参考手册

STM32F101xx 和 STM32F103xx
ARM 内核 32 位高性能微控制器

导言

本参考手册针对应用开发，提供关于如何使用 STM32F101xx 和 STM32F103xx 微控制器的存储器和外设的详细信息。在本参考手册中 STM32F101xx 和 STM32F103xx 被统称为 STM32F10xxx。

STM32F10xxx 系列拥有不同的存储器容量，封装和外设配置。

关于订货编号，电器和物理性能参数，请参考 STM32F101xx 和 STM32F103xx 数据手册。

关于芯片内部闪存的编程，擦除和保护操作，请参考 STM32F10xxx 闪存编程手册。

关于 ARM Cortex™-M3 内核的具体信息，请参考 Cortex™-M3 术参考手册。

* 感谢南京万利提供原始翻译文档

目录

1 文中的缩写

1.1 寄存器描述表中使用的缩写列表

在对寄存器的描述中使用了下列缩写：

read / write (rw)	软件能读写此位。
Read-only (r)	软件只能读此位。
write-only (w)	软件只能写此位，读此位将返回复位值。
read-clear (rc)	软件只能读或者清除此位。
read/clear (rc_w1)	软件可以读此位，也可以通过写 1 清除此位，写 0 对此位无影响。
read / clear (rc_w0)	软件可以读此位，也可以通过写 0 清除此位，写 1 对此位无影响。
read / set (rs)	软件可以读此位，也可以设置此位为 1， 写 0 对此位无影响。
Toggle (t)	软件只能通过写 1 来翻转此位，写 0 对此位无影响。

2 存储器和总线构架

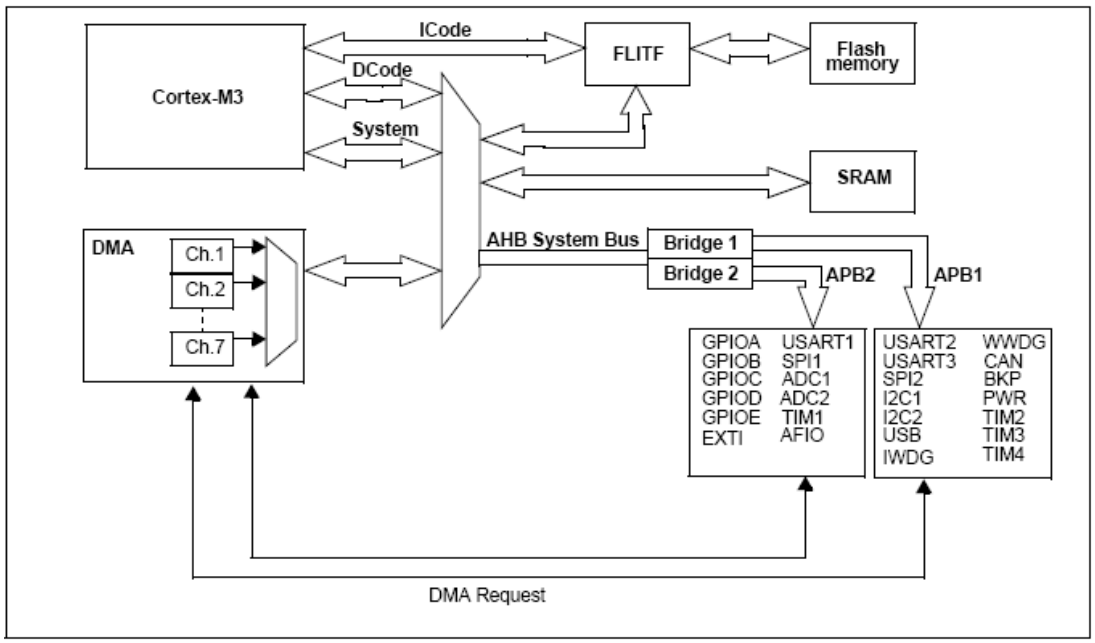
2.1 系统构架

主系统由以下部分构成：

- 四个驱动单元：
  - Cortex™-M3 内核 ICode 总线(I-bus)，DCode 总线(D-bus)，和系统总线(S-bus)
  - GP-DMA(通用 DMA)
- 三个被动单元
  - 内部 SRAM
  - 内部闪存存储器
  - AHB 到 APB 的桥(AHB2APBx)，它连接所有的 APB 设备

这些都是通过一个多级的AHB总线构架相互连接的，如 图 1 所示：

图1 系统结构



ICode总线

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

DCode总线

该总线将 Cortex™-M3 内核的 DCode 总线与闪存存储器的数据接口相连接(常量加载和调试访问)。

系统总线

此总线连接 Cortex™-M3 内核的系统总线(外设总线)到总线矩阵，总线矩阵协调着内核和 DMA 间的访问。

DMA总线

此总线将 DMA 的 AHB 主控接口与总线矩阵相联，总线矩阵协调着 CPU 的 DCode 和 DMA 到 SRAM、闪存和外设的访问。

总线矩阵

此总线矩阵协调内核系统总线和 DMA 主控总线之间的访问仲裁。此仲裁利用轮换算法。此总线矩阵由三个驱动部件(CPU 的 DCode、系统总线和 DMA 总线)和三个被动部件(闪存存储器接口、SRAM 和 AHB2APB 桥)构成。
AHB 外设通过总线矩阵与系统总线相连，允许 DMA 访问。

AHB/APB桥(APB)

两个 AHB/APB 桥在 AHB 和 2 个 APB 总线间提供同步连接。APB1 操作速度限于 36MHz，APB2 操作于全速(最高 72MHz)。
参考第 1 章有关连接到每个桥的不同外设的地址映射。

2.2 存储器组织

程序存储器、数据存储器、寄存器和输入输出端口被组织在同一个 4GB 的线性地址空间内。

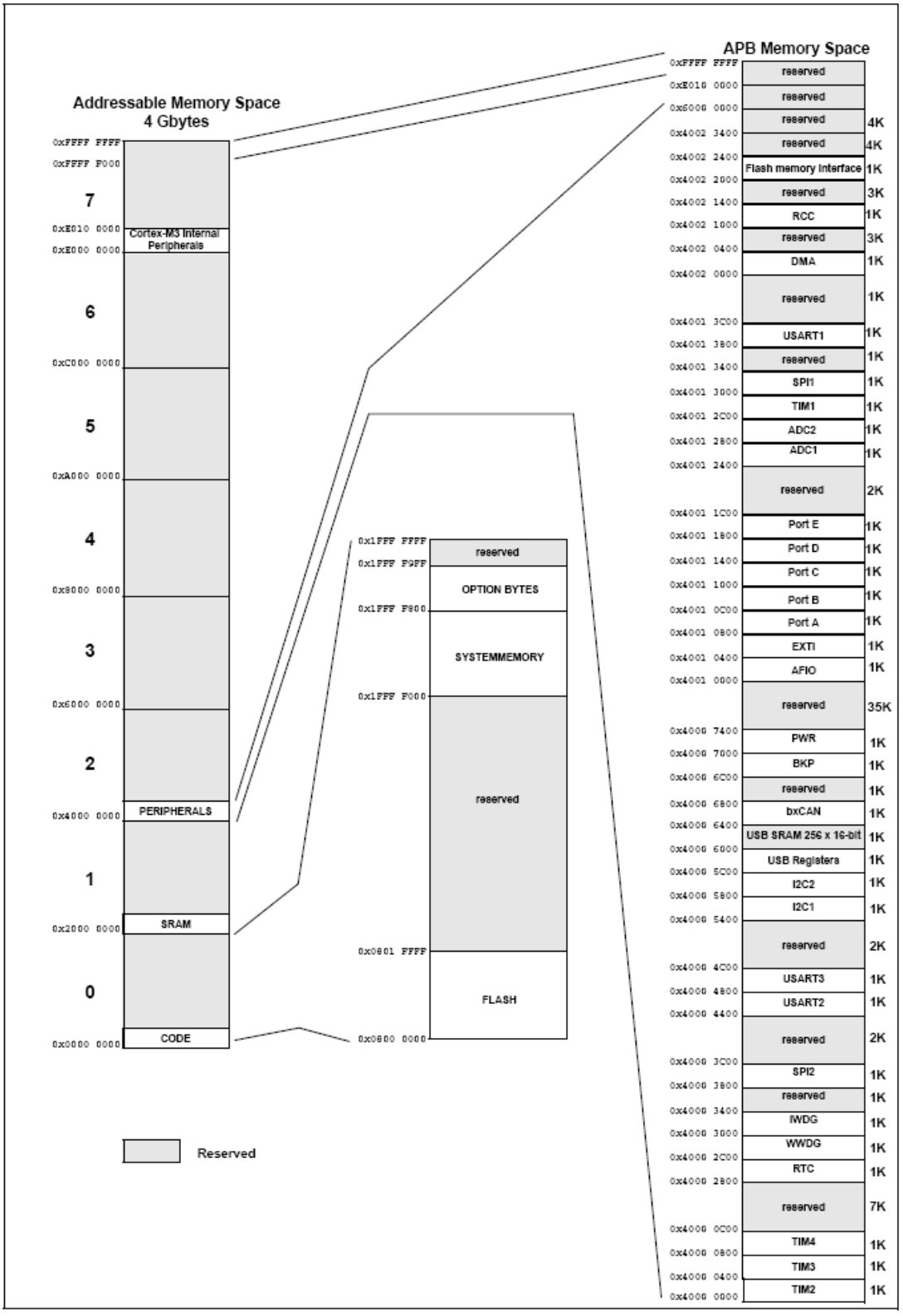
数据字节以小端格式存放在存储器中。一个字里的最低地址字节被认为是该字的最低有效字节，而最高地址字节是最高有效字节。

图 2 展示了STM32F10xxx的存储器映像。外设寄存器的映像请参考相关章节。

可访问的存储器空间被分成 8 个主要块，每个块为 512MB。
其他所有没有分配给片上存储器和外设的存储器空间都是保留的地址空间(图 2 中的阴影部分)。

2.3 存储器映像

图2 存储器映像



2.3.1 外设存储器映像

表1 寄存器组起始地址

起始地址	外设	总线	寄存器映像
0x4002 2400 - 0x4002 3FFF	保留	AHB	
0x4002 2000 - 0x4002 23FF	闪存存储器接口		
0x4002 1400 - 0x4002 1FFF	保留		
0x4002 1000 - 0x4002 13FF	复位和时钟控制		
0x4002 0400 - 0x4002 0FFF	保留		
0x4002 0000 - 0x4002 03FF	DMA		
0x4001 3C00 - 0x4001 3FFF	保留	APB1	
0x4001 3800 - 0x4001 3BFF	USART1		
0x4001 3400 - 0x4001 37FF	保留		
0x4001 3000 - 0x4001 33FF	SPI1		
0x4001 2C00 - 0x4001 2FFF	TIM1时钟		
0x4001 2800 - 0x4001 2BFF	ADC2		
0x4001 2400 - 0x4001 27FF	ADC1		
0x4001 2000 - 0x4001 1FFF	保留		
0x4001 1800 - 0x4001 1BFF	GPIO端口E		
0x4001 1400 - 0x4001 17FF	GPIO端口D		
0x4001 1000 - 0x4001 13FF	GPIO端口C		
0X4001 0C00 - 0x4001 0FFF	GPIO端口B		
0x4001 0800 - 0x4001 0BFF	GPIO端口A		
0x4001 0400 - 0x4001 07FF	EXTI		
0x4001 0000 - 0x4001 03FF	AFIO		
0x4000 8000 - 0x4000 77FF	保留		
0x4000 7000 - 0x4000	电源控制		

73FF			
0x4000 6C00 - 0x4000 6FFF	后备寄存器(BKP)		
0x4000 6800 - 0x4000 6BFF	保留		
0x4000 6400 - 0x4000 67FF	bxCAN		
0x4000 6000 - 0x4000 63FF	USB/CAN 共享的 SRAM 512字节		
0x4000 5C00 - 0x4000 5FFF	USB寄存器		
0x4000 5800 - 0x4000 5BFF	I2C2		
0x4000 5400 - 0x4000 57FF	I2C1		
0x4000 5000 - 0x4000 4FFF	保留		
0x4000 4800 - 0x4000 4BFF	USART3		
0x4000 4400 - 0x4000 47FF	USART2		
0x4000 4000 - 0x4000 3FFF	保留		
0x4000 3800 - 0x4000 3BFF	SPI2		
0x4000 3400 - 0x4000 37FF	保留	APB1	
0x4000 3000 - 0x4000 33FF	独立看门狗(IWDG)		
0x4000 2C00 - 0x4000 2FFF	窗口看门狗(WWDG)		
0x4000 2800 - 0x4000 2BFF	RTC		
0x4000 2400 - 0x4000 0FFF	保留		
0x4000 0800 - 0x4000 0BFF	TIM4定时器		
0x4000 0400 - 0x4000 07FF	TIM3定时器		
0x4000 0000 - 0x4000 03FF	TIM2定时器		

2.3.2 嵌入式SRAM

STM32F10xxx 内置 20K 字节的静态 SRAM。它可以以字节、半字(16 位)或全字(32 位)访问。SRAM 的起始地址是 0x2000 0000。

2.3.3 位段

Cortex™-M3 存储器映像包括两个位段(bit-band)区。这两个位段区将别名存储器区中的每个字映射到位段存储器区的一个位，在别名存储区写入一个字具有对位段区的目标位执行读-改-写操作的相同效果。

在 STM32F10xxx 里，外设寄存器和 SRAM 都被映射到一个位段区里，这允许执行单一的位段的写和读操作。

下面的映射公式给出了别名区中的每个字是如何对应位带区的相应位的：
bit_word_addr = bit_band_base + (byte_offset x 32) + (bit_number × 4)
其中：
bit_word_addr 是别名存储器区中字的地址，它映射到某个目标位。
bit_band_base 是别名区的起始地址。
byte_offset 是包含目标位的字节在位段里的序号
bit_number 是目标位所在位置（0-31）

例子：
下面的例子说明如何映射别名区中 SRAM 地址为 0x20000300 的字节中的位 2：
0x22006008 = 0x22000000 + (0x300*32) + (2*4).
对 0x22006008 地址的写操作和对 SRAM 中地址 0x20000300 字节的位 2 执行读-改-写操作有着相同的效果。

读 0x22006008 地址返回 SRAM 中地址 0x20000300 字节的位 2 的值(0x01 or 0x00)。

请参考《Cortex™-M3 技术参考手册》以了解更多有关位段的信息。

2.3.4 嵌入式闪存

高性能的闪存模块有以下的主要特性：

- 128K 字节闪存
- 存储器结构：闪存存储器有主存储块和信息块组成：
  - 主存储块为 16Kx64 位，每个主存储块又划分为 128 个 1K 字节的页。
  - 信息块为 258x64 位，每个信息块又划分为一个 2K 字节的页和一个 16 字节的页

闪存存储器接口的特性为：

- 带预取缓冲器的读接口(每字为 2x64 位)
- 选择字节加载器

- 闪存编程/擦除操作
- 访问/写保护

表2 闪存模块的组织

模块	名称	地址	大小(字节)
主存储块	页0	0x0800 0000 - 0x0800 03FF	1K
	页1	0x0800 0400 - 0x0800 07FF	1K
	页2	0x0800 0800 - 0x0800 0BFF	1K
	页3	0x0800 0C00 - 0x0800 0FFF	1K
	页4	0x0800 1000 - 0x0800 13FF	1K
	… …	… …	… …
	页127	0x0801 FC00 - 0x0801 FFFF	1K
信息块	系统存储器	0x1FFF F000 - 0x1FFF F7FF	2K
	用户选择字节	0x1FFF F800 - 0x1FFF F80F	16
闪存存储器寄存器	FLASH_ACR	0x4002 2000 - 0x4002 2003	4
	FALSH_KEYR	0x4002 2004 - 0x4002 2007	4
	FLASH_OPTKEYR	0x4002 2008 - 0x4002 200B	4
	FLASH_SR	0x4002 200C - 0x4002 200F	4
	FLASH_CR	0x4002 2010 - 0x4002 2013	4
	FLASH_AR	0x4002 2014 - 0x4002 2017	4
	保留	0x4002 2018 - 0x4002 201B	4
	FLASH_OBR	0x4002 201C - 0x4002 201F	4
	FLASH_WRPR	0x4002 2020 - 0x4002 2023	4
	保留	0x4002 2024 - 0x4002 2087	100

注： 有关闪存寄存器的详细信息，请参考《STM32F10xxx 闪存编程手册》

闪存读取

闪存的指令和数据访问是通过 AHB 总线完成的。预取模块是用于通过 ICode 总线读取指令的。仲裁是作用在闪存接口，并且 DCode 总线上的数据访问优先。读访问可以有以下配置选项：

- 等待时间：可以随时更改的用于读取操作的等待状态的数量。
- 预取：可以随时被激活/被禁止，以优化 CPU 的执行。
- 半周期：用于功耗优化。

注： 1 这些选项应与闪存存储器的访问时间一起使用。等待周期体现了系统时钟 (SYSCLK)频率与闪存访问时间的关系：

0 等待周期，当 0 < SYSCLK < 24MHz

1 等待周期，当 24MHz < SYSCLK ≤ 48MHz

1 等待周期，当 48MHz < SYSCLK ≤ 72MHz

2 半周期配置不能与使用了预分频器的 AHB 一起使用，时钟系统应该等于 HCLK 时钟。该特性只能用在直接使用 8MHz 的内部 RC 振荡器(HSI)或主振荡器 (HSE)时。

3 当 AHB 预分频系数不为 1 时，必须置预取缓冲区处于开启状态。

4 使用 DMA：DMA 在 DCode 总线上访问闪存存储器，它的优先级比 ICode 上的取指高。DMA 在每次传送完成后具有一个空余的周期。有些指令可以和 DMA 传输一起执行。

编程和擦除闪存

闪存编程一次可以写入 16 位(半字)。
闪存擦除操作可以按页面擦除或完全擦除(全擦除)。全擦除不影响信息块。
为了确保不发生过度编程， 闪存编程和擦除控制器块是由一个固定的时钟控制的。
写操作(编程或擦除)结束时可以触发中断。仅当闪存控制器接口时钟开启时，此中断可以用来从 WFI 模式退出。

注： 有关闪存存储器的操作和寄存器配置，请参考 STM32F10xxx 闪存编程手册。

2.4 启动配置

在 STM32F10xxx 里，可以通过 BOOT[1:0]引脚选择三种不同启动模式。

表3 启动模式

启动模式选择管脚		启动模式	说明
BOOT1	**BOOT0**		
X	0	用户闪存存储器	用户闪存存储器被选为启动区域
0	1	系统存储器	系统存储器被选为启动区域
1	1	内嵌SRAM	内嵌SRAM被选为启动区域

通过设置选择管脚，对应到各种启动模式的不同物理地址将被映像到第 0 块(启动存储区)。在系统复位后，SYSCLK 的第 4 个上升沿，BOOT 管脚的值将被锁存。用户可以通过设置 BOOT1 和 BOOT0 引脚的状态，来选择在复位后的启动模式。

即使被映像到启动存储区，仍然可以在它原先的存储器空间内访问相关的存储器。

在经过启动延迟后，CPU 从位于 0x0000 0000 开始的启动存储区执行代码。

内嵌的自举程序

内嵌的自举程序用于通过 USART1 串行接口对闪存存储器进行重新编程。这个程序位于系统存储器中，由 ST 在生产线上写入。

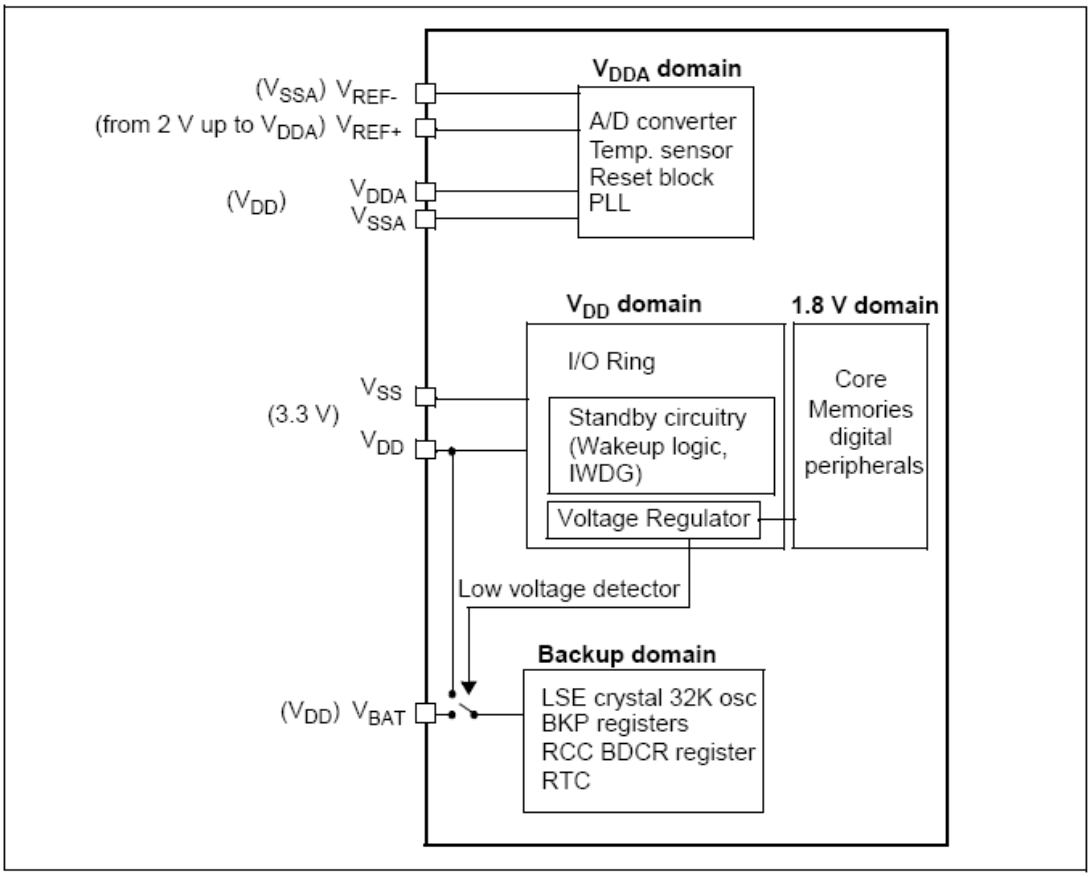
3 电源控制(PWR)

3.1 电源

STM32 的工作电压(V_{DD})为 2.0～3.6V。通过内置的电压调节器提供所需的 1.8V 电源。

当主电源 V_{DD} 掉电后，通过 V_{BAT} 脚为实时时钟(RTC)和备份寄存器提供电源。

图3 电源框图



注： V_{DDA} 和 V_{SSA} 必须分别联到 V_{DD} 和 V_{SS}。

3.1.1 独立的A/D转换器供电和参考电压

为了提高转换的精确度，ADC 使用一个独立的电源供电，过滤和屏蔽来自印刷电路板上的毛刺干扰。

- ADC 的电源引脚为 V_{DDA}
- 独立的电源地 V_{SSA}

如果有 V_{REF-}引脚（根据封装而定），它必须连接到 V_{SSA}。

100-脚封装

为了确保输入为低压时获得更好精度，用户可以连接一个独立的外部参考电压 ADC 到 V_{REF+}和 V_{REF-}脚上。在 V_{REF+}的电压范围为 2.4V～V_{DDA}。

64 脚或更少封装

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

3.1.2 电池备份区域

使用电池或其他电源连接到 V_{BAT}脚上，当 V_{DD}断电时，可以保存备份寄存器的内容和维持 RTC 的功能。
V_{BAT}脚也为 RTC、LSE 振荡器和 PC13 至 PC15 供电，这保证当主要电源被切断时 RTC 能继续工作。切换到 V_{BAT}供电由复位模块中的掉电复位功能控制。
如果应用中没有使用外部电池，V_{BAT}必须连接到 V_{DD}引脚上。

注意：
在 V_{DD}上升阶段($t_{RSTTEMPO}$)，V_{BAT}和 V_{DD}之间的电源开关仍会保持连接在 V_{BAT}。当 V_{DD}快速上升且达到稳定状态，当 V_{BAT}低于 V_{DD}−0.6V 时，电流通过 V_{DD}和 VBAT 之间的二极管流入到 V_{BAT}。关于 $t_{RSTTEMPO}$可参考数据手册中的相关部分。

如果在应用中没有外部电池，V_{BAT}必须在芯片外连到 V_{DD}。
当备份区域由 V_{DD}(内部模拟开关连到 V_{DD})供电时，下述功能可用：

- PC14 和 PC15 可以用于 GPIO 或 LSE 引脚
- PC13 可以作为通用 I/O 口、TAMPER 引脚、RTC 校准时钟、RTC 闹钟或秒输出(参见后备寄存器(BKP)部分)

注： 因为模拟开关只能通过少量的电流，使用 PC13 至 PC15 的 I/O 口功能是有限制的：同一时间内只有一个 I/O 口可以作为输出，速度必须限制在 2MHz 以下，最大负载为 30pF，而且这些 I/O 口绝不能当作电流源(如驱动 LED)。

当后备区域由 VBAT(VDD 消失后模拟开关连到 VBAT)，可以使用下述功能：

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

3.1.3 电压调节器

复位后调节器总是使能的。根据应用方式它以 3 种不同的模式工作。

- 运转模式：调节器以正常功耗模式提供 1.8V 电源（内核，内存和外设）。

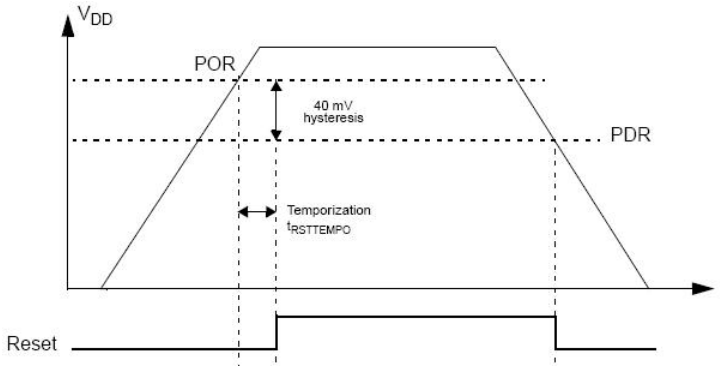
- 停止模式：调节器以低功耗模式提供 1.8V 电源，以保存寄存器和 SRAM 的内容。
- 待机模式：调节器停止供电。除了备用电路和备份领域以外，寄存器和 SRAM 的内容全部丢失。

3.2 电源管理器

3.2.1 上电复位(POR)和掉电复位(PDR)

STM32 内部有一个完整的上电复位(POR)和掉电复位(PDR)电路，当供电电压达到 2V 时系统既能正常工作。
当 V_{DD} 低于指定的限位电压 V_{POR}/V_{PDR} 时，系统保持为复位状态，而无需外部复位电路。更多关于上电复位和掉电复位的细节请参考数据手册的电气特性部分。

图4 上电复位和掉电复位的波形图



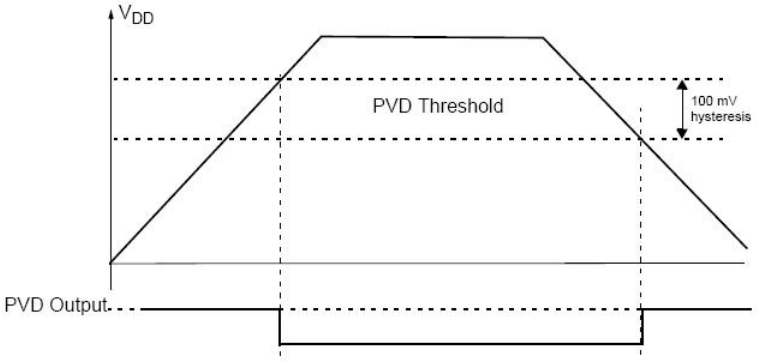
3.2.2 可编程电压监测器(PVD)

用户可以利用 PVD 对 V_{DD} 电压与电源控制寄存器(PWR_CR)中的 PLS[2:0]位进行比较来监控电源，这几位选择监控电压的阀值。

通过设置 PVDE 位来使能 PVD。

电源控制/状态寄存器(PWR_CSR)中的 PVDO 标志用来表明 V_{DD} 是高于还是低于 PVD 的电压阀值。该事件在内部连接到外部中断的第 16 线，如果该中断在外部中断寄存器中是使能的，该事件就会产生中断。当 V_{DD} 下降到 PVD 阀值以下和（或）当 V_{DD} 上升到 PVD 阀值之上时，根据外部中断第 16 线的上升/下降边沿触发设置，就会产生 PVD 中断。例如，这一特性可用于用于执行紧急关闭任务。

图5 PVD 的门限



3.3 低功耗模式

在系统或电源复位以后，微控制器处于运行状态。运行状态下的 HCLK 为 CPU 提供时钟，内核执行程序代码。当 CPU 不需继续运行时，可以利用多个低功耗模式来节省功耗，例如等待某个外部事件时。根据最低电源消耗，最快速启动时间和可用的唤醒源的需求，选取一个最佳的折中方案来帮助用户选定一个低功耗模式。

STM32F10xxx 有三中低功耗模式：

- 睡眠模式（Cortex™-M3 内核停止，外设仍在运行）
- 停止模式（所有的时钟都以停止）
- 待机模式（1.8V 电源关闭）

此外，在运行模式下，可以通过以下方式中的一种降低功耗：

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

表4 低功耗模式一览

模式	进入操作	唤醒	对1.8V区域时钟的影响	对VDD区域时钟的影响	电压调节器
睡眠(SLEEP-NOW或SLEEP-ON-EXIT)	WFI	任一中断	CPU时钟关，对其他时钟和ADC时钟无影响	无	开
	WFE	唤醒事件			
停机	PDDS 和 LPDS 位+SLEEPDEEP 位+WFI或WFE	任一外部中断(在外部中断寄存器中设置)	所有使用1.8V的区域的时钟都已关闭，HSI 和HSE的振荡器关闭		在低功耗模式下可进行开/关设置(依据电源控制寄存器(PWR_CR)的设定)
待机	PDDS 位+SLEEPDEEP 位+WFI或WFE	WKUP 引脚的上升沿、RTC警告事件、NRST引脚上的外部复位、IWDG复位			关

3.3.1 降低系统时钟

在运行模式下，通过对预分频器的寄存器进行编程，可以降低任意一个系统时钟(SYSCLK、HCLK、PCLK1、PCLK2)的速度。在进入睡眠模式前，也可以利用预分频器来降低外设的时钟。

更多细节参考 4.3.2 时钟配置寄存器（RCC_CFGR）

3.3.2 外部时钟的控制

在运行模式下，任何时候都可以停止为外设和内存提供时钟(HCLK 和 PCLKx)来减少功耗。

为了在睡眠模式下减少更多的功耗，可在执行 WFI 或 WFE 指令前关闭所有外设的时钟。

通过设置 AHB 外设时钟使能寄存器(RCC_AHBENR)、APB1 外设的时钟使能寄存器(RCC_APB1ENR)和 APB1 外设的时钟使能寄存器(RCC_APB2ENR)来开关外设部时钟。

3.3.3 睡眠模式

进入睡眠模式

通过执行 WFI 或 WFE 指令进入睡眠状态。根据 Cortex™-M3 系统控制寄存器中的 SLEEPONEXIT 位的值，有两种选项可用于选择睡眠模式进入机制：

- SLEEP-NOW：如果 SLEEPONEXIT 位被清除，当 WRI 或 WFE 被执行时，微控制器立即进入睡眠模式。
- SLEEP-ON-EXIT：如果 SLEEPONEXIT 位被置位，当执行 WFI 时，系统从最低优先级的中断处理程序中退出时，微控制器就立即进入睡眠模式。

关于如何进入睡眠模式，更多的细节参考 表 5 和 表 6。

退出睡眠模式

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

如果执行 WFE 指令进入睡眠模式，当下述任一事件发生时，微处理器都将从睡眠模式退出；外设控制寄存器中被使能的中断事件，但该中断事件未被嵌套向量中断控制器使能，或者设置为事件模式的外部中断线上的事件。

该模式唤醒所需的时间最短，因为没有时间损失在中断的进入或退出上。

关于如何退出睡眠模式，更多的细节参考 表 5 和 表 6。

表5 SLEEP-NOW 模式

SLEEP-NOW模式	说明
进入	在以下条件下执行WFI或WFE指令： – SLEEPDEEP = 0 和 – SLEEPONEXIT = 0 参考Cortex-M3系统控制寄存器。
退出	如果执行WFI进入睡眠模式： 中断：参考中断向量表 如果执行WFE进入睡眠模式： 唤醒事件：参考唤醒事件管理
唤醒延时	无

表6 SLEEP-ON-EXIT 模式

SLEEP-ON_EXIT模式	说明
进入	在以下条件下执行WFI或WFE指令： – SLEEPDEEP = 0和 – SLEEPONEXIT = 1 参考Cortex™-M3系统控制寄存器
退出	如果执行WFI进入睡眠模式： 中断或清除Cortex-M3控制寄存器位1 如果执行WFE进入睡眠模式： 唤醒事件：参考唤醒事件管理
唤醒延时	无

3.3.4 停止模式

停止模式是在 Cortex™-M3 的深睡眠模式基础上结合了外设的时钟控制机制，在停止模式下电压调节器可运行在正常或低功耗模式。此时在 1.8V 供电区域的的所有时钟都被停止，PLL、HIS 和 HSE RC 振荡器的功能被禁止，SRAM 和寄存器内容被保留下来。

进入停止模式

关于如何进入停止模式，详见 表 7。

在停止模式下，通过设置电源控制寄存器(PWR_CR)的 LPDS 位使内部调节器进入低功耗模式，能够降低更多的功耗。

如果正在进行 Flash 编程，直到对内存访问完成，系统才进入停止模式。
如果正在进行对 APB 的访问，直到对 APB 访问完成，系统才进入停止模式。

可以通过对独立的控制位进行编程，可选择以下功能：

- 独立看门狗(IWDG)：可通过看门狗的 Key 寄存器的写操作或硬件选择来启动 IWDG。一旦启动除了系统复位，看门狗就不能被停止。更多细节参见独立看门狗章节。
- 实时时钟(RTC)：通过备用区域控制寄存器(RCC_BDCR)的 RTCEN 位来设置。
- 内部 RC 振荡器(LSI RC)：通过控制/状态寄存器(RCC_CSR)的 LSION 位来设置。
- 外部 32.768kHz 振荡器(LSE OSC)：通过备用区域控制寄存器(RCC_BDCR)的 LSEON 位设置。

退出停止模式

关于如何退出停止模式，详见 表 7。

当一个中断或唤醒事件导致退出停止模式时，HSI RC 振荡器被选为系统时钟。

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

表7 停止模式

停止模式	说明
进入	在以下条件下执行WFI或WFE指令： – 设置Cortex-M3系统控制寄存器中的SLEEPDEEP位 – 清除电源控制寄存器（PWR_CR）中的PDDS位 – 通过设置PWR_CR中LPDS位选择电压调节器的模式 注：为了进入停止模式，所有的外部中断的请求位(挂起寄存器(EXTI_PR))和RTC的闹钟标志都必须被清除，否则停止模式的进入流程将会被跳过，程序继续运行。
退出	在以下条件下执行WFI指令： 任一外部中断引线被设置为中断模式(相应的外部中断向量在NVIC中必须使能)。参见中断向量表 在以下条件下执行WFE指令： 任一外部中断引线被设置为事件模式。参见唤醒事件管理。
唤醒延时	HSI RC唤醒时间 + 电压调节器从低功耗唤醒的时间。

3.3.5 待机模式

待机模式可实现系统的最低功耗。该模式是在 Cortex-M3 深睡眠模式时关闭电压调节器。整个 1.8V 供电区域被断电。PLL、HSI 和 HSE 振荡器也被断电。SRAM 和寄存器内容丢失。只有备份的寄存器和待机电路维持供电。

进入待机模式

关于如何进入待机模式，详见 表 8。

可以通过设置独立的控制位，选择以下待机模式的功能：

- 独立看门狗(IWDG)：可通过看门狗的 Key 寄存器的写操作或硬件选择来启动 IWDG。一旦启动除了系统复位，看门狗就不能被停止。更多细节参见独立看门狗章节。
- 实时时钟(RTC)：通过备用区域控制寄存器(RCC_BDCR)的 RTCEN 位来设置。
- 内部 RC 振荡器(LSI RC)：通过控制/状态寄存器(RCC_CSR)的 LSION 位来设置。
- 外部 32.768kHz 振荡器(LSE OSC)：通过备用区域控制寄存器(RCC_BDCR)的 LSEON 位设置。

退出待机模式

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

从待机模式唤醒后的代码执行等同于复位后的执行（采样启动模式引脚，读取复位向量等）。电源控制/状态寄存器(PWR_CSR)将会指示内核由待机状态退出。

关于如何退出待机模式，详见 表 8。

表8 待机模式

待机模式	说明
进入	在以下条件下执行WFI或WFE指令： - 设置Cortex™-M3系统控制寄存器中的SLEEPDEEP位 - 设置电源控制寄存器(PWR_CR)中的PDDS位 - 清除电源控制/状态寄存器(PWR_CSR)中的WUF位被
退出	WKUP引脚的上升沿、RTC闹钟、NRST引脚上外部复位、IWDG复位。
唤醒延时	复位阶段时电压调节器的启动。

待机模式下的输入/输出端口状态

在待机模式下，所有的 I/O 引脚处于高阻态，除了以下的引脚：

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

调试模式

默认情况下，如果在进行调试微处理器时，使微处理器进入停止或待机模式，将失去调试连接。这是因为 Cortex™-M3 的内核失去了时钟。

然而，通过设置 DBGMCU_CR 寄存器中的某些配置位，可以在使用低功耗模式下调试软件。更多的细节参考低功耗模式下的调试。

3.3.6 低功耗模式下的自动唤醒(AWU)

RTC 可以在不需要依赖外部中断的情况下唤醒低功耗模式下的微控制器（自动唤醒模式）。RTC 提供一个可编程的时间基数，用于周期性从停止或待机模式下唤醒。通过对备份区域控制寄存器(RCC_BDCR)的 RTCSEL[1:0]位的编程，三个 RTC 时钟源中的二个时钟源可以选作实现此功能。

- 低功耗 32.768kHz 外部晶振(LSE OSC)

  该时钟源提供了一个功耗很低且精确的时间基数。（在典型情形下消耗小于 1μA）

- 低功耗内部 RC 振荡器(LSI RC)

  使用该时钟源，节省了一个 32.768kHz 晶振的成本。但是 RC 振荡器将少许增加电源消耗。

为了用 RTC 闹钟事件将微处理器从停止模式下唤醒，必须进行如下操作：

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

如果要从待机模式中唤醒，不必配置外部中断线 17。

3.4 电源控制寄存器

3.4.1 电源控制寄存器(PWR_CR)

地址偏移：0x00
复位值：0x0000 0000 (从待机模式唤醒时清除)

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留							DBP	PLS[2:0]			PVDE	CSBF	CWUF	PDDS	LPDS
							rw	rw	rw	rw	rw	rc_w1	rc_w1	rw	rw

位 31:9	保留。始终读为0。
位 8	DBP：取消后备区域的写保护 在复位后，RTC和后备寄存器处于被保护状态以防意外写入。设置这位允许写入这些寄存器。 0：禁止写入RTC和后备寄存器 1：允许写入RTC和后备寄存器
位 7:5	PLS[2:0]：PVD电平选择 这些位用于选择电源电压监测器的电压阀值 000：2.2V 001：2.3V 010：2.4V 011：2.5V 100：2.6V 101：2.7V 110：2.8V 111：2.9V 注：详细说明参见数据手册中的电气特性部分。
位 4	PVDE：电源电压监测器(PVD)使能 0：禁止PVD 1：开启PVD
位 3	CSBF：清除待机位 始终读出为0 0：无功效 1：清除SBF待机位(写)

位 2	CWUF：清除唤醒位 始终读出为0 0：无功效 1：2个系统时钟周期后清除WUF唤醒位(写)
位 1	PDDS：掉电深睡眠 与LPDS位协同操作 0：当CPU进入深睡眠时进入停机模式，调压器的状态由LPDS位控制。 1：CPU进入深睡眠时进入待机模式。
位 0	LPDS：深睡眠下的低功耗 PDDS=0时，与PDDS位协同操作 0：在停机模式下电压调压器开启 1：在停机模式下电压调压器处于低功耗模式

3.4.2 电源控制/状态寄存器

地址偏移：0x04
复位值：0x0000 0000 (从待机模式唤醒时不被清除)
与标准的 APB 读相比，读此寄存器需要额外的 APB 周期

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留							EWUP	保留					PVDO	SBF	WUF
							rw						r	r	r

位31:9	保留。始终读为0。
位 8	EWUP：使能WKUP管脚 0：WKUP管脚为通用I/O。WKUP管脚上的事件不能将CPU从待机模式唤醒 1：WKUP管脚用于将CPU从待机模式唤醒，WKUP管脚被强置为输入下拉的配置(WKUP管脚上的上升沿将系统从待机模式唤醒) 注：在系统复位时清除这一位。
位 7:3	保留。始终读为0。
位 2	PVDO：PVD输出 当PVD被PVDE位使能后该位才有效 0：V_{DD}高于由PLS[2:0]选定的PVD阀值 1：V_{DD}低于由PLS[2:0]选定的PVD阀值 注：在待机模式下PVD被停止。因此，待机模式后或复位后，直到设置PVDE位之前，该位为0。

位 1	SBF：待机标志 该位由硬件设置，并只能由POR/PDR(上电/掉电复位)或设置电源控制寄存器(PWR_CR)的CSBF位清除。 0：系统不在待机模式 1：系统进入待机模式
位 0	WUF：唤醒标志 该位由硬件设置，并只能由POR/PDR(上电/掉电复位)或设置电源控制寄存器(PWR_CR)的CWUF位清除。 0：没有发生唤醒事件 1：在WKUP管脚上发生唤醒事件或出现RTC闹钟事件。 注：当WKUP管脚已经是高电平时，在(通过设置EWUP位)使能WKUP管脚时，会检测到一个额外的事件。

3.5 PWR寄存器地址映像

以下表格列出所有 PWR 寄存器。

表9 PWR 寄存器地址映像和复位值

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
000h	PWR_CR	保留																							DBP	PLS[2:0]			PVDE	CSBF	CWUF	PDDS	LPDS
	复位值																								0	0	0	0	0	0	0	0	0
004h	PWR_CSR	保留																							EWUP	保留					PVDO	SBF	WUF
	复位值																								0						0	0	0

4 复位和时钟控制

4.1 复位

STM32F10xxx 支持三种复位形式，分别为系统复位、上电复位和备份区域复位。

4.1.1 系统复位

系统复位将复位除时钟控制寄存器CSR中的复位标志和备份区域中的寄存器以外的所有寄存器(见 图 3)。
当以下事件中的一件发生时，产生一个系统复位：

1. NRST管脚上的低电平（外部复位）
2. 窗口看门狗计数终止（WWDG复位）
3. 独立看门狗计数终止（IWDG复位）
4. 软件复位（SW复位）
5. 低功耗管理复位

可通过查看 RCC_CSR 控制状态寄存器中的复位状态标志位来确认复位事件来源。

软件复位

通过将 Cortex™-M3 中断应用和复位控制寄存器中的 SYSRESETREQ 位置 1，可实现软件复位。请参考 Cortex™-M3 技术参考手册获得进一步信息。

低功耗管理复位

在以下两种情况下可产生低功耗管理复位：

1. 在进入待机模式时产生低功耗管理复位：
   通过将用户选择字节中的nRST_STDBY位置1将使能该复位。这时，即使执行了进入待机模式的过程，系统将被复位而不是进入待机模式。
2. 在进入停止模式时产生低功耗管理复位：
   通过将用户选择字节中的nRST_STOP位置1将使能该复位。这时，即使执行了进入停机模式的过程，系统将被复位而不是进入停机模式。

关于用户选择字节的进一步信息，请参考 STM32F10xxx 闪存编程手册。

4.1.2 电源复位

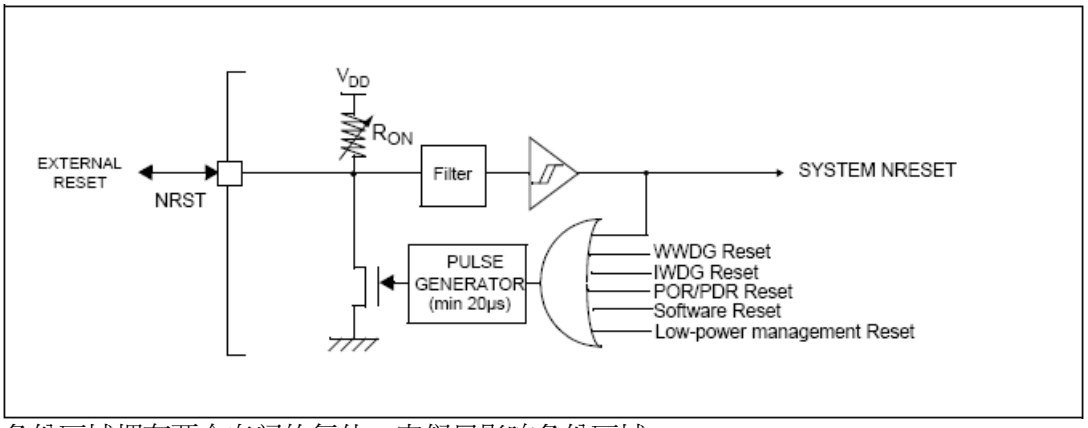
当以下事件中之一发生时，产生电源复位：

1. 上电/掉电复位（POR/PDR复位）
2. 从待机模式中返回

电源复位将复位除了备份区域外的所有寄存器。（见 图 3）

图中复位源将最终作用于 RESET 管脚，并在复位过程中保持低电平。复位入口矢量被固定在地址 0x0000_0000~0x0000_0004。

图6 复位电路



备份区域拥有两个专门的复位，它们只影响备份区域。

4.1.3 备份域复位

当以下事件中之一发生时，产生备份区域复位。

1. 软件复位，备份区域复位可由设置备份区域控制寄存器RCC_BDCR中的BDRST位产生。
2. 在V_{DD}和V_{BAT}两者掉电的前提下，V_{DD}或V_{BAT}上电将引发备份区域复位。

4.2 时钟

三种不同的时钟源可被用来驱动系统时钟(SYSCLK)：

- HSI 振荡器时钟
- HSE 振荡器时钟
- PLL 时钟

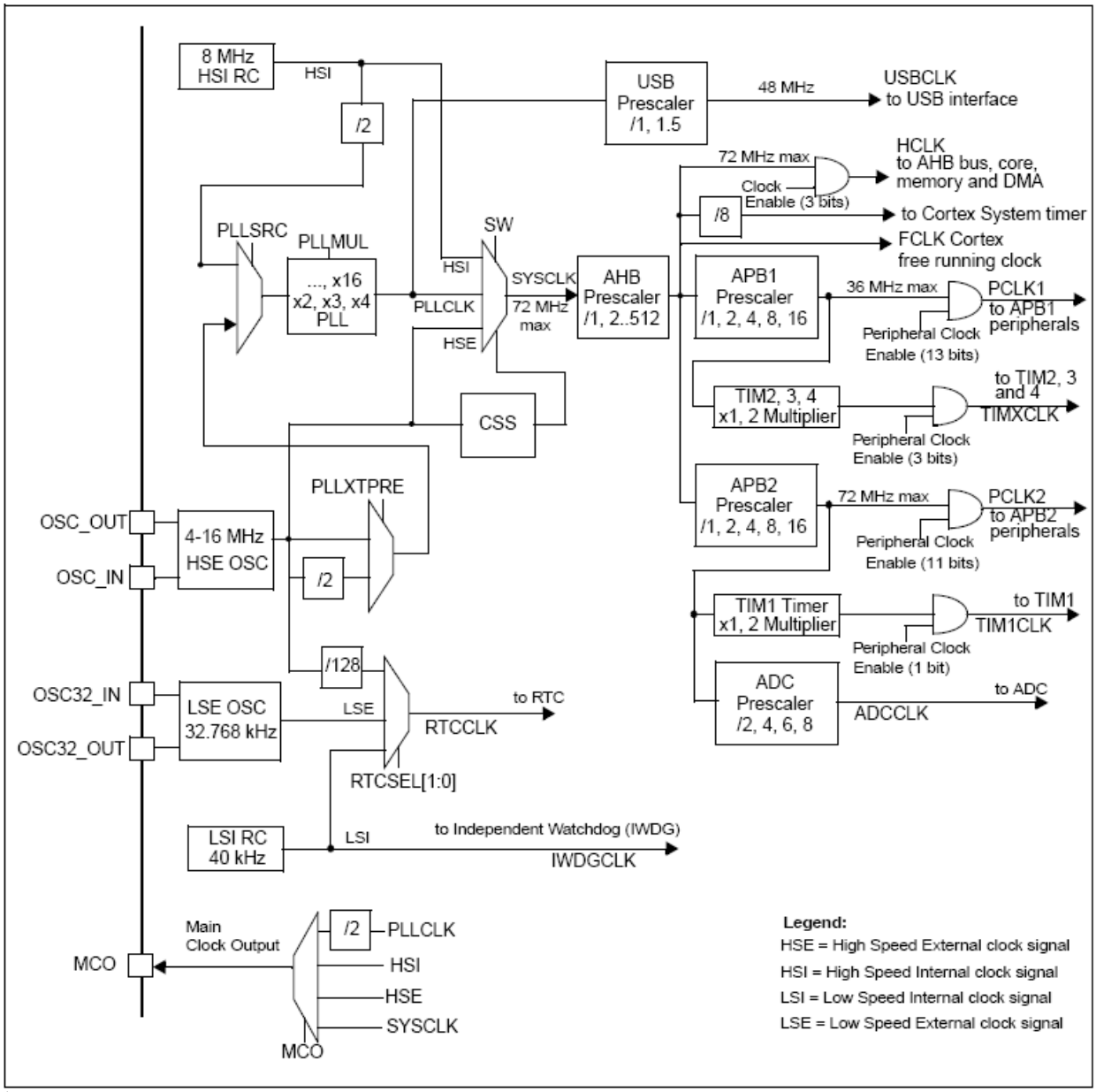
这些设备有以下 2 种二级时钟源：

- 40kHz 低速内部 RC，可以用于驱动独立看门狗和通过程序选择驱动 RTC。RTC 用于从停机/待机模式下自动唤醒系统。

- 32.768kHz 低速外部晶体也可用来通过程序选择驱动 RTC(RTCCLK)。

当不被使用时，任一个时钟源都可被独立地启动或关闭，由此优化系统功耗。

图7 时钟树



[1]当 HSI 被用于作为 PLL 时钟的输入时，系统时钟的最大频率不得超过 64MHz。

用户可通过多个预分频器配置 AHB、高速 APB(APB2)和低速 APB(APB1)域的频率。AHB 和 APB2 域的最大频率是 72MHZ。APB1 域的最大允许频率是 36MHZ。RCC 通过 AHB 时钟 8 分频后供给 Cortex 系统定时器的(SysTick)外部时钟。通过对 SysTick 控制与状态寄存器的设置，可选择上述时钟或 Cortex AHB 时钟作为 SysTick 时钟。ADC 时钟由高速 APB2 时钟经 2、4、6 或 8 分频后获得。

定时器时钟频率是其所在 APB 总线频率的两倍。然而，如果相应的 APB 预分频系数是 1，定时器的时钟频率与所在 APB 总线频率一致。

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

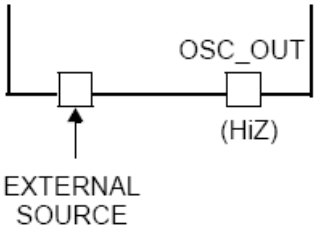
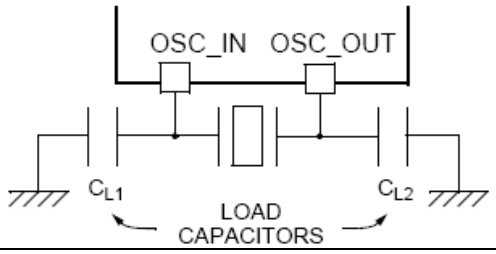
4.2.1 HSE时钟

高速外部时钟信号(HSE)由以下两种时钟源产生：

- HSE 外部晶体/陶瓷谐振器
- HSE 用户外部时钟

为了减少时钟输出的失真和缩短启动稳定时间，晶体/陶瓷谐振器和负载电容器必须尽可能地靠近振荡器管脚。负载电容值必须根据所选择的振荡器来调整。

图8 HSE/LSE 时钟源

	硬件配置
外部时钟	
晶体/陶瓷谐振器	

外部时钟源（HSE旁路）

在这个模式里，必须提供外部时钟。它的频率最高可达 25MHz。用户可通过设置在时钟控制寄存器中的HSEBYP和HSEON位来选择这一模式。外部时钟信号（50%占空比的方波、正弦波或三角波）必须连到SOC_IN管脚，同时保证OSC_OUT管脚悬空。见 图 8。

外部晶体/陶瓷谐振器（HSE晶体）

4-16Mz外部振荡器可为系统提供更为精确的主时钟。相关的硬件配置可参考 图 8，进一步信息可参考数据手册的电气特性部分。

在时钟控制寄存器 RCC_CR 中的 HSERDY 位用来指示高速外部振荡器是否稳定。在启动时，直到这一位被硬件置 1，时钟才被释放出来。如果在时钟中断寄存器 RCC_CIR 中被激活，将会产生相应中断。

HSE 晶体可以通过设置时钟控制寄存器里 RCC_CR 中的 HSEON 位被启动和关闭。

4.2.2 HSI时钟

HSI 时钟信号由内部 8MHZ 的 RC 振荡器产生，它可以直接作为系统时钟或者在 2 分频后作为 PLL 输入。

HSI RC 振荡器能够在不需要任何外部器件的条件下提供系统时钟。它的启动时间比 HSE 晶体振荡器短。然而，即使在校准之后它的时钟频率精度仍较差。

校准

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

如果用户的应用基于不同的电压或环境温度，这将会影响 RC 振荡器的精度。你可以通过利用在时钟控制寄存器里的 HSITRIM[4:0]位来调整 HSI 频率。

时钟控制寄存器中的 HSIRDY 位用来指示 HSI RC 振荡器是否稳定。在时钟启动过程中，直到这一位被硬件置 1，HSI RC 输出时钟才被释放。HSI RC 可由时钟控制寄存器中的 HSION 位来启动和关闭。

如果HSE晶体振荡器失效，HSI时钟会被作为备用时钟源。参考 4.2.7 节 时钟安全系统。

4.2.3 PLL

内部PLL可以用来倍频HSI RC的输出时钟或HSE晶体输出时钟。参考 图 7 和时钟控制寄存器。
PLL 的设置（选择 HIS 振荡器除 2 或 HSE 振荡器为 PLL 的输入时钟，和选择倍频因子）必须在其被激活前完成。一旦 PLL 被激活，这些参数就不能被改动。

如果 PLL 中断在时钟中断寄存器里被允许，当 PLL 准备就绪时，可产生中断申请。

如果需要在应用中使用 USB 接口，PLL 必须被设置为输出 48 或 72MHZ 时钟，用于提供 48MHz 的 USBCLK 时钟。

4.2.4 LSE时钟

LSE 晶体是一个 32.768kHz 的低速外部晶体或陶瓷谐振器。它为实时时钟或者其他定时功能提供一个低功耗且精确的时钟源。
LSE 晶体通过在备份域控制寄存器(RCC_BDCR)里的 LSEON 位启动和关闭。
在备份域控制寄存器(RCC_BDCR)里的 LSERDY 指示 LSE 晶体振荡是否稳定。
在启动阶段，直到这个位被硬件置 1 后，LSE 时钟信号才被释放出来。如果在时钟中断寄存器里被允许，可产生中断申请。

外部时钟源（LSE旁路）

在这个模式里必须提供一个 32.768kHz频率的外部时钟源。你可以通过设置在备份域控制寄存器(RCC_BDCR)里的LSEBYP和LSEON位来选择这个模式。具有50%占空比的外部时钟信号（方波，正弦波或三角波）必须连到OSC32_IN管脚，同时保证OSC32_OUT管脚悬空。见 图 8。

4.2.5 LSI时钟

LSI RC 担当一个低功耗时钟源的角色，它可以在停机和待机模式下保持运行，为独立看门狗和自动唤醒单元提供时钟。LSI 时钟频率大约 40kHz（在 30kHz 和 60kHz 之间）。进一步信息请参考数据手册中有关电气特性部分。

LSI RC 可以通过控制/状态寄存器(RCC_CSR)里的 LSION 位来启动或关闭。

在控制/状态寄存器(RCC_CSR)里的 LSIRDY 位指示低速内部振荡器是否稳定。在启动阶段，直到这个位被硬件设置为 1 后，此时钟才被释放。如果在时钟中断寄存器(RCC_CIR)里被允许，将产生 LSI 中断申请。

4.2.6 系统时钟(SYSCLK)选择

系统复位后，HSI 振荡器被选为系统时钟。当时钟源被直接或通过 PLL 间接作为系统时钟时，它将不能被停止。

只有当目标时钟源准备就绪了(经过启动稳定阶段的延迟或 PLL 稳定)，从一个时钟源到另一个时钟源的切换才会发生。在被选择时钟源没有就绪时，系统时钟的切换不会发生。直至目标时钟源就绪，才发生切换。

在时钟控制寄存器(RCC_CR)里的状态位指示哪个时钟已经准备好了，哪个时钟目前被用作系统时钟。

4.2.7 时钟安全系统(CSS)

时钟安全系统可以通过软件被激活。一旦其被激活，时钟监测器将在 HSE 振荡器启动延迟后被使能，并在 HSE 时钟关闭后关闭。
如果 HSE 时钟发生故障，此振荡器自动地被关闭，时钟失效事件将被送到高级定时器 TIM1 的断路输入端，并产生时钟安全中断 CSSI，允许软件完成营救操作。此 CSSI 中断被连接到 Cortex™-M3 NMI 的中断。

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

如果 HSE 振荡器被直间或间接地作为系统时钟来用的话，（间接的意思是：它被作为 PLL 输入时钟，并且 PLL 时钟被作为系统时钟），时钟故障将导致系统时钟

自动切换到 HSI 振荡器，同时外部 HSE 振荡器被关闭。在时钟失效时，如果 HSE 振荡器时钟（被分频或未被分频）是用作系统时钟的 PLL 的输入时钟，PLL 也将被关闭。

4.2.8 RTC时钟

通过设置备份域控制寄存器(RCC_BDCR)里的 RTCSEL[1:0]位，RTCCLK 时钟源可以由 HSE/128、LSE 或 LSI 时钟提供。除非备份域复位，此选择不能被改变。LSE 时钟在备份域里，但 HSE 和 LSI 时钟不是。因此：

- 如果 LSE 被选为 RTC 时钟：
- 只要 V_{BAT} 维持供电，尽管 V_{DD} 供电被切断，RTC 仍继续工作。
- 如果 LSI 被选为自动唤醒单元(AWU)时钟：
- 如果 V_{DD} 供电被切断， AWU 状态不能被保证
- 如果 HSE 时钟 128 分频后作为 RTC 时钟：
- 如果 V_{DD} 供电被切断或内部电压调压器被关闭(1.8V 域的供电被切断)，RTC 状态不能被保证。

4.2.9 看门狗时钟

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

4.2.10 时钟输出

微控制器允许输出时钟信号到外部 MCO 管脚。

相应的 GPIO 端口寄存器必须被配置为相应功能。以下四个时钟信号可被选作 MCO 时钟：

- . SYSCLK
- . HSI
- . HSE
- . 除 2 的 PLL 时钟

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

4.3 RCC寄存器描述

请参考第 1 章中有关寄存器描述中用到的缩写。

4.3.1 时钟控制寄存器(RCC_CR)

偏移地址: 0x00
复位值: 0x000 XX83，X 代表未定义
访问: 无等待状态, 字, 半字 和字节访问

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留						PLL RDY	PLLON	保留				CSS ON	HSE BYP	HSE RDY	HSE ON
						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]					保留	HSI RDY	HSION
r	r	r	r	r	r	r	r	rw	rw	rw	rw	rw		r	rw

位31:26	保留，始终读为0。
位25	PLLRDY：PLL时钟就绪标志 PLL锁定后由硬件置1。 0：PLL未锁定 1：PLL锁定
位24	PLLON：PLL使能 由软件置1或清零。当进入待机和停止模式时，该位由硬件清零。 当PLL时钟被用作或被选择将要作为系统时钟时，该位不能被清零。 0：PLL关闭 1：PLL使能
位23:20	保留，始终读为0。
位19	CSSON：时钟安全系统使能 使能时钟监测器。 0：时钟监测器关闭 1：如果外部1-25MHz时钟就绪，时钟监测器开启。
位18	HSEBYP：外部高速时钟旁路 在调试模式下由软件置1或清零来旁路外部晶体振荡器。只有在外部1-25MHz振荡器关闭的情况下，该位才可以写入。 0：外部1-25MHz振荡器没有旁路 1：外部1-25MHz外部晶体振荡器被旁路。
位17	HSERDY：外部高速时钟就绪标志 由硬件置1来指示外部时钟已经稳定。在HSEON位清零后，该位需要6个外部时钟周期清零。 0：外部1-25MHz时钟没有就绪 1：外部1-25MHz时钟就绪
位16	HSEON：外部高速时钟使能 由软件置1或清零。当进入待机和停止模式时，该位由硬件清零，关闭外部时钟。当外部时钟被用作或被选择将要作为系统时钟时，该位不能被清零。 0：HSE振荡器关闭

	1：HSE振荡器开启
位15:8	HSICAL[7:0]：内部高速时钟校准 在系统启动时，这些位被自动初始化
位7:3	HSITRIM[4:0]：内部高速时钟调整 由软件写入来调整内部高速时钟，它们被叠加在HSICAL[5:0]数值上。 这些位在HSICAL[7:0]的基础上，让用户可以输入一个调整数值，根据电压和温度的变化调整内部HSI RC振荡器的频率。 默认数值为16，在TA= 25°C时这个默认的数值可以把HSI调整到8MHz；增大HSICAL的数值则增大HSI RC振荡器的频率，反之则减小RC振荡器的频率；每步HSICAL的变化调整约40kHz。
位2	保留，始终读为0。
位1	HSIRDY：内部高速时钟就绪标志 由硬件置1来指示内部8MHz时钟已经稳定。在HSION位清零后，该位需要6个内部时钟周期清零。 0：内部8MHz时钟没有就绪 1：内部8MHz时钟就绪
位0	HSION：内部高速时钟使能 由软件置1或清零。 当从待机和停止模式返回或用作系统时钟的外部1-25MHz时钟发生故障时，该位由硬件置1来启动内部8MHz的RC振荡器。当内部8MHz时钟被直接或间接地用作或被选择将要作为系统时钟时，该位不能被清零。 0：内部8MHz时钟关闭 1：内部8MHz时钟开启

4.3.2 时钟配置寄存器(RCC_CFGR)

偏移地址: 0x04
复位值: 0x0000 0000
访问: 0 到 2 个等待周期, 字, 半字 和字节访问
只有当访问发生在时钟切换时，才会插入 1 或 2 个等待周期。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留					MCO[2:0]			保留	USB PRE	PLLMUL[3:0]				PLL XTPRE	PLL SRC
					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
ADCPRE[1:0]		PPRE2[2:0]			PPRE1[2:0]			HPRE[3:0]				SWS[1:0]		SW[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	r	r	rw	rw

位31:26	保留，始终读为0。
位26:24	MCO 微控制器时钟输出 由软件置1或清零。 0xx: 没有时钟输出 100: 系统时钟输出

	101: 内部8MHz的RC振荡器时钟输出 110: 外部1-25MHz振荡器时钟输出 111: PLL时钟2分频后输出 注意：– 该时钟输出在启动和切换MCO时钟源时可能会被截断。 - 在系统时钟作为输出时钟时，请保证输出时钟频率不超过50MHz (IO口最高频率)
位22	USBPRE：USB预分频 由软件设置来产生48MHz的USB时钟。在RCC_APB1ENR寄存器中使能USB时钟之前，必须保证该位已经有效。如果USB时钟被使能，该位可以被清零。 0：PLL时钟1.5倍分频作为USB时钟 1：PLL时钟直接作为USB时钟
位21:18	PLLMUL：PLL倍频系数 由软件设置来确定PLL倍频系数。只有在PLL关闭的情况下才可被写入。 注意：PLL的输出频率不能超过72MHz 0000：PLL 2倍频输出 0001：PLL 3倍频输出 0010：PLL 4倍频输出 0011：PLL 5倍频输出 0100：PLL 6倍频输出 0101：PLL 7倍频输出 0110：PLL 8倍频输出 0111：PLL 9倍频输出 1000：PLL 10倍频输出 1001：PLL 11倍频输出 1010：PLL 12倍频输出 1011：PLL 13倍频输出 1100：PLL 14倍频输出 1101：PLL 15倍频输出 1110：PLL 16倍频输出 1111：PLL 16倍频输出
位17	PLLXTPRE：HSE分频器作为PLL输入 由软件设置来分频HSE后作为PLL输入时钟。该位只有在PLL关闭时才可以被写入。 0：HSE不分频 1：HSE 2分频
位16	PLLSRC：PLL输入时钟源 由软件设置来选择PLL输入时钟源。该位只有在PLL关闭时才可以被写入。 0：HSI时钟2分频后作为PLL输入时钟 1：HSE时钟作为PLL输入时钟。
位15:14	ADCPRE：ADC预分频 由软件设置来确定ADC时钟频率 00：PCLK2 2分频后作为ADC时钟 01：PCLK2 4分频后作为ADC时钟 10：PCLK2 6分频后作为ADC时钟 11：PCLK2 8分频后作为ADC时钟
位13:11	PPRE2：高速APB预分频（APB2）

	由软件设置来控制高速APB2预分频系数。 0xx：HCLK不分频 100：HCLK 2分频 101：HCLK 4分频 110：HCLK 8分频 111：HCLK 16分频
位10:8	PPRE1：低速APB预分频（APB1） 由软件设置来控制低速APB1预分频系数。软件必须保证APB1时钟频率不超过36MHz。 0xx：HCLK不分频 100：HCLK 2分频 101：HCLK 4分频 110：HCLK 8分频 111：HCLK 16分频
位7:4	HPRE： AHB预分频 由软件设置来控制AHB预分频系数。 0xxx：SYSCLK不分频 1000：SYSCLK 2分频 1001：SYSCLK 4分频 1010：SYSCLK 8分频 1011：SYSCLK 16分频 1100：SYSCLK 64分频 1101：SYSCLK 128分频 1110：SYSCLK 256分频 1111：SYSCLK 512分频 注意：当AHB时钟的预分频系数大于1时，必须保持预取缓冲器开启。详见读闪存存储器一节。
位3:2	SWS：系统时钟切换状态 由硬件置1和清零来指示哪一个时钟源被作为系统时钟。 00：HSI作为系统时钟 01：HSE作为系统时钟 10：PLL输出作为系统时钟 11：不可用
位1:0	SW：系统时钟切换 由软件设置来选择系统时钟源。 在从停止或待机模式中返回时或直接或间接作为系统时钟的HSE出现故障时，由硬件强制选择HSI作为系统时钟（如果时钟安全系统已经启动） 00：HSI作为系统时钟 01：HSE作为系统时钟 10：PLL输出作为系统时钟 11：不可用

4.3.3 时钟中断寄存器 (RCC_CIR)

偏移地址: 0x08
复位值: 0x0000 0000
访问:无等待周期, 字, 半字 和字节访问

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留								CSSC	保留		PLL RDYC	HSE RDYC	HIS RDYC	LSE RDYC	LSI RDYC
								w			w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留			PLL RDYIE	HSE RDYIE	HSI RDYIE	LSE RDYIE	LSI RDYIE	CSSF	保留		PLL RDYF	HSE RDYF	HSI RDYF	LSE RDYF	LSI RDYF
			rw	rw	rw	rw	rw	r			r	r	r	r	r

位31：24	保留，始终读为0。
位23	CSSC：时钟安全系统中断清除 由软件置1来清除CSSF安全系统中断标志位CSSF。 在清除操作完成后，该位由硬件复位。 0: CSSF安全系统中断标志位未清零 1: CSSF安全系统中断标志位清零
位22：21	保留，始终读为0。
位20	PLLRDYC：PLL就绪中断清除 由软件置1来清零PLL就绪中断标志位PLLRDYF。 在清除操作完成后，该位由硬件复位。 0: PLL就绪中断标志位PLLRDYF未清零 1: PLL就绪中断标志位PLLRDYF清零
位19	HSERDYC：HSE就绪中断清除 由软件置1来清零HSE就绪中断标志位HSERDYF。 在清除操作完成后，该位由硬件复位。 0: HSE就绪中断标志位HSERDYF未清零 1: HSE就绪中断标志位HSERDYF清零
位18	HSIRDYC HSI就绪中断清除 由软件置1来清零HSI就绪中断标志位HSIRDYF。 在清除操作完成后，该位由硬件复位。 0: HSI就绪中断标志位HSIRDYF未清零 1: HSI就绪中断标志位HSIRDYF清零
位17	LSERDYC：LSE就绪中断清除 由软件置1来清零LSE就绪中断标志位LSERDYF。 在清除操作完成后，该位由硬件复位。 0: LSE就绪中断标志位LSERDYF未清零 1: LSE就绪中断标志位LSERDYF清零
位16	LSIRDYC：LSI就绪中断清除 由软件置1来清零LSI就绪中断标志位LSIRDYF。 在清除操作完成后，该位由硬件复位。

	0: LSI就绪中断标志位LSIRDYF未清零 1: LSI就绪中断标志位LSIRDYF清零
位15:13	保留，始终读为0。
位12	PLLRDYIE：PLL就绪中断使能 由软件置1或清零来使能或关闭PLL就绪中断 0：PLL就绪中断关闭 1：PLL就绪中断使能
位11	HSERDYIE：HSE就绪中断使能 由软件置1或清零来使能或关闭外部1-25MHz振荡器就绪中断 0：HSE就绪中断关闭 1：HSE就绪中断使能
位10	HSIRDYIE：HSI就绪中断使能 由软件置1或清零来使能或关闭内部8MHz RC振荡器就绪中断 0：HSI就绪中断关闭 1：HSI就绪中断使能
位9	LSERDYIE：LSE就绪中断使能 由软件置1或清零来使能或关闭外部40kHz RC振荡器就绪中断 0：LSE就绪中断关闭 1：LSE就绪中断使能
位8	LSIRDYIE：LSI就绪中断使能 由软件置1或清零来使能或关闭内部40kHz RC振荡器就绪中断 0：LSI就绪中断关闭 1：LSI就绪中断使能
位7	CSSF：时钟安全系统中断标志 由软件通过置1 CSSC位来清零 在外部1-25MHz振荡器时钟出现故障时，由硬件置1 0：无HSE时钟失效安全系统中断 1：产生HSE时钟失效安全系统中断
位6：5	保留，始终读为0。
位4	PLLRDYF：PLL就绪中断标志 由软件通过置1 PLLRDYC位来清零 在PLL就绪且PLLRDYIE位被置1时，由硬件置1 0：无PLL就绪中断 1：产生PLL就绪中断
位3	HSERDYF：HSE就绪中断标志 由软件通过置1 HSERDYC位来清零 在外部低速时钟就绪且HSERDYIE位被置1时，由硬件置1 0：无外部1-25MHz振荡器就绪中断 1：产生外部1-25MHz振荡器就绪中断
位2	HSIRDYF：HSI就绪中断标志 由软件通过置1 HSIRDYC位来清零 在内部高速时钟就绪且HSIRDYIE位被置1时，由硬件置1 0：无内部8MHz RC振荡器就绪中断 1：产生内部8MHz RC振荡器就绪中断

位1	LSERDYF：LSE就绪中断标志 由软件通过置1 LSERDYC位来清零 在外部低速时钟就绪且LSERDYIE位被置1时，由硬件置1 0：无外部32kHz振荡器就绪中断 1：产生外部32kHz振荡器就绪中断
位0	LSIRDYF：LSI就绪中断标志 由软件通过置1 LSIRDYC位来清零 在内部低速时钟就绪且LSIRDYIE位被置1时，由硬件置1 0：无内部40kHz RC振荡器就绪中断 1：产生内部40kHz RC振荡器就绪中断

4.3.4 APB2 外设复位寄存器 (RCC_APB2RSTR)

偏移地址: 0x0C
复位值: 0x0000 0000
访问：无等待周期, 字, 半字 和字节访问

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

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

位31:15	保留，始终读为0。
位14	USART1RST：USART1复位 由软件置1或清零 0：无效 1：复位USART1
位13	保留，始终读为0。
位12	SPI1RST：SPI1复位 由软件置1或清零 0：无效 1：复位SPI1
位11	TIM1RST：TIM1复位 由软件置1或清零 0：无效 1：复位TIM1
位10	ADC2RST：ADC2复位 由软件置1或清零 0：无效 1：复位ADC2

位9	ADC1RST：ADC1复位 由软件置1或清零 0：无效 1：复位ADC1
位8:7	保留，始终读为0。
位6	IOPERST：IO口E复位 由软件置1或清零 0：无效 1：复位IO口E
位5	IOPDRST：IO口D复位 由软件置1或清零 0：无效 1：复位IO口D
位4	IOPCRST：IO口C复位 由软件置1或清零 0：无效 1：复位IO口C
位3	IOPBRST：IO口B复位 由软件置1或清零 0：无效 1：复位IO口B
位2	IOPARST：IO口A复位 由软件置1或清零 0：无效 1：复位IO口A
位1	保留，始终读为0。
位0	AFIORST：辅助功能IO复位 由软件置1或清零 0：无效 1：复位辅助功能

4.3.5 APB1 外设复位寄存器 (RCC_APB1RSTR)

偏移地址：0x10
复位值：0x0000 0000
访问：无等待周期，字， 半字和字节访问

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留			PWR RST	BKP RST	保留	CAN RST	保留	USB RST	I2C2 RST	I2C1 RST	保留		USART3 RST	USART2 RST	保留
			rw	rw		rw		rw	rw	rw			rw	rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留	SPI2 RST	保留		WWDG RST	保留								TIM4 RST	TIM3 RST	TIM2 RST
	rw			rw									rw	rw	rw

位31:29	保留，始终读为0。
位28	PWRRST：电源复位 由软件置1或清零 0：无效 1：复位电源电路
位27	BKPRST：备份复位 由软件置1或清零 0：无效 1：复位备份电路
位26	保留，始终读为0。
位25	CANRST：CAN复位 由软件置1或清零 0：无效 1：复位CAN
位24	保留，始终读为0。
位23	USBRST：USB复位 由软件置1或清零 0：无效 1：复位USB
位22	I2C2RST：I2C 2复位 由软件置1或清零 0：无效 1：复位I2C 2
位21	I2C1RST：I2C 1复位 由软件置1或清零 0：无效 1：复位I2C 1
位20:19	保留，始终读为0。
位18	USART3RST：USART3复位

	由软件置1或清零 0：无效 1：复位USART3
位17	USART2RST：USART2复位 由软件置1或清零 0：无效 1：复位USART2
位16:15	保留，始终读为0。
位14	SPI2RST：SPI2复位 由软件置1或清零 0：无效 1：复位SPI2
位13:12	保留，始终读为0。
位11	WWDGRST：窗口看门狗复位 由软件置1或清零 0：无效 1：复位窗口看门狗
位10：3	保留，始终读为0。
位2	TIM4RST：定时器4复位 由软件置1或清零 0：无效 1：复位定时器4
位1	TIM3RST：定时器3复位 由软件置1或清零 0：无效 1：复位定时器3
位0	TIM2RST：定时器2复位 由软件置1或清零 0：无效 1：复位定时器2

4.3.6 AHB外设时钟使能寄存器 (RCC_AHBENR)

偏移地址：0x14
复位值：0x0000 0014
访问：无等待周期, 字, 半字 和字节访问

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留											FLITFEN	保留	SRAMEN	保留	DMAEN
											rw		rw		rw

位31:5	保留，始终读为0。
位4	FLITFEN：闪存接口电路时钟使能 由软件来置1或清零来开启或关闭睡眠模式时闪存接口电路时钟。 0：睡眠模式时闪存接口电路时钟关闭 1：睡眠模式时闪存接口电路时钟开启
位3	保留，始终读为0。
位2	SRAMEN：SRAM时钟使能 由软件来置1或清零来开启或关闭睡眠模式时SRAM时钟。 0：睡眠模式时SRAM时钟关闭 1：睡眠模式时SRAM时钟开启
位1	保留，始终读为0。
位0	DMAEN：DMA时钟使能 由软件来置1或清零来开启或关闭DMA时钟。 0：DMA时钟关闭 1：DMA时钟开启

4.3.7 APB2 外设时钟使能寄存器(RCC_APB2ENR)

偏移地址：0x18
复位值：0x0000 0000
访问：字，半字和字节访问
通常无访问等待周期。但在 APB2 总线上外设被访问时，等待状态将被插入直到外设访问结束。

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留	USART1EN	保留	SPI1EN	TIM1EN	ADC2EN	ADC1EN	保留		IOPEEN	IOPDEN	IOPCEN	IOPBEN	IOPAEN	保留	AFIOEN
	rw		rw	rw	rw	rw			rw	rw	rw	rw	rw		rw

位31:15	保留，始终读为0。
位14	USART1EN：USART1时钟使能 由软件来置1或清零 0：USART1时钟关闭 1：USART1时钟开启
位13	保留，始终读为0。
位12	SPI1EN：SPI1时钟使能 由软件来置1或清零 0：SPI1时钟关闭 1：SPI1时钟开启
位11	TIM1EN：TIM1时钟使能 由软件来置1或清零 0：TIM1时钟关闭 1：TIM1时钟开启
位10	ADC2EN：ADC2时钟使能 由软件来置1或清零 0：ADC2时钟关闭 1：ADC2时钟开启
位9	ADC1EN：ADC1时钟使能 由软件来置1或清零 0：ADC1时钟关闭 1：ADC1时钟开启
位8:7	保留，始终读为0。
位6	IOPEEN：IO口E时钟使能 由软件来置1或清零 0：IO口E时钟关闭 1：IO口E时钟开启

位5	IOPDEN：IO口D时钟使能 由软件来置1或清零 0：IO口D时钟关闭 1：IO口D时钟开启
位4	IOPCEN：IO口C时钟使能 由软件来置1或清零 0：IO口C时钟关闭 1：IO口C时钟开启
位3	IOPBEN：IO口B时钟使能 由软件来置1或清零 0：IO口B时钟关闭 1：IO口B时钟开启
位2	IOPAEN：IO口A时钟使能 由软件来置1或清零 0：IO口A时钟关闭 1：IO口A时钟开启
位1	保留，始终读为0。
位0	AFIOEN：辅助功能IO时钟使能 由软件来置1或清零 0：辅助功能IO时钟关闭 1：辅助功能IO时钟开启

4.3.8 APB1 外设时钟使能寄存器(RCC_APB1ENR)

偏移地址：0x1C
复位值：0x0000 0000
访问：字、半字和字节访问

通常无访问等待周期。但在 APB1 总线上外设被访问时，等待状态将被插入直到外设访问结束。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留			PWR EN	BKP EN	保留	CAN EN	保留	USB EN	I2C2 EN	I2C1 EN	保留		USART3 EN	USART2 EN	保留
			rw	rw		rw		rw	rw	rw			rw	rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留	SPI2 EN	保留		WWDG EN	保留								TIM4 EN	TIM3 EN	TIM2 EN
	rw			rw									rw	rw	rw

位31:29	保留，始终读为0。
位28	PWREN：电源时钟使能 由软件来置1或清零 0：电源时钟关闭 1：电源时钟开启

位27	BKPEN：备份时钟使能 由软件来置1或清零 0：备份时钟关闭 1：备份时钟开启
位26	保留，始终读为0。
位25	CANEN：CAN时钟使能 由软件来置1或清零 0：CAN时钟关闭 1：CAN时钟开启
位24	保留，始终读为0。
位23	USBEN：USB时钟使能 由软件来置1或清零 0：USB时钟关闭 1：USB时钟开启
位22	I2C2EN：I2C 2时钟使能 由软件来置1或清零 0：I2C 2时钟关闭 1：I2C 2时钟开启
位21	I2C1EN：I2C 1时钟使能 由软件来置1或清零 0：I2C 1时钟关闭 1：I2C 1时钟开启
位20:19	保留，始终读为0。
位18	USART3EN：USART 3时钟使能 由软件来置1或清零 0：USART 3时钟关闭 1：USART 3时钟开启
位17	USART2EN：USART 2时钟使能 由软件来置1或清零 0：USART 2时钟关闭 1：USART 2时钟开启
位16:15	保留，始终读为0。
位14	SPI2EN：SPI 2时钟使能 由软件来置1或清零 0：SPI 2时钟关闭 1：SPI 2时钟开启
位13:12	保留，始终读为0。
位11	WWDGEN：窗口看门狗时钟使能 由软件来置1或清零 0：窗口看门狗时钟关闭 1：窗口看门狗时钟开启
位10:3	保留，始终读为0。

位2	TIM4EN：定时器4时钟使能 由软件来置1或清零 0：定时器4时钟关闭 1：定时器4时钟开启
位1	TIM3EN：定时器3时钟使能 由软件来置1或清零 0：定时器3时钟关闭 1：定时器3时钟开启
位0	TIM2EN：定时器2时钟使能 由软件来置1或清零 0：定时器2时钟关闭 1：定时器2时钟开启

4.3.9 备份域控制寄存器 (RCC_BDCR)

偏移地址：0x20
复位值：0x0000 0000，只能由备份域复位有效复位
访问：0 到 3 等待周期，字、半字和字节访问
一旦连续对该寄存器进行访问，等待状态将被插入。

注意：　备份域控制寄存器中(RCC_BDCR)的LSEON、LSEBYP、RTCSEL和RTCEN位处于备份域。由此，这些位在复位后被写保护，只有在电源控制寄存器(PWR_CR)中的DBP位置1之后才能对这些位进行改动。进一步信息请参考9.1节。这些位只能由备份域复位或V_{BAT}上电复位。任何内部或外部复位不会影响这些位。

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RTC EN	保留					RTCSEL[1:0]		保留					LSE BYP	LSE RDY	LSEON
rw						rw	rw						rw	rw	rw

位31:17	保留，始终读为0。
位16	BDRST：备份域软件复位 由软件置1或清零 0：复位未激活 1：复位整个备份域
位15	RTCEN：RTC时钟使能 由软件置1或清零 0：RTC时钟关闭 1：RTC时钟开启
位14:10	保留，始终读为0。

位9:8	RTCSEL[1:0]：RTC时钟源选择 由软件设置来选择RTC时钟源。一旦RTC时钟源被选定，直到下次后备域被复位，它不能在被改变。可通过设置BDRST位来清除。 00：无时钟 01：LSE振荡器作为RTC时钟 10：LSI振荡器作为RTC时钟 11：HSE振荡器在128分频后作为RTC时钟
位7:3	保留，始终读为0。
位2	LSEBYP：外部低速时钟振荡器旁路 在调试模式下由软件置1或清零来旁路LSE。只有在外部32kHz振荡器关闭时，才能写入该位 0：LSE时钟未被旁路 1：LSE时钟被旁路
位1	LSERDY：外部低速LSE就绪 由硬件置1或清零来指示是否外部32kHz振荡器就绪。在LSEON被清零后，该位需要6个外部低速振荡器的周期才被清零。 0：外部32kHz振荡器未就绪 1：外部32kHz振荡器就绪
位0	LSEON：外部低速振荡器使能 由软件置1或清零。 0：外部32kHz振荡器关闭 1：外部32kHz振荡器开启

4.3.10 控制/状态寄存器 (RCC_CSR)

偏移地址：0x24
复位值：0x0C00 0000，除复位标志外由系统复位清除，复位标志只能由电源复位清除。
访问：0 到 3 等待周期，字、半字和字节访问
一旦连续对该寄存器进行访问，等待状态将被插入。

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	保留	RMVF	保留							
rw	rw	rw	rw	rw	rw		rw								

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留														LSI RDY	LSION
														r	rw

位31	LPWRRSTF：低功耗复位标志 由软件通过写RMVF位清零，在低功耗管理复位发生时由硬件置1。 0：无低功耗管理复位发生 1：发生低功耗管理复位 关于低低功耗管理复位的进一步信息，请参考低功耗管理复位章节。
位30	WWDGRSTF：窗口看门狗复位标志

	由软件通过写RMVF位清零，在窗口看门狗复位发生时由硬件置1。 0：无窗口看门狗复位发生 1：发生窗口看门狗复位
位29	IWDGRSTF：独立看门狗复位标志 由软件通过写RMVF位清零，在独立看门狗复位发生时由硬件置1。 0：无独立看门狗复位发生 1：发生独立看门狗复位
位28	SFTRSTF：软件复位标志 由软件通过写RMVF位清零，在软件复位发生时由硬件置1。 0：无软件复位发生 1：发生软件复位
位27	PORRSTF：上电/掉电复位标志 由软件通过写RMVF位清零，在上电/掉电复位发生时由硬件置1。 0：无上电/掉电复位发生 1：发生上电/掉电复位
位26	PINRSTF：NRST管脚复位标志 由软件通过写RMVF位清零，在NRST管脚复位发生时由硬件置1。 0：无NRST管脚复位发生 1：发生NRST管脚复位
位25	保留，读操作返回0
位24	RMVF 清除复位标志 由软件置1或清零来清除复位标志。 0：保持复位标志 1：清零复位标志
位23:2	保留，读操作返回0
位1	LSIRDY：内部低速时钟就绪 由硬件置1或清零来指示内部40kHz RC振荡器是否就绪。在LSION清零后，3个内部40kHz RC振荡器的周期后LSIRDY被清零。 0：LSI(内部40kHz RC振荡器)时钟未就绪 1：LSI(内部40kHz RC振荡器)时钟就绪
位0	LSION：内部低速振荡器使能 由软件置1或清零。 0：内部40kHz RC振荡器关闭 1：内部40kHz RC振荡器开启

4.4 RCC寄存器地址映像

表10 RCC 寄存器地址映像和复位值

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
000h	RCC_CR	保留						PLLRDY	PLLON	保留				CSSON	HSEBYP	HSERDY	HSEON	HSICAL[7:0]								HSITRIM[4:0]					保留	HSIRDY	HSION
	复位值							0	0					0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0		1	1
004h	RCC_CFGR	保留					MCO[2:0]			保留	USBPRE	PLLMUL[3:0]				PLLXTPRE	PLLSRC	ADC PRE [1:0]		PRRE2 [2:0]			PRRE1 [2:0]			HPRE[3:0]				SWS [1:0]		SW [1:0]	
	复位值						0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
008h	RCC_CIR	保留								CSSC	保留	PLLRDYC	HSERDYC	HSIRDYC	LSERDYC	LSIRDYC	保留			PLLRDYIE	HSERDYIE	HSIRDYIE	LSERDYIE	LSIRDYIE	CSSF	保留			PLLRDYF	HSERDYF	HSIRDYF	LSERDYF	LSIRDYF
	复位值									0		0	0	0	0	0				0	0	0	0	0	0				0	0	0	0	0
00Ch	RCC_APB2RSTR	保留																	USART1RST	保留	SPI1RST	TIM1RST	ADC2RST	ADC1RST	保留		IOPERST	IOPDRST	IOPCRST	IOPBRST	IOPARST	保留	AFIORST
	复位值																		0		0	0	0	0			0	0	0	0	0		0
010h	RCC_APB1RSTR	保留			PWRRST	BKPRST	保留	CANRST	保留	USBRST	I2C2RST	I2C1RST	保留	USART3RST	USART2RST	保留		SPI2RST	保留		WWDGRST	保留									TIM4RST	TIM3RST	TIM2RST
	复位值				0	0		0		0	0	0		0	0			0			0										0	0	0
014h	RCC_APB1RSTR	保留																											FLITFEN	保留	SRAMEN	保留	DMAEN
	复位值																												1		1		0
018h	RCC_APB2ENR	保留																	USART1EN	保留	SPI1REN	TIM1EN	ADC2EN	ADC1EN	保留		IOPEEN	IOPDEN	IOPCEN	IOPBEN	IOPAEN	保留	AFIOEN
	复位值																		0		0	0	0	0			0	0	0	0	0		0
01Ch	RCC_APB1ENR	保留			PWREN	BKPEN	保留	CANEN	保留	USBEN	I2C2EN	I2C1EN	保留	USART3EN	USART2EN	保留		SPI2EN	保留		WWDGEN	保留									TIM4EN	TIM3EN	TIM2EN
	复位值				0	0		0		0	0	0		0	0			0			0										0	0	0
020h	RCC_BDCR	保留														BDRST	RTCEN	保留					RTC SEL [1:0]		保留							LSEBYP	LSERDYF	LSEON
	复位值															0	0						0	0								0	0	0
024h	RCC_CSR	LPWRRSTF	WWDGRSTF	IWDGRSTF	SFTRSTF	PORRSTF	PINRSTF	保留	RMVF	保留																						LSIRDY	LSION
	复位值	0	0	0	0	1	1		0																							0	0

5 通用和复用功能I/O(GPIO和AFIO)

5.1 GPIO功能描述

每个 GPI/O 端口有两个 32 位配置寄存器(GPIOx_CRL，GPIOx_CRH)，两个 32 位数据寄存器(GPIOx_IDR，GPIOx_ODR)，一个 32 位置位/复位寄存器(GPIOx_BSRR)，一个 16 位复位寄存器(GPIOx_BRR)和一个 32 位锁定寄存器(GPIOx_LCKR)。

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

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

每个 I/O 端口位可以自由编程，然而 I/0 端口寄存器必须按 32 位字被访问(不允许半字或字节访问)。GPIOx_BSRR 和 GPIOx_BRR 寄存器允许对任何 GPIO 寄存器的读/更改的独立访问；这样，在读和更改访问之间产生 IRQ 时不会发生危险。

图 9 给出了一个I/O端口位的基本结构。

图9 I/O 端口位的基本结构

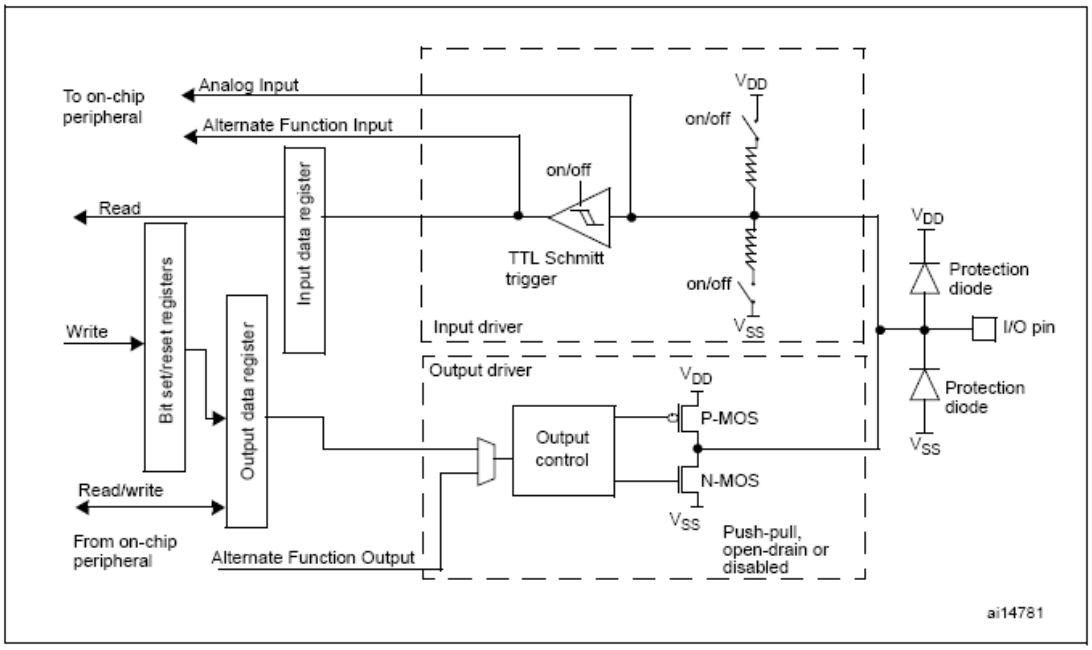
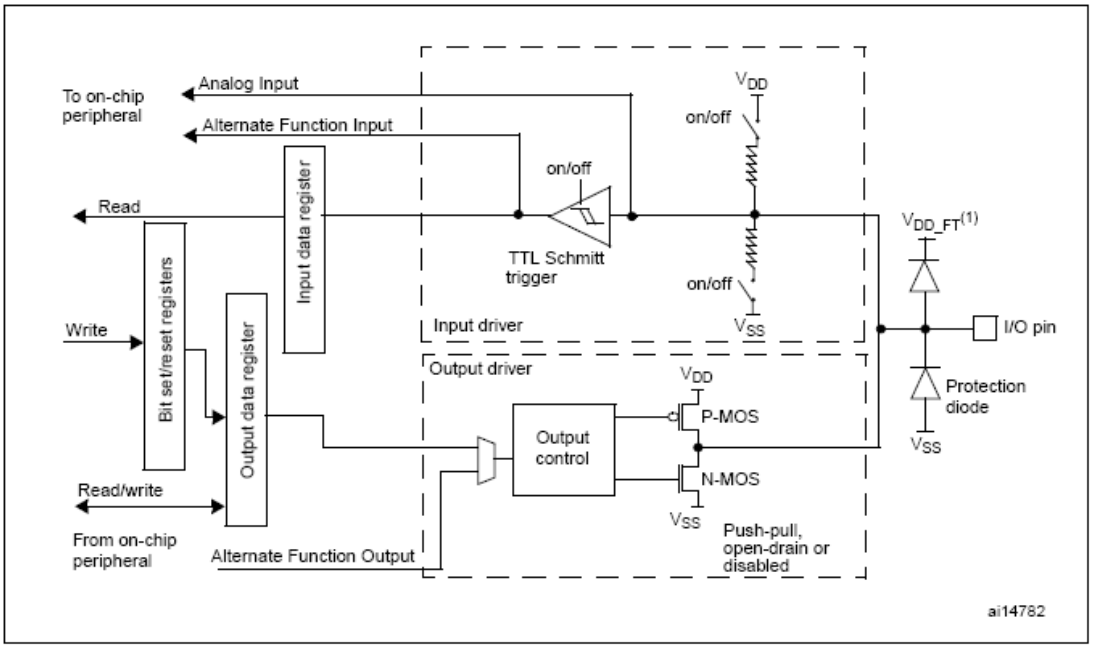


图10 5 伏兼容 I/O 端口位的基本结构



(1) VDD_FT 对 5 伏兼容 I/O 脚是特殊的，它与 VDD 不同

表11 端口位配置表

配置模式		CNF1	CNF0	MODE1	MODE0	PxODR寄存器
通用输出	推挽式(Push-Pull)	0	0	01 10 11 见表12		0 或 1
	开漏(Open-Drain)		1			0 或 1
复用功能输出	推挽式(Push-Pull)	1	0			不使用
	开漏(Open-Drain)		1			不使用
输入	模拟输入	0	0	00		不使用
	浮空输入		1			不使用
	下拉输入	1	0			0
	上拉输入					1

表12 输出模式位

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

5.1.1 通用I/O(GPIO)

复位期间和刚复位后，复用功能未开启，I/O 端口被配置成浮空输入模式(CNFx[1:0]=01b，MODEx[1:0]=00b)。

复位后，JTAG 引脚被置于输入上拉或下拉模式：

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

当作为输出配置时，写到输出数据寄存器上的值(GPIOx_ODR)输出到相应的 I/O 引脚。可以以推挽模式或开漏模式(当输出 0 时，只有 N-MOS 被打开)使用输出驱动器。

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

所有 GPIO 引脚有一个内部弱上拉和弱下拉，当配置为输入时，它们可以被激活也可以不被激活。

5.1.2 单独的位设置或位清除

当对 GPIOx_ODR 的个别位编程时，软件不需要禁止中断：在单次 APB2 写操作里，可以只更改一个或多个位。
这是通过对“置位/复位寄存器”(GPIOx_BSRR，复位是 GPIOx_BRR)中想要更改的位写 1 来实现的。没被选择的位将不被更改。

5.1.3 外部中断/唤醒线

所有端口都有外部中断能力。为了使用外部中断线，端口必须配置成输入模式。
更多的关于外部中断的信息，参考：

- 6.2：外部中断/事件控制器
- 6.2.3：唤醒事件管理

5.1.4 复用功能(AF)

使用默认复用功能前必须对端口位配置寄存器编程。

- 对于复用的输入功能，端口可以配置成：
  - 输入模式（浮空、上拉或下拉）
  - 复用功能输出模式：输入驱动器被配置成浮空输入模式
- 对于复用输出功能，端口必须配置成复用功能输出模式（推挽或开漏）。
- 对于双向复用功能，端口位必须配置复用功能输出模式（推挽或开漏）。这时，输入驱动器被配置成浮空输入模式。

如果把一端口配置成复用输出功能，将使引脚和输出寄存器断开，并和片上外设的输出信号连接。

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

5.1.5 软件重新映射I/O复用功能

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

5.1.6 GPIO锁定机制

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

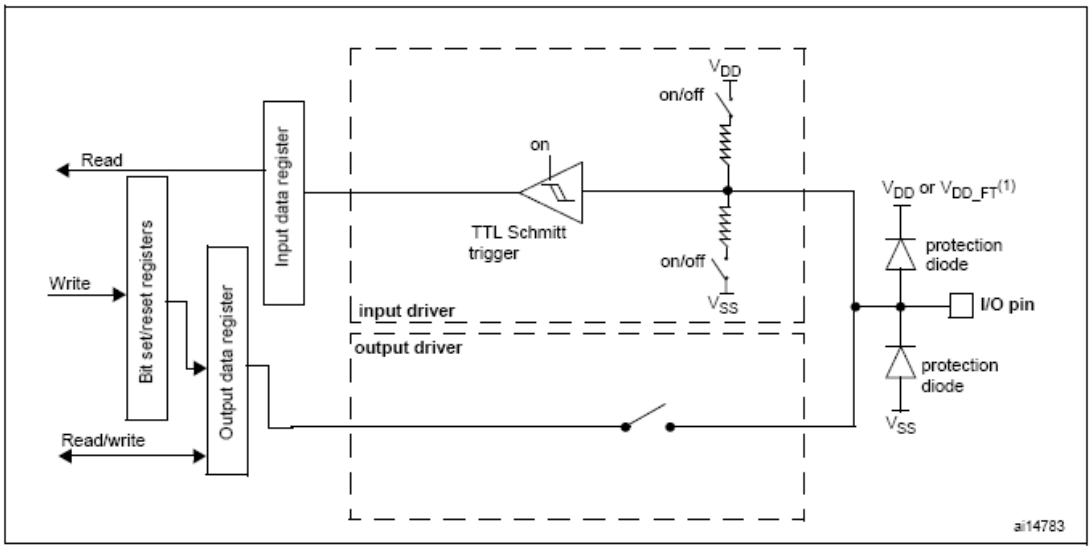
5.1.7 输入配置

当 I/O 端口配置为输入时：

- 输出缓冲器被禁止
- 施密特触发输入被激活
- 根据输入配置(上拉，下拉或浮动)的不同，弱上拉和下拉电阻被连接
- 出现在 I/O 脚上的数据在每个 APB2 时钟被采样到输入数据寄存器
- 对输入数据寄存器的读访问可得到 I/O 状态

图 11 给出了I/O端口位的输入配置

图11 输入浮空/上拉/下拉配置



(1) VDD_FT 对 5 伏兼容 I/O 脚是特殊的，它与 VDD 不同

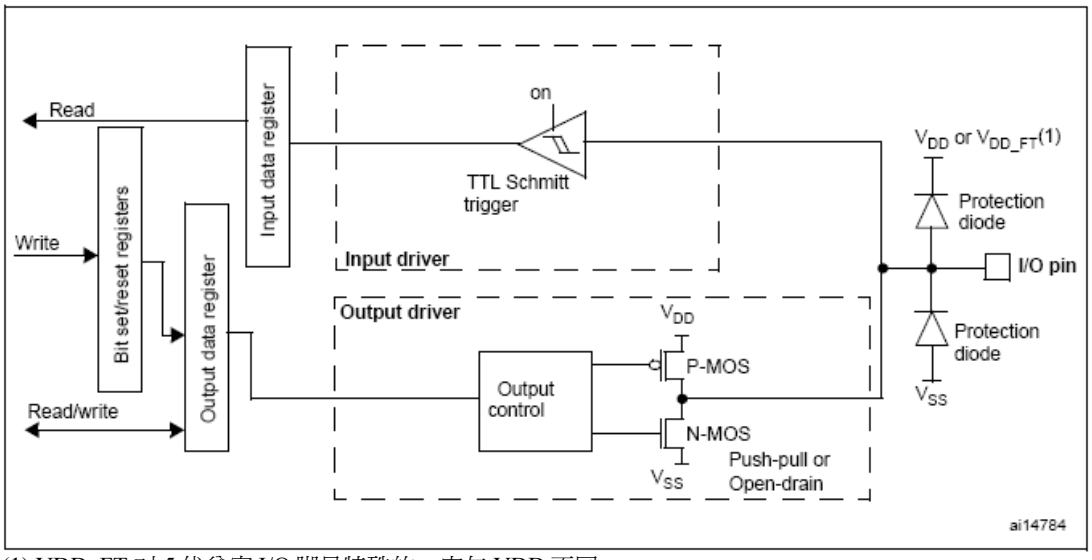
5.1.8 输出配置

当 I/O 端口被配置为输出时：

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

图 12 给出了I/O端口位的输出配置。

图12 输出配置



(1) VDD_FT 对 5 伏兼容 I/O 脚是特殊的，它与 VDD 不同

5.1.9 复用功能配置

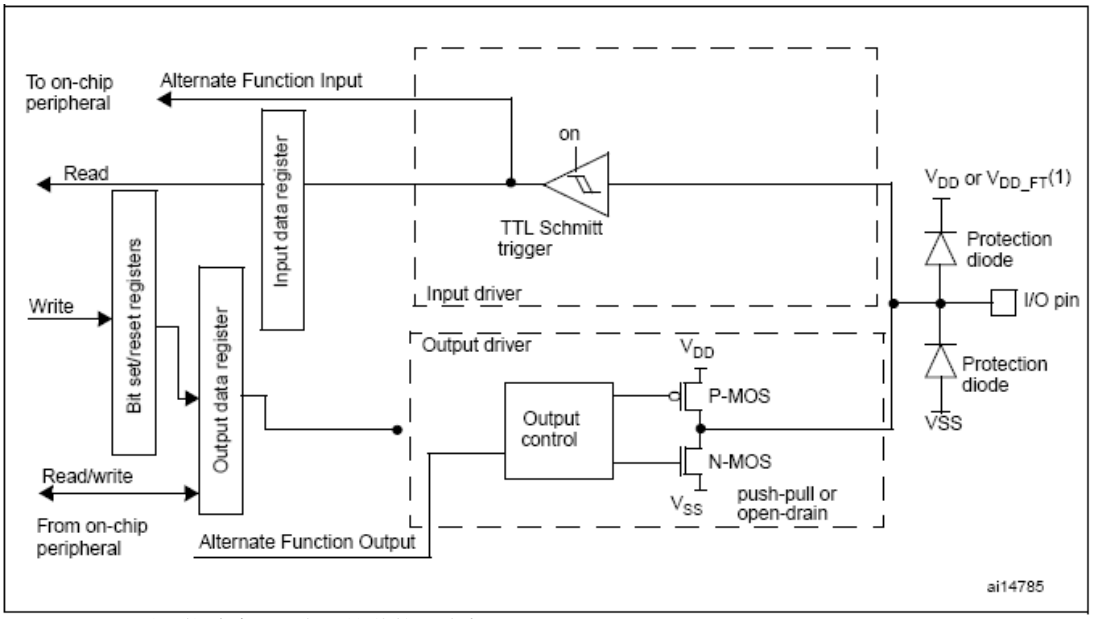
当 I/O 端口被配置为复用功能时：

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

图 13 示出了I/O端口位的复用功能配置。参考 5.4 节 AFIO寄存器描述。

一组复用功能 I/O 寄存器允许用户把一些复用功能重新映象到不同的引脚。

图13 复用功能配置



(1) VDD_FT 对 5 伏兼容 I/O 脚是特殊的，它与 VDD 不同

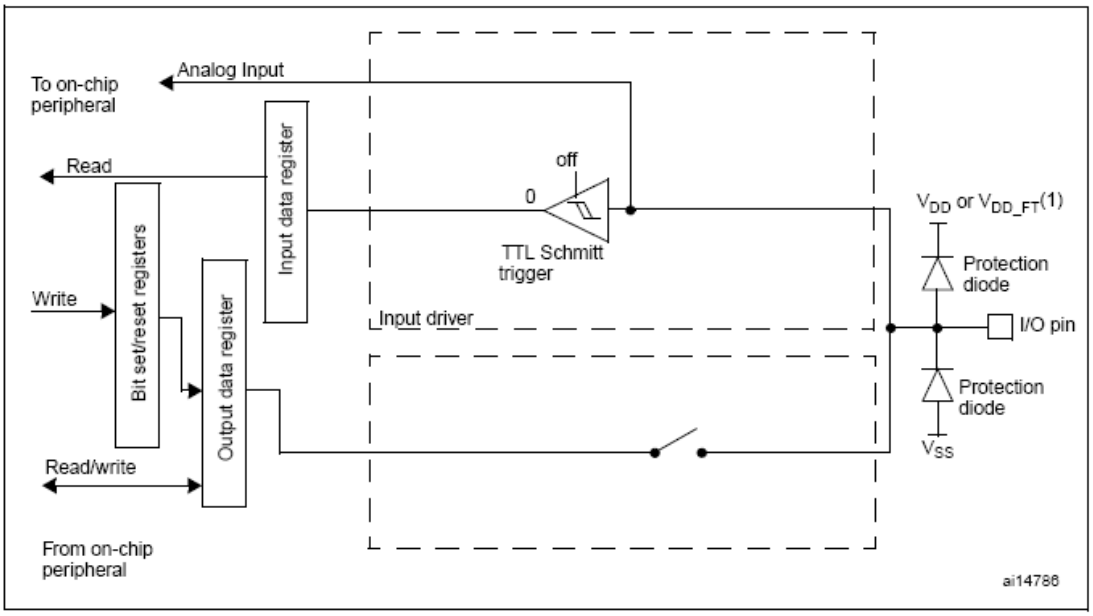
5.1.10 模拟输入配置

当 I/O 端口被配置为模拟输入配置时：

- 输出缓冲器被禁止
- 施密特触发输入被禁止，实现了每个模拟 I/O 引脚上的零消耗。施密特触发输出值被强置为 0
- 弱上拉和下拉电阻被禁止
- 读取输入数据寄存器时值 0

图 14 示出了I/O端口位的高阻抗模拟输入配置

图14 高阻抗的模拟输入配置



(1) VDD_FT 对 5 伏兼容 I/O 脚是特殊的，它与 VDD 不同

5.2 GPIO寄存器描述

请参考第 1 章中有关寄存器描述中用到的缩写。

5.2.1 端口配置低寄存器(GPIOx_CRL) (x=A..E)

偏移地址：0x00
复位值：0x4444 4444

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CNF7[1:0]		MODE7[1:0]		CNF6[1:0]		MODE6[1:0]		CNF5[1:0]		MODE5[1:0]		CNF4[1:0]		MODE4[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNF3[1:0]		MODE3[1:0]		CNF2[1:0]		MODE2[1:0]		CNF1[1:0]		MODE1[1:0]		CNF0[1:0]		MODE0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位	CNFy[1:0]：端口x配置位(y = 0…7) 软件通过这些位配置相应的I/O端口，请参考表11端口位配置表。 在输入模式(MODE[1:0]=00)： 00：模拟输入模式 01：浮空输入模式(复位后的状态) 10：上拉/下拉输入模式 11：保留 在输出模式(MODE[1:0]>00)： 00：通用推挽输出模式 01：通用开漏输出模式 10：复用功能推挽输出模式 11：复用功能开漏输出模式
位 9 : 28 25 : 24 21 : 20 17 : 16 13 : 12 9 : 8 5 : 4 1: 0	MODEy[1:0]：端口x的模式位(y = 0…7) 软件通过这些位配置相应的I/O端口，请参考表11端口位配置表。 00：输入模式(复位后的状态) 01：输出模式，最大速度10MHz 10：输出模式，最大速度2MHz 11：输出模式，最大速度50MHz

5.2.2 端口配置高寄存器(GPIOx_CRH) (x=A..E)

偏移地址：0x04
复位值：0x4444 4444

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CNF15[1:0]		MODE15[1:0]		CNF14[1:0]		MODE14[1:0]		CNF13[1:0]		MODE13[1:0]		CNF12[1:0]		MODE12[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNF11[1:0]		MODE11[1:0]		CNF10[1:0]		MODE10[1:0]		CNF9[1:0]		MODE9[1:0]		CNF8[1:0]		MODE8[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:30 27:26 23:22 19:18 15:14 11:10 7:6 3:2	CNFy[1:0]：端口x配置位(y = 8…15) 软件通过这些位配置相应的I/O端口，请参考表11端口位配置表。 在输入模式(MODE[1:0]=00)： 00：模拟输入模式 01：浮空输入模式(复位后的状态) 10：上拉/下拉输入模式 11：保留 在输出模式(MODE[1:0]>00)： 00：通用推挽输出模式 01：通用开漏输出模式 10：复用功能推挽输出模式 11：复用功能开漏输出模式
位 9:28 25:24 21:20 17:16 13:12 9:8 5:4 1:0	MODEy[1:0]：端口x的模式位(y = 8…15) 软件通过这些位配置相应的I/O端口，请参考表11端口位配置表。 00：输入模式(复位后的状态) 01：输出模式，最大速度10MHz 10：输出模式，最大速度2MHz 11：输出模式，最大速度50MHz

5.2.3 端口输入数据寄存器(GPIOx_IDR) (x=A..E)

地址偏移：0x08
复位值：0x0000 0000

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

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，始终读为0。
位15:0	IDRy[15:0]：端口输入数据(y = 0…15) 这些位为只读并只能以字(16位)的形式读出。读出的值为对应I/O口的状态。

5.2.4 端口输出数据寄存器(GPIOx_ODR) (x=A..E)

地址偏移：0Ch
复位值：00000000h

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

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，始终读为0。
位15:0	ODRy[15:0]：端口输出数据(y = 0…15) 这些位可读可写并只能以字(16位)的形式操作。 注：对GPIOx_BSRR(x = A…E)，可以分别地对各个ODR位进行独立的设置/清除。

5.2.5 端口位设置/复位寄存器(GPIOx_BSRR) (x=A..E)

地址偏移：0x10
复位值：0x0000 0000

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

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

位31:16	BRy: 清除端口x的位y (y = 0…15) 这些位只能写入并只能以字(16位)的形式操作。 0：对对应的ODRy位不产生影响 1：清除对应的ODRy位为0 注：如果同时设置了BSy和BRy的对应位，BSy位起作用。
位15:0	BSy: 设置端口x的位y (y = 0…15) 这些位只能写入并只能以字(16位)的形式操作。 0：对对应的ODRy位不产生影响 1：设置对应的ODRy位为1

5.2.6 端口位复位寄存器(GPIOx_BRR) (x=A..E)

地址偏移：0x14
复位值：0x0000 0000

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

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

位31:16	保留。
位15:0	BRy: 清除端口x的位y (y = 0…15) 这些位只能写入并只能以字(16位)的形式操作。 0：对对应的ODRy位不产生影响 1：清除对应的ODRy位为0

5.2.7 端口配置锁定寄存器(GPIOx_LCKR) (x=A..E)

当执行正确的写序列设置了位 16(LCKK)时，该寄存器用来锁定端口位的配置。位[15:0]用于锁定 GPIO 端口的配置。在规定的写入操作期间，不能改变 LCKP[15:0]。当对相应的端口位执行了 LOCK 序列后，在下次系统复位之前将不能再更改端口位的配置。
每个锁定位锁定控制寄存器(CRL, CRH)中相应的 4 个位。

地址偏移：0x18
复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															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。
位16	LCKK：锁键 该位可随时读出，它只可通过锁键写入序列修改。 0：端口配置锁键位激活 1：端口配置锁键位被激活，下次系统复位前GPIOx_LCKR寄存器被锁住。 锁键的写入序列： 写1 -> 写0 -> 写1 -> 读0 -> 读1 最后一个读可省略，但可以用来确认锁键已被激活。 注：在操作锁键的写入序列时，不能改变LCK[15:0]的值。 操作锁键写入序列中的任何错误将不能激活锁键。
位15:0	LCKy: 端口x的锁位y (y = 0…15) 这些位可读可写但只能在LCKK位为0时写入。 0：不锁定端口的配置 1：锁定端口的配置

5.3 复用功能I/O和调试配置(AFIO)

为了优化 64 脚或 100 脚封装的外设数目，可以把一些复用功能重新映射到其他引脚上。设置 复用重映射和调试I/O配置寄存器(AFIO_MAPR)实现引脚的重新映射。这时，复用功能不再映射到它们的原始分配上。

5.3.1 把OSC32_IN/OSC32_OUT作为GPIO 端口 PC14/PC15

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

注： 1. 当关闭 1.8V 电压区(进入待机模式)或后备区域使用 V_{BAT} 供电(不再有 V_{DD} 供电)时，不能使用 PC14/PC15 的 I/O 口功能；
2．参见第 3.1.2 节有关 I/O 口使用的限制

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

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

这个重映射只适用于 36、48 和 64 脚的封装(100 脚的封装上有单独的 PD0 和 PD1 的引脚，不必重映射)

注： PD0 和 PD1 的输出模式是有限制的，只能使用 50MHz 的输出模式。

5.3.3 BXCAN复用功能重映射

BXCAN信号可以被映射到端口A、端口B或端口E上，如 表 13 所示

表13 BXCAN 复用功能重映射

复用功能	CAN_REMAP[1:0]=”00”	CAN_REMAP[1:0]=”10”(1)	CAN_REMAP[1:0]=”11”
CANRX	PA11	PB8	PD0
CANTX	PA12	PB9	PD1

1. 重映射不适用于36脚的封装

5.3.4 JTAG/SWD复用功能重映射

调试接口信号被映射到GPIO端口上，如 表 14 所示。

表14 调试接口信号

复用功能	GPIO端口
JTMS/SWDIO	PA13
JTCK/SWCLK	PA14
JTDI	PA15
JTDO/TRACESWO	PB3
JNTRST	PB4
TRACECK	PE2
TRACED0	PE3
TRACED1	PE4
TRACED2	PE5
TRACED3	PE6

为了在调试期间可以使用更多GPIOs，通过设置 复用重映射和调试I/O配置寄存器(AFIO_MAPR)的SWJ_CFG[2:0]位，可以改变上述重映像配置。参见 表 15。

表15 调试端口映像

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

1. I/O 口只可在不使用异步跟踪时使用。

5.3.5 定时器复用功能重映射

定时器 4 的通道 1 到通道 4 可以从端口B重映射到端口D。其他定时器的重映射可能性列在 表 17 到 表 19 里。

参见复用重映射和调试 I/O 配置寄存器(AFIO_MAPR)

表16 定时器 4 复用功能重映像

复用功能	TIM4_REMAP = 0	TIM4_REMAP = 1(1)
TIM4_CH1	PB6	PD12
TIM4_CH2	PB7	PD13

TIM4_CH3	PB8	PD14
TIM4_CH4	PB9	PD15

1. 重映像只适用于 64 和 100 脚的封装

表17 定时器 3 复用功能重映像

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

1. 重映像只适用于 64 和 100 脚的封装

表18 定时器 2 复用功能重映像

复用功能	TIM2_REMAP[1:0]= 00 (没有重映像)	TIM2_REMAP[1:0]= 01 (部分重映像)	TIM2_REMAP[1:0]= 10 (部分重映像)(1)	TIM2_REMAP[1:0]= 11 (完全重映像)(1)
TIM2_CH1/ETR	PA0	PA15	PA0	PA15
TIM2_CH2	PA1	PB3	PA1	PB3
TIM2_CH3	PA2		PB10	
TIM2_CH4	PA3		PB11	

1. 重映像不适用于 36 脚的封装

表19 定时器 1 复用功能重映像

复用功能映像	TIM1_REMAP[1:0] = 00 (没有重映像)	TIM1_REMAP[1:0] = 01 (部分重映像)	TIM1_REMAP[1:0] = 11 (完全重映像)(1)
TIM1_ETR	PA12		PE7
TIM1_CH1	PA8		PE9
TIM1_CH2	PA9		PE11
TIM1_CH3	PA10		PE13
TIM1_CH4	PA11		PE14
TIM1_BKIN	PB12(2)	PA6	PE15
TIM1_CH1N	PB13(2)	PA7	PE8
TIM1_CH2N	PB14(2)	PB0	PE10
TIM1_CH3N	PB15(2)	PB1	PE12

1. 重映像只适用于100脚的封装
2. 重映像不适用于36脚的封装

5.3.6 USART复用功能重映射

参见 复用重映射和调试I/O配置寄存器(AFIO_MAPR)

表20 USART3 重映像

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

1. 重映像只适用于64和100脚的封装
2. 重映像只适用于100脚的封装

表21 USART2 重映像

复用功能	USART2_REMAP = 0	USART2_REMAP = 1(1)
USART2_CTS	PA0	PD3
USART2_RTS	PA1	PD4
USART2_TX	PA2	PD5
USART2_RX	PA3	PD6
USART2_CK	PA4	PD7

1. 重映像只适用于 100 脚的封装

表22 USART1 重映像

复用功能	USART1_REMAP = 0	USART1_REMAP = 1
USART1_TX	PA9	PB6
USART1_RX	PA10	PB7

5.3.7 I2C 1 复用功能重映射

参见 复用重映射和调试I/O配置寄存器(AFIO_MAPR)

表23 I2C 1 重映像

复用功能	I2C1_REMAP = 0	I2C1_REMAP = 1(1)
I2C1_SCL	PB6	PB8
I2C1_SDA	PB7	PB9

1. 重映像不适用于 36 脚封装

5.3.8 SPI 1 复用功能重映射

参见 复用重映射和调试I/O配置寄存器(AFIO_MAPR)

表24 SPI1 重映像

复用功能	SPI1_REMAP = 0	SPI1_REMAP = 1

SPI1_NSS	PA4	PA15
SPI1_SCK	PA5	PB3
SPI1_MISO	PA6	PB4
SPI1_MOSI	PA7	PB5

5.4 AFIO寄存器描述

请参考第 1 章中有关寄存器描述中用到的缩写。

5.4.1 事件控制寄存器(AFIO_EVCR)

地址偏移：0x00
复位值：0x0000 0000

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

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

位31:8	保留。
位7	EVOE：允许事件输出 该位可由软件读写。当设置该位后，Cortex的EVENTOUT将连接到由PORT[2:0]和PIN[3:0]选定的I/O口。
位6:4	PORT[2:0]：端口选择 选择用于输出Cortex的EVENTOUT信号的端口： 000：选择PA 001：选择PB 010：选择PC 011：选择PD 100：选择PE
位3:0	PIN[3:0]：管脚选择 选择用于输出Cortex的EVENTOUT信号的管脚： 0000：选择Px0 0001：选择Px1 0010：选择Px2 0011：选择Px3 0100：选择Px4 0101：选择Px5 0110：选择Px6 0111：选择Px7 1000：选择Px8 1001：选择Px9 1010：选择Px10 1011：选择Px11 1100：选择Px12 1101：选择Px13 1110：选择Px14 1111：选择Px15

5.4.2 复用重映射和调试I/O配置寄存器(AFIO_MAPR)

地址偏移：0x04
复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留					SWJ_CFG[2:0]			保留							
					rw	rw	rw								

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

位31:27	保留。
位26:24	SWJ_CFG[2:0]：串行线JTAG配置 这些位可由软件读写，用于配置SWJ和跟踪复用功能的I/O口。SWJ(串行线JTAG)支持JTAG或SWD访问Cortex的调试端口。系统复位后的默认状态是启用SWJ但没有跟踪功能，这种状态下可以通过JTMS/JTCK脚上的特定信号选择JTAG或SW(串行线)模式。 000：完全SWJ(JTAG-DP + SW-DP)：复位状态 001：完全SWJ(JTAG-DP + SW-DP)但没有JNTRST 010：关闭JTAG-DP，启用SW-DP 100：关闭JTAG-DP，关闭SW-DP 其它组合：禁用
位23:16	保留。
位15	PD01_REMAP：端口D0/端口D1映像到OSC_IN/OSC_OUT 该位可由软件读写，它控制PD0和PD1的GPIO功能映像。当不使用主振荡器HSE时(系统运行于内部的8MHz阻容振荡器)PD0和PD1可以映像到OSC_IN和OSC_OUT引脚。此功能只能适用于36、48和64管脚的封装(PD0和PD1出现在TQFP100的封装上，不必重映像)。 0：不进行PD0和PD1的重映像 1：PD0映像到OSC_IN，PD1映像到OSC_OUT。
位14:13	CAN_REMAP[1:0]：CAN复用功能重映像 这些位可由软件读写，控制复用功能CANRX和CANTX的重映像 00：CANRX映像到PA11，CANTX映像到PA12 01：未用组合 10：CANRX映像到PB8，CANTX映像到PB9(不能用于36脚的封装) 11：CANRX映像到PD0，CANTX映像到PD1(只适用于100脚的封装)
位12	TIM4_REMAP：定时器4的重映像 该位可由软件读写，只控制100脚封装中定时器4的通道1至4的映像。 0：没有重映像(TIM4_CH1/PB6，TIM4_CH2/PB7，TIM4_CH3/PB8，TIM4_CH4/PB9) 1：完全映像(TIM4_CH1/PD12，TIM4_CH2/PD13，TIM4_CH3/PD14，TIM4_CH4/PD15) 注：重映像不影响在PE0上的TIM4_ETR。

位11:10	TIM4_REMAP[1:0]：定时器3的重映像 这些位可由软件读写，控制定时器3的通道1至4在GPIO端口的映像。 00：没有重映像(CH1/PA6, CH2/PA7, CH3/PB0, CH4/PB1) 01：未用组合 10：部分映像(CH1/PB4, CH2/PB5, CH3/PB0, CH4/PB1) 11：完全映像(CH1/PC6, CH2/PC7, CH3/PC8, CH4/PC9) 注：重映像不影响在PD2上的TIM3_ETR。
位9:8	TIM2_REMAP[1:0]：定时器2的重映像 这些位可由软件读写，控制定时器2的通道1至4和外部触发(ETR)在GPIO端口的映像。 00：没有重映像(CH1/ETR/PA0, CH2/PA1, CH3/PA2, CH4/PA3) 01：部分映像(CH1/ETR/PA15, CH2/PB3, CH3/PA2, CH4/PA3) 10：部分映像(CH1/ETR/PA0, CH2/PA1, CH3/PB10, CH4/PB11) 11：完全映像(CH1/ETR/PA15, CH2/PB3, CH3/PB10, CH4/PB11)
位7:6	TIM1_REMAP[1:0]：定时器1的重映像 这些位可由软件读写，控制定时器1的通道1至4、1N至3N、外部触发(ETR)和断线输入(BKIN)在GPIO端口的映像。 00：没有重映像(ETR/PA12, CH1/PA8, CH2/PA9, CH3/PA10, CH4/PA11, BKIN/PB12, CH1N/PB13, CH2N/PB14, CH3N/PB15) 01：部分映像(ETR/PA12, CH1/PA8, CH2/PA9, CH3/PA10, CH4/PA11, BKIN/PA6, CH1N/PA7, CH2N/PB0, CH3N/PB1) 10: 未用组合 11: 完全映像(ETR/PE7, CH1/PE9, CH2/PE11, CH3/PE13, CH4/PE14, BKIN/PE15, CH1N/PE8, CH2N/PE10, CH3N/PE12)
位5:4	USART3_REMAP[1:0]：USART3的重映像 这些位可由软件读写，控制USART3的CTS、RTS、CK、TX和RX复用功能在GPIO端口的映像。 00: 没有重映像(TX/PB10, RX/PB11, CK/PB12, CTS/PB13, RTS/PB14) 01: 部分映像(TX/PC10, RX/PC11, CK/PC12, CTS/PB13, RTS/PB14) 10: 未用组合 11: 完全映像(TX/PD8, RX/PD9, CK/PD10, CTS/PD11, RTS/PD12)
位3	USART2_REMAP：USART2的重映像 该位可由软件读写，控制USART2的CTS、RTS、CK、TX和RX复用功能在GPIO端口的映像。 0: 没有重映像(CTS/PA0, RTS/PA1, TX/PA2, RX/PA3, CK/PA4) 1: 重映像(CTS/PD3, RTS/PD4, TX/PD5, RX/PD6, CK/PD7)
位2	USART1_REMAP：USART1的重映像 该位可由软件读写，控制USART1的TX和RX复用功能在GPIO端口的映像。 0: 没有重映像(TX/PA9, RX/PA10) 1: 重映像(TX/PB6, RX/PB7)
位1	I2C1_REMAP：I2C1的重映像 该位可由软件读写，控制I2C1的SCL和SDA复用功能在GPIO端口的映像。 0: 没有重映像(SCL/PB6, SDA/PB7) 1: 重映像(SCL/PB8, SDA/PB9)

位0	SPI1_REMAP：SPI1的重映像 该位可由软件读写，控制SPI1的NSS、SCK、MISO和MOSI复用功能在GPIO端口的映像。 0: 没有重映像(NSS/PA4, SCK/PA5, MISO/PA6, MOSI/PA7) 1: 重映像(NSS/PA15, SCK/PB3, MISO/PB4, MOSI/PB5)

5.4.3 外部中断配置寄存器 1(AFIO_EXTICR1)

地址偏移：0x08
复位值：0x0000

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

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。
位15:0	EXTIx[3:0]：EXTIx配置(x = 0 … 3) 这些位可由软件读写，用于选择EXTIx外部中断的输入源。参看6.2.5节外部中断/事件线映像。 0000：PA[x]脚 0001：PB[x]脚 0010：PC[x]脚 0011：PD[x]脚 0100：PE[x]脚

5.4.4 外部中断配置寄存器 2(AFIO_EXTICR2)

地址偏移：0x0C
复位值：0x0000

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

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。
位15:0	EXTIx[3:0]：EXTIx配置(x = 4 … 7) 这些位可由软件读写，用于选择EXTIx外部中断的输入源。 0000：PA[x]脚 0001：PB[x]脚 0010：PC[x]脚 0011：PD[x]脚 0100：PE[x]脚

5.4.5 外部中断配置寄存器 3(AFIO_EXTICR3)

地址偏移：0x10
复位值：0x0000

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

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。
位15:0	EXTIx[3:0]：EXTIx配置(x = 8 … 11) 这些位可由软件读写，用于选择EXTIx外部中断的输入源。 0000：PA[x]脚 0001：PB[x]脚 0010：PC[x]脚 0011：PD[x]脚 0100：PE[x]脚

5.4.6 外部中断配置寄存器 4(AFIO_EXTICR4)

地址偏移：0x14
复位值：0x0000

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

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。
位15:0	EXTIx[3:0]：EXTIx配置(x = 12 … 15) 这些位可由软件读写，用于选择EXTIx外部中断的输入源。 0000：PA[x]脚 0001：PB[x]脚 0010：PC[x]脚 0011：PD[x]脚 0100：PE[x]脚

5.5 GPIO 和AFIO寄存器地址映象

5.5.1 GPIO寄存器地址映象

表25 GPIO 寄存器地址映像和复位值

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
000h	GPIOx_CRL	CNF7[1:0]		MODE7[1:0]		CNF6[1:0]		MODE6[1:0]		CNF5[1:0]		MODE5[1:0]		CNF4[1:0]		MODE4[1:0]		CNF3[1:0]		MODE3[1:0]		CNF2[1:0]		MODE2[1:0]		CNF1[1:0]		MODE1[1:0]		CNF0[1:0]		MODE0[1:0]	
	复位值	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
004h	GPIOx_CRH	CNF15[1:0]		MODE15[1:0]		CNF14[1:0]		MODE14[1:0]		CNF13[1:0]		MODE13[1:0]		CNF12[1:0]		MODE12[1:0]		CNF11[1:0]		MODE11[1:0]		CNF10[1:0]		MODE10[1:0]		CNF9[1:0]		MODE9[1:0]		CNF8[1:0]		MODE8[1:0]	
	复位值	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
008h	GPIOx_IDR	保留																IDR[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
00Ch	GPIOx_ODR	保留																ODR[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
010h	GPIOx_BSRR	BR[15:0]																BSR[15:0]															
	复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
014h	GPIOx_BRR	保留																BR[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
018h	GPIOx_LCKR	保留															LCKK	LCK[15:0]															
	复位值																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

5.5.2 AFIO寄存器地址映象

表26 AFIO 寄存器地址映像和复位值

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
000h	AFIO_EVCR	保留																								EVOE	PORT[2:0]			PIN[3:0]			
	复位值																									0	0	0	0	0	0	0	0
004h	AFIO_MAPR	保留					SWJ_CFG[2:0]			保留								PD01_REMAP	CAN_REMAP[1:0]		TIM4_REMAP	TIM3_REMAP[1:0]		TIM2_REMAP[1:0]		TIM1_REMAP[1:0]		USART3_REMAP[1:0]		USART2_REMAP	USART1_REMAP	I2C1_REMAP	SPI1_REMAP
	复位值						0	0	0									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
008h	AFIO_EXTICR1	保留																EXTI3[3:0]				EXTI2[3:0]				EXTI1[3:0]				EXTI0[3:0]			
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
00Ch	AFIO_EXTICR2	保留																EXTI7[3:0]				EXTI6[3:0]				EXTI5[3:0]				EXTI4[3:0]			
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
010h	AFIO_EXTICR3	保留																EXTI11[3:0]				EXTI10[3:0]				EXTI9[3:0]				EXTI8[3:0]			
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
014h	AFIO_EXTICR4	保留																EXTI15[3:0]				EXTI14[3:0]				EXTI13[3:0]				EXTI12[3:0]			
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

6 中断和事件

6.1 嵌套向量中断控制器

特性

- 43 个可屏蔽中断通道（不包含 16 个 Cortex™-M3 的中断线）；
- 16 个可编程的优先等级(使用了 4 位中断优先级)；
- 低延迟的异常和中断处理；
- 电源管理控制；
- 系统控制寄存器的实现；

嵌套向量中断控制器(NVIC)和处理器核的接口紧密相连，可以实现低延迟的中断处理和有效处理地处理晚到的中断。

嵌套向量中断控制器管理着包括核异常等中断。关于更多的异常和 NVIC 编程的说明请参考 ARM《Cortex™-M3 技术参考手册》的第 5 章的异常和第 8 章的嵌套向量中断控制器。

6.1.1 系统嘀嗒(SysTick)校准值寄存器

系统嘀嗒校准值固定到 9000，当系统嘀嗒时钟设定为 9 兆赫，产生 1ms 时基。

6.1.2 中断和异常向量

表27 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)联接到NMI向量	0x0000_0008
	-1	固定	硬件失效	所有类型的失效	0x0000_000C
	0	可设置	存储管理	存储器管理	0x0000_0010
	1	可设置	总线错误	预取指失败，存储器访问失败	0x0000_0014
	2	可设置	错误应用	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指令的系统服务调用	0x0000_002C
	4	可设置	调试监控	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	窗口定时器中断	0x0000_0040
1	8	可设置	PVD	联到EXTI的电源电压检测(PVD)中断	0x0000_0044
2	9	可设置	TAMPER	侵入检测中断	0x0000_0048
3	10	可设置	RTC	实时时钟(RTC)全局中断	0x0000_004C
4	11	可设置	FLASH	闪存全局中断	0x0000_0050
5	12	可设置	RCC	复位和时钟控制(RCC)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	DMA通道1	DMA通道1全局中断	0x0000_006C
12	19	可设置	DMA通道2	DMA通道2全局中断	0x0000_0070
13	20	可设置	DMA通道3	DMA通道3全局中断	0x0000_0074
14	21	可设置	DMA通道4	DMA通道4全局中断	0x0000_0078
15	22	可设置	DMA通道5	DMA通道5全局中断	0x0000_007C
16	23	可设置	DMA通道6	DMA通道6全局中断	0x0000_0080
17	24	可设置	DMA通道7	DMA通道7全局中断	0x0000_0084
18	25	可设置	ADC	ADC全局中断	0x0000_0088
19	26	可设置	USB_HP_CAN_TX	USB高优先级或CAN发送中断	0x0000_008C
20	27	可设置	USB_LP_CAN_RX0	USB低优先级或CAN接收0中断	0x0000_0090
21	28	可设置	CAN_RX1	CAN接收1中断	0x0000_0094
22	29	可设置	CAN_SCE	CAN 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	TIM1断开中断	0x0000_00A0
25	32	可设置	TIM1_UP	TIM1更新中断	0x0000_00A4
26	33	可设置	TIM1_TRG_COM	TIM1触发和通信中断	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	RTCAlarm	联到EXTI的RTC闹钟中断	0x0000_00E4
42	49	可设置	USB唤醒	联到EXTI的从USB待机唤醒中断	0x0000_00E8

6.2 外部中断/事件控制器(EXTI)

