参考手册

小，中和大容量的 STM32F101xx, STM32F102xx 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，提供关于如何使用小容量、中容量和大容量的STM32F101xx、STM32F102xx或者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、STM32F102xx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、电气和物理性能参数，请参考STM32F101xx、STM32F102xx和STM32F103xx的数据手册。

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技术参考手册。

相关文档

- Cortex™-M3技术参考手册，可按下述链接下载：

http://infocenter.arm.com/help/topic/com.arm.doc.ddi0337e/DDI0337E_cortex_m3_r1p1_trm.pdf

下述文档可在ST网站下载(http://www.st.com/mcu/)：

- STM32F101xx、STM32F102xx和STM32F103xx的数据手册。
- STM32F10xxx闪存编程手册。

* 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_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。
toggle (t)	软件只能通过写'1'来翻转此位，写'0'对此位无影响。
Reserved(Res.)	保留位，必须保持默认值不变

1.2 术语表

- 小容量产品是指闪存存储器容量在16K至32K字节之间的STM32F101xx、STM32F102xx和STM32F103xx微控制器。
- 中容量产品是指闪存存储器容量在64K至128K字节之间的STM32F101xx、STM32F102xx和STM32F103xx微控制器。
- 大容量产品是指闪存存储器容量在256K至512K字节之间的STM32F101xx和STM32F103xx微控制器。

1.3 可用的外设

有关STM32微控制器系列全部型号中，某外设存在与否及其数目，请查阅相应的小容量、中容量或者大容量STM32F101xx和STM32F103xx以及小容量和中容量STM32F102xx的数据手册。

2 存储器和总线构架

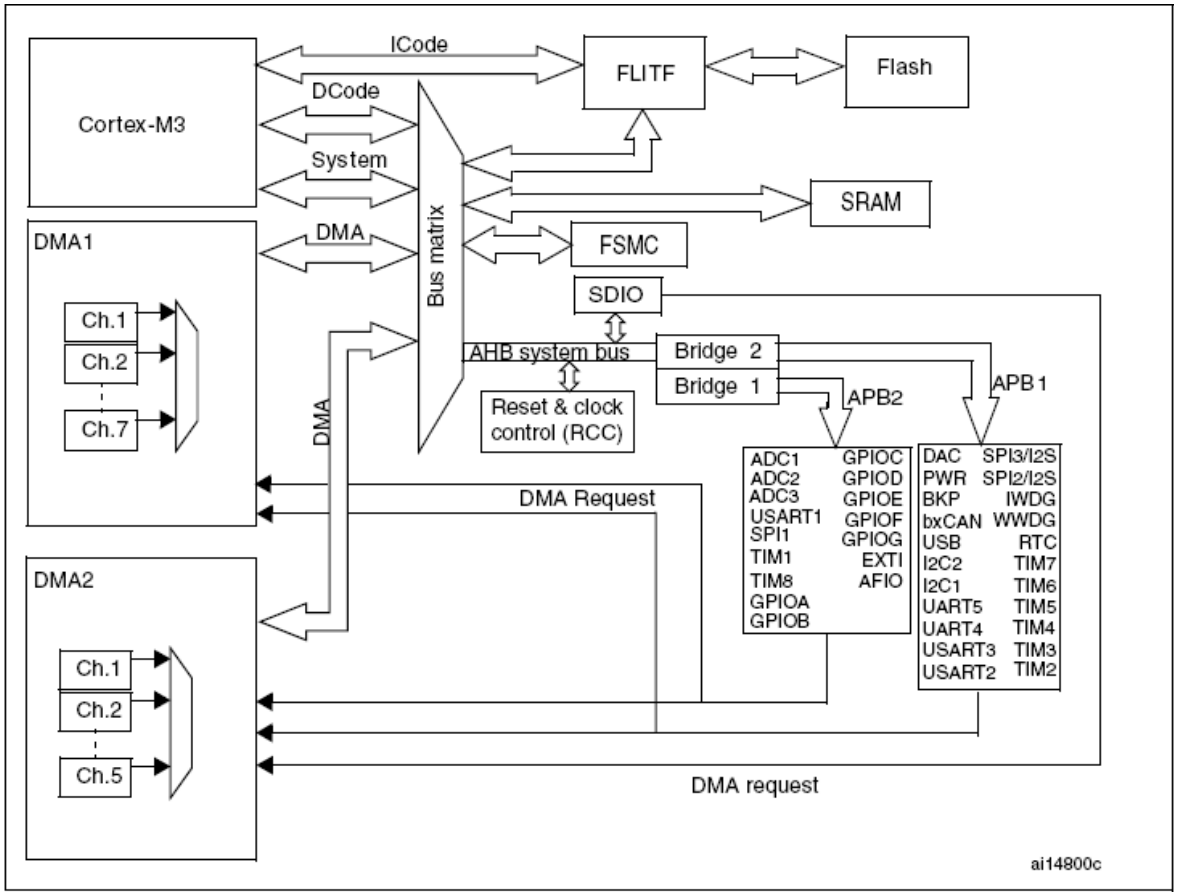
2.1 系统构架

主系统由以下部分构成：

- 四个驱动单元：
  - Cortex™-M3 内核 DCode 总线(D-bus)，和系统总线(S-bus)
  - 通用 DMA1 和通用 DMA2
- 四个被动单元
  - 内部 SRAM
  - 内部闪存存储器
  - FSMC
  - 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内核的指令总线与闪存指令接口相连接。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。此总线矩阵由四个驱动部件(CPU的DCode、系统总线、DMA1总线和DMA2总线)和四个被动部件(闪存存储器接口(FLITF)、SRAM、FSMC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)。

有关连接到每个桥的不同外设的地址映射请参考表1。在每一次复位以后，所有除SRAM和FLITF以外的外设都被关闭，在使用一个外设之前，必须设置寄存器RCC_AHBENR来打开该外设的时钟。

注意：当对APB寄存器进行8位或者16位访问时，该访问会被自动转换成32位的访问：桥会自动将8位或者32位的数据扩展以配合32位的向量。

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。

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，请参考相应器件的数据手册中的存储器映像图。

2.3 存储器映像

请参考相应器件的数据手册中的存储器映像图。表1列出了所用STM32F10xxx中内置外设的起始地址。

表1 寄存器组起始地址

起始地址	外设	总线	寄存器映像
0x4002 3400 - 0x4002 3FFF	保留	AHB	
0x4002 3000 - 0x4002 33FF	CRC		参见3.4.4节
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(RCC)		参见6.3.11节
0x4002 0800 - 0x4002 0FFF	保留		
0x4002 0400 - 0x4002 07FF	DMA2		参见9.4.7节
0x4002 0000 - 0x4002 03FF	DMA1		参见9.4.7节
0x4001 8400 - 0x4001 7FFF	保留		
0x4001 8000 - 0x4001 83FF	SDIO		参见19.9.16节
0x4001 4000 - 0x4001 7FFF	保留	APB2	
0x4001 3C00 - 0x4001 3FFF	ADC3		参见10.12.15节
0x4001 3800 - 0x4001 3BFF	USART1		参见24.6.8节
0x4001 3400 - 0x4001 37FF	TIM8定时器		参见12.4.21节
0x4001 3000 - 0x4001 33FF	SPI1		参见22.5节
0x4001 2C00 - 0x4001 2FFF	TIM1定时器		参见12.4.21节
0x4001 2800 - 0x4001 2BFF	ADC2		参见10.12.15节
0x4001 2400 - 0x4001 27FF	ADC1		参见10.12.15节
0x4001 2000 - 0x4001 23FF	GPIO端口G		参见7.5节
0x4001 2000 - 0x4001 23FF	GPIO端口F		参见7.5节
0x4001 1800 - 0x4001 1BFF	GPIO端口E		参见7.5节
0x4001 1400 - 0x4001 17FF	GPIO端口D		参见7.5节
0x4001 1000 - 0x4001 13FF	GPIO端口C		参见7.5节
0X4001 0C00 - 0x4001 0FFF	GPIO端口B		参见7.5节
0x4001 0800 - 0x4001 0BFF	GPIO端口A		参见7.5节
0x4001 0400 - 0x4001 07FF	EXTI		参见8.3.7节
0x4001 0000 - 0x4001 03FF	AFIO		参见7.5节
0x4000 7800 - 0x4000FFFF	保留	APB1	
0x4000 7400 - 0x4000 77FF	DAC		参见11.5.14节
0x4000 7000 - 0x4000 73FF	电源控制(PWR)		参见4.4.3节
0x4000 6C00 - 0x4000 6FFF	后备寄存器(BKP)		参见5.4.5节
0x4000 6800 - 0x4000 6BFF	保留		
0x4000 6400 - 0x4000 67FF	bxCAN		参见21.6.5节
0x4000 6000 - 0x4000 63FF	USB/CAN共享的SRAM 512字节		
0x4000 5C00 - 0x4000 5FFF	USB全速设备寄存器		参见20.5.4节
0x4000 5800 - 0x4000 5BFF	I2C2		参见23.6.10节
0x4000 5400 - 0x4000 57FF	I2C1		参见23.6.10节

0x4000 5000 - 0x4000 53FF	UART5		参见24.6.8节
0x4000 4C00 - 0x4000 4FFF	UART4		参见24.6.8节
0x4000 4800 - 0x4000 4BFF	USART3		参见24.6.8节
0x4000 4400 - 0x4000 47FF	USART2		参见24.6.8节
0x4000 4000 - 0x4000 3FFF	保留		
0x4000 3C00 - 0x4000 3FFF	SPI3/I2S3		参见22.5节
0x4000 3800 - 0x4000 3BFF	SPI2/I2S3		参见22.5节
0x4000 3400 - 0x4000 37FF	保留		
0x4000 3000 - 0x4000 33FF	独立看门狗(IWDG)		参见16.4.5节
0x4000 2C00 - 0x4000 2FFF	窗口看门狗(WWDG)		参见17.6.4节
0x4000 2800 - 0x4000 2BFF	RTC		参见15.4.7节
0x4000 1800 - 0x4000 27FF	保留		
0x4000 1400 - 0x4000 17FF	TIM7定时器		参见13.4.19节
0x4000 1000 - 0x4000 13FF	TIM6定时器		参见13.4.19节
0x4000 0C00 - 0x4000 0FFF	TIM5定时器		参见13.4.19节
0x4000 0800 - 0x4000 0BFF	TIM4定时器		参见13.4.19节
0x4000 0400 - 0x4000 07FF	TIM3定时器		参见13.4.19节
0x4000 0000 - 0x4000 03FF	TIM2定时器		参见13.4.19节

2.3.1 嵌入式SRAM

STM32F10xxx内置64K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。

2.3.2 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地址的写操作与对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 或 0x00)。

请参考《Cortex™-M3技术参考手册》以了解更多有关位段的信息。

2.3.3 嵌入式闪存

高性能的闪存模块有以下的主要特性：

- 高达512K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：
  - 主存储块容量：
    小容量产品主存储块为 4Kbx64 位，每个主存储块划分为 32 个 1K 字节的页。
    中容量产品主存储块为 16Kbx64 位，每个主存储块划分为 128 个 1K 字节的页。
    大容量产品主存储块为 64Kbx64 位，每个主存储块划分为 256 个 2K 字节的页。
  - 信息块为 258x64 位，每个信息块划分为一个 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(小容量产品)

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31	0x0800 1000 - 0x0800 13FF	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
闪存存储器接口寄存器	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

表3 闪存模块的组织(中容量产品)

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
闪存存储器接口寄存器	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

表4 闪存模块的组织(大容量产品)

模块	名称	地址	大小(字节)
主存储块	页0	0x0800 0000 - 0x0800 07FF	2K
	页1	0x0800 0800 - 0x0800 0FFF	2K
	页2	0x0800 1000 - 0x0800 17FF	2K
	页3	0x0800 0C00 - 0x0800 0FFF	2K
	页4	0x0800 1800 - 0x0800 1FFF	2K
	… …	… …	… …
	页127	0x0801 F800 - 0x0801 FFFF	2K
信息块	系统存储器	0x1FFF F000 - 0x1FFF F7FF	2K
	用户选择字节	0x1FFF F800 - 0x1FFF F80F	16
闪存存储器接口寄存器	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

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。
- 预取缓冲区(2个64位)：在每一次复位以后被自动打开，由于每个缓冲区的大小(64位)与闪存的带宽相同，因此只通过需一次读闪存的操作即可更新整个缓冲区的内容。由于预取缓冲区的存在，CPU可以工作在更高的主频。CPU每次取指最多为32位的字，取一条指令时，下一条指令已经在缓冲区中等待。
- 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

2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。该特性只能用在时钟频率为8MHz或低于8MHz时，可以直接使用的内部RC振荡器(HSI)，或者是主振荡器(HSE)，但不能用PLL。

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. 预取缓冲器的打开和关闭操作只有在系统时钟(SYSCLK)小于24MHz时才能执行。一般而言，预取缓冲器的打开和关闭操作在初始化过程中执行，这时微控制器的时钟由8MHz的内部RC振荡器(HSI)提供。

5. 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5 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

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。

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

因为固定的存储器映像，代码区始终从地址0x0000 0000开始(通过ICode和DCode总线访问)，而数据区(SRAM)始终从地址0x2000 0000开始(通过系统总线访问)。Cortex-M3的CPU始终从ICode总线获取复位向量，即启动仅适合于从代码区开始(典型地从Flash启动)。STM32F10xxx微控制器实现了一个特殊的机制，系统可以不仅仅从Flash存储器或系统存储器启动，还可以从内置SRAM启动。

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

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

注意： 当从内置SRAM启动，在应用程序的初始化代码中，必须使用NVIC的异常表和偏移寄存器，从新映射向量表之SRAM中。

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。进一步的细节请查询AN2606。

3 CRC计算单元(CRC)

小容量产品是指闪存存储器容量在16K至32K字节之间的STM32F101xx、STM32F102xx和STM32F103xx微控制器。

中容量产品是指闪存存储器容量在64K至128K字节之间的STM32F101xx、STM32F102xx和STM32F103xx微控制器。

大容量产品是指闪存存储器容量在256K至512K字节之间的STM32F101xx和STM32F103xx微控制器。

除非特别说明，本章节描述的模块应用于整个STM32F10xxx微控制器系列。

3.1 CRC简介

循环冗余校验(CRC)计算单元是根据固定的生成多项式得到任一32位全字的CRC计算结果。

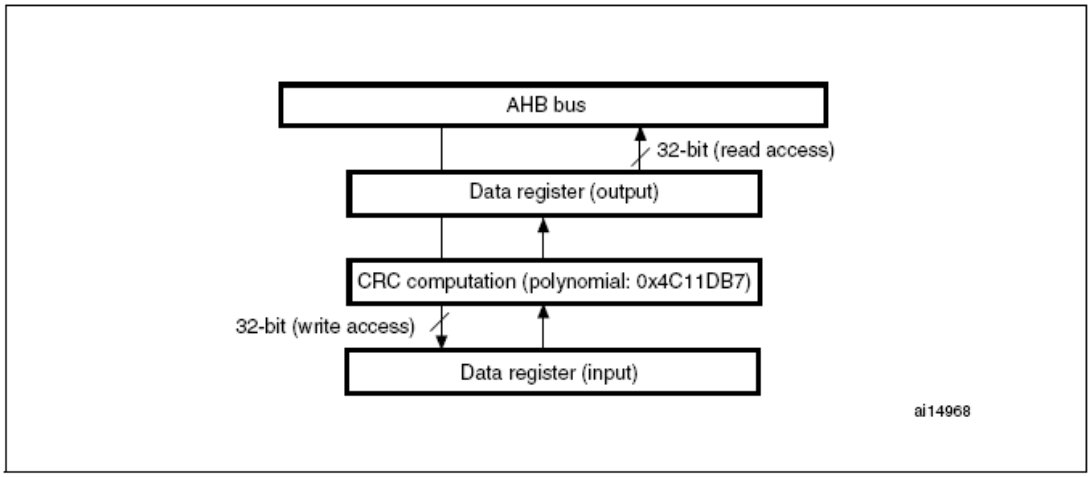
在其他的应用中， CRC技术主要应用于核实数据传输或者数据存储的正确性和完整性。标准EN/IEC 60335-1即提供了一种核实闪存存储器完整性的方法。CRC计算单元可以在程序运行时计算出软件的标识，之后与在连接时生成的参考标识比较，然后存放在指定的存储器空间。

3.2 CRC主要特性

- 使用CRC-32(以太网)多项式：0x4C11DB7
  - $X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + X^8 + X^7 + X^4 + X^2 + X + 1$
- 一个32位数据寄存器用于输入 / 输出
- CRC计算时间：4个AHB时钟周期(HCLK)
- 通用8位寄存器(可用于存放临时数据)

下图为CRC计算单元框图

图2 CRC 计算单元框图



3.3 CRC功能描述

CRC计算单元含有1个32位数据寄存器：

- 对该寄存器进行写操作时，作为输入寄存器，可以输入要进行CRC计算的新数据。
- 对该寄存器进行读操作时，返回上一次CRC计算的结果。

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

计算进行时会暂停CPU，无需加入软件等待周期，因此可以对寄存器CRC_DR进行背靠背写入或者连续地写-读操作。

可以通过把寄存器CRC_CR的RESET位置'0'来重置寄存器CRC_DR为0xFFFF FFFF。该操作不影响寄存器CRC_IDR内的数据。

3.4 CRC寄存器

CRC计算单元包括2个数据寄存器和1个控制寄存器

3.4.1 数据寄存器(CRC_DR)

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

位31:0	数据寄存器位 写入CRC计算器的新数据时作为输入寄存器 读取时返回CRC计算的结果

3.4.2 独立数据寄存器(CRC_IDR)

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								IDR[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

位31:8	保留。
位7:0	通用8位数据寄存器位 可用于临时存放1字节的数据。 寄存器CRC_CR的RESET位产生的CRC复位对本寄存器没有影响

3.4.3 控制寄存器(CRC_CR)

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
保留															RESET
															w

位31:1	保留。
位0	RESET位 复位CRC计算单元，设置数据寄存器为0xFFFF FFFF 该位只能置'1'，它由硬件自动清'0'

3.4.4 CRC寄存器映像

下表列出了CRC的寄存器映像和复位值

表6 CRC 计算单元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
0x00	CRC_DR	DR																															
	复位值	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x04	CRC_IDR	保留																								IDR[7:0]							
	复位值																									0	0	0	0	0	0	0	0
0x08	CRC_CR	保留																															RESET
	复位值																																0

4 电源控制(PWR)

小容量产品是指闪存存储器容量在16K至32K字节之间的STM32F101xx、STM32F102xx和STM32F103xx微控制器。

中容量产品是指闪存存储器容量在64K至128K字节之间的STM32F101xx、STM32F102xx和STM32F103xx微控制器。

大容量产品是指闪存存储器容量在256K至512K字节之间的STM32F101xx和STM32F103xx微控制器。

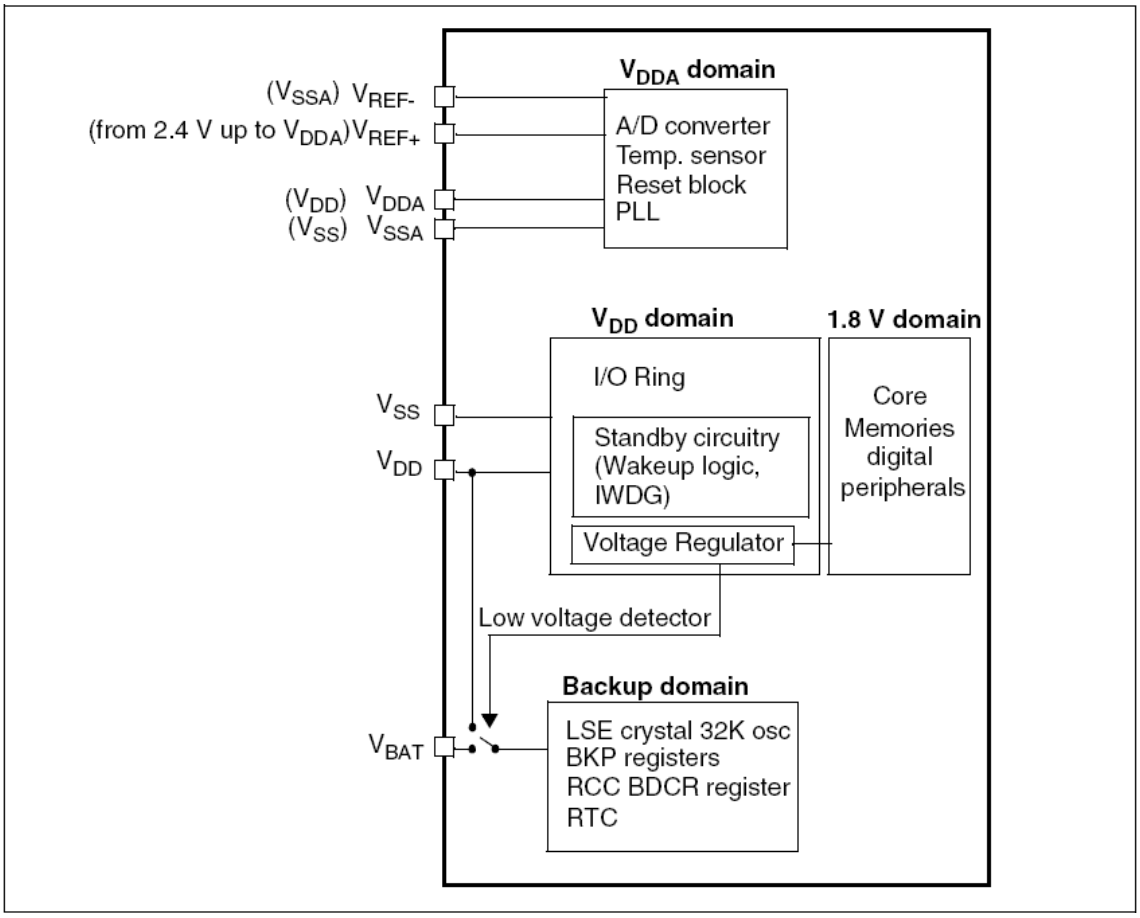
除非特别说明，本章节描述的模块应用于整个STM32F10xxx微控制器系列。

4.1 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}。*

4.1.1 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脚和144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。

4.1.2 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($t_{RSTTEMPO}$)或者探测到PVD之后，V_{BAT}和V_{DD}之间的电源开关仍会保持连接在V_{BAT}。在V_{DD}上升阶段，如果V_{DD}在小于$t_{RSTTEMPO}$的时间内达到稳定状态(关于$t_{RSTTEMPO}$可参考数据手册中的相关部分)，且$V_{DD} > V_{BAT} + 0.6V$时，电流可能通过V_{DD}和VBAT之间的内部二极管注入到V_{BAT}。

如果与V_{BAT}连接的电源或者电池不能承受这样的注入电流，强烈建议在外部V_{BAT}和电源之间连接一个低压降二极管。

如果在应用中没有外部电池，建议V_{BAT}在外部通过一个100nF的陶瓷电容与V_{DD}相连，更多细节参阅AN2586。

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(3mA)，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供电时(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))

4.1.3 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域外，寄存器和SRAM的内容全部丢失。

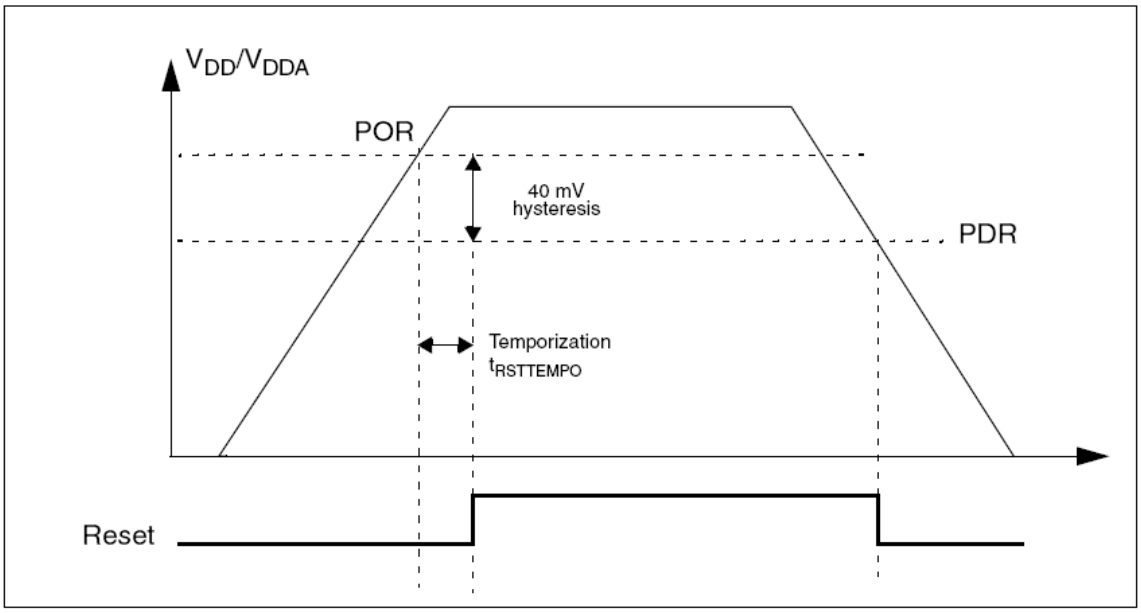
4.2 电源管理器

4.2.1 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}/V_{DDA}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。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



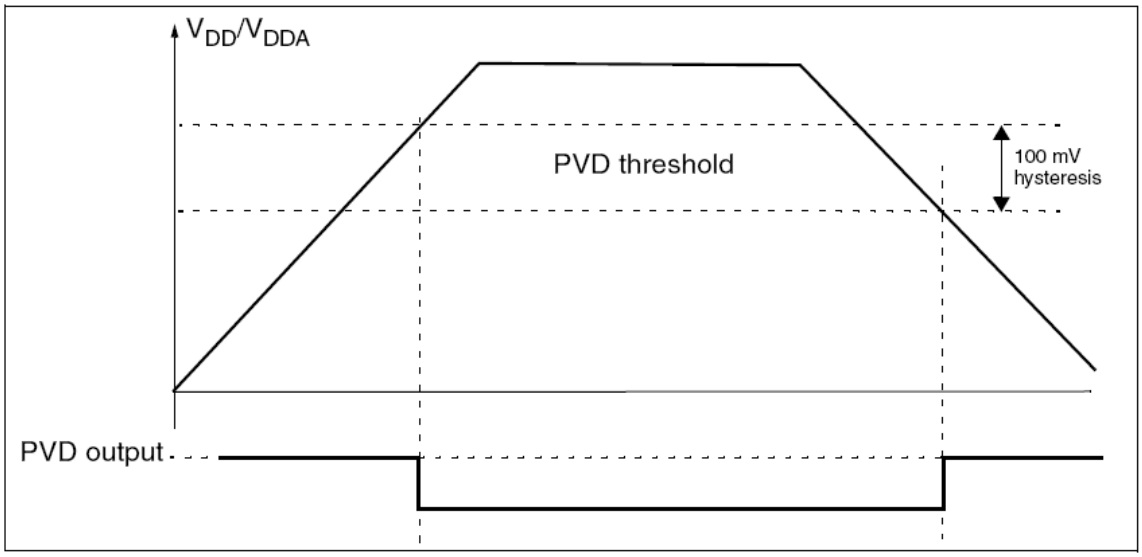
4.2.2 可编程电压监测器(PVD)

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



4.3 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。用户需要根据最低电源消耗，最快速启动时间和可用的唤醒源等条件，选定一个最佳的低功耗模式。

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7 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		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	无	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

4.3.1 降低系统时钟

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

详见6.3.2节。

4.3.2 外部时钟的控制

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

为了在睡眠模式下更多地减少功耗，可在执行WFI或WFE指令前关闭所有外设的时钟。

通过设置AHB外设时钟使能寄存器(RCC_AHBENR)、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。

4.3.3 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，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8和表9。

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，则一旦发生唤醒事件时，微处理器都将从睡眠模式退出。唤醒事件可以通过下述方式产生：

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

该模式唤醒所需的时间最短，因为没有时间损失在中断的进入或退出上。

关于如何退出睡眠模式，更多的细节参考表8和表9。

表8 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9 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

4.3.4 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、HSI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10。

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。

如果正在进行闪存编程，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)：可通过写入看门狗的键寄存器或硬件选择来启动IWDG。一旦启动了独立看门狗，除了系统复位，它不能再被停止。详见16.3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)：通过备用域控制寄存器(RCC_BDCR)的LSEON位设置。

在停止模式下，如果在进入该模式前ADC和DAC没有被关闭，那么这些外设仍然消耗电流。通过设置寄存器ADC_CR2的ADON位和寄存器DAC_CR的ENx位为0可关闭这2个外设。

退出停止模式

关于如何退出停止模式，详见下表。

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10 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。

4.3.5 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11。

可以通过设置独立的控制位，选择以下待机模式的功能：

- 独立看门狗(IWDG)：可通过写入看门狗的键寄存器或硬件选择来启动IWDG。一旦启动了独立看门狗，除了系统复位，它不能再被停止。详见16.3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)：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，详见下表。

表11 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。

4.3.6 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)
  该时钟源提供了一个低功耗且精确的时间基准。(在典型情形下消耗小于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闹钟事件将系统从停止模式下唤醒，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。

4.4 电源控制寄存器

4.4.1 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　　100：2.6V 001：2.3V　　101：2.7V 010：2.4V　　110：2.8V 011：2.5V　　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

4.4.2 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}/V_{DDA}高于由PLS[2:0]选定的PVD阀值 1：V_{DD}/V_{DDA}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。

4.4.3 PWR寄存器地址映像

以下表格列出所有PWR寄存器。

表12 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

关于寄存器的起始地址，参见表1。

5 备份寄存器(BKP)

小容量产品是指闪存存储器容量在16K至32K字节之间的STM32F101xx、STM32F102xx和STM32F103xx微控制器。

中容量产品是指闪存存储器容量在64K至128K字节之间的STM32F101xx、STM32F102xx和STM32F103xx微控制器。

大容量产品是指闪存存储器容量在256K至512K字节之间的STM32F101xx和STM32F103xx微控制器。

除非特别说明，本章节描述的模块应用于整个STM32F10xxx微控制器系列。

5.1 BKP简介

备份寄存器是42个16位的寄存器，可用来存储84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。执行以下操作可以使能对备份寄存器和RTC的访问。

- 通过设置寄存器RCC_APB1ENR的PWREN和BKPEN位来打开电源和后备接口的时钟
- 电源控制寄存器(PWR_CR)的DBP位来使能对后备寄存器和RTC的访问。

5.2 BKP特性

- 20字节数据后备寄存器(中容量和小容量产品)，或84字节数据后备寄存器(大容量产品)
- 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。
- 在PC13管脚(当该管脚不用于侵入检测时)上输出RTC校准时钟，RTC闹钟脉冲或者秒脉冲

5.3 BKP功能描述

5.3.1 侵入检测

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

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

在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。*

5.3.2 RTC校准

为方便测量，RTC时钟可以经64分频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校准和如何提高精度，请看AN2604“STM32F101xx和STM32F103xx的RTC校准”

5.4 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

5.4.1 备份数据寄存器x(BKP_DRx) (x = 1 … 10)

地址偏移：0x04 到 0x28，0x40到0xBC

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。

5.4.2 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RTC时钟的周期。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220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/(220)ppm的比例减慢时钟。 RTC时钟可以被减慢0~121ppm。

5.4.3 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。

5.4.4 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)。

5.4.5 BKP寄存器映像

BKP寄存器是16位的可寻址寄存器。

表13 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_CR	保留																													TPAL	TPE
	复位值																														0	0
034h	RTC_CSR	保留																						TIF	TFE	保留				TPIE	CTI	CTE
	复位值																							0	0					0	0	0
038h	保留																																
03Ch	保留																																
040h	BKP_DR11	保留																D[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
044h	BKP_DR12	保留																D[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
048h	BKP_DR13	保留																D[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
04Ch	BKP_DR14	保留																D[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
050h	BKP_DR15	保留																D[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
054h	BKP_DR16	保留																D[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
058h	BKP_DR17	保留																D[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
05Ch	BKP_DR18	保留																D[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
060h	BKP_DR19	保留																D[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
064h	BKP_DR20	保留																D[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
068h	BKP_DR21	保留																D[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
06Ch	BKP_DR22	保留																D[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
070h	BKP_DR23	保留																D[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
074h	BKP_DR24	保留																D[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
078h	BKP_DR25	保留																D[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
07Ch	BKP_DR26	保留																D[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
080h	BKP_DR27	保留																D[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
084h	BKP_DR28	保留																D[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
088h	BKP_DR29	保留																D[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
08Ch	BKP_DR30	保留																D[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
090h	BKP_DR31	保留																D[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
094h	BKP_DR32	保留																D[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
098h	BKP_DR33	保留																D[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
09Ch	BKP_DR34	保留																D[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0A0h	BKP_DR35	保留																D[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0A4h	BKP_DR36	保留																D[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0A8h	BKP_DR37	保留																D[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0ACh	BKP_DR38	保留																D[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0B0h	BKP_DR39	保留																D[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0B4h	BKP_DR40	保留																D[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0B8h	BKP_DR41	保留																D[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0BCh	BKP_DR42	保留																D[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

有关寄存器的起始地址，请参考表1。

6 复位和时钟控制(RCC)

小容量产品是指闪存存储器容量在16K至32K字节之间的STM32F101xx，STM32F102xx和STM32F103xx微控制器。

中容量产品是指闪存存储器容量在64K至128K字节之间的STM32F101xx，STM32F102xx和STM32F103xx微控制器。

大容量产品是指闪存存储器容量在256K至512K字节之间的STM32F101xx和STM32F103xx微控制器。

除非特别说明，本章节描述的模块应用于整个STM32F10xxx微控制器系列。

6.1 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。

6.1.1 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识别复位事件来源。

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。

6.1.2 电源复位

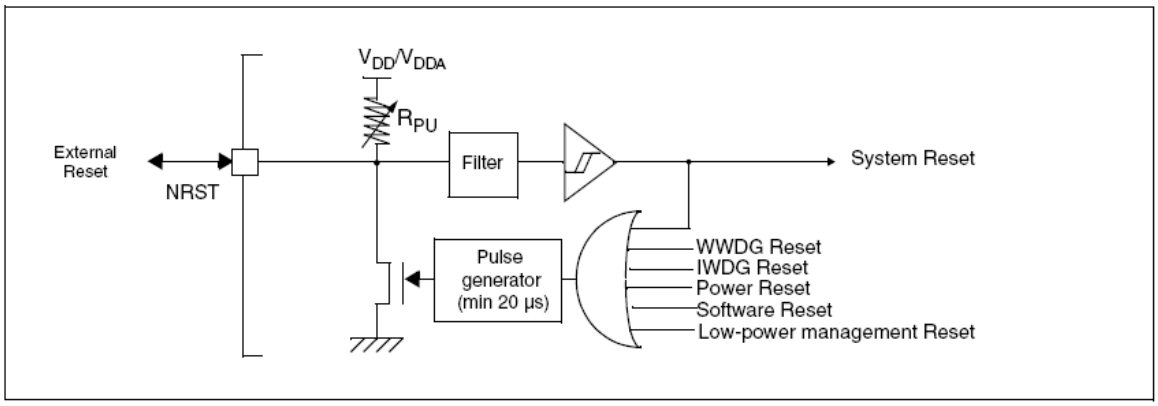
当以下事件中之一发生时，产生电源复位：

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_0004。更多细节，参阅表36。

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。

6.1.3 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。

6.2 时钟

三种不同的时钟源可被用来驱动系统时钟(SYSCLK)：

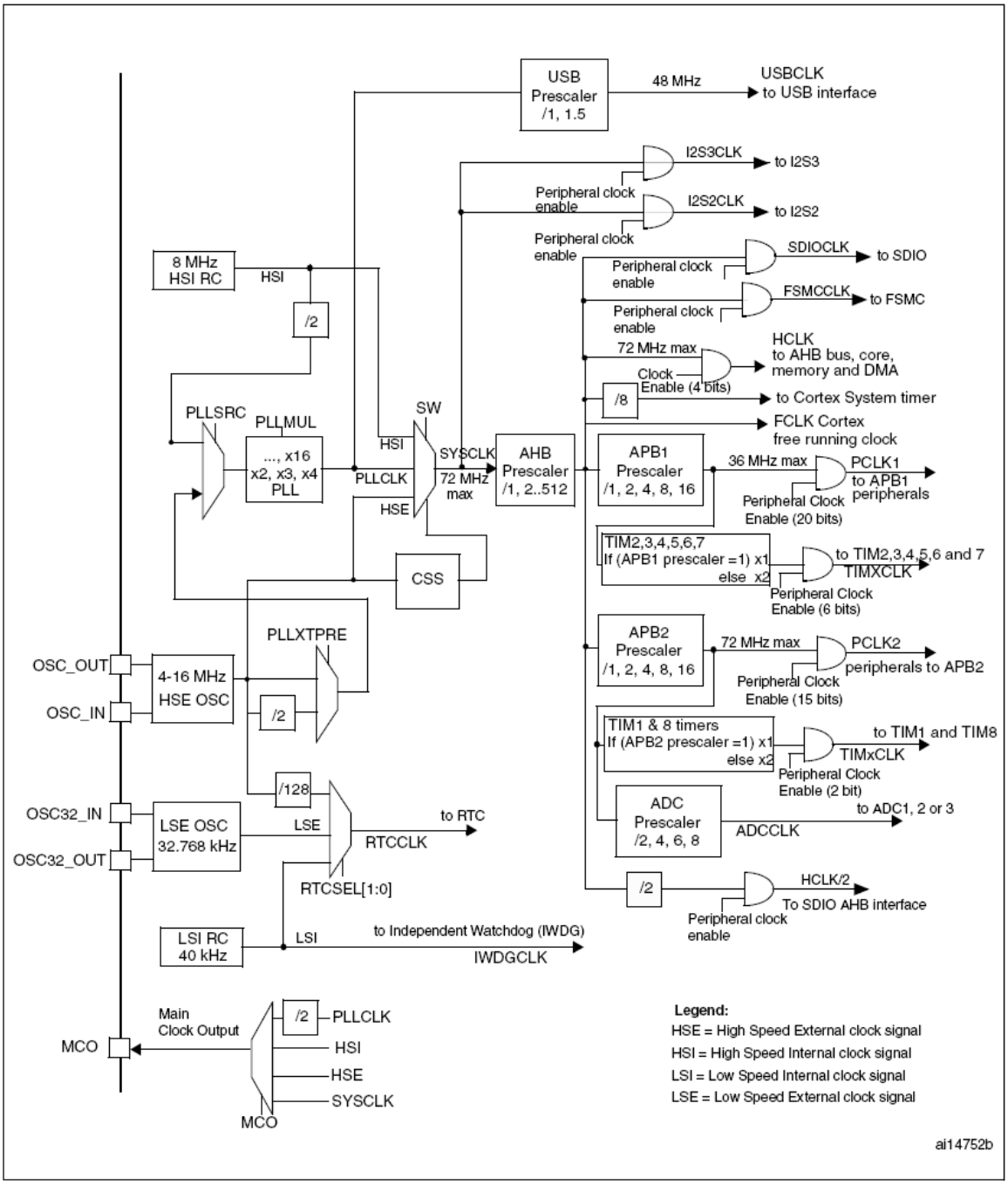
- HSI振荡器时钟
- HSE振荡器时钟
- PLL时钟

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。SDIO接口的时钟频率固定为HCLK/2。

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。

定时器时钟频率分配由硬件按以下2种情况自动设置：

1. 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。
2. 否则，定时器的时钟频率被设为与其相连的APB总线频率的2倍。

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

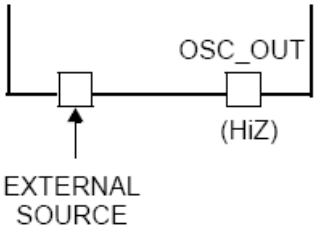
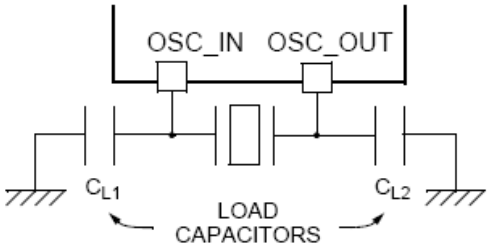
6.2.1 HSE时钟

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中允许产生中断，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。

6.2.2 HSI时钟

HSI时钟信号由内部8MHz的RC振荡器产生，可直接作为系统时钟或在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HSI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6.2.7节时钟安全系统。

6.2.3 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。

6.2.4 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。

6.2.5 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。

注意： *只有大容量产品可以进行LSI校准*

LSI校准

可以通过校准内部低速振荡器LSI来补偿其频率偏移，从而获得精度可接受的RTC时间基数，以及独立看门狗(IWDG)的超时时间(当这些外设以LSI为时钟源)。

校准可以通过使用TIM5的输入时钟(TIM5_CLK)测量LSI时钟频率实现。测量以HSE的精度为保证，软件可以通过调整RTC的20位预分频器来获得精确的RTC时钟基数，以及通过计算得到精确的独立看门狗(IWDG)的超时时间。

LSI校准步骤如下：

1. 打开TIM5，设置通道4为输入捕获模式；
2. 设置AFIO_MAPR的TIM5_CH4_IREMAP位为'1'，在内部把LSI连接到TIM5的通道4；
3. 通过TIM5的捕获/比较4事件或者中断来测量LSI时钟频率；
4. 根据测量结果和期望的RTC时间基数和独立看门狗的超时时间，设置20位预分频器。

6.2.6 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。

6.2.7 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，HSE振荡器被自动关闭，时钟失效事件将被送到高级定时器TIM1的刹车输入端，并产生时钟安全中断CSSI，允许软件完成营救操作。此CSSI中断连接到Cortex™-M3 的NMI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，(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。

6.2.8 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：详见6.2.5节LSI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)，则RTC状态不确定。

6.2.9 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。

6.2.10 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。

6.3 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。

6.3.1 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 由软件置'1'或清零以使能时钟监测器。 0：时钟监测器关闭； 1：如果外部4-25MHz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4-25MHz振荡器关闭的情况下，才能写入该位。 0：外部4-25MHz振荡器没有旁路； 1：外部4-25MHz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4-25MHz时钟已经稳定。在HSEON位清零后，该位需要6个外部4-25MHz时钟周期清零。 0：外部4-25MHz时钟没有就绪； 1：外部4-25MHz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4-25MHz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±1%；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8MHz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4-25MHz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。

6.3.2 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31:27	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(SYSCLK)输出； 101：内部8MHz的RC振荡器时钟输出； 110：外部4-25MHz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至MCO管脚时，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 由软件置'1'或清'0'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　　1000：PLL 10倍频输出 0001：PLL 3倍频输出　　1001：PLL 11倍频输出 0010：PLL 4倍频输出　　1010：PLL 12倍频输出 0011：PLL 5倍频输出　　1011：PLL 13倍频输出 0100：PLL 6倍频输出　　1100：PLL 14倍频输出 0101：PLL 7倍频输出　　1101：PLL 15倍频输出 0110：PLL 8倍频输出　　1110：PLL 16倍频输出 0111：PLL 9倍频输出　　1111：PLL 16倍频输出
位17	**PLLXTPRE**：HSE分频器作为PLL输入 由软件置'1'或清'0'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 由软件置'1'或清'0'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 由软件置'1'或清'0'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） 由软件置'1'或清'0'来控制高速APB2时钟(PCLK2)的预分频系数。 0xx：HCLK不分频 100：HCLK 2分频 101：HCLK 4分频 110：HCLK 8分频 111：HCLK 16分频

位10:8	**PPRE1**：低速APB预分频（APB1） 由软件置'1'或清'0'来控制低速APB1时钟(PCLK1)的预分频系数。 注意：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 由软件置'1'或清'0'来控制AHB时钟的预分频系数。 0xxx：SYSCLK不分频 1000：SYSCLK 2分频　　1100：SYSCLK 64分频 1001：SYSCLK 4分频　　1101：SYSCLK 128分频 1010：SYSCLK 8分频　　1110：SYSCLK 256分频 1011：SYSCLK 16分频　　1111：SYSCLK 512分频 注意：当AHB时钟的预分频系数大于1时，必须开启预取缓冲器。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'或清'0'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 由软件置'1'或清'0'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。

6.3.3 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**：清除时钟安全系统中断 由软件置'1'来清除CSSF安全系统中断标志位CSSF。 0：无作用； 1：清除CSSF安全系统中断标志位。
位22:21	保留，始终读为0。
位20	**PLLRDYC**：清除PLL就绪中断 由软件置'1'来清除PLL就绪中断标志位PLLRDYF。 0：无作用； 1：清除PLL就绪中断标志位PLLRDYF。
位19	**HSERDYC**：清除HSE就绪中断 由软件置'1'来清除HSE就绪中断标志位HSERDYF。 0：无作用； 1：清除HSE就绪中断标志位HSERDYF。
位18	**HSIRDYC**： 清除HSI就绪中断 由软件置'1'来清除HSI就绪中断标志位HSIRDYF。 0：无作用； 1：清除HSI就绪中断标志位HSIRDYF。
位17	**LSERDYC**：清除LSE就绪中断 由软件置'1'来清除LSE就绪中断标志位LSERDYF。 0：无作用； 1：清除LSE就绪中断标志位LSERDYF。
位16	**LSIRDYC**：清除LSI就绪中断 由软件置'1'来清除LSI就绪中断标志位LSIRDYF。 0：无作用； 1：清除LSI就绪中断标志位LSIRDYF。
位15:13	保留，始终读为0。
位12	**PLLRDYIE**：PLL就绪中断使能 由软件置'1'或清'0'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'0'来使能或关闭外部4-25MHz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'0'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'0'来使能或关闭外部32kHz 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'0'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 在外部4-25MHz振荡器时钟出现故障时，由硬件置'1'。 由软件通过置'1' CSSC位来清除。 0：无HSE时钟失效产生的安全系统中断； 1：HSE时钟失效导致了时钟安全系统中断。
位6:5	保留，始终读为0。
位4	**PLLRDYF**：PLL就绪中断标志 在PLL就绪且PLLRDYIE位被置'1'时，由硬件置'1'。 由软件通过置'1' PLLRDYC位来清除。 0：无PLL上锁产生的时钟就绪中断； 1：PLL上锁导致时钟就绪中断。
位3	**HSERDYF**：HSE就绪中断标志 在外部低速时钟就绪且HSERDYIE位被置'1'时，由硬件置'1'。 由软件通过置'1' HSERDYC位来清除。 0：无外部4-25MHz振荡器产生的时钟就绪中断； 1：外部4-25MHz振荡器导致时钟就绪中断。
位2	**HSIRDYF**：HSI就绪中断标志 在内部高速时钟就绪且HSIRDYIE位被置'1'时，由硬件置'1'。 由软件通过置'1' HSIRDYC位来清除。 0：无内部8MHz RC振荡器产生的时钟就绪中断； 1：内部8MHz RC振荡器导致时钟就绪中断。
位1	**LSERDYF**：LSE就绪中断标志 在外部低速时钟就绪且LSERDYIE位被置'1'时，由硬件置'1'。 由软件通过置'1' LSERDYC位来清除。 0：无外部32kHz振荡器产生的时钟就绪中断； 1：外部32kHz振荡器导致时钟就绪中断。
位0	**LSIRDYF**：LSI就绪中断标志 在内部低速时钟就绪且LSIRDYIE位被置'1'时，由硬件置'1'。 由软件通过置'1' LSIRDYC位来清除。 0：无内部40kHz RC振荡器产生的时钟就绪中断； 1：内部40kHz RC振荡器导致时钟就绪中断。

6.3.4 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
ADC3 RST	USART1 RST	TIM8 RST	SPI1 RST	TIM1 RST	ADC2 RST	ADC1 RST	IOPG RST	IOPF RST	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	rw	rw	rw	res	rw

位31:16	保留，始终读为0。
位15	ADC3RST：ADC3接口复位 由软件置'1'或清'0' 0：无作用； 1：复位ADC3接口。

位14	USART1RST：USART1复位 由软件置'1'或清'0' 0：无作用； 1：复位USART1。
位13	TIM8RST TIM8定时器复位 由软件置'1'或清'0' 0：无作用； 1：复位TIM8定时器。
位12	SPI1RST：SPI1复位 由软件置'1'或清'0' 0：无作用； 1：复位SPI1。
位11	TIM1RST：TIM1定时器复位 由软件置'1'或清'0' 0：无作用； 1：复位TIM1定时器。
位10	ADC2RST：ADC2接口复位 由软件置'1'或清'0' 0：无作用； 1：复位ADC2接口。
位9	ADC1RST：ADC1接口复位 由软件置'1'或清'0' 0：无作用； 1：复位ADC1接口。
位8	IOPGRST IO端口G复位 由软件置'1'或清'0' 0：无作用； 1：复位IO端口G。
位7	IOPFRST：IO端口F复位 由软件置'1'或清'0' 0：无作用； 1：复位IO端口F。
位6	IOPERST：IO端口E复位 由软件置'1'或清'0' 0：无作用； 1：复位IO端口E。
位5	IOPDRST：IO端口D复位 由软件置'1'或清'0' 0：无作用； 1：复位IO端口D。
位4	IOPCRST：IO端口C复位 由软件置'1'或清'0' 0：无作用； 1：复位IO端口C。
位3	IOPBRST：IO端口B复位 由软件置'1'或清'0' 0：无作用； 1：复位IO端口B。

位2	IOPARST：IO端口A复位 由软件置'1'或清'0' 0：无作用； 1：复位IO端口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'0' 0：无作用； 1：复位辅助功能。

6.3.5 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		DACRST	PWR RST	BKP RST	保留	CAN RST	保留	USB RST	I2C2 RST	I2C1 RST	UART5RST	UART4RST	USART3 RST	USART2 RST	保留
		rw	rw	rw		rw		rw	rw	rw	rw	rw	rw	rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPI3 RST	SPI2 RST	保留		WWDG RST	保留					TIM7 RST	TIM6 RST	TIM5 RST	TIM4 RST	TIM3 RST	TIM2 RST
rw	rw			rw						rw	rw	rw	rw	rw	rw

位31:30	保留，始终读为0。
位29	**DACRST**：DAC接口复位 由软件置'1'或清'0' 0：无作用； 1：复位DAC接口。
位28	**PWRRST**：电源接口复位 由软件置'1'或清'0' 0：无作用； 1：复位电源接口。
位27	**BKPRST**：备份接口复位 由软件置'1'或清'0' 0：无作用； 1：复位备份接口。
位26	保留，始终读为0。
位25	**CANRST**：CAN复位 由软件置'1'或清'0' 0：无作用； 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'0' 0：无作用； 1：复位USB。

位22	**I2C2RST**：I2C 2复位 由软件置'1'或清'0' 0：无作用； 1：复位I2C 2。
位21	**I2C1RST**：I2C 1复位 由软件置'1'或清'0' 0：无作用； 1：复位I2C 1。
位20	**UART5RST**：UART5复位 由软件置'1'或清'0' 0：无作用； 1：复位UART5。
位19	**UART4RST**：UART4复位 由软件置'1'或清'0' 0：无作用； 1：复位UART4。
位18	**USART3RST**：USART3复位 由软件置'1'或清'0' 0：无作用； 1：复位USART3。
位17	**USART2RST**：USART2复位 由软件置'1'或清'0' 0：无作用； 1：复位USART2。
位16	保留，始终读为0。
位15	**SPI3RST** SPI3 复位 由软件置'1'或清'0' 0：无作用； 1：复位SPI3。
位14	**SPI2RST**：SPI2复位 由软件置'1'或清'0' 0：无作用； 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'0' 0：无作用； 1：复位窗口看门狗。
位10:6	保留，始终读为0。
位5	**TIM7RST**：定时器7复位 由软件置'1'或清'0' 0：无作用； 1：复位TIM7定时器。
位4	**TIM6RST**：定时器6复位 由软件置'1'或清'0' 0：无作用； 1：复位TIM6定时器。

位3	**TIM5RST**：定时器5复位 由软件置'1'或清'0' 0：无作用； 1：复位TIM5定时器。
位2	**TIM4RST**：定时器4复位 由软件置'1'或清'0' 0：无作用； 1：复位TIM4定时器。
位1	**TIM3RST**：定时器3复位 由软件置'1'或清'0' 0：无作用； 1：复位TIM3定时器。
位0	**TIM2RST**：定时器2复位 由软件置'1'或清'0' 0：无作用； 1：复位TIM2定时器。

6.3.6 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					SDIOEN	保留	FSMCEN	保留	CRCEN	保留	FLITFEN	保留	SRAMEN	DMA2EN	DMA1EN
					rw		rw		rw		rw		rw	rw	rw

位31:11	保留，始终读为0。
位10	**SDIOEN**：SDIO时钟使能 由软件置'1'或清'0'。 0：SDIO时钟关闭； 1：SDIO时钟开启。
位9	保留，始终读为0。
位8	**FSMCEN**：FSMC时钟使能 由软件置'1'或清'0'。 0：FSMC时钟关闭； 1：FSMC时钟开启。
位7	保留，始终读为0。
位6	**CRCEN**：CRC时钟使能 由软件置'1'或清'0'。 0：CRC时钟关闭； 1：CRC时钟开启。
位5	保留，始终读为0。

位4	**FLITFEN**：闪存接口电路时钟使能 由软件置'1'或清'0'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 由软件置'1'或清'0'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	**DMA2EN**：DMA2时钟使能 由软件置'1'或清'0'。 0：DMA2时钟关闭； 1：DMA2时钟开启。
位0	**DMA1EN**：DMA1时钟使能 由软件置'1'或清'0'。 0：DMA1时钟关闭； 1：DMA1时钟开启。

6.3.7 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总线上的外设被访问时，将插入等待状态直到外设访问结束。

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
ADC3 EN	USART1 EN	TIM8 EN	SPI1 EN	TIM1 EN	ADC2 EN	ADC1 EN	IOPG EN	IOPF EN	IOPE EN	IOPD EN	IOPC EN	IOPB EN	IOPA EN	保留	AFIO EN
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	**ADC3EN**：ADC3接口时钟使能 由软件置'1'或清'0' 0：ADC3接口时钟关闭； 1：ADC3接口时钟开启。
位14	**USART1EN**：USART1时钟使能 由软件置'1'或清'0' 0：USART1时钟关闭； 1：USART1时钟开启。
位13	**TIM8EN**：TIM8定时器时钟使能 由软件置'1'或清'0' 0：TIM8定时器时钟关闭； 1：TIM8定时器时钟开启。
位12	**SPI1EN**：SPI1时钟使能 由软件置'1'或清'0' 0：SPI1时钟关闭； 1：SPI1时钟开启。

位11	**TIM1EN**：TIM1定时器时钟使能 由软件置'1'或清'0' 0：TIM1定时器时钟关闭； 1：TIM1定时器时钟开启。
位10	**ADC2EN**：ADC2接口时钟使能 由软件置'1'或清'0' 0：ADC2接口时钟关闭； 1：ADC2接口时钟开启。
位9	**ADC1EN**：ADC1接口时钟使能 由软件置'1'或清'0' 0：ADC1接口时钟关闭； 1：ADC1接口时钟开启。
位8	**IOPGEN**：IO端口G时钟使能 由软件置'1'或清'0' 0：IO端口G时钟关闭； 1：IO端口G时钟开启。
位7	**IOPFEN**：IO端口F时钟使能 由软件置'1'或清'0' 0：IO端口F时钟关闭； 1：IO端口F时钟开启。
位6	**IOPEEN**：IO端口E时钟使能 由软件置'1'或清'0' 0：IO端口E时钟关闭； 1：IO端口E时钟开启。
位5	**IOPDEN**：IO端口D时钟使能 由软件置'1'或清'0' 0：IO端口D时钟关闭； 1：IO端口D时钟开启。
位4	**IOPCEN**：IO端口C时钟使能 由软件置'1'或清'0' 0：IO端口C时钟关闭； 1：IO端口C时钟开启。
位3	**IOPBEN**：IO端口B时钟使能 由软件置'1'或清'0' 0：IO端口B时钟关闭； 1：IO端口B时钟开启。
位2	**IOPAEN**：IO端口A时钟使能 由软件置'1'或清'0' 0：IO端口A时钟关闭； 1：IO端口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 由软件置'1'或清'0' 0：辅助功能IO时钟关闭； 1：辅助功能IO时钟开启。

6.3.8 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总线上的外设被访问时，将插入等待状态直到外设访问结束。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留		DACEN	PWR EN	BKP EN	保留	CAN EN	保留	USB EN	I2C2 EN	I2C1 EN	UART5 EN	UART4 EN	USART3 EN	USART2 EN	保留
		rw	rw	rw		rw		rw	rw	rw	rw	rw	rw	rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPI3 EN	SPI2 EN	保留		WWDG EN	保留					TIM7 EN	TIM6 EN	TIM5 EN	TIM4 EN	TIM3 EN	TIM2 EN
rw	rw			rw						rw	rw	rw	rw	rw	rw

位31:30	保留，始终读为0。
位29	**DACEN**: DAC接口时钟使能 由软件置'1'或清'0' 0：DAC接口时钟关闭； 1：DAC接口时钟开启。
位28	**PWREN**：电源接口时钟使能 由软件置'1'或清'0' 0：电源接口时钟关闭； 1：电源接口时钟开启。
位27	**BKPEN**：备份接口时钟使能 由软件置'1'或清'0' 0：备份接口时钟关闭； 1：备份接口时钟开启。
位26	保留，始终读为0。
位25	**CANEN**：CAN时钟使能 由软件置'1'或清'0' 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 由软件置'1'或清'0' 0：USB时钟关闭； 1：USB时钟开启。
位22	**I2C2EN**：I2C 2时钟使能 由软件置'1'或清'0' 0：I2C 2时钟关闭； 1：I2C 2时钟开启。
位21	**I2C1EN**：I2C 1时钟使能 由软件置'1'或清'0' 0：I2C 1时钟关闭； 1：I2C 1时钟开启。
位20	**UART5EN**：UART5时钟使能 由软件置'1'或清'0' 0：UART5时钟关闭； 1：UART5时钟开启。

位19	**UART4EN**：UART4时钟使能 由软件置'1'或清'0' 0：UART4时钟关闭； 1：UART4时钟开启。
位18	**USART3EN**：USART3时钟使能 由软件置'1'或清'0' 0：USART3时钟关闭； 1：USART3时钟开启。
位17	**USART2EN**：USART2时钟使能 由软件置'1'或清'0' 0：USART2时钟关闭； 1：USART2时钟开启。
位16	保留，始终读为0。
位15	**SPI3EN**：SPI 3时钟使能 由软件置'1'或清'0' 0：SPI 3时钟关闭； 1：SPI 3时钟开启。
位14	**SPI2EN**：SPI 2时钟使能 由软件置'1'或清'0' 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 由软件置'1'或清'0' 0：窗口看门狗时钟关闭； 1：窗口看门狗时钟开启。
位10:6	保留，始终读为0。
位5	**TIM7EN**：定时器7时钟使能 由软件置'1'或清'0' 0：定时器7时钟关闭； 1：定时器7时钟开启。
位4	**TIM6EN**：定时器6时钟使能 由软件置'1'或清'0' 0：定时器6时钟关闭； 1：定时器6时钟开启。
位3	**TIM5EN**：定时器5时钟使能 由软件置'1'或清'0' 0：定时器5时钟关闭； 1：定时器5时钟开启。
位2	**TIM4EN**：定时器4时钟使能 由软件置'1'或清'0' 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 由软件置'1'或清'0' 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 由软件置'1'或清'0' 0：定时器2时钟关闭； 1：定时器2时钟开启。

6.3.9 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5.1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	r	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'0' 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'0' 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'0'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'0'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'0' 0：外部32kHz振荡器关闭； 1：外部32kHz振荡器开启。

6.3.10 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 在低功耗管理复位发生时由硬件置'1'；由软件通过写RMVF位清除。 0：无低功耗管理复位发生; 1：发生低功耗管理复位. 关于低功耗管理复位的详细信息，请参考低功耗管理复位章节。
位30	**WWDGRSTF**：窗口看门狗复位标志 在窗口看门狗复位发生时由硬件置'1'；由软件通过写RMVF位清除。 0：无窗口看门狗复位发生； 1：发生窗口看门狗复位。
位29	**IWDGRSTF**：独立看门狗复位标志 在独立看门狗复位发生在V_{DD}区域时由硬件置'1'；由软件通过写RMVF位清除。 0：无独立看门狗复位发生； 1：发生独立看门狗复位。
位28	**SFTRSTF**：软件复位标志 在软件复位发生时由硬件置'1'；由软件通过写RMVF位清除。 0：无软件复位发生； 1：发生软件复位。
位27	**PORRSTF**：上电/掉电复位标志 在上电/掉电复位发生时由硬件置'1'；由软件通过写RMVF位清除。 0：无上电/掉电复位发生； 1：发生上电/掉电复位。
位26	**PINRSTF**：NRST管脚复位标志 在NRST管脚复位发生时由硬件置'1'；由软件通过写RMVF位清除。 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'来清除复位标志。 0：无作用； 1：清除复位标志。
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'0'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：内部40kHz RC振荡器时钟未就绪； 1：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'0'。 0：内部40kHz RC振荡器关闭； 1：内部40kHz RC振荡器开启。

6.3.11 RCC寄存器地址映像

下表列出了RCC寄存器的映像和复位值。

表14 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			PLLRDYIH	HSERDYIH	HSIRDYIH	LSERDYIH	LSIRDYIH	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																ADC3RST	USART1RST	TIM8RST	SPIRST	TIM1RST	ADC2RST	ADC1RST	IOPGRST	IOPFRST	IOPERST	IOPDRST	IOPCRST	IOPBRST	IOPARST	保留	AFIORST
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0
010h	RCC_APB1RSTR	保留		DACRST	PWRRST	BKPRST	保留	CANRST	保留	USBRST	I2C2RST	I2C1RST	UART5RST	UART4RST	USART3RST	USART2RST	保留	SPI3RST	SPI2RST	保留		WWDGRST	保留						TIM7RST	TIM6RST	TIM5RST	TIM4RST	TIM3RST	TIM2RST
	复位值			0	0	0		0		0	0	0	0	0	0	0		0	0			0							0	0	0	0	0	0
014h	RCC_APB1RSTR	保留																					SDIOEN	保留	FSMCEN	保留	CRCEN	保留	FLITFEN	保留	SRAMEN	DMA2EN	DMA1EN
	复位值																						0		0		0		1		1	0	0
018h	RCC_APB2ENR	保留																ADC3EN	USART1EN	TIM8RST	SPIREN	TIM1EN	ADC2EN	ADC1EN	IOPGEN	IOPFEN	IOPEEN	IOPDEN	IOPCEN	IOPBEN	IOPAEN	保留	AFIOEN
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0
01Ch	RCC_APB1ENR	保留		DACRST	PWREN	BKPEN	保留	CANEN	保留	USBEN	I2C2EN	I2C1EN	UART5EN	UART4EN	USART3EN	USART2EN	保留	SPI3RST	SPI2EN	保留		WWDGEN	保留						TIM7EN	TIM6EN	TIM5EN	TIM4EN	TIM3EN	TIM2EN
	复位值			0	0	0		0		0	0	0	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

7 通用和复用功能I/O(GPIO和AFIO)

小容量产品是指闪存存储器容量在16K至32K字节之间的STM32F101xx、STM32F102xx和STM32F103xx微控制器。

中容量产品是指闪存存储器容量在64K至128K字节之间的STM32F101xx、STM32F102xx和STM32F103xx微控制器。

大容量产品是指闪存存储器容量在256K至512K字节之间的STM32F101xx和STM32F103xx微控制器。

除非特别说明，本章节描述的模块应用于整个STM32F10xxx微控制器系列。

7.1 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。

下图给出了一个I/O端口位的基本结构。

图9 I/O端口位的基本结构

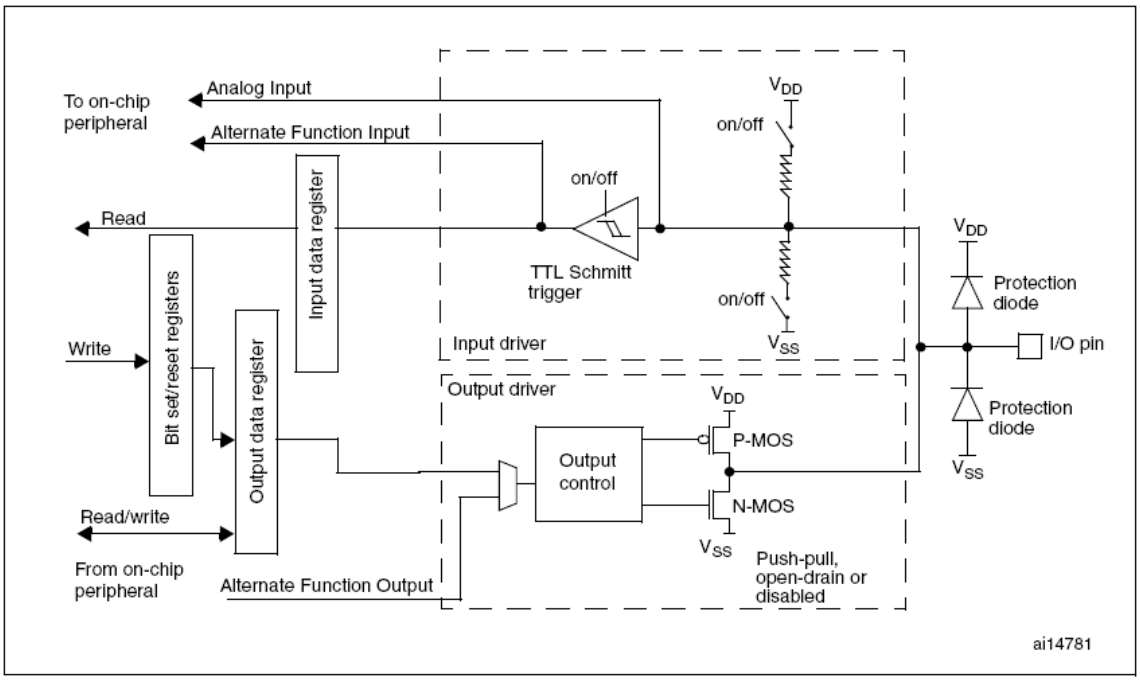
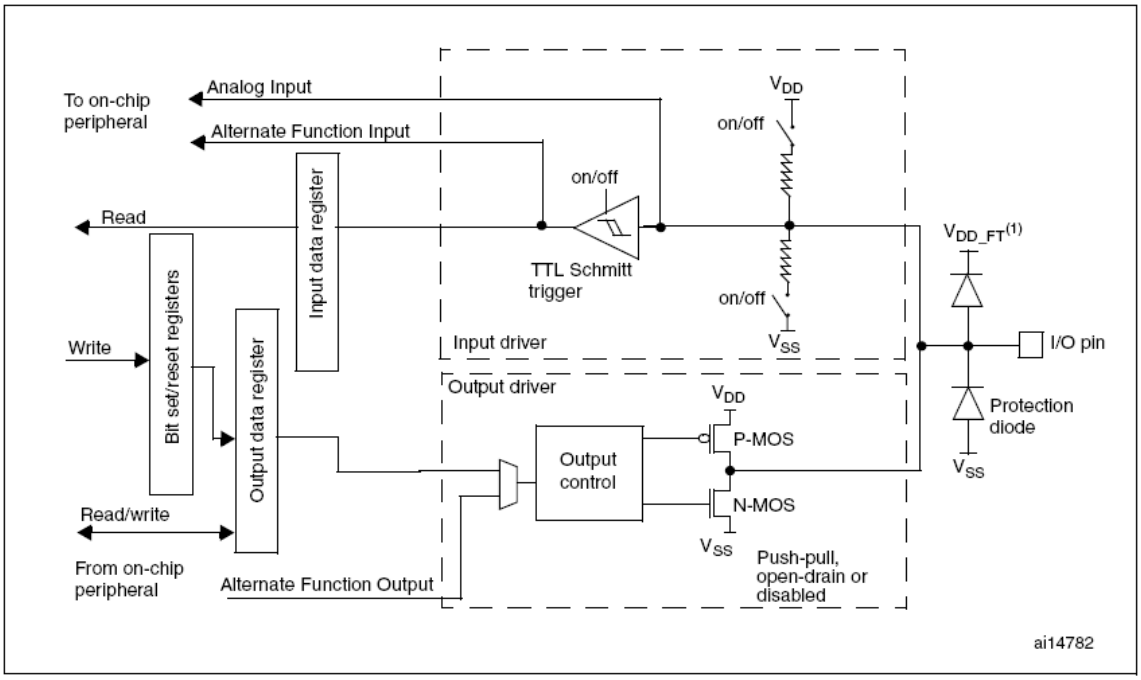


图10 5伏兼容I/O端口位的基本结构



(1) V_{DD_FT} 对5伏兼容I/O脚是特殊的，它与V_{DD}不同

表15 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16		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16 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

7.1.1 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被断开。

7.1.2 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。

7.1.3 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：

- 8.2节：外部中断/事件控制器
- 8.2.3节：唤醒事件管理

7.1.4 复用功能(AF)

使用默认复用功能前必须对端口位配置寄存器编程。

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

注意：也可以通过软件来模拟复用功能输入管脚，这种模拟可以通过对GPIO控制器编程来实现。此时，端口应当被设置为复用功能输出模式。显然，这时相应的管脚不再由外部驱动，而是通过GPIO控制器由软件来驱动。

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

如果把端口配置成复用输出功能，则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。

7.1.5 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。

7.1.6 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。

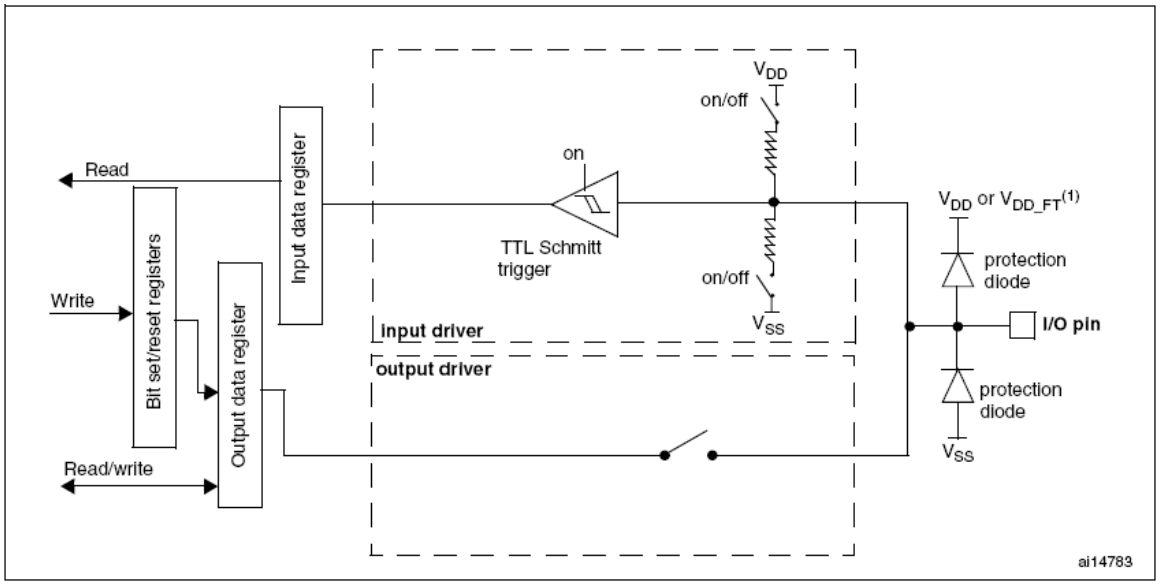
7.1.7 输入配置

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

下图给出了I/O端口位的输入配置

图11 输入浮空/上拉/下拉配置



(1) V_{DD_FT} 对5伏兼容I/O脚是特殊的，它与V_{DD}不同

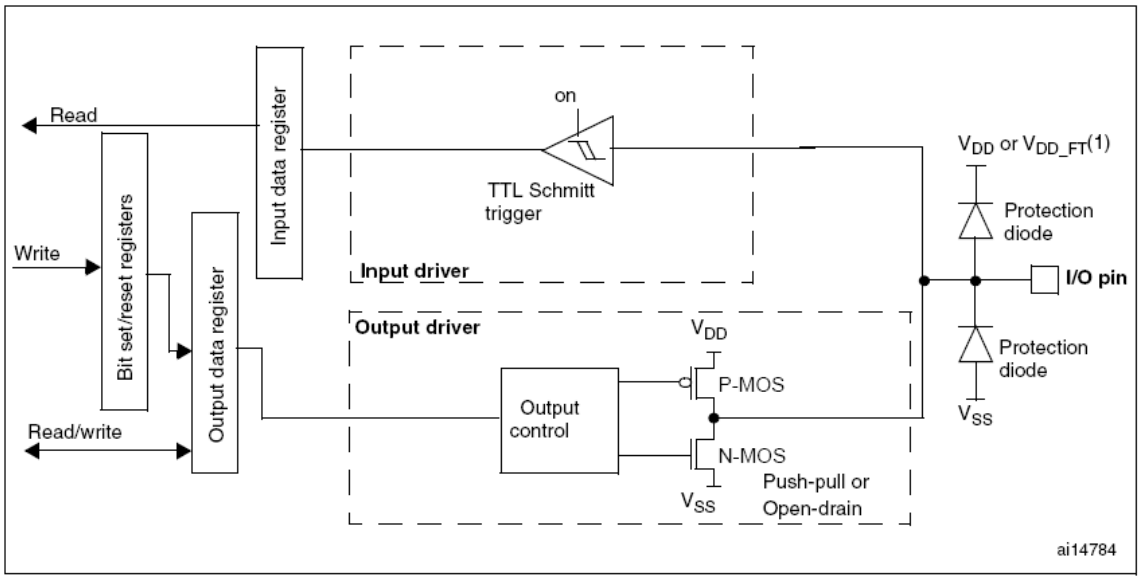
7.1.8 输出配置

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。

下图给出了I/O端口位的输出配置。

图12 输出配置



(1) V_{DD_FT} 对 5 伏兼容 I/O 脚是特殊的，它与 V_{DD} 不同

7.1.9 复用功能配置

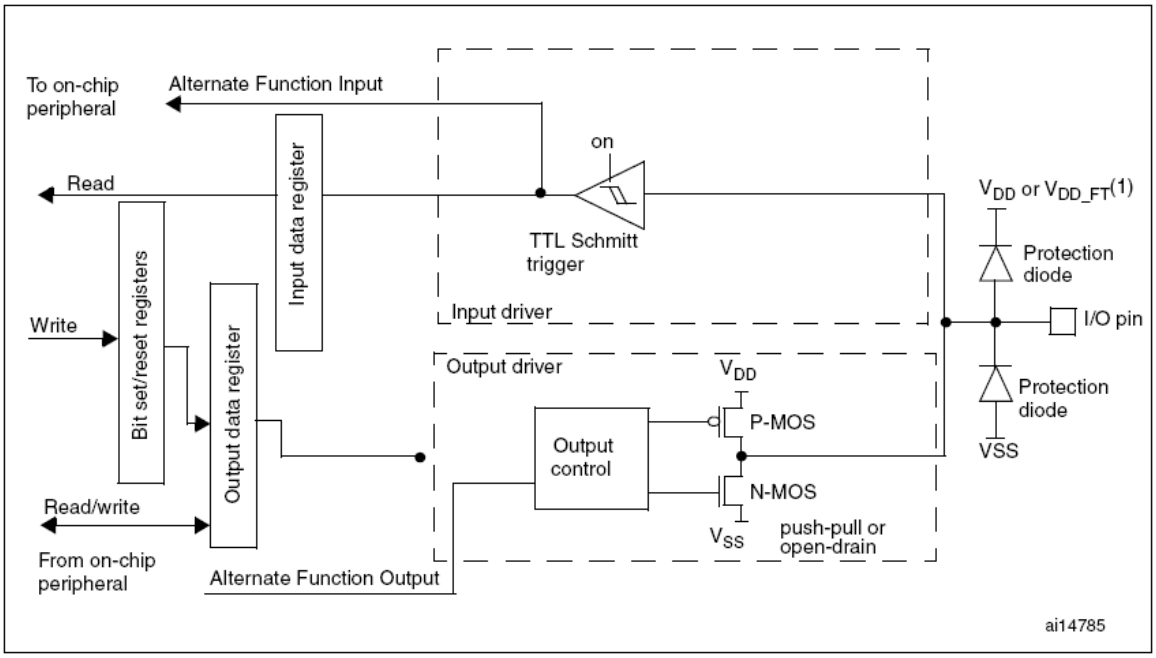
当I/O端口被配置为复用功能时：

- 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

下图示出了I/O端口位的复用功能配置。详见7.4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) V_{DD_FT} 对5伏兼容I/O脚是特殊的，它与V_{DD}不同

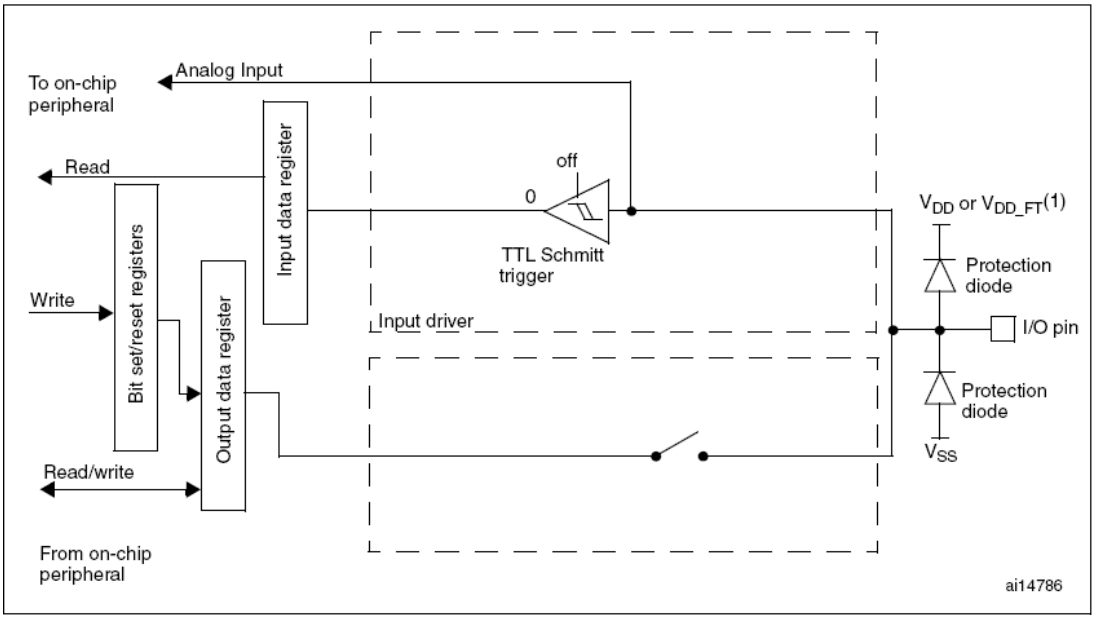
7.1.10 模拟输入配置

当I/O端口被配置为模拟输入配置时：

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

下图示出了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) V_{DD_FT} 对5伏兼容I/O脚是特殊的，它与V_{DD}不同

7.2 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。

7.2.1 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 = 0…7) 软件通过这些位配置相应的I/O端口，请参考表15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29:28 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15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

7.2.2 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15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15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

7.2.3 端口输入数据寄存器(GPIOx_IDR) (x=A..E)

地址偏移：0x08

复位值：0x0000 XXXX

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。

7.2.4 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。

7.2.5 端口位设置/清除寄存器(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

7.2.6 端口位清除寄存器(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

7.2.7 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

7.3 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。

7.3.1 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GPIO口功能；

2．参见第4.1.2节有关I/O口使用的限制

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

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

这个重映射只适用于36、48和64脚的封装(100脚和144脚的封装上有单独的PD0和PD1的引脚，不必重映射)

注：外部中断/事件功能没有被重映射。在36、48和64脚的封装上，PD0和PD1不能用来产生外部中断/事件。

7.3.3 CAN复用功能重映射

CAN信号可以被映射到端口A、端口B或端口D上，如下表所示。对于端口D，在36、48和64脚的封装上没有重映射功能。

表17 CAN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"(2)
CAN_RX	PA11	PB8	PD0
CAN_TX	PA12	PB9	PD1

1. 重映射不适用于36脚的封装
2. 当PD0和PD1没有被重映射到OSC_IN和OSC_OUT时，重映射功能只适用于100脚和144脚的封装上。

7.3.4 JTAG/SWD复用功能重映射

调试接口信号被映射到GPIO端口上，如下表所示。

表18 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。参见下表。

表19 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。

7.3.5 ADC复用功能重映射

参阅AF重映射和除错IO配置寄存器(AFIO_MAPR)。

表20 ADC1外部触发注入转换复用功能重映射(1)

复用功能	ADC1_ETRGINJ_REMAP = 0	ADC1_ETRGINJ_REMAP = 0
ADC1外部触发注入转换	ADC1外部触发注入转换与EXTI15相连	ADC1外部触发注入转换与TIM8_CH4相连

1. 重映射仅存在于大容量产品

表21 ADC1外部触发规则转换复用功能重映射(1)

复用功能	ADC1_ETRGINJ_REMAP = 0	ADC1_ETRGINJ_REMAP = 0
ADC1外部触发规则转换	ADC1外部触发注入转换与EXTI11相连	ADC1外部触发注入转换与TIM8_TRGO相连

1. 重映射仅存在于大容量产品

表22 ADC2外部触发注入转换复用功能重映射(1)

复用功能	ADC1_ETRGINJ_REMAP = 0	ADC1_ETRGINJ_REMAP = 0
ADC2外部触发注入转换	ADC1外部触发注入转换与EXTI15相连	ADC1外部触发注入转换与TIM8_CH4相连

1. 重映射仅存在于大容量产品

表23 ADC2外部触发规则转换复用功能重映射(1)

复用功能	ADC1_ETRGINJ_REMAP = 0	ADC1_ETRGINJ_REMAP = 0
ADC2外部触发规则转换	ADC1外部触发注入转换与EXTI11相连	ADC1外部触发注入转换与TIM8_TRGO相连

1. 重映射仅存在于大容量产品

7.3.6 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列在表26~表28。

参见复用重映射和调试I/O配置寄存器(AFIO_MAPR)

表24 定时器5复用功能重映像(1)

复用功能	TIM5CH4_IREMAP = 0	TIM5CH4_IREMAP = 1
TIM5_CH4	TIM5的通道4连至PA3	LSI内部时钟连至TIM5_CH4的输入作为校准使用

1. 重映像只适用于大容量产品

表25 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26 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27 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 (2)	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
2. TIM_CH1 和 TIM_ETR 共享一个管脚，但不能同时使用(这也正是我们在此处使用表达式 TIM2_CH1_ETR 的原因)

表28 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

7.3.7 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29 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30 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31 USART1重映像

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

7.3.8 I^2C 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32 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

7.3.9 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33 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

7.4 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。

注意：对寄存器AFIO_EVCR,AFIO_MAPR和AFIO_EXTICRX进行读写操作前，应当首先打开AFIO的时钟。参考章节6.3.7 APB2外设时钟使能寄存器(RCC_APB2ENR)。

7.4.1 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 注：端口PF和PG并不具有EVENTPUT能力 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

7.4.2 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			ADC2_ETRGREG_REMAP	ADC2_ETRGINJ_REMAP	ADC1_ETRGREG_REMAP	ADC1_ETRGINJ_REMAP	TIM5CH4_IREMAP
					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23:21	保留。
位20	**ADC2_ETRGREG_REMAP**：ADC2规则转换外部触发重映射 该位可由软件置'1'或置'0'。它控制与ADC2规则转换外部触发相连的触发输入映像。当该位置'0'时，ADC2规则转换外部触发与EXTI11相连；当该位置'1'时，ADC2规则转换外部触发与TIM8_TRGO相连。
位19	**ADC2_ETRGINJ_REMAP**：ADC2注入转换外部触发重映射 该位可由软件置'1'或置'0'。它控制与ADC2注入转换外部触发相连的触发输入映像。当该位置'0'时，ADC2注入转换外部触发与EXTI15相连；当该位置'1'时，ADC2注入转换外部触发与TIM8通道4相连。
位18	**ADC1_ETRGREG_REMAP**：ADC1规则转换外部触发重映射 该位可由软件置'1'或置'0'。它控制与ADC2规则转换外部触发相连的触发输入映像。当该位置'0'时，ADC1规则转换外部触发与EXTI11相连；当该位置'1'时，ADC1规则转换外部触发与TIM8_TRGO相连。
位17	**ADC1_ETRGINJ_REMAP**：ADC1注入转换外部触发重映射 该位可由软件置'1'或置'0'。它控制与ADC2注入转换外部触发相连的触发输入映像。当该位置'0'时，ADC2注入转换外部触发与EXTI15相连；当该位置'1'时，ADC1注入转换外部触发与TIM8通道4相连。
位16	**TIM5CH4_IREMAP**：TIM5通道4内部重映射 该位可由软件置'1'或置'0'。它控制TIM5通道4内部映像。当该位置'0'时，TIM5_CH4与PA3相连；当该位置'1'时，LSI内部振荡器与TIM5_CH4相连，目的是对其进行校准。
位15	**PD01_REMAP**：端口D0/端口D1映像到OSC_IN/OSC_OUT 该位可由软件置'1'或置'0'。它控制PD0和PD1的GPIO功能映像。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 这些位可由软件置'1'或置'0'，控制复用功能CAN_RX和CAN_TX的重映像。 00：CAN_RX映像到PA11，CAN_TX映像到PA12； 01：未用组合； 10：CAN_RX映像到PB8，CAN_TX映像到PB9(不能用于36脚的封装)； 11：CAN_RX映像到PD0，CAN_TX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 该位可由软件置'1'或置'0'，控制将TIM4的通道1-4映射到GPIO端口上。 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	**TIM3_REMAP[1:0]**：定时器3的重映像 这些位可由软件置'1'或置'0'，控制定时器3的通道1至4在GPIO端口的映像。 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 这些位可由软件置'1'或置'0'，控制定时器2的通道1至4和外部触发(ETR)在GPIO端口的映像。 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 这些位可由软件置'1'或置'0'，控制定时器1的通道1至4、1N至3N、外部触发(ETR)和断线输入(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 这些位可由软件置'1'或置'0'，控制USART3的CTS、RTS、CK、TX和RX复用功能在GPIO端口的映像。 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 这些位可由软件置'1'或置'0'，控制USART2的CTS、RTS、CK、TX和RX复用功能在GPIO端口的映像。 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 该位可由软件置'1'或置'0'，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 该位可由软件置'1'或置'0'，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 该位可由软件置'1'或置'0'，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)。

7.4.3 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0节。 0000：PA[x]管脚　　0100：PE[x]管脚 0001：PB[x]管脚　　0101：PF[x]管脚 0010：PC[x]管脚　　0110：PG[x]管脚 0011：PD[x]管脚

7.4.4 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]管脚　　0100：PE[x]管脚 0001：PB[x]管脚　　0101：PF[x]管脚 0010：PC[x]管脚　　0110：PG[x]管脚 0011：PD[x]管脚

7.4.5 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]管脚　　0100：PE[x]管脚 0001：PB[x]管脚　　0101：PF[x]管脚 0010：PC[x]管脚　　0110：PG[x]管脚 0011：PD[x]管脚

7.4.6 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]管脚　　0100：PE[x]管脚 0001：PB[x]管脚　　0101：PF[x]管脚 0010：PC[x]管脚　　0110：PG[x]管脚 0011：PD[x]管脚

7.5 GPIO 和AFIO寄存器地址映象

关于寄存器起始地址，请参考表1。下面列出了GPIO和AFIO寄存器映象和复位数值。

表34 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

表35 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				ADC2_ETRGREG_REMA	ADC2_ETRGINJ_REMA	ADC1_ETRGREG_REMA	ADC1_ETRGINJ_REMA	TIM5CH4_IREMAP	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

8 中断和事件

小容量产品是指闪存存储器容量在16K至32K字节之间的STM32F101xx、STM32F102xx和STM32F103xx微控制器。

中容量产品是指闪存存储器容量在64K至128K字节之间的STM32F101xx、STM32F102xx和STM32F103xx微控制器。

大容量产品是指闪存存储器容量在256K至512K字节之间的STM32F101xx和STM32F103xx微控制器。

除非特别说明，本章节描述的模块应用于整个STM32F10xxx微控制器系列。

8.1 嵌套向量中断控制器

特性

- 60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。

8.1.1 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9MHz，产生1ms时基。

8.1.2 中断和异常向量

表36 STM32F10xxx产品的向量表

位置	优先级	优先级类型	名称	说明	地址
	-	-	-	保留	0x0000_0000
	-3	固定	Reset	复位	0x0000_0004
	-2	固定	NMI	不可屏蔽中断 RCC时钟安全系统(CSS)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	DMA1通道1	DMA1通道1全局中断	0x0000_006C
12	19	可设置	DMA1通道2	DMA1通道2全局中断	0x0000_0070
13	20	可设置	DMA1通道3	DMA1通道3全局中断	0x0000_0074
14	21	可设置	DMA1通道4	DMA1通道4全局中断	0x0000_0078
15	22	可设置	DMA1通道5	DMA1通道5全局中断	0x0000_007C
16	23	可设置	DMA1通道6	DMA1通道6全局中断	0x0000_0080
17	24	可设置	DMA1通道7	DMA1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

43	50	可设置	TIM8_BRK	TIM8断开中断	0x0000_00EC
44	51	可设置	TIM8_UP	TIM8更新中断	0x0000_00F0
45	52	可设置	TIM8_TRG_COM	TIM8触发和通信中断	0x0000_00F4
46	53	可设置	TIM8_CC	TIM8捕获比较中断	0x0000_00F8
47	54	可设置	ADC3	ADC3全局中断	0x0000_00FC
48	55	可设置	FSMC	FSMC全局中断	0x0000_0100
49	56	可设置	SDIO	SDIO全局中断	0x0000_0104
50	57	可设置	TIM5	TIM5全局中断	0x0000_0108
51	58	可设置	SPI3	SPI3全局中断	0x0000_010C
52	59	可设置	UART4	UART4全局中断	0x0000_0110
53	60	可设置	UART5	UART5全局中断	0x0000_0114
54	61	可设置	TIM6	TIM6全局中断	0x0000_0118
55	62	可设置	TIM7	TIM7全局中断	0x0000_011C
56	63	可设置	DMA2通道1	DMA2通道1全局中断	0x0000_0120
57	64	可设置	DMA2通道2	DMA2通道2全局中断	0x0000_0124
58	65	可设置	DMA2通道3	DMA2通道3全局中断	0x0000_0128
59	66	可设置	DMA2通道4_5	DMA2通道4和DMA2通道5全局中断	0x0000_012C

8.2 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。

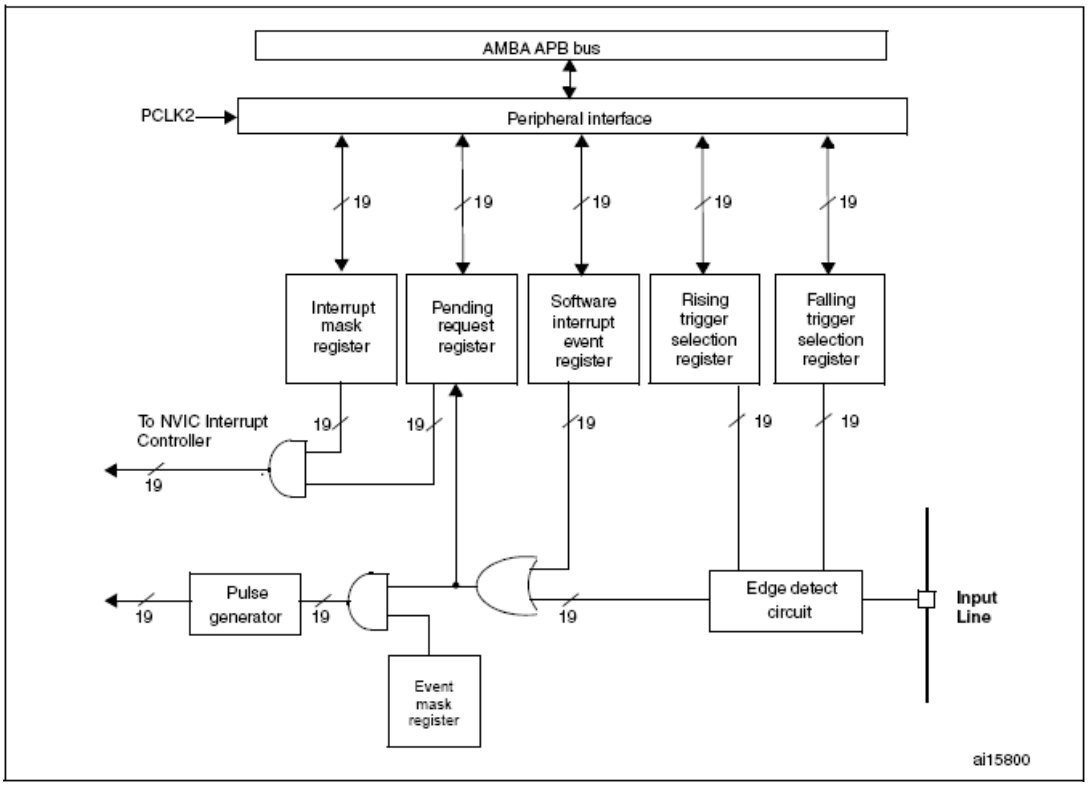
8.2.1 主要特性

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。

8.2.2 框图

图15 外部中断/事件控制器框图



8.2.3 唤醒事件管理

STM32F10xxx可以处理外部或内部事件来唤醒内核(WFE)。唤醒事件可以通过下述配置产生：

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

使用外部I/O端口作为唤醒事件，请参见8.2.4节的功能说明

8.2.4 功能说明

如果要产生中断，必须事先配置好并使能中断线。根据需要的边沿检测设置2个触发寄存器，同时在中断屏蔽寄存器的相应位写'1' 允许中断请求。当外部中断线上发生了需要的边沿时，将产生一个中断请求，对应的挂起位也随之被置'1'。在挂起寄存器的对应位写'1'，可以清除该中断请求。

如果要为产生事件，必须事先配置好并使能事件线。根据需要的边沿检测通过设置2个触发寄存器，同时在事件屏蔽寄存器的相应位写'1'允许事件请求。当事件线上发生了需要的边沿时，将产生一个事件请求脉冲，对应的挂起位不被置'1'。

通过在软件中断/事件寄存器写'1'，也可以通过软件产生中断/事件请求。

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和EXTI_FTSR)

软件中断/事件的选择

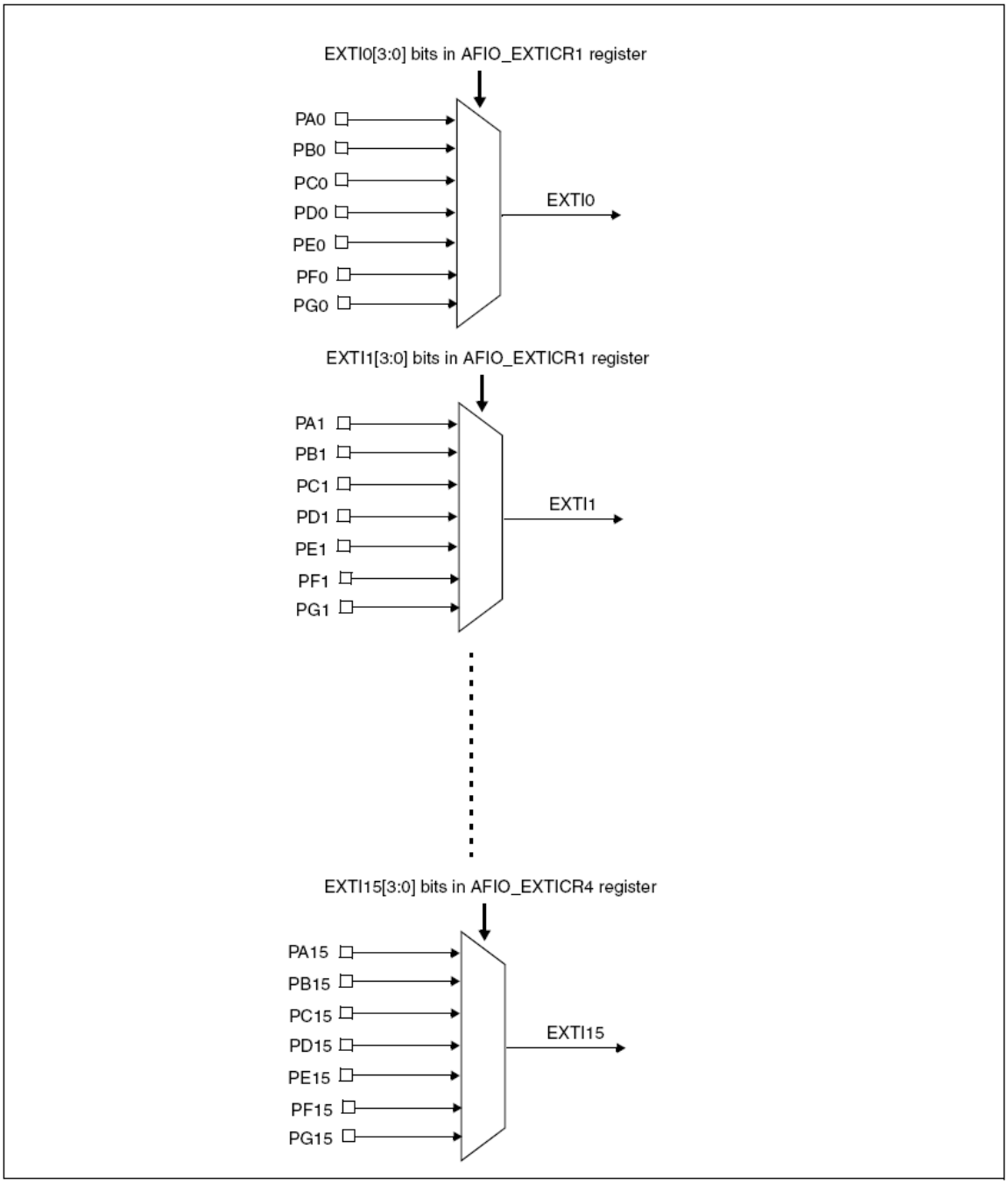
19个线路可以被配置成软件中断/事件线。下面是产生软件中断的过程：

- 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）

8.2.5 外部中断/事件线路映像

112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

8.3 EXTI 寄存器描述

关于寄存器描述中的缩略词，请参考1.1节。

8.3.1 中断屏蔽寄存器(EXTI_IMR)

偏移地址：0x00

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
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：屏蔽线x上的中断请求； 1：开放线x上的中断请求。

8.3.2 事件屏蔽寄存器(EXTI_EMR)

偏移地址：0x04

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
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：屏蔽线x上的事件请求； 1：开放线x上的事件请求。

8.3.3 上升沿触发选择寄存器(EXTI_RTSR)

偏移地址：0x08

复位值：0x0000 0000

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

8.3.4 下降沿触发选择寄存器(EXTI_FTSR)

偏移地址：0x0C

复位值：0x0000 0000

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

8.3.5 软件中断事件寄存器(EXTI_SWIER)

偏移地址：0x10

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
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。 注：通过清除EXTI_PR的对应位(写入'1')，可以清除该位为'0'。

8.3.6 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。

8.3.7 外部中断/事件寄存器映像

表37 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													x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

关于寄存器的起始地址，参见表1。

9 DMA 控制器(DMA)

小容量产品是指闪存存储器容量在16K至32K字节之间的STM32F101xx、STM32F102xx和STM32F103xx微控制器。

中容量产品是指闪存存储器容量在64K至128K字节之间的STM32F101xx、STM32F102xx和STM32F103xx微控制器。

大容量产品是指闪存存储器容量在256K至512K字节之间的STM32F101xx和STM32F103xx微控制器。

除非特别说明，本章节描述的模块应用于整个STM32F10xxx微控制器系列。

9.1 DMA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。

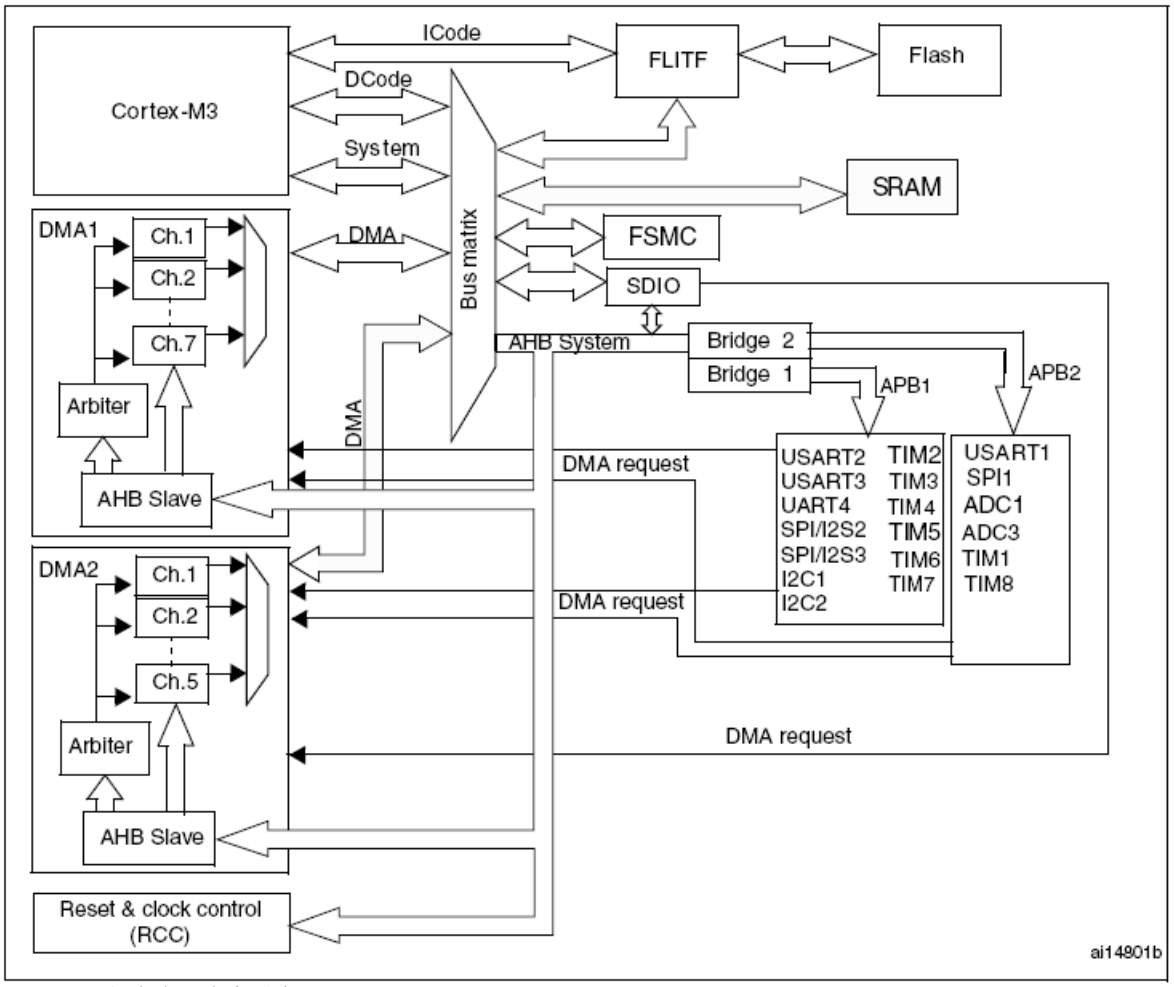
两个DMA控制器有12个通道(DMA1有7个通道，DMA2有5个通道)，每个通道专门用来管理来自于一个或多个外设对存储器访问的请求。还有一个仲裁器来协调各个DMA请求的优先权。

9.2 DMA主要特性

- 12个 独立的可配置的通道(请求)DMA1有7个通道，DMA2有5个通道
- 每个通道都直接连接专用的硬件DMA请求，每个通道都同样支持软件触发。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。源和目标地址必须按数据传输宽度对齐。
- 支持循环的缓冲器管理
- 每个通道都有3个事件标志(DMA 半传输，DMA传输完成和DMA传输出错)，这3个事件标志逻辑或成为一个单独的中断请求。
- 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 APB2和AHB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



1. DMA2仅存在于大容量产品
2. ADC3、SPI/I2S3、UART4、SDIO、TIM5、TIM6、DAC、TIM7、TIM8的DMA请求仅存在于大容量产品。

9.3 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(存储器或外设)带宽。

9.3.1 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。

9.3.2 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。

注意： *在大容量产品中，DMA1控制器拥有高于DMA2控制器的优先级*

9.3.3 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，传输结束后(即传输计数变为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。

9.3.4 可编程的数据传输宽度，对齐方式和数据大小端

当PSIZE和MSIZE不相同时，DMA模块按照下表进行数据对齐。

表38 可编程的数据传输宽度和大小端操作(当PINC=MINC=1)

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

操作一个不支持字节或半字写的AHB设备

当DMA模块开始一个AHB的字节或半字写操作时，数据将在HWDATA[31:0]总线中未使用的部分重复。因此，如果DMA以字节或半字写入不支持字节或半字写操作的AHB设备时(即HSIZE不适于该模块)，不会发生错误，DMA将按照下面两个例子写入32位HWDATA数据：

- 当HSIZE=半字时，写入半字'0xABCD'，DMA将设置HWDATA总线为'0xABCDABCD'。
- 当HSIZE=字节时，写入字节'0xAB'，DMA将设置HWDATA总线为'0xABABABAB'。

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

- 一个AHB上对地址0x0(或0x1、0x2或0x3)的写字节数据'0xB0'操作，将转换到APB上对地址0x0的写字数据'0xB0B0B0B0'操作。
- 一个AHB上对地址0x0(或0x2)的写半字数据'0xB1B0'操作，将转换到APB上对地址0x0的写字数据'0xB1B0B1B0'操作。

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

9.3.5 错误管理

读写一个保留的地址区域，将会产生DMA传输错误。当在DMA读写操作时发生DMA传输错误时，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。

9.3.6 中断

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

表39 DMA中断请求

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

注意：在大容量产品中，DMA2通道4和DMA2通道5的中断被映射在同一个中断向量上，其他的DMA通道都有自己的中断向量。

9.3.7 DMA请求映像

DMA1控制器

从外设(TIMx、ADC、SPIx、I²Cx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DMA1请求映像。

外设的DMA1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 DMA1请求映像

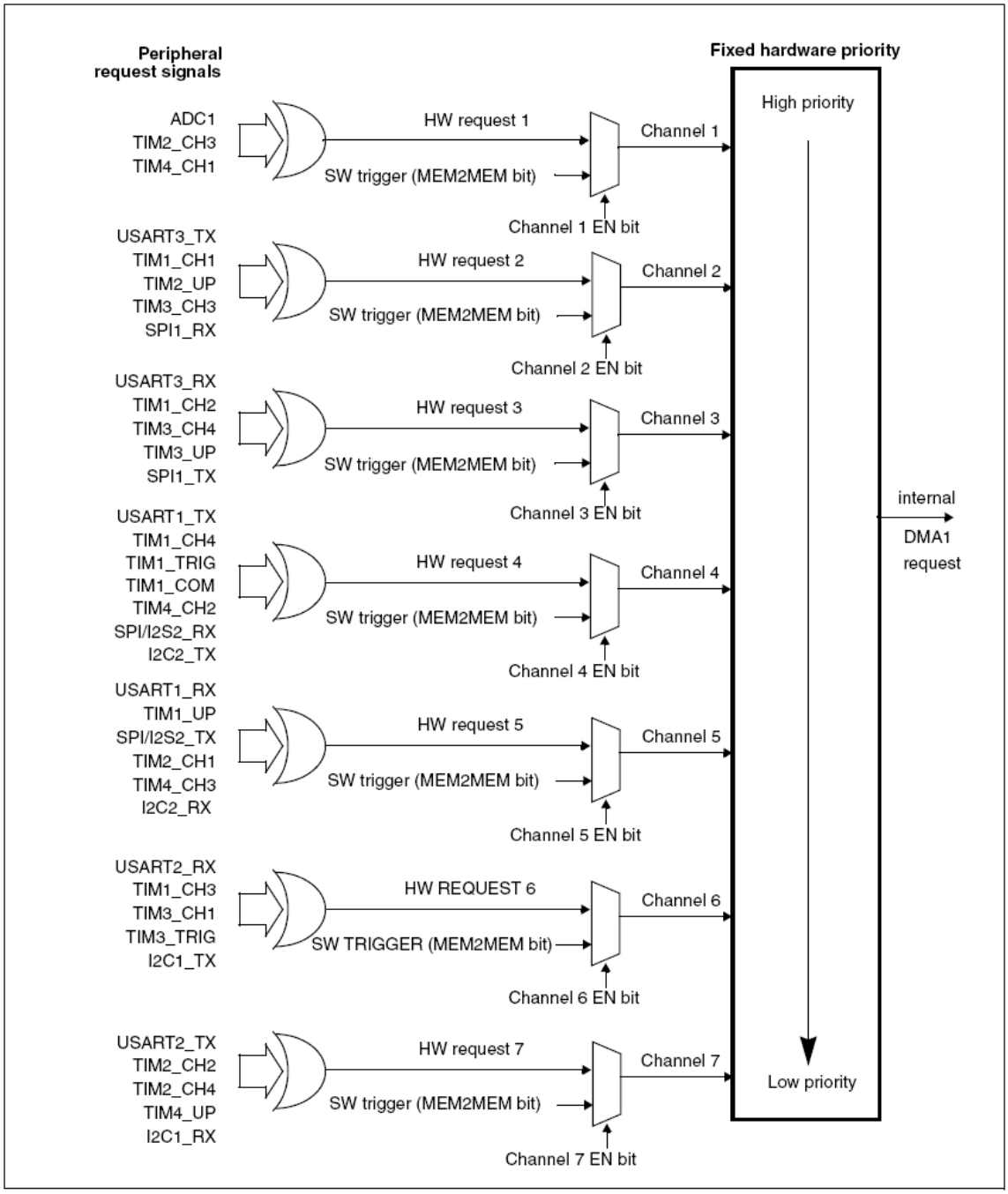


表40 各个通道的DMA1请求一览

外设	通道1	通道2	通道3	通道4	通道5	通道6	通道7
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

DMA2控制器

从外设(TIMx[5、6、7、8]、ADC3、SPI/I2S3、UART4、DAC通道1、2和SDIO)产生的5个请求，经逻辑或输入到DMA控制器，这意味着同时只能有一个请求有效。参见下图的DMA2请求映像。

外设的DMA2请求，可以通过设置相应外设寄存器中的DMA控制位，被独立地开启或关闭。

注意： DMA2控制器及相关请求仅存在于大容量产品

图19 DMA2请求映像

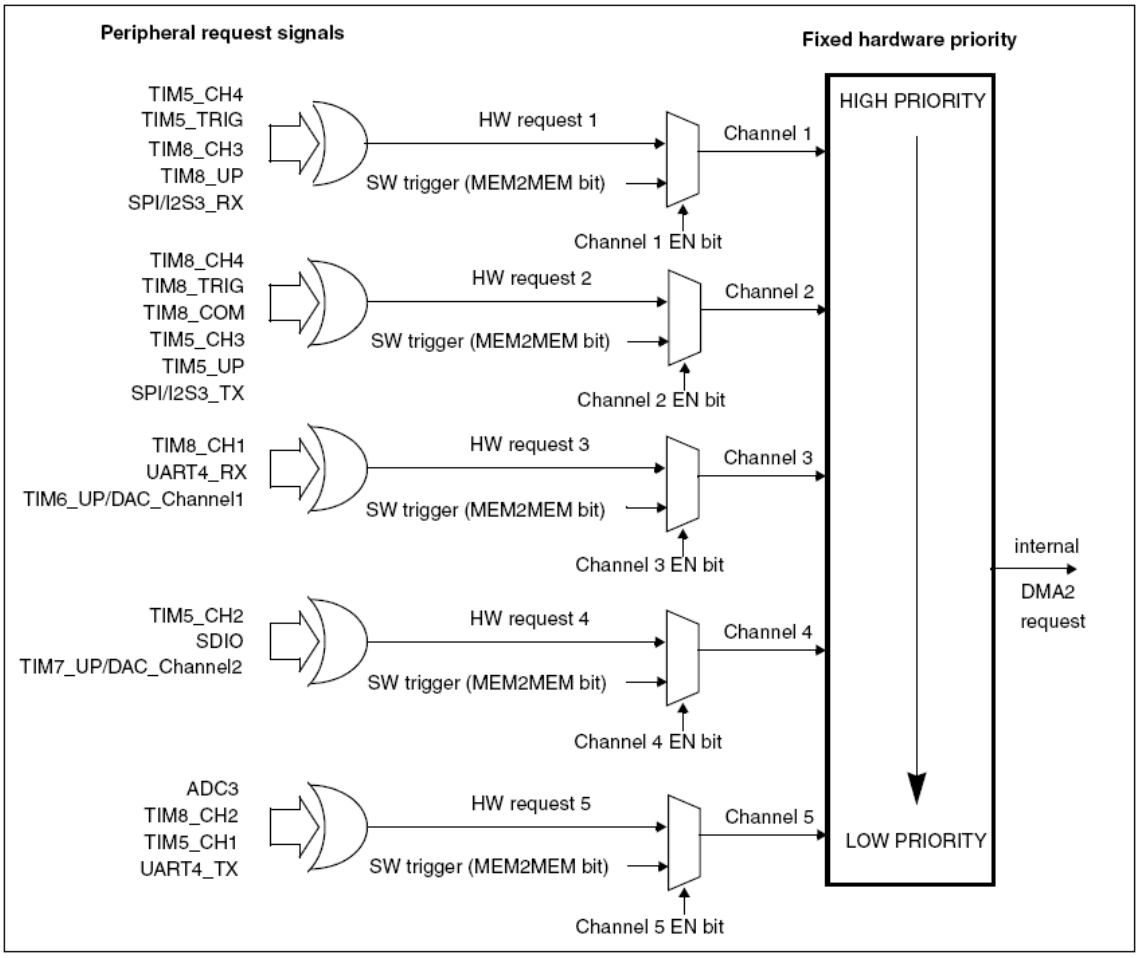


表41 各个通道的DMA2请求一览

外设	通道1	通道2	通道3	通道4	通道5
ADC3					ADC3
SPI/I2S3	SPI/I2S3_RX	SPI/I2S3_TX			
UART4			UART4_RX		UART4_TX
SDIO				SDIO	
TIM5	TIM5_CH4 TIM5_TRIG	TIM5_CH3 TIM5_UP		TIM5_CH2	TIM5_CH1
TIM6/ DAC通道1			TIM6_UP/ DAC通道1		
TIM7/ DAC通道2				TIM7_UP/ DAC通道2	
TIM8	TIM8_CH3 TIM8_UP	TIM8_CH4 TIM8_TRIG TIM8_COM	TIM8_CH1		TIM8_CH2

9.4 DMA寄存器

关于寄存器描述中用到的缩写，请参见第1章。

注意： *在以下列举的所有寄存器中，所有与通道6和通道7相关的位，对DMA2都不适用，因为DMA2只有5个通道。*

9.4.1 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发生了传输错误(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产生了半传输事件(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产生了传输完成事件(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产生了TE、HT或TC事件。

9.4.2 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。

9.4.3 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

9.4.4 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。

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

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

9.4.7 DMA寄存器映像

表42 DMA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	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																																

10 模拟/数字转换(ADC)

小容量产品是指闪存存储器容量在16K至32K字节之间的STM32F101xx、STM32F102xx和STM32F103xx微控制器。

中容量产品是指闪存存储器容量在64K至128K字节之间的STM32F101xx、STM32F102xx和STM32F103xx微控制器。

大容量产品是指闪存存储器容量在256K至512K字节之间的STM32F101xx和STM32F103xx微控制器。

除非特别说明，本章节描述的模块应用于整个STM32F10xxx微控制器系列。

10.1 ADC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。

10.2 ADC主要特征

- 12-位分辨率
- 转换结束，注入转换结束和发生模拟看门狗事件时产生中断
- 单次和连续转换模式
- 从通道0到通道n的自动扫描模式
- 自校准
- 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个或以上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)
  - STM32F102xxUSB 型产品：ADC 时钟为 48MHz 时为 1.2μs
- ADC供电要求：2.4V到3.6V
- ADC输入范围：VREF- ≤ VIN ≤ VREF+
- 规则通道转换期间有DMA请求产生。

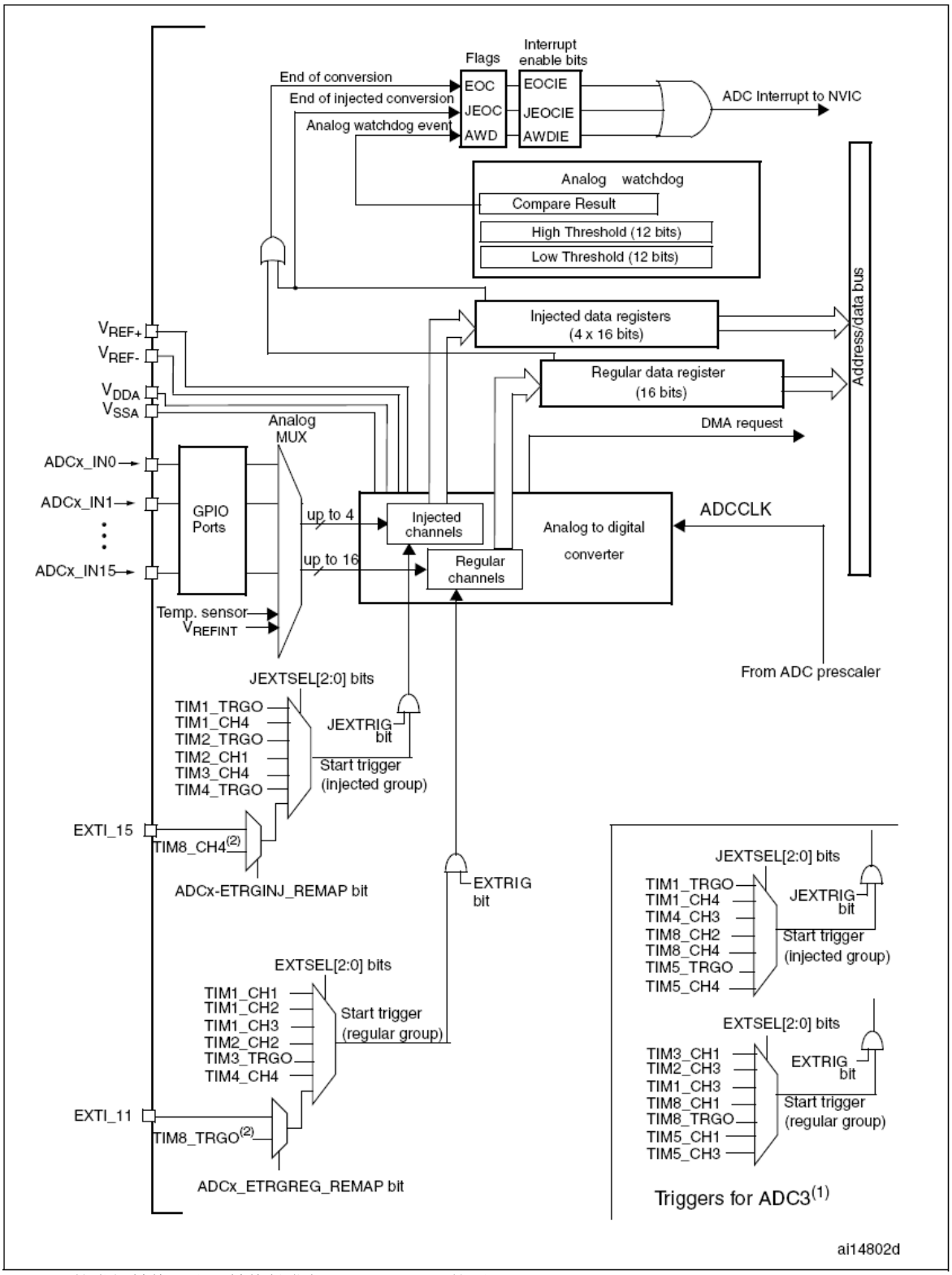
下图是ADC模块的方框图。

注意： *如果有V_{REF-}管脚(取决于封装)，必须和VSSA相连接*

10.3 ADC功能描述

下图为一个ADC模块的框图，表43为ADC管脚的说明。

图20 单个ADC框图



1.ADC3的常规转换和注入转换触发与ADC1和ADC2的不同。

2.TIM8_CH4和TIM8_TRGO及他们的重映射位只存在于大容量产品中

表43 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

10.3.1 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)。

10.3.2 ADC时钟

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

10.3.3 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/ V_{REFINT}内部通道

温度传感器和通道ADCx_IN16相连接，内部参照电压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。*

10.3.4 单次转换模式

单次转换模式下，ADC只执行一次转换。该模式既可通过设置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。

10.3.5 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。

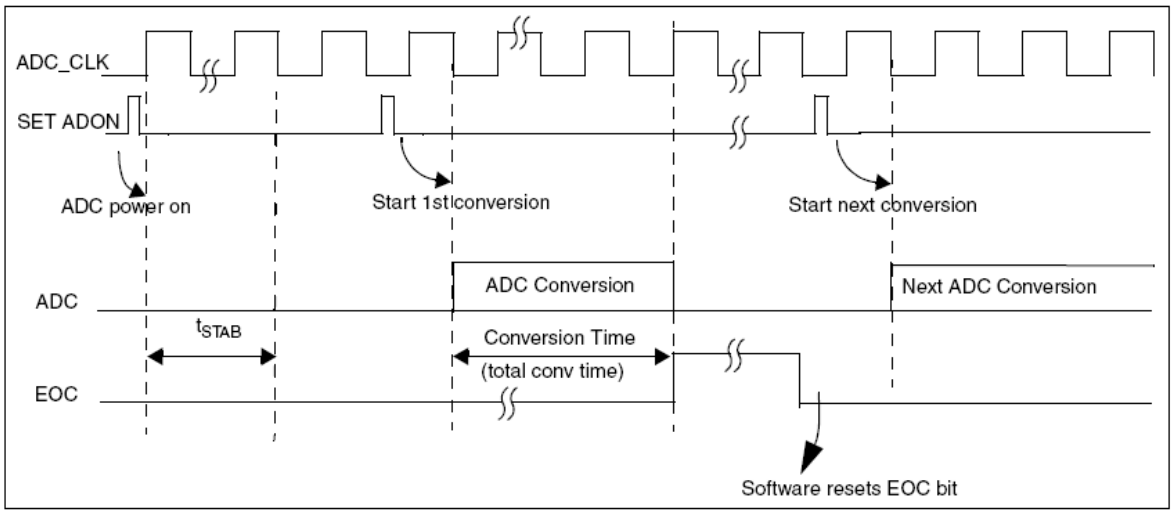
每个转换后：

- 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。

10.3.6 时序图

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

图21 时序图



10.3.7 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10.5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44所示。

图22 模拟看门狗警戒区

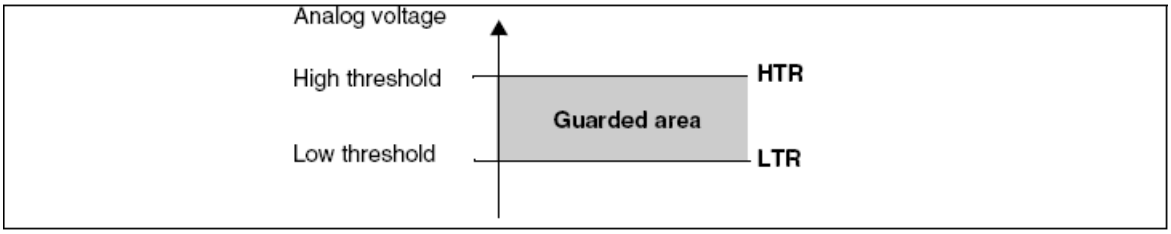


表44 模拟看门狗通道选择

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

10.3.8 扫描模式

此模式用来扫描一组模拟通道。

扫描模式可通过设置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。

10.3.9 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23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。

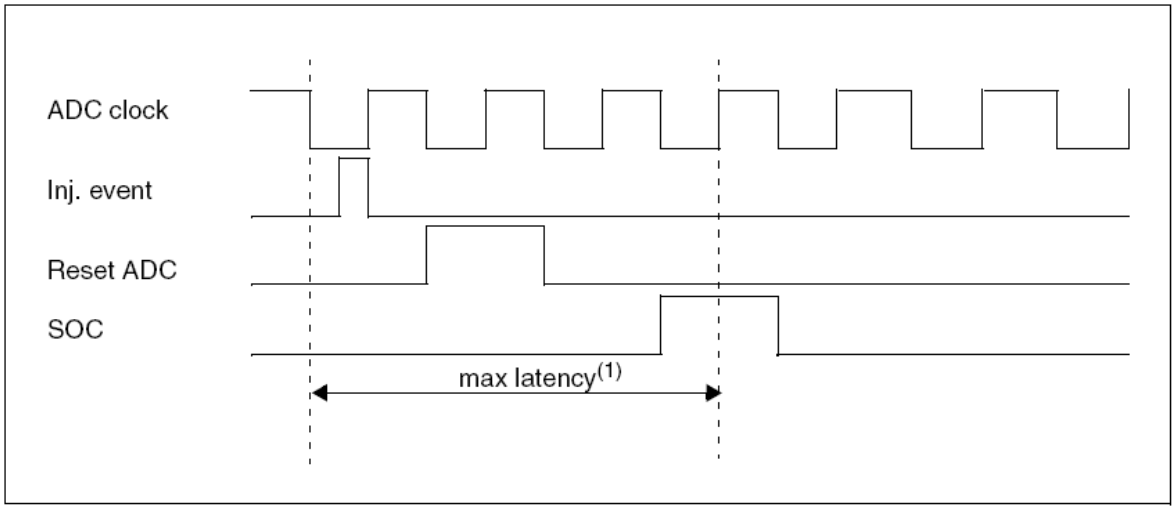
在此模式里，必须禁止注入通道的外部触发。

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；当ADC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23 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。

