否则寄存器描述中的位值以二进制表示。

OTG_HS 控制和状态寄存器 (OTG_HS_GOTGCTL)

OTG_HS control and status register

偏移地址：0x000

复位值：0x0000 0800

OTG 控制和状态寄存器控制模块的 OTG 功能并反映其状态。

31–20	19	18	17	16	15–12	11	10	9	8	7–2	1	0
Reserved	BSVLD	ASVLD	DBCT	CIDSTS	Reserved	DHNPEN	HSHNPEN	HNPRQ	HNGSCS	Reserved	SRQ	SRQSCS
	r	r	r	r		rw	rw	rw	r		rw	r

位 31:20 保留，必须保持复位值。

位 19 **BSVLD**：B 会话有效 (B-session valid)

指示设备模式下收发器的状态。
0：B 会话无效。
1：B 会话有效。
在 OTG 模式下，该位可用于确定设备处于连接状态还是断开状态。
注意：仅可在设备模式下访问。

位 18 **ASVLD**：A 会话有效 (A-session valid)

指示主机模式下收发器的状态。
0：A 会话无效
1：A 会话有效
注意：仅可在主机模式下访问。

位 17 **DBCT**：长/短去抖动时间 (Long/short debounce time)

指示已检测到连接的去抖动时间。
0：长去抖动时间，用于物理连接 (100 ms + 2.5 µs)
1：短去抖动时间，用于软连接 (2.5 µs)
注意：仅可在主机模式下访问。

位 16 **CIDSTS**：连接器 ID 状态 (Connector ID status)

指示发生连接事件时的连接器 ID 状态。
0：OTG_HS 控制器处于 A 器件模式
1：OTG_HS 控制器处于 B 器件模式
注意：在设备模式和主机模式下均可访问。

位 15:12 保留，必须保持复位值。

位 11 **DHNPEN**：使能设备 HNP 特性 (Device HNP enabled)

从所连 USB 主机处成功接收到 SetFeature.SetHNPEnable 命令时，应用程序将该位置 1。
0：未在应用程序中使能 HNP
1：已在应用程序中使能 HNP
注意：仅可在设备模式下访问。

位 10 **HSHNPEN**：主机设置 HNP 使能 (Host set HNP enable)

（使用 SetFeature.SetHNPEnable 命令）在所连接设备上成功使能 HNP 后，应用程序将该位置 1。
0：未使能主机设置 HNP
1：已使能主机设置 HNP
注意：仅可在主机模式下访问。

位 9 **HNPRQ**：HNP 请求 (HNP request)

应用程序将该位置 1 时，将对所连接 USB 主机发起 HNP 请求。当 OTG 中断寄存器中的主机协商成功状态更改位（OTG_HS_GOTGINT 中的 HNSSCHG 位）置 1 时，应用程序可通过写 0 将该位清零。HNSSCHG 位清零时，模块会将该位清零。
0：不发送 HNP 请求
1：发送 HNP 请求
注意：仅可在设备模式下访问。

位 8 **HNGSCS**：主机协商成功 (Host negotiation success)

当主机协商成功时，模块会将该位置 1。当该寄存器中的 HNP 请求位 (HNPRQ) 置 1 时，模块会将该位清零。
0：主机协商失败
1：主机协商成功
注意：仅可在设备模式下访问。

位 7:2 保留，必须保持复位值。

位 1 **SRQ**：会话请求 (Session request)

应用程序将该位置 1 时，将在 USB 上发起会话请求。当 OTG 中断寄存器中的主机协商成功状态更改位（OTG_HS_GOTGINT 中的 HNSSCHG 位）置 1 时，应用程序可通过写 0 将该位清零。HNSSCHG 位清零时，模块会将该位清零。

如果使用 USB 1.1 全速串行收发器接口来启动会话请求，则应用程序必须在该寄存器中的 B 会话有效位（OTG_HS_GOTGCTL 中的 BSVLD 位）清零后，等候 V_{BUS} 放电至 0.2 V。该放电时间因 PHY 不同而异，可从 PHY 供应商处了解相关信息。

0：无会话请求

1：会话请求

注意：仅可在设备模式下访问。

位 0 **SRQSCS**：会话请求成功 (Session request success)

当会话请求成功时，模块会将该位置 1。

0：会话请求失败

1：会话请求成功

注意：仅可在设备模式下访问。

OTG_HS 中断寄存器 (OTG_HS_GOTGINT)

OTG_HS interrupt register

偏移地址：0x04

复位值：0x0000 0000

应用程序会在出现 OTG 中断时读取该寄存器，并通过将该寄存器中的位清零来清除 OTG 中断。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												DBCDNE	ADTOCHG	HNGDET	Reserved							HNSSCHG	SRSSCHG	Reserved					SEDET	Res.	
												rc_w1	rc_w1	rc_w1								rc_w1	rc_w1						rc_w1		

位 31:20 保留，必须保持复位值。

位 19 **DBCDNE**：去抖动完成 (Debounce done)

模块会在设备连接后且去抖动结束时将该位置 1。应用程序会在看见该中断后，开始驱动 USB 复位。该位仅在模块 USB 配置寄存器中的 HNP 使能位或 SRP 使能位（分别为 OTG_HS_GUSBCFG 中的 HNPCAP 位或 SRPCAP 位）置 1 时有效。

注意：仅可在主机模式下访问。

位 18 **ADTOCHG**：A 器件超时更改 (A-device timeout change)

模块将该位置 1 时，指示 A 器件在等待 B 器件连接时超时。

注意：在设备模式和主机模式下均可访问。

位 17 **HNGDET**：检测到主机协商 (Host negotiation detected)

当检测到 USB 上的主机协商请求时，模块会将该位置 1。

注意：在设备模式和主机模式下均可访问。

位 16:10 保留，必须保持复位值。

位 9 **HNSSCHG**：主机协商成功状态更改 (Host negotiation success status change)

模块将在 USB 主机协商请求成功或失败时将此位置 1。应用程序必须读取 OTG 控制和状态寄存器中的主机协商成功位（OTG_HS_GOTGCTL 中的 HNGSCS）来确定请求成功还是失败。

注意：在设备模式和主机模式下均可访问。

位 7:3 保留，必须保持复位值。

位 8 **SRSSCHG**：会话请求成功状态更改 (Session request success status change)

模块将在会话请求成功或失败时将此位置 1。应用程序必须读取 OTG 控制和状态寄存器中的会话请求成功位（OTG_HS_GOTGCTL 中的 SRQSCS 位）来确定请求成功还是失败。

注意：在设备模式和主机模式下均可访问。

位 2 **SEDET**：检测到会话结束 (Session end detected)

模块将该位置 1 时，指示 V_{BUS} < 0.8 V，V_{BUS} 上的电压不再适用于 B 器件会话。

位 1:0 保留，必须保持复位值。

OTG_HS AHB 配置寄存器 (OTG_HS_GAHBCFG)

OTG_HS AHB configuration register

偏移地址：0x008

复位值：0x0000 0000

该寄存器可用于在上电后或更改角色模式时对模块进行配置。该寄存器主要包含 AHB 系统相关的配置参数。应用程序必须在开始任何 AHB 或 USB 事务前对该寄存器进行编程。请勿在初始编程后更改该寄存器。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																							PTXFELVL	TXFELVL	Reserved	DMAEN	HBSTLEN				GINT
																							rw	rw		rw					rw

位 31:20 保留，必须保持复位值。

位 8 **PTXFELVL**：周期性 TxFIFO 空门限 (Periodic TxFIFO empty level)

指示何时触发模块中断寄存器中的周期性 TxFIFO 空中断位（OTG_HS_GINTSTS 中的 PTXFE 位）。

0：PTXFE（位于 OTG_HS_GINTSTS）中断指示周期性 TxFIFO 为半空状态
1：PTXFE（位于 OTG_HS_GINTSTS）中断指示周期性 TxFIFO 为全空状态

注意：仅可在主机模式下访问。

位 7 **TXFELVL**：TxFIFO 空级别 (TxFIFO empty level)

在设备模式下，该位指示何时触发 IN 端点发送 FIFO 空中断（OTG_HS_DIEPINTx 中的 TXFE）。

0：TXFE（位于 OTG_HS_DIEPINTx）中断指示 IN 端点 TxFIFO 为半空状态
1：TXFE（位于 OTG_HS_DIEPINTx）中断指示 IN 端点 TxFIFO 为全空状态

注意：仅可在设备模式下访问。

位 6 保留，必须保持复位值。

位 5 **DMAEN**：DMA 使能 (DMA enable)

0：模块以从模式运行

1：模块以 DMA 模式运行

位 4:1 **HBSTLEN**：批量长度/类型 (Burst length/type)

0000 单次

0001 INCR

0011 INCR4

0101 INCR8

0111 INCR16

其它值：保留

位 0 **GINT**：全局中断屏蔽 (Global interrupt mask)

该位用于屏蔽全局中断或对全局中断取消屏蔽。中断状态寄存器由模块进行更新，与此位的设置无关。

0：屏蔽应用程序触发的中断。

1：取消对应用程序触发的中断的屏蔽。

注意：在设备模式和主机模式下均可访问。

OTG_HS USB 配置寄存器 (OTG_HS_GUSBCFG)

OTG_HS USB configuration register

偏移地址：0x00C

复位值：0x0000 0A00

该寄存器可用于在上电或更改角色模式后对模块进行配置。其中包含与 USB 和 USB-PHY 相关的配置参数。应用程序必须在开始任何 AHB 或 USB 事务前对该寄存器进行编程。请勿在初始编程后更改该寄存器。

31	30	29	28 27 26	25	24	23	22	21	20	19	18	17	16	15	14	13 12 11 10	9	8	7	6	5 4 3	2 1 0
CTXPKT	FDMOD	FHMOD	Reserved	ULPIIPD	PTCI	PCCI	TSDPS	ULPIEVBUSI	ULPIEVBUSD	ULPICSM	ULPIAR	ULPIFSLS	Reserved	PHYLPCS	Reserved	TRDT	HNPCAP	SRPCAP	Reserved	PHSEL	Reserved	TOCAL
rw	rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw		rw		rw	r/rw	r/rw		wo		rw

位 31 **CTXPKT**：损坏的发送数据包 (Corrupt Tx packet)

该位仅用于调试。切勿将其置 1。

注意：在设备模式和主机模式下均可访问。

位 30 **FDMOD**：强制设备模式 (Forced peripheral mode)

向该位写入 1 时，可将模块强制为设备模式，而无需考虑 OTG_HS_ID 输入引脚。

0：正常模式

1：强制设备模式

将强制位置 1 后，应用程序必须等待至少 25 ms 后更改方可生效。

注意：在设备模式和主机模式下均可访问。

位 29 **FHMOD**：强制主机模式 (Forced host mode)

向该位写入 1 时，可将模块强制为主机模式，而无需考虑 OTG_HS_ID 输入引脚。

0：正常模式

1：强制主机模式

将强制位置 1 后，应用程序必须等待至少 25 ms 后更改方可生效。

注意：在设备模式和主机模式下均可访问。

位 28:26 保留，必须保持复位值。

位 25 **ULPIIPD**：ULPI 接口保护禁止 (ULPI interface protect disable)

该位控制 PHY 中内置的电路，以保护 ULPI 接口。该功能所采用的任何上拉或下拉电阻均可禁止。有关详细信息，请参见 ULPI 规范。

0：使能接口保护电路

1：禁止接口保护电路

位 24 **PTCI**：通过指示符 (Indicator pass through)

该位将控制互补输出是否需要通过内部 V_{BUS} 有效电平比较器的验证，之后方可用于 RX CMD 中的 V_{BUS} 状态。有关详细信息，请参见 ULPI 规范。

0：互补输出信号通过内部 V_{BUS} 有效电平比较器的验证

1：互补输出信号不通过内部 V_{BUS} 有效电平比较器的验证

位 23 **PCCI**：互补指示符 (Indicator complement)

该位控制 PHY 将 ExternalVbusIndicator 输入信号反转并生成互补输出。有关详细信息，请参见 ULPI 规范。

0：PHY 不将 ExternalVbusIndicator 信号反转

1：PHY 将 ExternalVbusIndicator 信号反转

位 22 **TSDPS**：TermSel DLine 脉冲选择 (TermSel DLine pulsing selection)

该位选择 utmi_termselect 来驱动 SRP（会话请求协议）期间的数据线脉冲。

0：使用 utmi_txvalid 驱动数据线脉冲（默认）

1：使用 utmi_termsel 驱动数据线脉冲

位 21 **ULPIEVBUSI**：ULPI 外部 V_{BUS} 指示符 (ULPI external VBUS indicator)

该位指示 ULPI PHY 使用外部 V_{BUS} 过流指示器。

0：PHY 使用内部 V_{BUS} 有效比较器

1：PHY 使用外部 V_{BUS} 有效比较器

位 20 **ULPIEVBUSD**：ULPI 外部 V_{BUS} 驱动器 (ULPI External VBUS Drive)

该位选择 ULPI PHY 使用内部或外部电源来驱动 V_{BUS} 上的 5 V 电压。

0：PHY 使用内部电荷泵驱动 V_{BUS}（默认）

1：PHY 使用外部电源驱动 V_{BUS}。

位 19 **ULPICSM**：ULPI 时钟 SuspendM (ULPI Clock SuspendM)

该位置位 ULPI PHY 接口控制寄存器中的 ClockSuspendM 位。该位仅适用于串行和 carkit 模式。

0：PHY 在挂起期间断开内部时钟的电源

1：PHY 不断开内部时钟的电源

位 18 **ULPIAR**：ULPI 自动恢复 (ULPI Auto-resume)

该位置位 ULPI PHY 接口控制寄存器中的 AutoResume 位。

0：PHY 不支持自动恢复功能

1：PHY 使用自动恢复功能

位 17 **ULPIFSLS**：ULPI FS/LS 选择 (ULPI FS/LS select)

应用程序使用该位为 ULPI PHY 选择 FS/LS 串行接口。该位仅在 ULPI PHY 上选择了 FS 串行收发器的情况下有效。

0：ULPI 接口

1：ULPI FS/LS 串行接口

位 16 保留，必须保持复位值。

位 15 **PHYLPCS**：PHY 低功耗时钟选择 (PHY Low-power clock select)

该位用于选择 480 MHz 或 48 MHz（低功耗）PHY 模式。在 FS 和 LS 模式下，PHY 通常可使用 48 MHz 时钟运行，从而节约功耗。

0：480 MHz 内部 PLL 时钟

1：48 MHz 外部时钟

在 480 MHz 模式下，UTMI 接口以 60 MHz 或 30MHz 运行，具体取决于选择的是 8 位还是 16 位数据宽度。在 48 MHz 模式下，UTMI 接口在 FS 和 LS 模式下以 48 MHz 运行。

位 14 **保留，必须保持复位值。**

位 13:10 **TRDT**：USB 周转时间 (USB turnaround time)

以 PHY 时钟数为单位设置周转时间。

TRDT 值按以下公式计算：

TRDT = 4 × AHB 时钟频率 + 1 PHY 时钟频率。

例如：

如果 AHB 时钟频率 = 72 MHz（PHY 时钟频率 = 48 MHz），则 TRDT 必须设置为 9。

如果 AHB 时钟频率 = 48 MHz（PHY 时钟频率 = 48 MHz），则 TRDT 必须设置为 5。

注意：仅可在设备模式下访问。

位 9 **HNPCAP**：HNP 使能 (HNP-capable)

应用程序使用该位控制 OTG_HS 控制器的 HNP 功能。

0：不使能 HNP 功能

1：使能 HNP 功能

注意：在设备模式和主机模式下均可访问。

位 8 **SRPCAP**：SRP 使能 (SRP-capable)

应用程序使用该位控制 OTG_HS 控制器的 SRP 功能。如果模块作为不具有 SRP 功能的 B 器件运行，则无法请求连接的 A 器件激活 V_{BUS} 并开始会话。

0：不使能 SRP 功能

1：使能 SRP 功能

注意：在设备模式和主机模式下均可访问。

位 7 保留，必须保持复位值。

位 6 **PHSEL**：USB 2.0 高速 ULPI PHY 或 USB 1.1 全速串行收发器选择 (USB 2.0 high-speed ULPI PHY or USB 1.1 full-speed serial transceiver select)

0：USB 2.0 高速 ULPI PHY

1：USB 1.1 全速串行收发器

位 5:3 保留，必须保持复位值。

位 2:0 **TOCAL**：FS 超时校准 (FS timeout calibration)

PHY 引入的额外延迟包括应用程序在该字段中设置的 PHY 时钟数，以及模块的全速数据包间超时间隔。不同 PHY 引入的延迟对数据线状态的影响是不同的。

全速操作的 USB 标准超时值为 16 到 18（含）个位时间。应用程序必须根据枚举速度编程该字段。每个 PHY 时钟增加的位时间数为 0.25 个位时间。

OTG_HS 复位寄存器 (OTG_HS_GRSTCTL)

OTG_HS reset register

偏移地址：0x010

复位值：0x2000 0000

应用程序通过此寄存器复位模块中的各项硬件特性。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AHBIDL	DMAREQ	Reserved																			TXFNUM					TXFFLSH	RXFFLSH	Reserved	FCRST	HSRST	CSRST
r	r																				rw					rs	rs		rs	rs	rs

位 31 **AHBIDL**：AHB 主器件空闲 (AHB master idle)

指示 AHB 主器件状态机处于空闲情况。

注意：在设备模式和主机模式下均可访问。

位 30 **DMAREQ**：DMA 请求信号 (DMA request signal)

该位指示 DMA 请求正在进行中。用于调试。

位 29:11 保留，必须保持复位值。

位 10:6 **TXFNUM**：TxFIFO 编号 (TxFIFO number)

使用 TxFIFO 刷新位进行 FIFO 刷新的 FIFO 编号。只有在模块将 TxFIFO 刷新位清零后，方可更改此字段。

- 00000：
  - 主机模式下刷新非周期性 TxFIFO
  - 设备模式下刷新 Tx FIFO 0
- 00001：
  - 主机模式下刷新周期性 TxFIFO
  - 设备模式下刷新 TXFIFO 1
- 00010：设备模式下刷新 TXFIFO 2

...

- 00101：设备模式下刷新 TXFIFO 15
- 10000：在设备模式或主机模式下刷新所有的发送 FIFO。

注意：在设备模式和主机模式下均可访问。

位 5 **TXFFLSH**：TxFIFO 刷新 (TxFIFO flush)

此位选择性地刷新一个或所有的发送 FIFO，但当模块处理通信事务时无法执行该操作。

只有在确认模块当前未对 TxFIFO 执行读写操作后，应用程序方可对此位执行写操作。使用以下寄存器进行确认：

- 读：NAK 有效中断可确保模块当前未对 FIFO 执行读操作
- 写：OTG_HS_GRSTCTL 中的 AHBIDL 位可确保模块当前未对 FIFO 执行任何写操作

注意：在设备模式和主机模式下均可访问。

位 4 **RXFFLSH**：RxFIFO 刷新 (RxFIFO flush)

应用程序可使用此位刷新整个 RxFIFO，但必须首先确保模块当前未在处理通信事务。

只有在确认模块当前未对 RxFIFO 执行读写操作后，应用程序方可对此位执行写操作。

应用程序必须等到此位清零后，方可执行其它操作。通常需要等待 8 个时钟周期（以 PHY 或 AHB 时钟中最慢的为准）。

注意：在设备模式和主机模式下均可访问。

位 3 保留，必须保持复位值。

位 2 **FCRST**：主机帧计数器复位 (Host frame counter reset)

应用程序对该位执行写操作时，模块中的帧数计数器复位。帧计数器复位后，由模块发送的下一个 SOF 的帧号为 0。

注意：仅可在主机模式下访问。

位 1 **HSRST**：HCLK 软复位 (HCLK soft reset)

应用程序使用此位来刷新 AHB 时钟域中的控制逻辑。仅复位 AHB 时钟域流水线。

FIFO 不通过此位来刷新。

遵照协议终止 AHB 上的事务后，AHB 时钟域中的所有状态机均复位至空闲状态。

AHB 时钟域状态机所使用的 CSR 控制位清零。

要清零该中断，需要将由 AHB 时钟域状态机生成并用于控制中断状态的状态屏蔽位清零。

由于中断状态位并未清零，因此应用程序可以获取在该位置 1 后所发生的所有模块事件的状态。

此位为自清零位，模块将在其中所有必要逻辑复位后将该位清零。该过程需要若干个时钟的时间，具体取决于模块的当前状态。

注意：在设备模式和主机模式下均可访问。

位 0 **CSRST**：模块软复位 (Core soft reset)

按如下所述将 HCLK 和 PCLK 域复位：

除以下各位外，将各个中断和所有 CSR 寄存器位清零：

– OTG_HS_PCGCCTL 中的 RSTPDMODL 位
– OTG_HS_PCGCCTL 中的 GAYEHCLK 位
– OTG_HS_PCGCCTL 中的 PWRCLMP 位
– OTG_HS_PCGCCTL 中的 STPPCLK 位
– OTG_HS_HCFG 中的 FSLSPCS 位
– OTG_HS_DCFG 中的 DSPD 位

将所有模块状态机（AHB 从器件除外）复位至空闲状态，并清空所有发送 FIFO 和接收 FIFO。

在 AHB 传输的最后数据阶段结束后，尽快终止 AHB 主器件上的所有事务。立即终止 USB 上的所有事务。

应用程序可在需要复位模块时随时对该位执行写操作。该位为自清零位，模块将在其中所有必要逻辑复位后将该位清零，该过程需要若干个时钟的时间，具体取决于模块的当前状态。该位一旦清零，软件必须等待至少 3 个 PHY 时钟后才可以访问 PHY 域（同步延迟）。此外，软件还必须在确定该寄存器中的位 31 置 1（AHB 主器件空闲）后方可开始运行。

软件复位通常在两种情况下使用，一是软件开发期间，二是用户动态更改以上所列 USB 配置寄存器中的 PHY 选择位后。用户更改 PHY 时，将为 PHY 选择相应的时钟并用于 PHY 域中。一旦选择了新的时钟，则必须复位 PHY 域，才能保证正常运行。

注意：在设备模式和主机模式下均可访问。

OTG_HS 模块中断寄存器 (OTG_HS_GINTSTS)

OTG_HS core interrupt register

偏移地址：0x014

复位值：0x0400 0020

该寄存器用于在当前模式（设备模式或主机模式）下借助系统级别的事件来中断应用程序。

该寄存器中的某些位仅在主机模式下有效，而其它位则仅在设备模式下有效。此外，该寄存器还可指示当前模式。要将 rc_w1 类型中断状态位清零，应用程序必须向该位写入 1。

FIFO 状态中断为只读；如果软件在处理这些中断期间对 FIFO 执行读写操作，则 FIFO 中断标志将自动清零。

使能中断位前，应用程序必须在初始化时将 OTG_HS_GINTSTS 寄存器清零，才可以避免在初始化前产生任何中断。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WKUINT	SRQINT	DISCINT	CIDSCHG	Reserved	PTXFE	HCINT	HPRTINT	Reserved	DATAFSUSP	IPXFR/INCOMPISOOUT	IISOIXFR	OEPINT	IEPINT	Reserved		EOPF	ISOODRP	ENUMDNE	USBRST	USBSUSP	ESUSP	Reserved		BOUTNAKEFF	GINAKEFF	NPTXFE	RXFLVL	SOF	OTGINT	MMIS	CMOD
rc_w1					r	r	r			rc_w1		r	r			rc_w1								r	r	r	r	rc_w1	r	rc_w1	r

位 31 **WKUPINT**：检测到恢复/远程唤醒中断 (Resume/remote wakeup detected interrupt)

在设备模式下，当 USB 总线上检测到恢复信号时，将触发该中断。在主机模式下，当 USB 上检测到远程唤醒时，将触发该中断。

注意：在设备模式和主机模式下均可访问。

位 30 **SRQINT**：检测到会话请求/新会话中断 (Session request/new session detected interrupt)

在主机模式下，当检测到来自设备的会话请求时，将触发该中断。在设备模式下，当 V_{BUS} 在 B 器件的有效范围内时，将触发该中断。在设备模式和主机模式下均可访问。

位 29 **DISCINT**：检测到断开连接中断 (Disconnect detected interrupt)

当检测到设备断开连接时触发该中断。

注意：仅可在主机模式下访问。

位 28 **CIDSCHG**：连接器 ID 线状态更改 (Connector ID status change)

当连接器 ID 线状态发生更改时，模块将该位置 1。

注意：在设备模式和主机模式下均可访问。

位 27 保留，必须保持复位值。

位 26 **PTXFE**：周期性 TxFIFO 空 (Periodic TxFIFO empty)

当周期性发送 FIFO 为半空或全空状态，且周期性请求队列中存在可写入至少一个条目的空间时，将触发该中断。该 TxFIFO 为半空状态还是全空状态由模块 AHB 配置寄存器中的周期性 TxFIFO 空级别位（OTG_HS_GAHBCFG 中的 PTXFELVL 位）决定。

注意：仅可在主机模式下访问。

位 25 **HCINT**：主机通道中断 (Host channels interrupt)

模块将该位置 1 时，指示模块中一个通道上存在挂起的中断（在主机模式下）。应用程序必须读取主机全体通道中断 (OTG_HS_HAINT) 寄存器，以确定发生中断的通道的准确编号，然后读取相应的主机通道 x 中断 (OTG_HS_HCINTx) 寄存器，以确定引发中断的确切原因。应用程序必须先将 OTG_HS_HCINTx 寄存器的相应状态位清零，之后才能将该位清零。

注意：仅可在主机模式下访问。

位 24 **HPRTINT**：主机端口中断 (Host port interrupt)

模块将该位置 1 时，指示主机模式下 OTG_HS 控制器端口的状态发生变化。应用程序必须读取主机端口控制和状态 (OTG_HS_HPRT) 寄存器，以确定引发此中断的确切事件。应用程序必须先将主机端口控制和状态寄存器的相应状态位清零，之后才能将该位清零。

注意：仅可在主机模式下访问。

位 23 保留，必须保持复位值。

位 22 **DATAFSUSP**：数据获取挂起 (Data fetch suspended)

该中断仅在 DMA 模式下有效。该中断指示，模块因 TxFIFO 空间或请求队列空间不可用而停止为 IN 端点获取数据。应用程序将该中断用于端点不匹配算法中。例如，在检测到端点不匹配后，应用程序将执行以下操作：

- 将全局非周期性 IN NAK 握手信号置 1
- 禁止 IN 端点
- 清空 FIFO
- 根据 IN 令牌序列学习队列确定令牌序列
- 重新使能端点
- 如果全局非周期性 IN NAK 已清零但模块尚未为 IN 端点获取数据，同时又已接收到 IN 令牌，则清零全局非周期性 IN NAK 握手信号：模块将产生“FIFO 为空时接收到 IN 令牌”中断。然后，OTG 将 NAK 响应发送到主机。为避免这种情况的发生，应用程序可以检查 OTG_FS_GINTSTS 中的 FetSusp 中断，该中断可确保在 FIFO 存满后再将全局 NAK 握手信号清零。或者，应用程序可以在将全局 IN NAK 握手信号清零时屏蔽“当 FIFO 为空时接收到 IN 令牌”中断。

位 21 **IPXFR**：未完成周期性传输 (Incomplete periodic transfer)

在主机模式下，如果存在仍处于挂起状态的未完成周期性事务，而这些事务计划在当前帧期间完成，则模块会将该中断位置 1。

注意：仅可在主机模式下访问。

INCOMPISOOUT：未完成 OUT 同步传输 (Incomplete isochronous OUT transfer)

在设备模式下，模块将该中断置 1 时，指示当前帧中至少有一个同步 OUT 端点上的传输未完成。该中断随该寄存器中的周期性帧结束中断 (EOPF) 位一同触发。

注意：仅可在设备模式下访问。

位 20 **IISOIXFR**：未完成 IN 同步传输 (Incomplete isochronous IN transfer)

模块将该中断置 1 时，指示当前帧中至少有一个同步 IN 端点上的传输未完成。该中断随该寄存器中的周期性帧结束中断 (EOPF) 位一同触发。

注意：仅可在设备模式下访问。

位 19 **OEPINT**：OUT 端点中断 (OUT endpoint interrupt)

模块将该位置 1 时，指示模块中一个 OUT 端点上存在挂起的中断（在设备模式下）。应用程序必须读取设备全体端点中断 (OTG_HS_DAINT) 寄存器，以确定发生中断的 OUT 端点的准确编号，然后读取相应的设备 OUT 端点 x 中断 (OTG_HS_DOEPINTx) 寄存器，以确定引发中断的确切原因。应用程序必须先将相应 OTG_HS_DOEPINTx 寄存器的相应状态位清零，之后才能将该位清零。

注意：仅可在设备模式下访问。

位 18 **IEPINT**：IN 端点中断 (IN endpoint interrupt)

模块将该位置 1 时，指示模块中一个 IN 端点上存在挂起的中断（在设备模式下）。应用程序必须读取设备全体端点中断 (OTG_HS_DAINT) 寄存器，以确定发生中断的 IN 端点的准确编号，然后读取相应的设备 IN 端点 x 中断 (OTG_HS_DIEPINTx) 寄存器，以确定引发中断的确切原因。应用程序必须先将相应 OTG_HS_DIEPINTx 寄存器的相应状态位清零，之后才能将该位清零。

注意：仅可在设备模式下访问。

位 17:16 保留，必须保持复位值。

位 15 **EOPF**：周期性帧结束中断 (End of periodic frame interrupt)

指示当前帧已达到设备配置寄存器中周期性帧间隔字段（OTG_HS_DCFG 中的 PFIVL 位）所指定的周期。

注意：仅可在设备模式下访问。

位 14 **ISOODRP**：丢弃同步 OUT 数据包中断 (Isochronous OUT packet dropped interrupt)

如果由于 RxFIFO 空间不足，无法容纳同步 OUT 端点的最大数据包，从而导致模块无法向 RxFIFO 写入同步 OUT 数据包，模块会将该位置 1。

注意：仅可在设备模式下访问。

位 13 **ENUMDNE**：枚举完成 (Enumeration done)

模块将该位置 1 时，指示速度枚举已完成。应用程序必须读取设备状态 (OTG_HS_DSTS) 寄存器来获取枚举的速度。

注意：仅可在设备模式下访问。

位 12 **USBRST**：USB 复位 (USB reset)

模块将该位置 1 时，指示在 USB 上检测到复位信号。

注意：仅可在设备模式下访问。

位 11 **USBSUSP**：USB 挂起 (USB suspend)

模块将该位置 1 时，指示在 USB 上检测到挂起状态。当 USB 总线上的空闲状态保持 3 ms，模块便会进入挂起状态。

注意：仅可在设备模式下访问。

位 10 **ESUSP**：早期挂起 (Early suspend)

模块将该位置 1 时，指示已检测到 USB 处于空闲状态的时间达到 3 ms。

注意：仅可在设备模式下访问。

位 9:8 保留，必须保持复位值。

位 7 **GONAKEFF**：全局 OUT NAK 有效 (Global OUT NAK effective)

指示设备控制寄存器中由应用程序设置的“将全局 OUT NAK 置 1”位（OTG_HS_DCTL 中的 SGONAK 位）已在模块中生效。通过写设备控制寄存器中的“将全局 OUT NAK 清零”位（OTG_HS_DCTL 中的 CGONAK 位），可将该位清零。

注意：仅可在设备模式下访问。

位 6 **GINAKEFF**：全局非周期性 IN NAK 有效 (Global IN nonperiodic NAK effective)

指示设备控制寄存器中由应用程序设置的“将全局非周期性 IN NAK 置 1”位（OTG_HS_DCTL 中的 SGINAK 位）已在模块中生效。也就是说，模块已对应用程序设置的全局 IN NAK 位进行采样，结果已生效。通过清零设备控制寄存器中的“将全局非周期性 IN NAK 清零”位（OTG_HS_DCTL 中的 CGINAK 位），可将该位清零。

此中断不一定表示 USB 上已发送了一个 NAK 握手信号。STALL 位优先级高于 NAK 位。

注意：仅可在设备模式下访问。

位 5 **NPTXFE**：非周期性 TxFIFO 空 (Nonperiodic TxFIFO empty)

当非周期性 TxFIFO 为半空或全空状态，且非周期性发送请求队列中至少存在可写入一个条目的空间时，将触发该中断。该 TxFIFO 为半空状态还是全空状态由 OTG_HS_GAHBCFG 寄存器中的非周期性 TxFIFO 空级别位（OTG_HS_GAHBCFG 中的 TXFELVL 位）决定。

注意：仅可在主机模式下访问。

位 4 **RXFLVL**：RxFIFO 非空 (RxFIFO nonempty)

指示 RxFIFO 中至少有一个数据包等待读取。

注意：在主机模式和设备模式下均可访问。

位 3 **SOF**：帧起始 (Start of frame)

在主机模式下，模块将该位置 1 时，指示 USB 上已发送一个 SOF (FS) 或 Keep-Alive (LS) 信号。应用程序必须将此位置 1 才可清除该中断。

在设备模式下，模块将该位置 1 时，指示 USB 上已接收到一个 SOF 令牌。应用程序可通过读取设备状态寄存器来获得当前的帧编号。只有在模块以 FS 模式运行时，才会出现此中断。

注意：在主机模式和设备模式下均可访问。

位 2 **OTGINT**：OTG 中断 (OTG interrupt)

模块将该位置 1 时，指示出现 OTG 协议事件。应用程序必须读取 OTG 中断状态 (OTG_HS_GOTGINT) 寄存器，以确定引发此中断的确切事件。应用程序必须先将 OTG_HS_GOTGINT 寄存器的相应状态位清零，之后才能将该位清零。

注意：在主机模式和设备模式下均可访问。

位 1 **MMIS**：模式不匹配中断 (Mode mismatch interrupt)

当应用程序尝试访问以下寄存器时，模块将该位置 1：
模块运行在设备模式下访问主机模式寄存器
模块运行在主机模式下访问设备模式寄存器
寄存器访问在 AHB 上以 OKAY 响应结束，但该访问在内部被模块忽略并且不会影响模块运行。

注意：在主机模式和设备模式下均可访问。

位 0 **CMOD**：当前工作模式 (Current mode of operation)

指示当前模式。
0：设备模式
1：主机模式

注意：在主机模式和设备模式下均可访问。

OTG_HS 中断屏蔽寄存器 (OTG_HS_GINTMSK)

OTG_HS interrupt mask register

偏移地址：0x018

复位值：0x0000 0000

该寄存器与模块中断寄存器结合使用，以中断应用程序。如果将某个中断位屏蔽，则不会产生与该位相关的中断。但是，与该中断相对应的模块中断 (OTG_HS_GINTSTS) 寄存器位仍会置 1。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WUIM	SRQIM	DISCINT	CIDSCHGM	Reserved	PTXFEM	HCIM	PRTIM	Reserved	FSUSPM	IPXFRM/IISOOXFRM	IISOIXFRM	OEPINT	IEPINT	EPMISM	Reserved	EOPFM	ISOODRPM	ENUMDNEM	USBRST	USBSUSPM	ESUSPM	Reserved		GONAKEFFM	GINAKEFFM	NPTXFEM	RXFLVLM	SOFM	OTGINT	MMISM	Reserved
rw	rw	rw	rw		rw	rw	r		rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw	rw	

位 31 **WUIM**：检测到恢复/远程唤醒中断屏蔽 (Resume/remote wakeup detected interrupt mask)

0：屏蔽中断
1：使能中断

注意：在主机模式和设备模式下均可访问。

位 30 **SRQIM**：检测到会话请求/新会话中断屏蔽 (Session request/new session detected interrupt mask)

0：屏蔽中断
1：使能中断

注意：在主机模式和设备模式下均可访问。

位 29 **DISCINT**：检测到断开连接中断屏蔽 (Disconnect detected interrupt mask)
0：屏蔽中断
1：使能中断
注意：在主机模式和设备模式下均可访问。

位 28 **CIDSCHGM**：连接器 ID 状态更改屏蔽 (Connector ID status change mask)
0：屏蔽中断
1：使能中断
注意：在主机模式和设备模式下均可访问。

位 27 保留，必须保持复位值。

位 26 **PTXFEM**：周期性 TxFIFO 空屏蔽 (Periodic TxFIFO empty mask)
0：屏蔽中断
1：使能中断
注意：仅可在主机模式下访问。

位 25 **HCIM**：主机通道中断屏蔽 (Host channels interrupt mask)
0：屏蔽中断
1：使能中断
注意：仅可在主机模式下访问。

位 24 **PRTIM**：主机端口中断屏蔽 (Host port interrupt mask)
0：屏蔽中断
1：使能中断
注意：仅可在主机模式下访问。

位 23 保留，必须保持复位值。

位 22 **FSUSPM**：数据获取挂起中断屏蔽 (Data fetch suspended mask)
0：屏蔽中断
1：使能中断
注意：仅可在设备模式下访问。

位 21 **IPXFRM**：未完成周期性传输中断屏蔽 (Incomplete periodic transfer mask)
0：屏蔽中断
1：使能中断
注意：仅可在主机模式下访问。
IISOOXFRM：未完成 OUT 同步传输中断屏蔽 (Incomplete isochronous OUT transfer mask)
0：屏蔽中断
1：使能中断
注意：仅可在设备模式下访问。

位 20 **IISOIXFRM**：未完成 IN 同步传输中断屏蔽 (Incomplete isochronous IN transfer mask)
0：屏蔽中断
1：使能中断
注意：仅可在设备模式下访问。

位 19 **OEPINT**：OUT 端点中断屏蔽 (OUT endpoints interrupt mask)
0：屏蔽中断
1：使能中断
注意：仅可在设备模式下访问。

位 18 **IEPINT**：IN 端点中断屏蔽 (IN endpoints interrupt mask)
0：屏蔽中断
1：使能中断
注意：仅可在设备模式下访问。

位 17 **EPMISM**：端点不匹配中断屏蔽 (Endpoint mismatch interrupt mask)

0：屏蔽中断
1：使能中断

注意：仅可在设备模式下访问。

位 16 保留，必须保持复位值。

位 15 **EOPFM**：周期性帧结束中断屏蔽 (End of periodic frame interrupt mask)

0：屏蔽中断
1：使能中断

注意：仅可在设备模式下访问。

位 14 **ISOODRPM**：丢弃同步 OUT 数据包中断屏蔽 (Isochronous OUT packet dropped interrupt mask)

0：屏蔽中断
1：使能中断

注意：仅可在设备模式下访问。

位 13 **ENUMDNEM**：枚举完成中断屏蔽 (Enumeration done mask)

0：屏蔽中断
1：使能中断

注意：仅可在设备模式下访问。

位 12 **USBRST**：USB 复位中断屏蔽 (USB reset mask)

0：屏蔽中断
1：使能中断

注意：仅可在设备模式下访问。

位 11 **USBSUSPM**：USB 挂起中断屏蔽 (USB suspend mask)

0：屏蔽中断
1：使能中断

注意：仅可在设备模式下访问。

位 10 **ESUSPM**：早期挂起中断屏蔽 (Early suspend mask)

0：屏蔽中断
1：使能中断

注意：仅可在设备模式下访问。

位 9:8 保留，必须保持复位值。

位 7 **GONAKEFFM**：全局 OUT NAK 生效中断屏蔽 (Global OUT NAK effective mask)

0：屏蔽中断
1：使能中断

注意：仅可在设备模式下访问。

位 6 **GINAKEFFM**：全局非周期性 IN NAK 生效中断屏蔽 (Global nonperiodic IN NAK effective mask)

0：屏蔽中断
1：使能中断

注意：仅可在设备模式下访问。

位 5 **NPTXFEM**：非周期性 TxFIFO 空中断屏蔽 (Nonperiodic TxFIFO empty mask)

0：屏蔽中断
1：使能中断

注意：在设备模式和主机模式下均可访问。

位 4 **RXFLVLM**：接收 FIFO 非空中断屏蔽 (Receive FIFO nonempty mask)
0：屏蔽中断
1：使能中断
注意：在设备模式和主机模式下均可访问。

位 3 **SOFM**：帧起始中断屏蔽 (Start of frame mask)
0：屏蔽中断
1：使能中断
注意：在设备模式和主机模式下均可访问。

位 2 **OTGINT**：OTG 中断屏蔽 (OTG interrupt mask)
0：屏蔽中断
1：使能中断
注意：在设备模式和主机模式下均可访问。

位 1 **MMISM**：模式不匹配中断屏蔽 (Mode mismatch interrupt mask)
0：屏蔽中断
1：使能中断
注意：在设备模式和主机模式下均可访问。

位 0 保留，必须保持复位值。

OTG_HS 接收状态调试读取/OTG 状态读取和出栈寄存器 (OTG_HS_GRXSTSR/OTG_HS_GRXSTSP)

OTG_HS Receive status debug read/OTG status read and pop registers

读取的偏移地址：0x01C

出栈的偏移地址：0x020

复位值：0x0000 0000

读取接收状态调试读取寄存器将返回接收 FIFO 顶部的内容。读取接收状态读取和出栈寄存器将额外弹出 RxFIFO 顶部的数据条目。

接收状态内容在主机模式和设备模式下的解释不同。当接收 FIFO 为空时，模块会忽略对该寄存器的读取或出栈操作，并返回值 0x0000 0000。仅当模块中断寄存器的接收 FIFO 非空位（OTG_FS_GINTSTS 中的 RXFLVL 位）置位时，应用程序才能弹出接收状态 FIFO。

主机模式：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved											PKTSTS				DPID		BCNT														CHNUM			
											r				r		r														r			

位 31:21 保留，必须保持复位值。

位 20:17 **PKTSTS**：数据包状态 (Packet status)
指示接收的数据包的状态
0010：接收到 IN 数据包
0011：IN 传输完成（触发中断）
0101：数据同步错误（触发中断）
0111：暂停通道（触发中断）
其它值：保留

位 16:15 **DPID**：数据 PID (Data PID)
指示接收的数据包的数据 PID
00：DATA0
10：DATA1
01：DATA2
11：MDATA

位 14:4 **BCNT**：字节计数 (Byte count)
指示接收的 IN 数据包的字节数。

位 3:0 **CHNUM**：通道编号 (Channel number)
指示当前接收的数据包所属的通道编号。

设备模式：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved							FRMNUM				PKTSTS				DPID		BCNT												EPNUM			
							r				r				r		r												r			

位 31:25 保留，必须保持复位值。

位 24:21 **FRMNUM**：帧编号 (Frame number)
这是在 USB 上接收的数据包帧编号的 4 个最低有效位。只有当支持同步 OUT 端点时才支持此字段。

位 20:17 **PKTSTS**：数据包状态 (Packet status)
指示接收的数据包的状态
0001：全局 OUT NAK（触发中断）
0010：接收到 OUT 数据包
0011：OUT 传输完成（触发中断）
0100：SETUP 事务完成（触发中断）
0110：接收到 SETUP 数据包
其它值：保留

位 16:15 **DPID**：数据 PID (Data PID)
指示接收的 OUT 数据包的数据 PID
00：DATA0
10：DATA1
01：DATA2
11：MDATA

位 14:4 **BCNT**：字节计数 (Byte count)
指示接收的数据包的字节数。

位 3:0 **EPNUM**：端点编号 (Endpoint number)
指示当前接收的数据包所属的端点编号。

OTG_HS 接收 FIFO 大小寄存器 (OTG_HS_GRXFSIZ)

OTG_HS Receive FIFO size register

偏移地址：0x024

复位值：0x0000 0200

此应用程序可以对必须分配给 RxFIFO 的 RAM 大小进行编程。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																RXFD															
																r/rw															

位 31:16 保留，必须保持复位值。

位 15:0 **RXFD**：RxFIFO 深度 (RxFIFO depth)
以 32 位字为单位。
最小值为 16
最大值为 1024
上电复位值为最大 Rx 数据 FIFO 深度。

OTG_HS 非周期性发送 FIFO 大小/端点 0 发送 FIFO 大小寄存器 (OTG_HS_GNPTXFSIZ/OTG_HS_TX0FSIZ)

OTG_HS nonperiodic transmit FIFO size/Endpoint 0 transmit FIFO size register

偏移地址：0x028

复位值：0x0000 0200

主机模式：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NPTXFD																NPTXFSA															
r/rw																r/rw															

位 31:16 **NPTXFD**：非周期性 TxFIFO 深度 (Nonperiodic TxFIFO depth)
以 32 位字为单位。
最小值为 16
最大值为 1024

位 15:0 **NPTXFSA**：非周期性发送 RAM 起始地址 (Nonperiodic transmit RAM start address)
此字段包含非周期性发送 FIFO RAM 的存储器起始地址。

设备模式：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TX0FD																TX0FSA															
r/rw																r/rw															

位 31:16 **T0XFD**：端点 0 TxFIFO 深度 (Endpoint 0 TxFIFO depth)
以 32 位字为单位。
最小值为 16
最大值为 256

位 15:0 **TX0FSA**：端点 0 发送 RAM 起始地址 (Endpoint 0 transmit RAM start address)
此字段包含端点 0 发送 FIFO RAM 的存储器起始地址。

OTG_HS 非周期性发送 FIFO/队列状态寄存器 (OTG_HS_GNPTXSTS)

OTG_HS nonperiodic transmit FIFO/queue status register

偏移地址：0x02C

复位值：0x0008 0200

注意： *在设备模式下，此寄存器无效。*

此只读寄存器包含非周期性 TxFIFO 和非周期性发送请求队列的空闲空间信息。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	NPTXQTOP							NPTQXSAV								NPTXFSAV															
	r							r								r															

位 31 保留，必须保持复位值。

位 30:24 **NPTXQTOP**：非周期性发送请求队列顶部 (Top of the nonperiodic transmit request queue)
非周期性 Tx 请求队列中 MAC 目前正在处理的条目。
位 [30:27]：通道/端点编号 (Channel/endpoint number)
位 [26:25]：
- 00：IN/OUT 令牌
- 01：长度为零的发送数据包（设备 IN/主机 OUT）
- 10：PING/CSPLIT 令牌
- 11：通道停止命令

位 [24]：结束（所选通道/端点的最后一个条目）(Terminate (last entry for selected channel/endpoint))

位 23:16 **NPTQXSAV**：非周期性发送请求队列可用空间 (Nonperiodic transmit request queue space available)
指示非周期性发送请求队列中的可用空间大小。在主机模式下，此队列保存 IN 和 OUT 请求。设备模式仅具有 IN 请求。
00：非周期性发送请求队列已满
01：dx1 位置可用
10：dx2 位置可用
bxn：dx*n* 位置可用 ($0 \leq n \leq$ dx8)
其它值：保留

位 15:0 **NPTXFSAV**：非周期性 TxFIFO 可用空间 (Nonperiodic TxFIFO space available)
指示非周期性 TxFIFO 中的可用空间大小。
以 32 位字为单位。
00：非周期性 TxFIFO 已满
01：dx1 个字可用
10：dx2 个字可用
0xn：dx*n* 个字可用（其中，$0 \le n \le$ dx1024）
其它值：保留

OTG_HS I²C 访问寄存器 (OTG_HS_GI2CCTL)

OTG_HS I2C access register

偏移地址：0x030

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BSYDNE	RW	Reserved	I2CDATSE0	I2CDEV ADR		Reserved	ACK	I2CEN	ADDR							REGADDR								RWDATA							
rw	rw		rw	rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31 **BSYDNE**：I2C 忙碌/完成 (I2C Busy/Done)
应用程序将该位置 1 可启动 I²C 接口上的请求。传输完成后，模块会将此位清零。只要该位置 1，即表示 I²C 接口忙碌，应用程序便无法启动该接口的另一请求。

位 30 **RW**：读/写指示符 (Read/Write Indicator)
该位指示接口上必须执行读寄存器传输还是写寄存器传输。
0：写
1：读
注意：寄存器不支持突发读/写操作。

位 29 保留，必须保持复位值。

位 28 **I2CDATSE0**：I²C DatSe0 USB 模式 (I²C DatSe0 USB mode)
该位用于选择全速接口 USB 模式。
0：VP_VM USB 模式
1：DAT_SE0 USB 模式

位 27:26 **I2CDEVADR**：I²C 器件地址 (I2C Device Address)
利用此位可选择模块用于进行 OTG 信号传输所使用的 USB 1.1 全速串行收发器的 I²C 从器件的地址。

位 25 保留，必须保持复位值。

位 24 **ACK**：I²C 应答 (I²C ACK)
该位指示是否从 I²C 从器件中接收到 ACK 应答。当应用程序启动 I²C 通信，模块将 BSYDNE 位清零后，该位才有效。
0：NAK
1：ACK

位 23 **I2CEN**：I²C 使能 (I2C Enable)
该位可使能 I²C 主器件在 I²C 接口上启动通信事务。

位 22:16 **ADDR**：I^2C 地址 (I2C Address)

此位是 7 位 I^2C 器件地址，应用程序利用此地址来访问任何外部 I^2C 从器件，包括 USB 1.1 OTG 全速串行收发器上的 I^2C 从器件。

位 15:8 **REGADDR**：I^2C 寄存器地址 (I2C Register Address)

这些位用于编程要读取/写入的寄存器的地址。

位 7:0 **RWDATA**：I^2C 读/写数据 (I2C Read/Write Data)

执行寄存器读操作之后，这些位可存储应用程序的读取数据。

执行写操作期间，应用程序可使用此寄存器编程要写入寄存器的数据。

OTG_HS 通用模块配置寄存器 (OTG_HS_GCCFG)

OTG_HS general core configuration register

偏移地址：0x038

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved										NOVBUSSENS	SOFOUTEN	VBUSBSEN	VBUSASEN	I2CPADEN	.PWRDWN	Reserved															
										rw	rw	rw	rw	rw	rw																

位 31:22 保留，必须保持复位值。

位 21 **NOVBUSSENS**：V_{BUS} 感应禁止选项 (VBUS sensing disable option)

该位置 1 时，在内部将 V_{BUS} 视为始终处于 V_{BUS} 有效电平 (5 V)。该选项消除了对 V_{BUS} 引脚的需求，使该引脚可自由用于其它用途，例如该引脚上的其他功能。V_{BUS} 连接可重映射到其它通用输入引脚，并由软件监视。

此选项仅适用于只作主机或只作设备。

0：硬件支持 V_{BUS} 感应

1：硬件不支持 V_{BUS} 感应

位 20 **SOFOUTEN**：SOF 输出使能 (SOF output enable)

0：SOF 脉冲不从引脚输出

1：SOF 脉冲可从引脚输出

位 19 **VBUSBSEN**：使能“B”器件的 V_{BUS} 感应功能 (Enable the VBUS sensing "B" device)

0：禁止“B”器件的 V_{BUS} 感应功能

1：使能“B”器件的 V_{BUS} 感应功能

位 18 **VBUSASEN**：使能“A”器件的 V_{BUS} 感应功能 (Enable the V_{BUS} sensing "A" device)

0：禁止“A”器件的 V_{BUS} 感应功能

1：使能“A”器件的 V_{BUS} 感应功能

位 17 **I2CPADEN**：使能 I^2C 总线，以连接 I^2C 接口的外部 PHY (Enable I2C bus connection for the external I2C PHY interface)

0：禁止 I^2C 总线

1：使能 I^2C 总线

位 16 **PWRDWN**：掉电 (Power down)
用于在发送/接收时激活收发器
0：掉电激活
1：掉电停用（“收发器激活”）

位 15:0 保留，必须保持复位值。

OTG_HS 模块 ID 寄存器 (OTG_HS_CID)

OTG_HS core ID register

偏移地址：0x03C

复位值：0x0000 1200

该寄存器为只读寄存器，包含产品 ID。

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

位 31:0 **PRODUCT_ID**：产品 ID 字段 (Product ID field)
可通过应用程序编程的 ID 字段。

OTG_HS 主机周期性发送 FIFO 大小寄存器 (OTG_HS_HPTXFSIZ)

OTG_HS Host periodic transmit FIFO size register

偏移地址：0x100

复位值：0x0200 0600

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PTXFD																PTXSA															
r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw

位 31:16 **PTXFD**：主机周期性 TxFIFO 深度 (Host periodic TxFIFO depth)
以 32 位字为单位。
最小值为 16
最大值为 512

位 15:0 **PTXSA**：主机周期性 TxFIFO 起始地址 (Host periodic TxFIFO start address)
上电复位值是最大 Rx 数据 FIFO 深度与最大非周期性 Tx 数据 FIFO 深度之和。

OTG_HS 设备 IN 端点发送 FIFO 大小寄存器 (OTG_HS_DIEPTXFx)（x = 1..7，其中 x 为 FIFO 编号）

OTG_HS device IN endpoint transmit FIFO size register

偏移地址：0x104 + (FIFO_number – 1) × 0x04

复位值：0x02000400

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INEPTXFD																INEPTXSA															
r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw

位 31:16 **INEPTXFD**：IN 端点 TxFIFO 深度 (IN endpoint TxFIFO depth)

以 32 位字为单位。
最小值为 16
最大值为 512
上电复位值为最大 IN 端点 FIFO 深度。

位 15:0 **INEPTXSA**：IN 端点发送 FIFOx RAM 起始地址 (IN endpoint FIFOx transmit RAM start address)

此字段包含 IN 端点发送 FIFOx 的存储器起始地址。该地址必须与 32 位存储器位置对齐。

31.12.3 主机模式寄存器

除非特别说明，否则寄存器描述中的位值以二进制表示。

主机模式寄存器会影响主机模式下的模块操作。在设备模式下不得访问主机模式寄存器，因为产生的结果不明确。主机模式寄存器可进行如下分类：

OTG_HS 主机配置寄存器 (OTG_HS_HCFG)

OTG_HS host configuration register

偏移地址：0x400

复位值：0x0000 0000

此寄存器将在上电后对模块进行配置。请勿在初始化主机后更改此寄存器。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																													FSLSS	FSLSPCS	
																													r	rw	rw

位 31:3 保留，必须保持复位值。

位 2 **FSLSS**：仅支持 FS 和 LS (FS- and LS-only support)

应用程序使用此位控制模块的枚举速度。使用此位，应用程序可使模块工作为 FS 主机，即使所连接的设备支持 HS 通信也是如此。请勿在初始编程后更改此字段。
0：HS/FS/LS，取决于所连接设备支持的最大速度
1：仅限 FS/LS，即使所连接设备可支持 HS（只读）

位 1:0 **FSLSPCS**：FS/LS PHY 时钟选择 (FS/LS PHY clock select)

当模块处于 FS 主机模式时：

01：PHY 时钟以 48 MHz 运行

其它值：保留

当模块处于 LS 主机模式时：

00：保留

01：PHY 时钟以 48 MHz 运行。

10：选择 6 MHz PHY 时钟频率

11：保留

注意：当设备连上主机时，必须依照所连接设备的速度设置 FSLSPCS 位。更改此位后必须执行软件复位。

OTG_HS 主机帧间隔寄存器 (OTG_HS_HFIR)

OTG_HS Host frame interval register

偏移地址：0x404

复位值：0x0000 EA60

此寄存器用于存储 OTG_HS 控制器对已连接设备当前速度所设定的帧间隔信息。

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

位 31:16 保留，必须保持复位值。

位 15:0 **FRIVL**：帧间隔 (Frame interval)

应用程序在此字段编程的值用于指定两个连续 SOF (FS)、micro-SOF (HS) 或 Keep-Alive 令牌 (LS) 之间的间隔。此字段包含构成所需帧间隔的 PHY 时钟数。只有将主机端口控制和状态寄存器的端口使能位（OTG_HS_HPRT 的 PENA 位）置 1 后，应用程序才能向此寄存器中写入值。如果未对值进行编程，模块将根据在主机配置寄存器的 FS/LS PHY 时钟选择字段（OTG_HS_HCFG 中的 FSLSPCS）中指定的 PHY 时钟来计算：

帧持续时间 × PHY 时钟频率

注意：只要应用程序需要更改帧间隔时间，即可对 FRIVL 位进行修改。

OTG_HS 主机帧编号/帧剩余时间寄存器 (OTG_HS_HFNUM)

OTG_HS host frame number/frame time remaining register

偏移地址：0x408

复位值：0x0000 3FFF

此寄存器用于指示当前帧编号。它还指示当前帧的剩余时间（以 PHY 时钟数为单位）。

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

位 31:16 **FTREM**：帧剩余时间 (Frame time remaining)

指示当前帧的剩余时间（以 PHY 时钟数为单位）。每过去 1 个 PHY 时钟，此字段递减 1。当值达到零时，此字段将重新装载帧间隔寄存器中的值，并由模块在 USB 上发送一个新 SOF。

位 15:0 **FRNUM**：帧编号 (Frame number)

当在 USB 上发送 1 个新 SOF 时此字段的值将递增 1，当达到 0x3FFF 时会清零。

OTG_HS_Host 周期性发送 FIFO/队列状态寄存器 (OTG_HS_HPTXSTS)

OTG_HS_Host periodic transmit FIFO/queue status register

偏移地址：0x410

复位值：0x0008 0100

此只读寄存器包含周期性 TxFIFO 和周期性发送请求队列的空闲空间信息。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PTXQTOP								PTXQSAV								PTXFSAVL															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:24 **PTXQTOP**：周期性发送请求队列顶部 (Top of the periodic transmit request queue)

指示周期性 Tx 请求队列中 MAC 当前正在处理的项。

该寄存器用于调试。

位 [31]：奇数/偶数帧 (Odd/Even frame)

- 0：以偶数（微）帧发送
- 1：以奇数（微）帧发送

位 [30:27]：通道/端点编号 (Channel/endpoint number)

位 [26:25]：类型 (Type)

- 00：输入/输出
- 01：零长度数据包
- 11：禁止通道命令

位 [24]：结束（所选通道/端点的最后一个条目）(Terminate (last entry for the selected channel/endpoint))

位 23:16 **PTXQSAV**：周期性发送请求队列可用空间 (Periodic transmit request queue space available)

指示可供写入的周期性发送请求队列的空闲位置的数量。该队列既包含 IN 请求，又包含 OUT 请求。

00：周期性发送请求队列已满

01：dx1 位置可用

10：dx2 位置可用

bxn：dxn 位置可用 (0 ≤ dxn ≤ PTXFD)

其它值：保留

位 15:0 **PTXFSAVL**：周期性发送数据 FIFO 可用空间 (Periodic transmit data FIFO space available)

指示可供写入的周期性 TxFIFO 的空闲位置的数量。

以 32 位字为单位

0000：周期性 TxFIFO 已满

0001：dx1 个字可用

0010：dx2 个字可用

bxn：dxn 个字可用（其中 0 ≤ dxn ≤ dx512）

其它值：保留

OTG_HS 主机全体通道中断寄存器 (OTG_HS_HAINT)

OTG_HS Host all channels interrupt register

偏移地址：0x414

复位值：0x0000 000

当通道上有事件发生时，主机全体通道中断寄存器会使用模块中断寄存器中的主机通道中断位（OTG_HS_GINTSTS 中的 HCINT 位）中断应用程序。相关内容如*图 383* 所示。每个通道对应 1 个中断位，最多有 16 个位。当应用程序通过相应主机通道 x 中断寄存器清零中断时，该寄存器中的位也会清零。

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

位 31:16 保留，必须保持复位值。

位 15:0 **HAINT**：通道中断 (Channel interrupt)
每个通道对应一位：通道 0 对应位 0，通道 15 对应位 15

OTG_HS 主机全体通道中断屏蔽寄存器 (OTG_HS_HAINTMSK)

OTG_HS host all channels interrupt mask register

偏移地址：0x418

复位值：0x0000 0000

主机全体通道中断屏蔽寄存器与主机全体通道中断寄存器结合使用，进而在通道上发生事件时中断应用程序。每个通道对应 1 个中断屏蔽位，最多有 16 个位。

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

位 31:16 保留，必须保持复位值。

位 15:0 **HAINTM**：通道中断屏蔽 (Channel interrupt mask)
0：屏蔽中断
1：使能中断
每个通道对应一位：通道 0 对应位 0，通道 15 对应位 15

OTG_HS 主机端口控制和状态寄存器 (OTG_HS_HPRT)

OTG_HS host port control and status register

偏移地址：0x440

复位值：0x0000 0000

该寄存器仅在主机模式下可用。当前，OTG 主机仅支持一个端口。

该寄存器包含与 USB 端口相关的信息，例如 USB 复位、使能、挂起、恢复、连接状态以及测试模式。具体如*图 383* 中所示。该寄存器中的 rc_w1 位可通过模块中断寄存器中的主机端口中断位（OTG_HS_GINTSTS 中的 HPRTINT 位）触发应用程序中断。发生端口中断时，应用程序必须读取该寄存器，并将引起中断的位清零。对于 rc_w1 位，应用程序必须向该位写入 1 以清除该中断。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved													PSPD		PTCTL				PPWR	PLSTS		Reserved	PRST	PSUSP	PRES	POCCHNG	POCA	PENCHNG	PENA	PCDET	PCSTS
													r	r	rw	rw	rw	rw	rw	r	r		rw	rs	rw	rc_w1	r	rc_w1	rc_w0	rc_w1	r

位 31:19 保留，必须保持复位值。

位 18:17 **PSPD**：端口速度 (Port speed)

指示连接到该端口的设备的速度。

00：高速

01：全速

10：低速

11：保留

位 16:13 **PTCTL**：端口测试控制 (Port test control)

应用程序向该字段写入一个非零值，以将端口置于测试模式，同时端口上会产生对应模式的信号。

0000：测试模式禁止

0001：Test_J 模式

0010：Test_K 模式

0011：Test_SE0_NAK 模式

0100：Test_Packet 模式

0101：Test_Force_Enable

其它值：保留

位 12 **PPWR**：端口电源 (Port power)

应用程序使用该字段控制该端口的电源，且发生过流情况时，模块会将该位清零。

0：掉电

1：通电

位 11:10 **PLSTS**：端口线状态 (Port line status)

指示 USB 数据线的当前逻辑电平

位 [10]：OTG_HS_FS_DP 的逻辑电平 (Logic level of OTG_HS_FS_DP)

位 [11]：OTG_HS_FS_DM 的逻辑电平 (Logic level of OTG_HS_FS_DM)

位 9 保留，必须保持复位值。

位 8 **PRST**：端口复位 (Port reset)

应用程序将该位置 1 时，会在该端口上启动复位序列。应用程序必须为复位周期定时，并在复位序列完成后将该位清零。

0：端口未处于复位状态

1：端口处于复位状态

应用程序必须将该位置 1 并最少保持 10 ms，以在端口上启动复位。除所需的最少持续时间之外，在将该位清零前，应用程序可将该位置 1 的状态再保持 10 ms，即使 USB 标准并没有设置最大限制。

高速：50 ms

全速/低速：10 ms

位 7 **PSUSP**：端口挂起 (Port suspend)

应用程序将此位置 1 以将此端口置于挂起模式。只有此位置 1 时，模块才会停止发送 SOF。要停止 PHY 时钟，应用程序必须将端口时钟停止位置 1，这会使能 PHY 的挂起输入引脚。

此位的读取值反映该端口的当前挂起状态。检测到远程唤醒信号，或者应用程序将此寄存器中的端口复位位或端口恢复位置 1 后，模块可将此位清零；或应用程序将模块中断寄存器中的检测到恢复/远程唤醒中断位或检测到断开连接中断位（分别为 OTG_HS_GINTSTS 中的 WKUINT 或 DISCINT）置 1，模块也可将此位清零。

0：端口未处于挂起模式

1：端口处于挂起模式

位 6 **PRES**：端口恢复 (Port resume)

应用程序将此位置 1 以在该端口上驱动恢复信号。模块会持续驱动恢复信号直到应用程序将此位清零。

如模块中断寄存器中的检测到端口恢复/远程唤醒中断位（OTG_HS_GINTSTS 中 WKUINT 位）指示，如果模块检测到 USB 远程唤醒序列，则开始驱动恢复信号，而无需应用程序进行干预；如果模块检测到断开连接的情况，则将此位清零。此位的读取值指示当前模块是否正在驱动恢复信号。

0：不驱动恢复信号

1：驱动恢复信号

位 5 **POCCHNG**：端口过流变化 (Port overcurrent change)

该寄存器中端口过流激活位（位 4）状态发生变化时，模块将此位置 1。

位 4 **POCA**：端口过流激活 (Port overcurrent active)

此位指示端口的过流状况。

0：无过流状况

1：有过流状况

位 3 **PENCHNG**：端口使能/禁止变化 (Port enable/disable change)

该寄存器中的端口使能位 [2] 的状态发生变化时，模块将此位置 1。

位 2 **PENA**：端口使能 (Port enable)

端口执行复位序列后，只能由模块使能，并且可以由过流状况、断开连接状况或应用程序将此位清零来禁止。应用程序无法通过对寄存器执行写操作将此位置 1。只能将此位清零来禁止端口。对此位的操作不会触发应用程序的任何中断。

0：禁止端口

1：使能端口

位 1 **PCDET**：检测到端口连接 (Port connect detected)

当检测到设备连接时，模块将此位置 1，以使用模块中断寄存器中的主机端口中断位（OTG_HS_GINTSTS 中的 HPRTINT 位）触发应用程序的中断。应用程序必须将此位置 1 才可清除该中断。

位 0 **PCSTS**：端口连接状态 (Port connect status)

0：端口未连接设备

1：端口已连接设备

OTG_HS 主机通道 x 特性寄存器 (OTG_HS_HCCHARx)（x = 0..11，其中 x = 通道编号）

OTG_HS host channel-x characteristics register

偏移地址：0x500 + (通道编号 × 0x20)

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CHENA	CHDIS	ODDFRM	DAD							MC		EPTYP		LSDEV	Reserved	EPDIR	EPNUM				MPSIZ										
rs	rs	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31 **CHENA**：通道使能 (Channel enable)

此字段由应用程序软件置 1，并由 OTG 主机硬件清零。

0：禁止通道 (Channel disabled)

1：使能通道

位 30 **CHDIS**：通道禁止 (Channel disable)

应用程序将此位置 1 以停止通过通道发送/接收数据，即使通过该通道的传输还未完成，停止操作仍然生效。应用程序必须等待禁止通道的中断以确认通道已经被禁止。

位 29 **ODDFRM**：奇数帧 (Odd frame)

此字段由应用程序置位或复位，以分别指示 OTG 主机必须传输奇数帧或偶数帧。此字段只适用于周期性（同步和中断）事务。

0：偶数（微）帧

1：奇数（微）帧

位 28:22 **DAD**：设备地址 (Device address)

此字段用于指定要与该主机通信的特定设备。

位 21:20 **MC**：多重计数 (MC)/错误计数 (EC) (Multi Count (MC)/Error Count (EC))

– 当主机通道 x 分离控制寄存器 (OTG_HS_HCSPLTx) 中的分离使能位 (SPLITEN) 复位 (0) 时，此字段向主机指示此周期性端点每微帧必须执行的事务数。对于非周期性传输，此字段指定内部 DMA 引擎更改仲裁之前此通道要获取的数据包数。

00：保留，会产生不明确的结果对该字段的操作

01：1 个事务

b10：此端点每微帧需要发出 2 个事务

11：此端点每微帧需要发出 3 个事务。

– 当 OTG_HS_HCSPLTx 中的 SPLITEN 位置 (1) 时，此字段指示周期性分离事务发生错误时要对该事务执行的立即重试次数。此字段至少须置为 01。

位 19:18 **EPTYP**：端点类型 (Endpoint type)

指示选择的传输类型。

00：控制

01：同步

10：批量

11：中断

位 17 **LSDEV**：低速设备 (Low-speed device)

此字段由应用程序置 1，表示此通道正在与一个低速设备进行通信。

位 16 保留，必须保持复位值。

位 15 **EPDIR**：端点方向 (Endpoint direction)
指示通信事务的方向是输入还是输出。
0：输出
1：输入

位 14:11 **EPNUM**：端点编号 (Endpoint number)
指示要与该主机通道通信的 USB 设备的端点号。

位 10:0 **MPSIZ**：最大数据包大小 (Maximum packet size)
指示与该主机通道通信的设备端点的最大数据包大小。

OTG_HS 主机通道 x 分离控制寄存器 (OTG_HS_HCSPLTx)（x = 0..11，其中 x = 通道编号）

OTG_HS host channel-x split control register

偏移地址：0x504 + (通道编号 × 0x20)

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPLITEN	Reserved														COMPLSPLT	XACTPOS		HUBADDR							PRTADDR						
rw															rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31 **SPLITEN**：分离使能 (Split enable)
应用程序将此位置 1 时，指示允许该通道执行分离通信事务。

位 30:17 保留，必须保持复位值。

位 16 **COMPLSPLT**：执行完全分离 (Do complete split)
应用程序将此位置 1 时，可请求 OTG 主机执行完全分离通信事务。

位 15:14 **XACTPOS**：事务位置 (Transaction position)
此字段用于决定随各 OUT 事务发送全部、第一个、中间还是最后一个有效负载。
11：全部。指此事务的全部数据有效负载（小于或等于 188 字节）
10：起始。指此事务的第一个数据有效负载（大于 188 字节）
00：中间。指此事务的中间有效负载（大于 188 字节）
01：结尾。指此事务的最后一个有效负载（大于 188 字节）

