HK32L084CBT6/HK32L084RBT6
HK32L088CBT6/HK32L088RBT6
HK32L084CBT6H/HK32L084RBT6H
HK32L088CBT6H/HK32L088RBT6H

用户手册

Rev1.0.12

目录

History

Version	Date	Modify	Description
1.0.0	2020/5/10	Yan	初版
1.0.1	2020/5/10	Yan	补充及修改部分内容： 1. 修改 AES、ADC、DAC 章节内容； 2. 补充 DIVSQRT 章节内容； 3. 添加设备电子签名章节
1.0.2	2020/6/10	Yan	修改：更换时钟树图
1.0.3	2020/6/14	Jack	修改：Lpuart 485 DE 计算描述
1.0.4	2020/6/19	Thomas	修改：COMP 寄存器描述
1.0.5	2020/6/22	Yan	修改： 1. TIM1 寄存器描述 2. SYSCFG 寄存器描述 3.RCC 寄存器描述
1.0.6	2020/6/24	Wing	修改： 1. USART 寄存器描述 2. IIC 章节内容 3. LCD 寄存器描述
1.0.7	2020/6/28	Thomas	修改： 1. OPAMP 章节内容
1.0.8	2020/7/20	Thomas	修改： 1. COMP 章节内容 2. OPAMP 章节内容
1.0.9	2020/8/5	Felix	修改： 1.DIVSQRT 章节的”中断”部分描述
1.0.10	2020/8/10	Yan	修改： 1.删除不开放给客户的内容 2.修改 RCC 章节描述内容
1.0.11	2020/8/17	Vincent	修改： 1.修改 PWR 章节内容
1.0.12	2020/8/24	Yan	修改： 1.页眉修改 Logo，页脚修改格式

1 说明

本文档为 HK32L084CBT6/HK32L084RBT6/HK32L088CBT6/HK32L088RBT6、HK32L084CBT6H/HK32L084RBT6H/HK32L088CBT6H/HK32L088RBT6H 芯片用户手册。本参考手册为应用程序开发人员提供关于如何使用 HK32L08x 系列微控制器的内存和外设所涉及的全部信息。HK32L08x 系列芯片是深圳市航顺芯片技术研发有限公司开发的低功耗 MCU 芯片，请联系深圳市航顺芯片技术研发有限公司提供更多相关文档。

2 缩写与术语

2.1 寄存器描述中的缩写

缩写	对应的英文	缩写含义
rw	read/write	软件能读写这些位(或指定位)
r	read-only	软件只读这些位(或指定位)
w	write-only	软件只写这些位(或指定位)
rc_w1	read/clear	软件可读该位，可通过对该位写1 时清除该位。 对该位写0 时，该位值无变化。
rc_w0	read/clear	软件可读该位，可通过对该位写0 时清除该位。 对该位写1 时，该位值无变化。
rc_r	read/clear by read	软件可读该位，读后该位自动清为0。 对该位写0 时，该位值无变化。
rs	read/set	软件可读写该位。但其与rw 有区别，一般设置该位为1 时启动某种硬件动作，当完成硬件动作后该位会被硬件自动清0。
rt_w	read-only write trigger	软件可以读此位；对该位写0 或1 触发一个事件但对此位数值没有影响。
t	Toggle	软件只能对该位写1 来翻转此该位，写0 对该位无影响。
Res	Reserved	保留位，必须保持默认值不变

2.2 术语

- **SWD:** 为 Serial Wire Debug 的首字母缩写。其是 Cortex-M0 内核集成的一个调试口，是基于 SWD 协议的 2 线调试接口。
- **Word:** 字，32 位长的数据或指令长度。
- **Half word:** 半字，16 位长的数据或指令长度。
- **Byte:** 字节，8 位数据长度。
- **IAP:** in-application programming 的首字母缩写。直译为在应用编程，即用户程序可以更 新自身的程序，从而达到应用升级。
- **ICP:** in-circuit programming 的首字母缩写。直译为在电路编程，即用户可通过应用板上的 JTAG 口或 SWD 口对 MCU 的 FLASH 进行编程。
- **Option bytes:** 选项字节，保存在 Flash 中的 MCU 配置字节。
- **OBL:** option byte loader 的首字母缩写，选项字节装载器。
- **AHB:** advanced high-performance bus 的首字母缩写，直译为先进高性能总线。
- **NVM:** 非易失性存储器。
- **ECC:** 误码校正。
- **DMA:** 直接存储器访问。
- **MIF:** NVM 接口。

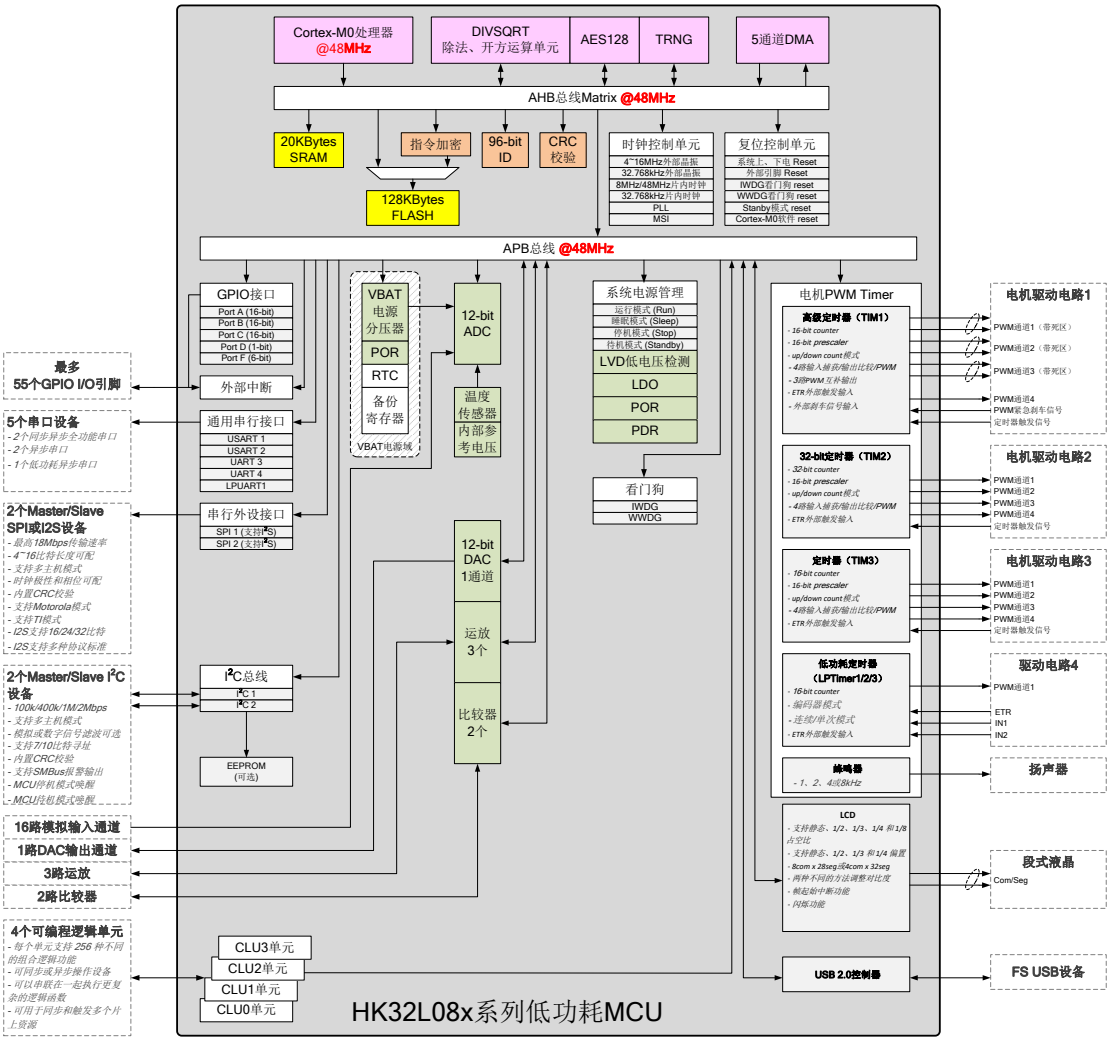
3 功能介绍

3.1 系统架构

系统主要由以下几个模块组成：

- 二个主模块 ：
  - Cortex-M0 内核及先进高性能总线
- 四个从模块：
  - 内部 SRAM
  - 内部闪存存储器
  - AHB 到 APB 的桥,所有的外设都挂在 APB 总线上

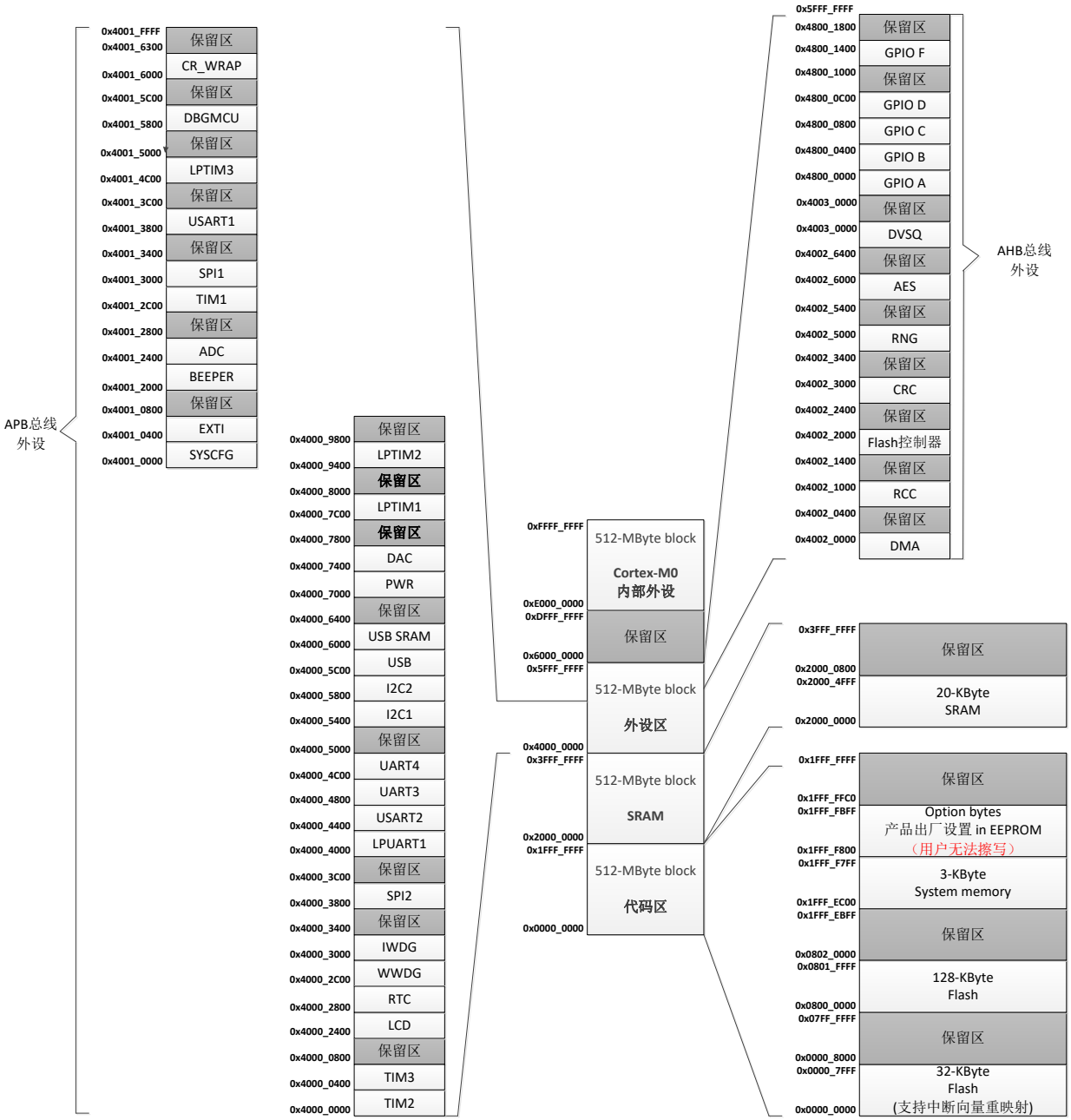
功能框图如下图：



3.2 存储器映射

程序存储器，数据存储器，寄存器及I/O 口统一编址，其线性地址空间达到4G。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。

寻址空间分成8 块，每块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。



3.3 SRAM

HK32L08x 内置有 20K 字节的静态 SRAM。它可以以字节(8 位)、半字(16 位)或字(32 位)进行访问。该类存储器，CPU 及 DMA 都可用最快的系统时钟且不插入任何等待进行访问。

SRAM 不支持 HW Parity check。

3.4 Flash

闪存存储器有两个不同的存储区域：

- 主闪存储块 128kbytes，它包括应用程序和用户数据区
- 信息块，其包含两个部分：
  - 选项字节(Option bytes)：内含硬件及存储保护用户配置选项。
  - 系统存储器内存(System memory)：其包含 bootloader 代码。

闪存接口基于 AHB 协议执行指令和数据存取。其预取缓冲的功能可加速 CPU 执行代码的速度。

3.5 EEPROM

内置 2Kbyte EEPROM。采用内置的 IIC 进行操作。

主要特性：

- 8 字节页写模式
- 部分页允许写保护
- 写操作前自动擦除
- EEPROM 大小:256x8 (2K)
- 高可靠性:典型的 100,000,000 循环的耐久性

3.6 启动配置

可根据用户选项字节中的 BOOTSEL_bit 配置是否使能 BOOT0 脚作为启动配置引脚或普通引脚 PF8。

当 BOOTSET_bit 为 1 时，用用户选项字节中的 BOOT1_bit 和 BOOT0_bit 来指定启动模式，如下表所示：

启动 BOOT0 脚	启动模式选择		启动模式	说明
BOOTSEL_bit	BOOT1_bit	BOOT0_bit		
1	x	1	主闪存存储器	主闪存存储器选为启动区域
1	0	0	内置SRAM	内置SRAM 选为启动区域
1	1	0	系统存储器	系统存储器选为启动区域

当 BOOTSET_bit 为 0 时，用用户选项字节中的 BOOT1_bit 和 BOOT0_pin 来指定启动模式，如下表所示：

启动 BOOT0 脚	启动模式选择		启动模式	说明
BOOTSEL_bit	BOOT1_bit	BOOT0_pin		
0	x	0	主闪存存储器	主闪存存储器选为启动区域
0	0	1	内置 SRAM	内置 SRAM 选为启动区域
0	1	1	系统存储器	系统存储器选为启动区域

从复位或待机模式唤醒时，CPU 会采样 boot 模式配置值，因此在有待机应用的场合需要保持启动模式的设置。在启动延迟之后，CPU 从地址 0x00000000 获取堆栈顶的地址，并从启动存储器的 0x00000004 指示的地址开始执行代码。

根据选定的启动模式，主闪存存储器，系统存储器或 SRAM 按照以下的说明访问：

- 从主闪存存储器启动:主闪存存储器被映射到启动存储空间(0x00000000),但仍然能从原有的地址空间(0x800 0000)访问。即闪存存储器的内容可从两个地址开始访问，0x0000 0000 或 0x800 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空间(0x1FFF EC00)访问。
- 从内置的 SRAM 启动:SRAM 映射到启动空间(0x0000 0000)，但其仍然能够在它原有的地址空间(0x2000 0000)访问。

物理重映射

无论从何种 boot 模式启动后，应用程序可以通过修改 SYSCFG_CFGR1 寄存器中的 MEM_MODE 位来重新映射存储器地址。Cortex-M0 CPU 与 Cortex-M3 和 Cortex-M4 不一样，M0 CPU 不支持中断向量表重映射，如果应用程序没有放在 0x0800 0000 地址，就必须要添加额外的代码用来作为中断服务程序。一种可行的方式就是吧中断向量表重映射到 SRAM：

- 把中断向量表从 Flash 拷贝到 SRAM 的 0x2000 0000 地址。
- 配置 SYSCFG_CFGR1 寄存器的 MEM_MODE[1:0]把 SRAM 映射到 0x0000 0000 地址。
- 配置完成后，一旦有中断发生，CPU 从重映射到 SRAM 中的中断向量表取中断服务程序地址，然后跳转到 Flash 中执行中断服务程序。

内嵌的自举程序

内嵌的自举程序存放在系统存储器，在生产时写入。该程序可以通过 USART1 的 PA14/PA15 或 PA9/PA10 组合对闪存进行重新编程。

4 嵌入式闪存

4.1 闪存主要特性

- 存储器结构
  - 主闪存模块：128K 字节
  - 信息模块：3K 字节
- Flash 特性：
  - 数据位宽：32 位
  - 页大小：1Kbytes
  - 扇区大小：4Kbytes
- Flash 访问位宽：支持字编程；32 位读
- 内含加解密模块，支持 Flash 指令自动加解密，保护片内软件知识产权
- Flash 内容支持 ECC 校验，每个字支持 2bits 出错判断 1bit 数据恢复
- 支持 Flash 读/写保护访问控制
- 包含预取指令 buffer

4.2 闪存功能描述

4.2.1 闪存结构

闪存空间由 32 位宽的存储单元组成，既可以存代码又可以存数据。主闪存块按 128 页（每页 1K 字节）或 32 扇区（每扇区 4K 字节）划分，以扇区为单位设置写保护。

Flash area	Flash memory addresses	Size(Byte)	Name	Description
Main Flash memory	0x0800 0000-0x0800 03FF	1KB	Page0	Sector 0
	0x0800 0400-0x0800 07FF	1KB	Page1	
	0x0800 0800-0x0800 0BFF	1KB	Page2	
	0x0800 0C00-0x0800 0FFF	1KB	Page3	
	……	……	……	……
	0x0800 7000-0x0800 73FF	1KB	Page28	Sector 7
	0x0800 7400-0x0800 77FF	1KB	Page29	
	0x0800 7800-0x0800 7BFF	1KB	Page30	
	0x0800 7C00-0x0800 7FFF	1KB	Page31	
	……	……	……	……
	0x0800 F000-0x0801 F3FF	1KB	Page124	Sector 31
	0x0800 F400-0x0801 F7FF	1KB	Page125	
	0x0800 F800-0x0801 FBFF	1KB	Page126	
	0x0800 FC00-0x0801 FFFF	1KB	Page127	
Information block	0x1FFF EC00-0x1FFF F7FF	3KB	-	System memory
	0x1FFF F800-0x1FFF F840	17 word	-	Option byte

4.2.2 读操作

嵌入式 Flash 模块可以像普遍存储空间一样直接寻址访问。任何对 Flash 模块内容的读操作都须经过专门的判断过程。

取指令和取数据都是通过 AHB 总线读取访问，能够按照 Flash 访问控制寄存器（Flash_ACR）中得选项所指定的方式执行：

- 取指：预取值缓冲区使能后可提高 CPU 运行速度
- 等待周期：等待位的个数，保证正确的读取

取指

Cortex-M0 通过 AHB 总线取指。预取指模块的功效在于提高取指效率。

预取缓冲区

预取指缓冲区分 3 块，每块 4 个字节，其中的内容与 Flash 相同，能够完全替代一次同样大小的读取访问。预取缓冲区的工作使得更快速的 CPU 执行成为可能，因为 CPU 取一个字指令的同时下一个字的指令内容页已经在预取缓冲区中，这意味着如果代码是按照 32 位对齐的前提下，可以达到 2 倍的取指加速。

预取缓冲区只有在等待周期大于 0 的时候采有效，在没有等待周期的时候，是否预取指令对性能没影响。并且预取的效率依赖与应用程序。

预取控制器

预取控制器会根据预取缓冲区的可用空间来把握访问 Flash 的时机。当预取缓冲区中存在至少一块可用空间时，预取控制器会发起一次读取请求。复位后，预取指缓冲区的默认状态是关闭的。**访问等待周期**

为了保护对 Flash 的正确读取，必须在 Flash 访问控制寄存器中的 LATENCY[2:0]中指定预取指控制器的速度比，这个数值等于每次访问 Flash 后到下次访问之间所需插入的等待周期的个数。复位后，这个值默认为零，也就是没有插入等待周期的状态。

4.2.3 Flash 写和擦除操作

ICP(in circuit programming)是指使用 SWD 或 Bootloader 的方法在线改变 Flash 的内容，将用户代码烧录到单片机中。ICP 提供了一种简单高效的方法，免除了烧写芯片时的芯片装夹等问题。

与 ICP 方法不同的是，IAP(in application programming)能够使用 MCU 支持的任何通信接口下载程序或者数据。IAP 允许用户在运行程序的过程中重写应用程序，前提是一部分应用程序必须预先用 ICP 的方法烧写进去。

烧写和擦除操作在整个产品工作电压范围内都可以完成。该操作由下列 8 个寄存器完成：

- 关键字寄存器（FLASH_KEYR）
- 选项字节关键字寄存器 (FLASH_OPRKEYR)
- Flash 控制寄存器 (FLASH_CR)
- Flash 状态寄存器 (FLASH_SR)
- Flash 地址寄存器 (FLASH_AR)
- 选项字节寄存器 (FLASH_OBR)
- 写保护寄存器 (FLASH_WRPR)
- Flash 控制寄存器 2 (FLASH_ECR)

只要 CPU 不去访问 Flash 空间，进行中的 Flash 写操作不会妨碍 CPU 的运行。也就是说，在对 Flash 进行写/擦除操作的同时，任何对 Flash 的访问都会令总线停顿，直到写/擦除操作完成后才会继续执行，这意味着在写/擦除 Flash 的同时不可以对它取指和访问数据。

在对 Flash 空间做写/擦除操作时，内部 RC 振荡器(HSI)必须处于开启状态。

对 Flash 空间的解锁

复位后，Flash 存储器默认是受保护状态的，这样可以防范意外的擦除动作。FLASH_CR 寄存器不允许被改写，除非执行一串针对 FLASH_KEYR 寄存器的解锁操作才能开启对 FLASH_ CR 的访问权限。这串操作由下面 2 个写操作构成：

- 写关键字 KEY1=0x45670123
- 写关键字 KEY2=0xCDEF89AB

任何错误的顺序将会锁死 FLASH_CR 直至下次复位。当发生关键字错误时，会由总线错误引发一次硬件错误中断。如果 KEY1 出错就会立即中断，或如果 KEY1 正确但 KEY2 错误时就会在 KEY2 错的那个时候引发中断。

主闪存编程

主闪存一次可以编程 32 位，直接对相应的地址写一个字（32 位），当 FLASH_CR 中的 WPG 位为 1 时，直接对相应的地址写一个字（32 位），就是一次编程操作。

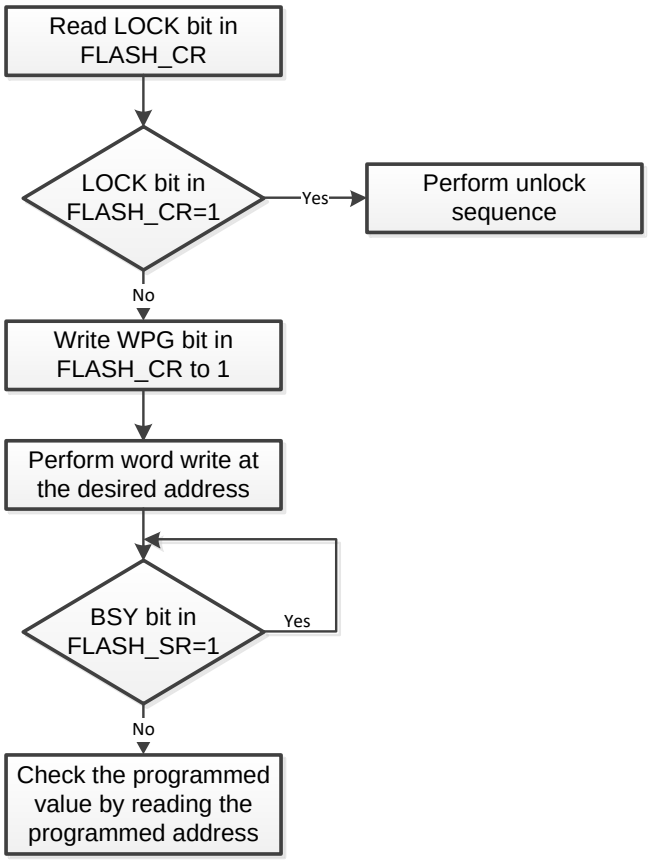
Flash 存储器接口会预读一下待编程字节后是否为全 1，如果不是，那么编程操作会自动取消，并且在 FLASH_SR 寄存器的 PGERR 位上提示编程错误告警。如果被编程的内容为全零，则会例外，这时会正确编程并且不告警。

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

主 Flash 存储器标准模式下的编程过程如下：

- 检查 FLASH_SR 中的 BSY 位，以确认上次操作已经结束
- 置位 FLASH_CR 寄存器中的 WPG 位
- 根据配置，以字为单位向目标地址写入数据

- 等待 FLASH_SR 寄存器中的 BSY 归零
- 检查 EOP 标志位（如果 Flash 编程成功会置位 EOP），然后软件清除该标志位



Flash 存储器擦除

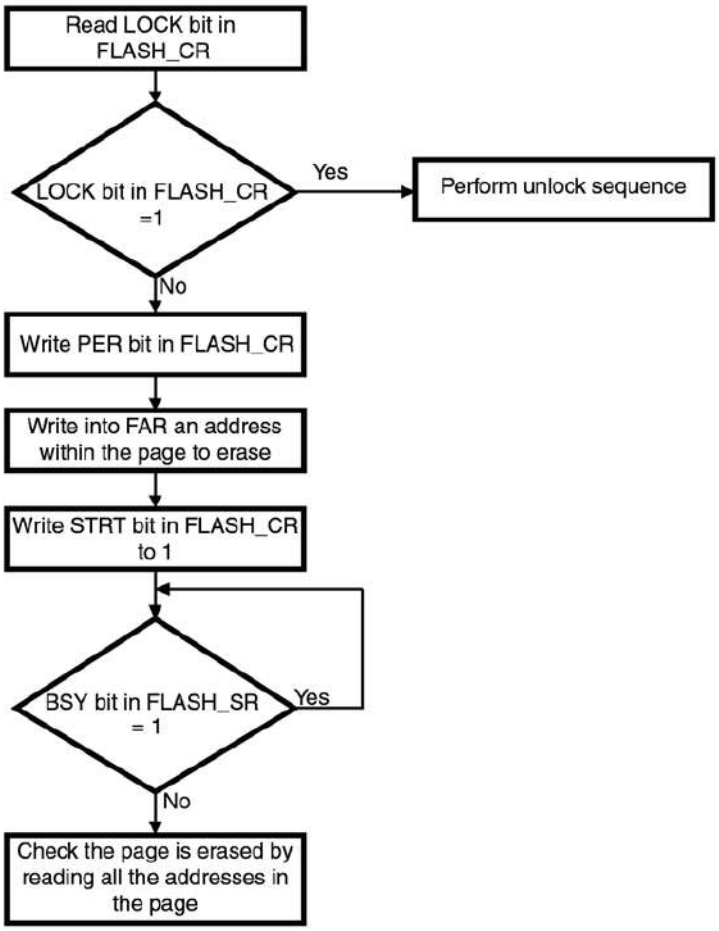
Flash 存储器可以按页为单位擦除、半页为单位擦除，也可以整片擦除。

页擦除

擦除页的步骤如下：

- 检查 FLASH_SR 中的 BSY 位，以确认上次操作已经结束
- 置位 FLASH_CR 寄存器中得 PER 位为 1
- 写 FLASH_AR 寄存器以选择待擦除的页
- 置位 FLASH_CR 寄存器中的 STRT 位为 1
- 等待 FLASH_SR 中的 BSY 归零
- 检查 EOP 标志位（如果 Flash 擦除成功会置位 EOP），然后软件清除该标志位

擦除页流程如下图：



半页擦除

Flash 半页为 512 字节，半页擦除流程和页擦除类似，区别在于把 FLASH_ECR 中的 HPER 位置 1，而不是把 FLASH_CR 中的 PER 置 1。

整片擦除

可以用整片擦除命令一次擦除整个 Flash 户区，但信息块不会受这个命令影响，具体步骤如下：

- 检查 FLASH_SR 中的 BSY 位，以确认上次操作已经结束
- 置 FLASH_CR 寄存器中的 MER 位为 1
- 置 FLASH_CR 寄存器中的 STRT 位为 1
- 等待 BSY 位归零
- 检查 EOP 标志位（如果 Flash 擦除成功会置位 EOP），然后软件清除该标志位

选项字节编程

选项字节的编程与常规用户地址不同，总共就是 12 个字。解除 Flash 访问限制后，还需要针对 FLASH_OPTKEYR 寄存器完成关键字写入操作。完成该操作后，FLASH_CR 寄存器中的 OPTWRE 位会被置 1，然后就可以先置位 FLASH_ CR 中的 OPTPG 位，再按字单位写目标地址。同样是会自动检查选项字节是否为 1，否则相关操作会被取消并且在 FLASH_SR 中的 WRPRTERR 位提示错误。编程操作结束后，会由 FLASH_SR 寄存器的 EOP 位给出提示。

在编程操作开始前，LSB 值会自动补到 MSB，这会保证选项字节的值总是对的。步骤如下：

- 检查 FLASH_SR 寄存器中的 BSY 位，以确保上次操作结束
- 解锁 FLASH_CR 寄存器中的 OPTWRE 位
- 置 FLASH_CR 寄存器中的 OPTPG 位为 1
- 写数据（字 32bit）到目标地址
- 等待 BSY 位归零

读取并校验当读保护选项字节由保护状态被改成非保护状态时，会自动引发一次整片擦除，然后才改写

读保护位数据。如果用户只想改写其他的字节，则不会引发整片擦除，这个机制用于保护 Flash 的内容。

擦除过程

选项字节的擦除过程如下：

- 检查 FLASH_SR 寄存器中的 BSY 位，以确保上次操作结束
- 解锁 FLASH_CR 寄存器中的 OPTWRE 位
- 置 FLASH_CR 寄存器中的 OPTER 位为 1
- 置 FLASH_CR 寄存器中的 STRT 位为 1
- 等待 BSY 位归零
- 读取并校验

4.2.4 读保护

将选项字节中的RDP 字节置位，然后重新复位，读保护就被激活了。存在 Level0（无保护）到Level2（最大限度）保护三个保护级别。系统存储区不受读保护字节的影响，但该区域不允许编程和擦除操作。Flash 存储器的保护级别和RDP 选项字节及其补数内容的对应关系，如下表：

RDP 字节值	RDP 取反值	读保护级别
0xAA	0x55	Level 0
任意值，除 0xAA 和 0xCC 外	任意值（不要求互补）除 0x55 和 0x33 外	Level 1（默认）
0xCC	0x33	Level 2

Level 0：无保护

针对主 Flash 区域的读写和擦除操作都被允许，选项字节也全都可以操作。

Level 1：读保护

这是 RDP 选项字节被擦除之后的默认保护级别。对应的 RDP 值为除 0xAA 和 0xCC 以外的任意值或者其补数不正确。

- 用户模式：在用户模式下执行的代码允许对主 Flash 和选项字节做全部操作。
- Debug 模式：包括 boot RAM 和 boot loader 模式。在调试模式下或运行在 boot RAM 和 bootloader 状态下，主 Flash 区和备份寄存器均不允许访问。在该状态下，任何简单的读访问都会引起总线错误并引发硬件错误中断。主 Flash 区同时也禁止写和擦除操作，以防范恶意程序修改代码，任何尝试改写的操作都会引起 FLASH_SR 中的 PGERR 标志置位。当 RPD 字节的内容有 0xAA 改为 Level 0 的级别会先执行整片擦除操作，并且备份寄存器的值也会 被复位。

Level 2：无 debug

在这个级别上，Level1 的保护功能肯定都是有的，除此之外，CortexM0 的调试接口也被禁止了，以及从 RAM 启动、系统区启动等功能也都没有了。

在用户执行模式下，允许对主 Flash 区做全部操作，相反对于选项字节却只能读取和写入而并不能做擦除。

此外，RDP 字节不能再改写，因此 Level2 这个保护级别永远不能清除掉，是一个不可恢复性的操作。当试图改写 RDP 字节时，FLASH_SR 寄存器中的保护错误标志 WRPRTERR 会被置位并引发一个中断。

保护状态和保护级别及运行模式对照表如下表：

区域	保护级别	用户代码执行			调试/从 RAM 启动/从系统区域启动		
		读	写	擦除	读	写	擦除
主 Flash 区域	1	Yes	Yes	Yes	No	No	No(3)
	2	Yes	Yes	Yes	N/A(1)	N/A(1)	N/A(1)
系统区域 (2)	1	Yes	No	No	Yes	No	No
	2	Yes	No	No	NA(1)	N/A(1)	N/A(1)
选项字节	1	Yes	Yes(3)	Yes	Yes	Yes(3)	Yes
	2	Yes	Yes(4)	No	N/A(1)	N/A(1)	N/A(1)

备份寄存器	1	Yes	Yes	N/A	No	No	Yes
	2	Yes	Yes	N/A	N/A(1)	N/A(1)	N/A(1)

(1) 当 Level/2 保护级别使能，调试口从 RAM 启动和从系统启动都被禁止
(2) 系统区是唯一在任何情况下可读的区域
(3) 当 RDP 被改成不保护时，主 Flash 会被擦除
(4) 所有的选项字节中，除 RDP 字节外都能被再次编程

改变读保护级别

改变 RDP 的值到其他值（除 0xCC 以外）就可以轻松的从 Level 0 级迁移到 Level 1 级别。将 RDP 写成 0xCC，就可以直接进入 Level 2 级别。相反的，绕开整片擦除动作而进入 Level 0 级别是不可能的。在把 RDP 成功改写成 0xAA 之前，整片擦除动作已经启动了。

4.2.5 写保护

写保护以一个扇区为单位（4 页）来控制，配置选项字节中的 WRP 位， 然后通过 FLASH_CR 寄存器中的 OBL_LAUNCH 位强制重新加载选项字节就可以使能这个保护。如果试图写入或擦除一个受保护的扇区，会引起 FLASH_SR 中的 WRPRTERR 标志位被置位。

写保护的解除

解除写保护有 2 个应用例子可以提供：

- 例 1：在解除写保护后禁止读保护
  - 使用 FLASH_CR 中的 OPTER 位擦除整个选项字节区域
  - 向 RDP 写入 0xAA 从而解除所有保护，这自然会引起整片擦除
  - 设置 FLASH_CR 中的 OBL_LAUNCH 位，引起选项字节(和新 WRP[1:0]位)重新加载写保护解除
- 例 2：在解除写保护后，读保护仍然有效，这种办法对使用用户 boot loader 进行在应用编程时很有用
  - 使用 FLASH_CR 中的 OPTER 位擦除整个选项字节区域
  - 设置 FLASH_CR 中的 OBL_LAUNCH 位，引起选项字节(和新 WRP[1:0]位) 重新加载写保护解除

4.2.6 选项字节的写保护

选项字节默认是写保护的并且任何时候都可读。为了对选项字节进行写/擦除操作，必须对 POTKEYR 顺序写入关键字。正确的关键字会引起 FLASH_CR 中的 OPTWRE 置位，表明解锁成功。同样，通过对该位清零，能够再度禁止对选项字节的写操作。

4.2.7 Flash 数据加密

Flash 控制器内部有一个加解密模块用于加密用户存储在 Flash 中的程序和数据，用户可以选择以密文的方式存储或者以明文的方式存储。当用户使能数据加解密后，Flash 控制器会自动进行加解密运算，加解密过程对应用软件透明。如果用户使能 Flash 数据加密后，同样的程序在两颗不同芯片上实际存储的内容也不一样。加密使用的用户秘钥为 64 位。

芯片默认不使能 Flash 加解密。

在 Flash 的选项字中存储有加密/解密使能位和用户 KEY，在系统复位的时候 Flash 控制器会自动从 Flash 载入加密/解密使能标志和用户 KEY。另外还可以配置一组影子寄存器来改写加解密标志和用户 KEY，这是为了方便在通过调试口发行程序的时候可以直接写入使能加密，写入密文到 Flash。如果已经把 KEY 保存到 Flash 选项字，然后再读选项字地址，则读到的值始终为 0xaaaa aaaa，以保证用户 KEY 不会泄露出去。

当使能加密后，只加密 Flash 主空间，不会加密 Flash 系统存储器和选项字部分。如果主空间的内容被加密后，但是没有使能解密，则 CPU 读到的是密文，会执行出错。如果主空间的内容没有被加密，但是使能解密了，则 CPU 读到的也是乱码，执行会出错。因此在程序执行时加密和解密必须同时使能或者同时不使能。

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

方法一：

- 往寄存器 0x40022078 写入 0x1357_eca8 （配置 ENCRY_EN）
- 往寄存器 0x40022084 和 0x40022080 写入 UKEY，UKEY 的高 32 位存到 0x40022084
- 往 Flash 烧录程序

- 往 Flash 地址 0x1FFFF820 写入 0x1357_eca8

例程：

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

- 往 Flash 地址 0x1FFFF824 写入 0x2468_db97

例程：

```
FLASH->KEYR = 0x45670123 ;
FLASH->KEYR = 0xCDEF89AB ;
FLASH->OPTKEYR = 0x45670123 ;
FLASH->OPTKEYR = 0xCDEF89AB ;
FLASH->CR |= 0x00000010 ；
*((volatile u16 *)(0x1FFFF824)) = 0xdb97;
while( ( FLASH->SR & 0x00000001 ) == 0x00000001 ) ;
*((volatile u16 *)(0x1FFFF826)) = 0x2468;
while( ( FLASH->SR & 0x00000001 ) == 0x00000001 ) ;
FLASH->SR = 0x20 ;
FLASH->CR &= 0xfffffffe ;
```

- 往 Flash 地址 0x1FFFF828~0x1FFFF82f 写入 UKEY，UKEY 的高 byte 写到高地址

例程：

```
FLASH->KEYR = 0x45670123 ;
FLASH->KEYR = 0xCDEF89AB ;
FLASH->OPTKEYR = 0x45670123 ;
FLASH->OPTKEYR = 0xCDEF89AB ;
FLASH->CR |= 0x00000010 ；
*((volatile u16 *)(0x1FFFF828)) = UKEY[15:0];
while( ( FLASH->SR & 0x00000001 ) == 0x00000001 ) ;
*((volatile u16 *)(0x1FFFF82a)) = UKEY[31:16];
while( ( FLASH->SR & 0x00000001 ) == 0x00000001 ) ;
*((volatile u16 *)(0x1FFFF82c)) = UKEY[47:32];
while( ( FLASH->SR & 0x00000001 ) == 0x00000001 ) ;
*((volatile u16 *)(0x1FFFF82e)) = UKEY[63:48];
while( ( FLASH->SR & 0x00000001 ) == 0x00000001 ) ;
FLASH->SR = 0x20 ;
FLASH->CR &= 0xfffffffe ;
```

方法二：

- 往 Flash 地址 0x1FFFF820 写入 0x1357_eca8

例程如方法一所示

- 往 Flash 地址 0x1FFFF824 写入 0x2468_db97

例程如方法一所示

- 往 Flash 地址 0x1FFFF828~0x1FFFF82f 写入 UKEY，UKEY 的高 byte 写到高地址

例程如方法一所示

- 复位
- 往 Flash 烧录程序

注意事项：

- 往 Flash 地址 0x1FFFF828~0x1FFFF82f 写入 UKEY 后，不能读出来校验，因为往 Flash 地址 0x1FFFF828~0x1FFFF82f 写入 UKEY 后再读 0x1FFFF828~0x1FFFF82f 得到的值始终为 0xaaaaaaaa
- 往寄存器 0x40022084 和 0x40022080 写入 UKEY 后也不能读出来校验

4.3 Flash 中断

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

4.4 Flash 寄存器描述

4.4.1 Flash 访问控制寄存器（FLASH_ACR）

偏移地址：0x00
复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

位 31：6	保留:必须保持复位值
位 5	**PRFTBS**：预取缓冲区状态该位提供预取缓冲区的状态； 0：预取缓冲区被禁止； 1：预取缓冲区被使能
位 4	**PRFTBE**：预取缓冲区使能； 0：禁止预取指； 1：使能预取指
位 3	保留:必须保持复位值
位 2：0	**LATENCY[2:0]**：等待周期；本位预设 HCLK 周期和 Flash 访问时间的比率关系； 000：零等待周期，适用于 0<HCLK≤24MHz； 001：1 个等待周期，适用于 24MHz<HCLK≤48MHz 010：2 个等待周期，适用于 48MHz<HCLK≤72MHz 011：3 个等待周期 100：7 个等待周期 101：9 个等待周期 110：19 个等待周期 111：39 个等待周期 LATENCY 配置为 011~111 适用于 CPU 可以在极低频率运行的应用程序，通过配置大的等待周期可以降低整个芯片的功耗。

4.4.2 Flash 关键字寄存器（FLASH_KEYR）

偏移地址：0x04
复位值：xxxx xxxx

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

位 31:0	**FKEY[31:0]**：关键字 该位用于输入关键字以解锁 Flash

4.4.3 Flash 选项关键字寄存器 (FLASH_OPTKEYR)

偏移地址：0x08

复位值：xxxx xxxx

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

位 31:0	**OPTKEY[31:0]**：关键字 该位用于输入关键字以解锁 Flash

4.4.4 Flash 状态寄存器 (FLASH_SR)

偏移地址：0x0C

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ECC2BIT_ERR	ECC1BIT_ERR	Res	Res	Res	SIZE_ERR	Res	Res	Res	Res	EOP	WRPRT_ERR	Res	PG_ERR	Res	BSY
rw	rw				rw					rw	rw		rw		r
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位 31:16	保留:必须保持复位值
位 15	**ECC2BIT_ERR** 1：读 Flash 时 ECC 检测到 2bit 数据出错 0：读 Flash 时 ECC 未检测到 2bit 数据出错 往 ECC2BIT_ERR 写 1 清零 ECC2BIT_ERR

位 14	**ECC1BIT_ERR** 1：读 Flash 时 ECC 检测到 1bit 数据出错 0：读 Flash 时 ECC 未检测到 1bit 数据出错 往 ECC1BIT_ERR 写 1 清零 ECC1BIT_ERR
位 13:11	保留:必须保持复位值
位 10	**SIZE_ERR** 1：用户编程时数据位宽 32 位 0：未发生编程位宽错 往 SIZEERR 写 1 清零 SIZEERR
位 9:6	保留:必须保持复位值
位 5	**EOP**：操作结束； 当 Flash 操作（写/擦除）完成时由硬件置位。软件写 1 后可清零。 Note：只有成功的写或擦除操作才会由硬件置位 EOP。
位 4	**WRPRT_ERR:** 写保护错误标志,当出现对写保护区域的写操作时被硬件置位。 软件写 1 后可清零
位 3	保留:必须保持复位值
位 2	**PGERR:** 写入错误标志； 字编程时，如果被编程区域的值不为‘0xffff ffff’，则执行写入操作时被硬件置位。 软件写 1 后可清零
位 1	保留:必须保持复位值
位 0	**BSY:** 忙标志 该位标明 Flash 操作处于过程中。当开始 Flash 操作的时候被硬件置位，当操作结束时或发生错误时被硬件清零。

4.4.5 Flash 控制寄存器 (FLASH_CR)

偏移地址: 0x10

复位值: 0x0000 0080

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ECC2BIT_ERRIE	ECC1BIT_ERRIE	ECC_AUTO_CORRECT
													rw	rw	rw
0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1

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

位 31:19	保留:必须保持复位值
位 18	**ECC2BIT_ERRIE** 1：使能读 Flash 时 ECC 检测到 2bit 数据出错中断 0：关闭读 Flash 时 ECC 检测到 2bit 数据出错中断
位 17	**ECC1BIT_ERRIE** 1：使能读 Flash 时 ECC 检测到 1bit 数据出错中断 0：关闭读 Flash 时 ECC 检测到 1bit 数据出错中断
位 16	**ECC_AUTO_CORRECT** 1：使能 ECC 自动纠错，当 ECC 判断到有 1bit 数据出错时，自动把出错位纠正 0：关闭 ECC 自动纠错，当 ECC 判断到有 1bit 数据出错时，不会纠正出错位
位 15:14	保留:必须保持复位值

位 13	**OBL_LAUNCH:** 选项字节强制更新 当被写为 1 时，该位强制选项字节的重加载，该操作会引起系统复位。 0: 无效 1: 有效
位 12	**EOPIE:** 操作结束中断使能 该位使能操作结束中断，使得 FLASH_SR 中的 EOP 位变成 1 的时候产生中断请求 0: 中断禁止 1: 中断使能
位 11	保留:必须保持复位值
位 10	**ERRIE:** 错误中断使能 该位使能操作错误中断，使得 FLASH_SR 中的 PGERR/WRPRTERR 位变成 1 的时候产生中断请求。 0: 中断禁止 1: 中断使能
位 9	**OPTWRE**: 选项字节写使能 该位为 1 时，选项字节即允许改写。对 FLASH_OPTKEYR 寄存器写入正确的关键字序列就可以将它置 1。
位 8	保留:必须保持复位值
位 7	**LOCK**: 锁定 Flash 标志 只能写 1。当该位为 1 时，表明 Flash 为锁定状态。该位可以解锁时序来清零，当发现解锁不成功时，该位就一直为 1 了，除非下次复位重新操作。
位 6	**STRT:** 启动 该位会触发一个擦除操作，仅由软件置 1，仅会在 BSY 被清零时清零。
位 5	**OPTER:** 选项字页擦除 选项字页擦除时选择
位 4	**OPTPG:** 选项字写入 选项字(32bit)写入时选择
位 3	保留:必须保持复位值
位 2	**MER**: 整片擦除 整片擦除时选择
位 1	**PER**: 页擦除 页擦除时选择
位 0	**WPG**: 字写入 1：字编程 Flash 0：无意义

4.4.6 Flash 地址寄存器 (FLASH_AR)

偏移地址: 0x14

复位值: 0x0000 0000

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

位 31:0	**FAR:** Flash 地址 当 WPG 位被选中时，选择待写入的地址，或当 PER 位被选中时，选择待擦除的页。

4.4.7 Flash 选项字节寄存器 (FLASH_OBR)

偏移地址: 0x1C
复位值 : 0xXXXX XX0X
本寄存器的复位值取决于选项字节的写入值

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATA1								DATA0							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BOOT_SEL	Res.	Res.	nBOOT1	nBOOT0	nRST_STDBY	nRST_STOP	WDG_SW	Res.	Res.	Res.	Res.	Res.	RDPRT[1:0]		OPTERR
r			r	r	r	r	r						r	r	r
0	0	0	0	0	1	1	0	1	0	0	0	0	0	0	0

位 31:24	**DATA1**
位 24:16	**DATA0**
位 15	保留:必须保持复位值
位 14:13	保留:必须保持复位值
位 12	nBOOT1
位 11	nBOOT0
位 10:8	用户选项字节 **nRST_STDBY** **nRST_STOP** **WDG_SW**
位 7:3	保留:必须保持复位值
位 2	**nRST_STDBY**
位 1	**RDPRT[1:0]:**读保护级别 00:保护状态,当置位时，表明当前处于 Level0 保护状态 01:保护状态,当置位时，表明当前处于 Level1 保护状态 11:保护状态,当置位时，表明当前处于 Level2 保护状态
位 0	**OPTERR** 选项字节错误 当置位时，表明加载选项字节发现互补关系不成立。

4.4.8 Flash 写保护寄存器 (FLASH_WRPR)

偏移地址: 0x20
复位值 : 0xXXXX XXXX

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

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

位 31:0	**WRP**：写保护 保持由 OBL 载入的写保护选项字。

4.4.9 Flash 选项字节寄存器 2(FLASH_OBR2)

偏移地址: 0x60

复位值 : 0x0000 00XX

本寄存器的复位值取决于选项字节的写入值

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

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

位 31:8	保留:必须保持复位值
位 7:0	**BOR_LEVEL**:由 option word 内容决定，复位时自动 load 到 BOR_LEVEL[7:0]寄存器

4.4.10 Flash 控制寄存器 2 (FLASH_ECR)

偏移地址: 0x70

复位值 : 0x0000 0000

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

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

位 31:1	保留:必须保持复位值
位 0	**HPER:** 半页擦除

4.4.11 Flash 数据加密控制寄存器 (ENCRY_CTL)

偏移地址: 0x78

复位值 : 0x0000 000X

ENCRY_EN 的复位值有选项字决定。

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

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

位 31:1	保留:必须保持复位值

位 0	**ENCRY_EN**: 加密使能 当 Flash 选项字地址 0x1FFF F820 中的值为 0x1357 eca8 时，ENCRY_EN 的复位值为 1，否则其复位值为 0。 如果软件向 ENCRY_CTL 寄存器写入值 0x1357 eca8 也会置位 ENCRY_EN，如果软件写入的值不是 0x1357 eca8，则把 ENCRY_EN 清零。

4.4.12 Flash 数据解密控制寄存器 (DECRY_CTL)

偏移地址: 0x7C
复位值 : 0x0000 000X
DNCRY_EN 的复位值有选项字决定。

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

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

位 31:1	保留:必须保持复位值
位 0	**DNCRY_EN**: 解密使能 当 Flash 选项字地址 0x1FFF F824 中的值为 0x2468_db97 时，DECRY_EN 的复位值为 1，否则其复位值为 0。 如果软件向 DECRY_CTL 寄存器写入值 0x2468_db97 也会置位 DECRY_EN，如果软件写入的值不是 0x2468_db97，则把 DECRY_EN 清零。

4.4.13 Flash 秘钥寄存器 1 (UKEY1)

偏移地址: 0x80
复位值 : 0x0000 0000

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

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

位 31:0	**UKEY[31:0]:** Flash 数据加密的用户秘钥低 32 位。 系统复位时，Flash 控制器会自动把选项字中的秘钥低 32 位载入到 UKEY1 寄存器，但是本寄存器软件不可读，软件读时返回值始终为 0。

4.4.14 Flash 秘钥寄存器 2 (UKEY2)

偏移地址: 0x84
复位值 : 0x0000 0000

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

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

位 31:0	**UKEY[63:32]:** Flash 数据加密的用户秘钥高 32 位。 系统复位时，Flash 控制器会自动把选项字中的秘钥低 32 位载入到 UKEY2 寄存器，但是本寄存器软件不可读，软件读时返回值始终为 0。 软件可以向 UKEY2 和 UKEY1 寄存器写人不同值来改写秘钥。 尤其适合在芯片初始化时，还没有配置 Flash 选项字中的秘钥，但同时想把数据加密后下载到 Flash 的应用场合。这种场景中，在芯片复位后，用户在开始下载数据之前，先向 UKEY2 和 UKEY1 寄存器写入秘钥，然后开始下载 Flash 数据，最后再把秘钥写人到选项字对应地址。这样下一次芯片复位时就能从选项字中载入正确的秘钥，并解密存储在芯片内 Flash 中的内容。

4.4.15 中断向量表偏移 (INT_VEC_OFFSET)

偏移地址: 0x90

复位值 : 0x0000 0000

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

位 31:15	保留:必须保持复位值
位 14:0	**INT_VEC_OFFSET[14:0]**:中断向量表偏移地址，最低两位必须等于 0. 当中断发生是，自动返回存放在 INT_VEC_OFFSET[14:0]地址的值作为中断服务程序入口地址。INT_VEC_OFFSET[14:0]只对 0x0000_0000~0x0000_0100 和 0x0800_0000~0x0800_0100 地址范围做重映射。如果配置 INT_VEC_OFFSET[14:0]为非 0 值，则在编译时，需要把最低 256bytes 地址空间都留给中断向量表，否则程序可能会执行出错。

4.5 Flash 选项字节描述

这些选项由终端用户的应用需求配置。配置示例：看门狗是由硬件还是软件启动的模式配置。

32 位字在选项字节中按如下方式拆分：

31-24	23-16	15 -8	7-0
选项字节 1 取反	选项字节 1	选项字节 0 取反	选项字节 0

在选择字的内容更新后，需要系统复位后才能生效。可以通过置位 OBL_LAUNCH 或者除法 NRST PAD 产生系统复位。

选项字节结构：

Address	[31:24]	[23:16]	[15:8]	[7:0]
0x1FFF F800	nUSER	USER	nRDP	RDP
0x1FFF F804	nData1	Data1	nData0	Data0
0x1FFF F808	nWRP1	WRP1	nWRP0	WRP0
0x1FFF F80C	nWRP3	WRP3	nWRP2	WRP2
0x1FFF F810	Reserved		nBOR_LEVEL	BOR_LEVEL

0x1FFF F814 ~0x1FFF F81C	Reserved	
0x1FFF F820	ENCRY_CTL[31:0]	
0x1FFF F824	DECRY_CTL[31:0]	
0x1FFF F828	UKEY[31:0]	
0x1FFF F82C	UKEY[63:32]	
0x1FFF F830	IWDG_INI_KEY	IWDG_RL_IV[11:0]
0x1FFF F834	Reserved	LSI_LP_CTL[15:0]
0x1FFF F838 ~0x1FFF F83C	Reserved	
0x1FFF F840	DBG_CLK_CTL[31:0]	

选项字节描述

闪存存储器地址	选项字节
0x1FFF F800	位 [31:24]: nUSER 位 [23:16]: USER 用户选项字节(存于 FLASH_OBR[15:8])该字节用于配置如下特性： –选择看门狗事件:硬件还是软件。 –当进入停止模式时复位事件。 –当进入待机模式时复位事件。 位 23 : BOOT_SEL:选择 BOOT0_PIN 或者 BOOT0_bit 作为系统启动配置 位 22： 保留 位 21： 保留 位 20 : BOOT1，连同 BOOT0,选择从主闪存存储器，SRAM 或系统内存启动。 位 19 : BOOT0 位 18 : nRST_STDBY 0: 当进入待机模式产生复位 . 1: 不产生复位 位 17 : nRST_STOP 0: 当进入停机模式产生复位 1: 不产生复位 位 16: WDG_SW 0: 硬件看门狗 1: 软件看门狗 位 [15:8]: nRDP 位 [7:0]: RDP: 读保护选项字节 该字节的值决定闪存存储器保护级别 0xAA： 水平 0 0xXX (除了 0xAA 和 0xCC):水平 1 0xCC： 水平 2
0x1FFF F804	Datax:两字节的用户数据。这些地址可由选项字节编程过程进行编程 位 [31:24]: nData1 位 [23:16]: Data1(存于 FLASH_OBR[31:24]) 位 [15:8]: nData0 位 [7:0]: Data0(存于 FLASH_OBR[23:16])
0x1FFF F808	WRPx: 闪存存储器写保护选项字节 位 [31:24]: nWRP1 位 [23:16]: WRP1 (存于 FLASH_WRPR[15:8]) 位 [15:8]: nWRP0 位 [7:0]: WRP0 (存于 FLASH_WRPR[7:0]) 0: 写保护使能 1: 写保护失能

0x1FFF F80C	WRPx: 闪存存储器写保护选项字节 位 [31:24]: nWRP3 位 [23:16]: WRP3 (存于 FLASH_WRPR[31:24]) 位 [15:8]: nWRP2 位 [7:0]: WRP2 (存于 FLASH_WRPR[23:16]) 0: 写保护使能 1: 写保护失能
0x1FFF F810	位 [15:8]: nBOR_LEVEL 位 [7:4]: 保留 位 [3:0]:BOR_LEVEL: 欠压复位阈值级别 (Brown out reset threshold level) 这些位复位 1.45 V 到 1.55 V 电压范围的阈值级别（仅限掉电）。在此特殊情况下，VDD 必须高于 VBOR0 才能启动器件 OBL 序列，从而禁止 BOR。 PDR 随后将监视掉电。如果禁止 BOR， 1.65 V 与 VPDR 阈值之间会存在一个“灰色区域”（这意味着， VDD 可低于最低工作电压 (1.65 V)，到达 VPDR 阈值之前不会复位）。 如果该配置在选项字节加载期间发生不匹配，则为其加载 0x8。 0xxx： BOR 关闭。这是 1.45 V - 1.55 V 电压范围的复位阈值级别（仅限掉电）。在此特殊情况下， VDD 必须高于 BOR 级别 1 才能启动器件 OBL 序列，从而禁止 BOR。 PDR 随后将监视掉电。 注： 如果禁止 BOR， 1.65 V 与 VPDR 阈值之间会存在一个“灰色区域”（这意味着， VDD 可低于最低工作电压 (1.65 V)，达到 VPDR 阈值之前不会引发复位） 1000： BOR 级别 1，是 VBOR0 的复位阈值电压（大约 1.8 V） 1001： BOR 级别 2，是 VBOR1 的复位阈值电压（大约 2.0 V） 1010： BOR 级别 3，是 VBOR2 的复位阈值电压（大约 2.5 V） 1011： BOR 级别 4，是 VBOR3 的复位阈值电压（大约 2.7 V） 1100： BOR 级别 5，是 VBOR4 的复位阈值电压（大约 3.0 V） 注： 有关 BOR 级别的确切定义，请参见器件数据手册。
0x1FFF F820	ENCRY_CTL: Flash 数据加密使能控制 0x1357_eca8： 使能 Flash 数据加密 其它值: 关闭 Flash 数据加密
0x1FFF F824	DECRY_CTL: Flash 数据解密使能控制 0x2468_db97： 使能 Flash 数据解密 其它值: 关闭 Flash 数据解密
0x1FFF F828	UKEY[31:0]: 用户秘钥低 31 位
0x1FFF F82C	UKEY[63:32]: 用户秘钥高 31 位
0x1FFF F830	位 [31:16]: IWDG_INI_KEY[15:0]：决定 IWDG_RL_IV 是否生效，当 IWDG_INI_KEY[15:0]为 0x5b1e 时，IWDG_RL_IV 配置有效，否则无效。 位 [15:12]: 保留 位 [11:0]: IWDG_RL_IV[11:0]，存储 IWDG_RLR 寄存器的初始值，当 IWDG 配置为 hardware watchdog 时，可以配置 IWDG_RL_IV[11:0]来设计 IWDG 的复位时间间隔。
0x1FFF F834	位 [31:16]: 保留 位 [15:0]: LSI_LP_CTL[15:0]：存储的值为 0x369c 时，MCU 进入 STOP 或者 STANDBY mode 后，LSI 可以根据 LSION 的设置关掉 LSI；在 MCU 唤醒后，LSI 恢复成进模式之前的状态。如果不配置 LSI_LP_CTL，则如果在使能 IWDG 后再进入 STOP 或者 STANDBY mode，系统会被 IWDG 周期唤醒。用户可以通过配置 LSI_LP_CTL 来决定在使能 IWDG 后再进入 STOP 或者 STANDBY mode 时，是否需要被 IWDG 周期唤醒。
0x1FFF F840	DBGCLK_CTL[31:0]： 0x1234_bcde： 关闭芯片内部调试组件的时钟 其它值：使能芯片内部调试组件的时钟

每次系统复位后，选项字节装载器(OBL)读信息块数据并且存储这些数据到相应的选项字节寄存器(FLASH_OBR)和闪存保护寄存器(FLASH_WRPR)中。选项字节有其值的取反数据同样存放在信息块中，其目的用于校验选项字节的正确性。当选项字节装载后，CPU 会检查选项字节的正确性，若选项字节与其取反值比较不一致时，会产生选项字节校验错 (OPTERR)信息。当比较错产生后，CPU 会强制相应的选项字节值变为 0xFF。当选项字节与其取反码都为 0XFF（擦除状态）时，CPU 就不会比较其与取反码的差异。

5 循环冗余校验计算单元 (CRC)

5.1 简介

CRC（循环冗余校验）计算单元使用多项式发生器从一个 8 位/16 位/32 位的数据字中产生 CRC 码。

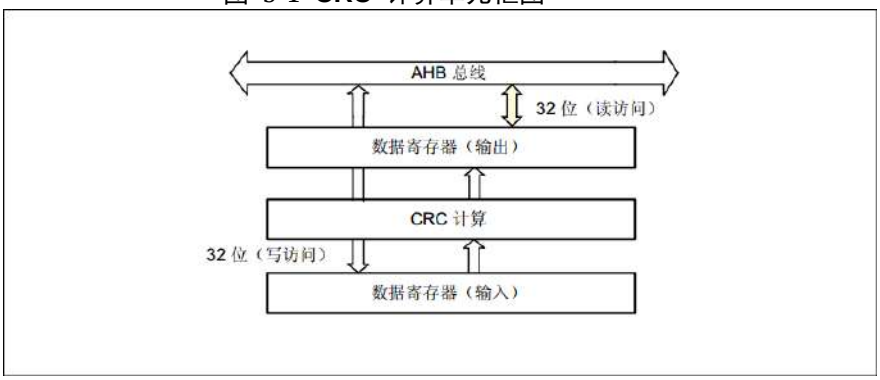
在众多的应用中，基于 CRC 的技术还常用来验证数据传输或存储的完整性。根据功能安全标准的规定，这些技术提供了验证 Flash 完整性的方法。 CRC 计算单元有助于在运行期间计算软件的签名，并将该签名与链接时生成并存储在指定存储单元的参考签名加以比较。

5.2 CRC 主要特性

- 位数可编程的（7 位、 8 位、 16 位和 32 位）的完全可编程多项式。
- 处理 8 位、 16 位、 32 位数据大小
- 可编程 CRC 初始值
- 单输入/输出 32 位数据寄存器
- 输入缓冲器可避免计算期间发生总线阻塞
- 对于 32 位数据大小， CRC 计算在 4 个 AHB 时钟周期 (HCLK) 内完成
- 8 位通用寄存器（可用于临时存储）
- I/O 数据的可逆性选项

5.3 CRC 功能说明

图 5-1 **CRC** 计算单元框图



CRC 计算单元具有单个 32 位读/写数据寄存器 (CRC_DR)。它用于输入新数据（写访问）和保存之前 CRC 计算的结果（读访问）。

对数据寄存器的每个写操作都会对之前的 CRC 值（存储在 CRC_DR 中）和新值再做一次 CRC 计算。 CRC 计算针对整个 32 位数据字或逐个字节完成，具体取决于数据的写入格式。

CRC_DR 寄存器可按字、右对齐半字和右对齐字节进行访问。对于其它寄存器，只允许进行 32 位访问。

计算时间取决于数据宽度：

- 32 位数据需要 4 个 AHB 时钟周期
- 16 位数据需要 2 个 AHB 时钟周期
- 8 位数据需要 1 个 AHB 时钟周期

输入缓冲器中可立即写入第二个数据，无需因之前的 CRC 计算而等待任何等待状态。

可动态调整数据大小，从而能最大程度地减少给定字节数的写访问次数。例如，对 5 个字节进行 CRC 计算时，可先写入字，然后写入字节。

输入数据的顺序可反转，以管理各种数据存放方式（双字/单字/字节，大端/小端等等）。可对 8 位、 16 位和 32 位数据执行反转操作，具体取决于 CRC_CR 寄存器中的 REV_IN[1:0] 位。

例如，输入数据 0x1A2B3C4D 在 CRC 计算中用作：

按字节执行位反转的 0x58D43CB2

按半字执行位反转的 0xD458B23C
按全字执行位反转的 0xB23CD458

通过将 CRC_CR 寄存器中 REV_OUT 位置 1 也可以将输出数据反转。

该操作按位进行：例如，输出数据 0x11223344 将转换为 0x22CC4488。

使用 CRC_CR 寄存器中的 RESET 控制位可将 CRC 计算器初始化为可编程值（默认值为 0xFFFFFFFF）。

可使用 CRC_INIT 寄存器对 CRC 初始值进行编程。对 CRC_INIT 寄存器进行写访问时会自动初始化 CRC_DR 寄存器。

CRC_IDR 寄存器可用于保存与 CRC 计算相关的临时值。它不受 CRC_CR 寄存器中的 RESET 位影响。

多项式可编程

多项式系数可完全通过 CRC_POL 寄存器进行编程，通过编程 CRC_CR 寄存器中的 POLYSIZE[1:0] 位可将多项式大小配置为 7 位、 8 位、 16 位或 32 位。不支持偶数多项式。

如果 CRC 数据小于 32 位，可从 CRC_DR 寄存器的最低有效位读取 CRC 的值。

为实现可靠的 CRC 计算， CRC 计算期间不能实时更改多项式的值或大小。因此，如果正在执行 CRC 计算，则在更改多项式前，应用程序必须先复位，或者先执行 CRC_DR 读操作。

默认的多项式值为 CRC-32（以太网）多项式： 0x4C11DB7。

5.4 CRC 寄存器

5.4.1 数据寄存器(CRC_DR)

地址偏移: 0x00
复位值: 0xFFFF FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DR[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
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
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

位 31:0	**DR:** 该寄存器用于接收待计算的新数据，直接将其写入即可。读取该寄存器得到的则是上次 CRC 计算的结果。如果数据不足 32 位，则读写到的正确数据只针对有意义的位。

5.4.2 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
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IDR[7:0]							
								rw							
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位 31:8	保留:必须保持复位值
位 7:0	IDR[7:0]: 通用目的的 8 位数据寄存器位 这些位可以当作一个字节的临时存储。该寄存器不受 CRC_CR 寄存器中的 RESET 位所引发的复位动作的影响。

5.4.3 控制寄存器 (CRC_CR)

地地址偏移: 0x8

复位值: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

位 31:5	保留:必须保持复位值
位 7	**REV_OUT:** 翻转输出数据 该位控制输出数据的翻转 0: 不翻转 1: 翻转
位 6:5	**REV_IN[1:0]:** 翻转输入数据 该位控制输入数据的翻转 00: 不翻转 01: 按字节为单位翻转 10: 按半字为单位翻转 11: 按字为单位翻转 ,
位 4:3	保留:必须保持复位值
位 2:1	保留:必须保持复位值
位 0	**RESET:** 复位控制，该位用来复位整个 CRC 计算单元，并将 CRC_INIT 寄存器中的值更新到当前计算状态中，由软件置位，由硬件清零。

5.4.4 CRC 初值寄存器 (CRC_INIT)

地址偏移: 0x10

复位值: 0xFFFF FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CRC_INIT [31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

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

位 31:0	**CRC_INIT:** CRC 初值预置，该寄存器用来设置 CRC 的初值。

6 电源控制 (PWR)

6.1 电源

当 BOR 可用时，器件的工作电压(VDD)要求介于 1.8V 到 3.6V 之间（掉电时下至 1.65V）。
当 BOR 不可用时，器件的工作电压(VDD)要求介于 1.65V 到 3.6V 之间。

嵌入式线性调压器用于提供内部数字电源，范围从 1.2V 到 1.8V。

- 当 BOR 可用时，V_{DD}=1.8V（上电时）或 1.65V（掉电时）到 3.6V。当 BOR 不可用时，V_{DD}=1.65 到 3.6V

  V_{DD} 是 I/O 和内部调压器的外部电源。通过 V_{DD} 引脚从外部提供
- V_{CORE}=1.2 到 1.8V

VCORE 是数字外设、SRAM 和 Flash 的电源。它由内部调压器产生。根据 VDD，可通过软件选 择三种 VCORE 范围（请参见图 11）。

- 当 BOR 可用时，V_{SSA}、V_{DDA}=1.8V（上电时）或 1.65V（掉电时）到 3.6V。当 BOR 不可用时，V_{SSA}、V_{DDA}=1.65V 到 3.6V。

  V_{DDA} 是 ADC、DAC、复位模块、RC 振荡器和 PLL 的外部模拟电源。使用 DAC 时，施加到 V_{DDA} 的最小电压为 1.8V。
- V_{REF+}

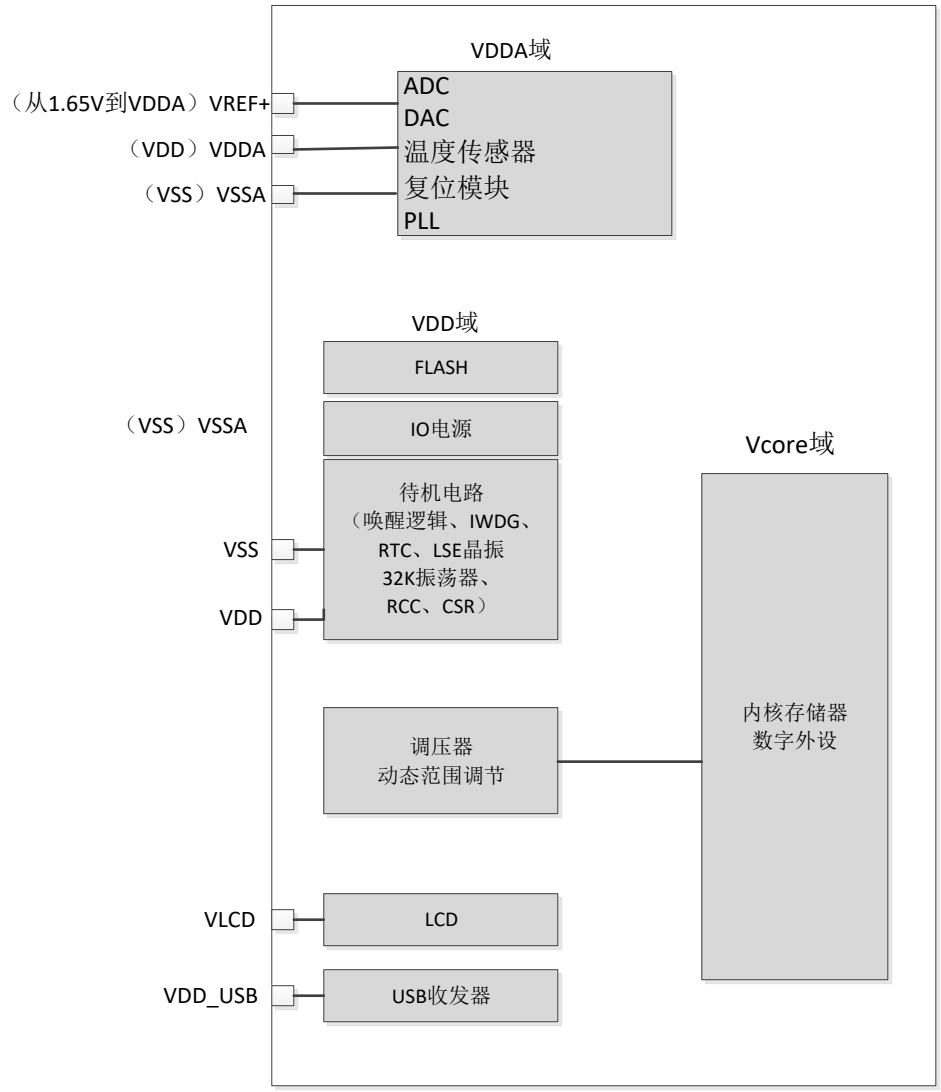
  V_{REF+}是输入参考电压。该电压仅在一些封装上用作外部引脚，否则它会连接到 V_{DDA}。
- V_{LCD}=2.5 到 3.6V

  LCD 控制器可通过 V_{LCD} 引脚外部供电，或由嵌入式升压转换器产生的内部电压内部供电。
- V_{DD_USB}=3.0 到 3.6V

  V_{DD_USB} 是为全速收发器供电的专用独立 USB 电源。如果将 PA11 和 PA12 引脚配置为 USB 复用功能，则可在这些引脚上提供该电源。

注：V_{DD_USB} 值与 V_{DD} 和 V_{DDA} 无关。但是，V_{DD_USB} 电源必须最后提供给器件并且最先关闭。当关闭三个电源时，如果 V_{DD_USB} 在短时间内保持激活并且 V_{DDA}/V_{DDIO} 低于功能范围，则器件不会损坏。当 V_{DD_USB} 关闭时，器件仍能工作。

图 10. 电源概述



1. VDDA 和 VSSA 必须分别连接到 VDD 和 VSS。
2. 根据使用的工作电压范围，某些外设可能只提供有限的功能或性能。
3. 仅在 TFBGA64 封装上提供 VREF+。

6.1.1 独立 A/D 和 DAC 转换器电源和参考电压

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

ADC 电压源从单独的 V_{DDA} 引脚输入。

V_{SSA} 引脚提供了独立的电源接地接。

在具有多于 64 个引脚和 UFBGA64 的封装上

为确保低电压输入和输出上的更好精度，用户可将 V_{REF+} 连接至一个独立的，低于 V_{DD} 的外部参考电压源。对于模拟输入(ADC)或输出(DAC)信号，V_{REF+} 为最高电压，以满量程值表 示。

对于 DAC：

$1.8V \leq V_{REF+} < V_{DDA}$

对于 ADC：

$1.65V \leq V_{REF+} < V_{DDA}$

在 64 引脚及以下的封装上（除了 BGA 封装）

V_{REF+}引脚不可用。它内部连接到ADC 电压源(V_{DA})。

6.1.2 独立 LCD 供电

V_{LCD} 引脚用于控制玻璃LCD 的对比度。可用两种方法使用这一引脚：

- 它可从外部电路接收所需的最大电压，由微控制器通过区段和公用线供给玻璃 LCD。
- 还可用它连接外部电容，微控制器将该电容用于内部的升压转换器。通过软件控制此升压转换器，以向玻璃LCD 的区段和公用线提供所需电压。

  向区段和公用线提供的电压定义了玻璃LCD 像素的对比度。当配置帧间死区时，可降低此对比度。
- 当向V_{LCD} 引脚提供外部电压时，电压范围应当为2.5 V 到3.6 V。它不取决于V_{DD}。
- 当LCD 基于内部升压转换器时，V_{LCD} 引脚应连接到电容（有关详细信息，请参见产品据手册）。

6.1.3 RTC 和 RTC 备份寄存器

实时时钟(RTC) 是一个独立的BCD 定时器/计数器。RTC 提供一个日历时钟、两个可编程闹钟中断，以及一个具有中断功能的可编程的周期唤醒标志。RTC 包含5 个备份数据寄存器。发生入侵检测事件时，将复位这些备份寄存器。有关详细信息，请参见*实时时钟(RTC)* 部分。

RTC 寄存器访问

复位后，RTC 寄存器（RTC 寄存器和RTC 备份寄存器）将受到保护，以防止可能的意外写访问。要使能对RTC 寄存器的访问，请按以下步骤操作：

1. 将RCC_APB1ENR 寄存器中的PWREN 位置1，使能电源接口时钟。
2. 将PWR_CR 寄存器中的DBP 位置1（请参见*第6.4.1 节*）。
3. 通过RCC_RCC 寄存器中的RTCSEL[1:0] 位选择RTC 时钟源。
4. 通过对RCC_CSR 寄存器中的RTCEN 位进行编程，使能RTC 时钟。

6.1.4 调压器

嵌入式线性调压器为待机电路以外的所有数字电路供电。调压器的输出电压(V_{CORE})可通过 软件编程为1.2 - 1.8 V（典型值）内的三个不同的范围（请参见*第6.1.5 节*）。

调压器在复位后始终处于使能状态。根据应用模式的不同，可采用三种不同的模式工作：主(MR) 模式、低功耗(LPR) 模式和掉电模式。

- 在运行模式下，调压器处于主(MR) 模式并为V_{CORE} 域（内核、存储器和数字外设）提供全功率。
- 低功耗运行模式下，调压器处于低功耗(LPR) 模式并为V_{CORE} 域提供低功率，保留寄部 SRAM 中的内容。
- 在睡眠模式下，调压器处于主(MR)模式并为 V_{CORE} 域提供全功率，保留寄存器和内部 SDRAM 中内容；
- 在低功耗睡眠模式下，调压器处于低功耗(LPR) 模式并为V_{CORE} 域提供低功率，保留寄存器和内部SRAM 中的内容。
- 在停止模式下，调压器为V_{CORE} 域提供低功率，保留寄存器和内部SRAM 中的内容。
- 在待机模式下，调压器掉电。除待机电路外，寄存器和SRAM 的内容都将丢失。

6.1.5 动态电压调节管理

动态电压调节是一种功率管理技术，可根据情况增加或减少用于数字外设的电压(V_{CORE})。

用于增加 V_{CORE} 的动态电压调节称为过电压。它可提高设备性能。有关器件工作状态与 CPU 性能关系的介绍，请参见图 11 ；有关 ADC 时钟频率与动态范围的关系，请参见数据手册的电气特性部分。

用于减少V_{CORE} 的动态电压调节称为欠电压。执行欠电压调节可以节省电能，特别是在电能来自电池并因此受限的笔记本电脑和其他移动设备中。

范围1

范围1 是“高性能”范围。

只要V_{DD} 输入电压超过2.1 V，电压调节器就会输出1.65V 电压（典型值）。可在此范围内执行Flash 编程和擦除操作。

仅当器件在范围1 中工作时，时钟恢复系统(CRS) 才可用（请参见*第8 节：时钟恢复系统(CRS)*）。

当 VDD 低于 2.0V 时，从初始到最终状态的 CPU 频率变化必须遵循以下条件：

•$f_{CPU最终} < 4xf_{CPU初始}$。

•此外，两次变化间必须有5 µs 的延迟。例如，要从4.2 切换到32 MHz，首先从4.2 切换到16 MHz，等待5 µs，然后从16 切换到32 MHz。

范围2 和3

也可对调压器进行编程，以输出经调整的1.5 V（典型值，范围2）或1.35 V（典型值，范围 3）而对V_{DD}（1.65 到3.6 V）没有任何限制。

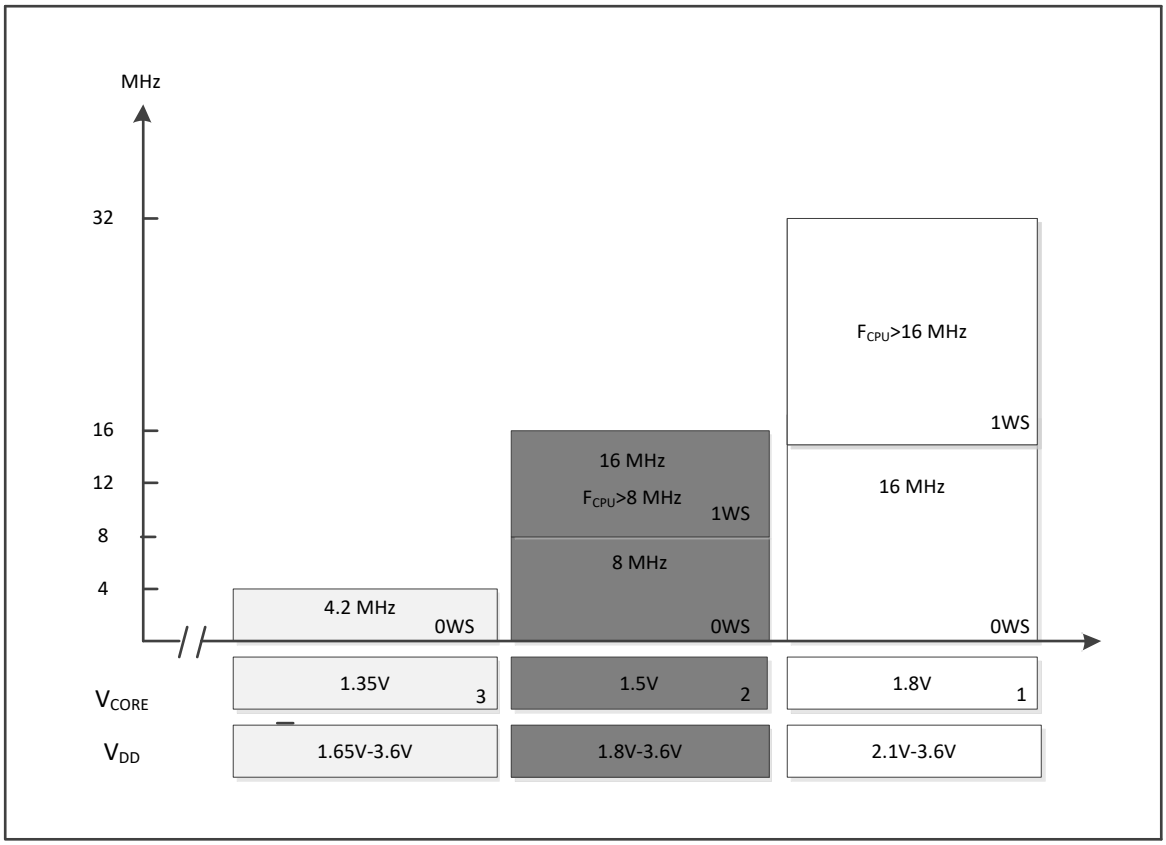
- 在 1.5 V 下，Flash 应可工作，但具有中等读取访问时间。这是“中等性能”范围。仍可对Flash 进行编程和擦除操作。
- 在 1.35 V 下，Flash 应可工作，但具有缓慢读取访问时间。这是“低性能”范围。在这些条件下，不能对Flash 进行编程和擦除操作。

有关各范围下性能的详细信息，请参见*表 30*。

表**30.** 性能与**V_{CORE}** 范围

CPU 性能	功耗性能	**V_{CORE}** 范围	典型值**(V)**	最大频率**(MHz)**		**V_{DD}** 范围
				1 WS	**0 WS**	
高	低	1	1.65	32	16	2.1-3.6
中	中	2	1.5	16	8	1.8-3.6
低	高	3	1.35	4.2	4.2	1.65-3.6

图 11. 性能与**V_{DD}** 和**V_{CORE}** 范围



6.1.6 动态电压调节配置

要编程调压器范围，需要按照以下顺序操作：

1. 检查V_{DD} 以确定允许哪些范围（请参见图11：性能与VDD 和VCORE 范围）。
2. 轮询PWR_CSR 中的VOSF 位。等到该位复位为0。
3. 通过设置PWR_CR 寄存器中的VOS[1:0] 位，配置电压调节范围。
4. 轮询PWR_CSR 寄存器中的VOSF 位。等到该位复位为0。

注： 在电压调节配置期间，系统时钟停止，直到调压器稳定(VOSF=0)。如果需要中断的关键响应时间，则应用开发过程中必须考虑这一点，具体取决于使用的外设（定时器、通信、……）。

6.1.7 当 VDD 降至 1.71 V 以下时的调压器和时钟管理

当选择 V_{CORE} 范围 1 并且 V_{DD} 降至 1.71V 以下时，应用程序必须重新配置系统。要重新配置系统，需要执行以下三步操作：

1. 检测 V_{DD} 电压是否低于 1.71V：
   使用 PVD 监视 V_{DD} 电压，当电压低于所选电平时生成中断。要检测 1.71V 电压限制，应用程序可以通过软件选择 PVD 阈值 2（典型值 2.26V）。有关 PVD 的详细信息，请参见*第 6.2.3 节*。
2. 根据将在下一步选择的电压范围调整时钟频率：
   低于 1.71V 时，系统时钟频率限制为 16MHz（针对范围 2）和 4.2MHz（针对范围 3）。
3. 选择所需电压范围：
   请注意，当 VDD 低于 1.71V 时，只能选择范围 2 或范围 3。

注：当选择 V_{CORE} 范围2 或范围3 并且 V_{DD} 降至1.71V 以下时，无需重新配置系统。

6.1.8 修改 VCORE 范围时的调压器和时钟管理

当 V_{DD} 高于 1.71V 时，可以选择 3 个电压范围中的任意一个。

- 当电压范围高于目标电压范围时（例如，从范围 1 到范围 2）：
  a) 根据将在下一步选择的更低电压范围调整时钟频率。
  b) 选择所需电压范围。
- 当电压范围低于目标电压范围时（例如，从范围 3 到范围 1）：
  a) 选择所需电压范围。
  b) 需要时调整时钟频率。

当 V_{DD} 低于 1.71V 时，只能选择范围 2 和范围 3：

- 从范围 2 到范围 3
  a) 根据电压范围 3 调整时钟频率。
  b) 选择电压范围 3。
- 从范围 3 到范围 2
  a) 选择电压范围 2。
  b) 需要时调整时钟频率。

6.1.9 当 VDD 介于 1.71V 到 2.0V 之间时的电压范围和限制

STM32L0x3 调压器基于针对超低功耗设计的架构。它不使用任何外部电容器。这种调压器对对负载的快速变化很敏感。在这种情况下，输出电压会在短时间内降低。考虑到核心电压必须高于 1.65V 以确保 32MHz 运行，这种现象对极低的 VDD 电压（如 1.71VVDD 最小值）至关重要。

要保证 V_{DD}=1.8V±5%（具有 1 个等待周期）和 V_{CORE} 范围 1 时的 32MHz 运行，必须对运行模式下的 CPU 频率进行管理以避免单触发中超过比率 4 的变化。两次变化间必须有 5μs 的延迟。从低功耗模式唤醒时没有限制。

6.2 电源监控器

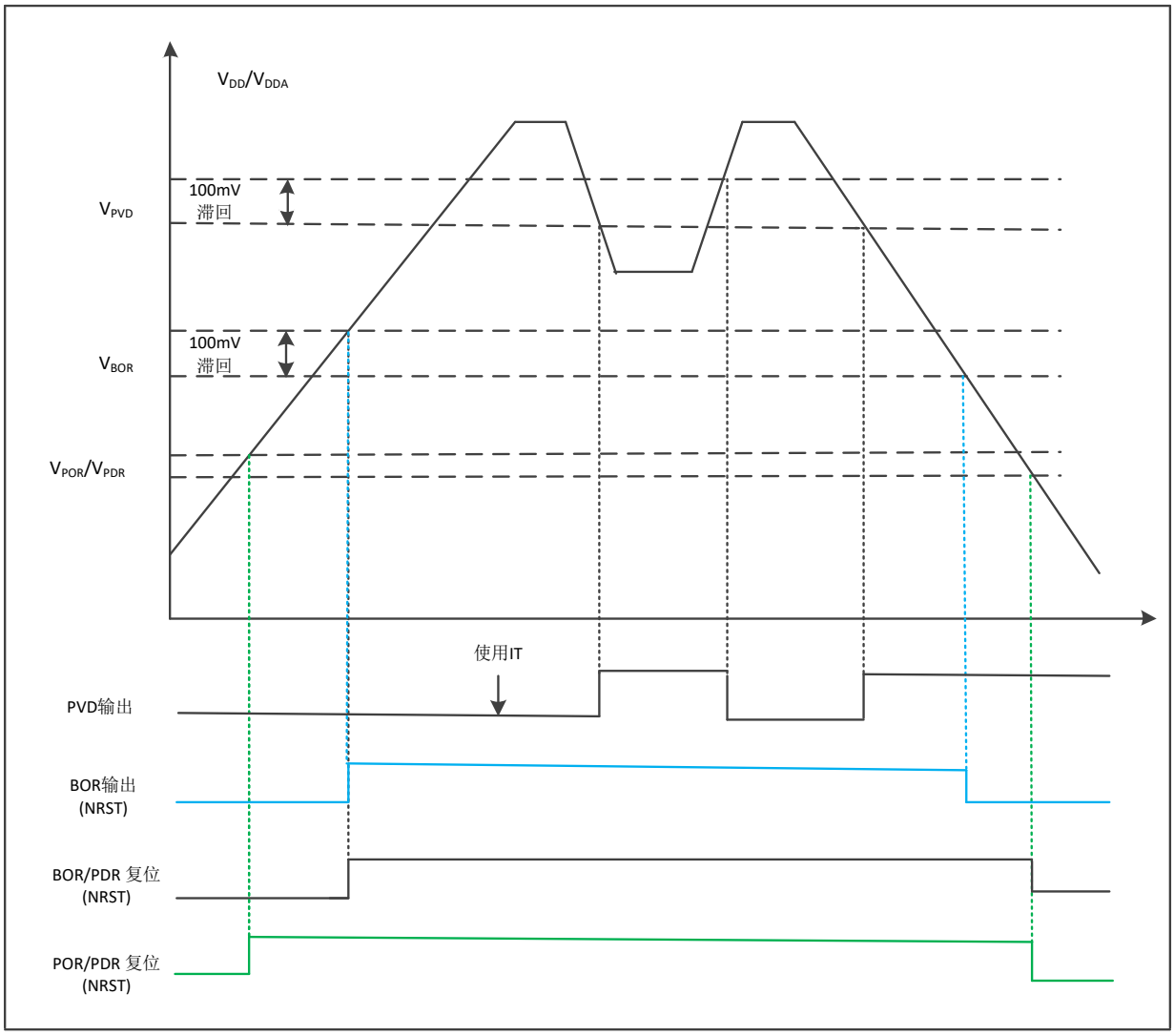
该器件具有一个集成的零功率上电复位(POR)/掉电复位(PDR)电路，与欠压复位(BOR)电路耦合。对于在 1.8 和 3.6V 之间工作的器件，BOR 在上电时始终激活，以确保从 1.8V 开始正常工作。达到 1.8VBOR 阈值之后，选项字节加载过程启动，以确认或修改默认阈值，或者永久禁止 BOR（在这种情况下，掉电时的 V_{DD} 最小值为 1.65V）。对于在 1.65V 和 3.6V 之间工作的器件，永久禁止 BOR。因此，上电时的启动时间通常可以减少到 1ms。

可以通过选项字节配置五个 BOR 阈值，从 1.65 到 3V。要减少停止模式下的功耗，可以自动关闭内部参考电压 V_{REFINT}。当 V_{DD} 低于指定阈值 V_{POR}、V_{PDR} 或 V_{BOR} 时，器件无需任何外部复位电路便可保持复位模式。

该器件还有一个嵌入式可编程电压检测器(PVD)，用于监视 V_{DD}/V_{DDA} 电源并将其与 V_{PVD} 阈值进行比较。可由软件选择 1.85V 和 3.05V 之间的 7 个不同的 PVD 电平，步长为 200mV。当 V_{DD}/V_{DDA} 低于 V_{PVD} 阈值和/或 V_{DD}/V_{DDA} 高于 V_{PVD} 阈值时，将产生中断。随后，中断服务程 序 会生成一条警告消息并且/或者使 MCU 进入安全状态。PVD 由软件使能。

图 12 中说明了不同的电源监控器（POR、PDR、BOR、PVD）。

图 12. 电源监控器



1. 所有器件上都有PVD，它可由软件使能或禁止。
2. BOR 仅在工作电压从1.8 到3.6 V 的器件上可用，除非被选项字节禁止，否则它会掩盖POR/PDR 阈值。
3. 如果通过选项字节禁止BOR，当V_{DD} 低于PDR 电平时，会发生复位。
4. 工作于1.65 到3.6 V 的器件没有 BOR，当 V_{DD} 高于 POR 电平时，会释放复位，当 V_{DD} 低于 PDR 电平时，会发生复位。

6.2.1 上电复位 (POR)/ 掉电复位 (PDR)

本器件内部集成有POR/PDR 电路，可以从1.5 V 开始工作。

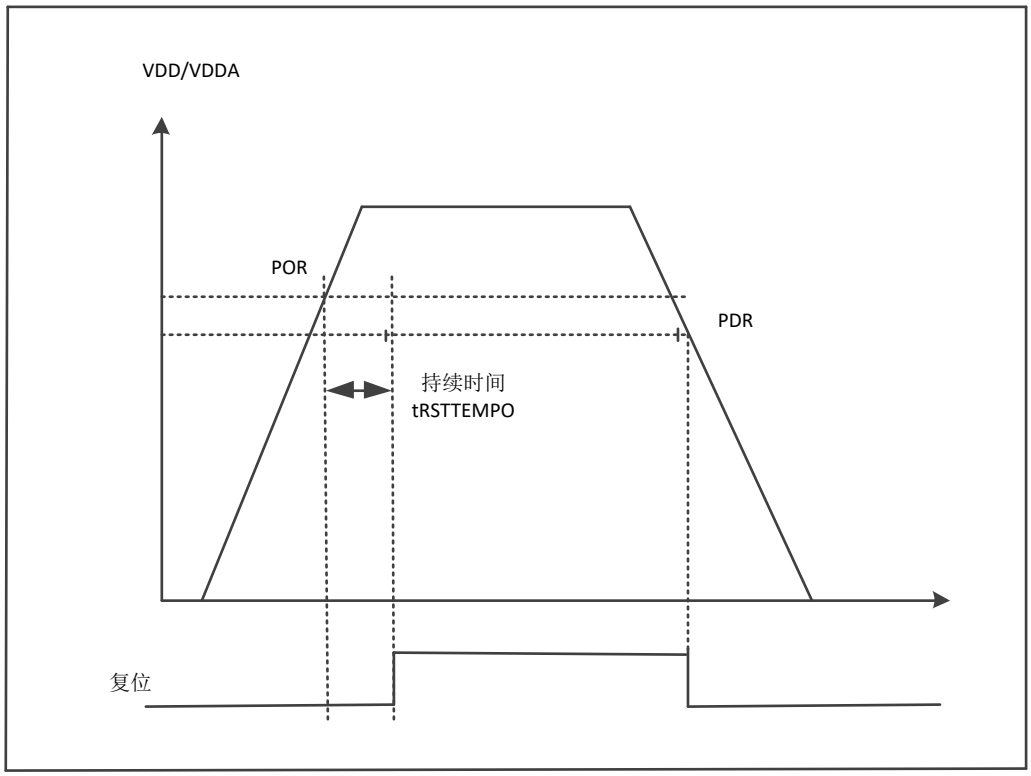
上电期间，当V_{DD}/V_{DDA}低于指定阈值V_{POR}时，器件无需外部复位电路便可保持复位模式。POR 功能始终使能，POR 阈值为1.5 V。

掉电期间，当电源电压(V_{DD}) 低于V_{PDR} 阈值时，PDR 将使器件保持复位状态。PDR 功能始终使能，PDR 阈值为1.5 V。

只有在禁止BOR 时才使用POR 和PDR（请参见*第6.2.2 节：欠压复位(BOR)*）。要确保最小工作电压 (1.65 V)，BOR 应配置为 BOR 级别 1。当禁止 BOR 时，最小工作电压 (1.65 V) 与V_{POR}/V_{PDR}阈值之间存在一个“盲区”。这意味着，V_{DD} 可低于1.65 V 并且器件不会复位，直至达到V_{PDR} 阈值。

有关上电/掉电复位阈值的相关详细信息，请参见数据手册的电气特性部分。

图 13. 上电复位/掉电复位波行



6.2.2 欠压复位 (BOR)

上电期间，欠压复位(BOR) 将使器件保持复位状态，直到电源电压达到指定的V_{BOR} 阈值。对于工作于1.65 到3.6 V 间的器件，BOR 选项不可用，电源供电由POR/PDR 监控。由于 POR/PDR 阈值为1.5 V，因此在V_{POR}/V_{PDR} 阈值和最低产品工作电压1.65 V 之间存在一个“盲区”。

对于工作在1.8 到3.6 V 之间的器件，BOR 在上电时始终激活，其阈值为1.8 V。

随后，当释放系统复位时，可通过加载选项字节重新配置或禁止BOR 级别。

若BOR 电平在开机时保持在最低电平1.8 V，在掉电时为1.65 V，则BOR 完全管理系统复位，产品工作电压处于安全范围内。

当通过选项字节禁止BOR 选项时，掉电复位由PDR 控制，在1.65 V 和V_{PDR} 之间存在“盲区”。

V_{BOR} 通过器件选项字节进行配置。默认情况下，激活4 级阈值。可以选择5 个可编程 V_{BOR} 阈值。

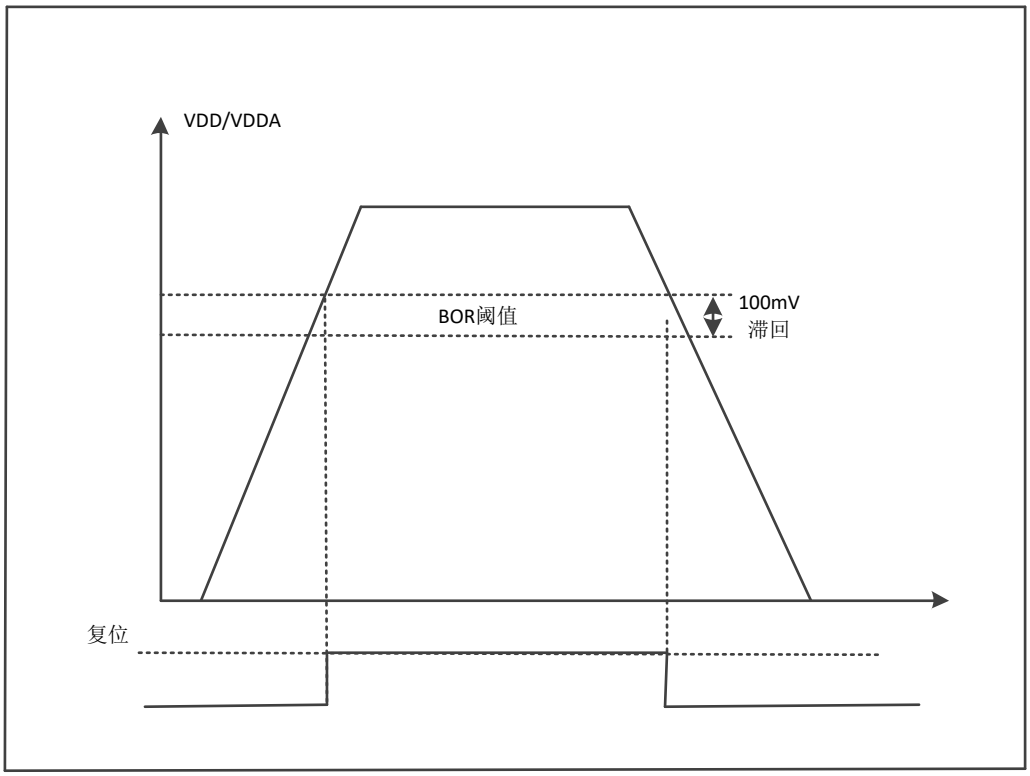
- BOR 级别1 (V_{BOR0})：1.69 V 到1.80 V 电压范围的复位阈值级别
- BOR 级别2 (V_{BOR1})：1.94 V 到2.1 V 电压范围的复位阈值级别
- BOR 级别3 (V_{BOR2})：2.3 V 到2.49 V 电压范围的复位阈值级别
- BOR 级别4 (V_{BOR3})：2.54 V 到2.74 V 电压范围的复位阈值级别
- BOR 级别5 (V_{BOR4})：2.77 V 到3.0 V 电压范围的复位阈值级别

当电源电压(V_{DD}) 降至所选V_{BOR} 阈值以下时，将使器件复位。当V_{DD} 高于V_{BOR} 上限时，释放器件复位，系统可以启动。

通过对器件选项字节进行编程可以禁止BOR。要禁止BOR 功能，V_{DD} 必须高于V_{BOR0}，以启动器件选项字节编程序列。POR 和PDR 将在随后监视上电和掉电（请参见*第6.2.1 节：上电复位 (POR)/掉电复位 (PDR)*）。

BOR 阈值滞回电压约为100 mV（电源电压的上升沿与下降沿之间）。

图**14. BOR** 阈值



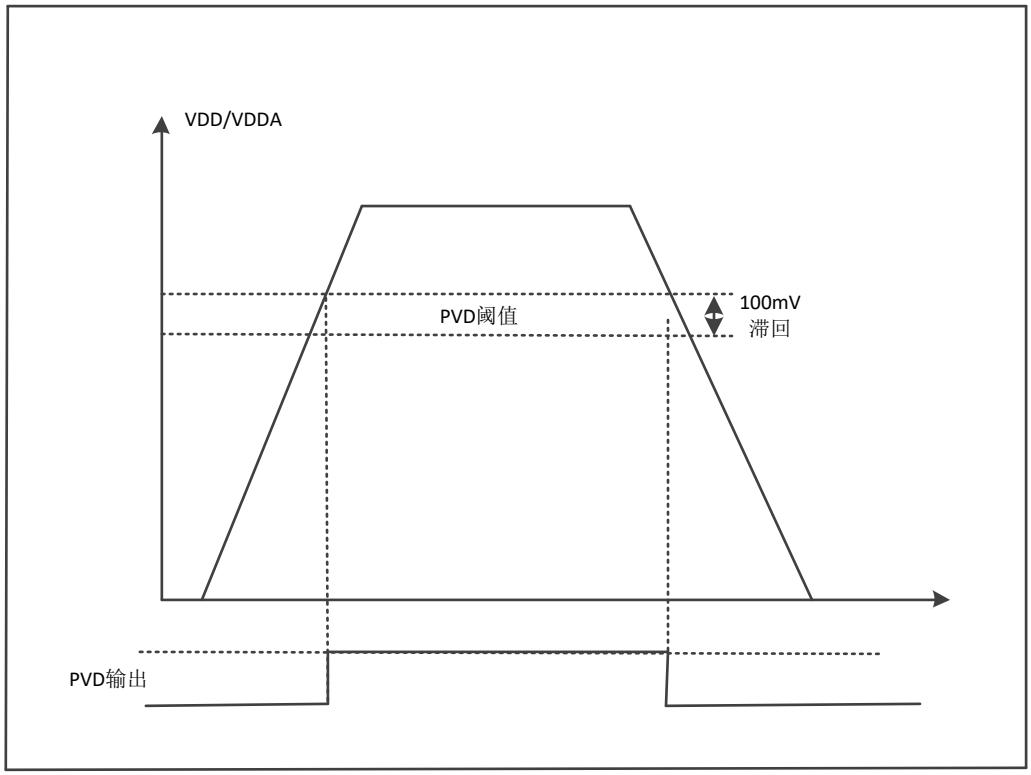
6.2.3 可编程电压检测器 (PVD)

可以使用 PVD 监视 V_{DD} 电源，方法是将其与 PWR_CR 中的 PLS[2:0] 位所选的阈值进行比较（请参见*第6.4.1 节*）。

PVD 可以使用外部输入模拟电压 (PVD_IN)，该电压在内部与 VREFINT 进行比较。 当 PLS[2:0] = 111 时，必须在模拟模式下配置 PVD_IN (PB7)。通过设置 PVDE 位来使能 PVD。

PWR_CSR 中提供了 PVDO 标志位（请参见*第6.4.2 节*），用于指示 V_{DD} 是大于还是小于 PVD 阈值。该事件内部连接到 EXTI 线 16，如果通过 EXTI 寄存器使能，则可以产生中断。当 V_{DD} 降至 PVD 阈值以下以及/或者当 V_{DD} 升至 PVD 阈值以上时，可以产生 PVD 输出中断，具体取决于 EXTI 线16 上升沿/下降沿的配置。该功能的用处之一就是可以在中断服务程序中执行紧 急关闭系统的任务。

图**15. PVD** 阈值



6.2.4 内部参考电压 (VREFINT)

内部参考电压(V_{REFINT}) 为模拟外设提供稳定的电压。通过内部参考电压(V_{REFINT}) 管理的功能包括 BOR、PVD、ADC、HSI48、LCD 和比较器。当使用这些功能之一时，始终使能内部参考电压(V_{REFINT})。

内部参考电压消耗不可忽略，尤其是在停止和待机模式下。为降低功耗，可以设置 PWR_CR 寄存器中的ULP 位（超低功耗）以禁止内部参考电压。不过，在这种情况下，当退出停止/待机模式时，在内部参考电压启动期间（最长3 ms）通过内部参考电压管理的功能不可靠。

停止/待机模式

要减少唤醒时间，器件可以退出停止/待机模式，而无需等待内部参考电压启动时间。为此，在进入停止/待机模式之前，在PWR_CR 寄存器中设置FWU 位（快速唤醒）。

如果 ULP 位置 1，则进入停止/待机模式之前使能的功能在这些模式下将被禁止，仅当内部参考电压启动时间结束后才重新使能，无论FWU 值如何。PWR_CSR 寄存器中的

VREFINTRDYF 标志指示内部参考电压已就绪。

当器件在NRST 脉冲上退出低功耗模式时，它不会等待内部参考电压启动（即使ULP = 1 且FWU=0）。必要时，应用程序应检查VREFINTRDYF 标志。

注： 当LCD 激活（LCD_CR 的LCDEN 位置1）时，需要VREFINT。因此ULP 位必须复位。

6.3 低功耗模式

丰富的功耗模式：

- RUN mode 下，动态功耗低至 62uA/MHz @3.3v
- 支持 SLEEP 睡眠模式. 动态功耗 50uA/MHz @3.3v ,且 唤醒时间仅 1 个机器时钟.
- 支持低功耗 SLEEP 模式
- 支持 LowPowerRUN 模式. 运行时仅需要几十 uA(T.B.D).
- 支持 STOP 停机模式， 静态功耗 450nA @3.3v ，最快 10uS 唤醒
- 支持 STANDBY 待机模式，静态功耗 200nA @3.3v ， 100uS 唤醒时间

丰富的唤醒源：

- SLEEP 模式下，可以由任何一个普通 IRQ 中断事件唤醒.
- LowPowerSleep 模式，可以由任何一个普通 IRQ 中断事件唤醒
- STOP 模式下， 可由任何一个 EXTI 外部中断线唤醒
- 同时，支持定时 ADC 采样预唤醒. 当满足条件后真正唤醒. (这个功能需要测试)
- 支持 DAC 输出保持,(这个功能需要测试)
- STANDBY 模式下，支持 1 个可配置极性的外部唤醒引脚 以及 RTC ALARM 唤醒.