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

注入组

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。

10.4 校准

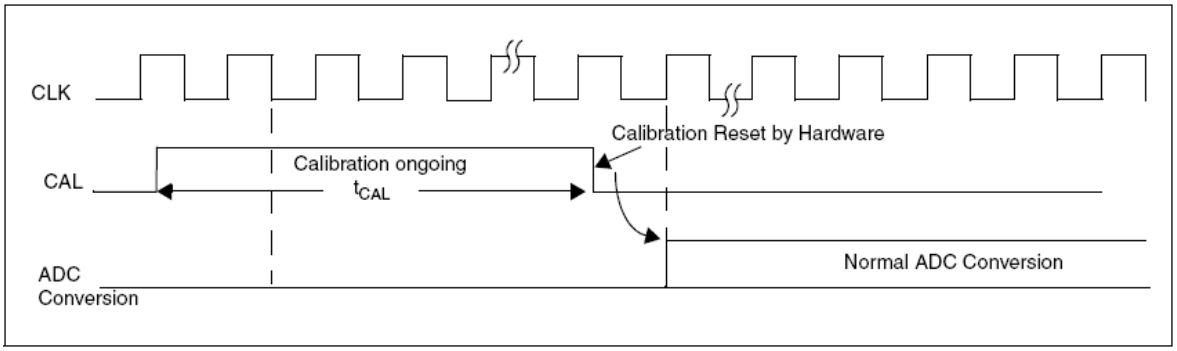
ADC有一个内置自校准模式。校准可大幅减小因内部电容器组的变化而造成的准精度误差。在校准期间，每个电容器上都会计算出一个误差修正码(数字值)，这个码用于消除在随后的转换中每个电容器上产生的误差。

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24 校准时序图



10.5 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25和0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25 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26 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

10.6 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

10.7 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45 ADC1和ADC2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

TIM8_TRGO事件只存在于大容量产品

对于规则通道，选中EXTI线路11和TIM8_TRGO作为外部触发事件，可以通过设置ADC1和ADC2的ADC1_ETRGREG_REMAP位和ADC2_ETRGREG_REMAP位实现。

表46 ADC1和ADC2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

TIM8_CC4事件只存在于大容量产品

对于规则通道，选中EXTI线路15和TIM8_CC4作为外部触发事件，可以通过设置ADC1和ADC2的ADC1_ENTRGINJ_REMAP位和ADC2_ ENTRGINJ_REMAP位实现。

表47 ADC3用于规则通道的外部触发

触发源	连接类型	EXTSEL[2:0]
定时器3的CC1输出	片上定时器的内部信号	000
定时器2的CC3输出		001
定时器1的CC3输出		010
定时器8的CC1输出		011
定时器8的TRGO输出		100
定时器5的CC1输出		101
定时器5的CC3输出		110
SWSTART	软件控制位	111

表48 ADC3用于注入通道的外部触发

触发源	连接类型	EXTSEL[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4的CC3输出		010
定时器8的CC2输出		011
定时器8的CC4输出		100
定时器5的TRGO输出		101
定时器5的CC4输出		110
JSWSTART	软件控制位	111

软件触发事件可以通过对寄存器ADC_CR2的SWSTART或JSWSTART位置'1'产生。

规则组的转换可以被注入触发打断。

10.8 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和ADC3拥有DMA功能。由ADC2转化的数据可以通过双ADC模式，利用ADC1的DMA性能来实现。*

10.9 双ADC模式

在有2个或以上ADC的器件中，可以使用双ADC模式(见图27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：

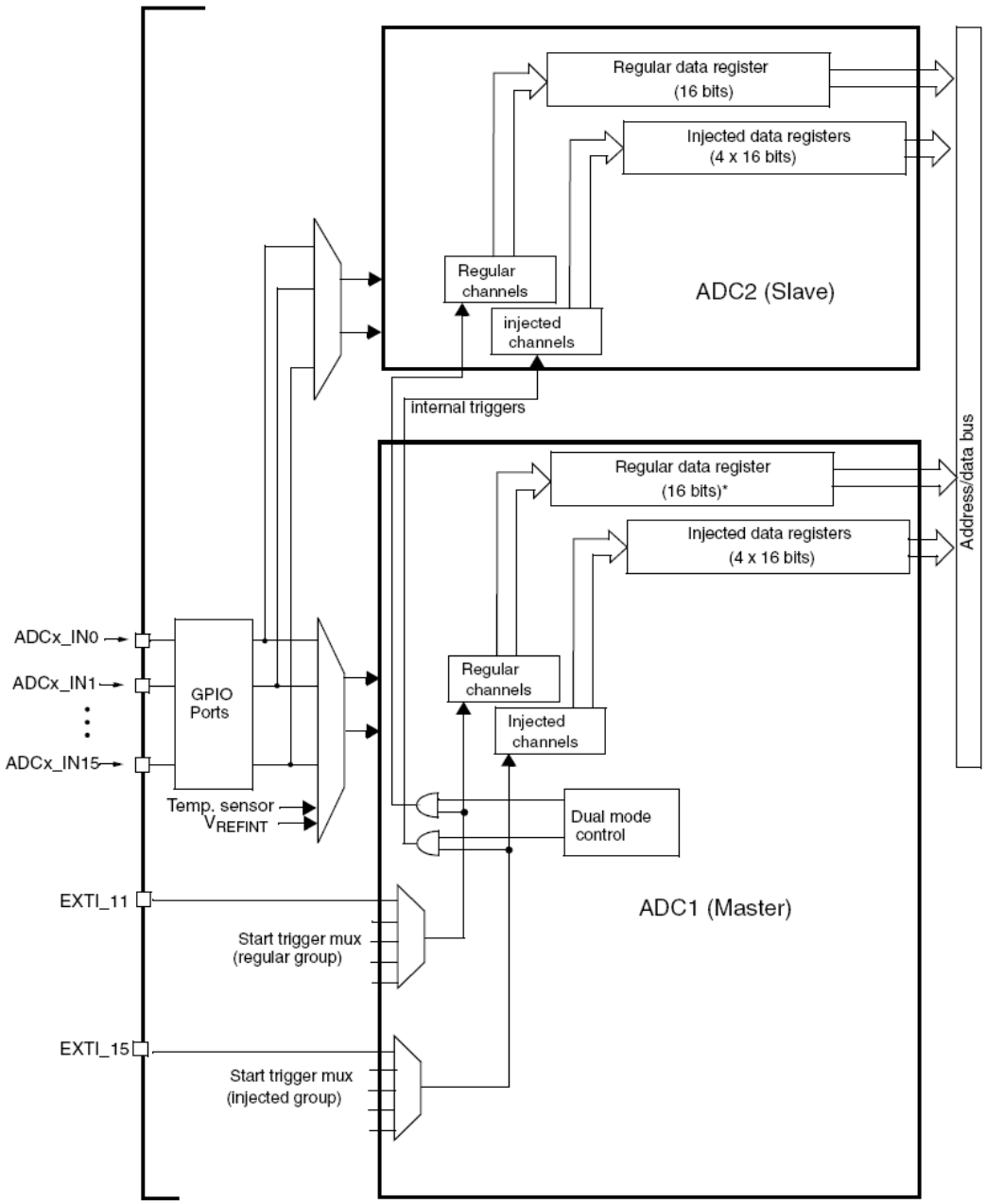
- 同时注入模式
- 同时规则模式
- 快速交替模式
- 慢速交替模式
- 交替触发模式
- 独立模式

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27 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。

10.9.1 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。

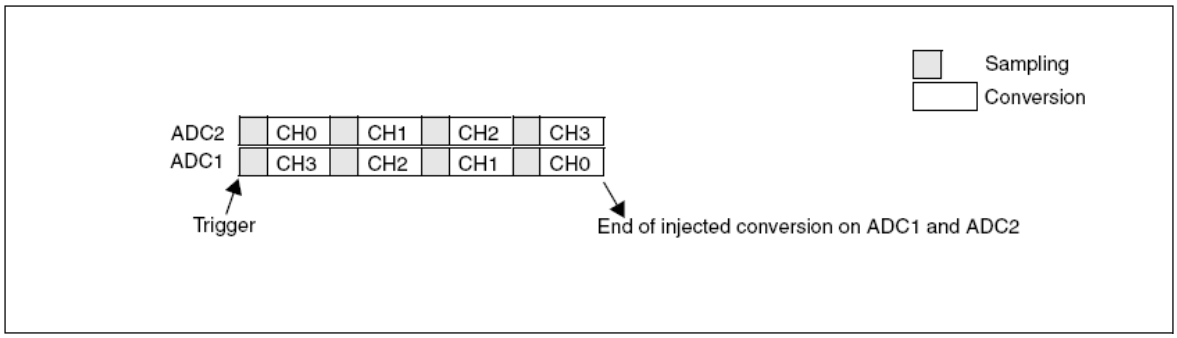
注意： *不要在2个ADC上转换相同的通道(如果转换两个ADC的相同通道，不可能提供重叠的采样时间)。*

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(若任一ADC接口开放了中断)。

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28 在4个通道上的同时注入模式



10.9.2 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。

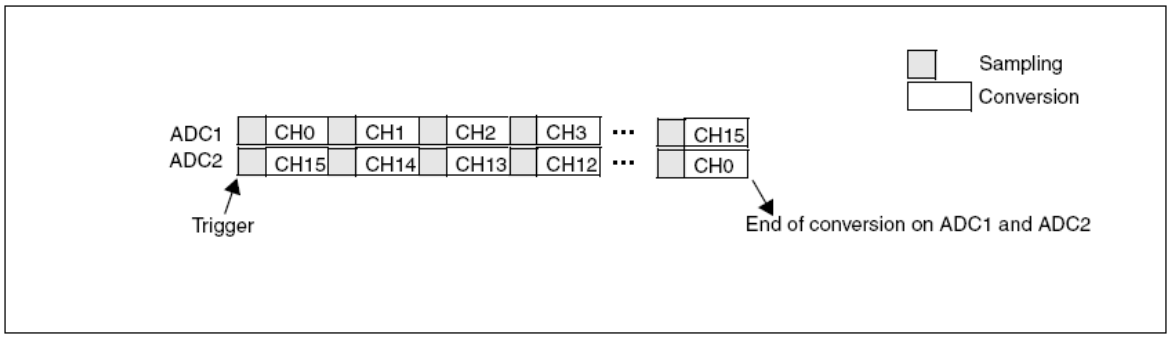
注意： *不要在2个ADC上转换相同的通道(如果转换两个ADC的相同通道，不可能提供重叠的采样时间)。*

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29 在16个通道上的同时规则模式



10.9.3 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：

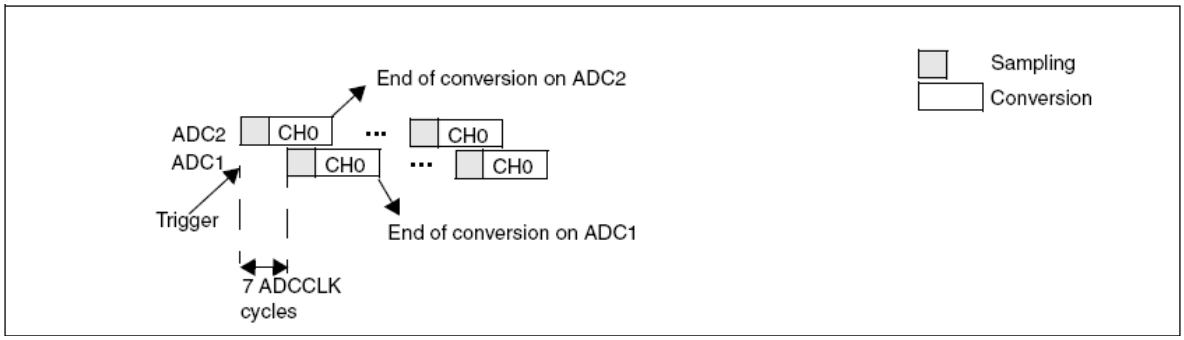
- ADC2立即启动并且
- ADC1在延迟7个ADC时钟周期后启动

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30 在1个通道上连续转换模式下的快速交替模式



10.9.4 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。*

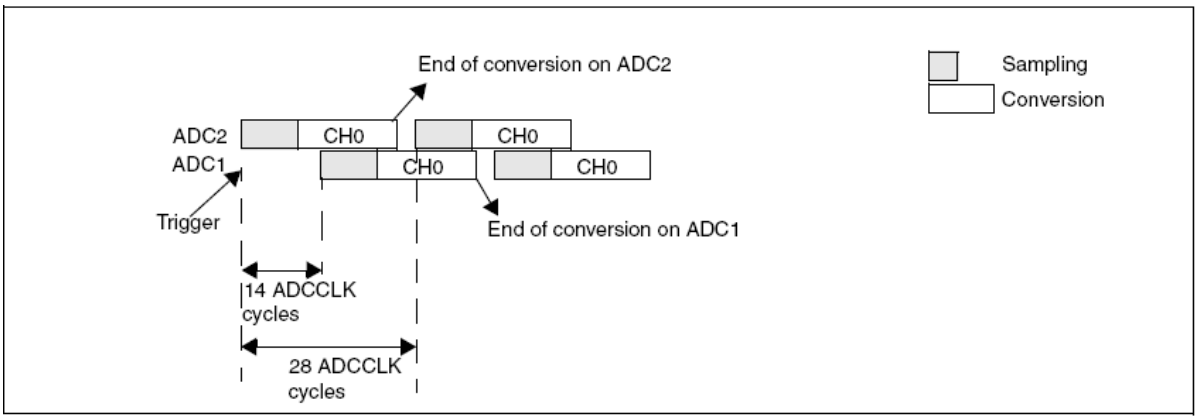
ADC1产生EOC中断后(由EOCIE使能)，产生一个32位的DMA传输请求(如果设置了DMA位)，ADC1_DR寄存器的32位数据被传输到SRAM，ADC1_DR的上半个字包含ADC2的转换数据，低半个字包含ADC1的转换数据。

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31 在1个通道上的慢速交替模式



10.9.5 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。

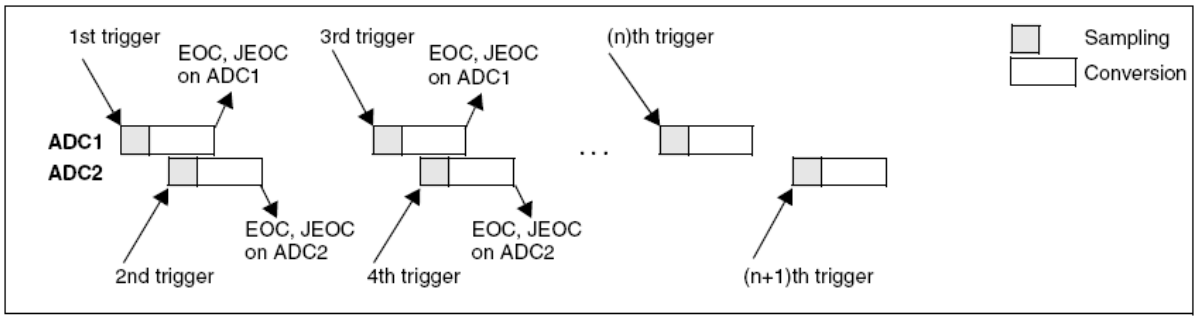
- 当第一个触发产生时，ADC1上的所有注入组通道被转换。
- 当第二个触发到达时，ADC2上的所有注入组通道被转换。
- 如此循环……

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32 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：

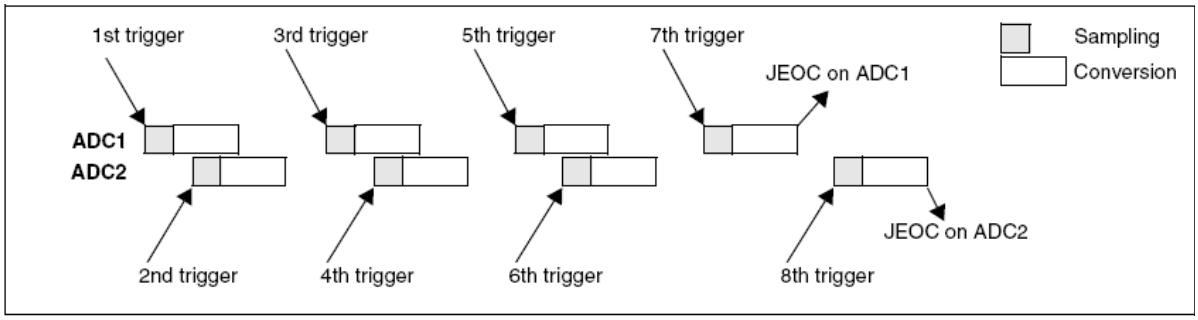
- 当第一个触发产生时，ADC1上的第一个注入通道被转换。
- 当第二个触发到达时，ADC2上的第一个注入通道被转换。
- 如此循环……

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33 交替触发：在间断模式下每个ADC上的4个注入通道



10.9.6 独立模式

此模式里，双ADC同步不工作，每个ADC接口独立工作。

10.9.7 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。

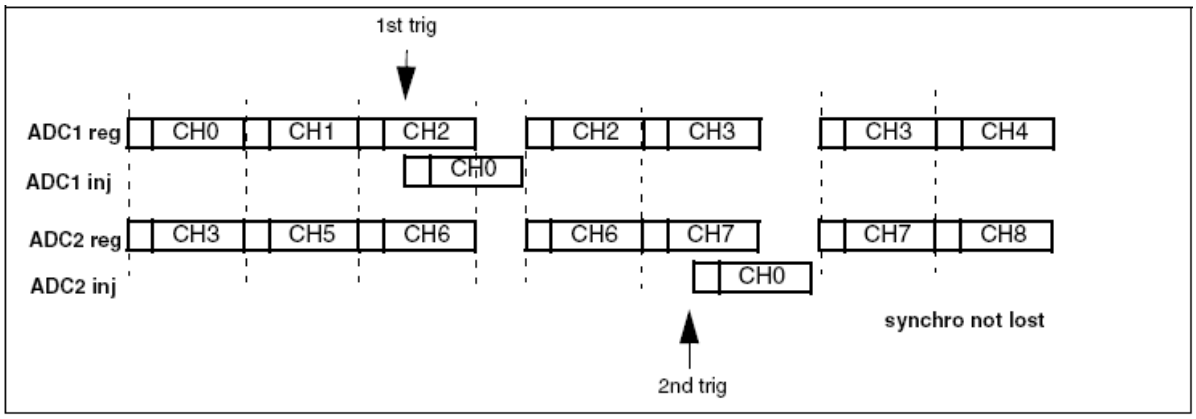
10.9.8 混合的同步规则+交替触发模式

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34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。

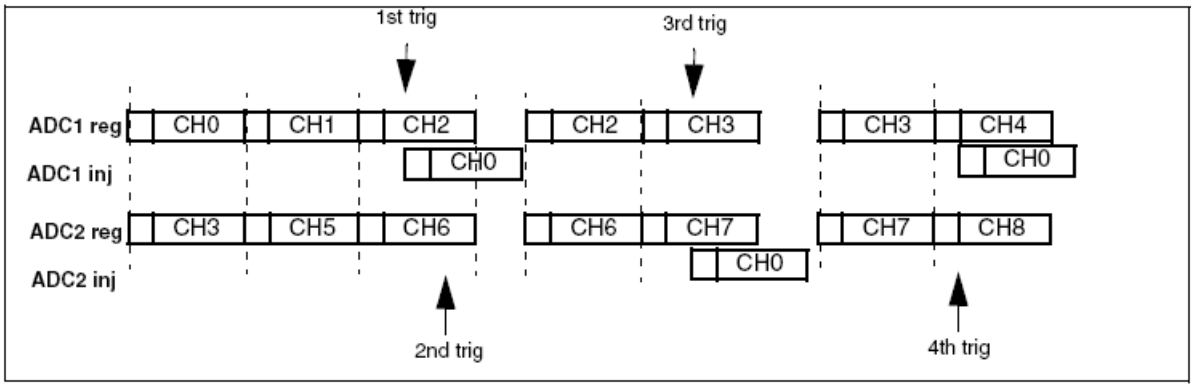
注：在混合的同步规则+交替触发模式中，必须转换具有相同长度的序列，或保证触发的间隔比2个序列中较长的序列长，否则当较长序列的转换还未完成时，具有较短序列的ADC转换会被重启。

图34 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35示出了这种情况的操作(第2个触发被忽略)。

图35 触发事件发生在注入转换期间

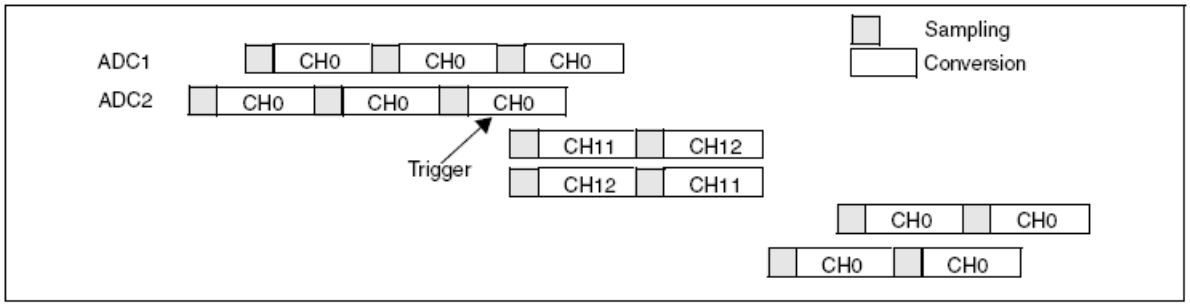


10.9.9 混合同步注入+交替模式

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36 交替的单通道转换被注入序列CH11和CH12中断



10.10 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推荐采样时间是17.1μs。

图37是温度传感器的方框图。

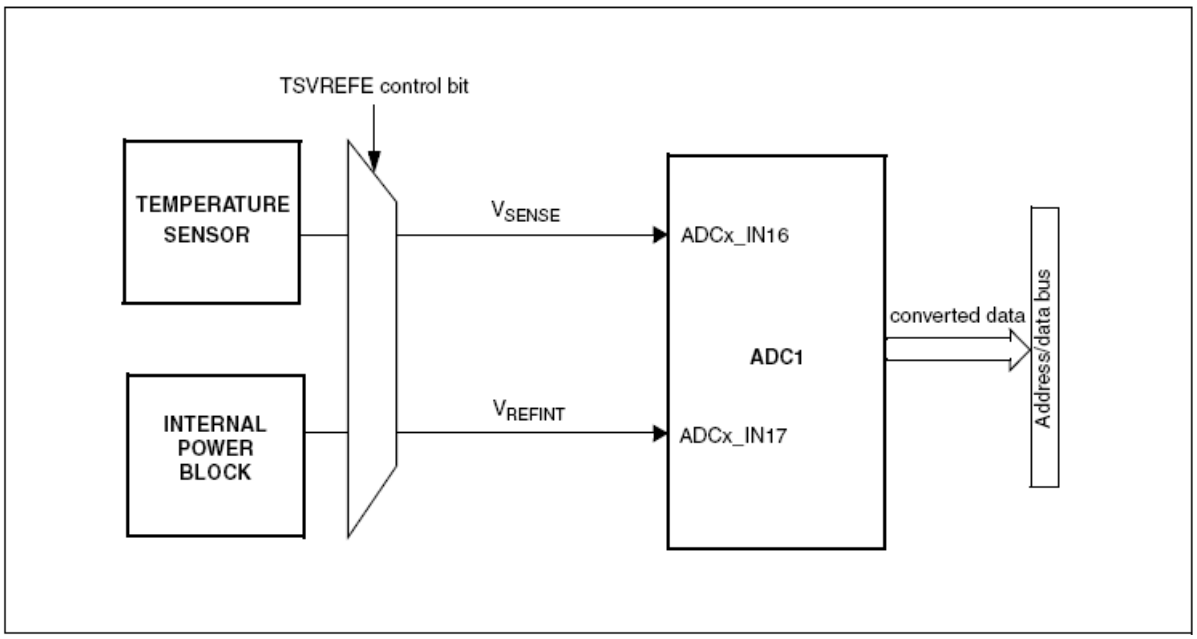
当没有被使用时，传感器可以置于关电模式。

注意： *必须设置TSVREFE位激活内部通道：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37 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。*

10.11 ADC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。

注： *ADC1和ADC2的中断映射在同一个中断向量上，而ADC3的中断有自己的中断向量。*

ADC_SR寄存器中有2个其他标志，但是它们没有相关联的中断：

- JSTRT(注入组通道转换的启动)
- STRT(规则组通道转换的启动)

表49 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

10.12 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

10.12.1 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

10.12.2 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和ADC3中这些位为保留位 在双模式中，改变通道的配置会产生一个重新开始的条件，这将导致同步丢失。建议在进行任何配置改变前关闭双模式。
位15:13	**DISCNUM[2:0]**：间断模式通道计数 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VREFINT。 ADC2的模拟输入通道16和通道17在芯片内部连到了VSS。 ADC3模拟输入通道9，14，15，16，17与Vss相连

10.12.3 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								TS VREFE	SW START	SW STARTJ	EXT TRIG	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
JEXT TRIG	JEXTSEL[2:0]			ALIGN	保留		DMA	保留				RST CAL	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 ADC1和ADC2的触发配置如下 000：定时器1的CC1事件　100：定时器3的TRGO事件 001：定时器1的CC2事件　101：定时器4的CC4事件 010：定时器1的CC3事件　110：EXTI线11/ TIM8_TRGO，仅大容量产品具有TIM8_TRGO功能 011：定时器2的CC2事件　111：SWSTART ADC3的触发配置如下 000：定时器3的CC1事件　100：定时器8的TRGO事件 001：定时器2的CC3事件　101：定时器5的CC1事件 010：定时器1的CC3事件　110：定时器5的CC3事件 011：定时器8的CC1事件　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 ADC1和ADC2的触发配置如下 000：定时器1的TRGO事件　100：定时器3的CC4事件 001：定时器1的CC4事件　101：定时器4的TRGO事件 010：定时器2的TRGO事件　110：EXTI线15/TIM8_CC4事件(仅大容量产品具有TIM8_CC4) 011：定时器2的CC1事件　111：JSWSTART ADC3的触发配置如下 000：定时器1的TRGO事件　100：定时器8的CC4事件 001：定时器1的CC4事件　101：定时器5的TRGO事件 010：定时器4的CC3事件　110：定时器5的CC4事件 011：定时器8的CC2事件　111：JSWSTART
位11	**ALIGN**：数据对齐 该位由软件设置和清除。参考图25和图26。 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21。 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。

10.12.4 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
SMP 15_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23:0	**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VREFINT。 – ADC2的模拟输入通道16和通道17在芯片内部连到了VSS。 - ADC3模拟输入通道14，15，16，17与Vss相连

10.12.5 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
SMP 5_0	SMP4[2:0]			SMP3[2:0]			SMP2[2:0]			SMP1[2:0]			SMP0[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位31:30	保留。必须保持为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 注：ADC3模拟输入通道9与Vss相连

10.12.6 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				JOFFSETx[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。

10.12.7 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。

10.12.8 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。

10.12.9 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

10.12.10 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

10.12.11 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

10.12.12 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

10.12.13 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25和图26所示。

10.12.14 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10.9：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25和图26所示。

10.12.15 ADC寄存器地址映像

下表列出了所有的ADC寄存器。

表50 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	0
18h	ADC_JOFR2	保留																					JOFFSET2[11:0]										
	复位值																						0	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	0
20h	ADC_JOFR4	保留																					JOFFSET4[11:0]										
	复位值																						0	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	0
20h	ADC_LTR	保留																					LT[11:0]										
	复位值																						0	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

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，请参见表1。

11 数字/模拟转换(DAC)

小容量产品是指闪存存储器容量在16K至32K字节之间的STM32F101xx、STM32F102xx和STM32F103xx微控制器。

中容量产品是指闪存存储器容量在64K至128K字节之间的STM32F101xx、STM32F102xx和STM32F103xx微控制器。

大容量产品是指闪存存储器容量在256K至512K字节之间的STM32F101xx和STM32F103xx微控制器。

本章内容只适用于大容量的STM32F101xx和STM32F103xx产品。

11.1 DAC简介

数字/模拟转换模块(DAC)是12位数字输入，电压输出的数字/模拟转换器。DAC可以配置成8位或者12位模式，也可以与DMA控制器配合使用。DAC工作在12位模式时，数据可以设置成左对齐，也可以设置成右对齐。DAC有2个输出通道，每个通道都有单独的转换器，可以工作在双DAC模式。在此模式下，可以同步地更新2个通道的输出，这2个通道的转换可以同时进行，也可以分别进行。DAC可以通过管脚输入参考电压V_{REF+}以获得更精确的转换结果。

11.2 DAC主要特征

- 2个DAC转换器：1个输出通道对应1个转换器
- 8位或者12位单调输出
- 12位模式下数据左对齐或者右对齐
- 同步更新功能
- 噪声波形生成
- 三角波形生成
- 双DAC通道同时或者分别转换
- 每个通道都有DMA功能
- 外部触发转换
- 输入参考电压V_{REF+}

单个DAC通道的框图如下图，表51表给出了管脚的描述

图38 DAC 通道模块框图

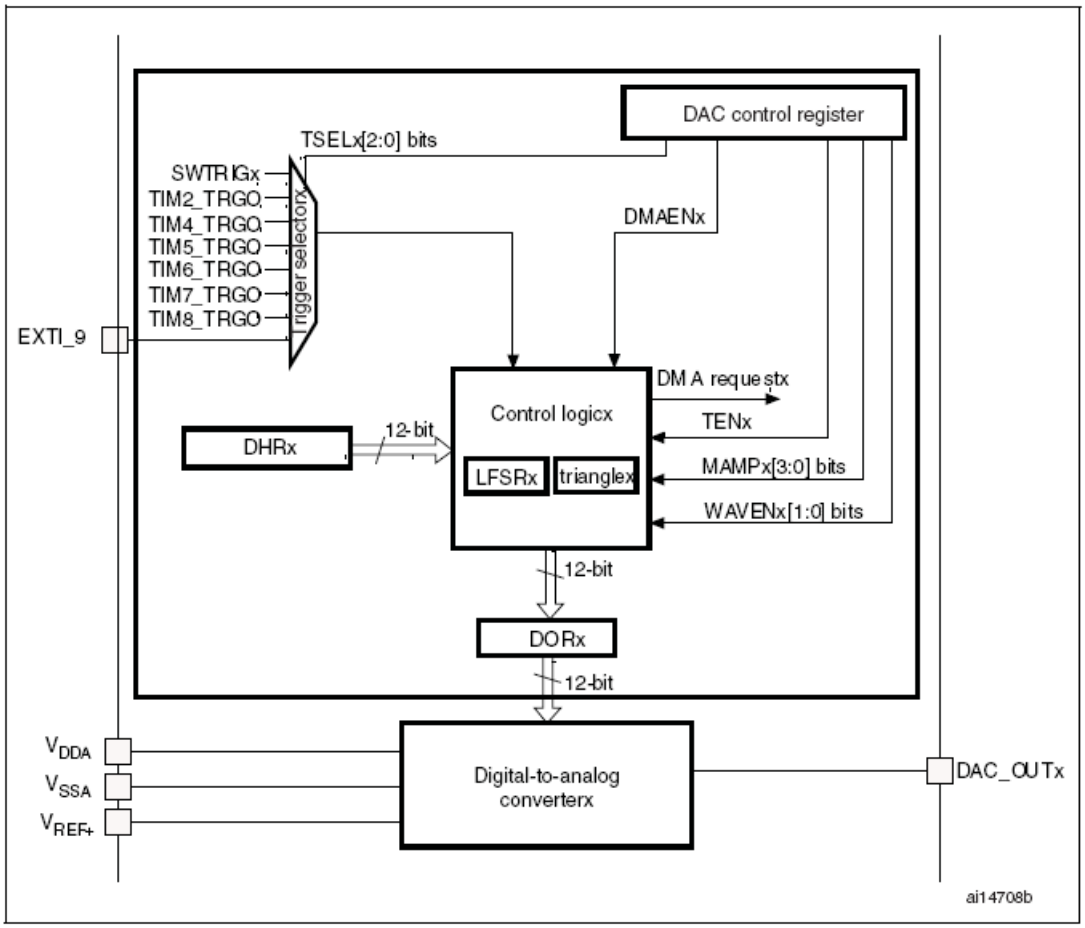


表51 DAC 管脚

名称	型号类型	注释
V_{REF+}	输入，正模拟参考电压	DAC使用的高端/正极参考电压， 2.4V ≤ VREF+ ≤ VDDA （3.3 V）
V_{DDA}	输入，模拟电源	模拟电源
V_{SSA}	输入，模拟电源地	模拟电源的地线
DAC_OUTx	模拟输出信号	DAC通道x的模拟输出

注意： *一旦使能DAC通道，相应的GPIO管脚(PA4或者PA5)就会自动与DAC的模拟输出相连(DAC_OUTx)。为了避免寄生的干扰和额外的功耗，管脚PA4或者PA5在之前应当设置成模拟输入(AIN)。*

11.3 DAC功能描述

11.3.1 使能DAC通道

将DAC_CR寄存器的ENx位置1即可打开对DAC通道x的供电。经过一段启动时间t_{WAKEUP}，DAC通道x即被使能。

注意： *ENx位只会使能DAC通道x的模拟部分，即便该位被置0，DAC通道x的数字部分仍然工作。*

11.3.2 使能DAC输出缓存

DAC集成了2个输出缓存，可以用来减少输出阻抗，无需外部运放即可直接驱动外部负载。每个DAC通道输出缓存可以通过设置DAC_CR寄存器的相应位BOFFx来使能或者关闭。

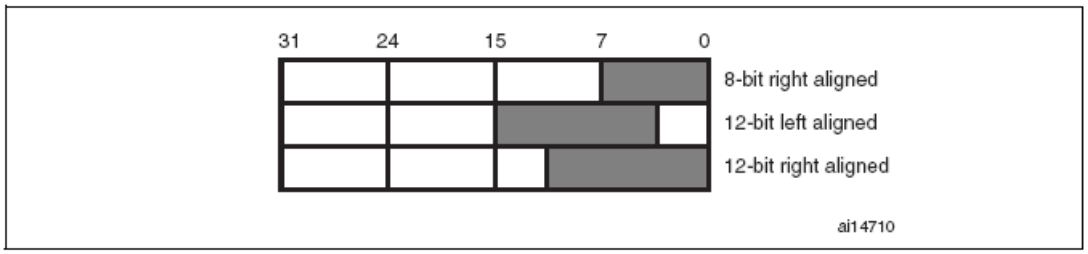
11.3.3 DAC数据格式

根据选则的配置模式，数据按照下文所述写入指定的寄存器：

- 单DAC通道x，有3种情况：
  - 8位数据右对齐：用户须将数据写入寄存器DAC_DHR8Rx的[7:0]位(实际是存入寄存器DHRx[11:4]位)
  - 12位数据左对齐：用户须将数据写入寄存器DAC_DHR12Lx的[15:4]位(实际是存入寄存器DHRx[11:0]位)
  - 12位数据右对齐：用户须将数据写入寄存器DAC_DHR8Rx的[11:0]位(实际存入寄存器DHRx[11:0]位)

根据对DAC_DHRyyyx寄存器的操作，经过相应的移位后，写入的数据被转存到DHRx寄存器中(DHRx是内部的数据保存寄存器x)。随后，DHRx的内容或被自动地传送到DORx寄存器，或通过软件触发或外部事件触发被传送到DORx寄存器。

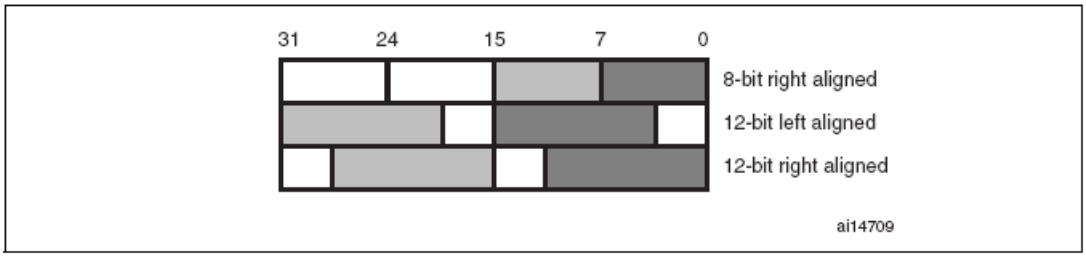
图39 单 DAC 通道模式的数据寄存器



- 双DAC通道，有3种情况：
  - 8位数据右对齐：用户须将DAC通道1数据写入寄存器DAC_DHR8Rx的[7:0]位(实际是存入寄存器DHR1[11:4]位)，将DAC通道2数据写入寄存器DAC_DHR8Rx的[15:8]位(实际是存入寄存器DHR2[11:4]位)
  - 12位数据左对齐：用户须将DAC通道1数据写入寄存器DAC_DHR12LD的[15:4]位(实际是存入寄存器DHR1[11:0]位)，将DAC通道2数据写入寄存器DAC_DHR12LD的[31:20]位(实际是存入寄存器DHR2[11:0]位)
  - 12位数据右对齐：用户须将DAC通道1数据写入寄存器DAC_DHR12LD的[11:0]位(实际是存入寄存器DHR1[11:4]位)，将DAC通道2数据写入寄存器DAC_DHR12LD的[27:16]位(实际是存入寄存器DHR2[11:0]位)

根据对DAC_DHRyyyD寄存器的操作，经过相应的移位后，写入的数据被转存到DHR1和DHR2寄存器中(DHR1和DHR2是内部的数据保存寄存器x)。随后，DHR1和DHR2的内容或被自动地传送到DORx寄存器，或通过软件触发或外部事件触发被传送到DORx寄存器。

图40 双 DAC 通道模式的数据寄存器



11.3.4 DAC转换

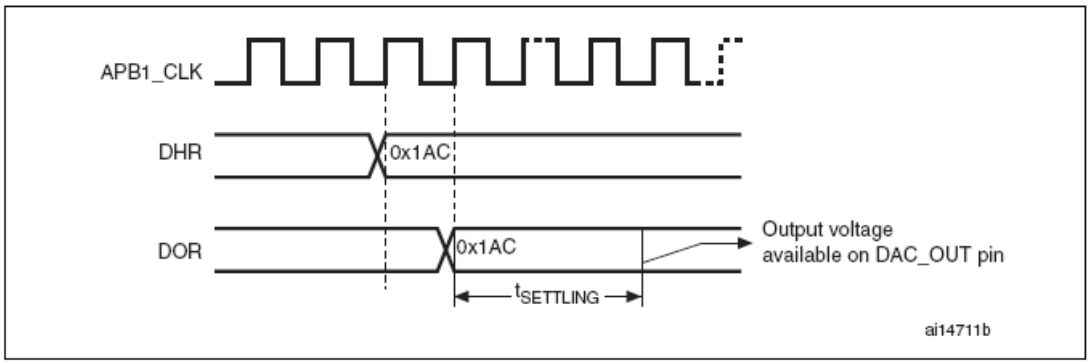
用户不可以直接对寄存器DAC_DORx写入数据，任何作为DAC通道x输出的数据都必须通过从寄存器DAC_DHRx装入(用户实际将数据写入寄存器DAC_DHR8Rx， DAC_DHR12Lx，DAC_DHR12Rx，DAC_DHR8RD，DAC_DHR12LD，或者DAC_DHR12RD)。

如果没有选中硬件触发(寄存器DAC_CR1的TENx位置'0')，存入寄存器DAC_DHRx的数据会在一个APB1时钟周期后自动传给寄存器DAC_DORx。

如果某硬件触发被选中(寄存器DAC_CR1的TENx位置'1')，数据传输在触发发生以后3个APB1时钟周期后完成。

一旦数据从寄存器DAC_DHRx装入寄存器DAC_DORx，在经过时间$t_{SETTLING}$之后，输出即有效，这段时间的长短按照电源电压和模拟输出负载的不同会有所变化。

图41 TEN =0 触发失能时转换的时间框图



11.3.5 DAC输出电压

数字输入经过DAC转换成模拟电压输出，其范围为0到V_{REF+}。

任一DAC通道管脚上的输出电压满足下面的关系：

DAC输出 = V_{REF} X DOR / 4095。

11.3.6 选择DAC触发

如果TENx位被置1，DAC转换可以由某外部事件触发(定时器计数器，外部中断线)。8个可能的事件中，由用户配置控制位TSELx[2:0]来选中其中之一触发DAC转换。

表52 外部触发

触发源	类型	TEL[2:0]
定时器6 TRGO事件	来自片上定时器的内部信号	000
定时器6 TRGO事件		001
定时器6 TRGO事件		010
定时器6 TRGO事件		011
定时器6 TRGO事件		100
定时器6 TRGO事件		101
EXTI线路9	外部管脚	110
SWTRIG（软件触发）	软件控制位	111

每次DAC接口侦测到来自选中定时器TRGO输出，或者外部中断线9的上升沿，最近存放在寄存器DAC_DHRx中的数据会被传送到寄存器DAC_DORx中。在3个APB1时钟周期后，寄存器DAC_DORx更新为新值。

如果选中软件触发，一旦SWTRIG位置'1'，转换即开始。在寄存器DAC_DORx从寄存器DAC_DHRx取得数据后，SWTRIG位由硬件自动置0。

注意： *1. TSELx[2:0]位在ENx位被置1时不能改变。*

2. 如果选中软件触发，数据从寄存器DAC_DHRx装入寄存器DAC_DORx只需要一个APB1时钟周期。

11.3.7 DMA请求

任一DAC通道都具有DMA功能。2个DMA通道可分别用于DAC通道1、2的DMA请求。

一旦有外部触发(而不是软件触发)发生，如果DMAENx位置'1'，则产生一个DMA请求。之后，寄存器DAC_DHRx的数据被传到寄存器DAC_DORx。

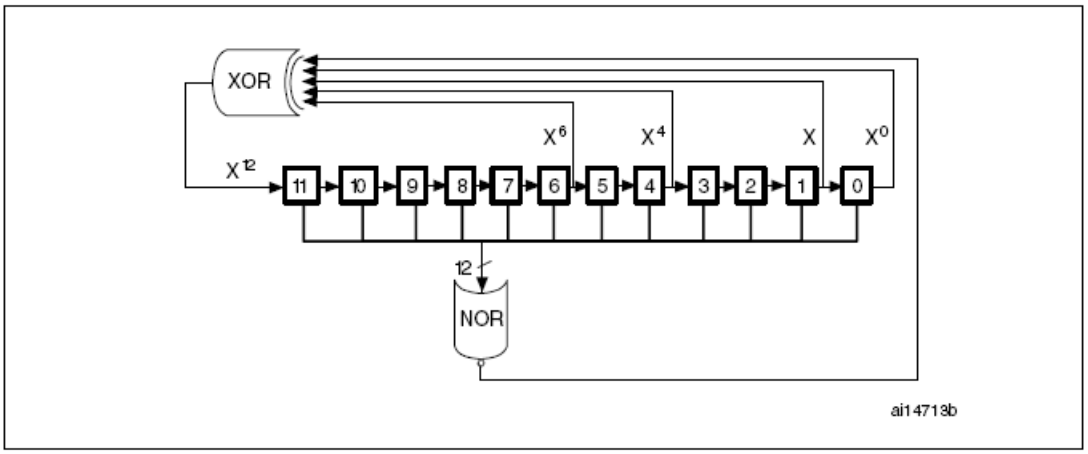
在双DAC模式下，如果2个通道的DMAENx位都被置'1'，就会产生2个DMA请求。如果用户实际只需要一个DMA传输，那么应该选择只把其中一个DMAENx位置'1'。这样，程序可以在只使用一个DMA请求，一个DMA通道的情况下，管理工作在双DAC模式的2个DAC通道。

DAC的DMA请求不会累计，因此如果第2个外部触发发生在在响应第1个外部触发之前，第2个DMA请求无效，也不会报错。

11.3.8 噪声生成

可以利用线性反馈移位寄存器(Linear Feedback Shift Register LFSR)产生幅度变化的伪噪声。通过设置WAVE[1:0]位为'01'来选中DAC噪声生成功能。寄存器LFSR的预装入值为0xAAA。按照特定算法，在每次触发事件后3个APB1时钟周期之后更新该寄存器的值。

图42 DAC LFSR 寄存器算法

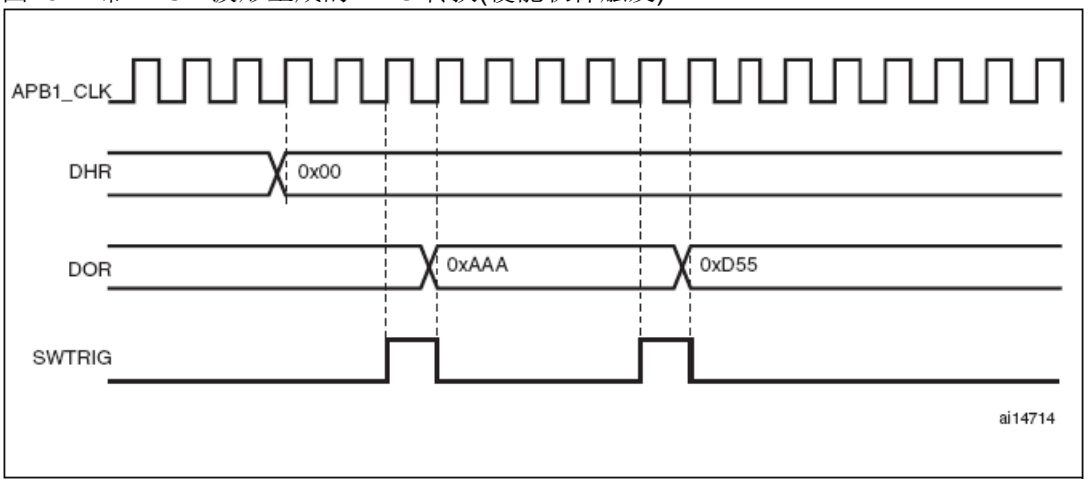


可以通过设置寄存器DAC_CR的MAMPx[3:0]位来屏蔽部分或者全部LFSR的数据。经屏蔽的LSFR值与DAC_DHRx的数据的无溢出和，即被写入寄存器DAC_DORx。

如果寄存器LFSR值为0x000，则会注入'1'(防锁定机制)。

可以通过将WAVEx[1:0]位置0来取消LFSR波形生成。

图43 带 LFSR 波形生成的 DAC 转换(使能软件触发)



注意： 为了产生噪声，必须使能DAC触发，即设DAC_CR寄存器的TENx位为1。

11.3.9 三角波生成

可以在DC或者缓慢变化的信号上加上一个小幅度的三角波。通过置WAVEx[1:0]位为'10'选中DAC的三角波生成功能。三角波的幅度可以通过设置DAC_CR寄存器的MAMPx[3:0]位来选择。内部的三角波计数器每次触发事件之后3个APB1时钟周期后累加1。计数器的值与寄存器DAC_DHRx数据的无溢出和，即被写入寄存器DAC_DORx。在传入寄存器DAC_DORx的数据值小于MAMP[3:0]位定义的最大幅度时，三角波计数器才会累加。一旦达到设置的最大幅度，计数器开始累减至0，再开始累加，周而复始。

可以通过将WAVEx[1:0]位置'0'来取消三角波生成。

图44 DAC 三角波生成

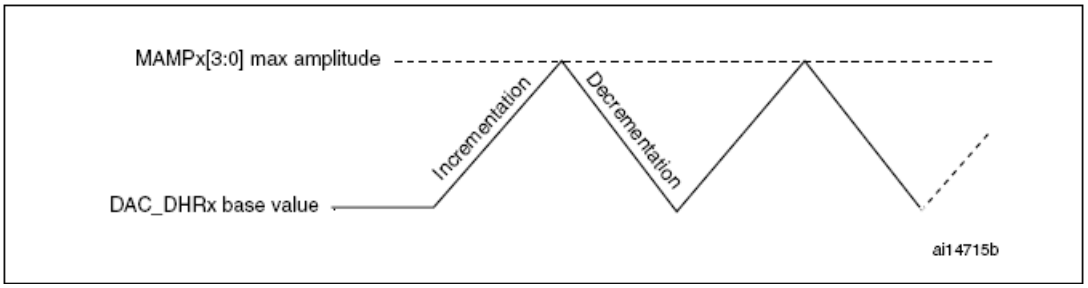
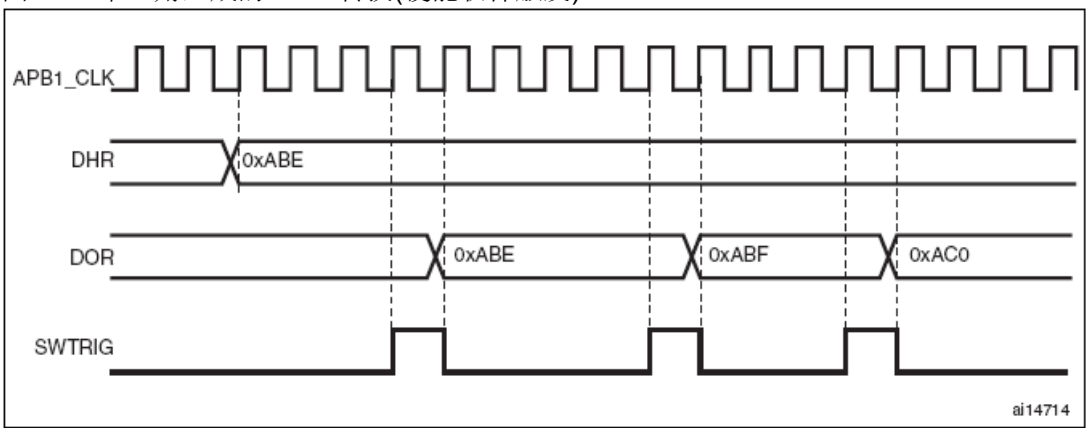


图45 带三角生成的 DAC 转换(使能软件触发)



注意： 1.为了产生三角波，必须使能DAC触发，即设DAC_CR寄存器的TENx位为1。

2. MAMP[3:0]位必须在使能DAC之前设置，否则其值不能修改。

11.4 双DAC通道转换

在需要2个DAC同时工作的情况下，为了更有效地利用总线带宽，DAC集成了3个供双DAC模式使用的寄存器：DHR8RD，DHR12RD和DHR12LD。只需要访问一个寄存器即可完成同时驱动2个DAC通道的任务。

对于双DAC通道转换和这些专用寄存器，共有11种转换模式可用。这些转换模式在只使用一个DAC通道的情况下仍然可用。

所有模式详述于以下章节。

11.4.1 无波形生成的独立触发

按照下列程序设置DAC工作在此转换模式：

- 2个DAC通道的触发使能位TENx置1
- 通过设置TSEL1[2:0]和 TSEL2[2:0]位为不同值，分别配置2个DAC通道的不同触发源。
- 将双DAC通道转换数据装入所需的DHR寄存器(DHR12RD，DHR12LD 或者DHR8RD)。

当DAC通道1触发事件发生，寄存器DHR1的值传入寄存器DAC_DOR1(3个APB1时钟周期后)。

当DAC通道2触发事件发生，寄存器DHR2的值传入寄存器DAC_DOR2(3个APB1时钟周期后)。

11.4.2 带相同LFSR生成的独立触发

按照下列程序设置DAC工作在此转换模式：

- 2个DAC通道的触发使能位TENx置1
- 通过设置TSEL1[2:0]和 TSEL2[2:0]位为不同值，分别配置2个DAC通道的不同触发源。
- 设置2个DAC通道的WAVEx[1:0]位为”01”，并设MAMPx[3:0]为相同的LFSR屏蔽值。
- 将双DAC通道转换数据装入所需的DHR寄存器(DHR12RD，DHR12LD 或者DHR8RD)。

当DAC通道1触发事件发生，带相同屏蔽的LFSR1计数器值加上DHR1寄存器的值，其和传入寄存器DAC_DOR1(3个APB1时钟周期后)。然后更新LFSR1计数器。

当DAC通道2触发事件发生，带相同屏蔽的LFSR2计数器值加上DHR2寄存器的值，其和传入寄存器DAC_DOR2(3个APB1时钟周期后)。然后更新LFSR2计数器。

11.4.3 带不同LFSR生成的独立触发

按照下列程序设置DAC工作在此转换模式：

- 2个DAC通道的触发使能位TENx置1
- 通过设置TSEL1[2:0]和 TSEL2[2:0]位为不同值，分别配置2个DAC通道的不同触发源。
- 设置2个DAC通道的WAVEx[1:0]位为”01”，并设MAMPx[3:0]为不同的LFSR屏蔽值。
- 将双DAC通道转换数据装入所需的DHR寄存器(DHR12RD，DHR12LD 或者DHR8RD)。

当DAC通道1触发事件发生，带MAMP1[3:0]所设屏蔽的LFSR1计数器值加上DHR1寄存器的值，其和传入寄存器DAC_DOR1(3个APB1时钟周期后)。然后更新LFSR1计数器。

当DAC通道2触发事件发生，带MAMP2[3:0]所设屏蔽的LFSR2计数器值加上DHR2寄存器的值，其和传入寄存器DAC_DOR2(3个APB1时钟周期后)。然后更新LFSR2计数器。

11.4.4 带相同三角波生成的独立触发

按照下列程序设置DAC工作在此转换模式：

- 2个DAC通道的触发使能位TENx置1
- 通过设置TSEL1[2:0]和 TSEL2[2:0]位为不同值，分别配置2个DAC通道的不同触发源。
- 设置2个DAC通道的WAVEx[1:0]位为”1x”，并设MAMPx[3:0]为相同的三角波幅值。
- 将双DAC通道转换数据装入所需的DHR寄存器（DHR12RD，DHR12LD 或者DHR8RD）。

当DAC通道1触发事件发生，相同的三角波幅值加上DHR1寄存器的值，其和传入寄存器DAC_DOR1(3个APB1时钟周期后)。然后更新DAC通道1三角波计数器。

当DAC通道2触发事件发生，相同的三角波幅值加上DHR2寄存器的值，其和传入寄存器DAC_DOR2(3个APB1时钟周期后)。然后更新DAC通道2三角波计数器。

11.4.5 带不同三角波生成的独立触发

按照下列程序设置DAC工作在此转换模式：

- 2个DAC通道的触发使能位TENx置’1’
- 通过设置TSEL1[2:0]和 TSEL2[2:0]位为不同值，分别配置2个DAC通道的不同触发源。
- 设置2个DAC通道的WAVEx[1:0]位为’1x’，并设MAMPx[3:0]为不同的三角波幅值。
- 将双DAC通道转换数据装入所需的DHR寄存器(DHR12RD，DHR12LD 或者DHR8RD)。

当DAC通道1触发事件发生，MAMP1[3:0]所设的三角波幅值加上DHR1寄存器的值，其和传入寄存器DAC_DOR1(3个APB1时钟周期后)。然后更新DAC通道1三角波计数器。

当DAC通道2触发事件发生，MAMP2[3:0]所设的三角波幅值加上DHR2寄存器的值，其和传入寄存器DAC_DOR2(3个APB1时钟周期后)。然后更新DAC通道2三角波计数器。

11.4.6 同时软件启动

按照下列程序设置DAC工作在此转换模式：

- 将双DAC通道转换数据装入所需的DHR寄存器(DHR12RD，DHR12LD 或者DHR8RD)。

该配置下，一个APB1时钟周期后，寄存器DHR1和DHR2的值即被分别传入寄存器DAC_DOR1和DAC_DOR2。

11.4.7 不带波形生成的同时触发

按照下列程序设置DAC工作在此转换模式：

- 2个DAC通道的触发使能位TENx置'1'
- 通过设置TSEL1[2:0]和 TSEL2[2:0]位为相同值，配置2个DAC通道有相同触发源。
- 将双DAC通道转换数据装入所需的DHR寄存器(DHR12RD，DHR12LD 或者DHR8RD)。

当触发事件发生，寄存器DHR1和DHR2的值分别传入寄存器DAC_DOR1和DAC_DOR2(3个APB1时钟周期后)。

11.4.8 带相同LFSR生成的同时触发

按照下列程序设置DAC工作在此转换模式：

- 2个DAC通道的触发使能位TENx置'1'
- 通过设置TSEL1[2:0]和 TSEL2[2:0]位为相同值，配置2个DAC通道有相同触发源。
- 设置2个DAC通道的WAVEx[1:0]位为'01'，并设MAMPx[3:0]为相同的LFSR屏蔽值。
- 将双DAC通道转换数据装入所需的DHR寄存器(DHR12RD，DHR12LD 或者DHR8RD)。

当触发事件发生，带MAMP1[3:0]所设屏蔽的LFSR1计数器值加上DHR1寄存器的值，其和传入寄存器DAC_DOR1(3个APB1时钟周期后)。然后更新LFSR1计数器。

同时，带MAMP1[3:0]所设屏蔽的LFSR2计数器值加上DHR2寄存器的值，其和传入寄存器DAC_DOR2(3个APB1时钟周期后)。然后更新LFSR2计数器。

11.4.9 带不同LFSR生成的同时触发

按照下列程序设置DAC工作在此转换模式：

- 2个DAC通道的触发使能位TENx置'1'
- 通过设置TSEL1[2:0]和 TSEL2[2:0]位为相同值，配置2个DAC通道有相同触发源。
- 设置2个DAC通道的WAVEx[1:0]位为'01'，并设MAMPx[3:0]为不同的LFSR屏蔽值。
- 将双DAC通道转换数据装入所需的DHR寄存器(DHR12RD，DHR12LD 或者DHR8RD)。

当触发事件发生，带相同屏蔽的LFSR1计数器值加上DHR1寄存器的值，其和传入寄存器DAC_DOR1(3个APB1时钟周期后)。然后更新LFSR1计数器。

同时，带相同屏蔽的LFSR2计数器值加上DHR2寄存器的值，其和传入寄存器DAC_DOR2(3个APB1时钟周期后)。然后更新LFSR2计数器。

11.4.10 带相同三角波生成的同时触发

按照下列程序设置DAC工作在此转换模式：

- 2个DAC通道的触发使能位TENx置'1'
- 通过设置TSEL1[2:0]和 TSEL2[2:0]位为相同值，配置2个DAC通道有相同触发源。
- 设置2个DAC通道的WAVEx[1:0]位为'1x'，并设MAMPx[3:0]为相同的三角波幅值。
- 将双DAC通道转换数据装入所需的DHR寄存器(DHR12RD，DHR12LD 或者DHR8RD)。

当触发事件发生，相同的三角波幅值加上DHR1寄存器的值，其和传入寄存器DAC_DOR1(3个APB1时钟周期后)。然后更新LFSR1计数器。

同时，相同的三角波幅值加上DHR2寄存器的值，其和传入寄存器DAC_DOR2(3个APB1时钟周期后)。然后更新LFSR2计数器。

11.4.11 带不同三角波生成的同时触发

按照下列程序设置DAC工作在此转换模式：

- 2个DAC通道的触发使能位TENx置'1'
- 通过设置TSEL1[2:0]和 TSEL2[2:0]位为相同值，配置2个DAC通道有相同触发源。
- 设置2个DAC通道的WAVEx[1:0]位为'1x'，并设MAMPx[3:0]为不同的三角波幅值。
- 将双DAC通道转换数据装入所需的DHR寄存器(DHR12RD，DHR12LD 或者DHR8RD)。

当触发事件发生，MAMP1[3:0]所设的三角波幅值加上DHR1寄存器的值，其和传入寄存器DAC_DOR1(3个APB1时钟周期后)。然后更新LFSR1计数器。

同时，MAMP2[3:0]所设的三角波幅值加上DHR2寄存器的值，其和传入寄存器DAC_DOR2(3个APB1时钟周期后)。然后更新LFSR2计数器。

11.5 DAC寄存器

11.5.1 DAC控制寄存器(DAC_CR)

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			DMAEN2	MAMP2[3:0]				WAVE2[2:0]		TSEL2[2:0]			TEN2	BOFF2	EN2
			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			DMAEN1	MAMP13:0]				WAVE1[2:0]		TSEL1[2:0]			TEN1	BOFF1	EN1
			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位31:29	保留。
位28	**DMAEN2**：DAC通道2 DMA使能 该位由软件设置和清除。 0：DAC通道2 DMA模式失能 1：DAC通道2 DMA模式使能
位27:24	**MAMP2[3:0]**：DAC通道2屏蔽/幅值选择器 由软件设置该位，用来在噪声生成模式下选择屏蔽位，在三角波生成模式下选择波形的幅值 0000：不屏蔽LSFR位0 / 三角波幅值等于1 0001：不屏蔽LSFR位[1:0] / 三角波幅值等于3 0010：不屏蔽LSFR位[2:0] / 三角波幅值等于7 0011：不屏蔽LSFR位[3:0] / 三角波幅值等于15 0100：不屏蔽LSFR位[4:0] / 三角波幅值等于31 0101：不屏蔽LSFR位[5:0] / 三角波幅值等于63 0110：不屏蔽LSFR位[6:0] / 三角波幅值等于127 0111：不屏蔽LSFR位[7:0] / 三角波幅值等于255 1000：不屏蔽LSFR位[8:0] / 三角波幅值等于511 1001：不屏蔽LSFR位[9:0] / 三角波幅值等于1023 1010：不屏蔽LSFR位[10:0] / 三角波幅值等于2047 ≥1011：不屏蔽LSFR位[11:0] / 三角波幅值等于4095
位23:22	**WAVE2[1:0]**：DAC通道2噪声/三角波生成使能 该位由软件设置和清除。 00：失能波形生成 10：噪声波形生成 1x：三角波生成
位21:19	**TSEL2[2:0]**：DAC通道2触发选择 该位用于选择DAC通道2的外部触发事件。 000：TIM6 TRGO事件 000：TIM8 TRGO事件 000：TIM7 TRGO事件 000：TIM5 TRGO事件 000：TIM2 TRGO事件 000：TIM4 TRGO事件 000：外部中断线9 000：软件触发 注意：该位只能在TEN2 = 1（DAC通道2触发使能）时设置

位18	**TEN2**：DAC通道2触发使能 该位由软件设置和清除，用来使能/失能DAC通道2的触发。 0：DAC通道2 触发失能，写入寄存器DAC_DHRx的数据在1个APB1时钟周期后传入寄存器DAC_DORx 1：DAC通道2 触发使能，写入寄存器DAC_DHRx的数据在3个APB1时钟周期后传入寄存器DAC_DORx 注意：如果选择软件触发，写入寄存器DAC_DHRx的数据只需要1个APB1时钟周期就可以传入寄存器DAC_DORx
位17	**BOFF2**：DAC通道2输出缓存失能 该位由软件设置和清除，用来使能/失能DAC通道2的输出缓存。 0：DAC通道2输出缓存使能 1：DAC通道2输出缓存失能
位16	**EN2**：DAC通道2使能 该位由软件设置和清除，用来使能/失能DAC通道2。 0：DAC通道2失能 1：DAC通道2使能
位15:13	保留。
位12	**DMAEN1**：DAC通道1 DMA使能 该位由软件设置和清除。 0：DAC通道1 DMA模式失能 1：DAC通道1 DMA模式使能
位11:8	**MAMP1[3:0]**：DAC通道1屏蔽/幅值选择器 由软件设置该位，用来在噪声生成模式下选择屏蔽位，在三角波生成模式下选择波形的幅值 0000：不屏蔽LSFR位0 / 三角波幅值等于1 0001：不屏蔽LSFR位[1:0] / 三角波幅值等于3 0010：不屏蔽LSFR位[2:0] / 三角波幅值等于7 0011：不屏蔽LSFR位[3:0] / 三角波幅值等于15 0100：不屏蔽LSFR位[4:0] / 三角波幅值等于31 0101：不屏蔽LSFR位[5:0] / 三角波幅值等于63 0110：不屏蔽LSFR位[6:0] / 三角波幅值等于127 0111：不屏蔽LSFR位[7:0] / 三角波幅值等于255 1000：不屏蔽LSFR位[8:0] / 三角波幅值等于511 1001：不屏蔽LSFR位[9:0] / 三角波幅值等于1023 1010：不屏蔽LSFR位[10:0] / 三角波幅值等于2047 ≥1011：不屏蔽LSFR位[11:0] / 三角波幅值等于4095
位7:6	**WAVE1[1:0]**：DAC通道1噪声/三角波生成使能 该位由软件设置和清除。 00：失能波形生成 10：噪声波形生成 1x：三角波生成
位5:3	**TSEL1[2:0]**：DAC通道1触发选择 该位用于选择DAC通道1的外部触发事件。 000：TIM6 TRGO事件 000：TIM8 TRGO事件 000：TIM7 TRGO事件 000：TIM5 TRGO事件 000：TIM2 TRGO事件 000：TIM4 TRGO事件 000：外部中断线9 000：软件触发 注意：该位只能在TEN1= 1（DAC通道1触发使能）时设置

位2	**TEN1**：DAC通道1触发使能 该位由软件设置和清除，用来使能/失能DAC通道2的触发。 0：DAC通道1 触发失能，写入寄存器DAC_DHRx的数据在1个APB1时钟周期后传入寄存器DAC_DORx 1：DAC通道1 触发使能，写入寄存器DAC_DHRx的数据在3个APB1时钟周期后传入寄存器DAC_DORx 注意：如果选择软件触发，写入寄存器DAC_DHRx的数据只需要1个APB1时钟周期就可以传入寄存器DAC_DORx
位1	**BOFF1**：DAC通道1输出缓存失能 该位由软件设置和清除，用来使能/失能DAC通道1的输出缓存。 0：DAC通道1输出缓存使能 1：DAC通道1输出缓存失能
位0	**EN1**：DAC通道1使能 该位由软件设置和清除，用来使能/失能DAC通道1。 0：DAC通道1失能 1：DAC通道1使能

11.5.2 DAC软件触发寄存器(DAC_SWTRIGR)

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														SWTRIG2	SWTRIG1
														w	w

位31:2	保留。
位1	**SWTRIG2**：DAC通道2 软件触发 该位由软件设置和清除，用来使能 / 失能软件触发。 0：DAC通道2软件触发失能 1：DAC通道2软件触发使能 注意：一旦寄存器DAC_DHR2的数据传入寄存器DAC_DOR2，该位由硬件置'0'(1个APB1时钟周期后)。
位0	**SWTRIG1**：DAC通道1 软件触发 该位由软件设置和清除，用来使能 / 失能软件触发。 0：DAC通道1软件触发失能 1：DAC通道1软件触发使能 注意：一旦寄存器DAC_DHR1的数据传入寄存器DAC_DOR1，该位由硬件置'0'(1个APB1时钟周期后)。

11.5.3 DAC通道 1 的 12 位右对齐数据保持寄存器(DAC_DHR12R1)

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
保留				DACC1DHR[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位31:12	保留。
位11:0	**DACC1DHR[11:0]**：DAC通道1的12位右对齐数据 该位由软件写入，表示DAC通道1的12位数据

11.5.4 DAC通道 1 的 12 位左对齐数据保持寄存器(DAC_DHR12L1)

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															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DACC1DHR[11:0]												保留			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw				

位31:16	保留。
位15:4	**DACC1DHR[11:0]**：DAC通道1的12位左对齐数据 该位由软件写入，表示DAC通道1的的12位数据
位3:0	保留。

11.5.5 DAC通道 1 的 8 位右对齐数据保持寄存器(DAC_DHR8R1)

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															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留								DACC1DHR[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

位31:18	保留。
位7:0	**DACC1DHR[7:0]**：DAC通道1的8位右对齐数据 该位由软件写入，表示DAC通道1的的8位数据

11.5.6 DAC通道 2 的 12 位右对齐数据保持寄存器(DAC_DHR12R2)

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
保留				DACC2DHR[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位31:12	保留。
位11:0	**DACC2DHR[11:0]**：DAC通道2的12位右对齐数据 该位由软件写入，表示DAC通道2的12位数据

11.5.7 DAC通道 2 的 12 位左对齐数据保持寄存器(DAC_DHR12L2)

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															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DACC2DHR[11:0]												保留			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw				

位31:16	保留。
位15:4	**DAC2DHR[11:0]**：DAC通道2的12位左对齐数据 该位由软件写入，表示DAC通道2的的12位数据
位3:0	保留。

11.5.8 DAC通道 2 的 8 位右对齐数据保持寄存器(DAC_DHR8R2)

地址偏移：0x1C

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								DACC2DHR[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw				

位31:18	保留。
位7:0	**DACC2DHR[7:0]**：DAC通道2的8位右对齐数据 该位由软件写入，表示DAC通道2的的8位数据

11.5.9 双DAC的 12 位右对齐数据保持寄存器(DAC_DHR12RD)

地址偏移：0x20

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留				DACC2DHR[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留				DACC1DHR[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位31:28	保留。
位27:16	**DACC2DHR[11:0]**：DAC通道2的12位右对齐数据 该位由软件写入，表示DAC通道2的12位数据
位15:12	保留。
位11:0	**DACC1DHR[11:0]**：DAC通道1的12位右对齐数据 该位由软件写入，表示DAC通道2的12位数据

11.5.10 双DAC的 12 位左对齐数据保持寄存器(DAC_DHR12LD)

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
DACC2DHR[11:0]												保留			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DACC1DHR[11:0]												保留			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw				

位31:20	**DACC2DHR[11:0]**：DAC通道2的12位左对齐数据 该位由软件写入，表示DAC通道2的12位数据
位19:16	保留。
位15:4	**DACC1DHR[11:0]**：DAC通道1的12位左对齐数据 该位由软件写入，表示DAC通道1的12位数据
位3:0	保留。

11.5.11 双DAC的 8 位右对齐数据保持寄存器(DAC_DHR8RD)

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
DACC2DHR[7:0]								DACC1DHR[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位31:16	保留。

位15:8	**DACC2DHR[7:0]**：DAC通道2的8位右对齐数据 该位由软件写入，表示DAC通道2的的8位数据
位7:0	**DACC1DHR[7:0]**：DAC通道1的8位右对齐数据 该位由软件写入，表示DAC通道1的的8位数据

11.5.12 DAC通道 1 数据输出寄存器(DAC_DOR1)

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															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留				DACC1DOR[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位31:12	保留。
位11:0	**DACC1DOR[11:0]**：DAC通道1 输出数据 该位由软件写入，表示DAC通道1的输出数据

11.5.13 DAC通道 2 数据输出寄存器(DAC_DOR2)

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															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留				DACC2DOR[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位31:12	保留。
位11:0	**DACC2DOR[11:0]**：DAC通道2 输出数据 该位由软件写入，表示DAC通道2的输出数据

11.5.14 DAC寄存器映像

表53 DAC 寄存器映像

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	DAC_CR	保留			DMAEN2	MAMP2[3:0]				WAVE2[1:0]		TSEL2[2:0]			TEN2	BOFF2	EN2	保留			DMAEN1	MAMP1[3:0]				WAVE1[1:0]		TSEL1[2:0]			TEN1	BOFF1	EN1
	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
0x04	DAC_SWTRIGR	保留																														SWTRIG2	SWTRIG1
	复位值																															0	0
0x08	DAC_DHR12R1	保留																			DACC1DHR[11:0]												
	复位值																				0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	DAC_DHR12L1	保留																DACC1DHR[11:0]														保留	
	复位值																	0	0	0	0	0	0	0	0	0	0						
0x10	DAC_DHR8R1	保留																								DACC1DHR[7:0]							
	复位值																									0	0	0	0	0	1	1	1
0x14	DAC_DHR12R2	保留																			DACC2DHR[11:0]												
	复位值																				0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	DAC_DHR12L2	保留																DACC2DHR[11:0]														保留	
	复位值																	0	0	0	0	0	0	0	0	0	0						
0x1C	DAC_DHR8R2	保留																								DACC2DHR[7:0]							
	复位值																									0	0	0	0	0	1	1	1
0x20	DAC_DHR12LD	保留				DACC2DHR[11:0]												保留			DACC1DHR[11:0]												
	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
0x24	DAC_DHR12RD	DACC2DHR[11:0]											保留					DACC1DHR[11:0]														保留	
	复位值	0	0	0	0	0	0	0	0	0	0	0						0	0	0	0	0	0	0	0	0	0						
0x28	DAC_DHR8RD	保留																DACC2DHR[7:0]								DACC1DHR[7:0]							
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1
0x2C	DAC_DOR1	保留																			DACC1DOR[11:0]												
	复位值																				0	0	0	0	0	0	0	0	0	0	0	0	0
0x30	DAC_DOR2	保留																			DACC2DOR[11:0]												
	复位值																				0	0	0	0	0	0	0	0	0	0	0	0	0

注意：寄存器的基地址请查询表1

12 高级控制定时器(TIM1和TIM8)

小容量产品是指闪存存储器容量在16K至32K字节之间的STM32F101xx、STM32F102xx和STM32F103xx微控制器。

中容量产品是指闪存存储器容量在64K至128K字节之间的STM32F101xx、STM32F102xx和STM32F103xx微控制器。

大容量产品是指闪存存储器容量在256K至512K字节之间的STM32F101xx和STM32F103xx微控制器。

小容量和中容量产品的STM32F103xx含有一个高级控制定时器(TIM1)，而大容量产品的STM32F103xx含有二个高级控制定时器(TIM1和TIM8)。

12.1 TIM1和TIM8简介

高级控制定时器(TIM1和TIM8)由一个16位的自动装载计数器组成，它由一个可编程的预分频器驱动。

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。

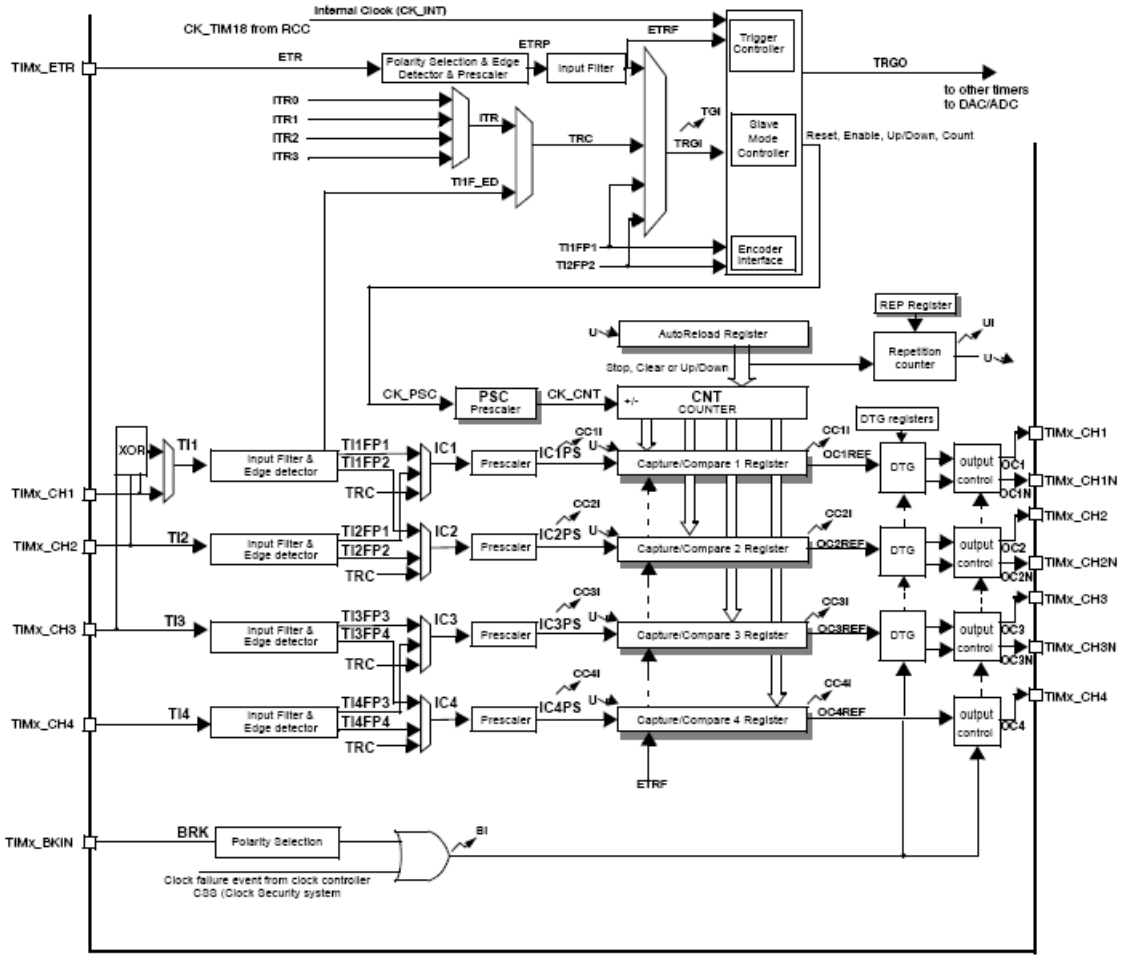
高级控制定时器(TIM1和TIM8)和通用定时器(TIMx)是完全独立的，它们不共享任何资源。它们可以同步操作，具体描述参看12.3.20节。

12.2 TIM1和TIM8主要特性

TIM1和TIM8定时器的功能包括：

- 16位向上、向下、向上/下自动装载计数器
- 16位可编程(可以实时修改)预分频器，计数器时钟频率的分频系数为1～65535之间的任意数值
- 多达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联的同步电路
- 允许在指定数目的计数器周期之后更新定时器寄存器的重复计数器
- 刹车输入信号可以将定时器输出信号置于复位状态或者一个已知状态
- 如下事件发生时产生中断/DMA：
  - 更新：计数器向上溢出/向下溢出，计数器初始化(通过软件或者内部/外部触发)
  - 触发事件(计数器启动、停止、初始化或者由内部/外部触发计数)
  - 输入捕获
  - 输出比较
  - 刹车信号输入
- 支持针对定位的增量（正交）编码器和霍尔传感器电路
- 触发输入作为外部时钟或者按周期的电流管理

图46 高级控制定时器框图



注： Reg *根据控制位的设定，在U事件时传送预加载寄存器的内容至工作寄存器*

事件

中断和DMA输出

12.3 TIM1和TIM8功能描述

12.3.1 时基单元

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

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

时基单元包含：

- 计数器寄存器(TIMx_CNT)
- 预分频器寄存器 (TIMx_PSC)
- 自动装载寄存器 (TIMx_ARR)
- 重复次数寄存器 (TIMx_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TIMx_CR1寄存器中的自动装载预装载使能位(ARPE)的设置，预装载寄存器的内容被立即或在每次的更新事件UEV时传送到影子寄存器。当计数器达到溢出条件(向下计数时的下溢条件)并当

TIMx_CR1寄存器中的UDIS位等于0时，产生更新事件。更新事件也可以由软件产生。随后会详细描述每一种配置下更新事件的产生。

计数器由预分频器的时钟输出CK_CNT驱动，仅当设置了计数器TIMx_CR1寄存器中的计数器使能位(CEN)时，CK_CNT才有效。(更多有关使能计数器的细节，请参见控制器的从模式描述)。

注意，在设置了TIMx_CR寄存器的CEN位的一个时钟周期后，计数器开始计数。

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，它能够在运行时被改变。新的预分频器的参数在下一次更新事件到来时被采用。

图47和图48给出了在预分频器运行时，更改计数器参数的例子。

图47 当预分频器的参数从1变到2时，计数器的时序图

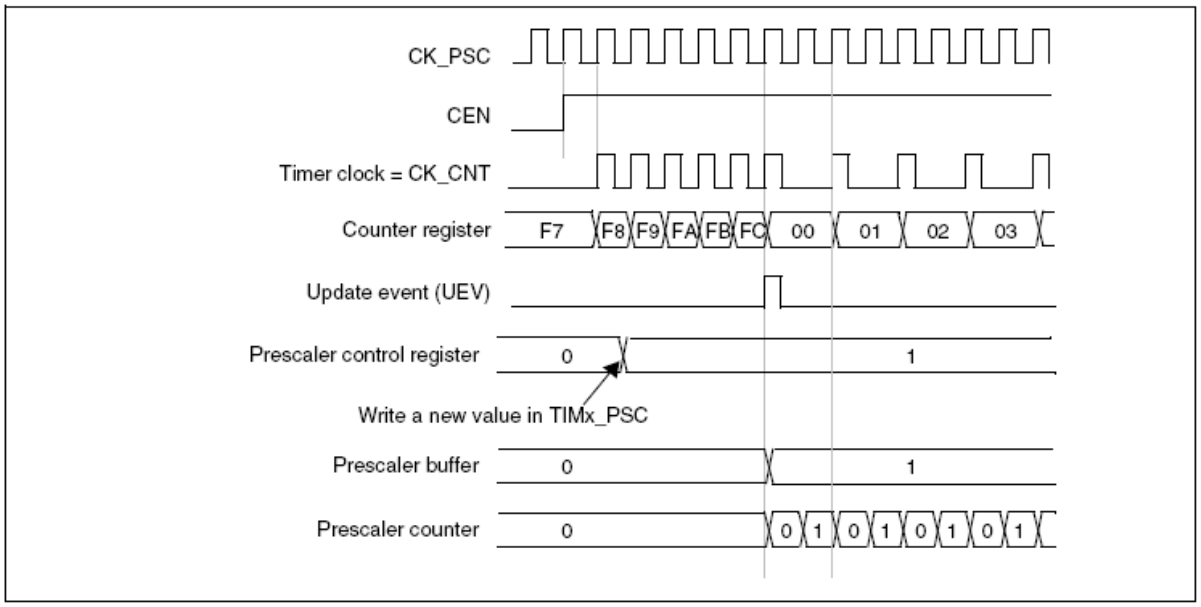
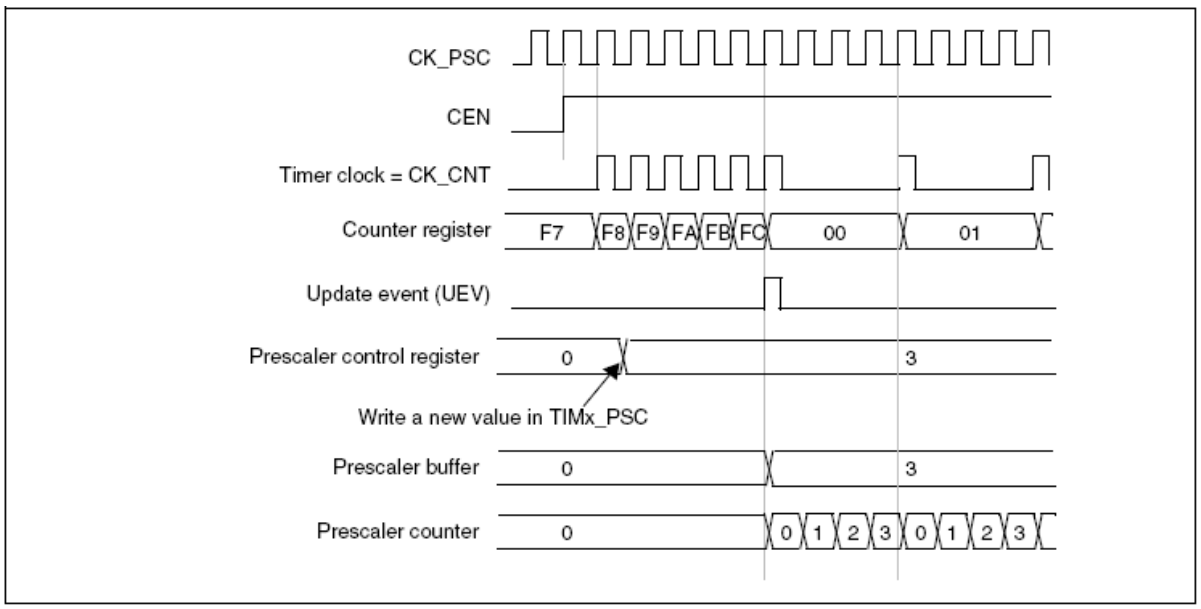


图48 当预分频器的参数从1变到4时，计数器的时序图



12.3.2 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。

如果使用了重复计数器功能，在向上计数达到设置的重复计数次数(TIMx_RCR)时，产生更新事件(UEV)；否则每次计数器溢出时才产生更新事件。

在TIMx_EGR寄存器中设置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。

设置TIMx_CR1寄存器中的UDIS位，可以禁止更新事件；这样可以避免在向预装载寄存器中写入新值时更新影子寄存器。在UDIS位被清0之前，将不产生更新事件。但是在应该产生更新事件时，计数器仍会被清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，如果设置了TIMx_CR1寄存器中的URS位(选择更新请求)，设置UG位将产生一个更新事件UEV，但硬件不设置UIF标志(即不产生中断或DMA请求)。这是为了避免在捕获模式下清除计数器时，同时产生更新和捕获中断。

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

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

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

图49 计数器时序图，内部时钟分频因子为1

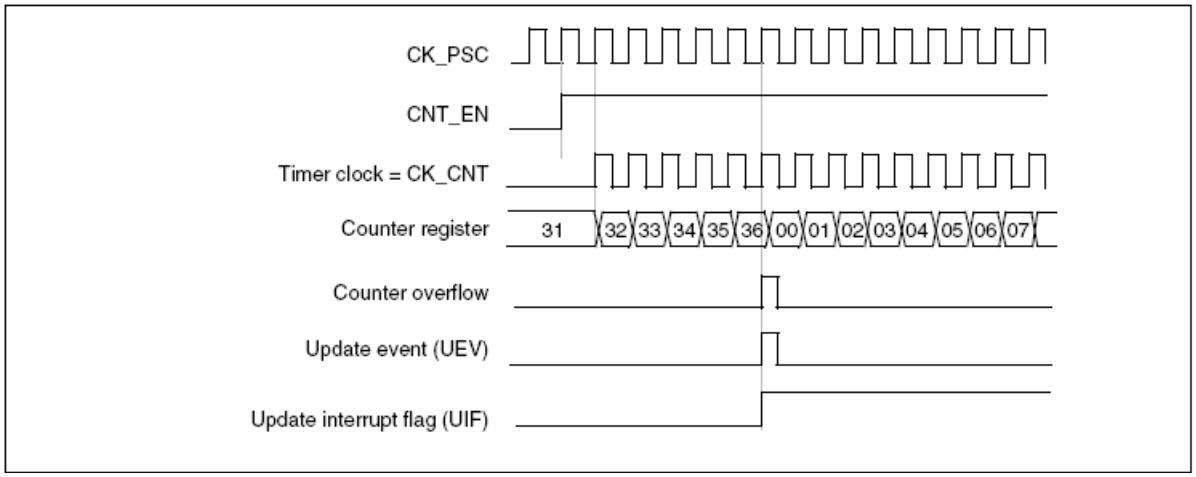


图50 计数器时序图，内部时钟分频因子为2

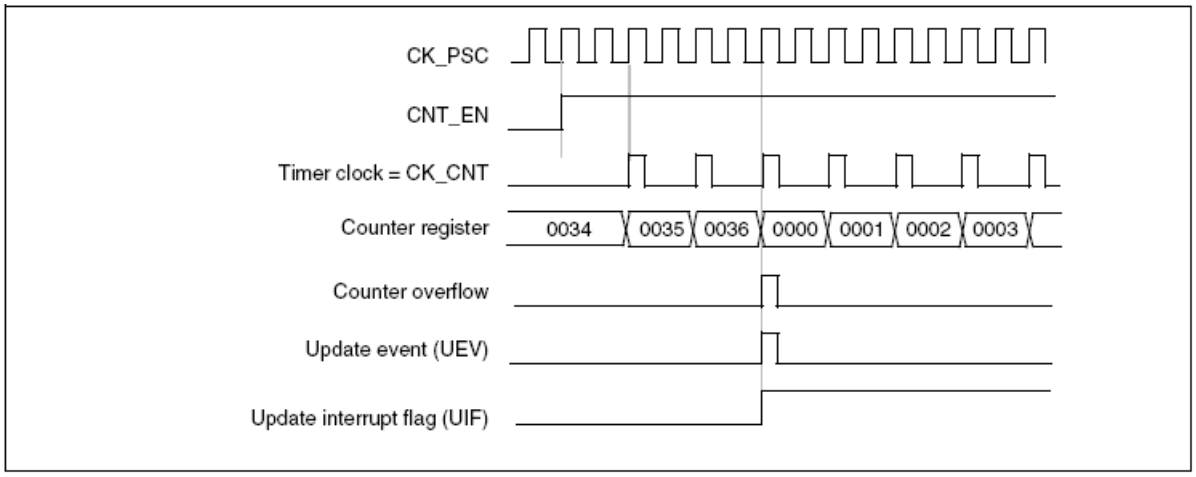


图51 计数器时序图，内部时钟分频因子为4

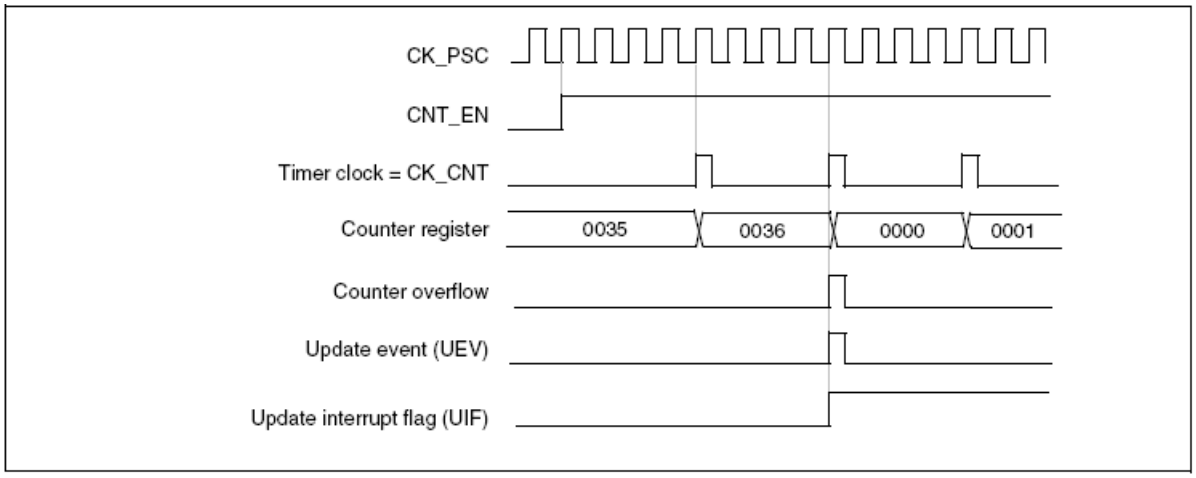


图52 计数器时序图，内部时钟分频因子为N

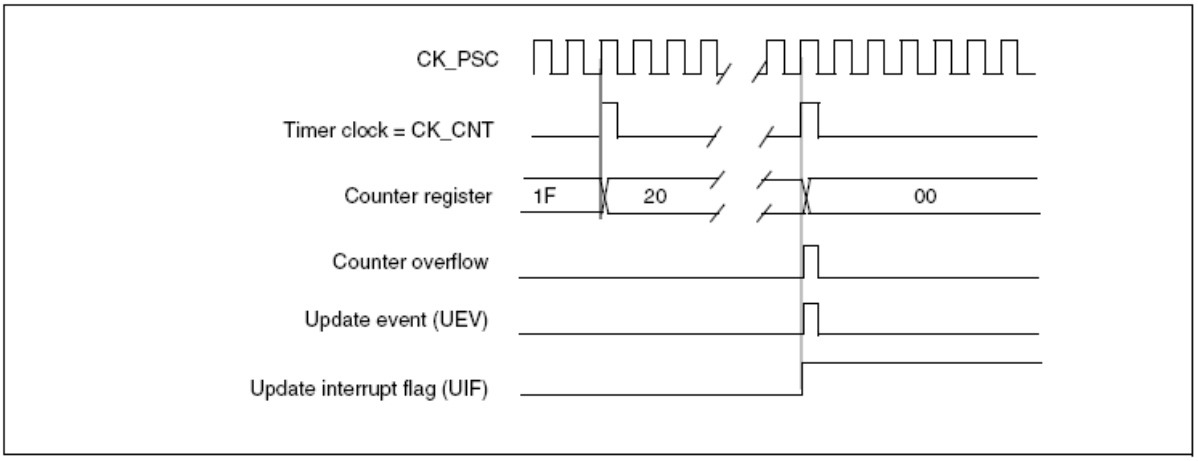


图53 计数器时序图，当ARPE=0时的更新事件(TIMx_ARR没有预装入)

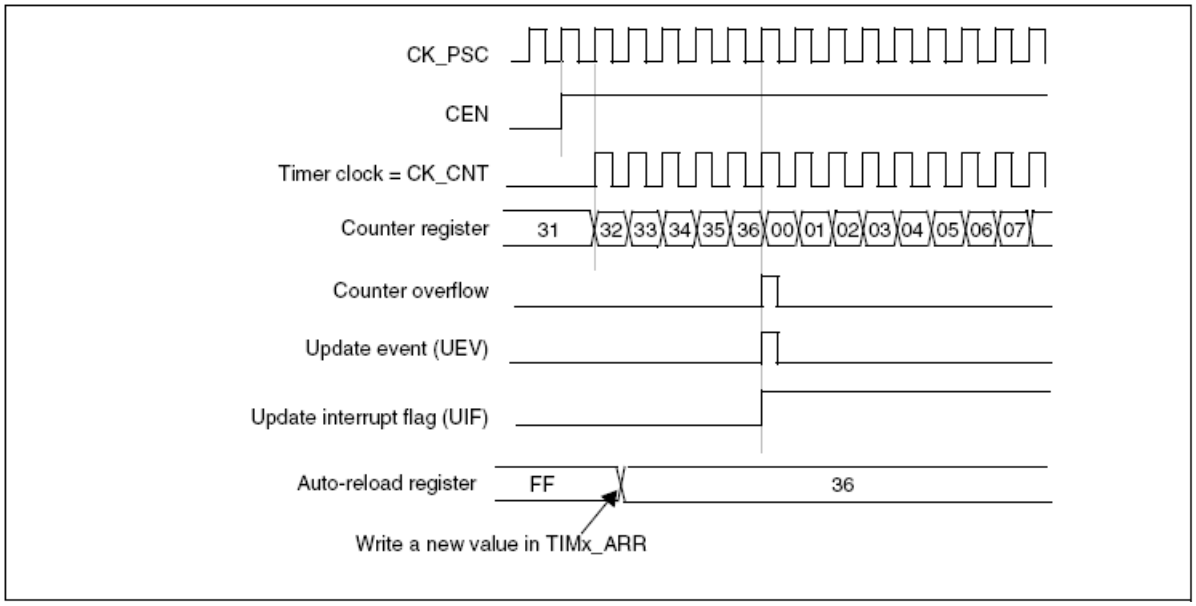
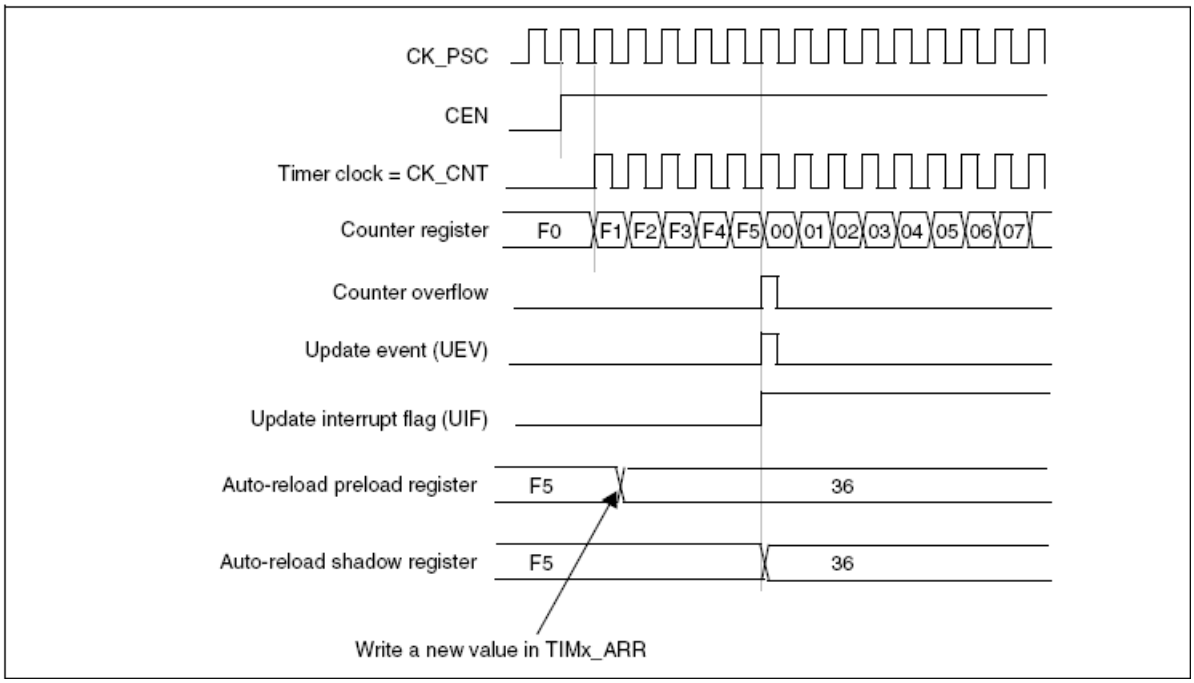


图54 计数器时序图，当ARPE=1时的更新事件(预装入了TIMx_ARR)



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。

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

在TIMx_EGR寄存器中设置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。

设置TIMx_CR1寄存器中的UDIS位可以禁止UEV事件。这样可以避免在向预装载寄存器中写入新值时更新影子寄存器。因此UDIS位被清为0之前不会产生更新事件。然而，计数器仍会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(选择更新请求) ，设置UG位将产生一个更新事件UEV但不设置UIF标志(因此不产生中断和DMA请求)，这是为了避免在发生捕获事件并清除计数器时，同时产生更新和捕获中断。

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

- 重复计数器被重置为TIMx_RCR寄存器中的内容
- 预分频器的缓存器被加载为预装载的值(TIMx_PSC寄存器的值)。
- 当前的自动加载寄存器被更新为预装载值(TIMx_ARR寄存器中的内容)。注：自动装载在计数器重载入之前被更新，因此下一个周期将是预期的值。

以下是一些当TIMx_ARR=0x36时，计数器在不同时钟频率下的操作例子。

图55 计数器时序图，内部时钟分频因子为1

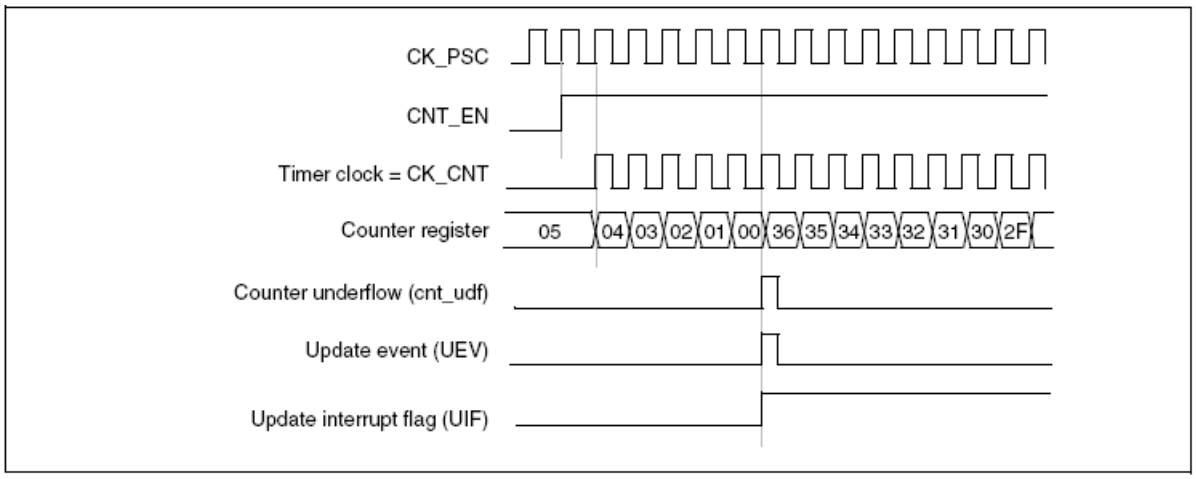


图56 计数器时序图，内部时钟分频因子为2

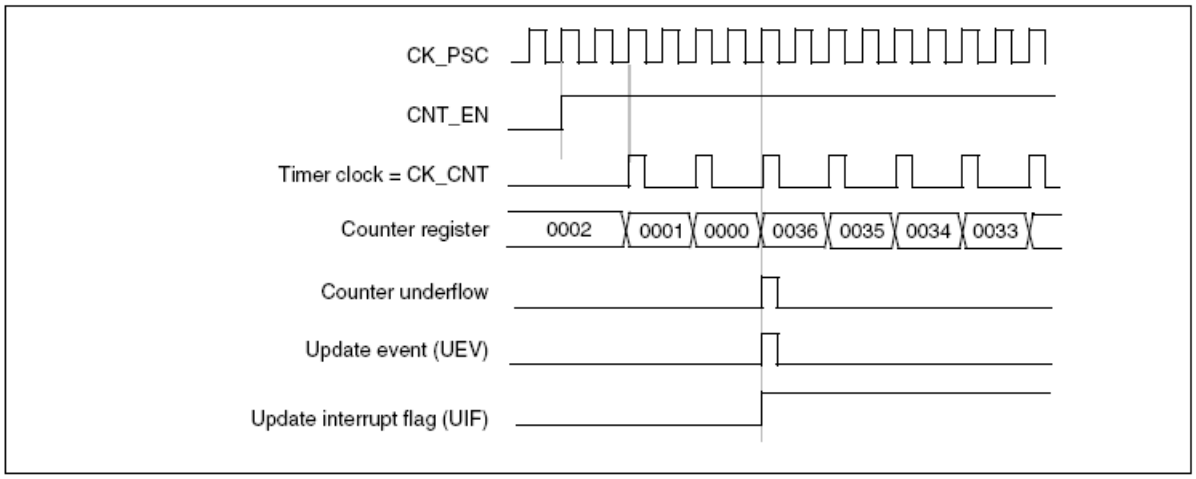


图57 计数器时序图，内部时钟分频因子为4

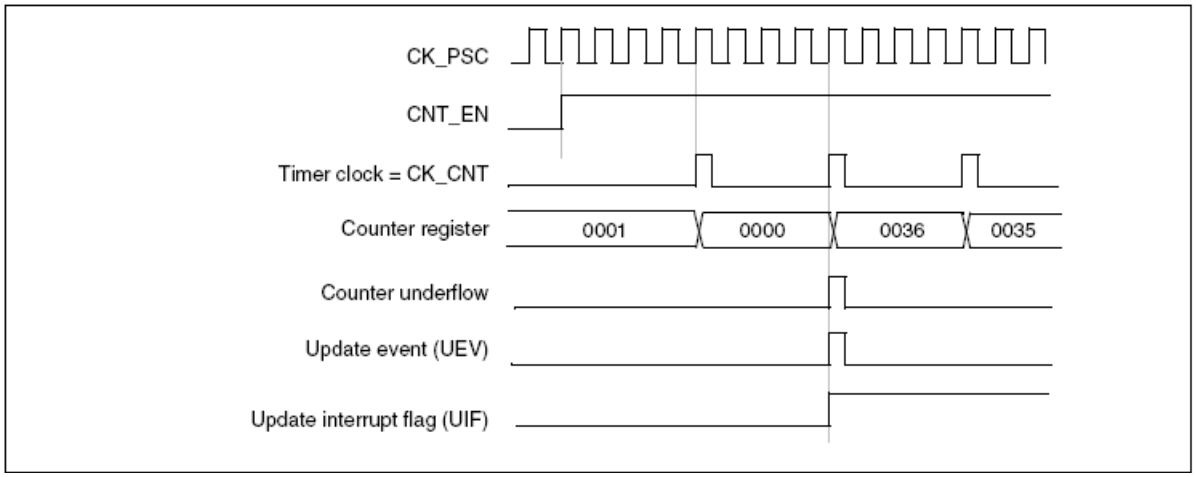


图58 计数器时序图，内部时钟分频因子为N

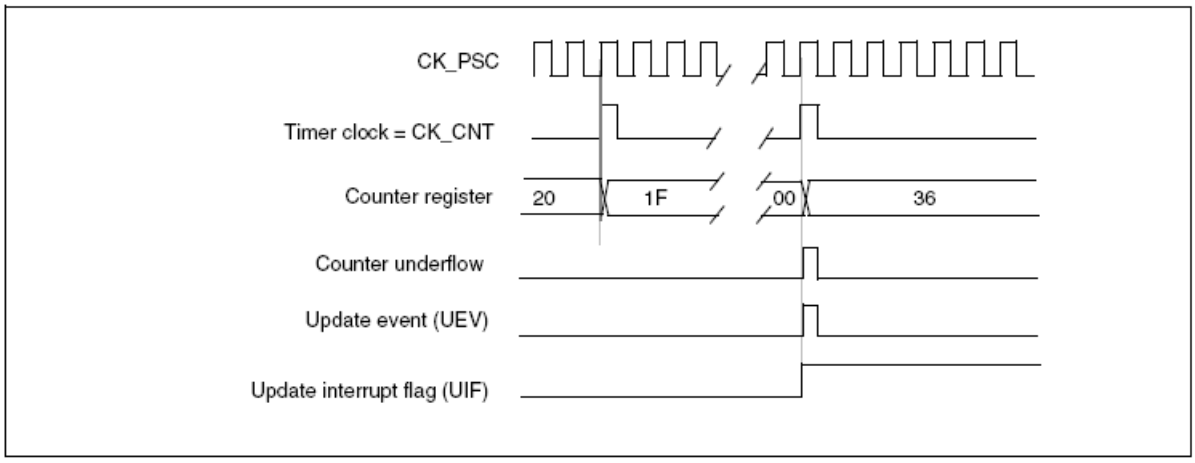
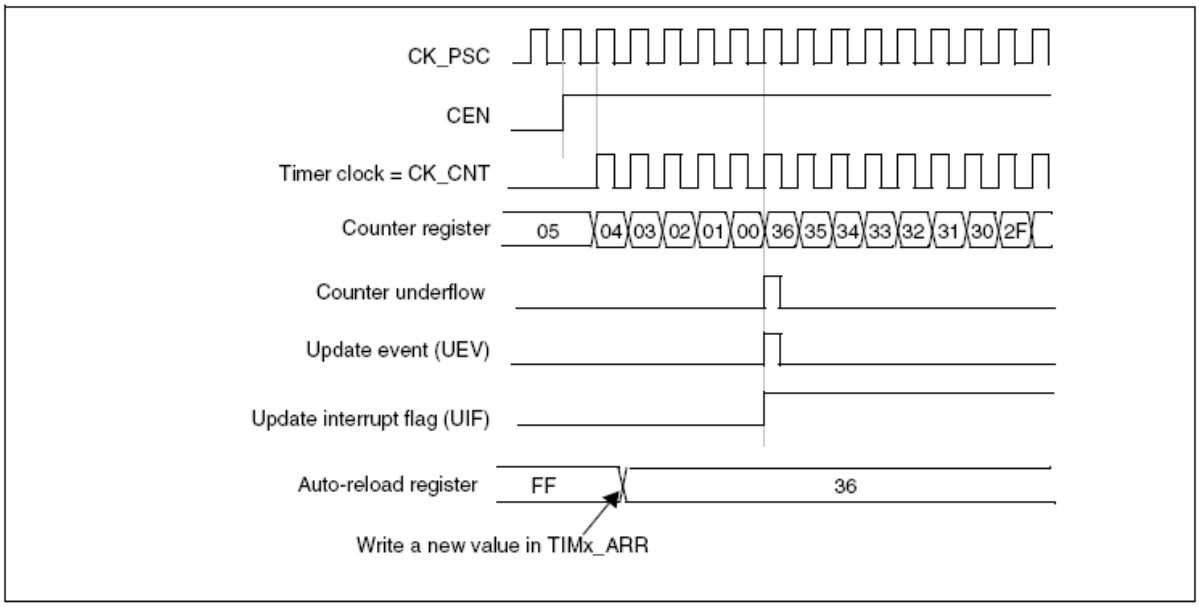


图59 计数器时序图，当没有使用重复计数器时的更新事件



中央对齐模式(向上/向下计数)

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

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

更新事件可以产生在每次计数上溢和每次计数下溢；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位产生。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。

设置TIMx_CR1寄存器中的UDIS位可以禁止UEV事件。这样可以避免在向预装载寄存器中写入新值时更新影子寄存器。因此UDIS位被清为0之前不会产生更新事件。然而，计数器仍会根据当前自动重加载的值，继续向上或向下计数。

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

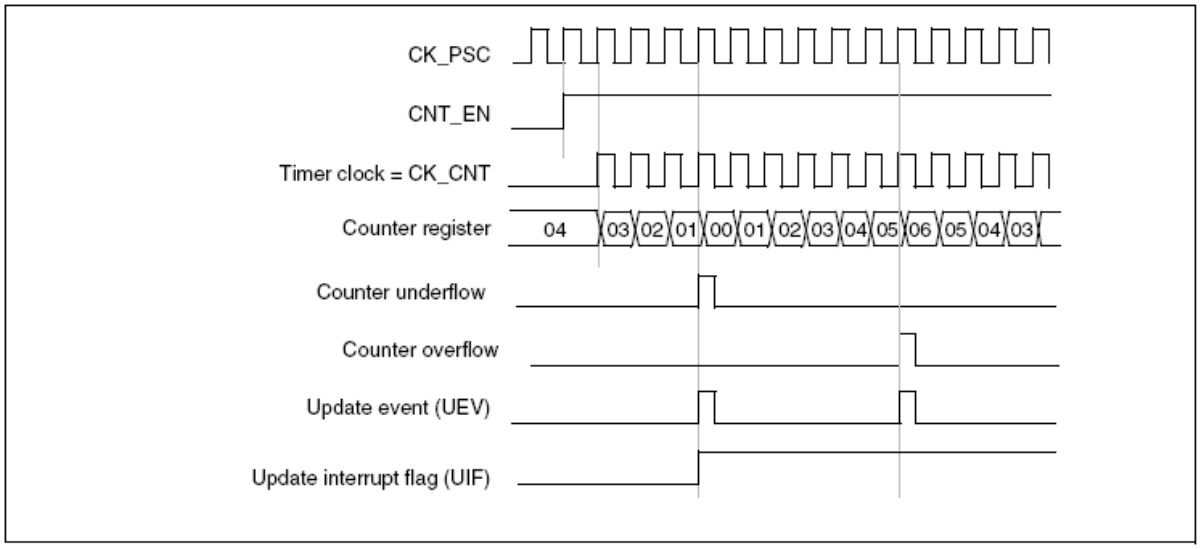
当发生更新事件时，所有的寄存器都被更新，并且(根据URS位的设置)更新标志位(TIMx_SR寄存器中的UIF位)也被设置。

- 重复计数器被重置为TIMx_RCR寄存器中的内容
- 预分频器的缓存器被加载为预装载(TIMx_PSC寄存器)的值。

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

以下是一些计数器在不同时钟频率下的操作的例子：

图60 计数器时序图，内部时钟分频因子为1，TIMx_ARR=0x6



1. 这里使用了中心对齐模式 1(详见12.4.1节)。

图61 计数器时序图，内部时钟分频因子为2

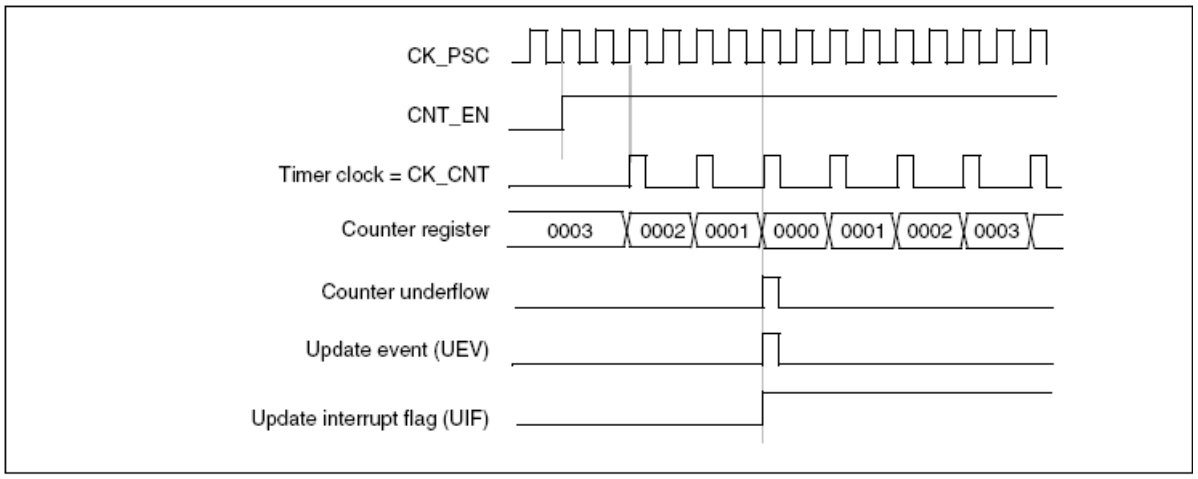


图62 计数器时序图，内部时钟分频因子为4，TIMx_ARR=0x36

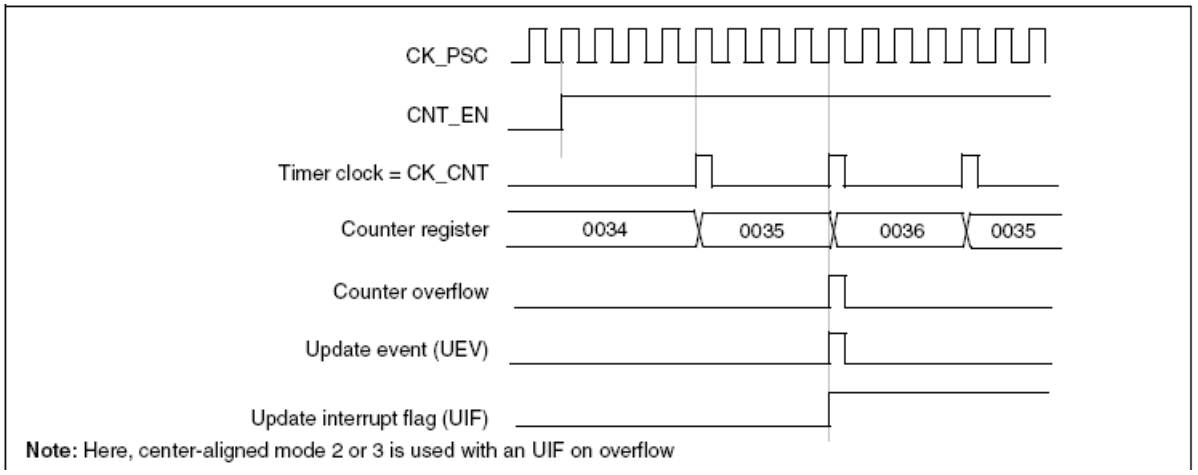


图63 计数器时序图，内部时钟分频因子为N

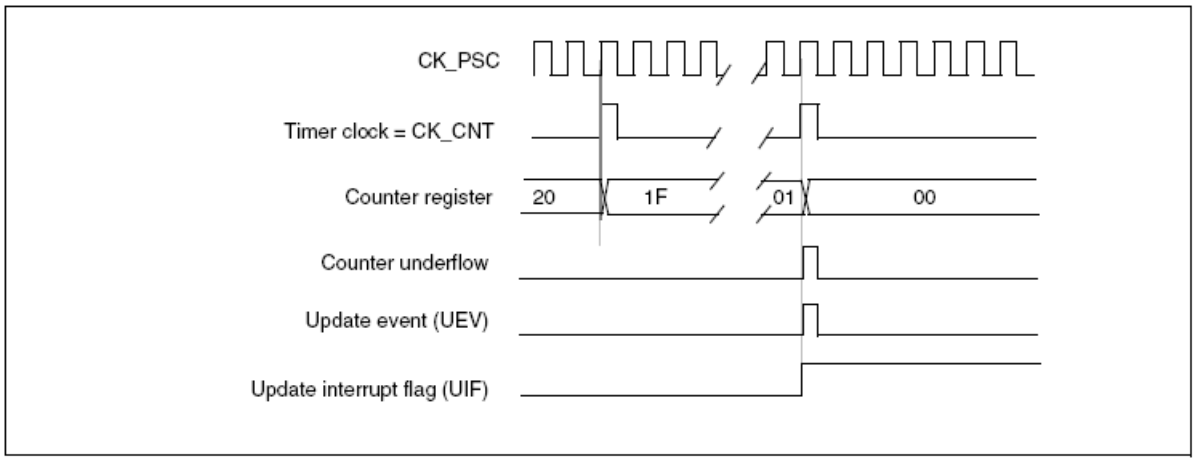


图64 计数器时序图，ARPE=1时的更新事件(计数器下溢)

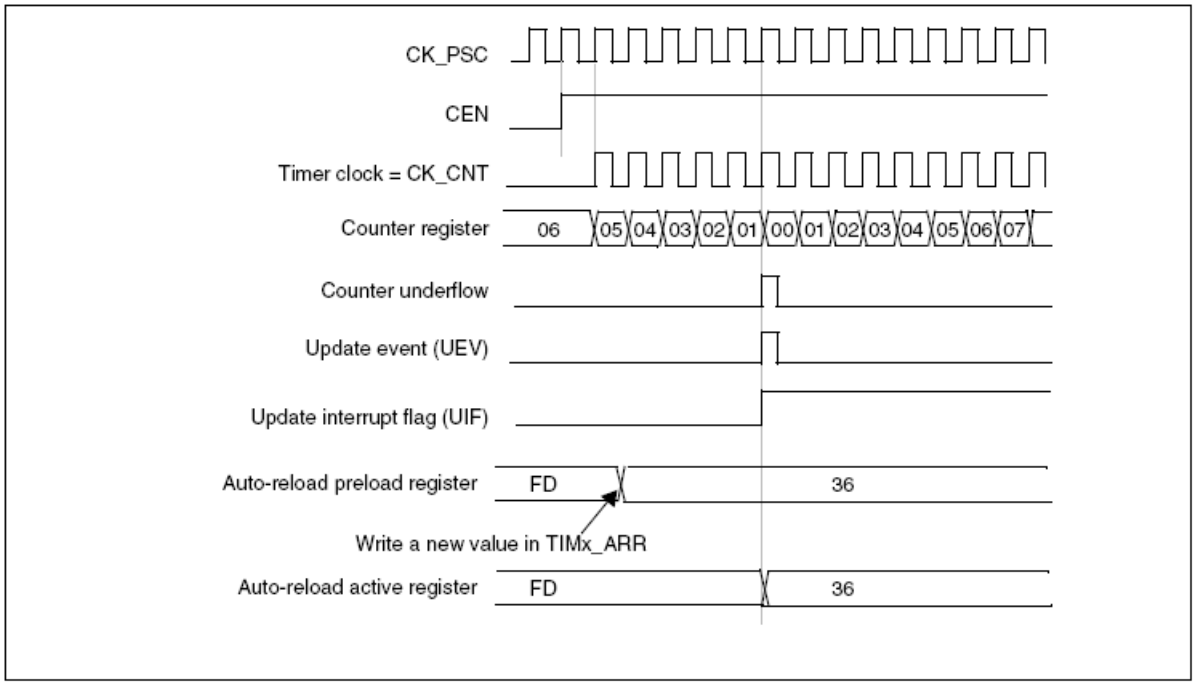
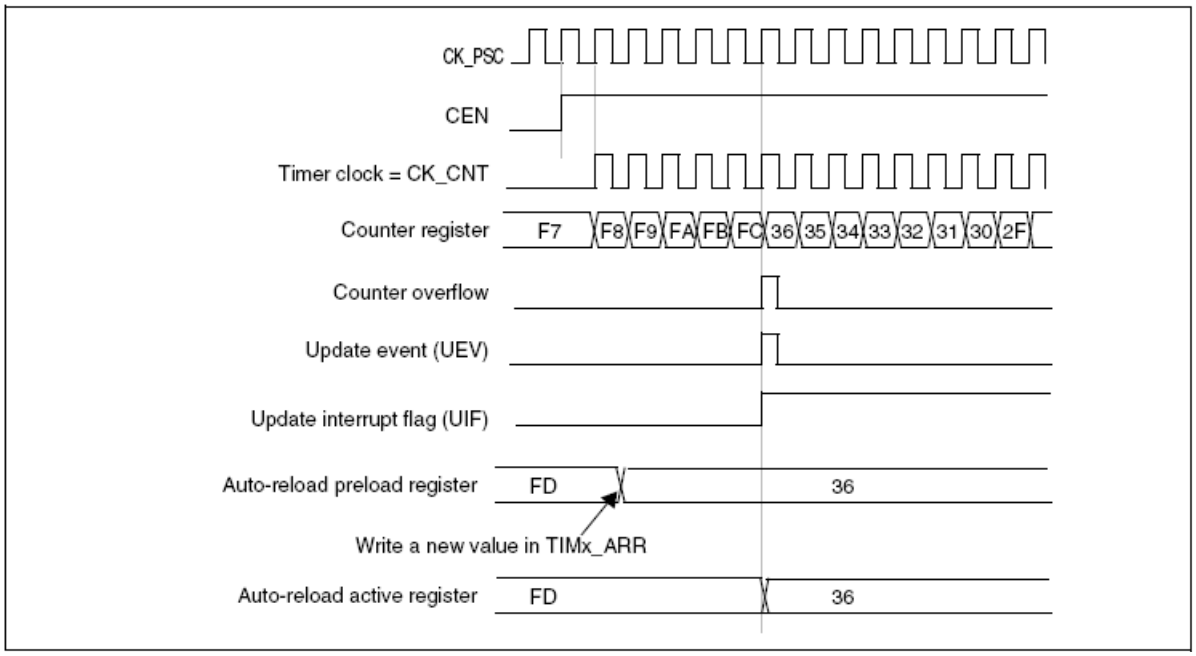


图65 计数器时序图，ARPE=1时的更新事件(计数器溢出)



12.3.3 重复计数器

12.3.1节“时基单元”解释了计数器上溢/下溢时更新事件(UEV)是如何产生的，然而事实上它只能在重复计数达到0的时候产生。这个特性对产生PWM信号非常有用。

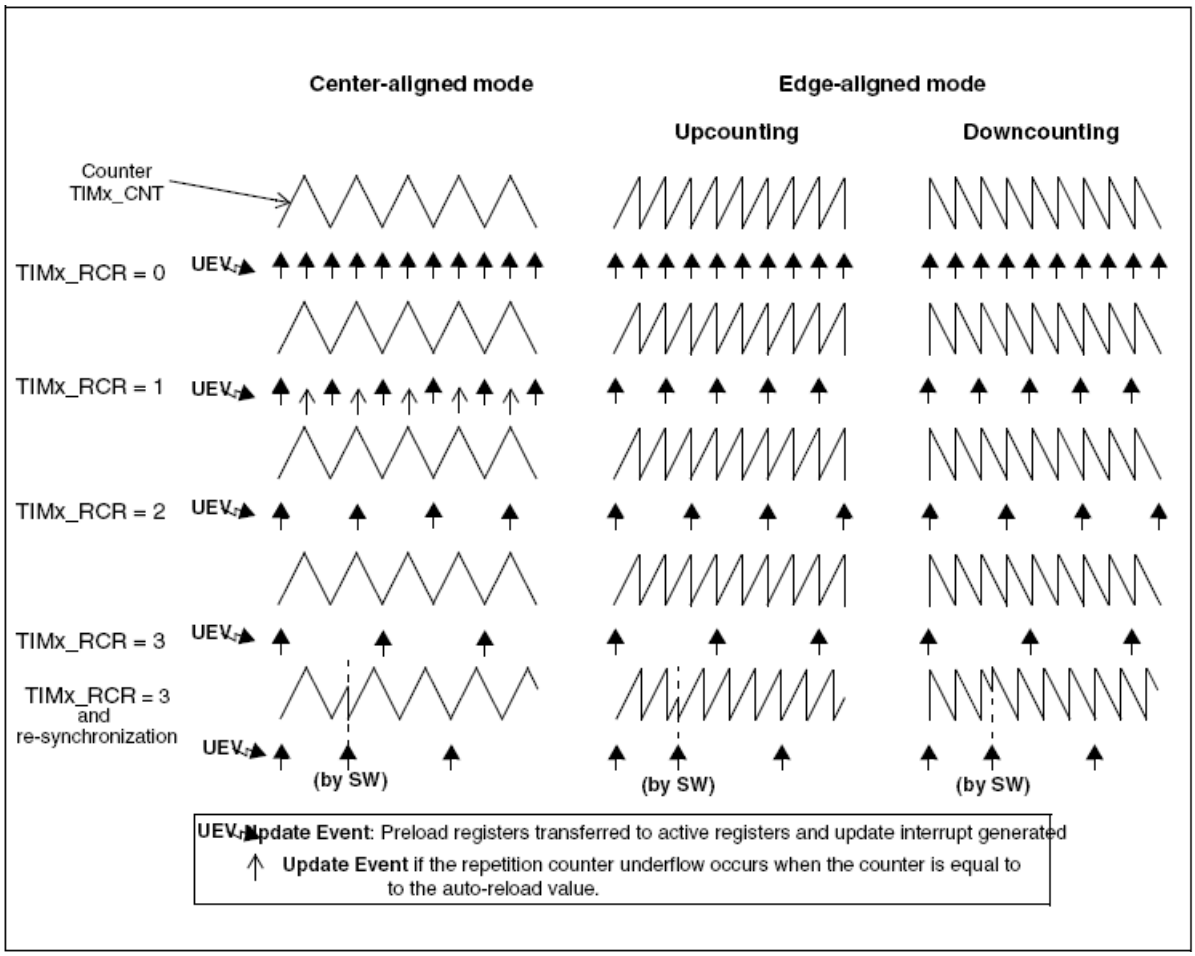
这意味着在每N次计数上溢或下溢时，数据从预装载寄存器传输到影子寄存器(TIMx_ARR自动重载入寄存器，TIMx_PSC预装载寄存器，还有在比较模式下的捕获/比较寄存器TIMx_CCRx)，N是TIMx_RCR重复计数寄存器中的值。