位 13:7 **HUBADDR**：集线器地址 (Hub address)
此字段存储事务转发器的集线器设备地址。

位 6:0 **PRTADDR**：端口地址 (Port address)
此字段是接收方事务转发器的端口编号。

OTG_HS 主机通道 x 中断寄存器 (OTG_HS_HCINTx)（x = 0..11，其中 x = 通道编号）

OTG_HS host channel-x interrupt register

偏移地址：0x508 + (通道编号 × 0x20)

复位值：0x0000 0000

该寄存器指示在出现 USB 和 AHB 相关事件时通道的状态。具体如*图 383* 中所示。当模块中断寄存器中的主机通道中断位（OTG_HS_GINTSTS 中的 HCINT 位）置 1 时，应用程序必须读取该寄存器。在对寄存器执行读操作之前，应用程序必须先读取主机全体通道中断 (OTG_HS_HAINT) 寄存器，以获取主机通道 x 中断寄存器的准确通道编号。应用程序必须将该寄存器中的相应位清零，才能将 OTG_HS_HAINT 和 OTG_HS_GINTSTS 寄存器中的对应位清零。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																					DTERR	FRMOR	BBERR	TXERR	NYET	ACK	NAK	STALL	AHBERR	CHH	XFRC
																					rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

位 31:11 保留，必须保持复位值。

位 10 **DTERR**：数据同步错误 (Data toggle error)

位 9 **FRMOR**：帧溢出错误 (Frame overrun)

位 8 **BBERR**：串扰错误 (Babble error)

位 7 **TXERR**：通信事务错误 (Transaction error)
指示 USB 上发生下列错误之一：
CRC 校验失败
超时
位填充错误
错误的 EOP

位 6 **NYET**：收到 NYET 响应 (Response received interrupt)

位 5 **ACK**：收到/发出 ACK 响应 (ACK response received/transmitted interrupt)

位 4 **NAK**：收到 NAK 响应 (NAK response received interrupt)

位 3 **STALL**：收到 STALL 响应 (STALL response received interrupt)

位 2 **AHBERR**：AHB 错误 (AHB error)
仅当处于内部 DMA 模式下且 AHB 读/写操作期间发生 AHB 错误时才生成此错误。应用程序可通过读取相应的 DMA 通道地址寄存器来获取错误地址。

位 1 **CHH**：通道停止 (Channel halted)
因任意 USB 事务错误或为响应应用程序的禁止请求而导致传输非正常结束。

位 0 **XFRC**：传输完成 (Transfer completed)
未出现任何错误，正常完成传输。

OTG_HS 主机通道 x 中断屏蔽寄存器 (OTG_HS_HCINTMSKx)（x = 0..11，其中 x = 通道编号）

OTG_HS host channel-x interrupt mask register

偏移地址：0x50C + (通道编号 × 0x20)

复位值：0x0000 0000

此寄存器反映了先前部分中介绍的各通道状态的屏蔽情况。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																					DTERRM	FRMORM	BBERRM	TXERRM	NYET	ACKM	NAKM	STALLM	AHBERR	CHHM	XFRCM
																					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:11 保留，必须保持复位值。

位 10 **DTERRM**：数据同步错误屏蔽 (Data toggle error mask)
0：屏蔽中断
1：使能中断

位 9 **FRMORM**：帧溢出屏蔽 (Frame overrun mask)
0：屏蔽中断
1：使能中断

位 8 **BBERRM**：串扰错误屏蔽 (Babble error mask)
0：屏蔽中断
1：使能中断

位 7 **TXERRM**：通信事务错误屏蔽 (Transaction error mask)
0：屏蔽中断
1：使能中断

位 6 **NYET**：NYET 响应接收中断屏蔽 (response received interrupt mask)
0：屏蔽中断
1：使能中断

位 5 **ACKM**：ACK 响应接收/发送中断屏蔽 (ACK response received/transmitted interrupt mask)
0：屏蔽中断
1：使能中断

位 4 **NAKM**：NAK 响应接收中断屏蔽 (NAK response received interrupt mask)
0：屏蔽中断
1：使能中断

位 3 **STALLM**：STALL 响应接收中断屏蔽 (STALL response received interrupt mask)
0：屏蔽中断
1：使能中断

位 2 **AHBERR**：AHB 错误中断屏蔽 (AHB error)
仅当处于内部 DMA 模式下且 AHB 读/写操作期间发生 AHB 错误时才生成此错误。应用程序可通过读取相应通道的 DMA 地址寄存器来获取错误地址。

位 1 **CHHM**：通道停止中断屏蔽 (Channel halted mask)
0：屏蔽中断
1：使能中断

位 0 **XFRCM**：传输完成中断屏蔽 (Transfer completed mask)
0：屏蔽中断
1：使能中断

OTG_HS 主机通道 x 传输大小寄存器 (OTG_HS_HCTSIZx)（x = 0..11，其中 x = 通道编号）

OTG_HS host channel-x transfer size register

偏移地址：0x510 + (通道编号 × 0x20)

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	DPID		PKTCNT										XFRSIZ																		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31 **DOPING**：执行 ping 操作 (Do ping)
此位仅用于 OUT 传输。将此位置 1 可指示主机执行 PING 协议。
注意：请勿针对 IN 传输将此位置 1。如果针对 IN 传输将此位置 1，将禁止通道。

位 30:29 **DPID**：数据 PID (Data PID)
应用程序在此字段设置数据通信的初始同步 PID。主机在此次传输事务过程中保留该字段的设置。
00：DATA0
01：DATA2
10：DATA1
11：MDATA（控制传输）/SETUP（非控制传输）

位 28:19 **PKTCNT**：数据包计数 (Packet count)
应用程序在此字段中设置将要发送 (OUT) 或接收 (IN) 的数据包数。
主机每成功发送或接收一个 OUT/IN 数据包便递减一次计数值。此值达到 0 后，将中断应用程序来指示操作正常完成。

位 18:0 **XFRSIZ**：传输大小 (Transfer size)
对于 OUT 操作，此字段为传输期间主机发送的数据字节数。
对于 IN 操作，此字段为应用程序保留给传输的缓冲区大小。对于 IN 事务（周期性和非周期性），应用程序会将此字段编程为最大数据包大小的整数倍。

OTG_HS 主机通道 x DMA 地址寄存器 (OTG_HS_HCDMAx)（x = 0..11，其中 x = 通道编号）

OTG_HS host channel-x DMA address register

偏移地址：0x514 + (通道编号 × 0x20)

复位值：0x0000 0000

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

位 31:0 **DMAADDR**：DMA 地址 (DMA address)

此字段存储主机从设备端点获取数据或往设备端点发送数据所用 DMA 传输的存储器的地址。每次 AHB 传输结束，该寄存器都会递增。

31.12.4 设备模式寄存器

OTG_HS 设备配置寄存器 (OTG_HS_DCFG)

OTG_HS device configuration register

偏移地址：0x800

复位值：0x0220 0000

此寄存器在上电、执行某些控制命令或枚举后，会将模块配置为设备模式。请勿在初始编程后更改该寄存器。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						PERSCHIVL		Reserved	Reserved										PFIVL		DAD							Reserved	NZLSOHSK	DSPD	
						rw	rw												rw		rw	rw	rw	rw	rw	rw	rw		rw	rw	rw

位 31:26 保留，必须保持复位值。

位 25:24 **PERSCHIVL**：周期性调度间隔 (Periodic scheduling interval)

此字段指定内部 DMA 引擎必须为获取周期性 IN 端点数据分配的时间。根据周期性端点数量的不同，必须将此值指定为 25%、50% 或 75%（微）帧。

- 只要存在活动的周期性端点，内部 DMA 引擎便会为获取周期性 IN 端点数据分配指定的时间
- 没有任何活动周期性端点时，内部 DMA 引擎将仅用于非周期性端点并忽略此字段
- 经过一（微）帧中指定的时间后，DMA 切换为获取非周期性端点上的传输数据

00：25%（微）帧
01：50%（微）帧
10：75%（微）帧
11：保留

位 23:13 保留，必须保持复位值。

位 12:11 **PFIVL**：周期性（微）帧间隔 (Periodic (micro)frame interval)

指示一（微）帧内必须使用周期性（微）帧结束中断通知应用程序的时间点。此功能可用于确定该帧的所有同步通信是否完成。

00：80% 帧间隔
01：85% 帧间隔
10：90% 帧间隔
11：95% 帧间隔

位 10:4 **DAD**：设备地址 (Device address)

应用程序必须在执行每个 SetAddress 控制命令后根据命令参数对该字段进行设置。

位 3 保留，必须保持复位值。

位 2 **NZLSOHSK**：非零长度状态 OUT 握手信号 (Nonzero-length status OUT handshake)

在控制传输状态阶段的 OUT 事务期间，当模块收到非零长度数据包后，应用程序可以使用此字段选择要发送的握手信号。

1：收到非零长度状态 OUT 事务时，回复 STALL 握手信号，收到的 OUT 数据包不发送给应用程序。

0：将收到的 OUT 数据包（零长度或非零长度）发送给应用程序，并基于设备端点控制寄存器中端点的 NAK 和 STALL 位回复握手信号。

位 1:0 **DSPD**：设备速度 (Device speed)

指示应用程序要求模块进行枚举所采用的速度，或应用程序支持的最大速度。但是，实际总线速度只有在完成 chirp 序列后才能确定，同时此速度基于与模块连接的 USB 主机的速度。

00：高速
01：保留
10：保留
11：全速（USB 1.1 收发器时钟为 48 MHz）

OTG_HS 设备控制寄存器 (OTG_HS_DCTL)

OTG_HS device control register

偏移地址：0x804

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																				POPRGDNE	CGONAK	SGONAK	CGINAK	SGINAK	TCTL			GONSTS	GINSTS	SDIS	RWUSIG
																				rw	w	w	w	w	rw	rw	rw	r	r	rw	rw

位 31:12 保留，必须保持复位值。

位 11 **POPRGDNE**：上电编程完成 (Power-on programming done)

应用程序使用此位指示寄存器从掉电模式唤醒后已完成编程。

位 10 **CGONAK**：将全局 OUT NAK 清零 (Clear global OUT NAK)

对此位执行写操作会将全局 OUT NAK 清零。

位 9 **SGONAK**：将全局 OUT NAK 置 1 (Set global OUT NAK)

对此位执行写操作会将全局 OUT NAK 置 1。
应用程序使用此位在所有 OUT 端点发送 NAK 握手信号。
应用程序只有确定模块中断寄存器中全局 OUT NAK 有效位（OTG_HS_GINTSTS 中 GONAKEFF 位）已清零时，才可以将此位置 1。

位 8 **CGINAK**：将全局 IN NAK 清零 (Clear global IN NAK)

对此位执行写操作会将全局 IN NAK 清零。

位 7 **SGINAK**：将全局 IN NAK 置 1 (Set global IN NAK)

对此字段执行写操作会将全局非周期性 IN NAK 置 1。应用程序使用此位使所有非周期性 IN 端点发送 NAK 握手信号。
应用程序只有确定模块中断寄存器中全局 IN NAK 有效位（OTG_HS_GINTSTS 中 GINAKEFF 位）已清零时，才可以将此位置 1。

位 6:4 **TCTL**：测试控制 (Test control)

000：测试模式禁止
001：Test_J 模式
010：Test_K 模式
011：Test_SE0_NAK 模式
100：Test_Packet 模式
101：Test_Force_Enable
其它值：保留

位 3 **GONSTS**：全局 OUT NAK 状态 (Global OUT NAK status)

0：将根据 FIFO 状态和 NAK 和 STALL 位设置发送握手信号。
1：无论 RxFIFO 中是否还有空闲空间都不接收数据。除 SETUP 事务之外，对所有收到的数据包回复 NAK 握手信号。所有同步类型的 OUT 数据包都将被丢弃。

位 2 **GINSTS**：全局 IN NAK 状态 (Global IN NAK status)

0：将根据发送 FIFO 中的数据可用性回复握手信号。
1：使所有非周期性 IN 端点回复 NAK 握手信号，无需考虑发送 FIFO 中的数据可用性。

位 1 **SDIS**：软断开 (Soft disconnect)

应用程序使用该位向 USB OTG 模块发出执行软断开的信号。该位置 1 时，主机不会看到设备已连接，且该设备也不会接收 USB 上的信号。在应用程序将此位清零之前，模块会保持断开状态。
0：正常工作。此位在软断开之后清零，会使主机收到设备已连接的事件。重新连接设备之后，USB 主机会重新启动设备枚举。
1：使主机收到设备断开连接的事件。

位 0 **RWUSIG**：发送远程唤醒信号 (Remote wakeup signaling)

应用程序将此位置 1 时，模块会启动远程发送信号，以唤醒 USB 主机。应用程序必须将此位置 1 以使模块退出挂起状态。根据 USB 2.0 规范，应用程序必须在将此位置 1 之后的 1 ms 到 15 ms 内将其清零。

表 183 显示了为使 USB 主机检测到设备断开连接所需的软断开 (SDIS) 位置 1 的最短时间（取决于设备状态）。为了协调时钟抖动，建议应用程序在指定的最小时间基础上再加入一段延迟。

表 183. 软断开的最小时间

运行速度	设备状态	最小时间
高速	非空闲或挂起（正在通信时）	125 µs
全速	挂起	1 ms + 2.5 µs
全速	空闲	2.5 µs
全速	非空闲或挂起（正在通信时）	2.5 µs

OTG_HS 设备状态寄存器 (OTG_HS_DSTS)

OTG_HS device status register

偏移地址：0x808

复位值：0x0000 0010

此寄存器指示模块在出现 USB 相关事件时的状态。发生中断时，必须从设备全体中断 (OTG_HS_DAINT) 寄存器读取发生中断的端点信息。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved										FNSOF														Reserved				EERR	ENUMSPD		SUSPSTS
										r	r	r	r	r	r	r	r	r	r	r	r	r	r					r	r	r	r

位 31:22 保留，必须保持复位值。

位 21:8 **FNSOF**：接收 SOF 的帧编号 (Frame number of the received SOF)

位 7:4 保留，必须保持复位值。

位 3 **EERR**：不定错误 (Erratic error)

模块将该位置 1 以报告任何不定错误。

由于不定错误，OTG_HS 控制器会进入挂起状态，并且会以模块中断寄存器的早期挂起位（OTG_HS_GINTSTS 中的 ESUSP 位）生成一个中断。如果早期挂起中断是由不定错误触发，则应用程序只能执行软断开以恢复通信。

位 2:1 **ENUMSPD**：枚举速度 (Enumerated speed)

指示 OTG_HS 控制器通过 chirp 序列检测速度后被枚举成的速度。

00：高速

01：保留

10：保留

11：全速（PHY 时钟运行频率为 48 MHz）

其它值：保留

位 0 **SUSPSTS**：挂起状态 (Suspend status)

在设备模式下，只要在 USB 上检测到挂起条件，该位就会置 1。当 USB 总线上的空闲状态保持 3 ms，模块便会进入挂起状态。出现以下情况时，模块会退出挂起状态：

– USB 数据线上有活动

– 应用程序对设备控制寄存器的远程唤醒信号位（OTG_HS_DCTL 中的 RWUSIG 位）执行写操作。

OTG_HS 设备 IN 端点通用中断屏蔽寄存器 (OTG_HS_DIEPMSK)

OTG_HS device IN endpoint common interrupt mask register

偏移地址：0x810

复位值：0x0000 0000

此寄存器与设备 IN 端点中断 (OTG_HS_DIEPINTx) 寄存器配合使用，以便在每个 IN 端点上生成中断。通过对此寄存器的相应位执行写操作，可屏蔽 OTG_HS_DIEPINTx 寄存器中的 IN 端点中断。默认情况下，状态中断都被屏蔽。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																						BIM	TXFURM	Reserved	INEPNEM	INEPNMM	ITTXFEMSK	TOM	Reserved	EPDM	XFRCM
																						rw	rw		rw	rw	rw	rw		rw	rw

位 31:10 保留，必须保持复位值。

位 9 **BIM**：BNA 中断屏蔽 (BNA interrupt mask)
0：屏蔽中断
1：使能中断

位 8 **TXFURM**：发送 FIFO 下溢中断屏蔽 (FIFO underrun mask)
0：屏蔽中断
1：使能中断

位 7 保留，必须保持复位值。

位 6 **INEPNEM**：IN 端点 NAK 有效中断屏蔽 (IN endpoint NAK effective mask)
0：屏蔽中断
1：使能中断

位 5 **INEPNMM**：EP 不匹配时接收到 IN 令牌中断屏蔽 (IN token received with EP mismatch mask)
0：屏蔽中断
1：使能中断

位 4 **ITTXFEMSK**：TxFIFO 为空时接收到 IN 令牌中断屏蔽 (IN token received when TxFIFO empty mask)
0：屏蔽中断
1：使能中断

位 3 **TOM**：超时中断屏蔽（非同步端点）(Timeout condition mask (nonisochronous endpoints))
0：屏蔽中断
1：使能中断

位 2 保留，必须保持复位值。

位 1 **EPDM**：端点禁止中断屏蔽 (Endpoint disabled interrupt mask)
0：屏蔽中断
1：使能中断

位 0 **XFRCM**：传输完成中断屏蔽 (Transfer completed interrupt mask)
0：屏蔽中断
1：使能中断

OTG_HS 设备 OUT 端点通用中断屏蔽寄存器 (OTG_HS_DOEPMSK)

OTG_HS device OUT endpoint common interrupt mask register

偏移地址：0x814

复位值：0x0000 0000

此寄存器与设备 OUT 端点中断 (OTG_HS_DOEPINTx) 寄存器配合使用，以便可以在每个 OUT 端点上生成中断。通过对此寄存器的相应位执行写操作，可屏蔽 OTG_HS_DOEPINTx 寄存器中的 OUT 端点中断。默认情况下，状态中断都被屏蔽。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																						BOIM	OPEM	Reserved	B2BSTUP	Reserved	OTEPDM	STUPM	Reserved	EPDM	XFRCM
																						rw	rw		rw		rw	rw		rw	rw

位 31:10 保留，必须保持复位值。

位 9 **BOIM**：BNA 中断屏蔽 (BNA interrupt mask)
0：屏蔽中断
1：使能中断

位 8 **OPEM**：OUT 数据包错误中断屏蔽 (OUT packet error mask)
0：屏蔽中断
1：使能中断

位 7 保留，必须保持复位值。

位 6 **B2BSTUP**：接收到连续的 SETUP 数据包中断屏蔽 (Back-to-back SETUP packets received mask)
仅适用于控制 OUT 端点。
0：屏蔽中断
1：使能中断

位 5 保留，必须保持复位值。

位 4 **OTEPDM**：端点禁止时接收到 OUT 令牌中断屏蔽 (OUT token received when endpoint disabled mask)
仅适用于控制 OUT 端点。
0：屏蔽中断
1：使能中断

位 3 **STUPM**：SETUP 阶段完成中断屏蔽 (SETUP phase done mask)
仅适用于控制端点。
0：屏蔽中断
1：使能中断

位 2 保留，必须保持复位值。

位 1 **EPDM**：端点禁止中断屏蔽 (Endpoint disabled interrupt mask)
0：屏蔽中断
1：使能中断

位 0 **XFRCM**：传输完成中断屏蔽 (Transfer completed interrupt mask)
0：屏蔽中断
1：使能中断

OTG_HS 设备全体端点中断寄存器 (OTG_HS_DAINT)

OTG_HS device all endpoints interrupt register

偏移地址：0x818

复位值：0x0000 0000

当端点上发生有效事件时，设备全体端点中断寄存器将通过模块中断寄存器中的设备 OUT 端点中断位或设备 IN 端点中断位（分别为 OTG_HS_GINTSTS 中的 OEPINT 或 IEPINT 位）来中断应用程序。每个端点对应一个中断位，OUT 端点和 IN 端点均最多有 16 个中断位。双向端点将使用相应的 IN 和 OUT 中断位。当应用程序将相应设备端点 x 中断寄存器 (OTG_HS_DIEPINTx/OTG_HS_DOEPINTx) 中的位置 1 和清零时，此寄存器中的相应位也将置 1 和清零。

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

位 31:16 **OEPINT**：OUT 端点中断位 (OUT endpoint interrupt bits)
每个 OUT 端点对应一位：
OUT 端点 0 对应位 16，而 OUT 端点 15 对应位 31

位 15:0 **IEPINT**：IN 端点中断位 (IN endpoint interrupt bits)
每个 IN 端点对应一位：
IN 端点 0 对应位 0，而 IN 端点 15 对应位 15

OTG_HS 全体端点中断屏蔽寄存器 (OTG_HS_DAINTMSK)

OTG_HS all endpoints interrupt mask register

偏移地址：0x81C

复位值：0x0000 0000

设备端点中断屏蔽寄存器与设备端点中断寄存器结合使用，进而在设备端点上发生事件时中断应用程序。不过，与该中断对应的设备全体端点中断 (OTG_HS_DAINT) 寄存器位仍会置 1。

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

位 31:16 **OEPM**：OUT EP 中断屏蔽位 (OUT EP interrupt mask bits)
每个 OUT 端点对应一位：
OUT EP 0 对应位 16，而 OUT EP 3 对应位 18
0：屏蔽中断
1：使能中断

位 15:0 **IEPM**：IN EP 中断屏蔽位 (IN EP interrupt mask bits)
每个 IN 端点对应一位：
IN EP 0 对应位 0，而 IN EP 3 对应位 3
0：屏蔽中断
1：使能中断

OTG_HS 设备 V_{BUS} 放电时间寄存器 (OTG_HS_DVBUSDIS)

OTG_HS device V_{BUS} discharge time register

偏移地址：0x0828

复位值：0x0000 17D7

该寄存器指定 SRP 期间 V_{BUS} 发出脉冲之后的 V_{BUS} 放电时间。

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

位 31:16 保留，必须保持复位值。

位 15:0 **VBUSDT**：设备 V_{BUS} 放电时间 (Device VBUS discharge time)
指定 SRP 期间 V_{BUS} 发出脉冲之后的 V_{BUS} 放电时间。该时间值等于：
V_{BUS} 放电时间（PHY 时钟数）/1024
该值可基于不同的 V_{BUS} 负载根据需要进行调整。

OTG_HS 设备 V_{BUS} 脉冲时间寄存器 (OTG_HS_DVBUSPULSE)

OTG_HS device V_{BUS} pulsing time register

偏移地址：0x082C

复位值：0x0000 05B8

该寄存器指定 SRP 期间的 V_{BUS} 脉冲时间。

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

位 31:12 保留，必须保持复位值。

位 11:0 **DVBUSP**：设备 V_{BUS} 脉冲时间 (Device VBUS pulsing time)
指定 SRP 期间的 V_{BUS} 脉冲时间。该时间值等于：
V_{BUS} 脉冲时间（PHY 时钟数）/1024

OTG_HS 设备阈值控制寄存器 (OTG_HS_DTHRCTL)

OTG_HS Device threshold control register

偏移地址：0x0830

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				ARPEN	Reserved	RXTHRLEN									RXTHREN	Reserved					TXTHRLEN									ISOTHREN	NONISOTHREN
				rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:28 保留，必须保持复位值。

位 27 **ARPEN**：仲裁器驻留使能 (Arbiter parking enable)

该位用于控制内部 DMA 仲裁对 IN 端点的驻留。当阈值使能且该位置 1 后，DMA 会对收到令牌的 IN 端点驻留其仲裁。采用这种方式可以避免出现下溢情况。默认情况下使能驻留。

位 26 保留，必须保持复位值。

位 25:17 **RXTHRLEN**：接收阈值长度 (Receive threshold length)

该字段以双字为单位指定接收阈值的大小。该字段指定模块在 AHB 上启动传输之前在 USB 上接收的数据量。阈值长度最小值为八个双字。推荐 RXTHRLEN 和设定的 AHB 批量传输长度（OTG_HS_GAHBCFG 中的 HBSTLEN 位）相同。

位 16 **RXTHREN**：接收阈值使能 (Receive threshold enable)

该位置 1 时，模块会使能接收方向的阈值。

位 15:11 保留，必须保持复位值。

位 10:2 **TXTHRLEN**：发送阈值长度 (Transmit threshold length)

该字段以双字为单位指定发送阈值的大小。该字段指定模块在 USB 上启动传输之前相应端点发送 FIFO 中的数据量。阈值长度最小值为八个双字。该字段控制同步和非同步 IN 端点阈值。推荐 TXTHRLEN 和设定的 AHB 批量传输长度（OTG_HS_GAHBCFG 中的 HBSTLEN 位）相同。

位 1 **ISOTHREN**：ISO IN 端点阈值使能 (ISO IN endpoint threshold enable)

该位置 1 时，模块会使能同步 IN 端点的阈值。

位 0 **NONISOTHREN**：非同步 IN 端点阈值使能 (Nonisochronous IN endpoints enable)

该位置 1 时，模块会使能非同步 IN 端点的阈值。

OTG_HS 设备 IN 端点 FIFO 空中断屏蔽寄存器：OTG_HS_DIEPEMPMSK)

OTG_HS device IN endpoint FIFO empty interrupt mask register

偏移地址：0x834

复位值：0x0000 0000

此寄存器用于控制 IN 端点 FIFO 空中断的生成 (TXFE_OTG_HS_DIEPINTx)。

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

位 31:16 保留，必须保持复位值。

位 15:0 **INEPTXFEM**：IN EP Tx FIFO 空中断屏蔽位 (IN EP Tx FIFO empty interrupt mask bits)
这些位用作 OTG_HS_DIEPINTx 的屏蔽位。
每个位对应一个 IN 端点的 TXFE 中断：
IN 端点 0 对应位 0，而 IN 端点 15 对应位 15
0：屏蔽中断
1：使能中断

OTG_HS 设备单个端点中断寄存器 (OTG_HS_DEACHINT)

OTG_HS device each endpoint interrupt register

偏移地址：0x0838

复位值：0x0000 0000

IN 端点 1 对应一个中断位，OUT 端点 1 对应一个中断位。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														OEP1INT		Reserved														IEP1INT	Reserved

位 31:18 保留，必须保持复位值。

位 17 **OEP1INT**：OUT 端点 1 中断位 (OUT endpoint 1 interrupt bit)

位 16:2 保留，必须保持复位值。

位 1 **IEP1INT**：IN 端点 1 中断位 (IN endpoint 1interrupt bit)

位 0 保留，必须保持复位值。

OTG_HS 设备单个端点中断屏蔽寄存器 (OTG_HS_DEACHINTMSK)

OTG_HS device each endpoint interrupt register mask

偏移地址：0x083C

复位值：0x0000 0000

IN 端点 1 对应一个中断位，OUT 端点 1 对应一个中断位。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														OEP1INTM	Reserved															IEP1INTM	Reserved

位 31:18 保留，必须保持复位值。

位 17 **OEP1INTM**：OUT 端点 1 中断屏蔽位 (OUT Endpoint 1 interrupt mask bit)

位 16:2 保留，必须保持复位值。

位 1 **IEP1INTM**：IN 端点 1 中断屏蔽位 (IN Endpoint 1 interrupt mask bit)

位 0 保留，必须保持复位值。

OTG_HS 设备 IN 端点 1 中断屏蔽寄存器 (OTG_HS_DIEPEACHMSK1)

OTG_HS device each in endpoint-1 interrupt register

偏移地址：0x844

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																		NAKM	Reserved			BIM	TXFURM	Reserved	INEPNEM	INEPNMM	ITTXFEMSK	TOM	Reserved	EPDM	XFRCM
																		rw				rw	rw		rw	rw	rw	rw		rw	rw

位 31:14 保留，必须保持复位值。

位 13 **NAKM**：NAK 中断屏蔽 (NAK interrupt mask)

0：屏蔽中断
1：使能中断

位 12:10 保留，必须保持复位值。

位 9 **BIM**：BNA 中断屏蔽 (BNA interrupt mask)

0：屏蔽中断
1：使能中断

位 8 **TXFURM**：发送 FIFO 下溢中断屏蔽 (FIFO underrun mask)

0：屏蔽中断
1：使能中断

位 7 保留，必须保持复位值。

位 6 **INEPNEM**：IN 端点 NAK 有效中断屏蔽 (IN endpoint NAK effective mask)
0：屏蔽中断
1：使能中断

位 5 **INEPNMM**：EP 不匹配时接收到 IN 令牌中断屏蔽 (IN token received with EP mismatch mask)
0:屏蔽中断
1：使能中断

位 4 **ITTXFEMSK**：TxFIFO 为空时接收到 IN 令牌中断屏蔽 (IN token received when TxFIFO empty mask)
0：屏蔽中断
1：使能中断

位 3 **TOM**：超时中断屏蔽（非同步端点）(Timeout condition mask (nonisochronous endpoints))
0：屏蔽中断
1：使能中断

位 2 保留，必须保持复位值。

位 1 **EPDM**：端点禁止中断屏蔽 (Endpoint disabled interrupt mask)
0：屏蔽中断
1：使能中断

位 0 **XFRCM**：传输完成中断屏蔽 (Transfer completed interrupt mask)
0：屏蔽中断
1：使能中断

OTG_HS 设备 OUT 端点 1 中断屏蔽寄存器 (OTG_HS_DOEPEACHMSK1)

OTG_HS device each OUT endpoint-1 interrupt register

偏移地址：0x884

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																	NYETM	NAKM	BERRM	Reserved		BIM	TXFURM	Reserved	INEPNEM	INEPNMM	ITTXFEMSK	TOM	Reserved	EPDM	XFRCM
																	rw	rw	rw			rw	rw		rw	rw	rw	rw		rw	rw

位 31:15 保留，必须保持复位值。

位 14 **NYETM**：NYET 中断屏蔽 (NYET interrupt mask)
0：屏蔽中断
1：使能中断

位 13 **NAKM**：NAK 中断屏蔽 (NAK interrupt mask)
0：屏蔽中断
1：使能中断

位 12 **BERRM**：Babble 错误中断屏蔽 (Babble error interrupt mask)
0：屏蔽中断
1：使能中断

位 11:10 保留，必须保持复位值。

位 9 **BIM**：BNA 中断屏蔽 (BNA interrupt mask)
0：屏蔽中断
1：使能中断

位 8 **OPEM**：OUT 数据包错误中断屏蔽 (OUT packet error mask)
0：屏蔽中断
1：使能中断

位 7:3 保留，必须保持复位值。

位 2 **AHBERRM**：AHB 错误中断屏蔽 (AHB error mask)
0：屏蔽中断
1：使能中断

位 1 **EPDM**：端点禁止中断屏蔽 (Endpoint disabled interrupt mask)
0：屏蔽中断
1：使能中断

位 0 **XFRCM**：传输完成中断屏蔽 (Transfer completed interrupt mask)
0：屏蔽中断
1：使能中断

OTG 设备端点 x 控制寄存器 (OTG_HS_DIEPCTLx)（x = 0..7，其中 x = 端点编号）

OTG device endpoint-x control register

偏移地址：0x900 + (端点编号 × 0x20)

复位值：0x0000 0000

应用程序使用此寄存器控制各个逻辑端点（端点 0 除外）的行为。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EPENA	EPDIS	SODDFRM	SD0PID/SEVNFRM	SNAK	CNAK	TXFNUM				Stall	Reserved	EPTYP		NAKSTS	EONUM/DPID	USBAEP	Reserved				MPSIZ											
rs	rs	w	w	w	w	rw	rw	rw	rw	rw/ rs		rw	rw	r	r	rw					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31 **EPENA**：端点使能 (Endpoint enable)
应用程序将此位置 1 以在端点上启动数据发送。
在此端点上触发以下任一中断之前，模块会将此位清零：
– SETUP 阶段完成 (SETUP phase done)
– 端点禁止
– 传输完成

位 30 **EPDIS**：端点禁止 (Endpoint disable)
即使在该端点上的传送完成之前，应用程序也可将此位置 1，以停止端点上的数据发送/接收。应用程序必须等到发生端点禁止中断后，才能将端点视为禁止端点。在端点禁止中断位置 1 前，模块会将此位清零。只有在该端点的端点使能位置 1 后，应用程序才可将该位置 1。

位 29 **SODDFRM**：设置奇数帧 (Set odd frame)
仅适用于同步 IN 和 OUT 端点。
对此字段进行写操作会将偶数/奇数帧 (EONUM) 字段设置为奇数帧。

位 28 **SD0PID**：设置 DATA0 PID (Set DATA0 PID)

仅适用于中断/批量 IN 端点。

对此字段进行写操作会将此寄存器中的端点数据 PID (DPID) 字段设置为 DATA0。

SEVNFRM：设置偶数帧 (Set even frame)

仅适用于同步 IN 端点。

对此字段进行写操作会将偶数/奇数帧 (EONUM) 字段设置为偶数帧。

位 27 **SNAK**：将 NAK 置 1 (Set NAK)

对此位进行写操作会将端点的 NAK 位置 1。

通过此位，应用程序可以控制端点上 NAK 握手信号的发送。发生传输完成中断时或端点上接收到 SETUP 后，模块也可以将 OUT 端点的这个位置 1。

位 26 **CNAK**：将 NAK 清零 (Clear NAK)

对此位进行写操作会将端点的 NAK 位清零。

位 25:22 **TXFNUM**：TxFIFO 编号 (TxFIFO number)

这些位用于指定与此端点相关联的 FIFO 编号。必须为每个有效的 IN 端点设置单独的 FIFO 编号。

此字段仅针对 IN 端点有效。

位 21 **STALL**：STALL 握手 (STALL handshake)

仅适用于非控制、非同步 IN 端点（访问类型为 rw）。

应用程序将此位置 1 使得设备对来自 USB 主机的所有令牌都回复 STALL。如果 NAK 位、全局 IN NAK 或全局 OUT NAK 与此位同时置 1，则 STALL 位优先。只有应用程序能够将此位清零，而模块则不能。

仅适用于控制端点（访问类型为 rs）

此端点接收到 SETUP 令牌时，应用程序只能将此位置 1，而模块会将其清零。如果 NAK 位、全局 IN NAK 或全局 OUT NAK 与此位同时置 1，则 STALL 位优先。无论此位如何设置，模块总是通过 ACK 握手响应 SETUP 数据包。

位 20 保留，必须保持复位值。

位 19:18 **EPTYP**：端点类型 (Endpoint type)

以下是这个逻辑端点支持的传输类型。

00：控制

01：同步

10：批量

11：中断

位 17 **NAKSTS**：NAK 状态 (NAK status)

它指示以下结果：

0：模块根据 FIFO 状态回复非 NAK 握手。

1：模块在此端点上回复 NAK 握手。

当应用程序或模块将此位置 1 时：

对于非同步 IN 端点：即使 TxFIFO 中存在可用数据，模块也会停止通过 IN 端点发送任何数据。

对于同步 IN 端点：即使 TxFIFO 中存在可用数据，模块也会发送长度为零的数据包。

无论此位如何设置，模块总是通过 ACK 握手响应 SETUP 数据包。

位 16 **EONUM**：偶数/奇数帧 (Even/odd frame)

仅适用于同步 IN 端点。

指示模块为此端点发送/接收同步的数据所在的帧的编号。应用程序必须通过此寄存器中的 SEVNFRM 和 SODDFRM 字段对偶数/奇数帧编号进行编程，以便此端点发送/接收同步数据。

0：偶数帧

1：奇数帧

DPID：端点数据 PID (Endpoint data PID)

仅适用于中断/批量 IN 端点。

包含此端点上将要接收或发送的数据包的 PID。端点激活后，应用程序必须对要在此端点上接收或发送的首个数据包的 PID 进行编程。应用程序使用 SD0PID 寄存器字段对 DATA0 或 DATA1 PID 进行编程。

0：DATA0

1：DATA1

位 15 **USBAEP**：USB 活动端点 (USB active endpoint)

指示此端点在当前配置和接口中是否激活。检测到 USB 复位后，模块会为所有端点（端点 0 除外）将此位清零。接收到 SetConfiguration 和 SetInterface 命令后，应用程序必须相应地对端点寄存器进行编程并将此位置 1。

位 14:11 保留，必须保持复位值。

位 10:0 **MPSIZ**：最大数据包大小 (Maximum packet size)

应用程序必须将此字段编程为当前逻辑端点的最大数据包大小。此值以字节为单位。

OTG_HS 设备控制 OUT 端点 0 控制寄存器 (OTG_HS_DOEPCTL0)

OTG_HS device control OUT endpoint 0 control register

偏移地址：0xB00

复位值：0x0000 8000

本节介绍设备控制 OUT 端点 0 控制寄存器。非零控制端点使用编号 1-15 的寄存器。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EPENA	EPDIS	Reserved		SNAK	CNAK	Reserved				Stall	SNPM	EPTYP		NAKSTS	Reserved	USBAEP	Reserved													MPSIZ	
w	r			w	w					rs	rw	r	r	r		r														r	r

位 31 **EPENA**：端点使能 (Endpoint enable)

应用程序将此位置 1 以在端点 0 上启动数据发送。

在此端点上触发以下任一中断之前，模块会将此位清零：

– SETUP 阶段完成

– 端点禁止

– 传输完成

位 30 **EPDIS**：端点禁止 (Endpoint disable)

应用程序无法禁止控制 OUT 端点 0。

位 29:28 保留，必须保持复位值。

位 27 **SNAK**：将 NAK 置 1 (Set NAK)

对此位进行写操作会将端点的 NAK 位置 1。

通过此位，应用程序可以控制端点上 NAK 握手信号的发送。发生传输完成中断时或端点上接收到 SETUP 后，模块也可以将此位置 1。

位 26 **CNAK**：将 NAK 清零 (Clear NAK)

对此位进行写操作会将端点的 NAK 位清零。

位 25:22 保留，必须保持复位值。

位 21 **STALL**：STALL 握手 (STALL handshake)

此端点接收到 SETUP 令牌时，应用程序只能将此位置 1，而模块会将其清零。如果 NAK 位、全局 OUT NAK 与此位同时置 1，则 STALL 位优先。无论此位如何设置，模块总是通过 ACK 握手响应 SETUP 数据包。

位 20 **SNPM**：监听模式 (Snoop mode)

此位用于将端点配置为监听模式。在监听模式下，模块不会在将 OUT 数据包传输到应用存储区前检查其是否正确。

位 19:18 **EPTYP**：端点类型 (Endpoint type)

硬件固定为二进制 00，表示端点为控制传输类型。

位 17 **NAKSTS**：NAK 状态 (NAK status)

指示以下结果：

0：模块根据 FIFO 状态回复非 NAK 握手。

1：模块在此端点上回复 NAK 握手。

当应用程序或模块将此位置 1 时，即使 RxFIFO 中存在空间可继续容纳收到的数据包，模块也会停止接收数据。无论此位如何设置，模块总是通过 ACK 握手响应 SETUP 数据包。

位 16 保留，必须保持复位值。

位 15 **USBAEP**：USB 活动端点 (USB active endpoint)

此位总是置 1，指示在所有配置和接口中控制端点 0 始终处于激活状态。

位 14:2 保留，必须保持复位值。

位 1:0 **MPSIZ**：最大数据包大小 (Maximum packet size)

控制 OUT 端点 0 的最大数据包大小与在控制 IN 端点 0 中进行编程的值相同。

00：64 字节

01：32 字节

10：16 字节

11：8 字节

OTG_HS 设备端点 x 控制寄存器 (OTG_HS_DOEPCTLx)（x = 1..3，其中 x = 端点编号）

OTG_HS device endpoint-x control register

OUT 端点的偏移地址：0xB00 + (端点编号 × 0x20)

复位值：0x0000 0000

应用程序使用此寄存器控制各个逻辑端点（端点 0 除外）的行为。

31	30	29	28	27	26	25 24 23 22	21	20	19 18	17	16	15	14 13 12 11	10 9 8 7 6 5 4 3 2 1 0
EPENA	EPDIS	SODDFRM	SD0PID/SEVNFRM	SNAK	CNAK	Reserved	Stall	SNPM	EPTYP	NAKSTS	EONUM/DPID	USBAEP	Reserved	MPSIZ
rs	rs	w	w	w	w		rw/ rs	rw	rw rw	r	r	rw		rw rw rw rw rw rw rw rw rw rw rw

位 31 **EPENA**：端点使能 (Endpoint enable)

适用于 IN 和 OUT 端点。

应用程序将此位置 1 以在端点上启动数据发送。

在此端点上触发以下任一中断之前，模块会将此位清零：

– SETUP 阶段完成

– 端点禁止

– 传输完成

位 30 **EPDIS**：端点禁止 (Endpoint disable)

即使在该端点上的传送完成之前，应用程序也可将此位置 1，以停止端点上的数据发送/接收。应用程序必须等到发生端点禁止中断后，才能将端点视为禁止端点。在端点禁止中断位置 1 前，模块会将此位清零。只有在该端点的端点使能位置 1 后，应用程序才可将该位置 1。

位 29 **SODDFRM**：设置奇数帧 (Set odd frame)

仅适用于同步 OUT 端点。

对此字段进行写操作会将偶数/奇数帧 (EONUM) 字段设置为奇数帧。

位 28 **SD0PID**：设置 DATA0 PID (Set DATA0 PID)

仅适用于中断/批量 OUT 端点。

对此字段进行写操作会将此寄存器中的端点数据 PID (DPID) 字段设置为 DATA0。

SEVNFRM：设置偶数帧 (Set even frame)

仅适用于同步 OUT 端点。

对此字段进行写操作会将偶数/奇数帧 (EONUM) 字段设置为偶数帧。

位 27 **SNAK**：将 NAK 置 1 (Set NAK)

对此位进行写操作会将端点的 NAK 位置 1。

通过此位，应用程序可以控制端点上 NAK 握手信号的发送。发生传输完成中断时或端点上接收到 SETUP 后，模块也可以将 OUT 端点的这个位置 1。

位 26 **CNAK**：将 NAK 清零 (Clear NAK)

对此位进行写操作会将端点的 NAK 位清零。

位 25:22 保留，必须保持复位值。

位 21 **STALL**：STALL 握手 (STALL handshake)

仅适用于非控制、非同步 OUT 端点（访问类型为 rw）。

应用程序将此位置 1 使得设备对来自 USB 主机的所有令牌都回复 STALL。如果 NAK 位、全局 IN NAK 或全局 OUT NAK 与此位同时置 1，则 STALL 位优先。只有应用程序能够将此位清零，而模块则不能。

仅适用于控制端点（访问类型为 rs）。

此端点接收到 SETUP 令牌时，应用程序只能将此位置 1，而模块会将其清零。如果 NAK 位、全局 IN NAK 或全局 OUT NAK 与此位同时置 1，则 STALL 位优先。无论此位如何设置，模块总是通过 ACK 握手响应 SETUP 数据包。

位 20 **SNPM**：监听模式 (Snoop mode)

此位用于将端点配置为监听模式。在监听模式下，模块不会在将 OUT 数据包传输到应用存储区前检查其是否正确。

位 19:18 **EPTYP**：端点类型 (Endpoint type)

以下是这个逻辑端点支持的传输类型。

00：控制

01：同步

10：批量

11：中断

位 17 **NAKSTS**：NAK 状态 (NAK status)

指示以下结果：

0：模块根据 FIFO 状态回复非 NAK 握手。

1：模块在此端点上回复 NAK 握手。

当应用程序或模块将此位置 1 时：

即使 RxFIFO 存在空间可容纳传入数据包，模块也会停止在 OUT 端点上接收任何数据。

无论此位如何设置，模块总是通过 ACK 握手响应 SETUP 数据包。

位 16 **EONUM**：偶数/奇数帧 (Even/odd frame)

仅适用于同步 IN 和 OUT 端点。

指示模块为此端点发送/接收同步的数据所在的帧的编号。应用程序必须通过此寄存器中的 SEVNFRM 和 SODDFRM 字段对偶数/奇数帧编号进行编程，以便此端点发送/接收同步数据。

0：偶数帧

1：奇数帧

DPID：端点数据 PID (Endpoint data PID)

仅适用于中断/批量 OUT 端点。

包含此端点上将要接收或发送的数据包的 PID。端点激活后，应用程序必须对要在此端点上接收或发送的首个数据包的 PID 进行编程。应用程序使用 SD0PID 寄存器字段对 DATA0 或 DATA1 PID 进行编程。

0：DATA0

1：DATA1

位 15 **USBAEP**：USB 活动端点 (USB active endpoint)

指示此端点在当前配置和接口中是否激活。检测到 USB 复位后，模块会为所有端点（端点 0 除外）将此位清零。接收到 SetConfiguration 和 SetInterface 命令后，应用程序必须相应地对端点寄存器进行编程并将此位置 1。

位 14:11 保留，必须保持复位值。

位 10:0 **MPSIZ**：最大数据包大小 (Maximum packet size)

应用程序必须将此字段编程为当前逻辑端点的最大数据包大小。此值以字节为单位。

OTG_HS 设备端点 x 中断寄存器 (OTG_HS_DIEPINTx)（x = 0..7，其中 x = 端点编号）

OTG_HS device endpoint-x interrupt register

偏移地址：0x908 + (端点编号 × 0x20)

复位值：0x0000 0080

此寄存器指示端点在出现 USB 和 AHB 相关事件时的状态。具体如*图 383* 中所示。当模块中断寄存器中的 IN 端点中断位（OTG_HS_GINTSTS 中的 IEPINT 位）置 1 时，应用程序必须读取此寄存器。应用程序必须先读取设备全体端点中断 (OTG_HS_DAINT) 寄存器，以获取设备端点 x 中断寄存器的准确端点编号，然后才能读取此寄存器。应用程序必须将该寄存器中的相应位清零，才能将 OTG_HS_DAINT 和 OTG_HS_GINTSTS 寄存器中的对应位清零。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																		NAK	BERR	PKTDRPSTS	Reserved	BNA	TXFIFOUDRN	TXFE	INEPNE	Reserved	ITTXFE	TOC	Reserved	EPDISD	XFRC
																								r	rc_w1/rw		rc_w1	rc_w1		rc_w1	rc_w1

位 31:14 保留，必须保持复位值。

位 13 **NAK**：NAK 中断 (NAK interrupt)

当设备发出或收到 NAK 时，模块将生成该中断。如果是同步 IN 端点，由于 Tx FIFO 中无数据可发而发送长度为零的数据包时也会生成该中断。

位 12 **BERR**：Babble 错误中断 (Babble error interrupt)

位 11 **PKTDRPSTS**：数据包丢弃状态 (Packet dropped status)

该位用于向应用程序指示有 ISOC OUT 数据包被丢弃。该位没有相应的中断屏蔽位，也不会生成中断。

位 10 保留，必须保持复位值。

位 9 **BNA**：缓冲区不可用中断 (Buffer not available interrupt)

当所访问的描述符没有准备好供模块处理时（例如主机繁忙或 DMA 已完成），模块将生成该中断。

位 8 **TXFIFOUDRN**：发送 Fifo 下溢 (TxfifoUndrn) ((Transmit Fifo Underrun) (TxfifoUndrn)) 当模块检测到该端点发送 FIFO 下溢时，将生成该中断。

相关性：该中断仅在使能了阈值时有效

位 7 **TXFE**：发送 FIFO 为空 (Transmit FIFO empty)

当此端点的 TxFIFO 为半空或全空时，此中断被置位。该 TxFIFO 为半空状态还是全空状态由模块 AHB 配置寄存器中的 TxFIFO 空级别位（OTG_HS_GAHBCFG 中的 TXFELVL 位）决定。

位 6 **INEPNE**：IN 端点 NAK 有效 (IN endpoint NAK effective)

当应用程序通过向 OTG_HS_DIEPCTLx 中的 CNAK 位写入数据来将 IN 端点 NAK 清零时，此位可被清零。

该中断指示模块已对（由应用程序或模块）置 1 的 NAK 采样，结果已生效。该中断指示由应用程序置 1 的 IN 端点 NAK 位已在模块中起作用。

此中断不保证在 USB 上发送了 NAK 握手信号。STALL 位的优先级高于 NAK 位。

位 5 保留，必须保持复位值。

位 4 **ITTXFE**：TxFIFO 为空时接收到 IN 令牌 (IN token received when TxFIFO is empty)
仅适用于非周期性 IN 端点。
当和该端点相对应的 TxFIFO（周期性/非周期性）为空时，接收到 IN 令牌，从而产生中断。

位 3 **TOC**：超时条件 (Timeout condition)
仅适用于控制 IN 端点。
指示该端点对最近收到的 IN 令牌响应超时。

位 2 保留，必须保持复位值。

位 1 **EPDISD**：端点禁止中断 (Endpoint disabled interrupt)
此位指示该端点已经由应用程序禁止掉。

位 0 **XFRC**：传输完成中断 (Transfer completed interrupt)
此字段指示在此端点上设置的传输已经在 USB 和 AHB 上传输完成。

OTG_HS 设备端点 x 中断寄存器 (OTG_HS_DOEPINTx)（x = 0..7，其中 x = 端点编号）

OTG_HS device endpoint-x interrupt register

偏移地址：0xB08 + (端点编号 × 0x20)

复位值：0x0000 0080

此寄存器指示端点在出现 USB 和 AHB 相关事件时的状态。具体如*图 383* 中所示。当模块中断寄存器中的 OUT 端点中断位（OTG_HS_GINTSTS 中的 OEPINT 位）置 1 时，应用程序必须读取此寄存器。应用程序必须先读取设备全体端点中断 (OTG_HS_DAINT) 寄存器，以获取设备端点 x 中断寄存器的准确端点编号，然后才能读取此寄存器。应用程序必须将该寄存器中的相应位清零，才能将 OTG_HS_DAINT 和 OTG_HS_GINTSTS 寄存器中的对应位清零。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																	NYET	Reserved							B2BSTUP	Reserved	OTEPDIS	STUP	Reserved	EPDISD	XFRC
																									rc_w1/rw		rc_w1	rc_w1		rc_w1	rc_w1

位 31:15 保留，必须保持复位值。

位 14 **NYET**：NYET 中断 (NYET interrupt)
当非同步 OUT 端点回复 NYET 握手信号时，模块将产生此中断。

位 13:7 保留，必须保持复位值。

位 6 **B2BSTUP**：接收到连续的 SETUP 数据包 (Back-to-back SETUP packets received)
仅适用于控制 OUT 端点。
此位指示该端点已接收到三个以上的连续 SETUP 数据包。

位 5 保留，必须保持复位值。

位 4 **OTEPDIS**：端点禁止时接收到 OUT 令牌 (OUT token received when endpoint disabled)
仅适用于控制 OUT 端点。
指示在尚未使能端点时接收到 OUT 令牌，从而产生中断。

位 3 **STUP**：SETUP 阶段完成 (SETUP phase done)
仅适用于控制 OUT 端点。
指示控制端点的 SETUP 阶段已完成，当前控制传输中不再接收到连续的 SETUP 数据包。在此中断上，应用程序可以对接收到的 SETUP 数据包进行解码。

位 2 保留，必须保持复位值。

位 1 **EPDISD**：端点禁止中断 (Endpoint disabled interrupt)
此位指示该端点已经由应用程序禁止掉。

位 0 **XFRC**：传输完成中断 (Transfer completed interrupt)
此字段指示在此端点上设置的传输已经在 USB 和 AHB 上传输完成。

OTG_HS 设备 IN 端点 0 传输大小寄存器 (OTG_HS_DIEPTSIZ0)

OTG_HS device IN endpoint 0 transfer size register

偏移地址：0x910

复位值：0x0000 0000

在使能端点 0 之前，应用程序必须修改此寄存器。通过设备控制端点 0 控制寄存器中的端点使能位（OTG_HS_DIEPCTL0 中的 EPENA）使能端点 0 后，模块对此寄存器进行修改。仅当模块将端点使能位清零后，应用程序才能读取此寄存器。

非零端点使用端点 1-15 的寄存器。

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

位 31:21 保留，必须保持复位值。

位 20:19 **PKTCNT**：数据包计数 (Packet count)
指示端点 0 的一次数据传输包含的数据包个数。
每次从 TxFIFO 读取数据包（最大大小或短数据包）时，此字段将递减。

位 18:7 保留，必须保持复位值。

位 6:0 **XFRSIZ**：传输大小 (Transfer size)
指示端点 0 的一次数据传输包含的数据量，以字节为单位。仅当应用程序传输完这些数据后，模块才会中断该应用程序。传输大小可以设置为端点的最大数据包大小，以在每个数据包结束时中断。
每次向 TxFIFO 写入来自外部存储器的数据包时，模块会使此字段递减。

OTG_HS 设备 OUT 端点 0 传输大小寄存器 (OTG_HS_DOEPTSIZ0)

OTG_HS device OUT endpoint 0 transfer size register

偏移地址：0xB10

复位值：0x0000 0000

在使能端点 0 之前，应用程序必须修改此寄存器。通过设备控制端点 0 控制寄存器中的端点使能位（OTG_HS_DOEPCTL0 中的 EPENA 位）使能端点 0 后，模块对此寄存器进行修改。仅当模块将端点使能位清零后，应用程序才能读取此寄存器。

非零端点使用端点 1-15 的寄存器。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	STUPCNT		Reserved									PKTCNT	Reserved												XFRSIZ						
	rw	rw										rw													rw	rw	rw	rw	rw	rw	rw

位 31 保留，必须保持复位值。

位 30:29 **STUPCNT**：SETUP 数据包计数 (SETUP packet count)

此字段指定端点能连续接收的 SETUP 数据包数量。

01：1 个数据包

10：2 个数据包

11：3 个数据包

位 28:20 保留，必须保持复位值。

位 19 **PKTCNT**：数据包计数 (Packet count)

每向 RxFIFO 写入一个数据包，此字段递减。

位 18:7 保留，必须保持复位值。

位 6:0 **XFRSIZ**：传输大小 (Transfer size)

指示端点 0 的一次数据传输包含的数据量，以字节为单位。仅当应用程序传输完这些数据后，模块才会中断该应用程序。传输大小可以设置为端点的最大数据包大小，以在每个数据包结束时中断。

每次从 RxFIFO 读取数据包并将其写入外部存储器时，模块会使此字段递减。

OTG_HS 设备端点 x 传输大小寄存器 (OTG_HS_DIEPTSIZx)（x = 1..3，其中 x = 端点编号）

OTG_HS device endpoint-x transfer size register

偏移地址：0x910 + (端点编号 × 0x20)

复位值：0x0000 0000

在使能该端点之前，应用程序必须修改此寄存器。通过设备端点 x 控制寄存器中的端点使能位（OTG_HS_DIEPCTLx 中的 EPENA 位）使能该端点后，模块对此寄存器进行修改。仅当模块将端点使能位清零后，应用程序才能读取此寄存器。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	MCNT		PKTCNT										XFRSIZ																		
	rw/r/rw	rw/r/rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31 保留，必须保持复位值。

位 30:29 **MCNT**：多重计数 (Multi count)

对于周期性 IN 端点，此字段指示在 USB 上每帧必须发送的数据包数。模块使用此字段计算同步 IN 端点的数据 PID。

01：1 个数据包

10：2 个数据包

11：3 个数据包

位 28:19 **PKTCNT**：数据包计数 (Packet count)

指示该端点上的一次数据传输包含的数据包个数。

每次从 TxFIFO 读取数据包（最大大小或短数据包）时，此字段将递减。

位 18:0 **XFRSIZ**：传输大小 (Transfer size)

此字段包含当前端点的一次数据传输包含的数据量，以字节为单位。仅当应用程序传输完这些数据后，模块才会中断该应用程序。传输大小可以设置为端点的最大数据包大小，以在每个数据包结束时中断。

每次向 TxFIFO 写入来自外部存储器的数据包时，模块会使此字段递减。

OTG_HS 设备 IN 端点发送 FIFO 状态寄存器 (OTG_HS_DTXFSTSx)（x = 0..5，其中 x = 端点编号）

OTG_HS device IN endpoint transmit FIFO status register

IN 端点的偏移地址：0x918 + (端点编号 × 0x20) 此只读寄存器包含设备 IN 端点 TxFIFO 的空闲空间信息。

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

31:16 保留，必须保持复位值。

15:0 **INEPTFSAV**：IN 端点 TxFIFO 可用空间 (IN endpoint TxFIFO space available)

指示端点 TxFIFO 中的可用空闲空间大小。

以 32 位字为单位：

0x0：端点 TxFIFO 已满

0x1：1 个字可用

0x2：2 个字可用

0xn：n 个字可用 (0 < n < 512)

其它值：保留

OTG_HS 设备端点 x 传输大小寄存器 (OTG_HS_DOEPTSIZx)（x = 1..5，其中 x = 端点编号）

OTG_HS device endpoint-x transfer size register

偏移地址：0xB10 + (端点编号 × 0x20)

复位值：0x0000 0000

在使能该端点之前，应用程序必须修改此寄存器。通过设备端点 x 控制寄存器中的端点使能位（OTG_HS_DOEPCTLx 中的 EPENA 位）使能该端点后，模块对此寄存器进行修改。仅当模块将端点使能位清零后，应用程序才能读取此寄存器。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	RXDPID/STUPCNT		PKTCNT										XFRSIZ																		
	rw/r/rw	rw/r/rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31 保留，必须保持复位值。

位 30:29 **RXDPID**：接收到的数据 PID (Received data PID)

仅适用于同步 OUT 端点。

这是此端点收到的上一个数据包的 PID。

00：DATA0

01：DATA2

10：DATA1

11：MDATA

STUPCNT：SETUP 数据包计数 (SETUP packet count)

仅适用于控制 OUT 端点。

此字段指定端点能连续接收的 SETUP 数据包数量。

01：1 个数据包

10：2 个数据包

11：3 个数据包

位 28:19 **PKTCNT**：数据包计数 (Packet count)

指示该端点上的一次数据传输包含的数据包个数。

每次向 RxFIFO 写入数据包（最大大小或短数据包）后，此字段将递减。

位 18:0 **XFRSIZ**：传输大小 (Transfer size)

此字段包含当前端点的一次数据传输包含的数据量，以字节为单位。仅当应用程序传输完这些数据后，模块才会中断该应用程序。传输大小可以设置为端点的最大数据包大小，以在每个数据包结束时中断。

每次从 RxFIFO 读取数据包并将其写入外部存储器时，模块会使此字段递减。

OTG_HS 设备端点 *x* DMA 地址寄存器 (OTG_HS_DIEPDMAx/OTG_HS_DOEPDMAx)（x = 1..5，其中 x = 端点编号）

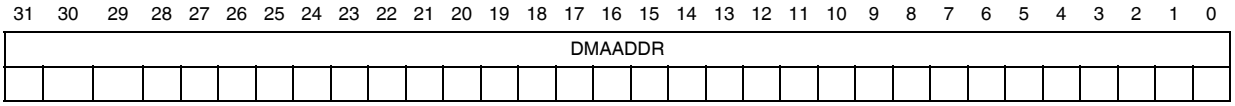
OTG_HS device endpoint-x DMA address register

IN 端点的偏移地址：0x914 + (端点编号 × 0x20)

复位值：0xXXXX XXXX

OUT 端点的偏移地址：0xB14 + (端点编号 × 0x20)

复位值：0xXXXX XXXX



位 31:0 **DMAADDR**：DMA 地址 (DMA address)

该位包含使用 DMA 进行端点上数据存储或发送时的外部存储区起始地址。

注意：对于控制端点，该字段所指向的存储区也用于存储控制 OUT 数据包以及 SETUP 事务数据包。连续接收到三个以上的 SETUP 数据包时，存储器中的 SETUP 数据包将被覆盖。每次进行 AHB 传输，该寄存器都会递增。应用程序必须设定一个双字对齐地址。

31.12.5 OTG_HS 电源和时钟门控控制寄存器 (OTG_HS_PCGCCTL)

OTG_HS power and clock gating control register

偏移地址：0xE00

复位值：0x0000 0000

此寄存器在主机模式和设备模式下均可用。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																											PHYSUSP	Reserved		GATEHCLK	STPPCLK
																											rw			rw	rw

位 31:5 保留，必须保持复位值。

位 4 **PHYSUSP**：PHY 挂起 (PHY suspended)

指示 PHY 已挂起。应用程序将 STPPCLK 位（位 0）置 1 后，一旦 PHY 挂起，此位就会更新。

位 3:2 保留，必须保持复位值。

位 1 **GATEHCLK**：门控 HCLK (Gate HCLK)

当 USB 通信挂起或会话无效时，应用程序会将此位置 1，以停止对除 AHB 总线从接口、主接口和唤醒逻辑之外的模块提供时钟。当 USB 恢复通信或新会话启动时，应用程序将此位清零。

位 0 **STPPCLK**：停止 PHY 时钟 (Stop PHY clock)

当 USB 通信挂起、会话无效或设备断开连接时，应用程序将此位置 1 以停止 PHY 时钟。当 USB 恢复通信或新会话启动时，应用程序将此位清零。

31.12.6 OTG_HS 寄存器映射

下表提供了 USB OTG 寄存器映射和复位值。

表 184. OTG_HS 寄存器映射和复位值

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	OTG_HS_GOTGCTL	Reserved												BSVLD	ASVLD	DBCT	CIDSTS	Reserved				DHNPEN	HSHNPEN	HNPRQ	HNGSCS	Reserved						SRQ	SRQSCS
	Reset value													0	0	0	1					0	0	0	0							0	0
0x004	OTG_HS_GOTGINT	Reserved												DBCDNE	ADTOCHG	HNGDET	Reserved							HNSSCHG	SRSSCHG	Reserved					SEDET	Res.	
	Reset value													0	0	0								0	0						0		
0x008	OTG_HS_GAHBCFG	Reserved																							PTXFELVL	TXFELVL	Reserved						GINT
	Reset value																								0	0							0
0x00C	OTG_HS_GUSBCFG	CTXPKT	FDMOD	FHMOD	Reserved			ULPIIPD	PTCI	PCCI	TSDPS	ULPIEVBUSI	ULPIEVBUSD	ULPICSM	ULPIAR	ULPIFSLS	Reserved	PHYLPCS	Reserved	TRDT				HNPCAP	SRPCAP	Reserved	PHSEL	Reserved				TOCAL		
	Reset value	0	0	0				0	0	0	0	0	0	0	0	0		0		0	0	1	0	1	0		1				0	0	0

表 184. OTG_HS 寄存器映射和复位值 （续）

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x010	OTG_HS_GRSTCTL	AHBIDL	DMAREQ	Reserved																			TXFNUM					TXFFLSH	RXFFLSH	Reserved	FCRST	HSRST	CSRST
	Reset value	1	0																				0	0	0	0	0	0	0	0	0	0	0
0x014	OTG_HS_GINTSTS	WKUINT	SRQINT	DISCINT	CIDSCHG	Reserved	PTXFE	HCINT	HPRTINT	Reserved	DATAFSUSP	IPXFR/INCOMPISOOUT	IISOIXFR	OEPINT	IEPINT	Reserved		EOPF	ISOODRP	ENUMDNE	USBRST	USBSUSP	ESUSP	Reserved		BOUTNAKEFF	GINAKEFF	NPTXFE	RXFLVL	SOF	OTGINT	MMIS	CMOD
	Reset value	0	0	0	0		1	0	0		0	0	0	0	0			0	0	0	0	0	0			0	0	1	0	0	0	0	0
0x018	OTG_HS_GINTMSK	WUIM	SRQIM	DISCINT	CIDSCHGM	Reserved	PTXFEM	HCIM	PRTIM	Reserved	FSUSPM	IPXFRM/IISOOXFRM	IISOIXFRM	OEPINT	IEPINT	EPMISM	Reserved	EOPFM	ISOODRPM	ENUMDNEM	USBRST	USBSUSPM	ESUSPM	Reserved		GONAKEFFM	GINAKEFFM	NPTXFEM	RXFLVLM	SOFM	OTGINT	MMISM	Reserved
	Reset value	0	0	0	0		0	0	0		0	0	0	0	0	0		0	0	0	0	0	0			0	0	0	0	0	0	0	
0x01C	OTG_HS_GRXSTSR (Host mode)	Reserved											PKTSTS				DPID	BCNT													CHNUM		
	Reset value												0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	OTG_HS_GRXSTSR (peripheral mode)	Reserved							FRMNUM				PKTSTS				DPID	BCNT													EPNUM		
	Reset value								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x020	OTG_HS_GRXSTSP (Host mode)	Reserved											PKTSTS				DPID	BCNT													CHNUM		
	Reset value												0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	OTG_HS_GRXSTSP (peripheral mode)	Reserved							FRMNUM				PKTSTS				DPID	BCNT													EPNUM		
	Reset value								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x024	OTG_HS_GRXFSIZ	Reserved																RXFD															
	Reset value																	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0x028	OTG_HS_GNPTXFSIZ (Host mode)	NPTXFD																NPTXFSA															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
	OTG_HS_GNPTXFSIZ (peripheral mode)	TX0FD																TX0FSA															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0x02C	OTG_HS_GNPTXSTS	Res.	NPTXQTOP							NPTQXSAV								NPTXFSAV															
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0x030	OTG_HS_GI2CCTL	BSYDNE	RW	Reserved	I2CDATSE0	I2CDEVADR		Reserved	ACK	I2CEN	ADDR							REGADDR								RWDATA							
	Reset value	0	0		0	0	0		0	0	0	0	0	0	0	0	0																

表 184. OTG_HS 寄存器映射和复位值 （续）

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x038	OTG_HS_GCCFG	Reserved										NOVBUSSENS	SOFOUTEN	VBUSBSEN	VBUSASEN	.I2CPADEN	.PWRDWN	Reserved															
	Reset value											0	0	0	0	0	0																
0x03C	OTG_HS_CID	PRODUCT_ID																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0
0x100	OTG_HS_HPTXFSIZ	PTXFD																PTXSA															
	Reset value	0	0	0	0	0	1	1	1	0	1	1	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	1	0	0
0x104	OTG_HS_DIEPTXF1	INEPTXFD																INEPTXSA															
	Reset value	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
0x108	OTG_HS_DIEPTXF2	INEPTXFD																INEPTXSA															
	Reset value	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
0x10C	OTG_HS_DIEPTXF3	INEPTXFD																INEPTXSA															
	Reset value	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
0x110	OTG_HS_DIEPTXF4	INEPTXFD																INEPTXSA															
	Reset value	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
0x400	OTG_HS_HCFG	Reserved																													FSLSS	FSLSPCS	
	Reset value																														0	0	0
0x404	OTG_HS_HFIR	Reserved																FRIVL															
	Reset value																	1	1	1	0	1	0	1	0	0	1	1	0	0	0	0	0
0x408	OTG_HS_HFNUM	FTREM																FRNUM															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x410	OTG_HS_HPTXSTS	PTXQTOP								PTXQSAV								PTXFSAVL															
	Reset value	0	0	0	0	0	0	0	0	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
0x414	OTG_HS_HAINT	Reserved																HAINT															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x418	OTG_HS_HAINTMSK	Reserved																HAINTM															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x440	OTG_HS_HPRT	Reserved													PSPD			PTCTL			PPWR	PLSTS	Reserved	PRST	PSUSP	PRES	POCCHNG	POCA	PENCHNG	PENA	PCDET	PCSTS
	Reset value														0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x500	OTG_HS_HCCHAR0	CHENA	CHDIS	ODDFRM	DAD							MC		EPTYP		LSDEV	Reserved	EPDIR	EPNUM				MPSIZ										
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x520	OTG_HS_HCCHAR1	CHENA	CHDIS	ODDFRM	DAD							MC		EPTYP		LSDEV	Reserved	EPDIR	EPNUM				MPSIZ										
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x540	OTG_HS_HCCHAR2	CHENA	CHDIS	ODDFRM	DAD							MC		EPTYP		LSDEV	Reserved	EPDIR	EPNUM				MPSIZ										
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x560	OTG_HS_HCCHAR3	CHENA	CHDIS	ODDFRM	DAD							MC		EPTYP		LSDEV	Reserved	EPDIR	EPNUM				MPSIZ										
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

表 184. OTG_HS 寄存器映射和复位值 （续）

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x580	OTG_HS_HCCHAR4	CHENA	CHDIS	ODDFRM	DAD							MC		EPTYP		LSDEV	Reserved	EPDIR	EPNUM				MPSIZ										
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x5A0	OTG_HS_HCCHAR5	CHENA	CHDIS	ODDFRM	DAD							MC		EPTYP		LSDEV	Reserved	EPDIR	EPNUM				MPSIZ										
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x5C0	OTG_HS_HCCHAR6	CHENA	CHDIS	ODDFRM	DAD							MC		EPTYP		LSDEV	Reserved	EPDIR	EPNUM				MPSIZ										
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x5E0	OTG_HS_HCCHAR7	CHENA	CHDIS	ODDFRM	DAD							MC		EPTYP		LSDEV	Reserved	EPDIR	EPNUM				MPSIZ										
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x600	OTG_HS_HCCHAR8	CHENA	CHDIS	ODDFRM	DAD							MC		EPTYP		LSDEV	Reserved	EPDIR	EPNUM				MPSIZ										
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x620	OTG_HS_HCCHAR9	CHENA	CHDIS	ODDFRM	DAD							MC		EPTYP		LSDEV	Reserved	EPDIR	EPNUM				MPSIZ										
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x640	OTG_HS_HCCHAR10	CHENA	CHDIS	ODDFRM	DAD							MC		EPTYP		LSDEV	Reserved	EPDIR	EPNUM				MPSIZ										
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x660	OTG_HS_HCCHAR11	CHENA	CHDIS	ODDFRM	DAD							MC		EPTYP		LSDEV	Reserved	EPDIR	EPNUM				MPSIZ										
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x504	OTG_HS_HCSPLT0	SPLITEN	Reserved														COMPLSPLT	XACTPOS	HUBADDR								PRTADDR						
	Reset value	0															0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x508	OTG_HS_HCINT0	Reserved																					DTERR	FRMOR	BBERR	TXERR	NYET	ACK	NAK	STALL	AHBERR	CHH	XFRC
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
0x524	OTG_HS_HCSPL1	SPLITEN	Reserved														COMPLSPLT	XACTPOS	HUBADDR								PRTADDR						
	Reset value	0															0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x528	OTG_HS_HCINT1	Reserved																					DTERR	FRMOR	BBERR	TXERR	NYET	ACK	NAK	STALL	AHBERR	CHH	XFRC
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
0x544	OTG_HS_HCSPLT2	SPLITEN	Reserved														COMPLSPLT	XACTPOS	HUBADDR								PRTADDR						
	Reset value	0															0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