外部中断/事件控制器由 19 个产生事件/中断要求的边沿检测器组成。每个输入线可以独立地配置输入类型（脉冲或挂起）和对应的触发事件（上升沿或下降沿或者双边沿都触发）。每个输入线都可以被独立的屏蔽。挂起寄存器保持着状态线的中断要求。

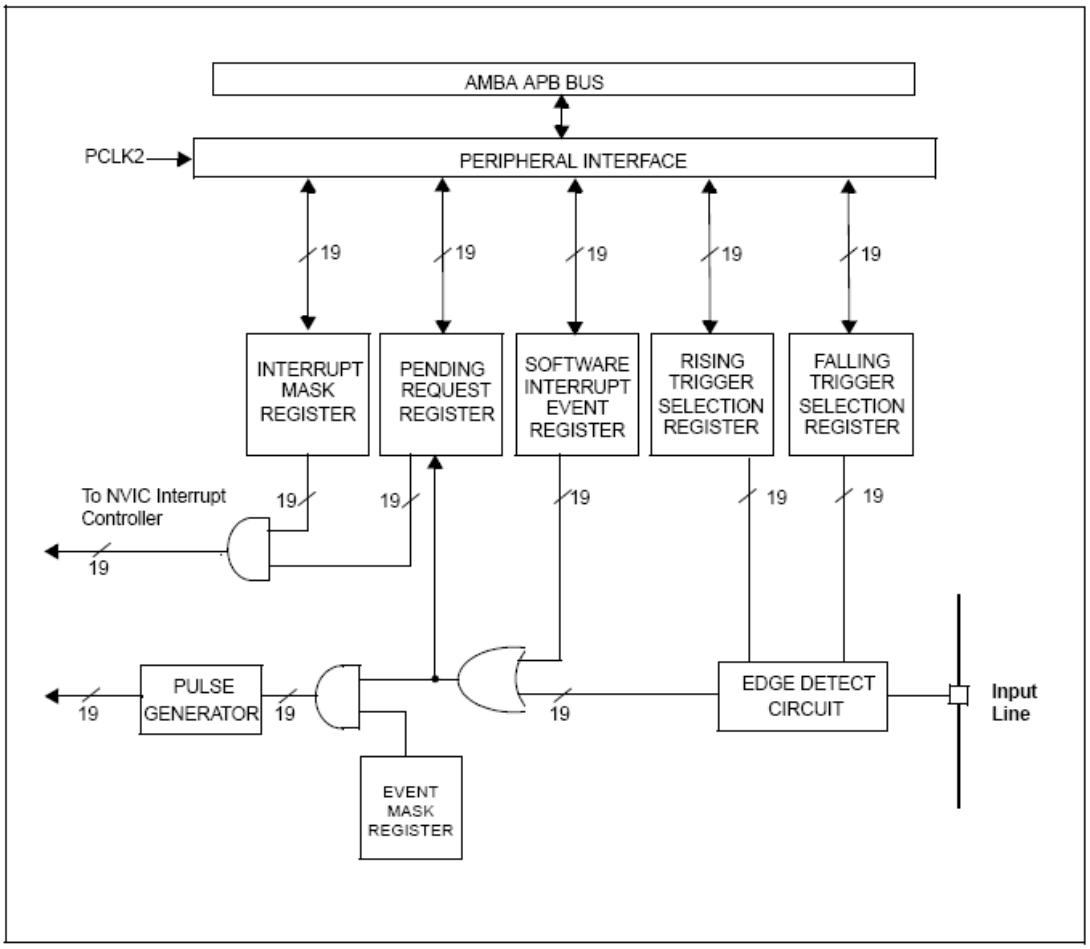
6.2.1 主要特性

EXTI 控制器的主要特性如下：

- 每个中断/事件都有独立的触发和屏蔽
- 每个中断线都有专用的状态位
- 支持多达 19 个中断/事件请求
- 检测脉冲宽度低于 APB2 时种宽度的外部信号。参见数据手册中电气特性部分的相关参数。

6.2.2 框图

图15 外部中断/事件控制器框图



6.2.3 唤醒事件管理

Cortex™-M3 可以处理外部时间或内部中断来唤醒内核(WFE)。通过配置任何一个外部 I/O 端口、RTC 闹钟和 USB 唤醒事件可以唤醒 CPU（内核从 WFE 退出）。

使用外部I/O端口作为唤醒事件，请参见 6.2.4 节的 功能说明

6.2.4 功能说明

如要产生中断，中断线必须事先配置好并被激活。这是根据需要的边沿检测通过设置 2 个触发寄存器，和在中断屏蔽寄存器的相应位写“1”到来允许中断请求。当需要的边沿在外部中断线上发生时，将产生一个中断请求，对应的挂起位也随之被置 1。通过写“1”到挂起寄存器，可以清除该中断请求。

为产生事件触发，事件连接线必须事先配置好并被激活。这是根据需要的边沿检测通过设置 2 个触发寄存器，和在事件屏蔽寄存器的相应位写“1”到来允许事件请求。当需要的边沿在事件连线上发生时，将产生一个事件请求脉冲，对应的挂起位不被置 1。

通过在软件中断/事件寄存器写“1”，一个中断/事件请求也可以通过软件来产生。

硬件中断选择

通过下面的过程来配置 19 个线路做为中断源：

- 配置 19 个中断线的屏蔽位（EXTI_IMR）
- 配置所选中断线的触发选择位（EXTI_RTSR 和 EXTI_FTSR）；
- 配置那些控制映像到外部中断控制器(EXTI)的 NVIC 中断通道的使能和屏蔽位，使得 19 个中断线中的请求可以被正确地响应。

硬件事件选择

通过下面的过程，可以配置 19 个线路为事件源

- 配置 19 个事件线的屏蔽位（EXTI_EMR）
- 配置事件线的触发选择位（EXTI_RTSR and EXTI_FTSR）

软件中断/事件的选择

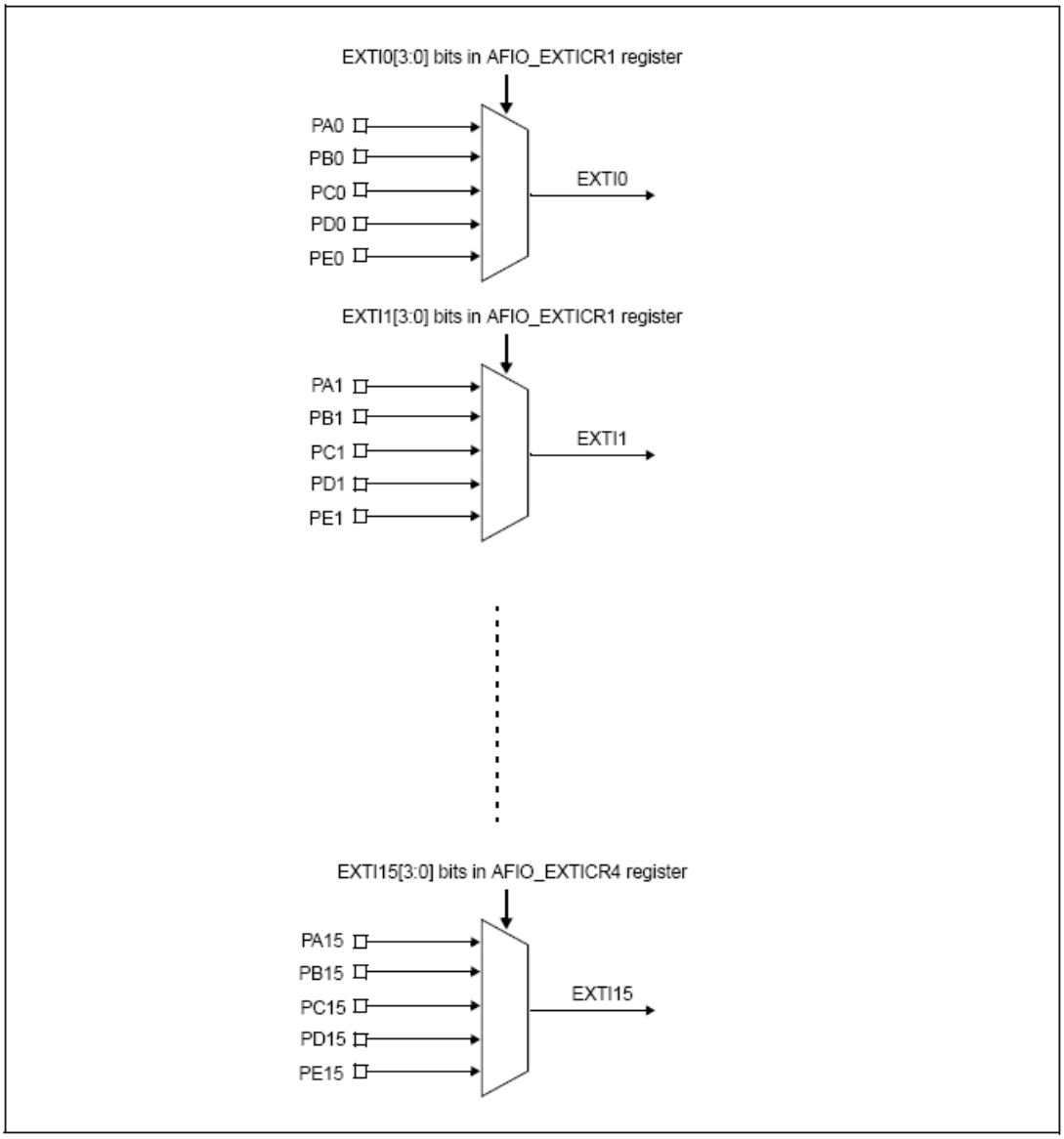
19 个线路可以被配置成软件中断/事件线。下面是产生软件中断的过程：

- 配置 19 个中断/事件线屏蔽位（EXTI_IMR, EXTI_EMR）
- 设置软件中断寄存器的请求位（EXTI_SWIER）

6.2.5 外部中断/事件线路映像

80 通用 I/O 端口以下图的方式连接到 16 个外部中断/事件线上：

图16 外部中断通用 I/O 映像



另外三种其他的外部中断/事件控制器的连接如下：

- 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 唤醒事件

6.3 EXTI 寄存器描述

关于寄存器描述中的缩略词，请参考 1.1 节。

中断屏蔽寄存器（EXTI_IMR）

偏移地址：0x00
复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留													MR18	MR17	MR16
													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	MR4	MR4	MR3	MR2	MR1	MR0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位31:19	保留，必须始终保持为复位状态(0)。
位18:0	MRx: 线x上的中断屏蔽 0：线x上的中断请求被屏蔽 1：线x上的中断请求不被屏蔽

事件屏蔽寄存器（EXTI_EMR）

偏移地址：0x04
复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留													MR18	MR17	MR16
													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	MR4	MR4	MR3	MR2	MR1	MR0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位31:19	保留，必须始终保持为复位状态(0)。
位18:0	MRx: 线x上的事件屏蔽 0：线x上的事件请求被屏蔽

	1：线x上的事件请求不被屏蔽

上升沿触发选择寄存器（EXTI_RTSR）

偏移地址：0x08
复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留													TR18	TR17	TR16
													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31:19	保留，必须始终保持为复位状态(0)。
位18:0	TRx: 线x上的上升沿触发事件配置位 0：禁止输入线x上的上升沿触发(中断和事件) 1：允许输入线x上的上升沿触发(中断和事件)

注意: 外部唤醒线是边沿触发的，这些线上不能出现毛刺信号。
在写 EXTI_RTSR 寄存器时在外部中断线上的上升沿信号不能被识别，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。

下降沿触发选择寄存器（EXTI_FTSR）

偏移地址：0x0C
复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留													TR18	TR17	TR16
													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31:19	保留，必须始终保持为复位状态(0)。
位18:0	TRx: 线x上的下降沿触发事件配置位

	0：禁止输入线x上的下降沿触发(中断和事件) 1：允许输入线x上的下降沿触发(中断和事件)

注意: 外部唤醒线是边沿触发的，这些线上不能出现毛刺信号。
在写 EXTI_RTSR 寄存器时在外部中断线上的下降沿信号不能被识别，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。

软件中断事件寄存器（EXTI_SWIER）

偏移地址：0x10
复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留													SWIER18	SWIER17	SWIER16
													rw	rw	rw

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

位31:19	保留，必须始终保持为复位状态(0)。
位18:0	SWIERx: 线x上的软件中断 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，则此时将产生一个中断。 通过清除EXTI_PR的对应位(写入1)，可以清除该位为0。

挂起寄存器（EXTI_PR）

偏移地址：0x14
复位值：0xxxxx xxxx

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留													PR18	PR17	PR16
													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31:19	保留，必须始终保持为复位状态(0)。

位18:0	PRx: 挂起位 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。在该位中写入1可以清除它，也可以通过改变边沿检测的极性清除。 注：如果在进入停机模式前的一个周期发生了一个中断，则EXTI_PR寄存器将只在系统从停机模式退出后才被修改，并在EXTI_IMR寄存器中未屏蔽该中断时产生中断请求。

6.3.1 外部中断/事件寄存器映像

表28 外部中断/事件控制器寄存器映像和复位值

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
000h	EXTI_IMR	保留												MR[18:0]																			
	复位值													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
004h	EXTI_EMR	保留												MR[18:0]																			
	复位值													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
008h	EXTI_RTSR	保留												TR[18:0]																			
	复位值													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
00Ch	EXTI_FTSR	保留												TR[18:0]																			
	复位值													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
010h	EXTI_SWIER	保留												SWIER[18:0]																			
	复位值													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
014h	EXTI_PR	保留												PR[18:0]																			
	复位值													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

7 DMA 控制器（DMA）

7.1 简介

直接存储器存取用来提供在外设和存储器之间或者存储器和存储器之间的高速数据传输。无须 CPU 任何干预，通过 DMA 数据可以快速地移动。这就节省了 CPU 的资源来做其他操作。

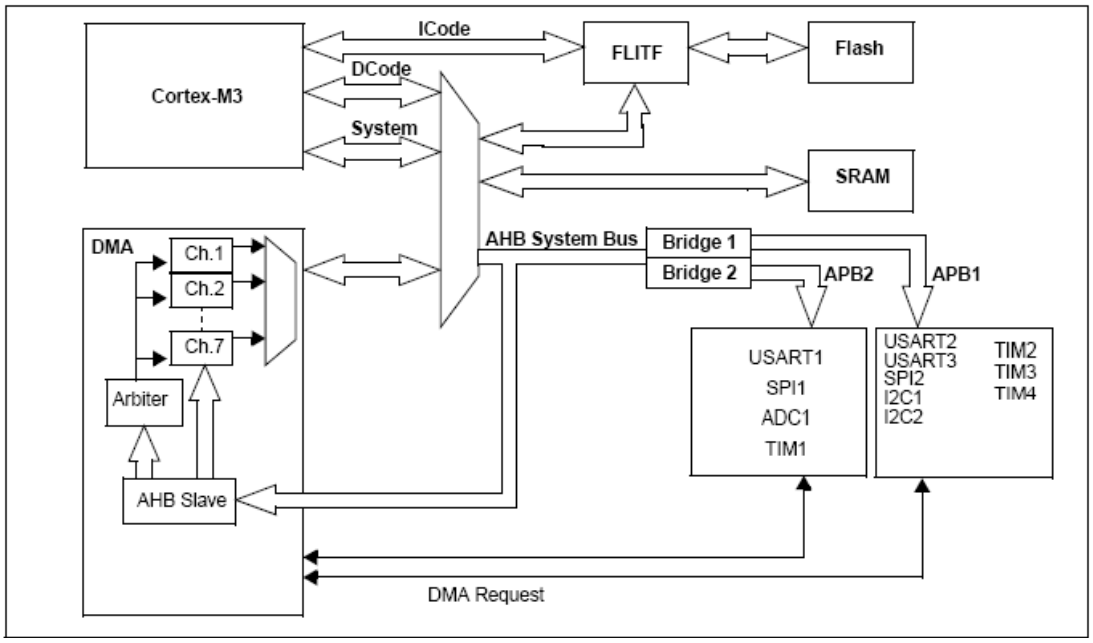
DMA 控制器有 7 个通道，每个通道专门用来管理来自于一个或多个外设对存储器访问的请求。还有一个仲裁器来协调各个 DMA 请求的优先权。

7.2 主要特性

- 7 个 独立的可配置的通道（请求）
- 每个通道都直接连接专用的硬件 DMA 请求，每个通道都同样支持软件触发。这些功能通过软件来配置。
- 在七个请求间的优先权可以通过软件编程设置(共有四级：很高、高、中等和低)，假如在相等优先权时由硬件决定(请求 0 优先于请求 1，依此类推) 。
- 独立的源和目标数据区的传输宽度(字节、半字、全字)，模拟打包和拆包的过程。
- 支持循环的缓冲器管理
- 每个通道都有 3 个事件标志(DMA 半传输，DMA 传输完成和 DMA 传输出错)，这 3 个事件标志逻辑或成为一个单独的中断请求。
- 存储器和存储器间的传输
- 外设和存储器，存储器和外设的传输
- 闪存、SRAM、外设的 SRAM、APB1 和 APB2 外设均可作为访问的源和目标。
- 可编程的数据传输数目：最大为 65536

下面为功能框图：

图17 DMA 框图



7.3 功能描述

DMA 控制器和 Cortex™-M3 核共享系统数据总线执行直接存储器数据传输。当 CPU 和 DMA 同时访问相同的目标(RAM 或外设)时，DMA 请求可能会停止 CPU 访问系统总线达若干个周期，总线仲裁器执行循环调度，以保证 CPU 至少可以得到一半的系统总线(RAM 或外设)带宽。

7.3.1 DMA处理

在发生一个事件后，外设发送一个请求信号到 DMA 控制器。DMA 控制器根据通道的优先权处理请求。当 DMA 控制器开始访问外设的时候，DMA 控制器立即发送给外设一个应答信号。当从 DMA 控制器得到应答信号时，外设立即释放它的请求。一旦外设释放了这个请求，DMA 控制器同时撤销应答信号。如果发生更多的请求时，外设可以启动下次处理。

总之，每个 DMA 传送由 3 个操作组成：

- 从外设数据寄存器或者从 DMA_CMARx 寄存器指定地址的存储器单元执行加载操作。
- 存数据到外设数据寄存器或者存数据到 DMA_CMARx 寄存器指定地址的存储器单元。
- 执行一次 DMA_CNDTRx 寄存器的递减操作。该寄存器包含未完成的操作数目。

7.3.2 仲裁器

仲裁器根据通道请求的优先级来启动外设/存储器的访问。

优先权管理分 2 个阶段：

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

7.3.3 DMA 通道

每个通道都可以在有固定地址的外设寄存器和存储器地址之间执行 DMA 传输。DMA 传输的数据量是可编程的，最大达到 65535。包含要传输的数据项数量的寄存器，在每次传输后递减。

可编程的数据量

外设和存储器的传输数据量可以通过 DMA_CCRx 寄存器中的 PSIZE 和 MSIZE 位编程。

指针增量

通过设置 DMA_CCRx 寄存器中 PINC 和 MINC 标志位，外设和存储器的指针在每次传输后可以有选择地完成自动增量。当设置为增量模式时，下一个要传输的地址将是前一个地址加上增量值，增量值取决与所选的数据宽度为 1、2 或 4。第一个传输的地址存放在 DMA_CPARx/DMA_CMARx 寄存器中。

通道配置为非循环模式时，在传输结束后(即传输数据量变为 0)将不再产生 DMA 操作。

通道配置过程

下面是配置 DMA 通道 x 的过程(x 代表通道号)：

1. 在DMA_CPARx寄存器中设置外设寄存器的地址。发生外设数据传输请求时，这个地址将是数据传输的源或目标。
2. 在DMA_CMARx寄存器中设置数据存储器的地址。发生外设数据传输请求时，传输的数据将从这个地址读出或写入这个地址。

3. 在DMA_CNDTRx寄存器中设置要传输的数据量。在每个数据传输后，这个数值递减。
4. 在DMA_CCRx寄存器的PL[1:0]位中设置通道的优先级。
5. 在DMA_CCRx寄存器中设置数据传输的方向、循环模式、外设和存储器的增量模式、外设和存储器的数据宽度、传输一半产生中断或传输完成产生中断。
6. 设置DMA_CCRx寄存器的ENABLE位，启动该通道。

一旦启动了 DMA 通道，它既可响应联到该通道上的外设的 DMA 请求。

当传输一半的数据后，半传输标志(HTIF)被置 1，当设置了允许半传输中断位(HTIE)时，将产生一个中断请求。在数据传输结束后，传输完成标志(TCIF)被置 1，当设置了允许传输完成中断位(TCIE)时，将产生一个中断请求。

循环模式

循环模式用于处理循环缓冲区和连续的数据传输(如 ADC 的扫描模式)。在 DMA_CCRx 寄存器中的 CIRC 位用于开启这一功能。当启动了循环模式，数据传输的数目变为 0 时，将会自动地被恢复成配置通道时设置的初值，DMA 操作将会继续进行。

存储器到存储器模式

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

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

7.3.4 错误管理

在 DMA 读写操作时一旦发生总线错误，硬件会自动地清除发生错误的通道所对应的通道配置寄存器(DMA_CCRx)的 EN 位，该通道操作被停止。此时，在 DMA_IFT 寄存器中对应该通道的传输错误中断标志位(TEIF)将被置位，如果在 DMA_CCRx 寄存器中设置了传输错误中断允许位，则将产生中断。

7.3.5 DMA请求映像

从外设(TIMx、ADC、SPIx、I^2Cx 和 USARTx)产生的 7 个请求，通过逻辑或输入到 DMA 控制器，这意味着同时只能有一个请求有效。参见下图的 DMA 请求映像。

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

图18 DMA 请求映像

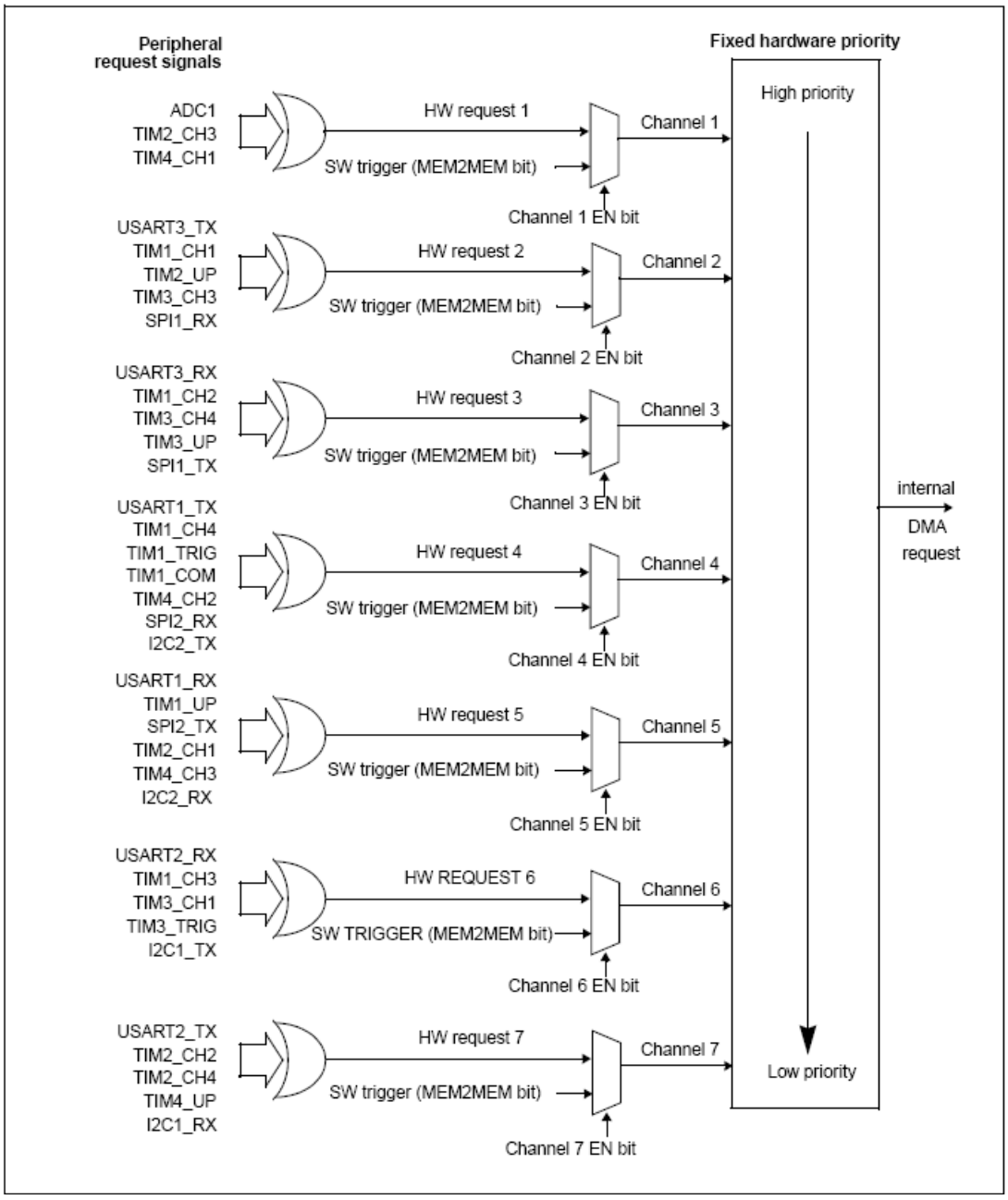

Peripheral request signals
Fixed hardware priority
High priority
ADC1
TIM2_CH3
TIM4_CH1
HW request 1
SW trigger (MEM2MEM bit)
Channel 1
Channel 1 EN bit
USART3_TX
TIM1_CH1
TIM2_UP
TIM3_CH3
SPI1_RX
HW request 2
SW trigger (MEM2MEM bit)
Channel 2
Channel 2 EN bit
USART3_RX
TIM1_CH2
TIM3_CH4
TIM3_UP
SPI1_TX
HW request 3
SW trigger (MEM2MEM bit)
Channel 3
Channel 3 EN bit
USART1_TX
TIM1_CH4
TIM1_TRIG
TIM1_COM
TIM4_CH2
SPI2_RX
I2C2_TX
HW request 4
SW trigger (MEM2MEM bit)
Channel 4
Channel 4 EN bit
USART1_RX
TIM1_UP
SPI2_TX
TIM2_CH1
TIM4_CH3
I2C2_RX
HW request 5
SW trigger (MEM2MEM bit)
Channel 5
Channel 5 EN bit
USART2_RX
TIM1_CH3
TIM3_CH1
TIM3_TRIG
I2C1_TX
HW REQUEST 6
SW TRIGGER (MEM2MEM bit)
Channel 6
Channel 6 EN bit
USART2_TX
TIM2_CH2
TIM2_CH4
TIM4_UP
I2C1_RX
HW request 7
SW trigger (MEM2MEM bit)
Channel 7
Channel 7 EN bit
Low priority
internal
DMA
request

表29 各个通道的 DMA 请求一览

外设	通道1	通道2	通道3	通道4	通道5	通道6	通道7
ADC	ADC1						
SPI		SPI1_RX	SPI1_TX	SPI2_RX	SPI2_TX		
USART		USART3_TX	USART3_RX	USART1_TX	USART1_RX	USART2_RX	USART2_TX
I^2C				I2C2_TX	I2C2_RX	I2C1_TX	I2C1_RX
TIM1		TIM1_CH1	TIM1_CH2	TIM1_TX4 TIM1_TRIG TIM1_COM	TIM1_UP	TIM1_CH3	
TIM2	TIM2_CH3	TIM2_UP			TIM2_CH1		TIM2_CH2 TIM2_CH4
TIM3		TIM3_CH3	TIM3_CH4 TIM3_UP			TIM3_CH1 TIM3_TRIG	
TIM4	TIM4_CH1			TIM4_CH2	TIM4_CH3		TIM4_UP

7.4 DMA寄存器

关于寄存器描述中用到的缩写，请参见第 1 章。

7.4.1 DMA中断状态寄存器(DMA_ISR)

偏移地址：0x00
复位值：0x0000 0000

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

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

位31:28	保留，始终读为0。
位 27， 23，19， 15，11，7，3	TEIFx：通道x的传输错误标志(x = 1 … 7) 硬件设置这些位。在DMA_IFCR寄存器的相应位写入1可以清除这里对应的标志位。 0：在通道x没有传输错误(TE) 1：在通道x发生传输错误(TE)
位 26， 22，18， 14，10，6，2	HTIFx：通道x的半传输标志(x = 1 … 7) 硬件设置这些位。在DMA_IFCR寄存器的相应位写入1可以清除这里对应的标志位。 0：在通道x没有半传输事件(HT) 0：在通道x产生半传输事件(HT)
位 25， 21，17， 13，9，5，1	TCIFx：通道x的传输完成标志(x = 1 … 7) 硬件设置这些位。在DMA_IFCR寄存器的相应位写入1可以清除这里对应的标志位。 0：在通道x没有传输完成事件(TC) 0：在通道x产生传输完成事件(TC)
位 24， 20，16， 12，8，4，0	GIFx：通道x的全局中断标志(x = 1 … 7) 硬件设置这些位。在DMA_IFCR寄存器的相应位写入1可以清除这里对应的标志位。 0：在通道x没有TE、HT或TC事件 0：在通道x产生TE、HT或TC事件

7.4.2 DMA中断标志清除寄存器(DMA_IFCR)

偏移地址：0x04
复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留				CTEIF7	CHTIF7	CTCIF7	CGIF7	CTEIF6	CHTIF6	CTCIF6	CGIF6	CTEIF5	CHTIF5	CTCIF5	CGIF5
				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
CTEIF4	CHTIF4	CTCIF4	CGIF4	CTEIF3	CHTIF3	CTCIF3	CGIF3	CTEIF2	CHTIF2	CTCIF2	CGIF2	CTEIF1	CHTIF1	CTCIF1	CGIF1
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位31:28	保留，始终读为0。
位 27， 23，19， 15，11，7，3	CTEIFx：清除通道x的传输错误标志(x = 1 … 7) 这些位由软件设置和清除。 0：不起作用 1：清除DMA_ISR寄存器中的对应TEIF标志。
位 26， 22，18， 14，10，6，2	CHTIFx：清除通道x的半传输标志(x = 1 … 7) 这些位由软件设置和清除。 0：不起作用 0：清除DMA_ISR寄存器中的对应HTIF标志。
位 25， 21，17， 13，9，5，1	CTCIFx：清除通道x的传输完成标志(x = 1 … 7) 这些位由软件设置和清除。 0：不起作用 0：清除DMA_ISR寄存器中的对应TCIF标志。
位 24， 20，16，12，8，4，0	CGIFx：清除通道x的全局中断标志(x = 1 … 7) 这些位由软件设置和清除。 0：不起作用 0：清除DMA_ISR寄存器中的对应的GIF、TEIF、HTIF和TCIF标志。

7.4.3 DMA通道x配置寄存器(DMA_CCRx)(x = 1…7)

偏移地址：0x08 + 20d x 通道编号
复位值：0x0000 0000

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

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

位31:15	保留，始终读为0。
位14	MEM2MEM：存储器到存储器模式 该位由软件设置和清除。 0：非存储器到存储器模式 1：启动存储器到存储器模式
位13:12	PL[1:0]：通道优先级 这些位由软件设置和清除。 00：低 01：中 10：高 11：最高
位11:10	MSIZE[1:0]：存储器数据宽度 这些位由软件设置和清除。 00：8位 01：16位 10：32位 11：保留
位9:8	PSIZE[1:0]：外设数据宽度 这些位由软件设置和清除。 00：8位 01：16位 10：32位 11：保留
位7	MINC：存储器地址增量模式 该位由软件设置和清除。 0：不执行存储器地址增量操作 1：执行存储器地址增量操作
位6	PINC：外设地址增量模式 该位由软件设置和清除。 0：不执行外设地址增量操作 1：执行外设地址增量操作

位5	CIRC：循环模式 该位由软件设置和清除。 0：不执行循环操作 1：执行循环操作
位4	DIR：数据传输方向 该位由软件设置和清除。 0：从外设读 1：从存储器读
位3	TEIE：允许传输错误中断 该位由软件设置和清除。 0：禁止TE中断 0：允许TE中断
位2	HTIE：允许半传输中断 该位由软件设置和清除。 0：禁止HT中断 0：允许HT中断

位1	TCIE：允许传输完成中断 该位由软件设置和清除。 0：禁止TC中断 0：允许TC中断
位0	EN：通道开启 该位由软件设置和清除。 0：通道不工作 1：通道开启

7.4.4 DMA通道x传输数量寄存器(DMA_CNDTRx)(x = 1…7)

偏移地址：0x0C + 20d x 通道编号
复位值：0x0000 0000

位31:16	保留，始终读为0。
位15:0	NDT[15:0]：数据传输数量 数据传输数量为0至65535。这个寄存器只能在通道不工作(DMA_CCRx的EN=0)时写入。通道开启后该寄存器变为只读，指示剩余的待传输的字节数目。寄存器内容在每次DMA传输后递减。 数据传输结束后，寄存器的内容或者变为0；或者当该通道配置为自动重加载模式时，寄存器的内容将被自动重新加载为之前配置时的数值。 当寄存器的内容为0时，无论通道是否开启，都不会发生任何数据传输。

7.4.5 DMA通道x外设地址寄存器(DMA_CPARx)(x = 1…7)

偏移地址：0x10 + 20d x 通道编号
复位值：0x0000 0000

位31:0	PA[31:0]：外设地址 外设数据寄存器的基地址，作为数据传输的源或目标。

7.4.6 DMA通道x存储器地址寄存器(DMA_CPARx)(x = 1…7)

偏移地址：0x14 + 20d x 通道编号
复位值：0x0000 0000

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

7.5 DMA寄存器映像

表30 DMA 寄存器映像和复位

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
000h	DMA_ISR	保留				TEIF7	HTIF7	TCIF7	GIF7	TEIF6	HTIF6	TCIF6	GIF6	TEIF5	HTIF5	TCIF5	GIF5	TEIF4	HTIF4	TCIF4	GIF4	TEIF3	HTIF3	TCIF3	GIF3	TEIF2	HTIF2	TCIF2	GIF2	TEIF1	HTIF1	TCIF1	GIF1
	复位值					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
004h	DMA_IFCR	保留				CTEIF7	CHTIF7	CTCIF7	CGIF7	CTEIF6	CHTIF6	CTCIF6	CGIF6	CTEIF5	CHTIF5	CTCIF5	CGIF5	CTEIF4	CHTIF4	CTCIF4	CGIF4	CTEIF3	CHTIF3	CTCIF3	CGIF3	CTEIF2	CHTIF2	CTCIF2	CGIF2	CTEIF1	CHTIF1	CTCIF1	CGIF1
	复位值					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
008h	DMA_CCR1	保留																	MEM2MEM	PL [1:0]		MSIZE [1:0]		PSIZE [1:0]		MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN
	复位值																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
00Ch	DMA_CNDTR1	保留																	NDT[15:0]														
	复位值																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
010h	DMA_CPAR1	PA[31:0]																															
	复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
014h	DMA_CMAR1	MA[31:0]																															
	复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
018h	保留																																
01Ch	DMA_CCR2	保留																	MEM2MEM	PL [1:0]		MSIZE [1:0]		PSIZE [1:0]		MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN
	复位值																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
020h	DMA_CNDTR2	保留																	NDT[15:0]														
	复位值																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
024h	DMA_CPAR2	PA[31:0]																															
	复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
028h	DMA_CMAR2	MA[31:0]																															
	复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
02Ch	保留																																
030h	DMA_CCR3	保留																	MEM2MEM	PL [1:0]		MSIZE [1:0]		PSIZE [1:0]		MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN
	复位值																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
034h	DMA_CNDTR3	保留																	NDT[15:0]														
	复位值																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
038h	DMA_CPAR3	PA[31:0]																															
	复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
03Ch	DMA_CMAR3	MA[31:0]																															
	复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
040h	保留																																

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
044h	DMA_CCR4	保留																	MEM2MEM	PL [1:0]		MSIZE [1:0]		PSIZE [1:0]		MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN
	复位值																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
048h	DMA_CNDTR4	保留																	NDT[15:0]														
	复位值																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
04Ch	DMA_CPAR4	PA[31:0]																															
	复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
050h	DMA_CMAR4	MA[31:0]																															
	复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
054h	保留																																
058h	DMA_CCR5	保留																	MEM2MEM	PL [1:0]		MSIZE [1:0]		PSIZE [1:0]		MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN
	复位值																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
05Ch	DMA_CNDTR5	保留																	NDT[15:0]														
	复位值																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
060h	DMA_CPAR5	PA[31:0]																															
	复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
064h	DMA_CMAR5	MA[31:0]																															
	复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
068h	保留																																
06Ch	DMA_CCR6	保留																	MEM2MEM	PL [1:0]		MSIZE [1:0]		PSIZE [1:0]		MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN
	复位值																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
070h	DMA_CNDTR6	保留																	NDT[15:0]														
	复位值																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
074h	DMA_CPAR6	PA[31:0]																															
	复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
078h	DMA_CMAR6	MA[31:0]																															
	复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
07Ch	保留																																
080h	DMA_CCR7	保留																	MEM2MEM	PL [1:0]		MSIZE [1:0]		PSIZE [1:0]		MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN
	复位值																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
084h	DMA_CNDTR7	保留																	NDT[15:0]														
	复位值																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
088h	DMA_CPAR7	PA[31:0]																															
	复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
08Ch	DMA_CMAR7	MA[31:0]																															
	复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
090h	保留																																

8 实时时钟（RTC）

8.1 简介

实时时钟是一个独立的定时器。RTC 模块拥有一组连续计数的计数器，在相应软件配置下，可提供时钟日历的功能。修改计数器的值可以重新设置系统当前的时间和日期。

RTC 模块和时钟配置系统(RCC_BDCR 寄存器)是在后备区域，即在系统复位或从待机模式唤醒后 RTC 的设置和时间维持不变。

系统复位后，访问后备寄存器和 RTC 被暂时禁止，后备区域被保护以防意外的写操作，必须设置电源控制寄存器(PWR CR)的 DBP 位才能允许访问后备寄存器和 RTC。

8.2 主要特性

- 可编程的预分频系数：分频系数最高为 2^{20}。
- 32 位的可编程计数器，可用于较长时间段的测量。
- 2 个单独的时钟：用于 APB1 接口的 PCLK1 和 RTC 时钟（此时的 RTC 时钟必须小于 PCLK1 时钟的四分之一以上）
- 2 种独立的复位类型：
  - APB1 接口由系统复位
  - RTC 只能由后备域复位。
- 3 个专门的可屏蔽中断：
  - 闹钟中断，用来产生一个软件可编程的闹钟中断。
  - 秒中断，用来产生一个可编程的周期性中断信号（最长可达 1 秒）。
  - 溢出中断，检测内部可编程计数器溢出并回转为 0 的状态。

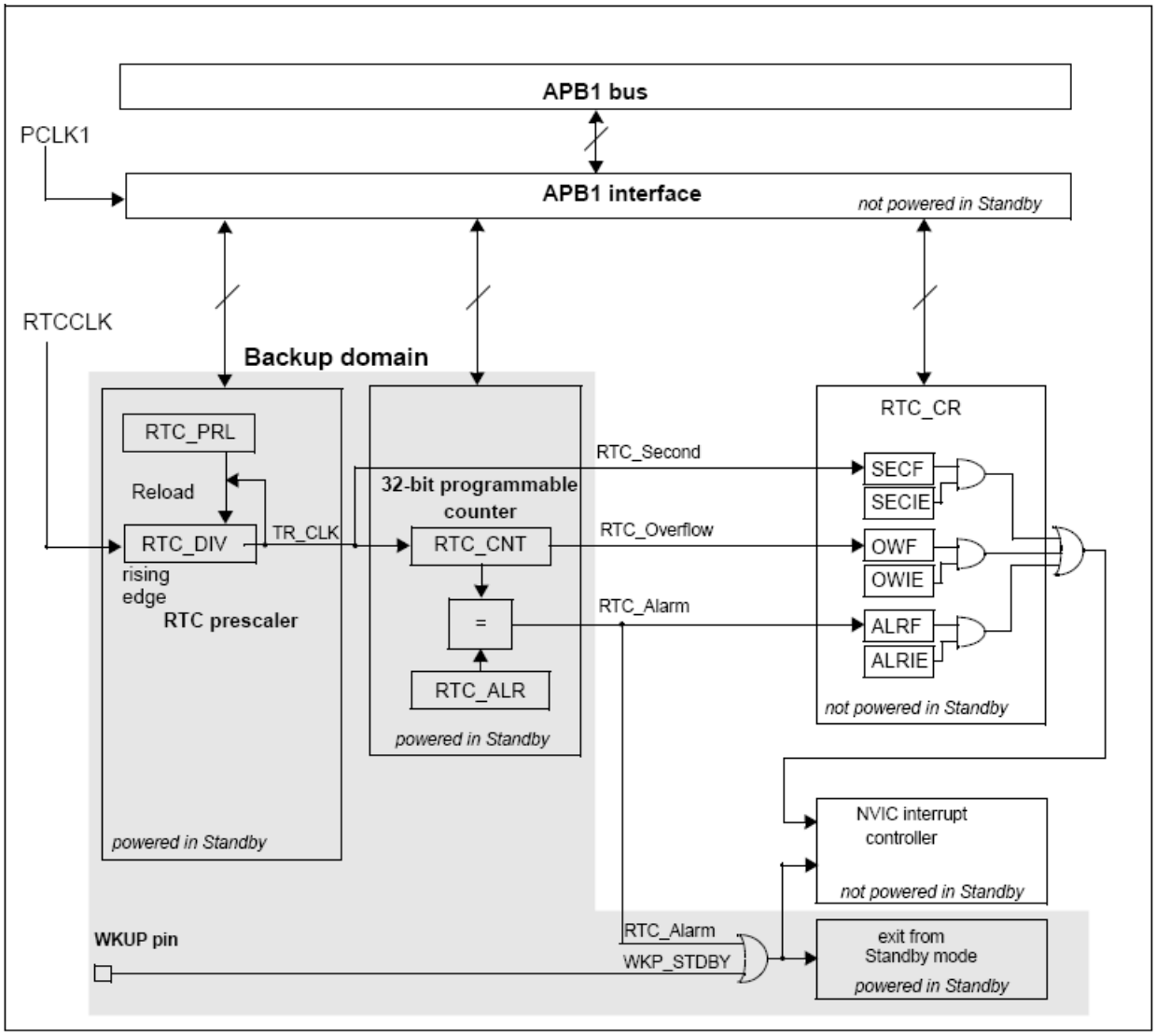
8.3 功能描述

8.3.1 概述

RTC 由两个主要部分组成。第一部分（APB1 接口）用来和 APB1 总线相连。此单元还包含一组 16 位寄存器，可通过 APB1 总线对其进行读写操作。APB1 接口以 APB1 总线时钟为时钟，用来与 APB1 总线接口。

另一部分（RTC 核）由一系列可编程计数器组成，分成两个主要模块。第一个模块是 RTC 的预分频模块，它可编程产生最长为 1 秒的 RTC 时间基准 TR_CLK。RTC 的预分频模块包含了一个 20 位的可编程分频器（RTC 预分频器）。在每个 TR_CLK 周期中，如果在 RTC_CR 寄存器中设置了相应允许位，则 RTC 产生一个中断（秒中断）。第 2 个模块是一个 32 位的可编程的计数器，它可被初始化为当前的系统时间。系统时间以 TR_CLK 速度增长并与存储在 RTC_ALR 寄存器中的可编程的时间相比较，如果 RTC_CR 控制寄存器中设置了相应允许位，则比较匹配时将产生一个闹钟中断。

图19 RTC 简化框图



8.3.2 复位过程

除了 RTC_PRL，RTC_ALR，RTC_CNT 和 RTC_DIV 寄存器外，所有的系统寄存器都由系统复位或电源复位进行异步复位。

RTC_PRL，RTC_ALR，RTC_CNT 和 RTC_DIV 寄存器仅能通过备份域复位信号复位。

8.3.3 读RTC寄存器

RTC 核完全独立于 RTC APB1 接口。

软件通过 APB1 接口访问 RTC 的预分频值、计数器值、和闹钟值。但是，相关的可读寄存器只在 RTC 时钟的上升沿被更新，而 RTC 时钟会与 RTC APB1 时钟进行重新同步。RTC 标志也是如此的。

这意味着，如果 APB1 接口刚刚被开启之后，在第一次的内部寄存器更新之前，从 APB1 上的 RTC 寄存器中读出的第一个值可能被破坏了。下述几种情况下能够发生这种情形：

- 发生系统复位或电源复位
- 系统刚从待机模式唤醒。
- 系统刚从停机模式唤醒。

在所有以上情况中，在 APB1 接口被禁止时(复位、无时钟或断电)RTC 核仍保持运行状态。

因此，若在读取 RTC 寄存器之前曾经禁止了 RTC 的 APB1 接口，软件首先须等待 RTC_CRL 寄存器中的 RSF 位(寄存器同步标志)被硬件置 1。

注： RTC 的 APB1 接口不受 WFI 和 WFE 等低功耗模式的影响。

8.3.4 配置RTC寄存器

要对 RTC_PRL、RTC_CNT、RTC_ALR 寄存器进行写操作，RTC 必须进入配置模式。通过对 RTC_CRL 寄存器中的 CNF 位置位使 RTC 进入配置模式。

另外，对 RTC 的任何寄存器的写操作都必须在前一次写操作结束以后进行。要使用软件来查询当前的状态，可通过查询 RTC_CR 寄存器中的 RTOFF 状态位，来判断 RTC 寄存器是否处于更新中。仅当 RTOFF 状态位是“1”时，RTC 寄存器可以写入新的值。

配置过程

1. 查询 RTOFF 位，直到 RTOFF 的值变为“1”
2. 置 CNF 值为 1，进入配置模式
3. 对一个或多个 RTC 寄存器进行写操作
4. 清除 CNF 标志位，退出配置模式
5. 查询 RTOFF，直至 RTOFF 位变为“1” 以确认写操作已经完成。

仅当 CNF 标志位被清除时，写操作才能进行，这个过程至少需要 3 个 RTCCLK 周期。

8.3.5 RTC标志的设置

RTC 秒标志(SECF)是在每一个 RTC 核心的时钟周期中，更改 RTC 计数器之前设置。

RTC 溢出标志(OWF)是在计数器到达 0x0000 之前的最后一个 RTC 时钟周期中设置。

RTC_Alarm 和 RTC 闹钟标志(ALRF)是在计数器的值到达闹钟寄存器的值加 1 之前的 RTC 时钟周期中设置。对 RTC 闹钟的写操作必须使用下述过程之一与 RTC 秒标志同步：

- 使用 RTC 闹钟中断，并在 RTC 中断处理程序中修改 RTC 闹钟和/或 RTC 计数器寄存器。
- 等待 RTC 控制寄存器中的 SECF 位被设置，再更改 RTC 闹钟和/或 RTC 计数器寄存器。

图20 RTC 秒和闹钟波形图示例，PR=0003，ALARM=00004

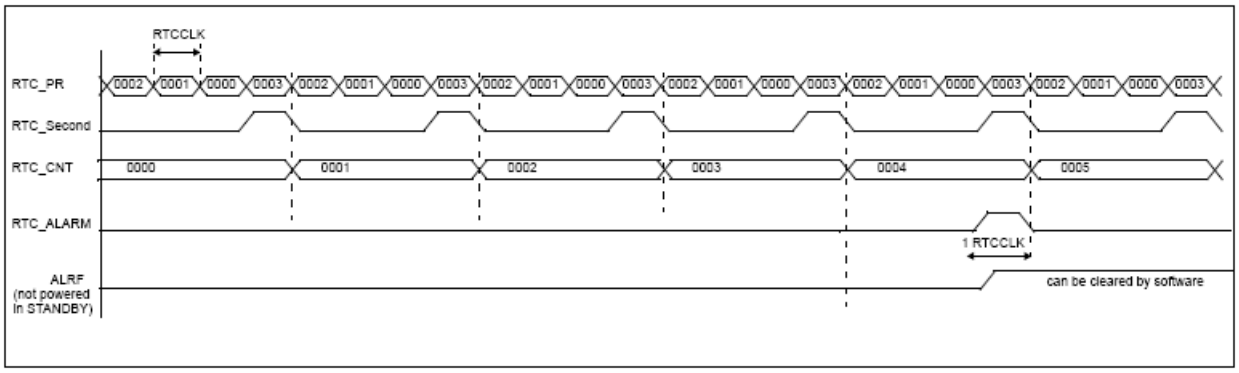
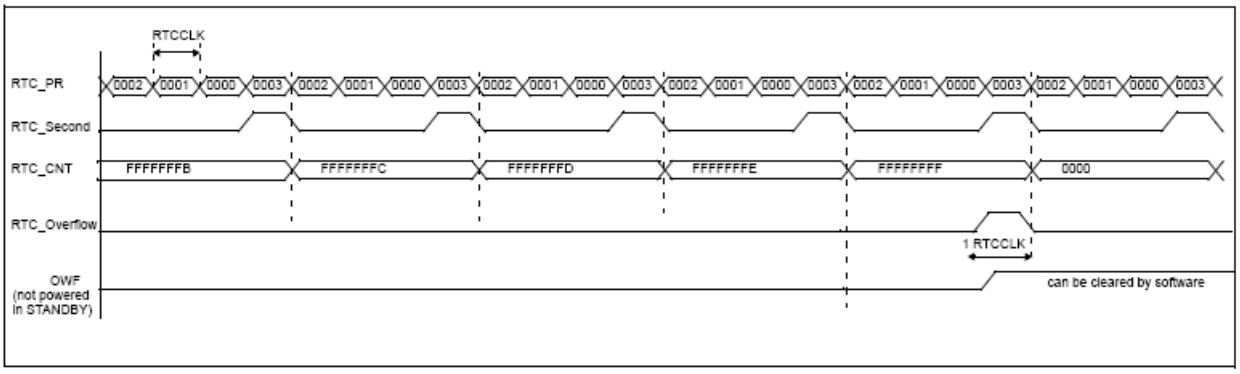


图21 RTC 溢出波形图示例，PR=0003



8.4 RTC寄存器描述

关于寄存器描述中的缩略词，请参考 1.1 节。

8.4.1 RTC控制寄存器高位（RTC_CRH）

地址偏移量：0x00
复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留													OWIE	ALRIE	SECIE
													rw	rw	rw

位15:3	保留，被硬件强制为0。
位2	OWIE：允许溢出中断位 0：屏蔽(不允许)溢出中断 1：允许溢出中断
位1	ALRIE：允许闹钟中断 0：屏蔽(不允许)闹钟中断 1：允许闹钟中断
位0	SECIE：允许秒中断 0：屏蔽(不允许)秒中断 1：允许秒中断

这些位用来屏蔽中断请求。注意：系统复位后所有的中断被屏蔽，因此可通过写 RTC 寄存器来确保在初始化后没有挂起的中断请求。当外设正在完成前一次写操作时（标志位 RTOFF=0），不能对 RTC_CRH 寄存器进行写操作。

RTC 功能由这个控制寄存器控制。一些位的写操作必须经过一个特殊的配置过程来完成(见配置过程)。

8.4.2 RTC控制寄存器低位（RTC_CRL）

偏移地址：0x04
复位值：0x0020

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留										RTOFF	CNF	RSF	OWF	ALRF	SECF
										r	rw	rc w0	rc w0	rc w0	rc w0

位15:6	保留，被硬件强制为0。
位5	RTOFF: RTC操作关闭 RTC模块利用这位来指示对其寄存器进行的最后一次操作的状态，指示操作是否

	完成。若此位为0，则无法对任何的RTC寄存器进行写操作。此位为只读位。 0：上一次对RTC寄存器的写操作仍在进行; 1：上一次对RTC寄存器的写操作已经完成。
位4	CNF:配置标志 此位必须由软件置1以进入配置模式，从而允许向RTC_CNT、RTC_ALR或RTC_PRL寄存器写入数据。只有当此位在被置1后，并重新由软件清0后，才会执行写操作。 0：退出配置模式(开始更新RTC寄存器）； 1：进入配置模式。
位3	RSF:寄存器同步标志 每当RTC_CNT寄存器和RTC_DIV寄存器由软件刷新或清0时，此位由硬件置1。在APB1复位后，或APB1时钟停止后，此位必须由软件清0。要进行任何的读操作之前，用户程序必须等待这位被硬件置1，以确保RTC_CNT、RTC_ALR或RTC_PRL已经被同步。 0：寄存器尚未被同步； 1：寄存器已经被同步。
位2	OWF:溢出标志 当32位可编程计数器溢出时，此位由硬件置1。如果RTC_CRH寄存器中OWIE=1，则产生中断。此位只能由软件清0。对此位写1是无效的。 0：无溢出； 1：32位可编程计数器溢出。
位1	ALRF:闹钟标志 当32位可编程计数器达到RTC_ALR寄存器所设置的门限值，此位由硬件置1。如果RTC_CRH寄存器中ALRIE=1，则产生中断。此位只能由软件清0。对此位写1是无效的。 0：无闹钟； 1：有闹钟。
位0	SECF:秒标志 当32位可编程预分频器溢出时，此位由硬件置1，RTC计数器递增。因此，此标志为分辨率可编程的RTC计数器提供一个周期性的信号（通常为1秒）。如果RTC_CRH寄存器中SECIE=1，则产生中断。此位只能由软件清除。对此位写“1”是无效的。 0：秒标志条件不成立； 1：秒标志条件成立。

RTC 的功能由这个控制寄存器控制。当正在前一个写操作还未完成时(RTOFF=0 时)，不能写 RTC_CR 寄存器。

注： *1 任何标志位都将保持挂起状态，直到适当的RTC_CR请求位被软件复位，表示所请求的中断已经被接受。*

2 在复位时禁止所有中断，无挂起的中断请求，并且可以对RTC寄存器进行写操作。

3 当APB1时钟不运行时，OWF、ALRF、SECF和RSF位不被更新。

4 OWF、ALRF、SECF和RSF位只能由硬件置位，由软件来清零。

5 若ALRF=1且ALRIE=1，则允许产生RTC全局中断。如果在EXTI控制器中允许产生EXTI线 17中断，则允许产生RTC全局中断和RTC闹钟中断。

6 若ALRF=1，如果在EXTI控制器中设置了EXTI线 17的中断模式，则允许产生RTC闹钟中断；如果在EXTI控制器中设置了EXTI线 17的事件模式，则这条线上会产生一个脉冲(不会产生RTC闹钟中断)。

8.4.3 RTC预分频装载寄存器（RTC_PRLH /RTC_PRLL）

预分频装载寄存器用来保存 RTC 预分频器的周期计数值。它们由在 RTC_CR 寄存器的 RTOFF 位写保护，并且如果 RTOFF 值是 1，允许进行写操作。

RTC预分频装载寄存器高位（RTC_PRLH）

偏移地址：0x08
只写(参见 8.3.4 节)
复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留												PRL[19:16]			
												w	w	w	w

位15:6	保留，被硬件强制为0。
位3:0	PRL[19:16]：RTC预分频装载值高位 根据以下公式，这些位用来定义计数器的时钟频率： $f_{TR\_CLK} = f_{RTCCLK}/(PRL[19:0]+1)$ 注：不推荐使用0值。这样，无法正确的产生RTC中断和标志位。

RTC预分频装载寄存器低位（RTC_PRLL）

偏移地址：0x0C
只写(参见 8.3.4 节)
复位值：0x8000

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

位15:0	PRL[15:0]：RTC预分频装载值低位 根据以下公式，这些位用来定义计数器的时钟频率： $f_{TR\_CLK} = f_{RTCCLK}/(PRL[19:0]+1)$

注： *如果输入时钟频率是32.768KHZ(f_{RTCCLK})，在这个寄存器上写7FFFh，可获得周期为1秒钟的信号。*

8.4.4 RTC预分频分频因子寄存器(RTC_DIVH / RTC_DIVL)

在 TR_CLK 的每个周期里，RTC 预分频器中计数器的值都会被重新设置为 RTC_PRL 寄存器的值。用户可读取存储在 RTC_DIV 寄存器里的预分频计数器的当前值，而不暂停其工作，从而获得精确的时间测量。此寄存器是只读寄存器，其值在 RTC_PRL 或 RTC_CNT 寄存器中的值发生改变后，由硬件重新装载。

RTC预分频分频因子寄存器高位（RTC_DIVH）

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
保留												RTC_DIV[19:16]			
												r	r	r	r

位15:4	保留
位3:0	RTC_DIV[19:16]：RTC时钟分频因子高位。

RTC预分频分频因子寄存器低位（RTC_DIVL）

偏移地址：0x14
复位值：0x8000

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

位15:0	RTC_DIV[15:0]：RTC时钟分频因子低位。

8.4.5 RTC计数器寄存器 (RTC_CNTH / RTC_CNTL)

RTC 核有一个 32 位可编程的计数器，可通过两个 16 位的寄存器访问。计数器以预分频器产生的 TR_CLK 时间基准为参考进行计数。RTC_CNT 寄存器用来存放计数器的计数值。他们被 RTC_CR 上的位 RTOFF 写保护，并且如果 RTOFF 值是 1 的话，允许写操作。在高或低寄存器(RTC_CNTH 或 RTC_CNTL)上的写操

作，能够直接装载到相应的可编程计数器，并且重新装载 RTC 预分频器。当进行读操作时，直接返回计数器内的计数值(系统时间)。

RTC 计数器寄存器高位（RTC_CNTH）
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
RTC_CNT[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位15:0	RTC_CNT[31:16]：RTC计数器高位。 可通过读RTC_CNTH寄存器来获得RTC计数器的当前值的高位部分。要对此寄存器进行写操作，必须先通过使用RTC_CR寄存器的RTOFF位来进入配置模式(参见8.3.4节)。

RTC计数器寄存器低位（RTC_CNTL）

偏移地址：0x1C
复位值：0x0000

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

位15:0	RTC_CNT[15:0]：RTC计数器低位。 可通过读RTC_CNTL寄存器来获得RTC计数器的当前值的低位部分。要对此寄存器进行写操作，必须先通过使用RTC_CR寄存器的RTOFF位来进入配置模式(参见8.3.4节)。

8.4.6 RTC闹钟寄存器（RTC_ALRH/RTC_ALRL）

当可编程计数器的值与 RTC_ALR 中的 32 位值相等时，即触发一个闹钟事件，并且产生 RTC 闹钟中断。此寄存器被 RTC_CR 寄存器里的 RTOFF 位写保护，如果 RTOFF 值是 1 时，允许写操作。

RTC闹钟寄存器高位（RTC_ALRH）

偏移地址：0x20
只写(参见 8.3.4 节)
复位值：0xFFFF

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RTC_ALR[31:16]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

位15:0	RTC_ALR[31:16]：RTC闹钟值高位。 此寄存器用来保存由软件写入的闹钟时间的高位部分。要对此寄存器进行写操作，必须先通过使用RTC_CR寄存器的RTOFF位来进入配置模式(参见8.3.4节)。

RTC闹钟寄存器低位（RTC_ALRL）

偏移地址：0x24
只写(参见 8.3.4 节)
复位值：0xFFFF

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

位15:0	RTC_ALR[15:0]：RTC闹钟值低位。 此寄存器用来保存由软件写入的闹钟时间的低位部分。要对此寄存器进行写操作，必须先通过使用RTC_CR寄存器的RTOFF位来进入配置模式(参见8.3.4节)。

8.5 RTC寄存器映像

RTC 寄存器是 16 位可寻址寄存器，具体描述如下：

表31 RTC-寄存器映像和复位值

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
000h	RTC_CRH	保留																													OWIE	ALRIE	SECIE
	复位值																														0	0	0
004h	RTC_CRL	保留																										RTOFF	CNF	RSF	OWF	ALRF	SECF
	复位值																											0	0	0	0	0	0
008h	RTC_PRLH	保留																													PRL[19:16]		
	复位值																														0	0	0	0
00Ch	RTC_PRLL	保留																PRL[15:0]															
	复位值																	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
010h	RTC_DIVH	保留																DIV[31:16]															
	复位值																	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
014h	RTC_DIVL	保留																DIV[15:0]															
	复位值																	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
018h	RTC_CNTH	保留																CNT[31:16]															
	复位值																	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
01Ch	RTC_CNTL	保留																CNT[15:0]															
	复位值																	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
020h	RTC_ALRH	保留																ALR[31:16]															
	复位值																	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
024h	RTC_ALRL	保留																ALR[15:0]															
	复位值																	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

9 备份寄存器(BKP)

9.1 简介

备份寄存器是 10 个 16 位的寄存器，可用来存储 20 个字节的用户应用程序数据。他们处在备份域里，当 V_{DD} 电源被切断，他们仍然由 V_{BAT} 维持供电。当系统在待机模式下被唤醒，或系统复位或电源复位时，他们也不会被复位。

此外，BKP 控制寄存器用来管理侵入检测和 RTC 校准功能。

复位后，对备份寄存器和 RTC 的访问被禁止，并且备份域被保护以防止可能存在的意外的写操作。

电源控制寄存器(PWR_CR)的 DBP 位必须被置 1，以允许访问备份寄存器和 RTC.

9.2 特性

- 20 字节的数据寄存器
- 用来管理防侵入检测并具有中功能的状态/控制寄存器
- 用来存储 RTC 校验值的校验寄存器。

9.3 侵入检测

当 TAMPER 引脚上的信号从 0 变成 1 或者从 1 变成 0(取决于备份控制寄存器 BKP_CR 的 TPAL 位)，会产生一个侵入检测事件。侵入检测事件将所有数据备份寄存器内容清除。

然而为了避免丢失侵入事件，侵入检测信号是边沿检测的信号与侵入检测允许位的逻辑与，从而在侵入检测引脚被允许前发生的侵入事件也可以被检测到。

- **当 TPAL=0 时：**如果在启动侵入检测 TAMPER 引脚前(通过设置 TPE 位)该引脚已经为高电平，一旦启动侵入检测功能，则会产生一个额外的侵入事件(尽管在 TPE 位置 1 后并没有出现上升沿)。
- **当 TPAL=1 时：**如果在启动侵入检测引脚 TAMPER 前(通过设置 TPE 位)该引脚已经为低电平，一旦启动侵入检测功能，则会产生一个额外的侵入事件(尽管在 TPE 位置 1 后并没有出现下降沿)。

在一个侵入事件被检测到并被清除后，侵入检测引脚 TAMPER 应该被禁止。然后，在再次写入备份数据寄存器前重新用 TPE 位启动侵入检测功能。这样，可以

阻止软件在侵入检测引脚上仍然有侵入事件时对备份数据寄存器进行写操作。这相当于对侵入引脚 TAMPER 进行电平检测。

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

9.4 RTC校准

为方便测量，32.768kHz 的 RTC 时钟可以输出到侵入检测引脚 TAMPER 上。通过设置 RTC 校验寄存器(BKP_RTCCR)的 CCO 位来开启这一功能。

通过配置 CAL[6:0]位，此时钟可以最多减慢 121ppm。
关于 RTC 校准和如何提高计时精度，请看 AN2604 “STM32F101xx 和 STM32F103xx 的 RTC 校准”

9.5 BKP寄存器描述

关于在寄存器描述里面所用到的缩写，可参考 1.1 节

9.5.1 备份数据寄存器x(BKP_DRx) (x = 1 … 10)

地址偏移：0x04 到 0x28
复位值：0x0000 0000

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

位15:0	D[15:0]：备份数据 这些位可以被用来写入用户数据。 注意：BKP_DRx寄存器不会被系统复位、电源复位、从待机模式唤醒所复位。 它们可以由备份域复位来复位或(如果侵入检测引脚TAMPER功能被开启时)由侵入引脚事件复位。

9.5.2 RTC时钟校准寄存器（BKP_RTCCR）

地址偏移：0x2C
复位值：0x0000 0000

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

位15:8	保留，始终读为0。
位9	ASOS：闹钟或秒输出选择 当设置了ASOE位，ASOS位可用于选择在TAMPER引脚上输出的是RTC秒脉冲还是闹钟脉冲信号。 0：输出RTC闹钟脉冲 1：输出秒脉冲 注：该位只能被后备区的复位所清除
位8	ASOE：允许输出闹钟或秒脉冲 根据ASOS位的设置，该位允许RTC闹钟或秒脉冲输出到TAMPER引脚上。输出脉冲的宽度为一个LSE的周期。设置了ASOE位时不能开启TAMPER的功能。 注：该位只能被后备区的复位所清除
位7	CCO：校准时钟输出 0：无影响 1：此位置1可以在侵入检测引脚输出经64分频后的RTC时钟。当CCO位置1时，必须关闭侵入检测功能以避免检测到无用的侵入信号。 注：当VDD供电断开时，该位被清除。
位6:0	CAL[6:0]：校准值 校准值表示在每2^{20}个时钟脉冲内将有多少个时钟脉冲被跳过。这可以用来对RTC进行校准，以1000000/(2^{20})ppm的比例减慢时钟。 RTC时钟可以被减慢0~121ppm。

9.5.3 备份控制寄存器(BKP_CR)

偏移地址：0x30
复位值：0x0000 0000

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

位15:2	保留，始终读为0。
位1	TPAL：侵入检测TAMPER引脚有效电平 0：侵入检测TAMPER引脚上的高电平会清除所有数据备份寄存器（如果TPE位为1） 1：侵入检测TAMPER引脚上的低电平会清除所有数据备份寄存器（如果TPE位为1）

位0	TPE：启动侵入检测TAMPER引脚 0：侵入检测TAMPER引脚作为通用IO口使用 1：开启侵入检测引脚作为侵入检测使用

注： *同时设置TPAL和TPE位总是安全的。然而，同时清除两者会产生一个假的侵入事件。因此，推荐只在TPE为0时才改变TPAL位的状态。*

9.5.4 备份控制/状态寄存器(BKP_CSR)

偏移地址：0x34
复位值：0x0000 0000

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

位15:10	保留，始终读为0。
位9	TIF：侵入中断标志 当检测到有侵入事件且TPIE位为1时，此位由硬件置1。通过向CTI位写1来清除此标志位(同时也清除了中断)。如果TPIE位被清除，则此位也会被清除。 0：无侵入中断 1：产生侵入中断 注意：仅当系统复位或由待机模式唤醒后才复位该位
位8	TEF：侵入事件标志 当检测到侵入事件时此位由硬件置1。通过向CTE位写1可清除此标志位 0：无侵入事件 1：检测到侵入事件 注：侵入事件会复位所有的BKP_DRx寄存器。只要TEF为1，所有的BKP_DRx寄存器就一直保持复位状态。当此位被置1时，若对BKP_DRx进行写操作，写入的值不会被保存。
位7:3	保留，始终读为0。
位2	TPIE：允许侵入TAMPER引脚中断 0：禁止侵入检测中断 1：允许侵入检测中断(BKP_CR寄存器的TPE位也必须被置1) 注1：侵入中断无法将系统内核从低功耗模式唤醒。 注2：仅当系统复位或由待机模式唤醒后才复位该位。
位1	CTI：清除侵入检测中断 此位只能写入，读出值为0。 0：无效 1：清除侵入检测中断和TIF侵入检测中断标志
位0	CTE：清除侵入检测事件 此位只能写入，读出值为0。 0：无效 1：清除TEF侵入检测事件标志(并复位侵入检测器)。

9.6 BKP寄存器映像

BKP 寄存器是 16 位的可寻址寄存器。

表32 BKP 寄存器映像和复位值

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
000h	保留																																
004h	BKP_DR1	保留																D[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
008h	BKP_DR2	保留																D[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
00Ch	BKP_DR3	保留																D[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
010h	BKP_DR4	保留																D[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
014h	BKP_DR5	保留																D[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
018h	BKP_DR6	保留																D[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
01Ch	BKP_DR7	保留																D[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
020h	BKP_DR8	保留																D[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
024h	BKP_DR9	保留																D[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
028h	BKP_DR10	保留																D[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
02Ch	BKP_RTCCR	保留																						ASOS	ASOE	CCO	CAL[6:0]						
	复位值																							0	0	0	0	0	0	0	0	0	0
030h	RTC_CNTL	保留																													TPAL	TPE	
	复位值																														0	0	
034h	RTC_ALRH	保留																						TIF	TFE	保留					TPIE	CTI	CTE
	复位值																							0	0						0	0	0

10 独立看门狗(IWDG)

STM32F10xxx 内置两个看门狗，提供了更高的安全性、时间的精确性和使用的灵活性。两个看门狗设备(独立看门狗和窗口看门狗)可用来检测和解决由软件错误引起的故障；当计数器达到给定的超时值时，触发一个中断或产生系统复位。

独立看门狗(IWDG)由专用的 40kHz 的低速时钟为驱动；因此，即使主时钟发生故障它也仍然有效。窗口看门狗由从 APB1 时钟分频后得到的时钟驱动，通过可配置的时间窗口来检测应用程序非正常的过迟或过早的行为。

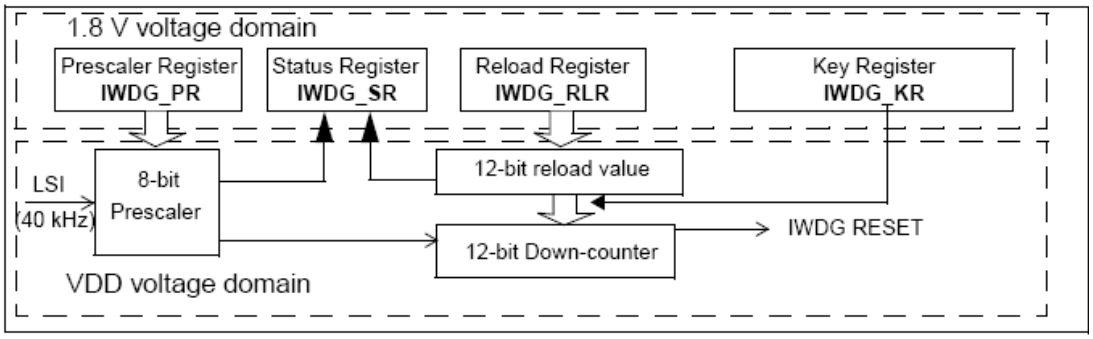
IWDG 最适合应用于那些需要看门狗作为一个在主程序之外，能够完全独立工作，并且对时间精度要求较低的场合。WWDG 最适合那些要求看门狗在精确计时窗口起作用的应用程序。

关于窗口看门狗的详情，请参看第 11 章。

10.1 简介

下图为独立看门狗模块的功能框图。

图22 独立看门狗框图



注： 看门狗功能处于VDD 供电区，但在停机和待机模式时仍能正常工作。

在键寄存器(IWDG_KR)中写入 0xCCCC，开始启用独立看门狗；此时计数器开始从其复位值 0xFFF 递减计数。当计数器计数到末尾 0x000 时，会产生一个复位信号(IWDG_RESET)。

无论何时，只要键寄存器 IWDG_KR 中被写入 0xAAAA， IWDG_RLR 中的值就会被重新加载到计数器中从而避免产生看门狗复位 。

10.1.1 硬件看门狗

如果用户在选择字节中启用了“硬件看门狗“功能，在系统上电复位后，看门狗会自动开始运行；如果在计数器计数结束前，若软件没有向键寄存器写入相应的值，则系统会产生复位。

10.1.2 寄存器访问保护

IWDG_PR 和 IWDG_RLR 寄存器具有写保护功能。 要修改这两个寄存器的值，必须先向 IWDG_KR 寄存器中写入 0x5555。以不同的值写入这个寄存器将会打乱操作顺序，寄存器访问将重新被保护。重装载操作(即写入 0xAAAA)也会启动写保护功能。

状态寄存器指示预分频值和递减计数器是否正在被更新。

10.1.3 调试模式

当微控制器进入调试模式时(Cortex-M3 核心停止)，根据调试模块中的 DBG_IWDG_STOP 配置位的状态，IWDG 的计数器能够继续工作或停止。详见有关调试模块的章节。

表33 看门狗超时时间(32kHz 的输入时钟)

预分频系数	PR[2:0]位	最短时间(ms) RL[11:0]=0x000	最长时间(ms) RL[11:0]=0xFFF
/4	0	0.1	4096
/8	1	0.2	8192
/16	2	0.4	16384
/32	3	0.8	32768
/64	4	1.6	65536
/128	5	3.2	131072
/256	(6或7)	6.4	262144

注： *尽管这个时钟号称是40kHz时钟，但是MCU内部RC的频率会在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振荡器的40KHZ时钟间的相位差，因此总会有一个完整的RC周期是不确定的。*

10.2IWDG寄存器描述

关于在寄存器描述里面所用到的缩写，详见第 1 章。

10.2.1 键值寄存器（IWDG_KR）

地址偏移：0x00
复位值：0x0000 0000 (在待机模式复位)

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

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31:16	保留，始终读为0。
位15:0	KEY[15:0]: 键值(只写寄存器，读出值为0x0000) 此寄存器必须由软件以一定的间隔写入0xAAAA，否则，当计数器为0时，看门狗会产生复位。 对此寄存器写入0x5555表示允许访问IWDG_PR和IWDG_RLR寄存器。（见10.1.2节） 对此寄存器写入0xCCCC，启动看门狗工作（若选择了硬件看门狗则不受此命令字限制）。

10.2.2 预分频寄存器(IWDG_PR)

地址偏移：0x04
复位值：0x0000 0000

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

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

位31:3	保留，始终读为0。
位2:0	PR[2:0]: 预分频因子 这些位具有写保护设置，参见10.1.2节。通过设置这些位来选择计数器时钟的预分频因子。要改变预分频因子，IWDG_SR寄存器的PVU位必须为0。

	000: /4 001: /8 010: /16 011: /32 100: /64 101: /128 110: /256 111: /256 注意：对此寄存器进行读操作，将从VDD电压域返回预分频值。如果写操作正在进行，则读回的值可能是无效的。因此，只有当IWDG_SR寄存器的PVU位为0时，从此寄存器读出的值才有效。

10.2.3 重装载寄存器(IWDG_RLR)

地址偏移：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
保留															

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

位31:12	保留，始终读为0。
位11:0	RL[11:0]: 看门狗计数器重装载值 这些位具有写保护功能，参看10.1.2节。用于定义看门狗计数器的重装载值，每当向IWDG_KR寄存器写入0xAAAA时，重装载值就会被装载到计数器中。随后，看门狗计数器从这个值开始递减计数。看门狗超时周期可通过此重装载值和时钟预分频值来计算，参照表33。 只有当IWDG_SR寄存器中的RVU位为0时，才能对此寄存器进行修改。 注：对此寄存器进行读操作，将从VDD电压域返回预分频值。如果写操作正在进行，则读回的值可能是无效的。因此，只有当IWDG_SR寄存器的RVU位为0时，从此寄存器读出的值才有效。

10.2.4 状态寄存器(IWDG_SR)

地址偏移：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
保留															

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

位31:2	保留。
位1	RVU: 看门狗计数器重装载值更新 此位由硬件置1用来指示重装载值的更新正在进行中。当在VDD域中的重装载更新结束后，此位由硬件清0（最多需5个40kHz的RC周期）。重装载值只有在RVU位被清0后才可更新。
位0	PVU: 看门狗预分频值更新 此位由硬件置1用来指示预分频值的更新正在进行中。当在VDD域中的预分频值更新结束后，此位由硬件清0（最多需5个40kHz的RC周期）。预分频值只有在RVU位被清0后才可更新。

注： *如果在应用程序中使用了多个重装载值或预分频值，则必须在RVU位复位后才能重新改变预装载值，在PVU位复位后才能重新改变预分频值。然而，在预分频和/或重装值更新后，不必等待RVU或PVU复位，可继续执行下面的代码。（即是在低功耗模式下，此写操作仍会被继续执行完成。）*

10.3 IWDG寄存器映像

表34 IWDG 寄存器映像和复位置

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
000h	IWDG_KR	保留																KEY[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
004h	IWDG_PR	保留																														PR[2:0]	
	复位值																															0	0	0
008h	IWDG_RLR	保留																RL[11:0]															
	复位值																	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
00Ch	IWDG_SR	保留																														RVU	PVU
	复位值																															0	0

11 窗口看门狗(WWDG)

11.1 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，此看门狗电路在达到可编程的时间周期时，会产生一个 MCU 复位。在递减计数器达到窗口寄存器值之前，如果递减计数器值的第 7 位(在控制寄存器中)被刷新， 那么也将产生一个 MCU 复位。这表明递减计数器需要在一个有限的窗口中被刷新。

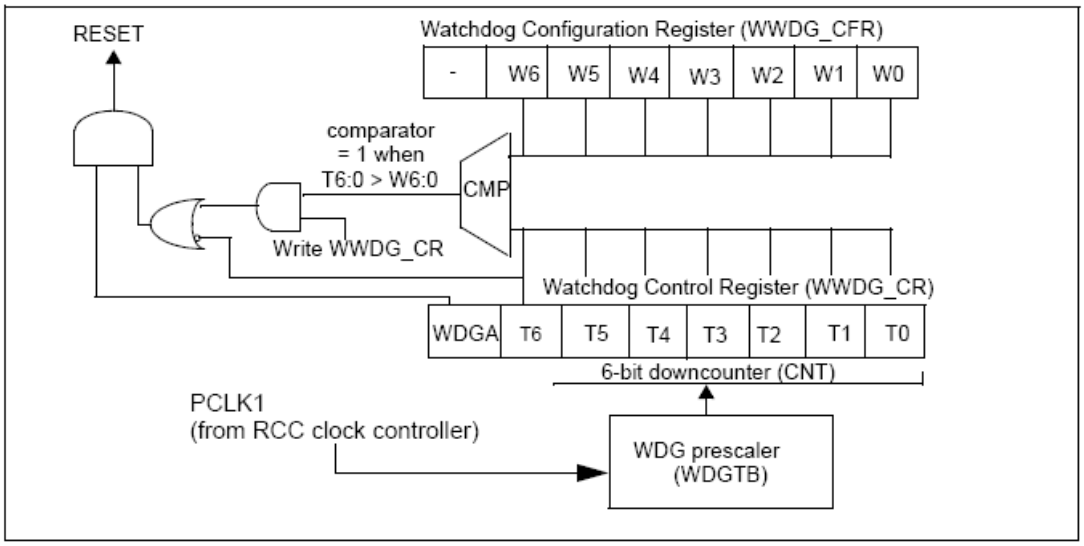
11.2 主要特性

- 可编程的自由运行递减计数器
- 条件复位
  - 当递减计数器的值小于 0x40，(若看门狗被启动)则产生复位。
  - 当递减计数器在窗口外被重新装载，(若看门狗被启动)则产生复位。见 图 24。
- 如果启动了看门狗且允许中断，当递减计数器等于 0x40 时产生早期唤醒中断(EWI)，它可以被用于重装载计数器以避免 WWDG 复位。

11.3 功能描述

如果看门狗被启动(WWDG_CR 寄存器中的 WDGA 位被置 1)， 并且当 7 位(T[6:0])递减计数器从 0x40 翻转到 0x3F(T6 位清零)时，则产生一个复位。如果软件在计数器值大于窗口寄存器中的值时重新装载计数器，将产生一个复位。

图23 看门狗框图



应用程序在正常的运行过程中必须每隔一定的时间间隔写 WWDG_CR 寄存器以防止 MCU 发生复位。只有当计数器值小于窗口寄存器的值时，才能进行这个写操作。这个要被储存在 WWDG_CR 寄存器中的值必须在 0xFF 和 0xC0 之间：

- 启动看门狗
- 看门狗通常在复位后被禁止。设置 WWDG_CR 寄存器中的 WDGA 位将启动看门狗，一旦被启动后，看门狗则不能再被关闭，除非发生复位。
- 控制递减计数器
- 递减计数器处于自由运行状态，即使看门狗被禁止，递减计数器仍继续递减计数。当看门狗被启用时，T6 位必须被设置，以防止立即产生一个复位。
- T[5:0]位包含了在看门狗产生复位之前的延时增量；复位前的延时时间在一个最小值和一个最大值之间变化，这是因为写入 WWDG_CR 寄存器时，预分频值是未知的。
- 配置寄存器(WWDG_CFR) 中包含窗口的上限值：要避免产生复位，递减计数器必须在其值小于窗口寄存器的值并且大于 0x3F时被重新装载，图 24 描述了窗口寄存器的工作过程。
- 另一个重装载计数器的方法是利用早期唤醒中断(EWI)。该中断由设置 WWDG_CFR 寄存器中的 WEI 位开启，当递减计数器到达 0x40 时，则产生此中断，相应的中断服务程序(ISR)可以用来加载计数器以防止 WWDG 复位。在 WWDG_SR 寄存器中写 0 可以清除该中断。

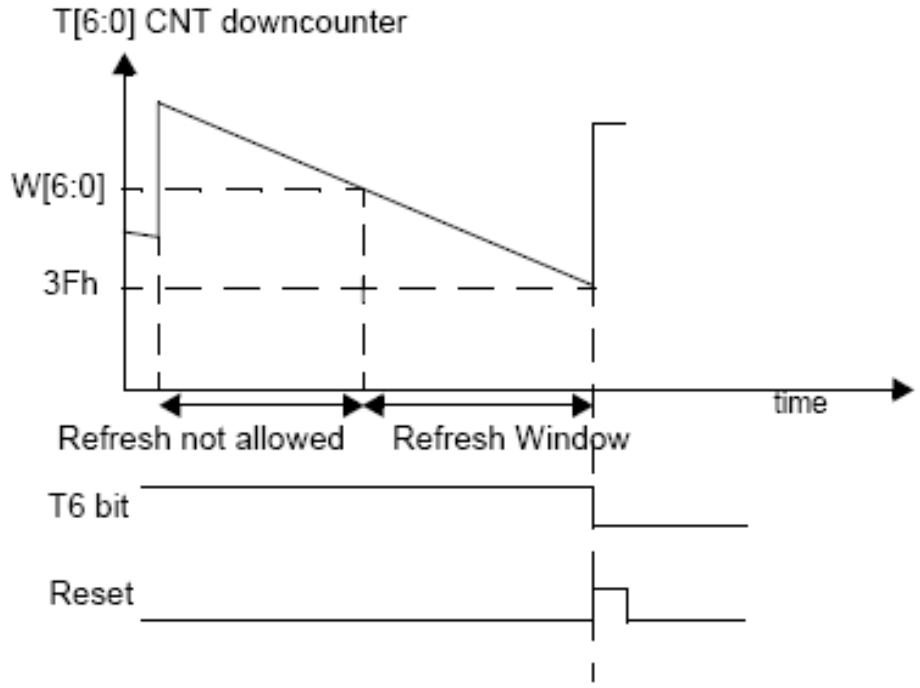
注： T6 位可以被用来产生一个软件复位(WDGA 位被置位，T6 位清零)

11.4如何编写看门狗超时程序

图 24 显示了装载到看门狗计数器（CNT）中的 6 位计数值和看门狗的延迟时间之间的线性关系（以ms为单位）。此图可用来做为快速计算的参考而未将时间的偏差考虑在内。如果需要更高的精度，可以使用 图 24 提供的计算公式。

警告：当写入 WWDG_CR 寄存器时，始终把 T6 位写 1 来避免立即产生一个复位。

图24 窗口看门狗时序图



计算超时的公式如下：

$T_{WWDG} = T_{PCLK1} \times 4096 \times 2^{WDGTB} \times (T[5:0] + 1)$; (ms)

其中：

T_{WWDG}：WWDG 超时时间

T_{PCLK1}：APB1 以 ms 为单位的时钟间隔

在 PCLK1=36MHz 时的最小-最大超时值

WDGTB	最小超时值	最大超时值
0	113μs	7.28ms
1	227μs	14.56ms
2	455μs	29.12ms
3	910μs	58.25ms

11.5 调试模式

当微控制器进入调试模式时(Cortex-M3 核心停止)，根据调试模块中的 DBG_WWDG_STOP 配置位的状态，WWDG 的计数器能够继续工作或停止。详见有关调试模块的章节。

11.6 寄存器描述

关于在寄存器描述里面所用到的缩写，详见第 1 章。

11.6.1 控制寄存器(WWDG_CR)

地址偏移量：0x00
复位值：0x7F

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留								WDGA	T6	T5	T4	T3	T2	T1	T0
								rs	rw	rw	rw	rw	rw	rw	rw

位31:8	保留。
位7	WDGA: 激活位 此位由软件置1，但仅能由硬件在硬件复位后清0。当WDGA=1时，看门狗可以产生复位。 0：禁止看门狗 1：启用看门狗
位6:0	T[6:0]: 7位计数器（MSB至LSB） 这些位用来存储看门狗的计数器值。每个PCLK1周期（$4096x2^{WDGTB}$）减1.当计数器值从40h变为3Fh时（T6被清0），产生看门狗复位。

11.6.2 配置寄存器(WWDG_CFR)

地址偏移量：0x04
复位值：0x7F

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留						EWI	WDG TB1	WDG TB0	W6	W5	W4	W3	W2	W1	W0
						rs	rw	rw	rw	rw	rw	rw	rw	rw	rw

位31:8	保留。
位9	EWI: 提前唤醒中断 此位若置1，则无论何时，当计数器值达到40h，即产生中断。 此中断只能由硬件在复位后清除。
位8:7	WDGTB[1:0]: 时基 预分频器的时基可根据如下修改： 00: CK计时器时钟(PCLK1除以4096) 除以1 01: CK计时器时钟(PCLK1除以4096) 除以2 10: CK计时器时钟(PCLK1除以4096) 除以4 11: CK计时器时钟(PCLK1除以4096) 除以8
位6:0	W[6:0]: 7位窗口值 这些位包含了用来与递减计数器进行比较用的窗口值。

11.6.3 状态寄存器(WWDG_SR)

地址偏移量：0x08
复位值：0x00

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留															EWIF
															rc w0

位31:1	保留。
位0	EWIF: 提前唤醒中断标志 当计数器值达到40h时，此位由硬件置1。它必须通过软件写“0”来清除。对此位写“1”无效。若中断未被使能，此位也会被置1。

11.7 WWDG寄存器映像

表35 WWDG 寄存器映像和复位值

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
000h	WWDG_CR	保留																								WDGA	T[6:0]						
	复位值																									0	1	1	1	1	1	1	1
004h	WWDG_CFR	保留																						EWI	WDGTB1	WDGTB0	W[6:0]						
	复位值																							0	0	0	1	1	1	1	1	1	1
008h	WWDG_SR	保留																															EWIF
	复位值																																0

12 高级控制定时器(TIM1)

12.1 简介

高级控制定时器(TIM1)由一个 16 位的自动装载计数器组成，它由一个可编程预分频器驱动。

它适合多种用途，包含测量输入信号的脉冲宽度(输入捕获)，或者产生输出波形(输出比较，PWM，嵌入死区时间的互补 PWM 等)。

使用定时器预分频器和 RCC 时钟控制预分频器，可以实现脉冲宽度和波形周期从几个微秒到几个毫秒的调节。

高级控制(TIM1)和通用(TIMx)定时器是完全独立的，它们不共享任何资源。它们可以同步操作，具体描述参看 12.4.20.

12.2 主要特性

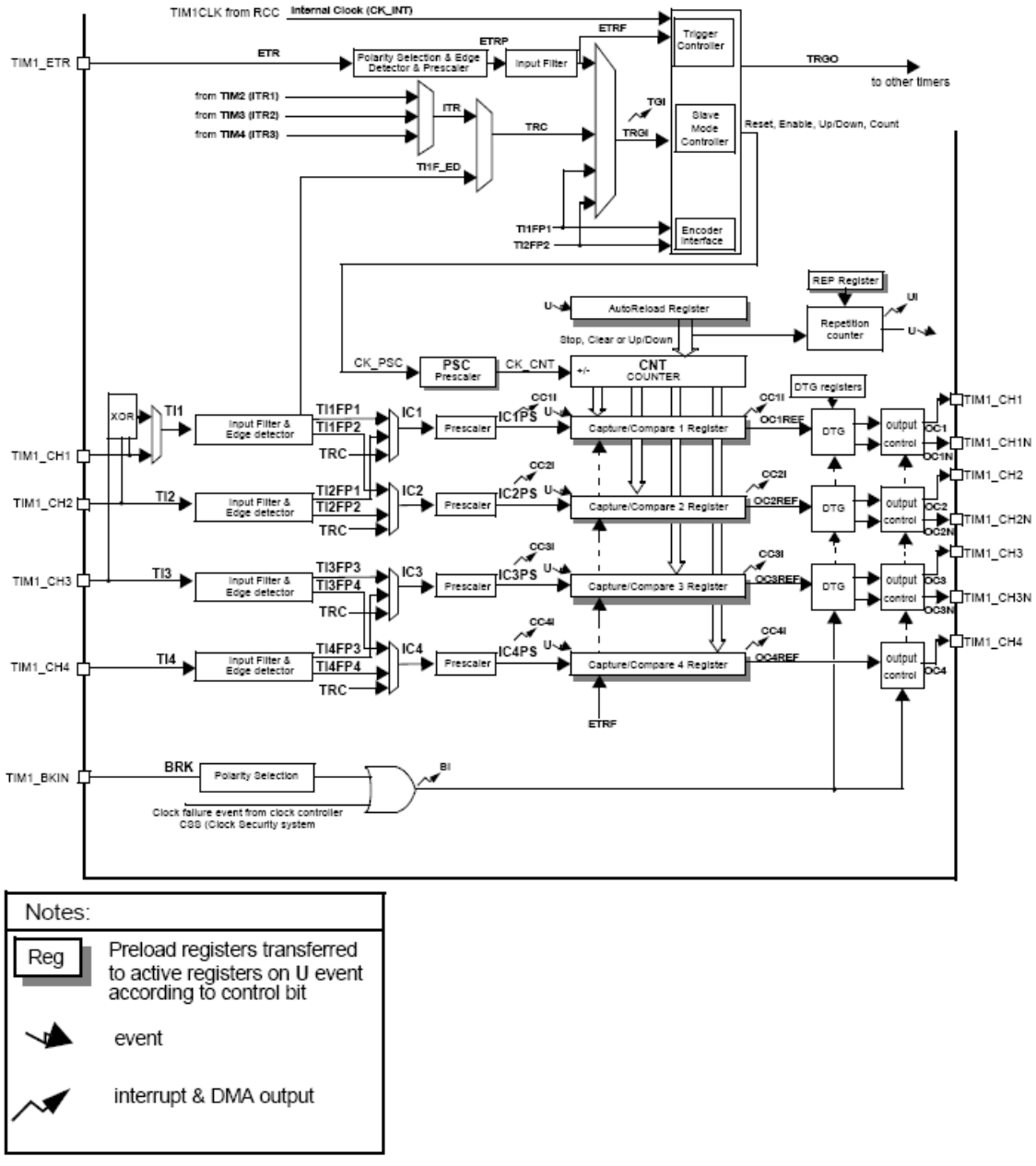
TIM1 定时器的功能包括：

- 16 位上，下，上/下自动装载计数器
- 16 位可编程预分频器，计数器时钟频率的分频系数为 1～65535 之间的任意数值
- 4 个独立通道：
  - 输入捕获
  - 输出比较
  - PWM 生成(边缘或中间对齐模式)
  - 单脉冲模式输出
  - 死区时间可编程的互补输出
- 使用外部信号控制定时器和定时器互连的同步电路
- 在指定数目的计数器周期之后更新定时器寄存器
- 刹车输入信号可以将定时器输出信号置于复位状态或者一个已知状态
- 如下事件发生时产生中断/DMA：
  - 更新：计数器向上溢出/向下溢出，计数器初始化(通过软件或者内部/外部触发)
  - 触发事件(计数器启动，停止，初始化或者由内部/外部触发计数)

- 输入捕获
- 输出比较
- 刹车信号输入

12.3 框图

图25 高级控制定时器(TIM1)框图



12.4功能描述

12.4.1 时基单元

可编程高级控制定时器的主要部分是一个 16 位计数器和与其相关的自动装载寄存器。这个计数器可以向上计数、向下计数或者向上向下双向计数。此计数器时钟由预分频器分频得到。

计数器、自动装载寄存器和预分频器寄存器可以由软件读写，即使计数器还在运行读写仍然有效。

时基单元包含：

- 计数器寄存器(TIM1_CNT)
- 预分频器寄存器 (TIM1_PSC)
- 自动装载寄存器 (TIM1_ARR)
- 周期计数寄存器 (TIM1_RCR)

自动装载寄存器是预先装载的。写或读自动重装载寄存器将访问预装载寄存器。根据在 TIM1_CR1 寄存器中的自动装载预装载使能位(ARPE)的设置，预装载寄存器的内容被永久地或在每次的更新事件 UEV 时传送到影子寄存器。当计数器达到溢出条件(向下计数时的下溢条件)并当 TIM1_CR1 寄存器中的 UDIS 位等于 0 时，产生更新事件。更新事件也可以由软件产生。随后会详细描述每一种配置下更新事件的产生。

计数器由预分频器的时钟输出 CK_CNT 驱动，仅当设置了计数器 TIM1_CR1 寄存器中的计数器使能位(CEN)时，CK_CNT 才有效。(有关更多的计数器使能的细节，请参见控制器的从模式描述)。

注：真正的计数器使能信号 CNT_EN 是在 CEN 后的一个时钟周期后被设置。

预分频器描述

预分频器可以将计数器的时钟频率按 1 到 65536 之间的任意值分频。它是基于一个(在 TIM1_PSC 寄存器中的)16 位寄存器控制的 16 位计数器。因为这个控制寄存器带有缓冲器，它能够在工作时被改变。新的预分频器的参数在下一次更新事件到来时被采用。

图 27 和 图 28 给出了一些在预分频器工作时，更改其参数的情况下计数器操作的例子。

图26 当预分频器的参数从 1 变到 2 时，计数器的时序图

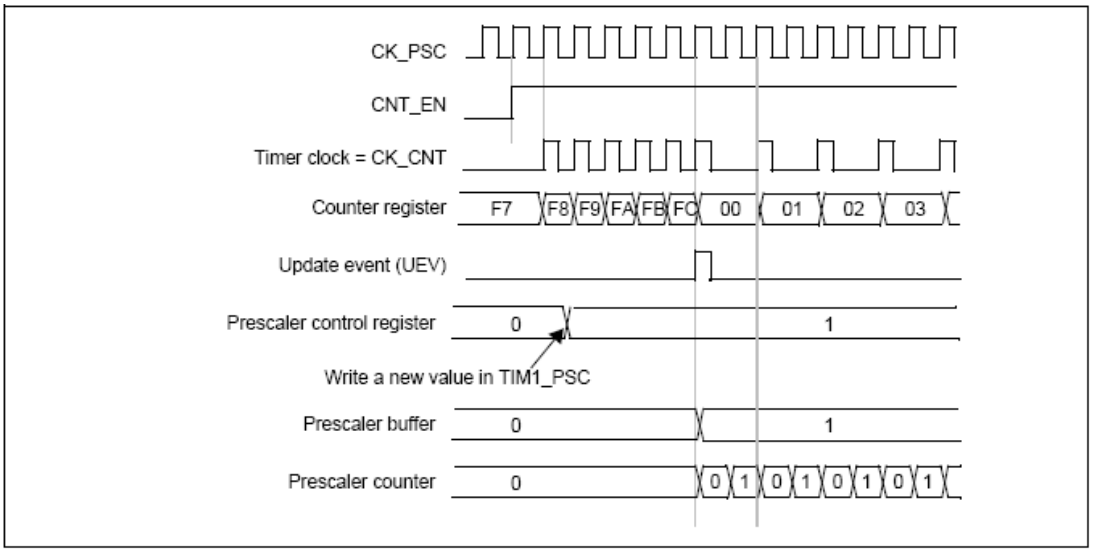
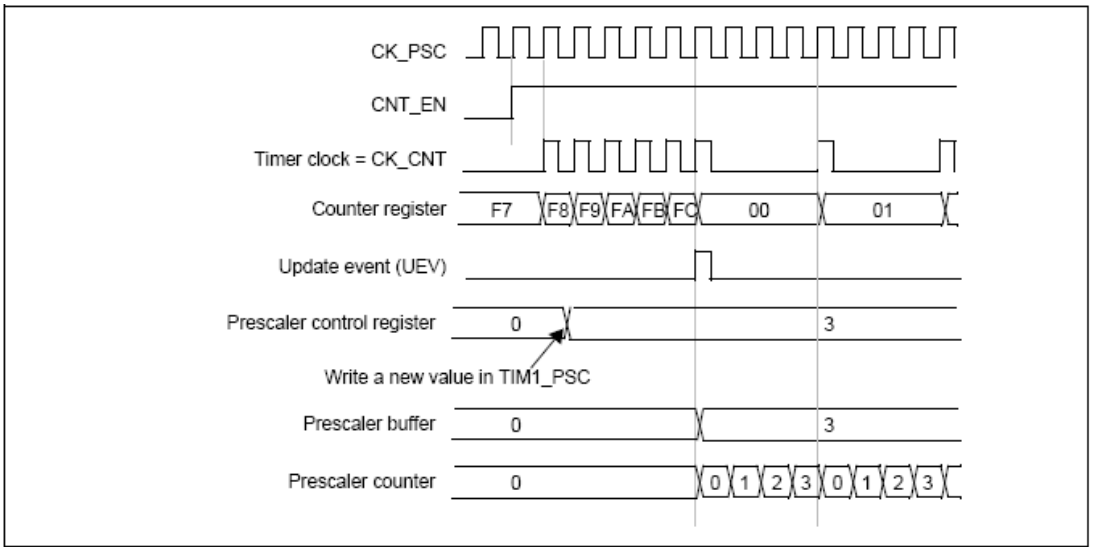


图27 当预分频器的参数从 1 变到 4 时，计数器的时序图



12.4.2 计数器模式

向上计数模式

在向上计数模式中，计数器从 0 计数到自动加载值(TIM1_ARR 计数器的内容)，然后重新从 0 开始计数并且产生一个计数器溢出事件。

如果使用了周期计数功能，在向上计数达到设置的周期计数次数(TIM1_RCR)时，将产生更新事件(UEV)；否则每次计数器溢出时才产生更新事件。

在 TIM1_EGR 寄存器中设置 UG 位(通过软件方式或者使用从模式控制器)也同样可以产生一个更新事件。

通过软件设置 TIM1_CR1 寄存器中的 UDIS 位，将禁止更新事件；这样可以避免在向预装载寄存器中写入新值时更新影子寄存器。在 UDIS 位被清成 0 之前，将

没有更新事件产生。即使这样，在应该产生更新事件时，计数器仍会被清 0，同时预分频器的计数也被请 0(但预分频器的数值不变)。此外，如果 TIM1_CR1 寄存器中的 URS 位(更新请求选择)被设置，设置 UG 位将产生一个更新事件 UEV，但硬件不设置 UIF 标志(即不产生中断或者 DMA 请求)。这是为了避免在捕获模式下清除计数器时，同时产生更新和捕获中断。

当发生一个更新事件，所有的寄存器都被更新，硬件同时(依据 URS 位)设置更新标志位(TIM1_SR 寄存器中的 UIF 位)。

- 周期计数器被重新加载为 TIM1_RCR 寄存器的内容。
- 自动装载影子寄存器被重新置入预装载寄存器的值(TIM1_ARR)。
- 预分频器的缓冲区被置入预装载寄存器的值(TIM1_PSC 寄存器的内容)。

下图给出一些例子，当 TIM1_ARR=0x36 时计数器在不同时钟频率下的动作。

图28 计数器时序图，内部时钟分频因子为 1

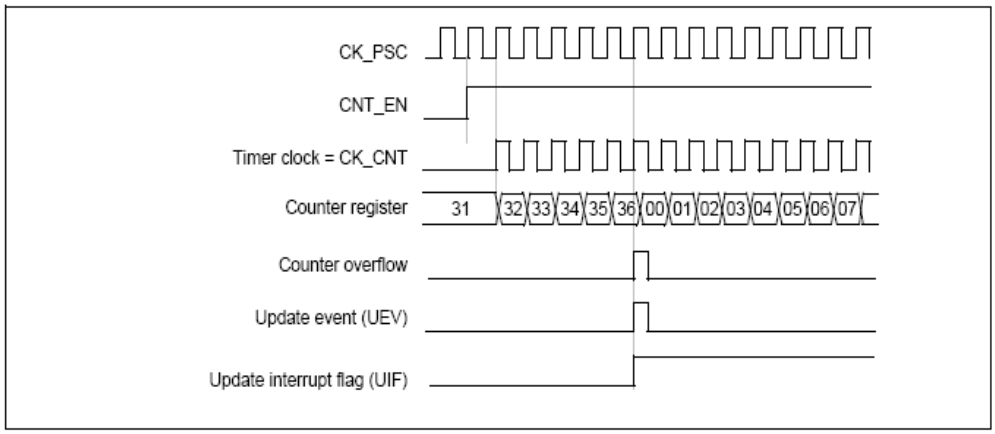


图29 计数器时序图，内部时钟分频因子为 2

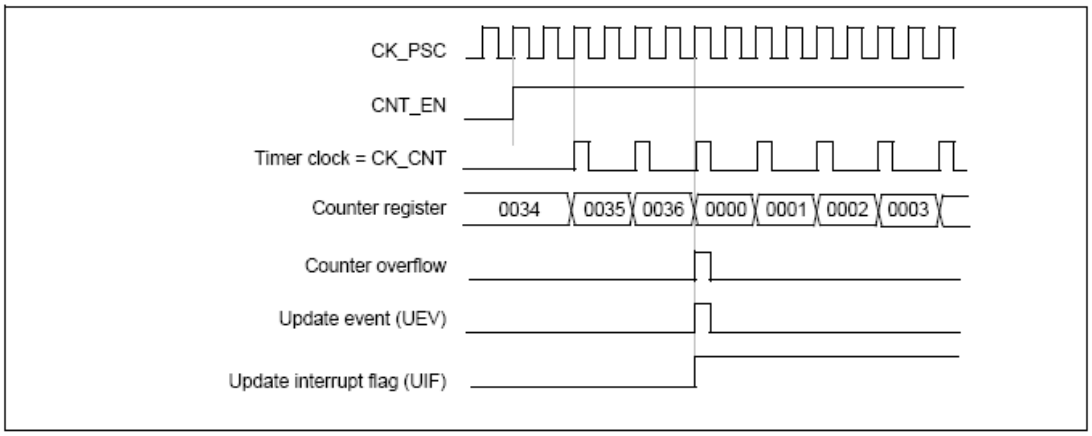


图30　计数器时序图，内部时钟分频因子为 4

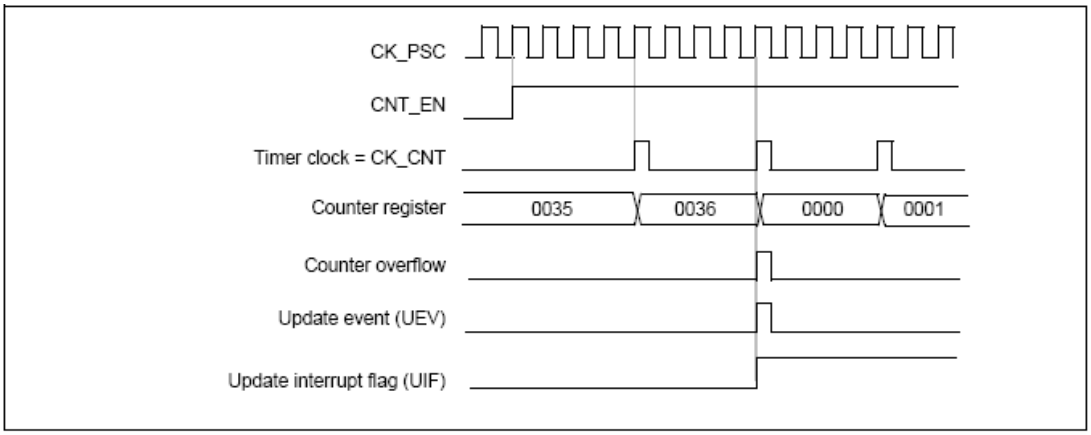


图31　计数器时序图，内部时钟分频因子为 N

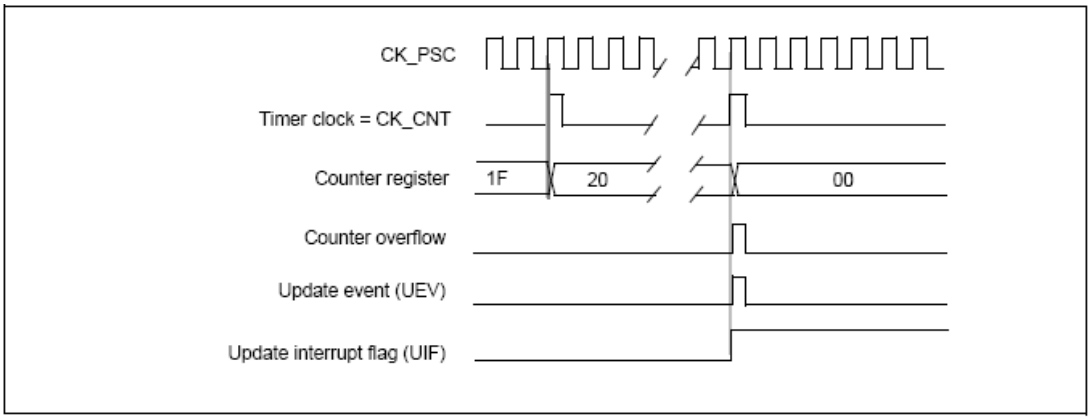


图32　计数器时序图，当 ARPE=0 时的更新事件(TIM1_ARR 没有预装入)

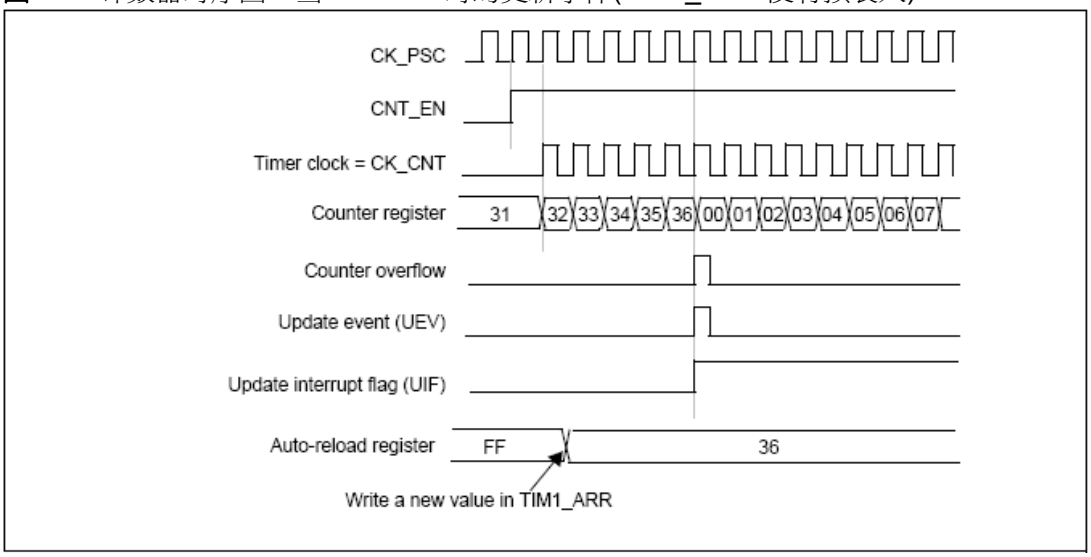
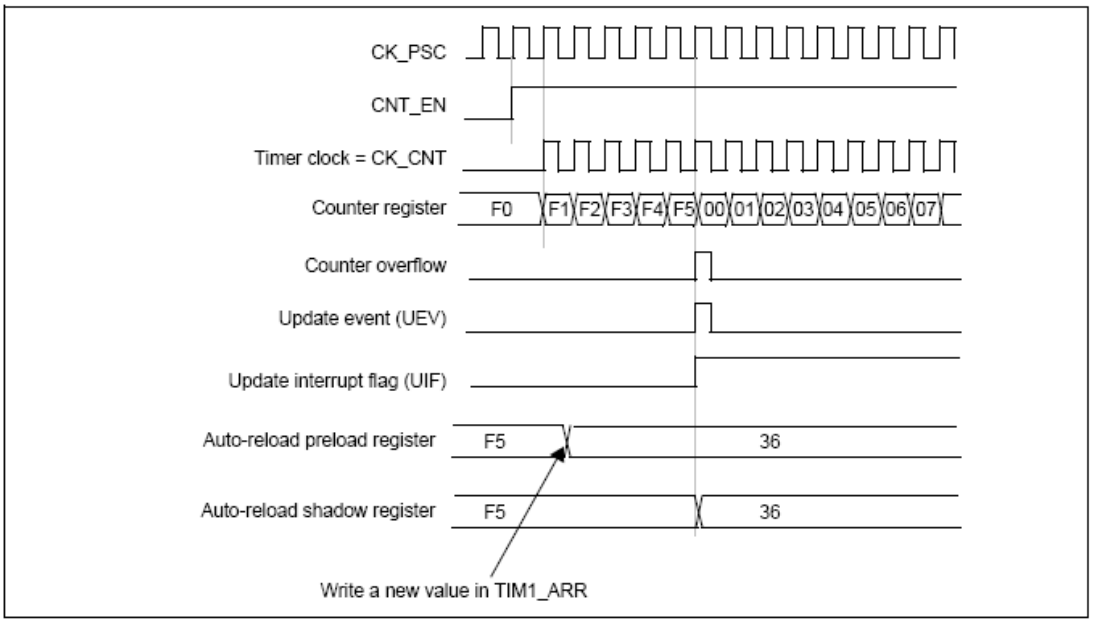


图33 计数器时序图，当 ARPE=1 时的更新事件(预装入了 TIM1_ARR)



向下计数模式

在向下模式中，计数器从自动装入的值(TIM1_ARR 计数器的值)开始向下计数到 0，然后从自动装入的值重新开始并且产生一个计数器向下溢出事件。

如果使用周期计数器，当向下计数重复了周期计数寄存器(TIM1_RCR)中设定的次数后，将产生更新事件(UEV)，否则每次计数器下溢时才产生更新事件。

在 TIM1_EGR 寄存器中设置 UG 位(通过软件方式或者使用从模式控制器)也同样可以产生一个更新事件。

UEV 事件可以通过软件设置 TIM1_CR1 寄存器中的 UDIS 位被禁止。这样可以避免在向预装载寄存器中写入新值时更新影子寄存器。这样 UDIS 位被写成 0 之前不会产生更新事件。然而，计数器仍会从当前自动加载值重新开始计数，并且预分频器的计数器重新从 0 开始(但预分频器的速率不能被修改)。

此外，如果设置了 TIM1_CR1 寄存器中的 URS 位(更新请求选择) ，设置 UG 位将产生一个更新事件 UEV 但不设置 UIF 标志(因此不产生中断和 DMA 请求)，这是为了避免在发生捕获事件并清除计数器时，同时产生更新和捕获中断。

当发生更新事件时，所有的寄存器都被更新，并且(根据 URS 位的设置)更新标志位(TIM1_SR 寄存器中的 UIF 位)也被设置。

- 周期计数器被重置为 TIM1_RCR 寄存器中的内容
- 当前的自动加载寄存器被更新为预装载值(TIM1_ARR 寄存器中的内容)。
  注：自动装载在计数器重载入之前被更新，因此下一个周期将是预期的值。

以下的图显示一些当 TIM1_ARR=0x36 时计数器在不同时钟频率下的操作例子。

图34　计数器时序图，内部时钟分频因子为 1

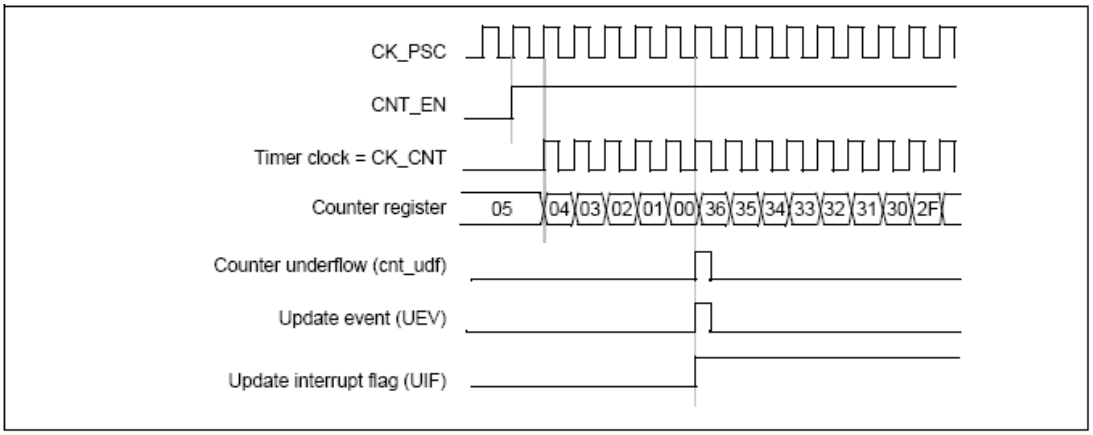


图35　计数器时序图，内部时钟分频因子为 2

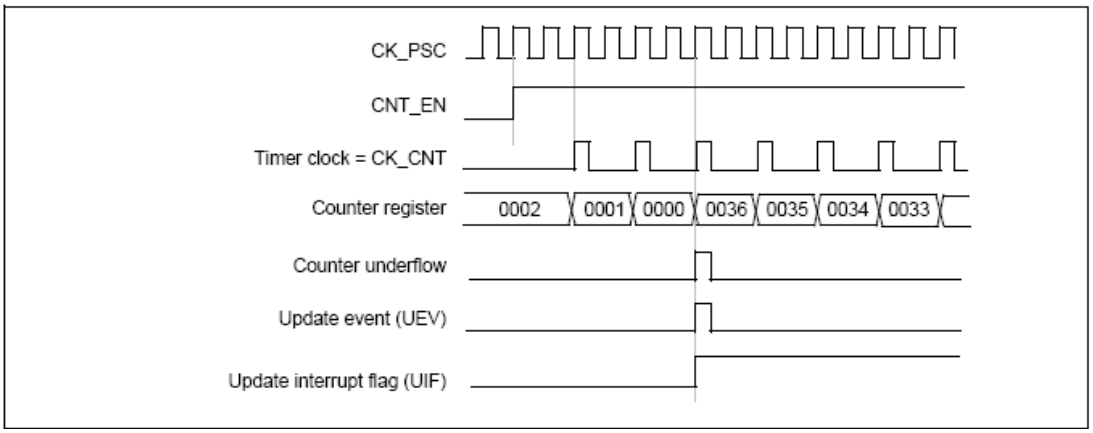


图36　计数器时序图，内部时钟分频因子为 4

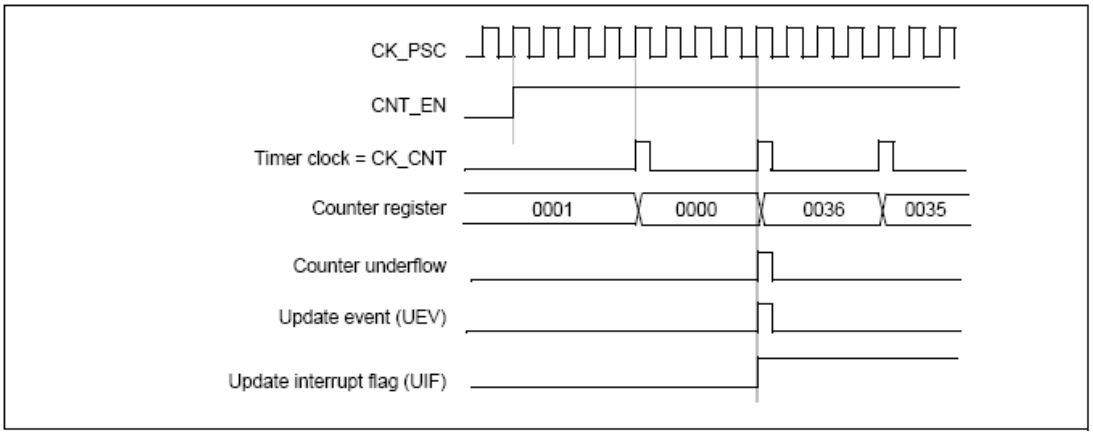


图37 计数器时序图，内部时钟分频因子为 N

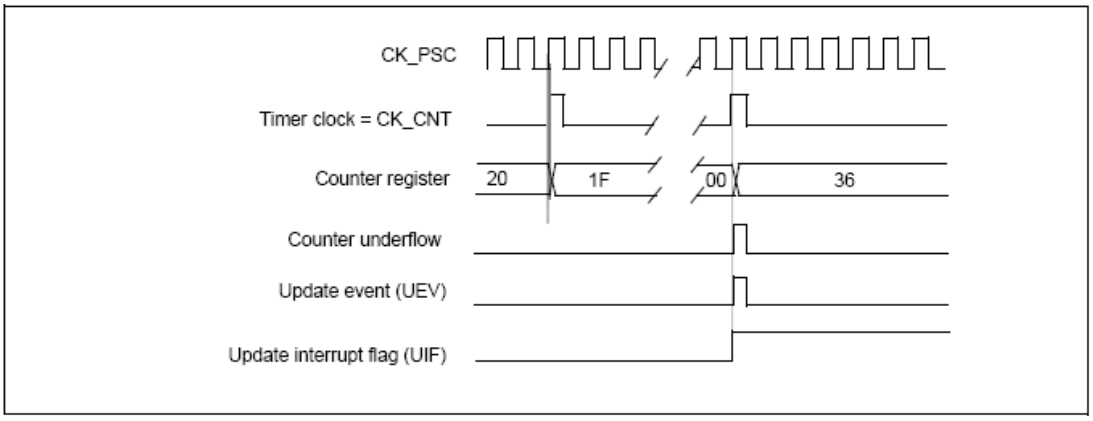
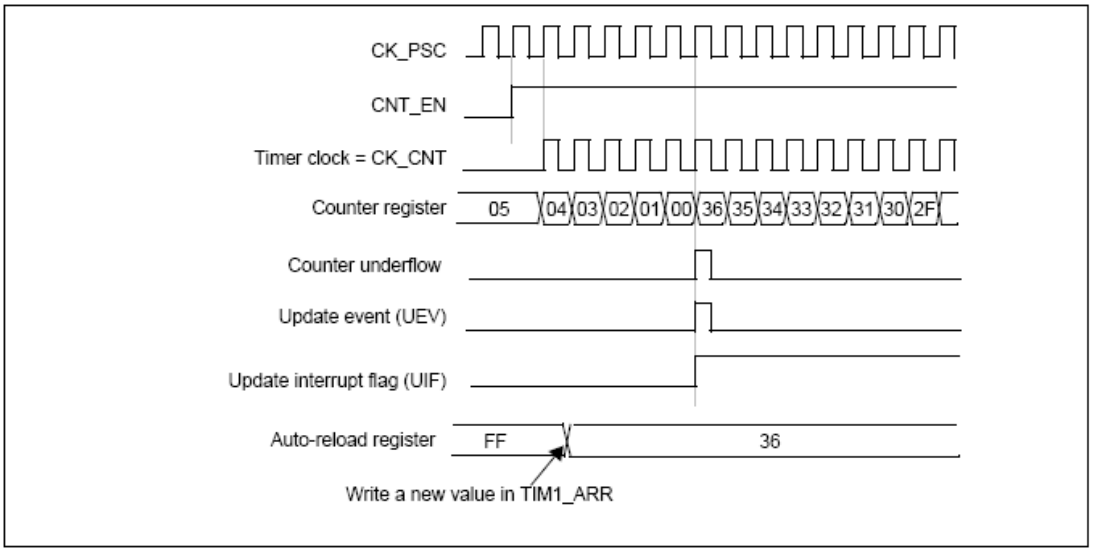


图38 计数器时序图，当没有使用周期计数器时的更新事件



中央对齐模式(向上/向下计数)

在中央对齐模式中，计数器从 0 开始计数到自动加载的值(TIM1_ARR 寄存器的内容)-1，产生一个计数器溢出事件，然后向下计数到 1 并且产生一个计数器下溢事件；然后再从 0 开始重新计数。

在这个模式下，不能写入 TIM1_CR1 中的 DIR 方向位。它由硬件更新并指示当前的计数方向。

更新事件可以产生在每一次计数溢出和每一次计数下溢；也可以通过(软件或者使用从模式控制器)设置 TIM1_EGR 寄存器中的 UG 位来产生更新事件，此时，计数器重新从 0 开始计数，预分频器也重新从 0 开始计数。

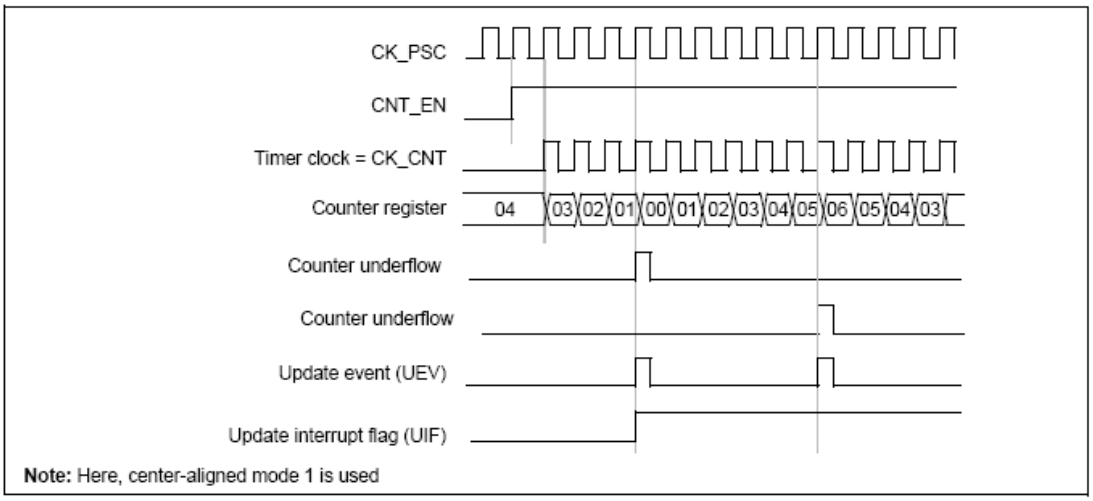
UEV 事件可以通过软件设置 TIM1_CR1 寄存器中的 UDIS 位被禁止。这样可以避免在向预装载寄存器中写入新值时更新影子寄存器。这样 UDIS 位被写成 0 之前不会产生更新事件。然而，计数器仍会根据当前自动重加载的值，继续向上或向下计数。
此外，如果设置了 TIM1_CR1 寄存器中的 URS 位(更新请求选择) ，设置 UG 位将产生一个更新事件 UEV 但不设置 UIF 标志(因此不产生中断和 DMA 请求)，这是为了避免在发生捕获事件并清除计数器时，同时产生更新和捕获中断。

当发生更新事件时，所有的寄存器都被更新，并且(根据 URS 位的设置)更新标志位(TIM1_SR 寄存器中的 UIF 位)也被设置。

- 周期计数器被重置为 TIM1_RCR 寄存器中的内容
- 当前的自动加载寄存器被更新为预装载值(TIM1_ARR 寄存器中的内容)。注：如果因为计数器溢出而产生更新，自动重装载将在计数器重载入之前被更新，因此下一个周期将是预期的值(计数器被装载为新的值)。

以下的图显示一些计数器在不同时钟频率下的操作的例子：

图39 计数器时序图，内部时钟分频因子为 1，TIM1_ARR=0x6



Note: Here, center-aligned mode 1 is used

图40 计数器时序图，内部时钟分频因子为 2

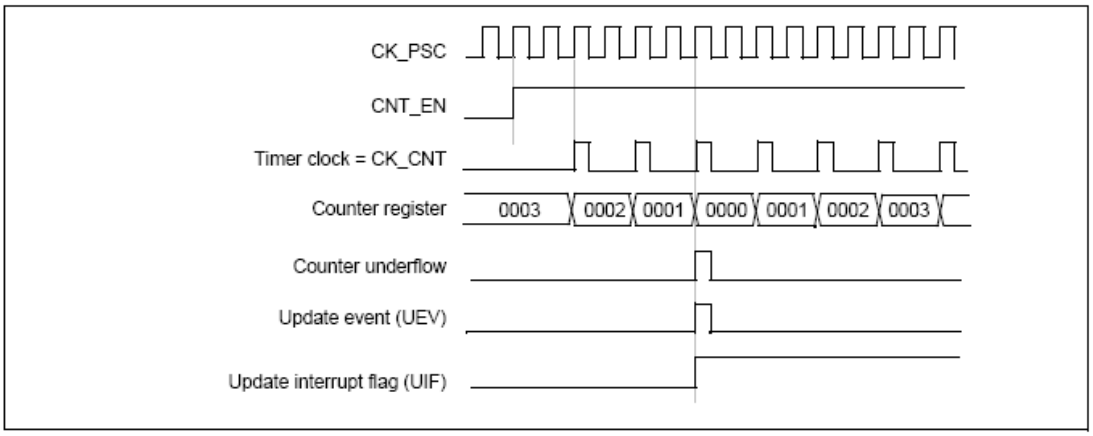


图41 计数器时序图，内部时钟分频因子为 4，TIM1_ARR=0x36

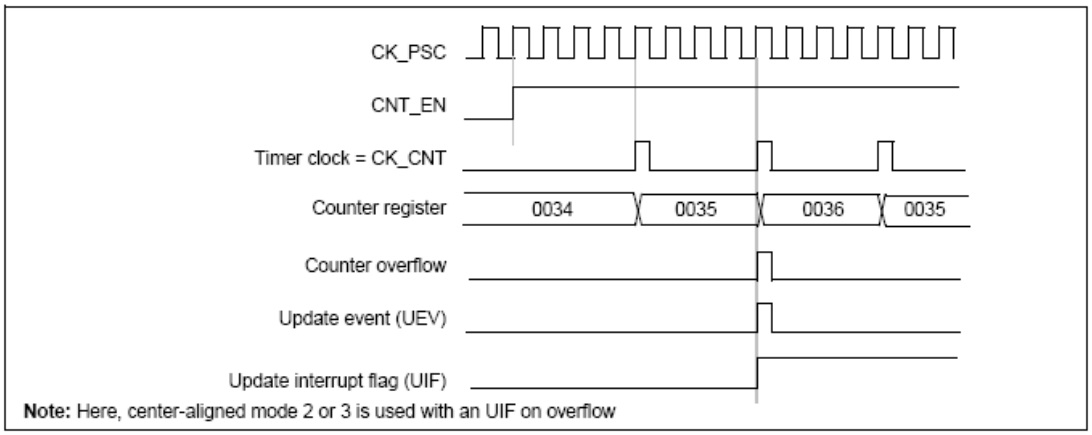


图42 计数器时序图，内部时钟分频因子为 N

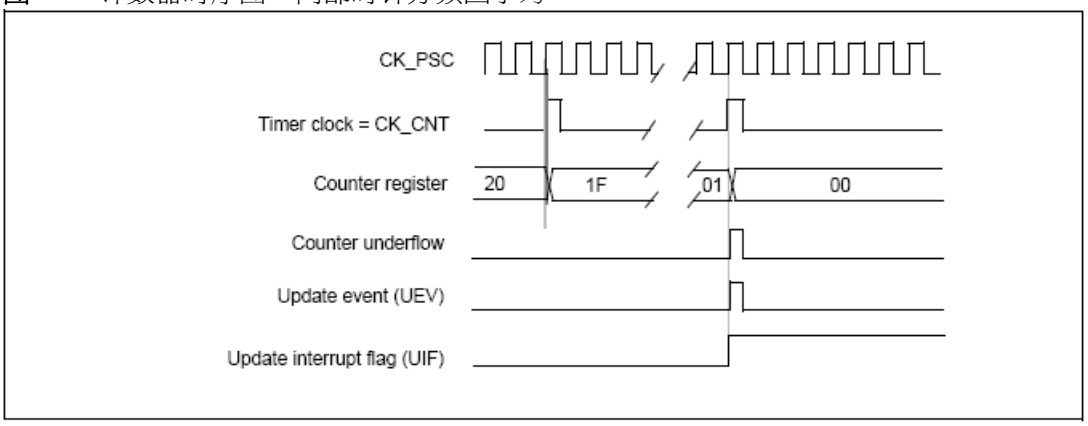


图43 计数器时序图，ARPE=1 时的更新事件(计数器下溢)

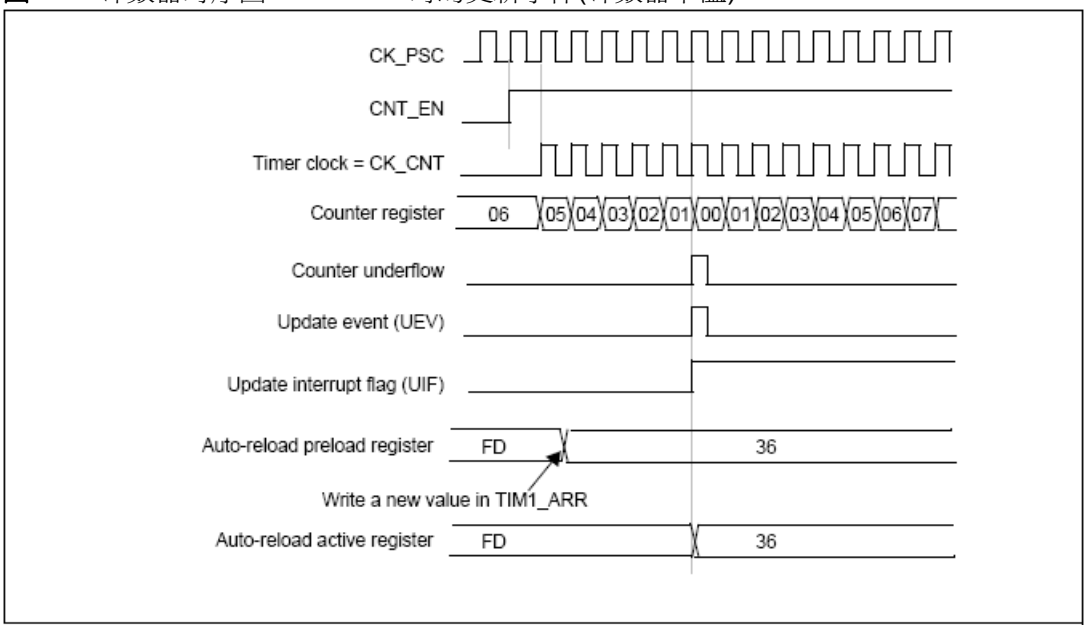
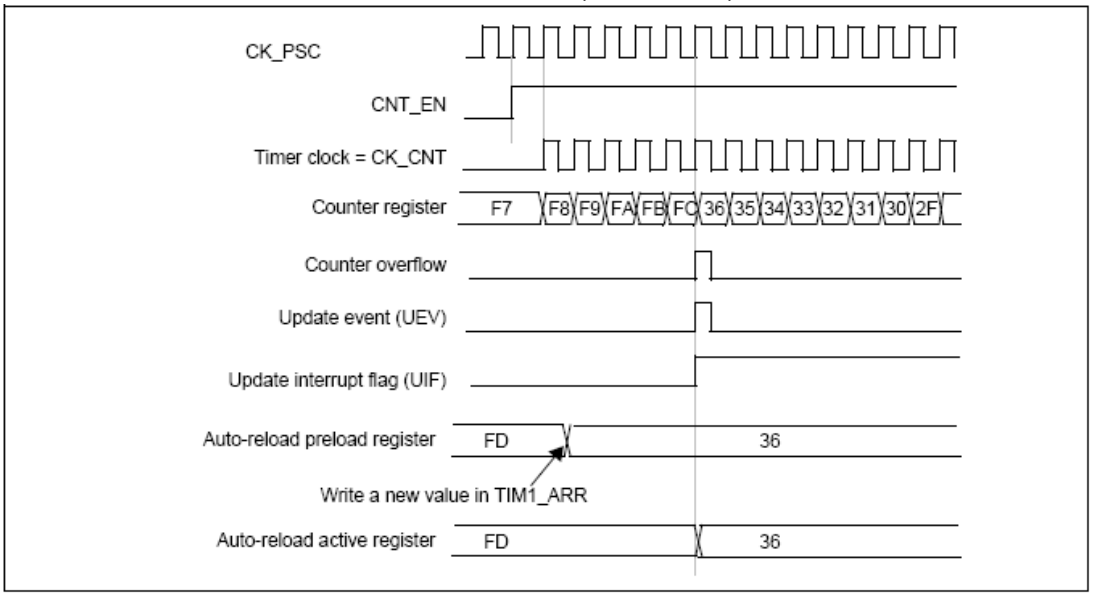


图44 计数器时序图，ARPE=1 时的更新事件(计数器溢出)