重复计数器在下述任一条件成立时递减：

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

重复计数器是自动加载的，重复速率是由TIMx_RCR寄存器的值定义(参看图66)。当更新事件由软件产生(通过设置TIMx_EGR 中的UG位)或者通过硬件的从模式控制器产生，则无论重复计数器的值是多少，立即发生更新事件，并且TIMx_RCR寄存器中的内容被重载入到重复计数器。

图66 不同模式下更新速率的例子，及TIMx_RCR的寄存器设置



12.3.4 时钟选择

计数器时钟可由下列时钟源提供：

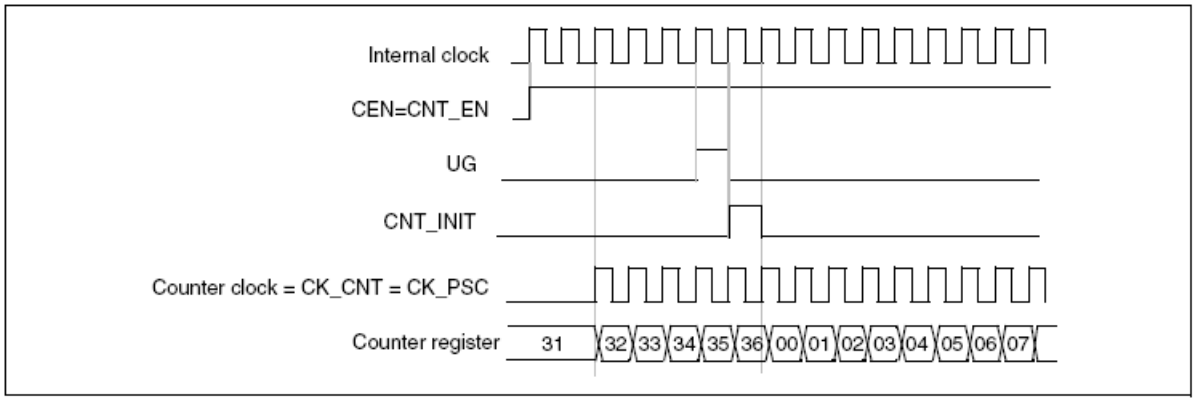
- 内部时钟(CK_INT)
- 外部时钟模式1：外部输入管脚
- 外部时钟模式2：外部触发输入ETR
- 内部触发输入(ITRx)：使用一个定时器作为另一个定时器的预分频器。如可以配置一个定时器Timer1而作为另一个定时器Timer2的预分频器。详见下一章。

内部时钟源(CK_INT)

如果禁止了从模式控制器(SMS=000)，则CEN、DIR(TIMx_CR1寄存器)和UG位(TIMx_EGR寄存器)是事实上的控制位，并且只能被软件修改(UG位仍被自动清除)。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。

下图显示控制电路和向上计数器在一般模式下，不带预分频器时的操作。

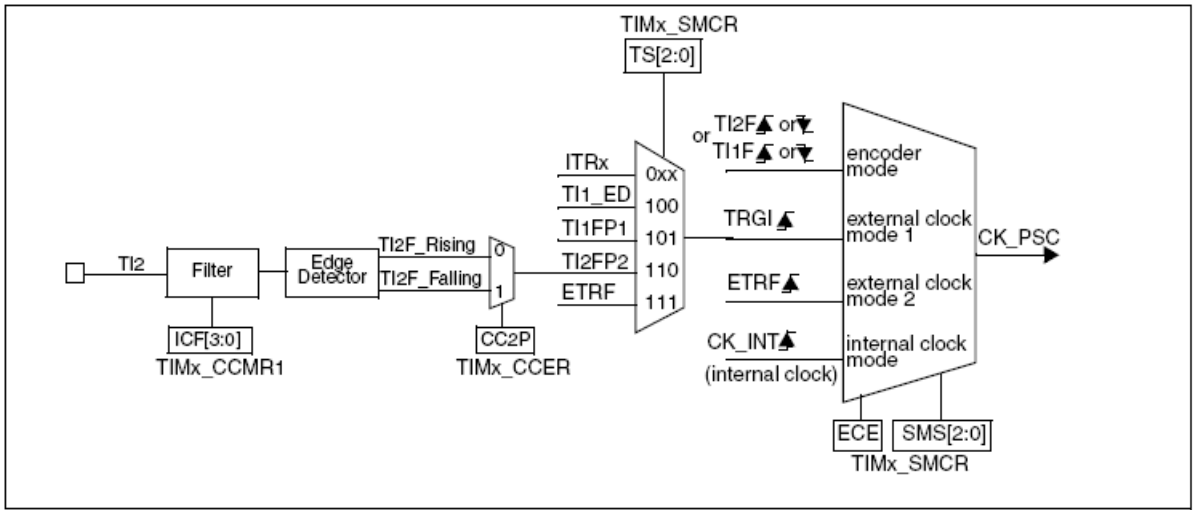
图67 一般模式下的控制电路，内部时钟分频因子为1



外部时钟源模式1

当TIMx_SMCR寄存器的SMS=111时，此模式被选中。计数器可以在选定输入端的每个上升沿或下降沿计数。

图68 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：

配置TIMx_CCMR1寄存器CC2S=01，配置通道2检测TI2输入的上升沿

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)

配置TIMx_CCER寄存器的CC2P=0，选定上升沿极性

配置TIMx_SMCR寄存器的SMS=111，选择定时器外部时钟模式1

配置TIMx_SMCR寄存器中的TS=110，选定TI2作为触发输入源

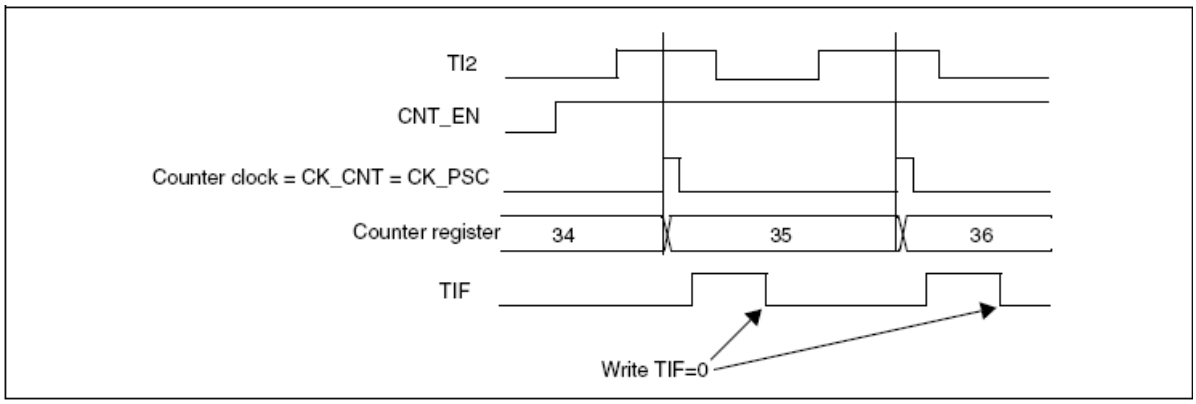
设置TIMx_CR1寄存器的CEN=1，启动计数器

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69 外部时钟模式1下的控制电路



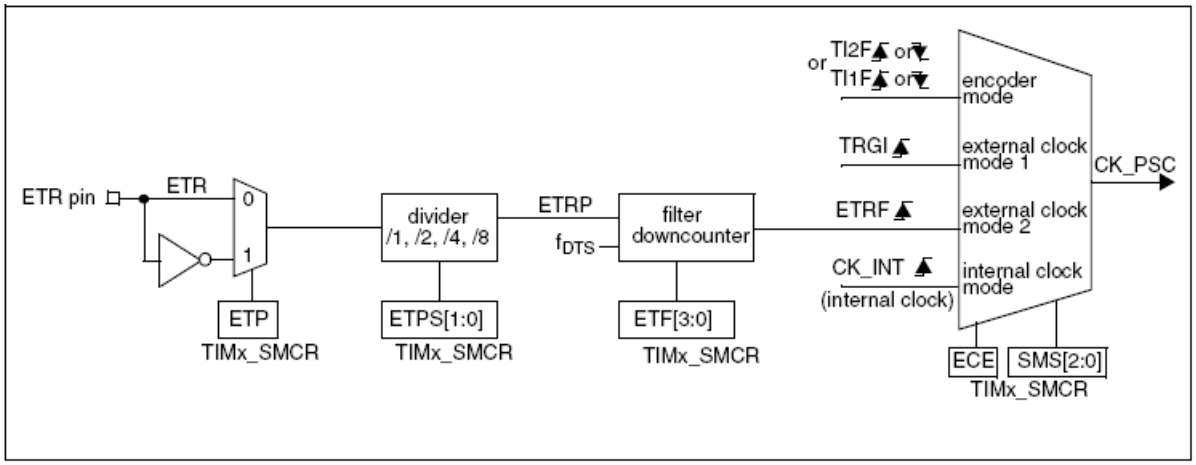
外部时钟源模式2

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。

下图是外部触发输入的总体框图

图70 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：

4. 本例中不需要滤波器，置TIMx_SMCR寄存器中的ETF[3:0]=0000

设置预分频器，置TIMx_SMCR寄存器中的ETPS[1:0]=01

选择ETR的上升沿检测，置TIMx_SMCR寄存器中的ETP=0

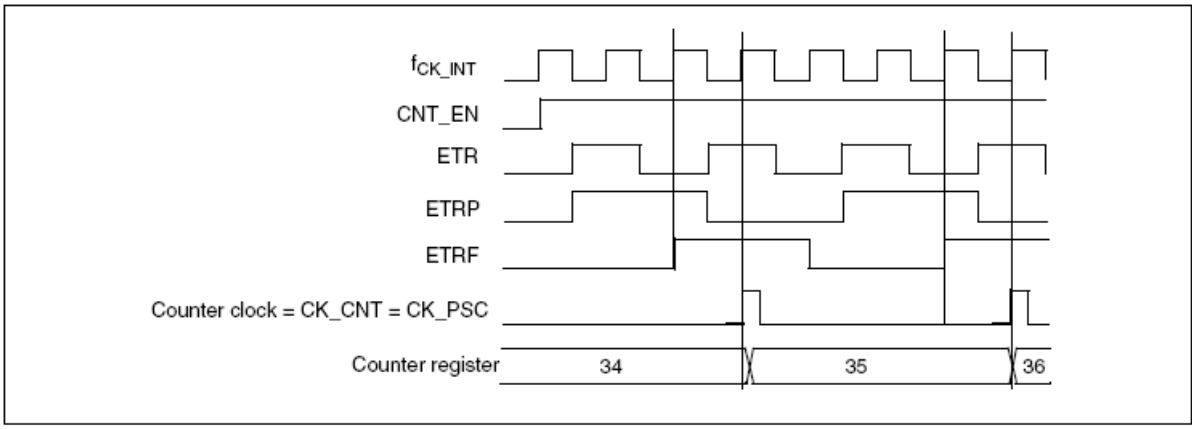
开启外部时钟模式2，写TIMx_SMCR寄存器中的ECE=1

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71 外部时钟模式2下的控制电路



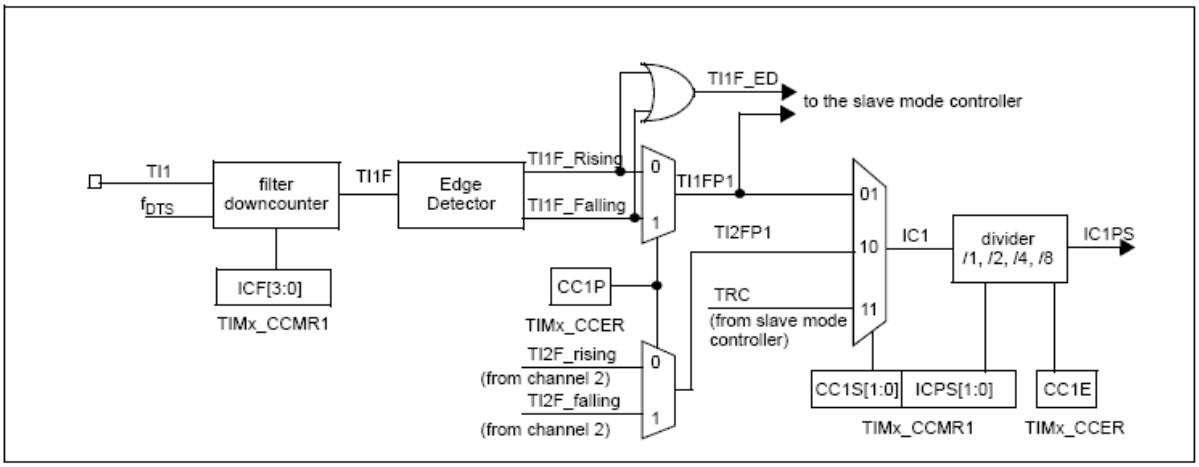
12.3.5 捕获/比较通道

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

图72至图75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。该信号通过预分频进入捕获寄存器(ICxPS)。

图72 捕获/比较通道(如：通道1输入部分)



输出部分产生一个中间波形OCxRef(高有效)作为基准，链的末端决定最终输出信号的极性。

图73 捕获/比较通道1的主电路

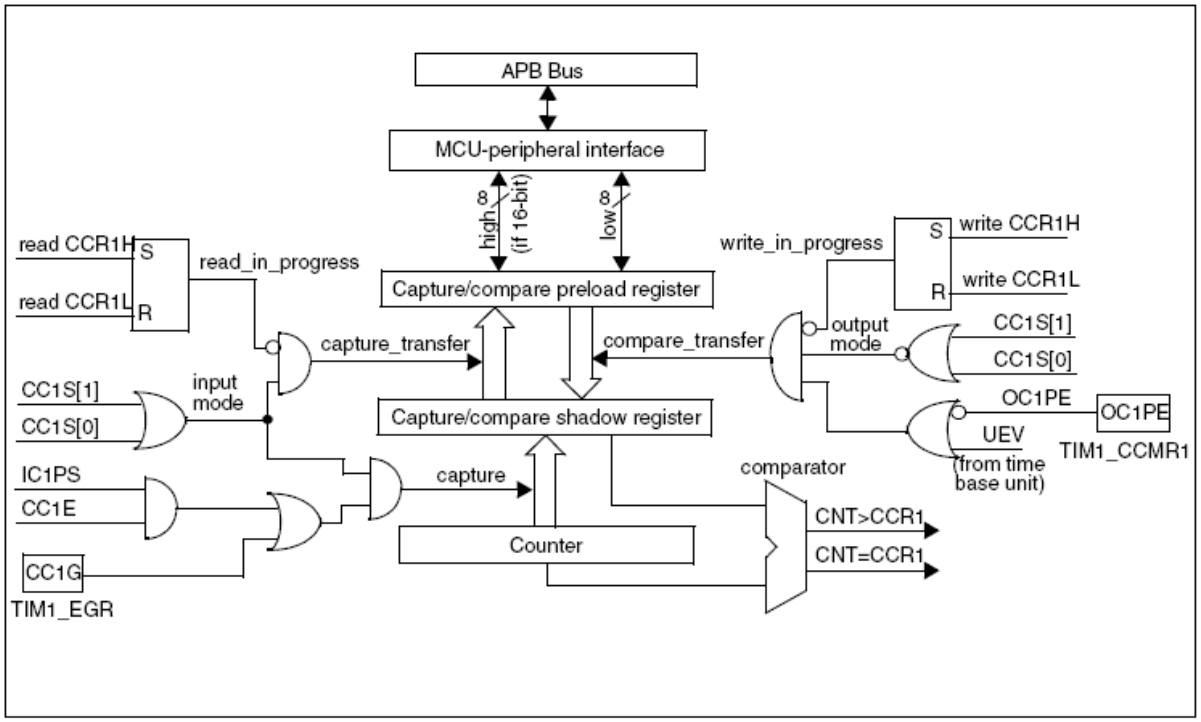


图74 捕获/比较通道的输出部分(通道1至3)

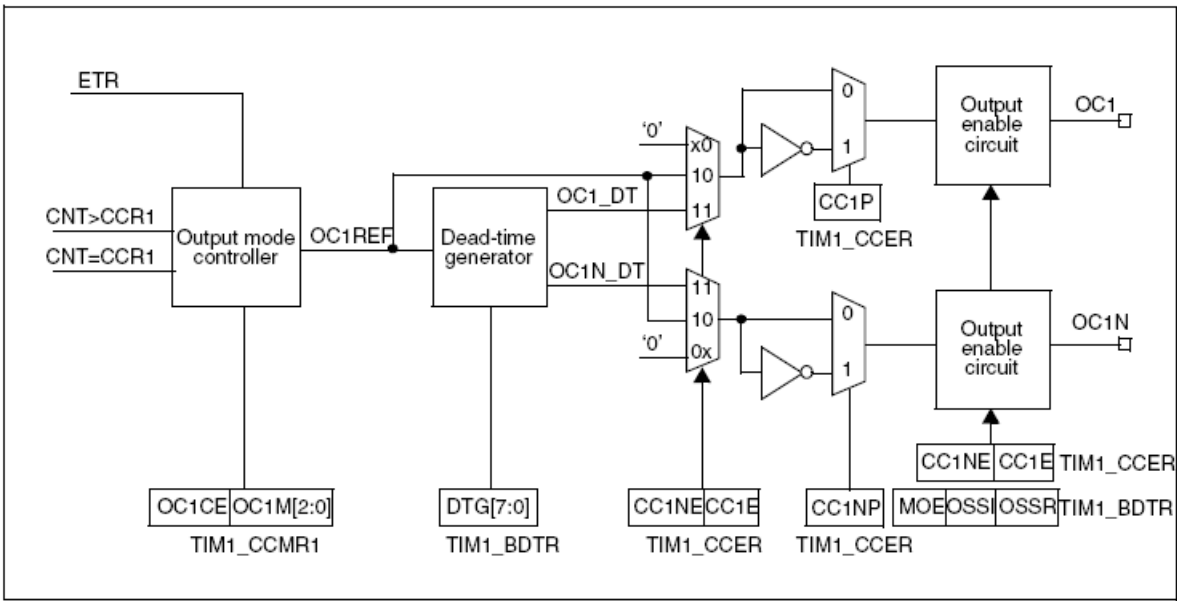
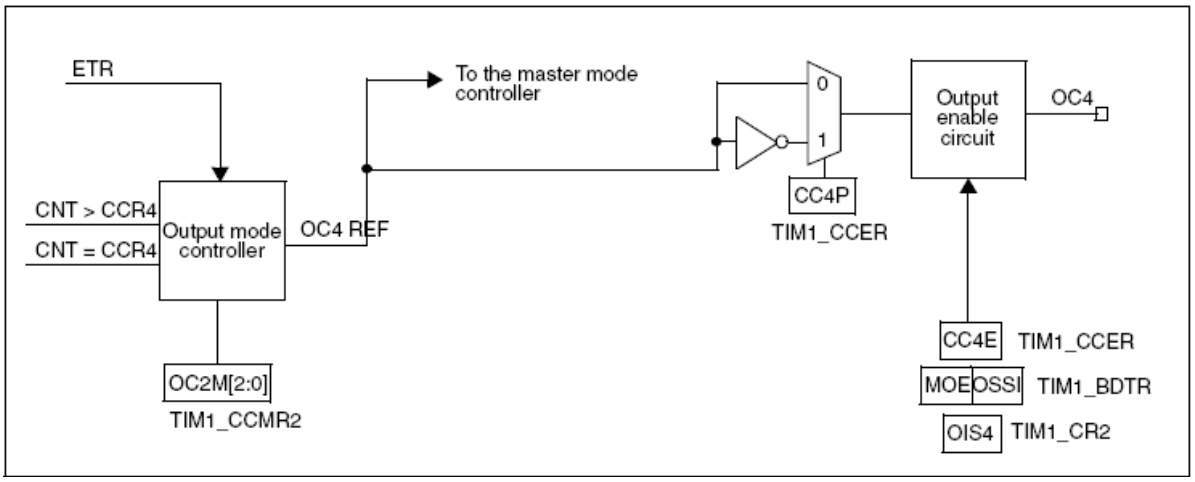


图75 捕获/比较通道的输出部分(通道4)



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

在捕获模式下，捕获发生在影子寄存器上，然后再复制到预装载寄存器中。

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

12.3.6 输入捕获模式

在输入捕获模式下，当检测到ICx信号上相应的边沿后，计数器的当前值被锁存到捕获/比较寄存器(TIMx_CCRx)中。当发生捕获事件时，相应的CCxIF标志(TIMx_SR寄存器)被置1，如果开放了中断或者DMA操作，则将产生中断或者DMA请求。如果发生捕获事件时CCxIF标志已经为高，那么重复捕获标志CCxOF(TIMx_SR寄存器)被置1。写CCxIF=0可清除CCxIF，或读取存储在TIMx_CCRx寄存器中的捕获数据也可清除CCxIF。写CCxOF=0可清除CCxOF。

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

- 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TIMx_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(即输入为TIx时，输入滤波器控制位是TIMx_CCMRx寄存器中的ICxF位)。假设输入信号在最多5个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f_{DTS}频率)连续采样8次，以确认在TI1上一次真实的边沿变换，即在TIMx_CCMR1寄存器中写入IC1F=0011。
- 选择TI1通道的有效转换边沿，在TIMx_CCER寄存器中写入CC1P=0(上升沿)。
- 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。

当发生一个输入捕获时：

- 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。*

12.3.7 PWM输入模式

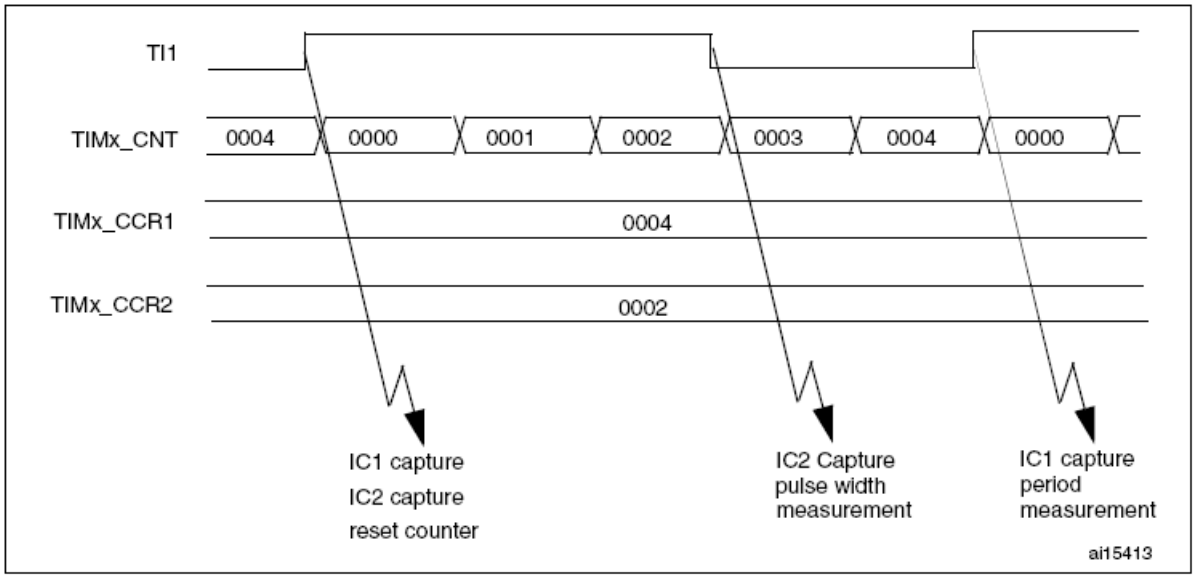
该模式是输入捕获模式的一个特例，除下列区别外，操作与输入捕获模式相同：

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

例如，你需要测量输入到TI1上的PWM信号的长度(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：置TIMx_CCMR1寄存器的CC1S=01(选中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(上升沿有效)。
- 选择TIMx_CCR2的有效输入：置TIMx_CCMR1寄存器的CC2S=10(选中TI1)。
- 选择TI1FP2的有效极性(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76 PWM输入模式时序



因为只有TI1FP1和TI2FP2连到了从模式控制器，所以PWM输入模式只能使用TIMx_CH1/TIMx_CH2信号。

12.3.8 强置输出模式

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

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

例如：CCxP=0(OCx高电平有效)，则OCx被强置为高电平。

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

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

12.3.9 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(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=000)、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)。
- 若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位选择TIMx_CCRx寄存器是否需要使用预装载寄存器。

在输出比较模式下，更新事件UEV对OCxREF和OCx输出没有影响。

同步的精度可以达到计数器的一个计数周期。输出比较模式(在单脉冲模式下)也能用来输出一个单脉冲。

输出比较模式的配置步骤：

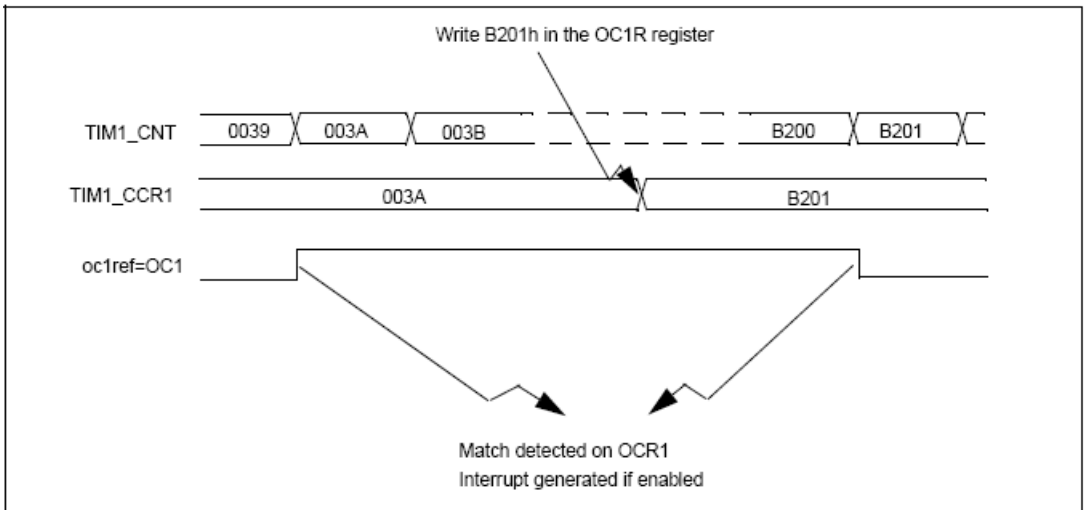
1. 选择计数器时钟(内部，外部，预分频器)。
2. 将相应的数据写入TIMx_ARR和TIMx_CCRx寄存器中。
3. 如果要产生一个中断请求，设置CCxIE位。
4. 选择输出模式，例如：

— 要求计数器与 CCRx 匹配时翻转 OCx 的输出管脚，设置 OCxM=011
— 置 OCxPE = 0 禁用预装载寄存器
— 置 CCxP = 0 选择极性为高电平有效
— 置 CCxE = 1 使能输出

5. 设置TIMx_CR1寄存器的CEN位启动计数器

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

图77 输出比较模式，翻转OC1



12.3.10 PWM模式

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

在TIMx_CCMRx寄存器中的OCxM位写入'110'(PWM模式1)或'111'(PWM模式2)，能够独立地设置每个OCx输出通道产生一路PWM。必须通过设置TIMx_CCMRx寄存器的OCxPE位使能相应的预装载寄存器，最后还要设置TIMx_CR1寄存器的ARPE位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或低电平有效。OCx的输出使能通过(TIMx_CCER和TIMx_BDTR寄存器中)CCxE、CCxNE、MOE、OSSI和OSSR位的组合控制。详见TIMx_CCER寄存器的描述。

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

根据TIMx_CR1寄存器中CMS位的状态，定时器能够产生边沿对齐的PWM信号或中央对齐的PWM信号。

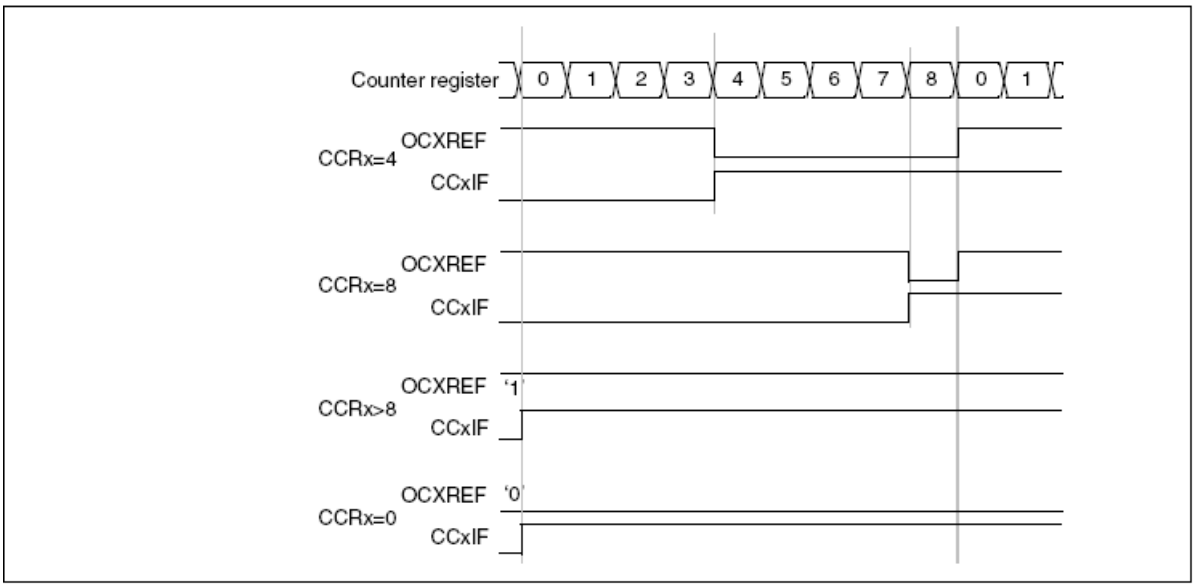
PWM 边沿对齐模式

- 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12.3.2节。

  下面是一个PWM模式1的例子。当TIMx_CNT<TIMx_CCRx时，PWM参考信号OCxREF为高，否则为低。如果TIMx_CCRx中的比较值大于自动重装载值(TIMx_ARR)，则OCxREF保持为'1'。如果比较值为0，则OCxREF保持为'0'。 图78为TIMx_ARR=8时边沿对齐的PWM波形实例。

图78 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12.3.2节。

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

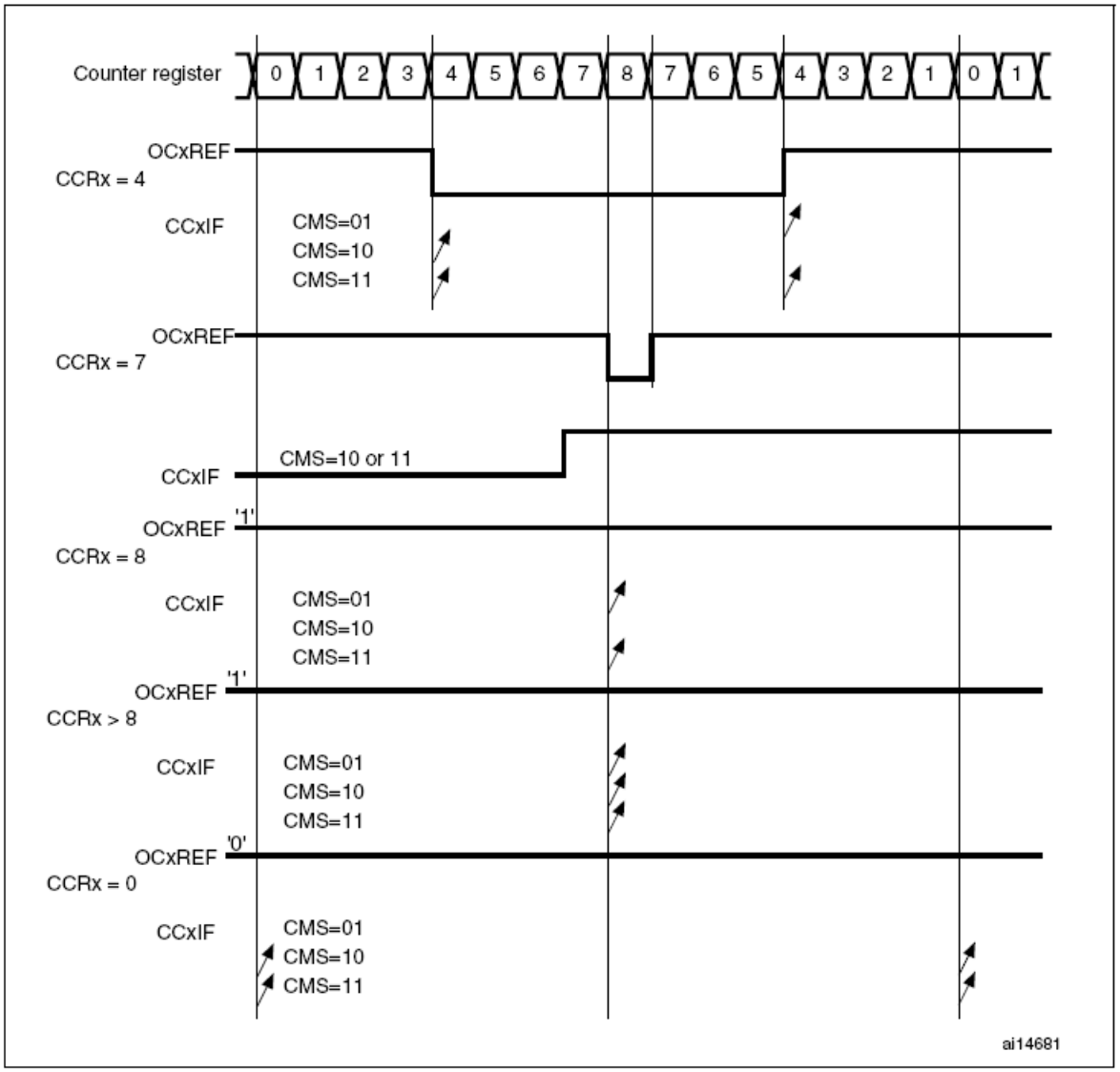
PWM 中央对齐模式

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，比较标志可以在计数器向上计数时被置1、在计数器向下计数时被置1、或在计数器向上和向下计数时被置1。TIMx_CR1寄存器中的计数方向位(DIR)由硬件更新，不要用软件修改它。参看12.3.2节的中央对齐模式。

图79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寄存器的CMS=01，在中央对齐模式1下，当计数器向下计数时设置比较标志。

图79 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，使用当前的上/下计数配置；这就意味着计数器向上还是向下计数取决于TIMx_CR1寄存器中DIR位的当前值。此外，软件不能同时修改DIR和CMS位。
- 不推荐当运行在中央对齐模式时改写计数器，因为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)，并且不要在计数进行过程中修改计数器的值。

12.3.11 互补输出和死区插入

高级控制定时器(TIM1和TIM8)能够输出两路互补信号，并且能够管理输出的瞬时关断和接通。

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

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

互补信号OCx和OCxN通过下列控制位的组合进行控制：TIMx_CCER寄存器的CCxE和CCxNE位，TIMx_BDTR和TIMx_CR2寄存器中的MOE、OISx、OISxN、OSSI和OSSR位，详见表56带刹车功能的互补输出通道OCx和OCxN的控制位。特别的是，在转换到IDLE状态时(MOE下降到0)死区被激活。

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

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

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

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

图80 带死区插入的互补输出

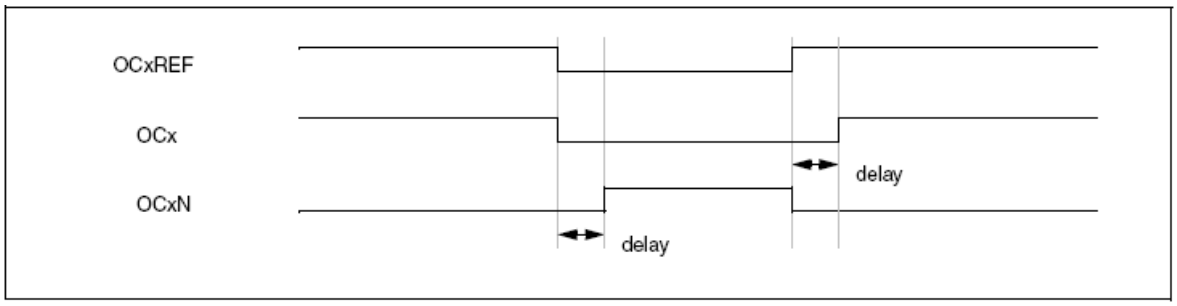


图81 死区波形延迟大于负脉冲

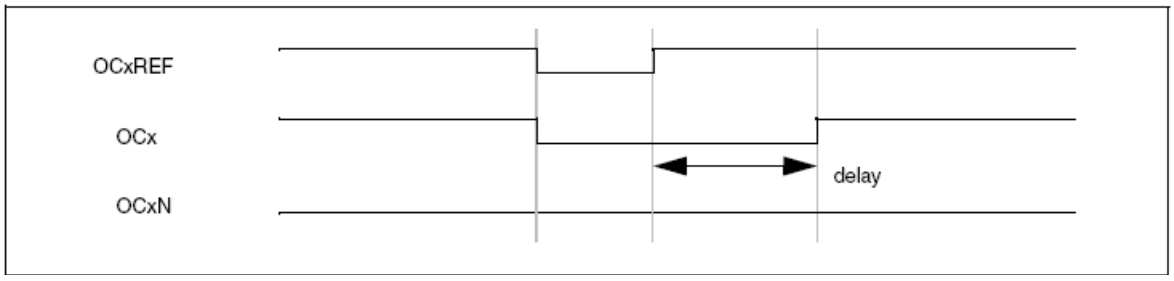
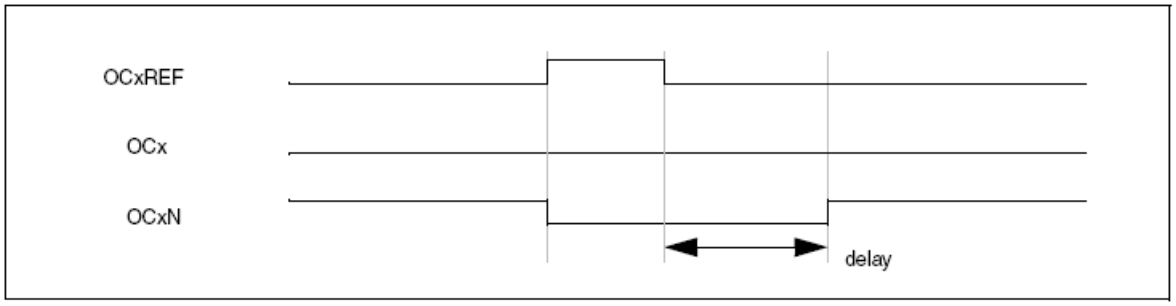


图82 死区波形延迟大于正脉冲



每一个通道的死区延时都是相同的，是由TIMx_BDTR寄存器中的DTG位编程配置。详见12.4.18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)，通过配置TIMx_CCER寄存器的CCxE和CCxNE位，OCxREF可以被重定向到OCx或者OCxN的输出。

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

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

12.3.12 使用刹车功能

当使用刹车功能时，依据相应的控制位(TIMx_BDTR寄存器中的MOE、OSSI和OSSR位，TIMx_CR2寄存器中的OISx和OISxN位)，输出使能信号和无效电平都会被修改。但无论何时，OCx和OCxN输出不能在同一时间同时处于有效电平上。详见表56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，MOE位为低。设置TIMx_BDTR寄存器中的BKE位可以使能刹车功能。刹车输入信号的极性可以通过配置同一个寄存器中的BKP位选择。BKE和BKP可以被同时修改。

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TIMx_BDTR寄存器中)之间设置了一个再同步电路。这个再同步电路会在异步信号和同步信号之间产生延迟。特别的，如果当它为低时写MOE=1，则读出它之前必须先插入一个延时(空指令)才能读到正确的值。这是因为写入的是异步信号而读的是同步信号。

当发生刹车时(在刹车输入端出现选定的电平)，有下述动作：

- MOE位被异步地清除，将输出置于无效状态、空闲状态或者复位状态(由OSSI位选择)。这个特性在MCU的振荡器关闭时依然有效。
- 一旦MOE=0，每一个输出通道输出由TIMx_CR2寄存器中的OISx位设定的电平。如果OSSI=0，则定时器释放使能输出，否则使能输出始终为高。
- 当使用互补输出时：
  - 输出首先被置于复位状态即无效的状态(取决于极性)。这是异步操作，即使定时器没有时钟时，此功能也有效。
  - 如果定时器的时钟依然存在，死区生成器将会重新生效，在死区之后根据 OISx 和 OISxN 位指示的电平驱动输出端口。即使在这种情况下，OCx 和 OCxN 也不能被同时驱动到有效的电平。注，因为重新同步 MOE，死区时间比通常情况下长一些(大约 2 个 ck_tim 的时钟周期)。
  - 如果 OSSI=0，定时器释放使能输出，否则保持使能输出；或一旦 CCxE 与 CCxNE 之一变高时，使能输出变为高。
- 如果设置了TIMx_DIER寄存器中的BIE位，当刹车状态标志(TIMx_SR寄存器中的BIF位)为'1'时，则产生一个中断。如果设置了TIMx_DIER寄存器中的BDE位，则产生一个DMA请求。
- 如果设置了TIMx_BDTR寄存器中的AOE位，在下一个更新事件UEV时MOE位被自动置位；例如，这可以用来进行整形。否则，MOE始终保持低直到被再次置'1'；此时，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。

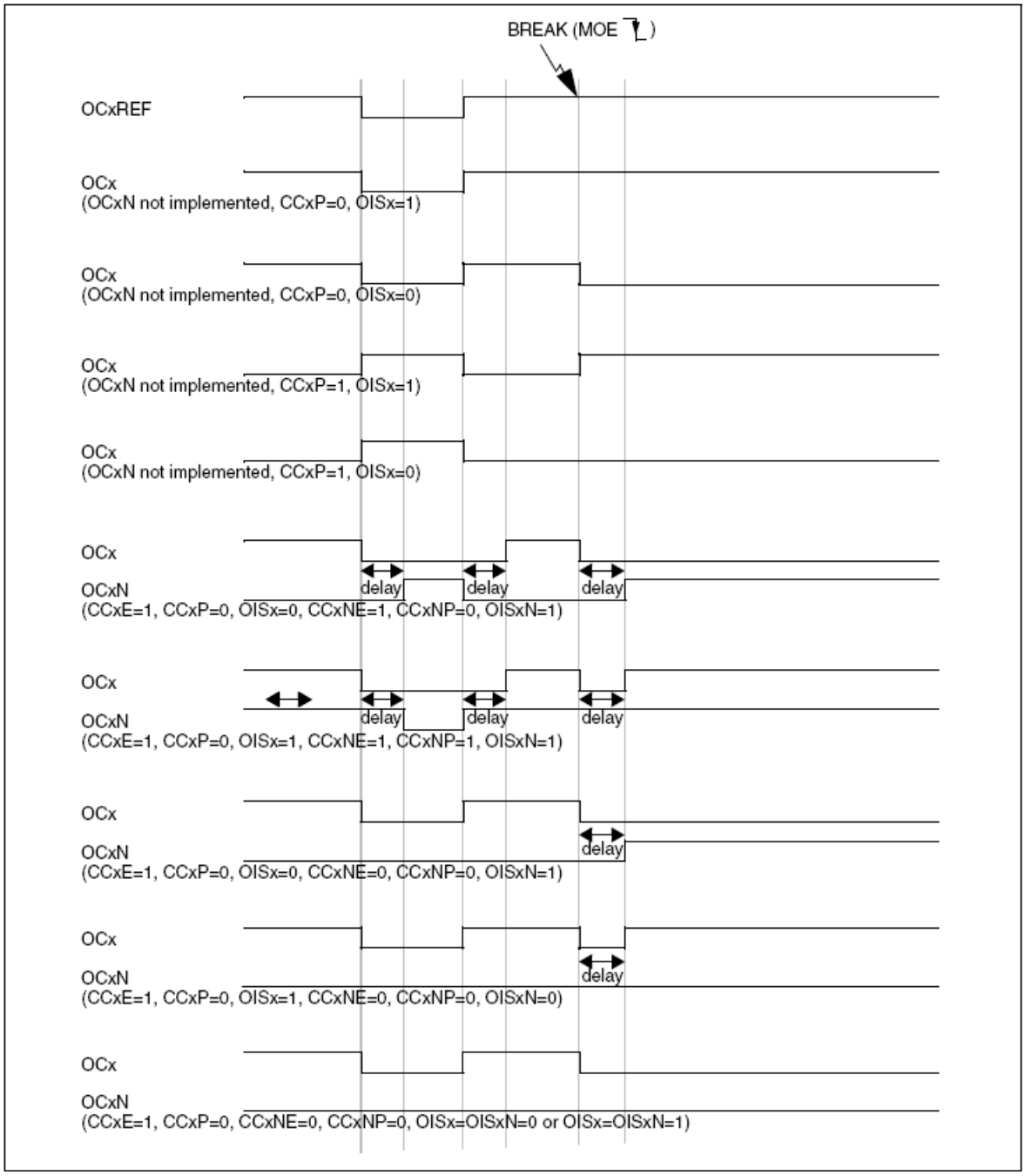
注： *刹车输入为电平有效。所以，当刹车输入有效时，不能同时(自动地或者通过软件)设置MOE。同时，状态标志BIF不能被清除。*

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

除了刹车输入和输出管理，刹车电路中还实现了写保护以保证应用程序的安全。它允许用户冻结几个配置参数(死区长度，OCx/OCxN极性和被禁止的状态，OCxM配置，刹车使能和极性)。用户可以通过TIMx_BDTR寄存器中的LOCK位，从三级保护中选择一种，参看12.4.18节。在MCU复位后LOCK位只能被修改一次。

下图显示响应刹车的输出实例。

图83 响应刹车的输出



12.3.13 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(设置TIMx_CCMRx寄存器中对应的OCxCE位为'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。

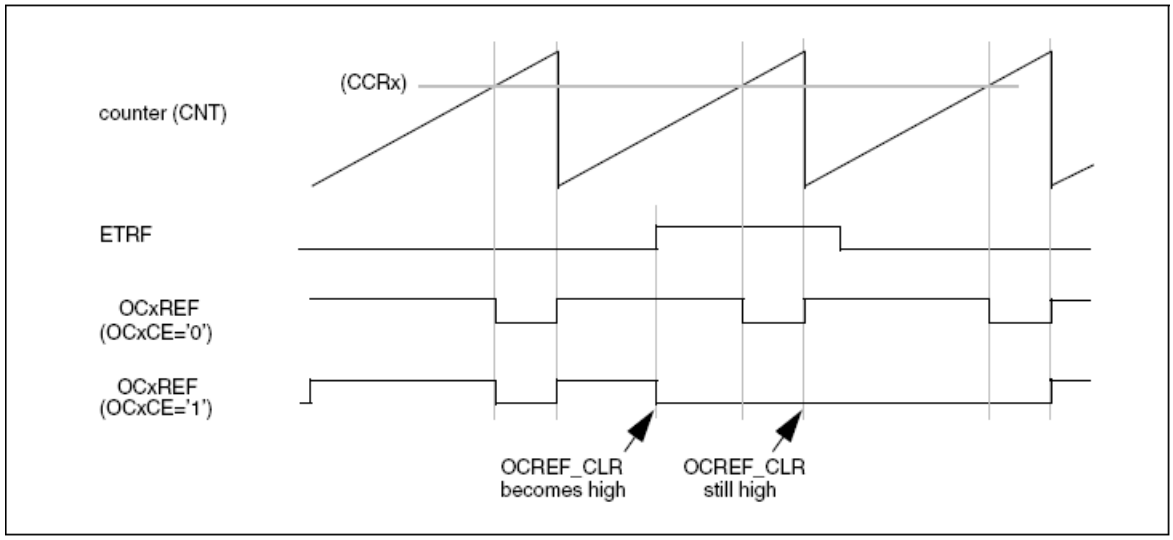
该功能只能用于输出比较和PWM模式，而不能用于强置模式。

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

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

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

图84 清除TIMx的OCxREF



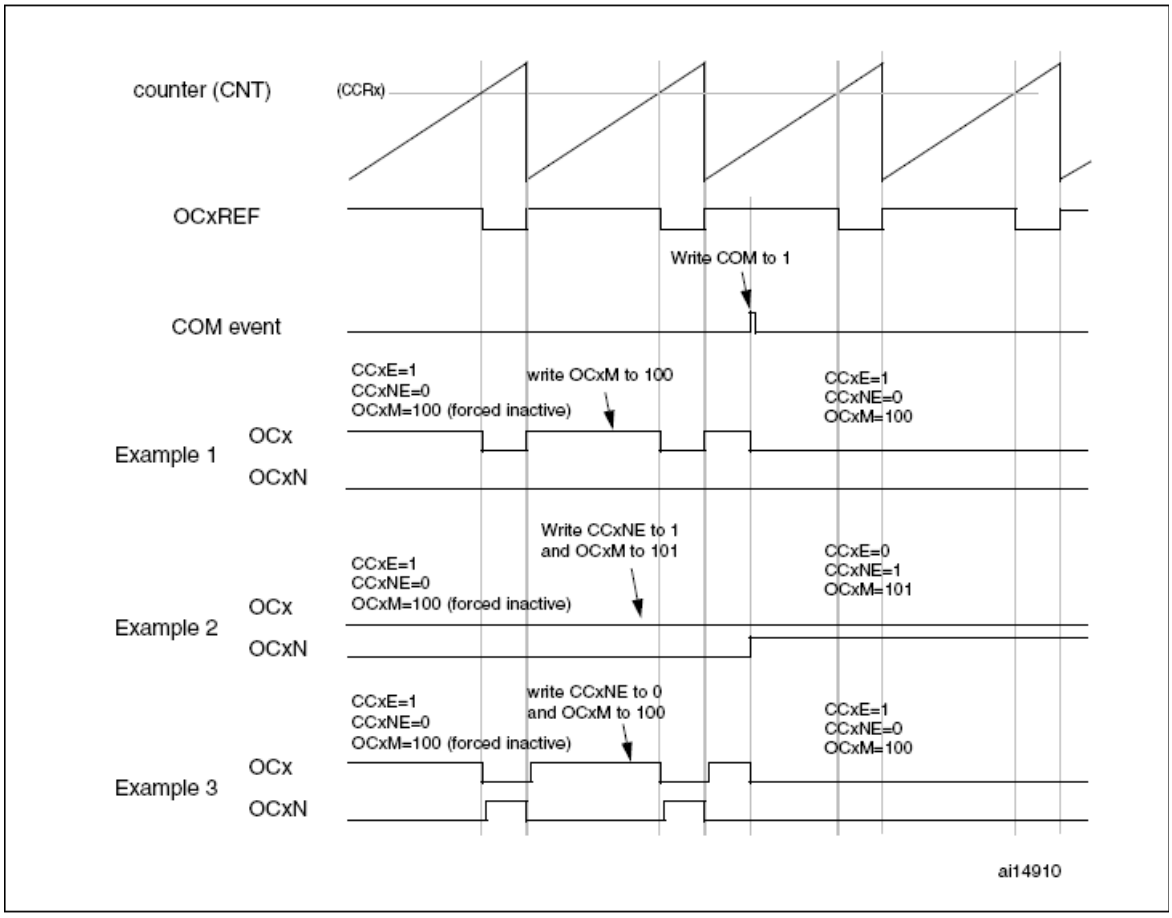
12.3.14 产生六步PWM输出

当在一个通道上需要互补输出时，预装载位有OCxM、CCxE和CCxNE。在发生COM换相事件时，这些预装载位被传送到影子寄存器位。这样你就可以预先设置好下一步骤配置，并在同一个时刻同时修更改所有通道的配置。COM可以通过设置TIMx_EGR寄存器的COM位由软件产生，或在TRGI上升沿由硬件产生。

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

下图显示当发生COM事件时，三种不同配置下OCx和OCxN输出。

图85 产生六步PWM，使用COM的例子(OSSR=1)



12.3.15 单脉冲模式

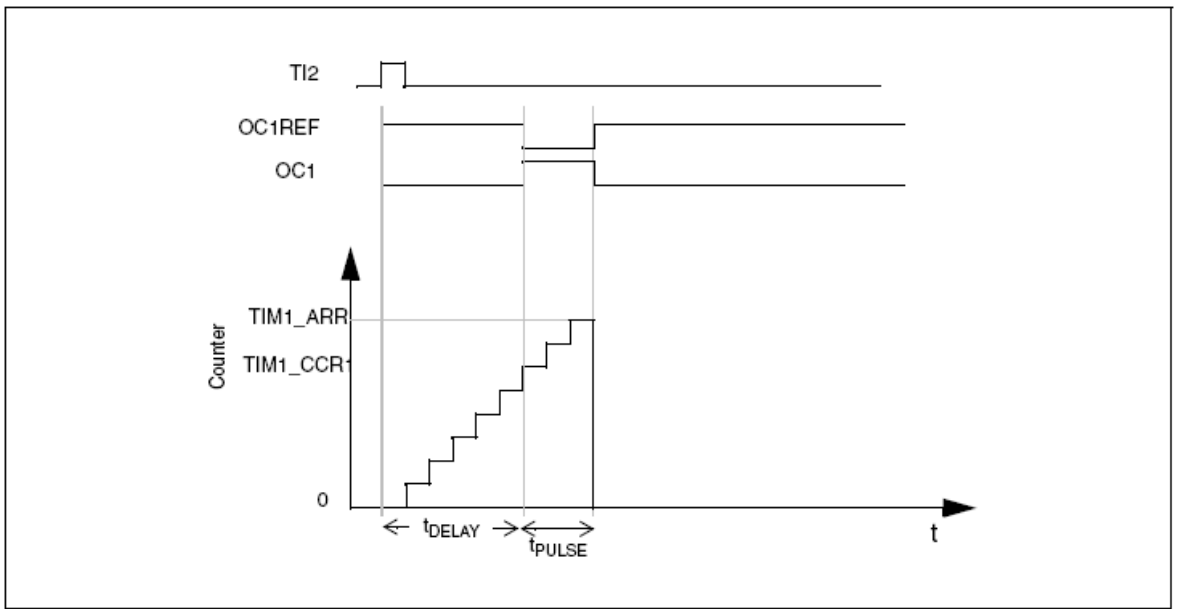
单脉冲模式(OPM)是前述众多模式的一个特例。这种模式允许计数器响应一个激励，并在一个程序可控的延时之后产生一个脉宽可程序控制的脉冲。

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

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

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

图86 单脉冲模式的例子



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

假定TI2FP2作为触发1:

- 置TIMx_CCMR1寄存器中的CC2S=01，把TI2FP2映像到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，TI2FP2作为从模式控制器的触发(TRGI)。
- 置TIMx_SMCR寄存器中的SMS=110(触发模式)，TI2FP2被用来启动计数器。

OPM的波形由写入比较寄存器的数值决定(要考虑时钟频率和计数器预分频器)

- t_{DELAY}由TIMx_CCR1寄存器中的值定义。
- t_{PULSE}由自动装载值和比较值之间的差值定义(TIMx_ARR - TIMx_CCR1)。
- 假定当发生比较匹配时要产生从0到1的波形，当计数器达到预装载值时要产生一个从1到0的波形；首先要置TIMx_CCMR1寄存器的OC1M=111，进入PWM模式2；根据需要有选择地使能预装载寄存器：置TIMx_CCMR1中的OC1PE=1和TIMx_CR1寄存器中的ARPE；然后在TIMx_CCR1寄存器中填写比较值，在TIMx_ARR寄存器中填写自动装载值，设置UG位来产生一个更新事件，然后等待在TI2上的一个外部触发事件。本例中，CC1P=0。

在这个例子中，TIMx_CR1寄存器中的DIR和CMS位应该置低。

因为只需要一个脉冲，所以必须设置TIMx_CR1寄存器中的OPM=1，在下一个更新事件(当计数器从自动装载值翻转到0)时停止计数。

特殊情况：OCx快速使能：

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

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

12.3.16 编码器接口模式

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

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

两个输入TI1和TI2被用来作为增量编码器的接口。参看表54，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，同时硬件对TIMx_CR1寄存器的DIR位进行相应的设置。不管计数器是依靠TI1计数、依靠TI2计数或者同时依靠TI1和TI2计数，在任一输入端(TI1或者TI2)的跳变都会重新计算DIR位。

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

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

表54 计数方向与编码器信号的关系

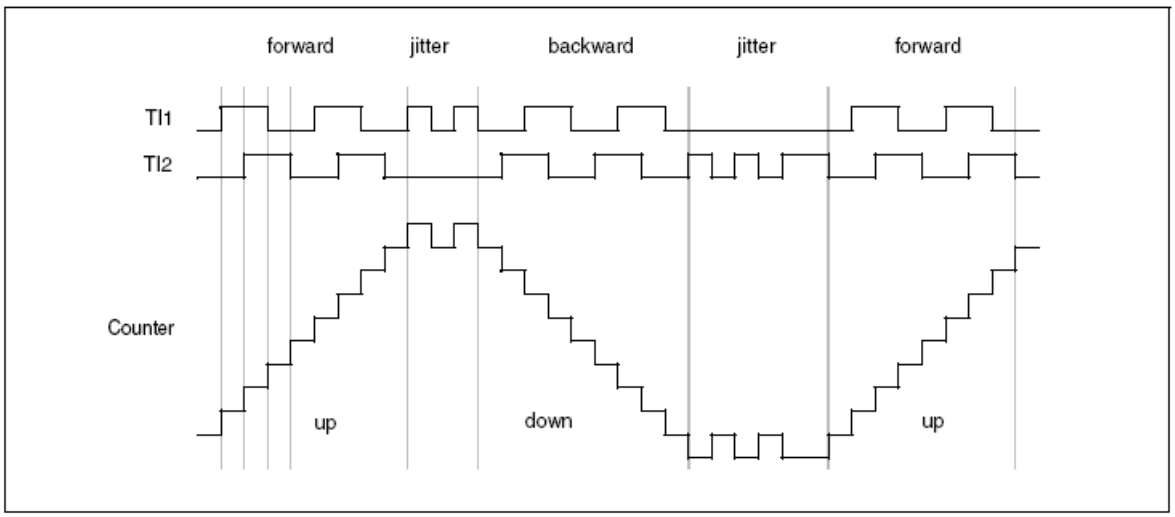
有效边沿	相对信号的电平 (TI1FP1对应TI2, TI2FP2对应TI1)	TI1FP1信号		TI2FP2信号	
		上升	下降	上升	下降
仅在TI1计数	高	向下计数	向上计数	不计数	不计数
	低	向上计数	向下计数	不计数	不计数
仅在TI2计数	高	不计数	不计数	向上计数	向下计数
	低	不计数	不计数	向下计数	向上计数
在TI1和TI2上计数	高	向下计数	向上计数	向上计数	向下计数
	低	向上计数	向下计数	向下计数	向上计数

一个外部的增量编码器可以直接与MCU连接而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，可以把它连接到一个外部中断输入并触发一个计数器复位。

下图是一个计数器操作的实例，显示了计数信号的产生和方向控制。它还显示了当选择了双边沿时，输入抖动是如何被抑制的；抖动可能会在传感器的位置靠近一个转换点时产生。在这个例子中，我们假定配置如下：

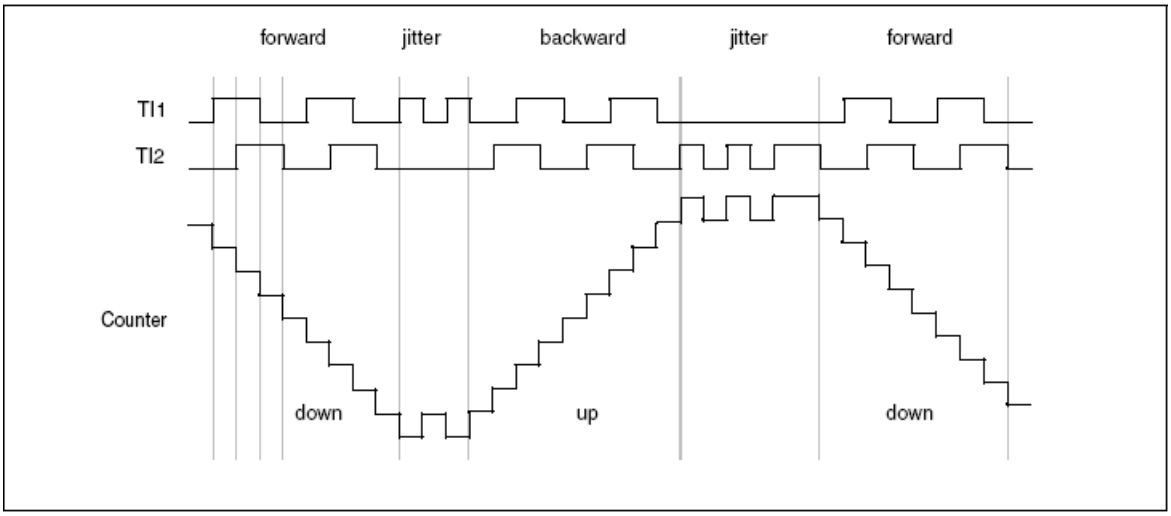
- CC1S=’01’ (TIMx_CCMR1寄存器，IC1FP1映射到TI1)
- CC2S=’01’ (TIMx_CCMR2寄存器，IC2FP2映射到TI2)
- CC1P=’0’ (TIMx_CCER寄存器，IC1FP1不反相，IC1FP1=TI1)
- CC2P=’0’ (TIMx_CCER寄存器，IC2FP2不反相，IC2FP2=TI2)
- SMS=’011’ (TIMx_SMCR寄存器，所有的输入均在上升沿和下降沿有效).
- CEN=’1’ (TIMx_CR1寄存器，计数器使能)

图87 编码器模式下的计数器操作实例



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

图88 IC1FP1反相的编码器接口模式实例



当定时器配置成编码器接口模式时，提供传感器当前位置的信息。使用第二个配置在捕获模式的定时器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。

12.3.17 定时器输入异或功能

TIMx_CR2寄存器中的TI1S位，允许通道1的输入滤波器连接到一个异或门的输出端，异或门的3个输入端为TIMx_CH1、TIMx_CH2和TIMx_CH3。

异或输出能够被用于所有定时器的输入功能，如触发或输入捕获。下节12.3.18给出了此特性用于连接霍尔传感器的例子。

12.3.18 与霍尔传感器的接口

使用高级控制定时器(TIM1或TIM8)产生PWM信号驱动马达时，可以用另一个通用TIMx(TIM2、TIM3、TIM4或TIM5)定时器作为“接口定时器”来连接霍尔传感器，见图89，3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。每当3个输入之一变化时，计数器从新从0开始计数。这样产生一个由霍尔输入端的任何变化而触发的时间基准。

“接口定时器”上的捕获/比较通道1配置为捕获模式，捕获信号为TRC(见图72)。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(通过触发一个COM事件)用于改变高级定时器TIM1或TIM8各个通道的属性，而高级控制定时器产生PWM信号驱动马达。因此“接口定时器”通道必须编程为在一个指定的延时(输出比较或PWM模式)之后产生一个正脉冲，这个脉冲通过TRGO输出被送到高级控制定时器TIM1或TIM8。

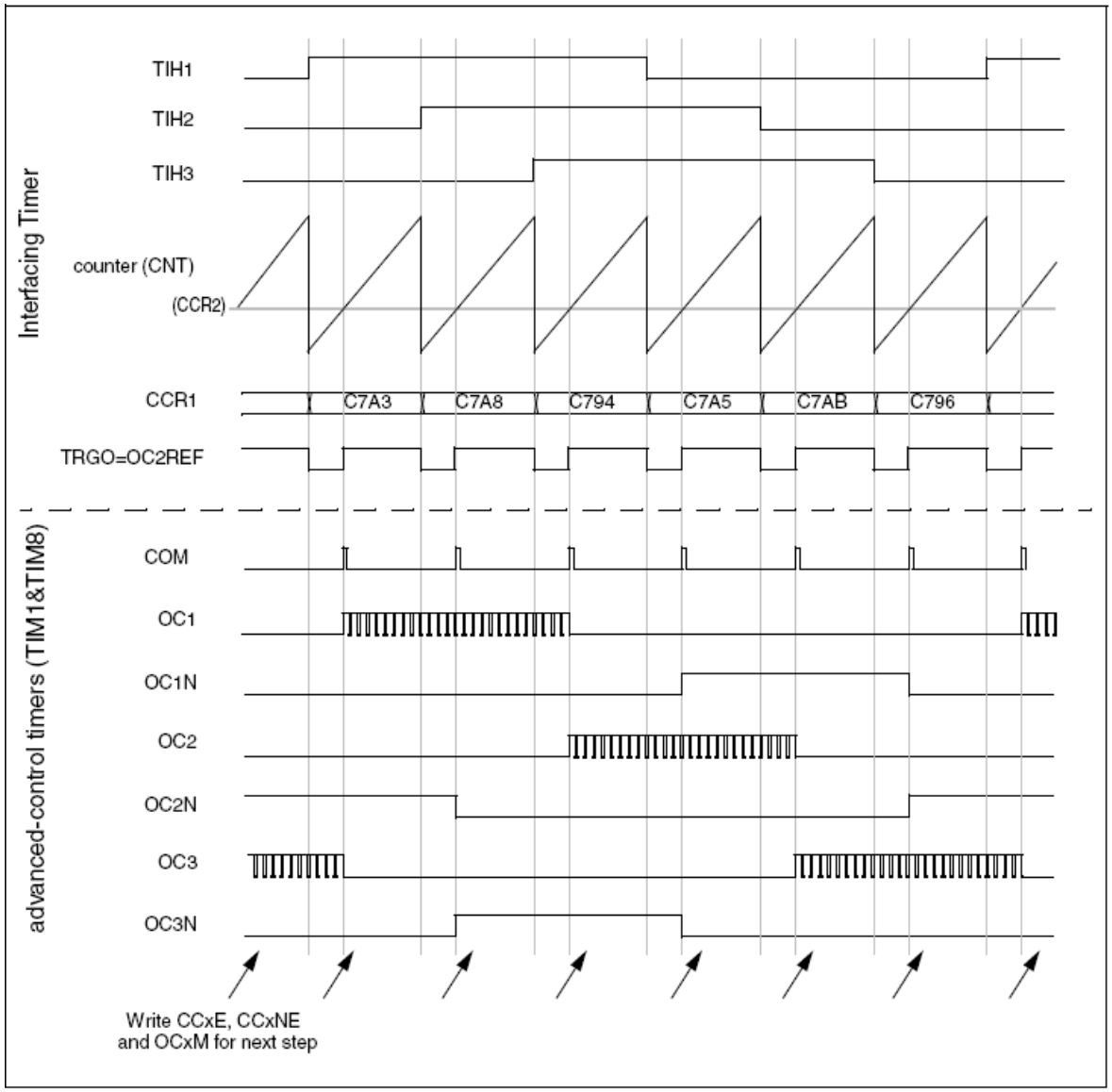
举例：霍尔输入连接到TIMx定时器，要求每次任一霍尔输入上发生变化之后的一个指定的时刻，改变高级控制定时器TIMx的PWM配置。

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

在高级控制寄存器TIM1中，正确的ITR输入必须是触发器输入，定时器被编程为产生PWM信号，捕获/比较控制信号为预装载的(TIMx_CR2寄存器中CCPC=1)，同时触发输入控制COM事件(TIMx_CR2寄存器中CCUS=1)。在一次COM事件后，写入下一步的PWM控制位(CCxE、OCxM)，这可以在处理OC2REF上升沿的中断子程序里实现。

下图显示了这个实例

图89 霍尔传感器接口的实例



12.3.19 TIMx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。

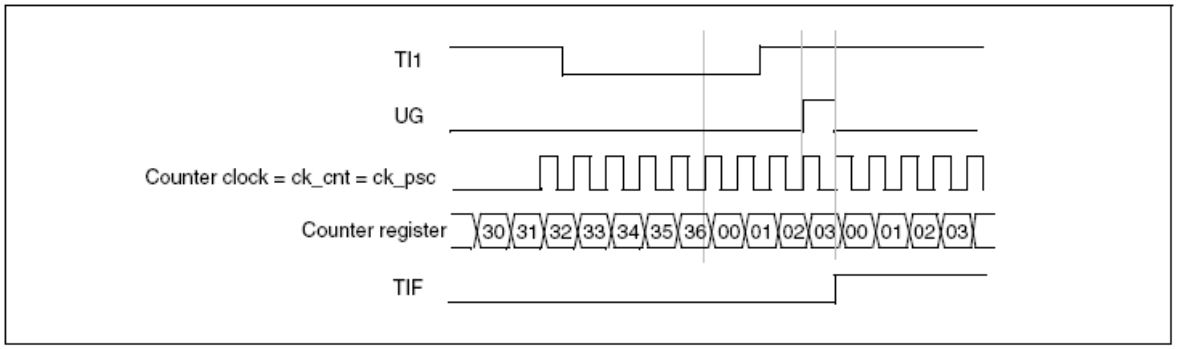
在以下的例子中，TI1输入端的上升沿导致向上计数器被清零：

- 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90 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。

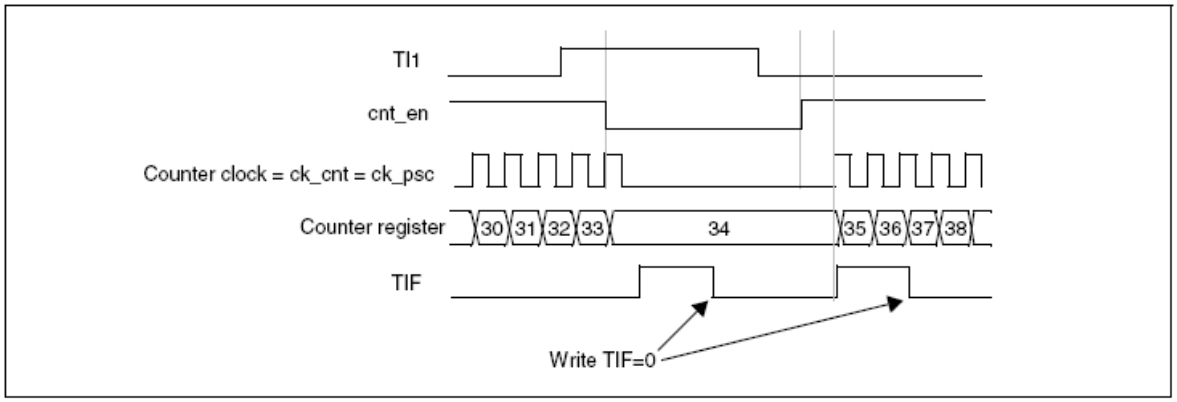
在如下的例子中，计数器只在TI1为低时向上计数：

- 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，一旦TI1变高则停止计数。当计数器开始或停止时都设置TIMx_SR中的TIF标置。

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91 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IC2F=0000)。触发操作中不使用捕获预分频器，不需要配置。CC2S位只用于选择输入捕

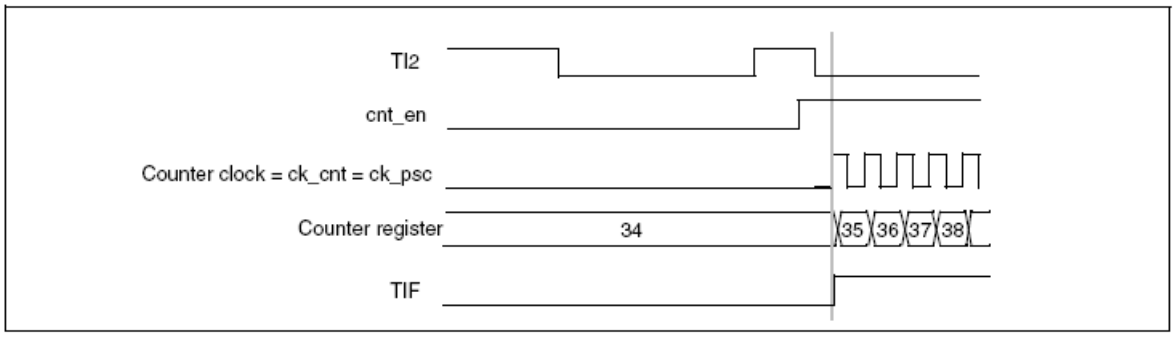
获源，置TIMx_CCMR1寄存器中CC2S=01。置TIMx_CCER寄存器中CC2P=1以确定极性(只检测低电平)。

- 置TIMx_SMCR寄存器中SMS=110，配置定时器为触发模式；置TIMx_SMCR寄存器中TS=110，选择TI2作为输入源。

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

TI2上升沿和计数器启动计数之间的延时取决于TI2输入端的重同步电路。

图92 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、门控模式或触发模式可以选择另一个输入作为触发输入。不建议使用TIMx_SMCR寄存器的TS位选择ETR作为TRGI。

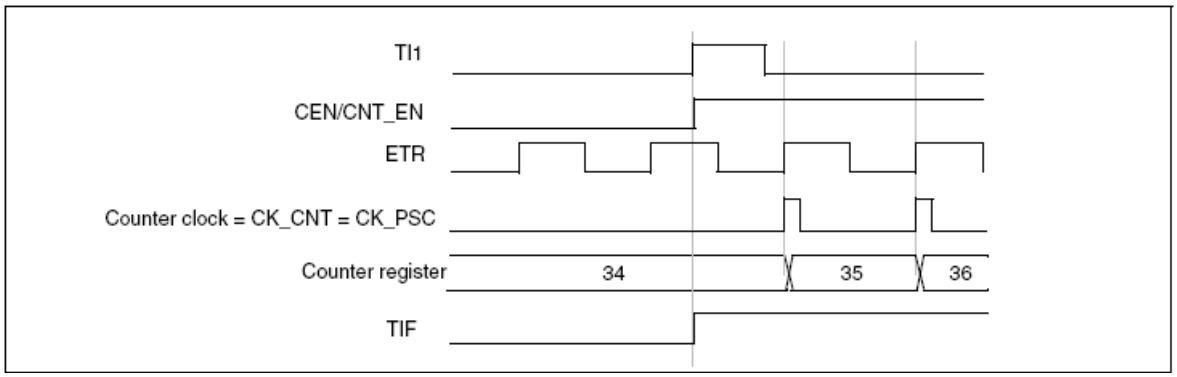
在下面的例子中，一旦在TI1上出现一个上升沿，计数器即在ETR的每一个上升沿向上计数一次：

1. 通过TIMx_SMCR寄存器配置外部触发输入电路：
   - ETF=0000：没有滤波
   - ETPS=00：不用预分频器
   - ETP=0：检测 ETR 的上升沿，置 ECE=1 使能外部时钟模式 2。
2. 按如下配置通道1，检测TI的上升沿：
   - IC1F=0000：没有滤波
   - 触发操作中不使用捕获预分频器，不需要配置
   - 置 TIMx_CCMR1 寄存器中 CC1S=01，选择输入捕获源
   - 置 TIMx_CCER 寄存器中 CC1P=0 以确定极性(只检测上升沿)
3. 置TIMx_SMCR寄存器中SMS=110，配置定时器为触发模式。置TIMx_SMCR寄存器中TS=101，选择TI1作为输入源。

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

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

图93 外部时钟模式2+触发模式下的控制电路



12.3.20 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13.3.15节。

12.3.21 调试模式

当微控制器进入调试模式时(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。详见随后的26.15.2节。

12.4 TIM1和TIM8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。

12.4.1 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、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：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/下溢才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：使能计数器。 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。

12.4.2 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	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(TIMx_BKR寄存器)级别1、2或3后，该位不能被修改。
位8	**OIS1**：输出空闲状态1(OC1输出)。 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(TIMx_BKR寄存器)级别1、2或3后，该位不能被修改。
位7	**TI1S**：TI1选择 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。

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(从模式控制器处于复位模式)产生复位，则TRGO上的信号相对实际的复位会有一个延迟。 001：**使能** – 计数器使能信号CNT_EN被用于作为触发输出(TRGO)。有时需要在同一时间启动多个定时器或控制在一段时间内使能从定时器。计数器使能信号是通过CEN控制位和门控模式下的触发输入信号的逻辑或产生。当计数器使能信号受控于触发输入时，TRGO上会有一个延迟，除非选择了主/从模式(见TIMx_SMCR寄存器中MSM位的描述)。 010：**更新** – 更新事件被选为触发输入(TRGO)。例如，一个主定时器的时钟可以被用作一个从定时器的预分频器。 011：**比较脉冲** – 一旦发生一次捕获或一次比较成功时，当要设置CC1IF标志时(即使它已经为高)，触发输出送出一个正脉冲(TRGO)。 100：**比较** – OC1REF信号被用于作为触发输出(TRGO)。 101：**比较** – OC2REF信号被用于作为触发输出(TRGO)。 110：**比较** – OC3REF信号被用于作为触发输出(TRGO)。 111：**比较** – OC4REF信号被用于作为触发输出(TRGO)。
位3	**CCDS**：捕获/比较的DMA选择 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。

12.4.3 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**：外部时钟使能位 该位启用外部时钟模式2 0：禁止外部时钟模式2； 1：使能外部时钟模式2。计数器由ETRF信号上的任意有效上升沿驱动。 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)。 注3：外部时钟模式1和外部时钟模式2同时被使能时，外部时钟的输入是ETRF。

位13:12	**ETPS[1:0]**：外部触发预分频 外部触发信号ETRP的频率必须最多是TIMxCLK频率的1/4。当输入较快的外部时钟时，可以使用预分频降低ETRP的频率。 00：关闭预分频； 01：ETRP频率除以2； 10：ETRP频率除以4； 11：ETRP频率除以8。
位11:8	**ETF[3:0]**：外部触发滤波 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0(ITR0) 001：内部触发1(ITR1) 010：内部触发2(ITR2) 011：内部触发3(ITR3) 100：TI1的边沿检测器(TI1F_ED) 101：滤波后的定时器输入1(TI1FP1) 110：滤波后的定时器输入2(TI2FP2) 111：外部触发输入(ETRF) 更多有关ITRx的细节，参见表55。 注：这些位只能在未用到(如SMS=000)时被改变，以避免在改变时产生错误的边沿检测。
位3	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 – 根据另一个输入的电平，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。

表55 TIMx内部触发连接

从定时器	ITR0(TS=000)	ITR1(TS=001)	ITR2(TS=010)	ITR3(TS=011)
TIM1	TIM5	TIM2	TIM3	TIM4
TIM8	TIM1	TIM2	TIM4	TIM5

12.4.4 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	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**：触发中断使能 0：禁止触发中断； 1：使能触发中断。
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。

12.4.5 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	BIF	TIF	COMIF	CC4IF	CC3IF	CC2IF	CC1IF	UIF
			rc_w0	rc_w0	rc_w0	rc_w0		rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0

位15:13	保留，始终读为0。
位12	**CC4OF**：捕获/比较4重复捕获标记 参见CC1OF描述。
位11	**CC3OF**：捕获/比较3重复捕获标记 参见CC1OF描述。
位10	**CC2OF**：捕获/比较2重复捕获标记 参见CC1OF描述。
位9	**CC1OF**：捕获/比较1重复捕获标记 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：无重复捕获产生； 1：计数器的值被捕获到TIMx_CCR1寄存器时，CC1IF的状态已经为1。
位8	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：触发中断等待响应。
位5	**COMIF**：COM中断标记 一旦产生COM事件(当捕获/比较控制位：CCxE、CCxNE、OCxM已被更新)该位由硬件置1。它由软件清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：计数器值已被捕获(拷贝)至TIMx_CCR1(在IC1上检测到与所选极性相同的边沿)。
位0	**UIF**：更新中断标记 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(重复计数器上溢或下溢时)。 – 若TIMx_CR1寄存器的UDIS=0、URS=0，当设置TIMx_EGR寄存器的UG=1时产生更新事件(软件对计数器CNT重新初始化)。 – 若TIMx_CR1寄存器的UDIS=0、URS=0，当计数器CNT被触发事件重新初始化时产生更新事件。(参考12.4.3: 从模式控制寄存器(TIMx_SMCR))。

12.4.6 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								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：TIMx_SR寄存器的TIF=1，若开启对应的中断和DMA，则产生相应的中断和DMA。
位5	**COMG**：捕获/比较事件，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**：该位只对拥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：** 当前的计数器值被捕获至TIMx_CCR1寄存器，设置CC1IF=1，若开启对应的中断和DMA，则产生相应的中断和DMA。若CC1IF已经为1，则设置CC1OF=1。
位0	**UG**：产生更新事件 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TIMx_ARR的值。

12.4.7 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位定义。该寄存器其它位的作用在输入和输出模式下不同。OCxx描述了通道在输出模式下的功能，ICxx描述了通道在输入模式下的功能。因此必须注意，同一个位在输出模式和输入模式下的功能是不同的。

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC2CE	OC2M[2:0]			OC2PE	OC2FE	CC2S[1:0]		OC1CE	OC1M[2:0]			OC1PE	OC1FE	CC1S[1:0]	
IC2F[3:0]				IC2PSC[1:0]				IC1F[3:0]				IC1PSC[1:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	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3位定义了输出参考信号OC1REF的动作，而OC1REF决定了OC1、OC1N的值。OC1REF是高电平有效，而OC1、OC1N的有效电平取决于CC1P、CC1NP位。 000：冻结。输出比较寄存器TIMx_CCR1与计数器TIMx_CNT间的比较对OC1REF不起作用； 001：匹配时设置通道1为有效电平。当计数器TIMx_CNT的值与捕获/比较寄存器1(TIMx_CCR1)相同时，强制OC1REF为高。 010：匹配时设置通道1为无效电平。当计数器TIMx_CNT的值与捕获/比较寄存器1(TIMx_CCR1)相同时，强制OC1REF为低。 011：翻转。当TIMx_CCR1=TIMx_CNT时，翻转OC1REF的电平。 100：强制为无效电平。强制OC1REF为低。 101：强制为有效电平。强制OC1REF为高。 110：PWM模式1－ 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，并且新写入的数值立即起作用。 1：开启TIMx_CCR1寄存器的预装载功能，读写操作仅对预装载寄存器操作，TIMx_CCR1的预装载值在更新事件到来时被加载至当前寄存器中。 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(TIMx_CR1寄存器的OPM=1)，可以在未确认预装载寄存器情况下使用PWM模式，否则其动作不确定。
位2	**OC1FE**：输出比较1 快速使能 该位用于加快CC输出对触发输入事件的响应。 0：根据计数器与CCR1的值，CC1正常操作，即使触发器是打开的。当触发器的输入有一个有效沿时，激活CC1输出的最小延时为5个时钟周期。 1：输入到触发器的有效沿的作用就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[3:0]**：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
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。

12.4.8 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。 该2位定义通道的方向(输入/输出)，及输入脚的选择： 00：CC4通道被配置为输出； 01：CC4通道被配置为输入，IC4映射在TI4上； 10：CC4通道被配置为输入，IC4映射在TI3上； 11：CC4通道被配置为输入，IC4映射在TRC上。此模式仅工作在内部触发器输入被选中时(由TIMx_SMCR寄存器的TS位选择)。 **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	**OC3M[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[3:0]**：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	**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7:4	**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，IC3映射在TRC上。此模式仅工作在内部触发器输入被选中时（由TIMx_SMCR寄存器的TS位选择）。 注：CC3S仅在通道关闭时(TIMx_CCER寄存器的CC3E=0)才是可写的。

12.4.9 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	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(TIMx_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和CC1E位的值。 1：开启－ OC1N信号输出到对应的输出引脚，其输出电平依赖于MOE、OSSI、OSSR、OIS1、OIS1N和CC1E位的值。
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(TIMx_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和CC1NE位的值。 1： 开启－ OC1信号输出到对应的输出引脚，其输出电平依赖于MOE、OSSI、OSSR、OIS1、OIS1N和CC1NE位的值。 **CC1通道配置为输入：** 该位决定了计数器的值是否能捕获入TIMx_CCR1寄存器。 0：捕获禁止； 0：捕获使能。

表56 带刹车功能的互补输出通道OCx和OCxN的控制位

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

1. 如果一个通道的2个输出都没有使用(CCxE = CCxNE = 0)，那么OISx, OISxN, CCxP和CCxNP都必须清零。

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。*

12.4.10 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

12.4.11 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。

12.4.12 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12.3.1节：有关ARR的更新和动作。 当自动重装载的值为空时，计数器不工作。

12.4.13 重复计数寄存器(TIMx_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]**: 重复计数器的值 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)；如果允许产生更新中断，则会同时影响产生更新中断的速率。 每次向下计数器REP_CNT达到0，会产生一个更新事件并且计数器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TIMx_RCR寄存器写入的新值只在下次周期更新事件发生时才起作用。 这意味着在PWM模式中，(REP+1)对应着： — 在边沿对齐模式下，PWM周期的数目； — 在中心对称模式下，PWM半周期的数目；

12.4.14 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包含了装入当前捕获/比较1寄存器的值(预装载值)。 如果在TIMx_CCMR1寄存器(OC1PE位)中未选择预装载功能，写入的数值会立即传输至当前寄存器中。否则只有当更新事件发生时，此预装载值才传输至当前捕获/比较1寄存器中。 当前捕获/比较寄存器参与同计数器TIMx_CNT的比较，并在OC1端口上产生输出信号。 **若CC1通道配置为输入：** CCR1包含了由上一次输入捕获1事件(IC1)传输的计数器值。

12.4.15 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包含了装入当前捕获/比较2寄存器的值(预装载值)。 如果在TIMx_CCMR2寄存器(OC2PE位)中未选择预装载特性，写入的数值会立即传输至当前寄存器中。否则只有当更新事件发生时，此预装载值才传输至当前捕获/比较2寄存器中。 当前捕获/比较寄存器参与同计数器TIMx_CNT的比较，并在OC2端口上产生输出信号。 **若CC2通道配置为输入：** CCR2包含了由上一次输入捕获2事件(IC2)传输的计数器值。

12.4.16 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包含了装入当前捕获/比较3寄存器的值(预装载值)。 如果在TIMx_CCMR3寄存器(OC3PE位)中未选择预装载特性，写入的数值会立即传输至当前寄存器中。否则只有当更新事件发生时，此预装载值才传输至当前捕获/比较3寄存器中。 当前捕获/比较寄存器参与同计数器TIMx_CNT的比较，并在OC3端口上产生输出信号。 **若CC3通道配置为输入：** CCR3包含了由上一次输入捕获3事件(IC3)传输的计数器值。

12.4.17 捕获/比较寄存器(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包含了装入当前捕获/比较4寄存器的值(预装载值)。 如果在TIMx_CCMR4寄存器(OC4PE位)中未选择预装载特性，写入的数值会立即传输至当前寄存器中。否则只有当更新事件发生时，此预装载值才传输至当前捕获/比较4寄存器中。 当前捕获/比较寄存器参与同计数器TIMx_CNT的比较，并在OC4端口上产生输出信号。 **若CC4通道配置为输入：** CCR4包含了由上一次输入捕获4事件(IC4)传输的计数器值。

12.4.18 刹车和死区寄存器(TIMx_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： *根据锁定设定，AOE、BKP、BKE、OSSI、OSSR和DTG[7:0]位均可被写保护，有必要在第一次写入TIMx_BDTR寄存器时对它们进行配置。*

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位的设置值，该位可以由软件清0或被自动置1。它仅对配置为输出的通道有效。 0：禁止OC和OCN输出或强制为空闲状态； 1：如果设置了相应的使能位(TIMx_CCER寄存器的CCxE、CCxNE位)，则开启OC和OCN输出。 有关OC/OCN使能的细节，参见12.4.9节，捕获/比较使能寄存器(TIMx_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被自动置1(如果刹车输入无效)。 注：一旦LOCK级别(TIMx_BDTR寄存器中的LOCK位)设为1，则该位不能被修改。
位13	**BKP**: 刹车输入极性 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(TIMx_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(TIMx_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(12.4.9节，捕获/比较使能寄存器(TIMx_CCER))。 0：当定时器不工作时，禁止OC/OCN输出(OC/OCN使能输出信号=0)； 1：当定时器不工作时，一旦CCxE=1或CCxNE=1，首先开启OC/OCN并输出无效电平，然后置OC/OCN使能输出信号=1。 注：一旦LOCK级别(TIMx_BDTR寄存器中的LOCK位)设为2，则该位不能被修改。
位10	**OSSI**: 空闲模式下“关闭状态”选择 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(12.4.9节，捕获/比较使能寄存器(TIMx_CCER))。 0：当定时器不工作时，禁止OC/OCN输出(OC/OCN使能输出信号=0)； 1：当定时器不工作时，一旦CCxE=1或CCxNE=1，OC/OCN首先输出其空闲电平，然后OC/OCN使能输出信号=1。 注：一旦LOCK级别(TIMx_BDTR寄存器中的LOCK位)设为2，则该位不能被修改。
位9:8	**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TIMx_BDTR寄存器的DTG、BKE、BKP、AOE位和TIMx_CR2寄存器的OISx/OISxN位； 10：锁定级别2，不能写入锁定级别1中的各位，也不能写入CC极性位(一旦相关通道通过CCxS位设为输出，CC极性位是TIMx_CCER寄存器的CCxP/CCNxP位)以及OSSR/OSSI位； 11：锁定级别3，不能写入锁定级别2中的各位，也不能写入CC控制位(一旦相关通道通过CCxS位设为输出，CC控制位是TIMx_CCMRx寄存器的OCxM/OCxPE位)； 注：在系统复位后，只能写一次LOCK位，一旦写入TIMx_BDTR寄存器，则其内容冻结直至复位。
位7:0	**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(TIMx_BDTR寄存器中的LOCK位)设为1、2或3，则不能修改这些位。

12.4.19 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寄存器进行读或写时，定时器则进行一次连续传送)，即：定义传输的次数，传输可以是半字(双字节)或字节： 00000：1次传输　　00001：2次传输 00010：3次传输　　..... .....　　10001：18次传输 例：我们考虑这样的传输：DBL=7，DBA=TIM2_CR1 - 如果DBL=7，DBA=TIM2_CR1表示待传输数据的地址，那么传输的地址由下式给出： (TIMx_CR1的地址) + DBA + (DMA索引), 其中 DMA索引 = DBL 其中(TIMx_CR1的地址) + DBA再加上7，给出了将要写入或者读出数据的地址，这样数据的传输将发生在从地址(TIMx_CR1的地址) + DBA开始的7个寄存器。 根据DMA数据长度的设置，可能发生以下情况： - 如果设置数据为半字(16位)，那么数据就会传输给全部7个寄存器。 - 如果设置数据为字节，数据仍然会传输给全部7个寄存器：第一个寄存器包含第一个MSB字节，第二个寄存器包含第一个LSB字节，以此类推。因此对于定时器，用户必须指定由DMA传输的数据宽度。
位7:5	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， .....

12.4.20 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所在寄存器的存取操作： TIMx_CR1地址 + DBA + DMA索引，其中： “TIMx_CR1地址”是控制寄存器1(TIMx_CR1)所在的地址； “DBA”是TIMx_DCR寄存器中定义的基地址； “DMA索引”是由DMA自动控制的偏移量，它取决于TIMx_DCR寄存器中定义的DBL。

12.4.21 TIM1 和TIM8 寄存器图

下表中将TIM1和TIM8的所有寄存器映射到一个16位可寻址(编址)空间。

表57 TIM1和TIM8 – 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																	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	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	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	TIMx_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	TIMx_EGR	保留																								BG	TG	COM	CC4G	CC3G	CC2G	CC1G	UG
	复位值																									0	0	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	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	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	TIMx_RCR	保留																								REP[7:0]							
	复位值																									0	0	0	0	0	0	0	0
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	TIMx_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	TIMx_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	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

13 通用定时器(TIMx)

小容量产品是指闪存存储器容量在16K至32K字节之间的STM32F101xx、STM32F102xx和STM32F103xx微控制器。

中容量产品是指闪存存储器容量在64K至128K字节之间的STM32F101xx、STM32F102xx和STM32F103xx微控制器。

大容量产品是指闪存存储器容量在256K至512K字节之间的STM32F101xx和STM32F103xx微控制器。

除非特别说明，本章节描述的模块应用于整个STM32F10xxx微控制器系列。

13.1 TIMx简介

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(输入捕获)或者产生输出波形(输出比较和PWM)。

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

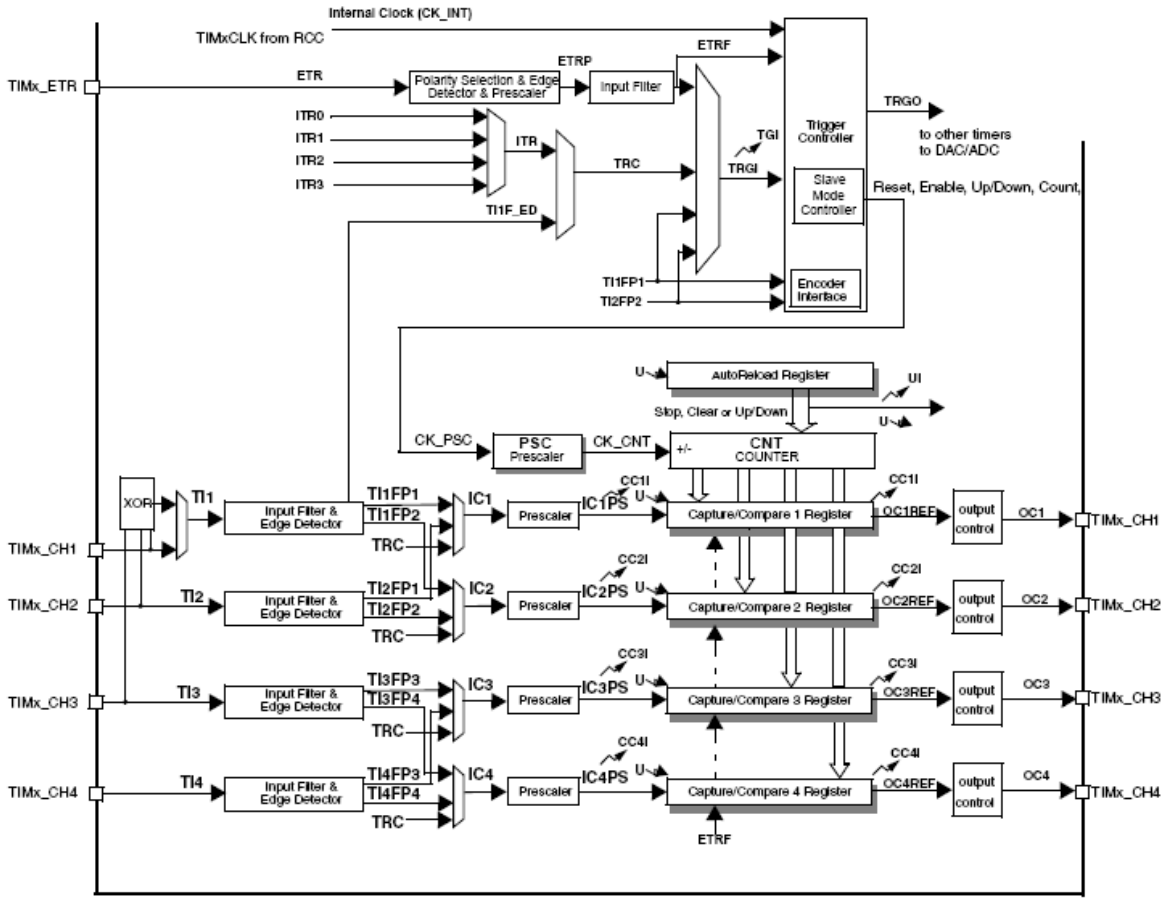
定时器是完全独立的，而且没有互相共享任何资源。它们可以一起同步操作，参见13.3.15节。

13.2 TIMx主要功能

通用TIMx (TIM2、TIM3、TIM4和TIM5)定时器功能包括：

- 16位向上、向下、向上/向下自动装载计数器
- 16位可编程(可以实时修改)预分频器，计数器时钟频率的分频系数为1～65535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
  - 支持针对定位的增量(正交)编码器和霍尔传感器电路
  - 触发输入作为外部时钟或者按周期的电流管理

图94 通用定时器框图



注： Reg *根据控制位的设定，在U事件时传送预加载寄存器的内容至工作寄存器*

事件

中断和DMA输出

13.3 TIMx功能描述

13.3.1 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，在计数器运行时仍可以读写。

时基单元包含：

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

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

计数器由预分频器的时钟输出CK_CNT驱动，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95和图96给出了在预分频器运行时，更改计数器参数的例子。

图95 当预分频器的参数从1变到2时，计数器的时序图

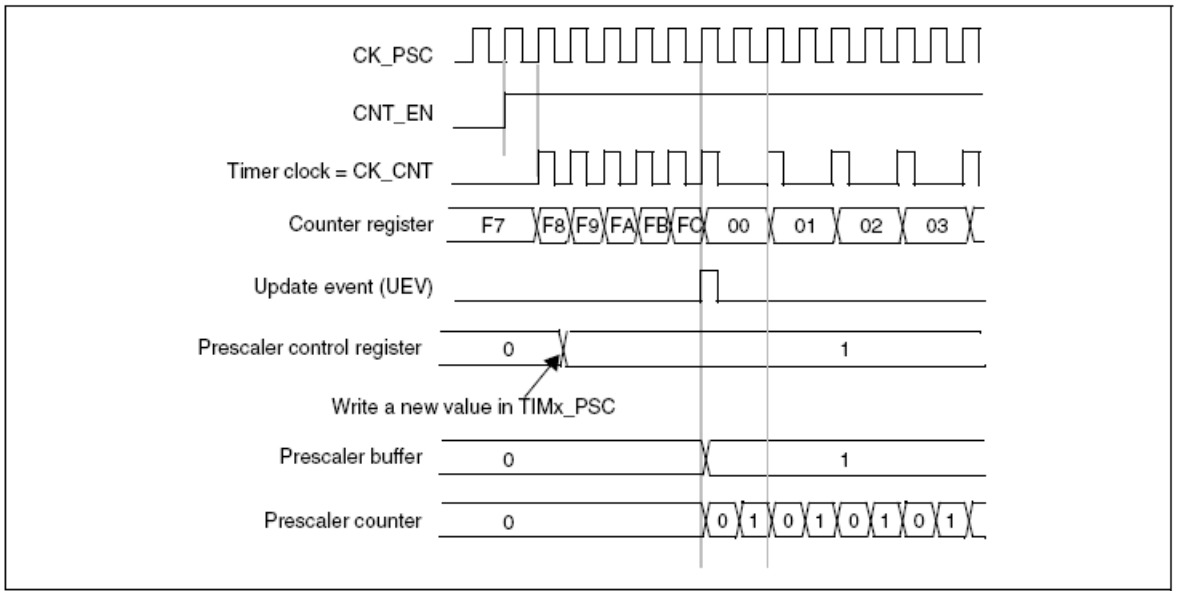
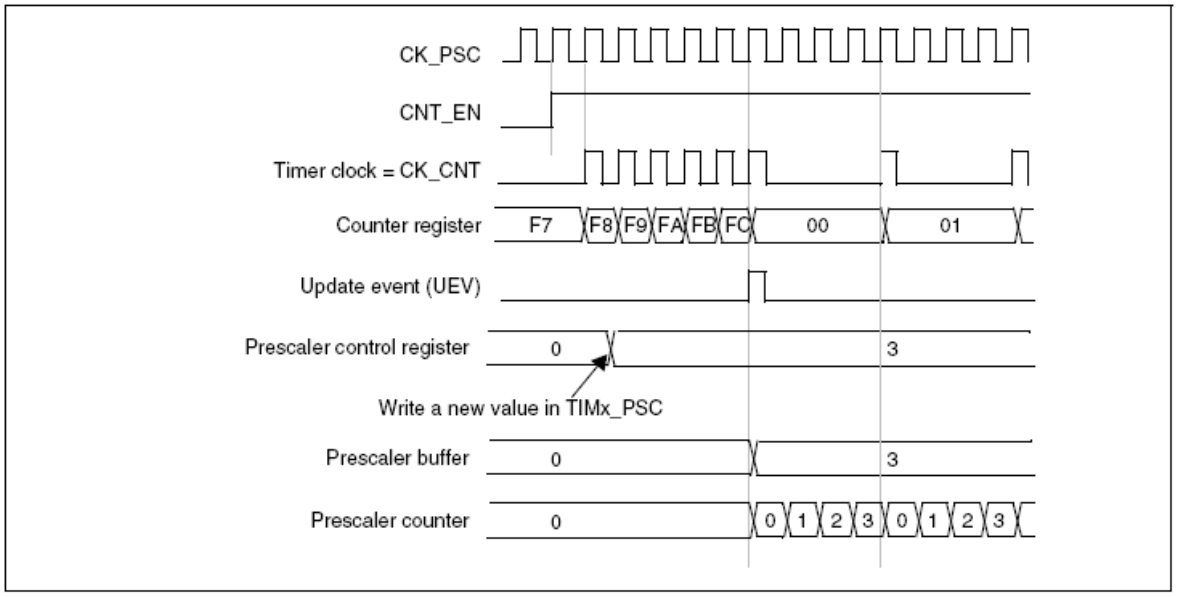


图96 当预分频器的参数从1变到4时，计数器的时序图



13.3.2 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。

设置TIMx_CR1寄存器中的UDIS位，可以禁止更新事件；这样可以避免在向预装载寄存器中写入新值时更新影子寄存器。在UDIS位被清0之前，将不产生更新事件。但是在应该产生更新事件时，计数器仍会被清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，如果设置了TIMx_CR1寄存器中的URS位(选择更新请求)，设置UG位将产生一个更新事件UEV，但硬件不设置UIF标志(即不产生中断或DMA请求)。这是为了避免在捕获模式下清除计数器时，同时产生更新和捕获中断。

当发生一个更新事件时，所有的寄存器都被更新，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97 计数器时序图，内部时钟分频因子为1

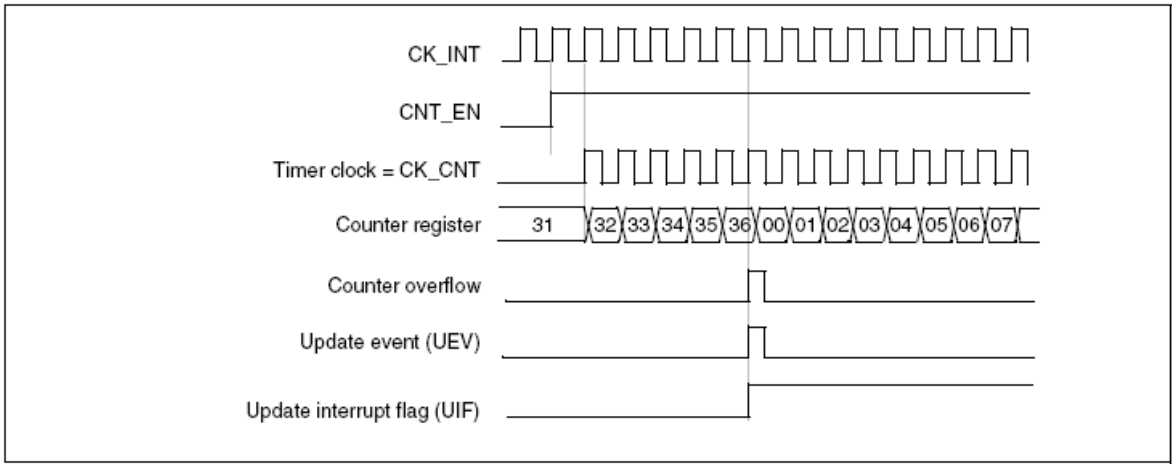


图98 计数器时序图，内部时钟分频因子为2

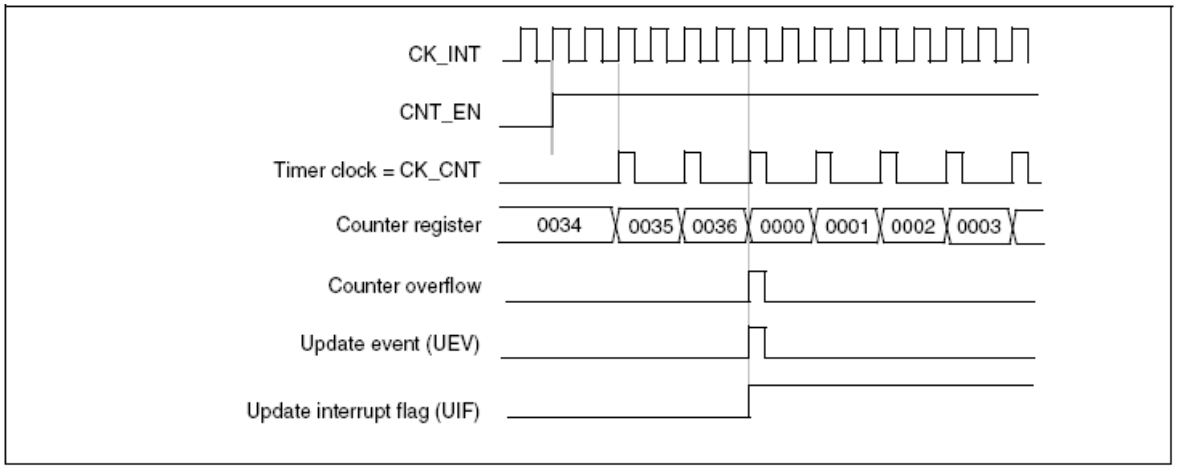


图99 计数器时序图，内部时钟分频因子为4

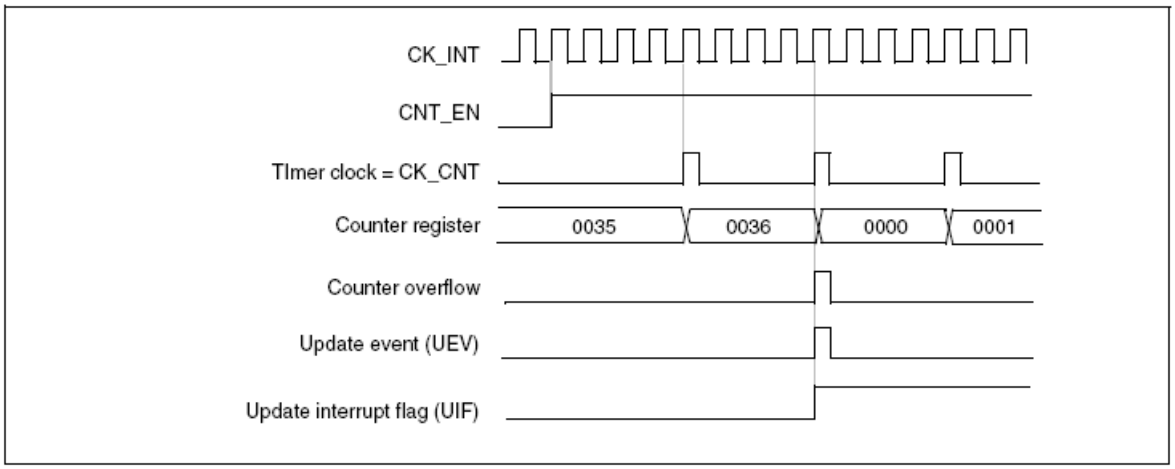


图100 计数器时序图，内部时钟分频因子为N

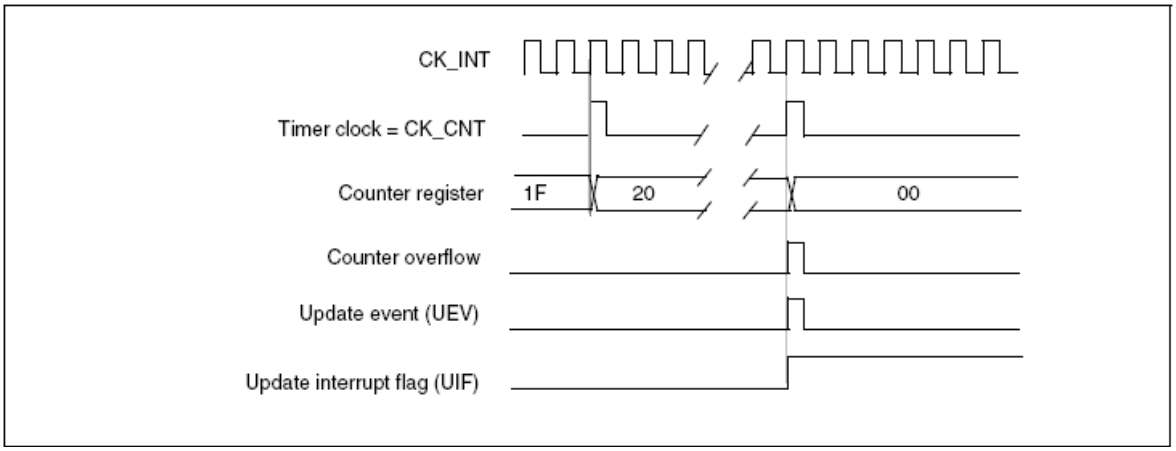


图101 计数器时序图，当ARPE=0时的更新事件(TIMx_ARR没有预装入)

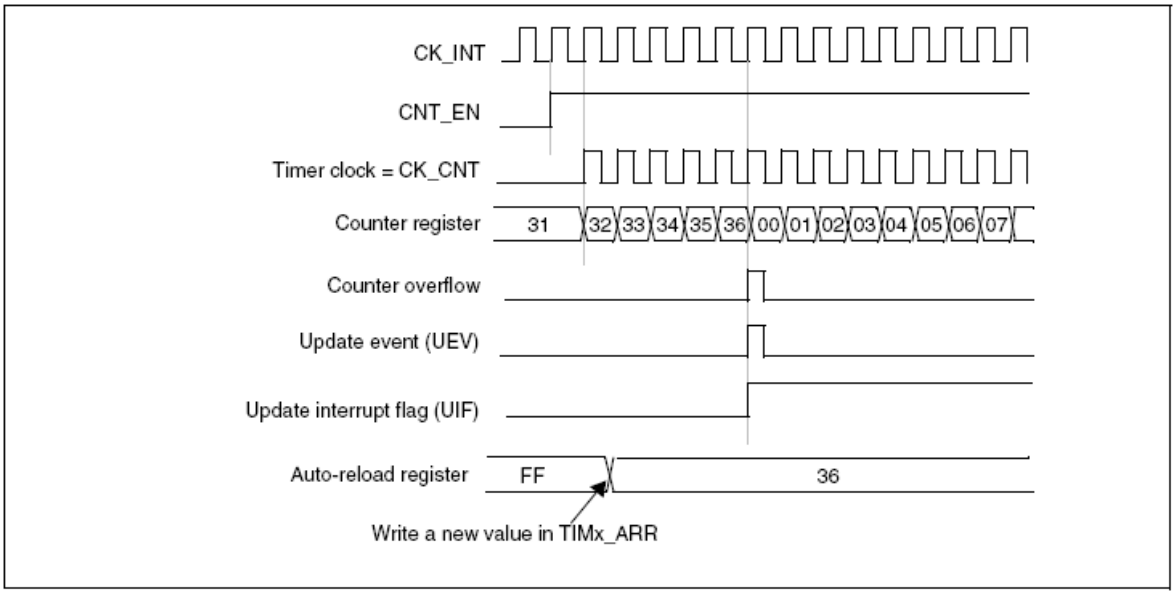
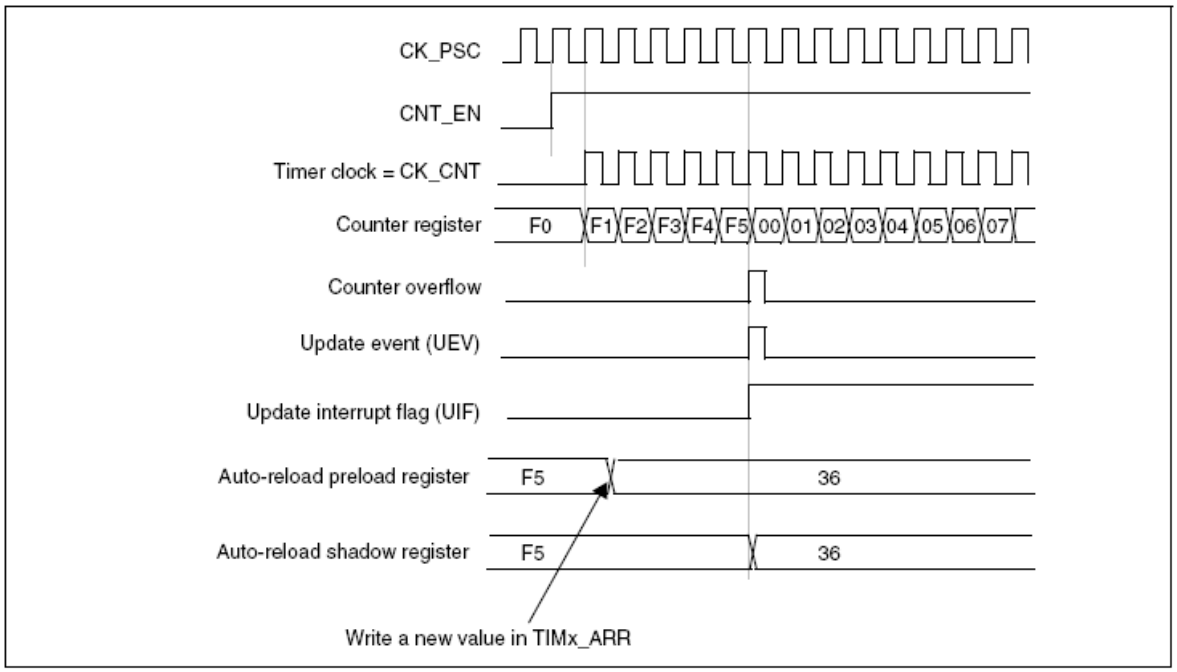


图102 计数器时序图，当ARPE=1时的更新事件(预装入了TIMx_ARR)



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。

设置TIMx_CR1寄存器的UDIS位可以禁止UEV事件。这样可以避免向预装载寄存器中写入新值时更新影子寄存器。因此UDIS位被清为0之前不会产生更新事件。然而，计数器仍会从当前自动加载值重新开始计数，同时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(选择更新请求) ，设置UG位将产生一个更新事件UEV但不设置UIF标志(因此不产生中断和DMA请求)，这是为了避免在发生捕获事件并清除计数器时，同时产生更新和捕获中断。

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

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

以下是一些当TIMx_ARR=0x36时，计数器在不同时钟频率下的操作例子。

图103 计数器时序图，内部时钟分频因子为1

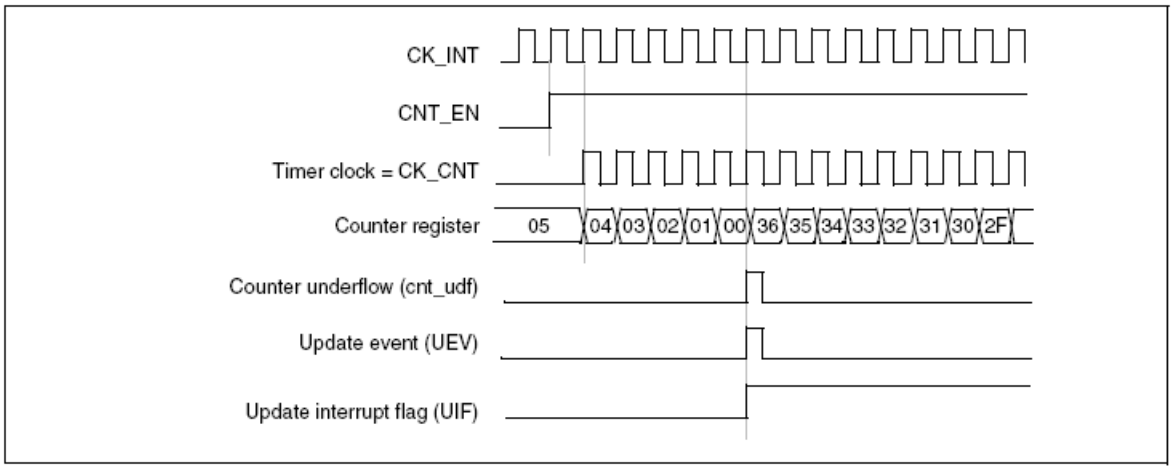


图104 计数器时序图，内部时钟分频因子为2

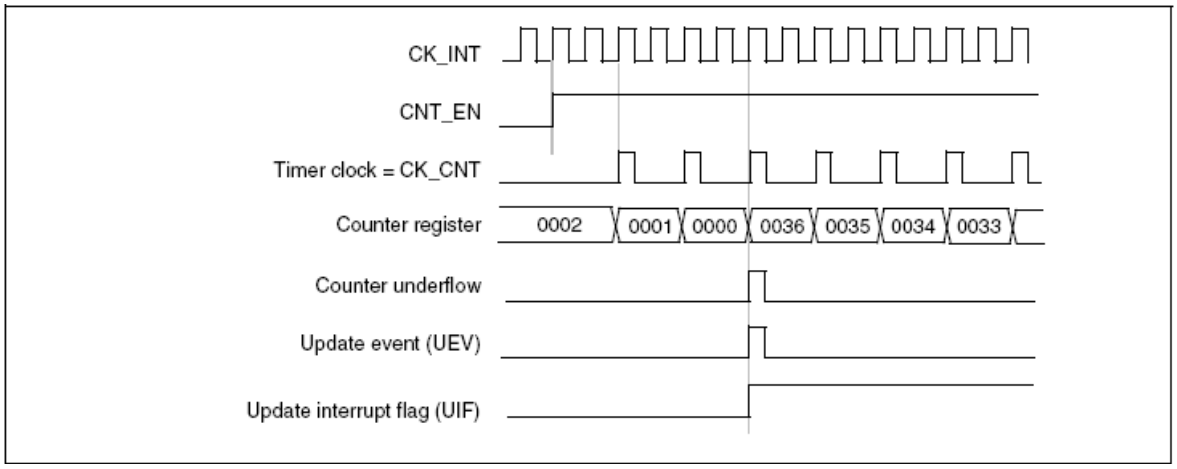


图105 计数器时序图，内部时钟分频因子为4

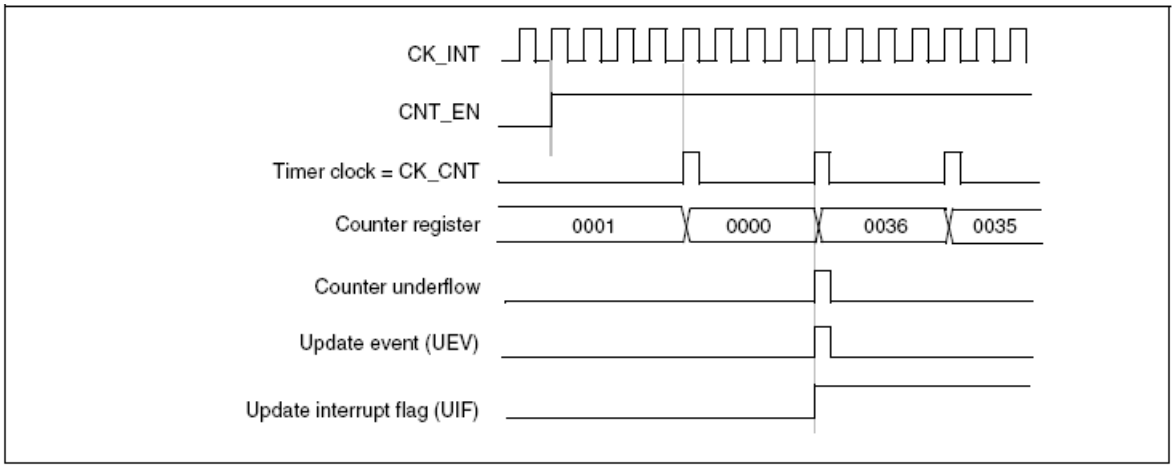


图106 计数器时序图，内部时钟分频因子为N

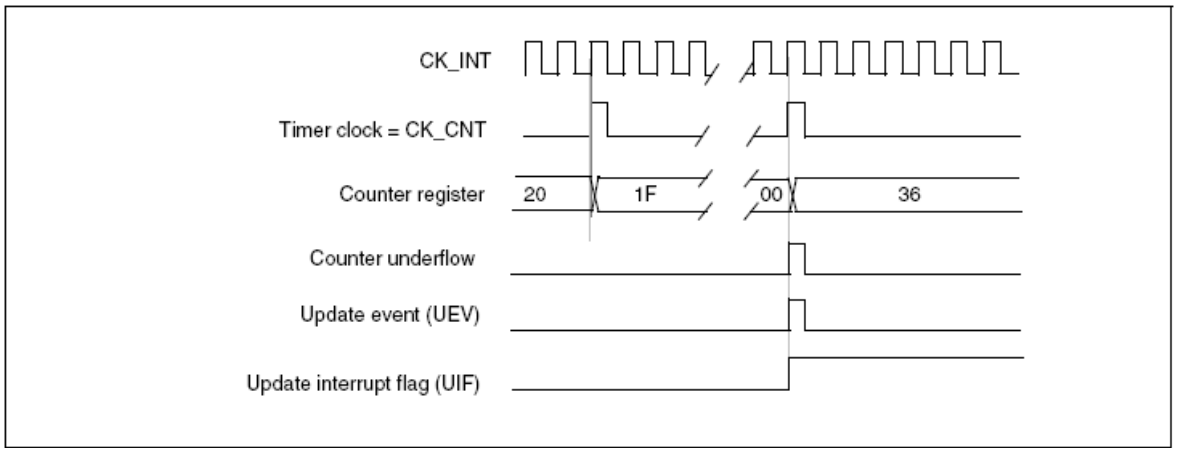
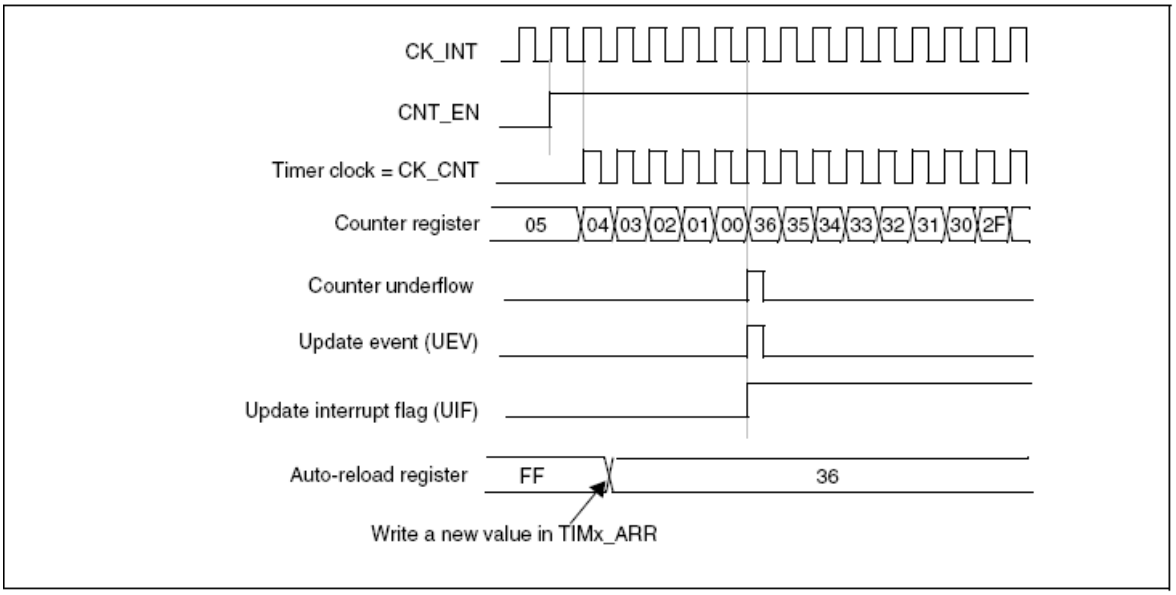


图107 计数器时序图，当没有使用重复计数器时的更新事件



中央对齐模式(向上/向下计数)

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

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

更新事件可以产生在每次计数溢出和每次计数下溢；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位产生，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。

设置TIMx_CR1寄存器中的UDIS位可以禁止UEV事件。这样可以避免在向预装载寄存器中写入新值时更新影子寄存器。因此UDIS位被清为0之前不会产生更新事件。然而，计数器仍会根据当前自动重加载的值，继续向上或向下计数。