表 184. OTG_HS 寄存器映射和复位值 （续）

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x548	OTG_HS_HCINT2	Reserved																					DTERR	FRMOR	BBERR	TXERR	NYET	ACK	NAK	STALL	AHBERR	CHH	XFRC
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
0x564	OTG_HS_HCSPLT3	SPLITEN	Reserved														COMPLSPLT	XACTPOS		HUBADDR							PRTADDR							
	Reset value	0															0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x568	OTG_HS_HCINT3	Reserved																					DTERR	FRMOR	BBERR	TXERR	NYET	ACK	NAK	STALL	AHBERR	CHH	XFRC
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
0x584	OTG_HS_HCSPLT4	SPLITEN	Reserved														COMPLSPLT	XACTPOS		HUBADDR							PRTADDR							
	Reset value	0															0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x588	OTG_HS_HCINT4	Reserved																					DTERR	FRMOR	BBERR	TXERR	NYET	ACK	NAK	STALL	AHBERR	CHH	XFRC
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
0x5A4	OTG_HS_HCSPLT5	SPLITEN	Reserved														COMPLSPLT	XACTPOS		HUBADDR							PRTADDR							
	Reset value	0															0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x5A8	OTG_HS_HCINT5	Reserved																					DTERR	FRMOR	BBERR	TXERR	NYET	ACK	NAK	STALL	AHBERR	CHH	XFRC
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
0x5C4	OTG_HS_HCSPLT6	SPLITEN	Reserved														COMPLSPLT	XACTPOS		HUBADDR							PRTADDR							
	Reset value	0															0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x5C8	OTG_HS_HCINT6	Reserved																					DTERR	FRMOR	BBERR	TXERR	NYET	ACK	NAK	STALL	AHBERR	CHH	XFRC
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
0x5E4	OTG_HS_HCSPLT7	SPLITEN	Reserved														COMPLSPLT	XACTPOS		HUBADDR							PRTADDR							
	Reset value	0															0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x5E8	OTG_HS_HCINT7	Reserved																					DTERR	FRMOR	BBERR	TXERR	NYET	ACK	NAK	STALL	AHBERR	CHH	XFRC
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
0x604	OTG_HS_HCSPLT8	SPLITEN	Reserved														COMPLSPLT	XACTPOS		HUBADDR							PRTADDR							
	Reset value	0															0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

表 184. OTG_HS 寄存器映射和复位值 （续）

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x608	OTG_HS_HCINT8	Reserved																					DTERR	FRMOR	BBERR	TXERR	NYET	ACK	NAK	STALL	AHBERR	CHH	XFRC
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
0x624	OTG_HS_HCSPLT9	SPLITEN	Reserved														COMPLSPLT	XACTPOS		HUBADDR							PRTADDR							
	Reset value	0															0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x628	OTG_HS_HCINT9	Reserved																					DTERR	FRMOR	BBERR	TXERR	NYET	ACK	NAK	STALL	AHBERR	CHH	XFRC
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
0x644	OTG_HS_HCSPLT10	SPLITEN	Reserved														COMPLSPLT	XACTPOS		HUBADDR							PRTADDR							
	Reset value	0															0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x648	OTG_HS_HCINT10	Reserved																					DTERR	FRMOR	BBERR	TXERR	NYET	ACK	NAK	STALL	AHBERR	CHH	XFRC
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
0x664	OTG_HS_HCSPLT11	SPLITEN	Reserved														COMPLSPLT	XACTPOS		HUBADDR							PRTADDR							
	Reset value	0															0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x668	OTG_HS_HCINT11	Reserved																					DTERR	FRMOR	BBERR	TXERR	NYET	ACK	NAK	STALL	AHBERR	CHH	XFRC
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
0x50C	OTG_HS_HCINTMSK0	Reserved																					DTERRM	FRMORM	BBERRM	TXERRM	NYET	ACKM	NAKM	STALLM		CHHM	XFRCM
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
0x52C	OTG_HS_HCINTMSK1	Reserved																					DTERRM	FRMORM	BBERRM	TXERRM	NYET	ACKM	NAKM	STALLM	AHBERR	CHHM	XFRCM
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
0x54C	OTG_HS_HCINTMSK2	Reserved																					DTERRM	FRMORM	BBERRM	TXERRM	NYET	ACKM	NAKM	STALLM	AHBERR	CHHM	XFRCM
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
0x56C	OTG_HS_HCINTMSK3	Reserved																					DTERRM	FRMORM	BBERRM	TXERRM	NYET	ACKM	NAKM	STALLM	AHBERR	CHHM	XFRCM
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
0x58C	OTG_HS_HCINTMSK4	Reserved																					DTERRM	FRMORM	BBERRM	TXERRM	NYET	ACKM	NAKM	STALLM	AHBERR	CHHM	XFRCM
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
0x5AC	OTG_HS_HCINTMSK5	Reserved																					DTERRM	FRMORM	BBERRM	TXERRM	NYET	ACKM	NAKM	STALLM	AHBERR	CHHM	XFRCM
	Reset value																						0	0	0	0	0	0	0	0	0	0	0

表 184. OTG_HS 寄存器映射和复位值 （续）

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x5CC	**OTG_HS_HCINTMSK6**	Reserved																					DTERRM	FRMORM	BBERRM	TXERRM	NYET	ACKM	NAKM	STALLM	AHBERR	CHHM	XFRCM
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
0x5EC	**OTG_HS_HCINTMSK7**	Reserved																					DTERRM	FRMORM	BBERRM	TXERRM	NYET	ACKM	NAKM	STALLM	AHBERR	CHHM	XFRCM
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
0x60C	**OTG_HS_HCINTMSK8**	Reserved																					DTERRM	FRMORM	BBERRM	TXERRM	NYET	ACKM	NAKM	STALLM	AHBERR	CHHM	XFRCM
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
0x62C	**OTG_HS_HCINTMSK9**	Reserved																					DTERRM	FRMORM	BBERRM	TXERRM	NYET	ACKM	NAKM	STALLM	AHBERR	CHHM	XFRCM
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
0x64C	**OTG_HS_HCINTMSK10**	Reserved																					DTERRM	FRMORM	BBERRM	TXERRM	NYET	ACKM	NAKM	STALLM	AHBERR	CHHM	XFRCM
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
0x66C	**OTG_HS_HCINTMSK11**	Reserved																					DTERRM	FRMORM	BBERRM	TXERRM	NYET	ACKM	NAKM	STALLM	AHBERR	CHHM	XFRCM
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
0x510	**OTG_HS_HCTSIZ0**	Reserved	DPID		PKTCNT										XFRSIZ																		
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x530	**OTG_HS_HCTSIZ1**	Reserved	DPID		PKTCNT										XFRSIZ																		
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x550	**OTG_HS_HCTSIZ2**	Reserved	DPID		PKTCNT										XFRSIZ																		
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x570	**OTG_HS_HCTSIZ3**	Reserved	DPID		PKTCNT										XFRSIZ																		
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x590	**OTG_HS_HCTSIZ4**	Reserved	DPID		PKTCNT										XFRSIZ																		
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x5B0	**OTG_HS_HCTSIZ5**	Reserved	DPID		PKTCNT										XFRSIZ																		
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x5D0	**OTG_HS_HCTSIZ6**	Reserved	DPID		PKTCNT										XFRSIZ																		
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x5F0	**OTG_HS_HCTSIZ7**	Reserved	DPID		PKTCNT										XFRSIZ																		
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x610	**OTG_HS_HCTSIZ8**	Reserved	DPID		PKTCNT										XFRSIZ																		
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x630	**OTG_HS_HCTSIZ9**	Reserved	DPID		PKTCNT										XFRSIZ																		
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

表 184. OTG_HS 寄存器映射和复位值 （续）

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x650	**OTG_HS_HCTSIZ10**	Reserved	DPID		PKTCNT										XFRSIZ																		
	Reset value	Reserved	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x670	**OTG_HS_HCTSIZ11**	Reserved	DPID		PKTCNT										XFRSIZ																		
	Reset value	Reserved	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x514	**OTG_HS_HCDMA0**	DMAADDR																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x524	**OTG_HS_HCDMA1**	DMAADDR																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x544	**OTG_HS_HCDMA2**	DMAADDR																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x564	**OTG_HS_HCDMA3**	DMAADDR																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x584	**OTG_HS_HCDMA4**	DMAADDR																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x5A4	**OTG_HS_HCDMA5**	DMAADDR																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x5C4	**OTG_HS_HCDMA6**	DMAADDR																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x5E4	**OTG_HS_HCDMA7**	DMAADDR																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x604	**OTG_HS_HCDMA8**	DMAADDR																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x624	**OTG_HS_HCDMA9**	DMAADDR																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x644	**OTG_HS_HCDMA10**	DMAADDR																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x664	**OTG_HS_HCDMA11**	DMAADDR																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x800	**OTG_HS_DCFG**	Reserved						PERSCHIVL		Reserved	Reserved										PFIVL		DAD							Reserved	NZLSOHSK	DSPD	
	Reset value							1	0												0	0	0	0	0	0	0	0	0	0	0	0	0
0x804	**OTG_HS_DCTL**	Reserved																				POPRGDNE	CGONAK	SGONAK	CGINAK	SGINAK	TCTL			GONSTS	GINSTS	SDIS	RWUSIG
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0
0x808	**OTG_HS_DSTS**	Reserved													FNSOF															Reserved	EERR	ENUMSPD	SUSPSTS	
	Reset value														0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0
0x810	**OTG_HS_DIEPMSK**	Reserved																							BIM	TXFURM	Reserved	INEPNEM	INEPNMM	ITTXFEMSK	TOM	Reserved	EPDM	XFRCM
	Reset value																							0	0	0	0	0	0	0	0		0	0

表 184. OTG_HS 寄存器映射和复位值 （续）

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x814	OTG_HS_DOEPMSK	Reserved																						BOIM	OPEM	Reserved	B2BSTUP	Reserved	OTEPDM	STUPM	Reserved	EPDM	XFRCM
	Reset value																							0	0		0	0	0	0		0	0
0x818	OTG_HS_DAINT	OEPINT																IEPINT															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x81C	OTG_HS_DAINTMSK	OEPM																IEPM															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x828	OTG_HS_DVBUSDIS	Reserved																VBUSDT															
	Reset value																	0	0	0	1	0	1	1	1	1	1	0	1	0	1	1	1
0x82C	OTG_HS_DVBUSPULSE	Reserved																				DVBUSP											
	Reset value																					0	1	0	1	1	0	1	1	1	0	0	0
0x830	OTG_HS_DTHRCTL	Reserved				ARPEN	Reserved	RXTHRLEN									RXTHREN	Reserved				TXTHRLEN									ISOTHREN	NONISOTHREN	
	Reset value					0		0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0
0x834	OTG_HS_DIEPEMPMSK	Reserved																INEPTXFEM															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x838	OTG_HS_DEACHINT	Reserved																Reserved															Reserved
	Reset value															0																0	
0x83C	OTG_HS_DEACHINTMSK	Reserved																Reserved															Reserved
	Reset value															0																0	
0x844	OTG_HS_DIEPEACHMSK1	Reserved																	NAKM	Reserved		BIM	TXFURM	Reserved	INEPNEM	INEPNMM	ITTXFEMSK	TOM	Reserved	EPDM	XFRCM	
	Reset value																		0			0	0		0	0	0	0		0	0		
0x884	OTG_HS_DOEPEACHMSK1	Reserved																NYETM	NAKM	BERRM	Reserved	BIM	TXFURM	Reserved	INEPNEM	INEPNMM	ITTXFEMSK	TOM	Reserved	EPDM	XFRCM	
	Reset value																	0	0	0		0	0		0	0	0	0		0	0		
0x900	OTG_HS_DIEPCTL0	EPENA	EPDIS	SODDFRM	SD0PID/SEVNFRM	SNAK	CNAK	TXFNUM				STALL	Reserved	EPTYP		NAKSTS	EONUM/DPID	USBAEP	Reserved					MPSIZ									
	Reset value	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0						0	0	0	0	0	0	0	0	0	0
0x918	TG_FS_DTXFSTS0	Reserved																INEPTFSAV															
	Reset value																	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0x920	OTG_HS_DIEPCTL1	EPENA	EPDIS	SODDFRM	SD0PID/SEVNFRM	SNAK	CNAK	TXFNUM				Stall	Reserved	EPTYP		NAKSTS	EONUM/DPID	USBAEP	Reserved					MPSIZ									
	Reset value	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0						0	0	0	0	0	0	0	0	0	0
0x938	TG_FS_DTXFSTS1	Reserved																INEPTFSAV															
	Reset value																	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0

表 184. OTG_HS 寄存器映射和复位值 （续）

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x940	**OTG_HS_DIEPCTL2**	EPENA	EPDIS	SODDFRM	SD0PID/SEVNFRM	SNAK	CNAK	TXFNUM				Stall	Reserved	EPTYP		NAKSTS	EONUM/DPID	USBAEP	Reserved				MPSIZ										
	Reset value	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0
0x958	**TG_FS_DTXFSTS2**	Reserved																	INEPTFSAV														
	Reset value																		0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
0x960	**OTG_HS_DIEPCTL3**	EPENA	EPDIS	SODDFRM	SD0PID/SEVNFRM	SNAK	CNAK	TXFNUM				Stall	Reserved	EPTYP		NAKSTS	EONUM/DPID	USBAEP	Reserved				MPSIZ										
	Reset value	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0
0x978	**TG_FS_DTXFSTS3**	Reserved																	INEPTFSAV														
	Reset value																		0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
0x980	**OTG_HS_DIEPCTL4**	EPENA	EPDIS	SODDFRM	SD0PID/SEVNFRM	SNAK	CNAK	TXFNUM				Stall	Reserved	EPTYP		NAKSTS	EONUM/DPID	USBAEP	Reserved				MPSIZ										
	Reset value	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0
0x9A0	**OTG_HS_DIEPCTL5**	EPENA	EPDIS	SODDFRM	SD0PID/SEVNFRM	SNAK	CNAK	TXFNUM				STALL	Reserved	EPTYP		NAKSTS	EONUM/DPID	USBAEP	Reserved				MPSIZ										
	Reset value	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0
0x9C0	**OTG_HS_DIEPCTL6**	EPENA	EPDIS	SODDFRM	SD0PID/SEVNFRM	SNAK	CNAK	TXFNUM				STALL	Reserved	EPTYP		NAKSTS	EONUM/DPID	USBAEP	Reserved				MPSIZ										
	Reset value	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0
0x9E0	**OTG_HS_DIEPCTL7**	EPENA	EPDIS	SODDFRM	SD0PID/SEVNFRM	SNAK	CNAK	TXFNUM				STALL	Reserved	EPTYP		NAKSTS	EONUM/DPID	USBAEP	Reserved				MPSIZ										
	Reset value	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0
0xB00	**OTG_HS_DOEPCTL0**	EPENA	EPDIS	Reserved		SNAK	CNAK	Reserved				STALL	SNPM	EPTYP		NAKSTS	Reserved	USBAEP	Reserved													MPSIZ	
	Reset value	0	0			0	0					0	0	0	0	0		1														0	0
0xB20	**OTG_HS_DOEPCTL1**	EPENA	EPDIS	SODDFRM	SD0PID/SEVNFRM	SNAK	CNAK	Reserved				STALL	SNPM	EPTYP		NAKSTS	EONUM/DPID	USBAEP	Reserved				MPSIZ										
	Reset value	0	0	0	0	0	0					0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0

表 184. OTG_HS 寄存器映射和复位值 （续）

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xB40	OTG_HS_DOEPCTL2	EPENA	EPDIS	SODDFRM	SD0PID/SEVNFRM	SNAK	CNAK	Reserved				Stall	SNPM	EPTYP		NAKSTS	EONUM/DPID	USBAEP	Reserved				MPSIZ										
	Reset value	0	0	0	0	0	0					0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0
0xB60	OTG_HS_DOEPCTL3	EPENA	EPDIS	SODDFRM	SD0PID/SEVNFRM	SNAK	CNAK	Reserved				Stall	SNPM	EPTYP		NAKSTS	EONUM/DPID	USBAEP	Reserved				MPSIZ										
	Reset value	0	0	0	0	0	0					0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0
0x908	OTG_HS_DIEPINT0	Reserved																		NAK	BERR	PKTDRPSTS	Reserved	BNA	TXFIFOUDRN	TXFE	INEPNE	Reserved	ITTXFE	TOC	Reserved	EPDISD	XFRC
	Reset value																			0	0	0		0	0	1	0		0	0		0	0
0x928	OTG_HS_DIEPINT1	Reserved																		NAK	BERR	PKTDRPSTS	Reserved	BNA	TXFIFOUDRN	TXFE	INEPNE	Reserved	ITTXFE	TOC	Reserved	EPDISD	XFRC
	Reset value																			0	0	0		0	0	1	0		0	0		0	0
0x948	OTG_HS_DIEPINT2	Reserved																		NAK	BERR	PKTDRPSTS	Reserved	BNA	TXFIFOUDRN	TXFE	INEPNE	Reserved	ITTXFE	TOC	Reserved	EPDISD	XFRC
	Reset value																			0	0	0		0	0	1	0		0	0		0	0
0x968	OTG_HS_DIEPINT3	Reserved																		NAK	BERR	PKTDRPSTS	Reserved	BNA	TXFIFOUDRN	TXFE	INEPNE	Reserved	ITTXFE	TOC	Reserved	EPDISD	XFRC
	Reset value																			0	0	0		0	0	1	0		0	0		0	0
0x988	OTG_HS_DIEPINT4	Reserved																		NAK	BERR	PKTDRPSTS	Reserved	BNA	TXFIFOUDRN	TXFE	INEPNE	Reserved	ITTXFE	TOC	Reserved	EPDISD	XFRC
	Reset value																			0	0	0		0	0	1	0		0	0		0	0
0x9A8	OTG_HS_DIEPINT5	Reserved																		NAK	BERR	PKTDRPSTS	Reserved	BNA	TXFIFOUDRN	TXFE	INEPNE	Reserved	ITTXFE	TOC	Reserved	EPDISD	XFRC
	Reset value																			0	0	0		0	0	1	0		0	0		0	0
0x9C8	OTG_HS_DIEPINT6	Reserved																		NAK	BERR	PKTDRPSTS	Reserved	BNA	TXFIFOUDRN	TXFE	INEPNE	Reserved	ITTXFE	TOC	Reserved	EPDISD	XFRC
	Reset value																			0	0	0		0	0	1	0		0	0		0	0
0x9E8	OTG_HS_DIEPINT7	Reserved																		NAK	BERR	PKTDRPSTS	Reserved	BNA	TXFIFOUDRN	TXFE	INEPNE	Reserved	ITTXFE	TOC	Reserved	EPDISD	XFRC
	Reset value																			0	0	0		0	0	1	0		0	0		0	0

表 184. OTG_HS 寄存器映射和复位值 （续）

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xB08	**OTG_HS_DOEPINT0**	Reserved																	NYET	Reserved							B2BSTUP	Reserved	OTEPDIS	STUP	Reserved	EPDISD	XFRC
	Reset value																		0								0		0	1		0	0
0xB28	**OTG_HS_DOEPINT1**	Reserved																	NYET	Reserved							B2BSTUP	Reserved	OTEPDIS	STUP	Reserved	EPDISD	XFRC
	Reset value																		0								0		0	0		0	0
0xB48	**OTG_HS_DOEPINT2**	Reserved																	NYET	Reserved							B2BSTUP	Reserved	OTEPDIS	STUP	Reserved	EPDISD	XFRC
	Reset value																		0								0		0	0		0	0
0xB68	**OTG_HS_DOEPINT3**	Reserved																	NYET	Reserved							B2BSTUP	Reserved	OTEPDIS	STUP	Reserved	EPDISD	XFRC
	Reset value																		0								0		0	0		0	0
0xB88	**OTG_HS_DOEPINT4**	Reserved																	NYET	Reserved							B2BSTUP	Reserved	OTEPDIS	STUP	Reserved	EPDISD	XFRC
	Reset value																		0								0		0	0		0	0
0xBA8	**OTG_HS_DOEPINT5**	Reserved																	NYET	Reserved							B2BSTUP	Reserved	OTEPDIS	STUP	Reserved	EPDISD	XFRC
	Reset value																		0								0		0	0		0	0
0xBC8	**OTG_HS_DOEPINT6**	Reserved																	NYET	Reserved							B2BSTUP	Reserved	OTEPDIS	STUP	Reserved	EPDISD	XFRC
	Reset value																		0								0		0	0		0	0
0xBE8	**OTG_HS_DOEPINT7**	Reserved																	NYET	Reserved							B2BSTUP	Reserved	OTEPDIS	STUP	Reserved	EPDISD	XFRC
	Reset value																		0								0		0	0		0	0
0x910	**OTG_HS_DIEPTSIZ0**	Reserved											PKTCNT		Reserved												XFRSIZ						
	Reset value												0	0													0	0	0	0	0	0	0
0x930	**OTG_HS_DIEPTSIZ1**	Reserved	MCNT		PKTCNT										XFRSIZ																		
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x934	**OTG_HS_DIEPDMA1**	DMAADDR																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x93C	**OTG_HS_DIEPDMAB1**	DMABADDR																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x950	**OTG_HS_DIEPTSIZ2**	Reserved	MCNT		PKTCNT										XFRSIZ																		
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x954	**OTG_HS_DIEPDMA2**	DMAADDR																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x95C	**OTG_HS_DIEPDMAB2**	DMABADDR																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x970	**OTG_HS_DIEPTSIZ3**	Reserved	MCNT		PKTCNT										XFRSIZ																		
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x974	**OTG_HS_DIEPDMA3**	DMAADDR																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

表 184. OTG_HS 寄存器映射和复位值 （续）

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x97C	OTG_HS_DIEPDMAB3	DMABADDR																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xB10	OTG_HS_DOEPTSIZ0	Reserved	STUPCNT		Reserved									PKTCNT	Reserved												XFRSIZ						
	Reset value		0	0										0													0	0	0	0	0	0	0
0xB30	OTG_HS_DOEPTSIZ1	Reserved	RXDPID/STUPCNT		PKTCNT										XFRSIZ																		
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xB34	OTG_HS_DOEPDMA1	DMAADDR																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xB3C	OTG_HS_DOEPDMAB1	DMABADDR																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xB50	OTG_HS_DOEPTSIZ2	Reserved	RXDPID/STUPCNT		PKTCNT										XFRSIZ																		
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xB54	OTG_HS_DOEPDMA2	DMAADDR																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xB5C	OTG_HS_DOEPDMAB2	DMABADDR																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xB70	OTG_HS_DOEPTSIZ3	Reserved	RXDPID/STUPCNT		PKTCNT										XFRSIZ																		
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xB74	OTG_HS_DOEPDMA3	DMAADDR																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xB7C	OTG_HS_DOEPDMAB3	DMABADDR																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xE00	OTG_HS_PCGCCTL	Reserved																											PHYSUSP	Reserved		GATEHCLK	STPPCLK
	Reset value																																

有关寄存器边界地址的信息，请参见*第 52 页的表 2*。

31.13 OTG_HS 编程模型

31.13.1 模块初始化

应用程序必须执行模块初始化序列。如果在上电期间连接 USB 线，则控制器的当前模式在模块中断寄存器的当前操作模式位（OTG_HS_GINTSTS 中的 CMOD 位）中反映。连接“A 型”插头后，OTG_HS 控制器进入主机模式；连接“B 型”插头后，OTG_HS 控制器进入设备模式。

本节介绍了 OTG_HS 控制器在上电后的初始化过程。无论是以主机模式还是设备模式工作，应用程序都必须遵循初始化序列。根据模块配置对所有模块全局寄存器进行初始化：

1. 在全局 AHB 配置 (OTG_HS_GAHBCFG) 寄存器中编程以下字段：
   - DMA 模式位
   - AHB 突发传输长度字段
   - 全局中断屏蔽位 GINT = 1
   - RxFIFO 非空（OTG_HS_GINTSTS 中的 RXFLVL 位）
   - 周期性 TxFIFO 空门限
2. 在 OTG_HS_GUSBCFG 寄存器中编程以下字段：
   - HNP 功能位
   - SRP 功能位
   - FS 超时校准字段
   - USB 周转时间字段
3. 软件必须取消对 GINTMSK 寄存器中以下位的屏蔽：

   OTG 中断屏蔽

   模式不匹配中断屏蔽
4. 通过读取 OTG_HS_GINTSTS 中的 CMOD 位，软件可确定 OTG_HS 控制器是在主机模式还是设备模式下工作。

31.13.2 主机初始化

要将模块作为主机进行初始化，应用程序必须执行以下步骤：

1. 编程 GINTMSK 中的 PRTIM 以取消屏蔽。
2. 编程 OTG_HS_HCFG 寄存器以选择全速主机。
3. 将 OTG_HS_HPRT 中的 PPWR 位编程为 1，给 USB 总线提供 V_{BUS}。
4. 等待 OTG_HS_HPRT0 中的 PCDET 中断。这表示某设备已连接到主机端口。
5. 将 OTG_HS_HPRT 中的 PRST 位编程为 1，在 USB 总线上发出复位信号。
6. 至少等待 10 ms，以便完成复位过程。
7. 将 OTG_HS_HPRT 中的 PRST 位编程为 0。
8. 等待 OTG_HS_HPRT 中的 PENCHNG 中断。
9. 读取 OTG_HS_HPRT 中的 PSPD 位以获取枚举速度。
10. 使用所选 PHY 时钟，相应地设置 HFIR 寄存器。
11. 根据步骤 9 中读取的被检测设备的速度对 OTG_FS_HCFG 寄存器中的 FSLSPCS 字段进行编程。如果 FSLSPCS 发生更改，则执行端口复位。
12. 编程 OTG_HS_GRXFSIZ 寄存器以选择接收 FIFO 的大小。
13. 编程 OTG_HS_GNPTXFSIZ 寄存器，以选择用于非周期性通信事务的非周期性发送 FIFO 的大小和起始地址。
14. 编程 OTG_HS_HPTXFSIZ 寄存器，以选择用于周期性通信事务的周期性发送 FIFO 的大小和起始地址。

要与设备通信，系统软件必须初始化并使能至少一个通道。

31.13.3 设备初始化

上电期间或者从主机模式切换为设备模式后，应用程序必须执行下列步骤来将模块作为设备进行初始化。

1. 在 OTG_HS_DCFG 寄存器中编程以下字段：
   - 设备速度
   - 非零长度状态 OUT 握手信号
2. 编程 OTG_HS_GINTMSK 寄存器以取消屏蔽以下中断：
   - USB 复位
   - 枚举完成
   - 早期挂起
   - USB 挂起
   - SOF
3. 编程 OTG_HS_GCCFG 寄存器中的 VBUSBSEN 位，以使能“B”设备模式的 V_{BUS} 感应并把 DP 线上的上拉电阻上拉到 5V。
4. 等待 OTG_HS_GINTSTS 中的 USBRST 中断。这表示已在 USB 上检测到复位信号，复位过程自接收到此中断后约持续 10 ms。

等待 OTG_HS_GINTSTS 中的 ENUMDNE 中断。此中断指示 USB 上复位过程结束。接收到此中断时，应用程序必须读取 OTG_HS_DSTS 寄存器以确定枚举速度并执行*第 1168 页的枚举完成时的端点初始化*中所列的步骤。

此时，设备已准备好接受 SOF 数据包并在控制端点 0 上执行控制传输。

31.13.4 DMA 模式

OTG 主机使用 AHB 主接口来获取发送数据包数据（AHB 到 USB）和接收数据更新（USB 到 AHB）。AHB 主接口使用经过编程的 DMA 地址（主机模式下的 HCDMAx 寄存器和设备模式下的 DIEPDMAx/DOEPDMAx 寄存器）来访问数据缓冲区。

31.13.5 主机编程模型

通道初始化

应用程序必须初始化一个或多个通道，之后才能与所连接的设备通信。要初始化和使能通道，应用程序必须执行以下步骤：

1. 编程 GINTMSK 寄存器以取消对以下位的中断屏蔽：
2. 通道中断
   - 用于 OUT 事务的非周期性发送 FIFO 为空（适用于从模式，该模式在流水线事务级别工作且数据包计数字段被编程为大于 1）。
   - 用于 OUT 事务的非周期性发送 FIFO 为半空（适用于从模式，该模式在流水线事务级别工作且数据包计数字段被编程为大于 1）。
3. 编程 OTG_HS_HAINTMSK 寄存器以使能所选通道中断。
4. 编程 OTG_HS_HCINTMSK 寄存器，以使能主机通道中断寄存器中反映的和通信事务有关的中断。
5. 编程所选通道的 OTG_HS_HCTSIZx 寄存器，指定以字节为单位的总传输大小和包括短数据包在内的预期数据包个数。应用程序必须使用初始数据 PID（用于第一个 OUT 事务或预期从第一个 IN 事务获取）编程 PID 字段。

6. 使用集线器地址和端口地址在 OTG_HS_HCSPLTx 寄存器中对所选通道进行编程（仅分离事务）。
7. 使用缓冲区起始地址在 HCDMAx 寄存器中对所选通道进行编程。
8. 编程所选通道的 OTG_HS_HCCHARx 寄存器，指定设备的端点特性，例如类型、速度、方向等。（仅当应用程序准备好发送或接收数据包时，才能通过将通道使能位置 1 来使能通道）

通道的停止

应用程序可以通过编程 OTG_HS_HCCHARx 寄存器将 CHDIS 和 CHENA 位置 1 来禁止任何通道。这会使 OTG_HS 主机清空之前在该通道上发出的请求（如果有）并生成通道停止中断。应用程序在将通道重新分配给其它通信事务之前，必须等待 OTG_HS_HCINTx 中的 CHH 中断。OTG_HS 主机不会中断已在 USB 上启动的通信事务。

要禁止某个在 DMA 模式下工作的通道，应用程序无需检查请求队列中是否有可用空间。OTG_HS 主机将检查是否有空间，在通道仲裁给要禁止的通道时，写入通道禁止的请求。同时，当 HCCHARx 中的 CHDIS 位置 1 时，将从请求队列中丢弃所有已发出的请求。

禁止通道前，应用程序必须确保非周期性请求队列（禁止非周期性通道时）或周期性请求队列（禁止周期性通道时）中至少有一个空闲空间。应用程序可以在请求队列已满时（禁止通道之前），通过编程 OTG_HS_HCCHARx 寄存器将 CHDIS 位置 1 和将 CHENA 位清零，清空请求队列。

出现以下任一情况时，应用程序将禁止通道：

1. 在非周期性 IN 传输或高带宽中断 IN 传输期间 OTG_HS_HCINTx 中接收到 XFRC 中断（仅限从模式）。
2. IN 或 OUT 通道的 OTG_HS_HCINTx 中接收到 STALL、TXERR、BBERR 或 DTERR 中断（仅限从模式）。对于从模式下的高带宽中断 IN 通道，一旦应用程序接收到 DTERR 中断，就必须禁用通道并等待通道停止中断。应用程序在接收到通道停止信号之前，必须能够接收相同通道的其它中断（DTERR、NAK、Data、TXERR）。
3. 接收到 OTG_HS_GINTSTS 中的 DISCINT（断开设备连接）中断。（应用程序将禁止所有已使能的通道）
4. 应用程序在传输正常完成之前将其中止。

Ping 协议

当 OTG_HS 主机工作在高速模式下时，如果应用程序要与高速批量或控制（数据和状态阶段）OUT 端点进行通信，则必须使用 ping 协议。

当应用程序接收到 NAK/NYET/TXERR 中断时，必须使用 ping 协议。当 HS_OTG 主机接收到上述其中一个响应时，它不会在该端点上继续执行任何通信事务，而是丢弃所有已发出或已获取的 OUT 请求（从请求队列中），然后清空发送 FIFO 中的相应数据。

这仅在从模式下有效。在从模式下，应用程序可通过以下两种方式发送 ping 令牌：在使能通道之前，将 HCTSIZx 中的 DOPING 位置 1，或者在通道已使能之后，对 HCTSIZx 寄存器执行写操作时将 DOPING 位置 1。这将使能 HS_OTG 主机将 ping 请求写入到请求队列中。应用程序必须等待设备对 ping 令牌的响应（NAK、ACK 或 TXERR 中断），然后才能继续执行通信事务或发送另一个 ping 令牌。仅当应用程序从设备的 OUT 端点接收到对 ping 令牌的 ACK 响应后，才能继续执行数据通信事务。在 DMA 模式下工作时，对于批量/控制 OUT 事务，应用程序不需要在收到 NAK/NYET 回复时将 HCTSIZx 中的 DOPING 位置 1。OTG_HS 主机会自动将 HCTSIZx 中的 DOPING 位置 1，然后在批量/控制 OUT 传输时发出 ping 令牌。HS_OTG 主机会持续发送 ping 令牌，直至收到 ACK 为止，随后自动切换到数据通信事务。

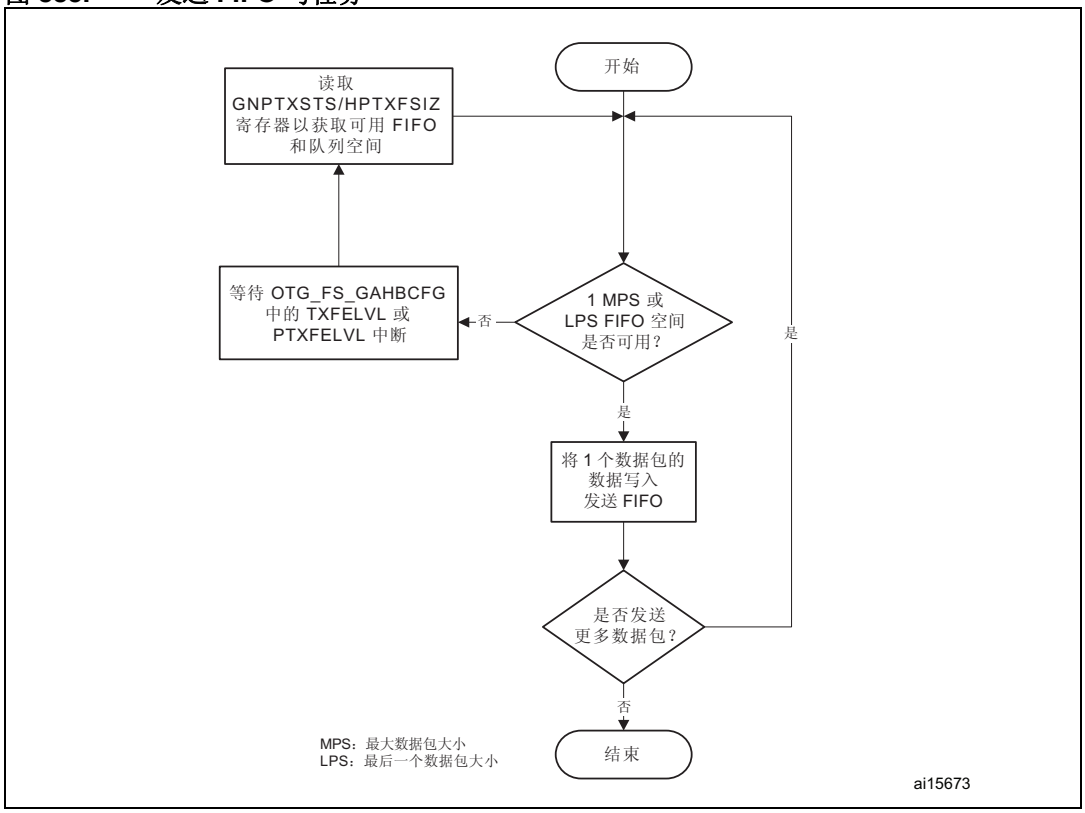
操作模型

应用程序必须初始化一个通道，之后才能与所连接的设备通信。本节介绍了针对不同 USB 事务类型要执行的操作序列。

- **写入发送 FIFO**

应用程序对数据包执行最后一个 DWORD 写操作的同时，OTG_HS 主机自动向周期性/非周期性请求队列写入一个条目（OUT 请求）。因此开始向发送 FIFO 写入数据之前，应用程序必须确保周期性/非周期性请求队列中至少有一个空闲空间。应用程序必须始终以 DWORD 形式向发送 FIFO 写入数据。如果数据包大小不是 DWORD 的整数倍，则应用程序必须将数据包填充到 DWORD 的整数倍。OTG_HS 主机根据设定的最大数据包大小和传输大小确定实际数据包大小。

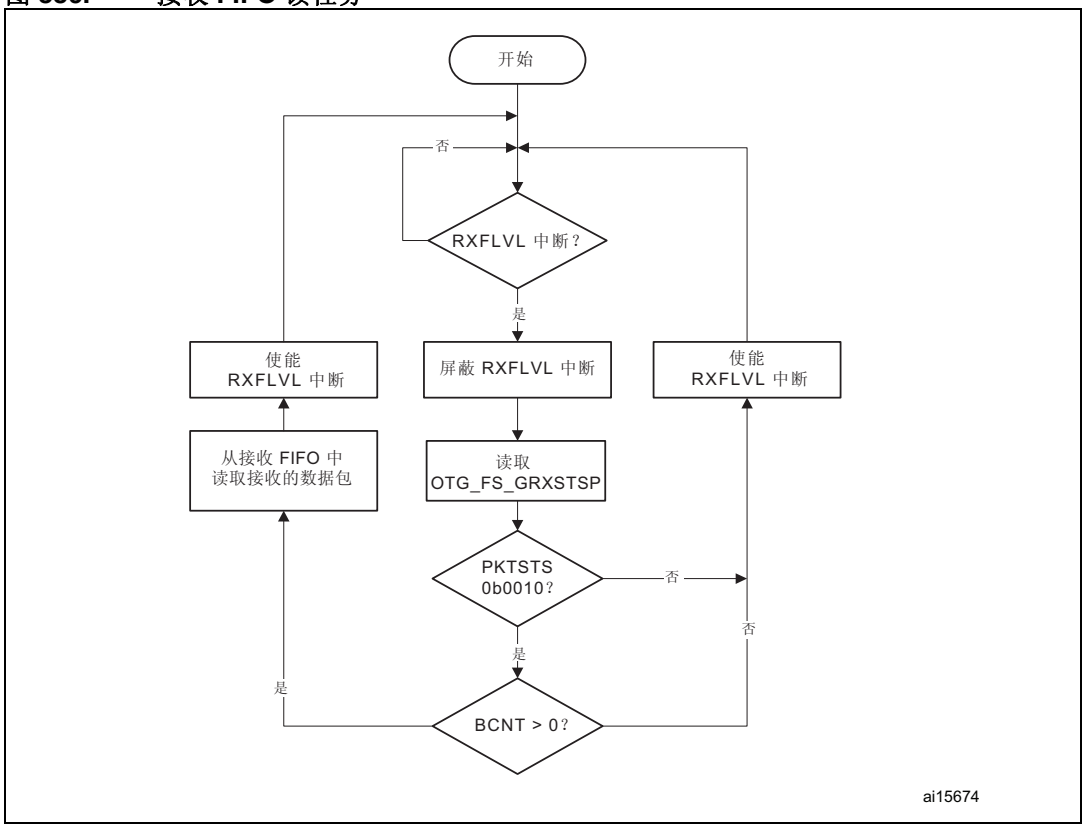
图 385. 发送 FIFO 写任务



- **读取接收 FIFO**

应用程序必须忽略除 IN 数据包 (bx0010) 以外的所有数据包状态。

图 386. 接收 FIFO 读任务



- **批量和控制传输类型的 OUT/SETUP 通信事务**

图 387 显示了典型的批量或控制 OUT/SETUP 流水线事务级操作。请参见通道 1 (ch_1)。该通道发送了两个批量 OUT 数据包。控制 SETUP 事务的工作方式相同，只不过只包含一个数据包。假设：

– 应用程序尝试发送两个最大数据包大小的数据包（传输大小 = 1024 字节）。
– 非周期性发送 FIFO 可存储两个数据包（FS 对应 128 字节）。
– 非周期性请求队列深度 = 4。

- **正常批量和控制传输类型的 OUT/SETUP 操作**

对于通道 1，操作顺序如下：

a) 初始化通道 1
b) 写入通道 1 的第一个数据包
c) 在应用执行最后一次 DWORD 写操作时，模块将向非周期性请求队列写入一个请求条目
d) 只要非周期性队列非空，模块即会尝试在当前帧内发送一个 OUT 令牌
e) 写入通道 1 的第二个（最后一个）数据包
f) 成功完成最后一个事务后，模块立即生成 XFRC 中断
g) 为了响应 XFRC 中断，将释放通道以供其它传输操作使用
h) 处理非 ACK 响应

图 387. 正常批量/控制 OUT/SETUP 和批量/控制 IN 事务 - DMA 模式

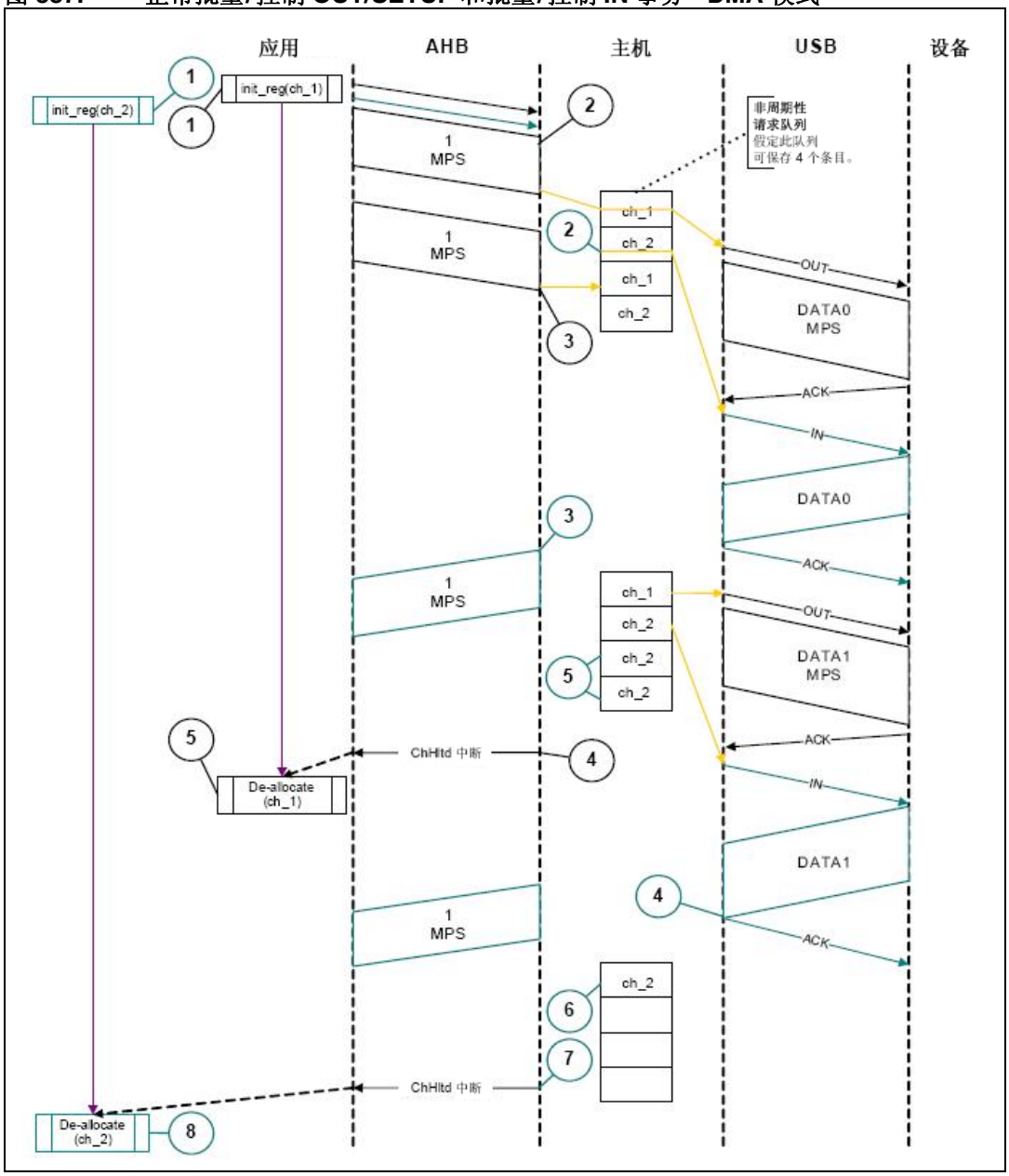
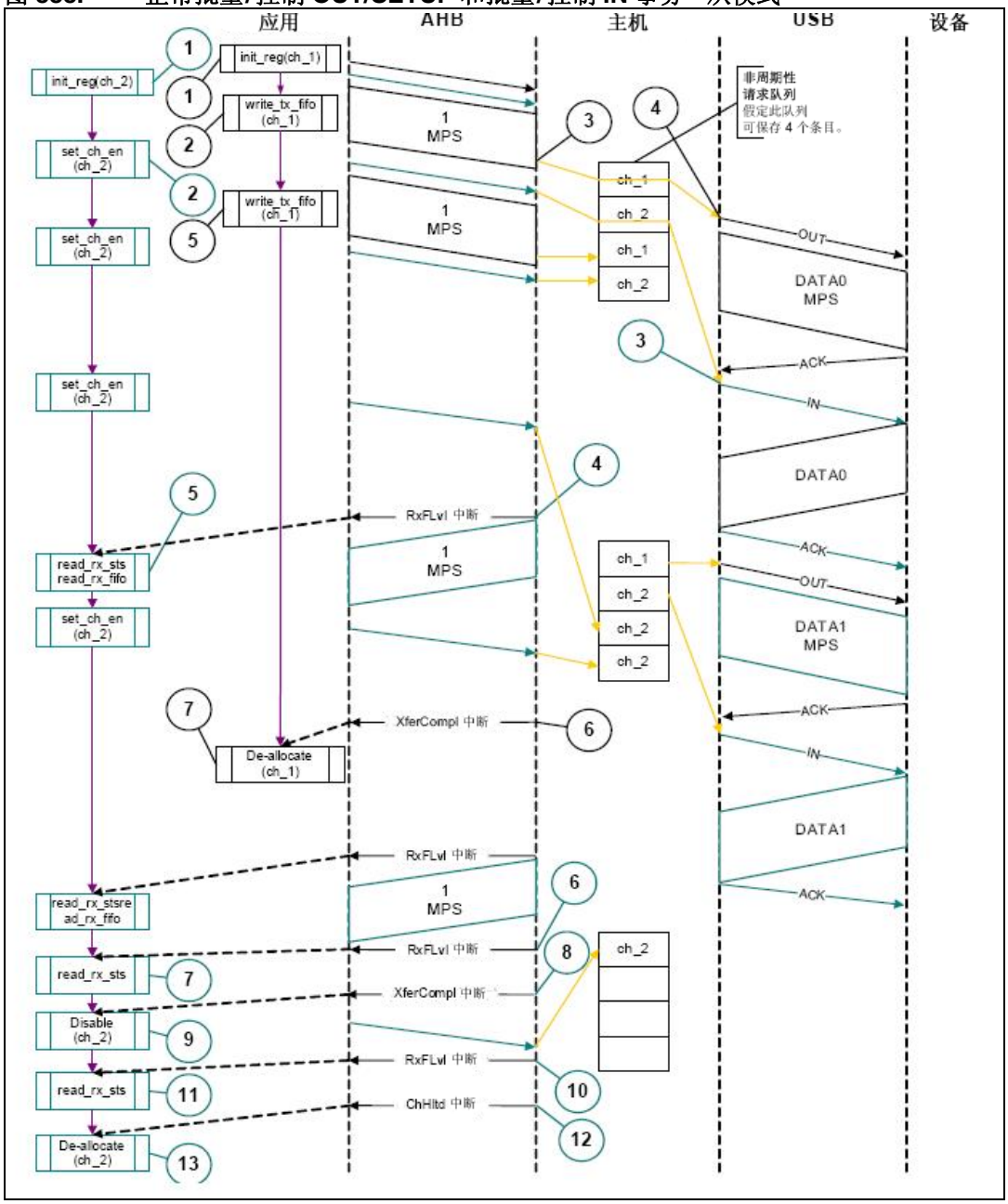


图 388. 正常批量/控制 OUT/SETUP 和批量/控制 IN 事务 - 从模式



以下代码示例说明了从模式中批量和控制传输类型 OUT/SETUP 事务的通道相关中断服务程序。

- **批量/控制 OUT/SETUP 和批量/控制 IN 事务的中断服务程序**

a) 批量/控制 OUT/SETUP

```
Unmask (NAK/TXERR/STALL/XFRC)
if (XFRC)
   {
   Reset Error Count
   Mask ACK
   De-allocate Channel
   }
else if (STALL)
   {
   Transfer Done = 1
   Unmask CHH
   Disable Channel
   }
else if (NAK or TXERR )
   {
   Rewind Buffer Pointers
   Unmask CHH
   Disable Channel
   if (TXERR)
      {
      Increment Error Count
      Unmask ACK
      }
   else
      {
      Reset Error Count
      }
      }
else if (CHH)
   {
   Mask CHH
   if (Transfer Done or (Error_count == 3))
      {
      De-allocate Channel
      }
   else
      {
      Re-initialize Channel
      }
   }
   else if (ACK)
      {
      Reset Error Count
      Mask ACK
      }
```

当发送 FIFO 和请求队列中有可用空间时，应用程序会将数据包写入发送 FIFO。应用程序可利用 OTG_HS_GINTSTS 中的 NPTXFE 中断确定发送 FIFO 空间。

b) 批量/控制 IN

```
Unmask (TXERR/XFRC/BBERR/STALL/DTERR)
if (XFRC)
   {
   Reset Error Count
   Unmask CHH
   Disable Channel
   Reset Error Count
   Mask ACK
   }
else if (TXERR or BBERR or STALL)
   {
   Unmask CHH
   Disable Channel
   if (TXERR)
      {
      Increment Error Count
      Unmask ACK
      }
   }
else if (CHH)
   {
   Mask CHH
   if (Transfer Done or (Error_count == 3))
      {
      De-allocate Channel
      }
   else
      {
      Re-initialize Channel
      }
   }
else if (ACK)
   {
   Reset Error Count
   Mask ACK
   }
else if (DTERR)
   {
   Reset Error Count
   }
```

当请求队列空间可用时，应用程序会写入请求，直到接收到 XFRC 中断。

- **批量和控制 IN 事务**

  图 389 显示了典型的批量或控制传输类型的 IN 流水线事务级操作。请参见通道 2 (ch_2)。假设：

  – 应用程序要接收两个最大数据包大小的数据包（传输大小 = 1024 字节）。

  – 接收 FIFO 可以包含至少一个最大数据包大小的数据包和每个数据包的两个状态 DWORD（FS 对应 72 字节）。

  – 非周期性请求队列深度 = 4。

图 389. 批量/控制 IN 事务 - DMA 模式

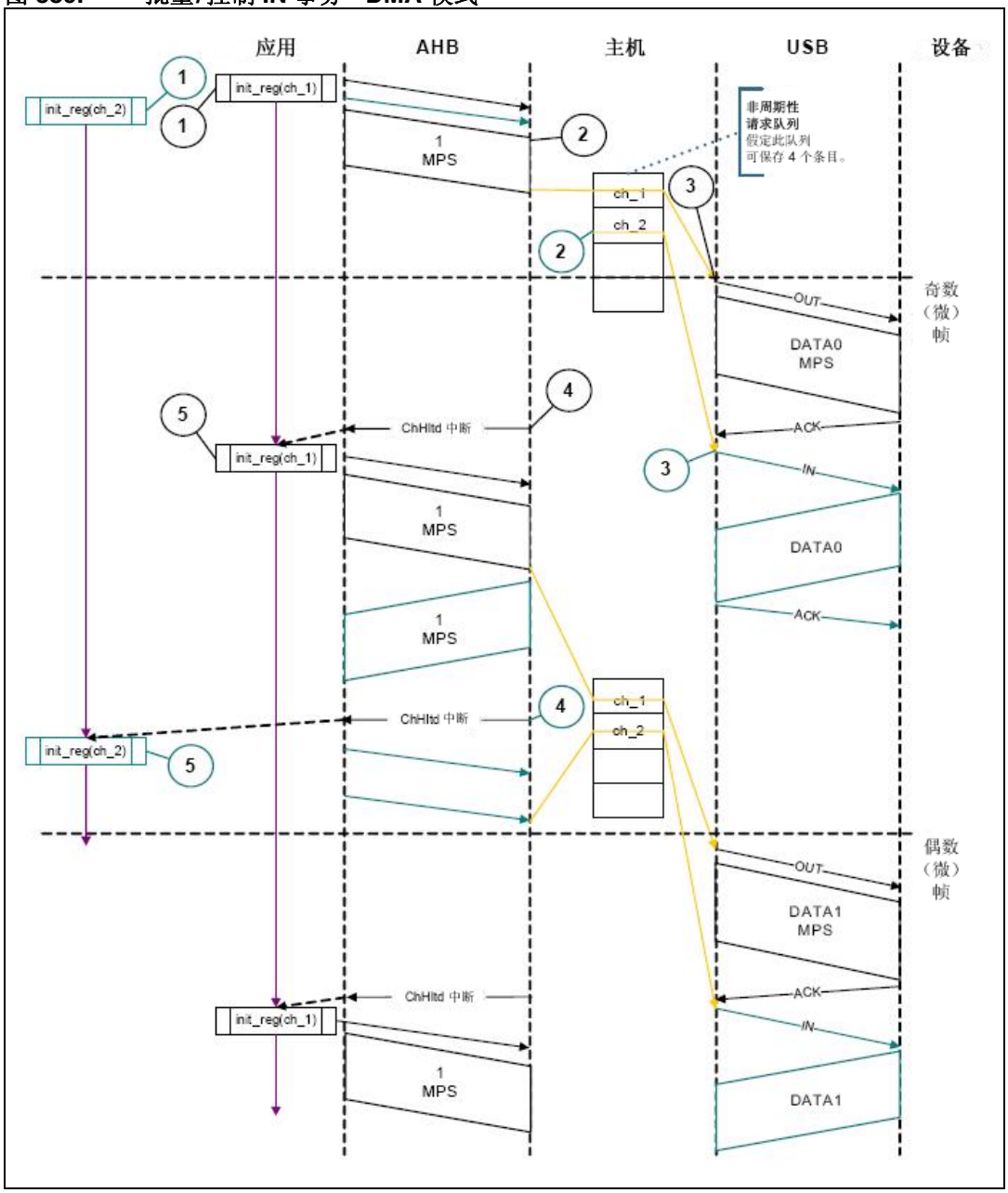
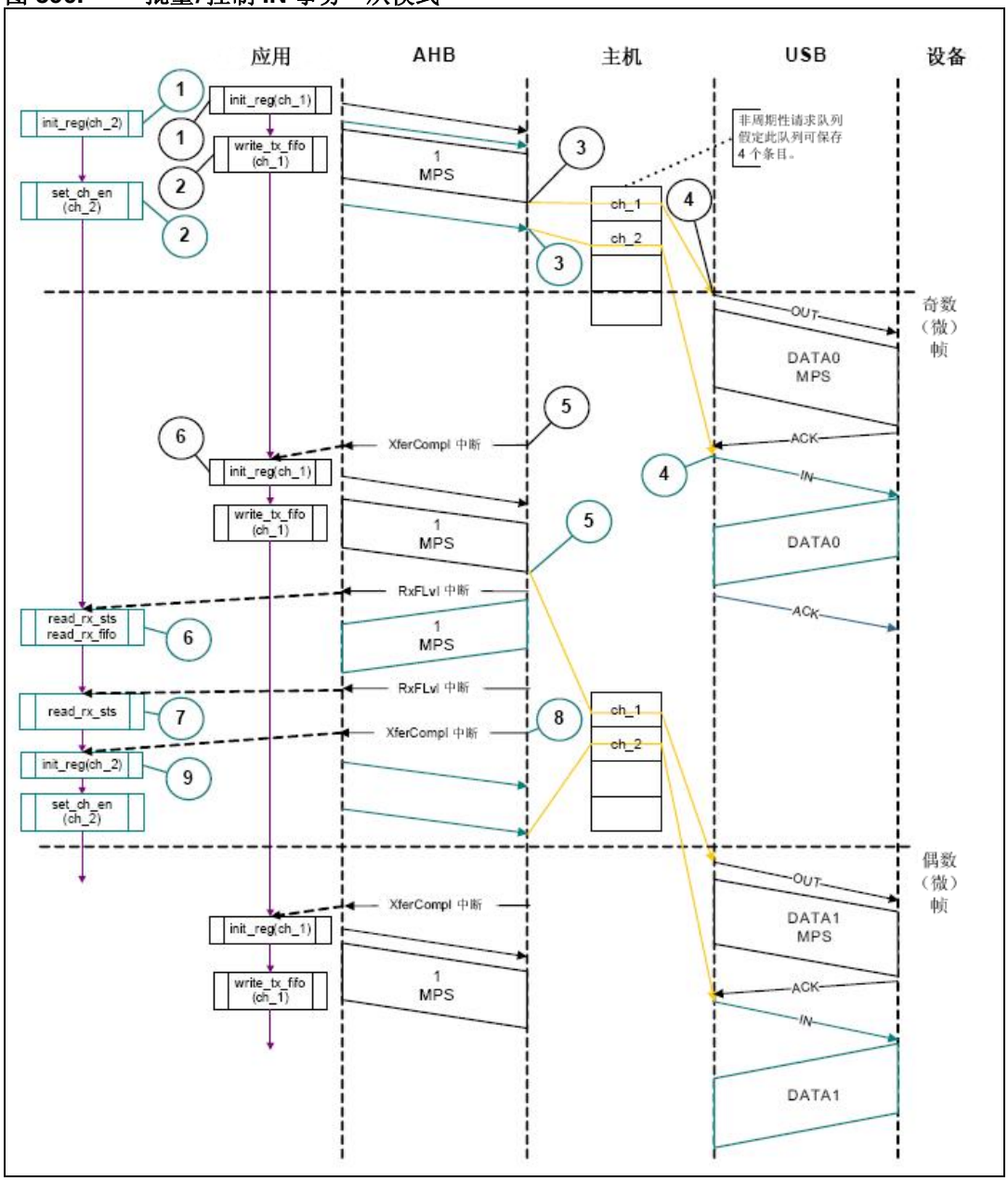


图 390. 批量/控制 IN 事务 - 从模式



操作顺序如下：

a) 初始化通道 2。

b) 将 HCCHAR2 中的 CHENA 位置 1，以向非周期性请求队列写入 IN 请求。

c) 模块尝试在完成当前 OUT 事务后发送 IN 令牌。

d) 接收到的数据包写入接收 FIFO 后，模块立即生成 RXFLVL 中断。

e) 为了响应 RXFLVL 中断，将屏蔽 RXFLVL 中断并读取接收到的数据包状态，以此确定接收的字节数，然后相应地读取接收 FIFO。之后取消对 RXFLVL 中断的屏蔽。

f) 模块针对接收 FIFO 中的传输完成状态条目生成 RXFLVL 中断。

g) 应用程序必须读取接收数据包状态，并在不是 IN 数据包（GRXSTSR 中的 PKTSTS ≠ 0b0010）时忽略它。

h) 读取接收数据包状态后，模块立即生成 XFRC 中断。

i) 为了响应 XFRC 中断，将禁止通道并停止针对其它请求向 OTG_HS_HCCHAR2 写入数据。向 OTG_HS_HCCHAR2 寄存器写入数据时，模块立即向非周期性请求队列写入通道禁止请求。

j) 停止状态写入接收 FIFO 后，模块立即生成 RXFLVL 中断。

k) 读取并忽略接收数据包状态。

l) 只要停止状态从接收 FIFO 弹出，模块立即生成 CHH 中断。

m) 为了响应 CHH 中断，将释放通道以供其它传输操作使用。

n) 处理非 ACK 响应。

- **从模式下的控制事务**

  控制传输的建立、数据和状态阶段必须作为三个独立的传输过程来执行。建立、数据和状态阶段的 OUT 事务与上文所述的批量 OUT 事务的执行方式类似。数据或状态阶段的 IN 事务与上文所述的批量 IN 事务的执行方式类似。在所有这三个阶段，应用程序都会将 OTG_HS_HCCHAR1 中的 EPTYP 字段设置为控制传输类型。在建立阶段期间，应用程序会将 OTG_HS_HCTSIZ1 中的 PID 字段设置为 SETUP。

- **中断 OUT 事务**

  图 391 显示了典型的从模式下的中断 OUT 操作。假设：

  – 应用程序尝试从奇数帧开始，每帧发送一个数据包（高达 1 个最大数据包大小）（传输大小 = 1024 字节）

  – 周期性发送 FIFO 可存储一个数据包 (1 KB)

  – 周期性请求队列深度 = 4

  操作顺序如下：

  a) 初始化和使能通道 1。应用程序必须将 OTG_HS_HCCHAR1 中的 ODDFRM 位置 1。

  b) 写入通道 1 的第一个数据包。对于高带宽中断传输，应用程序必须写入后续数据包，直至达到 MCNT 个数据包（下一帧要发送的最大数据包个数），然后才能切换到其它通道。

  c) 应用程序在执行每个数据包的最后一次 DWORD 写操作时，OTG_HS 主机向周期性请求队列写入一个请求条目。

  d) OTG_HS 主机尝试在下一（奇数）帧发送 OUT 令牌。

  e) 成功发送最后一个数据包后，OTG_HS 主机立即生成 XFRC 中断。

  f) 为了响应 XFRC 中断，将重新初始化通道以供下一传输操作使用。

图 391. 正常中断 OUT/IN 事务 - DMA 模式

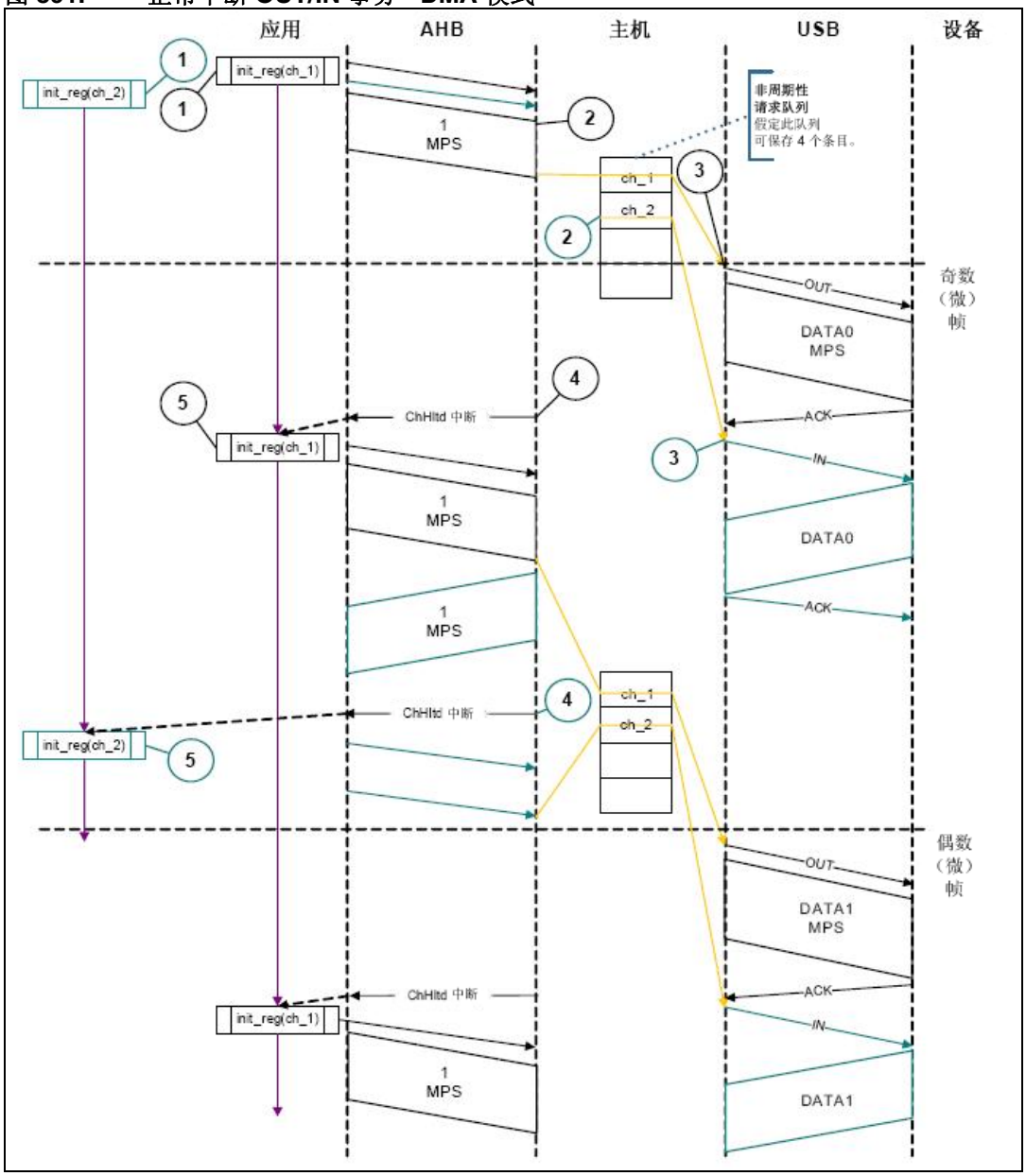
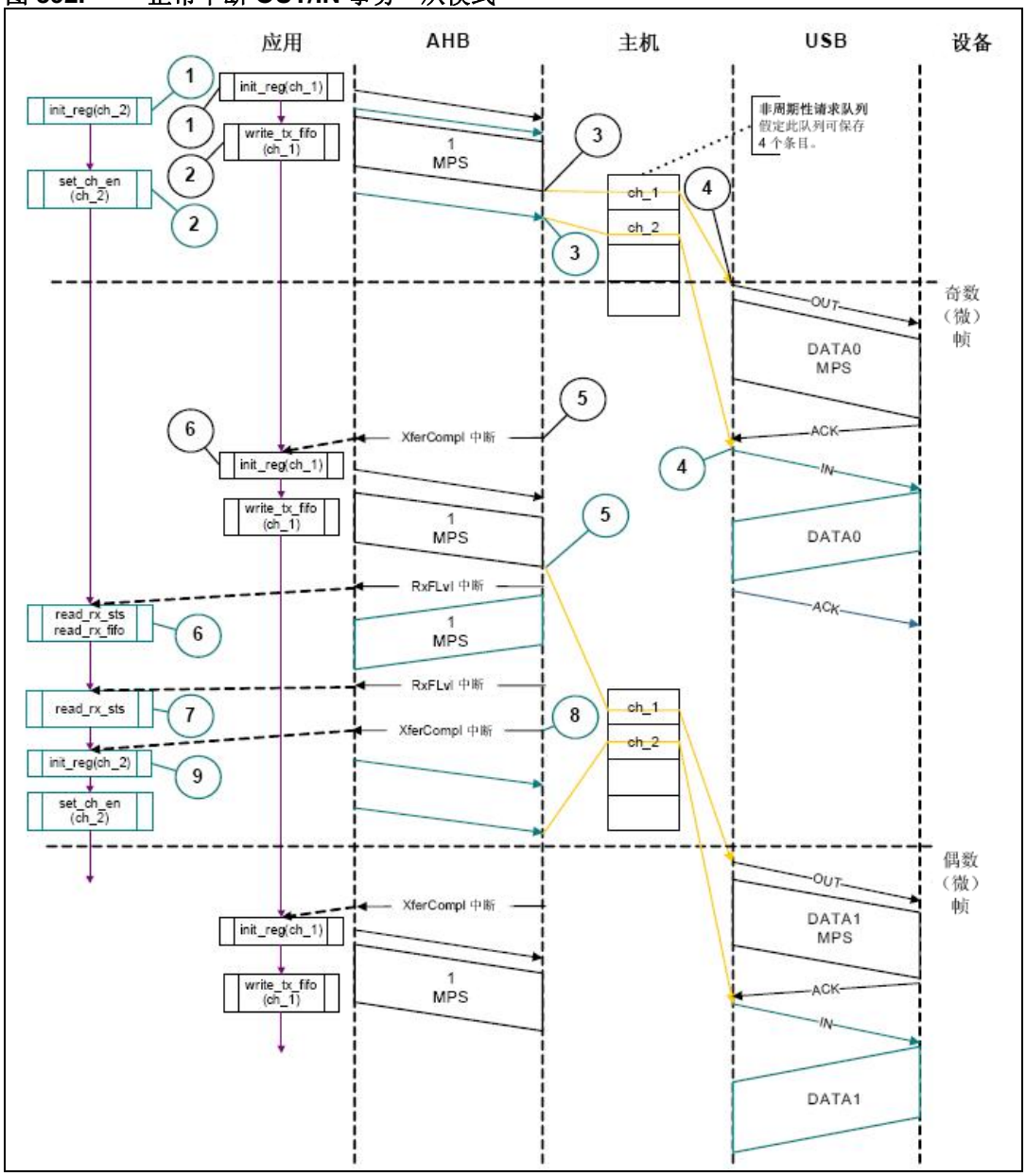