12.4.3 重复向下计数器

12.4.1时基单元解释了计数器溢出/下溢时更新事件(UEV)是如何产生的，然而事实上它只能在重复向下计数达到 0 的时候产生。这对于能产生PWM信号非常有用。

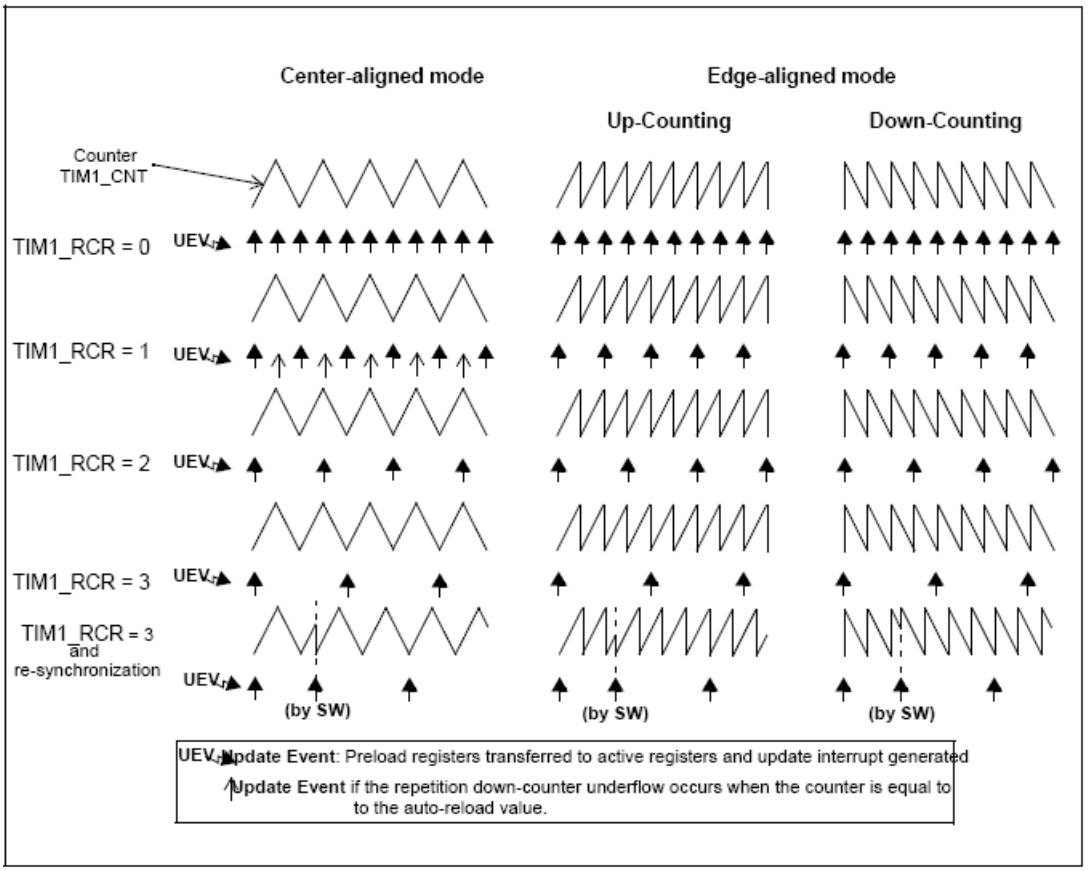
这意味着在每 N 次计数溢出或下溢时，数据从预装载寄存器传输到影子寄存器(TIM1_ARR 自动重载入寄存器，TIM1_PSC 预装载寄存器，还有在比较模式下的捕获/比较寄存器)，N 是 TIM1_RCR 周期计数寄存器中的值。

重复向下计数器在下述任一条件成立时递减：

- 向上计数模式下每次计数器溢出时。
- 向下计数模式下每次计数器下溢时。
- 中央对齐模式下每次溢出和下溢时。虽然这样限制了 PWM 的最大循环周期为 128，但它能够在每个 PWM 周期 2 次更新占空比。在中央对齐模式下，因为波形是对称的，如果每个 PWM 周期中仅刷新一次比较寄存器，则最大的分辨率为 $2xT_{ck}$。

重复向下计数器是自动加载的，重复速率是由TIM1_RCR寄存器的值定义(参看 图45图 45)。当更新事件由软件产生(通过设置TIM1_EGR 中的UG位)或者通过硬件的从模式控制器产生，则无论重复向下计数器的值时多少，立即发生更新事件，并且TIM1_RCR寄存器中的内容被重载入到重复向下计数器

图45 不同模式下更新速率的例子，及 TIM1_RCR 的寄存器设置



12.4.4 时钟选择

计数器时钟可由下列时钟源提供：

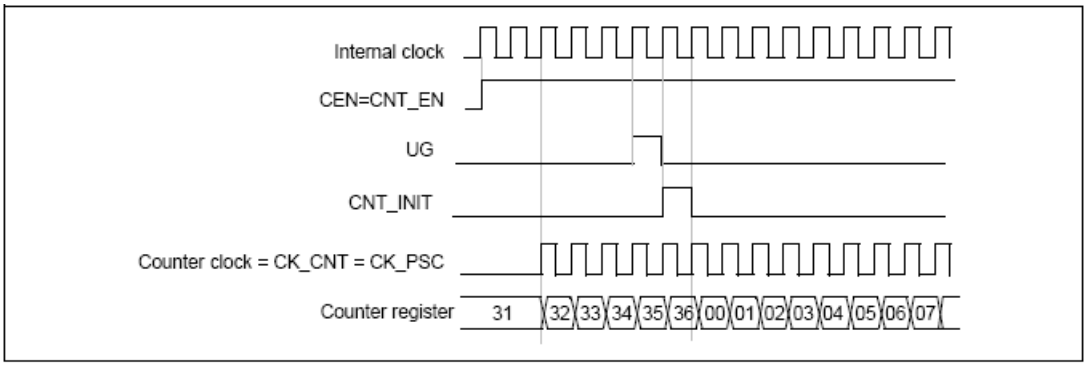
- 内部时钟(CK_INT)
- 外部时钟模式 1：外部输入脚
- 外部时钟模式 2：外部触发输入 ETR
- 内部触发输入(ITRx)：使用一个定时器作为另一个定时器的预分频器，例如你可以配置一个定时器 Timer1 而作为另一个定时器 Timer2 的预分频器。详见下一章。

内部时钟源(CK_INT)

如果从模式控制器被禁止(SMS=000)，则 CEN、DIR(TIM1_CR1 寄存器中)和 UG 位(TIM1_EGR 寄存器中)是事实上的控制位同时只能被软件修改(UG 位仍被自动清除)。一旦 CEN 位被写成 1，预分频器的时钟就由内部时钟 CK_INT 提供。

图 46图 46 显示了控制电路和向上计数器在一般模式下，不带预分频器时的操作。

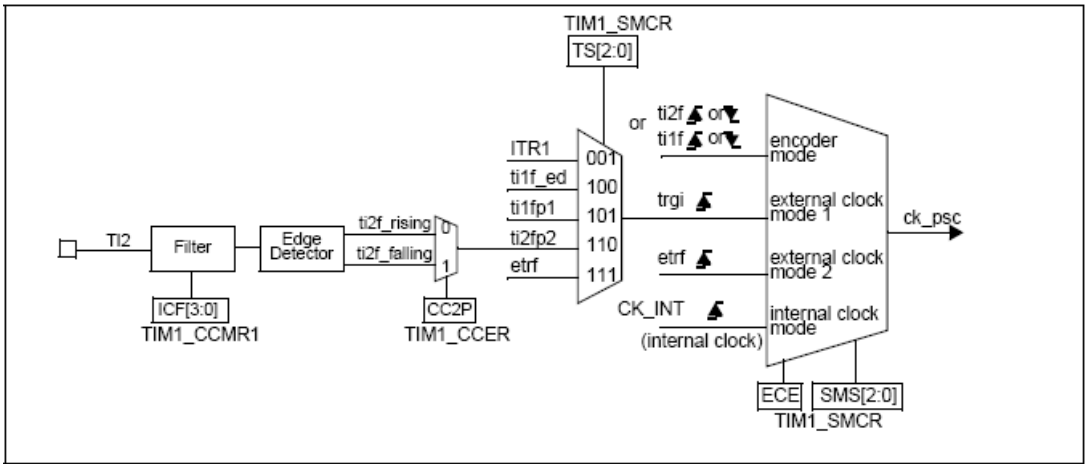
图46 一般模式下的控制电路，内部时钟分频因子为 1



外部时钟源模式 1

当 TIM1_SMCR 寄存器中的 SMS=111 时，此模式被选中。计数器可以在选定的输入上每个上升沿或下降沿计数。

图47 TI2 外部时钟连接例子



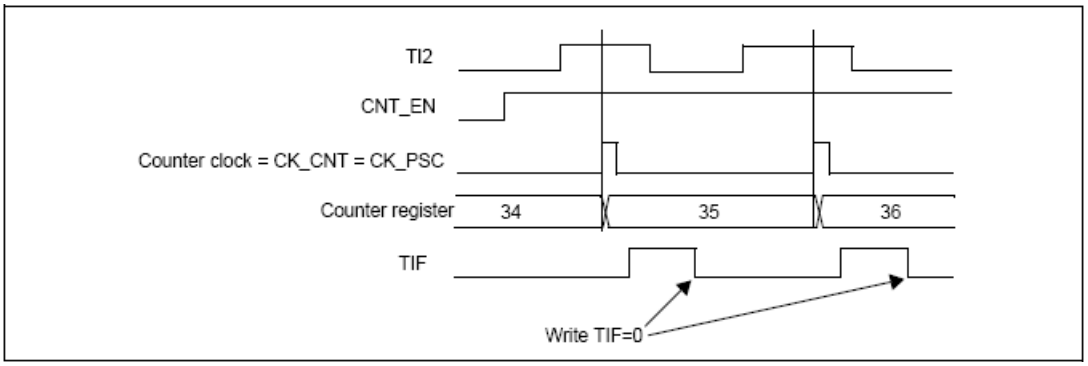
例如，要配置向上计数器在 T12 输入端的上升沿计数，使用下列步骤：

1. 配置TIM1_CCMR1寄存器CC2S=01，配置通道2检测TI2输入的上升沿
2. 配置TIM1_CCMR1寄存器的IC2F[3:0]，选择输入滤波器带宽(如果不需要滤波器，保持IC2F=0000)
3. 配置TIM1_CCER寄存器的CC2P=0，选定上升沿极性
4. 配置TIM1_SMCR寄存器的SMS=111，选择定时器外部时钟模式1
5. 配置TIM1_SMCR寄存器中的TS=110，选定TI2作为触发输入源
6. 设置TIM1_CR1寄存器的CEN=1，启动计数器

注： *捕获预分频器不用作触发，所以不需要对它进行配置*

当上升沿出现在 TI2，计数器计数一次，且 TIF 标志被设置。
在 TI2 的上升沿和计数器实际时钟之间的延时取决于在 TI2 输入端的重新同步电路。

图48 外部时钟模式 1 下的控制电路

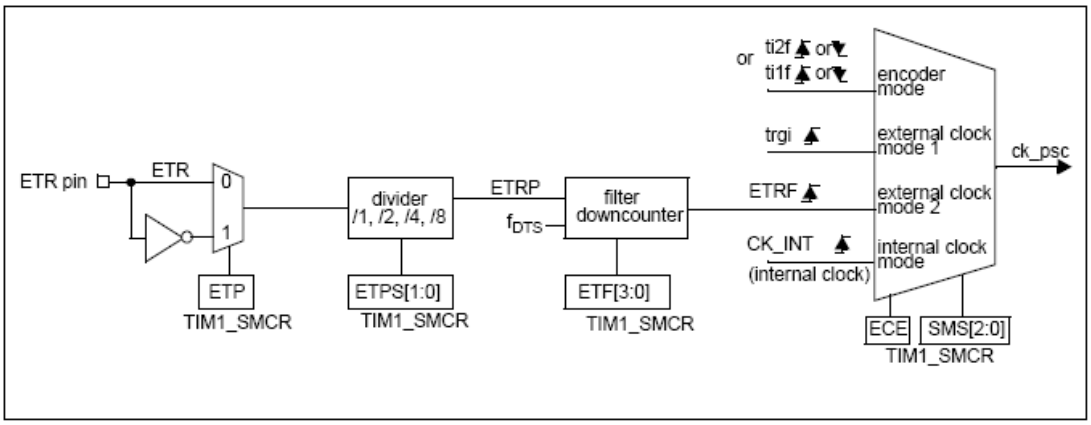


外部时钟源模式 2

选定此模式的方法为：令 TIM1_SMCR 寄存器中的 ECE=1

计数器能够在外部触发 ETR 的每一个上升沿或下降沿计数。
图 49图 49 是外部触发输入的总体框图

图49 外部触发输入框图



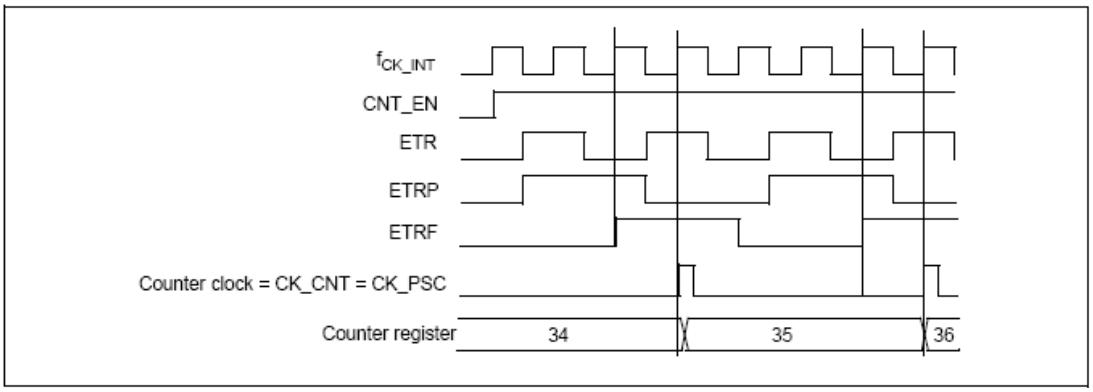
例如，要配置在 ETR 下每 2 个上升沿计数一次的向上计数器，使用下列步骤：

1. 本例中不需要滤波器，写TIM1_SMCR寄存器中的ETF[3:0]=0000
2. 设置预分频器，写TIM1_SMCR寄存器中的ETPS[1:0]=01
3. 设置在ETR下的上升沿检测，写TIM1_SMCR寄存器中的ETP=0
4. 开启外部时钟模式2，写TIM1_SMCR寄存器中的ECE=1
5. 启动计数器，写TIM1_CR1寄存器中的CEN=1

计数器在每 2 个 ETR 上升沿计数一次。

在 ETR 的上升沿和计数器实际时钟之间的延时取决于在 ETRP 信号端的重新同步电路。

图50　外部时钟模式 2 下的控制电路



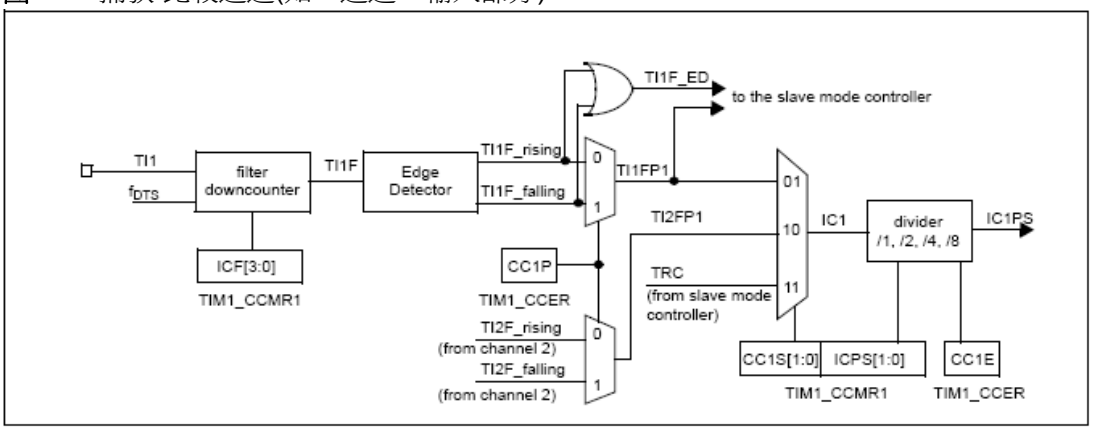
12.4.5　捕获/比较通道

每一个捕获/比较通道是围绕着一个捕获/比较寄存器(包含影子寄存器)，包括捕获的输入部分(包含数字滤波、多路复用和预分频器)，和输出部分(包含比较器和输出控制)。

图 51 和 图 54 是一个捕获/比较通道概览。

输入部分对相应的 TIx 输入信号采样，并产生一个滤波后的信号 TIxF。然后，一个带极性选择的边缘监测器产生一个信号(TIxFPx)，它可以作为从模式控制器的输入触发或者作为捕获控制。该信号在捕获寄存器(ICxPS)之前已被整形。

图51　捕获/比较通道(如：通道 1 输入部分)



输出部分产生一个中间波形 OCxRef(高有效)作为基准，链的末端信号决定最终的输出极性。

图52　捕获/比较通道 1 的主电路

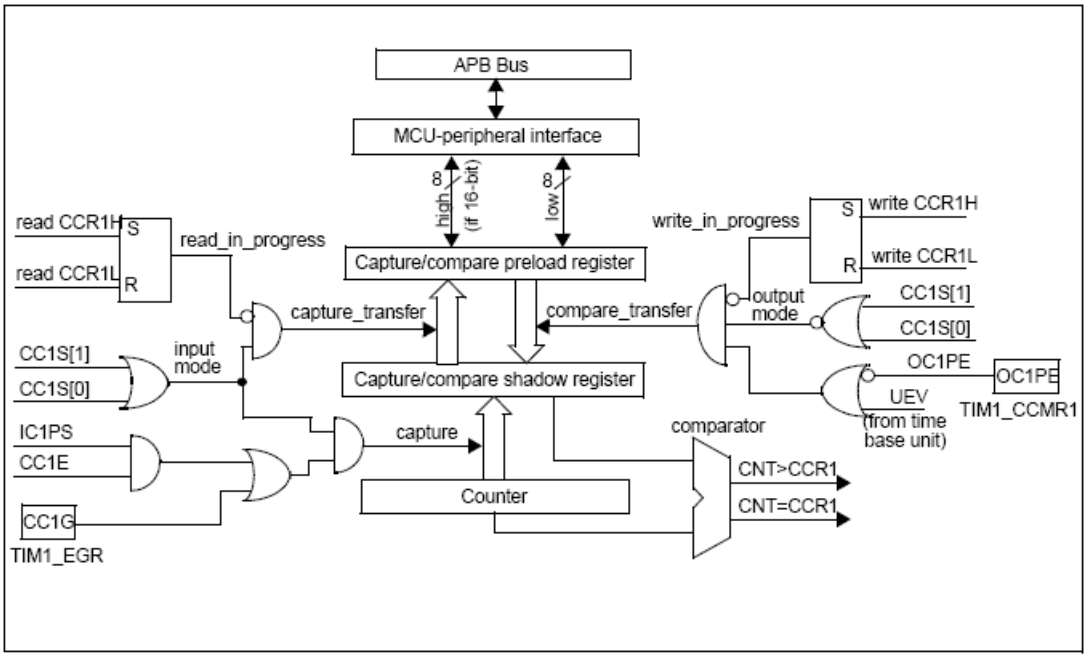


图53　捕获/比较通道的输出部分(通道 1 至 3)

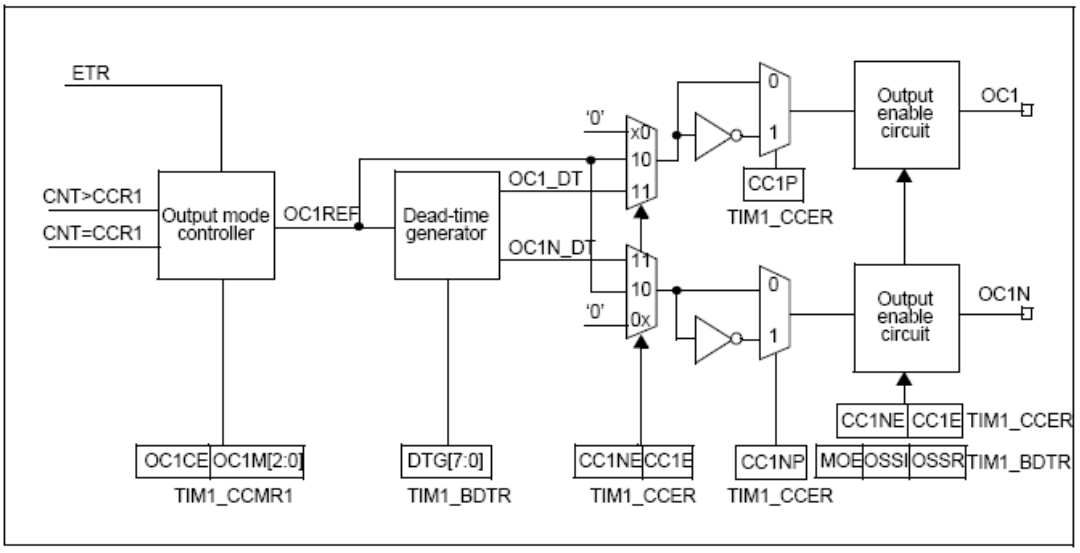
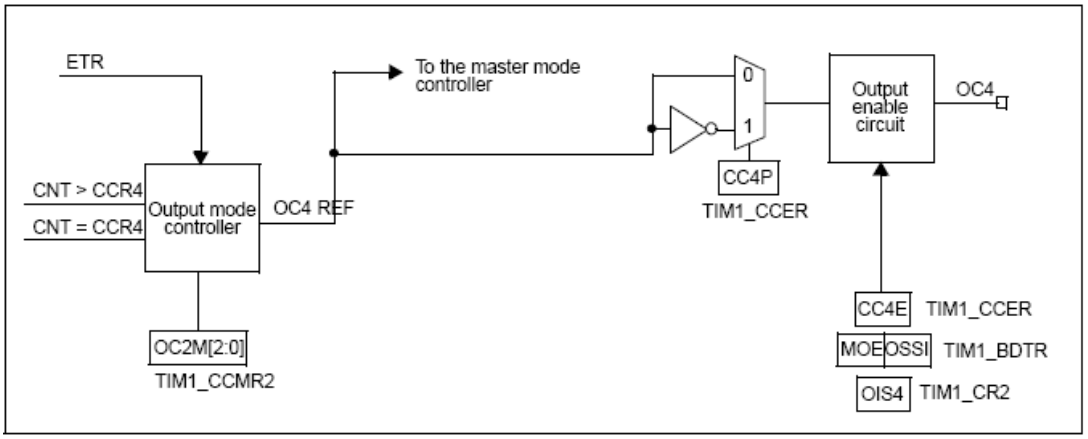


图54　捕获/比较通道的输出部分(通道 4)



捕获/比较模块由一个预装载寄存器和一个影子寄存器组成。读写过程仅操作预装载寄存器。

在捕获模式下，捕获发生在影子寄存器上，然后再复制到预装载寄存器中。

在比较模式下，预装载寄存器的内容被复制到影子寄存器中，然后和计数器进行比较。

12.4.6 输入捕获模式

在输入捕获模式下，当检测到 ICx 信号上相应的边沿后，捕获/比较寄存器(TIM1_CCRx)被用来锁存计数器的值。当一个捕获事件发生时，相应的 CCXIF 标志(TIM1_SR 寄存器)被置 1，如果开放了中断或者 DMA 操作，则将产生中断或者 DMA 操作。如果一个捕获事件发生时 CCxIF 标志已经为高，那么重复捕获标志 CCxOF(TIM1_SR 寄存器)被置 1。写 CCxIF=0 可清除 CCxIF，或读取存储在 TIM1_CCRx 寄存器中的捕获数据也可清除 CCxIF。写 CCxOF=0 可清除 CCxOF。

以下例子说明如何在 TI1 输入的上升沿时捕获计数器的值到 TIM1_CCR1 寄存器中，步骤如下：

- 选择有效输入端：TIM1_CCR1 必须连接到 TI1 输入，所以写入 TIM1_CCR1 寄存器中的 CC1S=01，一旦 CC1S 不为 00 时，通道被配置为输入，并且 TM1_CCR1 寄存器变为只读。
- 根据输入信号的特点，配置输入滤波器为所需的带宽(输入为 TIx 时，TIM1_CCMRx 寄存器中的 ICxF 位)。假设输入信号在最多 5 个时钟周期的时间内抖动，我们须配置滤波器的带宽长于 5 个时钟周期。因此我们可以(以 fDTS 频率)连续采样 8 次，以确认在 TI1 上一次真实的边沿变换，即在 TIM1_CCMR1 寄存器中写入 IC1F=0011。
- 选择 TI1 通道的有效转换边沿，在 TIM1_CCER 寄存器中写入 CC1P=0(即上升沿)。
- 配置输入预分频器。在本例中，我们希望捕获发生在每一个有效的电平转换时刻，因此预分频器被禁止(写 TIM1_CCMR1 寄存器的 IC1PS=00)。
- 设置 TIM1_CCER 寄存器的 CC1E=1，允许捕获计数器的值到捕获寄存器中。
- 如果需要，通过设置 TIM1_DIER 寄存器中的 CC1IE 位允许相关中断请求，通过设置 TIM1_DIER 寄存器中的 CC1DE 位允许 DMA 请求。

发生当一个输入捕获时：

- 当产生有效的电平转换时，计数器的值被传送到 TIM1_CCR1 寄存器。
- CC1IF 标志被设置(中断标志)。当发生至少 2 个连续的捕获时，而 CC1IF 未曾被清除，CC1OF 也被置 1。
- 如设置了 CC1IE 位，则会产生一个中断。
- 如设置了 CC1DE 位，则还会产生一个 DMA 请求。
-

为了处理捕获溢出，建议在读出捕获溢出标志之前读取数据，这是为了避免丢失在读出捕获溢出标志之后和读取数据之前可能产生的捕获溢出信息。

注： 设置 TIM1_EGR 寄存器中相应的 CCxG 位，可以通过软件产生输入捕获中断和/或 DMA 请求。

12.4.7 PWM输入模式

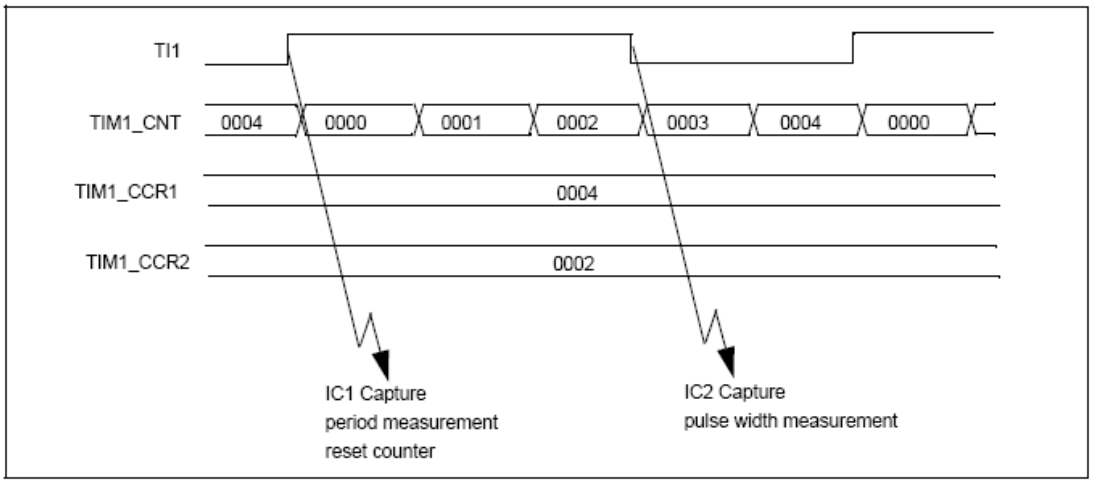
该模式是输入捕获模式的一个特例，除下列区别外，工作过程与输入捕获模式相同：

- 两个 ICx 信号被映射同一个 TIx 输入。
- 这 2 个 ICx 信号为边沿有效，但是极性相反。
- 其中一个 TIxFP 信号被作为触发输入信号，并且从模式控制器被配置成复位模式。

例如，你能够测量输入到 TI1 上的 PWM 信号的长度(TIM1_CCR1 寄存器)和占空比(TIM1_CCR2 寄存器)，具体步骤如下(取决于 CK_INT 的频率和预分频器的值)

- 选择 TIM1_CCR1 的有效输入端：置 TIM1_CCMR1 寄存器的 CC1S=01(选择 TI1)；
- 选择 TI1FP1 的有效极性(用来捕获数据到 TIM1_CCR1 中和清除计数器)：置 CC1P=0；
- 选择 TIM1_CCR2 的有效输入端：置 TIM1_CCMR1 寄存器的 CC2S=10(选择 TI1)；
- 选择 TI1FP2 的有效极性(用来捕获数据到 TIM1_CCR2)：置 CC2P=1(下降沿有效)；
- 选择有效的触发输入信号：置 TIM1_SMCR 寄存器中的 TS=101(选择 TI1FP1)；
- 配置从模式控制器为复位模式：置 TIM1_SMCR 中的 SMS=100；
- 使能捕获：置 TIM1_CCER 寄存器中 CC1E=1 且 CC2E=1。

图55 PWM 输入模式时序



12.4.8 强置输出模式

在输出模式(TIM1_CCMRx 寄存器中 CCxS=00)下，输出比较信号(OCxREF 和相应的 OCx/OCxN)能够直接由软件强置为有效或无效状态，而不依赖于输出比较寄存器和计数器间的比较结果。

置 TIM1_CCMRx 寄存器中相应的 OCxM=101，即可强置输出比较信号(OCxREF/OCx)为有效状态。这样 OCxREF 被强置为高电平(OCxREF 始终为高电平有效)，同时 OCx 得到 CCxP 极性位相反的值。

例如：CCxP=0(OCx 高电平有效)，则 OCx 被强置为高电平。

置 TIM1_CCMRx 寄存器中的 OCxM=100，可强置 OCxREF 信号为低。

该模式下，在 TIM1_CCRx 影子寄存器和计数器之间的比较仍然在进行，相应的标志也会被修改。因此仍然会产生相应的中断和 DMA 请求。这将会在下面的输出比较模式一节中介绍。

12.4.9 输出比较模式

此项功能是用来控制一个输出波形或者指示何时一段给定的的时间已经到时。

当计数器与捕获/比较寄存器的内容相同时，输出比较功能做如下操作：

- 将输出比较模式(TIM1_CCMRx 寄存器中的 OCxM 位)和输出极性(TIM1_CCER 寄存器中的 CCxP 位)定义的值输出到对应的管脚上。在比较匹配时，输出管脚可以保持它的电平(OCxM=011)、被设置成有效电平(OCxM=001)、被设置成无有效电平(OCxM=010)或进行翻转(OCxM=011)。
- 设置中断状态寄存器中的标志位(TIM1_SR 寄存器中的 CCxIF 位)。
- 如果设置了相应的中断屏蔽(TIM1_DIER 寄存器中的 CCXIE 位)，则产生一个中断。

- 若设置了相应的使能位(TIM1_DIER 寄存器中的 CCxDE 位，TIM1_CR2 寄存器中的 CCDS 位选择 DMA 请求功能)，则产生一个 DMA 请求。

TIM1_CCMRx 中的 OCxPE 位用于选择 TIM1_CCRx 寄存器是否需要使用预装载寄存器。

在输出比较模式下，更新事件 UEV 对 OCxREF 和 OCx 输出没有影响。

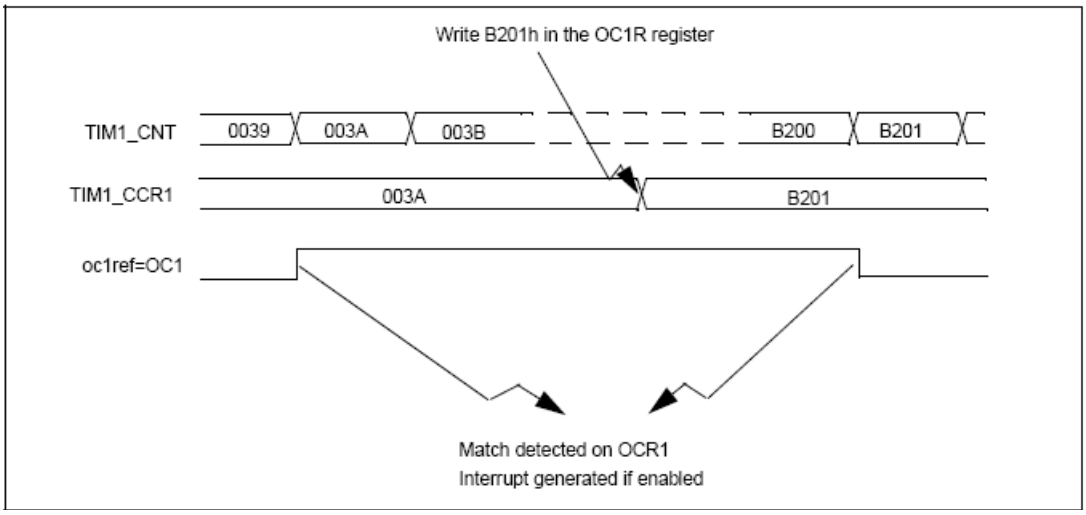
同步的精度到达计数器的一个计数周期。输出比较模式(在单脉冲模式下)也能用来输出一个单脉冲。

输出比较模式的配置步骤：

1. 选择计数器时钟(内部，外部，预分频器)
2. 将相应的数据写入TIM1_ARR和TIM1_CCRx寄存器中
3. 如果要产生一个中断请求，设置CCxIE位。
4. 选择输出模式，例如：
   a) 要求CNT与CCRx匹配时翻转OCx的输出管脚，设置OCxM=011
   b) 置OCxPE = 0禁用预装载寄存器
   c) 置CCxP = 0选择高电平有效极性
   d) 置CCxE = 1使能输出
5. 设置TIM1_CR1寄存器的CEN位启动计数器

TIM1_CCRx寄存器能够在任何时候通过软件进行更新以控制输出波形，条件是未使用预装载寄存器(OCxPE='0'，否则TIM1_CCRx影子寄存器只能在下一次更新事件发生时被更新)。图 56 给出了一个例子。

图56 输出比较模式，翻转 OC1



12.4.10 PWM 模式

脉冲宽度调制模式可以产生一个由 TIM1_ARR 寄存器确定频率、由 TIM1_CCRx 寄存器确定占空比的信号。

在 TIM1_CCMRx 寄存器中的 OCxM 位写入“110”(PWM 模式 1)或“111”(PWM 模式 2)，能够独立地设置每个通道工作在 PWM 模式，每个 OCx 输出一路 PWM。必须通过设置 TIM1_CCMRx 寄存器 OCxPE 位使能相应的预装载寄存器，最后还要设置 TIM1_CR1 寄存器的 ARPE 位使能自动重装载的预装载寄存器(在向上计数或中心对称模式中)。

因为仅当发生一个更新事件的时候，预装载寄存器才能被传送到影子寄存器，因此在计数器开始计数之前，必须通过设置 TIM1_EGR 寄存器中的 UG 位来初始化所有的寄存器。

OCx 的极性可以通过软件在 TIM1_CCER 寄存器中的 CCxP 位设置，它可以设置为高电平有效活和低电平有效。OCx 输出通过 CCxE、CCxNE、MOE、OSSI 和 OSSR 位(在 TIM1_CCER 和 TIM1_BDTR 寄存器中)的组合控制。详见 TIM1_CCER 寄存器的描述。

在 PWM 模式(模式 1 或模式 2)下，TIM1_CNT 和 TIM1_CCRx 始终在进行比较，(依据计数器的计数方向)以确定是否符合 TIM1_CCRx≤TIM1_CNT 或者 TIM1_CNT≤TIM1_CCRx。

根据 TIM1_CR1 寄存器中 CMS 位的状态，定时器能够产生边沿对齐的或中央对齐的 PWM 信号。

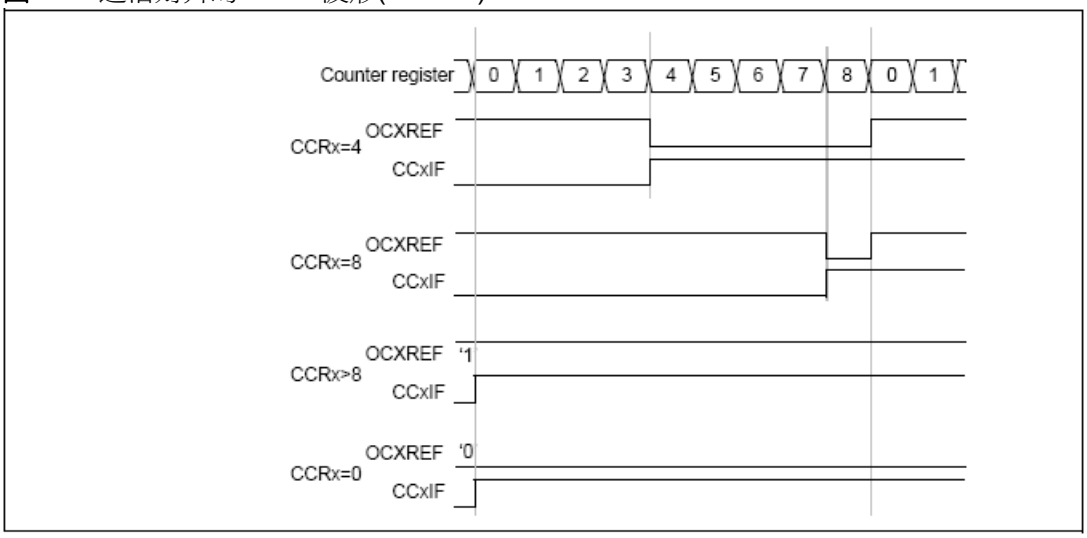
PWM 边沿对齐模式

- 向上计数配置

当TIM1_CR1 寄存器中的DIR位为低的时候执行向上计数。参看 12.4.2 节。

下面是一个PWM模式 1 的例子。当TIM1_CNT<TIM1_CCRx时PWM参考信号OCxREF为高，否则为低。如果TIM1_CCRx中的比较值大于自动重装载值(TIM1_ARR)，则OCxREF保持为“1”。如果比较值为 0，则OCxREF保持为“0”。 图 57 为TIM1_ARR=8 时边沿对齐的PWM波形实例。

图57 边沿对齐的 PWM 波形(ARR=8)



- 向下计数的配置

当TIM1_CR1 寄存器的DIR位为高时执行向下计数。参看 12.4.2 节。

在 PWM 模式 1，当 TIM1_CNT>TIM1_CCRx 时参考信号 OCxREF 为低，否则为高。如果 TIM1_CCRx 中的比较值大于 TIM1_ARR 中的自动重装载值，则 OCxREF 保持为 “1” 。该模式下不能产生 0%的 PWM 波形。

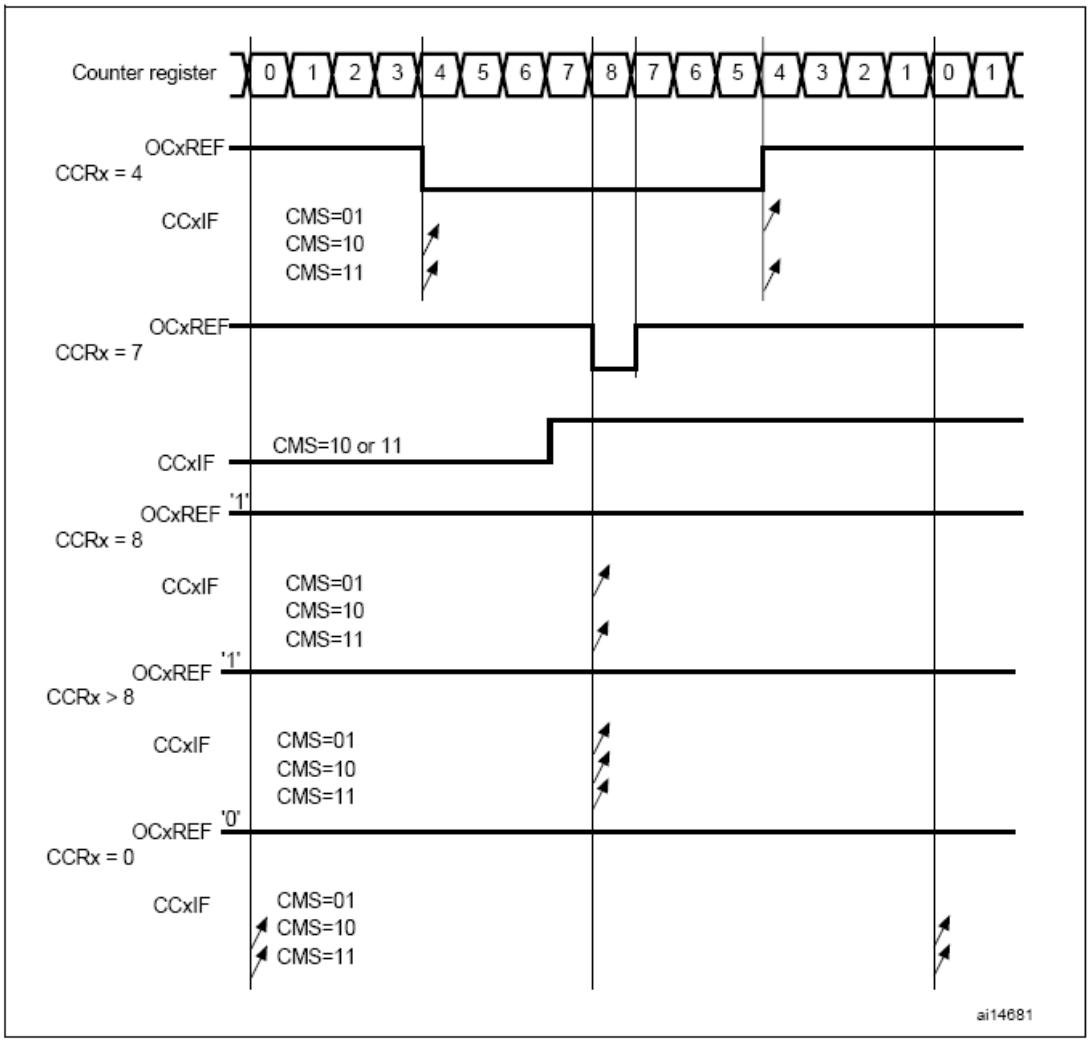
PWM 中央对齐模式

当TIM1_CR1 寄存器中的CMS位不为 00 时为中央对齐模式(所有其他的配置对 OCxREF/OCx信号都有相同的作用)。根据不同的CMS位的设置，比较标志可能在计数器向上计数时被置 1、在计数器向下计数时被置 1、或在计数器向上和向下计数时被置 1。TIM1_CR1 寄存器中的计数方向位(DIR)由硬件更新，不要用软件修改它。参看 12.4.2 节的中央对齐模式。

图 58 给出了一些中央对齐的PWM波形的例子

- TIM1_ARR=8
- PWM 模式 1
- TIM1_CR1 寄存器中的 CMS=01，在中央对齐模式 1 时，当计数器向下计数时标志被设置

图58 中央对齐的 PWM 波形(APR=8)



使用中央对齐模式的提示

- 进入中央对齐模式时，当前的上-下配置被使用；这就意味着计数器向上还是向下计数取决于 TIM1_CR1 寄存器中 DIR 位的当前值。此外，DIR 和 CMS 位不能同时被软件修改。
- 不推荐当运行在中央对齐模式时改写计数器，因为会产生不可预知的结果。特别地：
  - 如果写入计数器的值大于自动重加载的值(TIM1_CNT>TIM1_ARR)，则方向不会被更新。例如，如果计数器正在向上计数，它就会继续向上计数。
  - 如果将 0 或者 TIM1_ARR 的值写入计数器，方向被更新，但不产生更新事件 UEV。

使用中央对齐模式最保险的方法，就是在启动计数器之前产生一个软件更新(设置 TIM1_EGR 位中的 UG 位)，不要在计数进行过程中修改计数器的值。

12.4.11 互补输出和死区插入

高级控制定时器 TIM1 能够输出两路互补信号并且能够管理输出的瞬时关断和接通。

这段时间通常被称为死区，你应该根据连接到输出的器件和它们的特性(电平转换的延时、电源开关的延时等)来调整死区时间。

配置 TIM1_CCER 寄存器中的 CCxP 和 CCxNP 位，可以为每一个输出独立地选择极性(主输出 OCx 或互补输出 OCxN)。

互补信号OCx和OCxN通过下列控制位的组合进行控制：TIM1_CCER寄存器的CCxE和CCxNE位，TIM1_BDTR和TIM1_CR2 寄存器中的MOE、OISx、OISxN、OSSI和OSSR位，详见 表 37带刹车功能的互补输出通道OCx和OCxN的控制位。特别的是，在转换到IDLE状态时(MOS下降到 0)死区被激活。

同时设置 CCxE 和 CCxNE 位将插入死区，如果存在刹车电路，则还要设置 MOE 位。每一个通道都有一个 10 位的死区发生器。参考信号 OCxREF 可以产生 2 路输出 OCx 和 OCxN。如果 OCx 和 OCxN 为高有效：

- OCx 输出信号与参考信号相同，只是它的上升沿相对于参考信号的上升沿有一个延迟。
- OCxN 输出信号与参考信号相反，只是它的上升沿相对于参考信号的下降沿有一个延迟。

如果延迟大于当前有效的输出宽度(OCx 或者 OCxN)，则不会产生相应的脉冲。

下列几张图显示了死区发生器的输出信号和当前参考信号 OCxREF 之间的关系。(假设 CCxP=0、CCxNP=0、MOE=1、CCxE=1 并且 CCxNE=1)

图59 带死区插入的互补输出

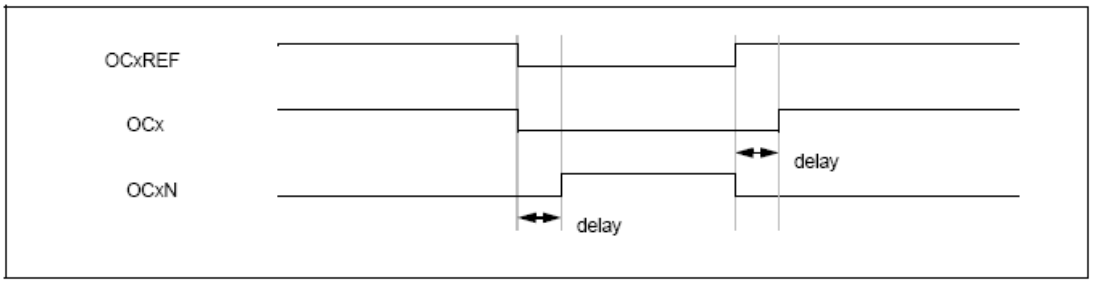


图60 死区波形延迟大于负脉冲

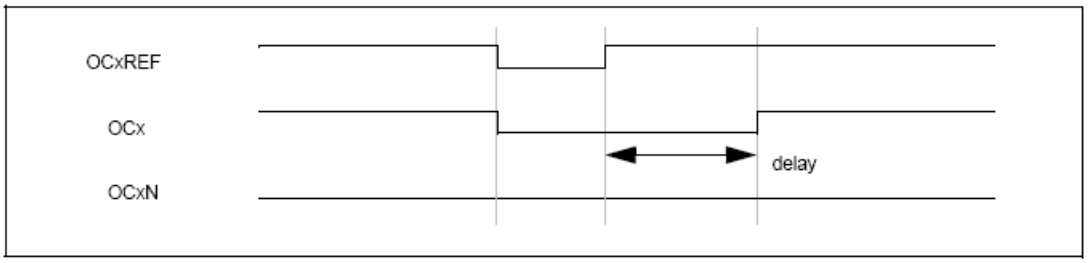
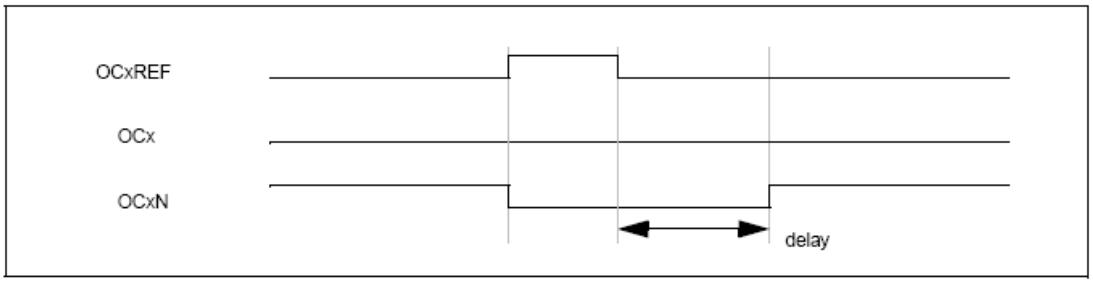


图61 死区波形延迟大于正脉冲



每一个通道的死区延时都是相同的，是由TIM1_BDTR寄存器中的DTG位编程配置。详见 12.5.18 节中的延时计算。

重定向OCxREF到OCx或OCxN

在输出模式下(强置、输出比较或 PWM)，OCxREF 可以被重定向到 OCx 或者 OCxN 的输出，通过配置 TIM1_CCER 寄存器的 CCxE 和 CCxNE 位实现。

这个功能可以在互补输出处于无效电平时，在某个输出上送出一个特殊的波形(例如 PWM 或者静态有效电平)。另一个作用是，让两个输出同时处于无效电平，或处于有效电平和带死区的互补输出。

注： 当只有 OCxN 被使能(CCxE=0, CCxNE=1)时，它不会反相，并当 OCxREF 有效时立即变高。例如，如果 CCxNP=0，则 OCxN=OCxREF。另一方面，当 OCx 和 OCxN 都被使能时(CCxE=CCxNE=1)，当 OCxREF 为高时 OCx 有效；而 OCxN 相反，当 OCxREF 变低时 OCxN 有效。

12.4.12 使用刹车功能

当使用刹车功能时，依据相应的控制位(TIM1_BDTR寄存器中的MOE、OSSI和OSSR位，TIM1_CR2 寄存器中的OISx和OISxN位)，输出使能信号和无效电平都会被修改。但无论何时，OCx和OCxN输出不能在同一时间同时处于有效电平上。详见 表 37带刹车功能的互补输出通道OCx和OCxN的控制位。

刹车源既可以是刹车输入管脚又可以是一个时钟失败事件，时钟失败事件是由复位时钟控制器中的时钟安全系统产生。详见第 4 章。
系统复位后，刹车电路被禁止，MOS 位为低。通过设置 TIM1_BDTR 寄存器中的 BKE 位可以使能刹车功能。刹车输入信号的极性可以通过配置同一个寄存器中的 BKP 位选择。BKE 和 BKP 可以被同时修改。

因为 MOE 下降沿可以是异步的，在实际信号(作用在输出端)和同步控制位(在 TIM1_BDTR 寄存器中)之间设置了一个再同步电路。这个再同步电路会在异步信号和同步信号之间产生延迟。特别的，如果当它为低时写 MOE=1，则读出它之前必须先插入一个延时(空指令)才能读到正确的值。这是因为你写的是异步信号而读的是同步信号。

当发生刹车时(在刹车输入端出现选定的电平)

- MOE 位被异步地清除，将输出置于无效状态、空闲状态或者复位状态(由 OSSI 位选择)。这个特性在 MCU 的振荡器关闭时依然能够实现。

- 一旦 MOE=0，每一个输出通道输出由 TIM1_CR2 寄存器中的 OISx 位设定的电平。如果 OSSI=0，定时器释放使能输出否则使能输出始终为高。
- 当使用互补输出时：
  - 输出首先被置于复位状态即无效的状态(取决于极性)。这是异步实现的，即使定时器没有时钟时，也是如此。
  - 如果定时器的时钟存在，死区生成器将会重新生效，在死区之后根据 OISx 和 OISxN 位指示的电平驱动输出端口。即使在这种情况下，OCx 和 OCxN 也不能被同时驱动到有效的电平。注，因为重新同步 MOE，死区时间比通常情况下长一些(大约 2 个 ck_tim 的时钟周期)。
  - 如果OSSI=0，定时器释放使能输出，否则保持使能输出；或一旦CCxE与CCxNE之一变高时变为高。
- 如果设置了 TIM1_SR 寄存器中的 BIF 位，当刹车状态标志(TIM1_SR 寄存器中的 BIF 位)为 1 时，则产生一个中断。如果设置了 TIM1_DIER 寄存器中的 BDE 位，则产生一个 DMA 请求。
- 如果设置了 TIM1_BDTR 寄存器中的 AOE 位，在生下一个更新事件 UEV 时 MOE 位被自动置位；例如，这可以用来进行整形。否则，MOE 始终保持低直到被再次置 1；在这种情况下，这个特性可以被用在安全方面，你可以把刹车输入连到电源驱动的报警输出、热敏传感器或者其他安全器件上。

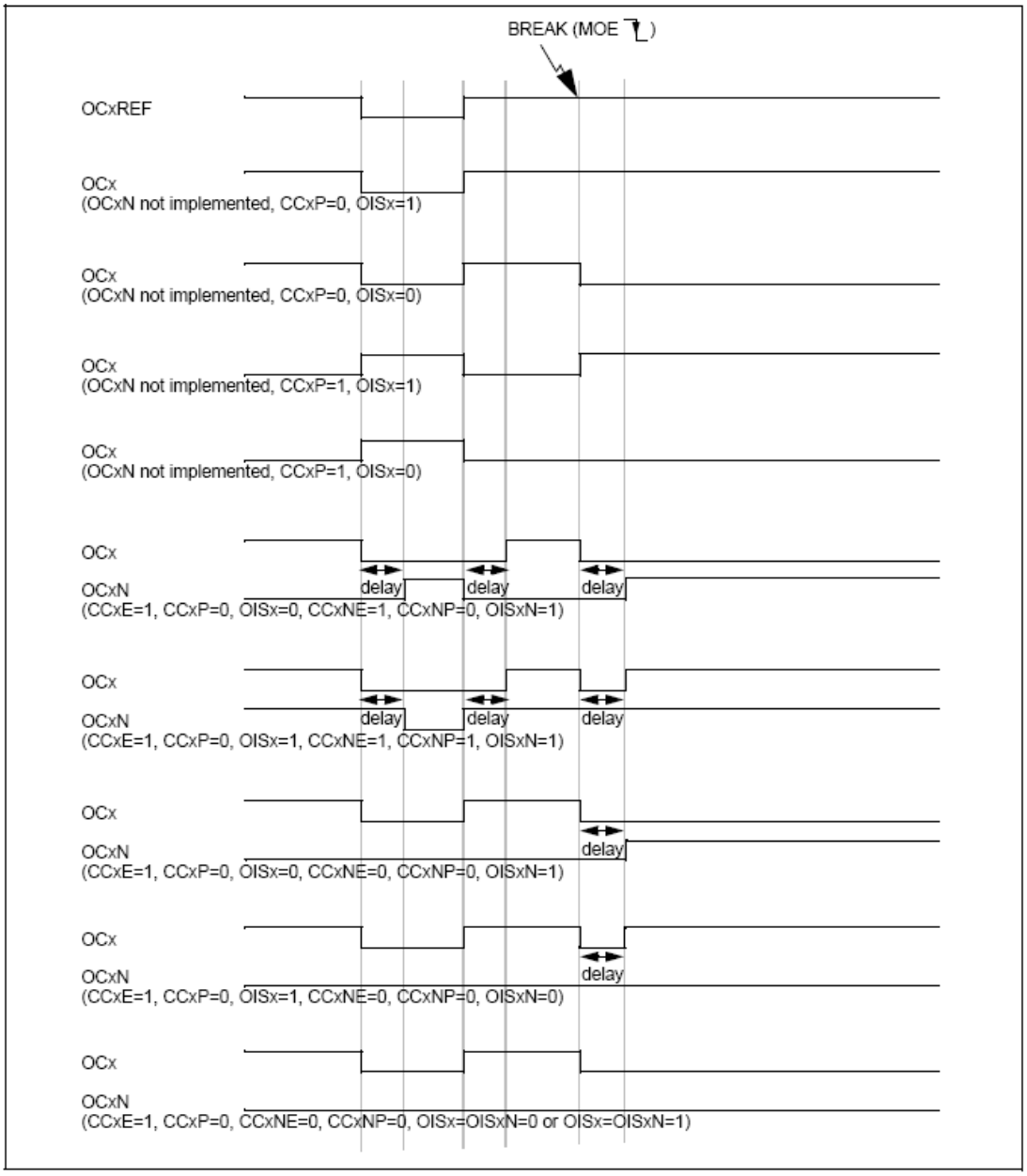
注： 刹车输入为电平有效。所以，当刹车输入有效(自动或者通过软件)MOE 不能同时被修改。此时，状态标志 BIF 不能被清除。

刹车由 BRK 输入产生，它有效极性是可编程的，且由 TIM1_BDTR 寄存器中的 BKE 位开启。

除了刹车输入和输出管理，刹车电路中还实现了写保护以保证应用程序的安全。它允许用户冻结几个配置参数(死区长度，OCx/OCxN极性和被禁止的状态，OCxM配置，刹车使能和极性)。用户可以通过TIM1_BDTR寄存器中的LOCK位，从三级保护中任选其一，参看 12.5.18 节。在MCU复位后LOCK位只能被修改一次。

图 62 显示响应刹车的输出实例。

图62 响应刹车的输出



12.4.13 在外部事件时清除OCxREF信号

对于一个给定的通道，在 ETRF 输入端(TIM1_CCMRx 寄存器中对应的 OCxCE 允许位置“1”)的高电平能够把 OCxREF 信号拉低，OCxREF 信号将保持为低直到发生下一次的更新事件 UEV。

该功能只能用于输出比较和 PWM 模式，而不能用于强置模式。

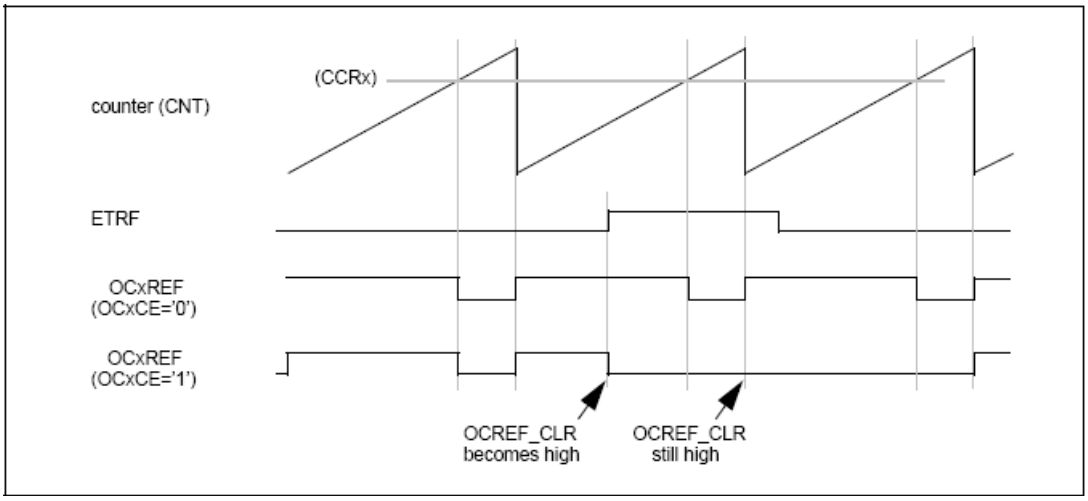
例如，OCxREF 信号可以联到一个比较器的输出，用于处理电流。这时，ETR 必须配置如下：

1. 外部触发预分频器必须处于关闭：TIM_SMCR 寄存器中的 ETPS[1:0]=00。

2. 必须禁止外部时钟模式2：TIM1_SMCR寄存器中的ECE=0。
3. 外部触发极性(ETP)和外部触发滤波器(ETF)可以根据需要配置。

图 63 显示了当ETRF输入变为高时，对应不同OCxCE的值，OCxREF信号的动作。在这个例子中，定时器TIM1 被置于PWM模式。

图63 清除 TIM1 的 OCxREF



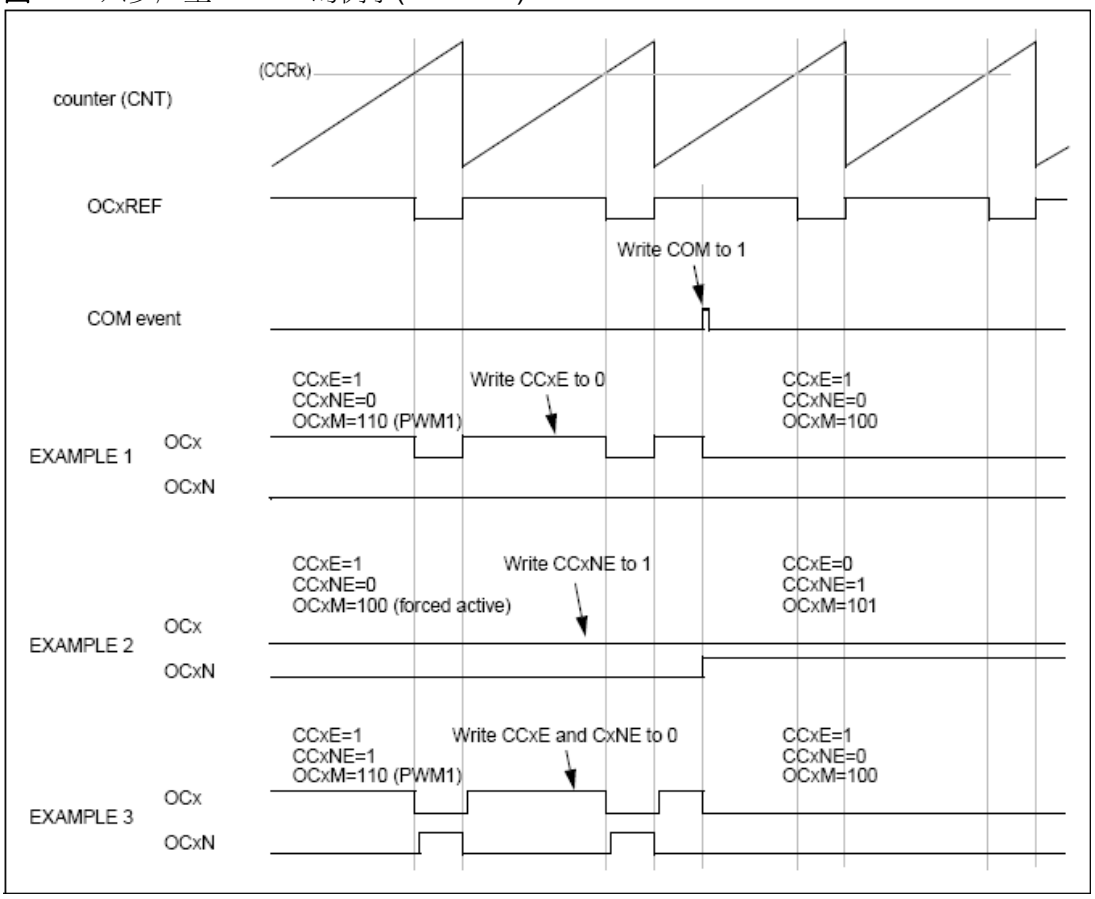
12.4.14 六步PWM的产生

在一个通道上应用了互补输出，OCxM、CCxE 和 CCxNE 的预装载位有效时，在 COM 通信事件发生时，预装载位被传送到影子位；因而你可以预先设置好下一步的配置，并在同一时间更改所有通道的配置。软件修改 TIM1_EGR 寄存器中的 COM 位或者硬件(在 TRGI 的上升沿)设置可以产生 COM 事件。

当发生 COM 事件时会设置一个标志位(TIM1_SR 寄存器中的 COMIF 位)，这时如果已设置了 TIM1_DIER 寄存器的 COMIE 位，则产生一个中断；或者如果已设置了 TIM1_DIER 寄存器的 COMDE 位，则产生一个 DMA 请求。

图 64 显示当COM事件发生时，三种不同配置下OCx和OCxN输出。

图64 六步产生，COM 的例子(OSSR=1)



12.4.15 单脉冲模式

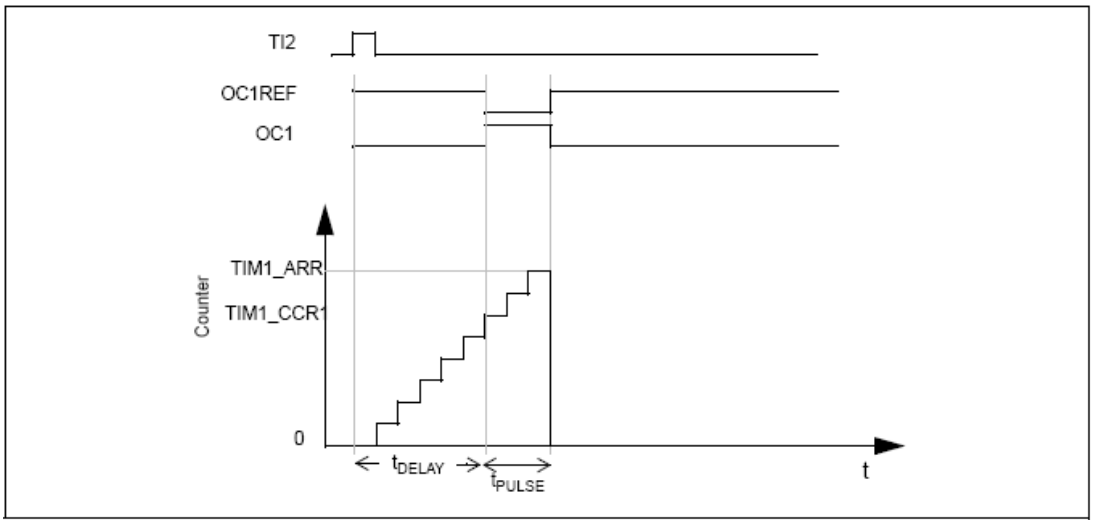
单脉冲模式(OPM)是前述众多模式的一个特例。这种模式允许计数器响应一个激励，并在一个程序可控的延时之后产生一个脉宽可程序控制的脉冲。

可以通过从模式控制器启动计数器，在输出比较模式或者 PWM 模式下产生波形。设置 TIM1_CR1 寄存器中的 OPM 位将选择单脉冲模式，这样可以让计数器自动地在产生下一个更新事件 UEV 时停止。

仅当比较值与计数器的初始值不同时，才能产生一个脉冲。启动之前(当定时器正在等待触发)，必须如下配置：

- 向上计数方式：CNT<CCRx≤ARR (特别地, 0<CCRx)，
- 向下计数方式：CNT>CCRx。

图65 单脉冲模式的例子



例如，你需要在从 TI2 输入脚上检测到一个上升沿开始，延迟 t_{DELAY} 之后，在 OC1 上产生一个长度为 t_{PULSE} 的正脉冲。

假定 TI2FP2 作为触发 1:

- 置 TIM1_CCMR1 寄存器中的 IC2S=01，把 TI2FP2 映像到 TI2。
- 置 TIM1_CCER 寄存器中的 CC2P=0，使 TI2FP2 能够检测上升沿。
- 置 TIM1_SMCR 寄存器中的 TS=110，配置 TI2FP2 作为从模式控制器的触发(TRGI)。
- 置 TIM1_SMCR 寄存器中的 SMS=110(触发模式)，TI2FP2 被用来启动计数器。

OPM 波形由写入比较寄存器决定(要考虑时钟频率和计数器预分频器)

- t_{DELAY} 由写入 TIM1_CCR1 寄存器中的值定义。
- t_{PULSE} 由自动装载值和比较值之间的差值定义(TIM1_ARR - TIM1_CCR1)。
- 假定当发生比较匹配时要产生从 0 到 1 的波形，当计数器到达预装载值是要产生一个从 1 到 0 的波形；首先要置 TIM1_CCMR1 寄存器中的 OC1M=111，进入 PWM 模式 2；有选择的置 TIM1_CCMR1 中的 OC1PE=1 和 TIM1_CR1 寄存器中的 ARPE，使能预装载寄存器；然后在 TIM1_CCR1 寄存器中填写比较值，在 TIM1_ARR 寄存器中填写自动装载值，修改 UG 位来产生一个更新事件，然后等待在 TI2 上的一个外部触发事件。本例中，CC1P=0。

在这个例子中，TIM1_CR1 寄存器中的 DIR 和 CMS 位应该置低。

因为只需一个脉冲，所以须在下一个更新事件(当计数器从自动装载值翻转到 0)时设置 TIM1_CR1 寄存器中的 OPM=1，以停止计数。

特殊情况：OCx快速使能

在单脉冲模式下，在 TIx 输入脚的边沿检测逻辑设置 CEN 位以启动计数器。然后计数器和比较值间的比较操作产生了输出的转换。但是这些操作需要一定的时钟周期，因此它限制了可得到的最小延时 tDELAY。
如果要以最小延时输出波形，可以设置 TIM1_CCMRx 寄存器中的 OCxFE 位；此时 OCxREF(和 OCx)被强制响应激励而不再依赖比较的结果，输出的波形与比较匹配时的波形一样。OCxFE 只在通道配置为 PWM1 和 PWM2 模式时起作用。

12.4.16 编码器接口模式

选择编码器接口模式的方法是：如果计数器只在 TI2 的边沿计数，则置 TIM1_SMCR 寄存器中的 SMS=001；如果只在 TI1 边沿计数，则置 SMS=010；如果计数器同时在 TI1 和 TI2 边沿计数，则置 SMS=011。

通过设置 TIM1_CCER 寄存器中的 CC1P 和 CC2P 位，可以选择 TI1 和 TI2 极性；如果需要，还可以对输入滤波器编程。

两个输入TI1 和TI2 被用来作为增量编码器的接口。参看 表 36，假定计数器已经启动(TIM1_CR1 寄存器中的CEN=1)，则计数器由每次在TI1FP1 或TI2FP2 上的有效跳变驱动。TI1FP1 和TI2FP2 是TI1 和TI2 在通过输入滤波器和极性控制后的信号；如果没有滤波和变相，则TI1FP1=TI1；如果没有滤波和变相，则TI2FP2=TI2。根据两个输入信号的跳变顺序，产生了计数脉冲和方向信号。依据两个输入信号的跳变顺序，计数器向上或向下计数，因此TIM1_CR1 寄存器的DIR位由硬件进行相应的设置。不管计数器是依靠TI1 计数、依靠TI2 计数或者同时依靠TI1 和TI2 计数。在任一输入(TI1 或者TI2)跳变时都会重新计算DIR位。

编码器接口模式基本上相当于使用了一个带有方向选择的外部时钟。这意味着计数器只在 0 到 TIM1_ARR 寄存器中自动装载值之间连续计数(根据方向，或是 0 到 ARR 计数，或是 ARR 到 0 计数)。所以在开始计数之前必须配置 TIM1_ARR；同样，捕获器、比较器、预分频器、周期计数器、触发输出特性等仍工作如常。编码模式和外部时钟模式 2 不兼容，因此不能同时操作。

在这个模式下，计数器依照增量编码器的速度和方向被自动的修改，因此，它的内容始终指示着编码器的位置。计数方向与相连的传感器旋转的方向对应。表 36 列出了所有可能的可能的组合，假设TI1 和TI2 不同时变换。

表36 计数方向与编码器信号的关系

有效边沿	相对信号的电平 (TI1FP1对应TI2, TI2FP2对应TI1)	TI1FP1信号		TI2FP2信号	
		上升	下降	上升	下降
仅在TI1计数	高	向下计数	向上计数	不计数	不计数
	低	向上计数	向下计数	不计数	不计数
仅在TI2计数	高	不计数	不计数	向上计数	向下计数
	低	不计数	不计数	向下计数	向上计数

在TI1和TI2上计数	高	向下计数	向上计数	向上计数	向下计数
	低	向上计数	向下计数	向下计数	向上计数

一个外部的增量编码器直接和 MCU 连接不需要外部接口逻辑。但是，一般使用比较器将编码器的差动输出转换到数字信号，这大大增加了抗噪声干扰能力。编码器输出的第三个信号表示机械零点，可以连接到一个外部中断输入和触发一个计数器复位。

图 66 给出一个计数器操作的实例，显示了计数信号的产生和方向控制。它还显示了当选择了双边沿时，输入抖动是如何被抑制的；这种情况可能会在传感器的位置靠近一个转换点事发生。在这个例子中，我们假定配置如下：

- CC1S=’01’ (TIM1_CCMR1 寄存器，IC1FP1 映射到 TI1)
- CC2S=’01’ (TIM1_CCMR2 寄存器，IC2FP2 映射到 TI2)
- CC1P=’0’ (TIM1_CCER 寄存器，IC1FP1 不反相，IC1FP1=TI1)
- CC2P=’0’ (TIM1_CCER 寄存器，IC2FP2 不反相，IC2FP2=TI2)
- SMS=’011’ (TIM1_SMCR 寄存器，所有的输入均在上升沿和下降沿有效).
- CEN=’1’ (TIM1_CR1 寄存器，计数器使能)

图66 编码器模式下的计数器操作实例

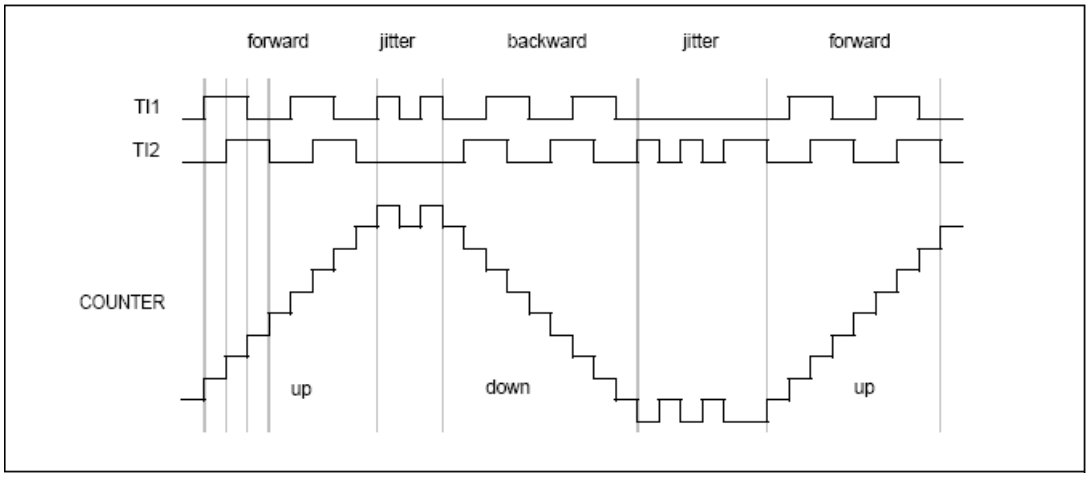
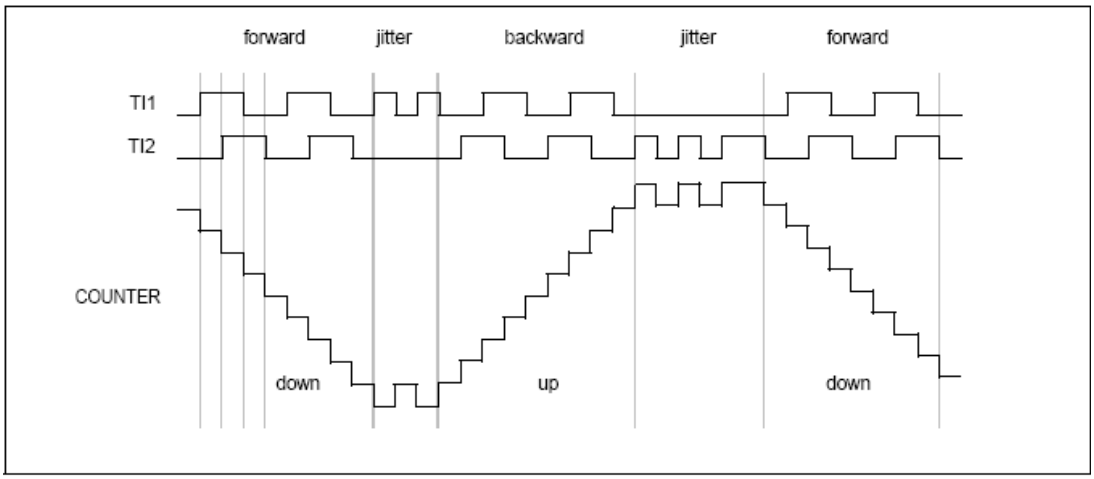


图 67 为当IC1FP1 极性反相时计数器的操作实例(CC1P=’1’，其他配置与上例相同)

图67 IC1FP1 反相的编码器接口模式实例



当定时器配置成编码器接口模式时，提供传感器当前位置的信息。使用第二个配置在捕获模式定时器测量两个编码器事件的间隔，可以获得动态的信息(速度，加速度，减速度)。指示机械零点的编码器输出可被用做此目的。根据两个事件间的间隔，可以按照固定的时间读出计数器。如果可能的话，你可以把计数器的值锁存到第三个输入捕获寄存器(捕获信号必须是周期的并且可以由另一个定时器产生)。它也可以通过一个由实时时钟产生的 DMA 请求来读取它的值。

12.4.17 定时器输入异或功能

TIM1_CR2 寄存器中的 TI1S 位，允许通道 1 的输入滤波器连接到一个异或门的输出端，异或门的 3 个输入端为 TIM1_CH1、TIM1_CH2 和 TIM1_CH3。

异或输出能够被用于所有定时器的输入功能，如触发或输入捕获。下节 12.4.18 给出了此特性用于连接霍尔传感器的例子。

12.4.18 与霍尔传感器的接口

使用高级控制定时器TIM1 定时器产生PWM信号用来驱动马达，而另一个TIMx定时器作为“接口定时器”来连接霍尔传感器，见 图 68，3 个定时器输入脚(CC1、CC2、CC3)通过一个异或门连接到TI1 输入通道(通过设置TIMx_CR2 寄存器中的TI1S位来选择)，“接口定时器”捕获这个信号。

从模式控制器被配置于复位模式，从输入是 TI1F_ED。因而，每次 3 个输入之一变化，计数器从 0 开始计数。这会产生一个由霍尔输入端的任何变化而触发的时间基准。
“接口定时器”上的捕获/比较通道 1 配置为捕获模式，捕获信号为TRC(参看 图 51)。捕获值反映了两个输入变化间的时间延迟，给出了马达速度的信息。

“接口定时器”可以用来在输出模式产生一个脉冲，这个脉冲可以用于改变定时器 TIM1 通道的属性(通过触发一个 COM 事件)。高级控制定时器 TIM1 用来产生 PWM 信号驱动马达。因此，“接口定时器”通道必须编程为在一个指定的延时(输出比较或 PWM 模式)之后产生一个正脉冲，这个脉冲通过 TRGO 输出被送到高级控制定时器 TIM1。

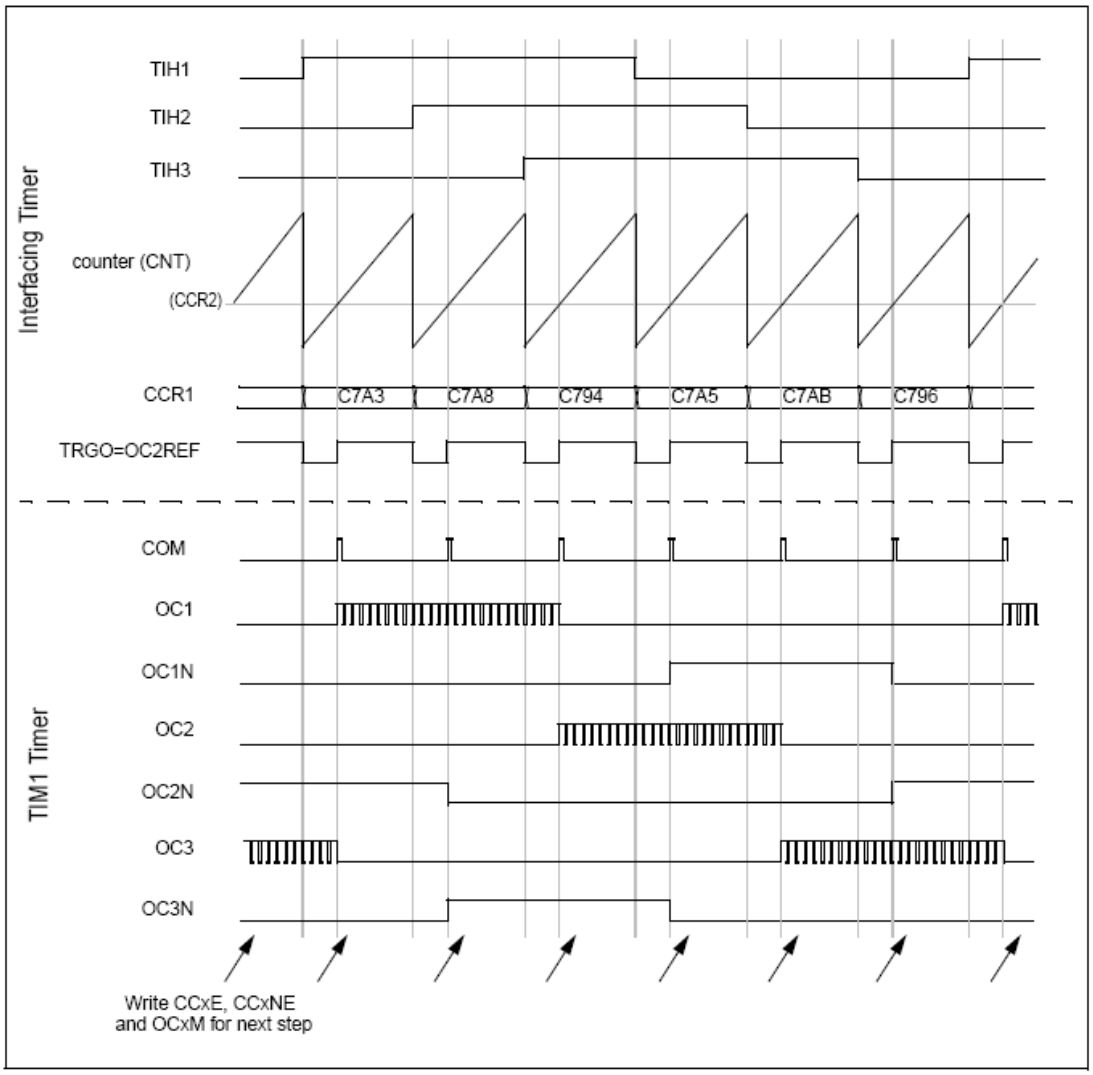
举例：霍尔输入连接到 TIMx 定时器，要求每次任一霍尔输入上发生变化之后的一个指定的时刻，改变高级控制定时器 TIM1 的 PWM 配置。

- 置 TIMx_CR2 寄存器的 TI1S 位为“1”，配置三个定时器输入逻辑或到 TI1 输入通道
- 对时基编程：置 TIMx_ARR 为其最大值(计数器必须通过 TI1 的变化清零)。设置预分频器得到一个最大的计数器周期，它长于传感器上的两次变化的时间间隔。
- 设置通道 1 为捕获模式(TRC 被选中)：置 TIMx_CCMR1 寄存器中 CC1S=01，如果有需要的话，还可以设置数字滤波器。
- 设置通道 2 为 PWM2 模式，并具有要求的延时：置 TIM1_CCMR1 寄存器中的 OC2M=111 和 CC2S=00。
- 选择 OC2REF 作为 TRGO 上的触发器输出：置 TIMx_CR2 寄存器中的 MMS=101。

在高级控制寄存器 TIM1 中，正确的 ITR 输入必须是触发器输入，定时器被编程为产生 PWM 信号，捕获/比较控制信号为预装载的(TIM1_CR2 寄存器中 CCPC=1)，同时触发输入控制 COM 事件(TIM1_CR2 寄存器中 CCUS=1)。在一次 COM 事件后，写入下一步的 PWM 控制位(CCxE、OCxM)，这可以在处理 OC2REF 上升沿的中断子程序里实现。

图 68 描述了这个实例

图68 霍尔传感器接口的实例



12.4.19 TIM1 定时器和外部触发的同步

TIM1 定时器能够在多种模式下和一个外部的触发同步：复位模式、门控模式和触发模式。

从模式：复位模式

在发生一个触发输入事件时，计数器和它的预分频器能够重新被初始化；同时，如果 TIM1_CR1 寄存器的 URS 位为低，还产生一个更新事件 UEV；然后所有的预装载寄存器(TIM1_ARR， TIM1_CCRx)都被更新了。

在以下的例子中，TI1 输入端的上升沿导致向上计数器被清零：

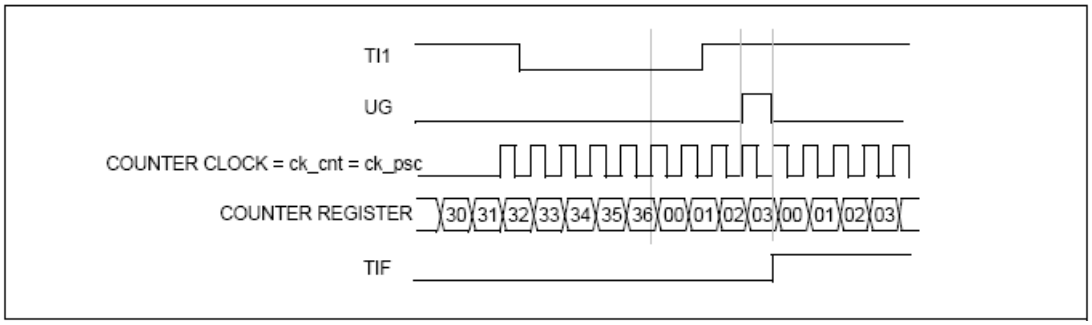
- 配置通道 1 以检测 TI1 的上升沿。配置输入滤波器的带宽(在本例中，不需要任何滤波器，因此保持 IC1F=0000)。触发操作中不使用捕获预分频器，所以不需要配置。CC1S 位只选择输入捕获源，即 TIM1_CCMR1 寄存器中 CC1S=01。置 TIM1_CCER 寄存器中 CC1P=0 以确定极性(只检测上升沿)。

- 置 TIM1_SMCR 寄存器中 SMS=100，配置定时器为复位模式；置 TIM1_SMCR 寄存器中 TS=101，选择 TI1 作为输入源。
- 置 TIM1_CR1 寄存器中 CEN=1，启动计数器。

计数器开始依据内部时钟计数，然后正常运转直到 TI1 出现一个上升沿；此时，计数器被清零然后从 0 重新开始计数。同时，触发标志(TIM1_SR 寄存器中的 TIF 位)被设置，根据 TIM1_DIER 寄存器中 TIE(中断使能)位和 TDE(DMA 使能)位的设置，产生一个中断请求或一个 DMA 请求。

下图显示当自动重装载寄存器 TIM1_ARR=0x36 时的动作。在 TI1 上升沿和计数器的实际复位之间的延时取决于 TI1 输入端的重同步电路。

图69 复位模式下的控制电路



从模式：门控模式

计数器的使能依赖于选中的输入端的电平。

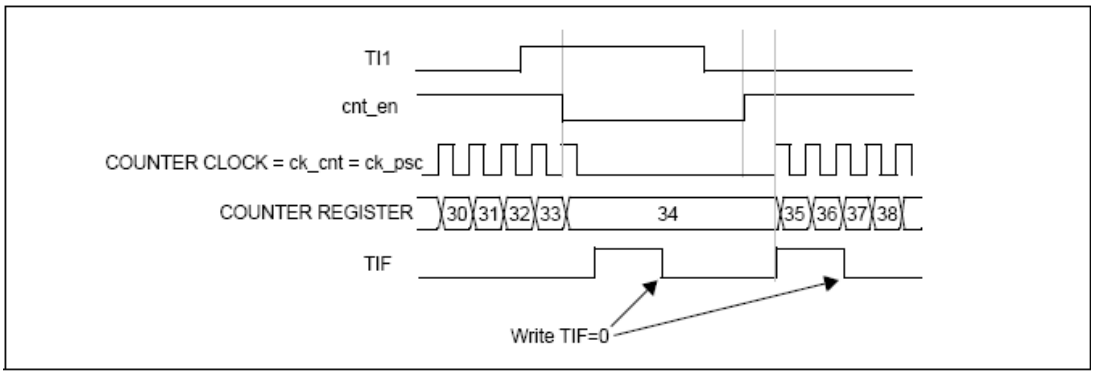
在如下的例子中，计数器只在 TI1 为低时向上计数：

- 配置通道 1 以检测 TI1 上的低电平。配置输入滤波器带宽(本例中，不需要滤波，所以保持 IC1F=0000)。触发操作中不使用捕获预分频器，所以不需要配置。CC1S 位用于选择输入捕获源，置 TIM1_CCMR1 寄存器中 CC1S=01。置 TIM1_CCER 寄存器中 CC1P=1 以确定极性(只检测低电平)。
- 置 TIM1_SMCR 寄存器中 SMS=101，配置定时器为门控模式；置 TIM1_SMCR 寄存器中 TS=101，选择 TI1 作为输入源。
- 置 TIM1_CR1 寄存器中 CEN=1，启动计数器。在门控模式下，如果 CEN=0，则计数器不能启动，不论触发输入电平如何。

只要 TI1 为低，计数器开始依据内部时钟计数，在 TI1 变高时停止计数。当计数器开始或停止时都设置 TIM1_SR 中的 TIF 标置。

TI1 上升沿和计数器实际停止之间的延时取决于 TI1 输入端的重同步电路。

图70　门控模式下的控制电路



从模式：触发模式

计数器的使能依赖于选中的输入端上的事件。

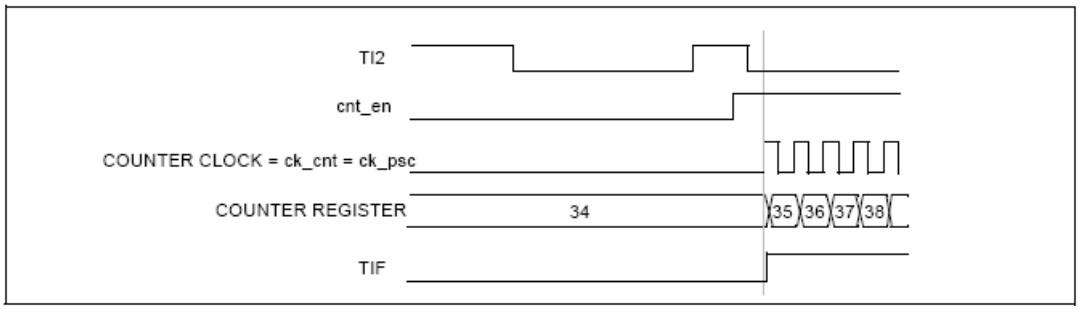
在下面的例子中，计数器在 TI2 输入的上升沿开始向上计数：

- 配置通道 2 检测 TI2 的上升沿。配置输入滤波器带宽(本例中，不需要任何滤波器，保持 IC2=0000)。触发操作中不使用捕获预分频器，不需要配置。CC2 位只用于选择输入捕获源，置 TIM1_CCMR1 寄存器中 CC2S=01。置 TIM1_CCER 寄存器中 CC1P=1 以确定极性(只检测低电平)。
- 置 TIM1_SMCR 寄存器中 SMS=110，配置定时器为触发模式；置 TIM1_SMCR 寄存器中 TS=110，选择 TI2 作为输入源。

当 TI2 上出现一个上升沿时，计数器开始在内部时钟驱动下计数，同时 TIF 标志被设置。

TI2 上升沿和计数器启动计数之间的延时取决于 TI2 输入端的重同步电路。

图71　触发器模式下的控制电路



从模式：外部时钟模式 2 + 触发模式

外部时钟模式 2 可以与另一种从模式(外部时钟模式 1 和编码器模式除外)一起使用。这时，ETR 信号被用作外部时钟的输入，可以选择另一个输入作为触发输入(在复位模式下，门控模式或触发模式)。不建议在 TIM1_SMCR 寄存器的 TS 位中选择 ETR 作为 TRGI。
在下面的例子中，一旦在 TI1 上出现一个上升沿，计数器即在 ETR 的每一个上升沿向上计数：

1. 通过TIM1_SMCR寄存器配置外部触发输入电路：

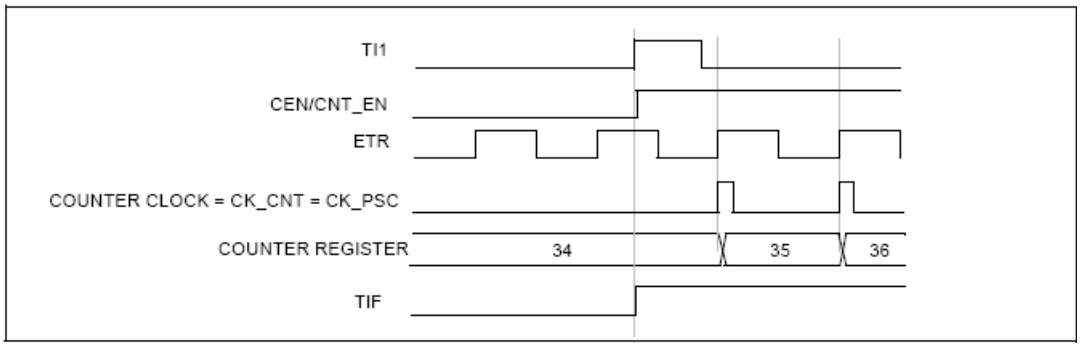
- ETF=0000：没有滤波
- ETPS=00：不用预分频器
- ETP=0：检测ETR的上升沿，置ECE=1使能外部时钟模式2

2. 按如下配置通道1，检测TI的上升沿：
   - IC1F=0000：没有滤波
   - 触发操作中不使用捕获预分频器，不需要配置
   - 置TIM1_CCMR1寄存器中CC1S=01，选择输入捕获源
   - 置TIM1_CCER寄存器中CC1P=0以确定极性(只检测上升沿)
3. 置TIM1_SMCR寄存器中SMS=110，配置定时器为触发模式。置TIM1_SMCR寄存器中TS=101，选择TI1作为输入源。

当 TI1 上出现一个上升沿时，TIF 标志被设置，计数器开始在 ETR 的上升沿计数。

ETR 信号的上升沿和计数器实际复位间的延时取决于 ETRP 输入端的重同步电路。

图72 外部时钟模式 2＋触发模式下的控制电路



12.4.20 定时器同步

所有 TIM 定时器在内部相连，用于定时器同步或链接。详见下一章

12.4.21 调试模式

当微控制器进入调试模式(Cortex-M3 核心停止)，根据 DBG 模块中 DBG_TIM1_STOP 的设置，TIM1 计数器可以或者继续正常操作，或者停止。详见随后章节。

12.5 TIM1 寄存器描述

关于在寄存器描述里面所用到的缩写，详见第 1 章。

12.5.1 控制寄存器 1(TIM1_CR1)

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
保留						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，始终读为0。
位9:8	CKD[1:0]: 时钟分频因子 这2位定义在定时器时钟(CK_INT)频率、死区时间和由死区发生器与数字滤波器(ETR,TIx)所用的采样时钟之间的分频比例。 00：$t_{DTS} = t_{CK\_INT}$ 01：$t_{DTS} = 2 \times t_{CK\_INT}$ 10：$t_{DTS} = 4 \times t_{CK\_INT}$ 11：保留，不要使用这个配置
位7	ARPE：自动重装载预装载允许位 0：TIM1_ARR寄存器没有缓冲 1：TIM1_ARR寄存器被装入缓冲器
位6:5	CMS[1:0]：选择中央对齐模式 00：边沿对齐模式。计数器依据方向位(DIR)向上或向下计数。 01：中央对齐模式1。计数器交替地向上和向下计数。配置为输出的通道(TIM1_CCMRx寄存器中CCxS=00)的输出比较中断标志位，只在计数器向下计数时被设置。 10：中央对齐模式2。计数器交替地向上和向下计数。计数器交替地向上和向下计数。配置为输出的通道(TIM1_CCMRx寄存器中CCxS=00)的输出比较中断标志位，只在计数器向上计数时被设置。 11：中央对齐模式3。计数器交替地向上和向下计数。计数器交替地向上和向下计数。配置为输出的通道(TIM1_CCMRx寄存器中CCxS=00)的输出比较中断标志位，在计数器向上和向下计数时均被设置。 注：在计数器开启时(CEN=1)，不允许从边沿对齐模式转换到中央对齐模式。
位4	DIR：方向 0：计数器向上计数 1：计数器向下计数 注：当计数器配置为中央对齐模式或编码器模式时，该位为只读。
位3	OPM：单脉冲模式 0：在发生更新事件时，计数器不停止 1：在发生下一次更新事件(清除CEN位)时，计数器停止。

位2	URS：更新请求源 软件通过该位选择UEV事件的源 0：如果允许产生更新中断或DMA请求，则下述任一事件产生一个更新中断或DMA请求： – 计数器溢出/下溢 – 设置UG位 – 从模式控制器产生的更新 1：如果允许产生更新中断或DMA请求，则只有计数器溢出/下溢产生一个更新中断或DMA请求
位1	UDIS：禁止更新 软件通过该位允许/禁止UEV事件的产生 0：允许UEV。更新(UEV)事件由下述任一事件产生： – 计数器溢出/下溢 – 设置UG位 – 从模式控制器产生的更新 被缓存的寄存器被装入它们的预装载值。 1：禁止UEV。不产生更新事件，影子寄存器(ARR,PSC,CCRx)保持它们的值。如果设置了UG位或从模式控制器发出了一个硬件复位，则计数器和预分频器被重新初始化。
位0	CEN：允许计数器 0：禁止计数器 1：开启计数器 注：在软件设置了CEN位后，外部时钟、门控模式和编码器模式才能工作。触发模式可以自动地通过硬件设置CEN位。

12.5.2 控制寄存器 2(TIM1_CR2)

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
保留	OIS4	OIS3N	OIS3	OIS2N	OIS2	OIS1N	OIS1	TI1S	MMS[2:0]			CCDS	CCUS	保留	CCPC
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw

位15	保留，始终读为0。
位14	OIS4：输出空闲状态4(OC4输出)。参见OIS1位。
位13	OIS3N：输出空闲状态3(OC3N输出)。参见OIS1N位。
位12	OIS3：输出空闲状态3(OC3输出)。参见OIS1位。
位11	OIS2N：输出空闲状态2(OC2N输出)。参见OIS1N位。
位10	OIS2：输出空闲状态2(OC2输出)。参见OIS1位。

位9	OIS1N：输出空闲状态1(OC1N输出)。 0：当MOE=0时，死区后OC1N=0 1：当MOE=0时，死区后OC1N=1 注：已经设置了LOCK(TIM1_BKR寄存器)级别1、2或3后，该位不能被修改。
位8	OIS1：输出空闲状态1(OC1输出)。 0：当MOE=0时，如果实现了OC1N，则死区后OC1=0 1：当MOE=0时，如果实现了OC1N，则死区后OC1=1 注：已经设置了LOCK(TIM1_BKR寄存器)级别1、2或3后，该位不能被修改。
位7	TI1S：TI1选择 0：TIM1_CH1管脚连到TI1输入。 1：TIM1_CH1、TIM1_CH2和TIM1_CH3管脚经异或后连到TI1输入。
位6:4	MMS[1:0]：主模式选择 这两位用于选择在主模式下送到从定时器的同步信息(TRGO)。可能的组合如下： 000：复位 – TIM1_EGR寄存器的UG位被用于作为触发输出(TRGO)。如果触发输入(复位模式下的从模式控制器)产生复位，则TRGO上的信号相对实际的复位会有一个延迟。 001：允许 – 计数器使能信号CNT_EN被用于作为触发输出(TRGO)。有时需要在同一时间启动多个定时器或控制从定时器的一个窗口。计数器使能信号是通过CEN控制位和门控模式下的触发输入信号的逻辑或产生。当计数器使能信号受控于触发输入时，TRGO上会有一个延迟，除非选择了主/从模式(见TIM1_SMCR寄存器中MSM位的描述)。 010：更新 – 更新事件被选为触发输入(TRGO)。例如，一个主定时器的时钟可以被用作一个从定时器的预分频器。 011：比较脉冲 – 一旦发生一次捕获或一次比较成功时，当要设置CC1IF标志时(即是它已经为高)，触发输出送出一个正脉冲(TRGO)。 100：比较 – OC1REF信号被用于作为触发输出(TRGO)。 101：比较 – OC2REF信号被用于作为触发输出(TRGO)。 110：比较 – OC3REF信号被用于作为触发输出(TRGO)。 111：比较 – OC4REF信号被用于作为触发输出(TRGO)。
位3	CCDS：捕获/比较的DMA选择 0：当发生CCx事件时，送出CCx的DMA请求。 1：当发生更新事件时，送出CCx的DMA请求。
位2	CCUS：捕获/比较控制更新选择 0：如果捕获/比较控制位是预装载的(CCPC=1)，只能通过设置COM位更新它们。 1：如果捕获/比较控制位是预装载的(CCPC=1)，可以通过设置COM位或TRGI上的一个上升沿更新它们。 注：该位只对具有互补输出的通道起作用。
位1	保留，始终读为0。
位0	CCPC：捕获/比较预装载控制位 0：CCxE，CCxNE和OCxM位不是预装载的。 1：CCxE，CCxNE和OCxM位是预装载的；设置该位后，它们只在设置了COM位后被更新。 注：该位只对具有互补输出的通道起作用。

12.5.3 从模式控制寄存器(TIM1_SMCR)

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]			保留	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 该位选择是用ETR还是ETR的反相来作为触发操作 0：ETR不反相，高电平或上升沿有效 1：ETR被反相，低电平或下降沿有效
位14	ECE：外部时钟允许位 该位启用外部时钟模式2 0：禁止外部时钟模式2 1：启用外部时钟模式2。计数器由ETRF信号上的任意有效上升沿驱动。 注1：设置ECE位与选择外部时钟模式1并将TRGI连到ETRF(SMS=111和TS=111)具有相同功效。 注2：下述从模式可以与外部时钟模式2同时使用：复位模式，门控模式和触发模式；但是，这时TRGI不能连到ETRF(TS位不能是111)。
位13:12	ETPS[1:0]：外部触发预分频 外部触发信号ETRP的频率必须最多是TIM1CLK频率的1/4。当输入较快的外部时钟时，可以使用预分频降低ETRP的频率。 00：关闭预分频 01：ETRP频率除以2 10：ETRP频率除以4 11：ETRP频率除以8
位11:8	ETF[3:0]：外部触发滤波 这些位定义了对ETRP信号采样的频率和对ETRP数字滤波的带宽。实际上，数字滤波器是一个事件计数器，它记录到N个事件后会产生一个输出的跳变。 0000：无滤波器，以f_{DTS}采样 0001：采样频率$f_{SAMPLING}=f_{CK_INT}$，N=2 0010：采样频率$f_{SAMPLING}=f_{CK_INT}$，N=4 0011：采样频率$f_{SAMPLING}=f_{CK_INT}$，N=8 0100：采样频率$f_{SAMPLING}=f_{DTS}/2$，N=6 0101：采样频率$f_{SAMPLING}=f_{DTS}/2$，N=8 0110：采样频率$f_{SAMPLING}=f_{DTS}/4$，N=6 0111：采样频率$f_{SAMPLING}=f_{DTS}/4$，N=8 1000：采样频率$f_{SAMPLING}=f_{DTS}/8$，N=6 1001：采样频率$f_{SAMPLING}=f_{DTS}/8$，N=8 1010：采样频率$f_{SAMPLING}=f_{DTS}/16$，N=5 1011：采样频率$f_{SAMPLING}=f_{DTS}/16$，N=6 1100：采样频率$f_{SAMPLING}=f_{DTS}/16$，N=8 1101：采样频率$f_{SAMPLING}=f_{DTS}/32$，N=5 1110：采样频率$f_{SAMPLING}=f_{DTS}/32$，N=6 1111：采样频率$f_{SAMPLING}=f_{DTS}/32$，N=8

位7	MSM：主/从模式 0：无作用 1：触发输入(TRGI)上的事件被延迟了，以允许在当前定时器(通过TRGO)与它的从定时器间的完美同步。这对要求把几个定时器同步到一个单一的外部事件时是非常有用的。
位6:4	TS[2:0]：触发选择 这3位选择用于同步计数器的触发输入。 000：保留　100：TI1的边沿检测器(TI1F_ED) 001：TIM2(ITR1)　101：滤波后的定时器输入1(TI1FP1) 010：TIM3(ITR2)　110：滤波后的定时器输入2(TI2FP2) 011：TIM4(ITR3)　111：外部触发输入(ETRF) 注：为避免在信号转变时产生错误的边沿检测，必须在未使用这些位时修改它们。
位3	保留，始终读为0。
位2:0	SMS：从模式选择 当选择了外部信号，触发信号(TRGI)的有效边沿与选中的外部输入极性相关(见输入控制寄存器和控制寄存器的说明) 000：关闭从模式 – 如果CEN=1，则预分频器直接由内部时钟驱动。 001：编码器模式1 – 根据TI1FP1的电平，计数器在TI2FP2的边沿向上/下计数。 010：编码器模式2 – 根据TI2FP2的电平，计数器在TI1FP1的边沿向上/下计数。 011：编码器模式3 – 根据其他输入的电平，计数器在TI1FP1和TI2FP2的边沿向上/下计数。 100：复位模式 – 选中的触发输入(TRGI)的上升沿重新初始化计数器，并且产生一个更新寄存器的信号。 101：门控模式 – 当触发输入(TRGI)为高时，计数器的时钟开启。一旦触发输入变为低，则计数器停止(但不复位)。计数器的启动和停止都是受控的。 110：触发模式 – 计数器在触发输入TRGI的上升沿启动(但不复位)，只有计数器的启动是受控的。 111：外部时钟模式1 – 选中的触发输入(TRGI)的上升沿驱动计数器。 注：如果TI1F_EN被选为触发输入(TS=100)时，不要使用门控模式。这是因为，TI1F_ED在每次TI1F变化时输出一个脉冲，然而门控模式是要检查触发输入的电平。

12.5.4 DMA/中断使能寄存器(TIM1_DIER)

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
保留	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，始终读为0。
位14	TDE：允许触发DMA请求 0：禁止触发DMA请求 1：允许触发DMA请求

位13	COMDE：允许COM的DMA请求 0：禁止COM的DMA请求 1：允许COM的DMA请求
位12	CC4DE：允许捕获/比较4的DMA请求 0：禁止捕获/比较4的DMA请求 1：允许捕获/比较4的DMA请求
位11	CC3DE：允许捕获/比较3的DMA请求 0：禁止捕获/比较3的DMA请求 1：允许捕获/比较3的DMA请求
位10	CC2DE：允许捕获/比较2的DMA请求 0：禁止捕获/比较2的DMA请求 1：允许捕获/比较2的DMA请求
位9	CC1DE：允许捕获/比较1的DMA请求 0：禁止捕获/比较1的DMA请求 1：允许捕获/比较1的DMA请求
位8	UDE：允许更新的DMA请求 0：禁止更新的DMA请求 1：允许更新的DMA请求
位7	BIE：允许刹车中断 0：禁止刹车中断 1：允许刹车中断
位6	TIE：允许触发中断 0：禁止触发中断 1：允许触发中断
位5	COMIE：允许COM中断 0：禁止COM中断 1：允许COM中断
位4	CC4IE：允许捕获/比较4中断 0：禁止捕获/比较4中断 1：允许捕获/比较4中断
位3	CC3IE：允许捕获/比较3中断 0：禁止捕获/比较3中断 1：允许捕获/比较3中断
位2	CC2IE：允许捕获/比较2中断 0：禁止捕获/比较2中断 1：允许捕获/比较2中断
位1	CC1IE：允许捕获/比较1中断 0：禁止捕获/比较1中断 1：允许捕获/比较1中断
位0	UIE：允许更新中断 0：禁止更新中断 1：允许更新中断

12.5.5 状态寄存器(TIM1_SR)

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
保留			CC4OF	CC3OF	CC2OF	CC1OF	保留	BIF	TIF	COMIF	CC4IF	CC3IF	CC2IF	CC1IF	UIF
			rc	rc	rc	rc		rc	rc	rc	rc	rc	rc	rc	rc

位15:13	保留，始终读为0。
位12	**CC4OF**：捕获/比较4 过捕获标记 参见CC1OF描述
位11	**CC3OF**：捕获/比较3 过捕获标记 参见CC1OF描述
位10	**CC2OF**：捕获/比较2 过捕获标记 参见CC1OF描述
位9	**CC1OF**：捕获/比较1 过捕获标记 仅当相应的通道被配置为输入捕获时，该标记可由硬件置1。写0可清除该位。 0：无过捕获产生； 1：CC1IF置1时，计数器的值已经被捕获到TIM1_CCR1寄存器。
位8	保留，始终读为0。
位7	**BIF**：刹车中断标记 一旦刹车输入有效，由硬件对该位置1。如果刹车输入无效，则该位可由软件清0。 0：无刹车事件产生； 1：刹车输入上检测到有效电平。
位6	**TIF**：触发器中断标记 当发生触发事件（当从模式控制器处于除门控模式外的其它模式时,在TRGI输入端检测到有效边沿，或或门控模式下的任一边沿）时由硬件对该位置1。它由软件清0。 0：无触发器事件产生； 1：触发器中断等待响应
位5	**COMIF**：COM中断标记 一旦产生COM事件（当CCxE、CCxNE、OCxM已被更新）该位由硬件置1。它由软件清0。 0：无COM事件产生； 1：COM中断等待响应
位4	**CC4IF**：捕获/比较4 中断标记 参考CC1IF描述
位3	**CC3IF**：捕获/比较3 中断标记 参考CC1IF描述
位2	**CC2IF**：捕获/比较2 中断标记 参考CC1IF描述

位1	**CC1IF**：捕获/比较1 中断标记 **如果通道CC1配置为输出模式：** 当计数器值与比较值匹配时该位由硬件置1，但在中心对称模式下除外(参考TIM1_CR1寄存器的CMS位)。它由软件清0。 0：无匹配发生； 1：TIM1_CNT的值与TIM1_CCR1的值匹配。 **如果通道CC1配置为输入模式：** 当捕获事件发生时该位由硬件置1，它由软件清0或通过读TIM1_CCR1清0。 0：无输入捕获产生； 1：输入捕获产生并且计数器值已装入TIM1_CCR1(在IC1上检测到与所选极性相同的边沿)。
位0	**UIF**：更新中断标记 当产生更新事件时该位由硬件置1。它由软件清0。 0：无更新事件产生； 1：更新事件等待响应。当寄存器被更新时该位由硬件置1： – 若TIM1_CR1寄存器的UDIS=0，当REP_CNT=0时产生更新事件(重复向下计数器上溢或下溢时)； – 若TIM1_CR1寄存器的UDIS=0、URS=0，当TIM1_EGR寄存器的UG=1时产生更新事件(软件对CNT重新初始化)； – 若TIM1_CR1寄存器的UDIS=0、URS=0，当CNT被触发事件重初始化时产生更新事件。（参考12.5.3: 从模式控制寄存器(TIM1_SMCR)）

12.5.6 事件产生寄存器(TIM1_EGR)

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
保留								BG	TG	COMG	CC4G	CC3G	CC2G	CC1G	UG
								w	w	w	w	w	w	w	w

位15:8	保留，始终读为0。
位7	**BG**：产生刹车事件 该位由软件置1，用于产生一个刹车事件，由硬件自动清0。 0：无动作； 1：产生一个刹车事件。此时MOE=0、BIF=1，若开启对应的中断和DMA，则产生相应的中断和DMA。
位6	**TG**：产生触发事件 该位由软件置1，用于产生一个触发事件，由硬件自动清0。 0：无动作； 1：TIM1_SR寄存器的TIF=1，若开启对应的中断和DMA，则产生相应的中断和DMA。
位5	**COMG**：捕获/比较事件，产生控制更新 该位由软件置1，由硬件自动清0。 0：无动作； 1：当CCPC=1，允许更新CCxE、CCxNE、OCxM位。 **注**：该位只对有互补输出的通道有效。

位4	**CC4G**：产生捕获/比较4事件 参考CC1G描述
位3	**CC3G**：产生捕获/比较3事件 参考CC1G描述
位2	**CC2G**：产生捕获/比较2事件 参考CC1G描述
位1	**CC1G**：产生捕获/比较1事件 该位由软件置1，用于产生一个捕获/比较事件，由硬件自动清0。 0：无动作； 1：在通道CC1上产生一个捕获/比较事件： **若通道CC1配置为输出：** 设置CC1IF=1，若开启对应的中断和DMA，则产生相应的中断和DMA。 **若通道CC1配置为输入：** 当前的计数器值捕获至TIM1_CCR1寄存器，设置CC1IF=1，若开启对应的中断和DMA，则产生相应的中断和DMA。若CC1IF已经为1，则设置CC1OF=1。
位0	**UG**：产生更新事件 该位由软件置1，由硬件自动清0。 0：无动作； 1：重新初始化计数器，并产生一个更新事件。注意预分频器的计数器也被清0(但是预分频系数不变)。若在中心对称模式下或DIR=0(向上计数)则计数器被清0，若DIR=1(向下计数)则计数器取TIM1_ARR的值。

12.5.7 捕获/比较模式寄存器 1(TIM1_CCMR1)

偏移地址：0x18
复位值：0x0000

通道可用于输入(捕获模式)或输出(比较模式)，通道的方向由相应的 CCxS 定义。该寄存器其它位的作用在输入和输出模式下不同。OCxx 描述了通道在输出模式下的功能，ICxx 描述了通道在输出模式下的功能。因此你必须注意，同一个位在输出模式和输入模式下的功能是不同的。

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC2CE	OC2M[2:0]			OC2PE	OC2FE	CC2S[1:0]		OC1CE	OC1M[2:0]			OC1PE	OC1FE	CC1S[1:0]	
IC2F[3:0]				IC2PSC[1:0]				IC1F[3:0]				IC1PSC[1:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

输出比较模式：

位15	OC2CE：输出比较2清0使能
位14:12	0C2M[2:0]：输出比较2模式
位11	OC2PE：输出比较2预装载使能
位10	OC2FE：输出比较2快速使能

位9:8	CC2S[1:0]：捕获/比较2选择。 该位定义通道的方向（输入/输出），及输入脚的选择： 00：CC2通道被配置为输出； 01：CC2通道被配置为输入，IC2映射在TI2上； 10：CC2通道被配置为输入，IC2映射在TI1上； 11：CC2通道被配置为输入，IC2映射在TRC上。此模式仅工作在内部触发器输入被选中时（由TIM1_SMCR寄存器的TS位选择）。 **注：**CC2S仅在通道关闭时(TIM1_CCER寄存器的CC2E=0)才是可写的。
位7	OC1CE：输出比较1清0使能 0：OC1REF 不受ETRF输入的影响； 1：一旦检测到ETRF输入高电平，清除OC1REF=0。
位6:4	0C1M[2:0]：输出比较1模式 该位定义了输出参考信号OC1REF的动作，而OC1REF决定了OC1、OC1N的值。OC1REF是高电平有效，而OC1、OC1N的有效电平取决于CC1P、CC1NP位。 000：冻结。输出比较寄存器TIM1_CCR1与计数器TIM1_CNT间的比较对OC1REF不起作用； 001：匹配时设置通道1为有效电平。当计数器TIMx_CNT的值与捕获/比较寄存器1(TIMx_CCR1)相同时，强制OC1REF为高。 010：匹配时设置通道1为无效电平。当计数器TIMx_CNT的值与捕获/比较寄存器1(TIMx_CCR1)相同时，强制OC1REF为低。 011：翻转。当TIM1_CCR1=TIM1_CNT时，翻转OC1REF的电平。 100：强制为无效电平。强制OC1REF为低。 101：强制为有效电平。强制OC1REF为高。 110：PWM模式1－ 在向上计数时，一旦TIM1_CNT<TIM1_CCR1时通道1为有效电平，否则为无效电平；在向下计数时，一旦TIM1_CNT>TIM1_CCR1时通道1为无效电平(OC1REF=0)，否则为有效电平(OC1REF=1)。 111：PWM模式2－ 在向上计数时，一旦TIM1_CNT<TIM1_CCR1时通道1为无效电平，否则为有效电平；在向下计数时，一旦TIM1_CNT>TIM1_CCR1时通道1为有效电平，否则为无效电平。 注1：一旦LOCK级别设为3(TIMx_BDTR寄存器中的LOCK位)并且CC1S=00(该通道配置成输出)则该位不能被修改。 注2：在PWM模式1或PWM模式2中，只有当比较结果改变了或在输出比较模式中从冻结模式切换到PWM模式时，OC1REF电平才改变。
位3	**OC1PE**：输出比较1预装载使能 0：禁止TIM1_CCR1寄存器的预装载功能，可随时写入TIM1_CCR1寄存器，且新值马上起作用。 1：开启TIM1_CCR1寄存器的预装载功能，读写操作仅对预装载寄存器操作，TIM1_CCR1的预装载值在更新事件到来时被载入当前寄存器中。 注1：一旦LOCK级别设为3(TIMx_BDTR寄存器中的LOCK位)并且CC1S=00(该通道配置成输出)则该位不能被修改。 注2：仅在单脉冲模式下，可以在未确认预装载寄存器情况下使用PWM模式，否则其动作不确定。
位2	**OC1FE**：输出比较1 快速使能 该位用于加快CC输出对触发器输入事件的响应。 0：根据计数器与CCR1的值，CC1正常操作，即使触发器是打开的。当触发器的输入有一个有效沿时，激活CC1输出的最小延时为5个时钟周期。 1：输入到触发器的有效沿的作用就象发生了一次比较匹配。因此，OC被设置为比较电平而与比较结果无关。采样触发器的有效沿和CC1输出间的延时被缩短为3个时钟周期。 OCFE的只在通道被配置成PWM1或PWM2模式时起作用。

位1:0	CC1S[1:0]：捕获/比较1 选择。 这2位定义通道的方向（输入/输出），及输入脚的选择： 00：CC1通道被配置为输出； 01：CC1通道被配置为输入，IC1映射在TI1上； 10：CC1通道被配置为输入，IC1映射在TI2上； 11：CC1通道被配置为输入，IC1映射在TRC上。此模式仅工作在内部触发器输入被选中时（由TIM1_SMCR寄存器的TS位选择）。 注：CC1S仅在通道关闭时(TIM1_CCER寄存器的CC1E=0)才是可写的。

输入捕获模式：

位15:12	IC2F：输入捕获2滤波器
位11:10	IC2PSC[1:0]：输入/捕获2预分频器
位9:8	CC2S[1:0]：捕获/比较2选择。 这2位定义通道的方向（输入/输出），及输入脚的选择： 00：CC2通道被配置为输出； 01：CC2通道被配置为输入，IC2映射在TI2上； 10：CC2通道被配置为输入，IC2映射在TI1上； 11：CC2通道被配置为输入，IC2映射在TRC上。此模式仅工作在内部触发器输入被选中时（由TIM1_SMCR寄存器的TS位选择）。 注：CC2S仅在通道关闭时(TIM1_CCER寄存器的CC2E=0)才是可写的。
位7:4	IC1F[3:0]：输入捕获1滤波器 这几位定义了TI1输入的采样频率及数字滤波器长度。数字滤波器由一个事件计数器组成，它记录到N个事件后会产生一个输出的跳变： 0000：无滤波器，以f_{DTS}采样　　1000：采样频率$f_{SAMPLING}=f_{DTS}/8$，N=6 0001：采样频率$f_{SAMPLING}=f_{CK_INT}$，N=2　　1001：采样频率$f_{SAMPLING}=f_{DTS}/8$，N=8 0010：采样频率$f_{SAMPLING}=f_{CK_INT}$，N=4　　1010：采样频率$f_{SAMPLING}=f_{DTS}/16$，N=5 0011：采样频率$f_{SAMPLING}=f_{CK_INT}$，N=8　　1011：采样频率$f_{SAMPLING}=f_{DTS}/16$，N=6 0100：采样频率$f_{SAMPLING}=f_{DTS}/2$，N=6　　1100：采样频率$f_{SAMPLING}=f_{DTS}/16$，N=8 0101：采样频率$f_{SAMPLING}=f_{DTS}/2$，N=8　　1101：采样频率$f_{SAMPLING}=f_{DTS}/32$，N=5 0110：采样频率$f_{SAMPLING}=f_{DTS}/4$，N=6　　1110：采样频率$f_{SAMPLING}=f_{DTS}/32$，N=6 0111：采样频率$f_{SAMPLING}=f_{DTS}/4$，N=8　　1111：采样频率$f_{SAMPLING}=f_{DTS}/32$，N=8
位3:2	IC1PSC[1:0]：输入/捕获1预分频器 这2位定义了CC1输入（IC1）的预分频系数。一旦CC1E=0(TIM1_CCER寄存器中)，则预分频器复位。 00：无预分频器，捕获输入口上检测到的每一个边沿都触发一次捕获； 01：每2个事件触发一次捕获； 10：每4个事件触发一次捕获； 11：每8个事件触发一次捕获。

位1:0	**CC1S[1:0]**：捕获/比较1选择。 这2位定义通道的方向（输入/输出），及输入脚的选择： 00：CC1通道被配置为输出； 01：CC1通道被配置为输入，IC1映射在TI1上； 10：CC1通道被配置为输入，IC1映射在TI2上； 11：CC1通道被配置为输入，IC1映射在TRC上。此模式仅工作在内部触发器输入被选中时（由TIM1_SMCR寄存器的TS位选择）。 **注**：CC1S仅在通道关闭时(TIM1_CCER寄存器的CC1E=0)才是可写的。

12.5.8 捕获/比较模式寄存器 2(TIM1_CCMR2)

偏移地址：0x1C
复位值：0x0000
参看以上 CCMR1 寄存器的描述

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC4CE	OC4M[2:0]			OC4PE	OC4FE	CC4S[1:0]		OC3CE	OC3M[2:0]			OC3PE	OC3FE	CC3S[1:0]	
IC4F[3:0]				IC4PSC[1:0]				IC3F[3:0]				IC3PSC[1:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

输出比较模式：

位15	OC4CE：输出比较4清0使能
位14:12	0C4M[2:0]：输出比较4模式
位11	OC4PE：输出比较4预装载使能
位10	OC4FE：输出比较4快速使能
位9:8	CC4S[1:0]：捕获/比较4选择。 该位定义通道的方向（输入/输出），及输入脚的选择： 00：CC4通道被配置为输出； 01：CC4通道被配置为输入，IC4映射在TI4上； 10：CC4通道被配置为输入，IC4映射在TI3上； 11：CC4通道被配置为输入，IC4映射在TRC上。此模式仅工作在内部触发器输入被选中时（由TIM1_SMCR寄存器的TS位选择）。 **注**：CC4S仅在通道关闭时(TIM1_CCER寄存器的CC4E=0)才是可写的。
位7	OC3CE：输出比较3清0使能
位6:4	0C3M[2:0]：输出比较3模式
位3	OC3PE：输出比较3预装载使能
位2	OC3FE：输出比较3快速使能

位1:0	CC3S[1:0]：捕获/比较3选择。 这2位定义通道的方向（输入/输出），及输入脚的选择： 00：CC3通道被配置为输出； 01：CC3通道被配置为输入，IC3映射在TI3上； 10：CC3通道被配置为输入，IC3映射在TI4上； 11：CC3通道被配置为输入，IC3映射在TRC上。此模式仅工作在内部触发器输入被选中时（由TIM1_SMCR寄存器的TS位选择）。 注：CC3S仅在通道关闭时(TIM1_CCER寄存器的CC3E=0)才是可写的。

输入捕获模式：

位15:12	IC4F：输入捕获4滤波器
位11:10	IC4PSC[1:0]：输入/捕获4预分频器
位9:8	CC2S[1:0]：捕获/比较4选择。 这2位定义通道的方向（输入/输出），及输入脚的选择： 00：CC4通道被配置为输出； 01：CC4通道被配置为输入，IC4映射在TI4上； 10：CC4通道被配置为输入，IC4映射在TI3上； 11：CC4通道被配置为输入，IC4映射在TRC上。此模式仅工作在内部触发器输入被选中时（由TIM1_SMCR寄存器的TS位选择）。 注：CC4S仅在通道关闭时(TIM1_CCER寄存器的CC4E=0)才是可写的。
位7:4	IC3F[3:0]：输入捕获3滤波器
位3:2	IC3PSC[1:0]：输入/捕获3预分频器
位1:0	CC3S[1:0]：捕获/比较3选择。 这2位定义通道的方向（输入/输出），及输入脚的选择： 00：CC3通道被配置为输出； 01：CC3通道被配置为输入，IC3映射在TI3上； 10：CC3通道被配置为输入，IC3映射在TI4上； 11：CC3通道被配置为输入，IC1映射在TRC上。此模式仅工作在内部触发器输入被选中时（由TIM1_SMCR寄存器的TS位选择）。 注：CC3S仅在通道关闭时(TIM1_CCER寄存器的CC3E=0)才是可写的。

12.5.9 捕获/比较使能寄存器(TIM1_CCER)

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
保留		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，始终读为0。
位13	CC4P：输入/捕获4输出极性。参考CC1P的描述。
位12	CC4E：输入/捕获4输出使能。参考CC1E 的描述。

位11	CC3NP：输入/捕获3互补输出极性。参考CC1NP的描述。
位10	CC3NE：输入/捕获3互补输出使能。参考CC1NE的描述。
位9	CC3P：输入/捕获3输出极性。参考CC1P的描述。
位8	CC3E：输入/捕获3输出使能。参考CC1E 的描述。
位7	CC2NP：输入/捕获2互补输出极性。参考CC1NP的描述。
位6	CC2NE：输入/捕获2互补输出使能。参考CC1NE的描述。
位5	CC2P：输入/捕获2输出极性。参考CC1P的描述。
位4	CC2E：输入/捕获2输出使能。参考CC1E的描述。
位3	CC1NP：输入/捕获1互补输出极性 0：OC1N高电平有效 1：OC1N低电平有效 注：一旦LOCK级别(TIM1_BDTR寄存器中的LCCK位)设为3或2且CC1S=00(通道配置为输出)则该位不能被修改。
位2	CC1NE：输入/捕获1互补输出使能 0：关闭－ OC1N禁止输出，因此OC1N的输出电平依赖于MOE，OSSI，OSSR，OIS1，OIS1N，CC1E位的值。 1：开启－ OC1N信号输出到对应的输出引脚，其输出电平依赖于MOE，OSSI，OSSR，OIS1，OIS1N，CC1E位的值。
位1	CC1P：输入/捕获1输出极性 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： 该位选择是IC1还是IC1的反相信号作为触发或捕获信号。 0：不反相：捕获发生在IC1的上升沿；当用作外部触发器时，IC1不反相。 1：反相：捕获发生在IC1的下降沿；当用作外部触发器时，IC1反相。 注：一旦LOCK级别(TIM1_BDTR寄存器中的LCCK位)设为3或2，则该位不能被修改。
位0	**CC1E：**输入/捕获1输出使能 **CC1通道配置为输出：** 0： 关闭－ OC1禁止输出，因此OC1的输出电平依赖于MOE，OSSI，OSSR，OIS1，OIS1N，CC1NE位的值。 1： 开启－ OC1信号输出到对应的输出引脚，其输出电平依赖于MOE，OSSI，OSSR，OIS1，OIS1N，CC1NE位的值。 **CC1 通道配置为输入：** 该位决定了计数器的值是否能捕获入TIM1_CCR1寄存器。 0：捕获禁止； 0：捕获使能。

表37 带刹车功能的互补输出通道 OCx 和 OCxN 的控制位

控制位					输出状态	
MOE 位	**OSSI 位**	**OSSR 位**	**CCxE 位**	**CCxNE 位**	**OCx 输出状态**	**OCxN 输出状态**
1	X	0	0	0	输出禁止（与定时器断开）， OCx=CCxP ， OCx_EN=0	输出禁止（与定时器断开）， OCxN=CCxNP ， OCxN_EN=0
		0	0	1	输出禁止（与定时器断开）， OCx=CCxP ， OCx_EN=0	OCxREF + 极性， OCxN= OCxREF xor CCxNP， OCxN_EN=1
		0	1	0	OCxREF + 极性， OCx= OCxREF xor CCxP， OCx_EN=1	输出禁止（与定时器断开）， OCxN=CCxNP ， OCxN_EN=0
		0	1	1	OCxREF + 极性 + 死区， OCx_EN=1	OCxREF反相 + 极性 + 死区， OCxN_EN=1
		1	0	0	输出禁止（与定时器断开）， OCx=CCxP ， OCx_EN=0	输出禁止（与定时器断开）， OCxN=CCxNP ， OCxN_EN=0
		1	0	1	关闭状态（输出使能且为无效电平）， OCx=CCxP ， OCx_EN=1	OCxREF + 极性， OCxN= OCxREF xor CCxNP， OCxN_EN=1
		1	1	0	OCxREF + 极性， OCx= OCxREF xor CCxP， OCx_EN=1	关闭状态（输出使能且为无效电平）， OCxN=CCxNP ， OCxN_EN=1
		1	1	1	OCxREF + 极性 + 死区， OCx_EN=1	OCxREF反相 + 极性 + 死区， OCxN_EN=1
0	0	X	0	0	输出禁止（与定时器断开）； 异步地：OCx=CCxP ，OCx_EN=0 ，OCxN=CCxNP，OCxN_EN=0； 若时钟存在：经过一个死区时间后 OCx=OISx ，OCxN=OISxN ，假设 OISx 与 OISxN并不都对应OCx和OCxN的有效电平。	
	0		0	1		
	0		1	0		
	0		1	1		
	1		0	0	关闭状态（输出使能且为无效电平）； 异步地：OCx=CCxP ，OCx_EN=1 ，OCxN=CCxNP，OCxN_EN=1； 若时钟存在：经过一个死区时间后 OCx=OISx ，OCxN=OISxN ，假设 OISx 与 OISxN并不都对应OCx和OCxN的有效电平。	
	1		0	1		
	1		1	0		
	1		1	1		

注： 管脚连接到互补的 OCx 和 OCxN 通道的外部 I/O 管脚的状态，取决于 OCx 和 OCxN 通道状态和 GPIO 以及 AFIO 寄存器。

12.5.10 计数器(TIM1_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]：计数器的值

12.5.11 预分频器(TIM1_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]：预分频器的值 计数器的时钟频率（CK_CNT）等于f_{CK_PSC}/(PSC[15:0]+1)。 PSC包含了当更新事件产生时装入当前预分频器寄存器的值；更新事件包括计数器被TIM_EGR的UG位清0或被工作在复位模式的从控制器清0。

12.5.12 自动重装载寄存器(TIM1_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]: 自动重装载的值 ARR包含了将要装载入实际的自动重装载寄存器的值。 详细参考12.4.1：时基单元有关ARR的更新和动作。 当自动重装载的值为空时，计数器不工作。

12.5.13 周期计数寄存器(TIM1_RCR)

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
保留								REP[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

位15:8	保留，始终读为0。
位7:0	REP[7:0]: 周期计数器的值 开启了预装载功能后，这些位允许用户设置比较寄存器的更新速率（即周期性地从预装载寄存器传输到当前寄存器）；如允许产生更新中断，则会同时影响产生更新中断的速率。 每次向下计数器REP_CNT达到0，会产生一个更新事件并且计数器REP_CNT重新从REP值开始计数。由于REP_CNT只有在周期更新事件U_RC发生时才重载REP值，因此对TIM1_RCR寄存器写入的新值只在下次周期更新事件发生时才起作用。 这意味着在PWM模式中，(REP+1)对应着： — 在边沿对齐模式下，PWM周期的数目； — 在中心对称模式下，PWM半周期的数目；

12.5.14 捕获/比较寄存器 1(TIM1_CCR1)

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的值 **若CC1通道配置为输出：** CCR1包含了装入当前捕获/比较1寄存器的值（预装载值）。 如果在TIM1_CCMR1寄存器(OC1PE位)中未选择预装载特性，其始终装入当前寄存器中。否则，只有当更新事件发生时，此预装载值才装入当前捕获/比较1寄存器中。 当前捕获/比较寄存器包含了与计数器TIM1_CNT比较的值，并且在OC1端口上输出信号。 **若CC1通道配置为输入：** CCR1包含了由上一次输入捕获1事件（IC1）传输的计数器值。

12.5.15 捕获/比较寄存器 2(TIM1_CCR2)

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的值 **若CC2通道配置为输出：** CCR2包含了装入当前捕获/比较2寄存器的值（预装载值）。 如果在TIM1_CCMR2寄存器(OC2PE位)中未选择预装载特性，其始终装入当前寄存器中。否则，只有当更新事件发生时，此预装载值才装入当前捕获/比较2寄存器中。 当前捕获/比较寄存器包含了与计数器TIM1_CNT比较的值，并且在OC端口上输出信号。 **若CC2通道配置为输入：** CCR2包含了由上一次输入捕获2事件（IC2）传输的计数器值。

12.5.16 捕获/比较寄存器 3(TIM1_CCR3)

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3的值 **若CC3通道配置为输出：** CCR3包含了装入当前捕获/比较3寄存器的值（预装载值）。 如果在TIM1_CCMR3寄存器(OC3PE位)中未选择预装载特性，其始终装入当前寄存器中。否则，只有当更新事件发生时，此预装载值才装入当前捕获/比较3寄存器中。 当前捕获/比较寄存器包含了与计数器TIM1_CNT比较的值，并且在OC端口上输出信号。 **若CC3通道配置为输入：** CCR3包含了由上一次输入捕获3事件（IC3）传输的计数器值。

12.5.17 捕获/比较寄存器(TIM1_CCR4)

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4的值 **若CC4通道配置为输出：** CCR4包含了装入当前捕获/比较4寄存器的值（预装载值）。 如果在TIM1_CCMR4寄存器(OC4PE位)中未选择预装载特性，其始终装入当前寄存器中。否则，只有当更新事件发生时，此预装载值才装入当前捕获/比较4寄存器中。 当前捕获/比较寄存器包含了与计数器TIM1_CNT比较的值，并且在OC端口上输出信号。 **若CC4通道配置为输入：** CCR4包含了由上一次输入捕获4事件（IC4）传输的计数器值。

12.5.18 刹车和死区寄存器(TIM1_BDTR)

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

注释： 根据锁定设置，AOE, BKP, BKE, OSSI, OSSR 和DTG[7:0]位均可被写保护，有必要在第一次写入 TIM1_BDTR 寄存器时对它们进行配置。

位15	MOE: 主输出使能 一旦刹车输入有效，该位被硬件异步清0。根据AOE位的值，可由软件清0或自动置1。它仅对配置为输出通道有效。 0：禁止OC和OCN输出或强制为空闲状态； 1：如果设置了相应的使能位（TIM1_CCER寄存器的CCxE、CCxNE位），则开启OC和OCN输出。 参考OC/OCN使能的详细描写（12.5.9节，捕获/比较使能寄存器(TIM1_CCER)）。
位14	AOE: 自动输出使能 0：MOE只能被软件置1； 1：MOE能被软件置1或在下一个更新事件自动置1（如果刹车输入无效）。 注：一旦LOCK级别(TIM1_BDTR寄存器中的LOCK位)设为1，则该位不能被修改。

位13	BKP: 刹车输入极性 0：刹车输入低电平有效； 1：刹车输入高电平有效。 注：一旦LOCK级别(TIM1_BDTR寄存器中的LOCK位)设为1，则该位不能被修改。
位12	BKE: 刹车功能使能 0：禁止刹车输入（BRK及BRK_ACTH）； 1：开启刹车输入（BRK及BRK_ACTH）。 注：一旦LOCK级别(TIM1_BDTR寄存器中的LOCK位)设为1，则该位不能被修改。
位11	OSSR: 运行模式下“关闭状态”选择 该位用于当MOE=1且通道为互补输出时。没有互补输出的定时器中不存在OSSR位。 参考OC/OCN使能的详细说明（12.5.9节，捕获/比较使能寄存器(TIM1_CCER)）。 0：当定时器不工作时，禁止OC/OCN输出（OC/OCN使能输出信号=0）； 1：当定时器不工作时，一旦CCxE=1或CCxNE=1，开启OC/OCN输出并输出无效电平。OC/OCN使能输出信号=1。 注：一旦LOCK级别(TIM1_BDTR寄存器中的LOCK位)设为2，则该位不能被修改。
位10	OSSI: 空闲模式下“关闭状态”选择 该位用于当MOE=0且通道设为输出时。 参考OC/OCN使能的详细说明（12.5.9节，捕获/比较使能寄存器(TIM1_CCER)）。 0：当定时器不工作时，禁止OC/OCN输出（OC/OCN使能输出信号=0）； 1：当定时器不工作时，一旦CCxE=1或CCxNE=1，OC/OCN首先输出其空闲电平。OC/OCN使能输出信号=1。 注：一旦LOCK级别(TIM1_BDTR寄存器中的LOCK位)设为2，则该位不能被修改。
位9:8	LOOK[1:0]: 锁定设置 该位为防止软件错误而提供写保护。 00：锁定关闭，寄存器无写保护； 01：锁定级别1，不能写入TIM1_BDTR寄存器的DTG/BKE/BKP/AOE位、TIM1_CR2寄存器的OISx/OISxN位； 10：锁定级别2，不能写入锁定级别1中的各位，也不能写入CC极性位（一旦相关通道通过CCxS位设为输出，TIM1_CCER寄存器的CCxP/CCNxP位）以及OSSR/OSSI位； 11：锁定级别3，不能写入锁定级别2中的各位，也不能写入CC控制位（一旦相关通道通过CCxS位设为输出，TIM1_CCMRx寄存器的OCxM/OCxPE位）； 注：在系统复位后，只能写一次LOCK位，一旦写入TIM1_BDTR寄存器，则其内容冻结直至复位。
位7:0	UTG[7:0]: 死区发生器设置 这些位定义了插入互补输出之间的死区持续时间。假设DT表示其持续时间： DTG[7:5]=0xx => DT=DTG[7:0] × T_{dtg}，T_{dtg} = T_{DTS}； DTG[7:5]=10x => DT=(64+DTG[5:0]) × T_{dtg}，T_{dtg} = 2 × T_{DTS}； DTG[7:5]=110 => DT=(32+DTG[4:0]) × T_{dtg}，T_{dtg} = 8 × T_{DTS}； DTG[7:5]=111 => DT=(32+DTG[4:0]) × T_{dtg}，T_{dtg} = 16 × T_{DTS}； 例：若T_{DTS} = 125ns(8MHZ)，可能的死区时间为： 0到15875ns，若步长时间为125ns； 16us到31750ns，若步长时间为250ns； 32us到63us，若步长时间为1us； 64us到126us，若步长时间为2us； 注：一旦LOCK级别(TIM1_BDTR寄存器中的LOCK位)设为1、2或3，则这些位不能被修改。

12.5.19 DMA控制寄存器(TIM1_DCR)

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
保留			DBL[4:0]					保留			DBA[4:0]				
			rw	rw	rw	rw	rw				rw	rw	rw	rw	rw

位15:13	保留，始终读为0。
位12:8	DBL[4:0]: DMA连续传送长度 这些位定义了DMA在连续模式下的传送长度（当对TIM1_DMAR寄存器的地址进行读或写时，定时器则进行一次连续传送），即：定义被传送的字节数目： 00000：1 字节　　00001：2 字节 00010：3 字节　　..... .....　　10001：18 字节
位7:5	保留，始终读为0。
位4:0	DBA[4:0]: DMA基地址 这些位定义了DMA在连续模式下的基地址（当对TIM1_DMAR寄存器的地址进行读或写时），DBA定义为从TIM1_CR1寄存器所在地址开始的偏移量： 00000：TIM1_CR1， 00001：TIM1_CR2， 00010：TIM1_SMCR， .....