HK32L08x 系列芯片支持多种功耗模式，可以在要求低功耗、短启动时间和多种唤醒事件之间达到最佳的平衡。

- Sleep 睡眠模式
  在睡眠模式，只有 CPU 停止，所有外设处于工作状态并可在发生中断/事件时唤醒 CPU。
- LowPowerSleep 睡眠模式
  在低功耗睡眠模式，只有 CPU 停止，调压器处于低功耗运行状态，所有外设处于工作状态并可在发生中断/事件时唤醒 CPU。
- LPR_RUN 模式
  LPR_RUN (LowPower RUN)模式为 L08x 新增的模式. 在此模式下，时钟主频不高于 128KHz (时钟源可以是 LSE , LSI 和 MSI 的 65K/128K 档位). 同时 LDO 切换为低功耗模式.
- Stop 停机模式
  在保持 SRAM 和寄存器内容不丢失的情况下，停机模式可以达到最低的电能消耗。在停机模式下，所有内部时钟被关闭，PLL、HSI 和 HSE 的 RC 振荡器被关闭。 可以通过任一配置成 EXTI 的信号把微控制器从停机模式中唤醒，EXTI 信号可以是 16 个外部 I/O 口之一、PVD 的输出、RTC 闹钟或 USB 的唤醒信号。
- Standby 待机模式
  在待机模式下可以达到最低的电能消耗。CPU 及主要数字逻辑被关闭，仅保留电源管理待机电路；PLL、HSI、HSE、 HSI、MSI、LSI 振荡器也被关闭；进入待机模式后，SRAM 和寄存器的内容将消失，但后备寄存器的内容仍然保留，待机电路仍工作。 从待机模式退出的条件是：NRST 上的外部复位信号、IWDG 复位、WKUP 管脚上的一个上升边沿或 RTC 的闹钟到时。

工作模式	功耗指标	唤醒时间
RUN mode	动态功耗低至 60uA/MHz @3.3v	
SLEEP 睡眠模式	动态功耗 50uA/MHz @3.3v	唤醒时间 1 个机器时钟
LPSLEEP mode		
LPRUN mode	动态功耗低至 50uA/MHz @3.3v	
STOP 停机模式	静态功耗 450nA @3.3v	最快 10uS 唤醒
STANDBY 待机模式	静态功耗 200nA @3.3v	150uS 唤醒时间

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

器件有五个低功耗模式：

- 低功耗运行模式：调压器处于低功耗模式，时钟频率受限，运行的外设数受限（请参见*第6.3.4 节*）
- 睡眠模式：Cortex®-M0+ 内核停止，外设保持运行（请参见*第6.3.7 节*）
- 低功耗睡眠模式：Cortex®-M0+ 内核停止，时钟频率受限，运行的外设数受限，调压器处于低功耗模式，Flash 停止（请参见*第6.3.8 节*）
- 停止模式：所有时钟停止，调压器运行，调压器处于低功耗模式（请参见*第 6.3.9 节*）
- 待机模式：V_{CORE} 域断电（请参见*第6.3.10 节*）

此外，可通过下列方法之一降低运行模式下的功耗：

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

表**31.** 低功耗模式汇总

模式名称	进入	唤醒	对V_{CORE} 域时钟的影响	对V_{DD} 域时钟的影响	调压器
低功耗运行	LPSDSR 和 LPRUN 位+ 时钟设置	强制调压器处于主调压器模式(1.8 V)	无	无	处于低功耗模式
睡眠（立即睡眠或退出时睡眠）	WFI 或从ISR 返回	任意中断	CPU CLK 关闭对其它时钟或模拟时钟源无影响	无	启动
	WFE	唤醒事件			
低功耗睡眠（立即睡眠或退出时睡眠）	LPSDSR 位和 LPRUN 位+ WFI 或从ISR 返 回	任意中断	CPU CLK 关闭对其它时钟或模拟时钟源无影响，Flash CLK 关闭	无	处于低功耗模式
	LPSDSR 位+ WFE	唤醒事件			
停止	PDDS、LPSDSR 位+ SLEEPDEEP 位+ WFI，从ISR 返回或WFE	任意EXTI 线（在EXTI 寄存器中配置，内部线和外部线）	所有V_{CORE} 域时钟都关闭	HSI16(1)、HSE 和MSI 振 荡器关闭	在低功耗模式下打开（取决于PWR_CR）
待机	PDDS 位+ SLEEPDEEP 位+ WFI，从ISR 返回或WFE	WKUP 引脚上升沿、RTC 闹钟（闹钟A 或闹钟B）、RTC 唤醒事件、RTC 入侵事件、RTC 时间戳事件、NRST 引脚外部复位、IWDG 复位			关闭

1. 如果在*时钟控制寄存器(RCC_CR)* 中设置HSI16KERON，HSI16 可在停止模式下运行。

6.3.1 低功耗模式下时钟的行为

可通过软件禁止 APB 外设和 DMA 时钟。

睡眠和低功耗睡眠模式

在睡眠和低功耗睡眠模式下CPU 时钟停止。睡眠期间可通过软件停止存储器接口（Flash 和 RAM 接口）时钟和所有外设时钟。在低功耗睡眠模式下，存储器接口时钟停止，RAM 处于掉电状态。当连接到AHB-APB 总线桥时钟的所有外设时钟均被禁止时，睡眠/低功耗睡眠模 式期间将通过硬件禁止AHB-APB 总线桥时钟。

停止和待机模式

在停止和待机模式下系统时钟和所有高速时钟都停止：

- 禁止PLL
- 禁止内部RC 16 MHz (HSI16) 振荡器
- 禁止外部1-24 MHz (HSE) 振荡器
- 禁止内部65 kHz - 4.2 MHz (MSI) 振荡器

通过中断退出此模式（停止模式）时，可以选择内部MSI 或HSI16 作为系统时钟。对于这两种振荡器，退出停止模式时保持其相应配置（范围和微调）。

通过复位退出此模式（待机模式）时，可以选择MSI 振荡器作为系统时钟。范围和微调值复位为默认值2.1 MHz。

如果正在执行Flash 编程操作或正在访问APB 域，停止/待机模式的进入将延迟到Flash 或 APB 访问完成后执行。

6.3.2 降低系统时钟速度

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

有关详细信息，请参见*第 7.3.4 节：时钟配置寄存器 (RCC_CFGR)*。

6.3.3 外设时钟门控

在运行模式下，可随时停止各外设和存储器的HCLK 和PCLKx 以降低功耗。要进一步降低睡眠模式的功耗，可在执行WFI 或WFE 指令之前禁止外设时钟。

外设时钟门控由AHB 外设时钟使能寄存器(RCC_AHBENR)、APB2 外设时钟使能寄存器 (RCC_APB2ENR) 和APB1 外设时钟使能寄存器(RCC_APB1ENR) 进行控制（参见 *第7.3.13 节：AHB 外设时钟使能寄存器(RCC_AHBENR)*、*第7.3.15 节：APB1 外设时钟使能寄存器 (RCC_APB1ENR)*和*第 7.3.14 节：APB2 外设时钟使能寄存器 (RCC_APB2ENR)*）。

在睡眠模式下，复位RCC_AHBLPENR 和RCC_APBxLPENR 寄存器（x 可以是1 或2）中的对应位可以自动禁止外设时钟。

6.3.4 低功耗运行模式（LP 运行）

当系统处于停止模式时，要进一步降低功耗，可以在低功耗模式下配置调压器。在此模式下，系统频率不应超过f_MSI 范围1。

有关调压器和外设工作条件的详细信息，请参见产品数据手册。

*注：为了能够在 APB1 时钟频率低于 RTC 时钟频率的七倍 (7*RTCLCK) 时读取 RTC 日历寄存器，软件必须分两次读取日历时间和日期寄存器。*

这样，当两次读取的 RTC_TR 结果相同时，才能确保数据正确。否则必须执行第三次读 访问。

仅当V_{CORE} 处于范围2 时，才能进入低功耗运行模式。此外，选择低功耗运行模式时，不能使用动态电压调节。选择低功耗运行模式时，只允许停止和睡眠模式以及在低功耗模式下配置的调压器。

注：　在低功耗运行模式下，所有I/O 引脚的状态与运行模式下相同。

进入低功耗运行模式

要进入低功耗运行模式，请执行下列操作：

1. 必须使用 RCC_APBxENR 和 RCC_AHBENR 寄存器使能或禁止每个数字 IP 时钟。
2. 系统时钟的频率必须降低到不超过 f_MSI 范围 1 的频率。
3. 通过软件强制调压器处于低功耗模式（LPRUN 和 LPSDSR 位置 1）。

退出低功耗运行模式

要退出低功耗运行模式，请执行下列操作：

1. 通过软件强制调压器处于主调压器模式。
2. 如果需要，打开Flash。
3. 可以增加时钟系统的频率。

6.3.5　进入低功耗模式

通过执行WFI（等待中断）或WFE（等待事件）指令，或者从ISR 返回时Cortex®-M0+ 系统时钟寄存器中的SLEEPONEXIT 位置1 时，即可进入低功耗模式（低功耗运行模式除外）。
仅当没有中断和事件挂起时，才会执行通过WFI 或WFE 进入低功耗模式。

6.3.6　退出低功耗模式

微控制器根据模式的进入方式退出睡眠和停止模式：

- 如果使用 WFI 指令或从 ISR 返回进入低功耗模式，通过 NVIC 确认的任何外设中断都能唤醒器件。这包括EXTI 线和任何GPIO 切换。
- 如果使用WFE 指令进入低功耗模式，微控制器将在有事件发生时立即退出低功耗模式。 唤醒事件可通过以下方式产生：
  – NVIC IRQ 中断：

通过以下操作实现：在外设的控制寄存器使能一个中断，但不在 NVIC 中使能，同时使能Cortex®-M0+ 系统控制寄存器中的SEVONPEND 位。当微控制器从WFE 恢复时，需要清除相应外设的中断挂起位和外设NVIC 中断通道挂起位（在NVIC 中断清除挂起寄存器中）。

– 事件：

通过以下操作实现：配置一个外部或内部EXTI 线为事件模式。当CPU 从WFE 恢复时，因为对应事件线的挂起位没有被置1，不必清除相应外设的中断挂起位或 NVIC 中断通道挂起位。

6.3.7　睡眠模式

睡眠模式下的 I/O 状态

在睡眠模式下，所有I/O 引脚的状态与运行模式下相同。

进入睡眠模式

根据*第 6.3.5　节：进入低功耗模式*进入睡眠模式。 有关如何进入睡眠模式的详细信息，请参见*表　32：立即睡眠*和*表　33：退出时睡眠*。

退出睡眠模式

根据*第 6.3.6　节：退出低功耗模式*进入睡眠模式。 有关如何进入睡眠模式的详细信息，请参见*表　32：立即睡眠*和*表　33：退出时睡眠*。

表**32.** 立即睡眠

立即睡眠模式	说明

进入模式	WFI（等待中断）或WFE（等待事件），且： – SLEEPDEEP = 0 及 – 没有中断（对于WFI）或事件（对于WFE）挂起 请参见Cortex®-M0+ 系统控制寄存器（请参见PM0223 编程手册）。
	从ISR 返回，且： – SLEEPDEEP = 0 及 – SLEEPONEXIT = 1 及 – 没有中断挂起 请参见Cortex®-M0+ 系统控制寄存器（请参见PM0223 编程手册）。
退出模式	如果使用WFI 或从ISR 返回进入： 中断：请参见*表 51：向量列表* 如果使用WFE 进入且SVONPEND = 0 唤醒事件：请参见*第 13.3.2 节：唤醒事件管理* 如果使用WFE 进入且SVONPEND = 1 中断事件（当在NVIC 中禁止时，请参见*表51：向量列表*）或唤醒事件（请参见*第13.3.2 节：唤醒事件管理*）
唤醒延迟	无

表**33.** 退出时睡眠

退出时睡眠	说明
进入模式	WFI（等待中断），且： – SLEEPDEEP = 0 及 – 没有中断（对于WFI）或事件（对于WFE）挂起 请参见Cortex®-M0+ 系统控制寄存器（请参见PM0223 编程手册）。
	从ISR 返回，且： – SLEEPDEEP = 0 及 – SLEEPONEXIT = 1 及 – 没有中断挂起 请参见Cortex®-M0+ 系统控制寄存器（请参见PM0223 编程手册）。
退出模式	中断：请参见*表 51：向量列表*
唤醒延迟	无

6.3.8 低功耗睡眠模式（LP 睡眠）

低功耗睡眠模式下的I/O 状态

在低功耗睡眠模式下，所有I/O 引脚的状态与运行模式下相同。

进入低功耗睡眠模式

要进入低功耗睡眠模式，请执行下列操作：

1. 可以使用FLASH_ACR 寄存器中的控制位SLEEP_PD 关闭Flash。这会降低功耗，但增加了唤醒时间。
2. 必须使用RCC_APBxENR 和RCC_AHBENR 寄存器使能或禁止每个数字IP 时钟。
3. 必须降低系统时钟的频率。
4. 通过软件强制调压器处于低功耗模式（LPSDSR 位置1）。
5. 执行*第 6.3.5 节：进入低功耗模式*中所述的步骤。

有关如何进入低功耗睡眠模式的详细信息，请参见*表 34：立即睡眠（低功耗睡眠）* 和*表 35：退出时睡眠（低功耗睡眠）*。

在低功耗睡眠模式下，可以关闭Flash，RAM 存储器保持可用。

在此模式下，系统频率不应超过f_MSI 范围1。

有关调压器和外设操作条件的详细信息，请参见产品数据手册。

仅当V_{CORE} 处于范围2 时，才能进入低功耗睡眠模式。

注： *为了能够在 APB1 时钟频率低于 RTC 时钟频率的七倍 (7*RTCLCK) 时读取 RTC 日历寄存器，软件必须分两次读取日历时间和日期寄存器。这样，当两次读取的RTC_TR 结果相同时，才能确保数据正确。否则必须执行第三次读访问。*

退出低功耗睡眠模式

根据第 6.3.6 节：退出低功耗模式退出低功耗睡眠模式。通过发出中断或唤醒事件退出低功耗睡眠模式时，在主调压器模式下配置调压器，Flash 关闭（如有必要），并且可以增加系统时钟。
当调压器在低功耗模式下工作时，将器件从低功耗睡眠模式唤醒将需要额外的启动延时。
有关如何退出低功耗睡眠模式的详细信息，请参见表 34：立即睡眠（低功耗睡眠） 和表 35：退出时睡眠（低功耗睡眠）。

表**34.** 立即睡眠（低功耗睡眠）

立即睡眠模式	说明
进入模式	调压器处于低功耗模式且Flash 关闭 WFI（等待中断）或WFE（等待事件），且： – SLEEPDEEP = 0 及 – 没有中断（对于WFI）或事件（对于WFE）挂起 请参见Cortex®-M0+ 系统控制寄存器（请参见PM0223 编程手册）。
	从ISR 返回，且： – SLEEPDEEP = 0 及 – SLEEPONEXIT = 1 及 – 没有中断挂起 请参见Cortex®-M0+ 系统控制寄存器（请参见PM0223 编程手册）。
退出模式	调压器处于主调压器模式且Flash 打开 如果使用WFI 或从ISR 返回进入： 中断：请参见*表 51：向量列表* 如果使用WFE 进入且SEVONPEND = 0 唤醒事件：请参见*第 13.3.2 节：唤醒事件管理* 如果使用WFE 进入且SVONPEND = 1 中断事件（当在NVIC 中禁止时，请参见*表51：向量列表*）或唤醒事件（请参见*第13.3.2 节：唤醒事件管理*）
唤醒延迟	调压器从低功耗模式唤醒的时间

表**35.** 退出时睡眠（低功耗睡眠）

退出时睡眠	说明
	WFI（等待中断），且： – SLEEPDEEP = 0 及 – 没有中断（对于WFI）或事件（对于WFE）挂起 请参见Cortex®-M0+ 系统控制寄存器（请参见PM0223 编程手册）。

进入模式	从ISR 返回，且： – SLEEPDEEP = 0 及 – SLEEPONEXIT = 1 及 – 没有中断挂起 请参见Cortex®-M0+ 系统控制寄存器（请参见PM0223 编程手册）。
退出模式	中断：请参见*表 51：向量列表*。
唤醒延迟	调压器从低功耗模式唤醒的时间

6.3.9 停止模式

停止模式基于Cortex®-M0+ 深度睡眠模式与外设时钟门控。调压器既可以配置为正常模式，也可以配置为低功耗模式。在停止模式下，V_{CORE} 域中的所有时钟都会停止，PLL、MSI、HSI16 和 HSE RC 振荡器也被禁止。内部SRAM 和寄存器内容将保留。

要使停止模式下的功耗最低，内部Flash 也进入低功耗模式。Flash 处于掉电模式时，将器件从停止模式唤醒将需要额外的启动延时。

要使停止模式下的功耗最低，可在进入停止模式前关闭 V_{REFINT}、BOR、PVD 和温度传感器。退出停止模式后，可以使用PWR_CR 寄存器中的ULP 位通过软件重新打开它们。

低功耗睡眠模式下的I/O 状态

在停止模式下，所有I/O 引脚的状态与运行模式下相同。

进入停止模式

有关如何进入停止模式的详细信息，请参见*第6.3.5 节：进入低功耗模式*和*表36*。

如果应用程序需要在进入停止模式之前禁止外部时钟，必须首先禁止HSEON 位并将系统时钟切换到HSI16。

否则，如果在进入停止模式前HSEON 位保持使能并且可以移除外部时钟（外部振荡器），则必须使能时钟安全系统(CSS) 功能以检测任何外部振荡器故障并避免进入停止模式时出现故障行为。

要进一步降低停止模式的功耗，可将内部调压器设置为低功耗模式。通过PWR_CR 寄存器中的LPSDSR 位进行配置（请参见*第6.4.1 节*）。

如果正在执行Flash 编程操作或正在访问APB 域，停止模式的进入将延迟到Flash 或APB 访问完成后执行。

在停止模式下，可以通过对各控制位进行编程来选择以下功能：

- 独立的看门狗 (IWDG)：IWDG 通过写入其密钥寄存器或使用硬件选项来启动。而且一旦启动便无法停止，除非复位。请参见*第25 节：独立看门狗(IWDG)*中的*第25.3 节： IWDG 功能说明*。
- 实时时钟(RTC)：通过RCC_CSR 寄存器中的RTCEN 位进行配置（请参见*第7.3.21 节*）。
- 内部RC 振荡器(LSI RC)：通过RCC_CSR 寄存器中的LSION 位进行配置。
- 外部32.768KHz 振荡器 (LSE OSC)：通过 RCC_CSR 寄存器中的 LSEON 位进行配置。

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

退出停止模式

有关如何退出停止模式的详细信息，请参见*第6.3.6 节：退出低功耗模式*和*表36*。通过发出中断或唤醒事件退出停止模式时，将根据RCC_CFGR 寄存器中的STOPWUCK 位

选择MSI 或HSI16 RC 振荡器作为系统时钟。

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

表**36.** 停止模式

停止模式	说明
进入模式	WFI（等待中断）或WFE（等待事件），且： – 没有中断（对于WFI）或事件（对于WFE）挂起。 – 将Cortex®-M0+ 系统控制寄存器中的SLEEPDEEP 位置1 – 电源控制寄存器(PWR_CR) 中的PDDS 位= 0 – 电源控制/状态寄存器(PWR_CSR) 中的WUF 位= 0 – 通过配置 RCC_CFGR 寄存器中的 STOPWUCK 位退出停止模式时，选择MSI 或HSI16 RC 振荡器作为系统时钟。 *注：要进入停止模式，所有EXTI 线挂起位（在第13.5.6 节：EXTI 挂起寄存器(EXTI_PR) 中）、所有外设中断挂起位、RTC 闹钟（闹钟A 和闹钟B）、RTC 唤醒、RTC 入侵和RTC 时间戳标志位必须复 位。否则将忽略进入停止模式这一过程，继续执行程序。*
	从ISR 返回，且： – 没有中断挂起。 – 将Cortex®-M0+ 系统控制寄存器中的SLEEPDEEP 位置1 – SLEEPONEXIT = 1 – 电源控制寄存器(PWR_CR) 中的PDDS 位= 0 – 电源控制/状态寄存器(PWR_CSR) 中的WUF 位= 0 – 通过配置 RCC_CFGR 寄存器中的 STOPWUCK 位退出停止模式时，选择MSI 或HSI16 RC 振荡器作为系统时钟。 *注：要进入停止模式，所有EXTI 线挂起位（在第13.5.6 节：EXTI 挂起寄存器(EXTI_PR) 中）、所有外设中断挂起位、RTC 闹钟（闹钟A 和闹钟B）、RTC 唤醒、RTC 入侵和RTC 时间戳标志位必须复 位。否则将忽略进入停止模式这一过程，继续执行程序。*
退出模式	如果使用WFI 或从ISR 返回进入： 任何配置为中断模式的EXTI 线（必须在NVIC 中使能对应的EXTI 中断向量）。请参见*表 51：向量列表。* 如果使用WFE 进入且SEVONPEND = 0 任何配置为事件模式的EXTI 线。请参见*第267 页上的第13.3.2 节：唤醒事件管理*。 如果使用WFE 进入且SEVONPEND = 1 – 任何配置为事件模式的 EXTI 线（即使在 NVIC 中禁止对应的 EXTI 断）。中断源可以是外部中断或具有唤醒功能的外设（请参见*表51：向量列表*）。 – 唤醒事件（请参见*第267 页上的第13.3.2 节：唤醒事件管理*）
唤醒延迟	MSI 或HSI16 RC 唤醒时间+ 调压器从低功耗模式唤醒的时间+ FLASH 唤醒时间

6.3.10 待机模式

待机模式下可达到最低功耗。待机模式基于 Cortex®-M0+ 深度睡眠模式，其中调压器被禁止。因此V_{CORE} 域断电。PLL、MSI、HSI16 振荡器和HSE 振荡器也将关闭。除RTC 寄存器、RTC 备份寄存器和待机电路外，SRAM 和寄存器内容都将丢失（参见*图10*）。

待机模式下的 I/O 状态

在待机模式下，除以下各部分以外，所有I/O 引脚都处于高阻态：

- 复位脚
- 唤醒引脚（WKUP1、WKUP2、WKUP3）
- 以下I/O 上的RTC 功能（入侵、时间戳、RTC 闹钟输出、RTC 时钟校准输出）：
  - 类别3：PC13、PA0
  - 类别5：PC13、PA0、PE6

进入待机模式

有关如何进入待机模式的详细信息，请参见*第 6.3.5 节：进入低功耗模式*和*表 37*。在待机模式下，可以通过对各控制位进行编程来选择以下功能：

- 独立的看门狗 (IWDG)：IWDG 通过写入其密钥寄存器或使用硬件选项来启动。而且一旦启动便无法停止，除非复位。请参见*第 580 页上的第 25.3 节：IWDG 功能说明*。
- 实时时钟(RTC)：通过RCC_CSR 寄存器中的RTCEN 位进行配置（请参见*第7.3.21 节*）。
- 内部RC 振荡器(LSI RC)：通过RCC_CSR 寄存器中的LSION 位进行配置。
- 外部 32.768 kHz 振荡器 (LSE OSC)：通过 RCC_CSR 寄存器中的 LSEON 位进行 配置。

退出待机模式

检测到外部复位（NRST 引脚）、IWDG 复位、WKUP 引脚（WUKP1、WKUP2 或WKUP3）上升沿、RTC 闹钟、入侵事件或时间戳事件时，微控制器退出待机模式。从待机模式唤醒后，除 *PWR 电源控制/ 状态寄存器 (PWR_CSR)* 外，所有寄存器都将复位。

从待机模式唤醒后，程序将按照复位（启动引脚采样、复位向量已获取等）后的方式重新执行。PWR_CSR 寄存器（请参见*第6.4.2 节*）中的SBF 状态标志指示MCU 已处于待机模式。

有关如何退出待机模式的详细信息，请参见*第 6.3.6 节：退出低功耗模式*和*表 37*。

表**37.** 待机模式

待机模式	说明
进入模式	WFI（等待中断）或WFE（等待事件），且： – Cortex®-M0+ 系统控制寄存器中的SLEEPDEEP 位= 1 – 电源控制寄存器(PWR_CR) 中的PDDS 位= 1 – 没有中断（对于WFI）或事件（对于WFE）挂起。 – 电源控制/状态寄存器(PWR_CSR) 中的WUF 位= 0 – 将与所选唤醒源（RTC 闹钟A、RTC 闹钟B、RTC 唤醒、入侵或时间戳标志）对应的RTC 标志清零
	从ISR 返回，且： – Cortex®-M0+ 系统控制寄存器中的SLEEPDEEP 位= 1 – SLEEPONEXIT = 1 – 电源控制寄存器(PWR_CR) 中的PDDS 位= 1 – 没有中断挂起 – 电源控制/状态寄存器(PWR_CSR) 中的WUF 位= 0 – 将与所选唤醒源（RTC 闹钟A、RTC 闹钟B、RTC 唤醒、入侵或时间戳标志）对应的RTC 标志清零
退出模式	WKUP 引脚上升沿、RTC 闹钟（闹钟A 和闹钟B）、RTC 唤醒事件、RTC 入侵事件、RTC 时间戳事件、NRST 引脚外部复位和IWDG 复位。
唤醒延迟	复位阶段

调试模式

默认情况下，如果使用调试功能时应用程序将MCU 置于停止模式或待机模式，调试连接将中断。这是因为Cortex®-M0+ 内核时钟停止了。

不过，通过设置DBG_CR 寄存器中的一些配置位，即使MCU 进入低功耗模式，仍可使用软件对其进行调试。有关详细信息，请参见*第 33.9.1 节：对低功耗模式的调试支持*。

6.3.11 使用 RTC 和比较器从停止和待机模式唤醒器件

可以通过RTC 闹钟事件、RTC 唤醒事件、入侵事件、时间戳事件或比较器事件将MCU 从低功耗模式唤醒，而不依赖于外部中断（自动唤醒模式）。

这些 RTC 复用功能可将系统从停止和待机低功耗模式唤醒，而比较器事件只能将系统从停止模式唤醒。

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

RTC 提供了可编程时基，便于定期从停止或待机模式唤醒器件。为此，通过对 RCC_CSR 寄存器（请参见*第7.3.21 节*）中的RTCSEL[1:0] 位进行编程，可以选择三个复用RTC 时钟 源中的其中两个：

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

从停止模式的**RTC** 自动唤醒**(AWU)**

- 要通过RTC 闹钟事件从停止模式唤醒，必须：
  a) 将EXTI 线17 配置为检测外部信号的上升沿（中断或事件模式）
  b) 使能RTC_CR 寄存器中的RTC 闹钟中断
  c) 配置RTC 以生成RTC 闹钟
- 要通过RTC 入侵事件或时间戳事件从停止模式唤醒，必须：
  a) 将EXTI 线19 配置为检测外部信号的上升沿（中断或事件模式）

b) 使能RTC_CR 寄存器中的RTC 时间戳中断，或者使能RTC_TCR 寄存器中的RTC 入侵中断

c) 配置RTC 以检测入侵事件或时间戳事件

- 要通过RTC 唤醒事件从停止模式唤醒，必须：
  a) 将EXTI 线20 配置为检测外部信号的上升沿（中断或事件模式）
  b) 使能RTC_CR 寄存器中的RTC 唤醒中断
  c) 配置RTC 以生成RTC 唤醒事件

从待机模式的**RTC** 自动唤醒**(AWU)**

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

从停止模式的比较器自动唤醒**(AWU)**

- 要通过比较器1 或比较器2 唤醒事件从停止模式唤醒，必须：

a) 将EXTI21（硬连线到比较器1）和EXTI22（硬连线到比较器2）（中断或事件模式）配置为检测选定沿（下降、上升或下降和上升）

b) 配置比较器以生成事件

6.4 PWR 寄存器

6.4.1 电源控制寄存器 (PWR_CR)

偏移地址： 0x00

复位值： 0x0000 0000 (从待机模式唤醒时清除)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	LPRUN	Res.	VOS		FWU	ULP	DBP	PLS			PVDE	CSBF	CWUF	PDDS	LPSDSR
rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	w1	w1	rw	rw
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位 31:15	保留，必须保持为复位值。
位 14	**LPRUN:** 低功耗运行模式 (Low-power run mode) 当 LPRUN 位与 LPSDSR 位一起置 1 时，调压器从主模式切换到低功耗模式。否则，它将保持在主模式。当 LPRUN 复位时，调压器返回到在主模式下工作。 如果该位置 1（LPSDSR 位也置 1）且 CPU 进入睡眠或深度睡眠模式（LP 睡眠或停止模式），则当 CPU 从这些模式唤醒时，它将进入运行模式，但 LPRUN 位会置 1。要重新进入低功耗运行模式，必须执行复位并将 LPRUN 位重新置 1。 当 MCU 处于低功耗运行模式时，禁止复位 LPS DSR。 LPSDSR 用作进入低功率模式的前置条件，向系统指示进入低功率模式时将选择调压器的哪个配置。在 LPRUN 位置 1 前 LPSDSR 位必须置 1。仅当 LPRUN 位 = 0 时才能复位 LPSDSR。 0：低功耗运行模式下调压器进入主模式 1：低功耗运行模式下调压器进入低功耗模式
位 13	保留，必须保持为复位值。
位 12:11	**VOS:** 选择电压调节范围 (Voltage scaling range selection) 这些位用于选择内部调压器电压范围。 在通过复位 RCC_APB1RSTR 寄存器中的 PWRRST 位复位电源接口之前，这些位必须设置为 ‘10’ 并且必须相应地配置系统频率。 00：禁止（位保持不变，并保留以前的值，没有发生电压变化） 01： 1.8 V（范围 1） 10： 1.5 V（范围 2） 11： 1.2 V（范围 3）
位 10	**FWU:** 快速唤醒 (Fast wakeup) 此位与 ULP 位结合使用。 如果 ULP = 0，则忽略 FWU 如果 ULP = 1 且 FWU = 1：从低功耗模式退出时忽略 VREFINT 启动时间。当 VREFINT 重新就 绪时，通过 PWR_CSR 寄存器中的 VREFINTRDYF 标志加以指示。 如果 ULP = 1 且 FWU = 0：仅当 VREFINT 就绪（其启动时间后）时， 才会从低功耗模式退出。通过复位 RCC_APB1RSTR 寄存器中的 PWRRST 位不会复位该位。 0：仅当 VREFINT 就绪时，才会退出低功耗模式 1：退出低功耗模式时忽略 VREFINT 启动时间

位 9	**ULP:** 超低功耗模式 (Ultra-low-power mode) 置 1 时， VREFINT 在低功耗模式下关闭。通过复位 RCC_APB1RSTR 寄存器中的 PWRRST 位不会复位该位。当该位置 1 时，寄存器 LCD_CR 的 LCDEN 位不能置 1。 0： VREFINT 在低功耗模式下打开 1： VREFINT 在低功耗模式下关闭
位 8	**DBP:** 禁止备份写保护 (Disable backup write protection) 在复位状态下， RTC、 RTC 备份寄存器和 RCC CSR 寄存器均受到保护，以防止寄生写访问。必须将此位置 1 才能使能对这些寄存器的写访问。 0：禁止访问 RTC、 RTC 备份寄存器和 RCC CSR 寄存器 1：允许访问 RTC、 RTC 备份寄存器和 RCC CSR 寄存器 注： 如果被 2、 4、 8 或 16 分频的 HSE 用作 RTC 时钟，该位必须保持置 1。
位 7:5	**PLS:** PVD 电平选择 (PVD level selection) 这些位由软件写入，用于选择电源电压检测器检测的电压阈值： 000： 1.9 V 001： 2.1 V 010： 2.3 V 011： 2.5 V 100： 2.7 V 101： 2.9 V 110： 3.1 V 111：外部输入模拟电压（内部与 VREFINT 进行比较） 当 PLS[2:0] = 111 时， PVD_IN 输入 (PB7) 必须配置为模拟输入。 注： 有关详细信息，请参见数据手册的电气特性。
位 4	**PVDE:** 使能电源电压检测器 (Power voltage detector enable) 此位由软件置 1 和清零。 0：禁止 PVD 1：使能 PVD
位 3	**CSBF:** 将待机标志清零 (Clear standby flag) 此位始终读为 0。 0：不起作用 1：将 SBF 待机标志位清零（写）。
位 2	**CWUF:** 将唤醒标志清零 (Clear wakeup flag) 此位始终读为 0。 0：不起作用 1： 2 个系统时钟周期后将 WUF 唤醒标志清零
位 1	**PDDS:** 掉电深度睡眠 (Power-down deepsleep) 此位由软件置 1 和清零。 0：器件在 CPU 进入深度睡眠时进入停止模式。调压器处于低功耗模式。 1：器件在 CPU 进入深度睡眠时进入待机模式。
位 0	**LPSDSR:** 低功耗深度睡眠/睡眠/低功率运行 (Low-power deepsleep/Sleep/Low-power run) – DeepSleep/Sleep 模式 如果该位置 1，当 CPU 进入睡眠或深度睡眠模式时稳压器切换到低功耗模式。 CPU 从这些模式退出时，调压器返回到主模式。 – 低功耗运行模式 如果该位置 1，当 LPRUN 位置 1 时稳压器切换到低功耗模式。 LPRUN 位复位时，调压器返回到主模式。 此位由软件置 1 和清零。 0：深度睡眠/睡眠/低功率运行模式下调压器打开 1：深度睡眠/睡眠/低功率运行模式下调压器进入低功耗模式

6.4.2 电源控制/状态寄存器 (PWR_CSR)

偏移地址： 0x04

复位值： 0x0000 0000 (从待机模式下唤醒时该值不被清除)
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
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

位 31:10	保留，必须保持为复位值。
位 9	**EWUP2:** 使能 WKUP 引脚 2 (Enable WKUP pin 2) 此位由软件置 1 和清零。 0： WKUP 引脚 2 用于通用 I/O。 WKUP 引脚 2 上的事件不会将器件从待机模式唤醒。 1： WKUP 引脚 2 用于从待机模式唤醒器件并被强制配置成输入下拉（WKUP 引脚 2 出现上升沿时从待机模式唤醒系统）。 注： 此位通过系统复位进行复位。
位 8	**EWUP1:** 使能 WKUP 引脚 1 (Enable WKUP pin 1) 此位由软件置 1 和清零。 0： WKUP 引脚 1 用于通用 I/O。 WKUP 引脚 1 上的事件不会将器件从待机模式唤醒。 1： WKUP 引脚 1 用于从待机模式唤醒器件并被强制配置成输入下拉（WKUP 引脚 1 出现上升沿时从待机模式唤醒系统）。 注： 此位通过系统复位进行复位。
位 7:6	保留，必须保持为复位值。
位 5	**REGLPF:** 调压器 LP 标志 (Regulator LP flag) MCU 处于低功耗运行模式时，此位由硬件置 1。 当 MCU 从低功耗运行模式退出时，该位保持为 1，直到调压器在主模式下准备就绪。建议轮询此位以等待调压器主模式。当调压器准备就绪后，此位由硬件复位。 0：调压器在主模式下准备就绪 1：调压器处于低功耗模式
位 4	**VOSF:** 电压调节选择标志 (Voltage Scaling select flag) 电压范围变更后，内部调压器要准备好需要一定的延时。 VOSF 位指示调压器已达到通过 PWR_CR 寄存器的 VOS 位定义的电压电平。 当 PWR_CR 寄存器中的 VOS[1:0] 变化时，该位复位。调压器准备好后，该位置 1。 0：调压器在所选电压范围内准备就绪 1：调压器输出电压更改为所需 VOS 电平。
位 3	**VREFINTRDYF:** 内部参考电压 (VREFINT) 就绪标志 (Internal voltage reference (VREFINT) ready flag) 该位指示内部参考电压 VREFINT 的状态。 0： VREFINT 关闭 1： VREFINT 就绪
位 2	**PVDO:** PVD 输出 (PVD output) 此位通过硬件置 1 和清零。仅当通过 PVDE 位使能 PVD 时此位才有效。 0： VDD 高于 PLS[2:0] 位选择的 PVD 阈值。 1： VDD 低于 PLS[2:0] 位选择的 PVD 阈值。 注： PVD 在进入待机模式时停止。因此，进入待机模式或执行复位后，此位等于 0，直到 PVDE 位置 1。

位 1	**SBF:** 待机标志 (Standby flag) 此位由硬件置 1，清零则只能通过 POR/PDR（上电复位/掉电复位）或将 PWR 电源控制寄存器 (PWR_CR) 中的 CSBF 位置 1 来实现。 0：器件未进入待机模式 1：器件已进入待机模式
位 0	**WUF:** 唤醒标志 (Wakeup flag) 该位通过硬件置 1，并且只能通过系统复位或通过将 PWR 电源控制寄存器 (PWR_CR) 中的 CWUF 位置 1 清零 0：未发生唤醒事件 1：收到唤醒事件，可能来自 WKUP 引脚、 RTC 闹钟（闹钟 A 和闹钟 B）、 RTC 入侵事件、RTC 时间戳事件或 RTC 唤醒事件。 注： 如果使能 WKUP 引脚（将 EWUPx (x=1, 2, 3) 位置 1）时 WKUP 引脚已为高电平，系统将检测到另一唤醒事件。

6.4.3 WKUP 引脚极性控制寄存器 (PWR_WUP_POL)

偏移地址 ： 0x34

复位值 ： 0x0000 0000 (从待机模式下唤醒时该值不被清除)

与标准的 APB 读相比，读此寄存器需要额外的 APB 周期

NOTE ： 当对应的 EWUPx=1 时，禁止设置对应的寄存器.只有当对应的 EWUPx == 0 时，才能设置相应的寄存器.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

位 31:2	保留，必须保持为复位值。
位 1	**WUPOL2:** WKUP2 唤醒极性选择，该位用软件设置或清除。（EWUP2=1 时有效） 0: WKUP2 引脚被强置为输入下拉的配置 (WKUP2 引脚上的上升沿将系统从待机模式唤醒) 1: WKUP2 引脚被强置为输入上拉的配置 (WKUP2 引脚上的下降沿将系统从待机模式唤醒)
位 0	**WUPOL1:** WKUP1 唤醒极性选择，该位用软件设置或清除。（EWUP1=1 时有效） 0: WKUP1 引脚被强置为输入下拉的配置 (WKUP1 引脚上的上升沿将系统从待机模式唤醒) 1: WKUP1 引脚被强置为输入上拉的配置 (WKUP1 引脚上的下降沿将系统从待机模式唤醒)

6.4.4 低功耗模式下 DAC 控制寄存器 (PWR_DAC_LP_CTL)

偏移地址 ： 0x40

复位值 ： 0x0000 0000 (从待机模式下唤醒时该值不被清除)

与此寄存器只允许按照 Halfword/word 方式写入， 尝试读取只会返回 0x00000000

通过设置此寄存器， 将在 Standby 模式下关闭掉电检测 PDR， 以进一步降低待机功耗。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Res.	Res.	Res.	Res.	Res.	Res.	Res.	DAC2_ALP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DAC1_ALP
							rw								rw
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位 31:9	保留，必须保持为复位值。
位 8	**DAC2_ALP:** DAC2 在低功耗模式下处于活动状态 (DAC2 Alive in LP mode) 当使能了 LSI 之后: 0 : DAC2 在 STOP/Standby 不会保持输出，处于高阻 1 : DAC2 在 STOP/Standby 保持输出.
位 7:1	保留，必须保持为复位值。
位 0	**DAC1_ALP:** DAC1 在低功耗模式下处于活动状态 (DAC1 Alive in LP mode) 当使能了 LSI 之后: 0 : DAC1 在 STOP/Standby 不会保持输出，处于高阻 1 : DAC1 在 STOP/Standby 保持输出.

6.4.5 低功耗模式下 DAC 配置寄存器 (PWR_DAC_LP_CFG)

偏移地址： 0x44

复位值： 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RELOAD2				DUTY2				RELOAD1				DUTY1			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位 31:16	保留，必须保持为复位值。
位 15:12	RELOAD2: DAC2 在低功耗模式下的加载值 用于构造 DAC2 在低功耗模式下的输出
位 11:8	DUTY2: DAC2 在低功耗模式下的占空比值 用于构造 DAC2 在低功耗模式下的输出
位 7:4	RELOAD1: DAC1 在低功耗模式下的加载值 用于构造 DAC1 在低功耗模式下的输出
位 3:0	DUTY1: DAC1 在低功耗模式下的占空比值 用于构造 DAC1 在低功耗模式下的输出

7 复位和时钟控制 (RCC)

7.1 复位

有三种复位：系统复位、电源复位和备份域复位。

7.1.1 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域中的寄存器以外的所有寄存器为它们的复位数值。

当以下事件中的一件发生时，产生一个系统复位：

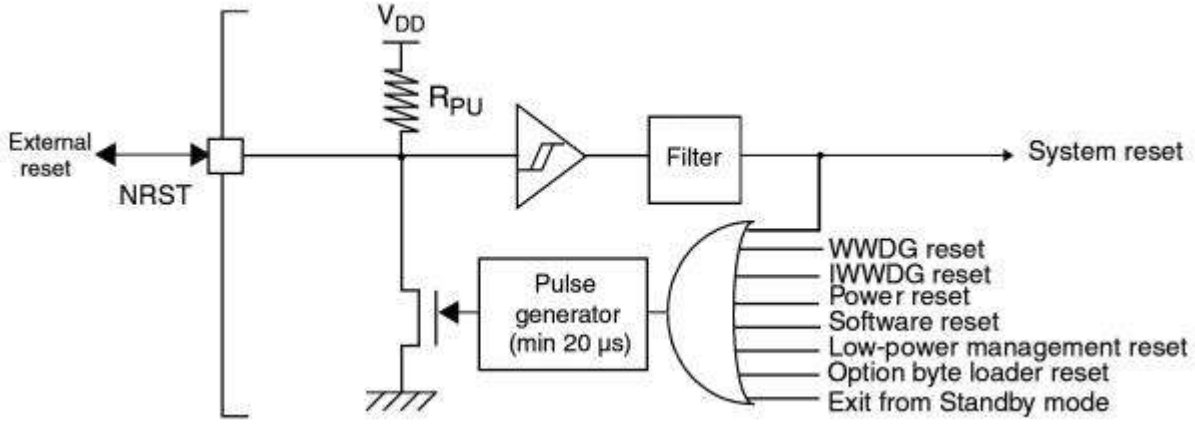
1. NRST 引脚上的低电平(外部复位)
2. 窗口看门狗事件(WWDG 复位)
3. 独立看门狗事件(IWWDG 复位)
4. 软件复位(SW 复位)
5. 低功耗管理复位
6. 选项字节装载器复位

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。

图中的复位源将最终作用于 NRST 引脚，并在一定的延时时段内保持低电平。复位入口地址被固定在 0x0000_0004 处。

内部的复位信号会在 NRST 引脚上输出，脉冲发生器保证每一个(外部或内部)复位源都能有至少 20μs 的脉冲延时；当 NRST 引脚被拉低产生外部复位时，它将产生复位脉冲。

图 7-1 复位电路简化图



软件复位

通过将 Cortex-M0 应用中断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-M0 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。

选项字节装载器复位

在设置 FORCE_OBL(FLASH_CR 寄存器中)为 1 的情况下，当软件读取选项字时，发出选项字节装载器复位。

7.1.2 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
3. BOR 复位

电源复位将复位除了备份域外的所有寄存器。

7.1.3 备份域复位

备份域拥有两个专门的复位，它们只影响备份域。

当以下事件中之一发生时，产生备份域复位：

1. 软件复位，由备份域控制寄存器(RCC_BDCR)的 BDRST 位触发。
2. 当 VDD 和 VBAT 都掉电的情况下，VDD 或 VBAT 上电。

7.2 时钟

可以使用四种不同的时钟源来驱动系统时钟(SYSCLK)：

- 内部高速(HSI)8MHzRC 振荡器时钟
- 外部高速(HSE)振荡器时钟
- PLL 时钟
- MSI（多速内部）振荡器时钟

从电源复位、系统或 RTC 域复位启动后，以及从待机模式唤醒后，会使用工作频率为 2.1MHz 的 MSI 作为系统时钟源。

器件具有两个次级时钟源：

- 37kHz 低速内部 RC(LSIRC)，该 RC 用于驱动独立看门狗，也可选择驱动用于从停止/待机模式自动唤醒的 RTC、LCD 以及 LPTIMER。
- 32.768kHz 低速外部晶振（LSE 晶振），该晶振可选择驱动实时时钟(RTCCLK)、LPTIMER、LCD 以及 USART 等。

对于每个时钟源来说，在未使用时都可单独打开或者关闭，以降低功耗。可通过多个预分频器配置 AHB 频率以及两个 APB（APB1 和 APB2）域。AHB、APB1 和 APB2 域的最大频率为 48MHz，具体取决于器件的电压范围。除以下时钟外，所有外设时钟均由系统时钟(SYSCLK)提供：

- 由以下两个时钟源之一提供的 48MHzUSB 时钟：
  - PLLVCO 时钟。
  - RC48 时钟(HSI48)
- RNG 可由 HSI48 时钟提供。
- ADC 可由 APB 时钟或 HSI16 时钟提供。
- 由以下五个时钟源之一提供的 LPTIM1,LPTIM2,LPUART1,USART2 和 UART3/4 时钟（由软件选择）：
  - MSI 时钟
  - HSI16 时钟
  - LSI 时钟
  - LSE 时钟
  - APB1 时钟
- 由以下四个时钟源之一提供的 I2C1/2 时钟（由软件选择）：
  - 系统时钟
  - HSI 时钟
  - MSI 时钟
  - APB1 时钟

- 由以下五个时钟源之一提供的 LPTIM3 和 USART1 时钟（由软件选择）：
  - MSI 时钟
  - HSI16 时钟
  - LSE 时钟
  - LSI 时钟
  - APB2 时钟
- 由以下时钟源提供的 RTC/LCD 时钟：
  - MSI 时钟
  - LSE 时钟
  - LSI 时钟
  - HSE/32 时钟
- 始终为 LSI 时钟的 IWDG 时钟。
- 由以下三个时钟源之一提供的 BEEPER 时钟
  - LSI 时钟
  - HSE 时钟
  - APB2 时钟

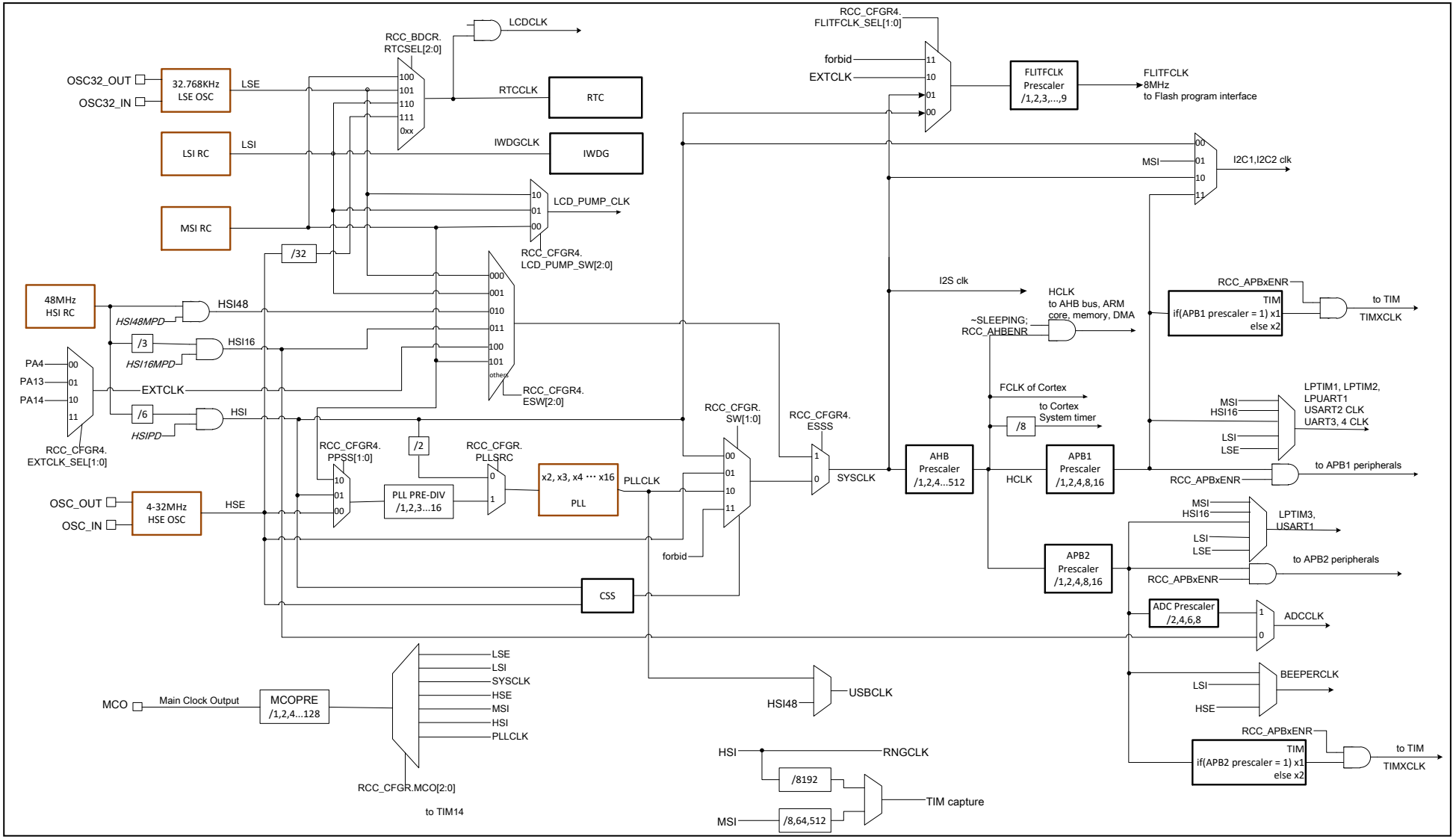
系统时钟(SYSCLK)频率必须大于或等于 RTC/LCD 时钟频率。

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

定时器时钟频率由硬件自动设置为固定值。分为两种情况：

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

图 7-2 时钟树


LCDCLK
RCC_BDCR.
RTCSEL[2:0]
OSC32_OUT
OSC32_IN
32.768KHz
LSE OSC
LSE
RTCCLK
RTC
LSI RC
LSI
IWDGCLK
IWDG
MSI RC
LCD_PUMP_CLK
RCC_CFGR4.
LCD_PUMP_SW[2:0]
/32
48MHz
HSI RC
HSI48
HSI48MPD
/3
HSI16
HSI16MPD
PA4
PA13
PA14
EXTCLK
/6
HSI
HSIPD
RCC_CFGR4.
EXTCLK_SEL[1:0]
RCC_CFGR4.
ESW[2:0]
RCC_CFGR4.
PPSS[1:0]
/2
RCC_CFGR.
PLLSRC
PLL PRE-DIV
/1,2,3...16
x2, x3, x4 ··· x16
PLL
PLLCLK
RCC_CFGR.
SW[1:0]
forbid
CSS
OSC_OUT
OSC_IN
4-32MHz
HSE OSC
HSE
RCC_CFGR4.
FLITFCLK_SEL[1:0]
forbid
EXTCLK
FLITFCLK
Prescaler
/1,2,3,...,9
FLITFCLK
8MHz
to Flash program interface
MSI
I2C1,I2C2 clk
I2S clk
HCLK
to AHB bus, ARM
core, memory, DMA
~SLEEPING;
RCC_AHBENR
RCC_APBxENR
TIM
if(APB1 prescaler = 1) x1
else x2
to TIM
TIMXCLK
FCLK of Cortex
to Cortex
System timer
/8
RCC_CFGR4.
ESSS
SYSCLK
AHB
Prescaler
/1,2,4...512
HCLK
APB1
Prescaler
/1,2,4,8,16
MSI
HSI16
LSI
LSE
LPTIM1, LPTIM2,
LPUART1
USART2 CLK
UART3, 4 CLK
RCC_APBxENR
to APB1 peripherals
APB2
Prescaler
/1,2,4,8,16
MSI
HSI16
LSI
LSE
LPTIM3,
USART1
RCC_APBxENR
to APB2 peripherals
ADC Prescaler
/2,4,6,8
ADCCLK
BEEPERCLK
LSI
HSE
RCC_APBxENR
TIM
if(APB2 prescaler = 1) x1
else x2
to TIM
TIMXCLK
MCO
Main Clock Output
MCOPRE
/1,2,4...128
LSE
LSI
SYSCLK
HSE
MSI
HSI
PLLCLK
RCC_CFGR.MCO[2:0]
to TIM14
HSI48
USBCLK
HSI
RNGCLK
/8192
MSI
/8,64,512
TIM capture

7.2.1 HSE 时钟

高速外部时钟信号(HSE)有 1 个时钟源：

- HSE 外部晶振/陶瓷谐振器

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

外部晶振/陶瓷谐振器（HSE 晶振）

4MHz 到 32MHz 外部振荡器的优点是精度非常高。

RCC_CR 寄存器的 HSERDY 标志指示 HSE 振荡器是否稳定。在启动时，硬件将该位置 1 后，HSE 时钟才可以使用。如果在 RCC_CR 寄存器中使能中断，则可产生中断。

HSE 晶振可通过 RCC_CR 寄存器中的 HSEON 位打开或关闭。

7.2.2 HSI8 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。

从低功耗停止模式唤醒后，可使用 HSI16 时钟。与使用 MSI 时钟唤醒相比，使用 HSI16 时钟的唤醒时间更短。

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 (相比于晶体振荡器或陶瓷谐振器)。

7.2.3 HSI16 时钟

HSI16 时钟信号是从 48MHz 内部 RC 振荡器进行 3 分频生成的。该时钟信号可直接用作系统时钟。

HSI16RC 振荡器的优点是成本较低（无需使用外部元件）。此外，其启动速度也要比 HSE 晶振快，但即使校准后，其精度也不及外部晶振或陶瓷谐振器。

7.2.4 MSI 时钟

MSI 时钟信号是从内部 RC 振荡器生成的。频率范围可通过软件使用 RCC_ICSCR 寄存器中的 MSIRANGE[2:0]位进行调整。有七个频率范围可用：65.5kHz、131kHz、262kHz、524kHz、1.05MHz、2.1MHz（默认值）和 4.2MHz。

在从复位重启、从待机唤醒后，始终会将 MSI 时钟用作系统时钟。从停止模式唤醒后，可选择 MSI 时钟作为系统时钟，而不使用 HSI8M 作为系统时钟。

当器件在复位后重启或者从待机模式唤醒时，MSI 频率会设为其默认值。从停止模式唤醒后，MSI 频率保持不变。

MSIRC 振荡器的优势在于可提供一个低成本（无外部元件）低功耗的时钟源。此振荡器用作低功耗模式下的唤醒时钟，以降低功耗。

RCC_CR 寄存器中的 MSIRDY 标志指示 MSIRC 是否稳定。在启动时，硬件将该位置 1 后，MSIRC 输出时钟才可以使用。

MSIRC 可通过 RCC_CR 寄存器中的 MSION 位打开或关闭。

MSIRC 还可作为备份时钟源（辅助时钟）使用，以防 HSE 晶振发生故障。

7.2.5 HSI48 时钟

HSI48 时钟信号由内部 48MHzRC 振荡器生成，可直接用于 USB。

内部 48MHzRC 振荡器主要用于向 USB 外设提供高精度时钟。

HSI48 要求使能与 48MHzRC 相关的 VREFINT 及其缓冲区。

时钟控制寄存器 2(RCC_CR2)中的 HSI48RDY 标志指示 HSI48RC 是否稳定。在启动时，硬件将该位置 1 后，HSI48 才可以使用。

HSI48 可通过控制寄存器 2(RCC_CR2)中的 HSI48ON 位打开或关闭。

7.2.6 PLL

内部 PLL 可用 HSI 、MSI 或 HSE 倍频得到， 参考图 7-2（时钟树） 和时钟控制寄存器 (RCC_CR)。

PLL 配置 (选择输入时钟，倍频因子) 须在使能 PLL 前配置好， 一旦 PLL 使能， PLL 使用到的这些参数就不能被改变。

改变 PLL 配置过程如下：

1. 设置 PLLON=0， 禁用 PLL。

2. 等待 PLLRDY 清 0， PLLRDY 清 0 时才表明 PLL 已经完全停止。
3. 改变 PLL 所需参数。
4. 设置 PLLON=1 重新使能 PLL。

当使能时钟中断寄存器 (RCC_CIR) 的相应位，当 PLL 时钟就绪时会产生一个中断。

PLL 输出频率设置的范围是 16-48 MHz.

7.2.7 LSE 时钟

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

LSE 晶振可通过 RCC_BDCR 寄存器中的 LSEON 位打开和关闭。

通过 RCC_BDCR 寄存器中的 LSEDRV[1:0]位，可在运行时更改晶振驱动强度，以实现稳健性、短启动时间和低功耗之间的最佳平衡。可在不同驱动水平之间动态更改驱动能力，但到达低驱动模式的情况除外。此时，只能通过上电复位或 RTC 复位切换到另一模式。

RCC_BDCR 寄存器中的 LSERDY 标志指示 LSE 晶振是否稳定。在启动时，硬件将该位置 1 后，LSE 晶振输出时钟信号才可以使用。如果在 RCC_CIER 寄存器中使能中断，则可产生中断。

外部源（LSE 旁路）

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

7.2.8 LSI 时钟

LSIRC 可作为低功耗时钟源在停止和待机模式下保持运行，供独立看门狗(IWDG)使用。时钟频率在 37k Hz 左右。

LSIRC 振荡器可通过 RCC_CSR 寄存器中的 LSION 位打开或关闭。

RCC_CSR 中的 LSIRDY 标志指示低速内部振荡器是否稳定。在启动时，硬件将该位置 1 后，此时钟才可以使用。如果在 RCC_CIR 中使能中断，则可产生中断。

从 IWDG 激活时候起，LSI 振荡器不能通过 LSION=0 停止。LSI 振荡器通过系统复位停止（通过硬件使用 FLASH_OPTR 寄存器中的 WDG_SW 选项位使能 IWDG 的情况除外）。如果 IWDG 已通过软件使能，则必须在系统复位后再次使能 LSI 振荡器，以确保 IWDG 和/或 RTC 正常工作。

LSI 测量可通过测量 LSI 振荡器的频散来获得准确的 RTC 时基和/或实现精度水平可以接受的 IWDG 超时（LSI 用作这些外设的时钟源时除外）。

7.2.9 系统时钟(SYSCLK)选择

可以使用四种不同的时钟源来驱动系统时钟(SYSCLK)：

- HSI16 振荡器
- HSI48 振荡器
- HSI8 振荡器
- LSI 振荡器
- LSE 振荡器
- HSE 振荡器
- EXTCLK 外部时钟
- PLL
- MSI 振荡器时钟（复位后的默认时钟）

在直接使用或者通过 PLL 使用某一时钟源作为系统时钟时，该时钟源无法停止。

只有在目标时钟源已就绪时（时钟在启动延迟或 PLL 锁相后稳定时），才可从一个时钟源切换到另一个。如果选择尚未就绪的时钟源，则切换在该时钟源就绪时才会进行。

7.2.10 HSE 时钟安全系统(CSSHSE)

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

如果检测到 HSE 时钟故障，则会自动禁止该振荡器，并且会产生 CSSHSEI 中断（时钟安全系统中断）以通知软件振荡器发生故障，从而使 MCU 执行救援操作。CSSHSEI 与 Cortex®-M0+NMI（不可屏蔽中断）

异常向量相链接。

注：当 CSSHSE 使能后，如果 HSE 时钟发生故障，则会产生 CSSHSE 中断，并会自动产生 NMI。NMI 将无限期执行，除非将 CSSHSE 中断挂起位清零。因此，NMI 中断服务程序(ISR)必须将 RCC_CICR 寄存器中的 CSSHSEC 位置 1，以清除 CSSHSE 中断。

如果直接或间接使用 HSE 振荡器作为系统时钟（间接是指它用作 PLL 输入时钟，PLL 时钟用作系统时钟），检测到故障时会导致系统时钟切换并禁止 HSE 振荡器。当故障发生时，如果 HSE 振荡器时钟为正在用作系统时钟的 PLL 时钟输入，则也会禁止该 PLL。

发生 HSE 故障时，系统时钟可切换为 MSI 或内部 8MHzHSI 时钟，具体取决于 RCC_CFGR 寄存器中 STOPWUCK 位的值。

7.2.11 LSE 时钟安全系统(CSSLSE)

时钟安全系统可通过软件在 LSE 上激活。可通过写入 RCC_CSR 寄存器中的 CSSLSEON 位来完成此操作。可通过硬件复位、RTC 软件复位、或在检测到 LSE 时钟故障后禁止该位。LSE 和 LSI 时钟使能（LSEON 和 LSION 置 1）并就绪（通过硬件将 LSERDY 和 LSIRDY 位置 1）后，以及通过 RTCSEL 位选择 RTC 时钟后，必须写入 CSSLSEON 位。

LSECSS 适用于所有模式：运行、睡眠、停止和待机。

如果在外部 32kHz 振荡器上检测到故障，则不会再向 RTC 提供 LSE 时钟，但寄存器的内容保持不变。

在待机模式下，会产生唤醒。在任何其它模式下，可发送中断来唤醒软件。之后，软件必须将 CSSLSEON 位复位，并通过将 LSEON 位复位的方式停用出现故障的

32kHz 振荡器。软件可通过 RTCSEL 位更改 RTC 时钟源（LSI、HSE 或无时钟），也可采取任何必要措施来确保应用的安全。

7.2.12 RTC 和 LCD 时钟

RTC 和 LCD 使用同一时钟源，可以是 LSE、LSI、MSI 或 HSE1MHz 时钟（HSE 经过可编程预分频器分频获得）。选择方式是编程 RCC_BDCR 寄存器中的 RTCSEL[1:0]位和 RCC_CR 寄存器中的 RTCPRE[1:0]位。

一旦选定 RTC 和 LCD 时钟源后，只能通过将 RCC_BDCR 寄存器中的 BDRST 位置 1 的方式或者通过 POR 来修改。

如果使用 LSE 或 LSI 作为 RTC 时钟源，RTC 会继续在停止和待机两种低功耗模式下工作，并可用作唤醒源。但是，如果使用 HSE 作为 RTC 时钟源，则 RTC 不能在停止和待机两种低功耗模式下使用。如果使用 LSE 或 LSI 作为 RTC 时钟源，则 LCD 可在停止低功耗模式下使用。

如果 RTC 的时钟由 LSE 提供，则 RTC 在系统复位后仍可获得时钟并保持正常工作。

*注： 为了能够在 APB1 时钟频率低于 RTC 时钟频率的七倍(7*RTCLCK)时读取 RTC 日历寄存器，软件必须分两次读取日历时间和日期寄存器。*

这样，当两次读取的 RTC_TR 结果相同时，才能确保数据正确。否则必须执行第三次读访问。

7.2.13 看门狗时钟

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

如果已通过软件使能 IWDG，则系统复位后会禁止 LSI。之后，必须再次使能 LSI 振荡器，以确保 IWDG 正常工作。

7.2.14 时钟输出功能

微控制器时钟输出(MCO)功能允许使用可配置的预分频值（1、2、4、8 或 16）将时钟输出到外部 MCO 引脚上。必须在复用功能模式下对相应 GPIO 端口的配置寄存器进行编程。可选择以下 7 种时钟信号之一作为 MCO 时钟：

- SYSCLK
- HSI
- MSI
- HSE
- PLL
- LSI
- LSE

信号选择由 RCC_CFGR 寄存器中的 MCO [3:0]位控制。

7.3 RCC 寄存器

7.3.1 时钟控制寄存器 (RCC_CR)

偏移地址： 0x00
复位值： 0x7800 XXX3
访问： 无等待状态，字、半字和字节访问
X 表示未定义

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	PLLRDY	PLLON	Res.				CSSON	Res.	HSERDY	HSEON
						rw	rw					rw		r	rw
0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HSIRDY	HSION
														r	rw
0	0	X	X	X	X	X	X	X	X	X	X	X	X	1	1

位 31:26	保留，必须保持复位值。
位 25	**PLLRDY:** PLL 时钟就绪标志 (HSE clock ready flag) 该位由硬件置 1，用于指示 PLL 已锁定。 0：PLL 未锁定 1：PLL 已锁定
位 24	**PLLON:** PLL 使能位 (PLL enable bit) 此位由软件置 1 和清零，以使能 PLL。 当进入停止或待机模式时由硬件清零。如果 PLL 时钟用作系统时钟或被选作系统时钟，该位不能复位。 0：PLL 关闭 1：PLL 开启
位 23:20	保留，必须保持复位值。
位 19	**CSSON:** 时钟安全系统使能 (Clock security system enable) 同软件设置或清零。 0: 时钟检测器关闭 1: 时钟检测器打开。
位 18	保留，必须保持复位值。
位 17	**HSERDY:** HSE 时钟就绪标志 (HSE clock ready flag) 由硬件设置，表明 HSE 振荡器是否稳定。当 HSEON 清零后，该位需要 6 个 HSE 振荡器周期才清零。 0: HSE 振荡器未就绪 1: HSE 振荡器就绪
位 16	**HSEON:** HSE 时钟使能 (HSE clock enable) 由软件设置和清零。 当进入停止或待机模式后由硬件清除并停止 HSE 振荡器。当直接或简接使用 HSE 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 15:2	保留，必须保持复位值。

位 1	**HSIRDY:** HSI 时钟就绪标志 (HSI clock ready flag) 由硬件置 1 来指示内部 HSI 振荡器已经稳定。在 HSION 位清零后，HSIRDY 位需 6 个 HSI 振荡器周期清零。. 0: HSI 振荡器未就绪 1: HSI 振荡器就绪
位 0	**HSION:** HSI 时钟使能 (HSI clock enable) 由软件设置和清零。 当从待机和停止模式返回或用作系统时钟的 HSE 振荡器发生故障时，该位由硬件置 1 来启动 HSI 振荡器。当 HSI 振荡器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: HSI 振荡器关闭 1: HSI 振荡器开启

7.3.2 时钟配置寄存器 (RCC_CFGR)

偏移地址 ： 0x04
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
Res.	MCOPRE			MCO				Res.	Res.	PLLMUL				PLLXTPRE	PLLSRC
	rw	rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
STOPWUCK	Res.	PPRE2			PPRE1			HPRE				SWS		SW	
rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	r	r	rw	rw
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位 31	保留，必须保持复位值。
位 30:28	**MCOPRE:** MCO 输出分频系数 (Microcontroller Clock Output Prescaler) 该 Bit 被软件设置选择 MCO 分频因子。推荐在 MCO 输出关闭时进行设置。 000: MCO/1 001: MCO/2 010: MCO/4 ..... 111: MCO/128
位 27:24	**MCO:** 微控制器时钟输出 (Microcontroller clock output) 通过软件设置和清除。 0000：MCO 输出禁用，MCO 上无时钟 0001：选择外部低速（LSE）振荡器时钟 0010：选择内部低速（LSI）振荡器时钟 0011：选择系统时钟 0100：选择外部 4-32 MHz（HSE）振荡器时钟 0101：选择内部 MSI 振荡器时钟 0110：选择内部 RC 8 MHz（HSI）振荡器时钟 0111：选择 PLL 时钟 1xxx：保留，必须保持重置值。 注：该时钟输出在启动或 MCO 时钟源切换期间可能有一些截短的周期。
位 23:22	保留，必须保持复位值。

位 21:18	**PLLMUL:** PLL 倍频系数 (PLL multiplication factor) 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。 0000: PLL 输入时钟的 2 倍频 0001: PLL 输入时钟的 3 倍频 0010: PLL 输入时钟的 4 倍频 0011: PLL 输入时钟的 5 倍频 0100: PLL 输入时钟的 6 倍频 0101: PLL 输入时钟的 7 倍频 0110: PLL 输入时钟的 8 倍频 0111: PLL 输入时钟的 9 倍频 1000: PLL 输入时钟的 10 倍频 1001: PLL 输入时钟的 11 倍频 1010: PLL 输入时钟的 12 倍频 1011: PLL 输入时钟的 13 倍频 1100: PLL 输入时钟的 14 倍频 1101: PLL 输入时钟的 15 倍频 1110: PLL 输入时钟的 16 倍频 1111: PLL 输入时钟的 16 倍频
位 17	**PLLXTPRE:** PLL 输入时钟 HSE 分频器 (HSE divider for PLL input clock) PLLXTPRE 与 RCC_CFGR2.PREDIV[0]意义相同。
位 16	**PLLSRC:** PLL 输入时钟源 (PLL entry clock source) 由软件置’1’或清’0’来选择 PLL 输入时钟源。只能在关闭 PLL 时才能写入此位。 0：HSI/2 作为 PLL 输入时钟 1：RCC_CFGR4.PPSS 选择的时钟作为 PLL 输入时钟
位 15	**STOPWUCK:** 从停止模式唤醒时钟选择 (Wake-up from Stop clock selection) 此位由软件置 1 和清零，用以选择从停止模式唤醒时钟。 1：芯片从 STOP 模式唤醒后系统时钟选择 MSI 0：芯片从 STOP 模式唤醒后系统时钟选择 HSI8MCLK
位 14	保留，必须保持复位值。
位 13:11	**PPRE2:** APB 高速预分频器 (APB2) (APB high-speed prescaler (APB2)) 这些位由软件置 1 和清零，用于控制 APB 高速时钟 (PCLK2) 的分频系数。 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:** APB 低速预分频器 (APB1) (APB low-speed prescaler (APB1)) 这些位由软件置 1 和清零，用于控制 APB 低速时钟 (PCLK1) 的分频系数。 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 7:4	**HPRE:** AHB 预分频器 (AHB prescaler) 这些位由软件置 1 和清零，用于控制 APB 时钟的分频系数。 0xxx：SYSCLK 不分频 1000：SYSCLK 2 分频 1001：SYSCLK 4 分频 1010：SYSCLK 8 分频 1011：SYSCLK 16 分频 1100：SYSCLK 64 分频 1101：SYSCLK 128 分频 1110：SYSCLK 256 分频 1111：SYSCLK 512 分频
位 3:2	**SWS:** 系统时钟切换状态 (System clock switch status) 这些位由硬件置 1 和清零，用于指示用作系统时钟的时钟源。 00：HSI 用作系统时钟 01：HSE 用作系统时钟 10：PLL 输出用作系统时钟 11：保留
位 1:0	**SW:** 系统时钟切换 (System clock switch) 由软件置 1 或清 0，与 RCC_CFGR4.ESW 一起来选择系统时钟源。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)。当 RCC_CFGR4.ESSS 为 0 时，系统时钟源如下： 00：HSI 用作系统时钟 01：HSE 用作系统时钟 10：PLL 输出用作系统时钟 11：保留

7.3.3 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
CSSLSEC	CSSLSEF	Res.	Res.	Res.	Res.	Res.	Res.	CSSHSEC	MSIRDYC	HSI16RDYC	PLLRDYC	HSERDYC	HSIRDYC	LSERDYC	LSIRDYC
w	r							w	w	w	w	w	w	w	w
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	MSIRDYIE	HSI16RDYIE	PLLRDYIE	HSERDYIE	HSIRDYIE	LSERDYIE	LSIRDYIE	CSSHSEF	MSIRDYF	HSI16RDYF	PLLRDYF	HSERDYF	HSIRDYF	LSERDYF	LSIRDYF
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位 31	**CSSLSEC:** LSE 时钟安全系统中断清零 (LSE Clock Security System Interrupt clear) 该位由软件置 1，用于将 CSSLSEF 标志清零。该位由硬件复位。 0：不起作用 1：将 CSSLSEF 标志清零
位 30	**CSSLSEF:** LSE 时钟安全系统中断标志 (LSE Clock Security System Interrupt flag) 该位通过软件写入 CSSLSEC 位复位。如果 LSE 时钟出现故障且 CSSLSE 置 1，则由硬件置 1。 0：未检测到 LSE 时钟故障 1：检测到 LSE 时钟故障
位 29:24	保留，必须保持复位值。

位 23	**CSSHSEC:** 时钟安全系统中断清零 (Clock Security System Interrupt clear) 该位由软件置 1，用于将 CSSHSEF 标志清零。该位由硬件复位。 0：不起作用 1：将 CSSHSEF 标志清零
位 22	**MSIRDYC:** MSI 就绪中断清零 (MSI ready Interrupt clear) 该位由软件置 1，用于将 MSIRDYF 标志清零。该位由硬件复位。 0：不起作用 1：将 MSIRDYF 标志清零
位 21	**HSI16RDYC:** HSI16 就绪中断清零 (HSI16 ready Interrupt clear) 该位由软件置 1，用于将 HSI16RDYF 标志清零。该位由硬件复位。 0：不起作用 1：将 HSI16RDYF 标志清零
位 20	**PLLRDYC:** PLL 就绪中断清零 (PLL ready Interrupt clear) 该位由软件置 1，用于将 PLLRDYF 标志清零。该位由硬件复位。 0：不起作用 1：将 PLLRDYF 标志清零
位 19	**HSERDYC:** HSE 就绪中断清零 (HSE ready Interrupt clear) 该位由软件置 1，用于将 HSERDYF 标志清零。该位由硬件复位。 0：不起作用 1：将 HSERDYF 标志清零
位 18	**HSIRDYC:** HSI 就绪中断清零 (HSI ready Interrupt clear) 该位由软件置 1，用于将 HSIRDYF 标志清零。该位由硬件复位。 0：不起作用 1：将 HSIRDYF 标志清零
位 17	**LSERDYC:** LSE 就绪中断清零 (LSE ready Interrupt clear) 该位由软件置 1，用于将 LSERDYF 标志清零。该位由硬件复位。 0：不起作用 1：将 LSERDYF 标志清零
位 16	**LSIRDYC:** LSI 就绪中断清零 (LSI ready Interrupt clear) 该位由软件置 1，用于将 LSIRDYF 标志清零。该位由硬件复位。 0：不起作用 1：将 LSIRDYF 标志清零
位 15	保留，必须保持复位值。
位 14	**MSIRDYIE:** MSI 就绪中断标志 (MSI ready interrupt flag) 该位由软件置 1 和复位，用于使能/禁止由 MSI 振荡器稳定所引起的中断。 0：禁止 MSI 就绪中断 1：使能 MSI 就绪中断
位 13	**HSI16RDYIE:** HSI16 就绪中断标志 (HSI16 ready interrupt flag) 该位由软件置 1 和复位，用于使能/禁止由 HSI16 振荡器稳定所引起的中断。 0：禁止 HSI16 就绪中断 1：使能 HSI16 就绪中断
位 12	**PLLRDYIE:** PLL 就绪中断标志 (PLL ready interrupt flag) 该位由软件置 1 和复位，用于使能/禁止由 PLL 锁定引起的中断。 0：禁止 PLL 锁定中断 1：使能 PLL 锁定中断
位 11	**HSERDYIE:** HSE 就绪中断标志 (HSE ready interrupt flag) 该位由软件置 1 和复位，用于使能/禁止由 HSE 振荡器稳定所引起的中断。 0：禁止 HSE 就绪中断 1：使能 HSE 就绪中断
位 10	**HSIRDYIE:** HSI 就绪中断标志 (HSI ready interrupt flag) 该位由软件置 1 和复位，用于使能/禁止由 HSI 振荡器稳定所引起的中断。 0：禁止 HSI 就绪中断 1：使能 HSI 就绪中断

位 9	**LSERDYIE:** LSE 就绪中断标志 (LSE ready interrupt flag) 该位由软件置 1 和复位，用于使能/禁止由 LSE 振荡器稳定所引起的中断。 0：禁止 LSE 就绪中断 1：使能 LSE 就绪中断
位 8	**LSIRDYIE:** LSI 就绪中断标志 (LSI ready interrupt flag) 该位由软件置 1 和复位，用于使能/禁止由 LSI 振荡器稳定所引起的中断。 0：禁止 LSI 就绪中断 1：使能 LSI 就绪中断
位 7	**CSSHSEF:** 时钟安全系统中断标志 (Clock Security System Interrupt flag) 该位通过软件写入 CSSHSEC 位复位。如果 HSE 时钟出现故障，则由硬件置 1。 0：当前未因 HSE 时钟故障而引起时钟安全中断 1：因 HSE 时钟故障而引起时钟安全中断
位 6	**MSIRDYF:** MSI 就绪中断标志 (MSI ready interrupt flag) 该位通过软件写入 MSIRDYC 位复位。当 MSI 时钟稳定且 MSIRDYIE 置 1 时，该位由硬件置 1。 0：当前未因 MSI 时钟故障而引起时钟就绪中断 1：因 MSI 时钟故障而引起时钟就绪中断
位 5	**HSI16RDYF:** HSI16 就绪中断标志 (HSI16 ready interrupt flag) 该位通过软件写入 HSI16RDYC 位复位。当 HSE 时钟稳定且 HSI16RDYIE 置 1 时，该位由硬件置 1。 0：当前未因 HSI16 时钟故障而引起时钟就绪中断 1：因 HSI16 时钟故障而引起时钟就绪中断
位 4	**PLLRDYF:** PLL 就绪中断标志 (PLL ready interrupt flag) 该位通过软件写入 PLLRDYC 位复位。当 PLL 时钟稳定且 PLLRDYIE 置 1 时，该位由硬件置 1。 0：当前未因 PLL 时钟故障而引起时钟就绪中断 1：因 PLL 时钟故障而引起时钟就绪中断
位 3	**HSERDYF:** HSE 就绪中断标志 (HSE ready interrupt flag) 该位通过软件写入 HSERDYC 位复位。当 HSE 时钟稳定且 HSERDYIE 置 1 时，该位由硬件置 1。 0：当前未因 HSE 时钟故障而引起时钟就绪中断 1：因 HSE 时钟故障而引起时钟就绪中断
位 2	**HSIRDYF:** HSI 就绪中断标志 (HSI ready interrupt flag) 该位通过软件写入 HSIRDYC 位复位。当 CSS 稳定且 HSIRDYIE 置 1 时，该位由硬件置 1。 0：当前未因 HSI 时钟故障而引起时钟就绪中断 1：因 HSI 时钟故障而引起时钟就绪中断
位 1	**LSERDYF:** LSE 就绪中断标志 (LSE ready interrupt flag) 该位通过软件写入 LSERDYC 位复位。当 LSE 时钟稳定且 LSERDYIE 置 1 时，该位由硬件置 1。 0：当前未因 LSE 时钟故障而引起时钟就绪中断 1：因 LSE 时钟故障而引起时钟就绪中断
位 0	**LSIRDYF:** LSI 就绪中断标志 (LSI ready interrupt flag) 该位通过软件写入 LSIRDYC 位复位。当 LSI 时钟稳定且 LSIRDYIE 置 1 时，该位由硬件置 1。 0：当前未因 LSI 时钟故障而引起时钟就绪中断 1：因 LSI 时钟故障而引起时钟就绪中断