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

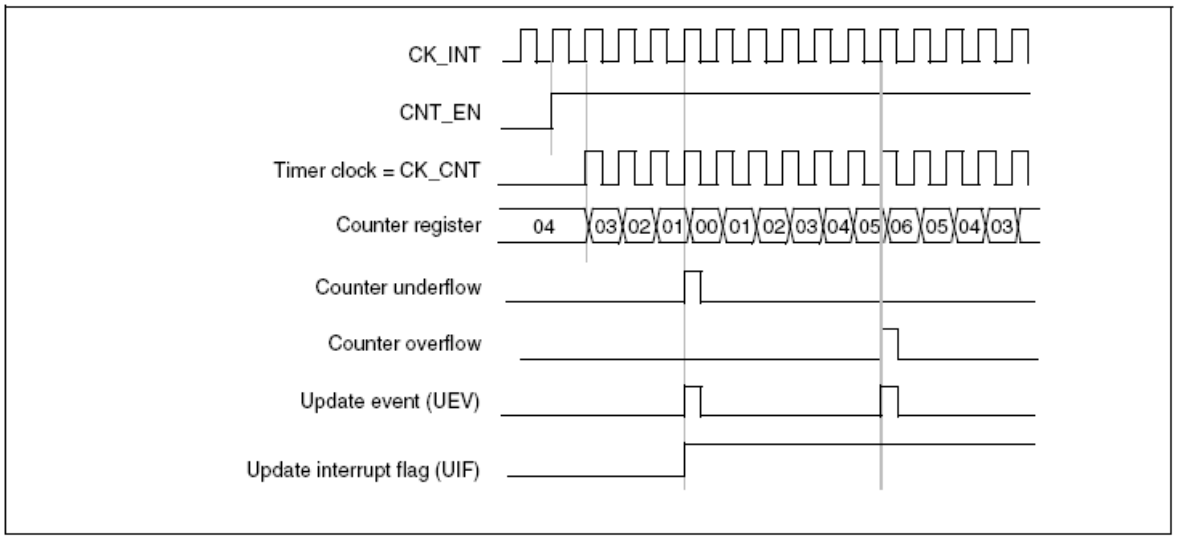
当发生更新事件时，所有的寄存器都被更新，并且(根据URS位的设置)更新标志位(TIMx_SR寄存器中的UIF位)也被设置。

- 预分频器的缓存器被加载为预装载(TIMx_PSC寄存器)的值。

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

以下是一些计数器在不同时钟频率下的操作的例子：

图108 计数器时序图，内部时钟分频因子为1，TIMx_ARR=0x6



1. 这里使用了中心对齐模式 1(详见13.4.1节)。

图109 计数器时序图，内部时钟分频因子为2

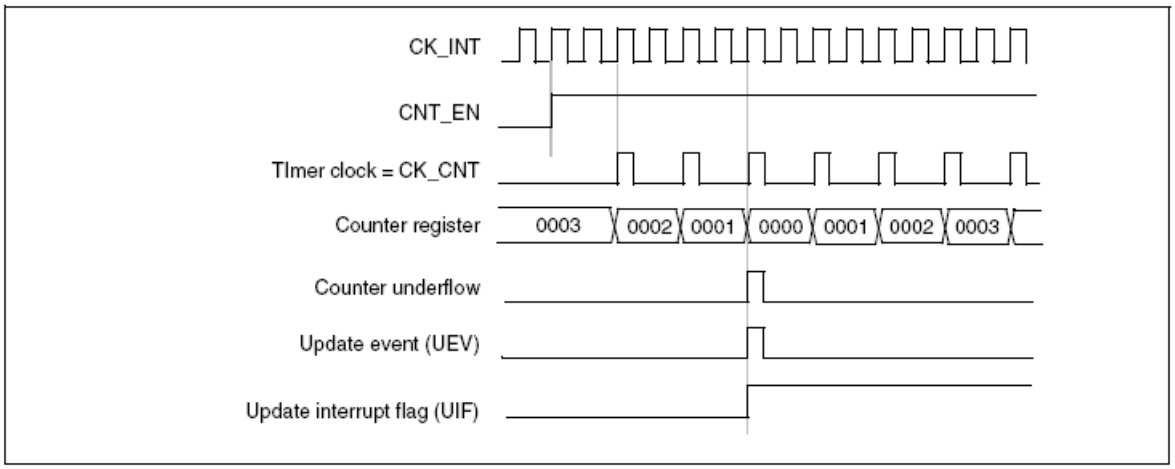


图110 计数器时序图，内部时钟分频因子为4，TIMx_ARR=0x36

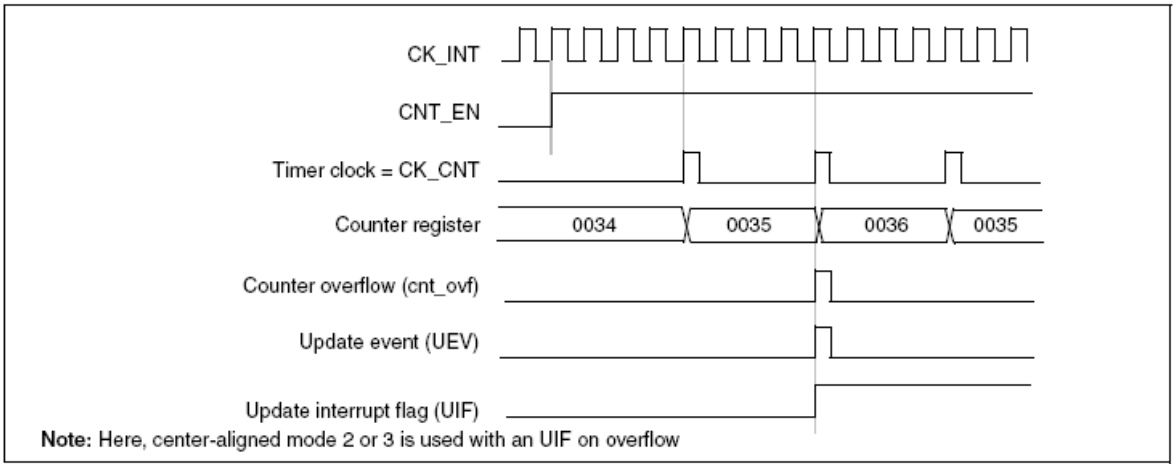


图111 计数器时序图，内部时钟分频因子为N

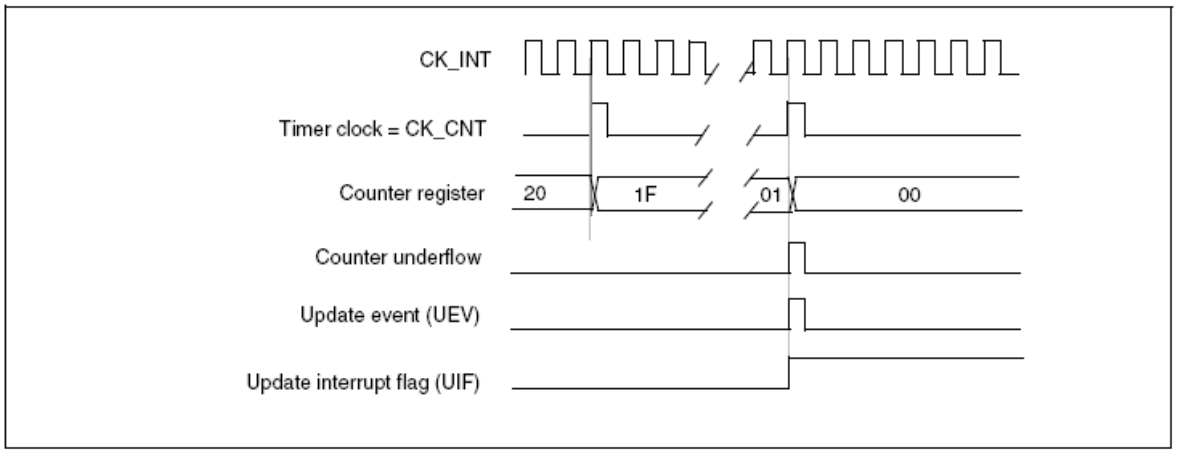


图112 计数器时序图，ARPE=1时的更新事件(计数器下溢)

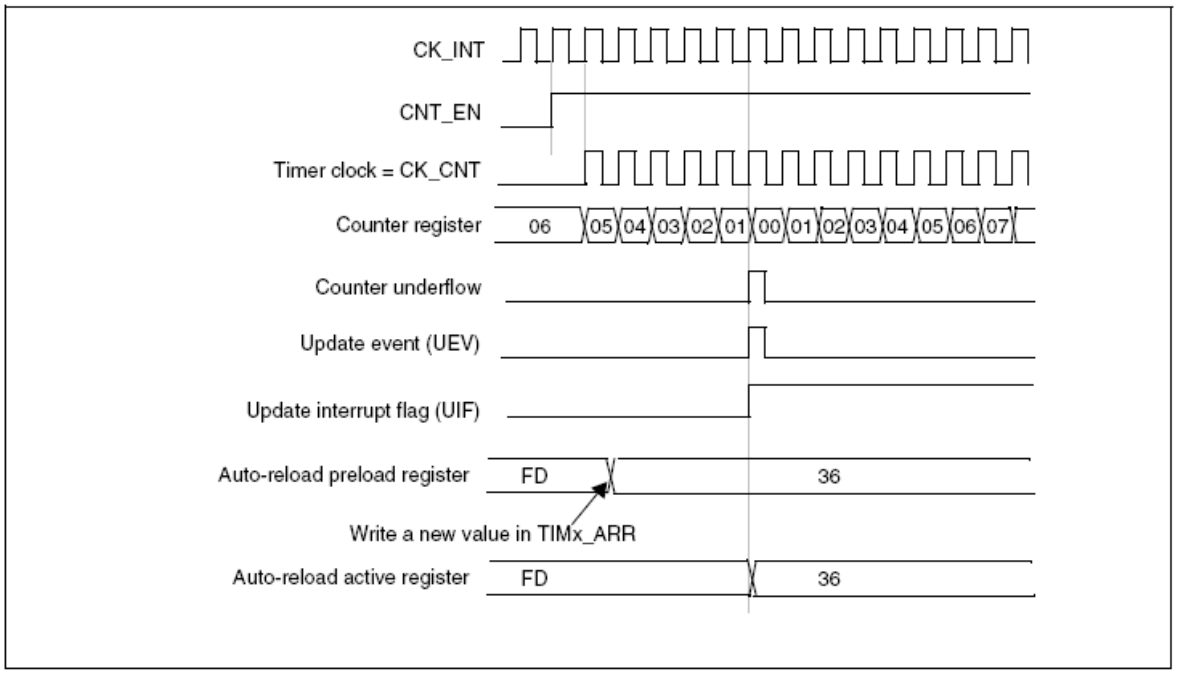
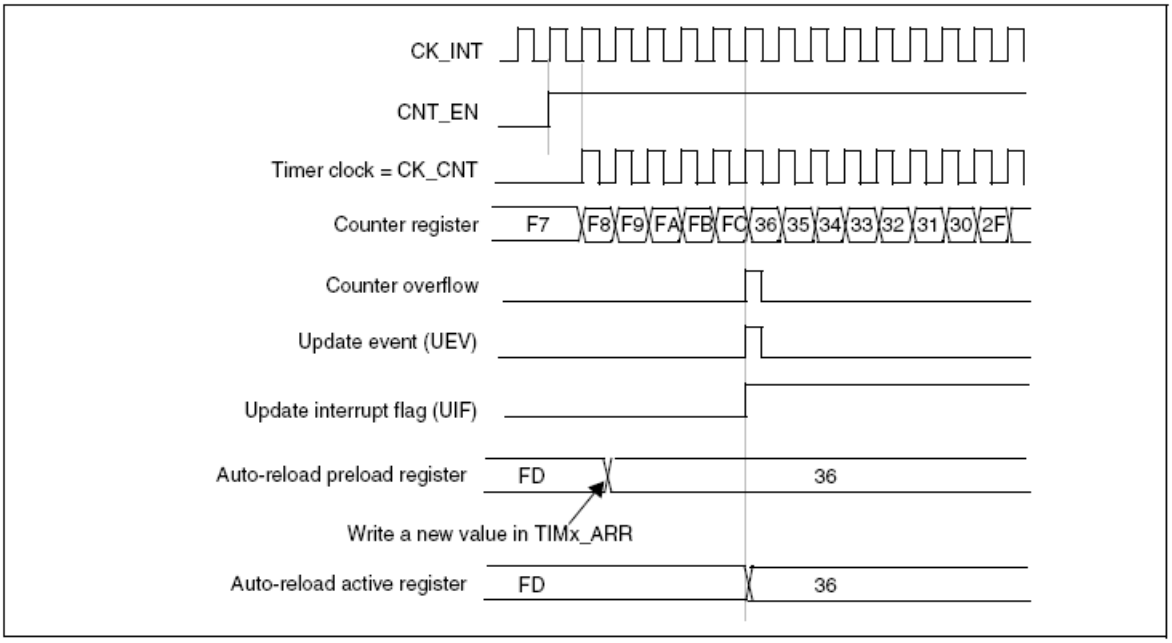


图113 计数器时序图，ARPE=1时的更新事件(计数器溢出)



13.3.3 时钟选择

计数器时钟可由下列时钟源提供：

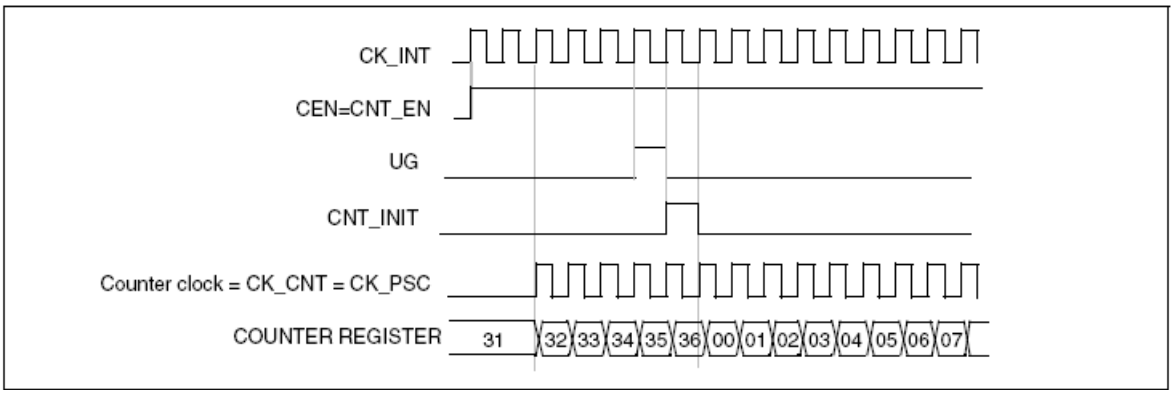
- 内部时钟(CK_INT)
- 外部时钟模式1：外部输入脚(TIx)
- 外部时钟模式2：外部触发输入(ETR)
- 内部触发输入(ITRx)：使用一个定时器作为另一个定时器的预分频器，如可以配置一个定时器Timer1而作为另一个定时器Timer2的预分频器。参见13.3.15。

内部时钟源(CK_INT)

如果禁止了从模式控制器(SMS=000)，则CEN、DIR(TIMx_CR1寄存器)和UG位(TIMx_EGR寄存器)是事实上的控制位，并且只能被软件修改(UG位仍被自动清除)。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。

下图显示了控制电路和向上计数器在一般模式下，不带预分频器时的操作。

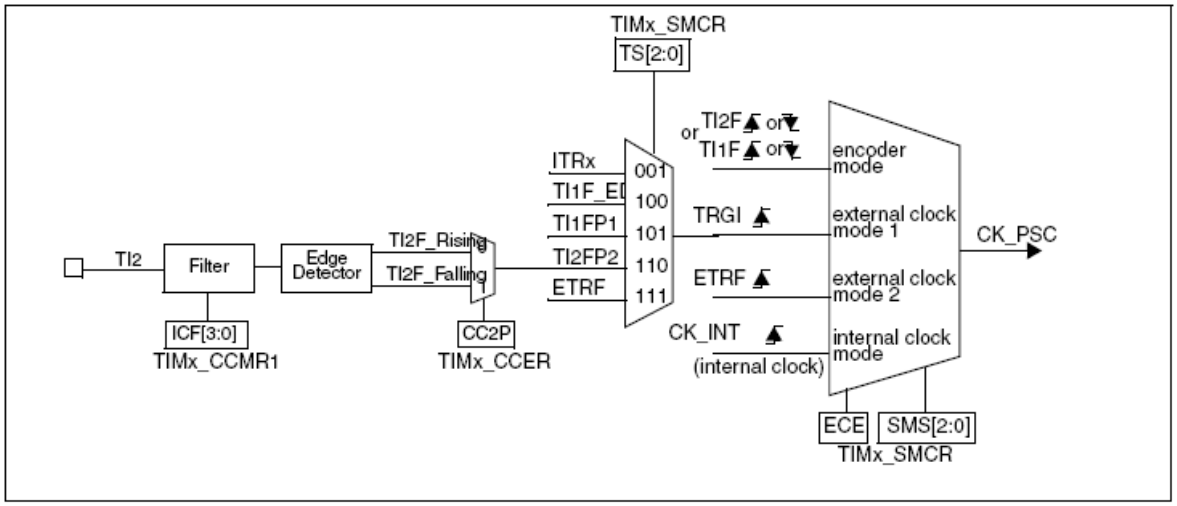
图114 一般模式下的控制电路，内部时钟分频因子为1



外部时钟源模式1

当TIMx_SMCR寄存器的SMS=111时，此模式被选中。计数器可以在选定输入端的每个上升沿或下降沿计数。

图115 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)

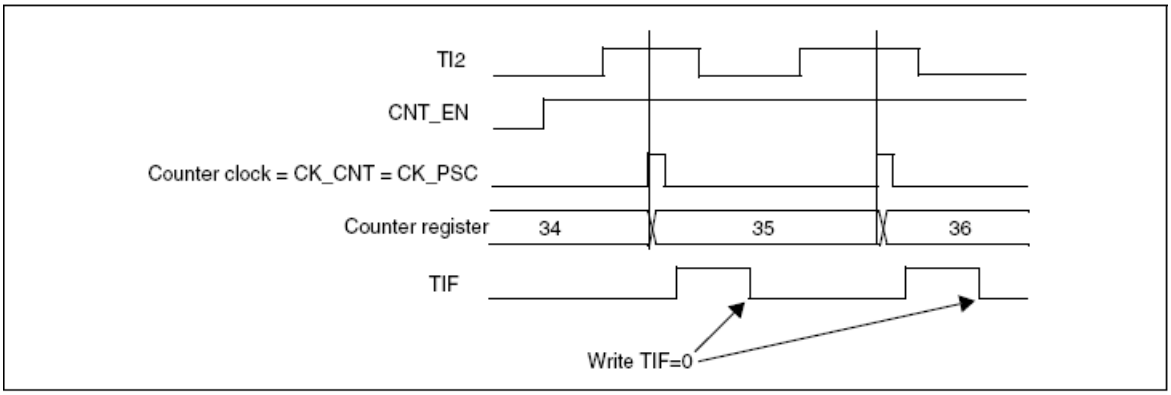
注： *捕获预分频器不用作触发，所以不需要对它进行配置*

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116 外部时钟模式1下的控制电路



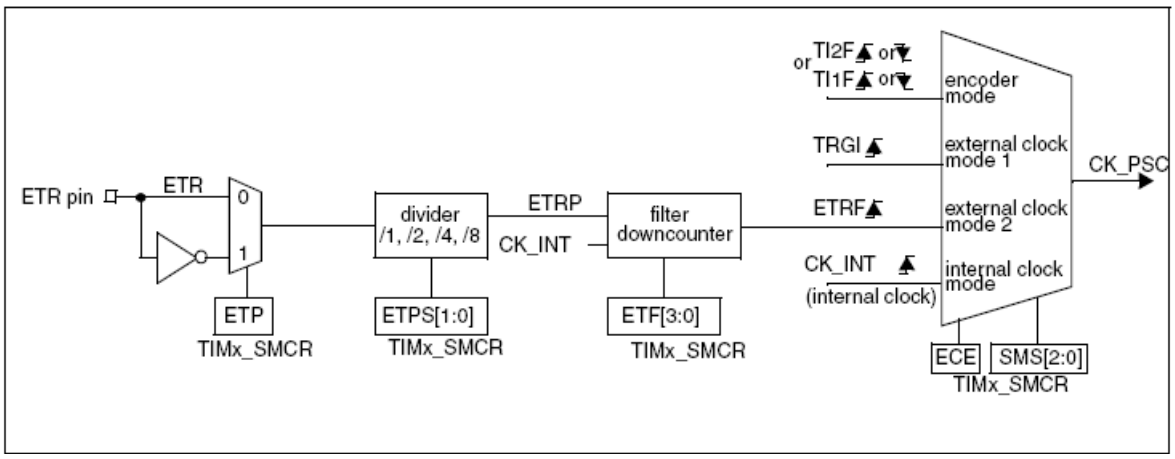
外部时钟源模式2

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。

下图是外部触发输入的总体框图

图117 外部触发输入框图



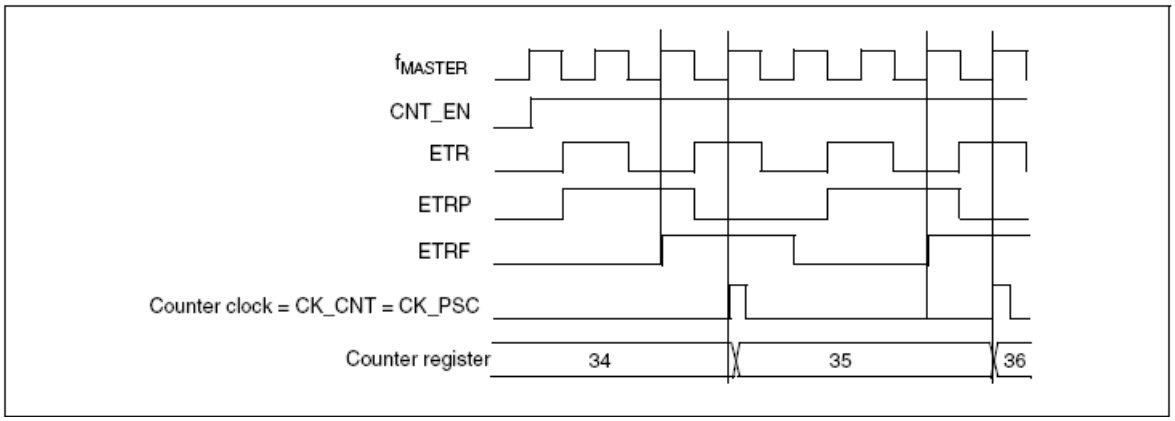
例如，要配置在ETR下每2个上升沿计数一次的向上计数器，使用下列步骤：

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

计数器在每2个ETR上升沿计数一次。

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

图118 外部时钟模式2下的控制电路



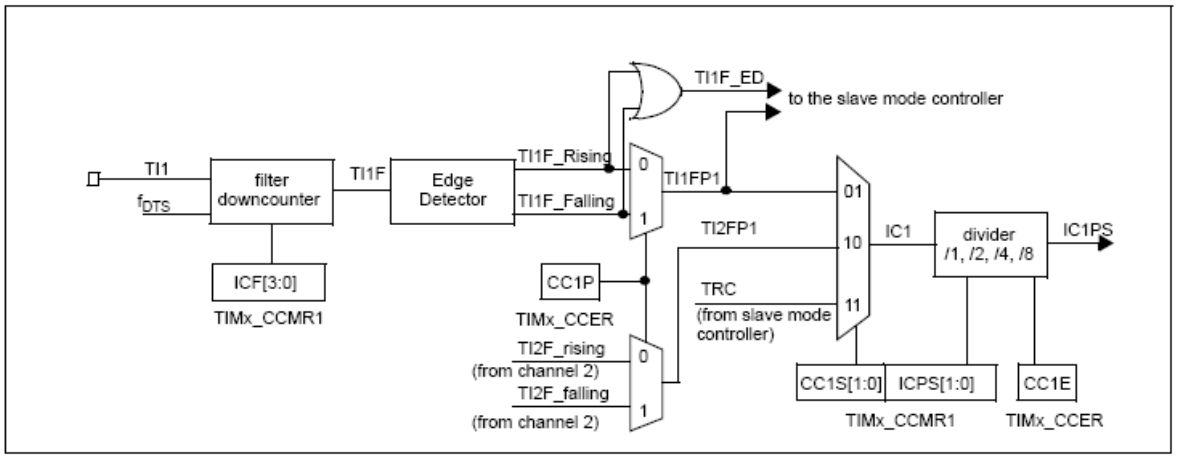
13.3.4 捕获/比较通道

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

下面几张图是一个捕获/比较通道概览。

输入部分对相应的TIx输入信号采样，并产生一个滤波后的信号TIxF。然后，一个带极性选择的边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。该信号通过预分频进入捕获寄存器(ICxPS)。

图119 捕获/比较通道(如：通道1输入部分)



输出部分产生一个中间波形OCxRef(高有效)作为基准，链的末端决定最终输出信号的极性。

图120 捕获/比较通道1的主电路

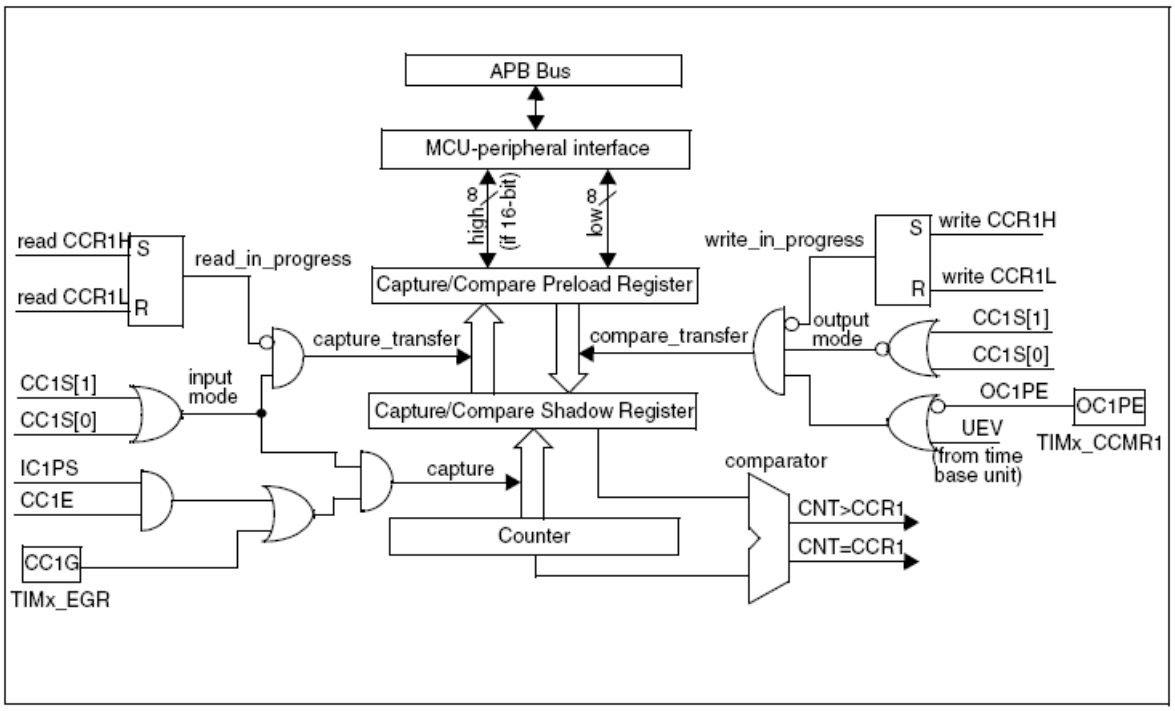
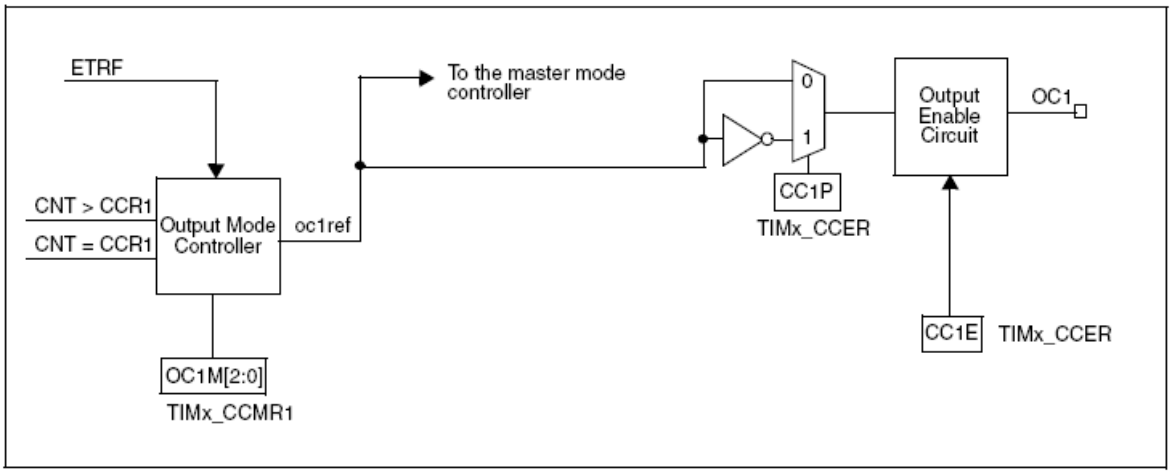


图121 捕获/比较通道的输出部分(通道1)



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

在捕获模式下，捕获发生在影子寄存器上，然后再复制到预装载寄存器中。

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

13.3.5 输入捕获模式

在输入捕获模式下，当检测到ICx信号上相应的边沿后，计数器的当前值被锁存到捕获/比较寄存器(TIMx_CCRx)中。当捕获事件发生时，相应的CCxIF标志(TIMx_SR寄存器)被置1，如果开放了中断或者DMA操作，则将产生中断或者DMA操作。如果捕获事件发生时CCxIF标志已经为高，那么重复捕获标志CCxOF(TIMx_SR寄存器)被置1。写CCxIF=0可清除CCxIF，或读取存储在TIMx_CCRx寄存器中的捕获数据也可清除CCxIF。写CCxOF=0可清除CCxOF。

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

- 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(即输入为TIx时，输入滤波器控制位是TIMx_CCMRx寄存器中的ICxF位)。假设输入信号在最多5个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f_{DTS}频率)连续采样8次，以确认在TI1上一次真实的边沿变换，即在TIMx_CCMR1寄存器中写入IC1F=0011。
- 选择TI1通道的有效转换边沿，在TIMx_CCER寄存器中写入CC1P=0(上升沿)。
- 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。*

13.3.6 PWM输入模式

该模式是输入捕获模式的一个特例，除下列区别外，操作与输入捕获模式相同：

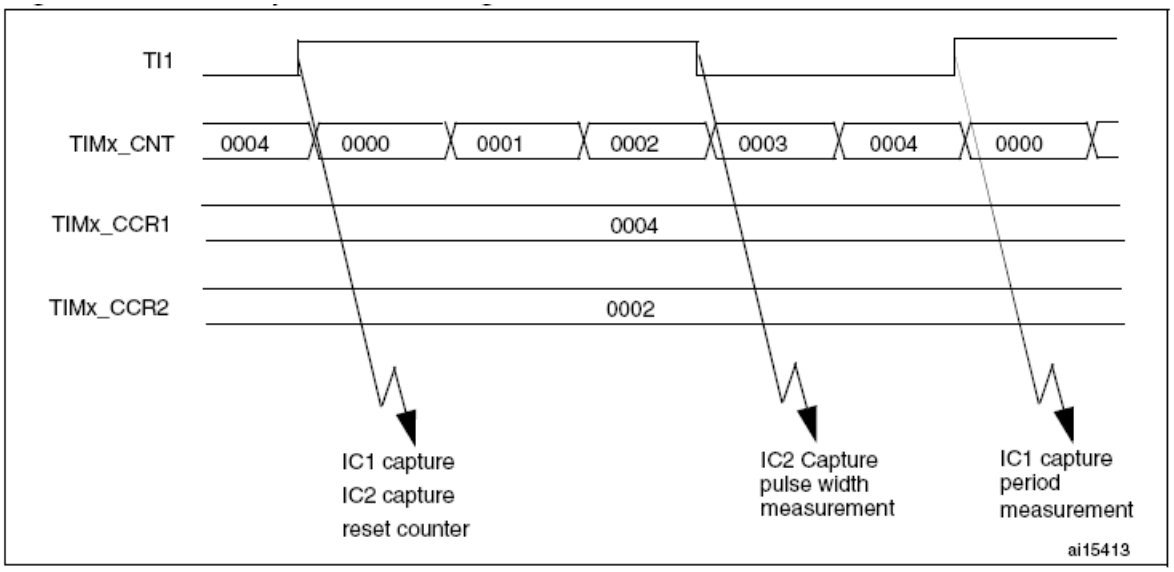
- 两个ICx信号被映射同一个TIx输入。
- 这2个ICx信号为边沿有效，但是极性相反。
- 其中一个TIxFP信号被作为触发输入信号，而从模式控制器被配置成复位模式。

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

- 选择TIMx_CCR1的有效输入：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(上升沿有效)。
- 选择TIMx_CCR2的有效输入：置TIMx_CCMR1寄存器的CC2S=10(选择TI1)。
- 选择TI1FP2的有效极性(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122 PWM输入模式时序



由于只有TI1FP1和TI2FP2连到了从模式控制器，所以PWM输入模式只能使用TIMx_CH1/TIMx_CH2信号。

13.3.7 强置输出模式

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

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

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。

13.3.8 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=000)、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)。
- 若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位选择TIMx_CCRx寄存器是否需要使用预装载寄存器。

在输出比较模式下，更新事件UEV对OCxREF和OCx输出没有影响。

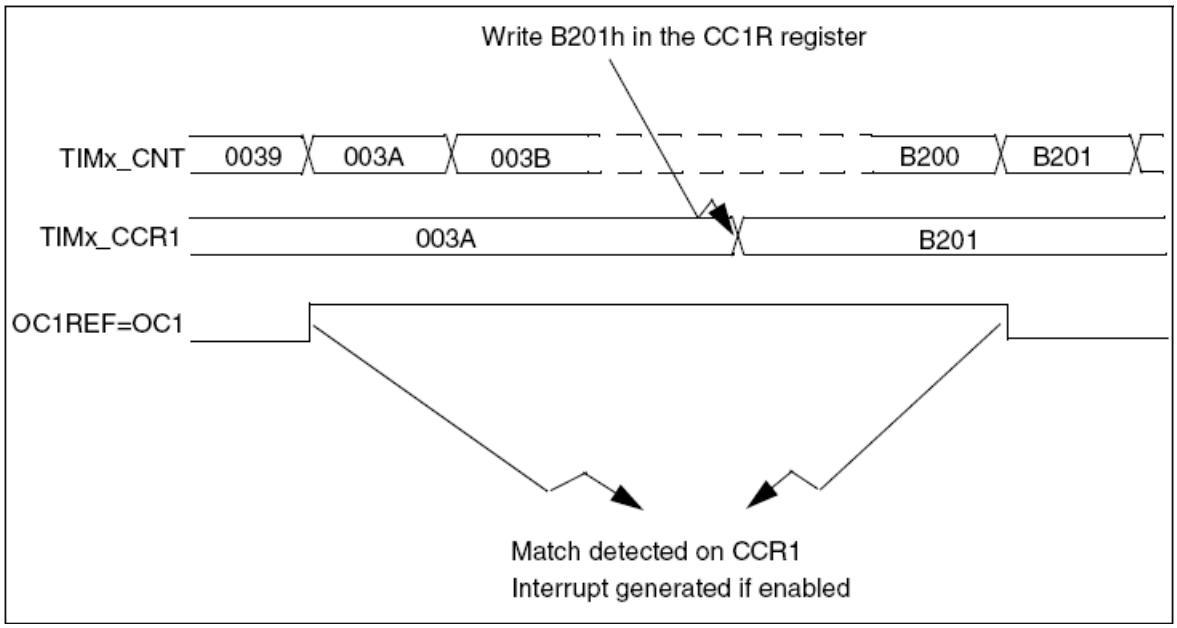
同步的精度可以达到计数器的一个计数周期。输出比较模式(在单脉冲模式下)也能用来输出一个单脉冲。

输出比较模式的配置步骤：

1. 选择计数器时钟(内部，外部，预分频器)
2. 将相应的数据写入TIMx_ARR和TIMx_CCRx寄存器中
3. 如果要产生一个中断请求和/或一个DMA请求，设置CCxIE位和/或CCxDE位。
4. 选择输出模式，例如：必须设置OCxM='011'、OCxPE='0'、CCxP='0'和CCxE='1'，当计数器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影子寄存器只能在发生下一次更新事件时被更新)。下图给出了一个例子。

图123 输出比较模式，翻转OC1



13.3.9 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OCx输出通道产生一路PWM。必须设置TIMx_CCMRx寄存器OCxPE位以使能相应的预装载寄存器，最后还要设置TIMx_CR1寄存器的ARPE位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或低电平有效。TIMx_CCER寄存器中的CCxE位控制OCx输出使能。详见TIMx_CCERx寄存器的描述。

在PWM模式(模式1或模式2)下，TIMx_CNT和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，ETR信号上的一个外部事件能够清除OCxREF)一致，OCxREF信号只能在下述条件下产生：

- 当比较的结果改变，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PWM信号或中央对齐的PWM信号。

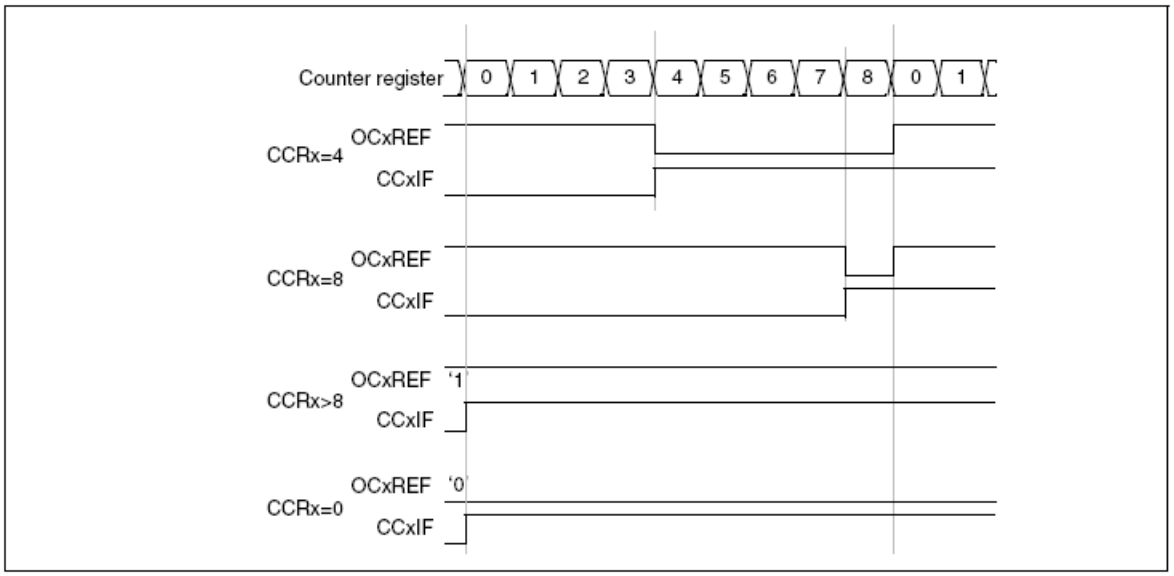
PWM 边沿对齐模式

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13.3.2节。

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

图124 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13.3.2节。

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

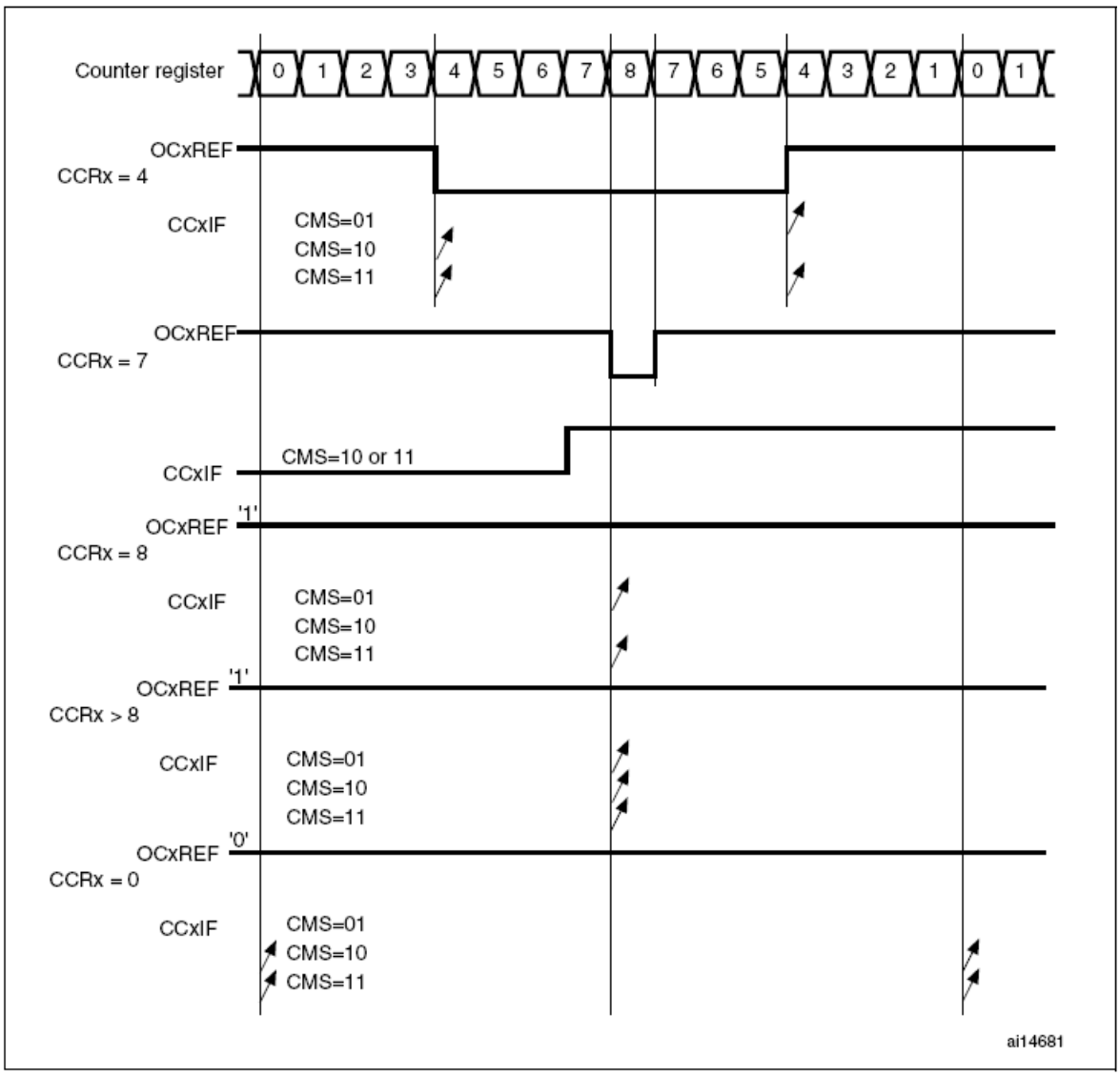
PWM 中央对齐模式

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，比较标志可以在计数器向上计数时被置1、在计数器向下计数时被置1、或在计数器向上和向下计数时被置1。TIMx_CR1寄存器中的计数方向位(DIR)由硬件更新，不要用软件修改它。参看13.3.2节的中央对齐模式。

下图给出了一些中央对齐的PWM波形的例子

- TIMx_ARR=8
- PWM模式1
- TIMx_CR1寄存器中的CMS=01，在中央对齐模式1时，当计数器向下计数时设置比较标志。

图125 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，使用当前的上/下计数配置；这就意味着计数器向上还是向下计数取决于TIMx_CR1寄存器中DIR位的当前值。此外，软件不能同时修改DIR和CMS位。
- 不推荐当运行在中央对齐模式时改写计数器，因为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。

13.3.10 单脉冲模式

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

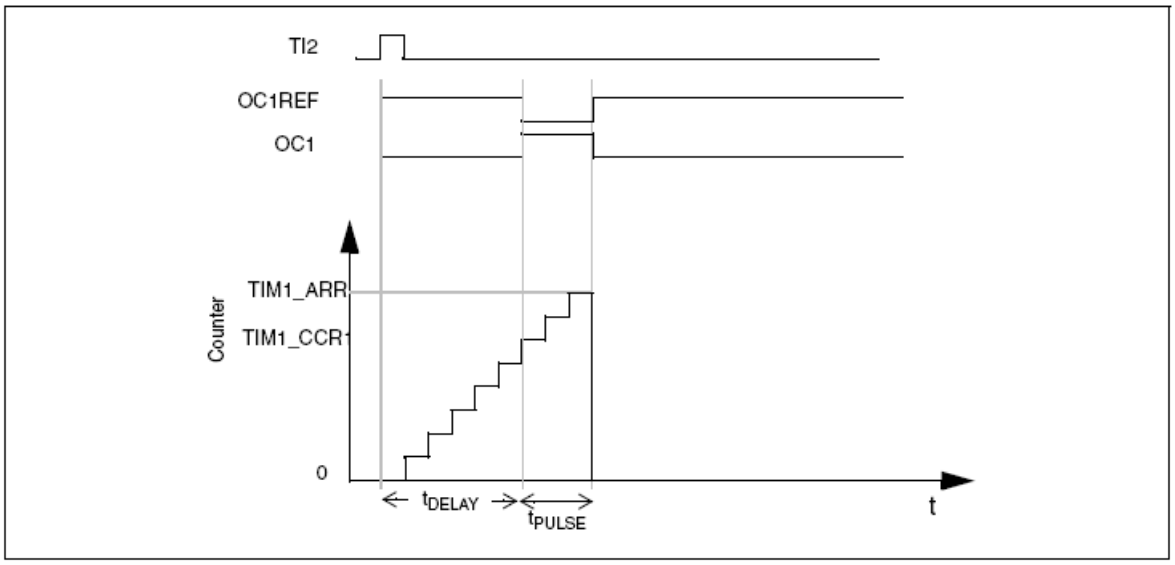
可以通过从模式控制器启动计数器，在输出比较模式或者PWM模式下产生波形。设置TIMx_CR1寄存器中的OPM位将选择单脉冲模式，这样可以让计数器自动地在产生下一个更新事件UEV时停止。

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

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

向下计数方式：CNT > CCRx。

图126 单脉冲模式的例子



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

假定TI2FP2作为触发1:

- 置TIMx_CCMR1寄存器中的CC2S=01，把TI2FP2映像到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，TI2FP2作为从模式控制器的触发(TRGI)。
- 置TIMx_SMCR寄存器中的SMS=110(触发模式)，TI2FP2被用来启动计数器。

OPM波形由写入比较寄存器的数值决定(要考虑时钟频率和计数器预分频器)

- t_{DELAY}由写入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寄存器的OC1M=111，进入PWM模式2；根据需要有选择地使能预装载寄存器：置TIMx_CCMR1中的OC1PE=1和TIMx_CR1寄存器中的ARPE；然后在TIMx_CCR1寄存器中填写比较值，在TIMx_ARR寄存器中填写自动装载值，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，所以必须设置TIMx_CR1寄存器中的OPM=1，在下一个更新事件(当计数器从自动装载值翻转到0)时停止计数。

特殊情况：OCx快速使能：

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

如果要以最小延时输出波形，可以设置TIMx_CCMRx寄存器中的OCxFE位；此时强制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。

13.3.11 在外部事件时清除OCxREF信号

对于一个给定的通道，在ETRF输入端设置TIMx_CCMRx寄存器中对应的OCxCE位为'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。

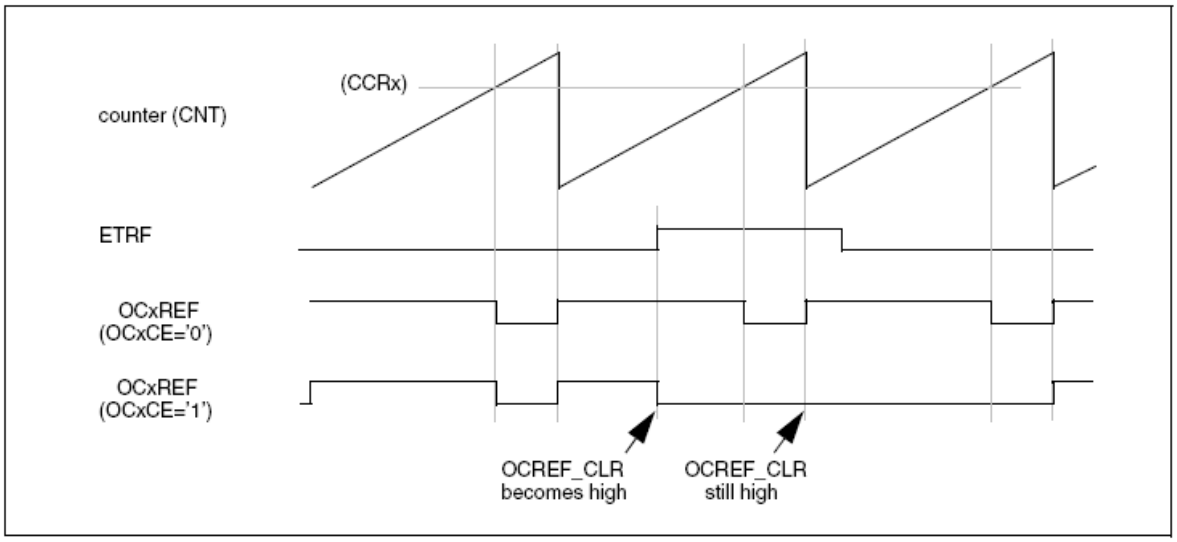
该功能只能用于输出比较和PWM模式，而不能用于强置模式。

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

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

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

图127 清除TIMx的OCxREF



13.3.12 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58，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，同时硬件对TIMx_CR1寄存器的DIR位进行相应的设置。不管计数器是依靠TI1计数、依靠TI2计数或者同时依靠TI1和TI2计数。在任一输入端(TI1或者TI2)的跳变都会重新计算DIR位。

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

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

表58 计数方向与编码器信号的关系

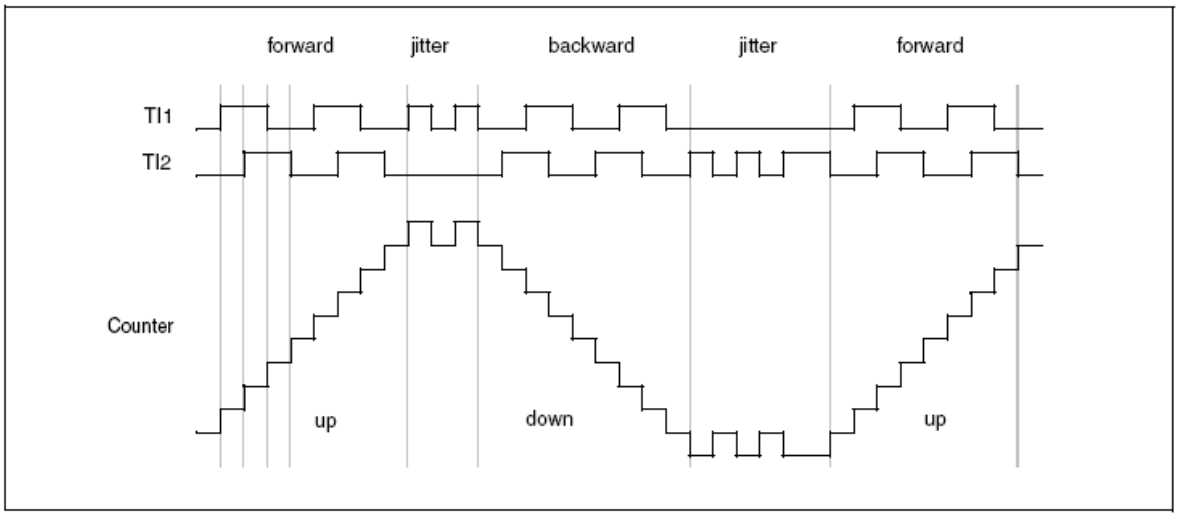
有效边沿	相对信号的电平 (TI1FP1对应TI2, TI2FP2对应TI1)	TI1FP1信号		TI2FP2信号	
		上升	下降	上升	下降
仅在TI1计数	高	向下计数	向上计数	不计数	不计数
	低	向上计数	向下计数	不计数	不计数
仅在TI2计数	高	不计数	不计数	向上计数	向下计数
	低	不计数	不计数	向下计数	向上计数
在TI1和TI2上计数	高	向下计数	向上计数	向上计数	向下计数
	低	向上计数	向下计数	向下计数	向上计数

一个外部的增量编码器可以直接与MCU连接而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，可以把它连接到一个外部中断输入并触发一个计数器复位。

下图是一个计数器操作的实例，显示了计数信号的产生和方向控制。它还显示了当选择了双边沿时，输入抖动是如何被抑制的；抖动可能会在传感器的位置靠近一个转换点时产生。在这个例子中，我们假定配置如下：

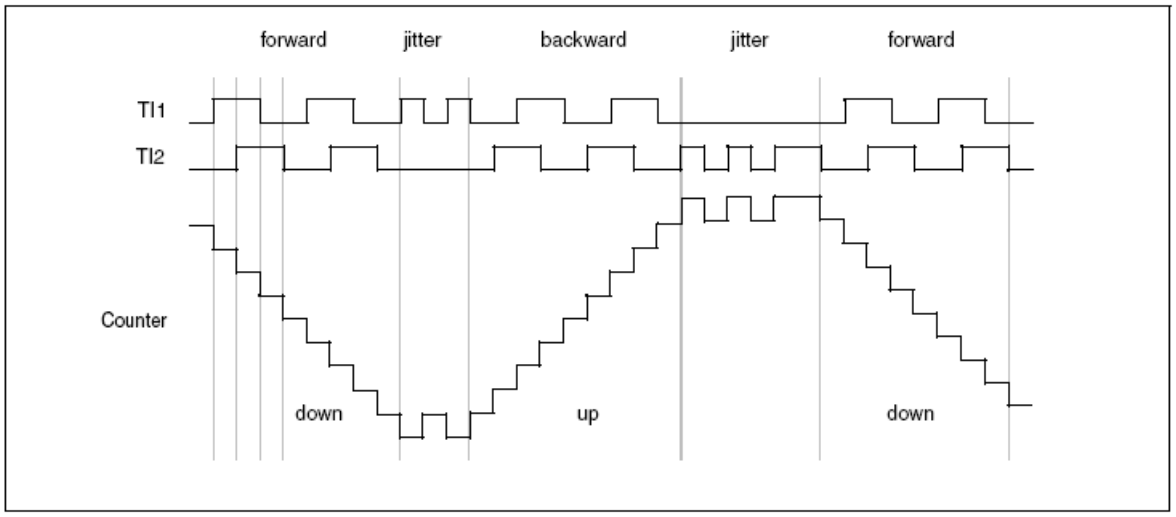
- CC1S=’01’ (TIMx_CCMR1寄存器，IC1FP1映射到TI1)
- CC2S=’01’ (TIMx_CCMR2寄存器，IC2FP2映射到TI2)
- CC1P=’0’ (TIMx_CCER寄存器，IC1FP1不反相，IC1FP1=TI1)
- CC2P=’0’ (TIMx_CCER寄存器，IC2FP2不反相，IC2FP2=TI2)
- SMS=’011’ (TIMx_SMCR寄存器，所有的输入均在上升沿和下降沿有效).
- CEN=’1’ (TIMx_CR1寄存器，计数器使能)

图128 编码器模式下的计数器操作实例



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

图129 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。

13.3.13 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。

异或输出能够被用于所有定时器的输入功能，如触发或输入捕获。上一章12.3.18节给出了此特性用于连接霍尔传感器的例子。

13.3.14 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。

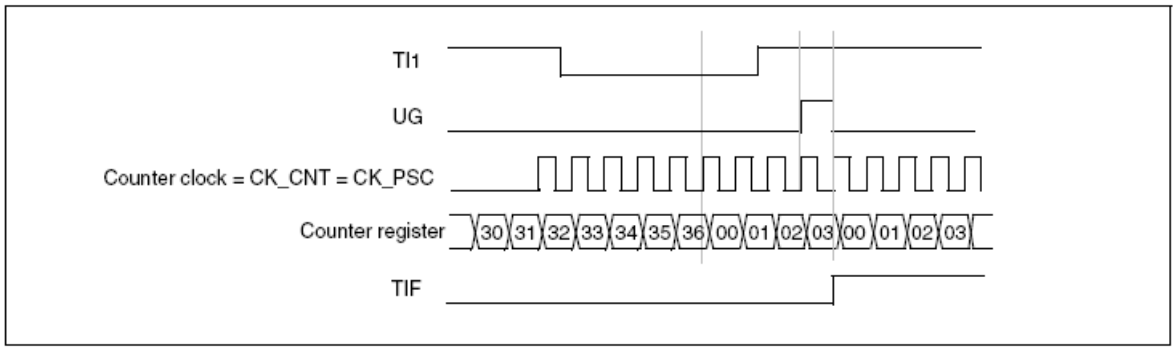
在以下的例子中，TI1输入端的上升沿导致向上计数器被清零：

- 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130 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。

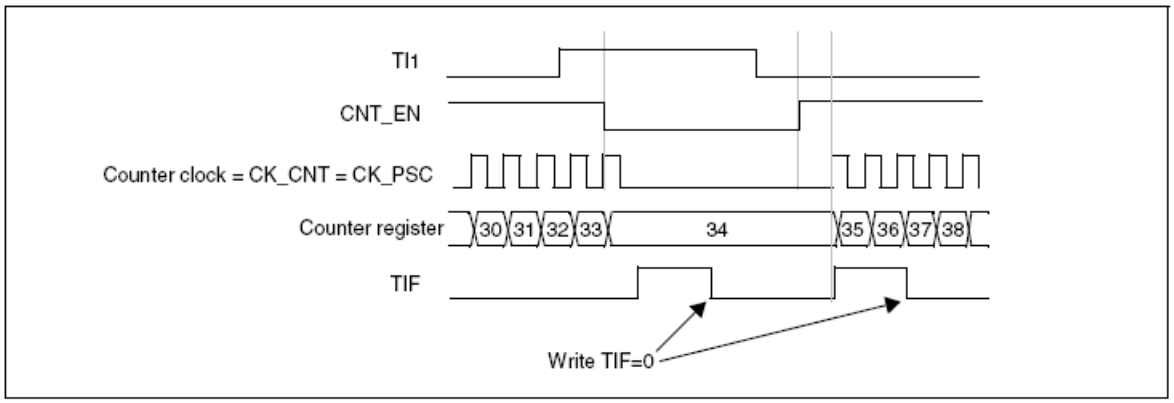
在如下的例子中，计数器只在TI1为低时向上计数：

- 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131 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。

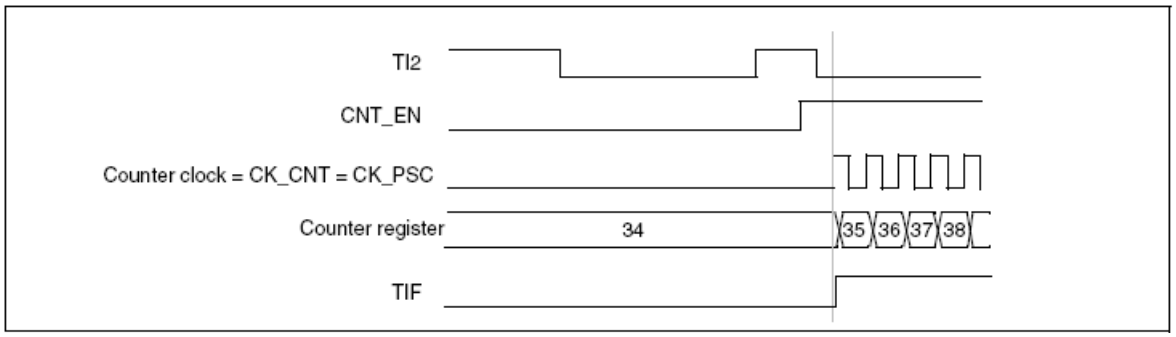
在下面的例子中，计数器在TI2输入的上升沿开始向上计数：

- 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IC2F=0000)。触发操作中不使用捕获预分频器，不需要配置。CC2S位只用于选择输入捕获源，置TIMx_CCMR1寄存器中CC2S=01。置TIMx_CCER寄存器中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出现一个上升沿时，计数器开始在内部时钟驱动下计数，同时设置TIF标志。

TI2上升沿和计数器启动计数之间的延时取决于TI2输入端的重同步电路。

图132 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、门控模式或触发模式时可以选择另一个输入作为触发输入。不建议使用TIMx_SMCR寄存器的TS位选择ETR作为TRGI。

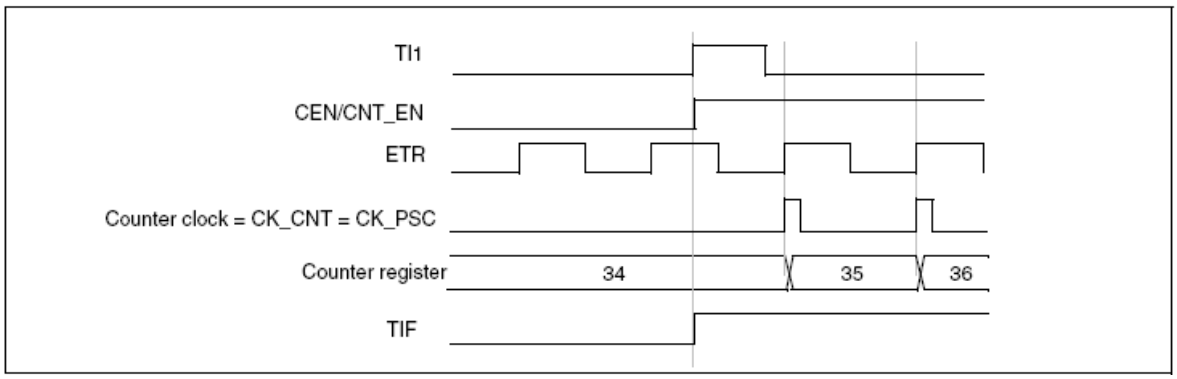
在下面的例子中，一旦在TI1上出现一个上升沿，计数器即在ETR的每一个上升沿向上计数一次：

1. 通过TIMx_SMCR寄存器配置外部触发输入电路：
   - ETF=0000：没有滤波
   - ETPS=00：不用预分频器
   - ETP=0：检测 ETR 的上升沿，置 ECE=1 使能外部时钟模式 2
2. 按如下配置通道1，检测TI的上升沿：
   - IC1F=0000：没有滤波
   - 触发操作中不使用捕获预分频器，不需要配置
   - 置 TIMx_CCMR1 寄存器中 CC1S=01，选择输入捕获源
   - 置 TIMx_CCER 寄存器中 CC1P=0 以确定极性(只检测上升沿)
3. 置TIMx_SMCR寄存器中SMS=110，配置定时器为触发模式。置TIMx_SMCR寄存器中TS=101，选择TI1作为输入源。

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

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

图133 外部时钟模式2+触发模式下的控制电路



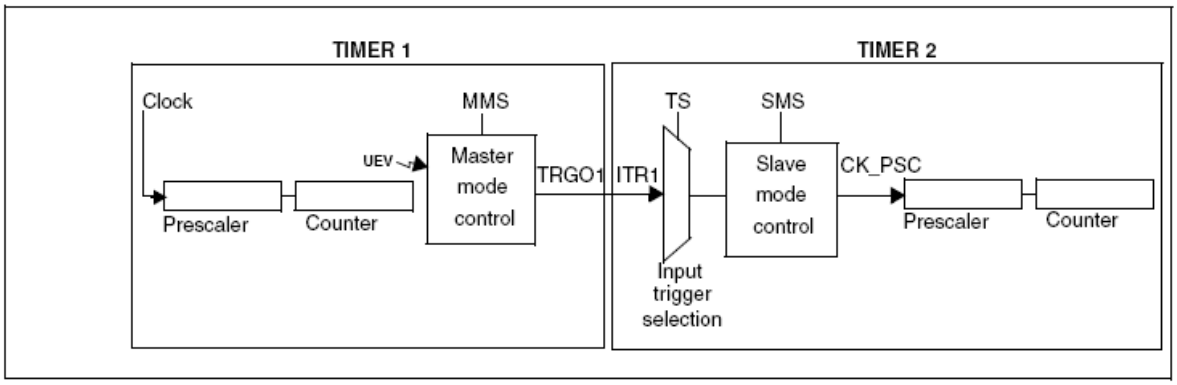
13.3.15 定时器同步

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定时器的预分频器

图134 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134，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时输出一个周期性的触发信号。在TIM1_CR2寄存器的MMS='010'时，每当产生一个更新事件时在TRGO1上输出一个上升沿信号。
- 连接定时器1的TRGO1输出至定时器2，设置TIM2_SMCR寄存器的TS='000'，配置定时器2为使用ITR1作为内部触发的从模式。
- 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周期性的上升沿(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。*

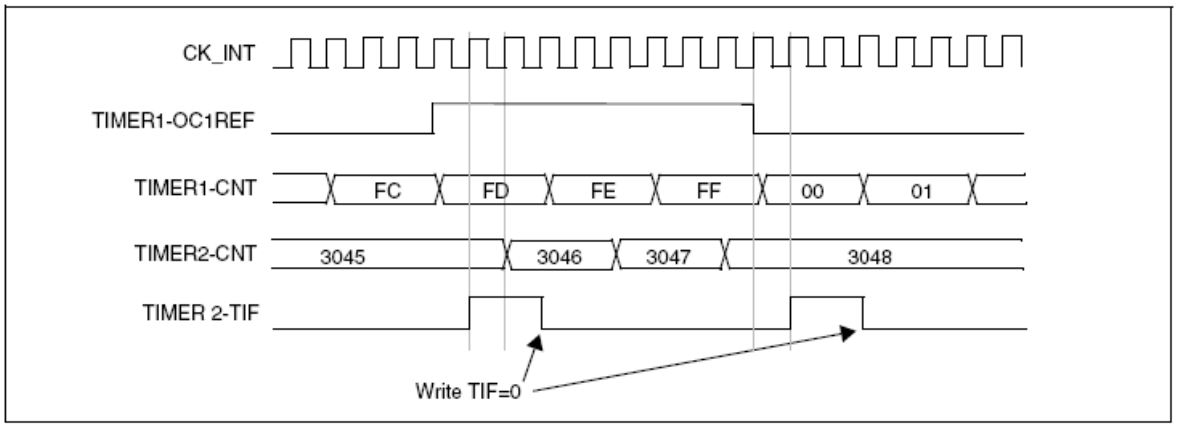
使用一个定时器使能另一个定时器

在这个例子中，定时器2的运行受由定时器1的输出比较控制。参考图134的连接。只当定时器1的OC1REF为高时定时器2才对分频后的内部时钟计数。两个定时器的时钟频率都是由预分频器对CK_INT除以3($f_{CK\_CNT}=f_{CK\_INT}/3$)得到。

- 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计数器的使能信号。*

图135 定时器1的OC1REF控制定时器2

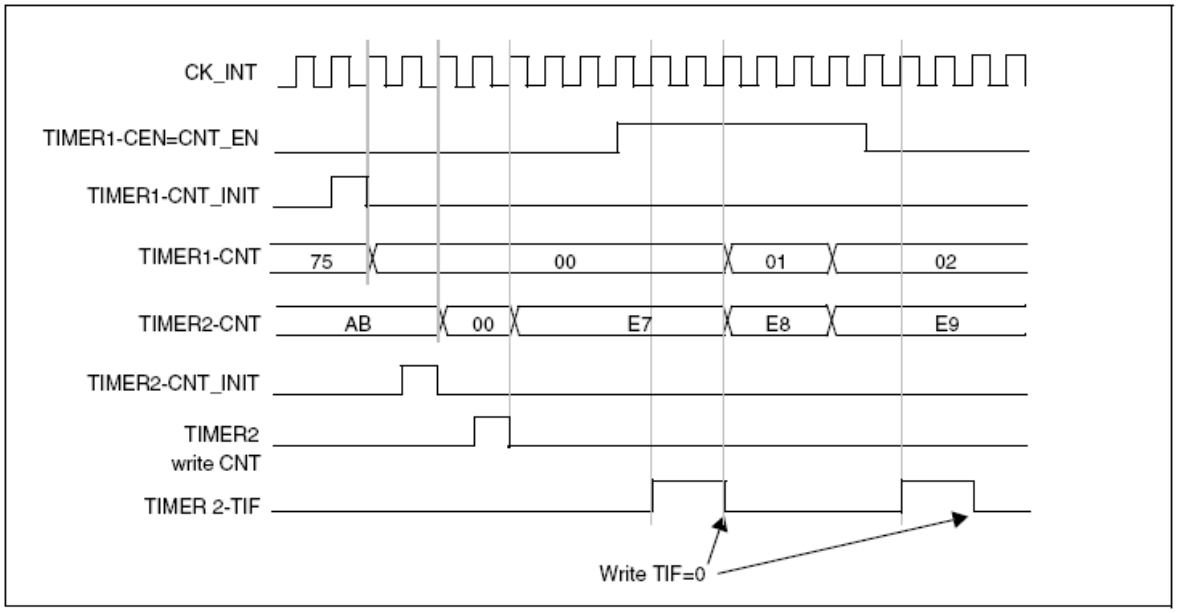


在图135的例子中，在定时器2启动之前，它们的计数器和预分频器未被初始化，因此它们从当前的数值开始计数。可以在启动定时器1之前复位2个定时器，使它们从给定的数值开始，即在定时器计数器中写入需要的任意数值。写TIMx_EGR寄存器的UG位即可复位定时器。

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=000)
- 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2。
- 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136 通过使能定时器1可以控制定时器2

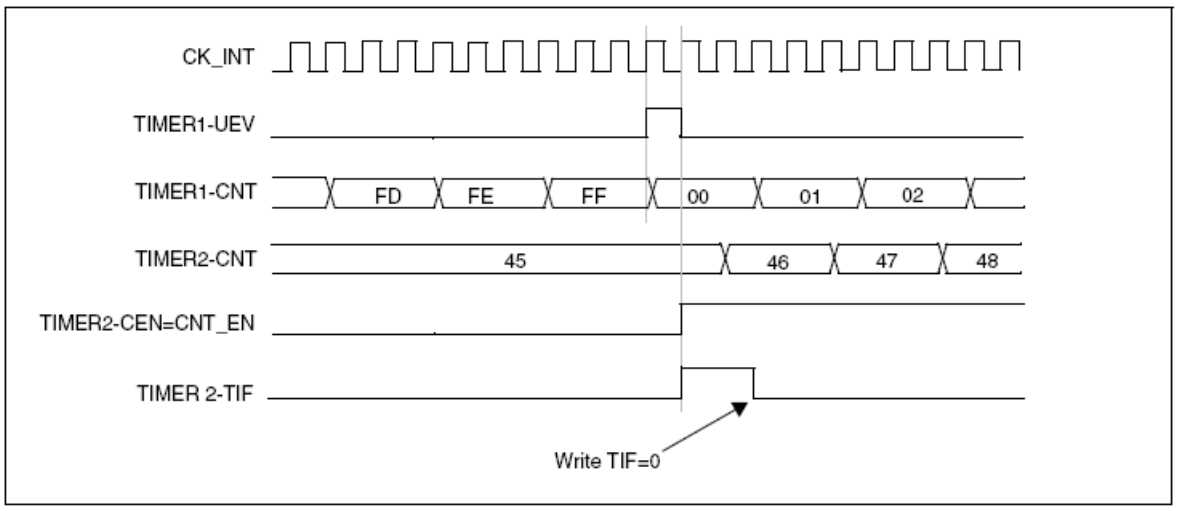


使用一个定时器去启动另一个定时器

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134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)按照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($f_{CK\_CNT}=f_{CK\_INT}/3$)。

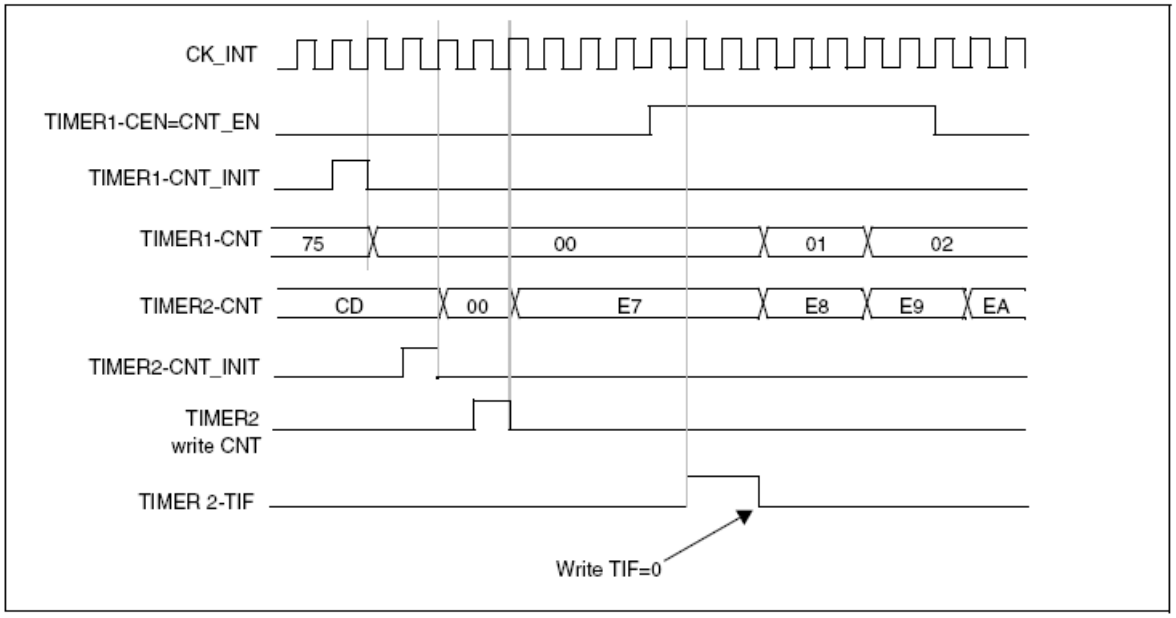
- 配置定时器1为主模式，送出它的更新事件(UEV)做为触发输出(TIM1_CR2寄存器的MMS=010)。
- 配置定时器1的周期(TIM1_ARR寄存器)。
- 配置定时器2从定时器1获得输入触发(TIM2_SMCR寄存器的TS=000)
- 配置定时器2为触发模式(TIM2_SMCR寄存器的SMS=110)
- 置TIM1_CR1寄存器的CEN=1以启动定时器1。

图137 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138显示在与图137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138 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

这个例子使用定时器1作为定时器2的预分频器。参考图134的连接，配置如下：

- 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')。然后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=000)
- 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

这个例子中当定时器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2，参见图134。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)：

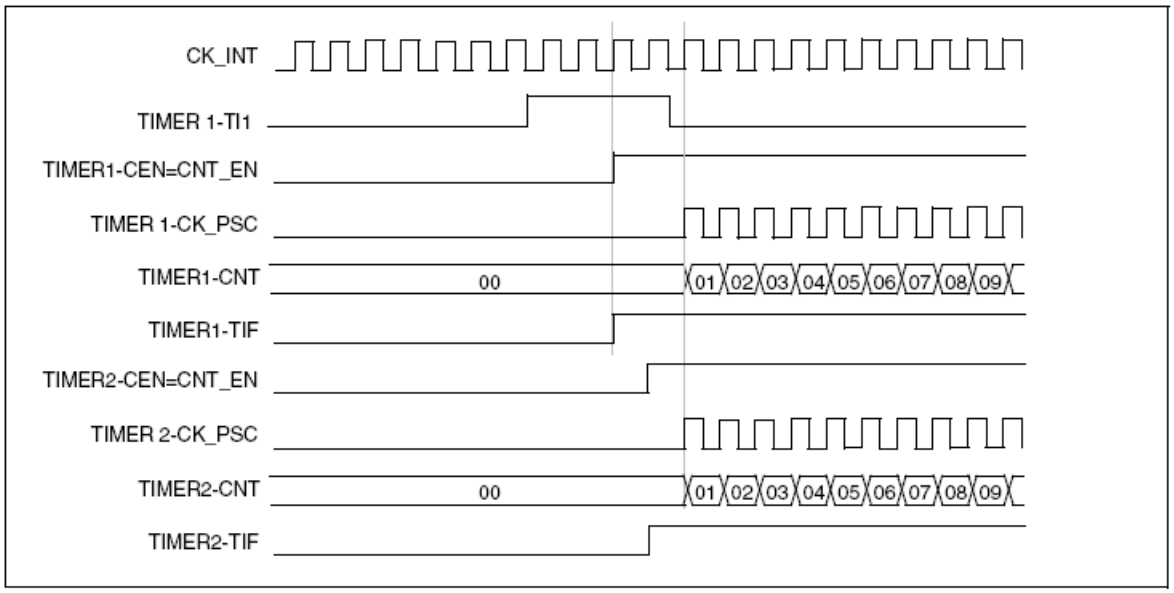
- 配置定时器1为主模式，送出它的使能做为触发输出(TIM1_CR2寄存器的MMS='001')。

- 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=000)
- 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，两个定时器同步地按照内部时钟开始计数，两个TIF标志也同时被设置。

注：在这个例子中，在启动之前两个定时器都被初始化(设置相应的UG位)，两个计数器都从0开始，但可以通过写入任意一个计数器寄存器(TIMx_CNT)在定时器间插入一个偏移。下图中能看到主/从模式下在定时器1的CNT_EN和CK_PSC之间有个延迟。

图139 使用定时器1的TI1输入触发定时器1和定时器2



13.3.16 调试模式

当微控制器进入调试模式(Cortex-M3核心停止)，根据DBG模块中DBG_TIMx_STOP的设置，TIMx计数器或者继续正常操作，或者停止。详见随后调试模块章节。

13.4 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。

13.4.1 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 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/下溢才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**：使能计数器 0：禁止计数器； 1：使能计数器。 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。

13.4.2 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12.3.18的与霍尔传感器的接口一节。
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(从模式控制器处于复位模式)产生复位，则TRGO上的信号相对实际的复位会有一个延迟。 001：**使能** – 计数器使能信号CNT_EN被用于作为触发输出(TRGO)。有时需要在同一时间启动多个定时器或控制在一段时间内使能从定时器。计数器使能信号是通过CEN控制位和门控模式下的触发输入信号的逻辑或产生。当计数器使能信号受控于触发输入时，TRGO上会有一个延迟，除非选择了主/从模式(见TIMx_SMCR寄存器中MSM位的描述)。 010：**更新** – 更新事件被选为触发输入(TRGO)。例如，一个主定时器的时钟可以被用作一个从定时器的预分频器。 011：**比较脉冲** – 一旦发生一次捕获或一次比较成功时，当要设置CC1IF标志时(即使它已经为高)，触发输出送出一个正脉冲(TRGO)。 100：**比较** – OC1REF信号被用于作为触发输出(TRGO)。 101：**比较** – OC2REF信号被用于作为触发输出(TRGO)。 110：**比较** – OC3REF信号被用于作为触发输出(TRGO)。 111：**比较** – OC4REF信号被用于作为触发输出(TRGO)。
位3	**CCDS**：捕获/比较的DMA选择 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。

13.4.3 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**：外部时钟使能位 该位启用外部时钟模式2 0：禁止外部时钟模式2； 1：使能外部时钟模式2。计数器由ETRF信号上的任意有效上升沿驱动。 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)。 注3：外部时钟模式1和外部时钟模式2同时被使能时，外部时钟的输入是ETRF。
位13:12	**ETPS[1:0]**：外部触发预分频 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0(ITR0)，TIM1 001：内部触发1(ITR1)，TIM2 010：内部触发2(ITR2)，TIM3 011：内部触发3(ITR3)，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) 关于每个定时器中ITRx的细节，参见表59。 注：这些位只能在未用到(如SMS=000)时被改变，以避免在改变时产生错误的边沿检测。
位3	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 – 根据另一个输入的电平，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。

表59 TIMx内部触发连接

从定时器	ITR0 (TS = 000)	ITR1 (TS = 001)	ITR2 (TS = 010)	ITR3 (TS = 011)
TIM2	TIM1	TIM8	TIM3	TIM4
TIM3	TIM1	TIM2	TIM5	TIM4
TIM4	TIM1	TIM2	TIM3	TIM8
TIM5	TIM2	TIM3	TIM4	TIM8

13.4.4 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**：触发中断使能 0：禁止触发中断； 1：使能触发中断。
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。

13.4.5 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_w0	rc_w0	rc_w0	rc_w0			rc_w0		rc_w0	rc_w0	rc_w0	rc_w0	rc_w0

位15:13	保留，始终读为0。
位12	**CC4OF**：捕获/比较4重复捕获标记 参见CC1OF描述。
位11	**CC3OF**：捕获/比较3重复捕获标记 参见CC1OF描述。
位10	**CC2OF**：捕获/比较2重复捕获标记 参见CC1OF描述。

位9	**CC1OF**：捕获/比较1重复捕获标记 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：无重复捕获产生； 1：计数器的值被捕获到TIMx_CCR1寄存器时，CC1IF的状态已经为1。
位8:7	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：计数器值已被捕获(拷贝)至TIMx_CCR1(在IC1上检测到与所选极性相同的边沿)。
位0	**UIF**：更新中断标记 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(软件对计数器CNT重新初始化)； – 若TIMx_CR1寄存器的UDIS=0、URS=0，当计数器CNT被触发事件重初始化时产生更新事件。（参考同步控制寄存器的说明）

13.4.6 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TIMx_ARR的值。

13.4.7 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3位定义了输出参考信号OC1REF的动作，而OC1REF决定了OC1的值。OC1REF是高电平有效，而OC1的有效电平取决于CC1P位。 000：冻结。输出比较寄存器TIMx_CCR1与计数器TIMx_CNT间的比较对OC1REF不起作用； 001：匹配时设置通道1为有效电平。当计数器TIMx_CNT的值与捕获/比较寄存器1(TIMx_CCR1)相同时，强制OC1REF为高。 010：匹配时设置通道1为无效电平。当计数器TIMx_CNT的值与捕获/比较寄存器1(TIMx_CCR1)相同时，强制OC1REF为低。 011：翻转。当TIMx_CCR1=TIMx_CNT时，翻转OC1REF的电平。 100：强制为无效电平。强制OC1REF为低。 101：强制为有效电平。强制OC1REF为高。 110：PWM模式1－ 在向上计数时，一旦TIMx_CNT<TIMx_CCR1时通道1为有效电平，否则为无效电平；在向下计数时，一旦TIMx_CNT>TIMx_CCR1时通道1为无效电平(OC1REF=0)，否则为有效电平(OC1REF=1)。 111：PWM模式2－ 在向上计数时，一旦TIMx_CNT<TIMx_CCR1时通道1为无效电平，否则为有效电平；在向下计数时，一旦TIMx_CNT>TIMx_CCR1时通道1为有效电平，否则为无效电平。 注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，并且新写入的数值立即起作用。 1：开启TIMx_CCR1寄存器的预装载功能，读写操作仅对预装载寄存器操作，TIMx_CCR1的预装载值在更新事件到来时被载入当前寄存器中。 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(TIMx_CR1寄存器的OPM=1)，可以在未确认预装载寄存器情况下使用PWM模式，否则其动作不确定。
位2	**OC1FE**：输出比较1 快速使能 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[3:0]**：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f_{DTS}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。

13.4.8 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	**OC3M[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[3:0]**：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，IC3映射在TRC上。此模式仅工作在内部触发器输入被选中时(由TIMx_SMCR寄存器的TS位选择)。 注：CC3S仅在通道关闭时(TIMx_CCER寄存器的CC3E=0)才是可写的。

13.4.9 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60 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。

13.4.10 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

13.4.11 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。

13.4.12 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包含了将要装载入实际的自动重装载寄存器的数值。 详细参考13.3.1节：有关ARR的更新和动作。 当自动重装载的值为空时，计数器不工作。

13.4.13 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包含了装入当前捕获/比较1寄存器的值(预装载值)。 如果在TIMx_CCMR1寄存器(OC1PE位)中未选择预装载特性，写入的数值会立即传输至当前寄存器中。否则只有当更新事件发生时，此预装载值才传输至当前捕获/比较1寄存器中。 当前捕获/比较寄存器参与同计数器TIMx_CNT的比较，并在OC1端口上产生输出信号。 **若CC1通道配置为输入：** CCR1包含了由上一次输入捕获1事件(IC1)传输的计数器值。

13.4.14 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包含了装入当前捕获/比较2寄存器的值(预装载值)。 如果在TIMx_CCMR2寄存器(OC2PE位)中未选择预装载特性，写入的数值会立即传输至当前寄存器中。否则只有当更新事件发生时，此预装载值才传输至当前捕获/比较2寄存器中。 当前捕获/比较寄存器参与同计数器TIMx_CNT的比较，并在OC2端口上产生输出信号。 **若CC2通道配置为输入：** CCR2包含了由上一次输入捕获2事件(IC2)传输的计数器值。

13.4.15 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包含了装入当前捕获/比较3寄存器的值(预装载值)。 如果在TIMx_CCMR3寄存器(OC3PE位)中未选择预装载特性，写入的数值会立即传输至当前寄存器中。否则只有当更新事件发生时，此预装载值才传输至当前捕获/比较3寄存器中。 当前捕获/比较寄存器参与同计数器TIMx_CNT的比较，并在OC3端口上产生输出信号。 **若CC3通道配置为输入：** CCR3包含了由上一次输入捕获3事件(IC3)传输的计数器值。

13.4.16 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包含了装入当前捕获/比较4寄存器的值(预装载值)。 如果在TIMx_CCMR4寄存器(OC4PE位)中未选择预装载特性，写入的数值会立即传输至当前寄存器中。否则只有当更新事件发生时，此预装载值才传输至当前捕获/比较4寄存器中。 当前捕获/比较寄存器参与同计数器TIMx_CNT的比较，并在OC4端口上产生输出信号。 **若CC4通道配置为输入：** CCR4包含了由上一次输入捕获4事件(IC4)传输的计数器值。

13.4.17 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寄存器进行读或写时，定时器则进行一次连续传送)，即：定义传输的次数，传输可以是半字(双字节)或字节： 00000：1次传输　　00001：2次传输 00010：3次传输　　..... .....　　10001：18次传输
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， .....

13.4.18 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所在寄存器的存取操作： TIMx_CR1地址 + DBA + DMA索引，其中： “TIMx_CR1地址”是控制寄存器1(TIMx_CR1)所在的地址； “DBA”是TIMx_DCR寄存器中定义的基地址； “DMA索引”是由DMA自动控制的偏移量，它取决于TIMx_DCR寄存器中定义的DBL。

13.4.19 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61 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

有关寄存器的起始地址，参见表1。

14 基本定时器(TIM6和TIM7)

小容量产品是指闪存存储器容量在16K至32K字节之间的STM32F101xx、STM32F102xx和STM32F103xx微控制器。

中容量产品是指闪存存储器容量在64K至128K字节之间的STM32F101xx、STM32F102xx和STM32F103xx微控制器。

大容量产品是指闪存存储器容量在256K至512K字节之间的STM32F101xx和STM32F103xx微控制器。

本章节描述的模块仅适用于大容量的STM32F101xx和STM32F103xx系列。

14.1 TIM6和TIM7简介

基本定时器TIM6和TIM7各包含一个16位自动装载计数器，由各自的可编程预分频器驱动。

它们可以作为通用定时器提供时间基准，特别地可以为数模转换器(DAC)提供时钟。实际上，它们在芯片内部直接连接到DAC并通过触发输出直接驱动DAC。

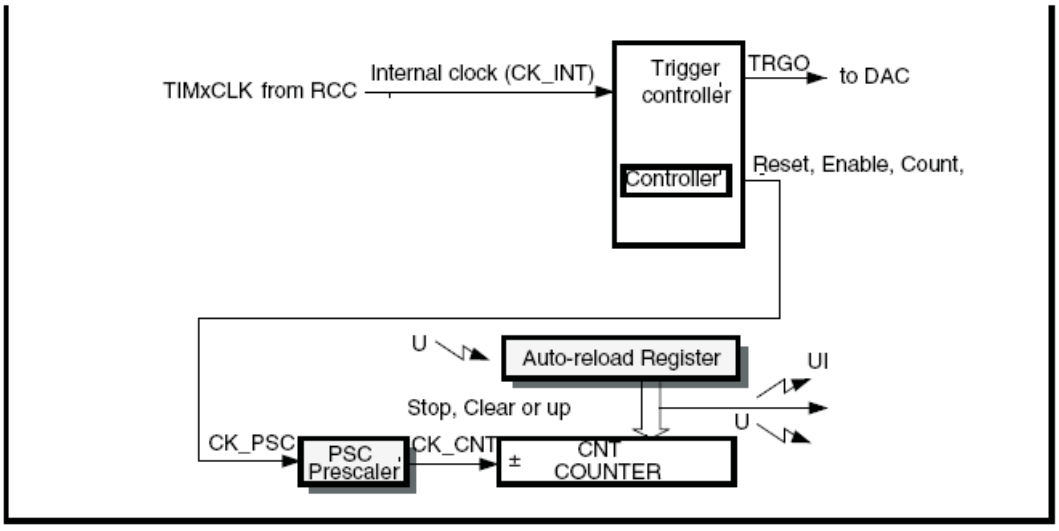
这2个定时器是互相独立的，不共享任何资源。

14.2 TIM6和TIM7的主要特性

TIM6和TIM7定时器的主要功能包括：

- 16位自动装载累加计数器
- 16位可编程(可实时修改)预分频器，用于对输入的时钟按系数为1～65535之间的任意数值分频
- 触发DAC的同步电路
- 在更新事件(计数器溢出)时产生中断/DMA请求

图140 基本定时器框图



Flag 根据控制位的设定，在U事件时传送预转载寄存器至实际寄存器

事件

中断和DMA输出

14.3 TIM6和TIM7的功能

14.3.1 时基单元

这个可编程的定时器的主要部分是一个带有自动重装载的16位累加计数器，计数器的时钟通过一个预分频器得到。

软件可以读写计数器、自动重装载寄存器和预分频寄存器，即使计数器运行时也可以操作。

时基单元包含：

- 计数器寄存器(TIMx_CNT)
- 预分频寄存器(TIMx_PSC)
- 自动重装载寄存器(TIMx_ARR)

自动重装载寄存器是预加载的，每次读写自动重装载寄存器时，实际上是通过读写预加载寄存器实现。根据TIMx_CR1寄存器中的自动重装载预加载使能位(ARPE)，写入预加载寄存器的内容能够立即或在每次更新事件时，传送到它的影子寄存器。当TIMx_CR1寄存器的UDIS位为0，则每当计数器达到溢出值时，硬件发出更新事件；软件也可以产生更新事件；关于更新事件的产生，随后会有详细的介绍。

计数器由预分频输出CK_CNT驱动，设置TIMx_CR1寄存器中的计数器使能位(CEN)使能计数器计数。

注意：实际的设置计数器使能信号CNT_EN相对于CEN滞后一个时钟周期。

预分频器

预分频可以以系数介于1至65536之间的任意数值对计数器时钟分频。它是通过一个16位寄存器(TIMx_PSC)的计数实现分频，因为TIMx_PSC控制寄存器具有缓冲，可以在运行过程中改变它的数值，新的预分频数值将在下一个更新事件时起作用。

以下2图是在运行过程中改变预分频系数的例子。

图141 预分频系数从1变到2的计数器时序图

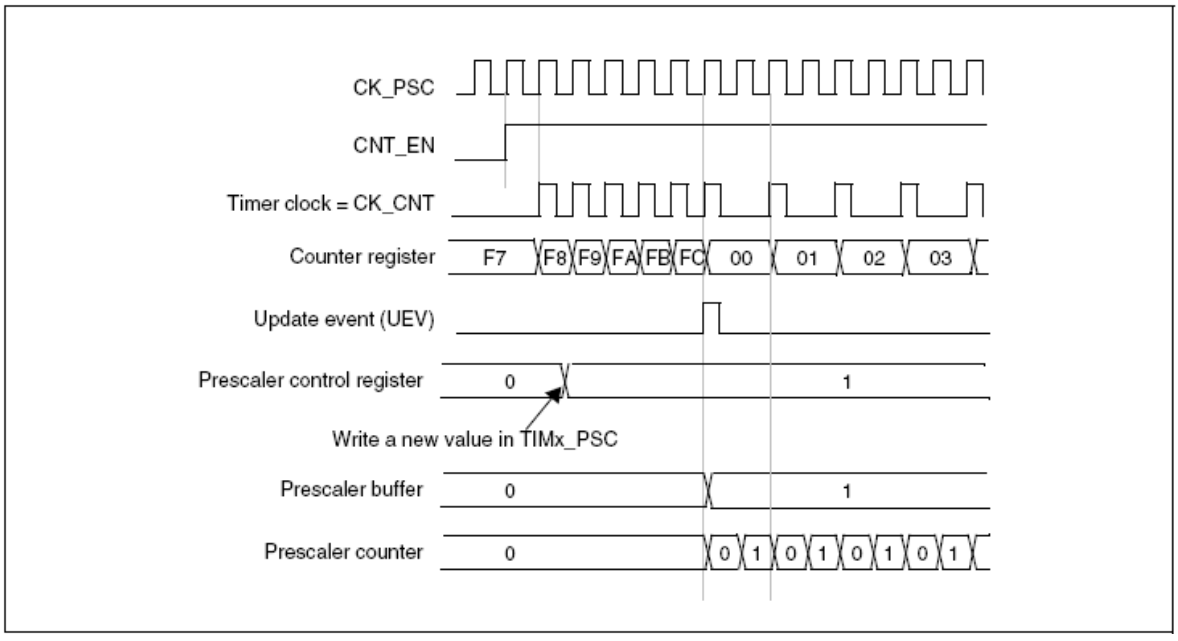
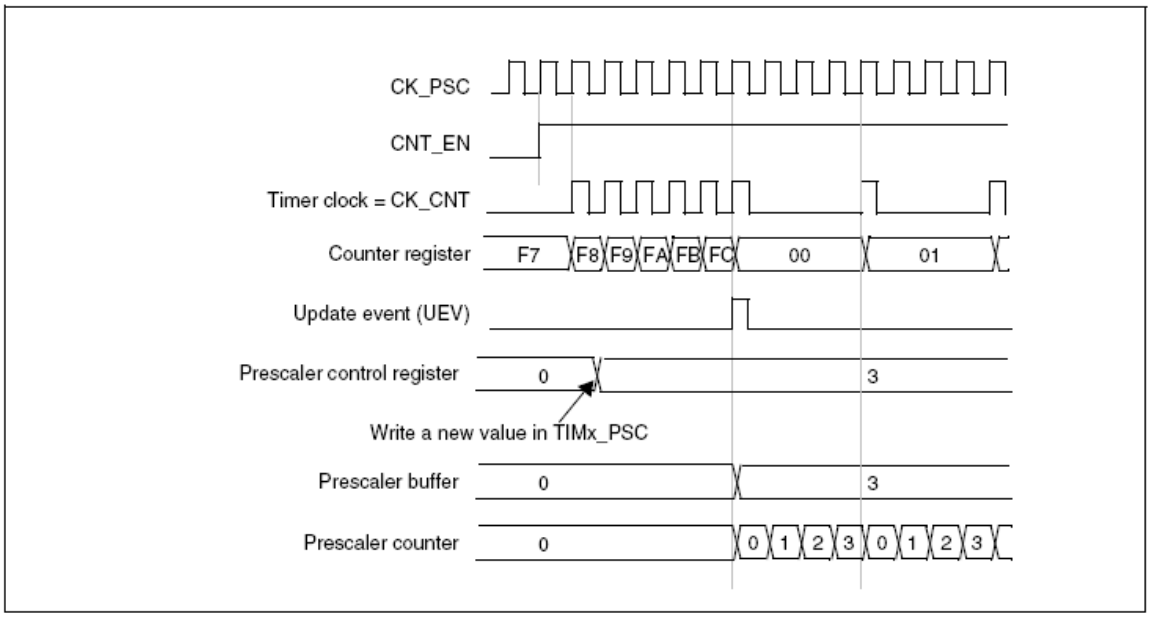


图142 预分频系数从1变到4的计数器时序图



14.3.2 计数模式

计数器从累加计数到自动重装载数值(TIMx_ARR寄存器)，然后重新从0开始计数并产生一个计数器溢出事件。

这每次计数器溢出时可以产生更新事件，(通过软件或使用从模式控制器)设置TIMx_EGR寄存器的UG位也可以产生更新事件。

设置TIMx_CR1中的UDIS位可以禁止产生UEV事件，这可以避免在写入预加载寄存器时更改影子寄存器。在清除UDIS位为0之前，将不再产生更新事件，但计数器和预分频器依然会在应产生更新事件时重新从0开始计数(但预分频系数不变)。另外，如果设置了TIMx_CR1寄存器中的URS(选择更新请求)，设置UG位可以产生一次更新事件UEV，但不设置UIF标志(即没有中断或DMA请求)。

当发生一次更新事件，所有寄存器会被更新并(根据URS位)设置更新标志(TIMx_SR寄存器的UIF位)：

- 传送预装载值(TIMx_PSC寄存器的内容)至预分频器的缓冲区。
- 自动重装载影子寄存器被更新为预装载值(TIMx_ARR)。

以下是一些在TIMx_ARR=0x36时不同时钟频率下计数器工作的图示例子。

图143 计数器时序图，内部时钟分频系数为1

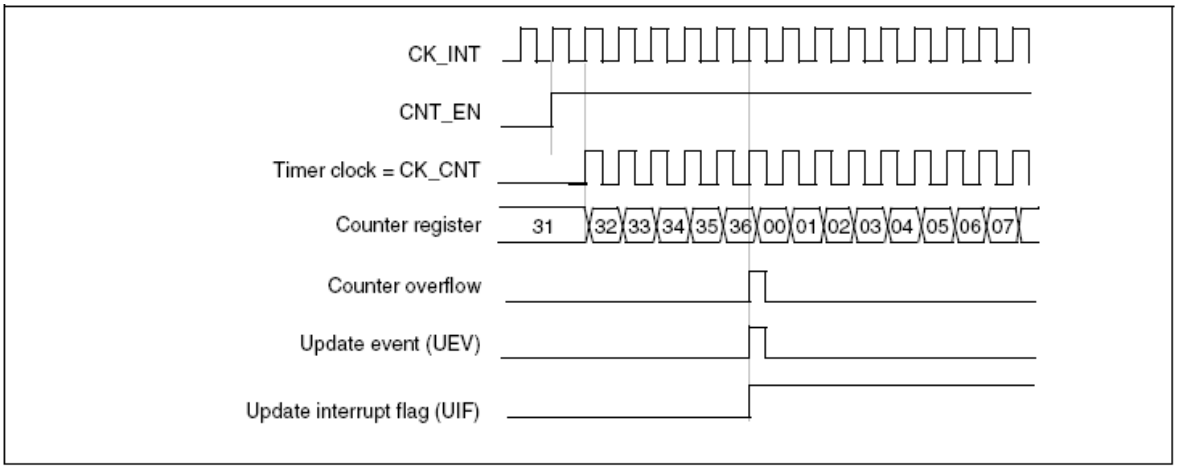


图144 计数器时序图，内部时钟分频系数为2

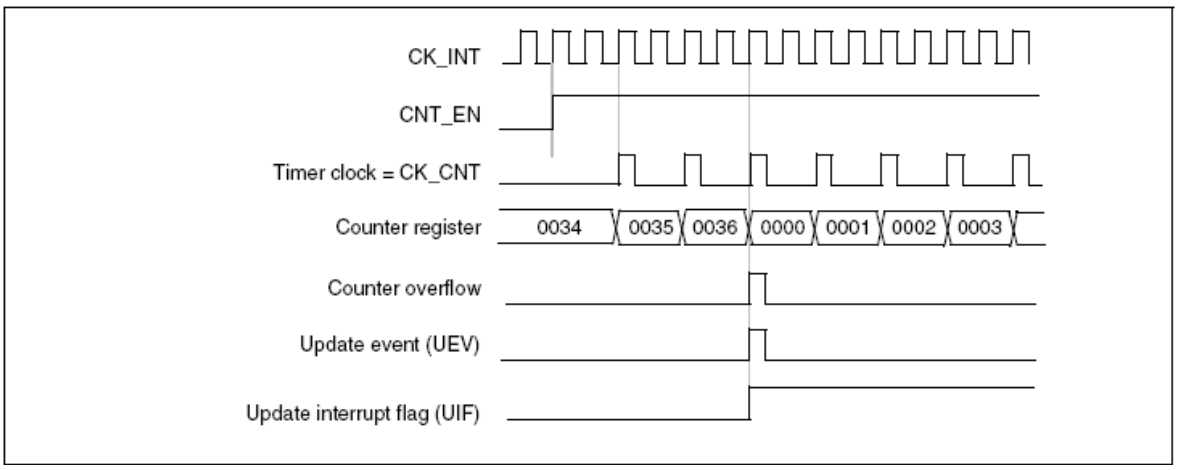


图145 计数器时序图，内部时钟分频系数为4

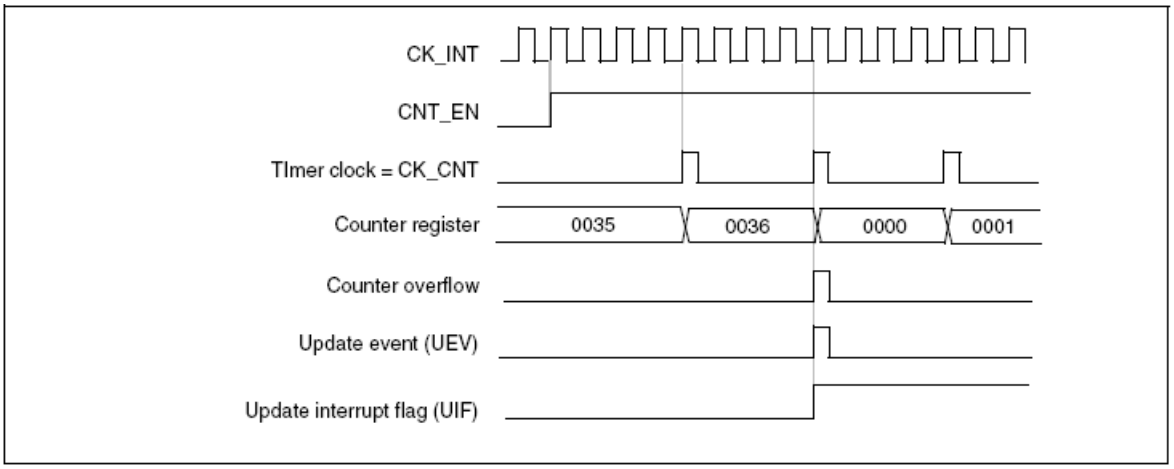


图146 计数器时序图，内部时钟分频系数为N

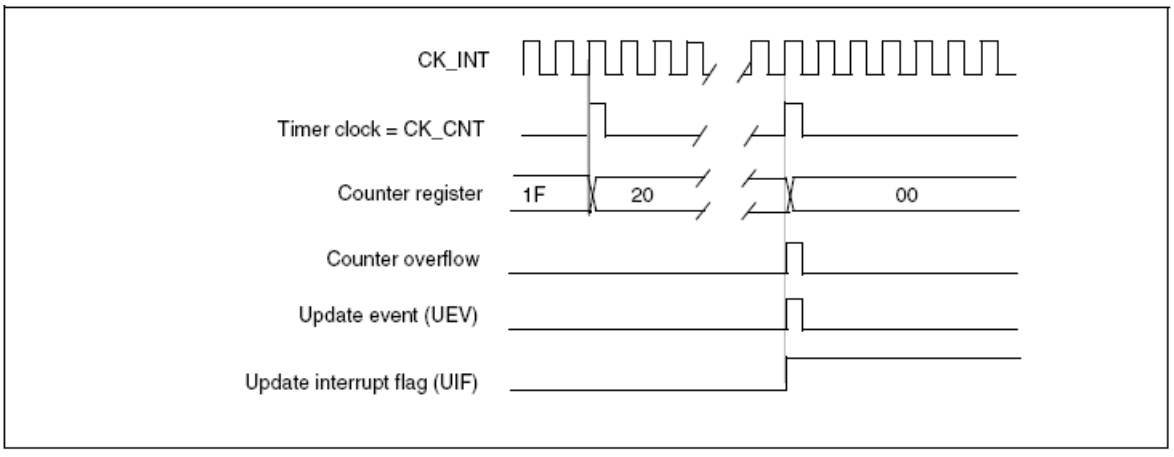


图147 计数器时序图，当ARPE=0时的更新事件(TIMx_ARR没有预装载)

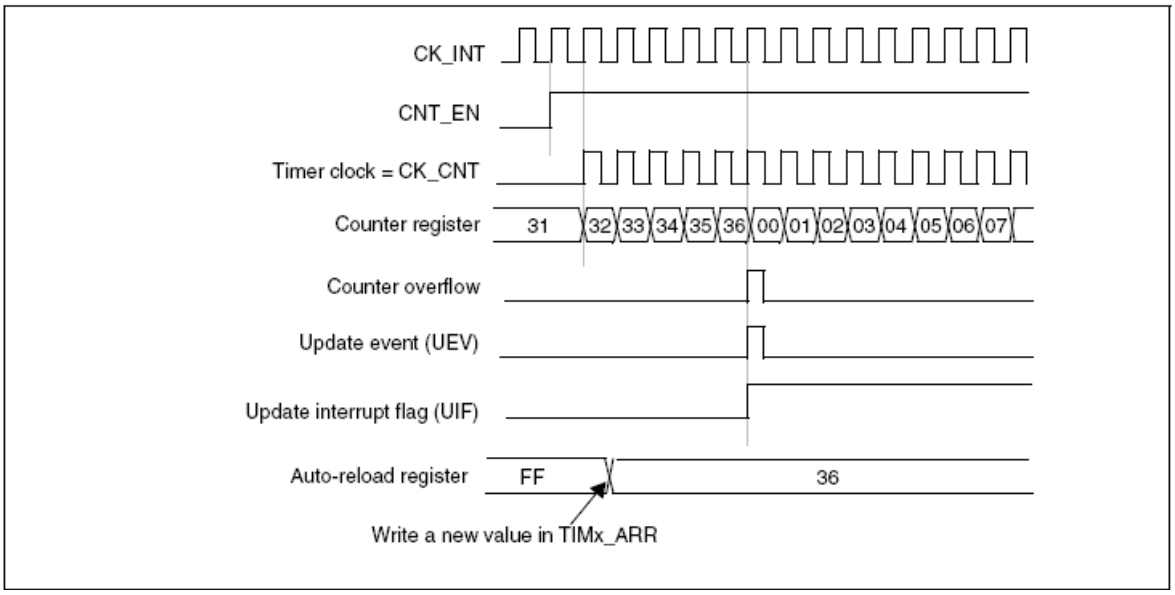
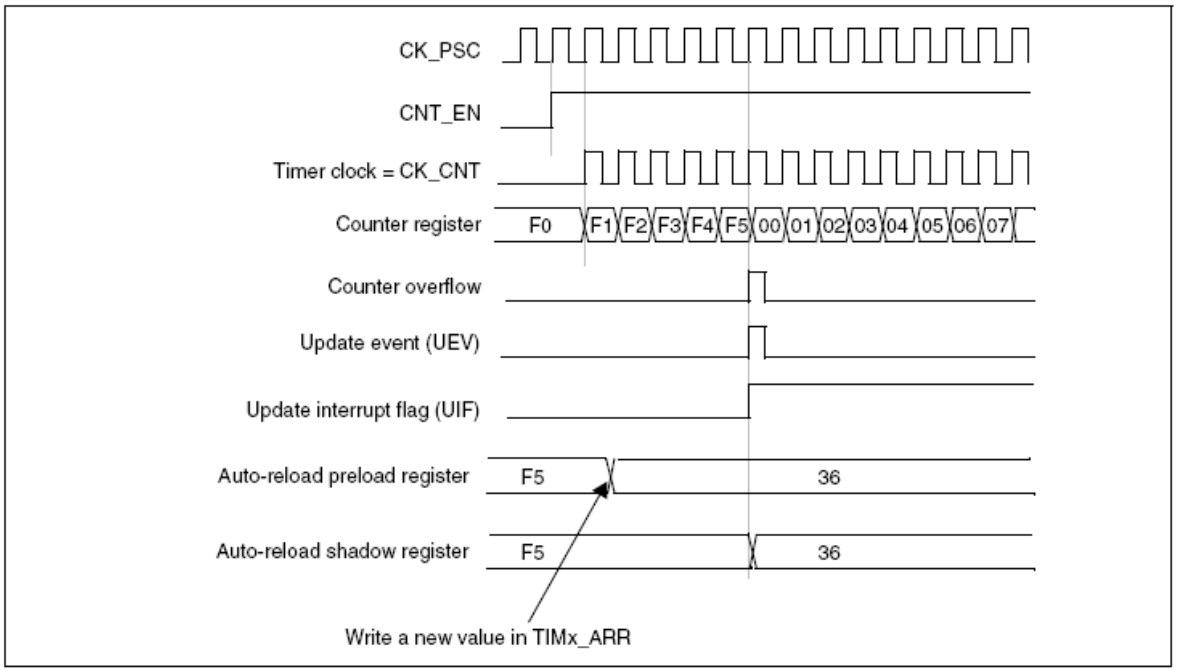


图148 计数器时序图，当ARPE=1时的更新事件(预装载TIMx_ARR)



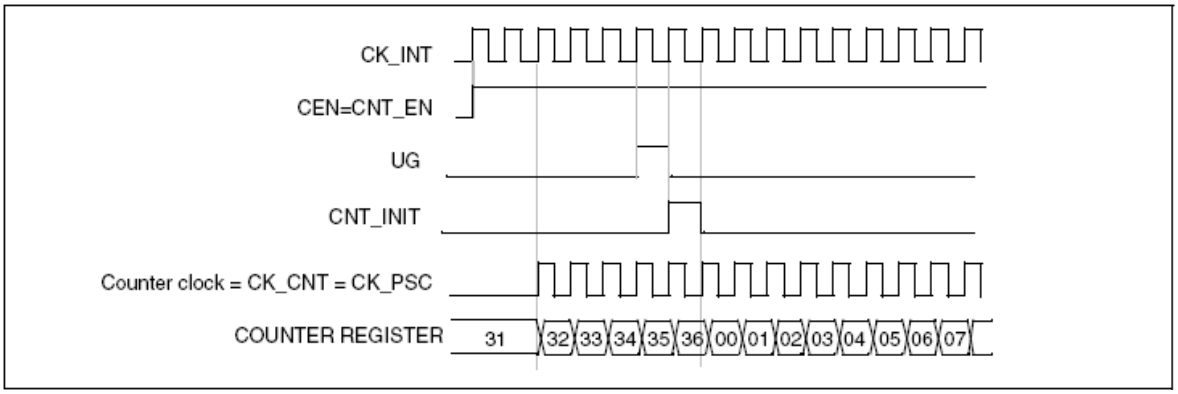
14.3.3 时钟源

计数器的时钟由内部时钟(CK_INT)提供。

TIMx_CR1寄存器的CEN位和TIMx_EGR寄存器的UG位是实际的控制位，(除了UG位被自动清除外)只能通过软件改变它们。一旦置CEN位为'1'，内部时钟即向预分频器提供时钟。

下图示出控制电路和向上计数器在普通模式下，没有预分频器时的操作。

图149 普通模式时序图，内部时钟分频系数为1



14.3.4 调试模式

当微控制器进入调试模式(Cortex-M3核心停止)时，根据DBG模块中的配置位DBG_TIMx_STOP的设置，TIMx计数器或者继续计数或者停止工作。详见26.15.2节。

14.4 TIM6和TIM7寄存器

有关寄存器描述中用到的缩写，请参考1.1节。

14.4.1 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								ARPE	保留			OPM	URS	UDIS	CEN
res								rw	res			rw	rw	rw	rw

位15:8	保留，始终读为0。
位7	**ARPE**：自动重装载预装载使能 0：TIMx_ARR寄存器没有缓冲 1：TIMx_ARR寄存器具有缓冲
位6:4	保留，始终读为0。
位3	**OPM**：单脉冲模式 0：在发生更新事件时，计数器不停止 1：在发生下次更新事件时，计数器停止计数(清除CEN位)。
位2	**URS**：更新请求源 该位由软件设置和清除，以选择UEV事件的请求源。 0：如果使能了中断或DMA，以下任一事件产生一个更新中断或DMA请求： - 计数器上溢或下溢 - 设置UG位 - 通过从模式控制器产生的更新 1：如果使能了中断或DMA，只有计数器上溢或下溢可以产生更新中断或DMA请求。

位1	**UDIS**：禁止更新 该位由软件设置和清除，以使能或禁止UEV事件的产生。 0：UEV使能。更新事件(UEV)可以由下列事件产生： - 计数器上溢或下溢 - 设置UG位 - 通过从模式控制器产生的更新 产生更新事件后，带缓冲的寄存器被加载为预加载数值。 1：禁止UEV。不产生更新事件(UEV)，影子寄存器保持它的内容(ARR，PSC)。但是如果设置了UG位或从模式控制器产生了一个硬件复位，则计数器和预分频器将被重新初始化。
位0	**CEN**：计数器使能 0：关闭计数器 1：使能计数器 注：门控模式只能在软件已经设置了CEN位时有效，而触发模式可以自动地由硬件设置CEN位。 在单脉冲模式下，当产生更新事件时CEN被自动清除。

14.4.2 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									MMS[2:0]			保留			
res									rw			res			

位15:7	保留，始终读为0。
位6:4	**MMS**：主模式选择 这些位用于选择在主模式下向从定时器发送的同步信息(TRGO)： 000：**复位** – 使用TIMx_EGR寄存器的UG位作为触发输出(TRGO)。如果触发输入产生了复位(从模式控制器配置为复位模式)，则相对于实际的复位信号，TRGO上的信号有一定的延迟。 001：**使能** – 计数器使能信号CNT_EN被用于作为触发输出(TRGO)。它可用于在同一时刻启动多个定时器，或控制使能从定时器的时机。计数器使能信号是通过CEN控制位和配置为门控模式时的触发输入的逻辑或产生。 当计数器使能信号是通过触发输入控制时，在TRGO输出上会有一些延迟，除非选择了主/从模式(见TIMx_SMCR寄存器的MSM位)。 010：更新 – 更新事件被用于触发输出(TRGO)。例如一个主定时器可以作为从定时器的预分频器使用。
位3:0	保留，始终读为0。

14.4.3 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							UDE	保留							UIE
res							rw	res							rw

位15:9	保留，始终读为0。
位8	**UDE**：更新DMA请求使能 0：禁止更新DMA请求 1：使能更新DMA请求
位7:1	保留，始终读为0。

位0	**UIE**：更新中断使能 0：禁止更新中断 1：使能更新中断

14.4.4 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															UIF
res															rc w0

位15:1	保留，始终读为0。
位0	**UIF**：更新中断标志 硬件在更新中断时设置该位，它由软件清除。 0：没有产生更新。 1：产生了更新中断。下述情况下由硬件设置该位： – 计数器产生上溢或下溢并且TIMx_CR1中的UDIS=0； – 如果TIMx_CR1中的URS=0并且UDIS=0，当使用TIMx_EGR寄存器的UG位重新初始化计数器CNT时。

14.4.5 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															UG
res															w

位15:1	保留，始终读为0。
位0	**UG**：产生更新事件 该位由软件设置，由硬件自动清除。 0：无作用 1：重新初始化定时器的计数器并产生对寄存器的更新。注意：预分频器也被清除(但预分频不变)。

14.4.6 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															

位15:0	**CNT[15:0]**：计数器数值。

14.4.7 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															

位15:0	**PSC[15:0]**：预分频器数值。 计数器的时钟频率CK_CNT是f_{CK_PSC}/(PSC[15:1]+1)。 在每一次更新事件时，PSC的数值被传送到实际的预分频寄存器中。

14.4.8 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															

位15:0	**ARR[15:0]**：自动重装载数值。 ARR的数值将传送到实际的自动重装载寄存器中。 关于ARR的更新和作用，详见14.3.1。 如果自动重装载数值为0，则计数器停止。

14.4.9 TIM6 和TIM7 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62 TIM6和TIM7– 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																								ARPE	保留			OPM	URS	UDIS	CEN
	复位值																									0				0	0	0	0
004h	TIMx_CR2	保留																								MMS[2:0]			保留				
	复位值																									0	0	0					
008h	保留																																
00Ch	TIMx_DIER	保留																							UDE	保留							UIE
	复位值																								0								0
010h	TIMx_SR	保留																															UIF
	复位值																																0
014h	TIMx_EGR	保留																															UG
	复位值																																0
018h	保留																																
01Ch	保留																																
020h	保留																																
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
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

有关寄存器的起始地址，参见表1。

15 实时时钟(RTC)

小容量产品是指闪存存储器容量在16K至32K字节之间的STM32F101xx、STM32F102xx和STM32F103xx微控制器。

中容量产品是指闪存存储器容量在64K至128K字节之间的STM32F101xx、STM32F102xx和STM32F103xx微控制器。

大容量产品是指闪存存储器容量在256K至512K字节之间的STM32F101xx和STM32F103xx微控制器。

除非特别说明，本章节描述的模块应用于整个STM32F10xxx微控制器系列。

15.1 RTC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，禁止访问后备寄存器和RTC，防止对后备区域(BKP)的意外写操作。执行以下操作使能对后备寄存器和RTC的访问：

- 设置寄存器RCC_APB1ENR的PWREN和BKPEN位来使能电源和后备接口时钟
- 设置寄存器PWR_CR的DBP位使能对后备寄存器和RTC的访问。

15.2 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)
- 可以选择以下三种RTC的时钟源：
  - HSE 时钟除以 128
  - LSE 振荡器时钟
  - LSI振荡器时钟(详见6.2.8节RTC时钟)
- 2种独立的复位类型：
  - APB1 接口由系统复位
  - RTC核心(预分频器、闹钟、计数器和分频器)只能由后备域复位(详见6.1.3节)。
- 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。

15.3 功能描述

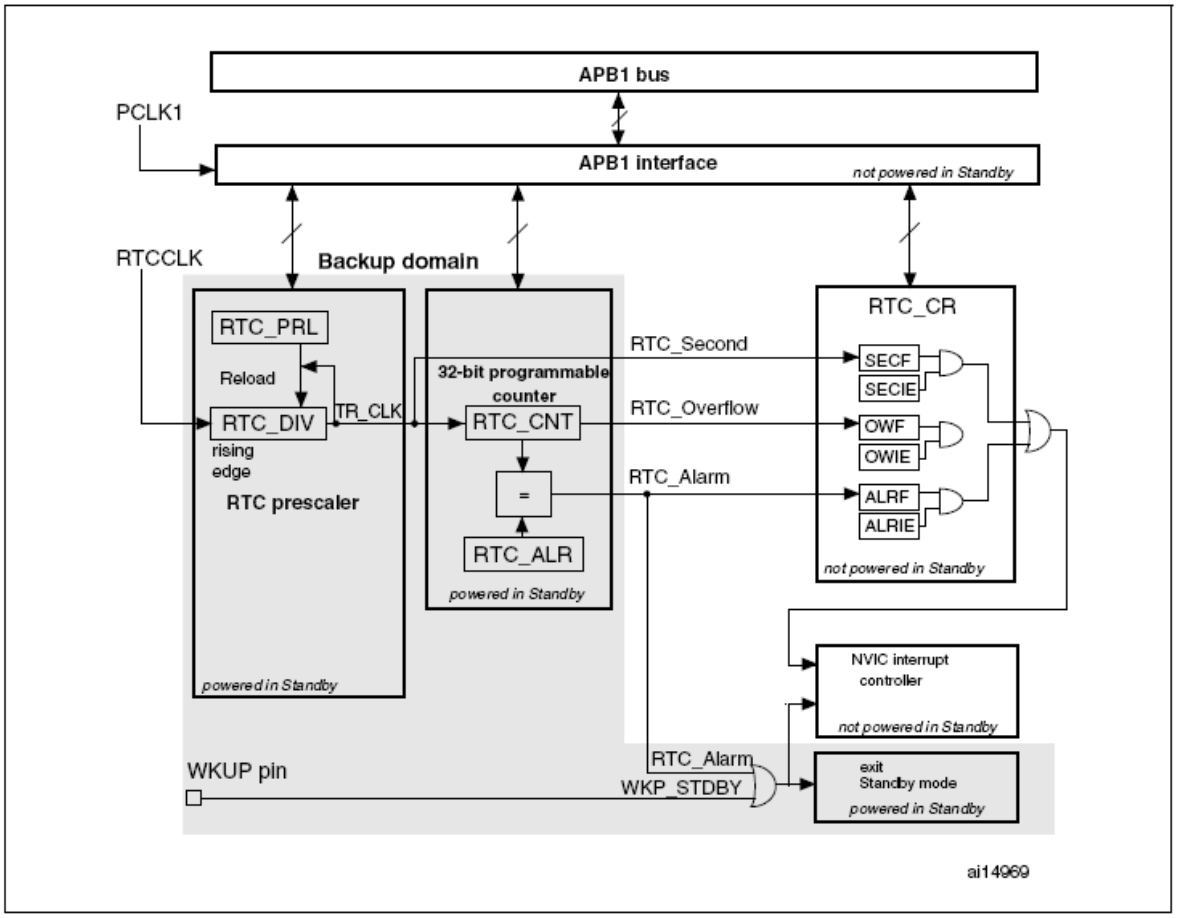
15.3.1 概述

RTC由两个主要部分组成(参见下图)。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(参见15.4节)。APB1接口由APB1总线时钟驱动，用来与APB1总线接口。

另一部分(RTC核心)由一组可编程计数器组成，分成两个主要模块。第一个模块是RTC的预分频模块，它可编程产生最长为1秒的RTC时间基准TR_CLK。RTC的预分频模块包含了一个20位的可编程分频器(RTC预分频器)。如果在RTC_CR寄存器中设置了相应的允许位，则在每个

TR_CLK周期中RTC产生一个中断(秒中断)。第二个模块是一个32位的可编程计数器，可被初始化为当前的系统时间。系统时间按TR_CLK周期累加并与存储在RTC_ALR寄存器中的可编程时间相比较，如果RTC_CR控制寄存器中设置了相应允许位，比较匹配时将产生一个闹钟中断。

图150 RTC简化框图



15.3.2 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，详见6.1.3节。

15.3.3 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与RTC APB1时钟进行重新同步的RTC时钟的上升沿被更新。RTC标志也是如此的。

这意味着，如果APB1接口刚刚被开启之后，在第一次的内部寄存器更新之前，从APB1上读出RTC寄存器的第一个值可能被破坏了(通常读到0)。下述几种情况下能够发生这种情形：

- 发生系统复位或电源复位
- 系统刚从待机模式唤醒(参见4.3节)。
- 系统刚从停机模式唤醒(参见4.3节)。

所有以上情况中，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寄存器曾经被禁止的RTC 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。

15.3.4 配置RTC寄存器

必须设置RTC_CRL寄存器中的CNF位，使RTC进入配置模式后，才能写入RTC_PRL、RTC_CNT、RTC_ALR寄存器。

另外，对RTC任何寄存器的写操作，都必须在前一次写操作结束后进行。可以通过查询RTC_CR寄存器中的RTOFF状态位，判断RTC寄存器是否处于更新中。仅当RTOFF状态位是'1'时，才可以写入RTC寄存器。

配置过程：

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

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。

15.3.5 RTC标志的设置

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

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

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

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

图151 RTC秒和闹钟波形图示例，PR=0003，ALARM=00004

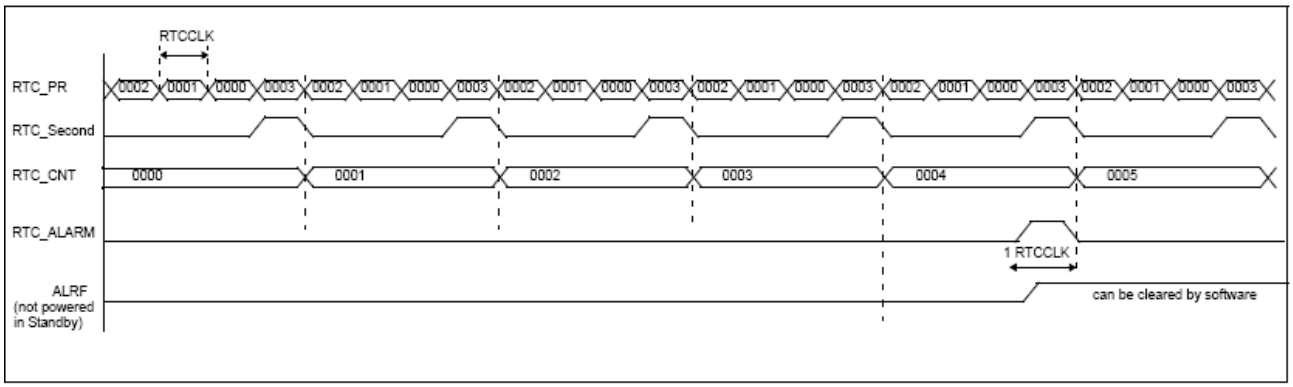
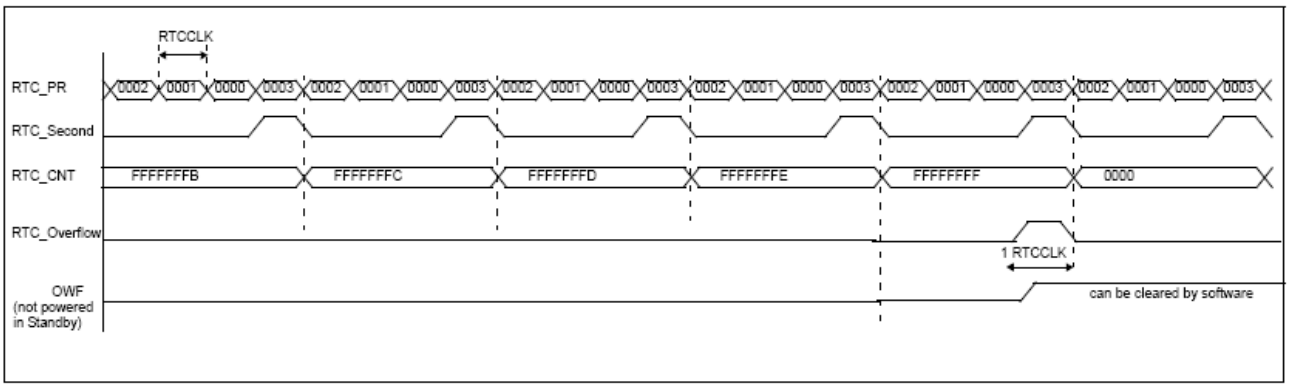