12.5.20 连续模式的DMA地址(TIM1_DMAR)

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 对TIM1_DMAR寄存器的读或写会导致对以下地址的寄存器的存取操作： TIM1_CR1地址 + DBA + DMA指针，其中： “TIM1_CR1地址”是控制寄存器1的地址； “DBA”是TIM1_DCR寄存器中定义的基地址； “DMA指针”是由DMA自动控制的偏移量，它取决于TIM1_DCR寄存器中定义的DBL。

12.6TIM1 寄存器图

下表中将 TIM1 的所有寄存器映射到一个 16 位可寻址(编址)空间。

表38　TIM1 – 寄存器图和复位值

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
000h	TIM1_CR1	保留																						CKD[1:0]		ARPE	CMS[1:0]		DIR	OPM	URS	UDIS	CEN
	复位值																							0	0	0	0	0	0	0	0	0	0
004h	TIM1_CR2	保留																	OIS4	OIS3N	OIS3	OIS2N	OIS2	OIS1N	OIS1	TI1S	MMS[2:0]			CCDS	CCUS	保留	CCPC
	复位值																		0	0	0	0	0	0	0	0	0	0	0	0	0		0
008h	TIM1_SMCR	保留																ETP	ECE	ETPS[1:0]		EFT[3:0]				MSM	TS[2:0]			保留	SMS[2:0]		
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0
00Ch	TIM1_DIER	保留																	TDE	COMDE	CC4DE	CC3DE	CC2DE	CC1DE	UDE	BIE	TIE	COMIE	CC4IE	CC3IE	CC2IE	CC1IE	UIE
	复位值																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
010h	TIM1_SR	保留																				CC4OF	CC3OF	CC2OF	CC1OF	保留	BIF	TIF	COMIF	CC4IF	CC3IF	CC2IF	CC1IF	UIF
	复位值																					0	0	0	0		0	0	0	0	0	0	0	0
014h	TIM1_EGR	保留																								BG	TG	COM	CC4G	CC3G	CC2G	CC1G	UG
	复位值																									0	0	0	0	0	0	0	0
018h	TIM1_CCMR1 输出比较模式	保留																OC2CE	OC2M[2:0]			OC2PE	OC2FE	CC2S[1:0]		OC1CE	OC1M[2:0]			OC1PE	OC1FE	CC1S[1:0]	
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	TIM1_CCMR1 输入捕获模式	保留																IC2F[3:0]				IC2PSC[1:0]		CC2S[1:0]		IC1F[3:0]				IC1PSC[1:0]		CC1S[1:0]	
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
01Ch	TIM1_CCMR2 输出比较模式	保留																OC4CE	OC4M[2:0]			OC4PE	OC4FE	CC4S[1:0]		OC3CE	OC3M[2:0]			OC3PE	OC3FE	CC3S[1:0]	
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	TIM1_CCMR2 输入捕获模式	保留																IC4F[3:0]				IC4PSC[1:0]		CC4S[1:0]		IC3F[3:0]				IC3PSC[1:0]		CC3S[1:0]	
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
020h	TIM1_CCER	保留																		CC4P	CC4E	CC3NP	CC3NE	CC3P	CC3E	CC2NP	CC2NE	CC2P	CC2E	CC1NP	CC1NE	CC1P	CC1E
	复位值																			0	0	0	0	0	0	0	0	0	0	0	0	0	0
024h	TIM1_CNT	保留																CNT[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
028h	TIM1_PSC	保留																PSC[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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
02Ch	TIM1_ARR	保留																ARR[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
030h	TIM1_RCR	保留																								REP[7:0]							
	复位值																									0	0	0	0	0	0	0	0
034h	TIM1_CCR1	保留																CCR1[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
038h	TIM1_CCR2	保留																CCR2[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
03Ch	TIM1_CCR3	保留																CCR3[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
040h	TIM1_CCR4	保留																CCR4[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
044h	TIM1_BDTR	保留																MOE	AOE	BKP	BKE	OSSR	OSSI	LOCK[1:0]		DT[7:0]							
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
048h	TIM1_DCR	保留																			DBL[4:0]					保留	DBA[4:0]						
	复位值																				0	0	0	0	0		0	0	0	0	0	0	0
04Ch	TIM1_DMAR	保留																DMAB[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

13 通用定时器(TIMx)

13.1 概述

通用定时器是一个通过可编程预分频器驱动的 16 位自动装载计数器构成。

它适用于多种场合，包括测量输入信号的脉冲长度(输入采集)或者产生输出波形(输出比较和 PWM)。

使用定时器预分频器和 RCC 时钟控制器预分频器，脉冲长度和波形周期可以在几个微秒到几个毫秒间调整。

定时器是完全独立的，而且没有互相共享任何资源。它们可以一起同步操作，描述见 14.4.14

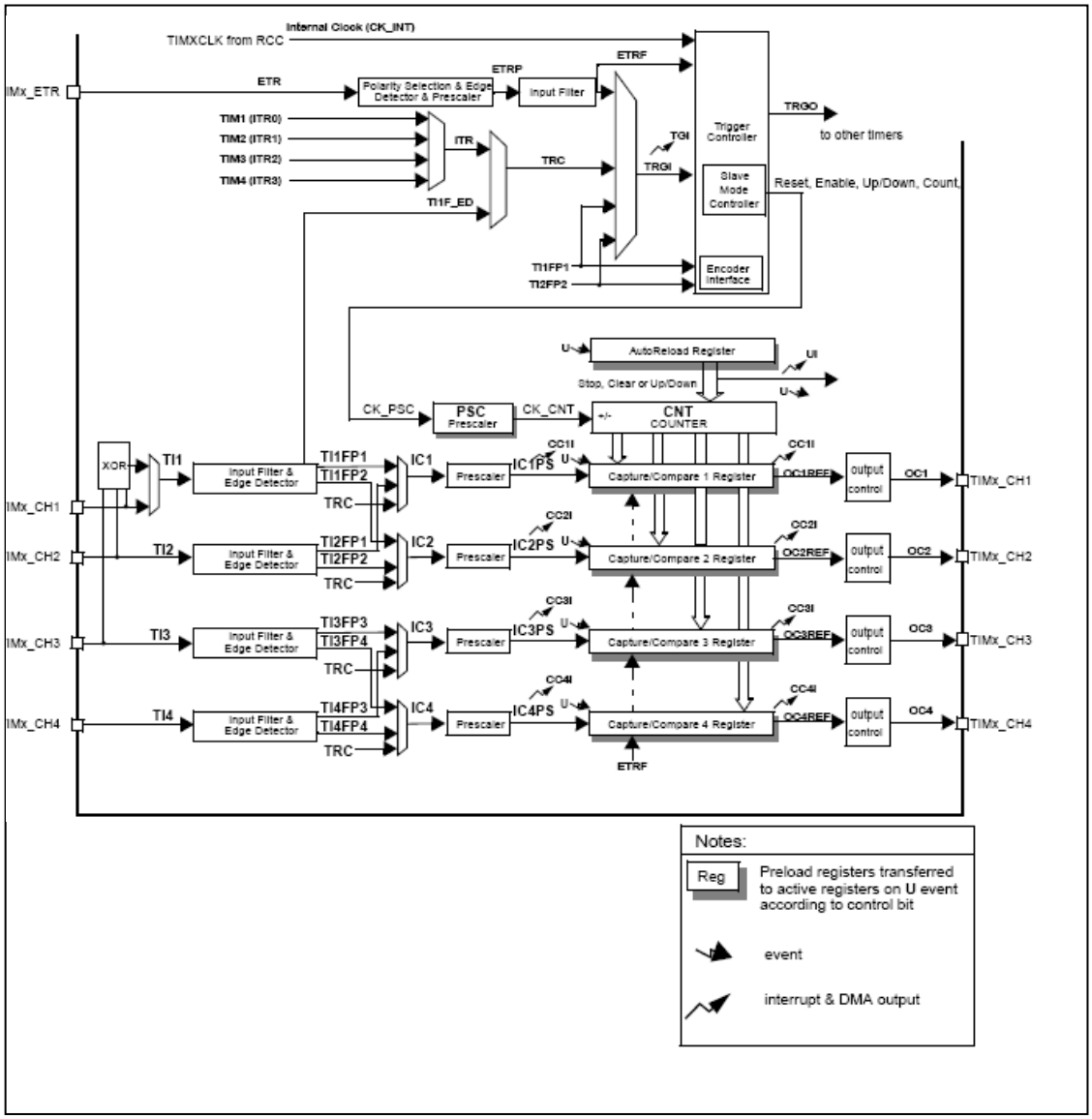
13.2 主要特性

通用 TIMx 定时器特性包括：

- 16 位向上，向下，向上/向下自动装载计数器
- 16 位可编程预分频器，计数器时钟频率的分频系数为 1～65535 之间的任意数值
- 4 个独立通道：
  - 输入捕获
  - 输出比较
  - PWM 生成(边缘或中间对齐模式)
  - 单脉冲模式输出
- 使用外部信号控制定时器和定时器互连的同步电路
- 如下事件发生时产生中断/DMA：
  - 更新：计数器向上溢出/向下溢出，计数器初始化(通过软件或者内部/外部触发)
  - 触发事件(计数器启动，停止，初始化或者由内部/外部触发计数)
  - 输入捕获
  - 输出比较

13.3框图

图73 通用定时器框图



13.4功能描述

13.4.1 时基单元

可编程通用定时器的主要部分是一个 16 位计数器和与其相关的自动装载寄存器。这个计数器可以向上计数、向下计数或者向上向下双向计数。此计数器时钟由预分频器分频得到。

计数器、自动装载寄存器和预分频器寄存器可以由软件读写，即使计数器还在运行读写仍然有效。

时基单元包含：

- 计数器寄存器(TIMx_CNT)
- 预分频器寄存器 (TIMx_PSC)
- 自动装载寄存器 (TIMx_ARR)

自动装载寄存器是预先装载的。根据在 TIMX_CR1 寄存器中的自动装载预装载使能位(ARPE)的设置，预装载寄存器的内容被永久地或在每次的更新事件 UEV 时传送到影子寄存器。当计数器达到溢出条件(向下计数时的下溢条件)并当 TIMX_CR1 寄存器中的 UDIS 位等于 0 时，产生更新事件。更新事件也可以由软件产生。随后会详细描述每一种配置下更新事件的产生。

计数器由预分频器的时钟输出 CK_CNT 驱动，仅当设置了计数器 TIMX_CR1 寄存器中的计数器使能位(CEN)时，CK_CNT 才有效。(有关更多的计数器使能的细节，请参见控制器的从模式描述)。

注：真正的计数器使能信号 CNT_EN 是在 CEN 后的一个时钟周期后被设置。

预分频器描述

预分频器可以将计数器的时钟频率按 1 到 65536 之间的任意值分频。它是基于一个(在 TIMx_PSC 寄存器中的)16 位寄存器控制的 16 位计数器。因为这个控制寄存器带有缓冲器，它能够在工作时被改变。新的预分频器的参数在下一次更新事件到来时被采用。

图 74 和 图 75 给出了一些在预分频器工作时，更改其参数的情况下计数器操作的例子。

图74 当预分频器的参数从 1 变到 2 时，计数器的时序图

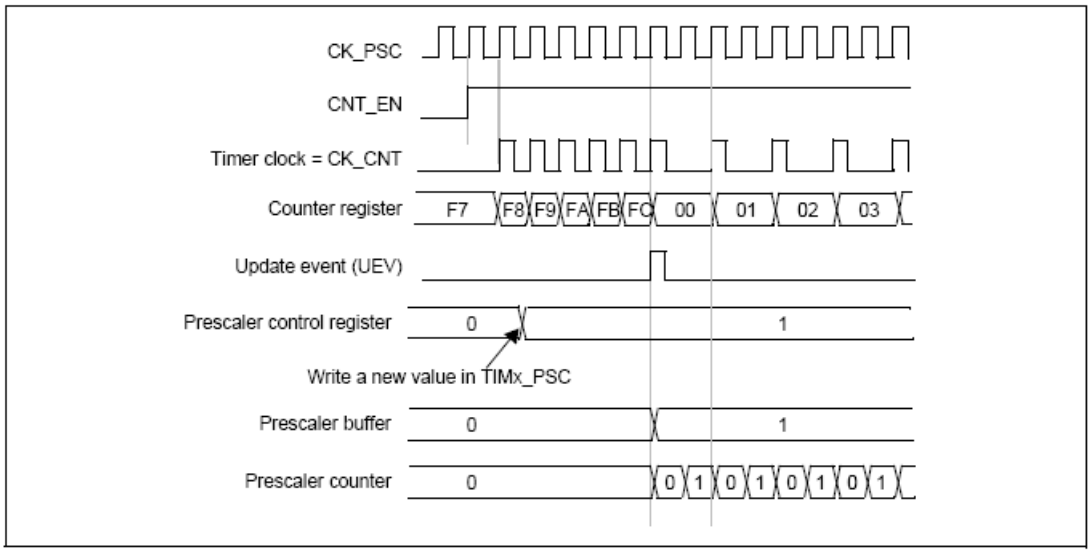
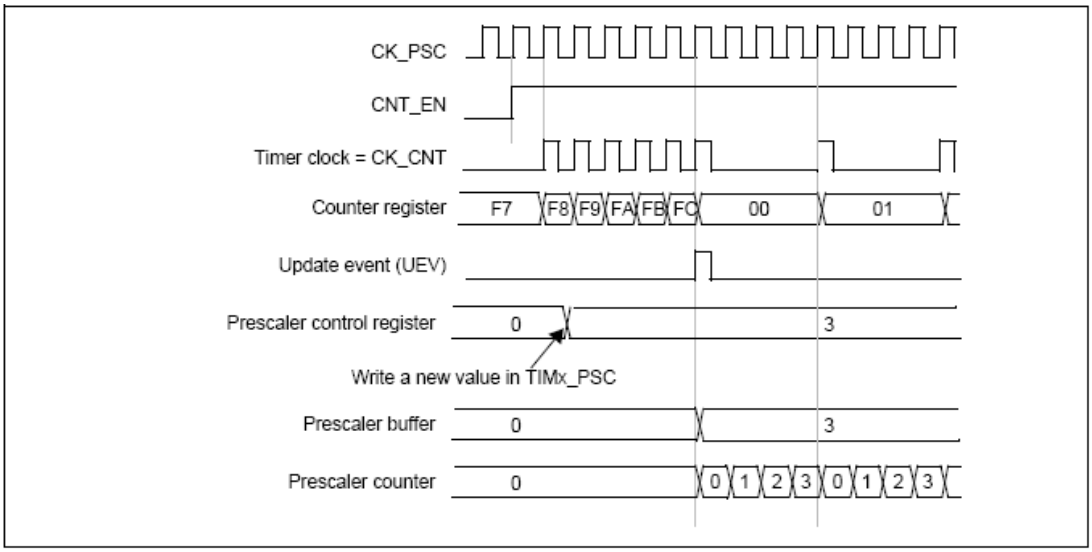


图75 当预分频器的参数从 1 变到 4 时，计数器的时序图



13.4.2 计数器模式

向上计数模式

在向上计数模式中，计数器从 0 计数到自动加载值(TIMx_ARR 计数器的内容)，然后重新从 0 开始计数并且产生一个计数器溢出事件。

每次计数器溢出时可以产生更新事件，在 TIMx_EGR 寄存器中设置 UG 位(通过软件方式或者使用从模式控制器)也同样可以产生一个更新事件。

通过软件设置 TIMx_CR1 寄存器中的 UDIS 位，将禁止更新事件；这样可以避免在向预装载寄存器中写入新值时更新影子寄存器。在 UDIS 位被清成 0 之前，将没有更新事件产生。即使这样，在应该产生更新事件时，计数器仍会被清 0，同时预分频器的计数也被请 0(但预分频器的数值不变)。此外，如果 TIMx_CR1 寄

存器中的 URS 位(更新请求选择)被设置，设置 UG 位将产生一个更新事件 UEV，但硬件不设置 UIF 标志(即不产生中断或者 DMA 请求)。这是为了避免在捕获模式下清除计数器时，同时产生更新和捕获中断。

当发生一个更新事件，所有的寄存器都被更新，硬件同时(依据 URS 位)设置更新标志位(TIMx_SR 寄存器中的 UIF 位)。

- 预分频器的缓冲区被置入预装载寄存器的值(TIMx_PSC 寄存器的内容)。
- 自动装载影子寄存器被重新置入预装载寄存器的值(TIMx_ARR)。

下图给出一些例子，当 TIMx_ARR=0x36 时计数器在不同时钟频率下的动作。

图76 计数器时序图，内部时钟分频因子为 1

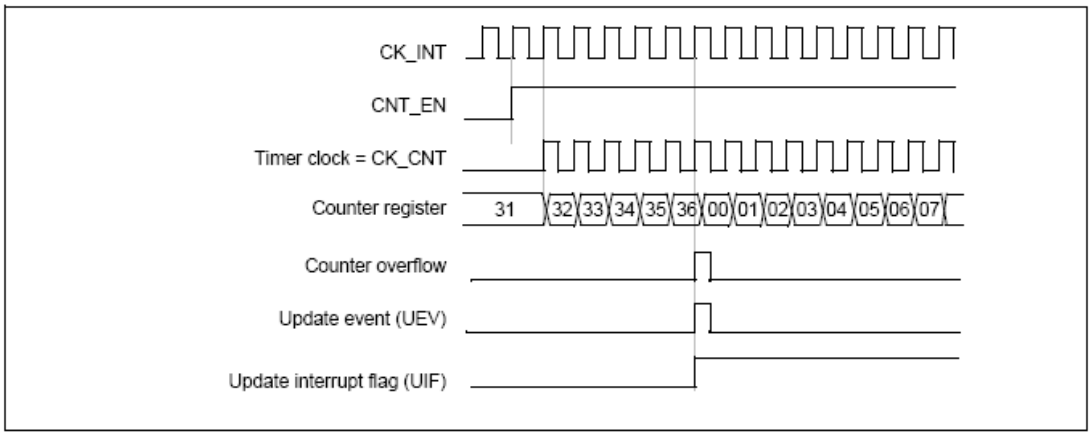


图77 计数器时序图，内部时钟分频因子为 2

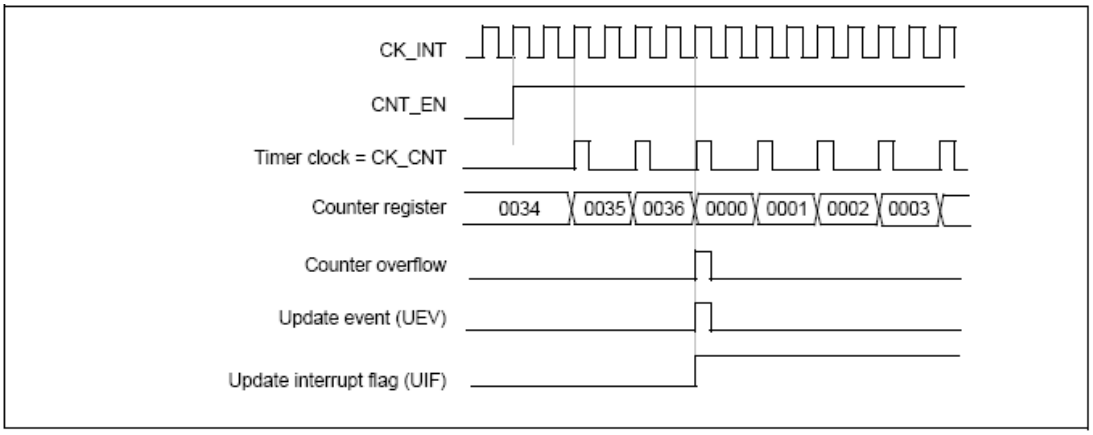


图78 计数器时序图，内部时钟分频因子为 4

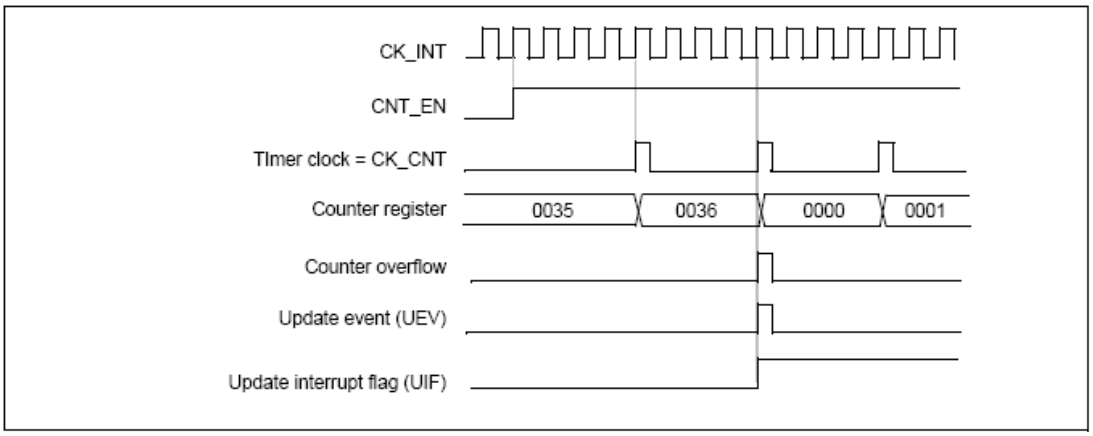


图79 计数器时序图，内部时钟分频因子为 N

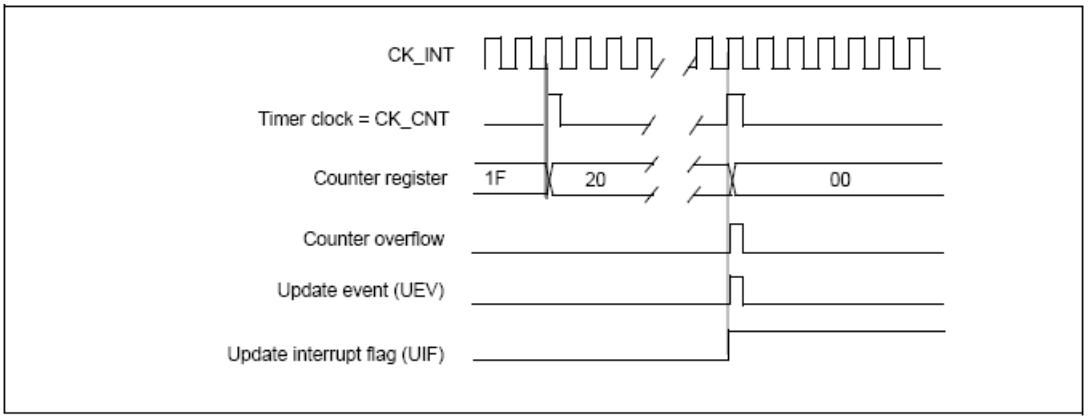


图80 计数器时序图，当 ARPE=0 时的更新事件(TIMx_ARR 没有预装入)

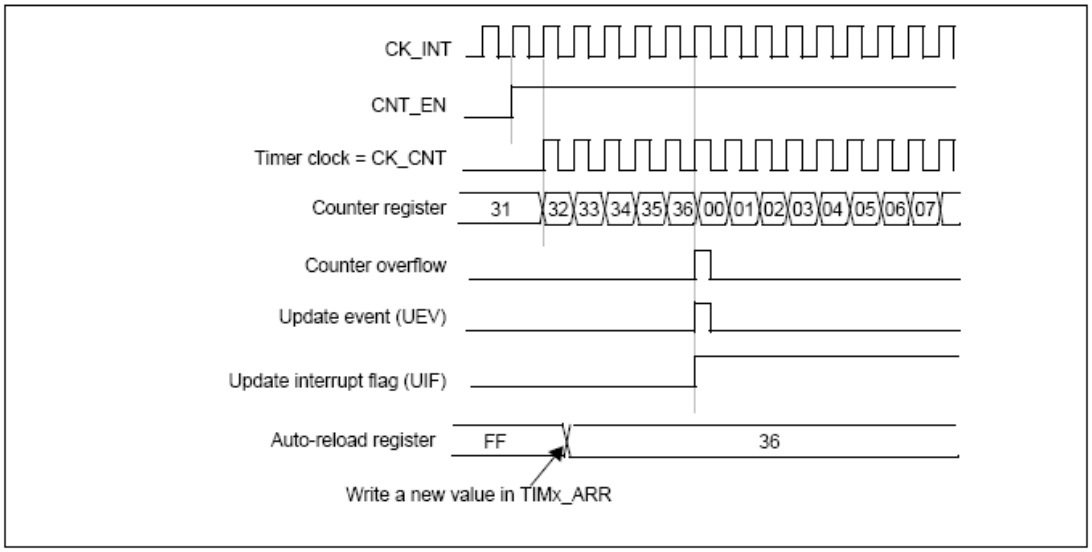
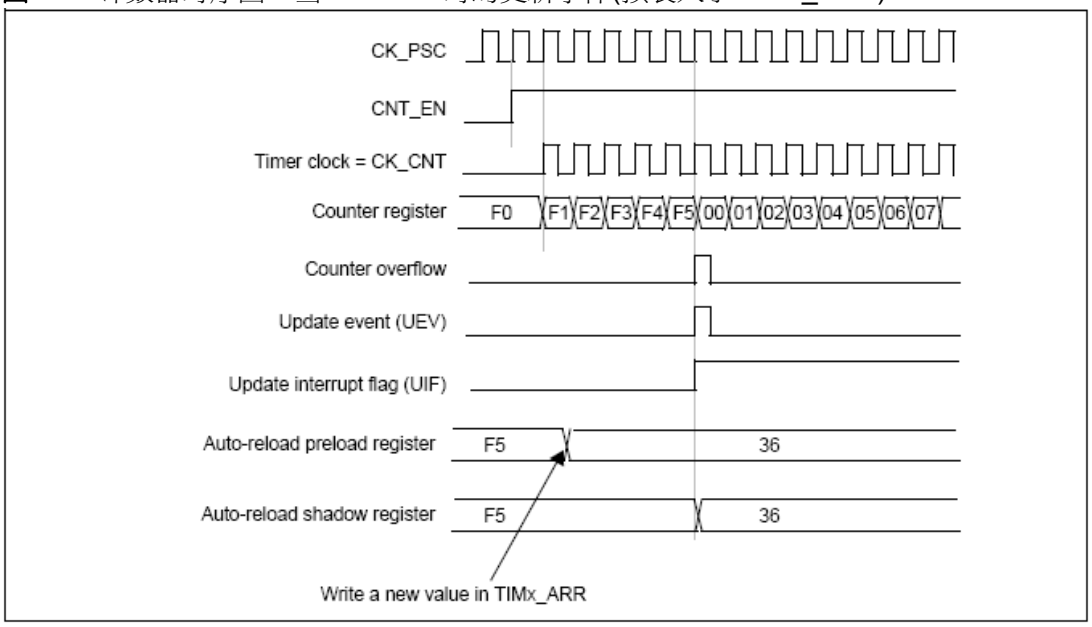


图81 计数器时序图，当 ARPE=1 时的更新事件(预装入了 TIMx_ARR)



向下计数模式

在向下模式中，计数器从自动装入的值(TIMx_ARR 计数器的值)开始向下计数到 0，然后从自动装入的值重新开始并且产生一个计数器向下溢出事件。

每次计数器溢出时可以产生更新事件，在 TIMx_EGR 寄存器中设置 UG 位(通过软件方式或者使用从模式控制器)也同样可以产生一个更新事件。

UEV 事件可以通过软件设置 TIMx_CR1 寄存器中的 UDIS 位被禁止。这样可以避免在向预装载寄存器中写入新值时更新影子寄存器。这样 UDIS 位被写成 0 之前不会产生更新事件。然而，计数器仍会从当前自动加载值重新开始计数，并且预分频器的计数器重新从 0 开始(但预分频器的速率不能被修改)。

此外，如果设置了 TIMx_CR1 寄存器中的 URS 位(更新请求选择) ，设置 UG 位将产生一个更新事件 UEV 但不设置 UIF 标志(因此不产生中断和 DMA 请求)，这是为了避免在发生捕获事件并清除计数器时，同时产生更新和捕获中断。

当发生更新事件时，所有的寄存器都被更新，并且(根据 URS 位的设置)更新标志位(TIMx_SR 寄存器中的 UIF 位)也被设置。

- 预分频器的缓冲区被置入预装载寄存器的值(TIMx_PSC 寄存器的内容)。
- 当前的自动加载寄存器被更新为预装载值(TIMx_ARR 寄存器中的内容)。
  注：自动装载在计数器重载入之前被更新，因此下一个周期将是预期的值。

以下的图显示一些当 TIMx_ARR=0x36 时计数器在不同时钟频率下的操作例子。

图82 计数器时序图，内部时钟分频因子为 1

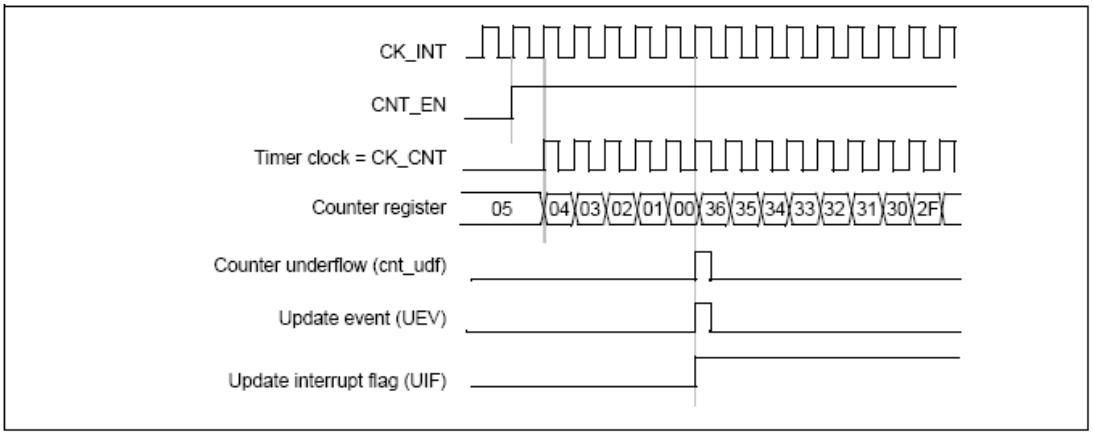


图83 计数器时序图，内部时钟分频因子为 2

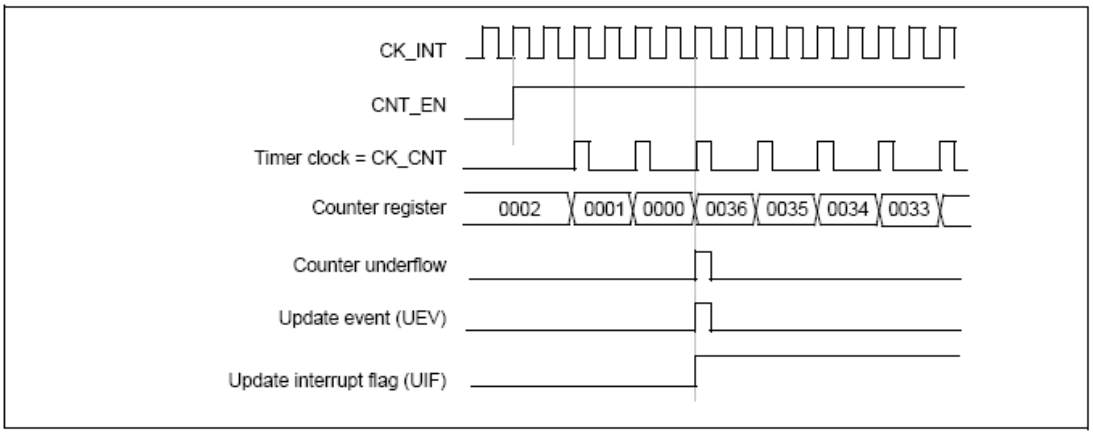


图84 计数器时序图，内部时钟分频因子为 4

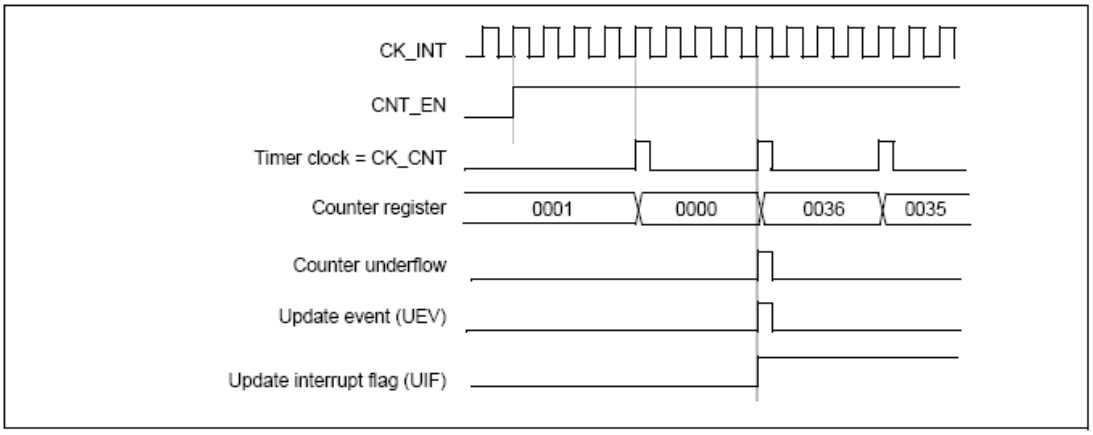


图85 计数器时序图，内部时钟分频因子为 N

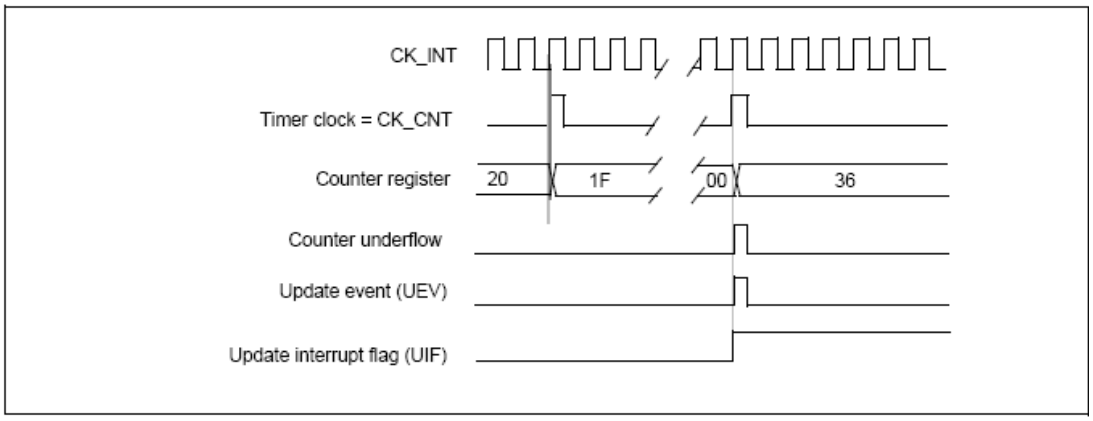
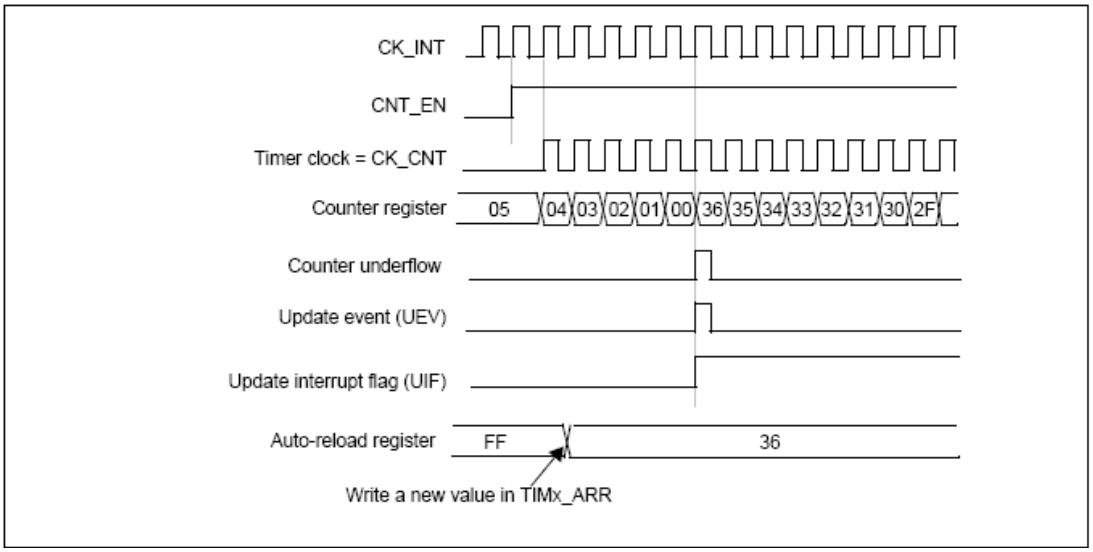


图86 计数器时序图，当没有使用周期计数器时的更新事件



中央对齐模式(向上/向下计数)

在中央对齐模式中，计数器从 0 开始计数到自动加载的值(TIMx_ARR 寄存器的内容)-1，产生一个计数器溢出事件，然后向下计数到 1 并且产生一个计数器下溢事件；然后再从 0 开始重新计数。

在这个模式下，不能写入 TIMx_CR1 中的 DIR 方向位。它由硬件更新并指示当前的计数方向。

更新事件可以产生在每一次计数溢出和每一次计数下溢；也可以通过(软件或者使用从模式控制器)设置 TIMx_EGR 寄存器中的 UG 位来产生更新事件，此时，计数器重新从 0 开始计数，预分频器也重新从 0 开始计数。

UEV 事件可以通过软件设置 TIMx_CR1 寄存器中的 UDIS 位被禁止。这样可以避免在向预装载寄存器中写入新值时更新影子寄存器。这样 UDIS 位被写成 0 之前不会产生更新事件。然而，计数器仍会根据当前自动重加载的值，继续向上或向下计数。
此外，如果设置了 TIMx_CR1 寄存器中的 URS 位(更新请求选择) ，设置 UG 位将产生一个更新事件 UEV 但不设置 UIF 标志(因此不产生中断和 DMA 请求)，这是为了避免在发生捕获事件并清除计数器时，同时产生更新和捕获中断。

当发生更新事件时，所有的寄存器都被更新，并且(根据 URS 位的设置)更新标志位(TIMx_SR 寄存器中的 UIF 位)也被设置。

- 预分频器的缓冲区被置入预装载寄存器的值(TIMx_PSC 寄存器的内容)。
- 当前的自动加载寄存器被更新为预装载值(TIMx_ARR 寄存器中的内容)。
  注：如果因为计数器溢出而产生更新，自动重装载将在计数器重载入之前被更新，因此下一个周期将是预期的值(计数器被装载为新的值)。

以下的图显示一些计数器在不同时钟频率下的操作的例子：

图87 计数器时序图，内部时钟分频因子为 1，TIMx_ARR=0x6

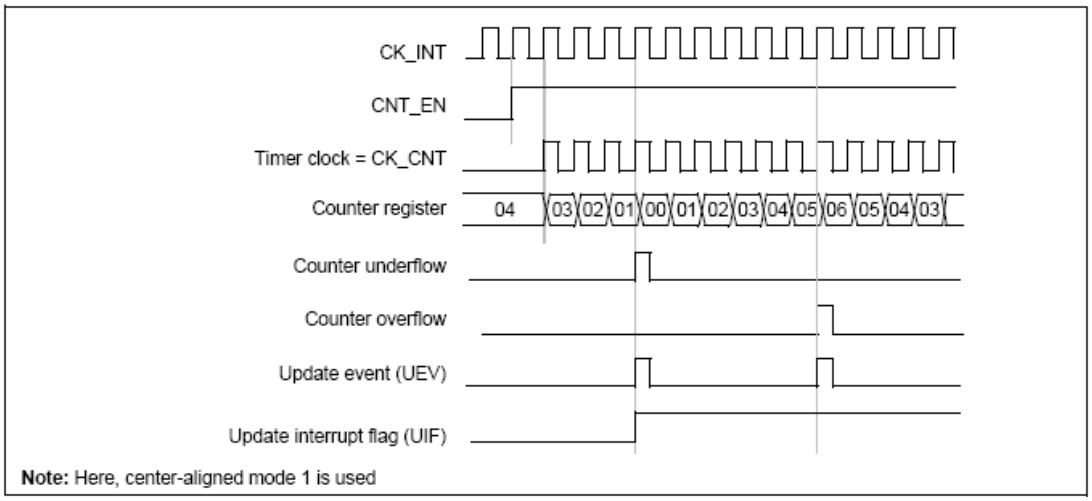


图88 计数器时序图，内部时钟分频因子为 2

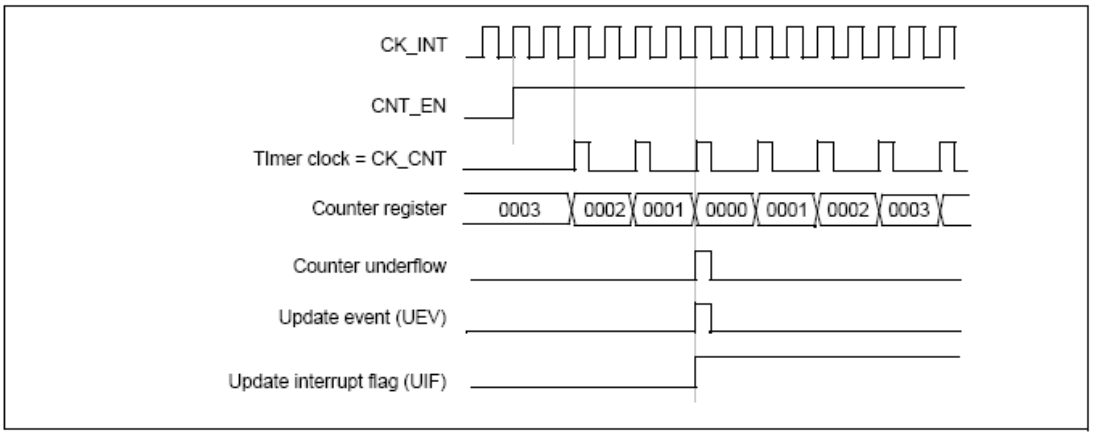


图89 计数器时序图，内部时钟分频因子为 4，TIMx_ARR=0x36

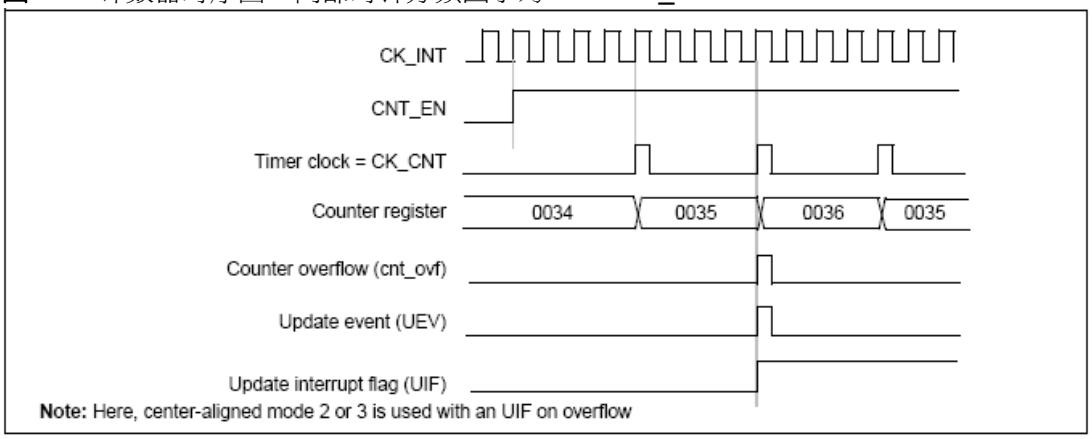


图90 计数器时序图，内部时钟分频因子为 N

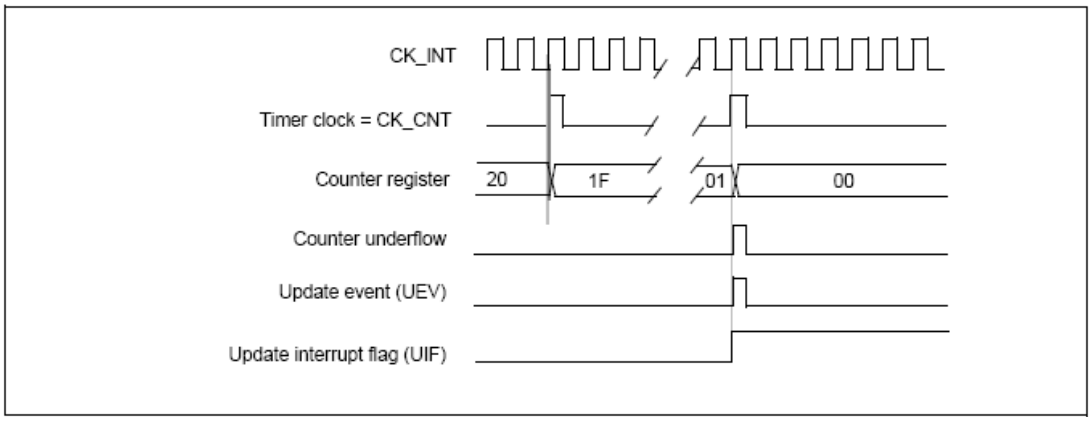


图91 计数器时序图，ARPE=1 时的更新事件(计数器下溢)

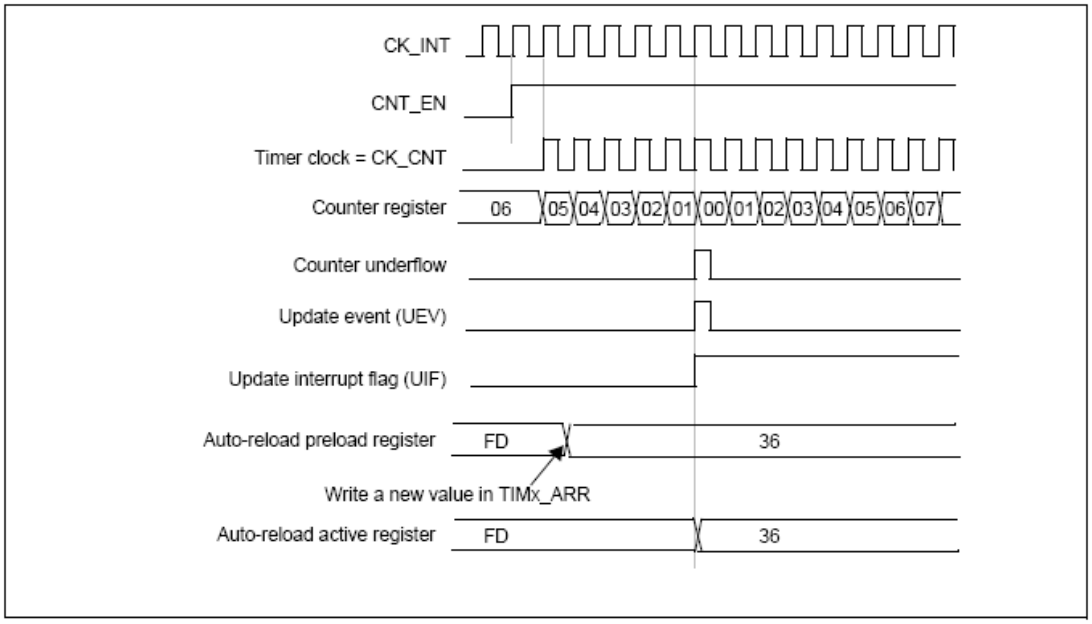
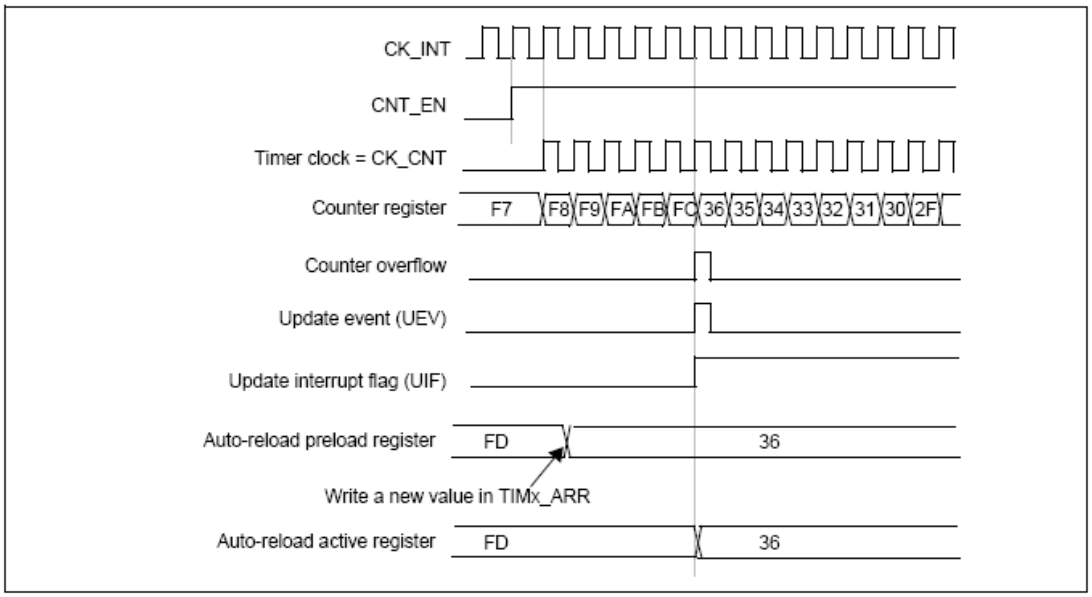


图92 计数器时序图，ARPE=1 时的更新事件(计数器溢出)



13.4.3 时钟选择

计数器时钟可由下列时钟源提供：

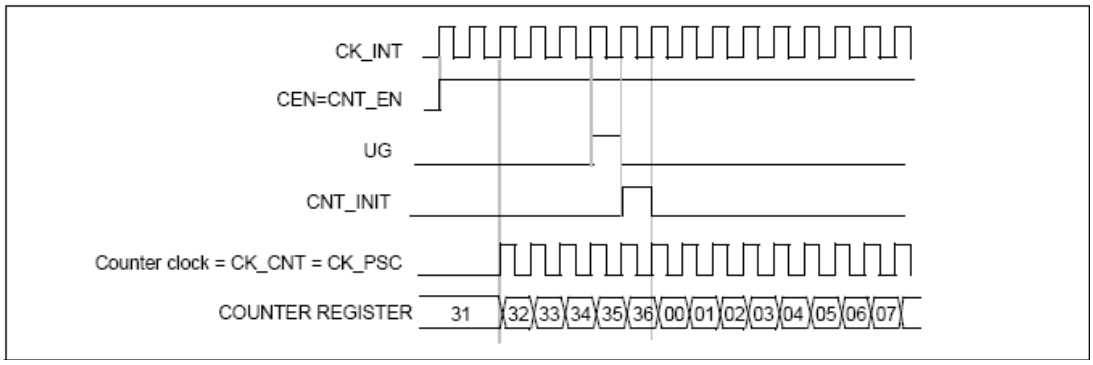
- 内部时钟(CK_INT)
- 外部时钟模式 1：外部输入脚(TIx)
- 外部时钟模式 2：外部触发输入(ETR)
- 内部触发输入(ITRx)：使用一个定时器作为另一个定时器的预分频器，例如你可以配置一个定时器Timer1 而作为另一个定时器Timer2 的预分频器。参见 13.4.15。

内部时钟源(CK_INT)

如果从模式控制器被禁止(SMS=000)，则 CEN、DIR(TIMx_CR1 寄存器中)和 UG 位(TIMx_EGR 寄存器中)是事实上的控制位同时只能被软件修改(UG 位仍被自动清除)。一旦 CEN 位被写成 1，预分频器的时钟就由内部时钟 CK_INT 提供。

图 93 显示了控制电路和向上计数器在一般模式下，不带预分频器时的操作。

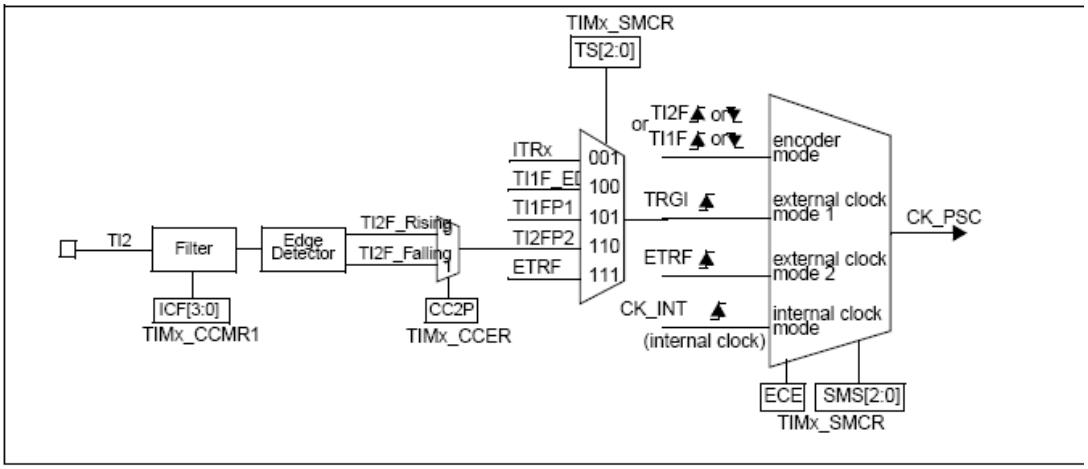
图93 一般模式下的控制电路，内部时钟分频因子为 1



外部时钟源模式 1

当 TIMx_SMCR 寄存器中的 SMS=111 时，此模式被选中。计数器可以在选定的输入上的每个上升沿或下降沿计数。

图94 TI2 外部时钟连接例子



例如，要配置向上计数器在 T12 输入端的上升沿计数，使用下列步骤：

1. 配置TIMx_CCMR1寄存器CC2S=01，配置通道2检测TI2输入的上升沿
2. 配置TIMx_CCMR1寄存器的IC2F[3:0]，选择输入滤波器带宽(如果不需要滤波器，保持IC2F=0000)

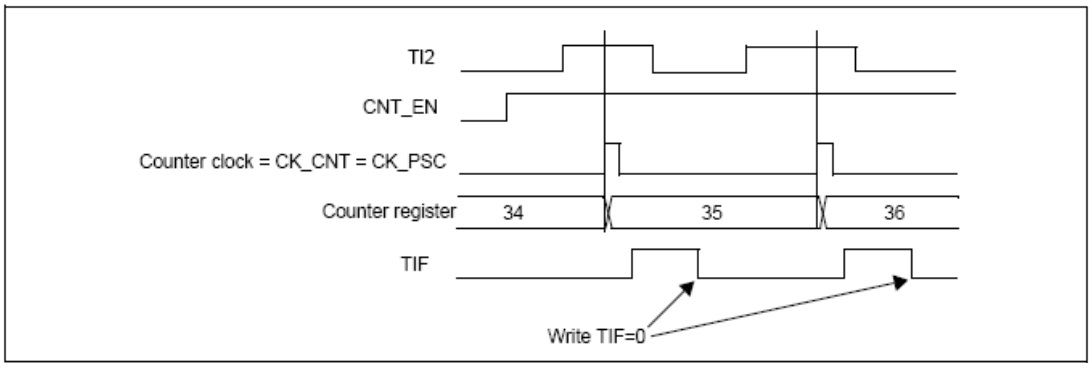
注： *捕获预分频器不用作触发，所以不需要对它进行配置*

3. 配置TIMx_CCER寄存器的CC2P=0，选定上升沿极性
4. 配置TIMx_SMCR寄存器的SMS=111，选择定时器外部时钟模式1
5. 配置TIMx_SMCR寄存器中的TS=110，选定TI2作为触发输入源
6. 设置TIMx_CR1寄存器的CEN=1，启动计数器

当上升沿出现在 TI2，计数器计数一次，且 TIF 标志被设置。

在 TI2 的上升沿和计数器实际时钟之间的延时取决于在 TI2 输入端的重新同步电路。

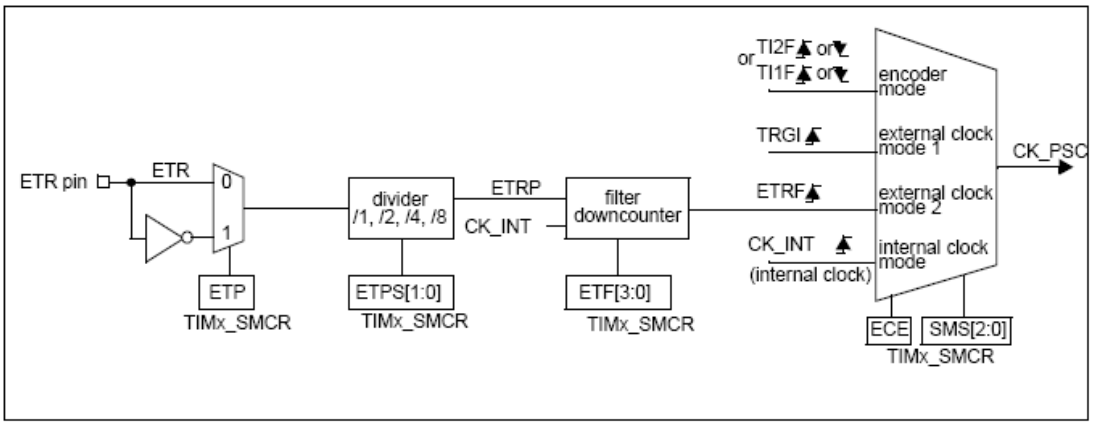
图95 外部时钟模式 1 下的控制电路



外部时钟源模式 2

选定此模式的方法为：令 TIMx_SMCR 寄存器中的 ECE=1
计数器能够在外部触发 ETR 的每一个上升沿或下降沿计数。
图 96 是外部触发输入的总体框图

图96 外部触发输入框图



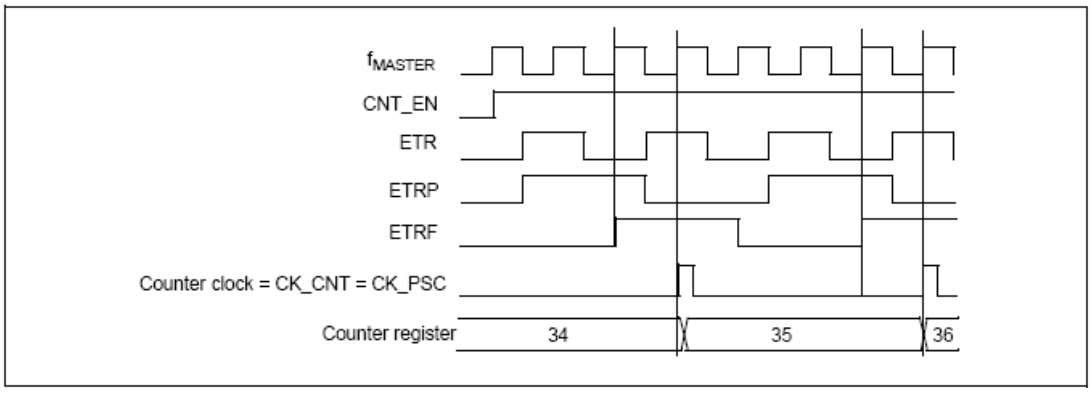
例如，要配置在 ETR 下每 2 个上升沿计数一次的向上计数器，使用下列步骤：

1. 本例中不需要滤波器，写TIMx_SMCR寄存器中的ETF[3:0]=0000
2. 设置预分频器，写TIMx_SMCR寄存器中的ETPS[1:0]=01
3. 设置在ETR下的上升沿检测，写TIMx_SMCR寄存器中的ETP=0
4. 开启外部时钟模式2，写TIMx_SMCR寄存器中的ECE=1
5. 启动计数器，写TIMx_CR1寄存器中的CEN=1

计数器在每 2 个 ETR 上升沿计数一次。

在 ETR 的上升沿和计数器实际时钟之间的延时取决于在 ETRP 信号端的重新同步电路。

图97 外部时钟模式 2 下的控制电路



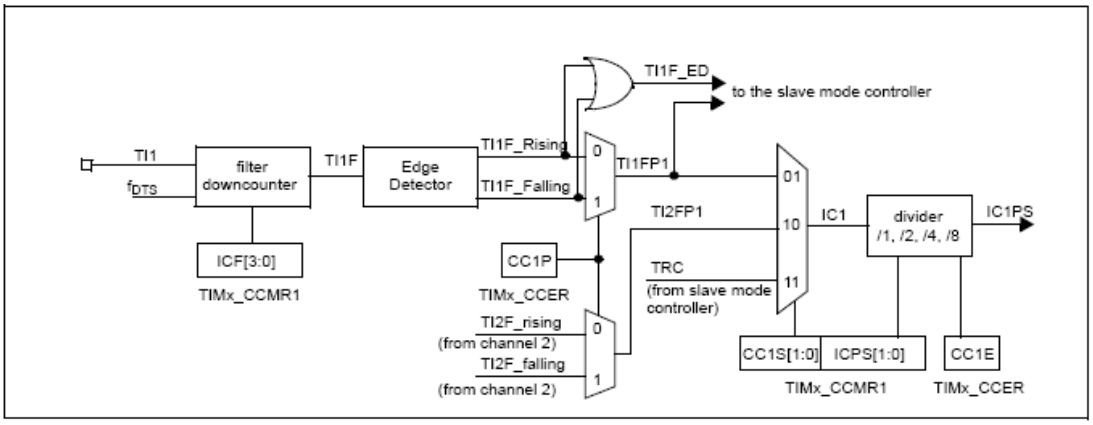
13.4.4 捕获/比较通道

每一个捕获/比较通道是围绕着一个捕获/比较寄存器(包含影子寄存器)，包括捕获的输入部分(包含数字滤波、多路复用和预分频器)，和输出部分(包含比较器和输出控制)。

下图是一个捕获/比较通道概览。

输入部分对相应的 TIx 输入信号采样，并产生一个滤波后的信号 TIxF。然后，一个带极性选择的边缘监测器产生一个信号(TIxFPx)，它可以作为从模式控制器的输入触发或者作为捕获控制。该信号在捕获寄存器(ICxPS)之前已被整形。

图98 捕获/比较通道(如：通道 1 输入部分)



输出部分产生一个中间波形 OCxRef(高有效)作为基准，链的末端信号决定最终的输出极性。

图99 捕获/比较通道 1 的主电路

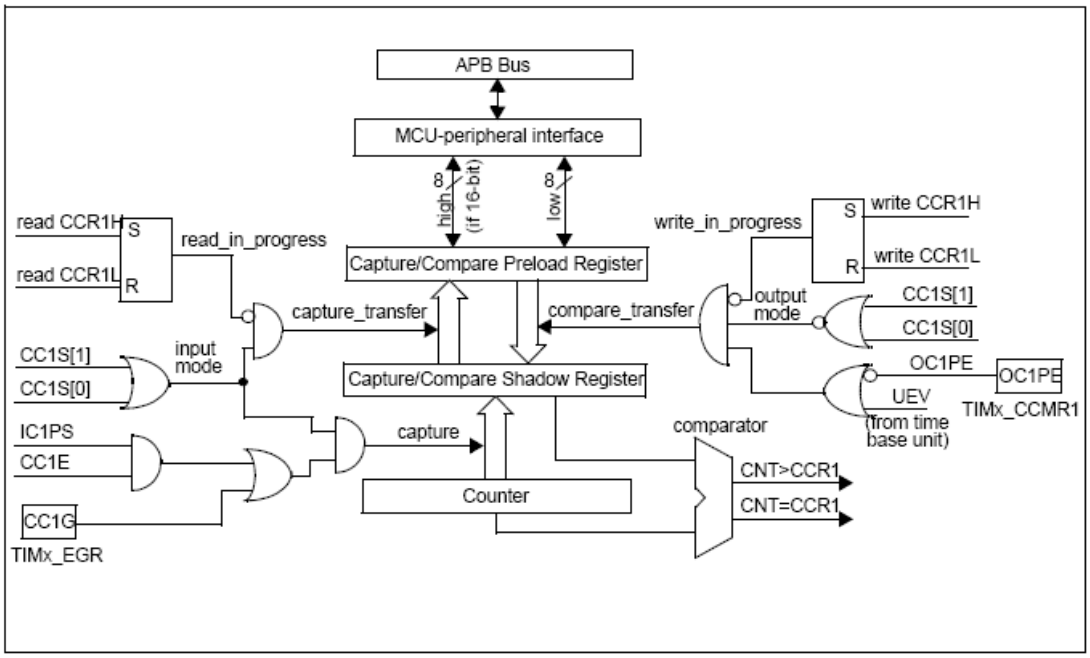
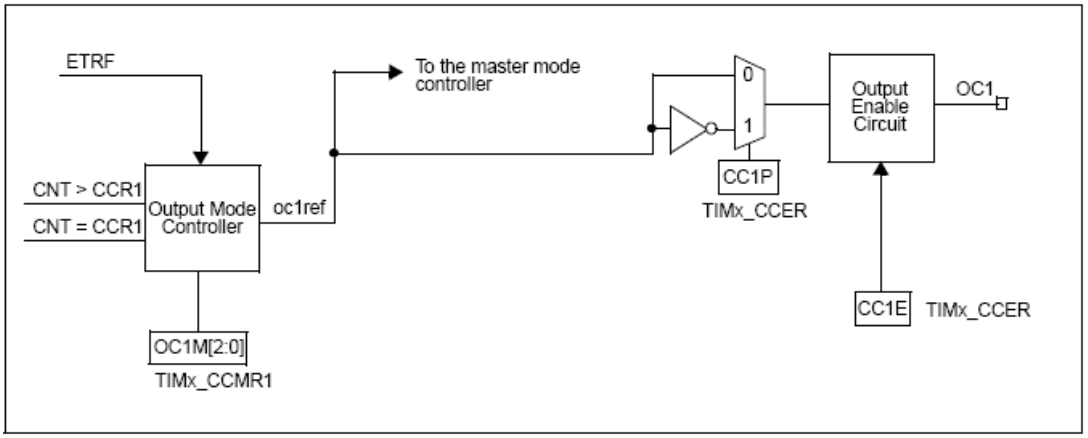


图100 捕获/比较通道的输出部分(通道 1)



捕获/比较模块由一个预装载寄存器和一个影子寄存器组成。读写过程仅操作预装载寄存器。

在捕获模式下，捕获发生在影子寄存器上，然后再复制到预装载寄存器中。

在比较模式下，预装载寄存器的内容被复制到影子寄存器中，然后和计数器进行比较。

13.4.5 输入捕获模式

在输入捕获模式下，当检测到 ICx 信号上相应的边沿后，捕获/比较寄存器(TIMx_CCRx)被用来锁存计数器的值。当一个捕获事件发生时，相应的 CCXIF 标志(TIMx_SR 寄存器)被置 1，如果开放了中断或者 DMA 操作，则将产生中断或者 DMA 操作。如果一个捕获事件发生时 CCxIF 标志已经为高，那么重复捕获标志 CCxOF(TIMx_SR 寄存器)被置 1。写 CCxIF=0 可清除 CCxIF，或读取存储在

TIMx_CCRx 寄存器中的捕获数据也可清除 CCxIF。写 CCxOF=0 可清除 CCxOF。

以下例子说明如何在 TI1 输入的上升沿时捕获计数器的值到 TIMx_CCR1 寄存器中，步骤如下：

- 选择有效输入端：TIMx_CCR1 必须连接到 TI1 输入，所以写入 TIMx_CCR1 寄存器中的 CC1S=01，一旦 CC1S 不为 00 时，通道被配置为输入，并且 TM1_CCR1 寄存器变为只读。
- 根据输入信号的特点，配置输入滤波器为所需的带宽(输入为 TIx 时，TIMx_CCMRx 寄存器中的 ICxF 位)。假设输入信号在最多 5 个时钟周期的时间内抖动，我们须配置滤波器的带宽长于 5 个时钟周期。因此我们可以(以 fDTS 频率)连续采样 8 次，以确认在 TI1 上一次真实的边沿变换，即在 TIMx_CCMR1 寄存器中写入 IC1F=0011。
- 选择 TI1 通道的有效转换边沿，在 TIMx_CCER 寄存器中写入 CC1P=0(即上升沿)。
- 配置输入预分频器。在本例中，我们希望捕获发生在每一个有效的电平转换时刻，因此预分频器被禁止(写 TIMx_CCMR1 寄存器的 IC1PS=00)。
- 设置 TIMx_CCER 寄存器的 CC1E=1，允许捕获计数器的值到捕获寄存器中。
- 如果需要，通过设置 TIMx_DIER 寄存器中的 CC1IE 位允许相关中断请求，通过设置 TIMx_DIER 寄存器中的 CC1DE 位允许 DMA 请求。
- 发生当一个输入捕获时：
- 当产生有效的电平转换时，计数器的值被传送到 TIMx_CCR1 寄存器。
- CC1IF 标志被设置(中断标志)。当发生至少 2 个连续的捕获时，而 CC1IF 未曾被清除，CC1OF 也被置 1。
- 如设置了 CC1IE 位，则会产生一个中断。
- 如设置了 CC1DE 位，则还会产生一个 DMA 请求。

为了处理捕获溢出，建议在读出捕获溢出标志之前读取数据，这是为了避免丢失在读出捕获溢出标志之后和读取数据之前可能产生的捕获溢出信息。

注： *设置TIMx_EGR寄存器中相应的CCxG位，可以通过软件产生输入捕获中断和/或DMA请求。*

13.4.6 PWM输入模式

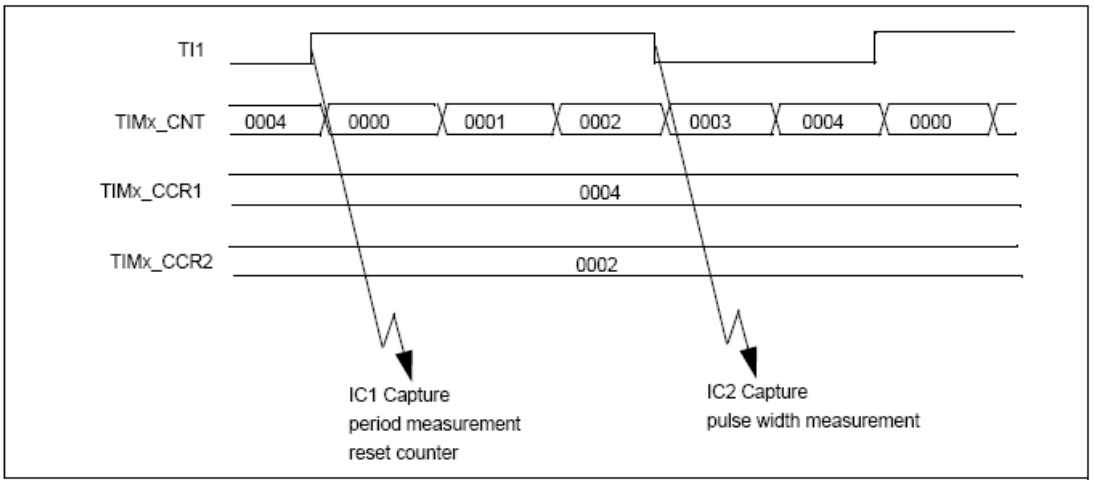
该模式是输入捕获模式的一个特例，除下列区别外，工作过程与输入捕获模式相同：

- 两个 ICx 信号被映射同一个 TIx 输入。
- 这 2 个 ICx 信号为边沿有效，但是极性相反。
- 其中一个 TIxFP 信号被作为触发输入信号，并且从模式控制器被配置成复位模式。

例如，你能够测量输入到 TI1 上的 PWM 信号的长度(TIMx_CCR1 寄存器)和占空比(TIMx_CCR2 寄存器)，具体步骤如下(取决于 CK_INT 的频率和预分频器的值)

- 选择 TIMx_CCR1 的有效输入端：置 TIMx_CCMR1 寄存器的 CC1S=01(选择 TI1)；
- 选择 TI1FP1 的有效极性(用来捕获数据到 TIMx_CCR1 中和清除计数器)：置 CC1P=0；
- 选择 TIMx_CCR2 的有效输入端：置 TIMx_CCMR1 寄存器的 CC2S=10(选择 TI1)；
- 选择 TI1FP2 的有效极性(用来捕获数据到 TIMx_CCR2)：置 CC2P=1(下降沿有效)；
- 选择有效的触发输入信号：置 TIMx_SMCR 寄存器中的 TS=101(选择 TI1FP1)；
- 配置从模式控制器为复位模式：置 TIMx_SMCR 中的 SMS=100；
- 使能捕获：置 TIMx_CCER 寄存器中 CC1E=1 且 CC2E=1。

图101 PWM 输入模式时序



13.4.7 强置输出模式

在输出模式(TIMx_CCMRx 寄存器中 CCxS=00)下，输出比较信号(OCxREF 和相应的 OCx/OCxN)能够直接由软件强置为有效或无效状态，而不依赖于输出比较寄存器和计数器间的比较结果。

置 TIMx_CCMRx 寄存器中相应的 OCxM=101，即可强置输出比较信号(OCxREF/OCx)为有效状态。这样 OCxREF 被强置为高电平(OCxREF 始终为高电平有效)，同时 OCx 得到 CCxP 极性位相反的值。

例如：CCxP=0(OCx 高电平有效)，则 OCx 被强置为高电平。

置 TIMx_CCMRx 寄存器中的 OCxM=100，可强置 OCxREF 信号为低。
该模式下，在 TIMx_CCRx 影子寄存器和计数器之间的比较仍然在进行，相应的标志也会被修改。因此仍然会产生相应的中断和 DMA 请求。这将会在下面的输出比较模式一节中介绍。

13.4.8 输出比较模式

此项功能是用来控制一个输出波形或者指示何时一段给定的的时间已经到时。

当计数器与捕获/比较寄存器的内容相同时，输出比较功能做如下操作：

- 将输出比较模式(TIMx_CCMRx 寄存器中的 OCxM 位)和输出极性(TIMx_CCER 寄存器中的 CCxP 位)定义的值输出到对应的管脚上。在比较匹配时，输出管脚可以保持它的电平(OCxM=011)、被设置成有效电平(OCxM=001)、被设置成无有效电平(OCxM=010)或进行翻转(OCxM=011)。
- 设置中断状态寄存器中的标志位(TIMx_SR 寄存器中的 CCxIF 位)。
- 如果设置了相应的中断屏蔽(TIMx_DIER 寄存器中的 CCXIE 位)，则产生一个中断。
- 若设置了相应的使能位(TIMx_DIER 寄存器中的 CCxDE 位，TIMx_CR2 寄存器中的 CCDS 位选择 DMA 请求功能)，则产生一个 DMA 请求。

TIMx_CCMRx 中的 OCxPE 位用于选择 TIMx_CCRx 寄存器是否需要使用预装载寄存器。

在输出比较模式下，更新事件 UEV 对 OCxREF 和 OCx 输出没有影响。

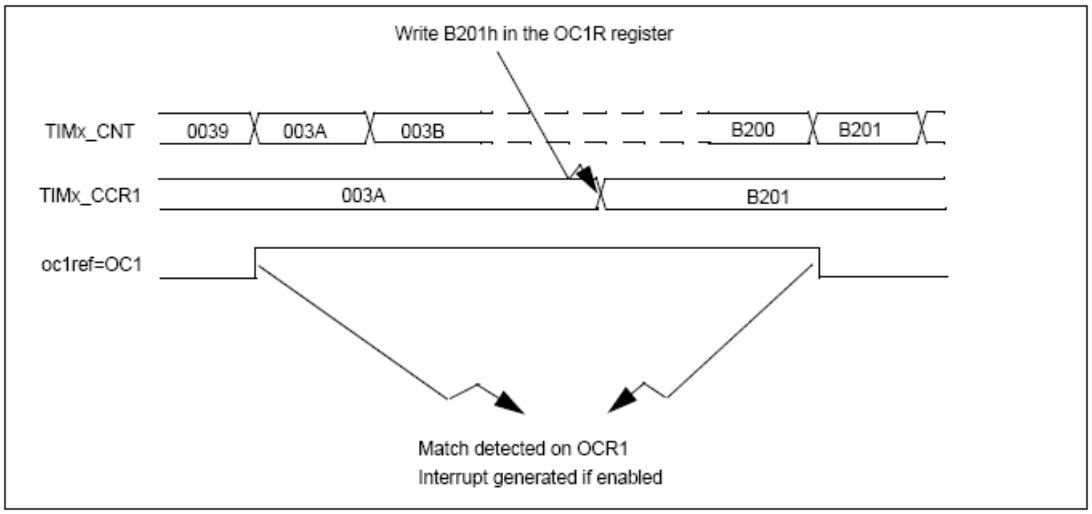
同步的精度到达计数器的一个计数周期。输出比较模式(在单脉冲模式下)也能用来输出一个单脉冲。

输出比较模式的配置步骤：

1. 选择计数器时钟(内部，外部，预分频器)
2. 将相应的数据写入TIMx_ARR和TIMx_CCRx寄存器中
3. 如果要产生一个中断请求和/或一个DMA请求，设置CCxIE位和/或CCxDE位。
4. 选择输出模式，例如：必须设置OCxM=’011’、OCxPE=’0’、CCxP=’0’和CCxE=’1’，当CNT与CCRx匹配时翻转OCx的输出管脚，CCRx预装载未用，开启OCx输出且高电平有效。
5. 设置TIMx_CR1寄存器的CEN位启动计数器

TIMx_CCRx寄存器能够在任何时候通过软件进行更新以控制输出波形，条件是未使用预装载寄存器(OCxPE=’0’，否则TIMx_CCRx影子寄存器只能在下一次更新事件发生时被更新)。图 102 给出了一个例子。

图102 输出比较模式，翻转 OC1



13.4.9 PWM 模式

脉冲宽度调制模式可以产生一个由 TIMx_ARR 寄存器确定频率、由 TIMx_CCRx 寄存器确定占空比的信号。

在 TIMx_CCMRx 寄存器中的 OCxM 位写入“110”(PWM 模式 1)或“111”(PWM 模式 2)，能够独立地设置每个通道工作在 PWM 模式，每个 OCx 输出一路 PWM。必须通过设置 TIMx_CCMRx 寄存器 OCxPE 位使能相应的预装载寄存器，最后还要设置 TIMx_CR1 寄存器的 ARPE 位使能自动重装载的预装载寄存器(在向上计数或中心对称模式中)。

因为仅当发生一个更新事件的时候，预装载寄存器才能被传送到影子寄存器，因此在计数器开始计数之前，必须通过设置 TIMx_EGR 寄存器中的 UG 位来初始化所有的寄存器。

OCx 的极性可以通过软件在 TIMx_CCER 寄存器中的 CCxP 位设置，它可以设置为高电平有效活和低电平有效。OCx 输出通过 TIMx_CCER 寄存器中的 CCxE 位使能。详见 TIMx_CCERx 寄存器的描述。

在 PWM 模式(模式 1 或模式 2)下，TIMx_CNT 和 TIM1_CCRx 始终在进行比较，(依据计数器的计数方向)以确定是否符合 TIM1_CCRx≤TIM1_CNT 或者 TIM1_CNT≤TIM1_CCRx。然而，为了与 OCREF_CLR 的功能(在下一个 PWM 周期之前，OCREF 能够通过 ETR 信号被一个外部事件清除)一致，OCREF 信号只能在下述条件下产生：

- 当比较的结果改变，或
- 当输出比较模式(TIMx_CCMRx 寄存器中的 OCxM 位)从“冻结”(无比较，OCxM=’000’)切换到某个 PWM 模式(OCxM=’110’或’111’)。

这样在运行中可以通过软件强置 PWM 输出。
根据 TIMx_CR1 寄存器中 CMS 位的状态，定时器能够产生边沿对齐的或中央对齐的 PWM 信号。

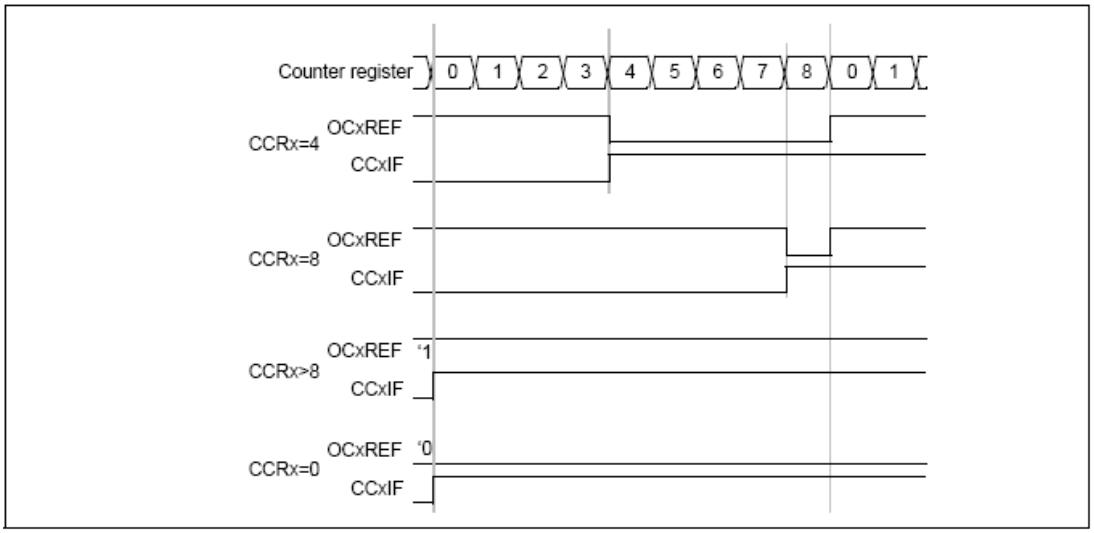
PWM 边沿对齐模式

向上计数配置

当TIMx_CR1 寄存器中的DIR位为低的时候执行向上计数。参看 13.4.2 节。

下面是一个PWM模式 1 的例子。当TIMx_CNT<TIMx_CCRx时PWM信号OCxREF为高，否则为低。如果TIMx_CCRx中的比较值大于自动重装载值(TIMx_ARR)，则OCxREF保持为“1”。如果比较值为 0，则OCxREF保持为“0”。 图 103 为TIMx_ARR=8 时边沿对齐的PWM波形实例。

图103 边沿对齐的 PWM 波形(ARR=8)



向下计数的配置

当TIMx_CR1 寄存器的DIR位为高时执行向下计数。参看 13.4.2 节。

在 PWM 模式 1，当 TIMx_CNT>TIMx_CCRx 时 OCxREF 为低，否则为高。如果 TIMx_CCRx 中的比较值大于 TIMx_ARR 中的自动重装载值，则 OCxREF 保持为“1”。该模式下不能产生 0%的 PWM 波形。

PWM 中央对齐模式

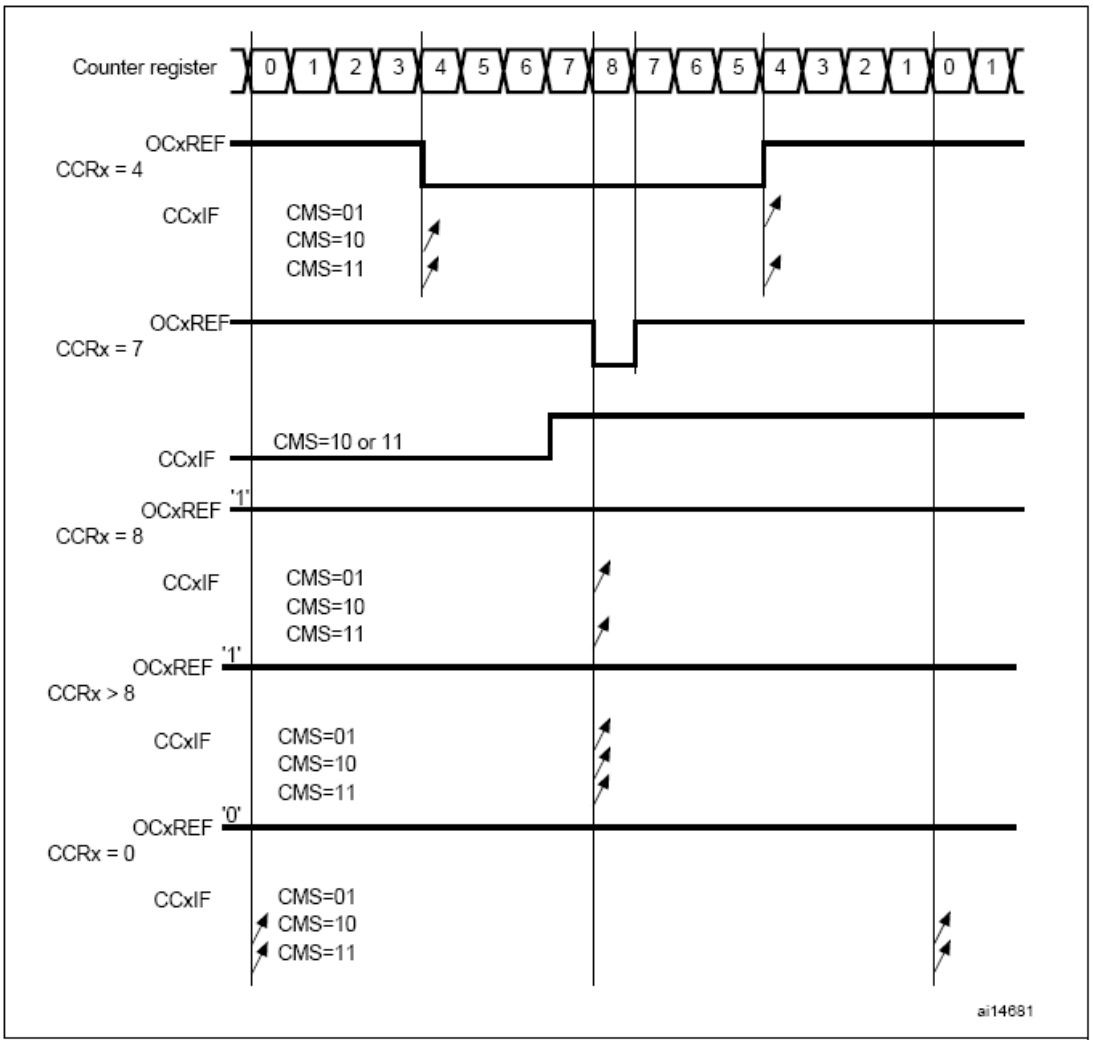
当TIMx_CR1 寄存器中的CMS位不为 00 时为中央对齐模式(所有其他的配置对OCxREF/OCx信号都有相同的作用)。根据不同的CMS位的设置，比较标志可能在计数器向上计数时被置 1、在计数器向下计数时被置 1、或在计数器向上和向下计数时被置 1。TIMx_CR1 寄存器中的计数方向位(DIR)由硬件更新，不要用软件修改它。参看 12.4.2 节的中央对齐模式。

图 104 给出了一些中央对齐的PWM波形的例子

- TIMx_ARR=8
- PWM 模式 1

- TIMx_CR1 寄存器中的 CMS=01，在中央对齐模式 1 时，当计数器向下计数时标志被设置

图104 中央对齐的 PWM 波形(APR=8)



使用中央对齐模式的提示

- 进入中央对齐模式时，当前的上-下配置被使用；这就意味着计数器向上还是向下计数取决于 TIMx_CR1 寄存器中 DIR 位的当前值。此外，DIR 和 CMS 位不能同时被软件修改。
- 不推荐当运行在中央对齐模式时改写计数器，因为会产生不可预知的结果。特别地：
  - 如果写入计数器的值大于自动重加载的值(TIMx_CNT>TIMx_ARR)，则方向不会被更新。例如，如果计数器正在向上计数，它就会继续向上计数。
  - 如果将 0 或者 TIMx_ARR 的值写入计数器，方向被更新，但不产生更新事件 UEV。

使用中央对齐模式最保险的方法，就是在启动计数器之前产生一个软件更新(设置 TIMx_EGR 位中的 UG 位)，不要在计数进行过程中修改计数器的值。

13.4.10 单脉冲模式

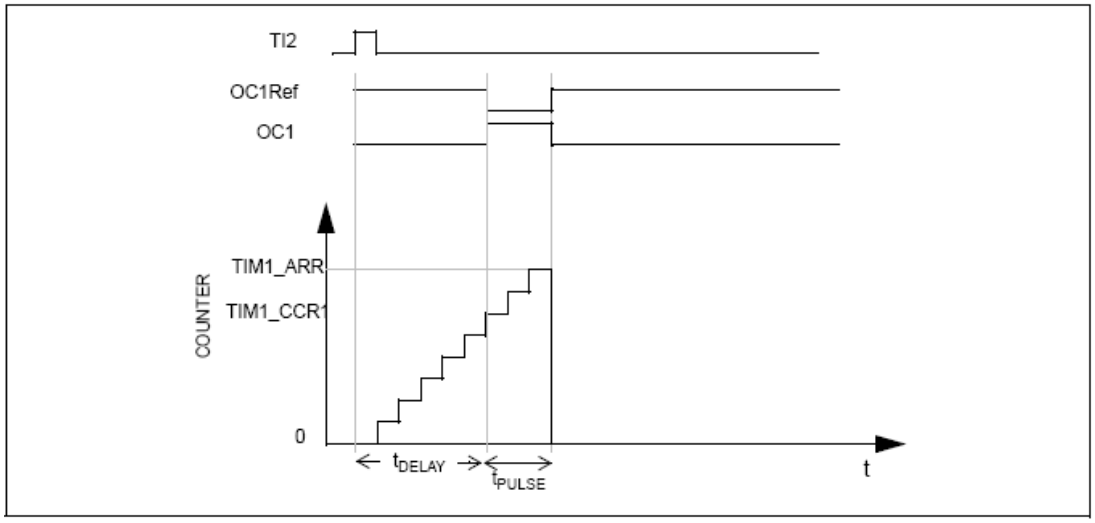
单脉冲模式(OPM)时前述众多模式的一个特例。这种模式允许计数器响应一个激励，并在一个可编程延时之后产生一个可对脉宽可编程的脉冲。

可以通过从模式控制器启动计数器，在输出比较模式或者 PWM 模式下产生波形。设置 TIMx_CR1 寄存器中的 OPM 位将选择单脉冲模式，这样可以让计数器自动地在产生下一个更新事件 UEV 时停止。

仅当比较值与计数器的初始值不同时，才能产生一个脉冲。启动之前(当定时器正在等待触发)，必须如下配置：

向上计数方式：CNT<CCRx≤ARR (特别地, 0<CCRx)，
向下计数方式：CNT>CCRx。

图105 单脉冲模式的例子



例如，你需要在从 TI2 输入脚上检测到一个上升沿开始，延迟 t_{DELAY} 之后，在 OC1 上产生一个长度为 t_{PULSE} 的正脉冲。

假定 TI2FP2 作为触发 1:

- 置 TIMx_CCMR1 寄存器中的 IC2S=01，把 TI2FP2 映像到 TI2。
- 置 TIMx_CCER 寄存器中的 CC2P=0，使 TI2FP2 能够检测上升沿。
- 置 TIMx_SMCR 寄存器中的 TS=110，配置 TI2FP2 作为从模式控制器的触发(TRGI)。
- 置 TIMx_SMCR 寄存器中的 SMS=110(触发模式)，TI2FP2 被用来启动计数器。
- OPM 波形由写入比较寄存器决定(要考虑时钟频率和计数器预分频器)
- t_{DELAY} 由写入 TIMx_CCR1 寄存器中的值定义。
- t_{PULSE} 由自动装载值和比较值之间的差值定义(TIMx_ARR - TIMx_CCR1)。
- 假定当发生比较匹配时要产生从0到1的波形，当计数器到达预装载值是要产生一个从1到0的波形；首先要置TIMx_CCMR1寄存器中的OC1M=111，

进入PWM模式2；有选择的置TIMx_CCMR1中的OC1PE=1和TIMx_CR1寄存器中的ARPE，使能预装载寄存器；然后在TIMx_CCR1寄存器中填写比较值，在TIMx_ARR寄存器中填写自动装载值，修改UG位来产生一个更新事件，然后等待在TI2上的一个外部触发事件。本例中，CC1P=0。

在这个例子中，TIMx_CR1 寄存器中的 DIR 和 CMS 位应该置低。

因为只需一个脉冲，所以须在下一个更新事件(当计数器从自动装载值翻转到 0)时设置 TIMx_CR1 寄存器中的 OPM=1，以停止计数。

特殊情况：OCx快速使能

在单脉冲模式下，在 TIx 输入脚的边沿检测逻辑设置 CEN 位以启动计数器。然后计数器和比较值间的比较操作产生了输出的转换。但是这些操作需要一定的时钟周期，因此它限制了可得到的最小延时 tDELAY。

如果要以最小延时输出波形，可以设置 TIMx_CCMRx 寄存器中的 OCxFE 位；此时 OCxREF(和 OCx)被强制响应激励而不再依赖比较的结果，输出的波形与比较匹配时的波形一样。OCxFE 只在通道配置为 PWM1 和 PWM2 模式时起作用。

13.4.11 在外部事件时清除OCxREF信号

对于一个给定的通道，在 ETRF 输入端(TIMx_CCMRx 寄存器中对应的 OCxCE 允许位置“1”)的高电平能够把 OCxREF 信号拉低，OCxREF 信号将保持为低直到发生下一次的更新事件 UEV。

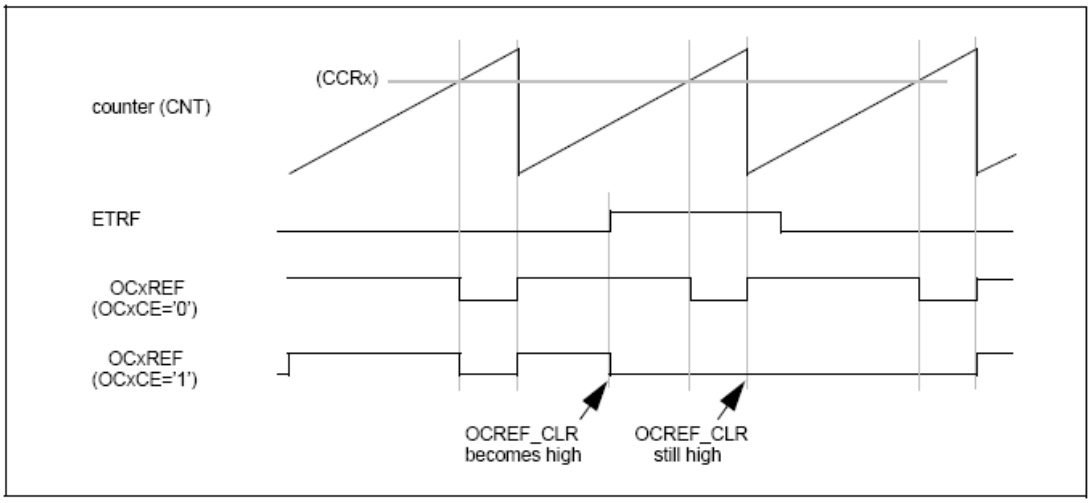
该功能只能用于输出比较和 PWM 模式，而不能用于强置模式。

例如，OCxREF 信号可以联到一个比较器的输出，用于处理电流。这时，ETR 必须配置如下：

1. 外部触发预分频器必须处于关闭：TIMx_SMCR 寄存器中的 ETPS[1:0]=00。
2. 必须禁止外部时钟模式2：TIMx_SMCR寄存器中的ECE=0。
3. 外部触发极性(ETP)和外部触发滤波器(ETF)可以根据需要配置。

图 63 显示了当ETRF输入变为高时，对应不同OCxCE的值，OCxREF信号的动作。在这个例子中，定时器TIMx被置于PWM模式。

图106 清除 TIMx 的 OCxREF



13.4.12 编码器接口模式

选择编码器接口模式的方法是：如果计数器只在 TI2 的边沿计数，则置 TIMx_SMCR 寄存器中的 SMS=001；如果只在 TI1 边沿计数，则置 SMS=010；如果计数器同时在 TI1 和 TI2 边沿计数，则置 SMS=011。

通过设置 TIMx_CCER 寄存器中的 CC1P 和 CC2P 位，可以选择 TI1 和 TI2 极性；如果需要，还可以对输入滤波器编程。

两个输入TI1 和TI2 被用来作为增量编码器的接口。参看 表 39，假定计数器已经启动(TIMx_CR1 寄存器中的CEN=1)，则计数器由每次在TI1FP1 或TI2FP2 上的有效跳变驱动。TI1FP1 和TI2FP2 是TI1 和TI2 在通过输入滤波器和极性控制后的信号；如果没有滤波和变相，则TI1FP1=TI1；如果没有滤波和变相，则TI2FP2=TI2。根据两个输入信号的跳变顺序，产生了计数脉冲和方向信号。依据两个输入信号的跳变顺序，计数器向上或向下计数，因此TIMx_CR1 寄存器的DIR位由硬件进行相应的设置。不管计数器是依靠TI1 计数、依靠TI2 计数或者同时依靠TI1 和TI2 计数。在任一输入(TI1 或者TI2)跳变时都会重新计算DIR位。

编码器接口模式基本上相当于使用了一个带有方向选择的外部时钟。这意味着计数器只在 0 到 TIMx_ARR 寄存器中自动装载值之间连续计数(根据方向，或是 0 到 ARR 计数，或是 ARR 到 0 计数)。所以在开始计数之前必须配置 TIMx_ARR；同样，捕获器、比较器、预分频器、触发输出特性等仍工作如常。

在这个模式下，计数器依照增量编码器的速度和方向被自动的修改，因此，它的内容始终指示着编码器的位置。计数方向与相连的传感器旋转的方向对应。表 39 列出了所有可能的可能的组合，假设TI1 和TI2 不同时变换。

表39 计数方向与编码器信号的关系

有效边沿	相对信号的电平 (TI1FP1对应TI2, TI2FP2对应TI1)	TI1FP1信号		TI2FP2信号	
		上升	下降	上升	下降
仅在TI1计数	高	向下计数	向上计数	不计数	不计数
	低	向上计数	向下计数	不计数	不计数

仅在TI2计数	高	不计数	不计数	向上计数	向下计数
	低	不计数	不计数	向下计数	向上计数
在TI1和TI2上计数	高	向下计数	向上计数	向上计数	向下计数
	低	向上计数	向下计数	向下计数	向上计数

一个外部的增量编码器直接和 MCU 连接不需要外部接口逻辑。但是，一般使用比较器将编码器的差动输出转换到数字信号，这大大增加了抗噪声干扰能力。编码器输出的第三个信号表示机械零点，可以连接到一个外部中断输入和触发一个计数器复位。

图 107 给出一个计数器操作的实例，显示了计数信号的产生和方向控制。它还显示了当选择了双边沿时，输入抖动是如何被抑制的；这种情况可能会在传感器的位置靠近一个转换点事发生。在这个例子中，我们假定配置如下：

- CC1S=’01’ (TIMx_CCMR1 寄存器，IC1FP1 映射到 TI1)
- CC2S=’01’ (TIMx_CCMR2 寄存器，IC2FP2 映射到 TI2)
- CC1P=’0’ (TIMx_CCER 寄存器，IC1FP1 不反相，IC1FP1=TI1)
- CC2P=’0’ (TIMx_CCER 寄存器，IC2FP2 不反相，IC2FP2=TI2)
- SMS=’011’ (TIMx_SMCR 寄存器，所有的输入均在上升沿和下降沿有效).
- CEN=’1’ (TIMx_CR1 寄存器，计数器使能)

图107 编码器模式下的计数器操作实例

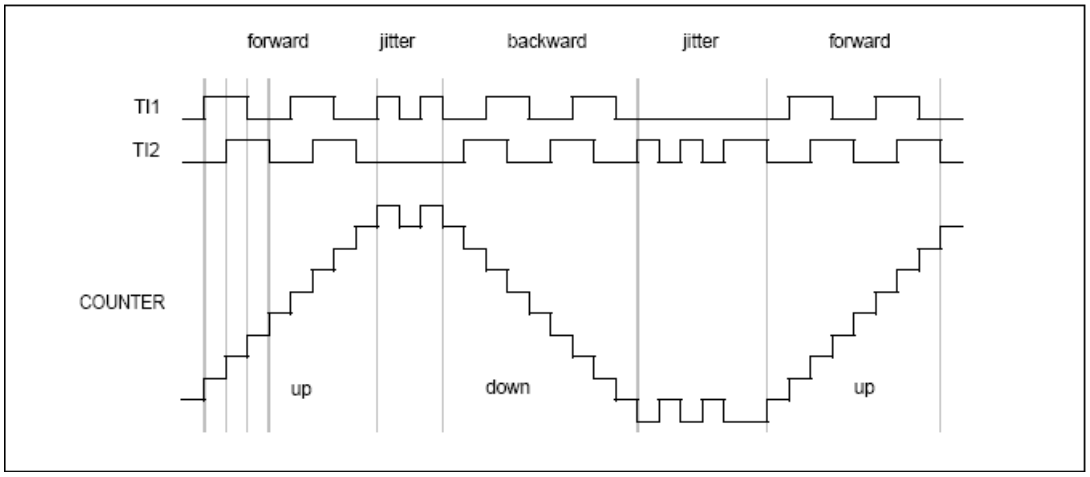
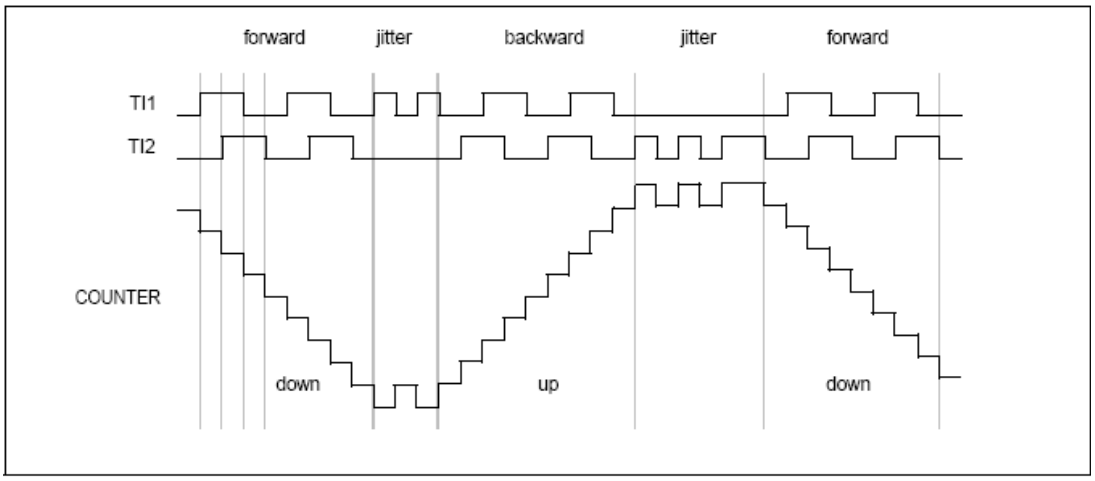


图 108 为当IC1FP1 极性反相时计数器的操作实例(CC1P=’1’，其他配置与上例相同)

图108 IC1FP1 反相的编码器接口模式实例



当定时器配置成编码器接口模式时，提供传感器当前位置的信息。使用第二个配置在捕获模式定时器测量两个编码器事件的间隔，可以获得动态的信息(速度，加速度，减速度)。指示机械零点的编码器输出可被用做此目的。根据两个事件间的间隔，可以按照固定的时间读出计数器。如果可能的话，你可以把计数器的值锁存到第三个输入捕获寄存器(捕获信号必须是周期的并且可以由另一个定时器产生)。它也可以通过一个由实时时钟产生的 DMA 请求来读取它的值。

13.4.13 定时器输入异或功能

TIMx_CR2 寄存器中的 TI1S 位，允许通道 1 的输入滤波器连接到一个异或门的输出端，异或门的 3 个输入端为 TIMx_CH1、TIMx_CH2 和 TIMx_CH3。

异或输出能够用于所有定时器的输入功能，如触发或输入捕获。上一章给出了此特性用于连接霍尔传感器的例子。

13.4.14 定时器和外部触发的同步

TIMx 定时器能够在多种模式下和一个外部的触发同步：复位模式、门控模式和触发模式。

从模式：复位模式

在发生一个触发输入事件时，计数器和它的预分频器能够重新被初始化；同时，如果 TIMx_CR1 寄存器的 URS 位为低，还产生一个更新事件 UEV；然后所有的预装载寄存器(TIMx_ARR， TIMx_CCRx)都被更新了。

在以下的例子中，TI1 输入端的上升沿导致向上计数器被清零：

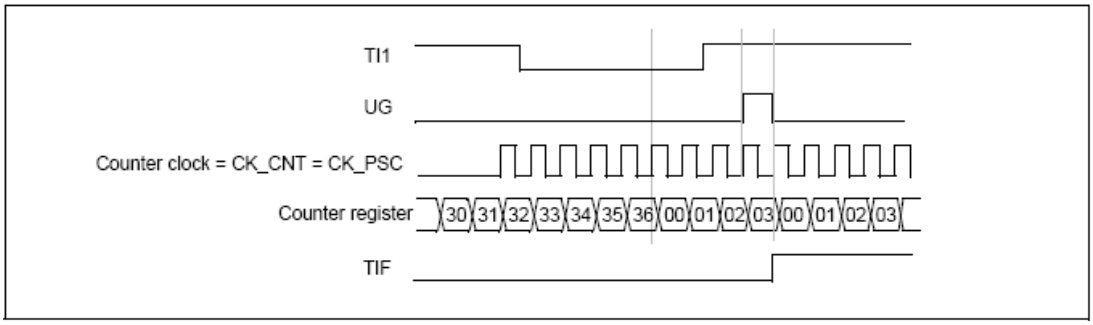
- 配置通道 1 以检测 TI1 的上升沿。配置输入滤波器的带宽(在本例中，不需要任何滤波器，因此保持 IC1F=0000)。触发操作中不使用捕获预分频器，所以不需要配置。CC1S 位只选择输入捕获源，即 TIMx_CCMR1 寄存器中 CC1S=01。置 TIMx_CCER 寄存器中 CC1P=0 以确定极性(只检测上升沿)。

- 置 TIMx_SMCR 寄存器中 SMS=100，配置定时器为复位模式；置 TIMx_SMCR 寄存器中 TS=101，选择 TI1 作为输入源。
- 置 TIMx_CR1 寄存器中 CEN=1，启动计数器。

计数器开始依据内部时钟计数，然后正常运转直到 TI1 出现一个上升沿；此时，计数器被清零然后从 0 重新开始计数。同时，触发标志(TIMx_SR 寄存器中的 TIF 位)被设置，根据 TIMx_DIER 寄存器中 TIE(中断使能)位和 TDE(DMA 使能)位的设置，产生一个中断请求或一个 DMA 请求。

下图显示当自动重装载寄存器 TIMx_ARR=0x36 时的动作。在 TI1 上升沿和计数器的实际复位之间的延时取决于 TI1 输入端的重同步电路。

图109 复位模式下的控制电路



从模式：门控模式

计数器的使能依赖于选中的输入端的电平。

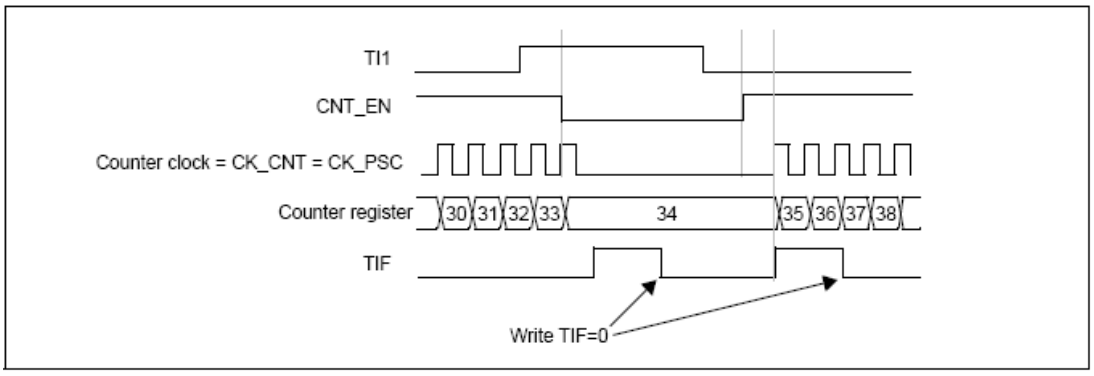
在如下的例子中，计数器只在 TI1 为低时向上计数：

- 配置通道 1 以检测 TI1 上的低电平。配置输入滤波器带宽(本例中，不需要滤波，所以保持 IC1F=0000)。触发操作中不使用捕获预分频器，所以不需要配置。CC1S 位用于选择输入捕获源，置 TIMx_CCMR1 寄存器中 CC1S=01。置 TIMx_CCER 寄存器中 CC1P=1 以确定极性(只检测低电平)。
- 置 TIMx_SMCR 寄存器中 SMS=101，配置定时器为门控模式；置 TIMx_SMCR 寄存器中 TS=101，选择 TI1 作为输入源。
- 置 TIMx_CR1 寄存器中 CEN=1，启动计数器。在门控模式下，如果 CEN=0，则计数器不能启动，不论触发输入电平如何。

只要 TI1 为低，计数器开始依据内部时钟计数，在 TI1 变高时停止计数。当计数器开始或停止时都设置 TIMx_SR 中的 TIF 标置。

TI1 上升沿和计数器实际停止之间的延时取决于 TI1 输入端的重同步电路。

图110 门控模式下的控制电路



从模式：触发模式

计数器的使能依赖于选中的输入端上的事件。

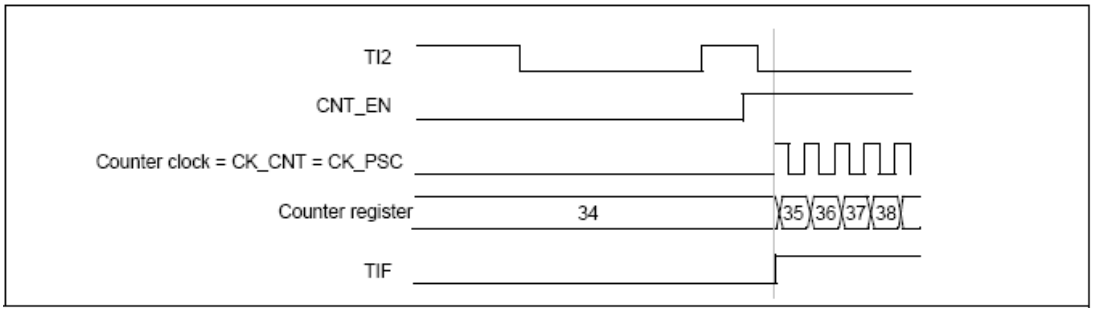
在下面的例子中，计数器在 TI2 输入的上升沿开始向上计数：

- 配置通道 2 检测 TI2 的上升沿。配置输入滤波器带宽(本例中，不需要任何滤波器，保持 IC2=0000)。触发操作中不使用捕获预分频器，不需要配置。CC2 位只用于选择输入捕获源，置 TIMx_CCMR1 寄存器中 CC2S=01。置 TIMx_CCER 寄存器中 CC1P=1 以确定极性(只检测低电平)。
- 置 TIMx_SMCR 寄存器中 SMS=110，配置定时器为触发模式；置 TIMx_SMCR 寄存器中 TS=110，选择 TI2 作为输入源。

当 TI2 上出现一个上升沿时，计数器开始在内部时钟驱动下计数，同时 TIF 标志被设置。

TI2 上升沿和计数器启动计数之间的延时取决于 TI2 输入端的重同步电路。

图111 触发器模式下的控制电路



从模式：外部时钟模式 2 + 触发模式

外部时钟模式 2 可以与另一种从模式(外部时钟模式 1 和编码器模式除外)一起使用。这时，ETR 信号被用作外部时钟的输入，在复位模式下，门控模式或触发模式时可以选择另一个输入作为触发输入。不建议在 TIMx_SMCR 寄存器的 TS 位中选择 ETR 作为 TRGI。

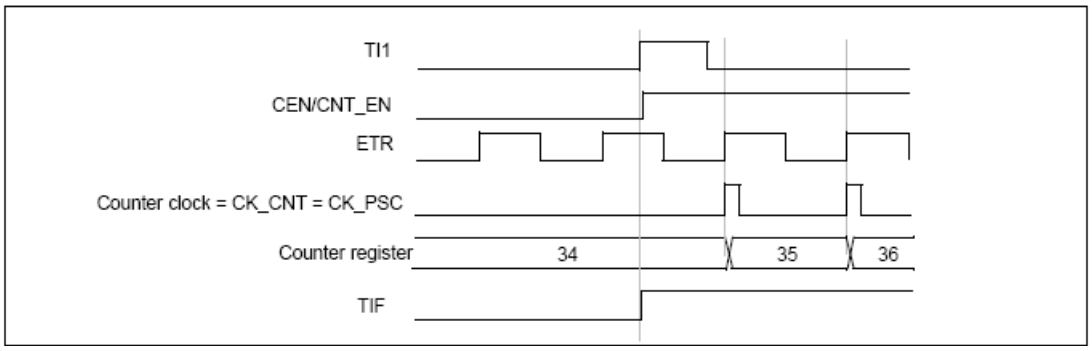
在下面的例子中，一旦在 TI1 上出现一个上升沿，计数器即在 ETR 的每一个上升沿向上计数：

1. 通过TIMx_SMCR寄存器配置外部触发输入电路：
   - ETF=0000：没有滤波
   - ETPS=00：不用预分频器
   - ETP=0：检测ETR的上升沿，置ECE=1使能外部时钟模式2
2. 按如下配置通道1，检测TI的上升沿：
   - IC1F=0000：没有滤波
   - 触发操作中不使用捕获预分频器，不需要配置
   - 置TIMx_CCMR1寄存器中CC1S=01，选择输入捕获源
   - 置TIMx_CCER寄存器中CC1P=0以确定极性(只检测上升沿)
3. 置TIMx_SMCR寄存器中SMS=110，配置定时器为触发模式。置TIMx_SMCR寄存器中TS=101，选择TI1作为输入源。

当 TI1 上出现一个上升沿时，TIF 标志被设置，计数器开始在 ETR 的上升沿计数。

ETR 信号的上升沿和计数器实际复位间的延时取决于 ETRP 输入端的重同步电路。

图112 外部时钟模式 2＋触发模式下的控制电路



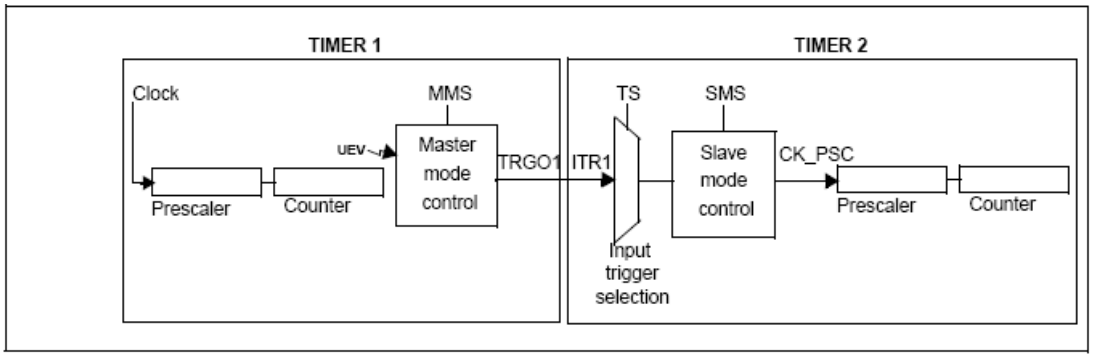
13.4.15 定时器同步

所有 TIMx 定时器在内部相连，用于定时器同步或链接。当一个定时器处于主模式时，它可以对另一个处于从模式的定时器的计数器进行复位、启动、停止或提供时钟等操作。

下图显示了触发选择和主模式选择模块的概况。

使用一个定时器作为另一个的预分频器

图113 主/从定时器的例子



如：可以配置定时器 1 作为定时器 2 的预分频器。参考 图 113，进行下述操作：

- 配置定时器 1 为主模式，它可以在每一个更新事件 UEV 时输出一个周期的触发信号。在 TIM1_CR2 寄存器的 MMS=’010’时，每次产生一个更新事件时在 TRGO1 上产生一个上升沿信号。
- 连接定时器 1 的 TRGO1 输出至定时器 2，定时器 2 必须配置为使用 ITR1 作为内部触发的从模式。设置 TIM2_SMCR 寄存器的 TS=’001’。
- 然后把从模式控制器置于外部时钟模式 1(TIM2_SMCR 寄存器的 SMS=111)；这样定时器 2 即可由定时器 1 周期的上升沿(即定时器 1 的计数器溢出)信号驱动。
- 最后，必须设置相应(TIMx_CR1 寄存器)的 CEN 位分别启动两个定时器。

注： 如果 OCx 已被选中为定时器 1 的触发输出(MMS=1xx)，它的上升沿用于驱动定时器 2 的计数器。

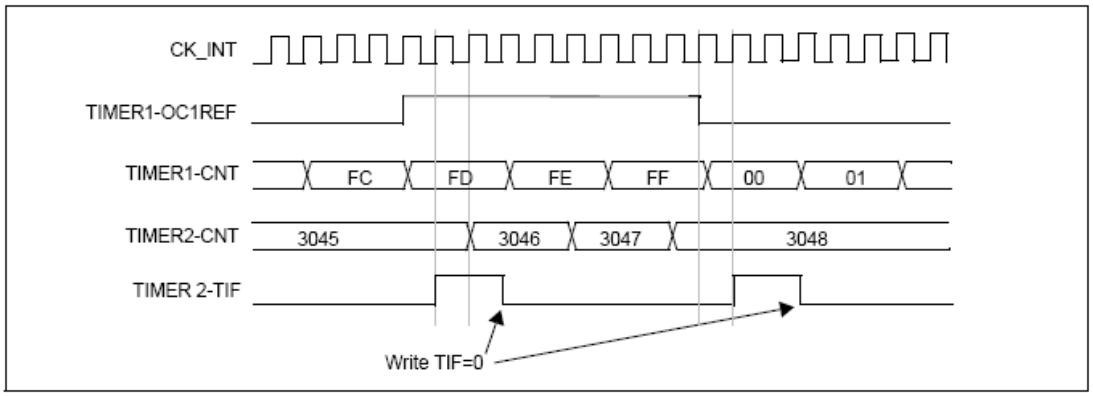
使用一个定时器去使能另一个定时器

在这个例子中，定时器 2 的由定时器 1 的输出比较启动。参考 图 113 的连接。定时器 2 只当定时器 1 的OC1REF为高时才对分频的内部时钟计数。两个定时器的时钟频率都是由预分频器对CK_INT除以 3($f_{CK\_CNT}=f_{CK\_INT}/3$)。

- 配置定时器 1 为主模式，送出它的输出比较参考信号(OC1REF)为触发输出(TIM1_CR2 寄存器的 MMS=100)
- 配置定时器 1 的 OC1REF 波形(TIM1_CCMR1 寄存器)
- 配置定时器 2 从定时器 1 获得输入触发(TIM2_SMCR 寄存器的 TS=001)
- 配置定时器 2 为门控模式(TIM2_SMCR 寄存器的 SMS=101)
- 置 TIM2_CR1 寄存器的 CEN=1 以使能定时器 2
- 置 TIM1_CR1 寄存器的 CEN=1 以启动定时器 1

注： 定时器 2 的时钟不与定时器 1 的时钟同步，这个模式只影响定时器 2 计数器的启动信号。

图114 定时器 1 的 OC1REF 控制定时器 2

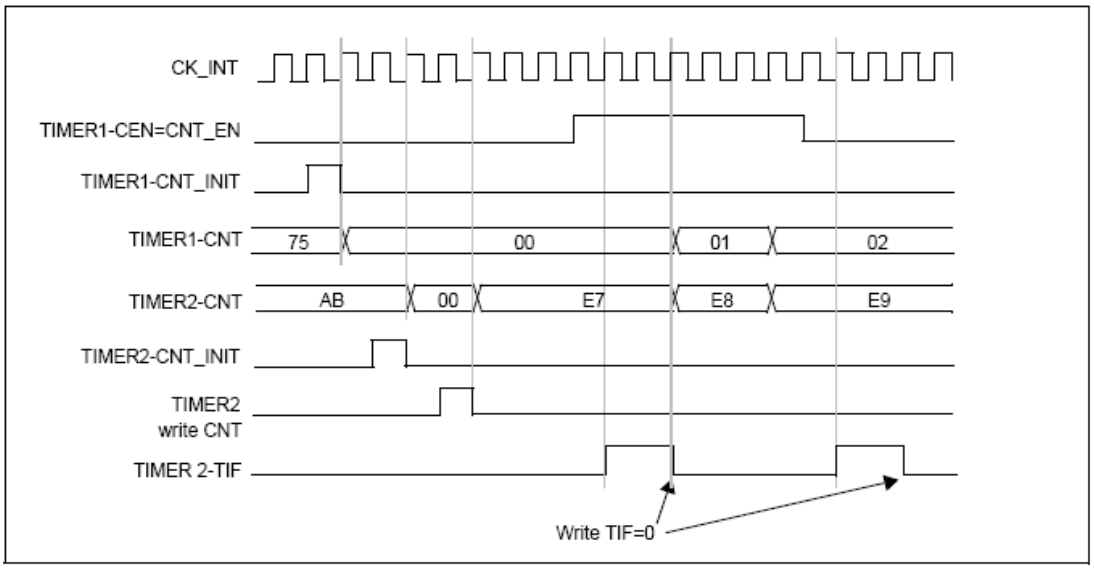


在 图 114 的例子中，在定时器 2 启动之前，它们的计数器和预分频器未被初始化，因此它们从当前的数值开始计数。可以在启动定时器 1 之前复位 2 个定时器，使它们从给定的数值开始，即在定时器计数器中写入需要的任意数值。写TIMx_EGR寄存器的UG位即可复位定时器。

在下一个例子中，需要同步定时器 1 和定时器 2。定时器 1 是主模式并从 0 开始，定时器 2 是从模式并从 0xE7 开始；2 个定时器的预分频器系数相同。写 0 到 TIM1_CR1 的 CEN 位将禁止定时器 1，定时器 2 随即停止。

- 配置定时器 1 为主模式，送出输出比较 1 参考信号(OC1REF)做为触发输出(TIM1_CR2 寄存器的 MMS=100)。
- 配置定时器 1 的 OC1REF 波形(TIM1_CCMR1 寄存器)。
- 配置定时器 2 从定时器 1 获得输入触发(TIM2_SMCR 寄存器的 TS=001)
- 配置定时器 2 为门控模式(TIM2_SMCR 寄存器的 SMS=101)
- 置 TIM1_EGR 寄存器的 UG=1，复位定时器 1。
- 置 TIM2_EGR 寄存器的 UG=1，复位定时器 1。
- 写 0xE7 至定时器 2 的计数器(TIM2_CNTL)，初始化它为 0xE7。
- 置 TIM2_CR1 寄存器的 CEN=1 以使能定时器 2。
- 置 TIM1_CR1 寄存器的 CEN=1 以启动定时器 1。
- 置 TIM1_CR1 寄存器的 CEN=0 以停止定时器 1。

图115 通过使能定时器 1 可以控制定时器 2

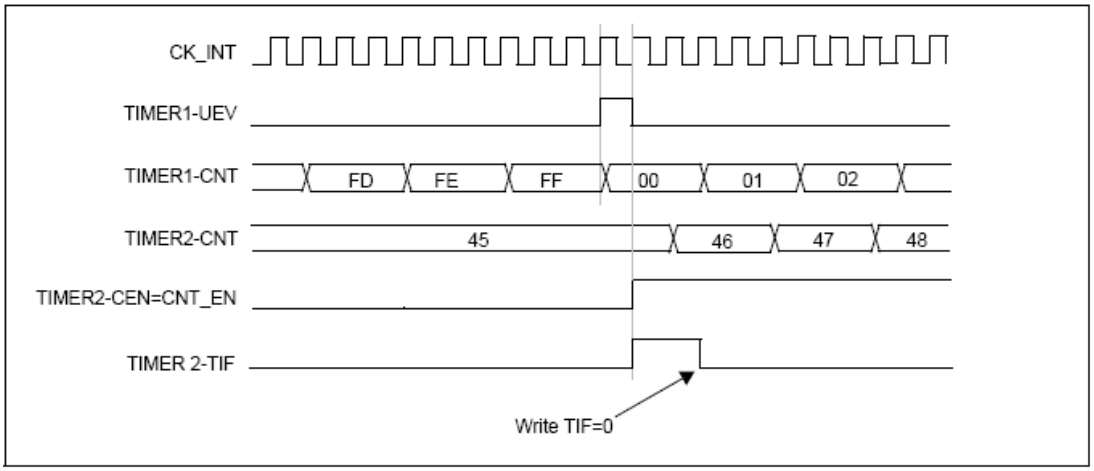


使用一个定时器去启动另一个定时器

在这个例子中，使用定时器 1 的更新事件使能定时器 2。参考 图 113 的连接。一旦定时器 1 产生更新事件，定时器 2 即从它当前的数值(可以是非 0)依分频的内部时钟开始计数。在收到触发信号时，定时器 2 的CEN位被自动地置 1，同时计数器开始计数直到写 0 到TIM2_CR1 寄存器的CEN位。两个定时器的时钟频率都是由预分频器对CK_INT除以 3($f_{CK\_CNT}=f_{CK\_INT}/3$)。

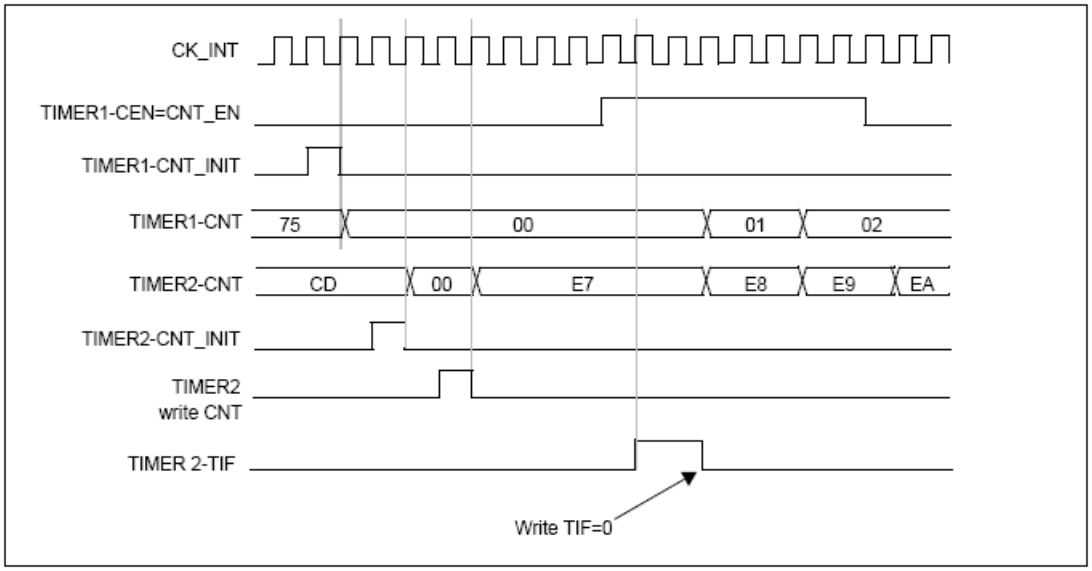
- 配置定时器 1 为主模式，送出它的更新事件(UEV)做为触发输出(TIM1_CR2 寄存器的 MMS=010)。
- 配置定时器 1 的周期(TIM1_ARR 寄存器)。
- 配置定时器 2 从定时器 1 获得输入触发(TIM2_SMCR 寄存器的 TS=001)
- 配置定时器 2 为触发模式(TIM2_SMCR 寄存器的 SMS=110)
- 置 TIM1_CR1 寄存器的 CEN=1 以启动定时器 1。