7.3.4 APB2 外设复位寄存器 (RCC_APB2RSTR)

偏移地址 ： 0x0C
复位值 ： 0x00000 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
Res.	Res.	Res.	COMPRST	OAMPRST	Res.	BEEPRST	LPTIM3RST	Res.	DBGRST	Res.	Res.	Res.	Res.	Res.	Res.
			rw	rw		rw	rw		rw						
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	USART1RST	Res.	SPI1RST	TIM1RST	Res.	ADC1RST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SYSCFGRST
	rw		rw	rw		rw									rw
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位 31:29	保留，必须保持复位值。
位 28	**COMPRST:** 电压比较器控制模块复位 (COMP reset) 0：无作用 1：复位电压比较器控制模块
位 27	**OAMPRST:** 运放控制模块复位 (OAMP reset) 0：无作用 1：复位运放控制模块
位 26	保留，必须保持复位值。
位 25	**BEEPRST:** 蜂鸣器控制模块复位 (BEEPER reset) 0：无作用 1：复位蜂鸣器控制模块
位 24	**LPTIM3RST:** LPTIM3 复位 (LPTIM3 reset) 0：无作用 1：复位 LPTIM3
位 23	保留，必须保持复位值。
位 22	**DBGRST:** DBG 复位 (DBG reset) 此位由软件置 1 和清零。 0：不起作用 1：复位 DBG
位 21:16	保留，必须保持复位值。
位 15	保留，必须保持复位值。
位 14	**USART1RST:** USART1 复位 (USART1 reset) 此位由软件置 1 和清零。 0：不起作用 1：复位 USART1
位 13	保留，必须保持复位值。
位 12	**SPI1RST:** SPI 1 复位 (SPI 1 reset) 此位由软件置 1 和清零。 0：不起作用 1：复位 SPI 1
位 11	**TIM1RST:** TIM1 定时器复位 (TIM1 timer reset) 此位由软件置 1 和清零。 0：不起作用 1：复位 TIM1 定时器
位 10	保留，必须保持复位值。
位 9	**ADC1RST:** ADC 接口复位 (ADC interface reset) 此位由软件置 1 和清零。 0：不起作用 1：复位 ADC 接口

位 8:1	保留，必须保持复位值。
位 0	**SYSCFGRST:** 系统配置控制器复位 (System configuration controller reset) 此位由软件置 1 和清零。 0：不起作用 1：复位系统配置控制器

7.3.5 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
LPTIM1RST	Res.	DACRST	PWRRST	Res.	UART4RST	UART3RST	LPUART1RST	USBRST	I2C2RST	I2C1RST	Res.	Res.	Res.	USART2RST	Res.
rw		rw	rw		rw	rw	rw	rw	rw	rw				rw	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	SPI2RST	Res.	Res.	WWDGRST	Res.	LCDRST	LPTIM2RST	Res.	Res.	Res.	Res.	Res.	Res.	TIM3RST	TIM2RST
	rw			rw		rw	rw							rw	rw
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位 31	**LPTIM1RST:** 低功耗定时器 1 复位 (Low-power timer reset) 此位由软件置 1 和清零。 0：不起作用 1：复位低功耗定时器
位 30	保留，必须保持复位值。
位 29	**DACRST:** DAC 接口复位 (DAC interface reset) 此位由软件置 1 和清零。 0：不起作用 1：复位 DAC 接口
位 28	**PWRRST:** 电源接口复位 (Power interface reset) 此位由软件置 1 和清零。 0：不起作用 1：复位电源接口
位 27	保留，必须保持复位值。
位 26	**UART4RST:** UART4 复位 (UART4 reset) 此位由软件置 1 和清零。 0：不起作用 1：复位 UART4
位 25	**UART3RST:** UART3 复位 (UART3 reset) 此位由软件置 1 和清零。 0：不起作用 1：复位 UART3
位 24	**LPUART1RST:** LPUART1 复位 (USART1 reset) 此位由软件置 1 和清零。 0：不起作用 1：复位 LPUART1

位 23	**USBRST:** USB 复位 (USB reset) 此位由软件置 1 和清零。 0：不起作用 1：复位 USB
位 22	**I2C2RST:** I2C2 复位 (I2C2 reset) 此位由软件置 1 和清零。 0：不起作用 1：复位 I2C2
位 21	**I2C1RST:** I2C1 复位 (I2C1 reset) 此位由软件置 1 和清零。 0：不起作用 1：复位 I2C1
位 20:18	保留，必须保持复位值。
位 17	**USART2RST:** USART2 复位 (USART5 reset) 此位由软件置 1 和清零。 0：不起作用 1：复位 USART2
位 16:15	保留，必须保持复位值。
位 14	**SPI2RST:** SPI2 复位 (SPI2 reset) 此位由软件置 1 和清零。 0：不起作用 1：复位 SPI2
位 13:12	保留，必须保持复位值。
位 11	**WWDGRST:** 窗口看门狗复位 (Window watchdog reset) 此位由软件置 1 和清零。 0：不起作用 1：复位窗口看门狗
位 10	保留，必须保持复位值。
位 9	**LCDRST:** LCD 复位 (LCD reset) 此位由软件置 1 和清零。 0：不起作用 1：复位 LCD
位 8	**LPTIM2RST:** 低功耗定时器 2 复位 (Low-power timer reset) 此位由软件置 1 和清零。 0：不起作用 1：复位低功耗定时器
位 7:2	保留，必须保持复位值。
位 1	**TIM3RST:** 定时器 3 复位 (Timer 3 reset) 由软件置 1 和清零。 0：不起作用 1：复位定时器 3
位 0	**TIM2RST:** 定时器 2 复位 (Timer 2 reset) 由软件置 1 和清零。 0：不起作用 1：复位定时器 2

7.3.6 AHB 外部时钟使能寄存器 (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

Res.	Res.	Res.	Res.	Res.	Res.	RNGEN	CRYPTEN	Res.	IOPFEN	Res.	IOPDEN	IOPCEN	IOPBEN	IOPAEN	Res.
						rw	rw		rw		rw	rw	rw	rw	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	DVSQEN	Res.	CRCEN	Res.	FLITFEN	Res.	SRAMEN	Res.	DMAEN
							rw		rw		rw		rw		rw
0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

位 31:26	保留，必须保持复位值。
位 25	**RNGEN:** 随机数发生器时钟使能位 (Random Number Generator clock enable bit) 该位由软件置 1 和复位。 0：无作用 1：使能 RNG 时钟
位 24	**CRYPTEN:** 加密时钟使能位 (Crypto clock enable bit) 该位由软件置 1 和复位。 0：无作用 1：使能加密（AES 模块）时钟
位 23	保留，必须保持复位值。
位 22	**IOPFEN:** GPIOF 时钟使能位 (GPIOF clock enable bit) 由软件置 1 或清 0. 0: 无作用 1: GPIOF 时钟开启
位 21	保留，必须保持复位值。
位 20	**IOPDEN:** GPIOF 时钟使能位 (GPIOD clock enable) 由软件置 1 或清 0. 0: 无作用 1: GPIOD 时钟开启
位 19	**IOPCEN:** GPIOC 时钟使能位 (GPIOC clock enable bit) 由软件置 1 或清 0. 0: 无作用 1: GPIOC 时钟开启
位 18	**IOPBEN:** GPIOB 时钟使能位 (GPIOB clock enable bit) 由软件置 1 或清 0. 0: 无作用 1: GPIOB 时钟开启
位 17	**IOPAEN:** GPIOA 时钟使能位 (GPIOA clock enable bit) 由软件置 1 或清 0. 0: 无作用 1: GPIOA 时钟开启
位 16:9	保留，必须保持复位值。
位 8	**DVSQEN:** DVSQ 时钟使能位 (DVSQ clock enable bit) 由软件置 1 或清 0. 0: 无作用 1: DVSQ 时钟开启
位 7	保留，必须保持复位值。
位 6	**CRCEN:** CRC 时钟使能位 (CRC clock enable bit) 该位由软件置 1 和复位。 0：无作用 1：使能 CRC 时钟

位 5	保留，必须保持复位值。
位 4	**FLITFEN:** FLITF 时钟使能位 (FLITF clock enable bit) 由软件置 1 或清 0 来关闭/ 开启在睡眠模式下的 FLITF 时钟 0: 无作用 1: 在睡眠模式下 FLITF 时钟开启
位 3	保留，必须保持复位值。
位 2	**SRAMEN:** SRAM 接口时钟使能位 (SRAM clock enable bit) 由软件置 1 或清 0 来关闭/ 开启在睡眠模式下的 SRAM 时钟 0: 无作用 1: 在睡眠模式下 SRAM 接口时钟开启
位 1	保留，必须保持复位值。
位 0	**DMAEN:** DMA 时钟使能位 (DMA clock enable bit) 该位由软件置 1 和复位。 0：无作用 1：使能 DMA 时钟

7.3.7 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

无等待周期，除了出现先前的 APB 访问未完成的情况下必须插入等待直至先前的 APB 外设访问完成。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	COMPEN	OAMPEN	Res.	BEEPEREN	LPTIM3EN	Res.	DBGEN	Res.	Res.	Res.	Res.	Res.	Res.
			rw	rw		rw	rw		rw						
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	USART1EN	Res.	SPI1EN	TIM1EN	Res.	ADC1EN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SYSCFGEN
	rw	rw	rw	rw		rw									rw
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位 31:29	保留，必须保持复位值。
位 28	**COMPEN:** 电压比较器控制模块时钟使能位 (Comparator Control Module clock enable bit) 0：无作用 1：打开电压比较器控制模块时钟
位 27	**OAMPEN:** 运放控制模块时钟使能位 (OAMP clock enable bit) 0：无作用 1：打开 OAMP 时钟
位 26	保留，必须保持复位值。
位 25	**BEEPEREN:** 蜂鸣器控制模块时钟使能位 (BEEPER clock enable bit) 0：无作用 1：打开 BEEPER 时钟
位 24	**LPTIM3EN:** LPTIM3 时钟使能位 (LPTIM3 clock enable bit) 0：无作用 1：打开 LPTIM3 时钟
位 23	保留，必须保持复位值。

位 22	**DBGEN:** DBG 时钟使能位 (DBG clock enable bit) 此位由软件置 1 和清零。 0：禁止 DBG 时钟 1：使能 DBG 时钟
位 21:15	保留，必须保持复位值。
位 14	**USART1EN:** USART1 时钟使能位 (USART1 clock enable bit) 此位由软件置 1 和清零。 0：无作用 1：使能 USART1 时钟
位 13	保留，必须保持复位值。
位 12	**SPI1EN:** SPI1 时钟使能位 (SPI1 clock enable bit) 此位由软件置 1 和清零。 0：无作用 1：使能 SPI1 时钟
位 11	**TIM1EN:** TIM1 定时器时钟使能位 (TIM1 timer clock enable bit) 此位由软件置 1 和清零。 0：无作用 1：使能 TIM1 时钟
位 10	保留，必须保持复位值。
位 9	**ADC1EN:** ADC 时钟使能位 (ADC clock enable bit) 此位由软件置 1 和清零。 0：无作用 1：使能 ADC 时钟
位 8:1	保留，必须保持复位值。
位 0	**SYSCFGEN:** 系统配置控制器时钟使能位 (System configuration controller clock enable bit) 此位由软件置 1 和清零。 0：无作用 1：使能系统配置控制器时钟

7.3.8 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

无等待周期，除了出现先前的 APB1 访问未完成的情况下必须插入等待直至先前的 APB1 外设访问完成。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LPTIM1EN	Res.	DACEN	PWREN	Res.	UART4EN	UART3EN	LPUART1EN	USBEN	I2C2EN	I2C1EN	Res.	Res.	Res.	USART2EN	Res.
rw		rw	rw		rw	rw	rw	rw	rw	rw				rw	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	SPI2EN	Res.	Res.	WWDGEN	Res.	Res.	LPTIM2EN	Res.	Res.	Res.	Res.	Res.	Res.	TIM3EN	TIM2EN
	rw			rw			rw							rw	rw
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位 31	**LPTIM1EN:** 低功耗定时器 1 时钟使能位 (Low-power timer 1 clock enable bit) 此位由软件置 1 和清零。 0：无作用 1：使能低功耗定时器时钟
位 30	保留，必须保持复位值。
位 29	**DACEN:** DAC 接口时钟使能位 (DAC interface clock enable bit) 此位由软件置 1 和清零。 0：无作用 1：使能 DAC 接口时钟
位 28	**PWREN:** 电源接口时钟使能位 (Power interface clock enable bit) 此位由软件置 1 和清零。 0：无作用 1：使能电源接口时钟
位 27	保留，必须保持复位值。
位 26	**UART4EN:** UART4 时钟使能位 (UART4 clock enable bit) 此位由软件置 1 和清零。 0：无作用 1：使能 UART4 时钟
位 25	**UART3EN:** UART3 时钟使能位 (UART3 clock enable bit) 此位由软件置 1 和清零。 0：无作用 1：使能 UART3 时钟
位 24	**LPUART1EN:** LPUART1 时钟使能位 (LPUART1 clock enable bit) 此位由软件置 1 和清零。 0：无作用 1：使能 LPUART1 时钟
位 23	**USBEN:** USB 时钟使能位 (USB clock enable bit) 此位由软件置 1 和清零。 0：无作用 1：使能 USB 时钟
位 22	**I2C2EN:** I2C2 时钟使能位 (I2C2 clock enable bit) 此位由软件置 1 和清零。 0：无作用 1：使能 I2C2 时钟
位 21	**I2C1EN:** I2C1 时钟使能位 (I2C1 clock enable bit) 此位由软件置 1 和清零。 0：无作用 1：使能 I2C1 时钟
位 20:18	保留，必须保持复位值。
位 17	**USART2EN:** USART2 时钟使能位 (USART2 clock enable bit) 此位由软件置 1 和清零。 0：无作用 1：使能 USART2 时钟
位 16:15	保留，必须保持复位值。
位 14	**SPI2EN:** SPI2 时钟使能位 (SPI2 clock enable bit) 此位由软件置 1 和清零。 0：无作用 1：使能 SPI2 时钟
位 13:12	保留，必须保持复位值。
位 11	**WWDGEN:** 窗口看门狗时钟使能位 (Window watchdog clock enable bit) 此位由软件置 1 和清零。 0：无作用 1：使能窗口看门狗时钟
位 10:9	保留，必须保持复位值。

位 8	**LPTIM2EN:** 低功耗定时器 2 时钟使能位 (Low-power timer 2 clock enable bit) 此位由软件置 1 和清零。 0：无作用 1：使能低功耗定时器时钟
位 7:2	保留，必须保持复位值。
位 1	**TIM3EN:** 定时器 3 时钟使能位 (Timer 3 clock enable bit) 由软件置 1 和清零。 0：无作用 1：使能定时器 3 时钟
位 0	**TIM2EN:** 定时器 2 时钟使能位 (Timer 2 clock enable bit) 由软件置 1 和清零。 0：无作用 1：使能定时器 2 时钟

7.3.9 备份域控制寄存器 (RCC_BDCR)

偏移地址： 0x20
复位值： 0x0000 0000，由备份域复位电路复位
访问： 0 到 3 个等待周期，字、半字和字节访问
当连续对该寄存器进行访问时，将插入等待状态。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	LCDEN	LCDPUMPCLK			Res.	Res.	Res.	Res.	Res.	Res.	Res.	BDRST
				rw	rw	rw	rw								rw
0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RTCEN	Res.	Res.	Res.	Res.	RTC_SEL			Res.	Res.	Res.	LSEDRV		LSEBYP	LSERDY	LSEON
rw					rw	rw	rw				rw	rw	rw	rw	rw
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位 31:28	保留，必须保持复位值。
位 27	**LCDEN:** LCD 时钟使能位 (LCD clock enable bit) 此位由软件置 1 和清零。 0：无作用 1：使能 LCD 时钟
位 26:24	**LCDPUMPCLK:** LCD change PUMP 时钟选择 000：选择 MSI 作为 LCD change PUMP 时钟 001：选择 LSI 作为 LCD change PUMP 时钟 010：选择 LSE 作为 LCD change PUMP 时钟 其他：保留
位 23:17	保留，必须保持复位值。
位 16	**BDRST:** 备份域软件复位 (RTC domain software reset) 由软件置 1 或清零. 0: 复位未激活 1: 复位整个 RTC 备份域
位 15	**RTCEN:** RTC 时钟使能位 (RTC clock enable bit) 此位由软件置 1 和清零。 通过将 RTCRST 位置 1 或者通过 POR 将该位复位。 0：禁止 RTC 时钟 1：使能 RTC 时钟
位 14:11	保留，必须保持复位值。

位 10:8	**RTC_SEL:** RTC 和 LCD 时钟源选择位 (RTC and LCD clock source selection bits) 这些位由软件置 1，用于选择 RTC 的时钟源。 一旦选择 RTC 和 LCD 时钟源后，便不能进行切换，直到 RTCRST 置 1 或者发生上电复位。唯一的例外情况是选择了 LSE 振荡器时钟、 LSE 时钟停止并且被 CSSHSE 检测到，此时可切换时钟。 100：选择 MSI 作为 RTC 时钟 101：选择 LSE 作为 RTC 钟 110：选择 LSI 作为 RTC 时钟 111：选择 HSE/32 作为 RTC 时钟 如果 LSE 或 LSI 用作 RTC 时钟源， RTC 会继续在停止和待机两种低功耗模式下工作，并可用作唤醒源。但是，如果使用 HSE 时钟作为 RTC 时钟源，则 RTC 不能在停止和待机低功耗模式下使用。
位 7:5	保留，必须保持复位值。
位 4:3	**LSEDRV:** LSE 振荡器驱动能力位 (LSE oscillator Driving capability bits) 这些位由软件置 1，用于选择 LSE 振荡器的驱动能力。 这些位通过上电复位或 RTC 复位清零。一旦写入“00”后， LSEDRV 的内容便不能通过软件进行更改。 00：最低驱动 01：中低驱动 10：中高驱动 11：最高驱动
位 2	**LSEBYP:** 外部低速振荡器旁路位 (External low-speed oscillator bypass bit) 此位由软件置 1 和清零，用于在调试模式下旁路振荡器。只有在禁止 LSE 振荡器后才能写入该位。 通过将 RTCRST 位置 1 或者通过 POR 将该位复位。 0：不旁路 LSE 振荡器 1：旁路 LSE 振荡器
位 1	**LSERDY:** 外部低速振荡器就绪位 (External low-speed oscillator ready bit) 该位由硬件置 1 和清零，用于指示 LSE 振荡器已稳定。在将 LSEON 位清零后， LSERDY 将在 6 个 LSE 振荡器时钟周期后转为低电平。 通过将 RTCRST 位置 1 或者通过 POR 将该位复位。 0：外部 32 kHz 振荡器未就绪 1：外部 32 kHz 振荡器已就绪
位 0	**LSEON:** 外部低速振荡器使能位 (External low-speed oscillator enable bit) 此位由软件置 1 和清零。 通过将 RTCRST 位置 1 或者通过 POR 将该位复位。 0：LSE 振荡器关闭 1：LSE 振荡器开启

7.3.10 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复位，复位标志只能由电源复位清除

访问： 字、半字和字节访问，当连续对该寄存器进行访问时，将插入等待状态

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LPWRRSTF	WWDGRSTF	IWDGRSTF	SFTRSTF	PORRSTF	PINRSTF	OBLRSTF	RMVF	VCPURSTF	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r	r	r	r	r	r	r	rt_w	r							
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Res.	MSION	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LSIRDY	LSION
	rw													r	rw
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位 31	**LPWRRSTF:** 低功耗复位标志 (Low-power reset flag) 发生低功耗管理复位时，由硬件置 1。 通过写入 RMVF 位或者通过 POR 将该位清零。 0：未发生低功耗管理复位 1：发生低功耗管理复位
位 30	**WWDGRSTF:** 窗口看门狗复位标志 (Window watchdog reset flag) 发生窗口看门狗复位时，该位由硬件置 1。 通过写入 RMVF 位或者通过 POR 将该位清零。 0：未发生窗口看门狗复位 1：发生窗口看门狗复位
位 29	**IWDGRSTF:** 独立看门狗复位标志 (Independent watchdog reset flag) VDD 域发生独立看门狗复位时，该位由硬件置 1。 通过写入 RMVF 位或者通过 POR 将该位清零。 0：未发生看门狗复位 1：发生看门狗复位
位 28	**SFTRSTF:** 软件复位标志 (Software reset flag) 发生软件复位时，该位由硬件置 1。 通过写入 RMVF 位或者通过 POR 将该位清零。 0：未发生软件复位 1：发生软件复位
位 27	**PORRSTF:** POR/PDR 复位标志 (POR/PDR reset flag) 发生 POR/PDR 复位时，该位由硬件置 1。 通过写入 RMVF 位清零。 0：未发生 POR/PDR 复位 1：发生 POR/PDR 复位
位 26	**PINRSTF:** 引脚复位标志 (PIN reset flag) 发生来自 NRST 引脚的复位时，该位由硬件置 1。 通过写入 RMVF 位或者通过 POR 将该位清零。 0：未发生来自 NRST 引脚的复位 1：发生来自 NRST 引脚的复位
位 25	**OBLRSTF:** 选项字节加载复位标志 (Options bytes loading reset flag) 发生 OBL 复位时，该位由硬件置 1。 通过写入 RMVF 位或者通过 POR 将该位清零。 0：未发生 OBL 复位 1：发生 OBL 复位
位 24	**RMVF:** 清除复位标志 (Remove reset flag) 该位由软件置 1，用于清除复位标志。 0：不起作用 1：清除复位标志
位 23	**VCPURSTF:** 1.2 V 域的复位标志 当发生 1.2 V 域的 POR / PDR 时，由硬件置 1。 通过写 RMVF 位清零。 0: 没有发生 1.2 V 域的 POR / PDR 复位 1: 发生 1.2 V 域的 POR / PDR 复位
位 22:15	保留，必须保持复位值。

位 14	**MSION:** MSI 时钟使能位 (MSI clock enable bit) 此位由软件置 1 和清零。 由硬件置 1，用于在退出停止或待机模式时或者在直接或间接用作系统时钟的 HSE 振荡器发生故障时强制 MSI 振荡器打开。如果 MSI 用作系统时钟，则该位不可清零。 0： MSI 振荡器关闭 1： MSI 振荡器开启
位 13:2	保留，必须保持复位值。
位 1	**LSIRDY:** 内部低速振荡器就绪位 (Internal low-speed oscillator ready bit) 该位由硬件置 1 和清零，用于指示 LSI 振荡器已稳定。在将 LSION 位清零后， LSIRDY 将在 3 个 LSI 振荡器时钟周期后转为低电平。 该位通过系统复位进行复位。 0： LSI 振荡器未就绪 1： LSI 振荡器已就绪
位 0	**LSION:** 内部低速振荡器使能位 (Internal low-speed oscillator enable bit) 此位由软件置 1 和清零。 该位通过系统复位进行复位。 0： LSI 振荡器关闭 1： LSI 振荡器开启

7.3.11 AHB 外设复位寄存器(RCC_AHBRSTR)

偏移地址 ： 0x28
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
Res.	Res.	Res.	Res.	Res.	Res.	RNGRST	CRYPTRST	Res.	IOPFRST	Res.	IOPDRST	IOPCRST	IOPBRST	IOPARST	Res.
						rw	rw		rw		rw	rw	rw	rw	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	DVSQRST	Res.	CRCRST	Res.	Res.	Res.	Res.	Res.	DMARST
							rw		rw						rw
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位 31:26	保留，必须保持复位值。
位 25	**RNGRST:** 随机数发生器模块复位 (Random Number Generator module reset) 该位由软件置 1 和复位。 0：不起作用 1：复位 RNG 模块
位 24	**CRYPTRST:**加密模块复位 (Crypto module reset) 该位由软件置 1 和复位。 0：不起作用 1：复位 CRYPTO 模块
位 23	保留，必须保持复位值。
位 22	**IOPFRST:** GPIOF 复位 (I/O port F reset) 0: 无作用 1: 复位 GPIOF 口
位 21	保留，必须保持复位值。

位 20	**IOPDRST:** GPIOD 复位(I/O port D reset) 0: 无作用 1: 复位 GPIOD 口
位 19	**IOPCRST:** GPIOC 复位(I/O port C reset) 0: 无作用 1: 复位 GPIOC 口
位 18	**IOPBRST:** GPIOB 复位(I/O port B reset) 0: 无作用 1: 复位 GPIOB 口
位 17	**IOPARST:** GPIOA 复位(I/O port A reset) 0: 无作用 1: 复位 GPIOA 口
位 16:9	保留，必须保持复位值。
位 8	**DVSQRST:** DVSQ 复位 该位由软件置 1 和复位。 0：不起作用 1：复位 DVSQ
位 7	保留，必须保持复位值。
位 6	**CRCRST:** 测试集成模块复位 (Test integration module reset) 该位由软件置 1 和复位。 0：不起作用 1：复位测试集成模块
位 5:1	保留，必须保持复位值。
位 0	**DMARST:** DMA 复位 (DMA reset) 该位由软件置 1 和复位。 0：不起作用 1：复位 DMA

7.3.12 时钟配置寄存器 2 (RCC_CFGR2)

偏移地址： 0x2c
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
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

位 31:4	保留，必须保持复位值。

位 3:0	**PLL_PREDIV:** PREDIV 分频因子 这些位用于设置或清除 PREDIV 分频因子。这些位仅能在 PLL 关闭后改写。 注：位 0 与 RCC_CFGR 的位 17 相同 ，修改 RCC_CFGR 的位 17 同时改变这里的位 0。 0000: HSE 作为 PLL 的输入，不分频 0001: HSE 作为 PLL 的输入 2 分频 0010: HSE 作为 PLL 的输入 3 分频 0011: HSE 作为 PLL 的输入 4 分频 0100: HSE 作为 PLL 的输入 5 分频 0101: HSE 作为 PLL 的输入 6 分频 0110: HSE 作为 PLL 的输入 7 分频 0111: HSE 作为 PLL 的输入 8 分频 1000: HSE 作为 PLL 的输入 9 分频 1001: HSE 作为 PLL 的输入 10 分频 1010: HSE 作为 PLL 的输入 11 分频 1011: HSE 作为 PLL 的输入 12 分频 1100: HSE 作为 PLL 的输入 13 分频 1101: HSE 作为 PLL 的输入 14 分频 1110: HSE 作为 PLL 的输入 15 分频 1111: HSE 作为 PLL 的输入 16 分频

7.3.13 时钟配置寄存器 3(RCC_CFGR3)

偏移地址： 0x30
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
Res.	Res.	Res.	Res.	LPTIM3SW			LPTIM2SW			LPTIM1SW			LPUART1SW		
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
USBSEL	Res.	Res.	Res.	UART4SW			UART3SW			USART2SW			USART1SW		
rw				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位 31:28	保留，必须保持复位值。
位 27:25	**LPTIM3SW:** LPTIM3 时钟源选择 000：选择 PCLK2 001：选择 MSI clock 010：选择 LSE 011：选择 LSI 100：选择 HSI16M clock
位 24:22	**LPTIM2SW:** LPTIM2 时钟源选择 000：选择 PCLK1 001：选择 MSI clock 010：选择 LSE 011：选择 LSI 100：选择 HSI16M clock

位 21:19	**LPTIM1SW:** LPTIM1 时钟源选择 000：选择 PCLK1 001：选择 MSI clock 010：选择 LSE 011：选择 LSI 100：选择 HSI16M clock
位 18:16	**LPUART1SW:** LPUART1 时钟源选择 000：选择 PCLK1 001：选择 MSI clock 010：选择 LSE 011：选择 LSI 100：选择 HSI16M clock
位 15	**USBSEL:** USB 时钟源选择 0：选择 PLL clock 作为 USB 时钟 1：选择 HSI48M clock 作为 USB 时钟
位 14:12	保留，必须保持复位值。
位 11:9	**UART4SW:** UART4 时钟源选择 000：选择 PCLK1 001：选择 MSI clock 010：选择 LSE 011：选择 LSI 100：选择 HSI16M clock
位 8:6	**UART3SW:** UART3 时钟源选择 000：选择 PCLK1 001：选择 MSI clock 010：选择 LSE 011：选择 LSI 100：选择 HSI16M clock
位 5:3	**USART2SW:** USART2 时钟源选择 000：选择 PCLK1 001：选择 MSI clock 010：选择 LSE 011：选择 LSI 100：选择 HSI16M clock
位 2:0	**USART1SW:** USART1 时钟源选择 000：选择 PCLK2 001：选择 MSI clock 010：选择 LSE 011：选择 LSI 100：选择 HSI16M clock

7.3.14 控制寄存器 2 (RCC_CR2)

偏移地址 ： 0x34

复位值 ： 0x1e00 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
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MSIRANGE		
													rw	rw	rw
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Res.	Res.	Res.	Res.	Res.	Res.	MSIRDY	Res.	Res.	Res.	Res.	HSI48RDY	HSI48ON	HSI16DIS	HSI16RDY	HSI16ON
						rw					rw	rw	rw	rw	rw
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位 31:19	保留，必须保持复位值。
位 18:16	**MSIRANGE:** 选择 MSI 输出范围 000: 65.5KHz 001: 131KHz 010: 262KHz 011: 524KHz 100: 1.05MHz 101: 2.1MHz 110: 4.2MHz
位 15:10	保留，必须保持复位值。
位 9	**MSIRDY:** MSI 时钟就绪标志 1：MSI 已 ready 0：MSI 未 ready
位 8:5	保留，必须保持复位值。
位 4	**HSI48RDY:** HSI48 时钟就绪标志 1：HSI48M 时钟已 ready 0：HSI48M 时钟未 ready
位 3	**HSI48ON:** HSI48 时钟使能，由软件置 1 或清 0 1：打开 HSI48M 时钟 0：关闭 HSI48M 时钟
位 2	**HSI16DIS:** ADC HSI16 时钟请求禁止，由软件置 1 或清 0 1：ADC 控制器不能请求打开 HSI16M 时钟 0：ADC 控制器可以请求打开 HSI16M 时钟
位 1	**HSI16RDY:** HSI16 时钟就绪标志 1：HSI16M 时钟已 ready 0：HSI16M 时钟未 ready
位 0	**HSI16ON:** HSI16 时钟使能，由软件置 1 或清 0 1：打开 HSI16M 时钟 0：关闭 HSI16M 时钟

7.3.15 RCC HSE 时钟控制寄存器 (RCC_HSECTL)

偏移地址： 0xe0
复位值： 0x1f4d 0100
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
HSECSS_THRESHOLD							Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw									
0	0	0	1	1	1	1	1	0	1	0	0	1	1	0	1

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

位 31:25	**HSECSS_THRESHOLD:** 控制 HSE CSS 的计数阈值 当 HSE 频率低于 4M/ HSECSS_THRESHOLD[6:0]时，就会产生 HSEFAIL
位 24:16	保留，必须保持复位值。
位 15	**HSENF_BYP:** HSE 时钟信号上的 noisefilter 旁路设置 1：Bypass HSE 时钟信号上的 noisefilter，HSE 时钟不经过内部的 noisefilter 0：HSE 时钟要经过内部的 noisefilter
位 14:12	保留，必须保持复位值。
位 11:0	**HSEWT:** 设置 HSE 稳定等待时间 HSE 稳定等待时间为 HSEWTx8 个 HSE 时钟脉冲。

7.3.16 RCC PLL 时钟控制寄存器 (RCC_PLLCTL)

偏移地址： 0xe4
复位值： 0x4021 0a21
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
PLLWT					Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw											
0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	1
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0	0	0	0	1	0	1	0	0	0	1	0	0	0	0	1

位 31:27	**PLLWT:** 设置 PLL 稳定等待时间 PLL 稳定等待时间为 PLLWTx1024 个 PLL 时钟脉冲。
位 26:0	保留，必须保持复位值。

7.3.17 时钟配置寄存器 4(RCC_CFGR4)

偏移地址： 0xe8
复位值： 0x0000 0038
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
Res.	Res.	Res.	Res.	Res.	Res.	BEEPER_SW		Res.	Res.	Res.	Res.	I2C2CLK_SW		I2C1CLK_SW	
						rw	rw					rw	rw	rw	rw
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PPSS		EXTCLK_SEL		FLITFCLK_PRE		FLITFCLK_SEL		ESS	Res.	ESWS			ESW		
rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw
0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0

位 31:26	保留，必须保持复位值。

位 25:24	**BEEPER_SW:** 选择 BEEPER 的时钟信号 00：选择 PCLK2 作为 BEEPER 时钟 01：选择 LSI 作为 BEEPER 时钟 10：选择 HSE 作为 BEEPER 时钟 11：保留
位 23:20	保留，必须保持复位值。
位 19:18	**I2C2CLK_SW:** 选择 I2C2 的时钟信号 00：选择 HSI8M clock 作为 I2C2 时钟 01：选择 MSI 作为 I2C2 时钟 10：选择 SYSCLK 作为 I2C2 时钟 11：选择 PCLK1 作为 I2C2 时钟
位 17:16	**I2C1CLK_SW:** 选择 I2C1 的时钟信号 00：选择 HSI8M clock 作为 I2C1 时钟 01：选择 MSI 作为 I2C1 时钟 10：选择 SYSCLK 作为 I2C1 时钟 11：选择 PCLK1 作为 I2C1 时钟
位 15:14	**PPSS:** 选择 PLL 前置分频的时钟信号 00：选择 HSE 作为 PLL 前置分频的时钟信号 01：选择 HSI8M 作为 PLL 前置分频的时钟信号 10：选择 MSI 作为 PLL 前置分频的时钟信号 11：保留
位 13:12	**EXTCLK_SEL:** 选择 GPIO 时钟输入 00：选择 PA4 作为 GPIO 时钟输入信号 01：选择 PA13 作为 GPIO 时钟输入信号 10：选择 PA14 作为 GPIO 时钟输入信号 11：保留
位 11:10	**FLITFCLK_PRE:** 配置 FLITFCLK 的分频数，分频后 FLITFCLK 必须等于 8MHz 00：/1 01：/2 10：/4 11：/6
位 9:8	**FLITFCLK_SEL:** 选择 FLITFCLK 分频器输入时钟 00：选择 HSI8M 01：选择 SYSCLK 10：选择 GPIO 输入时钟 11：保留
位 7	**ESSS:** 选择由 RCC_CFGR.SW 配置 SYSCLK 还是有 RCC_CFGR4.ESW 配置 SYSCLK 0：使用 RCC_CFGR.SW 配置选择 SYSCLK 1：使用 RCC_CFGR4.ESW 配置选择 SYSCLK
位 6	保留，必须保持复位值。
位 5:3	**ESWS:** 当 ESSS 为 1 时指示 SYSCLK 时钟状态 000： LSE 目前作为 SYSCLK 001： LSI 目前作为 SYSCLK 010： HSI48M 目前作为 SYSCLK 011： HSI16M 目前作为 SYSCLK 100： GPIO 输入目前作为 SYSCLK 101： MSI 目前作为 SYSCLK 其他： 保留

位 2:0	**ESW:** 当 ESSS 为 1 时，选择不同时钟源作为 SYSCLK 000：选择 LSE 作为 SYSCLK 001：选择 LSI 作为 SYSCLK 010：选择 HSI48M 作为 SYSCLK 011：选择 HSI16M 作为 SYSCLK 100：选择 GPIO 输入作为 SYSCLK 101：选择 MSI 作为 SYSCLK 其他： 保留

8 通用 I/O (GPIO)

8.1 简介

每个通用 I/O 端口包括 5 个 32 位的配置寄存器(包括: 端口模式寄存器(GPIOx_MODER), 端口输出类型寄存器(GPIOx_OTYPER), 端口输出速度寄存器(GPIOx_OSPEEDR), 端口上下拉寄存器(GPIOx_PUPDR), 端口施密特寄存器(GPIOx_IOSR)), 2 个 32 位的数据寄存器(包括: 端口输入数据寄存器(GPIOx_IDR)和端口输出数据寄存器(GPIOx_ODR)), 1 个 32 位的置位/复位寄存器(GPIOx_BSRR)。此外, 所有的 GPIO 都包括 1 个 32 位锁定寄存器(GPIOx_LCKR)和 2 个 32 位复用功能选择寄存器(GPIOx_AFRH 和 GPIOx_AFRL)。

8.2 GPIO 的主要功能

- 输出状态: 推挽或开漏 + 上拉/下拉
- 从输出数据寄存器(GPIOx_ODR)或外设(复用功能输出)输出数据
- 可以为每个 I/O 选择不同的速度
- 支持的输入模式: 浮空、上拉/下拉、模拟
- 将数据输入到输入寄存器(GPIOx_IDR)或外设(复用功能输入)
- 置位和复位寄存器(GPIOx_BSRR), 对 GPIOx_ODR 具有按位写权限
- 锁定机制(GPIOx_LCKR), 可冻结 I/O 端口配置
- 模拟功能
- 复用功能选择寄存器
- 快速翻转, 每次翻转最快只需要两个时钟周期(红色表示不确定)
- 引脚复用灵活, 允许将 I/O 引脚用作 GPIO 或者多种外设功能中的一种
- 所有 I/O 都可以选择打开或关闭施密特特性

8.3 GPIO 功能说明

根据数据手册中列出的每个 I/O 端口的特性, 可通过软件将通用 I/O(GPIO)端口的各个端口位分别配置为多种模式:

- 输入浮空
- 输入上拉
- 输入下拉
- 模拟输入
- 具有上拉或下拉的开漏输出
- 具有上拉或下拉的推挽输出
- 具有上拉或下拉的复用功能推挽
- 具有上拉或下拉的复用功能开漏
- 所有 I/O 可以打开或关闭施密特特性

每个 I/O 端口均可自由编程, 但 I/O 端口寄存器必须按字(32 位)、半字(16 位)、或者是字节进行访问。GPIOx_BSRR 寄存器和 GPIOx_BRR 寄存器旨在实现对 GPIOx_ODR 寄存器进和地原子读取/修改访问, 这样便可以确保在读取和修改访问之间发生中断请求也不会有问题。

图 8-1 I/O 端口位的基本结构

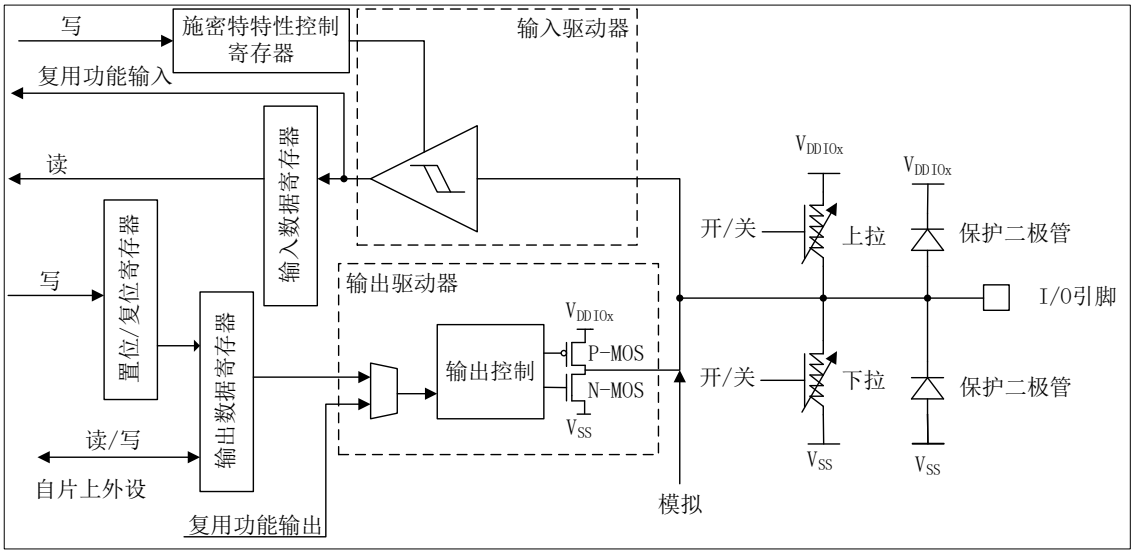


表 8-1 端口位配置表(1)

MODE(i)[1:0]	OTYPER(i)	OSPEED(i)[1:0]		PUPD(i)[1:0]		I/O 配置	
01	0	SPEED[1:0]		0	0	GP 输出	PP
	0			0	1	GP 输出	PP+PU
	0			1	0	GP 输出	PP+PD
	0			1	1	保留	
	1			0	0	GP 输出	OD
	1			0	1	GP 输出	OD+PU
	1			1	0	GP 输出	OD+PD
	1			1	1	保留(GP 输出 OD)	
10	0	SPEED[1:0]		0	0	AF	PP
	0			0	1	AF	PP+PU
	0			1	0	AF	PP+PD
	0			1	1	保留	
	1			0	0	AF	OD
	1			0	1	AF	OD+PU
	1			1	0	AF	OD+PD
	1			1	1	保留	
00	X	X	X	0	0	输入	浮空
	X	X	X	0	1	输入	PU
	X	X	X	1	0	输入	PD
	X	X	X	1	1	保留(输入浮空)	
11	X	X	X	0	0	输入/输出	模拟
	X	X	X	0	1	保留	
	X	X	X	1	0		
	X	X	X	1	1		

1. GP = 通用、PP = 推挽、PU = 上拉、PD = 下拉、OD = 开漏、AF = 复用功能

8.3.1 通用 I/O(GPIO)

在复位期间及复位刚刚完成后，复用功能尚未激活，大多数 I/O 端口都会默认为输入模式，但调试引脚例外:

- PA14: SWCLK 处于下拉状态;
- PA13: SWDIO 处于上拉状态。

当引脚配置为输出后，写入到输出寄存器(GPIOx_ODR)的值将在 I/O 引脚上输出。可以在推挽模式下或

开漏模式下使用输出驱动器(仅驱动低电平，高电平为高阻态)。

输入数据寄存器(GPIOx_IDR)每隔 1 个 AHB 时钟周期捕获一次 I/O 引脚的数据。

所有 GPIO 引脚都具有内部弱上拉及下拉电阻，可根据 GPIOx_PUPDR 寄存器中的值来打开/关闭。

8.3.2 I/O 引脚复用功能复用器和映射

器件 I/O 引脚通过一个复用器连接到外设/模块，该复用器一次仅允许一个外设的复用功能(AF)连接到同一个 I/O 引脚。这可以确保共用同一个 I/O 引脚的外设之间不会发生冲突。

每个 I/O 引脚都有一个复用器，该复用器采用多达 16 路复用功能输入(AF0 到 AF15)，可通过 GPIOx_AFRL(针对引脚 0 至 7)和 GPIOx_AFRH(针对引脚 8 至 15)寄存器对这些输入进行配置。

复位后，复用器选择为复用功能 0(AF0)。在复用模式下通过 GPIOx_MODER 寄存器配置 IO。

器件数据手册中详细说明了每个引脚的特定复用功能分配。

除了这种灵活的 I/O 复用架构外，各外设还可以将复用功能映射到不同 I/O 引脚，这可以优化小型封装中可用外设的数量。

要在指定配置下使用 I/O，用户必须按照以下步骤操作:

调试功能: 每个器件复位后，立即将这些引脚分配为可由调试主机使用的复用功能引脚。

GPIO: 在 GPIOx_MODER 寄存器中将所需 I/O 配置为输出、输入或模拟。

外设复用功能:

在 GPIOx_AFRL 或 GPIOx_AFRH 寄存器中，将 I/O 连接到所需的 AFx。

通过 GPIOx_OTYPER、GPIOx_PUPDR 和 GPIOx_SPEEDR 寄存器，分别选择类型、上拉/下拉以及输出速度。

在 GPIOx_MODER 寄存器中将所需的 I/O 配置为复用功能。

其它功能:

对于 ADC、DAC 和 COMP，在 GPIOx_MODER 寄存器中将所需 I/O 配置为模拟模式，并在 ADC、DAC 和 COMP 寄存器中配置所需功能。

对于 RTC、WKUPx 和振荡器的其它功能，在相关的 RTC、PWR 和 RCC 寄存器中配置所需功能。这些功能优先于标准 GPIO 寄存器中的配置。

8.3.3 I/O 端口控制寄存器

每个 GPIO 端口有 5 个 32 位存储器映射的控制寄存器(GPIOx_MODER、GPIOx_OTYPER、GPIOx_OSPEEDR、GPIOx_PUPDR、GPIOx_IOSR)，可配置多达 16 个 IO。GPIOx_MODER 寄存器用于选择 I/O 模式(输入、输出、AF 或模拟)。GPIOx_OTYPER 和 GPIOx_OSPEEDR 寄存器用于选择输出类型(推挽或开漏)和速度。无论采用哪种 I/O 方向，GPIOx_PUPDR 寄存器都用于选择上拉/下拉。GPIOx_IOSR 用于配置 I/O 的施密特特性，器件复位后 I/O 的施密特特性默认开启(包括 SWDIO 和 SWCLK 两个引脚)。

8.3.4 I/O 端口数据寄存器

每个 GPIO 都具有 2 个 16 位的数据寄存器: 输入和输出数据寄存器(GPIOx_IDR 和 GPIOx_ODR)。GPIOx_ODR 用于存储待输出数据，可以其进行读/写访问。通过 I/O 输入的数据存储到输入数据寄存器(GPIOx_IDR)中。

8.3.5 I/O 数据位操作

置位复位寄存器(GPIOx_BSRR)是一个 32 位寄存器，它允许应用程序在输出数据寄存器(GPIOx_ODR)中对各个单独的数据位执行置位和复位操作。GPIOx_ODR 中的每个数据位对应于 GPIOx_BSRR 中的两个控制位: BS(i)和 BR(i)。当将 BS(i)置位时会置位对应的 ODR(i)；当将 BR(i)置位时会复位对应的 ODR(i)。向 GPIOx_BSRR 中任何位写 0 都不会对 GPIOx_ODR 中的对应位产生任何的影响。如果在 GPIOx_BSRR 中同时尝试对 BS(i)和 BR(i)写'1'时，对 BS(i)写'1'的操作优先，对 BR(i)写'1'的操作被忽略掉。

使用 GPIOx_BSRR 寄存器更改 GPIOx_ODR 中各位的值是一个"单次"操作，不会锁定 GPIOx_ODR 位。随时都可以直接访问 GPIOx_ODR 位。GPIOx_BSRR 寄存器只是提供了一种对 GPIOx_ODR 寄存器执行原子按位处理的方法。

在对 GPIOx_ODR 进行位操作时，软件无需禁止中断: 在一次原子 AHB 写访问中，可以修改一个或多个位。

8.3.6 GPIO 锁定机制

通过将特定的写序列应用到 GPIOx_LCKR 寄存器，可以冻结 GPIO 控制寄存器。冻结的寄存器包括 GPIOx_MODER、GPIOx_OTYPER、GPIOx_PUPDR、GPIOx_AFRL、GPIOx_AFRH 和 GPIOx_IOSR。

要对 GPIOx_LCKR 寄存器执行写操作必须应用特定的写/读序列。当正确的 LOCK 序列应用到此寄存器的第 16 位后，会使用 LCKR[15:0]的值来锁定对应 I/O 的配置(在写序列期间，LCKR[15:0]的值必须保持不变)。将 LOCK 序列应用到某个端口位后，在执行下一次 MCU 复位或外设复位之前，将无法对该端口位的值进行更改。每个 GPIOx_LCKR 位都对应决定着 GPIO 控制寄存器(GPIOx_MODER、GPIOx_OTYPER、GPIOx_OSPEEDR、GPIOx_PUPDR、GPIOx_AFRL、GPIOx_AFRH 和 GPIOx_IOSR)中的对应位。

8.3.7 I/O 复用功能输入输出

有两个寄存器(GPIOx_AFRL 和 GPIOx_AFRH)可以用来从每个 I/O 可用的复用功能输入/输出中进行选择。借助这些寄存器，用户可根据应用程序的要求将某个复用功能连接到指定的 I/O 引脚上。由于 AF 选择信号由复位功能输入和复用功能输出共用，所以只需要为指定的 I/O 的复用功能输入/输出选择一个通道即可。

8.3.8 外部中断线/唤醒线

所有端口都具有外部中断功能。要使用外部中断线，必须将端口配置为输入模式。

8.3.9 输入配置

对 I/O 端口进行编程作为输入时:

- 输出驱动器被禁用
- 根据 GPIOx_PUPDR 寄存器中的值决定是否打开上拉或和下拉电阻
- 输入数据寄存器每隔 1 个 AHB 时钟周期对 I/O 引脚上的数据进行一次采样
- 对输入数据寄存器的读访问可获取 I/O 状态

8.3.10 输出配置

- 对 I/O 端口进行编程作为输出时:
- 输出缓冲器被打开:
  - 开漏模式: 输出寄存器中的'0'可激活 N-MOS，而输出寄存器中的'1'会使端口保持高阻态(Hi-Z)（P-MOS 始终不激活）
  - 推挽模式: 输出寄存器中的'0'可激活 N-MOS，输出寄存器中的'1'可激活 P-MOS。
- 根据 GPIOx_PUPDR 寄存器中的值决定是否打开上拉和下拉电阻
- 输入数据寄存器每隔 1 个 AHB 时钟周期对 I/O 引脚上的数据采样一次
- 对输入数据寄存器的读访问可获取 I/O 的状态
- 对输出数据寄存器的读访问可获取最后的写入值

8.3.11 复用功能配置

对 I/O 端口进行编程作为复用功能时:

- 可将输出缓冲器配置为开漏或推挽模式
- 输出缓冲器由来自外设的信号驱动(发送使能和数据)
- 根据 GPIOx_IOSR 寄存器的配置使能或禁用施密特触发器
- 根据 GPIOx_PUPDR 寄存器的配置决定是否打开上拉电阻和下拉电阻
- 输入数据寄存器每隔 1 个 AHB 时钟周期对 I/O 引脚上的数据进行一次采样
- 对输入数据寄存器的读访问可获取 I/O 状态

注意: 当所选复用功能为 LCD 功能时，不应用上述复用功能配置。这在种情况下，编程为复用功能输出的 I/O 按模拟配置中所述进行配置。

8.3.12 模拟配置

对 I/O 端口进行编程作为模拟配置时:

- 输出缓冲器被禁用
- 施密特触发器被强制停用, I/O 引脚每个模拟输入的功耗变为零。施密特触发器的输出被强制处理为恒定值'0'。
- 弱上拉和下拉电阻被硬件强制关闭。
- 对输入数据寄存器的读访问值为'0'。

8.3.13 施密特特性配置

当 I/O 工作在模拟模式下时，施密特触发器被强制关闭。除此之外的其他模式下，I/O 的施密特触发器状态由 GPIOx_IOSR 寄存器中的位决定。在数字信号采样时，建议开启施密特触发器，这样可以增强数据采样的搞干扰能力。

8.3.14 将 HSE 或 LSE 引脚用作 GPIO

当 HSE 或 LSE 振荡器关闭(复位后的默认状态)时，可将相关的振荡器引脚用作常规 GPIO。

当 HSE 或 LSE 打开的时候，振荡器会控制与其相关的引脚，此时这些引脚的 GPIO 配置不起作用。

将振荡器配置为用户外部输入时钟模式时，仅为时钟输入保留 OSC_IN 或 OSC32_IN 引脚，OSC_OUT 或 OSC32_OUT 引脚仍可以作为 GPIO 使用。

8.3.15 在 RTC 电源域中使用 GPIO 引脚

当内核电源域掉电时(器件进入待机模式时)，PC13/PC14/PC15 GPIO 功能会丢失。在这种情况下，如果不被配置成 RTC 相关复用功能，则会被自动设置成模拟输入模式。

8.4 GPIO 寄存器

8.4.1 GPIO 模式寄存器(GPIOx_MODER)

其中 x = A…D

偏移地址: 0x00

复位值: 端口 A 为 0x2800 0000; 其它端口为 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MODER15[1:0]		MODER14[1:0]		MODER13[1:0]		MODER12[1:0]		MODER11[1:0]		MODER10[1:0]		MODER9[1:0]		MODER8[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	wr
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MODER7[1:0]		MODER6[1:0]		MODER5[1:0]		MODER4[1:0]		MODER3[1:0]		MODER2[1:0]		MODER1[1:0]		MODER0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位 2y+1:2y	**MODERy[1:0]:** 端口 x 的 y 引脚的工作模式配置位(y=0…15)，这些位可由软件修改来配置 I/O 的工作模式，具体如下: 00: 输入模式(复位默认) 01: 通用输出模式 10: 复用功能模式 11: 模拟模式

8.4.2 GPIO 端口输出类型寄存器(GPIOx_OTYPER)

其中 x = A…D

偏移地址: 0x04

复位值: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OT15	OT14	OT13	OT12	OT11	OT10	OT9	OT8	OT7	OT6	OT5	OT4	OT3	OT2	OT1	OT0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位 31:16	**保留。**必须保持为复位值。
位 15:0	**OTy:** 端口 x 的 y 引脚输出类型配置位(y=0…15), 这些位由软件修改来配置 I/O 口的输出类型。 0: 推挽输出(复位默认状态) 1: 开漏输出

8.4.3 GPIO 端口输出速度寄存器(GPIOx_OSPEEDR)

其中 x = A…D
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
OSPEED15[1:0]		OSPEED14[1:0]		OSPEED13[1:0]		OSPEED12[1:0]		OSPEED11[1:0]		OSPEED10[1:0]		OSPEED9[1:0]		OSPEED8[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OSPEED7[1:0]		OSPEED6[1:0]		OSPEED5[1:0]		OSPEED4[1:0]		OSPEED3[1:0]		OSPEED2[1:0]		OSPEED1[1:0]		OSPEED0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位 2y+1:2y	**OSPEEDy[1:0]:** 端口 x 的 y 引脚的速度配置位(y=0…15), 软件通过修改这些位来设置 I/O 的速度, 具体定义如下: x0: 低速 01: 中速 11: 高速

8.4.4 GPIO 端口上拉/下拉寄存器(GPIOx_PUPDR)

其中 x = A…D
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
PUPU15[1:0]		PUPU14[1:0]		PUPU13[1:0]		PUPU12[1:0]		PUPU11[1:0]		PUPU10[1:0]		PUPU9[1:0]		PUPU8[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PUPU7[1:0]		PUPU6[1:0]		PUPU5[1:0]		PUPU4[1:0]		PUPU3[1:0]		PUPU2[1:0]		PUPU1[1:0]		PUPU0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位 2y+1:2y	**PUPDy[1:0]:** 端口 x 的 y 引脚的上拉/下拉配置位(y=0…15), 软件修改这些位来设置 I/O 引脚的上拉或下拉: 00: 无上拉和下拉 01: 使能上拉 10: 使能下拉 11: 保留

8.4.5 GPIO 端口输入数据寄存器(GPIOx_IDR)

其中 x = A…D
偏移地址: 0x10
复位值: 0x0000 XXXX(X 表示不确定)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID15	ID14	ID13	ID12	ID11	ID10	ID9	ID8	ID7	ID6	ID5	ID4	ID3	ID2	ID1	ID0
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

位 31:16	**保留:** 必须保持为复位值。
位 15:0	**ID[15:0]:** 端口 x 的输入数据，这些位只读，它们包含相应 I/O 口的输入值。

8.4.6 GPIO 端口输出数据寄存器(GPIOx_ODR)

其中 x = A…D
偏移地址: 0x14
复位值: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OD15	OD14	OD13	OD12	OD11	OD10	OD9	OD8	OD7	OD6	OD5	OD4	OD3	OD2	OD1	OD0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位 31:16	**保留:** 必须保持复位值。
位 15:0	**ODy:** 端口 x 的 y 引脚的输出值(y=0…15)，这些位用于配置端口的输出状态: 0: 端口 x 的第 y 个引脚输出低电平 1: 端口 x 的第 y 个引脚输出高电平

8.4.7 GPIO 端口置位/复位寄存器(GPIOx_BSRR)

其中 x = A…D
偏移地址: 0x18
复位值: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
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
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位 31:16	**BRy:** 端口 x 的复位控制位(y=0…15)，这些位只写。如果读这些位，则返回 0x0000。若 BSy 和 BRy 同时设置，BSy 有优先权。 0: 对端口 x 的 ODRy 位无影响 1: 复位端口 x 的 ODRy 位
位 15:0	**BSy:** 端口 x 的设置控制位(y=0…15)，这些位只写。如果读这些位则返回 0x0000。 0: 对端口 x 的相应的 ODRy 位无影响 1: 置位端口 x 的相应的 ODRy 位

8.4.8 GPIO 端口配置锁定寄存器(GPIOx_LCKR)

其中 x = A…D
偏移地址: 0x1C
复位值: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LCKK
															rw
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位 31:17	**保留:** 必须保持为复位值。
位 16	**LCKK:** 锁定键，该位可随时读取，它仅能由锁键写序列来改写。 0: 端口配置锁定键不激活 1: 端口配置锁定键激活。GPIOx_LCKR 寄存器锁定直到一个 MCU 复位产生。 锁定键写序列: 写 LCKR[16] = ‘1’ + LCKR[15:0] 写 LCKR[16] = ‘0’ + LCKR[15:0] 写 LCKR[16] = ‘1’ + LCKR[15:0] 读 LCKR 读 LCKR[16] = ‘1’(此读操作为可选操作，但它可确认锁定已激活) 注意: 在锁定键写序列期间，不能更改 LCK[15:0]的值。锁定序列中的任何错误操作都将中止锁定操作。在任一端口位上的第一个锁定序列后，对 LCKK 位的任何读访问都将返回’1’，直到下一次 MCU 复位或外设复位为止。
位 15:0	**LCKy:** 端口 x 锁定位 y(y=0…15)，这些位都是读/写位，但只能在 LCKK 位等于’0’时执行写操作。 0: 端口配置未锁定 1: 端口配置已锁定

8.4.9 GPIO 端口复用功能低位寄存器(GPIOx_AFRL)

其中 x = A…D
偏移地址: 0x20
复位值: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AFSEL7[3:0]				AFSEL6[3:0]				AFSEL5[3:0]				AFSEL4[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AFSEL3[3:0]				AFSEL2[3:0]				AFSEL1[3:0]				AFSEL0[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位 4y+3:4y	**AFSELy[3:]:** 端口 x 的引脚 y 的复用功能选择(y=0…7)，软件修改这些位来配置 I/O 的复用功能: 0000: AF0 0001: AF1 0010: AF2 0011: AF3 0100: AF4 0101: AF5 0110: AF6 0111: AF7 其他值: 保留。

8.4.10 GPIO 端口复用功能高位寄存器(GPIOx_AFRH)

其中 x = A…D

偏移地址: 0x24

复位值: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AFRH15[3:0]				AFRH14[3:0]				AFRH13[3:0]				AFRH12[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AFRH11[3:0]				AFRH10[3:0]				AFRH9[3:0]				AFRH8[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位 4y+3:4y	**AFRHy[3:0]:** 端口 x 引脚 y 的复用功能选择(y=8…15)，软件通过修改这些位来配置 I/O 的复用功能: 0000: AF0 0001: AF1 0010: AF2 0011: AF3 0100: AF4 0101: AF5 0110: AF6 0111: AF7 其他值: 保留。

8.4.11 GPIO 端口复位寄存器(GPIOx_BRR)

其中 x = A…D

偏移地址: 0x28

复位值: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
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
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位 31:16	**保留:** 必须保持为复位值。
位 15:0	**BRy:** 端口 x 的 y 引脚的复位控制位(y=0…15), 这些位只能写, 如果对这些位进行读操作则返回 0x0000。 0: 对应端口 x 的 ODy 位无影响。 1: 复位端口 x 相应的 ODy 位。

8.4.12 GPIO 端口输入输出施密特寄存器(GPIOx_IOSR)

其中 x = A…D

偏移地址: 0x30

复位值: 0x0000 0000

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

位 31:16	**保留:** 必须保持为复位值。
位 15:0	**SENy:** 端口 x 的 y 引脚的施密特特性开关位(y=0…15)。 0: 使能 I/O 施密特特性 1: 禁用 I/O 施密特特性

9 系统配置控制器 (SYSCFG)

9.1 简介

该器件具有一组配置寄存器，系统配置控制器的主要用途如下：

- 在部分 IO 口上启用或禁用 I2C 超快模式 (Fast Mode Plus)；
- 重映射部分 DMA 触发源到其它不同的 DMA 通道上；
- 重映射存储器到代码起始区域；
- 管理连接到 GPIO 口的外部中断；
- 管理系统的可靠性特性；
- LCD 电源线去耦；
- 温度传感器和内部参考电压管理。

9.2 SYSCFG 寄存器

9.2.1 SYSCFG 配置寄存器 1(SYSCFG_CFGR1)

该寄存器用于存储器和 DMA 请求重映射的特定配置，并控制特殊的 I / O 功能。。

偏移地址： 0x00

复位值： 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	I2C_PA10_FMP	I2C_PA9_FMP	I2C2_FMP	I2C1_FMP	I2C_PB9_FMP	I2C_PB8_FMP	I2C_PB7_FMP	I2C_PB6_FMP
								rw	rw	rw	rw	rw	rw	rw	rw
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	USART1_RX_DMA_RMP	USART1_TX_DMA_RMP	ADC_DMA_RMP	Res.	LCD_CAPA					MEM_MODE	
					rw	rw	rw							rw	rw
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位 31:24	保留，必须保持复位值。
位 23	**I2C_PA10_FMP:** 超快模式(FM+)驱动能力激活位 (PA10 Fast Mode Plus (FM+) driving capability activation bits) 由软件设置和清 0 这位。为 PA10 口线开启 I2C 超快模式(FM+)。 0: PA10 引脚设置为标准模式 1: PA10 引脚配置为 I2C 超快模式(FM+)，且 I2C 速度控制被旁路
位 22	**I2C_PA9_FMP:** 超快模式(FM+)驱动能力激活位 (PA9 Fast Mode Plus (FM+) driving capability activation bits) 由软件设置和清 0 这位。为 PA9 口线开启 I2C 超快模式(FM+)。 0: PA9 引脚设置为标准模式 1: PA9 引脚配置为 I2C 超快模式(FM+)，且 I2C 速度控制被旁路
位 21	**I2C2_FMP:** I2C2 的超快模式(FM+)驱动能力激活位 (FM+ driving capability activation for I2C2) 由软件设置和清 0 这位，这位是与 I2C_Pxx_FMP 位或运算。 0: 超快模式(FM+)只是由 I2C_Pxx_FMP 位控制 1: 通过 GPIOx_AFR 中的选择位选择的所有 I2C2 引脚都启用超快模式(FM+)，这是为没有专用 I2C_Pxx_FMP 控制位的焊盘启用超快模式(FM+)的唯一方法

位 20	**I2C1_FMP:** I2C1 的超快模式(FM+)驱动能力激活位 (FM+ driving capability activation for I2C1) 由软件设置和清 0 这位，这位是与 I2C_Pxx_FMP 位或运算。 0: 超快模式(FM+)只是由 I2C_Pxx_FMP 位控制 1: 通过 GPIOx_AFR 寄存器中的选择位选择的所有 I2C1 引脚都启用超快模式(FM+)，这是为没有专用 I2C_Pxx_FMP 控制位的焊盘启用超快模式(FM+)的唯一方法
位 19	**I2C_PB9_FMP:** 超快模式(FM+)驱动能力激活位 (PB9 Fast Mode Plus (FM+) driving capability activation bits) 由软件设置和清 0 这位。为 PB9 口线开启 I2C 超快模式(FM+)。 0: PB9 引脚设置为标准模式 1: PB9 引脚配置为 I2C 超快模式(FM+)，且 I2C 速度控制被旁路
位 18	**I2C_PB8_FMP:** 超快模式(FM+)驱动能力激活位 (PB8 Fast Mode Plus (FM+) driving capability activation bits) 由软件设置和清 0 这位。为 PB8 口线开启 I2C 超快模式(FM+)。 0: PB8 引脚设置为标准模式 1: PB8 引脚配置为 I2C 超快模式(FM+)，且 I2C 速度控制被旁路
位 17	**I2C_PB7_FMP:** 超快模式(FM+)驱动能力激活位 (PB7 Fast Mode Plus (FM+) driving capability activation bits) 由软件设置和清 0 这位。为 PB7 口线开启 I2C 超快模式(FM+)。 0: PB7 引脚设置为标准模式 1: PB7 引脚配置为 I2C 超快模式(FM+)，且 I2C 速度控制被旁路
位 16	**I2C_PB6_FMP:** 超快模式(FM+)驱动能力激活位 (PB6 Fast Mode Plus (FM+) driving capability activation bits) 由软件设置和清 0 这位。为 PB6 口线开启 I2C 超快模式(FM+)。 0: PB6 引脚设置为标准模式 1: PB6 引脚配置为 I2C 超快模式(FM+)，且 I2C 速度控制被旁路
位 15:11	保留，必须保持复位值。
位 10	**USART1_RX_DMA_RMP:** USART1_RX DMA 请求重映射位 由软件设置和清除该位。它控制着 USART1_RXDMA 通道请求的重映射 0: 无重映射(USART1_RX 请求映射在 DMA 通道 3 上) 1: 重映射 (USART1_RX 请求映射在 DMA 通道 5 上)
位 9	**USART1_TX_DMA_RMP:** USART1_TX DMA 请求重映射位 由软件设置和清除该位。它控制着 USART1_TX DMA 通道请求的重映射 0: 无重映射(USART1_TX 请求映射在 DMA 通道 2 上) 1: 重映射 (USART1_TX 请求映射在 DMA 通道 4 上)
位 8	**ADC_DMA_RMP:** ADC DMA 请求重映射位 由软件设置和清除该位。它控制着 ADC DMA 通道请求的重映射 0: 无重映射(ADC DMA 请求映射在 DMA 通道 1 上) 1: 重映射 (ADC DMA 请求映射在 DMA 通道 2 上)
位 7	保留，必须保持复位值。
位 6:2	**LCD_CAPA:** 去耦电容连接 这些位由软件设置和清除。它们控制内部 VLCD 导轨供电电压与专用 I/O（LCD-VLCD1、LCD-VLCD2、LCD-VLCD3）的连接，以执行可选的解耦。 **位 1 控制 PB2/LCD 上 VLCD 导轨 1 的连接 VLCD1** 0:VLCD 导轨 1 未连接到 PB2/LCD 上 VLCD1 1:VLCD 导轨 1 连接到 PB2/LCD 上 VLCD1 **位 2 控制 PB12 上 VLCD 导轨 2 的连接** **位 3 控制 PB0 上 VLCD 导轨 3 的连接** **位 4 控制 PE11 上 VLCD 导轨 2 的连接** **位 5 控制 PE12 上 VLCD 导轨 3 的连接**

位 1:0	**MEM_MODE:** 存储映射选择位 由软件设置和清除这些位。它控制存储器内部映射到地址 0x0000 0000。当复位后这些位值由实际引导模式配置选择的值决定。 x0: 主闪存存储器映射到 0x0000 0000 01: 系统闪存映射到 0x0000 0000 11: 嵌入式 RAM 映射到 0x0000 0000

9.2.2 SYSCFG 外部中断配置寄存器 1(SYSCFG_EXTICR1)

偏移地址 ： 0x08

复位值 ： 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

位 31:16	保留，必须保持复位值。
位 15:12	**EXTI3:** EXTI 3 配置位 (EXTI 3 configuration bits) 这些位由软件进行改写来选择 EXTI3 的外部中断源。 x000: PA[x] 引脚 x001: PB[x] 引脚 x010: PC[x] 引脚 x011: PD[x] 引脚 x100: 保留 x101: PF[x] 引脚 其它配置：保留
位 11:8	**EXTI2:** EXTI 2 配置位 (EXTI 2 configuration bits) 这些位由软件进行改写来选择 EXTI2 的外部中断源。 x000: PA[x] 引脚 x001: PB[x] 引脚 x010: PC[x] 引脚 x011: PD[x] 引脚 x100: 保留 x101: PF[x] 引脚 其它配置：保留
位 7:4	**EXTI1:** EXTI 1 配置位 (EXTI 1 configuration bits) 这些位由软件进行改写来选择 EXTI1 的外部中断源。 x000: PA[x] 引脚 x001: PB[x] 引脚 x010: PC[x] 引脚 x011: PD[x] 引脚 x100: 保留 x101: PF[x] 引脚 其它配置：保留

位 3:0	**EXTI0:** EXTI 0 配置位 (EXTI 0 configuration bits) 这些位由软件进行改写来选择 EXTI0 的外部中断源。 x000: PA[x] 引脚 x001: PB[x] 引脚 x010: PC[x] 引脚 x011: PD[x] 引脚 x100: 保留 x101: PF[x] 引脚 其它配置：保留

9.2.3 SYSCFG 外部中断配置寄存器 2(SYSCFG_EXTICR2)

偏移地址 ： 0x0c

复位值 ： 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

位 31:16	保留，必须保持复位值。
位 15:12	**EXTI7:** EXTI 7 配置位 (EXTI 7 configuration bits) 这些位由软件进行改写来选择 EXTI7 的外部中断源。 x000: PA[x] 引脚 x001: PB[x] 引脚 x010: PC[x] 引脚 x011: PD[x] 引脚 x100: 保留 x101: PF[x] 引脚 其它配置：保留
位 11:8	**EXTI6:** EXTI 6 配置位 (EXTI 6 configuration bits) 这些位由软件进行改写来选择 EXTI6 的外部中断源。 x000: PA[x] 引脚 x001: PB[x] 引脚 x010: PC[x] 引脚 x011: PD[x] 引脚 x100: 保留 x101: PF[x] 引脚 其它配置：保留
位 7:4	**EXTI5:** EXTI 5 配置位 (EXTI 5 configuration bits) 这些位由软件进行改写来选择 EXTI5 的外部中断源。 x000: PA[x] 引脚 x001: PB[x] 引脚 x010: PC[x] 引脚 x011: PD[x] 引脚 x100: 保留 x101: PF[x] 引脚 其它配置：保留

位 3:0	**EXTI4:** EXTI 4 配置位 (EXTI 4 configuration bits) 这些位由软件进行改写来选择 EXTI4 的外部中断源。 x000: PA[x] 引脚 x001: PB[x] 引脚 x010: PC[x] 引脚 x011: PD[x] 引脚 x100: 保留 x101: PF[x] 引脚 其它配置：保留

9.2.4 SYSCFG 外部中断配置寄存器 3(SYSCFG_EXTICR3)

偏移地址 ： 0x10

复位值 ： 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

位 31:16	保留，必须保持复位值。
位 15:12	**EXTI11:** EXTI 11 配置位 (EXTI 11 configuration bits) 这些位由软件进行改写来选择 EXTI11 的外部中断源。 x000: PA[x] 引脚 x001: PB[x] 引脚 x010: PC[x] 引脚 x011: PD[x] 引脚 x100: 保留 x101: PF[x] 引脚 其它配置：保留
位 11:8	**EXTI10:** EXTI 10 配置位 (EXTI 10 configuration bits) 这些位由软件进行改写来选择 EXTI10 的外部中断源。 x000: PA[x] 引脚 x001: PB[x] 引脚 x010: PC[x] 引脚 x011: PD[x] 引脚 x100: 保留 x101: PF[x] 引脚 其它配置：保留
位 7:4	**EXTI9:** EXTI 9 配置位 (EXTI 9 configuration bits) 这些位由软件进行改写来选择 EXTI9 的外部中断源。 x000: PA[x] 引脚 x001: PB[x] 引脚 x010: PC[x] 引脚 x011: PD[x] 引脚 x100: 保留 x101: PF[x] 引脚 其它配置：保留

位 3:0	**EXTI8:** EXTI 8 配置位 (EXTI 8 configuration bits) 这些位由软件进行改写来选择 EXTI8 的外部中断源。 x000: PA[x] 引脚 x001: PB[x] 引脚 x010: PC[x] 引脚 x011: PD[x] 引脚 x100: 保留 x101: PF[x] 引脚 其它配置：保留

9.2.5 SYSCFG 外部中断配置寄存器 4(SYSCFG_EXTICR4)

偏移地址 ： 0x14

复位值 ： 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

位 31:16	保留，必须保持复位值。
位 15:12	**EXTI15:** EXTI 15 配置位 (EXTI 15 configuration bits) 这些位由软件进行改写来选择 EXTI15 的外部中断源。 x000: PA[x] 引脚 x001: PB[x] 引脚 x010: PC[x] 引脚 x011: PD[x] 引脚 x100: 保留 x101: PF[x] 引脚 其它配置：保留
位 11:8	**EXTI14:** EXTI 14 配置位 (EXTI 14 configuration bits) 这些位由软件进行改写来选择 EXTI14 的外部中断源。 x000: PA[x] 引脚 x001: PB[x] 引脚 x010: PC[x] 引脚 x011: PD[x] 引脚 x100: 保留 x101: PF[x] 引脚 其它配置：保留
位 7:4	**EXTI13:** EXTI 13 配置位 (EXTI 13 configuration bits) 这些位由软件进行改写来选择 EXTI13 的外部中断源。 x000: PA[x] 引脚 x001: PB[x] 引脚 x010: PC[x] 引脚 x011: PD[x] 引脚 x100: 保留 x101: PF[x] 引脚 其它配置：保留

位 3:0	**EXTI12:** EXTI 12 配置位 (EXTI 12 configuration bits) 这些位由软件进行改写来选择 EXTI12 的外部中断源。 x000: PA[x] 引脚 x001: PB[x] 引脚 x010: PC[x] 引脚 x011: PD[x] 引脚 x100: 保留 x101: PF[x] 引脚 其它配置：保留

9.2.6 SYSCFG 配置寄存器 2(SYSCFG_CFGR2)

偏移地址： 0x18

复位值： 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TIM3_CH4_REMAP		
													rw	rw	rw
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	SRAM_PEF	Res.	Res.	Res.	Res.	Res.	PVD_LOCK	SRAM_PARITY_LOCK	LOCKUP_LOCK
							rc_w1						rw	rw	rw
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位 31:19	保留，必须保持复位值。
位 18:16	**TIM3_CH4_REMAP:** TIMER3 CH4 重新映射控制位 () 000: GPIO 输入 001: LSI 时钟 010: 分频 HSI 时钟 011: LSE 时钟 100: USB SOF 101: 分频(1/8) MSI 时钟 110: 分频(1/64) MSI 时钟 111: 分频(1/512) MSI 时钟
位 15:9	保留，必须保持复位值。
位 8	**SRAM_PEF:** SRAM 校验标志 当 SRAM 校验错被检测到时硬件自动设置该位。对该位写 1 时清该位。 0: 无 SRAM 校验错 1: 检测到 SRAM 校验错
位 7:3	保留，必须保持复位值。
位 2	**PVD_LOCK:** PVD 锁定使能位 该位由软件设置，由系统复位清除。可用于使能并锁定 PVD 连接到 TIM1 的刹车(Break) 输入, 锁定 PVDE 和 PLS[2:0] (PWR_CR 寄存器) 0: PVD 中断信号断开与 TIM1 刹车(Break) 输入的连接。PVDE 和 PLS[2:0] 位可被应用编程。 1: PVD 中断信号连接到 TIM1 刹车(Break) 输入, PVDE 和 PLS[2:0] 位为只 读。

位 1	**SRAM_PARITY_LOCK:** SRAM 校验锁定位 该位由软件设置，由系统复位清除。可用于锁定 SRAM 校验错信号连接到 TIM1 的刹车(Break) 输入。 0: SRAM 校验错信号断开与 TIM1 刹车(Break) 输入的连接 1: SRAM 校验错信号连接到 TIM1 刹车(Break) 输入
位 0	**LOCKUP_LOCK:** Cortex-M0 LOCKUP 位使能位 该位由软件设置，由系统复位清除。它可用于使能并锁定连接 Cortex-M0 LOCKUP (硬件故障) 输出到 TIM1 的刹车(Break) 输入。 0: Cortex-M0 LOCKUP 输出断开与 TIM1 刹车(Break) 输入的连接 1: Cortex-M0 LOCKUP 输出连接到 TIM1 刹车(Break) 输入

9.2.7 SYSCFG 配置寄存器 3(SYSCFG_CFGR3)

偏移地址： 0x20

复位值： 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REF_LOCK	VREFINT_RDYF	Res.	Res.	Res.	Res.	Res.	Res.	EN_BUF_AIN19	Res.	EN_SW_AIN19	Res.	Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	ENREF_HSI48	ENBUF_VREFINT_COMP2	Res.	ENBUF_SENSOR_OUT	ENBUF_SENSOR_ADC	ENBUF_VREFINT_ADC	Res.	SEL_VREF_OUT			Res.	Res.	Res.	EN_VREFINT
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位 31	**REF_LOCK:** SYSCFG_CFGR3 锁定位 () 该位由软件设置，由硬件复位清除。它锁定了参考控制/状态寄存器 SYSCFG_CFGR3[31:0]的全部内容。 0: SYSCFG_CFGR3[31:0] 位是可读/写的 1: SYSCFG_CFGR3[31:0] 位是只读的
位 30	**VREFINT_RDYF:** VREFINT 读标志 这个位只读，它显示了内部基准电压 VREFINT 的状态。置位时，表明 VREFINT 可用于 BOR，PVD 和 LCD。 0: VREFINT 关闭 1: VREFINT 已就绪
位 29:24	保留，必须保持复位值。
位 23	**EN_BUF_AIN19:** 使能 ADC_AIN19 的 buffer 0: 使能 ADC_AIN19 的 buffer 1: 失能 ADC_AIN19 的 buffer
位 22	保留，必须保持复位值。
位 21	**EN_SW_AIN19:** 使能 ADC_IN19_WEAK_IN 的开关 0: 使能 ADC_IN19_WEAK_IN 的开关 1: 失能 ADC_IN19_WEAK_IN 的开关
位 20:14	保留，必须保持复位值。

位 13	**ENREF_HSI48:** HSI48 振荡器使能位的 VREFINT 参考 此位由软件设置和清除（仅当未设置 REF_LOCK 时）。 0: 用于为 HSI48 振荡器关闭而生成 VREFINT 参考的缓冲器. 1: 用于为 HSI48 振荡器打开而生成 VREFINT 参考的缓冲器.
位 12	**ENBUF_VREFINT_COMP2:** 比较器 2 的 VREFINT 参考使能位 (VREFINT reference for COMP2 scaler enable bit) 此位由软件设置和清除（仅当未设置 REF_LOCK 时）。 0: 用于生成比较器 2 的 VREFINT 参考的缓冲区关闭 . 1: 用于生成比较器 2 的 VREFINT 参考的缓冲区开启 .
位 11	保留，必须保持复位值。
位 10	**ENBUF_SENSOR_OUT:** SENSOR_OUT 开关的使能位 0: 使能 SENSOR_OUT 开关 1: 失能 SENSOR_OUT 开关
位 9	**ENBUF_SENSOR_ADC:** ADC 的温度传感器参考使能位(Temperature sensor reference for ADC enable bit) 此位由软件设置和清除（仅当未设置 REF_LOCK 时）。置位时，VREFINT 自动使能。 0: 用于生成 ADC 的温度传感器参考的缓冲区关闭 . 1: 用于生成 ADC 的温度传感器参考的缓冲区开启 .
位 8	**ENBUF_VREFINT_ADC:** ADC 的 VREFINT 参考使能位(VREFINT reference for ADC enable bit) 此位由软件设置和清除（仅当未设置 REF_LOCK 时）。 0: 禁用用于为 ADC 生成 VREFINT 参考的缓冲区. 1: 使能用于为 ADC 生成 VREFINT 参考的缓冲器.
位 7	保留，必须保持复位值。
位 6:4	**SEL_VREF_OUT:** VREFINT_ADC 连接位 (VREFINT_ADC connection bit) 这些位由软件设置和清除（仅当未设置 REF_LOCK 时）。当设置 ENBUF_VREFINT_ADC 时，这些位选择哪个焊盘连接到 VREFINT_ADC。 00: 未连接焊盘 01: PB0 已连接 10: PB1 已连接 11: PB0 和 PB1 已连接
位 3:1	保留，必须保持复位值。
位 0	**EN_VREFINT:** VREFINT enable and scaler control for COMP2 enable bit 此位由软件设置和清除（仅当未设置 REF_LOCK 时）。它打开 VREFINT 内部参考电压，并启用 COMP2 的定标器。 0: 低功耗模式（如果 ULP = 1）禁用 VREFINT 电压，禁用 COMP2 的缩放器 1: 低功耗模式下启用 VREFINT 电压，COMP2 启用定标器

注：如果设备处于停止模式或睡眠/低功率睡眠模式，则禁止同时配置 EN_VREFINT=1 和 ULP=1。如果设备未处于低功耗模式，则无论 EN_VREFINT 和 ULP 的状态如何，VREFINT 始终处于启用状态。EN_VREFINT 仅控制 COMP2 定标器。

9.2.8 奇偶校验设置寄存器 (PAR_SET)

偏移地址 ： 0x3fc

复位值 ： 0x0000 0000

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

DATA[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位 31:0	**DATA:** 向该寄存器写入 1acc55aa 设置校验标志 SRAM_PEF 位

10 直接存储器访问控制器 (DMA)

10.1 介绍

直接存储读取器(DMA)用于在外设与存储器之间或是内存到内存的高速数据传输。数据可以在不占用任何 CPU 资源的情况下快速地从指定的源地址搬移到目标地址，以便让 CPU 有更多的资源去处理其它应用。

DMA 控制器拥有 7 个通道，每个通道都专门用于管理来自于一个或多个外设的存储访问请求. 同时它拥有一个仲裁器去处理不同的 DMA 请求的优先级。

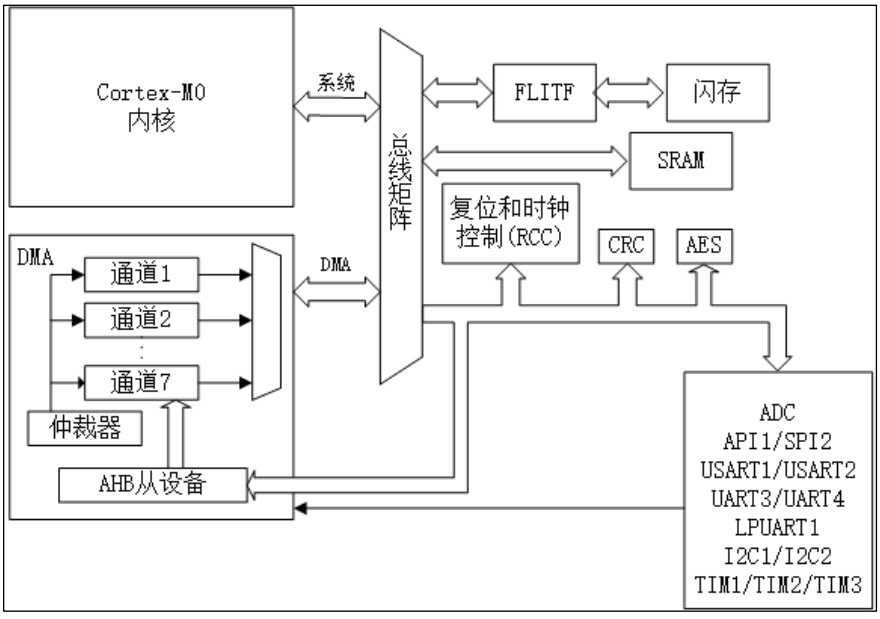
10.2 DMA 的主要功能

- 多达 7 个独立的可配置通道(请求).
- 第个通道与专用的硬件 DMA 请求相连，同时每个通道也都支持软件触发. 这些配置都需要由软件完成。
- 不同通道的 DMA 请求优先级可以由软件进行配置(共 4 个可配的优先等级：很高，高，中，低)；当某些通道的 DMA 请求优先级配置相同时，则这些通道 DMA 请求的优先级由硬件来决定(例如：同等优先级配置的情况下，通道 1 的 DMA 请求高于通道 2 的 DMA 请求)。
- 独立可配的源以及目标传输位宽(字节，半字，字)，模拟封包和拆包. 源/目标地址必须以传输数据的位宽作为最小单位来对齐。
- 支持循环缓冲管理。
- 每个通道支持 3 个事件标志(DMA 完成一半传输标志, DMA 传输完成标志, DMA 传输错误标志)以逻辑或的关系为每个通道请求对应的中断。
- 支持内存到内存传输模式。
- 支持外设到内存，内存到外设，外设到外设传输模式。
- 支持将 Flash, SRAM, APB 或 AHB 总线上的外设配置为源或目标。
- 传输数据的个数(或者说每个通道上 DMA 传输的次数)可配置：最多可配置到 65535。

10.2.1 功能说明

DMA 的结构框图如下：

图 10-1 DMA 结构图



DMA 控制器通过与 Cortex-M0+内核共享系统总线完成直接存储传输功能，当 CPU 和 DMA 访问同一个

目标地址的时候(内存或外设)，DMA 请求可能会让 CPU 访问系统总线的某些循环暂停。总线矩阵实现了循环机制，以保证至少有一半的系统总线带宽(对于内存和外设都有效)供给 CPU。

10.2.2 DMA 传输(DMA transactions)

发生一次事件之后，外设会发送一个请求信号给 DMA 控制器。DMA 控制器根据对应通道的配置来处理收到的这个请求。在 DMA 控件器访问外设的同时，DMA 控制器也会发送一个应答给外设。外设一旦发到 DMA 控制器应答的信号就会释放当前的 DMA 请求。一旦外设的 DMA 请求失效，DMA 控制器也会释放掉应答信号。如果有很多的请求，外设可以发起下一次传输。

归纳一下就是每个 DMA 传输由三个操作组成:

- 根据当前外设/内存地址寄存器中的值从外设的数据寄存器或者指定地址的内存加载数据。用于第一次传输的起始地址通过对 DMA_CPARx 或 DMA_CMARx 寄存器编程确定。
- 根据当前外设/内存地址寄存器中的值，将上一步中加载得到的数据加载到指定的外设寄存器或者指定的地址中。用于第一次传输的起始地址是通过对 DMA_CPARx 或 DMA_CMARx 编程来确定的。
- 用于存剩余传输次数的 DMA_CNDTRx 做减计数。

10.2.3 仲裁器(Arbiter)

仲裁器根据通道优先级管理 DMA 的通道请求并执行外设/内存访问序列。

DMA 通道间的优先级规则有两种:

软件可配的优先级：DMA 的每个通道可以通过对 DMA_CCRx 寄存器编程配置成以下四种优先级之一:

- 很高的优先级
- 高优先级
- 中优先级
- 低优先级

硬件默认优先级：如果两个通道配成了相同的软件优先级，则序号较小的通道具有更高的优先级，例如通道 2 的优先级高于通道 4 的优先级。

10.2.4 DMA 通道(DMA channels)

每个 DMA 通道都能完成在固定的外设地址与内存地址之间的 DMA 传输。传输数据的总量(最大 65535)可以编程控制。每完成一次传输，存储剩余传输次数的寄存器就会进行一次减计数。

可编程的数据传输位宽(Programmable data size)

外设和内存传输的数据宽度可以通过对 DMA_CCRx 寄存器中的 PSIZE 以及 MSIZE 进行编程确定。

指针自加(Pointer incrementation)

通过对 DMA_CCRx 寄存器中的 PINC 和 MINC 进行编程可以将外设和内存指针设置为每次传输后自动增加(如果使能了自增模式，每次增加的值与位宽有关,如果位宽设置为字节,则加 1,如果位宽设置为半字则加 2,字则加 3)。第一次传输所使用到的外设或内存地址都由软件对 DMA_CPARx 和 DMA_CMARx 寄存器进行编程确定。DMA 传输过程中，DMA_CPARx 和 DMA_CMARx 寄存器值保持为初次配置的值，因此通道的外设或内存地址自增功能使能时软件是无法访问到当前传输对应的外设或者内存地址的。

如果通道被配置为非循环模式，那么在完成了最后一次传输之后(存储剩余传输次数的寄存器 DMA_CNDTRx 值变为 0)，就不会再处理该通道上的 DMA 请求。如果要加载新的值到 DMA_CNDTRx 寄存器中，对应的 DMA 通道必须先关闭。

注意: 如果一个 DMA 通道关闭了，DMA 寄存器不会被复位。DMA_CCRx，DMA_CPARx 和 DMA_CMARx 保持之前配置的值。

如果通道被配置为循环工作模式，在最后一次传输完成后，DMA_CNDTRx 寄存器会自动加载初始配置的值。当前内部地址寄存器也会重新加载 DMA_CPARx 和 DMA_CMARx 寄存器中的值。

DMA 通道配置步骤(Channel configuration procedure)

请遵循以下顺序去配置一个 DMA 通道 x(x 代表通道编号)。

1. 配置 DMA_CPARx 寄存器设置首次传输的外设地址。传输时会根据这个寄存器中的值将数据从这个地址对应的外设寄存器中般出或者是从内存中将数据搬入到这个地址对应的外设寄存器。

2. 配置 DMA_CMARx 寄存器，配置首次传输的内存地址。传输时会根据这个寄存器中的值将数据从内存中这个地址搬出或者从外设(或其他地址的内存)将数据搬入。
3. 配置 DMA_CNDTRx 寄存器，配置总共要传输的数据个数，每完成一个数据的传输，DMA_CNDTRx 的值会减 1。
4. 通过配置 DMA_CCRx 寄存器中的 PL[1:0]配置该通道的优先级。
5. 配置 DMA_CCRx 寄存器，配置数据传输方向，循环模式，外设和内存地址自增模式，外设和内存的数据传输位宽，以及是否在传输完成一半或传输完成时产生中断。
6. 通过置位 DMA_CCRx 寄存器中的 ENABLE 位激活该通道。

通道一旦被激活，就可以处理来自与该通道相连外设的 DMA 请求了。

当已完成的传输个数达到设定的总数一半时，半传输标志(HTIF)置位，此时如果半传输中断使能位(HTIE)为'1'，则会产生一个中断。当所有传输完成时，传输完成标志(TCIF)置位，此时如果传输完成中断使能位(TCIE)为'1'，则会产生一个传输完成中断。

循环模式(Circular mode)

循环模式用于循环地向一个指定大小的数据缓存空间中搬移数据(数据流传输，如在 ADC 扫描模式下可能会用到)。循环模式可以通过配置 DMA_CCRx 寄存器中的 CIRC 位为'1'为使能。循环模式激活的情况下，每当所有指定个数的数据传输完成后，DMA_CNDTRx 的值会自动地加载为初始配置时的值，对应的通道也会一直处理 DMA 请求。

内存到内存传输模式(Memory-to-memory mode)

DMA 通道也可以在没有被外设请求触发的情况下工作，这种模式叫做内存到内存传输模式。

如果 DMA_CCRx 寄存器中 MEM2MEM 位为'1'，则一旦 DMA_CCRx 中的 EN 位置位，该通道上的 DMA 传输就会立即启动。当 DMA_CNDTRx 寄存器值变为 0 时，该通道的数据传输停止。内存到内存传输模式不能与循环模式一起使用。

10.2.5 可编程的数据位宽，数据对齐及字节顺序(Progra-mmable data width, data alignment and endianness)

当 PSIZE 和 MSIZE 配置不一致时，DMA 会按下表描述进行数据对齐：可编程数据宽度、字节存储次序(位 PINC = MINC = 1 时)所述方式进行对齐。

表 10-1 可编程的数据宽度、字节存储次序(位 PINC = MINC = 1 时)

源端口宽度	目标端口宽度	要传输的数据项的数目(NDT)	源内容:地址/数据	传输操作	目标内容:地址/数据
8	8	4	@0x0/B0 @0x1/B1 @0x2/B2 @0x3/B3	1:读取 B0[7:0]@0x0 然后写入 B0[7:0]@0x0 2:读取 B1[7:0]@0x0 然后写入 B1[7:0]@0x1 3:读取 B2[7:0]@0x0 然后写入 B2[7:0]@0x2 4:读取 B3[7:0]@0x0 然后写入 B3[7:0]@0x3	@0x0/B0 @0x1/B1 @0x2/B2 @0x3/B3
8	16	4	@0x0/B0 @0x1/B1 @0x2/B2 @0x3/B3	1:读取 B0[7:0]@0x0 然后写入 00B0[15:0]@0x0 2:读取 B1[7:0]@0x1 然后写入 00B1[15:0]@0x2 3:读取 B2[7:0]@0x2 然后写入 00B2[15:0]@0x4 4:读取 B3[7:0]@0x3 然后写入 00B3[15:0]@0x6	@0x0/00B0 @0x2/00B1 @0x4/00B2 @0x6/00B3