图152 RTC溢出波形图示例，PR=0003



15.4 RTC寄存器描述

关于寄存器描述中的缩略词，请参考1.1节。

15.4.1 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15.3.4节)。

15.4.2 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，则表示无法对任何的RTC寄存器进行写操作。此位为只读位。 0：上一次对RTC寄存器的写操作仍在进行; 1：上一次对RTC寄存器的写操作已经完成。
位4	**CNF**：配置标志 此位必须由软件置'1'以进入配置模式，从而允许向RTC_CNT、RTC_ALR或RTC_PRL寄存器写入数据。只有当此位在被置'1'并重新由软件清'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寄存器由软件更新或清'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寄存器所设置的预定值，此位由硬件置'1'。如果RTC_CRH寄存器中ALRIE=1，则产生中断。此位只能由软件清'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'同时RTC计数器加1。因此，此标志为分辨率可编程的RTC计数器提供一个周期性的信号(通常为1秒)。如果RTC_CRH寄存器中SECIE=1，则产生中断。此位只能由软件清除。对此位写'1'是无效的。 0：秒标志条件不成立； 1：秒标志条件成立。

RTC的功能由这个控制寄存器控制。当正在前一个写操作还未完成时(RTOFF=0时，详见15.3.4节)，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，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)。

15.4.3 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。它们受RTC_CR寄存器的RTOFF位保护，仅当RTOFF值为'1'时允许进行写操作。

RTC预分频装载寄存器高位(RTC_PRLH)

偏移地址：0x08

只写(参见15.3.4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： fTR_CLK = fRTCCLK/(PRL[19:0]+1) 注：不推荐使用0值，否则无法正确的产生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15.3.4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})，这个寄存器中写入7FFFh可获得周期为1秒钟的信号。

15.4.4 RTC预分频器余数寄存器(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。用户可通过读取RTC_DIV寄存器，以获得预分频计数器的当前值，而不停止分频计数器的工作，从而获得精确的时间测量。此寄存器是只读寄存器，其值在RTC_PRL或RTC_CNT寄存器中的值发生改变后，由硬件重新装载。

RTC预分频器余数寄存器高位(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时钟分频器余数高位。

RTC预分频器余数寄存器低位(RTC_DIVL)

偏移地址：0x14

复位值：0x8000

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

位15:0	**RTC_DIV[15:0]**：RTC时钟器余数低位。

15.4.5 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。他们受RTC_CR上的位RTOFF写保护，仅当RTOFF值为'1'时，允许写操作。在高或低寄存器

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

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计数器当前值的高位部分。要对此寄存器进行写操作前，必须先进入配置模式(参见15.3.4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计数器当前值的低位部分。要对此寄存器进行写操作，必须先进入配置模式(参见15.3.4节)。

15.4.6 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。此寄存器受RTC_CR寄存器里的RTOFF位写保护，仅当RTOFF值为'1'时，允许写操作。

RTC闹钟寄存器高位(RTC_ALRH)

偏移地址：0x20

只写(参见15.3.4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，必须先进入配置模式(参见15.3.4节)。

RTC闹钟寄存器低位(RTC_ALRL)

偏移地址：0x24

只写(参见15.3.4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，必须先进入配置模式(参见15.3.4节)。

15.4.7 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63 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																	0	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																	0	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																	0	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	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
024h	RTC_ALRL	保留																ALR[15:0]															
	复位值																	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

有关寄存器的起始地址，参见表1。

16 独立看门狗(IWDG)

小容量产品是指闪存存储器容量在16K至32K字节之间的STM32F101xx、STM32F102xx和STM32F103xx微控制器。

中容量产品是指闪存存储器容量在64K至128K字节之间的STM32F101xx、STM32F102xx和STM32F103xx微控制器。

大容量产品是指闪存存储器容量在256K至512K字节之间的STM32F101xx和STM32F103xx微控制器。

除非特别说明，本章节描述的模块应用于整个STM32F10xxx微控制器系列。

16.1 简介

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(仅适用于窗口型看门狗)或产生系统复位。

独立看门狗(IWDG)由专用的40kHz的低速时钟驱动，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操作。

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

关于窗口看门狗的详情，请参看第17章。

16.2 IWDG主要性能

- 自由运行的递减计数器
- 时钟由独立的RC振荡器提供(可在停止和待机模式下工作)
- 看门狗被激活后，则在计数器计数至0x000时产生复位

16.3 IWDG功能描述

图153为独立看门狗模块的功能框图。

在键寄存器(IWDG_KR)中写入0xCCCC，开始启用独立看门狗；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 。

16.3.1 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。

16.3.2 寄存器访问保护

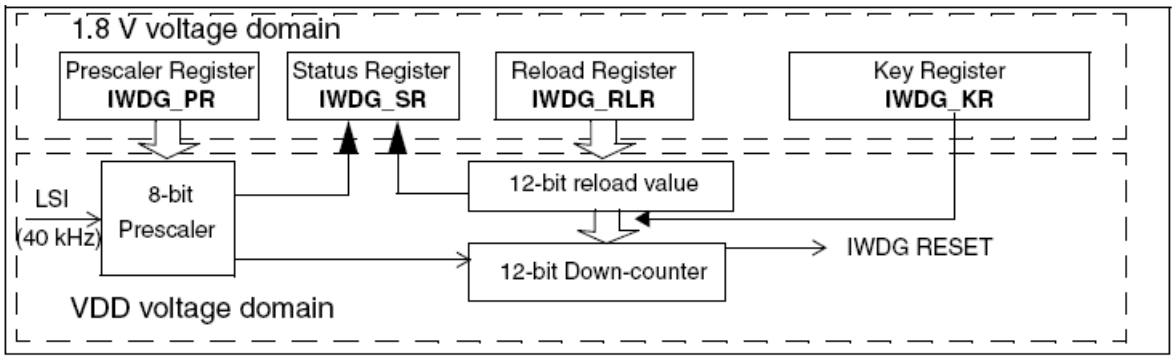
IWDG_PR和IWDG_RLR寄存器具有写保护功能。 要修改这两个寄存器的值，必须先向IWDG_KR寄存器中写入0x5555。以不同的值写入这个寄存器将会打乱操作顺序，寄存器将重新被保护。重装载操作(即写入0xAAAA)也会启动写保护功能。

状态寄存器指示预分频值和递减计数器是否正在被更新。

16.3.3 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。

图153 独立看门狗框图



注：看门狗功能处于VDD供电区，即在停机和待机模式时仍能正常工作。

表64 看门狗超时时间(40kHz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	409.6
/8	1	0.2	819.2
/16	2	0.4	1638.4
/32	3	0.8	3276.8
/64	4	1.6	6553.6
/128	5	3.2	13107.2
/256	(6或7)	6.4	26214.4

注：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。

通过对LSI进行校准可获得相对精确的看门狗超时时间。有关LSI校准的问题，详见6.2.5节。

16.4 IWDG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。

16.4.1 键寄存器(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) 软件必须以一定的间隔写入0xAAAA，否则，当计数器为0时，看门狗会产生复位。 写入0x5555表示允许访问IWDG_PR和IWDG_RLR寄存器。(见16.3.2节) 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)。

16.4.2 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16.3.2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: 预分频因子=4　　100: 预分频因子=64 001: 预分频因子=8　　101: 预分频因子=128 010: 预分频因子=16　　110: 预分频因子=256 011: 预分频因子=32　　111: 预分频因子=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，读出的值才有效。

16.4.3 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16.3.2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，重装载值会被传送到计数器中。随后计数器从这个值开始递减计数。看门狗超时周期可通过此重装载值和时钟预分频值来计算，参照表64。 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，读出的值才有效。

16.4.4 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PVU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。)*

16.4.5 IWDG寄存器映像

表65 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

有关寄存器的起始地址，参见表1。

17 窗口看门狗(WWDG)

小容量产品是指闪存存储器容量在16K至32K字节之间的STM32F101xx、STM32F102xx和STM32F103xx微控制器。

中容量产品是指闪存存储器容量在64K至128K字节之间的STM32F101xx、STM32F102xx和STM32F103xx微控制器。

大容量产品是指闪存存储器容量在256K至512K字节之间的STM32F101xx和STM32F103xx微控制器。

除非特别说明，本章节描述的模块应用于整个STM32F10xxx微控制器系列。

17.1 WWDG简介

窗口看门狗通常被用来监测由外部干扰或不可预见的逻辑条件造成的应用程序背离正常的运行序列而产生的软件故障。除非递减计数器的值在T6位变成0前被刷新，看门狗电路在达到预置的时间周期时，会产生一个MCU复位。在递减计数器达到窗口寄存器数值之前，如果7位的递减计数器数值(在控制寄存器中)被刷新， 那么也将产生一个MCU复位。这表明递减计数器需要在一个有限的时间窗口中被刷新。

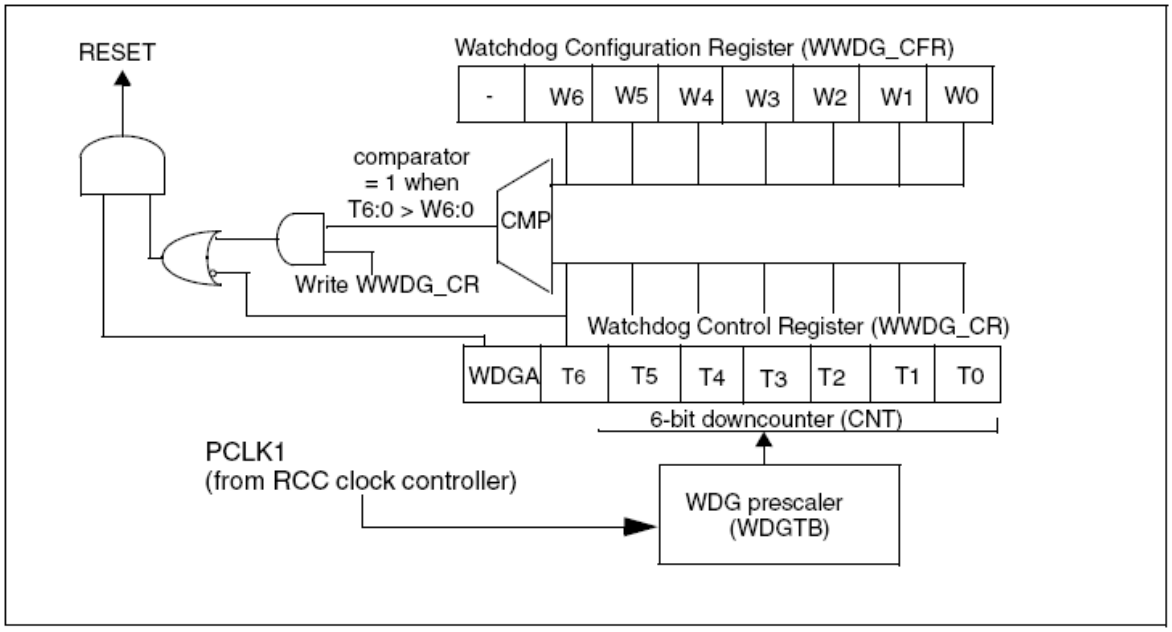
17.2 WWDG主要特性

- 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 155。
- 如果启动了看门狗并且允许中断，当递减计数器等于0x40时产生早期唤醒中断(EWI)，它可以被用于重装载计数器以避免WWDG复位。

17.3 WWDG功能描述

如果看门狗被启动(WWDG_CR寄存器中的WDGA位被置'1')， 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数值时重新装载计数器，将产生一个复位。

图154 看门狗框图



应用程序在正常运行过程中必须定期地写入WWDG_CR寄存器以防止MCU发生复位。只有当计数器值小于窗口寄存器的值时，才能进行写操作。储存在WWDG_CR寄存器中的数值必须在0xFF和0xC0之间：

- 启动看门狗
  在系统复位后，看门狗总是处于关闭状态，设置WWDG_CR寄存器的WDGA位能够开启看门狗，随后它不能再被关闭，除非发生复位。
- 控制递减计数器
  递减计数器处于自由运行状态，即使看门狗被禁止，递减计数器仍继续递减计数。当看门狗被启用时，T6位必须被设置，以防止立即产生一个复位。
  T[5:0]位包含了看门狗产生复位之前的计时数目；复位前的延时时间在一个最小值和一个最大值之间变化，这是因为写入WWDG_CR寄存器时，预分频值是未知的。
  配置寄存器(WWDG_CFR) 中包含窗口的上限值：要避免产生复位，递减计数器必须在其值小于窗口寄存器的数值并且大于0x3F时被重新装载，图155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)。设置WWDG_CFR寄存器中的WEI位开启该中断。当递减计数器到达0x40时，则产生此中断，相应的中断服务程序(ISR)可以用来加载计数器以防止WWDG复位。在WWDG_SR寄存器中写'0'可以清除该中断。

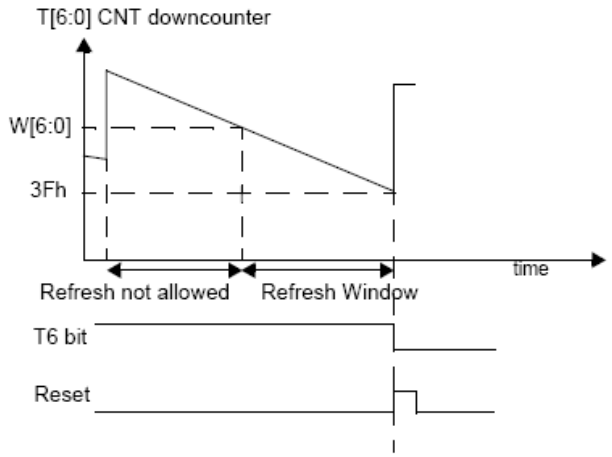
注： *T6位可以被用来产生一个软件复位(WDGA位被置位，T6位清零)*

17.4 如何编写看门狗超时程序

图155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155提供的计算公式。

警告：当写入 WWDG_CR 寄存器时，始终置 T6 位为'1'以避免立即产生一个复位。

图155 窗口看门狗时序图



计算超时的公式如下：

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

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

17.5 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，26.15.2节。

17.6 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。

17.6.1 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。

17.6.2 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。

17.6.3 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'。

17.6.4 WWDG寄存器映像

表66 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

18 灵活的静态存储器控制器(FSMC)

小容量产品是指闪存容量介于16K字节至32K字节的STM32F101xx、STM32F102xx和STM32F103xx微控制器。

中容量产品是指闪存容量介于64K字节至128K字节的STM32F101xx、STM32F102xx和STM32F103xx微控制器。

大容量产品是指闪存容量介于256K至512K字节的STM32F101xx和STM32F103xx微控制器。

本章内容只适用于大容量产品。

18.1 FSMC功能描述

FSMC模块能够与同步或异步存储器和16位PC存储器卡接口，它的主要作用是：

- 将AHB传输信号转换到适当的外部设备协议
- 满足访问外部设备的时序要求

所有的外部存储器共享控制器输出的地址、数据和控制信号，每个外部设备可以通过一个唯一的片选信号加以区分。FSMC在任一时刻只访问一个外部设备。

FSMC具有下列主要功能：

- 具有静态存储器接口的器件包括：
  - 静态随机存储器(SRAM)
  - 只读存储器(ROM)
  - NOR 闪存
  - PSRAM(4 个存储器块)
- 两个NAND闪存块，支持硬件ECC并可检测多达8K字节数据
- 16位的PC卡兼容设备
- 支持对同步器件的成组(Burst)访问模式，如NOR闪存和PSRAM
- 8或16位数据总线
- 每一个存储器块都有独立的片选控制
- 每一个存储器块都可以独立配置
- 时序可编程以支持各种不同的器件：
  - 等待周期可编程(多达 15 个周期)
  - 总线恢复周期可编程(多达 15 个周期)
  - 输出使能和写使能延迟可编程(多达 15 周期)
  - 独立的读写时序和协议，可支持宽范围的存储器和时序
- PSRAM和SRAM器件使用的写使能和字节选择输出
- 将32位的AHB访问请求，转换到连续的16位或8位的，对外部16位或8位器件的访问
- 具有16个字，每个字32位宽的写入FIFO，允许在写入较慢存储器时释放AHB进行其它操作。在开始一次新的FSMC操作前，FIFO要先被清空。

通常在系统复位或上电时，应该设置好所有定义外部存储器类型和特性的FSMC寄存器，并保持它们的内容不变；当然，也可以在任何时候改变这些设置。

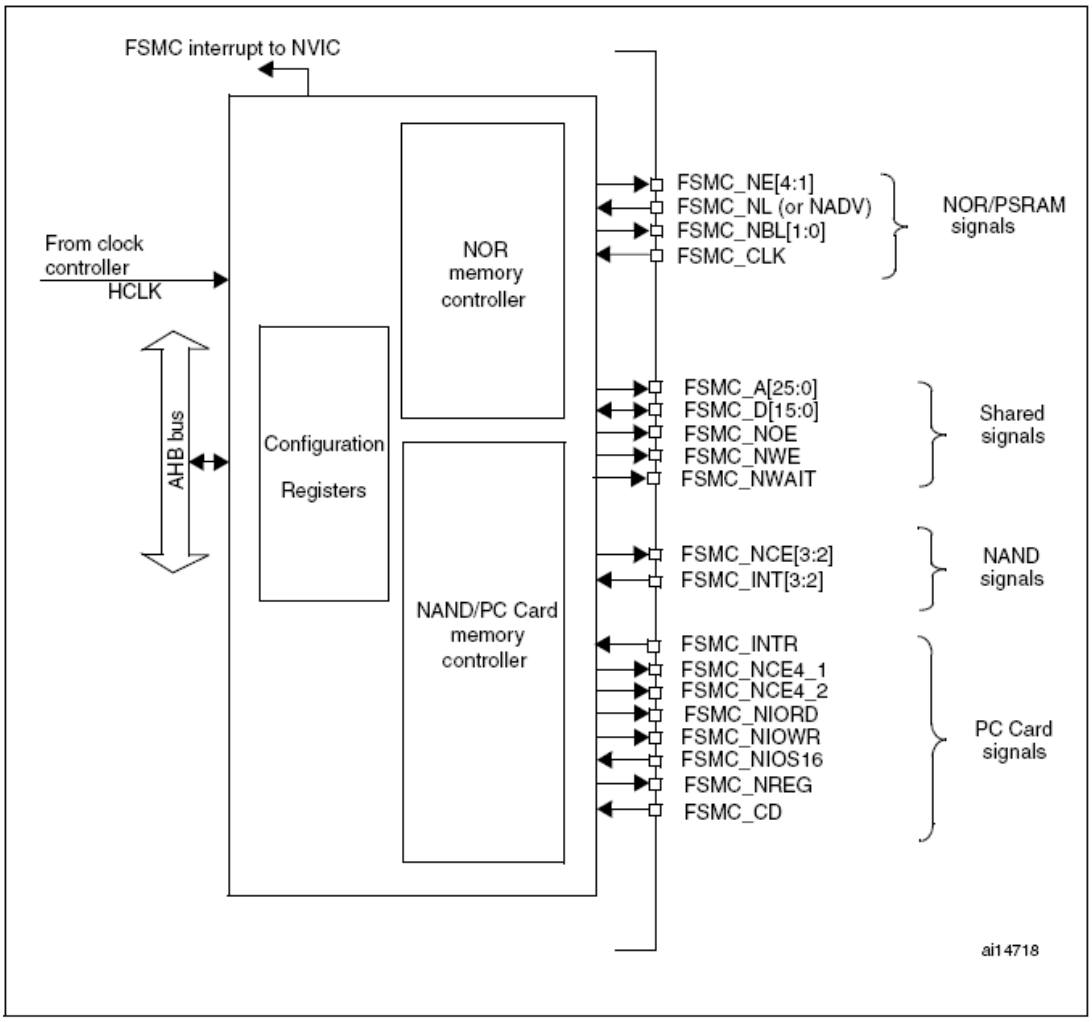
18.2 框图

FSMC包含四个主要模块：

- AHB接口(包含FSMC配置寄存器)
- NOR闪存和PSRAM控制器
- NAND闪存和PC卡控制器
- 外部设备接口

FSMC框图如下：

图156 FSMC框图



18.3 AHB接口

AHB接口为内部CPU和其它总线控制设备访问外部静态存储器提供了通道。

AHB操作被转换到外部设备的操作。当选择的外部存储器的数据通道是16或8位时，在AHB上的32位数据会被分割成连续的16或8位的操作。

AHB时钟(HCLK)是FSMC的参考时钟。

18.3.1 支持的存储器和操作

一般的操作规则

请求AHB操作的数据宽度可以是8位、16位或32位，而外部设备则是固定的数据宽度，此时需要保障实现数据传输的一致性。

因此，FSMC执行下述操作规则：

- AHB操作的数据宽度与存储器数据宽度相同：无数据传输一致性的问题。
- AHB操作的数据宽度大于存储器的数据宽度：此时FSMC将AHB操作分割成几个连续的较小数据宽度的存储器操作，以适应外部设备的数据宽度。
- AHB操作的数据宽度小于存储器的数据宽度：
  依据外部设备的类型，异步的数据传输有可能不一致。

– 与具有字节选择功能的存储器(SRAM、ROM、PSRAM 等)进行异步传输时，FSMC 执行读写操作并通过它的字节通道 BL[1:0]访问正确的数据。
– 与不具有字节选择功能的存储器(NOR 和 16 位 NAND 等)进行异步传输时，即需要对 16 位宽的闪存存储器进行字节访问；显然不能对存储器进行字节模式访问(只允许 16 位的数据传输)，因此：

a. 不允许进行写操作
b. 可以进行读操作(控制器读出完整的 16 位存储器数据，只使用需要的字节)。

配置寄存器

FSMC由一组寄存器进行配置。18.5.6节详细描述了NOR闪存和PSRAM控制器寄存器。18.6.7节详细描述了NAND闪存和PC卡寄存器。

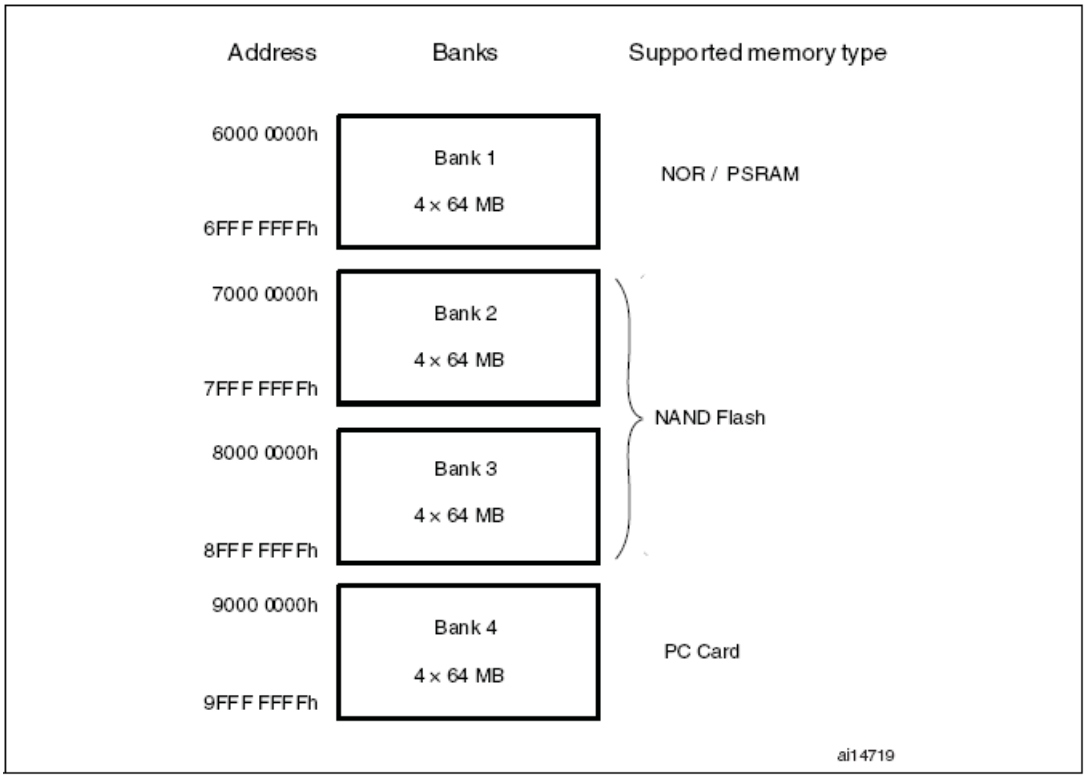
18.4 外部设备地址映像

从FSMC的角度看，可以把外部存储器划分为固定大小为256M字节的四个存储块，见图157。

- 存储块1用于访问最多4个NOR闪存或PSRAM存储设备。这个存储区被划分为4个NOR/PSRAM区并有4个专用的片选。
- 存储块2和3用于访问NAND闪存设备，每个存储块连接一个NAND闪存。
- 存储块4用于访问PC卡设备

每一个存储块上的存储器类型是由用户在配置寄存器中定义的。

图157 FSMC存储块



18.4.1 NOR和PSRAM地址映像

HADDR[27:26]位用于选择四个存储块之一：

表67 NOR/PSRAM存储块选择

HADDR[27:26](1)	选择的存储块
00	存储块1 NOR/PSRAM 1
01	存储块1 NOR/PSRAM 2
10	存储块1 NOR/PSRAM 3
11	存储块1 NOR/PSRAM 4

(1) HADDR是需要转换到外部存储器的内部AHB地址线。

HADDR[25:0]包含外部存储器地址。HADDR是字节地址，而存储器访问不都是按字节访问，因此接到存储器的地址线依存储器的数据宽度有所不同，如下表：

表68 外部存储器地址

数据宽度(1)	连到存储器的地址线	最大访问存储器空间(位)
8位	HADDR[25:0]与FSMC_A[25:0]对应相连	64M字节 x 8 = 512 M位
16位	HADDR[25:1]与FSMC_A[24:0]对应相连，HADDR[0]未接	64M字节/2 x 16 = 512 M位

(1) 对于16位宽度的外部存储器，FSMC将在内部使用HADDR[25:1]产生外部存储器的地址FSMC_A[24:0]。不论外部存储器的宽度是多少(16位或8位)，FSMC_A[0]始终应该连到外部存储器的地址线A[0]。

NOR闪存和PSRAM的非对齐访问支持

每个NOR闪存或PSRAM存储器块都可以配置成支持非对齐的数据访问。

在存储器一侧，依据访问的方式是异步或同步，需要考虑两种情况：

- **异步模式**：这种情况下，只要每次访问都有准确的地址，完全支持非对齐的数据访问。
- **同步模式**：这种情况下，FSMC只发出一次地址信号，然后成组的数据传输通过时钟CLK顺序进行。
  某些NOR存储器支持线性的非对齐成组访问，固定数目的数据字可以从连续的以N为模的地址读取(典型的N为8或16，可以通过NOR闪存的配置寄存器设置)。此种情况下，可以把存储器的非对齐访问模式设置为与AHB相同的模式。

如果存储器的非对齐访问模式不能设置为与AHB相同的模式，应该通过FSMC配置寄存器的相应位禁止非对齐访问，并把非对齐的访问请求分开成两个连续的访问操作。

18.4.2 NAND和PC卡地址映像

三个存储块可以用于NAND或PC卡的操作，每个存储块被划分为下述访问空间：

表69 存储器映像和时序寄存器

起始地址	结束地址	FSMC存储块	存储空间	时序寄存器
0x9C00 0000	0x9FFF FFFF	块4 – PC卡	I/O	FSMC_PIO4(0xB0)
0x9800 0000	0x9BFF FFFF		属性	FSMC_PATT4(0xAC)
0x9000 0000	0x93FF FFFF		通用	FSMC_PMEM4(0xA8)
0x8800 0000	0x8BFF FFFF	块3 – NAND闪存	属性	FSMC_PATT3(0x8C)
0x8000 0000	0x83FF FFFF		通用	FSMC_PMEM3(0x88)
0x7800 0000	0x7BFF FFFF	块2 – NAND闪存	属性	FSMC_PATT2(0x6C)
0x7000 0000	0x73FF FFFF		通用	FSMC_PMEM2(0x68)

对于NAND闪存存储器，通用和属性空间又可以在低256K字节部分划分为3个区(见表70)

- 数据区(通用/属性空间的前64K字节区域)
- 命令区(通用/属性空间的第2个64K字节区域)

- 地址区(通用/属性空间的第2个128K字节区域)

表70 NAND存储块选择

区域名称	HADDR[17:16]	地址范围
地址区	1X	0x020000~0x03FFFF
命令区	01	0x010000~0x01FFFF
数据区	00	0x000000~0x00FFFF

应用软件使用这3个区访问NAND闪存存储器：

- **发送命令到NAND闪存存储器**：软件只需对命令区的任意一个地址写入命令即可。
- **指定操作NAND闪存存储器的地址**：软件只需对地址区的任意一个地址写入命令即可。因为一个NAND地址可以有4或5个字节(依实际的存储器容量而定)，需要连续地执行对地址区的写才能输出完整的操作地址。
- **读写数据**：软件只需对数据区的任意一个地址写入或读出数据即可。

因为NAND闪存存储器自动地累加其内部的操作地址，读写数据时没有必要变换数据区的地址，即不必对连续的地址区操作。

18.5 NOR闪存和PSRAM控制器

FSMC可以产生适当的信号时序，驱动下述类型的存储器：

- 异步SRAM和ROM
  - 8 位
  - 16 位
  - 32 位
- PSRAM(Cellular RAM)
  - 异步模式
  - 突发模式
- NOR闪存
  - 异步模式或突发模式
  - 复用模式或非复用模式

FSMC对每个存储块输出一个唯一的片选信号NE[4:1]，所有其它的(地址、数据和控制)信号则是共享的。

在同步方式中，FSMC向选中的外部设备产生时钟(CLK)，该时钟的频率是HCLK时钟的整除因子。每个存储块的大小固定为64M字节。

每个存储块都有专门的寄存器控制(见18.6.7节)。

可编程的存储器参数包括访问时序(见下标)、是否支持非对齐数据存取和等待周期管理(只针对突发模式下访问PSRAM和NOR闪存)。

表71 可编程的NOR/PSRAM访问参数

参数	功能	访问方式	单位	最小	最大
地址建立时间	地址建立阶段的时间	异步	AHB时钟周期(HCLK)	1	16
地址保持时间	地址保持阶段的时间	异步，复用I/O	AHB时钟周期(HCLK)	1	16
数据建立时间	数据建立阶段的时间	异步	AHB时钟周期(HCLK)	1	256
总线恢复时间	总线恢复阶段的时间	异步或同步读	AHB时钟周期(HCLK)	1	16
时钟分频因子	存储器访问的时钟周期(CLK)与AHB时钟周期的比例	同步	AHB时钟周期(HCLK)	1	16
数据产生时间	突发模式下产生第一个数据所需的时钟数目	同步	存储器时钟周期(CLK)	2	17

18.5.1 外部存储器接口信号

表72、表73、表74列出了与NOR闪存和PSRAM接口的典型信号。

注： *具有前缀“N”的信号表示低有效信号*

NOR闪存，非复用接口

表72 非复用NOR闪存接口

FSMC信号名称	信号方向	功能
CLK	输出	时钟(同步突发模式使用)
A[25:0]	输出	地址总线
D[15:0]	输入/输出	双向数据总线
NE[x]	输出	片选，x = 1...4
NOE	输出	输出使能
NWE	输出	写使能
NWAIT	输入	NOR闪存要求FSMC等待的信号

NOR闪存存储器是按16位的字寻址，最大容量达64兆字节(26条地址线)。

NOR闪存，复用接口

表73 复用NOR闪存接口

FSMC信号名称	信号方向	功能
CLK	输出	时钟(同步突发模式使用)
A[25:16]	输出	地址总线
AD[15:0]	输入/输出	16位复用的，双向地址/数据总线
NE[x]	输出	片选，x = 1...4
NOE	输出	输出使能
NWE	输出	写使能
NL(=NADV)	输出	锁存使能(某些NOR闪存器件命名该信号为地址有效，NADV)
NWAIT	输入	NOR闪存要求FSMC等待的信号

NOR闪存存储器是按16位的字寻址，最大容量达64兆字节(26条地址线)。

PSRAM

表74 PSRAM

FSMC信号名称	信号方向	功能
CLK	输出	时钟(同步突发模式使用)
A[25:0]	输出	地址总线
D[15:0]	输入/输出	双向数据总线
NE[x]	输出	片选，x = 1...4 (PSRAM称其为NCE(Cellular RAM既CRAM))
NOE	输出	输出使能
NWE	输出	写使能
NL(=NADV)	输出	地址有效(存储器信号名称为：NADV)
NWAIT	输入	PSRAM要求FSMC等待的信号
NBL[1]	输出	高字节使能(存储器信号名称为：NUB)
NBL[0]	输出	低字节使能(存储器信号名称为：NLB)

PSRAM存储器是按16位的字寻址，最大容量达64兆字节(26条地址线)。

18.5.2 支持的存储器及其操作

下表列出了支持的存储器、访问模式和操作方式，FSMC不支持阴影部分的操作方式。

表75 FSMC支持的NOR闪存/PSRAM存储器和操作方式

存储器	模式	读/写	AHB数据宽度	存储器数据宽度	是否支持	注释
NOR闪存(总线复用和非总线复用)	异步	读	8	16	支持	
	异步	写	8	16	不支持	
	异步	读	16	16	支持	
	异步	写	16	16	支持	
	异步	读	32	16	支持	分成2次FSMC访问
	异步	写	32	16	支持	分成2次FSMC访问
	异步页	读	-	16	不支持	不支持这种模式
	同步	读	8	16	不支持	
	同步	读	16	16	支持	
	同步	读	32	16	支持	
PSRAM(总线复用和非总线复用)	异步	读	8	16	支持	
	异步	写	8	16	支持	使用字节线NBL[1:0]
	异步	读	16	16	支持	
	异步	写	16	16	支持	
	异步	读	32	16	支持	分成2次FSMC访问
	异步	写	32	16	支持	分成2次FSMC访问
	异步页	读	-	16	不支持	不支持这种模式
	同步	读	8	16	不支持	
	同步	读	16	16	支持	
	同步	读	32	16	支持	
SRAM和ROM	异步	读	8/16/32	8/16	支持	使用字节线NBL[1:0]
	异步	写	8/16/32	8/16	支持	使用字节线NBL[1:0]

18.5.3 时序规则

信号同步

- 所有的控制器输出信号在内部时钟(HCLK)的上升沿变化
- 在同步写模式(PSRAM)下，输出的数据在存储器时钟(CLK)的下降沿变化。

18.5.4 NOR闪存和PSRAM时序图

异步静态存储器(NOR闪存和PSRAM)

- 所有信号由内部时钟HCLK保持同步，但时钟不会输出到存储器；
- FSMC始终在片选信号NE失效前对数据线采样，这样能够保证符合存储器的数据保持时序(片选失效至数据失效的间隔，通常最小为0ns)；
- 当设置了扩展模式，可以在读和写时混合使用模式A、B、C和D(例如，允许以模式A进行读，而以模式B进行写)。

模式1 —— SRAM/CRAM

图158 模式1读操作

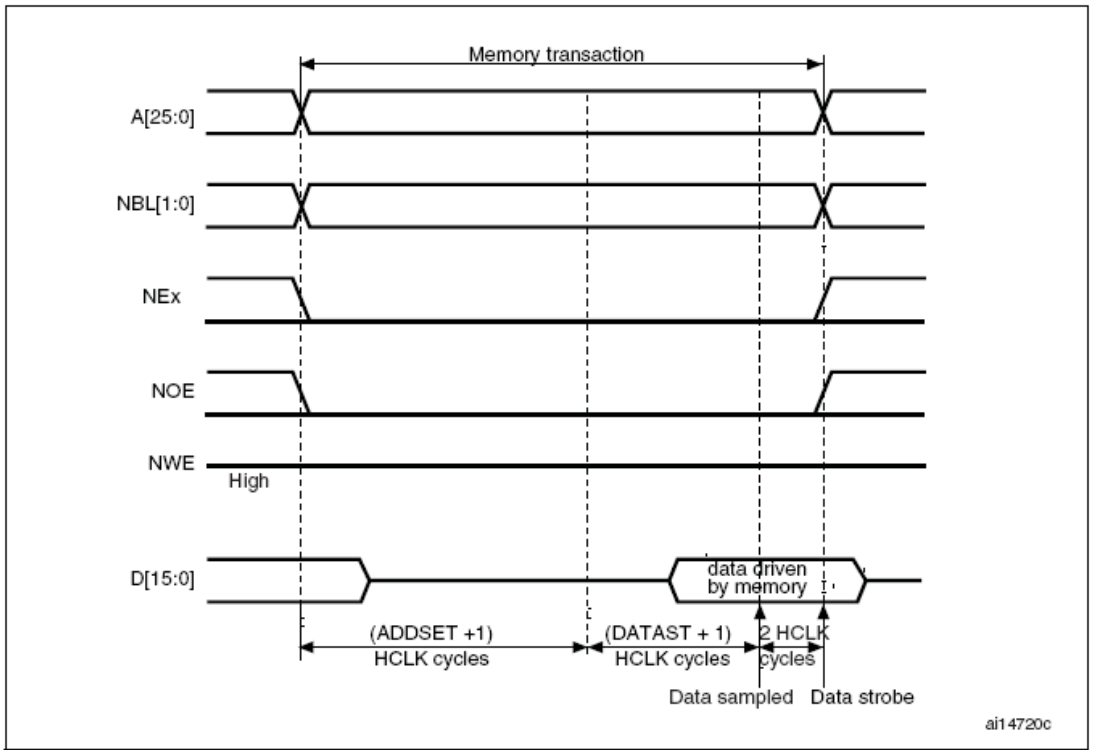
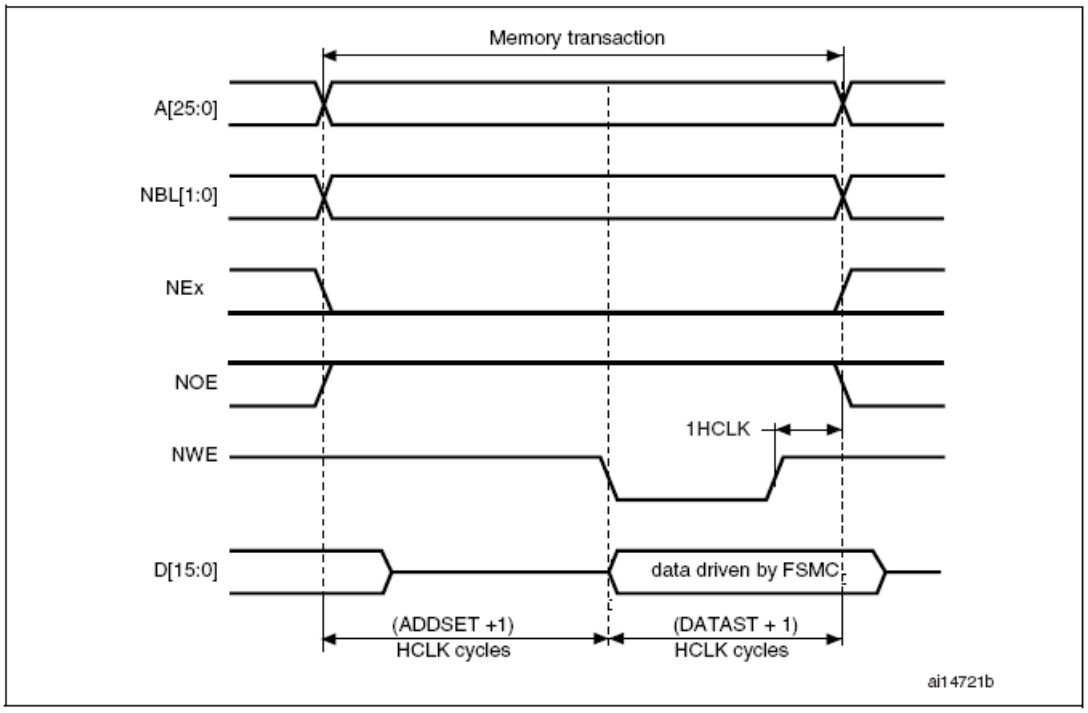


图159 模式1写操作



在写操作的最后一个HCLK周期可以保证NEW上升沿后地址和数据的保持时间，因为存在这个HCLK周期，DATAST的数值必须大于0(DATAST > 0)。

表76 FSMC_BCRx位域(模式1)

位编号	位名称	设置的数值
31-15		0x0000
14-10		0x0
9	WAITPOL	仅当位15为'1'时有意义。
8	BURSTEN	0x0
7		-
6	FACCEN	-
5-4	MWID	需要时设置
3-2	MTYP	需要时设置，不包含10(NOR闪存)
1	MUXEN	0x0
0	MBKEN	0x1

表77 FSMC_TCRx位域(模式1)

位编号	位名称	设置的数值
31-16		0x0000
15-8	DATAST	操作的第2个阶段的长度，写操作为(DATAST+1个HCLK周期)，读操作为(DATAST+3个HCLK周期)。这个域不能为0，至少为1。
7-4		0x0
3-0	ADDSET	操作的第1个阶段的长度(ADDSET+1个HCLK周期)。

模式A —— SRAM/PSRAM(CRAM) OE翻转

图160 模式A读操作

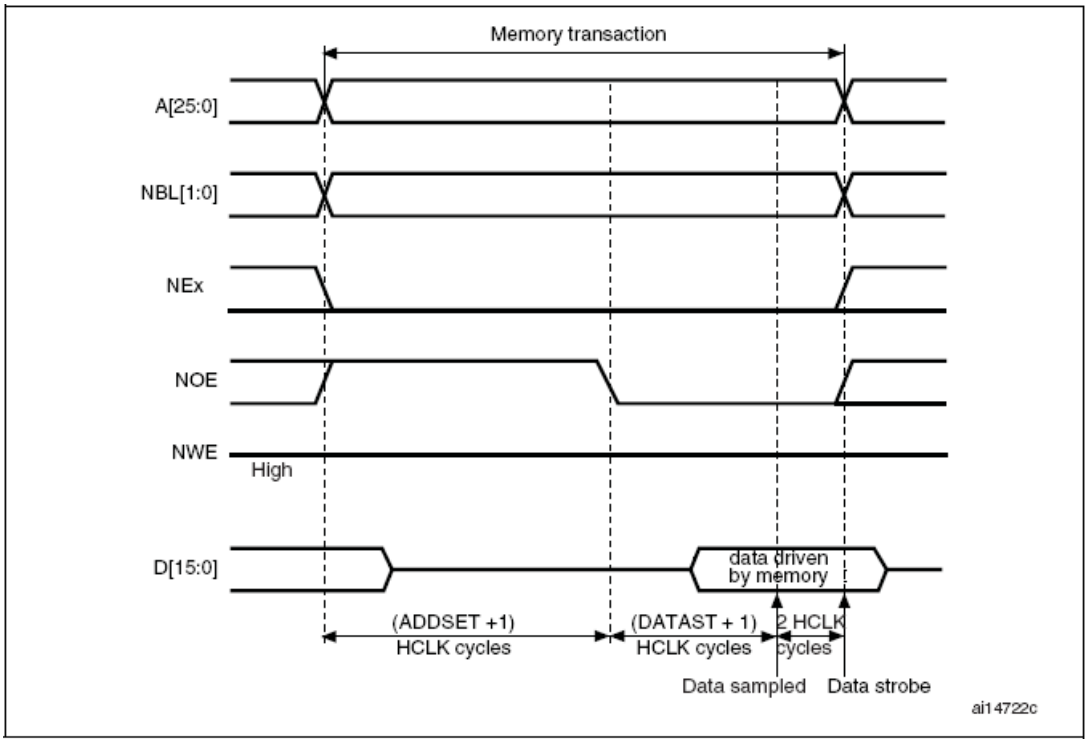
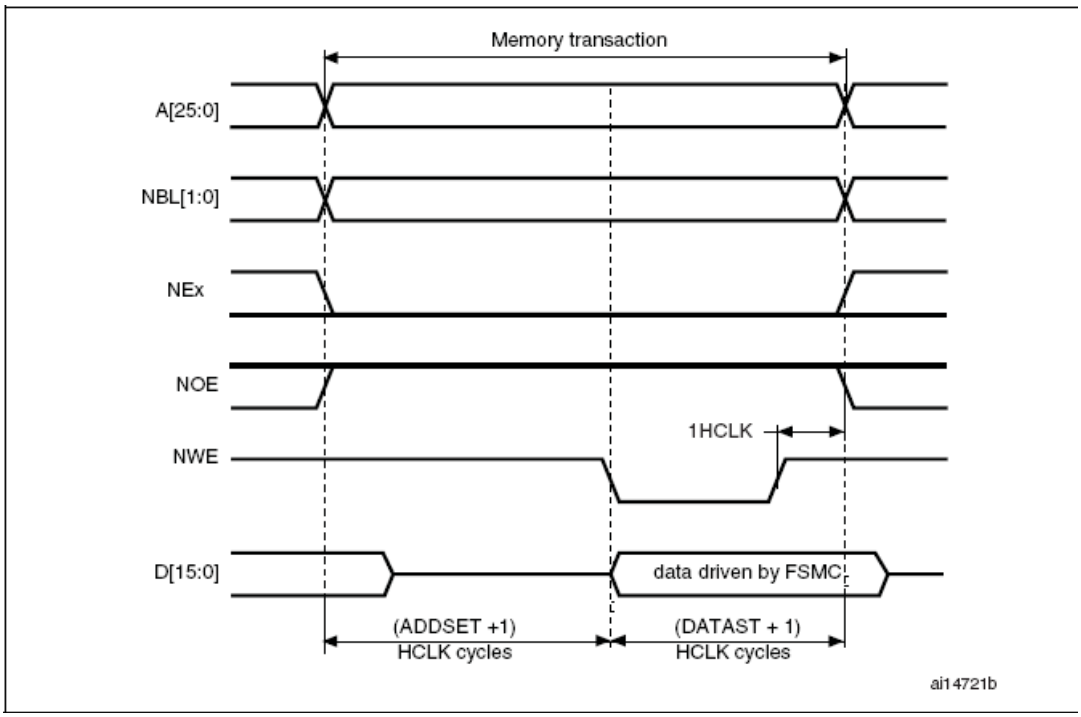


图161 模式A写操作



模式A与模式1的区别是NOE的变化和相互独立的读写时序。

表78 FSMC_BCRx位域(模式A)

位编号	位名称	设置的数值
31-16		0x0000
15		0x0
14	EXTMOD	0x1
13-10		0x0
9	WAITPOL	仅当位15为'1'时有意义。
8	BURSTEN	0x0
7		-
6	FACCEN	-
5-4	MWID	需要时设置
3-2	MTYP	需要时设置，不包含10(NOR闪存)
1	MUXEN	0x0
0	MBKEN	0x1

表79 FSMC_TCRx位域(模式A)

位编号	位名称	设置的数值
31-30		0x0
29-28	ACCMOD	0x0
27-16		0x000
15-8	DATAST	读操作的第2个阶段的长度(DATAST+3个HCLK周期)。这个域不能为0(至少为1)。
7-4		0x0
3-0	ADDSET	读操作的第1个阶段的长度(ADDSET+1个HCLK周期)。

表80 FSMC_BWTRx位域(模式A)

位编号	位名称	设置的数值
31-30		0x0
29-28	ACCMOD	0x0
27-16		0x000
15-8	DATAST	写操作的第2个阶段的长度(DATAST+1个HCLK周期)。这个域不能为0(至少为1)。
7-4		0x0
3-0	ADDSET	写操作的第1个阶段的长度(ADDSET+1个HCLK周期)。

模式2/B —— NOR闪存

图162 模式2/B读操作

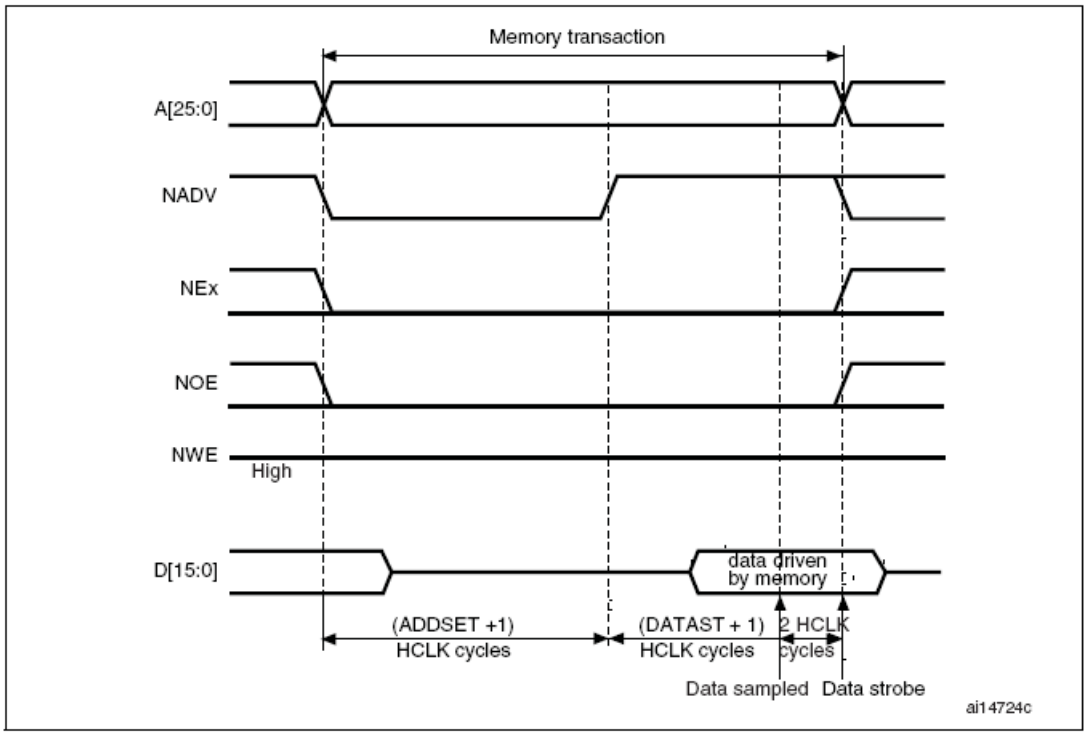


图163 模式2写操作

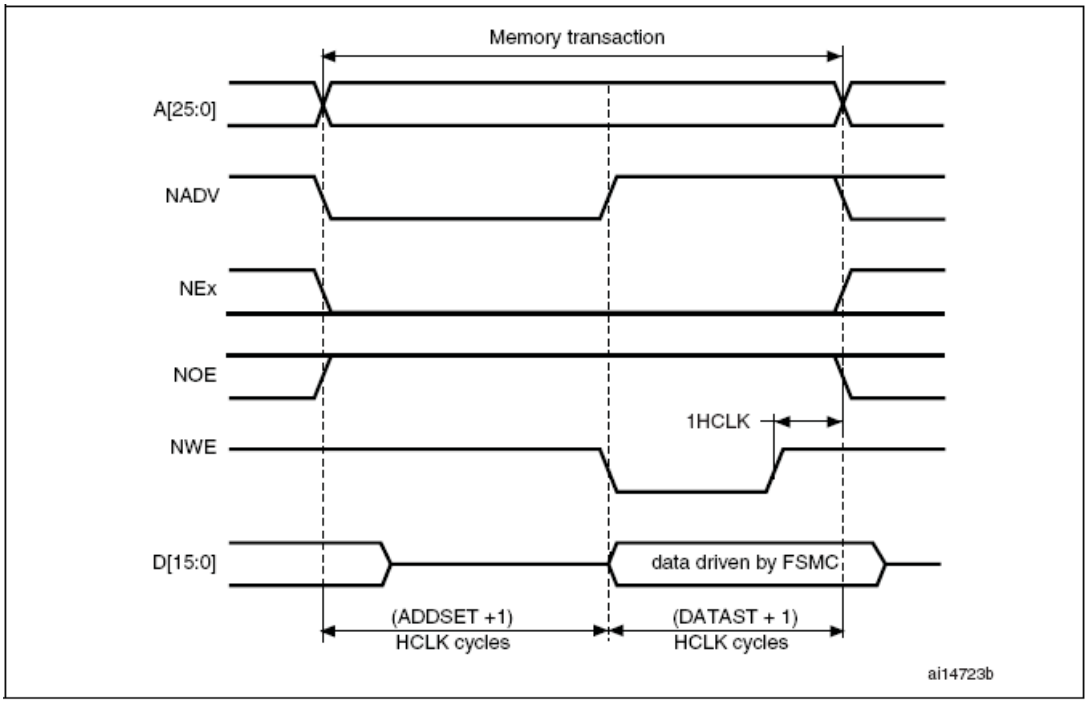
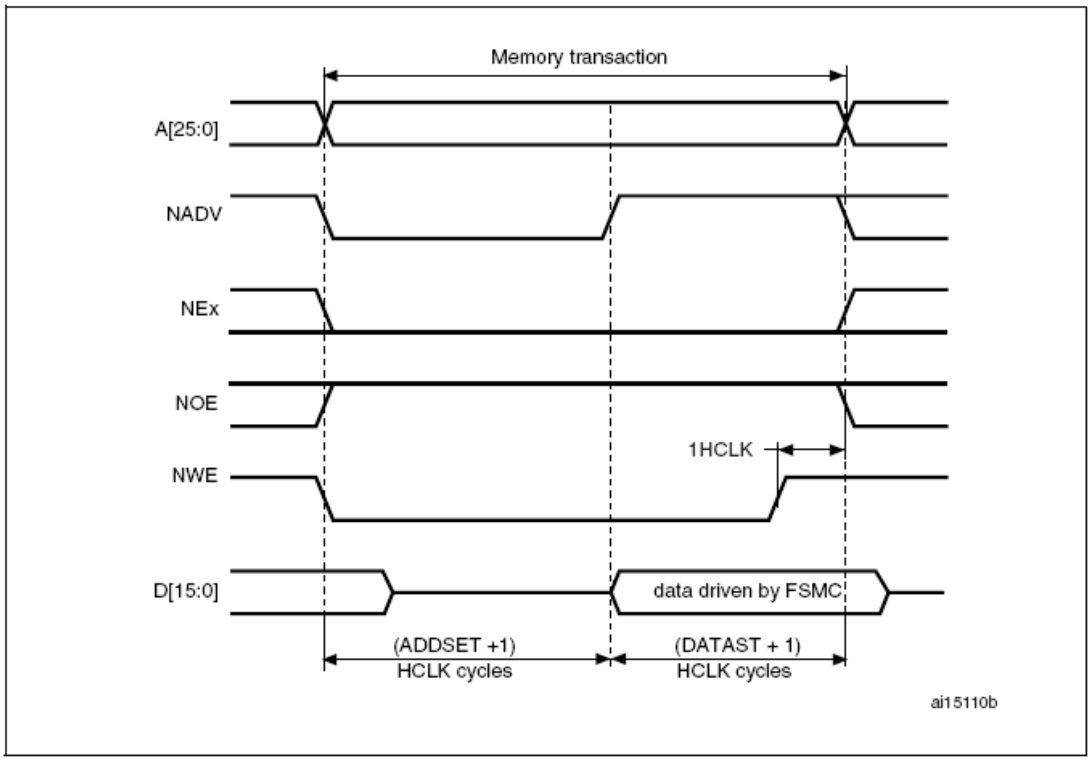


图164 模式B写操作



模式2/B与模式1相比较，不同的是NADV的变化，且在扩展模式下(模式B)读写时序相互独立。

表81 FSMC_BCRx位域(模式2/B)

位编号	位名称	设置的数值
31-15		0x0000
14	EXTMOD	模式B：0x1；模式2：0x0
13-10		0x0
9	WAITPOL	仅当位15为'1'时有意义。
8	BURSTEN	0x0
7		-
6	FACCEN	0x1
5-4	MWID	需要时设置
3-2	MTYP	10(NOR闪存)
1	MUXEN	0x0
0	MBKEN	0x1

表82 FSMC_TCRx位域(模式2/B)

位编号	位名称	设置的数值
31-30		0x0
29-28	ACCMOD	如果使用了扩展模式，设置为0x1
27-16		0x000
15-8	DATAST	读操作的第2个阶段的长度(DATAST+3个HCLK周期)。这个域不能为0(至少为1)。
7-4		0x0
3-0	ADDSET	读操作的第1个阶段的长度(ADDSET+1个HCLK周期)。

表83 FSMC_BWTRx位域(模式2/B)

位编号	位名称	设置的数值
31-30		0x0
29-28	ACCMOD	如果使用了扩展模式，设置为0x1
27-16		0x000
15-8	DATAST	写操作的第2个阶段的长度(DATAST+1个HCLK周期)。这个域不能为0(至少为1)。
7-4		0x0
3-0	ADDSET	写操作的第1个阶段的长度(ADDSET+1个HCLK周期)。

注: 只有当设置了扩展模式时(模式B)，FSMC_BWTRx才有效，否则该寄存器的内容不起作用。

模式C —— NOR闪存 - OE翻转

图165 模式C读操作

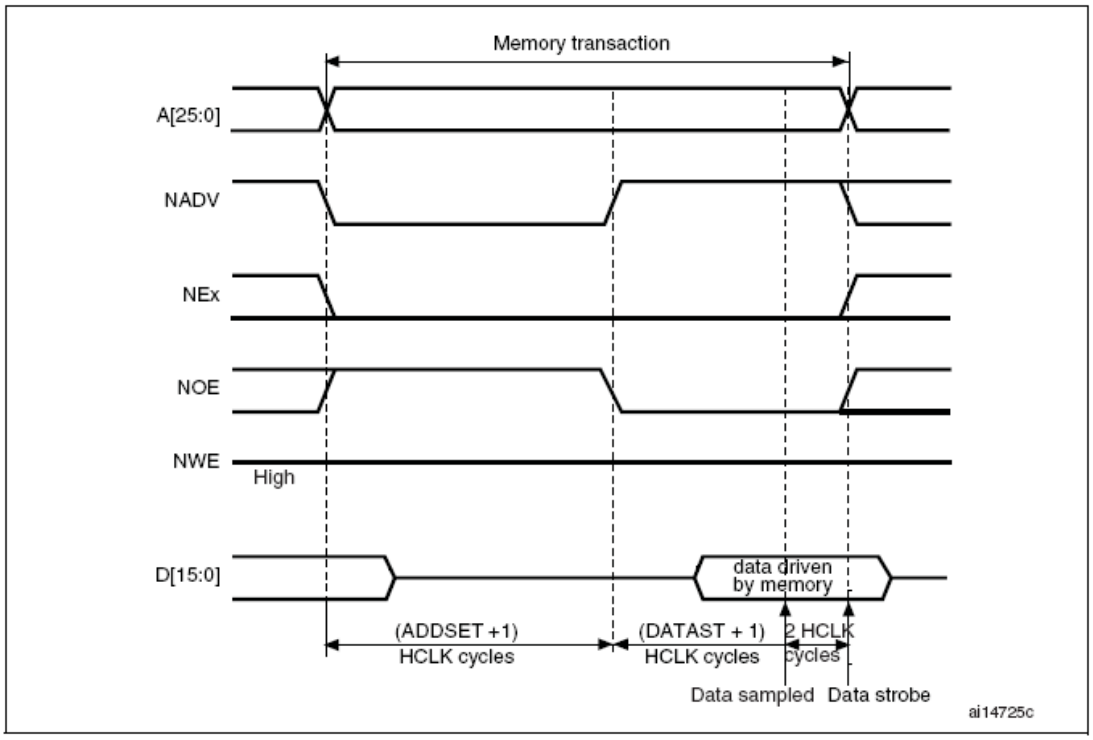
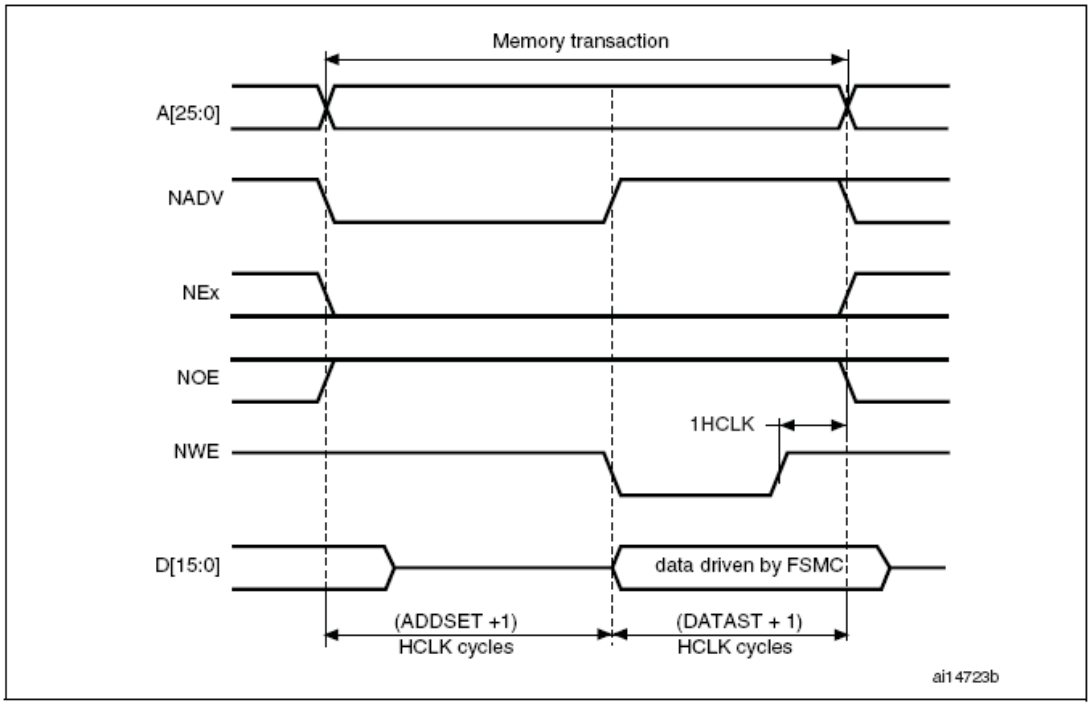


图166 模式C写操作



模式C与模式1不同的是，NOE和NADV的翻转变化，以及独立的读写时序。

表84 FSMC_BCRx位域(模式C)

位编号	位名称	设置的数值
31-15		0x0000
14	EXTMOD	0x1
13-10		0x0
9	WAITPOL	仅当位15为'1'时有意义。
8	BURSTEN	0x0
7		-
6	FACCEN	0x1
5-4	MWID	需要时设置
3-2	MTYP	0x2(NOR闪存)
1	MUXEN	0x0
0	MBKEN	0x1

表85 FSMC_TCRx位域(模式C)

位编号	位名称	设置的数值
31-30		0x0
29-28	ACCMOD	0x2
27-16		0x000
15-8	DATAST	读操作的第2个阶段的长度(DATAST+3个HCLK周期)。这个域不能为0(至少为1)。
7-4		0x0
3-0	ADDSET	读操作的第1个阶段的长度(ADDSET+1个HCLK周期)。

表86 FSMC_BWTRx位域(模式C)

位编号	位名称	设置的数值
31-30		0x0
29-28	ACCMOD	0x2
27-16		0x000
15-8	DATAST	写操作的第2个阶段的长度(DATAST+1个HCLK周期)。这个域不能为0(至少为1)。
7-4		0x0
3-0	ADDSET	写操作的第1个阶段的长度(ADDSET+1个HCLK周期)。

模式D —— 带地址扩展的异步操作

图167 模式D读操作

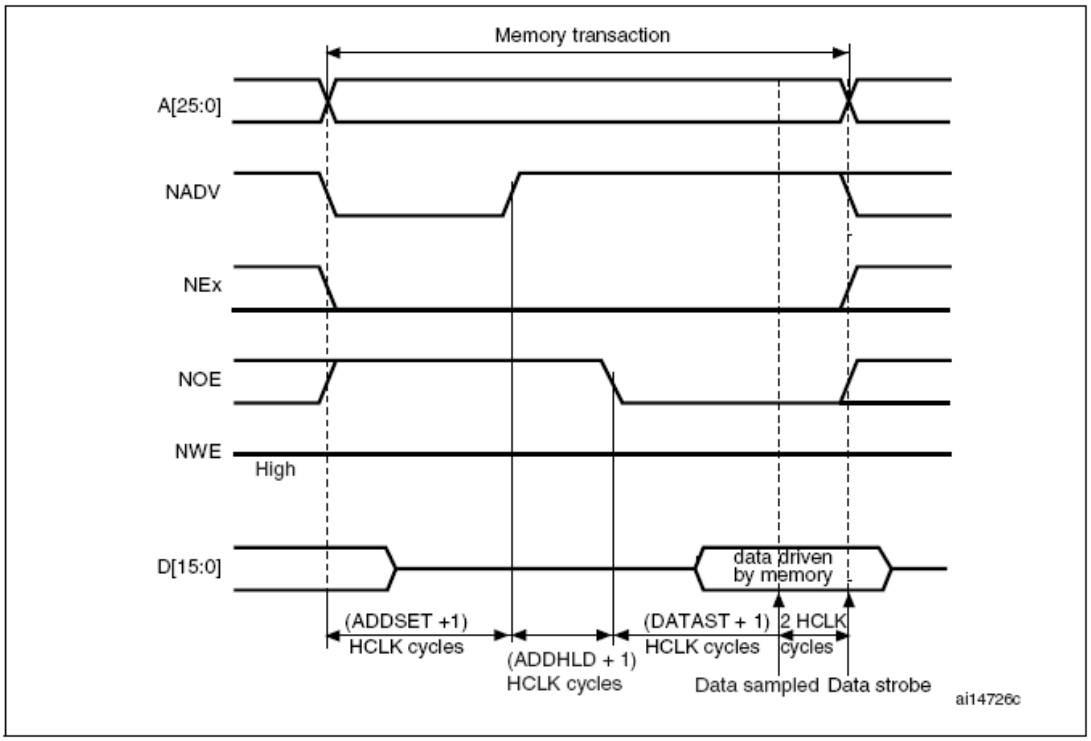
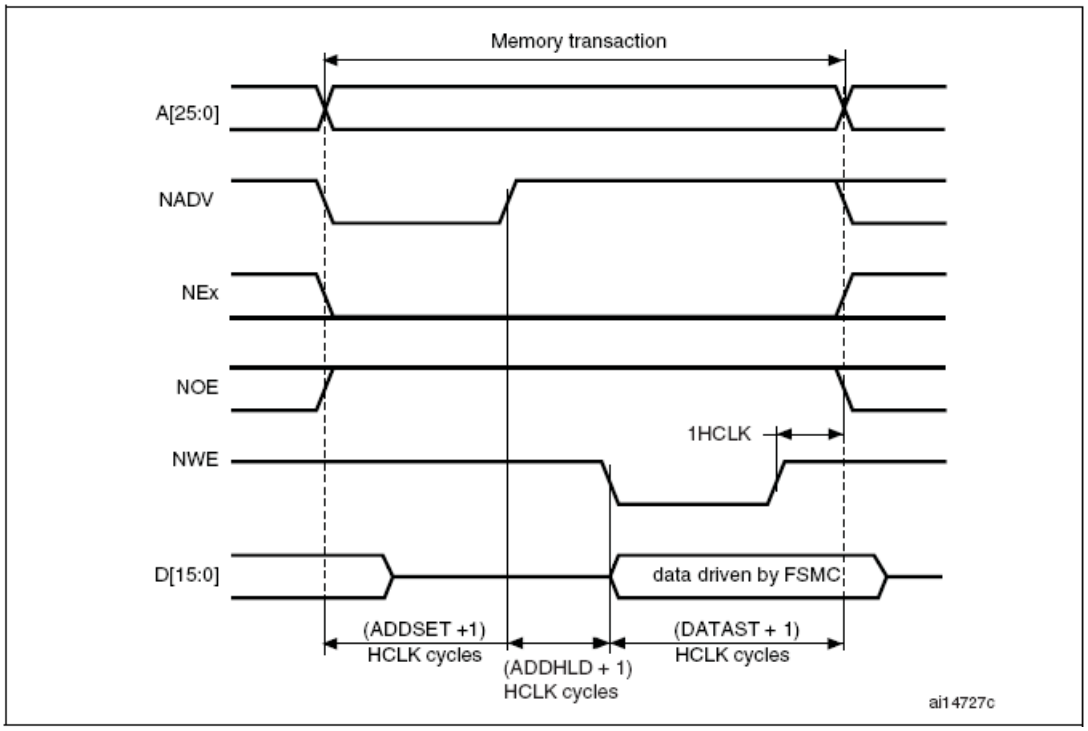


图168 模式D写操作



模式D与模式1不同的是NADV的翻转变化，NOE的翻转出现在NADV翻转之后，并且具有独立的读写时序。

表87 FSMC_BCRx位域(模式D)

位编号	位名称	设置的数值
31-15		0x0000
14	EXTMOD	0x1
13-10		0x0
9	WAITPOL	仅当位15为'1'时有意义。
8	BURSTEN	0x0
7		-
6	FACCEN	根据存储器设置
5-4	MWID	需要时设置
3-2	MTYP	需要时设置
1	MUXEN	0x0
0	MBKEN	0x1

表88 FSMC_TCRx位域(模式D)

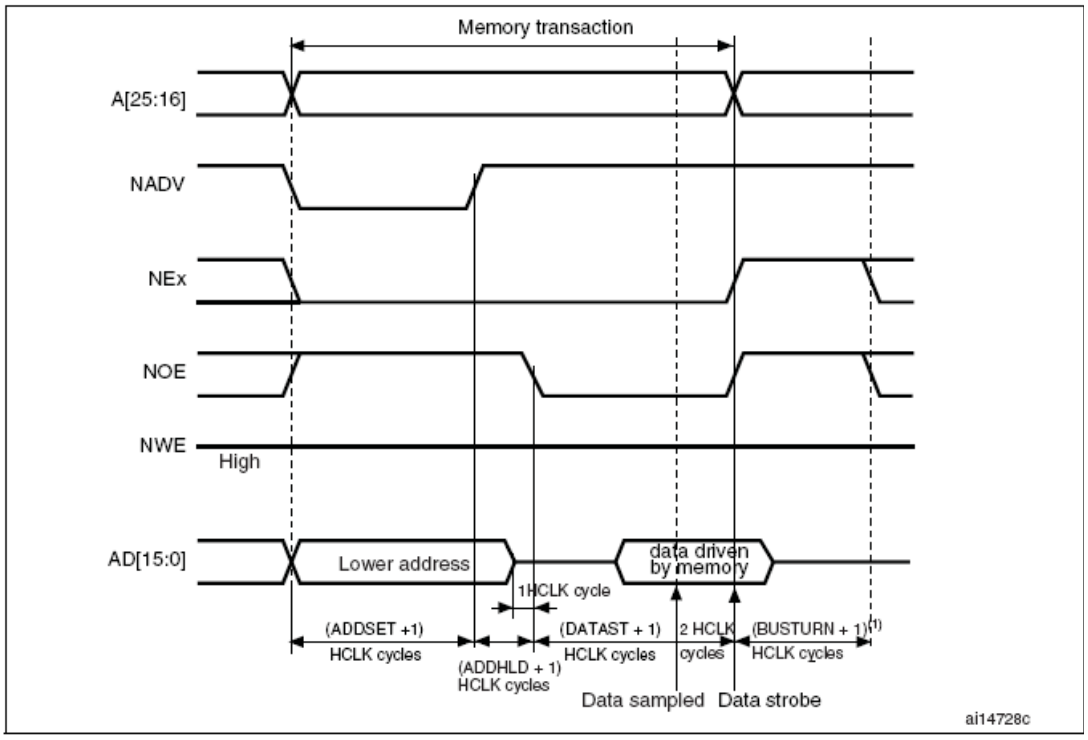
位编号	位名称	设置的数值
31-30		0x0
29-28	ACCMOD	0x2
27-16		0x000
15-8	DATAST	读操作的第2个阶段的长度(DATAST+3个HCLK周期)。这个域不能为0(至少为1)。
7-4	ADDHLD	读操作的中间阶段的长度(ADDHLD+1个HCLK周期)。
3-0	ADDSET	读操作的第1个阶段的长度(ADDSET+1个HCLK周期)。

表89 FSMC_BWTRx位域(模式D)

位编号	位名称	设置的数值
31-30		0x0
29-28	ACCMOD	0x2
27-16		0x000
15-8	DATAST	写操作的第2个阶段的长度(DATAST+1个HCLK周期)。这个域不能为0(至少为1)。
7-4	ADDHLD	写操作的中间阶段的长度(ADDHLD+1个HCLK周期)。
3-0	ADDSET	写操作的第1个阶段的长度(ADDSET+1个HCLK周期)。

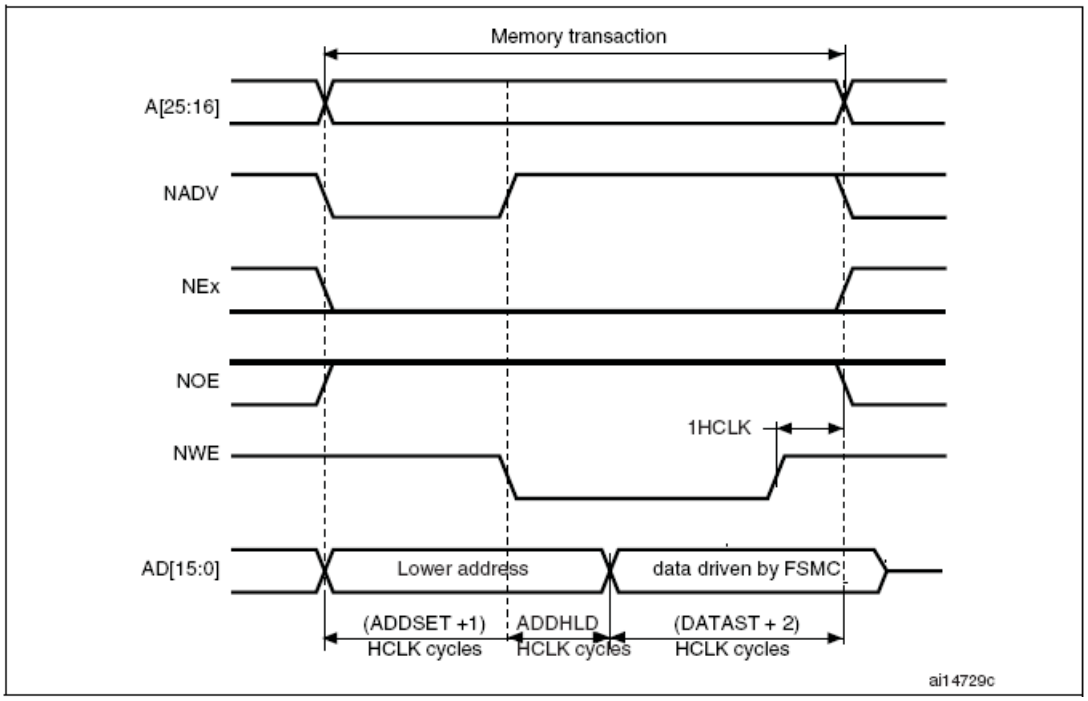
复用模式 —— 地址/数据复用的NOR闪存异步操作

图169 复用读操作



(1) 总线恢复延迟(BUSTURN+1)与连续 2 次读操作之间在内部产生的延迟有部分重叠，因此 BUSTURN≤5 时将不影响输出时序。

图170 复用写操作



复用模式与模式D不同的是地址的低16位出现在数据总线上。

表90 FSMC_BCRx位域(复用模式)

位编号	位名称	设置的数值
31-15		0x0000
14	EXTMOD	0x0
13-10		0x0
9	WAITPOL	仅当位15为'1'时有意义。
8	BURSTEN	0x0
7		-
6	FACCEN	0x1
5-4	MWID	需要时设置
3-2	MTYP	0x2(NOR闪存)
1	MUXEN	0x1
0	MBKEN	0x1

表91 FSMC_TCRx位域(复用模式)

位编号	位名称	设置的数值
31-30		0x0
29-20		
19-16	BUSTURN	操作的最后阶段的长度(BUSTURN+1个HCLK周期)。
15-8	DATAST	操作的第2个阶段的长度，读操作为(DATAST+3个HCLK周期)，写操作为(DATAST+1个HCLK周期)。这个域不能为0(至少为1)。
7-4	ADDHLD	操作的中间阶段的长度(ADDHLD+1个HCLK周期)。这个域不能为0(至少为1)。
3-0	ADDSET	操作的第1个阶段的长度(ADDSET+1个HCLK周期)。

18.5.5 同步的成组读

根据参数CLKDIV的不同数值，存储器时钟CLK的周期是HCLK的整数倍。

NOR闪存存储器有一个从NADV有效至CLK变高的最小时间限制，为了满足这个限制，在同步访问(NADV有效之前)的第一个内部时钟周期中，FSMC不会输出时钟到存储器。这样可以保证存储器时钟的上升沿产生于NADV低脉冲的中间。

数据延时与NOR闪存的延时

数据延时是指在采样数据之前需等待的周期数目，DATLAT数值必须与NOR闪存配置寄存器中定义的数值相符合。FSMC不把NADV为低时的时钟周期包含在数据延时这个参数中。

注意： *有些NOR闪存把NADV为低时的时钟周期包含在数据延时这个参数中，因此NOR闪存延时与FSMC的DATLAT参数的关系可以是：*

- NOR闪存延时 = DATLAT + 2；或
- NOR闪存延时 = DATLAT + 3

有些新出的存储器会在数据保持阶段产生一个NWAIT信号，这种情况下可以设置DATLAT为其最小值。FSMC会对NWAIT信号采样并等待足够长的时间直到数据有效，在FSMC检测到存储器结束了保持阶段后，读取正确的数据。

另外一些存储器不在数据保持阶段输出NWAIT信号，这时FSMC和存储器端的数据保持时间必须设置为相同的数值，否则将得不到正确的数据，或在存储器访问的初始阶段会有数据丢失。

单次成组传输

当选中的存储器块配置为同步成组模式，如果仅需要进行一次AHB单次成组传输，因为AHB需要传输16位数据，FSMC会执行一次长度为1的成组传输；因为AHB需要传输32位数据，FSMC会分成2次16位传输，执行一次长度为2的成组传输；最后一个数据传输完毕时撤消片选信号。

显然，从效率上讲这种传输方式(与异步读相比)不是最有效的；但是一次随机的异步访问需要重新配置存储器访问模式，这同样需要较长时间。

等待管理

对于同步的NOR闪存成组访问，在预置的保持时间(DATLAT+1个CLK时钟周期)之后，需检测NWAIT信号。

如果检测到NWAIT为有效电平时(当WAITPOL=0时有效电平为低，WAITPOL=1时有效电平为高)，FSMC将插入等待周期直到NWAIT变为无效电平(当WAITPOL=0时无效电平为高，WAITPOL=1时无效电平为低)。

当NWAIT变为无效时，FSMC认为数据已经有效(WAITCFG=1)，或数据将在下一个时钟边沿有效(WAITCFG=0)。

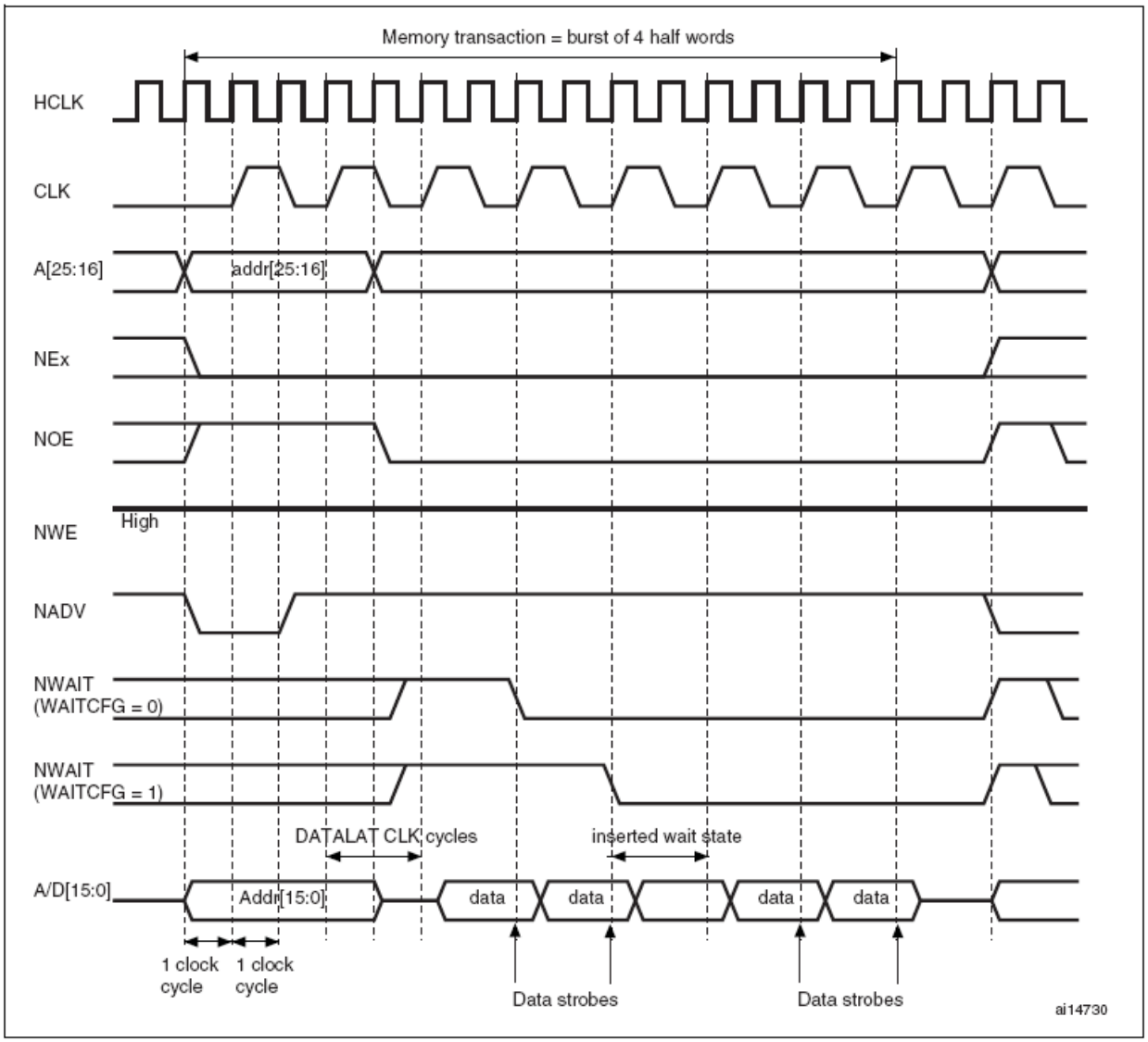
在NWAIT信号控制的等待状态插入期间，控制器会连续地向存储器发送时钟脉冲、保持片选信号和输出有效信号，同时忽略无效的数据信号。

在成组传输模式下，NOR闪存的NWAIT信号有2种时序配置：

- 闪存存储器在等待状态之前的一个数据周期插入NWAIT信号(复位后的默认设置)；
- 闪存存储器在等待状态期间插入NWAIT信号。

通过配置FSMC_BCRx寄存器中的WAITCFG位，FSMC在每个片选上都支持这2种NOR闪存的等待状态配置。

图171 同步读模式 – NOR, PSRAM(CRAM)



BL信号没有显示在图中，对于NOR闪存BL应该为高；对于PSRAM(CRAM)，BL应该为低。

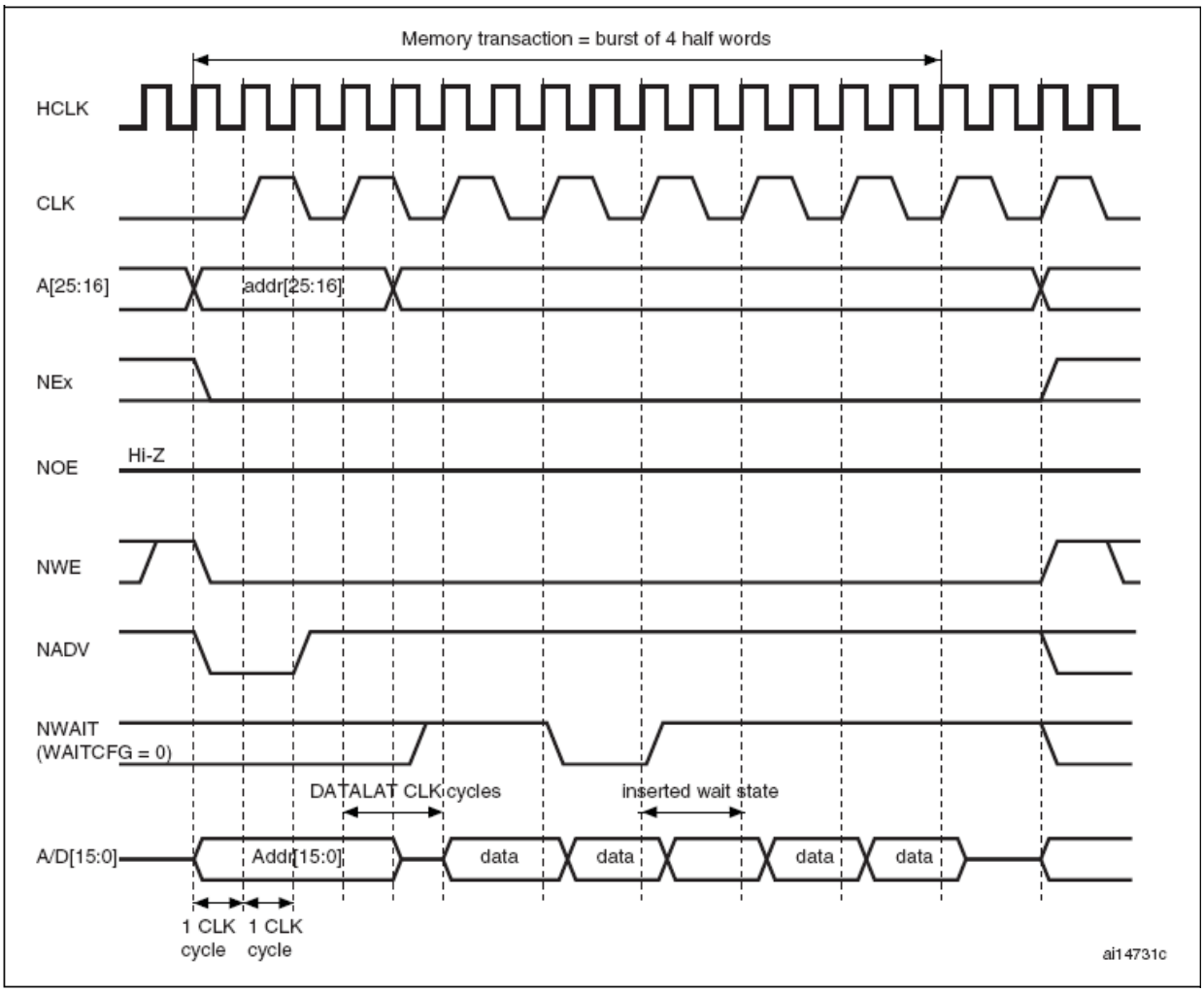
表92 FSMC_BCRx位域(同步模式)

位编号	位名称	设置的数值
31-20		0x0000
19	CBURSTRW	对同步读模式不起作用
18-15		0x0
14	EXTMOD	0x0
13	WAITEN	当此位为高时，不管存储器的等待数值是多少，FSMC视数据保持阶段结束后的第一个数据有效。
12	WREN	对同步读模式不起作用
11	WAITCFG	根据存储器特性设置
10	WRAPMOD	根据存储器特性设置
9	WAITPOL	根据存储器特性设置
8	BURSTEN	0x1
7	FWPRLVL	设置此位防止存储器被意外写
6	FACCEN	根据存储器设置
5-4	MWID	根据需要设置
3-2	MTYP	0x1 或 0x2
1	MUXEN	根据需要设置
0	MBKEN	0x1

表93 FSMC_TCRx位域(同步模式)

位编号	位名称	设置的数值
27-24	DATLAT	数据保持时间
23-20	CLKDIV	0x0 – 得到CLK = HCLK(不支持) 0x1 – 得到CLK = 2 x HCLK
19-16	BUSTURN	不起作用
15-8	DATAST	不起作用
7-4	ADDHLD	不起作用
3-0	ADDSET	不起作用

图172 同步写模式 – PSRAM(CRAM)



(1) 存储器必须提前一个周期产生NWAIT信号，同时WAITCFG应配置为0。

(2) 字节选择BL输出没有显示在图中，当NEx为有效时它们为低。

表94 FSMC_BCRx位域(同步模式)

位编号	位名称	设置的数值
31-20		0x0000
19	CBURSTRW	0x1
18-15		0x0
14	EXTMOD	0x0
13	WAITEN	当此位为高时，不管存储器的等待数值是多少，FSMC视数据保持阶段结束后的第一个数据有效。
12	WREN	对同步读模式不起作用
11	WAITCFG	0x0
10	WRAPMOD	根据存储器特性设置
9	WAITPOL	根据存储器特性设置
8	BURSTEN	对同步写模式不起作用
7	FWPRLVL	设置此位防止存储器被意外写
6	FACCEN	根据存储器设置
5-4	MWID	根据需要设置
3-2	MTYP	0x1
1	MUXEN	根据需要设置

0	MBKEN	0x1

表95 FSMC_TCRx位域(同步模式)

位编号	位名称	设置的数值
31-30	-	0x0
27-24	DATLAT	数据保持时间
23-20	CLKDIV	0x0 – 得到CLK = HCLK(不支持) 0x1 – 得到CLK = 2 x HCLK
19-16	BUSTURN	不起作用
15-8	DATAST	不起作用
7-4	ADDHLD	不起作用
3-0	ADDSET	不起作用

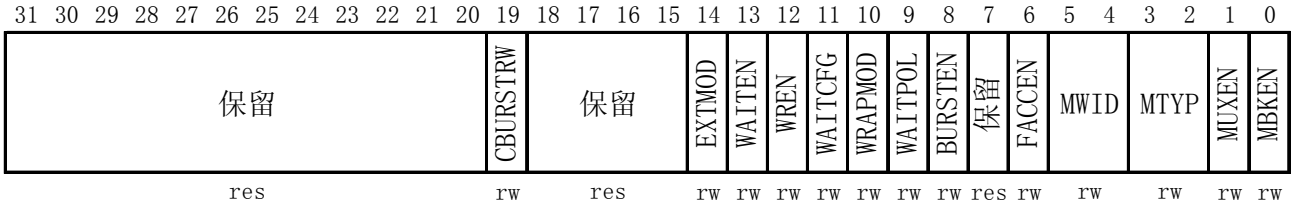
18.5.6 NOR闪存和PSRAM控制器寄存器

SRAM/NOR闪存片选控制寄存器 1…4 (FSMC_BCR1…4)

地址偏移：0xA000 0000 + 8 * (x-1), x=1…4

复位值：0x0000 30DX

这个寄存器包含了每个存储器块的控制信息，可以用于SRAM、ROM、异步或成组传输的NOR闪存存储器。



位19	**CBURSTRW**：成组写使能位 对于Cellular RAM，该位使能写操作的同步成组传输协议。 对于处于成组传输模式的闪存存储器，这一位允许/禁止通过NWAIT信号插入等待状态。读操作的同步成组传输协议使能位是FSMC_BCRx寄存器的BURSTEN位。 0：写操作始终处于异步模式 1：写操作为同步模式
位14	**EXTMOD**：扩展模式使能 该位允许FSMC使用FSMC_BWTR寄存器，即允许读和写使用不同的时序。 0：不使用FSMC_BWTR寄存器，这是复位后的默认状态。 1：FSMC使用FSMC_BWTR寄存器。
位13	**WAITEN**：等待使能位 当闪存存储器处于成组传输模式时，这一位允许/禁止通过NWAIT信号插入等待状态。 0：禁用NWAIT信号，在设置的闪存保持周期之后不会检测NWAIT信号插入等待状态。 1：使用NWAIT信号，在设置的闪存保持周期之后根据NWAIT信号插入等待状态；这是复位后的默认状态。
位12	**WREN**：写使能位 该位指示FSMC是否允许/禁止对存储器的写操作。 0：禁止FSMC对存储器的写操作，否则产生一个AHB错误。 1：允许FSMC对存储器的写操作；这是复位后的默认状态。

位11	**WAITCFG：**配置等待时序 当闪存存储器处于成组传输模式时，NWAIT信号指示从闪存存储器出来的数据是否有效或是否需要插入等待周期。该位决定存储器是在等待状态之前的一个时钟周期产生NWAIT信号，还是在等待状态期间产生NWAIT信号。 0：NWAIT信号在等待状态前的一个数据周期有效；这是复位后的默认状态。 1：NWAIT信号在等待状态期间有效(不适用于Cellular RAM)。
位10	**WRAPMOD：**支持非对齐的成组模式 该位决定控制器是否支持把非对齐的AHB成组操作分割成2次线性操作；该位仅在存储器的成组模式下有效。 0：不允许直接的非对齐成组操作；这是复位后的默认状态。 1：允许直接的非对齐成组操作。
位9	**WAITPOL：**等待信号极性 设置存储器产生的等待信号的极性；该位仅在存储器的成组模式下有效。 0：NWAIT等待信号为低时有效；这是复位后的默认状态。 1：NWAIT等待信号为高时有效。
位8	**BURSTEN：**成组模式使能 允许对闪存存储器进行成组模式访问；该位仅在闪存存储器的同步成组模式下有效。 0：禁用成组访问模式；这是复位后的默认状态。 1：使用成组访问模式。
位7	保留。
位6	**FACCEN：**闪存访问使能 允许对NOR闪存存储器的访问操作。 0：禁止对NOR闪存存储器的访问操作. 1：允许对NOR闪存存储器的访问操作。
位5:4	**MWID：**存储器数据总线宽度 定义外部存储器总线的宽度，适用于所有类型的存储器。 00：8位， 01：16位(复位后的默认状态)， 10：保留，不能用 11：保留，不能用
位3:2	**MTYP：**存储器类型 定义外部存储器的类型： 00：SRAM、ROM(存储器块2...4在复位后的默认值) 01：PSRAM(Cellular RAM: CRAM) 10：NOR闪存(存储器块1在复位后的默认值) 11：保留
位1	**MUXEN：**地址/数据复用使能位 当设置了该位后，地址的低16位和数据将共用数据总线，该位仅对NOR和PSRM存储器有效。 0：地址/数据不复用。 1：地址/数据复用数据总线；这是复位后的默认状态。
位0	**MBKEN：**存储器块使能位 开启对应的存储器块。复位后存储器块1是开启的，其它所有存储器块为禁用。访问一个禁用的存储器块将在AHB总线上产生一个错误。 0：禁用对应的存储器块。 1：启用对应的存储器块。

SRAM/NOR闪存片选时序寄存器 1...4 (FSMC_BTR1...4)

地址偏移：0xA000 0000 + 0x04 + 8 * (x-1), x=1...4

复位值：0x0FFF FFFF

这个寄存器包含了每个存储器块的控制信息，可以用于SRAM、ROM和NOR闪存存储器。如果FSMC_BCRx寄存器中设置了EXTMOD位，则有两个时序寄存器分别对应读(本寄存器)和写操作(FSMC_BWTRx寄存器)。

31 30	29 28	27 26 25 24	23 22 21 20	19 18 17 16	15 14 13 12 11 10 9 8	7 6 5 4	3 2 1 0
保留	ACCMOD	DATLAT	CLKDIV	BUSTURN	DATAST	ADDHLD	ADDSET
res	rw	rw	rw	rw	rw	rw	rw

位29:28	**ACCMOD：**访问模式 定义异步访问模式。这2位只在FSMC_BCRx寄存器的EXTMOD位为1时起作用。 00：访问模式A 01：访问模式B 10：访问模式C 11：访问模式D
位27:24	**DATLAT**(见本表格下面的注释)：(同步成组式NOR闪存的)数据保持时间 处于同步成组模式的NOR闪存，需要定义在读取第一个数据之前等待的存储器周期数目。 这个时间参数不是以HCLK表示，而是以闪存时钟(CLK)表示。在访问异步NOR闪存、SRAM或ROM时，这个参数不起作用。操作CRAM时，这个参数必须为0。 0000：第一个数据的保持时间为2个CLK时钟周期 …… 1111：第一个数据的保持时间为17个CLK时钟周期(这是复位后的默认数值)。
位23:20	**CLKDIV：**时钟分频比(CLK信号) 定义CLK时钟输出信号的周期，以HCLK周期数表示： 0000：保留 0001：1个CLK周期=2个HCLK周期 0010：1个CLK周期=3个HCLK周期 …… 1111：1个CLK周期=16个HCLK周期(这是复位后的默认数值)。 在访问异步NOR闪存、SRAM或ROM时，这个参数不起作用。
位19:16	**BUSTURN：**总线恢复时间 这些位用于定义一次读操作之后在总线上的延迟(仅适用于总线复用模式的NOR闪存操作)，一次读操作之后控制器需要在数据总线上为下次操作送出地址，这个延迟就是为了防止总线冲突。如果扩展的存储器系统不包含总线复用模式的存储器，或最慢的存储器可以在6个HCLK时钟周期内将数据总线恢复到高阻状态，可以设置这个参数为其最小值。 0000：总线恢复时间=1个HCLK时钟周期 …… 1111：总线恢复时间=16个HCLK时钟周期(这是复位后的默认数值)。
位15:8	**DATAST：**数据保持时间 这些位定义数据的保持时间(见图158至图170)，适用于SRAM、ROM和异步总线复用模式的NOR闪存操作。 0000 0000：DATAST保持时间=1个HCLK时钟周期 …… 0000 1111：DATAST保持时间=16个HCLK时钟周期(这是复位后的默认数值)。

位7:4	**ADDHLD**：地址保持时间 这些位定义地址的保持时间(见图167至图170)，适用于SRAM、ROM和异步总线复用模式的NOR闪存操作。 0000：ADDHLD保持时间=1个HCLK时钟周期 …… 1111：ADDHLD保持时间=16个HCLK时钟周期(这是复位后的默认数值)。 注：在同步操作中，这个参数不起作用，地址保持时间始终是1个存储器时钟周期。
位3:0	**ADDSET**：地址建立时间 这些位以HCLK周期数定义地址的建立时间(见图158至图170)，适用于SRAM、ROM和异步总线复用模式的NOR闪存操作。 0000：ADDSET建立时间=1个HCLK时钟周期 …… 1111：ADDSET建立时间=16个HCLK时钟周期(这是复位后的默认数值)。 注：在同步操作中，这个参数不起作用，地址建立时间始终是1个存储器时钟周期。

注： *因为内部的刷新，PSRAM(CRAM)具有可变的保持延迟，因此这样的存储器会在数据保持期间输出NWAIT信号以延长数据的保持时间。使用PSRAM(CRAM)时DATLAT域应置为0，这样FSMC可以及时地退出自己的保持阶段并开始对存储器发出的NWAIT信号进行采样，然后在存储器准备好时开始读或写操作。*

这个操作方式还可以用于操作最新的能够输出NWAIT信号的同步闪存存储器，详细信息请参考相应的闪存存储器手册。

SRAM/NOR闪存写时序寄存器 1…4 (FSMC_BWTR1…4)

地址偏移：0xA000 0000 + 0x104 + 8 * (x-1), x=1…4

复位值：0x0FFF FFFF

这个寄存器包含了每个存储器块的控制信息，可以用于SRAM、ROM和NOR闪存存储器。如果FSMC_BCRx寄存器中设置了EXTMOD位，则这个寄存器对应写操作。

31–30	29–28	27–26	25–24	23–22	21–20	19–18	17–16	15–14	13–12	11–10	9–8	7–6	5–4	3–2	1–0
保留	ACCMOD	DATLAT		CLKDIV		保留		DATAST				ADDHLD		ADDSET	
res	rw	rw		rw		res		rw				rw		rw	

位29:28	**ACCMOD**：访问模式 定义异步访问模式。这2位只在FSMC_BCRx寄存器的EXTMOD位为1时起作用。 00：访问模式A 01：访问模式B 10：访问模式C 11：访问模式D
位27:24	**DATLAT**：(NOR闪存的同步成组模式)数据保持时间 处于同步成组模式的NOR闪存，需要定义在读取第一个数据之前等待的存储器周期数目。 0000：第一个数据的保持时间为2个CLK时钟周期 …… 1111：第一个数据的保持时间为17个CLK时钟周期(这是复位后的默认数值)。 注：这个时间参数不是以HCLK表示，而是以闪存时钟(CLK)表示。 注：在访问异步NOR闪存、SRAM或ROM时，这个参数不起作用。 注：操作CRAM时，这个参数必须为0。

位23:20	**CLKDIV**：时钟分频比(CLK信号) 定义CLK时钟输出信号的周期，以HCLK周期数表示： 0000：保留 0001：1个CLK周期=2个HCLK周期 0010：1个CLK周期=3个HCLK周期 …… 1111：1个CLK周期=16个HCLK周期(这是复位后的默认数值)。 在访问异步NOR闪存、SRAM或ROM时，这个参数不起作用。
位19:16	保留
位15:8	**DATAST**：数据保持时间 这些位定义数据的保持时间(见图158至图170)，适用于SRAM、ROM和异步总线复用模式的NOR闪存操作。 0000 0000：DATAST保持时间=1个HCLK时钟周期 …… 0000 1111：DATAST保持时间=16个HCLK时钟周期(这是复位后的默认数值)。
位7:4	**ADDHLD**：地址保持时间 这些位定义地址的保持时间(见图167至图170)，适用于SRAM、ROM和异步总线复用模式的NOR闪存操作。 0000：ADDHLD保持时间=1个HCLK时钟周期 …… 1111：ADDHLD保持时间=16个HCLK时钟周期(这是复位后的默认数值)。 注：在同步NOR闪存操作中，这个参数不起作用，地址保持时间始终是1个闪存时钟周期。
位3:0	**ADDSET**：地址建立时间 这些位以HCLK周期数定义地址的建立时间(见图158至图170)，适用于SRAM、ROM和异步总线复用模式的NOR闪存操作。 0000：ADDSET建立时间=1个HCLK时钟周期 …… 1111：ADDSET建立时间=16个HCLK时钟周期(这是复位后的默认数值)。 注：在同步NOR闪存操作中，这个参数不起作用，地址建立时间始终是1个闪存时钟周期。

18.6 NAND闪存和PC卡控制器

FSMC可以为下列类型的设备产生合适的信号时序：

- NAND闪存
  - 8 位
  - 16 位
- 与16位PC卡兼容的设备

NAND/PC卡控制器能够控制3个外部存储块，存储块2和存储块3支持NAND闪存，存储块4支持PC卡设备。

每个存储块由专门的寄存器配置(见18.6.7节)，可编程的存储器参数包括操作时序和ECC配置。

表96 可编程NAND/PC卡操作参数

参数	功能	操作模式	单位	最小	最大
存储器建立时间	发出命令之前建立地址的(HCLK)时钟周期数目	读/写	AHB 时钟周期(HCLK)	1	256
存储器等待时间	发出命令的最短持续时间(HCLK周期数目)	读/写		1	256
存储器保持时间	在发送命令结束后保持地址的(HCLK)时钟周期数目，写操作时也是数据的保持时间	读/写		1	255
存储器数据总线高阻时间	启动写操作之后保持数据总线为高阻态的时间	写		0	255

18.6.1 外部存储器接口信号

下表列出了用于接口NAND闪存和PC卡的典型信号线。

注意： *当在I/O模式下使用PC卡或CF卡，在整个操作期间NIOS16输入管脚必须保持接地电平，否则FSMC可能不能正常操作。即NIOS16输入管脚一定不能连接到卡上，但可以直接接地(仅允许16位访问)。*

前缀'N'代表对应的信号线为低电平有效。

8位NAND闪存

表97 8位NAND闪存

FSMC信号名称	输入/输出	功能
A[17]	输出	NAND闪存地址锁存允许信号(ALE)
A[16]	输出	NAND闪存命令锁存允许信号(CLE)
D[7:0]	入/出	8位复用的、双向地址/数据总线
NCE[x]	输出	片选，x = 2，3
NOE(=NRE)	输出	输出使能(存储器端信号名称：读使能，NRE)
NWE	输出	写使能
NWAIT/INT[3:2]	输入	NAND闪存就绪/繁忙，输入至FSMC的信号

FSMC可以根据需要产生多个地址周期，理论上FSMC不限制可以访问的NAND容量。

16位NAND闪存

表98 16位NAND闪存

FSMC信号名称	输入/输出	功能
A[17]	输出	NAND闪存地址锁存允许信号(ALE)
A[16]	输出	NAND闪存命令锁存允许信号(CLE)
D[15:0]	入/出	16位复用的、双向地址/数据总线
NCE[x]	输出	片选，x = 2，3

NOE(=NRE)	输出	输出使能(存储器端信号名称：读使能，NRE)
NWE	输出	写使能
NWAIT/INT[3:2]	输入	NAND闪存就绪/繁忙，输入至FSMC的信号

FSMC可以根据需要产生多个地址周期，理论上FSMC不限制可以访问的NAND容量。

表99 16位PC卡

FSMC信号名称	输入/输出	功能
A[10:0]	输出	地址总线
NIOS16	输入	I/O空间的数据传输宽度(仅适合16位传输)
NIORD	输出	I/O空间的输出使能
NIOWR	输出	I/O空间的写使能
NREG	输出	指示操作是在通用或属性空间的寄存器信号
D[15:0]	入/出	双向数据总线
NCE4_1	输出	片选1
NCE4_2	输出	片选2(指示操作是16位还是8位)
NOE	输出	输出使能
NWE	输出	写使能
NWAIT	输入	进入FSMC的PC卡等待信号(存储器信号为IORDY)
INTR	输入	进入FSMC的PC卡中断信号(仅适合可以产生中断的PC卡)
CD	输入	PC卡存在的检测信号

18.6.2 NAND闪存/PC卡支持的存储器及其操作

下表列出了支持的设备、操作模式和操作方式。在表格中有阴影的部分表示NAND闪存/PC卡控制器不支持对应的操作方式。

表100 支持的存储器及其操作

存储器	模式	读/写	AHB数据宽度	存储器数据宽度	是否支持	注释
8位 NAND	异步	读	8	8	支持	
	异步	写	8	8	支持	
	异步	读	16	8	支持	分成2次FSMC访问
	异步	写	16	8	支持	分成2次FSMC访问
	异步	读	32	8	支持	分成4次FSMC访问
	异步	写	32	8	支持	分成4次FSMC访问
16位 NAND	异步	读	8	16	支持	
	异步	写	8	16	不支持	
	异步	读	16	16	支持	
	异步	写	16	16	支持	
	异步	读	32	16	支持	分成2次FSMC访问
	异步	写	32	16	支持	分成2次FSMC访问

18.6.3 NAND闪存、ATA和PC卡时序图

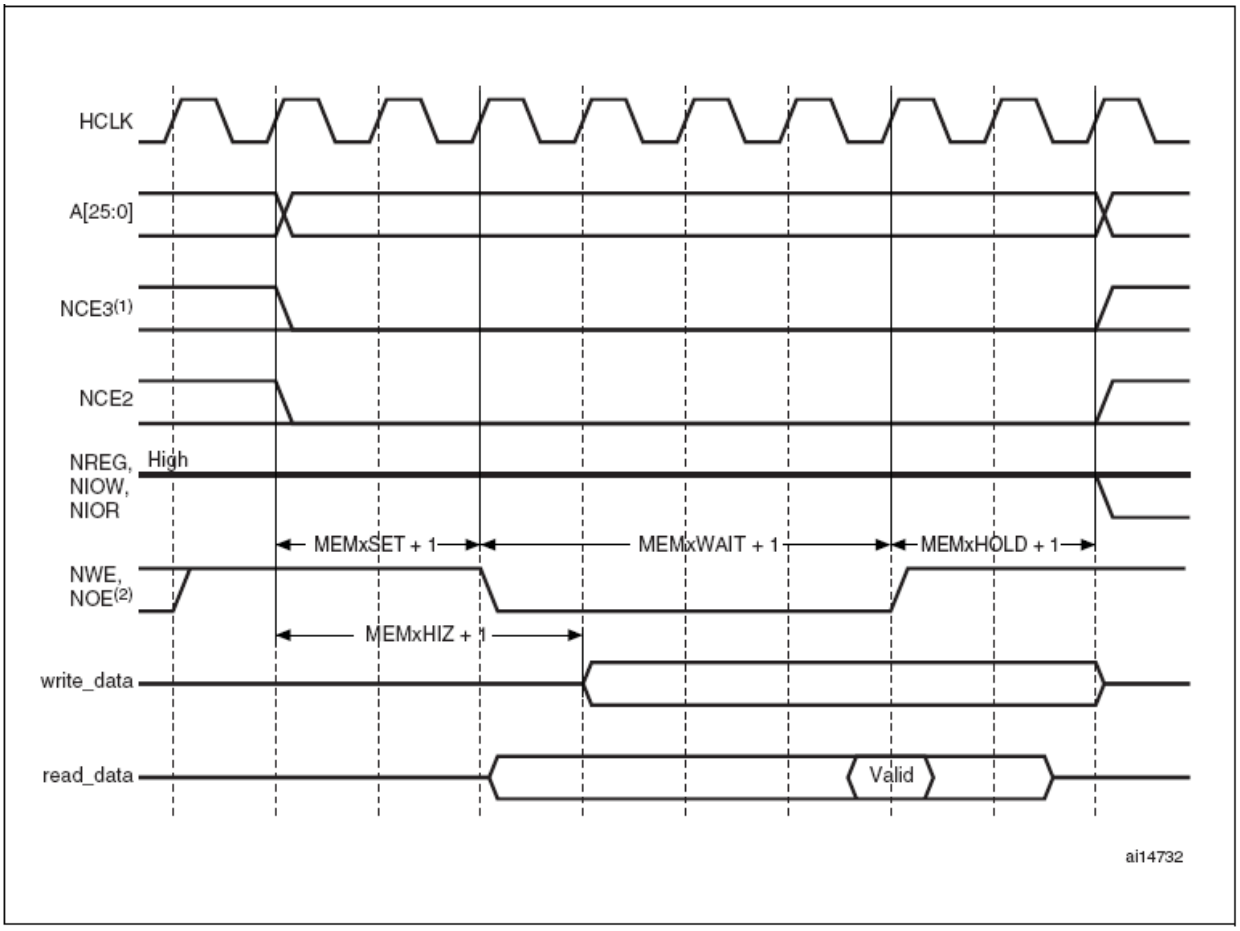
每个PC卡/CF卡和NAND闪存存储器块都是通过一组寄存器管理：

- 控制寄存器：FSMC_PCRx
- 中断状态寄存器：FSMC_SRx
- ECC寄存器：FSMC_ECCRx

- 通用存储器空间的时序寄存器：FSMC_PMEMx
- 属性存储器空间的时序寄存器：FSMC_PATTx
- I/O空间的时序寄存器：FSMC_PIOx

每一个时序控制寄存器都包含3个参数，用于定义PC卡/CF或NAND闪存操作中三个阶段的HCLK周期数目，还有一个定义了写操作中FSMC开始驱动数据总线时机的参数。下图给出了在通用存储空间中操作的时序参数定义，属性存储空间和I/O空间(只适用于PC卡)中操作与此相似。

图173 NAND/PC卡控制器通用存储空间的访问时序



1. *只要请求NAND访问，NCEx信号就变低并在访问其它存储器块之前保持为低。*
2. *在写操作时NOE始终保持高(无效状态)，在读操作时NWE始终保持高(无效状态)。*

18.6.4 NAND闪存操作

正如前面所述，NAND闪存的命令锁存使能(CLE)和地址锁存使能(ALE)信号由FSMC的地址信号线驱动。这意味着在向NAND闪存发送命令或地址时，CPU需要对存储空间中的特定地址执行写操作。

一个典型的对NAND闪存的读操作有如下步骤：

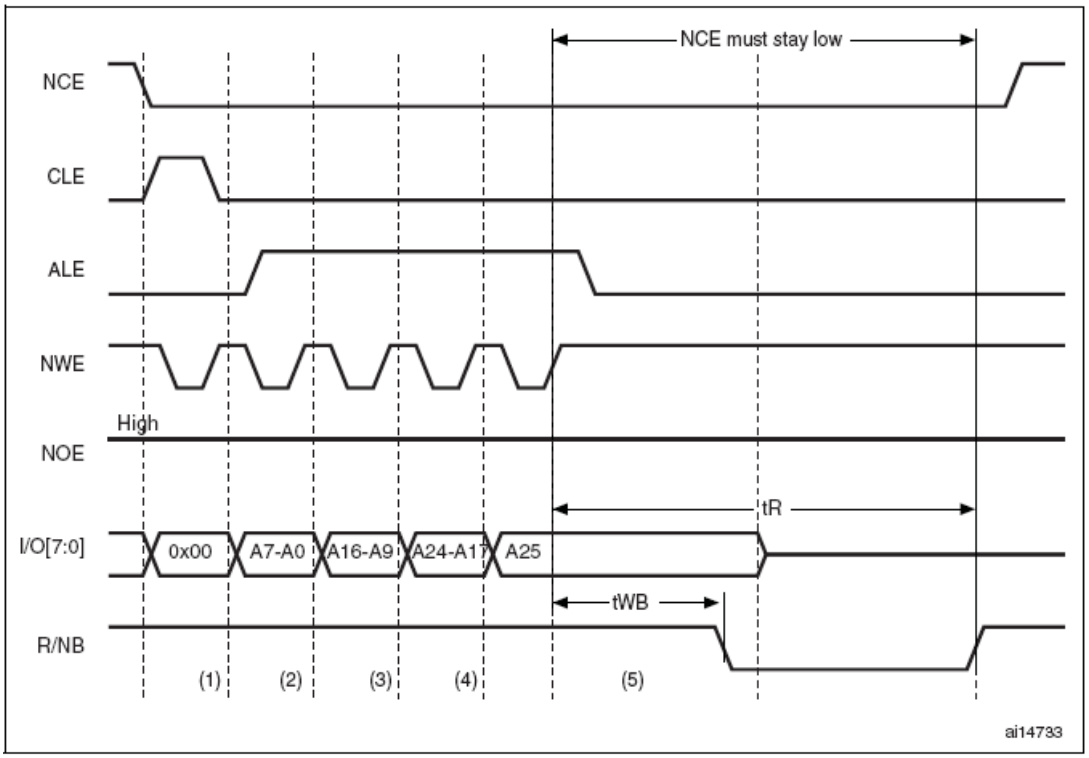
1. 根据NAND闪存的特性，通过FSMC_PCRx和FSMC_PMEMx寄存器配置和使能相应的存储器块，对于某些NAND闪存可能还要操作FSMC_PATTx寄存器(见18.6.5节——NAND闪存预等待功能)。需要配置的位包括：PWID指示NAND闪存的数据总线宽度，PTYP=1，PWAITEN=1，PBKEN=1，参见FSMC_PMEM2..4寄存器的时序配置。
2. CPU在通用存储空间写入闪存命令字节(例如对于Samsung的NAND闪存，该字节为0x00)，在写信号有效期间(NWE的低脉冲)NAND闪存的CLE输入变为有效(高电平)，这时CPU写的字节被NAND闪存识别为一个命令。一旦NAND闪存锁存了这个命令，随后的页读操作不必再发送相同的命令。

3. CPU在通用存储器空间或属性空间写入四个字节(较小容量的NAND闪存可能只需要三个字节)作为读操作的开始地址(STARTAD)，以64Mbx8的NAND闪存为例，按照STARTAD[7:0]、STARTAD[16:9]、STARTAD[24:17]和STARTAD[25] 的顺序写入。在写信号有效期间(NWE的低脉冲)NAND闪存的ALE输入变为有效(高电平)，这时CPU写的字节被NAND闪存识别为读操作的开始地址。使用属性存储空间，可以使FSMC产生不同的时序，实现某些NAND闪存所需的预等待功能(详见18.6.5节—— NAND闪存预等待功能)。
4. 控制器在开始(对相同的或另一个存储器块)新的操作之前等待NAND闪存准备就绪(R/NB信号变为高)，在等待期间控制器保持NCE信号有效(低电平)。
5. CPU可以在通用存储空间执行字节读操作，逐字节地读出NAND闪存的存储页(数据域和后备域)。
6. 在CPU不写入命令或地址的情况下，NAND闪存的下一个页可以以下述任一种方式读出：
   - 按照步骤 5 进行操作
   - 返回步骤 3 开始输入一个新的地址
   - 返回步骤 2 开始输入一个新的命令

18.6.5 NAND闪存预等待功能

某些NAND闪存要求在输入最后一个地址字节后，控制器等待R/NB信号变低，如下图：

图174 操作CE敏感型NAND闪存



1. CPU写字节命令0x00至0x7001 0000
2. CPU写NAND的地址A7~A0至0x7002 0000
3. CPU写NAND的地址A16~A9至0x7002 0000
4. CPU写NAND的地址A24~A17至0x7002 0000
5. CPU写NAND的地址A25至0x7802 0000：这时FSMC使用FSMC_PATT2的时序定义执行写操作，此时ATTHOLD≥7 ((7+1)xHCLK=112ns > t_{WB}的最大值)。这样可以保证R/NB变低再变高的过程中NCE保持为低，只有那些对CE敏感型的NAND闪存有此要求。

当需要这样的功能时，可以通过配置MEMHOLD的数值来保证t_{WB}的时序，但是任何随后的CPU对NAND闪存的读或写操作中，控制器都会在NWE信号的上升沿至下一次操作之间插入一个保持延迟，延迟长度为(MEMHOLD+1)个HCLK周期。

为了克服这个时序的限制，这里使用了属性存储空间配置ATTHOLD的数值使之符合t_{WB}的时序，同时保持MEMHOLD为其最小值。此时，CPU必须在所有NAND闪存的读写操作时使用通用存储空间，只有在写入NAND闪存地址的最后一个字节时，CPU需要写入属性存储空间。

18.6.6 NAND闪存的纠错码ECC计算(NAND闪存)

FSMC的PC卡控制器包含了2个纠错码计算的硬件模块，存储块2和3各有一个。该模块可以用于减小CPU在处理纠错码时的软件工作量。

有两个相同的寄存器分别对应存储块2和存储块3，存储块4没有包含硬件的ECC计算模块。

FSMC中实现的纠错码(ECC)算法可以在读或写NAND闪存时，在每256、512、1024、2048、4096或8192个字节中，矫正1个比特位的错误并且检测出2个比特位的错误。

每当NAND闪存存储器卡被激活时，ECC模块监测NAND闪存的数据总线和读/写信号(NCE和NWE)。

该功能的操作如下：

- 当在存储器块2或存储器块3访问NAND闪存时，出现在D[15:0]总线上的数据被锁存并用于ECC计算。
- 当对NAND闪存的操作发生在其它地址时，ECC电路不进行任何操作。因此输出NAND闪存命令和地址的写操作不会参与ECC计算。

当规定数目的字节已经写入NAND闪存或从NAND闪存读出，软件必须读出FSMC_ECCR2/3寄存器以获得计算的ECC数值。读出ECC数值后，再次计算ECC时需要通过先置ECCEN为'0'清除这个寄存器，再在FSMC_PCR2/3寄存器的ECCEN位写'1'重新使能ECC计算。

18.6.7 NAND闪存和PC卡控制器寄存器

PC卡/NAND闪存控制寄存器 2..4 (FSMC_PCR2..4)

地址偏移：0xA000 0000 + 0x40 + 0x20 * (x-1), x=2..4

复位值：0x0000 0018

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留												ECCPS			TAR				TCLR				保留		ECCEN	PWID			PTYP	PBKEN	PWAITEN	保留
res												rw			rw				rw				res		rw	rw			rw	rw	rw	res

位19:17	**ECCPS**：ECC页面大小 定义扩展的ECC页面大小： 000：256字节； 001：512字节； 010：1024字节； 011：2048字节； 100：4096字节； 101：8192字节。
位16:13	**TAR**：ALE至RE的延迟 以AHB时钟周期(HCLK)为单位设置从ALE变低至RE变低的时间。 时间计算：t_ar = (TAR + SET + 4) x THCLK，这里THCLK表示HCLK周期长度 0000：1个HCLK周期(默认值)； …… 1111: 16个HCLK周期。 注：根据不同的地址空间，SET是MEMSET或是ATTSET。

位12:9	**TCLR**：CLE至RE的延迟 以AHB时钟周期(HCLK)为单位设置从CLE变低至RE变低的时间。 时间计算：t_clr = (TCLR + SET + 4) x THCLK，这里THCLK表示HCLK周期长度 0000：1个HCLK周期(默认值)； …… 1111: 16个HCLK周期。 注：根据不同的地址空间，SET是MEMSET或是ATTSET。
位8:7	保留
位6	**ECCEN**：ECC计算电路使能位 0：关闭并复位ECC电路(复位后的默认值)； 1：使能ECC电路。
位5:4	**PWID**：数据总线宽度 定义外部NAND闪存数据总线的宽度。 00：8位(复位后的默认值)； 01：16位(PC卡必须使用此设置)； 10：保留，不要使用； 11：保留，不要使用。
位3	**PTYP**：存储器类型 定义对应的存储器块上连接的存储器类型。 0：PC卡、CF卡、CF+卡或PCMCIA； 1：NAND闪存(复位后的默认值)。
位2	**PBKEN**：PC卡/NAND存储器块使能位 使能存储器块。访问一个未使能的存储器块会产生一个AHB总线错误。 0：关闭对应的存储器块(复位后的默认值)； 1：使能对应的存储器块。
位1	**PWAITEN**：等待功能使能位 使能PC卡/NAND闪存存储器块的等待功能 0：关闭(复位后的默认值)； 1：使能。 注：对于PC卡，如果使能了等待功能，MEMWAITx/ATTWAITx/IOWAITx位的值必须高于$t_{v(IORDY-NOE)}/T_{HCLK}+4$，其中$t_{v(IORDY-NOE)}$是一旦NOE为低时NWAIT变低所需的最长时间。
位0	保留

FIFO状态和中断寄存器 2..4 (FSMC_SR2..4)

地址偏移：0xA000 0000 + 0x44 + 0x20 * (x-1), x=2..4

复位值：0x0000 0040

该寄存器包含FIFO状态和中断的信息。FSMC有一个FIFO，在写存储器时用于保存从AHB送来的多达16个字的数据。

这个功能可以在操作FSMC时不过多地占用AHB，在FSMC从FIFO传送数据到存储器时施放AHB的带宽并操作其他外设。

为了计算ECC的需要，该寄存器有一个指示位反映了FIFO的状态。数据写到存储器时同时进行ECC计算，因此软件必须等待FIFO变空后才能读到正确的ECC数值。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留																									FEMPT	IFEN	ILEN	IREN	IFS	ILS	IRS
res																									r	rw	rw	rw	rw	rw	rw

位6	**FEMPT**：FIFO空标志 只读位，指示FIFO状态 0：FIFO不空； 1：FIFO空。
位5	**IFEN**：中断下降沿检测使能 0：关闭中断下降沿检测请求； 1：使能中断下降沿检测请求。
位4	**ILEN**：中断高电平检测使能 0：关闭中断高电平检测请求； 1：使能中断高电平检测请求。
位3	**IREN**：上升沿中断检测使能 0：关闭中断上升沿检测请求； 1：使能中断上升沿检测请求。
位2	**IFS**：中断下降沿状态 该位由硬件设置，软件清除。 0：没有产生中断下降沿； 1：产生了中断下降沿。
位1	**IRS**：中断高电平状态 该位由硬件设置，软件清除。 0：没有产生中断高电平； 1：产生了中断高电平。
位0	**IRS**：中断上升沿状态 该位由硬件设置，软件清除。 0：没有产生中断上升沿； 1：产生了中断上升沿。

通用存储空间时序寄存器 2..4 (FSMC_PMEM2..4)

地址偏移：0xA000 0000 + 0x48 + 0x20 * (x-1), x=2..4

复位值：0xFCFC FCFC

每个FSMC_PMEMx(x=2..4)寄存器都包含操作PC卡或NAND闪存存储块x的时序参数，这些参数适用于在通用存储空间操作16位PC卡/CF卡，或发送NAND闪存的命令、地址和进行数据的读写操作。

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

位31:24	**MEMHIZx**：在通用空间x数据总线的高阻时间 当在通用存储空间x开始执行对PC卡/NAND闪存的写操作后，数据总线需要保持一段时间的高阻状态，该参数以HCLK时钟周期数目(NAND类型时+1)定义数据总线高阻态的时间。这个参数仅对写操作有效。 0000 0000：(0x00)0个HCLK周期； …… 1111 1111：(0xFF)255个HCLK周期(复位后的默认值)。