图116 使用定时器 1 的更新触发定时器 2



在上一个例子中，可以在启动计数之前初始化两个计数器。图 117 显示在与 图 116 相同配置情况下，使用触发模式而不是门控模式(TIM2_SMCR寄存器的 SMS=110)的动作。

图117 利用定时器 1 的使能触发定时器 2



使用一个定时器作为另一个的预分频器

如：可以配置定时器 1 作为定时器 2 的预分频器。参考 图 113，进行下述操作：

- 配置定时器 1 为主模式，送出它的更新事件 UEV 做为触发输出(TIM1_CR2 寄存器的 MMS=’010’)。每次计数器溢出时输出一个周期信号。
- 配置定时器 1 的周期(TIM1_ARR 寄存器)。
- 配置定时器 2 从定时器 1 获得输入触发(TIM2_SMCR 寄存器的 TS=001)
- 配置定时器 2 使用外部时钟模式(TIM2_SMCR 寄存器的 SMS=111)
- 置 TIM1_CR2 寄存器的 CEN=1 以启动定时器 2。
- 置 TIM1_CR1 寄存器的 CEN=1 以启动定时器 1。

使用一个外部触发同步地启动 2 个定时器

在这个例子中，设置当定时器 1 的TI1 输入上升时使能定时器 1，使能定时器 1 的同时使能定时器 2。参考 图 113 的连结。为保证计数器的对齐，定时器 1 必须配置为主/从模式(对应TI1 为从，对应定时器 2 为主)：

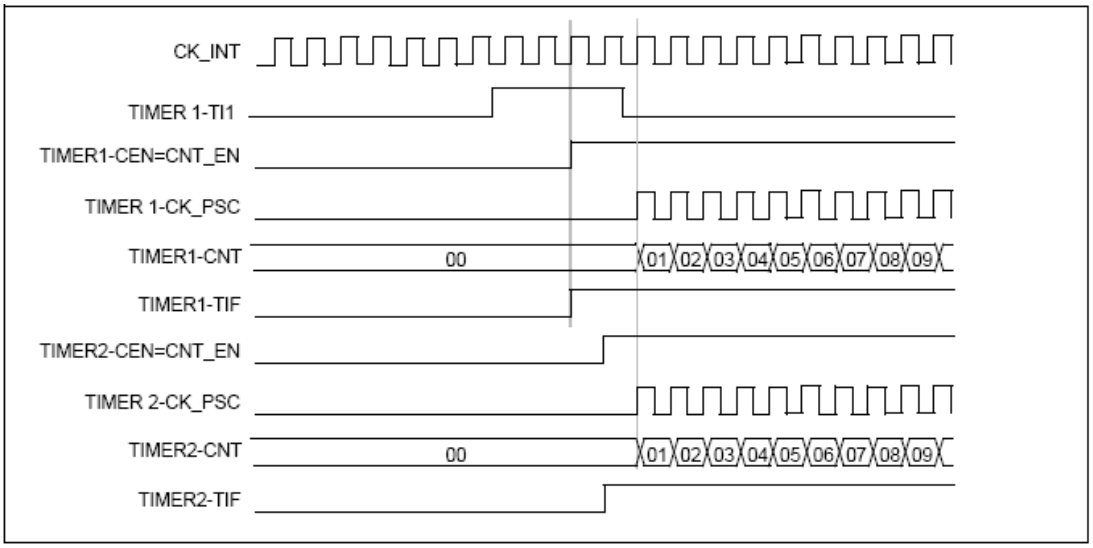
- 配置定时器 1 为主模式，送出它的使能做为触发输出(TIM1_CR2 寄存器的 MMS=’001’)。
- 配置定时器 1 为从模式，从 TI1 获得输入触发(TIM1_SMCR 寄存器的 TS=’100’)。
- 配置定时器 1 为触发模式(TIM1_SMCR 寄存器的 SMS=’110’)。
- 配置定时器 1 为主/从模式，TIM1_SMCR 寄存器的 MSM=’1’。

- 配置定时器 2 从定时器 1 获得输入触发(TIM2_SMCR 寄存器的 TS=001)
- 配置定时器 2 为触发模式(TIM2_SMCR 寄存器的 SMS='110')。

当定时器 1 的 TI1 上出现一个上升沿时，两个定时器同步地依内部时钟开始计数，两个 TIF 标志也同时设置。

注： 在这个例子中，在启动之前两个定时器都被初始化，两个计数器都从 0 开始，但可以通过写入任意一个计数器寄存器(TIMx_CNT)在定时器间插入一个偏移。下图中能看到主/从模式下在定时器 1 的 CNT_EN 和 CK_PSC 之间有个延迟。

图118 使用定时器 1 的 TI1 输入触发定时器 1 和定时器 2



13.4.16 调试模式

当微控制器进入调试模式(Cortex-M3 核心停止)，根据 DBG 模块中 DBG_TIMx_STOP 的设置，TIMx 计数器可以或者继续正常操作，或者停止。详见随后调试模块章节。

13.5 TIMx寄存器描述

关于在寄存器描述里面所用到的缩写，详见第 1 章。

13.5.1 控制寄存器 1(TIMx_CR1)

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
保留						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，始终读为0。
位9:8	CKD[1:0]: 时钟分频因子 这2位定义在定时器时钟(CK_INT)频率与数字滤波器(ETR,TIx)使用的采样频率之间的分频比例。 00：$t_{DTS} = t_{CK\_INT}$ 01：$t_{DTS} = 2 \times t_{CK\_INT}$ 10：$t_{DTS} = 4 \times t_{CK\_INT}$ 11：保留
位7	ARPE：自动重装载预装载允许位 0：TIMx_ARR寄存器没有缓冲 1：TIMx_ARR寄存器被装入缓冲器
位6:5	CMS[1:0]：选择中央对齐模式 00：边沿对齐模式。计数器依据方向位(DIR)向上或向下计数。 01：中央对齐模式1。计数器交替地向上和向下计数。配置为输出的通道(TIMx_CCMRx寄存器中CCxS=00)的输出比较中断标志位，只在计数器向下计数时被设置。 10：中央对齐模式2。计数器交替地向上和向下计数。计数器交替地向上和向下计数。配置为输出的通道(TIMx_CCMRx寄存器中CCxS=00)的输出比较中断标志位，只在计数器向上计数时被设置。 11：中央对齐模式3。计数器交替地向上和向下计数。计数器交替地向上和向下计数。配置为输出的通道(TIMx_CCMRx寄存器中CCxS=00)的输出比较中断标志位，在计数器向上和向下计数时均被设置。 注：在计数器开启时(CEN=1)，不允许从边沿对齐模式转换到中央对齐模式。
位4	DIR：方向 0：计数器向上计数 1：计数器向下计数 注：当计数器配置为中央对齐模式或编码器模式时，该位为只读。
位3	OPM：单脉冲模式 0：在发生更新事件时，计数器不停止 1：在发生下一次更新事件(清除CEN位)时，计数器停止。

位2	URS：更新请求源 软件通过该位选择UEV事件的源 0：如果允许产生更新中断或DMA请求，则下述任一事件产生一个更新中断或DMA请求： – 计数器溢出/下溢 – 设置UG位 – 从模式控制器产生的更新 1：如果允许产生更新中断或DMA请求，则只有计数器溢出/下溢产生一个更新中断或DMA请求
位1	UDIS：禁止更新 软件通过该位允许/禁止UEV事件的产生 0：允许UEV。更新(UEV)事件由下述任一事件产生： – 计数器溢出/下溢 – 设置UG位 – 从模式控制器产生的更新 被缓存的寄存器被装入它们的预装载值。 1：禁止UEV。不产生更新事件，影子寄存器(ARR,PSC,CCRx)保持它们的值。如果设置了UG位或从模式控制器发出了一个硬件复位，则计数器和预分频器被重新初始化。
位0	CEN：允许计数器 0：禁止计数器 1：开启计数器 注：在软件设置了CEN位后，外部时钟、门控模式和编码器模式才能工作。触发模式可以自动地通过硬件设置CEN位。 在单脉冲模式下，当发生更新事件时，CEN被自动清除。

13.5.2 控制寄存器 2(TIMx_CR2)

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
保留								TI1S	MMS[2:0]			CCDS	保留		
								rw	rw	rw	rw	rw			

位15:8	保留，始终读为0。
位7	TI1S：TI1选择 0：TIMx_CH1管脚连到TI1输入。 1：TIMx_CH1、TIMx_CH2和TIMx_CH3管脚经异或后连到TI1输入。 见上一章的“与霍尔元件的接口”一节。

位6:4	MMS[1:0]：主模式选择 这两位用于选择在主模式下送到从定时器的同步信息(TRGO)。可能的组合如下： 000：复位 – TIMx_EGR寄存器的UG位被用于作为触发输出(TRGO)。如果触发输入(复位模式下的从模式控制器)产生复位，则TRGO上的信号相对实际的复位会有一个延迟。 001：允许 – 计数器使能信号CNT_EN被用于作为触发输出(TRGO)。有时需要在同一时间启动多个定时器或控制从定时器的一个窗口。计数器使能信号是通过CEN控制位和门控模式下的触发输入信号的逻辑或产生。当计数器使能信号受控于触发输入时，TRGO上会有一个延迟，除非选择了主/从模式(见TIMx_SMCR寄存器中MSM位的描述)。 010：更新 – 更新事件被选为触发输入(TRGO)。例如，一个主定时器的时钟可以被用作一个从定时器的预分频器。 011：比较脉冲 – 一旦发生一次捕获或一次比较成功时，当要设置CC1IF标志时(即是它已经为高)，触发输出送出一个正脉冲(TRGO)。 100：比较 – OC1REF信号被用于作为触发输出(TRGO)。 101：比较 – OC2REF信号被用于作为触发输出(TRGO)。 110：比较 – OC3REF信号被用于作为触发输出(TRGO)。 111：比较 – OC4REF信号被用于作为触发输出(TRGO)。
位3	CCDS：捕获/比较的DMA选择 0：当发生CCx事件时，送出CCx的DMA请求。 1：当发生更新事件时，送出CCx的DMA请求。
位2:0	保留，始终读为0。

13.5.3 从模式控制寄存器(TIMx_SMCR)

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]			保留	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 该位选择是用ETR还是ETR的反相来作为触发操作 0：ETR不反相，高电平或上升沿有效 1：ETR被反相，低电平或下降沿有效
位14	ECE：外部时钟允许位 该位启用外部时钟模式2 0：禁止外部时钟模式2 1：启用外部时钟模式2。计数器由ETRF信号上的任意有效上升沿驱动。 注1：设置ECE位与选择外部时钟模式1并将TRGI连到ETRF(SMS=111和TS=111)具有相同功效。 注2：下述从模式可以与外部时钟模式2同时使用：复位模式，门控模式和触发模式；但是，这时TRGI不能连到ETRF(TS位不能是111)。

位13:12	ETPS[1:0]：外部触发预分频 外部触发信号ETRP的频率必须最多是CK_INT频率的1/4。当输入较快的外部时钟时，可以使用预分频降低ETRP的频率。 00：关闭预分频 01：ETRP频率除以2 10：ETRP频率除以4 11：ETRP频率除以8
位11:8	ETF[3:0]：外部触发滤波 这些位定义了对ETRP信号采样的频率和对ETRP数字滤波的带宽。实际上，数字滤波器是一个事件计数器，它记录到N个事件后会产生一个输出的跳变。 0000：无滤波器，以f_{DTS}采样 0001：采样频率$f_{SAMPLING}=f_{CK_INT}$，N=2 0010：采样频率$f_{SAMPLING}=f_{CK_INT}$，N=4 0011：采样频率$f_{SAMPLING}=f_{CK_INT}$，N=8 0100：采样频率$f_{SAMPLING}=f_{DTS}/2$，N=6 0101：采样频率$f_{SAMPLING}=f_{DTS}/2$，N=8 0110：采样频率$f_{SAMPLING}=f_{DTS}/4$，N=6 0111：采样频率$f_{SAMPLING}=f_{DTS}/4$，N=8 1000：采样频率$f_{SAMPLING}=f_{DTS}/8$，N=6 1001：采样频率$f_{SAMPLING}=f_{DTS}/8$，N=8 1010：采样频率$f_{SAMPLING}=f_{DTS}/16$，N=5 1011：采样频率$f_{SAMPLING}=f_{DTS}/16$，N=6 1100：采样频率$f_{SAMPLING}=f_{DTS}/16$，N=8 1101：采样频率$f_{SAMPLING}=f_{DTS}/32$，N=5 1110：采样频率$f_{SAMPLING}=f_{DTS}/32$，N=6 1111：采样频率$f_{SAMPLING}=f_{DTS}/32$，N=8
位7	MSM：主/从模式 0：无作用 1：触发输入(TRGI)上的事件被延迟了，以允许在当前定时器(通过TRGO)与它的从定时器间的完美同步。这对要求把几个定时器同步到一个单一的外部事件时是非常有用的。
位6:4	TS[2:0]：触发选择 这3位选择用于同步计数器的触发输入。 000：内部触发器0(ITR0)，TIM1 001：内部触发器0(ITR1)，TIM2 010：内部触发器0(ITR1)，TIM3 011：内部触发器0(ITR1)，TIM4 100：TI1的边沿检测器(TI1F_ED) 101：滤波后的定时器输入1(TI1FP1) 110：滤波后的定时器输入2(TI2FP2) 111：外部触发输入(ETRF) 注：为避免在信号转变时产生错误的边沿检测，必须在未使用这些位(如SMS=000)时修改它们。
位3	保留，始终读为0。

位2:0	SMS：从模式选择 当选择了外部信号，触发信号(TRGI)的有效边沿与选中的外部输入极性相关(见输入控制寄存器和控制寄存器的说明) 000：关闭从模式 – 如果CEN=1，则预分频器直接由内部时钟驱动。 001：编码器模式1 – 根据TI1FP1的电平，计数器在TI2FP2的边沿向上/下计数。 010：编码器模式2 – 根据TI2FP2的电平，计数器在TI1FP1的边沿向上/下计数。 011：编码器模式3 – 根据其他输入的电平，计数器在TI1FP1和TI2FP2的边沿向上/下计数。 100：复位模式 – 选中的触发输入(TRGI)的上升沿重新初始化计数器，并且产生一个更新寄存器的信号。 101：门控模式 – 当触发输入(TRGI)为高时，计数器的时钟开启。一旦触发输入变为低，则计数器停止(但不复位)。计数器的启动和停止都是受控的。 110：触发模式 – 计数器在触发输入TRGI的上升沿启动(但不复位)，只有计数器的启动是受控的。 111：外部时钟模式1 – 选中的触发输入(TRGI)的上升沿驱动计数器。 注：如果TI1F_EN被选为触发输入(TS=100)时，不要使用门控模式。这是因为，TI1F_ED在每次TI1F变化时输出一个脉冲，然而门控模式是要检查触发输入的电平。

13.5.4 DMA/中断使能寄存器(TIMx_DIER)

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
保留	TDE	保留	CC4DE	CC3DE	CC2DE	CC1DE	UDE	保留	TIE	保留	CC4IE	CC3IE	CC2IE	CC1IE	UIE
	rw		rw	rw	rw	rw	rw		rw		rw	rw	rw	rw	rw

位15	保留，始终读为0。
位14	TDE：允许触发DMA请求 0：禁止触发DMA请求 1：允许触发DMA请求
位13	保留，始终读为0。
位12	CC4DE：允许捕获/比较4的DMA请求 0：禁止捕获/比较4的DMA请求 1：允许捕获/比较4的DMA请求
位11	CC3DE：允许捕获/比较3的DMA请求 0：禁止捕获/比较3的DMA请求 1：允许捕获/比较3的DMA请求
位10	CC2DE：允许捕获/比较2的DMA请求 0：禁止捕获/比较2的DMA请求 1：允许捕获/比较2的DMA请求
位9	CC1DE：允许捕获/比较1的DMA请求 0：禁止捕获/比较1的DMA请求 1：允许捕获/比较1的DMA请求

位8	UDE：允许更新的DMA请求 0：禁止更新的DMA请求 1：允许更新的DMA请求
位7	保留，始终读为0。
位6	TIE：允许触发中断 0：禁止触发中断 1：允许触发中断
位5	保留，始终读为0。
位4	CC4IE：允许捕获/比较4中断 0：禁止捕获/比较4中断 1：允许捕获/比较4中断
位3	CC3IE：允许捕获/比较3中断 0：禁止捕获/比较3中断 1：允许捕获/比较3中断
位2	CC2IE：允许捕获/比较2中断 0：禁止捕获/比较2中断 1：允许捕获/比较2中断
位1	CC1IE：允许捕获/比较1中断 0：禁止捕获/比较1中断 1：允许捕获/比较1中断
位0	UIE：允许更新中断 0：禁止更新中断 1：允许更新中断

13.5.5 状态寄存器(TIMx_SR)

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
保留			CC4OF	CC3OF	CC2OF	CC1OF	保留		TIF	保留	CC4IF	CC3IF	CC2IF	CC1IF	UIF
			rc	rc	rc	rc			rc		rc	rc	rc	rc	rc

位15:13	保留，始终读为0。
位12	**CC4OF**：捕获/比较4 过捕获标记 参见CC1OF描述
位11	**CC3OF**：捕获/比较3 过捕获标记 参见CC1OF描述
位10	**CC2OF**：捕获/比较2 过捕获标记 参见CC1OF描述

位9	**CC1OF**：捕获/比较1 过捕获标记 仅当相应的通道被配置为输入捕获时，该标记可由硬件置1。写0可清除该位。 0：无过捕获产生； 1：CC1IF置1时，计数器的值已经被捕获到TIMx_CCR1寄存器。
位8:7	保留，始终读为0。
位6	**TIF**：触发器中断标记 当发生触发事件（当从模式控制器处于除门控模式外的其它模式时,在TRGI输入端检测到有效边沿，或或门控模式下的任一边沿）时由硬件对该位置1。它由软件清0。 0：无触发器事件产生； 1：触发器中断等待响应
位5	保留，始终读为0。
位4	**CC4IF**：捕获/比较4 中断标记 参考CC1IF描述
位3	**CC3IF**：捕获/比较3 中断标记 参考CC1IF描述
位2	**CC2IF**：捕获/比较2 中断标记 参考CC1IF描述
位1	**CC1IF**：捕获/比较1 中断标记 **如果通道CC1配置为输出模式：** 当计数器值与比较值匹配时该位由硬件置1，但在中心对称模式下除外(参考TIMx_CR1寄存器的CMS位)。它由软件清0。 0：无匹配发生； 1：TIMx_CNT的值与TIMx_CCR1的值匹配。 **如果通道CC1配置为输入模式：** 当捕获事件发生时该位由硬件置1，它由软件清0或通过读TIMx_CCR1清0。 0：无输入捕获产生； 1：输入捕获产生并且计数器值已装入TIMx_CCR1(在IC1上检测到与所选极性相同的边沿)。
位0	**UIF**：更新中断标记 当产生更新事件时该位由硬件置1。它由软件清0。 0：无更新事件产生； 1：更新事件等待响应。当寄存器被更新时该位由硬件置1： – 若TIMx_CR1寄存器的UDIS=0，当REP_CNT=0时产生更新事件(重复向下计数器上溢或下溢时)； – 若TIMx_CR1寄存器的UDIS=0、URS=0，当TIMx_EGR寄存器的UG=1时产生更新事件(软件对CNT重新初始化)； – 若TIMx_CR1寄存器的UDIS=0、URS=0，当CNT被触发事件重初始化时产生更新事件。（参考同步控制寄存器的说明）

13.5.6 事件产生寄存器(TIMx_EGR)

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
保留									TG	保留	CC4G	CC3G	CC2G	CC1G	UG
									w		w	w	w	w	w

位15:7	保留，始终读为0。
位6	**TG**：产生触发事件 该位由软件置1，用于产生一个触发事件，由硬件自动清0。 0：无动作； 1：TIMx_SR寄存器的TIF=1，若开启对应的中断和DMA，则产生相应的中断和DMA。
位5	保留，始终读为0。
位4	**CC4G**：产生捕获/比较4事件 参考CC1G描述
位3	**CC3G**：产生捕获/比较3事件 参考CC1G描述
位2	**CC2G**：产生捕获/比较2事件 参考CC1G描述
位1	**CC1G**：产生捕获/比较1事件 该位由软件置1，用于产生一个捕获/比较事件，由硬件自动清0。 0：无动作； 1：在通道CC1上产生一个捕获/比较事件： **若通道CC1配置为输出：** 设置CC1IF=1，若开启对应的中断和DMA，则产生相应的中断和DMA。 **若通道CC1配置为输入：** 当前的计数器值捕获至TIMx_CCR1寄存器，设置CC1IF=1，若开启对应的中断和DMA，则产生相应的中断和DMA。若CC1IF已经为1，则设置CC1OF=1。
位0	**UG**：产生更新事件 该位由软件置1，由硬件自动清0。 0：无动作； 1：重初始化计数器，并产生一个更新事件。注意预分频器的计数器也被清0(但是预分频系数不变)。若在中心对称模式下或DIR=0(向上计数)则计数器被清0，若DIR=1(向下计数)则计数器取TIMx_ARR的值。

13.5.7 捕获/比较模式寄存器 1(TIMx_CCMR1)

偏移地址：0x18
复位值：0x0000

通道可用于输入(捕获模式)或输出(比较模式)，通道的方向由相应的 CCxS 定义。该寄存器其它位的作用在输入和输出模式下不同。OCxx 描述了通道在输出模式下的功能，ICxx 描述了通道在输出模式下的功能。因此你必须注意，同一个位在输出模式和输入模式下的功能是不同的。

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC2CE	OC2M[2:0]			OC2PE	OC2FE	CC2S[1:0]		OC1CE	OC1M[2:0]			OC1PE	OC1FE	CC1S[1:0]	
IC2F[3:0]				IC2PSC[1:0]				IC1F[3:0]				IC1PSC[1:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

输出比较模式：

位15	OC2CE：输出比较2清0使能
位14:12	0C2M[2:0]：输出比较2模式
位11	OC2PE：输出比较2预装载使能
位10	OC2FE：输出比较2快速使能
位9:8	CC2S[1:0]：捕获/比较2选择。 该位定义通道的方向（输入/输出），及输入脚的选择： 00：CC2通道被配置为输出； 01：CC2通道被配置为输入，IC2映射在TI2上； 10：CC2通道被配置为输入，IC2映射在TI1上； 11：CC2通道被配置为输入，IC2映射在TRC上。此模式仅工作在内部触发器输入被选中时（由TIMx_SMCR寄存器的TS位选择）。 **注：**CC2S仅在通道关闭时(TIMx_CCER寄存器的CC2E=0)才是可写的。
位7	OC1CE：输出比较1清0使能 0：OC1REF 不受ETRF输入的影响； 1：一旦检测到ETRF输入高电平，清除OC1REF=0。

位6:4	0C1M[2:0]：输出比较1模式 该位定义了输出参考信号OC1REF的动作，而OC1REF决定了OC1、OC1N的值。OC1REF是高电平有效，而OC1、OC1N的有效电平取决于CC1P、CC1NP位。 000：冻结。输出比较寄存器TIMx_CCR1与计数器TIMx_CNT间的比较对OC1REF不起作用； 001：匹配时设置通道1为有效电平。当计数器TIMx_CNT的值与捕获/比较寄存器1(TIMx_CCR1)相同时，强制OC1REF为高。 010：匹配时设置通道1为无效电平。当计数器TIMx_CNT的值与捕获/比较寄存器1(TIMx_CCR1)相同时，强制OC1REF为低。 011：翻转。当TIMx_CCR1=TIMx_CNT时，翻转OC1REF的电平。 100：强制为无效电平。强制OC1REF为低。 101：强制为有效电平。强制OC1REF为高。 110：PWM模式1－ 在向上计数时，一旦TIMx_CNT<TIMx_CCR1时通道1为有效电平，否则为无效电平；在向下计数时，一旦TIMx_CNT>TIMx_CCR1时通道1为无效电平(OC1REF=0)，否则为有效电平(OC1REF=1)。 111：PWM模式2－ 在向上计数时，一旦TIMx_CNT<TIMx_CCR1时通道1为无效电平，否则为有效电平；在向下计数时，一旦TIMx_CNT>TIMx_CCR1时通道1为有效电平，否则为无效电平。 注1：一旦LOCK级别设为3(TIMx_BDTR寄存器中的LOCK位)并且CC1S=00(该通道配置成输出)则该位不能被修改。 注2：在PWM模式1或PWM模式2中，只有当比较结果改变了或在输出比较模式中从冻结模式切换到PWM模式时，OC1REF电平才改变。
位3	**OC1PE**：输出比较1预装载使能 0：禁止TIMx_CCR1寄存器的预装载功能，可随时写入TIMx_CCR1寄存器，且新值马上起作用。 1：开启TIMx_CCR1寄存器的预装载功能，读写操作仅对预装载寄存器操作，TIMx_CCR1的预装载值在更新事件到来时被载入当前寄存器中。 注1：一旦LOCK级别设为3(TIMx_BDTR寄存器中的LOCK位)并且CC1S=00(该通道配置成输出)则该位不能被修改。 注2：仅在单脉冲模式下，可以在未确认预装载寄存器情况下使用PWM模式，否则其动作不确定。
位2	**OC1FE**：输出比较1 快速使能 该位用于加快CC输出对触发器输入事件的响应。 0：根据计数器与CCR1的值，CC1正常操作，即使触发器是打开的。当触发器的输入有一个有效沿时，激活CC1输出的最小延时为5个时钟周期。 1：输入到触发器的有效沿的作用就象发生了一次比较匹配。因此，OC被设置为比较电平而与比较结果无关。采样触发器的有效沿和CC1输出间的延时被缩短为3个时钟周期。 OCFE的只在通道被配置成PWM1或PWM2模式时起作用。
位1:0	CC1S[1:0]：捕获/比较1 选择。 这2位定义通道的方向（输入/输出），及输入脚的选择： 00：CC1通道被配置为输出； 01：CC1通道被配置为输入，IC1映射在TI1上； 10：CC1通道被配置为输入，IC1映射在TI2上； 11：CC1通道被配置为输入，IC1映射在TRC上。此模式仅工作在内部触发器输入被选中时（由TIMx_SMCR寄存器的TS位选择）。 注：CC1S仅在通道关闭时(TIMx_CCER寄存器的CC1E=0)才是可写的。

输入捕获模式：

位15:12	IC2F：输入捕获2滤波器
位11:10	IC2PSC[1:0]：输入/捕获2预分频器
位9:8	CC2S[1:0]：捕获/比较2选择。 这2位定义通道的方向（输入/输出），及输入脚的选择： 00：CC2通道被配置为输出； 01：CC2通道被配置为输入，IC2映射在TI2上； 10：CC2通道被配置为输入，IC2映射在TI1上； 11：CC2通道被配置为输入，IC2映射在TRC上。此模式仅工作在内部触发器输入被选中时（由TIMx_SMCR寄存器的TS位选择）。 注：CC2S仅在通道关闭时(TIMx_CCER寄存器的CC2E=0)才是可写的。
位7:4	IC1F[3:0]：输入捕获1滤波器 这几位定义了TI1输入的采样频率及数字滤波器长度。数字滤波器由一个事件计数器组成，它记录到N个事件后会产生一个输出的跳变： 0000：无滤波器，以f_{DTS}采样　　1000：采样频率$f_{SAMPLING}=f_{DTS}/8$，N=6 0001：采样频率$f_{SAMPLING}=f_{CK_INT}$，N=2　　1001：采样频率$f_{SAMPLING}=f_{DTS}/8$，N=8 0010：采样频率$f_{SAMPLING}=f_{CK_INT}$，N=4　　1010：采样频率$f_{SAMPLING}=f_{DTS}/16$，N=5 0011：采样频率$f_{SAMPLING}=f_{CK_INT}$，N=8　　1011：采样频率$f_{SAMPLING}=f_{DTS}/16$，N=6 0100：采样频率$f_{SAMPLING}=f_{DTS}/2$，N=6　　1100：采样频率$f_{SAMPLING}=f_{DTS}/16$，N=8 0101：采样频率$f_{SAMPLING}=f_{DTS}/2$，N=8　　1101：采样频率$f_{SAMPLING}=f_{DTS}/32$，N=5 0110：采样频率$f_{SAMPLING}=f_{DTS}/4$，N=6　　1110：采样频率$f_{SAMPLING}=f_{DTS}/32$，N=6 0111：采样频率$f_{SAMPLING}=f_{DTS}/4$，N=8　　1111：采样频率$f_{SAMPLING}=f_{DTS}/32$，N=8 注：在现在的芯片版本中，当ICxF[3:0]=1,2或3时，公式中的fDTS由CK_INT替代。
位3:2	IC1PSC[1:0]：输入/捕获1预分频器 这2位定义了CC1输入（IC1）的预分频系数。一旦CC1E=0(TIMx_CCER寄存器中)，则预分频器复位。 00：无预分频器，捕获输入口上检测到的每一个边沿都触发一次捕获； 01：每2个事件触发一次捕获； 10：每4个事件触发一次捕获； 11：每8个事件触发一次捕获。
位1:0	**CC1S[1:0]**：捕获/比较1选择。 这2位定义通道的方向（输入/输出），及输入脚的选择： 00：CC1通道被配置为输出； 01：CC1通道被配置为输入，IC1映射在TI1上； 10：CC1通道被配置为输入，IC1映射在TI2上； 11：CC1通道被配置为输入，IC1映射在TRC上。此模式仅工作在内部触发器输入被选中时（由TIMx_SMCR寄存器的TS位选择）。 **注**：CC1S仅在通道关闭时(TIMx_CCER寄存器的CC1E=0)才是可写的。

13.5.8 捕获/比较模式寄存器 2(TIMx_CCMR2)

偏移地址：0x1C
复位值：0x0000
参看以上 CCMR1 寄存器的描述

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC4CE	OC4M[2:0]			OC4PE	OC4FE	CC4S[1:0]		OC3CE	OC3M[2:0]			OC3PE	OC3FE	CC3S[1:0]	
IC4F[3:0]				IC4PSC[1:0]				IC3F[3:0]				IC3PSC[1:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

输出比较模式：

位15	OC4CE：输出比较4清0使能
位14:12	0C4M[2:0]：输出比较4模式
位11	OC4PE：输出比较4预装载使能
位10	OC4FE：输出比较4快速使能
位9:8	CC4S[1:0]：捕获/比较4选择。 这2位定义通道的方向（输入/输出），及输入脚的选择： 00：CC4通道被配置为输出； 01：CC4通道被配置为输入，IC4映射在TI4上； 10：CC4通道被配置为输入，IC4映射在TI3上； 11：CC4通道被配置为输入，IC4映射在TRC上。此模式仅工作在内部触发器输入被选中时（由TIMx_SMCR寄存器的TS位选择）。 **注：**CC4S仅在通道关闭时(TIMx_CCER寄存器的CC4E=0)才是可写的。
位7	OC3CE：输出比较3清0使能
位6:4	0C3M[2:0]：输出比较3模式
位3	OC3PE：输出比较3预装载使能
位2	OC3FE：输出比较3快速使能
位1:0	CC3S[1:0]：捕获/比较3选择。 这2位定义通道的方向（输入/输出），及输入脚的选择： 00：CC3通道被配置为输出； 01：CC3通道被配置为输入，IC3映射在TI3上； 10：CC3通道被配置为输入，IC3映射在TI4上； 11：CC3通道被配置为输入，IC3映射在TRGI上。此模式仅工作在内部触发器输入被选中时（由TIMx_SMCR寄存器的TS位选择）。 注：CC3S仅在通道关闭时(TIMx_CCER寄存器的CC3E=0)才是可写的。

输入捕获模式：

位15:12	IC4F：输入捕获4滤波器
位11:10	IC4PSC[1:0]：输入/捕获4预分频器
位9:8	CC2S[1:0]：捕获/比较4选择。 这2位定义通道的方向（输入/输出），及输入脚的选择： 00：CC4通道被配置为输出； 01：CC4通道被配置为输入，IC4映射在TI4上； 10：CC4通道被配置为输入，IC4映射在TI3上； 11：CC4通道被配置为输入，IC4映射在TRC上。此模式仅工作在内部触发器输入被选中时（由TIMx_SMCR寄存器的TS位选择）。 注：CC4S仅在通道关闭时(TIMx_CCER寄存器的CC4E=0)才是可写的。
位7:4	IC3F[3:0]：输入捕获3滤波器
位3:2	IC3PSC[1:0]：输入/捕获3预分频器
位1:0	CC3S[1:0]：捕获/比较3选择。 这2位定义通道的方向（输入/输出），及输入脚的选择： 00：CC3通道被配置为输出； 01：CC3通道被配置为输入，IC3映射在TI3上； 10：CC3通道被配置为输入，IC3映射在TI4上； 11：CC3通道被配置为输入，IC1映射在TRC上。此模式仅工作在内部触发器输入被选中时（由TIMx_SMCR寄存器的TS位选择）。 注：CC3S仅在通道关闭时(TIMx_CCER寄存器的CC3E=0)才是可写的。

13.5.9 捕获/比较使能寄存器(TIMx_CCER)

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
保留		CC4P	CC4E	保留		CC3P	CC3E	保留		CC2P	CC2E	保留		CC1P	CC1E
		rw	rw			rw	rw			rw	rw			rw	rw

位15:14	保留，始终读为0。
位13	CC4P：输入/捕获4输出极性。参考CC1P的描述。
位12	CC4E：输入/捕获4输出使能。参考CC1E 的描述。
位11:10	保留，始终读为0。
位9	CC3P：输入/捕获3输出极性。参考CC1P的描述。
位8	CC3E：输入/捕获3输出使能。参考CC1E 的描述。
位7:6	保留，始终读为0。
位5	CC2P：输入/捕获2输出极性。参考CC1P的描述。
位4	CC2E：输入/捕获2输出使能。参考CC1E的描述。
位3:2	保留，始终读为0。

位1	CC1P：输入/捕获1输出极性 **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：** 该位选择是IC1还是IC1的反相信号作为触发或捕获信号。 0：不反相：捕获发生在IC1的上升沿；当用作外部触发器时，IC1不反相。 1：反相：捕获发生在IC1的下降沿；当用作外部触发器时，IC1反相。
位0	CC1E：输入/捕获1输出使能 **CC1通道配置为输出：** 0： 关闭－ OC1禁止输出。 1： 开启－ OC1信号输出到对应的输出引脚。 **CC1 通道配置为输入：** 该位决定了计数器的值是否能捕获入TIMx_CCR1寄存器。 0：捕获禁止； 0：捕获使能。

表40 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： 连接到标准OCx通道的外部I/O管脚状态，取决于OCx通道状态和GPIO以及AFIO寄存器。

13.5.10 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]：计数器的值

13.5.11 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]：预分频器的值 计数器的时钟频率CK_CNT等于fCK_PSC/(PSC[15:0]+1)。 PSC包含了当更新事件产生时装入当前预分频器寄存器的值。

13.5.12 自动重装载寄存器(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]: 自动重装载的值 ARR是将要装载入实际的自动重装载寄存器的数值。 详细参考13.4.1：时基单元有关ARR的更新和动作。 当自动重装载的值为空时，计数器不工作。

13.5.13 捕获/比较寄存器 1(TIMx_CCR1)

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的值 **若CC1通道配置为输出：** CCR1是装入当前捕获/比较1寄存器的值（预装载值）。 如果在TIMx_CCMR1寄存器(OC1PE位)中未选择预装载特性，其始终装入当前寄存器中。否则，只有当更新事件发生时，此预装载值才装入当前捕获/比较1寄存器中。 当前捕获/比较寄存器包含了与计数器TIMx_CNT比较的值，并且在OC1端口上输出信号。 **若CC1通道配置为输入：** CCR1包含了由上一次输入捕获1事件（IC1）传输的计数器值。

13.5.14 捕获/比较寄存器 2(TIMx_CCR2)

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的值 **若CC2通道配置为输出：** CCR2是装入当前捕获/比较2寄存器的值（预装载值）。 如果在TIMx_CCMR2寄存器(OC2PE位)中未选择预装载特性，其始终装入当前寄存器中。否则，只有当更新事件发生时，此预装载值才装入当前捕获/比较2寄存器中。 当前捕获/比较寄存器包含了与计数器TIMx_CNT比较的值，并且在OC端口上输出信号。 **若CC2通道配置为输入：** CCR2包含了由上一次输入捕获2事件（IC2）传输的计数器值。

13.5.15 捕获/比较寄存器 3(TIMx_CCR3)

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3的值 **若CC3通道配置为输出：** CCR3是装入当前捕获/比较3寄存器的值（预装载值）。 如果在TIMx_CCMR3寄存器(OC3PE位)中未选择预装载特性，其始终装入当前寄存器中。否则，只有当更新事件发生时，此预装载值才装入当前捕获/比较3寄存器中。 当前捕获/比较寄存器包含了与计数器TIMx_CNT比较的值，并且在OC端口上输出信号。 **若CC3通道配置为输入：** CCR3包含了由上一次输入捕获3事件（IC3）传输的计数器值。

13.5.16 捕获/比较寄存器 4(TIMx_CCR4)

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4的值 **若CC4通道配置为输出：** CCR4是了装入当前捕获/比较4寄存器的值（预装载值）。 如果在TIMx_CCMR4寄存器(OC4PE位)中未选择预装载特性，其始终装入当前寄存器中。否则，只有当更新事件发生时，此预装载值才装入当前捕获/比较4寄存器中。 当前捕获/比较寄存器包含了与计数器TIMx_CNT比较的值，并且在OC端口上输出信号。 **若CC4通道配置为输入：** CCR4包含了由上一次输入捕获4事件（IC4）传输的计数器值。

13.5.17 DMA控制寄存器(TIMx_DCR)

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
保留			DBL[4:0]					保留			DBA[4:0]				
			rw	rw	rw	rw	rw				rw	rw	rw	rw	rw

位15:13	保留，始终读为0。
位12:8	DBL[4:0]: DMA连续传送长度 这些位定义了DMA在连续模式下的传送长度（当对TIMx_DMAR寄存器的地址进行读或写时，定时器则进行一次连续传送），即：定义被传送的字节数目： 00000：1 字节　　00001：2 字节 00010：3 字节　　..... .....　　10001：18 字节
位7:5	保留，始终读为0。
位4:0	DBA[4:0]: DMA基地址 这些位定义了DMA在连续模式下的基地址（当对TIMx_DMAR寄存器的地址进行读或写时），DBA定义为从TIMx_CR1寄存器所在地址开始的偏移量： 00000：TIMx_CR1， 00001：TIMx_CR2， 00010：TIMx_SMCR， .....

13.5.18 连续模式的DMA地址(TIMx_DMAR)

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 对TIMx_DMAR寄存器的读或写会导致对以下地址的寄存器的存取操作： TIMx_CR1地址 + DBA + DMA指针，其中： “TIMx_CR1地址”是控制寄存器1的地址； “DBA”是TIMx_DCR寄存器中定义的基地址； “DMA指针”是由DMA自动控制的偏移量，它取决于TIMx_DCR寄存器中定义的DBL。

13.6 TIMx寄存器图

下表中将 TIMx 的所有寄存器映射到一个 16 位可寻址(编址)空间。

表41 TIMx – 寄存器图和复位值

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
000h	TIMx_CR1	保留																						CKD[1:0]		ARPE	CMS[1:0]		DIR	OPM	URS	UDIS	CEN
	复位值																							0	0	0	0	0	0	0	0	0	0
004h	TIMx_CR2	保留																								TI1S	MMS[2:0]			CCDS	保留		
	复位值																									0	0	0	0	0			
008h	TIMx_SMCR	保留																ETP	ECE	ETPS[1:0]		EFT[3:0]				MSM	TS[2:0]			保留	SMS[2:0]		
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0
00Ch	TIMx_DIER	保留																	TDE	保留	CC4DE	CC3DE	CC2DE	CC1DE	UDE	保留	TIE	保留	CC4IE	CC3IE	CC2IE	CC1IE	UIE
	复位值																		0		0	0	0	0	0		0		0	0	0	0	0
010h	TIMx_SR	保留																			CC4OF	CC3OF	CC2OF	CC1OF	保留		TIF	保留	CC4IF	CC3IF	CC2IF	CC1IF	UIF
	复位值																				0	0	0	0			0		0	0	0	0	0
014h	TIMx_EGR	保留																								保留	TG	保留	CC4G	CC3G	CC2G	CC1G	UG
	复位值																										0		0	0	0	0	0
018h	TIMx_CCMR1 输出比较模式	保留																	OC2CE	OC2M[2:0]			OC2PE	OC2FE	CC2S[1:0]		OC1CE	OC1M[2:0]			OC1PE	OC1FE	CC1S[1:0]	
	复位值																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	TIMx_CCMR1 输入捕获模式	保留																	IC2F[3:0]				IC2PSC[1:0]		CC2S[1:0]		IC1F[3:0]				IC1PSC[1:0]		CC1S[1:0]	
	复位值																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
01Ch	TIMx_CCMR2 输出比较模式	保留																	OC4CE	OC4M[2:0]			OC4PE	OC4FE	CC4S[1:0]		OC3CE	OC3M[2:0]			OC3PE	OC3FE	CC3S[1:0]	
	复位值																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	TIMx_CCMR2 输入捕获模式	保留																	IC4F[3:0]				IC4PSC[1:0]		CC4S[1:0]		IC3F[3:0]				IC3PSC[1:0]		CC3S[1:0]	
	复位值																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
020h	TIMx_CCER	保留																			CC4P	CC4E	保留		CC3P	CC3E	保留		CC2P	CC2E	保留		CC1P	CC1E
	复位值																				0	0			0	0			0	0			0	0
024h	TIMx_CNT	保留																CNT[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
028h	TIMx_PSC	保留																PSC[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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
02Ch	TIMx_ARR	保留																ARR[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
030h	保留																																
034h	TIMx_CCR1	保留																CCR1[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
038h	TIMx_CCR2	保留																CCR2[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
03Ch	TIMx_CCR3	保留																CCR3[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
040h	TIMx_CCR4	保留																CCR4[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
044h	保留																																
048h	TIMx_DCR	保留																保留			DBL[4:0]					保留			DBA[4:0]			
	复位值																				0	0	0	0	0				0	0	0	0	0
04Ch	TIMx_DMAR	保留																DMAB[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

14 控制器局域网(bxCAN)

14.1 简介

bxCAN 是基本扩展 CAN(Basic Extended CAN)的缩写，它支持 CAN 协议 2.0A 和 2.0B。它的设计目标是，以最小的 CPU 负荷来高效处理大量收到的报文。它也支持报文发送的优先级要求（优先级特性可软件配置）。

对于安全紧要的应用，bxCAN 提供所有支持时间触发通信模式所需的硬件功能。

14.2 主要特点

- 支持 CAN 协议 2.0A 和 2.0B 主动模式
- 波特率最高可达 1 兆位/秒
- 支持时间触发通信功能

发送

- 3 个发送邮箱
- 发送报文的优先级特性可软件配置
- 记录发送 SOF 时刻的时间戳

接收

- 3 级深度的 2 个接收 FIFO
- 14 个位宽可变的过滤器组－由整个 CAN 共享
- 标识符列表
- FIFO 溢出处理方式可配置
- 记录接收 SOF 时刻的时间戳

时间触发通信模式

- 禁止自动重传模式
- 16 位自由运行定时器
- 定时器分辨率可配置
- 可在最后 2 个数据字节发送时间戳

管理

- 中断可屏蔽

- 邮箱占用单独 1 块地址空间，便于提高软件效率

注： USB 和 CAN 共用一个专用的 512 字节的 SRAM 存储器用于数据的发送和接收，因此不同同时使用 USB 和 CAN(共享的 SRAM 被 USB 和 CAN 模块互斥地访问)。USB 和 CAN 可以同时用于一个应用中但不能在同一个时间使用。

14.3 总体描述

在当今的 CAN 应用中，CAN 网络的节点在不断增加，并且多个 CAN 常常通过网关连接起来，因此整个 CAN 网中的报文数量（每个节点都需要处理）急剧增加。除了应用层报文外，网络管理和诊断报文也被引入。

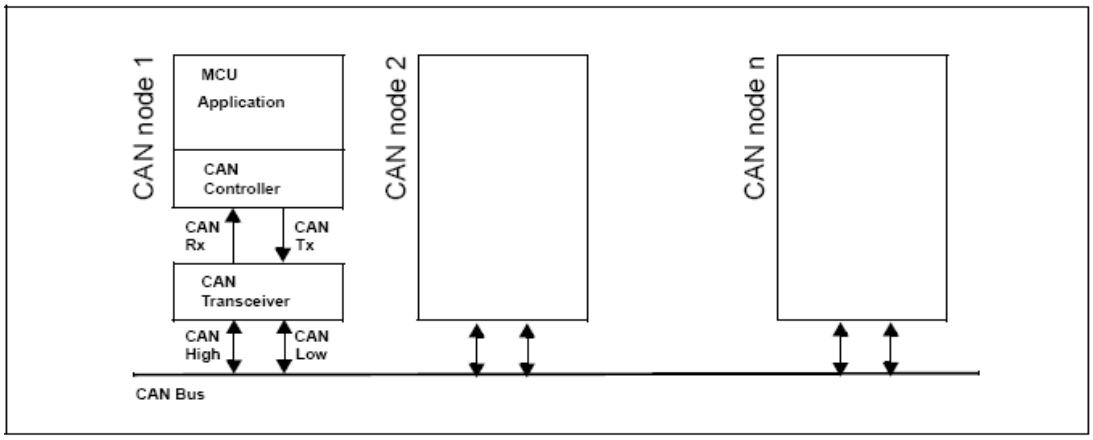
- 需要一个增强的过滤机制来处理各种类型的报文

此外，应用层任务需要更多 CPU 时间，因此报文接收所需的实时响应程度需要减轻。

- 接收 FIFO 的方案允许，CPU 花很长时间处理应用层任务而不会丢失报文。

构筑在底层 CAN 驱动程序上的高层协议软件，要求跟 CAN 控制器之间有高效的接口。

图119 CAN 网拓扑结构



14.3.1 CAN 2.0B内核

bxCAN 模块可以完全自动地接收和发送 CAN 报文；且完全支持标准标识符（11 位）和扩展标识符（29 位）。

14.3.2 控制、状态和配置寄存器

应用程序通过这些寄存器，可以：

- 配置 CAN 参数，如波特率
- 请求发送报文

- 处理报文接收
- 管理中断
- 获取诊断信息

14.3.3 发送邮箱

共有 3 个发送邮箱供软件来发送报文。发送调度器根据优先级决定哪个邮箱的报文先被发送。

14.3.4 接收过滤器

共有 14 个位宽可变/可配置的标识符过滤器组，软件通过对它们编程，从而在引脚收到的报文中选择它需要的报文，而把其它报文丢弃掉。

14.3.5 接收FIFO

共有 2 个接收 FIFO，每个 FIFO 都可以存放 3 个完整的报文。它们完全由硬件来管理。

图120 CAN 功能框图

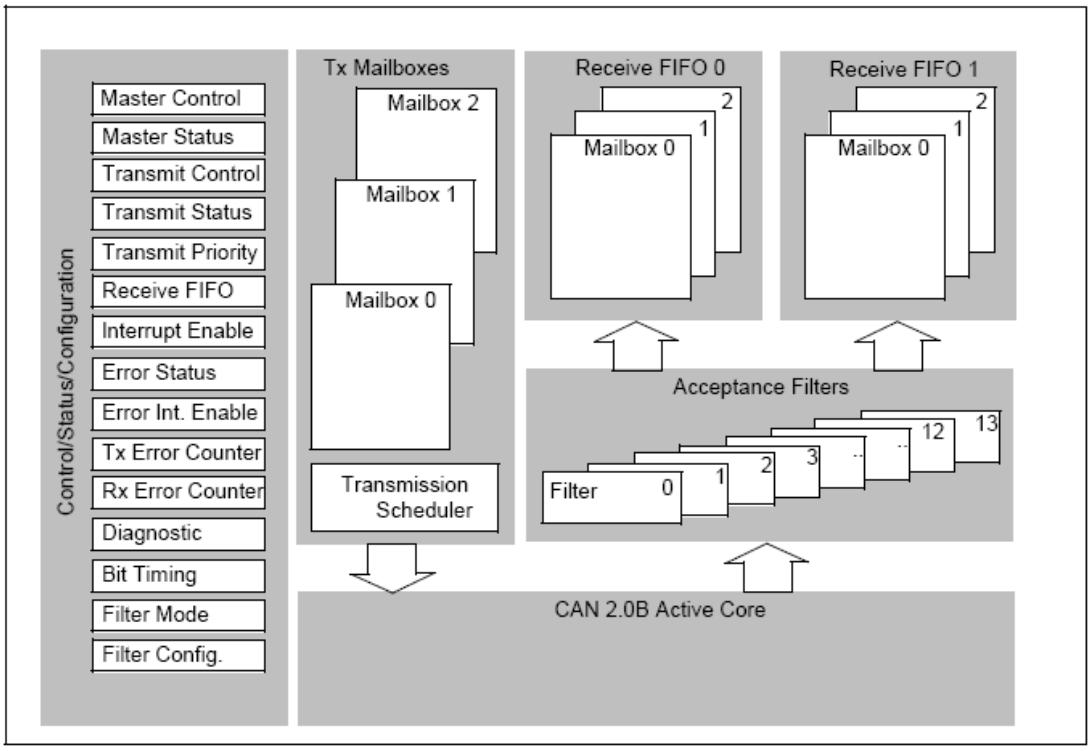
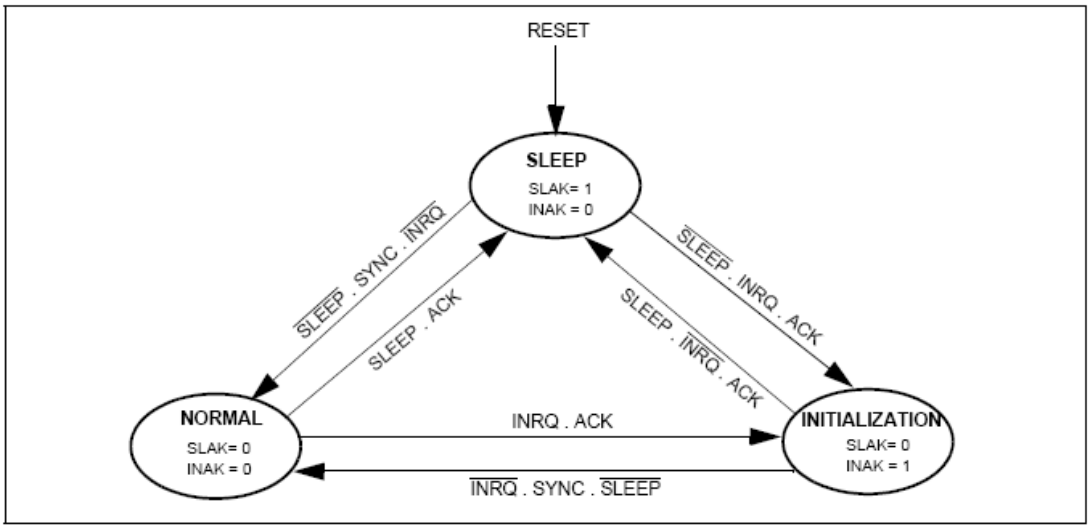


图121 bxCAN 工作模式



注： 1 ACK = 硬件响应睡眠或初始化请求,而对 CAN_MSR 寄存器的 INAK 或 SLAK 位置 1 的状态
2 SYNC = bxCAN 等待 CAN 总线变为空闲的状态，即在 CANRX 引脚上检测到连续的 11 个隐性位

14.4 工作模式

bxCAN 有 3 个主要的工作模式：**初始化**、**正常**和**睡眠模式**。在硬件复位后，bxCAN 工作在睡眠模式以节省电能，同时 CANTX 引脚的内部上拉电阻被激活。软件通过对 CAN_MCR 寄存器的 INRQ 或 SLEEP 位置 1，可以请求 bxCAN 进入**初始化**或**睡眠模式**。一旦进入了**初始化**或**睡眠模式**，bxCAN 就对 CAN_MSR 寄存器的 INAK 或 SLAK 位置 1 来进行确认，同时内部上拉电阻被禁用。当 INAK 和 SLAK 位都为 0 时，bxCAN 就处于**正常模式**。在进入**正常模式**前，bxCAN 必须跟 CAN 总线取得**同步**；为取得同步，bxCAN 要等待 CAN 总线达到空闲状态，即在 CANRX 引脚上监测到 11 个连续的隐性位。

14.4.1 初始化模式

通常软件初始化应该在硬件处于初始化模式时进行。软件通过对 CAN_MCR 寄存器的 INRQ 位置 1，来请求 bxCAN 进入初始化模式，然后等待硬件对 CAN_MSR 寄存器的 INAK 位置 1 来进行确认。

软件通过对 CAN_MCR 寄存器的 INRQ 位清 0，来请求 bxCAN 退出初始化模式，当硬件对 CAN_MSR 寄存器的 INAK 位清 0 就确认了初始化模式的退出。

当 bxCAN 处于初始化模式时，报文的接收和发送都被禁止，并且 CANTX 引脚输出隐性位（高电平）。

初始化模式的进入，不会改变配置寄存器。

软件对 bxCAN 的初始化，至少包括位时间特性(CAN_BTR)和控制(CAN_MCR)这 2 个寄存器。

在对 bxCAN 的过滤器组（模式、位宽、FIFO 关联、激活和过滤器值）进行初始化前，软件要对 CAN_FMR 寄存器的 FINIT 位设置 1。对过滤器的初始化可以在非初始化模式下进行。

注： 当 FINIT=1 时，报文的接收被禁止。
可以先对过滤器激活位清 0（在 CAN_FA0R 中），然后修改相应过滤器的值。
如果过滤器组没有使用，那么就应该让它处于非激活状态（保持其 FACT 位为清 0 状态）。

14.4.2 正常模式

在初始化完成后，软件应该让硬件进入正常模式，以便正常接收和发送报文。软件可以通过对 CAN_MCR 寄存器的 INRQ 位清 0，来请求从初始化模式进入正常模式，然后要等待硬件对 CAN_MSR 寄存器的 INAK 位置 1 的确认。在跟 CAN 总线取得同步，即在 CANRX 引脚上监测到 11 个连续的隐性位（等效于总线空闲）后，bxCAN 才能正常接收和发送报文。

过滤器初值的设置不需要在初始化模式下进行，但必须在它处在非激活状态下完成（相应的 FACT 位为 0）。而过滤器的位宽和模式的设置，则必须在初始化模式下，进入正常模式前完成。

14.4.3 睡眠模式（低功耗）

bxCAN 可工作在低功耗的睡眠模式。软件通过对 CAN_MCR 寄存器的 SLEEP 位置 1，来请求进入这一模式。在该模式下，bxCAN 的时钟停止了，但软件仍然可以访问邮箱寄存器。

当 bxCAN 处于睡眠模式，软件必须对 CAN_MCR 寄存器的 INRQ 位置 1 并且同时对 SLEEP 位清 0，才能进入初始化模式。

有 2 种方式可以唤醒（退出睡眠模式）bxCAN：通过软件对 SLEEP 位清 0，或硬件检测到 CAN 总线的活动。

如果 CAN_MCR 寄存器的 AWUM 位为 1，一旦检测到 CAN 总线的活动，硬件就自动对 SLEEP 位清 0 来唤醒 bxCAN。如果 CAN_MCR 寄存器的 AWUM 位为 0，软件必须在唤醒中断里对 SLEEP 位清 0 才能退出睡眠状态。

注： 如果唤醒中断被允许（CAN_IER 寄存器的 WKUIE 位为 1），那么一旦检测到 CAN 总线活动就会产生唤醒中断，而不管硬件是否会自动唤醒 bxCAN。

在对SLEEP位清 0 后，睡眠模式的退出必须与CAN总线同步，请参考 0：

bxCAN 工作模式。当硬件对 SLAK 位清 0 时，就确认了睡眠模式的退出。

14.4.4 测试模式

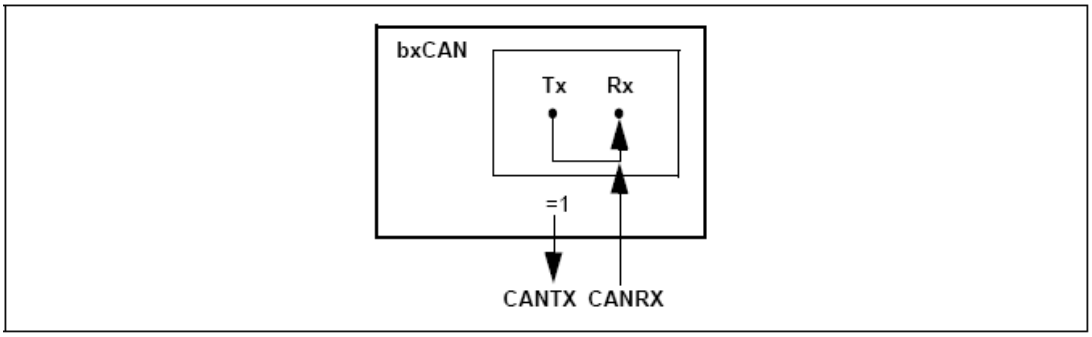
通过对 CAN_BTR 寄存器的 SILM 和/或 LBKM 位置 1，来选择一种测试模式。只能在初始化模式下，修改这 2 位。在选择了一种测试模式后，软件需要对 CAN_MCR 寄存器的 INRQ 位清 0，来真正进入测试模式。

14.4.5 静默模式

通过对 CAN_BTR 寄存器的 SILM 位置 1，来选择静默模式。

在静默模式下，bxCAN 可以正常地接收数据帧和远程帧，但只能发出隐性位，而不能真正发送报文。如果 bxCAN 需要发出显性位（确认位、过载标志、主动错误标志），那么这样的显性位在内部被接回来从而可以被 CAN 内核检测到，同时 CAN 总线不会受到影响而仍然维持在隐性位状态。因此，静默模式通常用于分析 CAN 总线的活动，而不会对总线造成影响－显性位（确认位、错误帧）不会真正发送到总线上。

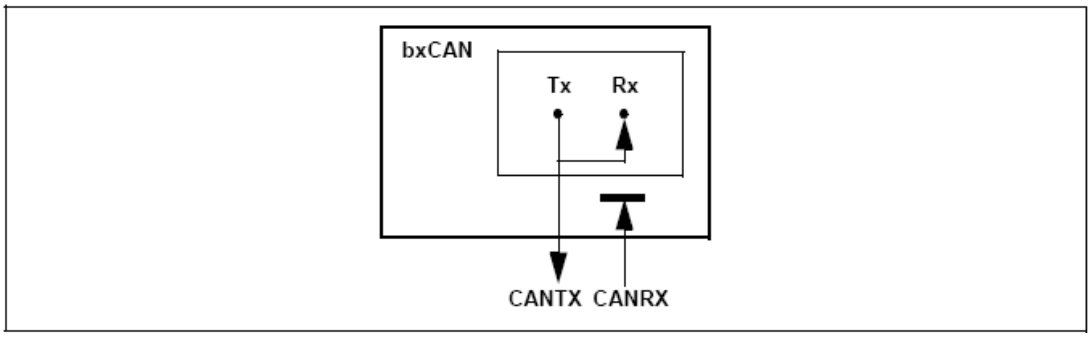
图122 bxCAN 工作在静默模式



14.4.6 环回模式

通过对 CAN_BTR 寄存器的 LBKM 位置 1，来选择环回模式。在环回模式下，bxCAN 把发送的报文当作接收的报文并保存（如果可以通过接收过滤）在接收邮箱里。

图123 bxCAN 工作在环回模式



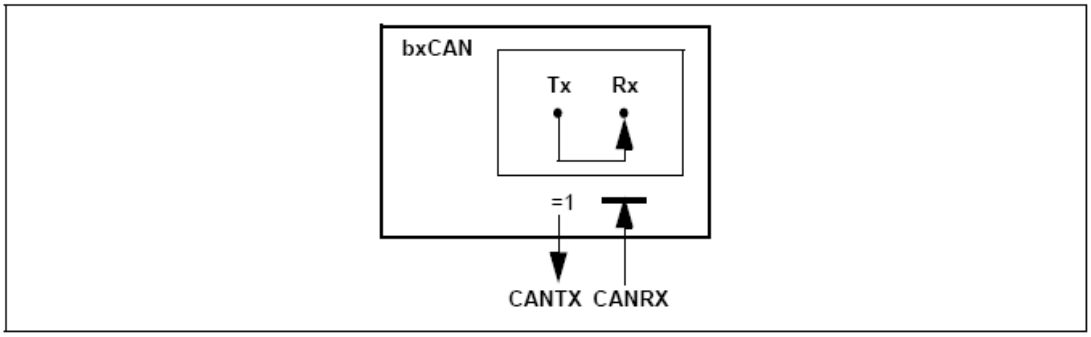
环回模式可用于自测试。为了避免外部的影响，在环回模式下 CAN 内核忽略确认错误（在数据/远程帧的确认位时刻，不检测是否有显性位）。在环回模式下，

bxCAN 在内部把 Tx 输出回馈到 Rx 输入上，而完全忽略 CANRX 引脚的实际状态。发送的报文可以在 CANTX 引脚上检测到。

14.4.7 环回静默模式

通过对 CAN_BTR 寄存器的 LBKM 和 SILM 位同时置 1，可以选择环回静默模式。该模式可用于“热自测试”，即可以象环回模式那样测试 bxCAN，但却不会影响 CANTX 和 CANRX 所连接的整个 CAN 系统。在环回静默模式下，CANRX 引脚与 CAN 总线断开，同时 CANTX 引脚被驱动到隐性位状态。

图124 bxCAN 工作在环回静默模式



14.5 功能描述

14.5.1 发送处理

发送报文的流程为：应用程序选择 1 个**空**发送邮箱；设置标识符，数据长度和待发送数据；然后对 CAN_TIxR 寄存器的 TXRQ 位置 1，来请求发送。TXRQ 位置 1 后，邮箱就不再是空邮箱；而一旦邮箱不再为**空**，软件对邮箱寄存器就不再有写的权限。TXRQ 位置 1 后，邮箱马上进入**挂号**状态，并等待成为最高优先级的邮箱，参见*发送优先级*。一旦邮箱成为最高优先级的邮箱，其状态就变为**预定**发送状态。一旦 CAN 总线进入空闲状态，预定发送邮箱中的报文就马上被发送（进入**发送**状态）。一旦邮箱中的报文被成功发送后，它马上变为**空**邮箱；硬件相应地对 CAN_TSR 寄存器的 RQCP 和 TXOK 位置 1，来表明一次成功发送。

如果发送失败，由于仲裁引起的就对 CAN_TSR 寄存器的 ALST 位置 1，由于发送错误引起的就对 TERR 位置 1。

发送优先级

由标识符决定

当有超过 1 个发送邮箱在挂号时，发送顺序由邮箱中报文的标识符决定。根据 CAN 协议，标识符数值最低的报文具有最高的优先级。如果标识符的值相等，那么邮箱号小的报文先被发送。

由发送请求次序决定

通过对 CAN_MCR 寄存器的 TXFP 位置 1，可以把发送邮箱配置为发送 FIFO。在该模式下，发送的优先级由发送请求次序决定。

该模式对分段发送很有用。

中止

通过对 CAN_TSR 寄存器的 ABRQ 位置 1，可以中止发送请求。邮箱如果处于**挂号**或**预定**状态，发送请求马上就被中止了。如果邮箱处于**发送**状态，那么中止请求可能导致 2 种结果。如果邮箱中的报文被成功发送，那么邮箱变为**空**邮箱，并且 CAN_TSR 寄存器的 TXOK 位被硬件置 1。如果邮箱中的报文发送失败了，那么邮箱变为**预定**状态，然后发送请求被中止，邮箱变为**空**邮箱且 TXOK 位被硬件清 0。因此如果邮箱处于**发送**状态，那么在发送操作结束后，邮箱都会变为**空**邮箱。

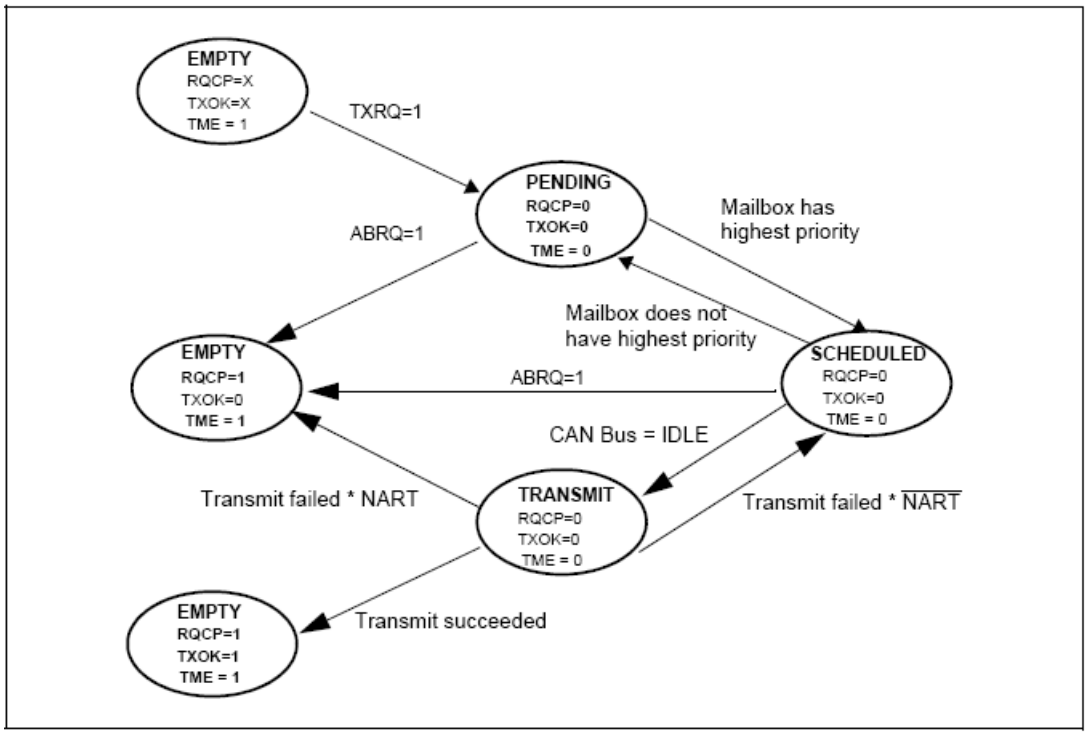
禁止自动重传模式

该模式主要用于满足 CAN 标准中，时间触发通信选项的需求。通过对 CAN_MCR 寄存器的 NART 位置 1，来让硬件工作在该模式。

在该模式下，发送操作只会执行一次。如果发送操作失败了，不管是由于仲裁丢失或出错，硬件都不会再自动发送该报文。

在一次发送操作结束后，硬件认为发送请求已经完成，从而对 CAN_TSR 寄存器的 RQCP 位置 1，同时发送的结果反映在 TXOK、ALST 和 TERR 位上。

图125 发送邮箱状态



14.5.2 时间触发通信模式

在该模式下，CAN 硬件的内部定时器被激活，并且被用于产生时间戳，分别存储在 CAN_RDTxR/CAN_TDTxR 寄存器中。内部定时器在接收和发送的帧起始位的采样点位置被采样，并生成时间戳。

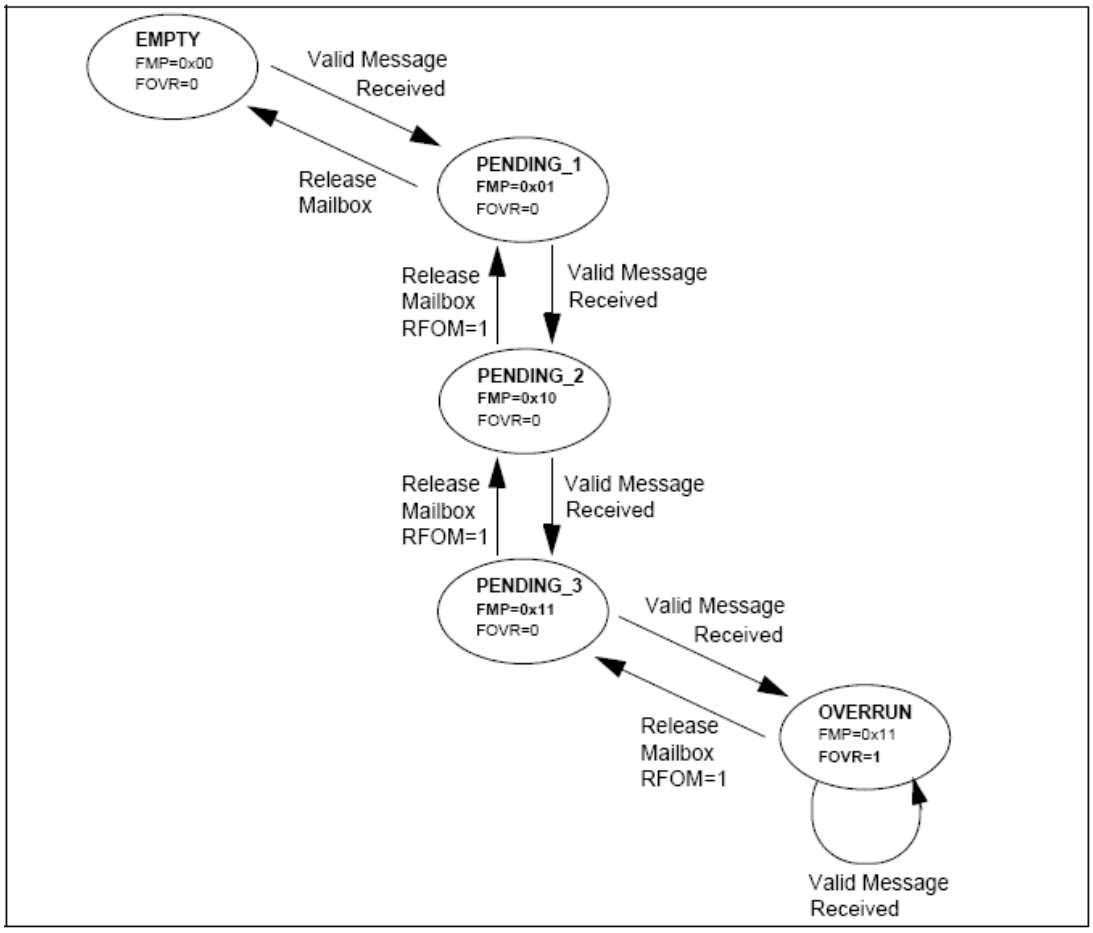
14.5.3 接收管理

接收到的报文，被存储在 3 级邮箱深度的 FIFO 中。FIFO 完全由硬件来管理，从而节省了 CPU 的处理负荷，简化了软件并保证了数据的一致性。应用程序只能通过读取 FIFO 输出邮箱，来读取 FIFO 中最先收到的报文。

有效报文

根据CAN协议，当报文被正确接收（直到EOF域的最后 1 位都没有错误），且通过了标识符过滤，那么该报文被认为是有效报文。请参考 14.5.4：标识符过滤。

图126 接收 FIFO 状态



FIFO管理

FIFO 从**空**状态开始，在接收到第 1 个有效的报文后，FIFO 状态变为**挂号_1**（pending_1），硬件相应地把 FMP[1:0]设置为 01（二进制 01b）。软件可以读取 FIFO 输出邮箱来读出邮箱中的报文，然后通过对 CRFR 寄存器的 RFOM 位设置 1 来释放邮箱，这样 FIFO 又变为**空**状态了。如果在释放邮箱的同时，又收到了 1 个有效的报文，那么 FIFO 仍然保留在**挂号_1** 状态，软件可以读取 FIFO 输出邮箱来读出新收到的报文。

如果应用程序不释放邮箱，在接收到下 1 个有效的报文后，FIFO 状态变为**挂号_2**（pending_2），硬件相应地把 FMP[1:0]设置为 10（二进制 10b）。重复上面的过程，第 3 个有效的报文把 FIFO 变为**挂号_3** 状态（FMP[1:0]＝11b）。此时，软件必须对 RFOM 位设置 1 来释放邮箱，以便 FIFO 可以有空间来存放下 1 个有效的报文；否则，下 1 个有效的报文到来时就会导致 1 个报文的丢失。

参见 14.5.5：报文存储

溢出

当 FIFO 处于**挂号_3** 状态（即 FIFO 的 3 个邮箱都是满的），下 1 个有效的报文就会导致**溢出**，并且 1 个报文会丢失。此时，硬件对 CAN_RFxR 寄存器的 FOVR 位进行置 1 来表明溢出情况。至于哪个报文会被丢弃，取决于对 FIFO 的设置：

- 如果禁用了FIFO锁定功能（CAN_MCR寄存器的RFLM位被清0），那么FIFO中最后收到的报文就被新报文所覆盖。这样，最新收到的报文不会被丢弃掉。
- 如果启用了FIFO锁定功能（CAN_MCR寄存器的RFLM位被置1），那么新收到的报文就被丢弃，软件可以读到FIFO中最早收到的3个报文。

接收相关的中断

一旦往 FIFO 存入 1 个报文，硬件就会更新 FMP[1:0]位，并且如果 CAN_IER 寄存器的 FMPIE 位为 1，那么就会产生一个中断请求。

当 FIFO 变满时（即第 3 个报文被存入），CAN_RFxR 寄存器的 FULL 位就被置 1，并且如果 CAN_IER 寄存器的 FFIE 位为 1，那么就会产生一个满中断请求。

在溢出的情况下，FOVR 位被置 1，并且如果 CAN_IER 寄存器的 FOVIE 位为 1，那么就会产生一个溢出中断请求。

14.5.4 标识符过滤

在 CAN 协议里，报文的标识符不代表节点的地址，而是跟报文的内容相关的。因此，发送者乙广播的形式把报文发送给所有的接收者。节点在接收报文时－根据标识符的值－决定软件是否需要该报文；如果需要，就拷贝到 SRAM 里；如果不需要，报文就被丢弃且无需软件的干预。

为满足这一需求，bxCAN 为应用程序提供了 14 个位宽可变的、可配置的过滤器组（13~0），以便只接收那些软件需要的报文。硬件过滤的做法节省了 CPU 开销，否则就必须由软件过滤从而占用一定的 CPU 开销。每个过滤器组 x 由 2 个 32 位寄存器，CAN_FxR0 和 CAN_FxR1 组成。

可变的位宽

每个过滤器组的位宽都可以独立配置，以满足应用程序的不同需求。根据位宽的不同，每个过滤器组可提供：

- 1 个 32 位过滤器，包括：STDID[10：0]、EXTID[17：0]、IDE 和 RTR 位
- 2 个 16 位过滤器，包括：STDID[10：0]、IDE、RTR 和 EXTID[17：15]位

可参见 图 127。
此外过滤器可配置为，屏蔽位模式和标识符列表模式。

屏蔽位模式

在屏蔽位模式下，标识符寄存器和屏蔽寄存器一起，指定报文标识符的任何一位，应该按照“必须匹配”或“不用关心”处理。

标识符列表模式

在标识符列表模式下，屏蔽寄存器也被当作标识符寄存器用。因此，不是采用 1 个标识符加 1 个屏蔽位的方式，而是使用 2 个标识符寄存器。接收报文标识符的每一位都必须跟过滤器标识符相同。

过滤器组位宽和模式的设置

在配置一个过滤器组前，必须把它设置为禁用状态，这可以通过清 0 CAN_FA0R 寄存器的相应FACT位来实现。通过设置CAN_FS0R的相应FSCx位，可以配置一个过滤器组的位宽，请参见 图 127。通过设置CAN_FM0R的FBMx位，可以配置过滤器组为**标识符列表**模式或**屏蔽位**模式。
为了过滤出一组标识符，应该设置过滤器组工作在屏蔽位模式。

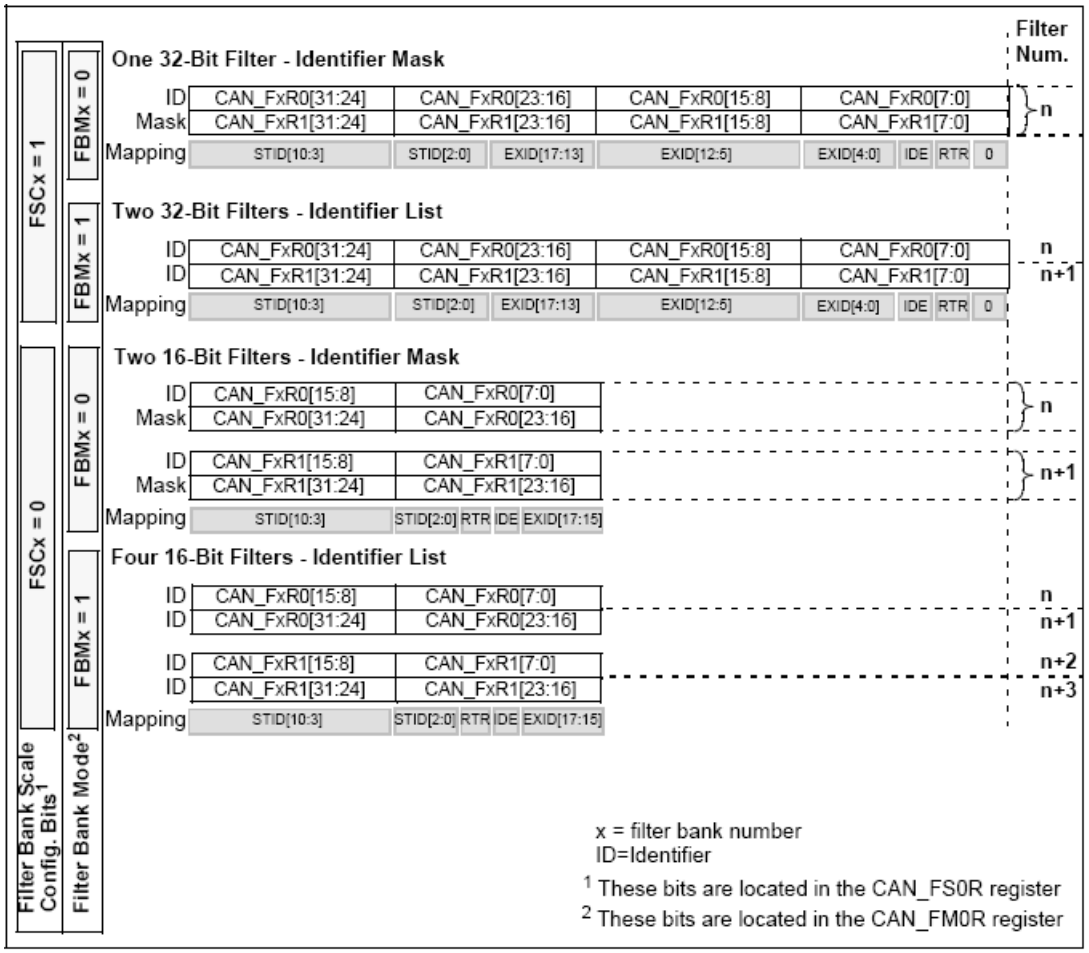
为了过滤出一个标识符，应该设置过滤器组工作在标识符列表模式。

应用程序不用的过滤器组，应该保持在禁用状态。

过滤器组中的每个过滤器，都被编号为（叫做过滤器号）从 0 开始，到某个最大数值－取决于 14 个过滤器组的模式和位宽的设置。

关于过滤器配置，可参见 图 127。

图127 过滤器组位宽设置－寄存器组织



过滤器匹配序号

一旦收到的报文被存入 FIFO，就可被应用程序访问。通常情况下，报文中的数据被拷贝到 SRAM 中；为了把数据拷贝到合适的位置，应用程序需要根据报文的标识符来辨别不同的数据。bxCAN 提供了过滤器匹配序号，以简化这一辨别过程。

根据过滤器优先级规则，过滤器匹配序号和报文一起，被存入邮箱中。因此每个收到的报文，都有与它相关联的过滤器匹配序号。

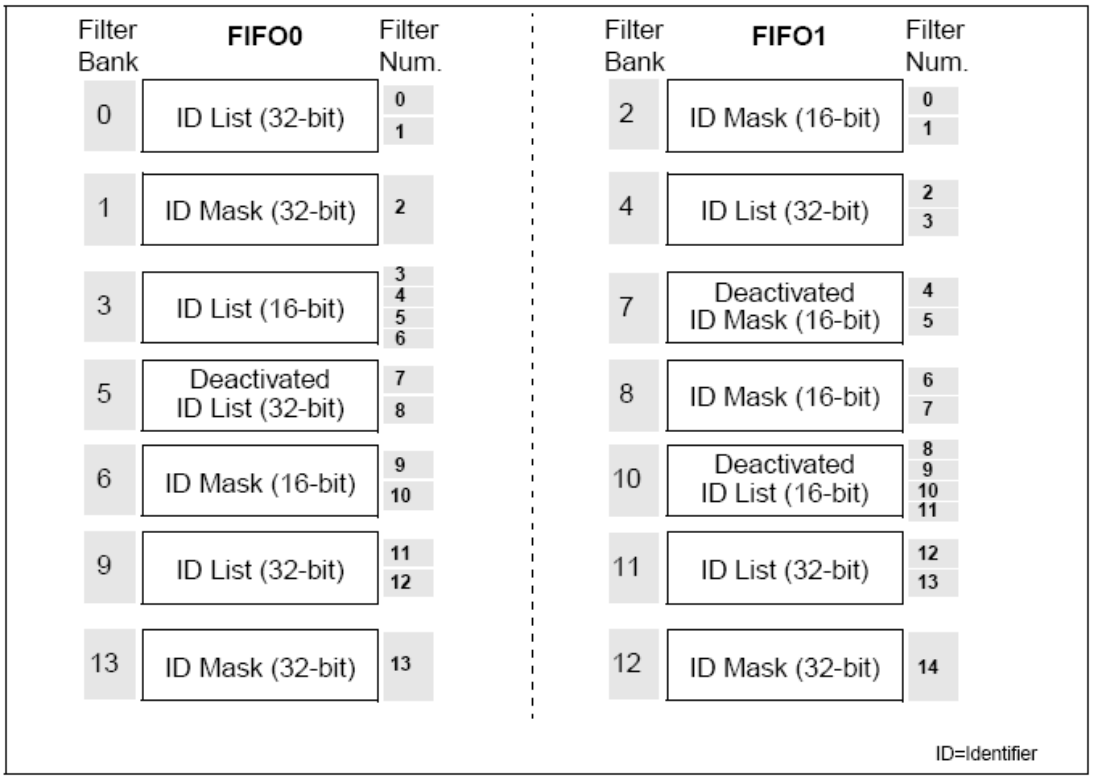
过滤器匹配序号可以通过下面两种方式来使用：

- 把过滤器匹配序号跟一系列所期望的值进行比较
- 把过滤器匹配序号当作一个索引来访问目标地址

对于标识符列表模式下的过滤器(非屏蔽方式的过滤器)，软件不需要直接跟标识符进行比较。

对于屏蔽位模式下的过滤器，软件只需要跟关心的那些屏蔽位（必须匹配的位）进行比较即可。
在给过滤器编号时，并不考虑过滤器组是否为激活状态。另外，每个FIFO各自对其关联的过滤器进行编号。请参考 图 128 的例子。

图128 过滤器编号的例子

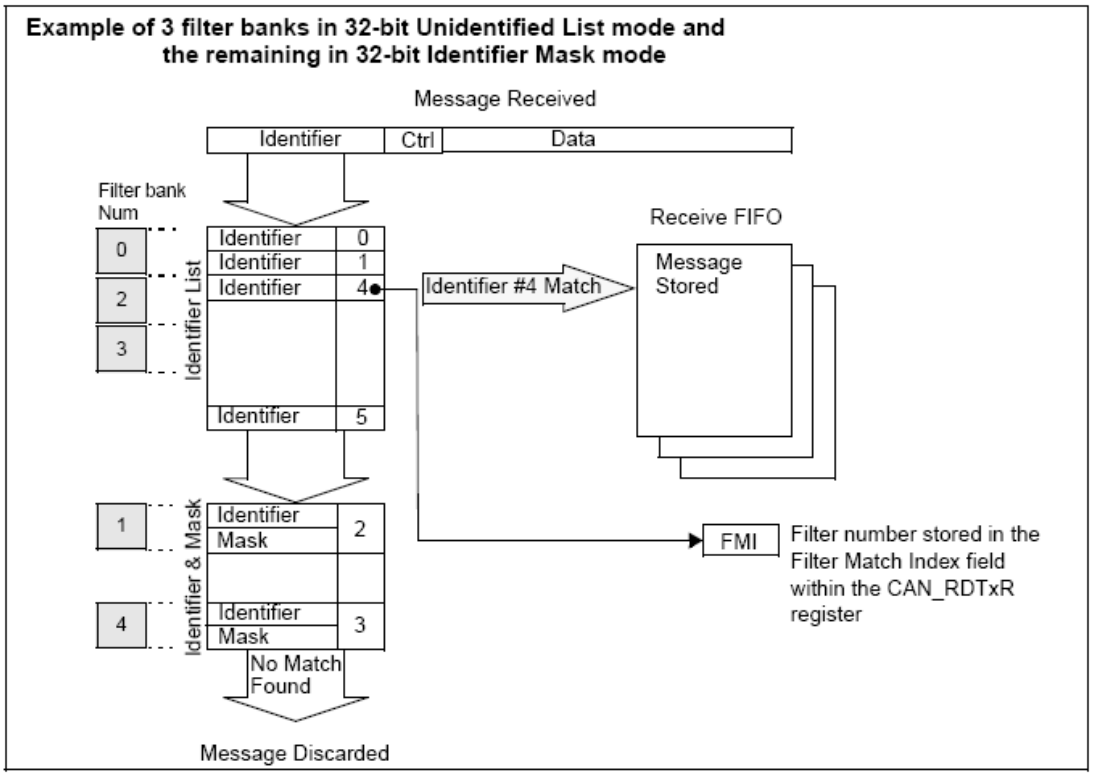


过滤器优先级规则

根据过滤器的不同配置，有可能一个报文标识符能通过多个过滤器的过滤；在这种情况下，存放在接收邮箱中的过滤器匹配序号，根据下列优先级规则来确定：

- 位宽为 32 位的过滤器，优先级高于位宽为 16 位的过滤器
- 对于位宽相同的过滤器，标识符列表模式的优先级高于屏蔽位模式
- 位宽和模式都相同的过滤器，优先级由过滤器号决定，过滤器号小的优先级高

图129 过滤器机制的例子



上面的例子说明了 bxCAN 的过滤器规则：在接收一个报文时，其标识符首先与配置在标识符列表模式下的过滤器相比较；如果匹配上，报文就被存放到相关联的 FIFO 中，并且所匹配的过滤器的序号被存入过滤器匹配序号中。如同例子中所显示，报文标识符跟#4 标识符匹配，因此报文内容和 FMI4 被存入 FIFO。

如果没有匹配，报文标识符接着与配置在屏蔽位模式下的过滤器进行比较。

如果报文标识符没有跟过滤器中的任何标识符相匹配，那么硬件就丢弃该报文，且不会对软件有任何打扰。

14.5.5 报文存储

邮箱是软件和硬件之间关于报文的接口。邮箱包含了所有跟报文有关的信息：标识符、数据、控制、状态和时间戳信息。

发送邮箱

软件需要在一个空的发送邮箱中，把待发送报文的各种信息设置好（然后再发出发送的请求）。发送的状态可通过查询 CAN_TSR 寄存器获知。

表42 发送邮箱寄存器列表

相对发送邮箱基地址的偏移量	寄存器名
0	CAN_TIxR
4	CAN_TDTxR
8	CAN_TDLxR

12	CAN_TDHxR

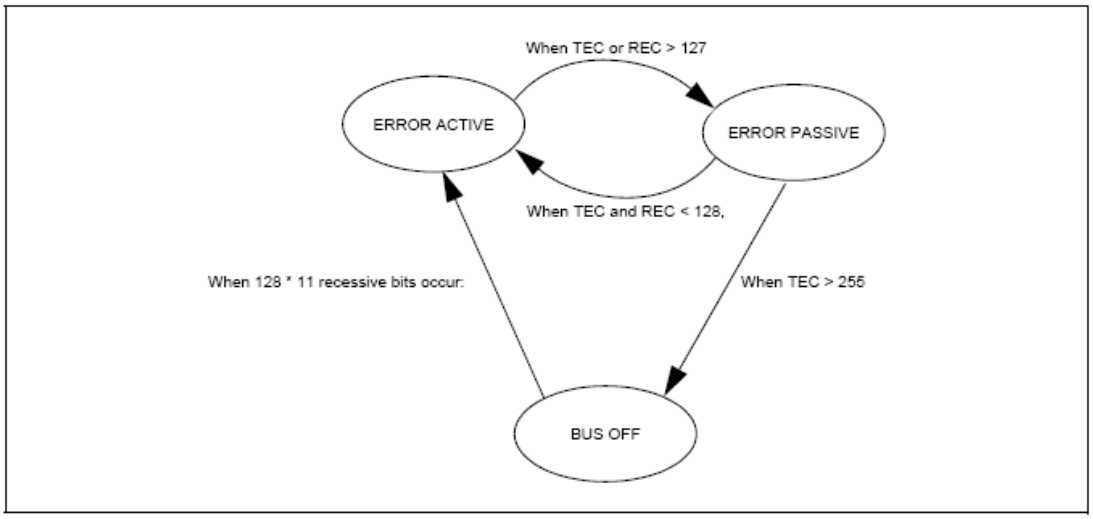
接收邮箱（FIFO）

在接收到一个报文后，软件就可以访问接收 FIFO 的输出邮箱来读取它。一旦软件处理了报文（如把它读出来），软件就应该对 CAN_RFxR 寄存器的 RFOM 位进行置 1，来释放该报文，以便为后面收到的报文留出存储空间。过滤器匹配序号存放在 CAN_RDTxR 寄存器的 FMI 域中。16 位的时间戳存放在 CAN_RDTxR 寄存器的 TIME[15:0]域中。

表43 接收邮箱寄存器列表

相对接收邮箱基地址的偏移量	寄存器名
0	CAN_RIxR
4	CAN_RDTxR
8	CAN_RDLxR
12	CAN_RDHxR

图130 CAN 错误状态图



14.5.6 出错管理

CAN 协议描述的出错管理，完全由硬件通过发送错误计数器（CAN_ESR 寄存器里的 TEC 域），和接收错误计数器（CAN_ESR 寄存器里的 REC 域）来实现，其值根据错误的情况而增加或减少。关于 TEC 和 REC 管理的详细信息，请参考 CAN 标准。

软件可以读出它们的值来判断 CAN 网络的稳定性。
此外，CAN_ESR 寄存器提供了当前错误状态的详细信息。通过设置 CAN_IER 寄存器（比如 ERRIE 位），软件可以灵活地控制中断的产生——当检测到出错时。

离线恢复

当 TEC 对于 255 时，bxCAN 就进入离线状态，同时 CAN_ESR 寄存器的 BOFF 位被置 1。在离线状态下，bxCAN 无法接收和发送报文。

根据 CAN_MCR 寄存器的 ABOM 位的设置，bxCAN 可以自动或在软件的请求下，从离线状态恢复（变为错误主动状态）。在这两种情况下，bxCAN 都必须等待一个 CAN 标准所描述的恢复过程（CAN RX 引脚上检测到 128 次 11 个连续的隐性位）。

如果 ABOM 位为 1，bxCAN 进入离线状态后，就自动开启恢复过程。

如果 ABOM 位为 0，软件必须先请求 bxCAN 进入然后再退出初始化模式，随后恢复过程才被开启。

*注： 在初始化模式下，bxCAN 不会监视 CAN RX 引脚的状态，这样就不能完成恢复过程。**为了完成恢复过程，bxCAN 必须工作在正常模式**。*

14.5.7 位时间特性

位时间特性逻辑通过采样来监视串行的 CAN 总线，并且通过跟帧起始位的边沿进行同步，及通过跟后面的边沿进行重新同步，来调整其采样点。

它的操作可以简单解释为，如下所述把名义上的每位的时间分为 3 段：

- **同步段(SYNC_SEG)**：通常期望位的变化发生在该时间段内。其值固定为 1 个时间单元（1 x t_{CAN}）。
- **时间段 1(BS1)**：定义采样点的位置。它包含 CAN 标准里的 PROP_SEG 和 PHASE_SEG1。其值可以编程为 1 到 16 个时间单元，但也可以被自动延长，以补偿因为网络中不同节点的频率差异所造成的相位的正向漂移。
- **时间段 2(BS2)**：定义发送点的位置。它代表 CAN 标准里的 PHASE_SEG2。其值可以编程为 1 到 8 个时间单元，但也可以被自动缩短以补偿相位的负向漂移。

重新同步跳跃宽度(SJW)定义了，在每位中可以延长或缩短多少个时间单元的上限。其值可以编程为 1 到 4 个时间单元。

有效跳变被定义为，当 bxCAN 自己没有发送隐性位时，从显性位到隐性位的第 1 次转变。

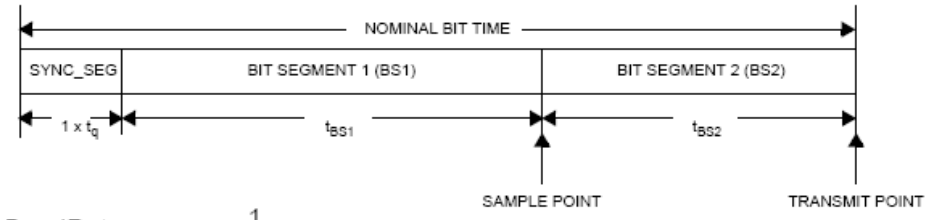
如果在时间段 1(BS1)而不是在同步段(SYNC_SEG)检测到有效跳变，那么 BS1 的时间就被延长最多 SJW 那么长，从而采样点被延迟了。

相反如果在时间段 2(BS2)而不是在 SYNC_SEG 检测到有效跳变，那么 BS2 的时间就被缩短最多 SJW 那么长，从而采样点被提前了。

为了避免软件的编程错误，对位时间特性寄存器(CAN_BTR)的设置，只能在 bxCAN 处于初始化状态下进行。

注： 关于 CAN 位时间特性和重同步机制的详细信息，请参考 ISO11898 标准。

图131 位时间特性



$$BaudRate = \frac{1}{NominalBitTime}$$

$$NominalBitTime = 1 \times t_q + t_{BS1} + t_{BS2}$$

with:

$t_{BS1} = t_q \times (TS1[3:0] + 1)$,

$t_{BS2} = t_q \times (TS2[2:0] + 1)$,

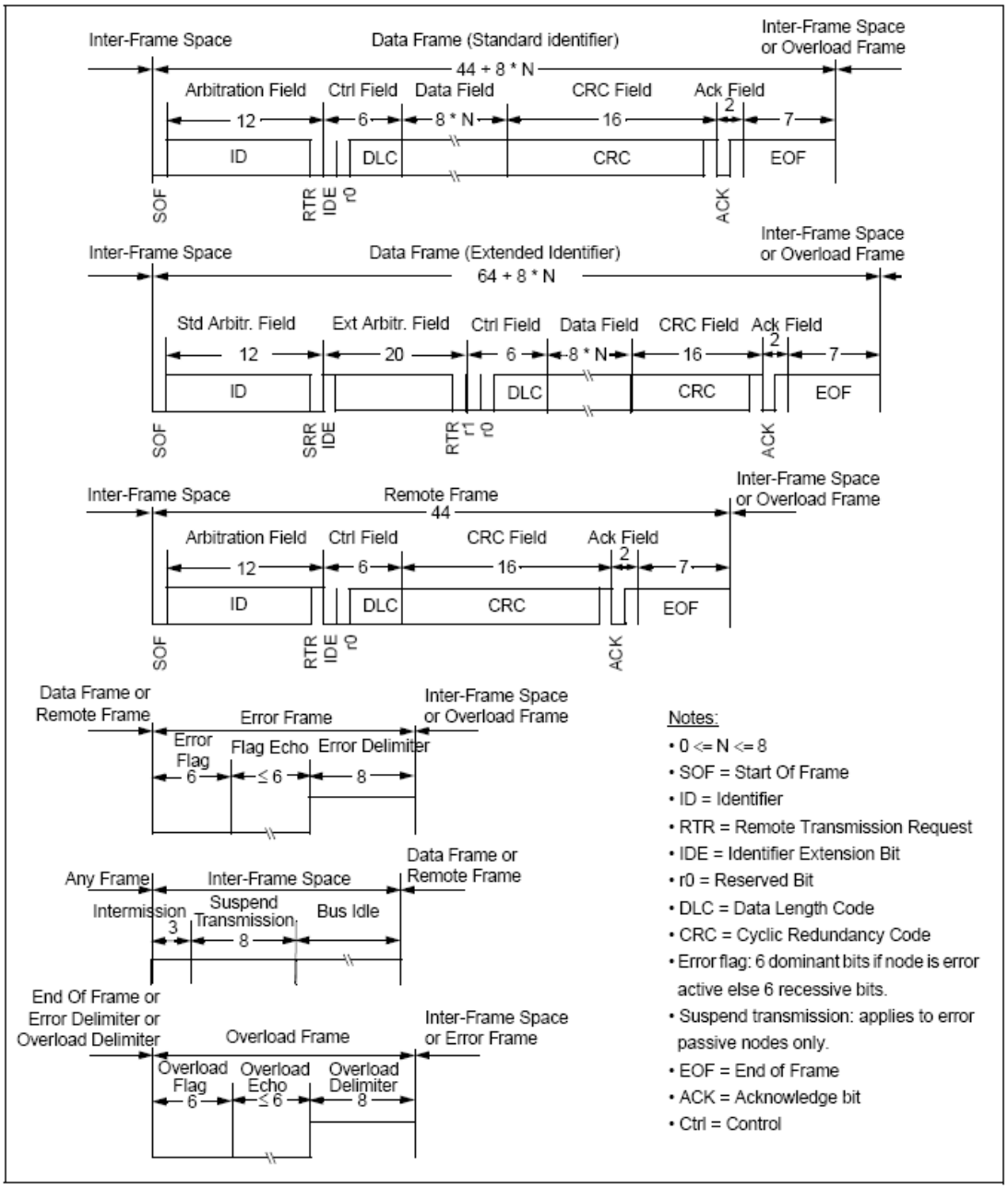
$t_q = (BRP[9:0] + 1) \times t_{PCLK}$

where t_q refers to the Time quantum

t_{PCLK} = time period of the APB clock,

BRP[9:0], TS1[3:0] and TS2[2:0] are defined in the CAN_BTR Register.

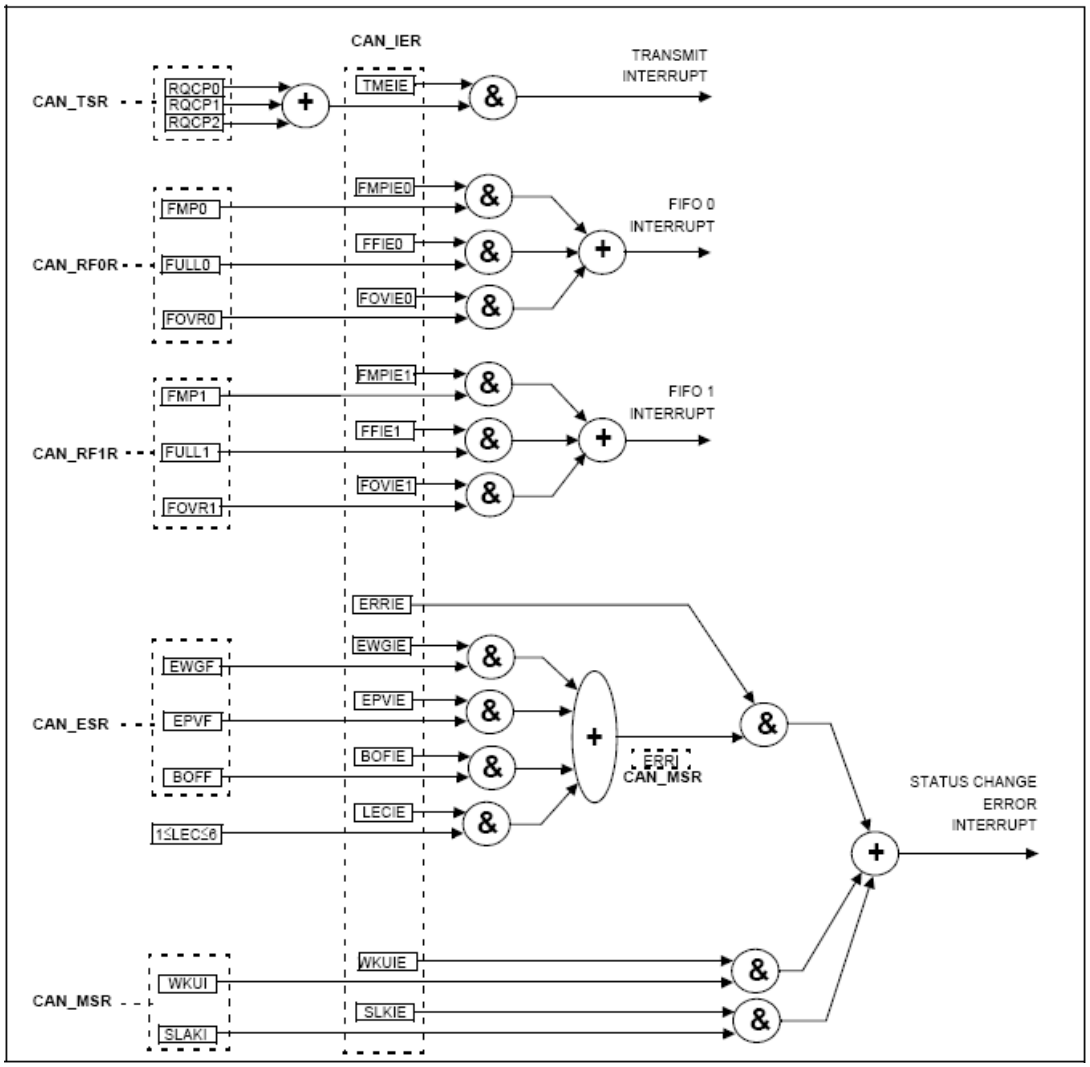
图132 各种 CAN 帧


Inter-Frame Space
Data Frame (Standard identifier)
Inter-Frame Space or Overload Frame
44 + 8 * N
Arbitration Field
Ctrl Field
Data Field
CRC Field
Ack Field
12
6
8 * N
16
2
7
ID
DLC
CRC
EOF
SOF
RTR
IDE
r0
ACK
Inter-Frame Space
Data Frame (Extended Identifier)
Inter-Frame Space or Overload Frame
64 + 8 * N
Std Arbitr. Field
Ext Arbitr. Field
Ctrl Field
Data Field
CRC Field
Ack Field
12
20
6
8 * N
16
2
7
ID
DLC
CRC
EOF
SOF
SRR
IDE
RTR
r1
r0
ACK
Inter-Frame Space
Remote Frame
Inter-Frame Space or Overload Frame
44
Arbitration Field
Ctrl Field
CRC Field
Ack Field
12
6
16
2
7
ID
DLC
CRC
EOF
SOF
RTR
IDE
r0
ACK
Data Frame or Remote Frame
Error Frame
Inter-Frame Space or Overload Frame
Error Flag
Flag Echo
Error Delimiter
6
≤ 6
8
Any Frame
Inter-Frame Space
Data Frame or Remote Frame
Intermission
Suspend Transmission
Bus Idle
3
8
End Of Frame or Error Delimiter or Overload Delimiter
Overload Frame
Inter-Frame Space or Error Frame
Overload Flag
Overload Echo
Overload Delimiter
6
≤ 6
8
Notes:
• 0 <= N <= 8
• SOF = Start Of Frame
• ID = Identifier
• RTR = Remote Transmission Request
• IDE = Identifier Extension Bit
• r0 = Reserved Bit
• DLC = Data Length Code
• CRC = Cyclic Redundancy Code
• Error flag: 6 dominant bits if node is error active else 6 recessive bits.
• Suspend transmission: applies to error passive nodes only.
• EOF = End of Frame
• ACK = Acknowledge bit
• Ctrl = Control

14.6中断

bxCAN 占用 4 个专用的中断向量。通过设置 CAN 中断允许寄存器(CAN_IER)，每个中断源都可以单独允许和禁用。

图133 事件标志和中断产生



- **发送中断**可由下列事件产生：
  - 发送邮箱 0 变为空，CAN_TSR 寄存器的 RQCP0 位被置 1。
  - 发送邮箱 1 变为空，CAN_TSR 寄存器的 RQCP1 位被置 1。
  - 发送邮箱 2 变为空，CAN_TSR 寄存器的 RQCP2 位被置 1。
- **FIFO0 中断**可由下列事件产生：
  - FIFO0 接收到一个新报文，CAN_RF0R 寄存器的 FMP0 位不再是‘00’。
  - FIFO0 变为满的情况，CAN_RF0R 寄存器的 FULL0 位被置 1。
  - FIFO0 发生溢出的情况，CAN_RF0R 寄存器的 FOVR0 位被置 1。

- **FIFO1 中断**可由下列事件产生：
  - FIFO1 接收到一个新报文，CAN_RF1R 寄存器的 FMP1 位不再是‘00’。
  - FIFO1 变为满的情况，CAN_RF1R 寄存器的 FULL1 位被置 1。
  - FIFO1 发生溢出的情况，CAN_RF1R 寄存器的 FOVR1 位被置 1。
- **错误和状态变化中断**可由下列事件产生：
  - 出错情况，关于出错情况的详细信息请参考 CAN 错误状态寄存器 (CAN_ESR)。
  - 唤醒情况，在 CAN 接收引脚上监视到帧起始位(SOF)。
  - CAN 进入睡眠模式。

14.7 寄存器访问保护

对某些寄存器的错误访问会导致，一个 CAN 节点对整个 CAN 网络的暂时性干扰。因此，CAN_BTR 寄存器只能在 CAN 处于初始化模式时，由软件来修改。

虽然错误数据的发送对CAN网的网络层不会带来问题，但却会对应用层造成严重影响。因此，发送邮箱只能在它为空的状态下，由软件改写。请参见 图 125：发送邮箱状态。

同样道理，对过滤器的修改，也只能在禁用其所属的过滤器组的状态下，或设置整个过滤器为初始化模式（对 FINIT 位设置 1）下进行。此外，只有在设置整个过滤器为初始化模式下，才能修改过滤器的设置，即修改 CAN_FM0R，CAN_FS0R 和 CAN_FFA0R 寄存器。

14.8CAN 寄存器描述

关于寄存器描述中所用到的缩略词可参见第 1.1 节。

14.8.1 控制和状态寄存器

CAN 主控制寄存器 (CAN_MCR)

地址偏移量: 0x00
复位值: 0x0001 0002

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESET	保留							TTCM	ABOM	AWUM	NART	RFLM	TXFP	SLEEP	INRQ
rs								rw	rw	rw	rw	rw	rw	rw	rw

位31:16	保留，硬件强制为0。
位15	**RESET:** bxCAN 软件复位 0: 本外设正常工作 1: 对bxCAN进行强行复位，复位后bxCAN进入睡眠模式(FMP 位和CAN_MCR寄存器被初始化为其复位值)。此后硬件自动对该位清0
位14:8	保留，硬件强制为0。
位7	**TTCM:** 时间触发通信模式 0: 禁止时间触发通信模式 1: 允许时间触发通信模式 **注:** 要想了解关于时间触发通信模式的更多信息，请参考14.5.2：时间触发通信模式
位6	**ABOM:** 自动离线(Bus-Off)管理 该位决定CAN硬件在什么条件下可以退出离线状态。 0: 离线状态的退出是在，软件对CAN_MCR寄存器的INRQ位进行置1随后清0后，一旦硬件检测到128次11位连续的隐性位，就退出离线状态。 1: 一旦硬件检测到128次11位连续的隐性位，自动退出离线状态 **注:** 要想了解关于离线状态的更多信息，请参考14.5.6：出错管理
位5	**AWUM:** *自动唤醒模式* 该位决定CAN处在睡眠模式时由硬件还是软件唤醒 0: 睡眠模式通过清除CAN_MCR寄存器的SLEEP位，由软件唤醒 1: 睡眠模式通过检测CAN报文，由硬件自动唤醒。唤醒的同时，硬件自动对CAN_MSR寄存器的SLEEP和SLAK位清0

位4	**NART:** 禁止报文自动重传 0: 按照CAN标准，CAN硬件在发送报文失败时会一直自动重传直到发送成功 1: CAN报文只被发送1次，不管发送的结果如何（成功、出错或仲裁丢失）
位3	**RFLM:** 接收FIFO锁定模式 0: 在接收溢出时FIFO未被锁定，当接收FIFO的报文未被读出，下一个收到的报文会覆盖原有的报文 1: 在接收溢出时FIFO被锁定，当接收FIFO的报文未被读出，下一个收到的报文会被丢弃
位2	**TXFP:** 发送FIFO优先级 当有多个报文同时在等待发送时，该位决定这些报文的发送顺序 0: 优先级由报文的标识符来决定 1: 优先级由发送请求的顺序来决定
位1	**SLEEP:** 睡眠模式请求 软件对该位置1可以请求CAN进入睡眠模式，一旦当前的CAN活动（发送或接收报文）结束，CAN就进入睡眠 软件对该位清0使CAN退出睡眠模式 当设置了AWUM位且在CAN Rx信号中检测出SOF位时，硬件对该位清0。 在复位后该位被置1－CAN在复位后处于睡眠模式
位0	INRQ: 初始化请求 软件对该位清0可使CAN从初始化模式进入正常工作模式：当CAN在接收引脚检测到连续的11个隐性位后，CAN就达到同步，并为接收和发送数据作好准备了。为此，硬件相应地对CAN_MSR寄存器的INAK位清0。 软件对该位置1可使CAN从正常工作模式进入初始化模式：一旦当前的CAN活动（发送或接收）结束，CAN就进入初始化模式。相应地，硬件对CAN_MSR寄存器的INAK位置1。

CAN 主状态寄存器 (CAN_MSR)

地址偏移量: 0x04
复位值: 0x0000 0C02

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留				RX	SAMP	RXM	TXM	保留			SLAKI	WKUI	ERRI	SLAK	INAK
				r	r	r	r				rc_w1	rc_w1	rc_w1	r	r

位31:12	保留位，硬件强制为0
位11	RX: CAN接收电平 该位反映CAN接收引脚（CAN_RX）的实际电平
位10	SAMP: 上次采样值 CAN接收引脚的上次采样值(对应于当前接收位的值).
位9	RXM: 接收模式 该位为1表示CAN当前为接收器

位8	TXM: 发送模式 该位为1表示CAN当前为发送器
位7:5	保留位，硬件强制为0
位4	SLAKI: 睡眠确认中断 当SLKIE=1, 一旦CAN进入睡眠模式硬件就对该位置1，紧接着相应的中断被触发。软件可对该位清0，当SLAK位被清0时硬件也对该位清0。 注: 当SLKIE=0, 不应该查询该位，而应该查询SLAK位来获知睡眠状态
位3	WKUI: 唤醒中断挂号 当CAN处于睡眠状态，一旦帧起始位(SOF)被检测到，硬件就对该位置1；并且如果CAN_IER寄存器的WKUIE位为1，则相应的中断被触发。 该位由软件清0
位2	ERRI: 出错中断挂号 当由于检测到出错而对CAN_ESR 寄存器的某位置1，并且CAN_IER 寄存器的相应中断使能位也被置1时，硬件对该位置1；并且如果CAN_IER寄存器的ERRIE位为1则错误中断被触发。 该位由软件清0
位1	SLAK: 睡眠模式确认 当CAN进入睡眠模式时硬件就对该位置1，从而供软件进行状态查询。 该位是对软件请求进入睡眠模式的确认(对CAN_MCR寄存器的SLEEP位置1)。当CAN退出睡眠模式时硬件对该位清0 (需要跟CAN总线同步)。 这里跟CAN总线同步是指，硬件需要在CAN的RX引脚上检测到连续的11位隐性位。 注: 通过软件或硬件对CAN_MCR 的SLEEP位清0，是开启退出睡眠模式过程的唯一途径。请参考CAN_MCR寄存器的AWUM位的描述，以了解对SLEEP位进行硬件清0的详细信息
位0	INAK: 初始化确认 当CAN进入初始化模式时硬件就对该位置1，从而供软件进行状态查询。 该位是对软件请求进入初始化模式的确认(对CAN_MCR寄存器的INRQ位置1)。 当CAN退出初始化模式时硬件对该位清0 (需要跟CAN总线同步)。这里跟CAN总线同步是指，硬件需要在CAN的RX引脚上检测到连续的11位隐性位。

CAN发送状态寄存器 (CAN_TSR)

地址偏移量: 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	保留			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	保留			TERR1	ALST1	TXOK1	RQCP1	ABRQ0	保留			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* 当由多个邮箱在等待发送报文，且邮箱2的优先级最低时，硬件对该位置1
位30	**LOW1:** *邮箱1最低优先级标志* 当由多个邮箱在等待发送报文，且邮箱1的优先级最低时，硬件对该位置1

位29	**LOW0:** *邮箱0最低优先级标志* 当由多个邮箱在等待发送报文，且邮箱0的优先级最低时，硬件对该位置1
位28	**TME2:** *发送邮箱2空* 当邮箱2中没有等待发送的报文时，硬件对该位置1
位27	**TME1:** *发送邮箱1空* 当邮箱1中没有等待发送的报文时，硬件对该位置1
位26	**TME0:** *发送邮箱0空* 当邮箱0中没有等待发送的报文时，硬件对该位置1
位25:24	**CODE[1:0]:** *邮箱号* 当有至少1个发送邮箱为空时，邮箱号为下一个空的发送邮箱号。 当所有的发送邮箱都为空时，邮箱号为优先级最低的那个发送邮箱号
位23	**ABRQ2:** *邮箱2中止发送* 软件对该位置1可以中止邮箱2的发送请求，当邮箱2的发送报文被清除时硬件对该位清0。 如果邮箱2中没有等待发送的报文，则对该位置1没有任何效果
位22:20	保留位，硬件强制其值为0
位19	**TERR2:** *邮箱2发送失败* 当邮箱2因为出错而导致发送失败时，对该位置1
位18	**ALST2:** *邮箱2仲裁丢失* 当邮箱2因为仲裁丢失而导致发送失败时，对该位置1
位17	**TXOK2:** *邮箱2发送成功* 每次在邮箱2进行发送尝试后，硬件对该位进行更新： 0: 上次发送尝试失败 1: 上次发送尝试成功 当邮箱2的发送请求被成功完成后，硬件对该位置1。请参见图125
位16	**RQCP2:** *邮箱2请求完成* 当上次对邮箱2的请求（发送或中止）完成后，硬件对该位置1。 软件对该位写“1”可以对其清0；当硬件接收到发送请求时也对该位清0（CAN_TI2R 寄存器的TXRQ位被置1）。 该位被清0时，邮箱2的其它发送状态位（TXOK2, ALST2和TERR2）也被清0
位15	**ABRQ1:** *邮箱1中止(发送)* 软件对该位置1可以中止邮箱1的发送请求，当邮箱1的发送报文被清除时硬件对该位清0。 如果邮箱1中没有等待发送的报文，则对该位置1没有任何效果
位14:12	保留位，硬件强制其值为0
位11	**TERR1:** *邮箱1发送失败* 当邮箱1因为出错而导致发送失败时，对该位置1
位10	**ALST1:** *邮箱1仲裁丢失* 当邮箱1因为仲裁丢失而导致发送失败时，对该位置1
位9	**TXOK1:** *邮箱1发送成功* 每次在邮箱1进行发送尝试后，硬件对该位进行更新： 0: 上次发送尝试失败 1: 上次发送尝试成功 当邮箱1的发送请求被成功完成后，硬件对该位置1。请参见图125

位8	**RQCP1:** *邮箱1请求完成* 当上次对邮箱1的请求（发送或中止）完成后，硬件对该位置1。 软件对该位写“1”可以对其清0；当硬件接收到发送请求时也对该位清0（CAN_TI1R 寄存器的TXRQ位被置1）。 该位被清0时，邮箱1的其它发送状态位（TXOK1, ALST1和TERR1）也被清0
位7	**ABRQ0:** *邮箱0中止(发送)* 软件对该位置1可以中止邮箱0的发送请求，当邮箱0的发送报文被清除时硬件对该位清0。 如果邮箱0中没有等待发送的报文，则对该位置1没有任何效果
位6:4	保留位，硬件强制其值为0
位3	**TERR0:** *邮箱0发送失败* 当邮箱0因为出错而导致发送失败时，对该位置1
位2	**ALST0:** *邮箱0仲裁丢失* 当邮箱0因为仲裁丢失而导致发送失败时，对该位置1
位1	**TXOK0:** *邮箱0发送成功* 每次在邮箱0进行发送尝试后，硬件对该位进行更新： 0: 上次发送尝试失败 1: 上次发送尝试成功 当邮箱0的发送请求被成功完成后，硬件对该位置1。请参见图125
位0	**RQCP1:** *邮箱0请求完成* 当上次对邮箱0的请求（发送或中止）完成后，硬件对该位置1。 软件对该位写“1”可以对其清0；当硬件接收到发送请求时也对该位清0（CAN_TI0R 寄存器的TXRQ位被置1）。 该位被清0时，邮箱0的其它发送状态位（TXOK0, ALST0和TERR0）也被清0

CAN接收FIFO 0 寄存器 (CAN_RF0R)

地址偏移量: 0x0C
复位值: 0x00

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留										RFOM0	FOVR0	FULL0	保留	FMP0[1:0]	
										rs	rc w1	rc w1		r	r

位31:6	保留位，硬件强制为0
位5	**RFOM0:** *释放接收FIFO 0输出邮箱* 软件通过对该位置1来释放接收FIFO的输出邮箱。如果接收FIFO为空，那么对该位置1没有任何效果，即只有当FIFO中有报文时对该位置1才有意义。如果FIFO中有2个以上的报文，由于FIFO的特点，软件为了访问第2个报文，就需要释放输出邮箱才行。 当输出邮箱被释放时，硬件对该位清0
位4	**FOVR0:** *FIFO 0 溢出*

	当FIFO 0已满，又收到新的报文且报文符合过滤条件，硬件对该位置1。 该位由软件清0
位3	**FULL0:** *FIFO 0 满* 当有3个报文被存入FIFO 0时，硬件对该位置1。 该位由软件清0
位2	保留位，硬件强制其值为0
位1:0	**FMP0[1:0]:** *FIFO 0 报文数目* FIFO 0报文数目这2位反映了当前接收FIFO 0中存放的报文数目。 每当1个新的报文被存入接收FIFO 0，硬件就对FMP0加1。 每当软件对RFOM0位写1来释放输出邮箱，FMP0就被减1，直到其为0

CAN接收FIFO 1 寄存器(CAN_RF1R)

地址偏移量: 0x10
复位值: 0x00

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留										RFOM1	FOVR1	FULL1	保留	FMP1[1:0]	
										rs	rc w1	rc w1		r	r

位31:6	保留位，硬件强制为0
位5	**RFOM1:** *释放接收FIFO 1输出邮箱* 软件通过对该位置1来释放接收FIFO的输出邮箱。如果接收FIFO为空，那么对该位置1没有任何效果，即只有当FIFO中有报文时对该位置1才有意义。如果FIFO中有2个以上的报文，由于FIFO的特点，软件为了访问第2个报文，就需要释放输出邮箱才行。 当输出邮箱被释放时，硬件对该位清0
位4	**FOVR1:** *FIFO 1 溢出* 当FIFO 1已满，又收到新的报文且报文符合过滤条件，硬件对该位置1。 该位由软件清0
位3	**FULL1:** *FIFO 1 满* 当有3个报文被存入FIFO 1时，硬件对该位置1。 该位由软件清0
位2	保留位，硬件强制其值为0
位1:0	**FMP1[1:0]:** *FIFO 1报文数目* FIFO 1报文数目这2位反映了当前接收FIFO 1中存放的报文数目。 每当1个新的报文被存入接收FIFO 1，硬件就对FMP1加1。 每当软件对RFOM1位写1来释放输出邮箱，FMP1就被减1，直到其为0

CAN 中断允许寄存器 (CAN_IER)

地址偏移量: 0x14
复位值: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ERRIE	保留			LECIE	BOFIE	EPVIE	EWGIE	保留	EOVIE1	FFIE1	FMPIE1	FOVIE1	FFIE0	FMPIE0	TMEIE
rw				rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw

位31:18	保留位，硬件强制为0
位17	**SLKIE:** *睡眠中断允许* 0: 当SLAKI位被置1时，没有中断产生 1: 当SLAKI位被置1时，产生中断
位16	**WKUIE:** *睡眠唤醒中断允许* 0: 当WKUI位被置1时，没有中断产生 1: 当WKUI位被置1时，产生中断
位15	**ERRIE:** *错误中断允许* 0: 当CAN_ESR寄存器有错误挂号时，没有中断产生 1: 当CAN_ESR寄存器有错误挂号时，产生中断
位14:12	保留位，硬件强制为0
位11	**LECIE:** *上次错误号中断允许* 0: 当检测到错误从而硬件对LEC[2:0]写入非0值时，不会对ERRI位置1 1: 当检测到错误从而硬件对LEC[2:0]写入非0值时，对ERRI位置1
位10	**BOFIE:** *离线中断允许* 0: 当BOFF位被置1时，不会对ERRI位置1 1: 当BOFF位被置1时，对ERRI位置1
位9	**EPVIE:** *Error Passive Interrupt Enable* 0: 当EPVF位被置1时，不会对ERRI位置1 1: 当EPVF位被置1时，对ERRI位置1
位8	**EWGIE:** *错误警告中断允许* 0: 当EWGF位被置1时，不会对ERRI位置1 1: 当EWGF位被置1时，对ERRI位置1
位7	保留位，硬件强制为0
位6	**FOVIE1:** *FIFO1溢出中断允许* 0: 当FIFO1的FOVR位被置1时，没有中断产生 1: 当FIFO1的FOVR位被置1时，产生中断
位5	**FFIE1:** *FIFO1满中断允许* 0: 当FIFO1的FULL位被置1时，没有中断产生 1: 当FIFO1的FULL位被置1时，产生中断

位4	**FMPIE1:** *FIFO1消息挂号中断允许* 0: 当FIFO1的FMP[1:0]位被写入非0值时，没有中断产生 1: 当FIFO1的FMP[1:0]位被写入非0值时，产生中断
位3	**FOVIE0:** *FIFO0溢出中断允许* 0: 当FIFO0的FOVR位被置1时，没有中断产生 1: 当FIFO0的FOVR位被置1时，产生中断
位2	**FFIE0:** *FIFO0满中断允许* 0: 当FIFO0的FULL位被置1时，没有中断产生 1: 当FIFO0的FULL位被置1时，产生中断
位1	**FMPIE0:** *FIFO0消息挂号中断允许* 0: 当FIFO0的FMP[1:0]位被写入非0值时，没有中断产生 1: 当FIFO0的FMP[1:0]位被写入非0值时，产生中断
位0	**TMEIE:** *发送邮箱空中断允许* 0: 当RQCPx位被置1时，没有中断产生 1: 当RQCPx位被置1时，产生中断 **注:** 请参考14.6

CAN 错误状态寄存器 (CAN_ESR)

地址偏移量: 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
保留									LEC[2:0]			保留	BOFF	EPVF	WEGF
									rw	rw	rw		r	r	r

位31:24	**REC[7:0]:** *接收错误计数器* 这是对CAN协议的故障界定机制接收部分的实现。按照CAN的标准，当接收出错时， 根据出错的情况该计数器加1或加8；而在每次接收成功后，该计数器减1，或减少其值为120－当该计数器的值大于127时。当该计数器的值超过127时，CAN进入错误被动状态。
位23:16	**TEC[7:0]:** *发送错误计数器* 与上面相似，这是对CAN协议的故障界定机制发送部分的实现。
位15:7	保留位，硬件强制为0
位6:4	**LEC[2:0]:** *上次错误代码* 在检测到CAN总线上发生错误时，硬件根据出错情况设置其为1～6的值。当报文被正确发送或接收后，硬件清除其值为‘0’。 硬件没有使用错误代码7，软件可以设置该值，从而可以检测代码的更新。000: 没有错误 001: 位填充错 010: 格式(Form)错 011: 确认(ACK)错 100: 隐性位错

	101: 显性位错 110: CRC错 111: 由软件设置
位3	保留位，硬件强制为0
位2	**BOFF:** *离线(Bus Off)标志* 当进入离线状态时，硬件对该位置1。当发送错误计数器TEC溢出，即大于255时，CAN进入离线状态。请参考14.5.6
位1	**EPVF:** *错误被动(Error Passive)标志* 当出错次数达到错误被动的阈值时，硬件对该位置1。 (接收错误计数器或发送错误计数器的值>127)
位0	**EWGF:** *错误警告标志* 当出错次数达到警告的阈值时，硬件对该位置1。 (接收错误计数器或发送错误计数器的值≥96)

CAN 位时间特性寄存器 (CAN_BTR)

地址偏移量: 0x1C
复位值: 0x0123 0000

注： 当CAN处于初始化模式时，该寄存器只能由软件访问。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SILM	LBKM	保留				SJW[1:0]		保留	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
保留	保留					BRP[9:0]									
~~rw~~						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位31	**SILM:** *静默模式（用于调试）* 0: 正常状态 1: 静默模式
位30	**LBKM:** *环回模式（用于调试）* 0: 禁止环回模式 1: 允许环回模式
位29:26	保留位，硬件强制为0
位25:24	**SJW[1:0]:** *重新同步跳跃宽度* 为了重新同步，该位域定义了CAN硬件在每位中可以延长或缩短多少个时间单元的上限。 $t_{RJW} = t_{CAN} \times (SJW[1:0] + 1)$
位23	保留位，硬件强制为0
位22:20	**TS2[2:0]:** *时间段2* 该位域定义了时间段2占用了多少个时间单元 tBS2 = tCAN x (TS2[2:0] + 1)

位19:16	**TS1[3:0]:** *时间段1* 该位域定义了时间段1占用了多少个时间单元 tBS1 = tCAN x (TS1[3:0] + 1) 关于位时间特性的详细信息，请参考14.5.7：位时间特性
位15	保留位，应强制其值为0
位14:10	保留位，硬件强制其值为0
位9:0	**BRP[9:0]:** *波特率分频器* 该位域定义了时间单元（tq）的时间长度 tq = (BRP[9:0]+1) x tPCLK

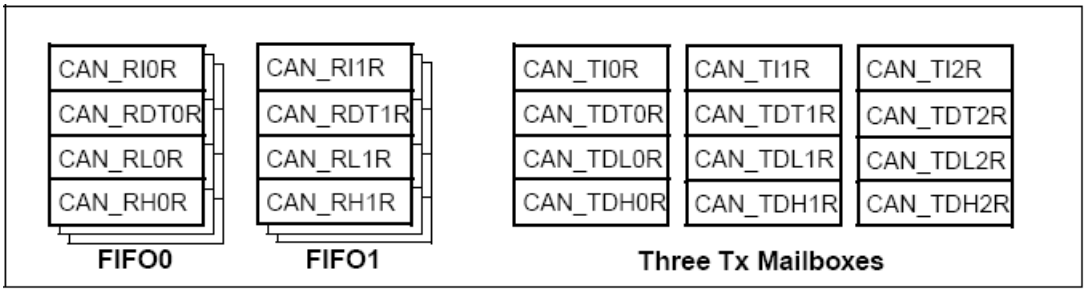
14.8.2 邮箱寄存器

本节描述发送和接收邮箱寄存器。关于寄存器映像的详细信息，请参考 14.5.5：报文存储。

除了下述例外，发送和接收邮箱几乎一样：

- CAN_RDTxR 寄存器的FMI域
- 接收邮箱是只读的
- 接发送邮箱只有在它为空时才是可写的，发送邮箱为空对应于CAN_TSR 寄存器的相应TME位为1

共有 3 个发送和 2 个接收邮箱。每个接收邮箱为 3 级深度的 FIFO，并且只能访问 FIFO 中最先收到的报文。
每个邮箱包含 4 个寄存器。



发送邮箱标识符寄存器 (CAN_TIxR) (x=0..2)

地址偏移量: 0x180，0x190，0x1A0
复位值: 0xXXXX XXXX，X 未定义（除了第 0 位，复位时 TXRQ=0）

注： 1 当其所属的邮箱处在等待发送的状态时，该寄存器是写保护的
2 该寄存器实现了发送请求控制功能（第0位）－复位值为0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
STID[10:0]											EXID[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]:** *标准标识符* 标识符的标准部分
位20:3	**EXID[17:0]:** *扩展标识符* 标识符的扩展部分（如果使用了扩展标识符）
位2	**IDE:** *标识符选择* 该位决定发送邮箱中报文使用的标识符类型 0: 使用标准标识符 1: 使用扩展标识符
位1	**RTR:** *远程发送请求* 0: 数据帧 1: 远程帧
位0	**TXRQ:** *发送数据请求* 由软件对其置1，来请求发送其邮箱的数据。当数据发送完成，邮箱为空时，硬件对其清0

发送邮箱数据长度和时间戳寄存器 (CAN_TDTxR) (x=0..2)

地址偏移量: 0x184，0x194，0x1A4
复位值: 0xXXXX XXXX，X 未定义

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
保留							TGT	保留				DLC[3:0]			
							rw					rw	rw	rw	rw

位31:16	**TIME[15:0]:** *报文时间戳* 该域包含了，在发送该报文SOF的时刻，16位定时器的值
位15:9	保留位
位8	**TGT:** *发送时间戳* 只有在CAN处于时间触发通信模式，即CAN_MCR寄存器的TTCM位为1时，该位才有效。 0: 不发送时间戳 1: 发送时间戳TIME[15:0]。在长度为8的报文中，时间戳TIME[15:0]是最后2个发送的字节：TIME[7:0]作为第7个字节，TIME[15:8]为第8个字节，它们替换了

	写入CAN_TDHxR[31:16]的数据(DATA6[7:0]和DATA7[7:0])。为了把时间戳的2个字节发送出去，DLC必须编程为8。
位7:4	保留位
位3:0	**DLC[15:0]:** *发送数据长度* 该域指定了数据报文的数据长度或者远程帧请求的数据长度。1个报文包含0到8个字节数据，而这由DLC决定。

发送邮箱低字节数据寄存器 (CAN_TDLxR) (x=0..2)

地址偏移量: 0x188，0x198，0x1A8
复位值: 0xXXXX XXXX，X 未定义
当邮箱为空时，寄存器中的所有位为只读。

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] :** *字节3* 报文的数据字节3
位23:16	**DATA2[7:0] :** *字节2* 报文的数据字节2
位15:8	**DATA1[7:0] :** *字节1* 报文的数据字节1
位7:0	**DATA0[7:0] :** *字节0* 报文的数据字节0。 报文包含0到8个字节数据，且从字节0开始。

发送邮箱高字节数据寄存器 (CAN_TDHxR) (x=0..2)

地址偏移量: 0x18C，0x19C，0x1AC
复位值: 0xXXXX XXXX，X 未定义
当邮箱为空时，寄存器中的所有位为只读。

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] :** *字节7* 报文的数据字节7 **注:** 如果CAN_MCR寄存器的TTCM位为1，且该邮箱的TGT位也为1，那么DATA7和DATA6将被TIME时间戳代替。
位23:16	**DATA6[7:0] :** *字节6* 报文的数据字节6
位15:8	**DATA5[7:0] :** *字节5* 报文的数据字节5
位7:0	**DATA4[7:0] :** *字节4* 报文的数据字节4。

接收FIFO邮箱标识符寄存器 (CAN_RIxR) (x=0..1)

地址偏移量: 0x1B0，0x1C0
复位值: 0xXXXX XXXX，X 未定义

注： 所有接收邮箱寄存器都是只读的。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
STID[10:0]											EXID[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	保留
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	

位31:21	**STID[10:0]:** *标准标识符* 标识符的标准部分
位20:3	**EXID[17:0]:** *扩展标识符* 标识符的扩展部分（如果使用了扩展标识符）
位2	**IDE:** *标识符选择* 该位决定接收邮箱中报文使用的标识符类型 0: 使用标准标识符 1: 使用扩展标识符

位1	**RTR:** *远程发送请求* 0: 数据帧 1: 远程帧
位0	保留位

接收FIFO邮箱数据长度和时间戳寄存器 (CAN_RDTxR) (x=0..1)

地址偏移量: 0x1B4，0x1C4
复位值: 0xXXXX XXXX，X 未定义

注： 所有接收邮箱寄存器都是只读的。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TIME[15:0]															
r	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]								保留				DLC[3:0]			
r	r	r	r	r	r	r	r					r	r	r	r