图 392. 正常中断 OUT/IN 事务 - 从模式



- **中断 OUT/IN 事务的中断服务程序**

a) 中断 OUT

```
Unmask (NAK/TXERR/STALL/XFRC/FRMOR)
if (XFRC)
  {
  Reset Error Count
  Mask ACK
  De-allocate Channel
  }
else
  if (STALL or FRMOR)
    {
    Mask ACK
    Unmask CHH
    Disable Channel
    if (STALL)
      {
      Transfer Done = 1
      }
    }
  else
    if (NAK or TXERR)
      {
      Rewind Buffer Pointers
      Reset Error Count
      Mask ACK
      Unmask CHH
      Disable Channel
      }
    else
      if (CHH)
      {
      Mask CHH
      if (Transfer Done or (Error_count == 3))
        {
        De-allocate Channel
        }
      else
        {
        Re-initialize Channel (in next b_interval - 1 Frame)
        }
      }
    else
      if (ACK)
      {
      Reset Error Count
      Mask ACK
      }
```

当发送 FIFO 和请求队列中有可用空间时，应用程序会将数据包写入发送 FIFO，直至达到 MCNT 字段中指定的计数值，然后才能切换到其它通道。应用程序利用 OTG_HS_GINTSTS 中的 NPTXFE 中断确定发送 FIFO 空间。

b) 中断 IN

```
Unmask (NAK/TXERR/XFRC/BBERR/STALL/FRMOR/DTERR)
if (XFRC)
   {
   Reset Error Count
   Mask ACK
   if (OTG_HS_HCTSIZx.PKTCNT == 0)
      {
      De-allocate Channel
      }
   else
      {
      Transfer Done = 1
      Unmask CHH
      Disable Channel
      }
   }
else
   if (STALL or FRMOR or NAK or DTERR or BBERR)
      {
      Mask ACK
      Unmask CHH
      Disable Channel
      if (STALL or BBERR)
         {
         Reset Error Count
         Transfer Done = 1
         }
      else
         if (!FRMOR)
         {
         Reset Error Count
         }
      }
else
   if (TXERR)
      {
      Increment Error Count
      Unmask ACK
      Unmask CHH
      Disable Channel
      }
else
   if (CHH)
      {
      Mask CHH
      if (Transfer Done or (Error_count == 3))
         {
         De-allocate Channel
```

```
        }
      else
        Re-initialize Channel (in next b_interval - 1 /Frame)
      }
   }
else
   if (ACK)
      {
      Reset Error Count
      Mask ACK
      }
```

当请求队列有可用空间时，应用程序会为同一个通道写入请求，直至达到 MCNT 字段中指定的计数值，然后才能切换到其它通道（如果存在）。

- **中断 IN 事务**

  假设：

  – 应用程序要从奇数帧开始，每帧接收一个数据包（最大 1 个最大数据包大小）（传输大小 = 1024 字节）。

  – 接收 FIFO 可以保存至少一个最大数据包大小的数据包和每个数据包的两个状态 DWORD（1031 字节）。

  – 周期性请求队列深度 = 4。

- **正常中断 IN 操作**

  操作顺序如下：

  a) 初始化通道 2。应用程序必须将 OTG_HS_HCCHAR2 中的 ODDFRM 位置 1。

  b) 将 OTG_HS_HCCHAR2 中的 CHENA 位置 1，以将 IN 请求写入周期性请求队列。对于中断传输，应用程序必须对 OTG_HS_HCCHAR2 寄存器执行 MCNT 次写操作（下一帧要发送的数据包的最大个数），然后才能切换到其它通道。

  c) 只要 CHENA 置位，对于每次 OTG_HS_HCCHAR2 寄存器写操作，OTG_HS 主机都会将一个 IN 请求写入周期性请求队列。

  d) OTG_HS 主机尝试在下一（奇数）帧发送 IN 令牌。

  e) 接收到 IN 数据包并写入接收 FIFO 后，OTG_HS 主机便会立即生成 RXFLVL 中断。

  f) 为响应 RXFLVL 中断，读取接收到的数据包状态以确定接收的字节数，然后相应地读取接收 FIFO。应用程序必须在读取接收 FIFO 前屏蔽 RXFLVL 中断，并且在读取整个数据包后取消屏蔽。

  g) 模块针对接收 FIFO 中的传输完成状态条目生成 RXFLVL 中断。应用程序必须读取接收数据包状态，并在不是 IN 数据包（GRXSTSR 中的 PKTSTS ≠ 0b0010）时忽略它。

  h) 读取接收数据包状态后，模块立即生成 XFRC 中断。

  i) 为响应 XFRC 中断，将读取 OTG_HS_HCTSIZ2 中的 PKTCNT 字段。如果 OTG_HS_HCTSIZ2 中的 PKTCNT 位不等于 0，则在重新初始化通道以进行下次传输前（如果存在），禁止该通道。如果 OTG_HS_HCTSIZ2 中的 PKTCNT 位等于 0，则重新初始化通道以进行下次传输。此时，应用程序必须复位 OTG_HS_HCCHAR2 中的 ODDFRM 位。

- **同步 OUT 事务**

  图 393 显示了典型的从模式下的同步 OUT 操作。假设：

  – 应用程序尝试从奇数帧开始，每帧发送一个数据包（最大 1 个最大数据包大小）。（传输大小 = 1024 字节）。

  – 周期性发送 FIFO 可存储一个数据包 (1 KB)。

  – 周期性请求队列深度 = 4。

  操作顺序如下：

  a) 初始化和使能通道 1。应用程序必须将 OTG_HS_HCCHAR1 中的 ODDFRM 位置 1。

  b) 写入通道 1 的第一个数据包。对于高带宽同步传输，应用程序必须写入后续数据包，直至达到 MCNT 个数据包（下一帧要发送的数据包的最大个数），然后才能切换到其它通道。

  c) 应用程序在执行每个数据包的最后一次 DWORD 写操作时，OTG_HS 主机向周期性请求队列写入一个请求条目。

  d) OTG_HS 主机尝试在下一（奇数）帧发送 OUT 令牌。

  e) 成功发送最后一个数据包后，OTG_HS 主机立即生成 XFRC 中断。

  f) 为了响应 XFRC 中断，将重新初始化通道以供下一传输操作使用。

  g) 处理非 ACK 响应。

图 393. 正常同步 OUT/IN 事务 - DMA 模式

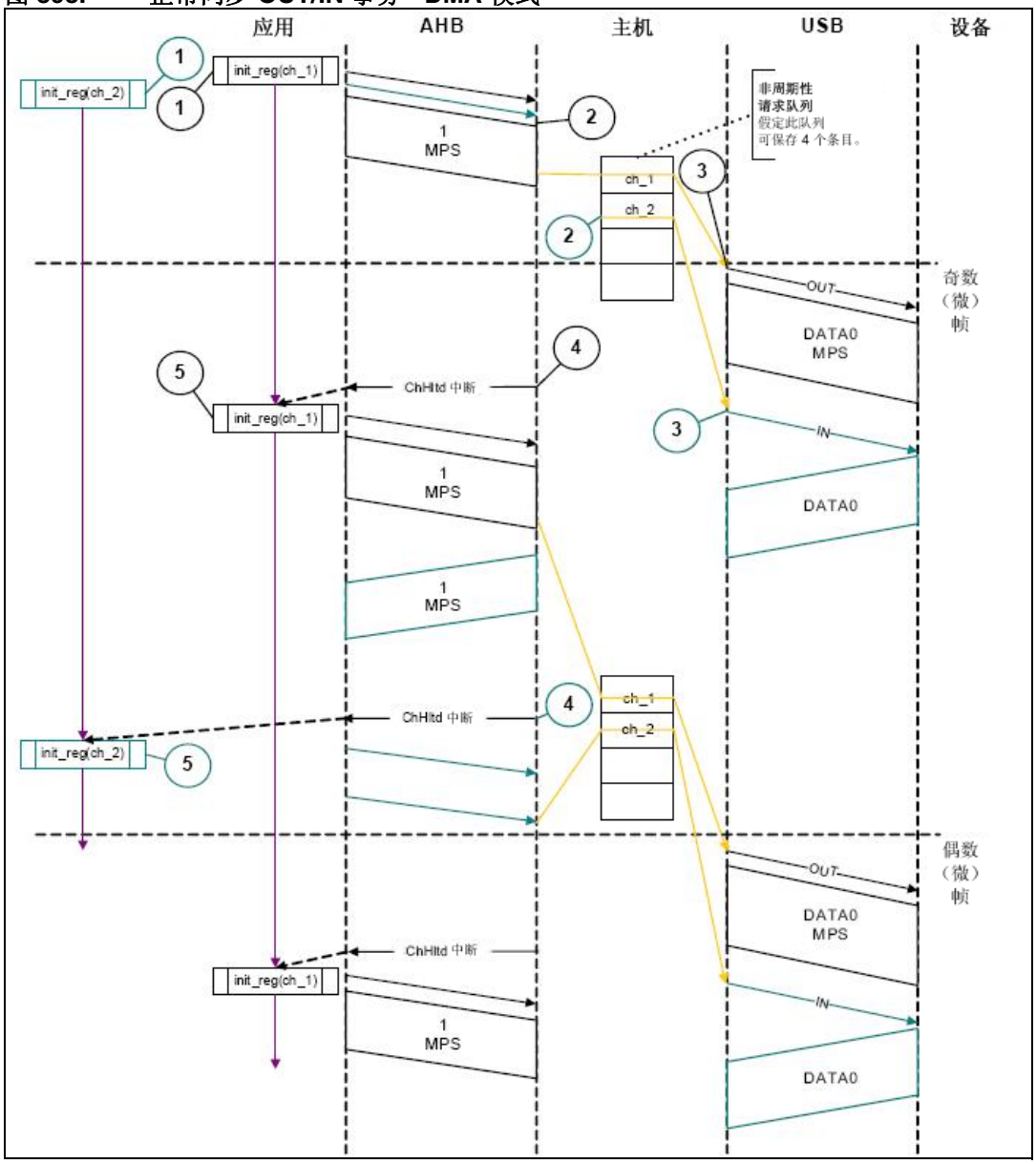
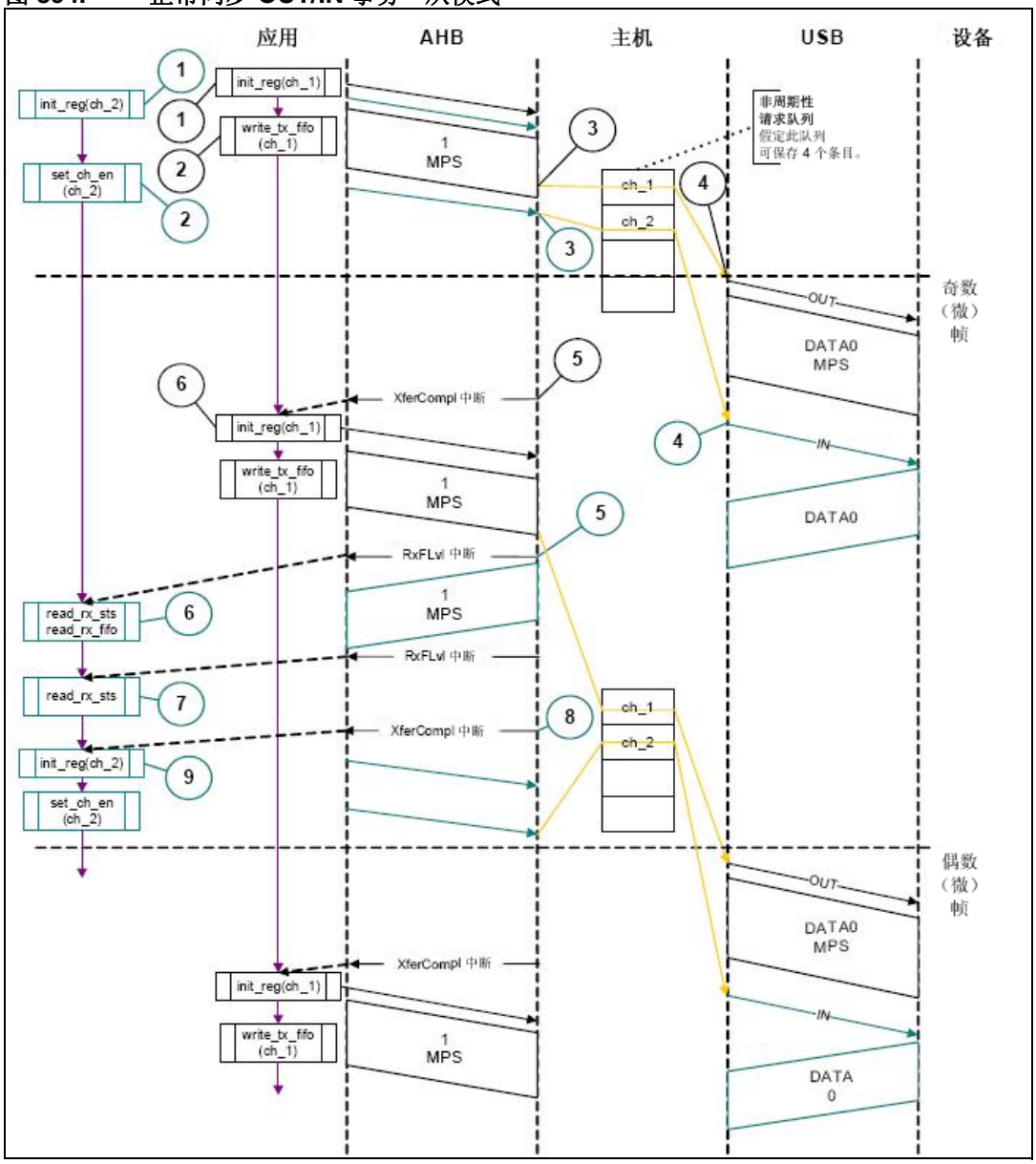


图 394. 正常同步 OUT/IN 事务 - 从模式



- **同步 OUT/IN 事务的中断服务程序**

代码示例：同步 OUT

```
Unmask (FRMOR/XFRC)
if (XFRC)
   {
   De-allocate Channel
   }
else
   if (FRMOR)
      {
      Unmask CHH
      Disable Channel
      }
```

```
   else
   if (CHH)
      {
      Mask CHH
      De-allocate Channel
      }
```

代码示例：Isochronous IN

```
Unmask (TXERR/XFRC/FRMOR/BBERR)
if (XFRC or FRMOR)
   {
   if (XFRC and (OTG_HS_HCTSIZx.PKTCNT == 0))
      {
      Reset Error Count
      De-allocate Channel
      }
   else
      {
      Unmask CHH
      Disable Channel
      }
   }
   else
      if (TXERR or BBERR)
         {
         Increment Error Count
         Unmask CHH
         Disable Channel
         }
      else
         if (CHH)
            {
            Mask CHH
            if (Transfer Done or (Error_count == 3))
               {
               De-allocate Channel
               }
            else
               {
               Re-initialize Channel
               }
            }
```

- **同步 IN 事务**

  假设：

  – 应用程序尝试从下一个奇数帧开始，每帧接收一个数据包（最大 1 个最大数据包大小）（传输大小 = 1024 字节）。

  – 接收 FIFO 可以保存至少一个最大数据包大小的数据包和每个数据包的两个状态 DWORD（1031 字节）。

  – 周期性请求队列深度 = 4。

  操作顺序如下：

  a) 初始化通道 2。应用程序必须将 OTG_HS_HCCHAR2 中的 ODDFRM 位置 1。

  b) 将 OTG_HS_HCCHAR2 中的 CHENA 位置 1，以将 IN 请求写入周期性请求队列。对于高带宽同步传输，应用程序必须对 OTG_HS_HCCHAR2 寄存器执行 MCNT 次写操作（下一帧要发送的数据包的最大个数），然后才能切换到其它通道。

  c) 只要 CHENA 置位，对于每次 OTG_HS_HCCHAR2 寄存器写操作，OTG_HS 主机都会将一个 IN 请求写入周期性请求队列。

  d) OTG_HS 主机尝试在下一奇数帧发送 IN 令牌。

  e) 接收到 IN 数据包并写入接收 FIFO 后，OTG_HS 主机便会立即生成 RXFLVL 中断。

  f) 为响应 RXFLVL 中断，读取接收到的数据包状态以确定接收的字节数，然后相应地读取接收 FIFO。应用程序必须在读取接收 FIFO 前屏蔽 RXFLVL 中断，并且在读取整个数据包后取消屏蔽。

  g) 模块针对接收 FIFO 中的传输完成状态条目生成 RXFLVL 中断。应用程序必须读取接收数据包状态，并在不是 IN 数据包（OTG_HS_GRXSTSR 中的 PKTSTS 位 ≠ 0b0010）时忽略它。

  h) 读取接收数据包状态后，模块立即生成 XFRC 中断。

  i) 为响应 XFRC 中断，将读取 OTG_HS_HCTSIZ2 中的 PKTCNT 字段。如果在 OTG_HS_HCTSIZ2 中的 PKTCNT≠ 0，则在重新初始化通道以进行下次传输前（如果存在），禁止该通道。如果 OTG_HS_HCTSIZ2 中的 PKTCNT = 0，则重新初始化通道以进行下次传输。此时，应用程序必须复位 OTG_HS_HCCHAR2 中的 ODDFRM 位。

- **选择队列深度**

  请谨慎选择周期性和非周期性请求队列深度，以与要访问的周期性/非周期性端点数量相匹配。

  非周期性请求队列深度会影响非周期性传输的性能。队列越深（加上足够的 FIFO 大小），模块就更能对非周期性传输进行流水线处理。如果队列大小太小，则仅在队列空间释放时模块才能放入新请求。

  要按调度计划执行周期性传输，模块的周期性请求队列深度至关重要。根据一个微帧内要安排的周期性传输次数，选择周期性队列深度。但是在从模式下，应用程序还需要将必须放入队列中的禁止条目考虑在内。因此，如果存在两个非高带宽周期性端点，周期性请求队列深度必须至少为 4。如果至少支持一个高带宽端点，则队列深度必须为 8。如果周期性请求队列深度小于一个微帧内要安排的周期性传输数，则会发生帧溢出情况。

- **处理 babble 情况**

  OTG_HS 控制器处理两种情况的 babble：数据包 babble 和端口 babble。如果设备发送的数据量超过通道的最大数据包大小，则会发生数据包 babble。如果模块从 EOF2（非常接近下一帧的 SOF）时刻还在从设备接收数据，则发生端口 babble。

  当 OTG_HS 控制器检测到数据包 babble 时，它停止向 Rx 缓冲区中写入数据并等待数据包结束信号 (EOP)。当该控制器检测到 EOP 时，它会清空 Rx 缓冲区中的已写入数据并对应用程序生成 Babble 中断。

当 OTG_HS 控制器检测到端口 babble 时，它会清空 RxFIFO 并禁止端口。模块随后将生成“端口已禁止”中断（OTG_HS_GINTSTS 中的 HPRTINT 位和 OTG_HS_HPRT 中的 PENCHNG 位）。在收到该中断时，应用程序必须通过检查 OTG_HS_HPRT 中的 POCA 位，来确定该中断并非由过流（“端口已禁止”中断的另一个原因）所致，然后执行软复位。检测到端口 babble 情况后，模块不再发送任何其它令牌。

- **DMA 模式下的批量和控制 OUT/SETUP 事务**

  操作顺序如下：

  a) 按照*通道初始化*一节中的说明初始化并使能通道 1。

  b) 使能通道后，HS_OTG 主机即开始获取第一个数据包以发送。对于内部 DMA 模式，OTG_HS 主机使用经过编程的 DMA 地址来获取数据包。

  c) 获取第二个（最后一个）数据包的最后一个 DWORD 后，OTG_HS 主机会在内部屏蔽通道 1，不参与仲裁。

  d) 最后一个数据包发送出去后，HS_OTG 主机立即生成一个 CHH 中断。

  e) 为了响应 CHH 中断，将释放通道以供其它传输操作使用。

- **使用内部 DMA 处理 NAK 和 NYET**

  a) OTG_HS 主机发送批量传输类型的 OUT 事务。

  b) 设备回复 NAK 或 NYET。

  c) 如果应用程序没有屏蔽 NAK 或 NYET 中断，模块将为应用程序生成相应的中断。应用程序并不需要处理这些中断，因为模块会负责调整缓冲区指针和重新初始化通道，而无需应用程序干预。

  d) 模块自动发出一个 ping 令牌。

  e) 当设备返回 ACK 后，模块会继续传输。应用程序也可以选择使用这些中断（在这种情况下，NAK 或 NYET 中断将被应用程序屏蔽）。

  当主机接收到 NAK 或 NYET 时，模块不会生成单独的中断。

- **DMA 模式下的批量和控制 IN 事务**

  操作顺序如下：

  a) 按照*通道初始化*一节中的说明初始化并使能所使用的通道（通道 x）。

  b) 当通道接收到来自仲裁器的授权后（以循环方式执行仲裁），OTG_HS 会立即向请求队列写入一个 IN 请求。

  c) 当正确接收到最后一个字节后，OTG_HS 主机便开始将接收到的数据写入到系统存储器中。

  d) 当接收到最后一个数据包后，OTG_HS 会将一个内部标志置 1，以便从请求队列中删除任何额外的 IN 请求。

  e) OTG_HS 主机会清空额外请求。

  f) 向请求队列写入最后一个禁止通道的请求。此时，会在内部屏蔽通道 2，不参与仲裁。

  g) 当禁止请求到达队列顶端时，OTG_HS 主机即会生成 CHH 中断。

  h) 为了响应 CHH 中断，将释放通道以供其它传输操作使用。

- **DMA 模式下的中断 OUT 事务**

  a) 按照*通道初始化*一节中的说明初始化并使能通道 x。

  b) 使能通道后，OTG_HS 主机开始获取第一个数据包以发送，并在获取数据包的最后一个后写入 DWORD 写入 OUT 请求。在高带宽传输模式下，HS_OTG 主机会继续获取下一个数据包（直至数据包数量达到 MC 字段中指定的数值），然后才能切换到下一个通道。

c) OTG_HS 主机尝试在下一个奇数帧/微帧的起始位置发送 OUT 令牌。
d) 成功发送数据包后，OTG_HS 主机将生成一个 CHH 中断。
e) 为了响应 CHH 中断，将重新初始化通道以供下一传输操作使用。

- **DMA 模式下的中断 IN 事务**

通道 x 的操作顺序如下：

a) 按照*通道初始化*一节中的说明初始化并使能通道 x。
b) 当通道获取了来自仲裁器的授权后（以循环方式执行仲裁），OTG_HS 主机会立即向请求队列写入一个 IN 请求。在高带宽传输模式下，OTG_HS 主机将执行连续写入操作，直到写入的数据包个数达到 MC。
c) OTG_HS 主机尝试在下一个（奇数）帧/微帧的起始位置发送 IN 令牌。
d) 接收到 IN 数据包并写入接收 FIFO 后，OTG_HS 主机便会立即生成 CHH 中断。
e) 为了响应 CHH 中断，将重新初始化通道以供下一传输操作使用。

- **DMA 模式下的同步 OUT 事务**

a) 按照*通道初始化*一节中的说明初始化并使能通道 x。
b) 使能通道后，OTG_HS 主机开始获取第一个数据包，并在获取数据包的最后一个 DWORD 后写入 OUT 请求。在高带宽传输模式下，OTG_HS 主机会继续获取下一个数据包（直到写入的数据包个数达到 MC），然后才能切换到下一个通道。
c) OTG_HS 主机尝试在下一个（奇数）帧/微帧的起始位置发送 OUT 令牌。
d) 成功发送数据包后，HS_OTG 主机将生成一个 CHH 中断。
e) 为了响应 CHH 中断，将重新初始化通道以供下一传输操作使用。

- **DMA 模式下的同步 IN 事务**

通道 x 的操作顺序如下：

a) 按照*通道初始化*一节中的说明初始化并使能通道 x。
b) 当通道获取了来自仲裁器的授权后（以循环方式执行仲裁），OTG_HS 主机会立即向请求队列写入一个 IN 请求。在高带宽传输模式下，OTG_HS 主机将执行连续写入操作，直到写入的数据包个数达到 MC。
c) OTG_HS 主机尝试在下一个（奇数）帧/微帧的起始位置发送 IN 令牌。
d) 接收到 IN 数据包并写入接收 FIFO 后，OTG_HS 主机便会立即生成 CHH 中断。
e) 为了响应 CHH 中断，将重新初始化通道以供下一传输操作使用。

- **DMA 模式下的批量和控制 OUT/SETUP 分离事务**

（通道 x）中的操作顺序如下：

a) 按照*通道初始化*一节中的说明初始化并使能通道 x 以用于启动分离。
b) 使能通道后，OTG_HS 主机开始获取第一个数据包以发送，并在获取数据包的最后一个 DWORD 后写入 OUT 请求。
c) 成功发送启动分离后，OTG_HS 主机将生成 CHH 中断。
d) 作为对 CHH 中断的响应，将 HCSPLT1 中的 COMPLSPLT 位置 1，以发送完成分离。
e) 成功发送完成分离后，OTG_HS 主机将生成 CHH 中断。
f) 为了响应 CHH 中断，将会释放通道。

- **DMA 模式下的批量/控制 IN 分离事务**

通道 x 的操作顺序如下：

a) 按照*通道初始化*一节中的说明初始化并使能通道 x。
b) 获取了来自仲裁器的授权后，OTG_HS 主机将会向非周期性请求队列写入启动分离请求。写入该请求后，OTG_HS 主机将在内部屏蔽通道 x，不参与仲裁。

c) 发送 IN 令牌后，OTG_HS 主机立即生成 CHH 中断。
d) 作为对 CHH 中断的响应，将 HCSPLT2 中的 COMPLSPLT 位置 1 并重新使能通道，以发送完成分离令牌。这将取消对通道 x 的屏蔽，并让它参与后续仲裁。
e) 接收到来自仲裁器的授权后，OTG_HS 主机将会向非周期性请求队列写入完成分离请求。
f) 成功接收到数据包后，OTG_HS 主机将开始向系统存储器写入数据包。
g) 只要接收到的数据包写入系统存储器，OTG_HS 主机即会生成一个 CHH 中断。
h) 为了响应 CHH 中断，将会释放通道。

- **DMA 模式下的中断 OUT 分离事务**

（通道 x）中的操作顺序如下：

a) 按照*通道初始化*一节中的说明初始化并使能通道 1 以用于启动分离。应用程序必须将 HCCHAR1 中的 ODDFRM 位置 1。
b) HS_OTG 主机开始读取数据包。
c) HS_OTG 主机尝试发送启动分离事务。
d) 成功发送启动分离后，OTG_HS 主机将生成 CHH 中断。
e) 作为对 CHH 中断的响应，将 HCSPLT1 中的 COMPLSPLT 位置 1，以发送完成分离。
f) 成功结束完成分离事务后，OTG_HS 主机将生成 CHH 中断。
g) 为了响应 CHH 中断，将会释放通道。

- **DMA 模式下的中断 IN 分离事务**

（通道 x）中的操作顺序如下：

a) 按照*通道初始化*一节中的说明初始化并使能通道 x 以用于启动分离。
b) 当通道 x 接收到来自仲裁器的授权后，OTG_HS 主机会立即向请求队列写入一个 IN 请求。
c) OTG_HS 主机尝试在下一个奇数微帧的起始位置发送启动分离 IN 令牌。
d) 成功发送启动分离 IN 令牌后，OTG_HS 主机将生成 CHH 中断。
e) 作为对 CHH 中断的响应，将 HCSPLT2 中的 COMPLSPLT 位置 1，以发送完成分离。
f) 成功接收到数据包后，OTG_HS 主机即会开始向系统存储器写入数据。
g) 将接收到的数据传输到系统存储器后，OTG_HS 主机会生成 CHH 中断。
h) 为了响应 CHH 中断，将释放或重新初始化通道以供下一启动分离使用。

- **DMA 模式下的同步 OUT 分离事务**

通道 x 的操作顺序如下：

a) 按照*通道初始化*一节中的说明初始化并使能通道 x 以用于启动分离（开始）。应用程序必须将 HCCHAR1 中的 ODDFRM 位置 1。编程 MPS 字段。
b) HS_OTG 主机开始读取数据包。
c) 成功发送启动分离后（开始），HS_OTG 主机将生成 CHH 中断。
d) 为了响应 CHH 中断，将重新初始化寄存器以发送启动分离（结束）。
e) 成功发送启动分离后（结束），OTG_HS 主机将生成 CHH 中断。
f) 为了响应 CHH 中断，将会释放通道。

- **DMA 模式下的同步 IN 分离事务**

通道 x 的操作顺序如下：

a) 按照*通道初始化*一节中的说明初始化并使能通道 x 以用于启动分离。
b) 当通道 x 接收到来自仲裁器的授权后，OTG_HS 主机会立即向请求队列写入一个 IN 请求。

c) OTG_HS 主机尝试在下一个奇数微帧的起始位置发送启动分离 IN 令牌。
d) 成功发送启动分离 IN 令牌后，OTG_HS 主机将生成 CHH 中断。
e) 作为对 CHH 中断的响应，将 HCSPLT2 中的 COMPLSPLT 位置 1，以发送完成分离。
f) 成功接收到数据包后，OTG_HS 主机即会开始向系统存储器写入数据。
g) 将接收到的数据传输到系统存储器后，OTG_HS 主机会生成 CHH 中断。为了响应 CHH 中断，将释放或重新初始化通道以供下一启动分离使用。

31.13.6 设备编程模型

USB 复位时的端点初始化

1. 为所有 OUT 端点将 NAK 位置 1
   – OTG_HS_DOEPCTLx 中，SNAK = 1（对于所有 OUT 端点）
2. 取消对以下中断位的屏蔽
   – OTG_HS_DAINTMSK 中，INEP0 = 1（控制 0 IN 端点）
   – OTG_HS_DAINTMSK 中，OUTEP0 = 1（控制 0 OUT 端点）
   – DOEPMSK 中，STUP = 1
   – DOEPMSK 中，XFRC = 1
   – DIEPMSK 中，XFRC = 1
   – DIEPMSK 中，TOC = 1
3. 为每个 FIFO 设置数据 FIFO RAM
   – 对 OTG_HS_GRXFSIZ 寄存器进行编程，以能够接收控制传输的 OUT 数据和设置数据。如果未使能阈值，则该寄存器必须至少等于控制端点 0 的 1 个最大数据包大小 + 2 个 DWORD（用于控制 OUT 数据包的状态）+ 10 个 DWORD（用于 SETUP 数据包）。
   – 对 OTG_HS_TX0FSIZ 寄存器进行编程（取决于所选的 FIFO 编号），以能够发送控制 IN 数据。该寄存器至少必须等于控制端点 0 的 1 个最大数据包大小。
4. 对端点相关寄存器中的以下字段进行编程，以使控制 OUT 端点 0 接收 SETUP 数据包
   – OTG_HS_DOEPTSIZ0 中的 STUPCNT = 3（接收最多 3 个连续的 SETUP 数据包）
5. 在 DMA 模式下，DOEPDMA0 寄存器应具有一个有效的存储器地址，以存储接收到的任何 SETUP 数据包。

此时，接收 SETUP 数据包所需的所有初始化工作便已完成。

枚举完成时的端点初始化

1. 在枚举完成中断（OTG_HS_GINTSTS 中的 ENUMDNE）中，读取 OTG_HS_DSTS 寄存器以确定设备的枚举速度。
2. 对 OTG_HS_DIEPCTL0 中的 MPSIZ 字段进行编程以设置最大数据包大小。该步骤配置控制端点 0。控制端点的最大数据包大小取决于枚举速度。
3. 在 DMA 模式下，编程 DOEPCTL0 寄存器来使能控制 OUT 端点 0，以接收 SETUP 数据包。
   – 将 DOEPCTL0 中的 EPENA 位置 1

此时，设备已准备好接收 SOF 数据包并配置为在控制端点 0 上执行控制传输。

收到 SetAddress 命令时的端点初始化

本节介绍了应用程序在 SETUP 数据包中接收到 SetAddress 命令时必须执行的操作。

1. 使用在 SetAddress 命令中接收到的设备地址来对 OTG_HS_DCFG 寄存器进行编程
2. 对模块进行编程以发出状态阶段的 IN 数据包

收到 SetConfiguration/SetInterface 命令时的端点初始化

本节介绍了应用程序在 SETUP 包中接收 SetConfiguration 或 SetInterface 命令时必须执行的操作。

1. 接收到 SetConfiguration 命令时，应用程序必须对端点寄存器进行编程，以使用新配置中有效端点的特性来配置这些端点寄存器。
2. 接收到 SetInterface 命令时，应用程序必须对命令指定的端点的端点寄存器进行编程。
3. 在先前配置或其它设置中有效的端点在新的配置或其它设置中无效。必须停用这些无效端点。
4. 使用 OTG_HS_DAINTMSK 寄存器使能有效端点的中断，屏蔽无效端点的中断。
5. 为每个 FIFO 设置数据 FIFO RAM。
6. 配置完所有必需的端点后，应用程序必须对模块进行编程以发送状态阶段的 IN 数据包。

此时，设备模块已可以接收和发送任何类型的数据包。

端点激活

本节介绍激活设备端点或者将现有设备端点配置为新类型所需的步骤。

1. 在 OTG_HS_DIEPCTLx 寄存器（对于 IN 或双向端点）或 OTG_HS_DOEPCTLx 寄存器（对于 OUT 或双向端点）的以下字段中，对所需端点的特性进行编程。
   - 最大数据包大小
   - USB 活动端点位置 1
   - 端点初始数据同步位（对于中断和批量端点）
   - 端点类型
   - TxFIFO 编号
2. 激活端点后，模块便开始解码发送到该端点的令牌，并在收到的令牌有效的情况下回复有效握手信号。

端点停用

本节介绍停用现有端点所需的步骤。

1. 在要停用的端点中，将 OTG_HS_DIEPCTLx 寄存器（对于 IN 或双向端点）或 OTG_HS_DOEPCTLx 寄存器（对于 OUT 或双向端点）中的 USB 活动端点位清零。
2. 停用端点后，模块便会忽略发送到该端点的令牌，从而导致 USB 超时。

注意： *应用程序必须满足以下条件才能设置设备模块以处理通信：GINTMSK 中的 NPTXFEM 和 RXFLVLM 必须清零。*

31.13.7 操作模型

SETUP 和 OUT 数据传输

本节介绍了数据 OUT 传输和 SETUP 事务期间的内部数据流和应用程序操作步骤。

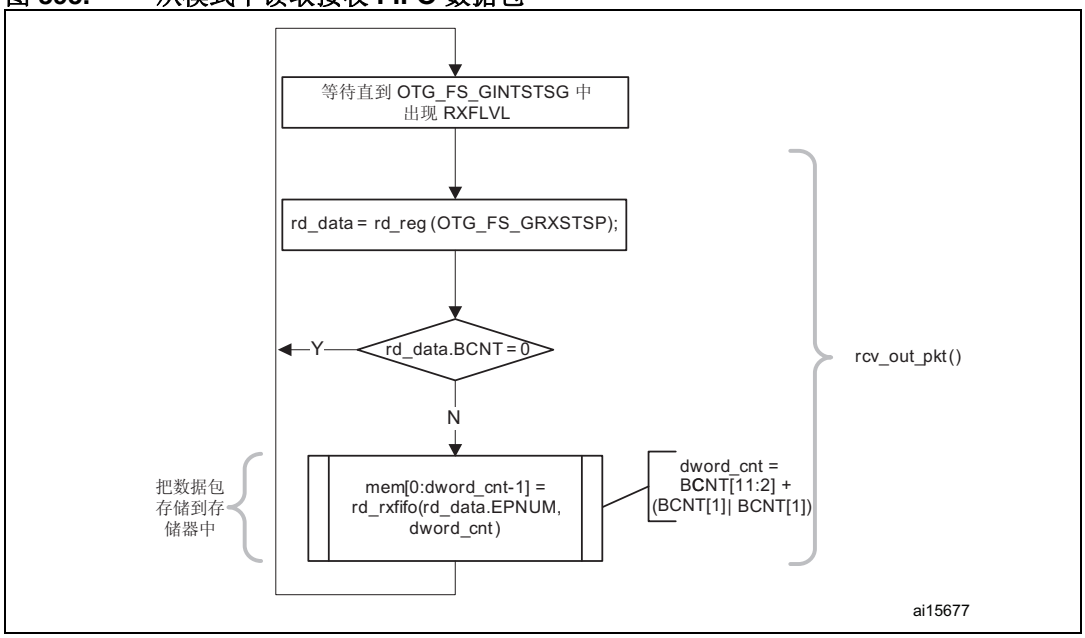
- **数据包读取**

本节介绍如何在从模式下从接收 FIFO 读取数据包（OUT 数据和 SETUP 数据包）。

1. 捕获到 RXFLVL 中断（OTG_HS_GINTSTS 寄存器）时，应用程序必须读取接收状态弹出寄存器 (OTG_HS_GRXSTSP)。
2. 应用程序可以通过写入 RXFLVL = 0（在 GINTMSK 中）来屏蔽 RXFLVL 中断（在 OTG_HS_GINTSTS 中），直到它把数据包从接收 FIFO 中读取出来。
3. 如果已接收数据包的字节计数不是 0，则从接收数据 FIFO 中弹出这些数据并存储在存储器中。如果接收到的数据包字节计数为 0，则不会从接收数据 FIFO 中弹出任何数据。
4. 从接收 FIFO 读出的数据包状态有以下几种状态：
   a) 全局 OUT NAK：
   PKTSTS = 全局 OUT NAK，BCNT = 0x000，EPNUM 和 DPID 的值无关紧要。这些数据表示全局 OUT NAK 位已生效。
   b) SETUP 数据包：
   PKTSTS = SETUP，BCNT = 0x008，EPNUM = 控制 EP 编号，DPID = D0。这些数据表示指定端点上收到的 SETUP 数据包现在可从接收 FIFO 中读取。
   c) 建立阶段完成：
   PKTSTS = 建立阶段完成，BCNT = 0x0，EPNUM = 控制 EP 编号，DPID 值无关紧要。
   这些数据表示指定端点的建立阶段完成并且数据阶段已启动。在此状态条目从接收 FIFO 中弹出后，模块将在该控制 OUT 端点上产生建立中断。
   d) OUT 数据包：
   PKTSTS = DataOUT，BCNT = 接收的 OUT 数据包的大小 ($0 \le BCNT \le 1024$)，EPNUM = 收到数据包的端点编号，DPID = 实际数据 PID。
   e) 数据传输完成：
   PKTSTS = OUT 数据传输完成，BCNT = 0x0，EPNUM = 完成数据传输的 OUT EP 编号，DPID 值无关紧要。
   这些数据表示指定 OUT 端点的 OUT 数据传输完成。在此状态条目从接收 FIFO 中弹出后，模块将在指定的 OUT 端点上引发“传输完成”中断。
5. 从接收 FIFO 中弹出数据后，必须取消对 RXFLVL 中断的屏蔽 (OTG_HS_GINTSTS)。
6. 每次应用程序检测到 OTG_HS_GINTSTS 中的 RXFLVL 中断时，都将重复步骤 1 到 5。读取空的接收 FIFO 可能导致未定义的模块行为。

*图 395*提供了上述过程的流程图。

图 395. 从模式下读取接收 FIFO 数据包



- **SETUP 事务**

本节介绍了模块处理 SETUP 数据包的方式以及应用程序处理 SETUP 事务的顺序。

- **应用程序要求**

1. 要接收 SETUP 数据包，必须将控制 OUT 端点中的 STUPCNT 字段 (OTG_HS_DOEPTSIZx) 编程为非零值。如果应用程序将 STUPCNT 字段编程为非零值，模块会接收 SETUP 数据包并将其写入接收 FIFO，而不考虑 NAK 状态和 OTG_HS_DOEPCTLx 中的 EPENA 位设置。控制端点每收到一个 SETUP 数据包后，STUPCNT 字段都会递减。如果在接收 SETUP 数据包之前，未将 STUPCNT 字段编程为适当值，模块仍能接收 SETUP 数据包并使 STUPCNT 字段递减，但应用程序可能无法确定在控制传输的建立阶段中接收的 SETUP 数据包正确数量。
   - 在 OTG_HS_DOEPTSIZx 中，STUPCNT = 3
2. 应用程序必须始终在接收数据 FIFO 中分配一些额外空间，以便能够在控制端点上接收连续的最多三个 SETUP 数据包。
   - 预留空间 10 个双字。第一个 SETUP 数据包需要 3 个双字，“建立阶段完成”状态双字需要 1 个双字，还需要 6 个双字以存储两个额外的 SETUP 数据包。
   - 每个 SETUP 数据包需要 3 个双字以存储 8 个字节的 SETUP 数据和 4 个字节的 SETUP 状态。模块将在接收 FIFO 中保留这些空间。
   - 这段 FIFO 仅用于存储 SETUP 包，绝对不会将该空间用于数据包。
3. 应用程序必须从接收 FIFO 中读取 SETUP 数据包的 2 个双字。
4. 应用程序必须从接收 FIFO 中读取并丢弃“建立阶段完成”状态双字。

- **内部数据流**

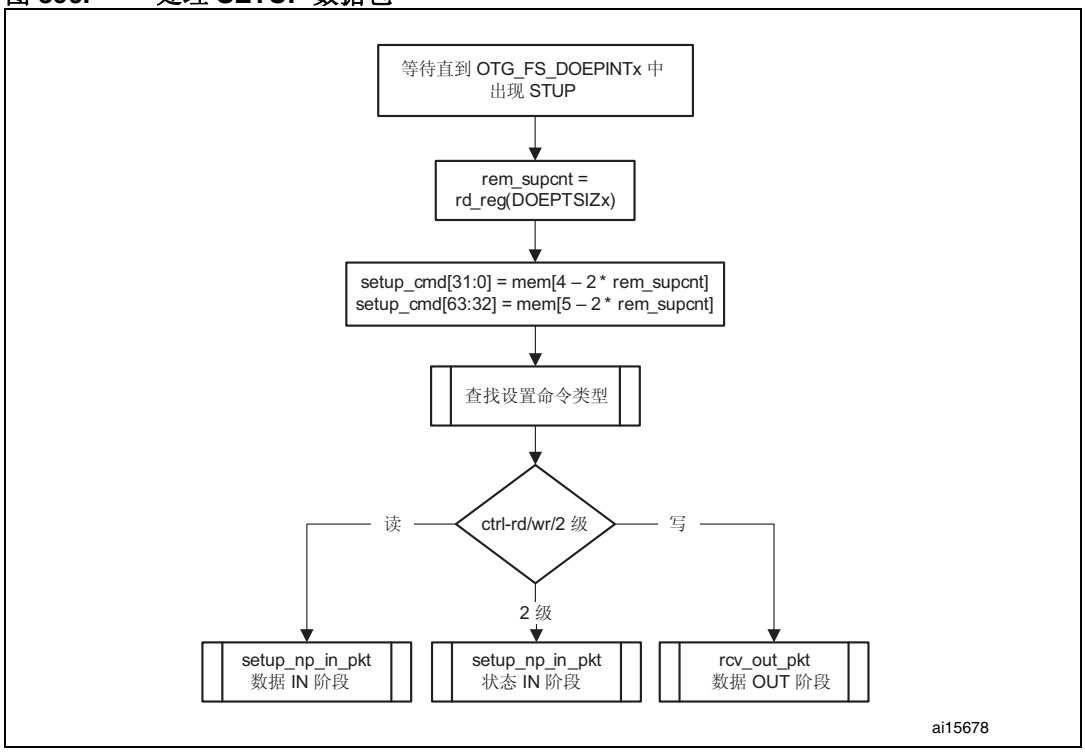
1. 接收到 SETUP 数据包时，模块会将接收到的数据写入接收 FIFO，而不会检查接收 FIFO 中的可用空间，且不考虑端点的 NAK 和 STALL 位设置。
   - 模块会在内部将接收到 SETUP 数据包的控制 IN/OUT 端点的 IN NAK 和 OUT NAK 位置 1。

2. USB 上接收到的每个 SETUP 数据包，模块会将 3 个双字的数据写入接收 FIFO，并且将 STUPCNT 字段递减 1。
   – 第一个双字包含由模块所使用的内部控制信息
   – 第二个双字包含 SETUP 命令的前 4 个字节
   – 第三个双字包含 SETUP 命令的最后 4 个字节
3. 当建立阶段结束，数据 IN/OUT 阶段开始时，模块会将一个状态条目（“建立阶段完成”双字）写入接收 FIFO，指示建立阶段完成。
4. 在 AHB 端，SETUP 数据包被应用程序读取。
5. 当应用程序从接收 FIFO 中弹出“建立阶段完成”双字时，模块将使用 STUP 中断 (OTG_HS_DOEPINTx) 来中断应用程序，指示其可以处理接收到的 SETUP 数据包。
   – 模块会将控制 OUT 端点的端点使能位清零。

- **应用程序编程顺序**

1. 对 OTG_HS_DOEPTSIZx 寄存器进行编程。
   – STUPCNT = 3
2. 等待 RXFLVL 中断 (OTG_HS_GINTSTS) 并且从接收 FIFO 中读取数据包。
3. STUP 中断的触发 (OTG_HS_DOEPINTx) 表示 SETUP 数据传输成功完成。
   – 发生该中断时，应用程序必须读取 OTG_HS_DOEPTSIZx 寄存器以确定接收的 SETUP 数据包数量并处理最后接收的 SETUP 数据包。

图 396. 处理 SETUP 数据包



- **处理三个以上连续的 SETUP 数据包**

根据 USB 2.0 规范，在 SETUP 数据包错误中，主机通常不会向同一个端点发送 3 个以上连续的 SETUP 数据包。但是，USB 2.0 规范并未限制主机可以向同一个端点发送的连续 SETUP 数据包数量。出现这种情况时，OTG_HS 控制器将生成中断（OTG_HS_DOEPINTx 中的 B2BSTUP）。

- **将全局 OUT NAK 置 1**

内部数据流：

1. 如果应用程序将全局 OUT NAK（OTG_HS_DCTL 中的 SGONAK 位）置 1，模块将停止向接收 FIFO 中写入 SETUP 数据包以外的数据。无论接收 FIFO 中可用空间大小如何，设备都会对主机发送的非同步 OUT 令牌回复 NAK，而对同步 OUT 数据包直接予以忽略。
2. 模块将全局 OUT NAK 写入接收 FIFO。应用程序必须为此留出足够空间。
3. 当应用程序从接收 FIFO 中弹出全局 OUT NAK 双字时，模块会将 GONAKEFF 中断 (OTG_HS_GINTSTS) 置 1。
4. 应用程序检测到该中断后，会认为模块处于全局 OUT NAK 模式。应用程序可以通过将 OTG_HS_DCTL 中的 SGONAK 位清零来清除该中断。

应用程序编程顺序：

1. 要停止接收任何类型的数据到接收 FIFO 中，应用程序必须通过编程以下字段以将全局 OUT NAK 位置 1。
   – 在 OTG_HS_DCTL 中，SGONAK = 1
2. 等待 OTG_HS_GINTST 中的 GONAKEFF 中断。一旦被触发，该中断表示模块已停止接收 SETUP 数据包以外的任何类型数据。
3. 如果应用程序已将 OTG_HS_DCTL 中的 SGONAK 位置 1，则在模块引发 GONAKEFF 中断 (OTG_HS_GINTSTS) 之前，应用程序可以接收有效 OUT 数据包。
4. 应用程序可通过对 GINTMSK 中的 GINAKEFFM 位执行写操作来暂时屏蔽此中断。
   – 在 GINTMSK 中，GINAKEFFM = 0
5. 当应用程序准备退出全局 OUT NAK 模式时，必须将 OTG_HS_DCTL 中的 SGONAK 位清零。此操作还会清除 GONAKEFF 中断 (OTG_HS_GINTSTS)。
   – 在 CGONAK 中，OTG_HS_DCTL = 1
6. 如果应用程序在之前已屏蔽此中断，则必须按以下方式取消对该中断的屏蔽：
   – 在 GINTMSK 中，GINAKEFFM = 1

- **禁止 OUT 端点**

应用程序必须使用以下顺序禁止已使能的 OUT 端点。

应用程序编程顺序：

1. 禁止任何 OUT 端点前，应用程序必须在模块中使能全局 OUT NAK 模式。
   – 在 OTG_HS_DCTL 中，SGONAK = 1
2. 等待 GONAKEFF 中断 (OTG_HS_GINTSTS)
3. 通过编程以下字段来禁止 OUT 端点：
   – 在 OTG_HS_DOEPCTLx 中，EPDIS = 1
   – 在 OTG_HS_DOEPCTLx 中，SNAK = 1
4. 等待 EPDISD 中断 (OTG_HS_DOEPINTx)，该中断表示已完全禁止 OUT 端点。引发 EPDISD 中断时，模块还会将以下位清零：
   – 在 OTG_HS_DOEPCTLx 中，EPDIS = 0
   – 在 OTG_HS_DOEPCTLx 中，EPENA = 0
5. 应用程序必须将全局 OUT NAK 位清零，以开始从其它未禁止的 OUT 端点接收数据。
   – 在 OTG_HS_DCTL 中，SGONAK = 0

- **通用非同步 OUT 数据传输**

本节介绍一种常规非同步 OUT 数据传输（控制、批量或中断）。

应用程序要求：

1. 建立 OUT 传输前，应用程序必须在存储器中分配一个缓冲区，以容纳要作为 OUT 传输的一部分而接收的所有数据。
2. 对于 OUT 传输，端点的传输大小寄存器中的传输大小字段必须是端点的最大数据包大小的倍数且以双字对齐。
   - 传输大小[EPNUM] = $n \times$ (MPSIZ[EPNUM] + 4 – (MPSIZ[EPNUM] mod 4))
   - 数据包计数[EPNUM] = n
   - $n > 0$
3. 发生 OUT 端点中断时，应用程序必须读取端点的传输大小寄存器以计算存储器中有效数据量。接收的有效数据量可能小于编程的传输大小。
   - 存储器中的有效数据量 = 应用程序设置的初始传输量 – 模块更新后的剩余传输量
   - 接收到 USB 数据包数 = 应用程序设置的初始数据包数 – 模块更新后的剩余数据包数

内部数据流：

1. 应用程序必须在端点相关寄存器中设置传输大小和数据包计数字段，将 NAK 位清零，并使能端点来接收数据。
2. NAK 位清零后，模块便开始接收数据并将数据写入接收 FIFO（只要接收 FIFO 中有空间）。对于 USB 上接收的每个数据包，数据包及其状态都会写入接收 FIFO。写入接收 FIFO 的每个数据包（数据量达到最大数据包大小的数据包或短数据包）都会使该端点的数据包计数字段递减 1。
   - 收到的数据包若 CRC 无效，则自动被从接收 FIFO 中清除。
   - 在 USB 上为数据包回复 ACK 后，模块将丢弃主机因无法检测到 ACK 而重新发送的非同步 OUT 数据包。应用程序不会在具有相同数据 PID 的相同端点上检测到多个连续的 OUT 数据包。在这种情况下，数据包计数不会递减。
   - 如果接收 FIFO 中没有空间，则会忽略同步或非同步数据包并且不会将它们写入接收 FIFO。此外，非同步 OUT 令牌将会收到 NAK 握手应答。
   - 在上述所有三种情况中，数据包计数都不会递减，因为没有任何数据写入接收 FIFO。
3. 当数据包计数变为 0 或者在端点上接收到短数据包时，该端点的 NAK 位将置 1。NAK 位置 1 后，将忽略同步或非同步数据包并且不会将它们写入接收 FIFO，同时非同步 OUT 令牌会收到 NAK 握手应答。
4. 在数据写入接收 FIFO 后，应用程序将从接收 FIFO 中读取数据并将数据写入外部存储器，一次一个数据包，逐个端点过来。
5. 在 AHB 上向外部存储器写入完每个数据包后，端点的传输大小都会自动减去该数据包的大小。
6. 在以下情况时，OUT 端点的 OUT 数据传输完成状态将写入接收 FIFO：
   - 传输大小为 0 并且数据包计数为 0
   - 写入接收 FIFO 的最后一个 OUT 数据包是短数据包（0 ≤ 数据包大小 < 最大数据包大小）
7. 当应用程序弹出此状态条目（OUT 数据传输完成），并生成该端点的传输完成中断，同时清零端点使能位。

应用程序编程顺序：

1. 使用传输大小和相应数据包个数对 OTG_HS_DOEPTSIZx 寄存器进行编程。
2. 使用端点特性对 OTG_HS_DOEPCTLx 寄存器进行编程，并将 EPENA 和 CNAK 位置 1。
   – 在 OTG_HS_DOEPCTLx 中，EPENA = 1
   – 在 OTG_HS_DOEPCTLx 中，CNAK = 1
3. 等待 RXFLVL 中断（在 OTG_HS_GINTSTS 中）并且从接收 FIFO 中读走数据包。
   – 此步骤可重复多次，具体取决于传输大小。
4. 触发 XFRC 中断 (OTG_HS_DOEPINTx)，以表示非同步 OUT 数据传输成功完成。
5. 读取 OTG_HS_DOEPTSIZx 寄存器，以确定有效数据量。

- **通用同步 OUT 数据传输**

本节介绍常规的同步 OUT 数据传输。

应用程序要求：

1. 非同步 OUT 数据传输的所有应用程序要求均适用于同步 OUT 数据传输。
2. 对于同步 OUT 数据传输中的传输大小和数据包计数字段，必须始终将其设置为单个帧中可接收的最大数据包大小的数据包数目。同步类型的 OUT 数据传输事务必须在一个帧内完成。
3. 在周期性帧结束（OTG_HS_GINTSTS 中的 EOPF 中断）之前，应用程序必须从接收 FIFO 中读取所有同步 OUT 数据包（数据条目和状态条目）。
4. 要接收下一帧中的数据，必须在 EOPF (OTG_HS_GINTSTS) 之后 SOF (OTG_HS_GINTSTS) 之前使能一个同步 OUT 端点。

内部数据流：

1. 同步 OUT 端点的内部数据流与非同步 OUT 端点的基本相同，但稍有差异。
2. 同步 OUT 端点通过将端点使能位置 1 并将 NAK 位清零来使能时，必须相应地将偶数/奇数帧位置 1。仅当符合以下条件时，模块才会在同步 OUT 端点上接收特定帧中的数据：
   – EONUM (in OTG_HS_DOEPCTLx) = SOFFN[0] (in OTG_HS_DSTS)
3. 当应用程序从接收 FIFO 中完整地读取一个同步 OUT 数据包（数据和状态）后，模块会根据从接收 FIFO 中读取的最后一个同步 OUT 数据包的数据 PID 更新 OTG_HS_DOEPTSIZx 中的 RXDPID 字段。

应用程序编程顺序：

1. 使用传输大小和相应数据包计数对 OTG_HS_DOEPTSIZx 寄存器进行编程。
2. 使用端点特性对 OTG_HS_DOEPCTLx 寄存器进行编程，并将端点使能位、清除 NAK 位和奇数/偶数帧位置 1。
   – EPENA = 1
   – CNAK = 1
   – EONUM =（0：偶数/1：奇数）
3. 在从模式中，等待 RXFLVL 中断（在 OTG_HS_GINTSTS 中）并从接收 FIFO 中读走数据包
   – 此步骤可重复多次，具体取决于传输大小。
4. XFRC 中断（在 OTG_HS_DOEPINTx 中）表示同步 OUT 数据传输完成。该中断不一定意味着存储器中的数据是有效的。
5. 对于同步 OUT 传输，应用程序可能并不总会检测到该中断。而是可以检测到 OTG_HS_GINTSTS 中的 IISOOXFRM 中断。

6. 读取 OTG_HS_DOEPTSIZx 寄存器以确定接收的传输大小以及帧中所接收数据的有效性。仅当符合以下条件之一时，应用程序才必须将存储器中接收的数据视为有效数据：
   - RXDPID = D0（在 OTG_HS_DOEPTSIZx 中）并且接收该有效数据的 USB 数据包数量 = 1
   - RXDPID = D1（在 OTG_HS_DOEPTSIZx 中）并且接收该有效数据的 USB 数据包数量 = 2
   - RXDPID = D2（在 OTG_HS_DOEPTSIZx 中）并且接收该有效数据的 USB 数据包数量 = 3

   接收该有效数据的 USB 数据包数量 =
   应用程序编程的初始数据包个数 - 模块更新后的剩余数据包个数

   应用程序可将无效数据包丢弃。

- **不完整的同步 OUT 数据传输**

本节介绍了同步 OUT 数据包出现丢包时应用程序编程顺序。

内部数据流：

1. 对于同步 OUT 端点，可能并不总是触发 XFRC 中断（在 OTG_HS_DOEPINTx 中）。如果模块丢弃同步 OUT 数据包，则在以下情况下，应用程序可能无法检测到 XFRC 中断 (OTG_HS_DOEPINTx)：
   - 在接收 FIFO 无法容纳完整的 ISO OUT 数据包时，模块将丢弃接收到的 ISO OUT 数据
   - 接收到的同步 OUT 数据包存在 CRC 错误
   - 模块接收到的同步 OUT 令牌损坏
   - 应用程序从接收 FIFO 中读取数据的速度非常缓慢
2. 如果模块在所有同步 OUT 端点的传输完成前检测到周期性帧结束，将触发未完成同步 OUT 数据中断（OTG_HS_GINTSTS 中的 IISOOXFRM），指示至少有一个同步 OUT 端点上未触发 XFRC 中断（在 OTG_HS_DOEPINTx 中）。此时，未完成传输的端点仍保持使能，但在 USB 的该端点上，没有进行中的有效传输。

应用程序编程顺序：

1. 硬件触发 IISOOXFRM 中断 (OTG_HS_GINTSTS) 表示当前帧中至少有一个同步 OUT 端点具有未完成的传输。
2. 如果因未从端点完全读取同步 OUT 数据而发生这种情况，应用程序必须确保首先从接收 FIFO 读取走所有同步 OUT 数据（包括数据条目和状态条目），然后再继续处理。
   - 从接收 FIFO 读取所有数据后，应用程序即可检测到 XFRC 中断 (OTG_HS_DOEPINTx)。在此情况下，应用程序必须重新使能端点以接收下一个帧中的同步 OUT 数据。
3. 当应用程序接收到 IISOOXFRM 中断（在 OTG_HS_GINTSTS 中）时，应用程序必须读取所有同步 OUT 端点的控制寄存器 (OTG_HS_DOEPCTLx)，以确定哪些端点在当前微帧中具有未完成的传输。同时满足以下两个条件时，表示端点传输未完成：
   - EONUM 位（在 OTG_HS_DOEPCTLx 中）= SOFFN[0]（在 OTG_HS_DSTS 中）
   - EPENA = 1（在 OTG_HS_DOEPCTLx 中）
4. 在检测到 SOF 中断（在 OTG_HS_GINTSTS 中）前，必须执行完成上一步操作，以确保当前帧编号未发生更改。
5. 对于具有未完成传输的同步 OUT 端点，应用程序必须丢弃存储器中的数据，并通过将 OTG_HS_DOEPCTLx 中的 EPDIS 位置 1 来禁止该端点。

6. 等待 EPDIS 中断（在 OTG_HS_DOEPINTx 中），并使能该端点，以便接收下一个帧中的新数据。
   – 由于模块可能需要一些时间才能禁止端点，因此应用程序在接收到无效同步数据后，可能无法接收下一个帧中的数据。

- **停止非同步 OUT 端点**

本节介绍应用程序如何才能停止非同步端点。

1. 将模块置于全局 OUT NAK 模式。
2. 禁止所需的端点
   – 禁止端点时，请设置 STALL = 1（在 OTG_HS_DOEPCTL 中），而不是将 OTG_HS_DOEPCTL 中的 SNAK 位置 1。
   STALL 位的优先级始终高于 NAK 位。
3. 当应用程序不再需要端点回复 STALL 握手信号时，必须将 STALL 位（在 OTG_HS_DOEPCTLx 中）清零。
4. 如果应用程序由于收到主机的 SetFeature.Endpoint Halt 或 ClearFeature.Endpoint Halt 命令来设置或清除端点的 STALL 状态，则必须在该控制端点上的状态阶段传输前，将 STALL 位置 1 或清零。

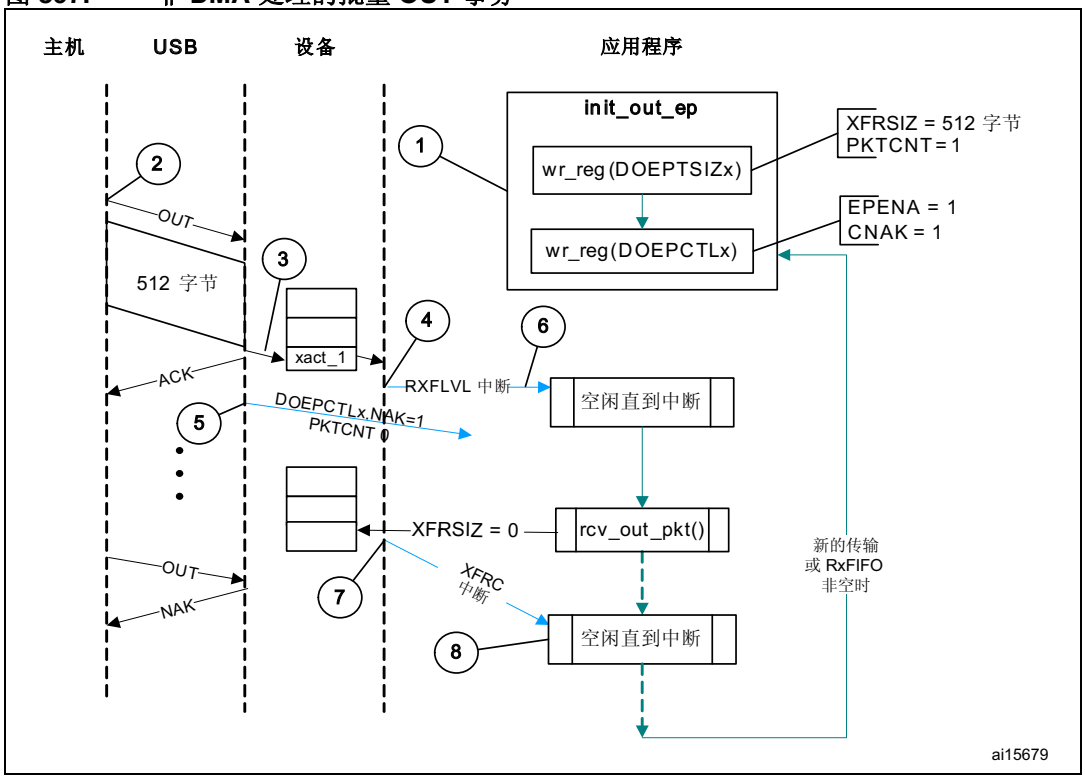
示例

本节介绍并描述了一些基本的传输类型和情景。

- 非 DMA 处理的批量 OUT 事务

图 397 描述了将单个批量 OUT 数据包从 USB 接收到 AHB 中的过程，并介绍了这一过程中涉及到的事件。

图 397. 非 DMA 处理的批量 OUT 事务



接收到 SetConfiguration/SetInterface 命令后，应用程序将初始化所有 OUT 端点：将 CNAK 和 EPENA 置 1（在 OTG_HS_DOEPCTLx 中），并将 OTG_HS_DOEPTSIZx 寄存器中的 XFRSIZ 和 PKTCNT 设置成合适的值。

1. 主机尝试将数据（OUT 令牌）发送到端点。
2. 当模块在 USB 上接收到 OUT 令牌时，模块会将数据包存储在 RxFIFO 中，因为其中有可用空间。
3. 在 RxFIFO 中写入完整数据包后，模块随后会触发 RXFLVL 中断（在 OTG_HS_GINTSTS 中）。
4. 接收到 PKTCNT 所指定数量的 USB 数据包后，模块在内部将该端点的 NAK 位置 1，以避免其再接收任何数据包。
5. 应用程序处理中断并从 RxFIFO 中读取数据。
6. 应用程序读取完所有数据后（相当于 XFRSIZ），模块将生成 XFRC 中断（在 OTG_HS_DOEPINTx 中）。
7. 应用程序处理中断并通过 XFRC 中断的触发得知本次传输完成。

IN 数据传输

- **数据包写入**

本节介绍在已使能专用发送 FIFO 的情况下应用程序如何在非 DMA 模式下将数据包写入端点 FIFO。

1. 应用程序可以选择轮询模式或中断模式。
   - 在轮询模式下，应用程序通过读取 OTG_HS_DTXFSTSx 寄存器来监视端点发送数据 FIFO 的状态，从而确定该数据 FIFO 中是否有足够空间。
   - 在中断模式中，应用程序等待 TXFE 中断（在 OTG_HS_DIEPINTx 中），然后读取 OTG_HS_DTXFSTSx 寄存器，以确定数据 FIFO 中是否有足够空间。
   - 要写入单个非零长度的数据包，数据 FIFO 中必须有足够的空间来容纳整个数据包。
   - 要写入零长度的数据包，应用程序不能查看 FIFO 空间。
2. 如果使用上述方法之一，当应用程序确定有足够空间来写入发送数据包时，应用程序必须首先对端点控制寄存器进行相应写操作，然后再将数据写入数据 FIFO。通常，应用程序必须对 OTG_HS_DIEPCTLx 寄存器执行读-修改-写操作，以避免在将端点使能位置 1 的同时，修改寄存器中的其它内容。

如果有足够空间，应用程序可将同一端点的多个数据包写入发送 FIFO。对于周期性 IN 端点，应用程序只能一次写入一个微帧内的多个数据包。只有先前一个微帧的通信事务传输完成之后，应用程序才会写入下一个微帧内要发送的所有数据包。

- **将 IN 端点 NAK 置 1**

内部数据流：

1. 当应用程序将特定端点的 IN NAK 置 1 时，模块将停止端点上的数据发送，而不考虑端点发送 FIFO 中的数据是否可用。
2. 非同步端点收到 IN 令牌，回复 NAK 握手应答
   - 同步端点收到 IN 令牌，返回零长度数据包
3. 模块在 OTG_HS_DIEPINTx 中触发 INEPNE（IN 端点 NAK 有效）中断以响应 OTG_HS_DIEPCTLx 中的 SNAK 位。
4. 应用程序检测到该中断后，便会认为端点处于 IN NAK 模式。应用程序可以通过将 OTG_HS_DIEPCTLx 中的 CNAK 位置 1 来清除该中断。

应用程序编程顺序：

1. 要在特定 IN 端点上停止发送任何数据，应用程序必须将 IN NAK 位置 1。要将该位置 1，必须编程以下字段。
   – 在 OTG_HS_DIEPCTLx 中，SNAK = 1
2. 等待 OTG_HS_DIEPINTx 中的 INEPNE 中断触发。该中断表示模块已在端点上停止发送数据。
3. 在应用程序将 NAK 位置 1 但“NAK 有效”中断尚未触发时，模块可以在端点上发送有效 IN 数据。
4. 应用程序可通过写入 DIEPMSK 中的 INEPNEM 位来临时屏蔽该中断。
   – 在 DIEPMSK 中，INEPNEM = 0
5. 要退出端点 NAK 模式，应用程序必须将 OTG_HS_DIEPCTLx 中的 NAK 状态位 (NAKSTS) 清零。此操作还会将 INEPNE 中断位（在 OTG_HS_DIEPINTx 中）清零。
   – 在 OTG_HS_DIEPCTLx 中，CNAK = 1
6. 如果应用程序已将该中断屏蔽，则必须按以下方式取消屏蔽：
   – 在 DIEPMSK 中，INEPNEM = 1