位23:16	**MEMHOLDx：**在通用空间x的保持时间 当在通用存储空间x对PC卡/NAND闪存进行读或写操作时，该参数以HCLK(+1)时钟周期数目定义了发送命令(NWE、NOE变高)后，地址信号(对于写操作则是数据信号)保持的时间。 0000 0000：保留，不要使用这个数值； 0000 0001：1个HCLK周期； …… 1111 1111：(0xFF)255个HCLK周期(复位后的默认值)。
位15:8	**MEMWAITx：**在通用空间x的等待时间 当在通用存储空间x对PC卡/NAND闪存进行读或写操作时，该参数以HCLK(+1)时钟周期数目定义了保持命令(NWE、NOE为低)的最小时间。当该参数定义的时间结束时如果等待信号(NWAIT)有效(低)，则命令的保持时间会被拉长。 0000 0000：保留，不要使用这个数值； 0000 0001：2个HCLK周期(加上由NWAIT信号变低引入的等待周期)； …… 1111 1111：256个HCLK周期(加上由卡的NWAIT信号变低引入的等待周期)(复位后的默认值)。
位7:0	**MEMSETx：**在通用空间x的建立时间 当在通用存储空间x对PC卡/NAND闪存进行读或写操作时，该参数以HCLK(操作PC卡时+1，操作NAND闪存时+2)时钟周期数目定义了发送命令(NWE、NOE变低)之前建立地址信号的时间。 0000 0000：1个HCLK周期； …… 1111 1111：256个HCLK周期(复位后的默认值)。

属性存储空间时序寄存器 2..4 (FSMC_PATT2..4)

地址偏移：0xA000 0000 + 0x4C + 0x20 * (x-1), x=2..4

复位值：0xFCFC FCFC

每个FSMC_PATTx(x=2..4)读/写寄存器都包含操作PC卡/CF卡或NAND闪存存储块x的时序参数，这些参数适用于在属性存储空间操作8位PC卡/CF卡(每个AHB操作被分解为一系列的8位操作)，或在NAND闪存的最后一个地址写操作的时序与其它操作不同的时候(关于就绪/繁忙的管理，参见18.6.5节)。

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

位31:24	**ATTHIZx：**在属性空间x数据总线的高阻时间 当在属性存储空间x开始执行对PC卡/NAND闪存的写操作后，数据总线需要保持一段时间的高阻状态，该参数以HCLK时钟周期数目定义数据总线高阻态的时间。这个参数仅对写操作有效。 0000 0000：0个HCLK周期； …… 1111 1111：255个HCLK周期(复位后的默认值)。
位23:16	**ATTHOLDx：**在属性空间x的保持时间 当在属性存储空间x对PC卡/NAND闪存进行读或写操作时，该参数以HCLK时钟周期数目定义了发送命令(NWE、NOE变高)后，地址信号(对于写操作则是数据信号)保持的时间。 0000 0000：1个HCLK周期； …… 1111 1111：256个HCLK周期(复位后的默认值)。

位15:8	**ATTWAITx：**在属性空间x的等待时间 当在属性存储空间x对PC卡/NAND闪存进行读或写操作时，该参数以HCLK(+1)时钟周期数目定义了保持命令(NWE、NOE为低)的最小时间。当该参数定义的时间结束时如果等待信号(NWAIT)有效(低)，则命令的保持时间会被拉长。 0000 0000：1个HCLK周期(加上由NWAIT信号变低引入的等待周期)； …… 1111 1111：256个HCLK周期(加上由卡的NWAIT信号变低引入的等待周期)(复位后的默认值)。
位7:0	**ATTSETx：**在属性空间x的建立时间 当在属性存储空间x对PC卡/NAND闪存进行读或写操作时，该参数以HCLK(+1)时钟周期数目定义了发送命令(NWE、NOE变低)之前建立地址信号的时间。 0000 0000：1个HCLK周期； …… 1111 1111：256个HCLK周期(复位后的默认值)。

I/O空间时序寄存器4 (FSMC_PIO4)

地址偏移：0xA000 0000 + 0xB0

复位值：0xFCFC FCFC

FSMC_PIO4寄存器包含了在I/O空间操作16位PC卡/CF卡的时序参数。

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

位31:24	**IOHIZx**：在I/O空间x数据总线的高阻时间 当在I/O空间x开始执行对PC卡的写操作后，数据总线需要保持一段时间的高阻状态，该参数以HCLK时钟周期数目定义数据总线高阻态的时间。这个参数仅对写操作有效。 0000 0000：0个HCLK周期； …… 1111 1111：255个HCLK周期(复位后的默认值)。
位23:16	**IOHOLDx**：在I/O空间x的保持时间 当在I/O空间x对PC卡进行读或写操作时，该参数以HCLK时钟周期数目定义了发送命令(NWE、NOE变高)后，地址信号(对于写操作则是数据信号)保持的时间。 0000 0000：1个HCLK周期； …… 1111 1111：256个HCLK周期(复位后的默认值)。
位15:8	**IOWAITx**：在I/O空间x的等待时间 当在I/O空间x对PC卡进行读或写操作时，该参数以HCLK(+1)时钟周期数目定义了保持命令(SMNWE、SMNOE为低)的最小时间。当该参数定义的时间结束时如果等待信号(NWAIT)有效(低)，则命令的保持时间会被拉长。 0000 0000：保留，不要使用这个数值； 0000 0001：2个HCLK周期(加上由NWAIT信号变低引入的等待周期)； …… 1111 1111：256个HCLK周期(加上由卡的NWAIT信号变低引入的等待周期)(复位后的默认值)。
位7:0	**IOSETx**：在I/O空间x的建立时间 当在I/O空间x对PC卡进行读或写操作时，该参数以HCLK(+1)时钟周期数目定义了发送命令(NWE、NOE变低)之前建立地址信号的时间。 0000 0000：1个HCLK周期； …… 1111 1111：256个HCLK周期(复位后的默认值)。

ECC结果寄存器2/3 (FSMC_ECCR2/3)

地址偏移：0xA000 0000 + 0x54 + 0x20 * (x-1), x=2或3

复位值：0x0000 0000

这2个寄存器包含了由FSMC控制器的ECC计算模块得到的纠错码的当前数值，每个NAND闪存存储器块有一个ECC计算模块。当CPU在正确的地址(见18.6.6节)读/写NAND闪存的数据时，ECC模块会自动地处理写入或读出的数据。根据FSMC_PCRx中ECCPS域的设置，在读出了每页的最后一个字节后，CPU必须读出FSMC_ECCx寄存器中的ECC数值，并与记录在NAND闪存后备区域的数据进行比较，据此判断该页的数据是否正确并在可能的情况下，实行矫正。在读出FSMC_ECCRx寄存器的数值后应设置ECCEN位为'0'清除它的内容。需要计算一个新的数据页时，再次设置ECCEN为'1'。

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

位31:0	**ECCx：**ECC结果 ECC计算电路产生的计算结果。下表显示了这些位的内容。

表101 ECC结果有效位

ECCPS[2:0]	页大小(字节)	ECC有效位
000	256	ECC[21:0]
001	512	ECC[23:0]
010	1024	ECC[25:0]
011	2048	ECC[27:0]
100	4096	ECC[29:0]
101	8192	ECC[31:0]

18.7 FSMC寄存器地址映象

注： *本节为译者整理所得，原英文版本没有这一节。*

图2-1 FSMC寄存器地址映象

偏移	寄存器名称	复位值	说明
000h	FSMC_BCR1	0x0000 30DB	SRAM/NOR闪存片选控制寄存器 1
004h	FSMC_BTR1	0x0FFF FFFF	SRAM/NOR闪存片选时序寄存器 1
008h	FSMC_BCR2	0x0000 30D2	SRAM/NOR闪存片选控制寄存器 2
00Ch	FSMC_BTR2	0x0FFF FFFF	SRAM/NOR闪存片选时序寄存器 2
010h	FSMC_BCR3	0x0000 30D2	SRAM/NOR闪存片选控制寄存器 3
014h	FSMC_BTR3	0x0FFF FFFF	SRAM/NOR闪存片选时序寄存器 3
018h	FSMC_BCR4	0x0000 30D2	SRAM/NOR闪存片选控制寄存器 4
01Ch	FSMC_BTR4	0x0FFF FFFF	SRAM/NOR闪存片选时序寄存器 4
060h	FSMC_PCR2	0x0000 0018	PC卡/NAND闪存控制寄存器 2
064h	FSMC_SR2	0x0000 0040	FIFO状态和中断寄存器 2
068h	FSMC_PMEM2	0xFCFC FCFC	通用存储空间时序寄存器 2
06Ch	FSMC_PATT2	0xFCFC FCFC	属性存储空间时序寄存器 2
080h	FSMC_PCR3	0x0000 0018	PC卡/NAND闪存控制寄存器 3
084h	FSMC_SR3	0x0000 0040	FIFO状态和中断寄存器 3
088h	FSMC_PMEM3	0xFCFC FCFC	通用存储空间时序寄存器 3
08Ch	FSMC_PATT3	0xFCFC FCFC	属性存储空间时序寄存器 3
0A0h	FSMC_PCR4	0x0000 0018	PC卡/NAND闪存控制寄存器 4
0A4h	FSMC_SR4	0x0000 0040	FIFO状态和中断寄存器 4
0A8h	FSMC_PMEM4	0xFCFC FCFC	通用存储空间时序寄存器 4
0ACh	FSMC_PATT4	0xFCFC FCFC	属性存储空间时序寄存器 4
0B0h	FSMC_PIO4	0xFCFC FCFC	I/O存储空间时序寄存器 4
104h	FSMC_BWTR1	0x0FFF FFFF	SRAM/NOR闪存写时序寄存器 1
10Ch	FSMC_BWTR2	0x0FFF FFFF	SRAM/NOR闪存写时序寄存器 2
114h	FSMC_BWTR3	0x0FFF FFFF	SRAM/NOR闪存写时序寄存器 3
11Ch	FSMC_BWTR4	0x0FFF FFFF	SRAM/NOR闪存写时序寄存器 4

19 SDIO接口(SDIO)

小容量产品是指闪存容量介于16K字节至32K字节的STM32F101xx、STM32F102xx和STM32F103xx微控制器。

中容量产品是指闪存容量介于64K字节至128K字节的STM32F101xx、STM32F102xx和STM32F103xx微控制器。

大容量产品是指闪存容量介于256K至512K字节的STM32F101xx和STM32F103xx微控制器。

本章内容只适用于大容量产品。

19.1 SDIO主要功能

SD/SDIO MMC卡主机模块(SDIO)在AHB外设总线和多媒体卡(MMC)、SD存储卡、SDIO卡和CE-ATA设备间提供了操作接口。

多媒体卡系统规格书由MMCA技术委员会发布，可以在多媒体卡协会的网站上(www.mmca.org)获得。

CE-ATA系统规格书可以在CE-ATA工作组的网站上(www.ce-ata.org)获得。

SDIO的主要功能如下：

- 与多媒体卡系统规格书版本4.2全兼容。支持三种不同的数据总线模式：1位(默认)、4位和8位。
- 与较早的多媒体卡系统规格版本全兼容(向前兼容)。
- 与SD存储卡规格版本2.0全兼容。
- 与SD I/O卡规格版本2.0全兼容：支持良种不同的数据总线模式：1位(默认)和4位。
- 完全支持CE-ATA功能(与CE-ATA数字协议版本1.1全兼容)。
- 8位总线模式下数据传输速率可达48MHz。
- 数据和命令输出使能信号，用于控制外部双向驱动器。

注： *1. SDIO没有SPI兼容的通信模式*

2．在多媒体卡系统规格书版本2.11中，定义SD存储卡协议是多媒体卡协议的超集。只支持I/O模式的SD卡或复合卡中的I/O部分不能支持SD存储设备中很多需要的命令，这里有些命令在SD I/O设备中不起作用，如擦除命令，因此SDIO不支持这些命令。另外，SD存储卡和SD I/O卡中有些命令是不同的，SDIO也不支持这些命令。细节可以参考SD I/O卡规格书版本1.0。使用现有的MMC命令机制，在MMC接口上可以实现CE-ATA的支持 。SDIO接口的电气和信号定义详见MMC参考资料。

多媒体卡/SD总线将所有卡与控制器相连。

当前版本的SDIO在同一时间里只能支持一个SD/SDIO/MMC 4.2卡，但可以支持多个MMC版本4.1或以前版本的卡。

19.2 SDIO总线拓扑

总线上的通信是通过传送命令和数据实现。

在多媒体卡/SD/SD I/O总线上的基本操作是命令/相应结构，这样的总线操作在命令或总线机制下实现信息交换；另外，某些操作还具有数据令牌。

在SD/SDIO存储器卡上传送的数据是以数据块的形式进行；在MMC上传送的数据是以数据块或数据流的形式进行；在CE-ATA设备上传送的数据也是以数据块的形式进行。

图175 SDIO“无响应”和“无数据”操作

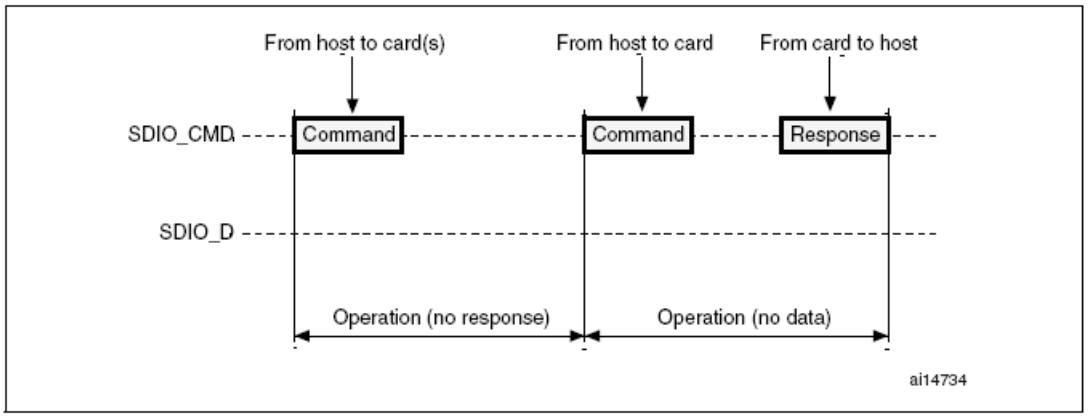


图176 SDIO(多)数据块读操作

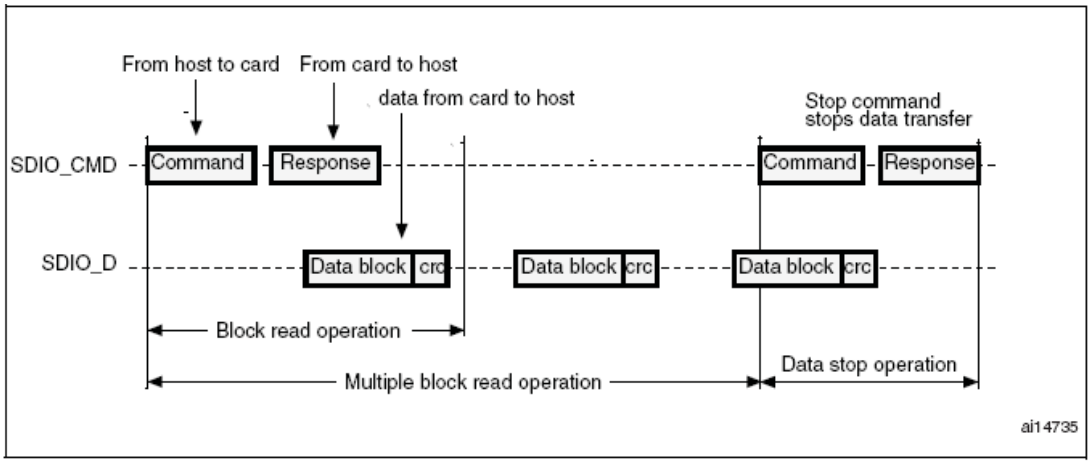
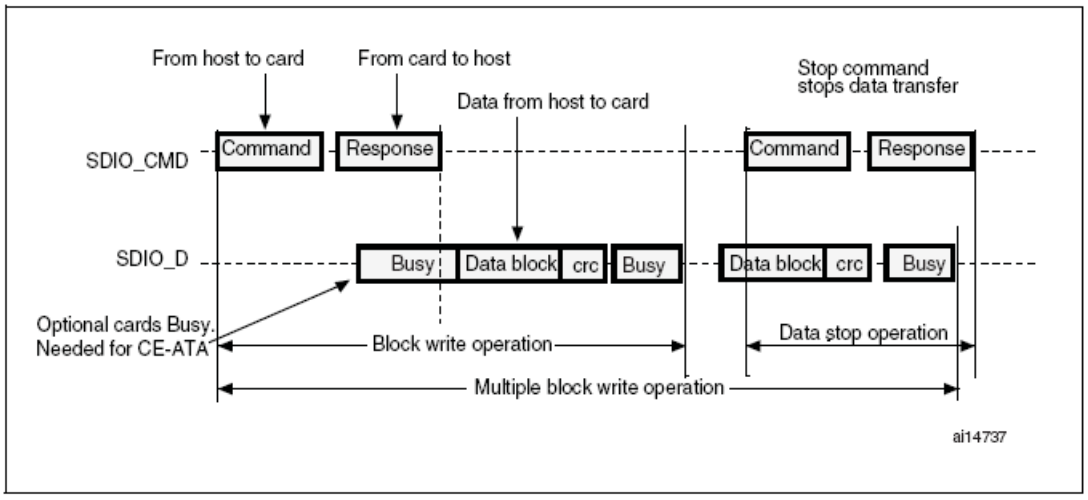


图177 SDIO(多)数据块写操作



注： *当有Busy信号时，SDIO(SDIO_D0被拉低)将不会发送任何数据。*

图178 SDIO连续读操作

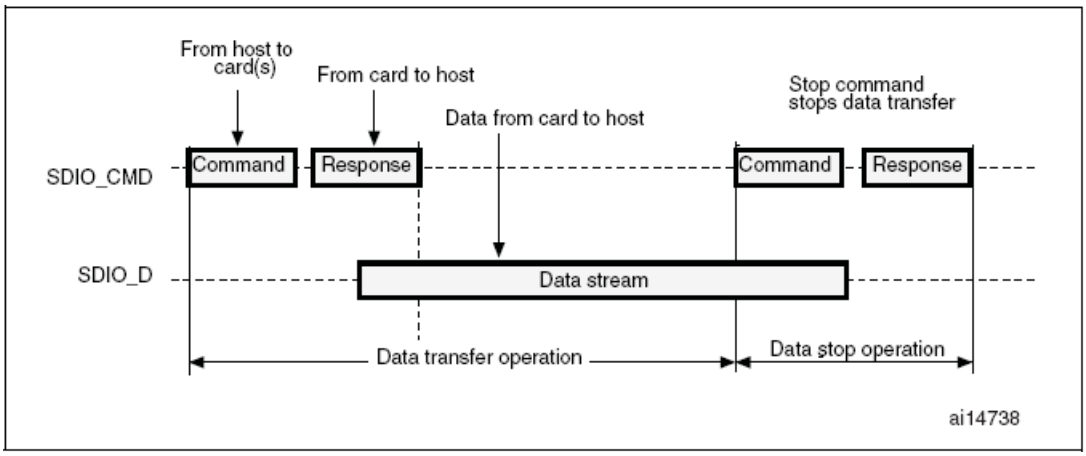
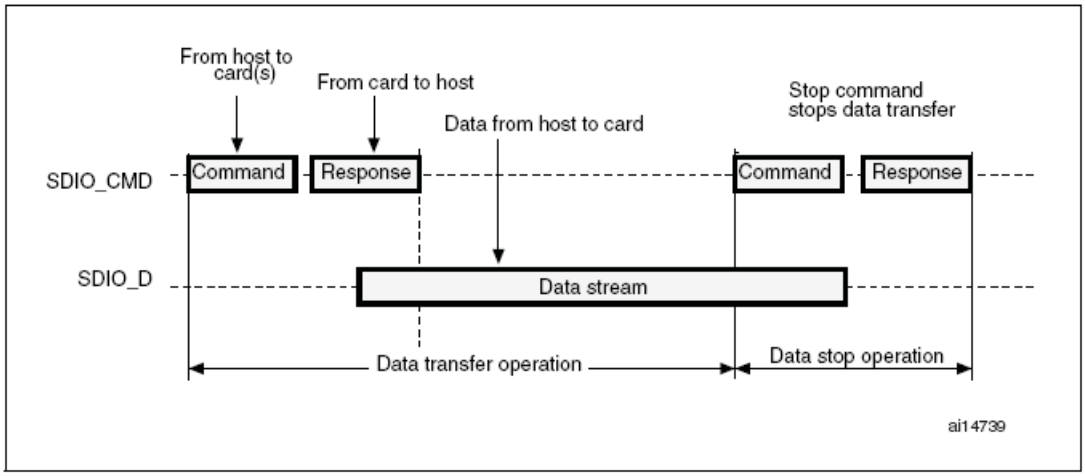


图179 SDIO连续写操作

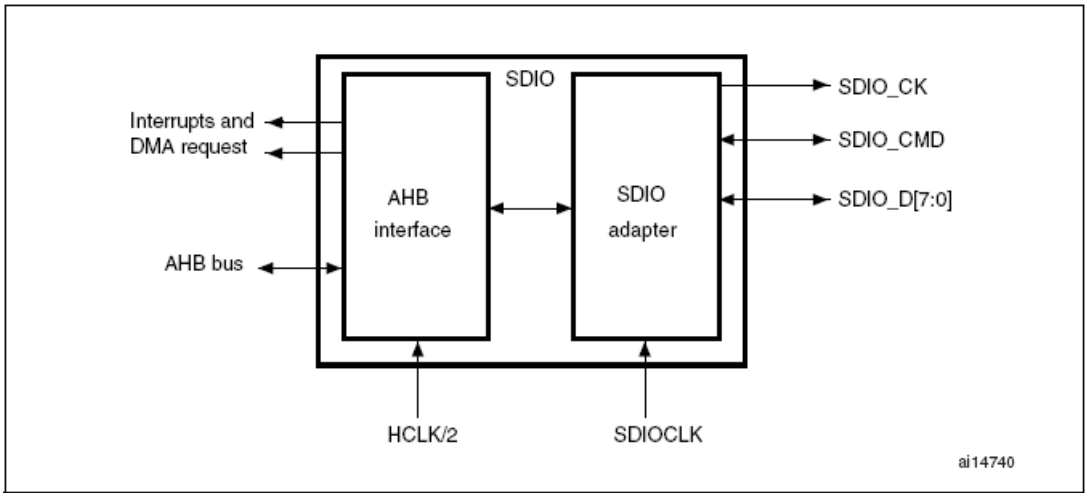


19.3 SDIO功能描述

SDIO包含2个部分：

- SDIO适配器模块实现所有MMC/SD/SD I/O卡的相关功能，如时钟的产生、命令和数据的传送。
- AHB总线接口操作SDIO适配器模块中的寄存器，并产生中断和DMA请求信号。

图180 SDIO框图



复位后默认情况下SDIO_D0用于数据传输。初始化后主机可以改变数据总线的宽度。

如果一个多媒体卡接到了总线上，则SDIO_D0、SDIO_D[3:0]或SDIO_D[7:0]可以用于数据传输。MMC版本V3.31和之前版本的协议只支持1位数据线，所以只能用SDIO_D0。

如果一个SD或SD I/O卡接到了总线上，可以通过主机配置数据传输使用SDIO_D0或SDIO_D[3:0]。所有的数据线都工作在推挽模式。

SDIO_CMD有两种操作模式：

- 用于初始化时的开路模式(仅用于MMC版本V3.31或之前版本)
- 用于命令传输的推挽模式(SD/SD I/O卡和MMC V4.2在初始化时也使用推挽驱动)

SDIO_CK是卡的时钟：每个时钟周期在命令和数据线上传输1位命令或数据。对于多媒体卡V3.31协议，时钟频率可以在0MHz至20MHz间变化；对于多媒体卡V4.0/4.2协议，时钟频率可以在0MHz至48MHz间变化；对于SD或SD I/O卡，时钟频率可以在0MHz至25MHz间变化。

SDIO使用两个时钟信号：

- SDIO适配器时钟(SDIOCLK=HCLK)
- AHB总线时钟(HCLK/2)

下表适用于多媒体卡/SD/SD I/O卡总线：

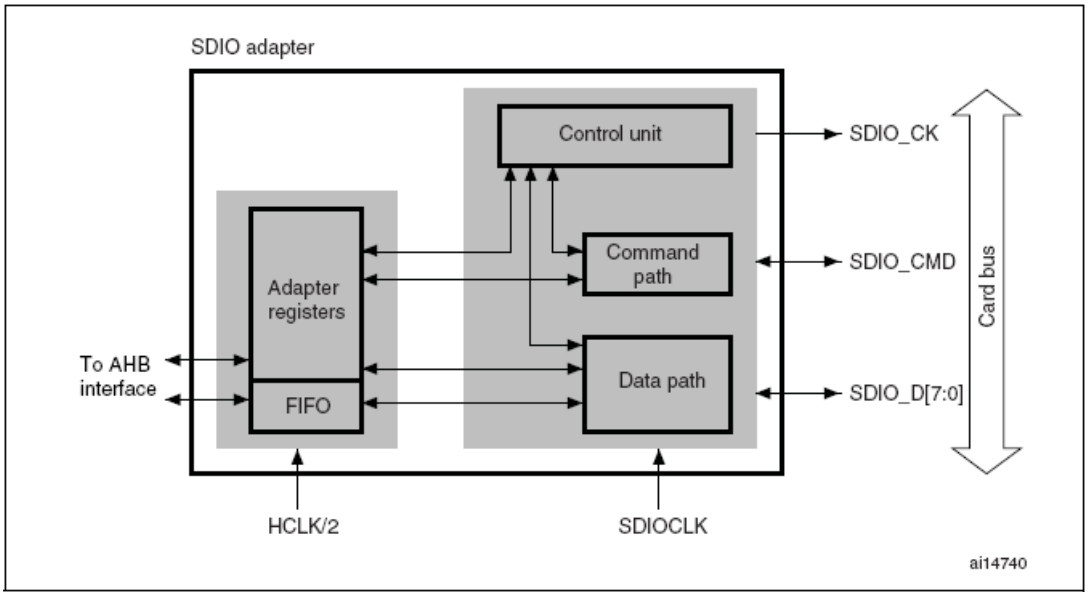
表102 SDIO I/O定义

管脚	方向	说明
SDIO_CK	输出	多媒体卡/SD/SDIO卡时钟。这是从主机至卡的时钟线。
SDIO_CMD	双向	多媒体卡/SD/SDIO卡命令。这是双向的命令/响应信号线。
SDIO_D[7:0]	双向	多媒体卡/SD/SDIO卡数据。这些是双向的数据总线。

19.3.1 SDIO适配器

下图是简化的SDIO适配器框图：

图181 SDIO适配器



SDIO适配器是多媒体/加密数字存储卡总线的主设备(主机)，用于连接一组多媒体卡或加密数字存储卡，它包含以下5个部分：

- 适配器寄存器模块
- 控制单元
- 命令通道
- 数据通道
- 数据FIFO

注：适配器寄存器和FIFO使用AHB总线一侧的时钟(HCLK/2)，控制单元、命令通道和数据通道使用SDIO适配器一侧的时钟(SDIOCLK)。

适配器寄存器模块

适配器寄存器模块包含所有系统寄存器。该模块还产生清除多媒体卡中静态标记的信号，当在SDIO清除寄存器中的相应位写'1'时会产生清除信号。

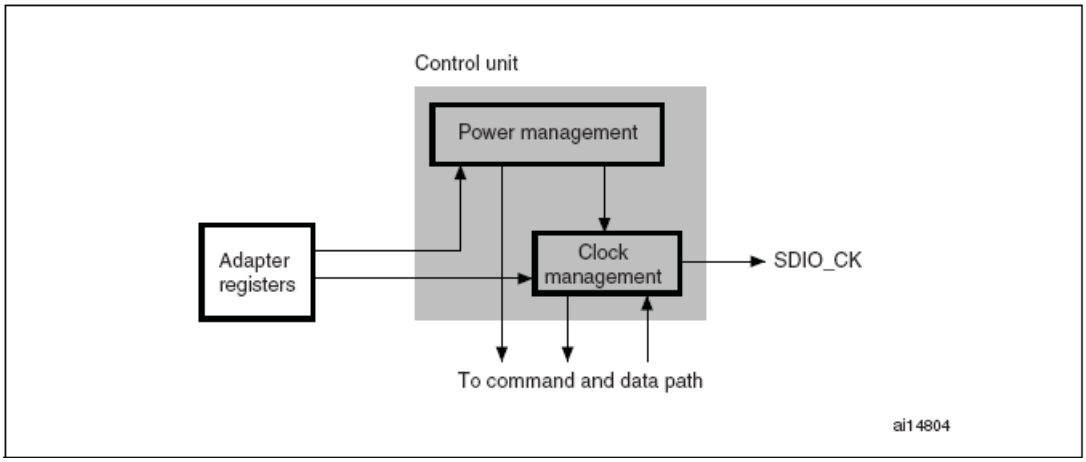
控制单元

控制单元包含电源管理功能和为存储器卡提供的时钟分频。

共有三种电源阶段：

- 电源关闭
- 电源启动
- 电源开

图182 控制单元



上图为控制单元的框图，有电源管理和时钟管理子单元。

在电源关闭和电源启动阶段，电源管理子单元会关闭卡总线上的输出信号。

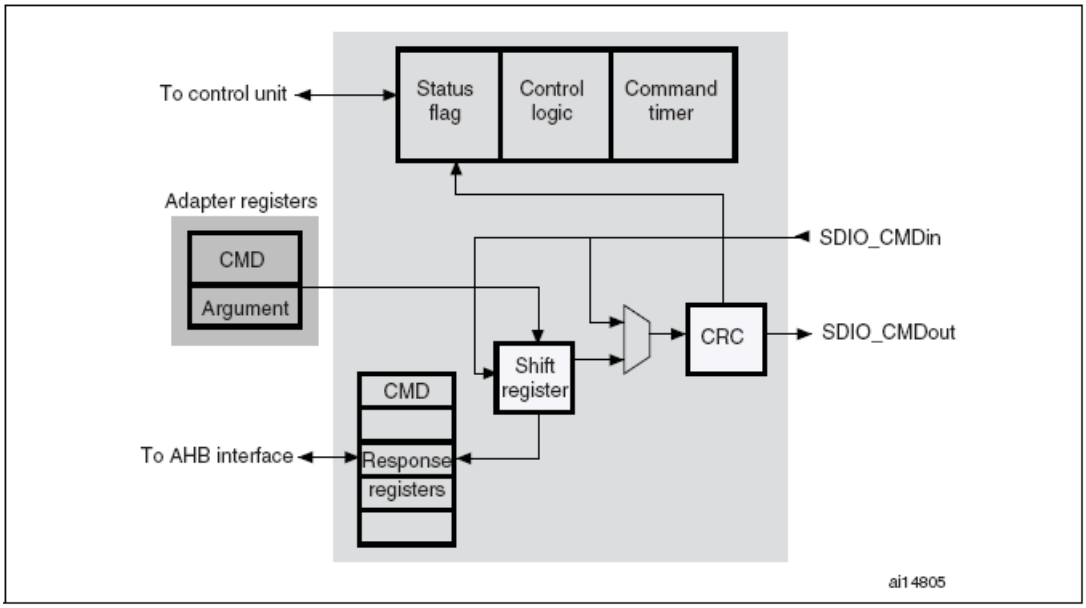
时钟管理子单元产生和控制SDIO_CK信号。SDIO_CK输出可以使用时钟分频或时钟旁路模式。下述情况下没有时钟输出：

- 复位后
- 在电源关闭和电源启动阶段
- 当启动了省电模式并且卡总线处于空闲状态(命令通道和数据通道子单元进入空闲阶段后的8个时钟周期)

命令通道

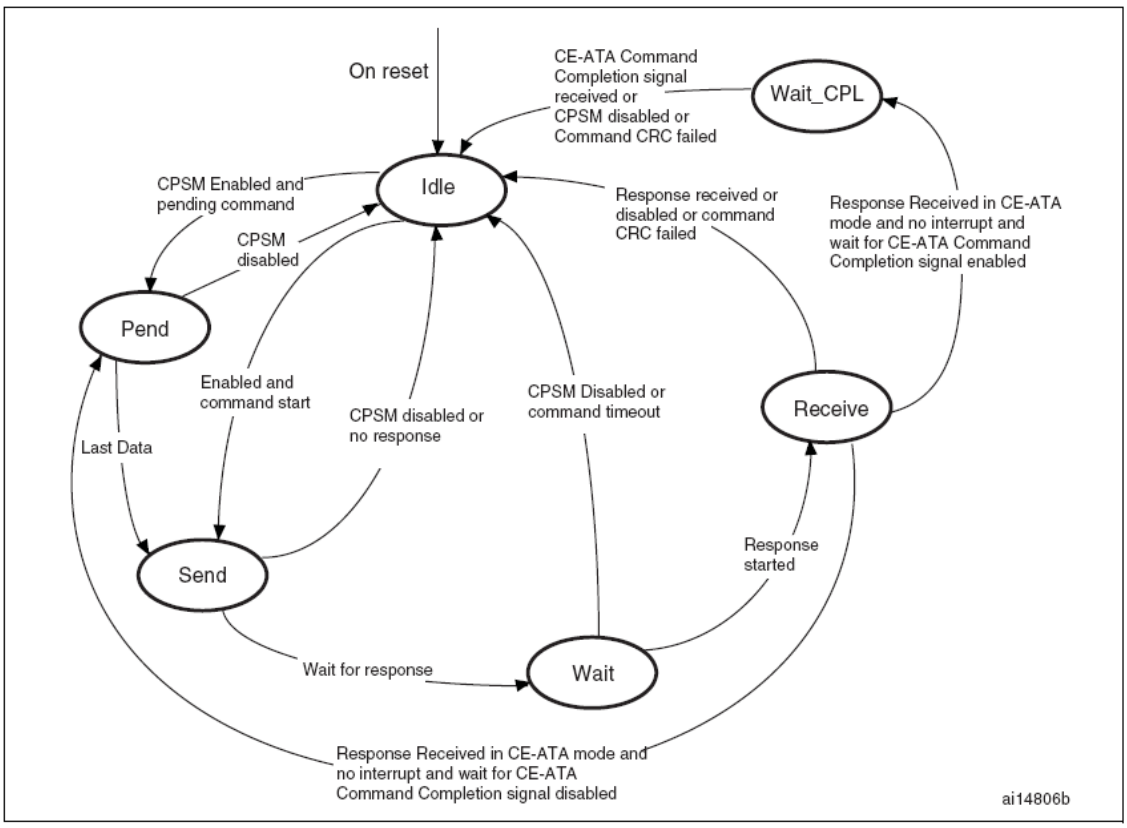
命令通道单元向卡发送命令并从卡接收响应。

图183 SDIO适配器命令通道



- 命令通道状态机(CPSM)
  - 当写入命令寄存器并设置了使能位，开始发送命令。命令发送完成时，命令通道状态机(CPSM)设置状态标志并在不需要响应时进入空闲状态(见下图)。当收到响应后，接收到的 CRC 码将会与内部产生的 CRC 码比较，然后设置相应的状态标志。

图184 命令通道状态机(CPSM)



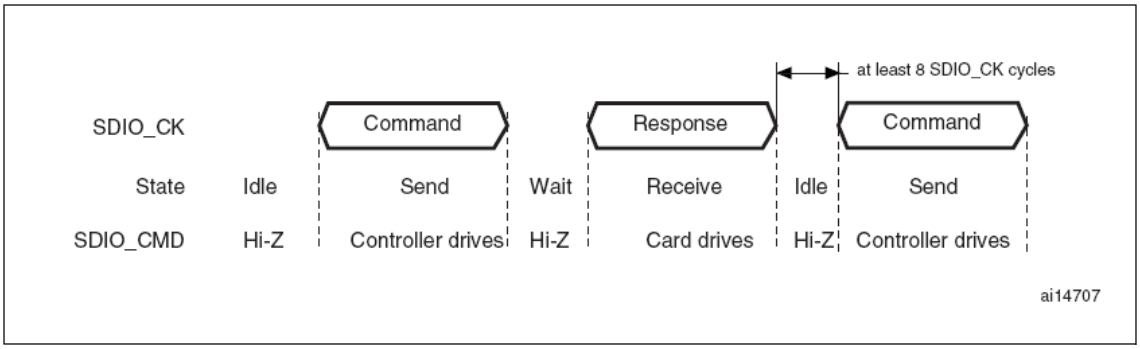
当进入等待状态时，命令定时器开始运行；当CPSM进入接收状态之前，产生了超时，则设置超时标志并进入空闲状态。

注： *命令超时固定为64个SDIO_CK时钟周期。*

如果在命令寄存器设置了中断位，则关闭定时器，CPSM等待某一个卡发出的中断请求。如果命令寄存器中设置挂起位，CPSM进入挂起状态并等待数据通道子单元发出的CmdPend信号，在检测到CmdPend信号时，CPSM进入发送状态，这将触发数据计数器发送停止命令的功能。

注： *CPSM保持在空闲状态至少8个SDIO_CK周期，以满足N_{CC}和N_{RC}时序限制。N_{CC}是两个主机命令间的最小间隔；N_{RC}是主机命令与卡响应之间的最小间隔。*

图185 SDIO命令传输



- 命令格式

  – 命令：命令是用于开始一项操作。主机向一个指定的卡或所有的卡发出带地址的命令或广播命令(广播命令只适合于MMC V3.31 或之前的版本)。命令在CMD线上串行传送。所有命令的长度固定为 48 位，下表给出了多媒体卡、SD存储卡和SDIO卡上一般的命令格式。CE-ATA命令是MMC V4.2 命令的扩充，所以具有相同的格式。
  命令通道操作于半双工模式，这样命令和响应可以分别发送和接收。如果CPSM不处在发

送状态，SDIO_CMD输出处于高阻状态，如图 185所示。SDIO_CMD上的数据与SDIO_CK的上升沿同步。

表103 命令格式

位	宽度	数值	说明
47	1	0	开始位
46	1	1	传输位
[45:40]	6	-	命令索引
[39:8]	32	-	参数
[7:1]	7	-	CRC7
0	1	1	结束位

— 响应：响应是由一个被指定地址的卡发送到主机，对于 MMC V3.31 或以前版本所有的卡同时发送响应；响应是对先前接收到命令的一个应答。响应在 CMD 线上串行传送。

SDIO支持2种响应类型，2种类型都有CRC错误检测：

- 48位短响应
- 136位长响应

注： *如果响应不包含CRC(如CMD1的响应)，设备驱动应该忽略CRC失败状态。*

表104 短响应格式

位	宽度	数值	说明
47	1	0	开始位
46	1	0	传输位
[45:40]	6	-	命令索引
[39:8]	32	-	参数
[7:1]	7	-	CRC7(或1111111)
0	1	1	结束位

表105 长响应格式

位	宽度	数值	说明
135	1	0	开始位
134	1	0	传输位
[133:128]	6	111111	保留
[127:1]	127	-	CID或CSD(包含内部CRC7)
0	1	1	结束位

命令寄存器包含命令索引(发至卡的6位)和命令类型；命令本身决定了是否需要响应和响应的类型，48位还是136位(见19.9.4节)。命令通道中的状态标志示于下表：

表106 命令通道状态标志

标志	说明
CMDREND	响应的CRC正确
CCRCFAIL	响应的CRC错误
CMDSENT	命令(不需要响应的命令)已经送出
CTIMEOUT	响应超时
CMDACT	正在发送命令

CRC发生器计算CRC码之前所有位的CRC校验和，包括开始位、发送位、命令索引和命令参数(或卡状态)。对于长响应格式，CRC校验和计算的是CID或CSD的前120位；注意，长响应格式中的开始位、传输位和6个保留位不参与CRC计算。

CRC校验和是一个7位的数值：

CRC[6:0] = 余数[(M(x) * x^7) / G(x)]

G(x) = x^7 + x^3 + 1

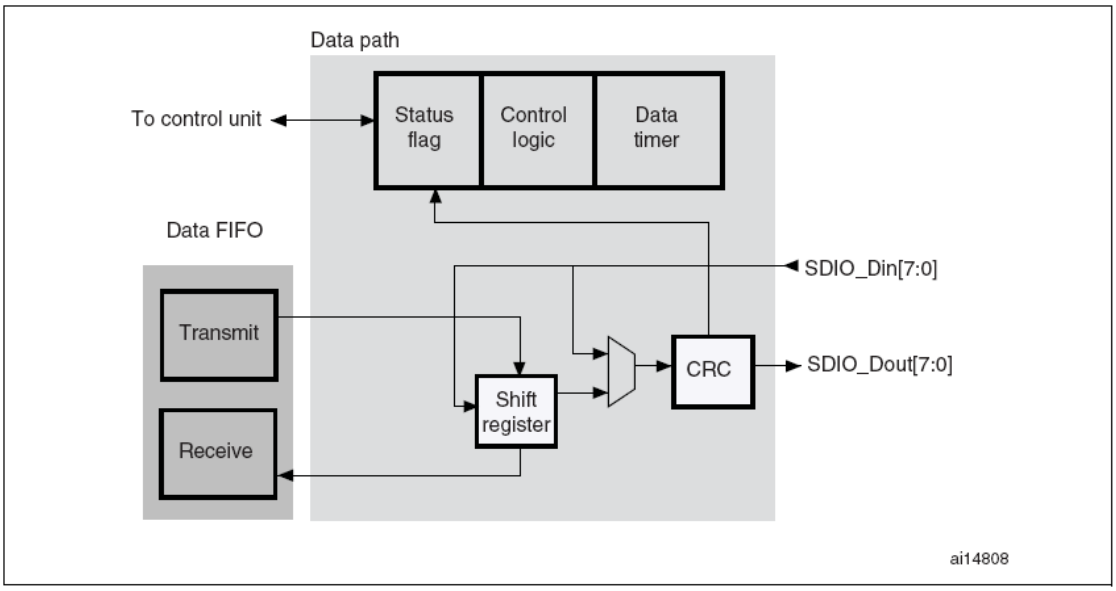
M(x) = (开始位) * x^{39} + … + (CRC前的最后一位) * x^0, 或

M(x) = (开始位) * x^{119} + … + (CRC前的最后一位) * x^0, 或

数据通道

数据通道子单元在主机与卡之间传输数据。下图是数据通道的框图。

图186 数据通道



在时钟控制寄存器中可以配置卡的数据总线宽度。如果选择了4位总线模式，则每个时钟周期四条数据信号线(SDIO_D[3:0])上将传输4位数据；如果选择了8位总线模式，则每个时钟周期八条数据信号线(SDIO_D[7:0])上将传输8位数据；如果没有选择宽总线模式，则每个时钟周期只在SDIO_D0上传输1位数据。

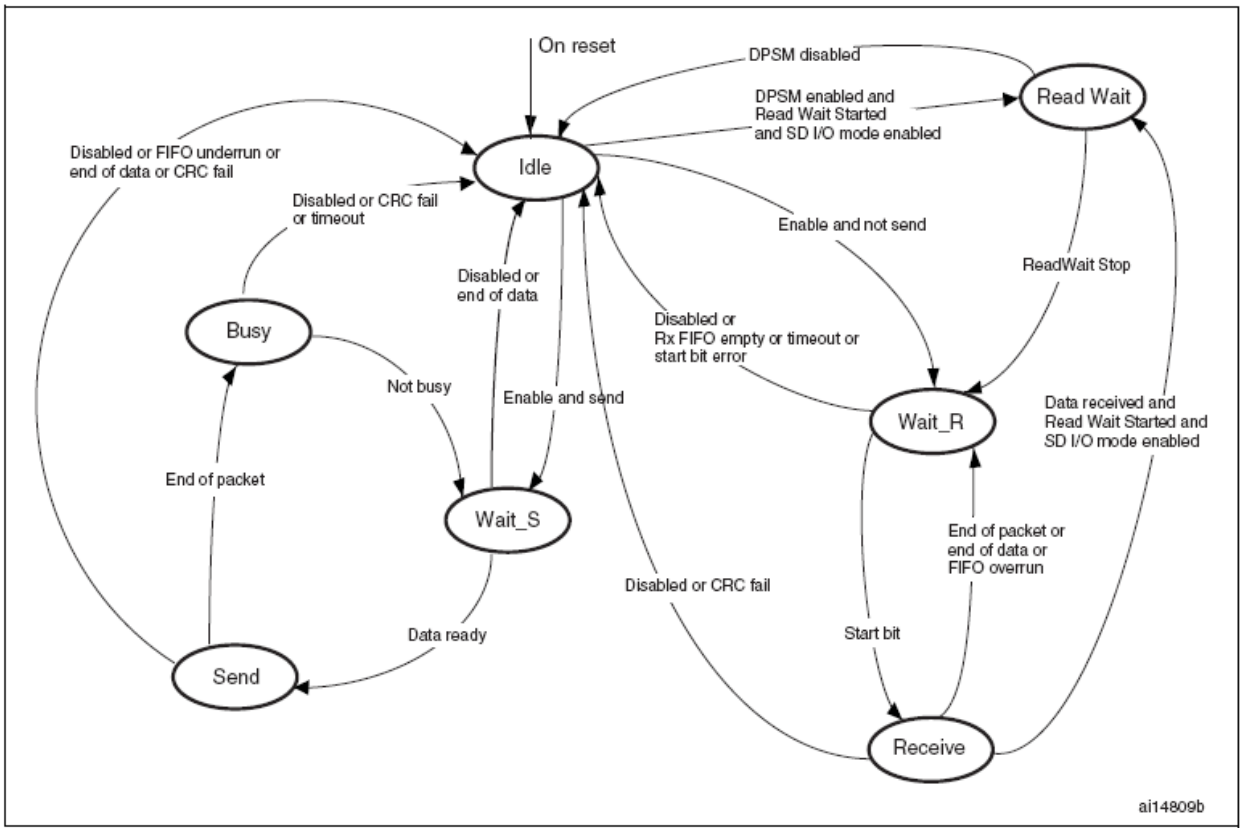
根据传输的方向(发送或接收)，使能时数据通道状态机(DPSM)将进入Wait_S或Wait_R状态：

- 发送：DPSM进入Wait_S状态。如果发送FIFO中有数据，则DPSM进入发送状态，同时数据通道子单元开始向卡发送数据。
- 接收：DPSM进入Wait_R状态并等待开始位；当收到开始位时，DPSM进入接收状态，同时数据通道子单元开始从卡接收数据。

数据通道状态机

DPSM工作在SDIO_CK频率，卡总线信号与SDIO_CK的上升沿同步。DPSM有6个状态，如下图所示：

图187 数据通道状态机(DPSM)



- 空闲：数据通道不工作，SDIO_D[7:0]输出处于高阻状态。当写入数据控制寄存器并设置使能位时，DPSM为数据计数器加载新的数值，并依据数据方向位进入Wait_S或Wait_R状态。
- Wait_R：如果数据计数器等于0，当接收FIFO为空时DPSM进入到空闲状态。如果数据计数器不等于0，DPSM等待SDIO_D上的开始位。如果DPSM在超时之前接收到一个开始位，它会进入接收状态并加载数据块计数器。如果DPSM在检测到一个开始位前出现超时，或发生开始位错误，DPSM将进入空闲状态并设置超时状态标志。
- 接收：接收到的串行数据被组合为字节并写入数据FIFO。根据数据控制寄存器中传输模式位的设置，数据传输模式可以是块传输或流传输：
  - 在块模式下，当数据块计数器达到 0 时，DPSM 等待接收 CRC 码，如果接收到的代码与内部产生的 CRC 码匹配，则 DPSM 进入 Wait_R 状态，否则设置 CRC 失败状态标志同时 DPSM 进入到空闲状态。
  - 在流模式下，当数据计数器不为 0 时，DPSM 接收数据；当计数器为 0 时，将移位寄存器中的剩余数据写入数据 FIFO，同时 DPSM 进入 Wait_R 状态。
    如果产生了 FIFO 上溢错误，DPSM 设置 FIFO 的错误标志并进入空闲状态。
- Wait_S：如果数据计数器为0，DPSM进入空闲状态；否则DPSM等待数据FIFO空标志消失后，进入发送状态。

注： *DPSM会在Wait_S状态保持至少2个时钟周期，以满足N_{WR}的时序要求，N_{WR}是接收到卡的响应至主机开始数据传输的间隔。*

- 发送：DPSM开始发送数据到卡设备。根据数据控制寄存器中传输模式位的设置，数据传输模式可以是块传输或流传输：
  - 在块模式下，当数据块计数器达到 0 时，DPSM 发送内部产生的 CRC 码，然后是结束位，并进入繁忙状态。
  - 在流模式下，当使能位为高同时数据计数器不为 0 时，DPSM 向卡设备发送数据，然后进入空闲状态。
    如果产生了 FIFO 下溢错误，DPSM 设置 FIFO 的错误标志并进入空闲状态。

- 繁忙：DPSM等待CRC状态标志：
  - 如果没有接收到正确的 CRC 状态，则 DPSM 进入空闲状态并设置 CRC 失败状态标志。
  - 如果接收到正确的 CRC 状态，则当 SDIO_D0 不为低时(卡不繁忙)DPSM 进入 Wait_S 状态。
    当 DPSM 处于繁忙状态时发生了超时，DPSM 则设置数据超时标志并进入空闲状态。
    当 DPSM 处于 Wait_R 或繁忙状态时，数据定时器被使能，并能够产生数据超时错误：
  - 发送数据时，如果 DPSM 处于繁忙状态超过程序设置的超时间隔，则产生超时。
  - 接收数据时，如果未收完所有数据，并且 DPSM 处于 Wait_R 状态超过程序设置的超时间隔，则产生超时。
- 数据：数据可以从主机传送到卡，也可以反向传输。数据在数据线上传输。数据存储在一个32字的FIFO中，每个字为32位宽。

表107 数据格式

说明	开始位	数据	CRC16	结束位
块数据	0	-	有	1
流数据	0	-	无	1

数据FIFO

数据FIFO(先进先出)子单元是一个具有发送和接收单元的数据缓冲区。

FIFO包含一个每字32位宽、共32个字的数据缓冲区，和发送与接收电路。因为数据FIFO工作在AHB时钟区域(HCLK/2)，所有与SDIO时钟区域(SDIOCLK)连接的信号都进行了重新同步。

依据TXACT和RXACT标志，可以关闭FIFO、使能发送或使能接收。TXACT和RXACT由数据通道子单元设置而且是互斥的：

- 当 TXACT 有效时，发送 FIFO 代表发送电路和数据缓冲区
- 当 RXACT 有效时，接收 FIFO 代表接收电路和数据缓冲区

- 发送FIFO：当使能了SDIO的发送功能，数据可以通过AHB接口写入发送FIFO。
  发送FIFO有32个连续的地址。发送FIFO中有一个数据输出寄存器，包含读指针指向的数据字。当数据通道子单元装填了移位寄存器后，它移动读指针至下个数据并传输出数据。
  如果未使能发送FIFO，所有的状态标志均处于无效状态。当发送数据时，数据通道子单元设置TXACT为有效。

表108 发送FIFO状态标志

标志	说明
TXFIFOF	当所有32个发送FIFO字都有有效的数据时，该标志为高。
TXFIFOE	当所有32个发送FIFO字都没有有效的数据时，该标志为高。
TXFIFOHE	当8个或更多发送FIFO字为空时，该标志为高。该标志可以作为DMA请求。
TXDAVL	当发送FIFO包含有效数据时，该标志为高。该标志的意思刚好与TXFIFOE相反。
TXUNDERR	当发生下溢错误时，该标志为高。写入SDIO清除寄存器时清除该标志。

- 接收FIFO：当数据通道子单元接收到一个数据字，它会把数据写入FIFO，写操作结束后，写指针自动加一；在另一端，有一个读指针始终指向FIFO中的当前数据。如果关闭了接收FIFO，所有的状态标志会被清除，读写指针也被复位。在接收到数据时数据通道子单元设置RXACT。下表列出了接收FIFO的状态标志。通过32个连续的地址可以访问接收FIFO。

表109 接收FIFO状态标志

标志	说明
RXFIFOF	当所有32个接收FIFO字都有有效的数据时，该标志为高。
RXFIFOE	当所有32个接收FIFO字都没有有效的数据时，该标志为高。
RXFIFOHF	当8个或更多接收FIFO字有有效的数据时，该标志为高。该标志可以作为DMA请求。

RXDAVL	当接收FIFO包含有效数据时，该标志为高。该标志的意思刚好与RTXFIFOE相反。
RXOVERR	当发生上溢错误时，该标志为高。写入SDIO清除寄存器时清除该标志。

19.3.2 SDIO AHB接口

AHB接口产生中断和DMA请求，并访问SDIO接口寄存器和数据FIFO。它包含一个数据通道、寄存器译码器和中断/DMA控制逻辑。

SDIO中断

当至少有一个选中的状态标志为高时，中断控制逻辑产生中断请求。有一个屏蔽寄存器用于选择可以产生中断的条件，如果设置了相应的屏蔽标志，则对应的状态标志可以产生中断。

SDIO/DMA接口：在SDIO和存储器之间数据传输的过程

在下面的例子中，从主机控制器使用CMD24(WRITE_BLOCK)传送512字节到MMC卡，DMA控制器用于从存储器向SDIO的FIFO填充数据。

1. 执行卡识别过程
2. 提高SDIO_CK频率
3. 发送CMD7命令选择卡
4. 按下述步骤配置DMA2:
   a) 使能DMA2控制器并清除所有的中断标志位
   b) 设置DMA2通道4的源地址寄存器为存储器缓冲区的基地址，DMA2通道4的目标地址寄存器为SDIO_FIFO寄存器的地址
   c) 设置DMA2通道4控制寄存器(存储器递增，非外设递增，外设和源的数据宽度为字宽度)
   d) 使能DMA2通道4
5. 发送CMD24(WRITE_BLOCK)，操作如下：
   a) 设置SDIO数据长度寄存器(SDIO数据时钟寄存器应该在执行卡识别过程之前设置好)
   b) 设置SDIO参数寄存器为卡中需要传送数据的地址
   c) 设置SDIO命令寄存器：CmdIndex置为24(WRITE_BLOCK)；WaitRest置为1(SDIO卡主机等待响应)；CPSMEN置为1(使能SDIO卡主机发送命令)，保持其它域为他们的复位值。
   d) 等待SDIO_STA[6]=CMDREND中断，然后设置SDIO数据寄存器：DTEN置为1(使能SDIO卡主机发送数据)；DTDIR置为0(控制器至卡方向)；DTMODE置为0(块数据传送)；DMAEN置为1(使能DMA)；DBLOCKSIZE置为9(512字节)；其它域不用设置。
   e) 等待SDIO_STA[10]=DBCKEND
6. 查询DMA通道的使能状态寄存器，确认没有通道仍处于使能状态。

19.4 卡功能描述

19.4.1 卡识别模式

在卡识别模式，主机复位所有的卡、检测操作电压范围、识别卡并为总线上每个卡设置相对地址(RCA)。在卡识别模式下，所有数据通信只使用命令线(CMD)。

19.4.2 卡复位

GO_IDLE_STATE命令(CMD0)是一个软件复位命令，它把多媒体卡和SD存储器置于空闲状态。IO_RW_DIRECT命令(CMD52)复位SD I/O卡。上电后或执行CMD0后，所有卡的输出端都处于高阻状态，同时所有卡都被初始化至一个默认的相对卡地址(RCA=0x0001)和默认的驱动器寄存器设置(最低的速度，最大的电流驱动能力)。

19.4.3 操作电压范围确认

所有的卡都可以使用任何规定范围内的电压与SDIO卡主机通信，可支持的最小和最大电压V_{DD}数值由卡上的操作条件寄存器(OCR)定义。

内部存储器存储了卡识别号(CID)和卡特定数据(CSD)的卡，仅能在数据传输V_{DD}条件下传送这些信息。当SDIO卡主机模块与卡的V_{DD}范围不一致时，卡将不能完成识别周期，也不能发送CSD数据；因此，在V_{DD}范围不匹配时，SDIO卡主机可以用下面几个特殊命令去识别和拒绝卡：SEND_OP_COND(CMD1)、SD_APP_OP_COND(SD存储卡的ACMD41)和IO_SEND_OP_COND(SD I/O卡的CMD5)。SDIO卡主机在执行这几个命令时会产生需要的V_{DD}电压。不能在指定的电压范围进行数据传输的卡，将从总线断开并进入非激活状态。

使用这些不包含电压范围作为操作数的命令，SDIO卡主机能够查询每个卡并在确定公共的电压范围前，把不在此范围内的卡置于非激活状态。当SDIO卡主机能够选择公共的电压范围或用户需要知道卡是否能用时，SDIO卡主机可以进行这样的查询。

19.4.4 卡识别过程

多媒体卡和SD卡的卡识别过程是有区别的；对于多媒体卡，卡识别过程以时钟频率F_{od}开始，所有SDIO_CMD输出为开路驱动，允许在这个过程中的卡的并行连接，识别过程如下：

1. 总线被激活
2. SDIO卡主机广播发送SEND_OP_COND(CMD1)命令，并接收操作条件
3. 得到的响应是所有卡的操作条件寄存器内容的“线与”
4. 不兼容的卡会被置于非激活状态
5. SDIO卡主机广播发送ALL_SEND_CID(CMD2)至所有激活的卡
6. 所有激活的卡同时串行地发送他们的CID号，那些检测到输出的CID位与命令线上的数据不相符的卡必须停止发送，并等待下一个识别周期。最终只有一个卡能够成功地传送完整的CID至SDIO卡主机并进入识别状态。
7. SDIO卡主机发送SET_RELATIVE_ADDR(CMD3)命令至这个卡，这个新的地址被称为相对卡地址(RCA)，它比CID短，用于对卡寻址。至此，这个卡转入待机状态，并不再响应新的识别过程，同时它的输出驱动从开路转变为推挽模式。
8. SDIO卡主机重复上述步骤5至7，直到收到超时条件。

对于SD卡而言，卡识别过程以时钟频率F_{od}开始，所有SDIO_CMD输出为推挽驱动而不是开路驱动，识别过程如下：

1. 总线被激活
2. SDIO卡主机广播发送SEND_APP_OP_COND(ACMD41)命令
3. 得到的响应是所有卡的操作条件寄存器的内容
4. 不兼容的卡会被置于非激活状态
5. SDIO卡主机广播发送ALL_SEND_CID(CMD2)至所有激活的卡
6. 所有激活的卡发送回他们唯一卡识别号(CID)并进入识别状态。
7. SDIO卡主机发送SET_RELATIVE_ADDR(CMD3)命令和一个地址到一个激活的卡，这个新的地址被称为相对卡地址(RCA)，它比CID短，用于对卡寻址。至此，这个卡转入待机状态。SDIO卡主机可以再次发送该命令更改RCA，卡的RCA将是最后一次的赋值。
8. SDIO卡主机对所有激活的卡重复上述步骤5至7。

对于SD I/O卡而言，卡识别过程如下：

1. 总线被激活
2. SDIO卡主机发送IO_SEND_OP_COND(CMD5)命令
3. 得到的响应是卡的操作条件寄存器的内容
4. 不兼容的卡会被置于非激活状态

5. SDIO卡主机发送SET_RELATIVE_ADDR(CMD3)命令和一个地址到一个激活的卡，这个新的地址被称为相对卡地址(RCA)，它比CID短，用于对卡寻址。至此，这个卡转入待机状态。SDIO卡主机可以再次发送该命令更改RCA，卡的RCA将是最后一次的赋值。

19.4.5 写数据块

执行写数据块命令(CMD24-27)时，主机把一个或多个数据块从主机传送到卡中，同时在每个数据块的末尾传送一个CRC码。一个支持写数据块命令的卡应该始终能够接受由WRITE_BL_LEN定义的数据块。如果CRC校验错误，卡通过SDIO_D线指示错误，传送的数据被丢弃而不被写入，所有后续(在多块写模式下)传送的数据块将被忽略。

如果主机传送部分数据而累计的数据长度未与数据块对齐，当不允许块错位(未设置CSD的参数WRITE_BLK_MISALIGN)，卡将在第一个错位的块之前检测到块错位错误(设置状态寄存器中的ADDRESS_ERROR错误位)。当主机试图写一个写保护区域时，写操作也会被中止，此时卡会设置WP_VIOLATION位。

设置CID和CSD寄存器不需要事先设置块长度，传送的数据也是通过CRC保护的。如果CSD或CID寄存器的部分是存储在ROM中，则这个不能更改的部分必须与接收缓冲区的对应部分相一致，如果有不一致之处，卡将报告一个错误同时不修改任何寄存器的内容。有些卡需要长的甚至不可预计的时间完成写一个数据块，在接收一个数据块并完成CRC检验后，卡开始写操作，如果它的写缓冲区已经满并且不能再从新的WRITE_BLOCK命令接受新的数据时，它会把SDIO_D线拉低。主机可以在任何时候使用SEND_STATUS(CMD13)查询卡的状态，卡将返回当前状态。READY_FOR_DATA状态位指示卡是否可以接受新的数据或写操作是否 还在进行。主机可以使用CMD7(选择另一个卡)不选择某个卡，而把这个卡置于断开状态，这样可以释放SDIO_D线而不中断未完成的写操作；当重新选择了一个卡，如果写操作仍然在进行并且写缓冲区仍不能使用，它会重新通过拉低SDIO_D指示忙的状态。

19.4.6 读数据块

在读数据块模式下，数据传输的基本单元是数据块，它的大小在CSD中(READ_BL_LEN)定义。如果设置了READ_BL_PARTIAL，同样可以传送的较小数据块，较小数据块是指开始和结束地址完全包含在一个物理块中，READ_BL_LEN定义了物理块的大小。为保证数据传输的正确，每个数据块后都有一个CRC校验码。CMD17(READ_SINGLE_BLOCK)启动一次读数据块操作，在传输结束后卡返回到发送状态。

CMD18(READ_MULTIPLE_BLOCK)启动一次连续多个数据块的读操作。

主机可以在多数据块读操作的任何时候中止操作，而不管操作的类型。发送停止传输命令即可中止操作。

如果在多数据块读操作中(任一种类型)卡检测到错误(例如：越界，地址错位或内部错误)，它将停止数据传输并仍处于数据状态；此时主机必须发送停止传输命令中止操作。在停止传输命令的响应中报告读错误。

如果主机发送停止传输命令时，卡已经传输完一个确定数目的多个数据块操作中的最后一个数据块，因为此时卡已经不在数据状态，主机会得到一个非法命令的响应。如果主机传输部分数据块，而累计的数据长度不能与物理块对齐同时不允许块错位，卡会在出现第一个未对齐的块时检测出一个块对齐错误，并在状态寄存器中设置ADDRESS_ERROR错误标志。

19.4.7 数据流操作，数据流写入和数据流读出(只适用于多媒体卡)

在数据流模式，数据按字节传输同时每个数据块

数据流写(只适用于多媒体卡)

WRITE_DAT_UNTIL_STOP(CMD20)开始从SDIO卡主机至卡的数据传输，从指定的地址开始连续传输直到SDIO卡主机发出一个停止命令。如果允许部分数据块传输(设置了CSD参数WRITE_BL_PARTIAL)，则数据流可以在卡的地址空间中的任意地址开始和停止，否则数据流只能在数据块的边界开始和停止。因为传输的数据数目没有事先设定，不能使用CRC校验。如

果发送数据时达到了存储器的最大地址，即使SDIO卡主机没有发送停止命令，随后传输的数据也会被丢弃。

数据流写操作的最大时钟频率可以通过下式计算

$$Maximumspeed = Min(\ TRANSPEED,\ \frac{(8 \times 2^{writebllen})(-NSAC)}{TAAC \times R2WFACTOR}\ )$$

- Maximumspeed = 最大写频率
- TRANSPEED = 最大数据传输率
- writebllen = 最大写数据块长度
- NSAC = 以CLK周期计算的数据读操作时间2
- TAAC = 数据读操作时间1
- R2WFACTOR = 写速度因子

如果主机试图使用更高的频率，卡可能不能处理数据并停止编程，同时在状态寄存器中设置OVERRUN错误位，丢弃所有随后传输的数据并(在接收数据状态)等待停止命令。如果主机试图写入一个写保护区域，写操作将被中止，同时卡将设置WP_VIOLATION位。

数据流读(只适用于多媒体卡)

READ_DAT_UNTIL_STOP(CMD11)数据流数据传输。

这个命令要求卡从指定的地址读出数据，直到SDIO卡主机发送STOP_TRANSMISSION (CMD12)。因为穿行命令传输的延迟，停止命令的执行会有延迟，数据传送会在停止命令的结束位后停止。如果发送数据时达到了存储器的最大地址，SDIO卡主机没有发送停止命令，随后传输的数据将是无效数据。

数据流读操作的最大时钟频率可以通过下式计算

$$Maximumspeed = Min(\ TRANSPEED,\ \frac{(8 \times 2^{readbllen})(-NSAC)}{TAAC \times R2WFACTOR}\ )$$

- Maximumspeed = 最大写频率
- TRANSPEED = 最大数据传输率
- readbllen = 最大读数据块长度
- NSAC = 以CLK周期计算的数据读操作时间2
- TAAC = 数据读操作时间1
- R2WFACTOR = 写速度因子

如果主机试图使用更高的频率，卡将不能处理数据传输，此时卡在状态寄存器中设置UNDERRUN错误位，中止数据传输并在数据状态等待停止命令。

19.4.8 擦除：成组擦除和扇区擦除

多媒体卡的擦除单位是擦除组，擦除组是以写数据块计算，写数据块是卡的基本写入单位。擦除组的大小是卡的特定参数，在CSD中定义。

主机可以擦除一个连续范围的擦除组，开始擦除操作有三个步骤。

首先，主机使用ERASE_GROUP_START(CMD35)命令定义连续范围的的开始地址，然后使用ERASE_GROUP_END(CMD36)命令定义连续范围的的结束地址，最后发送擦除命令ERASE(CMD38)开始擦除操作。擦除命令的地址域是以字节为单位的擦除组地址。卡会舍弃未与擦除组大小对齐的部分，把地址边界对齐到擦除组的边界。

如果未按照上述步骤收到了擦除命令，卡在状态寄存器中设置ERASE_SEQ_ERROR位，并重新等待第一个步骤。

如果收到了除SEND_STATUS和擦除命令之外的其它命令，卡在状态寄存器中设置ERASE_RESET位，解除擦除序列并执行新的命令。

如果擦除范围包含了写保护数据块，这些块不被擦除，只有未保护的块被擦除，同时卡在状态寄存器中设置WP_ERASE_SKIP状态位。

在擦除过程中，卡拉低SDIO_D信号。实际的擦除时间可能很长，主机可以使用CMD7解除卡的选择。

19.4.9 宽总线选择和解除选择

可以通过SET_BUS_WIDTH(ACMD6)命令选择或不选择宽总线(4位总线宽度)操作模式，上电后或GO_IDLE_STATE(CMD0)命令后默认的总线宽度为1位。SET_BUS_WIDTH(ACMD6)命令仅在传输状态时有效，即只有在使用SELECT/DESELECT_CARD(CMD7)命令选择了卡后才能改变总线宽度。

19.4.10 保护管理

SDIO卡主机模块支持三种保护方式：

1. 内部卡保护(卡内管理)
2. 机械写保护开关(仅由SDIO卡主机模块管理)
3. 密码管理的卡锁操作

内部卡的写保护

卡的数据可以被保护不被覆盖或擦除。在CSD中永久地或临时地设置写保护位，生产厂商或内容提供商可以永久地对整个卡施行写保护。对于支持在CSD中设置WP_GRP_ENABLE位从而提供一组扇区写保护的卡，部分数据可以被保护，写保护可以通过程序改变。写保护的基本单位是CSD参数WP_GRP_SIZE个扇区。SET_WRITE_PROT和CLR_WRITE_PROT命令控制指定组的保护，SEND_WRITE_PROT命令与单数据块读命令类似，卡送出一个包含32个写保护位(代表从指定地址开始的32个写保护组)的数据块，跟着一个16位的CRC码。写保护命令的地址域是一个以字节为单位的组地址。

卡将截断所有组大小以下的地址。

机械写保护开关

在卡的侧面有一个机械的滑动开关，允许用户设置或清除卡的写保护。当滑动开关置于小窗口打开的位置时，卡处于写保护状态，当滑动开关置于小窗口关闭的位置时，可以更改卡中内容。在卡的插槽上的对应部位也有一个开关指示SDIO卡主机模块，卡是否处于写保护状态。卡的内部电路不知道写保护开关的位置。

密码保护

密码保护功能允许SDIO卡主机模块使用密码对卡实行上锁或解锁。密码存储在128位的PWD寄存器中，它的长度设置在8位的PWD_LEN寄存器中。这些寄存器是不可挥发的，即掉电后它们的内容不丢失。已上锁的卡能够响应和执行相应的命令，即允许SDIO卡主机模块执行复位、初始化和查询状态等操作，但不允许操作卡中的数据。当设置了密码后(即PWD_LEN的数值不为0)，上电后卡自动处于上锁状态。正如CSD和CID寄存器写命令，上锁/解锁命令仅在传输状态下有效，在这个状态下，命令中没有地址参数，但卡已经被选中。卡的上锁/解锁命令具有单数据块写命令的结构和总线操作类型，传输的数据块包含所有命令所需要的信息(密码设置模式、PWD内容和上锁/解锁指示)。在发送卡的上锁/解锁命令之前，命令数据块的长度由SDIO卡主机模块定义，，命令结构示于表123。

位的设置如下：

- ERASE：设置该位将执行强制擦除，所有其它位必须为0，只发送命令字节。
- LOCK_UNLOCK：设置该位锁住卡，LOCK_UNLOCK与SET_PWD可以同时设置，但不能与CLR_PWD同时设置。

- CLR_PWD：设置该位清除密码数据。
- SET_PWD：设置该位将密码数据保存至存储器。
- PWD_LEN：以字节为单位定义密码的长度。
- PWD：密码(依不同的命令，新的密码或正在使用的密码)

以下几节列出了设置/清除密码、上锁/解锁和强制擦除的命令序列。

设置密码

1. 选择一个卡(SELECT/DESELECT_CARD，CMD7)。
2. 定义要在8位的卡上锁/解锁模式下发送的数据块长度(SET_BLOCKLEN，CMD16)，8位的PWD_LEN，新密码的字节数目。当更换了密码后，发送命令的数据块长度必须同时考虑新旧密码的长度。
3. 以合适的数据块长度在数据线上发送LOCK/UNLOCK(CMD42)命令，并包含16位的CRC码。数据块包含了操作模式(SET_PWD=1)、长度(PWD_LEN)和密码(PWD)。当更换了密码后，长度数值(PWD_LEN)包含了新旧两个密码的长度，PWD域包含了旧的密码(正在使用的)和新的密码。
4. 当旧的密码匹配后，新的密码和它的长度被分别存储在PWD和PWD_LEN域。如果送出的旧密码与期望的密码(长度或内容)不吻合，则设置状态寄存器中的LOCK_UNLOCK_FAILED错误位，同时密码不变。

密码长度域(PWD_LEN)指示当前是否设置了密码，如果该域为非零，则表示使用了密码，卡在上电时自动上锁。在不断电的情况下，如果设置了密码，可以通过设置LOCK_UNLOCK位或发送一个额外的上锁命令，立即锁住卡。

复位密码

1. 选择一个卡(SELECT/DESELECT_CARD，CMD7)。
2. 定义要在8位的卡上锁/解锁模式下发送的数据块长度(SET_BLOCKLEN，CMD16)，8位的PWD_LEN，当前使用密码的字节数目。
3. 当密码匹配后，PWD域被清除同时PWD_LEN被设为0。如果送出的密码与期望的密码(长度或内容)不吻合，则设置状态寄存器中的LOCK_UNLOCK_FAILED错误位，同时密码不变。

卡上锁

1. 选择一个卡(SELECT/DESELECT_CARD，CMD7)
2. 定义要在8位的卡上锁/解锁模式(见表123的字节0)下发送的数据块长度(SET_BLOCKLEN，CMD16)，8位的PWD_LEN，和当前密码的字节数目。
3. 以合适的数据块长度在数据线上发送LOCK/UNLOCK(CMD42)命令，并包含16位的CRC码。数据块包含了操作模式(LOCK_UNLOCK=1)、长度(PWD_LEN)和密码(PWD)。
4. 当密码匹配后，卡被上锁并则设置状态寄存器中的CARD_IS_LOCKED状态位。如果送出的密码与期望的密码(长度或内容)不吻合，则设置状态寄存器中的LOCK_UNLOCK_FAILED错误位，同时上锁操作失败。

设置密码和为卡上锁可以在同一个操作序列中进行，此时SDIO卡主机模块按照前述的步骤设置密码，但在发送新密码命令的第3步需要设置LOCK_UNLOCK位。

如果曾经设置过密码(PWD_LEN不为0)，卡会在上电复位时自动地上锁。对已经上锁卡执行上锁操作或对没有密码的卡执行上锁操作会导致失败，并设置状态寄存器中的LOCK_UNLOCK_FAILED错误位。

卡解锁

1. 选择一个卡(SELECT/DESELECT_CARD，CMD7)
2. 定义要在8位的卡上锁/解锁模式(见表123的字节0)下发送的数据块长度(SET_BLOCKLEN，CMD16)，8位的PWD_LEN，和当前密码的字节数目。

3. 以合适的数据块长度在数据线上发送LOCK/UNLOCK(CMD42)命令，并包含16位的CRC码。数据块包含了操作模式(LOCK_UNLOCK=0)、长度(PWD_LEN)和密码(PWD)。
4. 当密码匹配后，卡锁被解除，同时状态寄存器中的CARD_IS_LOCKED位被清除。如果送出的密码与期望的密码(长度或内容)不吻合，则设置状态寄存器中的LOCK_UNLOCK_FAILED错误位，同时卡仍保持上锁状态。

解锁状态只在当前的供电过程中有效，只要不清除PWD域，下次上电后卡会被自动上锁。

试图对已经解了锁的卡执行解锁操作会导致操作失败，并设置状态寄存器中的LOCK_UNLOCK_FAILED错误位。

强制擦除

如果用户忘记了密码(PWD的内容)，可以在清除卡中的所有内容后使用卡。强制擦除操作擦除所有卡中的数据和密码。

1. 选择一个卡(SELECT/DESELECT_CARD，CMD7)
2. 设置下发送的数据块长度(SET_BLOCKLEN，CMD16)为1，仅发送8位的卡上锁/解锁字节(见表123的字节0)。
3. 以合适的数据块长度在数据线上发送LOCK/UNLOCK(CMD42)命令，并包含16位的CRC码。数据块包含了操作模式(ERASE=1)所有其它位为0。
4. 当ERASE位为数据域仅有的域时，卡中的所有内容将被擦除，包括PWD和PWD_LEN域，同时卡不再被上锁。如果有任何其它为不为0，则设置状态寄存器中的LOCK_UNLOCK_FAILED错误位，卡中的数据保持不变，同时卡仍保持上锁状态。

试图对已经解了锁的卡执行擦除操作会导致操作失败，并设置状态寄存器中的LOCK_UNLOCK_FAILED错误位。

19.4.11 卡状态寄存器

响应格式R1包含了一个32位的卡状态域，这个域是用于向卡主机发送卡的状态信息(这些信息有可能存在本地的状态寄存器中)。除非特别说明，卡返回的状态始终是与之前的命令相关的。

表110定义了不同的状态信息。表中有关类型和清除条件域的缩写定义如下：

类型：

- E: 错误位
- S: 状态位
- R: 检测位，并依据实际的命令响应而设置
- X: 检测位，在命令的执行中设置。SDIO卡主机通过发送状态命令读出这些位而查询卡的状态。

清除条件：

- A: 依据卡的当前状态
- B: 始终与之前的命令相关。接收到正确的命令即可清除(具有一个命令的延迟)。
- C: 读即可清除

表110 卡状态

位	名称	类型	数值	说明	清除条件
31	ADDRESS_OUT_OF_RANGE	E R X	‘0’= 无错误 ‘1’= 错误	命令中的地址参数超出了卡的允许范围。 一个多数据块或数据流读/写操作(即使从一个合法的地址开始)试图读或写超出卡的容量的部分。	C
30	ADDRESS_MISALIGN		‘0’= 无错误 ‘1’= 错误	命令中的地址参数(与当前的数据块长度对照)定义的第一个数据块未与卡的物理块对齐。 一个多数据块或数据流读/写操作(即使从一个合法的地址开始)试图读或写未与物理块对齐的数据块。	C

29	BLOCK_LEN_ERROR		‘0’= 无错误 ‘1’= 错误	SET_BLOCKLEN命令的参数超出了卡的最大允许范围，或先前定义的数据块长度对于当前命令来说是非法的(例如：主机发出一个写命令，当前的块长度小于卡所允许的最小长度，同时又不允许写入部分数据块)。	C
28	ERASE_SEQ_ERROR		‘0’= 无错误 ‘1’= 错误	发送擦除命令的顺序错误。	C
27	ERASE_PARAM	E X	‘0’= 无错误 ‘1’= 错误	擦除时选择了非法的擦除组。	C
26	WP_VIOLATION	E X	‘0’= 无错误 ‘1’= 错误	试图对一个写保护的数据块编程。	C
25	CARD_IS_LOCKED	S R	‘0’= 卡未锁 ‘1’= 卡已锁	当设置了该位，表示卡已经被锁住。	A
24	LOCK_UNLOCK_FAILED	E X	‘0’= 无错误 ‘1’= 错误	在上锁/解锁中有命令的顺序错误或检测到密码错误。	C
23	COM_CRC_ERROR	E R	‘0’= 无错误 ‘1’= 错误	之前的命令中CRC校验错误。	B
22	ILLEGAL_COMMAND	E R	‘0’= 无错误 ‘1’= 错误	对于当前的卡状态，命令非法。	B
21	CARD_ECC_FAILED	E X	‘0’= 成功 ‘1’= 失败	卡的内部实施了ECC校验，但在更正数据时失败。	C
20	CC_ERROR	E R	‘0’= 无错误 ‘1’= 错误	(标准中未定义)卡内部发生错误，与主机的命令无关。	C
19	ERROR	E X	‘0’= 无错误 ‘1’= 错误	产生了与执行上一个主机命令相关的(标准中未定义)卡内部的错误(例如：读或写错误)。	C
18	保留				
17	保留				
16	CID/CSD_OVERWRITE	E X	‘0’= 无错误 ‘1’= 错误	可以是任何一个下述的错误： ■ 已经写入了CID寄存器，不能覆盖 ■ CSD的只读部分与卡的内容不匹配 ■ 试图进行拷贝或永久写保护的反向操作，即恢复原状或解除写保护。	C
15	WP_ERASE_SKIP	E X	‘0’= 未保护 ‘1’= 已保护	遇到已经存在的写保护数据块，仅有部分地址空间被擦除时。	C
14	CARD_ECC_DISABLED	S X	‘0’= 允许 ‘1’= 不允许	执行命令时没有使用内部的ECC。	A
13	ERASE_RESET		‘0’= 清除 ‘1’= 设置	因为收到一个擦除顺序之外的命令(非CMD35、CMD36、CMD38或CMD13命令)，进入擦除过程的序列被中止。	C

12:9	CURRENT_STATE	S R	0 = 空闲 1 = 就绪 2 = 识别 3 = 待机 4 = 发送 5 = 数据 6 = 接收 7 = 编程 8 = 断开 9 = 忙测试 10~15 = 保留	当收到命令时卡的状态机的状态。如果命令的执行导致状态的变化，这个变化将会在下个命令的响应中反映出来。这四个位按一个0至15的数解释。	B
8	READY_FOR_DATA	S R	‘0’= 未就绪 ‘1’= 就绪	与总线上的缓冲器空的信号相对应。	
7	SWITCH_ERROR	E X	‘0’= 无错误 ‘1’= 转换错	卡没有按照SWITCH命令的要求转换到希望的模式。	B
6	保留				
5	APP_CMD	S R	‘0’= 不允许 ‘1’= 允许	卡期望ACMD，或指示命令已经被解释为ACMD命令。	C
4	保留给SD I/O卡				
3	AKE_SEQ_ERROR	E R	‘0’= 无错误 ‘1’= 错误	验证的顺序有错误。	C
2	保留给与应用相关的命令。				
1,0	保留给生产厂家的测试模式。				

19.4.12 SD状态寄存器

SD状态包含与SD存储器卡特定功能相关的状态位和一些与未来应用相关的状态位，SD状态的长度是一个512位的数据块。收到ACMD13命令(CMD55，然后是CMD13)后，这个寄存器的内容被传送到SDIO卡主机。只有卡处于传输状态时(卡已被选择)才能发送ACMD13命令。

表111定义了不同的SD状态寄存器信息。表中有关类型和清除条件域的缩写定义如下：

类型：

- E: 错误位
- S: 状态位
- R: 检测位，并依据实际的命令响应而设置
- X: 检测位，在命令的执行中设置。SDIO卡主机通过发送状态命令读出这些位而查询卡的状态。

清除条件：

- A: 依据卡的当前状态
- B: 始终与之前的命令相关。接收到正确的命令即可清除(具有一个命令的延迟)。
- C: 读即可清除

表111 SD状态

位	名称	类型	数值	说明	清除条件
511:510	DAT_BUS_WIDTH	S R	‘00’= 1(默认) ‘01’= 保留 ‘10’= 4位宽 ‘11’= 保留	由SET_BUS_WIDTH命令定义的当前数据总线宽度。	A

509	SECURED_MODE	S R	'0'= 未处于保密模式 '1'= 处于保密模式	卡处于保密操作模式(详见“SD保密规范”)。	A
508:496	保留				
495:480	SD_CARD_TYPE	S R	'00xxh'= 在物理规范版本1.01~2.00的SD存储器卡('x'表示任意值)。已定义的卡有: '0000'= 通用SD读写卡 '0001'= SD ROM卡	这个域的低8位可以在未来定义SD存储卡的不同变种(每个位可以用于定义不同的SD类型)。高8位可以用于定义哪些不遵守当前的SD物理层规范的SD卡。	A
479:448	SIZE_OF_PROTECTED_AREA	S R	受保护的区域大小(见以下说明)	(见以下说明)	A
447:440	SPEED_CLASS	S R	卡的速度类型(见以下说明)	(见以下说明)	A
439:432	PERFORMANCE_MOVE	S R	以1MB/秒为单位的传输性能(见以下说明)	(见以下说明)	A
431:428	AU_SIZE	S R	AU的大小(见以下说明)	(见以下说明)	A
427:424	保留				
423:408	ERASE_SIZE	S R	一次可以擦除的AU数目	(见以下说明)	A
407:402	ERASE_TIMEOUT	S R	擦除 UNIT_OF_ERASE_AU 指定的范围的超时数值	(见以下说明)	A
401:400	ERASE_OFFSET	S R	在擦除时增加的固定偏移数值	(见以下说明)	A
399:312	保留				
311:0	保留给生产厂商				

SIZE_OF_PROTECTED_AREA

标准容量卡和高容量卡设置该位的方式不同。对于标准容量卡，受保护区域的容量由下式计算:

受保护区域 = SIZE_OF_PROTECTED_AREA * MULT * BLOCK_LEN

SIZE_OF_PROTECTED_AREA的单位是MULT * BLOCK_LEN。

对于高容量卡，受保护区域的容量由下式计算:

受保护区域 = SIZE_OF_PROTECTED_AREA

SIZE_OF_PROTECTED_AREA的单位是字节。

SPEED_CLASS

这8位指示速度的类型和可以通过计算$P_W/2$的数值(P_W是写的性能)。

表112 速度类型代码

SPEED_CLASS	数值定义
00h	类型0
01h	类型2
02h	类型4
03h	类型6
04h~FFh	保留

PERFORMANCE_MOVE

这8位以1MB/秒为单位指示移动性能(Pm)。如果卡不用RU(纪录单位)移动数据，应该Pm认为是无穷大。设置这个域为FFh表示无穷大。

表113 移动性能代码

PERFORMANCE_MOVE	数值定义
00h	未定义
01h	1MB/秒
02h	2MB/秒
……	……
FEh	254MB/秒
FFh	无穷大

AU_SIZE

这4位指示AU的长度，数值是16K字节为单位2的幂次的倍数。

表114 AU_SIZE代码

AU_SIZE	数值定义
00h	未定义
01h	16KB
02h	32KB
03h	64KB
04h	128KB
05h	256KB
06h	512KB
07h	1MB
08h	2MB
09h	4MB
Ah~Fh	保留

依据卡的容量，最大的AU长度由下表定义。卡可以在RU长度和最大的AU长度之间设置任意的AU长度。

表115 最大的AU长度

容量	16MB~64MB	128MB~256MB	512MB	1GB~32GB
最大的AU长度	512KB	1MB	2MB	4MB

ERASE_SIZE

这个16位域给出了N_{ERASE}，当N_{ERASE}个AU被擦除时，ERASE_TIMEOUT定义了超时时间。主机应该确定适当的一次操作中擦除的AU数目，这样主机可以显示擦除操作的进度。如果该域为0，则不支持擦除的超时计算。

表116 ERASE_SIZE代码

ERASE_SIZE	数值定义
0000h	不支持擦除的超时计算
0001h	1个AU
0002h	2个AU
0003h	3个AU
……	……
FFFFh	65535个AU

ERASE_TIMEOUT

这6位给出了T_{ERASE}，当ERASE_SIZE指示的多个AU被擦除时，这个数值给出了从偏移量算起的擦除超时。ERASE_TIMEOUT的范围可以定义到最多63秒，卡的生产商可以根据具体实现选

择合适的ERASE_SIZE与ERASE_TIMEOUT的组合，先确定ERASE_TIMEOUT再确定ERASE_SIZE。

表117 擦除超时代码

ERASE_TIMEOUT	数值定义
00	不支持擦除的超时计算
01	1秒
02	2秒
03	3秒
……	……
63	63秒

ERASE_OFFSET

这2位给出了T_{OFFSET}，当ERASE_SIZE和ERASE_TIMEOUT同为0时这个数值没有意义。

表118 擦除偏移代码

ERASE_OFFSET	数值定义
0	0秒
1	1秒
2	2秒
3	3秒

19.4.13 SD I/O模式

SD I/O中断

为了让SD I/O卡能够中断多媒体卡/SD模块，在SD接口上有一个具有中断功能的管脚——第8脚，在4位SD模式下这个脚是SDIO_D1，卡用它向多媒体卡/SD模块提出中断申请。对于每一个卡或卡内的功能，中断功能是可选的。SD I/O的中断是电平有效，即在被识别并得到多媒体卡/SD模块的响应之前，中断信号线必须保持有效电平(低)，在中断过程结束后保持无效电平(高)。在多媒体卡/SD模块服务了中断请求后，通过一个I/O写操作，写入适当的位到SD I/O卡的内部寄存器，即可清除中断状态位。所有SD I/O卡的中断输出是低电平有效，多媒体卡/SD模块在所有数据线(SDIO/D[3:0])上提供上拉电阻。多媒体卡/SD模块在中断阶段对第8脚(SDIO_D/IRQ)采样并进行中断检测，其它时间该信号线上的数值将被忽略。

存储器操作和I/O操作都具有中断阶段，单个数据块操作的中断阶段定义与多个数据块传输操作的中断阶段定义不同。

SD I/O暂停和恢复

在一个多功能的SD I/O卡或同时具有I/O和存储器功能的卡中，多个设备(I/O和存储器)共用MMC/SD总线。为了使MMC/SD模块中的多个设备能够共用总线，SD I/O卡和复合卡可以有选择地实现暂停/恢复的概念；如果一个卡支持暂停/恢复，MMC/SD模块能够暂时地停止一个功能或存储器的数据传输操作(暂停)，借此让出总线给具有更高优先级的其它功能或存储器，在这个具有更高优先级的传输完成后，再恢复原先暂停的传输。支持暂停/恢复的操作是可选的。在MMC/SD总线上执行暂停/恢复操作有下述步骤：

1. 确定SDIO_D[3:0]信号线的当前功能
2. 请求低优先级或慢的操作暂停
3. 等待暂停操作完成，确认设备已暂停
4. 开始高优先级的传输
5. 等待高优先级的传输结束
6. 恢复暂停的操作

SD I/O读等待

可选的读等待(RW)操作只适用于SD卡的1位或4位模式。读等待操作允许MMC/SD模块在一个卡正在读多个寄存器(IO_RW_EXTENDED，CMD53)时，要求它暂时停止数据传输，同时允许MMC/SD模块发送命令到SD I/O设备中的其他功能。判断一个卡是否支持读等待协议，MMC/SD模块应该检测卡的内部寄存器。读等待的时间与中断阶段相关。

19.4.14 命令与响应

应用相关命令和通用命令

SD卡主机模块系统是用于提供一个适用于多种应用类型的标准接口，但同时又要兼顾特定用户和应用的功能，因此标准中定义了两类通用命令：应用相关命令(ACMD)和通用命令(GEN_CMD)。

当卡收到APP_CMD(CMD55)命令时，卡期待下一个命令是应用相关命令。应用相关命令(ACMD)具有普通多媒体卡相同的格式结构，并可以使用相同的CMD号码，因为它是出现在APP_CMD(CMD55)后面，所以卡把它识别为ACMD命令。如果跟随APP_CMD(CMD55)之后不是一个已经定义应用相关命令，则认为它是一个标准命令；例如：有一个SD_STATUS(ACMD13)应用相关命令，如果在紧随APP_CMD(CMD55)之后收到CMD13，它将被解释为SD_STATUS(ACMD13)；但是如果卡在紧随APP_CMD(CMD55)之后收到CMD7，而这个卡没有定义ACMD7，则它将被解释为一个标准的CMD7(SELECT/DESELECT_CARD)命令。

如果要使用生产厂商自定义的ACMD，SD卡主机需要作以下操作：

1. 发送APP_CMD(CMD55)命令
   卡送回响应给多媒体/SD卡模块，指示设置了APP_CMD位并等待ACMD命令。
2. 发送指定的ACMD
   卡送回响应给多媒体/SD卡模块，指示设置了APP_CMD位，收到的命令已经正确地按照ACMD命令解析；如果发送了一个非ACMD命令，卡将按照普通的多媒体卡命令处理同时清除卡中状态寄存器的APP_CMD位。

如果发送了一个非法的命令(不管是ACMD还是CMD)，将被按照标准的非法多媒体卡命令错误处理。

GEN_CMD命令的总线操作过程，与单数据块读写命令(WRITE_BLOCK，CMD24或READ_SINGLE_BLOCK，CMD17)相同；这时命令的参数表示数据传输的方向而不是地址，数据块具有用户自定义的格式和意义。

发送GEN_CMD(CMD56)命令之前，卡必须被选中(状态机处于传输状态)，数据块的长度由SET_BLOCKLEN(CMD16)定义。GEN_CMD(CMD56)命令的响应是R1b格式。

命令类型

应用相关命令和通用命令有四种不同的类型：

1. 广播命令(BC)：发送到所有卡，没有响应返回。
2. 带响应的广播命令(BCR)：发送到所有卡，同时收到从所有卡返回的响应。
3. 带寻址(点对点)的命令(AC)：发送到选中的卡，在SDIO_D信号线上不包括数据传输。
4. 带寻址(点对点)的数据传输命令(AC)：发送到选中的卡，在SDIO_D信号线上包含数据传输。

命令格式

命令格式参见表103。

多媒体卡/SD卡模块的命令

表119 基于块传输的写命令

CMD索引	类型	参数	响应格式	缩写	说明
CMD23	ac	[31:16] = 0 [15:0] = 数据块数目	R1	SET_BLOCK_COUNT	定义在随后的多块读或写命令中需要传输块的数目。
CMD24	adtc	[31:0] = 数据地址	R1	WRITE_BLOCK	按照SET_BLOCKLEN命令选择的长度写一个块。
CMD25	adtc	[31:0] = 数据地址	R1	WRITE_MULTIPLE_BLOCK	收到一个STOP_TRANSMISSION命令或达到了指定的块数目之前，连续地写数据块。
CMD26	adtc	[31:0] = 填充位	R1	PROGRAM_CID	对卡的识别寄存器编程。对于每个卡只能发送一次这个命令。卡中有硬件机制防止多次的编程操作。通常该命令保留给生产厂商。
CMD27	adtc	[31:0] = 填充位	R1	PROGRAM_CSD	对卡的CSD中可编程的位编程。
CMD28	ac	[31:0] = 数据地址	R1b	SET_WRITE_PROT	如果卡有写保护功能，该命令设置指定组的写保护位。写保护特性设置在卡的特殊数据区(WP_GRP_SIZE)。
CMD29	ac	[31:0] = 数据地址	R1b	CLR_WRITE_PROT	如果卡有写保护功能，该命令清除指定组的写保护位。
CMD30	adtc	[31:0] = 写保护数据地址	R1	SEND_WRITE_PROT	如果卡有写保护功能，该命令要求卡发送写保护位的状态。
CMD31	保留				

表120 基于块传输的写保护命令

CMD索引	类型	参数	响应格式	缩写	说明
CMD28	ac	[31:0] = 数据地址	R1b	SET_WRITE_PROT	如果卡具有写保护的功能，该命令设置指定组的写保护位。写保护的属性设置在卡的特定数据域(WP_GRP_SIZE)。
CMD29	ac	[31:0] = 数据地址	R1b	CLR_WRITE_PROT	如果卡具有写保护的功能，该命令清除指定组的写保护位。
CMD30	adtc	[31:0] = 写保护数据地址	R1	SEND_WRITE_PROT	如果卡具有写保护的功能，该命令要求卡发送写保护位的状态。

表121 擦除命令

CMD索引	类型	参数	响应格式	缩写	说明
CMD32 … CMD34	保留。为了与旧版本的对媒体卡协议向后兼容，不能使用这些命令代码。				
CMD35	ac	[31:0] = 数据地址	R1	ERASE_GROUP_START	在选择的擦除范围内，设置第一个擦除组的地址。
CMD36	ac	[31:0] = 数据地址	R1	ERASE_GROUP_END	在选择的连续擦除范围内，设置最后一个擦除组的地址。
CMD37	保留。为了与旧版本的对媒体卡协议向后兼容，不能使用这个命令代码。				
CMD38	ac	[31:0] = 填充位	R1	ERASE	擦除之前选择的数据块。

表122 I/O模式命令

CMD索引	类型	参数	响应格式	缩写	说明
CMD39	ac	[31:16] = RCA [15] = 寄存器写标志 [14:8] = 寄存器地址 [7:0] = 寄存器数据	R4	FAST_IO	用于写和读8位(寄存器)数据域。该命令指定一个卡和寄存器，如果设置了写标志还提供写入的数据。R4响应包含从指定寄存器读出的数据。该命令访问未在多媒体卡标准中定义的与应用相关的寄存器。
CMD40	bcr	[31:0] = 填充位	R5	GO_IRQ_STATE	置系统于中断模式。
CMD41	保留。				

表123 上锁命令

CMD索引	类型	参数	响应格式	缩写	说明
CMD42	adtc	[31:0] = 填充位	R1b	LOCK_UNLOCK	设置/清除密码或对卡上锁/解锁。数据块的长度由SET_BLOCKLEN命令设置。
CMD43 … CMD54	保留。				

表124 应用相关命令

CMD索引	类型	参数	响应格式	缩写	说明
CMD55	ac	[31:16] = RCA [15:0] = 填充位	R1	APP_CMD	指示卡下一个命令是应用相关命令而不是一个标准命令。
CMD56	adtc	[31:1] = 填充位 [0] = RD/WR			在通用或应用相关命令中，或者用于向卡中传输一个数据块，或者用于从卡中读取一个数据块。数据块的长度由SET_BLOCKLEN命令设置。
CMD57 … CMD59	保留。				
CMD60 … CMD63	保留给生产厂商。				

19.5 响应格式

所有的响应是通过MCCMD命令在SDIO_CMD线上传输。响应的传输总是从对应响应字的位串的最左面开始，响应字的长度与 响应的类型相关。

一个响应总是有一个起始位(始终为0)，跟随着传输的方向位(卡=0)。下表中标示为x的数值表示一个可变的部分。除了R3响应类型，所有的响应都有CRC保护。每一个命令码字都有一个结束位(始终为1)。

共有5种响应类型，它们的格式定义如下：

19.5.1 R1(普通响应命令)

代码长度=48位。位45:40指示要响应的命令索引，它的数值介于0至63之间。卡的状态由32位进行编码。

表125 R1响应

位	域宽度	数值	说明
47	1	0	开始位
46	1	0	传输位
[45:40]	6	X	命令索引
[39:8]	32	X	卡状态
[7:1]	7	X	CRC7
0	1	1	结束位

19.5.2 R1b

与R1格式相同，但可以选择在数据线上发送一个繁忙信号。收到这些命令后，依据收到命令之前的状态，卡可能变为繁忙。

19.5.3 R2(CID、CSD寄存器)

代码长度=136位。CID寄存器的内容将作为CMD2和CMD10的响应发出。CSD寄存器的内容将作为CMD9的响应发出。卡只送出CID和CSD的位[127...1]，在接受端这些寄存器的位0被响应的结束位所取代。卡通过拉低MCDAT指示它正在进行擦除操作；实际的擦除操作的时间可能非常长，主机可以发送CMD7命令不选中这个卡。

表126 R2响应

位	域宽度	数值	说明
135	1	0	开始位
134	1	0	传输位
[133:128]	6	‘111111’	命令索引
[127:1]	127	X	卡状态
0	1	1	结束位

19.5.4 R3(OCR寄存器)

代码长度=48位。OCR寄存器的内容将作为CMD1的响应发出。电平代码的定义是：限制的电压窗口 = 低，卡繁忙 = 低。

表127 R3响应

位	域宽度	数值	说明
47	1	0	开始位
46	1	0	传输位
[45:40]	6	‘111111’	保留
[39:8]	32	X	OCR寄存器
[7:1]	7	‘1111111’	保留
0	1	1	结束位

19.5.5 R4(快速I/O)

代码长度=48位。参数域包含指定卡的RCA、需要读出或写入寄存器的地址、和它的内容。

表128 R4响应

位		域宽度	数值	说明
47		1	0	开始位
46		1	0	传输位
[45:40]		6	‘111111’	保留
[39:8]参数域	[31:16]	16	X	RCA
	[15:8]	8	X	寄存器地址
	[7:0]	8	X	读寄存器的内容
[7:1]		7	‘1111111’	CRC7
0		1	1	结束位

19.5.6 R4b

仅适合SD I/O卡：一个SDIO卡收到CMD5后将返回一个唯一的SDIO响应R4。

表129 R4b响应

位		域宽度	数值	说明
47		1	0	开始位
46		1	0	传输位
[45:40]		6	X	保留
[39:8]参数域	39	1	X	卡已就绪
	[38:36]	3	X	I/O功能数目
	35	1	X	当前存储器
	[34:32]	3	X	填充位
	[31:8]	24	X	I/O ORC
[7:1]		7	X	保留
0		1	1	结束位

当一个SD I/O卡收到命令CMD5，卡的I/O部分被使能并能够正常地响应所有后续的命令。I/O卡的使能状态将保持到下一次复位、断电或收到I/O复位的CMD52命令。注意，一个只包含存储器功能的SD卡可以响应CMD5命令，它的正确响应可以是：当前存储器=1，I/O功能数目=0。按照SD存储器卡规范版本1.0设计的只包含存储器功能的SD卡，可以检测到CMD5命令为一个非法命令并不响应它。可以处理I/O卡的主机将发送CMD5命令，如果卡返回响应R4，则主机会依据R4响应中的数据确定卡的配置。

19.5.7 R5(中断请求)

仅适用于多媒体卡。代码长度=48位。如果这个响应有主机产生，则参数中的RCA域为0x0。

表130 R5响应

位		域宽度	数值	说明
47		1	0	开始位
46		1	0	传输位
[45:40]		6	‘111111’	CMD40
[39:8]参数域	[31:16]	16	X	成功的卡或主机的RCA[31:16]
	[15:0]	16	X	未定义。可以作为中断数据。
[7:1]		7	X	CRC7

0	1	1	结束位

19.5.8 R6(中断请求)

仅适用于SD I/O卡。这是一个存储器设备对CMD3命令的正常响应。

表131 R6响应

位		域宽度	数值	说明
47		1	0	开始位
46		1	0	传输位
[45:40]		6	'101000'	CMD40
[39:8]参数域	[31:16]	16	X	成功的卡或主机的RCA[31:16]
	[15:0]	16	X	未定义。可以作为中断数据。
[7:1]		7	X	CRC7
0		1	1	结束位

当CMD3命令送到只有I/O功能的卡时，卡的状态位[23:8]会改变；此时，响应的16位将是只有I/O功能的SD卡中的数值：

- 位15 = COM_CRC_ERROR
- 位14 = ILLEGAL_COMMAND
- 位13 = ERROR
- 位[12:0] = 保留

19.6 SDIO I/O卡特定的操作

下述功能是SD I/O卡特定的操作:

- SDIO读等待操作，由SDIO_D2信号线指示。
- SDIO读等待操作，通过停止时钟实现。
- SDIO暂停/恢复操作(写和读暂停)
- SDIO中断

只有设置了SDIO_DCTRL[11]位时，SDIO才支持这些操作；但读暂停除外，因为它不需要特殊的硬件操作。

19.6.1 使用SDIO_D2 信号线的SDIO I/O读等待操作

在收到第一个数据块之前即可以开始读等待过程，使能数据通道(设置SDIO_DCTRL[0]位)、使能SDIO特定操作(设置SDIO_DCTRL[11]位)、开始读等待(SDIO_DCTRL[10]=0并且SDIO_DCTRL[8]=1)，同时数据传输方向是从卡至SDIO主机(SDIO_DCTRL[1]=1)，DPSM将直接从空闲进入读等待状态。在读等待状态，在2个SDIO_CK时钟周期后，DPSM驱动SDIO_D2为0，在此状态，如果设置RWSTOP位(SDIO_DCTRL[9])，则DPSM会在等待状态多停留2个SDIO_CK时钟周期，(根据SDIO规范)并在一个时钟周期中驱动SDIO_D2为1。然后DPSM开始等待从卡里接收数据。在接收数据块时，即使设置了开始读等待，DPSM也不会进入读等待，读等待过程将在收到CRC后开始。必须清除RWSTOP才能开始新的读等待操作。在读等待期间，SDIO主机可以在SDIO_D1上监测SDIO中断。

19.6.2 使用停止SDIO_CK的SDIO读等待操作

如果SDIO卡不能支持前述的读等待操作，SDIO可以停止SDIO_CK进入读等待(按照19.6.1节介绍的方式设置SDIO_DCTRL，但置SDIO_DCTRL[10]=1)，在接收当前数据块结束位之后的2个SDIO_CK周期后，DSPM停止时钟，在设置了读等待开始位后恢复时钟。

因为SDIO_CK停止了，可以向卡发送任何命令。在读等待期间，SDIO主机可以在SDIO_D1上监测SDIO中断。

19.6.3 SDIO暂停/恢复操作

在向卡发送数据时，SDIO可以暂停写操作。设置SDIO_CMD[11]位，这指示CPSM当前的命令是一个暂停命令。CPSM分析响应，在从卡收到ACK时(暂停被接受)，它确认在收到当前数据块的CRC后进入空闲状态。

结束暂停操作时，硬件不会保存需要剩下的发送数据块数目。

可以通过软件暂停写操作：在收到卡对暂停命令的ACK时，停止DPSM(SDIO_DCTRL[0]=0)，DPSM即可进入空闲状态。

暂停读操作：DPSM在Wait_r状态等待，在停止数据传输进入暂停之前首先发送完成完整的数据包，应用程序继续读出RxFIFO直到FIFO变空，随后DPSM自动地进入空闲状态。

19.6.4 SDIO中断

当设置了SDIO_DCTRL[11]位，SDIO主机在SDIO_D1上监测SDIO中断。

19.7 CE-ATA特定操作

下面是CE-ATA的特定操作：

- 送出命令完成信号能够关闭CE-ATA设备
- 从CE-ATA设备接收命令完成信号
- 使用状态位和/或中断，向CPU发送CE-ATA命令完成信号

仅当设置了SDIO_CMD[14]位时，即SDIO主机只对CE-ATA的CMD61命令支持这些操作。

19.7.1 命令完成指示关闭

如果未设置“允许CMD结束位”SDIO_CMD[12]并且设置了“非中断使能位”SDIO_CMD[13]，命令完成关闭信号是在收到一个短响应之后的8位个周期之后发出。

为命令移位寄存器加载关闭序列“00001”并且在命令计数器写入43，则CPSM进入暂停状态。8个周期后，一个触发将CPSM移至发送状态。当命令计数器达到48时，因为没有响应在等待，CPSM变为空闲状态。

19.7.2 命令完成指示使能

如果设置“允许CMD结束位”SDIO_CMD[12]并且设置了“非中断使能位”SDIO_CMD[13]，CPSM在Waitcpl状态下等待命令完成信号。

当在CMD信号上收到’0’，CPSM进入空闲状态。在7位个周期之内不能发送新命令。然后，在最后5个周期(上述7个周期之外)，在推挽模式下CMD信号变为’1’。

19.7.3 CE-ATA中断

命令完成是由状态位SDIO_STA[23]知会CPU，使用清除位SDIO_ICR[23]可以清除该位。

根据屏蔽位SDIO_MASKx[23]的设置，SDIO_STA[23]状态位可以在每一个中断线上产生中断。

19.7.4 中止CMD61

如果还未发送“命令完成指示关闭”信号，但需要中止CMD61命令，命令状态机必须被关闭。然后它变成空闲，并且可以发送CMD12命令。在此操作期间，没有“命令完成指示关闭”信号传送。

19.8 硬件流控制

使用硬件流控制功能可以避免FIFO下溢(发送模式)和上溢(接收模式)错误。

动作是停止SDIO_CK并冻结SDIO状态机，在FIFO不能进行发送和接收数据时，数据传输比暂停。只有SDIOCLK驱动的状态机被冻结，AHB接口还在工作。即使在流控制起作用时，仍然可以读出或写入FIFO。

必须设置SDIO_CLKCR[14]位为1，可以使能硬件流控制。复位后，硬件流控制功能关闭。

19.9 SDIO寄存器

设备通过可以在AHB上操作的32位控制寄存器与系统通信。

19.9.1 SDIO电源控制寄存器(SDIO_POWER)

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

位31:2	保留，始终读为0。
位1:0	**PWRCTRL**：电源控制位 这些位用于定义卡时钟的当前功能状态： 00：电源关闭，卡的时钟停止。 01：保留。 10：保留的上电状态。 11：上电状态，卡的时钟开启。

注意： *写数据后的7个HCLK时钟周期内，不能写入这个寄存器。*

19.9.2 SDIO时钟控制寄存器(SDIO_CLKCR)

地址偏移：0x04

复位值：0x0000 0000

SDIO_CLKCR寄存器控制SDIO_CK输出时钟。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留																	HWFC_EN	NEGEDGE	WIDBUS		BYPASS	PWRSAV	CLKEN	CLKDIV							
res																	rw	rw	rw		rw	rw	rw	r/w							

位31:15	保留，始终读为0。
位14	**HWFC_EN**：硬件流控制使能 0：关闭硬件流控制 1：使能硬件流控制 当使能硬件流控制后，关于TXFIFOE和RXFIFOF中断信号的意义请参考19.9.11节的SDIO状态寄存器的定义。
位13	**NEGEDGE**：SDIO_CK相位选择位 0：在主时钟SDIOCLK的上升沿产生SDIO_CK。 1：在主时钟SDIOCLK的下降沿产生SDIO_CK。
位12:11	**WIDBUS**：宽总线模式使能位 00：默认总线模式，使用SDIO_D0。 01：4位总线模式，使用SDIO_D[3:0]。 10：8位总线模式，使用SDIO_D[7:0]。

位10	**BYPASS**：旁路时钟分频器 0：关闭旁路：驱动SDIO_CK输出信号之前，依据CLKDIV数值对SDIOCLK分频。 1：使能旁路：SDIOCLK直接驱动SDIO_CK输出信号。
位9	**PWRSAV**：省电配置位 为了省电，当总线为空闲时，设置PWRSAV位可以关闭SDIO_CK时钟输出。 0：始终输出SDIO_CK。 1：仅在有总线活动时才输出SDIO_CK。
位8	**CLKEN**：时钟使能位 0：SDIO_CK关闭。 1：SDIO_CK使能。
位7:1	**CLKDIV**：时钟分频系数 这个域定义了输入时钟(SDIOCLK)与输出时钟(SDIO_CK)间的分频系数： SDIO_CK频率 = SDIOCLK/[CLKDIV + 2]。

注意： 1 当SD/SDIO卡或多媒体卡在识别模式，SDIO_CK的频率必须低于400kHz。

2 当所有卡都被赋予了相应的地址后，时钟频率可以改变到卡总线允许的最大频率。

3 写数据后的7个HCLK时钟周期内不能写入这个寄存器。对于SD I/O卡，在读等待期间可以停止SDIO_CK，此时SDIO_CLKCR寄存器不控制SDIO_CK。

19.9.3 SDIO参数寄存器(SDIO_ARG)

地址偏移：0x08

复位值：0x0000 0000

SDIO_ARG寄存器包含32位命令参数，它将作为命令的一部分发送到卡中。

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

位31:0	**CMDARG**：命令参数 命令参数是发送到卡中的命令的一部分，如果一个命令包含一个参数，必须在写命令到命令寄存器之前加载这个寄存器。

19.9.4 SDIO命令寄存器(SDIO_CMD)

地址偏移：0x0C

复位值：0x0000 0000

SDIO_CMD寄存器包含命令索引和命令类型位。命令索引是作为命令的一部分发送到卡中。命令类型位控制命令通道状态机(CPSM)。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留																	CE_ATACMD	nIEN	ENCMDcompl	SDIOSuspend	CPSMEN	WAITPEND	WAITINT	WAITRESP		CMDINDEX						
res																	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw						

位31:15	保留，始终读为0
位14	**ATACMD**：CE-ATA命令 如果设置该位，CPSM转至CMD61。

位13	**nIEN**：不使能中断 如果未设置该位，则使能CE-ATA设备的中断。
位12	**ENCMDcompl**：使能CMD完成 如果设置该位，则使能命令完成信号。
位11	**SDIOSuspend**：SD I/O暂停命令 如果设置该位，则将要发送的命令是一个暂停命令(只能用于SDIO卡)。
位10	**CPSMEN**：命令通道状态机(CPSM)使能位 如果设置该位，则使能CPSM。
位9	**WAITPEND**：CPSM等待数据传输结束(CmdPend内部信号) 如果设置该位，则CPSM在开始发送一个命令之前等待数据传输结束。
位8	**WAITINT**：CPSM等待中断请求 如果设置该位，则CPSM关闭命令超时控制并等待中断请求。
位7:6	**WAITRESP**：等待响应位 这2位指示CPSM是否需要等待响应，如果需要等待响应，则指示响应类型。 00：无响应，期待CMDSENT标志 01：短响应，期待CMDREND或CCRCFAIL标志 10：无响应，期待CMDSENT标志 11：长响应，期待CMDREND或CCRCFAIL标志
位5:0	**CMDINDEX**：命令索引 命令索引是作为命令的一部分发送到卡中。

注意： 1 写数据后的7个HCLK时钟周期内不能写入这个寄存器。

2 多媒体卡可以发送2种响应：48位长的短响应，或136位长的长响应。SD卡和SD I/O卡只能发送短响应，参数可以根据响应的类型而变化，软件将根据发送的命令区分响应的类型。CE-ATA设备只发送短响应。

19.9.5 SDIO命令响应寄存器(SDIO_RESPCMD)

地址偏移：0x10

复位值：0x0000 0000

SDIO_RESPCMD寄存器包含最后收到的命令响应中的命令索引。如果传输的命令响应不包含命令索引(长响应或OCR响应)，尽管它应该包含111111b(响应中的保留域值)，但RESPCMD域的内容未知。

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

位31:6	保留，始终读为0
位5:0	**RESPCMD**：响应的命令索引 只读位，包含最后收到的命令响应中的命令索引。

19.9.6 SDIO响应 1..4 寄存器(SDIO_RESPx)

地址偏移：0x14 + 4*(x-1)，其中 x = 1..4

复位值：0x0000 0000

SDIO_RESP1/2/3/4寄存器包含卡的状态，即收到响应的部分信息。

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

位31:0	**CARDSTATUSx**：见下表。

根据响应状态，卡的状态长度是32位或127位。

表132 响应类型和SDIO_RESPx寄存器

寄存器	短响应	长响应
SDIO_RESP1	卡状态[31:0]	卡状态[127:96]
SDIO_RESP2	不用	卡状态[95:64]
SDIO_RESP3	不用	卡状态[63:32]
SDIO_RESP4	不用	卡状态[31:1]

总是先收到卡状态的最高位，SDIO_RESP3寄存器的最低位始终为0。

19.9.7 SDIO数据定时器寄存器(SDIO_DTIMER)

地址偏移：0x24

复位值：0x0000 0000

SDIO_DTIMER寄存器包含以卡总线时钟周期为单位的数据超时时间。

一个计数器从SDIO_DTIMER寄存器加载数值，并在数据通道状态机(DPSM)进入Wait_R或繁忙状态时进行递减计数，当DPSM处在这些状态时，如果计数器减为0，则设置超时标志。

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

位31:0	**DATATIME**：数据超时时间 以卡总线时钟周期为单位的数据超时时间。

注意 在写入数据控制寄存器进行数据传输之前，必须先写入数据定时器寄存器和数据长度寄存器。

19.9.8 SDIO数据长度寄存器(SDIO_DLEN)

地址偏移：0x28

复位值：0x0000 0000

SDIO_DLEN寄存器包含需要传输的数据字节长度。当数据传输开始时，这个数值被加载到数据计数器中。

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

位31:25	保留，始终读为0。
位24:0	**DATALENGTH**：数据长度 要传输的数据字节数目。

注意 对于块数据传输，数据长度寄存器中的数值必须是数据块长度(见SDIO_DCTRL)的倍数。在写入数据控制寄存器进行数据传输之前，必须先写入数据定时器寄存器和数据长度寄存器。

19.9.9 SDIO数据控制寄存器(SDIO_DCTRL)

地址偏移：0x2C

复位值：0x0000 0000

SDIO_DCTRL寄存器控制数据通道状态机(DPSM)。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留																				SDIOEN	RWMOD	RWSTOP	RWSTART	DBLOCKSIZE				DMAEN	DTMODE	DTDIR	DTEN
res																				rw	rw	rw	rw	rw				rw	rw	rw	rw

位31:12	保留，始终读为0。
位11	**SDIOEN**：SD I/O使能功能 如果设置了该位，则DPSM执行SD I/O卡特定的操作。
位10	**RWMOD**：读等待模式 0：停止SDIO_CK控制读等待； 1：使用SDIO_D2控制读等待。
位9	**RWSTOP**：读等待停止 0：如果设置了RWSTART，执行读等待； 1：如果设置了RWSTART，停止读等待。
位8	**RWSTART**：读等待开始 设置该位开始读等待操作。
位7:4	**DBLOCKSIZE**：数据块长度 当选择了块数据传输模式，该域定义数据块长度： 0000：(十进制0)块长度 = 2^0 = 1字节； 0001：(十进制1)块长度 = 2^1 = 2字节； 0010：(十进制2)块长度 = 2^2 = 4字节； 0011：(十进制3)块长度 = 2^3 = 8字节； 0100：(十进制4)块长度 = 2^4 = 16字节； 0101：(十进制5)块长度 = 2^5 = 32字节； 0110：(十进制6)块长度 = 2^6 = 64字节； 0111：(十进制7)块长度 = 2^7 = 128字节； 1000：(十进制8)块长度 = 2^8 = 256字节； 1001：(十进制9)块长度 = 2^9 = 512字节； 1010：(十进制10)块长度 = 2^{10} = 1024字节； 1011：(十进制11)块长度 = 2^{11} = 2048字节； 1100：(十进制12)块长度 = 2^{12} = 4096字节； 1101：(十进制13)块长度 = 2^{13} = 8192字节； 1110：(十进制14)块长度 = 2^{14} = 16384字节； 1111：(十进制15)保留。
位3	**DMAEN**：DMA使能位 0：关闭DMA； 1：使能DMA。
位2	**DTMODE**：数据传输模式 0：块数据传输； 1：流数据传输。
位1	**DTDIR**：数据传输方向 0：控制器至卡； 1：卡至控制器。
位0	**DTEN**：数据传输使能位 如果设置该位为1，则开始数据传输。根据DTSIR方向位，DPSM进入Wait_S或Wait_R状态，如果在传输的一开始就设置了RWSTART位，则DPSM进入读等待状态。不需要在数据传输结束后清除使能位，但必须更改SDIO_DCTRL以允许新的数据传输。

注意　写数据后的7个HCLK时钟周期内不能写入这个寄存器。

19.9.10 SDIO数据计数器寄存器(SDIO_DCOUNT)

地址偏移：0x30

复位值：0x0000 0000

当DPSM从空闲状态进入Wait_R或Wait_S状态时，SDIO_DCOUNT寄存器从数据长度寄存器加载数值(见SDIO_DLEN)，在数据传输过程中，该计数器的数值递减直到减为0，然后DPSM进入空闲状态并设置数据状态结束标志DATAEND。

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

位31:25	保留，始终读为0。
位24:0	**DATACOUNT**：数据计数数值 读这个寄存器时返回待传输的数据字节数，写这个寄存器无作用。

注意　只能在数据传输结束时读这个寄存器。

19.9.11 SDIO状态寄存器(SDIO_STA)

地址偏移：0x34

复位值：0x0000 0000

SDIO_STA是一个只读寄存器，它包含两类标志：

- 静态标志(位[23:22、10:0])：写入SDIO中断清除寄存器(见SDIO_ICR)，可以清除这些位。
- 动态标志(位[21:11])：这些位的状态变化根据它们对应的那部分逻辑而变化(例如：FIFO满和空标志变高或变低随FIFO的数据写入变化)。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留								CEATAEND	SDIOIT	RXDAVL	TXDAVL	RXFIFOE	TXFIFOE	RXFIFOF	TXFIFOF	RXFIFOHF	TXFIFOHE	RXACT	TXACT	CMDACT	DBCKEND	STBITERR	DATAEND	CMDSENT	CMDREND	RXOVERR	TXUNDERR	DTMEOUT	CTIMEOUT	DCRCFAIL	CCRCFAIL
res								r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

位31:24	保留，始终读为0。
位23	**CEATAEND**：在CMD61接收到CE-ATA命令完成信号。
位22	**SDIOIT**：收到SDIO中断。
位21	**RXDVAL**：在接收FIFO中的数据可用。
位20	**TXDVAL**：在发送FIFO中的数据可用。
位19	**RXFIFOE**：接收FIFO空。
位18	**TXFIFOE**：发送FIFO空 若使用了硬件流控制，当FIFO包含2个字时，TXFIFOE信号变为有效。
位17	**RXFIFOF**：接收FIFO满 若使用了硬件流控制，当FIFO还差2个字满时，RXFIFOF信号变为有效。
位16	**TXFIFOF**：发送FIFO满。
位15	**RXFIFOHF**：接收FIFO半满：FIFO中至少还有8个字。
位14	**TXFIFOHE**：发送FIFO半空：FIFO中至少还可以写入8个字。
位13	**RXACT**：正在接收数据。
位12	**TXACT**：正在发送数据。
位11	**CMDACT**：正在传输命令。
位10	**DBCKEND**：已发送/接收数据块(CRC检测成功)。

位9	**STBITERR**：在宽总线模式，没有在所有数据信号上检测到起始位。
位8	**DATAEND**：数据结束(数据计数器，SDIO_DCOUNT = 0)
位7	**CMDSENT**：命令已发送(不需要响应)
位6	**CMDREND**：已接收到响应(CRC检测成功)。
位5	**RXOVERR**：接收FIFO上溢错误。
位4	**TXUNDERR**：发送FIFO下溢错误。
位3	**DTIMEOUT**：数据超时。
位2	**CTIMEOUT**：命令响应超时 命令超时时间是一个固定的值，为64个SDIO_CK时钟周期。
位1	**DCRCFAIL**：已发送/接收数据块(CRC检测失败)。
位0	**CCRCFAIL**：已收到命令响应(CRC检测失败)。

19.9.12 SDIO清除中断寄存器(SDIO_ICR)

地址偏移：0x38

复位值：0x0000 0000

SDIO_ICR是一个只写寄存器，在寄存器位写'1'将清除SDIO_STA状态寄存器中的对应位。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留								CEATAENDC	SDIOITC	保留											DBCKENDC	STBITERRC	DATAENDC	CMDSENTC	CMDRENDC	RXOVERRC	TXUNDERRC	DTIMEOUTC	CTIMEOUTC	DCRCFAILC	CCRCFAILC
res								rw	rw	res											rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位31:24	保留，始终读为0。
位23	**CEATAENDC**：CEATAEND标志清除位。
位22	**SDIOITC**：SDIOIT标志清除位。
位21:11	保留，始终读为0。
位10	**DBCKENDC**：DBCKEND标志清除位。
位9	**STBITERRC**：STBITERR标志清除位。
位8	**DATAENDC**：DATAEND标志清除位。
位7	**CMDSENTC**：CMDSENT标志清除位。
位6	**CMDRENDC**：CMDREND标志清除位。
位5	**RXOVERRC**：RXOVERR标志清除位。
位4	**TXUNDERRC**：TXUNDERR标志清除位。
位3	**DTIMEOUTC**：DTIMEOUT标志清除位。
位2	**CTIMEOUT**：CTIMEOUT标志清除位。
位1	**DCRCFAILC**：DCRCFAIL标志清除位。
位0	**CCRCFAILC**：CCRCFAIL标志清除位。

19.9.13 SDIO中断屏蔽寄存器(SDIO_MASK)

地址偏移：0x3C

复位值：0x0000 0000

在对应位置'1'，SDIO_MASK中断屏蔽寄存器决定哪一个状态位产生中断。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留								CEATAENDIE	SDIOITIE	RXDAVLIE	TXDAVLIE	RXFIFOEIE	TXFIFOEIE	RXFIFOFIE	TXFIFOFIE	RXFIFOHFIE	TXFIFOHEIE	RXACTIE	TXACTIE	CMDACTIE	DBCKENDIE	STBITERRIE	DATAENDIE	CMDSENTIE	CMDRENDIE	RXOVERRIE	TXUNDERRIE	DTIMEOUTIE	CTIMEOUTIE	DCRCFAILIE	CCRCFAILIE
res								rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位31:24	保留，始终读为0。
位23	**CEATAENDIE**：允许接收到CE-ATA命令完成信号产生中断 由软件设置/清除该位，允许/关闭在收到CE-ATA命令完成信号产生中断功能。 0：收到CE-ATA命令完成信号时不产生中断 1：收到CE-ATA命令完成信号时产生中断
位22	**SDIOITIE**：允许SDIO模式中断已接收中断 由软件设置/清除该位，允许/关闭SDIO模式中断已接收中断功能。 1：SDIO模式中断已接收不产生中断 0：SDIO模式中断已接收产生中断
位21	**RXDVALIE**：接收FIFO中的数据有效产生中断 由软件设置/清除该位，允许/关闭接收FIFO中的数据有效中断。 0：接收FIFO中的数据有效不产生中断 1：接收FIFO中的数据有效产生中断
位20	**TXDVALIE**：发送FIFO中的数据有效产生中断 由软件设置/清除该位，允许/关闭发送FIFO中的数据有效中断。 0：发送FIFO中的数据有效不产生中断 1：发送FIFO中的数据有效产生中断
位19	**RXFIFOEIE**：接收FIFO空产生中断 由软件设置/清除该位，允许/关闭接收FIFO空中断。 0：接收FIFO空不产生中断 1：接收FIFO空产生中断
位18	**TXFIFOEIE**：发送FIFO空产生中断 由软件设置/清除该位，允许/关闭发送FIFO空中断。 0：发送FIFO空不产生中断 1：发送FIFO空产生中断
位17	**RXFIFOFIE**：接收FIFO满产生中断 由软件设置/清除该位，允许/关闭接收FIFO满中断。 0：接收FIFO满不产生中断 1：接收FIFO满产生中断
位16	**TXFIFOFIE**：发送FIFO满产生中断 由软件设置/清除该位，允许/关闭发送FIFO满中断。 0：发送FIFO满不产生中断 1：发送FIFO满产生中断
位15	**RXFIFOHFIE**：接收FIFO半满产生中断 由软件设置/清除该位，允许/关闭接收FIFO半满中断。 0：接收FIFO半满不产生中断 1：接收FIFO半满产生中断

位14	**TXFIFOHE**：发送FIFO半空产生中断 由软件设置/清除该位，允许/关闭发送FIFO半空中断。 0：发送FIFO半空不产生中断 1：发送FIFO半空产生中断
位13	**RXACTIE**：正在接收数据产生中断 由软件设置/清除该位，允许/关闭正在接收数据中断。 0：正在接收数据不产生中断 1：正在接收数据产生中断
位12	**TXACTIE**：正在发送数据产生中断 由软件设置/清除该位，允许/关闭正在发送数据中断。 0：正在发送数据不产生中断 1：正在发送数据产生中断
位11	**CMDACTIE**：正在传输命令产生中断 由软件设置/清除该位，允许/关闭正在传输命令中断。 0：正在传输命令不产生中断 1：正在传输命令产生中断
位10	**DBCKENDIE**：数据块传输结束产生中断 由软件设置/清除该位，允许/关闭数据块传输结束中断。 0：数据块传输结束不产生中断 1：数据块传输结束产生中断
位9	**STBITERRIE**：起始位错误产生中断 由软件设置/清除该位，允许/关闭起始位错误中断。 0：起始位错误不产生中断 1：起始位错误产生中断
位8	**DATAENDIE**：数据传输结束产生中断 由软件设置/清除该位，允许/关闭数据传输结束中断。 0：数据传输结束不产生中断 1：数据传输结束产生中断
位7	**CMDSENTIE**：命令已发送产生中断 由软件设置/清除该位，允许/关闭命令已发送中断。 0：命令已发送不产生中断 1：命令已发送产生中断
位6	**CMDRENDIE**：接收到响应产生中断 由软件设置/清除该位，允许/关闭接收到响应中断。 0：接收到响应不产生中断 1：接收到响应产生中断
位5	**RXOVERRIE**：接收FIFO上溢错误产生中断 由软件设置/清除该位，允许/关闭接收FIFO上溢错误中断。 0：接收FIFO上溢错误不产生中断 1：接收FIFO上溢错误产生中断
位4	**TXUNDERRIE**：发送FIFO下溢错误产生中断 由软件设置/清除该位，允许/关闭发送FIFO下溢错误中断。 0：发送FIFO下溢错误不产生中断 1：发送FIFO下溢错误产生中断
位3	**DTIMEOUTIE**：数据超时产生中断 由软件设置/清除该位，允许/关闭数据超时中断。 0：数据超时不产生中断 1：数据超时产生中断