位31:16	**TIME[15:0]:** *报文时间戳* 该域包含了，在接收该报文SOF的时刻，16位定时器的值。
位15:8	**FMI[15:0]:** *过滤器匹配序号* 这里是存在邮箱中的信息传送的过滤器序号。关于标识符过滤的细节，请参考14.5.4中有关过滤器匹配序号。
位7:4	保留位，硬件强制为0
位3:0	**DLC[15:0]:** *接收数据长度* 该域表明接收数据帧的数据长度（0～8）。对于远程帧，数据长度DLC恒为0。

接收FIFO邮箱低字节数据寄存器 (CAN_RDLxR) (x=0..1)

地址偏移量: 0x1B8，0x1C8
复位值: 0xXXXX XXXX，X 未定义

注： 所有接收邮箱寄存器都是只读的。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATA3[7:0]								DATA2[7:0]							
r	r	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] :** *字节3* 报文的数据字节3
位23:16	**DATA2[7:0] :** *字节2* 报文的数据字节2
位15:8	**DATA1[7:0] :** *字节1* 报文的数据字节1
位7:0	**DATA0[7:0] :** *字节0* 报文的数据字节0。 报文包含0到8个字节数据，且从字节0开始。

接收FIFO邮箱高字节数据寄存器 (CAN_RDHxR) (x=0..1)

地址偏移量: 0x1BC，0x1CC
复位值: 0xXXXX XXXX，X 未定义

注： 所有接收邮箱寄存器都是只读的。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATA7[7:0]								DATA6[7:0]							
r	r	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] :** *字节7* 报文的数据字节7 **注:** 如果CAN_MCR寄存器的TTCM位为1，且该邮箱的TGT位也为1，那么DATA7和DATA6将被TIME时间戳代替。
位23:16	**DATA6[7:0] :** *字节6* 报文的数据字节6
位15:8	**DATA5[7:0] :** *字节5* 报文的数据字节5
位7:0	**DATA4[7:0] :** *字节4* 报文的数据字节4。

14.8.3 CAN过滤器寄存器

(这些寄存器只出现在主 CAN—CAN0 中)

CAN 过滤器主控寄存器 (CAN_FMR)

地址偏移量: 0x200
复位值: 0x2A1C 0E01

注： 该寄存器的非保留位完全由软件控制。

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

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

位31:1	保留位，强制为复位值
位0	**FINIT :** 过滤器初始化模式 针对所有过滤器组的初始化模式设置。 0: 过滤器组工作在正常模式 1: 过滤器组工作在初始化模式

CAN 过滤器模式寄存器 (CAN_FM0R)

地址偏移量: 0x204
复位值: 0x0000 0000

注： 只有在设置 CAN_FMR(FINIT=1)，使过滤器处于初始化模式下，才能对该寄存器写入。

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

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

注： 请参考 图 127：过滤器组位宽设置－寄存器组织。

位31:14	保留位，硬件强制为0
位13:0	**FBMx :** *过滤器模式* 过滤器组x的工作模式。 0: 过滤器组x的2个32位寄存器工作在标识符屏蔽位模式 1: 过滤器组x的2个32位寄存器工作在标识符列表模式

CAN 过滤器位宽寄存器 (CAN_FS0R)

地址偏移量: 0x20C

复位值: 0x0000 0000

注： 只有在设置CAN_FMR(FINIT=1)，使过滤器处于初始化模式下，才能对该寄存器写入。

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

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

注： 请参考 图 127图127：过滤器组位宽设置－寄存器组织。

位31:14	保留位，硬件强制为0
位13:0	**FSCx :** 过滤器位宽设置 过滤器组x（13～0）的位宽。 0: 过滤器位宽为2个16位 1: 过滤器位宽为单个32位

CAN 过滤器FIFO关联寄存器 (CAN_FFA0R)

地址偏移量: 0x214

复位值: 0x0000 0000

注： 只有在设置CAN_FMR(FINIT=1)，使过滤器处于初始化模式下，才能对该寄存器写入。

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留		FFA13	FFA12	FFA11	FFA10	FFA9	FFA8	FFA7	FFA6	FFA5	FFA4	FFA3	FFA2	FFA1	FFA0
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位31:14	保留位，硬件强制为0
位13:0	**FFAx :** *过滤器位宽设置* 报文在通过了某过滤器的过滤后，将被存放到其关联的FIFO中。 0: 过滤器被关联到FIFO0 1: 过滤器被关联到FIFO1

CAN 过滤器激活寄存器 (CAN_FA0R)

地址偏移量: 0x21C
复位值: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留		FACT13	FACT12	FACT11	FACT10	FACT9	FACT8	FACT7	FACT6	FACT5	FACT4	FACT3	FACT2	FACT1	FACT0
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位31:14	保留位，硬件强制为0
位13:0	FACTx : *过滤器激活* 软件对某位设置1来激活相应的过滤器。只有对FACTx位清0，或对CAN_FMR寄存器的FINIT位设置1后，才能修改相应的过滤器寄存器x（CAN_FxR[0:1]） 0: 过滤器被禁用 1: 过滤器被激活

CAN 过滤器组x寄存器 (CAN_FxR[1：0]) (x＝0..13)

地址偏移量: 0x240h..0x2AC
复位值: 0xXXXX XXXX，X 未定义

注： 共有 14 组过滤器：x=0..13。每组过滤器由 2 个 32 位的寄存器，CAN_FxR[1:0]组成。
只有在相应的 FACTx 位清 0，或 CAN_FMR 寄存器的 FINIT 位为 1 的情况下，才能修改相应的过滤器寄存器。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FB31	FB30	FB29	FB28	FB27	FB26	FB25	FB24	FB23	FB22	FB21	FB20	FB19	FB18	FB17	FB16
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FB15	FB14	FB13	FB12	FB11	FB10	FB9	FB8	FB7	FB6	FB5	FB4	FB3	FB2	FB1	FB0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

在所有的配置情况下：

位31:0	FB[31:0] : *过滤器位* 标识符模式 寄存器的每位对应于，所期望的标识符的相应位的电平。 0: 期望相应位为显性位 1: 期望相应位为隐性位 屏蔽位模式

	0: 不关心，该位不用于比较 1: 必须匹配，

注： 根据过滤器位宽和模式的不同设置，过滤器组中的两个寄存器的功能也不尽相同。
关于过滤器的映射，功能描述和屏蔽寄存器的关联，请参见 14.5.4：标识符过滤。
屏蔽位模式下的屏蔽/标识符寄存器，跟**标识符列表模式**下的寄存器位定义相同。

关于过滤器组寄存器的地址，请参见 表 44。

14.9 bxCAN寄存器列表

关于寄存器的起始地址，请参见第 1 章。

表44 bxCAN－寄存器列表及其复位值

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
000h	CAN_MCR	保留																								TTCM	ABOM	AWUM	NART	RFLM	TXFP	SLEEP	INRQ
	复位值																									0	0	0	0	0	0	1	0
004h	CAN_MSR	保留																				RX	SAMP	RXM	TXM	保留			SLAKI	WKUI	ERRI	SLAK	INAK
	复位值																					1	1	0	0				0	0	0	1	0
008h	CAN_TSR	LOW[2:0]			TME[2:0]			CODE[1:0]		ABRQ2	保留			TERR2	ALST2	TXOK2	RQCP2	ABRQ1	保留			TERR1	ALST1	TXOK1	RQCP1	ABRQ0	保留			TERR0	ALST0	TXOK0	RQCP0
	复位值	0	0	0	1	1	1	0	0	0				0	0	0	0	0				0	0	0	0	0				0	0	0	0
00Ch	CAN_RFOR	保留																										RFOM0	FOVR0	FULL0	保留	FMP0[1:0]		
	复位值																											0	0	0		0	0	
010h	CAN_RF1F	保留																										RFOM1	FOVR1	FULL1	保留	FMP1[1:0]		
	复位值																											0	0	0		0	0	
014h	CAN_IER	保留														SLKIE	WKUIE	ERRIE	保留			LECIE	BOFIE	EPVIE	EWGIE	保留	FOVIE1	FFIE1	FMPIE1	FOVIE0	FFIE0	FMPIE0	TMEIE
	复位值															0	0	0				0	0	0	0		0	0	0	0	0	0	0
018h	CAN_ESR	REC[7:0]								TEC[7:0]									保留							LEC[2:0]			保留	BOFF	EPVF	EWGF	
	复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0									0	0	0		0	0	0	
01Ch	CAN_BTR	SILM	LBKM	保留					SJW[1:0]		保留		TS2[2:0]			TS1[3:0]			保留							BRP[9:0]								
	复位值	0	0						0	0			0	1	0	0	0	1	1							0	0	0	0	0	0	0	0	0
020h~17Fh	保留																																
180h	CAN_TIOR	STID[10:0]											EXID[17:0]																			IDE	RTR	TXRQ
	复位值	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
184h	CAN_TDTOR	TIME[15:0]																保留							TGT	保留				DLC[3:0]			
	复位值	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x								x					x	x	x	x
188h	CAN_TDLOR	DATA3[7:0]								DATA2[7:0]								DATA1[7:0]								DATA0[7:0]							
	复位值	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
18Ch	CAN_TDHOR	DATA7[7:0]								DATA6[7:0]								DATA5[7:0]								DATA4[7:0]							
	复位值	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

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
190h	CAN_TI1R	STID[10:0]											EXID[17:0]																		IDE	RTR	TXRQ
	复位值	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
194h	CAN_TDT1R	TIME[15:0]																保留							TGT	保留					DLC[3:0]			
	复位值	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x								x						x	x	x	x
198h	CAN_TDL1R	DATA3[7:0]								DATA2[7:0]								DATA1[7:0]								DATA0[7:0]							
	复位值	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
19Ch	CAN_TDH1R	DATA7[7:0]								DATA6[7:0]								DATA5[7:0]								DATA4[7:0]							
	复位值	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
1A0h	CAN_TI2R	STID[10:0]											EXID[17:0]																		IDE	RTR	TXRQ
	复位值	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
1A4h	CAN_TDT2R	TIME[15:0]																保留							TGT	保留					DLC[3:0]			
	复位值	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x								x						x	x	x	x
1A8h	CAN_TDL2R	DATA3[7:0]								DATA2[7:0]								DATA1[7:0]								DATA0[7:0]							
	复位值	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
1ACh	CAN_TDH2R	DATA7[7:0]								DATA6[7:0]								DATA5[7:0]								DATA4[7:0]							
	复位值	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
1B0h	CAN_RI0R	STID[10:0]											EXID[17:0]																		IDE	RTR	保留
	复位值	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	
1B4h	CAN_RDT0R	TIME[15:0]																FMI[7:0]								保留					DLC[3:0]			
	复位值	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
1B8h	CAN_RDL0R	DATA3[7:0]								DATA2[7:0]								DATA1[7:0]								DATA0[7:0]							
	复位值	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
1BCh	CAN_RDH0R	DATA7[7:0]								DATA6[7:0]								DATA5[7:0]								DATA4[7:0]							
	复位值	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
1C0h	CAN_RI1R	STID[10:0]											EXID[17:0]																		IDE	RTR	保留
	复位值	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	
1C4h	CAN_RDT1R	TIME[15:0]																FMI[7:0]								保留					DLC[3:0]			
	复位值	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
1C8h	CAN_RDL1R	DATA3[7:0]								DATA2[7:0]								DATA1[7:0]								DATA0[7:0]							
	复位值	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
1CCh	CAN_RDH1R	DATA7[7:0]								DATA6[7:0]								DATA5[7:0]								DATA4[7:0]							
	复位值	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
1D0h~ 1FFh	保留																																

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
200h	CAN_FMR	保留																															FINIT
	复位值																																1
204h	CAN_FMOR	保留																		FBM[13:0]													
	复位值																			0	0	0	0	0	0	0	0	0	0	0	0	0	0
208h	保留																																
20Ch	CAN_FSOR	保留																		FSC[13:0]													
	复位值																			0	0	0	0	0	0	0	0	0	0	0	0	0	0
210h	保留																																
214h	CAN_FFAOR	保留																		FFA[13:0]													
	复位值																			0	0	0	0	0	0	0	0	0	0	0	0	0	0
218h	保留																																
21Ch	CAN_FAOR	保留																		FACT[13:0]													
	复位值																			0	0	0	0	0	0	0	0	0	0	0	0	0	0
220h	保留																																
224~ 23Fh	保留																																
240h	CAN_FOR0	FB[31:0]																															
	复位值	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
244h	CAN_FOR1	FB[31:0]																															
	复位值	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
240h	CAN_F1R0	FB[31:0]																															
	复位值	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
244h	CAN_F1R1	FB[31:0]																															
	复位值	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
. . .	. . .	… …																															
2A8h	CAN_F13R0	FB[31:0]																															
	复位值	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
2ACh	CAN_F13R1	FB[31:0]																															
	复位值	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

15 I2C接口

15.1介绍

I^2C 总线接口连接微控制器和串行 I^2C 总线。它提供多主机功能，控制所有 I^2C 总线特定的时序、协议、仲裁和定时。支持标准和快速两种模式，同时与 SMBus 2.0 兼容。

I^2C 总线有多种用途，包括 CRC 码的生成和校验、SMBus(系统管理总线 System Management Bus)、PMBus(电源管理总线 Power Management Bus)。

根据特定设备的需要，可以使用 DMA 以减轻 CPU 的负担。

15.2主要特点

- 并行总线/I^2C 总线协议转换器
- 多主机功能：同一接口既可做主设备也可做从设备
- I^2C 主设备功能
  - 产生时钟
  - 产生起始和停止信号
- I^2C 从设备功能
  - 可编程的 I^2C 地址检测
  - 可响应 2 个从地址的双地址能力
  - 停止位检测
- 产生和检测 7 位/10 位地址和广播呼叫
- 支持不同的通讯速度
  - 标准速度(高至 100 kHz)
  - 快速(高至 400 kHz)
- 状态标志：
  - 发送器/接收器模式标志
  - 字节发送结束标志
  - I^2C 总线忙标志
- 错误标志
  - 主模式时的仲裁丢失

  - 地址/数据传输后的应答(ACK)错误
  - 检测到起始和停止错位
  - 禁止拉长时钟功能后的上溢或下溢
- 2 个中断向量
  - 1 个中断用于地址/数据通讯成功
  - 1 个中断用于出错
- 可选的拉长时钟功能
- 具单字节缓冲器的 DMA
- 可配置的 PEC(信息包错误检测)的产生或校验：
  - 发送模式中 PEC 值可以作为最后一个字节传输
  - 用于最后一个接收字节的 PEC 错误校验
- 兼容 SMBus 2.0
  - 25 ms 时钟低超时延时
  - 10 ms 主设备累积时钟低扩展时间
  - 25 ms 从设备累积时钟低扩展时间
  - 带 ACK 控制的硬件 PEC 产生/校验
  - 支持地址分辨协议（ARP）
- 兼容 SMBus

注意：　不是所有产品中都包含上述所有特性。请参考相关的数据手册，确认该产品支持的 I^2C 功能。

15.3 概述

I^2C 接口接收和发送数据，并将数据从串行转换成并行，或并行转换成串行。可以开启或禁止中断。接口通过数据引脚(SDA)和时钟引脚(SCL)连接到 I^2C 总线。允许连接到标准(高至 100 kHz)或快速(高至 400 kHz)I^2C 总线。

模式选择

接口可以下述 4 种模式中的一种运行：

- 从发送器模式
- 从接收器模式
- 主发送器模式
- 主接收器模式

默认模式为从模式。接口在生成起始条件后自动从从模式切换到主模式；当仲裁丢失或产生停止信号，则从主模式切换到从模式。允许多主机功能。

通信流

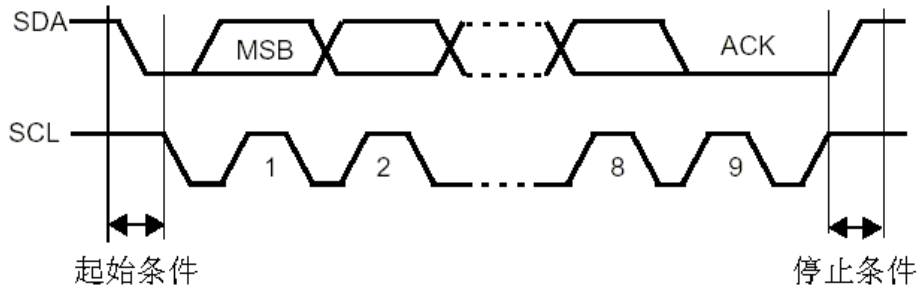
主模式时，I^2C 接口启动数据传输并产生时钟信号。串行数据传输总是以起始条件开始和停止条件结束。主模式时，由软件控制产生起始条件和停止条件。

从模式时，I^2C 接口能识别它自己的地址(7 位或 10 位)和广播呼叫地址。软件控制开启或禁止广播呼叫地址的识别。

数据和地址按 8 位/字节进行传输，高位在前。跟在起始条件后的第一、二个字节是地址（7 位模式为 1 个字节，10 位模式为 2 个字节）。地址只在主模式发送。

在一个字节传输的 8 个时钟后的第 9 个时钟期间，接收器必须回送一个应答位(ACK)给发送器。参考下图。

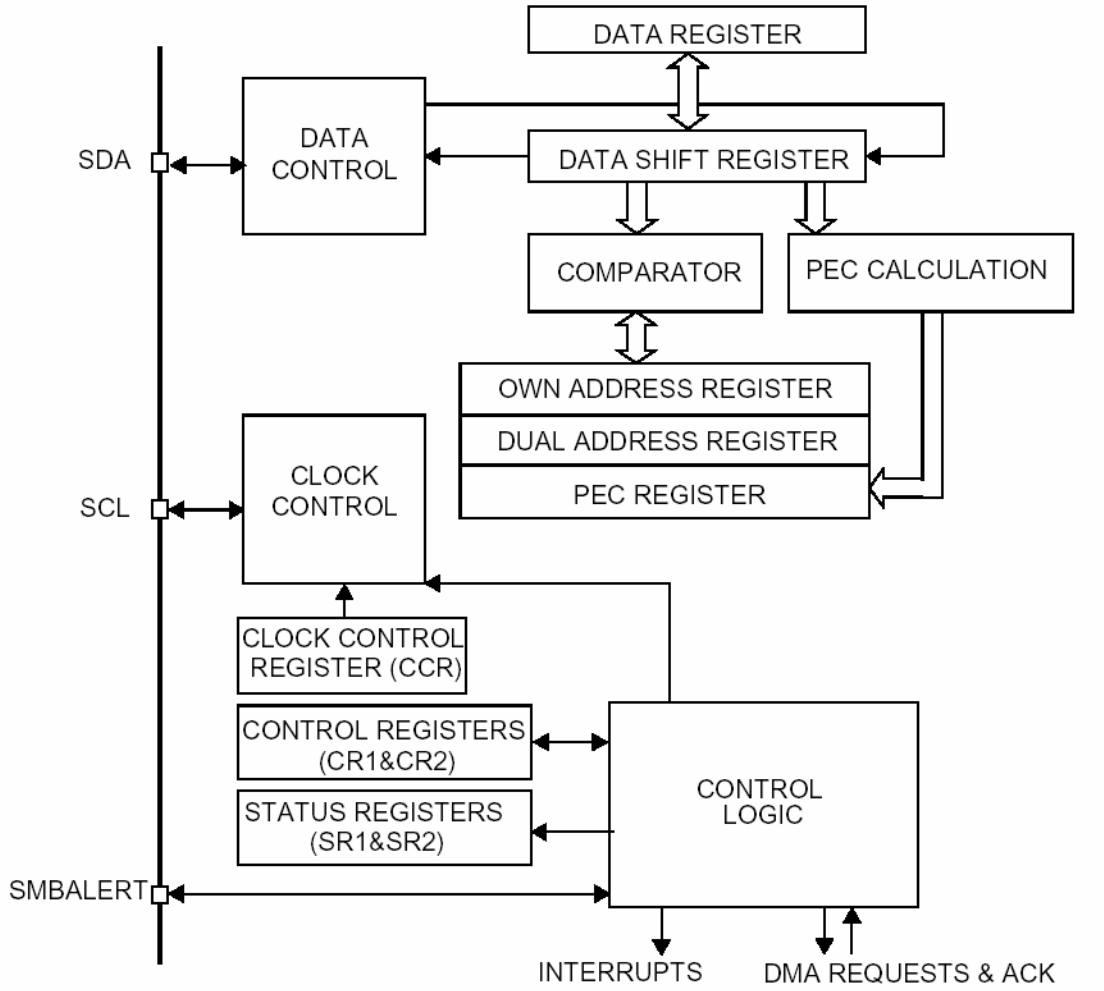
图134 I^2C 总线协议



软件可以开启或禁止应答(ACK)，I^2C 接口的地址(7 位、10 位地址或广播呼叫地址)可通过软件设置。

I^2C接口的功能框图示于 图 135。

图135 I^2C 的功能框图



注： 在SMBus 模式下，SMBALERT 是可选信号。如果 SMBus 被禁止，则该信号不可使用。

15.4功能描述

默认情况下，I^2C 接口总是工作在从模式。从默认的从模式切换到主模式，需要产生一个起始条件。

15.4.1 I2C从模式

为了产生正确的时序，必须在 I2C_CR2 寄存器中设定外设输入时钟。外设输入时钟的频率必须至少是：

- 标准模式下为：2MHz
- 快速模式下为：4MHz

一旦检测到起始条件，在 SDA 线上接收到的地址被送到移位寄存器。然后与芯片自己的地址 OAR1 和 OAR2(当 ENDUAL=1)或者广播呼叫地址(如果 ENGC=1)相比较。

注： 在 10-bit 地址模式时，比较包括头段序列(11110xx0)，其中的 xx 是地址的两个最高有效位。

头段或地址不匹配：接口将其忽视并等待另一个起始条件。

头段匹配(仅 10 位模式)：如果 ACK 位被置 1，接口产生一个应答脉冲并等待 8 位从地址。

地址匹配：接口产生以下时序：

- 如果 ACK 被置 1，则产生一个应答脉冲
- 硬件设置 ADDR 位，如果设置了 ITEVFEN 位，则产生一个中断
- 如果 ENDUAL=1，软件必须读 DUALF 位，以确认响应了哪个从地址。

在 10 位模式，接收到地址序列后，从设备总是处于接收器模式。在收到与地址匹配的头序列并且最低位为 1(即 11110xx1)后，当接收到重复的起始条件时，将进入发送器模式。

TAR 位在从模式下指示当前是处于接收器模式还是发送器模式。

从发送器

在接收到地址和清除 ADDR 位后，从发送器将字节从 DR 寄存器经由内部移位寄存器发送到 SDA 线上。

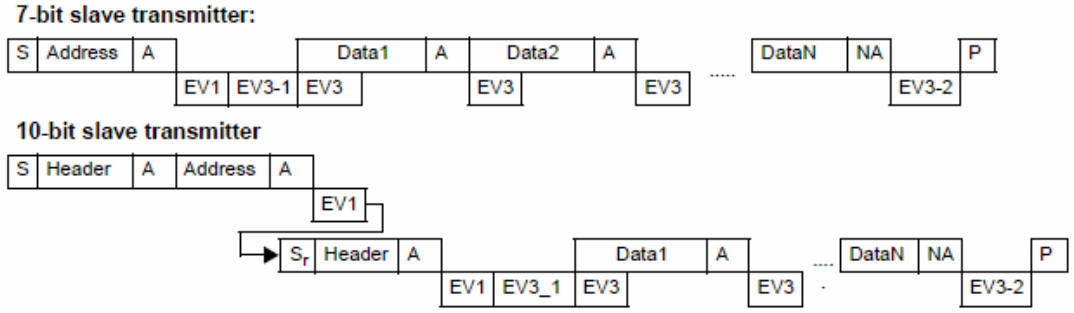
从设备保持SCL为低电平，直到ADDR位被清除并且待发送数据已写入DR寄存器。(见 图 136 中发送序列EV1 EV3)。

当收到应答脉冲时:

- TxE 位被硬件置位，如果设置了 ITEVFEN 和 ITBUFEN 位，则产生一个中断。

如果 TxE 位被置位，但在上一次数据发送结束之前没有数据写入到 DR 寄存器，则 BTF 位被置位，接口将保持 SCL 为低电平，以等待写入 DR 寄存器。

图136 从发送器的传送序列图



说明：S=Start(起始条件)，S_r=重复的起始条件，P=Stop(停止条件)，A=响应，NA=非响应，EVx=事件(ITEVFEN=1时产生中断)

EV1：ADDR=1，读SR1然后读SR2将清除该事件
EV3-1：TxE=1，移位寄存器空
EV3：TxE=1，读DR将清除该事件；移位寄存器非空
EV3-2：AF=1，在SR1寄存器的AF位写0可以清除AF位

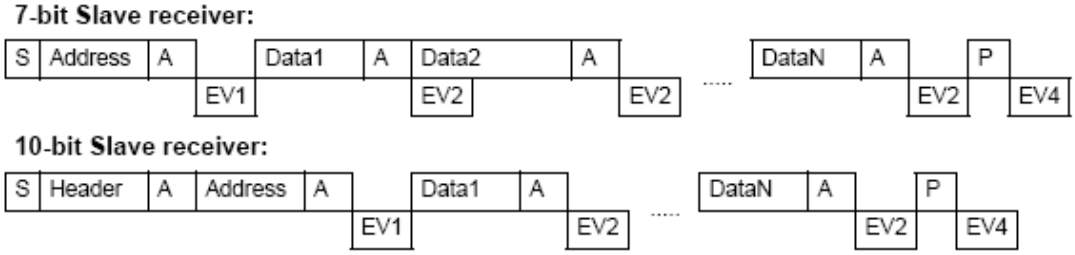
从接收器

在接收到地址和清除 ADDR 后，从接收器将通过内部移位寄存器从 SDA 线接收到的字节存进 DR 寄存器。接口在接收到每个字节后都生成以下时序:

- 如果设置了 ACK 位，则产生一个应答脉冲
- 硬件置位 RxNE。如果设置了 ITEVFEN 和 ITBUFEN 位，则产生一个中断。

如果RxNE被置位，并且在接收新的数据结束之前DR寄存器未被读出，BTF位被置位，接口保持SCL为低电平，等待读DR寄存器(见 图 137 传送序列)。

图137 从接收器的传送序列图



说明：S=Start(起始条件)，S_r=重复的起始条件，P=Stop(停止条件)，A=响应，NA=非响应，EVx=事件(ITEVFEN=1时产生中断)

EV1：ADDR=1，读SR1然后读SR2将清除该事件
EV2：RxNE=1，读DR将清除该事件
EV4：STOPF=1，读SR1然后写CR1寄存器将清除该事件

关闭从通信

在最后一个数据字节被发送后，主设备产生一个停止条件。 接口检测到这一条件时：

- STOPF 位被置位，如果设置了 ITEVFEN 位，则产生一个中断。

然后接口等待读SR1 寄存器，然后写CR1 寄存器。（见 图 137 传送时序 EV4）。

15.4.2 I2C主模式

在主模式时，I^2C 接口启动数据传输并产生时钟信号。串行数据传输总是以起始条件开始和以停止条件结束。当用 START 位在总线上产生了起始条件，设备就进入了主模式。

以下是主模式所要求的时序：

- 在 I2C_CR2 寄存器中设定外设时钟以产生正确的时序
- 配置时钟控制寄存器
- 配置上升时间寄存器
- 编程 I2C_CR2 寄存器启动外设
- 置 I2C_CR2 寄存器中的 START 位为 1，用于产生起始条件

外设输入时钟频率必须至少是:

- 标准模式下为：2MHz
- 快速模式下为：4MHz

起始条件

当 BUSY 位处于清除状态时对 START 位置位，使接口产生一个开始条件并切换到主模式(M/SL 位置位)。

注： 在主模式下，设置 START 位将在当前字节传输完后由硬件产生一个重开始条件。

一旦开始条件发出:

- SB 位被硬件置位，如果设置了 ITEVFEN 位，则会产生一个中断。

然后主设备等待读SR1 寄存器，紧跟着将从地址写入DR寄存器(见 图 138 和 图 139 传送时序EV5)。

从地址的发送

从地址通过内部移位寄存器被送到 SDA 线上。

- 在 10 位地址模式时，发送一个头段序列产生以下事件：
  - ADD10 位被硬件置位，如果设置了 ITEVFEN 位，则产生一个中断。
  - 然后主设备等待一次读SR1 寄存器，跟着将第二个地址字节写入DR寄存器(见 图 138 和 图 139 传送时序)。
  - ADDR 位被硬件置位，如果设置了 ITEVFEN 位，则产生一个中断。
  - 随后主设备等待一次读SR1 寄存器，跟着读SR2 寄存器(见 图 138 和 图 139 传送时序)。
- 在 7 位地址模式时，将送出一个地址字节。

  一旦该地址字节被送出，
  - ADDR 位被硬件置位，如果设置了 ITEVFEN 位，则产生一个中断。

随后主设备等待一次读SR1 寄存器，跟着读SR2 寄存器(见 图 138 和 图 139 传送时序)。

根据送出从地址的 LSB 位，主设备决定进入是发送器模式还是接收器模式。

- 在 7 位地址模式时，
  - 要进入发送器模式，主设备发送从地址时让 LSB 等于 0。
  - 要进入接收器模式，主设备发送从地址时让 LSB 等于 1。
- 在 10 位地址模式时
  - 要进入发送器模式，主设备先送头字节(11110xx0)，然后送 LSB 位等于 0 的从地址。（头段字节中的 xx 是 10 位地址中的最高 2 位。）
  - 要进入接收器模式，主设备先送头字节(11110xx0)，然后送 LSB 位等于 0 的从地址。然后再重新发送一个开始条件，后面跟着头字节(11110xx1)。（头字节中的 xx 是 10 位地址中的最高 2 位）。

TRA 位指示主设备是在接收器模式还是发送器模式。

主发送器

在发送了地址和清除了 ADDR 位后, 主设备通过内部移位寄存器将字节从 DR 寄存器发送到 SDA 线上。

主设备等到TxE被清除，(见 图 138 传送时序EV8)。

当收到应答脉冲时:

- TxE 位被硬件置位，如果设置了 INEVFEN 和 ITBUFEN 位，则产生一个中断。

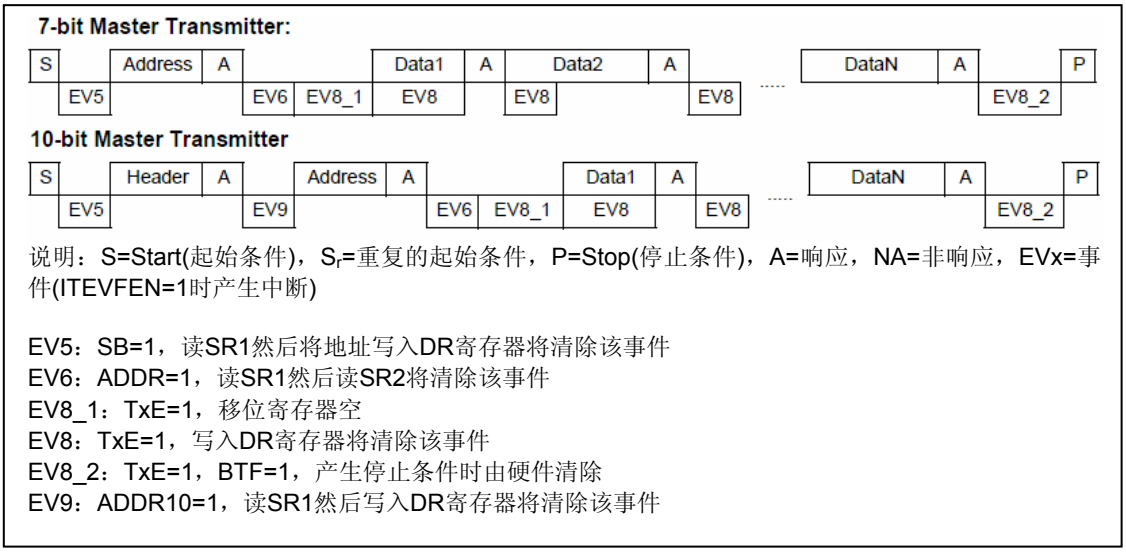
如果 TxE 被置位并且在上一次数据发送结束之前没有写数据字节到 DR，则 BTF 被置位，接口等待 BTF 被清除。

关闭通信

在DR寄存器中写入最后一个字节之后，通过将STOP位置位产生一个停止条件(见 图 138 传送时序EV8_2)。然后接口自动回到从模式(M/S位清除)。

注： 当 TxE 或 BTF 位置位时，停止条件应安排在出现 EV8_2 事件时。

图138 主发送器传送序列图



说明：S=Start(起始条件)，S_r=重复的起始条件，P=Stop(停止条件)，A=响应，NA=非响应，EVx=事件(ITEVFEN=1时产生中断)

EV5：SB=1，读SR1然后将地址写入DR寄存器将清除该事件
EV6：ADDR=1，读SR1然后读SR2将清除该事件
EV8_1：TxE=1，移位寄存器空
EV8：TxE=1，写入DR寄存器将清除该事件
EV8_2：TxE=1，BTF=1，产生停止条件时由硬件清除
EV9：ADDR10=1，读SR1然后写入DR寄存器将清除该事件

主接收器

在发送地址和清除 ADDR 之后，I^2C 接口进入主接收器模式。在此模式时，接口从 SDA 线接收数据字节，并通过内部移位寄存器将其送进 DR 寄存器。在每个字节后，接口依次生成以下事件：

- 如果 ACK 位被置位，发出一个应答脉冲。
- RxNE位被硬件置位，如果INEVFEN和ITBUFEN位被置起位，则会产生一个中断(见 图 139 传输时序EV7)。

如果 RxNE 位被置位，并且在上一次数据接收之后 DR 寄存器中的数据没有被读走，硬件将置起 BTF 位，接口等待读 DR 寄存器。

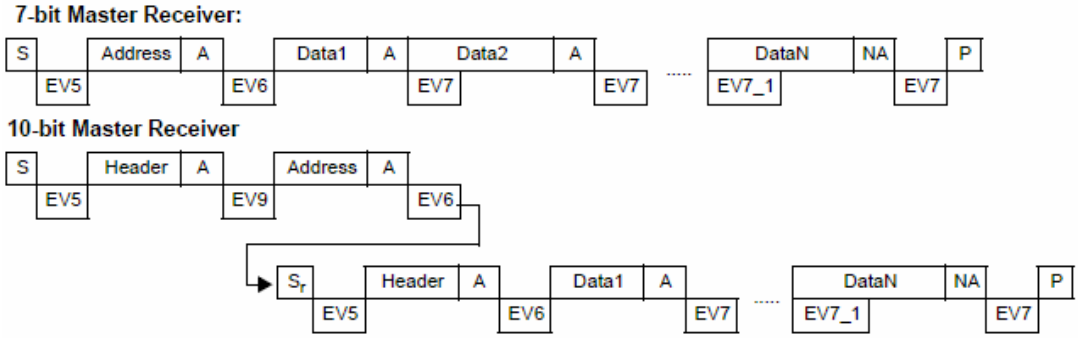
关闭通信

主设备在从从设备接收到的最后一个字节后发送一个 NACK。从设备接收到 NACK 后，释放对 SCL 和 SDA 线的控制。主设备就可以发送一个停止/重起始(Stop/Re-Start)条件。

- 为了在收到最后一个字节后产生一个 NACK 脉冲，在读倒数第二个数据字节之后(在倒数第二个 RxNE 事件之后)必须清除 ACK 位。
- 为了产生一个停止/重起始条件，软件必须在读倒数第二个数据字节之后(在倒数第二个 RxNE 事件之后)设置 STOP/START 位。

在产生了停止条件后，接口自动回到从模式(M/SL 位被清除)。

图139 主接收器传送序列图



说明：S=Start(起始条件)，S_r=重复的起始条件，P=Stop(停止条件)，A=响应，NA=非响应，EVx=事件(ITEVFEN=1时产生中断)

EV5：SB=1，读SR1然后将地址写入DR寄存器将清除该事件
EV6：ADDR=1，读SR1然后读SR2将清除该事件。在10位主接受模式下，该事件后应设置CR2的START=1。
EV7：RxNE=1，读DR寄存器清除该事件
EV7_1：RxNE=1，读DR寄存器清除该事件。设置ACK=0和STOP请求
EV9：ADDR10=1，读SR1然后写入DR寄存器将清除该事件

15.4.3 错误条件

以下条件可能造成通讯失败。

总线错误(BERR)

在一个字节传输期间，当 I2C 接口检测到一个停止或起始条件则产生此错误。此时：

- BERR 位被置位，如果设置了 ITERREN 位，则产生一个中断；
- 在从模式情况下，数据被丢弃，硬件释放总线：

- 在错误的开始条件情况下，从设备认为是一个重启动，并等待地址和停止条件。
- 在错误的停止条件情况下，从设备与正常停止表现得一样，同时硬件释放总线。

应答错误(AF)

当接口检测到一个无应答位时，产生此错误。此时：

- AF 位被置位，如果设置了 ITERREN 位，则产生一个中断；
- 当发送器接收到一个 NACK 时，必须复位通讯。
  - 如果是处于从模式，硬件释放总线
  - 如果是处于主模式，必须用软件生成一个停止条件

仲裁丢失(ARLO)

当 I2C 接口检测到一个仲裁丢失时产生此错误，此时：

- ARLO 位被硬件置位，如果设置了 ITERREN 位，则产生一个中断
- I2C 接口自动回到从模式(M/SL 位被清除)
- 硬件释放总线

过载/欠载错误(OVR)

在从模式下，时钟延长被禁止同时 I2C 接口正在接收数据时，当接口已经接收到一个字节(RxNE=1)，但在 DR 寄存器中上一个字节的数据还没有被读走，则可能发生过载错。此时：

- 上次接收的数据被丢弃
- 在过载错时，软件应清除 RxNE 位，发送器应该重新发送上一次发送的字节。

在从模式下，时钟延长被禁止同时 I2C 接口正在发送数据时，在下一个字节的时钟到达之前，下一个字节的数据还未写入更新 DR(TxE=1)，则可能欠载错。此时：

- 在 DR 寄存器中的上一个字节将被重发
- 用户应该确定在发生欠载错时，接收端应丢弃重复接收到的数据。发送端应按 I2C 总线标准在规定的时间更新 DR。

15.4.4 SDA/SCL线控制

- 如果允许时钟延长：

- 发送器模式：如果 TxE=1 且 BTF=1：接口在传输前保持时钟线路为低，以等待软件读取 SR1，然后把字节写进数据寄存器(缓冲器和移位寄存器都是空的)。
- 接收器模式：如果 RxNE=1 且 BTF=1：接口在接收到数据字节后保持时钟线为低，以等待软件读 SR1，然后读数据寄存器 DR(缓冲器和移位寄存器都是满的)。

- 如果在从模式中禁止时钟延长：
  - 如果 RxNE=1，在接收下个字节前 DR 还没有被读走，则发生过载错。接收到的最后一个字节丢失。
  - 如果 TxE=1，在下个字节必须发送之前却没有字节写进 DR，则发生欠载错。相同的字节将被重发。
  - 无写冲突管理

15.4.5 SMBus

介绍

系统管理总线(SMBus)是一个两线接口。通过它，各设备之间以及设备与系统的其他部分之间可以互相通信。它基于 I2C 操作原理。SMBus 为系统和电源管理相关的任务提供一条控制总线。一个系统利用 SMBus 可以和多个设备互传信息，而不需使用独立的控制线路。

系统管理总线(SMBus)标准涉及三类设备。*从设备*，接收或响应命令的设备。*主设备*，用来发布命令，产生时钟和终止发送的设备。*主机*，是一种专用的主设备，它提供与系统 CPU 的主接口。主机必须具有主-从机功能，并且必须支持 SMBus 通报协议。在一个系统里只允许有一个主机。

SMBus和I2C之间的相似点

- 2 条线的总线协议(1 个时钟，1 个数据) + 可选的 SMBus 提醒线
- 主-从通信，主设备提供时钟
- 多主机功能
- SMBus数据格式类似于I2C的 7 位地址格式（见 图 133）

SMBus和I2C之间的不同点

下表为 SMBus 和 I2C 的不同点。

表45 SMBus 与 I2C 的比较

SMBus	I^2C
最大传输速度100kHz	最大传输速度400kHz

最小传输速度10kHz	无最小传输速度
35ms时钟低超时	无时钟超时
固定的逻辑电平	逻辑电平由VDD决定
不同的地址类型(保留、动态等)	7位、10位和广播呼叫从地址类型
不同的总线协议（快速命令、处理呼叫等)	无总线协议

SMBus应用用途

利用系统管理总线，设备可提供制造商信息，告诉系统它的型号/部件号，保存暂停事件的状态，报告不同类型的错误，接收控制参数，和返回它的状态。SMBus为系统和电源管理相关的任务提供控制总线。

设备标识

在系统管理总线上，任何一个作为从模式的设备都有一个唯一的地址，叫做从地址。保留的从地址表请参考 2.0 版的SMBus规范(*http://smbus.org/specs/*)。

总线协议

SMBus技术规范支持 9 个总线协议。有关这些协议的详细资料和SMBus地址类型，请参考 2.0 版的SMBus规范(*http://smbus.org/specs/*)。这些协议由用户的软件来执行。

地址解析协议(ARP)

SMBus 从地址冲突可以通过给每个从设备动态分配一个新的唯一地址来解决。ARP 有以下的属性：

- 地址分配利用标准 SMBus 物理层仲裁机制
- 当设备维持供电期间，分配的地址仍保持不变，允许设备在断电时保留其地址。
- 在地址分配后，没有额外的 SMBus 的打包开销(也就是说访问分配地址的设备与访问固定地址的设备所用时间是一样的）。
- 任何一个 SMBus 主设备可以遍历总线。

唯一的设备标识符(UDID)

为了提供一种为进行地址分配目的而区分每个设备的机制，每个设备必须拥有一个唯一的设备标识符。

关于在ARP上 128 位的UDID细节的信息，参考 2.0 版的SMBus规范(*http://smbus.org/specs/*)。

SMBus提醒模式

SMBus 提醒是一个带中断线的可选信号，用于那些希望扩展他们的控制能力而牺牲一个引脚的设备。SMBALERT 和 SCL 和 SDA 信号一样，是一种线与信号。SMBALERT 通常和 SMBus 广播呼叫地址一起使用。与 SMBus 有关的消息为 2 字节。

单一的从设备可以通过 SMBALERT 发信号给主机表示它希望进行通信，这可通过设置 I2C_CR1 寄存器上的 ALERT 位实现。主机处理该中断并通过提醒响应地址 ARA(*Alert Response Address*，地址值为 0001100x)访问所有 SMBALERT 设备。只有那些将 SMBALERT 拉低的设备能应答 ARA。此状态是由 I2C_SR1 寄存器中的 SMBALERT 状态标记来标识的。主机执行一个修改过的接收字节操作。由从发送设备提供的 7 位设备地址被放在字节的 7 个最高位上，第八个位可以是 0 或 1。

如果多个设备把 SMBALERT 拉低，最高优先级设备(最小的地址)将在地址传输期间通过标准仲裁赢得通信权。在确认从地址后，此设备不得再拉低它的 SMBALERT，如果当信息传输完成后，主机仍看到 SMBALERT 低，就知道需要再次读 ARA。

没有执行SMBALERT信号的主机可以定期访问ARA。有关SMBus提醒模式的更多详细资料，请参考 2.0 版的SMBus规范(*http://smbus.org/specs/*)。

超时错误

在定时规范上 I2C 和 SMBus 之间有很多差别。

SMBus定义一个时钟低超时，35ms的超时。SMBus规定TLOW:SEXT为从设备的累积时钟低扩展时间。SMBus规定TLOW:MEXT为主设备的累积时钟低扩展时间。更多超时细节请参考 2.0 版的SMBus规范(*http://smbus.org/specs/*)。

I2C_SR1 中的状态标志 Timeout 或 Tlow 错误表明了这个特征的状态。

如何使用SMBus模式的接口

为了从 I2C 模式切换到 SMBus 模式，应该执行下列步骤：

- 设置 I2C_CR1 寄存器中的 SMBus 位
- 按应用要求配置 I2C_CR1 寄存器中的 SMBTYPE 和 ENARP 位。

如果你想把设备配置成主设备，产生起始条件的步骤见 15.4.2 I2C主模式。否则，参见 15.4.1 I2C从模式。

软件程序必须处理多种 SMBus 协议。

- 如果 ENARP=1 且 SMBTYPE=0，使用 SMB 设备默认地址。
- 如果 ENARP=1 且 SMBTYPE=1，使用 SMB 主设备头字段。
- 如果 SMBALERT=1，使用 SMB 提醒响应地址。

15.4.6 DMA请求

DMA 请求(当被使能时)仅用于数据传输。发送时数据寄存器变空或接收时数据寄存器变满，则产生 DMA 请求。当为相应 DMA 通道设置的数据传输量已经完成时，DMA 控制器发送传输结束信号 ETO 到 I2C 接口，并且在中断允许时产生一个传输完成中断：

- 主发送器：在 EOT 中断服务程序中，需禁止 DMA 请求，然后在等到 BTF 事件后设置停止条件。
- 主接收器：DMA 控制器发送一个硬件信号 EOT_1，它对应 DMA 传输(字节数－1)。如果在 I2C_CR2 寄存器中设置了 LAST 位，硬件在发送完 EOT_1 后的下一个字节，将自动发送 NACK。在中断允许的情况下，用户可以在 DMA 传输完成的中断服务程序中产生一个停止条件。

注： 请参考产品手册以确认您所选用型号有 DMA 控制器。如果 DMA 不可用，用户应该如前面所描述的方法使用 I2C。在 I2C ISR 中，可以清除 TxE/RxNE 标记以达到连续的通信。

利用DMA发送

DMA 模式可以通过设置 I2C_CR2 寄存器中的 DMAEN 位来激活。在接收到地址序列后并清除了 ADDR 位，才能设置 DMAEN 位。只要 TxE 位被置位，数据将由 DMA 从预置的存储区装载进 I2C_DR 寄存器。为 I2C 分配一个 DMA 通道，须执行以下步骤。(x 是通道号)

1. 在DMA_CPARx寄存器中设置I2C_DR寄存器地址。数据将在每个TxE事件后从存储器送进这个地址。
2. 在DMA_CMARx寄存器中设置存储器地址。数据在每个TxE事件后从这个存储区装载进I2C_DR。
3. 在DMA_CNDTRx寄存器中设置所需的传输字节数。在每个TxE事件后，此值将被递减。
4. 利用DMA_CCRx寄存器中的PL[0:1]位配置通道优先级。
5. 设置DMA_CCRx寄存器中的DIR位，并根据应用要求可以配置在整个传输完成一半或全部完成时发出中断请求。
6. 通过设置DMA_CCTx寄存器上的EN位激活通道。

当 DMA 控制器中设置的数据传输数目已经完成时，DMA 控制器给 I2C 接口发送一个传输结束的 EOT/ EOT_1 信号。在中断允许的情况下，将产生一个 DMA 中断。

注： 如果使用DMA进行发送时，不要设置I2C_CR2寄存器的ITEVTEN位。

利用DMA接收

DMA 接收模式可以通过设置 I2C_CR2 寄存器中的 DMAEN 位激活。只有在接收到地址序列后并且 ADDR 为被清除后，才能设置 DMAEN 位。接收到数据字节时，数据将由 DMA 从 I2C_DR 寄存器传送到设置的存储区(参考 DMA 说明)。设置 DMA 通道进行 I2C 接收，须执行以下步骤。(x 是通道号)

1. 在DMA_CPARx寄存器中设置I2C_DR寄存器的地址。数据将在每次RxNE事件后从此地址传送到存储区。
2. 在DMA_CMARx寄存器中设置存储区地址。数据将在每次RxNE事件后从I2C_DR寄存器传送到此存储区。
3. 在DMA_CNDTRx寄存器中设置所需的传输字节数。在每个RxNE事件后，此值将被递减。
4. 用DMA_CCRx寄存器中的PL[0:1]配置通道优先级。
5. 清除DMA_CCRx寄存器中的DIR位，根据应用要求可以设置在数据传输完成一半或全部完成时发出中断请求。
6. 设置DMA_CCRx寄存器中的EN位激活该通道。

当 DMA 控制器中设置的数据传输数目已经完成时，DMA 控制器给 I2C 接口发送一个传输结束的 EOT/ EOT_1 信号。在中断允许的情况下，将产生一个 DMA 中断。

注： 如果使用 DMA 进行接收时，不要设置 I2C_CR2 寄存器的 ITEVTEN 位。

15.4.7 包错误校验(PEC)

包错误校验(PEC)计算器是用于提高通信的可靠性，这个计算器使用一个可编程的多项式对每一位串行数据进行计算。

- PEC 计算由 I2C_CR1 寄存器的 ENPEC 位激活。PEC 使用 CRC-8 算法对所有信息字节进行计算，包括地址和读/写位在内。
  - 在发送时：在最后一个 TxE 事件时设置 I2C_CR1 寄存器中的 PEC 传输位，PEC 将在当前字节后被发送。
  - 在接收时：在最后一个 RxNE 事件时设置 I2C_CR1 寄存器中的 PEC 位，如果下个接收到的字节不等于内部计算的 PEC，接收器发送一个 NACK。如果是主接收器，不管校对的结果如何， PEC 后都将发送 NACK。
- 在 I2C_SR1 寄存器中可获得 PECERR 错误标记/中断。
- 如果 DMA 和 PEC 计算器都被激活：
  - 在发送时：当 I2C 接口从 DMA 控制器处接收到 EOT 信号时，它在最后一个字节后自动发送 PEC。
  - 在接收时：当 I2C 接口从 DMA 处接收到一个 EOT_1 信号时，它将自动把下一个字节作为 PEC，并且将检查它。在接收到 PEC 后产生一个 DMA 请求。
- 为了允许中间 PEC 传输，在 I2C_CR2 寄存器中有一个控制位(LAST 位)用于判别是否真是最后一个 DMA 传输。如果确实是最后一个主接收器的 DMA 请求，在接收到最后一个字节后自动发送 NACK。
- 仲裁丢失时 PEC 计算失效。

15.5 中断请求

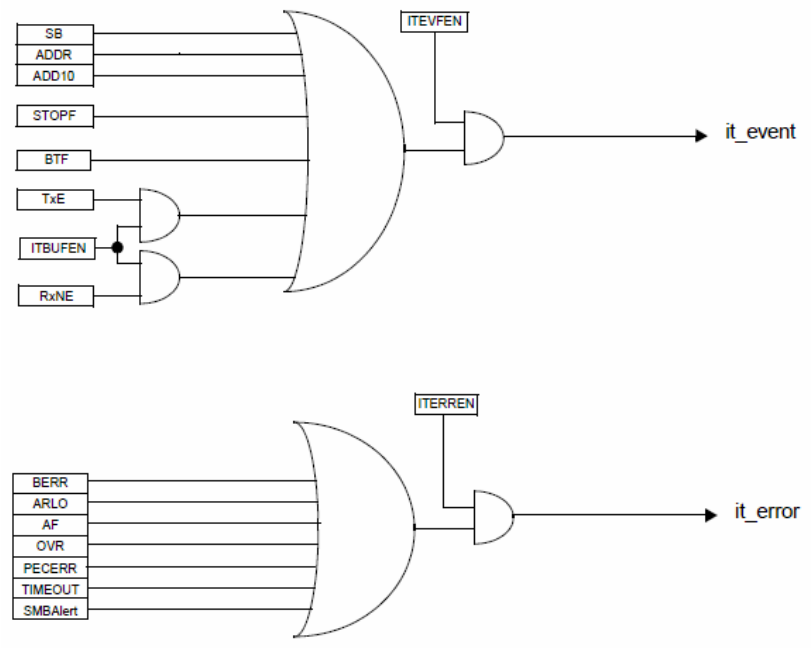
表46 I^2C 中断请求表：

中断事件	事件标志	开启控制位
起始位已发送(主)	SB	ITEVFEN
地址已发送(主) 或 地址匹配(从)	ADDR	
10位头段已发送(主)	ADD10	
已收到停止(从)	STOPF	
数据字节传输完成	BTF	
接收缓冲区非空	RxNE	ITEVFEN 和 ITBUFEN
发送缓冲区空	TxE	
总线错误	BERR	ITERREN
仲裁丢失(主)	ARLO	
响应失败	AF	
过载/欠载	OVR	
PEC错误	PECERR	
超时/Tlow错误	TIMEOUT	
SMBus提醒	SMBALERT	

注： 1 SB、ADDR、ADD10、STOPF、BTF、RxNE、TxE 通过逻辑或汇到同一个中断通道中。

2 BERR、ARLO、AF、OVR、PECERR、TIMEOUT、SMBALERT 通过逻辑或汇到同一个中断通道中。

图140 I^2C 中断映射图



15.6 I2C调试模式

当微控制器进入调试模式(Cortex-M3 核心处于停止状态)时，根据 DBG 模块中的 DBG_I2Cx_SMBUS_TIMEOUT 配置位，SMBUS 超时控制或者继续正常工作或者可以停止。详见后面的章节。

15.7 I2C寄存器描述

关于在寄存器描述里面所用到的缩写，详见第 1 章。

15.7.1 控制寄存器 1(I2C_CR1)

地址偏移：0x00
复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWRST	保留	ALERT	PEC	POS	ACK	STOP	START	NO STRETCH	ENGC	ENPEC	ENARP	SMB TYPE	保留	SMBUS	PE
rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw

位15	**SWRST**：*软件复位* 当被置位时，I2C处于复位状态。在复位该位前确信I2C的引脚被释放，总线是空的。 0：I^2C外设不处于复位状态； 1: I^2C外设处于复位状态。 ***注：****当在总线上没有检测不到停止条件时，该位可以用于BUSY位被置位的情况。*
位14	保留位，硬件强制为0
位13	**ALERT**：SMBus提醒 软件可以设置或清除该位；当PE=0时，由硬件清除。 0：释放SMBAlert引脚使其变高。提醒响应地址头紧跟在NACK信号后面。 1：驱动SMBAlert引脚使其变低。提醒响应地址头紧跟在ACK信号后面。
位12	**PEC**：数据包出错检测 软件可以设置或清除该位；当传送PEC后，或起始或停止条件时，或当PE=0时硬件将其清除。 0：无PEC传输 1：PEC传输(在发送或接收模式) **注**：PEC的计算可能因为仲裁的丢失而出现混乱。

位11	**POS**: 应答/PEC位置（用于数据接收）。 软件可以设置或清除该位，或当PE=0时，由硬件清除。 0：ACK位控制当前移位寄存器内正在接收的字节的(N)ACK。 PEC位表明当前移位寄存器内的字节是PEC。 1: ACK位控制在移位寄存器里接收的下一个字节的(N)ACK。PEC位表明在移位寄存器里接收的下一个字节是PEC。 **注：** 该位必须在数据接收开始之前设置。 该设置必须只用在地址延长事件中以防只有2个数据字节。
位10	**ACK**：应答使能 软件可以设置或清除该位，或当PE=0时，由硬件清除。 0：无应答返回； 1：在接收到一个字节后返回一个应答(匹配的地址或数据)
位9	**STOP**：停止条件产生 软件可以设置或清除该位；或当检测到停止条件时，由硬件清除；当检测到超时错误时，硬件将其置位。 在主模式下： 0：无停止条件产生； 1：在当前字节传输或在当前起始条件发出后产生停止条件。 在从模式下： 0：无停止条件产生； 1：在当前字节传输或释放SCL和SDA线。 **注**：在主模式下，当需要停止条件时，必须清除I2C_SR1寄存器中的BTF位。
位8	**START**：起始条件产生 软件可以设置或清除该位，或当起始条件发出后或PE=0时，由硬件清除。 在主模式下： 0: 无起始条件产生； 1：重复产生起始条件； 在从模式下： 0：无起始条件产生； 1：当总线空闲时，产生起始条件。
位7	**NOSTRETCH**：禁止时钟延长(从模式) 该位用于当ADDR或BTF标志被置位，在从模式下禁止时钟延长，直到它被软件复位。 0：允许时钟延长 1：禁止时钟延长
位6	**ENGC**：广播呼叫使能 0：禁止广播呼叫。以非应答响应地址00h。 1：允许广播呼叫. 以应答响应地址00h。
位5	**ENPEC**：PEC使能 0：禁止PEC计算； 1：开启PEC计算。
位4	**ENARP**：ARP使能 0：禁止ARP； 1：使能ARP。 如果SMBTYPE=0，使用SMBus设备的默认地址。 如果SMBTYPE=1，使用SMBus的主地址。

位3	**SMBTYPE**：SMBus类型 0：SMBus设备 1：SMBus主机
位2	保留位，硬件强制为0
位1	**SMBUS**：SMBus模式 0：I2C模式 1：SMBus模式
位0	**PE**：I2C外设使能 0：禁用I2C外设 1：启用I2C外设：根据SMBus位的选用，相应的I/O口执行响应的复用功能。 **注：**如果复位该位时通讯正在进行，在当前通讯结束后，I2C外设被禁用并返回空闲状态。 由于在通讯结束后发生PE＝0，所有的位被复位。 在主模式下，通讯结束之前，绝不能复位该位。

15.7.2 控制寄存器 2(I2C_CR2)

地址偏移：0x04
复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留			LAST	DMAEN	ITBUFEN	ITEVTEN	ITERREN	保留		FREQ[5:0]					
			rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw

位15:13	保留位，硬件强制为0
位12	**LAST**: DMA最后一次传输 0：下一次DMA的EOT不是最后的传输； 1：下一次DMA的EOT是最后的传输。 **注：**该位在主接收模式使用， 使得在最后一次接收数据时可以产生一个NACK。
位11	**DMAEN**：DMA请求使能 0：禁止DMA请求； 1：当TxE=1或RxNE =1时，允许DMA请求。 **注：**只有在收到地址序列，清除了ADDR位后，才能设置DMAEN位。
位10	**ITBUFEN**：缓冲器中断使能 0：当TxE=1或RxNE=1时，不产生任何中断； 1：当TxE=1或RxNE=1时，产生事件中断(不管DMAEN是何种状态)。

位9	**ITEVTEN**：事件中断使能 0：禁止事件中断； 1: 允许事件中断； 在下列条件下，将产生该中断： – SB = 1 (主模式)； – ADDR = 1 (主/从模式)； – ADD10= 1 (主模式)； – STOPF = 1 (从模式)； – BTF = 1，但是没有TxE或RxNE事件； – 如果ITBUFEN = 1，TxE事件为1； – 如果ITBUFEN = 1，RxNE事件为1。
位8	**ITERREN**：出错中断使能 0：禁止出错中断； 1：允许出错中断。 在下列条件下，将产生该中断： – BERR = 1 – ARLO = 1 – AF = 1 – OVR = 1 – PECERR = 1 – TIMEOUT = 1 – SMBAlert = 1
位7:6	保留位，硬件强制为0
位6:0	FREQ[5:0]: 外设时钟频率 必须设置正确的输入时钟频率以产生正确的时序，允许的范围在2～50MHz之间： 000000：禁止 000001：禁止 000010：2 MHz ... 110010：50 MHz 大于110010：禁止

15.7.3 自身地址寄存器 1 (I2C_OAR1)

复位地址偏移：0x08
复位值：0x0000

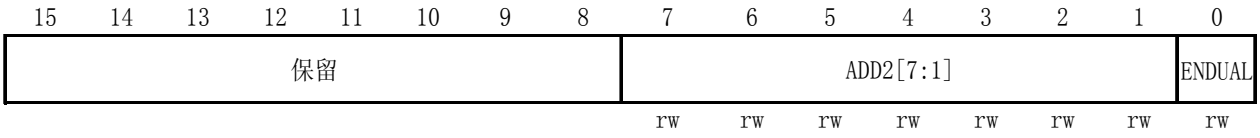
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADD MODE	保留	保留				ADD[9:8]		ADD[7:1]							ADD0
rw	rw					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位15	**ADDMODE** 地址模式（从模式） 0：7位从地址（10位地址不被识别) 1: 10位从地址（7位地址不被识别)
位14	必须设置并保持为1
位13:10	保留位，硬件强制为0

位9:8	ADD[9:8]：接口地址 7位地址模式时不用关心 10位地址模式时为地址的9~8位
位7:1	ADD[7:1]：接口地址 地址的7~1位
位0	**ADD0**：接口地址 7位地址模式时不用关心 10位地址模式时为地址第0位

15.7.4 自身地址寄存器 2(I2C_OAR2)

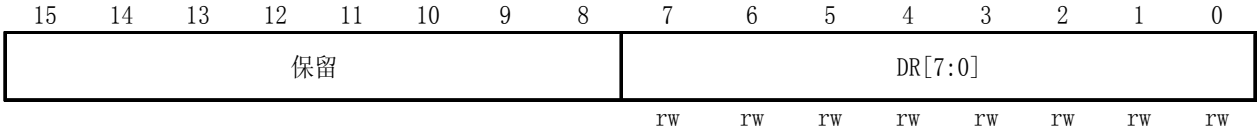
地址偏移：0x0C
复位值：0x0000



位15:8	保留位，硬件强制为0
位7:1	ADD2[7:1]：接口地址 在双地址模式下地址的7~1位。
位0	**ENDUAL**：双地址模式使能位 0: 在7位地址模式下，只有OAR1被识别； 1：在7位地址模式下，OAR1和OAR2都被识别。

15.7.5 数据寄存器(I2C_DR)

地址偏移：0x10
复位值：0x0000



位15:8	保留位，硬件强制为0
位7:0	**DR[7:0]**：8位数据寄存器(1)(2)(3) 用于存放接收到的数据或放置用于发送到总线的数据 发送器模式：当一个字节被写入DR寄存器，数据的传输自动启动。一旦传输开始(TxE=1)，如果能及时把下一个需传输的数据写入DR寄存器，I2C外设将保持持连续的数据流。 接收器模式：接收到的字节被拷贝到DR寄存器(RxNE=1)。在接收到下一个字节之前，必须读出数据寄存器内已收到的数据，否则已收到的数据将被后面收到的数据覆盖而丢失。

1. 在从模式下，地址不会被拷贝进数据寄存器；
2. 硬件不管理写冲突(如果TxE=0，仍能写入数据寄存器)

3. 如果在处理ACK脉冲时ARLO事件发生，接收到的字节不会被拷贝到数据寄存器里，因此不能读到它。

15.7.6 状态寄存器 1(I2C_SR1)

地址偏移：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	保留	PEC ERR	OVR	AF	ARLO	BERR	TxE	RxNE	保留	STOPF	ADD10	BTF	ADDR	SB
rc	rc	rc	rc	rc	rc	rc	rc	r	r		r	r	r	r	r

位15	SMBALERT：SMBus提醒 在SMBus主机模式下： 0: 无SMBus提醒； 1: 在引脚上产生SMBAlert提醒事件。 在SMBus从机模式下： 0: 没有SMBAlert响应地址头序列 1: 收到SMBAlert响应地址头序列至SMBAlert变低
位14	TIMEOUT: 超时或Tlow错误 0: 无超时错误； 1: SCL处于低已达到25ms(超时)；或者主机低电平累积时钟扩展时间超过10ms(Tlow:mext)；或从设备低电平累积时钟扩展时间超过25ms(Tlow:sext) 当在从模式下设置该位：从设备复位通讯，硬件释放总线。 当在主模式下设置该位：硬件发出停止条件。 软件可以通过写0清除该位，或PE=0时，由硬件清除。
位13	保留位，硬件强制为0
位12	**PECERR**: 在接收时发生PEC错误 0: 无PEC错误：接收到PEC后接收器返回ACK(如果ACK=1) 1: 有PEC错误：接收到PEC后接收器返回NACK(不管ACK是什么值) 软件可以通过写0清除该位，或PE=0时，由硬件清除。
位11	**OVR**：过载/欠载 0: 无过载/欠载 1: 出现过载/欠载 当NOSTRETCH=1时，在从模式下该位被硬件置位，同时： – 在接收模式中当收到一个新的字节时(包括应答脉冲)，数据寄存器里的内容还未被读出，则新接收的字节将丢失。 – 在发送模式中当要发送一个新的字节时，却没有新的数据写入数据寄存器，同样的字节将被发送两次。 – 软件可以通过写0清除该位，或PE=0时，由硬件清除。 **注**：如果数据寄存器的写操作发生时间非常接近SCL的上升沿，发送的数据是不确定的，并发生锁定延时错误。

位10	**AF**：应答失败 0: 没有应答失败； 1: 应答失败 – 当没有返回应答时，硬件将置该位为1。 – 软件可以通过写0清除该位，或PE=0时，由硬件清除。
位9	**ARLO**: 仲裁丢失(主模式) 0：没有检测到仲裁丢失 1：检测到仲裁丢失 当接口失去对总线的控制给另一个主机时，硬件将将置该位为1。 – 软件可以通过写0清除该位，或PE=0时，由硬件清除。 在ARLO事件之后，接口自动切换回从模式(M/SL=0)。 **注**：在SMBUS模式下，在从模式下对数据的仲裁仅仅发生在数据阶段，或应答传输区间(不包括地址的应答)。
位8	**BERR**: 总线出错 0：无起始或停止条件出错； 1：起始或停止条件出错。 – 当接口检测到起始或停止条件出错，硬件将该位置1。 – 软件可以通过写0清除该位，或PE=0时，由硬件清除。
位7	**TxE**: 数据寄存器为空（发送时） 0: 数据寄存器非空 1: 数据寄存器空 – 在发送数据时，数据寄存器为空时该位被置1，在发送地址阶段不设置该位。 – 软件可以通过写数据到DR寄存器清除该位；或在发生一个起始或停止条件后或当PE=0时由硬件自动清除。 如果收到一个NACK,或下一个要发送的字节是PEC（PEC=1），该位不被置位。
位6	**RxNE**: 数据寄存器非空（接收时） 0: 数据寄存器为空 1: 数据寄存器非空 – 在接收时，当数据寄存器不为空，该位被置1。在接收地址阶段，该位不被置位。 – 软件可以通过对数据寄存器的读写操作清除该位，或当PE=0时由硬件清除。 在发生ARLO事件时，RxNE不被置位。
位5	保留位，硬件强制为0
位4	**STOPF**: 停止条件检测位（从模式） 0：没有检测到停止条件； 1：检测到停止条件。 – 在一个应答之后(如果ACK=1)，当从设备在总线上检测到停止条件时，硬件将该位置1。 – 软件读取SR1寄存器后，对CR1寄存器的写操作将清除该位，或当PE=0时，硬件清除该位。 **注**：在收到NACK后，STOPF位不被置位。
位3	**ADD10**: 10位头序列已发送（主模式） 0: 没有ADD10事件发生； 1: 主设备已经将地址的第一部分发送出去。 – 在10位地址模式下，当主设备已经将第一个字节发送出去时，硬件将该位置1。 – 软件读取SR1寄存器后，对CR1寄存器的写操作将清除该位，或当PE=0时，硬件清除该位。 **注**：收到一个NACK后，ADD10位不被置位。

位2	**BTF**：字节发送结束 0: 字节发送未完成； 1: 字节发送结束。 当NOSTRETCH=0时，在下列情况下硬件将该位置1： – 在接收时，当收到一个新字节(包括ACK脉冲)且数据寄存器还未被读取(RxNE=1)。 – 在发送时，当一个新数据将被发送且数据寄存器还未被写入新的数据(TxE=1)。 在软件读取SR1寄存器后，对数据寄存器的读或写操作将清除该位；或在传输中发送一个起始或停止条件后或当PE=0时，由硬件清除该位。 **注**：在收到一个NACK后，BTF位不会被置位。 如果下一个要传输的字节是PEC(I2C_SR2寄存器中TRA为1，同时I2C_CR1寄存器中PEC为1)，BTF位不会被置位。
位1	**ADDR**：地址已被发送（主模式)/地址匹配（从模式） 在软件读取SR1寄存器后，对SR2寄存器的读操作将清除该位，或当PE=0时，由硬件清除该位。 **地址匹配（从模式）** 0: 地址不匹配或没有收到地址； 1: 收到的地址匹配。 当收到的从地址与OAR寄存器中的内容相匹配、或发生广播呼叫、或SMBus设备默认地址或SMBus主机识别出SMBus提醒时，硬件就将该位置1(当对应的设置被使能时)。 **地址已被发送（主模式）** 0: 地址发送没有结束； 1: 地址发送结束。 – 10位地址模式时，当收到地址的第二个字节的ACK后该位被置1。 – 7位地址模式时，当收到地址的ACK后该位被置位。 **注**：在收到NACK后，ADDR位不会被置位。
位0	**SB**：起始位（主模式） 0：未发送起始条件； 1：起始条件已发送。 – 当发送出起始条件时该位被置1。 – 软件读取SR1寄存器后，对数据寄存器的写操作将清除该位，或当PE=0时，硬件清除该位。

15.7.7 状态寄存器 2 (I2C_SR2)

地址偏移：0x18
复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PEC[7:0]								DUALF	SMB HOST	SMB DEFAULT	GEN CALL	保留	TRA	BUSY	MSL
r	r	r	r	r	r	r	r	r	r	r	r		r	r	r

位15:8	PEC[7:0]：数据包出错检测 当ENPEC=1时，PEC[7:0]存放内部的PEC的值。

位7	DUALF：双标志（从模式） 0: 接收到的地址与OAR1内的内容相匹配； 1: 接收到的地址与OAR2内的内容相匹配。 在产生一个停止条件或一个重复的起始条件时或PE=0时，硬件将该位清除。
位6	**SMBHOST**: SMBus主机头系列（从模式） 0：未收到SMBus主机的地址； 1：当SMBTYPE=1且ENARP=1时，收到SMBus主机地址。 在产生一个停止条件或一个重复的起始条件时或PE=0时，硬件将该位清除。
位5	**SMBDEFAULT**：MBus设备默认地址（从模式） 0：未收到SMBus设备的默认地址； 1：当ENARP=1时，收到SMBus设备的默认地址。 在产生一个停止条件或一个重复的起始条件时或PE=0时，硬件将该位清除。
位4	**GENCALL**：广播呼叫地址（从模式） 0: 未收到广播呼叫地址； 1: 当ENGC=1时，收到广播呼叫的地址。 在产生一个停止条件或一个重复的起始条件时或PE=0时，硬件将该位清除。
位3	保留位，硬件强制为0
位2	**TRA**：发送/接收 0: 接收到数据； 1: 数据已发送； 在整个地址传输阶段的结尾，根据地址字节的R/W位来设定。 在检测到停止条件(STOPF=1)、重复的起始条件或总线仲裁丢失(ARLO=1)后，或当PE=0时，硬件将其清除。
位1	**BUSY**：总线忙 0：在总线上无数据通讯； 1: 在总线上正在进行数据通讯。 – 在检测到SDA或SCl为低电平时，硬件将该位置1； – 当检测到一个停止条件时，硬件将该位清除。 该位用于指示当前正在进行的总线通讯，当接口被禁用(PE=0)时该信息仍然被更新。
位0	**MSL**：主从模式 0：从模式 1：主模式 – 当接口处于主模式(SB=1)时，硬件将该位置位； – 当总线上检测到一个停止条件、仲裁丢失(ARLO=1时)、或当PE=0时，硬件将该位清除。

15.7.8 时钟控制寄存器(I2C_CCR)

地址偏移：0x1C
复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
F/S	DUTY	保留		CCR[11:0]											
rw	rw			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位15	**F/S**：I2C主模式选项 0：标准模式的I^2C 1：快速模式的I^2C
位14	**DUTY**：快速模式时的占空比 0：快速模式下：T_{low}/T_{high} = 2 1: 快速模式下：T_{low}/T_{high} = 16/9(见CCR)
位13:12	保留位，硬件强制为0
位11:0	**CCR[11:0]**：快速/标准模式下的时钟控制分频系数（主模式） 该分频系数用于设置主模式下的SCL时钟。 在I^2C标准模式或SMBus模式下： T_{high} = CCR × TCK T_{low} = CCR × TCK 在I^2C快速模式下： 如果DUTY = 0： T_{high} = CCR × TCK T_{low} = 2 × CCR × TCK 如果DUTY = 1：（速度达到400kHz） T_{high} = 9 × CCR × TCK T_{low} = 16 × CCR × TCK 例如：在标准模式下，产生100kHz的SCL的频率： 如果FREQR = 08，Tck = 125ns，则CCR必须写入28h（28h：40×125ns = 5000 ns）。 **注**：1、允许设定的最小值为04h，在快速模式下允许的最小值为01h； 2. T_{high}包含SCL的上升边沿； 3. T_{low}包含SCL的下降边沿； 4、这些延时没有过滤器。

15.7.9 TRISE寄存器(I2C_TRISE)

地址偏移：0x20
复位值：0x0002

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

位15:6	保留位，硬件强制为0
位5:0	TRISE[5:0]：在快速/标准模式下的最大上升时间（主模式） 这些位必须使用I^2C总线规范里给出的最大的SCL上升时间来填充，增长步幅为1。 例如：标准模式中最大允许SCL上升时间为1000ns。如果在I2C_CR2寄存器中FREQ[5:0]中的值等于08h且tCK=125ns，故TRISE[5:0]中必须写入09h(1000ns/125 ns = 8+1)。 滤波器的值也可以加到TRISE[5:0]内。 如果结果不是一个整数，则将整数部分写入TRISE[5:0]以确保tHIGH参数。 注：只有当I^2C被禁用(PE=0)时，才能设置TRISE[5:0]。

15.8I2C寄存器地址映象

表47 I^2C 寄存器地址映象和复位值

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
000h	I2C_CR1	保留																SWRST	保留	ALERT	PEC	POS	ACK	STOP	START	NOSTRETCH	ENGC	ENPEC	ENARP	SMBTYPE	保留	SMBUS	PE
	复位值																	0		0	0	0	0	0	0	0	0	0	0	0		0	0
004h	I2C_CR2	保留																				LAST	DMAEN	ITBUFEN	ITEVTEN	ITERREN	保留	FREQ[5:0]					
	复位值																					0	0	0	0	0		0	0	0	0	0	0
008h	I2C_OAR1	保留																ADDMODE	保留	保留						ADD [9:8]		ADD[7:1]					ADD0
	复位值																	0	1							0	0	0	0	0	0	0	0
00Ch	I2C_OAR2	保留																										ADD2[7:1]					ENDUAL
	复位值																									0	0	0	0	0	0	0	0
010h	I2C_DR	保留																										DR[7:0]					
	复位值																									0	0	0	0	0	0	0	0
014h	I2C_SR1	保留																SMBALERT	TIMEOUT	保留	PECERR	OVR	AF	ARLO	BERR	TxE	RxNE	保留	STOPF	ADD10	BTF	ADDR	SB
	复位值																	0	0		0	0	0	0	0	0	0		0	0	0	0	0
018h	I2C_SR2	保留																PEC[7:0]								DUALF	SMBHOST	SMBDEFAU	GENCALL	保留	TRA	BUSY	MSL
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0
01Ch	I2C_CCR	保留																F/S	DUTY	保留	CCR[11:0]												
	复位值																	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0
20h	I2C_TRISE	保留																										TRISE[5:0]					
	复位值																											0	0	0	0	1	0

16 模拟/数字转换(ADC)

16.1介绍

12 位 ADC 是一种逐次逼近型模拟数字转换器。它有 18 个通道，可测量 16 个外部和 2 个内部信号源。各通道的 A/D 转换可以单次、连续、扫描或间断模式执行。ADC 的结果可以左对齐或右对齐方式存储在 16 位数据寄存器中。

模拟看门狗特性允许应用程序检测输入电压是否超出用户定义的高/低阀值值。

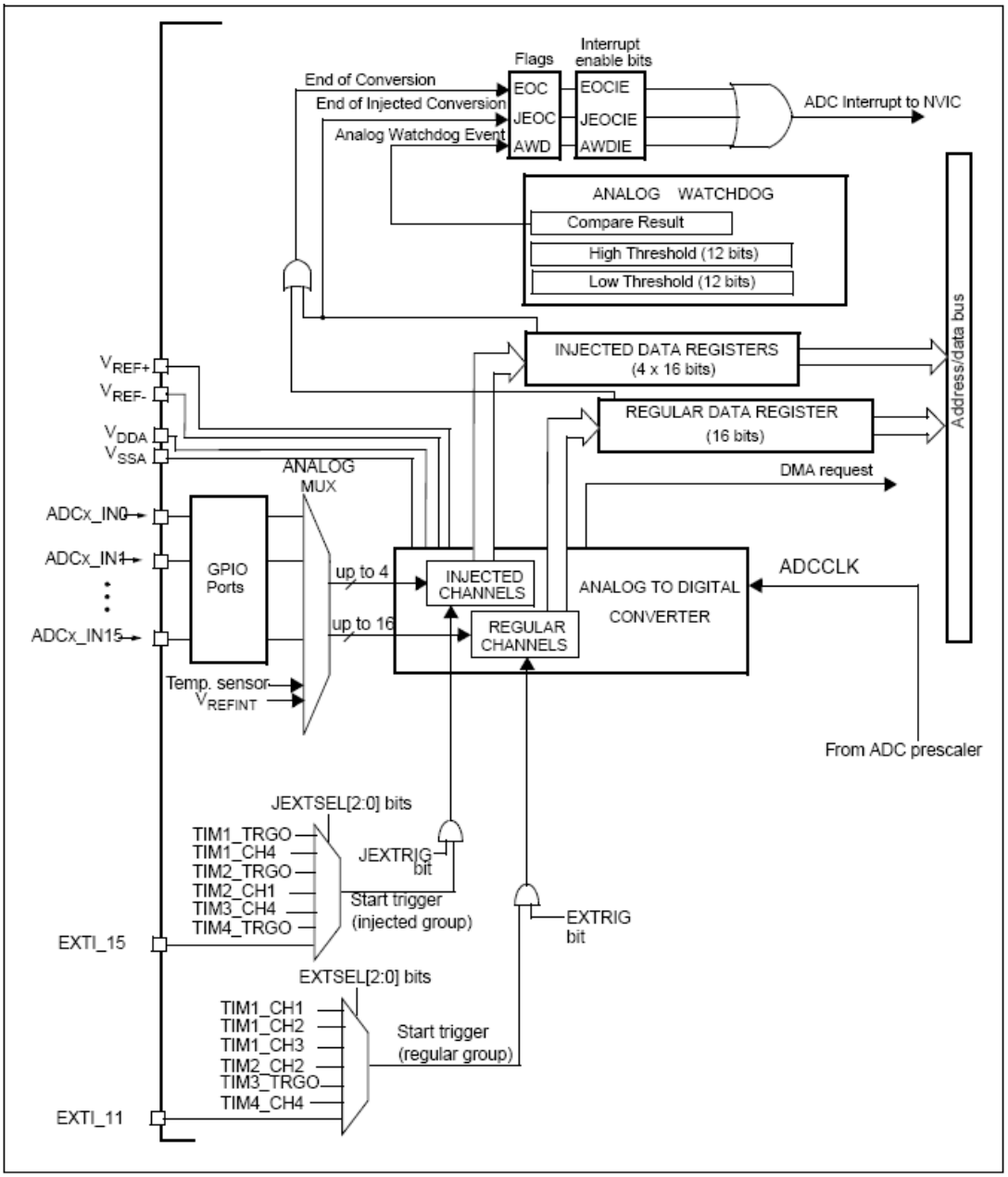
16.2主要特征

- 12-位分辨率
- 转换结束，注入转换结束和发生模拟看门狗事件时产生中断
- 单次和连续转换模式
- 从通道 0 到通道 n 的自动扫描模式
- 自校准
- 带内嵌数据一致的数据对齐
- 通道之间采样间隔可编程
- 规则转换和注入转换均有外部触发选项
- 间断模式
- 双重模式(带 2 个 ADC 的器件)
- ADC 转换时间：
  - STM32F103xx 增强型产品：ADC 时钟为 56MHz 时为 1μs(ADC 时钟为 72MHz 为 1.17μs)
  - STM32F101xx 基本型产品：ADC 时钟为 28MHz 时为 1μs(ADC 时钟为 36MHz 为 1.55μs)
- ADC 供电要求：2.4V 到 3.6V
- ADC 输入范围：$V_{REF-} \leq V_{IN} \leq V_{REF+}$
- 规则通道转换期间有 DMA 请求产生。

图 141 是ADC模块的方框图。

注意：　　如果有V_{REF-}管脚(取决于封装)，必须和VSSA 相连接

图141 单个 ADC 框图



16.3引脚描述

表48 ADC 管脚

名称	信号类型	注解
V_{REF+}	输入，模拟参考正极	ADC使用的高端/正极参考电压，$2.4V \le V_{REF+} \le V_{DDA}$
V_{DDA}	输入，模拟电源	等效于V_{DD}的模拟电源且：$2.4V \le V_{DDA} \le V_{DD}(3.6V)$
V_{REF-}	输入，模拟参考负极	ADC使用的低端/负极参考电压，$V_{REF-} = V_{SSA}$
V_{SSA}	输入，模拟电源地	等效于V_{SS}的模拟电源地
ADC_IN[15:0]	模拟输入信号	16个模拟输入通道

16.4功能描述

16.4.1 ADC开关控制

通过设置 ADC_CR1 寄存器的 ADON 位可给 ADC 上电。当第一次设置 ADON 位时，它将 ADC 从断电状态下唤醒。

ADC 上电延迟一段时间后(t_{STAB})，再次设置 ADON 位时开始进行转换。

通过清除 ADON 位可以停止转换，并将 ADC 置于断电模式。在这个模式中，ADC 几乎不耗电(仅几个 μA)。

16.4.2 ADC时钟

由时钟控制器提供的 ADCCLK 时钟和 PCLK2(APB2 时钟)同步。CLK 控制器为 ADC 时钟提供一个专用的可编程预分频器，更多详细信息请参考 CLK 章节。

16.4.3 通道选择

有 16 个多路通道。可以把转换分成两组：规则的和注入的。在任意多个通道上以任意顺序进行的一系列转换构成成组转换。例如，可以如下顺序完成转换：通道 3、通道 8、通道 2、通道 2、通道 0、通道 2、通道 2、通道 15。

- 规则组由多达 16 个转换组成。规则通道和它们的转换顺序在 ADC_SQRx 寄存器中选择。规则组中转换的总数写入 ADC_SQR1 寄存器的 L[3:0]位中。
- 注入组由多达 4 个转换组成。注入通道和它们的转换顺序在 ADC_JSQR 寄存器中选择。注入组里的转换总数目必须写入 ADC_JSQR 寄存器的 L[1:0]位中。

如果 ADC_SQRx 或 ADC_JSQR 寄存器在转换期间被更改，当前的转换被清除，一个新的启动脉冲将发送到 ADC 以转换新选择的组。

温度传感器/ VREFINT内部通道

温度传感器和通道 ADCx_IN16 相连接，内部参考电压 V_{REFINT} 和 ADCx_IN17 相连接。可以按注入或规则通道对这两个内部通道进行转换。

注意： *传感器和 V_{REFINT} 只能出现在主 ADC1 中。*

16.4.4 单次转换模式

单次转换模式里，ADC 只执行一次转换。这个模式既可通过设置 ADC_CR2 寄存器的 ADON 位(只适用于规则通道)启动也可通过外部触发启动(适用于规则通道或注入通道)，这时 CONT 位为 0。
一旦选择通道的转换完成：

- 如果一个规则通道被转换：
  - 转换数据被储存在 16 位 ADC_DR 寄存器中
  - EOC(转换结束)标志被设置
  - 如果设置了 EOCIE，则产生中断。
- 如果一个注入通道被转换：
  - 转换数据被储存在 16 位的 ADC_DRJ1 寄存器中
  - JEOC(注入转换结束)标志被设置
  - 如果设置了 JEOCIE 位，则产生中断。

然后 ADC 停止。

16.4.5 连续转换模式

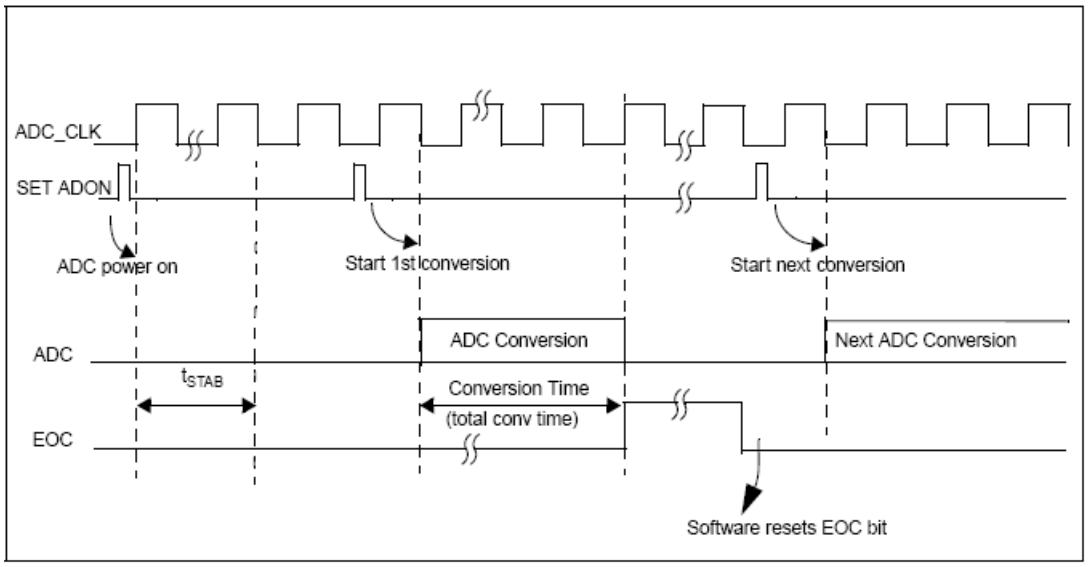
在连续转换模式中，当前面 ADC 转换一结束马上就启动另一次转换。此模式可通过外部触发启动或通过设置 ADC_CR2 寄存器上的 ADON 位启动，此时 CONT 位是 1。
每个转换后：

- 如果一个规则通道被转换：
  - 转换数据被储存在 16 位的 ADC_DR 寄存器中
  - EOC(转换结束)标志被设置
  - 如果设置了 EOCIE，则产生中断。
- 如果一个注入通道被转换：
  - 转换数据被储存在 16 位的 ADC_DRJ1 寄存器中
  - JEOC(注入转换结束)标志被设置
  - 如果设置了 JEOCIE 位，则产生中断。

16.4.6 时序图

如 图 142 所示，ADC在开始精确转换前需要一个稳定时间t_{STAB}。在开始ADC转换和 14 个时钟周期后，EOC标志被设置，16 位ADC数据寄存器包含转换的结果。

图142 时序图



16.4.7 模拟看门狗

如果被 ADC 转换的模拟电压低于低阀值或高于高阀值，AWD 模拟看门狗状态位被设置。这些阀值位于在 ADC_HTR 和 ADC_LTR 寄存器的最低 12 个有效位中。通过设置 ADC_CR1 寄存器的 AWDIE 位以允许产生相应中断。

阀值独立于由ADC_CR2 寄存器上的ALIGN位选择的数据对齐模式。比较是在对齐之前完成的(见 16.6 节)。

通过配置ADC_CR1 寄存器，模拟看门狗可以作用于 1 个或多个通道，如 表 49 所示。

图143 模拟看门狗警戒区

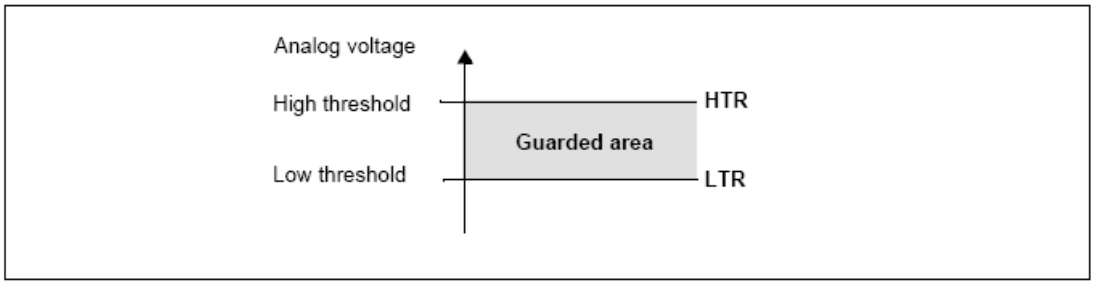


表49 模拟看门狗通道选择

模拟看门狗警戒的通道	ADC_CR1寄存器控制位		
	AWDSGL位	AWDEN位	JAWDEN位
无	任意值	0	0
所有注入通道	0	0	1
所有规则通道	0	1	0
所有注入和规则通道	0	1	1

单一的(1)注入通道	1	0	1
单一的(1)规则通道	1	1	0
单一的(1)注入或规则通道	1	1	1

(1) 由 AWDCH[4:0]位选择

16.4.8 扫描模式

此模式用来扫描一组模拟通道。

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

如果设置了 DMA 位，在每次 EOC 后，DMA 控制器把规则组通道的转换数据传输到 SRAM 中。而注入通道转换的数据总是存储在 ADC_JDRx 寄存器中。

16.4.9 注入通道管理

触发注入

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

1. 利用外部触发或通过设置ADC_CR2寄存器的ADON位，启动一组规则通道的转换。
2. 如果在规则通道转换期间产生一外部注入触发，当前转换被复位，注入通道序列被以单次扫描方式进行转换。
3. 然后，恢复上次被中断的规则组通道转换。如果在注入转换期间产生一规则事件，注入转换不会被中断，但是规则序列将在注入序列结束后被执行。图144是其定时图。

注： 当使用触发的注入转换时，必须保证触发事件的间隔长于注入序列。例如：序列长度为 28 个 ADC 时钟周期(即 2 个具有 1.5 个时钟间隔采样时间的转换)，触发之间最小的间隔必须是 29 个 ADC 时钟周期。

自动注入

如果设置了 JAUTO 位，在规则组通道之后，注入组通道被自动转换。这可以用来转换在 ADC_SQRx 和 ADC_JSQR 寄存器中设置的多至 20 个转换序列。

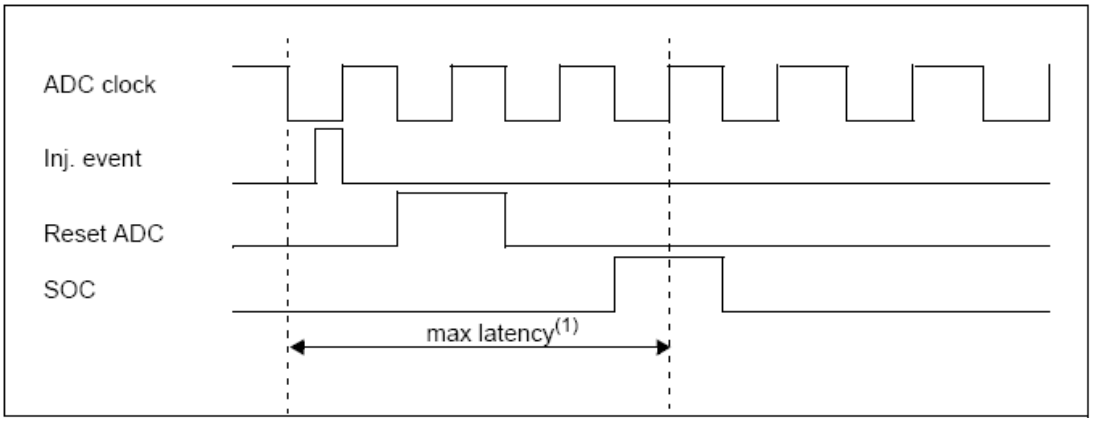
在此模式里，必须禁止注入通道的外部触发。

如果除 JAUTO 位外还设置了 CONT 位，规则通道至注入通道的转换序列被连续执行。

对于 ADC 时钟预分频系数为 4 至 8 时，当从规则转换切换到注入序列或从注入转换切换到规则序列时，会自动插入 1 个 ADC 时钟间隔；ADC 当时钟预分频系数为 2 时，则有 2 个 ADC 时钟间隔的延迟。

注意：　　不可能同时使用自动注入和间断模式。

图144 注入转换延时



(1) 最大延迟数值请参考 STM32F101xx 和 STM32F103xx 数据手册中有关电气特性部分。

16.4.10 间断模式

规则组

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

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

举例：

n=3，被转换的通道 = 0, 1, 2, 3, 6, 7, 9, 10
第一次触发：转换的序列为 0，1，2
第二次触发：转换的序列为 3，6，7
第三次触发：转换的序列为 9，10，并产生 EOC 事件
第四次触发：转换的序列 0，1，2

注意：　　当一规则组以间断模式转换时，转换序列结束后不自动从头开始。当所有子组被转换完成，下一次触发启动第一个子组的转换。在上面的例子中，第四次触发重新转换第一子组的通道 0，1 和 2。

注入组

此模式通过设置 ADC_CR1 寄存器的 JDISCEN 位激活。在一个外部触发事件后，给模式按序转换 ADC_JSQR 寄存器中选择的序列。

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

例子：

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

注意：1 当完成所有注入通道转换，下个触发启动第 1 个注入通道的转换。在上述例子中，第四个触发重新转换第 1 个注入通道 1。
2 不能同时使用自动注入和间断模式。
3 必须避免同时为规则和注入组设置间断模式。间断模式只能作用于一组转换。

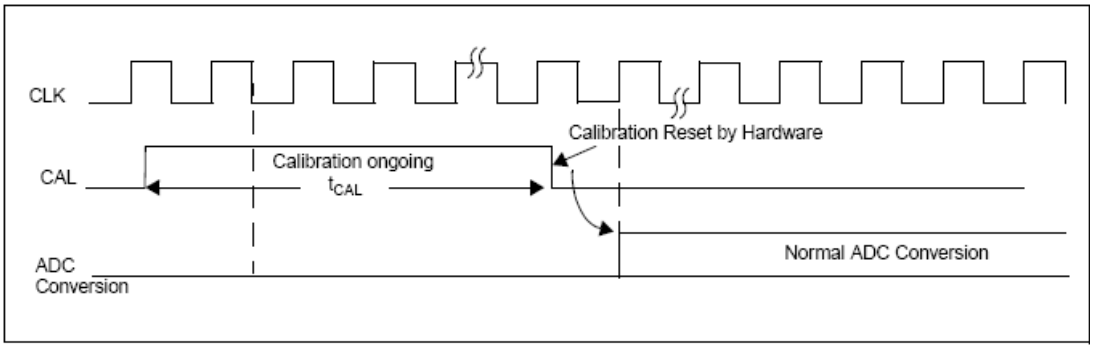
16.5 校准

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

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

注意：1 建议在每次上电后执行校准。
2 启动校准前，ADC 必须处于关电状态(ADON='0')超过至少两个 ADC 时钟周期。

图145 校准时序图



16.6数据对齐

ADC_CR2 寄存器中的ALIGN位选择转换后数据储存的对齐方式。数据可以左对齐或右对齐，如 图 146 和 图 147 所示。

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

对于规则组通道，不需减去偏移值，因此只有 12 个位有效。

图146 数据右对齐

注入组

SEXT	SEXT	SEXT	SEXT	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0

规则组

0	0	0	0	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0

图147 数据左对齐

注入组

SEXT	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	0	0	0

规则组

D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	0	0	0	0

16.7可编程的通道采样时间

ADC 使用若干个 ADC_CLK 周期对输入电压采样，采样周期数目可以通过 ADC_SMPR1 和 ADC_SMPR2 寄存器中的 SMP[2:0]位而更改。每个通道可以以不同的时间采样。总转换时间如下计算：

T_{CONV} = 采样时间+ 12.5 个周期

例如：

当 ADCCLK=14MHz 和 1.5 周期的采样时间

T_{CONV} = 1.5 + 12.5 = 14 周期 = 1μs

16.8外部触发转换

转换可以由外部事件触发(例如定时器捕获，EXTI 线)。如果设置了 EXTTRIG 控制位，则外部事件就能够触发转换。EXTSEL[2:0]和 JEXTSEL2:0]控制位允许应用程序选择 8 个可能的事件中的某一个可以触发规则和注入组的采样。

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

表50 用于规则通道的外部触发

触发源	类型	EXTSEL[2:0]
定时器1的CC1输出	片上定时器的内部信号	000
定时器1的CC2输出		001
定时器1的CC3输出		010
定时器2的CC2输出		011
定时器3的TRGO输出		100
定时器4的CC4输出		101
EXTI线11	外部管脚	110
SWSTART	软件控制位	111

表51 用于注入通道的外部触发

触发源	连接类型	JEXTSEL[2:0]
定时器1的TRGO输出	片上定时器的内部信号	000
定时器1的CC4输出		001
定时器2的TRGO输出		010
定时器2的CC1输出		011
定时器3的CC4输出		100
定时器4的TRGO输出		101
EXTI线15	外部管脚	110
JSWSTART	软件控制位	111

软件源触发事件可以通过设置一个寄存器位产生(ADC_CR2 的 SWSTART 和 JSWSTART)。

规则组的转换可以被注入触发打断。

16.9 DMA请求

因为规则通道转换的值储存在一个唯一的数据寄存器中，所以当转换多个规则通道时需要使用 DMA，这可以避免丢失已经存储在 ADC_DR 寄存器中的数据。

只有在规则通道的转换结束时才产生 DMA 请求，并将转换的数据从 ADC_DR 寄存器传输到用户指定的目的地址。

注： 只有 ADC1 能够产生 DMA 请求。

16.10 双ADC模式

在有 2 个ADC的器件中，可以使用双ADC模式(见 图 148双ADC框图)。

在双 ADC 模式里，根据 ADC1_CR1 寄存器中 DUALMOD[2:0]位所选的模式，转换的启动可以是 ADC1 主和 ADC2 从的交替触发或同时触发。

注意： *在双 ADC 模式里，当转换配置成由外部事件触发时，用户必须将其设置成仅触发主 ADC，从 ADC 设置成软件触发，这样可以防止意外的触发从转换。但是，主和从 ADC 的外部触发必须同时被激活。*

共有 6 种可能的模式：

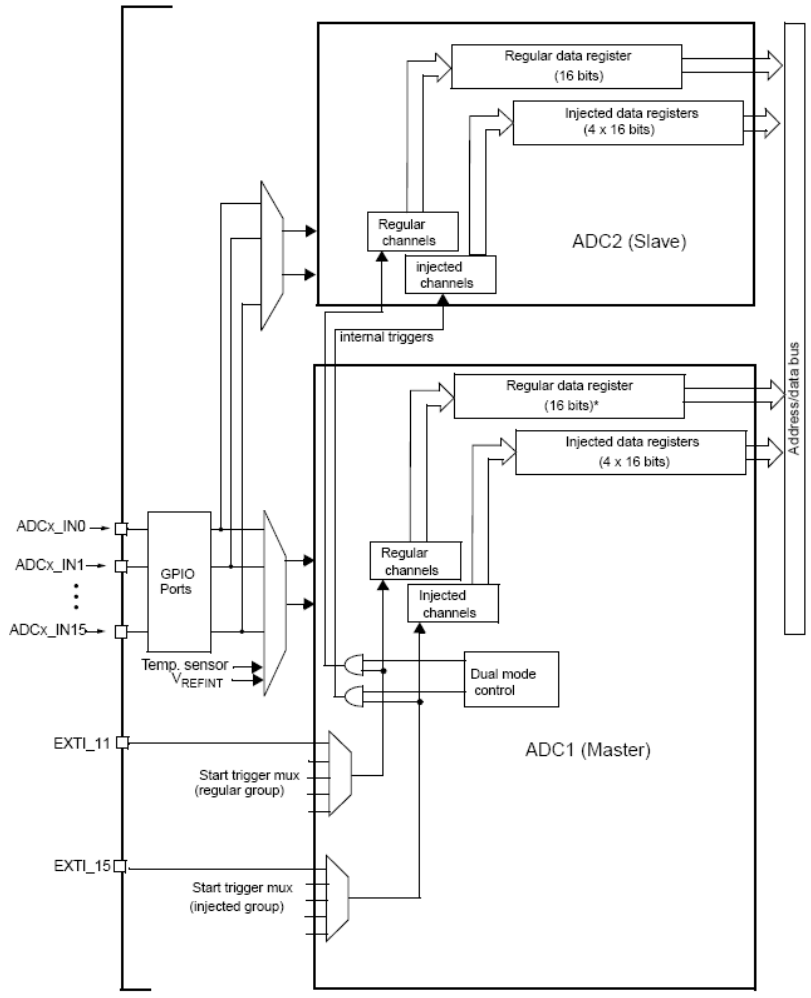
- 同时注入模式
- 同时规则模式
- 快速交替模式
- 慢速交替模式
- 交替触发模式
- 独立模式

还有可以用下列方式组合使用上面的模式：

- 同时注入模式+同时规则模式
- 同时规则模式+交替触发模式
- 同时注入模式+交替模式

注意： *在双 ADC 模式里，为了从主数据寄存器上读取从转换数据，DMA 位必须被使能，即使并不用它来传输规则通道数据。*

图148 双 ADC 框图



注：外部触发信号作用于ADC2，但在本图中没有显示。

* 某些双ADC模式中，在完整的32位ADC1数据寄存器(ADC1_DR)中包含了ADC1和ADC2的规则转换数据。

16.10.1 同步注入模式

此模式转换一个注入通道组。外部触发源来自 ADC1 的注入组多路器(由 ADC1_CR2 寄存器的 JEXTSEL[2:0]选择)。给 ADC2 提供同步触发。

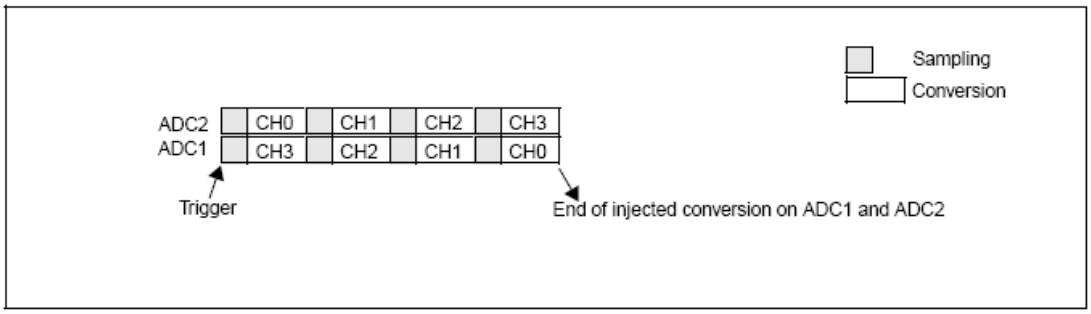
注意： 不要在 2 个 ADC 上转换相同的通道(如果转换两个 ADC 的相同通道，不可能提供重叠的采样时间)。

在 ADC1 或 ADC2 的转换结束时：

- 转换的数据存储在每个 ADC 接口的 ADC_JDRx 寄存器中。
- 当所有 ADC1/ADC2 注入通道都被转换时产生 JEOC 中断(如果任一 ADC 接口开放了中断的话)。

注： 在同步模式中，必须转换具有相同长度的序列，或保证触发的间隔比 2 个序列中较长的序列长，否则当较长序列的转换还未完成时，具有较短序列的 ADC 转换会被重启。

图149 在 4 个通道上的同时注入模式



16.10.2 同步规则模式

此模式在规则通道组上执行。外部触发源来自 ADC1 的规则组多路器(由 ADC1_CR2 寄存器的 EXTSEL[2:0]选择)。给 ADC2 提供同步触发。

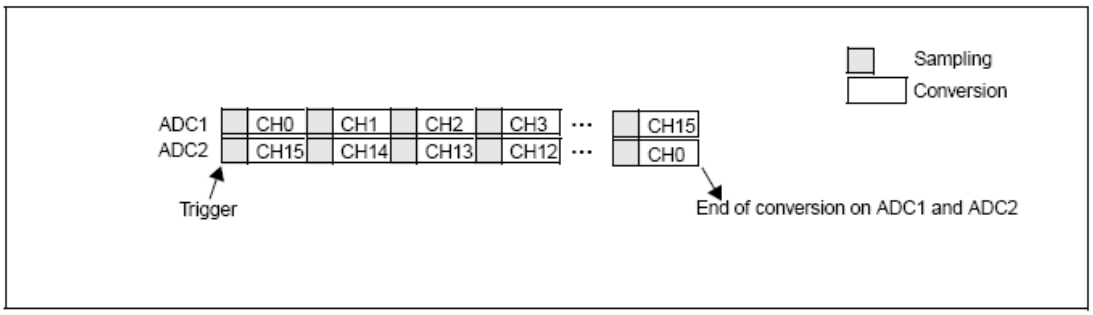
注意： 不要在 2 个 ADC 上转换相同的通道(如果转换两个 ADC 的相同通道，不可能提供重叠的采样时间)。

在 ADC1 或 ADC2 的转换结束时：

- 产生一个 32 位 DMA 传输请求(如果设置了 DMA 位)，传输到 SRAM 的 32 位 ADC1_DR 寄存器的上半个字包含 ADC2 的转换数据，低半个字包含 ADC1 的转换数据。
- 当所有 ADC1/ADC2 规则通道都被转换完时，产生 EOC 中断(如果任一 ADC 接口开放了中断的话)。

注： 在同步规则模式中，必须转换具有相同长度的序列，或保证触发的间隔比 2 个序列中较长的序列长，否则当较长序列的转换还未完成时，具有较短序列的 ADC 转换会被重启。

图150 在 16 个通道上的同时规则模式



16.10.3 快速交替模式

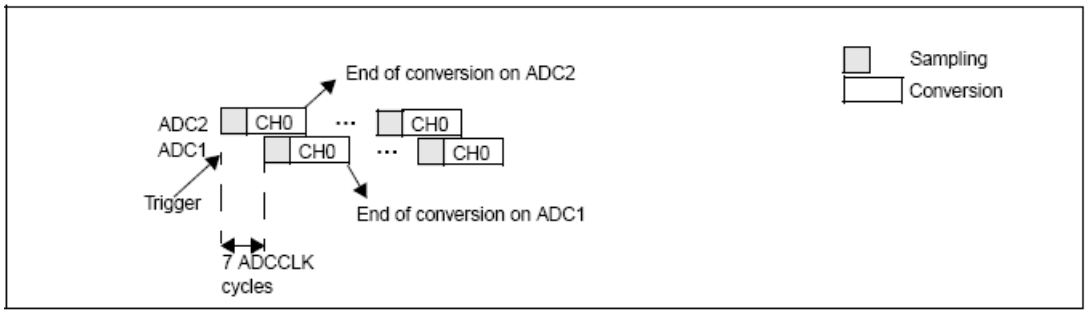
此模式只适用于规则通道组(通常一个通道)。外部触发源来自 ADC1 的规则通道多路器。外部触发产生后：

- ADC2 立即启动并且
- ADC1 在延迟 7 个 ADC 时钟周期后启动

如果同时设置了 ADC1 和 ADC2 的 CONT 位，所选的两个 ADC 规则通道将被连续地转换。
ADC1 产生一 EOC 中断后(由 EOCIE 使能)，产生一个 32 位的 DMA 传输请求(如果设置了 DMA 位)，ADC1_DR 寄存器的 32 位数据被传输到 SRAM，ADC1_DR 的上半个字包含 ADC2 的转换数据，低半个字包含 ADC1 的转换数据。

注意： 最大允许采样时间<7 个 ADCCLK 周期，避免 ADC1 和 ADC2 转换相同通道时发生两个采样周期的重叠。

图151 在 1 个通道上连续转换模式下的快速交替模式



16.10.4 慢速交替模式

此模式只适用于规则通道组(通常一个通道)。外部触发源来自 ADC1 的规则通道多路器。外部触发产生后：

- ADC2 立即启动并且
- ADC1 在延迟 14 个 ADC 时钟周期后启动
- 在延迟第二个 14 个 ADC 周期后 ADC2 再次启动，如此循环。

注意： 最大允许采样时间<14 个 ADCCLK 周期，以避免和下个转换重叠。

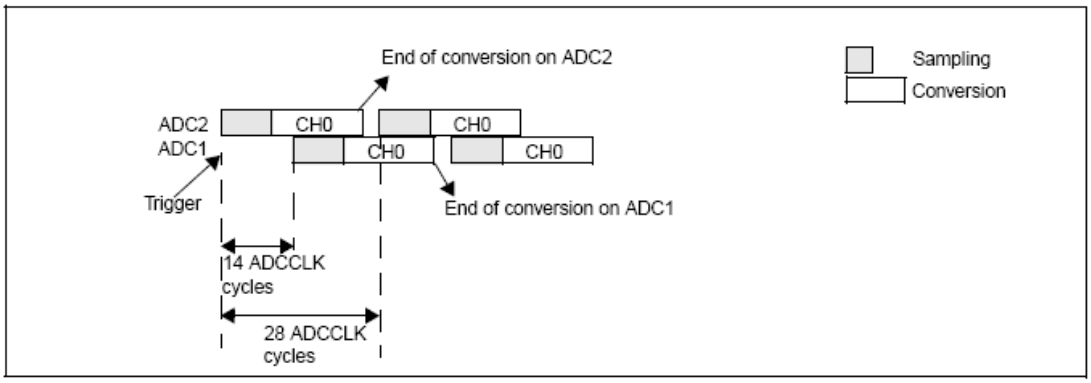
ADC1 产生 EOC 中断后(由 EOCIE 使能)，产生一个 32 位的 DMA 传输请求(如果设置了 DMA 位)，ADC1_DR 寄存器的 32 位数据被传输到 SRAM，ADC1_DR 的上半个字包含 ADC2 的转换数据，低半个字包含 ADC1 的转换数据。

在 28 个 ADC 时钟周期后自动启动新的 ADC2 转换。

在这个模式里不能设置 CONT 位，因为它将连续转换所选择的规则通道。

注意： 应用程序必须确保当使用交替模式时，不能有注入通道的外部触发产生。

图152 在 1 个通道上的慢速交替模式



16.10.5 交替触发模式

此模式只适用于注入通道组。外部触发源来自 ADC1 的注入通道多路器。

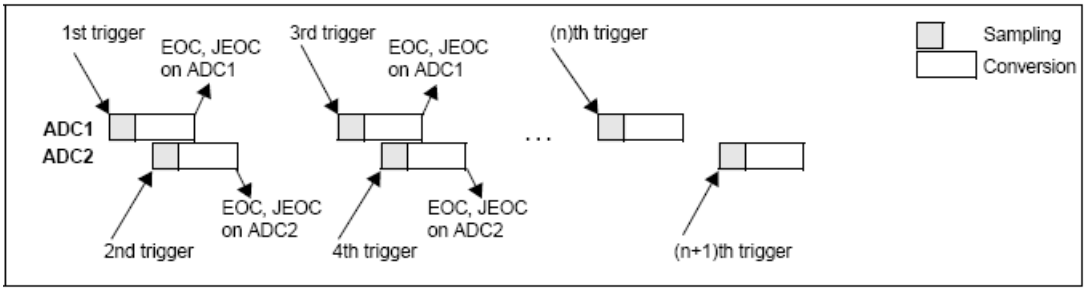
- 当第一个触发产生时，ADC1 上的所有注入组通道被转换。
- 当第二个触发到达时，ADC2 上的所有注入组通道被转换。
- 如此循环……

如果允许产生 JEOC 中断，在所有 ADC1 注入组通道转换后产生一个 JEOC 中断。

如果允许产生 JEOC 中断，在所有 ADC2 注入组通道转换后产生一个 JEOC 中断。

当所有注入组通道都转换完后如果产生另一个外部触发，交替触发处理从转换 ADC1 注入组通道重新开始。

图153 交替触发：每个 ADC1 的注入通道组



如果 ADC1 和 ADC2 上同时使用注入间断模式：

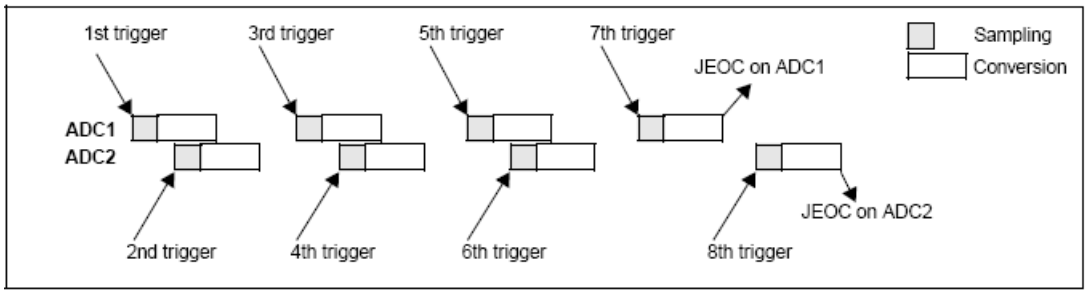
- 当第一个触发产生时，ADC1 上的第一个注入通道被转换。
- 当第二个触发到达时，ADC2 上的第一个注入通道被转换。
- 如此循环……

如果允许产生 JEOC 中断，在所有 ADC1 注入组通道转换后产生一个 JEOC 中断。

如果允许产生 JEOC 中断，在所有 ADC2 注入组通道转换后产生一个 JEOC 中断。

当所有注入组通道都转换完后如果产生另一个外部触发，重新开始。

图154 交替触发：在间断模式下每个 ADC 上的 4 个注入通道



16.10.6 独立模式

此模式里，双 ADC 同步不工作，每个 ADC 接口独立工作。

16.10.7 混合的规则/注入同步模式

有可能中断规则组同步转换以启动注入组的同步转换。

注： 在混合的规则/注入同步模式中，必须转换具有相同长度的序列，或保证触发的间隔比 2 个序列中较长的序列长，否则当较长序列的转换还未完成时，具有较短序列的 ADC 转换会被重启。

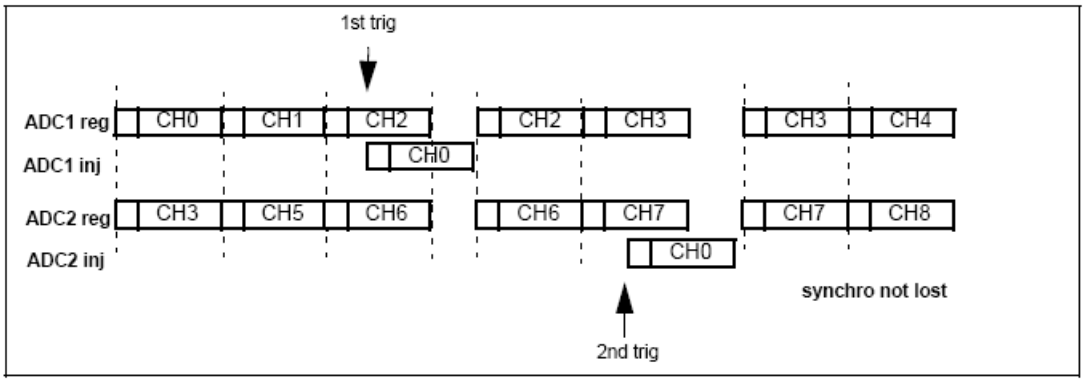
16.10.8 混合的同步规则+交替触发模式

有可能中断规则组同步转换以启动注入组交替触发转换。图 155 显示了一个规则同步转换被交替触发所中断。

注入交替转换在注入事件到达后立即启动。如果规则转换已经在运行，为了在注入转换后确保同步，所有的 ADC(主和从)的规则转换被停止，并在注入转换结束时同步恢复。

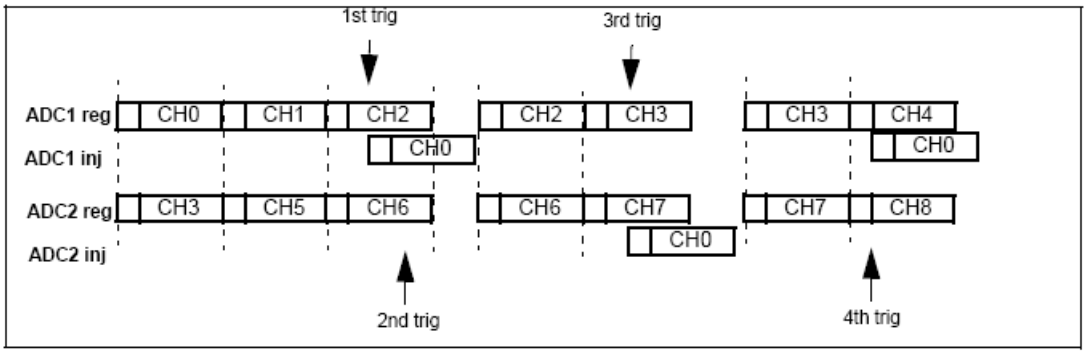
注： 在混合的同步规则+交替触发模式中，必须转换具有相同长度的序列，或保证触发的间隔比 2 个序列中较长的序列长，否则当较长序列的转换还未完成时，具有较短序列的 ADC 转换会被重启。

图155 交替+规则同步



如果触发事件发生在一个中断了规则转换的注入转换期间，这个触发事件将被忽略。图 156 示出了这种情况的操作(第 2 个触发被忽略)。

图156 触发事件发生在注入转换期间

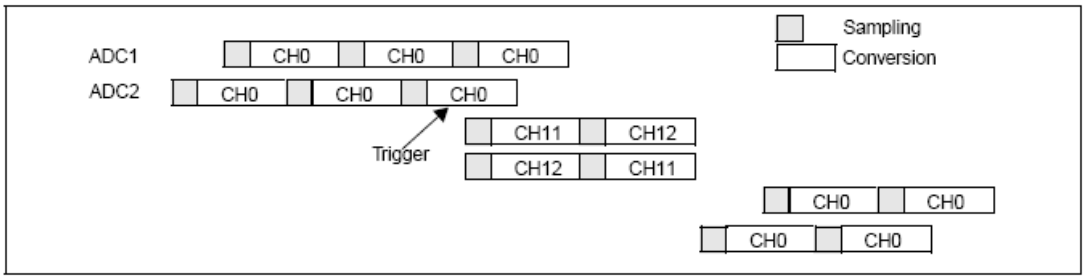


16.10.9 混合同步注入+交替模式

可以用一个注入事件来中断一个交替转换。这种情况下，交替转换被中断，注入转换被启动，在注入序列转换结束时，交替转换被恢复。0 是这种情况的一个例子。

注： 当 ADC 时钟预分频系数设置为 4 时，交替模式不会均匀地分配采样时间，采样间隔是 8 个 ADC 时钟周期与 6 个 ADC 时钟周期轮替，而不是均匀的 7 个 ADC 时钟周期。

图157 交替的单通道转换被注入序列 CH11 和 CH12 中断



16.11 温度传感器

温度传感器可以用来测量器件周围的温度(T_A)。

温度传感器在内部和 ADCx_IN16 输入通道相连接，此通道把传感器输出的电压转换成数字值。温度传感器模拟输入的采样时间必须大于 2.2 µs。

图 158 是温度传感器的方框图。

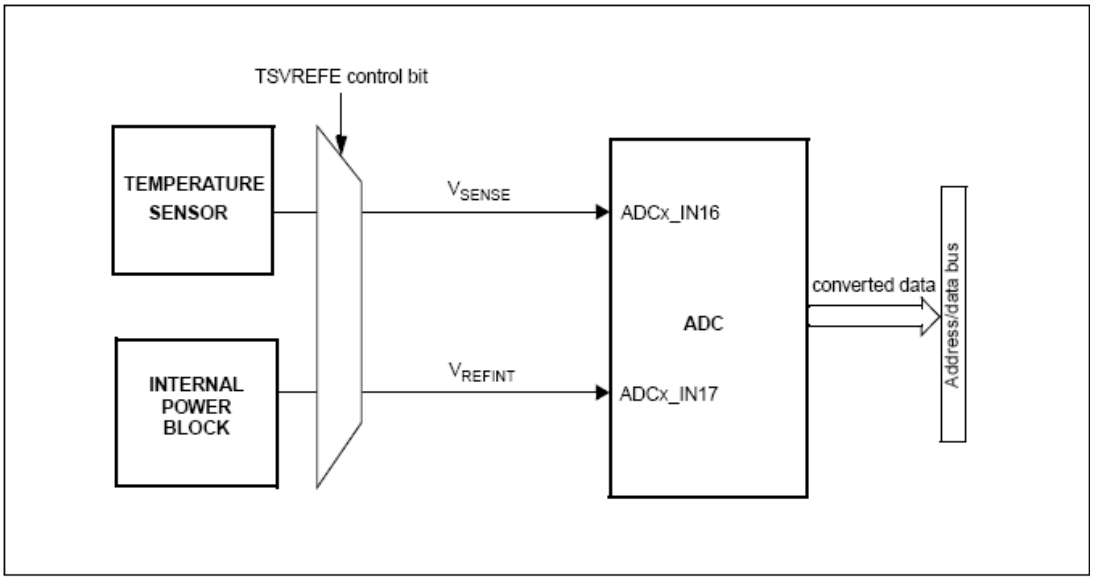
当没有被使用时，传感器可以置于关电模式。

注意： 必须设置 TSVREFE 位以激活内部通道：ADCx_IN16(温度传感器)和 ADCx_IN17(V_{REFINT})的转换。

主要特征

- 支持的温度范围：-40 到 125 度
- 精确度：+/- 1.5° C

图158 温度传感器和 V_{REFINT} 通道框图



读温度

为使用传感器：

1. 选择ADCx_IN16输入通道
2. 选择采样时间大于2.2 µs
3. 设置ADC控制寄存器2（ADC_CR2）的TSVREFE位，以唤醒关电模式下的温度传感器
4. 通过设置ADON位启动ADC转换（或用外部触发）
5. 读ADC数据寄存器上的V_{SENSE} 数据结果
6. 利用下列公式得出温度

   温度(°C) = {(V_{25} - V_{SENSE}) / Avg_Slope} + 25

   这里：

   V_{25} = V_{SENSE} 在 25°C 时的数值

   Avg_Slope ＝ 温度与 V_{SENSE} 曲线的平均斜率(单位为 mV/° C 或 µV/°C)

   参考电气特性章节中 V_{25} 和 Avg_Slope 的实际值。

注意：　传感器从关电模式唤醒后到可以输出正确水平的 V_{SENSE} 前，有一个建立时间。ADC 在上电后也有一个建立时间，为了缩短延时，应该同时设置 ADON 和 TSVREFE 位。

16.12 中断

规则和注入组转换结束时能产生中断，当模拟看门狗状态位被设置时也能产生中断。它们都有独立的中断使能位。

ADC_SR 寄存器中有 2 个其他标志，但是它们没有相关联的中断：

- JSTRT(注入组通道转换的启动)
- STRT(规则组通道转换的启动)

表52 ADC 中断

中断事件	事件标志	使能控制位
规则组转换结束	EOC	EOCIE
注入组转换结束	JEOC	JEOCIE
设置模拟看门狗状态位	AWD	AWDIE

16.13 ADC寄存器描述

寄存器描述中使用的一些缩略语请参考 1.1 节

16.13.1 ADC状态寄存器(ADC_SR)

地址偏移：0x00
复位值：0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留											STRT	JSTRT	JEOC	EOC	AWD
											rw	rw	rw	rw	rw

位31:15	保留。必须保持为0。
位4	STRT：规则通道开始位 该位由硬件在规则通道转换开始时设置，由软件清除。 0：规则通道转换未开始 1：规则通道转换已开始
位3	JSTRT：注入通道开始位 该位由硬件在注入通道组转换开始时设置，由软件清除。 0：注入通道转换未开始 1：注入通道转换已开始
位2	JEOC：注入通道转换结束位 该位由硬件在所有注入通道组转换结束时设置，由软件清除 0：转换未完成 1：转换完成
位1	EOC：转换结束位 该位由硬件在(规则或注入)通道组转换结束时设置，由软件清除或由读取ADC_DR时清除 0：转换未完成 1：转换完成
位0	AWD：模拟看门狗标志位 该位由硬件在转换的电压值超出了ADC_LTR和ADC_HTR寄存器定义的范围时设置，由软件清除 0：没有发生模拟看门狗事件 1：发生模拟看门狗事件

16.13.2 ADC控制寄存器 1(ADC_CR1)

地址偏移：0x04
复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留								AWDEN	AWDENJ	保留		DUALMOD[3:0]			
								rw	rw			rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DISCNUM[2:0]			DISCENJ	DISCEN	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31:24	保留。必须保持为0。
位23	AWDEN：在规则通道上开启模拟看门狗 该位由软件设置和清除。 0：在规则通道上禁用模拟看门狗 1：在规则通道上使用模拟看门狗
位22	JAWDEN：在注入通道上开启模拟看门狗 该位由软件设置和清除。 0：在注入通道上禁用模拟看门狗 1：在注入通道上使用模拟看门狗
位21:20	保留。必须保持为0。
位19:16	DUALMOD[3:0]：双模式选择 软件使用这些位选择操作模式。 0000：独立模式 0001：混合的同步规则+注入同步模式 0010：混合的同步规则+交替触发模式 0011：混合同步注入+快速交替模式 0100：混合同步注入+慢速交替模式 0101：注入同步模式 0110：规则同步模式 0111：快速交替模式 1000：慢速交替模式 1001：交替触发模式 注： ■ 在ADC2中这些位为保留位 ■ 在双模式中，改变通道的配置会产生一个重新开始的条件，这将导致同步丢失。建议在进行任何配置改变前关闭双模式。
位15:13	DISCNUM[2:0]：间断模式通道计数 软件通过这些位定义在间断模式下，收到外部触发后转换规则通道的数目 000：1个通道 001：2个通道 …… 111：8个通道

位12	JDISCEN：在注入通道上的间断模式 该位由软件设置和清除，用于开启或关闭注入通道组上的间断模式 0：注入通道组上禁用间断模式 1：注入通道组上使用间断模式
位11	DISCEN：在规则通道上的间断模式 该位由软件设置和清除，用于开启或关闭规则通道组上的间断模式 0：规则通道组上禁用间断模式 1：规则通道组上使用间断模式
位10	JAUTO：自动的注入通道组转换 该位由软件设置和清除，用于开启或关闭规则通道组转换结束后自动的注入通道组转换 0：关闭自动的注入通道组转换 1：开启自动的注入通道组转换
位9	AWDSGL：扫描模式中在一个单一的通道上使用看门狗 该位由软件设置和清除，用于开启或关闭由AWDCH[4:0]位定义的通道上的模拟看门狗功能 0：在所有的通道上使用模拟看门狗 1：在单一通道上使用模拟看门狗
位8	SCAN：扫描模式 该位由软件设置和清除，用于开启或关闭扫描模式。在扫描模式中，由ADC_SQRx或ADC_JSQRx寄存器选中的通道被转换。 0：关闭扫描模式 1：使用扫描模式 注：如果分别设置了EOCIE或JEOCIE位，只在最后一个通道转换完毕才会产生EOC或JEOC中断。
位7	JEOCIE：允许产生注入通道转换结束中断 该位由软件设置和清除，用于禁止或允许所有注入通道转换结束后产生中断。 0：禁止JEOC中断 1：允许JEOC中断。当硬件设置JEOC位时产生中断。
位6	AWDIE：允许产生模拟看门狗中断 该位由软件设置和清除，用于禁止或允许模拟看门狗。在扫描模式下，如果看门狗检测到超范围的数值时，只有在设置了该位时扫描才会中止。 0：禁止模拟看门狗中断 1：允许模拟看门狗中断。
位5	EOCIE：允许产生EOC中断 该位由软件设置和清除，用于禁止或允许转换结束后产生中断。 0：禁止EOC中断 1：允许EOC中断。当硬件设置EOC位时产生中断。

位4:0	AWDCH[4:0]：模拟看门狗通道选择位 这些位由软件设置和清除，用于选择模拟看门狗保护的输入通道。 00000：ADC模拟输入通道0 00001：ADC模拟输入通道1 …… 01111：ADC模拟输入通道15 10000：ADC模拟输入通道16 10001：ADC模拟输入通道17 保留所有其他数值。 注： – ADC1的模拟输入通道16和通道17在芯片内部分别连到了温度传感器和V_{REFINT}。 – ADC2的模拟输入通道16和通道17在芯片内部连到了V_{SS}。

16.13.3 ADC控制寄存器 2(ADC_CR2)

地址偏移：0x08
复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留								TSVREFE	SWSTART	SWSTARTJ	EXTTRIG	EXTSEL[2:0]			保留
								rw	rw	rw	rw	rw	rw	rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
JEXTTRIG	JEXTSEL[2:0]			ALIGN	保留		DMA	保留				RSTCAL	CAL	CONT	ADON
rw	rw	rw	rw	rw			rw					rw	rw	rw	rw

位31:24	保留。必须保持为0。
位23	AWDEN：温度传感器和V_{REFINT}使能 该位由软件设置和清除，用于开启或禁止温度传感器和V_{REFINT}通道。在双ADC的器件中，该位置出现在ADC1中。 0：禁止温度传感器和V_{REFINT} 1：启用温度传感器和V_{REFINT}
位22	SWSTART：开始转换规则通道 由软件设置该位以启动转换，转换开始后硬件马上清除此位。如果在EXTSEL[2:0]位中选择了SWSTART为触发事件，该位用于启动一组规则通道的转换， 0：复位状态 1：开始转换规则通道
位21	JSWSTART：开始转换注入通道 由软件设置该位以启动转换，软件可清除此位或在转换开始后硬件马上清除此位。如果在JEXTSEL[2:0]位中选择了JSWSTART为触发事件，该位用于启动一组注入通道的转换， 0：复位状态 1：开始转换注入通道

位20	EXTTRIG：规则通道的外部触发转换模式 该位由软件设置和清除，用于开启或禁止可以启动规则通道组转换的外部触发信号。 0：不用外部触发信号启动转换 1：使用外部触发信号启动转换
位19:17	EXTSEL[2:0]：选择启动规则通道组转换的外部事件 这些位选择用于启动规则通道组转换的外部事件 000：定时器1的CC1事件 001：定时器1的CC2事件 010：定时器1的CC3事件 011：定时器2的CC2事件 100：定时器3的TRGO事件 101：定时器4的CC4事件 110：EXTI线11 111：SWSTART
位16	保留。必须保持为0。
位15	JEXTTRIG：注入通道的外部触发转换模式 该位由软件设置和清除，用于开启或禁止可以启动注入通道组转换的外部触发信号。 0：不用外部触发信号启动转换 1：使用外部触发信号启动转换
位14:12	JEXTSEL[2:0]：选择启动注入通道组转换的外部事件 这些位选择用于启动注入通道组转换的外部事件 000：定时器1的TRGO事件 001：定时器1的CC4事件 010：定时器2的TRGO事件 011：定时器2的CC1事件 100：定时器3的CC4事件 101：定时器4的TRGO事件 110：EXTI线15 111：JSWSTART
位11	ALIGN：数据对齐 该位由软件设置和清除。参考图146和图147。 0：右对齐 1：左对齐
位10:9	保留。必须保持为0。
位8	DMA：直接数据访问模式 该位由软件设置和清除。详见DMA控制器章节。 0：不使用DMA模式 1：使用DMA模式 注：在多于一个ADC的器件中，只有ADC1能产生DMA请求。
位7:4	保留。必须保持为0。

位3	RSTCAL：复位校准 该位由软件设置并由硬件清除。在校准寄存器被初始化后该位将被清除。 0：校准寄存器已初始化 1：初始化校准寄存器 注：当正在进行转换时，如果设置RSTCAL，清除校准寄存器需要额外的周期。
位2	CAL：A/D校准 该位由软件设置以开始校准，并在校准结束时由硬件清除。 0：校准完成 1：开始校准
位1	CONT：连续转换 该位由软件设置和清除。如果设置了此位，则转换将连续进行直到该位被清除。 0：单次转换模式 1：连续转换模式
位0	ADON：开/关A/D转换器 该位由软件设置和清除。当该位为0时，写入1将把ADC从断电模式下唤醒。 当该位为1时，写入1将启动转换。在转换器上电至转换开始有一个延迟t_{STAB}，图142。 0：关闭ADC转换/校准，并进入断电模式 1：开启ADC并启动转换。 注：如果在这个寄存器中与ADON一起还有其他位被改变，则转换不被触发。这是为了防止触发错误的转换。

16.13.4 ADC采样时间寄存器 1(ADC_SMPR1)

地址偏移：0x0C
复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留								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31:24	保留。必须保持为0。

位26:0	SMPx[2:0]：选择通道x的采样时间 这些位用于独立地选择每个通道的采样时间。在采样周期中通道选择位必须保持不变。 000：1.5周期　　100：41.5周期 001：7.5周期　　101：55.5周期 010：13.5周期　　110：71.5周期 011：28.5周期　　111：239.5周期 注： – ADC1的模拟输入通道16和通道17在芯片内部分别连到了温度传感器和V_{REFINT}。 – ADC2的模拟输入通道16和通道17在芯片内部连到了V_{SS}。

16.13.5 ADC采样时间寄存器 2(ADC_SMPR2)

地址偏移：0x10
复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留		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
SMP5_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。必须保持为0。
位29:0	SMPx[2:0]：选择通道x的采样时间 这些位用于独立地选择每个通道的采样时间。在采样周期中通道选择位必须保持不变。 000：1.5周期　　100：41.5周期 001：7.5周期　　101：55.5周期 010：13.5周期　　110：71.5周期 011：28.5周期　　111：239.5周期

16.13.6 ADC注入通道数据偏移寄存器x (ADC_JOFRx)(x=1..4)

地址偏移：0x14-0x20
复位值：0x0000 0000

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

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

位31:12	保留。必须保持为0。
位11:0	JOFFSETx[11:0]：注入通道x的数据偏移 当转换注入通道时，这些位定义了用于从原始转换数据中减去的数值。转换的结果可以在ADC_JDRx寄存器中读出。