- **禁止 IN 端点**

使用以下顺序来禁止先前已使能的特定 IN 端点。

应用程序编程顺序：

1. 应用程序必须先停止在 AHB 上写入数据，之后才能禁止 IN 端点。
2. 应用程序必须将端点设置为 NAK 模式。
   – 在 OTG_HS_DIEPCTLx 中，SNAK = 1
3. 等待 OTG_HS_DIEPINTx 中的 INEPNE 中断。
4. 针对必须禁止的端点，将其 OTG_HS_DIEPCTLx 寄存器中的以下位置 1。
   – 在 OTG_HS_DIEPCTLx 中，EPDIS = 1
   – 在 OTG_HS_DIEPCTLx 中，SNAK = 1
5. OTG_HS_DIEPINTx 中的 EPDISD 中断的触发表示模块已完全禁止指定的端点。在触发中断的同时，模块还会将以下位清零：
   – 在 OTG_HS_DIEPCTLx 中，EPENA = 0
   – 在 OTG_HS_DIEPCTLx 中，EPDIS = 0
6. 应用程序必须读取周期性 IN EP 的 OTG_HS_DIEPTSIZx 寄存器，以计算该端点在 USB 上发送的数据量。
7. 应用程序必须通过将 OTG_HS_GRSTCTL 寄存器中的以下字段置 1，来清空端点发送 FIFO 中的数据：
   – TXFNUM（在 OTG_HS_GRSTCTL 中）= 端点发送 FIFO 编号
   – TXFFLSH（在 OTG_HS_GRSTCTL 中）= 1

应用程序必须轮询 OTG_HS_GRSTCTL 寄存器，直至模块将 TXFFLSH 位清零，这表示 FIFO 清空操作结束。要在该端点上发送新数据，应用程序可以稍后重新使能该端点。

- **通用非周期性 IN 数据传输**

应用程序要求：

1. 建立 IN 传输前，应用程序必须确保组成一次 IN 传输的每个数据包都可以容纳在单个缓冲区中。
2. 对于 IN 传输，端点传输大小寄存器中的传输大小字段表示本次传输的有效数据量，它由多个最大数据包大小和单个短数据包组成。该短数据包在传输结束时发送。
   - 要发送多个最大数据包大小的数据包并在传输结束时外加一个短数据包：

     传输大小[EPNUM] = *x* × MPSIZ[EPNUM] + sp

     如果 (sp > 0)，数据包计数[EPNUM] = x + 1。
     否则，数据包计数[EPNUM] = *x*
   - 要发送单个零长度数据包：

     传输大小[EPNUM] = 0

     数据包计数[EPNUM] = 1
   - 要发送多个最大数据包大小的数据包并在传输结束时外加一个零长度数据包，应用程序必须将传输拆分为两个部分。第一部分发送最大数据包大小的数据包，第二部分仅发送零长度数据包。

     第一次传输：传输大小[EPNUM] = *x* × MPSIZ[epnum]；数据包计数 = *n*；

     第二次传输：传输大小[EPNUM] = 0；数据包计数 = 1；
3. 使能某个端点进行数据传输后，模块会更新传输大小寄存器。在 IN 传输结束时，应用程序必须读取传输大小寄存器，以确定送入发送 FIFO 中的数据已有多少通过 USB 发送出去。
4. 送入发送 FIFO 中的数据量 = 应用程序编程的初始传输大小 - 模块更新后的最终传输大小
   - 通过 USB 已经发送的数据量 = (应用程序编程的初始数据包计数 - 模块更新后的最终数据包计数) × MPSIZ[EPNUM]
   - 要通过 USB 发送的剩余数据量 = (应用程序编程的初始传输大小 - 已通过 USB 发送的数据量)

内部数据流：

1. 应用程序必须在特定端点的寄存器中设置传输大小和数据包计数字段，并使能该端点来发送数据。
2. 应用程序还必须向该端点的发送 FIFO 写入必需的数据。
3. 应用程序每向发送 FIFO 写入一个数据包，该端点的传输大小便会自动减去该数据包大小。应用程序持续从存储器获取数据来写入发送 FIFO，直到该端点的传输大小变为 0。向 FIFO 写入数据后，“FIFO 中的数据包数”计数会递增（这是一个 3 位计数，由模块在内部进行维护，每个 IN 端点发送 FIFO 对应一个。在 IN 端点 FIFO 中，模块所维护的最大数据包数始终为八个）。对于零长度数据包，每个 FIFO 均另有一个单独的标志，FIFO 中没有任何数据。
4. 当数据写入发送 FIFO 后，模块会在接收到 IN 令牌时将这些数据送出。每个数据包发送出去并收到回复的 ACK 握手信号后，该端点的数据包计数都会递减 1，直到数据包计数变 0 为止。发生超时时，数据包计数不会递减。
5. 对于零长度数据包（由内部零长度标志指示），模块会针对 IN 令牌发出一个零长度数据包，并递减数据包计数字段的值。
6. 如果接收到 IN 令牌的端点对应的 FIFO 中无数据，且该端点的数据包计数字段为零，则模块会针对该端点生成一个“TxFIFO 为空时接收到 IN 令牌”(ITTXFE) 中断（前提是该端点的 NAK 位未置 1）。模块在该非同步端点上回复 NAK 握手信号。

7. 模块会在内部使 FIFO 指针重新返回到开头，并且不会生成超时中断。
8. 当传输大小为 0 且数据包计数为 0 时，将生成该端点的传输完成 (XFRC) 中断，同时将端点使能清零。

应用程序编程顺序：

1. 在 OTG_HS_DIEPTSIZx 寄存器中编程传输大小和相应的包计数。
2. 在 OTG_HS_DIEPCTLx 寄存器中编程端点特性，并将 CNAK 和 EPENA（端点使能）位置 1。
3. 发送非零长度数据包时，应用程序必须轮询 OTG_HS_DTXFSTSx 寄存器（其中 x 为与该端点相关联的 FIFO 编号），以确定数据 FIFO 中是否有足够的空间。写入数据前，应用程序可以选择使用 TXFE（在 OTG_HS_DIEPINTx 中）。

- **通用周期性 IN 数据传输**

本节介绍典型的周期性 IN 数据传输。

应用程序要求：

1. *第 1180 页的通用非周期性 IN 数据传输*的应用程序要求 1、2、3 和 4 对周期性 IN 数据传输同样适用（只是对要求 2 稍加修改）。
   - 应用程序只能发送若干个最大数据包大小的数据包或若干个最大数据包大小的包，外加传输结束时的一个短数据包。要发送多个最大数据包大小的数据包并在传输结束时外加一个短数据包，必须满足以下条件：

     传输大小[EPNUM] = x × MPSIZ[EPNUM] + sp
     （其中 *x* 是整数 ≥ 0，且 0 ≤ sp < MPSIZ[EPNUM]）

     如果 (sp > 0)，数据包计数[EPNUM] = *x* + 1
     否则，数据包计数[EPNUM] = *x*；

     MCNT[EPNUM] = 数据包计数[EPNUM]
   - 应用程序无法在传输结束时发送零长度数据包。应用程序可以单独发送一个零长度数据包。要发送单个零长度数据包：
   - 传输大小[EPNUM] = 0

     数据包计数[EPNUM] = 1

     MCNT[EPNUM] = 数据包计数[EPNUM]
2. 应用程序一次只能安排一帧的数据传输。
   - (MCNT – 1) × MPSIZ ≤ XFERSIZ ≤ MCNT × MPSIZ
   - PKTCNT = MCNT（在 OTG_HS_DIEPTSIZx 中）
   - 如果 XFERSIZ < MCNT × MPSIZ，则传输的最后一个数据包为短数据包。
   - 请注意：MCNT 在 OTG_HS_DIEPTSIZx 中，MPSIZ 在 OTG_HS_DIEPCTLx 中，PKTCNT 在 OTG_HS_DIEPTSIZx 中，而 XFERSIZ 在 OTG_HS_DIEPTSIZx 中
3. 接收到 IN 令牌前，应用程序必须将要在帧中发送的完整数据写入到发送 FIFO 中。在接收到 IN 令牌时，即使发送 FIFO 中该帧要发送的数据只差 1 个双字未写进来，模块也会像 FIFO 为空时一样操作。当发送 FIFO 为空时：
   - 同步端点上将回复零长度数据包
   - 中断端点上将回复 NAK 握手信号
4. 对于一个帧中含三个数据包的高带宽 IN 端点，应用程序可以将端点 FIFO 大小编程为最大数据包大小的两倍，并且在第一个数据包在 USB 上传输之后加载第三个数据包。

内部数据流：

1. 应用程序必须在特定端点的寄存器中设置传输大小和数据包计数字段，并使能该端点来发送数据。
2. 应用程序还必须向与该端点相关联的发送 FIFO 写入必需的数据。
3. 应用程序每向发送 FIFO 写入一个数据包，该端点的传输大小便会自动减去该数据包大小。应用程序持续从存储器获取数据来写入发送 FIFO，直到该端点的传输大小变为 0。
4. 当周期性端点接收到 IN 令牌时，模块将开始发送 FIFO 中的数据（如果 FIFO 中有数据）。如果 FIFO 中没有该帧要发送的数据的完整数据包，则模块将为该端点生成一个“TxFIFO 为空时接收到 IN 令牌”中断。
   - 同步端点上将回复零长度数据包
   - 中断端点上将回复 NAK 握手信号
5. 端点的数据包计数会在下列情况下递减 1：
   - 对于同步端点，在发送零长度或非零长度数据包时递减
   - 对于中断端点，在发送 ACK 握手信号时递减
   - 当传输大小和数据包计数均为 0 时，将生成该端点的传输完成中断，同时将端点使能位清零。
6. 在“周期性帧间隔”（由 OTG_HS_DCFG 中的 PFIVL 位控制）内，当模块发现为当前帧安排的任何同步 IN 端点 FIFO 中的数据还未发送完成时，都会在 OTG_HS_GINTSTS 中生成一个 IISOIXFR 中断。

应用程序编程顺序：

1. 在 OTG_HS_DIEPCTLx 寄存器中编程端点特性，并将 CNAK 和 EPENA 位置 1。
2. 将需要在下一帧中发送的数据写入发送 FIFO。
3. 硬件触发 ITTXFE 中断（在 OTG_HS_DIEPINTx 中）表示应用程序尚未将需要发送的所有数据写入发送 FIFO。
4. 如果在检测到中断前已使能中断端点，则将忽略该中断。如果中断端点未使能，则使能此端点，以便数据能够在收到下一次 IN 令牌时发送出去。
5. 硬件触发 XFRC 中断（在 OTG_HS_DIEPINTx 中）时如果 OTG_HS_DIEPINTx 中未产生 ITTXFE 中断，则表示同步 IN 传输成功完成。读取 OTG_HS_DIEPTSIZx 寄存器时始终得到传输大小 = 0 且数据包计数 = 0，表示所有数据都已通过 USB 发送完毕。
6. 无论是否产生 ITTXFE 中断（OTG_HS_DIEPINTx 中），只要触发 XFRC 中断（在 OTG_HS_DIEPINTx 中），即表示中断 IN 传输成功完成。读取 OTG_HS_DIEPTSIZx 寄存器时始终得到传输大小 = 0 且数据包计数 = 0，表示所有数据都已通过 USB 发送完毕。
7. 如果在未产生任何前述中断的情况下在 OTG_HS_GINTSTS 中触发未完成同步 IN 传输 (IISOIXFR) 中断，则表示模块在当前帧中至少未收到 1 个周期性 IN 令牌。

- **未完成同步 IN 数据传输**

本节介绍应用程序针对未完成同步 IN 数据传输必须执行的操作。

内部数据流：

1. 符合下列条件之一时，即认为同步 IN 传输未完成：
   a) 模块在至少一个同步 IN 端点上接收到损坏的同步 IN 令牌。此时，应用程序检测到未完成同步 IN 传输中断（OTG_HS_GINTSTS 中的 IISOIXFR 位）。
   b) 应用程序向发送 FIFO 写入数据的速度过慢，在将完整数据写入 FIFO 之前便接收到 IN 令牌。此时，应用程序在 OTG_HS_DIEPINTx 中检测到“TxFIFO 为空时接收到 IN 令牌”中断。应用程序可忽略此中断，因为最终将在周期性帧结束时产生一个未完成同步 IN 传输中断（OTG_HS_GINTSTS 中的 IISOIXFR 位）。

   模块会通过 USB 发送一个零长度数据包来响应接收到的 IN 令牌。

2. 应用程序必须尽快停止向发送 FIFO 写入数据。
3. 应用程序必须将端点的 NAK 位和禁止位置 1。
4. 模块会禁止该端点，将禁止位清零并触发端点的“端点禁止”中断。

应用程序编程顺序

1. 应用程序可以在任何同步 IN 端点上忽略 OTG_HS_DIEPINTx 中的“TxFIFO 为空时接收到 IN 令牌”中断，因为最终将产生一个未完成同步 IN 传输中断（在 OTG_HS_GINTSTS 中）。
2. 硬件触发未完成同步 IN 传输中断（在 OTG_HS_GINTSTS 中）表示在至少一个同步 IN 端点上存在未完成的同步 IN 传输。
3. 应用程序必须读取所有同步 IN 端点的“端点控制”寄存器来检测存在未完成 IN 数据传输的端点。
4. 应用程序必须停止向与这些端点相关联的“周期性发送 FIFO”写入数据。
5. 对 OTG_HS_DIEPCTLx 寄存器中的下列字段进行编程以禁止端点：
   – 在 OTG_HS_DIEPCTLx 中，SNAK = 1
   – 在 OTG_HS_DIEPCTLx 中，EPDIS = 1
6. 硬件触发 OTG_HS_DIEPINTx 中的“端点禁止”中断表示模块已禁止该端点。
   – 此时，应用程序必须清空相关联的发送 FIFO 中的数据，或者通过使能端点在下一微帧中发送新传输，来覆盖 FIFO 中的现有数据。要刷新数据，应用程序必须使用 OTG_HS_GRSTCTL 寄存器。

- **停止非同步 IN 端点**

本节介绍应用程序如何才能停止非同步端点。

应用程序编程顺序：

1. 禁止要停止的 IN 端点。同时将 STALL 位置 1。
2. OTG_HS_DIEPCTLx 中的 EPDIS = 1（当端点已使能时）
   – OTG_HS_DIEPCTLx 中的 STALL = 1
   – STALL 位的优先级始终高于 NAK 位
3. 硬件触发“端点禁止”中断（在 OTG_HS_DIEPINTx 中），应用程序获知模块已禁止指定的端点。
4. 应用程序必须根据端点类型清空非周期性或周期性发送 FIFO。对于非周期性端点，应用程序必须重新使能其他无需停止的非周期性端点来发送数据。
5. 当应用程序准备好结束该端点的 STALL 握手信号时，必须将 OTG_HS_DIEPCTLx 的 STALL 位清零。
6. 如果应用程序因收到来自主机的 SetFeature.Endpoint Halt 命令或ClearFeature.Endpoint Halt 命令来设置或清除端点的 STALL 状态，则必须在该控制端点的状态阶段传输之前将 STALL 位置 1 或清零。

特例：停止控制 OUT 端点

如果在控制传输的数据阶段，主机发送的 IN/OUT 令牌数超过 SETUP 数据包指定的值，则模块必须对这些多余的 IN/OUT 令牌回复 STALL。在这种情况下，在控制传输的数据阶段，当模块传输完 SETUP 数据包指定的数据量后，应用程序必须使能 OTG_HS_DIEPINTx 中的 ITTXFE 中断和 OTG_HS_DOEPINTx 中的 OTEPDIS 中断。随后，当应用程序收到此中断时，必须将相应端点控制寄存器中的 STALL 位置 1 并清除此中断。

31.13.8 最坏情况下的响应时间

当 OTG_HS 控制器作为设备使用时，对于任何跟随在同步 OUT 之后的令牌，都存在一个最坏响应时间。这个最坏情况响应时间随 AHB 时钟频率的不同而异。

模块寄存器位于 AHB 域内，在更新这些寄存器值之前，模块不会接受新令牌。对于任何跟随在同步 OUT 之后的令牌，最坏的情况是：由于同步事务不需要握手信号，所以下一个令牌可能很快就到达。当 AHB 与 PHY 时钟频率相同时，这个最坏情况值为 7 个 PHY 时钟。AHB 时钟越快，此值越小。

如果发生这种最坏情况，模块将以 NAK 响应批量/中断令牌，并丢弃同步和 SETUP 令牌。对于 SETUP，主机会将这种情况视为超时，并尝试重新发送 SETUP 数据包。对于同步传输，会产生未完成同步 IN 传输中断 (IISOIXFR) 和未完成同步 OUT 传输中断 (IISOOXFR)，来通知应用程序同步 IN/OUT 数据包被丢弃。

在 OTG_HS_GUSBCFG 中选择 TRDT 值

TRDT 的值 (OTG_HS_GUSBCFG) 是指在接收到 IN 令牌后以 PHY 时钟计算的时间长度，MAC 将在这段时间内获取 FIFO 状态并从 PFC 模块获取第一个数据。这段时间包含 PHY 和 AHB 时钟之间的同步延迟。最坏情况延迟是当 AHB 时钟与 PHY 时钟相等时的延迟。这种情况下的延迟为 5 个时钟周期。

MAC 接收到 IN 令牌后，此信息（接收到的令牌）将由 PFC（PFC 以 AHB 时钟运行）同步到 AHB 时钟。随后，PFC 从 SPRAM 中读取数据并将它们写入双时钟源缓冲区。MAC 再从源缓冲区读出数据（4 个深度）。

如果 AHB 的运行频率高于 PHY，应用程序可以使用较小的 TRDT 值（在 OTG_HS_GUSBCFG 中）。

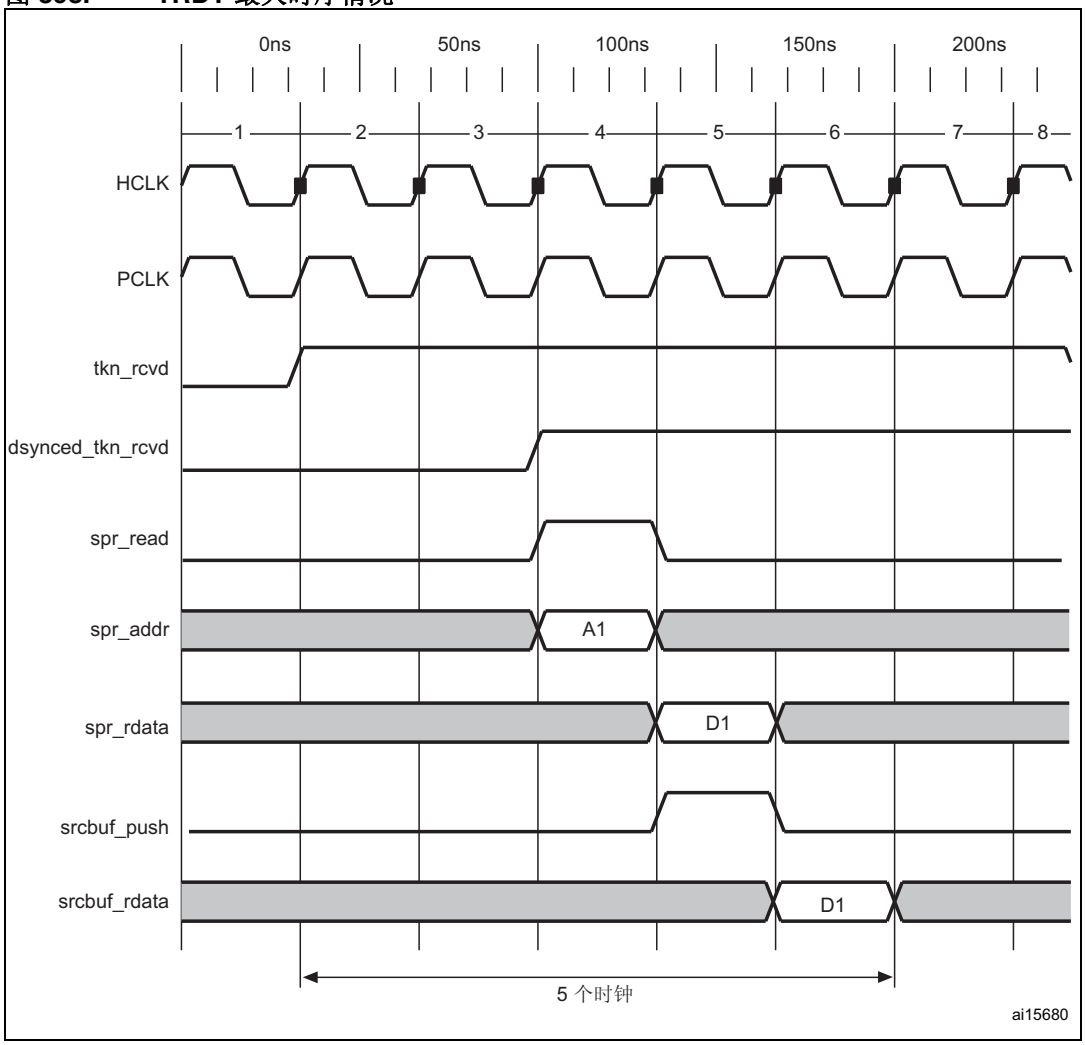
图 398 具有以下信号：

- tkn_rcvd：接收到令牌信息（从 MAC 到 PFC）
- dynced_tkn_rcvd：双倍同步 tkn_rcvd（从 PCLK 到 HCLK 域）
- spr_read：读取到 SPRAM
- spr_addr：寻址到 SPRAM
- spr_rdata：从 SPRAM 中读取数据
- srcbuf_push：压入源缓冲区
- srcbuf_rdata：从源缓冲区读取数据。MAC 看到的数据

应用程序可以使用下列公式来计算 TRDT 的值：

4 × AHB 时钟 + 1 个 PHY 时钟 = (2 个时钟进行同步 + 1 个时钟进行存储器寻址 + 1 个时钟从同步 RAM 读取存储器数据) + 1 个 PHY 时钟（下一个 PHY 时钟 MAC 可以对 2 个时钟 FIFO 输出进行采样）

图 398. TRDT 最大时序情况



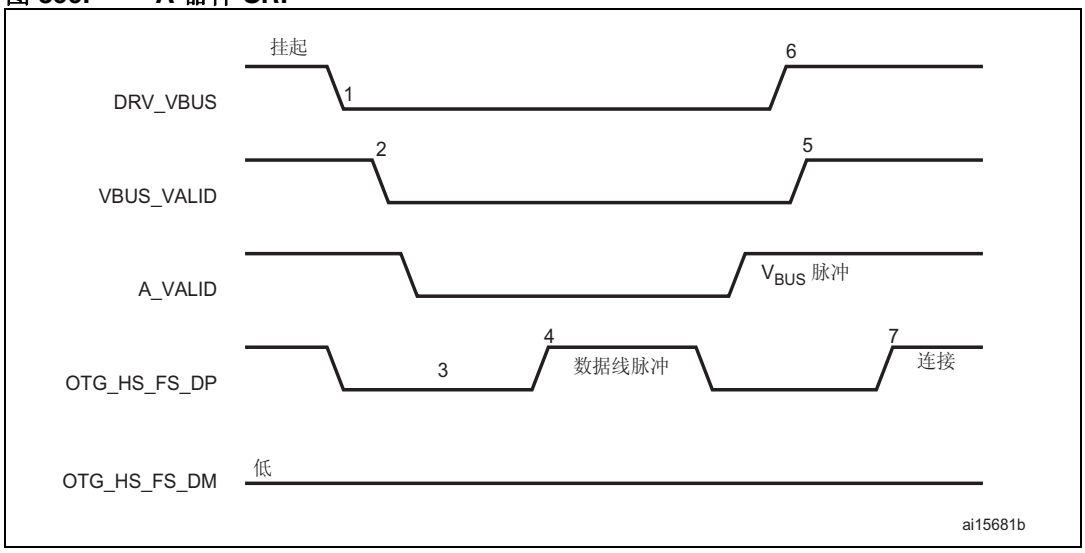
31.13.9 OTG 编程模型

OTG_HS 控制器是一种支持 HNP 和 SRP 的 OTG 设备。该模块接口与“A 型”插头连接时，该模块称为 A 器件。该模块接口与“B 型”插头连接时，该模块称为 B 器件。在主机模式中，OTG_HS 控制器将关闭 V_{BUS} 以节省电能。B 器件可以借助 SRP 请求 A 器件打开 V_{BUS} 电源。设备必须同时执行数据线脉冲和 V_{BUS} 脉冲，但主机可以检测数据线脉冲或者 V_{BUS} 脉冲中的一个用于 SRP。B 器件可以借助 HNP 协商并切换到主机角色。在协商模式下执行 HNP 后，B 器件会挂起总线并恢复到设备角色。

A 器件会话请求协议

应用程序必须将模块 USB 配置寄存器中的 SRP 使能位置 1。这将使能 OTG_HS 控制器在 A 器件模式下检测到 SRP。

图 399. A 器件 SRP



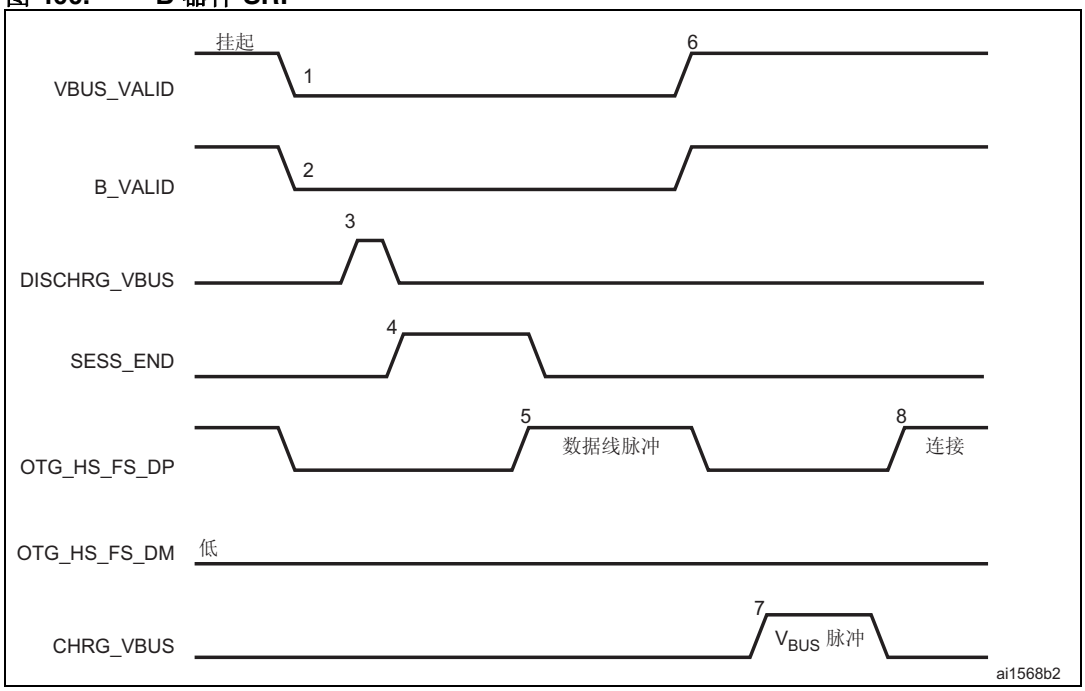
1. DRV_VBUS = 发送到 PHY 的 V_{BUS} 驱动信号
   VBUS_VALID = 来自 PHY 的 V_{BUS} 有效信号
   A_VALID = 发送到 PHY 的 A 器件 V_{BUS} 电平信号
   DP = 正向数据线
   DM = 负向数据线

1. 为节省电能，在总线空闲时，应用程序会通过在主机端口控制和状态寄存器中写入端口挂起位和端口电源位来挂起设备并关闭端口电源。
2. PHY 通过禁止 VBUS_VALID 信号来指示端口电源已关闭。
3. 当 V_{BUS} 电源关闭时，设备必须检测到至少 2 ms 的 SE0 信号才可以启动 SRP。
4. 要启动 SRP，设备需要打开其数据线的上拉电阻并维持 5 到 10 ms。OTG_HS 控制器会检测到数据线脉冲。
5. 设备会驱动 V_{BUS} 到 A 器件会话有效电平（最小 2.0 V）以上，以产生 V_{BUS} 脉冲。
   OTG_HS 控制器检测到 SRP 时将中断应用程序。在全局中断状态寄存器中，检测到会话请求位（OTG_HS_GINTSTS 中的 SRQINT 位）将置 1。
6. 应用程序必须响应“检测到会话请求”中断，并通过在主机端口控制和状态寄存器中写入端口电源位来打开端口电源位。PHY 通过输出 VBUS_VALID 信号来指示端口已通电。
7. 当 USB 通电后，设备将连接，从而完成 SRP 过程。

B 器件会话请求协议

应用程序必须将模块 USB 配置寄存器中的 SRP 使能位置 1。这将使能 OTG_HS 控制器在 B 器件模式下启动 SRP。OTG_HS 控制器可以借助 SRP 向主机请求新会话。

图 400. B 器件 SRP



1. VBUS_VALID = 来自 PHY 的 V_{BUS} 有效信号
   B_VALID = 发送到 PHY 的 B 器件有效会话
   DISCHRG_VBUS = 发送到 PHY 的放电信号
   SESS_END = 发送到 PHY 的会话结束信号
   CHRG_VBUS = 发送到 PHY 的 V_{BUS} 充电信号
   DP = 正向数据线
   DM = 负向数据线

1. 为节省电能，在总线空闲时主机会挂起并关闭端口电源。

   OTG_HS 控制器会在总线空闲 3 ms 后，将模块中断寄存器中的早期挂起位置 1。随后，OTG_HS 控制器会将模块中断寄存器中的 USB 挂起位置 1。

   OTG_HS 控制器会通知 PHY 对 V_{BUS} 进行放电。

2. PHY 指示会话结束。这是 SRP 的初始条件。启动 SRP 之前，OTG_HS 控制器需要 2 ms 的 SE0 信号。

   对于 USB 1.1 全速串行收发器，应用程序必须在 BSVLD 位（位于OTG_HS_GOTGCTL）被禁止后，等待 V_{BUS} 放电至 0.2 V。此放电时间可从收发器供应商处获取，该值可能因收发器而异。

3. USB OTG 模块通知 PHY 对 V_{BUS} 加速放电。

4. 应用程序通过将 OTG 控制和状态寄存器中的会话请求位置 1 来启动 SRP。OTG_HS 控制器先执行数据线脉冲，随后执行 V_{BUS} 脉冲。

5. 主机会从数据线脉冲或 V_{BUS} 脉冲检测到 SRP，然后打开 V_{BUS}。PHY 指示 V_{BUS} 对设备上电。

6. OTG_HS 控制器执行 V_{BUS} 脉冲。

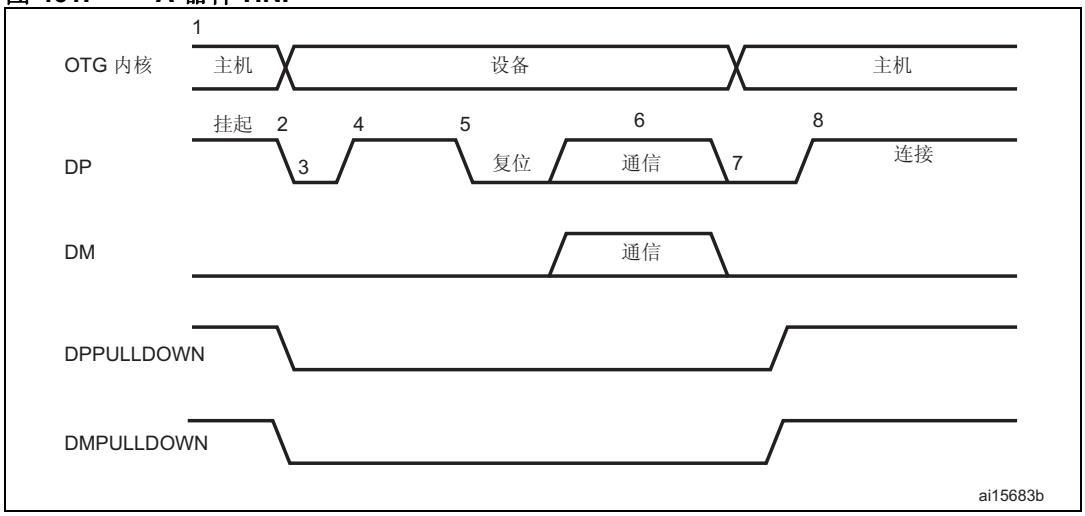
   主机通过打开 V_{BUS} 启动一个新会话，指示 SRP 已成功。OTG_HS 控制器通过将 OTG 中断状态寄存器中的会话请求成功状态更改位置 1 来中断应用程序。应用程序读取 OTG 控制和状态寄存器中的会话请求成功位。

7. 当 USB 通电时，OTG_HS 控制器将进行连接，并完成 SRP 过程。

A 器件主机协商协议

通过 HNP 可以将 USB 主机角色从 A 器件切换到 B 器件。应用程序必须将模块 USB 配置寄存器中的 HNP 使能位置 1，以使能 OTG_HS 控制器在 A 器件模式下执行 HNP 功能。

图 401. A 器件 HNP



1. DPPULLDOWN = 从模块发送到 PHY 的信号，用于使能/禁止 PHY 内部 DP 线的下拉电阻。
   DMPULLDOWN = 从模块发送到 PHY 的信号，用于使能/禁止 PHY 内部 DM 线的下拉电阻。

1. OTG_HS 控制器向 B 器件发送 SetFeature b_hnp_enable 描述符以使能 HNP 支持。B 器件的 ACK 响应表示 B 器件支持 HNP。应用程序必须将 OTG 控制和状态寄存器中的主机设置 HNP 使能位置 1，以向 OTG_HS 控制器表明 B 器件支持 HNP。
2. 应用程序不再使用总线时，会通过在主机端口控制和状态寄存器中写入端口挂起位来挂起总线。
3. B 器件检测到 USB 挂起时，将断开连接，指示 HNP 的初始条件。B 器件只有在切换到主机角色时才会启动 HNP，否则总线会保持挂起状态。

   OTG_HS 控制器将 OTG 中断状态寄存器中的检测到主机协商中断置 1，指示 HNP 已开始。

   OTG_HS 控制器禁止 PHY 中的 DM 下拉电阻和 DM 下拉电阻，以指示设备角色。PHY 使能 OTG_HS_DP 上拉电阻，以告知 B 器件 A 器件的连接。

   应用程序必须读取 OTG 控制和状态寄存器中的当前模式位，以确定设备工作模式。
4. B 器件检测到连接，发出 USB 复位信号，并枚举 OTG_HS 进行数据通信。
5. B 器件继续担当主机角色，发起数据通信，并在结束时挂起总线。

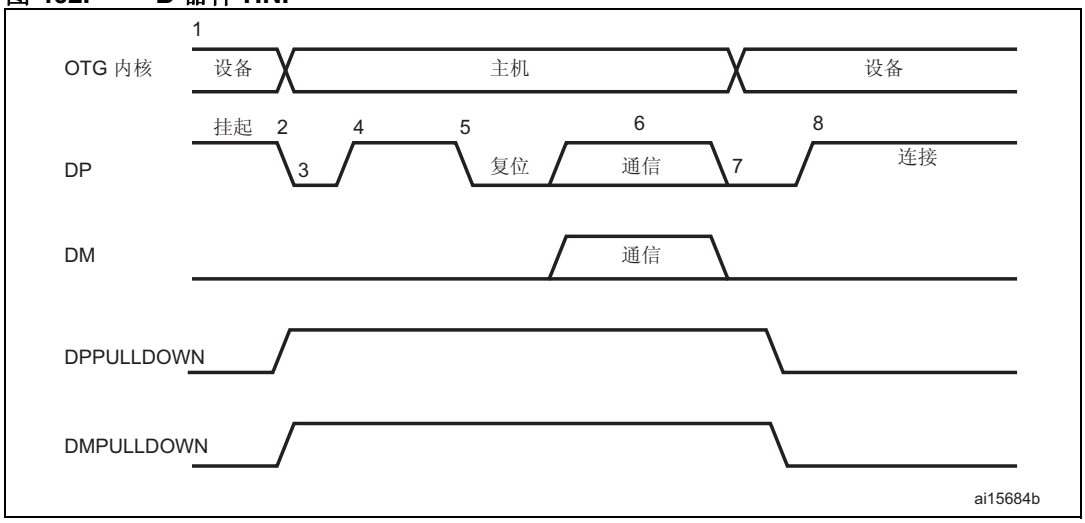
   OTG_HS 控制器会在总线空闲 3 ms 后，将模块中断寄存器中的早期挂起位置 1。随后，OTG_HS 控制器会将模块中断寄存器中的 USB 挂起位置 1。
6. 在协商模式下，OTG_HS 控制器会检测到总线挂起，连接断开，然后切换回主机角色。OTG_HS 控制器将使能 PHY 中 DP 和 DM 下拉电阻，以指示其担当着主机角色。

7. OTG_HS 控制器将 OTG 中断状态寄存器中的连接器 ID 状态更改中断置 1。应用程序必须读取 OTG 控制和状态寄存器中的连接器 ID 状态位，以确定 OTG_HS 控制器在 A 器件模式下工作。这向应用程序表示 HNP 已经完成。应用程序必须读取 OTG 控制和状态寄存器中的当前模式位，以确定主机工作模式。
8. 连接 B 器件，完成 HNP 过程。

B 器件主机协商协议

通过 HNP 可以将 USB 主机角色从 B 器件切换到 A 器件。应用程序必须将模块 USB 配置寄存器中的 HNP 使能位置 1，以使能 OTG_HS 控制器在 B 器件模式下执行 HNP 功能。

图 402. B 器件 HNP



1. DPPULLDOWN = 从模块发送到 PHY 的信号，用于使能/禁止 PHY 内部 DP 线的下拉电阻。
DMPULLDOWN = 从模块发送到 PHY 的信号，用于使能/禁止 PHY 内部 DM 线的下拉电阻。

1. A 器件发出一个 SetFeature b_hnp_enable 描述符以使能 HNP 支持。OTG_HS 控制器的 ACK 响应表示它支持 HNP。应用程序必须将 OTG 控制和状态寄存器中的设备 HNP 使能位置 1，以指示支持 HNP。

   应用程序将 OTG 控制和状态寄存器中的 HNP 请求位置 1，以向 OTG_HS 控制器指示启动 HNP。
2. A 器件不再使用总线时，会通过在主机端口控制和状态寄存器中写入端口挂起位来挂起总线。

   OTG_HS 控制器会在总线空闲 3 ms 后，将模块中断寄存器中的早期挂起位置 1。随后，OTG_HS 控制器会将模块中断寄存器中的 USB 挂起位置 1。

   OTG_HS 控制器会断开连接，A 器件在总线上检测到 SE0，指示 HNP 即将开始。OTG_HS 控制器将使能 PHY 中 DP 和 DM 下拉电阻，以指示其承担主机角色。

   在检测到 SE0 3 ms 内，A 器件会通过激活 OTG_HS_DP 上拉电阻进行响应。OTG_HS 控制器检测到此信号时认为有设备接入。

   OTG_HS 控制器将 OTG 中断状态寄存器中的主机协商成功状态更改中断位置 1，指示 HNP 的状态。应用程序必须读取 OTG 控制和状态寄存器中的主机协商成功位，以确定主机协商成功。应用程序必须读取模块中断寄存器 (OTG_HS_GINTSTS) 中的当前模式位，以确定主机工作模式。

3. 应用程序将复位位（OTG_HS_HPRT 中的 PRST 位）置 1，同时 OTG_HS 控制器发出 USB 复位信号，并枚举 A 器件进行数据通信。
4. OTG_HS 控制器继续保持发起通信的主机角色，在通信结束后，会通过将主机端口控制和状态寄存器中的端口挂起位置 1 来挂起总线。
5. 在协商模式下，当 A 器件检测到挂起时，会断开连接，然后切换回主机角色。OTG_HS 控制器将禁止 PHY 中 DP 和 DM 的下拉电阻，以指示其承担设备角色。
6. 应用程序必须读取模块中断寄存器 (OTG_HS_GINTSTS) 中的当前模式位，以确定主机工作模式。
7. OTG_HS 控制器进行连接，完成 HNP 过程。

32 灵活的静态存储控制器 (FSMC)

除非特别说明，否则本节适用于整个 STM32F4xx 系列器件。

32.1 FSMC 主要特性

FSMC 能够连接同步、异步存储器和 16 位 PC 存储卡。其主要用途如下：

- 将 AHB 数据通信事务转换为适当的外部器件协议
- 满足外部器件的访问时序要求

所有外部存储器共享地址、数据和控制信号，但有各自的片选信号。FSMC 一次只能访问一个外部器件。

FSMC 具有以下主要功能：

- 连接静态存储器映射的器件：
  - 静态随机访问存储器 (SRAM)
  - 只读存储器 (ROM)
  - NOR Flash/OneNAND Flash
  - PSRAM（4 个存储区域）
- 两个带有 ECC 硬件的 NAND Flash 存储区域，可检查多达 8 KB 的数据
- 16 位 PC 卡兼容设备
- 支持对同步器件（NOR Flash 和 PSRAM）的突发模式访问
- 8 或 16 位宽的数据总线
- 每个存储区域有独立的片选控制
- 每个存储区域可独立配置
- 可对时序进行编程，以支持各种器件，尤其是：
  - 等待周期可编程（最多 15 个时钟周期）
  - 总线周转周期可编程（最多 15 个时钟周期）
  - 输出使能和写入使能延迟可编程（最多 15 个时钟周期）
  - 独立的读和写时序和协议，以支持各种存储器和时序
- 写使能和字节通道选择输出，可配合 PSRAM 和 SRAM 器件使用
- 将 32 位的 AHB 事务转换为针对外部 16 位或 8 位器件进行的连续 16 位或 8 位访问。
- 用于写入的 FIFO，2 字长（对于 STM32F42x 和 STM32F43x，为 16 字长），每个字为 32 位宽，仅用于存储数据，而不存储地址。因此，此 FIFO 仅会缓冲 AHB 批量写事务。从而可对慢速存储器执行写入操作后能快速释放 AHB，以供其它操作使用。每次仅缓冲一个突发事务：如果在有操作正在进行时发生一个新的 AHB 突发事务或者一个单独事务，则 FIFO 将会清空。FSMC 将插入等待周期，直至当前存储器访问已完成）。
- 外部异步等待控制

定义外部器件类型和其特性的 FSMC 寄存器通常在启动时进行设置，并且在下次上电或复位前保持不变。但也可随时更改这些设置。

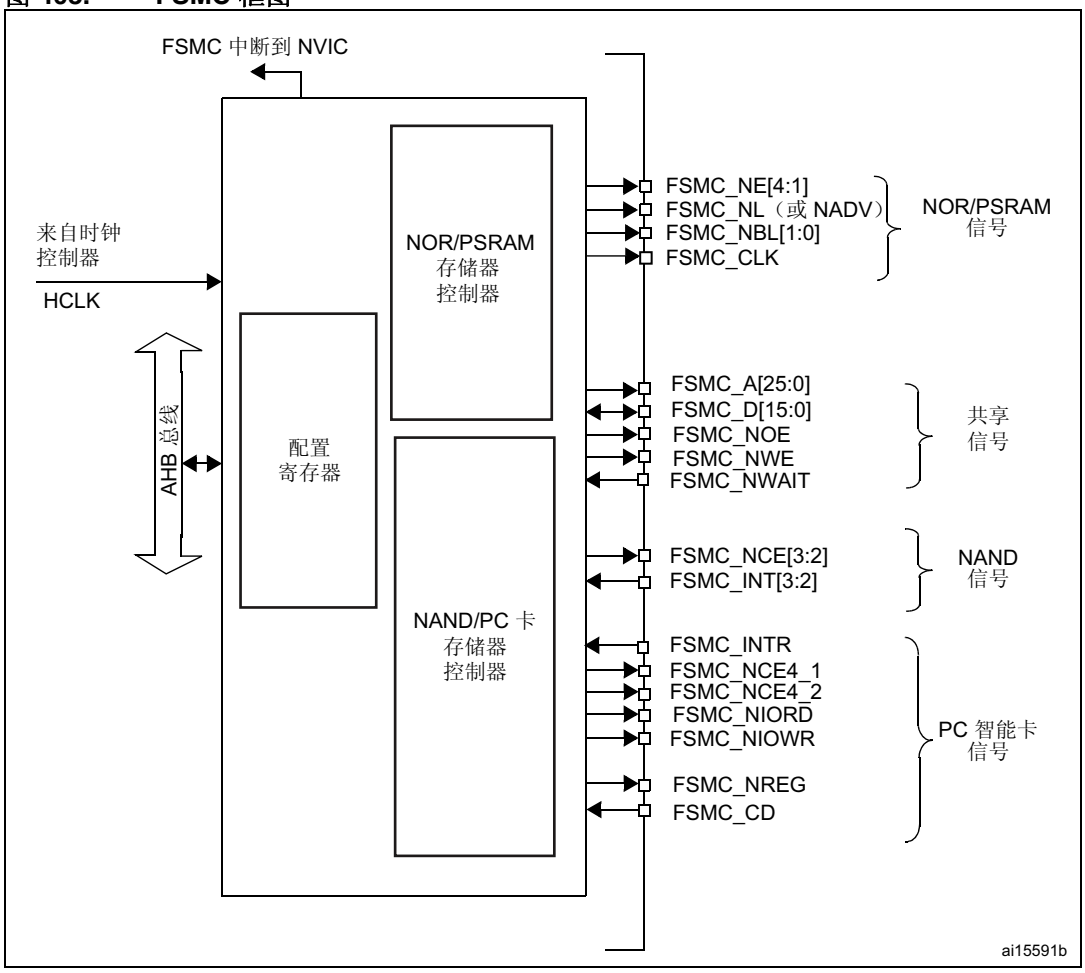
32.2 框图

FSMC 包含四个主要模块：

- AHB 接口（包括 FSMC 配置寄存器）
- NOR Flash/PSRAM 控制器
- NAND Flash/PC 卡控制器
- 外部器件接口

框图如*图 403* 所示。

图 403. FSMC 框图



32.3 AHB 接口

CPU 和其它 AHB 总线主设备可通过该 AHB 从设备接口访问外部静态存储器。

AHB 事务会转换为外部器件协议。尤其是当所选外部存储器的宽度为 16 位或 8 位时，AHB 中的 32 位宽事务将被划分成多个连续的 16 或 8 位访问。片选将在每次访问时进行切换。

出现以下条件时，FSMC 将产生 AHB 错误：

- 读取或写入未使能的 FSMC 存储区域
- 在 FSMC_BCRx 寄存器中的 FACCEN 位复位时读取或写入 NOR Flash 存储区域
- 在输入引脚 FSMC_CD（Card Presence Detection）为低电平时读取或写入 PC 卡存储区域

此 AHB 错误的影响具体取决于尝试进行读写访问的 AHB 主设备：

- 如果为 Cortex™-M4F CPU，则会生成硬性故障 (hard fault) 中断
- 如果为 DMA，则会生成 DMA 传输错误，并会自动禁用相应的 DMA 通道

AHB 时钟 (HCLK) 是 FSMC 的参考时钟。

32.3.1 支持的存储器和事务

通用事务规则

所请求的 AHB 事务传输大小可以是 8、16 或 32 位，但访问的外部器件具有固定的数据宽度。这可能会导致不一致的数据宽度。

因此，必须遵循一些简单的事务规则：

- AHB 事务数据宽度和存储器数据宽度相等
  在此情况下没有任何问题。
- AHB 事务数据宽度大于存储器宽度
  在此情况下，FSMC 会将 AHB 事务分为多个较小的连续存储器访问，以便符合外部数据宽度。
- AHB 事务数据宽度小于存储器宽度
  异步传送可能一致，也可能不一致，具体取决于外部器件的类型。
  - 对具有字节选择功能的器件（SRAM、ROM、PSRAM）进行异步访问。
    a) FSMC 允许写入事务通过其字节选择通道 NBL[1:0] 访问恰当的数据
    b) 允许读取事务。会读取所有存储器字节，并将丢弃无用的存储器字节。NBL[1:0] 在读取事务期间保持为低电平。
  - 对不具有字节选择功能的器件（16 位 NOR 和 NAND Flash）进行异步访问。
    当请求对 16 位宽的 Flash 存储器进行字节访问时会发生此情形。显然，不能在字节模式下访问此器件（只能针对 Flash 存储器读取或写入 16 位字），因此：
    a) 不允许写入事务
    b) 允许读取事务。会读取所有存储器字节，并将丢弃无用的存储器字节。NBL[1:0] 在读取事务期间保持低电平。

配置寄存器

FSMC 可通过一个寄存器组进行配置。有关 NOR Flash/PSRAM 控制寄存器的详细说明，请参见*第 32.5.6 节*。有关 NAND Flash/PC 卡寄存器的详细说明，请参见*第 32.6.8 节*。

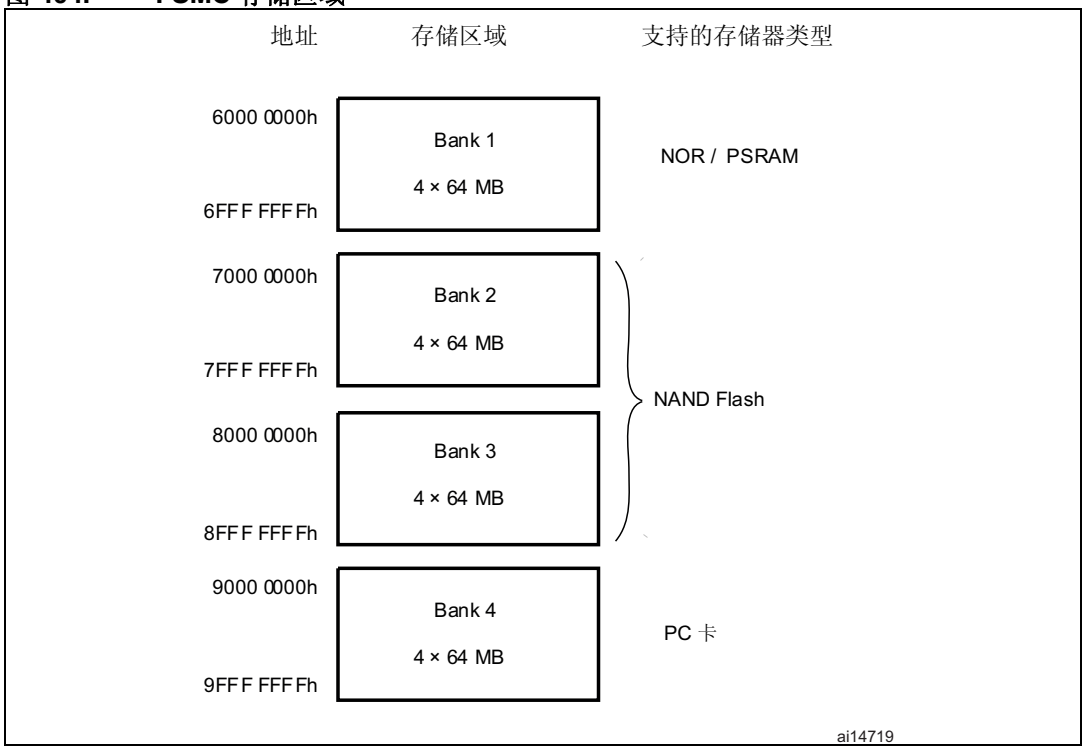
32.4 外部器件地址映射

从 FSMC 的角度，外部存储器被划分为 4 个固定大小的存储区域，每个存储区域的大小为 256 MB（请参见*图 404*）：

- 存储区域 1 可连接多达 4 个 NOR Flash 或 PSRAM 存储器器件。此存储区域被划分为 4 个 NOR/PSRAM 区域，带 4 个专用片选信号。
- 存储区域 2 和 3 用于连接 NAND Flash 器件（每个存储区域一个器件）
- 存储区域 4 用于连接 PC 卡设备

对于每个存储区域，所要使用的存储器类型由用户在配置寄存器中定义。

图 404. FSMC 存储区域



32.4.1 NOR/PSRAM 地址映射

HADDR[27:26] 位用于从*表 185* 中所示的四个存储区域之中选择其中一个存储区域。

表 185. NOR/PSRAM 存储区域选择

HADDR[27:26](1)	选择的存储区域
00	存储区域 1 NOR/PSRAM 1
01	存储区域 1 NOR/PSRAM 2
10	存储区域 1 NOR/PSRAM 3
11	存储区域 1 NOR/PSRAM 4

1. HADDR 是 AHB 内部地址线，但也会参与对外部存储器的寻址。

HADDR[25:0] 包含外部存储器地址。由于 HADDR 为字节地址，而存储器按字寻址，所以根据存储器数据宽度不同，实际向存储器发送的地址也将有所不同，如下表所示。

表 186. 外部存储器地址

存储器宽度(1)	向存储器发出的数据地址	最大存储器容量（位）
8 位	HADDR[25:0]	64 MB x 8 = 512 Mb
16 位	HADDR[25:1] >> 1	64 MB/2 x 16 = 512 Mb

1. 如果外部存储器的宽度为 16 位，FSMC 将使用内部的 HADDR[25:1] 地址来作为对外部存储器的寻址地址 FSMC_A[24:00]。
无论外部存储器的宽度为 16 位还是 8 位，FSMC_A[0] 都应连接到外部存储器地址 A[0]。

NOR Flash/PSRAM 的回卷支持

不支持同步存储器的回绕突发模式。存储器必须按未定义长度的线性突发模式进行配置。

32.4.2 NAND/PC 卡地址映射

在此情况下，有三个存储区域，每个存储区域分为各个存储空间，如*表 187* 所示。

表 187. 存储器映射和时序寄存器

起始地址	结束地址	FSMC 存储区域	存储空间	时序寄存器
0x9C00 0000	0x9FFF FFFF	存储区域 4 - PC 卡	I/O 区	FSMC_PIO4 (0xB0)
0x9800 0000	0x9BFF FFFF		特性区	FSMC_PATT4 (0xAC)
0x9000 0000	0x93FF FFFF		通用区	FSMC_PMEM4 (0xA8)
0x8800 0000	0x8BFF FFFF	存储区域 3 - NAND Flash	特性区	FSMC_PATT3 (0x8C)
0x8000 0000	0x83FF FFFF		通用区	FSMC_PMEM3 (0x88)
0x7800 0000	0x7BFF FFFF	存储区域 2 - NAND Flash	特性区	FSMC_PATT2 (0x6C)
0x7000 0000	0x73FF FFFF		通用区	FSMC_PMEM2 (0x68)

对于 NAND Flash 存储器，通用区和特性区存储空间分为三个部分，均位于低位 256 KB 中（见下面的*表 188*）：

- 数据区域（通用/特性存储空间中的第一个 64 KB）
- 命令区域（通用/特性存储空间中的第二个 64 KB）
- 地址区域（通用/特性存储空间中的下一个 128 KB）

表 188. NAND 存储区域选择

部分名称	HADDR[17:16]	地址范围
地址区域	1X	0x020000-0x03FFFF
命令区域	01	0x010000-0x01FFFF
数据区域	00	0x000000-0x0FFFF

应用程序软件使用这 3 个区域来访问 NAND Flash 存储器：

- **向 NAND Flash 存储器发送命令**：软件可以向命令区域中的任意存储器位置写入命令值。
- **指定读取或写入的 NAND Flash 地址**：软件可以向地址区域中的任意存储位置写入地址值。由于地址的长度可以是 4 或 5 个字节（具体取决于实际存储器大小），要指定完整的地址，需要对地址区域执行多个连续写入操作。
- **读取或写入数据**：软件将从数据区域中的任意存储器位置读取数据值，或者向其中写入数据值。

由于 NAND Flash 存储器会自动递增地址，所以在访问连续存储器位置时，无需递增数据区域的地址。

32.5 NOR Flash/PSRAM 控制器

FSMC 会生成适当的信号时序，以驱动以下类型的存储器：

- 异步 SRAM 和 ROM
  - 8 位
  - 16 位
  - 32 位
- PSRAM（Cellular RAM）
  - 异步模式
  - 突发模式
  - 复用或非复用
- NOR Flash
  - 异步模式或突发模式
  - 复用或非复用

FSMC 会为每个存储区域输出唯一的片选信号 NE[4:1]。所有其它信号（地址、数据和控制）均为共享信号。

对于同步访问，FSMC 只有在读/写事务期间才会向所选的外部器件发出时钟 (CLK)。HCLK 时钟频率是该时钟的整数倍。每个存储区域的大小固定，均为 64 MB。

每个存储区域都通过专用的寄存器进行配置（请参见*第 32.5.6 节*）。

存储器的可编程参数包括访问时序（请参见*表 189*）和对等待管理的支持（用于在突发模式下访问 NOR Flash 和 PSRAM）。

表 189. NOR/PSRAM 的可编程访问参数

参数	功能	访问模式	单位	最小值	最大值
地址建立	地址建立阶段的持续时间	异步	AHB 时钟周期 (HCLK)	0	15
地址保持	地址保持阶段的持续时间	异步，复用 I/O	AHB 时钟周期 (HCLK)	1	15
数据建立	数据建立阶段的持续时间	异步	AHB 时钟周期 (HCLK)	1	256
总线周转	总线周转阶段的持续时间	异步和同步读取	AHB 时钟周期 (HCLK)	0	15

表 189. NOR/PSRAM 的可编程访问参数 （续）

参数	功能	访问模式	单位	最小值	最大值
时钟分频比	构建一个存储器时钟周期 (CLK) 所需的 AHB 时钟周期 (HCLK) 数量	同步	AHB 时钟周期 (HCLK)	2	16
数据延迟	在发出突发的第一个数据前向存储器发出的时钟周期数量	同步	存储器时钟周期 (CLK)	2	17

32.5.1 外部存储器接口信号

表 190、*表 191* 和*表 192* 列出了通常用于连接 NOR Flash、SRAM 和 PSRAM 的信号。

注意： *前缀“N”表示相关的信号为低电平有效。*

NOR Flash，非复用 I/O

表 190. 非复用 I/O NOR Flash

FSMC 信号名称	I/O	功能
CLK	O	时钟（用于同步突发）
A[25:0]	O	地址总线
D[15:0]	I/O	双向数据总线
NE[x]	O	片选，x = 1..4
NOE	O	输出使能
NWE	O	写入使能
NL(=NADV)	O	锁存使能（对于部分 NOR Flash 器件，此信号也称为地址有效 (NADV)）
NWAIT	I	FSMC 的 NOR Flash 等待输入信号

NOR Flash 存储器采用 16 位字寻址。最大容量为 512 Mb（26 个地址线）。

NOR Flash，复用 I/O

表 191. 复用 I/O NOR Flash

FSMC 信号名称	I/O	功能
CLK	O	时钟（用于同步突发）
A[25:16]	O	地址总线
AD[15:0]	I/O	16 位复用双向地址/数据总线
NE[x]	O	片选，x = 1..4
NOE	O	输出使能
NWE	O	写入使能
NL(=NADV)	O	锁存使能（对于部分 NOR Flash 器件，此信号也称为地址有效 (NADV)）
NWAIT	I	FSMC 的 NOR Flash 等待输入信号

NOR-Flash 存储器采用 16 位字寻址。最大容量为 512 Mb（26 个地址线）。

PSRAM/SRAM，非复用 I/O

表 192. 非复用 I/O PSRAM/SRAM

FSMC 信号名称	I/O	功能
CLK	O	时钟（仅用于 PSRAM 同步突发）
A[25:0]	O	地址总线
D[15:0]	I/O	数据双向总线
NE[x]	O	片选，x = 1..4（在 PSRAM 应用中被称作 NCE（Cellular RAM，即 CRAM））
NOE	O	输出使能
NWE	O	写入使能
NL(= NADV)	O	仅用于 PSRAM 输入的地址有效信号（存储器信号名称：NADV）
NWAIT	I	PSRAM 发送给 FSMC 的等待输入信号
NBL[1]	O	高字节使能（存储器信号名称：NUB）
NBL[0]	O	低字节使能（存储器信号名称：NLB）

PSRAM 存储器采用 16 位字寻址。最大容量为 512 Mb（26 个地址线）。

PSRAM，复用 I/O

表 193. 复用 I/O PSRAM

FSMC 信号名称	I/O	功能
CLK	O	时钟（用于同步突发）
A[25:16]	O	地址总线
AD[15:0]	I/O	16 位复用双向地址/数据总线
NE[x]	O	片选，x = 1..4（在 PSRAM 应用中被称作 NCE（Cellular RAM，即 CRAM））
NOE	O	输出使能
NWE	O	写入使能
NL(= NADV)	O	仅用于 PSRAM 输入的地址有效信号（存储器信号名称：NADV）
NWAIT	I	PSRAM 发送给 FSMC 的等待输入信号
NBL[1]	O	高字节使能（存储器信号名称：NUB）
NBL[0]	O	低字节使能（存储器信号名称：NLB）

PSRAM 存储器采用 16 位字寻址。最大容量为 512 Mb（26 个地址线）。

32.5.2 支持的存储器和事务

下面的*表 194* 显示的是当 NOR、PSRAM 和 SRAM 的存储器数据总线为 16 位时所支持的设备、访问模式和事务的示例。在此示例中，FSMC 不允许（或不支持）的事务显示为灰色。

表 194. NOR Flash/PSRAM 控制器：支持的存储器和传输类型举例

设备	模式	R/W	AHB 数据大小	存储器数据大小	是否允许	注释
NOR Flash（复用 I/O 和非复用 I/O）	异步	R	8	16	是	
	异步	W	8	16	否	
	异步	R	16	16	是	
	异步	W	16	16	是	
	异步	R	32	16	是	分为 2 次 FSMC 访问
	异步	W	32	16	是	分为 2 次 FSMC 访问
	异步页	R	-	16	否	不支持该模式
	同步	R	8	16	否	
	同步	R	16	16	是	
	同步	R	32	16	是	
PSRAM（复用 I/O 和非复用 I/O）	异步	R	8	16	是	
	异步	W	8	16	是	使用字节通道 NBL[1:0]
	异步	R	16	16	是	
	异步	W	16	16	是	
	异步	R	32	16	是	分为 2 次 FSMC 访问
	异步	W	32	16	是	分为 2 次 FSMC 访问
	异步页	R	-	16	否	不支持该模式
	同步	R	8	16	否	
	同步	R	16	16	是	
	同步	R	32	16	是	
	同步	W	8	16	是	使用字节通道 NBL[1:0]
	同步	W	16/32	16	是	
SRAM 和 ROM	异步	R	8/16	16	是	
	异步	W	8/16	16	是	使用字节通道 NBL[1:0]
	异步	R	32	16	是	分为 2 次 FSMC 访问
	异步	W	32	16	是	分为 2 次 FSMC 访问。 使用字节通道 NBL[1:0]

32.5.3 通用时序规则

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
- 在同步模式（读取或写入）下，输出的数据在 HCLK 的上升沿变化。无论 CLKDIV 值为何，所有输出均会按以下方式变化：
  - 当出现 FSMC_CLK 时钟的下降沿时，NOEL/NWEL/NEL/NADVL/NADVH/NBLL/ 地址有效输出可发生变化。
  - 当出现 FSMC_CLK 时钟的上升沿时，NOEH/NWEH/NEH/NOEH/NBLH/ 地址有效输出可发生变化。

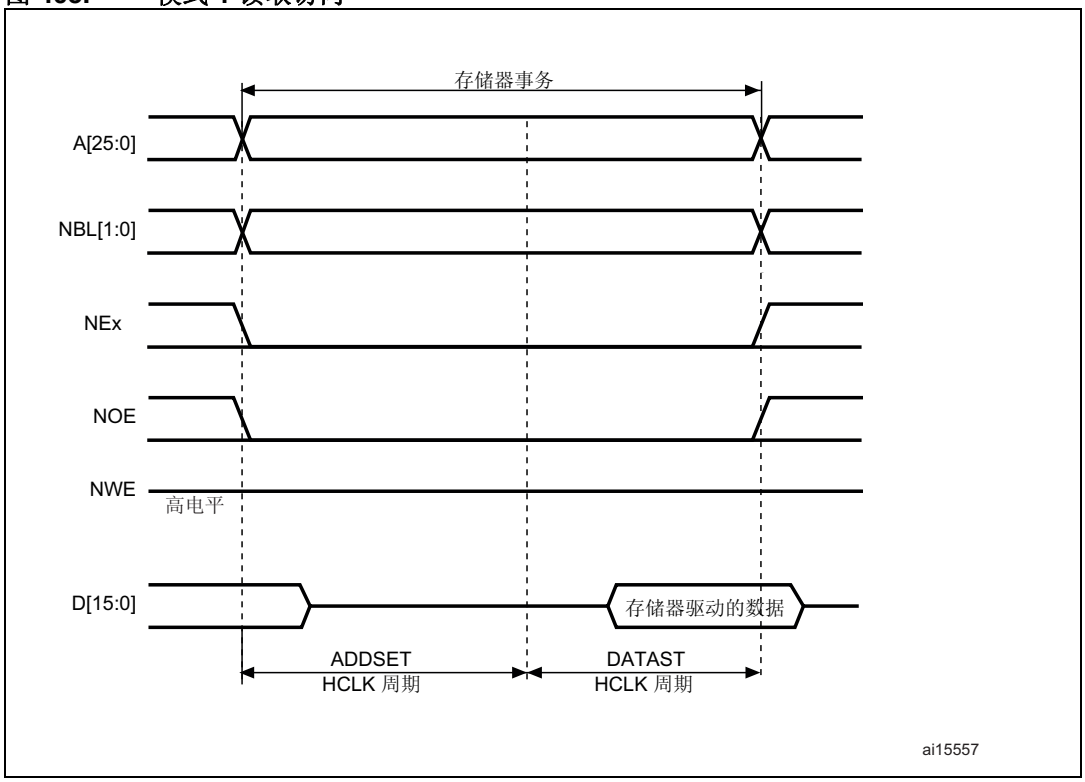
32.5.4 NOR Flash/PSRAM 控制器异步事务

异步静态存储器（NOR Flash、PSRAM、SRAM）

- 信号通过内部时钟 HCLK 进行同步。不会将此时钟发送到存储器
- FSMC 总是会先对数据进行采样，而后再禁止片选信号 NE。这样可以确保符合存储器数据保持时序的要求（数据转换的芯片使能高电平，通常最低为 0 ns。）
- 如果使能扩展模式（FSMC_BCRx 寄存器中的 EXTMOD 位置 1），则最多可提供四种扩展模式（A、B、C 和 D）。可以混合使用 A、B、C 和 D 模式来进行读取和写入操作。例如，可以在模式 A 下执行读取操作，而在模式 B 下执行写入操作。
- 如果禁用扩展模式（FSMC_BCRx 寄存器中的 EXTMOD 位复位），则 FSMC 可以在模式 1 或模式 2 下运行，如下所述：
  - 当选择 SRAM/CRAM 存储器类型时，模式 1 为默认模式（FSMC_BCRx 寄存器中 MTYP = 0x0 或 0x01）。
  - 当选择 NOR 存储器类型时，模式 2 为默认模式（FSMC_BCRx 寄存器中 MTYP = 0x10）。

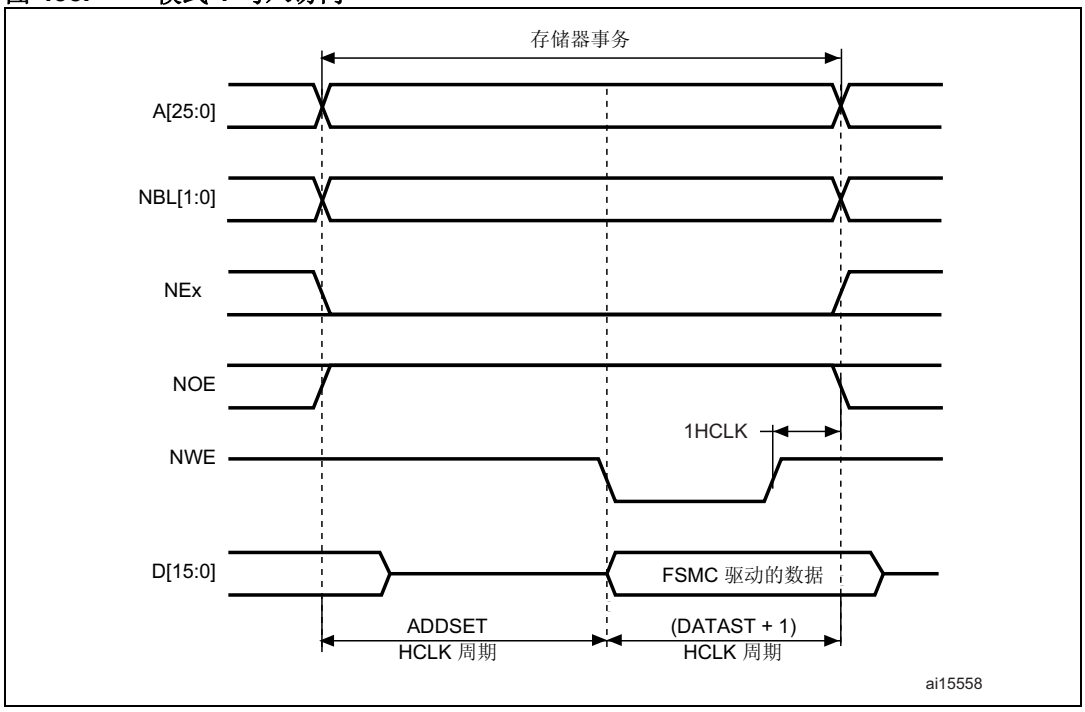
模式 1 - SRAM/PSRAM (CRAM)