位2	**CTIMEOUTIE**：命令超时产生中断 由软件设置/清除该位，允许/关闭命令超时中断。 0：命令超时不产生中断 1：命令超时产生中断
位1	**DCRCFAILIE**：数据块CRC检测失败产生中断 由软件设置/清除该位，允许/关闭数据块CRC检测失败中断。 0：数据块CRC检测失败不产生中断 1：数据块CRC检测失败产生中断
位0	**CCRCFAILIE**：命令CRC检测失败产生中断 由软件设置/清除该位，允许/关闭命令CRC检测失败中断。 0：命令CRC检测失败不产生中断 1：命令CRC检测失败产生中断

19.9.14 SDIO FIFO计数器寄存器(SDIO_FIFOCNT)

地址偏移：0x48

复位值：0x0000 0000

SDIO_FIFOCNT寄存器包含还未写入FIFO或还未从FIFO读出的数据字数目。当在数据控制寄存器(SDIO_DCTRL)中设置了数据传输使能位DTEN，并且DPSM处于空闲状态时，FIFO计数器从数据长度寄存器(见SDIO_DLEN)加载数值。如果数据长度未与字对齐(4的倍数)，则最后剩下的1~3个字节被当成一个字处理。

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

位31:24	保留，始终读为0。
位23:0	**FIFOCOUNT**：将要写入FIFO或将要从FIFO读出数据字的数目。

19.9.15 SDIO数据FIFO寄存器(SDIO_FIFO)

地址偏移：0x80

复位值：0x0000 0000

接收和发送FIFO是32位的宽度读或写一组寄存器，它在连续的32个地址上包含32个寄存器，CPU可以使用FIFO读写多个操作数。

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

位31:0	**FIFODATA**：接收或发送FIFO数据。 FIFO数据占据32个32位的字，地址为： (SDIO基址 + 0x80) 至 (SDIO基址 + 0xFC)

19.9.16 SDIO寄存器映像

下表是SDIO寄存器的总结。

表133 SDIO寄存器映像

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	SDIO_POWER	保留																														PWRCTRL	
0x04	SDIO_CLKCR	保留																	HWFC_EN	NEGEDGE	WIDBUS		BYPASS	PWRSAV	CLKEN	CLKDIV							
0x08	SDIO_ARG	CMDARG																															
0x0C	SDIO_CMD	保留																	CE-ATACMD	nIEN	ENCMDcompl	SDIOsuspend	CPSMEN	WAITPEND	WAITINT	WAITRESP	CMDINDEX						
0x10	SDIO_RESPCMD	保留																														RESPCMD	
0x14	SDIO_RESP1	CARDSTATUS1																															
0x18	SDIO_RESP2	CARDSTATUS2																															
0x1C	SDIO_RESP3	CARDSTATUS3																															
0x20	SDIO_RESP4	CARDSTATUS4																															
0x24	SDIO_DTIMER	DATATIME																															
0x28	SDIO_DLEN	保留																															
		DATALENGTH																															
0x2C	SDIO_DCTRL	保留																				SDIOEN	RWMOD	RWSTOP	RWSTART	DBLOCKSIZE				DMAEN	DTMODE	DTDIR	DTEN
0x30	SDIO_DCOUNT	保留																															
		DATACOUNT																															
0x34	SDIO_STA	保留									CEATAEND	SDIOIT	RXDAVL	TXDAVL	RXFIFOE	TXFIFOE	RXFIFOF	TXFIFOF	RXFIFOHF	TXFIFOHE	RXACT	TXACT	CMDACT	DBCKEND	STBITERR	DATAEND	CMDSENT	CMDREND	RXOVERR	TXUNDERR	DTIMEOUT	CTIMEOUT	DCRCFAIL	CCRCFAIL
0x38	SDIO_ICR	保留									CEATAENDC	SDIOITC	保留											DBCKENDC	STBITERRC	DATAENDC	CMDSENTC	CMDRENDC	RXOVERRC	TXUNDERRC	DTIMEOUTC	CTIMEOUTC	DCRCFAILC	CCRCFAILC
0x3C	SDIO_MASK	保留									CEATAENDIE	SDIOITIE	RXDAVLIE	TXDAVLIE	RXFIFOEIE	TXFIFOEIE	RXFIFOFIE	TXFIFOFIE	RXFIFOHFIE	TXFIFOHEIE	RXACTIE	TXACTIE	CMDACTIE	DBCKENDIE	STBITERRIE	DATAENDIE	CMDSENTIE	CMDRENDIE	RXOVERRIE	TXUNDERRIE	DTIMEOUTIE	CTIMEOUTIE	DCRCFAILIE	CCRCFAILIE
0x48	SDIO_FIFOCNT	保留									FIFOCOUNT																						
		FIFOCOUNT																															
0x80	SDIO_FIFO	FIFODATA																															

20 USB全速设备接口(USB)

小容量产品是指闪存存储器容量在16K至32K字节之间的STM32F101xx、STM32F102xx和STM32F103xx微控制器。

中容量产品是指闪存存储器容量在64K至128K字节之间的STM32F101xx、STM32F102xx和STM32F103xx微控制器。

大容量产品是指闪存存储器容量在256K至512K字节之间的STM32F101xx和STM32F103xx微控制器。

本章节描述的模块适用于增强型STM32F103xx和USB型STM32F102xx系列。

20.1 USB简介

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。

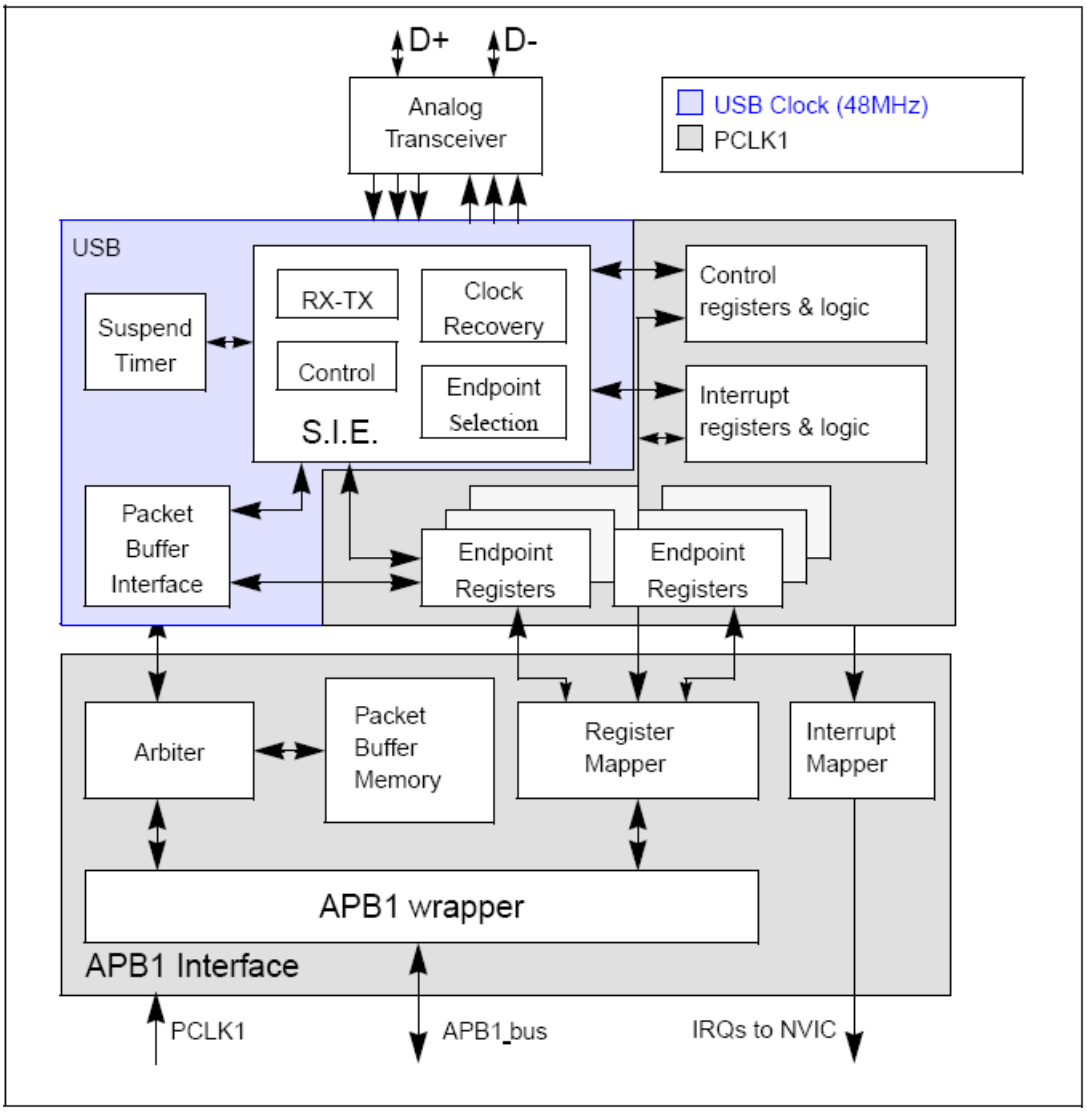
20.2 USB主要特征

- 符合USB2.0全速设备的技术规范
- 可配置1到8个USB端点
- 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。*

下图是USB外设的方框图

图188 USB设备框图



20.3 USB功能描述

USB模块为PC主机和微控制器所实现的功能之间提供了符合USB规范的通信连接。PC主机和微控制器之间的数据传输是通过共享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地址、大小和需要传输的字节数。

当USB模块识别出一个有效的功能/端点的令牌分组时，(如果需要传输数据并且端点已配置)随之发生相关的数据传输。USB模块通过一个内部的16位寄存器实现端口与专用缓冲区的数据交换。在所有的数据传输完成后，如果需要，则根据传输的方向，发送或接收适当的握手分组。

在数据传输结束时，USB模块将触发与端点相关的中断，通过读状态寄存器和/或者利用不同的中断处理程序，微控制器可以确定：

- 哪个端点需要得到服务
- 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型的传输。

USB模块对同步传输和高吞吐量的批量传输提供了特殊的双缓冲区机制，在微控制器使用一个缓冲区的时候，该机制保证了USB外设总是可以使用另一个缓冲区。

在任何不需要使用USB模块的时候，通过写控制寄存器总可以使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。

20.3.1 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：本模块的功能是产生一个与帧开始报文同步的时钟脉冲，并在3ms内没有数据传输的状态，检测出(主机的)全局挂起条件。
- 分组缓冲器接口：此模块管理那些用于发送和接收的临时本地内存单元。它根据SIE的要求分配合适的缓冲区，并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。
- APB1封装：此部件为缓冲区和寄存器提供了到APB1的接口，并将整个USB模块映射到APB1地址空间。
- 中断映射单元：将可能产生中断的USB事件映射到三个不同的NVIC请求线上：
  - USB 低优先级中断(通道 20)：可由所有 USB 事件触发(正确传输，USB 复位等)。固件在处理中断前应当首先确定中断源。
  - USB 高优先级中断(通道 19)：仅能由同步和双缓冲批量传输的正确传输事件触发，目的是保证最大的传输速率。
  - USB 唤醒中断(通道 42)：由 USB 挂起模式的唤醒事件触发。

20.4 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。

20.4.1 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。

20.4.2 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照电压。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)。接下来按照USB复位处理(参见下段)。

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*

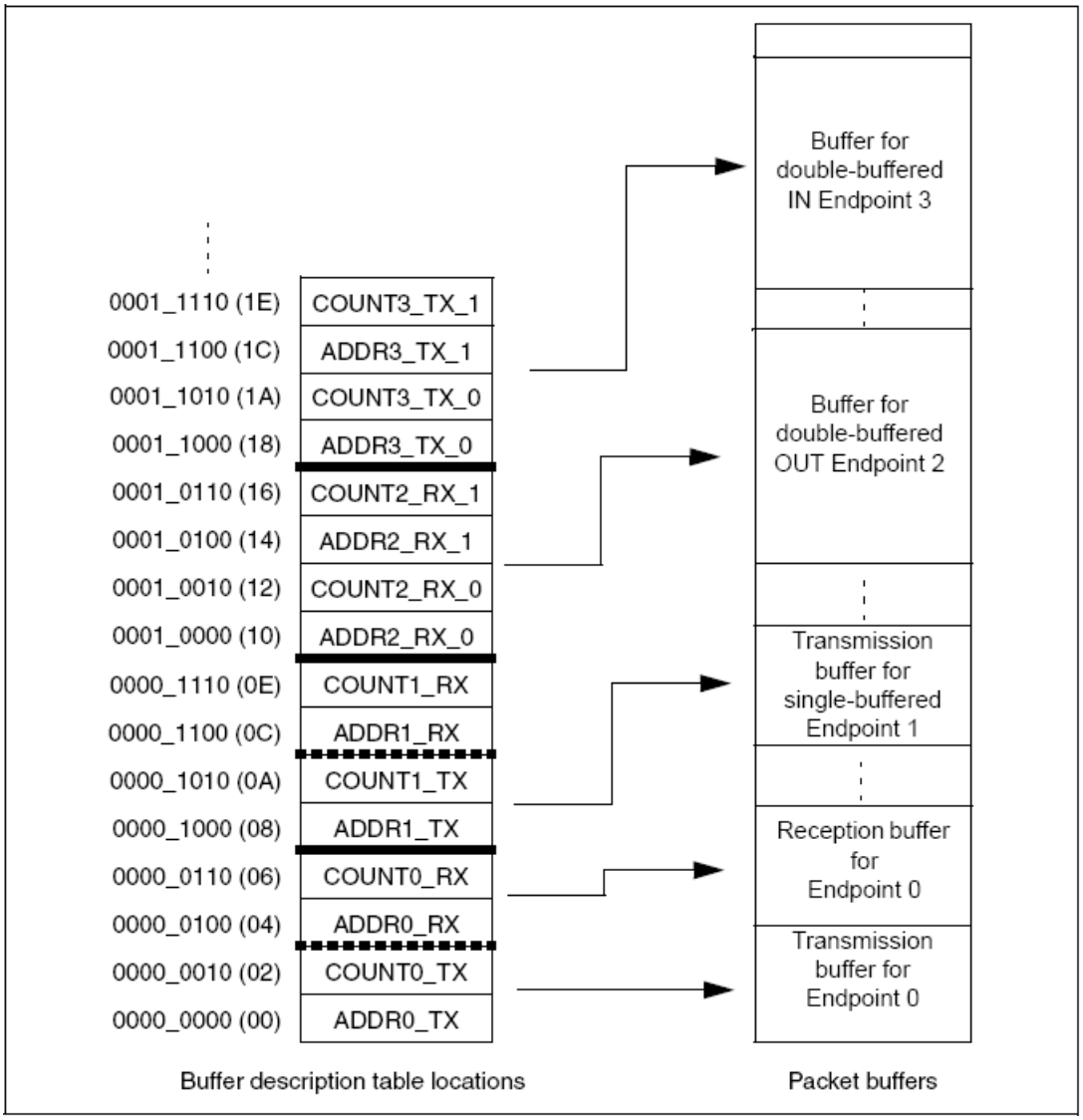
每个端点对应于两个分组缓冲区(一般一个用于发送，另一个用于接收)。这些缓冲区可以位于整个分组存储区的任意位置，因为它们的地址和长度都定义在缓冲区描述表中，而缓冲区描述表也同样位于分组缓冲区中，其地址由寄存器确定。

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20.5.3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20.4.4节：同步传输和20.4.3节：双缓冲端点)。图189描述了缓冲区描述表项和分组缓冲区区域的关系。

图189 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。

20.4.3 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134 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135 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135)，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。

20.4.4 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136 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。

20.4.5 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137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137 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。*

20.5 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0x4000 5C00。由于APB1总线按32位寻址，因此所有的16位寄存器的地址都是按32位字对齐的。同样的地址对齐方式也用于从0x4000 6000开始的分组缓冲存储区。

寄存器描述中使用的一些缩略语请参考1.1节。

20.5.1 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(USB_CNTR)

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	RESETM	SOFM	ESOFM	保留			RESUME	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(USB_ISTR)

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	PMA OVR	ERR	WKUP	SUSP	RESET	SOF	ESOF	保留			DIR	EP_ID[3:0]			
r	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	res			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。 应用程序可以利用该信息访问USB_EPnR位对应的操作，它表示挂起中断传输方向的信息。 该位为只读
位3:0	**EP_ID[3:0]**：端点ID。 此位在USB模块完成数据传输产生中断后由硬件根据请求中断的端点号写入。如果同时有多个端点的请求中断，硬件写入优先级最高的端点号。端点的优先级按以下方法定义：同步端点和双缓冲批量端点具有高优先级，其他的端点为低优先级。如果多个同优先级的端点请求中断，则根据端点号来确定优先级，即端点0具有最高优先级，端点号越小，优先级越高。应用程序可以通过上述的优先级策略顺序处理端点的中断请求。 该位为只读。

USB帧编号寄存器(USB_FNR)

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(USB_DADDR)

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(USB_BTABLE)

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	res		

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

20.5.2 端点寄存器

端点寄存器的数量由USB模块所支持的端点数目决定。USB模块最多支持8个双向端点。每个USB设备必须支持一个控制端点，控制端点的地址(EA位)必需为0。不同的端点必需使用不同的端点号，否则端点的状态不定。每个端点都有与之对应的USB_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]			
rc_w0	t	t	t	r	rw	rw	rw	rc_w0	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20.4.3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20.4.4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138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20.4.3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0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20.4.4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140。 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20.4.3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20.4.3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20.4.4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141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20.4.3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138 接收状态编码

STAT_RX[1:0]	描述
00	**DISABLED**：端点忽略所有的接收请求。
01	**STALL**：端点以STALL分组响应所有的接收请求。
10	**NAK**：端点以NAK分组响应所有的接收请求。
11	**VALID**：端点可用于接收。

表139 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140 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141 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。

20.5.3 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。

以下介绍两种地址表示方式：一种是应用程序访问分组缓冲区时使用的，另一种是相对于USB模块的本地地址。供应用程序使用的分组缓冲区地址需要乘以2才能得到缓冲区在微控制器中的真正地址。分组缓冲区的首地址为0x4000 6000。下面将描述与USB_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20.4.2节。

发送缓冲区地址寄存器 n(USB_ADDRn_TX)

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必须按字对齐，所以此位必须为0。

发送数据字节数寄存器 n(USB_COUNTn_TX)

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(USB_ADDRn_RX)

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(USB_COUNTn_RX)

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142

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142 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

20.5.4 USB寄存器映像

表143 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	0	0

关于寄存器的起始地址，请参见表1。

21 控制器局域网(bxCAN)

小容量产品是指闪存存储器容量在16K至32K字节之间的STM32F101xx、STM32F102xx和STM32F103xx微控制器。

中容量产品是指闪存存储器容量在64K至128K字节之间的STM32F101xx、STM32F102xx和STM32F103xx微控制器。

大容量产品是指闪存存储器容量在256K至512K字节之间的STM32F101xx和STM32F103xx微控制器。

本章节描述的模块仅应用于增强型STM32F103xx系列。

21.1 bxCAN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。

21.2 bxCAN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
- 标识符列表
- FIFO溢出处理方式可配置
- 记录接收SOF时刻的时间戳

时间触发通信模式

- 禁止自动重传模式
- 16位自由运行定时器
- 可在最后2个数据字节发送时间戳

管理

- 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：在中容量和大容量产品中，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。

21.2.1 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。

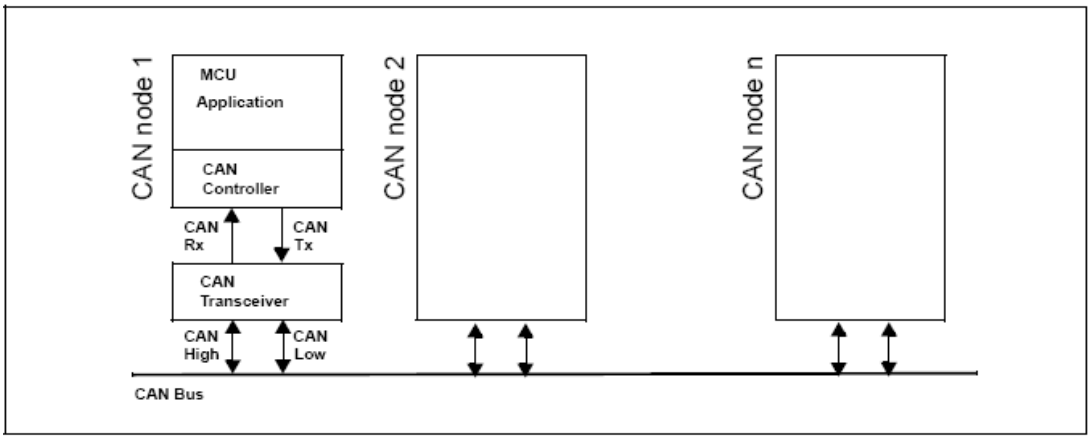
- 需要一个增强的过滤机制来处理各种类型的报文

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190 CAN网拓扑结构



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)。

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

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。

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。

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191 CAN功能框图

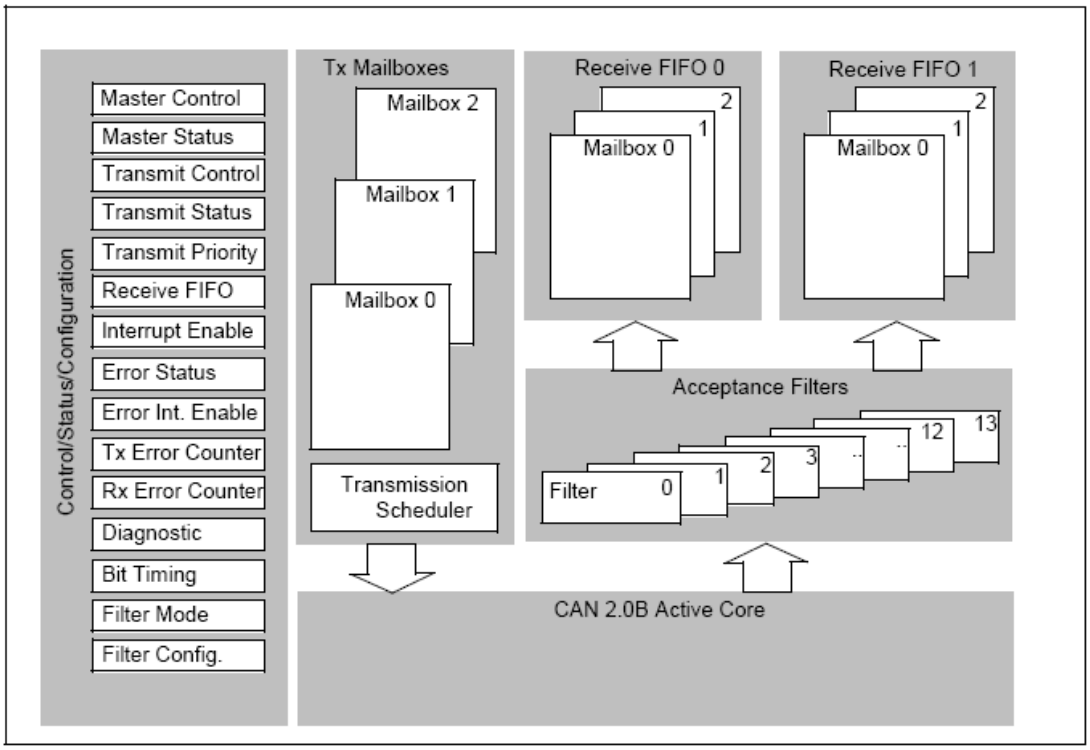
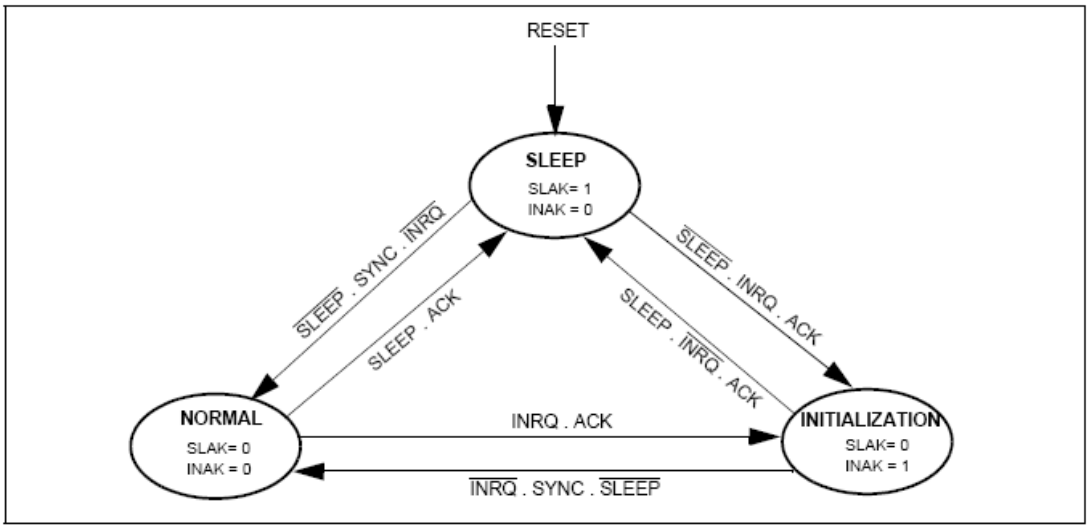


图192 bxCAN工作模式



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

21.3 bxCAN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。

21.3.1 初始化模式

软件初始化应该在硬件处于初始化模式时进行。设置CAN_MCR寄存器的INRQ位为'1'，请求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。

清除CAN_MCR寄存器的INRQ位为'0'，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，禁止报文的接收和发送，并且CANTX引脚输出隐性位(高电平)。

初始化模式的进入，不会改变配置寄存器。

软件对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_FA1R中），然后修改相应过滤器的值。
如果过滤器组没有使用，那么就应该让它处于非激活状态（保持其FACT位为清'0'状态）。

21.3.2 正常模式

在初始化完成后，软件应该让硬件进入正常模式，以便正常接收和发送报文。软件可以通过对CAN_MCR寄存器的INRQ位清'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。

不需要在初始化模式下进行过滤器初值的设置，但必须在它处在非激活状态下完成(相应的FACT位为0)。而过滤器的位宽和模式的设置，则必须在初始化模式中进入正常模式前完成。

21.3.3 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'1'，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，请参考图192：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。

21.3.4 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。

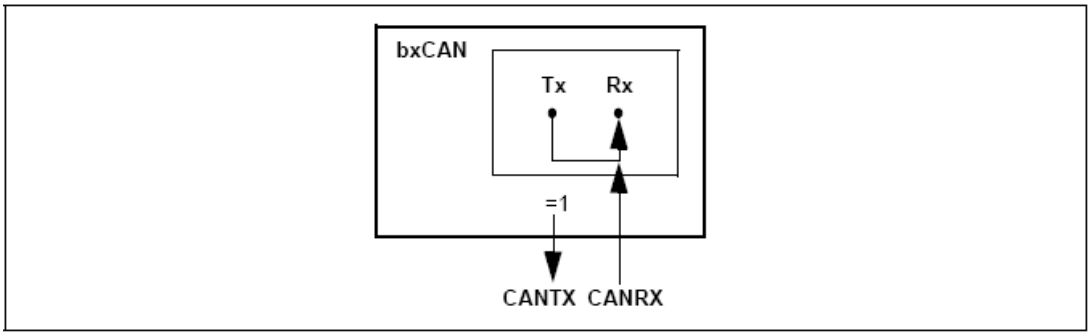
21.3.5 静默模式

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。

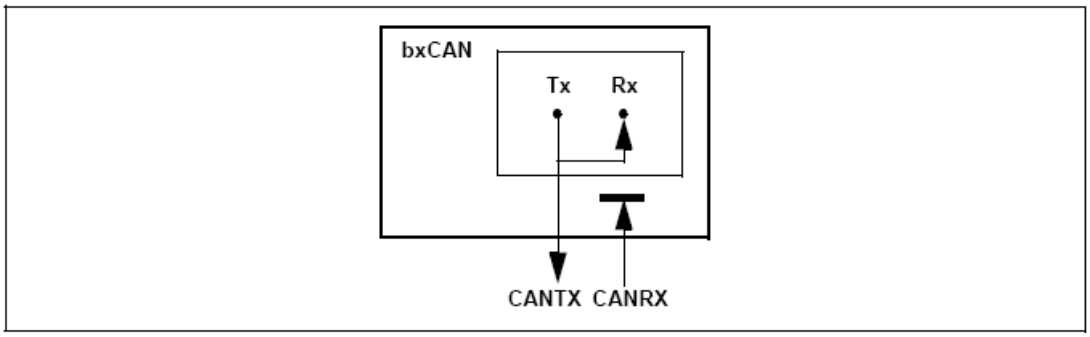
图193 bxCAN工作在静默模式



21.3.6 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194 bxCAN工作在环回模式

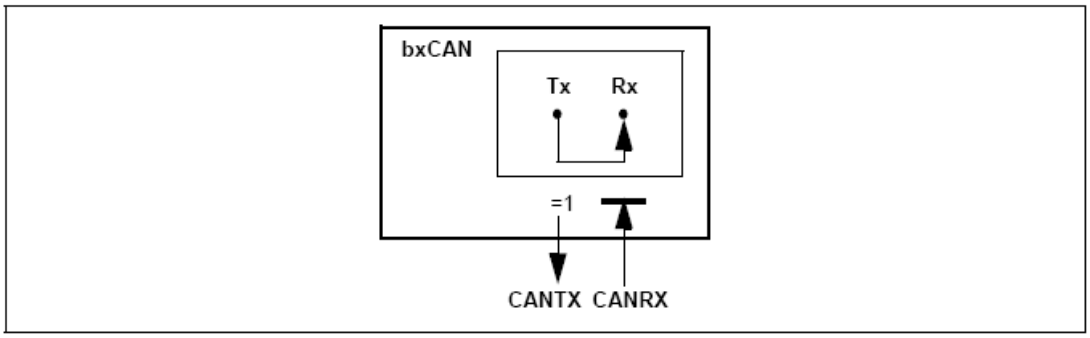


环回模式可用于自测试。为了避免外部的影响，在环回模式下CAN内核忽略确认错误(在数据/远程帧的确认位时刻，不检测是否有显性位)。在环回模式下，bxCAN在内部把Tx输出回馈到Rx输入上，而完全忽略CANRX引脚的实际状态。发送的报文可以在CANTX引脚上检测到。

21.3.7 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195 bxCAN工作在环回静默模式



21.4 bxCAN功能描述

21.4.1 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。

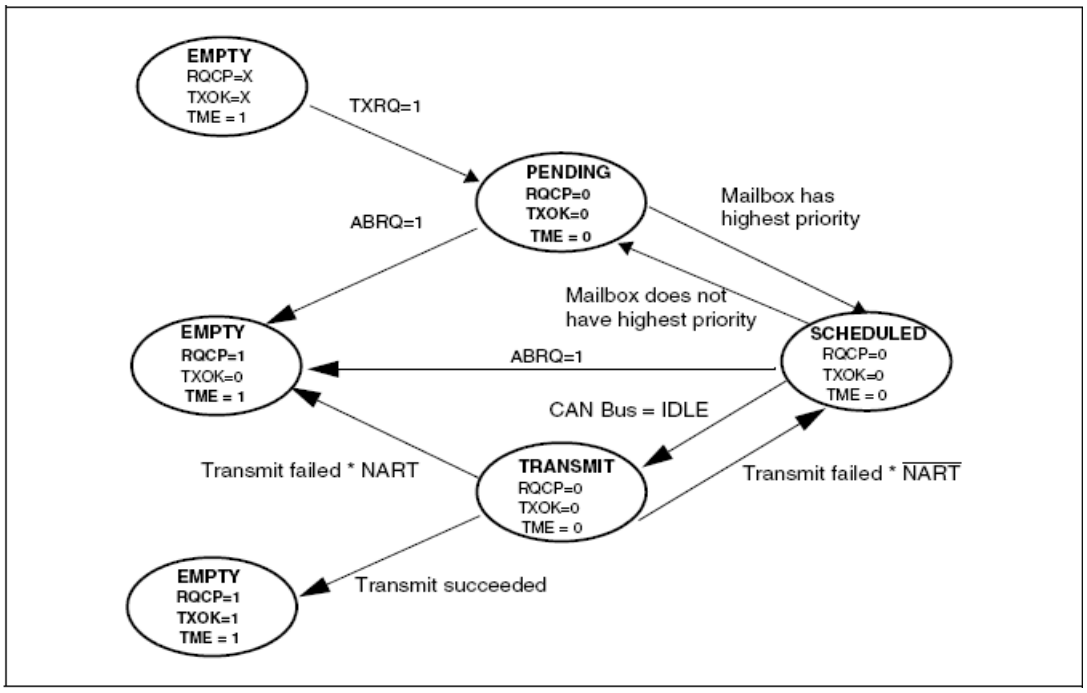
禁止自动重传模式

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196 发送邮箱状态



21.4.2 时间触发通信模式

在该模式下，CAN硬件的内部定时器被激活，并且被用于产生(发送与接收邮箱的)时间戳，分别存储在CAN_RDTxR/CAN_TDTxR寄存器中。内部定时器在每个CAN位时间(见21.4.7节)累加。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。

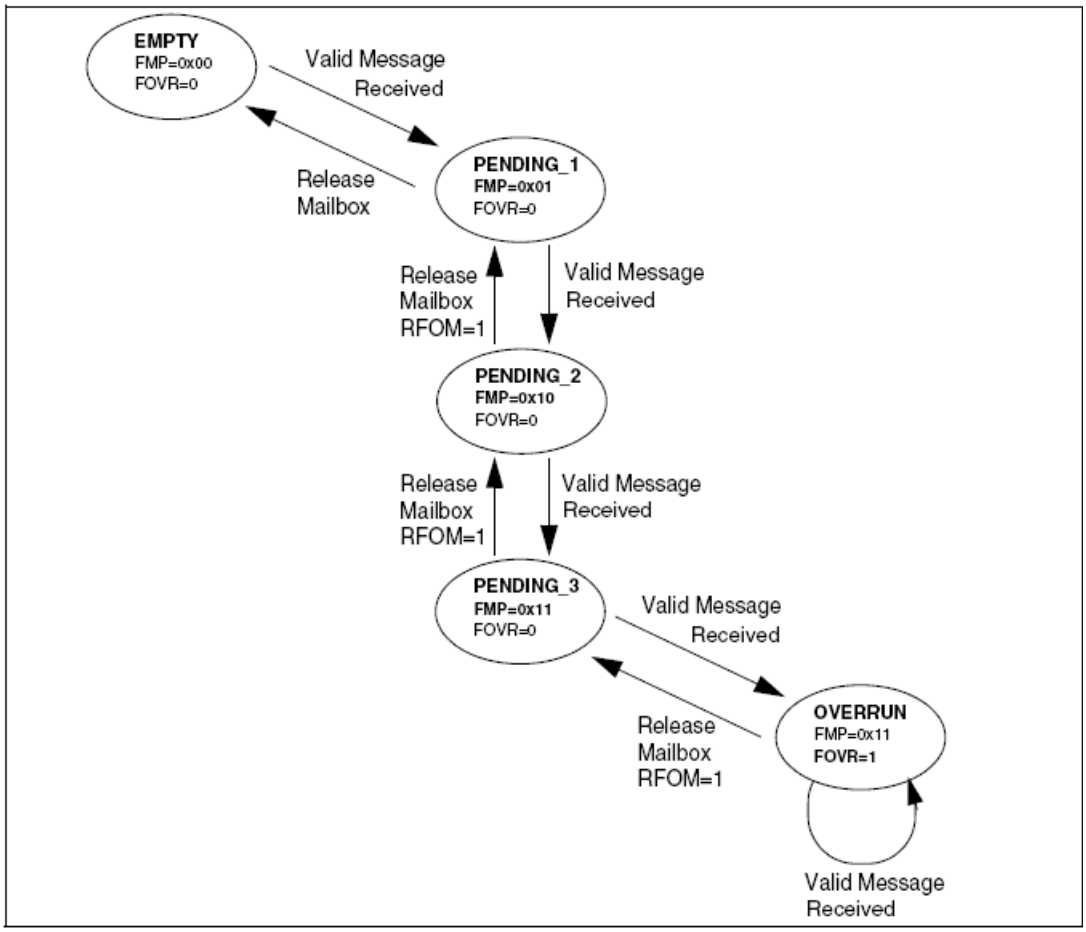
21.4.3 接收管理

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域的最后一位都没有错误)，且通过了标识符过滤，那么该报文被认为是有效报文。请参考21.4.4节：标识符过滤。

图197 接收FIFO状态



FIFO管理

FIFO从**空**状态开始，在接收到第一个有效的报文后，FIFO状态变为**挂号_1**(pending_1)，硬件相应地把CAN_RFR寄存器的FMP[1:0]设置为'01'(二进制01b)。软件可以读取FIFO输出邮箱来读出邮箱中的报文，然后通过对CAN_RFR寄存器的RFOM位设置'1'来释放邮箱，这样FIFO又变为**空**状态了。如果在释放邮箱的同时，又收到了一个有效的报文，那么FIFO仍然保留在**挂号_1**状态，软件可以读取FIFO输出邮箱来读出新收到的报文。

如果应用程序不释放邮箱，在接收到下一个有效的报文后，FIFO状态变为**挂号_2**(pending_2)，硬件相应地把FMP[1:0]设置为'10'(二进制10b)。重复上面的过程，第三个有效的报文把FIFO变为**挂号_3**状态(FMP[1:0]＝11b)。此时，软件必须对RFOM位设置1来释放邮箱，以便FIFO可以有空间来存放下一个有效的报文；否则，下一个有效的报文到来时就会导致一个报文的丢失。

参见21.4.5节：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)，下一个有效的报文就会导致**溢出**，并且一个报文会丢失。此时，硬件对CAN_RFR寄存器的FOVR位进行置'1'来表明溢出情况。至于哪个报文会被丢弃，取决于对FIFO的设置：

- 如果禁用了FIFO锁定功能(CAN_MCR寄存器的RFLM位被清'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存入一个报文，硬件就会更新FMP[1:0]位，并且如果CAN_IER寄存器的FMPIE位为'1'，那么就会产生一个中断请求。

当FIFO变满时(即第3个报文被存入)，CAN_RFR寄存器的FULL位就被置'1'，并且如果CAN_IER寄存器的FFIE位为'1'，那么就会产生一个满中断请求。

在溢出的情况下，FOVR位被置'1'，并且如果CAN_IER寄存器的FOVIE位为'1'，那么就会产生一个溢出中断请求。

21.4.4 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198。

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，不是采用一个标识符加一个屏蔽位的方式，而是使用2个标识符寄存器。接收报文标识符的每一位都必须跟过滤器标识符相同。

过滤器组位宽和模式的设置

过滤器组可以通过相应的CAN_FMR寄存器配置。在配置一个过滤器组前，必须通过清除CAN_FAR寄存器的FACT位，把它设置为禁用状态。通过设置CAN_FS1R的相应FSCx位，可以配置一个过滤器组的位宽，请参见图198。通过CAN_FMR的FBMx位，可以配置对应的屏蔽/标识符寄存器的**标识符列表**模式或**屏蔽位**模式。

为了过滤出一组标识符，应该设置过滤器组工作在屏蔽位模式。

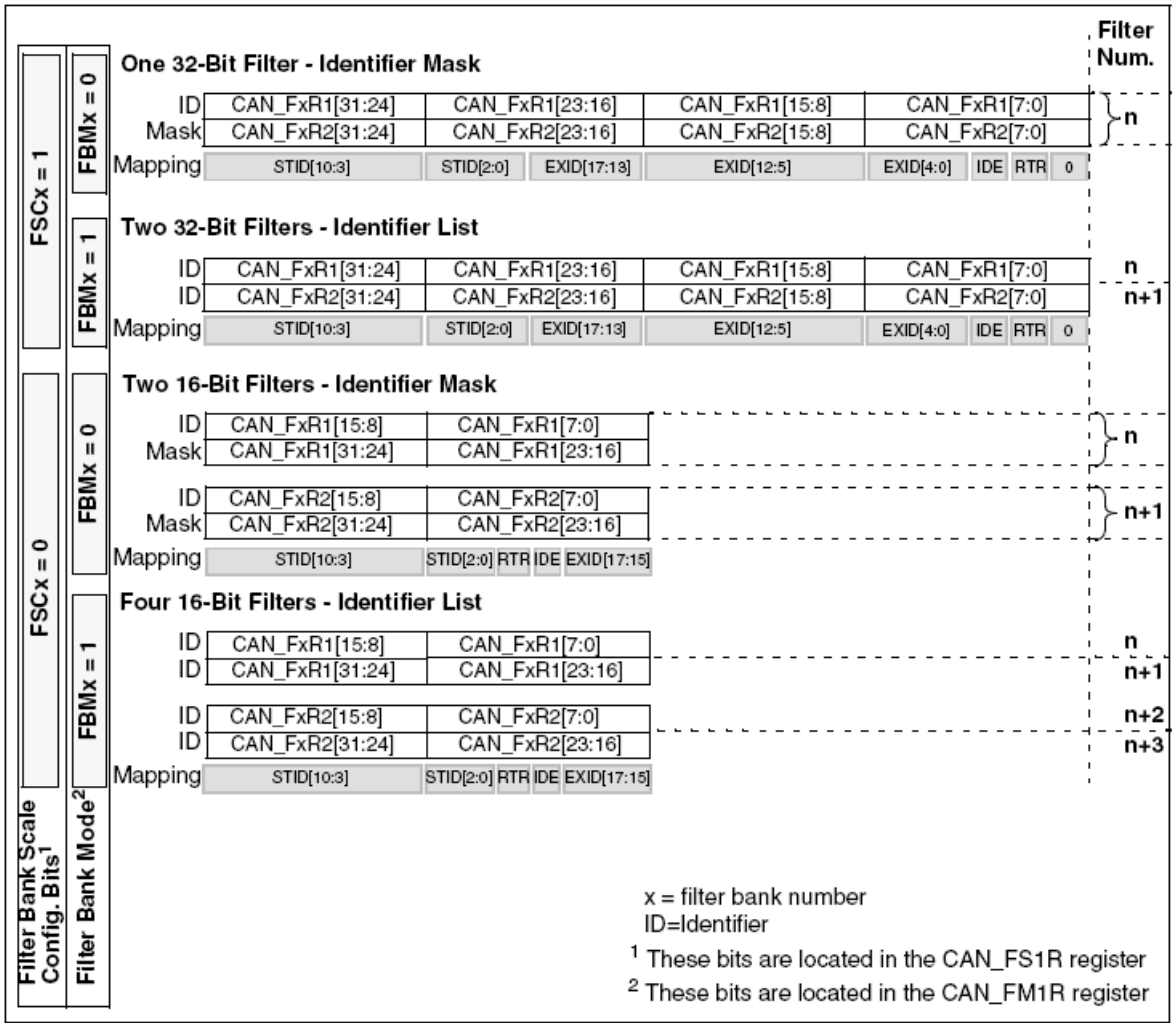
为了过滤出一个标识符，应该设置过滤器组工作在标识符列表模式。

应用程序不用的过滤器组，应该保持在禁用状态。

过滤器组中的每个过滤器，都被编号为(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198。

图198 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：

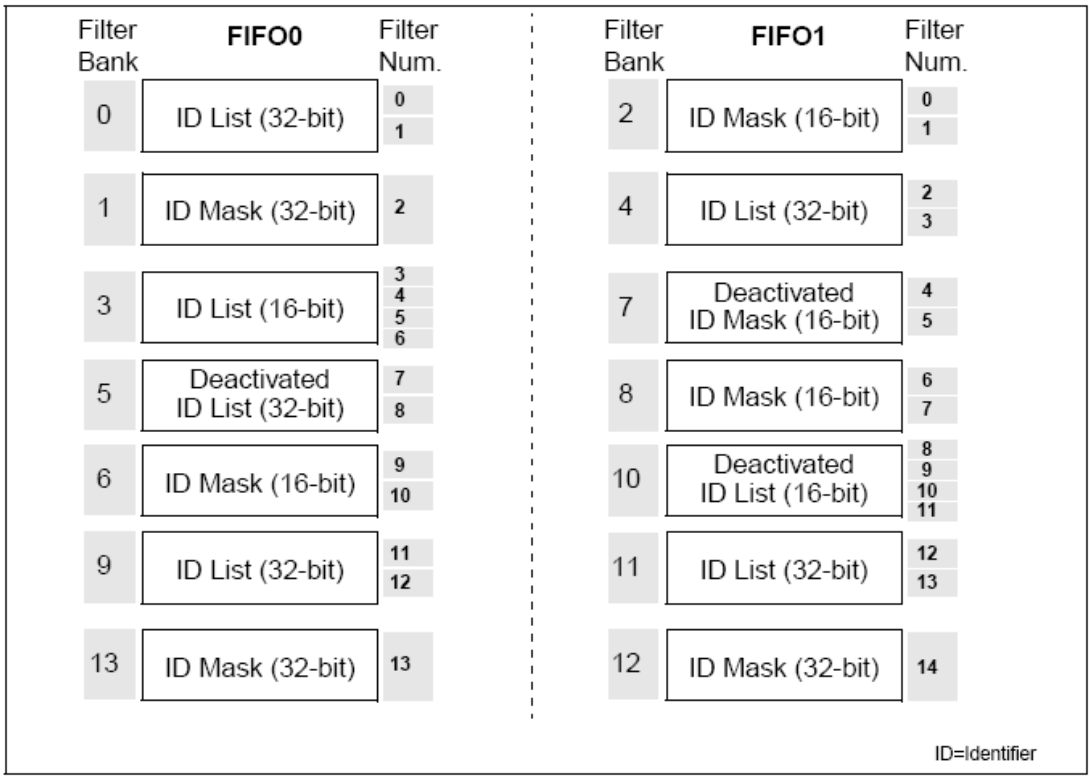
- 把过滤器匹配序号跟一系列所期望的值进行比较
- 把过滤器匹配序号当作一个索引来访问目标地址

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，软件只须对需要的那些屏蔽位(必须匹配的位)进行比较即可。

在给过滤器编号时，并不考虑过滤器组是否为激活状态。另外，每个FIFO各自对其关联的过滤器进行编号。请参考图199的例子。

图199 过滤器编号的例子

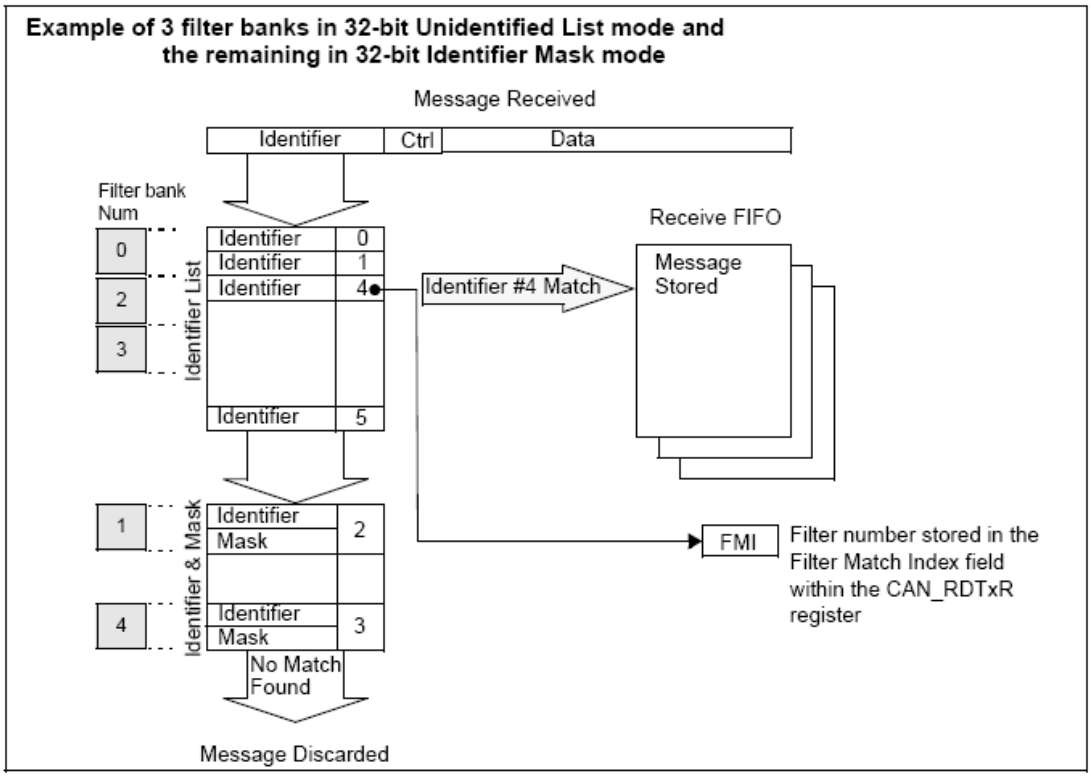


过滤器优先级规则

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200 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。

21.4.5 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144 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

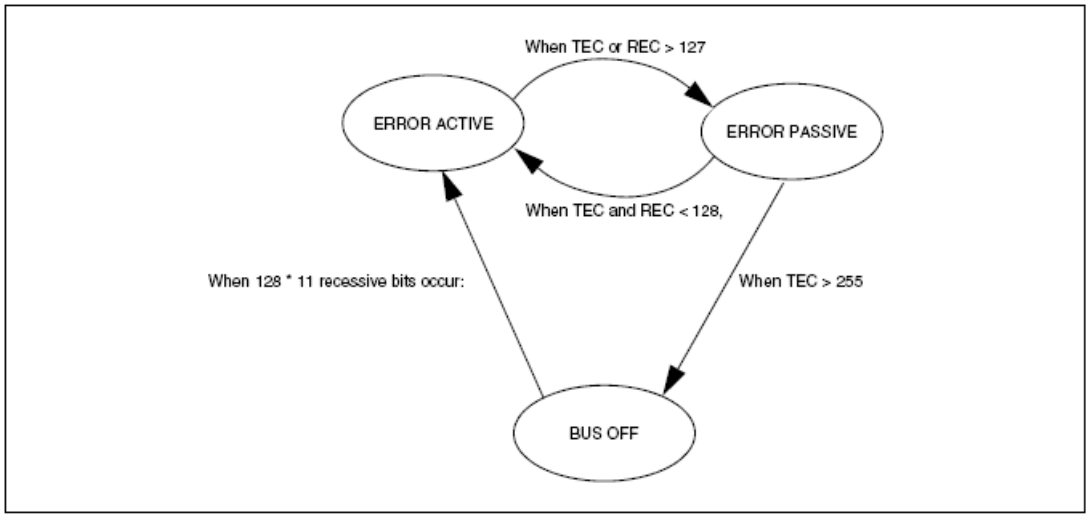
接收邮箱（FIFO）

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145 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201 CAN错误状态图



21.4.6 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**。*

21.4.7 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202 位时序

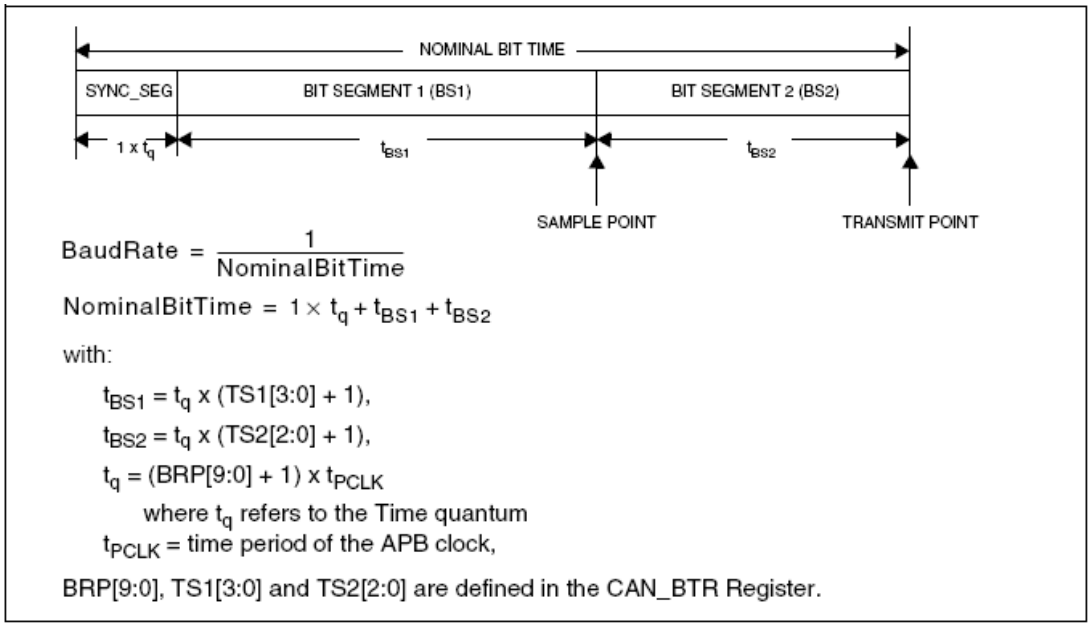
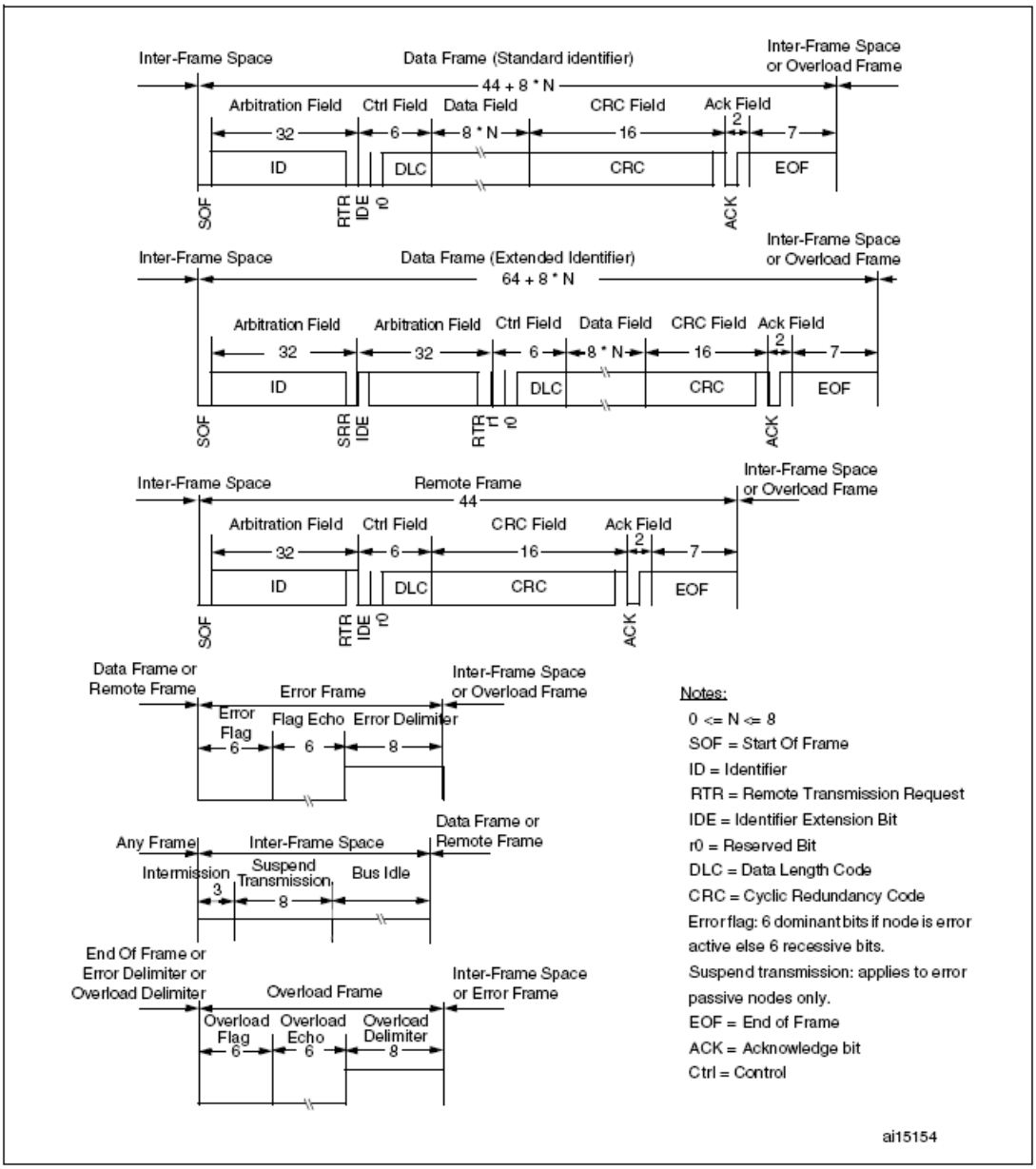


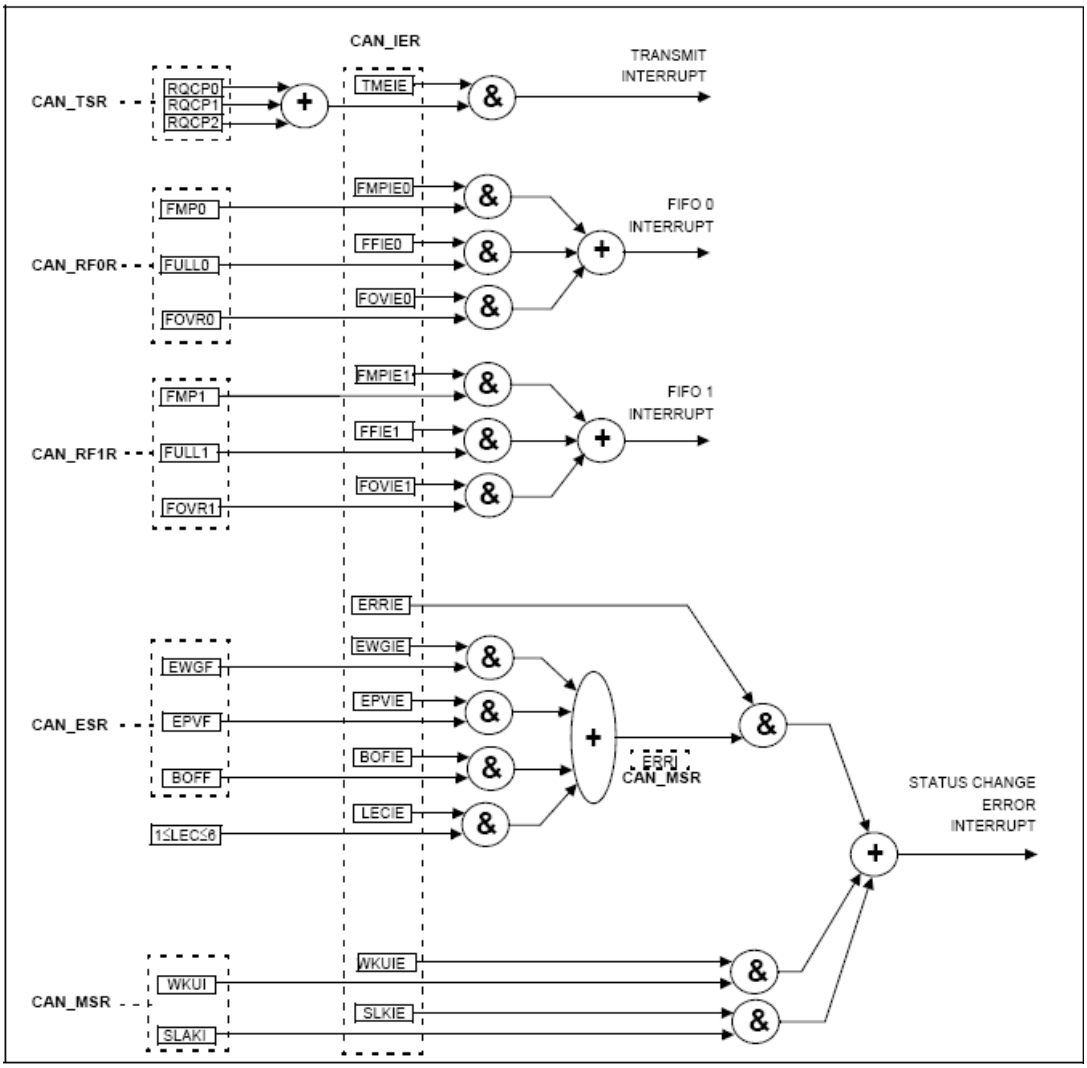
图203 各种CAN帧



21.5 bxCAN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204 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。

21.6 CAN 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。

21.6.1 寄存器访问保护

对某些寄存器的错误访问会导致一个CAN节点对整个CAN网络的暂时性干扰。因此，软件只能在CAN处于初始化模式时修改CAN_BTR寄存器。

虽然错误数据的发送对CAN网的网络层不会带来问题，但却会对应用程序造成严重影响。因此，软件只能在发送邮箱为空的状态改变它，请参见图196。

过滤器的数值只能在关闭对应过滤器组的状态下，或设置FINIT位为'1'后才能修改。此外，只有在设置整个过滤器为初始化模式下(即FINIT=1)，才能修改过滤器的设置，即修改CAN_FMxR，CAN_FSxR和CAN_FFAR寄存器。

21.6.2 控制和状态寄存器

有关寄存器说明中的缩写，参见1.1节。

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	res							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21.4.2：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21.4.6：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。有关清除SLEEP位的详细信息，参见CAN_MCR寄存器的AWUM位的描述。
位0	**INAK**: 初始化确认 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	res			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	res			rc_w1	rc_w1	rc_w1	rc_w1	rs	res			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196。
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196。
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196。

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]	
res										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	res			rw	rw	rw	rw	res	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21.5节bxCAN中断。

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21.4.6。
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	res				rw	rw	res	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						BRP[9:0]									
res						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 $t_{BS2} = t_{CAN} \times (TS2[2:0] + 1)$。

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 $t_{BS1} = t_{CAN} \times (TS1[3:0] + 1)$ 关于位时间特性的详细信息，请参考21.4.7节位时间特性。
位15:10	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(t_q)的时间长度 $t_q = (BRP[9:0]+1) \times t_{PCLK}$

21.6.3 邮箱寄存器

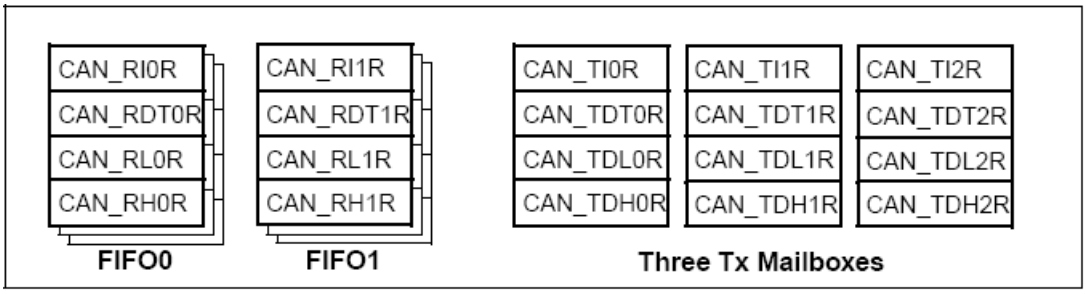
本节描述发送和接收邮箱寄存器。关于寄存器映像的详细信息，请参考21.4.5节报文存储。

除了下述例外，发送和接收邮箱几乎一样：

- CAN_RDTxR 寄存器的FMI域；
- 接收邮箱是只读的；
- 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=未定义位(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* 扩展身份标识的高字节。
位20:3	**EXID[17:0]:** *扩展标识符* 扩展身份标识的低字节。
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：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]			
res							rw	res				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：未定义位

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：未定义位

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：未定义位

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* 扩展身份标识的高字节。
位20:3	**EXID[17:0]:** *扩展标识符* 扩展身份标识的低字节。
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：未定义位

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	res				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21.4.4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：未定义位

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：未定义位

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。

21.6.4 CAN过滤器寄存器

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
保留		CAN2SB[5:0]						保留							FINIT
res		rw						res							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_FM1R)

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198：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_FS1R)

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198：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_FFA1R)

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_FA1R)

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_FiRx) (i＝0..13，x=1..2)

地址偏移量：0x240h..0x2AC

复位值：未定义位

注： 共有14组过滤器：i=0..13。每组过滤器由2个32位的寄存器，CAN_FiR[2:1]组成。

只有在CAN_FaxR寄存器相应的FACTx位清'0'，或CAN_FMR寄存器的FINIT位为'1'时，才能修改相应的过滤器寄存器。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FB31	FB30	FB29	FB28	FB27	FB26	FB25	FB24	FB23	FB22	FB21	FB20	FB19	FB18	FB17	FB16
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FB15	FB14	FB13	FB12	FB11	FB10	FB9	FB8	FB7	FB6	FB5	FB4	FB3	FB2	FB1	FB0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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 寄存器的每位指示是否对应的标识符寄存器位一定要与期望的标识符的相应位一致。 0: 不关心，该位不用于比较； 1: 必须匹配，到来的标识符位必须与滤波器对应的标识符寄存器位相一致。

注： *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21.4.4节标识符过滤。*

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146。

21.6.5 bxCAN寄存器列表

关于寄存器的起始地址，参见表1。

表146 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																RESET	保留							TTCM	ABOM	AWUM	NART	RFLM	TXFP	SLEEP	INRQ
	复位值																	0								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	FOVIE	FFIE1	FMPIE	FOVIE	FFIE0	FMPIE	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	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	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	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	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_FM1R	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_FS1R	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_FFA1R	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_FA1R	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_F0R1	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_F0R2	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_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
244h	CAN_F1R2	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_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
2ACh	CAN_F13R2	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

22 串行外设接口(SPI)

小容量产品是指闪存存储器容量在16K至32K字节之间的STM32F101xx、STM32F102xx和STM32F103xx微控制器。

中容量产品是指闪存存储器容量在64K至128K字节之间的STM32F101xx、STM32F102xx和STM32F103xx微控制器。

大容量产品是指闪存存储器容量在256K至512K字节之间的STM32F101xx和STM32F103xx微控制器。

除非特别说明，本章节描述的模块应用于整个STM32F10xxx微控制器系列。

22.1 SPI简介

在大容量产品上，SPI接口可以配置为支持SPI协议或者支持I^2S音频协议。SPI接口默认工作在SPI协议下，可以通过软件把功能从SPI模式切换到I^2S模式。

在小容量和中容量产品上，不支持I^2S音频协议。

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，并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，包括使用一条双向数据线的双线单工同步传输，还可使用CRC校验的可靠通信。

I^2S也是一种3管脚的同步串行接口通讯协议。它支持四种音频标准，包括飞利浦I^2S标准，MSB和LSB对齐标准，以及PCM标准。它在半双工通讯中，可以工作在主和从2种模式下。当它作为主设备时，通过接口向外部的从设备提供时钟信号。

警告： 由于SPI3/I2S3的部分管脚与JTAG管脚共享(SPI3_NSS/I2S3_WS与JTDI，SPI3_SCK/I2S3_CK与JTDO)，因此这些管脚不受IO控制器控制，他们(在每次复位后)被默认保留为JTAG用途。如果用户想把管脚配置给SPI3/I2S3，必须(在DEBUG时)关闭JTAG并切换至SWD接口，或者(在标准应用时)同时关闭JTAG和SWD接口。详见第7.3.4节。

22.2 SPI和I^2S主要特征

22.2.1 SPI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

22.2.2 I^2S功能

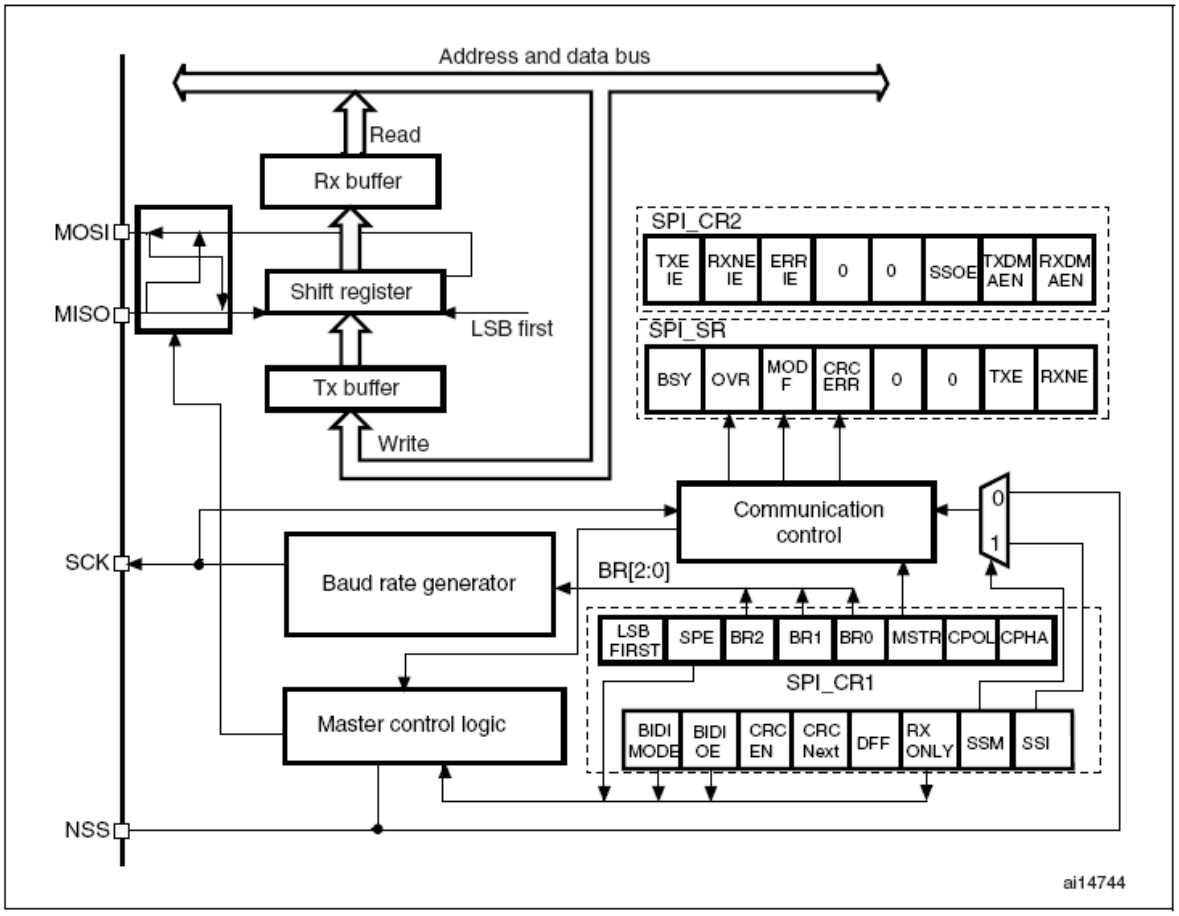
- 单工通信(仅发送或接收)
- 主或者从操作
- 8位线性可编程预分频器，获得精确的音频采样频率(8KHz到48kHz)
- 数据格式可以是16位，24位或者32位
- 音频信道固定数据包帧为16位(16位数据帧)或32位(16、24或32位数据帧)
- 可编程的时钟极性(稳定态)
- 从发送模式下的下溢标志位和主/从接收模式下的溢出标志位
- 16位数据寄存器用来发送和接收，在通道两端各有一个寄存器
- 支持的I^2S协议：
  - I^2S 飞利浦标准
  - MSB 对齐标准(左对齐)
  - LSB 对齐标准(右对齐)
  - PCM 标准(16 位通道帧上带长或短帧同步或者 16 位数据帧扩展为 32 位通道帧)
- 数据方向总是MSB在先
- 发送和接收都具有DMA能力
- 主时钟可以输出到外部音频设备，比率固定为256XFs(Fs为音频采样频率)

22.3 SPI功能描述

22.3.1 概述

SPI的方框图见图205。

图205 SPI框图

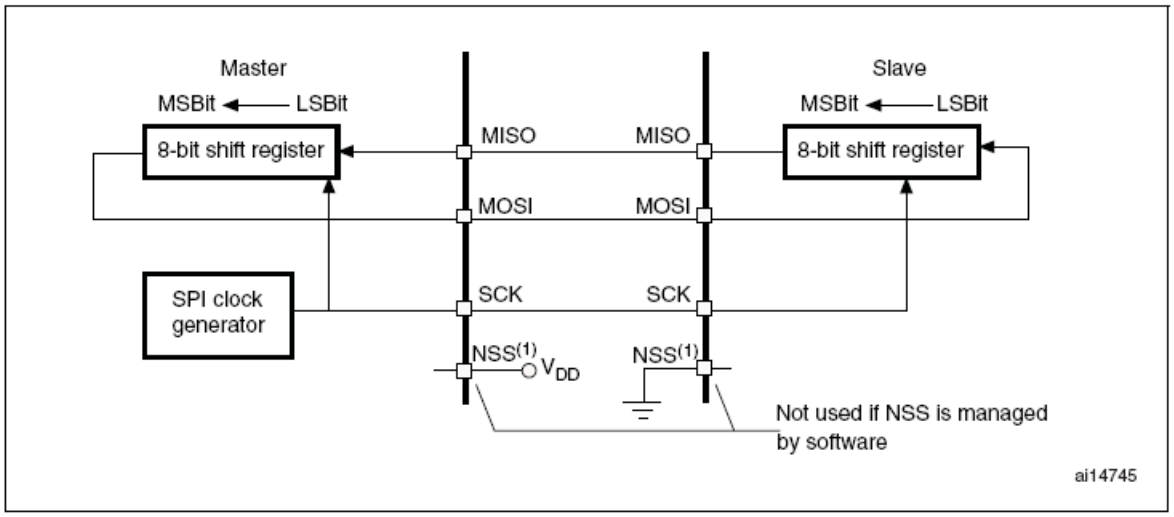


通常SPI通过4个管脚与外部器件相连：

- MISO：主设备输入/从设备输出管脚。该管脚在从模式下发送数据，在主模式下接收数据。
- MOSI：主设备输出/从设备输入管脚。该管脚在主模式下发送数据，在从模式下接收数据。
- SCK：串口时钟，作为主设备的输出，从设备的输入
- NSS：从设备选择。这是一个可选的管脚，用来选择主/从设备。它的功能是用来作为“片选管脚”，让主设备可以单独地与特定从设备通讯，避免数据线上的冲突。从设备的NSS管脚可以由主设备当作一个标准的IO来驱动。一旦被使能(SSOE位)，NSS管脚也可以作为输出管脚，并在SPI设置为主模式时拉低；此时，所有NSS管脚连接到主设备NSS管脚的SPI设备，会检测到低电平，如果它们被设置为NSS硬件模式，就会自动进入从设备状态。

下图是一个单主和单从设备互连的例子。

图206 单主和单从应用



1. 这里NSS管脚设置为输入

MOSI脚相互连接，MISO脚相互连接。这样，数据在主和从之间串行地传输(MSB位在前)。

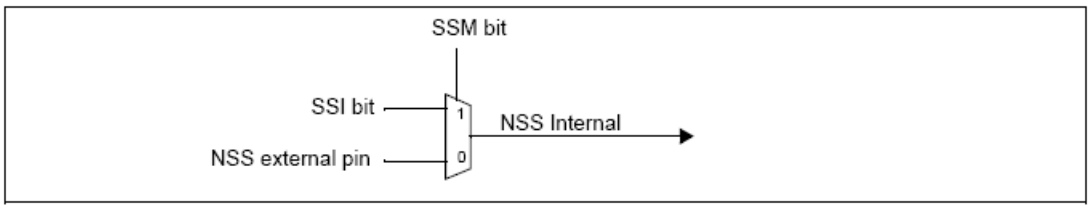
通信总是由主设备发起。主设备通过MOSI脚把数据发送给从设备，从设备通过MISO引脚回传数据。这意味全双工通信的数据输出和数据输入是用同一个时钟信号同步的；时钟信号由主设备通过SCK脚提供。

从选择(NSS)脚管理

有2种NSS模式：

- 软件NSS模式：可以通过设置SPI_CR1寄存器的SSM位来使能这种模式(见图207)。在这种模式下NSS管脚可以用作它用，而内部NSS信号电平可以通过写SPI_CR1的SSI位来驱动
- 硬件NSS模式，分两种情况：
  - NSS 输出被使能：当 STM32F10xx 工作为主 SPI 并且 NSS 输出已经通过 SPI_CR2 寄存器的 SSOE 位使能，这时 NSS 管脚被拉低，所有 NSS 管脚与它的 NSS 管脚相连并配置为硬件 NSS 的 SPI 设备，将自动变成从 SPI 设备。此时该设备不能工作在多主环境。
  - NSS 输出被关闭：允许操作于多主环境。

图207 硬件/软件的从选择管理



时钟信号的相位和极性

SPI_CR寄存器的CPOL和CPHA位，能够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被清'0'，SCK引脚在空闲状态保持低电平；如果CPOL被置'1'，SCK引脚在空闲状态保持高电平。

如果CPHA(时钟相位)位被置'1'，SCK时钟的第二个边沿(CPOL位为0时就是下降沿，CPOL位为1时就是上升沿)进行数据位的采样，数据在第二个时钟边沿被锁存。如果CPHA位被清'0'，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，数据在第一个时钟边沿被锁存。

CPOL时钟极性和CPHA时钟相位的组合选择数据捕捉的时钟边沿。

图208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。

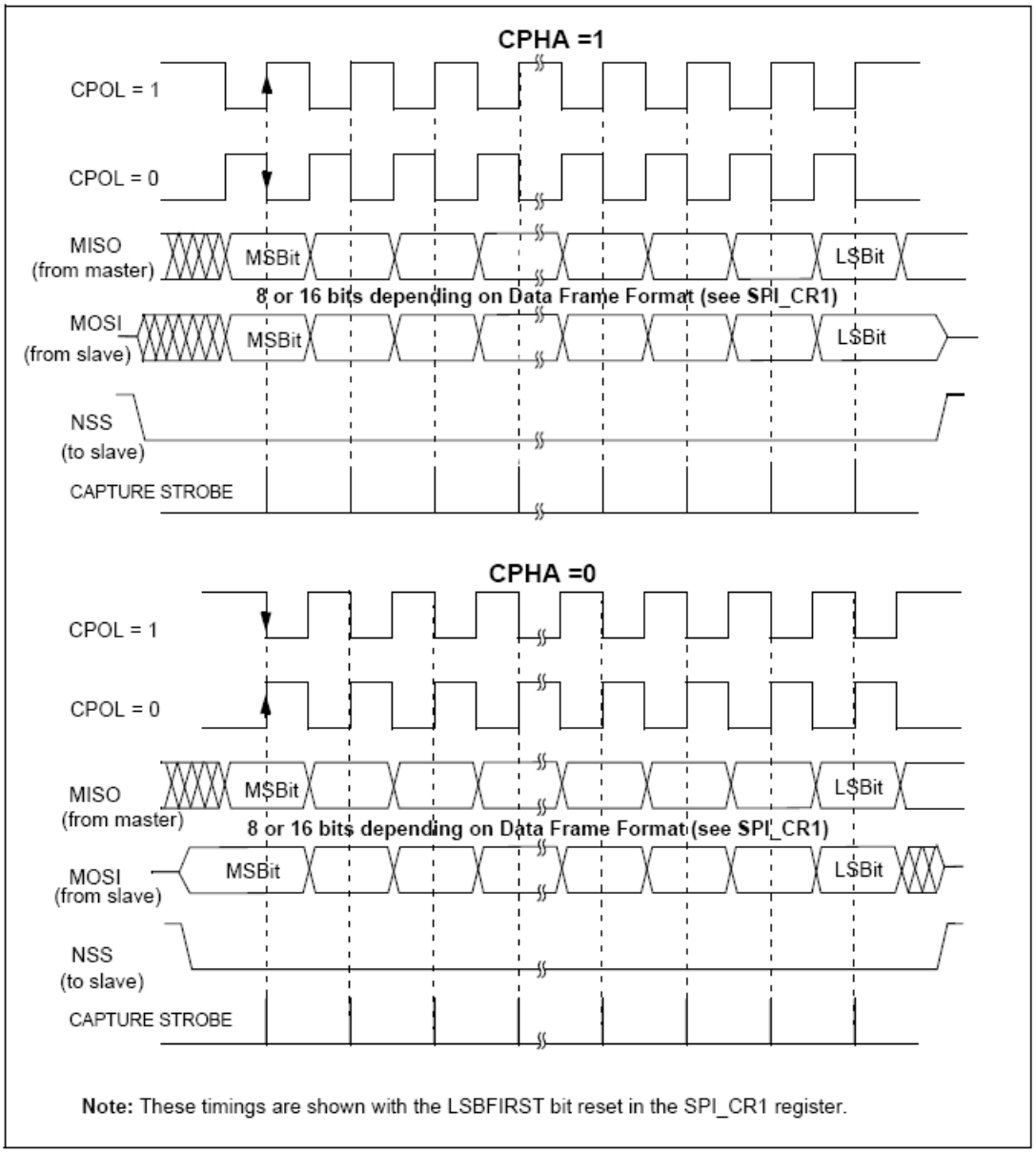
注意： 1. 在改变CPOL/CPHA位之前，必须清除SPE位将SPI禁止。

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208 数据时钟时序图



1. 这些时序体现了SPI_CR1寄存器的LSBFIRST被重置（置0）时的情况

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。

22.3.2 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208)。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

数据字被并行地写入发送缓冲器。

当从设备收到时钟信号，并且在MOSI引脚上出现第一个数据位时，发送过程开始，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。

数据接收过程

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位被置'1'，移位寄存器中接收到的数据字节被传送到接收缓冲器。当读SPI_DR寄存器时，SPI设备返回这个值。

读SPI_DR寄存器时，RXNE位被清除。

22.3.3 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208)。
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。

数据接收过程

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。

22.3.4 单工通信

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(双工模式下只读方式)

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(双工模式下只读方式)

为了释放一根I/O脚作为它用，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。

为启动只接收模式通信，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位为'0'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)可以当作通用IO口使用。因此读数据寄存器时，读不到接收的值。

22.3.5 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。

注：在主接收模式下(单线双向)，不要查询忙标志位(BUSY_FLAG)

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)

此标志为'1'时表明在接收缓冲器中包含有效的接收数据。读SPI数据寄存器可以清除此标志。

22.3.6 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：

- 设置CPOL、CPHA、LSBFirst、BR、SSM、SSI和MSTR的值；
- 在SPI_CRCPR寄存器输入多项式；
- 通过设置SPI_CR1寄存器CRCEN位使能CRC计算，该操作也会清除寄存器SPI_RXCRCR和SPI_TXCRC；
- 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。为了避免在接收最后的数据和CRC时出错，在发送CRC过程中应禁止函数调用。*

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。

当STM32F10xxx配置为从模式并且使用了NSS硬件模式，NSS管脚应该在数据传输和CRC传输期间保持为低。

22.3.7 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。

22.3.8 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。

22.3.9 关闭SPI

当通讯结束，可以通过关闭SPI外设来终止通讯。清除SPE位即可关闭SPI。只要设备不处于主传送模式下，在最后一个字节的传输未完成时关闭SPI并不会影响通讯的可靠性。

注意： *在主传送模式下(全双工或单工发送)，必须在关闭SPI前，通过查询SPI_SR寄存器的BSY标志位，保证没有任何正在进行的通讯。*

22.3.10 SPI中断

表147 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	

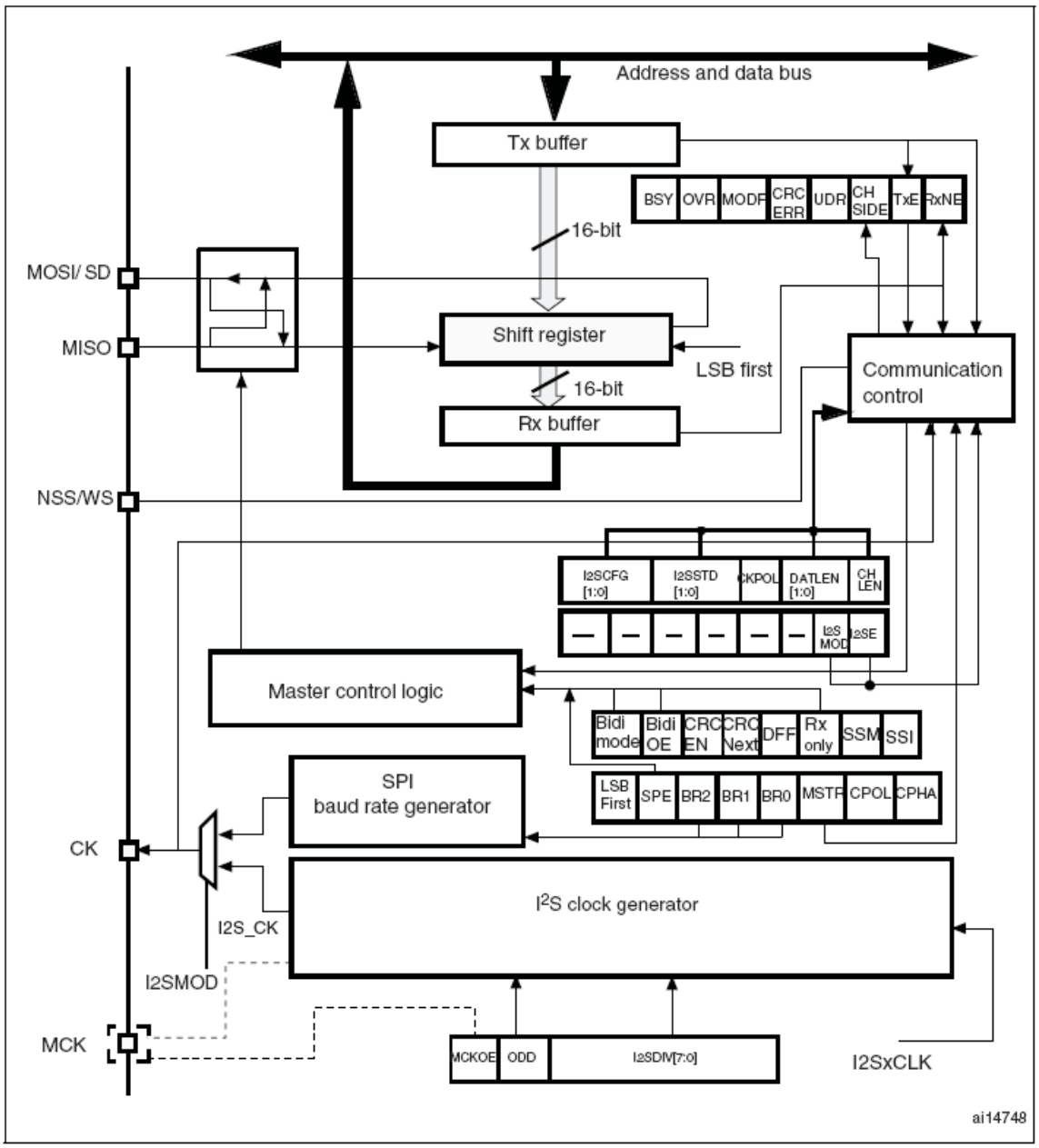
22.4 I²S功能描述

小容量和中容量的STM32不支持I²S音频协议。本节仅适用于大容量产品。

22.4.1 I²S功能描述

I²S的框图如下图所示：

图209 I²S框图



通过将寄存器SPI_I2SCFGR的I2SMOD位置1，即可使能I²S功能，此时，SPI模块可以用作I²S音频接口。I²S接口与SPI接口使用大致相同的管脚、标志和中断。

I²S与SPI共用3个管脚：

- SD：串行数据(映射至MOSI管脚)，用来发送和接收2路时分复用通道道的数据。
- WS：字选(映射至NSS管脚)，主模式下作为数据控制信号输出，从模式下作为输入。
- CK：串行时钟(映射至SCK管脚)，主模式下作为时钟信号输出，从模式下作为输入。

在某些外部音频设备需要主时钟时，可以另有一个附加管脚输出时钟：

- MCK：主时钟(独立映射)，在I²S配置为主模式，寄存器SPI_I2SPR的MCKOE位置1时，作为输出额外的时钟信号管脚使用。输出时钟信号的频率预先设置为256 × Fs，其中Fs是音频信号的采样频率。

设置成主模式时，I²S使用自身的时钟发生器来产生通信用的时钟信号。这个时钟发生器也是主时钟输出的时钟源。I²S模式下有2个额外的寄存器，一个是与时钟发生器配置相关的寄存器SPI_I2SPR，另一个是I²S通用配置寄存器SPI_I2SCFGR(可设置音频标准、从/主模式、数据格式、数据包帧、时钟极性等参数)。

在I²S模式下不使用寄存器SPI_CR1和所有的CRC寄存器。同样，I²S模式下也不使用寄存器SPI_CR2的SSOE位，和寄存器SPI_SR的MODF位和CRCERR位。

I²S使用与SPI相同的寄存器SPI_DR用作16位宽模式数据传输。

22.4.2 支持的音频协议

三线总线支持2个声道上音频数据的时分复用：左声道和右声道，但是只有一个16位寄存器用作发送或接收。因此，软件必须在对数据寄存器写入数据时，根据当前传输中的声道写入相应的数据；同样，在读取寄存器数据时，通过检查寄存器SPI_SR的CHSIDE位来判明接收到的数据属于哪个声道。左声道总是先于右声道发送数据(CHSIDE位在PCM协议下无意义)。

有四种可用的数据和包帧组合。可以通过以下四种数据格式发送数据：

- 16位数据打包进16位帧
- 16位数据打包进32位帧
- 24位数据打包进32位帧
- 32位数据打包进32位帧

在使用16位数据扩展到32位帧时，前16位(MSB)是有意义的数据，后16位(LSB)被强制为0，该操作不需要软件干预，也不需要有DMA请求(仅需要一次读/写操作)。

24位和32位数据帧需要CPU对寄存器SPI_DR进行2次读或写操作，在使用DMA时，需要2次DMA传输。对于24位数据，扩展到32位后，最低8位由硬件置0。

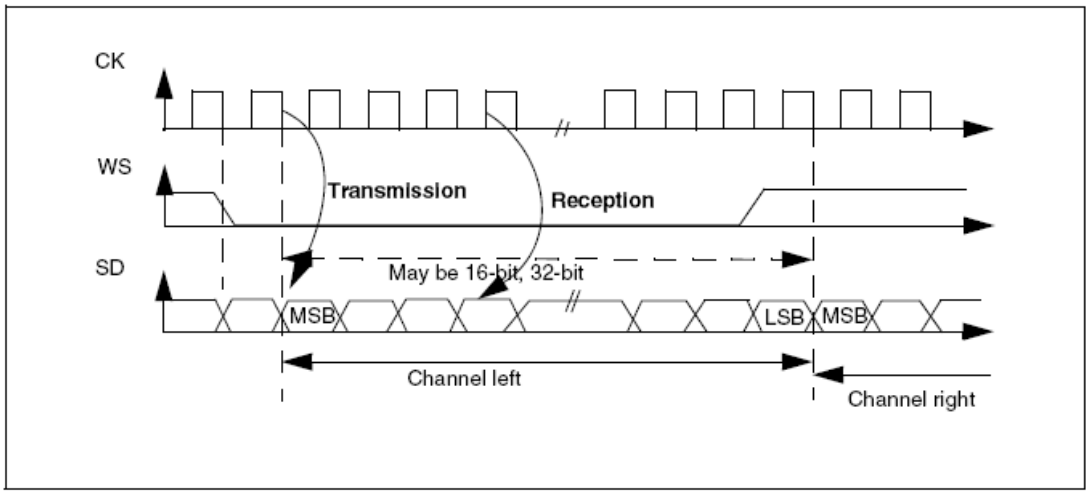
对于所有的数据格式和通讯标准，总是先发送最高位(MSB)。

I²S接口支持四种音频标准，可以通过设置寄存器SPI_I2SCFGR的I2SSTD[1:0]位和PCMSYNC位来选择。

I²S飞利浦标准

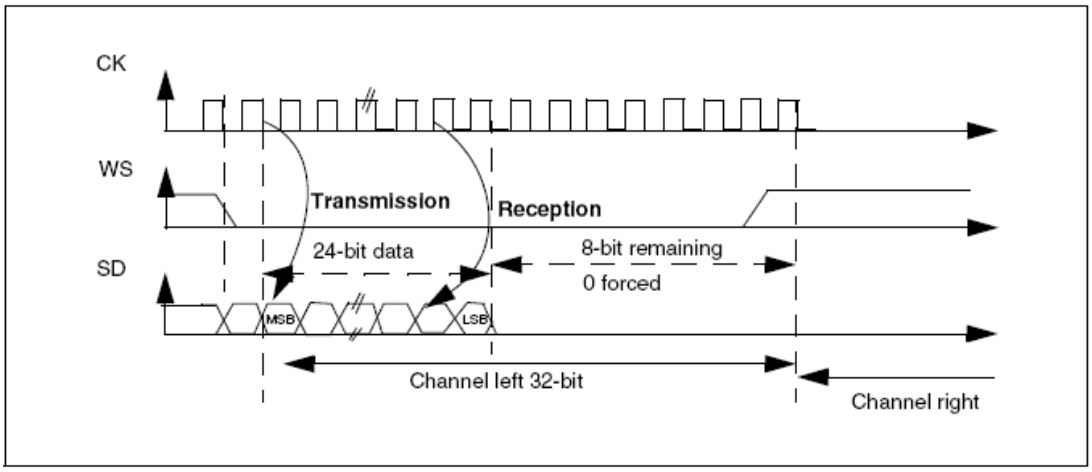
在此标准下，管脚WS用来指示正在发送的数据属于哪个声道。在发送第一位数据(MSB)前1个时钟周期，该管脚即为有效。

图210 I²S飞利浦协议波形(16/32位全精度，CPOL = 0)



发送方在时钟信号的下降沿改变数据，接收方时在上升沿读取数据。WS信号也在时钟信号的下降沿变化。

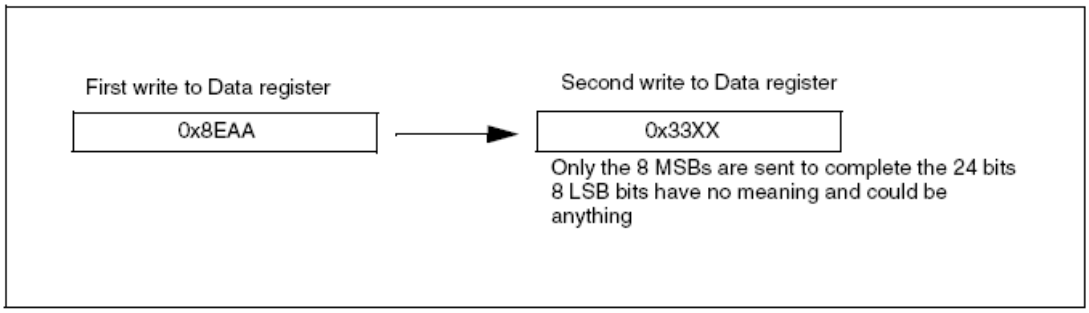
图211 I^2S飞利浦协议标准波形(24位帧，CPOL = 0)



此模式需要对寄存器SPI_DR进行2次读或写操作。

- 在发送模式下：如果需要发送0x8EAA33(24位)

图212 发送0x8EAA33



- 在接收模式下：如果接收0x8EAA33

图213 接收0x8EAA33

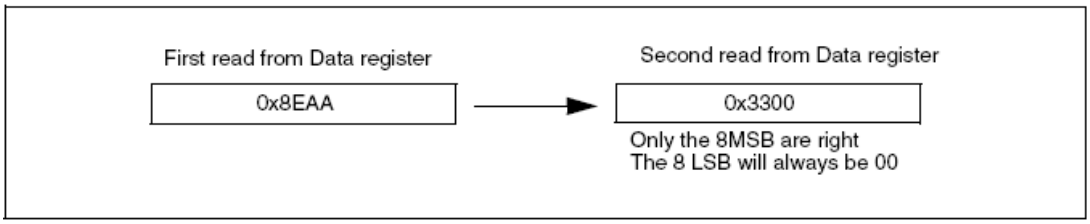
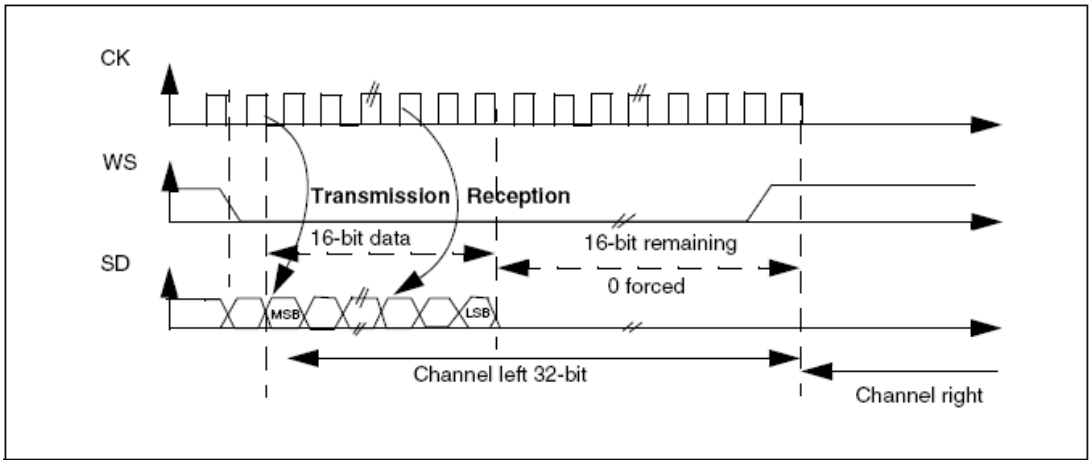


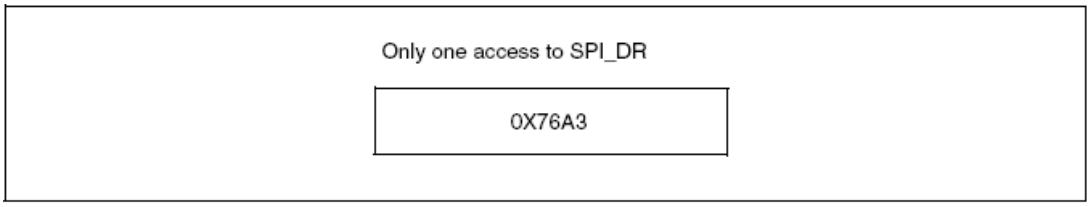
图214 I^2S飞利浦协议标准波形(16位扩展至32位包帧，CPOL = 0)



在I^2S配置阶段，如果选择将16位数据扩展到32声道帧，只需要访问一次寄存器SPI_DR。用来扩展到32位的低16位被硬件置为0x0000。

如果待传输或者接收的数据是0x76A3(扩展到32位是0x76A30000)，需要的操作如下图所示。

图215 示例



在发送时需要将MSB写入寄存器SPI_DR；标志位TXE为'1'表示可以写入新的数据，如果允许了相应的中断，则可以产生中断。发送是由硬件完成的，即使还未发送出后16位的0x0000，也会设置TXE并产生相应的中断。

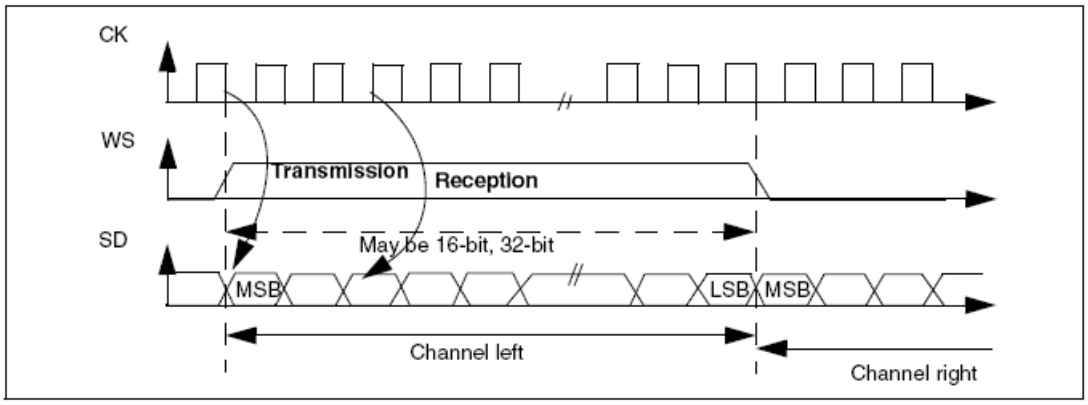
接收时，每次收到高16位半字(MSB)后，标志位RXNE置'1'，如果允许了相应的中断，则可以产生中断。

这样，在2次读和写之间有更多的时间，可以防止下溢或者上溢的情况发生。

MSB对齐标准

在此标准下，WS信号和第一个数据位，即最高位(MSB)同时产生。

图216 MSB对齐16位或32位全精度，CPOL = 0



发送方在时钟信号的下降沿改变数据；接收方时在上升沿读取数据。

图217 MSB对齐24位数据，CPOL = 0

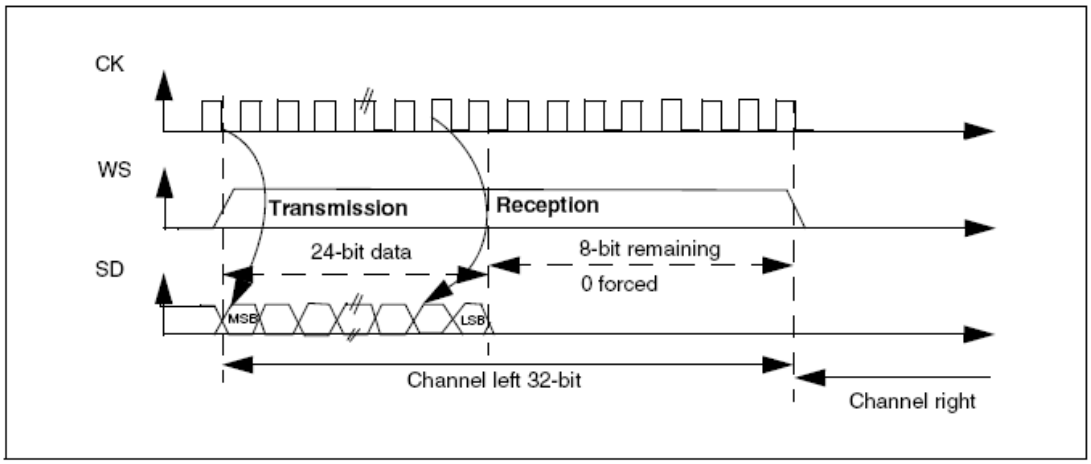
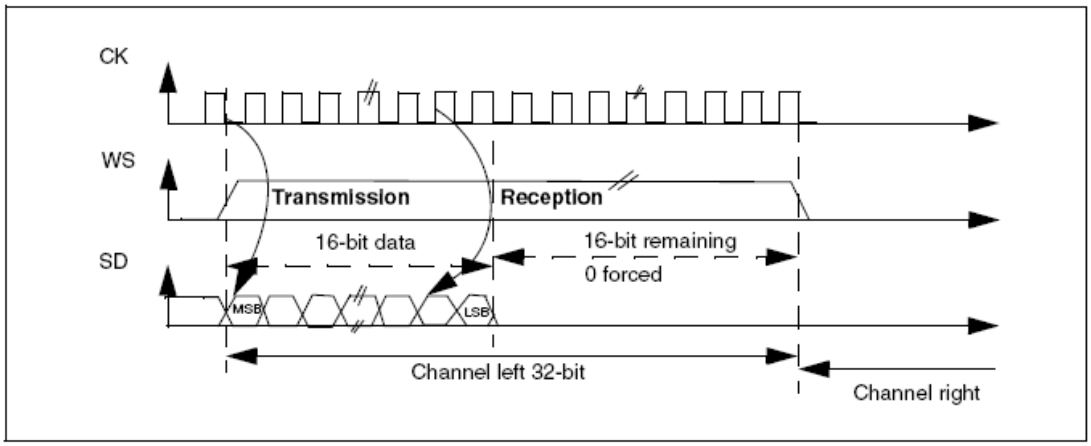


图218 MSB对齐16位数据扩展到32位包帧，CPOL = 0



LSB对齐标准

此标准与MSB对齐标准类似(在16位或32位全精度帧格式下无区别)。

图219 LSB对齐16位或32位全精度，CPOL = 0

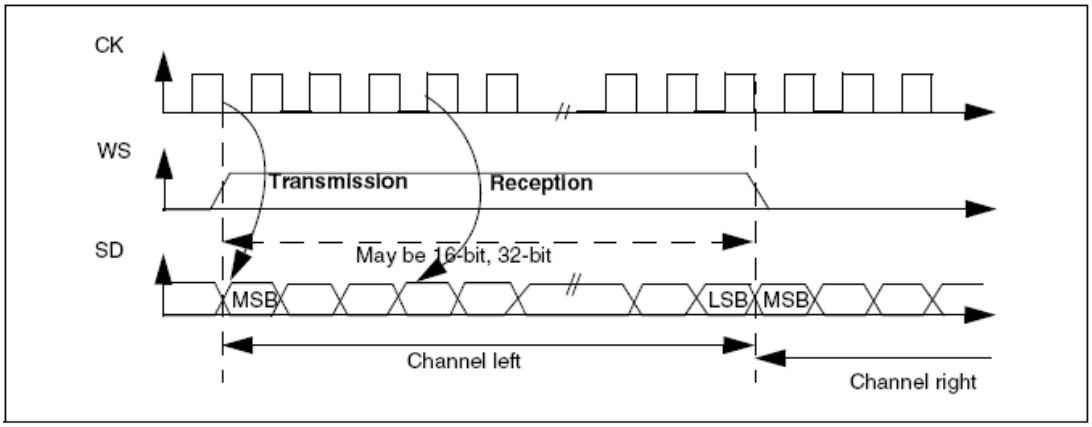
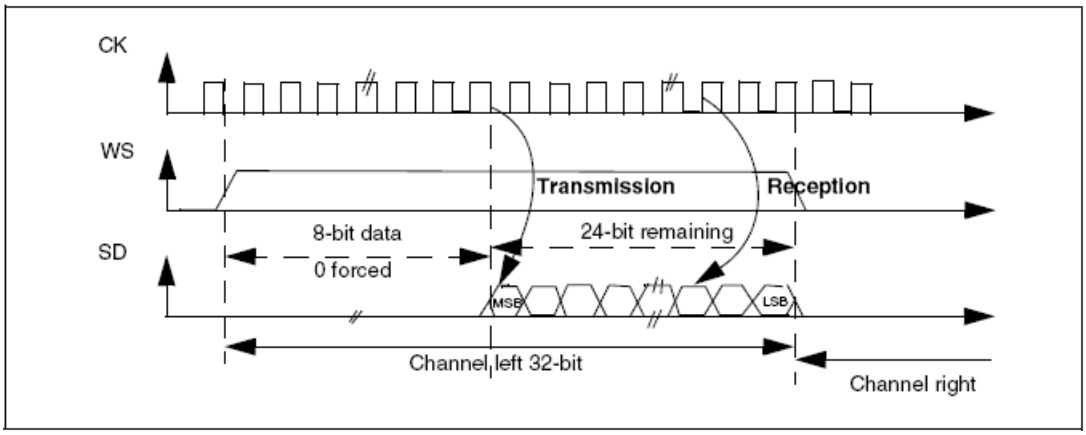


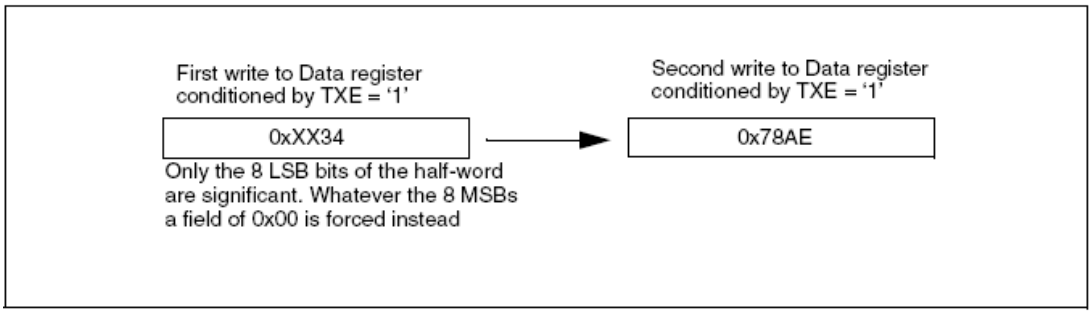
图220 LSB对齐24位数据，CPOL = 0



- 在发送模式下

如果要发送数据0x3478AE，需要通过软件或者DMA对寄存器SPI_DR进行2次写操作。操作流程如下图所示。

图221 要求发送0x3478AE的操作



- 在接收模式下

如果要接收数据0x3478AE，需要在2个连续的RXNE事件发生时，各对寄存器SPI_DR进行1次，共2次读操作。

图222 要求接收0x3478AE的操作

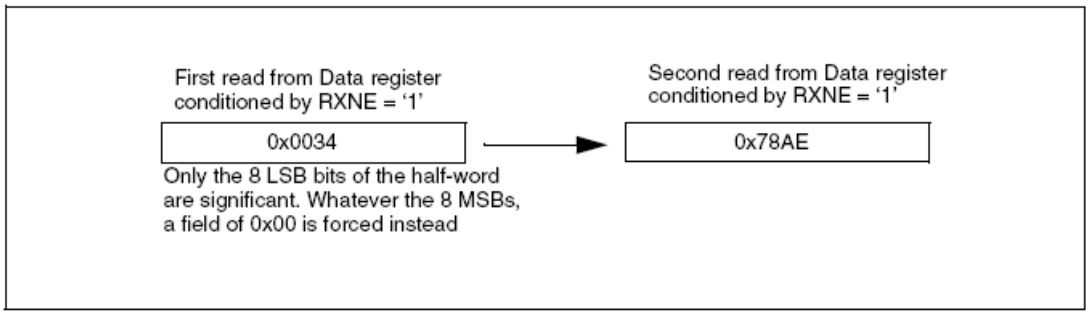
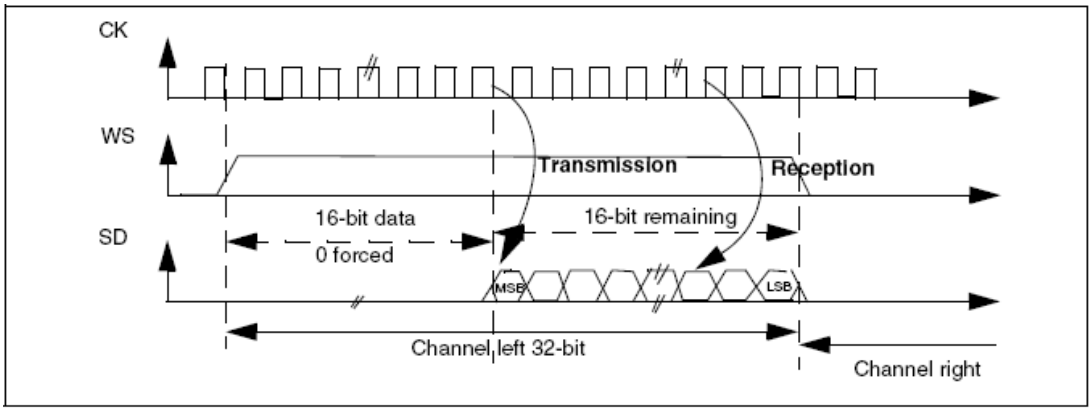


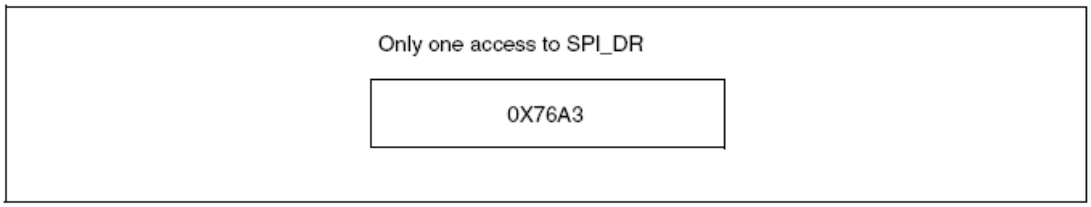
图223 LSB对齐16位数据扩展到32位包帧，CPOL = 0



在I^2S配置阶段，如果选择将16位数据扩展到32声道帧，只需要访问一次寄存器SPI_DR。此时，扩展到32位后的高半字(16位MSB)被硬件置为0x0000。

如果待传输或者接收的数据是0x76A3(扩展到32位是0x0000 76A3)，需要的操作如下图所示。

图224 示例



在发送时，如果TXE为'1'，用户需要写入待发送的数据(即0x76A3)。用来扩展到32位的0x0000部分由硬件首先发送出去，一旦有效数据开始从SD管脚送出，即发生下一次TXE事件。

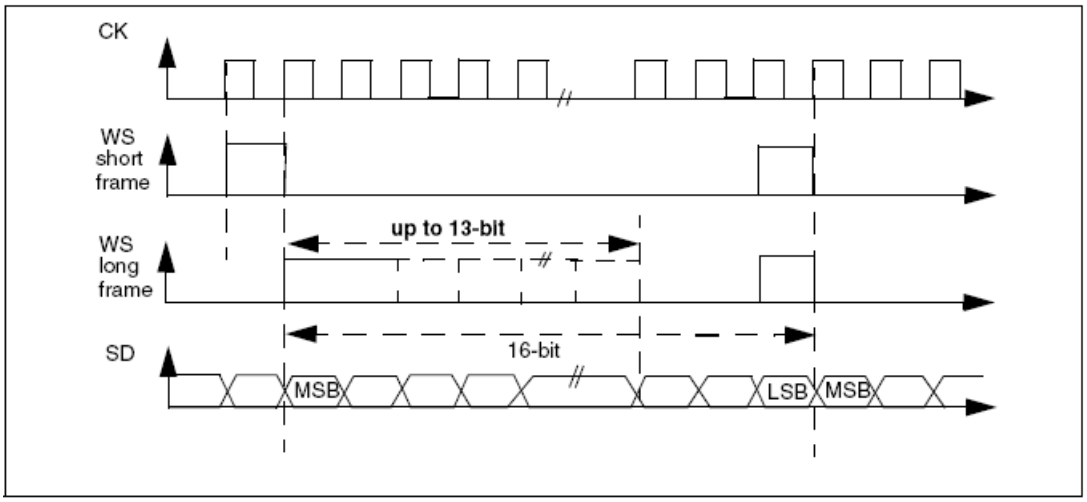
在接收时，一旦接收到有效数据(而不是0x0000部分)，即发生RXNE事件。

这样，在2次读和写之间有更多的时间，可以防止下溢或者上溢的情况发生。

PCM标准

在PCM标准下，不存在声道选择的信息。PCM标准有2种可用的帧结构，短帧或者长帧，可以通过设置寄存器SPI_I2SCFGR的PCMSYNC位来选择。

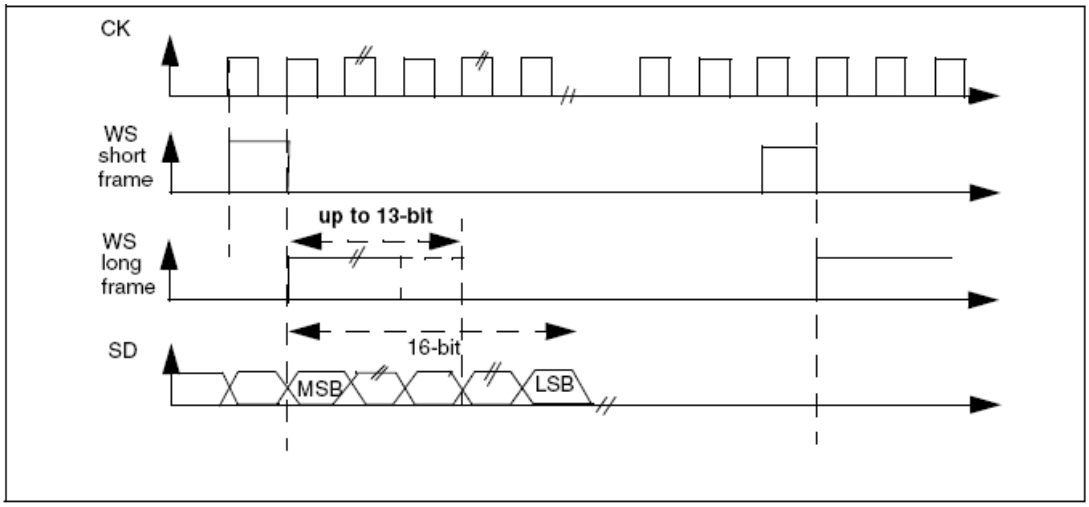
图225 PCM标准波形（16位）



对长帧，主模式下，用来同步的WS信号有效的时间固定为13位。

对短帧，用来同步的WS信号长度只有1位。

图226 PCM标准波形(16位扩展到32位包帧)



注意： *无论哪种模式(主或从)，哪种同步方式(短帧或长帧)，连续的2帧数据之间和2个同步信号之间的时间差，(即使是从模式)需要通过设置SPI_I2SCFGR寄存器的DATLEN位和CHLEN位来确定。*

22.4.3 时钟发生器

I^2S的比特率即确定了在I^2S数据线上的数据流和I^2S的时钟信号频率。

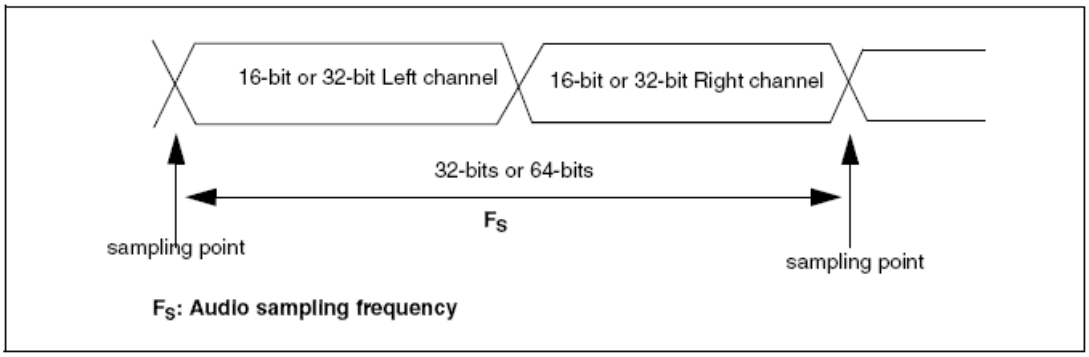
I^2S比特率 = 每个声道的比特数 × 声道数目 × 音频采样频率

对于一个左右声道，16位音频，I^2S比特率计算如下

I^2S比特率 = 16 × 2 × Fs

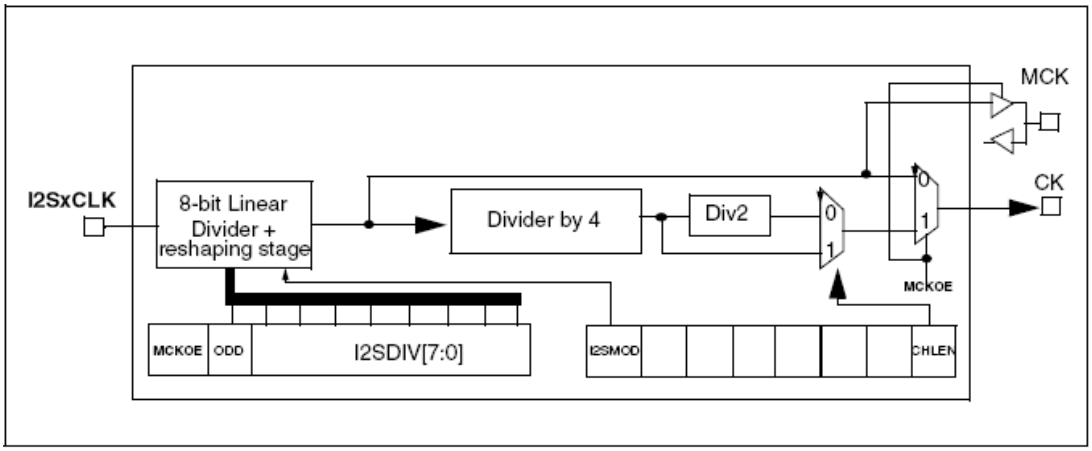
如果包长为32位，则有：I^2S比特率 = 32 × 2 × Fs

图227 音频采样频率定义



在主模式下，为了获得需要的音频频率，需要正确地对线性分频器进行设置。

图228 I²S时钟发生器结构



1. 图中x可以是2或者3。

上图中I2SxCLK的时钟源是系统时钟(即驱动AHB时钟的HSI、HSE或PLL）。

音频的采样频率可以是48kHz、44.1kHz、22.05kHz、16KHz或者8KHz。线性分频器需要按照以下的公式来设置，以获得需要的频率：

当需要生成主时钟时(寄存器SPI_I2SPR的MCKOE位为'1')：

声道的帧长为16位时，Fs = I2SxCLK / [(16*2) * ((2*I2SDIV) + ODD)*8]

声道的帧长为32位时，Fs = I2SxCLK / [(32*2) * ((2*I2SDIV) + ODD)*4]

当关闭主时钟时(MCKOE位为'0')：

声道的帧长为16位时，Fs = I2SxCLK / [(16*2) * ((2*I2SDIV) + ODD)]

声道的帧长为32位时，Fs = I2SxCLK / [(32*2) * ((2*I2SDIV) + ODD)]

22.4.4 I²S主模式

设置I²S工作在主模式，串行时钟由管脚CK输出，字选信号由管脚WS产生。可以通过设置寄存器SPI_I2SPR的MCKOE位来选择输出或者不输出主时钟(MCK)。

流程

1. 设置寄存器SPI_I2SPR的I2SDIV[7:0]定义与音频采样频率相符的串行时钟波特率。同时也要定义寄存器SPI_I2SPR的ODD位。
2. 设置CKPOL位定义通信用时钟在空闲时的电平状态。如果需要向外部的DAC/ADC音频器件提供主时钟MCK，将寄存器SPI_I2SPR的MCKOE位置1。(按照不同的MCK输出状态，计算I2SDIV和ODD的值，详见22.4.3节)。

3. 设置寄存器SPI_I2SCFGR的I2SMOD位为'1'激活I^2S功能，设置I2SSTD[1:0]和PCMSYNC位选择所用的I^2S标准，设置CHLEN选择每个声道的数据位数。还要设置寄存器SPI_I2SCFGR的I2SCFG[1:0]选择I^2S主模式和方向(发送端还是接收端)。
4. 如果需要，可以通过设置寄存器SPI_CR2来打开所需的中断功能和DMA功能。
5. 必须将寄存器SPI_I2SCFGR的I2SE位置1。
6. 管脚WS和CK需要配置为输出模式。如果寄存器SPI_I2SPR的MCKOE位置1，管脚MCK也要配置成输出模式。

发送流程

当写入1个半字(16位)的数据至发送缓存，发送流程开始。

假设第一个写入发送缓存的数据对应的是左声道数据。当数据从发送缓存移到移位寄存器时，标志位TXE置'1'，这时，要把对应右声道的数据写入发送缓存。标志位CHSIDE提示了目前待传输的数据对应哪个声道。标志位CHSIDE的值在TXE为高时更新，因此它在TXE为'1'时有意义。

在先左声道后右声道的数据都传输完成后，才能被认为是一个完整的数据帧。不可以只传输部分数据帧，如仅有左声道的数据。

当发出第一位数据的同时，半字数据被并行地传送至16位移位寄存器，然后后面的位依次按高位在先的顺序从管脚MOSI/SD发出。每次数据从发送缓存移至移位寄存器时，标志位TXE置'1'，如果寄存器SPI_CR2的TXEIE位为'1'，则产生中断。

写入数据的操作取决于所选则的I^2S标准，详见22.4.2节。

为了保证连续的音频数据传输，建议在当前传输完成之前，对寄存器SPI_DR写入下一个要传输的数据。

建议在要关闭I^2S功能时，等待标志位TXE = 0及BSY = 0，再将I2SE位清'0'。

接收流程

接收流程的配置步骤除了第3点外，与发送流程的一致。需要通过配置I2SCFG[1:0]来选则主接收模式。

无论何种数据和声道长度，音频数据总是以16位包的形式接收。即每次填满接收缓存后，标志位RXNE置'1'，如果寄存器SPI_CR2的RXNEIE位为'1'，则产生中断。根据配置的数据和声道长度，收到左声道或右声道的数据会需要1次或者2次把数据传送到接收缓存的过程。

对寄存器SPI_DR进行读操作即可清除RXNE标志位。

每次接收以后即更新CHSIDE。它的值取决于I^2S单元产生的WS信号。

读取数据的操作取决于所选则的I^2S标准，详见22.4.2节。

如果前一个接收到的数据还没有被读取，又接收到新数据，即发生上溢，标志位OVR置1，如果寄存器SPI_CR2的ERRIE位为1，则产生中断指示发生了错误。

若要关闭I^2S功能，需要在最后一个数据接收过程中，并且在接收结束之前，将I2SE位清0。即使在最后一个数据接收的过程中将I^2S功能关闭，时钟和传输仍然会维持到当前传输完成为止。