8	32	4	@0x0 / B0 @0x1 / B1 @0x2 / B2 @0x3 / B3	1: 读取 B0[7:0] @0x0 然后写入 000000B0[31:0] @0x0 2: 读取 B1[7:0] @0x1 然后写入 000000B1[31:0] @0x4 3: 读取 B2[7:0] @0x2 然后写入 000000B2[31:0] @0x8 4: 读取 B3[7:0] @0x3 然后写入 000000B3[31:0] @0xC	@0x0 / 000000B0 @0x4 / 000000B1 @0x8 / 000000B2 @0xC / 000000B3
16	8	4	@0x0 / B1B0 @0x2 / B3B2 @0x4 / B5B4 @0x6 / B7B6	1: 读取 B1B0[15:0] @0x0 然后写入 B0[7:0] @0x0 2: 读取 B3B2[15:0] @0x2 然后写入 B2[7:0] @0x1 3: 读取 B5B4[15:0] @0x4 然后写入 B4[7:0] @0x2 4: 读取 B7B6[15:0] @0x6 然后写入 B6[7:0] @0x3	@0x0 / B0 @0x1 / B2 @0x2 / B4 @0x3 / B6
16	16	4	@0x0 / B1B0 @0x2 / B3B2 @0x4 / B5B4 @0x6 / B7B6	1: 读取 B1B0[15:0] @0x0 然后写入 B1B0[15:0] @0x0 2: 读取 B3B2[15:0] @0x2 然后写入 B3B2[15:0] @0x2 3: 读取 B5B4[15:0] @0x4 然后写入 B5B4[15:0] @0x4 4: 读取 B7B6[15:0] @0x6 然后写入 B7B6[15:0] @0x6	@0x0 / B1B0 @0x2 / B3B2 @0x4 / B5B4 @0x6 / B7B6
16	32	4	@0x0 / B1B0 @0x2 / B3B2 @0x4 / B5B4 @0x6 / B7B6	1: 读取 B1B0[15:0] @0x0 然后写入 0000B1B0[31:0] @0x0 2: 读取 B3B2[15:0] @0x2 然后写入 0000B3B2[31:0] @0x4 3: 读取 B5B4[15:0] @0x4 然后写入 0000B5B4[31:0] @0x8 4: 读取 B7B6[15:0] @0x6 然后写入 0000B7B6[31:0] @0xC	@0x0 / 0000B1B0 @0x4 / 0000B3B2 @0x8 / 0000B5B4 @0xC / 0000B7B6
32	8	4	@0x0 / B3B2B1B0 @0x4 / B7B6B5B4 @0x8 / BBBAB9B8 @0xC / BFBEBDBC	1: 读取 B3B2B1B0[31:0] @0x0 然后写入 B0[7:0] @0x0 2: 读取 B7B6B5B4[31:0] @0x4 然后写入 B4[7:0] @0x1 3: 读取 BBBAB9B8[31:0] @0x8 然后写入 B8[7:0] @0x2 4: 读取 BFBEBDBC[31:0] @0xC 然后写入 BC[7:0] @0x3	@0x0 / B0 @0x1 / B4 @0x2 / B8 @0x3 / BC
32	16	4	@0x0 / B3B2B1B0 @0x4 / B7B6B5B4 @0x8 / BBBAB9B8 @0xC / BFBEBDBC	1: 读取 B3B2B1B0[31:0] @0x0 然后写入 B1B0[15:0] @0x0 2: 读取 B7B6B5B4[31:0] @0x4 然后写入 B5B4[15:0] @0x2 3: 读取 BBBAB9B8[31:0] @0x8 然后写入 B9B8[15:0] @0x4 4: 读取 BFBEBDBC[31:0] @0xC 然后写入 BDBC[15:0] @0x6	@0x0 / B1B0 @0x2 / B5B4 @0x4 / B9B8 @0x6 / BDBC

32	32	4	@0x0 / B3B2B1B0 @0x4 / B7B6B5B4 @0x8 / BBBAB9B8 @0xC / BFBEBDBC	1: 读取 B3B2B1B0[31:0] @0x0 然后写入 B3B2B1B0[31:0] @0x0 2: 读取 B7B6B5B4[31:0] @0x4 然后写入 B7B6B5B4[31:0] @0x4 3: 读取 BBBAB9B8[31:0] @0x8 然后写入 BBBAB9B8[31:0] @0x8 4: 读取 BFBEBDBC[31:0] @0xC 然后写入 BFBEBDBC[31:0] @0xC	@0x0 / B3B2B1B0 @0x4 / B7B6B5B4 @0x8 / BBBAB9B8 @0xC / BFBEBDBC

解决 AHB 外设无法支持字节或半字写操作的问题

如果 DMA 初始化了 AHB 字节或半字写操作，则会将数据复制到 HWDATA[31:0]总线的未使用通道上。因此，若所用的 AHB 从外设不支持字节或半字定操作(即外设未使用用 HSIZE)且不会产生任何错误，则 DMA 会对 HWDATA 的 32 个位执行写操作，如下两个示例所示:

- 要写入半字 ”0xABCD”，则 DMA 会将 HWDATA 总线设为”0xABCDABCD”，同时将 HSIZE 设为 HalfWord
- 要写入字节”0xAB”，则 DMA 会将 HWDATA 总线设为”0xABABABAB”，同时将 HSIZE 设为 Byte

假设 AHB/APB 桥为 AHB 32 位从外设，且该外设未设置数据的 HSIZE，则任何 AHB 字节或半字操作都将转换为 32 位 APB 操作，转换方式如下所示:

- 将 AHB 字节写操作转化为 APB 字写操作，如向 0x0(或 0x1、0x2、0x3)写入数据”0xB0”转化为向 0x0 写入数据”0xB0B0B0B0”
- 将 AHB 半字写操作转化为 APB 字写操作，如向 0x0(或 0x2)写入数据”0xB1B0”转化为向 0x0 写入数据”0xB1B0B1B0”

例如，若用户希望对 APB 备份寄存器(与 32 位地址边界对齐的 16 位寄存器)执行写操作，则必须将存储器源大小(MSIZE)配置为”16 位”，同时将外设目标大小(PSIZE)配置为”32 位”。

10.2.6 错误管理

当对保留的地址空间执行读取操作时，将生成 DMA 传输错误。若在 DMA 读或写访问过程中生成了 DMA 传输错误，则会将相应的通道配置寄存器(DMA_CCRx)的 EN 位进行硬件清零，从而自动禁止出错的通道。如果 DMA_CCRx 寄存器中的传输错误中断使能位(TEIE)置 1，则 DMA_IFR 寄存器中该通道的传输错误中断标志(TEIF)将置 1 并生成中断。

10.2.7 DMA 中断

对于每个 DMA 通道，在发生”半传输”、”传输完成”或”传输错误”时都可以产生中断。可以使用单独的中断使能位以提高灵活性。

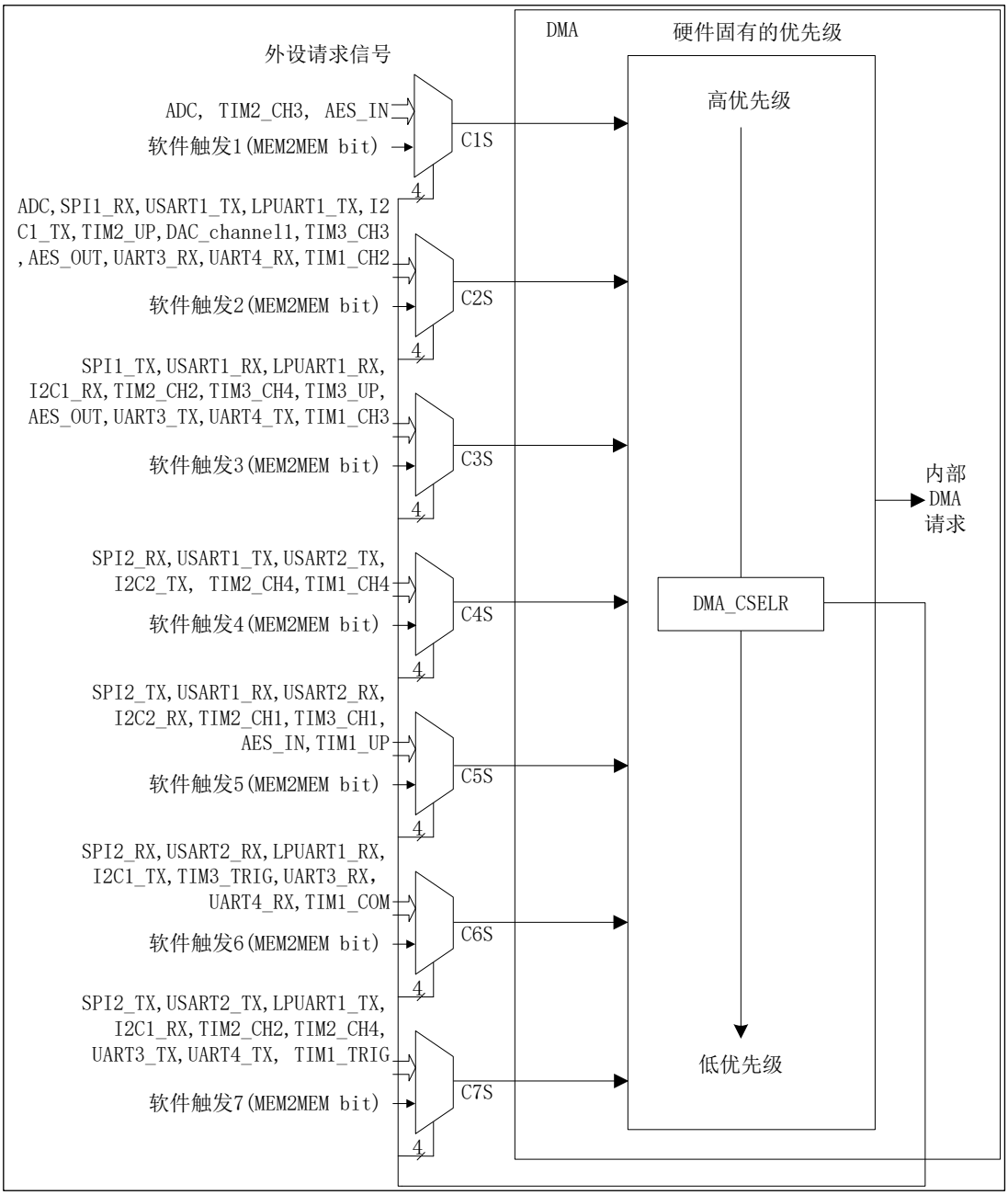
表 10-2 DMA 中断请求

中断事件	中断标志	使能控制位
半传输	HTIF	HTIE
传输完成	TCIF	TCIE
传输错误	TEIF	TEIE

10.2.8 DMA 请求映射

来自外设(TIM2/6、ADC、DAC、SPI1/2、I2C1/2、AES(仅限具有 AES 的第 3 类和第 5 类器件)、USART1/2 以及 LPUART1)的各硬件请求可通过 DMA 通道选择寄存器映射到 DMA 通道(1 到 7)。在一个通道上，一次只能响应一个请求(参考下图: DMA 请求映射)。外设 DMA 请求可对相应外设的寄存器的 DMA 控制位进行编程，从而单独进行激活或取消激活。

图 10-2 DMA 请求映射



下表列出了各通道的 DMA 请求:

表 10-3 每个通道的 DMA 请求归纳

请求编号	外设	通道 1	通道 2	通道 3	通道 4	通道 5	通道 6	通道 7
0	ADC	ADC	ADC	-	-	-	-	-
1	SPI1	-	SPI1_RX	SPI1_TX	-	-	-	-
2	SPI2	-	-	-	SPI2_RX	SPI2_TX	SPI2_RX	SPI2_TX
3	USART1	-	USART1_TX	USART1_RX	USART1_TX	USART1_RX	-	-
4	USART2	-	-	-	USART2_TX	USART2_RX	USART2_RX	USART2_TX
5	LPUART1	-	LPUART1_TX	LPUART1_RX	-	-	LPUART1_RX	LPUART1_TX

6	I2C1	-	I2C1_TX	I2C1_RX	-	-	I2C1_TX	I2C1_RX
7	I2C2	-	-	-	I2C2_TX	I2C2_RX	-	-
8	TIM2	TIM2_CH3	TIM2_UP	TIM2_CH2	TIM2_CH4	TIM2_CH1	-	TIM2_CH2 TIM2_CH4
9	DAC	-	DAC_CH1	-	-	-	-	-
10	TIM3	-	TIM3_CH3	TIM3_CH4 TIM3_UP	-	TIM3_CH1	TIM3_TRIG	-
11	AES(1)	AES_IN	AES_OUT	AES_OUT	-	AES_IN	-	-
12	UART3	-	UART3_RX	UART3_TX	-	-	UART3_RX	UART3_TX
13	UART4	-	UART4_RX	UART4_TX	-	-	UART4_RX	UART4_TX
14	TIM1	TIM1_CH1	TIM1_CH2	TIM1_CH3	TIM1_CH4	TIM1_UP	TIM1_COM	TIM1_TRIG

10.3 DMA 寄存器

10.3.1 DMA 中断状态寄存器(DMA_ISR)

DMA 中断状态寄存器(只读)

偏移地址: 0x00

复位值: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	TEIF7	HTIF7	TCIF7	GIF7	TEIF6	HTIF6	TCIF6	GIF6	TEIF5	HTIF5	TCIF5	GIF5
				r	r	r	r	r	r	r	r	r	r	r	r
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
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
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位 31:28	保留: 必须保持复位值
位 27,23,19,15,11,7,3	**TEIFx:** 通道 x 传输错误标志(x=1..7)。此位由硬件置 1，由软件清零(软件只需将 1 写入 DMA_IFCR 寄存 器的相应位即可)。 0: 通道 x 上无传输错误 1: 通道 x 上出现传输错误
位 26,22,18,14,10,6,2	**HTIFx:** 通道 x 半传输标志(x=1..7)。此位由硬件置 1，由软件清零(软件只需将 1 写入 DMA_IFCR 寄存器的 相应位即可)。 0: 通道 x 上无半传输事件 1: 通道 x 上发生半传输事件
位 25,21,17,13,9,15,1	**TCIFx:** 通道 x 传输完成标志(x=1..7)。此位由硬件置 1，由软件清零(软件只需将 1 写入 DMA_IFCR 寄存器的相应位即可)。 0: 通道 x 上无传输完成事件 1: 通道 x 上发生传输完成事件
位 24,20,16,12,8,4,0	**GIFx:** 通道 x 全局中断标志(x=1..7)。此位由硬件置 1，由软件清零(软件只需将 1 写入 DMA_IFCR 寄存器的相应位即可)。 0: 通道 x 上无 TE、HT 或 TC 事件中的任何一个 1: 通道 x 上发生了 TE、HT、或 TC 事件

10.3.2 DMA 中断标志清零寄存器(DMA_IFCR)

DMA 中断标志清零寄存器(只写)
偏移地址: 0x04
复位值: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	CTEIF7	CHTIF7	CTCIF7	CGIF7	CTEIF6	CHTIF6	CTCIF6	CGIF6	CTEIF5	CHTIF5	CTCIF5	CGIF5
				w	w	w	w	w	w	w	w	w	w	w	w
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CTEIF4	CHTIF4	CTCIF4	CGIF4	CTEIF3	CHTIF3	CTCIF3	CGIF3	CTEIF2	CHTIF2	CTCIF2	CGIF2	CTEIF1	CHTIF1	CTCIF1	CGIF1
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位 31:28	保留:必须保持复位值
位 27,23,19,15,11,7,3	**CTEIFx:** 通道 x 传输错误标志清零(x=1..7)。此位由软件置 1. 0: 无操作。 1: 将 DMA_ISR 寄存器中对应的 TEIF 标志清零。
位 26,22,18,14,10,6,2	**CHTIFx:** 通道 x 半传输标志清零(x=1..7)。此位由软件置 1. 0: 无操作。 1: 将 DMA_ISR 寄存器中对应的 HTIF 标志清零。
位 25,21,17,13,9,15,1	**CTCIFx:** 通道 x 传输完成标志清零(x=1,,7)。此位由软件置 1. 0: 无操作。 1: 将 DMA_ISR 寄存器中对应的 TCIF 标志清零。
位 24,20,16,12,8,4,0	**CGIFx:** 通道 x 全局中断标志清零(x=1..7)。此位由软件件置 1。 0: 无操作。 1: 将 DMA_ISR 寄存器中对应的 GIF 标志位清零。

10.3.3 DMA 通道 x 配置寄存器(DMA_CCRx)(x=1..7,其中 x 代表通道编号)

DMA 通道 x 配置寄存器。
偏移地址: 0x08 + 0D20 x (通道编号 - 1)
复位值: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	MEM2MEM	PL[1:0]		MSIZE[1:0]		PSIZE[1:0]		MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位 31:15	保留: 必须保持复位值
位 14	**MEM2MEM:** 存储器到存储器模式。 0: 不使能存储器到存储器模式。 1: 使能存储器到存储器模式。

位 13:12	**PL[1:0]:** 通道优先级配置位。 00: 低优先级。 01: 中优先级。 10: 高优先级。 11: 非常高的优先级。
位 11:10	**MSIZE[1:0]:** 存储器端数据位宽。 00: 8 位。 01: 16 位。 10: 32 位。 11: 保留。
位 9:8	**PSIZE[1:0]:** 外设端数据位宽。 00: 8 位。 01: 16 位。 10: 32 位。 11: 保留。
位 7	**MINC:** 存储器地址递增模式控制。 0: 不使能存储器地址递增模式。 1: 使能存储器地址递增模式。
位 6	**PINC:** 外设地址递增模式控制。 0: 不使能外设地址递增模式。 1: 使能外设地址递增模式。
位 5	**CIRC:** 循环传输模式控制。 0: 不使能循环传输模式。 1: 使能循环传输模式。
位 4	**DIR:** 数据传输方向控制。 0: 从外设读取数据。 1: 从存储器读取数据。
位 3	**TEIE:** 传输错误中断使能控制。 0: 不使能传输错误中断。 1: 使能传输错误中断。
位 2	**HTIE:** 半传输中断使能控制。 0: 不使能会传输中断。 1: 使能半传输中断。
位 1	**TCIE:** 传输完成中断使能控制。 0: 不使能传输完成中断。 1: 使能传输完成中断。
位 0	**EN:** 通道使能控制。 0: 不使能当前 DMA 通道(关闭通道)。 1: 使能当前 DMA 通道(打开通道)。

10.3.4 DAM 通道 x 数据数寄存器(DMA_CNDTRx)(x=1..7, 其中 x 代表通道编号)

DMA 通道 x 数据传输数量寄存器。

偏移地址: 0x0C + 0D20 x (通道编号 - 1)

复位值: 0x0000 0000

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

rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位 31:16	保留: 必须保持复位值
位 15:0	**NDT[15:0]:** 当前通道待传输的数据个数(0 至 65535)。只有在通道关闭(非使能状态)时才能对该寄存器执行写操作。通道使能后，该寄存器为只读，用于指示待传输的剩余传输次数。DMA 每执行一次传输，该寄存器的值减 1。传输完成后，该寄存器的值可以保持为零，若已将通道配置为循环模式，则自动重新加载先前编程的值。若该寄存器的值为 0，则无论该通道是否使能，都不会处理任何事务。

10.3.5 DAM 通道 x 外设地址寄存器(DMA_CPARx)(x=1..7，其中 x 代表通道编号)

DMA 通道 x 外设地址寄存器。

偏移地址: 0x10 + 0D20 x (通道编号 - 1)

复位值: 0x0000 0000

注意: 在通道使能的情况下不能对该寄存器进行写操作。

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

位 31:0	**PA[31:0]:** 外设地址。读/写数据的外设数据寄存器的基地址。 若 PSIZE[1:0] = [0:1](16 位)，则忽略 PA[0]位，访问将自动对齐到半字地址。 若 PSIZE[1:0] = [10](32 位)，则忽略 PA[1:0]位，访问将自动对齐到字地址。

10.3.6 DAM 通道 x 存储器址寄存器(DMA_CMARx)(x=1..7，其中 x 代表通道编号)

DMA 通道 x 外设地址寄存器。

偏移地址: 0x14 + 0D20 x (通道编号 - 1)

复位值: 0x0000 0000

注意: 在通道使能的情况下不能对该寄存器进行写操作。

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

位 31:0	**MA[31:0]:** 存储器地址。读/写数据的存储器的基地址。 若 MSIZE[1:0] = [0:1](16 位)，则忽略 PA[0]位，访问将自动对齐到半字地址。 若 MSIZE[1:0] = [10](32 位)，则忽略 PA[1:0]位，访问将自动对齐到字地址。

10.3.7 DMA 通道选择寄存器(DMA_CSELR)

DMA 通道选择寄存器。

偏移地址: 0xA8

复位值: 0x0000 0000

该寄存器用于管理 DMA 通道的重映射。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	C7S[3:0]				C6S[3:0]				C5S[3:0]			
				r	r	r	r	r	r	r	r	r	r	r	r
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
C4S[3:0]				C3S[3:0]				C2S[2:0]				C1S[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位 31:28	保留:必须保持复位值
位 27:24	**C7S[3:0]:** DMA 通道 7 选择位。 0010: DMA 通道 7 重映射到 SPI2_TX 0100: DMA 通道 7 重映射到 USART2_TX 0101: DMA 通道 7 重映射到 LPUART1_TX 0110: DMA 通道 7 重映射到 I2C1_RX 1000: DMA 通道 7 重映射到 TIM2_CH2/TIM2_CH4 1100: DMA 通道 7 重映射到 UART3_TX 1101: DMA 通道 7 重映射到 UART4_TX 1110: DMA 通道 7 重映射到 TIM1_TRIG 其他值: DMA 通道 7 不映射到外设。
位 23:20	**C6S[3:0]:** DMA 通道 6 选择位。 0010: DMA 通道 6 重映射到 SPI2_RX 0100: DMA 通道 6 重映射到 USART2_RX 0101: DMA 通道 6 重映射到 LPUART1_RX 0110: DMA 通道 6 重映射到 I2C1_TX 1010: DMA 通道 6 重映射到 TIM3_TRIG 1100: DMA 通道 6 重映射到 UART3_RX 1101: DMA 通道 6 重映射到 UART4_RX 1110: DMA 通道 6 重映射到 TIM1_COM 其他值: DMA 通道 6 不映射到外设。
位 19:16	**C5S[3:0]:** DMA 通道 5 选择位。 0010: DMA 通道 5 重映射到 SPI2_TX 0011: DMA 通道 5 重映射到 USART1_RX 0100: DMA 通道 5 重映射到 USART2_RX 0111: DMA 通道 5 重映射到 I2C2_RX 1000: DMA 通道 5 重映射到 TIM2_CH1 1010: DMA 通道 5 重映射到 TIM3_CH1 1011: DMA 通道 5 重映射到 AES_IN 1110: DMA 通道 5 重映射到 TIM1_UP 其他值: DMA 通道 5 不映射到外设。

位 15:12	**C4S[3:0]:** DMA 通道 4 选择位。 0010: DMA 通道 4 重映射到 SPI2_RX 0011: DMA 通道 4 重映射到 USART1_TX 0100: DMA 通道 4 重映射到 USART2_TX 0111: DMA 通道 4 重映射到 I2C2_TX 1000: DMA 通道 4 重映射到 TIM2_CH4 1110: DMA 通道 4 重映射到 TIM1_CH4 其他值: DMA 通道 4 不映射到外设。
位 11:8	**C3S[3:0]:** DMA 通道 3 选择位。 0001: DMA 通道 3 重映射到 SPI1_TX 0011: DMA 通道 3 重映射到 USART1_RX 0101: DMA 通道 3 重映射到 LPUART1_RX 0110: DMA 通道 3 重映射到 I2C1_RX 1000: DMA 通道 3 重映射到 TIM2_CH2 1010: DMA 通道 3 重映射到 TIM3_CH4/TIM3_UP 1011: DMA 通道 3 重映射到 AES_OUT 1100: DMA 通道 3 重映射到 UART3_TX 1101: DMA 通道 3 重映射到 UART4_TX 1110: DMA 通道 3 重映射到 TIM1_CH3 其他值: DMA 通道 3 不映射到外设。
位 7:4	**C2S[3:0]:** DMA 通道 2 选择位。 0000: DMA 通道 2 重映射到 ADC 0001: DMA 通道 2 重映射到 SPI1_RX 0011: DMA 通道 2 重映射到 USART1_TX 0101: DMA 通道 2 重映射到 LPUART1_TX 0110: DMA 通道 2 重映射到 I2C1_TX 1000: DMA 通道 2 重映射到 TIM2_UP 1001: DMA 通道 2 重映射到 DAC_CH1 1010: DMA 通道 2 重映射到 TIM3_CH3 1011: DMA 通道 2 重映射到 AES_OUT 1100: DMA 通道 2 重映射到 UART3_RX 1101: DMA 通道 2 重映射到 UART4_RX 1110: DMA 通道 2 重映射到 TIM1_CH2 其他值: DMA 通道 2 不映射到外设。
位 3:0	**C1S[3:0]:** DMA 通道 1 选择位。 0000: DMA 通道 1 重映射到 ADC 1000: DMA 通道 1 重映射到 TIM2_CH3 1011: DMA 通道 1 重映射到 AES_IN 1110: DMA 通道 1 重映射到 TIM1_CH1 其他值: DMA 通道 1 不映射到外设。

11 嵌套向量中断控制器 (NVIC)

11.1 NVIC 主要特性

嵌套向量中断控制器(NVIC) 和处理器核的接口紧密相连，可以实现低延迟的中断处理和高效地处理晚到的中断。嵌套向量中断控制器管理着包括内核异常等中断。

- 32 个可屏蔽中断通道(不包含16 个Cortex-M0 的中断线)
- 4 个可编程的优先级(使用了2 位的中断优先级)
- 低延时的异常和中断处理
- 电源管理控制
- 系统控制寄存器的实现

11.2 SysTick 校准值寄存器

SysTick 校准值设置为6000，当SysTick 时钟设置为6 MHz (fHCLK/8 的最大值) 时，会产生 1ms 时间基准。

11.3 中断和异常向量

Posit	Priority		Acronym	Interrupt vectors	Description	Address
-	-	-	-		Reserved	0x0000_0000
-	-3	fixed	Reset		Reset	0x0000_0004
-	-2	fixed	NMI		Non maskable interrupt. The RCC Clock Security System (CSS) is linked to the NMI vector.	0x0000_0008
-	-1	fixed	HardFault		All class of fault	0x0000_000c
-	3	settable	SVCall		System service call via SWI instruction	0x0000_002c
-	5	settable	PendSV		Pendable request for system service	0x0000_0038
-	6	settable	SysTick		System tick timer	0x0000_003c
0	7	settable	WWDG	WWDG_IRQHandler	Window watchdog interrupt	0x0000_0040
1	8	settable	PVD	PVD_IRQHandler	PVD interrupt (combined EXTI lines 16)	0x0000_0044
2	9	settable	RTC	RTC_IRQHandler	RTC interrupts (combined EXTI lines 17, 19 and 20)	0x0000_0048
3	10	settable	FLASH	FLASH_IRQHandler	Flash global interrupt	0x0000_004c
4	11	settable	RCC	RCC_IRQHandler	RCC global interrupt	0x0000_0050
5	12	settable	EXTI0_1	EXTI0_1_IRQHandler	EXTI Line 0 and 1 interrupt	0x0000_0054
6	13	settable	EXTI2_3	EXTI2_3_IRQHandler	EXTI Line 2 and 3 interrupts	0x0000_0058
7	14	settable	EXTI4_15	EXTI4_15_IRQHandler	EXTI Line 4 to 15 interrupts	0x0000_005c
8	15	settable	UART3	UART3_IRQHandler	UART3 global interrupt	0x0000_0060
9	16	settable	DMA_CH1	DMA1_Channel1_IRQHandler	DMA channel 1 global interrupt	0x0000_0064
10	17	settable	DMA_CH2_3	DMA1_Channel2_3_IRQHandler	DMA channel 2 and 3 interrupts	0x0000_0068
11	18	settable	DMA_CH4_7	DMA1_Channel4_7_IRQHandler	DMA channel 4 to 7 interrupts	0x0000_006c
12	19	settable	ADC_COMP1_COMP2	ADC1_COMP_IRQHandler	ADC and comparator 1and 2 interrupt through EXTI 21 and 22	0x0000_0070

13	20	settable	LPTIM1	LPTIM1_IRQHandler	LPTIMER1 interrupt through EXTI 29	0x0000_0074
14	21	settable	UART4	UART4_IRQHandler	UART4 global interrupt	0x0000_0078
15	22	settable	TIM2	TIM2_IRQHandler	TIM2 global interrupt	0x0000_007c
16	23	settable	TIM3	TIM3_IRQHandler	TIM3 global interrupt	0x0000_0080
17	24	settable	DAC	DAC_IRQHandler	DAC global interrupt	0x0000_0084
18	25	settable	LPTIM2	LPTIM2_IRQHandler	LPTIMER2 interrupt through EXTI 27	0x0000_0088
19	26	settable	LPUART1	LPUART1_IRQHandler	LPUART1 interrupt through EXTI 28	0x0000_008c
20	27	settable	LPTIM3	LPTIM3_IRQHandler	LPTIMER3 interrupt through EXTI 30	0x0000_0090
21	28	settable	DVSQ	DVSQ_IRQHandler	DVSQ global interrupt	0x0000_0094
22	29	settable	TIM1	TIM1_IRQHandler	TIMER1 global interrupt	0x0000_0098
23	30	settable	I2C1	I2C1_IRQHandler	I2C1 global interrupt through EXTI 23	0x0000_009c
24	31	settable	I2C2	I2C2_IRQHandler	I2C2 global interrupt through EXTI 24	0x0000_00a0
25	32	settable	SPI1	SPI1_IRQHandler	SPI1 global interrupt	0x0000_00a4
26	33	settable	SPI2	SPI2_IRQHandler	SPI2 global interrupt	0x0000_00a8
27	34	settable	USART1	USART1_IRQHandler	USART1 global interrupt through EXTI 25	0x0000_00ac
28	35	settable	USART2	USART2_IRQHandler	USART2 global interrupt through EXTI 26	0x0000_00b0
29	36	settable	AES_RNG	AES_RNG_IRQHandler	AES global interrupt + RNG global interrupt	0x0000_00b4
30	37	settable	LCD	LCD_IRQHandler	LCD global interrupt	0x0000_00b8
31	38	settable	USB	USB_IRQHandler	USB wakeup event interrupt through EXTI 18	0x0000_00bc

12 扩展中断和事件控制器 (EXTI)

12.1 简介

HK32F08X 内置 32 个 EXTI 口，其中 0~15 连接 IO，其余的 EXTI 口连接以下事件：

- EXTI 16 连接 PVD 输出
- EXTI 17 连接 RTC 的 Alarm 事件
- EXTI 18 连接 USB 的 wakeup 事件
- EXTI 19 连接 RTC 的 Tamper 和 TimeStamp 事件
- EXTI 20 连接 RTC 的 Wakeup 事件
- EXTI 21 连接 ADC comparator1 的 Wakeup 事件
- EXTI 22 连接 ADC comparator2 的 Wakeup 事件
- EXTI 23 连接 I2C1 的 wakeup 事件
- EXTI 24 连接 I2C2 的 wakeup 事件
- EXTI 25 连接 USART1 的 wakeup 事件
- EXTI 26 连接 USART2 的 wakeup 事件
- EXTI 27 连接 LPTIM2 的 wakeup 事件
- EXTI 28 接接 LPUART1 的 wakeup 事件
- EXTI 29 连接 LPTIM1 的 wakeup 事件
- EXTI 30 连接 LPTIM3 的 wakeup 事件
- EXTI 31 连接 ADC 的 AWD 事件

其中 18,23~31 作为内部事件没有 RTSR、FTSR、SWIER 和 PR 寄存器，只能在 STOPMODE 下采事件的上升沿产生 ERQ 和 IRQ 唤醒系统。

12.2 EXTI 寄存器

12.2.1 中断屏蔽寄存器(EXTI_IMR)

偏移地址： 0x00

复位值: 0x0FF4 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IM31	IM30	IM29	IM28	IM27	IM26	IM25	IM24	IM23	IM22	IM21	IM20	IM19	IM18	IM17	IM16
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IM15	IM14	IM13	IM12	IM11	IM10	IM9	IM8	IM7	IM6	IM5	IM4	IM3	IM2	IM1	IM0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位 31:0	**IMx:** 外部/内部线 x 的中断屏蔽位 0: 屏蔽来自线 x 上的中断请求 1: 开放来自线 x 上的中断请求

12.2.2 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
EM31	EM30	EM29	EM28	EM27	EM26	EM25	EM24	EM23	EM22	EM21	EM20	EM19	EM18	EM17	EM16
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EM15	EM14	EM13	EM12	EM11	EM10	EM9	EM8	EM7	EM6	EM5	EM4	EM3	EM2	EM1	EM0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位 31:0	**EMx:** 外部/内部线 x 的事件屏蔽位 0: 屏蔽来自线 x 上的事件请求 1: 开放来自线 x 上的事件请求

12.2.3 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
RT31	RT30	RT29	RT28	RT27	RT26	RT25	RT24	RT23	RT22	RT21	RT20	RT19	RT18	RT17	RT16
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RT15	RT14	RT13	RT12	RT11	RT10	RT9	RT8	RT7	RT6	RT5	RT4	RT3	RT2	RT1	RT0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位 31:0	**RTx:** 线 x 上的上升沿触发事件配置位(x=31 到 0) 0: 屏蔽来自线 x 上的事件请求 1: 开放来自线 x 上的事件请求

12.2.4 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
FT31	FT30	FT29	FT28	FT27	FT26	FT25	FT24	FT23	FT22	FT21	FT20	FT19	FT18	FT17	FT16
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FT15	FT14	FT13	FT12	FT11	FT10	FT9	FT8	FT7	FT6	FT5	FT4	FT3	FT2	FT1	FT0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位 31:0	**FTx:** 线 x 上的下降沿触发事件配置位(x=31 到 0) 0: 屏蔽来自线 x 上的下降沿触发（中断和事件） 1: 开放来自线 x 上的事件请求（中断和事件）

12.2.5 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
SWI31	SWI30	SWI29	SWI28	SWI27	SWI26	SWI25	SWI24	SWI23	SWI22	SWI21	SWI20	SWI19	SWI18	SWI17	SWI16
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWI15	SWI14	SWI13	SWI12	SWI11	SWI10	SWI9	SWI8	SWI7	SWI6	SWI5	SWI4	SWI3	SWI2	SWI1	SWI0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位 31:0	**SWIx:** 线 x 上的软件中断 (x=31 to 0) 当该位为’0’时，写’1’将设置 EXTI_PR 中相应的挂起位。如果在 EXTI_IMR 和 EXTI_EMR 中允许产生该中断，则此时将产生一个中断。通过清除 EXTI_PR 的对应位 (写入’1’)，可以清除该位为’0’。

12.2.6 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
PIF31	PIF30	PIF29	PIF28	PIF27	PIF26	PIF25	PIF24	PIF23	PIF22	PIF21	PIF20	PIF19	PIF18	PIF17	PIF16
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PIF15	PIF14	PIF13	PIF12	PIF11	PIF10	PIF9	PIF8	PIF7	PIF6	PIF5	PIF4	PIF3	PIF2	PIF1	PIF0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位 31:0	**PIFx:** 线 x 挂起位 (x=31 到 0) 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。

13 模数转换器（ADC）

13.1 简介

12 位 ADC 是逐次趋近型模数转换器。它具有多达 20 个复用通道，可测量来自 16 个外和 4 个内部源的信号。各种不同通道的 A/D 转换可在单次、连续、扫描或不连续采样模式下进行。ADC 的结果存储在一个左对齐或右对齐的 16 位数据寄存器中。

ADC 具有模拟看门狗特性，允许应用检测输入电压是否超过了用户自定义的阈值上限或下限。

采用了高效的低功耗模式，在低频下可实现极低的功耗。

内置硬件过采样器，可提高模拟性能，同时还能减轻 CPU 进行相关计算的负担。

13.2 ADC 主要特性

- 高性能
  - 可配置 12 位、 10 位、 8 位或 6 位分辨率
  - ADC 转换时间： 12 位分辨率对应的转换时间为 0.87 μs (1.14 MHz)， 10 位分辨率对应的转换时间为 0.81 μs，若降低分辨率，可进一步缩短转换时间
  - 自校准
  - 可编程采样时间
  - 内置数据一致性，可确保数据对齐
  - 支持 DMA
- 低功耗
  - 应用可降低 PCLK 频率从而以低功耗运行，同时仍可保持最优的 ADC 性能。例如，无论 PCLK 的频率如何，都会保持 1.0 μs 的转换时间
  - 等待模式：防止 ADC 在低频 PCLK 应用中溢出
  - 自动关闭模式： ADC 会自动断电，但正在进行转换时除外。这可大幅降低 ADC 的功耗
- 模拟输入通道
  - 16 路外部模拟输入
  - 1 条用于内部温度传感器 (VSENSE) 的通道
  - 1 条用于内部参考电压 (VREFINT) 的通道
  - 1 条用于监视外部 VLCD 电源针脚的通道
  - 1 条用于采集比较器 OPA 输出的通道
- 可通过以下方式启动转换过程：
  - 通过软件
  - 通过极性可配置的硬件触发器（来自 TIM1、TIM2、 TIM3 的内部定时器事件，或 GPIO 输入事件）
- 转换模式
  - 可转换单条通道，也可扫描一系列通道。
  - 单次模式会在每次触发时对选定的输入执行一次转换
  - 连续模式可连续转换选定的输入
  - 不连续转换模式

- 在采样结束、转换结束、序列转换结束以及发生模拟看门狗或溢出事件时产生中断
- 模拟看门狗
- 过采样器
  - 16 位数据寄存器
  - 过采样率可在 2x 到 256x 之间进行调整
  - 可编程数据最多可移位 8 位
- ADC 电源要求： 1.65 V 到 3.6 V
- ADC 输入范围： VSSA ≤ VIN ≤ VDDA

13.3 ADC 引脚和内部信号

表 13-1 ADC 引脚

名称	信号类型	备注
VDDA	模拟电源输入	ADC 的模拟电源和正参考电压，VDDA≥VDD
VSSA	模拟电源地输入	模拟电源接地引脚。电压必须等于 VSS
ADC_IN[15:0]	模拟输入信号	16 个模拟输入通道

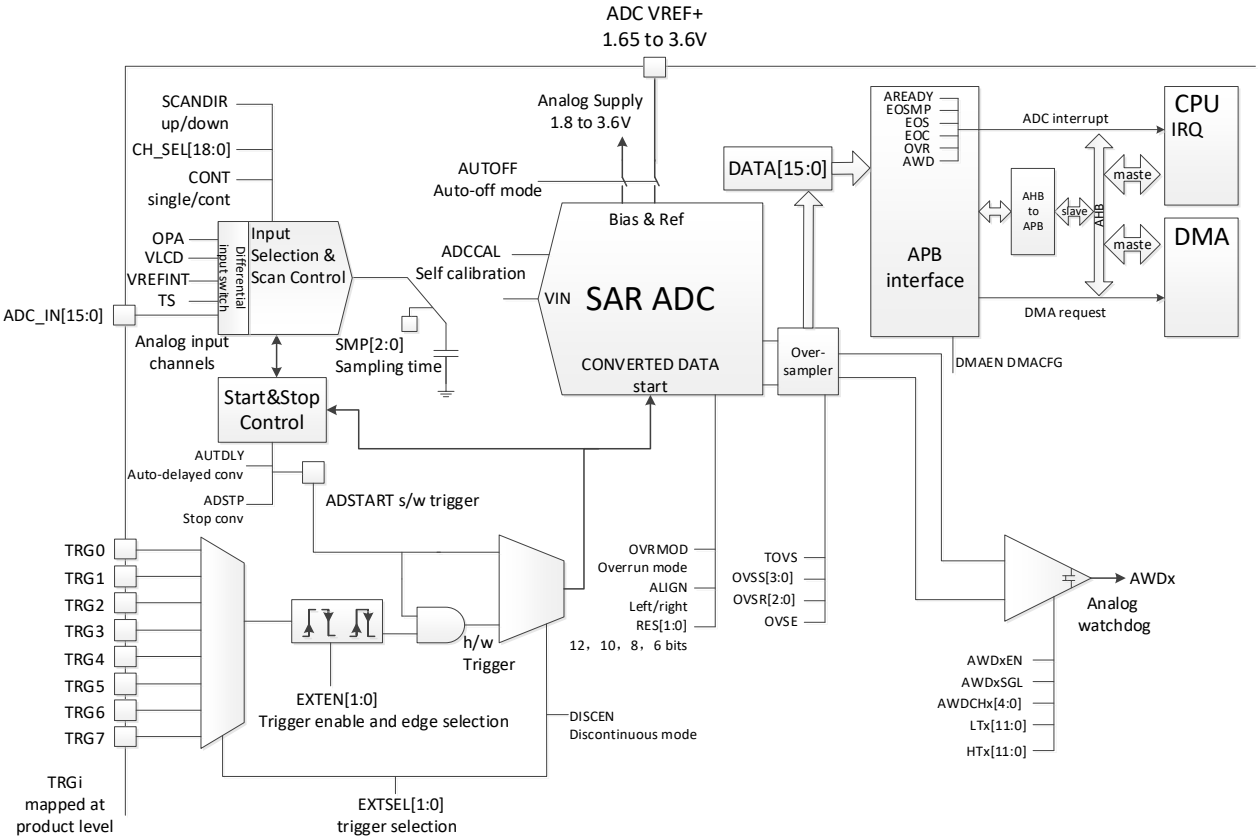
表 13-2 ADC 内部信号

内部信号名称	信号类型	说明
TRGx	输入	ADC 转换触发信号
VSENSE	输入	内部温度传感器输出电压
VREFINT	输入	内部参考电压输出电压
LCD_VLCD	输入	1/3 VLCD（LCD 配置为 1/3 偏置时） 1/4 VLCD（LCD 配置为 1/4 偏置或 1/2 偏置时）
OPA_OUT	输入	放大器的输出

13.4 ADC 功能描述

ADC 功能框图：

图 13-1 ADC 功能框图



13.4.1 校准 (ADCAL)

ADC 具有校准功能。校准过程中， ADC 会计算校准系数， ADC 下一次掉电之前，会在内部对 ADC 应用此校准系数。校准过程中，应用不得使用 ADC，必须等待校准完成。

校准应在启动 A/D 转换之前进行。校准可消除偏移误差，由于制造工艺的不同，各芯片的偏移误差也有所不同。

校准通过软件将 ADCAL 位置 1 发起。仅当 ADC 禁止 (ADEN=0) 后，才能发起校准。

ADCAL 位在所有校准序列过程中保持为 1。校准完成后，此位会立即由硬件清零。随后，可从 ADC_DR 寄存器中读取校准系数（位 5 到位 0）。

若禁止了 ADC (ADEN=0)，则会保留内部模拟校准。如果 ADC 工作条件发生变化（VDDA 变化是造成 ADC 偏移变化的主要原因，并会导致温度发生小幅变化），建议重新运行校准周期。

在下列情况下，校准系数会丢失：

- 产品处于待机模式（ADC 掉电）
- ADC 外设复位。

在低功耗运行、低功耗睡眠和停止这三种低功耗模式下，会保留校准系数。仍可通过软件保存并恢复校准系数，以节省 ADC 重启时间（前提是 ADC 掉电期间的温度和电压保持稳定）。

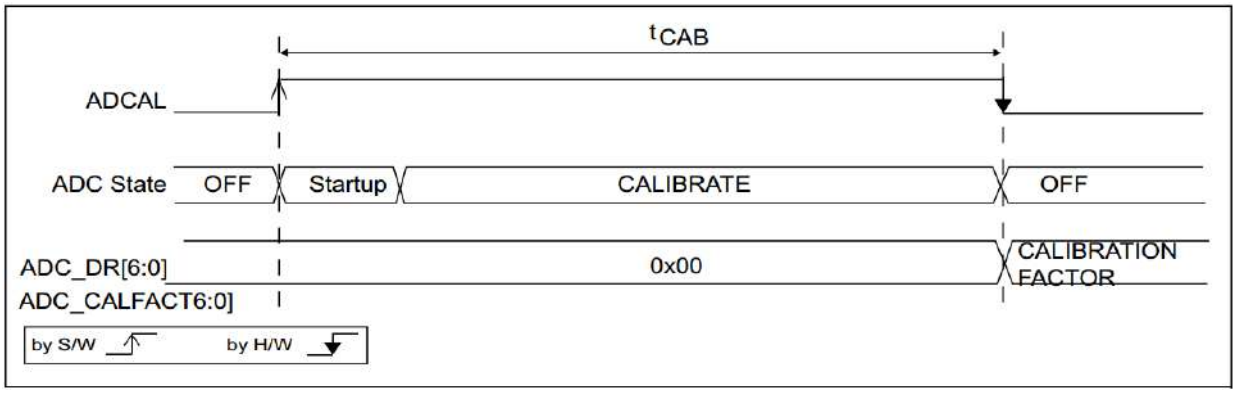
如果 ADC 已使能但未进行转换（ADEN=1 且 ADSTART=0），可写入校准系数。随后，下次转换启动时，校准系数会自动添加到模拟 ADC 中。这一载入过程是透明的，不会对转换的启动造成延迟。

校准软件程序

1. 确保 ADEN=0
2. 将 ADCAL 置 1

3. 等待，直至 ADCAL=0（或直至 EOCAL=1）。如果已通过将 ADC_IER 寄存器中的 EOCALIE 位置 1 来使能中断，可通过中断进行处理
4. 校准系数可从 ADC_DR 或 ADC_CALFACT 寄存器的位 6:0 读取。

图 13-2 ADC 校准



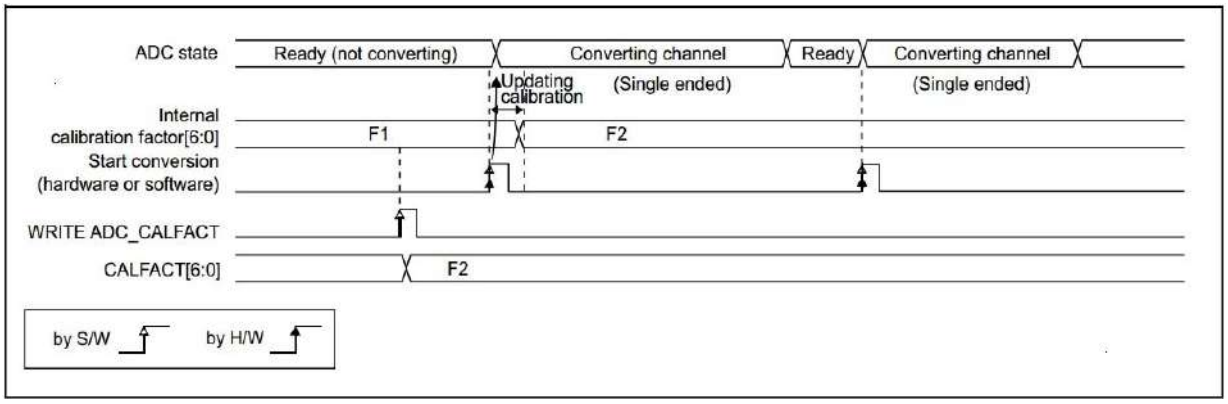
如果 ADC 调压器之前未置 1，则将 ADCAL 置 1 后。在这种情况下，由于要考虑 ADC 调压器的稳定时间，因此 ADC 校准时间会比较长。

校准结束时， ADC 调压器仍保持使能状态。

校准系数强制软件程序

1. 确保 ADEN= 1 且 ADSTART =0（ADC 启动时没有进行任何转换）
2. 将保存的校准系数写入 ADC_CALFACT
3. 启动新的转换后，将立即使用校准系数。

图 13-3 校准系数强制



13.4.2 ADC 开关控制（ADEN、ADDIS、ADRDY）

MCU 上电时，会禁止 ADC 并进入掉电模式 (ADEN=0)。

如图 14.4.2_1 所示， ADC 在开始精确转换之前需要一段稳定时间 tSTAB。

以下两个控制位可用于使能或禁止 ADC：

- 将 ADEN 置 1 可使能 ADC。 ADC 准备就绪后， ADRDY 标志会立即置 1。
- 将 ADDIS 置 1 可禁止 ADC 并使 ADC 进入掉电模式。随后， ADC 被完全禁止后，ADEN 位和 ADDIS 位会自动由硬件清零。

随后可通过将 ADSTART 置 1（请参见外部触发转换和触发极性（EXTSEL， EXTEN） ）开始进行转换，如果触发器已使能，也可在发生外部触发事件时开始进行转换。

请按照以下流程使能 ADC：

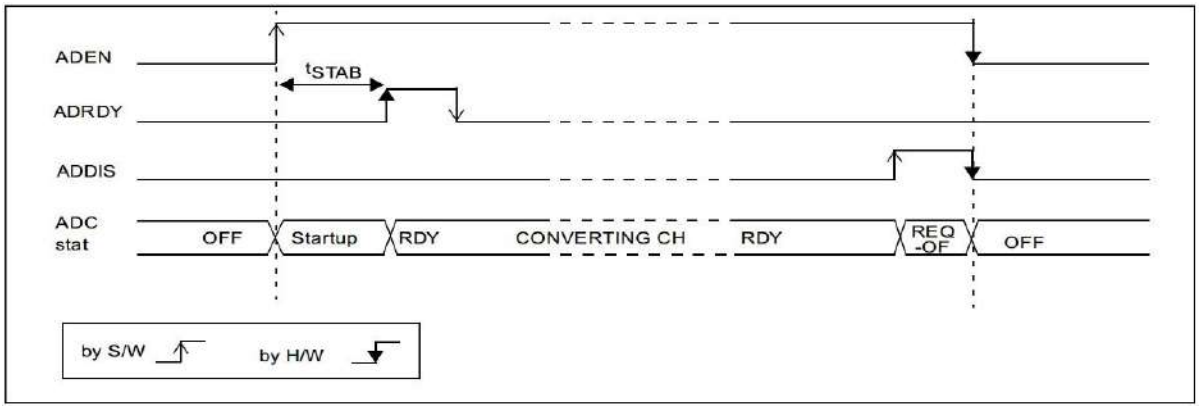
1. 将 ADC_ISR 寄存器中的 ADRDY 位编程为 1，将此位清零。
2. 将 ADC_CR 寄存器中的 ADEN 位置 1。
3. 等待，直至 ADC_ISR 寄存器中的 ADRDY=1（ADRDY 会在 ADC 启动时间后置 1）。

如果已通过将 ADC_IER 寄存器中的 ADRDYIE 位置 1 来使能中断，可通过中断进行处理。

请按照以下流程禁止 ADC：

1. 检查 ADC_CR 寄存器中的 ADSTART 是否为 0，以确保当前未执行任何转换。如有需要，可向 ADC_CR 寄存器中的 ADSTP 位写入 1 并等待此位读取值为 0，以此停止正在进行的转换。
2. 将 ADC_CR 寄存器中的 ADDIS 位置 1。
3. 如果应用要求，可等待 ADC_CR 寄存器中的 ADEN=0，这表明 ADC 已完全禁止（ADEN=0 后， ADDIS 会自动复位）。
4. 将 ADC_ISR 寄存器中的 ADRDY 位编程为 1，将此位清零（可选）。

图 13-4 使能/禁止 ADC

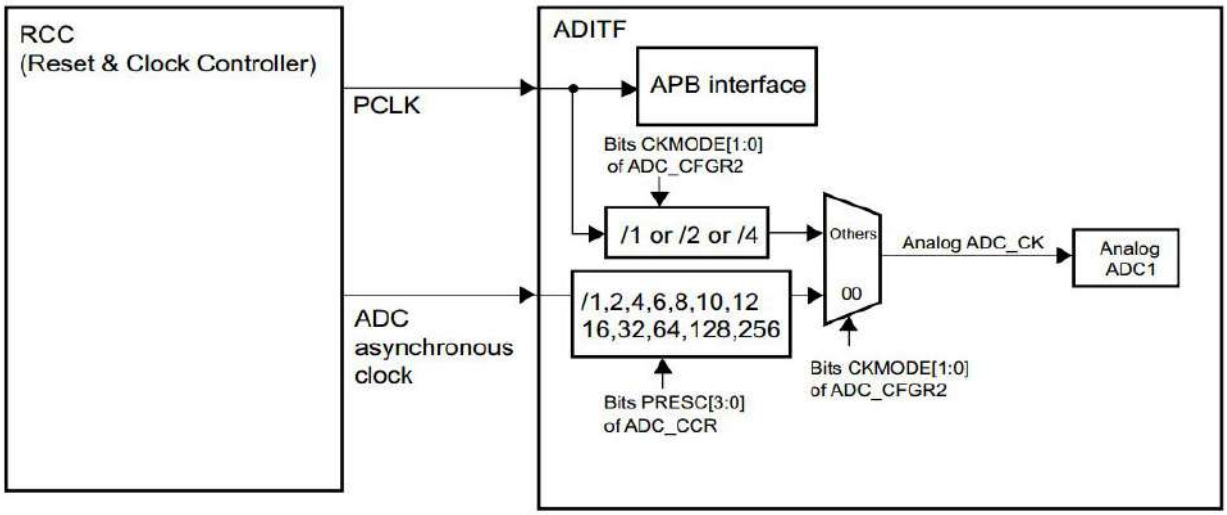


注：在自动关闭模式下 (AUTOFF=1)，上电/掉电阶段是由硬件自动执行的，不会将 ADRDY 标置 1。

13.4.3 ADC 时钟（CKMODE、 PRESC[3:0]、 LFMEN）

ADC 采用双时钟域架构，因此， ADC 可由独立于 APB 时钟 (PCLK) 的时钟提供时钟（ADC 异步时钟）。

图 13-5 ADC 时钟图



注：了解 PCLK 和 ADC 异步时钟的使能方式参见复位和时钟控制（RCC）章节

模拟 ADC 的输入时钟可在两个不同的时钟源之间进行选择（请参见图 14.4.3.1，了解 PCLK 和 ADC

异步时钟的使能方式）：

a) ADC 时钟可以是名为“ADC 异步时钟”的特定时钟源，该时钟源独立于 APB 时钟，并与 APB 时钟异步。

更多有关生成该时钟源的信息，请参见 RCC 部分。

要选择此时钟图，必须将 ADC_CFGR2 寄存器的 CKMODE[1:0] 位复位。

b) ADC 时钟可由 ADC 总线接口的 APB 时钟除以一个可编程因子（2 或 4，具体根据 CKMODE[1:0] 位而定）来提供。

要选择此时钟图， ADC_CFGR2 寄存器的 CKMODE[1:0] 位不得为“00”。

在选项 a) 中，对 ADC_CCR 寄存器中的 PRESC[3:0] 位进行编程时，生成的 ADC 时钟最后会除以预分频系数（1、 2、 4、 6、 8、 12、 16、 32、 64、 128、 256）。

选项 a) 的优点在于无论选择哪种 APB 时钟图，都可以达到最大 ADC 时钟频率。

选项 b) 的优势在于绕过了时钟域重新同步。如果 ADC 由定时器触发，并且应用要求 ADC 精确触发（不存在任何不确定性），可使用此选项（否则，重新同步两个时钟域会为触发时刻带来不确定性。

表 13-3 触发与转换开始之间的延迟

ADC 时钟源	CKMODE[1:0]	触发事件与转换开始之间的延迟
HSI 16MHZ 时钟	00	延迟是不确定的（存在抖动）
Pclk 2 分频	01	延迟是确定的（无抖动），等于 4.25 个 ADC 时钟周期
Pclk 4 分频	10	延迟是确定的（无抖动），等于 4.125 个 ADC 时钟周期
Pclk 1 分频	11	延迟是确定的（无抖动），等于 4.5 个 ADC 时钟周期

选择 CKMODE[1:0]=11（PCLK 1 分频）时，用户必须确保 PCLK 的占空比为 50%。为此，在 RCC 中，用户必须选择占空比为 50% 的系统时钟，并且必须以旁路模式在 RCC 内配置 APB 预分频系数（请参见 RCC 部分）。对于内部源时钟，这只意味着 AHB 和 APB 预分频器不会对时钟进行分频。

低频

如果选择的模拟 ADC 时钟频率低于 3.5 MHz，必须先将 ADC_CCR 寄存器中的 LFMEN 位置 1，从而使能低频模式。

13.4.4 配置 ADC

如果 ADC 已禁止，必须通过软件对 ADC_CR 寄存器中的 ADCAL 位和 ADEN 位执行写操作（ADEN 必须为 0）。

只有在 ADC 已使能、并且没有待处理的禁止 ADC 的请求时（ADEN=1，且 ADDIS=0），软件才必须只对 ADC_CR 寄存器中的 ADSTART 位和 ADDIS 位执行写操作。

对于 ADC_IER、ADC_CFGRi、ADC_SMPR、ADC_TR、ADC_CHSELR 和 ADC_CCR 寄存器中的其他控制位，只有在 ADC 已使能(ADEN=1)，且没有正在进行的转换(ADSTART=0)，软件才可以执行写操作。

如果 ADC 已使能（很可能正在进行转换）、并且没有待处理的停止 ADC 的请求时

（ADSTART=1 且 ADDIS=0），软件必须且只能对 ADC_CR 寄存器中的 ADSTP 位执行写操作才能停止 ADC。

注：未采取硬件保护机制来防止软件执行上述规则禁止的写操作。如果发生此类禁止的写访问，ADC 可能会进入未定义状态。要在这种情况下恢复正确的操作，必须禁止 ADC（将 ADEN 以及 ADC_CR 寄存器中的所有位都清零）。

13.4.5 通道选择

复用通道多达 20 条：

- 16 路来自 GPIO 引脚(ADC_IN0...ADC_IN15)的模拟输入
- 4 路内部模拟输入（温度传感器、内部参考电压、LCD_VLCD1 通道和运放的输出）

可转换单条通道，也可以自动扫描一系列通道。

待转换通道的顺序必须在 ADC_CHSELR 通道选择寄存器中进行编程：每条模拟输入通道都有专用的选择位(CHSEL0...CHSEL19)。

要配置通道的扫描顺序，可对 ADC_CFGR1 寄存器中的 SCANDIR 位进行编程：

- SCANDIR=0：正向扫描通道 0 到通道 19
- SCANDIR=1：反向扫描通道 19 到通道 0

温度传感器、VREFINT、LCD_VLCD1 和运放输出等内部通道

温度传感器连接至通道 ADC_IN18。内部参考电压 VREFINT 连接至通道 ADC_IN17。

LCD_VLCD1 通道连接至通道 ADC_IN16。运放输出连接至通道 ADC_IN19。

13.4.6 可编程采样时间（SMP）

开始转换之前，ADC 需要在待测量电压源与 ADC 内置采样电容之间建立直接连接。该采样时间必须足以使输入电压源为采样电容充电并将电容保持在输入电压水平。

使用可编程采样时间后，可根据输入电压源的输入电阻调整转换速度。

ADC 会在数个 ADC 时钟周期内对输入电压进行采样，时钟周期数可使用 ADC_SMPR 寄存器中的 SMP[2:0]位进行修改。

此可编程采样时间是所有通道共用的。如果应用要求，可通过软件在两次转换之间更改和调整此采样时间。

总转换时间的计算公式如下：

tCONV=采样时间+12.5xADC 时钟周期

示例：

如果 ADC_CLK=16MHz，采样时间为 1.5 个 ADC 时钟周期：

tCONV=1.5+12.5=14 个 ADC 时钟周期=0.875μs

ADC 通过将 EOSMP 标志置 1 来指示采样阶段结束。

13.4.7 单次转换模式（CONT=0）

在单次转换模式下，ADC 会执行单次转换序列，对所有通道进行一次转换。当 ADC_CFGR1 寄存器中的 CONT=0 时，会选择此模式。可通过以下方式开始转换：

- 将 ADC_CR 寄存器中的 ADSTART 位置 1
- 硬件触发事件在序列中，每次转换完成后：
- 转换后的数据会存储在 16 位 ADC_DR 寄存器中
- EOC（转换结束）标志置 1
- EOCIE 位置 1 时将产生中断转换序列完成后：
- EOSEQ（序列结束）标志置 1
- EOSEQIE 位置 1 时将产生中断
- 随后，ADC 会停止工作，直至发生新的外部触发事件或 ADSTART 位再次置 1。

注：要转换单个通道，可将序列长度编程为1。

13.4.8 连续转换模式（CONT=1）

在连续转换模式下，如果发生软件或硬件触发事件，ADC 会执行转换序列，对所有通道进行一次转换，随后会自动重启并持续执行相同的转换序列。当 ADC_CFGR1 寄存器中的 CONT=1 时，会选择此模式。可通过以下方式开始转换：

- 将 ADC_CR 寄存器中的 ADSTART 位置 1
- 硬件触发事件
- 在序列中，每次转换完成后：
- 转换后的数据会存储在 16 位 ADC_DR 寄存器中
- EOC（转换结束）标志置 1
- EOCIE 位置 1 时将产生中断
- 转换序列完成后：
- EOSEQ（序列结束）标志置 1
- EOSEQIE 位置 1 时将产生中断

随后，会立即重启新序列，ADC 会继续重复执行转换序列。

注：要转换单个通道，可将序列长度编程为 1。

不能同时使能不连续模式和连续模式：禁止同时将 DISCEN 和 CONT 位置 1。

13.4.9 开始转换（ADSTART）

软件通过将 ADSTART 置 1 的方式开始进行 ADC 转换。

ADSTART 置 1 后：

- 在 EXTEN=00 时会立即开始转换（软件触发）
- 在 EXTEN≠ 00 时，会在所选硬件触发器的下一有效边沿开始转换。

ADSTART 位还用于指示当前是否正在进行 ADC 操作。可以在 ADSTART=0 时重新配置 ADC，从而指示 ADC 处于空闲状态。

ADSTART 位由硬件清零：

- 在使用软件触发的单次模式下（CONT=0，EXTEN=00）
  - 只要转换序列结束(EOSEQ=1)就清零
- 在使用软件触发的不连续模式下（CONT=0、DISCEN=1、EXTEN=00）
  - 转换结束时(EOC=1)清零
- 在所有情况下（CONT=x、EXTEN=XX）
  - 执行由软件调用的 ADSTP 程序之后（参见停止转换（ADSTP 章节））清零

注：在连续模式下(CONT=1)，由于序列会自动重新启动，因此，当 EOSEQ 标志置 1 时，ADSTART 位不会清零。

如果在单次模式下选择了硬件触发（CONT=0，且 EXTEN≠ 00），则 EOSEQ 置 1 时，ADSTART 不会由硬件清零。这样，便无需通过软件将 ADSTART 再次置 1，并可确保不会错过下一触发事件。

13.4.10 时序

从转换开始到转换结束所经过的时间是配置的采样时间与逐次趋近时间（具体视数据分辨率而定）的总和：

tADC=tSMPL+tSAR=[1.5|min+12.5|12bit]xtADC_CLK

tADC=tSMPL+tSAR=93.8ns|min+781.3ns|12bit=0.875μs|min（对于 fADC_CLK=16MHz）

图 13-6 模数转换时间

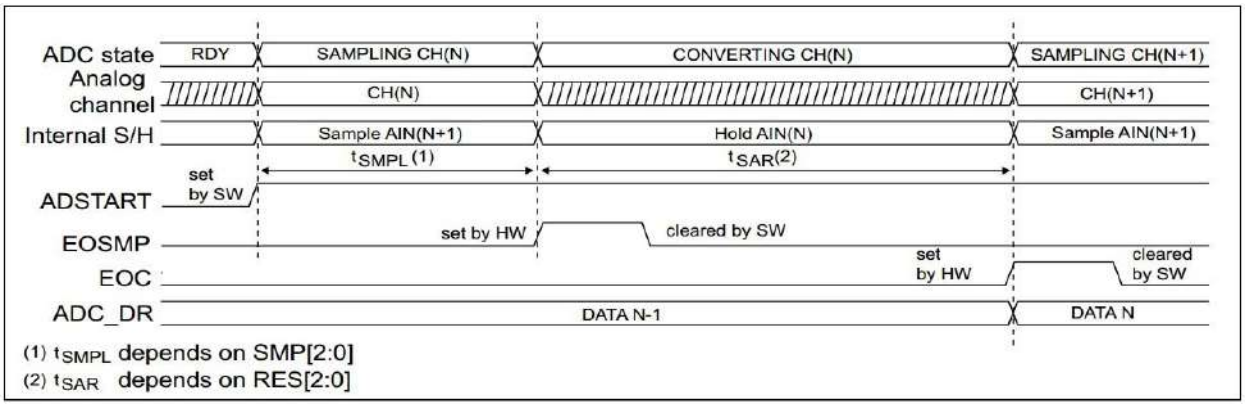
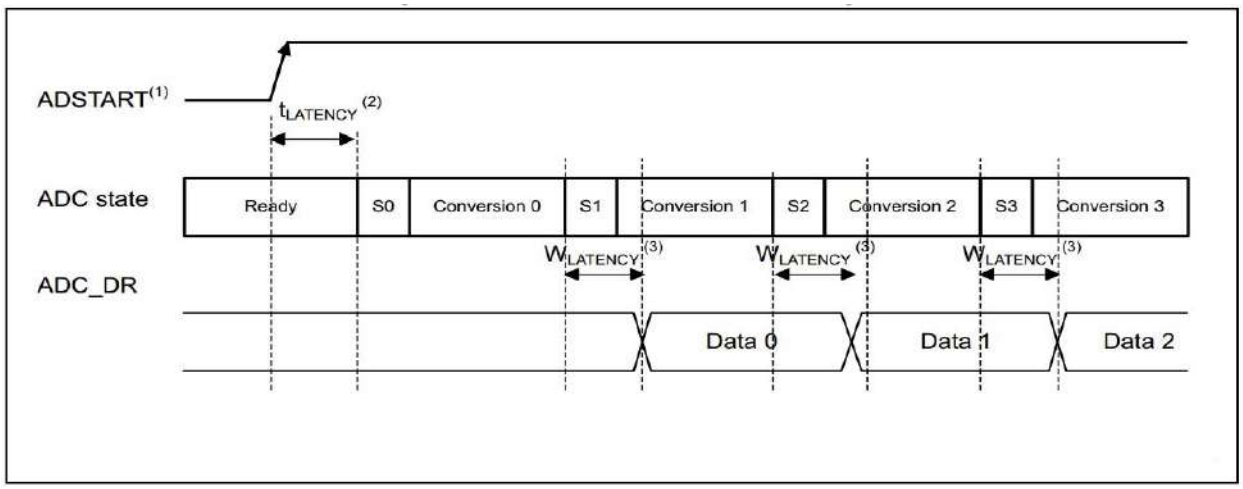


图 13-7 模数转换时序



1. EXTEN = 00 或 EXTEN ≠ 00
2. 触发延迟（更多详细信息，请参见数据手册）
3. ADC_DR 寄存器写入延迟（更多详细信息，请参见数据手册）

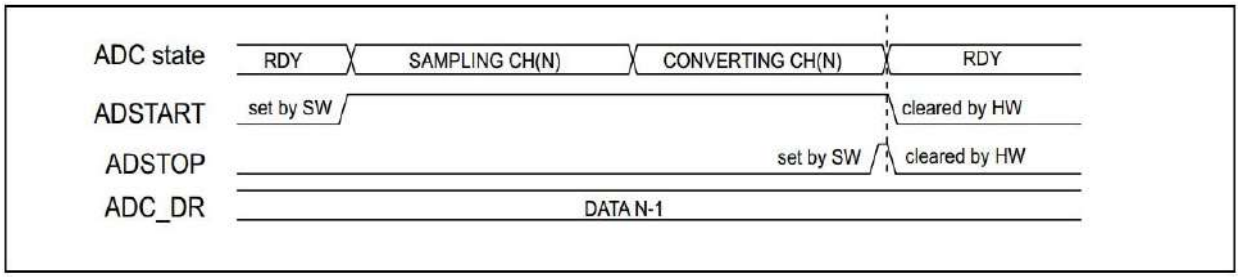
13.4.11 停止正在进行的转换（ADSTP）

可通过软件将 ADC_CR 寄存器中的 ADSTP 置 1，停止任何正在进行的转换。此操作会复位 ADC 操作， ADC 将处空闲状态，准备好进行新操作。

如果 ADSTP 位由软件置 1，则会中止任何正在进行的转换，并会丢弃转换结果（ADC_DR 寄存器不会更新为当前转换结果）。

扫描序列也会中止并会复位（这意味着重启 ADC 将重新开始新的序列）。此程序完成后， ADSTP 位和 ADSTART 位均会由硬件清零，软件必须等待 ADSTART=0，然后才能开始进行新的转换。

图 13-8 停止正在进行的转换



13.5 外部触发转换和触发极性（EXTSEL，EXTEN）

可通过软件或外部事件（定时器捕获事件）触发转换或转换序列。如果 EXTEN[1:0] 控制位不等于“0b00”，那么外部事件能够触发具有所选极性的转换。软件将 ADSTART 位置 1 后，触发选择将立即生效。

在转换进行时发生的硬件触发会被忽略。

如果位 ADSTART=0，则会忽略发生的任何硬件触发。

表 14.5_1 提供 EXTEN[1:0] 值与触发极性之间的对应关系。

表 13-4 配置触发极性

源	EXTEN[1:0]
禁止触发检测	00
在上升沿检测	01
在下降沿检测	10
在上升沿和下降沿均检测	11

注： 仅当 ADC 未进行转换 (ADSTART= 0) 时，才可以更改外部触发的极性。

EXTSEL[2:0] 控制位用于选择 8 个可能的事件中哪一事件可触发转换。

表 14.5_1 给出了可用于常规转换的外部触发。

可将 ADC_CR 寄存器中的 ADSTART 位置 1，从而生成软件源触发事件。

表 14.5_2 给出了可用于常规转换的外部触发。

可将 ADC_CR 寄存器中的 ADSTART 位置 1，从而生成软件源触发事件。

表 13-5 外部触发器

名称	源	EXTSEL[2:0]
TRG0	TIM1_TRGO	000
TRG1	TIM1_CC4	001
TRG2	TIM2_TRGO	010
TRG3	TIM2_CH4	011
TRG4	TIM1_CC2	100
TRG5	TIM2_CH3	101
TRG6	TIM3_TRGO	110
TRG7	EXTI Line 11	111

注： 仅当 ADC 未进行转换 (ADSTART= 0) 时，才可以更改触发选择。

13.5.1 不连续模式 (DISCEN)

可将 ADC_CFGR1 寄存器中的 DISCEN 位置 1 来使能此模式。

在该模式下(DISCEN=1)，需要通过硬件或软件触发事件开始序列中定义的各个转换。相反，如果 DISCEN=0，单个硬件或软件触发事件会连续启动序列中定义的所有转换。

示例：

- DISCEN=1，待转换通道=0、3、7、10
  - 第一次触发：转换通道 0 并生成 EOC 事件
  - 第二次触发：转换通道 3 并生成 EOC 事件
  - 第三次触发：转换通道 7 并生成 EOC 事件
  - 第四次触发：转换通道 10 并同时生成 EOC 和 EOSEQ 事件
  - 第五次触发：转换通道 0 并生成 EOC 事件
  - 第六次触发：转换通道 3 并生成 EOC 事件

- DISCEN=0，待转换通道=0、3、7、10
  - 第一次触发：转换整个序列：通道 0、然后是通道 3、7 和 10。每次转换都会生成 EOC 事件，最后一次转换还会生成 EOSEQ 事件。
  - 任何后续触发事件都将重启整个序列。

注：不能同时使能不连续模式和连续模式：禁止同时将 DISCEN 和 CONT 位置 1。

13.5.2 可编程分辨率 (RES) - 快速转换模式

可通过降低 ADC 分辨率缩短转换时间(tSAR)。

通过对 ADC_CFGR1 寄存器中的 RES[1:0]位进行编程，可将分辨率配置为 12 位、10 位、8 位或 6 位。对于不要求使用高数据精度的应用，分辨率越低，转换时间越短。

注：RES[1:0]位必须在 ADEN 位复位后才能进行更改。

转换结果的宽度始终为 12 位，任何未使用的 LSB 位都会读为零。

降低分辨率可缩短逐次趋近步骤所需的转换时间，如表 59 所示。

表 13-6 配置触发极性

RES[1:0]位	tSAR（ADC 时钟周期）	tSAR(ns)，fADC=16MHz	tSMPL(min)（ADC 时钟周期）	tCONV（ADC 时钟周期）（使用最短 tSMPL）	tCONV(μs)，fADC=16MHz
12	12.5	781ns	1.5	14	875ns
10	11.5	719ns	1.5	13	812ns
8	9.5	594ns	1.5	11	688ns
6	7.5	469ns	1.5	9	562ns

13.5.3 转换结束、采样阶段结束（EOC、EOSMP 标志）

ADC 指示每个转换结束(EOC)事件。

新的转换数据结果出现在 ADC_DR 寄存器中后，ADC 会立即将 ADC_ISR 寄存器中的 EOC 标志置 1。如果 ADC_IER 寄存器中的 EOCIE 位置 1，可产生中断。EOC 标志可通过由软件向其写入 1 或读取 ADC_DR 寄存器的方式来清零。

ADC 还通过将 ADC_ISR 寄存器中的 EOSMP 标志置 1 来指示采样阶段结束。EOSMP 标志可通过由软件向其写入 1 来清零。如果 ADC_IER 寄存器中的 EOSMPIE 位置 1，可产生中断。

此中断用于使处理与转换进行同步。通常情况下，可在转换阶段的隐藏时间中访问模拟复用器，这样在下次采样开始时便可放置好复用器。

注：由于采样结束与转换结束之间只留有非常短的时间，因此建议使用轮询或 WFE 指令，而不建议使用中断和 WFI 指令。

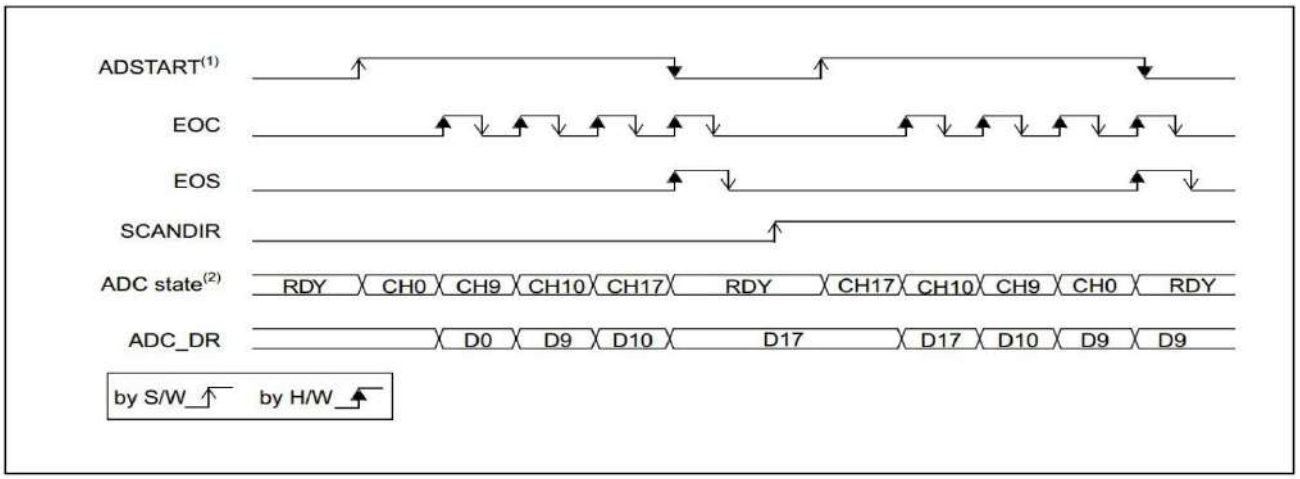
13.5.4 转换序列结束（EOSEQ 标志）

每次序列(EOSEQ)事件结束时，ADC 都会通知应用。

转换序列的上一数据结果出现在 ADC_DR 寄存器中时，ADC 会立即将 ADC_ISR 寄存器中的 EOSEQ 标志置 1。如果 ADC_IER 寄存器中的 EOSEQIE 位置 1，可产生中断。EOSEQ 标志可通过由软件向其写入 1 的方式来清零。

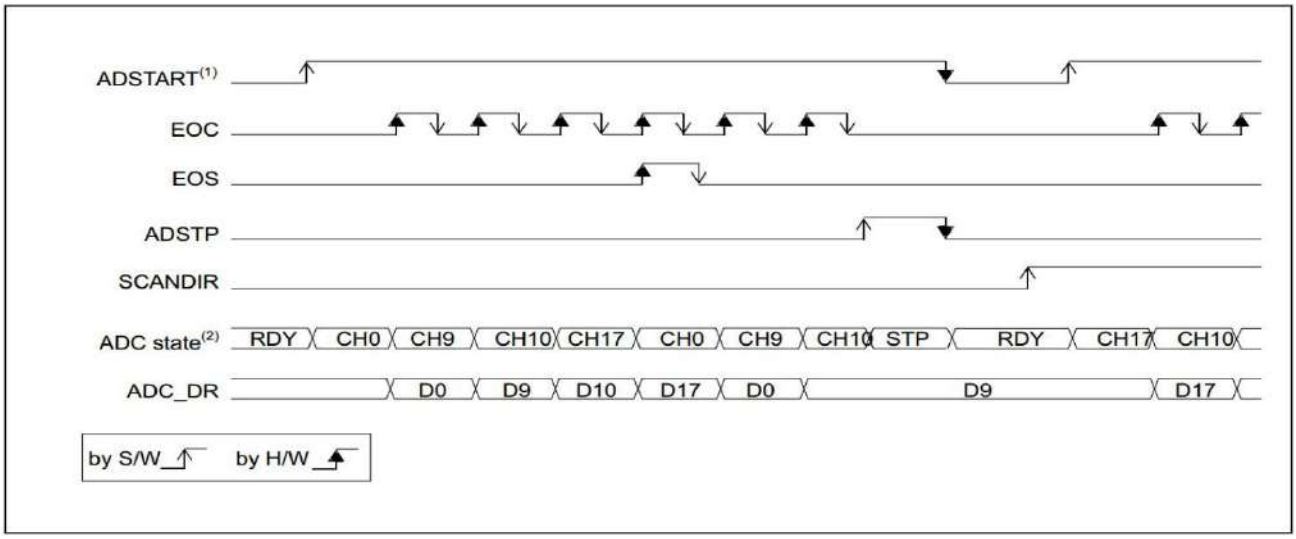
13.5.5 时序图示例（单次/连续模式硬件/软件触发）

图 13-9 单次序列转换，软件触发



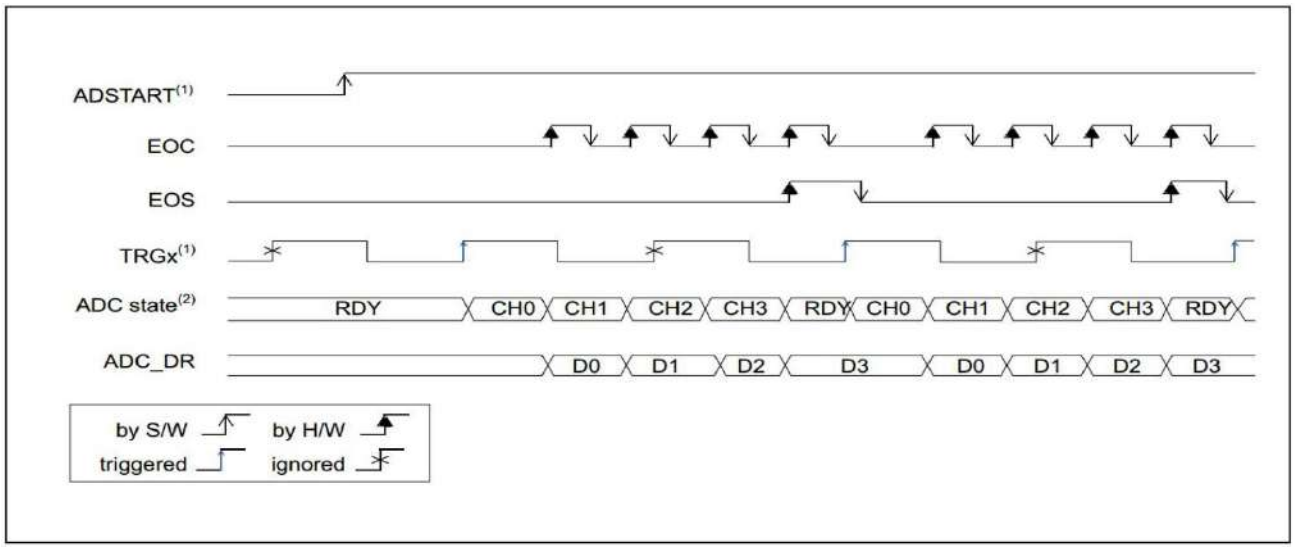
1. EXTEN=00，CONT=0
2. CHSEL=0x20601，WAIT=0，AUTOFF=0

图 13-10 连续序列转换，软件触发



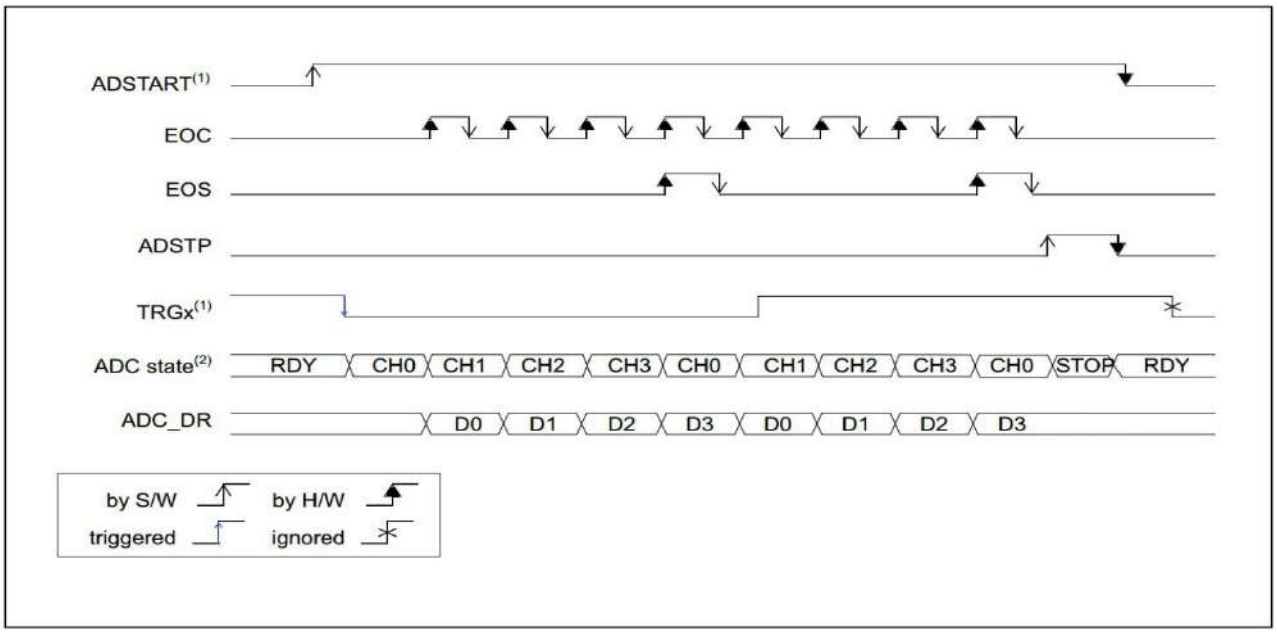
1. EXTEN=00，CONT=1，
2. CHSEL=0x20601，WAIT=0，AUTOFF=0

图 13-11 单次序列转换，硬件触发



1. EXTSEL=TRGx（过频），EXTEN=01（上升沿），CONT=0
2. CHSEL=0xF，SCANDIR=0，WAIT=0，AUTOFF=0

图 13-12 连续序列转换，硬件触发



1. EXTSEL=TRGx， EXTEN=10（下降沿）， CONT=1
2. CHSEL=0xF， SCANDIR=0， WAIT=0， AUTOFF=0

13.6 数据管理

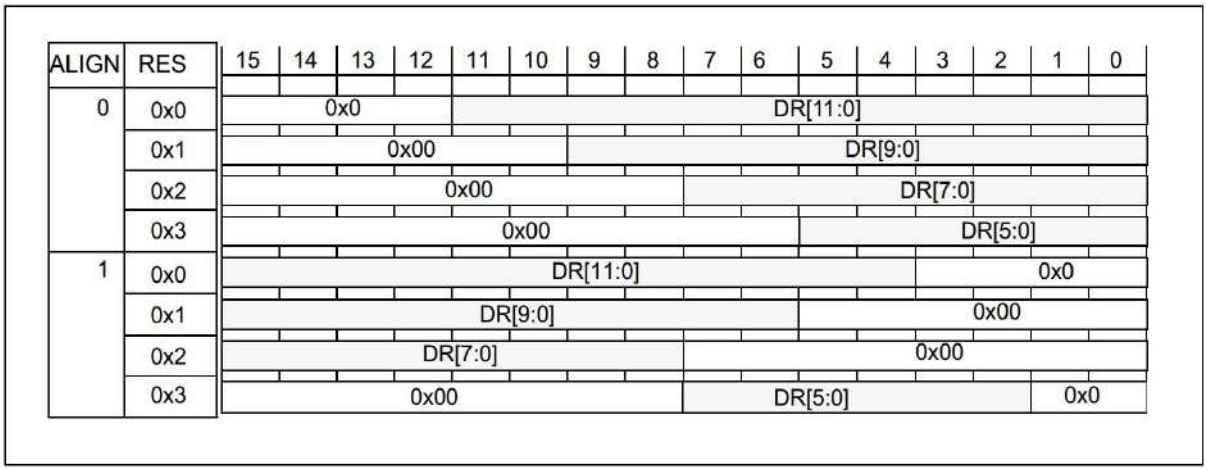
13.6.1 数据管理和数据对齐（ADC_DR、ALIGN）

每次转换结束时（发生 EOC 事件时），转换后数据的结果都会存储在宽度为 16 位的 ADC_DR 数据寄存器中。

ADC_DR 的格式取决于配置的数据对齐方式和分辨率。

ADC_CFGR1 寄存器中的 ALIGN 位用于选择转换后存储的数据的对齐方式。数据可右对齐(ALIGN=0)或左对齐(ALIGN=1)，如图 14.6.1_1 所示。

图 13-13 数据对齐方式和分辨率（过采样已禁止： OVSE = 0）



13.6.2 ADC 溢出（OVR、OVRMOD）

如果转换后的数据未由 CPU 或 DMA 及时读取，在新转换生成数据之前，会由溢出标志(OVR)指示数据溢出事件。

如果新转换完成时 EOC 标志仍为“1”，则 ADC_ISR 寄存器中的 OVR 标志会置 1。

如果 ADC_IER 寄存器中的 OVRIE 位置 1，可产生中断。

如果发生溢出情况，ADC 会保持工作状态并可继续进行转换，除非通过软件将 ADC_CR 寄存器中的 ADSTP 位置 1，从而停止并复位序列。

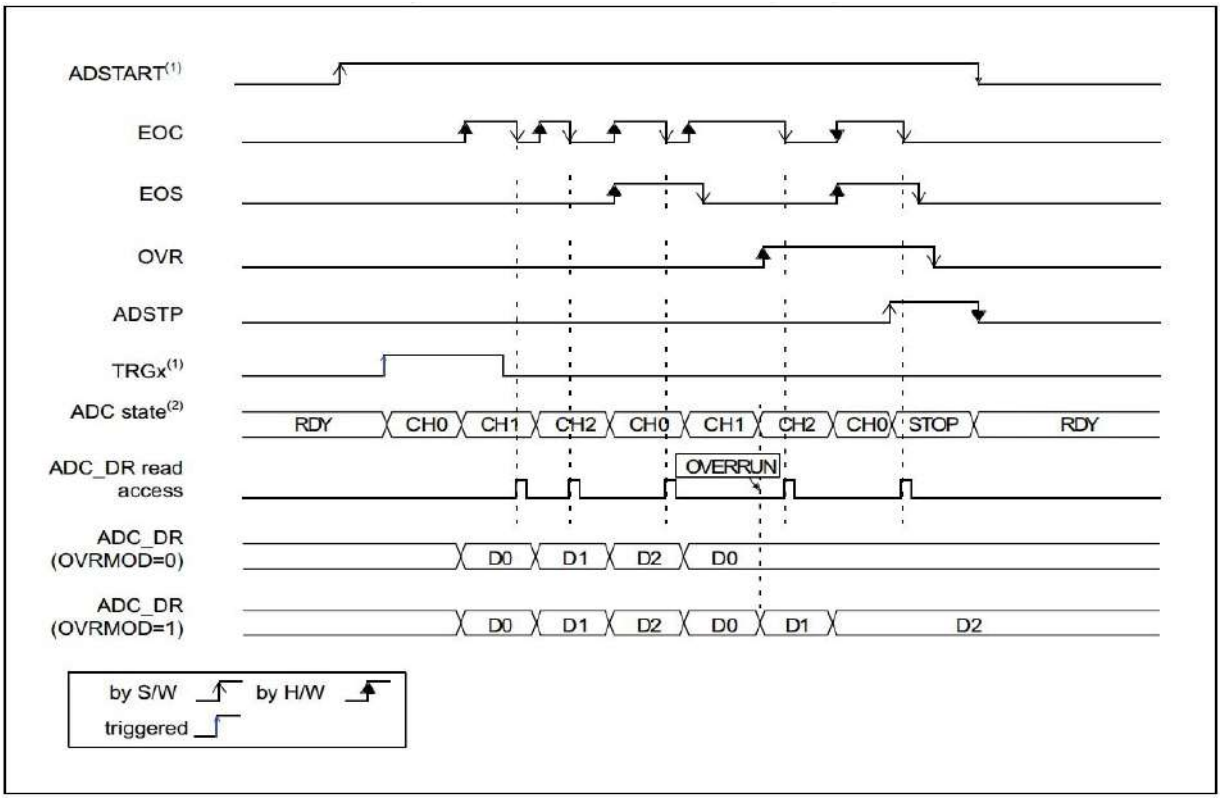
OVR 标志可通过由软件向其写入 1 的方式来清零。

可对 ADC_CFGR1 寄存器中的 OVRMOD 位进行编程，从而配置发生溢出事件时是要

保留数据还是要覆盖数据：

- OVRMOD=0
  - 溢出事件会保留数据寄存器的数据，防止其被覆盖：会保留原数据，并会丢弃新的转换结果。如果 OVR 保持为 1，可继续进行转换，但会丢弃所得的数据。
- OVRMOD=1
  - 数据寄存器会由上一次转换结果覆盖，之前未读取的数据会丢失。如果 OVR 保持为 1，可继续进行转换，ADC_DR 寄存器始终包含最新转换得出的数据。

图 13-14 溢出示例 (OVR)



13.6.3 在不使用 DMA 的情况下管理转换的数据序列

如果转换过程足够慢，则可使用软件来处理转换序列。在这种情况下，软件必须使用 EOC 标志及其相关中断来处理各个数据结果。每次转换完成时，都会将 ADC_ISR 寄存器中的 EOC 位置 1，并可读取 ADC_DR 寄存器。ADC_CFGR1 寄存器中的 OVRMOD 位应配置为 0，以便将溢出事件作为错误进行管理。

13.6.4 在不使用 DMA 且不发生溢出的情况下管理转换的数据

使 ADC 在转换一条或多条通道时无需在每次转换后都读取数据可能会很有用。在这种情况下，OVRMOD 位必须配置为 1，OVR 标志应被软件忽略。如果 OVRMOD=1，溢出事件不会阻止 ADC 继续进行转换，ADC_DR 寄存器始终包含最新的转换数据。

13.6.5 使用 DMA 管理转换的数据

由于转换得出的所有通道值都存储在单个数据寄存器中，因此，在转换多条通道时使用 DMA 可提高效率。这样可以避免存储在 ADC_DR 寄存器中的转换数据结果丢失。

若 DMA 模式已使能（ADC_CFGR1 寄存器中的 DMAEN 位置 1），则会在每个通道转换后生成 DMA 请求。这样便可将转换后的数据从 ADC_DR 寄存器传输到由软件选择的目标位置。

注：ADC 校准阶段完成后，必须将 ADC_CFGR1 寄存器中的 DMAEN 位置 1。

尽管如此，如果因 DMA 无法及时处理 DMA 传输请求而发生溢出(OVR=1)，ADC 会停止生成 DMA 请求，新转换对应的数据不会通过 DMA 进行传输。这意味着可将传输到 RAM 的所有数据都视为有效数据。

根据 OVRMOD 位的配置，可保留或覆盖数据（请参见 ADC 溢出（OVR、OVRMOD）章节）。

DMA 传输请求会禁止，直至软件将 OVR 位清零。

根据应用用途的不同，推荐使用两种不同的 DMA 模式，并使用 ADC_CFGR1 寄存器中的 DMACFG 位配置相应的模式：

- DMA 单次模式(DMACFG=0)。

如果通过编程将 DMA 设置为传输固定数目的数据字，应选择此模式。

- DMA 循环模式(DMACFG=1)