图 405. 模式 1 读取访问



1. NBL[1:0] 在进行读取访问时为低电平。

图 406. 模式 1 写入访问



位于写入事务末尾的一个 HCLK 周期有助于确保 NWE 上升沿之后的地址和数据保持时间。由于存在此 HCLK 周期，DATAST 值必须大于零 (DATAST > 0)。

表 195. FSMC_BCRx 位字段

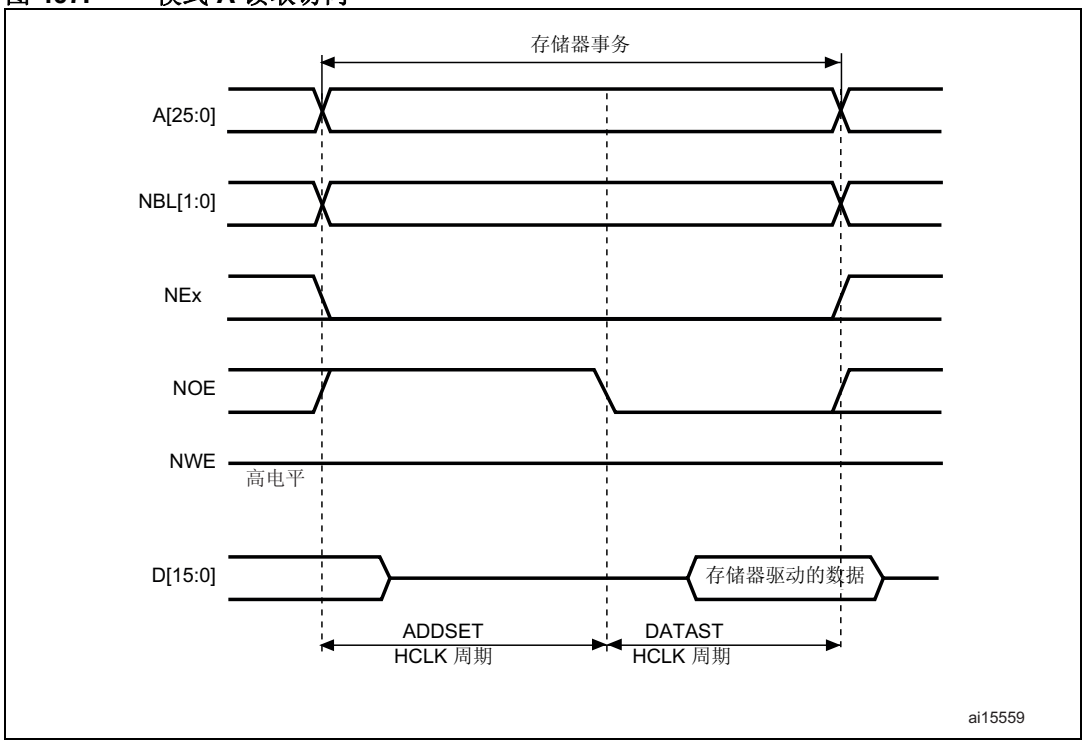
位号	位名	要设置的值
31-20	保留	0x000
19	CBURSTRW	0x0（对异步模式没有影响）
18:16	保留	0x0
15	ASYNCWAIT	如果存储器支持该特性，则置为 1。否则，保持为 0。
14	EXTMOD	0x0
13	WAITEN	0x0（对异步模式没有影响）
12	WREN	0x1
11	WAITCFG	根据需要进行设置
10	WRAPMOD	0x0
9	WAITPOL	仅当位 15 为 1 时才有意义
8	BURSTEN	0x0
7	保留	0x1
6	FACCEN	无关
5-4	MWID	根据需要进行设置
3-2	MTYP	根据需要进行设置，0x2 除外 (NOR Flash)
1	MUXE	0x0
0	MBKEN	0x1

表 196. FSMC_BTRx 位字段

位号	位名	要设置的值
31:30	保留	0x0
29-28	ACCMOD	0x0
27-24	DATLAT	无关
23-20	CLKDIV	无关
19-16	BUSTURN	NEx 变为高电平到 NEx 变为低电平之间的时间 (BUSTURN HCLK)
15-8	DATAST	第二个访问阶段的持续时间（写入访问为 DATAST+1 个 HCLK 周期，读取访问为 DATAST 个 HCLK 周期）。
7-4	ADDHLD	无关
3-0	ADDSET	第一个访问阶段的持续时间（ADDSET 个 HCLK 周期）。 ADDSET 的最小值为 0。

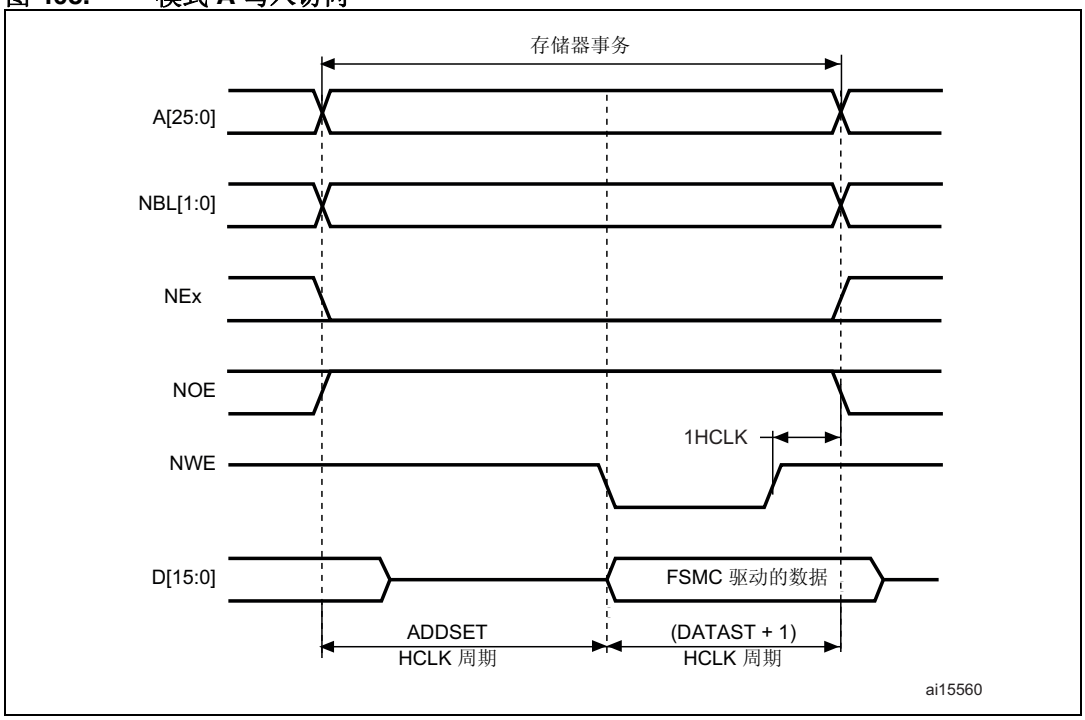
模式 A - SRAM/PSRAM (CRAM) OE 切换

图 407. 模式 A 读取访问



1. NBL[1:0] 在进行读取访问时为低电平。

图 408. 模式 A 写入访问



与模式 1 的不同之处在于 NOE 的切换与独立的读取和写入时序。

表 197. FSMC_BCRx 位字段

位号	位名	要设置的值
31-20	保留	0x000
19	CBURSTRW	0x0（对异步模式没有影响）
18:16	保留	0x0
15	ASYNCWAIT	如果存储器支持该特性，则置为 1。否则，保持为 0。
14	EXTMOD	0x1
13	WAITEN	0x0（对异步模式没有影响）
12	WREN	0x1
11	WAITCFG	根据需要进行设置
10	WRAPMOD	0x0
9	WAITPOL	仅当位 15 为 1 时才有意义
8	BURSTEN	0x0
7	保留	0x1
6	FACCEN	无关
5-4	MWID	根据需要进行设置
3-2	MTYP	根据需要进行设置，0x2 除外 (NOR Flash)
1	MUXEN	0x0
0	MBKEN	0x1

表 198. FSMC_BTRx 位字段

位号	位名	要设置的值
31:30	保留	0x0
29-28	ACCMOD	0x0
27-24	DATLAT	无关
23-20	CLKDIV	无关
19-16	BUSTURN	NEx 变为高电平到 NEx 变为低电平之间的时间 (BUSTURN HCLK)
15-8	DATAST	第二个访问阶段的持续时间（写入访问为 DATAST+1 个 HCLK 周期，读取访问为 DATAST 个 HCLK 周期）。
7-4	ADDHLD	无关
3-0	ADDSET	第一个访问阶段的持续时间（ADDSET 个 HCLK 周期）。 ADDSET 的最小值为 0。

表 199. FSMC_BWTRx 位字段

位号	位名	要设置的值
31:30	保留	0x0
29-28	ACCMOD	0x0
27-24	DATLAT	无关
23-20	CLKDIV	无关
19-16	BUSTURN	NEx 变为高电平到 NEx 变为低电平之间的时间 (BUSTURN HCLK)
15-8	DATAST	第二个访问阶段的持续时间（写入访问为 DATAST+1 个 HCLK 周期）
7-4	ADDHLD	无关
3-0	ADDSET	第一个访问阶段的持续时间（ADDSET 个 HCLK 周期）。 ADDSET 的最小值为 0。

模式 2/B - NOR Flash

图 409. 模式 2 和模式 B 读取访问

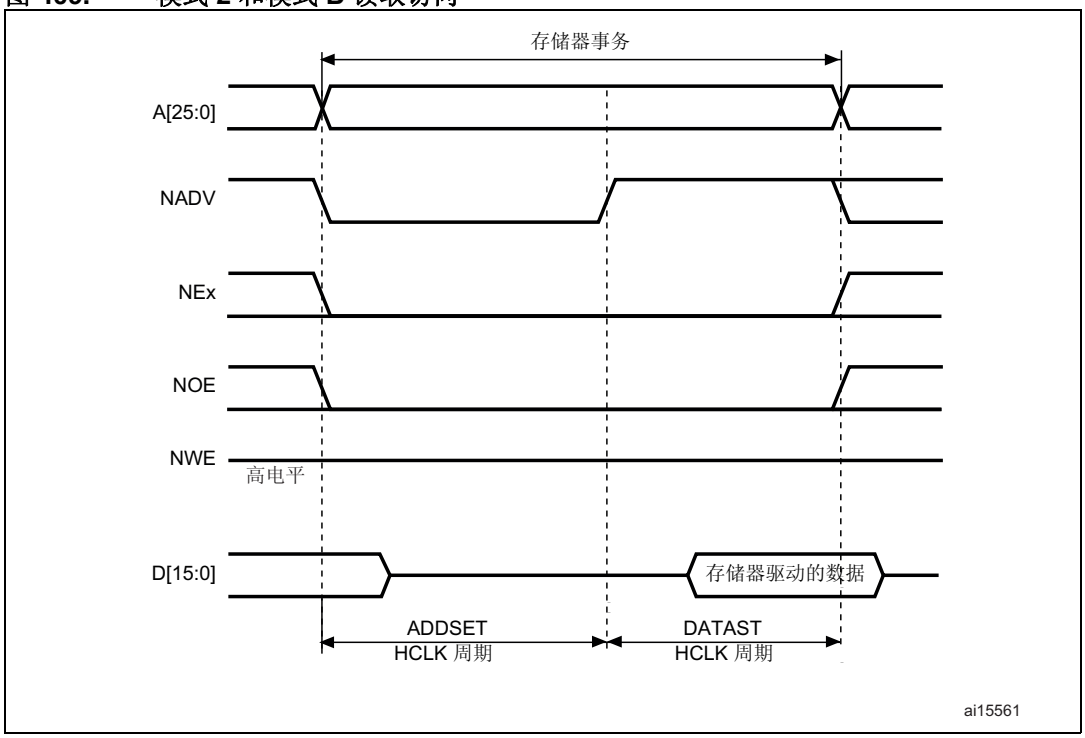


图 410. 模式 2 写入访问

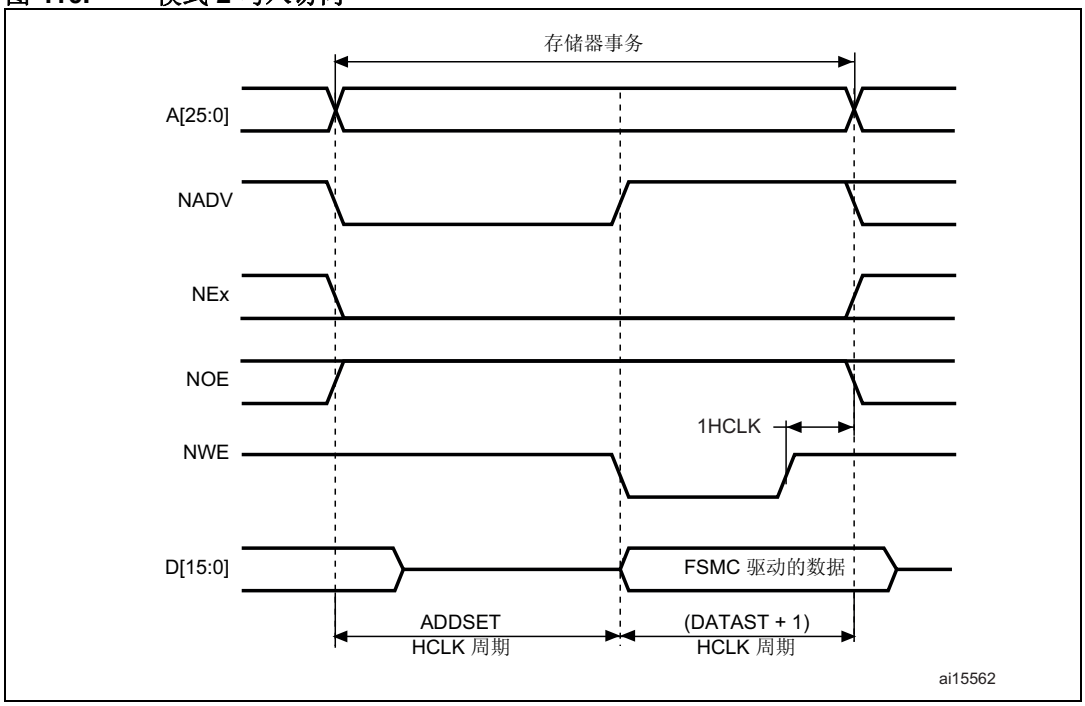
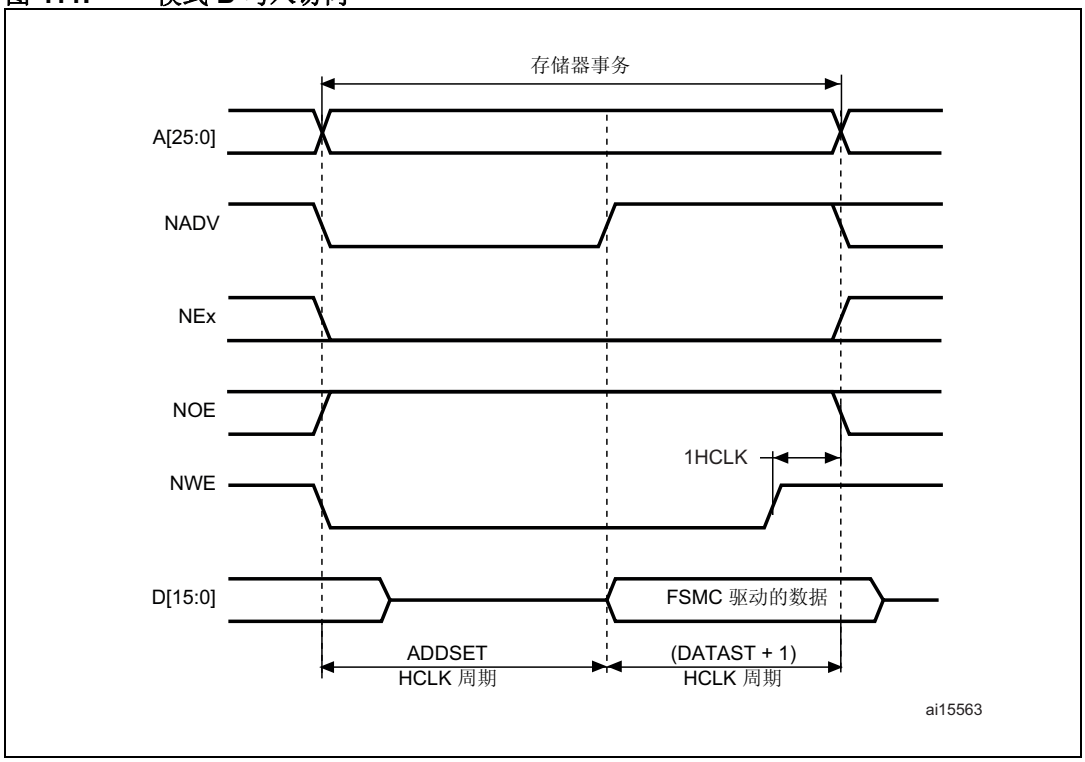


图 411. 模式 B 写入访问



与模式 1 的不同之处在于设置扩展模式（模式 B）时的 NWE 切换与独立的读取和写入时序。

表 200. FSMC_BCRx 位字段

位号	位名	要设置的值
31-20	保留	0x000
19	CBURSTRW	0x0（对异步模式没有影响）
18:16	保留	0x0
15	ASYNCWAIT	如果存储器支持该特性，则置为 1。否则，保持为 0。
14	EXTMOD	模式 B 为 0x1，模式 2 为 0x0
13	WAITEN	0x0（对异步模式没有影响）
12	WREN	0x1
11	WAITCFG	根据需要进行设置
10	WRAPMOD	0x0
9	WAITPOL	仅当位 15 为 1 时才有意义
8	BURSTEN	0x0
7	保留	0x1
6	FACCEN	0x1
5-4	MWID	根据需要进行设置
3-2	MTYP	0x2 (NOR Flash)
1	MUXEN	0x0
0	MBKEN	0x1

表 201. FSMC_BTRx 位字段

位号	位名	要设置的值
31:30	保留	0x0
29-28	ACCMOD	0x1
27-24	DATLAT	无关
23-20	CLKDIV	无关
19-16	BUSTURN	NEx 变为高电平到 NEx 变为低电平之间的时间 (BUSTURN HCLK)
15-8	DATAST	第二个访问阶段的持续时间（写入访问为 DATAST+1 个 HCLK 周期，读取访问为 DATAST 个 HCLK 周期）。
7-4	ADDHLD	无关
3-0	ADDSET	第一个访问阶段的持续时间（ADDSET 个 HCLK 周期）。 ADDSET 的最小值为 0。

表 202. FSMC_BWTRx 位字段

位号	位名	要设置的值
31:30	保留	0x0
29-28	ACCMOD	0x1
27-24	DATLAT	无关
23-20	CLKDIV	无关
19-16	BUSTURN	NEx 变为高电平到 NEx 变为低电平之间的时间 (BUSTURN HCLK)
15-8	DATAST	第二个访问阶段的持续时间（写入访问为 DATAST+1 个 HCLK 周期）
7-4	ADDHLD	无关
3-0	ADDSET	第一个访问阶段的持续时间（ADDSET 个 HCLK 周期）。 ADDSET 的最小值为 0。

注意： *仅当设置了扩展模式（模式 B）时，FSMC_BWTRx 寄存器才有效，否则其所有内容均为“无关”。*

模式 C - NOR Flash - OE 切换

图 412. 模式 C 读取访问

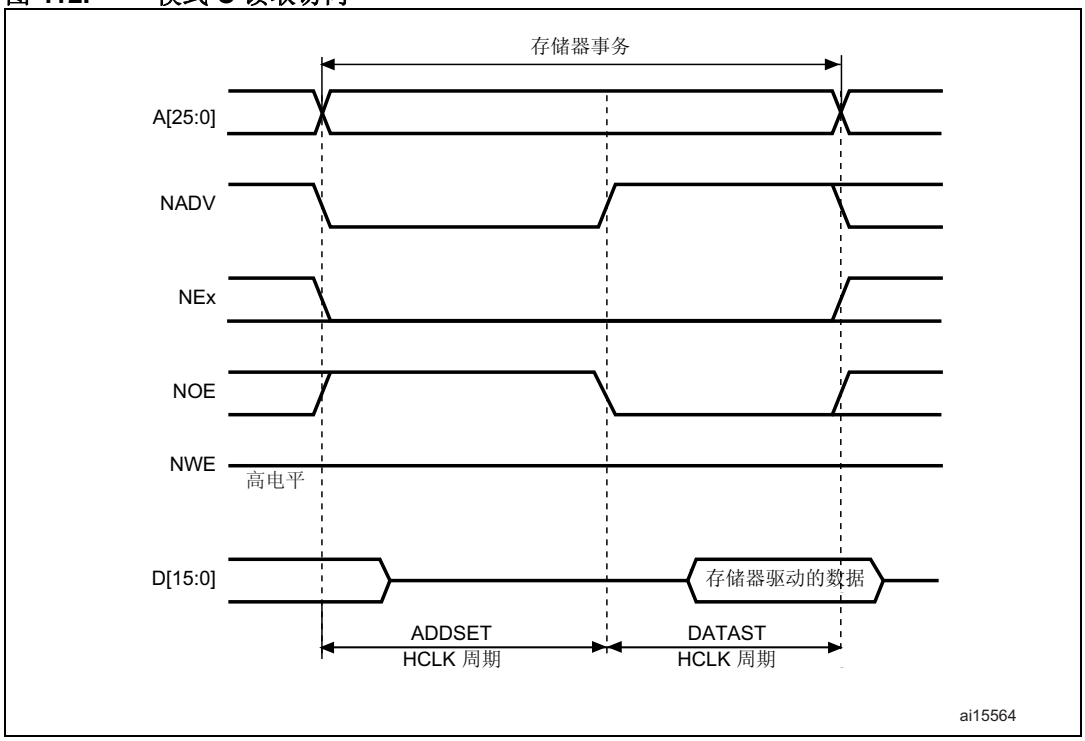
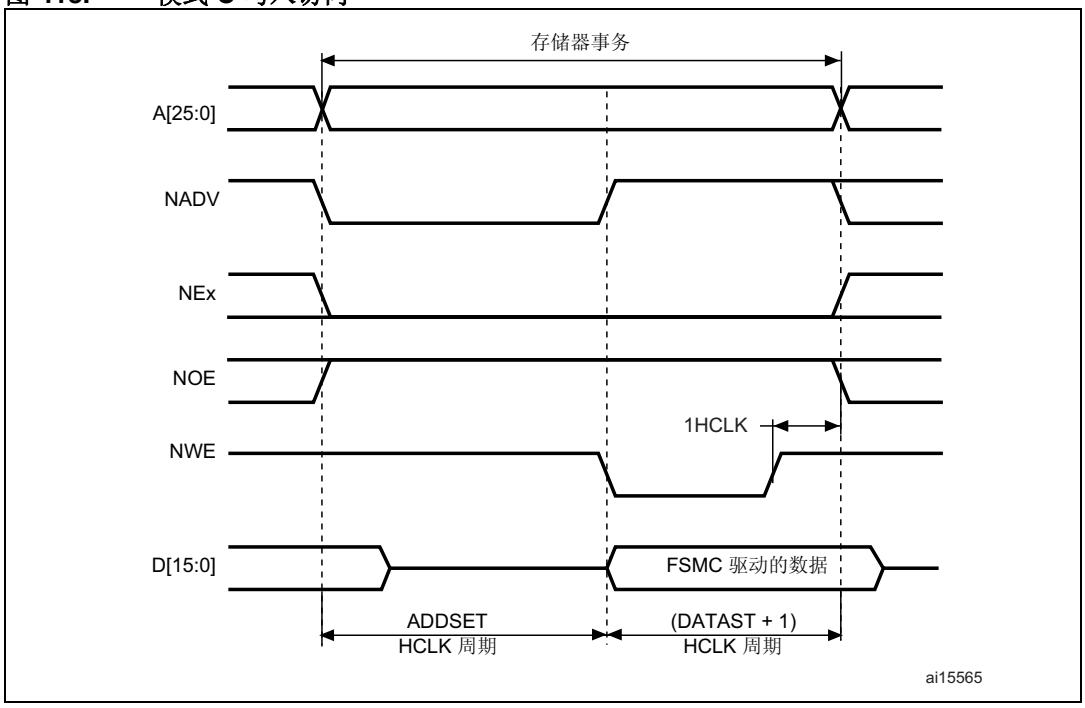


图 413. 模式 C 写入访问



与模式 1 的不同之处在于 NOE 的切换与独立的读取和写入时序。

表 203. FSMC_BCRx 位字段

位号	位名	要设置的值
31-20	保留	0x000
19	CBURSTRW	0x0（对异步模式没有影响）
18:16	保留	0x0
15	ASYNCWAIT	如果存储器支持该特性，则置为 1。否则，保持为 0。
14	EXTMOD	0x1
13	WAITEN	0x0（对异步模式没有影响）
12	WREN	0x1
11	WAITCFG	根据需要进行设置
10	WRAPMOD	0x0
9	WAITPOL	仅当位 15 为 1 时才有意义
8	BURSTEN	0x0
7	保留	0x1
6	FACCEN	0x1
5-4	MWID	根据需要进行设置
3-2	MTYP	0x2 (NOR Flash)
1	MUXEN	0x0
0	MBKEN	0x1

表 204. FSMC_BTRx 位字段

位号	位名	要设置的值
31:30	保留	0x0
29-28	ACCMOD	0x2
27-24	DATLAT	0x0
23-20	CLKDIV	0x0
19-16	BUSTURN	NEx 变为高电平到 NEx 变为低电平之间的时间 (BUSTURN HCLK)
15-8	DATAST	第二个访问阶段的持续时间（写入访问为 DATAST+1 个 HCLK 周期，读取访问为 DATAST 个 HCLK 周期）。
7-4	ADDHLD	无关
3-0	ADDSET	第一个访问阶段的持续时间（ADDSET 个 HCLK 周期）。 ADDSET 的最小值为 0。

表 205. FSMC_BWTRx 位字段

位号	位名	要设置的值
31:30	保留	0x0
29-28	ACCMOD	0x2
27-24	DATLAT	无关
23-20	CLKDIV	无关
19-16	BUSTURN	NEx 变为高电平到 NEx 变为低电平之间的时间 (BUSTURN HCLK)
15-8	DATAST	第二个访问阶段的持续时间（写入访问为 DATAST+1 个 HCLK 周期）
7-4	ADDHLD	无关
3-0	ADDSET	第一个访问阶段的持续时间（ADDSET 个 HCLK 周期）。 ADDSET 的最小值为 0。

模式 D - 扩展地址异步访问

图 414. 模式 D 读取访问

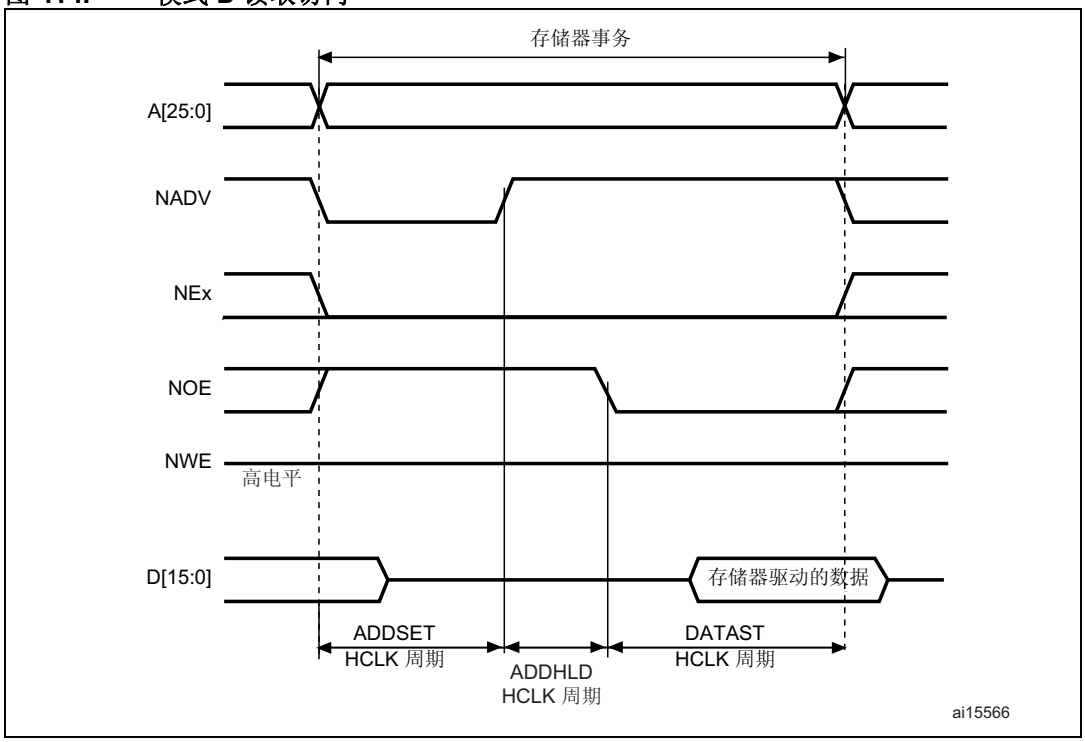
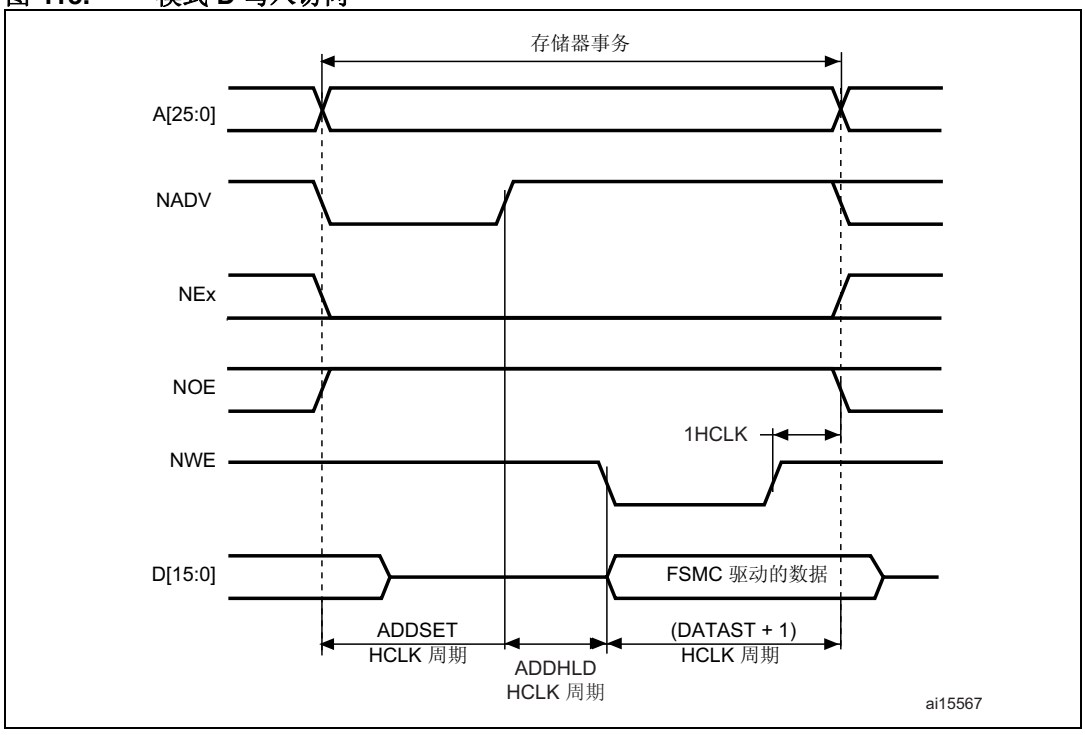


图 415. 模式 D 写入访问



与模式 1 的不同之处在于 NADV 变化后 NOE 的切换与独立的读取和写入时序。

表 206. FSMC_BCRx 位字段

位号	位名	要设置的值
31-20	保留	0x000
19	CBURSTRW	0x0（对异步模式没有影响）
18:16	保留	0x0
15	ASYNCWAIT	如果存储器支持该特性，则置为 1。否则，保持为 0。
14	EXTMOD	0x1
13	WAITEN	0x0（对异步模式没有影响）
12	WREN	0x1
11	WAITCFG	根据需要进行设置
10	WRAPMOD	0x0
9	WAITPOL	仅当位 15 为 1 时才有意义
8	BURSTEN	0x0
7	保留	0x1
6	FACCEN	根据存储器支持情况进行设置
5-4	MWID	根据需要进行设置
3-2	MTYP	根据需要进行设置
1	MUXEN	0x0
0	MBKEN	0x1

表 207. FSMC_BTRx 位字段

位号	位名	要设置的值
31:30	保留	0x0
29-28	ACCMOD	0x3
27-24	DATLAT	无关
23-20	CLKDIV	无关
19-16	BUSTURN	NEx 变为高电平到 NEx 变为低电平之间的时间 (BUSTURN HCLK)
15-8	DATAST	读取访问第二个阶段的持续时间（DATAST 个 HCLK 周期）。
7-4	ADDHLD	读取访问中间阶段的持续时间（ADDHLD 个 HCLK 周期）。
3-0	ADDSET	读取访问第一个阶段的持续时间（ADDSET 个 HCLK 周期）。 ADDSET 最小值为 0。

表 208. FSMC_BWTRx 位字段

位号	位名	要设置的值
31:30	保留	0x0
29-28	ACCMOD	0x3
27-24	DATLAT	0x0
23-20	CLKDIV	0x0
19-16	BUSTURN	NEx 变为高电平到 NEx 变为低电平之间的时间 (BUSTURN HCLK)
15-8	DATAST	写入访问第二个阶段的持续时间（DATAST+1 个 HCLK 周期）
7-4	ADDHLD	写入访问中间阶段的持续时间（ADDHLD 个 HCLK 周期）
3-0	ADDSET	写入访问第一个阶段的持续时间（ADDSET+1 个 HCLK 周期）。ADDSET 最小值为 0。

复用模式 - 复用异步访问 NOR Flash

图 416. 复用读取访问

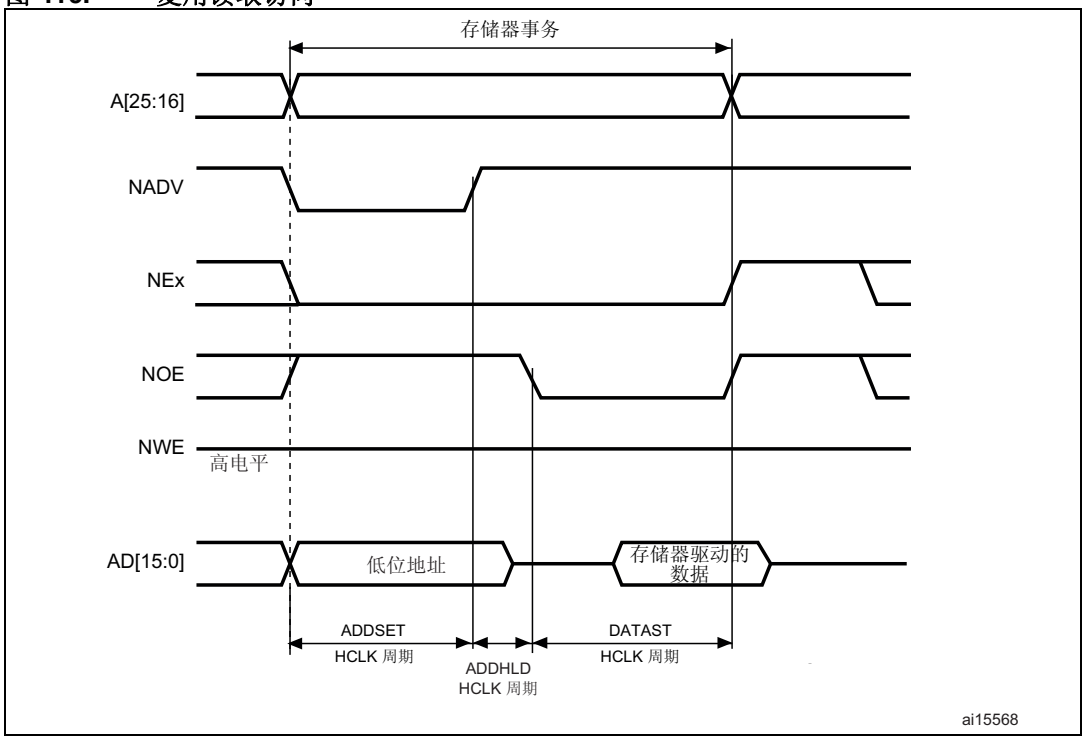
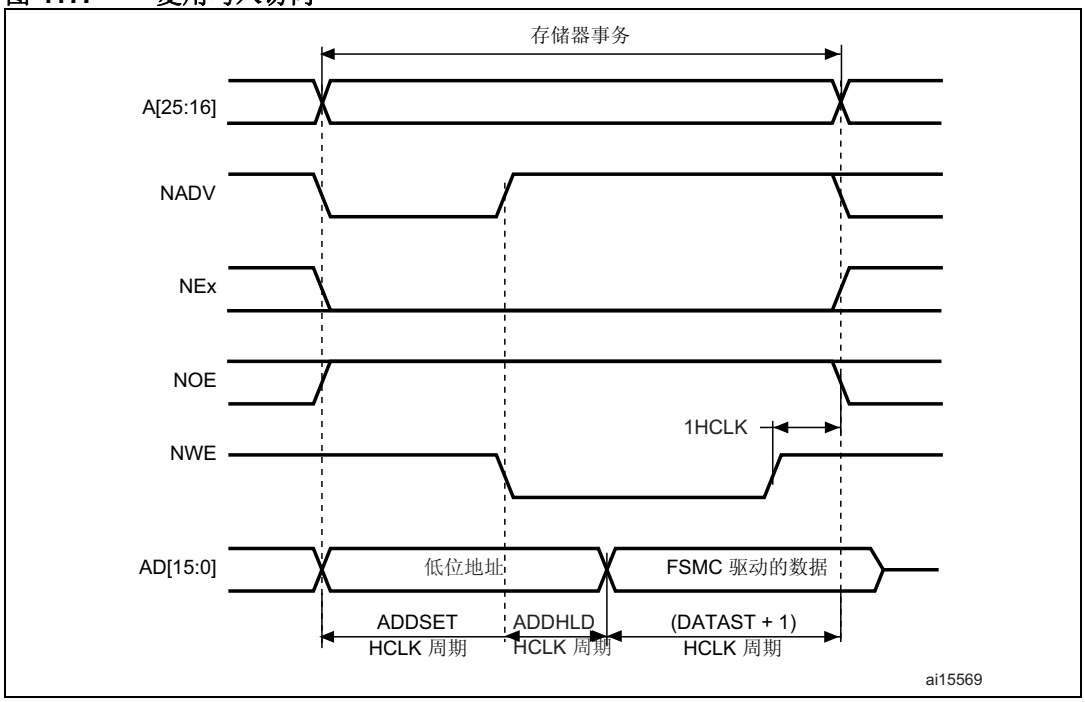


图 417. 复用写入访问



与模式 D 的不同之处在于在数据总线上驱动低地址字节。

表 209. FSMC_BCRx 位字段

位号	位名	要设置的值
31-21	保留	0x000
19	CBURSTRW	0x0（对异步模式没有影响）
18:16	保留	0x0
15	ASYNCWAIT	如果存储器支持该特性，则置为 1。否则，保持为 0。
14	EXTMOD	0x0
13	WAITEN	0x0（对异步模式没有影响）
12	WREN	0x1
11	WAITCFG	根据需要进行设置
10	WRAPMOD	0x0
9	WAITPOL	仅当位 15 为 1 时才有意义
8	BURSTEN	0x0
7	保留	0x1
6	FACCEN	0x1
5-4	MWID	根据需要进行设置
3-2	MTYP	0x2 (NOR Flash)
1	MUXEN	0x1
0	MBKEN	0x1

表 210. FSMC_BTRx 位字段

位号	位名	要设置的值
31:30	保留	0x0
29-28	ACCMOD	0x0
27-24	DATLAT	无关
23-20	CLKDIV	无关
19-16	BUSTURN	NEx 变为高电平到 NEx 变为低电平之间的时间 (BUSTURN HCLK)
15-8	DATAST	第二个访问阶段的持续时间（读取访问为 DATAST 个 HCLK 周期，写入访问为 DATAST+1 个 HCLK 周期）。
7-4	ADDHLD	访问中间阶段的持续时间（ADDHLD 个 HCLK 周期）。
3-0	ADDSET	第一个访问阶段的持续时间（ADDSET 个 HCLK 周期）。ADDSET 的最小值为 1。

异步访问中的 WAIT 管理

如果异步存储器发出 WAIT 信号，指示尚未准备好接受或提供数据，则 FSMC_BCRx 寄存器中的 ASYNCWAIT 位必须置 1。

如果 WAIT 信号处于有效状态（电平高低取决于 WAITPOL 位），则由 DATAST 位控制的第二个访问阶段（数据建立阶段）将延长，直到 WAIT 变为无效状态。与数据建立阶段不同，由 ADDSET 和 ADDHLD 位控制的第一个访问阶段（地址建立和地址保持阶段）对 WAIT 不敏感，因此第一个访问阶段不会延长。

必须配置数据建立阶段（FSMC_BTRx 寄存器中的 DATAST），以便在存储器事务结束前 4 个 HCLK 周期检测到 WAIT。必须考虑以下情况：

1. 存储器发出的 WAIT 信号和 NOE/NWE 信号对齐：

$$DATAST \geq (4 \times HCLK) + max\_wait\_assertion\_time$$

2. 存储器发出的 WAIT 信号和 NEx 对齐（或者 NOE/NWE 信号不翻转）：

   如果

$$max\_wait\_assertion\_time > address\_phase + hold\_phase$$

   那么

$$DATAST \geq (4 \times HCLK) + (max\_wait\_assertion\_time - address\_phase - hold\_phase)$$

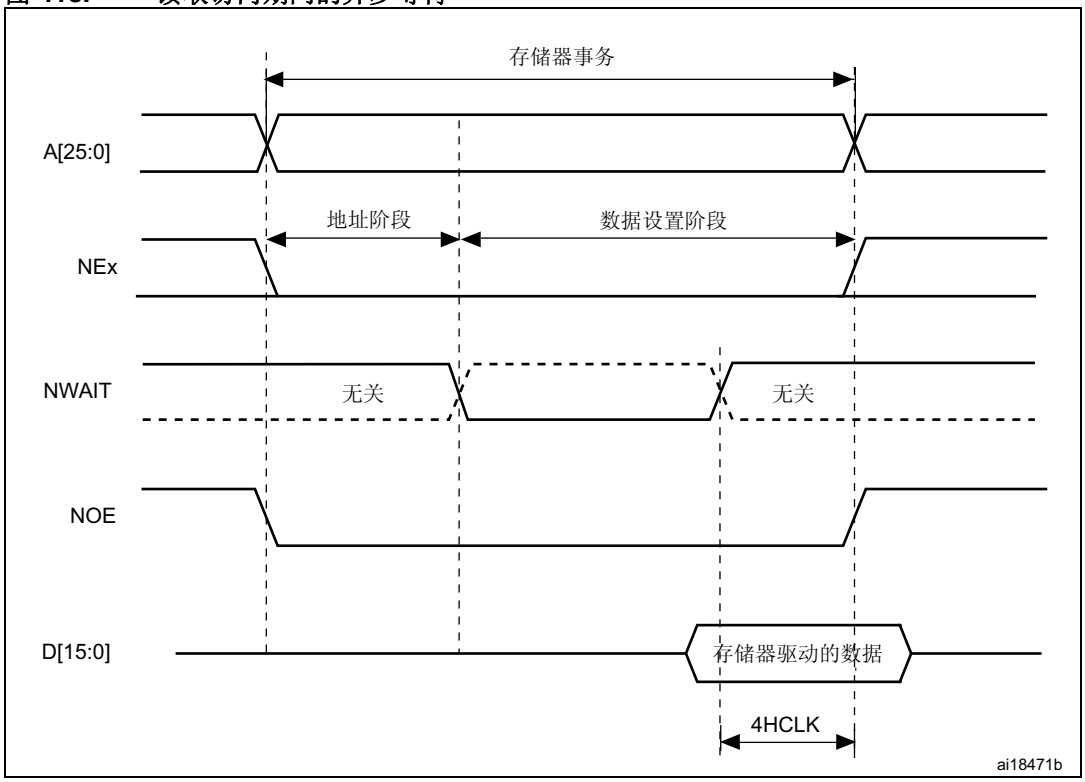
   否则

$$DATAST \geq 4 \times HCLK$$

   其中，max_wait_assertion_time 是在 NEx/NOE/NWE 变为低电平后存储器使能 WAIT 信号所花费的最长时间。

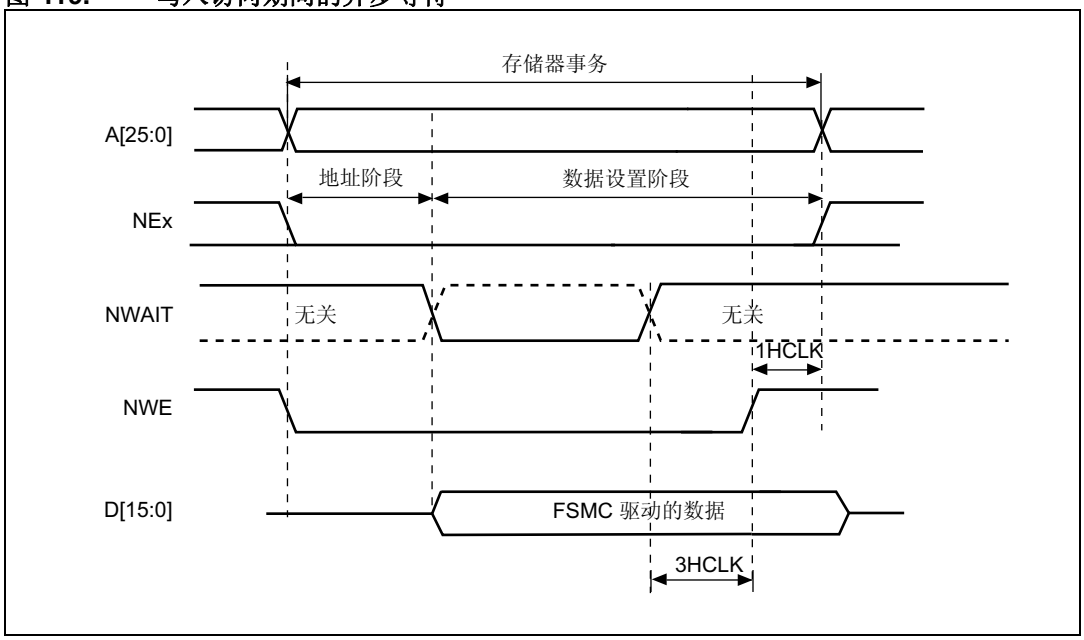
图 418 和 *图 419* 显示了异步存储器释放 WAIT 之后，在存储器访问过程中增加的 HCLK 时钟周期的个数（与上述情况无关）。

图 418. 读取访问期间的异步等待



1. NWAIT 极性取决于 FSMC_BCRx 寄存器中的 WAITPOL 位设置。

图 419. 写入访问期间的异步等待



1. NWAIT 极性取决于 FSMC_BCRx 寄存器中的 WAITPOL 位设置。

32.5.5 同步突发事务

存储器时钟 CLK 是 HCLK 的一个因数，因数大小取决于参数 CLKDIV 的值。

NOR Flash 指定了从 NADV 使能到 CLK 高电平的最短时间。为了符合这一限制，FSMC 不会在同步访问的第一个内部时钟周期内（NADV 使能之前）将时钟发到存储器。这样可以确保存储器时钟的上升沿出现在 NADV 低脉冲的*中间*。

数据延迟与 NOR Flash 延迟

数据延迟是对数据进行采样之前需要等待的周期数。DATLAT 的值必须与 NOR FLASH 配置寄存器中指定的延迟值一致。当数据延迟计数中 NADV 为低电平时，FSMC 不会计入时钟周期。

小心： 一些 NOR Flash 将 NADV 低电平周期计入数据延迟计数，这样 NOR Flash 延迟和 FMSC DATLAT 参数之间的确切关系可以是以下任一种：

- NOR Flash 延迟 = DATLAT + 2
- NOR Flash 延迟 = DATLAT + 3

近来有一些存储器会在延迟阶段使能 NWAIT。在这种情况下，可以将 DATLAT 设置为最小值。然后，FSMC 会对数据进行采样，并且等待足够长的时间来评估数据是否有效。这样，FSMC 就能检测到存储器存在延迟的时间，从而使用真实数据。

其他存储器不会在延迟期间使能 NWAIT。在这种情况下，必须正确设置 FSMC 和存储器的延迟，否则可能会将无效数据误用为有效数据，或者在存储器访问初始阶段丢失有效数据。

单次突发传输

当所选存储区域配置为同步突发模式时，如果请求了一个 AHB 单次突发事务，则 FSMC 会执行长度为 1 的突发事务（如果 AHB 传输为 16 位）或者长度为 2 的突发事务（如果 AHB 传输为 32 位），然后在最后一个数据选通时禁止片选信号。

很明显，这样的传输就周期而言不是最有效率的（与异步读取相比）。但是，随机异步读取需要先重新编程存储器访问模式，这样总时间会更长。

等待管理

对于同步突发 NOR Flash，会在配置过的延迟周期（(DATALAT+2) 个 CLK 时钟周期）之后对 NWAIT 进行评估。

如果检测到 NWAIT 有效（WAITPOL = 0 时为低电平，WAITPOL = 1 时为高电平），会插入等待状态，直到检测到 NWAIT 无效（WAITPOL = 0 时为高电平，WAITPOL = 1 时为低电平）。

当 NWAIT 无效时，数据将立即（位 WAITCFG = 1）或在下一个时钟边沿（位 WAITCFG = 0）被视为有效。

通过 NWAIT 信号插入等待周期期间，控制器会继续将时钟脉冲发送到存储器，保持片选和输出使能信号有效，同时不将数据视为有效。

在突发模式下，NOR Flash NWAIT 信号有两种时序配置：

- Flash 在等待周期之前一个数据周期使能 NWAIT（复位后的默认值）
- Flash 在等待周期期间使能 NWAIT

FSMC 支持这两种 NOR Flash 等待周期配置，通过 FSMC_BCRx 寄存器中的 WAITCFG 位 (x = 0..3) 针对每个片选单独进行配置。

图 420. 等待配置

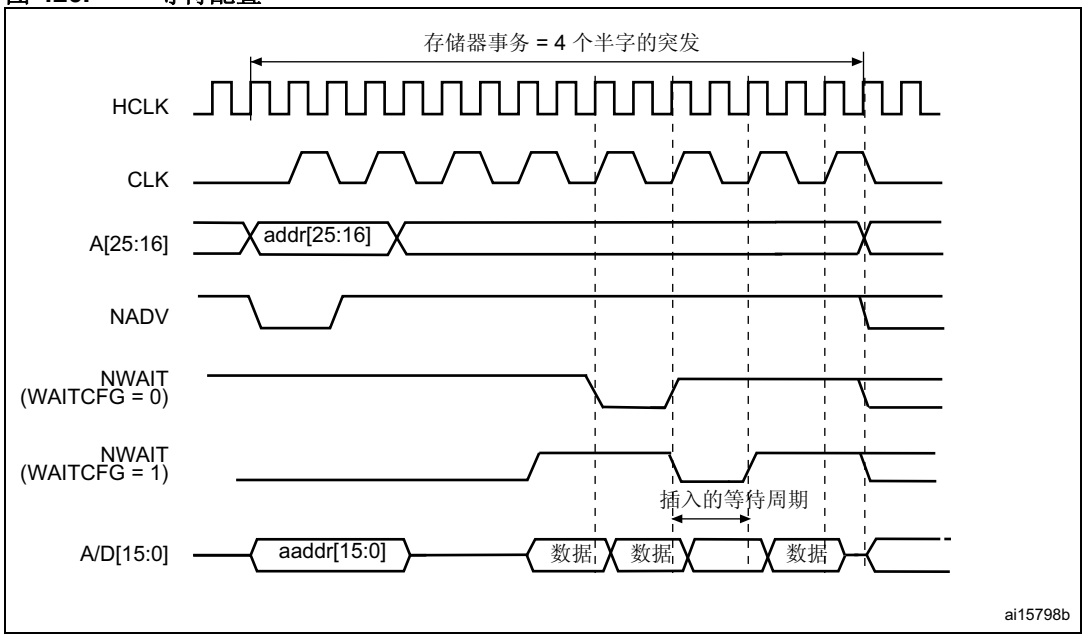
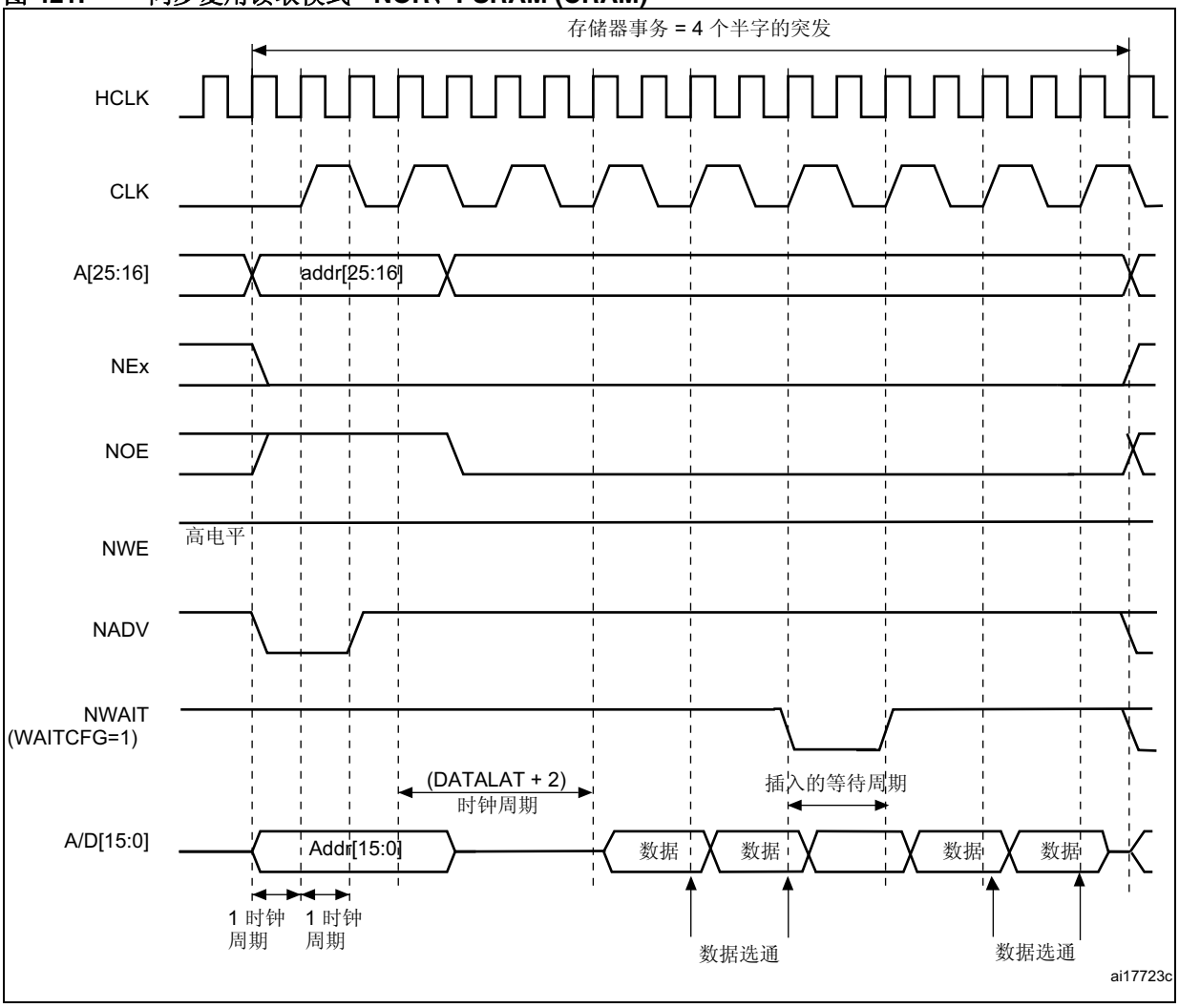


图 421. 同步复用读取模式 - NOR、PSRAM (CRAM)



1. 字节通道输出 BL 未显示，它们对于 NOR 访问保持高电平，对于 PSRAM (CRAM) 访问则保持低电平。
2. NWAIT 极性设置为 0。

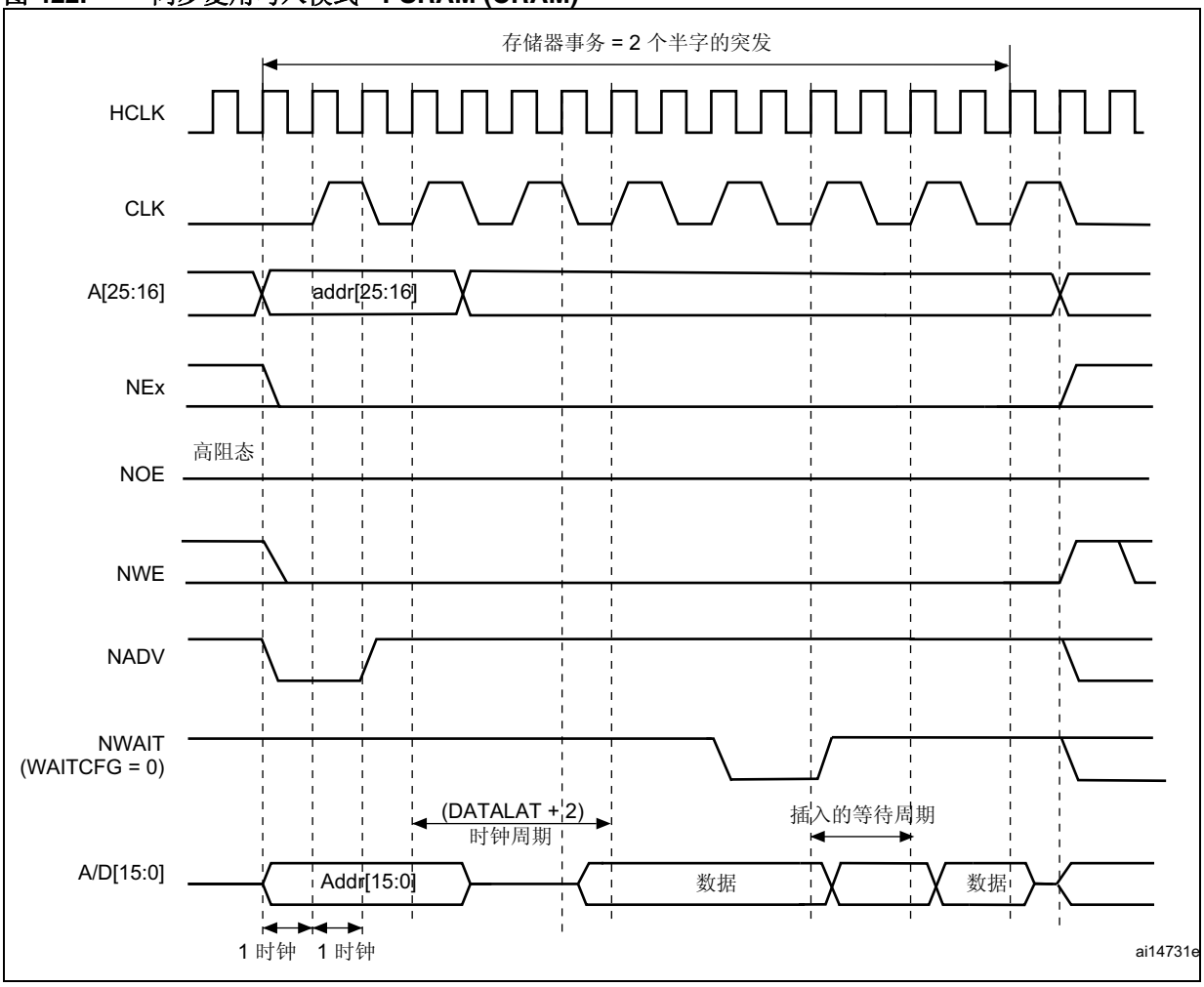
表 211. FSMC_BCRx 位字段

位号	位名	要设置的值
31-20	保留	0x000
19	CBURSTRW	对同步读取没有影响
18-16	保留	0x0
15	ASCYCWAIT	0x0
14	EXTMOD	0x0
13	WAITEN	在存储器支持该特性的情况下置位为 1，否则保持为 0。
12	WREN	对同步读取没有影响
11	WAITCFG	是否置位视存储器情况而定
10	WRAPMOD	0x0
9	WAITPOL	是否置位视存储器情况而定
8	BURSTEN	0x1
7	保留	0x1
6	FACCEN	在存储器支持的情况下置位 (NOR Flash)
5-4	MWID	根据需要进行设置
3-2	MTYP	0x1 或 0x2
1	MUXEN	根据需要进行设置
0	MBKEN	0x1

表 212. FSMC_BTRx 位字段

位号	位名	要设置的值
31:30	保留	0x0
29:28	ACCMOD	0x0
27-24	DATLAT	数据延迟
23-20	CLKDIV	0x0，使 CLK = HCLK（不支持） 0x1，使 CLK = 2 × HCLK ..
19-16	BUSTURN	NEx 变为高电平到 NEx 变为低电平之间的时间 (BUSTURN HCLK)
15-8	DATAST	无关
7-4	ADDHLD	无关
3-0	ADDSET	无关

图 422. 同步复用写入模式 - PSRAM (CRAM)



1. 存储器必须提前一个周期发出 NWAIT 信号，相应地 WAITCFG 必须编程为 0。
2. NWAIT 极性设置为 0。
3. 字节通道 (NBL) 输出未显示，当 NEx 有效时它们保持低电平。

表 213. FSMC_BCRx 位字段

位号	位名	要设置的值
31-20	保留	0x000
19	CBURSTRW	0x1
18-16	保留	0x0
15	ASCYCWAIT	0x0
14	EXTMOD	0x0
13	WAITEN	在存储器支持该特性的情况下置位为 1，否则保持为 0。
12	WREN	对同步读取没有影响
11	WAITCFG	0x0
10	WRAPMOD	0x0

表 213. FSMC_BCRx 位字段 （续）

位号	位名	要设置的值
9	WAITPOL	是否置位视存储器情况而定
8	BURSTEN	对同步写入没有影响
7	保留	0x1
6	FACCEN	根据存储器支持情况进行设置
5-4	MWID	根据需要进行设置
3-2	MTYP	0x1
1	MUXEN	根据需要进行设置
0	MBKEN	0x1

表 214. FSMC_BTRx 位字段

位号	位名	要设置的值
31:30	保留	0x0
29:28	ACCMOD	0x0
27-24	DATLAT	数据延迟
23-20	CLKDIV	0x0，使 CLK = HCLK（不支持） 0x1，使 CLK = 2 × HCLK
19-16	BUSTURN	无关
15-8	DATAST	无关
7-4	ADDHLD	无关
3-0	ADDSET	无关

32.5.6 NOR/PSRAM 控制寄存器

NOR/PSRAM 控制寄存器必须按字（32 位）进行访问。

SRAM/NOR-Flash 片选控制寄存器 1..4 (FSMC_BCR1..4)

SRAM/NOR-Flash chip-select control registers 1..4

偏移地址：0xA000 0000 + 8 * (x – 1)，x = 1...4

复位值：0x0000 30DX

该寄存器包含每个存储区域的控制信息，用于 SRAM、ROM 和异步或突发 NOR Flash。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												CBURSTRW	Reserved			ASCYCWAIT	EXTMOD	WAITEN	WREN	WAITCFG	WRAPMOD	WAITPOL	BURSTEN	Reserved	FACCEN	MWID		MTYP		MUXEN	MBKEN
												rw				rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw

位 31：20 保留，必须保持复位值。

位 19 **CBURSTRW**：写入突发使能 (Write burst enable)。

对于 Cellular RAM (PSRAM)，该位可在写操作时使能同步突发协议。读取访问期间同步突发协议的使能位为 FSMC_BCRx 寄存器中的 BURSTEN 位。

0：始终在异步模式下进行写入操作

1：在同步模式下进行写入操作。

位 18：16 保留，必须保持复位值。

位 15 **ASYNCWAIT**：异步传输期间的等待信号 (Wait signal during asynchronous transfers)

该位可使能/禁止 FSMC 使用等待信号，即使是在异步协议期间该位也有作用。

0：运行异步协议时不考虑 NWAIT 信号（复位后的默认值）

1：运行异步协议时考虑 NWAIT 信号

位 14 **EXTMOD**：扩展模式使能 (Extended mode enable)。

FSMC 可对 FSMC_BWTR 寄存器中的写入时间进行配置，此配置由 EXTMOD 位使能，进而使读取和写入操作采用不同时序。

0：不考虑 FSMC_BWTR 寄存器中的值（复位后的默认值）

1：考虑 FSMC_BWTR 寄存器中的值

位 13 **WAITEN**：等待使能位 (Wait enable bit)。

该位可使能/禁止在同步模式下访问 Flash 时通过 NWAIT 信号插入等待周期。

0：禁止 NWAIT 信号（不考虑其电平，不在配置过的 Flash 延迟周期后插入等待周期）

1：使能 NWAIT 信号（考虑其电平，如果使能，在配置过的 Flash 延迟周期后插入等待周期）（复位后的默认值）

位 12 **WREN**：写入使能位 (Write enable bit)。

该位指示 FSMC 是否在存储区域内使能/禁止了写入操作：

0：FSMC 在存储区域内禁止了写入操作，如果进行写操作将报告 AHB 错误，

1：FSMC 在存储区域内使能了写入操作（复位后的默认值）。

位 11 **WAITCFG**：等待时序配置 (Wait timing configuration)。

NWAIT 信号指示存储器中的数据是否有效，或者在同步模式下访问 Flash 时是否必须插入等待周期。该配置位决定存储器是在等待周期之前的一个时钟周期还是等待周期期间使能 NWAIT：

0：NWAIT 信号在等待周期之前的一个数据周期有效（复位后的默认值），

1：NWAIT 信号在等待周期期间有效（不适用于 Cellular RAM）。

位 10 **WRAPMOD**：环回突发模式支持 (Wrapped burst mode support)。

定义控制器是否会将一个 AHB 突发环回访问分割成为两个线性访问。仅在突发模式下访问存储器时有效

0：未使能直接环回突发（复位后的默认值），

1：使能直接环回突发。

注意：由于 CPU 和 DMA 无法生成环回突发传输，因此该位无效。

位 9 **WAITPOL**：等待信号极性位 (Wait signal polarity bit)。

定义存储器的等待信号极性。仅在突发模式下访问存储器时有效：

0：NWAIT 低电平有效（复位后的默认值），

1：NWAIT 高电平有效。

位 8 **BURSTEN**：突发使能位 (Burst enable bit)。

该位可使能/禁止读取操作期间的同步突发访问。仅对同步突发存储器有效：

0：禁止突发访问模式（复位后的默认值）

1：使能突发访问模式

位 7 保留，必须保持复位值。

位 6 **FACCEN**：Flash 访问使能 (Flash access enable)

使能 NOR Flash 访问操作。

0：禁止相应的 NOR Flash 访问

1：使能相应的 NOR Flash 访问（复位后的默认值）

位 5:4 **MWID**：存储器数据总线宽度 (Memory databus width)。

定义外部存储器器件宽度，对所有类型的存储器均有效。

00：8 位，

01：16 位（复位后的默认值），

10：保留，不使用，

11：保留，不使用。

位 3:2 **MTYP**：存储器类型 (Memory type)。

定义与相应存储区域相连的外部存储器类型：

00：SRAM、ROM（对于存储区域 2...4，复位后的默认值）

01：PSRAM（Cellular RAM：CRAM）

10：NOR Flash/OneNAND Flash （对于存储区域 1，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)。

该位置 1 时，地址和数据值在数据总线上复用，仅对 NOR 和 PSRAM 存储器有效：

0：地址/数据非复用

1：地址/数据在数据总线上复用（复位后的默认值）

位 0 **MBKEN**：存储区域使能位 (Memory bank enable bit)。

使能存储区域。复位后使能存储区域 1，其它存储区域均禁止。访问禁止的存储区域会引起 AHB 总线上的错误。

0：禁止相应的存储区域

1：使能相应的存储区域

SRAM/NOR-Flash 片选时序寄存器 1..4 (FSMC_BTR1..4)

SRAM/NOR-Flash chip-select timing registers 1..4

偏移地址：0xA000 0000 + 0x04 + 8 * (x – 1)，x = 1..4

复位值：0x0FFF FFFF

该寄存器包含每个存储区域的控制信息，用于 SRAM、ROM 和 NOR Flash。如果 FSMC_BCRx 寄存器中的 EXTMOD 位置 1，该寄存器将和另外一个寄存器配合来配置写入和读取的时序参数。也就是说有 2 个寄存器可用：一个用于配置读取访问（此寄存器），另一个用于配置写入访问（FSMC_BWTRx 寄存器）。

31 30	29 28	27 26 25 24	23 22 21 20	19 18 17 16	15 14 13 12 11 10 9 8	7 6 5 4	3 2 1 0
Reserved	ACCMOD	DATLAT	CLKDIV	BUSTURN	DATAST	ADDHLD	ADDSET
	rw rw	rw rw rw rw	rw rw rw rw	rw rw rw rw	rw rw rw rw rw rw rw rw	rw rw rw rw	rw rw rw rw

位 31:30 保留，必须保持复位值。

位 29:28 **ACCMOD**：访问模式 (Access mode)

指定异步访问模式，如时序图所示。仅当 FSMC_BCRx 寄存器中的 ECTMOD 位为 1 时才考虑这些位。

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**：同步突发 NOR Flash 的数据延迟 (Data latency for synchronous burst NOR Flash memory)

对于使能了同步突发模式的 NOR Flash，该位定义了获取首个数据前要发送给存储器的存储器时钟周期数 (+2)：

0000：首次突发访问时，2 个 CLK 时钟周期的数据延迟
1111：首次突发访问时，17 个 CLK 时钟周期的数据延迟（复位后的默认值）

注意：该时序参数以 Flash 时钟 (CLK) 周期而非 HCLK 周期表示。在异步 NOR Flash、SRAM 或 ROM 访问模式下，该值为无关值。如果是 CRAM，该字段必须设置为“0”。

位 23:20 **CLKDIV**：CLK 信号的时钟分频比 (Clock divide ratio (for CLK signal))

定义 CLK 时钟输出信号的周期，以 HCLK 周期数表示：

0000：Reserved
0001：CLK 周期 = 2 × HCLK 周期
0010：CLK 周期 = 3 × HCLK 周期
1111：CLK 周期 = 16 × HCLK 周期（复位后的默认值）

在异步 NOR Flash、SRAM 或 ROM 访问模式下，该值为无关值。

位 19:16 **BUSTURN**：总线周转阶段的持续时间 (Bus turnaround phase duration)

通过软件写入这些位可在写入/读取事务的结尾添加延迟。该延迟可以匹配连续事务之间的最短时间（t_{EHEL} 由 NEx 高电平变为 NEx 低电平）以及存储器在读取访问后释放数据总线所需的最长时间 (tEHQZ)：

(BUSTRUN + 1)HCLK 周期 ≥ $t_{EHELmin}$ 和 (BUSTRUN + 2)HCLK 周期 ≥ $t_{EHQZmax}$（如果 EXTMOD = “0”）

(BUSTRUN + 2)HCLK 周期 ≥ max ($t_{EHELmin}$，$t_{EHQZmax}$)（如果 EXTMOD = “1”）。

0000：BUSTURN 阶段的持续时间 = 增加 0 个 HCLK 时钟周期
...
1111：BUSTURN 阶段的持续时间 = 15 × HCLK 时钟周期（复位后的默认值）

位 15:8 **DATAST**：数据阶段的持续时间 (Data-phase duration)

通过软件写入这些位可定义数据阶段的持续时间（请参见*图 405* 到*图 417*），适用于 SRAM、ROM 和异步 NOR Flash 访问模式：

0000 0000：Reserved
0000 0001：DATAST 阶段的持续时间 = 1 × HCLK 时钟周期
0000 0010：DATAST 阶段的持续时间 = 2 × HCLK 时钟周期
...
1111 1111：DATAST 阶段的持续时间 = 255 × HCLK 时钟周期（复位后的默认值）

有关每种存储器类型和访问模式数据阶段持续时间的信息，请参见相应图片（*图 405* 到*图 417*）。

示例：模式 1，写入访问，DATAST=1：数据阶段的持续时间 = DATAST+1 = 2 个 HCLK 时钟周期。

注意：在同步访问模式下，该值为无关值。

位 7:4 **ADDHLD**：地址保持阶段的持续时间 (Address-hold phase duration)

通过软件写入这些位可定义*地址保持*阶段的持续时间（请参见*图 414* 到*图 417*），适用于模式 D 和复用访问：

0000：Reserved

0001：ADDHLD 阶段的持续时间 = 1 × HCLK 时钟周期

0010：ADDHLD 阶段的持续时间 = 2 × HCLK 时钟周期

...