22.4.5 I^2S从模式

在从模式下，I^2S可以设置成发送和接收模式。从模式的配置方式基本遵循和配置主模式一样的流程。在从模式下，不需要I^2S接口提供时钟。时钟信号和WS信号都由外部主I^2S设备提供，连接到相应的管脚上。因此用户无需配置时钟。

配置步骤列举如下：

1. 设置寄存器SPI_I2SCFGR的I2SMOD位激活I^2S功能；设置I2SSTD[1:0]来选择所用的I^2S标准；设置DATLEN[1:0]选择数据的比特数；设置CHLEN选择每个声道的数据位数。设置寄存器SPI_I2SCFGR的I2SCFG[1:0]选择I^2S从模式的数据方向(发送端还是接收端)。
2. 根据需要，设置寄存器SPI_CR2打开所需的中断功能和DMA功能。
3. 必须设置寄存器SPI_I2SCFGR的I2SE位为'1'。

发送流程

当1个对应左声道的半字(16位)写入发送缓存时，发送流程开始。当数据从发送缓存移到移位寄存器时，标志位TXE置'1'，这时，要把对应右声道的数据写入发送缓存。标志位CHSIDE提示了目前待传输的数据对应哪个声道。与主模式的发送流程相比，在从模式中，CHSIDE取决于来自外部主I^2S的WS信号。这意味着从I^2S在接收到主端生成的时钟信号之前，就要准备好第一个要发送的数据。WS信号为'1'表示先发送左声道。

注意： *将I2SE位置1的时间，应当比主I^2S的时钟信号到达管脚CK的时刻，早至少2个PCLK时钟周期。*

当发出第一位数据的时候，半字数据并行通过I^2S内部总线传输至16位移位寄存器，然后其它位依次按高位在先的顺序从管脚MOSI/SD发出。每次数据从发送缓存移至移位寄存器时，标志位TXE置'1'，如果寄存器SPI_CR2的TXEIE位为'1'，则产生中断。

注意，在对发送缓存写入数据前，要确认标志位TXE为'1'。

写入数据的操作取决于所选中的I^2S标准，详见22.4.2节。

为了保证连续的音频数据传输，建议在当前传输完成之前，对寄存器SPI_DR写入下一个要传输的数据。如果在代表下一个数据传输的第一个时钟边沿到达之前，新的数据仍然没有写入寄存器SPI_DR，下溢标志位会置'1'，并可能产生中断，它指示软件发送数据错误。如果寄存器SPI_CR2的ERRIE位为'1'，在寄存器SPI_SR的标志位UDR为高是，就会产生中断。建议在这时关闭I^2S，然后重新从左声道开始发送数据。

接收流程

配置步骤除了第1点外，与发送流程一致。需要通过配置I2SCFG[1:0]来选则主接收模式。

无论何种数据和声道长度，音频数据总是以16位包的形式接收，即每次填满接收缓存，标志位RXNE置'1'，如果寄存器SPI_CR2的RXNEIE位为'1'，则产生中断。按照不同的数据和声道长度设置，收到左声道或者右声道数据会需要1次或者2次传输数据至接收缓存的过程。

每次接收到将从SPI_DR读出的数据以后即更新CHSIDE，它的值对应I^2S单元产生的WS信号。

读取数据的操作取决于所选中的I^2S标准，详见22.4.2节。

如果还没有读出前一个接收到的数据，又接收到新数据，即产生上溢，标志位OVR置1，如果寄存器SPI_CR2的ERRIE位为'1'，则产生中断指示发生了错误。

想要关闭I^2S功能，需要在最后一个数据接收过程中且接收结束之前，将I2SE位清0。即使在最后一个数据接收的过程中将I^2S功能关闭，时钟和传输仍然会维持到当前传输完成为止。

注意： *外部主I^2S器件需要有通过音频声道发送/接收16位或32位数据包的功能。*

22.4.6 状态标志位

有3个状态标志位供用户监控I^2S总线的状态。

忙标志位(BSY)

该标志位指示I^2S通讯层的状态。该位为'1'时表明I^2S通讯正在进行中，或有一个有效的16位半字数据在发送缓存中等待发送。该标志位的作用即指示是否在I^2S总线上有正在进行的通信。一旦发生以下情况，该标志位就置1：

1. 在主模式下，数据写入寄存器SPI_DR
2. 在从模式下，管脚CK上接收到时钟信号

一旦接收或者发送完成一个16位半字数据，标志位BSY即被清'0'。它的置'1'和置'0'都由硬件控制。检查该标志位可以避免写入数据冲突。对该标志位写入数据没有任何效果。它仅在寄存器SPI_I2SCFGR的I2SE位为'1'时有意义。

发送缓存空标志位(TXE)

该标志位为'1'指示发送缓存为空，可以对发送缓存写入新的待发送数据。在发送缓存中现有数据时，标志位清'0'。在I^2S被关闭时(I2SE位为'0')，该标志位也为'0'。

接收缓存非空标志位(RXNE)

该标志位置'1'表示在接收缓存里有接收到的有效数据。在读取寄存器SPI_DR时，该位清'0'。

声道标志位(CHSIDE)

在发送模式下，该标志位在TXE为高时刷新，指示从SD管脚上发送的数据所在的声道。如果在从发送模式下发生了下溢错误，该标志位的值无效，在重新开始通讯前需要把I^2S关闭再打开。

在接收模式下，该标志位在寄存器SPI_DR接收到数据时刷新，指示接收到的数据所在的声道。注意，如果发生错误(如上溢OVR)，该标志位无意义，需要将I^2S关闭再打开(同时，如果必要修改I^2S的配置)。

在PCM标准下，无论短帧格式还是长帧格式，这个标志位都没有意义。

如果寄存器SPI_SR的标志位OVR或UDR为'1'，且寄存器SPI_CR2的ERRIE位为'1'，则会产生中断。中断可以通过读寄存器SPI_SR来清除(如果中断源已经被清除了)。

22.4.7 错误标志位

I^2S单元有2个错误标志位。

下溢标志位(UDR)

在从发送模式下，如果新的数据传输的第一个时钟边沿到达时，新的数据仍然没有写入寄存器SPI_DR，该标志位会置'1'。该标志位在寄存器SPI_I2SCFGR的I2SMOD位置'1'后可用。如果寄存器SPI_CR2的ERRIE位为'1'，就会产生中断。

通过对寄存器SPI_SR进行读操作来清除该标志位。

上溢标志位(OVR)

如果前一个接收到的数据还没有读出，又接收到新的数据，即产生上溢，该标志位置'1'，如果寄存器SPI_CR2的ERRIE位为'1'，则产生中断指示发生了错误。

这时，接收缓存的内容，不会刷新为从发送端送来的新数据。对寄存器SPI_DR的读操作返回最后一个正确接收到的数据。其他所有在上溢发生后由发送设备发出的16位数据都会丢失。

通过先读寄存器SPI_SR再读寄存器SPI_DR，来清除该标志位。

22.4.8 I^2S中断

下表列举了全部I^2S中断

表148 I^2S中断请求

中断事件	事件标志位	使能标志位
发送缓存空标志位	TXE	TXEIE
接收缓存非空标志位	RXNE	RXNEIE
下溢标志位	OVR	ERRIE
上溢标志位	UDR	

22.4.9 DMA功能

DMA的工作方式在I^2S模式除了CRC功能不可用以外，与在SPI模式完全相同。因为在I^2S模式下没有数据传输保护系统。

22.5 SPI和I²S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。

22.5.1 SPI控制寄存器 1(SPI_CR1)(I²S模式下不使用)

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。 注意：I²S模式下不使用。
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。 注意：I²S模式下不使用。
位13	**CRCEN**: 硬件CRC校验使能 0：禁止CRC计算； 1：启动CRC计算。 注意：只有在禁止SPI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。 注意：I²S模式下不使用。
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：在SPI_DR寄存器写入最后一个数据后应马上设置该位。 注意：I²S模式下不使用。
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禁止(SPE=0)时，才能写该位，否则出错。 注意：I²S模式下不使用。
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)。 注意：I²S模式下不使用。
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。 注意：I²S模式下不使用。

位8	**SSI**: 内部从设备选择 该位只在SSM位为'1'时有意义。它决定了NSS上的电平，在NSS引脚上的I/O操作无效。 注意：I^2S模式下不使用。
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。 注意：I^2S模式下不使用。
位6	**SPE**: SPI使能 0：禁止SPI设备； 1：开启SPI设备。 注意：I^2S模式下不使用。
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。 注意：I^2S模式下不使用。
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。 注意：I^2S模式下不使用。
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。 注意：I^2S模式下不使用。
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。 注意：I^2S模式下不使用。

22.5.2 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
res								rw	rw	rw	res		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。 注意：I^2S模式下不使用。
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。

22.5.3 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	UDR	CHSIDE	TXE	RXNE
res								r	r	r	rc_w0	r	r	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。 注：在只接收的主模式(单线双向)，禁止检查BSY标志。
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22.4.7节。
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22.4.7节。 注意：I^2S模式下不使用。

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。 I^2S模式下不使用。
位3	**UDR**：下溢标志位 0：未发生下溢； 1：发生下溢。 该标志位由硬件置1，由一个软件序列清0，详见22.4.7节。 注：在SPI模式下不使用。
位2	**CHSIDE**：声道 0：需要传输或者接收左声道； 1：需要传输或者接收右声道。 注：在PCM模式下无意义。 在SPI模式下不使用。
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。

22.5.4 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）。写操作将数据写到发送缓冲区；读操作将返回接收缓冲区里的数据。 **对SPI模式的注意**：根据SPI_CR1的DFF位对数据帧格式的选择，数据的发送和接收可以是8位或者16位的。要在启用SPI之前就确定好数据帧格式。 对于8位的数据，缓冲区是8位的，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，缓冲区是16位的，发送和接收时会用到整个数据寄存器，即SPI_DR[15:0]。

22.5.5 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，根据应用可以设置其他数值。 注：在I^2S模式下不使用。

22.5.6 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计算时，RXCRC[15:0]中包含了依据收到的字节计算的CRC数值。当在SPI_CR1的CRCEN位写入'1'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标志为'1'时读该寄存器，将可能读到不正确的数值。 注：在I^2S模式下不使用。

22.5.7 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计算时，TXCRC[15:0]中包含了依据将要发送的字节计算的CRC数值。当在SPI_CR1中的CRCEN位写入'1'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标志为'1'时读该寄存器，将可能读到不正确的数值。 注：在I^2S模式下不使用。

22.5.8 SPI_I²S配置寄存器(SPI_I2S_CFGR)

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
保留				I2S MOD	I2SE	I2SCFG		PCM SYNC	保留	I2SSTD		CKPOL	DATLEN		CHLEN
res				rw	rw	rw		rw	res	rw		rw	rw		rw

位15:12	保留位，硬件强制为0
位11	**I2SMOD**：I²S模式选择 0: 选则SPI模式； 1: 选则I²S模式。 注：该位只有在关闭了SPI或者I²S时才能设置。
位10	**I2SE**: I²S使能 0: 关闭I²S； 1: I²S使能。 注：在SPI模式下不使用。
位9:8	**I2SCFG**：I²S模式设置 00: 从设备发送； 01: 从设备接收； 10: 主设备发送； 11: 主设备接受。 注：该位只有在关闭了I²S时才能设置。 在SPI模式下不使用。
位7	**PCMSYNC**：PCM帧同步 0: 短帧同步； 1: 长帧同步。 注: 该位只在I2SSTD = 11 (使用PCM标准)时有意义。 在SPI模式下不使用。
位6	保留位，硬件强制为0。
位5:4	**I2SSTD**：I²S标准选择 00: I²S飞利浦标准； 01: 高字节对齐标准 (左对齐)； 10: 低字节对齐标准(右对齐)； 11: PCM 标准。 关于I²S标准的细节，详见22.4.2节。 注：为了正确操作，该位只有在关闭了I²S时才能设置。 在SPI模式下不使用。
位3	**CKPOL**：静止态时钟极性 0: I²S时钟静止态为低电平； 1: I²S时钟静止态为高电平。 注：为了正确操作，该位只有在关闭了I²S时才能设置。 在SPI模式下不使用。

位2:1	**DATLEN**：待传输数据长度 00: 16位数据长度； 01: 24位数据长度； 10: 32位数据长度； 11: 不允许。 注：为了正确操作，该位只有在关闭了I²S时才能设置。 在SPI模式下不使用。
位0	**CHLEN**：声道长度 (每个音频声道的数据位数) 0: 16位宽； 1: 32位宽。 只有在 DATLEN = 00 时该位的写操作才有意义，否则声道长度都由硬件固定为32位。 注：为了正确操作，该位只有在关闭了I²S时才能设置。 在SPI模式下不使用。

22.5.9 SPI_I2S预分频寄存器(SPI_I2SPR)

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						MCKOE	ODD	I2SDIV							
						rw	rw	rw							

位15:10	保留位，硬件强制为0
位9	**MCKOE**：主设备时钟输出使能 0: 关闭主设备时钟输出； 1: 主设备时钟输出使能。 注：为了正确操作，该位只有在关闭了I²S时才能设置。仅在I²S主设备模式下使用该位。 在SPI模式下不使用。
位8	**ODD**：奇系数预分频 0: 实际分频系数 = I2SDIV *2； 1: 实际分频系数 = (I2SDIV * 2)+1。 参见22.4.3节。 注：为了正确操作，该位只有在关闭了I²S时才能设置。仅在I²S主设备模式下使用该位。 在SPI模式下不使用。
位7:0	I2SDIV：I²S线性预分频 禁止设置I2SDIV [7:0] = 0或者I2SDIV [7:0] = 1 参见22.4.3节。 注：为了正确操作，该位只有在关闭了I²S时才能设置。仅在I²S主设备模式下使用该位。 在SPI模式下不使用。

22.5.10 SPI 寄存器地址映象

表149 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	TXDMAEN	RXDMAEN
	复位值																									0	0	0			0	0	0
008h	SPI_SR	保留																								BSY	OVR	MODF	CRCERR	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
01Ch	SPI_I2SCFGR	保留																				I2SMOD	I2SE	I2SCFG		PCMSYNC	保留	I2SSTD		CKPOL	DATLEN	CHLEN
	复位值																					0	0	0	0	0		0	0	0	0	0	0
020h	SPI_I2SPR	保留																							MCKOE	ODD	I2SDIV						
	复位值																								0	0	0	0	0	0	0	1	0

关于寄存器的起始地址，参见表1。

23 I²C接口

小容量产品是指闪存存储器容量在16K至32K字节之间的STM32F101xx、STM32F102xx和STM32F103xx微控制器。

中容量产品是指闪存存储器容量在64K至128K字节之间的STM32F101xx、STM32F102xx和STM32F103xx微控制器。

大容量产品是指闪存存储器容量在256K至512K字节之间的STM32F101xx和STM32F103xx微控制器。

除非特别说明，本章节描述的模块应用于整个STM32F10xxx微控制器系列。

23.1 I²C简介

I²C(芯片间)总线接口连接微控制器和串行I²C总线。它提供多主机功能，控制所有I²C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。

I²C模块有多种用途，包括CRC码的生成和校验、SMBus(系统管理总线—System Management Bus)和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。

23.2 I²C主要特点

- 并行总线/I²C总线协议转换器
- 多主机功能：该模块既可做主设备也可做从设备
- I²C主设备功能
  - 产生时钟
  - 产生起始和停止信号
- I²C从设备功能
  - 可编程的 I²C 地址检测
  - 可响应 2 个从地址的双地址能力
  - 停止位检测
- 产生和检测7位/10位地址和广播呼叫
- 支持不同的通讯速度
  - 标准速度(高达 100 kHz)
  - 快速(高达 400 kHz)
- 状态标志：
  - 发送器/接收器模式标志
  - 字节发送结束标志
  - I²C 总线忙标志
- 错误标志
  - 主模式时的仲裁丢失
  - 地址/数据传输后的应答(ACK)错误
  - 检测到错位的起始或停止条件
  - 禁止拉长时钟功能时的上溢或下溢
- 2个中断向量
  - 1 个中断用于地址/数据通讯成功
  - 1 个中断用于错误
- 可选的拉长时钟功能
- 具单字节缓冲器的DMA
- 可配置的PEC(信息包错误检测)的产生或校验：

- 发送模式中 PEC 值可以作为最后一个字节传输
- 用于最后一个接收字节的 PEC 错误校验

- 兼容SMBus 2.0
  - 25 ms 时钟低超时延时
  - 10 ms 主设备累积时钟低扩展时间
  - 25 ms 从设备累积时钟低扩展时间
  - 带 ACK 控制的硬件 PEC 产生/校验
  - 支持地址分辨协议(ARP)
- 兼容SMBus

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I²C功能。

23.3 I²C功能描述

I²C模块接收和发送数据，并将数据从串行转换成并行，或并行转换成串行。可以开启或禁止中断。接口通过数据引脚(SDA)和时钟引脚(SCL)连接到I²C总线。允许连接到标准(高达100kHz)或快速(高达400kHz)的I²C总线。

23.3.1 模式选择

接口可以下述4种模式中的一种运行：

- 从发送器模式
- 从接收器模式
- 主发送器模式
- 主接收器模式

该模块默认地工作于从模式。接口在生成起始条件后自动地从从模式切换到主模式；当仲裁丢失或产生停止信号时，则从主模式切换到从模式。允许多主机功能。

通信流

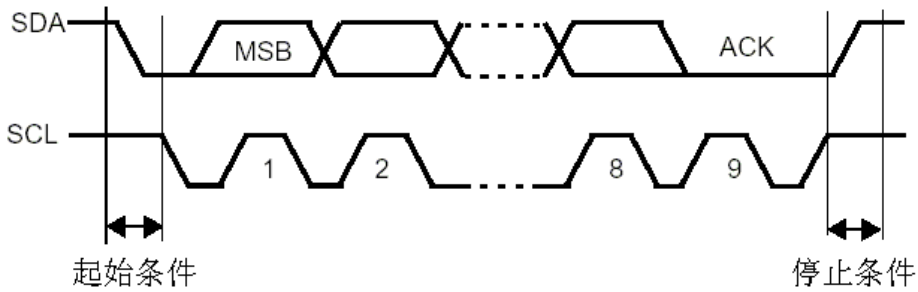
主模式时，I²C接口启动数据传输并产生时钟信号。串行数据传输总是以起始条件开始并以停止条件结束。起始条件和停止条件都是在主模式下由软件控制产生。

从模式时，I²C接口能识别它自己的地址(7位或10位)和广播呼叫地址。软件能够控制开启或禁止广播呼叫地址的识别。

数据和地址按8位/字节进行传输，高位在前。跟在起始条件后的1或2个字节是地址(7位模式为1个字节，10位模式为2个字节)。地址只在主模式发送。

在一个字节传输的8个时钟后的第9个时钟期间，接收器必须回送一个应答位(ACK)给发送器。参考下图。

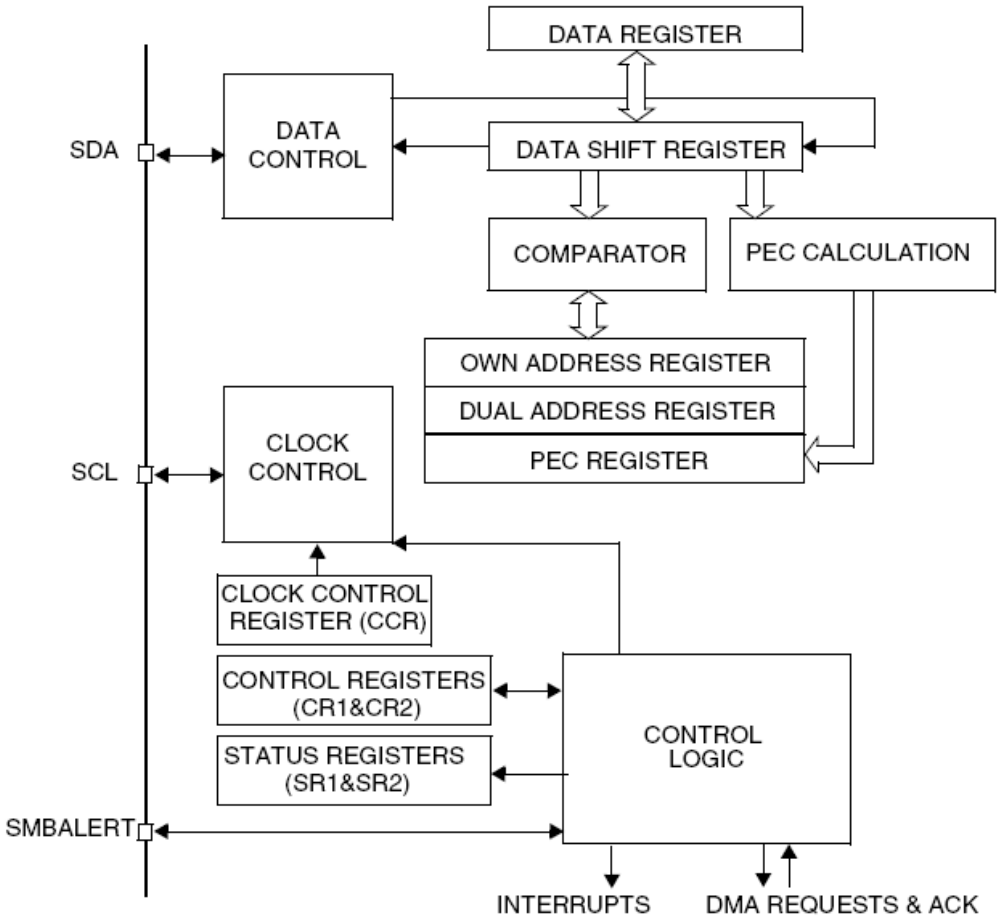
图229 I²C总线协议



软件可以开启或禁止应答(ACK)，并可以设置I²C接口的地址(7位、10位地址或广播呼叫地址)。

I²C接口的功能框图示于图230。

图230 I²C的功能框图



注： *在SMBus模式下，SMBALERT是可选信号。如果禁止了SMBus，则不能使用该信号。*

23.3.2 I²C从模式

默认情况下，I²C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。

为了产生正确的时序，必须在I2C_CR2寄存器中设定该模块的输入时钟。输入时钟的频率必须至少是：

- 标准模式下为：2MHz
- 快速模式下为：4MHz

一旦检测到起始条件，在SDA线上接收到的地址被送到移位寄存器。然后与芯片自己的地址OAR1和OAR2(当ENDUAL=1)或者广播呼叫地址(如果ENGC=1)相比较。

注： *在10位地址模式时，比较包括头段序列(11110xx0)，其中的xx是地址的两个最高有效位。*

头段或地址不匹配：I²C接口将其忽略并等待另一个起始条件。

头段匹配(仅10位模式)：如果ACK位被置'1'，I²C接口产生一个应答脉冲并等待8位从地址。

地址匹配：I²C接口产生以下时序：

- 如果ACK被置'1'，则产生一个应答脉冲
- 硬件设置ADDR位；如果设置了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。

在从模式下TRA位指示当前是处于接收器模式还是发送器模式。

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。

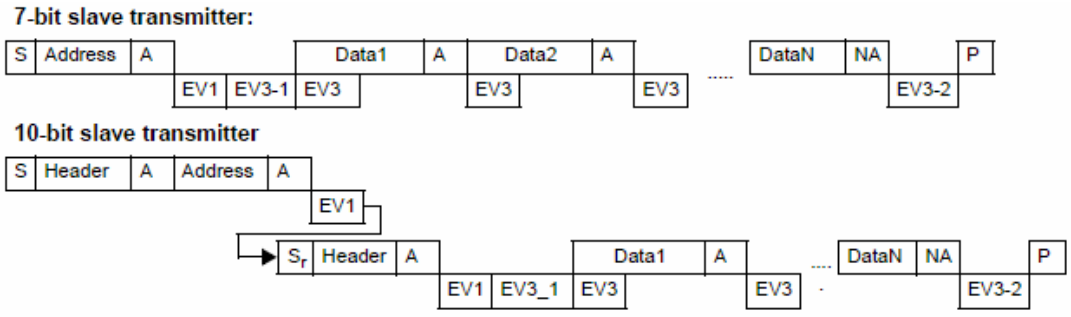
从设备保持SCL为低电平，直到ADDR位被清除并且待发送数据已写入DR寄存器。(见图231中的EV1和EV3)。

当收到应答脉冲时:

- TxE位被硬件置位，如果设置了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新数据写入到DR寄存器，则BTF位被置位，I²C接口将保持SCL为低电平，以等待写入DR寄存器。

图231 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，写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'可清除AF位。

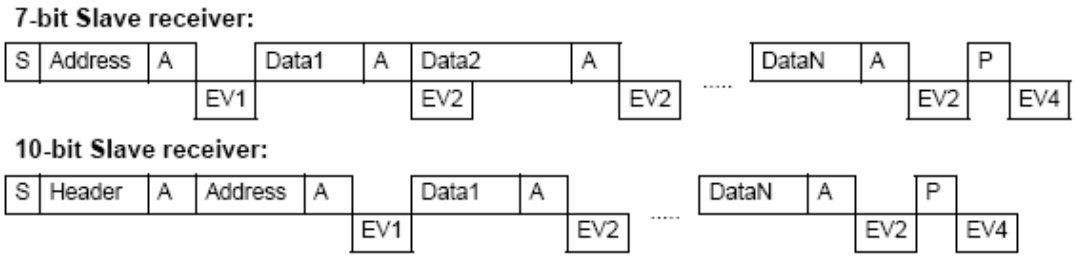
从接收器

在接收到地址并清除ADDR后，从接收器将通过内部移位寄存器从SDA线接收到的字节存进DR寄存器。I²C接口在接收到每个字节后都执行下列操作:

- 如果设置了ACK位，则产生一个应答脉冲
- 硬件设置RxNE=1。如果设置了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，I²C接口保持SCL为低电平，等待读DR寄存器(见下图)。

图232 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

在传输完最后一个数据字节后，主设备产生一个停止条件， I²C接口检测到这一条件时：

- 设置STOPF=1，如果设置了ITEVFEN位，则产生一个中断。

然后I²C接口等待读SR1寄存器，再写CR1寄存器。(见图232的EV4)。

23.3.3 I²C主模式

在主模式时，I²C接口启动数据传输并产生时钟信号。串行数据传输总是以起始条件开始并以停止条件结束。当通过START位在总线上产生了起始条件，设备就进入了主模式。

以下是主模式所要求的操作顺序：

- 在I2C_CR2寄存器中设定该模块的输入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_CR1寄存器启动外设
- 置I2C_CR1寄存器中的START位为1，产生起始条件

I²C模块的输入时钟频率必须至少是:

- 标准模式下为：2MHz
- 快速模式下为：4MHz

起始条件

当BUSY=0时，设置START=1，I²C接口将产生一个开始条件并切换至主模式(M/SL位置位)。

注： *在主模式下，设置START位将在当前字节传输完后由硬件产生一个重开始条件。*

一旦发出开始条件:

- SB位被硬件置位，如果设置了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233和图234的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。
    然后主设备等待读SR1寄存器，再将第二个地址字节写入DR寄存器(见图233和图234)。
  - 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233和图234)。
- 在7位地址模式时，只需送出一个地址字节。
  一旦该地址字节被送出，
  - ADDR 位被硬件置位，如果设置了 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233和图234)。

根据送出从地址的最低位，主设备决定进入发送器模式还是进入接收器模式。

- 在7位地址模式时，
  - 要进入发送器模式，主设备发送从地址时置最低位为'0'。
  - 要进入接收器模式，主设备发送从地址时置最低位为'1'。
- 在10位地址模式时
  - 要进入发送器模式，主设备先送头字节(11110xx0)，然后送最低位为'0'的从地址。(这里 xx 代表 10 位地址中的最高 2 位。)

– 要进入接收器模式，主设备先送头字节(11110xx0)，然后送最低位为'1'的从地址。然后再重新发送一个开始条件，后面跟着头字节(11110xx1)(这里 xx 代表 10 位地址中的最高 2 位。)

TRA位指示主设备是在接收器模式还是发送器模式。

主发送器

在发送了地址和清除了ADDR位后，主设备通过内部移位寄存器将字节从DR寄存器发送到SDA线上。

主设备等待，直到TxE被清除，(见图233的EV8)。

当收到应答脉冲时:

- TxE位被硬件置位，如果设置了INEVFEN和ITBUFEN位，则产生一个中断。

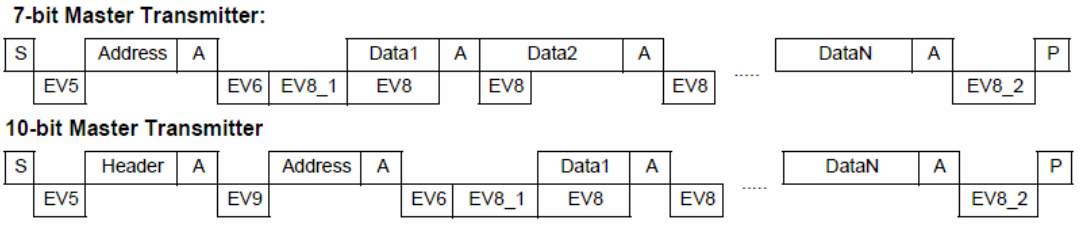
如果TxE被置位并且在上一次数据发送结束之前没有写新的数据字节到DR寄存器，则BTF被置位，I²C接口等待BTF被清除。

关闭通信

在DR寄存器中写入最后一个字节后，通过设置STOP位产生一个停止条件(见图233的EV8_2)，然后I²C接口将自动回到从模式(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233 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，I²C接口进入主接收器模式。在此模式下，I²C接口从SDA线接收数据字节，并通过内部移位寄存器送至DR寄存器。在每个字节后，I²C接口依次执行以下操作：

- 如果ACK位被置位，发出一个应答脉冲。
- 硬件设置RxNE=1，如果设置了INEVFEN和ITBUFEN位，则会产生一个中断(见图234的EV7)。

如果RxNE位被置位，并且在接收新数据结束前，DR寄存器中的数据没有被读走，硬件将设置BTF=1，I²C接口等待读DR寄存器。

关闭通信

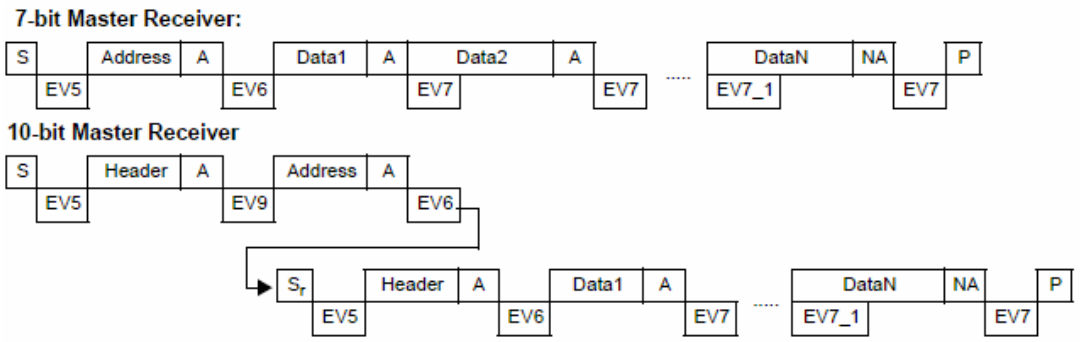
主设备在从从设备接收到最后一个字节后发送一个NACK。从设备接收到NACK后，释放对SCL和SDA线的控制；主设备就可以发送一个停止/重起始条件。

- 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。
- 只接收一个字节时，将在EV6时进行关闭应答和停止条件生成操作。

在产生了停止条件后，I²C接口自动回到从模式(M/SL位被清除)。

图234 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。

23.3.4 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总线错误。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：
  - 如果是错误的开始条件，从设备认为是一个重启动，并等待地址或停止条件。
  - 如果是错误的停止条件，从设备按正常的停止条件操作，同时硬件释放总线。

应答错误(AF)

当接口检测到一个无应答位时，产生应答错误。此时：

- AF位被置位，如果设置了ITERREN位，则产生一个中断；
- 当发送器接收到一个NACK时，必须复位通讯：
  - 如果是处于从模式，硬件释放总线。
  - 如果是处于主模式，软件必须生成一个停止条件。

仲裁丢失(ARLO)

当I²C接口检测到仲裁丢失时产生仲裁丢失错误，此时：

- ARLO位被硬件置位，如果设置了ITERREN位，则产生一个中断；
- I²C接口自动回到从模式(M/SL位被清除)；
- 硬件释放总线。

过载/欠载错误(OVR)

在从模式下，如果禁止时钟延长，I²C接口正在接收数据时，当它已经接收到一个字节(RxNE=1)，但在DR寄存器中前一个字节数据还没有被读出，则发生过载错误。此时：

- 最后接收的数据被丢弃；
- 在过载错误时，软件应清除RxNE位，发送器应该重新发送最后一次发送的字节。

在从模式下，如果禁止时钟延长，I²C接口正在发送数据时，在下一个字节的时钟到达之前，新的数据还未写入DR寄存器(TxE=1)，则发生欠载错误。此时：

- 在DR寄存器中的前一个字节将被重复发出；
- 用户应该确定在发生欠载错时，接收端应丢弃重复接收到的数据。发送端应按I²C总线标准在规定的时间更新DR寄存器。

23.3.5 SDA/SCL线控制

- 如果允许时钟延长：
  - 发送器模式：如果 TxE=1 且 BTF=1：I²C 接口在传输前保持时钟线为低，以等待软件读取 SR1，然后把数据写进数据寄存器(缓冲器和移位寄存器都是空的)。
  - 接收器模式：如果 RxNE=1 且 BTF=1：I²C 接口在接收到数据字节后保持时钟线为低，以等待软件读 SR1，然后读数据寄存器 DR(缓冲器和移位寄存器都是满的)。
- 如果在从模式中禁止时钟延长：
  - 如果 RxNE=1，在接收到下个字节前 DR 还没有被读出，则发生过载错。接收到的最后一个字节丢失。
  - 如果 TxE=1，在必须发送下个字节之前却没有新数据写进 DR，则发生欠载错。相同的字节将被重复发出。
  - 不控制重复写冲突。

23.3.6 SMBus

介绍

系统管理总线(SMBus)是一个双线接口。通过它，各设备之间以及设备与系统的其他部分之间可以互相通信。它基于I²C操作原理。SMBus为系统和电源管理相关的任务提供一条控制总线。一个系统利用SMBus可以和多个设备互传信息，而不需使用独立的控制线路。

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：用来发送命令、产生时钟和终止发送的设备。主机：一种专用的主设备，它提供与系统CPU的主接口。主机必须具有主-从机功能并且必须支持SMBus提醒协议。一个系统里只允许有一个主机。

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229)；

SMBus和I²C之间的不同点

下表列出了SMBus和I²C的不同点。

表150 SMBus与I²C的比较

SMBus	I²C
最大传输速度100kHz	最大传输速度400kHz
最小传输速度10kHz	无最小传输速度
35ms时钟低超时	无时钟超时

固定的逻辑电平	逻辑电平由VDD决定
不同的地址类型(保留的、动态的等)	7位、10位和广播呼叫从地址类型
不同的总线协议(快速命令、处理呼叫等)	无总线协议

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)

通过给每个从设备动态地分配一个新的唯一地址，可以解决SMBus的从地址冲突。地址解析协议(ARP)具有以下的特性：

- 使用标准SMBus物理层仲裁机制分配地址；
- 当设备维持供电期间，分配的地址仍保持不变，也允许设备在断电后保留其地址。
- 在地址分配后，没有额外的SMBus的打包开销(也就是说访问分配地址的设备与访问固定地址的设备所用时间是一样的)；
- 任何一个SMBus主设备可以遍历总线。

唯一的设备标识符(UDID)

为了分配地址，需要一种区分每个设备的机制，每个设备必须拥有一个唯一的设备标识符。

关于在ARP上128位的UDID的详细信息，参考2.0版的SMBus规范(*http://smbus.org/specs/*)。

SMBus提醒模式

SMBus提醒是一个带中断线的可选信号，用于那些希望扩展它们的控制能力而牺牲一个引脚的设备。SMBALERT和SCL、SDA信号一样，是一种线与信号。SMBALERT通常和SMBus广播呼叫地址一起使用。与SMBus有关的消息为2字节。

一个只具有从功能的设备，可以通过设置I2C_CR1寄存器上的ALERT位，使用SMBALERT给主机发信号表示它希望进行通信。主机处理该中断并通过提醒响应地址ARA(*Alert Response Address*，地址值为0001100x)访问所有SMBALERT设备。只有那些将SMBALERT拉低的设备能应答ARA。此状态是由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。

没有实现SMBALERT信号的主机可以定期访问ARA。

有关SMBus提醒模式的更多详细资料，请参考2.0版的SMBus规范(*http://smbus.org/specs/*)。

超时错误

在定时规范上I²C和SMBus之间有很多差别。

SMBus定义了一个时钟低超时，35ms的超时。SMBus规定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错误表明了这个特性的状态。

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。

如果要把设备配置成主设备，产生起始条件的步骤见23.3.3节I2C主模式。否则，参见23.3.2节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。

23.3.7 DMA请求

DMA请求(当被使能时)仅用于数据传输。发送时数据寄存器变空或接收时数据寄存器变满，则产生DMA请求。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中断服务程序中，可以清除TxE/RxNE标记以达到连续的通信。

利用DMA发送

通过设置I2C_CR2寄存器中的DMAEN位可以激活DMA模式。只要TxE位被置位，数据将由DMA从预置的存储区装载进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事件后从存储器传送至这个地址。
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传送至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ITBUFEN位。

利用DMA接收

通过设置I2C_CR2寄存器中的DMAEN位可以激活DMA接收模式。每次接收到数据字节时，将由DMA把I2C_DR寄存器的数据传送到设置的存储区(参考DMA说明)。设置DMA通道进行I²C接收，须执行以下步骤(x是通道号)：

1. 在DMA_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ITBUFEN位。*

23.3.8 包错误校验(PEC)

包错误校验(PEC)计算器是用于提高通信的可靠性，这个计算器使用下述CRC-8多项式对每一位串行数据进行计算：

$$C(x) = x^8 + x^2 + x + 1$$

- PEC计算由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 寄存器的 PEC 传输位，PEC 将在最后一个字节后被发送。
  - 在接收时：在最后一个 RxNE 事件之后设置 I2C_CR1 寄存器的 PEC 位，如果下个接收到的字节不等于内部计算的 PEC，接收器发送一个 NACK。如果是主接收器，不管校对的结果如何， PEC 后都将发送 NACK。
- 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。

23.4 I²C中断请求

下表列出了所有的I²C中断请求

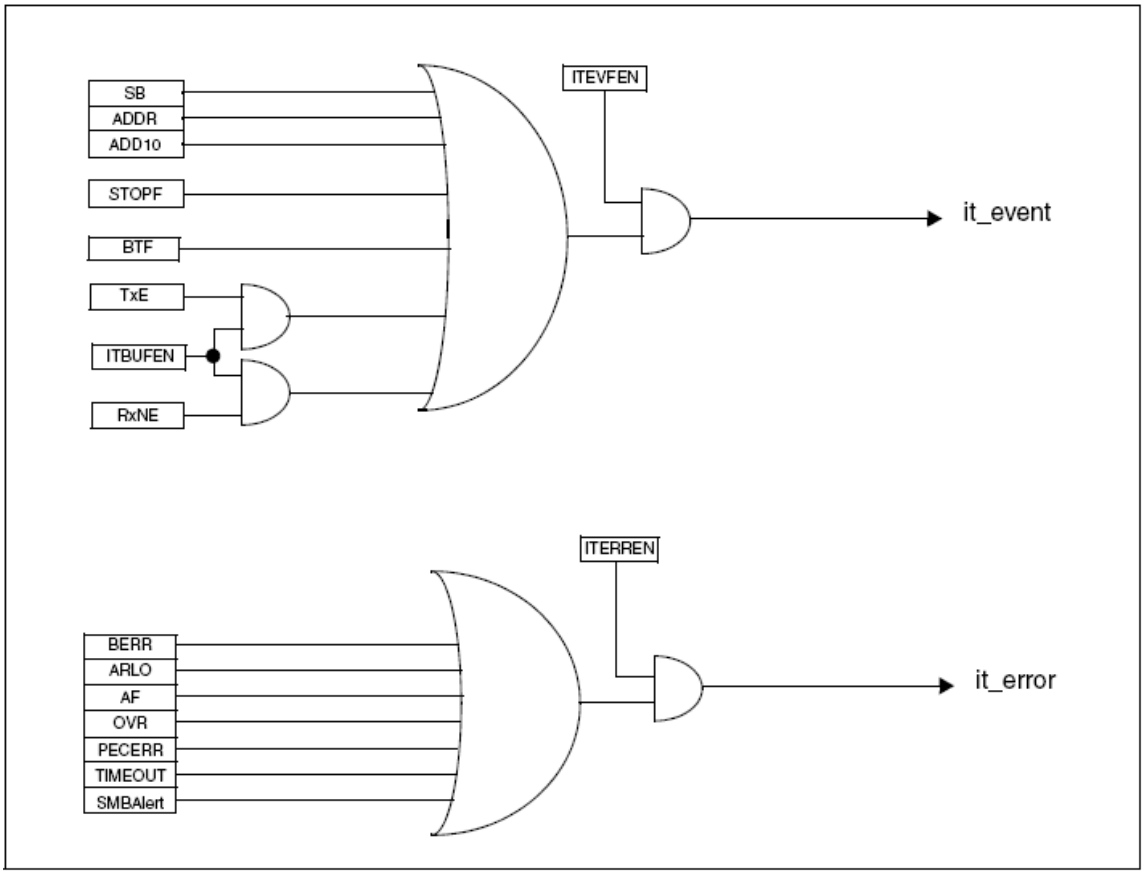
表151 I²C中断请求表：

中断事件	事件标志	开启控制位
起始位已发送(主)	SB	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和TxE通过逻辑或汇到同一个中断通道中。*

2. BERR、ARLO、AF、OVR、PECERR、TIMEOUT和SMBALERT通过逻辑或汇到同一个中断通道中。

图235 I²C中断映射图



23.5 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26.15.2节。

23.6 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。

23.6.1 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	res	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	res	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：I²C模块不处于复位状态； 1: I²C模块处于复位状态。 **注：**该位可以用于BUSY位为'1'，在总线上又没有检测到停止条件时。
位14	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：**仲裁丢失时，PEC的计算失效。
位11	**POS**: 应答/PEC位置(用于数据接收) 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模块使能 0：禁用I²C模块； 1：启用I²C模块：根据SMBus位的设置，相应的I/O口需配置为复用功能。 注：如果清除该位时通讯正在进行，在当前通讯结束后，I²C模块被禁用并返回空闲状态。 由于在通讯结束后发生PE＝0，所有的位被清除。 在主模式下，通讯结束之前，绝不能清除该位。

23.6.2 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。
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5:0	**FREQ[5:0]**：I²C模块时钟频率 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2～36MHz之间： 000000：禁用 000001：禁用 000010：2MHz ... 100100：36MHz 大于100100：禁用。

23.6.3 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	res	res				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位15	**ADDMODE**：寻址模式(从模式) 0：7位从地址(不响应10位地址)； 1：10位从地址(不响应7位地址)。
位14	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。

23.6.4 自身地址寄存器 2(I2C_OAR2)

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
保留								ADD2[7:1]							ENDUAL
res								rw	rw	rw	rw	rw	rw	rw	rw

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。

23.6.5 数据寄存器(I2C_DR)

地址偏移：0x10

复位值：0x0000

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

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：当写一个字节至DR寄存器时，自动启动数据传输。一旦传输开始(TxE=1)，如果能及时把下一个需传输的数据写入DR寄存器，I²C模块将保持连续的数据流。 接收器模式：接收到的字节被拷贝到DR寄存器(RxNE=1)。在接收到下一个字节之前，必须读出数据寄存器内已收到的数据，否则将产生过载错误同时最后一个字节将丢失。

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脉冲时发生ARLO事件，接收到的字节不会被拷贝到数据寄存器里，因此不能读到它。

23.6.6 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 w0	rc w0	res	rc w0	rc w0	rc w0	rc w0	rc w0	r	r	res	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。 – 该位由软件写'0'清除，或在PE=0时由硬件清除。
位14	**TIMEOUT**: 超时或Tlow错误 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。 – 该位由软件写'0'清除，或在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)。 – 该位由软件写'0'清除，或在PE=0时由硬件清除。
位11	**OVR**：过载/欠载 0: 无过载/欠载； 1: 出现过载/欠载。 – 当NOSTRETCH=1时，在从模式下该位被硬件置位，同时： – 在接收模式中当收到一个新的字节时(包括ACK应答脉冲)，数据寄存器里的内容还未被读出，则新接收的字节将丢失。 – 在发送模式中当要发送一个新的字节时，却没有新的数据写入数据寄存器，同样的字节将被发送两次。 – 该位由软件写'0'清除，或在PE=0时由硬件清除。。 **注**：如果数据寄存器的写操作发生时间非常接近SCL的上升沿，发送的数据是不确定的，并发生保持时间错误。

位10	**AF**：应答失败 0：没有应答失败； 1：应答失败。 – 当没有返回应答时，硬件将置该位为'1'。 – 该位由软件写'0'清除，或在PE=0时由硬件清除。
位9	**ARLO**: 仲裁丢失(主模式) 0：没有检测到仲裁丢失； 1：检测到仲裁丢失。 当接口失去对总线的控制给另一个主机时，硬件将将置该位为'1'。 – 该位由软件写'0'清除，或在PE=0时由硬件清除。 在ARLO事件之后，I²C接口自动切换回从模式(M/SL=0)。 **注：**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。 – 当接口检测到错误的起始或停止条件，硬件将该位置'1'。 – 该位由软件写'0'清除，或在PE=0时由硬件清除。
位7	**TxE**: 数据寄存器为空(发送时) 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。 – 软件写数据到DR寄存器可清除该位；或在发生一个起始或停止条件后，或当PE=0时由硬件自动清除。 如果收到一个NACK，或下一个要发送的字节是PEC(PEC=1)，该位不被置位。
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：主设备已经将第一个地址字节发送出去。 – 在10位地址模式下，当主设备已经将第一个字节发送出去时，硬件将该位置'1'。 – 软件读取SR1寄存器后，对CR1寄存器的写操作将清除该位，或当PE=0时，硬件清除该位。 **注：**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'1'。 **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，写数据寄存器的操作将清除该位，或当PE=0时，硬件清除该位。

23.6.7 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	res	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**：SMBus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，该位根据地址字节的R/W位来设定。 在检测到停止条件(STOPF=1)、重复的起始条件或总线仲裁丢失(ARLO=1)后，或当PE=0时，硬件将其清除。
位1	**BUSY**：总线忙 0：在总线上无数据通讯； 1：在总线上正在进行数据通讯。 – 在检测到SDA或SCl为低电平时，硬件将该位置'1'； – 当检测到一个停止条件时，硬件将该位清除。 该位指示当前正在进行的总线通讯，当接口被禁用(PE=0)时该信息仍然被更新。
位0	**MSL**：主从模式 0：从模式； 1：主模式。 – 当接口处于主模式(SB=1)时，硬件将该位置位； – 当总线上检测到一个停止条件、仲裁丢失(ARLO=1时)、或当PE=0时，硬件清除该位。

23.6.8 时钟控制寄存器(I2C_CCR)

地址偏移：0x1C

复位值：0x0000

注： *1. 要求F_{PCLK1}应当是10 MHz的整数倍，这样可以正确地产生400KHz的快速时钟。*

2. CCR寄存器只有在关闭I²C时才能设置

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I²C； 1：快速模式的I²C。
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I²C标准模式或SMBus模式下： T_{high} = CCR ×T_{PCLK1} T_{low} = CCR ×T_{PCLK1} 在I²C快速模式下： 如果DUTY = 0： T_{high} = CCR ×T_{PCLK1} T_{low} = 2 × CCR × T_{PCLK1} 如果DUTY = 1：(速度达到400kHz) T_{high} = 9 × CCR ×T_{PCLK1} T_{low} = 16 × CCR ×T_{PCLK1} 例如：在标准模式下，产生100kHz的SCL的频率： 如果FREQR = 08，T_{PCLK1} = 125ns，则CCR必须写入0x28(40×125ns = 5000 ns)。 **注**：1. 允许设定的最小值为0x04，在快速DUTY模式下允许的最小值为0x01； 2. T_{high}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； 5. CCR寄存器只有在关闭I²C时才能设置(PE = 0)； 6. f_{CK}应当是10MHz的整数倍，这样可以正确产生400kHz的快速时钟。

23.6.9 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]					
res										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) 这些位必须设置为I²C总线规范里给出的最大的SCL上升时间，增长步幅为1。 例如：标准模式中最大允许SCL上升时间为1000ns。如果在I2C_CR2寄存器中FREQ[5:0]中的值等于0x08且T_{PCLK1}=125ns，故TRISE[5:0]中必须写入09h(1000ns/125 ns = 8+1)。 滤波器的值也可以加到TRISE[5:0]内。 如果结果不是一个整数，则将整数部分写入TRISE[5:0]以确保t_{HIGH}参数。 注：只有当I²C被禁用(PE=0)时，才能设置TRISE[5:0]。

23.6.10 I²C寄存器地址映象

表152 I²C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
0x00	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
0x04	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	0
0x08	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	0	0
0x0C	I2C_OAR2	保留																								ADD2[7:1]							ENDUAL
	复位值																									0	0	0	0	0	0	0	0
0x10	I2C_DR	保留																								DR[7:0]							
	复位值																									0	0	0	0	0	0	0	0
0x14	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
0x18	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
0x1C	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
0x20	I2C_TRISE	保留																												TRISE[5:0]			
	复位值																													0	0	0	0	1	0

关于寄存器起始地址，参见表1。

24 通用同步异步收发器(USART)

小容量产品是指闪存存储器容量在16K至32K字节之间的STM32F101xx、STM32F102xx和STM32F103xx微控制器。

中容量产品是指闪存存储器容量在64K至128K字节之间的STM32F101xx、STM32F102xx和STM32F103xx微控制器。

大容量产品是指闪存存储器容量在256K至512K字节之间的STM32F101xx和STM32F103xx微控制器。

除非特别说明，本章节描述的模块应用于整个STM32F10xxx微控制器系列。

24.1 USART介绍

通用同步异步收发器(USART)提供了一种灵活的方法与使用工业标准NRZ异步串行数据格式的外部设备之间进行全双工数据交换。USART利用分数波特率发生器提供宽范围的波特率选择。

它支持同步单向通信和半双工单线通信，也支持LIN(局部互连网)，智能卡协议和IrDA(红外数据组织)SIR ENDEC规范，以及调制解调器(CTS/RTS)操作。它还允许多处理器通信。

使用多缓冲器配置的DMA方式，可以实现高速数据通信。

24.2 USART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，最高达 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 标准里定义的异步智能卡协议
  - 智能卡用到的 0.5 和 1.5 个停止位
- 单线半双工通信
- 可配置的使用DMA的多缓冲器通信
  - 在 SRAM 里利用集中式 DMA 缓冲接收/发送字节
- 单独的发送器和接收器使能位
- 检测标志
  - 接收缓冲器满
  - 发送缓冲器空
  - 传输结束标志
- 校验控制
  - 发送校验位
  - 对接收数据进行校验
- 四个错误检测标志
  - 溢出错误

- 噪音错误
- 帧错误
- 校验错误

- 10个带标志的中断源
  - CTS 改变
  - LIN 断开符检测
  - 发送数据寄存器空
  - 发送完成
  - 接收数据寄存器满
  - 检测到总线为空闲
  - 溢出错误
  - 帧错误
  - 噪音错误
  - 校验错误
- 多处理器通信 -- 如果地址不匹配，则进入静默模式
- 从静默模式中唤醒（通过空闲总线检测或地址标志检测）
- 两种唤醒接收器的方式：地址位(MSB，第9位)，总线空闲

24.3 USART功能概述

接口通过三个引脚与其他设备连接在一起(见图236)。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，并且不发送数据时，TX引脚处于高电平。在单线和智能卡模式里，此I/O口被同时用于数据的发送和接收。

- 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0节。

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。

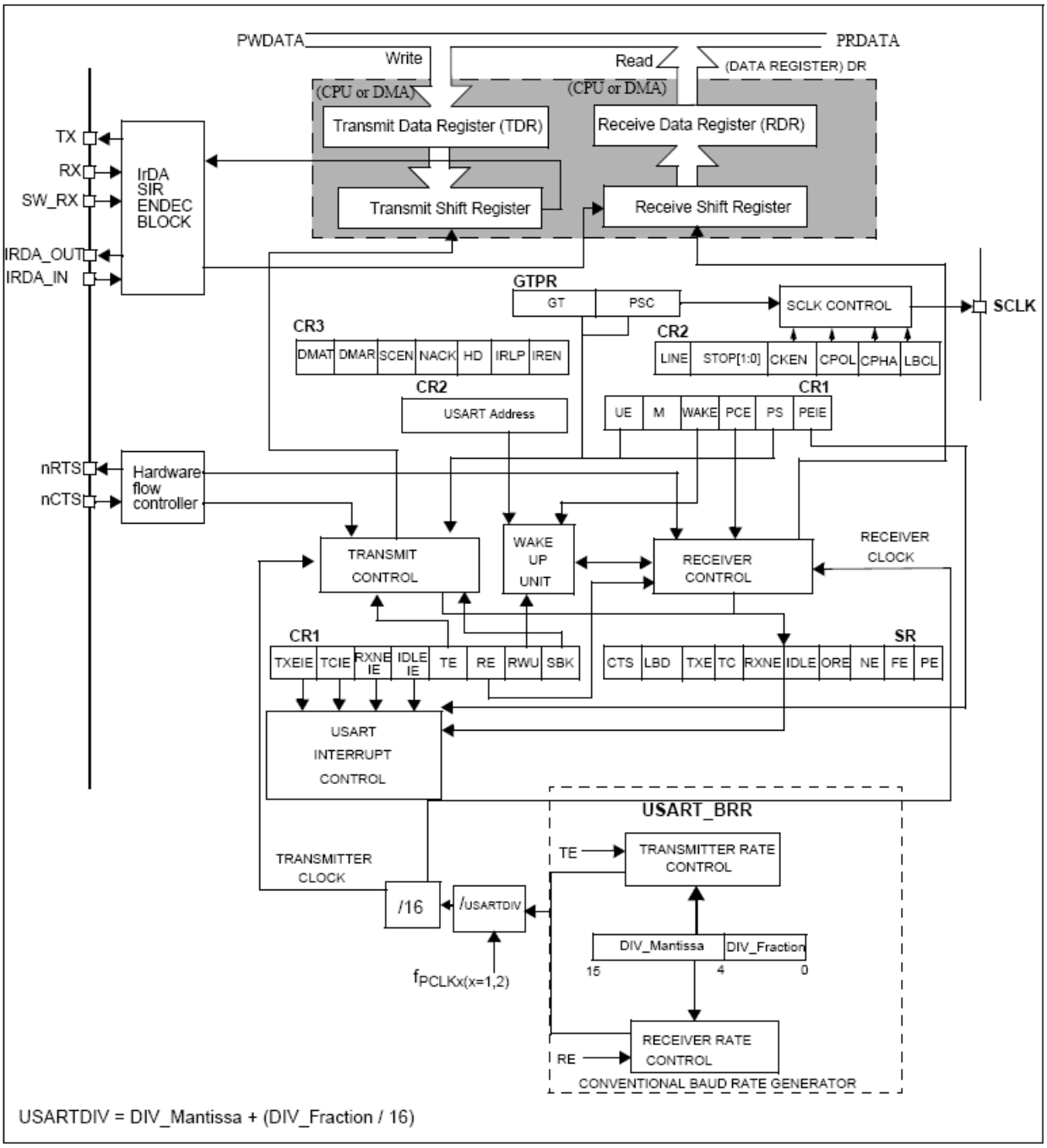
在IrDA模式里需要下列引脚：

- **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

图236 USART框图



24.3.1 USART 特性描述

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0)。在起始位期间，TX脚处于低电平，在停止位期间处于高电平。

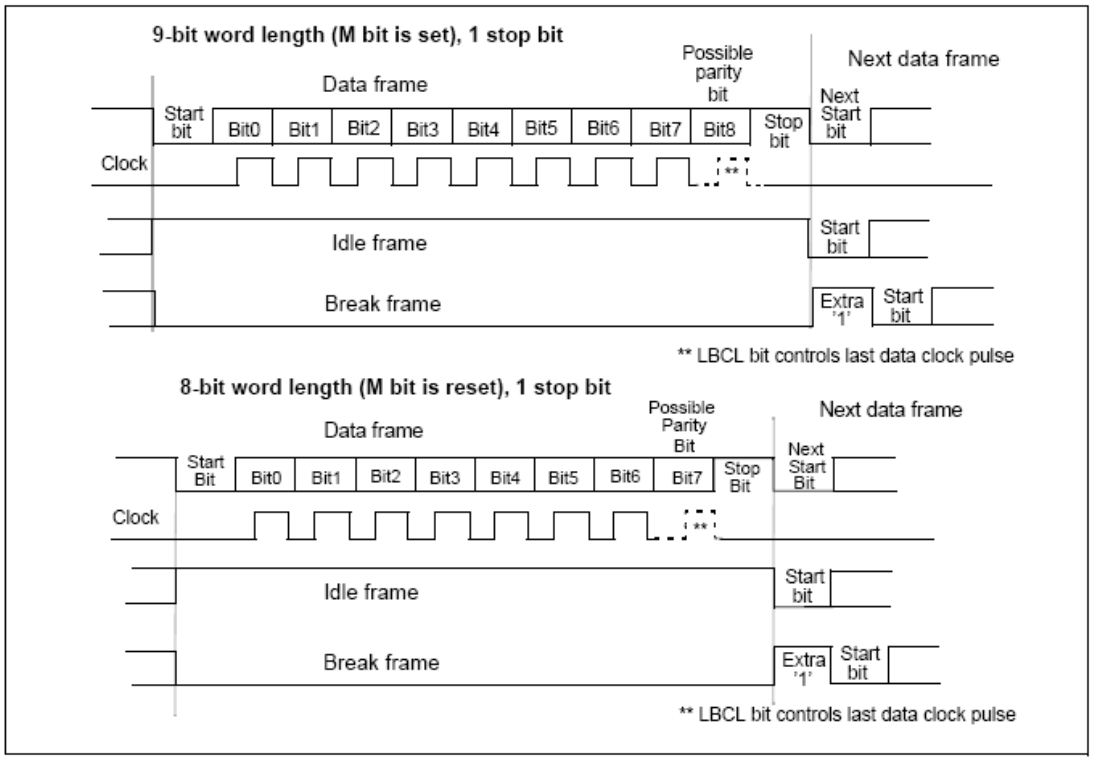
空闲符号被视为完全由'1'组成的一个完整的数据帧，后面跟着包含了数据的下一帧的开始位('1'的位数也包括了停止位的位数)。

断开符号 被视为在一个帧周期内全部收到'0'(包括停止位期间，也是'0')。在断开帧结束时，发送器再插入1或2个停止位('1')来应答起始位。

发送和接收由一共用的波特率发生器驱动，当发送器和接收器的使能位分别置位时，分别为其产生时钟。

随后将有每个功能块的详细说明。

图237 字长设置



24.3.2 发送器

发送器根据M位的状态发送8位或9位的数据字。当发送使能位(TE)被设置时，发送移位寄存器中的数据在TX脚上输出，相应的时钟脉冲在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236)。

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

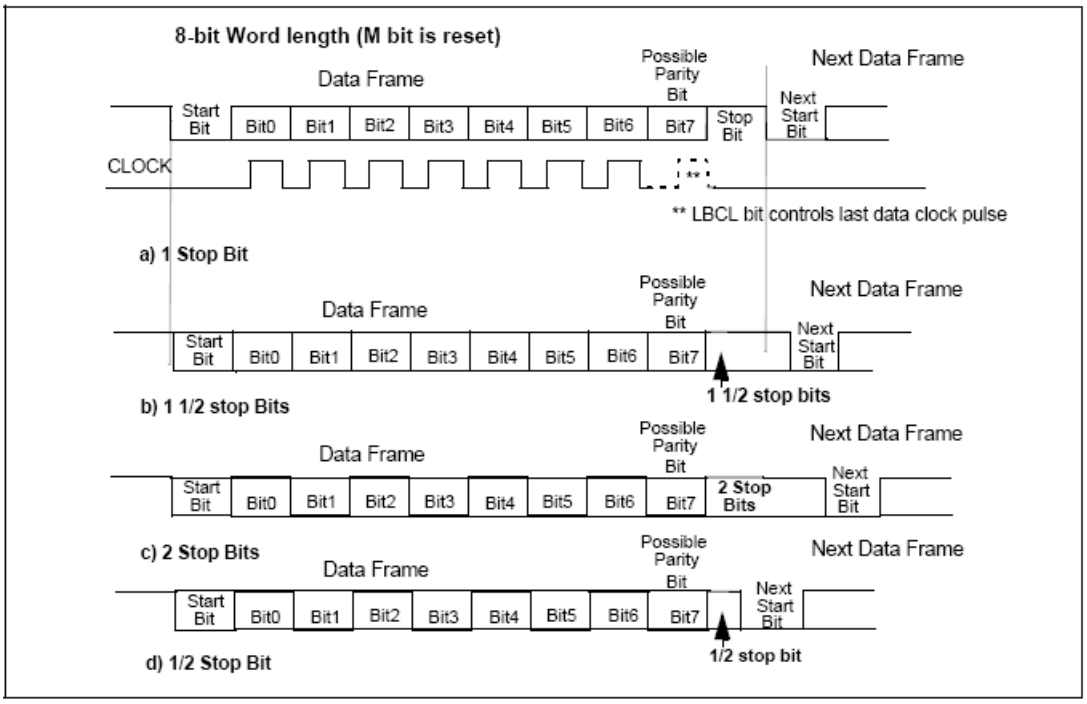
随每个字符发送的停止位的位数可以通过控制寄存器2的位13、12进行编程。

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238 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

设置SBK可发送一个断开符号。断开帧长度取决M位(见0)。如果设置SBK=1，在完成当前数据发送后，将在TX线上发送一个断开符号。断开字符发送完成时(在断开符号的停止位时)SBK被硬件复位。USART在最后一个断开帧的结束处插入一逻辑'1'，以保证能识别下一帧的起始位。

注意：如果在开始发送断开帧之前，软件又复位了SBK位，断开符号将不被发送。如果要发送两个连续的断开帧，SBK位应该在前一个断开符号的停止位之后置起。

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。

24.3.3 接收器

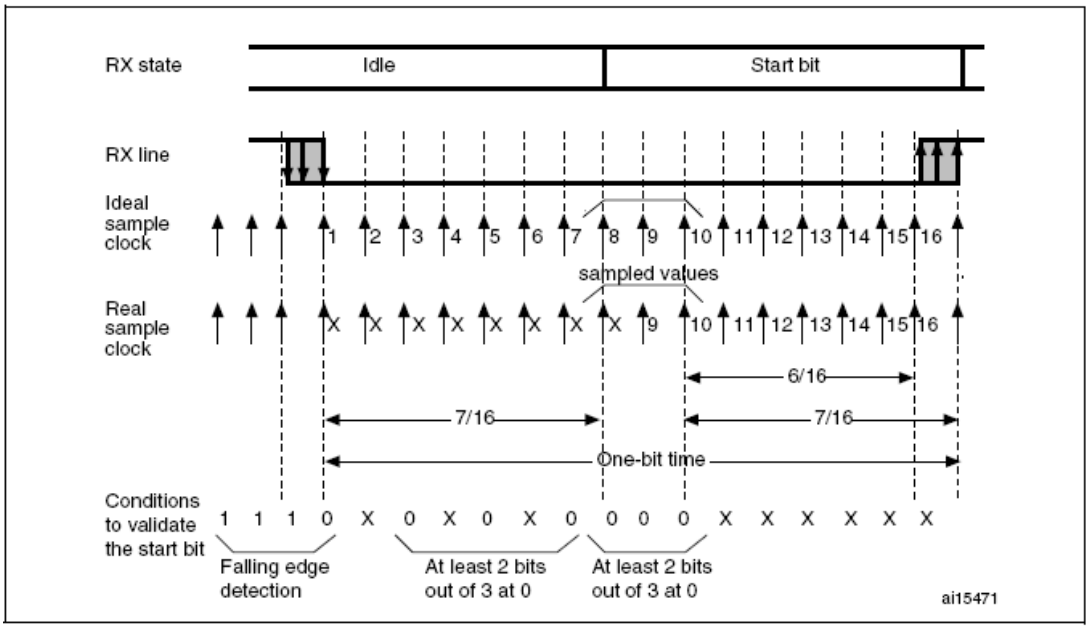
USART可以根据USART_CR1的M位接收8位或9位的数据字。

起始位侦测

在USART中，如果辨认出一个特殊的采样序列，那么就认为侦测到一个起始位。

该序列为：1110X0X0X0X0X0X0

图239 起始位侦测



注意： *如果该序列不完整，那么接收端将退出起始位侦测并回到空闲状态(不设置标志位)等待下降沿。*

如果3个采样点上仅有2个是零(第3、第5和第7个采样点或者第8、第9和第10个采样点)，那么起始位仍然是有效的，但是会设置NE噪声标志位。

如果最后三个(第8、第9和第10)采样点为0，那么起始位将被确认。

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(DMAR)。按多缓冲器通信所要求的配置DMA寄存器。

5. 利用波特率寄存器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240 检测噪声的数据采样

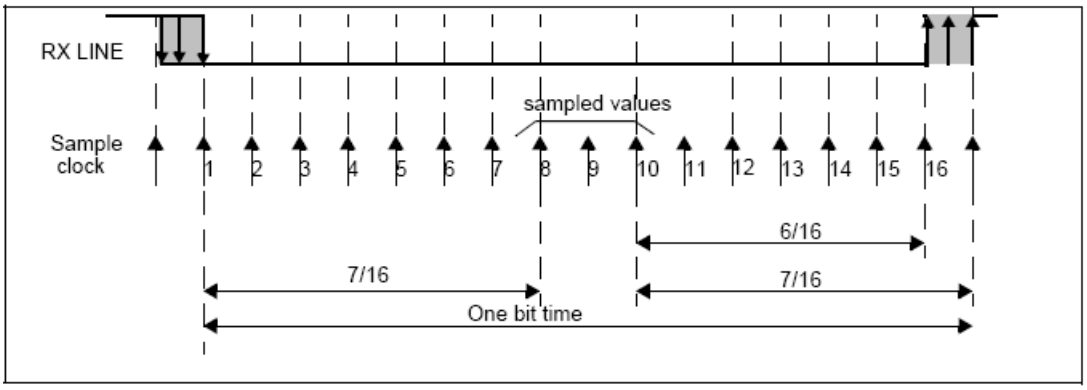


表153 检测噪声的数据采样

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。
- 在单字节通信时，没有中断产生。然而，这个位和RXNE位同时置起，后者将产生中断。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24.3.10 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。

24.3.4 分数波特率的产生

接收器和发送器的波特率在USARTDIV的整数和小数寄存器中的值应设置成相同。

$$\text{Tx / Rx 波特率} = \frac{f_{PCLKx}}{(16 * USARTDIV)}$$

这里的f_{PCLKx}(x=1、2)是给外设的时钟(PCLK1用于USART2、3、4、5，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154 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)。

24.3.5 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入。

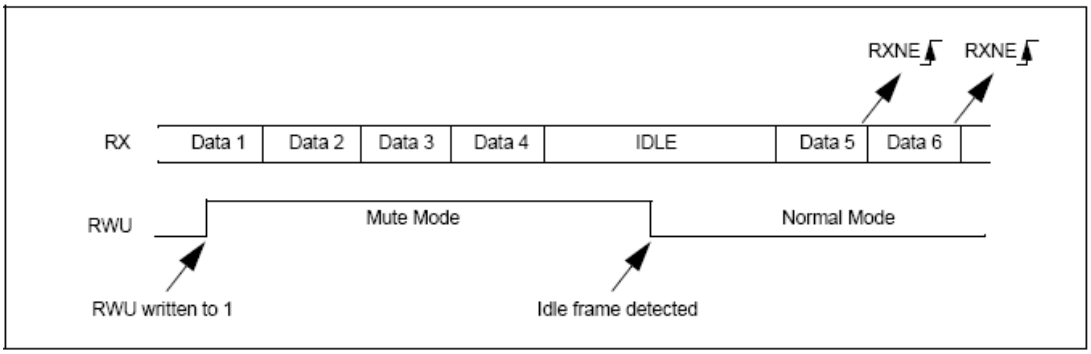
根据USART_CR1寄存器中的WAKE位状态，USART可以用二种方法进入或退出静默模式。

- 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241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241 利用空闲总线检测的静默模式



地址标记(address mark)检测(WAKE=1)

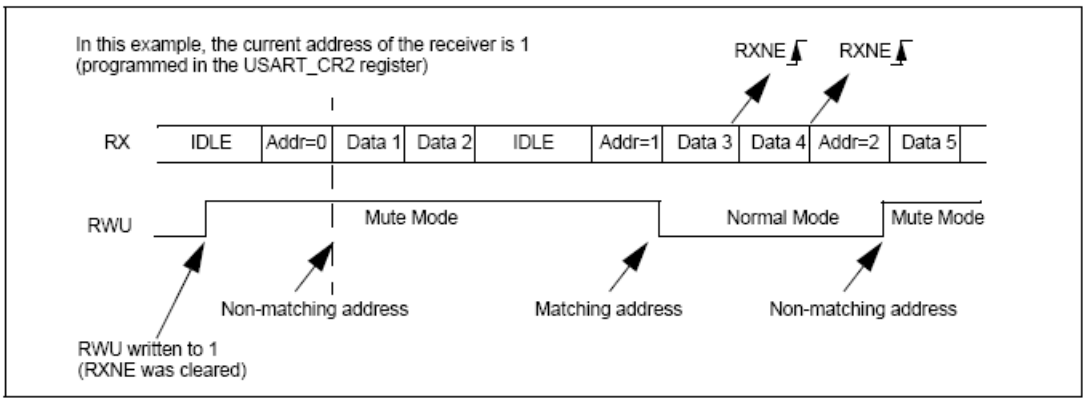
在这个模式里，如果MSB是1，该字节被认为是地址，否则被认为是数据。在一个地址字节中，目标接收器的地址被放在4个LSB中。这个4位地址被接收器同它自己地址做比较，接收器的地址被编程在USART_CR2寄存器的ADD。

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242给出利用地址标记检测来唤醒和进入静默模式的例子。

图242 利用地址标记检测的静默模式



24.3.6 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155中。

表155 帧格式

M位	PCE位	USART帧
0	0	| 起始位 | 8位数据 | 停止位 |
0	1	| 起始位 | 7位数据 | 奇偶检验位 | 停止位 |
1	0	| 起始位 | 9位数据 | 停止位 |
1	1	| 起始位 | 8位数据 | 奇偶检验位 | 停止位 |

注意：在用地址标记唤醒设备时，地址的匹配只考虑到数据的MSB位，而不用关心校验位。（MSB是数据位中最后发出的，后面紧跟校验位或者停止位）

偶校验：校验位使得一帧中的7或8个LSB数据以及校验位中'1'的个数为偶数。

例如：数据=00110101，有4个'1'，如果选择偶校验(在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：数据=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。

24.3.7 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。在LIN模式下，下列位必须保持为0：

- USART_CR2寄存器的CLKEN位
- USART_CR3寄存器的STOP[1:0]，SCEN，HDSEL和IREN

LIN发送

24.3.2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其间，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模式没有被激活(LINEN=0)，接收器仍然正常工作于USART模式，不会进行断开检测。

如果LIN模式被激活(LINEN=1)，只要一发生帧错误(也就是停止位检测到'0'，这种情况出现在断开帧)，接收器就停止，直到断开符号检测电路接收到一个'1'(这种情况发生于断开符号没有完整的发出来)，或一个定界符(这种情况发生于已经检测到一个完整的断开符号)。

图243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244给了一个断开帧的例子。

图243 LIN模式下的断开检测(11位断开长度 – 设置了LBDL位)

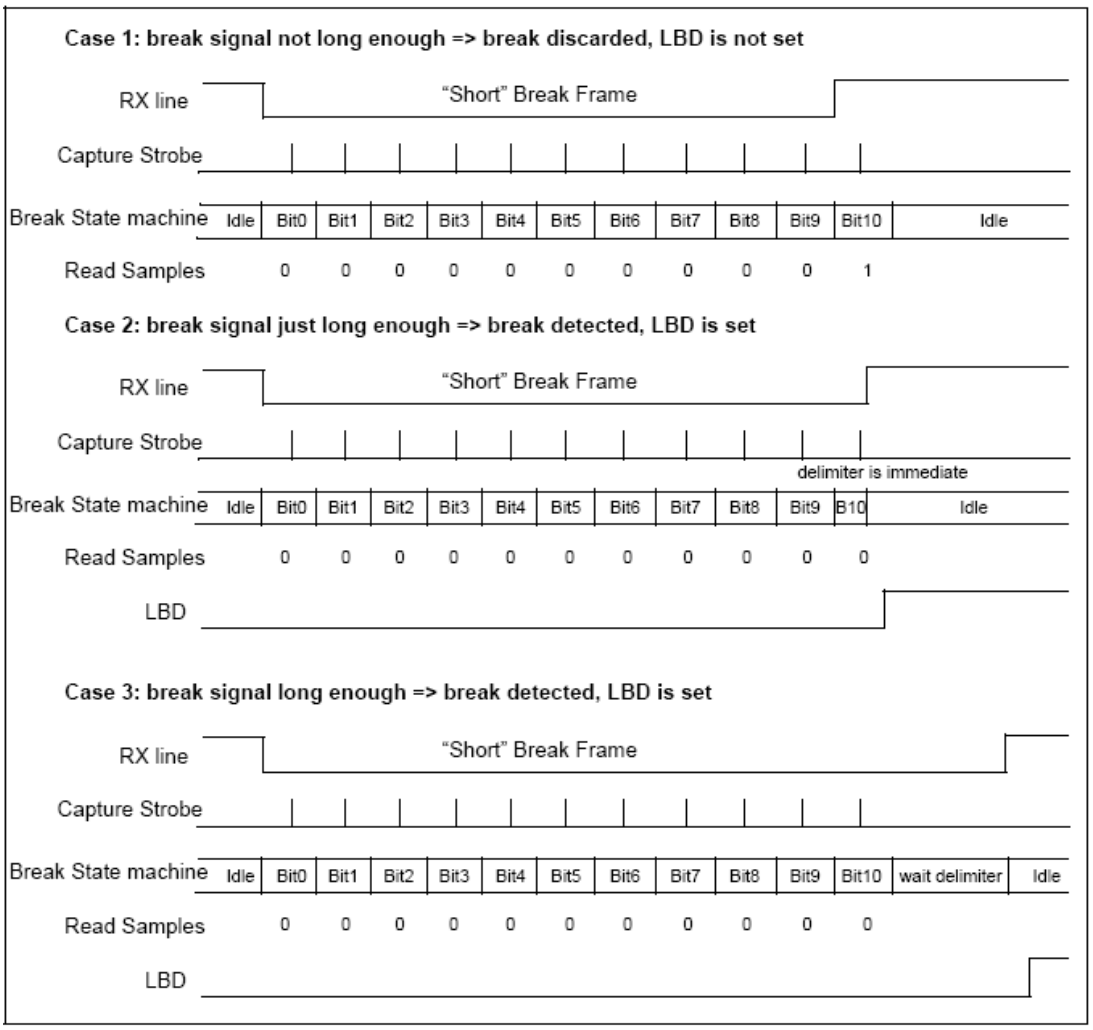
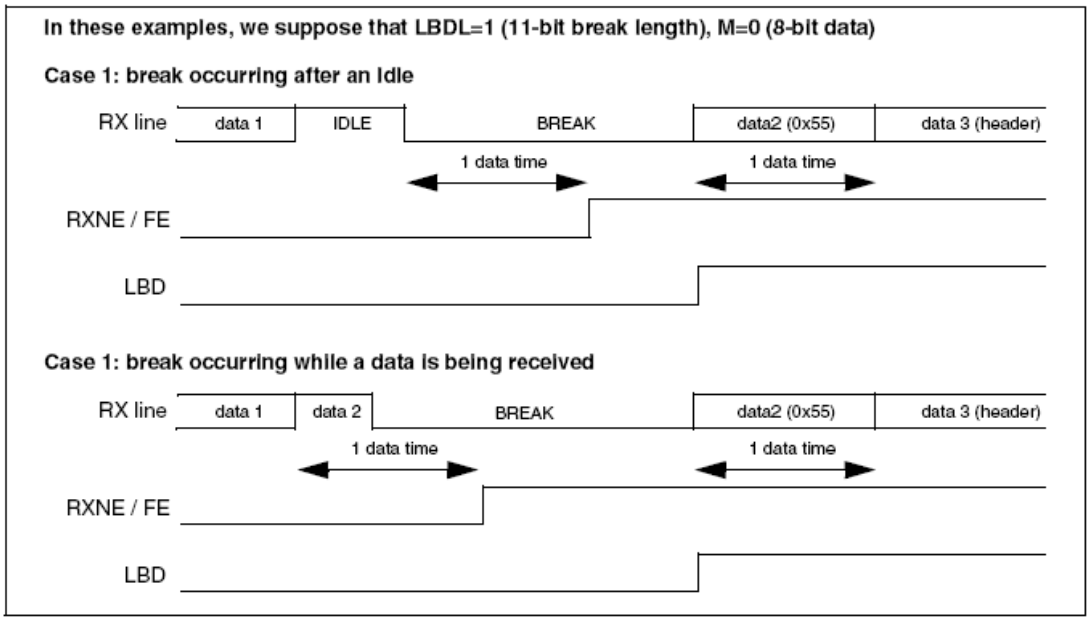


图244 LIN模式下的断开检测与帧错误的检测



24.3.8 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245、图246和图247)。

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245 USART同步传输的例子

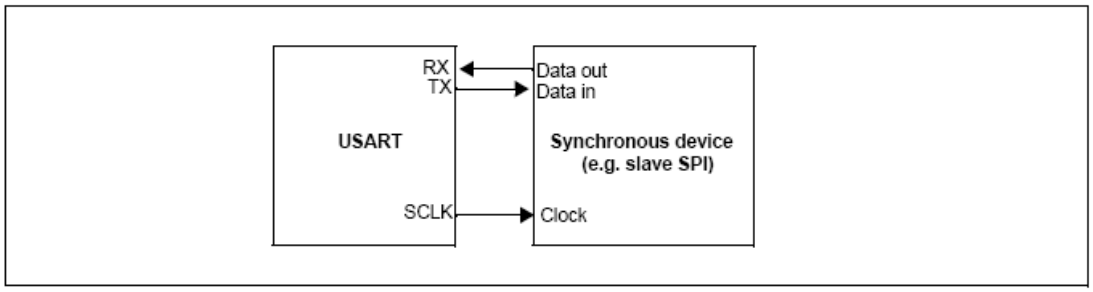


图246 USART数据时钟时序示例(M=0)

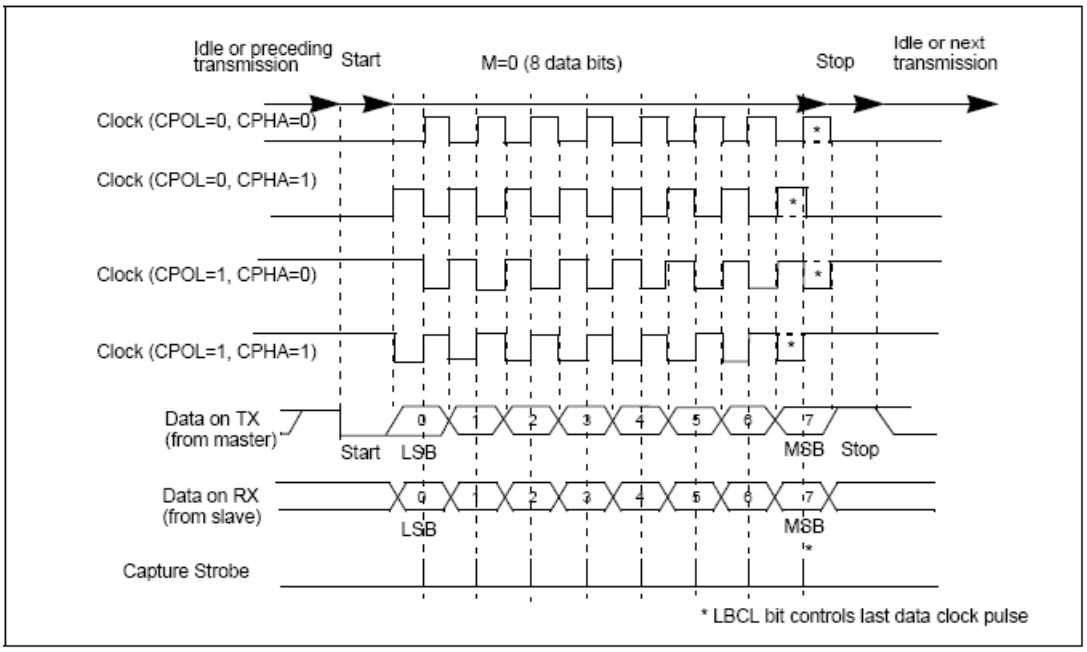


图247 USART数据时钟时序示例(M=1)

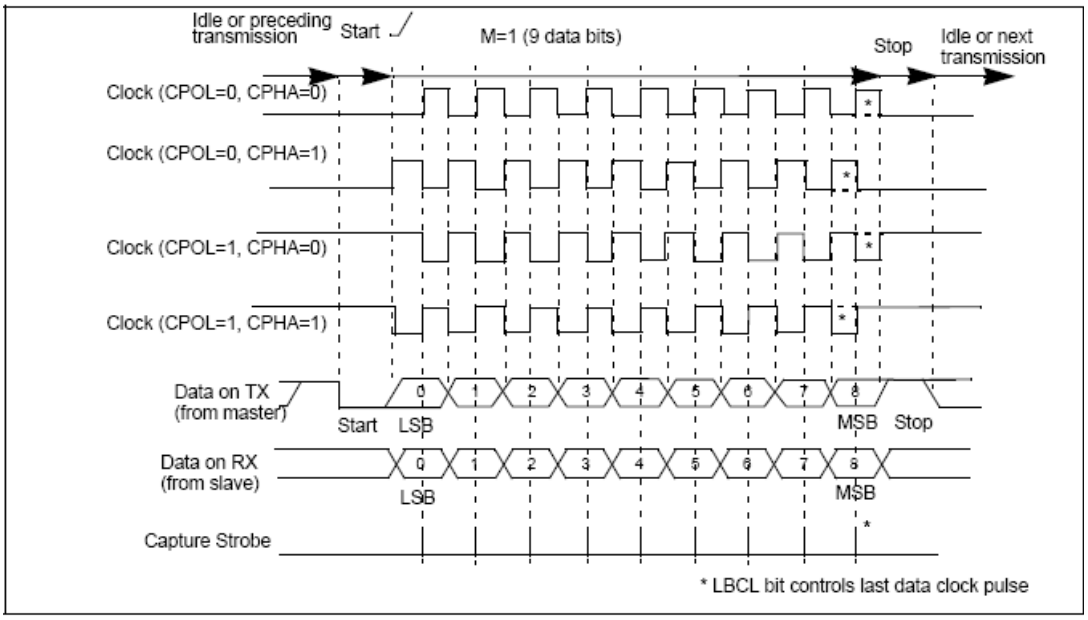
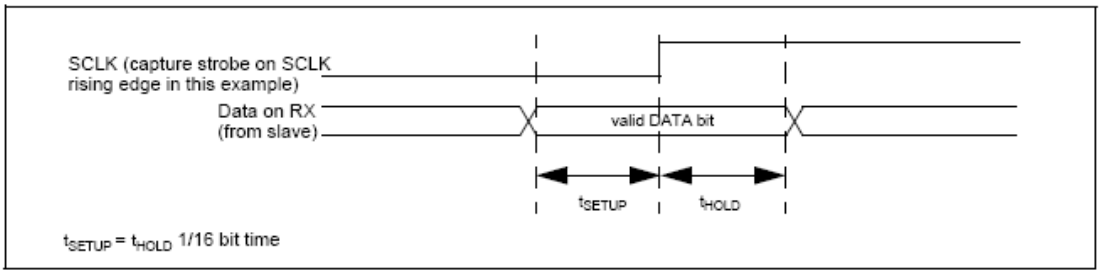


图248 RX数据采样/保持时间



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。

24.3.9 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。使用控制位“HALF DUPLEX SEL”选择半双工和全双工通信。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。

24.3.10 智能卡

设置USART_CR3寄存器的SCEN位选择智能卡模式。在智能卡模式下，下列位必须保持清零：

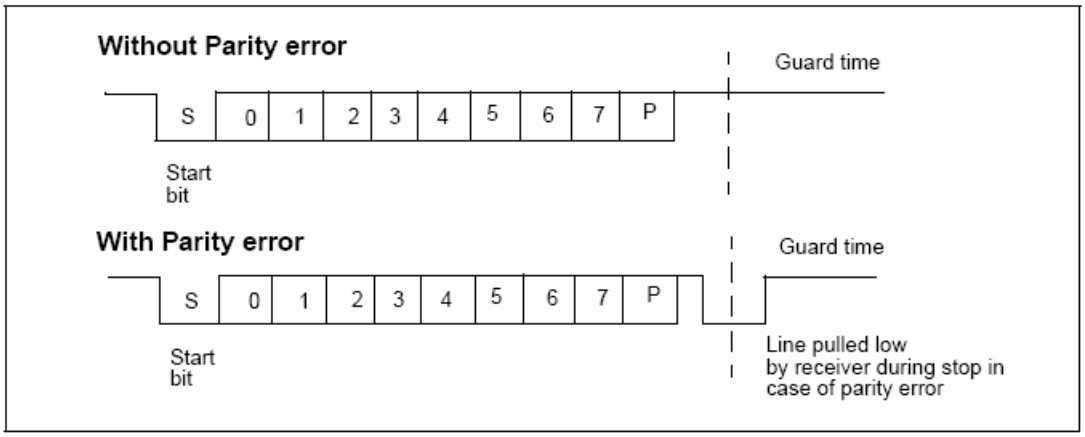
- USART_CR2寄存器的LINEN位
- USART_CR3寄存器的HDSEL 位和IREN位

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249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249 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。

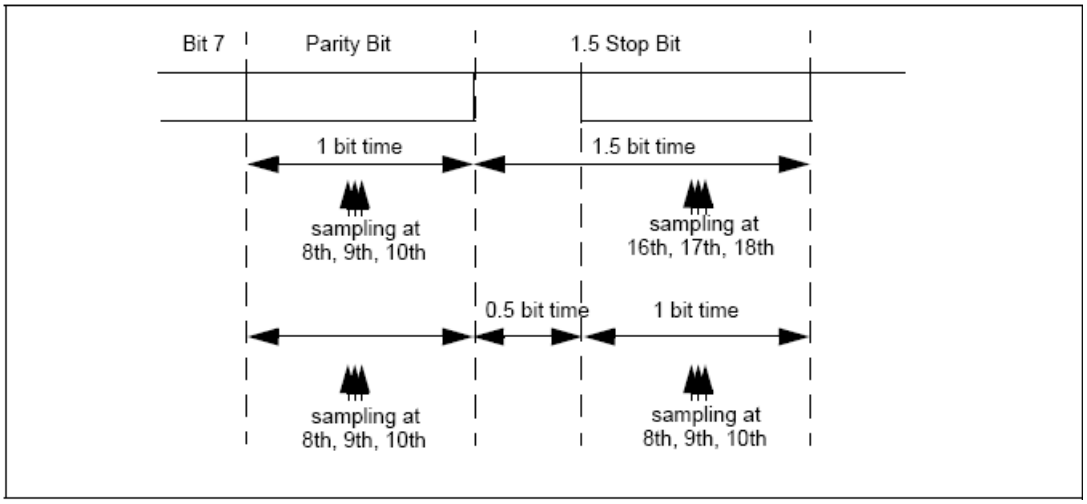
- 如果在接收数据帧期间，检测到一校验错误，该帧接收完成后(也就是在0.5停止位结束时)，发送线被拉低一个波特时钟周期。这是告诉智能卡发送到USART的数据没有被正确接收到。此NACK信号(拉低发送线一个波特时钟周期)在发送端将产生一个帧错误(发送端被配置成1.5个停止位)。应用程序可以根据协议处理重新发送的数据。如果NACK控制位被设置，发生校验错误时接收器会给出一个NACK信号；否则就不会发送NACK。
- TC标志的置起可以通过编程保护时间寄存器得以延时。在正常操作时，当发送移位寄存器变空并且没有新的发送请求出现时，TC被置起。在智能卡模式里，空的发送移位寄存器将触发保护时间计数器开始向上计数，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。
- 如果发送器检测到一个帧错误(收到接收器的NACK信号)，发送器的接收功能模块不会把NACK当作起始位检测。根据ISO协议，接收到的NACK的持续时间可以是1或2波特时钟周期。
- 在接收器这边，如果一个校验错误被检测到，并且NACK被发送，接收器不会把NACK检测成起始位。

注意： *1. 断开符号在智能卡模式里没有意义。一个带帧错误的00h数据将被当成数据而不是断开符号。*

2. 当来回切换TE位时，没有IDLE帧被发送。ISO协议没有定义IDLE帧。

图250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250 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f_{CK}/2到f_{CK}/62，这里的f_{CK}是外设输入时钟。

24.3.11 IrDA SIR ENDEC 功能块

通过设置USART_CR3寄存器的IREN位选择IrDA模式。在IRDA模式里，下列位必须保持清零：

- 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251)。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252)。
- 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1个PSC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251 IrDA SIR ENDEC – 框图

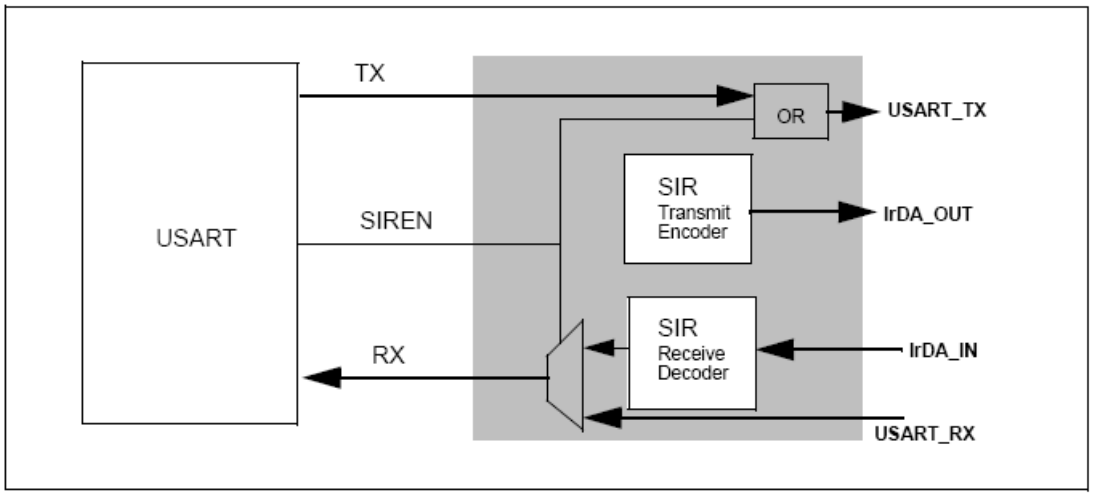
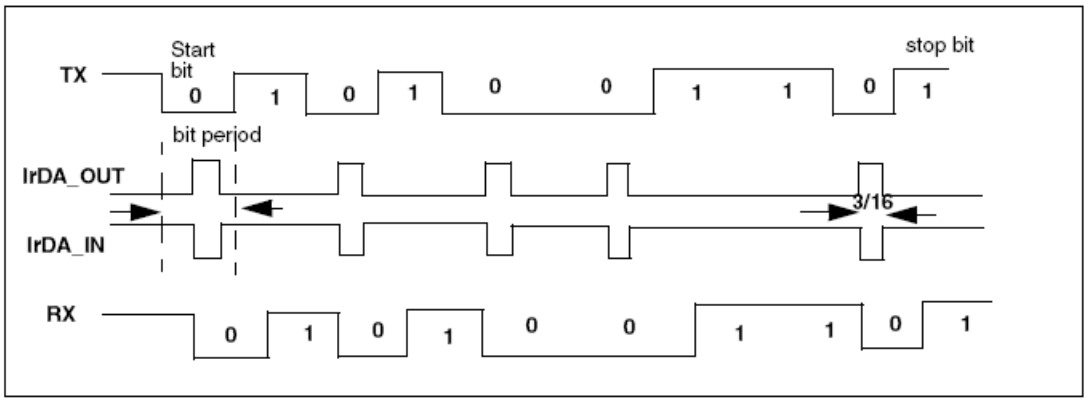


图252 IrDA数据调制(3/16) – 普通模式



24.3.12 利用DMA连续通信

USART可以利用DMA连续通信。Rx缓冲器和Tx缓冲器的DMA请求是分别产生的。

注意：参考产品技术说明以确定是否可用DMA控制器。如果所用产品无DMA功能，应按24.3.2节或24.3.3节里所描述的方法使用USART。在USART2_SR寄存器里，可以清零TXE/RXNE标志来实现连续通信。

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。
7. 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。

7. 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。

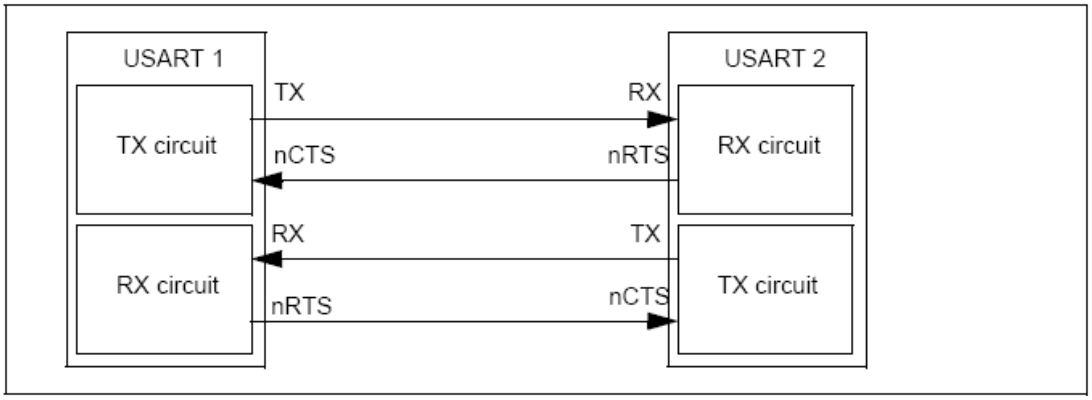
多缓冲器通信中的错误标志和中断产生

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。

24.3.13 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253表明在这个模式里如何连接2个设备。

图253 两个USART间的硬件流控制

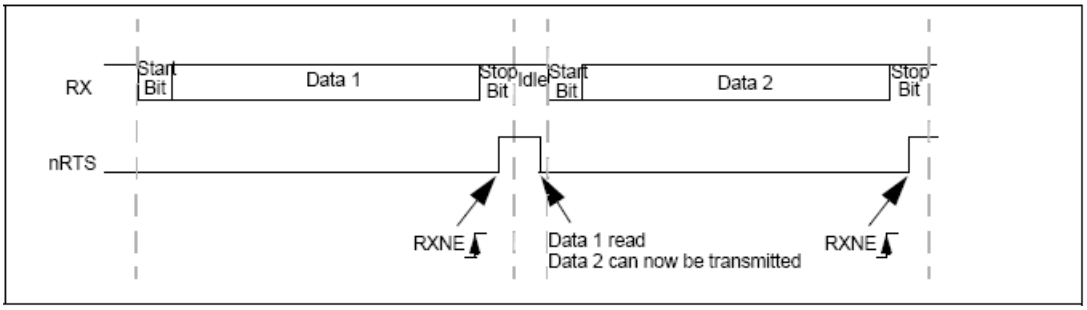


通过将UASRT_CR3中的RTSE和CTSE置位，可以分别独立地使能RTS和CTS流控制。

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254是一个启用RTS流控制的通信的例子。

图254 RTS流控制

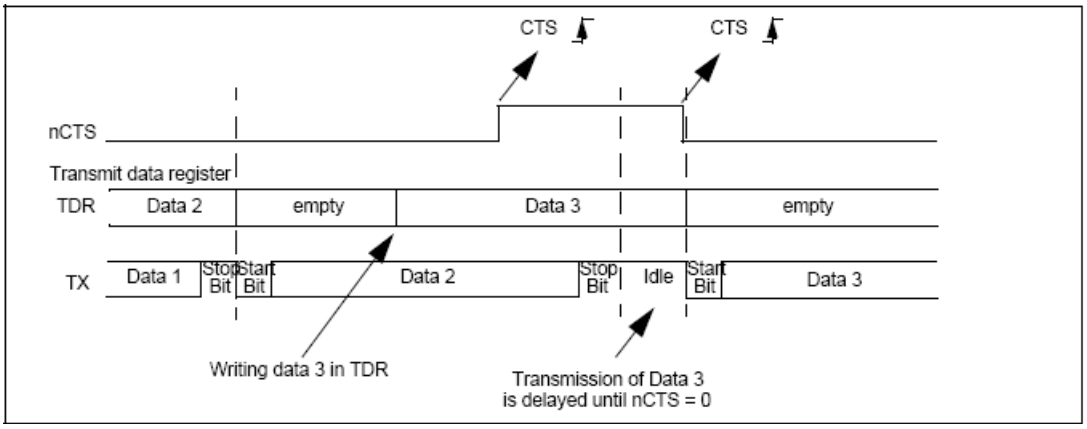


CTS流控制

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。下图是一个CTS流控制被启用的通信的例子。

图255 CTS流控制



24.4 USART中断请求

表156 USART中断请求

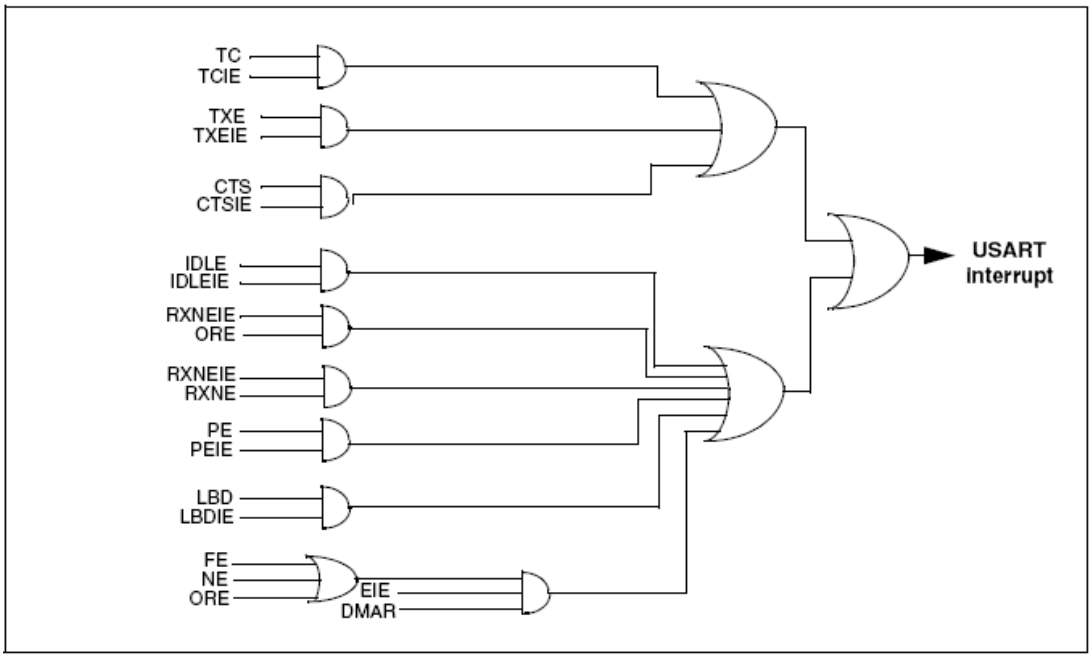
中断事件	事件标志	使能位
发送数据寄存器空	TXE	TXEIE
CTS标志	CTS	CTSIE
发送完成	TC	TCIE
接收数据就绪可读	TXNE	TXNEIE
检测到数据溢出	ORE	
检测到空闲线路	IDLE	IDLEIE
奇偶检验错	PE	PEIE
断开标志	LBD	LBDIE
噪声标志，多缓冲通信中的溢出错误和帧错误	NE或ORT或FE	EIE

USART的各种中断事件被连接到同一个中断向量(见图256)，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256 USART中断映像图



24.5 USART模式配置

表157 USART模式设置(1)

USART模式	USART1	USART2	USART3	UART4	UART5
异步模式	X	X	X	X	X
硬件流控制	X	X	X	NA	NA
多缓存通讯(DMA)	X	X	X	X	X
多处理器通讯	X	X	X	X	X
同步	X	X	X	NA	NA
智能卡	X	X	X	NA	NA
半双工(单线模式)	X	X	X	X	X
IrDA	X	X	X	X	X
LIN	X	X	X	X	X

（1）X = 支持，NA = 不支持该应用

24.6 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。

24.6.1 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_w0	rc_w0	r	rc_w0	rc_w0	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'1'，则产生中断。 0：nCTS状态线上没有变化； 1：nCTS状态线上发生变化。 注：UART4和UART5上不存在这一位。
位8	**LBD**: LIN断开检测标志（状态标志） 当探测到LIN断开时，该位由硬件置1，由软件清0(向该位写0)。如果USART_CR3中的LBDIE = 1，则产生中断。 0：没有检测到LIN断开； 1：检测到LIN断开。 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，则产生中断。由软件序列清除该位(先读USART_SR，然后写入USART_DR)。TC位也可以通过写入0来清除，只有在多缓存通讯中才推荐这种清除程序。 0：发送还未完成； 1：发送完成成。
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，则产生中断。对USART_DR的读操作可以将该位清零。RXNE位也可以通过写入0来清除，只有在多缓存通讯中才推荐这种清除程序。 0：数据没有收到； 1：收到数据，可以读出。

位4	**IDLE**：监测到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，则产生中断。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，则产生中断。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）。在清除PE位前，软件必须等待RXNE标志位被置1。如果USART_CR1中的PEIE为1，则产生中断。 0：没有校验错误； 1：校验错误。

24.6.2 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236)。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。

24.6.3 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。

24.6.4 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
res		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'0'。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=1)，在RXNE位被置位时，不能用软件修改RWU位。
位0	**SBK**: 发送断开帧 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。

24.6.5 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同步断开符(13位的低)，以及检测LIN同步断开符。
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； 注：UART4和UART5不能用0.5停止位和1.5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。 注：UART4和UART5上不存在这一位。
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。 注：UART4和UART5上不存在这一位。

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246和图247)。 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。 注：UART4和UART5上不存在这一位。
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)。 注：UART4和UART5上不存在这一位。
位7	保留位，硬件强制为0
位6	**LBDIE**: LIN断开符检测中断使能 断开符中断屏蔽(使用断开定界符来检测断开符) 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断开符检测长度 该位用来选择是11位还是10位的断开符检测 0：10位的断开符检测； 1：11位的断开符检测。
位4	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： 在使能发送后不能写这三个位(CPOL、CPHA、LBCL)。

24.6.6 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。 注：UART4和UART5上不存在这一位。

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为无效时，往数据寄存器里写数据，则这个数据要等到nCTS有效时才会被发送。 注：UART4和UART5上不存在这一位。
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)。 注：UART4和UART5上不存在这一位。
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。 注：UART4和UART5上不存在这一位。
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。 注：UART4和UART5上不存在这一位。
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。 注：UART4和UART5上不存在这一位。
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。 注：UART4和UART5上不存在这一位。
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

24.6.7 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。 注：UART4和UART5上不存在这一位。
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。

24.6.8 USART寄存器地址映象

表158 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，参见表1。

25 器件电子签名

小容量产品是指闪存存储器容量在16K至32K字节之间的STM32F101xx、STM32F102xx和STM32F103xx微控制器。

中容量产品是指闪存存储器容量在64K至128K字节之间的STM32F101xx、STM32F102xx和STM32F103xx微控制器。

大容量产品是指闪存存储器容量在256K至512K字节之间的STM32F101xx和STM32F103xx微控制器。

除非特别说明，本章节描述的模块应用于整个STM32F10xxx微控制器系列。

电子签名存放在闪存存储器模块的系统存储区域，可以通过JTAG/SWD或者CPU读取。它所包含的芯片识别信息在出厂时编写，用户固件或者外部设备可以读取电子签名，用以自动匹配不同配置的SRM32F10xxx微控制器。

25.1 存储器容量寄存器

25.1.1 闪存容量寄存器

基地址：0x1FFF F7E0

只读，其值在出厂时编写

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

位15:0	**F_SIZE**：闪存存储器容量 以K字节为单位指示产品中闪存存储器容量。 例：0x0080 = 128 K字节

25.2 产品唯一身份标识寄存器(96位)

产品唯一的身份标识非常适合：

- 用来作为序列号(例如USB字符序列号或者其他的终端应用)
- 用来作为密码，在编写闪存时，将此唯一标识与软件加解密算法结合使用，提高代码在闪存存储器内的安全性。
- 用来激活带安全机制的自举过程

96位的产品唯一身份标识所提供的参考号码对任意一个STM32微控制器，在任何情况下都是唯一的。用户在何种情况下，都不能修改这个身份标识。

这个96位的产品唯一身份标识，按照用户不同的用法，可以以字节(8位)为单位读取，也可以以半字(16位)或者全字(32位)读取。

基地址：0x1FFF F7E8

地址偏移：0x00

只读，其值在出厂时编写

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

位15:0	**U_ID**[15:0]：唯一身份标志15:0位

地址偏移：0x02

只读，其值在出厂时编写

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

位15:0	**U_ID[31:16]**：唯一身份标志31:16位

地址偏移：0x04

只读，其值在出厂时编写

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
U_ID[63:48]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
U_ID[47:32]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

位15:0	**U_ID[63:32]**：唯一身份标志63:32位

地址偏移：0x08

只读，其值在出厂时编写

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
U_ID[95:80]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
U_ID[79:64]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

位15:0	**U_ID[95:64]**：唯一身份标志15:0位

26 调试支持(DBG)

小容量产品是指闪存存储器容量在16K至32K字节之间的STM32F101xx、STM32F102xx和STM32F103xx微控制器。

中容量产品是指闪存存储器容量在64K至128K字节之间的STM32F101xx、STM32F102xx和STM32F103xx微控制器。

大容量产品是指闪存存储器容量在256K至512K字节之间的STM32F101xx和STM32F103xx微控制器。

除非特别说明，本章节描述的模块应用于整个STM32F10xxx微控制器系列。

26.1 概况

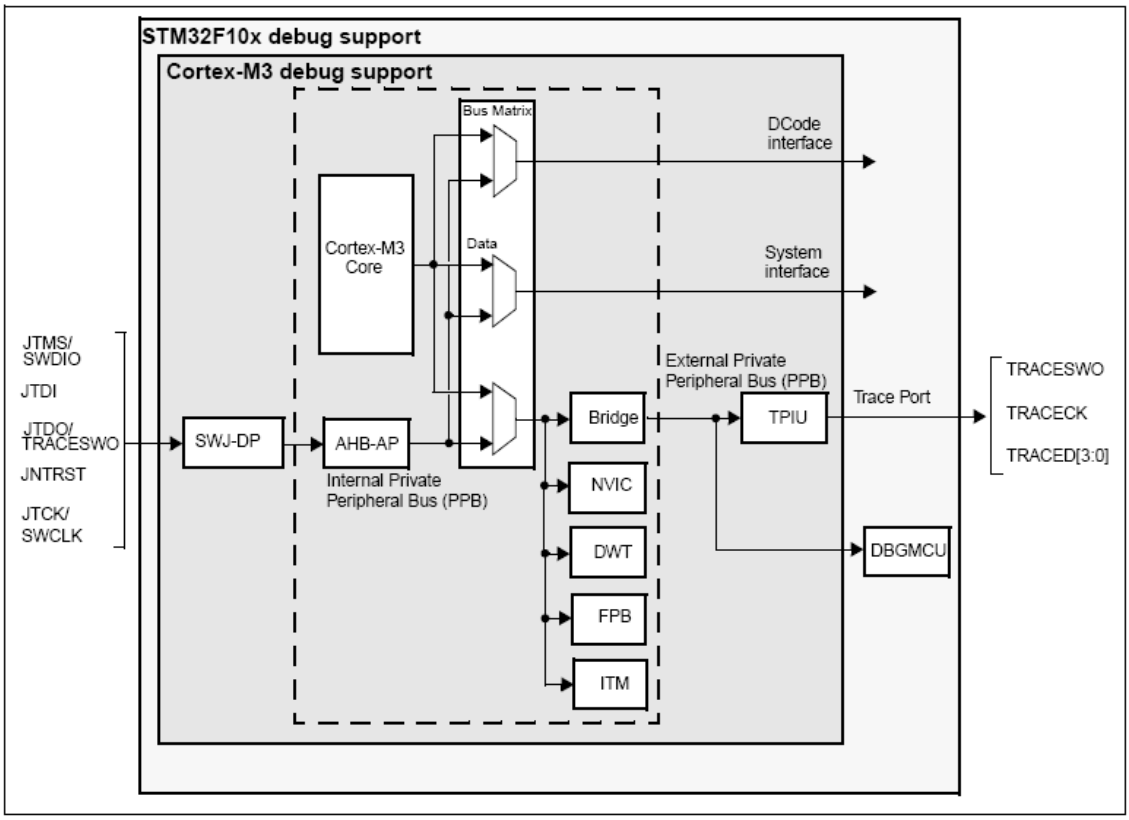
STM32F10xxx使用Cortex™-M3内核，该内核内含硬件调试模块，支持复杂的调试操作。硬件调试模块允许内核在取指（指令断点）或访问数据（数据断点）时停止。内核停止时，内核的内部状态和系统的外部状态都是可以查询的。完成查询后，内核和外设可以被复原，程序将继续执行。

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257 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：闪存指令断点
- 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。*

26.2 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

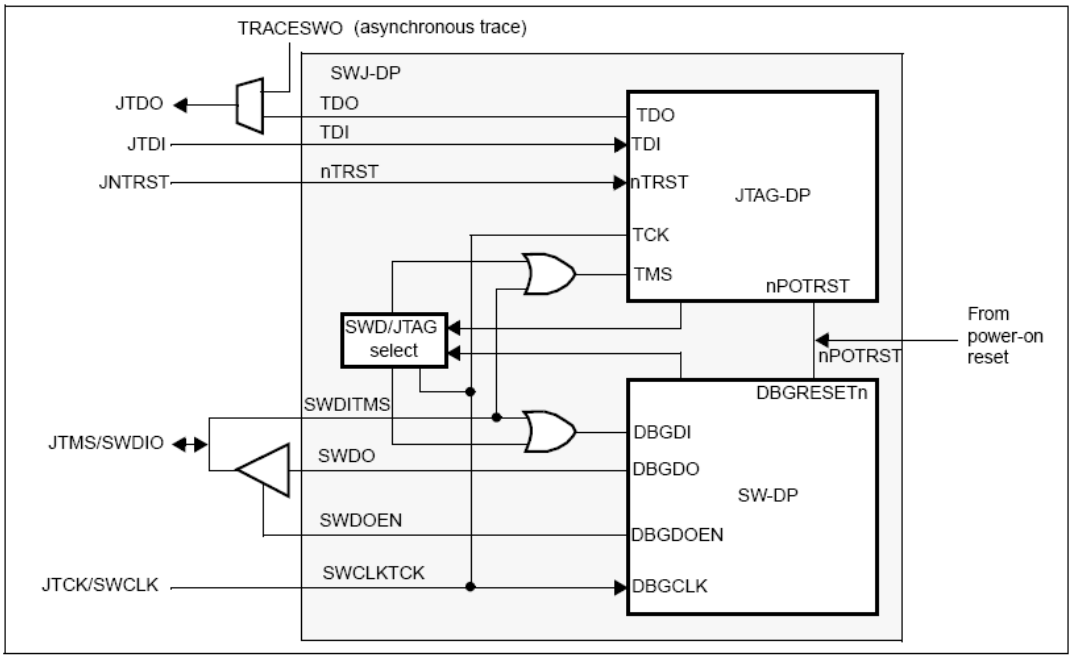
26.3 SWJ调试端口(serial wire and JTAG)

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258 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。

26.3.1 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

26.4 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。

26.4.1 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159 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

26.4.2 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(禁止同时设置一个以上的位)。

表160 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) 复位状态	专用	专用	专用	专用	专用
001	所有的SWJ引脚 (JTAG-DP + SW-DP) 除了JNTRST引脚	专用	专用	专用	专用	释放
010	JTAG-DP接口禁止，SW-DP接口允许	专用	专用	释放		
100	JTAG-DP接口和SW-DP接口都禁止	释放				
其他	禁止					

注意：当APB桥的写缓冲区满了时，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)。

26.4.3 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。

26.4.4 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。

26.5 STM32F10xxx JTAG TAP 连接

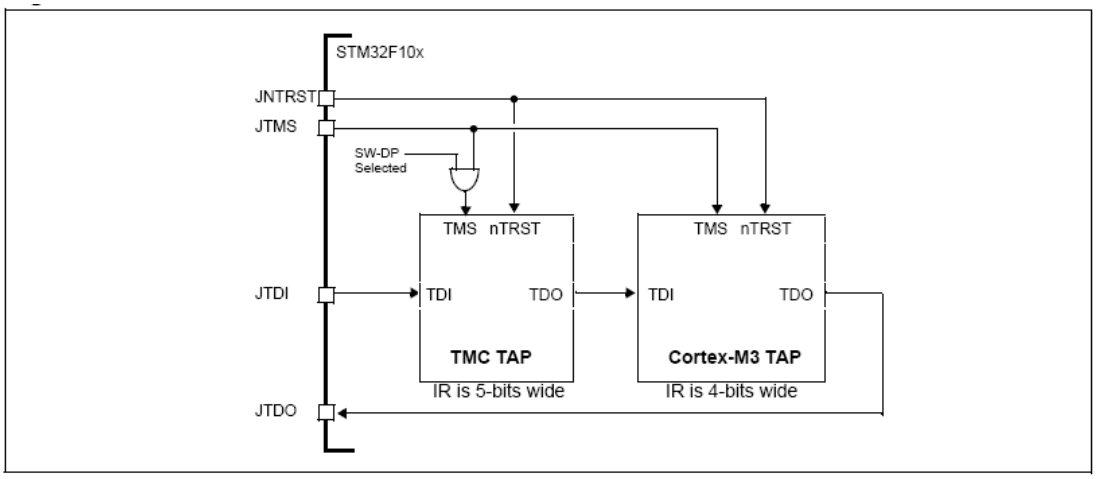
STM32F10xxx微控制器内部串联了两个JTAG TAP。边界扫描TAP专门用来进行测试(IR寄存器为5比特位宽)和Cortex™-M3 TAP(IR寄存器为有4比特位宽)。

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259 JTAG TAP连接



26.6 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。

26.6.1 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26.15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为内容不确定的位

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

位31:16	**REV_ID[15:0]**：版本识别 该域标识产品的版本 小容量产品：0x1000 = 版本A 中容量产品：0x0000 = 版本A；0x2000 = 版本B；0x2001 = 版本Z；0x2003 = 版本Y 大容量产品：0x1000 = 版本A；0x1001 = 版本Z 互联系列产品： — 0x1000 = 版本A
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： 小容量产品，设备编码为0x412； 中容量产品，设备编码为0x410； 大容量产品，设备编码为0x414； 互联系列产品，设备编码为0x418。

26.6.2 边界扫描TAP

JTAG ID编码

STM32F10xxx的边界扫描TAP集成了JTAG ID编码：

- 对于小容量产品
  - 0x06412041 = 版本 A
- 对于中容量产品
  - 0x06410041 = 版本 A
  - 0x16410041 = 版本 B 和版本 Z
- 对大容量产品
  - 0x06414041 = 版本 A

26.6.3 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(11:0)来识别芯片。

26.6.4 Cortex-M3 JEDEC-106 ID代码

ARM 的 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。

26.7 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(详见*Cortex-M3 r1p1 Technical Reference Manual*)实现的。

表161 JTAG调试端口数据寄存器

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 r1p1-01rel0 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162
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162 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)

26.8 SW调试端口

26.8.1 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。

26.8.2 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163 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162)
5	Parity	前面比特位的校验位
6	Stop	0
7	Park	不能由主机驱动，由于有上拉，目标永远读为1

有关DPACC和APACC寄存器描述的详细资料，请参考Cortex-M3 r1p1技术参考手册。

包请求后总是跟一个(缺省为1位)转换时间，此时主机和目标都不驱动线路。

表164 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165 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。

26.8.3 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技术参考手册和CoreSight Design Kit r1p0技术参考手册。

26.8.4 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。

26.8.5 SW-DP寄存器

当APnDP=0时，可以访问以下这些寄存器。

表166 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。

26.8.6 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

26.9 对于JTAG-DP或SWDP都有效的AHB-AP (AHB 访问端口)

功能：

- 系统访问是独立于处理器状态的。
- JTAG-DP和SW-DP都可以访问AHB-AP
- AHB-AP是总线矩阵的AHB主设备。因此，它可以访问所有的数据总线(Dcode总线，System总线，内部和外部 PPB总线)，只有ICode总线除外。
- 支持位寻址的传输
- 旁路FPB的AHB-AP传输

32位AHP-AP寄存器的地址是6-位宽(最多64个字或256个字节)，由以下部分组成：

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167 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技术参考手册

26.10 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168 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技术参考手册。

为了在复位后立即使内核进入调试状态，需要：

- 使能调试和异常监视控制寄存器(Debug and Exception Monitor Control Register)的位0 (VC_CORRESET)。
- 使能调试控制和状态寄存器(Debug Halting Control and Status Register)的位0 (C_DEBUGEN)。

26.11 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)。*

26.12 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。

26.13 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

26.14 ITM (instrumentation trace macrocell)

26.14.1 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。

26.14.2 时间戳包，同步和溢出包

时间戳包包含了时间戳信息，普通的控制和同步信息。它使用一个21位的时间戳计数器(及可能的预分频器)，此计数器在每个时间戳包发放时复位。计数器的时钟可以是CPU时钟也可以是SWV时钟。

同步包为0x80_00_00_00_00_00，按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：必须设置DWT控制寄存器(DWT Control Register)的位0(CYCCNTENA)。此外，也必须设置ITM跟踪控制寄存器(Trace Control Register)的位2(SYNCENA)。

注意： *如果SYNENA位没有被置起，DWT产生给TPIU的同步触发，将只发送TPIU同步包而不发送ITM同步包。*

溢出包是一个特殊的时间戳包，该包指示数据已经被写但是FIFO已满。

表169 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26.16.2节-跟踪引脚分配，26.15.3节-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)。

26.15 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

26.15.1 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。

26.15.2 支持定时器、看门狗、bxCAN和I^2C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。

26.15.3 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：0x0000 0000(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_TIM8_STOP	DBG_TIM7_STOPT	DBG_TIM6_STOPT	DBG_TIM5_STOP	DBG_I2C2_SMBUS TIMEOUT
res											rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DBG_I2C1_SMBUS TIMEOUT	DBG_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	res		rw	rw	rw

位31:21	保留，必须保持为0。
位20:17	**DBG_TIMx_STOP**：当核心停止时停止定时器计数器(x=8..5) 0: 当核心停止时，仍然向相关定时器的计数器提供时钟，定时器输出工作正常； 1: 当核心停止时，切断相关定时器的计数器的时钟，同时关闭定时器的输出(就好象对某一暂停事件的紧急响应，停止定时器)。
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)整个数字电路部分都断电。 从软件的观点看，退出STANDBY模式与复位是一样的(除了一些状态位指示了微控制器刚从STANDBY状态退出)。 1：(FCLK开，HCLK开)数字电路部分不下电，FCLK和HCLK时钟由内部RL振荡器提供时钟。另外，微控制器通过产生系统复位来退出STANDBY模式和复位是一样的。
位1	**DBG_STOP**：调试停止模式。 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，晶振等。 1：(FCLK开，HCLK开)在停止模式时，FCLK和HCLK时钟由内部RC振荡器提供。当退出停止模式时，软件必需重新配置时钟系统启动PLL，晶振等(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时钟都由原先配置好的系统时钟提供。

26.16 TPIU (trace port interface unit)

26.16.1 导言

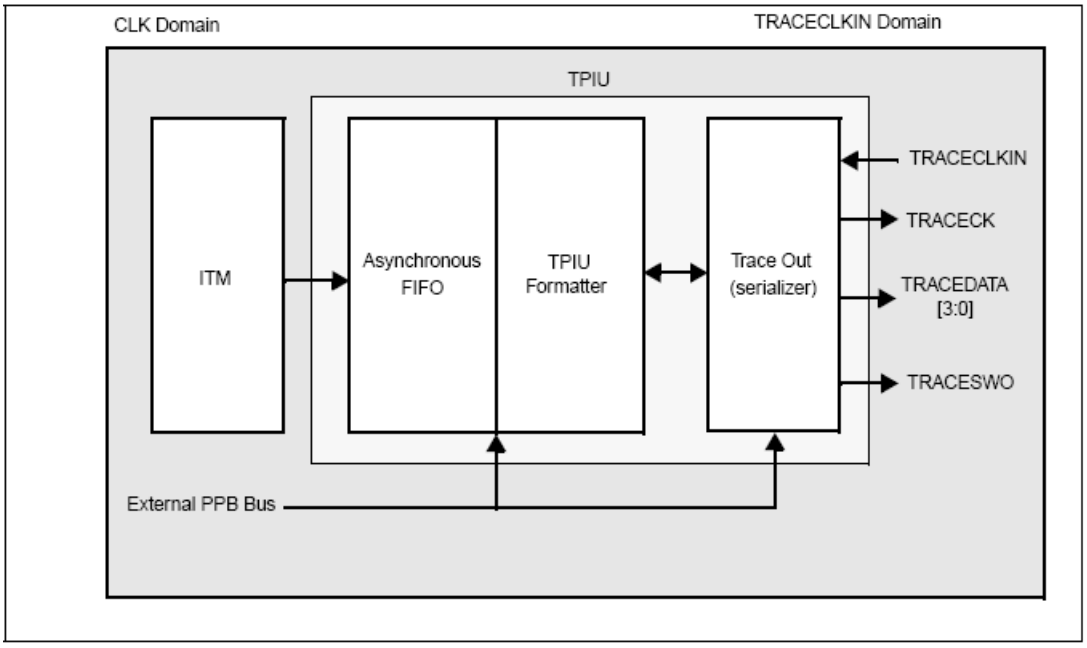
TPIU在片上追踪数据和ITM之间担当桥梁的作用。

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260 TPIU框图



26.16.2 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170 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171 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172 灵活的跟踪管脚分配

<table>
<tr><th colspan="2">DBGMCU_CR寄存器</th><th rowspan="2">引脚用途</th><th colspan="6">跟踪引脚分配</th></tr>
<tr><th>TRACE_ IOEN</th><th>TRACE_ MODE[1:0]</th><th>PB3/ JTDO/ TRACES WO</th><th>PE2/ TRACE CK</th><th>PE3/ TRACE D[0]</th><th>PE4/ TRACE D[1]</th><th>PE5/ TRACE D[2]</th><th>PE6/ TRACE D[3]</th></tr>
<tr><td>0</td><td>XX</td><td>无跟踪（默认状态）</td><td>释放(1)</td><td colspan="5" rowspan="2">释放（可用作普通I/O口）</td></tr>
<tr><td>1</td><td>00</td><td>异步跟踪</td><td>TRACES WO</td></tr>
<tr><td>1</td><td>01</td><td>同步跟踪1位</td><td rowspan="3">释放(1)</td><td>TRACE CK</td><td>TRACE D[0]</td><td colspan="3">释放
(可用作普通I/O口)</td></tr>
<tr><td>1</td><td>10</td><td>同步跟踪2位</td><td>TRACE CK</td><td>TRACE D[0]</td><td>TRACE D[1]</td><td colspan="2">释放(可用作普通I/O口)</td></tr>
<tr><td>1</td><td>11</td><td>同步跟踪4位</td><td>TRACE CK</td><td>TRACE D[0]</td><td>TRACE D[1]</td><td>TRACE D[2]</td><td>TRACE D[3]</td></tr>
<tr><td colspan="9">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。</td></tr>
</table>

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

26.16.3 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。

26.16.4 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。

26.16.5 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=0，只发送字 0x7F_FF_FF_FF，后面不跟任何格式的数据包。
  - 如果 ITM 的 SYNENA 位=1，ITM 同步包将跟在 (0x80_00_00_00_00_00)后面，由 TPUI 编排格式(加上跟踪源 ID)。

26.16.6 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。

26.16.7 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%。

26.16.8 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。

26.16.9 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173 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

26.16.10 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。

26.17 DBG寄存器地址映象

下列表格归纳了调试寄存器。

表174 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
0xE0042000	DBGMCU_IDCODE	REV_ID																保留				DEV_ID											
	复位值(1)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x					x	x	x	x	x	x	x	x	x	x	x	x
0xE0042004	DBGMCU_CR	保留											DBG_TIM8_STOP	DBG_TIM7_STOP	DBG_TIM6_STOP	DBG_TIM5_STOP	DBG_I2C2_SMBUS_TIMEOU	DBG_I2C1_SMBUS_TIMEOU	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	0				0	0	0

(1) 复位值与特定的产品相关。详情参见26.6.1节-微控制器设备ID编码。