16.13.7 ADC看门狗高阀值寄存器(ADC_HTR)

地址偏移：0x24
复位值：0x0000 0000

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

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

位31:12	保留。必须保持为0。
位11:0	HT[11:0]：模拟看门狗高阀值 这些位定义了模拟看门狗的阀值高限。

16.13.8 ADC看门狗低阀值寄存器(ADC_LRT)

地址偏移：0x28
复位值：0x0000 0000

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

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

位31:12	保留。必须保持为0。
位11:0	LT[11:0]：模拟看门狗低阀值 这些位定义了模拟看门狗的阀值低限。

16.13.9 ADC规则序列寄存器 1(ADC_SQR1)

地址偏移：0x2C
复位值：0x0000 0000

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

位31:24	保留。必须保持为0。
位23:20	L[3:0]：规则通道序列长度 这些位定义了在规则通道转换序列中转换总数。 0000：1个转换 0001：2个转换 …… 1111：16个转换
位19:15	SQ16[4:0]：规则序列中的第16个转换 这些位定义了转换序列中的第16个转换通道的编号(0~17)。
位14:10	SQ15[4:0]：规则序列中的第15个转换
位9:5	SQ14[4:0]：规则序列中的第14个转换
位4:0	SQ13[4:0]：规则序列中的第13个转换

16.13.10 ADC规则序列寄存器 2(ADC_SQR2)

地址偏移：0x30
复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留		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。必须保持为0。
位29:25	SQ12[4:0]：规则序列中的第12个转换 这些位定义了转换序列中的第12个转换通道的编号(0~17)。
位24:20	SQ11[4:0]：规则序列中的第11个转换
位19:15	SQ10[4:0]：规则序列中的第10个转换
位14:10	SQ9[4:0]：规则序列中的第9个转换

位9:5	SQ8[4:0]：规则序列中的第8个转换
位4:0	SQ7[4:0]：规则序列中的第7个转换

16.13.11 ADC规则序列寄存器 3(ADC_SQR3)

地址偏移：0x34
复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留		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。必须保持为0。
位29:25	SQ6[4:0]：规则序列中的第6个转换 这些位定义了转换序列中的第6个转换通道的编号(0~17)。
位24:20	SQ5[4:0]：规则序列中的第5个转换
位19:15	SQ4[4:0]：规则序列中的第4个转换
位14:10	SQ3[4:0]：规则序列中的第3个转换
位9:5	SQ2[4:0]：规则序列中的第2个转换
位4:0	SQ1[4:0]：规则序列中的第1个转换

16.13.12 ADC注入序列寄存器(ADC_JSQR)

地址偏移：0x38
复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留										JL[3:0]		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。必须保持为0。
位21:20	JL[1:0]：注入通道序列长度 这些位定义了在规则通道转换序列中转换总数。 00：1个转换 01：2个转换 10：3个转换 11：4个转换

位19:15	JSQ4[4:0]：注入序列中的第4个转换 这些位定义了转换序列中的第4个转换通道的编号(0~17)。 注：不同于规则转换序列，如果JL[1:0]的长度小于4，则转换的序列顺序是从(4-JL)开始。例如：ADC_JSQR[21:0] = 10 00011 00011 00111 00010，意味着扫描转换将按下列通道顺序转换：7、3、3，而不是2、7、3
位14:10	JSQ3[4:0]：注入序列中的第3个转换
位9:5	JSQ2[4:0]：注入序列中的第2个转换
位4:0	JSQ1[4:0]：注入序列中的第1个转换

16.13.13 ADC 注入数据寄存器x (ADC_JDRx) (x= 1..4)

地址偏移：0x3C – 0x48
复位值：0x0000 0000

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

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。必须保持为0。
位21:20	JDATA[15:0]：注入转换的数据 这些位为只读，包含了注入通道的转换结果。数据是左或右对齐，如图146和图147所示。

16.13.14 ADC规则数据寄存器(ADC_DR)

地址偏移：0x4C
复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADC2DATA[15:0]															
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
DATA[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

位31:16	ADC2DATA[15:0]：ADC2转换的数据 - 在ADC1中：双模式下，这些位包含了ADC2转换的规则通道数据。见16.10：双ADC模式 - 在ADC2中：不用这些位。
位15:0	DATA[15:0]：规则转换的数据 这些位为只读，包含了规则通道的转换结果。数据是左或右对齐，如图146和图147所示。

16.14 ADC寄存器地址映像

下表列出了所有的 ADC 寄存器。

表53 ADC 寄存器映像和复位值

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
00h	ADC_SR	保留																											STRT	JSTRT	JEOC	EOC	AWD
	复位值																												0	0	0	0	0
04h	ADC_CR1	保留								AWDEN	JAWDEN	保留		DUALMOD[3:0]				DISCNUM[2:0]			JDISCEN	DISCEN	JAUTO	AWDSGL	SCAN	JEOCIE	AWDIE	EOCIE	AWDCH[4:0]					
	复位值									0	0			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
08h	ADC_CR2	保留								TSVREFE	SWSTART	JSWSTART	EXTTRIG	EXTSEL[2:0]			保留	JEXTTRIG	JEXTSEL[2:0]			ALIGN	保留	DMA	保留			RSTCAL	CAL	CONT	ADON	
	复位值									0	0	0	0	0	0	0		0	0	0	0	0		0				0	0	0	0		
0Ch	ADC_SMPR1	采样时间位SMPx_x																															
	复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10h	ADC_SMPR2	采样时间位SMPx_x																															
	复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
14h	ADC_JOFR1	保留																						JOFFSET1[11:0]									
	复位值																							0	0	0	0	0	0	0	0	0	0
18h	ADC_JOFR2	保留																						JOFFSET2[11:0]									
	复位值																							0	0	0	0	0	0	0	0	0	0
1Ch	ADC_JOFR3	保留																						JOFFSET3[11:0]									
	复位值																							0	0	0	0	0	0	0	0	0	0
20h	ADC_JOFR4	保留																						JOFFSET4[11:0]									
	复位值																							0	0	0	0	0	0	0	0	0	0
1Ch	ADC_HTR	保留																						HT[11:0]									
	复位值																							0	0	0	0	0	0	0	0	0	0
20h	ADC_LTR	保留																						LT[11:0]									
	复位值																							0	0	0	0	0	0	0	0	0	0
2Ch	ADC_SQR1	保留								L[3:0]				规则通道序列SQx_x位																			
	复位值									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
30h	ADC_SQR2	保留	规则通道序列SQx_x位																														
	复位值		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
34h	ADC_SQR3	保留	规则通道序列SQx_x位																														
	复位值		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
38h	ADC_JSQR	保留										JL[1:0]		注入通道序列JSQx_x位																			
	复位值											0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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
3Ch	ADC_JDR1	保留																JDATA[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
40h	ADC_JDR2	保留																JDATA[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
44h	ADC_JDR3	保留																JDATA[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
48h	ADC_JDR4	保留																JDATA[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4Ch	ADC_DR	ADC2DATA[15:0]																规则 DATA[15:0]															
	复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

关于寄存器的起始地址，请参见第 2 章。

17 USB全速设备接口(USB)

17.1导言

USB 外设实现了 USB2.0 全速总线和 APB1 总线间的接口。

USB 外设支持 USB 挂起/恢复操作，可以停止设备时钟实现低功耗。

17.2主要特征

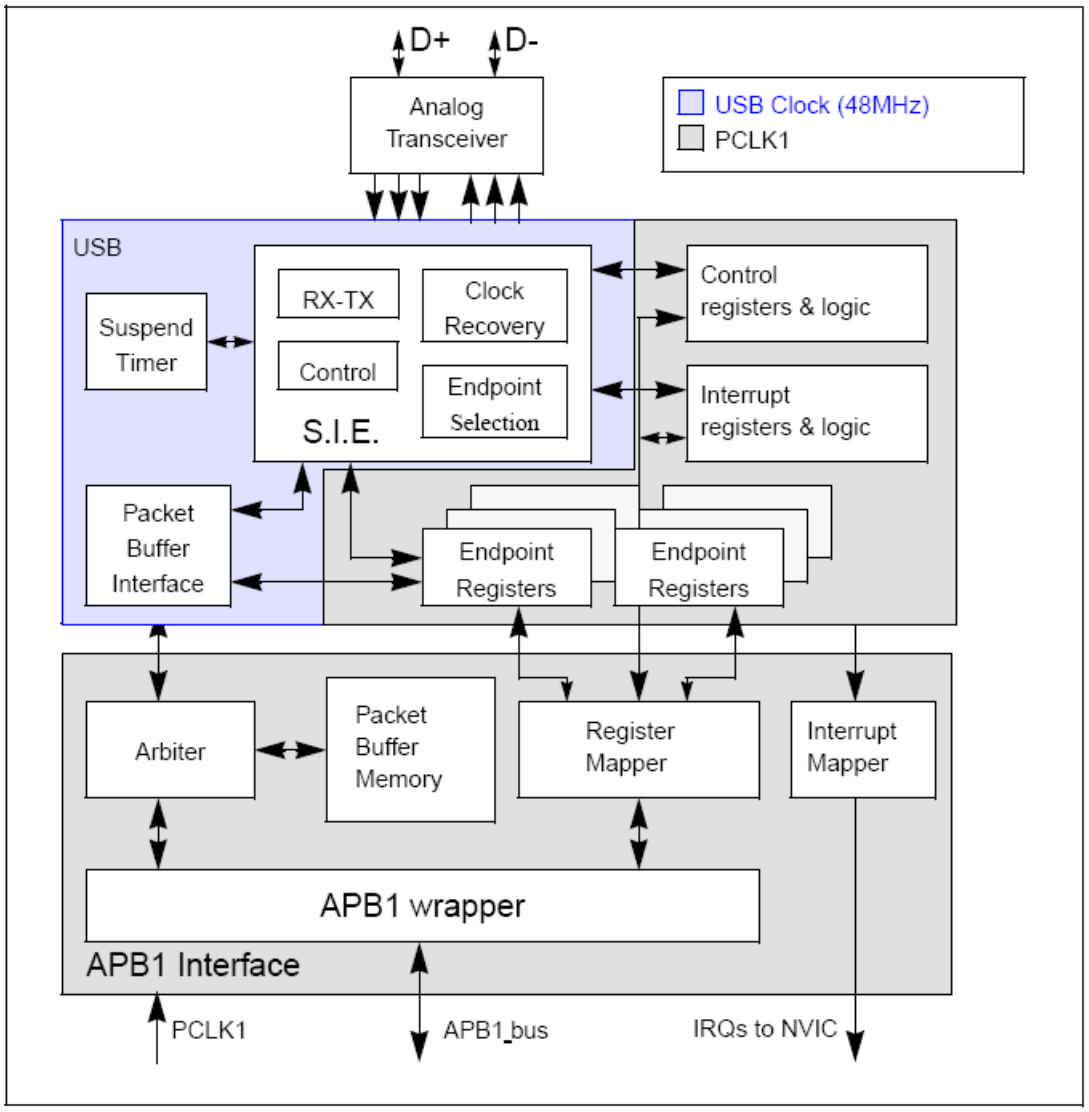
- 符合 USB2.0 全速设备的技术规范
- 可配置 1 到 8 个 USB 端点
- CRC(循环冗余校验)生成/校验，反向不归零（NRZI）编码/解码和位填充
- 支持同步传输
- 支持批量/同步端点的双缓冲区机制
- 支持 USB 挂起/恢复操作
- 帧锁定时钟脉冲生成

注： USB 和 CAN 共用一个专用的 512 字节的 SRAM 存储器用于数据的发送和接收，因此不同同时使用 USB 和 CAN(共享的 SRAM 被 USB 和 CAN 模块互斥地访问)。USB 和 CAN 可以同时用于一个应用中但不能在同一个时间使用。

17.3 方框图

图 159 是USB外设的方框图

图159 USB 设备框图



17.4 功能描述

USB 模块为 PC 主机和微控制器所实现的功能之间提供了符合 USB 规范的通信连接。PC 主机和微控制器之间的数据传输是通过共享一专用的数据缓冲区来完成的，该数据缓冲区能被 USB 外设直接访问。这块专用数据缓冲区的大小由所使用的端点数目和每个端点最大的数据分组大小所决定，每个端点最大可使用 512 字节缓冲区，最多可用于 16 个单向或 8 个双向端点。USB 模块同 PC 主机通信，根据 USB 规范实现令牌分组的检测，数据发送/接收的处理，和握手分组的处理。整个传输的格式由硬件完成，其中包括 CRC 的生成和校验。

每个端点都有一个缓冲区描述块，描述该端点使用的缓冲区的地址，大小和必须被传输的字节数。

当一有效的功能/端点的令牌分组被 USB 模块识别时，相关的数据传输随之发生(如果需要传输数据，并且端点已配置)。USB 模块通过一个内部的 16 位寄存器实现端口与专用缓冲区的数据交换。在所有的数据传输都完成后，如果需要，就根据传输的方向，发送或接收适当的握手分组。

在数据传输结束时，USB 模块将触发与端点相关的中断，通过读状态寄存器和/或者利用不同的中断处理程序，微控制器可以确定：

- 哪个端点需要得到服务
- 产生如位填充、格式、CRC、协议、缺失 ACK、缓冲区溢出/缓冲区未满等错误时，正在进行的是哪种类形的传输。

USB 模块对同步传输和高吞吐量的批量传输提供了特殊的双缓冲区机制，在微控制器使用一个缓冲区的时候，该机制保证了 USB 外设总是可以使用另一个缓冲区。

在任何不需要使用 USB 模块的时候，通过写控制寄存器总可以使 USB 模块置于低功耗模式(SUSPEND 模式)。在这种模式下，不产生任何静态电流消耗，同时 USB 时钟也会减慢或停止。通过对 USB 线上数据传输的检测，可以在低功耗模式下唤醒 USB 模块。也可以将一特定的中断输入源直接连接到唤醒引脚上，以使系统能立即恢复正常的时钟系统，并支持直接启动或停止时钟系统。

17.4.1 USB功能模块描述

USB 模块实现了标准 USB 接口的所有特性，它由以下部分组成：

- 串行接口引擎（SIE）：该模块包括的功能有：帧头同步域的识别，位填充，CRC 的产生和校验，PID 的验证/产生，和握手分组处理等。它与 USB 收发器交互，利用分组缓冲接口提供的虚拟缓冲区存储局部数据。它也根据 USB 事件，和类似于传输结束或一个包正确接收等与端点相关事件生成信号，例如帧首（Start of Frame），USB 复位，数据错误等等，这些信号用来产生中断。
- 定时器：此模块为任何要求帧首同步的其他部分产生帧锁时钟脉冲，并且在 USB 线上无数据传输的时间超过到 3ms 时，就判定为一 USB 全局挂起事件。
- 分组缓冲器接口：此模块管理那些用于发送和接收的临时本地内存单元。它根据 SIE 的要求分配合适的缓冲区，并定位到端点寄存器所指向的存储区地址。它在每个字节传输后，自动递增地址，直到数据分组传输结束。它记录传输的字节数并防止缓冲区溢出。
- 端点相关寄存器：每个端点都有一个与之相关的寄存器，用于描述端点类型和当前状态。对于单向和单缓冲器端点，一个寄存器就可以用于实现两个不同的端点。一共 8 个寄存器，可以用于实现最多 16 个单向/单缓冲的端点或者 7 个双缓冲的端点或者这些端点的组合。例如，可以同时实现 4 个双缓冲端点和 8 个单缓冲/单向端点。

- 控制寄存器：这些寄存器包含整个 USB 模块的状态信息，用来触发诸如恢复，低功耗等 USB 事件。
- 中断寄存器：这些寄存器包含中断屏蔽信息和中断事件的记录信息。配置和访问这些寄存器可以获取中断源，中断状态等信息，并能清除待处理中断的状态标志。

注意： 端点 0 总是作为单缓冲模式下的控制端点。

USB 模块通过 APB1 接口部件与 APB1 总线相连，APB1 接口部件包括以下部分：

- 分组缓冲区：数据分组缓存在分组缓冲区中，它由分组缓冲接口控制并创建数据结构。应用软件可以直接访问该缓冲区。它的大小为 512 字节，由 256 个 16 位的字构成。
- 仲裁器：该部件负责处理来自 APB1 总线和 USB 接口的存储器请求。它通过向 APB1 提供较高的访问优先权来解决总线的冲突，并且总是保留一半的存储器带宽供 USB 完成传输。它采用时分复用的策略实现了虚拟的双端口 SRAM，即在 USB 传输的同时，允许应用程序访问存储器。此策略也允许任意长度的多字节 APB1 传输。
- 寄存器映射单元：此部件将 USB 模块的各种字节宽度和位宽度的寄存器映射成能被 APB1 寻址的 16 位宽度的内存集合。
- 中断映射单元：此部件将可能产生中断的 USB 事件映射到 NVIC 的 IRQ 线上。
- APB1 封装：此部件为缓冲区和寄存器提供了到 APB1 的接口，并将整个 USB 模块映射到 APB1 地址空间。

17.5 编程中需要考虑的问题

在下面的章节中，将介绍 USB 模块和应用程序之间的交互过程，有利于简化应用程序的开发。

17.5.1 通用USB设备编程

这一部分描述了实现 USB 设备功能的应用程序需要完成的任务。除了介绍一般的 USB 事件中应该采取的操作外，还着重介绍了双缓冲端点和同步传输的操作。这些相关的操作都是由 USB 模块初始化，并由以下几节所描述的 USB 事件所驱动。

17.5.2 系统复位和上电复位

发生系统复位或者上电复位时，应用程序首先需要做的是提供 USB 模块所需要的时钟信号，然后清除复位信号，使程序可以访问 USB 模块的寄存器。复位之后的初始化流程如下所述：

首先，由应用程序激活寄存器单元的时钟，再配置设备时钟管理逻辑单元的相关控制位，清除复位信号。

其次，必须配置 CNTR 寄存器的 PDWN 位用以开启 USB 收发器相关的模拟部分，这点需要特别的处理。此位能打开为端点收发器供电的内部参考电压。由于打开内部电压需要一段启动时间(数据手册中的 $t_{STARTUP}$)，在此期间内 USB 收发器处于不确定状态，所以在设置 CNTR 寄存器的 PDWN 后必需等待一段时间之后，才能清除 USB 模块的复位信号(清除 CNTR 寄存器上的 FRES 位)，和 ISTR 寄存器的内容，以便在使能其他任何单元的操作之前清除未处理的假中断标志。最后，应用程序需要通过配置设备时钟管理逻辑的相应控制位来为 USB 模块提供标准所定义的 48MHz 时钟。

当系统复位时，应用程序应该初始化所有需要的寄存器和分组缓冲区描述表，使 USB 模块能够产生正常的中断和完成数据传输。所有与端点无关的寄存器需要根据应用的需求进行初始化(比如中断使能的选择，分组缓冲区地址的选择等)。接下来，就进入 USB 复位状态(参考以下章节)。

USB复位(RESET 中断)

发生 USB 复位时，USB 模块进入前面章节中描述过的系统复位状态：所有端点的通信都被禁止(USB 模块不会响应任何分组)。在 USB 复位后，USB 模块被使能，同时地址为 0 的默认控制端点(端点 0)也需要被使能。这可以通过配置 USB_DADDR 寄存器的 EF 位，EP0R 寄存器和相关的分组缓冲区来实现。在 USB 设备的枚举阶段，主机将分配给设备一个唯一的地址，这个地址必须写入 USB_DADDR 寄存器的 ADD[6:0]位中，同时配置其他所需的端点。

当复位中断产生时，应用程序必需在中断产生后的 10ms 之内使能端点 0 的传输。

分组缓冲区的结构和用途

每个双向端点都可以接收或发送数据。接收到的数据存储在该端点指定的专用缓冲区内，而另一个缓冲区则用于存放待发送的数据。对这些缓冲区的访问由分组缓冲区接口模块实现，它提出缓冲区访问请求，并等待确认信息后返回。为防止产生微控制器与 USB 模块对缓冲区的访问冲突，缓冲区接口模块使用仲裁机制，使 APB1 总线的一半周期用于微控制器的访问，另一半保证 USB 模块的访问。这样，微控制器和 USB 模块对分组缓冲区的访问如同对一个双端口 SRAM 的访问，即使微控制器连续访问缓冲区，也不会产生访问冲突。

USB 模块使用固定的时钟，此时钟被 USB 标准定义为 48MHz。APB1 总线的时钟可以大于或者小于这个频率。

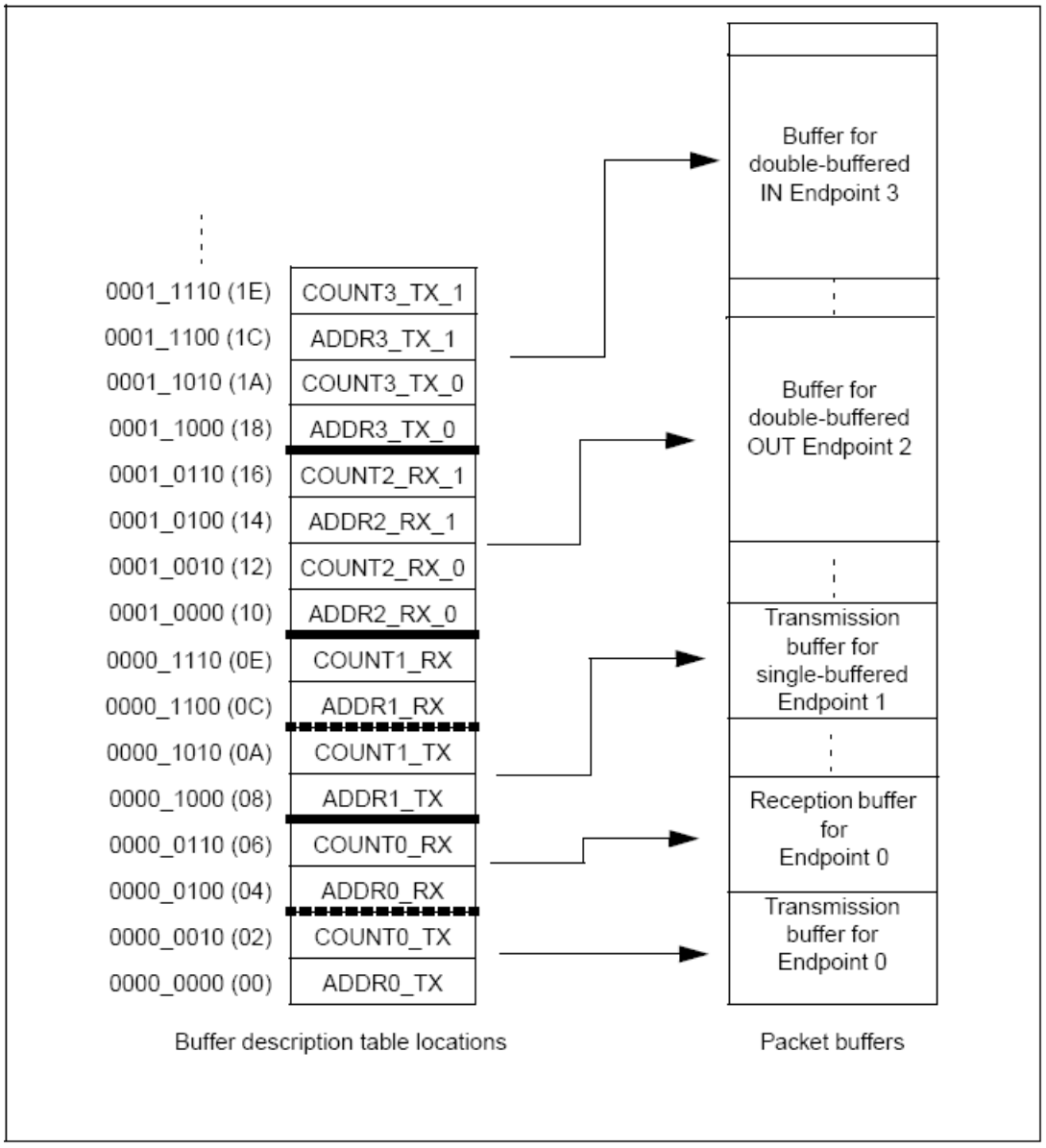
注意：　为满足 USB 数据传输率和分组缓冲区接口的系统需求，APB1 总线时钟的频率必须大于 8MHz，以避免数据缓冲区溢出或不满

每个端点对应于两个分组缓冲区(一般一个用于发送，另一个用于接收)。这些缓冲区可以位于整个分组存储区的任意位置，因为它们的地址和长度都定义在缓冲区描述表中，而缓冲区描述表也同样位于分组缓冲区中，其地址由寄存器确定。

缓冲区描述表的每个表项都关联到一个端点寄存器，它由 4 个 16 位的字组成，因此缓冲区描述表的起始地址按 8 字节对齐(寄存器的最低 3 位总是“000”)。缓冲区描述表表项的详细说明在 17.6.3 节里介绍。如果是非同步非双缓冲的单向端点，只需要一个分组缓冲区(即发送方向上的分组缓冲区)。

其他未用到的端点或某个未使用的方向上的缓冲区描述表项可以用于其他用途。同步和双缓冲批量端点有特殊的分组缓冲区处理方法(请分别参考 17.5.4 节：同步传输和 17.5.3 节：双缓冲端点)。图 160 描述了缓冲区描述表项和分组缓冲区区域的关系。

图160 分组缓冲区对应的缓冲区描述表项定位



不管是接收还是发送，分组缓冲区都是从底部开始使用的。USB 模块不会改变超出当前分配到的缓冲区区域以外的其他缓冲区的内容。如果缓冲区收到一个比自己大的数据分组，它只会接收最大为自身大小的数据，其他的丢掉，即发生了所谓的缓冲区溢出异常。

端点初始化

初始化端点的第一步是把适当的值写到 ADDRn_TX 或 ADDRn_RX 寄存器中，以便 USB 模块能找到要传输的数据或准备好接收数据的缓冲区。USB_EpnR 寄存器的 EP_TYPE 位确定端点的基本类型，EP_KIND 位确定端点的特殊特性。作为发送方，需要设置 USB_EpnR 寄存器的 STAT_TX 位来使能端点，并配置 COUNTn_TX 位决定发送长度。作为接收方，需要设置 STAT_RX 位来使能端点，并且设置 BL_SIZE 和 NUM_BLOCK 位，确定接收缓冲区的大小，以检测缓冲区溢出的异常。对于非同步非双缓冲批量传输的单向端点，只需要设置一个传输方向上的寄存器。一旦端点被使能，应用程序就不能再修改 USB_EpnR 寄存器的值和 ADDRn_TX / ADDRn_RX, COUNTn_TX / COUNTn_RX 所在的位置，因为这些值会被硬件实时修改。当数据传输完成时，CTR 中断会产生，此时上述寄存器可以被访问，并重新使能新的传输。

IN分组（用于数据发送）

当接收到一 IN 令牌分组时，如果接收到的地址和一个配置好的端点地址相符合的话，USB 模块将会根据缓冲区描述表的表项，访问相应的 ADDRn_TX 和 COUNTn_TX 寄存器，并将这些寄存器中的数值存储到内部的 16 位寄存器 ADDR 和 COUNT(应用程序无法访问)中。此时，USB 模块开始根据 DTOG_TX 位发送 DATA0 或 DATA1 分组，并访问缓冲区(请参考分组缓冲区的结构和用途章节)。在 IN 分组传输完毕之后，从缓冲区读到的第一个字节将被装载到输出移位寄存器中，并开始发送。最后一个数据字节发送完成之后，计算好的 CRC 将被发送。如果收到的分组所对应的端点是无效的，将根据 USB_EpnR 寄存器上的 STAT_TX 位发送 NAK 或 STALL 握手分组而不发送数据。

ADDR 内部寄存器被用作当前缓冲区的指针，COUNT 寄存器用于记录剩下未传输的字节数。USB 总线使用低字节在先的方式传输从缓冲区中读出的数据。数据从 ADDRn_TX 指向的数据分组缓冲区开始读取，长度为 COUNTn_TX/2 个字。如果发送的数据分组为奇数个字节，则只使用最后一个字的低 8 位。

在接收到主机响应的 ACK 后，USB_EpnR 寄存器的值有以下更新：DTOG_TX 位被翻转，STAT_TX 位为 10，使端点无效，CTR_TX 位被置位。应用程序需要通过 USB_ISTR 寄存器的 EP_ID 和 DIR 位识别产生中断的 USB 端点。CTR_TX 事件的中断服务程序需要首先清除中断标志位，然后准备好需要发送的数据缓冲区，更新 COUNTn_TX 为下次需要传输的字节数，最后再设置 STAT_TX 位为 11(端点有效)，再次使能数据传输。当 STAT_TX 位为 10 时(端点为 NAK 状态)，任何发送到该端点的 IN 请求都会被 NAK，USB 主机会重发 IN 请求直到该端点确认请求有效。上述操作过程是必需遵守的，以避免丢失紧随上一次 CTR 中断请求的下一个 IN 传输请求。

OUT分组和SETUP分组（用于数据接收）

USB 模块对这两种分组的处理方式基本相同；对 SETUP 分组的特殊处理将在下面关于控制传输的章节中详细说明。当接收到一个 OUT 或 SETUP 分组时，如果地址和某个有效端点的地址相匹配，USB 模块将访问缓冲区描述表，找到与该端点相关的 ADDRn_RX 和 COUNTn_RX 寄存器，并将 ADDRn_RX 寄存器的值保存在内部 ADDR 寄存器中。同时，COUNT 会被被复位，从 COUNTn_RX 中读

出的 BL_SIZE 和 NUM_BLOCK 的值用于初始化内部 16 位寄存器 BUF_COUNT，该寄存器用于检测缓冲区溢出（所有的内部寄存器都不能被应用程序访问）。USB 模块将随后收到的数据按字方式组织(先收到的为低字节)，并存储到 ADDR 指向的分组缓冲区中。同时，BUF_COUNT 的值自动递减，COUNT 值自动递增。当检测到数据分组的结束信号时，USB 模块校验收到的 CRC 的正确性。如果传输中没有任何错误发生，ACK 握手分组会被发送到主机。即使发生 CRC 错误或者其他类型的错误(位填充，帧错误等)，数据还是会被保存到分组缓冲区中，至少会保存到发生错误的数据点，只是不会发送 ACK 分组，并且 USB_ISTR 寄存器的 ERR 位将会置位。在这种情况下，应用程序通常不需要干涉处理，USB 模块将从传输错误中自动恢复，并为下一次传输做好准备。如果收到的分组所对应的端点没有准备好，USB 模块将根据 USB_EpnR 寄存器的 STAT_RX 位发送 NAK 或 STALL 分组，数据将不会被写入接收缓冲区。

ADDRn_RX 的值决定接收缓冲区的起始地址，长度由包含 CRC 的数据分组的长度（即有效数据长度+2）决定，但不能超过 BL_SIZE 和 NUM_BLOCK 所定义的缓冲区的长度。如果接收到的数据分组的长度超出了缓冲区的范围，超过范围的数据不会被写入缓冲区，USB 模块将报告缓冲区发生溢出，并向主机发送 STALL 握手分组，通知此次传输失败，也不产生中断。

如果传输正确完成，USB 模块将发送 ACK 握手分组，内部的 COUNT 寄存器的值会被复制到相应的 COUNTn_RX 寄存器中，BL_SIZE 和 NUM_BLOCK 的值保持不变，也不需要重写。USB_EpnR 寄存器按下列方式更新：DTOG_RX 位翻转，STAT_RX=10（NAK）使端点无效，CTR_RX 位置位(如果 CTR 中断已使能，将触发中断)。如果传输过程中发生了错误或者缓冲区溢出，前面所列出的动作都不会发生。CRT 中断发生时，应用程序需要首先根据 USB_ISTR 寄存器的 EP_ID 和 DIR 位识别是哪个端点的中断请求。在处理 CTR_RX 中断事件时，应用程序首先要确定传输的类型（根据 USB_EPnR 寄存器的 SETUP 位），同时清除中断标志位，然后读相关的缓冲区描述表表项指向的 COUNTn_RX 寄存器，获得此次传输的总字节数。处理完接收到的数据后，应用程序需要将 USB_EpnR 中的 STAT_RX 位置成 11，使能下一次的的传输。当 STAT_RX 位为 10 时(NAK)，任何一个发送到端点上的 OUT 请求都会被 NAK，PC 主机将不断重发被 NAK 的分组，直到收到端点的 ACK 握手分组。以上描述的操作次序是必需遵守的，以避免丢失紧随上一个 CTR 中断的另一个 OUT 分组请求。

控制传输

控制传输由 3 个阶段组成，首先是主机发送 SETUP 分组的 SETUP 阶段，然后是主机发送零个或多个数据的数据阶段，最后是状态阶段，由与数据阶段方向相反的数据分组构成。SETUP 传输只发生在控制端点，它非常类似于 OUT 分组的传输过程。使能 SETUP 传输除了需要分别初始化 DTOG_TX 位为 1，DTOG_RX 位为 0 外，还需要设置 STAT_TX 位和 STAT_RX 位为 10(NAK)，由应用程序根据 SETUP 分组的相应字段决定后面的传输是 IN 还是 OUT。控制端点在每次发生 CTR_RX 中断时，都必须检查 USB_EpnR 寄存器的 SETUP 位，以识别是普通的 OUT 分组还是 SETUP 分组。USB 设备应该能够通过 SETUP 分组中的相应数据决定数据阶段传输的字节数和方向，并且能在发生错误的情况下发送 STALL 分组，拒绝数据的传输。因此在数据阶段，未被使用到的方向都应该被设置成 STALL，并且在开始传输数据阶段的最后一个数据分组时，其反方向的传输仍设成 NAK 状态，这样，即使主机立刻改变了传输方向(进入状态阶段)，仍然可以保持为等待控制传输结束的状态。在控制传输成功结束后，应用程序可以

把 NAK 变为 VALD，如果控制传输出错，就改为 STALL。此时，如果状态分组是由主机发送给设备的，那么 STATUS_OUT 位(USB_EPnR 寄存器中的 EP_KIND)应该被置位，只有这样，在状态传输过程中收到了非零长度的数据分组，才会产生传输错误。在完成状态传输阶段后，应用程序应该清除 STATUS_OUT 位，并且将 STAT_RX 设为 VALID 表示已准备好接收一个新的命令请求，STAT_TX 则设为 NAK，表示在下一个 SETUP 分组传输完成前，不接受数据传输的请求。

USB 规范定义 SETUP 分组不能以非 ACK 握手分组来响应，如果 SETUP 分组传输失败，则会引发下一个 SETUP 分组。因此，以 NAK 或 STALL 分组响应主机的 SETUP 分组是被禁止的。

当 STAT_RX 位被设置为 01(STALL)或 10(NAK)时，如果收到 SETUP 分组，USB 模块会接收分组，开始分组所要求的数据传输，并回送 ACK 握手分组。如果应用程序在处理前一个 CTR_RX 事件时 USB 模块又收到了 SETUP 分组(即 CTR_RX 仍然保持置位)，USB 模块会丢掉收到的 SETUP 分组，并且不回答任何握手分组，以此来模拟一个接收错误，迫使主机再次发送 SETUP 分组。这样做是为了避免丢失紧随一次 CTR_RX 中断之后的又一个 SETUP 分组传输。

17.5.3 双缓冲端点

USB 标准不仅为不同的传输模式定义了不同的端点类型，而且对这些数据传输所需要的系统要求做了描述。其中，批量端点适用于在主机 PC 和 USB 设备之间传输大批量的数据，因为主机可以在一帧内利用尽可能多的带宽批量传输数据，使传输效率得到提高。然而，当 USB 设备处理前一次的数据传输时，又收到新的数据分组，它将回应 NAK 分组，使 PC 主机不断重发同样的数据分组，直到设备在可以处理数据时回应 ACK 分组。这样的重传占用了很多带宽，影响了批量传输的速率，因此引入了批量端点的双缓冲机制，提高数据传输率。

使用双缓冲机制时，单向端点的数据传输将使用到该端点的接收和发送两块数据缓冲区。数据翻转位用来选择当前使用到两块缓冲区中的哪一块，使应用程序可以在 USB 模块访问其中一块缓冲区的同时，对另一块缓冲区进行操作。例如，对一个双缓冲批量端点进行 OUT 分组传输时，USB 模块将来自 PC 主机的数据保存到一个缓冲区，同时应用程序可以对另一个缓冲区中的数据进行处理(对于 IN 分组来说，情况是一样的)。

因为切换缓冲区的管理机制需要用到所有 4 个缓冲区描述表的表项，分别用来表示每个方向上的两个缓冲区的地址指针和缓冲区大小，因此用来实现双缓冲批量端点的 USB_EpnR 寄存器必需配置为单向。所以，只需要设定 STAT_RX 位（作为双缓冲批量接收端点）或者 STAT_TX 位(作为双缓冲批量发送端点)。如果需要一个双向的双缓冲批量端点，就必须使用两个 USB_EpnR 寄存器。

为尽可能利用双缓冲的优势，达到较高的传输速率，双缓冲批量端点的流量控制流程与其他端点的稍有不同。它只在缓冲区发生访问冲突时才会设置端点为 NAK 状态，而不是在每次传输成功后都将端点设为 NAK 状态。

DTOG 位用来标识 USB 模块当前所使用的储存缓冲区。双缓冲批量端点接收方向的缓冲区由 DTOG_RX(USB_EpnR 寄存器的第 14 位)标识，而双缓冲批量端点发送方向的缓冲区由 DTOG_TX(USB_EpnR 寄存器的第 6 位)标识。同时，USB 模块也需要知道当前哪个缓冲区正在被应用程序使用，以避免发生冲突。由

于 USB_EpnR 寄存器中有 2 个 DTOG 位，而 USB 模块只使用其中的一位来标识硬件所使用的缓冲区，因此，应用程序可使用另一位来标识当前正在使用哪个缓冲区，这个新的标识被称为 SW_BUF 位。下表列出了双缓冲批量端点在实现发送和接收操作时，USB_EPNR 寄存器的 DTOG 位和 SW_BUF 位之间的关系。

表54 双缓冲批量端点缓冲区标识定义

缓冲区标识位	作为发送端点	作为接收端点
DTOG	DTOG_TX（USB_EPNR寄存器的第6位）	DTOG_RX（USB_EPNR寄存器的第14位）
SW_BUF	USB_EPNR寄存器的第14位	USB_EPNR寄存器的第6位

USB 模块当前使用的缓冲区由 DTOG 位标识，而应用程序所使用的缓冲区由 SW_BUF 位标识，这两个位的标识方式相同，下表描述了这种标识方式。

表55 双缓冲批量端点的缓冲区使用标识

端点类型	DTOG 位	SW_BUF 位	USB模块使用的缓冲区	应用程序使用的缓冲区
IN端点	0	1	ADDRn_TX_0 / COUNTn_TX_0	ADDRn_TX_1 / COUNTn_TX_1
	1	0	ADDRn_TX_1 / COUNTn_TX_1	ADDRn_TX_0 / COUNTn_TX_0
	0	0	端点处于NAK状态	ADDRn_TX_0 / COUNTn_TX_0
	1	1	端点处于NAK状态	ADDRn_TX_0 / COUNTn_TX_0
OUT端点	0	1	ADDRn_RX_0 / COUNTn_RX_0	ADDRn_RX_1 / COUNTn_RX_1
	1	0	ADDRn_RX_1 / COUNTn_RX_1	ADDRn_RX_0 / COUNTn_RX_0
	0	0	端点处于NAK状态	ADDRn_RX_0 / COUNTn_RX_0
	1	1	端点处于NAK状态	ADDRn_RX_0 / COUNTn_RX_0

可以通过以下方式设置一个双缓冲批量端点：

- 将 USB_EPnR 寄存器的 EP_TYPE 位设为 00，定义端点为批量端点
- 将 USB_EPnR 寄存器的 EP_KIND 位设为 1，定义端点为双缓冲端点

应用程序根据传输开始时用到的缓冲区来初始化DTOG和SW_BUF位；这需要考虑到这两位的数据翻转特性。设置好DBL_BUF位之后，每完成一次传输后，USB模块将根据双缓冲批量端点的流量控制操作，并且持续到DBL_BUF变为无效为止。每次传输结束，根据端点的传输方向，CTR_RX位或CTR_TX位将会置为1。与此同时，硬件将设置相应的DTOG位，完全独立于软件来实现缓冲区交换机制。DBL_BUF位设置后，每次传输结束时，双缓冲批量端点的STAT位的取值不会像其他类型端点一样受到传输过程的影响，而是一直保持为“11”(有效)。但是，如果在收到新的数据分组的传输请求时，USB模块和应用程序发生了缓冲区访问冲突(即DTOG和SW_BUF为相同的值，见 表 55)，状态位将会被置为“10”(NAK)。应用程序响应CTR中断时，首先要清除中断标志，然后再处理传输完成的数据。应用程序访问缓冲区之后，需要翻转SW_BUF位，以通知USB模块该块缓冲区已变为可用状态。由此，双缓冲批量传输的NAK分组的数目只由应用程序处理一次数据传输的快慢所决定：如果数据处理的时间小于USB总线上完成一次数据传输的时间，则不会发生重传，此时，数据的传输率仅受限于USB主机。

应用程序也可以不考虑双缓冲批量端点的特殊控制流程，直接在相应 USB_EPnR 寄存器的 STAT 位写入非“11”的任何状态，在这种情况下，USB 模块将按照写入的状态执行流程而忽略缓冲器实际的使用情况。

17.5.4 同步传输

USB 标准定义了一种全速的需要保持固定和精确的数据传输率的传输方式：同步传输。同步传输一般用于传输音频流、压缩的视频流等对数据传输率有严格要求

的数据。一个端点如果在枚举时被定义为“同步端点”，USB 主机则会为每个帧分配固定的带宽，并且保证每个帧正好传送一个 IN 分组或者 OUT 分组（由端点传输方向确定分组类型）。为了满足带宽要求，同步传输中没有出错重传；这也就意味着，同步传输在发送或接收数据分组之后，无握手协议，即不会发送 ACK 分组。同样，同步传输只传送 PID（分组 ID）为 DATA0 的数据包, 而不会用到数据翻转机制。

通过设置 USB_EPnR 寄存器 EP_TYPE 为“10”，可以使其成为同步端点。同步端点没有握手机制，根据 USB 标准中的说明，USB_EPnR 寄存器的 STAT_RX 位和 STAT_TX 位分别只能设成‘00’（禁止）和‘11’（有效）。同步传输通过实现双缓冲机制来简化软件应用程序开发，它同样使用两个缓冲区，以确保在 USB 模块使用其中一块缓冲区时，应用程序可以访问另外一块缓冲区。

USB 模块使用的缓冲区根据不同的传输方向，由不同的 DTOG 位来标识。（同一寄存器中的 DTOG_RX 位用来标识接收同步端点，DTOG_TX 位用来标识发送同步端点），见下表。

表56 同步端点的缓冲区使用标识

端点类型	DTOG位值	USB模块使用的缓冲区	应用程序使用的缓冲区
IN端点	0	ADDRn_TX_0 / COUNTn_TX_0	ADDRn_TX_1 / COUNTn_TX_1
	1	ADDRn_TX_1 / COUNTn_TX_1	ADDRn_TX_0 / COUNTn_TX_0
OUT端点	0	ADDRn_RX_0 / COUNTn_RX_0	ADDRn_RX_1 / COUNTn_RX_1
	1	ADDRn_RX_1 / COUNTn_RX_1	ADDRn_RX_0 / COUNTn_RX_0

与双缓冲批量端点一样，一个 USB_EPnR 寄存器只能处理同步端点单方向的数据传输，如果要求同步端点在两个传输方向上都有效，则需要使用两个 USB_EPnR 寄存器。

应用程序需要根据首次传输的数据分组来初始化 DTOG 位；它的取值还需要考虑到 DTOG_RX 或 DTOG_TX 两位的数据翻转特性。每次传输完成时，USB_EPnR 寄存器的 CTR_RX 位或 CTR_TX 位置位。与此同时，相关的 DTOG 位由硬件翻转，从而使得交换缓冲区的操作完全独立于应用程序。传输结束时，STAT_RX 或 STAT_TX 位不会发生变化，因为同步传输没有握手机制，所以不需要任何流量控制，而一直设为“11”（有效）。同步传输中，即使 OUT 分组发生 CRC 错误或者缓冲区溢出，本次传输仍被看作是正确的，并且可以触发 CTR_RX 中断事件；但是，发生 CRC 错误时硬件会设置 USB_ISTR 寄存器的 ERR 位，提醒应用程序数据可能损坏。

17.5.5 挂起/恢复事件

USB 标准中定义了一种特殊的设备状态，即挂起状态，在这种状态下 USB 总线上的平均电流消耗不超过 500uA。这种电流限制对于由总线供电的 USB 设备至关重要，而自供电的设备则不需要严格遵守这样的电流消耗限制。USB 主机以 3 毫秒内不发送任何信号标志进入挂起状态。通常情况下 USB 主机每毫秒会发送一

个 SOF，当 USB 模块检测到 3 个连续的 SOF 分组丢失事件即可判定主机发出了挂起请求，接着它会置位 SB_ISTR 寄存器的 SUSP 位，以触发挂起中断。USB 设备进入挂起状态之后，将由“唤醒”序列唤醒。所谓的“唤醒”序列, 可以由 USB 主机发起，也可以由 USB 设备本身触发；但是，只有 USB 主机可以结束“唤醒”序列。被挂起的 USB 模块必须至少还具备检测 RESET 信号的功能，它会将其当作一次正常的复位操作来执行。

实际的挂起操作过程对于不同的 USB 设备来说是不同的，因为需要不同的操作来降低电源消耗。下面描述了一起典型的挂起操作，重点介绍应用程序如何响应 USB 模块的 SUSP 信号。

1. 将USB_CNTR寄存器的FSUSP置为1，这将使USB模块进入挂起状态。USB模块一旦进入挂起状态，对SOF的检测立刻停止，以避免在USB挂起时又发生新的SUSP事件。
2. 消除或减少USB模块以外的其他模块的静态电流消耗。
3. 将USB_CNTR寄存器的LP_MODE位置为1，这将消除模拟USB收发器的静态电流消耗，但仍能检测到唤醒信号。
4. 可以选择关闭外部振荡器和设备的PLL，以停止设备内部的任何活动。

当设备处于挂起状态时发生 USB 事件，该设备会被唤醒，并需要调用“唤醒”例程来恢复系统时钟，和 USB 数据传输。如果唤醒设备的是 USB 复位操作，则应该保证唤醒的过程不要超过 10 毫秒(参见“USB 协议规范”)。USB 模块处于挂起状态时，唤醒或复位事件需要清除 USB_CNTR 寄存器的 LP_MODE 位。即使唤醒事件可以立刻触发一个 WKUP 中断事件，但由于恢复系统时钟需要比较长的延迟时间，处理 WKUP 中断的中断服务程序必须非常小心；为了减短系统唤醒的时间，建议将唤醒代码直接写在挂起代码后面，这样就可以在系统时钟重启后迅速进入唤醒代码中执行。为防止或减少 ESD 等干扰意外地唤醒系统（从挂起模式退出是一个异步事件），在挂起过程中数据线被过滤，滤波宽度大约为 70nS。

下面是唤醒操作的过程：

启动外部振荡器和设备的PLL（此项可选）。

1. 清零USB_CNTR寄存器的FSUSP位。
2. USB_FNR寄存器的RXDP和RXDM位可以用来判断是什么触发了唤醒事件，如表57所示，它还同时列出了各种情况软件应该采取的操作。如果需要的话，可以通过检测这两位变成“10”（代表空闲总线状态）的时间来知道唤醒或复位事件的结束。此外，在复位事件结束时，USB_ISTR寄存器的RESET位被置为1，如果RESET中断被使能，就会产生中断。此中断应该按正常的复位操作处理。

表57 唤醒事件检测

[RXDP，RXDM]的状态	唤醒事件	应用程序应执行的操作
00	复位	无
10	无（总线干扰）	恢复到挂起状态
01	恢复挂起	无

11	未定义的值（总线干扰）	恢复到挂起状态

设备可能不是被与 USB 模块相关的事件唤醒的（例如一个鼠标的移动可唤醒整个系统）。在这种情况下，先将 USB_CNTR 寄存器的 RESUME 位置为 1，然后在 1mS－15mS 之间再把它清为 0 可以启动唤醒序列（这个间隔可以用 ESOF 中断来实现，该中断在内核正常运行时每 1ms 发生一次）。RESUME 位被清零后，唤醒过程将由主机 PC 完成，可以利用 USB_FNR 寄存器的 RXDP 和 RXDM 位来判断唤醒是否完成。

注意：　只有在 USB 模块被设置为挂起状态时（设置 USB_CNTR 寄存器的 FSUSP 位为 1），才可以设置 RESUME 位。

17.6 USB寄存器描述

USB 模块的寄存器有以下三类:：

- 通用类寄存器：中断寄存器和控制寄存器
- 端点类寄存器：端点配置寄存器和状态寄存器
- 缓冲区描述表类寄存器：用来确定数据分组存放地址的寄存器

缓冲区描述表类寄存器的基地址由 USB_BTABLE 寄存器指定，所有其他寄存器的基地址则为 USB 模块的基地址 0xC000 8000。由于 APB1 总线按 32 位寻址，因此所有的 16 位寄存器的地址都是按 32 位字对齐的。同样的地址对齐方式也用于从 0xC000 8800 开始的分组缓冲存储区。

寄存器描述中使用的一些缩略语请参考 1.1 节。

17.6.1 通用寄存器

这类寄存器用于定义 USB 模块的工作模式，中断的处理，设备的地址和读取当前帧的编号。

USB控制寄存器

地址偏移：0x40
复位值：0x0003

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CTRM	PMA OVRM	ERRM	WKUPM	SUSPM	RESET M	SOFM	ESOFM	保留			RESUM E	FSUSP	LP MODE	PDMN	FRES
rw	rw	rw	rw	rw	rw	rw	rw				rw	rw	rw	rw	rw

位15	CTRM：正确传输（CTR）中断屏蔽位 0：正确传输（CTR）中断禁止 1：正确传输（CTR）中断使能，在中断寄存器的相应位被置1时产生中断。
位14	PMAOVRM：分组缓冲区溢出中断屏蔽位 0：PMAOVR中断禁止 1：PMAOVR中断使能，在中断寄存器的相应位被置1时产生中断。
位13	ERRM：出错中断屏蔽位 0：出错中断禁止 1：出错中断使能，在中断寄存器的相应位被置1时产生中断。
位12	WKUPM：唤醒中断屏蔽位 0：唤醒中断禁止 1：唤醒中断使能，在中断寄存器的相应位被置1时产生中断。
位11	SUSPM：挂起中断屏蔽位 0：挂起（SUSP）中断禁止 1：挂起（SUSP）中断使能，在中断寄存器的相应位被置1时产生中断。
位10	RESETM：USB复位中断屏蔽位 0：USB RESET中断禁止 1：USB RESET中断使能，在中断寄存器的相应位被置1时产生中断。
位9	SOFM：帧首中断屏蔽位 0：SOF中断禁止 1：SOF中断使能，在中断寄存器的相应位被置1时产生中断。
位8	ESOFM：期望帧首中断屏蔽位 0：ESOF中断禁止 1：ESOF中断使能，在中断寄存器的相应位被置1时产生中断。
位4	RESUME：唤醒请求 设置此位将向PC主机发送唤醒请求。根据USB协议，如果此位在1ms到15ms内保持有效，主机将对USB模块实行唤醒操作。
位3	FSUSP：强制挂起 当USB总线上保持3ms没有数据通信时，SUSP中断会被触发，此时软件必需设置此位。 0：无效 1：进入挂起模式，USB模拟收发器的时钟和静态功耗仍然保持。如果需要进入低功耗状态（总线供电类的设备），应用程序需要先置位FSUSP再置位LP_MODE。
位2	LP_MODE：低功耗模式 此模式用于在USB挂起状态下降低功耗。在此模式下，除了外接上拉电阻的供电，其他的静态功耗都被关闭，系统时钟将会停止或者降低到一定的频率来减少耗电。USB总线上的活动（唤醒事件）将会复位此位（软件也可以复位此位）。 0：非低功耗模式 1：低功耗模式

位1	PDWN：断电模式 此模式用于彻底关闭USB模块。当此位被置位时，不能使用USB模块。 0：退出断电模式 1：进入断电模式
位0	FRES：强制USB复位 0：清除USB复位信号 1：对USB模块强制复位，类似于USB总线上的复位信号。USB模块将一直保持在复位状态下直到软件清除此位。如果USB复位中断被使能，将产生一个复位中断。

USB中断状态寄存器

地址偏移：0x44
复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CTR	PMAOVR	ERR	WKUP	SUSP	RESET	SOF	ESOF	保留			DIR	EP_ID[3:0]			
r	rc	rc	rc	rc	rc	rc	rc				r	r	r	r	r

此寄存器包含所有中断源的状态信息，以供应用程序确认产生中断请求的事件。

寄存器的高 8 位各表示一个中断源。当相关事件发生时，这些位被硬件置位，如果 USB_CNTR 寄存器上的相应位也被置位，则会产生相应的中断。中断服务程序需要检查每个位，在执行必要的操作后必需清除相应的状态位，不然中断信号线一直保持为高，同样的中断会再次被触发。如果同时多个中断标志被设置，也只会产生一个中断。

应用程序可以使用不同的方式处理传输完成中断，以减少中断响应的延迟时间。端点在成功完成一次传输后，CTR 位会被硬件置起，如果 USB_CNTR 上的相应位也被设置的话，就会产生中断。与端点相关的中断标志和 USB_CNTR 寄存器的 CTRM 位无关。这两个中断标志位将一直保持有效，直到应用程序清除了 USB_EpnR 寄存器中的相关中断挂起位（CTR 位是个只读位）。USB 模块有两路中断请求源：

高优先级的 USB IRQ：用于高优先级的端点（同步和双缓冲批量端点）的中断请求，并且该中断不能被屏蔽。

低优先级 USB IRQ：用于其他中断事件，可以是低优先级的不可屏蔽中断，也可以是由 USB_ISTR 寄存器的高 8 位标识的可屏蔽中断。

对于端点产生的中断，应用程序可以通过 DIR 寄存器和 EP_ID 只读位来识别中断请求由哪个端点产生，并调用相应的中断服务程序。

用户在处理同时发生的多个中断事件时，可以在中断服务程序里检查 USB_ISTR 寄存器各个位的顺序来确定这些事件的优先级。在处理完相应位的中断后需要清零该中断标志。完成一次中断服务后，另一中断请求将会产生，用以请求处理剩下的中断事件。

为了避免意外清零某些位，建议使用加载指令，对所有不需改变的位写 1，对需要清除的位写 0。对于该寄存器，不建议使用读出－修改－写入的流程，因为在读写操作之间，硬件可能需要设置某些位，而这些位会在写入时被清零。

下面详细描述每个位：

位15	CTR：正确的传输 此位在端点正确完成一次数据传输后由硬件置位。应用程序可以通过DIR和EP_ID位来识别是哪个端点完成了正确的数据传输。 此位应用程序只读
位14	PMAOVR 分组缓冲区溢出 此位在微控制器长时间没有响应一个访问USB分组缓冲区请求时由硬件置位。USB模块通常在以下情况时置位该位：在接收过程中一个ACK握手分组没有被发送，或者在发送过程中发生了比特填充错误，在以上两种情况下主机都会要求数据重传。在正常的数据传输中不会产生PMAOVR中断。由于失败的传输都将由主机发起重传，应用程序就可以在这个中断的服务程序中加速设备的其他操作，并准备重传。但这个中断不会在同步传输中产生（同步传输不支持重传）因此数据可能会丢失。 此位应用程序可读可写，但只有写0有效，写1无效。
位13	ERR：出错 在下列错误发生时硬件会置位此位。 NANS：无应答。主机的应答超时。 CRC：循环冗余校验码错误。数据或令牌分组中的CRC校验出错。 BST：位填充错误。PID，数据或CRC中检测出位填充错误。 FVIO：帧格式错误。收到非标准帧（如EOP出现在错误的时刻，错误的令牌等）。 USB应用程序通常可以忽略这些错误，因为USB模块和主机在发生错误时都会启动重传机制。此位产生的中断可以用于应用程序的开发阶段，可以用来监测USB总线的传输质量，标识用户可能发生的错误（连接线松，环境干扰严重，USB线损坏等）。 此位应用程序可读可写，但只有写0有效，写1无效。
位12	WKUP：唤醒请求 当USB模块处于挂起状态时，如果检测到唤醒信号，此位将由硬件置位。此时CTLR寄存器的LP_MODE位将被清零，同时USB_WAKEUP被激活，通知设备的其他部分（如唤醒单元）将开始唤醒过程。 此位应用程序可读可写，但只有写0有效，写1无效。
位11	SUSP：挂起模块请求 此位在USB线上超过3ms没有信号传输时由硬件置位，用以指示一个来自USB总线的挂起请求。USB复位后硬件立即使能对挂起信号的检测，但在挂起模式下（FSUSP＝1）硬件不会再检测挂起信号直到唤醒过程结束。 此位应用程序可读可写，但只有写0有效，写1无效。
位10	RESET：USB复位请求 此位在USB模块检测到USB复位信号输入时由硬件置位。此时USB模块将复位内部协议状态机，并在中断使能的情况下触发复位中断来响应复位信号。USB模块的发送和接收部分将被禁止，直到此位被清除。所有的配置寄存器不会被复位，除非应用程序对他们清零。这用来保证在复位后USB传输还可以立即正确执行。但设备的地址和端点寄存器会被USB复位所复位。 此位应用程序可读可写，但只有写0有效，写1无效。
位9	SOF：帧首标志 此位在USB模块检测到总线上的SOF分组时由硬件置位，标志一个新的USB帧的开始。中断服务程序可以通过检测SOF事件来完成与主机的1ms同步，并正确读出寄存器在收到SOF分组时的更新内容（此功能在同步传输时非常有意义）。 此位应用程序可读可写，但只有写0有效，写1无效。

位8	ESOF：期望帧首标识位 此位在USB模块未收到期望的SOF分组时由硬件置位。主机应该每毫秒都发送SOF分组，但如果USB模块没有收到，挂起定时器将触发此中断。如果连续发生3次ESOF中断，也就是连续3次未收到SOF分组，将产生SUSP中断。即使在挂起定时器未被锁定时发生SOF分组丢失，此位也会被置位。 此位应用程序可读可写，但只有写0有效，写1无效。
位7:5	保留
位4	DIR：传输方向 此位在完成数据传输产生中断后由硬件根据传输方向写入。 如果DIR＝0，相应端点的CTR_TX位被置位，标志一个IN分组（数据从USB模块传输到PC主机）的传输完成。 如果DIR＝1，相应端点的CTR_RX位被置位，标志一个OUT分组（数据从PC主机传输到USB模块）的传输完成。如果CTR_TX位同时也被置位，就标志同时存在挂起的OUT分组和IN分组。 应用程序可以通过访问此位获得传输方向的信息。 此位应用程序只读
位3:0	EP_ID[3:0]：端点ID。 此位在USB模块完成数据传输产生中断后由硬件根据请求中断的端点号写入。如果同时有多个端点的请求中断，硬件写入优先级最高的端点号。端点的优先级按以下方法定义：同步端点和双缓冲批量端点具有高优先级，其他的端点为低优先级。如果多个同优先级的端点请求中断，则根据端点号来确定优先级，即端点0具有最高优先级，端点号越小，优先级越高。应用程序可以通过上述的优先级策略顺序处理端点的中断请求。 此位应用程序只读。

USB帧编号寄存器

地址偏移：0x48
复位值：0x0XXX，X 代表未定义数值

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXDP	RXDM	LCK	LSOF[1:0]		FN[10:0]										
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

位15	RXDP：D+状态位 此位用于观察USB D+数据线的状态，可在挂起状态下检测唤醒条件的出现。
位14	RXDM：D-状态位 此位用于观察USB D-数据线的状态，可在挂起状态下检测唤醒条件的出现。
位13	LCK：锁定位 USB模块在复位或唤醒序列结束后会检测SOF分组，如果连续检测到至少2个SOF分组，则硬件会置位此位。此位一旦锁定，帧计数器将停止计数，一直等到USB模块复位或总线挂起时再恢复计数。
位12:11	LSOF[1:0]：帧首丢失标志位 当ESOF事件发生时，硬件会将丢失的SOF分组的数目写入此位。如果再次收到SOF分组，引脚会清除此位。

位10:0	FN[10:0]：帧编号 此部分记录了最新收到的SOF分组中的11位帧编号。主机每发送一个帧，帧编号都会自加，这对于同步传输非常有意义。此部分发生SOF中断时更新。

USB设备地址寄存器

地址偏移：0x4C
复位值：0x0000

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

位7	EF：USB模块使能位 此位在需要使能USB模块时由应用程序置位。如果此位为0，USB模块将停止工作，忽略所有寄存器的设置，不响应任何USB通信。
位6:0	ADD[6:0]：设备地址 此位记录了USB主机在枚举过程中为USB设备分配的地址值。该地址值和端点地址（EA）必需和USB令牌分组中的地址信息匹配，才能在指定的端点进行正确的USB传输。

USB分组缓冲区描述表地址寄存器

地址偏移：0x50
复位值：0x0000

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

位15:3	BTABLE[15:3]：缓冲表 此位记录分组缓冲区描述表的起始地址。分组缓冲区描述表用来指示每个端点的分组缓冲区地址和大小，按8字节对齐（即最低3位为000）。每次传输开始时，USB模块读取相应端点所对应的分组缓冲区描述表获得缓冲区地址和大小信息。
位2:0	保留位，由硬件置为0

17.6.2 端点寄存器

端点寄存器的数量由 USB 模块所支持的端点数目决定。USB 模块最多支持 8 个双向端点。每个 USB 设备必须支持一个控制端点，控制端点的地址（EA 位）必需为 0。不同的端点必需使用不同的端点号，否则端点的状态不定。每个端点都有与之对应的 USB_EpnR 寄存器，用于存储该端点的各种状态信息。

USB 端点n寄存器(USB_EPnR), n=[0..7]

地址偏移：0x00 到 0x1C
复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CTR_RX	DTOG_RX	STAT_RX[1:0]		SETUP	EP TYPE[1:0]		EP_KIND	CTR_TX	DTOG_TX	STAT_TX[1:0]		EA[3:0]			
r-c	t	t	t	t	rw	rw	rw	r-c	t	t	t	rw	rw	rw	rw

当 USB 模块收到 USB 总线复位信号，或 CTLR 寄存器的 FRES 位置位时，USB 模块将会复位。该寄存器除了 CTR_RX 和 CTR_TX 位保持不变以处理紧随的 USB 传输外，其他位都被复位。每个端点对应一个 USB_EPnR 寄存器，其中 n 为端点地址，即端点 ID 号。

对于此类寄存器应避免执行读出－修改－写入操作，因为在读和写操作之间，硬件可能会设置某些位，而这些位又会在写入时被修改，导致应用程序错过相应的操作。因此，这些位都有一个写入无效的值，建议用 Load 指令修改这些寄存器，以免应用程序修改了不需要修改的位。

位15	CTR_RX：正确接收标志位 此位在正确接收到OUT或SETUP分组时由硬件置位，应用程序只能对此位清零。如果CTRM位已置位，相应的中断会产生。收到的是OUT分组还是SETUP分组可以通过下面描述的SETUP位确定。以NAK或STALL结束的分组和出错的传输不会导致此位置位，因为没有真正传输数据。 此位应用程序可读可写，但只有写0有效，写1无效。
位14	DTOG_RX：用于数据接收的数据翻转位 对于非同步端点，此位由硬件设置，用于标记希望接收的下一个数据分组的Toggle位（0=DATA0，1=DATA1）。在接收到PID（分组ID）正确的数据分组之后，USB模块发送ACK握手分组，并翻转此位。对于控制端点，硬件在收到SETUP分组后清除此位。 对于双缓冲端点，此位还用于支持双缓冲区的交换(请参考17.5.3双缓冲端点)。 对于同步端点，由于仅发送DATA0，因此此位仅用于支持双缓冲区的交换(请参考17.5.4同步传输)而不需进行翻转。同步传输不需要握手分组，因此硬件在收到数据分组后立即设置此位。 应用程序可以对此位进行初始化(对于非控制端点，初始化是必需的)，或者翻转此位用于特殊用途。 此位应用程序可读可写，但写0无效，写1可以翻转此位。
位13:12	STAT_RX[1:0] 用于数据接收的状态位 此位用于指示端点当前的状态，表58列出了端点的所有状态。当一次正确的OUT或SETUP数据传输完成后(CTR_RX=1)，硬件会自动设置此位为NAK状态，使应用程序有足够的时间在处理完当前传输的数据后，响应下一个数据分组。 对于双缓冲批量端点，由于使用特殊的传输流量控制策略，因此根据使用的缓冲区状态控制传输状态(请参考17.5.3双缓冲端点)。 对于同步端点，由于端点状态只能是有效或禁用，因此硬件不会在正确的传输之后设置此位。如果应用程序将此位设为STALL或者NAK，USB模块响应的操作是未定义的。 此位应用程序可读可写，但写0无效，写1翻转此位。

位11	SETUP：SETUP分组传输完成标志位 此位在USB模块收到一个正确的SETUP分组后由硬件置位，只有控制端点才使用此位。在接收完成后(CTR_RX=1)，应用程序需要检测此位以判断完成的传输是否是SETUP分组。为了防止中断服务程序在处理SETUP分组时下一个令牌分组修改了此位，只有CTR_RX为0时，此位才可以被修改，CTR_RX为1时不能修改。 此位应用程序只读。
位10:9	EP_TPYE[1:0] 端点类型位 此位用于指示端点当前的类型，所有的端点类型都在表59中列出。所有的USB设备都必需包含一个地址为0的控制端点，如果需要可以有其他地址的控制端点。只有控制端点才会有SETUP传输，其他类型的端点无视此类传输。SETUP传输不能以NAK或STALL分组响应，如果控制端点在收到SETUP分组时处于NAK状态，USB模块将不响应分组，就会出现接收错误。如果控制端点处于STALL状态，SETUP分组会被正确接收，数据会被正确传输，并产生一个正确传输完成的中断。控制端点的OUT分组安装普通端点的方式处理。 批量端点和中断端点的处理方式非常类似，仅在对EP_KIND位的处理上有差别。 同步端点的用法请参考17.5.4同步传输。
位8	EP_KIND：端点特殊类型位 此位的需要和EP_TYPE位配合使用，具体的定义请参考表60。 DBL_BUF：应用程序设置此位能使能批量端点的双缓冲功能。具体请参考17.5.3双缓冲端点。 STATUS_OUT：应用程序设置此位表示USB设备期望主机发送一个状态数据分组，此时，设备对于任何长度不为0的数据分组都响应STALL分组。此功能仅用于控制端点，有利于提供应用程序对于协议层错误的检测。如果STATUS_OUT位被清除，OUT分组可以包含任意长度的数据。
位7	CTR_TX：正确发送标志位 此位由硬件在一个正确的IN分组传输完成后置位。如果CTRM位已被置位，会产生相应的中断。应用程序需要在处理完该事件后清除此位。在IN分组结束时，如果主机响应NAK或STALL则此位不会被置位，因为数据传输没有成功。 此位应用程序可读可写，但写0有效，写1无效。
位6	DTOG_RX：发送数据翻转位 对于非同步端点，此位用于指示下一个要传输的数据分组的Toggle位(0＝DATA0，1=DATA1)。在一个成功传输的数据分组后，如果USB模块接收到主机发送的ACK分组，就会翻转此位。对于控制端点，USB模块会在收到正确的SETUP PID后置位此位。 对于双缓冲端点，此位还可用于支持分组缓冲区交换(请参考17.5.3双缓冲端点)。 对于同步端点，由于只传送DATA0，因此该位只用于支持分组缓冲区交换(请参考17.5.4同步传输)。由于同步传输不需要握手分组，因此硬件在接收到数据分组后即设置该位。 应用程序可以初始化该位（对于非控制端点，初始化此位时必需的），也可以设置该位用于特殊用途。 此位应用程序可读可写，但写0无效，写1翻转此位。
位5:4	STAT_TX[1:0]：用于发送数据的状态位 此位用于标识端点的当前状态，表61列出了所有的状态。应用程序可以翻转这些位来初始化状态信息。在正确完成一次IN分组的传输后(CTR_TX=1)，硬件会自动设置此位为NAK状态，保证应用程序有足够的时间准备好数据响应后续的数据传输。 对于双缓冲批量端点，由于使用特殊的传输流量控制策略，是根据缓冲区的状态控制传输的状态的(请参考17.5.3双缓冲端点)。 对于同步端点，由于端点的状态只能是有效或禁用，因此硬件不会在数据传输结束时改变端点的状态。如果应用程序将此位设为STALL或者NAK，则USB模块后续的操作是未定义的。 此位应用程序可读可写，但写0无效，写1翻转此位。
位3:0	EA[3:0] 端点地址 应用程序必需设置此4位，在使能一个端点前为它定义一个地址。

表58 接受状态编码

STAT_RX[1:0]	描述
00	DISABLED：端点忽略所有的接收请求。
01	STALL：端点以STALL分组响应所有的接收请求。
10	NAK：端点以NAK分组响应所有的接收请求。
11	VALID：端点可用于接收。

表59 端点类型编码

EP_TYPE[1:0]	描述
00	BULK：批量端点
01	CONTROL：控制端点
10	ISO：同步端点
11	INTERRUPT：中断端点

表60 端点特殊类型定义

EP_TYPE[1:0]		EP_KIND意义
00	BULK	DBL_BUF：双缓冲端点
01	CONTROL	STATUS_OUT
10	ISO	未使用
11	INTERRUPT	未使用

表61 发送状态编码

STAT_RX[1:0]	描述
00	DISABLED：端点忽略所有的发送请求。
01	STALL：端点以STALL分组响应所有的发送请求。
10	NAK：端点以NAK分组响应所有的发送请求。
11	VALID：端点可用于发送。

17.6.3 缓冲区描述表

虽然缓冲区描述表位于分组缓冲区内，但还是可以将它看作是特殊的寄存器，用以配置 USB 模块和微控制器内核共享的分组缓冲区的地址和大小。由于 APB1 总线按 32 位寻址，所以所有的分组缓冲区地址都使用 32 位对齐的地址，而不是 USB_BTABLE 寄存器和缓冲区描述表所使用的地址。

后面的部分介绍了两种地址表示方式：一种是应用程序访问分组缓冲区时使用的，另一种是相对于 USB 模块的本地地址。供应用程序使用的分组缓冲区地址需要乘以 2 才能得到缓冲区在微控制器中的真正地址。分组缓冲区的首地址为 0x4000 6000。下面将描述与 USB_EPnR 寄存器相关的缓冲区描述表。完整的分组缓冲区的说明和缓冲区描述表的用法请参考分组缓冲区的结构和用途章节。

发送缓冲区地址寄存器 n()

地址偏移：[USB_BTABLE] + n×16
USB 本地地址：[USB_BTABLE] + n×8

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

位15:1	ADDRn_TX[15:1]：发送缓冲区地址 此位记录了收到下一个IN分组时，需要发送的数据所在的缓冲区起始地址。
位0	因为分组缓冲区的地址按字对齐，所以此位必需为0。

发送数据字节数寄存器 n()

地址偏移：[USB_BTABLE] + n×16 + 4
USB 本地地址：[USB_BTABLE] + n×8 + 2

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

位15:10	由于USB模块支持的最大数据分组为1023个字节，所以USB模块忽略这些位。
位9:0	COUNTn_TX[9:0]： 发送数据字节数 此位记录了收到下一个IN分组时要传输的数据字节数。

注： 双缓冲区和同步 IN 端点有两个 USB_COUNTn_TX 寄存器：分别为 USB_COUNTn_TX_1 和 USB_COUNTn_TX_0，内容如下：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
-						COUNTn_TX_1[9:0]									
						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
-						COUNTn_TX_0[9:0]									
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

接收缓冲区地址寄存器 n()

地址偏移：[USB_BTABLE] + n×16 + 8
USB 本地地址：[USB_BTABLE] + n×8 + 4

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

位15:1	ADDRn_RX[15:1]：接收缓冲区地址 此位记录了收到下一个OUT或者SETUP分组时，用于保存数据的缓冲区起始地址。
位0	因为分组缓冲区的地址按字对齐，所以此位必需为0。

接收数据字节数寄存器 n()

地址偏移：[USB_BTABLE] + n×16 + 12
USB 本地地址：[USB_BTABLE] + n×8 + 6

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

该寄存器用于存放接收分组时需要使用到的两个参数。高 6 位定义了接收分组缓冲区的大小，以便 USB 模块检测缓冲区的溢出。低 10 位则用于 USB 模块记录实际接收到的字节数。由于有效位数的限制，缓冲区的大小由分配到的存储区块数表示，而存储区块的大小则由所需的缓冲区大小决定。缓冲区的大小在设备枚举过程中定义，由端点描述符的参数 maxPacketSize 表述。(具体信息请参考 USB 2.0 协议规范")

位15	BL_SIZE：存储区块的大小 此位用于定义决定缓冲区大小的存储区块的大小。 如果BL_SIZE=0，存储区块的大小为2字节，因此能分配的分组缓冲区的大小范围为2－62个字节。 如果BL_SIZE=1，存储区块的大小为32字节，因此能分配的分组缓冲区的大小范围为32－512字节，符合USB协议定义的最大分组长度限制。
位14:10	NUM_BLOCK[4:0]：存储区块的数目 此位用以记录分配的存储区块的数目，从而决定最终使用的分组缓冲区的大小。具体请参考表62
位9:0	COUNTn_RX[9:0]：接收到的字节数 此位由USB模块写入，用以记录端点收到的最新的OUT或SETUP分组的实际字节数。

注： 双缓冲区和同步 IN 端点有两个 USB_COUNTn_RX 寄存器：分别为 USB_COUNTn_RX_1 和 USB_COUNTn_TX_0，内容如下：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BLSIZE_1	NUM_BLOCK_1[4:0]					COUNTn_RX_1[9:0]									
rw	rw	rw	rw	rw	rw	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
BLSIZE_0	NUM_BLOCK_0[4:0]					COUNTn_RX_0[9:0]									
rw	rw	rw	rw	rw	rw	r	r	r	r	r	r	r	r	r	r

表62 分组缓冲区大小的定义

NUM_BLOCK[4:0]的值	BL_SIZE=0时的 分组缓冲区大小	当BL_SIZE=1时的 分组缓冲区大小
00000	不允许使用	32字节
00001	2字节	64字节
00010	4字节	96字节
00011	6字节	128字节
…	…	…
01111	30字节	512字节
10000	32字节	保留
10001	34字节	保留
10010	36字节	保留
…	…	…
11110	60字节	保留
11111	62字节	保留

17.7 USB寄存器映像

表63 USB 寄存器映像和复位值

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
00h	USB_EP0R	保留																CTR_RX	DTOG_RX	STAT_RX [1:0]		SETUP	EPTYPE [1:0]		EP_KIND	CTR_TX	DTOG_TX	STAT_TX [1:0]		EA[3:0]			
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
04h	USB_EP1R	保留																CTR_RX	DTOG_RX	STAT_RX [1:0]		SETUP	EPTYPE [1:0]		EP_KIND	CTR_TX	DTOG_TX	STAT_TX [1:0]		EA[3:0]			
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
08h	USB_EP2R	保留																CTR_RX	DTOG_RX	STAT_RX [1:0]		SETUP	EPTYPE [1:0]		EP_KIND	CTR_TX	DTOG_TX	STAT_TX [1:0]		EA[3:0]			
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0Ch	USB_EP3R	保留																CTR_RX	DTOG_RX	STAT_RX [1:0]		SETUP	EPTYPE [1:0]		EP_KIND	CTR_TX	DTOG_TX	STAT_TX [1:0]		EA[3:0]			
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10h	USB_EP4R	保留																CTR_RX	DTOG_RX	STAT_RX [1:0]		SETUP	EPTYPE [1:0]		EP_KIND	CTR_TX	DTOG_TX	STAT_TX [1:0]		EA[3:0]			
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
14h	USB_EP5R	保留																CTR_RX	DTOG_RX	STAT_RX [1:0]		SETUP	EPTYPE [1:0]		EP_KIND	CTR_TX	DTOG_TX	STAT_TX [1:0]		EA[3:0]			
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
18h	USB_EP6R	保留																CTR_RX	DTOG_RX	STAT_RX [1:0]		SETUP	EPTYPE [1:0]		EP_KIND	CTR_TX	DTOG_TX	STAT_TX [1:0]		EA[3:0]			
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1Ch	USB_EP7R	保留																CTR_RX	DTOG_RX	STAT_RX [1:0]		SETUP	EPTYPE [1:0]		EP_KIND	CTR_TX	DTOG_TX	STAT_TX [1:0]		EA[3:0]			
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
20h~ 3Fh	保留																																

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
40h	USB Control Register	保留																CTRM	PMAOVRM	ERRM	WKUPM	SUSPM	RESETM	SOFM	ESOFM	保留			RESUME	FSUSP	LPMODE	PDWN	FRES
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
44h	USB Interrupt Status Register	保留																CTR	PMAOVR	ERR	WKUP	SUSP	RESET	SOF	ESOF	保留			DIR	EP_ID [3:0]			
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
48h	USB frame number register	保留																RXDP	RXDM	LCK	LSOF [1:0]		FN[10:0]										
	复位值																	0	0	0	0	0	X	X	X	X	X	X	X	X	X	X	X
4Ch	USB device address	保留																保留								EF	ADD[6:0]						
	复位值																									0	0	0	0	0	0	0	0
50h	Buffer table address	保留																BTABLE[15:3]														保留	
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0		

关于寄存器的起始地址，请参见第 1 章。

18 串行外设接口(SPI)

18.1 简介

串行外设接口(SPI)允许芯片与外部设备以半/全双工、同步、串行方式通信。此接口可以被配置成主模式，这种工作模式下，它要为外部从设备提供通信时钟(SCK)。接口还能以多主配置方式工作。

它可用于多种用途，包括可选第三根双向数据线的双线单工同步传输，或使用 CRC 校验的可靠通信。

18.2 主要特征

- 3 线全双工同步传输
- 带或不带第三根双向数据线的双线单工同步传输
- 8 或 16 位传输帧格式选择
- 主或从操作
- 支持多主模式
- 8 个主模式波特率预分频系数(最大为 $f_{PCLK}/2$)
- 从模式频率 (最大为 $f_{PCLK}/2$)
- 主模式和从模式的快速通信：最大 SPI 速度达到 18MHz
- 主模式和从模式下均可以由软件或硬件进行 NSS 管理：主/从操作模式的动态改变
- 可编程的时钟极性和相位
- 可编程的数据顺序，MSB 在前或 LSB 在前
- 可触发中断的专用发送和接收标志
- SPI 总线忙状态标志
- 支持可靠通信的硬件 CRC
  - 在发送模式下，CRC 值可以被作为最后一个字节发送
  - 在全双工模式中对接收到的最后一个字节自动进行 CRC 校验
- 可触发中断的主模式故障、过载以及 CRC 错误标志
- 支持 DMA 功能的 1 字节发送和接收缓冲器：产生发送和接受请求

18.3 功能描述

18.3.1 概述

SPI的方框图见 图 161。通常，SPI通过 4 个引脚和外部设备相连。

- MISO：主入/从出数据口。此脚可以被用来在从模式中发送数据，在主模式中接收数据。
- MOSI：主出/从入数据口。此脚可以用来在主模式时发送数据，在从模式时接收数据。
- SCK：SPI 主设备输出串行时钟，SPI 从设备输入串行时钟
- NSS：从选择。这是一个用来选择主/从模式的可选引脚。SPI 主设备和从设备分别通信时，该引脚起到依次片选各个从设备的作用，以避免发生数据线冲突。从设备的 NSS 输入可以由主设备上的标准 I/O 端口驱动。SPI 工作在主设备配置时，如果 SSOE 位使能，则 NSS 引脚用作输出，并输出低电平；此时，所有 NSS 引脚连到该设备 NSS 引脚的其他设备都将收到低电平，当这些设备配置为 NSS 硬件模式时，就被自动地配置成了从设备。

图161 SPI 框图

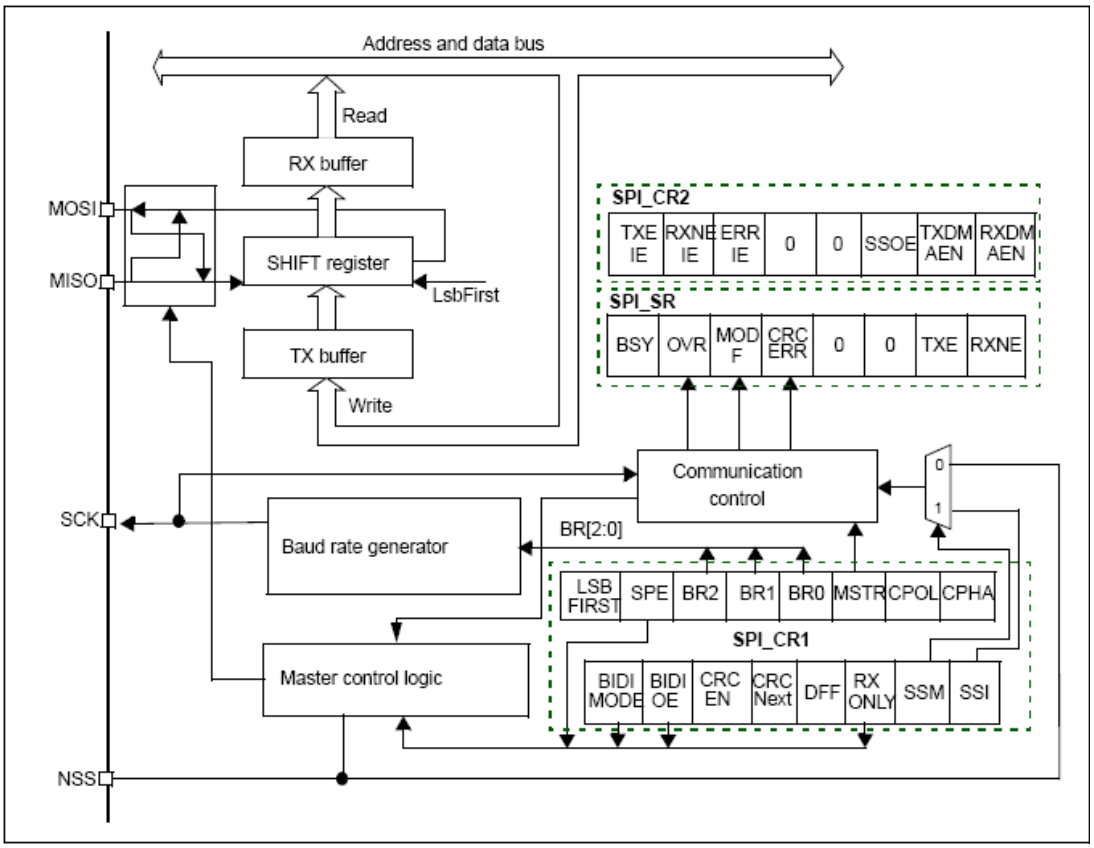
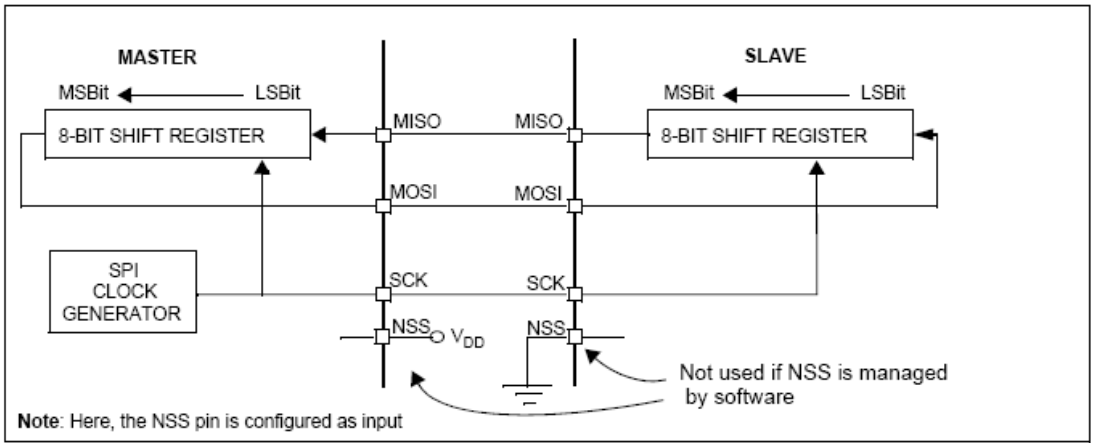


图 162 示出了一个基本的单个主设备和单个从设备相互连接的例子。

图162 单主和单从应用



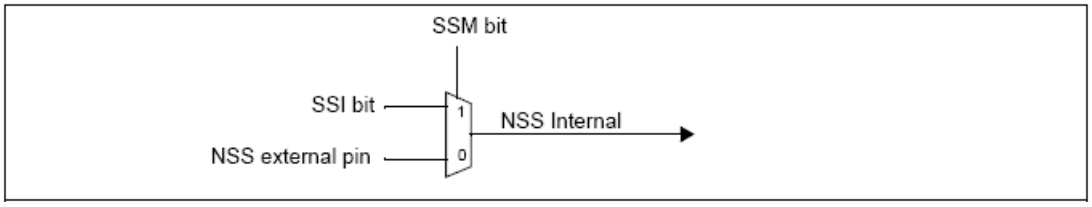
MOSI 脚相互连接，MISO 脚相互连接。用这种方式，数据在主和从之间串行地传输(MSB 位在前)。
通信总是由主设备发起。主设备通过 MOSI 脚把数据发送给从设备，从设备通过 MISO 引脚回传数据。这意味全双工通信的数据输出和数据输入是用同一个时钟信号同步的；时钟信号由主设备通过 SCK 脚提供。

从选择(NSS)脚管理

NSS 脚可以用做输入（硬件模式）和输出。SS 的输出可由 SPI_CR2 寄存器中的 SSOE 位使能或禁止。多主配置只有在 SS 输出被禁止时才有可能。当 NSS 脚被用作输出（SSOE 位）并且 SPI 是主模式配置时，NSS 脚被拉为低。因此当其他 SPI 设备被配置成 NSS 模式时，这些设备的所有 NSS 引脚与之相连，就都变成从设备了。

利用NSS脚来控制从设备选择信号(NSS脚)的另一个办法是：由软件来管理从选择信号，通过配置SPI_CR1 寄存器的SSM位实现(见 图 163)。在软件管理中，外部NSS脚可由应用程序作其他使用，内部NSS信号电平通过SPI_CR1 寄存器的SSI位来驱动。

图163 硬件/软件的从选择管理



时钟信号的相位和极性

使用 SPI_CR 寄存器的 CPOL 和 CPHA 位，组合成四种可能的时序关系。
CPOL(时钟极性)位控制在没有数据传输时时钟的空闲状态电平，此位对主模式和从模式下的设备都有效。如果 CPOL 被复位，SCK 引脚在空闲状态保持低电平；如果 CPOL 被置位，SCK 引脚在空闲状态保持高电平。

如果 CPHA(时钟相位)位被置位，SCK 时钟的第二个边沿(CPOL 位为 0 时就是下降沿，CPOL 位为 1 时就是上升沿)进行数据位的采样。数据在第一个时钟边沿被

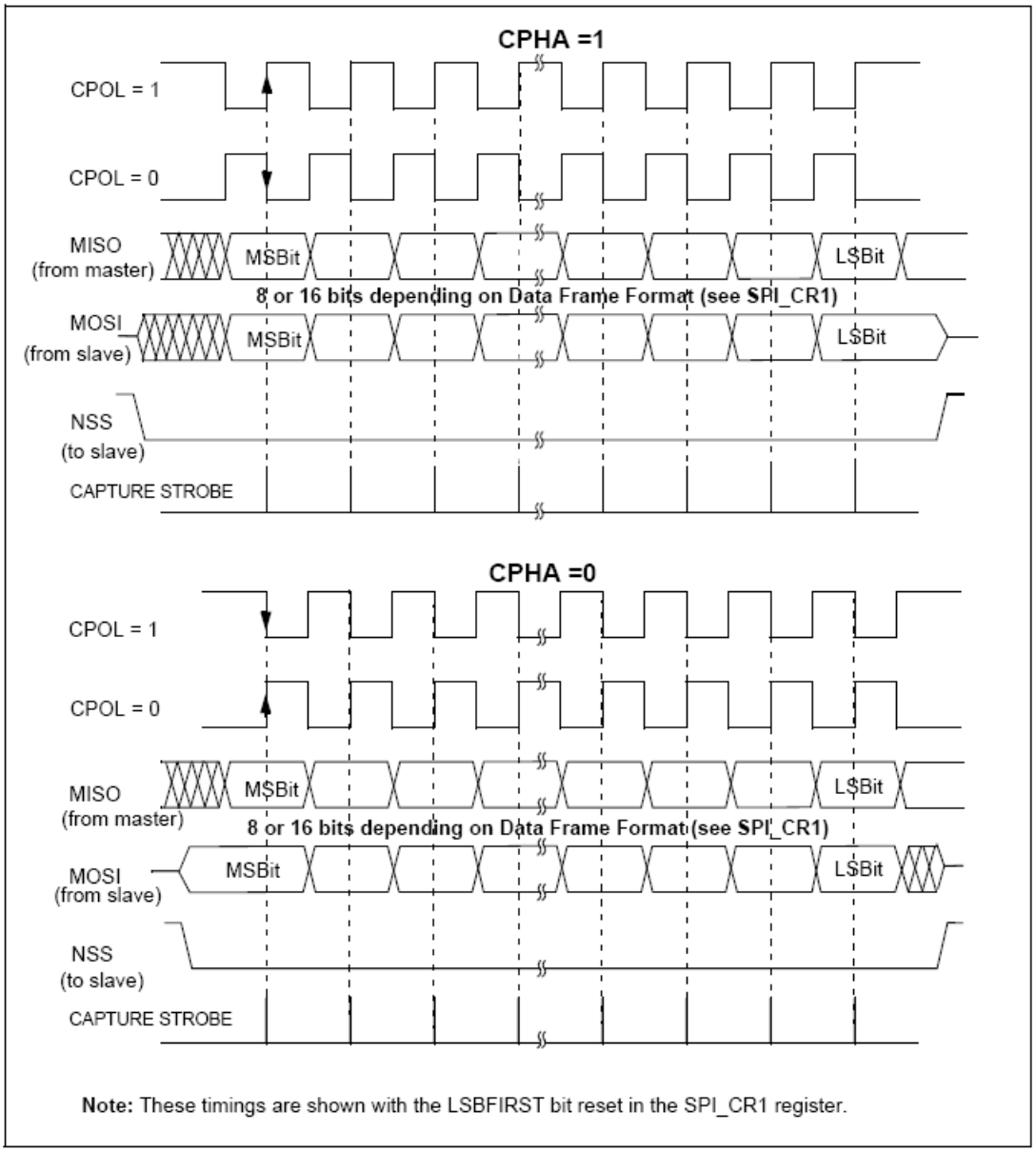
锁存。如果 CPHA 位被复位，SCK 时钟的第一边沿(CPOL 位为 0 时就是下降沿，CPOL 位为 1 时就是上升沿)进行数据位采样。数据在第二个时钟边沿被锁存。

CPOL 时钟极性和 CPHA 时钟相位的组合选择数据捕捉的时钟边沿。

图 164 显示了SPI传输的 4 种CPHA和CPOL位组合。此图可以解释为主设备和从设备的SCK脚、MISO脚、MOSI脚直接连接的主或从时序图。

注意：1. 在改变 CPOL/CPHA 位之前，必须清除 SPE 位将 SPI 禁止。
2. 主和从必须配置成相同的时序模式。
3. SCK 的空闲状态必须和 SPI_CR1 寄存器指定的极性一致（CPOL 为 1 时，空闲时应上拉 SCK 为高电平；CPOL 为 0 时，空闲时应下拉 SCK 为低电平）。
4. 数据帧格式(8 位或 16 位)由 SPI_CR1 寄存器的 DFF 位选择，并且决定发送/接收的数据长度。

图164 数据时钟时序图



Note: These timings are shown with the LSBFIRST bit reset in the SPI_CR1 register.

数据帧格式

根据 SPI_CR1 寄存器中的 LSBFIRST 位，输出数据位时可以 MSB 在先也可以 LSB 在先。

根据 SPI_CR1 寄存器的 DFF 位，每个数据帧可以是 8 位或是 16 位。所选择的数据帧格式对发送和/或接收都有效。

18.3.2 SPI从模式

在从配置里，SCK 引脚用于接收到从主设备来的串行时钟。SPI_CR1 寄存器中 BR[2:0]的设置不影响数据传输速率。

配置步骤

1. 设置DFF位以定义数据帧格式为8位或16位
2. 选择CPOL和CPHA位来定义数据传输和串行时钟之间的相位关系(见图164)。为保证正确的数据传输，从设备和主设备的CPOL和CPHA位必须配置成相同的方式。
3. 帧格式(MSB在前还是LSB在前取决于SPI_CR1寄存器中的LSBFIRST位)必须和主设备相同。
4. 硬件模式下(参考从选择(NSS)脚管理部分)，在完整的数据帧(8位或16位)发送过程中，NSS引脚必须为低电平。软件模式下，设置SPI_CR1寄存器中的SSM位并清除SSI位。
5. 清除MSTR位，设置SPE位，使相应引脚工作于SPI模式下。

在这个配置里，MOSI 引脚是数据输入，MISO 引脚是数据输出。

数据发送过程

数据字被并行写入发送缓冲器。

当从设备收到时钟信号，并且在 MOSI 引脚上出现第一个数据位时，发送过程开始，第一个位被发送出去。余下的位(对于 8 位数据帧格式，还有 7 位；对于 16 位数据帧格式，还有 15 位)被装进移位寄存器。当发送缓冲器中的数据传输到移位寄存器时，SPI_SP 寄存器里的 TXE 标志被设置。如果设置了 API_CR2 寄存器上的 TXEIE 位，将会产生中断。

对于接收方，当数据接收完成时：

- 移位寄存器中的数据传送到接收缓冲器，SPI_SR 寄存器中的 RXNE 标志被设置。
- 如果设置了 SPI_CR2 寄存器中的 RXEIE 位，则产生中断。

在最后一个采样时钟边沿后，RXNE 位被设置，移位寄存器中接收到的数据字节被传送到接收缓冲器。当读 SPI_DR 寄存器时，SPI 设备返回这个值，并且清除 RXNE 位。

18.3.3 SPI主模式

在主配置时，串行时钟在 SCK 脚产生。

配置步骤

1. 通过SPI_CR1寄存器的BR[2:0]位定义串行时钟波特率。
2. 选择CPOL和CPHA位，定义数据传输和串行时钟间的相位关系(见图164)。
3. 设置DFF位来定义8或16位数据帧格式。
4. 配置SPI_CR1寄存器的LSBFIRST位定义帧格式。
5. 如果NSS引脚需要工作在输入模式，硬件模式中在整个数据帧传输期间应把NSS脚连接到高电平；在软件模式中，需设置SPI_CR1寄存器的SSM和SSI位。如果NSS引脚工作在输出模式，则只需设置SSOE位。
6. 必须设置MSTR和SPE位(只当NSS脚被连到高电平，这些位才能保持置位)。

在这个配置中，MOSI 脚是数据输出，而 MISO 脚是数据输入。

数据发送过程

当一字节写进发送缓冲器时，发送过程开始。

在发送第一个数据位时，数据字被并行地(通过内部总线)传入移位寄存器，而后串行地移出到 MOSI 脚上；MSB 在先还是 LSB 在先，取决于 SPI_CR1 寄存器中的 LSBFIRST 位。数据从发送缓冲器传输到移位寄存器时 TXE 标志将被置位，如果设置 SPI_CR1 寄存器中的 TXEIE 位，将产生中断。

对于接收器来说，当数据传输完成时：

- 移位寄存器里的数据传送到接收缓冲器，并且 RXNE 标志被置位。
- 如果 SPI_CR2 寄存器中的 RXEIE 位被设置，则产生中断。

在最后采样时钟沿，RXNE 位被设置，在移位寄存器中接收到的数据字被传送到接收缓冲器。读 SPI_DR 寄存器时，SPI 设备返回接收到的数据字。读 SPI_DR 寄存器将清除 RXNE 位。

一旦传输开始，如果下一个将发送的数据被放进了发送缓冲器，就可以维持一个连续的传输流。在试图写发送缓冲器之前，需确认 TXE 标志应该是 1。

18.3.4 单向通信

SPI 能够以两种配置工作于单工方式：

- 1 条时钟线和 1 条双向数据线
- 1 条时钟线和 1 条数据线（只读方式）

1 条时钟线和 1 条双向数据线

设置 SPI_CR1 寄存器中的 BIDIMODE 位而启用此模式。在这个模式中，SCK 用作时钟，主模式中的 MOSI 或从模式中的 MISO 用作数据通信。传输的方向由 SPI_CR2 寄存器里的 BIDIOE 控制，当这个位是 1 的时候，数据线是输出，否则是输入。

1 条时钟和 1 条数据线（只读方式）

为了释放一根 I/O 脚作为他用，可以通过设置 SPI_CR1 寄存器中的 RXONLY 位来禁止 SPI 输出功能。这样的话，SPI 将运行于只接收模式。

只接收模式

为启动只接受模式通信，必须首先激活 SPI。在主模式中，一旦使能 SPI，通信立即启动，当 SPE 位复位时通信即停止；在从模式中，只要 NSS 被拉低(或 SSI 位为 0)以及 SCK 持续送到从设备，SPI 就一直在接收。

注意：　当 SPI_CR1 寄存器中的 RXONLY 位被复位时，SPI 可以工作于只发送模式。接收脚(主设备的 MISO，或者从设备的 MOSI)可以被用作通用 IO 口。因此读数据寄存器时，读到的不是接收的值。

18.3.5 状态标志

应用程序通过 3 个状态标志可以完全监控 SPI 总线的状态。

忙(Busy)标志

此标志表明 SPI 通信层的状态。当它被设置时，表明 SPI 正忙于通信，并且/或者在发送缓冲器里有一个有效的数据字正在等待被发送。此标志的目的是说明在 SPI 总线上是否有正在进行的通信。以下情况时此标志将被置位：

1. 数据被写进主设备的SPI_DR寄存器上。
2. SCK时钟出现在从设备的时钟引脚上。

发送/接收一个字(字节)完成后，BUSY 标志立即清除；此标志由硬件设置和清除。监视此标志可以避免写冲突错误。写此标志无效。仅当 SPE 位被设置时此标志才有意义。

发送缓冲器空闲标志(TXE)

此标志被置位时表明发送缓冲器为空，因此下一个待发送的数据可以写进缓冲器里。当发送缓冲器有一个待发送的数据时，TXE 标志被清除。当 SPI 被禁止时，此标志被清除。

接收缓冲器非空(RXNE)

此标志被置位时表明在接收缓冲器中有一个有效的接收数据。读 SPI 数据寄存器就可以清除此标志。

18.3.6 CRC计算

CRC 校验仅用于保证全双工通信的可靠性。数据发送和数据接收分别使用单独的 CRC 计算器。通过对每一个接收位进行可编程的多项式运算来计算 CRC。CRC 的计算是在由 SPI_CR1 寄存器中 CPHA 和 CPOL 位定义的采样时钟边沿进行的。

注意： 该 SPI 接口提供了两种 CRC 计算方法，取决于所选的发送和/或接收的数据帧格式：8 位数据帧采用 CR8；16 位数据帧采样 CRC16-CCITT。

CRC 计算是通过设置 SPI_CR1 寄存器中的 CRCEN 位启用的。设置 CRCEN 位时同时复位 CRC 寄存器(SPI_RXCRCR 和 SPI_TXCRCR)。当设置了 SPI_CR1 的 CRCNEXT 位，SPI_TXCRCR 的内容将在当前字节发送之后发出。

注意： 在传输 SPI_TXCRCR 的内容时，如果在移位寄存器中收到的数值与 SPI_RXCRCR 的内容不匹配，则 SPI_SR 寄存器的 CRCERR 标志位被置 1。如果在 TX 缓冲器中还有数据，CRC 的数值仅在数据字节传输结束后传送。在传输 CRC 期间，CRC 计算器关闭，寄存器的数值保持不变。

注意： 请参考产品说明书，以确认有此功能(不是所有型号都有此功能)。

SPI 通信可以通过以下步骤使用 CRC：

- 将计算 CRC 时用到的多项式写到 SPI_CRCPOLYR 寄存器中。
- 通过设置 SPI_CR1 寄存器中的 CRCEN 位启用 CRC 计算。此动作同时清除 SPI_RXCRCR 和 SPI_TXCRCR 寄存器。
- 设置 CPOL、CPHA、LSBfirst、DFF、BR、SSM、SSI 和 MSTR 的值。
- 设置 SPI_CR1 寄存器的 SPE 位启动 SPI 功能。
- 启动通信并且维持通信，直到只剩最后一个字节或者半字。
- 当把最后一个字节或半字写进发送缓冲器，设置 SPI_CR1 的 CRCNext 位，指示硬件在最后一个数据字节发送完成后，发送 CRC。在发送 CRC 期间，CRC 计算停止。

- 当最后一个字节或半字被发送后，SPI 发送 CRC，CRCNext 位被清除。同样，接收到的 CRC 和 SPI_RXCRCR 值进行比较，如果比较不相配，SPI_SR 上的 CRCERR 标志被置位，当设置了 SPI_CR2 寄存器的 ERRIE 时，则产生中断。

注意：　当 SPI 时钟频率较高时，用户在发送 CRC 时必须小心。因为在 CRC 传输期间，使用 CPU 的时间应尽可能少。在发送 CRC 过程中，应禁止函数调用，这是为了避免在最后的数据出错和接收 CRC 时出错。这个限制只适用于全双工模式；因为在单工模式里，CRC 的传输是由软件完成，不是通过设置 CRCNEXT 位来自动完成的。

当 SPI 时钟频率较高时，建议采用 DMA 模式以避免 SPI 速度性能的降低。

18.3.7 利用DMA的SPI通信

为了达到最大通信速度，需要及时往 SPI 发送缓冲区填数据，同样接收缓冲器中的数据也必须及时读走以防止溢出。为了方便高速率的数据传输，SPI 实现了一种采用简单的请求/应答的 DMA 机制。当 SPI_CR2 寄存器上的对应使能位被设置时，发出 DMA 传输请求。发送缓冲器和接收缓冲器亦有各自的 DMA 请求。

注意：　当 SPI 时钟频率较高时，建议采用 DMA 模式以避免 SPI 速度性能的降低。

带CRC的DMA功能

当 SPI 工作在全双工模式并使用 CRC 检验以及启用 DMA 模式时，通信结束时，CRC 字节的发送和接收是自动完成的。

数据和 CRC 传输结束时，SPI_SR 寄存器的 CRCERR 标志被置位表示在传输期间发生错误。

18.3.8 错误标志

主模式错误(MODF)

主模式故障仅发生在：在片选引脚硬件模式管理下，主设备的 NSS 脚被拉低；或者在片选引脚软件模式管理下，SSI 位被复位时；MODF 位被自动置位。主模式故障对 SPI 设备有以下影响：

- MODF 位被置位，如果设置了 ERRIE 位，则产生 SPI 中断。
- SPE 位被复位。这将停止一切输出，并且关闭 SPI 接口。
- MSTR 位被复位，因此强迫此设备进入从模式。

下面的步骤用于清除 MODF 位：

1. 当MODF位被置位时，执行一次对SPI_SR寄存器的读或写操作
2. 然后写SPI_CR1寄存器

在有多个 MCU 的系统中，为了避免出现多个从设备的冲突，必须先拉高该主设备的 NSS 脚，再对 MODF 位进行清零。在清零的过程中或者清零完成之后，SPE 和 MSTR 位可以恢复到它们的原始状态。

出于安全的考虑，当 MODF 位被置位的情况下，硬件不允许设置 SPE 和 MSTR 位。

通常配置下，从设备的 MODF 位不能被置位。然而，在多主配置里，一个设备可以在设置了 MODF 位的情况下，处于从设备模式；此时，MODF 位指示可能出现了多主冲突。中断程序可以执行一个复位或返回到默认状态来从错误状态中恢复。

溢出错误

当主设备已经发送了数据字节，而从设备还没有清除前一个数据字节产生的 RXNE 时，即为溢出错误。当产生溢出错误时：

- OVR位被设置；当设置了ERRIE位时，则产生中断。

此时，接收器缓冲器的数据不是主设备发送的新数据，读 SPI_DR 寄存器返回的是之前未读的字节，所有随后传送的字节都被丢弃。

依次读出 SPI_DR 寄存器和 SPI_SR 寄存器可将 OVR 清除。

CRC 错误

当设置了 SPI_CR 寄存器上的 CRCEN 位时，CRC 错误标志用来核对接收数据的有效性。在全双工模式下，如果移位寄存器中接收到的值(发送方发送的 SPI_TXCRCR)和接收方 SPI_RXCRCR 寄存器中的值不匹配，SPI_SR 寄存器上的 CRCERR 标志被置位。

18.3.9 中断

表64 SPI 中断请求

中断事件	事件标志	使能控制位
发送缓冲器空标志	TXE	TXEIE
接收缓冲器非空标志	RXNE	RXNEIE
主模式错误事件	MODF	ERRIE
溢出错误	OVR	

CRC错误标志	CRCERR	

18.4 SPI寄存器描述

关于寄存器描述中所用到的缩略词可参见第 1.1 节。

18.4.1 SPI控制寄存器 1(SPI_CR1)

地址偏移：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	CRCEN	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: 双向数据模式使能 0：选择“双线双向”模式 1：选择“单线双向”模式
位14	BIDIOE: 双向模式下的输出使能 和BIDIMODE位一起决定在“单线双向”模式下数据的输出方向 0：输出禁止（只收模式） 1：输出使能（只发模式） 这个“单线”数据线在主设备端为MOSI引脚，在从设备端为MISO引脚。
位13	CRCEN: 硬件CRC校验使能 0：禁止CRC计算 1：启动CRC计算 注意：只有在SPI被禁止时(SPE=0)，才能写该位，否则出错。 该位只能在全双工模式下使用。
位12	CRCNEXT: 下一个发送CRC 0：下一个发送的值来自发送缓冲区 1：下一个发送的值来自发送CRC寄存器 注意：最后一个数据被写入SPI_DR寄存器后应马上设置该位。该位只在全双工模式下使用。
位11	DFF: 数据帧格式 0：使用8位数据帧格式进行发送/接收 1：使用16位数据帧格式进行发送/接收 注意：只有当SPI被禁止(SPE=0)时，才能写该位，否则出错。
位10	RXONLY：只接收 该位和BIDIMODE位一起决定在“双线双向”模式下的传输方向。在多个从设备的配置中，在未被访问的从设备上该位被置1，使得只有被访问的从设备有输出，从而不会造成数据线上数据冲突。 0：全双工(发送和接收) 1：禁止输出(只接收模式)

位9	SSM：软件从设备管理 当SSM被置位时，NSS引脚上的电平由SSI位的值决定。 0：禁止软件从设备管理。 1：启用软件从设备管理。
位8	SSI: 内部从设备选择 该位只在SSM被置位时意义：它决定了NSS引脚上的电平，在NSS引脚上操作的I/O输出无效。
位7	LSBFIRST：帧格式 0：先发送MSB 1：先发送LSB 注：当通信在进行时不能改变该位的值。
位6	SPE: SPI使能 0：禁止SPI设备 1：开启SPI设备
位5:3	BR[2:0]：波特率控制 000： $f_{PCLK}/2$ 001： $f_{PCLK}/4$ 010： $f_{PCLK}/8$ 011： $f_{PCLK}/16$ 100： $f_{PCLK}/32$ 101： $f_{PCLK}/64$ 110： $f_{PCLK}/128$ 111： $f_{PCLK}/256$ 当通信正在进行的时候，不能修改这些位。
位2	MSTR: 主设备选择 0：配置为从设备 1：配置为主设备 注意：当通信正在进行的时候，不能修改该位。
位1	CPOL: 时钟极性 0： 空闲状态时，SCK保持低电平 1： 空闲状态时，SCK保持高电平 注意：当通信正在进行的时候，不能修改该位。
位0	CPHA: 时钟相位 0： 数据采样从第一个时钟边沿开始 1： 数据采样从第二个时钟边沿开始 注意：当通信正在进行的时候，不能修改该位。

18.4.2 SPI控制寄存器 2(SPI_CR2)

地址偏移：0x04
复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留								TXEIE	RXNEIE	ERRIE	保留		SSOE	TXDMAEN	RXDMAEN
								rw	rw	rw			rw	rw	rw

位15:8	保留位，硬件强制为0

位7	TXEIE：发送缓冲区空中断使能 0：禁止TXE中断 1：允许TXE中断，当TXE标志置位时产生中断请求 注意：不要同时设置TXEIE和TXDMAEN
位6	RXNEIE：接收缓冲区非空中断使能 0：禁止RXNE中断 1：允许RXNE中断，当RXNE标志置位时产生中断请求 注意：不要同时设置RXEIE和RXDMAEN
位5	ERRIR：错误中断使能 当错误(CRCERR、OVR、MODF)产生时，该位控制是否产生中断 0：禁止错误中断 1：允许错误中断
位4:3	保留位，硬件强制为0
位2	SSOE：SS输出使能 0：禁止在主模式下SS输出，该设备可以工作在多主设备模式 1：设备开启时，开启主模式下SS输出，该设备不能工作在多主设备模式
位1	TXDMAEN：发送缓冲区DMA使能 当该位被设置时，TXE标志一旦被置位就发出DMA请求 0：禁止发送缓冲区DMA 1：启动发送缓冲区DMA
位0	RXDMAEN：接收缓冲区DMA使能 当该位被设置时，RXNE标志一旦被置位就发出DMA请求 0：禁止接收缓冲区DMA 1：启动接收缓冲区DMA

18.4.3 SPI 状态寄存器(SPI_SR)

地址偏移：0x08
复位值：0x0002

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留								BSY	OVR	MODF	CRC ERR	保留		TXE	RXNE
								r	rc	rc	rc			r	r

位15:8	保留位，硬件强制为0
位7	BSY：忙标志 0：SPI不忙 1：SPI正忙于通信，或者发送缓冲非空 该位由硬件置位或者复位
位6	OVR：溢出标志 0：没有出现溢出错误 1：出现溢出错误 该位由硬件置位，由软件序列复位。关于软件序列的详细信息，参考章节18.3.8

位5	MODF：模式错误 0：没有出现模式错误 1：出现模式错误 该位由硬件置位，由软件序列复位。关于软件序列的详细信息，参考章节18.3.8
位4	CRCERR：CRC错误标志 0：收到的CRC值和SPI_RXCRCR寄存器中的值匹配 1：收到的CRC值和SPI_RXCRCR寄存器中的值不匹配 该位由硬件置位，由软件写0而复位 注意：该位只在全双工模式时有意义
位3:2	保留位，硬件强制为0
位1	TXE：发送缓冲为空 0：发送缓冲非空 1：发送缓冲为空
位0	RXNE：接收缓冲非空 0：接收缓冲为空 1：接收缓冲非空

18.4.4 SPI 数据寄存器(SPI_DR)

地址偏移：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 待发送或者已经收到的数据 数据寄存器对应两个缓冲区：一个用于写（发送缓冲）；另外一个用于读（接收缓冲）。写操作将写数据到发送缓冲区；读操作将返回接收缓冲区里的数据。 注意：根据SPI_CR1的DFF位对数据帧格式的选择，数据可以是8位或者16位的。要在启用SPI之前就确定好数据帧格式。 对于8位的数据，发送和接收时只会用到SPI_DR[7:0]。在接收时，SPI_DR[15:8]被强制为0。 对于16位的数据，发送和接收时会用到整个数据寄存器，即SPI_DR[15:0]。

18.4.5 SPI CRC多项式寄存器(SPI_CRCPR)

地址偏移：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 该寄存器包含了CRC计算时用到的多项式。其复位值为0x0007，根据应用要求可以做其他配置。

18.4.6 SPI Rx CRC寄存器(SPI_RXCRCR)

地址偏移：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 在启用CRC计算的情况下，RXCRC[15:0]中包含了依据收到的字节计算的CRC数值。当SPI_CR1的CRCEN位被置位时，该寄存器被复位。CRC计算使用SPI_CRCPR中的多项式。 当数据帧格式被设置为8位时，仅低8位参与计算，并且按照CRC8的方法进行；当数据帧格式为16位时，寄存器中的所有16位都参与计算，并且按照CRC16–CCITT的标准。 注意：当BSY标志被置位时读该寄存器，将可能读到不正确的数值。

18.4.7 SPI Tx CRC寄存器(SPI_TXCRCR)

地址偏移：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 在启用CRC计算的情况下，TXCRC[15:0]中包含了依据将要发送的字节计算的CRC数值。当SPI_CR1中的CRCEN位被置位时，该寄存器被复位。CRC计算使用SPI_CRCPR中的多项式。 当数据帧格式被设置为8位时，仅低8位参与计算，并且按照CRC8的方法进行；当数据帧格式为16位时，寄存器中的所有16个位都参与计算，并且按照CRC16–CCITT的标准。 注意：当BSY标志被置位时读该寄存器，将可能读到不正确的数值。

18.5 SPI 寄存器地址映象

表65 SPI 寄存器列表及其复位值

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
000h	SPI_CR1	保留																BIDIMODE	BIDIOE	CRCEN	CRCNEXT	DFF	RXOnly	SSM	SSI	LSBFIRST	SPE	BR[2:0]			MSTR	CPOL	CPHA
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
004h	SPI_CR2	保留																								TXEIE	RXNEIE	ERRIE	保留		SSOE	TXDMAE	RXDMAE
	复位值																									0	0	0			0	0	0
008h	SPI_SR	保留																								BSY	OVR	MODF	CRCER	保留		TXE	RXNE
	复位值																									0	0	0	0			1	0
00Ch	SPI_DR	保留																DR[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
010h	SPI_CRCPR	保留																CRCPOLY[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1
014h	SPI_RXCRCR	保留																RxCRC[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
018h	SPI_TXCRCR	保留																TxCRC[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

关于寄存器的起始地址，请参见第 1 章。

19 USART 通用同步异步收发器 (USART)

19.1 介绍

通用同步异步收发器（USART）提供了一种灵活的方法来与使用工业标准 NRZ 异步串行数据格式的外部设备之间进行全双工数据交换。USART 利用分数波特率发生器提供宽范围的波特率选择。

它支持同步单向通信和半双工单线通信。它也支持 LIN(局部互连网)，智能卡协议和 IrDA(红外数据组织)SIR ENDEC 规范，以及调制解调器(CTS/RTS)操作。它还允许多处理器通信。

使用多缓冲器配置的 DMA 方式，可以实现高速数据通信。

19.2 主要特性

- 全双工的，异步通信
- NRZ 标准格式
- 分数波特率发生器系统
  - 发送和接收共用的可编程波特率，最高到 4.5Mbits/s
- 可编程数据字长度（8 位或 9 位）
- 可配置的停止位-支持 1 或 2 个停止位
- LIN 主发送同步断开符的能力以及 LIN 从检测断开符的能力
  - 当 USART 硬件配置成 LIN 时，生成 13 位断开符；检测 10/11 位断开符
- 发送方为同步传输提供时钟
- IRDA SIR 编码器解码器
  - 在正常模式下支持 3/16 位的持续时间
- 智能卡模拟功能
  - 智能卡接口支持 ISO7816-3 标准里定义的异步协议智能卡
  - 智能卡用到的 0.5 和 1.5 个停止位
- 单线半双工通信
- 可配置的使用 DMA 的多缓冲器通信

  - 在 SRAM 里利用集中式 DMA 缓冲接收/发送字节
- 单独的发送器和接收器使能位
- 检测标志
  - 接收缓冲器满
  - 发送缓冲器空
  - 传输结束标志
- 校验控制
  - 发送校验位
  - 对接收数据进行校验
- 四个错误检测标志
  - 溢出错误
  - 噪音错误
  - 帧错误
  - 校验错误
- 10 个带标志的中断源
  - CTS 改变
  - LIN 断开符检测
  - 发送数据寄存器空
  - 发送完成
  - 接收数据寄存器满
  - 检测到总线为空闲
  - 溢出错误
  - 帧错误
  - 噪音错误
  - 校验错误
- 多处理器通信 -- 如果地址不匹配，则进入静默模式
- 从静默模式中唤醒（通过空闲总线检测或地址标志检测）
- 两种唤醒接收器的方式：地址位(MSB，第 9 位)，总线空闲

19.3 概述

接口通过三个引脚与其他设备连接在一起(见 图 165)。任何USART双向通信至少需要两个脚：接收数据输入(RX)和发送数据输出(TX)。

RX：接收数据串行输。通过过采样技术来区别数据和噪音，从而恢复数据。

TX：发送数据输出。当发送器被禁止时，输出引脚恢复到它的 I/O 端口配置。当发送器被激活，并且没东西发送时，TX 引脚处于高电平。在单线和智能卡模式里，此 I/O 口被用来发送和接收数据。

- 总线在发送或接收前应处于空闲状态
- 一个起始位
- 一个数据字（8 或 9 位），最低有效位在前
- 0.5，1.5，2 个的停止位，由此表明数据帧的结束
- 使用分数波特率发生器 —— 12 位整数和 4 位小数的表示方法。
- 一个状态寄存器(USART_SR)
- 数据寄存器(USART_DR)
- 一个波特率寄存器(USART_BRR)，12 位的整数和 4 位小数
- 一个智能卡模式下的保护时间寄存器(USART_GTPR)

关于以上寄存器中每个位的具体定义，请参考寄存器描述。

在同步模式中需要下列引脚：

SCLK：发送器时钟输出。此引脚输出用于同步传输的 时钟， (在 Start 位和 Stop 位上没有时钟脉冲，软件可选地，可以在最后一个数据位送出一个时钟脉冲)。数据可以在 RX 上同步被接收。这可以用来控制带有移位寄存器的外部设备(例如 LCD 驱动器)。时钟相位和极性都是软件可编程的。在智能卡模式里，SCLK 可以为智能卡提供时钟。

在 IrDA 模式里需要下列引脚：
IrDA_RDI: IrDA 模式下的数据输入。

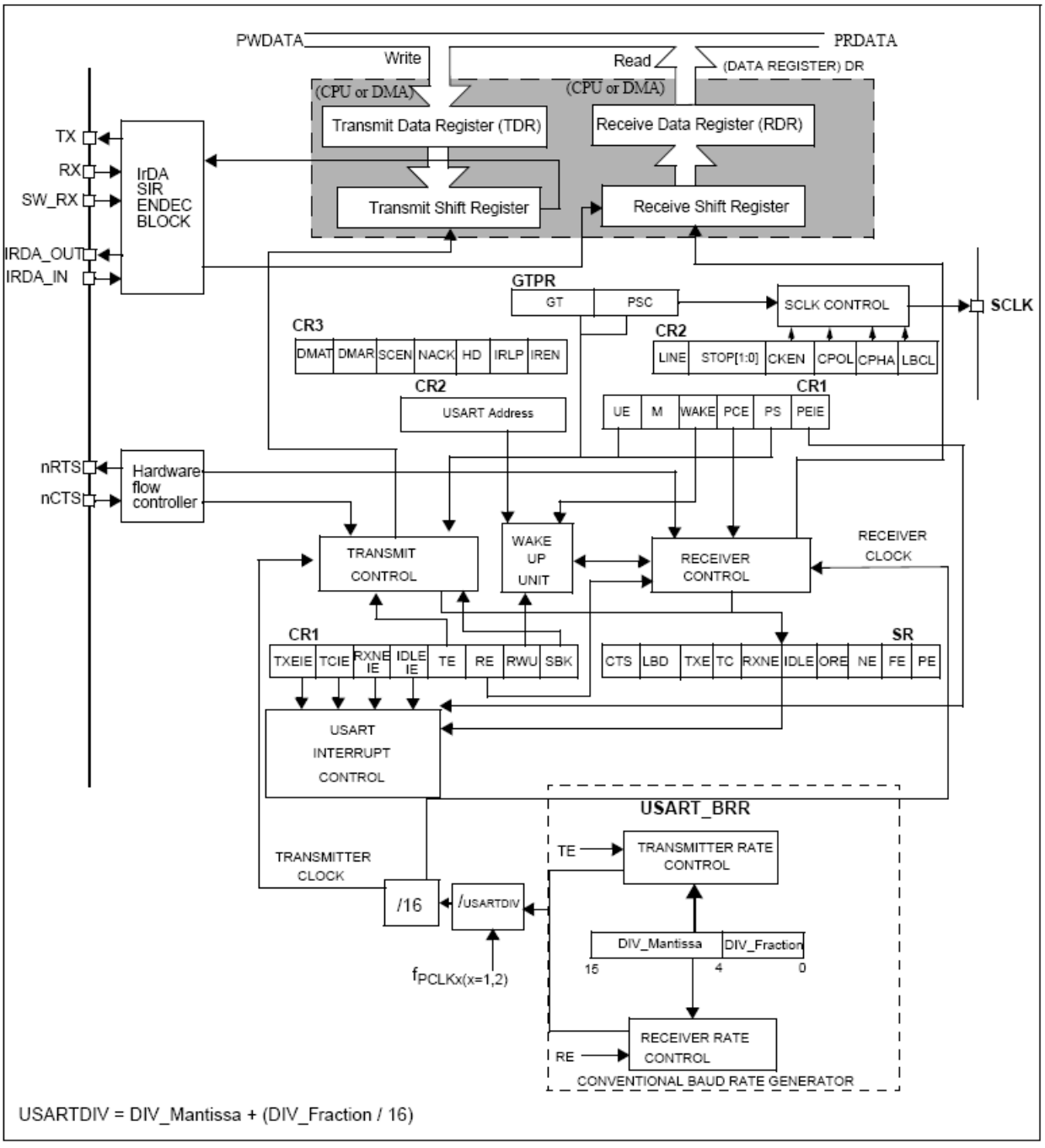
IrDA_TDO: IrDA 模式下的数据输出。

下列引脚在硬件流控模式中需要：
nCTS: 清除发送，若是高电平，在当前数据传输结束时阻断下一次的数据发送。

nRTS: 发送请求，若是低电平，表明 USART 准备好接收数据

19.3.1 框图

图165 USART 框图



19.3.2 USART 特征描述

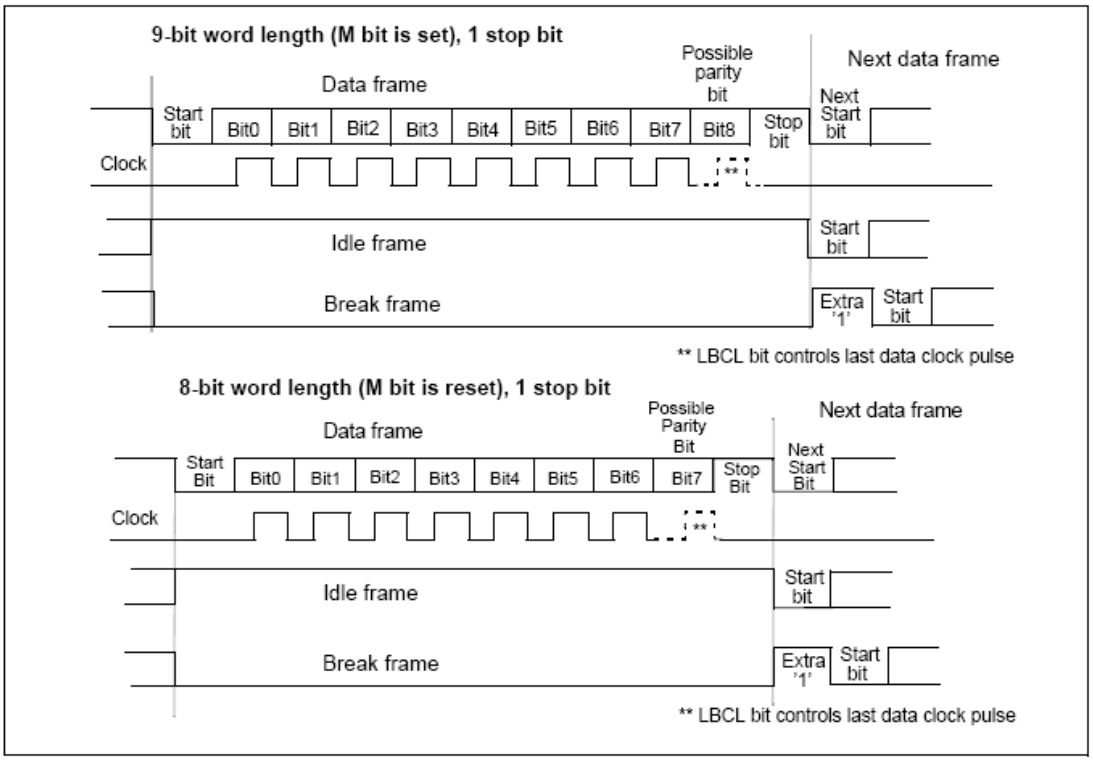
字长可以通过编程USART_CR1 寄存器中的M位，选择成 8 或 9 位(见 图 166)。在起始位期间，TX脚处于低电平，在停止位期间处于高电平。

空闲符号被视为完全由“1”组成的一个完整的数据帧，后面跟着包含了数据的下一帧的开始位。

断开符号 被视为在一个帧周期内全部收到“0”(包括停止位期间，也是“0”)。在断开帧结束时，发送器再插入 1 或 2 个停止位来应答起始位。

发送和接收由一共用的波特率发生器驱动，当发送器和接收器的使能位分别置位时，分别为其产生时钟。每个功能块的详细资料如下给出。

图166 字长设置



19.3.3 发送器

发送器根据 M 位的状态发送 8 位或 9 位的数据字。当发送使能位（TE）被设置时，发送移位寄存器中的数据在 TX 脚上输出，相应的时钟脉冲在 SCLK 脚上输出。

字符发送

在USART发送期间，在TX引脚上首先移出数据的最低有效位。在此模式里，USART_DR寄存器包含了一个内部总线和发送移位寄存器之间的缓冲器(见 图 165)。

每个字符之前都有一个低电平的起始位；之后跟着的停止位，其数目可配置。

注意： 1．在数据传输期间不能复位 TE 位，否则将破坏 TX 脚上的数据，因为波特率计数器停止计数。正在传输的当前数据将丢失。
2． TE 位被激活后将发送一个空闲帧。

可配置的停止位

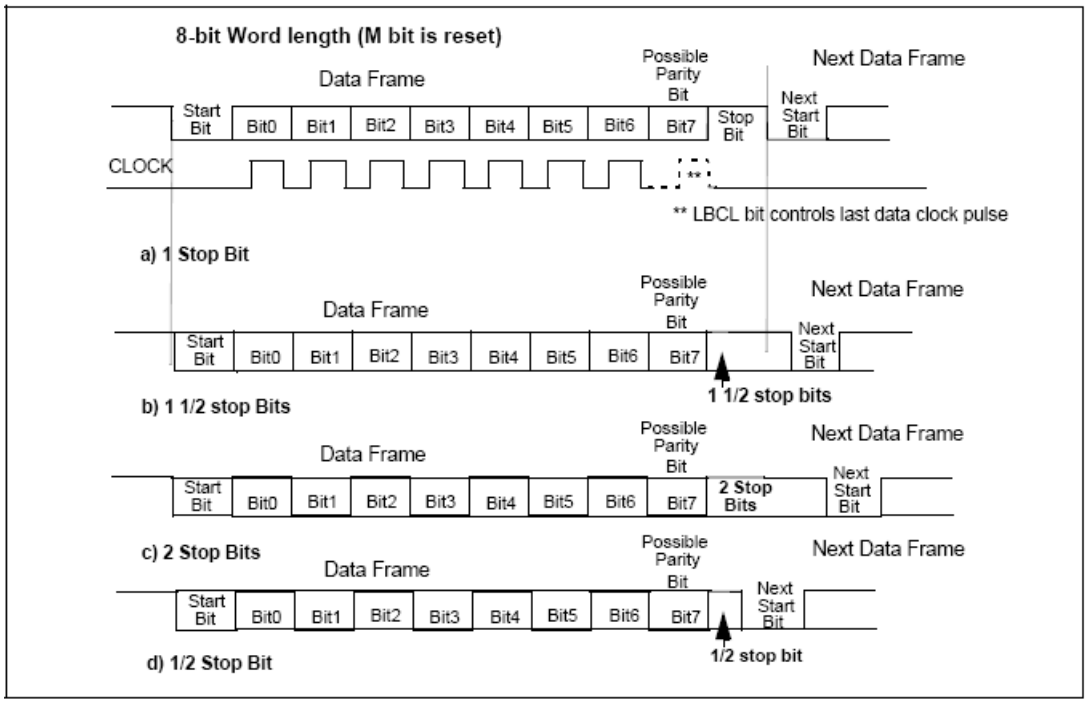
随每个字符发送的停止位的位数可以通过控制寄存器 2 的位 13、12 进行编程。

1. 1个停止位：停止位位数的默认值。
2. 2个停止位：可用于常规USART模式、单线模式以及调制解调器模式。
3. 0.5个停止位：在智能卡模式下接收数据时使用。
4. 1.5个停止位：在智能卡模式下发送数据时使用。

空闲帧包括了停止位。

断开帧是 10 位低电平，后跟停止位（当 m=0 时）；或者 11 位低电平，后跟停止位（m=1 时）。不可能传输更长的断开帧（长度大于 10 或者 11 位）。

图167 配置停止位



配置步骤：

1. 通过在USART_CR1寄存器上置位UE位来激活USART
2. 编程USART_CR1的M位来定义字长。
3. 在USART_CR2中编程停止位的位数。
4. 如果采用多缓冲器通信，配置USART_CR3中的DMA使能位(DMAT)。按多缓冲器通信中的描述配置DMA寄存器。
5. 设置USART_CR1中的TE位，发送一个空闲帧作为第一次数据发送。
6. 利用USART_BRR寄存器选择要求的波特率。
7. 把要发送的数据写进USART_DR寄存器(此动作清除TXE位)。在只有一个缓冲器的情况下，对每个待发送的数据重复步骤7。

单字节通信

清零 TXE 位总是通过对数据寄存器的写操作来完成的。TXE 位由硬件来设置，它表明：

- 数据已经从 TDR 移送到移位寄存器，数据发送已经开始
- TDR 寄存器被清空
- 下一个数据可以被写进 USART_DR 寄存器而不会覆盖先前的数据

如果 TXEIE 位被设置，此标志将产生一个中断。

如果此时 USART 正在发送数据，对 USART_DR 寄存器的写操作把数据存进 TDR 寄存器，并在当前传输结束时把该数据复制进移位寄存器。

如果此时 USART 没有在发送数据，处于空闲状态，对 USART_DR 寄存器的写操作直接把数据放进移位寄存器，数据传输开始，TXE 位立即被置起。当一帧发送完成时（停止位发送后），TC 位被置起，并且如果 USART_CR1 寄存器中的 TCIE 位被置起时，中断产生。

先读一下 USART_SR 寄存器，再写一下 USART_DR 寄存器，可以完成对 TC 位的清零。

注意： *TC 位也可以通过对它软件写 0 来清除。此清零方式只在多缓冲器通信模式下推荐使用。*

断开符号

置位SBK位可发送一个断开符号。断开帧长度取决M位(见 图 166)。如果SBK位被置 1，在完成当前数据发送后，将在TX线上发送一个断开符号。断开字符发送完成时（在断开符号的停止位时）SBK被硬件复位。USART在最后一个断开帧的结束处插入一逻辑 1 位，以保证能识别下一帧的起始位。

注意：如果在开始发送断开帧之前，软件又复位了 SBK 位，断开符号将不被发送。如果要发送两个连续的断开帧，SBK 位应该在前一个断开符号的停止位之后置起。

空闲符号

置位 TE 将使得 USART 在第一个数据帧前发送一空闲帧。

19.3.4 接收器

USART 可以根据 USART_CR1 的 M 位接收 8 位或 9 位的数据字

字符接收

在 USART 接收期间，数据的最低有效位首先从 RX 脚移进。在此模式里，USART_DR 寄存器包含的缓冲器位于内部总线和接收移位寄存器之间。

配置步骤：

1. 将USART_CR1寄存器的UE置1来激活USART。
2. 编程USART_CR1的M位定义字长
3. 在USART_CR2中编写停止位的个数
4. 如果需多缓冲器通信，选择USART_CR3中的DMA使能位（DMAT）。按多缓冲器通信所要求的配置DMA寄存器。
5. 利用波特率寄存器USART_BRR选择希望的波特率。
6. 设置USART_CR1的RE位。激活接收器，使它开始寻找起始位。

当一字符被接收到时，

- RXNE 位被置位。它表明移位寄存器的内容被转移到 RDR。换句话说，数据已经被接收并且可以被读出（包括与之有关的错误标志）。
- 如果 RXNEIE 位被设置，产生中断。
- 在接收期间如果检测到帧错误，噪音或溢出错误，错误标志将被置起，
- 在多缓冲器通信时，RXNE 在每个字节接收后被置起，并由 DMA 对数据寄存器的读操作而清零。
- 在单缓冲器模式里，由软件读 USART_DR 寄存器完成对 RXNE 位清除。RXNE 标志也可以通过对它写 0 来清除。RXNE 位必须在下一字符接收结束前被清零，以避免溢出错误。