1111：ADDHLD 阶段的持续时间 = 15 × HCLK 时钟周期（复位后的默认值）

有关每种访问模式地址保持阶段持续时间的信息，请参见相应图片（*图 414* 到*图 417*）。

注意：在同步访问模式下，该值不使用，因为地址保持阶段的持续时间始终是 1 个存储器时钟周期。

位 3:0 **ADDSET**：地址设置阶段的持续时间 (Address setup phase duration)

通过软件写入这些位可定义*地址设置*阶段的持续时间（请参见*图 405* 到*图 417*），适用于 SRAM、ROM 和异步 NOR Flash 访问模式：

0000：ADDSET 阶段的持续时间 = 0 × HCLK 时钟周期

...

1111：ADDSET 阶段的持续时间 = 1615 × HCLK 时钟周期（复位后的默认值）

有关每种访问模式地址设置阶段持续时间的信息，请参见相应图片（请参见*图 405* 到*图 417*）。

注意：在同步访问模式下，该值为无关值。

注意： *PSRAM (CRAM) 由于内部刷新而导致数据延时时间长度不确定。因此，这些存储器会在整个延迟阶段发送 NWAIT 信号，以便按照需要延长延迟。*
对于 PSRAM (CRAM)，字段 DATLAT 必须设置为 0，这样 FSMC 会立即退出延迟阶段，开始对存储器中的 NWAIT 采样，然后在存储器准备就绪后开始读取或写入。
此方法也适用于最新一代的同步 Flash，同早期 Flash 不同的是，此类 Flash 会发送 NWAIT 信号（检查所用的具体 Flash 数据表）。

SRAM/NOR-Flash 写入时序寄存器 1..4 (FSMC_BWTR1..4)

SRAM/NOR-Flash write timing registers 1..4

偏移地址：0xA000 0000 + 0x104 + 8 * (x – 1)，x = 1...4

复位值：0x0FFF FFFF

该寄存器包含每个存储区域的控制信息，用于 SRAM、ROM 和 NOR Flash。当 FSMC_BCRx 寄存器中的 EXTMOD 位置 1 时，该寄存器将处于有效状态，可以进行写入访问。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.		ACCMOD		DATLAT				CLKDIV				BUSTURN				DATAST								ADDHLD				ADDSET			
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:30 保留，必须保持复位值。

位 29:28 **ACCMOD**：访问模式 (Access mode)。

指定异步访问模式，如下一个时序图所示。仅当 FSMC_BCRx 寄存器中的 EXTMOD 位置 1 时，这些位才有效。

访问模式 A

访问模式 B

访问模式 C

访问模式 D

位 27:24 **DATLAT**：数据延迟（用于同步突发 NOR Flash）(Data latency (for synchronous burst NOR Flash))。

对于使能了同步突发模式的 NOR Flash，该位定义了获取首个数据前要发送给存储器的存储器时钟周期数 (+2)：

0000：(0x0) 首次突发访问时，2 个 CLK 时钟周期的数据延迟

...

1111：(0xF) 首次突发访问时，17 个 CLK 时钟周期的数据延迟（复位后的默认值）

*注意：该时序参数以 Flash 时钟 (**CLK**) 周期而非 HCLK 周期表示。在异步 NOR Flash、SRAM 或 ROM 访问模式下，该值为无关值。如果是 CRAM，则该字段必须置 0*

位 23:20 **CLKDIV**：CLK 信号的时钟分频比 (Clock divide ratio (for CLK signal))。

定义 CLK 时钟输出信号的周期，以 HCLK 周期数表示：

0000：保留

0001 CLK 周期 = 2 × HCLK 周期

0010 CLK 周期 = 3 × HCLK 周期

1111：CLK 周期 = 16 × HCLK 周期（复位后的默认值）

在异步 NOR Flash、SRAM 或 ROM 访问模式下，该值为无关值。

位 19:16 **BUSTURN**：总线周转阶段的持续时间 (Bus turnaround phase duration)

通过软件写入这些位可在写入事务结束时增加一个延迟，进而匹配两个连续事务之间的最小时间（t_{EHEL} 由 ENx 高电平变为 ENx 低电平）：

(BUSTRUN + 1) HCLK 周期 ≥ $t_{EHELmin}$。

0000：BUSTURN 阶段的持续时间 = 增加 0 个 HCLK 时钟周期

...

1111：BUSTURN 阶段的持续时间 = 增加 15 个 HCLK 时钟周期（复位后的默认值）

位 15:8 **DATAST**：数据阶段的持续时间 (Data-phase duration)。

通过软件写入这些位可定义数据阶段的持续时间（请参见*图 405* 到*图 417*），适用于 SRAM、ROM 和异步 NOR Flash 访问模式：

0000 0000：保留

0000 0001:DATAST 阶段的持续时间 = 1 × HCLK 时钟周期

0000 0010:DATAST 阶段的持续时间 = 2 × HCLK 时钟周期

...

1111 1111:DATAST 阶段的持续时间 = 255 × HCLK 时钟周期（复位后的默认值）

注意：在同步访问模式下，该值为无关值。

位 7:4 **ADDHLD**：地址保持阶段的持续时间 (Address-hold phase duration)。

通过软件写入这些位可定义*地址保持*阶段的持续时间（请参见*图 414* 到*图 417*），适用于 SRAM、ROM 和异步复用 NOR Flash 访问模式：

0000：保留

0001：ADDHLD 阶段的持续时间 = 1 × HCLK 时钟周期

0010：ADDHLD 阶段的持续时间 = 2 × HCLK 时钟周期

...

1111：ADDHLD 阶段的持续时间 = 15 × HCLK 时钟周期（复位后的默认值）

注意：在同步 NOR Flash 访问模式下，该值不使用，因为地址保持阶段的持续时间始终是 1 个 Flash 时钟周期。

位 3:0 **ADDSET**：地址设置阶段的持续时间 (Address setup phase duration)。

通过软件写入这些位可定义以 HCLK 周期表示的*地址设置*阶段的持续时间（请参见*图 414* 到*图 417*），适用于 SRAM、ROM 和异步 NOR Flash 访问模式：

0000：ADDSET 阶段的持续时间 = 0 × HCLK 时钟周期

...

1111：ADDSET 阶段的持续时间 = 15 × HCLK 时钟周期（复位后的默认值）

注意：在同步 NOR Flash 访问模式下，该值为无关值。

32.6 NAND Flash/PC 卡控制器

FSMC 会生成相应的信号时序，用于驱动以下类型的设备：

- NAND Flash
  - 8 位
  - 16 位
- 16 位 PC 卡兼容设备

NAND/PC 卡控制器可以控制三个外部存储区域。存储区域 2 和存储区域 3 支持 NAND Flash 设备。存储区域 4 支持 PC 卡设备。

每个存储区域都通过专用的寄存器配置（请参见*第 32.6.8 节*）。可编程的存储器参数包括访问时序（如*表 215* 所示）和 ECC 配置。

表 215. 可编程的 NAND/PC 卡访问参数

参数	功能	访问模式	单位	最小值	最大值
存储器建立时间	命令使能前地址建立的时钟周期 (HCLK) 数	读/写	AHB 时钟周期 (HCLK)	1	256
存储器等待	命令使能的最小持续时间（HCLK 时钟周期）	读/写	AHB 时钟周期 (HCLK)	2	256
存储器保持	命令禁止后保持地址（以及进行写访问时保持数据）的时钟周期 (HCLK) 数	读/写	AHB 时钟周期 (HCLK)	1	255
存储器数据总线高阻态	开始进行写访问后，数据总线保持高阻状态期间的时钟周期 (HCLK) 数	写	AHB 时钟周期 (HCLK)	0	255

32.6.1 外部存储器接口信号

下表列出了通常用于连接 NAND Flash 和 PC 卡的信号。

注意： *前缀“N”表示相关的信号为低电平有效。*

8 位 NAND Flash

表 216. 8 位 NAND Flash

FSMC 信号名称	I/O	功能
A[17]	O	NAND Flash 地址锁存使能 (ALE) 信号
A[16]	O	NAND Flash 命令锁存使能 (CLE) 信号
D[7:0]	I/O	8 位复用双向地址/数据总线
NCE[x]	O	片选，x = 2、3
NOE(= NRE)	O	输出使能（存储器信号名称：读取使能，NRE）
NWE	O	写入使能
NWAIT/INT[3:2]	I	指向 FSMC 的 NAND Flash 就绪/忙碌输入信号

由于 FSMC 能够管理足够多的地址周期，因此不存在理论容量限值。

16 位 NAND Flash

表 217. 16 位 NAND Flash

FSMC 信号名称	I/O	功能
A[17]	O	NAND Flash 地址锁存使能 (ALE) 信号
A[16]	O	NAND Flash 命令锁存使能 (CLE) 信号
D[15:0]	I/O	16 位复用双向地址/数据总线
NCE[x]	O	片选，x = 2、3
NOE(= NRE)	O	输出使能（存储器信号名称：读取使能，NRE）
NWE	O	写入使能
NWAIT/INT[3:2]	I	指向 FSMC 的 NAND Flash 就绪/忙碌输入信号

由于 FSMC 能够管理足够多的地址周期，因此不存在理论容量限值。

16 位 PC 卡

表 218. 16 位 PC 卡

FSMC 信号名称	I/O	功能
A[10:0]	O	地址总线
NIORD	O	I/O 空间的输出使能
NIOWR	O	I/O 空间的写入使能
NREG	O	指示在通用空间还是特性空间进行访问的寄存器信号
D[15:0]	I/O	双向数据总线
NCE4_1	O	片选 1
NCE4_2	O	片选 2（指示访问为 16 位还是 8 位）
NOE	O	通用空间和特性空间的输出使能
NWE	O	通用空间和特性空间的写入使能
NWAIT	I	指向 FSMC 的 PC 卡等待输入信号（存储器信号名称为 IORDY）
INTR	I	指向 FSMC 的 PC 卡中断（仅适用于能够生成中断的 PC 卡）
CD	I	PC 卡存在检测。高电平有效。如果 CD 处于低电平时对 PC 卡存储区域执行访问，会生成 AHB 错误。请参见*第 32.3 节：AHB 接口*

32.6.2 NAND Flash/PC 卡支持的存储器和事务

下面的*表 219* 介绍了所支持的设备、访问模式和事务。NAND Flash/PC 卡控制器不允许（或不支持）的事务以灰色显示。

表 219. 支持的存储器和事务

设备	模式	R/W	AHB 数据大小	存储器数据大小	允许/不允许	注释
8 位 NAND	异步	R	8	8	是	
	异步	W	8	8	是	
	异步	R	16	8	是	分为 2 次 FSMC 访问
	异步	W	16	8	是	分为 2 次 FSMC 访问
	异步	R	32	8	是	分为 4 次 FSMC 访问
	异步	W	32	8	是	分为 4 次 FSMC 访问
16 位 NAND	异步	R	8	16	是	
	异步	W	8	16	否	
	异步	R	16	16	是	
	异步	W	16	16	是	
	异步	R	32	16	是	分为 2 次 FSMC 访问
	异步	W	32	16	是	分为 2 次 FSMC 访问

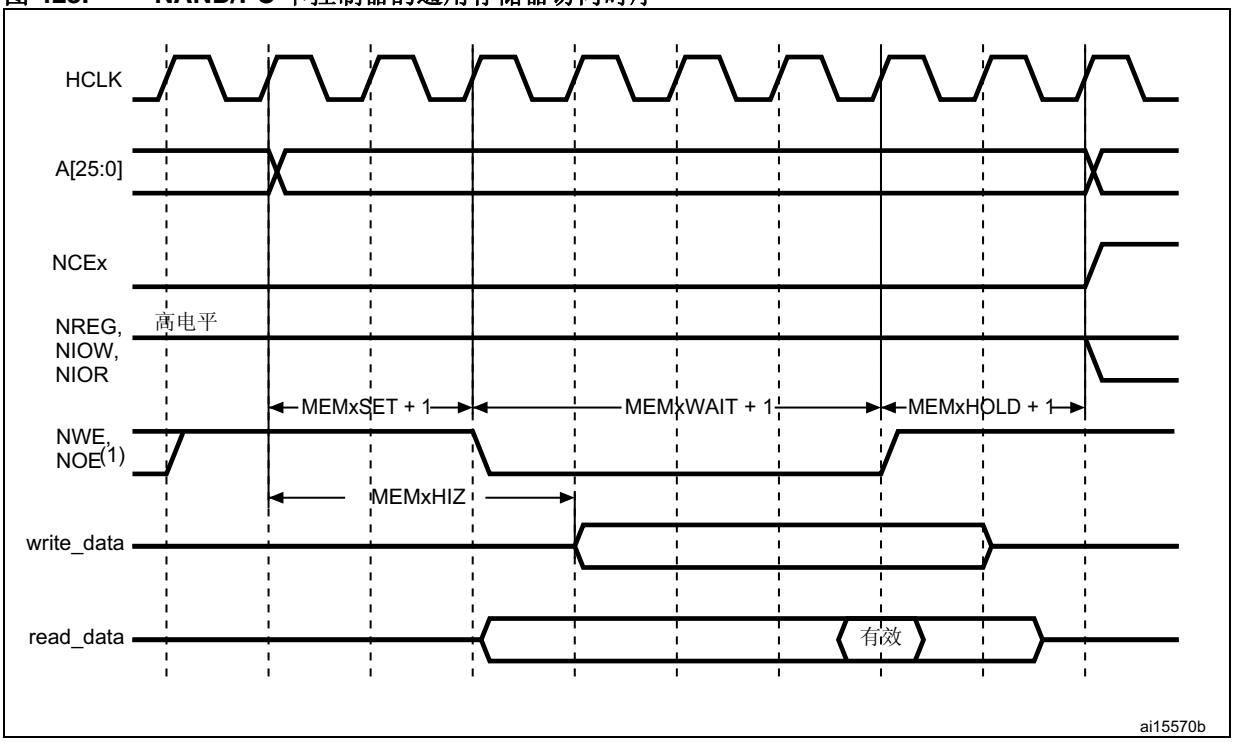
32.6.3 NAND 和 PC 卡的时序图

每个 PC 卡/CF 卡和 NAND Flash 存储区域均通过以下一组寄存器进行管理：

- 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
- 特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

每个时序配置寄存器包含四个参数，其中三个参数用于定义 PC 卡/CF 卡或 NAND Flash 的三个访问阶段的 HCLK 周期数，另一个参数用于定义进行写操作时开始驱动数据总线的时序。*图 423* 介绍了通用存储器访问的时序参数定义，特性存储器空间和 I/O（仅适用于 PC 卡）存储器空间的访问时序与此类似。

图 423. NAND/PC 卡控制器的通用存储器访问时序



1. 进行写访问期间 NOE 保持高电平（无效）。进行读访问期间 NWE 保持高电平（无效）。

32.6.4 NAND Flash 操作

NAND Flash 设备的命令锁存使能 (CLE) 信号和地址锁存使能 (ALE) 信号由 FSMC 控制器的某些地址信号驱动。这意味着要向 NAND Flash 发送命令或地址，CPU 必须对其存储器空间中的某个特定地址执行写操作。

从 NAND Flash 设备进行的典型页读取操作如下所示：

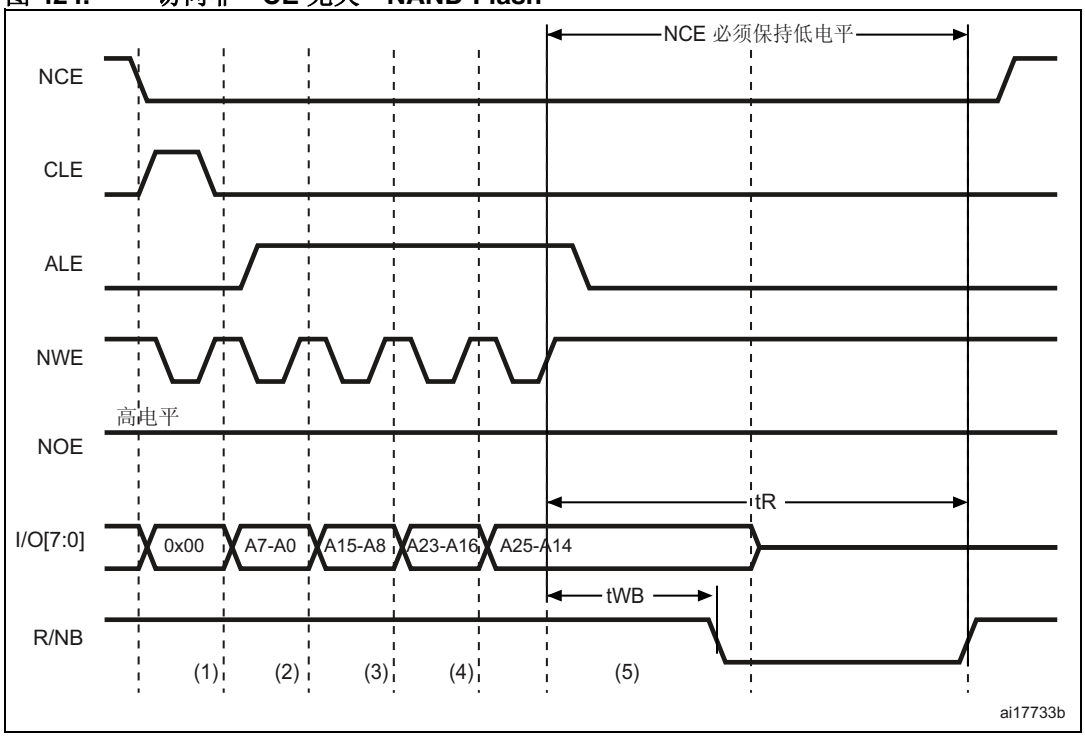
1. 根据 NAND Flash 的特性（PWID 位指示 NAND Flash 数据总线宽度，PTYP = 1，PWAITEN = 1，PBKEN = 1；有关时序配置，请参见*第 1237 页的通用存储器空间时序寄存器 2..4 (FSMC_PMEM2..4)*一节），配置 FSMC_PCRx 和 FSMC_PMEMx（对于某些设备，配置 FSMC_PATTx，请参见*第 1232 页的第 32.6.5 节：NAND Flash 预等待功能*）寄存器，进而配置和使能相应的存储区域。
2. CPU 在通用存储器空间执行字节写操作，此时数据字节等于一个 Flash 命令字节（例如 Samsung NAND Flash 设备的字节为 0x00）。在写入选通期间（NWE 上为低电平脉冲），NAND Flash 的 CLE 输入有效，因此可将已写入的字节视为 NAND Flash 的一个命令。该命令由 NAND Flash 设备锁存后，随后进行页读取操作时，不需要再对该命令执行写操作。
3. 通过在通用存储器空间或特性空间写入所需字节（例如 4 个字节，较小容量的设备写入 3 个字节，STARTAD[7:0]、STARTAD[15:8]、STARTAD[23:16] 以及最后为 64 Mb x 8 位的 NAND Flash 写入 STARTAD[25:24]），CPU 能够发送读操作的起始地址 (STARTAD)。在写入选通期间（NWE 上为低电平脉冲），NAND Flash 设备的 ALE 输入有效，因此可将已写入的字节视为读操作的起始地址。借助特性存储器空间，可使用 FSMC 的另一种不同时序配置，实现某些 NAND Flash 所需的预等待功能（有关详细信息，请参见*第 1232 页的第 32.6.5 节：NAND Flash 预等待功能*）。

4. 控制器在对同一个或另一个存储区域进行新访问之前，等待 NAND Flash 准备好（R/NB 信号处于高电平）进入有效状态。等待期间，控制器保持 NCE 信号有效（低电平）。
5. 然后 CPU 能够在通用存储器空间执行字节读操作，进而按字节读取 NAND Flash 页（数据字段 + 备用字段）。
6. 可按照以下三种不同的方式读取下一个 NAND Flash 页，而无需任何 CPU 命令或地址写操作：
   - 执行步骤 5 中介绍的操作
   - 通过重新开始步骤 3 中的操作随机访问一个新地址
   - 通过重新开始步骤 2 向 NAND Flash 设备发送新命令

32.6.5 NAND Flash 预等待功能

一些 NAND Flash 设备需要在写入地址的最后一部分后，控制器等待 R/NB 信号变为低电平，如*图 424* 所示。

图 424. 访问非“CE 无关”NAND-Flash



1. CPU 在地址 0x7001 0000 处写入字节 0x00。
2. CPU 在地址 0x7002 0000 处写入字节 A7-A0。
3. CPU 在地址 0x7002 0000 处写入字节 A15-A8。
4. CPU 在地址 0x7002 0000 处写入字节 A23-A16。
5. CPU 在地址 0x7802 0000 处写入字节 A25-A24：FSMC 通过 FSMC_PATT2 时序定义执行写访问，其中 ATTHOLD ≥ 7（假设 (7+1) × HCLK = 112 ns > t_{WB} 最大值）。这可确保 NCE 保持低电平，直到 R/NB 再次变为低电平后变为高电平（只有 NCE 信号对之有作用的 NAND Flash 才需要此功能）。

当需要此功能时，可通过编程 MEMHOLD 值确保满足 t_{WB} 时序，然而随后 CPU 对 NAND Flash 进行的所有读或写访问均会经过 (MEMHOLD + 1) 个 HCLK 周期的保持延迟（该延迟插入在 NWE 信号的上升沿与下一访问之间）。

要克服该时序限制，可使用特性存储器空间，将其时序寄存器编程为满足 t_{WB} 时序的 ATTHOLD 值并将 MEMHOLD 保持为其最小值。然后，CPU 必须使用通用存储器空间进行所有的 NAND Flash 读和写访问，向 NAND Flash 设备写入最后一个地址字节时除外，此时 CPU 必须对特性存储器空间执行写操作。

32.6.6 错误修正代码计算 ECC (NAND Flash)

FSMC PC 卡控制器包括两个错误修正代码计算硬件模块，每个存储区域各有一个。这些模块用于软件在系统中处理错误修正代码时，减少主机 CPU 工作负载。

这两个寄存器相同，分别与存储区域 2 和存储区域 3 相关联。因此，硬件 ECC 计算均不适用于与存储区域 4 相连的存储器。

对 NAND Flash 执行读或写操作时，相应每 256、512、1 024、2 048、4 096 或 8 192 个字节，FSMC 中使用的错误修正代码 (ECC) 算法可修正 1 位错误并且检测出 2 位错误。

每当 NAND Flash 存储区域处于激活状态时，ECC 模块均会监视 NAND Flash 数据总线和读/写信号（NCE 和 NWE）。

该功能的操作如下：

- 当在存储区域 2 或存储区域 3 中访问 NAND Flash 时，将锁存 D[15:0] 总线上出现的数据并将其用于 ECC 计算。
- 当在任何其它地址访问 NAND Flash 时，ECC 逻辑会进入空闲状态，不执行任何操作。因此，进行 ECC 计算时，用于定义 NAND Flash 命令或地址的写操作无效。

主机 CPU 从 NAND Flash 读取/向 NAND Flash 写入所需的字节数后，必须读取 FSMC_ECCR2/3 寄存器，才能检索计算出的值。读取后，应通过将 ECCEN 位复位为零来将这些寄存器清零。要计算新的数据块，必须将 FSMC_PCR2/3 寄存器中的 ECCEN 位置 1。

32.6.7 PC 卡/CF 卡操作

地址空间和存储器访问

FSMC 支持存储器模式和 I/O 模式下的 CF 卡存储或 PC 卡（不支持真正的 IDE 模式）。

CF 卡存储或 PC 卡由以下 3 个存储器空间组成：

- 通用存储器空间
- 特性空间
- I/O 存储器空间

nCE2 和 nCE1 引脚（分别位于 FSMC_NCE4_2 和 FSMC_NCE4_1 中）将选择卡，指示是否正在执行字节或字操作：nCE2 访问 D15-8 上的奇数字节；nCE1 访问 D7-0 上的偶数字节（A0=0 时）或 D7-0 上的奇数字节（A0=1 时）。如果 nCE2 和 nCE1 均处于低电平，则访问 D15-0 上的全字。

在读访问时将 nOE 置为低电平或写访问时将 nWE 置为低电平（同时将 nCE2/nCE1 和 nREG 置为低电平），可选择存储器空间。

- 如果存储器访问期间引脚 nREG=1，则选择通用存储器空间
- 如果存储器访问期间引脚 nREG=0，则选择特性存储器空间

在读访问时将 nIORD 置为低电平或写访问时将 nIOWR 置为低电平 [而非选择存储器空间时的 nOE/nWE]，同时将 nCE2/nCE1 置为低电平，可选择 I/O 空间。请注意，访问 I/O 空间期间，nREG 也必须置为低电平。

对于 16 位 PC 卡，允许三种类型的访问。

- 对通用存储器空间的访问（以进行数据存储）可以是偶地址上的 8 位访问，也可以是 AHB 地址上的 16 位访问。

  请注意，不支持在奇地址上进行 8 位访问，该访问也不会将 nCE2 置为低电平。32 位 AHB 请求会转换为 2 个 16 位存储器访问。
- 对特性存储器空间进行的访问（此时 PC 卡存储配置信息）必须是偶地址上的 8 位 AHB 访问。

  请注意，16 位 AHB 访问将转换为一个 8 位存储器传输：nCE1 将置为低电平，NCE2 将置为高电平，并且仅 D7-D0 上的偶数字节有效。32 位 AHB 访问将转换为偶地址上的两个 8 位存储器传输：nCE1 将置为低电平，NCE2 将置为高电平，并且仅偶数字节有效。
- 可通过 AHB 8 位或 16 位访问对 I/O 空间进行访问。

表 220. 16 位 PC 卡信号和访问类型

nCE2	nCE1	nREG	nOE/nWE	nIORD/nIOWR	A10	A9	A7-1	A0	空间	访问类型	是否允许
1	0	1	0	1	X	X	X-X	X	通用存储器空间	读/写 D7-D0 上的字节	是
0	1	1	0	1	X	X	X-X	X		读/写 D15-D8 上的字节	不支持
0	0	1	0	1	X	X	X-X	0		读/写 D15-D0 上的字节	是
X	0	0	0	1	0	1	X-X	0	特性空间	读取或写入配置寄存器	是
X	0	0	0	1	0	0	X-X	0		读取或写入 CIS（卡信息结构）	是
1	0	0	0	1	X	X	X-X	1	无效特性空间	读取或写入（奇地址）	是
0	1	0	0	1	X	X	X-X	x		读取或写入（奇地址）	是
1	0	0	1	0	X	X	X-X	0	I/O 空间	读取 D7-0 上的偶数字节	是
1	0	0	1	0	X	X	X-X	1		读取 D7-0 上的奇数字节	是
1	0	0	1	0	X	X	X-X	0		写入 D7-0 上的偶数字节	是
1	0	0	1	0	X	X	X-X	1		写入 D7-0 上的奇数字节	是
0	0	0	1	0	X	X	X-X	0		读取 D15-0 上的字	是
0	0	0	1	0	X	X	X-X	0		写入 D15-0 上的字	是
0	1	0	1	0	X	X	X-X	X		读取 D15-8 上的奇数字节	不支持
0	1	0	1	0	X	X	X-X	X		写入 D15-8 上的奇数字节	不支持

如*第 32.4.2 节：NAND/PC 卡地址映射*的*表 187：存储器映射和时序寄存器*所述，FSMC 存储区域 4 提供对相关 3 个存储空间的访问。

等待特性

如果通过 FSMC_PCRx 寄存器中的 PWAITEN 位使能了等待特性，则 CF 卡存储装置或 PC 卡可能会请求 FSMC 延长通过 MEMWAITx/ATTWAITx/IOWAITx 位配置的访问阶段的长度，并在激活 nOE/nWE 或 nIORD/nIOWR 后使能 nWAIT 信号。为了正确检测 nWAIT 使能情况，必须按如下方式编程 MEMWAITx/ATTWAITx/IOWAITx 位：

xxWAITx >= 4 + max_wait_assertion_time/HCLK

其中 max_wait_assertion_time 是在 nOE/nWE 或 nIORD/nIOWR 变为低电平后 NWAIT 变为低电平所花费的最长时间。

禁止 nWAIT 后，FSMC 会延长等待阶段 4 个 HCLK 时钟周期。

32.6.8 NAND Flash/PC 卡控制寄存器

NAND Flash/PC 卡控制寄存器必须通过字（32 位）访问。

PC 卡/NAND Flash 控制寄存器 2..4 (FSMC_PCR2..4)

PC Card/NAND Flash control registers 2..4

偏移地址：0xA0000000 + 0x40 + 0x20 * (x – 1)，x = 2..4

复位值：0x0000 0018

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												ECCPS			TAR				TCLR				Res.	ECCEN	PWID		PTYP	PBKEN	PWAITEN	Reserved
												rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw	

位 31:20 保留，必须保持复位值。

位 19:17 **ECCPS**：ECC 页大小 (ECC page size)。
定义扩展 ECC 的页大小：
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**：ALE 到 RE 的延迟 (ALE to RE delay)。
以 AHB 时钟周期数 (HCLK) 设置从 ALE 低电平到 RE 低电平的时间。
时间是：t_ar = (TAR + SET + 2) × THCLK，其中 THCLK 是 HCLK 时钟周期
0000：1 HCLK 周期（默认）
1111：16 HCLK 周期
注意：根据寻址空间，SET 为 MEMSET 或 ATTSET。

位 12:9 **TCLR**：CLE 到 RE 的延迟 (CLE to RE delay)。
以 AHB 时钟周期数 (HCLK) 设置从 CLE 低电平到 RE 低电平的时间。
时间是 t_clr = (TCLR + SET + 2) × THCLK，其中 THCLK 是 HCLK 时钟周期
0000：1 HCLK 周期（默认）
1111：16 HCLK 周期
注意：根据寻址空间，SET 为 MEMSET 或 ATTSET。

位 8:7 保留，必须保持复位值。

位 6 **ECCEN**：ECC 计算逻辑使能位 (ECC computation logic enable bit)
0：禁止和复位 ECC 逻辑（复位后为默认值），
1：使能 ECC 逻辑。

位 5:4 **PWID**：数据总线宽度 (Databus width)。
定义外部存储器设备宽度。
00：8 位（复位后为默认值）
01：16 位（PC 卡时必需如此）
10：保留，不使用
11：保留，不使用

位 3 **PTYP**：存储器类型 (Memory type)。
定义附加到相应存储区域的设备的类型：
0：PC 卡、CF 卡、CF+ 或 PCMCIA
1：NAND Flash（复位后为默认值）

位 2 **PBKEN**：PC 卡/NAND Flash 存储区域使能位 (PC Card/NAND Flash memory bank enable bit)。
使能存储区域。访问禁止的存储区域会引起 AHB 总线上的错误
0：禁止相应的存储区域（复位后为默认值）
1：使能相应的存储区域

位 1 **PWAITEN**：等待特性使能位 (Wait feature enable bit)。
使能 PC 卡/NAND Flash 存储区域的等待特性：
0：禁止
1：使能

注意：对于 PC 卡，使能等待特性时，必须按如下方式对 MEMWAITx/ATTWAITx/IOWAITx 位进行配置：
xxWAITx ≥ 4 + max_wait_assertion_time/HCLK
其中 max_wait_assertion_time 是在 nOE/nWE 或 nIORD/nIOWR 变为低电平后 NWAIT 变为低电平所花费的最长时间。

位 0 保留，必须保持复位值。

FIFO 状态和中断寄存器 2..4 (FSMC_SR2..4)

FIFO status and interrupt register 2..4

偏移地址：0xA000 0000 + 0x44 + 0x20 * (x-1)，x = 2..4

复位值：0x0000 0040

该寄存器包含有关 FIFO 状态和中断的信息。FSMC 有一个 FIFO，当向存储器执行写入操作以存储来自 AHB 的多达 16 字的数据时，使用该 FIFO。
当 FSMC 将其 FIFO 的内容移入存储器时，此寄存器用于快速写入 AHB，然后释放 AHB 供 FSMC 以外的外设的事务使用。该寄存器有一位用来指示 FIFO 的状态，供 ECC 使用。
当将数据写入存储器时计算 ECC，因此，为了读取正确的 ECC，软件必须等待到 FIFO 为空。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																									FEMPT	IFEN	ILEN	IREN	IFS	ILS	IRS
																									r	rw	rw	rw	rw	rw	rw

位 31:7 保留，必须保持复位值。

位 6 **FEMPT**：FIFO 为空 (FIFO empty)。
该位是提供 FIFO 状态的只读位
0：FIFO 非空
1：FIFO 为空

位 5 **IFEN**：中断下降沿检测使能位 (Interrupt falling edge detection enable bit)
0：中断下降沿检测请求禁止
1：中断下降沿检测请求使能

位 4 **ILEN**：中断高电平检测使能位 (Interrupt high-level detection enable bit)
0：中断高电平检测请求禁止
1：中断高电平检测请求使能

位 3 **IREN**：中断上升沿检测使能位 (Interrupt rising edge detection enable bit)
0：中断上升沿检测请求禁止
1：中断上升沿检测请求使能

位 2 **IFS**：中断下降沿状态 (Interrupt falling edge status)
此标志由硬件置 1，由软件复位。
0：未出现中断下降沿
1：出现中断下降沿

位 1 **ILS**：中断高电平状态 (Interrupt high-level status)
此标志由硬件置 1，由软件复位。
0：未出现中断高电平
1：出现中断高电平

位 0 **IRS**：中断上升沿状态 (Interrupt rising edge status)
此标志由硬件置 1，由软件复位。
0：未出现中断上升沿
1：出现中断上升沿

通用存储器空间时序寄存器 2..4 (FSMC_PMEM2..4)

Common memory space timing register 2..4

偏移地址：地址：0xA000 0000 + 0x48 + 0x20 * (x – 1)，x = 2..4

复位值：0xFCFC FCFC

每个 FSMC_PMEMx (x = 2..4) 读/写寄存器都包含 PC 卡或 NAND Flash 存储区域 x 的时序信息，用于访问 16 位 PC 卡/CF 卡的通用存储空间，或者访问 NAND Flash 来实现命令、地址写和数据读/写访问。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MEMHIZx								MEMHOLDx								MEMWAITx								MEMSETx							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:24 **MEMHIZx**：通用存储器 x 数据总线高阻态时间 (Common memory x databus HiZ time)
定义在存储区 x 上对 PC 卡/NAND Flash 的通用存储空间开始执行写访问之后，数据总线保持高阻态所持续的 HCLK 时钟周期数。仅对写入事务有效：
0000 0000：(0x00) 0 个 HCLK 周期（对于 PC 卡）
1111 1111：(0xFF) 255 个 HCLK 周期（对于 PC 卡）-（复位后的默认值）

位 23:16 **MEMHOLDx**：通用存储器 x 保持时间 (Common memory x hold time)

针对在存储区 x 上对 PC 卡/NAND Flash 的通用存储空间执行的读或写访问，定义禁止命令（NWE、NOE）之后保持地址（和写访问数据）的 HCLK 时钟周期数。

0000 0000：保留

0000 0001：1 个 HCLK 周期

1111 1111：255 个 HCLK 周期（复位后的默认值）

位 15:8 **MEMWAITx**：通用存储器 x 等待时间 (Common memory x wait time)

针对在存储区 x 上对 PC 卡/NAND Flash 的通用存储空间执行的读或写访问，定义使能命令（NWE、NOE）所需的 HCLK (+1) 时钟周期数最小值。如果等待信号 (NWAIT) 在编程的 HCLK 值末尾处有效（低电平），则命令使能的持续时间将延长。

0000 0000：保留

0000 0001：2 个 HCLK 周期（+ 禁止 NWAIT 时引入的等待周期）

1111 1111：256 个 HCLK 周期（+ 卡禁止 NWAIT 时引入的等待周期）（复位后的默认值）

位 7:0 **MEMSETx**：通用存储器 x 建立时间 (Common memory x setup time)

针对在存储区 x 上对 PC 卡/NAND Flash 的通用存储空间执行的读或写访问，定义使能命令（NWE，NOE）前建立地址所需的 HCLK () 时钟周期数：

0000 0000：1 个 HCLK 周期（对于 PC 卡）/HCLK 周期（对于 NAND Flash）

1111 1111：256 个 HCLK 周期（对于 PC 卡）/257 HCLK 周期（对于 NAND Flash） - （复位后的默认值）

属性存储器空间时序寄存器 2..4 (FSMC_PATT2..4)

Attribute memory space timing registers 2..4

偏移地址：0xA000 0000 + 0x4C + 0x20 * (x – 1)，x = 2..4

复位值：0xFCFC FCFC

每个 FSMC_PATTx (x = 2..4) 读/写寄存器都包含 PC 卡/CF 卡或 NAND Flash 存储区域 x 的时序信息。该寄存器用于对 PC 卡/CF 卡的特性存储空间的 8 位访问，或在最后一次地址写访问必须与先前访问的时序不同的情况下，访问 NAND Flash 来实现最后一次地址写访问（有关就绪/繁忙管理的信息，请参见*第 32.6.5 节：NAND Flash 预等待功能*）。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ATTHIZx								ATTHOLDx								ATTWAITx								ATTSETx							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:24 **ATTHIZx**：特性存储器 x 数据总线高阻态时间 (Attribute memory x databus HiZ time)

定义在存储区 x 上对 PC 卡/NAND Flash 的特性存储空间开始执行写访问之后，数据总线保持高阻态所持续的 HCLK 时钟周期数。仅对写入事务有效：

0000 0000：0 个 HCLK 周期

1111 1111：255 个 HCLK 周期（复位后的默认值）

位 23:16 **ATTHOLDx**：特性存储器 x 保持时间 (Attribute memory x hold time)

针对在存储区 x 上对 PC 卡/NAND Flash 的特性存储空间执行的读或写访问，定义禁止命令（NWE、NOE）之后保持地址（和写访问数据）的 HCLK 时钟周期数。

0000 0000：保留

0000 0001：1 个 HCLK 周期

1111 1111：255 个 HCLK 周期（复位后的默认值）

位 15:8 **ATTWAITx**：特性存储器 x 等待时间 (Attribute memory x wait time)

针对在存储区 x 上对 PC 卡/NAND Flash 的特性存储空间执行的读或写访问，定义使能命令（NWE、NOE）所需的 HCLK (+1) 时钟周期数最小值。如果等待信号 (NWAIT) 在编程的 HCLK 值末尾处有效（低电平），则命令使能的持续时间将延长。

0000 0000：保留

0000 0001：2 个 HCLK 周期（+ 禁止 NWAIT 时引入的等待周期）

1111 1111：256 个 HCLK 周期（+ 卡禁止 NWAIT 时引入的等待周期）（复位后的默认值）

位 7:0 **ATTSETx**：特性存储器 x 建立时间 (Attribute memory x setup time)

针对在存储区 x 上对 PC 卡/NAND Flash 的特性存储空间执行的读或写访问，定义使能命令（NWE，NOE）前建立地址所需的 HCLK (+1) 时钟周期数：

0000 0000：1 个 HCLK 周期

1111 1111：256 个 HCLK 周期（复位后的默认值）

I/O 空间时序寄存器 4 (FSMC_PIO4)

I/O space timing register 4

偏移地址：0xA000 0000 + 0xB0
复位值：0xFCFCFCFC

FSMC_PIO4 读/写寄存器中包含访问 16 位 PC 卡/CF 卡所需的时序信息。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IOHIZx								IOHOLDx								IOWAITx								IOSETx							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:24 **IOHIZx**：I/O x 数据总线高阻态时间 (I/O x databus HiZ time)

定义在存储区 x 上对 PC 卡的 I/O 空间开始执行写访问之后，数据总线保持高阻态所持续的 HCLK 时钟周期数。仅对写入事务有效：

0000 0000：0 个 HCLK 周期

1111 1111：255 个 HCLK 周期（复位后的默认值）

位 23:16 **IOHOLDx**：I/O x 保持时间 (I/O x hold time)

针对在存储区 x 上对 PC 卡的 I/O 空间执行的读或写访问，定义禁止命令（NWE，NOE）之后保持地址（和写访问数据）的 HCLK 时钟周期数。

0000 0000：保留

0000 0001：1 个 HCLK 周期

1111 1111：255 个 HCLK 周期（复位后的默认值）

位 15:8 **IOWAITx**：I/O x 等待时间 (I/O x wait time)

针对在存储区 x 上对 PC 卡的 I/O 空间执行的读或写访问，定义使能命令（SMNWE、SMNOE）所需的 HCLK (+1) 时钟周期数最小值。如果等待信号 (NWAIT) 在编程的 HCLK 值末尾处有效（低电平），则命令使能的持续时间将延长。

0000 0000：保留，不使用该值

0000 0001：2 个 HCLK 周期（+ 禁止 NWAIT 时引入的等待周期）

1111 1111：256 个 HCLK 周期（+ 卡禁止 NWAIT 时引入的等待周期）（复位后的默认值）

位 7:0 **IOSETx**：I/O x 建立时间 (I/O x setup time)

针对在存储区 x 上对 PC 卡的 I/O 空间执行的读或写访问，定义使能命令（NWE，NOE）前建立地址所需的 HCLK (+1) 时钟周期数：

0000 0000：1 个 HCLK 周期

1111 1111：256 个 HCLK 周期（复位后的默认值）

ECC 结果寄存器 2/3 (FSMC_ECCR2/3)

ECC result registers 2/3

偏移地址：0xA000 0000 + 0x54 + 0x20 * (x – 1)，x = 2 或 3

复位值：0x0000 0000

这些寄存器中包含由 FSMC 控制器的 ECC 计算模块（每个 NAND Flash 存储区域一个模块）计算所得的当前错误校正代码值。当 CPU 从 NAND Flash 页的正确地址读取数据时（请参见 *第 32.6.6 节：错误修正代码计算 ECC (NAND Flash)*），ECC 计算模块会自动处理读取自/写入 NAND Flash 的数据。当 X 字节读取操作结束时（根据 FSMC_PCRx 寄存器中的 ECCPS 字段判断），CPU 必须从 FSMC_ECCx 寄存器读取计算所得的 ECC 值，然后验证这些计算的奇偶校验数据是否与备用区记录的奇偶校验数据相同，从而确定该页是否有效，并根据需要进行校正。读取 FSMC_ECCRx 寄存器后应通过将 ECCEN 位置零将其清零。如要计算新的数据块，则必须将 ECCEN 位置 1。

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

位 31:0 **ECCx**：ECC 结果 (ECC result)

该字段提供由 ECC 计算逻辑计算所得的值。*表 221* 介绍了这些位字段的内容。

表 221. ECC 结果相关位

ECCPS[2:0]	以字节为单位的页大小	ECC 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]

32.6.9 FSMC 寄存器映射

下表对 FSMC 寄存器进行了汇总。

表 222. FSMC 寄存器映射

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xA000 0000	FSMC_BCR1	Reserved												CBURSTRW	Reserved			ASYNCWAIT	EXTMOD	WAITEN	WREN	WAITCFG	WRAPMOD	WAITPOL	BURSTEN	Reserved	FACCEN	MWID		MTYP		MUXEN	MBKEN
0xA000 0008	FSMC_BCR2	Reserved												CBURSTRW	Reserved			ASYNCWAIT	EXTMOD	WAITEN	WREN	WAITCFG	WRAPMOD	WAITPOL	BURSTEN	Reserved	FACCEN	MWID		MTYP		MUXEN	MBKEN
0xA000 0010	FSMC_BCR3	Reserved												CBURSTRW	Reserved			ASYNCWAIT	EXTMOD	WAITEN	WREN	WAITCFG	WRAPMOD	WAITPOL	BURSTEN	Reserved	FACCEN	MWID		MTYP		MUXEN	MBKEN
0xA000 0018	FSMC_BCR4	Reserved												CBURSTRW	Reserved			ASYNCWAIT	EXTMOD	WAITEN	WREN	WAITCFG	WRAPMOD	WAITPOL	BURSTEN	Reserved	FACCEN	MWID		MTYP		MUXEN	MBKEN
0xA000 0004	FSMC_BTR1	Res.		ACCMOD		DATLAT				CLKDIV				BUSTURN				DATAST								ADDHLD				ADDSET			
0xA000 000C	FSMC_BTR2	Res.		ACCMOD		DATLAT				CLKDIV				BUSTURN				DATAST								ADDHLD				ADDSET			
0xA000 0014	FSMC_BTR3	Res.		ACCMOD		DATLAT				CLKDIV				BUSTURN				DATAST								ADDHLD				ADDSET			
0xA000 001C	FSMC_BTR4	Res.		ACCMOD		DATLAT				CLKDIV				BUSTURN				DATAST								ADDHLD				ADDSET			
0xA000 0104	FSMC_BWTR1	Res.		ACCMOD		DATLAT				CLKDIV				BUSTURN				DATAST								ADDHLD				ADDSET			
0xA000 010C	FSMC_BWTR2	Res.		ACCMOD		DATLAT				CLKDIV				BUSTURN				DATAST								ADDHLD				ADDSET			
0xA000 0114	FSMC_BWTR3	Res.		ACCMOD		DATLAT				CLKDIV				BUSTURN				DATAST								ADDHLD				ADDSET			
0xA000 011C	FSMC_BWTR4	Res.		ACCMOD		DATLAT				CLKDIV				BUSTURN				DATAST								ADDHLD				ADDSET			
0xA000 0060	FSMC_PCR2	Reserved												ECCPS				TAR				TCLR			Res.	ECCEN		PWID		PTYP		PBKEN	PWAITEN	Reserved
0xA000 0080	FSMC_PCR3	Reserved												ECCPS				TAR				TCLR			Res.	ECCEN		PWID		PTYP		PBKEN	PWAITEN	Reserved
0xA000 00A0	FSMC_PCR4	Reserved												ECCPS				TAR				TCLR			Res.	ECCEN		PWID		PTYP		PBKEN	PWAITEN	Reserved
0xA000 0064	FSMC_SR2	Reserved																								FEMPT	IFEN	ILEN		IREN	IFS	ILS	IRS
0xA000 0084	FSMC_SR3	Reserved																								FEMPT	IFEN	ILEN		IREN	IFS	ILS	IRS
0xA000 00A4	FSMC_SR4	Reserved																								FEMPT	IFEN	ILEN		IREN	IFS	ILS	IRS
0xA000 0068	FSMC_PMEM2	MEMHIZx								MEMHOLDx								MEMWAITx								MEMSETx							
0xA000 0088	FSMC_PMEM3	MEMHIZx								MEMHOLDx								MEMWAITx								MEMSETx							

表 222. FSMC 寄存器映射 （续）

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xA000 00A8	FSMC_PMEM4	MEMHIZx								MEMHOLDx								MEMWAITx								MEMSETx							
0xA000 006C	FSMC_PATT2	ATTHIZx								ATTHOLDx								ATTWAITx								ATTSETx							
0xA000 008C	FSMC_PATT3	ATTHIZx								ATTHOLDx								ATTWAITx								ATTSETx							
0xA000 00AC	FSMC_PATT4	ATTHIZx								ATTHOLDx								ATTWAITx								ATTSETx							
0xA000 00B0	FSMC_PIO4	IOHIZx								IOHOLDx								IOWAITx								IOSETx							
0xA000 0074	FSMC_ECCR2	ECCx																															
0xA000 0094	FSMC_ECCR3	ECCx																															

有关寄存器边界地址的信息，请参见*第 52 页的表 2*。

33 调试支持 (DBG)

除非特别说明，否则本部分适用于整个 STM32F4xx 系列。

33.1 概述

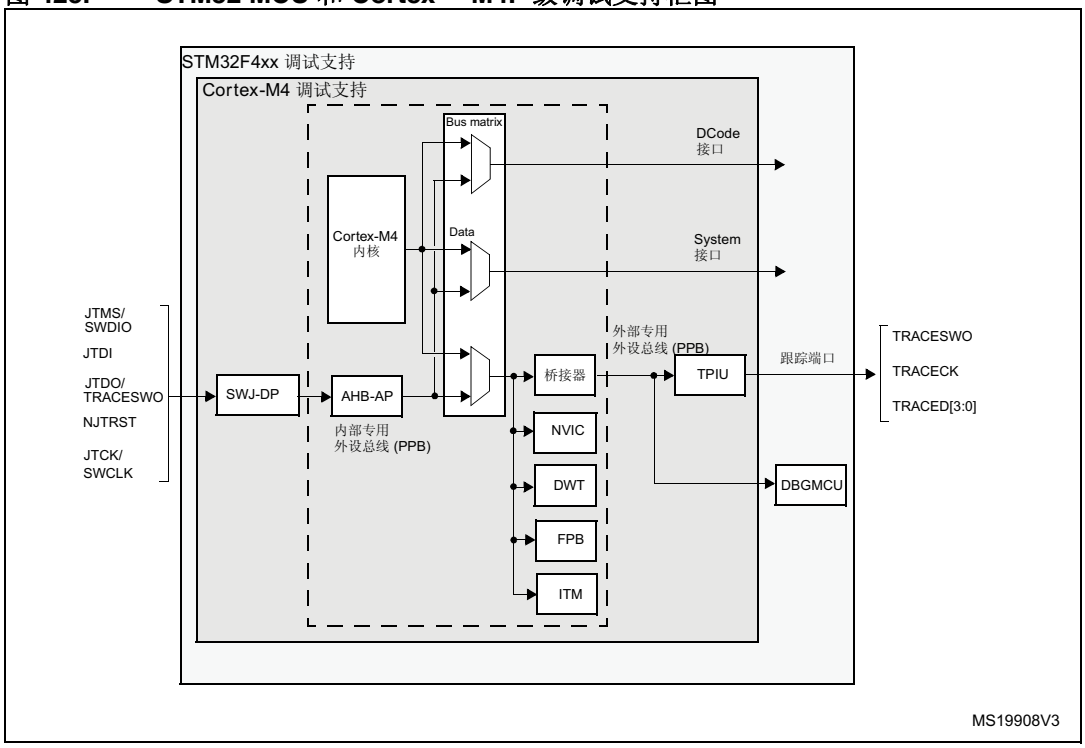
STM32F4xx 的内核是 Cortex™-M4F，该内核包含用于高级调试功能的硬件。利用这些调试功能，可以在取指（指令断点）或取访问数据（数据断点）时停止内核。内核停止时，可以查询内核的内部状态和系统的外部状态。查询完成后，将恢复内核和系统并恢复程序执行。

当调试器与 STM32F4xx MCU 相连并进行调试时，将使用内核的硬件调试模块。

提供两个调试接口：

- 串行接口
- JTAG 调试接口

图 425. STM32 MCU 和 Cortex™-M4F 级调试支持框图



注意： Cortex™-M4F *内核中内置的调试功能是 ARM CoreSight 设计套件的一部分。*

ARM Cortex™-M4F 内核提供集成片上调试支持。它包括：

- SWJ-DP：串行/JTAG 调试端口
- AHP-AP：AHB 访问端口
- ITM：指令跟踪单元
- FPB：Flash 指令断点
- DWT：数据断点触发
- TPUI：跟踪端口单元接口（大封装上提供，其中会映射相应引脚）
- ETM：嵌入式跟踪宏单元（大封装上提供，其中会映射相应引脚）

它还包括专用于 STM32F4xx 的调试功能：

- 灵活调试引脚分配
- MCU 调试盒（支持低功耗模式和对外设时钟的控制等）

注意： *有关 ARM Cortex™-M4F 内核支持的调试功能的更多信息，请参见《Cortex™-M4F-r0p1 技术参考手册》和《CoreSight 设计套件 r0p1 技术参考手册》（请参见第 33.2 节：ARM 参考文档）。*

33.2 ARM 参考文档

- Cortex™-M4F r0p1 技术参考手册 (TRM)
  （请参见第 1 页上的相关文档）
- ARM 调试接口 V5
- ARM CoreSight 设计套件版本 r0p1 技术参考手册

33.3 SWJ 调试端口（串行接口和 JTAG）

STM32F4xx 内核集成了串行/JTAG 调试端口 (SWJ-DP)。该端口是 ARM 标准 CoreSight 调试端口，具有 JTAG-DP（5 引脚）接口和 SW-DP（2 引脚）接口。

- JTAG 调试端口 (JTAG-DP) 提供用于连接到 AHP-AP 端口的 5 引脚标准 JTAG 接口。
- 串行线调试端口 (SW-DP) 提供用于连接到 AHP-AP 端口的 2 引脚（时钟 + 数据）接口。

在 SWJ-DP 中，SW-DP 的 2 个 JTAG 引脚与 JTAG-DP 的 5 个 JTAG 引脚中的部分引脚复用。

图 426. SWJ 调试端口

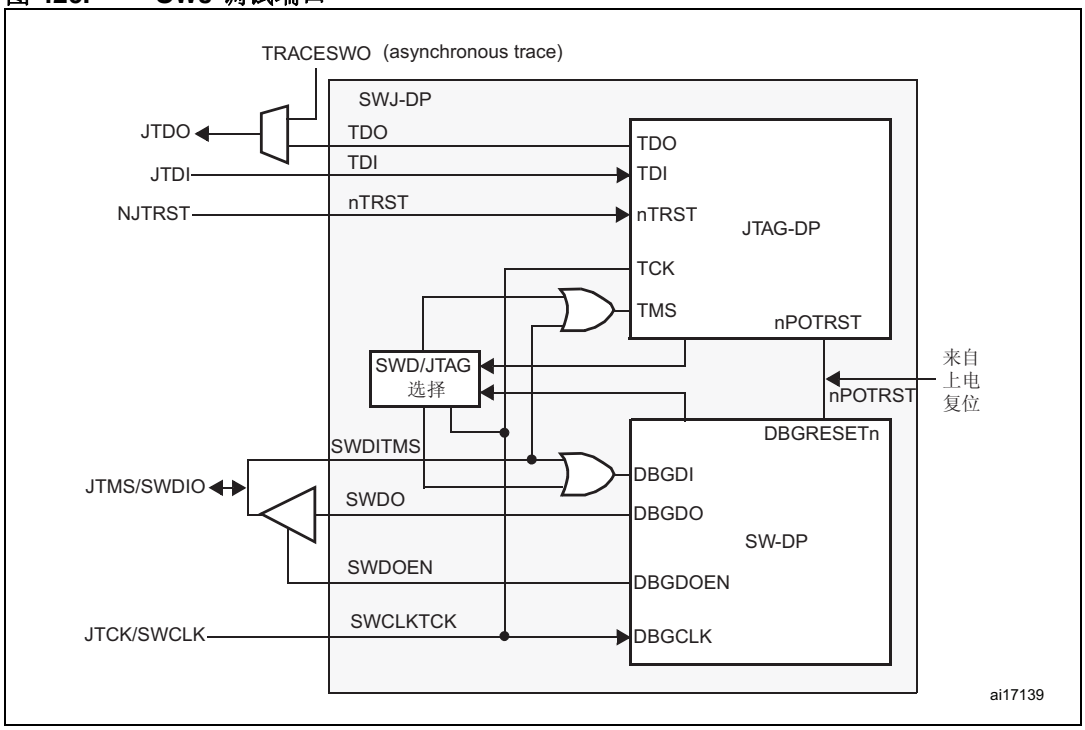


图 426 显示了异步 TRACE 输出 (TRACESWO) 与 TDO 复用。这意味着异步跟踪只能在 SW-DP 上实现，不能在 JTAG-DP 上实现。

33.3.1 JTAG-DP 或 SW-DP 的切换机制

默认调试接口是 JTAG 接口。

如果调试工具想要切换到 SW-DP，它必须在 TMS/TCK（分别映射到 SWDIO 和 SWCLK）上提供专用的 JTAG 序列，用于禁止 JTAG-DP 并使能 SW-DP。这样便可仅使用 SWCLK 和 SWDIO 引脚来激活 SWDP。

该序列为：

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

33.4 引脚排列和调试端口引脚

STM32F4xx MCU 的不同封装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。

33.4.1 SWJ 调试端口引脚

STM32F4xx 的 5 个普通 I/O 口可用作 SWJ-DP 接口引脚。所有封装都提供这些引脚。

表 223. SWJ 调试端口引脚

SWJ-DP 引脚名称	JTAG 调试端口		SW 调试端口		引脚分配
	类型	说明	类型	调试分配	
JTMS/SWDIO	I	JTAG 测试模式选择	IO	串行线数据输入/输出	PA13
JTCK/SWCLK	I	JTAG 测试时钟	I	串行线时钟	PA14
JTDI	I	JTAG 测试数据输入	-	-	PA15
JTDO/TRACESWO	O	JTAG 测试数据输出	-	TRACESWO（如果使能异步跟踪）	PB3
NJTRST	I	JTAG 测试 nReset	-	-	PB4

33.4.2 灵活的 SWJ-DP 引脚分配

复位（SYSRESETn 或 PORESETn）后，会将用于 SWJ-DP 的全部 5 个引脚指定为专用引脚，可供调试工具立即使用（请注意，除非由调试工具明确编程，否则不分配跟踪输出）。

但是，STM32F4xx MCU 可以禁止部分或全部 SWJ-DP 端口，进而释放相关引脚以用作通用 IO (GPIO)。有关如何禁止 SWJ-DP 端口引脚的更多详细信息，请参见*第 7.3.2 节：I/O 引脚复用器和映射*。

表 224. 灵活的 SWJ-DP 引脚分配

可用的调试端口	分配的 SWJ IO 引脚				
	PA13/ JTMS/ SWDIO	PA14 / JTCK/ SWCLK	PA15 / JTDI	PB3 / JTDO	PB4/ NJTRST
全部 SWJ (JTAG-DP + SW-DP) - 复位状态	X	X	X	X	X
全部 SWJ (JTAG-DP + SW-DP)，但不包括 NJTRST	X	X	X	X	
禁止 JTAG-DP 和使能 SW-DP	X	X			
禁止 JTAG-DP 和禁止 SW-DP	已释放				

注意： *当 APB 桥的写缓冲区已满后，还需要一个额外的 APB 周期来写入 GPIO_AFR 寄存器。这是因为释放 JTAGSW 引脚需要两个周期，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）。

33.4.3 JTAG 引脚上的内部上拉和下拉

必须确保 JTAG 输入引脚不悬空，因为这些引脚直接连接到用于控制调试模式功能的触发器。还必须特别注意 SWCLK/TCK 引脚，该引脚直接连接到一些触发器的时钟。

为避免 IO 电平浮空，器件在 JTAG 引脚上内置有内部上拉和下拉：

- NJTRST：内部上拉
- JTDI：内部上拉
- JTMS/SWDIO：内部上拉
- TCK/SWCLK：内部下拉

用户软件释放 JTAG IO 后，GPIO 控制器便会再次取得控制权。GPIO 控制寄存器的复位状态会将 I/O 置于：

- NJTRST：AF 输入上拉
- JTDI：AF 输入上拉
- JTMS/SWDIO：AF 输入上拉
- JTCK/SWCLK：AF 输入下拉
- JTDO：AF 输出悬空

软件可以把这些 I/O 口作为普通的 I/O 口使用。

注意： *JTAG IEEE 标准建议在 TDI、TMS 和 nTRST 上添加上拉，但对 TCK 没有特殊建议。但是，对于 JTCK，器件集成下拉。*

由于带有上拉和下拉电阻，因此无需添加外部电阻。

33.4.4 使用串行接口以及释放未使用的调试引脚以用作 GPIO

为了利用串行调试接口以便释放一些 GPIO，用户软件必须在 GPIO_MODER 寄存器中更改 GPIO（PA15、PB3 和 PB4）配置模式。这样便可释放 PA15、PB3 和 PB4，以将这些引脚用作 GPIO。

调试时，主机执行以下操作：

- 在系统复位状态下，分配所有 SWJ 引脚 (JTAG-DP + SW-DP)。
- 在系统复位状态下，调试主机发送 JTAG 序列，以从 JTAG-DP 切换到 SW-DP。
- 仍然在系统复位状态下，调试主机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。
- 从此所有调试通信均使用 SW-DP 完成。然后通过用户软件将其它 JTAG 引脚重新分配为 GPIO。

注意： *对于用户软件设计，请注意：*

请记住，要释放调试引脚，在复位后一直到用户软件释放引脚这段期间，这些引脚仍然处于输入上拉（nTRST、TMS 和 TDI）、下拉 (TCK) 或输出三态 (TDO)。

当这些引脚被配置为专用引脚时（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。

33.5 STM32F4xx JTAG TAP 连接

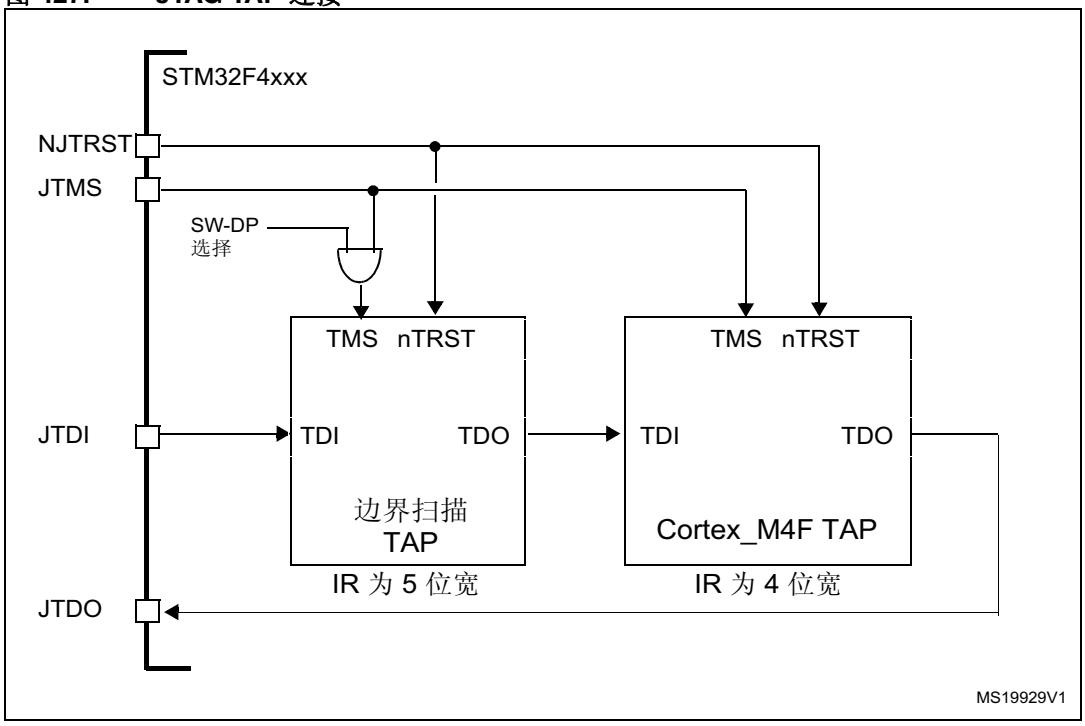
STM32F4xx MCU 集成了两个串连的 JTAG TAP、边界扫描 TAP（IR 宽度为 5 位）和 Cortex™-M4F TAP（IR 宽度为 4 位）。

要访问 Cortex™-M4F 的 TAP 以进行调试：

1. 首先，必须将 BYPASS 指令移位输入 TMC TAP。
2. 其次，在移位输入 IR 时，每个扫描链包含 9 个比特位 (=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比特位。

注意： ***重要提示：****使用专用 ARM JTAG 序列选择串行接口后，将自动禁止边界扫描 TAP（JTMS 强制为高电平）。*

图 427. JTAG TAP 连接



33.6 ID 代码和锁定机制

STM32F4xx MCU 中提供了一些 ID 代码。ST 强烈建议工具设计人员使用位于外部 PPB 存储器映射的地址 0xE0042000 中的 MCU DEVICE ID 代码锁定调试工具。

33.6.1 MCU 器件 ID 代码

STM32F4xx MCU 集成了 MCU ID 代码。此 ID 标识 ST MCU 的料号和晶片版本。它是 DBG_MCU 组件的一部分并映射到外部 PPB 总线上（请参见*第 1259 页的第 33.16 节*）。可通过 JTAG 调试端口（4 到 5 个引脚）或 SW 调试端口（2 个引脚）或者用户软件访问此代码。即使在 MCU 处于系统复位状态下也可以访问。