如果通过编程将 DMA 设置为循环模式或双缓冲区模式，应选择此模式。

DMA 单次模式(DMACFG=0)

在该模式下，每次出现新的转换数据字时，ADC 都会生成 DMA 传输请求，DMA 到达最后一个 DMA 传输操作时（发生 DMA_EOT 中断，请参见直接存储器访问控制器(DMA)），即使转换已再次开始，ADC 也会停止生成 DMA 请求。

DMA 传输完成后（在 DMA 控制器中配置的所有传输操作均已完成后）：

- ADC 数据寄存器的内容会冻结。
- 任何正在进行的转换都会中止，其部分结果会被丢弃
- 不会将任何新的 DMA 请求发送到 DMA 控制器。如果仍存在已开始的转换，这样可避免生成溢出错误。
- 扫描序列会停止并复位
- DMA 会停止

DMA 循环模式(DMACFG=1)

在该模式下，每次数据寄存器中出现新的转换数据字时，ADC 都会生成 DMA 传输请求，即使 DMA 已到达最后一次 DMA 传输操作也不例外。这样可将 DMA 配置为循环模式，从而处理连续的模拟输入数据流。

13.7 功耗特性

13.7.1 等待模式转换

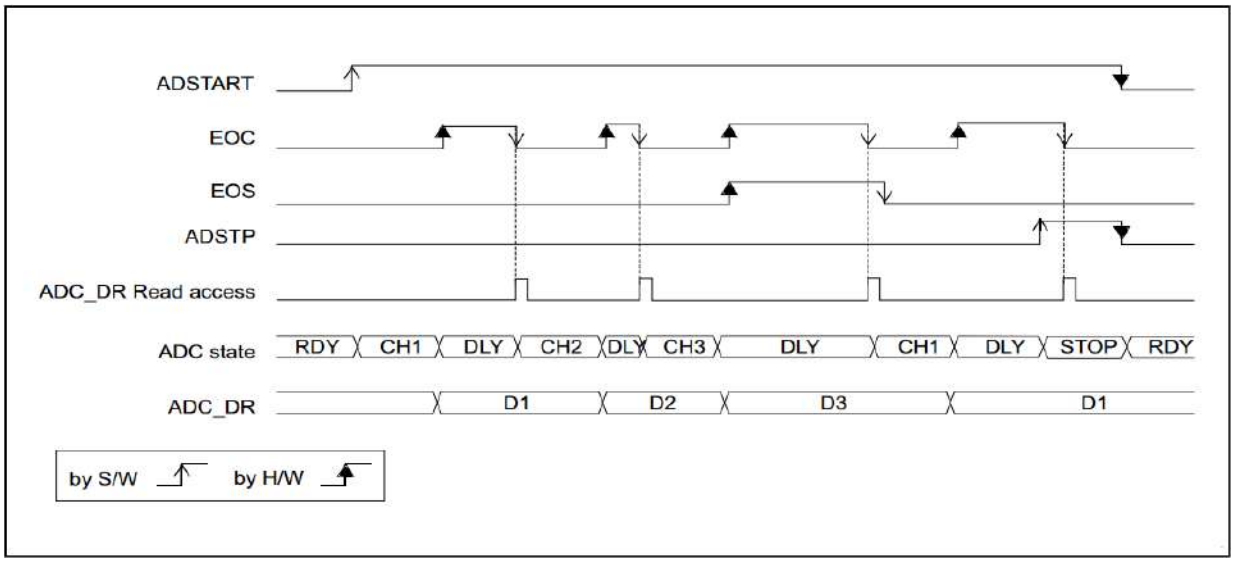
等待模式转换可用于简化软件，并可优化采用低频时钟的应用（此类应用可能存在 ADC 溢出的风险）的性能。

如果 ADC_CFGR1 寄存器中的 WAIT 位置 1，则仅当之前的数据已进行处理、ADC_DR 寄存器已读取或者 EOC 位已清零后，才会开始新的转换。

通过这种方式，可自动调整 ADC 的速度，使其适应系统读取数据的速度。

注：转换进行时发生的或在读访问之前的等待时间内发生的硬件触发会被忽略。

图 13-15　等待模式转换（连续模式，软件触发）



1. EXTEN=00，CONT=1
2. CHSEL=0x3，SCANDIR=0，WAIT=1，AUTOFF=0

13.7.2　自动关闭模式(AUTOFF)

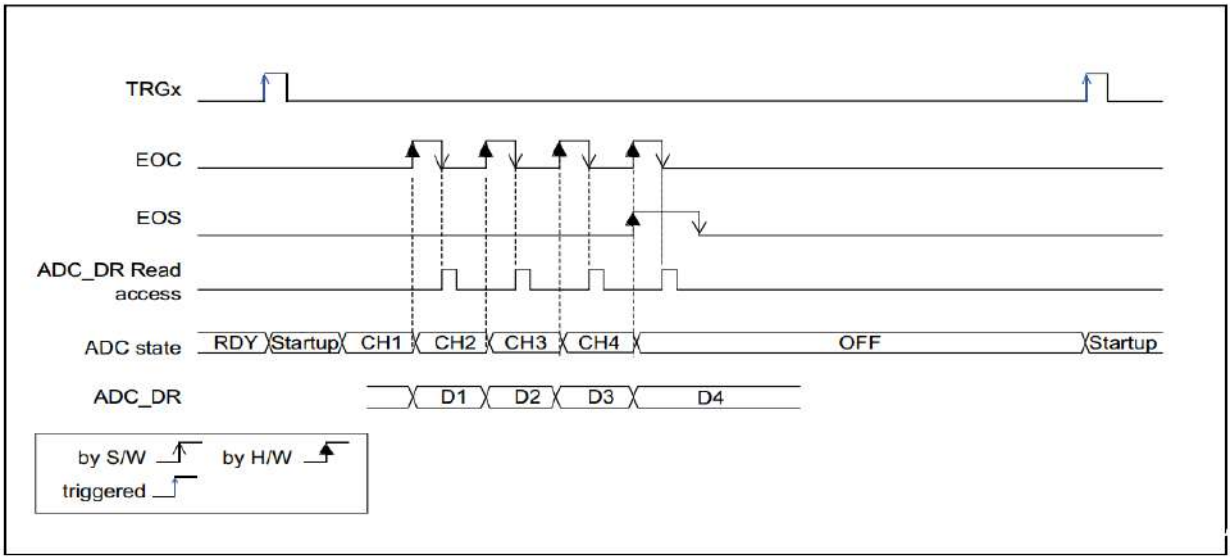
ADC 具有自动电源管理功能，也称为自动关闭模式，将 ADC_CFGR1 寄存器中的 AUTOFF 位置 1 可使能此模式。

如果 AUTOFF=1，ADC 始终会在未进行转换时关闭，并会在转换开始后自动唤醒（通过软件或硬件触发）。在启动转换的触发事件和 ADC 的采样时间之间，会自动插入启动时间。随后，转换序列完成后，ADC 会自动禁止。

如果应用需要进行的转换次数相对较少，或者为了证明开关 ADC 额外使用的功率和时间合理而将转换请求的间隔时间设定得足够长（例如采用低频硬件触发），使用自动关闭模式可显著降低应用的功耗。

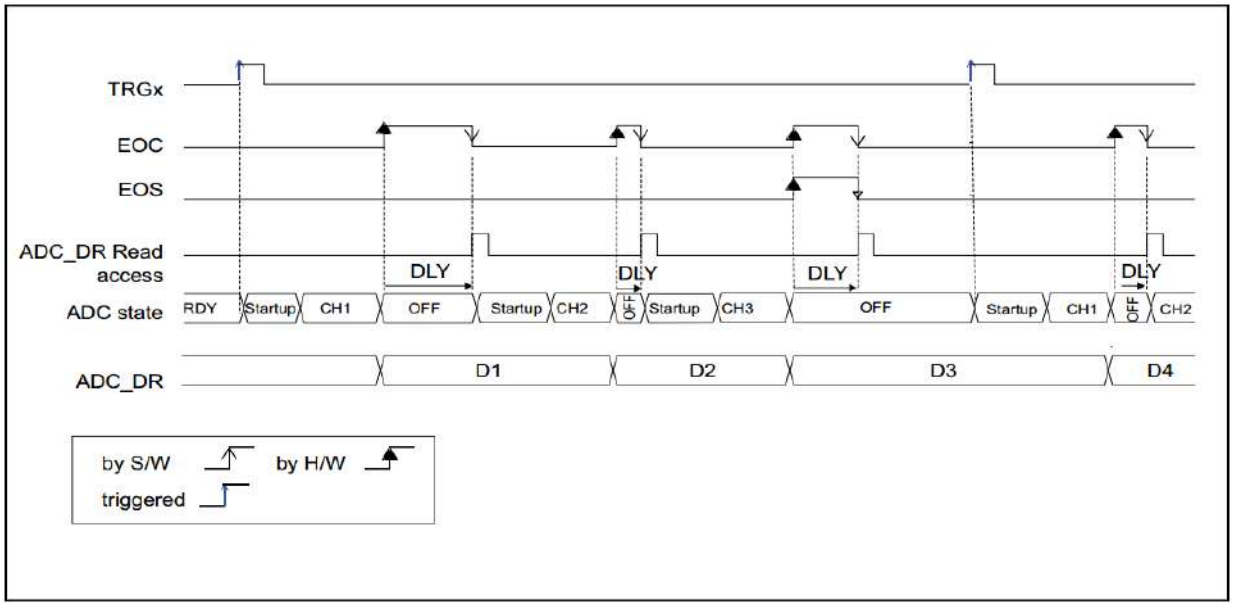
对于采用低频时钟的应用，可将自动关闭模式与等待模式转换(WAIT=1)结合使用，如果 ADC 在等待过程中自动掉电、并会在应用读取 ADC_DR 寄存器后立即重启，这种组合可显著降低功耗（请参见图 14.7.2_2：WAIT=1、AUTOFF=1 时的行为）。

图 13-16　WAIT=0、AUTOFF=1 时的行为



1. EXTSEL=TRGx，EXTEN=01（上升沿），CONT=x，ADSTART=1，CHSEL=0xF，SCANDIR=0，WAIT=1，AUTOFF=1

图 13-17　WAIT=1、AUTOFF=1 时的行为



1. EXTSEL=TRGx，EXTEN=01（上升沿），CONT=x，ADSTART=1，CHSEL=0xF，SCANDIR=0，WAIT=1，AUTOFF=1

13.8 模拟窗口看门狗（AWDEN、AWDSGL、AWDCH、AWD_HTR/LTR、AWD）

将 ADC_CFGR1 寄存器中的 AWDEN 位置 1，可使能 AWD 模拟看门狗功能。此功能用于监控一条选定的通道或所有已使能的通道（请参见表 14.8_2：模拟看门狗通道选择）是否仍处于所配置的电压范围（窗口）内，如图 14.8_1 所示。

如果 ADC 转换的模拟电压低于阈值下限或高于阈值上限，则会将 AWD 模拟看门狗状态位置 1。这些阈值在 ADC_HTR 和 ADC_LTR16 位寄存器的 12 个最低有效位中进行编程。可以通过将 ADC_IER 寄存器中的 AWDIE 位置 1 来使能中断。

AWD 标志可通过由软件向其写入 1 的方式来清零。

如果转换的数据分辨率小于 12 位（取决于 DRES[1:0]位），由于始终会在内部对完整的 12 位原始转换数据进行比较（左对齐），因此编程阈值的 LSB 必须保持清零状态。

表 14.8_1 介绍了如何对所有可能的分辨率进行比较。

表 13-7　模拟看门狗比较

分辨率位 RES[1:0]	模拟看门狗比较对象：		注释
	原始转换数据，左对齐	阈值	
00：12 位	DATA[11:0]	LT[11:0]和 HT[11:0]	-
01：10 位	DATA[11:2]，00	LT[11:0]和 HT[11:0]	用户必须将 LT1[1:0]和 HT1[1:0]配置为“00”
10：8 位	DATA[11:4]，0000	LT[11:0]和 HT[11:0]	用户必须将 LT1[3:0]和 HT1[3:0]配置为“0000”
11：6 位	DATA[11:6]，000000	LT[11:0]和 HT[11:0]	用户必须将 LT1[5:0]和 HT1[5:0]配置为“000000”

1. 进行任何对齐计算之前，会对原始转换数据进行看门狗比较。

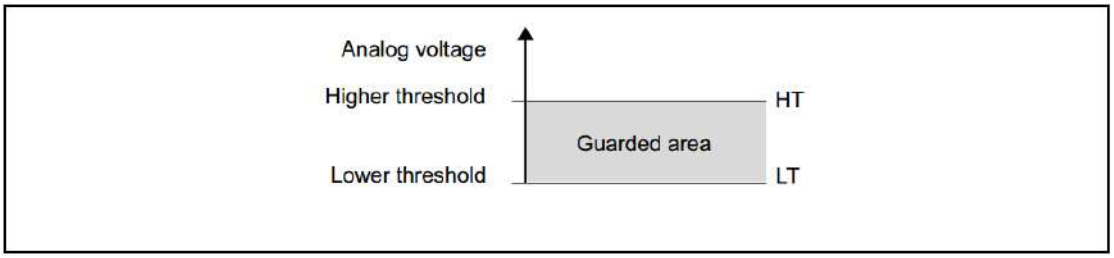
表 13 7 介绍了如何配置 ADC_CFGR1 寄存器中的 AWDSGL 位和 AWDEN 位，以使能一条或多条通道上的模拟看门狗。

表 13-8　模拟看门狗通道选择

模拟看门狗保护的通道	AWDSGL 位	AWDEN 位
无	X	0
所有通道	0	1
单通道	1	1

1. 通过 AWDCH[4:0]位选择。

图 13-18　模拟看门狗的保护区域



13.9 过采样器

过采样单元会进行数据预处理，以减轻 CPU 的负担。过采样单元可处理多个转换，并计算多个转换结果的平均值，得到数据宽度增大（高达 16 位）的单个数据。

它提供的结果采用以下形式，其中的 N 和 M 可以进行调整：

$$\text{结果} = \frac{1}{M} \times \sum_{n=0}^{n=N-1} \text{转换}(t_n)$$

允许通过硬件执行以下功能：计算平均值、降低数据速率、改进 SNR 以及基本滤波。

过采样率 N 通过 ADC_CFGR2 寄存器中的 OVFS[2:0]位进行定义，它的范围是 2x 到 256x。分频系数 M 由向右移位构成，最多可移 8 位。它可通过 ADC_CFGR2 寄存器中的 OVSS[3:0]位进行配置。

求和单元可得出多达 20 位（256x12 位）的结果，结果会先右移。随后，会将结果的高位截断，只留下 16 个最低有效位，这些位使用移位后剩下的最低有效位四舍五入为最接近的数值，然后将最终得到的结果传输到 ADC_DR 数据寄存器中。

注：如果移位后得到的中间结果超过 16 位，则会截断结果的高位。

图 13-19 20 位到 16 位结果的截断过程

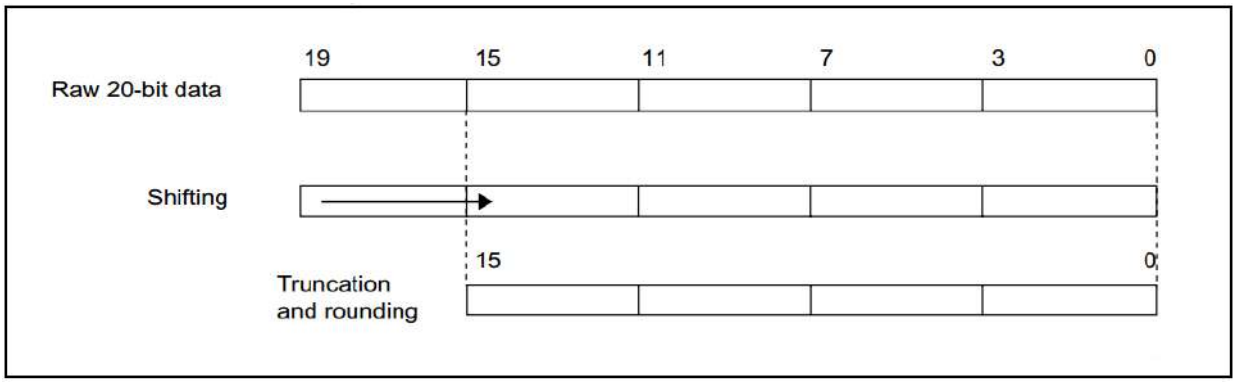
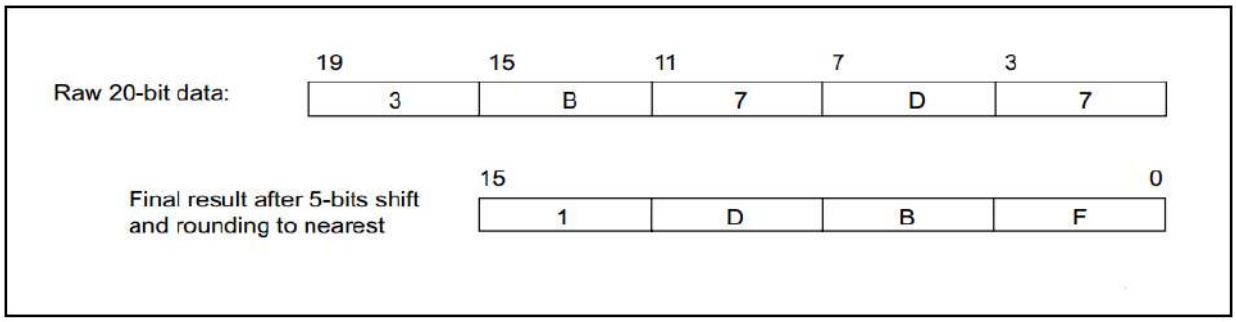


图 14.9_2 介绍了从原始的 20 位累加数据到最终 16 位结果的数值处理过程。

图 13-20 移 5 位并进行舍入的数值示例



以下表 13 8 列出了原始转换数据等于 0xFFF 时对应的各种 N 和 M 组合的数据格式。

表 13-9 最大输出结果与 N 和 M 的对应关系。呈灰色显示的数值表示截断的部分

过采样率	最大原始数据	不移位 OVSS = 0000	移 1 位 OVSS = 0001	移 2 位 OVSS = 0010	移 3 位 OVSS = 0011	移 4 位 OVSS = 0100	移 5 位 OVSS = 0101	移 6 位 OVSS = 0110	移 7 位 OVSS = 0111	移 8 位 OVSS = 1000
2x	0x1FFE	0x1FFE	0x0FFF	0x0800	0x0400	0x0200	0x0100	0x0080	0x0040	0x0020
4x	0x3FFC	0x3FFC	0x1FFE	0x0FFF	0x0800	0x0400	0x0200	0x0100	0x0080	0x0040
8x	0x7FF8	0x7FF8	0x3FFC	0x1FFE	0x0FFF	0x0800	0x0400	0x0200	0x0100	0x0080
16x	0xFFF0	0xFFF0	0x7FF8	0x3FFC	0x1FFE	0x0FFF	0x0800	0x0400	0x0200	0x0100
32x	0x1FFE0	0xFFE0	0xFFF0	0x7FF8	0x3FFC	0x1FFE	0x0FFF	0x0800	0x0400	0x0200
64x	0x3FFC0	0xFFC0	0xFFE0	0xFFF0	0x7FF8	0x3FFC	0x1FFE	0x0FFF	0x0800	0x0400
128x	0x7FF80	0x7FF80	0xFFC0	0xFFE0	0xFFF0	0x7FF8	0x3FFC	0x1FFE	0x0FFF	0x0800
256x	0xFFF00	0xFF00	0xFF80	0xFFC0	0xFFE0	0xFFF0	0x7FF8	0x3FFC	0x1FFE	0x0FFF

与标准转换模式相比，过采样模式下的转换时序不会发生变化：整个过采样序列中，采样时间保持相等。每完成 N 次转换都会提供新数据，等效延迟等于 N x tADC = N x (tSMPL + tSAR)。

各标志的情况如下：

- 每个采样阶段后都会将采样阶段结束标志 (EOSMP) 置 1
- 如果过采样结果可用，每完成 N 次转换都会发生转换结束事件 (EOC)
- 过采样数据序列完成后（即 N x 序列长度次转换之后），会发生序列结束事件 (EOCSEQ)

13.9.1 过采样时支持的 ADC 工作模式

在过采样模式下，大部分 ADC 工作模式都可用：

- 单次或连续模式转换、向前或向后扫描序列

- 可由软件或触发器启动 ADC 转换
- ADC 在转换过程中停止（中止）
- 通过 CPU 或 DMA 在支持溢出检测的情况下读取数据
- 低功耗模式（WAIT、 AUTOFF）
- 可编程分辨率：在这种情况下，会按照与 12 位转换相同的方式对分辨率降低的转换值（根据 ADC_CFGR1 寄存器中的 RES[1:0] 位）进行累加、截断、四舍五入和移位。

注： 处理过采样数据时，不可使用对齐模式。 ADC_CFGR1 中的 ALIGN 位会被忽略，且数据始终采用右对齐格式。

13.9.2 模拟看门狗

可以使用模拟看门狗功能（AWDSGL 位和 AWDEN 位），但存在以下区别：

- 会忽略 RES[1:0] 位，始终会使用完整的 12 位值 HT[11:0] 和 LT[11:0] 进行比较。
- 会比较 16 位过采样结果 ADC_DR[15:4] 的 12 个最高有效位

注： 移位位数较大时必须多加留意，因为这样会缩小比较范围。例如，如果过采样结果移了 4 位，得到 12 位右对齐数据，那么只能对 8 个数据位执行有效的模拟看门狗比较。比较操作会在 ADC_DR[11:4] 与 HT[0:7]/LT[0:7] 之间进行，并且 HT[11:8]/LT[11:8] 必须保持复位状态。

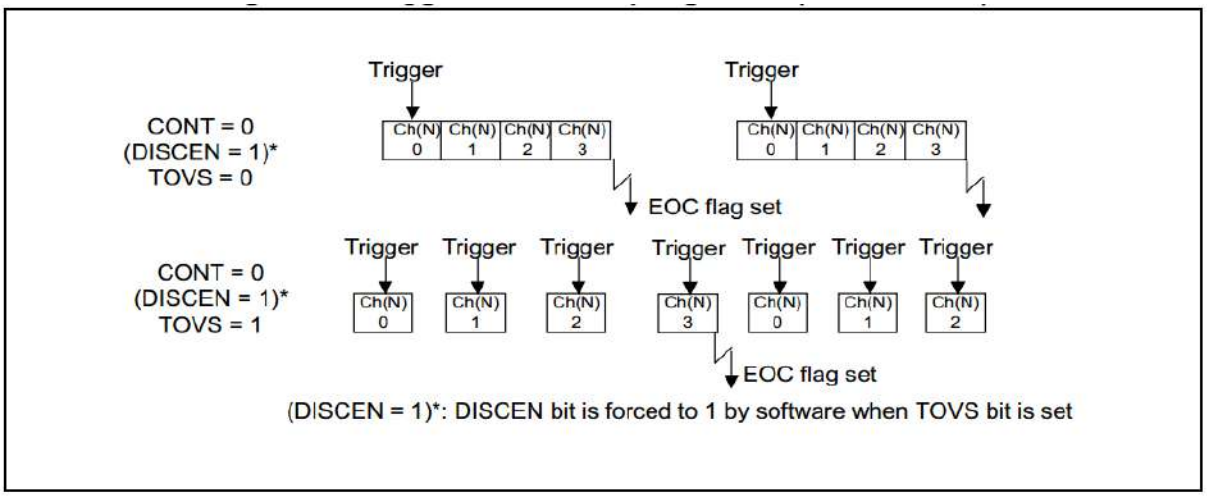
13.9.3 触发模式

平均值计算单元还可用于基本滤波，虽然它不是效率非常高的滤波器（衰减缓慢、停止频段衰减受限），但可用作陷波滤波器，用于抑制恒定的寄生频率（通常来自电源或切换模式电源）。为此，可使用 ADC_CFGR2 中的 TOVS 位使能特定的不连续模式，以获得由用户定义、且与转换时间本身无关的过采样频率。

图 13 21 显示了在不连续模式下如何响应触发从而开始转换。

如果 TOVS 位置 1，则会忽略 DISCEN 位，并将该位视为 1。

图 13-21 已触发的过采样模式（TOVS 位 = 1）



13.10 温度传感器和内部参考电压

温度传感器可用于测量器件的结温(TJ)。温度传感器在内部连接到 ADC_IN18 输入通道，该通道用于将传感器输出电压转换为数字值。温度传感器模拟引脚的采样时间必须大于数据手册中指定的最小 TS_temp 值。不使用时可将传感器置于掉电模式。

内部参考电压(VREFINT)为 ADC 和比较器提供了一个稳定的（带隙基准）电压输出。VREFINT 内部连接到 ADC_IN17 输入通道。VREFINT 的精确电压由 ST 在生产测试期间对每部分单独测量，并存储于系统存储区。

访问模式为只读。

图 14.10_1 显示的是温度传感器、内部参考电压与 ADC 之间连接的方框图。

必须将 TSEN 位置 1 才能使能 ADC_IN18（温度传感器）的转换，必须将 VREFEN 位置 1 才能使能 ADC_IN17(VREFINT)的转换。

温度传感器的输出电压随温度线性变化。由于工艺不同，该线的偏移量取决于各个芯片（芯片之间的温度变化可达 45°C）。

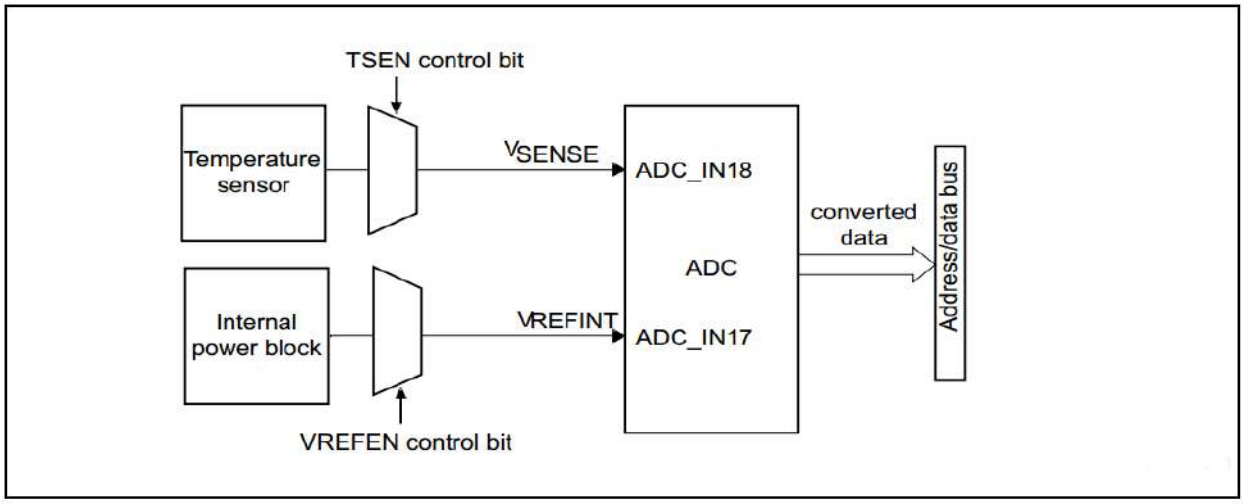
未校准的内部温度传感器更适用于对温度变量而非绝对温度进行测量的应用。为提高温度传感器测量的准确性，ST 在生产过程中将校准值存储在每个器件的系统存储器中。

在制造过程中，会将温度传感器的校准数据和内部参考电压存储在系统存储区。随后，用户应用可读取这些数据，并使用这些数据提高温度传感器或内部参考的准确性。其他相关信息，请参见数据手册。

主要特性

- 支持的温度范围： –40 °C 到 125 °C
- 线性度：最高 ±2 °C，精度取决于校准情况

图 13-22 温度传感器和 VREFINT 通道方框图



读取温度

1. 选择 ADC_IN18 输入通道
2. 选择器件数据手册中规定的合适的采样时间(TS_temp)
3. 将 ADC_CCR 寄存器中的 TSEN 位置 1，将温度传感器从掉电模式中唤醒，并等待其稳定时间(tSTART)
4. 将 ADC_CR 寄存器中的 ADSTART 位置 1（或通过外部触发），开始 ADC 转换
5. 读取 ADC_DR 寄存器中生成的 VSENSE 数据
6. 使用以下公式计算温度：

$$\text{Temperature(in°C)} = \frac{130°C - 30°C}{TS\_CAL2 - TS\_CAL1} \times (TS\_DATA - TS\_CAL1) + 30°C$$

$$\text{Temperature(in°C)} = \frac{V_{30} - V_{sense}}{Avg\_Slope} + 30°C$$

其中：

- TS_CAL2 是在 130°C 下获得的温度传感器校准值
- TS_CAL1 是在 30°C 下获得的温度传感器校准值
- TS_DATA 是由 ADC 转换得到的实际温度传感器输出值

更多关于 TS_CAL1 和 TS_CAL2 校准点的信息，请参见相应的器件数据手册。

注：传感器从掉电模式中唤醒需要一个启动时间，启动时间过后其才能输出正确的 VSENSE。ADC 在上电后同样需要一个启动时间，因此，为尽可能缩短延迟时间，应同时将 ADEN 和 TSEN 位置 1。

使用内部参考电压计算实际的 VDDA 电压

施加给微控制器的 VDDA 电源电压可能会有变化，或无法获得准确值。在制造过

程中由 ADC 在 VDDA=3V 的条件下获得的内置内部参考电压(VREFINT)及其校准数据可用于评估实际的 VDDA 电压水平。

以下公式可求得为器件供电的实际的 VDDA 电压：

$$VDDA=3V\times VREFINT\_CAL/VREFINT\_DATA$$

其中：

- VREFINT_CAL 是 VREFINT 校准值
- VREFINT_DATA 是由 ADC 转换得到的实际 VREFINT 输出值

将电源相关的 ADC 测量值转换为绝对电压值

ADC 用于提供对应于模拟电源与施加给转换通道的电压之比的数字值。对于大部分应用用例，需要将该比值转换成与 VDDA 无关的电压。对于 VDDA 已知、 ADC 转换值进行了右对齐的应用，可使用以下公式得到该绝对值：

$$\mathbf{V_{CHANNELx}} = \frac{\mathbf{V_{DDA}}}{\mathbf{FULL\_SCALE}} \times \mathbf{ADC\_DATA_x}$$

对于 VDDA 值未知的应用，必须使用内部参考电压， VDDA 可替换为使用内部参考电压计算实际的 VDDA 电压部分提供的表达式，从而得出以下公式：

$$\mathbf{V_{CHANNELx}} = \frac{\mathbf{3V \times VREFINT\_CAL \times ADC\_DATA_x}}{\mathbf{VREFINT\_DATA \times FULL\_SCALE}}$$

其中：

- VREFINT_CAL 是 VREFINT 校准值
- ADC_DATAx 是由 ADC 在通道 x 上测得的值（右对齐）
- VREFINT_DATA 是由 ADC 转换得到的实际 VREFINT 输出值
- FULL_SCALE 是 ADC 输出的最大数字值。例如，如果分辨率为 12 位，该值为2^{12} - 1 = 4095，如果分辨率为 8 位，该值为 2^8 - 1 = 255。

注： 如果执行 ADC 测量时使用的是输出格式而非 12 位右对齐格式，那么必须先将所有参数转换为兼容格式，然后再进行计算。

13.11 VLCD 电压监控

ADC_CCR 寄存器中的 VLCDEN 位用于测量 VLCD 引脚上的 LCD 电源电压。由于 VLCD 电压可能高于 VDDA，为确保 ADC 正常工作，VLCD 引脚需要内部连接到桥接分配器。VLCDEN 位置 1 时，会自动使能此桥，以将 LCD_VLCD1 连接到 ADC1_IN16 输入通道。因此，转换得出的数字值是 VLCD 电压的三分之一（LCD 配置为 1/3 偏置时）或 VLCD 电压的四分之一（LCD 配置为 1/4 偏置或 1/2 偏置时）。为防止电池出现意外的电能消耗，建议仅在必要时使能桥接分配器，即执行 ADC 转换。

13.12 ADC 中断

发生下列任一事件均可生成中断：

- 校准结束（EOCAL 标志）

- ADC 就绪后， ADC 上电（ADRDY 标志）
- 任何转换结束（EOC 标志）
- 转换序列结束（EOSEQ 标志）
- 进行模拟看门狗检测时（AWD 标志）
- 采样阶段结束时（EOSMP 标志）
- 发生数据溢出时（OVR 标志）

可以使用单独的中断使能位以提高灵活性。

表 13-10 ADC 中断

中断事件	事件标志	使能控制位
校准结束	EOCCAL	EOCALIE
ADC 就绪	ADRDY	ADRDYIE
转换结束	EOC	EOCIE
转换序列结束	EOSEQ	EOSEQIE
模拟看门狗状态位置 1	AWD	AWDIE
采样阶段结束	EOSMP	EOSMPIE
上溢	OVR	OVRIE

13.13 AWD 唤醒功能

系统在 STOP 模式下可以通过 RTC 计时发出信号到 ADC，ADC 采到该信号去唤醒 ADC 时钟，时钟准备好后触发 ADC 转换，根据 ADC 转换结果大小产生 AWD 事件，AWD 事件输出到 EXTI 就可以唤醒系统。

使用该功能除了配置 AWD 相关的阈值和通道设置以外还需要配置 WAKE_EN 寄存器，包括 BEEPER 内部对应的计时控制和 ADC 的唤醒功能使能。

WAKE_EN: 使能 AWD 唤醒功能

ADC 的 AWD 唤醒功能注意事项：

BEEPER 输出的触发信号不需要设置触发控制寄存器 EXTEN 和 EXTSEL；

ADC 的配置都在这种模式下生效，所以必须使能 AWDEN，不能使能循环和间断模式，在转换的过程中除 ADC_SR.AWD 以外的状态寄存器不会被置位；

和其他触发信号一样的效果触发 ADC 单个通道或全部通道的转换，如果在单个通道或全部通道转换完成后 ADC_SR.AWD 状态位没有置起则重新回到 STOPMODE。

AWD 的唤醒信号根据 AWDSGL 和 AWDCH 设置可以单个通道的结果比较也可以所有通道的结果比较，直接把 ADC_SR.AWD 发到 EXTI，不需要使能 AWDIE；

13.14 ADC 差分输入

支持差分输入模式，序号相连的偶数通道和奇数通道互相组成差分输入，如 AIN8 和 AIN9；

SDIF: 差分输入使能

GCMP: ADC 内部延迟控制。

13.15 ADC 寄存器

13.15.1 ADC 中断和状态寄存器(ADC_ISR)

偏移地址：0x00

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res	Res.	Res.	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	EOCAL	Res.	Res.	Res.	AWD	Res.	Res.	OVR	EOSEQ	EOC	EOSMP	ADRDY
				rw				rw			rw	rw	rw	rw	rw
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位 31:12	保留，必须保持复位值。
位 11	**EOCAL**：校准结束标志 (End Of Calibration flag) 校准完成时，该位由硬件置 1。通过软件写入 1 可将该位清零。 0：校准未完成 1：校准已完成
位 10:8	保留，必须保持复位值。
位 7	**AWD**：模拟看门狗标志 (Analog watchdog flag) 当转换电压超过在 ADC_LTR 和 ADC_HTR 寄存器中编程的值时，硬件会将该位置 1。通过软件写入 1 可将该位清零。 0：未发生模拟看门狗事件（或标志事件已通过软件确认并清零）。 1：发生模拟看门狗事件
位 6:5	保留，必须保持复位值。
位 4	**OVR**： ADC 溢出 (ADC overrun) 该位在发生溢出事件时由硬件置 1，这意味着在 EOC 标志已置 1 时，新转换已完成。通过软件写入 1 可将该位清零。 0：未发生溢出事件（或标志事件已通过软件确认并清零）。 1：发生溢出
位 3	**EOSEQ**：序列结束标志 (End of sequence flag) 在由 CHSEL 位选择的一系列通道转换结束时，会通过硬件将该位置 1。通过软件写入 1 可将该位清零。 0：转换序列未完成（或标志事件已通过软件确认并清零）。 1：转换序列已完成
位 2	**EOC**：转换结束标志 (End of conversion flag) 当通道的每次转换结束，新数据结果出现在 ADC_DR 寄存器时，会通过硬件将该位置 1。通过软件向 该位写入 1，或读取 ADC_DR 寄存器都可将该位清零。 0：通道转换未完成（或标志事件已通过软件确认并清零）。 1：通道转换已完成
位 1	**EOSMP**：采样结束标志 (End of sampling flag) 在转换过程中，当采样阶段结束时该位由硬件置 1。通过软件将该位编程为“1”，可将该位清零。 0：采样阶段未结束（或标志事件已通过软件确认并清零）。 1：采样阶段已结束
位 0	**ADRDY**： ADC 就绪（ADC ready） ADC 使能后（位 ADEN=1）以及 ADC 达到准备好接收转换请求的状态时，会通过硬件将该位置 1。 通过软件写入 1 可将该位清零。 0： ADC 未准备好开始转换（或标志事件已通过软件确认并清零）。 1： ADC 已准备好开始转换

13.15.2 ADC 中断使能寄存器(ADC_IER)

偏移地址：0x04

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res	Res.	Res.	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	EOCALIE	Res.	Res.	Res.	AWDIE	Res.	Res.	OVRIE	EOSEQIE	EOCIE	EOSMPIE	ADRDYIE
				rw				rw			rw	rw	rw	rw	rw
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位 31:12	保留，必须保持复位值。
位 11	**EOCALIE**：校准结束中断使能 (End of calibration interrupt enable) 此位由软件置 1 和清零，用于使能/禁止校准结束中断。 0：禁止校准结束中断 1：使能校准结束中断 注： 仅当 ADSTART=*0* 时（这可确保当前未进行任何转换），才允许通过软件对此位执行写操作。
位 10:8	保留，必须保持复位值。
位 7	**AWDIE**：模拟看门狗中断使能 (Analog watchdog interrupt enable) 此位由软件置 1 和清零，用于使能/禁止模拟看门狗中断。 0：禁止模拟看门狗中断 1：使能模拟看门狗中断 注： 仅当 ADSTART=*0* 时（这可确保当前未进行任何转换），才允许通过软件对此位执行写操作。
位 6:5	保留，必须保持复位值。
位 4	**OVRIE**：溢出中断使能 (Overrun interrupt enable) 此位由软件置 1 和清零，用于使能/禁止溢出中断。 0：禁止溢出中断 1：使能溢出中断 OVR 位置 1 时产生中断。 注： 仅当 ADSTART=*0* 时（这可确保当前未进行任何转换），才允许通过软件对此位执行写操作。
位 3	**EOSEQIE**：转换序列结束中断使能 (End of conversion sequence interrupt enable) 此位由软件置 1 和清零，用于使能/禁止转换序列结束中断。 0：禁止 EOSEQ 中断 1：使能 EOSEQ 中断 EOSEQ 位置 1 时产生中断。 注： 仅当 ADSTART=*0* 时（这可确保当前未进行任何转换），才允许通过软件对此位执行写操作。
位 2	**EOCIE**：转换结束中断使能 (End of conversion interrupt enable) 此位由软件置 1 和清零，用于使能/禁止转换结束中断。 0：禁止 EOC 中断 1：使能 EOC 中断 EOC 位置 1 时产生中断。 注： 仅当 ADSTART=*0* 时（这可确保当前未进行任何转换），才允许通过软件对此位执行写操作。
位 1	**EOSMPIE**：采样结束中断使能 (End of sampling interrupt enable) 此位由软件置 1 和清零，用于使能/禁止采样阶段结束中断。 0：禁止 EOSMP 中断。 1：使能 EOSMP 中断。 EOSMP 位置 1 时产生中断。 注： 仅当 ADSTART=*0* 时（这可确保当前未进行任何转换），才允许通过软件对此位执行写操作。
位 0	**ADRDYIE**： ADC 就绪中断使能 (ADC ready interrupt enable) 此位由软件置 1 和清零，用于使能/禁止 ADC 就绪中断。 0：禁止 ADRDY 中断。 1：使能 ADRDY 中断。 ADRDY 位置 1 时产生中断。 注： 仅当 ADSTART=*0* 时（这可确保当前未进行任何转换），才允许通过软件对此位执行写操作。

13.15.3 ADC 控制寄存器(ADC_CR)

偏移地址：0x08

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADCAL	Res	Res.	Res.	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
rs															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res	Res.	Res.	Res.	Res	Res.	Res.	ADSTP	Res	ADSTART	ADDIS	ADEN
											rs		rs	rs	rs
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位 31	**ADCAL**： ADC 校准 (ADC calibration) 该位由软件置 1，用于开始 ADC 校准。 校准完成后，该位由硬件清零。 0：校准已完成 1：写入 1 可校准 ADC。读取值为 1 表示正在进行校准。 注： 仅当 ADC 已禁止时（ADCAL=*0*、 ADSTART=*0*、 ADSTP=*0*、 ADDIS=*0* 且 ADEN=*0*），才允许通过软件将 ADCAL 位置 *1*。 注： 仅当 ADEN=*1* 且 ADSTART=*0*（ADC 已使能，当前未进行任何转换）时，才允许通过软件对 ADC_CALFACT 执行写操作来更新校准系数。
位 30:5	保留，必须保持复位值。
位 4	**ADSTP**： ADC 停止转换命令 (ADC stop conversion command) 该位由软件置 1，用于停止和丢弃正在进行的转换（ADSTP 命令）。 当转换已有效丢弃、并且 ADC 已准备好接收新的开始转换命令时，会通过硬件将该位清零。 0：当前未执行 ADC 停止转换命令 1：写入 1 可停止 ADC。读取值为 1 表示正在执行 ADSTP 命令。 注： 仅当 ADSTART=1 且 ADDIS=0 时（ADC 已使能、可能正在进行转换、并且没有任何待处理的禁止 ADC 的请求），才允许通过软件将 ADSTP 置 1。
位 3	保留，必须保持复位值。
位 2	**ADSTART**： ADC 开始转换命令 (ADC start conversion command) 此位由软件置 1，用于开始 ADC 转换。根据 EXTEN [1:0] 配置位的值，可以立即开始转换（软件触发配置），也可以在发生硬件触发事件后开始转换（硬件触发配置）。 该位通过硬件清零： – 在单次转换模式下（CONT=0、 DISCEN=0），如果选择了软件触发 (EXTEN=00)：出现转换序列结束 (EOSEQ) 标志时清零。 – 在不连续转换模式下（CONT=0、 DISCEN=1），如果选择了软件触发 (EXTEN=00)：出现转换结束(EOC) 标志时清零。 – 在所有其他情况下：执行完 ADSTP 命令后，由硬件将 ADSTP 位清零的同时清零。 0：当前未进行 ADC 转换。 1：写入 1 可开始 ADC。读取值为 1 表示 ADC 正在工作，可能正在进行转换。 注： 仅当 ADEN=1 且 ADDIS=0 时（ADC 已使能，并且没有任何待处理的禁止 ADC 的请求），才允许通过软件将 ADSTART 置 1。
位 1	**ADDIS**： ADC 禁止命令 (ADC disable command) 该位通过软件置 1，用于禁止 ADC（ADDIS 命令）并使其进入掉电状态（OFF 状态）。 ADC 已有效禁止后，会立即通过硬件将该位清零（此时也会通过硬件将 ADEN 清零）。 0：当前未执行 ADDIS 命令 1：写入 1 可禁止 ADC。读取值为 1 表示正在执行 ADDIS 命令。 注： 仅当 ADEN=1 且 ADSTART=0 时（这可确保当前未进行任何转换），才允许通过软件将 ADDIS 置 1。

位 0	**ADEN**： ADC 使能命令 (ADC enable command) 该位通过软件置 1，用于使能 ADC。 ADRDY 标志置 1 后， ADC 将立即准备好运行。 如果 ADC 已禁止，则执行 ADDIS 命令后，将通过硬件对该位清零。 0：禁止 ADC（OFF 状态） 1：写入 1 来使能 ADC。 注： 仅当 ADC_CR 寄存器的所有位均为 0 时（ADCAL=0、 ADSTP=0、 ADSTART=0、 ADDIS=0 且 ADEN=0），才允许通过软件将 ADEN 位置 1。

13.15.4 ADC 配置寄存器 1(ADC_CFGR1)

偏移地址：0x0C

复位值：0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AUTOFF.	WAIT	CONT	OVRMOD	EXTEN[1:0]		Res.	EXTSEL[2:0]			ALIGN	RES[1:0]		SCANDIR	DMACFG	DMAEN
rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位 31	保留，必须保持复位值。
位 31	保留，必须保持复位值。
位 30:26	**AWDCH[4:0]**：模拟看门狗通道选择 (Analog watchdog channel selection) 这些位由软件置 1 和清零。它们用于选择由模拟看门狗监控的输入通道。 00000：通过 AWD 监控 ADC 模拟输入通道 0 00001：通过 AWD 监控 ADC 模拟输入通道 1 ..... 10010：通过 AWD 监控 ADC 模拟输入通道 18 其他值：保留，不得使用 注： 由 AWDCH[4:0] 位选择的通道也必须设置到 CHSELR 寄存器中。 注： 仅当 ADSTART=0 时（这可确保当前未进行任何转换），才允许通过软件对这些位执行写操作。
位 25:24	保留，必须保持复位值。
位 23	**AWDEN**：模拟看门狗使能 (Analog watchdog enable) 此位由软件置 1 和清零。 0：禁止模拟看门狗 1：使能模拟看门狗 注： 仅当 ADSTART=0 时（这可确保当前未进行任何转换），才允许通过软件对此位执行写操作。
位 22	**AWDSGL**：在单一通道或所有通道上使能看门狗 (Enable the watchdog on a single channel or on all channels) 此位由软件置 1 和清零，用于使能由 AWDCH[4:0] 位确定的通道或所有通道上的模拟看门狗。 0：在所有通道上使能模拟看门狗 1：在单一通道上使能模拟看门狗 注： 仅当 ADSTART=0 时（这可确保当前未进行任何转换），才允许通过软件对此位执行写操作。
位 21:17	保留，必须保持复位值。

位 16	**DISCEN**：不连续模式 (Discontinuous mode) 此位由软件置 1 和清零，用于使能/禁止不连续模式。 0：禁止不连续模式 1：使能不连续模式 注： 不能同时使能不连续模式和连续模式：禁止同时将 DISCEN 和 CONT 位置 1。 注： 仅当 ADSTART=0 时（这可确保当前未进行任何转换），才允许通过软件对此位执行写操作。
位 15	**AUTOFF**：自动关闭模式 (Auto-off mode) 此位由软件置 1 和清零，用于使能/禁止自动关闭模式。 . 0：禁止自动关闭模式 1：使能自动关闭模式 注： 仅当 ADSTART=0 时（这可确保当前未进行任何转换），才允许通过软件对此位执行写操作。
位 14	**WAIT**：等待转换模式 (Wait conversion mode) 此位由软件置 1 和清零，用于使能/禁止等待转换模式。 . 0：等待转换模式关闭 1：等待转换模式开启 注： 仅当 ADSTART=0 时（这可确保当前未进行任何转换），才允许通过软件对此位执行写操作。
位 13	**CONT**：单次/连续转换模式 (Single/continuous conversion mode) 此位由软件置 1 和清零。该位置 1 时，转换将持续进行，直到该位清零。 0：单次转换模式 1：连续转换模式 注： 不能同时使能不连续模式和连续模式：禁止同时将 DISCEN 和 CONT 位置 1。 注： 仅当 ADSTART=0 时（这可确保当前未进行任何转换），才允许通过软件对此位执行写操作。
位 12	**OVRMOD**：溢出管理模式 (Overrun management mode) 该位通过软件进行置 1 和清零，并用于配置数据溢出的管理方式。 0：如果检测到溢出， ADC_DR 寄存器会保留原有数据。 1：如果检测到溢出， ADC_DR 寄存器会被上一转换结果覆盖。 注： 仅当 ADSTART=0 时（这可确保当前未进行任何转换），才允许通过软件对此位执行写操作。
位 11:10	**EXTEN[1:0]**：外部触发使能和极性选择 (External trigger enable and polarity selection) 这些位由软件置 1 和清零，用于选择外部触发极性并使能触发。 00：禁止硬件触发检测（可通过软件开始转换） 01：在上升沿执行硬件触发检测 10：在下降沿执行硬件触发检测 11：在上升沿和下降沿都执行硬件触发检测 注： 仅当 ADSTART=0 时（这可确保当前未进行任何转换），才允许通过软件对这些位执行写操作
位 9	保留，必须保持复位值。
位 8:6	**EXTSEL[2:0]**：外部触发选择 (External trigger selection) 这些位可选择用于触发转换开始的外部事件（有关详情，请参见表 14.5_2 外部触发器）： 000： TRG0 001： TRG1 010： TRG2 011： TRG3 100： TRG4 101： TRG5 110： TRG6 111： TRG7 注： 仅当 ADSTART=0 时（这可确保当前未进行任何转换），才允许通过软件对这些位执行写操作。

位 5	**ALIGN**：数据对齐 (Data alignment) 此位由软件置 1 和清零，用于选择右对齐或左对齐。请参见图 14.6.1_1 数据对齐方式和分辨率（过采样已禁止： OVSE = 0） 。 0：右对齐 1：左对齐 注： 仅当 ADSTART=0 时（这可确保当前未进行任何转换），才允许通过软件对此位执行写操作。
位 4:3	**RES[1:0]**：数据分辨率 (Data resolution) 通过软件写入这些位可选择转换的分辨率。 00： 12 位 01： 10 位 10： 8 位 11： 6 位 注： 仅当 ADEN=0 时，才允许通过软件对这些位执行写操作。
位 2	**SCANDIR**：扫描序列方向 (Scan sequence direction) 此位由软件置 1 和清零，用于选择扫描序列中各通道的方向。 0：向前扫描（CHSEL0 到 CHSEL19） 1：向后扫描（CHSEL19 到 CHSEL0） 注： 仅当 ADSTART=0 时（这可确保当前未进行任何转换），才允许通过软件对此位执行写操作。
位 1	**DMACFG**：直接存储器访问配置 (Direct memory access configuration) 此位由软件置 1 和清零，用于在 DMA 两种工作模式之间进行选择，仅当 DMAEN=1 时，该位才有效。 0：选择 DMA 单次模式 1：选择 DMA 循环模式 更多详细信息，请参见第 14.6.5 节：使用 DMA 管理转换的数据 注： 仅当 ADSTART=0 时（这可确保当前未进行任何转换），才允许通过软件对此位执行写操作。
位 0	**DMAEN**：直接存储器访问使能 (Direct memory access enable) 此位由软件置 1 和清零，用于使能 DMA 请求的生成。这样便可使用 DMA 控制器自动管理转换的数据。有关详细信息，请参见第 14.6.5 节：使用 DMA 管理转换的数据。 0：禁止 DMA 1：使能 DMA 注： 仅当 ADSTART=0 时（这可确保当前未进行任何转换），才允许通过软件对此位执行写操作。

13.15.5 ADC 配置寄存器 2(ADC_CFGR2)

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
CKMODE[1:0]		Res.	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
rw	rw														
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res	Res.	TOVS	OVSS[3:0]				OVSR[2:0]			Res.	OVSE
						rw	rw	rw	rw	rw	rw	rw	rw		rw
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位 31:30	**CKMODE[1:0]**： ADC 时钟模式 (ADC clock mode) 此位由软件置 1 和清零，用于定义为模拟 ADC 提供时钟的方式： 00： ADCCLK（异步时钟模式），在产品级生成（请参见 RCC 部分） 01： PCLK/2（同步时钟模式） 10： PCLK/4（同步时钟模式） 11： PCLK（同步时钟模式）。仅当 PCLK 的时钟占空比为 50% 时，才必须使能此配置（必须绕过 RCC 中配置的 APB 预分频系数，并且系统时钟占空比必须在 50% 以内）。 在所有异步时钟模式下，从定时器触发到转换开始的延迟过程中，不存在抖动。 注： 仅当 ADC 已禁止时（ADCAL=0、 ADSTART=0、 ADSTP=0、 ADDIS=0 且 ADEN=0），才允许通过软件对这些位执行写操作。
位 29:10	保留，必须保持复位值。
位 9	**TOVS**：已触发过采样 (Triggered Oversampling) 此位由软件置 1 和清零。 0：会在触发后连续完成某一通道的所有过采样转换 1：某一通道的每个过采样转换都需要触发 注： 仅当 ADSTART=0 时（这可确保当前未进行任何转换），才允许通过软件对此位执行写操作。
位 8:5	**OVSS[3:0]**：过采样移位 (Oversampling shift) 此位由软件置 1 和清零。 0000：不发生移位 0001：移 1 位 0010：移 2 位 0011：移 3 位 0100：移 4 位 0101：移 5 位 0110：移 6 位 0111：移 7 位 1000：移 8 位 其他代码保留 注： 仅当 ADSTART=0 时（这可确保当前未进行任何转换），才允许通过软件对此位执行写操作。
位 4:2	**OVSR[2:0]**：过采样率 (Oversampling ratio) 该位域定义过采样率的数值。 000： 2x 001： 4x 010： 8x 011： 16x 100： 32x 101： 64x 110： 128x 111： 256x 注： 仅当 ADSTART=0 时（这可确保当前未进行任何转换），才允许通过软件对此位执行写操作。
位 1	保留，必须保持复位值。
位 0	**OVSE**：过采样器使能 (Oversampler Enable) 此位由软件置 1 和清零。 0：禁止过采样器 1：使能过采样器 注： 仅当 ADSTART=0 时（这可确保当前未进行任何转换），才允许通过软件对此位执行写操作。

13.15.6 ADC 配置寄存器 3(ADC_CFGR3)

偏移地址：0x3f0

复位值：0x0000 0000

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

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

位 31	**WAKE_EN**: 使能 AWD 唤醒功能(Analog watchdog weak up enable) 使能后可以在 STOP 模式下检测到 RTC 的计时信号输出。 此位由软件置 1 和清零。 0: AWD 唤醒关闭（默认） 1: AWD 唤醒使能
位 30:2	保留，必须保持复位值。
位 1	**SDIF**: 差分输入使能（The differential input enable） 使能后如通道 0 和 1,2 和 3 等奇数和偶数通道组成差分输入。 此位由软件置 1 和清零。 0: ADC 通道单端输入模式（默认） 1: ADC 通道差分输入模式
位 0	保留，必须保持复位值。

13.15.7 ADC 采样时间寄存器(ADC_SMPR)

偏移地址：0x14

复位值：0x0000 0000

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

位 31:3	保留，必须保持复位值。
位 2:0	**SMP[2:0]**：采样时间选择 (Sampling time selection) 这些位由软件写入，用于选择应用于所有通道的采样时间。 000： 1.5 个 ADC 时钟周期 001： 3.5 个 ADC 时钟周期 010： 7.5 个 ADC 时钟周期 011： 12.5 个 ADC 时钟周期 100： 19.5 个 ADC 时钟周期 101： 39.5 个 ADC 时钟周期 110： 79.5 个 ADC 时钟周期 111： 160.5 个 ADC 时钟周期 注： 仅当 ADSTART=0 时（这可确保当前未进行任何转换），才允许通过软件对这些位执行写操作。

13.15.8 ADC 看门狗阀值寄存器(ADC_TR)

偏移地址：0x20

复位值：0x0000 0FFF

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

位 31:28	保留，必须保持复位值。
位 27:16	**HT[11:0]**： 模拟看门狗阈值上限 (Analog watchdog higher threshold) 这些位由软件写入，用于定义模拟看门狗的阈值上限。请参见第 14.8 节：模拟窗口看门狗（AWDEN、 AWDSGL、 AWDCH、 AWD_HTR/LTR、 AWD） 。 注： 仅当 ADSTART=0 时（这可确保当前未进行任何转换），才允许通过软件对这些位执行写操作。
位 15:12	保留，必须保持复位值。
位 11:0	**LT[11:0]**：模拟看门狗阈值下限 (Analog watchdog lower threshold) 这些位由软件写入，用于定义模拟看门狗的阈值下限。 请参见第 14.8 节：模拟窗口看门狗（AWDEN、AWDSGL、AWDCH、AWD_HTR/LTR、AWD） 。 注： 仅当 ADSTART=*0* 时（这可确保当前未进行任何转换），才允许通过软件对这些位执行写操作。

13.15.9 ADC 通道选择寄存器(ADC_CHSELR)

偏移地址：0x28

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CHSEL19	CHSEL18	CHSEL17	CHSEL16
												rw	rw	rw	rw
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CHSEL15	CHSEL14	CHSEL13	CHSEL12	CHSEL11	CHSEL10	CHSEL9	CHSEL8	CHSEL7	CHSEL6	CHSEL5	CHSEL4	CHSEL3	CHSEL2	CHSEL1	CHSEL0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位 31:19	保留，必须保持复位值。
位 18:0	**CHSELx**：通道 x 选择 (Channel-x selection) 这些位由软件写入，用于定义将哪些通道纳入要转换的通道序列。 0：未选择输入通道 x 进行转换 1：已选择输入通道 x 进行转换 注： 仅当 ADSTART=*0* 时（这可确保当前未进行任何转换），才允许通过软件对这些位执行写操作。

13.15.10 ADC 数据寄存器(ADC_DR)

偏移地址：0x40

复位值：0x0000 0000

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

Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位 31:16	保留，必须保持复位值。
位 15:0	**DATA[15:0]**：转换后的数据 (Converted data) 这些位为只读。其中包含上一转换通道的转换结果。数据可以采用左对齐和右对齐，如图 14.6.1_1：数据对齐方式和分辨率（过采样已禁止： OVSE = 0） 所示。 校准完成后， DATA[5:0] 会立即包含校准系数。

13.15.11 ADC 通用配置寄存器(ADC_CCR)

偏移地址：0x308

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	VLCDEN	TSEN	VREFEN	PRESC[3:0]				Res.	Res.
						rw	rw	rw	rw	rw	rw	rw	rw		
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

位 31:25	保留，必须保持复位值。
位 24	**VLCDEN**： VLCD 使能 (VLCD enable) 此位由软件置 1 和清零，用于使能/禁止 VLCD 读取电路。 0：禁止 VLCD 读取电路 1：使能 VLCD 读取电路 注： 仅当 ADSTART=0 时（这可确保当前未进行任何转换），才允许通过软件对此位执行写操作。
位 23	**TSEN**：温度传感器使能 (Temperature sensor enable) 此位由软件置 1 和清零，用于使能/禁止温度传感器。 0：禁止温度传感器 1：使能温度传感器 注： 仅当 ADSTART=0 时（这可确保当前未进行任何转换），才允许通过软件对此位执行写操作。
位 22	**VREFEN**： VREFINT 使能 (VREFINT enable) 此位由软件置 1 和清零，用于使能/禁止 VREFINT 通道。 0：禁止 VREFINT 1：使能 VREFINT 注： 仅当 ADSTART=0 时（这可确保当前未进行任何转换），才允许通过软件对此位执行写操作

位 21:18	**PRESC[3:0]**： ADC 预分频系数 (ADC prescaler) 此位由软件置 1 和清零，用于选择 ADC 的时钟频率。该时钟为所有 ADC 所共用。 0000：输入 ADC 时钟未分频 0001：输入 ADC 时钟 2 分频 0010：输入 ADC 时钟 4 分频 0011：输入 ADC 时钟 6 分频 0100：输入 ADC 时钟 8 分频 0101：输入 ADC 时钟 10 分频 0110：输入 ADC 时钟 12 分频 0111：输入 ADC 时钟 16 分频 1000：输入 ADC 时钟 32 分频 1001：输入 ADC 时钟 64 分频 1010：输入 ADC 时钟 128 分频 1011：输入 ADC 时钟 256 分频 其他：保留 注： 仅当 ADC 已禁止时（ADCAL=0、 ADSTART=0、 ADSTP=0、 ADDIS=0 且 ADEN=0），才允 许通过软件对这些位执行写操作。
位 17:0	保留，必须保持复位值。

14 数模转换器 (DAC)

14.1 简介

DAC 模块是 12 位电压输出数模转换器。 DAC 可以按 8 位或 12 位模式进行配置，并且可与 DMA 控制器配合使用。在 12 位模式下，数据可以采用左对齐或右对齐。可使用输入参考电压 VREF+（与 ADC 共享）。可以有选择地缓冲输出以实现更高的电流驱动。

14.2 DAC 主要特性

器件集成了 1 个 DAC 转换器，具有一个输出通道： DAC_OUT1。

DAC1 主要特性如下：

- 一个数据保持寄存器
- 12 位模式下数据采用左对齐或右对齐
- 同步更新功能
- 生成噪声波
- 生成三角波
- 双 DAC 通道，支持单独或同时转换
- DMA 功能（包括下溢检测）
- 通过外部触发信号进行转换
- 输入参考电压 VREF+

图图 14 1 所示为 DAC 通道的框图，表 14 1 则给出了引脚说明。

图 14-1 DAC 通道框图

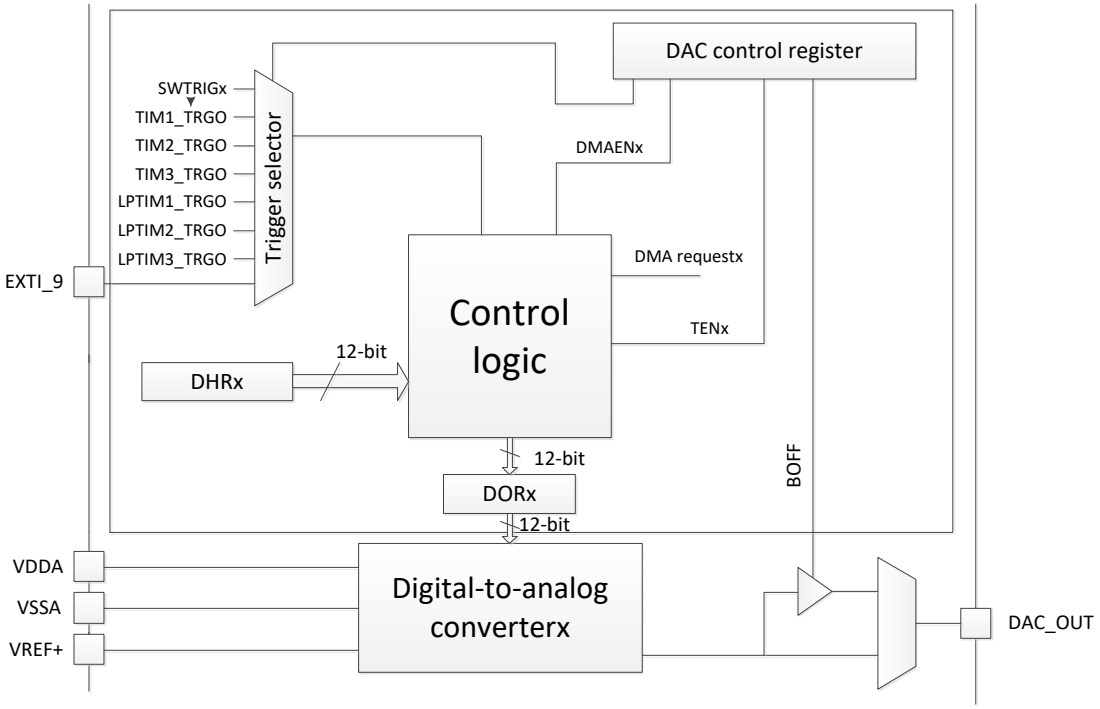


表 14-1 ADC 引脚

名称	信号类型	备注
VDDA	模拟电源输入	模拟电源

VSSA	模拟电源地输入	模拟电源地
VREF+	正模拟参考电压输入	DAC1 的高端/正极参考电压
DAC_OUT1/2	模拟输出信号	DAC 通道 x 模拟输出

注：使能 DAC_ChannelX 后，相应 GPIO 引脚将自动连接到模拟转换器输出(DAC_OUTx)。为了避免寄生电流消耗，应首先将引脚配置为模拟模式 (AIN)。

14.3 DAC 输出缓冲器使能

DAC 集成了 1 个输出缓冲器，可用来降低输出阻抗并在不增加外部运算放大器的情况下直接驱动外部负载。

通过 DAC_CR 寄存器中的 BOFF1 位，可使能或禁止 DAC 通道输出缓冲器。

14.4 DAC 通道使能

将 DAC_CR 寄存器中的相应 ENx 位置 1，即可使能对应 DAC 通道。经过一段启动时间 tWAKEUP 后，各 DAC 通道被真正使能。

注： ENx 位只会使能模拟 DAC 通道 x 宏单元。即使 ENx 位复位， DAC 通道 x 数字接口仍处于使能状态。

14.5 DAC 功能说明

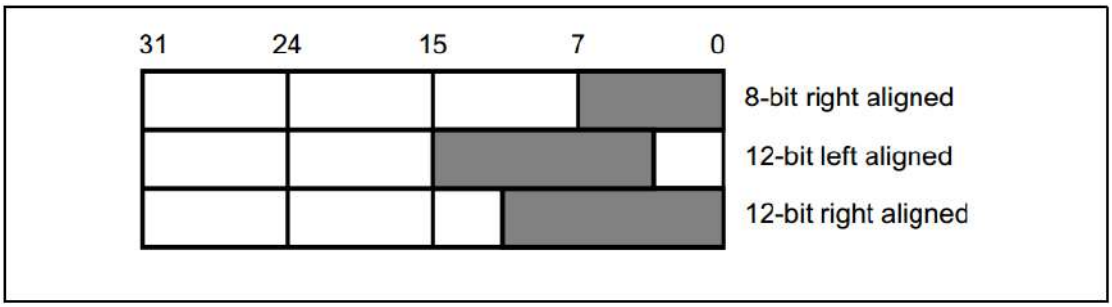
14.5.1 DAC 数据格式

根据所选配置模式，数据必须按如下方式写入指定寄存器：

- 存在三种可能：
  - 8 位右对齐：软件必须将数据加载到 DAC_DHR8R [7:0] 位（存储到 DHR [11:4] 位）。
  - 12 位左对齐：软件必须将数据加载到 DAC_DHR12L [15:4] 位（存储到 DHR [11:0] 位）。
  - 12 位右对齐：软件必须将数据加载到 DAC_DHR12R [11:0] 位（存储到 DHR [11:0] 位）。

根据加载的 DAC_DHRyyy 寄存器，用户写入的数据将移位并存储到相应的 DHR（数据保持寄存器，即内部非存储器映射寄存器）。之后， DHR 寄存器将被自动加载，或者通过软件或外部事件触发加载到 DOR 寄存器。

图 14-2 DAC 单通道模式下的数据寄存器



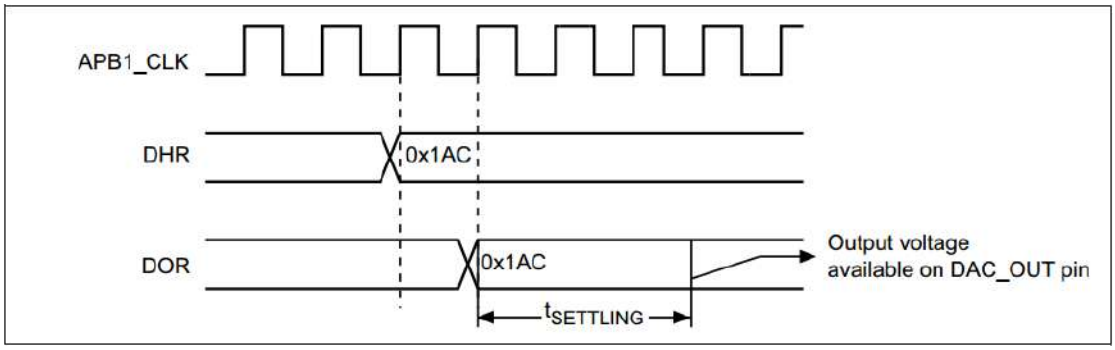
14.5.2 DAC 通道转换

DAC_DOR 无法直接写入，任何数据都必须通过加载 DAC_DHR 寄存器（写入 DAC_DHR8R、DAC_DHR12L、DAC_DHR12R）才能传输到 DAC 通道。

如果未选择硬件触发（DAC_CR 寄存器中的 TEN 位复位），那么经过一个 APB1 时钟周期后，DAC_DHR 寄存器中存储的数据将自动转移到 DAC_DOR 寄存器。但是，如果选择硬件触发（置位 DAC_CR 寄存器中的 TEN 位）且触发条件到来，将在三个 PCLK1 时钟周期后进行转移。

当 DAC_DOR 加载了 DAC_DHR 内容时，模拟输出电压将在一段时间 tSETTLING 后可用，具体时间取决于电源电压和模拟输出负载。

图 14-3　关闭触发(TEN=0)时的转换时序图 TEN=0



生成单个 LFSR 的独立触发

要将 DAC 配置为此转换模式（请参见第 15.6 节：生成噪声），需要遵循以下顺序：

1. 将 DAC 通道触发使能位 TEN 置 1。
2. 通过设置 TSEL[2:0]位配置触发源。
3. 将 DAC 通道的 WAVE[1:0]位配置为“01”，并在 MAMP[3:0]位中配置相同的 LFSR 掩码值。
4. 将 DAC 通道数据加载到所需 DAC_DHRx 寄存器（DHR12R、DHR12L 或 DHR8R）。

DAC 通道 x 触发信号到达时，LFSRx 计数器内容（使用相同的掩码）与 DHRx 寄存器内容相加，所得总和转移到 DAC_DORx 中（三个 APB 时钟周期之后）。LFSRx 计数器随即更新。

生成单个三角波的独立触发

要将 DAC 配置为此转换模式（请参见第 15.7 节：生成三角波），需要遵循以下顺序：

1. 将 DAC 通道触发使能位 TEN 置 1。
2. 通过设置 TSEL[2:0]位配置触发源。
3. 将 DAC 通道的 WAVE[1:0]位配置为“1x”，并在 MAMP[3:0]位中配置相同的最大振幅值。
4. 将 DAC 通道数据加载到所需 DAC_DHR 寄存器（DHR12R、DHR12L 或 DHR8R）。

DAC 通道触发信号到达时，DAC 通道三角波计数器内容（使用相同的三角波振幅）与 DHR 寄存器内容相加，所得总和转移到 DAC_DOR 中（三个 APB 时钟周期之后）。DAC 通道三角波计数器随即更新。

14.5.3 DAC 输出电压

- 经过线性转换后，数字输入会转换为 0 到 VREF+ 之间的输出电压。
- 各 DAC 通道引脚的模拟输出电压通过以下公式确定：

$$DACoutput = V_{REF} \times \frac{DOR}{4096}$$

14.5.4 DAC 触发选择

如果 TEN 控制位置 1，可通过外部事件（定时计数器、外部中断线）触发转换。TSEL[2:0]控制位将决定哪个可能事件将触发转换，如表 15.5.4_1 所示。

表 14-2　外部触发器

源	类型	TSEL[2:0]
TIM1 TRGO 事件	片上定时器的内部信号	000

TIM3 TRGO 事件		001
TIM3 CH3 TRGO 事件		010
		011
TIM2 TRGO 事件		100
		101
EXTI 线 9	外部引脚	110
SWTRIG	软件控制位	111

每当 DAC 接口在所选定时器 TRGO 输出或所选外部中断线 9 上检测到上升沿时，

DAC_DHRx 寄存器中存储的最后一个数据即会转移到 DAC_DORx 寄存器中。发生触发后再经过三个 APB1 周期，DAC_DORx 寄存器将会得到更新。

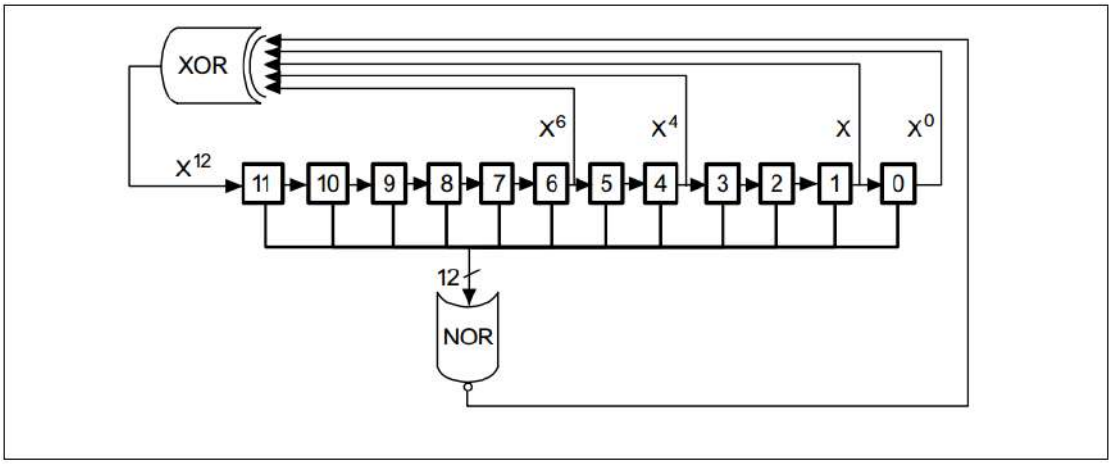
如果选择软件触发，一旦 SWTRIG 位置 1，转换即会开始。DAC_DHRx 寄存器内容加载到 DAC_DORx 寄存器中后，SWTRIG 即由硬件复位。

注：TEN 位置 1 时，无法更改 TSEL[2:0]位。如果选择软件触发，DAC_DHRx 寄存器的内容只需一个 APB1 时钟周期即可转移到 DAC_DOR 寄存器。

14.6 生成噪声

为了生成可变振幅的伪噪声，可使用 LFSR（线性反馈移位寄存器）。将 WAVE[1:0]置为“01”即可选择生成噪声。LFSR 中的预加载值为 0xAAA。在每次发生触发事件后，经过三个 APB 时钟周期，该寄存器会依照特定的计算算法完成更新。

图 14-4DACLFSR 寄存器计算算法

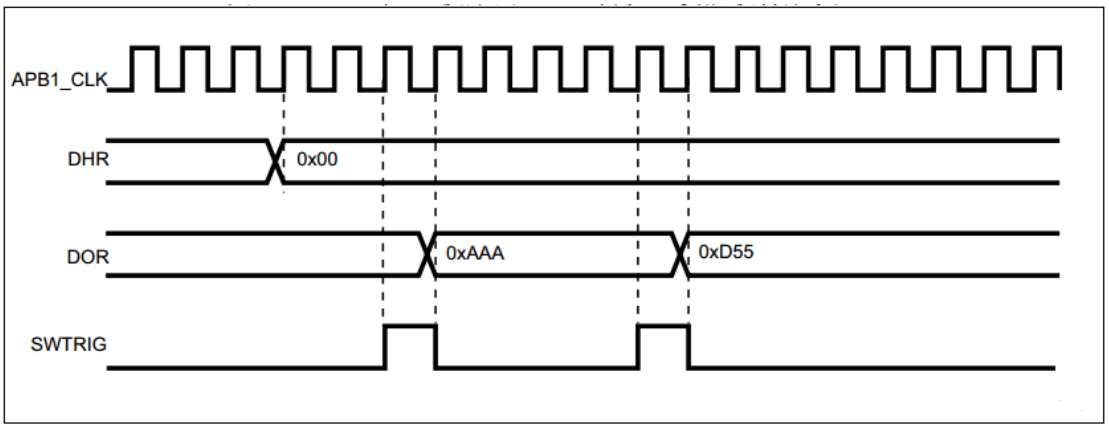


LFSR 值可以通过 DAC_CR 寄存器中的 MAMP[3:0]位来部分或完全屏蔽，在不发生溢出的情况下，该值将与 DAC_DHRx 的内容相加，然后存储到 DAC_DOR 寄存器中。

如果 LFSR 为 0x0000，将向其注入“1”（防锁定机制）。

可以通过复位 WAVE[1:0]位来将 LFSR 波形产生功能关闭。

图 14-5LFSR 产生波形的 DAC 转换（使能软件触发）



注：要生成三角波，必须通过将 DAC_CR 寄存器中的 TEN 位置 1 来使能 DAC 触发。

14.7 生成三角波

可以在直流电流或慢变信号上叠加一个小幅三角波。将 WAVE[1:0]置为“10”即可选择

DAC 生成三角波。振幅通过 DAC_CR 寄存器中的 MAMP[3:0]位进行配置。每次发生触发事件后，经过三个 APB 时钟周期，内部三角波计数器将会递增。在不发生溢出的情况下，该计数器的值将与 DAC_DHRx 寄存器内容相加，所得总和将存储到 DAC_DOR 寄存器中。只要小于 MAMP[3:0]位定义的最大振幅，三角波计数器就会一直递增。一旦达到配置的振幅，计数器将递减至零，然后再递增，以此类推。

可以通过复位 WAVE[1:0]位来将三角波产生功能关闭。

图 14-6 生成 DAC 三角波

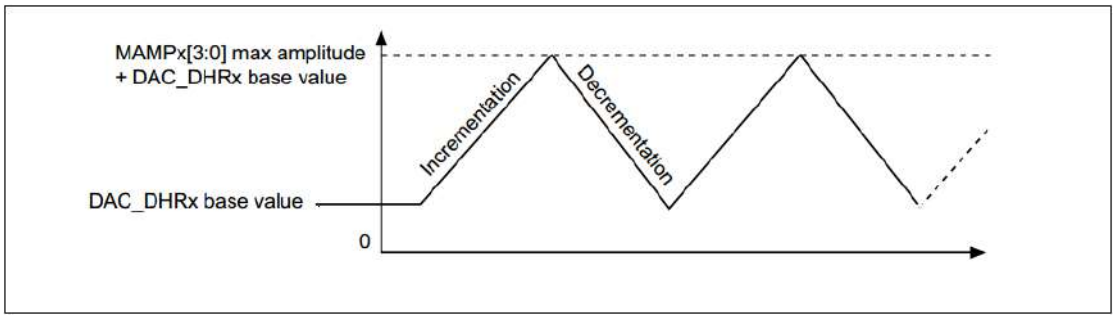
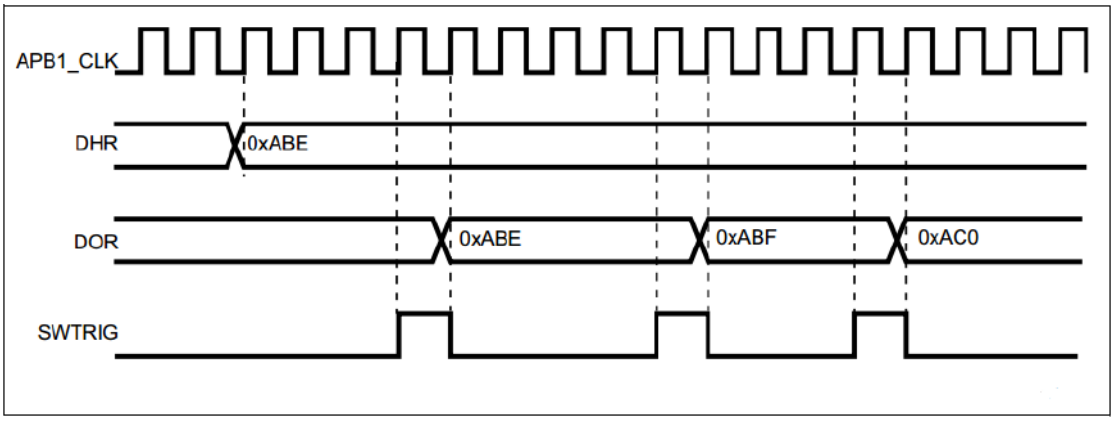


图 14-7 生成三角波波形的 DAC 转换（使能软件触发）



注： 要生成三角波，必须通过将 DAC_CR 寄存器中的 TEN 位置 1 来使能 DAC 触发。
MAMP[3:0] 位必须在使能 DAC 之前进行配置，否则将无法更改。

14.8 DMA 请求

每个 DAC 通道都具有 DMA 功能。两个 DMA 通道用于处理 DAC 通道的 DMA 请求。

当 DMAEN 位置 1 时，如果发生外部触发（而不是软件触发），则将产生 DACDMA 请求。DAC_DHRx 寄存器的值随后转移到 DAC_DOR 寄存器。

在双通道模式下，如果两个 DMAEN 位均置 1，则将产生两个 DMA 请求。如果只需要一个 DMA 请求，用户应仅将相应 DMAEN 位置 1。这样，应用程序可以在双通道模式下通过一个 DMA 请求和一个特定 DMA 通道来管理两个 DAC 通道。

DMA 下溢

DACDMA 请求没有缓冲队列。这样，如果第二个外部触发到达时尚未收到第一个外部触发的应答，将不会发出新的请求，并且 DAC_SR 寄存器中的 DAM 通道下溢标志 DMAUDR 将置 1，以报告这一错误状况。DMA 数据传输随即禁止，并且不再处理其他 DMA 请求。

DAC 通道仍将继续转换旧有数据。

软件应通过写入“1”来将 DMAUDR 标志清零，将所用 DMA 数据流的 DMAEN 位清零，并重新初始化 DMA 和 DAC 通道，以便正确地重新开始 DMA 传输。软件应修改 DAC 触发转换频率或减轻 DMA 工作负载，以避免再次发生 DMA 下溢。最后，可通过使能 DMA 数据传输和转换触发来继续完成 DAC 转换。

对于各 DAC 通道，如果使能 DAC_CR 寄存器中相应的 DMAUDRIE 位，还将产生中断。

14.9 DAC 寄存器

外设寄存器必须按字（32 位）进行访问。

14.9.1 DAC 控制寄存器(DAC_CR)

偏移地址：0x00

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	DMAUDRIE1	DMAEN1	MAMP1[3:0]				WAVE1[1:0]		TSEL1[2:0]			TEN1	BOFF1	EN1
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位 31:14	保留，必须保持复位值。
位 13	**DMAUDRIE1**： DAC 通道 DMA 下溢中断使能 (DAC channel DMA Underrun Interrupt enable) 此位由软件置 1 和清零。 0：禁止 DAC 通道 DMA 下溢中断 1：使能 DAC 通道 DMA 下溢中断
位 12	**DMAEN1**： DAC 通道 DMA 使能 (DAC channel DMA enable) 此位由软件置 1 和清零。 0：禁止 DAC 通道 DMA 模式 1：使能 DAC 通道 DMA 模式

位 11:8	**MAMP1[3:0]：** DAC 通道掩码/振幅选择器 (DAC channel mask/amplitude selector) 这些位由软件写入，用于在生成噪声波模式下选择掩码，或者在生成三角波模式下选择振幅。 0000：不屏蔽 LFSR 的位 0/三角波振幅等于 1 0001：不屏蔽 LFSR 的位 [1:0]/三角波振幅等于 3 0010：不屏蔽 LFSR 的位 [2:0]/三角波振幅等于 7 0011：不屏蔽 LFSR 的位 [3:0]/三角波振幅等于 15 0100：不屏蔽 LFSR 的位 [4:0]/三角波振幅等于 31 0101：不屏蔽 LFSR 的位 [5:0]/三角波振幅等于 63 0110：不屏蔽 LFSR 的位 [6:0]/三角波振幅等于 127 0111：不屏蔽 LFSR 的位 [7:0]/三角波振幅等于 255 1000：不屏蔽 LFSR 的位 [8:0]/三角波振幅等于 511 1001：不屏蔽 LFSR 的位 [9:0]/三角波振幅等于 1023 1010：不屏蔽 LFSR 的位 [10:0]/三角波振幅等于 2047 ≥ 1011：不屏蔽 LFSR 的位 [11:0]/三角波振幅等于 4095
位 7:6	**WAVE1[1:0]：** DAC 通道噪声/三角波生成使能 (DAC channel noise/triangle wave generation enable) 这些位由软件置 1 和清零。 00：禁止生成波 01：使能生成噪声波 1x：使能生成三角波 注： 只在位 TEN = 1（使能 DAC 通道触发）时使用。
位 5:3	**TSEL[2:0]1：** DAC 通道触发器选择 (DAC channel trigger selection) 这些位用于选择 DAC 通道的外部触发事件。 000：定时器 1 TRGO 事件 001：定时器 2 TRGO 事件 010：定时器 3 TRGO 事件 011：LP 定时器 1 TRGO 事件 100：LP 定时器 2 TRGO 事件 101：LP 定时器 3 TRGO 事件 110： EXTI 线 9 111：软件触发 注： 只在位 TEN = 1（使能 DAC 通道触发）时使用。
位 2	**TEN1：** DAC 通道触发使能 (DAC channel trigger enable) 此位由软件置 1 和清零，以使能/禁止 DAC 通道触发。 0：禁止 DAC 通道触发，写入 DAC_DHRx 寄存器的数据在一个 APB1 时钟周期之后转移到 DAC_DOR 寄存器 1：使能 DAC 通道触发， DAC_DHR 寄存器的数据在三个 APB1 时钟周期之后转移到 DAC_DOR 寄存器 注： 如果选择软件触发， DAC_DHR 寄存器的内容只需一个 APB1 时钟周期即可转移到 DAC_DOR 寄存器。
位 1	**BOFF1：** DAC 通道输出缓冲器禁止 (DAC channel output buffer disable) 此位由软件置 1 和清零，以使能/禁止 DAC 通道输出缓冲器。 0：使能 DAC 通道输出缓冲器 1：禁止 DAC 通道输出缓冲器
位 0	**EN1：** DAC 通道使能 (DAC channel enable) 此位由软件置 1 和清零，以使能/禁止 DAC 通道。 0：禁止 DAC 通道 1：使能 DAC 通道

14.9.2 DAC 软件触发寄存器 (DAC_SWTRIGR)

偏移地址：0x04

复位值：0x0000 0000

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

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWTRIG1
															w
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位 31:1	保留，必须保持复位值。
位 0	**SWTRIG1**： DAC 通道软件触发 (DAC channel software trigger) 此位由软件置 1 和清零，以使能/禁止软件触发。 0：禁止软件触发 1：使能软件触发 注： 一旦 DAC_DHR 寄存器值加载到 DAC_DOR 寄存器中，该位即会由硬件清零（一个 APB1 时钟周期之后）。

14.9.3 DAC 通道 12 右对齐数据保持寄存器 (DAC_DHR12R1)

偏移地址：0x08

复位值：0x0000 0000

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

位 31:12	保留，必须保持复位值。
位 11:0	**DACC1DHR[11:0]**： DAC 通道 12 位右对齐数据 (DAC channel 12-bit right-aligned data) 这些位由软件写入，用于为 DAC 通道指定 12 位数据。

14.9.4 DAC 通道 12 左对齐数据保持寄存器 (DAC_DHR12L1)

偏移地址：0x0C

复位值：0x0000 0000

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

位 31:16	保留，必须保持复位值。
位 15:4	**DACC1DHR[11:0]**： DAC 通道 12 位左对齐数据 (DAC channel 12-bit right-aligned data) 这些位由软件写入，用于为 DAC 通道指定 12 位数据。
位 3:0	保留，必须保持复位值。

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

偏移地址：0x10

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

位 31:8	保留，必须保持复位值。
位 7:0	**DACC1DHR[7:0]**： DAC 通道 8 位右对齐数据 (DAC channel 8-bit right-aligned data) 这些位由软件写入，用于为 DAC 通道指定 8 位数据。

14.9.6 DAC1 12 位右对齐数据保持寄存器 (DAC_DHR12RD)

偏移地址：0x20

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

15

位 31:12	保留，必须保持复位值。
位 11:0	**DACCDHR[11:0]**： DAC 通道 12 位右对齐数据 (DAC channel 12-bit right-aligned data) 这些位由软件写入，用于为 DAC 通道指定 12 位数据。

14.9.7 DAC1 12 位左对齐数据保持寄存器 (DAC_DHR12LD)

偏移地址：0x24

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

位 31:16	保留，必须保持复位值。
位 15:4	**DACC1DHR[11:0]**： DAC 通道 12 位左对齐数据 (DAC channel 12-bit right-aligned data) 这些位由软件写入，用于为 DAC 通道指定 12 位数据。
位 3:0	保留，必须保持复位值。

14.9.8 DAC1 8 位右对齐数据保持寄存器 (DAC_DHR8RD)

偏移地址：0x28

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

位 31:8	保留，必须保持复位值。
位 7:0	**DACCDHR[7:0]**： DAC 通道 8 位右对齐数据 (DAC channel 8-bit right-aligned data) 这些位由软件写入，用于为 DAC 通道指定 8 位数据。

14.9.9 DAC 通道 数据输出寄存器 (DAC_DOR1)

偏移地址： 0x2C

复位值： 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

位 31:12	保留，必须保持复位值。
位 11:0	**DACC1DOR[11:0]**： DAC 通道数据输出 (DAC channel data output) 这些位为只读，其中包含 DAC 通道的数据输出。

14.9.10 DAC 状态寄存器 (DAC_SR)

偏移地址： 0x34

复位值： 0x0000 0000

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

Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	DMAUDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
		rc_w													
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位 31:14	保留，必须保持复位值。
位 13	**DMAUDR1**： DAC 通道 DMA 下溢标志 (DAC channel DMA underrun flag) 此位由硬件置 1，由软件清零（写入 1）。 0： DAC 通道未发生 DMA 下溢错误状况 1： DAC 通道发生 DMA 下溢错误状况（当前所选触发源以高于 DMA 服务能力的频率驱动 DAC 通道转换）
位 12:0	保留，必须保持复位值。

15 液晶显示控制器 (LCD)

15.1 简介

LCD 控制器是一款适用于单色无源液晶显示器 (LCD) 的数字控制器/驱动器，最多具有 8 个公用端子和 52(a) 个区段端子，用以驱动 208 (4x52) 或 384 (8x48) 个 LCD 图像元素（像素）。端子的确切数量取决于数据手册中所述的器件引脚。

LCD 由若干区段（像素或完整符号）组成，这些区段均可点亮或熄灭。每个区段都包含一层在两根电极之间对齐的液晶分子。当向液晶施加高于阈值电压的电压时，相应的区段可见。

区段电压必须为交流，以避免液晶中出现电泳效应（这将影响显示效果）。之后，必须在区段两端生成波形以避免出现直流 (DC)。

15.2 LCD 主要特性

- 高度灵活的帧速率控制。
- 支持静态、 1/2、 1/3、 1/4 和 1/8 占空比。
- 支持静态、 1/2、 1/3 和 1/4 偏置。
- 双缓冲存储器，允许通过应用固件随时更新 LCD_RAM 寄存器中的数据，而不影响所显示数据的完整性。
  - 多达 16 × 32 位寄存器的 LCD 数据 RAM，其中包含像素信息（激活/未激活）。
- 可通过软件选择的 LCD 输出电压（对比度）： VLCDmin 到 VLCDmax。
- 无需外部模拟元件：
  - 内置的升压转换器可生成高于 VDD 的内部 VLCD 电压
  - 可通过软件在外部和内部 VLCD 电压源之间进行选择。如果选择外部电压源，将禁止内部升压电路以降低功耗
  - 内置的电阻网络可生成中间 VLCD 电压
  - 可通过软件配置电阻网络的结构以调整功耗，从而匹配 LCD 面板所需的电容电荷
- 可以通过两种不同的方法调整对比度：
  - 当使用内部升压转换器时，软件可以在 VLCDmin 和 VLCDmax 之间调整 VLCD。
  - 可编程的帧间死区（最多 8 个相位周期）。
- 完全支持低功耗模式： LCD 控制器可在睡眠、低功耗运行、低功耗睡眠和停止模式下进行显示，也可以完全禁止以降低功耗。
- 内置反相功能以降低功耗和 EMI（电磁干扰）。
- 帧起始中断，在更新 LCD 数据 RAM 时同步软件。
- 闪烁功能：
  - 可以设置 1、 2、 3、 4、 8 个或所有像素以可配置的频率闪烁
  - 可通过软件将闪烁频率调整到大约 0.5 Hz、 1 Hz、 2 Hz 或 4 Hz。
- 使用的 LCD 区段和公共引脚应配置为 GPIO 复用功能，未使用的区段和公共引脚可用于通用 I/O 或其他外设复用功能。
- VLCD 轨去耦功能