注意： *在接收数据时，RE 位不应该被复位。如果 RE 位在接收时被清零，当前字节的接收被丢失。*

断开符号

当接收到一个断开帧时，USART 像处理帧错误一样处理它。

空闲符号

当一空闲帧被检测到时，其处理步骤和接收到普通数据帧一样，但如果 IDLEIE 位被设置将产生一个中断。

溢出错误

如果 RXNE 还没有被复位，又接收到一个字符，则发生溢出错误。数据只有当 RXNE 位被清零后才能从移位寄存器转移到 RDR 寄存器。RXNE 标记是接收到

每个字节后被置位的。如果下一个数据已被收到或先前 DMA 请求还没被服务时，RXNE 标志仍是置起的，溢出错误产生。
当溢出错误产生时：

- ORE 位被置位。
- RDR 内容将不会丢失。读 USART_DR 寄存器仍能得到先前的数据。
- 移位寄存器中以前的内容将被覆盖。随后接收到的数据都将丢失。
- 如果 RXNEIE 位被设置或 EIE 和 DMAR 位都被设置，中断产生。
- 顺序执行对 USART_SR 和 USART_DR 寄存器的读操作，可复位 ORE 位

注意： *当 ORE 位置位时，表明至少有 1 个数据已经丢失。有两种可能性：*

- 如果 RXNE=1，上一个有效数据还在接收寄存器 RDR 上，可以被读出。
- 如果 RXNE=0，这意味着上一个有效数据已经被读走，RDR 已经没有东西可读。当上一个有效数据在 RDR 中被读取的同时又接收到新的（也就是丢失的）数据时，此种情况可能发生。在读序列期间（在 USART_SR 寄存器读访问和 USART_DR 读访问之间）接收到新的数据，此种情况也可能发生。

噪音错误

使用过采样技术（同步模式除外），通过区别有效输入数据和噪音来进行数据恢复。

图168 检测噪声的数据采样

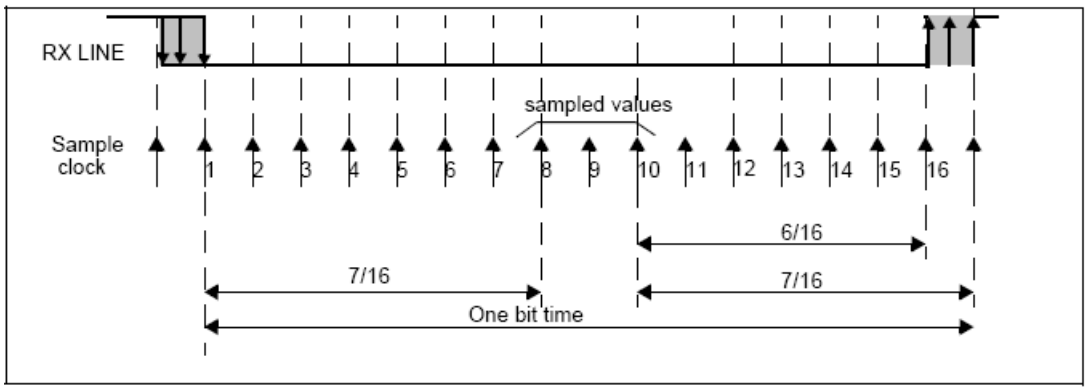


表66 检测噪声的数据采样

采样值	NE状态	接收的位值	数据有效性
000	0	0	有效
001	1	0	无效
010	1	0	无效
011	1	1	无效
100	1	0	无效
101	1	1	无效

110	1	1	无效
111	0	1	有效

当在接收帧中检测到噪音时：

- NE 在 RXNE 位的上升沿被置起。
- 无效数据从移位寄存器移送到 USART_DR 寄存器。
- 在单个字节通信情况下，没有中断产生。然而，NE 这个位和 RXNE 位同时置起，后者自己产生中断。在多缓冲器通信情况下，如果 USART_CR3 寄存器中 EIE 位被置位的话，将产生一中断

顺序执行对 USART_SR 和 USART_DR 寄存器的读操作，可复位 NE 位

帧错误

当以下情况发生时检测到帧错误：
由于没有同步上或大量噪音的原因，停止位没有在预期的时间上接和收识别出来。

当帧错误被检测到时：

- FE 位被硬件置起
- 无效数据从移位寄存器传送到 USART_DR 寄存器。
- 在单个字节通信情况下，没有中断产生。然而，这个位和 RXNE 位同时置起，后者自己产生中断。在多缓冲器通信情况下，如果 USART_CR3 寄存器中 EIE 位被置位的话，将产生一中断。

顺序执行对 USART_SR 和 USART_DR 寄存器的读操作，可复位 FE 位。

接收期间的可配置的停止位

被接收的停止位的个数可以通过控制寄存器 2 的控制位来配置，在正常模式时，可以是 1 或 2 个，在智能卡模式里可能是 0.5 或 1.5 个。

1. 0.5个停止位（智能卡模式里的接收）：不对0.5个停止位进行采样。因此，如果选择0.5个停止位则不能检测帧错误和断开帧。
2. 1个停止位：对1个停止位的采样在第8，第9和第10采样点上进行。
3. 1.5 个停止位(智能卡模式里的发送)：当以智能卡模式发送时，器件必须检查数据是否被正确的发送出去。所以接收器功能块必须被激活(USART_CR1寄存器中的RE =1)，并且在停止位的发送期间采样数据线上的信号。如果出现校验错误，智能卡会在发送方采样NACK信号时，即总线上停止位对应的时间内时，拉低数据线，以此表示出现了帧错误。FE在1.5个停止位结束时和RXNE一起被置起。对1.5个停止位的采样是在第16，第17和第18采样点进行的。1.5个的停止位可以被分成2部分：一个是

0.5个时钟周期，期间不做任何事情 。随后是1个时钟周期的停止位，在这段时间的中点处采样。参考19.3.11智能卡，以得到更多详细资料。

4. 2个停止位：对2个停止位的采样是在第一停止位的第8，第9和第10个采样点完成的。如果第一个停止位期间检测到一个帧错误，帧错误标志将被设置。第二个停止位不再检查帧错误。在第一个停止位结束时RXNE标志将被设置。

19.3.5 分数波特率的产生

接收器和发送器（Rx 和 Tx）的波特率在 USARTDIV 的整数和小数寄存器中的值应设置成相同。

$$\text{Tx / Rx 波特率} = \frac{f_{PCLKx}}{(16 * USARTDIV)}$$

这里的f_{PCLKx}(x=1、2)是给外设的时钟(PCLK1用于USART2、3，PCLK2用于USART1)

USARTDIV 是一个无符号的定点数。这 12 位的值设置在 USART_BRR 寄存器。

如何从USART_BRR寄存器值得到USARTDIV

例 1：

如果 DIV_Mantissa = 27d ， DIV_Fraction = 12d (USART_BRR=1BCh),
于是
Mantissa (USARTDIV) = 27d
Fraction (USARTDIV) = 12/16 = 0.75d
所以 USARTDIV = 27.75d

例 2：

要求 USARTDIV = 25.62d,
就有：
DIV_Fraction = 16*0.62d = 9.92d, 近似等于 10d = 0x0A
DIV_Mantissa = mantissa (25.620d) = 25d = 0x19
于是, USART_BRR = 0x19A

例 3：

要求 USARTDIV = 50.99d
就有：
DIV_Fraction = 16*0.99d = 15.84d =>近似等于 16d = 0x10
DIV_Mantissa = mantissa (50.990d) = 50d = 0x32

注意： 更新波特率寄存器 USART_BRR 后，波特率计数器中的值也立刻随之更新。所以在通信进行时不应改变 USART_BRR 中的值。

表67 设置波特率时的误差计算

波特率		f_{PCLK} = 36MHz			f_{PCLK} = 72MHz		
序号	Kbps	实际	置于波特率寄存器中的值	误差%	实际	置于波特率寄存器中的值	误差%
1	2.4	2.400	937.5	0%	2.4	1875	0%
2	9.6	9.600	234.375	0%	9.6	468.75	0%
3	19.2	19.2	117.1875	0%	19.2	234.375	0%
4	57.6	57.6	39.0625	0%	57.6	78.125	0%
5	115.2	115.384	19.5	0.15%	115.2	39.0625	0%
6	230.4	230.769	9.75	0.16%	230.769	19.5	0.16%
7	460.8	461.538	4.875	0.16%	461.538	9.75	0.16%
8	921.6	923.076	2.4375	0.16%	923.076	4.875	0.16%
9	2250	2250	1	0%	2250	2	0%
10	4500	不可能	不可能	不可能	4500	1	0%

注： 1. CPU 的时钟频率越低某一特定波特率的误差也越低
2. 只有 USART1 使用 PCLK2(最高 72MHz)。其它 USART 使用 PCLK1(最高 36MHz)。

19.3.6 多处理器通信

通过 USART 可以实现多处理器通信（将几个 USART 连在一个网络里）。例如某个 USART 设备可以是主，它的 TX 输出和其他 USART 从设备的 RX 输入相连接；USART 从设备各自的 TX 输出逻辑地与在一起，并且和主设备的 RX 输入相连接。

在多处理器配置中，我们通常希望只有被寻址的接收者才被激活，来接收随后的数据，这样就可以减少由未被寻址的接收器的参与带来的多余的 USART 服务开销。

未被寻址的设备可启用其静默功能置于静默模式。在静默模式里：

- 任何接收状态位都不会被设置。
- 所有接收中断被禁止。
- USART_CR1 寄存器中的 RWU 位被置 1。RWU 可以被硬件自动控制或在某个条件下由软件写。

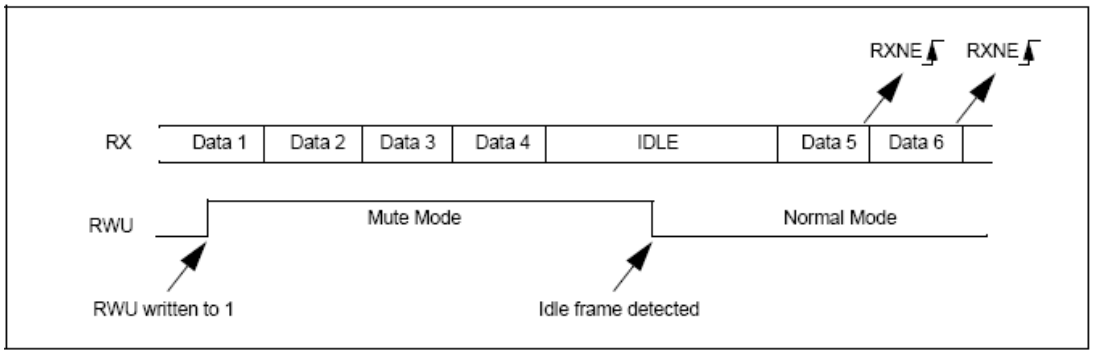
根据 USART_CR1 寄存器中的 WAKE 位状态，USART 可以用二种方法进入或退出静默模式。

- 如果 WAKE 位被复位：进行空闲总线检测。
- 如果 WAKE 位被设置：进行地址标记检测。

空闲总线检测(WAKE=0)

当RWU位被写 1 时，USART进入静默模式。当检测到一空闲帧时，它被唤醒。然后RWU被硬件清零，但是USART_SR寄存器中的IDLE位并不置起。RWU还可以被软件写 0。图 169 给出利用空闲总线检测来唤醒和进入静默模式的一个例子

图169 利用空闲总线检测的静默模式



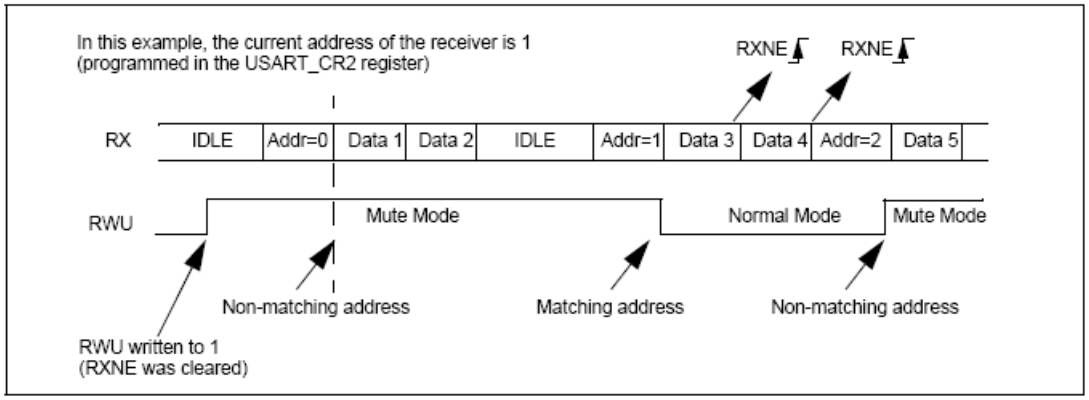
地址标记（address mark）检测(WAKE=1)

在这个模式里，如果 MSB 是 1，该字节被认为是地址，否则被认为是数据。在一个地址字节中，目标接收器的地址被放在 4 个 LSB 中。这个 4 位地址被接收器同它自己地址做比较，接收器的地址被编程在 USART_CR2 寄存器的 ADD。
如果接收到的字节与它的编程地址不匹配时，USART 进入静默模式。该字节的接收既不会置起 RXNE 标志也不会产生中断或发出 DMA 请求，因为 USART 已经在静默模式。

当接收到的字节与接收器内编程地址匹配时，USART 退出静默模式。然后 RWU 位被清零，随后的字节被正常接收。匹配的地址字节将置位 RXNE 位，因为 RWU 位已被清零。

当接收缓冲器不包含数据时（USART_SR的RXNE=0），RWU位可以被写 0 或 1。否则，该次写操作被忽略。图 170 给出利用地址标记检测来唤醒和进入静默模式的例子。

图170 利用地址标记检测的静默模式



19.3.7 校验控制

奇偶控制（发送时生成一个奇偶位，接收时进行奇偶校验）可以通过设置USART_CR1 寄存器上的PCE位而激活。根据M位定义的帧长度，可能的USART帧格式列在 表 68 中。

表68 帧格式

M位	PCE位	USART帧
0	0	| 起始位 | 8位数据 | 停止位 |
0	1	| 起始位 | 7位数据 | 奇偶检验位 | 停止位 |
1	0	| 起始位 | 9位数据 | 停止位 |
1	1	| 起始位 | 8位数据 | 奇偶检验位 | 停止位 |

注意： 在用地址标记唤醒设备时，地址的匹配只考虑到数据的MSB 位，而不用关心校验位。（MSB 是数据位中最后发出的，后面紧跟校验位或者停止位）

偶校验：校验位使得一帧中的 7 或 8 个 LSB 数据以及校验位中“1”的个数为偶数。
例如：data=00110101，有 4 个“1”，如果选择偶校验（在 USART_CR1 中的 PS＝0），校验位将是 0。

奇校验：此校验位使得一帧中的 7 或 8 个 LSB 数据以及校验位中“1”的个数为奇数。
例如：data=00110101，有 4 个“1”，如果选择奇校验（在 USART_CR1 中的 PS＝1），校验位将是 1。

传输模式：如果 USART_CR1 的 PCE 位被置位，写进数据寄存器的数据的 MSB 位被校验位替换后发送出去（如果选择偶校验偶数个 1，如果选择奇校验奇数个 1）。如果奇偶校验失败，USART_SR 寄存器中的 PE 标志被置 1，并且如果 USART_CR1 寄存器的 PEIE 在被预先设置的话，中断产生。

19.3.8 LIN（局域互联网）模式

LIN 模式是通过设置 USART_CR2 寄存器的 LINEN 位选择的。在 LIN 模式里，下列位必须保持清零：

- USART_CR2 寄存器的 CLKEN 位
- USART_CR3 寄存器的 STOP[1:0]，SCEN，HDSEL 和 IREN

LIN发送

19.3.3 节里所描述的同样步骤适用于LIN主发送，但和正常USART发送有以下区别：

- 清零 M 位以配置 8-位字长
- 置位 LINEN 位以进入 LIN 模式。这时，置位 SBK 将发送 13 位“0”作为断开符号。然后发一位“1”，以允许对下一个开始位的检测。

LIN接收

当 LIN 模式被使能时，断开符号检测电路被激活。该检测完全独立于 USART 接收器。断开符号只要一出现就能检测到，不管是在总线空闲时还是在发送某数据帧期间，数据帧还未完成，又插入了断开符号的发送。

当接收器被激活时(USART_CR1 的 RE=1)，电路监测 RX 上的起始信号。监测起始位的方法同检测断开符号或数据是一样的。当起始位被检测到后，电路对每个接下来的位，在每个位的第 8，9，10 个过采样时钟点上进行采样。如果 10 个(当 USART_CR2 的 LBDL = 0)或 11 个(当 USART_CR2 的 LBDL = 1)连续位都是“0”，并且又跟着一个定界符，USART_SR 的 LBD 标志被设置。如果 LBDIE 位=1，中断产生。在确认断开符号前，要检查定界符，因为它意味 RX 线已经回到高电平。

如果在第 10 或 11 个采样点之前采样到了“1”，检测电路取消当前检测并重新寻找起始位。如果 LIN 模式被禁止，接收器继续如正常 USART 那样工作，不需要考虑检测断开符号。

如果 LIN 模式被激活(LINEN=1)，只要一发生帧错误(也就是停止位检测到“0”，这种情况出现在断开帧)，接收器就停止，直到断开符号检测电路接收到一个“1”(这种情况发生于断开符号没有完整的发出来)，或一个定界符(这种情况发生于已经检测到一个完整的断开符号)。

图 171 说明了断开符号检测器状态机的行为和断开符号标志的关系。

图 172 给了一个断开帧的例子。

图171 LIN 模式下的断开检测(11 位断开长度 – 设置了 LBDL 位)

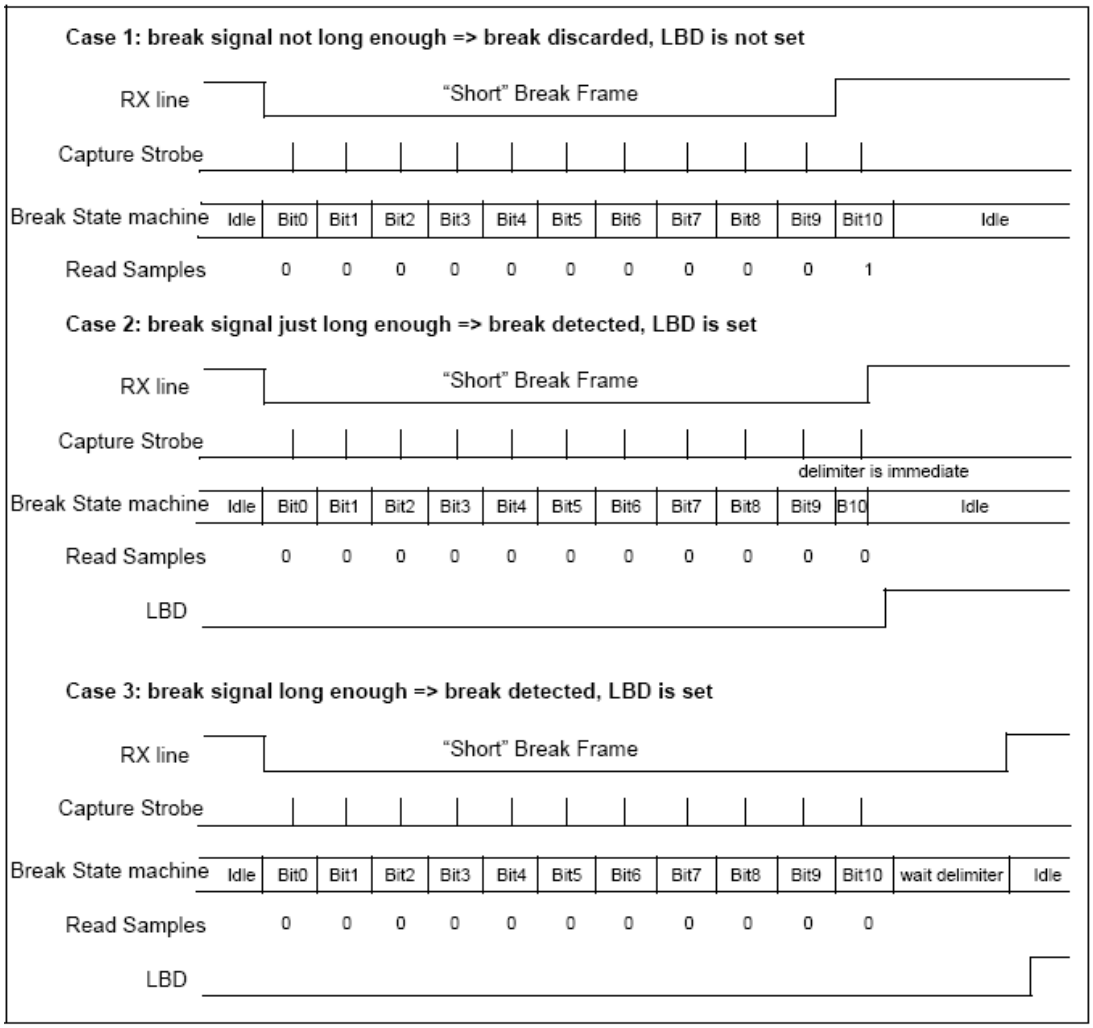
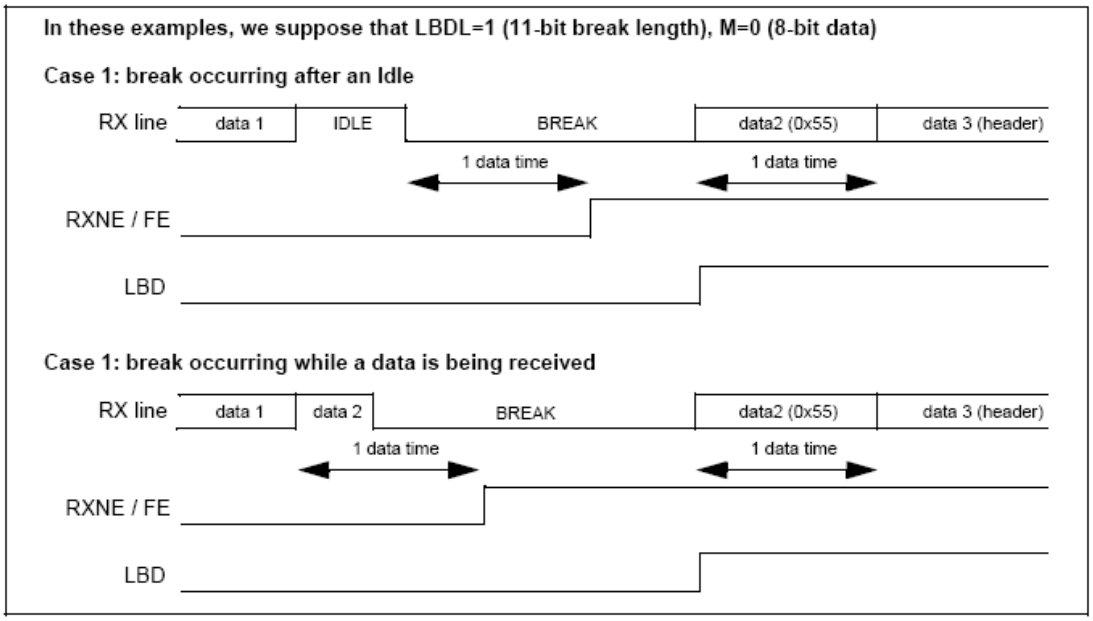


图172 LIN 模式下的断开检测与帧错误的检测



19.3.9 USART 同步模式

通过在 USART_CR2 寄存器上写 CLKEN 位选择同步模式

在同步模式里，下列位必须保持清零状态：

- USART_CR2 寄存器中的 LINEN 位
- USART_CR3 寄存器中的 SCEN,HDSEL 和 IREN 位

USART允许用户以主模式方式控制双向同步串行通信。SCLK脚是USART发送器时钟的输出。在起始位和停止位期间，SCLK脚上没有时钟脉冲。根据USART_CR2 寄存器中LBCL位的状态，决定在最后一个有效数据位期间产生或不产生时钟脉冲。USART_CR2 寄存器的CPOL位允许用户选择时钟极性，USART_CR2 寄存器上的CPHA位允许用户选择外部时钟的相位(见 图 173，图 174 和 图 175)。

在总线空闲期间，实际数据到来之前以及发送断开符号的时候，外部 SCLK 时钟不被激活。

同步模式时，USART 发送器和异步模式里工作一模一样。但是因为 SCLK 是与 TX 同步的(根据 CPOL 和 CPHA)，所以 TX 上的数据是随 SCLK 同步发出的。

同步模式的 USART 接收器工作方式与异步模式不同。如果 RE=1，数据在 SCLK 上采样（根据 CPOL 和 CPHA 决定在上升沿还是下降沿），不需要任何的过采样。但必须考虑建立时间和持续时间（取决于波特率，1/16 位时间）。

注意：1 · SCLK 脚同 TX 脚一起联合工作。因而，只有在发送器被激活(TE =1)，且数据被发送时(USART_DR 寄存器被写入)才提供时钟。这意味着在没有发送数据时是不可能接收一个同步数据的。
2 · LBCL,CPOL 和 CPHA 位的正确配置，应该在发送器和接收器都被禁止时；当发送器或接收器被激活时，这些位不能被改变

3．建议在同一条指令中设置 TE 和 RE，以减少接收器的建立时间和保持时间。
4． USART 只支持主模式：它不能用来自其他设备的输入时钟接收或发送数据 (SCLK 永远是输出)。

图173 USART 同步传输的例子

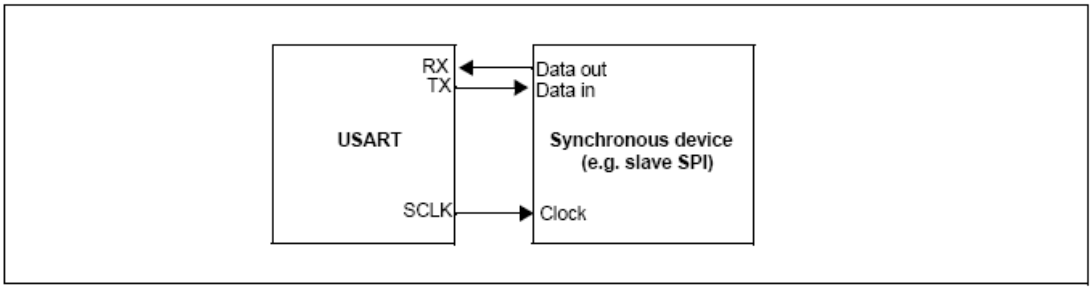


图174 USART 数据时钟时序示例(M=0)

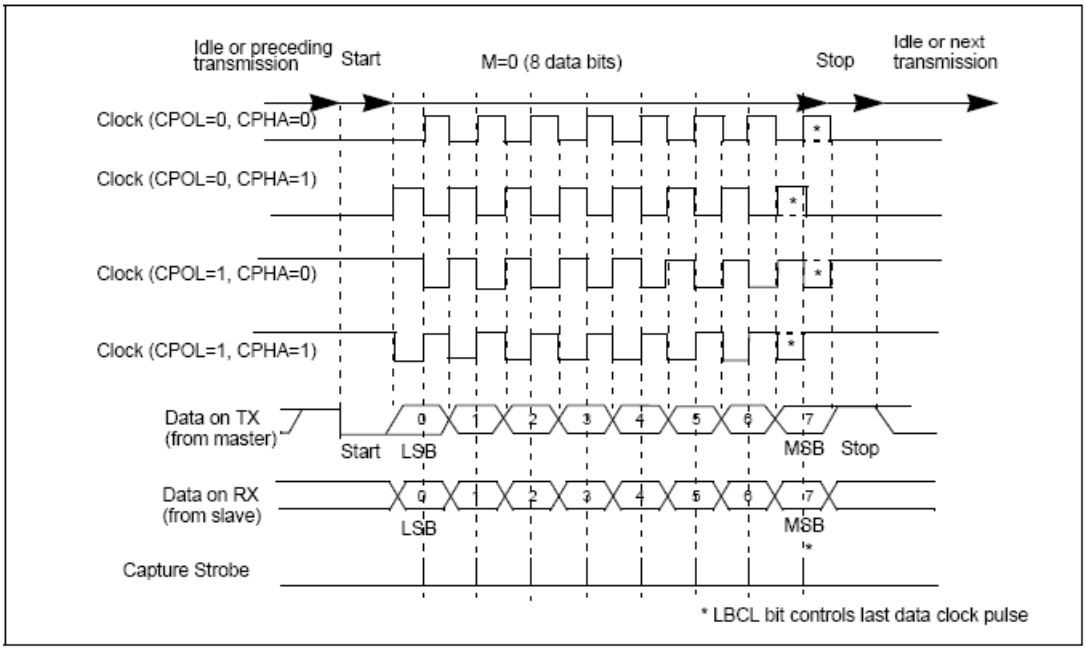


图175 USART 数据时钟时序示例(M=1)

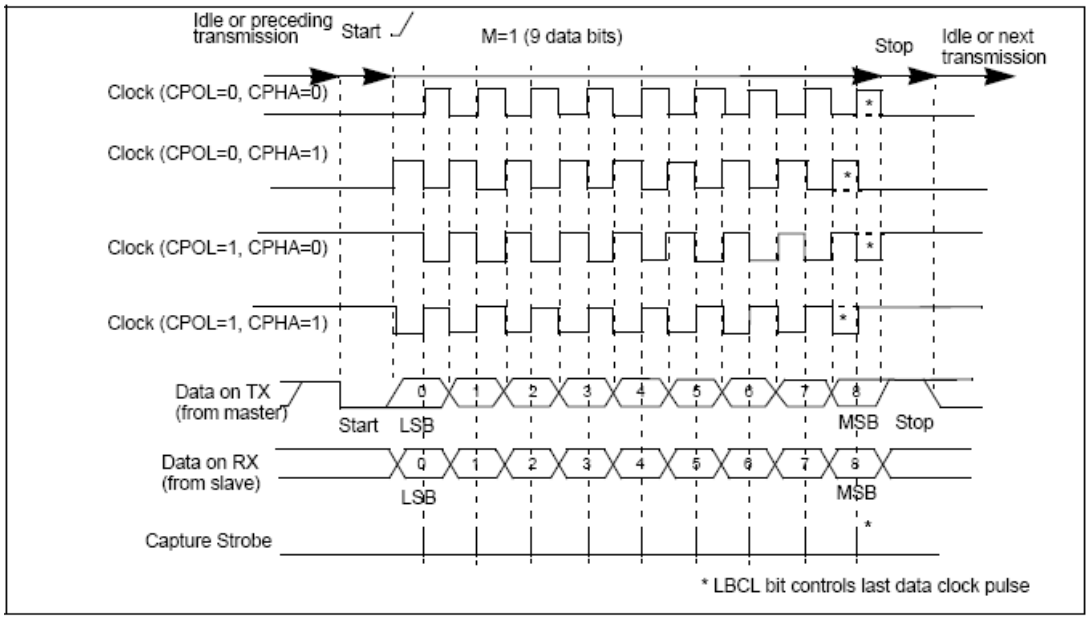
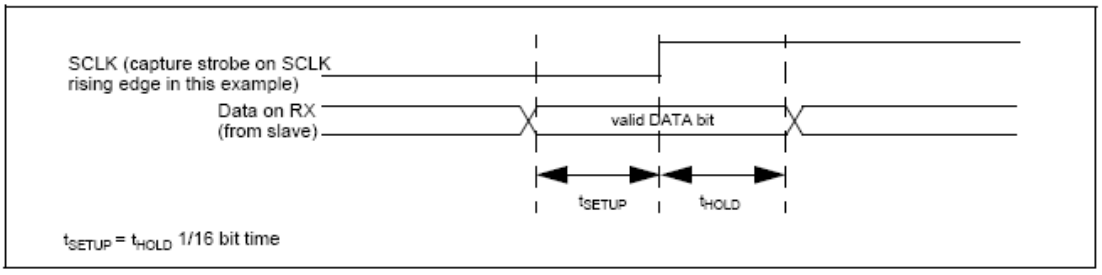


图176 RX 数据采样/保持时间



注： 在智能卡模式下SCLK 的功能不同，有关细节请参考智能卡模式部分。

19.3.10 单线半双工通信

单线半双方模式通过设置 USART_CR3 寄存器的 HDSEL 位选择。在这个模式里，下面的位必须保持清零状态：

- USART_CR2 寄存器的 LINEN 和 CLKEN 位
- USART_CR3 寄存器的 SCEN 和 IREN 位

USART 可以配置成遵循单线半双工协议。半双工和全双工通信是用控制位“HALF DUPLEX SEL”选择的。当 HDSEL 写“1”时

- RX 不再被使用
- 当没有数据传输时，TX 总是被释放。因此，它在空闲状态的或接收状态时表现为一个标准 I/O 口。这就意味该 I/O 在不被 USART 驱动时，必须配置成悬空输入（或开漏的输出高）。

除此以外，通信与正常 USART 模式类似。由软件来管理线上的冲突（例如通过使用一个中央仲裁器）。特别的是，发送从不会被硬件所阻碍。当 TE 位被设置时，只要数据一写到数据寄存器上，发送就继续。

19.3.11 智能卡

智能卡模式通过设置 USART_CR3 寄存器的 SCEN 位选择。在智能卡模式里，下列位必须保持清零：

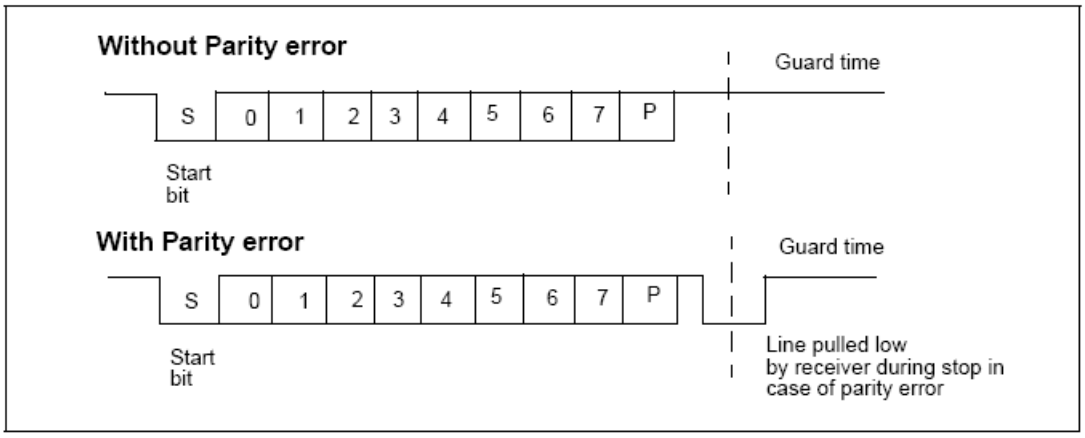
- USART_CR2 寄存器的 LINEN 位
- USART_CR3 寄存器的 HDSEL 位和 IREN 位

此外，CLKEN 位可以被设置，以提供时钟给智能卡。智能卡接口设计成 ISO7816-3 标准所定义的那样支持异步协议的智能卡。USART 应该被设置为：

- 8 位数据位加校验位： 此时 USART_CR1 寄存器 M=1,PCE=1，并且下列条件满足其一：
  - 接收时 0.5 个停止位：即 USART_CR2 寄存器的 STOP=01
  - 发送时 1.5 个停止位：即 USART_CR2 寄存器的 STOP=11

图 177 给出的例子说明了数据线上，在有校验错误和没校验错误两种情况下的信号。

图177 ISO7816-3 异步协议



当与智能卡相连接时，USART 的 TX 驱动一根智能卡也驱动的双向线。为了做到这点，SW_RX 必须和 TX 连接到相同的 I/O 口。在发送开始位和数据字节期间，发送器的输出使能位 TX_EN 被置起，在发送停止位期间被释放（弱上拉），因此在发现校验错误的情况下接收器可以将数据线拉低。如果 TX_EN 不被使用，在停止位期间 TX 被拉到高电平：这样的话，只要 TX 配置成开漏，接收器也可以驱动这根线。

智能卡是一个单线半双工通信协议

- 从发送移位寄存器把数据发送出去，要被延时最小 1/2 波特时钟。在正常操作时，一个满的发送移位寄存器将在下一个波特时钟沿开始向外移出数据。在智能卡模式里，此发送被延迟 1/2 波特时钟。
- 如果在接收数据帧期间，检测到一校验错误，该帧接收完成后(也就是在 0.5 停止位结束时)，发送线被拉低一个波特时钟周期。这是告诉智能卡发送到 USART 的数据没有被正确接收到。此 NACK 信号（拉低发送线一个波特时钟周期）在发送端将产生一个帧错误(发送端被配置成 1.5 个停止位)。应用

程序可以根据协议处理重新发送的数据。如果 NACK 控制位被设置，发生校验错误时接收器会给出一个 NACK 信号；否则就不会发送 NACK。

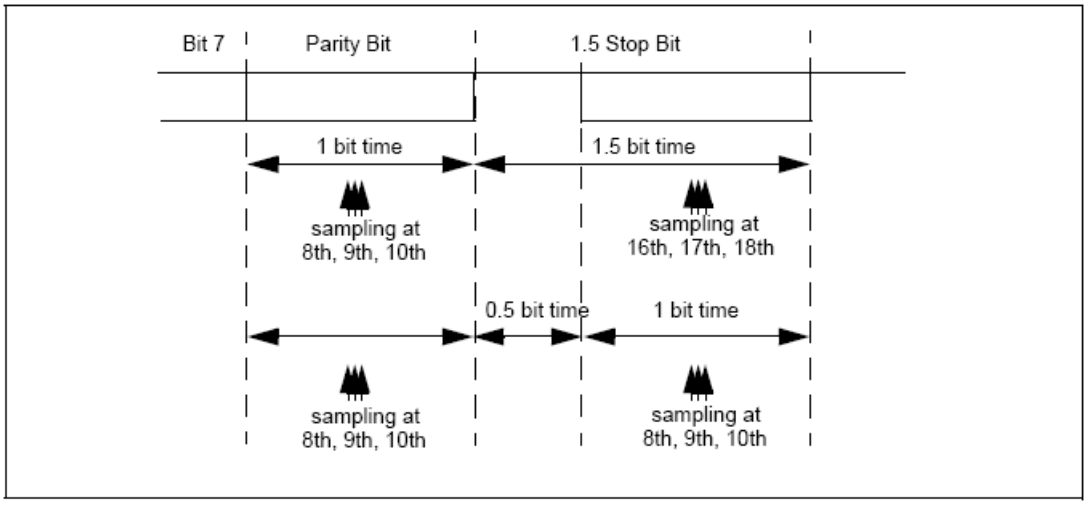
- TC 标志的置起可以通过编程保护时间寄存器得以延时。在正常操作时，当发送移位寄存器变空并且没有新的发送请求出现时，TC 被置起。在智能卡模式里，空的发送移位寄存器将触发保护时间计数器开始向上计数，直到保护时间寄存器中的值。TC 在这段时间被强制拉低。当保护时间计数器达到保护时间寄存器中的值时，TC 被置高。
- TC 标志的撤销不受智能卡模式的影响。
- 如果在发送器端检测到一个帧错误(收到接收器的 NACK 信号)，发送器的接收功能块不会把 NACK 当作起始位检测。根据 ISO 协议，接收到的 NACK 的持续时间可以是 1 或 2 波特时钟周期。
- 在接收器这边，如果一个校验错误被检测到，并且 NACK 被发送，接收器不会把 NACK 检测成起始位。

注意： 1. 断开符号在智能卡模式里没有意义。一个带帧错误的 00h 数据将被当成数据而不是断开符号。

2. 当来回切换 TE 位时，没有 IDLE 帧被发送。ISO 协议没有定义 IDLE 帧。

图 178 详述了USART是如何采样NACK信号的。在这个例子里，USART正在发送数据，并且被配置成 1.5 个停止位。为了检查数据的完整性和NACK信号，USART的接收功能块被激活。

图178 使用 1.5 停止位检测奇偶检验错



USART 可以通过 SCLK 输出为智能卡提供时钟。在智能卡模式里，SCLK 不和通信直接关联，而是先通过一个 5 位预分频器简单地用内部的外设输入时钟来驱动智能卡的时钟。分频率在预分频寄存器 USART_GTPR 中配置。SCLK 频率可以从 fck/2 到 fck/62，这里的 fck 是外设输入时钟。

19.3.12 IrDA SIR ENDEC 功能块

IrDA 模式是通过设置 USART_CR3 寄存器的 IREN 位选择的。在 IRDA 模式里，下列位必须保持清零：

- USART_CR2 寄存器的 LINEN,STOP 和 CLKEN 位
- USART_CR3 寄存器的 SCEN 和 HDSEL 位。

IrDA SIR物理层规定使用反相归零调制方案（RZI），该方案用一个红外光脉冲代表逻辑 0（见 图 179）。SIR发送编码器对从USART输出的NRZ（非归零）比特流进行调制。输出脉冲流被传送到一个外部输出驱动器和红外LED。USART为SIR ENDEC最高只支持到 115.2Kbps速率。在正常模式里，脉冲宽度规定为一个位周期的 3/16。

SIR 接收解码器对来自红外接收器的归零位比特流进行解调，并将接收到的 NRZ 串行比特流输出到 USART。在空闲状态里，解码器输入通常是高(标记状态 marking state)。发送编码器输出的极性和解码器的输入相反。当解码器输入低时，检测到一个起始位。

- IrDA 是一个半双工通信协议。如果发送器忙(也就是 USART 正在送数据给 IrDA 编码器)，IrDA 接收线上的任何数据将被 IrDA 解码器忽视。如果接收器忙(也就是 USART 正在接收从 IrDA 解码器来的解码数据)，从 USART 到 IrDA 的 TX 上的数据将不会被 IrDA 编码。当接收数据时，应该避免发送，因为将被发送的数据可能被破坏。
- SIR发送逻辑把“0”作为高脉冲发送，把“1”作为低电平发送。脉冲的宽度规定为正常模式时位周期的 3/16(见 图 180)。
- SIR 接收逻辑把高电平状态解释为“1”，把低脉冲解释为“0”。
- 发送编码器输出与解码器输入有着相反的极性。当空闲时，SIR 输出处于低状态。
- SIR 解码器把 IrDA 兼容的接收信号转变成给 USART 的比特流。
- IrDA 规范要求脉冲要宽于 1.41us。脉冲宽度是可编程的。接收器端的尖峰脉冲检测逻辑滤除宽度小于 2 个 PSC 周期的脉冲（PSC 是在 IrDA 低功耗波特率寄存器 USART_GTPR 中编程的预分频值）。宽度小于 1PSC 周期的脉冲一定被滤除掉，但是那些宽度大于 1 个而小于 2 个 PSC 周期的脉冲可能被接收或滤除，那些宽度大于 2 个周期的将被视为一个有效的脉冲。当 PSC=0 时，IrDA 编码器/解码器不工作。
- 接收器可以与一低功耗发送器通信。
- 在 IrDA 模式里，USART_CR2 寄存器上的 STOP 位必须配置成 1 个停止位。

IrDA低功耗模式

发送器

在低功耗模式，脉冲宽度不再持续 3/16 个位周期。取而代之，脉冲的宽度是低功耗波特率的 3 倍，它最小可以是 1.42MHz。通常这个值是 1.8432MHz（1.42 MHz < PSC< 2.12 MHz）。一个低功耗模式可编程分频器把系统时钟进行分频以达到这个值。

接收器

低功耗模式的接收类似于正常模式的接收。为了滤除尖峰干扰脉冲，USART 应该滤除宽度短于 1 个 PSC 的脉冲。只有持续时间大于 2 个周期的 IrDA 低功耗波特率时钟（USART_GTPR 中的 PSC）的低电平信号才被接受为有效的信号。

注意： *1 宽度小于 2 个大于 1 个 PSC 周期的脉冲可能会也可能不会被滤除。*

2 接收器的建立时间应该由软件管理。IrDA 物理层技术规范规定了在发送和接收之间最小要有 10ms 的延时（IrDA 是一个半双工协议）。

图179 IrDA SIR ENDEC – 框图

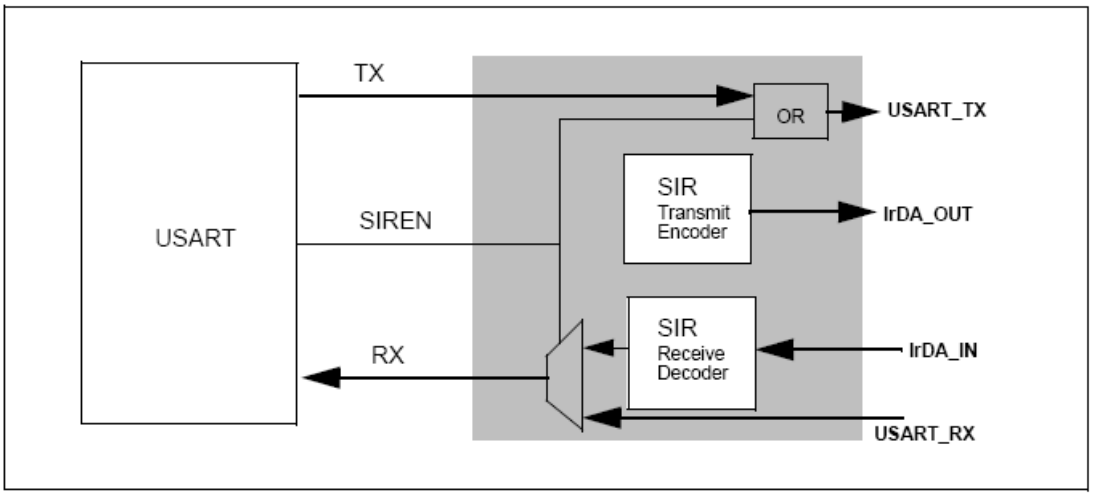
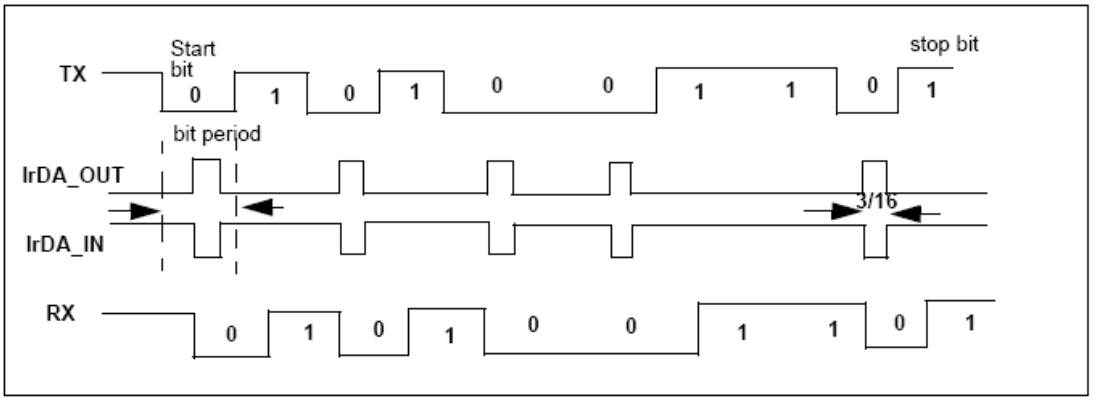


图180 IrDA 数据调制(3/16) – 普通模式



19.3.13 利用DMA连续通信

USART 可以利用 DMA 连续通信。Rx 缓冲器和 Tx 缓冲器的 DMA 请求是分别产生的。

注意： *你应该参考产品技术说明以确定是否可用DMA控制器。如果产品无DMA功能，你应该如 19.3.3 节或 19.3.4 里所描述的方法使用USART。在USART2_SR 寄存器里，你可以清零TXE/RXNE 标志来实现连续通信。*

利用DMA发送

使用 DMA 进行发送，可以通过设置 USART_CR3 寄存器上的 DMAT 位激活。只要 TXE 位被置起，就从配置成使用 DMA 外设的 SRAM 区装载数据到 USART_DR 寄存器。为 USART 的发送分配一个 DMA 通道的步骤如下（x 表示通道号）：

1. 在DMA控制寄存器上将USART_DR寄存器地址配置成DMA传输的目的地址。在每个TXE事件后，数据将被传送到这个地址。
2. 在DMA控制寄存器上将存储器地址配置成DMA传输的源地址。在每个TXE事件后，数据将从此存储器区传送到USART_DR寄存器。
3. 在DMA控制寄存器中配置要传输的总的字节数。
4. 在DMA寄存器上配置通道优先级。
5. 根据应用程序的要求配置在传输完成一半还是全部完成时产生DMA中断。
6. 在DMA寄存器上激活该通道。

当 DMA 控制器中指定的数据量传输完成时，DMA 控制器在该 DMA 通道的中断向量上产生一中断。在中断服务程序里，软件应将 USART_CR3 寄存器的 DMA 位清零。

注意：　如果 DMA 被用于发送，不要使能 TXEIE 位。

利用DMA接收

使用 DMA 进行接收，可以通过设置 USART_CR3 寄存器的 DMAR 位激活。只要接收到一个字节，数据就从 USART_DR 寄存器放到配置成使用 DMA 的 SRAM 区(参考 DMA 技术说明)。为 USART 的接收分配一个 DMA 通道步骤如下（x 表示通道号）：

1. 通过DMA控制寄存器把USART_DR寄存器地址配置成传输的源地址。在每个RXNE事件后此地址上的数据将传输到存储器。
2. 通过DMA控制寄存器把存储器地址配置成传输的目的地址。在每个RXNE事件后，数据将从USART_DR传输到此存储器区。
3. 在DMA控制寄存器中配置要传输的总的字节数。
4. 在DMA寄存器上配置通道优先级。。
5. 根据应用程序的要求配置在传输完成一半还是全部完成时产生DMA中断。
6. 在DMA控制寄存器上激活该通道。

当 DMA 控制器中指定的传输数据量接收完成时，DMA 控制器在该 DMA 通道的中断矢量上产生一中断。在中断程序里， USART_CR3 寄存器的 DMAR 位应该被软件清零。

注意：　如果 DMA 被用来接收，不要使能 RXNEIE 位。

多缓冲器通信中的错误标志和中断产生

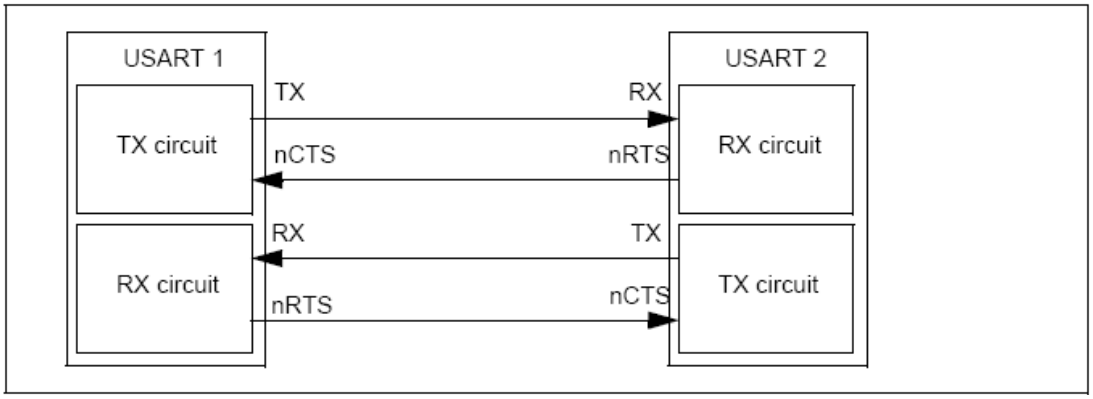
在多缓冲器通信的情况下，通信期间如果发生任何错误，在当前字节传输后将置起错误标志。如果中断使能位被设置，将产生中断。在单个字节接收的情况下，

和 RXNE 一起被置起的帧错误、溢出错误和噪音标志，有单独的错误标志中断使能位；如果设置了，会在当前字节传输结束后，产生中断。

19.3.14 硬件流控制

利用nCTS输入和nRTS输出可以控制 2 个设备间的串行数据流。图 181 表明在这个模式里如何连接 2 个设备。

图181 两个 USART 间的硬件流控制

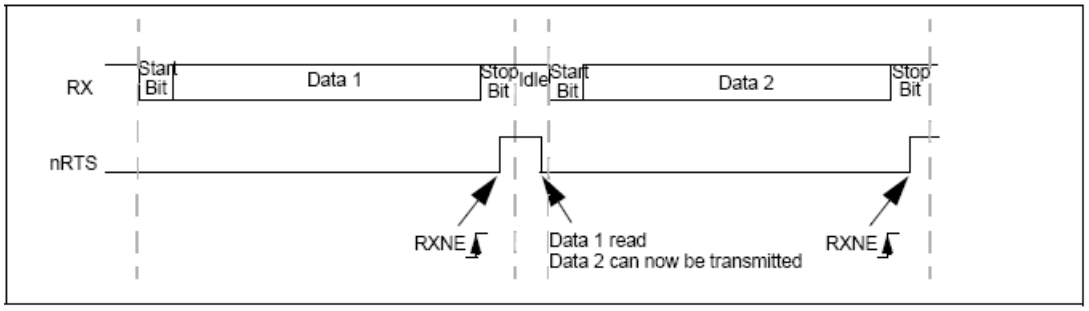


通过将 UASRT_CR3 中的 RTSE 和 CTSE 置位，可以分别独立地使能 RTS 和 CTS 流控制。

RTS流控制

如果RTS流控制被使能(RTSE=1)，只要USART接收器准备好接收新的数据，nRTS就变成有效(接低电平)。当接收寄存器内有数据到达时，nRTS被释放，由此表明希望在当前帧结束时停止数据传输。图 182 展示了一个启用RTS流控制的通信的例子。

图182 RTS 流控制

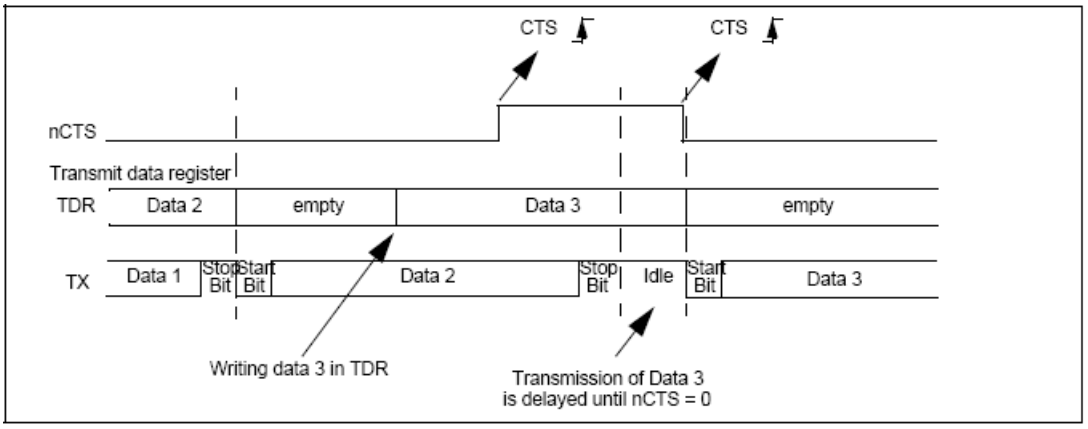


CTS流控制

如果 CTS 流控制被使能(CTSE=1)，发送器在发送下一帧前检查 nCTS 输入。如果 nCTS 有效(被拉成低电平)，则下一个数据被发送(假设那个数据是准备发送的，也就是 TXE=0)，否则下一帧数据不被发出去。若 nCTS 在传输期间被变成无效，当前的传输完成后停止发送。

当 CTSE=1 时，只要 nCTS 输入一变换状态，CTSIF 状态位就自动被硬件设置。它表明接收器是否准备好进行通信。如果 USART_CT3 寄存器的 CTSIE 位被设置，中断产生。下图展示了一个 CTS 流控制被启用的通信的例子。

图183 CTS 流控制



19.4 中断请求

表69 USART 中断请求

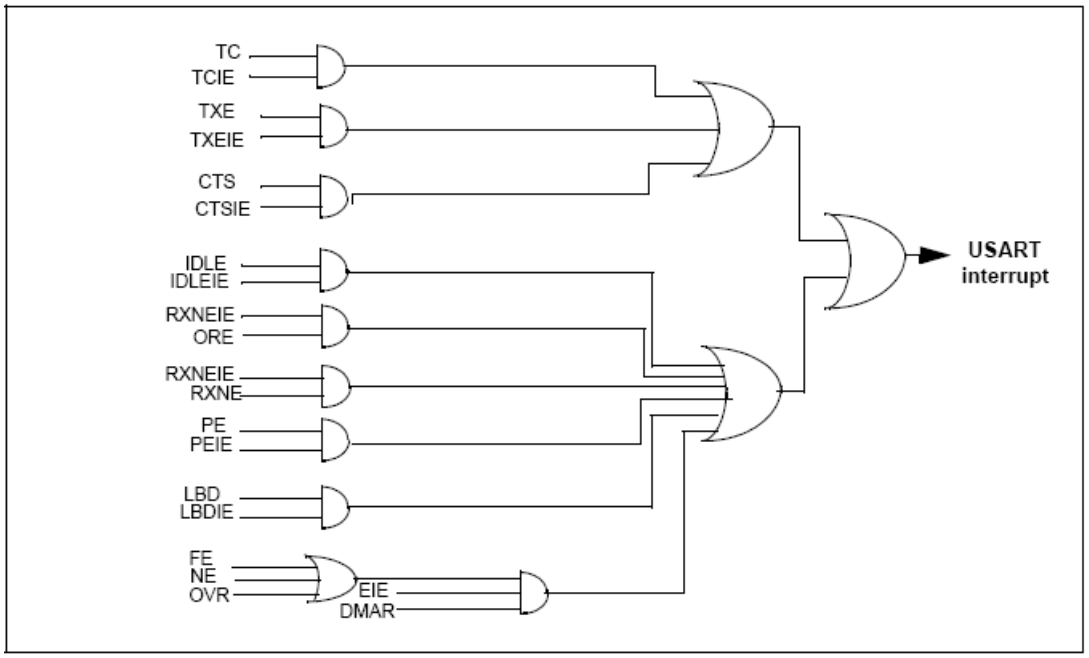
中断事件	事件标志	使能位
发送数据寄存器空	TXE	TXEIE
CTS标志	CTS	CTSIE
发送完成	TC	TCIE
接收数据就绪可读	TXNE	TXNEIE
检测到数据溢出	ORE	
检测到空闲线路	IDLE	IDLEIE
奇偶检验错	PE	PEIE
断开标志	LBD	LBDIE
噪声标志，多缓冲通信中的溢出错误和帧错误	NE或ORT或FE	EIE

USART的各种中断事件被连接到同一个中断向量(见 图 184)，有以下各种中断事件：

- 发送期间：发送完成中断、清除发送中断、发送数据寄存器空中断。
- 接收期间：空闲总线检测中断、溢出错误中断、接收数据寄存器非空中断、校验错误中断、LIN 断开符号检测中断、噪音中断（仅在多缓冲器通信）和帧错误中断（仅在多缓冲器通信）。

如果对应的使能控制位被设置，这些事件就会产生各自的中断。

图184 USART 中断映像图



19.5 USART寄存器描述

参考第 1 章中有关寄存器描述里所使用的缩写。

19.5.1 状态寄存器(USART_SR)

地址偏移：0x00
复位值：0x00C0

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留						CTS	LBD	TXE	TC	RXNE	IDLE	ORE	NE	FE	PE
						rc	rc	r	rc	r	r	r	r	r	r

位31:10	保留位，硬件强制为0
位9	CTS: CTS 标志 如果CTSE位置位，当nCTS输入变化状态时，该位被硬件置高。由软件将其清零。如果USART_CR3中的CTSIE为一，产生中断 0：nCTS状态线上没有变化 1：nCTS状态线上发生变化

位8	LBD: LIN break检测标志（状态标志） 0：没有检测到LIN break 1：检测到LIN break 注意：若LBDIE=1，当LBD为1时要产生中断
位7	TXE:发送数据寄存器空 当TDR寄存器中的数据被硬件转移到移位寄存器的时候，该位被硬件置位。如果USART_CR1寄存器中的TXEIE为1，则产生中断。对USART_DR的写操作，将该位清零。 0：数据还没有被转移到移位寄存器 1：数据已经被转移到移位寄存器 注意：单缓冲器传输中使用该位
位6	TC: 发送完成 当包含有数据的一帧发送完成后，由硬件将该位置位。如果USART_CR1中的TCIE为1，产生中断。由软件序列清除该位（先对USART_SR进行读操作，然后对USART_DR进行写操作） 0：发送还未完成 1：发送完成成
位5	RXNE:读数据寄存器非空 当RDR移位寄存器中的数据被转移到USART_DR寄存器中，该位被硬件置位。如果USART_CR1寄存器中的RXNEIE为1，中断产生。对USART_DR的读操作可以将改位清零。 0：数据没有收到 1：收到数据，可以读出
位4	IDLE:监测到IDLE总线 当检测到空闲总线时，该位被硬件置位。如果USART_CR1中的IDLEIE为1，产生中断。由软件序列清除该位（先读USART_SR，然后读USART_DR） 0：没有检测到空闲总线 1：检测到空闲总线 注意：IDLE位不会再次被置高直到RXNE位被置起(即又检测到一次空闲总线)
位3	ORE:过载错误 当RXNE还是1的时候，当前被接收在移位寄存器中的数据要往RDR寄存器中传送时，硬件将该位置位。如果USART_CR1中的RXNEIE为1的话，产生中断。由软件序列将其清零（先读USART_SR，然后读USART_CR） 0：没有过载错误 1：检测到过载错误 注意：该位被置位时，RDR寄存器中的值不会丢失，但是移位寄存器中的数据会被覆盖。如果EIE位被设置，在多缓冲器通信模式下，ORE标志置位会产生中断的。
位2	NE: 噪声错误标志 在接收到的帧检测到噪音时，由硬件对该位置位。由软件序列对其清玲（先读USART_SR，再读USART_DR） 0：没有检测到噪声 1：检测到噪声 注意：该位不会产生中断，因为它和RXNE一起出现，后者自己会在RXNE标志置位时产生中断，如果EIE位被设置，并且工作在多缓冲区通信模式下

位1	FE: 帧错误 当检测到同步错位，过多的噪声或者检测到break符，该位被硬件置位。由软件序列将其清零（先读USART_SR，再读USART_DR） 0：没有检测到帧错误 1：检测到帧错误或者break符 注意：该位不会产生中断，因为它和RXNE一起出现，后者自己会在RXNE标志置位时产生中断。如果当前传输的数据既产生了帧错误，又产生了过载错误，还是会继续该数据的传输，并且只有ORE位会被置位。 如果EIE位被置位，在多缓冲区通信模式下，随着FE标志被置位，中断产生。
位0	PE: 校验错误 在接收模式下，如果出现校验错误，硬件对该位置位。由软件序列对其清零（依次读USART_SR和USART_DR）。如果USART_CR1中的PEIE为1，产生中断。 0：没有校验错误

19.5.2 数据寄存器(USART_DR)

地址偏移：0x04
复位值：不确定

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

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

位31:9	保留位，硬件强制为0
位8:0	DR[8:0]:数据值 包含了发送或接收的数据。由于它是由两个寄存器组成的，一个给发送用（TDR），一个给接收用（RDR），该寄存器兼具读和写的功能。TDR寄存器提供了内部总线和输出移位寄存器之间的并行接口（参见图1）。RDR寄存器提供了输入移位寄存器和内部总线之间的并行接口。 当使能校验位（USART_CR1种PCE位被置位）进行发送时，写到MSB的值（根据数据的长度不同，MSB是第7位或者第8位）会被后来的校验位该取代。 当使能校验位进行接收时，读到的MSB位是接收到的校验位。

19.5.3 波特比率寄存器(USART_BRR)

注意： 如果 TE 或 RE 被分别禁止，波特计数器停止计数

地址偏移：0x08

复位值：0x0000

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

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	保留位，硬件强制为0
位15:4	DIV_Mantissa[11:0]:USARTDIV的小数部分 这12位定义了USART分频器除法因子(USARTDIV)的小数部分
位3:0	DIV_Fraction[3:0]:USARTDIV的整数部分 这4位定义了USART分频器除法因子(USARTDIV)的整数部分

19.5.4 控制寄存器 1 (USART_CR1)

地址偏移：0x0C

复位值：0x0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留		UE	M	WAKE	PCE	PS	PEIE	TXEIE	TCIE	RXNEIE	IDLEIE	TE	RE	RWU	SBK
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位31:14	保留位，硬件强制为0
位13	UE: USART使能 当该位被清零，USART的分频器和输出在当前字节传输完成后停止工作，以减少功耗。该位的置起和清零，是由软件操作的。 0：USART分频器和输出被禁止 1：USART模块使能
位12	M: 字长 该位定义了数据字的长度，由软件对其置位和清零操作 0：一个起始位，8个数据位，n个停止位 1：一个起始位，9个数据位，一个停止位 注意：在数据传输过程中（发送或者接收时），不能修改这个位

位11	WAKE: 唤醒的方法 这位决定了把USART唤醒的方法，由软件对该位置位或者清零。 0：被空闲总线唤醒 1：被地址标记唤醒
位10	PCE: 检验控制使能 用该位来选择是否进行硬件校验控制（对于发送来说就是校验位的产生；对于接收来说就是校验位的检测）。当使能了该位，在发送数据的MSB（如果M=1，MSB就是第9位；如果M=0,MSB就是第8位）插入校验位；对接收到的数据检查其校验位。软件对它置位或者清零。一旦该位被置位，当前字节传输完成后，校验控制才生效。 0：校验控制被禁止 1：校验控制被使能
位9	PS: 校验选择 该位用来选择当校验控制使能后，是采用偶校验还是奇校验。软件对它置位或者清零。当前字节传输完成后，该选择生效 0：偶校验 1：奇校验
位8	PEIE: PE中断使能 软件对该位置位或者清零 0：中断被禁止 1：当USART_SR中的PE为1时，产生USART中断
位7	TXEIE: 发送缓冲区空中断使能 软件对该位置位或者清零 0：中断被禁止 1：当USART_SR中的TXE为1时，产生USART中断
位6	TCIE: 发送完成中断使能 软件对该位置位或者清零 0：中断被禁止 1：当USART_SR中的TC为1时，产生USART中断
位5	RXNEIE: 接收缓冲区非空中断使能 软件对该位置位或者清零 0：中断被禁止 1：当USART_SR中的ORE或者RXNE为1时，产生USART中断
位4	IDLEIE: IDLE中断使能 软件对该位置位或者清零 0：中断被禁止 1：当USART_SR中的IDLE为1时，产生USART中断
位3	TE: 发送使能 该位使能发送器。软件对该位置位或者清零 0：发送被禁止 1：发送被使能 注意： 在数据传输过程中，除了在智能卡模式下，如果TE位上有个0脉冲（即“0”之后来一个“1”），会在当前数据字传输完成后，发送一个“预备状态”（空闲总线） 当TE被设置后，在真正发送开始之前，有一个比特时间的延迟。

位2	RE: 接收使能 软件对该位置位或者清零 0：接收被禁止 1：接收被使能，开始搜寻RX引脚上的起始位。
位1	RWU: 接收唤醒 该位用来决定是否把USART置于静默模式。软件对该位置位或者清零。当唤醒序列到来时，硬件也会将其清零。 0：接收器处于正常工作模式 1：接收器处于静默模式 注意： 在把USART置于静默模式(设置RWU位)之前，USART要已经先接收了一个数据字节。否则在静默模式下，不能被空闲总线检测唤醒。 当配置成地址标记检测唤醒(WAKE位为1)，在RXNE位被置位时，不能用软件来修改RWU位。
位0	SBK: 发送断开帧 使用该位来发送断开字符。软件可以对该位置位或者清零。应该由软件来置位它，然后在断开帧的停止位时，由硬件将该位复位。 0：没有发送断开字符 1：将要发送断开字符

19.5.5 控制寄存器 2(USART_CR2)

地址偏移：0x10
复位值：0x0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留	LINEN	STOP[1:0]		CLKEN	CPOL	CPHA	LBCL	保留	LBDIE	LBDL	保留	ADD[3:0]			
	rw	rw	rw	rw	rw	rw	rw		rw	rw		rw	rw	rw	rw

位31:15	保留位，硬件强制为0
位14	LINEN: LIN模式使能 软件对该位置位或者清零。 0：LIN模式被禁止 1：LIN模式被使能 LIN模式可以用USART_CR1寄存器中的SBK位发送LIN同步breaks,以及检测LIN同步break
位13:12	STOP: 停止位 用来设置停止位的位数 00：1个停止位 01：0.5个停止位 10：2个停止位 11：1.5个停止位

位11	CLKEN: 时钟使能 该位用来使能SCLK引脚 0：SCLK引脚被禁止 1：SCLK引脚被使能
位10	CPOL: 时钟极性 用户可以用该位来选择同步模式下SLCK引脚上时钟输出的极性。和CPHA位一起配合来产生用户希望的时钟/数据的采样关系 0：总线空闲时SCLK引脚上保持低电平 1：总线空闲时SCLK引脚上保持高电平
位9	CPHA: 时钟相位 用户可以用该位来选择同步模式下SLCK引脚上时钟输出的相位。和CPOL位一起配合来产生用户希望的时钟/数据的采样关系(参见图174和0) 0：时钟第一个边沿进行数据捕获 1：时钟第二个边沿进行数据捕获
位8	LBCL: 最后一位时钟脉冲 使用该位来控制是否在同步模式下，在SCLK引脚上输出最后发送的那个数据字节(MSB)对应的时钟脉冲 0：最后一位数据的时钟脉冲不从SCLK输出 1：最后一位数据的时钟脉冲会从SCLK输出 注意：最后一个数据位就是第8或者第9个发送的位（根据USART_CR1寄存器中的M位所定义的8或者9位数据帧格式）
位7	保留位，硬件强制为0
位6	LBDIE: LIN break检测中断使能 Break中断掩码（使用break定界符来检测break） 0：中断被禁止 1：只要USART_SR寄存器中的LBD为1就产生中断
位5	LBDL: LIN break检测长度 该位用来选择是11位还是10位的break检测 0：10位的break检测 1：11位的break检测
位4	保留位，硬件强制为0
位3:0	ADD[3:0]：该USART节点的地址 该位域给出这个USART节点的地址 这是在多处理器通信下的静默模式中使用的，使用地址标记来唤醒某个USART设备

注意： 在发送被使能后不能写这三个位（CPOL 、CPHA 、LBCL）

19.5.6 控制寄存器 3(USART_CR3)

地址偏移：0x14
复位值：0x0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留					CTSIE	CTSE	RTSE	DMAT	DMAR	SCEN	NACK	HDSEL	IRLP	IREN	EIE
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位31:11	保留位，硬件强制为0
位10	CTSIE: CTS中断使能 0：中断被禁止 1：只要USART_SR寄存器中的CTS为1就产生中断
位9	CTSE: CTS使能 0:CTS硬件流控制被禁止 1:CTS模式使能，只有nCTS输入信号有效（拉成低电平）时才能发送数据。如果在数据传输的过程中，nCTS信号变成无效,那么发完这个数据后，传输就停止下来。如果当nCTS为无效的时候，往数据寄存器里写了数据，那么这个数据要等到nCTS有效的时候才会被发送出去。
位8	RTSE: RTS使能 0：RTS硬件流控制被禁止 1：RTS中断使能，只有接收缓冲区内有空闲的空间时才请求下一个数据。当前数据发送完成后，发送操作就需要暂停下来。如果可以接收数据了，将nRTS输出置为有效（拉至低电平）
位7	DMAT: DMA使能发送 由软件对该位清零或者置位 1：发送时的DMA模式使能 0：发送时的DMA模式被禁止
位6	DMAR: DMA使能接收 由软件对该位清零或者置位 1：接收时的DMA模式使能 0：接收时的DMA模式被禁止
位5	SCEN: 智能卡模式使能 该位用来使能智能卡模式 0：智能卡模式使能 1：智能卡模式被禁止
位4	NACK:智能卡NACK使能 0：校验错误出现时，不发送NACK 1：校验错误出现时，发送NACK

位3	HDSEL：半双工选择 选择单线半双工模式 0：不选择半双工模式 1：选择半双工模式
位2	IRLP：红外低功耗 该位用来选择普通模式还是低功耗红外模式 0：通常模式 1：低功耗模式
位1	IREN：红外模式使能 由软件对该位清零或者置位 0：红外被禁止 1：红外使能
位0	EIE：错误中断使能 在多缓冲区通信模式下，当有帧错误、过载或者噪声错误时(USART_SR中德FE=1，或者ORE=1，或者NE=1)，产生中断。 0：中断被禁止 1：只要USART_CR3中的DMAR=1，并且USART_SR中的FE=1，或者ORE=1,或者NE=1，产生中断

19.5.7 保护时间和预分频寄存器(USART_GTPR)

地址偏移：0x18
复位值：0x0000

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

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	保留位，硬件强制为0
位15:8	GT[7:0]：保护时间值 该位域规定了以波特时钟为单位的保护时间的值。在智能卡模式下，需要这个功能。当保护时间过去后，发送完成标志才被置起。

位7:0	PSC[7:0]:预分频器值 **- 在红外低功耗模式下：** PSC[7:0]=红外低功耗波特率 对系统时钟分频已到达低功耗的频率： 源时钟被寄存器中的值(仅有8位有效)分频 00000000：保留 – 不要写入该值 00000001：对源时钟1分频 00000010：对源时钟2分频 …… **- 在红外的通常模式下**：PSC只能设置为0000001 **- 在智能卡模式下：** PSC[4:0]:预分频值 对系统时钟进行分频，给智能卡提供时钟。 寄存器中给出的值（5个有效位）乘以2后，作为对源时钟的分频因子 00000：保留 – 不要写入该值 00001：对源时钟进行2分频 00010：对源时钟进行4分频 00011：对源时钟进行6分频 …… 注意：位[7:5]在智能卡模式下没有意义

19.6USART寄存器地址映象

表70 USART 寄存器列表及其复位值

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
000h	USART_SR	保留																						CTS	LBD	TXEIE	TC	RXNE	IDLE	ORE	NE	FE	PE
	复位值																							0	0	1	1	0	0	0	0	0	0
004h	USART_DR	保留																						DR[8:0]									
	复位值																							0	0	0	0	0	0	0	0	0	0
008h	USART_BRR	保留																DIV_Mantissa[15:4]												DIV_Fraction[3:0]			
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
00Ch	USART_CR1	保留																		UE	M	WAKE	PCE	PS	PEIE	TXEIE	TCIE	RXNEIE	IDLEIE	TE	RE	RWU	SBK
	复位值																			0	0	0	0	0	0	0	0	0	0	0	0	0	0
010h	USART_CR2	保留																	LIEN	STOP[1:0]		CLKEN	CPOL	CPHA	LBCL	保留	LBDIE	LBDL	保留	ADD[3:0]			
	复位值																		0	0	0	0	0	0	0		0	0		0	0	0	0
014h	USART_CR3	保留																					CTSIE	CTSE	RTSE	DMAT	DMAR	SCEN	NACK	HDSEL	IRLP	IREN	EIE
	复位值																						0	0	0	0	0	0	0	0	0	0	0
018h	USART_GTPR	保留																GT[7:0]										PSC[7:0]					
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

关于寄存器的起始地址，请参见第 1 章。

20 调试支持(DBG)

20.1 概况

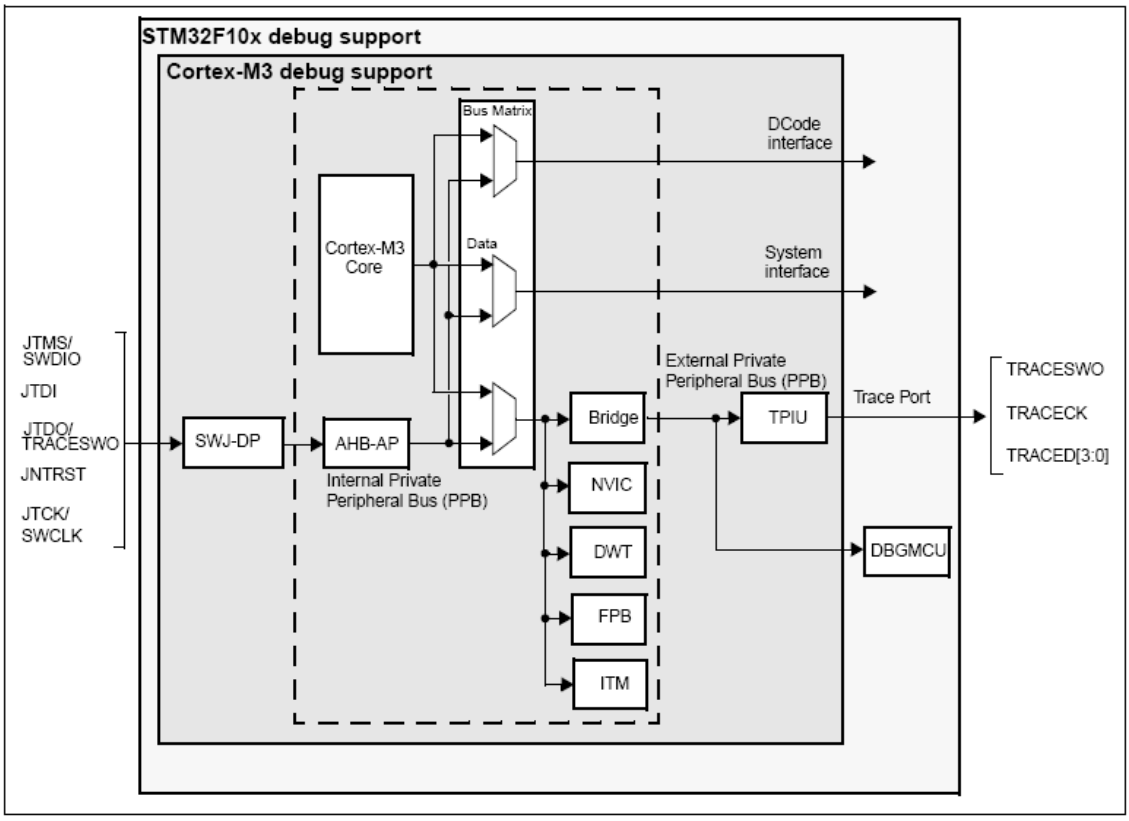
STM32F10xxx 使用 Cortex™-M3 内核，该内核内含硬件调试模块，支持复杂的调试操作。硬件调试模块允许内核在取指（指令断点）或访问数据（数据断点）时停止。内核停止时，内核的内部状态和系统的外部状态都是可以查询的。完成查询后，内核和外设可以被复原，程序将继续执行。

当 STM32F10x 微控制器连接到调试器并开始调试时，调试器将使用内核的硬件调试模块进行调试操作。

支持两种调试接口：

- 串行接口
- JTAG 调试接口

图185 STM32F10xxx 级别和 Cortex™-M3 级别的调试框图



注意： Cortex™-M3 内核内含的硬件调试模块是 ARM CoreSight 开发工具集的子集。

ARM Cortex™-M3 内核提供集成的片上调试功能。它由以下部分组成：

- SWJ-DP：串行/JTAG 调试端口

- AHP-AP：AHB 访问端口
- ITM：执行跟踪单元
- FPB：Flash 指令断点
- DWT：数据触发
- TPUI：跟踪单元接口（仅较大封装的芯片支持）

专用于 STM32F10xxx 的调试特性

- 灵活的调试引脚分配
- MCU 调试盒（支持低电源模式，控制外设时钟等）

注意：　　更多 ARM Cortex™-M3 内核的调试功能信息，请参考 Cortex™-M3（r1p1 版）技术参考手册（TRM）和 CoreSight 开发工具集（r1p0 版）TRM。

20.2 ARM参考文献

- Cortex™-M3（r1p1 版）技术参考手册（TRM）
- ARM 调试接口 V5
- ARM CoreSight 开发工具集（r1p0 版）技术参考手册

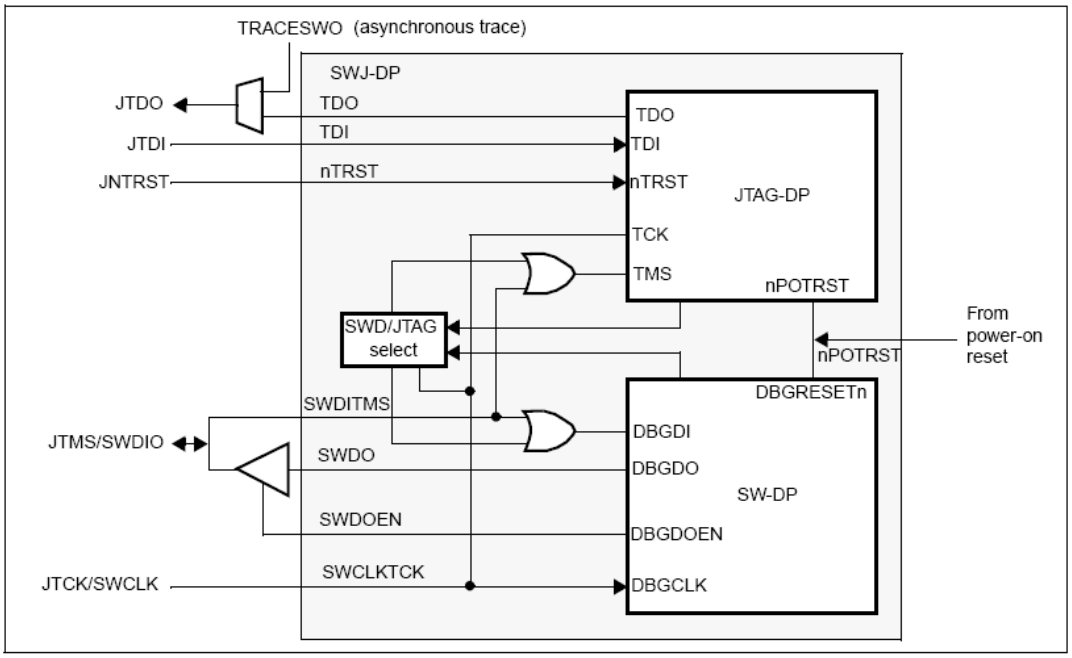
20.3 SWJ调试端口(serial wire and JTAG)

STM32F10xxx 内核集成了串行/JTAG 调试接口（SWJ-DP）。这是标准的 ARM CoreSight 调试接口，包括 JTAG-DP 接口（5 个引脚）和 SW-DP 接口（2 个引脚）。

- JTAG 调试接口(JTAG-DP)为 AHP-AP 模块提供 5 针标准 JTAG 接口。
- 串行调试接口(SW-DP)为 AHP-AP 模块提供 2 针（时钟＋数据）接口。

在 SWJ-DP 接口中，SW-DP 接口的 2 个引脚和 JTAG 接口的 5 个引脚中的一些是复用的。

图186 SWJ 调试端口



上面的图显示异步跟踪输出脚（TRACESWO）和 TDO 是复用的，因此异步跟踪功能只能在 SW-DP 调试接口上实现，不能在 JTAG-DP 调试接口上实现。

20.3.1 JTAG-DP和SW-DP切换的机制

JTAG 调试接口是默认的调试接口。

如果调试器想要切换到 SW-DP，必须在 TMS/TCK 上输出一指定的 JTAG 序列（分别映射到 SWDIO 和 SWCLK），该序列禁止 JTAG-DP，并激活 SW-DP。该方法可以只通过 SWCLK 和 SWDIO 两个引脚来激活 SW-DP 接口。

指定的序列是：

1. 输出超过50个TCK周期的TMS（SWDIO）＝ 1信号
2. 输出16个TMS（SWDIO）信号 0111100111100111 （MSB）
3. 输出超过50个TCK周期的TMS（SWDIO）＝ 1信号

20.4引脚分布和调试端口脚

STM32F10xxx 微控制器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。

20.4.1 SWJ调试端口脚

STM32F10xxx 的 5 个普通 I/O 口可用作 SWJ-DP 接口引脚。这些引脚在所有的封装里都存在。

表71 SWJ 调试端口管脚

SWJ-DP 端口引脚名称	JTAG 调试接口		SW 调试接口		引脚分配
	类型	描述	类型	调试功能	
JTMS/SWDIO	输入	JTAG模式选择	输入/输出	串行数据输入/输出	PA13
JTCK/SWCLK	输入	JTAG时钟	输入	串行时钟	PA14
JTDI	输入	JTAG数据输入	——	——	PA15
JTDO/TRACESWO	输出	JTAG数据输出	——	跟踪时为TRACESWO信号	PB3
JNTRST	输入	JTAG模块复位	——	——	PB4

20.4.2 灵活的SWJ-DP脚分配

复位（SYSRESETn 或 PORESETn）以后，属于 SWJ-DP 的所有 5 个引脚都立即被初始化为可被调试器使用的专用引脚（注意，并没有初始化跟踪输出脚，除非调试器对此脚进行定义）。

然而，STM32F10xxx 微控制器可以用 REMAP_DBGAFR 寄存器来禁止 SWJ-DP 接口的部分或所有引脚的功能，这些专用引脚将被释放以用作普通 I/O 口。此寄存器被映射到和 Cortex™-M3 系统总线相连接的 APB 桥上。对此寄存器的设置将由用户代码而不是调试器完成。

3 个控制位用来配置 SWJ-DP 接口的引脚，这 3 个位在系统复位时复位。

- REMAP_AF_REG（STM32F10xxx 微控制器中的地址是 0x40010004）
  - 读：APB，无等待状态
  - 写：APB，如果 AHB-APB 桥的写缓冲器满了，则一个等待状态

位 26：24=SWJ_CFG[2:0]
由软件置位和复位
这 3 位用来设置分配给 SWJ 调试接口的专用引脚数目，目的是在使用不同的调试接口时能释放尽可能多的引脚用作普通 I/O 口。复位后的初始值是 000(所有引脚都设置为 JTAG-DP 接口专用引脚)，同时只能置位 3 个位中的一个(同时置位一个以上的位是被禁止的)。

表72 灵活的 SWJ_DP 管脚分配

SWJ-CFG[2:0]	配置为调试专用的引脚	SWJ 接口的 I/O 口分配				
		PA13/ JTMS/ SWDIO	PA14/ JTCK/ SWCLK	PA15/ JTDI	PB3/ JTDO	PB4/ JNTRST
000	所有的SWJ引脚(JTAG-DP + SW-DP)（默认状态）	专用	专用	专用	专用	专用
001	所有的SWJ引脚(JTAG-DP + SW-DP)除了JNTRST引脚	专用	专用	专用	专用	释放
010	JTAG-DP接口禁止，SW-DP接口允许	专用	专用	释放	释放	释放
100	JTAG-DP接口和SW-DP接口都禁止	释放				
其他	禁止					

注意：　当 APB 桥的写缓冲区满了时，在写 REMAP_AF 寄存器时需要多用一个 APB 周期。这是因为 JTAGSW 脚的释放需要 2 个 APB 周期，以保证输入内核的 nTRST 和 TCK 信号的平稳。

- *周期 1：输入 1 / 0 的 JTAGSW 信号到内核（nTRST, TDI 和 TMS 为 1，TCK 为 0）。*
- *周期 2：GPI/O 控制器获得 SWJTAG I/O 引脚的控制信号（如对方向，上拉/下拉，施密特触发等的控制）。*

20.4.3 JTAG脚上的内部上拉和下拉

保证 JTAG 的输入引脚不是悬空的是非常必要的，因为他们直接连接到 D 触发器控制着调试模式。必须特别注意 SWCLK/TCK 引脚，因为他们直接连接到一些 D 触发器的时钟端。

为了避免任何未受控制的 I/O 电平，STM32F10xxx 在 JTAG 输入脚上嵌入了内部上拉和下拉。

- JINTRST：内部上拉
- JTDI：内部上拉
- JTMS/SWDIO:内部上拉
- TCK/SWCLK:内部下拉

一旦 JTAG I/O 被用户代码释放，GPIO 控制器再次取得控制。这些 I/O 口的状态将恢复到复位时的状态。

- JNTRST：带上拉的输入
- JTDI：带上拉的输入
- JTMS/SWDIO：带上拉的输入
- JICK/SWCLK：带下拉的输入
- JTDO：浮动输入

软件可以把这些 I/O 口作为普通的 I/O 口使用。

注意：　JTAG IEEE 标准建议对 TDI，TMS 和 nTRST 上拉，而对 TCK 没有特别的建议。但在 STM32F10xxx 中，JTCK 引脚带有下拉。
内嵌的上拉和下拉使芯片不再需要外加外部电阻。

20.4.4 利用串行接口并释放不用的调试脚作为普通 I/O口

为了利用串行调试接口来释放一些普通 I/O 口，用户软件必须在复位后设置 SWJ_CFG=010，从而释放 PA15，PB3 和 PB4 用做普通 I/O 口。

在调试时，调试器进行以下操作：

- 在系统复位时，所有 SWJ 引脚被分配为专用引脚(JTAG-DP + SW-DP)。
- 在系统复位状态下，调试器发送指定 JTAG 序列，从 JTAG-DP 切换到 SW-DP。
- 仍然在系统复位状态下，调试器在复位地址处设置断点
- 释放复位信号，内核停止在复位地址处。
- 从这里开始，所有的调试通信将使用 SW-DP 接口，其他 JTAG 引脚可以由用户代码改配为普通 I/O 口。

注意：　对于用户软件设计，应注意：
在复位后，这些专用引脚仍然处于带上拉的输入（nTRST, TMS, TDI），带下拉的输入（TCK），和输出（TDO）状态，并持续一段时间，直到用户代码释放这些引脚。
当这些引脚被配置成专用引脚时（JTAG 或者 SW 或者 TRACE），修改相应的普通 I/O 口配置寄存器是无效的。

20.5 STM32F10xxx JTAG TAP 连接

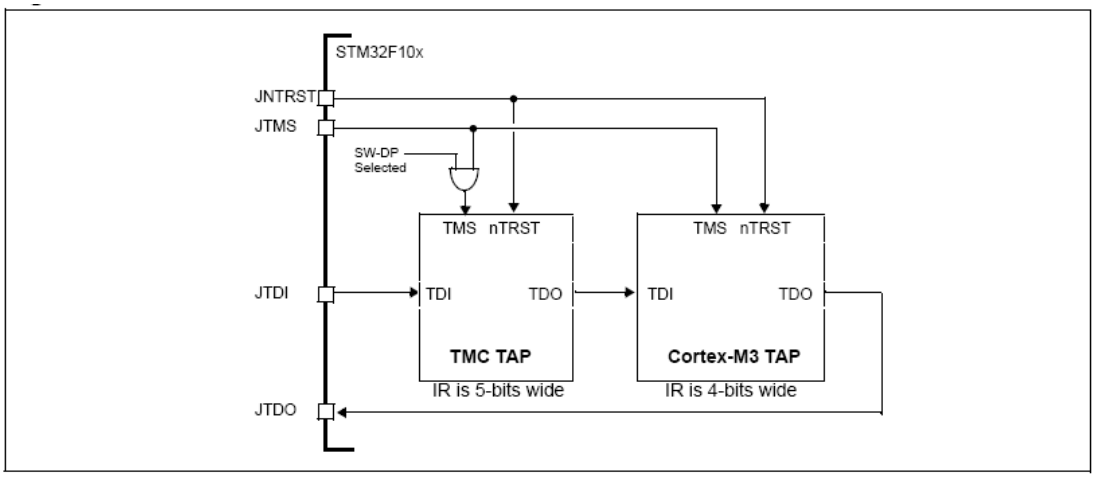
STM32F10xxx 微控制器内部串联了两个 JTAG TAP。TMC TAP 专门用来进行测试（IR 寄存器为 5 比特位宽）和 Cortex™-M3 TAP（IR 寄存器为有 4 比特位宽）。

为了访问 Cortex™-M3 TAP 对芯片进行调试，必须：

1. 首先，必须将BYPASS指令移位输入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：一旦使用了指定的 JTAG 序列选择了串行调试接口，TMC TAP 自动被禁止（JTMS 被强制为高）。

图187 JTAG TAP 连接



20.6 ID 代码和锁定机制

在 STM32F10x 微控制器内部有多个 ID 编码。ST 强烈建议工具设计者使用映射在外部 PPB 存储器上地址为 0xE0042000 的 MCU DEVICE ID 来锁定调试器。

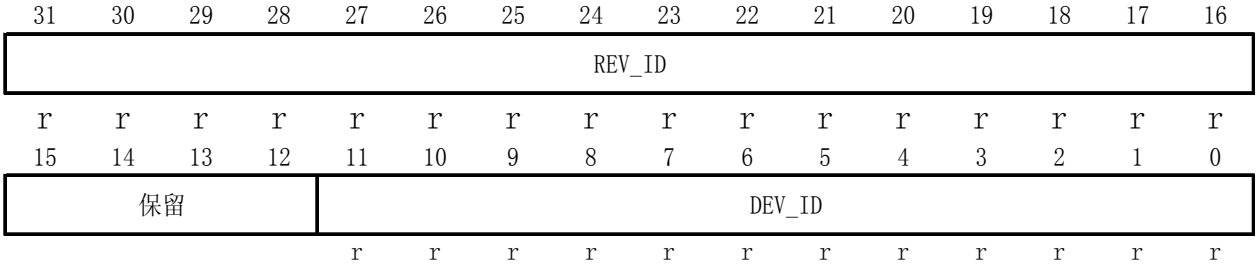
20.6.1 微控制器设备ID编码

微控制器 STM32F10xxx内含一个MCU ID编码。这个ID定义了ST MCU的部件号和硅片版本。它是DBG_MCU的一个组成部分，并且映射到外部PPB总线上(见 20.15 节)。使用JTAG调试口（4－5 个引脚）或SW调试口（2 个引脚）或通过用

户代码都可以访问此编码。即使当MCU处于系统复位状态下这个编码也可以被访问。

DBGMCU_IDCODE

地址：0xE004 2000
只支持 32 位访问
只读=0xXXXXX410，其中 X 为不确定



位31:16	REV_ID[15:0]：版本识别 该域标识产品的版本 0x0000 = 版本A 0x2000 = 版本B 0x2001 = 版本Z
位15:12	保留
位11:0	DEV_ID[11:0]：设备识别 这个部分指示了设备编码。对于STM32F10x微控制器，设备编码为0x410

20.6.2 TMC TAP

JTAG ID 编码

STM32F10xxx TMC（测试模式控制器）的 TAP 有一个等于 **0x1641 0041** 的 JTAG ID 编码。

20.6.3 Cortex-M3 TAP

ARM Cortex-M3 的 TAP 有一个 JTAG ID 编码。这个 ID 编码是 ARM 默认的，且没有被修改过，只能通过 JTAG 调试口访问。此编码是 **0x3BA00477**（对应于 Cortex-M3 r1p1）。

调试器/编程工具应该只通过 DEV_ID(15:0)来识别芯片。

20.6.4 Cortex-M3 JEDEC-106 ID代码

ARM Cortex-M3 有一个 JEDEC-106 ID 编码。它位于映射到内部 PPB 总线地址为 0xE00FF000_0xE00FFFFF 的 4KB ROM 表中。

20.7 JTAG调试端口

标准的 JTAG 状态机是通过一个 4 比特位的指令寄存器（IR）和 5 个数据寄存器（详细请见 *Cortex-M3 r1p1 Technical Reference Manual*）实现的。

表73 JTAG 调试模块数据寄存器

IR(3:0)	数据寄存器	描述
1111	BYPASS[1比特位]	
1110	IDCODE [32比特位]	**ID编码寄存器** 0x3BA00477（ARM Cortex-M3 r1p1 ID 编码
1010	DPACC [35比特位]	调试接口寄存器 初始化调试端口，并允许访问调试接口寄存器 - 输入数据时： Bits34:3=DATA[31:0]：对应写操作的32位数据位 Bits2:1=A[3:2]：调试接口寄存器的2位地址值 Bit0=RnW：读操作（1）或写操作（0） - 输出数据时： Bits34:3=DATA[31:0]：前一次读操作的32位数据结果 Bits2:0=ACK[2:0]：3比特位的应答 010＝成功/失败 001＝等待 其他＝未定义 A(3:2)的定义请参考表74
1011	APACC [35比特位]	存取接口寄存器 初始化存取接口并允许访问存取接口寄存器 - 输入数据时： Bits34:3=DATA[31:0]：对应写操作的32位数据位 Bits2:1=A[3:2]：2比特位地址（AP寄存器的部分地址） Bit0=RnW：读操作（1）或写操作（0） - 输出数据时： Bits34:3=DATA[31:0]：前一次读操作的32位数据结果 Bits2:0=ACK[2:0]：3比特位的应答 010＝成功/失败 001＝等待 其他＝未定义 关于AP寄存器请参考AHB-AP章节，这些寄存器的地址由以下部分组成：A[3:2] - 移位值A[3:2] - DP SELECT寄存器的当前值
1000	ABORT [35比特位]	中止寄存器 - Bits 31:1 未定义 - Bit 0＝DAPABORT：写1产生一个DAP中止

表74 由 A[3:2]定义的 32 位调试接口寄存器地址

地址	A(3:2)值	描述
0x0	00	未定义
0x4	01	DP CTRL/STAT 寄存器 - 请求一个系统或调试的上电操作 - 配置AP访问的操作模式 - 控制比较，校验操作 - 读取一些状态位（溢出，上电响应）
0x8	10	DP SELECT寄存器 用来选择当前的访问端口和有效的4字长寄存器窗口 - Bits31:24：APSEL 选择当前AP - Bits23:8：未定义 - Bits7:4：APBANKSEL：在当前AP上选择4字长寄存器窗口 - Bits3:0：未定义
0xC	11	DP RDBUFF寄存器：用来使调试器获得前一次操作的最终结果（不用再请求一个新的JTAG-DP操作）

20.8 SW调试端口

20.8.1 SW协议介绍

此同步串行协议使用 2 个引脚：

- SWCLK：从主机到目标的时钟信号
- SWDIO：双向数据信号

协议允许读写 2 个寄存器组（DPACC 和 APACC 寄存器组）。

数据位按 LSB 传输。

由于 SWDIO 为双向口，该引脚需有上拉（ARM 建议使用 100K 电阻）。

按协议每次 SWDIO 方向改变时，需插入一个转换时间。在该期间内主机和目标都不驱动此信号线。转换时间的默认值是 1 个比特，但可以通过配置 SWCLK 频率来调节。

20.8.2 SW协议序列

每个序列由 3 个阶段组成：

1. 主机发送包请求（8位）
2. 目标发送确认响应（3位）

3. 主机或目标发送数据（33位）

表75 请求包（8 比特位）

比特位	名称	描述
0	起始	必须为1
1	APnDP	0：访问DP 1：访问AP
2	RnW	0：写请求 1：读请求
4:3	A(3:2)	DP或AP寄存器的地址（请参考表74）
5	Parity	前面比特位的校验位
6	Stop	0
7	Park	不能由主机驱动，由于有上拉，目标永远读为1

请参考 *Cortex-M3 r1p1 TRM*，以得到有关 DPACC 和 APACC 寄存器描述的详细资料。

包请求后总是跟一个(缺省为 1 位)转换时间，此时主机和目标都不驱动线路。

表76 ACK 定义（3 比特位）

比特位	名称	描述
0..2	ACK	001：失败 010：等待 100：成功

当 ACK 为失败或等待，或者是一个回复读操作的 ACK，此 ACK 后有一个转换时间。

表77 传输数据（33 比特位）

比特位	名称	描述
0..31	WDATA/ RDATA	写或读的数据
32	Parity	32位数据的奇偶校验位

读操作的数据传输操作后有一个转换时间。

20.8.3 SW-DP状态机(Reset, idle states, ID code)

SW-DP 状态机有一个内部 ID 编码用来识别 SW-DP，它遵守 JEP-106 标准。此 ID 编码是 ARM 默认的编码，值为 **0x1BA01477**（对应于 Cortex-M3 r1p1）。

注意： 在调试器读这个 ID 编码之前，SW-DP 的状态机是不工作的。

- SW-DP 状态机将处于 RESET 状态，在上电复位后，或 DP 从 JTAG 切换到 SWD 后，或有超过 50 个周期的高电平。
- 当状态机处于 RESET 状态时，如果有至少 2 个周期的低电平，状态机将切换到 IDLE 状态。
- 当状态机处于 RESET 状态后，必需首先进入 IDLE 状态，并执行一个读 DP-SW ID 寄存器的操作。否则，调试器在执行其他传输时，只能获得一个失败的 ACK 响应。

更详细的 SW-DP 状态机资料请参考 Cortex-M3 r1p1 TRM 和 CoreSight Design Kit r1p0 TRM。

20.8.4 DP和AP读/写访问

- 对 DP 的读操作没有传递性：调试器将直接获得数据(如果 ACK＝成功)，或者等待(如果 ACK=等待)。
- 对 AP 的读操作具有传递性。这意味着前一次读操作的结果只能在下一次操作时获得。如果下一次的操作不是对 AP 的访问，则必需读 DP-RDBUFF 寄存器来获得上一次读操作的结果。
- DP-CTRL/STAT 寄存器的 READOK 标志位会在每次 AP 读操作和 RDBUFF 读操作后更新，以通知调试器 AP 的读操作是否成功。
- SW-DP 具有写缓冲区(DP 和 AP 都有写缓冲)，这使得其他传输在进行时，仍然可以接受写操作。如果写缓冲区满，调试器将获得一个等待的 ACK 响应。读 IDCODE 寄存器，读 CTRL/STAT 寄存器和写 ABORT 寄存器操作在写缓冲区满时仍被接受。
- 由于 SWCLK 和 HCLK 的异步性，需要在写操作后(在奇偶校验位后)插入 2 个额外的 SWCLK 周期，以确保内部写操作正确完成。这两个额外的时钟周期需要在线路为低时插入(IDLE 状态下)。

这个操作步骤在写 CTRL/STAT 寄存器以提出一个上电请求时尤其重要，否则下一个操作(在内核上电后才有效的操作)会立即执行，这将导致失败。

20.8.5 SW-DP寄存器

当 APnDP=0 时，可以访问以下这些寄存器。

表78 SW-DP 寄存器

A(3:2)	读/写	SELECT 寄存器的 CTRLSEL 位	寄存器	描述
00	读		IDCODE	固定为0x1BA01477（用于识别SW-DP）
00	写		ABORT	
01	读/写	0	DP-CTRL/STAT	－请求一个系统或调试的上电操作 －配置AP访问的操作模式 －控制比较，校验操作 －读取一些状态位（溢出，上电响应）
01	读/写	1	WIRE CONTROL	配置串行通信物理层协议（如转换时间长度等）
10	读		READ RESEND	允许从一个错误的调试传输中恢复数据而不用重复最初的AP传输

10	写		SELECT	选择当前的访问端口和有效的4字长寄存器窗口
11	读/写		READ BUFFER	由于AP的访问具有传递性（当前AP读操作的结果会在下次AP传输时传出），因此这个寄存器非常必要。这个寄存器会从AP捕获上一次读操作的数据结果，因此可以获得数据而不必再启动一个新的AP传输。

20.8.6 SW-AP寄存器

当 APnDP=1 时，可以访问以下这些寄存器。

AP 寄存器的访问地址由以下两部分组成：

- A[3:2]的值
- DP SELECT 寄存器的当前值

20.9 对于JTAG-DP或SWDP都有效的 AHB-AP (AHB 访问端口)

特性：

- 系统访问是独立于处理器状态的。
- JTAG-DP 和 SW-DP 都可以访问 AHB-AP
- AHB-AP 是总线矩阵的 AHB 主设备。因此，它可以访问所有的数据总线（Dcode 总线， System 总线，内部和外部 PPB 总线），只有 ICode 总线除外。
- 支持位寻址的传输
- 旁路 FPB 的 AHB-AP 传输

32 位 AHP-AP 寄存器的地址是 6-位宽（最多 64 个字或 256 个字节），由以下部分组成：

a）比特位[8:4]= DP SELECT 寄存器的位[7:4] APBANKSEL

b）比特位[3:2] = 35 位 SW-DP 包请求中的 A(3:2)。

Cortex-M3 的 AHB-AP 有 9 个 32 位的寄存器

表79 Cortex-M3 AHB-AP 寄存器

地址偏移	寄存器名	描述
0x00	AHB-AP Control and Status Word	配置AHB接口的传输特性（长度，地址自加模式，当前传输状态，特权模式等）
0x04	AHB-AP Transfer Address	

0x0C	AHB-AP Data Read/Write	
0x10	AHB-AP Banked Data 0	直接访问4个相连的字而不用重写访问地址
0x14	AHB-AP Banked Data 1	
0x18	AHB-AP Banked Data 2	
0x1C	AHB-AP Banked Data 3	
0xF8	AHB-AP Debug ROM Address	调试接口的基地址
0xFC	AHB-AP ID Register	

更多信息请参考 Cortex-M3 r1p1 TRM

20.10 内核调试

通过操作内核调试寄存器可以实行对内核的调试。对这些寄存器的访问通过先进高性能总线（AHB-AP）进行。处理器可以通过内部私有外设总线（PPB）直接访问这些寄存器。

它包括 4 个寄存器

表80 内核调试寄存器

寄存器	描述
DHCSR	32位的调试控制和状态寄存器 此寄存器提供内核状态信息，允许内核进入调试模式，和提供单步功能
DCRSR	17位的内核寄存器调试选择寄存器 此寄存器选择需要进行读写操作的内核寄存器
DCRDR	32位的内核寄存器调试数据寄存器 此寄存器存放由DCRSR选择的内核寄存器读出的或需要写入的数据
DEMCR	32位异常调试和监视控制寄存器 此寄存器提供向量传输和监视调试控制功能。TRCENA位启动TRACE功能

注意： ***重要****：这些寄存器在系统复位时不复位，仅在上电复位时复位。*

更多详细资料请参考 Cortex-M3 r1p1 TRM。

为了在复位后立即使内核进入调试状态，需要：

- 使能调试和异常监视控制寄存器（Debug and Exception Monitor Control Register）的比特位 0（VC_CORRESET）。
- 使能调试控制和状态寄存器（Debug Halting Control and Status Register）的比特位 0 （C_DEBUGEN）。

20.11 调试器主机在系统复位下的连接能力

STM32F10xxx 微控制器的复位系统由下列复位源组成：

- POR（上电复位），在每次上电时发起一次复位
- 内部看门狗复位
- 软件复位
- 外部复位

Cortex-M3 将调试部分的复位（通常是 PORRESETn）和其他复位（SYSRESETn）区分开。因此，当内核处于系统复位状态时，调试器可以连接到内核，配置内核调试寄存器，使能调试允许位，这样操作使内核在系统复位被释放时立即进入调试状态而不执行任何指令。同样的，可以在内核处于复位状态下时配置调试特性。

注意：　　强烈建议调试器在系统复位时连接内核（在复位向量处设置断点）。

20.12 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。此特性可以用来纠正代码区域内的软件错误。

软件补丁功能和硬件断点功能不能同时使用。

FPB 由以下部分组成：

- 2 个内容比较器，用来比较代码区域取得的内容并重映射到系统区域的相关地址。
- 6 个指令比较器，用来比较代码区域的指令。这些比较器可用来实现软件补丁或者硬件断点功能。

20.13 DWT（data watchpoint trigger）

DWT 模块由四个比较器组成，它们分别是：

- 一个硬件数据比较器
- 一个 ETM 触发器
- 一个 PC 值取样器
- 一个数据地址取样器

DWT 还可用来获取某些侧面的信息。通过一些计数器可以获得以下数据：

- 时钟周期
- 分支指令
- 存取单元操作
- 睡眠周期
- CPI（每条指令的执行时间）
- 中断开销

20.14 ITM (instrumentation trace macrocell)

20.14.1 概述

ITM 是一应用驱动的跟踪源，它支持 *printf* 类的调试手段来跟踪操作系统（OS）和应用事件，并发布判定的系统信息。ITM 以包的形式发布跟踪信息，它由以下部分组成：

- 软件跟踪：软件可以通过直接写 ITM 激发寄存器来发布包信息。
- 硬件跟踪：ITM 会发布由 DWT 产生的信息包。
- 时间戳：时间戳被发布到相应的包上。ITM 包含一个 21 位的计数器以产生时间戳。Cortex-M3 的时钟或串行线观测器（*Serial Wire Viewer*）的位时钟率给计数器提供时钟。

由 ITM 发送的信息包输出到 TPIU （Trace Port Interface Unit），TPIU 再添加一些额外的包（参考 TPIU），然后输出完整的包序列给调试器。

用户在设置或使用 ITM 之前，必需先使能异常调试和监视控制寄存器（Debug Exception and Monitor Control Register）的 TRCEN 位。

20.14.2 时间戳包，同步和溢出包

时间戳包编码时间戳信息，普通的控制和同步信息。它使用一个 21 位的时间戳计数器（及可能的预分频器），此计数器在每个时间戳包发放时复位。计数器的时钟可以是 CPU 时钟也可以是 SWV 时钟。

同步包为 0x80_00_00_00_00_00，按 00 00 00 00 00 80 发送给 TPIU（LSB 在前）。

同步包是时间戳包的控制信号。

它也在每个 DWT 触发时发送，因此 DWT 必须配置为触发 ITM：DWT 控制寄存器（DWT Control Register）的比特位 0（CYCCNTENA）必须被置起。此外，ITM 跟踪控制寄存器（Trace Control Register）的比特位 2（SYNCENA）也必须被置起。

注意：　如果 SYNENA 位没有被置起，DWT 产生给 TPIU 的同步触发，将只发送 TPIU 同步包而不发送 ITM 同步包。

溢出包是一个特殊的时间戳包，该包指示数据已经被写但是 FIFO 已满。

表81 主要的 ITM 寄存器

地址	寄存器	描述
@E0000FB0	ITM Lock Access	写入0xC5ACCE55允许写其他ITM寄存器
@E0000E80	ITM Trace Control	Bits 31-24 = 总是0 Bits 23 = 忙 Bits 22-16 = 7位的ATB ID用以识别跟踪数据源 Bits 15-10 = 总是0 Bits 9:8 = 时间戳的预分频 Bits7-5 = 未定义 Bit4 = 使能SWV功能即时间戳计数器使用SWV时钟 Bit3 = 使能DWT的激发功能 Bit2 = 此位必需设为1来使能DWT的产生同步触发功能，以使TPIU能够发送同步包 Bit1 = 时间戳使能 Bit0 = ITM的全局使能位
@E0000E40	ITM Trace Privilege	Bit3:置1使能跟踪端口31:24 Bit2:置1使能跟踪端口23:16 Bit1:置1使能跟踪端口15:8 Bit0:置1使能跟踪端口7:0
@E0000E00	ITM Trace Enable	每个比特位使能相应的触发端口产生跟踪
@E0000000－E000007C	Stimulus Port Registers 0-31	向选中的产生跟踪的触发端口（32个）写32位数据

关于配置的例子：

向 TUIU 输出一个简单值：

- 配置TPIU并使能I/IO_TRACEN以使MCU分配TRACE的引脚(参见 20.16.2 跟踪引脚分配，20.15.3调试MCU配置寄存器)
- 向 ITM Lock Access 寄存器写入 0xC5ACCE55，以允许写其他 ITM 寄存器
- 向 Trace Control 寄存器写入 0x00010005，使能 TPIU 的同步包并使能整个 ITM 功能，寄存器中的 ATB ID 为 0x01。
- 向 ITM Trace Enable 寄存器写入 0x1，以使能触发端口 0
- 向 ITM Trace Privilege 寄存器写入 0x1，关闭对触发端口 7：0 的屏蔽
- 把需要输出的值写入触发端口 0 寄存器，这个步骤可以通过软件完成（使用 printf 功能）

20.15 MCU调试模块(MCUDBG)

MCU 调试模块协助调试器提供以下功能：

- 低功耗模式
- 在断点时提供定时器，看门狗和 bxCAN 的时钟控制
- 对跟踪脚分配的控制

20.15.1 低功耗模式的调试支持

使用 WFI 和 WFE 可以进入低功耗模式。
MCU 支持多种低功耗模式，分别可以关闭 CPU 时钟，或降低 CPU 的能耗。

内核不允许在调试期间关闭 FCLK 或 HCLK。这些时钟对于调试操作是必要的，因此在调试期间，它们必须工作。MCU 使用一种特殊的方式，允许用户在低功耗模式下调试代码。

为实现这一功能，调试器必须先设置一些配置寄存器来改变低功耗模式的特性。

- 在睡眠模式下，调试器必须先置位 DBGMCU_CR 寄存器的 DBG_SLEEP 位。这将为 HCLK 提供与 FCLK（由代码配置的系统时钟）相同的时钟。
- 在停止模式下，调试器必须先置位 DBG_STOP 位。这将激活内部 RC 振荡器，在停止模式下为 FCLK 和 HCLK 提供时钟。

20.15.2 支持定时器、看门狗、bxCAN和I2C的调试

在产生断点时，有必要根据定时器和看门狗的不同用途选择计数器的工作模式：

- 在产生断点时，计数器继续计数。这在输出 PWM 控制电机时常常要用到。
- 在产生断点时，计数器停止计数。这对于看门狗的计数器是必需的。

对于 bxCAN，用户可以选择在断点期间阻止接收寄存器的更新。
对于 I^2C，用户可以选择在断点期间阻止 SMBUS 超时。

20.15.3 调试MCU配置寄存器

此寄存器允许在调试状态下下配置 MCU。包括：

- 支持低功耗模式
- 支持定时器和看门狗的计数器
- 支持 bxCAN 通信
- 分配跟踪引脚

DBGMCU_CR 寄存器被映射到外部 PPB 总线，基地址为 0xE0042000。

寄存器由 PORESET 异步复位（不被系统复位所复位）。当内核处于复位状态下时，调试器可写该寄存器。

如果调试器不支持这些特性，用户软件仍可写这些寄存器。

DBGMCU_CR

地址：0xE0042004
只支持 32 位访问
POR 复位：0x00000000（不被系统复位所复位）

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															DBG_I2C2_SMBUS TIMEOUT
															rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DBG_I2C1_SMBUS TIMEOUT	DBG CAN STOP	DBG TIM4 STOP	DBG TIM3 STOP	DBG TIM2 STOP	DBG TIM1 STOP	DBG WWDG STOP	DBG IWDG STOP	TRACE MODE[1:0]		TRACE IOEN	保留		DBG STANDBY	DBG STOP	DBG SLEEP
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw			rw	rw	rw

位31:17	保留，必须保持为0。

位16	DBG_I2C2_SMBUS_TIMEOUT：当核心停止时停止SMBUS超时模式 0: 与正常模式操作相同 1: 冻结SMBUS的超时控制
位15	DBG_I2C1_SMBUS_TIMEOUT：当核心停止时停止SMBUS超时模式 0: 与正常模式操作相同 1: 冻结SMBUS的超时控制
位14	DBG_CAN_STOP：当内核进入调试状态时，CAN停止运行 0：CAN仍然正常运行 1：CAN的接收寄存器不继续接收数据
位13:10	DBG_TIMx_STOP：当内核进入调试状态时计数器停止工作 x=4..1 0：选中定时器的计数器仍然正常工作 1：选中定时器的计数器停止工作
位9	DBG_WWDG_STOP：当内核进入调试状态时调试窗口看门狗停止工作 0：窗口看门狗计数器仍然正常工作 1：窗口看门狗计数器停止工作
位8	DBG_IWDG_STOP：当内核进入调试状态时看门狗停止工作 0：看门狗计数器仍然正常工作 1：看门狗计数器停止工作
位7:5	TRACE_MODE[1:0]和TRACE_IOEN：跟踪引脚分配控制 - 当TRACE_IOEN＝0时：TRACE_MODE＝xx：不分配跟踪引脚（默认状态） - 当TRACE_IOEN=1时： TRACE_MODE=00：跟踪引脚使用异步模式 TRACE_MODE=01：跟踪引脚使用同步模式，并且数据长度为1 TRACE_MODE=10:跟踪引脚使用同步模式，并且数据长度为2 TRACE_MODE=11:跟踪引脚使用同步模式，并且数据长度为4
位4:3	保留，必须保持为0。
位2	DBG_STANDBY：调试待机模式 0：（FCLK关，HCLK关）整个数电部分都断电 从软件的观点看，退出STANDBY模式与复位是一样的（除了一些状态位指示了微控制器刚从STANDBY状态退出） 1：（FCLK开，HCLK开）数电部分不下电，FCLK和HCLK时钟由内部RL振荡器提供时钟。另外，微控制器通过产生系统复位来退出STANDBY模式和复位是一样的。
位1	DBG_STOP：调试停止模式 0：（FCLK关，HCLK关）在停止模式时，时钟控制器禁止一切时钟（包括HCLK和FCLK）。当从STOP模式退出时，时钟的配置和复位之后的配置一样（微控制器由8MHz的内部RC振荡器（HIS）提供时钟）。因此，软件必需重新配置时钟控制系统启动PLL，Xtal等。 1：（FCLK开，HCLK开）在停止模式时，FCLK和HCLK时钟由内部RC振荡器提供。当退出停止模式时，软件必需重新配置时钟系统启动PLL，Xtal等（与配置此比特位为0时的操作一样）。
位0	DBG_SLEEP：调试睡眠模式 0：（FCLK开，HCLK关）在睡眠模式时，FCLK由原先已配置好的系统时钟提供，HCLK则关闭。由于睡眠模式不会复位已配置好的时钟系统，因此从睡眠模式退出时，软件不需要重新配置时钟系统。 1：（FCLK开，HCLK开）在睡眠模式时，FCLK和HCLK时钟都有原先已配置好的系统时钟提供。

20.16 TPIU (trace port interface unit)

20.16.1 导言

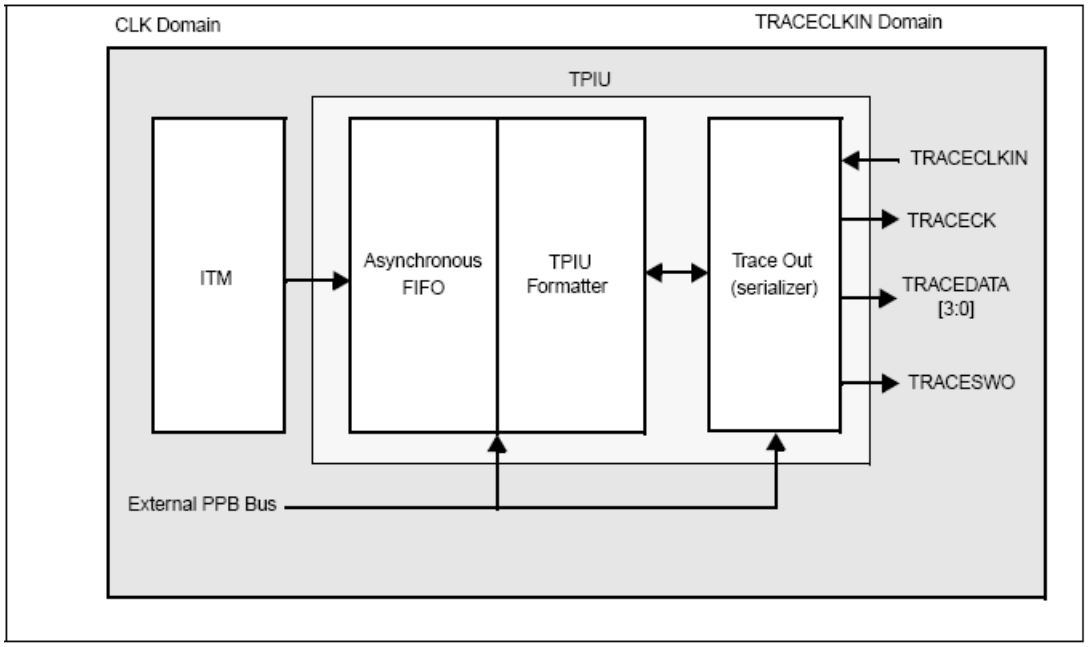
TPIU 在片上追踪数据和 ITM 之间担当桥梁的作用。

输出的数据流封装成跟踪源 ID，然后被追踪端口分析器（*Trace Port Analyzer*）采集。

内核嵌入了一个简单的专门为低价调试所设计的 TPIU（由一个特殊版本的 CoreSight TPIU 组成）。

该 TPIU 只支持 ITM 调试跟踪，是有条件的跟踪，只输出来自 TIM 的信息。

图188 TPIU 框图



20.16.2 跟踪引脚分配

- 异步模式

异步模式需要 1 个额外的引脚并且存在于所有的封装。此模式仅在串行调试接口有效（不支持 JTAG 调试接口）。

表82 异步跟踪管脚分配

TPUI引脚	同步跟踪模式		STM32F10xxx引脚分配
	类型	描述	
TRACESWO	输出	异步跟踪数据输出	PB3

- 同步模式

同步模式根据跟踪数据长度使用 2 到 6 个额外引脚，并且只存在于大封装芯片里。此外，此模式在 JTAG 调试接口和串行调试接口下都可使用，并提供比异步跟踪更好的数据输出量。

表83 同步跟踪管脚分配

TPUI引脚名	同步跟踪模式		STM32F10xxx引脚分配
	类型	描述	
TRACECK	输出	跟踪时钟	PE2
TRACED[3:0]	输出	同步跟踪数据输出，长度可以是1,2,或4	PE[6:3]

TPUI 跟踪引脚分配

这些引脚在默认状态下不是专用引脚。可以通过设置 *MCU Debug Component Configuration* 寄存器的 IOTRACEN 和 OTRACEMODE 位分配这些引脚。必需由调试器完成设置。

此外，由跟踪的配置决定分配的引脚数（异步还是同步）。

- 异步模式：需要 1 个额外引脚
- 同步模式：根据跟踪端口的数据长度（1，2 或 4）决定使用 2 到 5 个额外的引脚
  - TRACECK
  - TRACED(0)如果数据长度配置为 1，2 或 4
  - TRACED(1)如果数据长度配置为 2 或 4
  - TRACED(2)如果数据长度配置为 4
  - TRACED(3)如果数据长度配置为 4

调试器需要设置 Debug MCU Configuration 寄存器的 TRACE_IOEN 和 TRACE_MODE[1:0]位来分配跟踪引脚。默认时，跟踪脚是不分配的。

此寄存器被映射到外部 PPB 并且被 PORESET 所复位（系统复位不复位此寄存器）。调试器可以在系统复位的状态下写该寄存器。

表84 灵活的跟踪管脚分配

DBGMCU_CR寄存器		引脚用途	跟踪引脚分配					
TRACE_IOEN	TRACE_MODE[1:0]		PB3/ JTDO/ TRACESWO	PE2/ TRACECK	PE3/ TRACED[0]	PE4/ TRACED[1]	PE5/ TRACED[2]	PE6/ TRACED[3]
0	XX	无跟踪（默认状态）	释放(1)	释放（可用作普通I/O口）				
1	00	异步跟踪	TRACESWO					
1	01	同步跟踪1位	释放(1)	TRACECK	TRACED[0]	释放 (可用作普通I/O口)		
1	10	同步跟踪2位		TRACECK	TRACED[0]	TRACED[1]	释放(可用作普通I/O口)	
1	11	同步跟踪4位		TRACECK	TRACED[0]	TRACED[1]	TRACED[2]	TRACED[3]
注释(1)：使用串行调试接口使，此引脚被释放，使用JTAG调试接口时，此引脚用作JTDO。								

注意： TUIP 的输入时钟 TRACECLKIN 默认接地。所以在比特位 TRACE_IOEN 被置位后，HCLK 需要两个时钟周期。

然后，调试器可以通过写 TPIU 的 SPP_R（Selected Pin Protocol）寄存器的 PROTOCOL [1:0]位来配置跟踪模式。

- PROTOCOL=00: 跟踪模式（同步）
- PROTOCOL=01 或 10：串行模式（曼彻斯特或 NRZ 编码）。默认状态为 01

然后通过写 TPIU 的 CPSPS_R（Current Sync Port Size）寄存器的比特位[3:0] 来配置跟踪端口的大小。

- 0x1：1 个引脚(缺省)
- 0x2：2 个引脚
- 0x8：4 个引脚

20.16.3 TPUI格式器

协议格式器输出 16 个字节组成的帧：

- 7 个数据字节
- 8 个多用途字节，由以下部分组成：
  - 1 位(LSB)用来区分数据字节（0）和 ID 字节（1）。
  - 7 位(MSB)可以作为数据或跟踪源 ID 的变化。

- 1 个辅助字节，其中的每个位都对应于 8 个多用途字节中的一个：
  - 如果对应的多用途字节是数据字节，那么这个位是数据的比特 0 位。
  - 如果对应字节是 ID 字节，这个位表明 ID 变化何时生效。

注意： 更多信息，请参考 ARM CoreSight Architecture Specification v1.0（ARM IHI 0029B）

STM32F10xxx 微控制器格式器的使用

对于 STM32F10xxx 微控制器，只存在一个 TRACE 源（ITM）。但由于没有分配 TRACECTL 引脚，所以格式器不能被禁止，必须以旁路模式使用。据此，跟踪端口分析器可以解码部分格式协议以判断触发的位置。

20.16.4 TPUI帧异步包

TPUI 会产生两种类型的同步包：

- 帧同步包（或全字同步包）

  该包为 0x7F_FF_FF_FF（LSB 先发），这个序列只有在 0x7F 没有被作为 ID 源编码的时候才能使用。

  该包在帧之间周期性地输出。

  在连续模式里，一旦同步帧被发现，TPA 必须抛弃所有这些帧。

- 半字同步包

  该包为：0x7F_FF（LSB 先发）。

  它在帧之间或帧内周期性的输出。

  这些包只存在于连续模式中，并且使能 TPA 检测 IDLE 模式下的 TRACE 口（无 TRACE 被捕捉）。当被 TPA 检测到时，必须将其抛弃。

20.16.5 同步帧包的发送

由于内核的 TPIU 内没有同步计数寄存器，因此同步的触发只能由 DWT 产生。参 DWT Control Register 寄存器（SYNCTAP[11:10]位）和 DWT Current PC Sampler Cycle Count 寄存器。

TPUI 帧同步包（0x7F_FF_FF_FF）在下列情况时被发送：

- 在每个 TPIU 复位释放后。复位信号同步于 TRACECLKIN 时钟的上升沿释放，这意味着当 DBGMCU_CFG 寄存器的 IO_TRACEN 位被置位时，同步包就被发送。这种情况下，包 0x7F_FF_FF_FF 后面不跟任何格式的包。
- 在每个 DWT 触发时（假设已事先设置好 DWT），有以下两种情况：
  - 如果 ITM 的 SYNENA 位被复位，只发送字 0x7F_FF_FF_FF，后面不跟任何格式的数据包。

– 如果 ITM 的 SYNENA 位被置位，ITM 同步包将跟在(0x80_00_00_00_00)后面，由 TPUI 编排格式（加上跟踪源 ID）。

20.16.6 同步模式

跟踪输出数据的引脚数可以为 4 个，2 个或者 1 个，由 TRACED（3:0）

配置时钟输出到调试器（TRACECK）TRACECLKIN 在内部被驱动，并仅当使用 TRACE 时和 HCLK 相连接。

注意： *在此类同步模式中，不需要提供稳定的时钟频率。*

TRACE 的 I/O 端口（包括 TRACECK）由 TRACLKIN 的上升沿驱动（等同于 HCLK）。因此，TRACECK 的输出频率等于 HCLK/2.

20.16.7 异步模式

调试模块提供一个低成本的，只使用一个引脚的跟踪数据输出功能，即使用异步输出引脚 TRACESWO。但显然，这样的输出数据带宽是有限的。

TRACESWO 引脚与 JTDO 引脚复用，只在 SW-DP 调试接口有效，因此 STM32F10xxx 的所有封装都提供这种功能。

异步模式需要 TRACECLKIN 引脚有平稳的频率提供。对标准的 UART（NRZ）捕捉机制来说，需要 5%的正确度。曼彻斯特编码可放宽到 10%。

20.16.8 TRACECLKIN在STM32F10xxx内部的连接

TRACECLKIN 输入在 STM32F10xxx 内部与 HCLK 相连接。这意味着在使用异步跟踪模式时，应用程序应限制使用时间帧保证 CPU 频率的稳定。

注意：重要：当使用异步跟踪功能时需注意：

STM32F10xxx 微控制器的初时时钟是内部 RC 振荡器， 此振荡器在复位状态下的频率与复位后的频率不同。这是由于 RC 校准在复位状态下使用初始值，而这个值在复位释放后会更新。

因此，不应该在系统复位状态下激活跟踪端口分析器（Trace Port Analyzer ）的跟踪功能（置位 IOTRACEN）。因为在复位状态下的同步帧包的比特宽度与复位后的包不同。

20.16.9 TPIU寄存器

只有当 Debug Exception and Monitor Control（DEMCR）寄存器的 TRCENA 位被置位时，TPIU APB 寄存器才可以被读写。否则寄存器读出值为 0（这一位的输出使能 TPIU 的 PCLK）。

表85 重要的 TPIU 寄存器

地址	寄存器	描述
0xE0040004	Current port size	跟踪端口的长度： Bit0：端口长度为1 Bit1：端口长度为2 Bit2：端口长度为3，不支持 Bit3：端口长度为4 四个比特中只能同时置位一个比特。 默认状态下，端口长度为1（0x00000001）
0xE00400F0	Selected pin protocol	跟踪端口协议的选择： Bit1:0＝ 00：同步跟踪模式 01：串行输出－曼彻斯特编码（默认值） 10：串行输出－NRZ 11：未定义
0xE0040304	Formatter and flush control	Bit31-9 ：总是0 Bit8 ＝ TrigIn： 总是1，指示触发器 Bit7-4：总是0 Bit3-2：总是0 Bit1＝EnFCont：同步模式下（Select Pin Protocol寄存器的Bit1：0为00），此比特位强制为1，连续模式下格式器被自动使能。异步模式下（Select Pin Protocol寄存器的Bit1：0不为00），此比特可以被置位或复位来选择是否使能格式器。 Bit0：总是0 默认值为0x102 注意：在同步模式下，由于TRACECTL信号没有外部引脚，因此格式器会在连续模式下自动使能。这意味着格式器会插入一些控制包来识别跟踪包的源。
0xE0040300	Formatter and flush status	没有在 Cortex-M3 中使用，读出值始终为 0x00000008

20.16.10 配置的例子

- 设置 Debug Exception and Monitor Control 寄存器的 TRCENA 位
- 在 TPIU Current Port Size 寄存器中写入期望值（缺省是 0x1，指示端口长度为 1bit）
- 向 TPIU Formatter and Flush Control 寄存器中写入 0x102（默认值）

- 写 TPIU Select Pin Protocol 寄存器，选择同步或异步模式。例如写 0x2 选择 NRZ 编码的异步模式（类似 URAT）
- 向 DBGMCU Control 寄存器写入 0x20（置位 IO_TRACEN），为异步模式分配 TRACE 的 I/O 口。此时 TPIU 将发出一个同步包（FF_FF_FF_7F）。
- 配置 ITM 并且写 ITMStimulus 寄存器输出数据

20.17 DBG寄存器地址映象

下列表格归纳了调试寄存器。

表86 DBG – 寄存器和复位值

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
E00420	DBGMCU_IDCODE	REV_ID																保留				DEV_ID											
	复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0					0	1	0	0	0	0	0	1	0	0	0	0
E00420	DBGMCU_CR	保留																DBG_I2C2_SMBUS	DBG_I2C1_SMBUS	DBG_CAN_STOP	DBG_TIM4_STP	DBG_TIM3_STP	DBG_TIM2_STP	DBG_TIM1_STP	DBG_WWDG_STOP	DBG_IWDG_STOP	TRACE_MODE [1:0]		TRACE_IOEN	保留		DBG_STANDBY	DBG_STOP	DBG_SLEEP
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0			0	0	0

关于寄存器的起始地址，请参见第 1 章。

附录A　重要提示

附录所列的提示，仅对STM32F103xx增强型系列芯片的Z版本有效，关于芯片版本号的具体信息，请参考STM32F10xxx参考手册的 20.6.1 章节。

A.1　PD0 和PD1 在输出模式下

由于 PD0 和 PD1 仅工作在 50MHz 模式下，因此这两个引脚在用作输出模式时是限制的。

A.2　ADC自动注入通道

当 ADC 时钟使用 4 或 8 的预分频时，从普通模式转到注入转换时会自动插入一个 ADC 时钟的延迟， 当 ADC 使用 2 预分频的时钟时，插入的延迟为 2 个 ADC 时钟。

A.3　ADC的混合同步注入+交替模式

当 ADC 使用 4 预分频的时钟时，交替采样的时间间隔并不平均，也就是说采样的间隔并不是标准的 7 个 ADC 时钟，而是 8 个 ADC 时钟和 6 个 ADC 时钟交替。

A.4　ADC通道 0

当 ADC 处于注入触发模式时，在某些特殊情况下，ADC 通道 0 会产生一个低幅度的脉冲尖峰信号。

此脉冲由内部耦合器产生，与正在使用哪个 ADC 注入通道无关，在普通模式和注入模式切换时产生，并同步到注入序列的开头。

此脉冲的幅度小于 150mV，持续时间的典型值为 10ns。当数字输入和输出信号的负载低于 5kΩ时，不会产生影响。