DBGMCU_IDCODE

地址：0xE004 2000

仅支持 32 位访问。只读。

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

位 31:16 **REV_ID(15:0)** 版本标识符 (Revision identifier)
此字段指示器件的版本：
0x1000 = 版本 A
0x1001 = 版本 Z

位 15:12 保留，必须保持复位值。

位 11:0 **DEV_ID(11:0)**：器件标识符 (Device identifier)（STM32F405xx/07xx 和 STM32F415xx/17xx）
器件 ID 为 0x413。

位 11:0 **DEV_ID(11:0)**：器件标识符（STM32F42xxx 和 STM32F43xxx）
器件 ID 为 0x419。

33.6.2 边界扫描 TAP

JTAG ID 代码

STM32F4xx BSC（边界扫描）的 TAP 集成了 JTAG ID 代码（等于 0x06413041）。

33.6.3 Cortex™-M4F TAP

ARM Cortex™-M4F 的 TAP 集成了 JTAG ID 代码。此 ID 代码是 ARM 的默认代码，未经修改。此代码只能通过 JTAG 调试端口进行访问。
此代码为 **0x4BA00477**（对应于 Cortex™-M4F r0p1，请参见*第 33.2 节：ARM 参考文档*）。

调试软件/编程工具只能将 DEV_ID(11:0) 用于识别芯片。

33.6.4 Cortex™-M4F JEDEC-106 ID 代码

ARM Cortex™-M4F 集成了 JEDEC-106 ID 代码。它位于映射到内部 PPB 总线地址为 0xE00FF000_0xE00FFFFF 的 4KB ROM 表中。

可通过 JTAG 调试端口（4 到 5 个引脚）或 SW 调试端口（2 个引脚）或者用户软件访问此代码。

33.7 JTAG 调试端口

标准 JTAG 状态机使用一个 4 位指令寄存器 (IR) 和五个数据寄存器实现（有关完整详细信息，请参见《Cortex™-M4Fr0p1 技术参考手册 (TRM)》。*有关参考信息，请参见第 33.2 节：ARM 参考文档*）。

表 225. JTAG 调试端口数据寄存器

IR(3:0)	数据寄存器	详细信息
1111	BYPASS [1 位]	
1110	IDCODE [32 位]	*ID CODE* 0x4BA00477（ARM Cortex™-M4F r0p1 ID 代码）
1010	DPACC [35 位]	*调试接口寄存器* 初始化调试接口，并允许访问调试接口寄存器。 – 传输 IN 数据时： 位 34:3 = DATA[31:0] = 为写请求传输的 32 位数据 位 2:1 = A[3:2] = 调试端口寄存器的 2 位地址。 位 0 = RnW = 读请求 (1) 或写请求 (0)。 – 传输 OUT 数据时： 位 34:3 = DATA[31:0] = 读请求后读取的 32 位数据 位 2:0 = ACK[2:0] = 3 位确认： 010 = OK/FAULT 001 = WAIT 其它 = 保留 有关 A(3:2) 位的说明，请参见*表 226*
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。 – 传输 IN 数据时： 位 34:3 = DATA[31:0] = 为写请求移入的 32 位数据 位 2:1 = A[3:2] = 2 位地址（子地址 AP 寄存器）。 位 0 = RnW = 读请求 (1) 或写请求 (0)。 – 传输 OUT 数据时： 位 34:3 = DATA[31:0] = 读请求后读取的 32 位数据 位 2:0 = ACK[2:0] = 3 位确认： 010 = OK/FAULT 001 = WAIT 其它 = 保留 有多个 AP 寄存器（请参见 AHB-AP），这些寄存器按以下组合进行寻址： – 移位值 A[3:2] – DP SELECT 寄存器的当前值
1000	ABORT [35 位]	*中止寄存器* – 位 31:1 = 保留 – 位 0 = DAPABORT：写入 1 以生成 DAP 中止。

表 226. 32 位调试端口寄存器，通过移位值 A[3:2] 进行寻址

地址	A(3:2) 值	说明
0x0	00	保留，必须保持复位值。
0x4	01	DP CTRL/STAT 寄存器。用于： – 请求系统或调试上电 – 配置 AP 访问的传输操作 – 控制比较和验证操作 – 读取一些状态标志（上溢和上电确认）
0x8	10	DP SELECT 寄存器：用于选择当前访问端口和活动的 4 字寄存器窗口。 – 位 31:24：APSEL：选择当前 AP – 位 23:8：保留 – 位 7:4：APBANKSEL：在当前 AP 上选择活动的 4 字寄存器窗口 – 位 3:0：保留
0xC	11	DP RDBUFF 寄存器：用于通过调试软件在执行一系列操作后获取最后结果（无需请求新的 JTAG-DP 操作）

33.8 SW 调试端口

33.8.1 SW 协议简介

此同步串行协议使用两个引脚：

- SWCLK：从主机到目标的时钟
- SWDIO：双向

利用该协议，可以同时读取和写入两组寄存器组（DPACC 寄存器组和 APACC 寄存器组）。

传输数据时，LSB 在前。

对于 SWDIO，必须在电路板上对线路进行上拉（ARM 建议采用 100 KΩ）。

每次在协议中更改 SWDIO 的方向时，都会插入转换时间，此时线路即不受主机驱动也不受目标驱动。默认情况下，此转换时间为一位时间，但可以通过配置 SWCLK 频率来调整。

33.8.2 SW 协议序列

每个序列包括三个阶段：

1. 主机发送的数据包请求（8 位）
2. 目标发送的确认响应（3 位）
3. 主机或目标发送的数据传输阶段（33 位）

表 227. 数据包请求（8 位）

位	名称	说明
0	起始	必须为 1
1	APnDP	0：DP 访问 1：AP 访问

表 227. 数据包请求（8 位）（续）

位	名称	说明
2	RnW	0：写请求 1：读请求
4:3	A(3:2)	DP 或 AP 寄存器的地址字段（请参见*表 226*）
5	奇偶校验	前面几位的单位奇偶校验
6	停止	0
7	驻留	不受主机驱动。由于存在上拉，因此必须由目标读为 1

有关 DPACC 和 APACC 寄存器的详细说明，请参见 Cortex™-M4Fr0p1 *TRM*。

数据包请求后面始终为转换时间（默认 1 位），此时主机和目标都不会驱动线路。

表 228. ACK 响应（3 位）

位	名称	说明
0..2	ACK	001：FAULT 010：WAIT 100：OK

仅当发生 READ 事务或者接收到 WAIT 或 FAULT 确认时，ACK 响应后才必须是转换时间。

表 229. DATA 传输（33 位）

位	名称	说明
0..31	WDATA 或 RDATA	写入或读取数据
32	奇偶校验	32 个数据位的单奇偶校验

仅当发生 READ 事务时，DATA 传输后才必须是转换时间。

33.8.3 SW-DP 状态机（复位、空闲状态、ID 代码）

SW-DP 的状态机有一个用于标识 SW-DP 的内部 ID 代码。此代码遵循 JEP-106 标准。此 ID 代码是默认的 ARM 代码，设置为 **0x2BA01477**（相当于Cortex™-M4F r0p1）。

注意： *请注意，在目标读取此 ID 代码前，SW-DP 状态机是不工作的。*

- 在上电复位后、DP 从 JTAG 切换到 SWD 后或者线路处于高电平超过 50 个周期后，SW-DP 状态机处于复位状态。
- 如果在复位状态后线路处于低电平至少两个周期，SW-DP 状态机处于空闲状态。
- 复位状态后，该状态机**必须**首先进入空闲状态，然后对 DP-SW ID CODE 寄存器执行读访问。否则，目标将在另一个事务上发出 FAULT 确认响应。

有关 SW-DP 状态机的更多详细信息，请参见《Cortex™-M4F r0p1 TRM》和《CoreSight 设计套件 r0p1 TRM》。

33.8.4 DP 和 AP 读/写访问

- 不延迟对 DP 的读访问：可以立即发送目标响应（如果 ACK=OK），也可以延迟发送目标响应（如果 ACK=WAIT）。
- 延迟对 AP 的读访问。这意味着会在下次传输时返回访问结果。如果要执行的下次访问不是 AP 访问，则必须读取 DP-RDBUFF 寄存器来获取结果。
  每次进行 AP 读访问或 RDBUFF 读请求时都会更新 DP-CTRL/STAT 寄存器的 READOK 标志，以便了解 AP 读访问是否成功。
- SW-DP 有写缓冲区（用于 DP 或 AP 写入），这样即使在其它操作仍未完成时，也可以接受写入操作。如果写缓冲区已满，则目标确认响应为 WAIT。但 IDCODE 读取、CTRL/STAT 读取或 ABORT 写入除外，这几项操作在写缓冲区已满时也会被接受。
- 由于存在异步时钟域 SWCLK 和 HCLK，因此写操作后（奇偶校验位后）还需要两个额外的 SWCLK 周期，以使写入操作在内部生效。应在将线路驱动为低电平时（空闲状态）应用这些周期。
  在写 CTRL/STAT 寄存器以提出一个上电请求时，这一点特别重要。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。

33.8.5 SW-DP 寄存器

当 APnDP=0 时能够访问这些寄存器

表 230. SW-DP 寄存器

A(3:2)	R/W	SELECT 寄存器的 CTRLSEL 位	寄存器	注释
00	读		IDCODE	制造商代码未设置为 ST 代码。 **0x2BA01477**（标识 SW-DP）
00	写		ABORT	
01	读/写	0	DP-CTRL/STAT	目的： – 请求系统或调试上电 – 配置 AP 访问的传输操作 – 控制比较和验证操作 – 读取一些状态标志（上溢和上电确认）
01	读/写	1	WIRE CONTROL	用于配置物理串行端口协议（如转换时间的持续时间）
10	读		READ RESEND	允许从已损坏的调试软件传输中恢复读取数据，无需重复执行原始 AP 传输
10	写		SELECT	用于选择当前访问端口和活动的 4 字寄存器窗口
11	读/写		READ BUFFER	由于已发出 AP 访问，因此该读缓冲区非常有用（在执行下个 AP 事务时提供读取 AP 请求的结果） 此读取缓冲区捕获 AP 中的数据，显示为前一次读取的结果，无需启动新操作

33.8.6 SW-AP 寄存器

当 APnDP=1 时能够访问这些寄存器

有多个 AP 寄存器（请参见 AHB-AP），这些寄存器按以下组合进行寻址：

- 移位值 A[3:2]
- DP SELECT 寄存器的当前值

33.9 AHB-AP（AHB 访问端口）——对 JTAG-DP 和 SW-DP 同时有效

特性：

- 系统访问与处理器状态无关。
- SW-DP 或 JTAG-DP 访问 AHB-AP。
- AHB-AP 是总线矩阵内的 AHB 主设备。因此，它可以访问所有数据总线（Dcode 总线、系统总线、内部和外部 PPB 总线），但不能访问 ICode 总线。
- 支持位寻址的传输。
- 旁路 FPB 的 AHB-AP 传输。

32 位 AHP-AP 寄存器的地址宽度为 6 位（最多达 64 字或 256 字节），其中包括：

c) 位 [7:4] = DP SELECT 寄存器的位 [7:4] APBANKSEL

d) 位 [3:2] = SW-DP 的 35 位数据包请求的 2 个地址位 A(3:2)。

Cortex™-M4F 的 AHB-AP 包括 9 个 32 位寄存器：

表 231. Cortex™-M4F AHB-AP 寄存器

偏移地址	寄存器名称	注释
0x00	AHB-AP 控制和状态字	配置和控制通过 AHB 接口的传输（大小、hprot、当前传输的状态、地址递增类型）
0x04	AHB-AP 传输地址	
0x0C	AHB-AP 数据读/写	
0x10	AHB-AP 分组数据 0	直接访问 4 个相连的字而不用重写访问地址
0x14	AHB-AP 分组数据 1	
0x18	AHB-AP 分组数据 2	
0x1C	AHB-AP 分组数据 3	
0xF8	AHB-AP 调试 ROM 地址	调试接口的基址
0xFC	AHB-AP ID 寄存器	

有关更多详细信息，请参见《Cortex™-M4F r0p1 TRM》。

33.10 内核调试

通过内核调试寄存器调试内核。对这些寄存器的调试访问通过*先进高性能总线* (AHB-AP) 端口进行。处理器可通过内部 *专用外设总线* (PPB) 直接访问这些寄存器。

它由 4 个寄存器组成：

表 232. 内核调试寄存器

寄存器	说明
DHCSR	*32 位调试控制和状态寄存器：* 此寄存器提供有关处理器状态的信息，能够使内核进入调试停止状态并提供处理器步进功能
DCRSR	*17 位调试内核寄存器选择器寄存器：* 此寄存器选择需要进行读写操作的内核寄存器。
DCRDR	*32 位调试内核寄存器数据寄存器：* 此寄存器保存在寄存器与 DCRSR（选择器）寄存器选择的处理器之间读取和写入的数据。
DEMCR	*32 位调试异常和监视控制寄存器：* 此寄存器提供向量捕获和调试监视控制。此寄存器包含一个名为 ***TRCENA*** 的位，该位启动 TRACE 功能。

注意： ***重要提示：****这些寄存器在系统复位时不复位。它们只能通过上电复位来复位。*

有关更多详细信息，请参见《*Cortex™-M4F r0p1 TRM*》。

为了在复位后立即使内核进入调试状态，必须：

- 使能调试和异常监视控制寄存器的位 0 (VC_CORRESET)
- 使能调试停止控制和状态寄存器的位 0 (C_DEBUGEN)

33.11 调试主机在系统复位状态下建立连接的功能

STM32F4xx MCU 的复位系统由以下复位源组成：

- POR（上电复位），在每次上电时将执行复位
- 内部看门狗复位
- 软件复位
- 外部复位

Cortex™-M4F 将调试部分的复位（通常为 PORRESETn）和其它部分 (SYSRESETn) 的复位分开

这样，调试主机便能够在系统复位期间建立连接，对内核调试寄存器进行编程，以在获取复位向量时停止内核。随后，调试主机释放系统复位，内核将立即停止并且不执行任何指令。此外，还可以在内核处于复位状态下时配置调试特性。

注意： *强烈建议调试主机在系统复位状态下建立连接（在复位向量处设置断点）。*

33.12 FPB (Flash patch breakpoint)

FPB 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。利用此功能可以修正位于代码存储空间中的软件错误。

软件补丁和硬件断点都必须单独使用。

FPB 包括：

- 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。

33.13 DWT（数据观察点触发）

DWT 单元由四个比较器组成。这些比较器可配置为：

- 硬件观察点或
- ETM 的触发或
- PC 采样器或
- 数据地址采样器

DWT 还能以某种方式提供一些概要信息。为此，可访问某些计数器以获得以下数据：

- 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

33.14 ITM（指令跟踪宏单元）

33.14.1 概述

ITM 是应用驱动的跟踪源，支持 *printf* 类的调试以跟踪*操作系统* (OS) 和应用程序事件，并发布系统信息诊断。ITM 以数据包形式发送跟踪信息，这些信息包括：

- **软件跟踪。**软件可以直接写入 ITM 激励寄存器以发布包信息。
- **硬件跟踪。**DWT 生成数据包，然后 ITM 发送这些数据包。
- **时间戳。**发送与数据包相关的时间戳。ITM 包含 21 位计数器，用于生成时间戳。计数器由Cortex™-M4F 时钟或*串行线查看器* (SWV) 输出的位时钟驱动。

ITM 发送的数据包输出到 TPIU（跟踪端口接口单元）。TPIU 的格式化器添加一些额外的数据包（请参见 TPIU），然后将整个数据包序列输出到调试主机。

在设置或使用 ITM 前，必须使能调试异常和监视控制寄存器的位 TRCEN。

33.14.2 时间戳数据包、同步和溢出数据包

时间戳数据包会对时间戳信息、通用控制和同步进行编码。它使用 21 位时间戳计数器（带可能的预分频器），该计数器在每次发送时间戳数据包时复位。此计数器可由 CPU 时钟或 SWV 时钟驱动。

同步数据包由 6 个字节组成，等于 0x80_00_00_00_00_00，以 00 00 00 00 00 80 形式（首先发送 LSB）发送到 TPIU。

同步数据包是时间戳数据包的控制信号。它在每次触发 DWT 时发送。

为此，DWT 必须配置为触发 ITM：必须将 DWT 控制寄存器的位 CYCCNTENA（位 0）置 1。此外，还必须将 ITM 跟踪控制寄存器的位 2 (SYNCENA) 置 1。

注意： *如果未将 SYNENA 位置 1，DWT 产生给 TPIU 的同步触发，将仅发送 TPIU 同步数据包，不发送 ITM 同步数据包。*

溢出数据包是一种特殊的时间戳数据包，用于指示已写入数据但 FIFO 已满。

表 233. 主要的 ITM 寄存器

地址	寄存器	详细信息
@E0000FB0	ITM 锁定访问	写入 0xC5ACCE55 以解锁对其它 ITM 寄存器的写访问
@E0000E80	ITM 跟踪控制	位 31-24 = 始终为 0
		位 23 = 忙碌
		位 22-16 = 7 位 ATB ID，用于标识跟踪数据源
		位 15-10 = 始终为 0
		位 9:8 = TSPrescale = 时间戳预分频器
		位 7-5 = 保留
		位 4 = SWOENA = 使能 SWV 行为（由 SWV 时钟驱动时间戳计数器）
		位 3 = DWTENA：使能 DWT 激励
		位 2 = SYNCENA：此位必须置为 1 以使能 DWT，进而生成同步触发，这样 TPIU 便可发送同步数据包
		位 1 = TSENA（时间戳使能）
		位 0 = ITMENA：ITM 的全局使能位
@E0000E40	ITM 跟踪特权	位 3：置 1 以使能跟踪端口 31:24
		位 2：置 1 以使能跟踪端口 23:16
		位 1：置 1 以使能跟踪端口 15:8
		位 0：置 1 以使能跟踪端口 7:0
@E0000E00	ITM 跟踪使能	每个位使能相应的激励端口以产生跟踪
@E0000000-E000007C	激励端口寄存器 0-31	在要跟踪的选定激励端口上写入 32 位数据（32 个可用位）

配置示例

向 TPIU 输出一个简单值：

- 通过配置 DBGMCU_CR 来配置 TPIU 并分配 TRACE I/O（请参见*第 33.17.2 节：TRACE 引脚分配*和*第 33.16.3 节：MCU 调试配置寄存器*）
- 将 0xC5ACCE55 写入 ITM 锁定访问寄存器，以解锁对 ITM 寄存器的写保护
- 将 0x00010005 写入 ITM 跟踪控制寄存器，以使能 ITM（使能同步，ATB ID 不为 0x00）
- 将 0x1 写入 ITM 跟踪使能寄存器，以使能激励端口 0
- 将 0x1 写入 ITM 跟踪特权寄存器，以取消屏蔽激励端口 7:0
- 将要输出的值写入激励端口寄存器 0：此操作可用软件完成（使用 printf 函数）

33.15 ETM（嵌入式跟踪宏单元）

33.15.1 概述

ETM 能够重建程序执行。使用“数据观察点和跟踪”(DWT) 组件或“指令跟踪宏单元”(ITM) 跟踪数据，而使用“嵌入式跟踪宏单元”(ETM) 跟踪指令。

ETM 以数据包形式发送信息，由内置资源触发。这些资源必须单独编程，并且使用“触发事件寄存器”(0xE0041008) 选择触发源。事件可以是简单事件（地址比较器中的地址匹配），也可以是 2 个事件之间的逻辑运算结果。触发源是 DWT 模块的四个比较器之一，可以监视以下事件：

- 时钟周期匹配
- 数据地址匹配

有关触发资源的更多信息，请参见*第 33.13 节：DWT（数据观察点触发）*。

ETM 发送的数据包输出到 TPIU（跟踪端口接口单元）。TPIU 会添加一些额外的数据包（请参见*第 33.17 节：TPIU（跟踪端口接口单元）*），然后将整个数据包序列输出到调试主机。

33.15.2 信号协议和数据包类型

ARM IHI 0014N 文档的第 7 章“ETMv3 信号协议”中介绍了这部分内容。

33.15.3 主要的 ETM 寄存器

有关寄存器的更多信息，请参见 ARM IHI 0014N 规范的第 3 章。

表 234. 主要的 ETM 寄存器

地址	寄存器	详细信息
0xE0041FB0	ETM 锁定访问	写入 0xC5ACCE55 以解锁对其它 ETM 寄存器的写访问。
0xE0041000	ETM 控制	此寄存器控制 ETM 的一般操作，例如如何使能跟踪。
0xE0041010	ETM 状态	此寄存器提供有关跟踪和触发逻辑的当前状态的信息。
0xE0041008	ETM 触发事件	此寄存器定义控制触发的事件。
0xE004101C	ETM 跟踪使能控制	此寄存器定义选择哪个比较器。
0xE0041020	ETM 跟踪使能事件	此寄存器定义跟踪使能事件。
0xE0041024	ETM 跟踪启动/停止	此寄存器定义触发源启动和停止跟踪时分别使用的跟踪。

33.15.4 配置示例

从 TPIU 输出一个简单的数值：

- 配置 TPIU 并使能 I/IO_TRACEN，以分配 STM32F4xx 调试配置寄存器中的 TRACE I/O
- 将 0xC5ACCE55 写入 ETM 锁定访问寄存器，以解锁对 ITM 寄存器的写保护
- 将 0x00001D1E 写入控制寄存器（配置跟踪）
- 将 0000406F 写入触发事件寄存器（定义触发事件）
- 将 0000006F 写入跟踪使能事件寄存器（定义要启动/停止的事件）
- 将 00000001 写入跟踪启动/停止寄存器（使能跟踪）
- 将 0000191E 写入 ETM 控制寄存器（配置结束）

33.16 MCU 调试组件 (DBGMCU)

MCU 调试组件帮助调试器为以下各项提供支持：

- 低功耗模式
- 断点期间的定时器、看门狗、 I2C 和 bxCAN 的时钟控制
- 跟踪引脚分配的控制

33.16.1 对低功耗模式的调试支持

要进入低功耗模式，必须执行指令 WFI 或 WFE。

MCU 支持多个低功耗模式，这些模式可以禁止 CPU 时钟或降低 CPU 功耗。

内核不允许在调试会话期间关闭 FCLK 或 HCLK。由于调试期间需要使用它们进行调试连接，因此其必须保持激活状态。MCU 集成了特殊方法，允许用户在低功耗模式下调试软件。

为此，调试主机首先必须设置一些调试配置寄存器，以更改低功耗模式行为：

- 在睡眠模式下，调试主机必须事先将 DBGMCU_CR 寄存器的 DBG_SLEEP 位置 1。这样便可为 HCLK 和 FCLK 提供相同的时钟（之前配置的系统时钟）。
- 在停止模式下，调试主机必须事先将 DBG_STOP 位置 1。这样便可使能内部 RC 振荡器时钟，以在停止模式下为 FCLK 和 HCLK 提供时钟。

33.16.2 对定时器、看门狗、bxCAN 和 I^2C 的调试支持

断点期间，必须选择定时器和看门狗的计数器的行为方式：

- 在产生断点时，计数器继续计数。例如，当 PWM 控制电机时，通常需要这种方式。
- 在产生断点时，计数器停止计数。用于看门狗时需要这种方式。

对于 bxCAN，用户可以选择在断点期间阻止更新接收寄存器。

对于 I^2C，用户可以选择在断点期间阻止 SMBUS 超时。

33.16.3 MCU 调试配置寄存器

Debug MCU configuration register

可利用此寄存器在调试状态下配置 MCU。其中涉及：

- 低功耗模式支持
- 定时器和看门狗计数器支持
- bxCAN 通信支持
- TRACE 引脚分配

此 DBGMCU_CR 映射到外部 PPB 总线的地址 0xE0042004

它通过 PORESET（而非系统复位）异步复位。可在系统复位状态下用调试器写入该寄存器。

如果调试主机不支持这些功能，仍然可以通过用户软件写入这些寄存器。

DBGMCU_CR

地址：0xE004 2004

仅支持 32 位访问

POR 复位：0x0000 0000（不会被系统复位复位）

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								TRACE_MODE [1:0]		TRACE_IOEN	Reserved		DBG_STANDBY	DBG_STOP	DBG_SLEEP
								rw	rw	rw			rw	rw	rw

位 31:8 保留，必须保持复位值。

位 7:5 **TRACE_MODE[1:0] 和 TRACE_IOEN**：跟踪引脚分配控制 (Trace pin assignment control)

– *TRACE_IOEN=0 时：*

TRACE_MODE=xx：未分配 TRACE 引脚（默认状态）

– *TRACE_IOEN=1 时：*

– TRACE_MODE=00：异步模式下的 TRACE 引脚分配

– TRACE_MODE=01：同步模式下的 TRACE 引脚分配，TRACEDATA 大小为 1

– TRACE_MODE=10：同步模式下的 TRACE 引脚分配，TRACEDATA 大小为 2

– TRACE_MODE=11：同步模式下的 TRACE 引脚分配，TRACEDATA 大小为 4

位 4:3 保留，必须保持复位值。

位 2 **DBG_STANDBY**：调试待机模式 (Debug Standby mode)

0：（FCLK=关闭，HCLK=关闭）将整个数字部分断电。

从软件角度来看，退出待机模式相当于复位（指示 MCU 从待机状态恢复的几个状态位除外）

1：（FCLK=开启，HCLK=开启）在这种情况下，数字部分不断电，FCLK 和 HCLK 由仍保持激活状态的内部 RC 振荡器提供。此外，MCU 会在待机模式期间产生系统复位，这样退出待机模式相当于复位

位 1 **DBG_STOP**：调试停止模式 (Debug Stop mode)

0：（FCLK=关闭，HCLK=关闭）在停机模式下，时钟控制器禁止所有时钟（包括 HCLK 和 FCLK）。退出停机模式时，时钟配置与复位后的时钟配置相同（CPU 时钟由 8 MHz 内部 RC 振荡器 (HSI) 提供）。因此，软件必须重新编程时钟控制器以使能 PLL 和晶振等。

1：（FCLK=开启，HCLK=开启）在这种情况下，进入停机模式时，FCLK 和 HCLK 由在停机模式下仍保持激活状态的内部 RC 振荡器提供。退出停机模式时，软件必须重新编程时钟控制器以使能 PLL 和晶振等（操作步骤与在 DBG_STOP=0 时相同）

位 0 **DBG_SLEEP**：调试睡眠模式 (Debug Sleep mode)

0：（FCLK=开启，HCLK=关闭）在睡眠模式下，FCLK 由原先配置好的系统时钟驱动。在睡眠模式下，时钟控制器配置不复位，保持先前设置的状态。因此，退出睡眠模式时，软件不需要重新配置时钟控制器。

1：（FCLK=开启，HCLK=开启）在这种情况下，进入睡眠模式时，将为 HCLK 和 FCLK 馈送相同的时钟（软件先前配置的系统时钟）。

33.16.4 MCU APB1 调试冻结寄存器 (DBGMCU_APB1_FZ)

Debug MCU APB1 freeze register

在调试状态下，使用 DBGMCU_APB1_FZ 寄存器配置 MCU。它涉及 APB1 外设。它映射到外部 PPB 总线的地址 0xE004 2008。

寄存器通过 POR（而非系统复位）异步复位。可在系统复位状态下用调试器写入该寄存器。

地址：0xE004 2008

仅支持 32 位访问。

上电复位 (POR)：0x0000 0000（不会被系统复位信号复位）

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved					DBG_CAN2_STOP	DBG_CAN1_STOP	Reserved	DBG_I2C3_SMBUS_TIMEOUT	DBG_I2C2_SMBUS_TIMEOUT	DBG_I2C1_SMBUS_TIMEOUT	Reserved				
					rw	rw		rw	rw	rw					

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			DBG_IWDG_STOP	DBG_WWDG_STOP	DBG_RTC_STOP	Reserved	DBG_TIM14_STOP	DBG_TIM13_STOP	DBG_TIM12_STOP	DBG_TIM7_STOP	DBG_TIM6_STOP	DBG_TIM5_STOP	DBG_TIM4_STOP	DBG_TIM3_STOP	DBG_TIM2_STOP
				rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:27　保留，必须保持复位值。

位 26 **DBG_CAN2_STOP**：内核停止时停止调试 CAN2 (Debug CAN2 stopped when Core is halted)
0：行为方式与正常模式下相同
1：冻结 CAN2 接收寄存器

位 25 **DBG_CAN1_STOP**：内核停止时停止调试 CAN1 (Debug CAN1 stopped when Core is halted)
0：行为方式与正常模式下相同
1：冻结 CAN1 接收寄存器

位 24　保留，必须保持复位值。

位 23 **DBG_I2C3_SMBUS_TIMEOUT**：内核停止时停止 SMBUS 超时模式 (SMBUS timeout mode stopped when Core is halted)
0：行为方式与正常模式下相同
1：冻结 SMBUS 超时

位 22 **DBG_I2C2_SMBUS_TIMEOUT**：内核停止时停止 SMBUS 超时模式 (SMBUS timeout mode stopped when Core is halted)
0：行为方式与正常模式下相同
1：冻结 SMBUS 超时

位 21 **DBG_I2C1_SMBUS_TIMEOUT**：内核停止时停止 SMBUS 超时模式 (SMBUS timeout mode stopped when Core is halted)
0：行为方式与正常模式下相同
1：冻结 SMBUS 超时

位 20:13　保留，必须保持复位值。

位 12 **DBG_IWDG_STOP**：内核停止时停止独立看门狗 (Debug independent watchdog stopped when core is halted)
0：即使内核停止，独立看门狗计数器时钟仍继续工作
1：内核停止时，独立看门狗计数器时钟停止

位 11 **DBG_WWDG_STOP**：内核停止时停止窗口看门狗 (Debug Window Watchdog stopped when Core is halted)
0：即使内核停止，窗口看门狗计数器时钟仍继续工作
1：内核停止时，窗口看门狗计数器时钟停止

位 10 **DBG_RTC_STOP**：内核停止时停止 RTC (RTC stopped when Core is halted)
0：即使内核停止，RTC 计数器时钟仍继续工作
1：内核停止时，RTC 计数器时钟停止

位 9　保留，必须保持复位值。

位 8:0 **DBG_TIMx_STOP**：内核停止时 TIMx 计数器停止 (TIMx counter stopped when core is halted) (x=2..7, 12..14)
0：即使内核停止，仍然馈送相关定时器计数器的时钟
1：内核停止时停止相关定时器计数器的时钟

33.16.5 MCU APB2 调试冻结寄存器 (DBGMCU_APB2_FZ)

Debug MCU APB2 Freeze register

在调试状态下，使用 DBGMCU_APB2_FZ 寄存器配置 MCU。它涉及 APB2 外设。

此寄存器映射到外部 PPB 总线的地址 0xE004 200C。

它通过 POR（而非系统复位）异步复位。可在系统复位状态下用调试器写入该寄存器。

地址：0xE004 200C

仅支持 32 位访问。

POR：0x0000 0000（不会被系统复位信号复位）

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved													DBG_TIM11_STOP	DBG_TIM10_STOP	DBG_TIM9_STOP
													rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														DBG_TIM8_STOP	DBG_TIM1_STOP
														rw	rw

位 31:19　保留，必须保持复位值。

位 18:16　**DBG_TIMx_STOP**：内核停止时 TIMx 计数器停止 (TIMx counter stopped when core is halted) (x=9..11)

0：即使内核停止，仍然馈送相关定时器计数器的时钟

1：内核停止时停止相关定时器计数器的时钟

位 15:2　保留，必须保持复位值。

位 1　**DBG_TIM8_STOP**：内核停止时 TIM8 计数器停止 (TIM8 counter stopped when core is halted)

0：即使内核停止，仍然馈送相关定时器计数器的时钟

1：内核停止时停止相关定时器计数器的时钟

位 0　**DBG_TIM1_STOP**：内核停止时 TIM1 计数器停止 (TIM1 counter stopped when core is halted)

0：即使内核停止，仍然馈送相关定时器计数器的时钟

1：内核停止时停止相关定时器计数器的时钟

33.17 TPIU（跟踪端口接口单元）

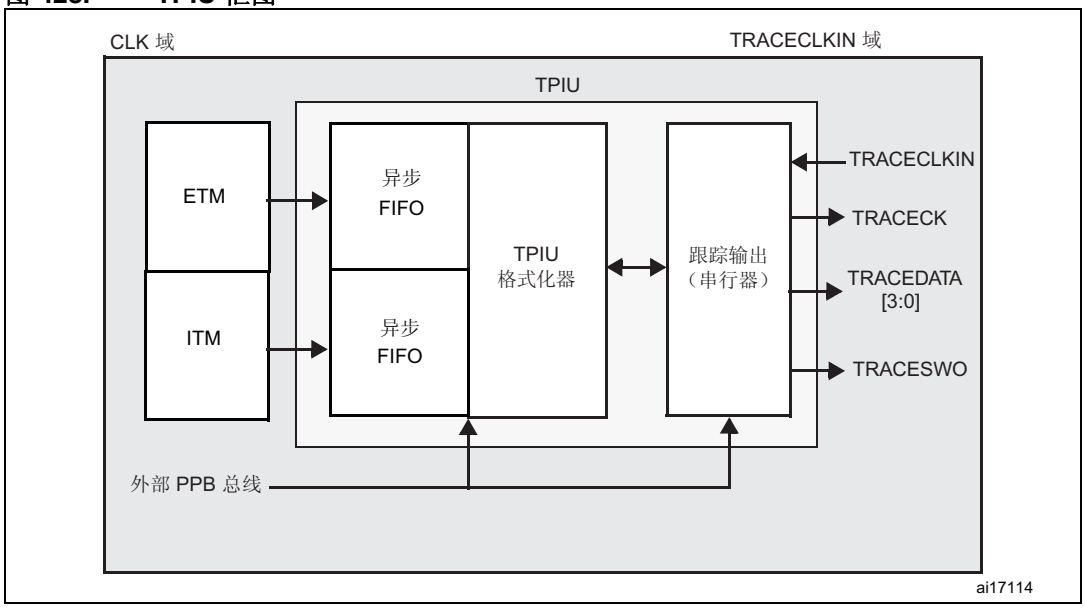
33.17.1 前言

TPIU 用作 ITM 和 ETM 中的片上跟踪数据之间的桥接器。

输出数据流封装成跟踪源 ID，然后被*跟踪端口分析器* (TPA) 捕获。

内核内置专为低成本调试设计的简单 TPIU（由特殊版本的 CoreSight TPIU 组成）。

图 428. TPIU 框图



33.17.2 TRACE 引脚分配

- 异步模式

  异步模式需要 1 个额外引脚，并且适用于所有封装。仅在使用串线行模式时异步模式才可用（在 JTAG 模式下不可用）。

表 235. 异步 TRACE 引脚分配

TPUI 引脚名称	跟踪同步模式		STM32F4xx 引脚分配
	类型	说明	
TRACESWO	O	TRACE 异步数据输出	PB3

- 同步模式

  同步模式需要 2 到 6 个额外引脚，具体取决于所跟踪数据的长度，并且仅适用于较大型的封装。此外，同步模式在 JTAG 模式和串行模式下均可用，并可提供比异步跟踪更高的带宽输出能力。

表 236. 同步 TRACE 引脚分配

TPUI 引脚名称	跟踪同步模式		STM32F4xx 引脚分配
	类型	说明	
TRACECK	O	TRACE 时钟	PE2
TRACED[3:0]	O	TRACE 同步数据输出 可以是 1、2 或 4。	PE[6:3]

TPUI TRACE 引脚分配

默认情况下，不分配这些引脚。可通过将 ***MCU 调试组件配置寄存器***中的 TRACE_IOEN 和 TRACE_MODE 位置 1 来分配这些引脚。必须由调试主机来完成此配置。

此外，要分配的引脚数目取决于跟踪配置（异步跟踪或同步跟踪）。

- **异步模式**：需要 1 个额外引脚
- **同步模式**：需要 2 到 5 个额外引脚，具体数目取决于数据跟踪端口寄存器的数据长度（1、2 或 4）：
  - TRACECK
  - TRACED(0)（如果端口数据长度配置为 1、2 或 4）
  - TRACED(1)（如果端口数据长度配置为 2 或 4）
  - TRACED(2)（如果端口数据长度配置为 4）
  - TRACED(3)（如果端口数据长度配置为 4）

要分配 TRACE 引脚，调试主机必须对 MCU 调试配置寄存器 (DBGMCU_CR) 的位 TRACE_IOEN 和 TRACE_MODE[1:0] 进行编程。默认情况下不分配 TRACE 引脚。

此寄存器映射到外部 PPB 上，通过 PORESET（而非系统复位）复位。可在系统复位状态下用调试器写入该寄存器。

表 237. 灵活的 TRACE 引脚分配

DBGMCU_CR 寄存器		引脚用途	分配的 TRACE IO 引脚					
TRACE_IOEN	TRACE_MODE[1:0]		PB3 / JTDO/ TRACESWO	PE2 / TRACECK	PE3 / TRACED[0]	PE4 / TRACED[1]	PE5 / TRACED[2]	PE6 / TRACED[3]
0	XX	无跟踪（默认状态）	已释放 (1)					
1	00	异步跟踪	TRACESWO			已释放（可用作 GPIO）		
1	01	同步跟踪 1 位	已释放 (1)	TRACECK	TRACED[0]			
1	10	同步跟踪 2 位		TRACECK	TRACED[0]	TRACED[1]		
1	11	同步跟踪 4 位		TRACECK	TRACED[0]	TRACED[1]	TRACED[2]	TRACED[3]

1. 使用串行模式时，释放此引脚。但使用 JTAG 时，此引脚分配给 JTDO。

注意： *默认情况下，TPIU 的 TRACECLKIN 输入时钟接地。因此，将 TRACE_IOEN 位置 1 后再经过两个时钟周期，会将该时钟分配给 HCLK。*

调试器必须通过对 TPIU 的 SPP_R（选择引脚协议）寄存器的 PROTOCOL[1:0] 位执行写操作来配置跟踪模式。

- PROTOCOL=00：跟踪端口模式（同步）
- PROTOCOL=01 或 10：串行（曼彻斯特或 NRZ）模式（异步模式）。默认状态为 01

随后，它还会通过对 TPIU 的 CPSPS_R（当前同步端口大小寄存器）中的位 [3:0] 执行写操作来配置跟踪端口大小：

- 0x1 表示 1 个引脚（默认状态）
- 0x2 表示 2 个引脚
- 0x8 表示 4 个引脚

33.17.3 TPUI 格式化器

格式化器协议以 16 字节帧形式输出数据：

- 七个字节的数据
- 八个字节的多用字节，包括：
  - 1 位 (LSB) 指示该字节是数据字节 (‘0) 还是 ID 字节 (‘1)。
  - 7 位 (MSB)，既可以是数据，也可用于更改源 ID 跟踪。
- 一个字节的辅助位，每个辅助位对应于八个多用字节中的一个：
  - 如果对应的字节是数据，则此位表示该数据的位 0。
  - 如果对应的字节是 ID 更改，则此位表示 ID 更改生效的时间。

注意： *有关更多信息，请参见《ARM CoreSight 架构规范 v1.0》(ARM IHI 0029B)*

33.17.4 TPUI 帧同步数据包

TPUI 可产生两种类型的同步数据包：

- 帧同步数据包（或全字同步数据包）

  它由以下字组成：0x7F_FF_FF_FF（首先发送 LSB）。只要未使用 ID 源代码 0x7F，就不会在任何其它时间出现此序列。

  它会在各数据帧*之间*定期输出。

  在连续模式下，找到同步帧后，TPA 必须丢弃所有这些帧。

- 半字同步数据包

  它由以下半字组成：0x7F_FF（首先发送 LSB）。

  它会在各数据帧*之间或内部*定期输出。

  这些数据包仅在连续模式下生成，能够使 TPA 检测到跟踪端口是否处于空闲模式（没有要捕获的 TRACE）。当 TPA 检测到这些数据包后，必须将其丢弃。

33.17.5 发送同步帧数据包

内核的 TPIU 中没有同步计数器寄存器。因此，只能通过 **DWT** 生成同步触发。请参见寄存器 DWT 控制寄存器（位 SYNCTAP[11:10]）和 DWT 当前 PC 采样器循环计数寄存器。

发送 TPUI 帧同步数据包 (0x7F_FF_FF_FF) 的条件：

- 每次 TPIU 复位释放后。此复位在 TRACECLKIN 时钟出现上升沿时同步释放。这意味着，当 DBGMCU_CFG 寄存器中的 TRACE_IOEN 位置 1 时传送此数据包。在这种情况下，字 0x7F_FF_FF_FF 后面不带任何格式化的数据包。
- 在每次触发 DWT 时（假设先前已配置 DWT）。存在两种情况：
  - 如果复位 ITM 的 SYNENA 位复位，则仅发送字 0x7F_FF_FF_FF，后面不带任何格式化的数据流。
  - 如果 ITM 的 SYNENA 位置 1，则 ITM 同步数据包将跟在 (0x80_00_00_00_00_00) 后面，并由 TPUI 格式化（添加跟踪源 ID）。

33.17.6 同步模式

跟踪数据输出大小可配置为 4 个、2 个或 1 个引脚：TRACED(3:0)

输出时钟输出到调试器 (TRACECK)

其中，TRACECLKIN 由内部驱动，并且仅在使用 TRACE 时连接到 HCLK。

注意： *在此同步模式下，不需要提供稳定的时钟频率。*

TRACE I/O（包括 TRACECK）由 TRACLKIN 的上升沿驱动（等于 HCLK）。因此，TRACECK 的输出频率等于 HCLK/2。

33.17.7 异步模式

这是一种只使用 1 个引脚输出跟踪的低成本备用方案：该引脚是异步输出引脚 TRACESWO。显然，这种方案的带宽有限。

使用 SW-DP 引脚时，TRACESWO 与 JTDO 复用。这样，所有 STM32F4xx 封装都可提供此功能。

此异步模式要求 TRACECLKIN 具有恒定的频率。对于标准 UART (NRZ) 捕获机制，要求 5% 精度。曼彻斯特编码的容限最高可达 10%。

33.17.8 STM32F4xx 内的 TRACECLKIN 连接

在 STM32F4xx 中，此 TRACECLKIN 输入内部连接到 HCLK。这意味着，在异步跟踪模式下，应用程序应限制使用时间帧保证 CPU 频率的稳定。

注意： ***重要提示：****使用异步跟踪时，了解以下内容非常重要：*

STM32F4xx MCU 的默认时钟是内部 RC 振荡器。复位状态下的频率与复位释放后的频率不同。原因是，由于系统复位状态下默认采用 RC 校准值，而在每次系统复位释放时会更新该 RC 校准值。

因此，跟踪端口分析器 (TPA) 在系统复位状态下不应使能跟踪（使用 TRACE_IOEN 位），原因是，在复位状态下的同步帧包的比特宽度与复位后的包不同。

33.17.9 TPIU 寄存器

仅当调试异常和监视控制寄存器 (DEMCR) 的位 TRCENA 置 1 时才能对 TPIU APB 寄存器进行读写操作。否则，这些寄存器将读为零（此位的输出会使能 TPIU 的 PCLK）。

表 238. 重要的 TPIU 寄存器

地址	寄存器	说明
0xE0040004	当前端口大小	允许选择跟踪端口大小： 位 0：端口大小 = 1 位 1：端口大小 = 2 位 2：端口大小 = 3，不支持 位 3：端口大小 = 4 只有 1 位必须置 1。默认情况下，端口大小为一位。(0x00000001)
0xE00400F0	选择引脚协议	允许选择跟踪端口协议： 位 1:0= 00：同步跟踪端口模式 01：串行线输出 - 曼彻斯特（默认值） 10：串行线输出 - NRZ 11：保留
0xE0040304	格式化器和刷新控制	位 31-9 = 始终为 0 位 8 = TrigIn = 始终为 1，表明已指示这些触发 位 7-4 = 始终为 0 位 3-2 = 始终为 0 位 1 = EnFCont。在同步跟踪模式下（选择引脚协议寄存器位 1:0=00），此位会强制置为 1：在连续模式下自动使能格式化器。在异步模式下（选择引脚协议寄存器位 1:0 <> 00），可写入该位以激活或取消激活格式化器。 位 0 = 始终为 0 产生的默认值为 0x102 **注意：**在同步模式下，由于不会在片外映射 TRACECTL 引脚，因此在连续模式下将始终使能格式化器。这样，格式化器便可插入一些控制数据包来标识跟踪数据包的源。
0xE0040300	格式化器和刷新状态	不适用于 Cortex™-M4F，始终读为 0x00000008

33.17.10 配置示例

- 将调试异常和监视控制寄存器 (DEMCR) 中的位 TRCENA 置 1
- 将所需值写入 TPIU 当前端口大小寄存器（对于 1 位端口大小，默认值为 0x1）
- 将 0x102 写入 TPIU 格式化器和刷新控制寄存器（默认值）
- 写入 TPIU 选择引脚协议以选择同步模式或异步模式。示例：0x2 表示异步 NRZ 模式（类似于 UART）
- 将 0x20 写入 DBGMCU 控制寄存器（位 IO_TRACEN），为异步模式分配 TRACE I/O。此时发送 TPIU 同步数据包 (FF_FF_FF_7F)
- 配置 ITM 并对 ITM 激励寄存器进行写操作以输出值

33.18 DBG 寄存器映射

下表对调试寄存器进行了汇总

表 239. DBG 寄存器映射和复位值

地址	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xE004 2000	**DBGMCU_IDCODE**	REV_ID																Reserved				DEV_ID											
	Reset value[1]	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X					X	X	X	X	X	X	X	X	X	X	X	X
0xE004 2004	**DBGMCU_CR**	Reserved											DBG_TIM7_STOP	DBG_TIM6_STOP	DBG_TIM5_STOP	DBG_TIM8_STOP	DBG_I2C2_SMBUS_TIMEOUT	Reserved								TRACE_MODE [1:0]		TRACE_IOEN	Reserved		DBG_STANDBY	DBG_STOP	DBG_SLEEP
	Reset value												0	0	0	0	0									0	0	0			0	0	0
0xE004 2008	**DBGMCU_APB1_FZ**	Reserved					DBG_CAN2_STOP	DBG_CAN1_STOP	Reserved	DBG_I2C3_SMBUS_TIMEOUT	DBG_I2C2_SMBUS_TIMEOUT	DBG_I2C1_SMBUS_TIMEOUT	Reserved								DBG_IWDG_STOP	DBG_WWDG_STOP	Reserved	DBG_RTC_STOP	DBG_TIM14_STOP	DBG_TIM13_STOP	DBG_TIM12_STOP	DBG_TIM7_STOP	DBG_TIM6_STOP	DBG_TIM5_STOP	DBG_TIM4_STOP	DBG_TIM3_STOP	DBG_TIM2_STOP
	Reset value						0	0	0	0	0	0									0	0		0	0	0	0	0	0	0	0	0	0
0xE004 200C	**DBGMCU_APB2_FZ**	Reserved													DBG_TIM11_STOP	DBG_TIM10_STOP	DBG_TIM9_STOP	Reserved													DBG_TIM8_STOP	DBG_TIM1_STOP
	Reset value														0	0	0														0	0

1. 复位值与产品有关。有关详细信息，请参见*第 33.6.1 节：MCU 器件 ID 代码*。

34 设备电子签名

电子签名存储在 Flash 区。可以使用 JTAG/SWD 或 CPU 对其进行读取。它包含出厂前编程的标识数据，这些标识数据允许用户固件或其它外部设备将其接口与 STM32F4xx 微控制器的特性自动匹配。

34.1 唯一设备 ID 寄存器（96 位）

唯一设备标识符最适合：

- 用作序列号（例如 USB 字符串序列号或其它终端应用程序）
- 在对内部 Flash 进行编程前将唯一 ID 与软件加密原语和协议结合使用时用作安全密钥以提高 Flash 中代码的安全性
- 激活安全自举过程等

96 位的唯一设备标识符提供了一个对于任何设备和任何上下文都唯一的参考号码。用户永远不能改变这些位。

96 位的唯一设备标识符也可以以单字节/半字/字等不同方式读取，然后使用自定义算法连接起来。

基址：0x1FFF 7A10

偏移地址：0x00

只读 = 0xXXXX XXXX，其中 X 是出厂前编程的

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

位 31:0 **U_ID(31:0)**：31:0 唯一 ID 位

偏移地址：0x04

只读 = 0xXXXX XXXX，其中 X 是出厂前编程的

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
U_ID(63:48)															
r	r	r	r	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
U_ID(47:32)															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

位 31:0 **U_ID(63:32)**：63:32 唯一 ID 位

偏移地址：0x08

只读 = 0xXXXX XXXX，其中 X 是出厂前编程的

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
U_ID(95:80)															
r	r	r	r	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
U_ID(79:64)															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

位 31:0 **U_ID(95:64)**：*95:64 唯一 ID 位*

34.2 Flash 大小

基址：0x1FFF 7A22

偏移地址：0x00

只读 = 0xXXXX，其中 X 是出厂前编程的

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

位 15:0 F_ID(15:0)：Flash 大小

此处的位字段指示以 KB 表示的设备 Flash 大小。

例如，0x0400 对应于 1024 KB。

A

C

D

E

F

G

H

I

O

P

R

S

T

U

W

版本历史

表 240. 文档版本历史

日期	版本	变更
2011 年 09 月 15 日	1	初始版本。
2012 年 10 月 19 日	2	封面页更新了参考文档并增加了*表 1：适用产品*。 **存储器：** 更新了*第 2.3.1 节：嵌入式 SRAM*。 **PWR：** *图 7：电源概述*更新了 VDDA 和 VREF+ 去耦电容。 *第 5.1.2 节：电池备份域*更新了无外部电池的情况。 *第 5.4.3 节：PWR 电源控制/状态寄存器 (PWR_CSR)* 中 VOSRDY 位变更为只读。 删除了*第 5.2.3 节：可编程电压检测器 (PVD)* 中的 VDDA，还删除了 PVDO 位说明中的 VDDA (*第 5.4.3 节：PWR 电源控制/状态寄存器 (PWR_CSR)*)。 **RCC：** 更新了*图 12：复位电路简图*和最小复位脉冲持续时间（由脉冲发生器保证且仅限内部复位源）。 **GPIO：** 更新了*第 7.3.1 节：通用 I/O (GPIO)*。
2012 年 10 月 19 日	2（续）	**DMA：** 更新了*第 9.2 节：DMA 主要特性*中的直接模式说明。 更新了*第 节：存储器到外设模式*和*第 9.3.12 节：FIFO/ 直接模式*中的直接模式说明。 修改了*表 35：DMA1 请求映射*中的流 2/通道 2。 在*第 9.5.5 节：DMA 数据流 x 配置寄存器 (DMA_SxCR) (x = 0..7)* 中增加了与 EN 位相关的注释。更新了*第 9.5.6 节：DMA 数据流 x 数据项数寄存器 (DMA_SxNDTR) (x = 0..7)* 中 NDT[15:0] 位的定义。 更新了*第 9.5 节：DMA 寄存器*中的寄存器访问。 中断： 将*第 10.1.1 节：NVIC 特性*中可屏蔽中断数更新为 82。 **EXTI：** 更新了*第 10.2 节：外部中断/事件控制器 (EXTI)*

表 240. 文档版本历史

日期	版本	变更
2012 年 10 月 19 日	2（续）	**ADC：** 在*第 11.5 节：可独立设置各通道采样时间*中将 ADCCLK 频率更改为 30 MHz。 在*第 11.8.1 节：使用 DMA* 和*第 11.8.2 节：在不使用 DMA 的情况下管理转换序列*中增加了从 ADC 序列恢复。 更新了*第 11.13.2 节：ADC 控制寄存器 1 (ADC_CR1)* 中的 AWDIE。在*第 11.13 节：ADC 寄存器*中增加了读写访问。 **高级控制定时器（TIM1 和 TIM8）** 更新了*第 14.2 节：TIM1 和 TIM8 主要特性*中的 16 位预分频器范围。 更新了*图 99. 捕获/比较通道的输出阶段（通道 1 到 3）*中的 OC1 框图。 更新了*第 14.3.2 节：计数器模式*和*第 14.3.3 节：重复计数器*中*递增计数模式*和*递减计数模式*中的更新事件生成。 更新了*第 14.3.11 节：互补输出和死区插入*中控制死区生成的位。 更新了*第 14.3.12 节：使用断路功能*中生成断路的方式。 在*第 14.3.13 节：发生外部事件时清除 OCxREF 信号*给出的示例中将 OCxREF 更改为 ETR 并 在*图 109：清除 TIMx 的 OCxREF* 中将 OCREF_CLR 更改为 ETRF。 更新了计数器操作示例的配置 （在*第 14.3.16 节：编码器接口模式*的编码器接口模式下。） 在*第 14.4 节：TIM1 和 TIM8 寄存器*中增加了寄存器访问。 更改了*第 14.4.12 节：TIM1 和 TIM8 自动重载寄存器 (TIMx_ARR)* 中 ARR[15:0] 位的定义。 更新了*第 14.4.18 节：TIM1 和 TIM8 断路和死区寄存器 (TIMx_BDTR)* 中的 BKE 定义。

表 240. 文档版本历史

日期	版本	变更
2012 年 10 月 19 日	2（续）	**通用定时器（TIM2 到 TIM5）** 删除了对“重复计数器”的所有引用。 增加了*图 119：通用定时器框图*。 更新了*第 15.2 节：TIM2 到 TIM5 主要特性*中的 16 位预分频器范围。 外部时钟模式 2 ETR 被限制为 TIM2 到 TIM4（在*第 15.3.3 节：时钟选择*和*第 15.3.6 节：PWM 输入模式*中）。 更新了*第 15.3.9 节：PWM 模式*和*第 15.3.11 节：发生外部事件时清除 OCxREF 信号*。 更新了*图 159：主/从定时器示例*，将 ITR1 更改为 ITR0。 更新了*第 15.4 节：TIM2 到 TIM5 寄存器*中对寄存器的读写访问。 恢复了 TIMx_SMCR 的 15 至 8 位以及*第 14.4.3 节*中的*表 76：TIMx 内部触发连接*。 删除了*第 15.4.13 节：TIMx 捕获/比较寄存器 1 (TIMx_CCR1)* 中与 OC1M 位相关的注释 1。 更新了对 TIM2 到 TIM5 的 TIMx_CCER 位的说明（在*第 15.4.9 节：TIMx 捕获/比较使能寄存器 (TIMx_CCER)*中）。 **通用定时器（TIM9 到 TIM14）** 更新了*第 16.2.1 节：TIM9/TIM12 主要特性*和*第 16.3 节：TIM10/TIM11 和 TIM13/TIM14 主要特性*中的 16 位预分频器范围。 更新了*图 166：通用定时器框图 (TIM10/11/13/14)*，以删除 TRGO 触发器控制器输出。 增加了寄存器访问（在 *第 16.5 节：TIM9 和 TIM12 寄存器*和*第 16.6 节：TIM10/11/13/14 寄存器*中）。 **基本定时器（TIM6 和 TIM7）** 删除了对“重复计数器”的所有引用。 更新了*第 17.2 节：TIM6 和 TIM7 的主要特性*中的 16 位预分频器范围。 **HASH：** 更新了*第 22.3.1 节：处理的持续时间*。 **RNG：** 更新了*第 21.1 节：RNG 简介*。

表 240. 文档版本历史

日期	版本	变更
2012 年 10 月 19 日	2（续）	**RTC：** 更新了*图 222：RTC 框图*。 在*第 23.3.1 节：时钟和预分频器*中增加了计算 fck_apre 的公式。 更新了*第 23.3.9 节：RTC 参考时钟检测*。 更新了*RTC 寄存器写保护*一节。 在*第 23.3.6 节：读取日历*中增加了 RTC_SSR 影子寄存器。 更新了*第 23.6.7 节：RTC 校准寄存器 (RTC_CALIBR)* 中 DC[4:0] 位的说明。 在*表 99：RTC 寄存器映射和复位值*中将 RTC_BKxR 重命名为 RTC_BKPxR。 增加了上电复位值并将复位值更改为系统复位值（在*第 23.6.11 节：RTC 亚秒寄存器 (RTC_SSR)* 中）。 更新了*第 23.6.17 节：RTC 入侵和复用功能配置寄存器 (RTC_TAFCR)* 中 ALARMOUTTYPE 的定义。 **I2C：** 修改了*第 25.3.8 节：DMA 请求*。 更新了*第 25.6.3 节：I^2C 自有地址寄存器 1 (I2C_OAR1)* 中位 14 的说明。 更新了 PE 的定义以及与 SWRST 位相关的注释； 将与 STOP 位相关的注释移到了整个寄存器（在*第 25.6.1 节：I^2C 控制寄存器 1 (I2C_CR1)*中）。 **USART：** *第 26.6.6 节：控制寄存器 3 (USART_CR3)*：删除了 DMAT 和 DMAR 说明中与 UART5 相关的注释。 更新了*表 116：采用 16 倍过采样时，在 fPCLK = 42 MHz 或 fPCLK = 84 Hz 下编程波特率时的误差计算*和*表 117：采用 8 倍过采样时，在 fPCLK = 42 MHz 或 fPCLK = 84 MHz 下编程波特率时的误差计算*。 **SPI/I2S：** 更新了*第 27.1 节：SPI 简介*。 将 I2S 单工通信/模式更改为半双工通信/模式。更新了*第 27.2.2 节：I^2S 特性*中接收/发送模式下的标志。在*表 126：I^2S 中断请求*中增加了帧错误标志。 在*第 27.5 节：SPI 和 I^2S 寄存器*中增加了寄存器访问。 更新了*第 27.5.2 节：SPI 控制寄存器 2 (SPI_CR2)* 中的 ERRIE 定义。 在*第 27.5.3 节：SPI 状态寄存器 (SPI_SR)* 中将 TIFRFE 重命名为 FRE 并更新了定义。

表 240. 文档版本历史

日期	版本	变更
2012 年 10 月 19 日	2（续）	**SDIO：** 更新了*表 154：R4 响应*中位 [45:40] 和 [7:1] 的值和说明。更新了*表 156：R5 响应*中位 [45:40] 的值。 **CAN：** 更新了*图 224：双 CAN 框图*。 修改了*CAN 筛选器主寄存器 (CAN_FMR)* 一节中 CAN2SB 位的定义。 增加了寄存器访问，位于 *第 24.9 节：CAN 寄存器* **ETHERNET：** 更新了精度网络化时钟同步的标准（在*第 29.1 节：以太网简介*和*第 29.2.1 节：MAC 内核特性*中）。 更新了*以太网 MAC MII 地址寄存器 (ETH_MACMIIAR)* 一节中 CR 位的定义。 在*表 167：源地址过滤*中用 PM 位替换 RTPR。 **USB OTG FS** 更新了远程唤醒信号位并在*挂起状态*一节中恢复了中断。 在*第 30.16 节：OTG_FS 控制和状态寄存器*中增加了外设寄存器访问。 更新了 OTG_FS_DIEPTXFx 中的 INEPTXSA 说明。 将 PHYSEL 从 OTG_FS_GUSBCFG 寄存器的位 7 更改为位 6。 **USB OTG HS** 更新了远程唤醒信号位并在*第 节：挂起状态*中恢复了中断。 在*第 31.12 节：OTG_HS 控制和状态寄存器*中增加了外设寄存器访问。 更新了 OTG_HS_DIEPTXFx 中的 INEPTXSA 说明。 在*OTG_HS 主机配置寄存器 (OTG_HS_HCFG)* 一节中更新了 LS 主机模式的 FSLSPCS 并增加了 PHYSEL。 将 PHYSEL 重命名为 PHSEL 并将其从 OTG_HS_GUSBCFG 寄存器的第 7 位更改为第 6 位。 更新了 OTG_HS_DIEPEACHMSK1 和 OTG_HS_DOEPEACHMSK1 复位值。

表 240. 文档版本历史

日期	版本	变更
2012 年 10 月 19 日	2（续）	**FSMC：** 更新了*第 32.3.1 节：支持的存储器和事务*中的步骤 b)。 更新了*表 196：FSMC_BTRx 位字段*和*表 204：FSMC_BTRx 位字段*。 在*表 215：可编程的 NAND/PC 卡访问参数*中更改了时钟分频比最小值。 更新了*第 32.5 节：NOR Flash/PSRAM 控制器*中同步访问的情况。 在*表 203*、*表 206*、*表 207*、*表 209* 和*表 210* 中将 ADDSET 的最小值更改为 0。 将注释从*图 406：模式 1 写入访问*移动到*图 405：模式 1 读取访问*。 将注释从*图 408：模式 A 写入访问*移动到*图 407：模式 A 读取访问*。 更新了*异步访问中的 WAIT 管理*一节。 在*第 32.5.6 节：NOR/PSRAM 控制寄存器*和*第 32.6.2 节：NAND Flash/PC 卡支持的存储器和事务*中增加了寄存器访问。 在*第 32.6.1 节：外部存储器接口信号*中删除了警告注释。 更新了*表 218：16 位 PC 卡*。 更新了*第 32.6.4 节：NAND Flash 操作*中的步骤 3。 更新了*图 424：访问非“CE 无关”NAND-Flash* 和*第 32.6.5 节：NAND Flash 预等待功能*下方的注释。 更新了*第 32.6.7 节：PC 卡/CF 卡操作*中对 I/O 空间的访问。更新了*表 220：16 位 PC 卡信号和访问类型*。更新了*第 节：SRAM/NOR-Flash 片选时序寄存器 1..4 (FSMC_BTR1..4)* 中的 BUSTURN 位定义。在*SRAM/NOR-Flash 写入时序寄存器 1..4 (FSMC_BWTR1..4)* 一节中将位 16 至 19 更改为 BUSTURN。 **DEBUG：** 更新了*第 33.4.3 节：JTAG 引脚上的内部上拉和下拉*。 **电子签名** 更新了*第 34 节：设备电子签名*简介。 在*第 33.6.1 节：MCU 器件 ID 代码*中更新了 REV_ID[15:0] 并增加了版本 Z。 更新了*第 34.2 节：Flash 大小*中的地址和示例。

表 240. 文档版本历史

日期	版本	变更
2012 年 11 月 13 日	3	增加了 STM32F42x 和 STM32F43x 器件。 在封面页删除了参考 Flash 编程手册。增加了*第 2.3.2 节：Flash 概述*和*第 3 节：嵌入式 Flash 接口*。 在*第 7.3.2 节：I/O 引脚复用器和映射*中将 RTC_50Hz 更改为 RTC_REFIN。在*第 7 节：通用 I/O (GPIO)* 和*第 23 节：实时时钟 (RTC)* 中修改了 RTC 备用功能命名。 更新了*第 23.3.1 节：时钟和预分频器*中的最大输入频率。 在*第 9.5.3 节：DMA 低中断标志清零寄存器 (DMA_LIFCR)* 和*第 9.5.4 节：DMA 高中断标志清零寄存器 (DMA_HIFCR)* 中将位访问类型从 'rw' 更改为 'w' 并更新了位说明。 更新了*图 15：TIM5 在输入捕获模式下的频率测量*。 更新了*信号同步*一节中的*第 32 节：灵活的静态存储控制器 (FSMC)* *第 30 节：全速 USB on-the-go (OTG_FS)*：更新了*图 360：USB 仅作主机的连接*、V_{BUS} *有效*一节和*主机检测设备连接*一节。 *第 31 节：高速 USB on-the-go (OTG_HS)*：更新了 V_{BUS} *有效*一节和*主机检测到设备连接*一节。
2013 年 2 月 19 日	4	更新了*第 2.3.1 节：嵌入式 SRAM*。 更新了*图 1：STM32F405xx/07xx 和 STM32F415xx/17xx 器件的系统架构*和*图 2：STM32F42xxx 和 STM32F43xxx 器件的系统架构*。 更新了*表 4：存储器映射与自举模式/物理重映射*。更新了*图 4：32 位连续指令的执行*，并删除了*表 7：CPU 时钟 (HCLK) 频率对应的等待周期数*和*表 8：编程/擦除并行位数*中的注释 1。 **PWR：** 更新了*图 7：电源概述*。 更新了*第 5.1.3 节：调压器*。 在*第 5.4.2 节：用于STM32F42xxx 和 STM32F43xxx 的 PWR 电源控制寄存器 (PWR_CR)* 中增加了 ADCDC1 位。 **SYSCFG：** 在*第 8.2.3 节：用于 STM32F42xxx 和 STM32F43xxx 的 SYSCFG 外设模式配置寄存器 (SYSCFG_PMC)* 中增加了 ADCxDC2 位。 **ADC：** 更新了*第 11.9.3 节：交替模式*、*第 11.9.4 节：交替触发模式*和*第 11.9.6 节：规则同时 + 交替触发组合模式*以说明中断的转换情况。 更新了*温度传感器*、V_{REFINT} *和* V_{BAT} *内部通道*一节、*第 11.10 节：温度传感器*和*第 11.11 节：电池充电监视*。 **RTC：** 更新了*第 23.6.20 节：RTC 备份寄存器 (RTC_BKPxR)* 中的 BKP[31:0] 位说明。 **I2C：** 更新了*第 25.3.5 节：可编程噪声滤波器*。

表 240. 文档版本历史

日期	版本	变更
2013 年 2 月 19 日	4（续）	**FSMC：** 更新了*第 32.1 节：FSMC 主要特性*中写 FIFO 的大小。 更新了*图 403：FSMC 框图*。 更新了*第 32.5.4 节：NOR Flash/PSRAM 控制器异步事务*。 修改了*模式 2/B - NOR Flash* 一节中模式 B 与模式 1 之间的差别。 修改了*模式 C - NOR Flash - OE 切换*一节中模式 C 与模式 1 之间的差别。 修改了*模式 D - 扩展地址异步访问*一节中模式 D 与模式 1 之间的差别。 更新了*图 418：读取访问期间的异步等待*、*图 419：写入访问期间的异步等待*、*图 420：等待配置*、*图 421：同步复用读取模式 - NOR、PSRAM (CRAM)* 和*图 422：同步复用写入模式 - PSRAM (CRAM)* 中的 NWAIT 信号。 更新了*表 195* 至*表 214*。 更新了 *SRAM/NOR-Flash 片选控制寄存器 1..4 (FSMC_BCR1..4)* 一节。 **DEBUG** 更新了*图 425：STM32 MCU 和 Cortex™-M4F 级调试支持框图*。

请仔细阅读：

中文翻译仅为方便阅读之目的。该翻译也许不是对本文档最新版本的翻译，如有任何不同，以最新版本的英文原版文档为准。

本文档中信息的提供仅与 ST 产品有关。意法半导体公司及其子公司 （“ST”）保留随时对本文档及本文所述产品与服务进行变更、更正、修改或改进的权利，恕不另行通知。

所有 ST 产品均根据 ST 的销售条款出售。

买方自行负责对本文所述 ST 产品和服务的选择和使用， ST 概不承担与选择或使用本文所述 ST 产品和服务相关的任何责任。

无论之前是否有过任何形式的表示，本文档不以任何方式对任何知识产权进行任何明示或默示的授权或许可。如果本文档任何部分涉及任何第三方产品或服务，不应被视为 ST 授权使用此类第三方产品或服务，或许可其中的任何知识产权，或者被视为涉及以任何方式使用任何此类第三方产品或服务或其中任何知识产权的保证。

除非在 ST 的销售条款中另有说明，否则，ST 对 ST 产品的使用和 / 或销售不做任何明示或默示的保证，包括但不限于有关适销性、适合特定用途（及其依据任何司法管辖区的法律的对应情况），或侵犯任何专利、版权或其他知识产权的默示保证。

意法半导体的产品不得应用于武器。此外，意法半导体产品也不是为下列用途而设计并不得应用于下列用途：（A）对安全性有特别要求的应用，例如，生命支持、主动植入设备或对产品功能安全有要求的系统；（B）航空应用；（C）汽车应用或汽车环境，且 / 或（D）航天应用或航天环境。如果意法半导体产品不是为前述应用设计的，而采购商擅自将其用于前述应用，即使采购商向意法半导体发出了书面通知，采购商仍将独自承担因此而导致的任何风险，意法半导体的产品设计规格明确指定的汽车、汽车安全或医疗工业领域专用产品除外。根据相关政府主管部门的规定，ESCC、QML 或 JAN 正式认证产品适用于航天应用。

经销的 ST 产品如有不同于本文档中提出的声明和 / 或技术特点的规定，将立即导致 ST 针对本文所述 ST 产品或服务授予的任何保证失效，并且不应以任何形式造成或扩大 ST 的任何责任。

ST 和 ST 徽标是 ST 在各个国家或地区的商标或注册商标。

本文档中的信息取代之前提供的所有信息。

ST 徽标是意法半导体公司的注册商标。其他所有名称是其各自所有者的财产。

意法半导体集团公司

澳大利亚 - 比利时 - 巴西 - 加拿大 - 中国 - 捷克共和国 - 芬兰 - 法国 - 德国 - 中国香港 - 印度 - 以色列 - 意大利 - 日本 - 马来西亚 - 马耳他 - 摩洛哥 – 菲律宾 – 新加坡 - 西班牙 - 瑞典 - 瑞士 - 英国 - 美国

www.st.com