注： 当 LCD 依赖于内部升压转换器时， VLCD 引脚应通过一个电容连接到 VSS。电容典型值为 1 μF（有关详细信息，请参见产品数据手册中的 CEXT 值）。

注： 如果不使用 LCD 外设，则 VLCD 引脚应连接到 VDDA。

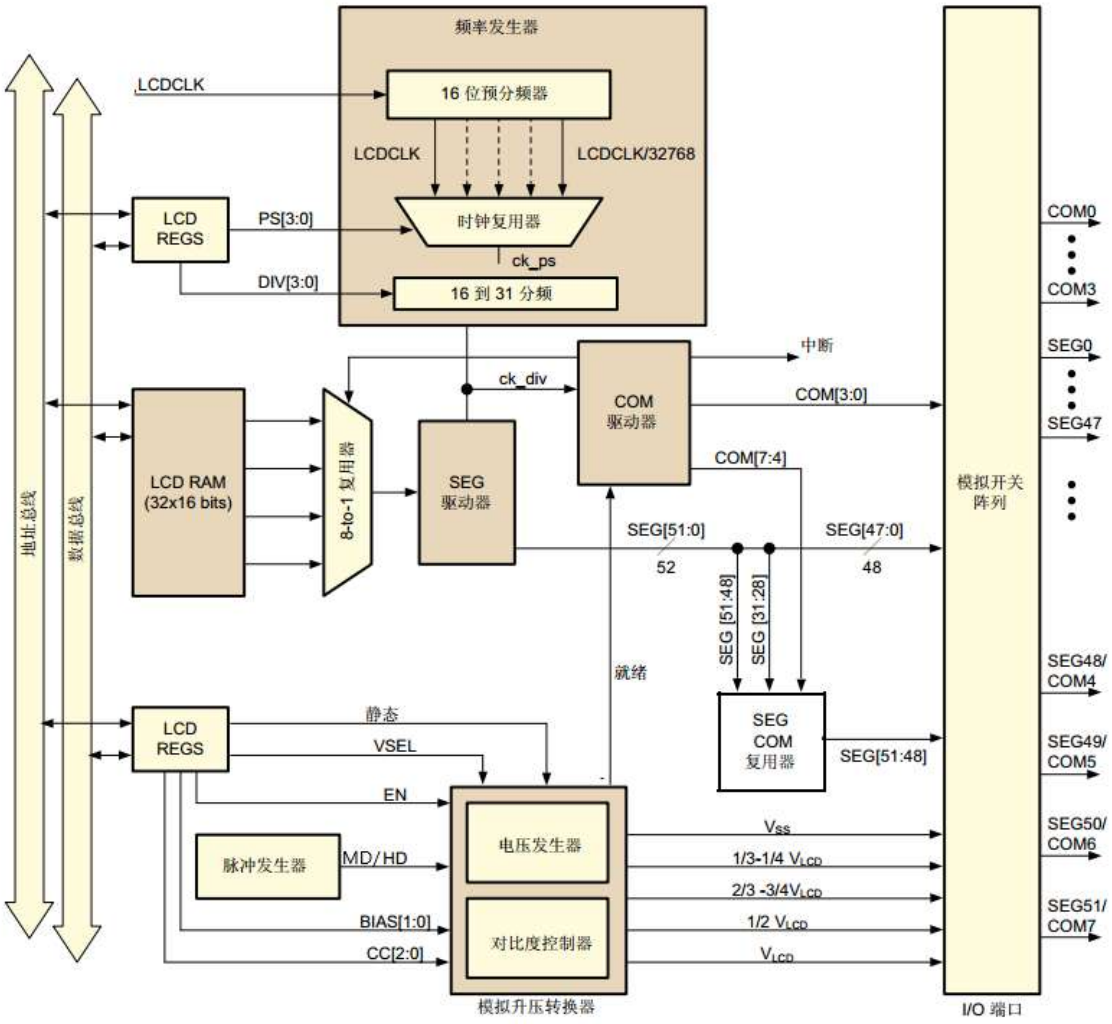
15.3 LCD 实现

15.4 LCD 功能说明

15.4.1 概述

LCD 控制器具有五个主要模块：

图 15-1 **LCD** 控制器框图



注： LCDCLK 与 RTCCLK 相同。请参考本手册的 RCC 部分中的 RTC/LCD 时钟说明。

凭借频率发生器，可以通过 LCD 输入时钟频率（LCDCLK，可从 32 kHz 直至 1 MHz）获得各种 LCD 帧速率。

可以使用 3 种不同的时钟源来提供 LCD 时钟 (LCDCLK/RTCCLK)：

- 32 kHz 低速外部 RC (LSE)
- 32 kHz 低速内部 RC (LSI)
- 2、 4、 8 或 16 分频的高速外部 RC (HSE) 以获得 1 MHz 时钟

15.4.2 频率发生器

此时钟源必须稳定以获得精确的 LCD 时序，并因此最大限度地减少 LCD 区段间的 DC 电压偏移。输入时钟 LCDCLK 可被从 1 到 2^{15}x 31 的任意值分频（请参见 *LCD 帧控制寄存器 (LCD_FCR)*）。频率发生器包含一个预分频器（16 位纹波计数器）和一个 16 到 31 时钟分频器。LCD_FCR 寄存器中的 PS[3:0] 位选择被 2$^{PS[3:0]}$ 分频的 LCDCLK。如果需要更精细的分辨率， LCD_FCR 寄存器中的 DIV[3:0] 位可用于进一步对时钟进行 16 到 31 分频。通过这种方式可以粗调频率，然后通过计数器线性调整时钟以进行微调。频率发生器模块的输出是 fck_div，它构成整个 LCD 控制器的时基。 ck_div 频率与 LCD 相位频率相等，而不等于帧频率（它们仅在静态占空比的情况下相等）。通过将 fck_div 除以激活的公用端子数（或者将该频率与占空比相乘），可获得帧频率 (fframe)。因此，频率发生器的输入时钟频率 (fLCDCLK) 与其输出时钟频率 fck_div 之间的关系为：

$$x = \frac{f_{LCDCLK}}{2^{PS}X(16+DIV)}$$

$f_{CLK}=f_{ckdiv}$ x duty

这使得频率发生器非常灵活。下表中给出了帧速率计算示例。

表 15-1 帧速率计算示例

LCDCLK	PS[3:0]	DIV[3:0]	比值	占空比	fframe
32.768 kHz	3	1	136	1/8	30.12 Hz
32.768 kHz	4	1	272	1/4	30.12 Hz
32.768 kHz	4	6	352	1/3	31.03 Hz
32.768 kHz	5	1	544	1/2	30.12 Hz
32.768 kHz	6	1	1088	静态	30.12 Hz
32.768 kHz	1	4	40	1/8	102.40 Hz
32.768 kHz	2	4	80	1/4	102.40 Hz
32.768 kHz	2	11	108	1/3	101.14 Hz
32.768 kHz	3	4	160	1/2	102.40 Hz
32.768 kHz	4	4	320	静态	102.40 Hz
1.00 MHz	6	3	1216	1/8	102.80 Hz
1.00 MHz	7	3	2432	1/4	102.80 Hz
1.00 MHz	7	10	3328	1/3	100.16 Hz
1.00 MHz	8	3	4864	1/2	102.80 Hz
1.00 MHz	9	3	9728	静态	102.80 Hz

选择的帧频率必须在大约 30 Hz 到 100 Hz 的范围内，并在功耗和可接受的刷新速率之间进行平衡。此外，可通过一个专用闪烁预分频器选择闪烁频率。此频率定义为：

fBLINK = fck_div/2(BLINKF + 3)，

其中， BLINKF[2:0] = 0、 1、 2、 ... 、 7 可实现的闪烁频率为 0.5 Hz、 1 Hz、 2 Hz 或 4 Hz。

15.4.3 公用驱动器

公用信号由公用驱动器模块产生。

COM 信号偏置

各个 COM 信号的波形相同，但相位不同。这些信号仅在周期的相应相位才具有最大振幅 VLCD 或 VSS，而在其它相位的信号幅值为：

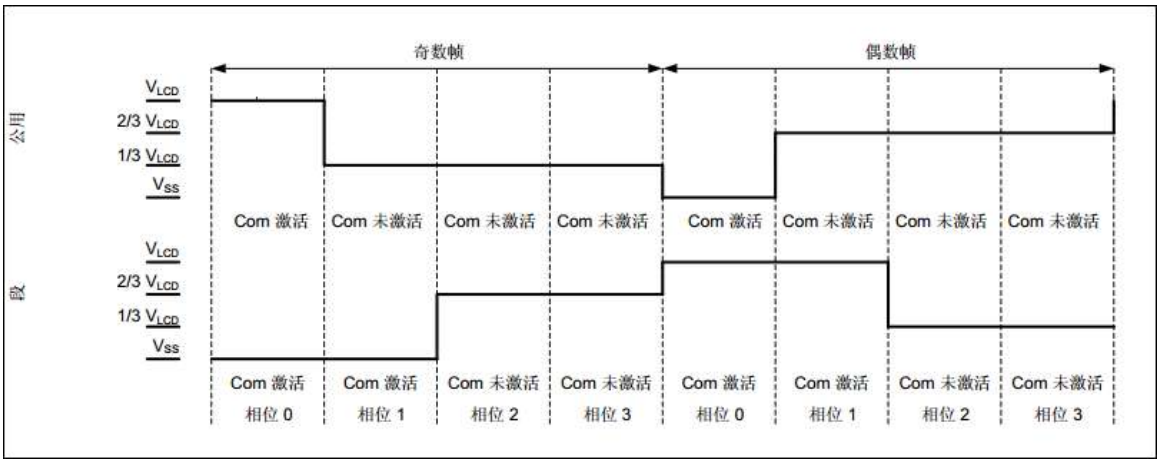
- 1/4 VLCD 或 3/4 VLCD（1/4 偏置时）
- 1/3 VLCD 或 2/3 VLCD（1/3 偏置时）
- 以及 1/2 VLCD（1/2 偏置时）。

可以通过 LCD_CR 寄存器中的 BIAS 位在 1/2、 1/3 和 1/4 偏置模式之间进行选择。

当某个像素对应的公用线和区段线在相同相位期间激活时，该像素激活，这意味着在此相位期间公用线

和区段线之间的电压差最大。对公用信号进行反相以减少 EMI。如图 15-2 **1/3** 偏置， **1/4** 占空比图 15-2 所示，通过反相，每个奇数周期结束时的平均电压为 1/2 VLCD。

图 15-2 1/3 偏置， 1/4 占空比



在 1/2 偏置 (BIAS = 01) 的情况下，VLCD 引脚在奇数帧和偶数帧的节点 b 上产生的中间电压等于 1/2 VLCD。

COM 信号占空比

根据 LCD_CR 寄存器中的 DUTY[2:0] 位，使用以下占空比产生 COM 信号：静态占空比、 1/2 占空比、 1/3 占空比、1/4 占空比或 1/8 占空比。

奇数帧的相位 *n* 期间 COM[*n*] *n*[0 到 7] 激活，即 COM 引脚被驱动到 VLCD。偶数帧的相位 *n* 期间 COM 引脚被驱动到 VSS。

在 1/3 或 1/4 偏置的情况下：

- 在 n 以外的相位期间 COM[*n*] 未激活，因此 COM 引脚被驱动到 1/3 (1/4) VLCD（奇数帧期间）和 2/3 (3/4) VLCD（偶数帧期间）

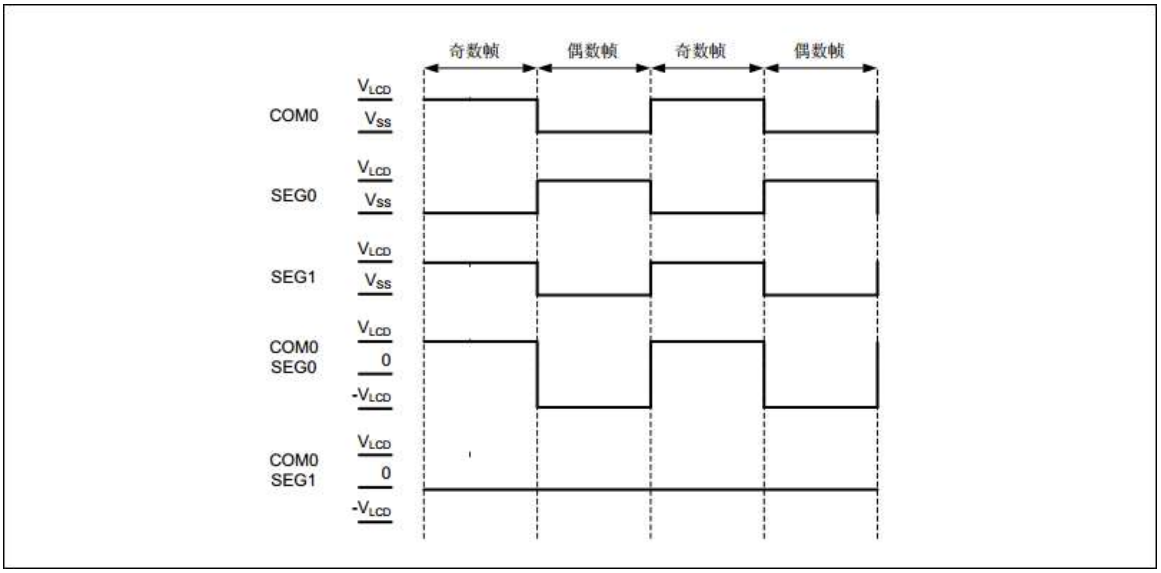
在 1/2 偏置的情况下：

- 如果在 n 以外的相位期间 COM[*n*] 未激活， COM 引脚始终驱动到 1/2 VLCD（奇数帧和偶数帧）。

当选择静态占空比时，区段线不多路复用，这意味着每个区段输出对应于一个像素。通过这种方式，最多只能驱动 51 个像素。 COM[7:1] 未使用，并被驱动到 VSS，与此同时，COM[0] 始终激活。

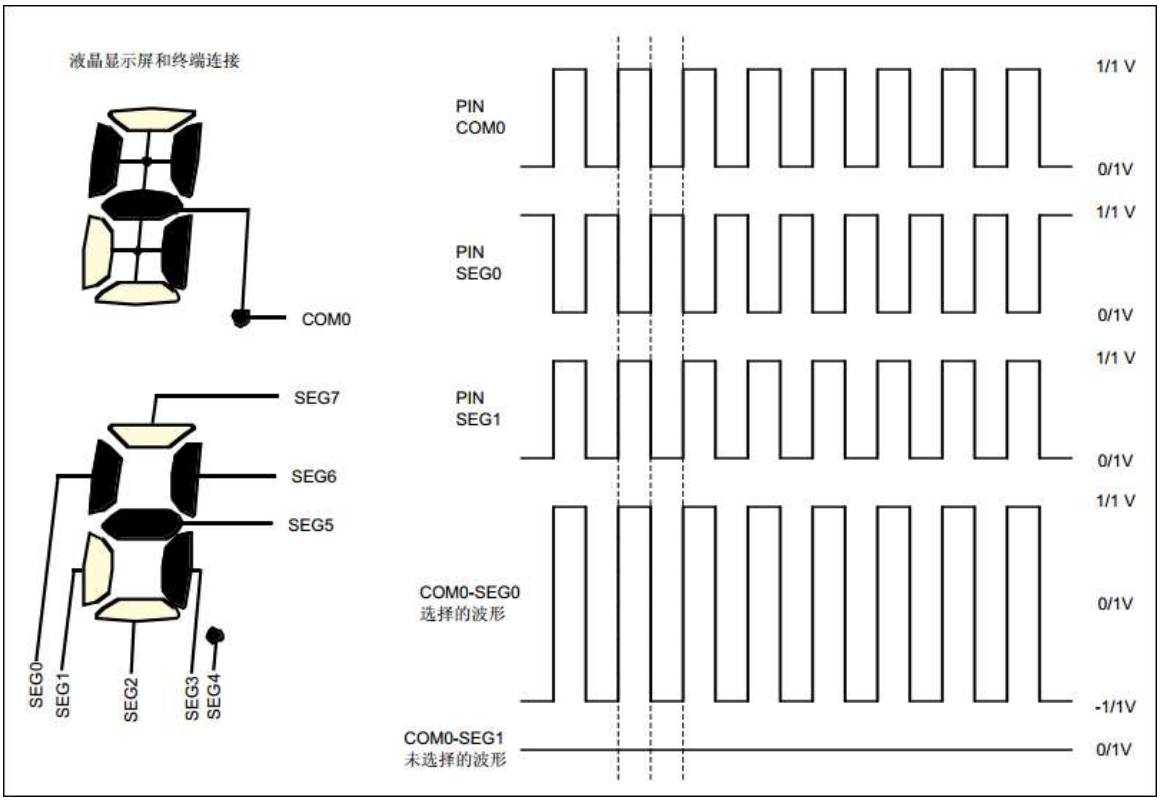
当 LCD_CR 寄存器中的 LCDEN 位复位时，所有公用线均下拉到 VSS，并且 LCD_RS 寄存器中的 ENS 标志变为 0。静态占空比意味着 COM[0] 始终激活，并且只有两个电压电平用于区段和公用线： VLCD 和 VSS。如果像素对应的 SEG 线的电压与 COM 线的电压相反，则像素激活；如果电压相等，则像素未激活。通过这种方式， LCD 可获得最大对比度（请参见图 15-3、 图 15-4）。在图 15-3 中，像素 0 激活而像素 1 未激活。

图 15-3 静态占空比用例 1



每个帧中只有一个相位，因此 fframe 等于 fLCD。如果选择 1/4 占空比，一个帧中有四个相位，其中，COM[0] 在相位 0 期间激活， COM[1] 在相位 1 期间激活， COM[2] 在相位 2 期间激活， COM[3] 在相位 3 期间激活。

图 15-4 静态占空比用例 2

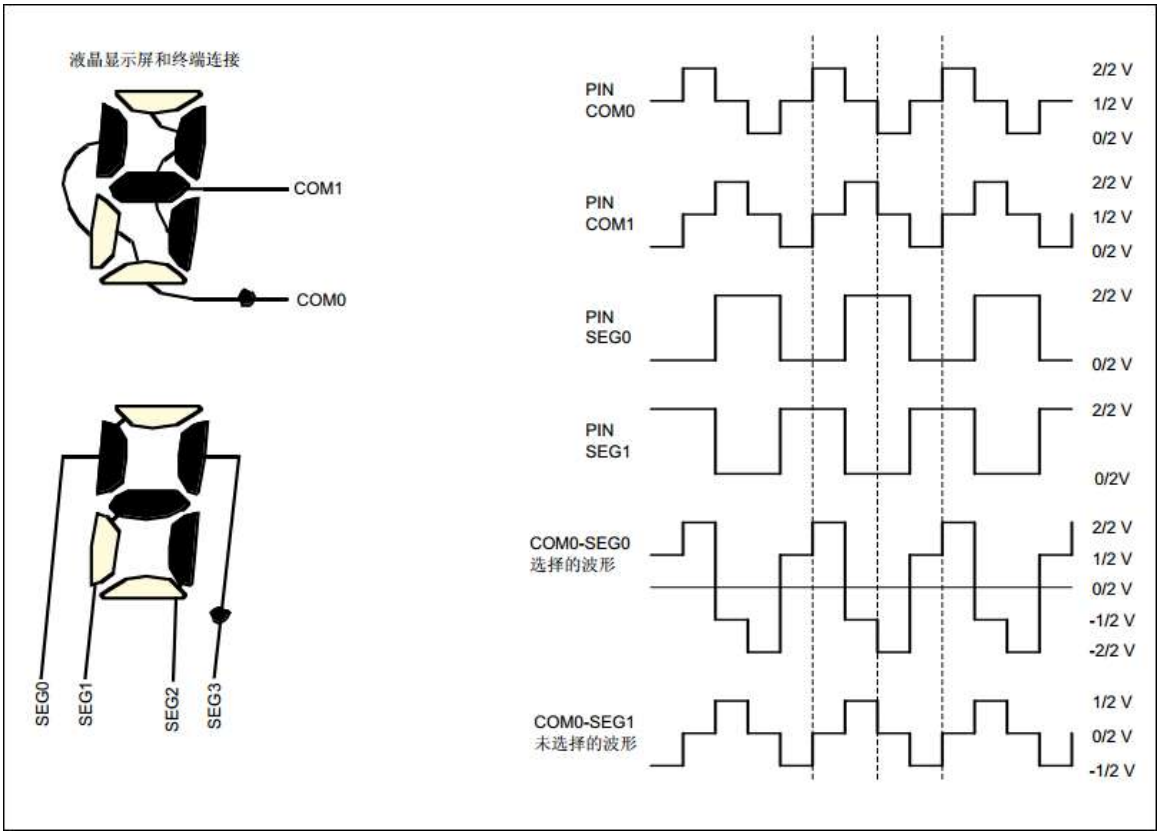


在此模式下，区段端子多路复用并且每个端子控制四个像素。仅当像素对应的 SEG 和 COM 线在相同相位期间激活时，该像素才激活。在 1/4 占空比的情况下，要禁用连接到 COM[0] 的像素 0，当 COM[0] 激活时， SEG[0] 需要在相位 0 期间处于未激活状态。要激活连接到 COM[1] 的像素 0，当 COM[1] 激活时， SEG[0] 需要在相位 1 期间处于激活状态（请参见图 15-7）。要激活连接到 COM[0] 的像素 0 到 51，当 COM[0] 激活时， SEG[0:51]需要在相位 0 期间处于激活状态。这些注意事项同样适用于其它像素。

8 选 **1** 多路复用器

当 COM[0] 激活时，公用驱动器模块也将驱动图 15-1 中所示的 8 选 1 多路复用器，以便选择前两个 RAM 寄存器存储单元的内容。当 *COM[7]* 激活时， 8 选 1 多路复用器的输出是最后两个 RAM 存储单元的内容。

图 15-5 **1/2** 占空比， **1/2** 偏置



15.4.4 区段驱动器

区段驱动器模块根据像素数据（来自各相位中由公用驱动器模块驱动的 8 选 1 多路复用器）来控制 SEG 线。

在 **1/4** 或 **1/8** 占空比的情况下当 COM[0] 激活时，与连接到 COM[0]（前两个 LCD_RAM 存储单元的内容）的像素相关的像素信息（激活/未激活）通过 8 选 1 多路复用器。

在奇数帧的相位 0 期间， SEG[*n*] 引脚 *n* [0 到 51] 被驱动到 VSS（指示当 COM[0] 激活时像素 *n* 激活）。

在偶数帧的相位 0 期间， SEG[*n*] 引脚被驱动到 VLCD。如果像素 *n* 未激活，则 SEG[*n*] 引脚被驱动到 2/3 (2/4) VLCD（奇数帧期间）或 1/3 (2/4) VLCD（偶数帧期间）（VLCD 焊盘中的当前转换）（请参见图 15-2）。

在 1/2 偏置的情况下，如果像素未激活， SEG[*n*] 引脚被驱动到 VLCD（奇数帧期间）和 VSS（偶数帧期间）。

当禁止 LCD 控制器（LCD_CR 寄存器中的 LCDEN 位清零）时， SEG 线下拉到 VSS。

图 15-6 **1/3** 占空比， **1/3** 偏置

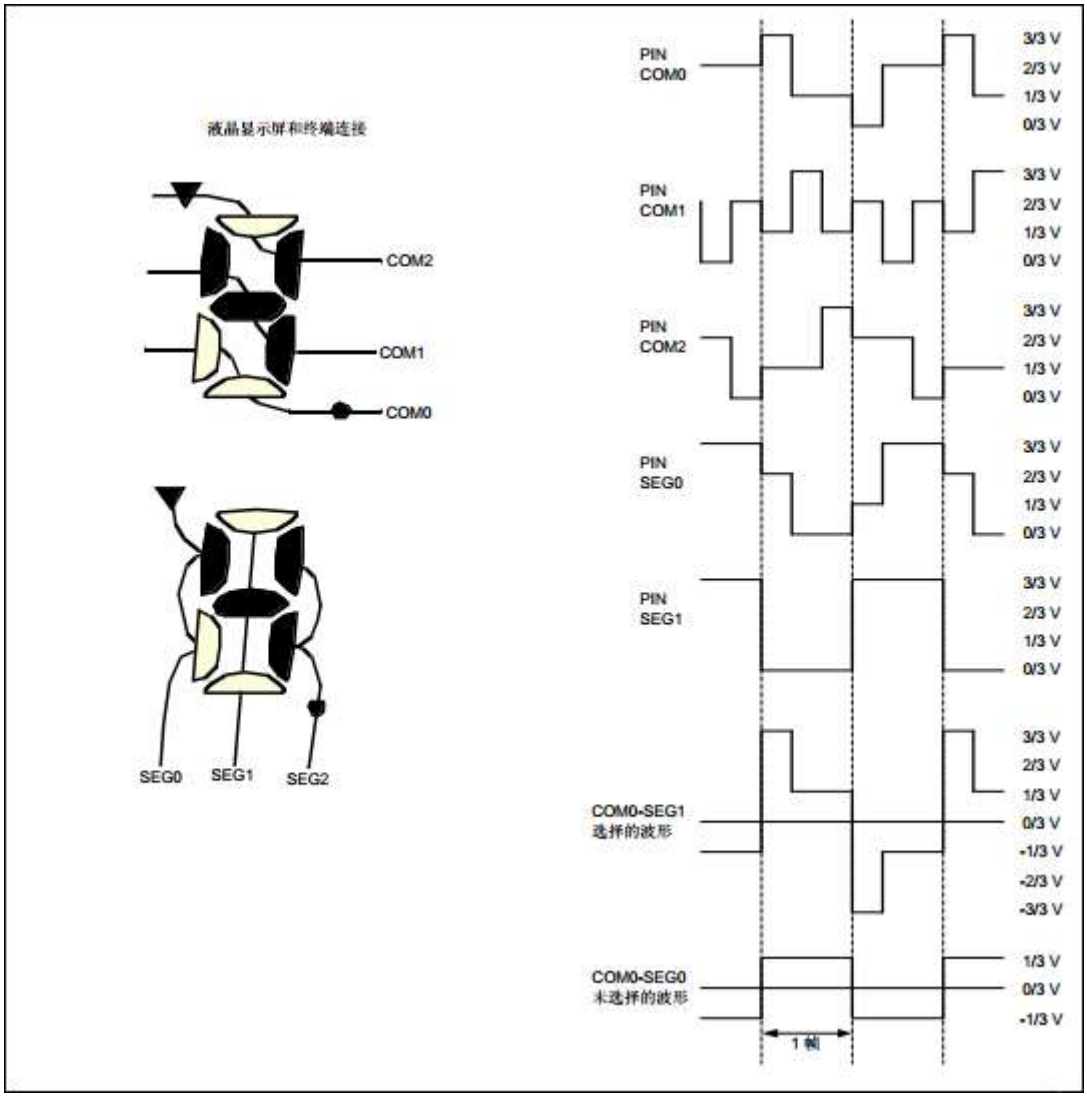


图 15-7 **1/4** 占空比，**1/3** 偏置

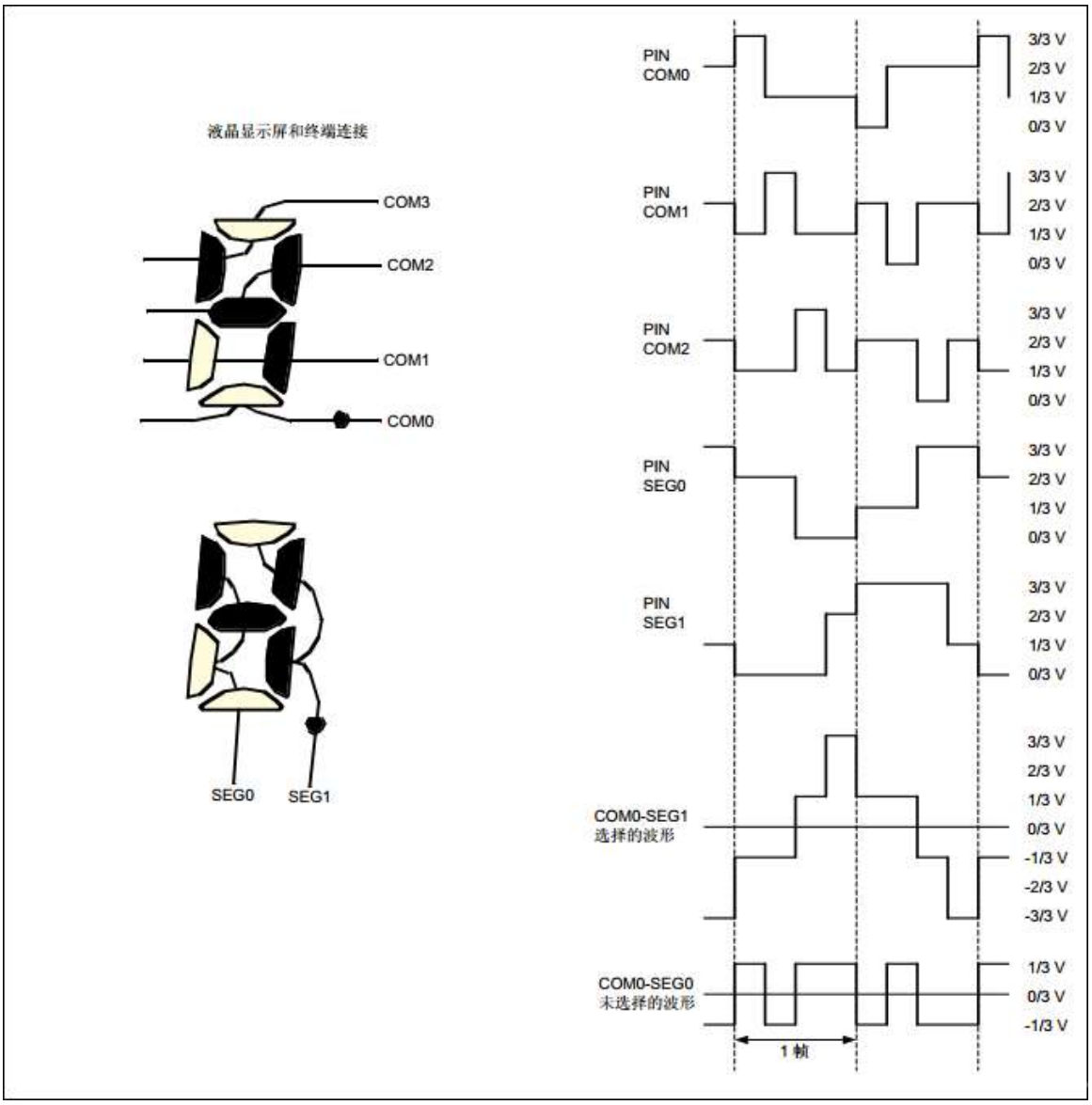
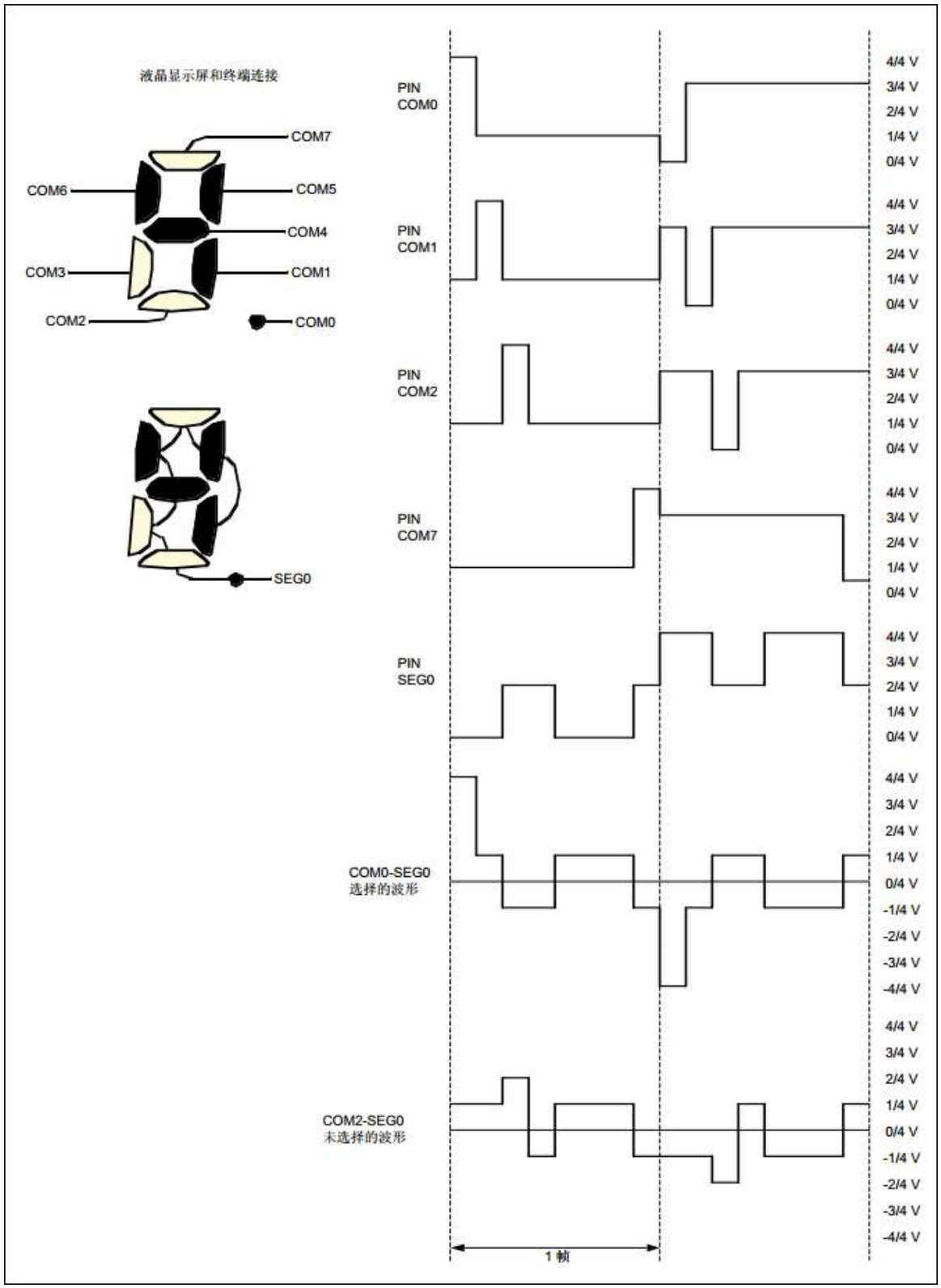


图 15-8 **1/8** 占空比，**1/4** 偏置



闪烁

区段驱动器还实现了一个可编程的闪烁功能，允许一些像素以特定频率连续地接通。可通过 LCD_FCR 寄存器中的 BLINK[1:0] 位配置闪烁模式，从而使 1、 2、 4、 8 或所有像素能够同时闪烁（请参见 *LCD 帧控制寄存器 (LCD_FCR)*）。可使用 LCD_FCR 寄存器中 BLINKF[2:0] 位从八个不同值中选择闪烁频率。

下表给出了不同闪烁频率的示例（以 ck_div 频率的函数形式）。

表 15-2 闪烁频率

BLINKF[2:0]位			Ck_div 频率（LCDCLK 频率为 32.768kHz）			
			32Hz	64	H	
0	0	0	4.0 Hz	N/A	N/A	N/A
0	0	1	2.0 Hz	4.0 Hz	N/A	N/A
0	1	0	1.0 Hz	2.0 Hz	4.0 Hz	N/A
0	1	1	0.5 Hz	1.0 Hz	2.0 Hz	4.0 Hz
1	0	0	0.25 Hz	0.5 Hz	1.0 Hz	2.0 Hz
1	0	1	N/A	0.25 Hz	0.5 Hz	1.0 Hz
1	1	0	N/A	N/A	0.25 Hz	0.5 Hz
1	1	1	N/A	N/A	N/A	0.25 Hz

15.4.5 电压发生器和对比度控制

LCD 电源

LCD 电源可能来自内部升压转换器，或来自施加在 VLCD 引脚上的外部电压。可以使用

LCD_CR 寄存器中的 VSEL 位选择内部或外部电压源。如果选择外部电压源，将禁止内部升压电路（升压转换器）以降低功耗。

当选择升压转换器作为 VLCD 源时，可以通过 LCD_FCR（请参见第 *17.7.2* 节）寄存器中的位 CC[2:0]（对比度控制）在多个值（从 VLCDmin 到 VLCDmax）中选择 VLCD 值。 VLCD 的新值在每次新帧开始时生效。

当选择外部电源作为 VLCD 源时，必须在 VLCDmin 到 VLCDmax 的范围内选择 VLCD 电压（请参见数据手册）。然后可以通过编程帧间死区来控制对比度（请参见图 15-10）。

LCD 中间电压

LCD 中间电压通过内部电阻分压器网络生成，如图 15-9 所示。

LCD 电压发生器发出介于 VSS 和 VLCD 之间的中间电压：

- 1/3 VLCD 和 2/3 VLCD（1/3 偏置时）
- 1/4 VLCD、 2/4 VLCD 和 3/4 VLCD（1/4 偏置时）
- 仅 1/2 VLCD（1/2 偏置时）。

LCD 驱动器选择

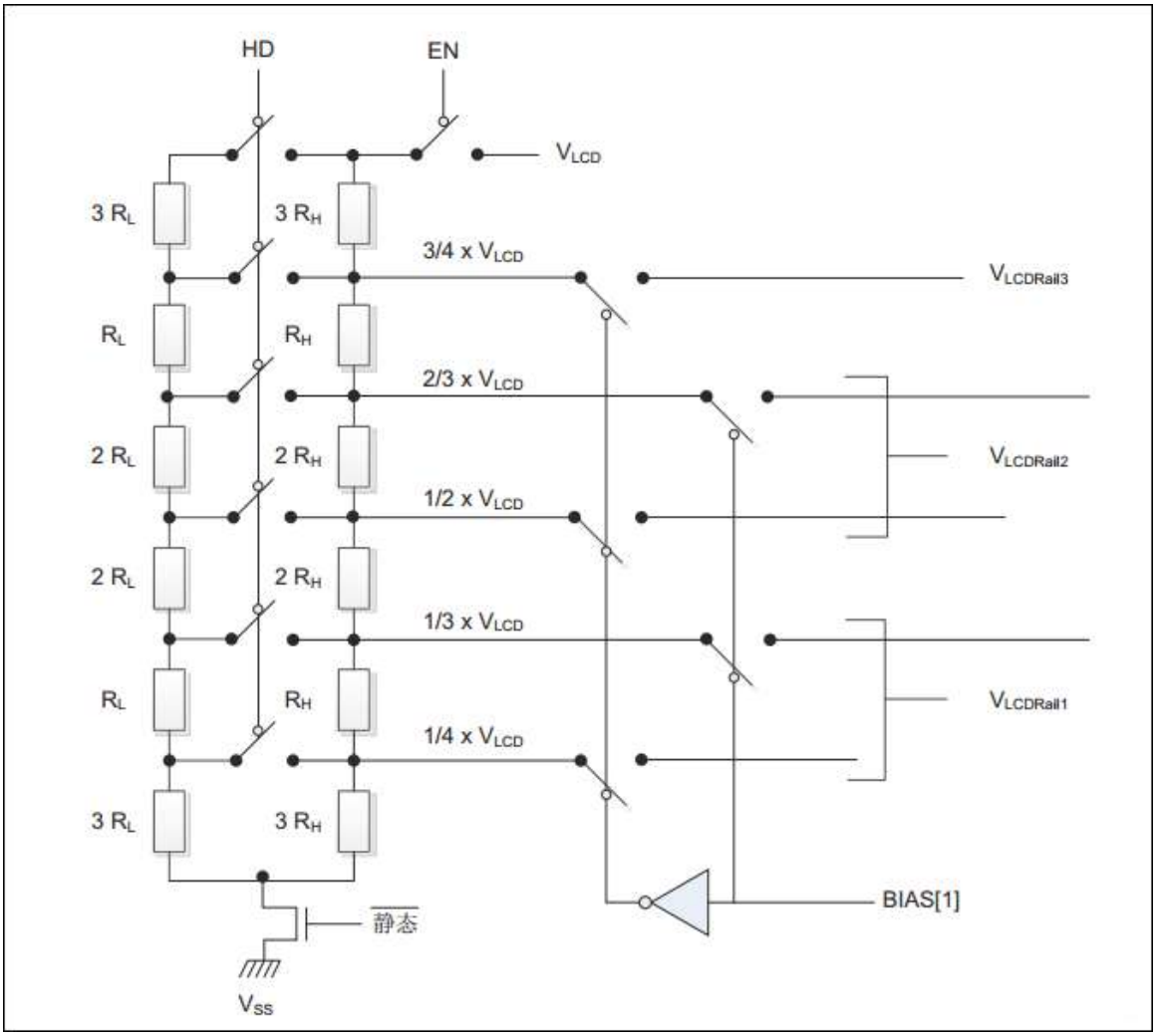
两个电阻网络（一个具有低值电阻 (RL)，一个具有高值电阻 (RH)）分别用于在转换过程中增加电流和在静态状态下降低功耗。

EN 开关遵循下述规则（请参见图 15-9）：

- 如果 LCD_CR 寄存器中的 LCDEN 位置 1， EN 开关闭合。
- 当 LCD_CR 寄存器中的 LCDEN 位清零时， EN 开关在偶数帧结束时打开，以避免中间电压不为 0（考虑整个奇数帧加偶数帧）。

当公用线和区段线的电平变化时， LCD_FCR 寄存器中的 PON[2:0]（脉冲打开持续时间）位配置通过 HD（高驱动）接通 RL 的时间（请参见图 15-9）。较短的驱动时间有助于降低功耗，但内部电阻较高的显示器可能需要更长的驱动时间才能达到令人满意的对比度。

图 15-9 **LCD** 电压控制



1. RLN 和 RHN 分别是低值电阻网络和高值电阻网络。

可通过 LCD_FCR 配置寄存器中的 HD 位随时接通 RLN 分频器。

HD 开关遵循下述规则：

- 如果 LCD_FCR 寄存器中的 HD 位和 PON[2:0] 位复位，则 HD 开关打开。
- 如果 LCD_FCR 寄存器中的 HD 位复位，并且 LCD_FCR 中的 PON[2:0] 位不为 00，则在 PON[2:0] 位中定义的脉冲数期间 HD 开关闭合。
- 如果 LCD_FCR 寄存器中的 HD 位为 1，则 HD 开关常闭。

LCDEN 位激活后， LCD_SR 寄存器中的 RDY 位置 1，以指示电压电平稳定并且 LCD 控制器可以开始工作。

外部去耦

具有 VLCD 轨去耦功能（请参见器件数据手册）的器件允许在 LCD_VLCD1、 LCD_VLCD2 和 LCD_VLCD3 上提供的 VLCD 中间电压轨上添加去耦电容，用于稳定用途（请参见图 *71*）。当施加到像素的电压为交流时，可能会观察到尖峰。在这种情况下，这些去耦电容将有助于获得稳定电压，从而提高对比度。

出于功耗考虑，此功能特别有用，因为它允许在 LCD_FCR 寄存器中选择较低的 PON[2:0] 值。要将表 15-1 中所述的 VLCD 轨连接到专用的 GPIO，配置 SYSCFG_CFGR2 寄存器的 LCD_CAPA[4:0] 位（请参见 *SYSCFG 外设模式配置寄存器 (SYSCFG_CFGR2)*）。

表 15-3 将 **VLCD** 轨连接到 **GPIO** 引脚

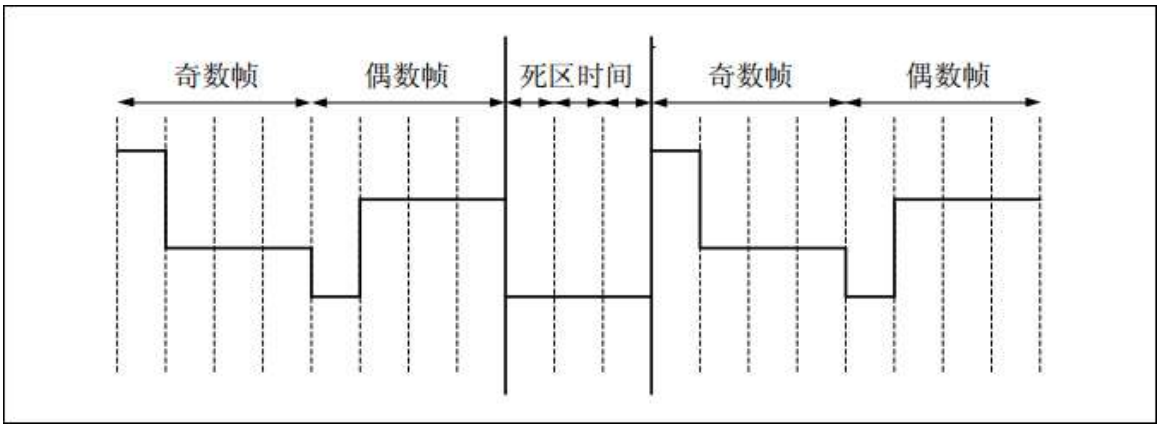
	偏置	引脚

	1/2	1/3	1/4	
VLCD 轨 1	未使用	未使用	3/4 V_{LCD}	PB0 或 PE12
VLCD 轨 2	1/2 V_{LCD}	2/3 V_{LCD}	1/2 V_{LCD}	PB2
VLCD 轨 3	未使用	1/3V_{LCD}	1/4 V_{LCD}	PB12 或 PE11

为了确保有效，必须根据 LCD 玻璃和 PCB 电容调整这些去耦电容的值。作为参考，用户可以将去耦电容值设置为 LCD 电容值的 10 倍左右。

死区除了使用 CC[2:0] 位外，还可以通过编程帧间死区来控制对比度。在死区内， COM 和 SEG 值置于 Vss。 LCD_FCR 寄存器中的 DEAD[2:0] 位可用于编程一段最长为八个相位周期的时间。此死区可降低对比度，而无需修改帧速率。

图 15-10 死区



15.4.6 双缓冲存储器

使用此双缓冲存储器， LCD 控制器可保证所显示信息连贯，而无需使用中断来控制 LCD_RAM 修改。

应用软件可以通过 APB 接口访问第一缓冲级 (LCD_RAM)。一旦它修改了 LCD_RAM，便会将 LCD_SR 寄存器中的 UDR 标志置 1。该 UDR 标志（更新显示请求）会请求将更新的信息移到第二缓冲级 (LCD_DISPLAY)。

该操作与帧同步进行（在下一帧开始时），直到更新完成， LCD_RAM 处于写保护状态并且 UDR 标志一直保持高电平状态。一旦更新完成，另一个标志（UDD - 更新显示完成）便会置 1 并产生一个中断（如果 LCD_FCR 寄存器中的 UDDIE 位置 1）。

在最坏的情况下，更新 LCD_DISPLAY 所花费的时间为一个奇数帧和一个偶数帧。

使能显示器之前 (LCDEN = 1)，不会发生更新（UDR = 1 且 UDD = 0）

15.4.7 COM 和 SEG 多路复用

输出引脚与占空比模式

输出引脚最多包含：

- SEG[51:0]
- COM[3:0]

根据占空比配置的不同， COM 和 SEG 输出引脚会具有不同的功能：

- 在静态、 1/2、 1/3、 1/4 占空比模式下，有多达 52 个 SEG 引脚，并分别有 1、 2、 3 和 4 个 COM 引脚
- 在 1/8 占空比模式下 (DUTY[2:0] = 100)，第 5 类器件和第 3 类器件分别在 SEG[51:48] 和 SEG[31:28] 引脚上提供 COM[7:4] 输出。这样可减少可用区段数。

小型封装的重映射功能

此外，可以通过将 LCD_CR 寄存器中 MUX_SEG 位置 1 重新映射 4 个区段。使用外部引脚较少的

小型器件时，该功能特别有用。当 MUX_SEG 位置 1 时，输出引脚 SEG [51:48] 与 SEG [31:28] 具有相同功能。

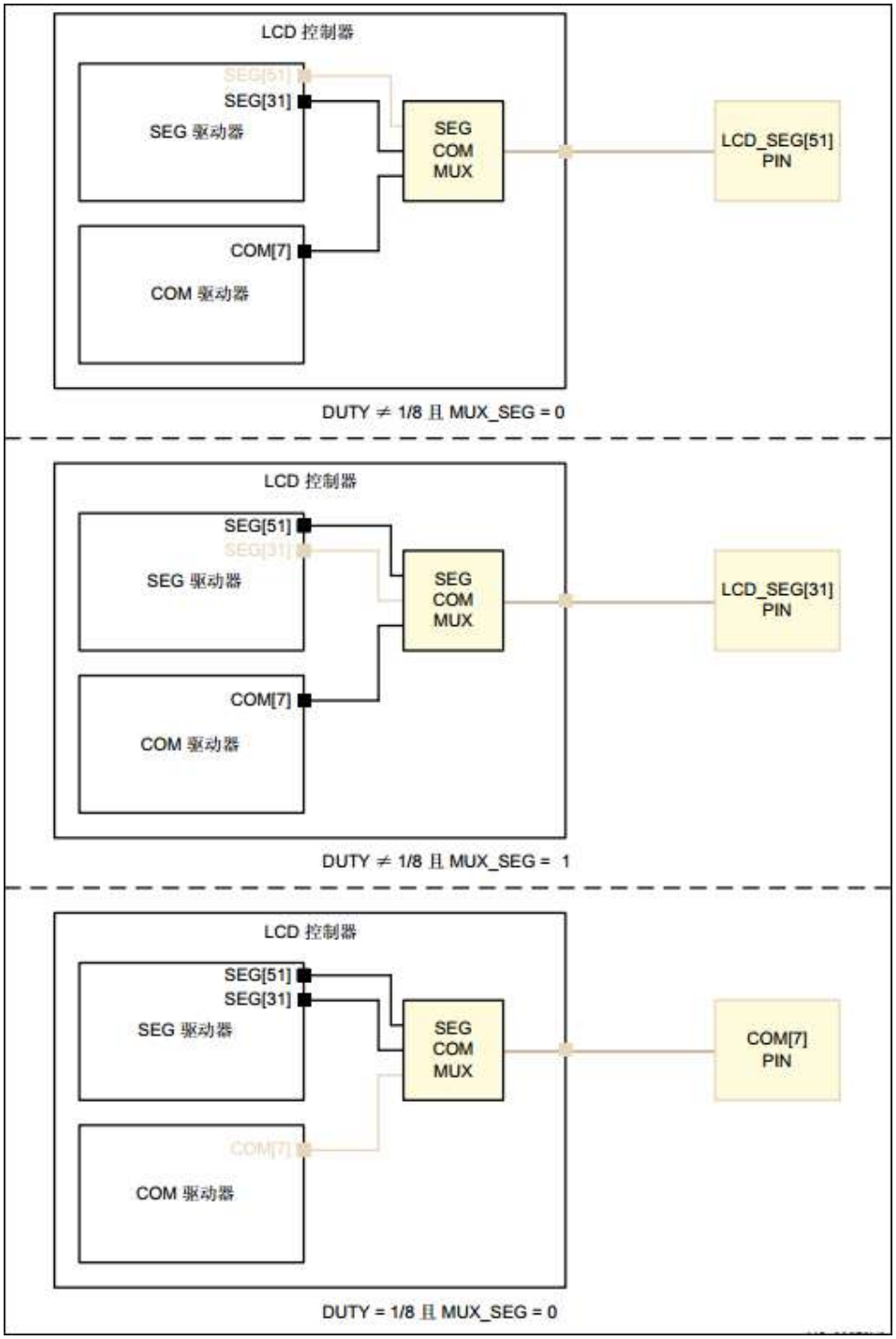
COM 和 **SEG** 功能与占空比和重映射的关系汇总

表 15-4 给出了多路复用 COM 和 SEG 功能的所有可能方式。表 15-4 给出了与外部引脚的信号连接示例。

表 15-4 重映射功能

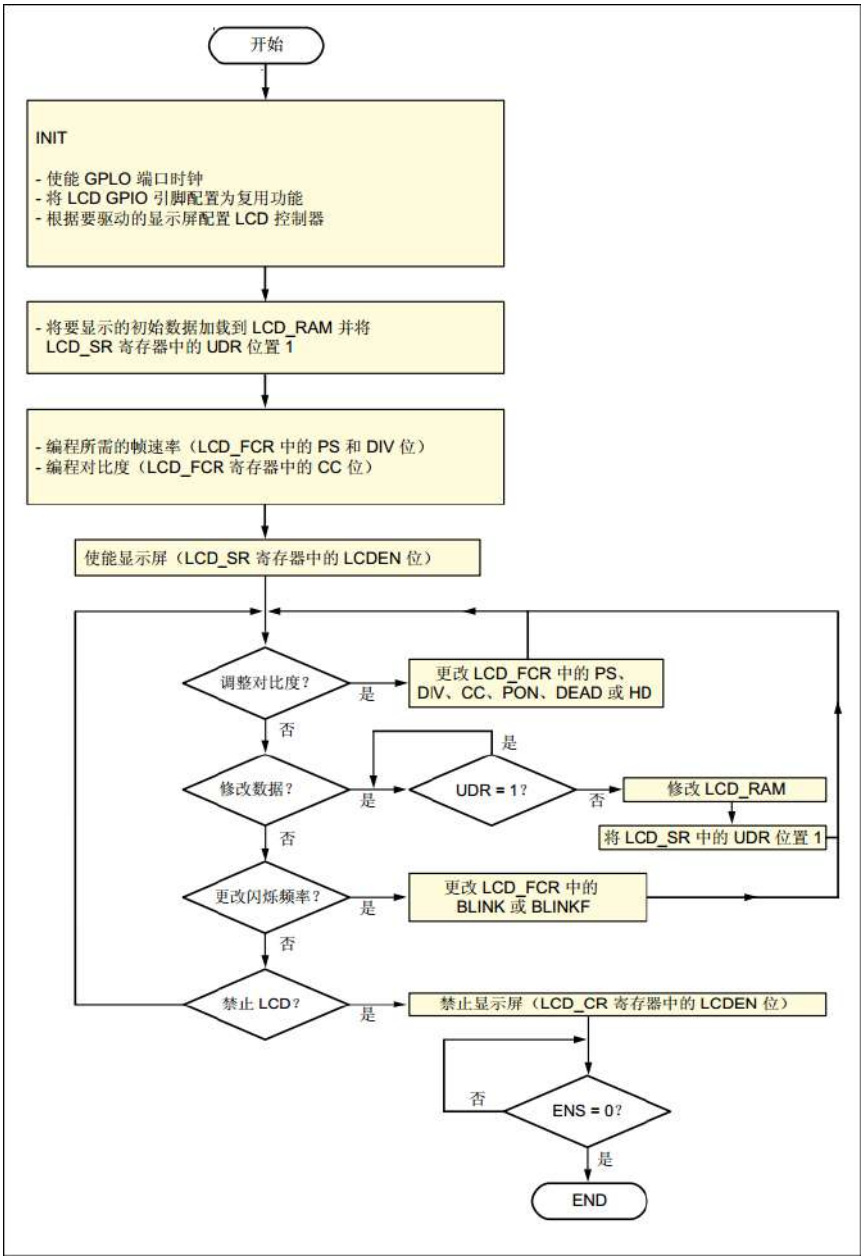
配置位		QFP64/BGA64	输出引脚	功能
占空比	MUX_SEG			
1/8	0/1	28x8	SEG[51:48]/SEG[31:28]/COM[7:4]	COM[7:4]
			COM[3:0]	COM[3:0]
			SEG[27:0]	SEG[27:0]
1/4	0	28x4	SEG[51:48]/SEG[31:28]/COM[7:4]	未使用
			SEG[27:0]	SEG[27:0]
			COM[3:0]	COM[3:0]
	1	32x4	COM[3:0]	COM[3:0]
			SEG[51:48]/SEG[31:28]/COM[7:4]	SEG[31:28]
			SEG[27:0]	SEG[27:0]
1/3		28x3	COM3	未使用
			COM[2:0]	COM[2:0]
			SEG[51:48]/SEG[31:28]/COM[7:4]	未使用
			SEG[31:0]	SEG[31:0]
		32x3	COM3	未使用
			COM[2:0]	COM[2:0]
			SEG[51:48]/SEG[31:28]/COM[7:4]	SEG[31:28]
			SEG[27:0]	SEG[27:0]
1/2	0	28x2	COM[3:2]	未使用
			COM[1:0]	COM[1:0]
			SEG[51:48]/SEG[31:28]/COM[7:4]	未使用
			SEG[27:0]	SEG[27:0]
	1	32x2	COM[3:2]	未使用
			COM[1:0]	COM[1:0]
			SEG[51:48]/SEG[31:28]/COM[7:4]	SEG[31:28]
			SEG[27:0]	SEG[27:0]
STATIC	0	28X1	COM[3:1]	未使用
			COM0	COM0
			SEG[51:48]/SEG[31:28]/COM[7:4]	未使用
			SEG[27:0]	SEG[27:0]
	1	32X1	COM[3:1]	未使用
			COM0	COM0
			SEG[51:48]/SEG[31:28]/COM[7:4]	SEG[31:28]
			SEG[27:0]	SEG[27:0]

图 15-11 **SEG/COM** 多路复用功能示例



15.4.8 流程图

图 15-12 流程图示例



15.4.9 LCD 低功耗模式

LCD 控制器可在停止模式下进行显示，也可以完全禁止以降低功耗。

表 15-5 低功耗模式下的 LCD 行为

模式	说明
停止	LCD 仍处于激活状态
待机	LCD 未处于激活状态

15.4.10 LCD 中断

下表列出了 LCD 中断请求。

表 15-6 **LCD** 中断请求

中断事件	事件标志	事件标志/中断清除方法	中断使能控制位
帧起始 (SOF)	SOF	写入 SOFC = 1	SOFIE
更新显示完成 (UDD)	UDD	写入 UDDC = 1	UDDIE

帧起始 (SOF)

如果 SOFIE（帧起始中断使能）位置 1，则执行 LCD 帧起始中断（请参见 *LCD 帧控制寄存器 (LCD_FCR)*）。执行相应的中断处理向量时在 LCD_CLR 寄存器中将 SOFC 位写为 1，可将 SOF 清除。

更新显示完成 (UDD)

如果 UDDIE（更新显示完成中断使能）位置 1，则执行 LCD 更新显示中断（请参见 *LCD 帧控制寄存器 (LCD_FCR)*）。执行相应的中断处理向量时在 LCD_CLR 寄存器中将 UDDC 位写为 1，可将 UDD 清除。

根据产品实现的不同，上述所有中断事件既可以共享同一个中断向量（LCD 全局中断），也可以分组到 2 个不同的中断向量上（LCD SOF 中断和 LCD UDD 中断）。

要使能 LCD 中断，需按照以下顺序操作：

1. 配置 NVIC 中的 LCD IRQ 通道并将其使能
2. 配置 LCD 以生成中断

15.5 段式 LCD 寄存器

15.5.1 LCD 控制寄存器（LCD_CR）

偏移地址： 0x00

复位值： 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PWR_MD	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MUX_SEG	BIAS[1:0]		DUTY[2:0]			VSEL	LCDEN
								rw	rw	rw	rw	rw	rw	rw	rw
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位 31:8	保留，必须保持复位值
位 15	**PWR_MD：** 当选择 VCC 作为 LCD 电源时，PWR_MD 置 1 可以无需外部物理连接 VLCD。 当 VSEL=0 时，PWR_MD 是否置位无影响。 当 VSEL=1 时： 0：需要外部物理连接 Vcc 作为 Vlcd 1：无需外部物理连接 Vcc 作为 Vlcd
位 7	**MUX_SEG**：多路复用区段使能 (Mux segment enable) 此位用于使能 SEG 引脚重映射。四个 SEG 引脚可与 SEG[31:28] 多路复用。 0：禁止 SEG 引脚多路复用 1： SEG[31:28] 与 SEG[43:40] 多路复用

位 6:5	**BIAS[1:0]**：偏置选择器 (Bias selector) 这些位决定使用的偏置。禁止使用值 11。 00：偏置 1/4 01：偏置 1/2 10：偏置 1/3 11：保留
位 4:2	**DUTY[2:0]**：占空比选择 (Duty selection) 这些位决定占空比。禁止使用值 101、 110 和 111。 000：静态占空比 001： 1/2 占空比 010： 1/3 占空比 011： 1/4 占空比 100： 1/8 占空比 101：保留 110：保留 111：保留
位 1	**VSEL**：电压源选择 (Voltage source selection) VSEL 位决定 LCD 的电压源。 0：内部源（电压升压转换器） 1：外部源（VLCD 引脚）
位 0	**LCDEN**： LCD 控制器使能 (LCD controller enable) 通过软件将此位置 1 可使能 LCD 控制器/驱动器。通过软件将此位清零可在下一帧开始时关闭 LCD。当 LCD 禁用时，所有 COM 和 SEG 引脚都被驱动到 Vss。当此位置 1 时，必须在 PWR_CR 中复位 ULP 位。 0：禁止 LCD 控制器 1：使能 LCD 控制器

注： 当使能 *LCD* 时（*LCD_SR* 中的 *ENS* 位置 *1*）， *VSEL*、 *MUX_SEGBIAS* 和 *DUTY* 位处于写保护状态。

15.5.2 LCD 帧控制寄存器（LCD_FCR）

偏移地址： 0x04

复位值： 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	PS[3:0]				DIV[3:0]				BLINK[1:0]	
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BLINKF[2:0]			CC[2:0]			DEAD[2:0]			PON[2:0]			UDDIE	MD	SOFIE	HD
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位 31:26	保留，必须保持复位值
位 25:22	**PS[3:0]**： PS 16 位预分频器 (PS 16-bit prescaler) 这些位通过软件写入，用于定义 PS 16 位预分频器的分频系数。 ck_ps = LCDCLK/(2)。 0000： ck_ps = LCDCLK 0001： ck_ps = LCDCLK/2 0002： ck_ps = LCDCLK/4 ... 1111： ck_ps = LCDCLK/32768

位 21:18	**DIV[3:0]**： DIV 时钟分频器 (DIV clock divider) 这些位通过软件写入，用于定义 DIV 分频器的分频系数。 0000： ck_div = ck_ps/16 0001： ck_div = ck_ps/17 0002： ck_div = ck_ps/18 ... 1111： ck_div = ck_ps/31
位 17:16	**BLINK[1:0]**：闪烁模式选择 (Blink mode selection) 00：禁止闪烁 01：在 SEG[0]、 COM[0] 上启用闪烁（1 个像素） 10：在 SEG[0]、所有 COM 上启用闪烁（最多 8 个像素，取决于编程的占空比） 11：在所有 SEG 和所有 COM 上启用闪烁（所有像素）
位 15:13	**BLINKF[2:0]**：闪烁频率选择 (Blink frequency selection) 000： f_{LCD}/8 001： f_{LCD}/16 010： f_{LCD}/32 011： f_{LCD}/64 100： f_{LCD}/128 101： f_{LCD}/256 110： f_{LCD}/512 111： f_{LCD}/1024
位 12:10	**CC[2:0]**：对比度控制 (Contrast control) 这些位指定 V_{LCD} 最大电压之一（与 V_{DD} 无关）。范围为 2.60 V 至 3.51 V。 000： V_{LCD0} 001： V_{LCD1} 010： V_{LCD2} 011： V_{LCD3} 100： V_{LCD4} 101： V_{LCD5} 110： V_{LCD6} 111： V_{LCD7} 有关 V_{LCDx} 的值，请参见产品数据手册。
位 9:7	**DEAD[2:0]**：死区持续时间 (Dead time duration) 这些位通过软件写入，用于配置帧间死区的长度。在死区内， COM 和 SEG 电压电平保持为 0 V 以降低对比度，而无需修改帧速率。 000：无死区 001： 1 个相位周期死区 010： 2 个相位周期死区 ..... 111： 7 个相位周期死区

位 6:4	**PON[2:0]**：脉冲打开持续时间 (Pulse ON duration) 这些位通过软件写入，用于根据 ck_ps 脉冲定义脉冲持续时间。较短的脉冲有助于降低功耗，但内部电阻较高的显示器可能需要更长的脉冲才能达到令人满意的对比度。 请注意，脉冲长度永远不会超过预分频 LCD 时钟周期的一半。 000:0 001： 1/ck_ps 010： 2/ck_ps 011： 3/ck_ps 100： 4/ck_ps 101： 5/ck_ps 110： 6/ck_ps 111： 7/ck_ps PON 持续时间示例（LCDCLK = 32.768 kHz 且 PS = 0x03）： 000： 0 µs 001： 244 µs 010： 488 µs 011： 782 µs 100： 976 µs 101： 1.22 ms 110： 1.46 ms 111： 1.71 ms
位 3	**UDDIE**：更新显示完成中断使能 (Update display done interrupt enable) 此位由软件置 1 和清零。 0：禁止 LCD 更新显示完成中断 1：使能 LCD 更新显示完成中断
位 2	**MD:** 中驱动强度输出使能。 此位通过软件写入，用于使能中电阻分频器。MD =1，且 HD=0 时，选择中等驱动强度。 HD=1 时，则不论 MD 为 0 或 1，输出都为高驱动使能。 0: 禁止永久中等驱动 1: 使能永久中等驱动
位 1	**SOFIE**：帧起始中断使能 (Start of frame interrupt enable) 此位由软件置 1 和清零。 0：禁止 LCD 帧起始中断 1：使能 LCD 帧起始中断
位 0	**HD**：高驱动使能 (High drive enable) 此位通过软件写入，用于使能低电阻分频器。内部电阻较高的显示器可能需要更长的驱动时间才能达到令人满意的对比度。在容许额外功耗的情况下，此位非常有用。 0: 禁止永久高驱动 1: 使能永久高驱动 当 HD=1 时，必须将 PON 位编程为 001。

注： 可以随时更新此寄存器中的数据，但仅在下一帧开始时才应用新值（能够立即影响器件行为的 *UDDIE* 和 *SOFIE* 除外）。

CC[2:0] 位的新值也会立即应用，但由于电压发生器的缘故，其对器件的作用会在下一帧开始时生效。

读取该寄存器可获得在寄存器中写入的最后一个值，而不是用于显示当前帧的配置。

15.5.3 LCD 状态寄存器（LCD_SR）

偏移地址： 0x08

复位值： 0x0000 0020

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FCRSF	RDY	UDD	UDR	SOF	ENS
										r	r	rw	r/w1	rw	rw
0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0

位 31:6	保留，必须保持复位值
位 5	**FCRSF**： LCD 帧控制寄存器同步标志 (LCD Frame Control Register Synchronization flag) 每次在 LCDCLK 域中更新 LCD_FCR 寄存器时，此位都会由硬件置 1。对 LCD_FCR 寄存器执行写操作时，此位将由硬件清零。 0： LCD 帧控制寄存器尚未同步 1： LCD 帧控制寄存器已同步
位 4	**RDY**：就绪标志 (Ready flag) 此位通过硬件置 1 和清零。它指示升压转换器的状态。 0：未就绪 1：升压转换器已使能且准备好提供正确的电压。
位 3	**UDD**：更新显示完成 (Update Display Done) 此位由硬件置 1，通过向 LCD_CLR 寄存器中的 UDDC 写入 1 进行清零。位置 1 的优先级高于 位清零。 0：无事件 1：更新显示请求完成。如果 LCD_FCR 寄存器中的 UDDIE 位置 1，将产生 UDD 中断。 注： 如果器件处于停止模式（未提供 *PCLK*），即使 *UDDIE = 1*， *UDD* 也不会产生中断。如果显示器未使能，将永远不会发生 *UDD* 中断。
位 2	**UDR**：更新显示请求 (Update display request) 每次软件修改 LCD_RAM 时，必须将 UDR 位置 1 以将更新的数据传送到第二级缓冲器。 UDR 位在更新结束之前保持置 1，在此期间 LCD_RAM 处于写保护状态。 0：无影响 1：更新显示请求 注： 当禁止显示器时，针对所有 *LCD_DISPLAY* 存储单元执行更新。当使能显示器时，仅针对公共线激活（取决于 *DUTY*）的存储单元执行更新。例如，如果 *DUTY = 1/2*，将仅更新 *COM0* 和 *COM1* 的 *LCD_DISPLAY*。 注： 当此位已为 *1* 时，写入 *0* 或写入 *1* 都无影响。此位只能由硬件清零。仅当 *LCDEN=1* 时，此位才能清零
位 1	**SOF**：帧起始标志 (Start of frame flag) 此位在新的帧开始时由硬件置 1，同时更新显示数据。此位通过向 LCD_CLR 寄存器中的 SOFC 位写入 1 进行清零。位清零的优先级高于位置 1。 0：无事件 1：发生帧起始事件。如果 SOFIE 位置 1，则产生 LCD 帧 起始中断。
位 0	**ENS**： LCD 使能状态 (LCD enabled status) 此位通过硬件置 1 和清零。它指示 LCD 控制器状态。 0：禁止 LCD 控制器 1：使能 LCD 控制器 注： 当 *LCD_CR* 寄存器中的 *LCDEN* 位从 *0* 变为 *1* 时， *ENS* 位立即置 *1*。禁用时，此位反映 *LCD* 的实际状态，因此在最后显示的帧结束时变为 *0*。

15.5.4 LCD 清零寄存器（LCD_CLR）

偏移地址： 0x0C

复位值： 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

位 31:4	保留，必须保持复位值
位 3	**UDDC**：更新显示完成清零 (Update display done clear) 此位由软件写入，用于将 LCD_SR 寄存器中的 UDD 标志清零。 0：无影响 1：将 UDD 标志清零
位 2	保留，必须保持复位值
位 1	**SOFC**：帧起始标志清零 (Start of frame flag clear) 此位由软件写入，用于将 LCD_SR 寄存器中的 SOF 标志清零。 0：无影响 1：将 SOF 标志清零
位 0	保留，必须保持复位值

15.5.5 LCD 显示器存储器（LCD_RAM）

偏移地址： 0x14 到 0x50

复位值： 0x0000 0000

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

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

位 31:0	**SEGMENT_DATA[31:0]** 每个位对应于 LCD 显示器的一个像素。 0：像素未激活 1：像素激活

16 模拟比较器 (COMP)

16.1 简介

HK32L08x 器件内置两个低功耗比较器 COMP1 和 COMP2，这两个比较器可分别用作独立器件，也可与定时器结合使用。

这两个比较器可用于多种功能，包括：

- 在模拟信号的触发下从低功耗模式唤醒
- 模拟信号调理
- 与 DAC 和定时器的 PWM 输出结合使用时，构成逐周期电流控制环路
- 两个比较器联合工作，作为窗口比较器

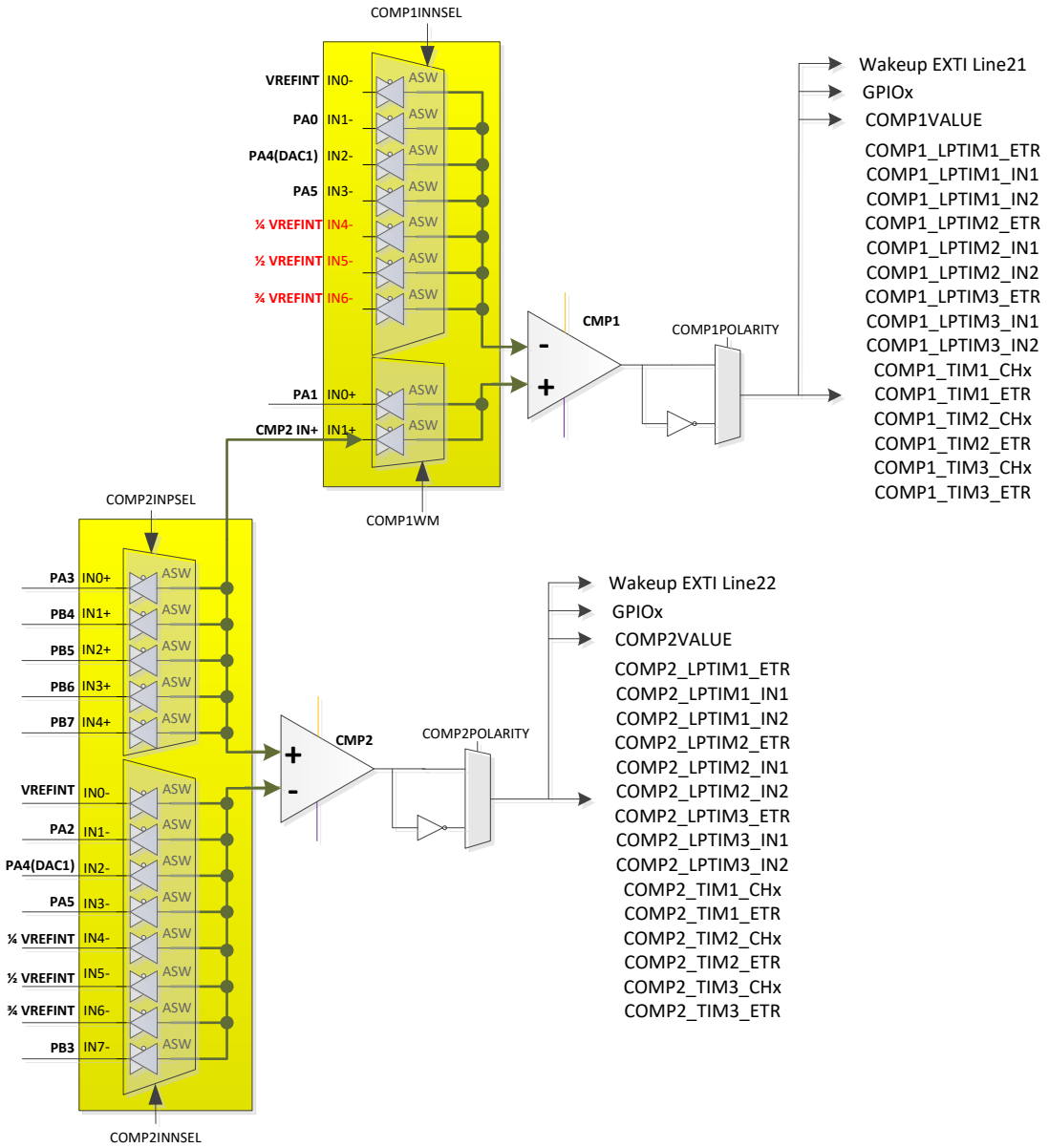
16.2 CMP 主要功能

- 比较器功耗模式可配置
- 比较器拥有施密特迟滞功能
- 比较器输出极性可配置
- 每个比较器都具有正输入和可配置的负输入，用于灵活选择电压：
  - I/O 引脚
  - DAC
  - 通过调节器（缓冲分压器）提供的内部参考电压和三个因数分压值（1/4、1/2、3/4）
- 输出可以重定向到用于触发以下事件的 I/O 或定时器输入：
  - 捕捉事件
  - GPIO
  - Timer/LP Timer
- 从睡眠模式和停止模式唤醒（通过 EXIT 控制器）

16.3 CMP 功能说明

16.3.1 CMP 框图

图 16-1 CMP 框图



16.3.2 CMP 引脚和内部信号

- 用作比较器输入的 I/O 必须在 GPIO 寄存器中配置为模拟模式。
- 比较器输出可以使用数据手册的“复用功能映射”表中给出的复用功能通道连接到 I/O。
- 输出也可以在内部重定向到用于以下用途的各种定时器输入：
  - 用于时序测量的输入捕捉
- 可以在内部和外部同时对比较器输出进行重定向。

16.3.3 CMP 复位和时钟

时钟控制器提供的 COMP 时钟与 PCLK（APB 时钟）同步。

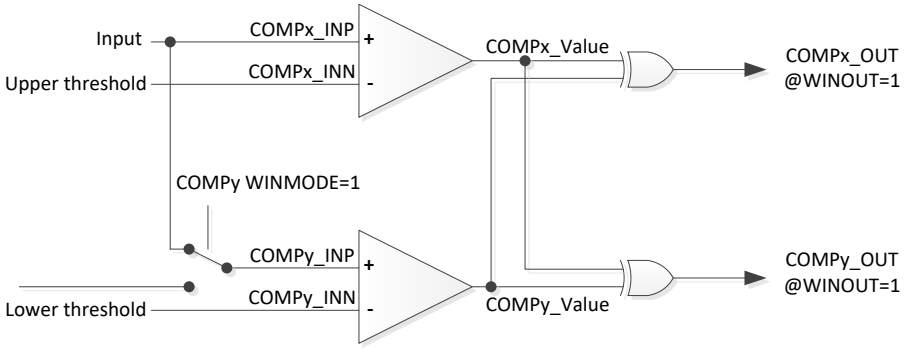
RCC 控制器中不单独提供其时钟使能控制位。COMP 和 SYSCFG 共用复位和时钟使能位。

16.3.4 CMP Window 模式

Window 模式用来监控模拟电压在规定的上下门限范围内。两个模拟比较器联合工作提供这个功能。被监控的电压连接到两个比较器的正端。上下门限电压分别接到比较器的负端。

两个比较器的正端，可以通过 WINMODE 设置在芯片内部短接。连接方式如下图。

图 16-2



16.3.5 CMP 锁定机制

这两个比较器可用于过流或热保护等安全用途。对于具有特定功能安全要求的应用，必须保证在发生意外寄存器访问或程序计数器损坏时，不能更改比较器编程。

为此，可以对比较器控制和状态寄存器进行写保护（只读）。

一旦编程完成，COMPxLOCK 位便会设置为 1。这将导致整个 COMPx_CSR 寄存器变成只读，包括 COMPxLOCK 位。

只能通过 MCU 复位来复位写保护。

16.4 CMP 中断

比较器输出从内部连接到扩展中断和事件控制器。每个比较器都有其各自的 EXTI 线，能够产生中断或事件。该机制还可用于退出低功耗模式。

16.5 模拟比较器寄存器

16.5.1 比较器 1 控制和状态寄存器 (COMP1_CSR)

COMP1_CSR 为比较器 1 控制/状态寄存器。此寄存器包含与比较器 1 相关的所有位/标志。

偏移地址：0x18

系统复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
COMP1LOCK	COMP1VALUE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	COMP1PowerMode	COMP1Hyster	

rs	r												rw	rw	rw
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
COMP 1POL ARITY	Comp 1wino ut	Res.	Res.	Res.	Res.	Res.	COMP 1WM	Res.	COMP1INNSEL			Res.	Res.	Res.	COMP 1EN
rw	rw						rw		rw	rw	rw				rw
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位 31	**COMP1LOCK:** COMP1_CSR 寄存器锁定位(COMP1_CSR register lock bit) 此位由软件置 1，通过硬件系统复位清零。它将锁定比较器 1 控制寄存器(COMP1_CSR[31:0])的全部内容 0：比较器 1 的 COMP1_CSR[31:0]可读/写 1：比较器 1 的 COMP1_CSR[31:0]只读
位 30	**COMP1VALUE:** 比较器 1 输出状态位(Comparator 1 output status bit) 此位为只读。它反映当前比较器 1 输出的状态（受到 COMP1POLARITY 位的影响）
位 29:19	保留，必须保持复位值
位 18	**COMP1PowerMode:** 0：低功耗模式 1：高功耗模式
位 17:16	**COMP1Hyster:** Hysteresis 电压选择 00：无施密特 01: 30mV 10: 70mV 11: 100mV
位 15	**COMP1POLARITY:** 比较器 1 极性选择位 (Comparator 1 polarity selection bit) 此位由软件置 1 和清零（COMP1LOCK 置 1 时除外）。它使比较器 1 的极性反相。 0：比较器 1 输出值不反相 1：比较器 1 输出值反相
位 14	**COMP1Winout:** Comp1 的输出选择 =0，comp1_value =1，comp1_value XOR comp2_value
位 12:9	保留，必须保持复位值
位 8	**COMP1WM:** 比较器 1 窗口模式选择位 (Comparator 1 window mode selection bit) 此位由软件置 1 和清零（COMP1LOCK 置 1 时除外）。它选择比较器 1 窗口模式，其中两个比较器的正输入连接在一起。 0：比较器 1 的正输入连接到 PA1。 1：比较器 1 的正输入与比较器 2 的正输入短接（请参见 COMP1_CSR）。
位 7	保留，必须保持复位值
位 6:4	**COMP1INNSEL:** 比较器 1 负输入连接配置位 (Comparator 1 Input Minus connection configuration bit) 这些位由软件置 1 和清零（COMP1LOCK 置 1 时除外）。它们选择哪个输入连接到比较器 1 的负输入 000：VREFINT 001：PA0 010：DAC1/PA4 011：DAC2/PA5 100：1/4 VREFINT 101：2/4 VREFINT 110：3/4 VREFINT
位 3:1	保留，必须保持复位值
位 0	**COMP1EN:** 比较器 1 使能位 (Comparator 1 enable bit) 此位由软件置 1 和清零（COMP1LOCK 置 1 时除外）。开关比较器 1 0：关闭比较器 1。 1：开启比较器 1。

16.5.2 比较器 2 控制和状态寄存器 (COMP2_CSR)

COMP2_CSR 为比较器 2 控制/状态寄存器。此寄存器包含与比较器 2 相关的所有位/标志。

偏移地址：0x1C

系统复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
COMP2LOCK	COMP2VALUE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	COMP2PowerMode	COMP2Hyster	
rs	r												rw	rw	rw
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
COMP2POLARITY	Comp2winout	Res.	Res.	Res.	COMP2INPSEL			Res.	COMP2INNSEL			Res.	Res.	Res.	COMP2EN
rw	rw				rw	rw	rw		rw	rw	rw				rw
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位 31	**COMP2LOCK:** COMP2_CSR 寄存器锁定位 (COMP2_CSR register lock bit) 此位由软件置 1，通过硬件系统复位清零。它将锁定比较器 2 控制寄存器(COMP2_CSR[31:0])的全部内容 0：比较器的 COMP2_CSR[31:0]可读/写 1：比较器的 COMP2_CSR[31:0]只读
位 30	**COMP2VALUE:** 比较器 2 输出状态位 (Comparator 2 output status bit) 此位为只读。它反映当前比较器 2 输出的状态（考虑了 COMP2POLARITY 位的影响）
位 29:19	保留，必须保持复位值
位 18	**COMP2PowerMode:** 0：低功耗模式 1：高功耗模式
位 17:16	**COMP2Hyster:** Hysteresis 电压选择 00：无施密特 01: 30mV 10: 70mV 11: 100mV
位 15	**COMP2POLARITY:** 比较器 2 极性选择位 (Comparator 2 polarity selection bit) 此位由软件置 1 和清零（COMP2LOCK 置 1 时除外）。它使比较器 2 的极性反相。 0：比较器 2 输出值不反相 1：比较器 2 输出值反相
位 14	**COMP2Winout:** Comp2 的输出选择 =0，comp2_value =1，comp1_value XOR comp2_value
位 13:11	保留，必须保持复位值
位 10:8	**COMP2INPSE:** 比较器 2 正输入连接配置位 (Comparator 2 Input Plus connection configuration bit) 这些位由软件置 1 和清零（COMP2LOCK 置 1 时除外）。它们选择哪个输入连接到比较器 2 的正输入 000：PA3 001：PB4 010：PB5 011：PB6 100：PB7 其它值：保留
位 7	保留，必须保持复位值

位 6:4	**COMP2INNSE:**比较器 2 负输入连接配置位 (Comparator 2 Input Minus connection configuration bit) 这些位由软件置 1 和清零（COMP2LOCK 置 1 时除外）。它们选择哪个输入连接到比较器 2 的负输入。 000：VREFINT 001：PA2 010：DAC1/PA4 011：DAC2/PA5 100：1/4 VREFINT 101：1/2 VREFINT 110：3/4 VREFINT 111：PB3
位 3:1	保留，必须保持复位值
位 0	**COMP2EN:** 比较器 2 使能位 (Comparator 2 enable bit) 此位由软件置 1 和清零（COMP2LOCK 置 1 时除外）。开关比较器 2。 0：关闭比较器 2。 1：开启比较器 2。

17 模拟运算放大器 (OPAMP)

17.1 简介

HK32L08x 集成了 3 个运算放大器。它们分别可以工作在 Standalone、Follower、PGA 三种模式。运放的输出，可以输出到管脚，也可以内部反馈到反相输入端，还可以被选通输入到内部 ADC 进行采样。

17.2 OPA 主要功能

- 轨到轨输入输出
- 低 offset 电压
- 支持 Standalone、Follower、PGA 模式
- 多路输入复用

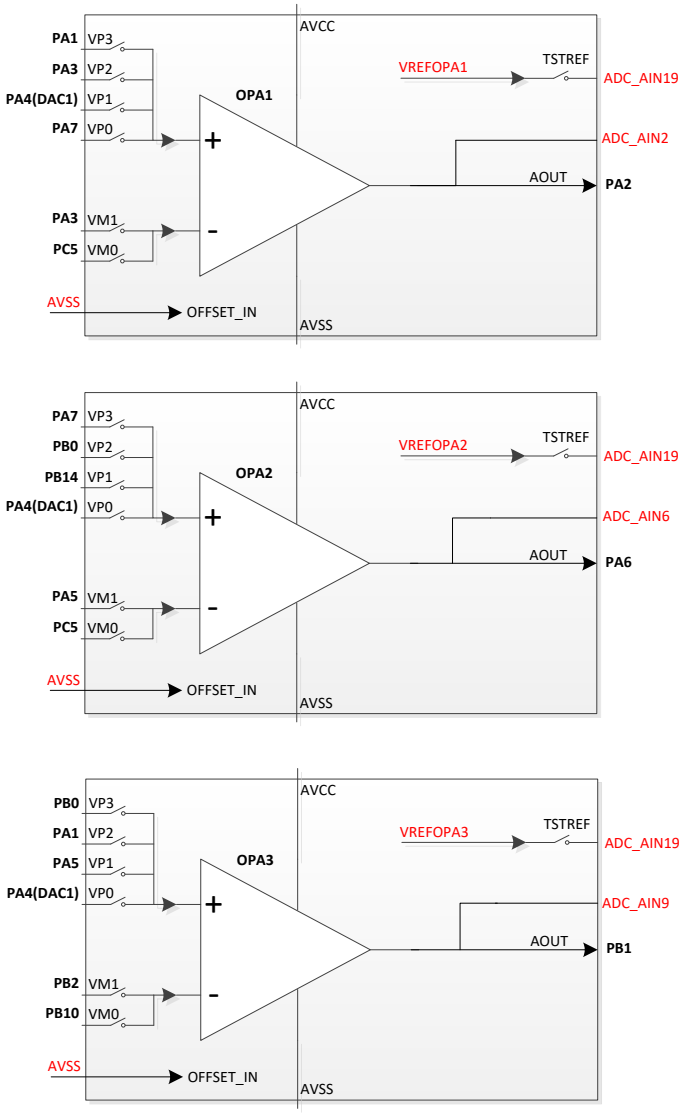
17.3 OPA 功能说明

每一个运放，正端支持从外部管脚 4 选 1 输入，负端支持从外部管脚 2 选 1 输入。输入选择通过寄存器 OPAx_CSR 的 VM_SEL/VP_SEL 进行配置。

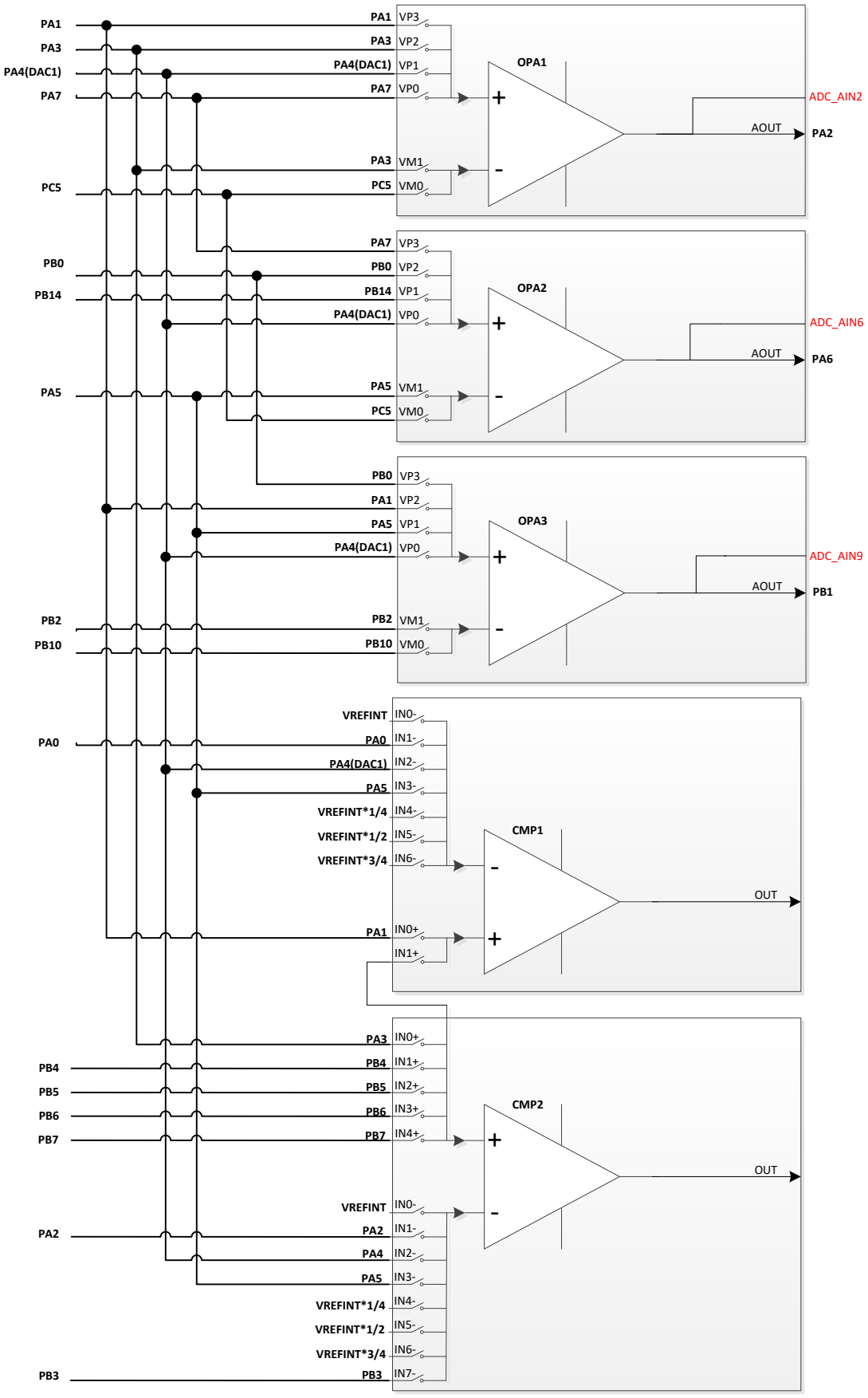
被用作运放输入输出的 IO 必须配置为模拟工作模式。

OPA1 Inverting	OPA1 non-Inverting	OPA1 Output
PA3(VM1)	PA1(VP3)	PA2
PC5(VM0)	PA3(VP2)	
	PA4(VP1)	
	PA7(VP0)	
OPA2 Inverting	OPA2 non-Inverting	OPA2 Output
PA5 (VM1)	PA7(VP3)	PA6
PC5 (VM0)	PB0(VP2)	
	PB14(VP1)	
	PA4(VP0)	
OPA3 Inverting	OPA3 non-Inverting	OPA3 Output
PB2(VM1)	PB0(VP3)	PB1
PB10(VM0)	PA1(VP2)	
	PA5(VP1)	
	PA4(VP0)	

17.3.1 框图



注意：PA4(DAC1)，如果 DAC 没有使能，选择从 PA4 管脚输入信号；如果 DAC 已经使能，PA4 管脚会被占用，可以作为监控测量功能。


OPA1
PA1
PA3
PA4(DAC1)
PA7
PC5
VP3
VP2
VP1
VP0
VM1
VM0
AOUT
ADC_AIN2
PA2
OPA2
PB0
PB14
PA5
ADC_AIN6
PA6
OPA3
PB2
PB10
ADC_AIN9
PB1
CMP1
VREFINT
PA0
VREFINT*1/4
VREFINT*1/2
VREFINT*3/4
IN0-
IN1-
IN2-
IN3-
IN4-
IN5-
IN6-
IN0+
IN1+
OUT
CMP2
PB4
PB5
PB6
PB7
PB3
IN7-

17.3.2 时钟

运放的时钟同步于 APB，在 RCC 控制器中没有单独的 OPA 时钟控制位。如果需要使用 OPA，需要将 SYSCFG 时钟使能。

17.3.3 ADC 采样 OPA 输出

OPA	ADC Unit	Channels	OPA Output Pin
OPA1	ADC1	AIN2	PA2
OPA2	ADC1	AIN6	PA6
OPA3	ADC1	AIN9	PB1

注意：

在使能 OPA 后，对应的输出管脚，将被一直占用。

在使能 OPA 后，对应的 ADC 输入通道也被占用，是否选通采样由 ADC 通道选择器控制

17.3.4 Multiplexer Mode 控制

HK32L08x 对运放输入通道提供了 2 套方案，2 套方案的配置寄存器分别是 VP_SEL/VM_SEL 和 VPS_SEL/VMS_SEL。这 2 套方案的选择通过 TCM_EN 寄存器来进行控制。

该方案配合 Standalone/Follower/PGA 工作模式，为用户电路设计提供了非常大的灵活性。

17.3.5 校正

运放的轨到轨输入级由 2 组差分对 MOS 管组成：一组 NMOS 差分对管和一组 PMOS 差分对管。每一个运放在出厂之前都会对 2 组 MOS 管进行 Trimming，并保存 Trimming 结果到芯片里。

用户也可以通过设置 OPA 控制器的 USER_TRIM 位，再次对运放进行 Trimming 操作。

TRIMOFFSETN 寄存器位负责 NMOS 差分对管校正，TRIMOFFSETP 寄存器位负责 PMOS 差分对管校正。

为了 NMOS 差分对管校正，需要配置 CALON=1 和 CALSEL=11。内部 0.9*VDDA 电压会接到 OPA 的正负端。正负输入端的电压可以被测量。运放的参考电压也可以通过配置 TSTREF 连接到 ADC 输入通道进行采样。软件配置 TRIMOFFSETN 寄存器，从 0x0 逐渐增加，一直到 OUTCAL 从 1 变为 0。如果 OUTCAL 位被复位，说明运放 Offset 被正确校正，并且对应的 Trimming 值将会被保存起来。

NMOS 差分对，软件校正流程如下：

- Enable OPAxEn bit
- Enable USERTRIM bit
- 设置 CALON bit 连接运放 VM、VP 到内部参考电压
- 设置 CALSEL=11
- 逐渐增加 TRIMOFFSETN，直到 OUTCAL 被清零。保存 TRIMOFFSETN

对于 PMOS 软件配置流程完全相同，只是需要配置 CALSEL=01，TRIMOFFSETP 寄存器需要被软件配置。内部电压 0.1*VDDA 会接到运放的正负输入端。

17.3.6 OPA 工作模式

运放的输入和输出，都连接到了管脚上。可以支持三种工作模式：

- Standalone mode，外部增益设置
- 跟随器模式
- PGA 模式

Standalone

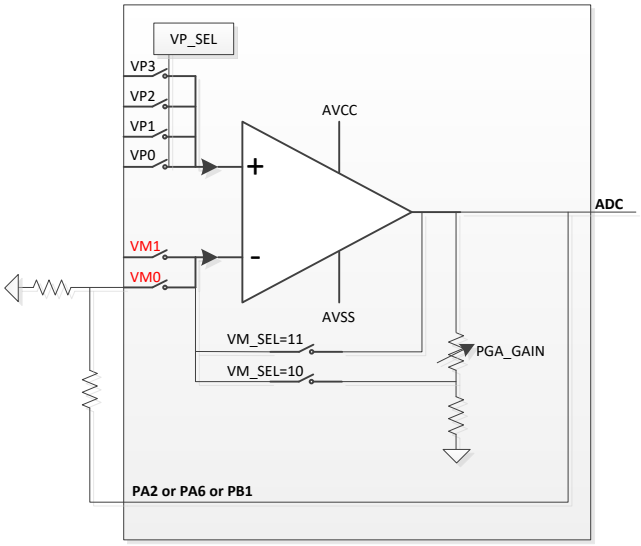
Standalone 模式的增益反馈网络在芯片外，运放的正负输入以及输出都连接到管脚，用户电路设计灵活。

通过设置 OPAx_CR 寄存器的 VP_SEL 对正端输入管脚进行 4 选 1。通过设置 OPAx_CR 寄存器的 VM_SEL 对负端输入管脚进行 2 选 1。运放输出，通过芯片外围电路与反相输入端连接。

不管是正端，还是负端，没有选通使用的 IO，都可以当作 GPIO 使用。

VM_SEL	管脚
00	VM0
01	VM1

结构如下图：



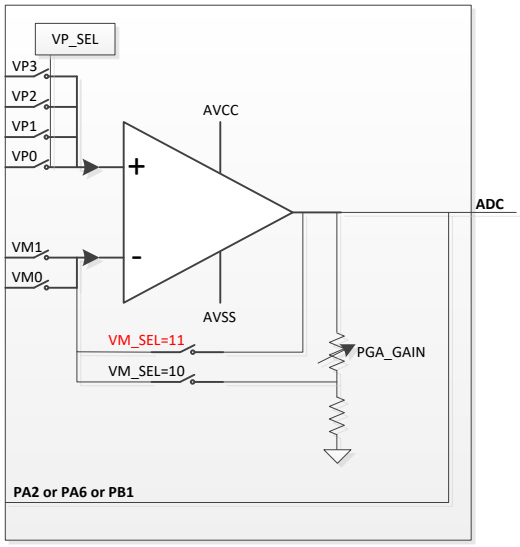
Follower

配置 VM_SEL=11，运放工作为跟随器模式。该模式对信号起到驱动作用，并提供高的输出阻抗。

运放的输出在芯片内部与反相输入端连接。通过设置 OPAx_CR 寄存器的 VP_SEL 对正端输入管脚进行 4 选 1。

正端没有选通使用的 IO，都可以当作 GPIO 使用。VM0/VM1 管脚都可以当作 GPIO 使用。

结构如下图：



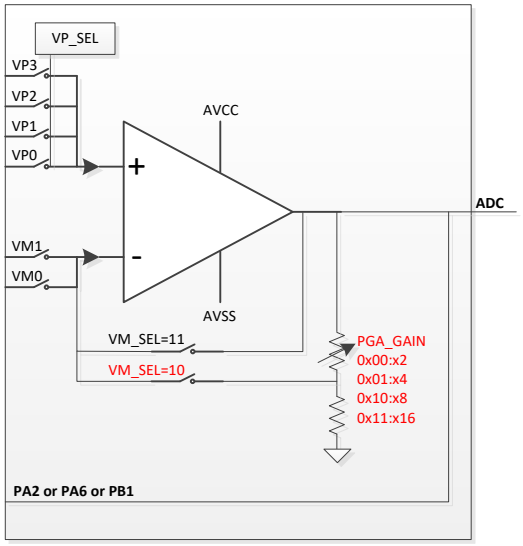
PGA

配置 VM_SEL=10，运放工作为内部增益可编程模式。负反馈增益网络通过 PGA_GAIN 寄存器设置，增益固定有 2/4/8/16 倍。通过设置 OPAx_CR 寄存器的 VP_SEL 对正端输入管脚进行 4 选 1。

在这个模式下，支持两种应用电路。

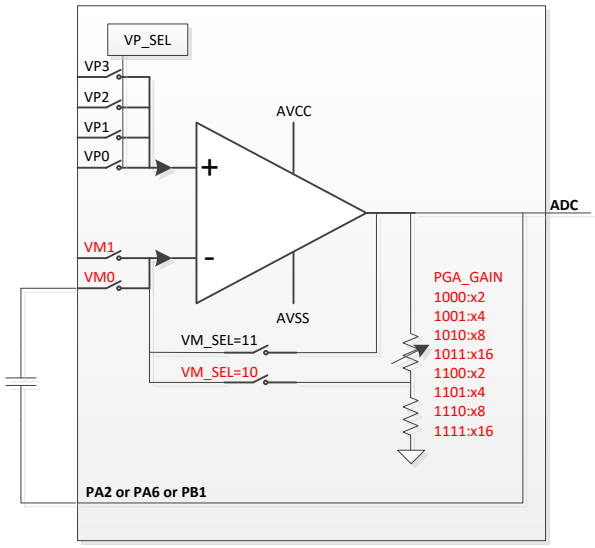
电路一：VM0/VM1 管脚不需要，内部增益可编程，内部增益 x2/x4/x8/x16 可选

正端没有选通使用的 IO，都可以当作 GPIO 使用。VM0/VM1 管脚都当作 GPIO 使用。



电路二：有源低通滤波器，VM0/VM1 管脚需要选通，并且 VM0/VM1 管脚与运放输出管脚之间有外接电路，内部增益可编程，内部增益 x2/x4/x8/x16 可选

正端没有选通使用的 IO，都可以当作 GPIO 使用。负端 VM0/VM1，没有选通的管脚可以当作 GPIO 使用。



PGA_GAIN 寄存器设置如下：

0X00 = Non-inverting gain = 2，VM0/VM1 无连接
0X01 = Non-inverting gain = 4，VM0/VM1 无连接
0X10 = Non-inverting gain = 8，VM0/VM1 无连接
0X11 = Non-inverting gain = 16，VM0/VM1 无连接
1000 = Non-inverting gain = 2 - Internal feedback connected to VM0
1001 = Non-inverting gain = 4 - Internal feedback connected to VM0
1010 = Non-inverting gain = 8 - Internal feedback connected to VM0
1011 = Non-inverting gain = 16 - Internal feedback connected to VM0
1100 = Non-inverting gain = 2 - Internal feedback connected to VM1

1101 = Non-inverting gain = 4 - Internal feedback connected to VM1
1110 = Non-inverting gain = 8 - Internal feedback connected to VM1
1111 = Non-inverting gain = 16 - Internal feedback connected to VM1

17.4 模拟运算放大器寄存器

17.4.1 OPA1 control register (OPA1_CSR)

偏移地址 ： 0x38

复位值 ： 0xXXXX 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LOCK	OUTCAL	TSTREF	TRIMOFFSETN					TRIMOFFSETP					USER_TRIM	PGA_GAIN	
rw	r	rw	rw					rw					rw	rw	rw
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PGA_GAIN		CALSEL		CALON	VPS_SEL		VMS_SEL	TCM_EN	VM_SEL		Res.	VP_SEL		FORCE_VP	OPAEN
rw		rw		rw	rw		rw	rw	rw	rw		rw	rw	rw	rw
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit 31	**LOCK:** OPA1 lock 这一位由软件置位，可以被系统复位清零。置位将使 CSR 寄存器只读。 0: OPA1_CSR is read-write. 1: OPA1_CSR is read-only
Bit 30	**OUTCAL:** OPA output status flag, when the OPA is used as comparator during calibration. 0: Non-inverting < inverting 1: Non-inverting > inverting
Bit 29	**TSTREF:** 软件置位或清零。 0: VREFOPA 输出到 ADC 1: VREFOPA 不输出到 ADC
Bits 28:24	**TRIMOFFSETN:** Offset trimming value (NMOS)
Bits 23:19	**TRIMOFFSETP:** Offset trimming value (PMOS)
Bit 18	**USER_TRIM:** User trimming enable 0: User trimming disabled. 1: User trimming enabled
Bits 17:14	**PGA_GAIN:** Gain in PGA mode 0X00 = Non-inverting gain = 2 0X01 = Non-inverting gain = 4 0X10 = Non-inverting gain = 8 0X11 = Non-inverting gain = 16 1000 = Non-inverting gain = 2 - Internal feedback connected to VM0 1001 = Non-inverting gain = 4 - Internal feedback connected to VM0 1010 = Non-inverting gain = 8 - Internal feedback connected to VM0 1011 = Non-inverting gain = 16 - Internal feedback connected to VM0 1100 = Non-inverting gain = 2 - Internal feedback connected to VM1 1101 = Non-inverting gain = 4 - Internal feedback connected to VM1 1110 = Non-inverting gain = 8 - Internal feedback connected to VM1 1111 = Non-inverting gain = 16 - Internal feedback connected to VM1

Bits 13:12	**CALSEL:** Calibration selection 软件可以置位和清零。It is used to select the offset calibration bus used to generate the internal reference voltage when CALON = 1 or FORCE_VP= 1. 00： VREFOPA = 3.3% VDDA 01： VREFOPA= 10% VDDA 10： VREFOPA= 50% VDDA 11： VREFOPA = 90% VDDA
Bit 11	**CALON:** Calibration mode enable 软件可以置位和清零。It is used to enable the calibration mode connecting VM and VP to the OPA internal reference voltage. 0: Calibration mode disabled. 1: Calibration mode enabled.
Bits 10:9	VPS_SEL: OPA1 Second Non inverting input selection 软件可以置位和清零。该配置，在 TCM_EN=1 时使用。 00: VP0 used as OPA1 non inverting input 01: VP1 used as OPA1 non inverting input 10: VP2 used as OPA1 non inverting input 11: VP3 used as OPA1 non inverting input
Bits 8	VMS_SEL: OPA1 Second inverting input selection 软件可以置位和清零。该配置，在 TCM_EN=1 时使用。 00: VM0 used as OPA1 inverting input 01: VM1 used as OPA1 inverting input
Bits 7	TCM_EN: OPA1 Default and Second Enable 软件可以置位和清零。 在 TCM_EN=0 时使用 VP_SEL/VM_SEL 配置通道。 在 TCM_EN=1 时使用 VPS_SEL/VMS_SEL 配置通道。
Bits 6:5	**VM_SEL:** OPA1 inverting input selection. 软件可以置位和清零。 They are used to select the OPA1 inverting input. 00: VM0 used as OPA1 inverting input 01: VM1 used as OPA1 inverting input 10: Resistor feedback output (PGA mode) 11: follower mode
Bit 4	保留
Bits 3:2	**VP_SEL:** OPA1 Non inverting input selection. 软件可以置位和清零。They are used to select the OPA1 non inverting input. 00: VP0 used as OPA1 non inverting input 01: VP1 used as OPA1 non inverting input 10: VP2 used as OPA1 non inverting input 11: VP3 used as OPA1 non inverting input
Bit 1	**FORCE_VP:** 强制待校验的运放输入 2 端短接。 0: Normal operating mode. Non-inverting input connected to inputs. 1: Calibration mode. Non-inverting input connected to calibration reference voltage.
Bit 0	**OPA1EN:** OPA1 enable. 软件可以置位和清零。It is used to enable the OPA1. 0: OPA1 is disabled 1: OPA1 is enabled

17.4.2 OPA2 control register (OPA2_CSR)

偏移地址 ： 0x3C

复位值 ： 0xXXXX 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LOCK	OUTCAL	TSTREF	TRIMOFFSETN					TRIMOFFSETP					USER_TRIM	PGA_GAIN	
rw	r	rw	rw					rw					rw	rw	rw

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PGA_GAIN		CALSEL		CALON	VPS_SEL		VMS_SEL	TCM_EN	VM_SEL		Res.	VP_SEL		FORCE_VP	OPAEN
rw		rw		rw	rw		rw	rw	rw	rw		rw	rw	rw	rw
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

17.4.3 OPA3 control register (OPA3_CSR)

偏移地址 ： 0x40

复位值: 0xXXXX 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LOCK	OUTCAL	TSTREF	TRIMOFFSETN					TRIMOFFSETP					USER_TRIM	PGA_GAIN	
rw	r	rw	rw					rw					rw	rw	rw
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PGA_GAIN		CALSEL		CALON	VPS_SEL		VMS_SEL	TCM_EN	VM_SEL		Res.	VP_SEL		FORCE_VP	OPAEN
rw		rw		rw	rw		rw	rw	rw	rw		rw	rw	rw	rw
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

18 高级控制定时器 (TIM1)

18.1 TIM1 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。

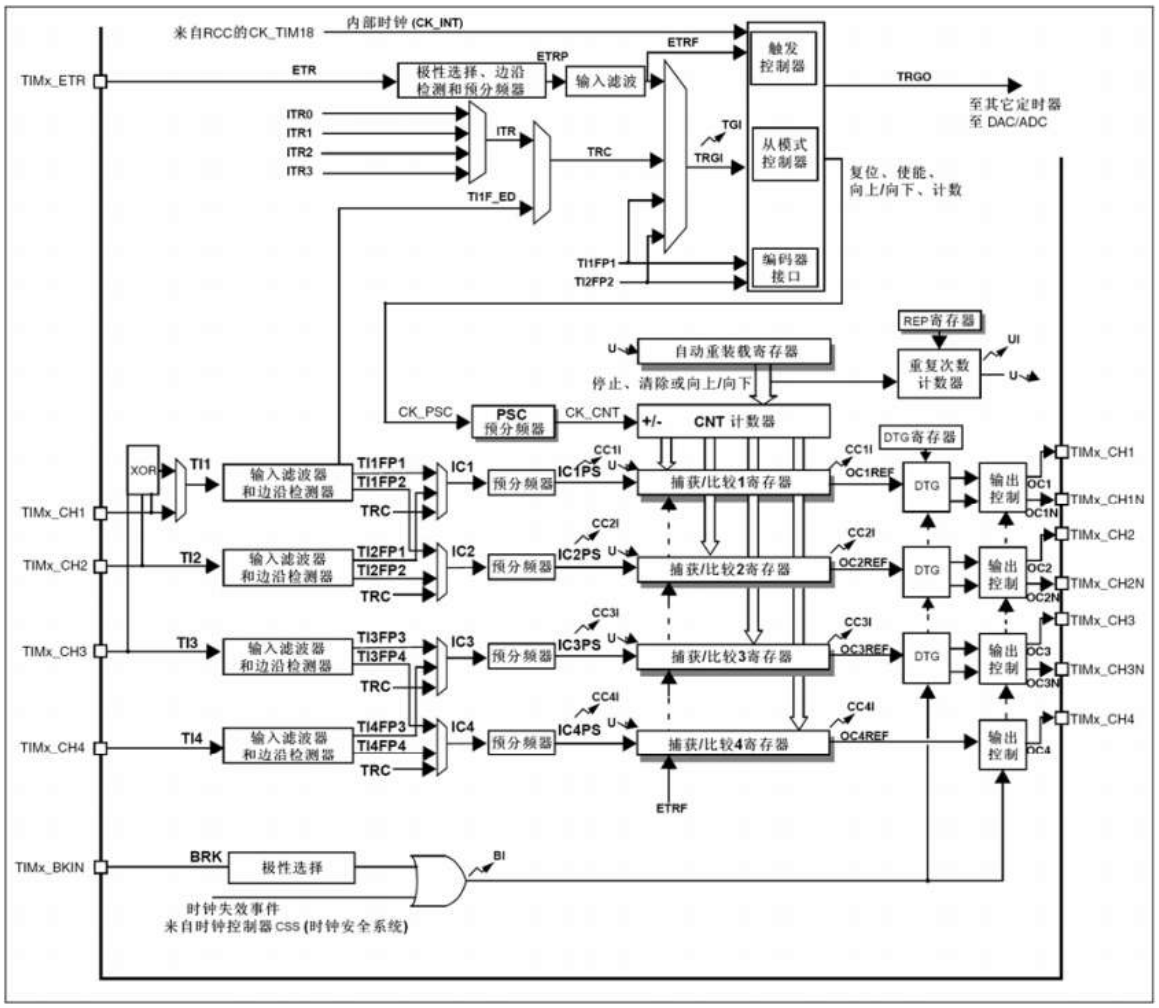
高级控制（TIM1）和其他定时器完全独立，不共享任何资源。

18.2 TIM1 主要特征

TIM1 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 1 高级控制定时器框图



注： Reg *根据控制位的设定，在 U(更新)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*

18.3 TIM1 功能描述

18.3.1 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 2 和图 3 给出了在预分频器运行时，更改计数器参数的例子。

图 2 当预分频器的参数从 1 变到 2 时，计数器的时序图

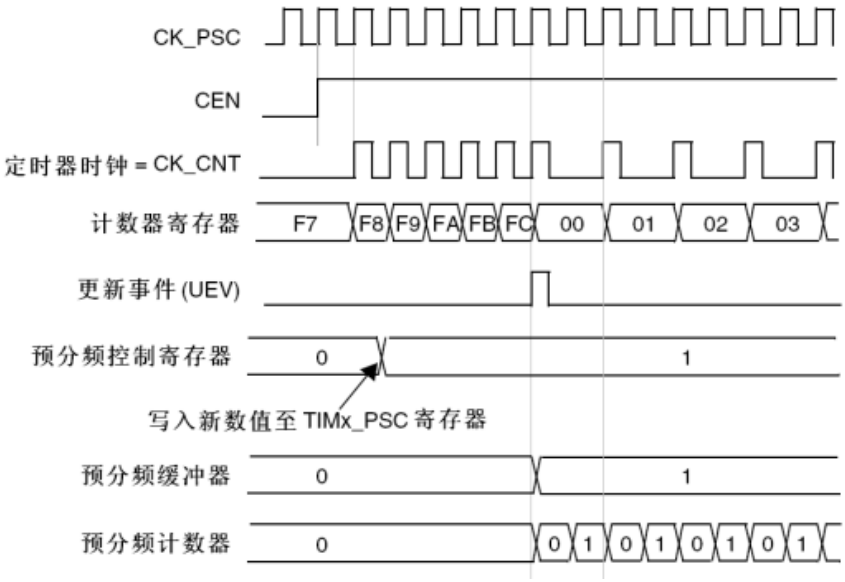
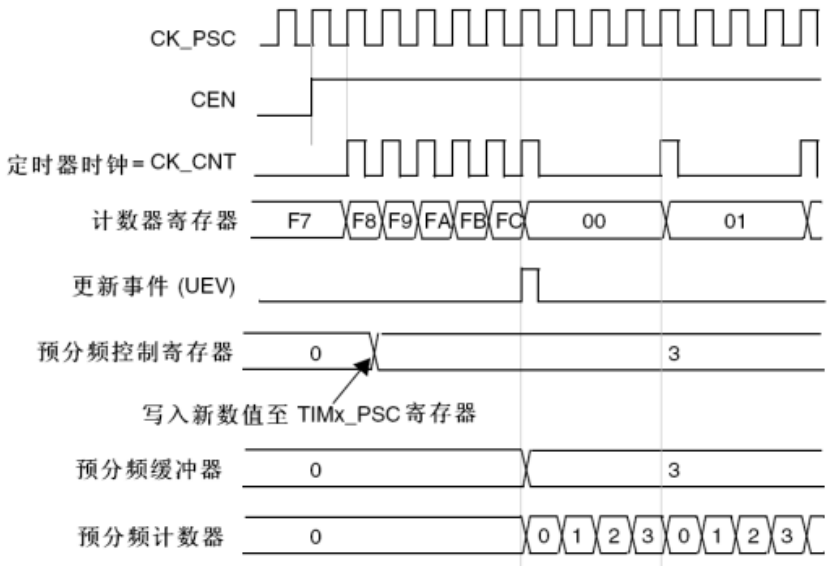


图 3 当预分频器的参数从 1 变到 4 时，计数器的时序图



18.3.2 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(通过软件方式或者使用从模式控制器)设置 UG 位也同样可以产生一个更新事件。

设置 TIMx_CR1 寄存器中的 UDIS 位，可以禁止更新事件；这样可以避免在向预装载寄存器中写入新值时更新影子寄存器。在 UDIS 位被清’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 18-1 计数器时序图，内部时钟分频因子为 1

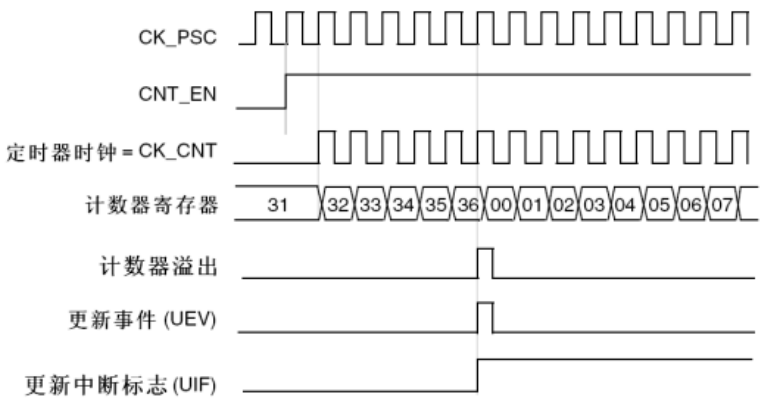


图 18-2 计数器时序图，内部时钟分频因子为 2

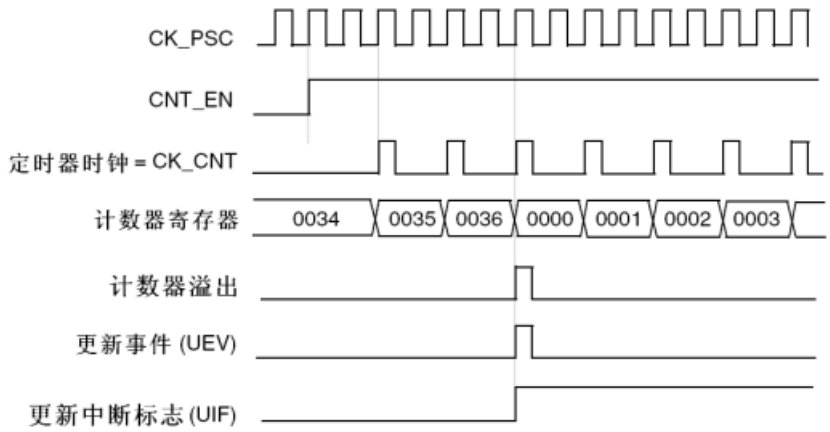


图 18-3 计数器时序图，内部时钟分频因子为 4

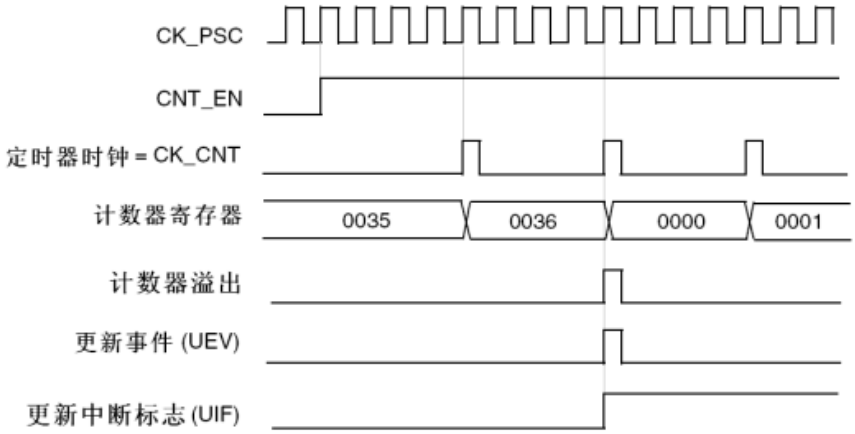


图 18-4 计数器时序图，内部时钟分频因子为 N

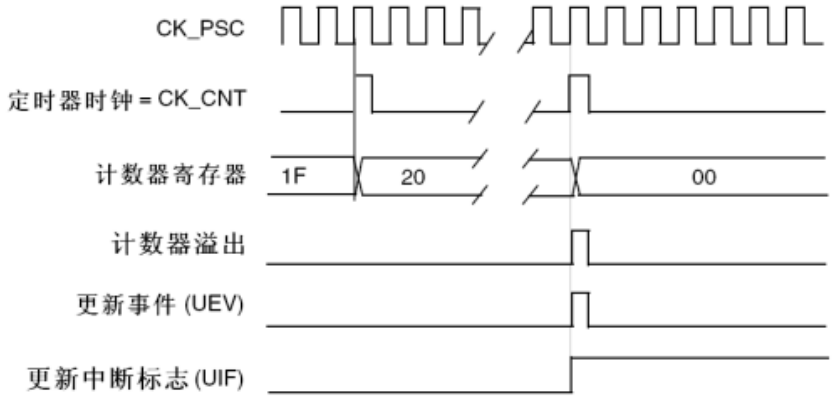


图 18-5 计数器时序图，当 ARPE=0 时的更新事件(TIMx_ARR 没有预装入)

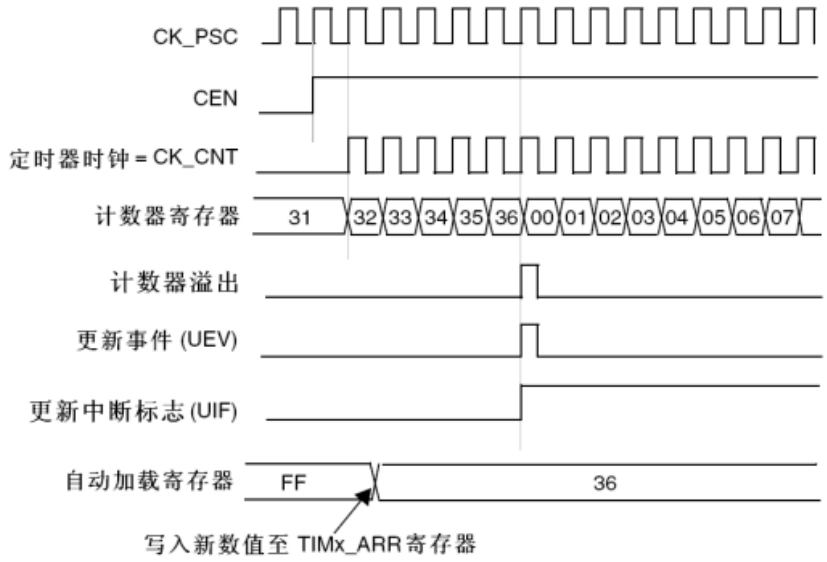
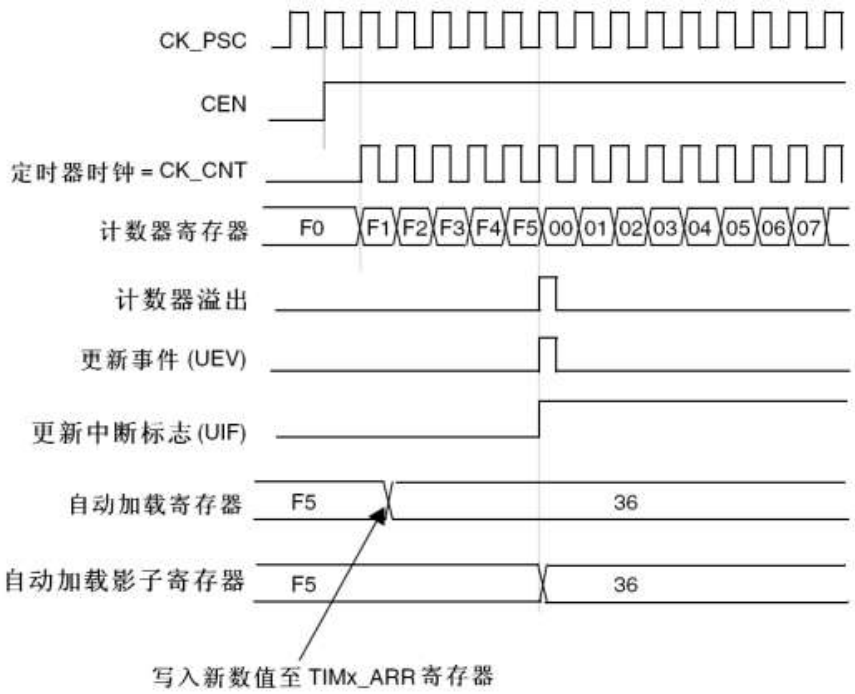


图 18-6 计数器时序图，当 ARPE=1 时的更新事件(预装入了 TIMx_ARR)



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

设置 TIMx_CR1 寄存器的 UDIS 位可以禁止 UEV 事件。这样可以避免向预装载寄存器中写入新值时更新影子寄存器。因此 UDIS 位被清为 0 之前不会产生更新事件。然而，计数器仍会从当前自动加载值重新开始

计数，并且预分频器的计数器重新从 0 开始(但预分频系数不变)。

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

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

- 重复计数器被重置为 TIMx_RCR 寄存器中的内容
- 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 10 计数器时序图，内部时钟分频因子为 1

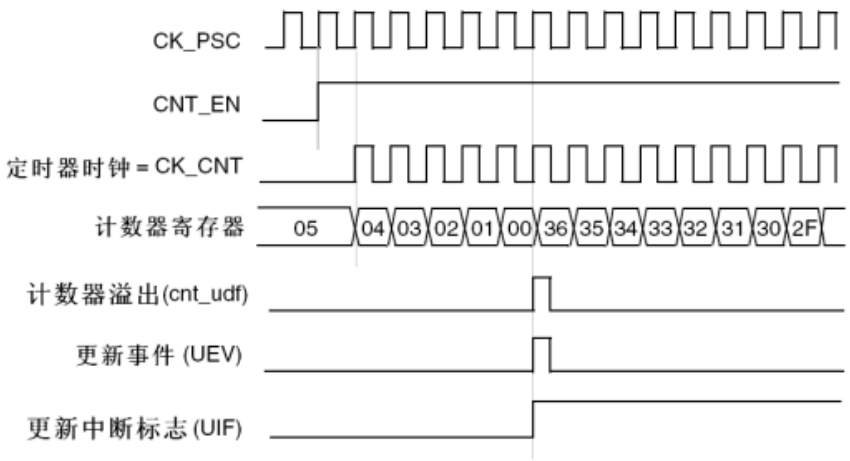


图 11 计数器时序图，内部时钟分频因子为 2

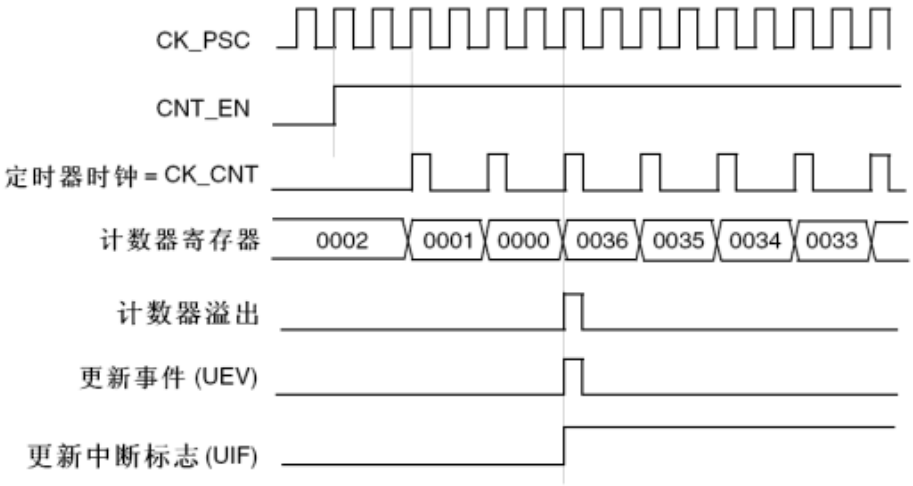


图 12 计数器时序图，内部时钟分频因子为 4

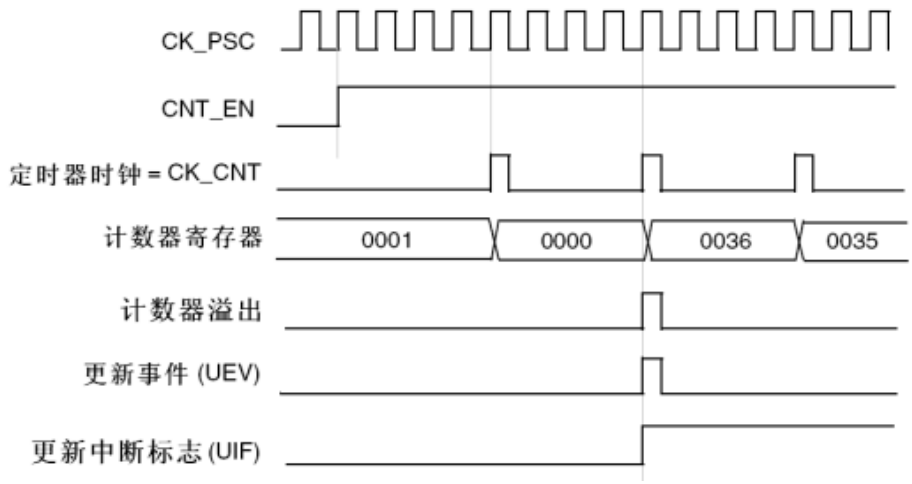


图 13 计数器时序图，内部时钟分频因子为 N

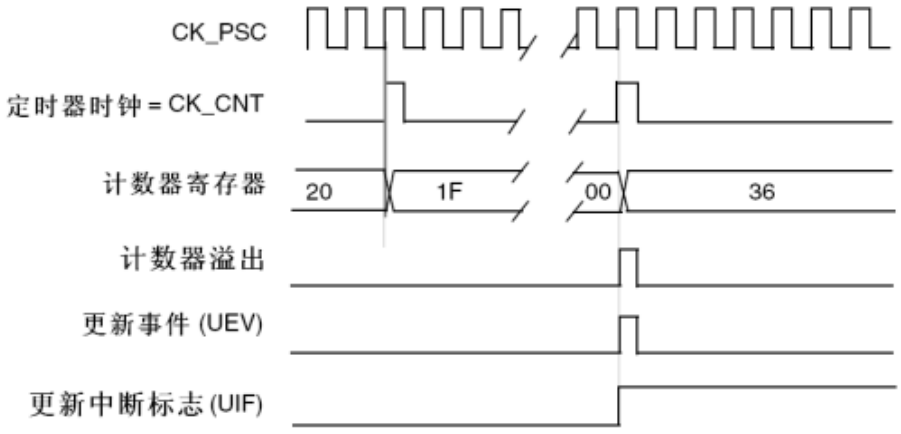
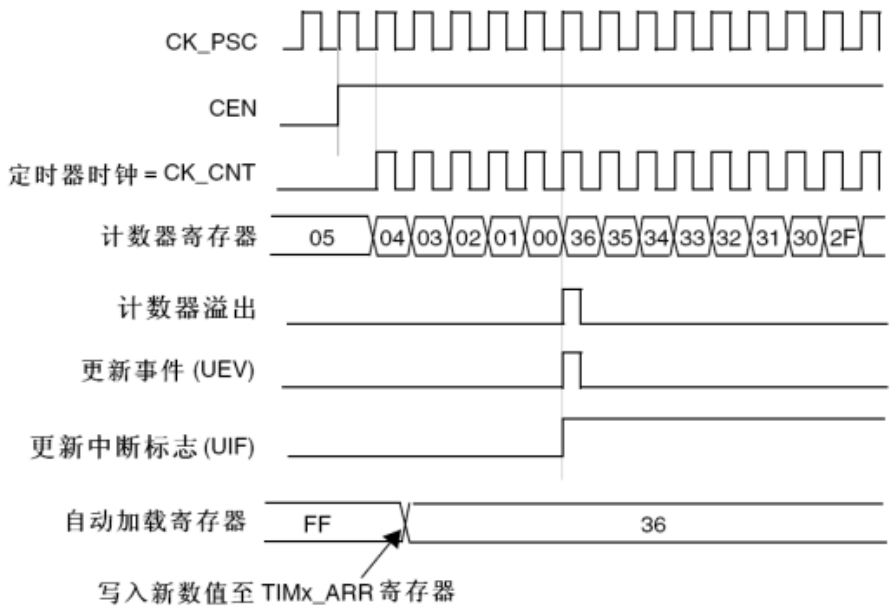


图 14 计数器时序图，当没有使用重复计数器时的更新事件



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。

可以在每次计数上溢和每次计数下溢时产生更新事件；也可以通过(软件或者使用从模式控制器)设置 TIMx_EGR 寄存器中的 UG 位产生更新事件。然后，计数器重新从 0 开始计数，预分频器也重新从 0 开始计数。

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

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

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

- 重复计数器被重置为 TIMx_RCR 寄存器中的内容
- 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 15 计数器时序图，内部时钟分频因子为 1， TIMx_ARR=0x6（这里使用了中心对齐模式 1）

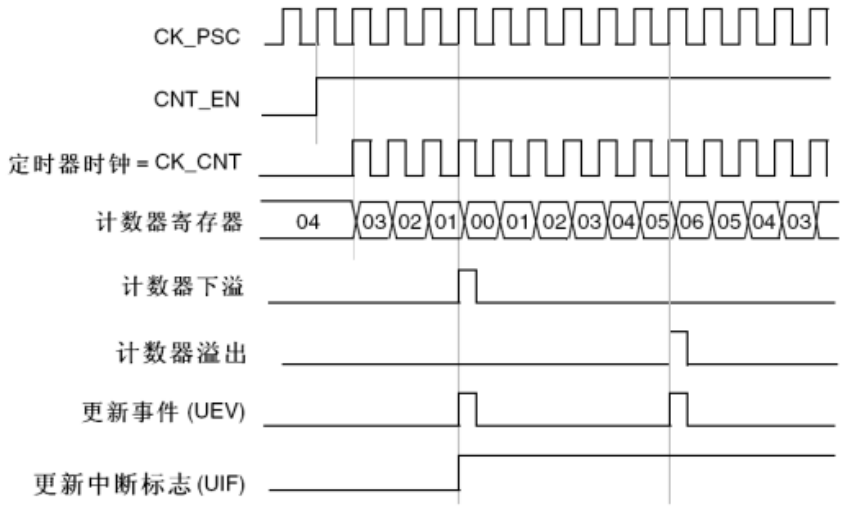


图 16 计数器时序图，内部时钟分频因子为 2（这里使用了中心对齐模式 1）

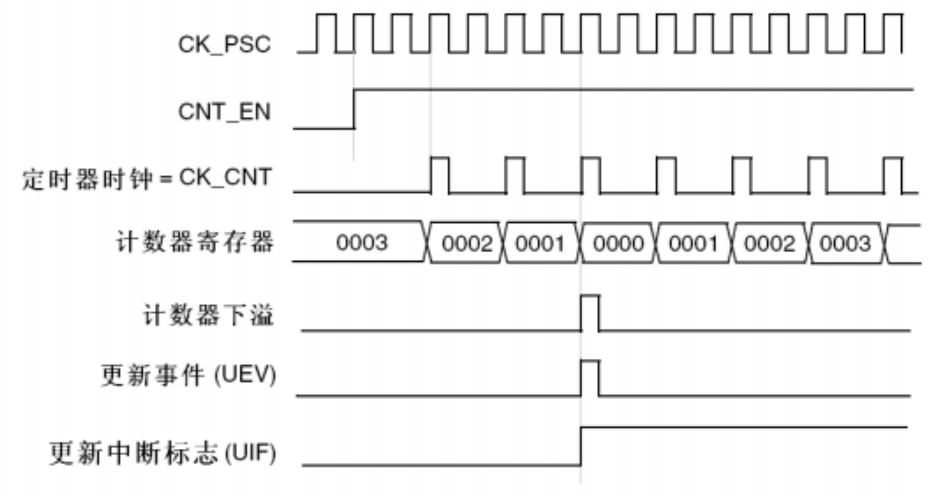


图 17 计数器时序图，内部时钟分频因子为 4， TIMx_ARR=0x36（这里使用了中心对齐模式 2 或 3）

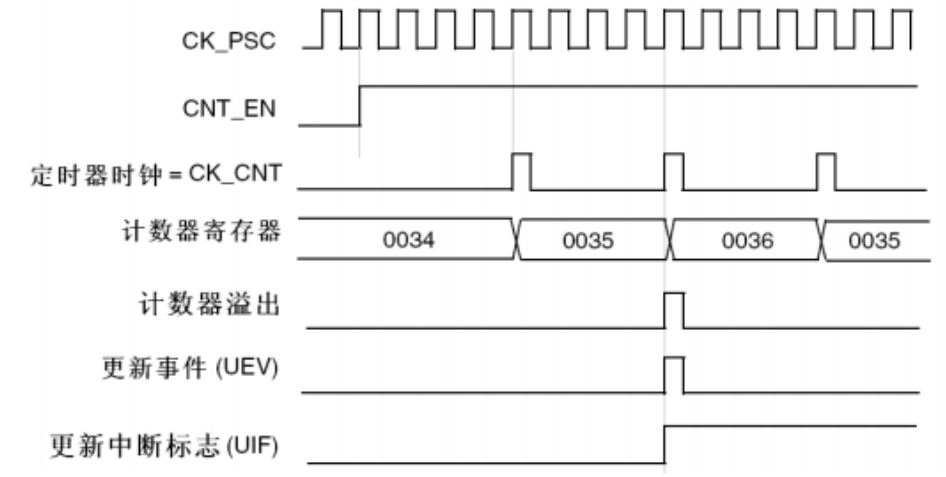


图 18 计数器时序图，内部时钟分频因子为 N

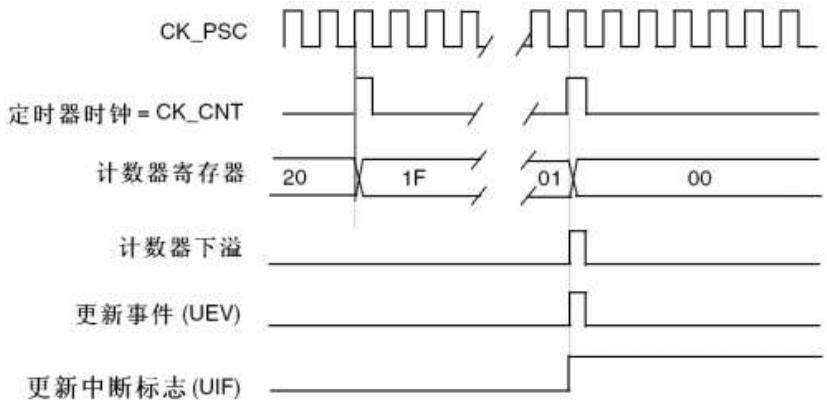


图 19 计数器时序图，ARPE=1 时的更新事件(计数器下溢)

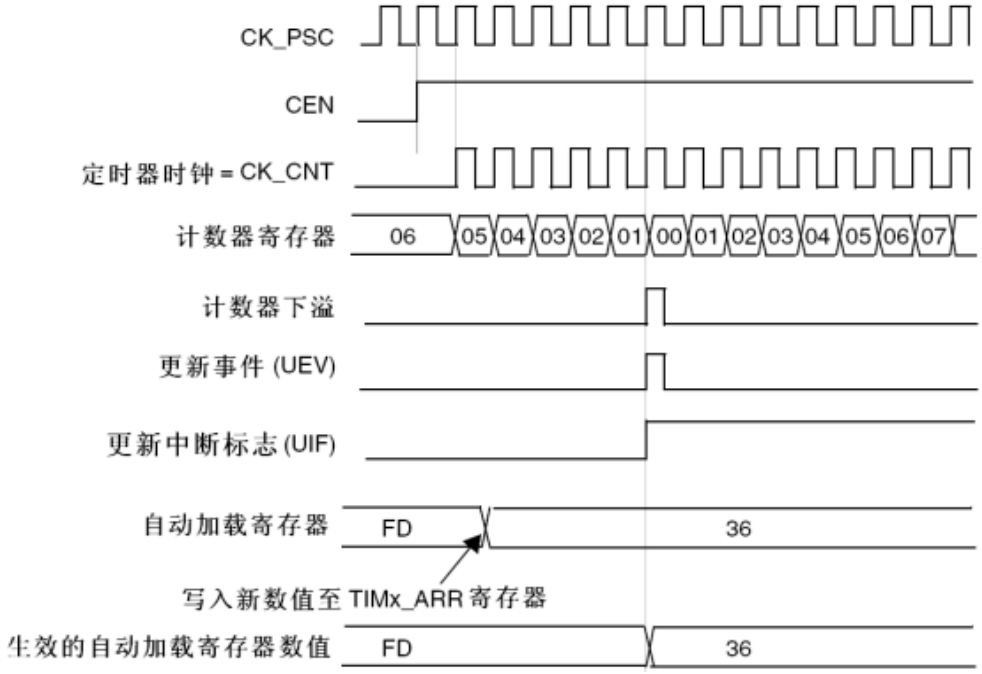
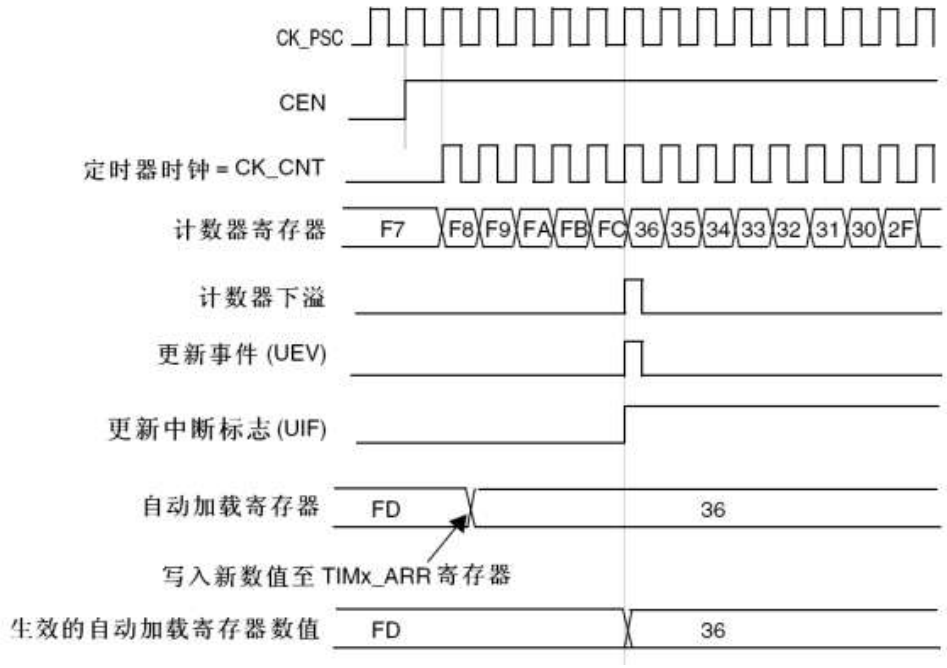


图 20 计数器时序图，ARPE=1 时的更新事件(计数器溢出)



18.3.3 重复计数器

“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。

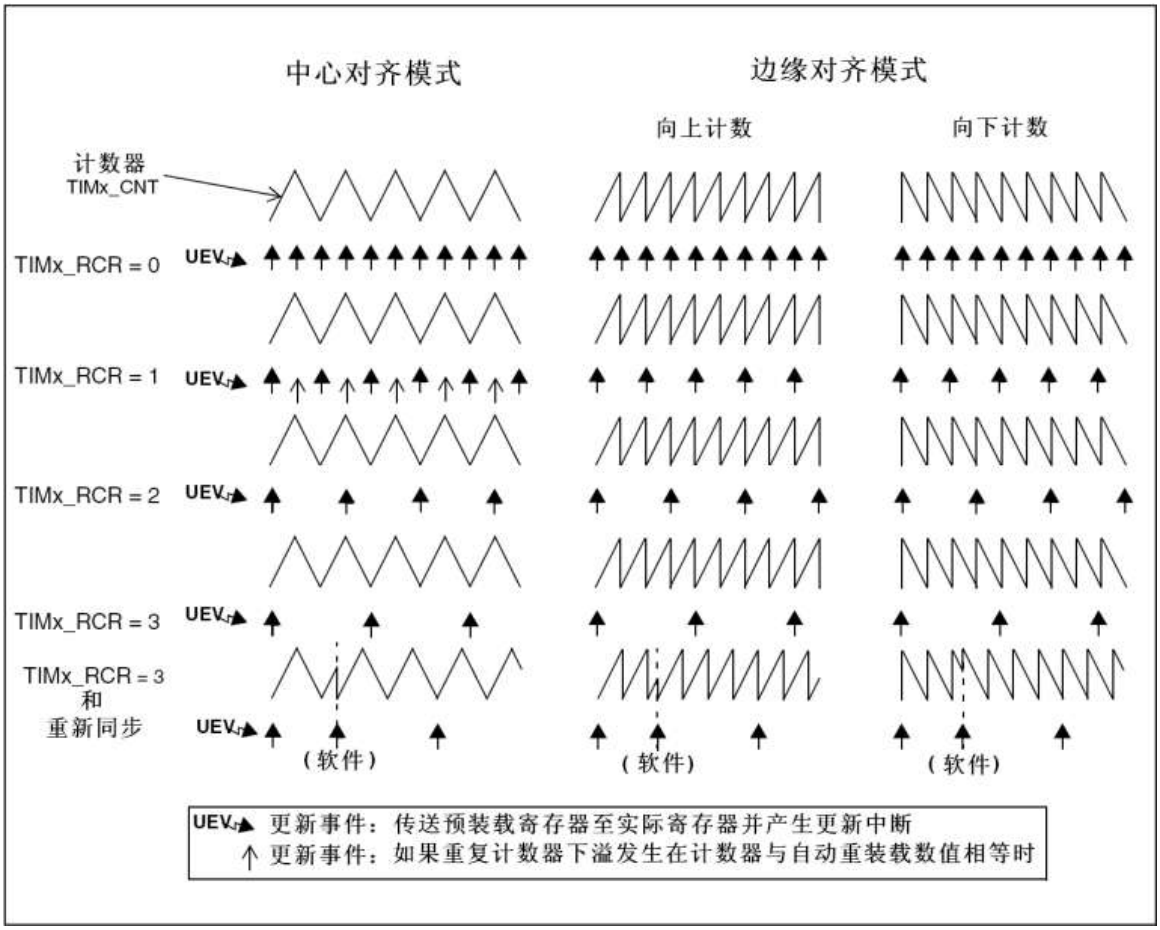
这意味着在每 N 次计数上溢或下溢时，数据从预装载寄存器传输到影子寄存器(TIMx_ARR 自动重载入寄存器， TIMx_PSC 预装载寄存器，还有在比较模式下的捕获/比较寄存器 TIMx_CCRx)， N 是 TIMx_RCR 重复计数寄存器中的值。

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 2xTck。

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 70)。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 21 不同模式下更新速率的例子，及 TIMx_RCR 的寄存器设置



18.3.4 时钟选择

计数器时钟可由下列时钟源提供：

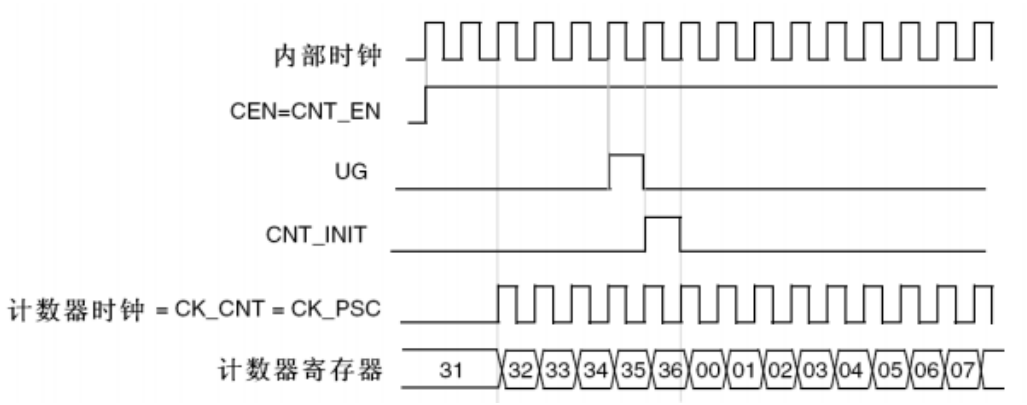
- 内部时钟(CK_INT)
- 外部时钟模式 1：外部输入引脚
- 外部时钟模式 2：外部触发输入 ETR
- 内部触发输入(ITRx)：使用一个定时器作为另一个定时器的预分频器。如可以配置一个定时器 Timer1 而作为另一个定时器 Timer2 的预分频器。详见下一章。

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)。只要 CEN 位被写成’1’，预分频器的时钟就由内部时钟 CK_INT 提供。

下图显示控制电路和向上计数器在一般模式下，不带预分频器时的操作

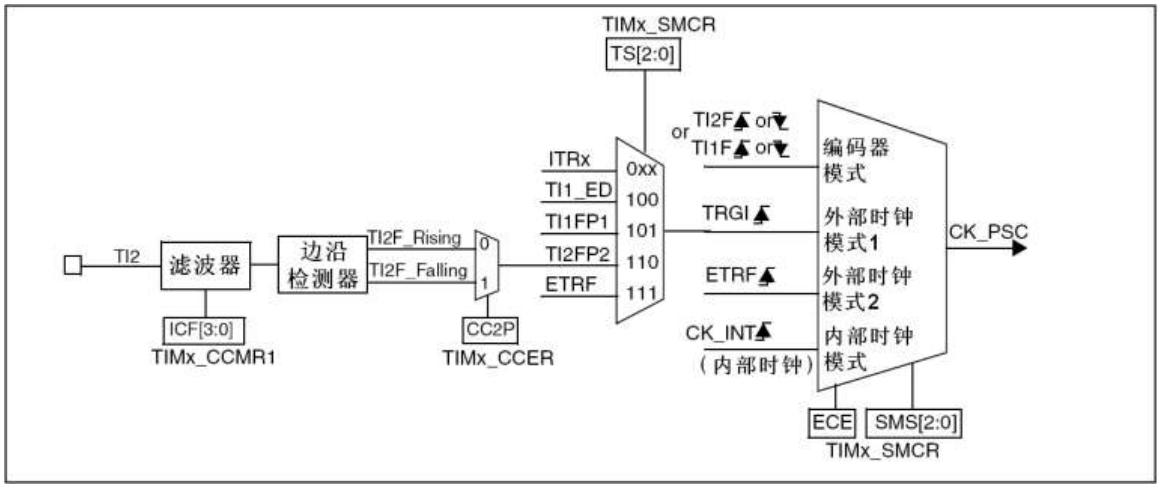
图 22 一般模式下的控制电路，内部时钟分频因子为 1



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 23 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：

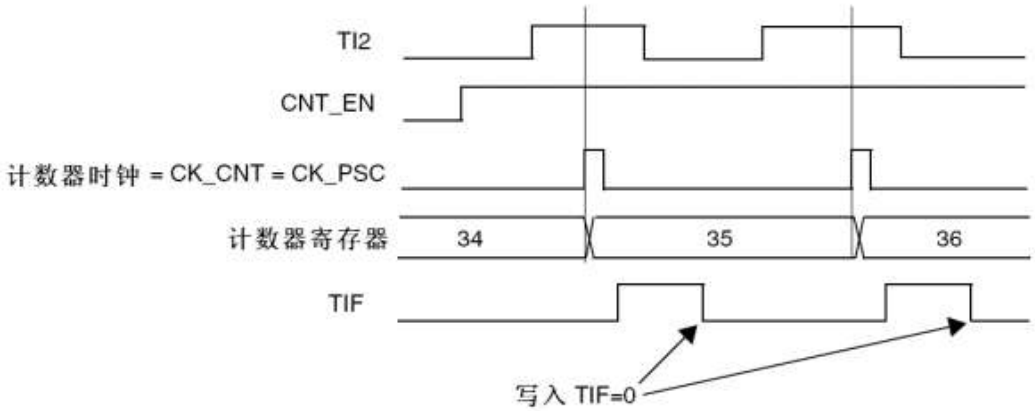
- 配置 TIMx_CCMR1 寄存器 CC2S=01，配置通道 2 检测 TI2 输入的上升沿
- 配置 TIMx_CCMR1 寄存器的 IC2F[3:0]，选择输入滤波器带宽(如果不需要滤波器，保持 IC2F=0000)
- 配置 TIMx_CCER 寄存器的 CC2P=0，选定上升沿极性
- 配置 TIMx_SMCR 寄存器的 SMS=111，选择定时器外部时钟模式 1
- 配置 TIMx_SMCR 寄存器中的 TS=110，选定 TI2 作为触发输入源
- 设置 TIMx_CR1 寄存器的 CEN=1，启动计数器

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 24 外部时钟模式 1 下的控制电路



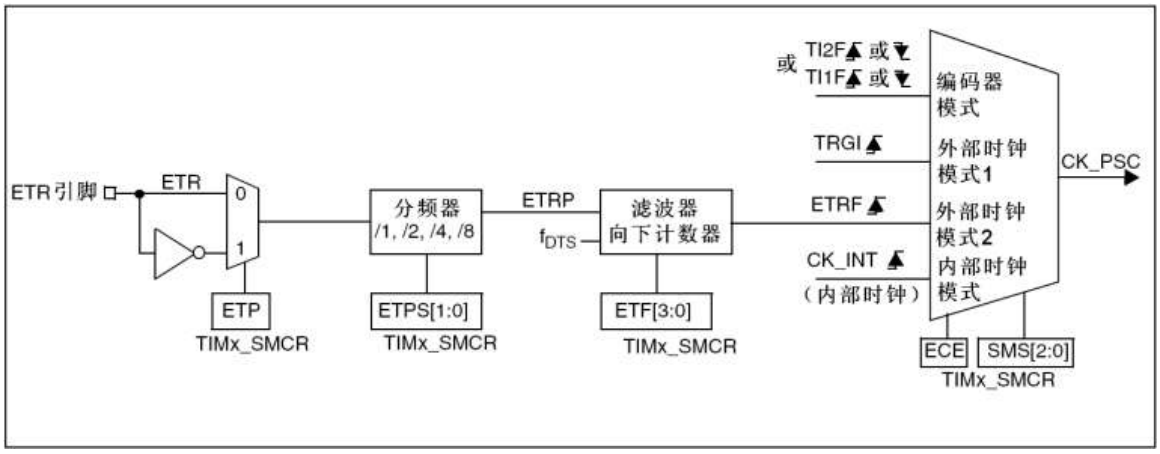
外部时钟源模式 2

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框图

图 25 外部触发输入框图

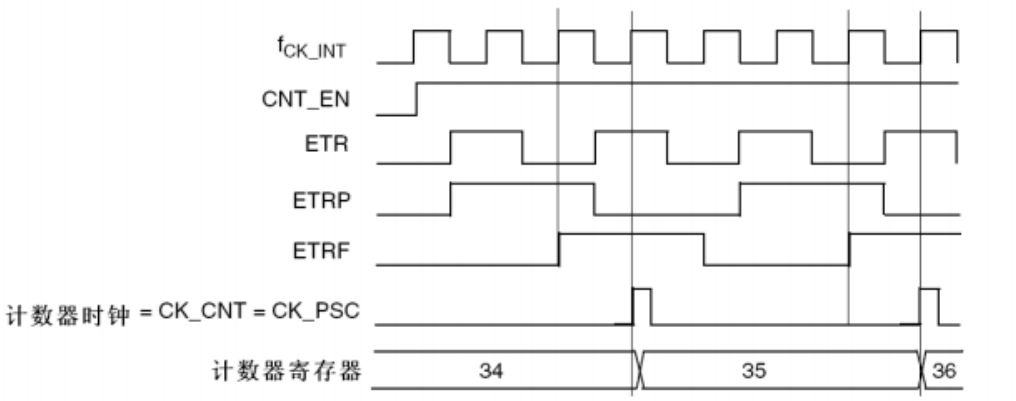


例如，要配置在 ETR 下每 2 个上升沿计数一次的向上计数器，使用下列步骤：

- 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 26 外部时钟模式 2 下的控制电路

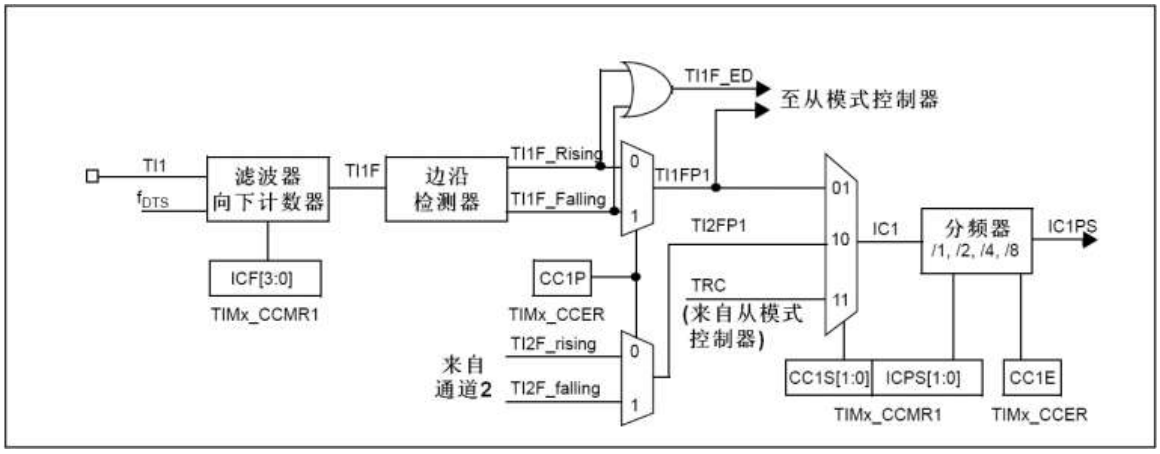


18.3.5 捕获/比较通道

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)。

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 27 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 28 捕获/比较通道 1 的主电路

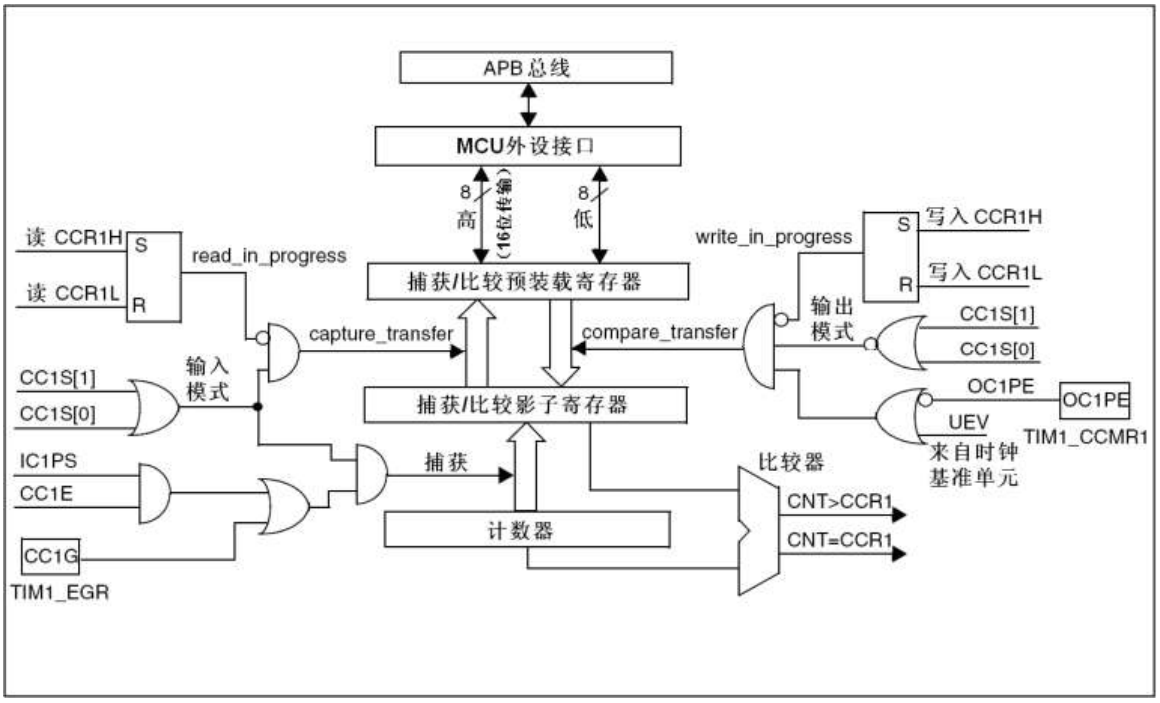


图 29 捕获/比较通道的输出部分(通道 1 至 3)

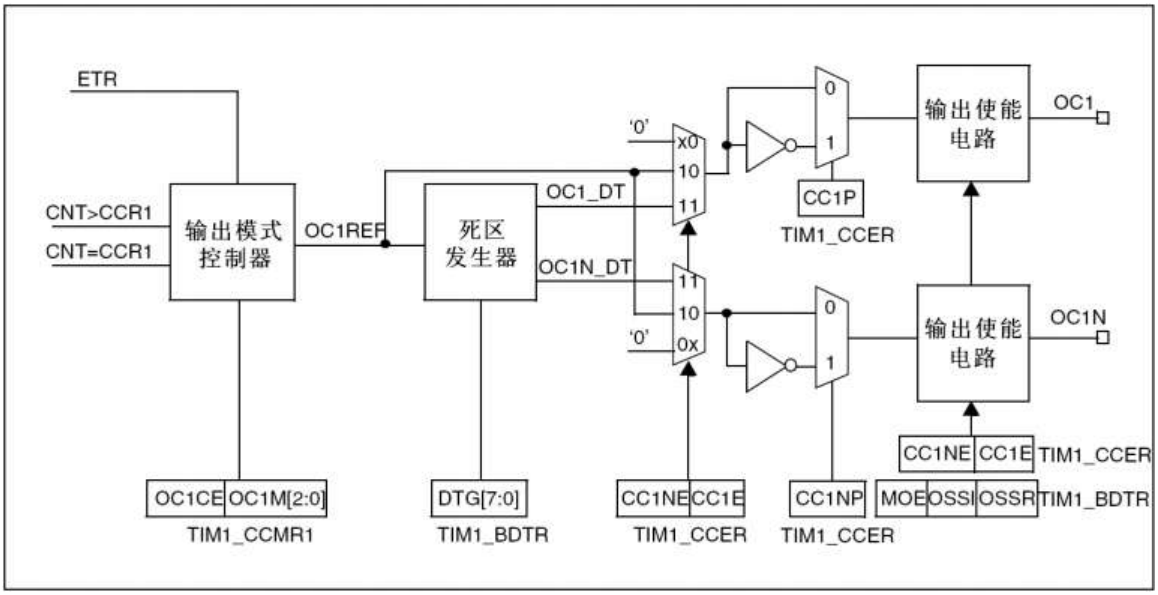
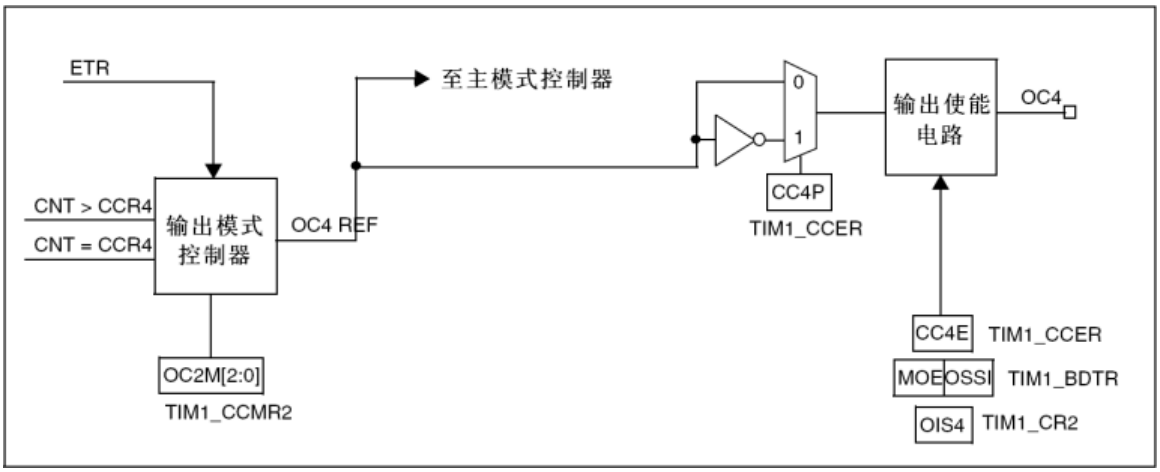


图 30 捕获/比较通道的输出部分(通道 4)



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

在捕获模式下，捕获发生在影子寄存器上，然后再复制到预装载寄存器中。

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

18.3.6 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，只要 CC1S 不为’ 00’，通道被配置为输入，并且 TIMx_CCR1 寄存器变为只读。
- 根据输入信号的特点，配置输入滤波器为所需的带宽(即输入为 TIx 时，输入滤波器控制位是 TIMx_CCMRx 寄存器中的 ICxF 位)。假设输入信号在最多 5 个内部时钟周期的时间内抖动，我们须配置滤波器的带宽长于 5 个时钟周期；因此我们可以(以 fDTS 频率)连续采样 8 次，以确认在 TI1 上一次真实的边沿变换，即在 TIMx_CCMR1 寄存器中写入 IC1F=0011。
- 选择 TI1 通道的有效转换边沿，在 TIMx_CCER 寄存器中写入 CC1P=0(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：

- 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。

18.3.7 PWM 输入模式

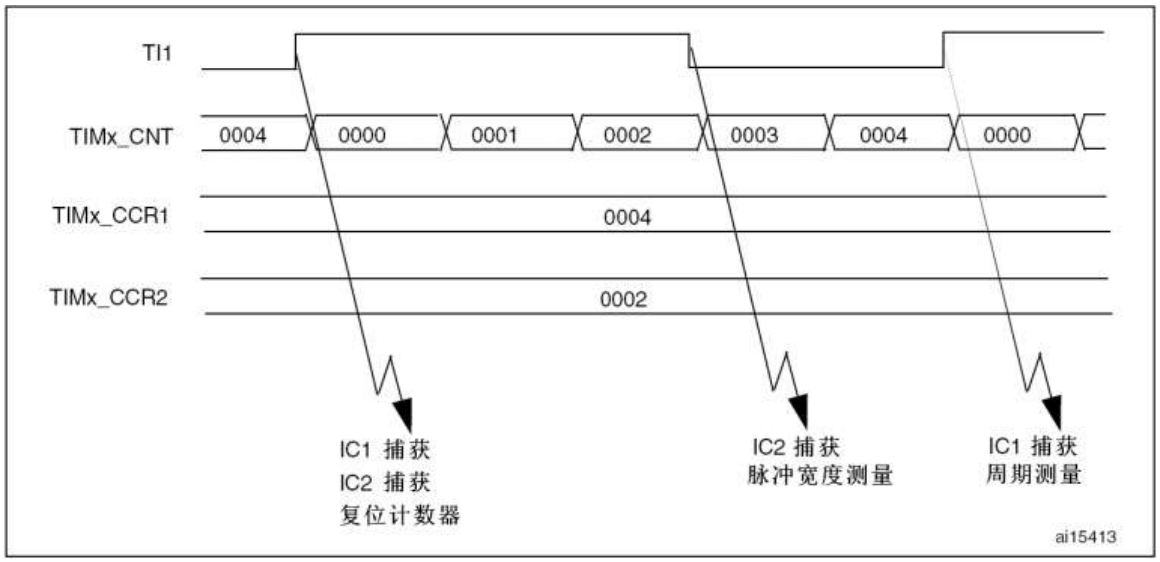
该模式是输入捕获模式的一个特例，除下列区别外，操作与输入捕获模式相同：

- 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 31 PWM 输入模式时序



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

18.3.8 强制输入模式

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。

18.3.9 输出比较模式

此项功能是用来控制一个输出波形，或者指示一段给定的的时间已经到时。

当计数器与捕获/比较寄存器的内容相同时，输出比较功能做如下操作：

● 将输出比较模式(TIMx_CCMRx 寄存器中的 OCxM 位)和输出极性(TIMx_CCER 寄存器中的 CCxP 位)定义的值输出到对应的引脚上。在比较匹配时，输出引脚可以保持它的电平(OCxM=000)、被设置成有效电平(OCxM=001)、被设置成无效电平(OCxM=010)或进行翻转(OCxM=011)。

● 设置中断状态寄存器中的标志位(TIMx_SR 寄存器中的 CCxIF 位)。

● 若设置了相应的中断屏蔽(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 的输出引脚，设置 OCxM=011

— 置 OCxPE = 0 禁用预装载寄存器

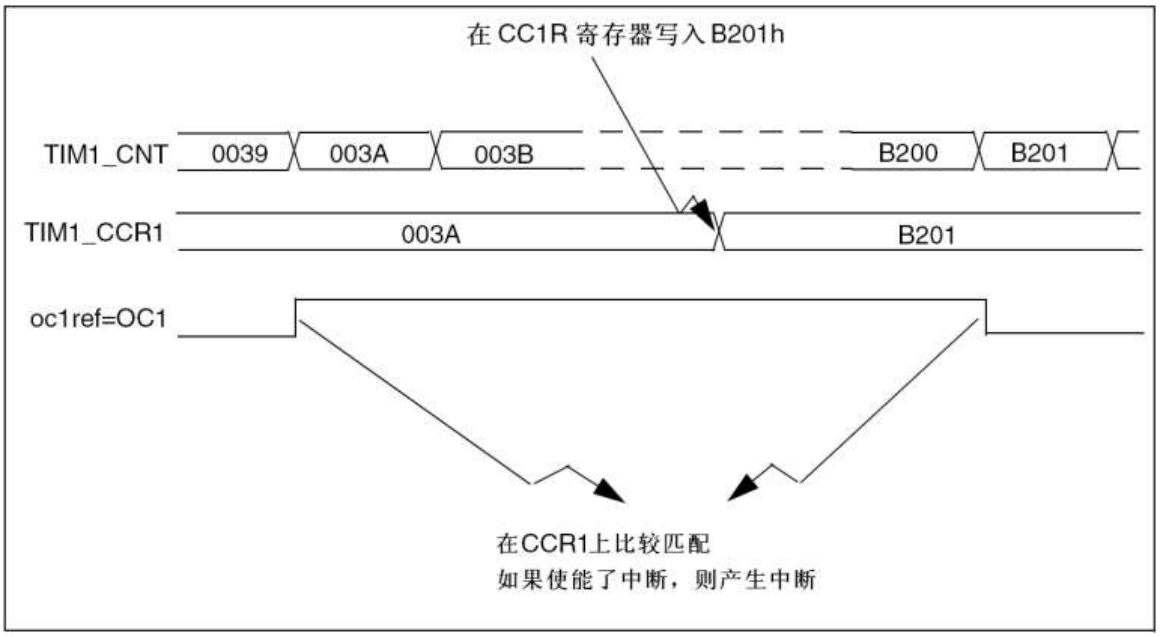
— 置 CCxP = 0 选择极性为高电平有效

— 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 32 输出比较模式，翻转 OC1



18.3.10 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， (在向上计数或中心对称模式中)使能自动重装载的预装载寄存器。

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

OCx 的极性可以通过软件在 TIMx_CCER 寄存器中的 CCxP 位设置，它可以设置为高电平有效或低电平有效。 OCx 的输出使能通过(TIMx_CCER 和 TIMx_BDTR 寄存器中)CCxE、 CCxNE、MOE、 OSSI 和 OSSR 位的组合控制。详见 TIMx_CCER 寄存器的描述。

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

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

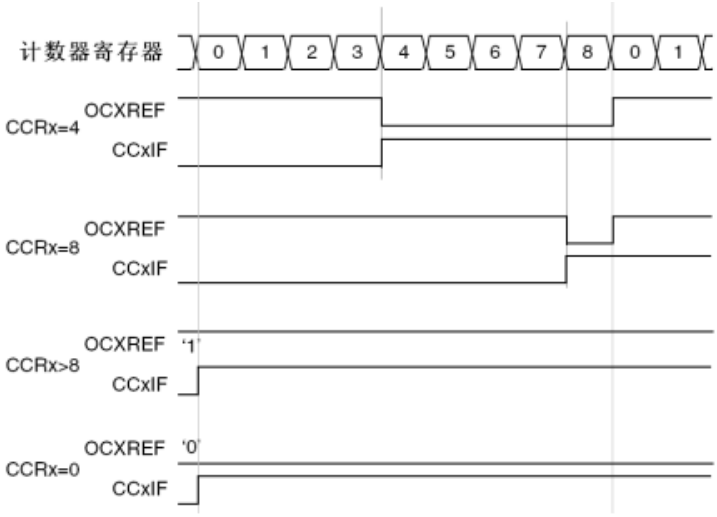
PWM 边沿对齐模式

- 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。

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

图 33 边沿对齐的 PWM 波形(ARR=8)



- 向下计数的配置

当 TIMx_CR1 寄存器的 DIR 位为高时执行向下计数。

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。

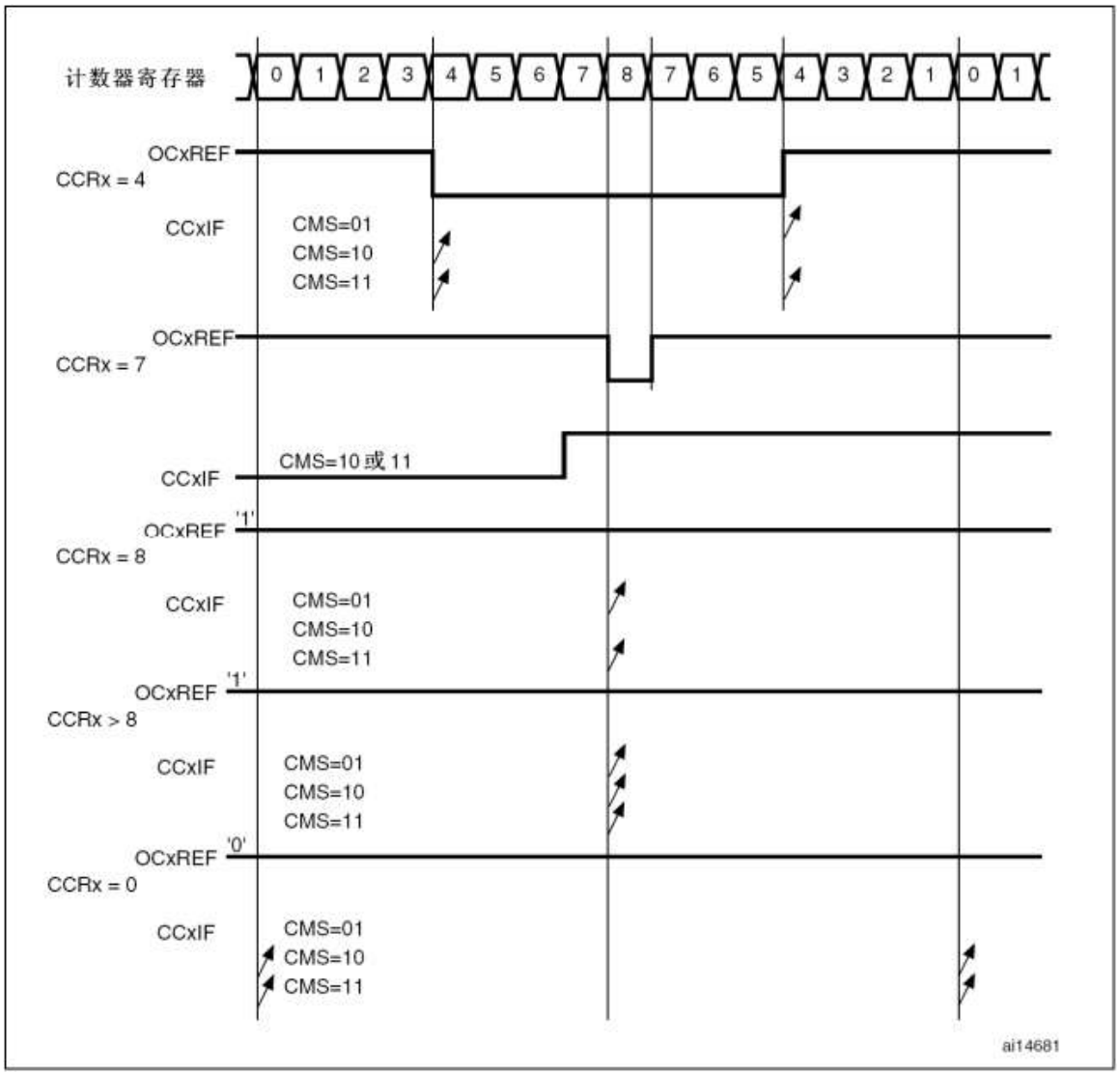
PWM 中央对齐模式

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。

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

18.3.11 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 75 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 35 带死区插入的互补输出

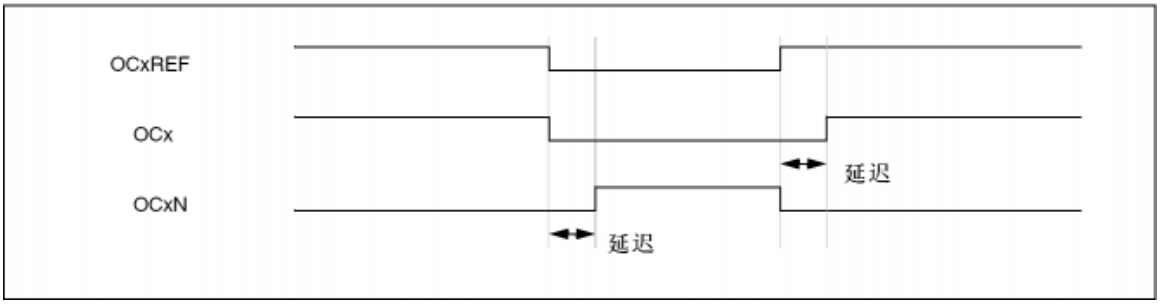


图 36 死区波形延迟大于负脉冲

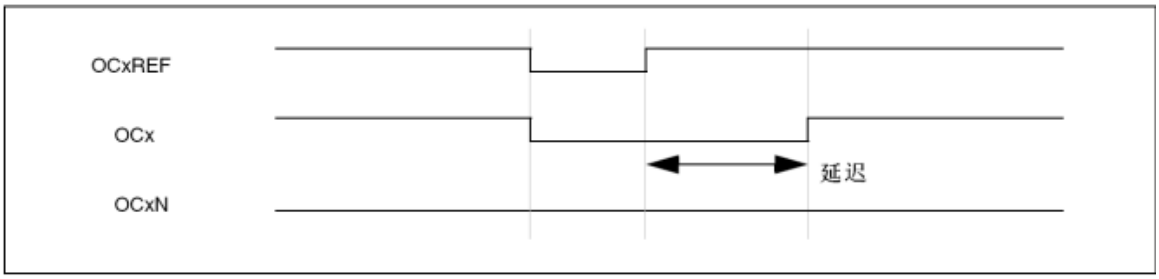
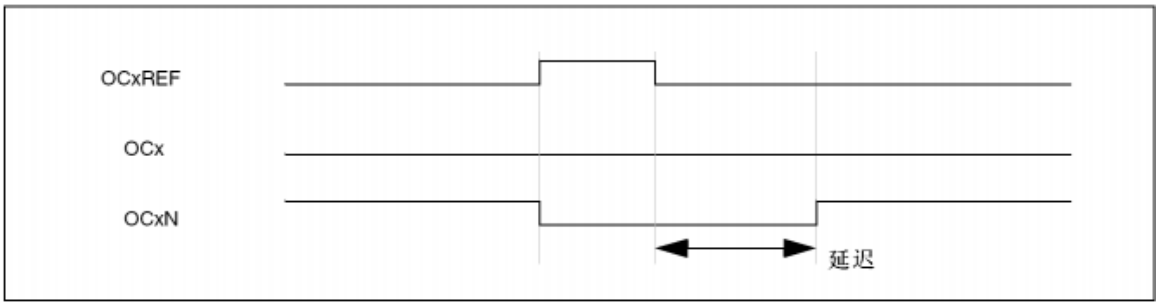


图 37 死区波形延迟大于正脉冲



每一个通道的死区延时都是相同的，是由 TIMx_BDTR 寄存器中的 DTG 位编程配置。详见 TIM1 刹车和死区寄存器(TIMx_BDTR)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，OCxR

EF 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

18.3.12 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 75 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引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 6.2.7 节时钟安全系统(CSS)。

系统复位后，刹车电路被禁止， MOE 位为低。设置 TIMx_BDTR 寄存器中的 BKE 位可以使能刹车功能，刹车输入信号的极性可以通过配置同一个寄存器中的 BKP 位选择。 BKE 和 BKP 可以同时被修改。当写入 BKE 和 BKP 位时，在真正写入之前会有 1 个 APB 时钟周期的延迟，因此需要等待一个 APB 时钟周期之后，才能正确地读回写入的位。

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

当发生刹车时(在刹车输入端出现选定的电平)，有下述动作：

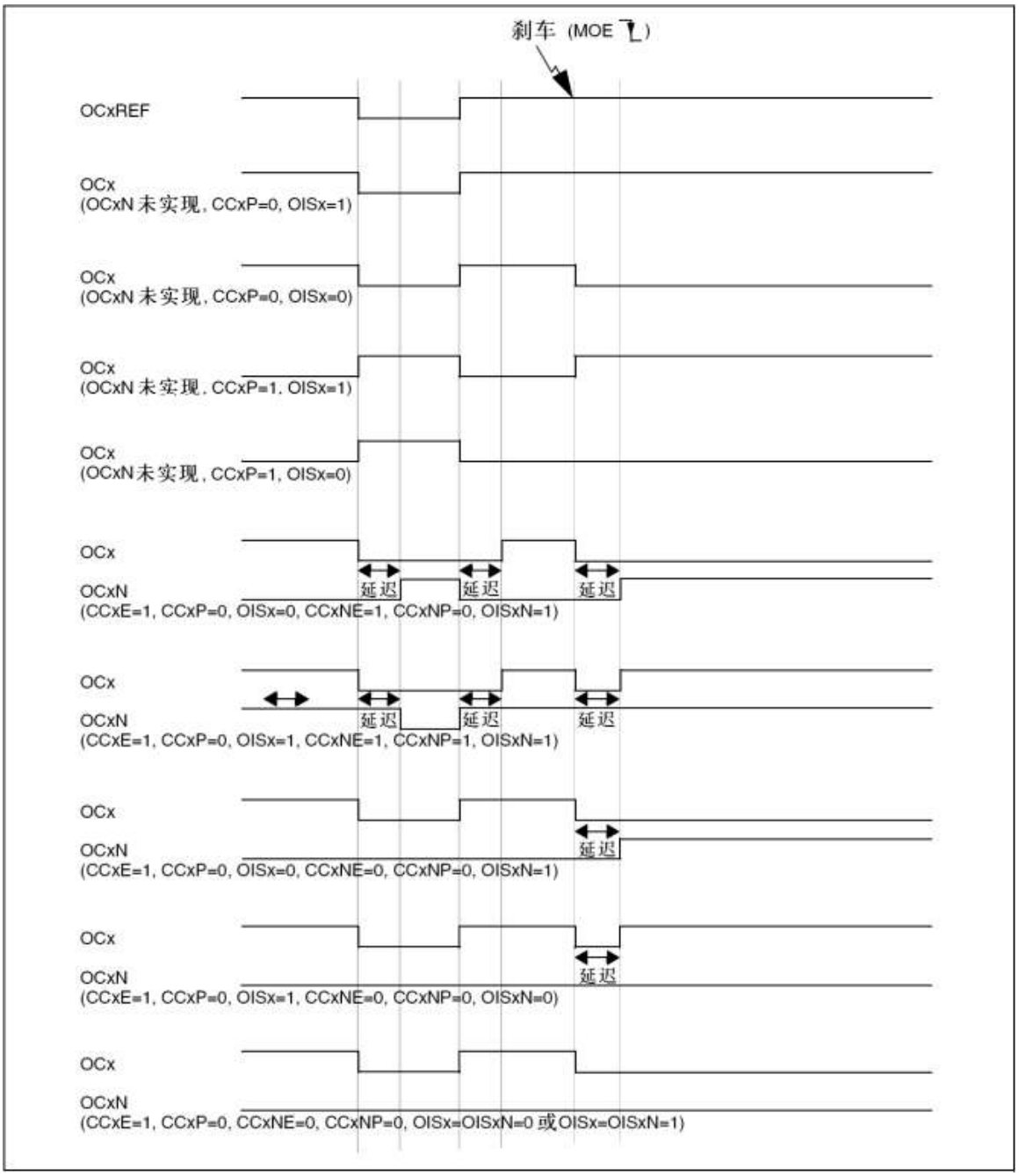
- MOE 位被异步地清除，将输出置于无效状态、空闲状态或者复位状态(由 OSSI 位选择)。这个特性在 MCU 的振荡器关闭时依然有效。
- 一旦 MOE=0，每一个输出通道输出由 TIMx_CR2 寄存器中的 OISx 位设定的电平。如果 OSSI=0，则定时器释放使能输出，否则使能输出始终为高。
- 当使用互补输出时：
  - 输出首先被置于复位状态即无效的状态(取决于极性)。这是异步操作，即使定时器没有时钟时，此功能也有效。
  - 如果定时器的时钟依然存在，死区生成器将会重新生效，在死区之后根据 OISx 和 OISxN 位指示的电平驱动输出端口。即使在这种情况下， OCx 和 OCxN 也不能被同时驱动到有效的电平。注，因为重新同步 MOE，死区时间比通常情况下长一些(大约 2 个 ck_tim 的时钟周期)。
  - 如果 OSSI=0，定时器释放使能输出，否则保持使能输出；或一旦 CCxE 与 CCxNE 之一变高时，使能输出变为高。
- 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 13.4.18 节 TIM1 和 TIM8 刹车和死区寄存器(TIMx_BDTR)。在 MCU 复位后 LOCK 位只能被修改一次。

下图显示响应刹车的输出实例。

图 38 响应刹车的输出



18.3.13 在外部事件时清除 OCxREF 信号

对于一个给定的通道，设置 TIMx_CCMRx 寄存器中对应的 OCxCE 位为’1’，能够用 ETRF 输入端的

高电平把 OCxREF 信号拉低， OCxREF 信号将保持为低直到发生下一次的更新事件 UEV。

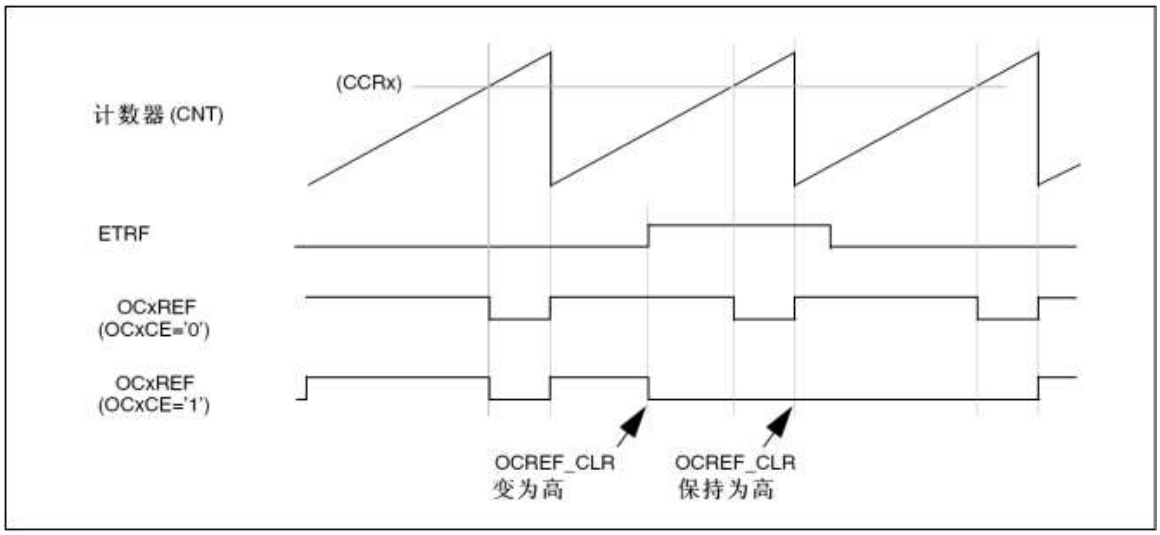
该功能只能用于输出比较和 PWM 模式，而不能用于强置模式。

例如， OCxREF 信号可以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 39 清除 TIMx 的 OCxREF



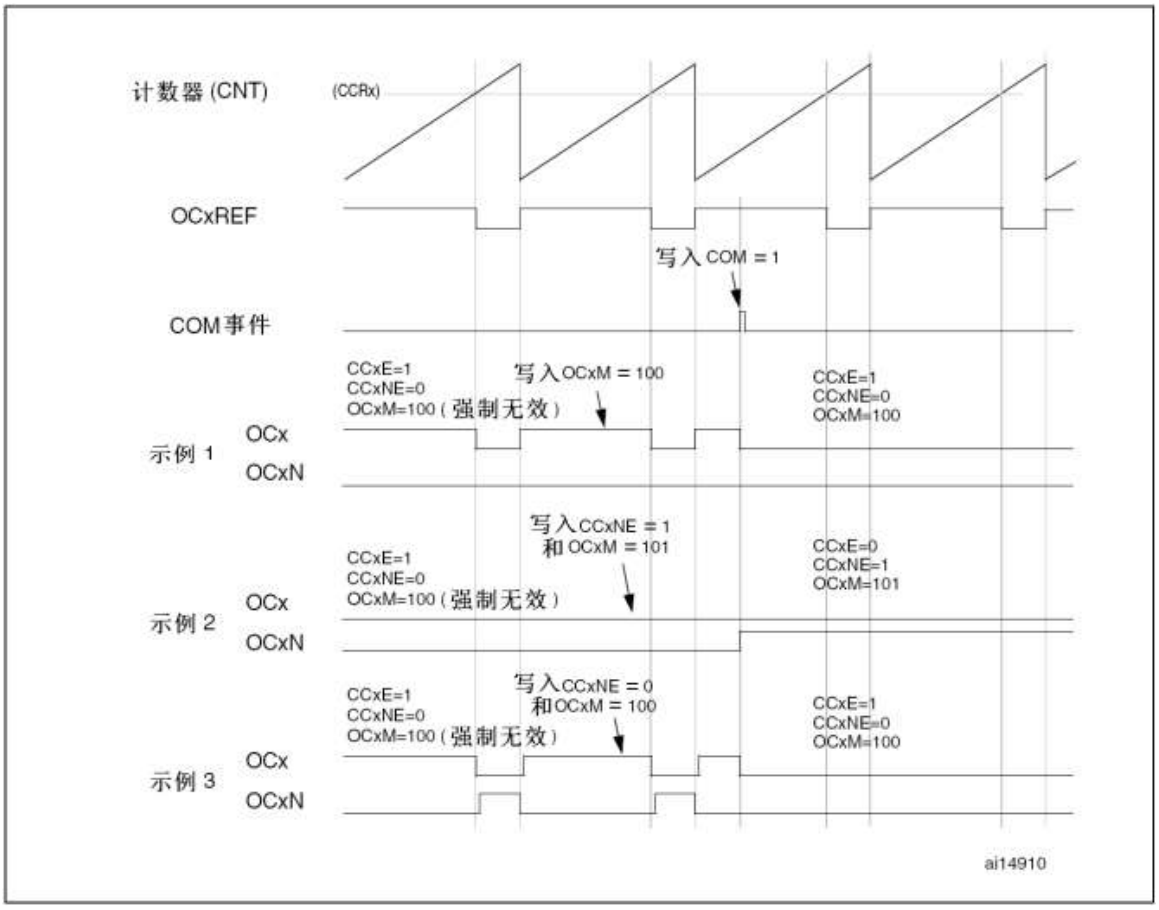
18.3.14 产生六步 PWM 输出

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 40 产生六步 PWM，使用 COM 的例子(OSSR=1)。



18.3.15 单脉冲模式

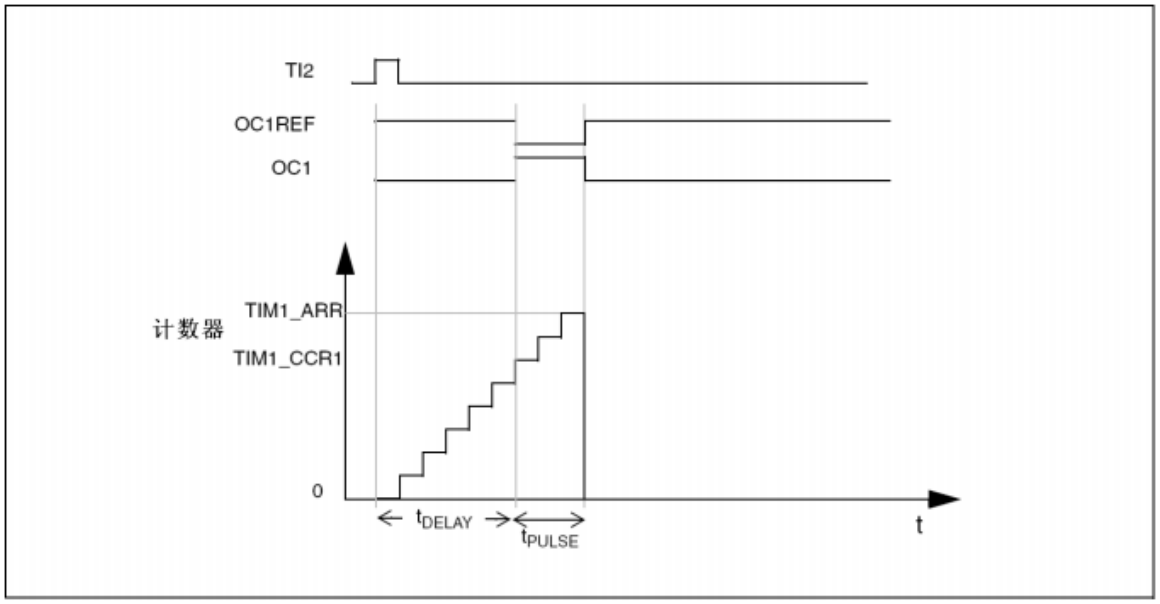
单脉冲模式(OPM)是前述众多模式的一个特例。这种模式允许计数器响应一个激励，并在一个程序可控的延时之后产生一个脉宽可程序控制的脉冲。

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 41 单脉冲模式的例子



例如，你需要在从 TI2 输入脚上检测到一个上升沿开始，延迟 tDELAY 之后，在 OC1 上产生一个长度为 tPULSE 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)

- tDELAY 由 TIMx_CCR1 寄存器中的值定义。
- tPULSE 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 tDELAY。

如果要以最小延时输出波形，可以设置 TIMx_CCMRx 寄存器中的 OCxFE 位；此时 OCxREF(和 OCx)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 PWM 1 和 PWM2 模式时起作用。

18.3.16 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 73，假定计数器已经启动(TIMx_CR1 寄存器中的 CEN=1)，则计数器由每次在 TI1FP1 或 TI2FP2 上的有效跳变驱动。 TI1FP1 和 TI2FP2 是 TI1 和 TI2 在通过输入滤波器和极性控制后的信号；如果没有滤波和变相，则 TI1FP1=TI1，TI2FP2=TI2。根据两个输入信号的跳变顺序，产生了计数脉冲和方向信号。依据两个输入信号的跳变顺序，计数器向上或向下计数，同时硬件对 TIMx_CR1 寄存器的 DIR 位进行相应的设置。不管计数器是依靠 TI1 计数、依靠 TI2 计数或者同时依靠 TI1 和 TI2 计数，在任一输入端(TI1 或者 TI2)的跳变都会重新计算 DIR 位。

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

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

表 42 计数方向与编码器信号的关系

表 18-1 计数方向与编码器信号的关系

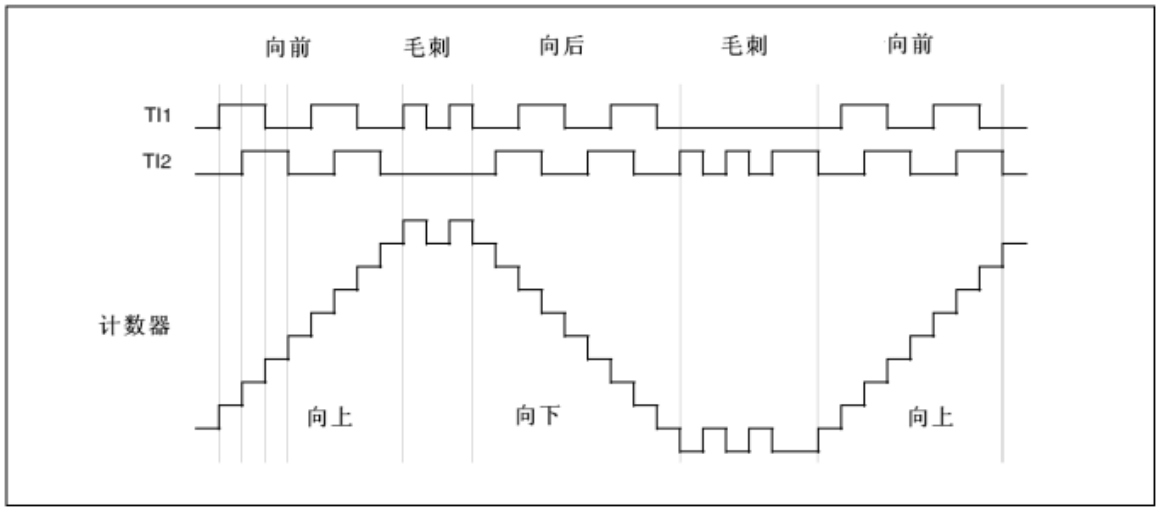
有效边沿	相对信号的电平 (TI1FP1 对应 TI2, TI2FP2 对应 TI1)	TI1FP1 信号		TI2FP2 信号	
		上升	下降	上升	下降
仅在 TI1 计数	高	向下计数	向上计数	不计数	不计数
	低	向上计数	向下计数	不计数	不计数
仅在 TI2 计数	高	不计数	不计数	向上计数	向下计数
	低	不计数	不计数	向下计数	向上计数
在 TI1 和 TI2 上计数	高	向下计数	向上计数	向上计数	向下计数
	低	向上计数	向下计数	向下计数	向上计数

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，一般会使用比较器将编码器的差动输出转换到数字信号，这大大增加了抗噪声干扰能力。编码器输出的第三个信号表示机械零点，可以把它连接到一个外部中断输入并触发一个计数器复位。

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

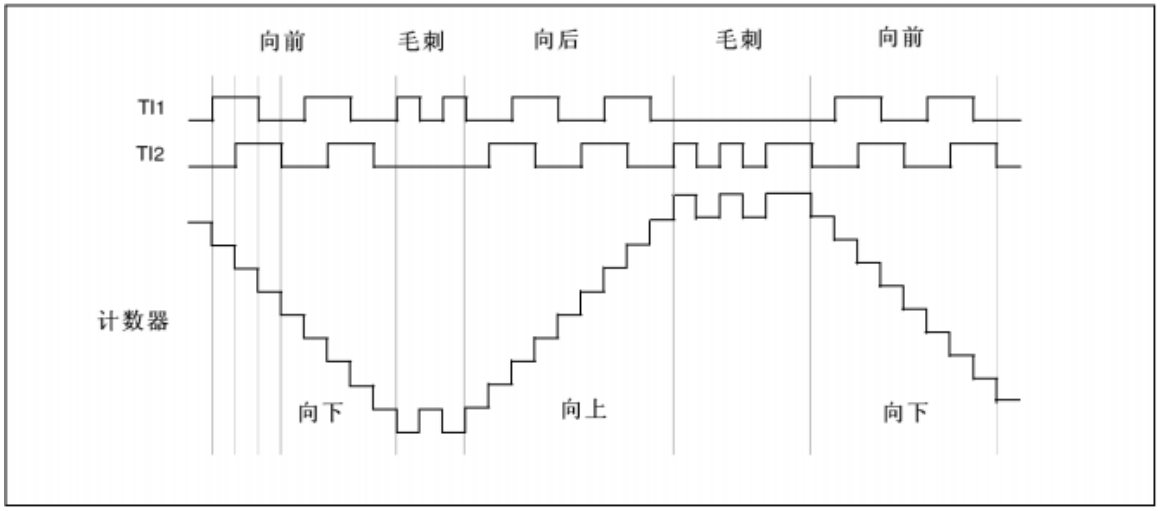
- CC1S=’01’ (TIMx_CCMR1 寄存器， IC1FP1 映射到 TI1)
- CC2S=’01’ (TIMx_CCMR2 寄存器， IC2FP2 映射到 TI2)
- CC1P=’0’ (TIMx_CCER 寄存器， IC1FP1 不反相， IC1FP1=TI1)
- CC2P=’0’ (TIMx_CCER 寄存器， IC2FP2 不反相， IC2FP2=TI2)
- SMS=’011’ (TIMx_SMCR 寄存器，所有的输入均在上升沿和下降沿有效).
- CEN=’1’ (TIMx_CR1 寄存器，计数器使能)

图 18-7 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 18-8 IC1FP1 反相的编码器接口模式实例



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

18.3.17 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 13.3.18 给出了此特性用于连接霍尔传感器的例子。

18.3.18 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 93， 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 76)。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。

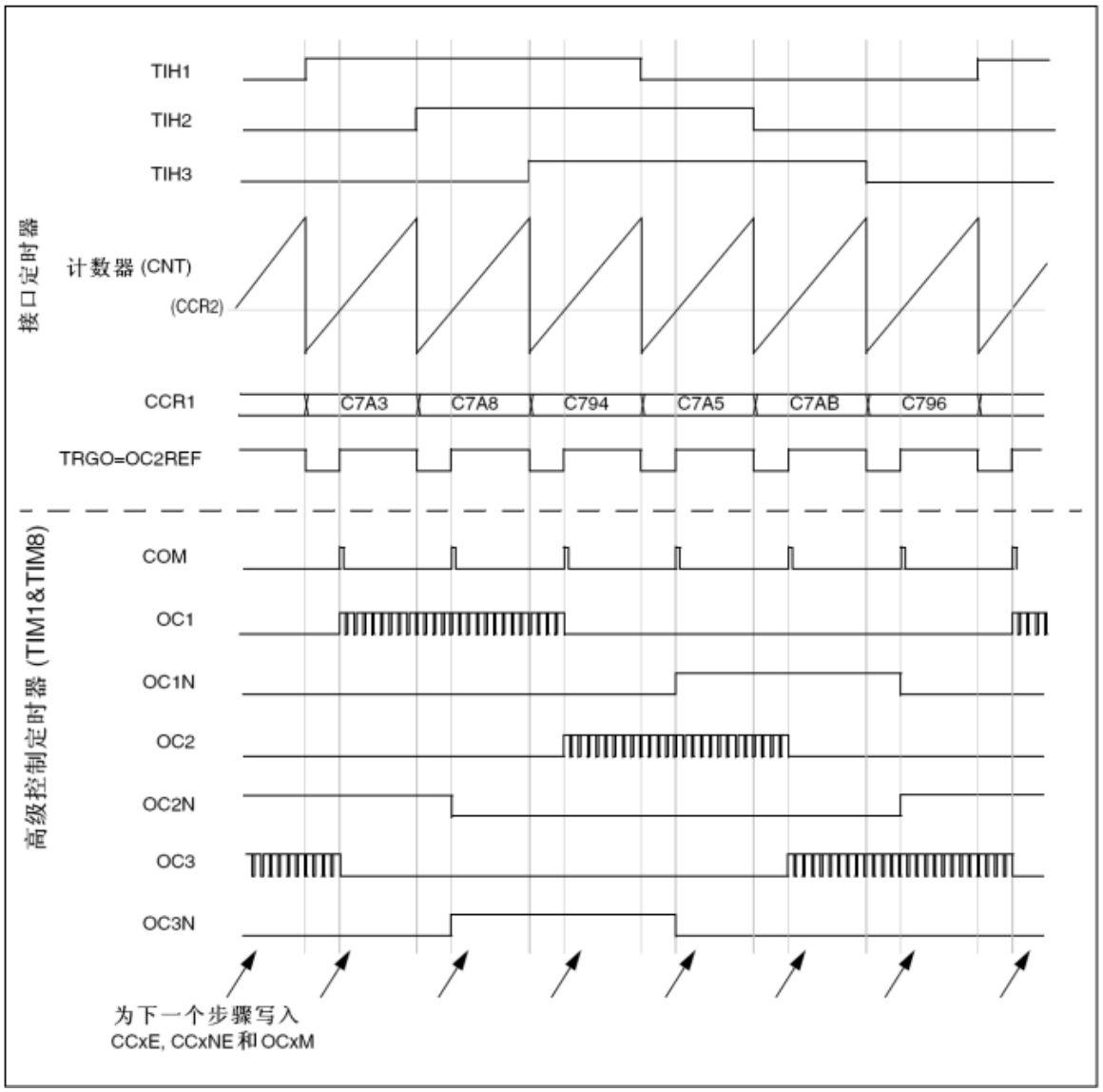
举例：霍尔输入连接到 TIMx 定时器，要求每次任一霍尔输入上发生变化之后的一个指定的时刻，改变高级控制定时器 TIMx 的 PWM 配置。

- 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 18-9 霍尔传感器接口的实例



18.3.19 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：

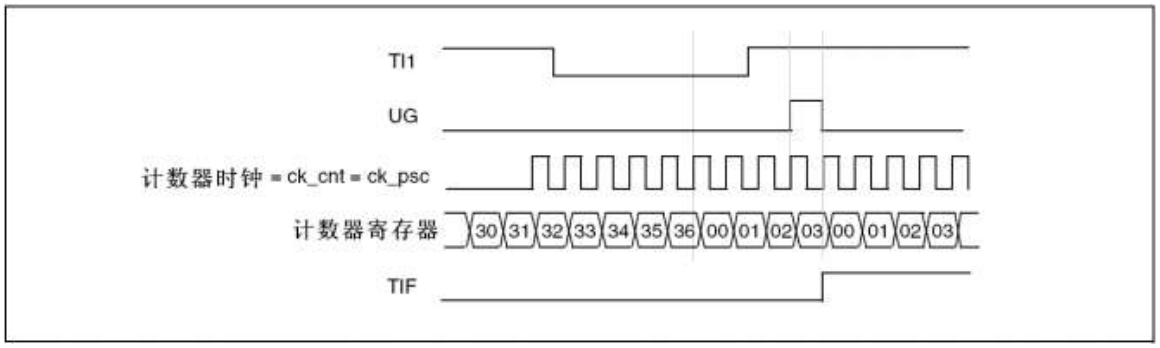
- 配置通道 1 以检测 TI1 的上升沿。配置输入滤波器的带宽(在本例中，不需要任何滤波器，因此保持 IC1F=0000)。触发操作中不使用捕获预分频器，所以不需要配置。 CC1S 位只选择输入捕获源，即 TIMx_CCMR1 寄存器中 CC1S=01。置 TIMx_CCER 寄存器中 CC1P=0 以确定极性(只检测上升沿)。
- 置 TIMx_SMCR 寄存器中 SMS=100，配置定时器为复位模式；置 TIMx_SMCR 寄存器中 TS=101，选择 TI1 作为输入源。

- 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 18-10 复位模式下的控制电路



从模式：门控模式

按照选中的输入端电平使能计数器。

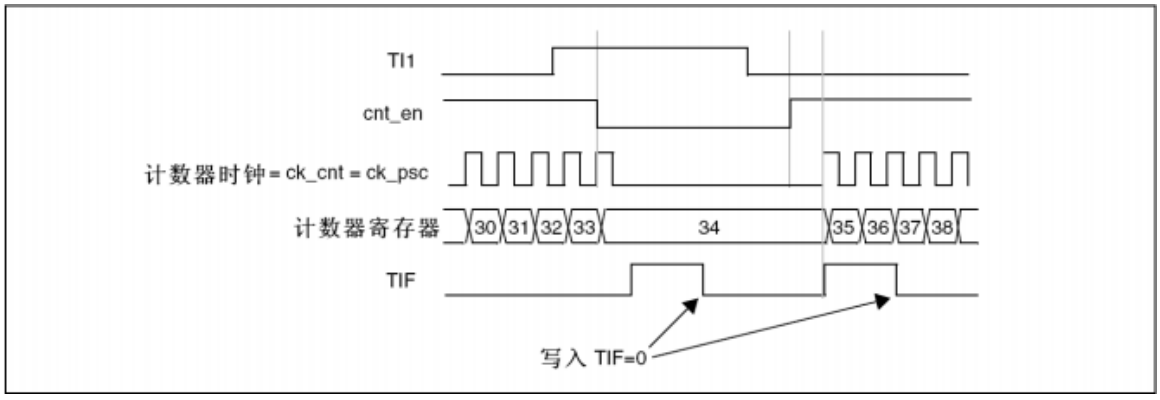
在如下的例子中，计数器只在 TI1 为低时向上计数：

- 配置通道 1 以检测 TI1 上的低电平。配置输入滤波器带宽(本例中，不需要滤波，所以保持 IC1F=0000)。触发操作中不使用捕获预分频器，所以不需要配置。 CC1S 位用于选择输入捕获源，置 TIMx_CCMR1 寄存器中 CC1S=01。置 TIMx_CCER 寄存器中 CC1P=1 以确定极性(只检测低电平)。
- 置 TIMx_SMCR 寄存器中 SMS=101，配置定时器为门控模式；置 TIMx_SMCR 寄存器中 TS=101，选择 TI1 作为输入源。
- 置 TIMx_CR1 寄存器中 CEN=1，启动计数器。在门控模式下，如果 CEN=0，则计数器不能启动，不论触发输入电平如何。

只要 TI1 为低，计数器开始依据内部时钟计数，一旦 TI1 变高则停止计数。当计数器开始或停止时都设置 TIMx_SR 中的 TIF 标置。

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 18-11 门控模式下的控制电路



从模式：触发模式

输入端上选中的事件使能计数器。

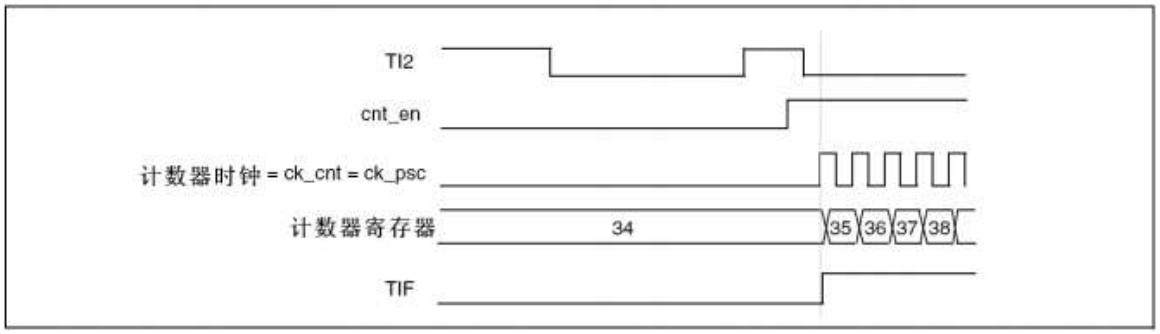
在下面的例子中，计数器在 TI2 输入的上升沿开始向上计数：

- 配置通道 2 检测 TI2 的上升沿。配置输入滤波器带宽(本例中，不需要任何滤波器，保持 IC2F=0000)。触发操作中不使用捕获预分频器，不需要配置。 CC2S 位只用于选择输入捕获源，置 TIMx_CCMR1 寄存器中 CC2S=01。置 TIMx_CCER 寄存器中 CC2P=1 以确定极性(只检测低电平)。
- 置 TIMx_SMCR 寄存器中 SMS=110，配置定时器为触发模式；置 TIMx_SMCR 寄存器中 TS=110，选择 TI2 作为输入源。

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

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

图 18-12　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

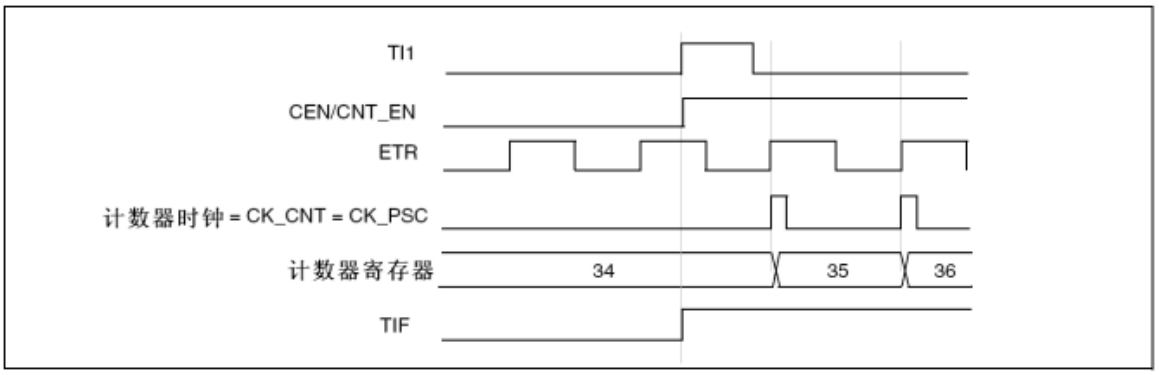
– 置 TIMx_CCER 寄存器中 CC1P=0 以确定极性(只检测上升沿)

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 18-13 外部时钟模式 2＋触发模式下的控制电路



18.3.20 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。

18.3.21 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 29.16.2 节。

18.4 TIM1 寄存器

18.4.1 TIM1 控制寄存器 1 (TIM1_CR1)

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
CC4_ADC_SEL.	Res.	CC2_ADC_SEL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	CKD[1:0]		ARPE	CMS[1:0]		DIR	OPM	URS	UDIS	CEN
						rw	rw	rw	rw	rw	rw	rw	rw		
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位 31	**CC4_ADC_SEL** 0：默认值；ADC 的 CC4 触发信号由端口 CH4 信号产生（与友商的行为一致）； 1：ADC 的 CC4 触发信号改由 Timer 内部 OC4REF 信号产生。
位 30	保留:必须保持复位值
位 29	**CC2_ADC_SEL** 0：默认值；ADC 的 CC2 触发信号由端口 CH2 信号产生； 1：ADC 的 CC2 触发信号改由 Timer 内部 OC2REF 信号产生。
位 28：10	保留:必须保持复位值

位 9：8	**CKD[1:0]**: 时钟分频因子（Clock division） 定义在定时器时钟 (CK_INT)频率、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：tDTS = tCK_INT 01：tDTS = 2 x tCK_INT 10：tDTS = 4 x tCK_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（Auto-reload preload enable） 0: TIMx_ARR 寄存器没有缓冲 1: TIMx_ARR 寄存器有缓冲
位 6：5	**CMS[1:0]**:选择中央对齐模式 (Center-aligned mode selection) 00：边沿对齐模式。计数器依据方向位 (DIR) 向上或向下计数。 01：中央对齐模式 1。计数器交替地向上和向下计数。配置为输出的通道 (TIMx_CCMRx 寄存器中 CCxS=00) 的输出比较中断标志位，只在计数器向下计数时被设置。 10：中央对齐模式 2。计数器交替地向上和向下计数。配置为输出的通道 (TIMx_CCMRx 寄存器中 CCxS=00) 的输出比较中断标志位，只在计数器向上计数时被设置。 11：中央对齐模式 3。计数器交替地向上和向下计数。配置为输出的通道 (TIMx_CCMRx 寄存器中 CCxS=00) 的输出比较中断标志位，在计数器向上和向下计数时均被设置。
位 4	**DIR:** 方向 (Direction) 0：计数器向上计数 1：计数器向下计数
位 3	**OPM:** 单脉冲模式 (One pulse mode) 0：在发生更新事件时，计数器不停止； 1：在发生下一次更新事件 (清除 CEN 位)时，计数器停止。
位 2	**URS:** 更新请求源(Update request source),软件通过该位选择 UEV 事件的源 0：如果使能了更新中断或 DMA 请求，则下述任一事件产生更新中断或 DMA 请求： — 计数器溢出/下溢 — 设置 UG 位 — 从模式控制器产生的更新 1：如果使能了更新中断或 DMA 请求，则只有计数器溢出/下溢才产生更新中断或 DMA 请求。
位 1	**UDIS:**禁止更新(Update disable),软件通过该位允许/禁止 UEV 事件的产生 0：允许 UEV。更新(UEV)事件由下述任一事件产生： — 计数器溢出/下溢 — 设置 UG 位 — 从模式控制器产生的更新,具有缓存的寄存器被装入它们的预装载值 1：禁止 UEV。不产生更新事件，影子寄存器 (ARR、PSC、CCRx)保持它们的值。如果设置了 UG 位或从模式控制器发出了一个硬件复位，则计数器和预分频器被重新初始化。
位 0	**CEN:** 使能计数器(Counter enable) 0：禁止计数器； 1：使能计数器。

18.4.2 TIM1 控制寄存器 2 (TIM1_CR2)

偏移地址: 0x04

复位值: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	OIS4	OIS3N	OIS3	OIS2N	OIS2	OIS1N	OIS1	TI1S	MMS[2:0]			CCDS	CCUS	Res.	CCPC
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位 15	保留:必须保持复位值
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)
位 12	**OIS3:** 输出空闲状态 3(OC3 输出)
位 11	**OIS2N:** 输出空闲状态 2(OC2N 输出)
位 10	**OIS2:** 输出空闲状态 2(OC2 输出)
位 9	**OIS1N:** 输出空闲状态 1(OC1N 输出) (Output Idle state 1) 0：当 MOE=0 时，死区后 OC1N=0 1：当 MOE=0 时，死区后 OC1N=1
位 8	**OIS1:** 输出空闲状态 1(OC1 输出) (Output Idle state 1) 0：当 MOE=0 时，如果完成了 OC1N，则死区后 OC1=0 1：当 MOE=0 时，如果完成了 OC1N，则死区后 OC1=1
位 7	**TI1S:** TI1 选择 (TI1 selection) 0：TIMx_CH1 引脚连到 TI1 输入 1：TIMx_CH1、TIMx_CH2 和 TIMx_CH3 引脚经异或后连到 TI1 输入
位 6：4	**MMS[1:0]:** 主模式选择 (Master mode selection) 这 3 位用于选择在主模式下送到从定时器的同步信息 (TRGO)。可能的组合如下： 000：复位 – TIMx_EGR 寄存器的 UG 位被用于作为触发输出 (TRGO)。如果是触发输入产生的复位 (从模式控制器处于复位模式)，则 TRGO 上的信号相对实际的复位会有一个延迟。 001：使能 - 计数器使能信号 CNT_EN 被用于作为触发输出 (TRGO)。可用于同时启动多个定时器或控制在一段时间内使能从定时器。在门控模式下，计数器使能信号是 CEN 控制位和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。 010：更新 - 更新事件被选为触发输入 (TRGO)。例如，一个主定时器的时钟可以被用作一个从定时器的预分频器。 011：比较脉冲 - 在发生一次捕获或一次比较成功时，当要设置 CC1IF 标志时 (即使它已经为高)，触发输出送出一个正脉冲 (TRGO)。 100：比较 – OC1REF 信号被用于作为触发输出 (TRGO) 101：比较 - OC2REF 信号被用于作为触发输出(TRGO) 110：比较 - OC3REF 信号被用于作为触发输出(TRGO) 111：比较 - OC4REF 信号被用于作为触发输出(TRGO)
位 3	**CCDS:** 捕获 /比较的 DMA 选择 (捕捉 /比较 DMA selection) 0：当发生 CCx 事件时，送出 CCx 的 DMA 请求； 1：当发生更新事件时，送出 CCx 的 DMA 请求。
位 2	**CCUS:** 捕获 /比较控制更新选择 (捕捉 / 比较 control update selection) 0：如果捕获 /比较控制位是预装载的 (CCPC=1)，只能通过设置 COMG 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 COMG 位或 TRGI 上的一个上升沿更新它们。
位 1	保留，必须始终为复位值 .
位 0	**CCPC:** 捕获/比较预装载控制位(捕捉/比较 preloaded control) 0：CCxE，CCxNE 和 OCxM 位不是预装载的； 1：CCxE，CCxNE 和 OCxM 位是预装载的；设置该位后，只能在发生通信(COM)事件（COMG 设置或 TRGI 是检测到上升沿，取决于 CCUS 位时被更新。

18.4.3 TIM1 从模式控制寄存器 (TIM1_SMCR)

偏移地址: 0x08

复位值: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ETP	ECE	ETPS[1:0]		ETF[3:0]				MSM	TS[2:0]			OCCS	SMS[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位 15	**ETP:**外部触发极性(External trigger polarity)该位选择是用 ETR 还是 ETR 的反相来作为触发操作 0：ETR 不反相，高电平或上升沿有效 1：ETR 被反相，低电平或下降沿有效
位 14	**ECE:** 外部时钟使能位(External clock enable)该位启用外部时钟模式 2 0：禁止外部时钟模式 2； 1：使能外部时钟模式 2。计数器由 ETRF 信号上的任意有效边沿驱动。
位 13:12	ETPS[1:0]: 外部触发预分频 (External trigger prescaler) 外部触发信号 ETRP 的频率最多是 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 (External trigger ?lter) 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。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 fSAMPLING=fDTS 采样 1000：采样频率 fSAMPLING=fDTS/8，N=6 0001：采样频率 fSAMPLING=fCK_INT，N=2 1001：采样频率 fSAMPLING=fDTS/8，N=8 0010：采样频率 fSAMPLING=fCK_INT，N=4 1010：采样频率 fSAMPLING=fDTS/16，N=5 0011：采样频率 fSAMPLING=fCK_INT，N=8 1011：采样频率 fSAMPLING=fDTS/16，N=6 0100：采样频率 fSAMPLING=fDTS/2，N=6 1100：采样频率 fSAMPLING=fDTS/16，N=8 0101：采样频率 fSAMPLING=fDTS/2，N=8 1101：采样频率 fSAMPLING=fDTS/32，N=5 0110：采样频率 fSAMPLING=fDTS/4，N=6 1110：采样频率 fSAMPLING=fDTS/32，N=6 0111：采样频率 fSAMPLING=fDTS/4，N=8 1111：采样频率 fSAMPLING=fDTS/32，N=8
位 7	**MSM:** 主/从模式(Master/slave mode) 0：无作用 1：触发输入(TRGI)上的事件被延迟了，以允许在当前定时器与它的从定时器间的完美同步(通过 TRGO)。这对要求把几个定时器同步到一个单一的外部事件时是非 常有用的。
位 6:4	TS[2:0]: 触发选择 (Trigger selection),这 3 位选择用于同步计数器的触发输入。 000：内部触发 0(ITR0) 100：TI1 的边沿检测器 (TI1F_ED) 001：内部触发 1(ITR1) 101：滤波后的定时器输入 1(TI1FP1) 010：内部触发 2(ITR2) 110：滤波后的定时器输入 2(TI2FP2) 011：内部触发 3(ITR3) 111：外部触发输入 (ETRF)
位 3	OCCS： OCREF Clear 选择，该 Bit 用来选择 OCREF clear source 0: OCREF_CLR_INT 被连接到 OCREF_CLR input 1: OCREF_CLR_INT 被连接到 ETRF
位 2:0	**SMS:** 从模式选择 (Slave mode selection) 当选择了外部信号，触发信号 (TRGI) 的有效边沿与选中的外部输入极性相关 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 – 根据另一个信号的输入电平，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 次更新寄存器。 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。

18.4.4 TIM1 DMA/中断使能寄存器 (TIM1_DIER)

偏移地址: 0x0C

复位值: 0x0000

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

Res.	TDE	COMDE	CC4DE	CC3DE	CC2DE	CC1DE	UDE	BIE	TIE	COMIE	CC4IE	CC3IE	CC2IE	CC1IE	UIE
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位 15	保留，必须始终为复位值 .
位 14	**TDE:** 允许触发 DMA 请求 (Trigger DMA request enable) 0: 触发 DMA 请求禁止 1: 触发 DMA 请求允许
位 13	**COMDE:** COM DMA 请求使能 0: COM DMA 请求禁止 1: COM DMA 请求允许
位 12	**CC4DE:** 捕捉 / 比较 4 DMA 请求使能 0: CC4 DMA 请求禁止 1: CC4 DMA 请求允许
位 11	**CC3DE:** 捕捉 / 比较 3 DMA 请求使能 0: CC3 DMA 请求禁止 1: CC3 DMA 请求允许
位 10	**CC2DE:** 捕捉 / 比较 2 DMA 请求使能 0: CC2 DMA 请求禁止 1: CC2 DMA 请求允许
位 9	**CC1DE:** 捕捉 / 比较 1 DMA 请求使能 0: CC1 DMA 请求禁止 1: CC1 DMA 请求允许
位 8	**UDE:** 更新 DMA 请求使能 0: 更新 DMA 请求禁止 1: 更新 DMA 请求允许
位 7	**BIE:** 刹车中断使能 0: 刹车中断禁止 1: 刹车中断允许
位 6	**TIE:** 触发中断使能 0: 触发中断禁止 1: 触发中断允许
位 5	**COMIE:** COM 中断使能 0: COM 中断禁止 1: COM 中断允许
位 4	**CC4IE:** 捕捉 / 比较 4 中断使能 0: CC4 中断禁止 1: CC4 中断允许
位 3	**CC3IE:** 捕捉 / 比较 3 中断使能 0: CC3 中断禁止 1: CC3 中断允许
位 2	**CC2IE:** 捕捉 / 比较 2 中断使能 0: CC2 中断禁止 1: CC2 中断允许
位 1	**CC1IE:** 捕捉 / 比较 1 中断使能 0: CC1 中断禁止 1: CC1 中断允许
位 0	**UIE:** 更新中断使能 0: 更新中断禁止 1: 更新中断允许

18.4.5 TIM1 状态寄存器 (TIM1_SR)

偏移地址: 0x10

复位值: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	CC4OF	CC3OF	CC2OF	CC1OF	Res.	BIF	TIF	COMIF	CC4IF	CC3IF	CC2IF	CC1IF	UIF
			rc w0	rc w0	rc w0	rc w0		rc w0	rc w0	rc w0	rc w0	rc w0	rc w0	rc w0	rc w0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位 15:13	保留，必须始终为复位值 .
位 12	**CC4OF:** 捕捉 / 比较 4 重复捕捉标志
位 11	**CC3OF:** 捕捉 / 比较 3 重复捕捉标志
位 10	**CC2OF:** 捕捉 / 比较 2 重复捕捉标志
位 9	**CC1OF:** 捕捉 / 比较 1 重复捕捉标志 (Capture/Compare 1 overcapture flag) 仅当相应的通道被配置为输入捕获时，该标记可由硬件置 1。软件写 0 可清除该位。 0：无重复捕获产生 1：当 CC1IF 的状态已经为’1’，计数器的值被捕获到 TIMx_CCR1 寄存器。
位 8	保留，必须始终为复位值 .
位 7	**BIF:** 刹车中断标志 (Break interrupt ?ag) 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:** 触发器中断标志 (Trigger interrupt ?ag) 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 中断标志 (COM interrupt ?ag) 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:** 捕捉 /比较 4 中断标志
位 3	**CC3IF:** 捕捉/比较 3 中断标志
位 2	**CC2IF:** 捕捉/比较 2 中断标志
位 1	**CC1IF:** 捕捉 /比较 1 中断标志 (Capture/Compare 1 interrupt flag) **如果通道 CC1 配置为输出模式：** 当计数器值与比较值匹配时该位由硬件置 1，但在中心对称模式下除外。它由软件清’0’。 0：无匹配发生 1：TIMx_CNT 的 值 与 TIMx_CCR1 的 值 匹 配。 当 TIMx_CCR1 的 内 容 大 于 TIMx_APR 的内容时，在向上或向上 /下计数模式时计数器溢出，或向下计数模式时的计数器下溢条件下，CC1IF 位变高。 **如果通道 CC1 配置为输入模式：** 当捕获事件发生时该位由硬件置’1’，它由软件清’0’或通过读 TIMx_CCR1 清’0’。 0：无输入捕获产生； 1：计数器值被捕获至 TIMx_CCR1(在 IC1 上检测到与所选极性相同的边沿)。

位 0	**UIF:**更新中断标志(Update interrupt flag) 当产生更新事件时该位由硬件置’1’。它由软件清’0’。 0：无更新事件产生； 1：更新中断等待响应。当寄存器被更新时该位由硬件置’1’： － 若 TIMx_CR1 寄存器的 UDIS=0，当重复计数器数值上溢或下溢时(重复计数器=0 时产生更新事件)。 － 若 TIMx_CR1 寄存器的 URS=0、UDIS=0，当设置 TIMx_EGR 寄存器 的 UG=1 时产生更新事件，通过软件对计数器 CNT 重新初始化时。 － 若 TIMx_CR1 寄存器的 URS=0、UDIS=0，当计数器 CNT 被触发事件重新初始化时。

18.4.6 TIM1 事件产生寄存器 (TIM1_EGR)

偏移地址: 0x14

复位值: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BG	TG	COMG	CC4G	CC3G	CC2G	CC1G	UG
								w	w	w	w	w	w	w	w
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位 15:8	保留，必须始终为复位值 .
位 7	**BG:** 产生刹车事件(Break generation)该位由软件置’1’，用于产生一个刹车事件，由硬件自动清’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:** 触发产生（Trigger generation） 该位由软件置’1’，用于产生一个事件，由硬件自动清’0’。 0：无动作 1：TIMx_SR 中 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:** 捕捉/比较控制更新产生(Capture/Compare control update generation）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
位 4	**CC4G:**捕捉/比较 4 发生
位 3	**CC3G:**捕捉/比较 3 发生
位 2	**CC2G:**捕捉/比较 2 发生
位 1	**CC1G:**捕捉/比较 1 发生(Capture/Compare 1 generation) 该位由软件置’1’，用于产生一个捕获 / 比较事件，由硬件自动清’0’。 0：无动作 1：在通道 1 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 (Update generation) 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 (寄存器) 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。

18.4.7 TIM1 捕捉/比较模式寄存器 1 (TIM1_CCMR1)

偏移地址：0x18

复位值：0x0000

通道可用于输入(捕获模式)或输出(比较模式)，通道的方向由相应的 CCxS 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
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

输出比较模式:

位 15	**OC2CE:** 输出比较 2 清 0 允许
位 14:12	**OC2M[2:0]:** 输出比较模式 2
位 11	**OC2PE:** 输出比较 2 预装允许
位 10	**OC2FE:** 输出比较 2 快速允许
位 9:8	**CC2S[1:0]:** 捕捉/比较 2 选择，该位定义通道的方向(输入/输出)，及输入信号的选择 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 允许 0: OC1REF 不受 ETRF 输入的影响； 1: 一旦检测到 ETRF 输入高电平，清除 OC1REF=0。
位 6：4	**OC1M:** 输出比较模式 1(Output Compare 1 mode)，该 3 位定义了输出参考信号 OC1REF 的动作，而 OC1REF 决定了 OC1、OC1N 的值。OC1REF 是高电平有效，而 OC1、OC1N 的有效电平取决于 CC1P、CC1NP 位。 000：冻结。输出比较寄存器 TIMx_CCR1 与计数器 TIMx_CNT 间的比较对 OC1REF 不起作用 001：匹配时设置通道 1 为有效电平。当计数器 TIMx_CNT 的值与捕获/比较寄存器 1(TIMx_CCR1) 相同时，强制 OC1REF 为高。 010 ：匹配时设置通道 1 为无效电平。当计数器 TIMx_CNT 的值与捕获/比较寄存器 1(TIMx_CCR1) 相同时，强制 OC1REF 为低。 011：翻转。当 TIMx_CCR1=TIMx_CNT 时，翻转 OC1REF 的电平。 100：强制为无效电平。强制 OC1REF 为低。 101：强制为有效电平。强制 OC1REF 为高。 110：PWM 模式 1－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 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。
位 3	**OC1PE:** 输出比较 1 预装允许(Output Compare 1 preload enable) 0：禁止 TIMx_CCR1 寄存器的预装载功能，可随时写入 TIMx_CCR1 寄存器，并且新写入的数值立即起作用。 1：开启 TIMx_CCR1 寄存器的预装载功能，读写操作仅对预装载寄存器操作，TIMx_CCR1 的预装载值在更新事件到来时被加载至当前寄存器中。

位 2	**OC1FE:** 输出比较 1 快速使能(Output Compare 1 fast enable) 该位用于加快 CC 输出对触发输入事件的响应。 0：CC1 的正常操作依赖于计数器与 CCR1 的值，即使工作于触发器状态。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:** 捕捉/比较 1 选择 (Capture/Compare 1 selection) 这 2 位定义通道的方向(输入/输出)，及输入脚的选择： 00：CC1 通道被配置为输出 01：CC1 通道被配置为输入，IC1 映射在 TI1 上 10：CC1 通道被配置为输入，IC1 映射在 TI2 上 11：CC1 通道被配置为输入，IC1 映射在 TRC 上。此模式仅工作在内部触发器输入被选中时(由 TIMx_SMCR 寄存器的 TS 位选择)。

输入捕捉模式

位 15：12	**IC2F:** 输入捕捉 2 滤波器
位 11：10	**IC2PSC[1:0]:** 输入捕捉 2 预分频器
位 9：8	**CC2S:** 捕捉/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)。
位 7：4	**IC1F[3:0]:** 输入捕捉 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 fDTS 采样 1000：采样频率 fSAMPLING=fDTS/8，N=6 0001：采样频率 fSAMPLING=fCK_INT，N=2 1001：采样频率 fSAMPLING=fDTS/8，N=8 0010：采样频率 fSAMPLING=fCK_INT，N=4 1010：采样频率 fSAMPLING=fDTS/16，N=5 0011：采样频率 fSAMPLING=fCK_INT，N=8 1011：采样频率 fSAMPLING=fDTS/16，N=6 0100：采样频率 fSAMPLING=fDTS/2，N=6 1100：采样频率 fSAMPLING=fDTS/16，N=8 0101：采样频率 fSAMPLING=fDTS/2，N=8 1101：采样频率 fSAMPLING=fDTS/32，N=5 0110：采样频率 fSAMPLING=fDTS/4，N=6 1110：采样频率 fSAMPLING=fDTS/32，N=6 0111：采样频率 fSAMPLING=fDTS/4，N=8 1111：采样频率 fSAMPLING=fDTS/32，N=8
位 3:2	**IC1PSC:** 输入捕捉 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:** 捕捉/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)。

18.4.8 TIM1 捕捉/比较模式寄存器 2 (TIM1_CCMR2)

偏移地址: 0x1C

复位值: 0x0000

参看以上 CCMR1 寄存器的描述

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

OC4CE	OC4M[2:0]			OC4PE	OC4FE	CC4S[1:0]		OC3CE	OC3M[2:0]			OC3PE	OC3FE	CC3S[1:0]	
IC4F[3:0]				IC4PSC[1:0]				IC3F[3:0]				IC3PSC[1:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

输出比较模式:

位 15	**OC4CE:** 输出比较 4 清 0 允许
位 14:12	OC4M[2:0]: 输出比较模式 4
位 11	**OC4PE:** 输出比较 4 预装允许
位 10	**OC4FE:** 输出比较 4 快速允许
位 9:8	CC4S[1:0]: 捕捉/比较 4 选择，该位定义通道的方向(输入/输出)，及输入信号的选择 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 允许 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	**OC3M:** 输出比较 3 模式(Output Compare 3 mode)
位 3	**OC3PE:** 输出比较 3 预装允许(Output Compare 3 preload enable)
位 2	**OC3FE:** 输出比较 3 快速使能(Output Compare 3 fast enable)
位 1：0	CC3S[1:0]: 捕捉/比较 3 选择 (Capture/Compare 3 selection) 这 2 位定义通道的方向(输入/输出)，及输入脚的选择： 00：CC3 通道被配置为输出 01：CC3 通道被配置为输入，IC3 映射在 TI3 上 10：CC3 通道被配置为输入，IC3 映射在 TI4 上 11：CC3 通道被配置为输入，IC3 映射在 TRC 上。此模式仅工作在内部触发器输入被选中时(由 TIMx_SMCR 寄存器的 TS 位选择)。

输入捕捉模式

位 15：12	**IC4F:** 输入捕捉 4 滤波器
位 11：10	**IC4PSC[1:0]:** 输入捕捉 4 预分频器
位 9：8	**CC4S:** 捕捉/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)。
位 7：4	**IC3F[3:0]:** 输入捕捉 3 滤波器，这几位定义了 TI3 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 fDTS 采样 1000：采样频率 fSAMPLING=fDTS/8，N=6 0001：采样频率 fSAMPLING=fCK_INT，N=2 1001：采样频率 fSAMPLING=fDTS/8，N=8 0010：采样频率 fSAMPLING=fCK_INT，N=4 1010：采样频率 fSAMPLING=fDTS/16，N=5 0011：采样频率 fSAMPLING=fCK_INT，N=8 1011：采样频率 fSAMPLING=fDTS/16，N=6 0100：采样频率 fSAMPLING=fDTS/2，N=6 1100：采样频率 fSAMPLING=fDTS/16，N=8 0101：采样频率 fSAMPLING=fDTS/2，N=8 1101：采样频率 fSAMPLING=fDTS/32，N=5 0110：采样频率 fSAMPLING=fDTS/4，N=6 1110：采样频率 fSAMPLING=fDTS/32，N=6 0111：采样频率 fSAMPLING=fDTS/4，N=8 1111：采样频率 fSAMPLING=fDTS/32，N=8

位 3:2	**IC3PSC:** 输入捕捉 3 预分频器，这 2 位定义了 CC3 输入(IC3) 的预分频系数。一旦 CC3E=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	**CC3S:** 捕捉/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)。

18.4.9 TIM1 捕捉/比较使能寄存器 (TIM1_CCER)

偏移地址: 0x20

复位值: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	CC4P	CC4E	CC3NP	CC3NE.	CC3P	CC3E	CC2NP	CC2NE	CC2P	CC2E	CC1NP	CC1NE	CC1P	CC1E
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位 15:14	保留，必须始终为复位值 .
位 13	**CC4P:** 捕捉/比较 4 输出极性
位 12	**CC4E:** 捕捉/比较 4 输出使能
位 11	**CC3NP:** 捕捉/比较 3 互补输出极性
位 10	**CC3NE:** 捕捉/比较 3 互补输出使能
位 9	**CC3P:** 捕捉/比较 3 输出极性
位 8	**CC3E:** 捕捉/比较 3 输出使能
位 7	**CC2NP:** 捕捉/比较 2 互补输出极性
位 6	**CC2NE:** 捕捉/比较 2 互补输出使能
位 5	**CC2P:** 捕捉/比较 2 输出极性
位 4	**CC2E:** 捕捉/比较 2 输出使能
位 3	**CC1NP:** 捕捉/比较 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（TIM_BDTR 寄存器中的 LOCK 位）设为 3 或 2 且 CC1S=00（通道配置为输出）则该位不能被修改。
位 2	**CC1NE:** 捕捉 /比较 1 互补输出使能 0：关闭－ OC1N 禁止输出，因此 OC1N 的电平依赖于 MOE、OSSI、OSSR、OIS1、OIS1N 和 CC1E 位的值。 1：开启－ OC1N 信号输出到对应的输出引脚，其输出电平依赖于 MOE、OSSI、OSSR 、OIS1、OIS1N 和 CC1E 位的值。

位 1	**CC1P:** 捕捉 /比较 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：** CC1NP/CC1P 位选择在触发或捕捉模式下 TI1FP1 和 TI2FP1 的有效极性。 00: 非反相 /上升沿 电路作用于 TIxFP1 的 上升沿 (在复位、外部时钟或触发模式下的捕捉或触发操作),TIxFP1 非反相 01: 反想 /下降沿 电路作用于 TIxFP1 的 下降沿 (在复位、外部时钟或触发模式下的捕捉或触发操作),TIxFP1 反相 00: 保留不用 11: 非反相 /上升或下降沿 电路作用于 TIxFP1 的上升沿 和下降沿 (在复位、外部时钟或触发模式下的捕捉或触发操作), TIxFP1 非反相 (在门控模式)。在编码模式下不能使用此配置。
位 0	**CC1E:** 捕捉 /比较 1 输出使能 **CC1 通道配置为输出：** 0：关闭一 OC1N 禁止输出，因此 OC1N 的电平依赖于 MOE、OSSI、OSSR、OIS1、OIS1N 和 CC1E 位的值。 1：开启一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。 **CC1 通道配置为输入：** 本位用于决定是否一个定时器值的捕捉要装载到捕捉 / 比较寄存器 1(TIMx_CCR1)。 0: 捕捉禁止 1: 捕捉允许

18.4.10 TIM1 计数器 (TIM1_CNT)

偏移地址: 0x24

复位值: 0x0000

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

位 15:0	CNT[15:0]: 计数器值

18.4.11 TIM1 预分频器 (TIM1_PSC)

偏移地址: 0x28

复位值: 0x0000

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

位 15:0	PSC[15:0]: 预分频值 预分频器的值 (Prescaler value)计数器的时钟频率 (CK_CNT)等于 fCK_PSC/(PSC[15:0]+1)。 每次当更新事件产生时，PSC 的值被装入当前预分频器寄存器；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’。

18.4.12 TIM1 自动重装载寄存器 (TIM1_ARR)

偏移地址: 0x2C

复位值: 0x0000

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

位 15:0	ARR[15:0]: 自动重装载的值 (Prescaler value) ARR 包含了将要装载入实际的自动重装载寄存器的值。

18.4.13 TIM1 重复计数寄存器 (TIM1_RCR)

偏移地址: 0x30

复位值: 0x0000

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

位 15:8	保留，必须始终为复位值
位 7:0	REP[7:0]: 重复计数器的值 (Repetition counter value) 预装载寄存器被使能后，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

18.4.14 TIM1 捕捉/比较寄存器 1 (TIM1_CCR1)

偏移地址: 0x34

复位值: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位 15:0	CCR1[15:0]: 捕捉 /比较通道 1 的值 **若 CC1 通道配置为输出：** CCR1 决定了装入当前捕获 /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。

18.4.15 TIM1 捕捉/比较寄存器 2 (TIM1_CCR2)

偏移地址: 0x38

复位值: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR2[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位 15:0	CCR2[15:0]: 捕捉 /比较通道 2 的值 **若 CC2 通道配置为输出：** CCR2 决定了装入当前捕获 /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 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。

18.4.16 TIM1 捕捉/比较寄存器 3 (TIM1_CCR3)

偏移地址: 0x3C

复位值: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR3[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位 15:0	**CCR3[15:0]:** 捕捉 /比较通道 3 的值 **若 CC3 通道配置为输出：** CCR3 决定了装入当前捕获 /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 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。

18.4.17 TIM1 捕捉/比较寄存器 4 (TIM1_CCR4)

偏移地址: 0x40

复位值: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR4[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位 15:0	**CCR4[15:0]:** 捕捉 /比较通道 4 的值 **若 CC4 通道配置为输出：** CCR4 决定了装入当前捕获 /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 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。

18.4.18 TIM1 刹车和死区寄存器 (TIM1_BDTR)

偏移地址: 0x44

复位值: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MOE	AOE	BKP	BKE	OSSR	OSSI	LOCK[1:0]		DTG[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位 15	**MOE:** 主输出使能 (Main output enable) 一旦刹车输入有效，该位被硬件异步清’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
位 14	**AOE:** 自动输出使能 (Automatic output enable) 0：MOE 只能被软件置’1’ 1：MOE 能被软件置’1’或在下一个更新事件被自动置’1’(如果刹车输入无效)
位 13	**BKP:** 刹车输入极性 (Break polarity) 0：刹车输入低电平有效 1：刹车输入高电平有效
位 12	**BKE:** 刹车使能 (Break enable) 0: 刹车输入禁止 (BRK 和 CCS 时钟失效事件) 1: 刹车输入允许 (BRK 和 CCS 时钟失效事件)
位 11	**OSSR:** 运行模式下“关闭状态”选择 (Off-state selection for Run mode) 该位用于当 MOE=1 且通道为互补输出时。没有互补输出的定时器中不存在 OSSR 位 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 使能并输出无效电平，然后置 OC/OCN 使能输出信号 =1
位 10	**OSSI:** 运行模式下“空闲状态”选择 (Off-state selection fo Idle mode) 该位用于当 MOE=0 时通道为输出。 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 被强制输出空闲电平，置 OC/OCN 使能输出信号 =1

位 9:8	LOCK[1:0]:锁定设置 (Lock con?guration) 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)
位 7:0	DTG[7:0]: 死区发生器设置 (Dead-time generator setup) 这些位定义了插入互补输出之间的死区持续时间。

18.4.19 TIM1 DMA 控制寄存器 (TIM1_DCR)

偏移地址: 0x48

复位值: 0x0000

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

位 15:13	保留，必须始终为复位值 .
位 12:8	**DBL[4:0]:** DMA 连续传送长度 (DMA burst length) 这 5 位定义了 DMA 在连续模式下的传送长度 (当对 TIMx_DMAR 寄存器进行读或写时，定时器则进行一次连续传送) 00000：1 次传输 00001：2 次传输 00010：3 次传输 .... 10001：18 次传输
位 7:5	保留，必须始终为复位值 .
位 4:0	**DBA[4:0]:** DMA 基地址 (DMA base address) 00000:TIM1_CR1 00001:TIM1_CR2 00010:TIM1_SMCR ……

18.4.20 TIM1 全部传输时 DMA 地址 (TIM1_DMAR)

偏移地址: 0x4C

复位值: 0x0000

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

位 15:0	**DMAB[15:0]:** DMA 并发（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访问： （TIMx_CR1 地址） + （DBA + DMA 索引）x 4，其中： “TIMx_CR1 地址”是控制寄存器 1(TIMx_CR1)所在的地址； “DBA”是 TIMx_DCR 寄存器中定义的基地址； “DMA 索引”是由 DMA 自动控制的偏移量，它取决于 TIMx_DCR 寄存器中定义的 DBL。

18.4.21 TIM1 比较器控制寄存器 (TIM1_OR)

偏移地址: 0x50

复位值: 0x0000

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

位 15:7	保留，必须始终为复位值 .
位 6:5	**OCREF_CLR_RMP:** 选择比较器 1 或比较器 2 输出作为 OCREF_CLR 的来源 00：OCREF_CLR 连接 COMP1_OUT 01: OCREF_CLR 连接 COMP2_OUT 10: OCREF_CLR 连接（COMP1_ OUT | COMP2_ OUT） 11: OCREF_CLR 连接（COMP1_ OUT | COMP2_ OUT）
位 4	保留，必须始终为复位值 .
位 3:2	**VCOMPO_RMP:** 选择比较器 1 或比较器 2 输出作为 BREAK 的来源 00：VCOMPO 连接 COMP1_OUT 01: VCOMPO 连接 COMP2_OUT 10 或 11: VCOMPO_CLR 连接（COMP1_ OUT | COMP2_ OUT）

19 通用定时器 (TIM2/TIM3)

19.1 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。

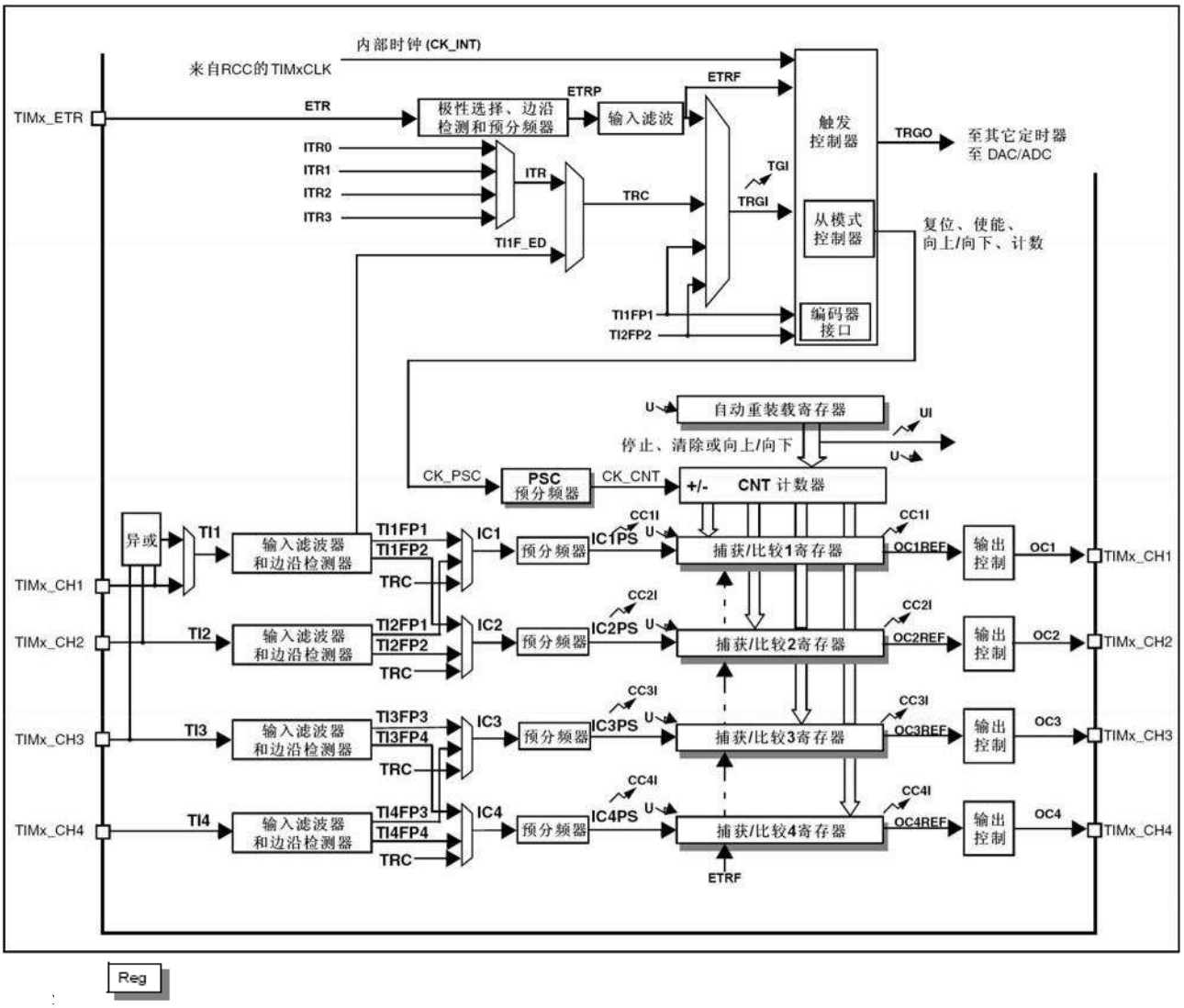
每个定时器都是完全独立的，没有互相共享任何资源。它们可以一起同步操作，参见章节：定时器同步。

19.2 TIMx 主要功能

通用 TIMx (TIM2、TIM3)定时器功能包括：

- 新增 DAC 触发功能
- 四路输入通道都新增下降沿触发，和双沿触发功能
- 16 位向上、向下、向上/向下自动装载计数器
- 16 位可编程(可以实时修改)预分频器，计数器时钟频率的分频系数为 1～65536 之间的任意数值
- 4 个独立通道：
  - 输入捕获
  - 输出比较
  - PWM 生成(边缘或中间对齐模式)
  - 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 19-1 通用定时器框图



事件

中断和 DMA 输出

19.3 TIMx 功能描述

19.3.1 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 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。这个控制寄存器带有缓冲器，它能够在工作时被改 变。新的预分频器参数在下一次更新事件到来时被采用。

下面两个图给出了在预分频器运行时，更改计数器参数的例子。

图 19-2 当预分频器的参数从 1 变到 2 时，计数器的时序图

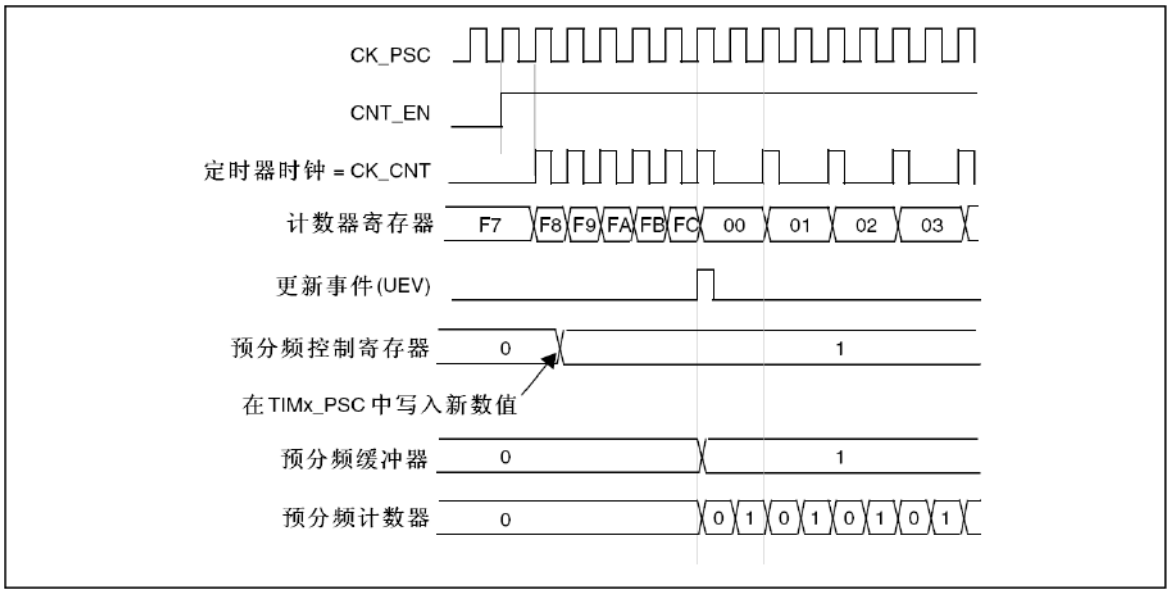
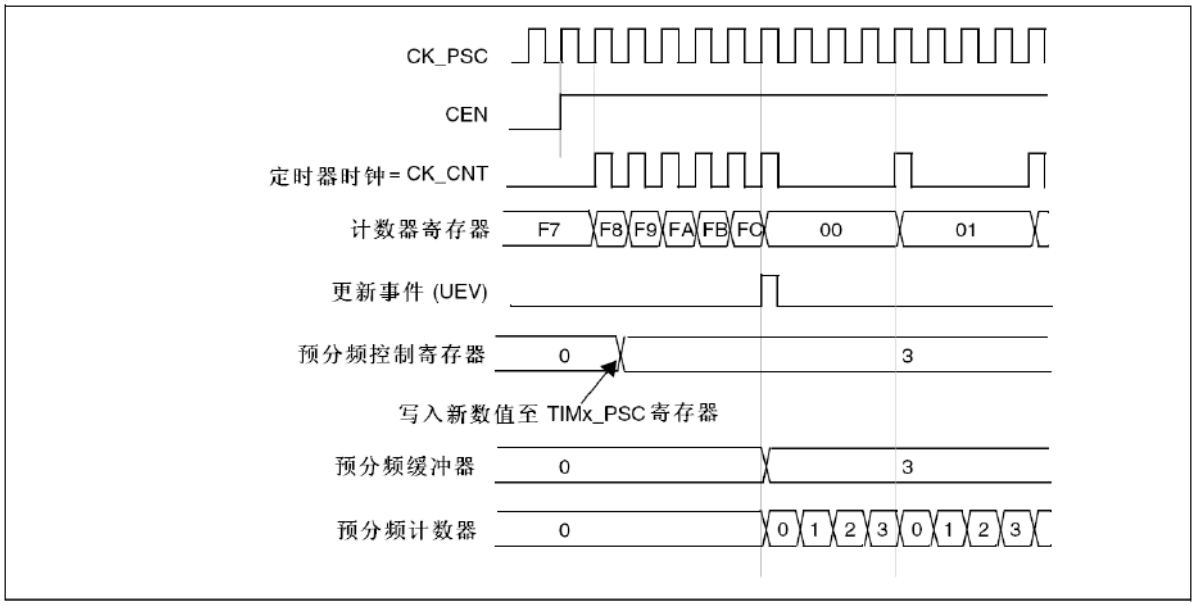


图 19-3 当预分频器的参数从 1 变到 4 时，计数器的时序图



19.3.2 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(通过软件方式或者使用从模式控制器)设置 UG 位也同样可以产生一个更新事件。

设置 TIMx_CR1 寄存器中的 UDIS 位，可以禁止更新事件；这样可以避免在向预装载寄存器中写入新值时更新影子寄存器。在 UDIS 位被清'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(但预分频系数不变)。此外，如果设置 了 TIMx_CR1 寄存器中的 URS 位(选择更新请求)，设置 UG 位将产生一个更新事件 UEV，但硬件 不设置 UIF 标志(即不产生中断或 DMA 请求)；这是为了避免在捕获模式下清除计数器时，同时产 生更新和捕获中断。

当发生一个更新事件时，所有的寄存器都被更新，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 19-4 计数器时序图，内部时钟分频因子为 1

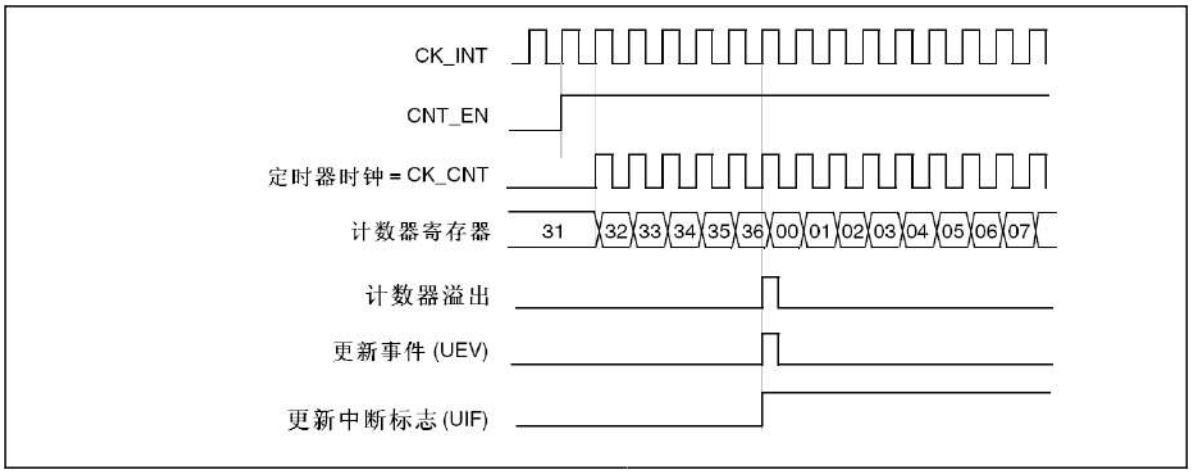


图 19-5 计数器时序图，内部时钟分频因子为 2

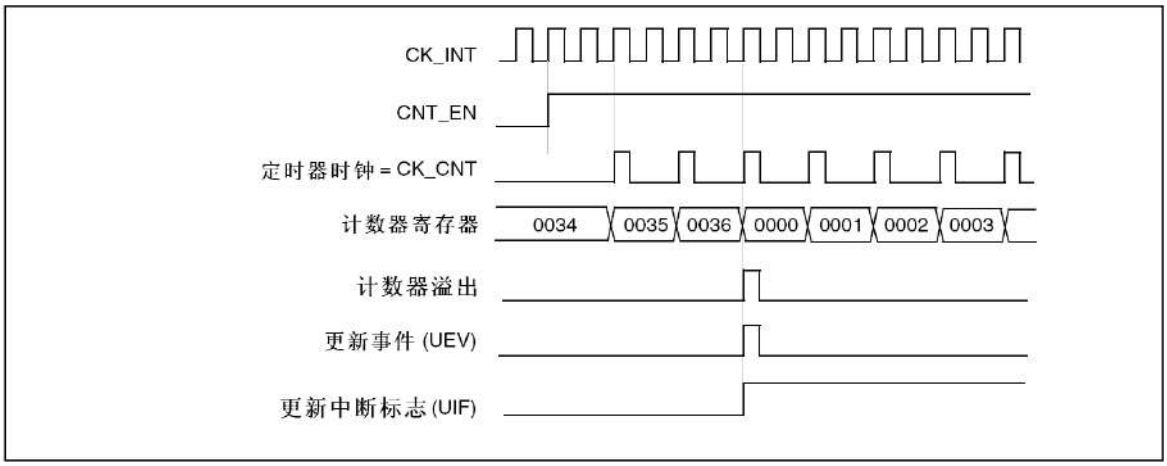


图 19-6 计数器时序图，内部时钟分频因子为 4

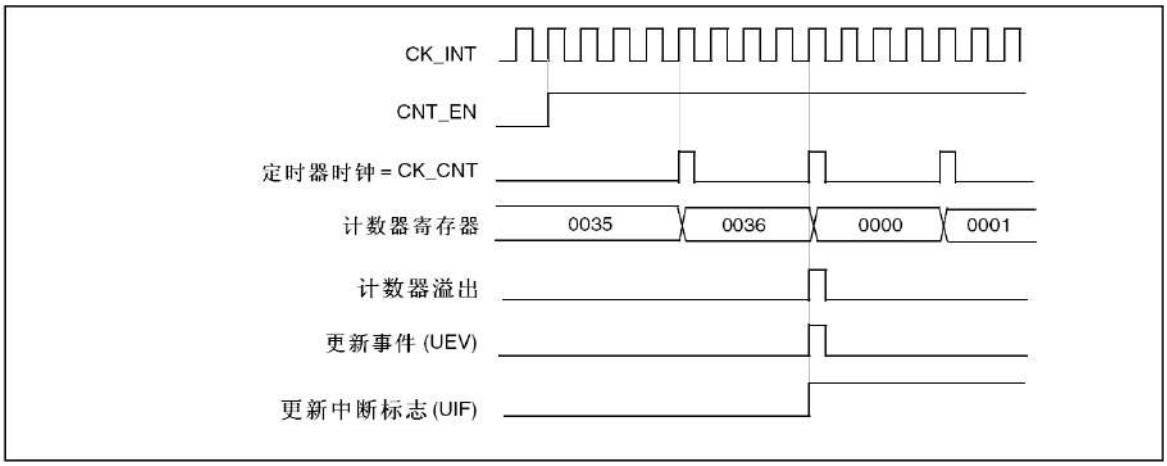


图 7 计数器时序图，内部时钟分频因子为 N

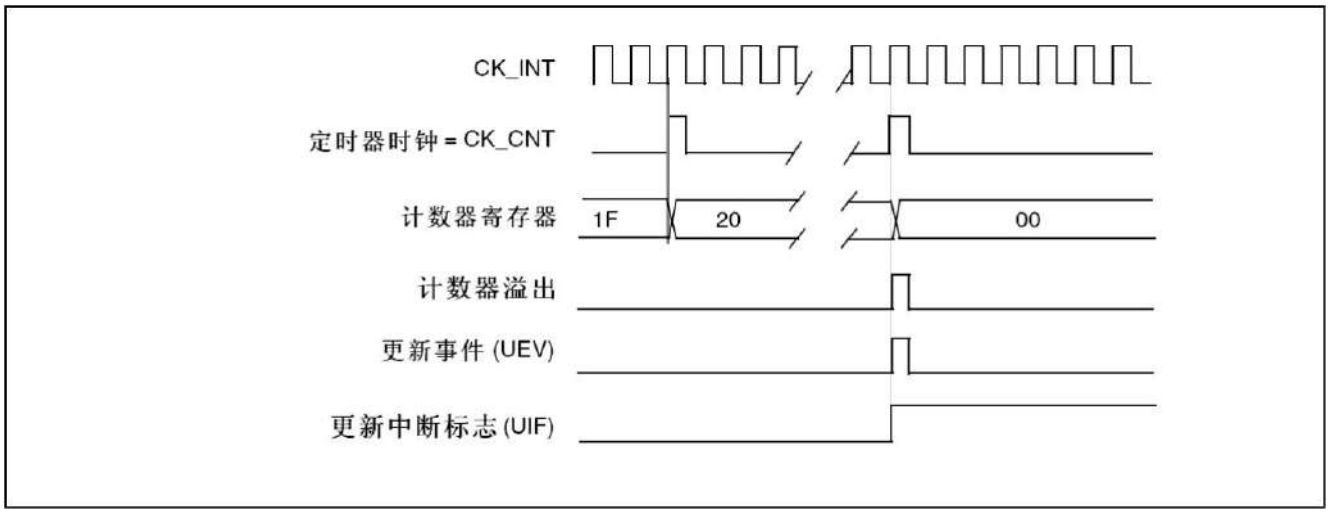


图 19-7 计数器时序图，当 ARPE=0 时的更新事件(TIMx_ARR 没有预装入)

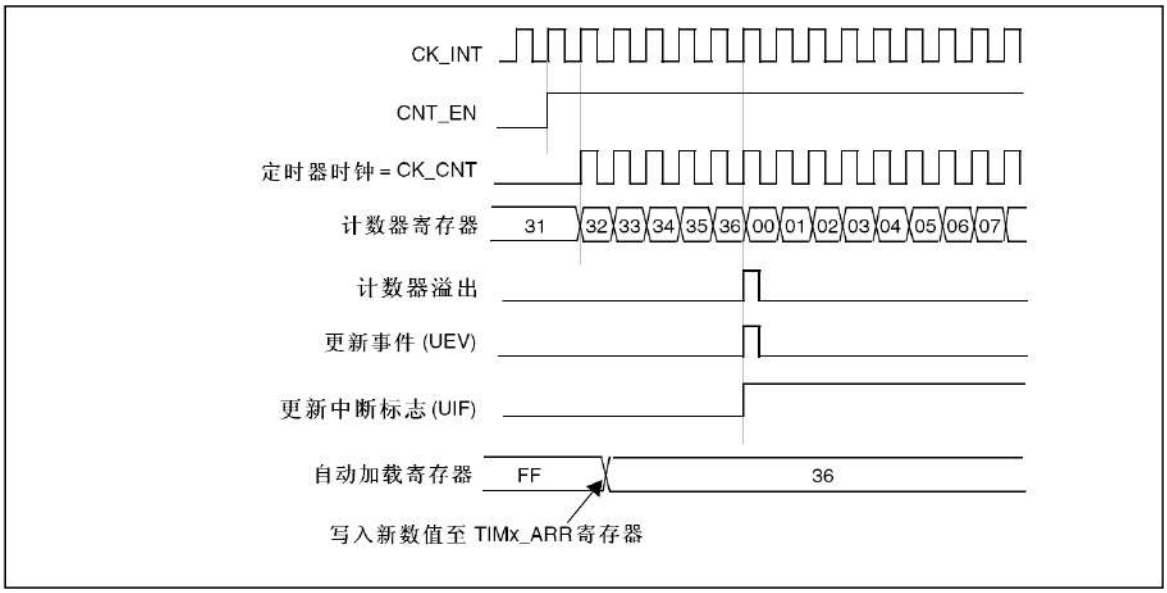
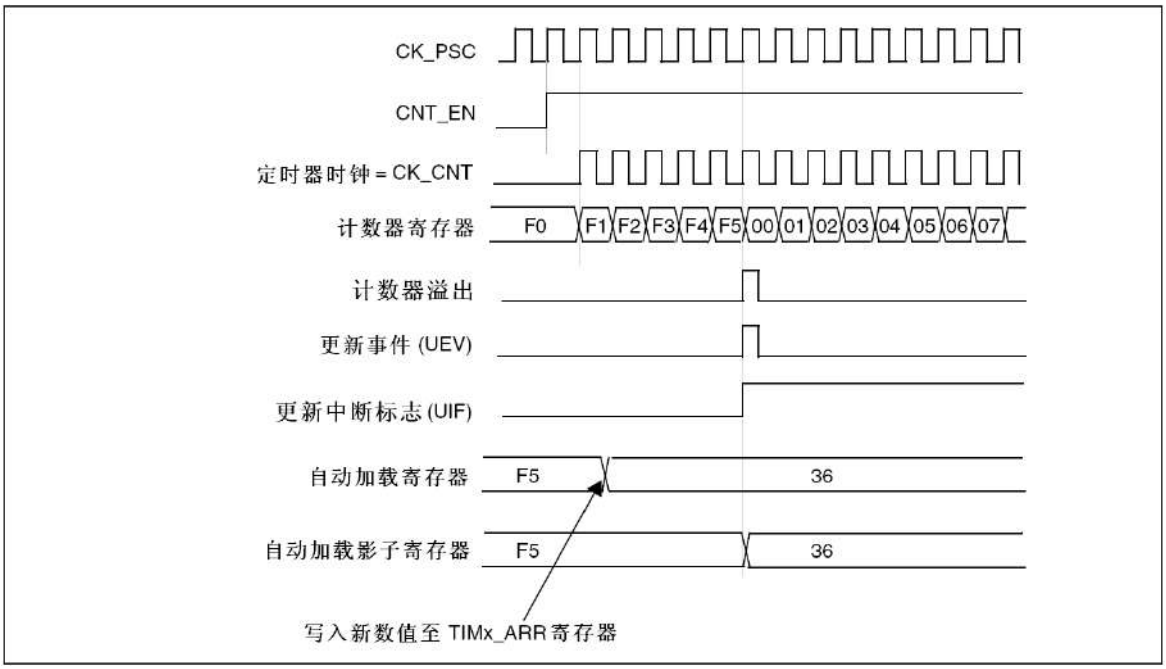


图 19-8 计数器时序图，当 ARPE=1 时的更新事件(预装入了 TIMx_ARR)



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

设置 TIMx_CR1 寄存器的 UDIS 位可以禁止 UEV 事件。这样可以避免向预装载寄存器中写入新值 时更新影子寄存器。因此 UDIS 位被清为'0'之前不会产生更新事件。然而，计数器仍会从当前自 动加载值重新开始计数，同时预分频器的计数器重新从 0 开始(但预分频系数不变)。

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

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

- 预分频器的缓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 19-9 计数器时序图，内部时钟分频因子为 1

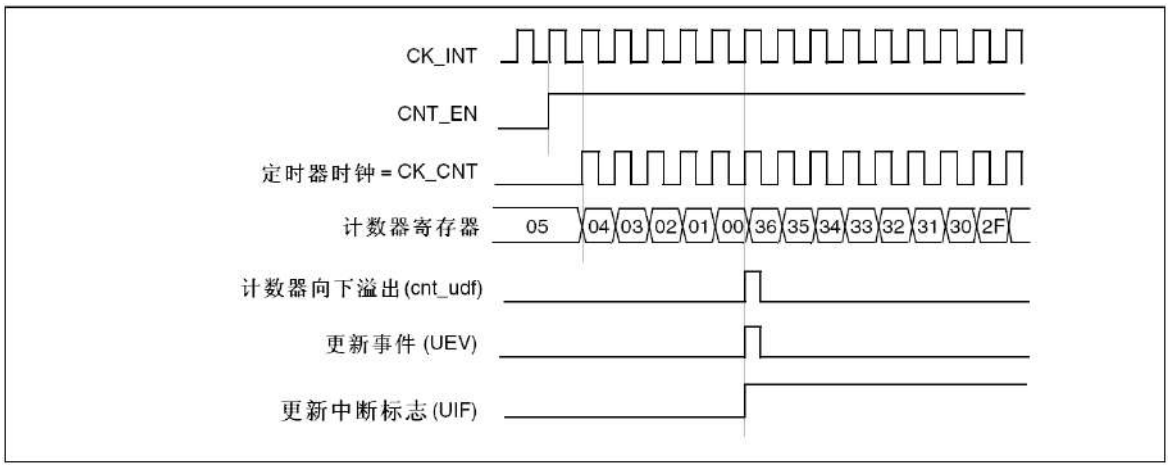


图 19-10 计数器时序图，内部时钟分频因子为 2

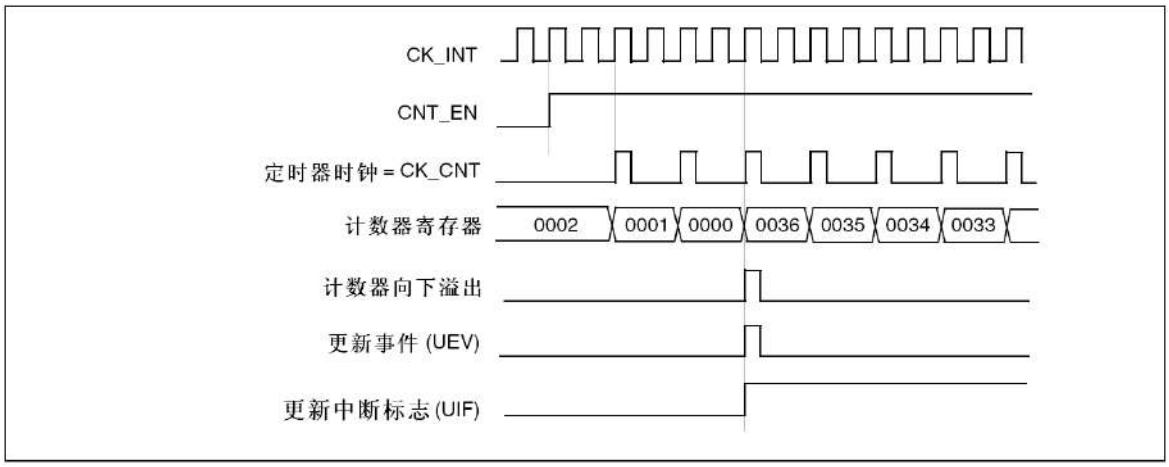


图 19-11 计数器时序图，内部时钟分频因子为 4

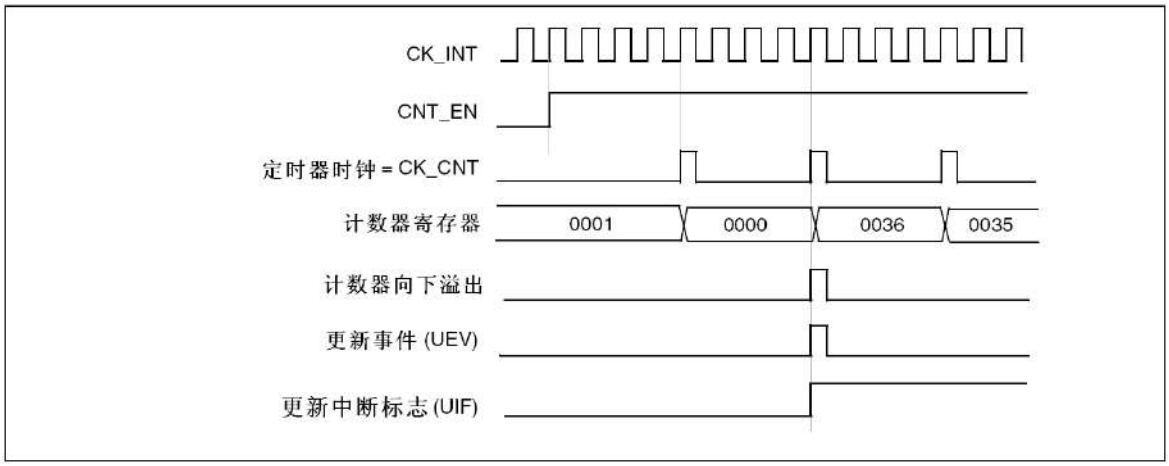


图 19-12 计数器时序图，内部时钟分频因子为 N

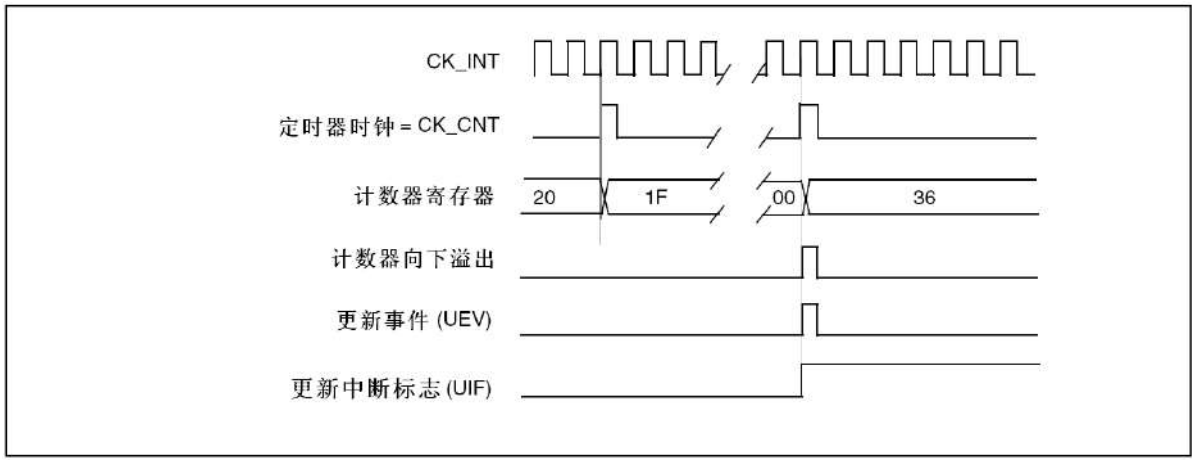
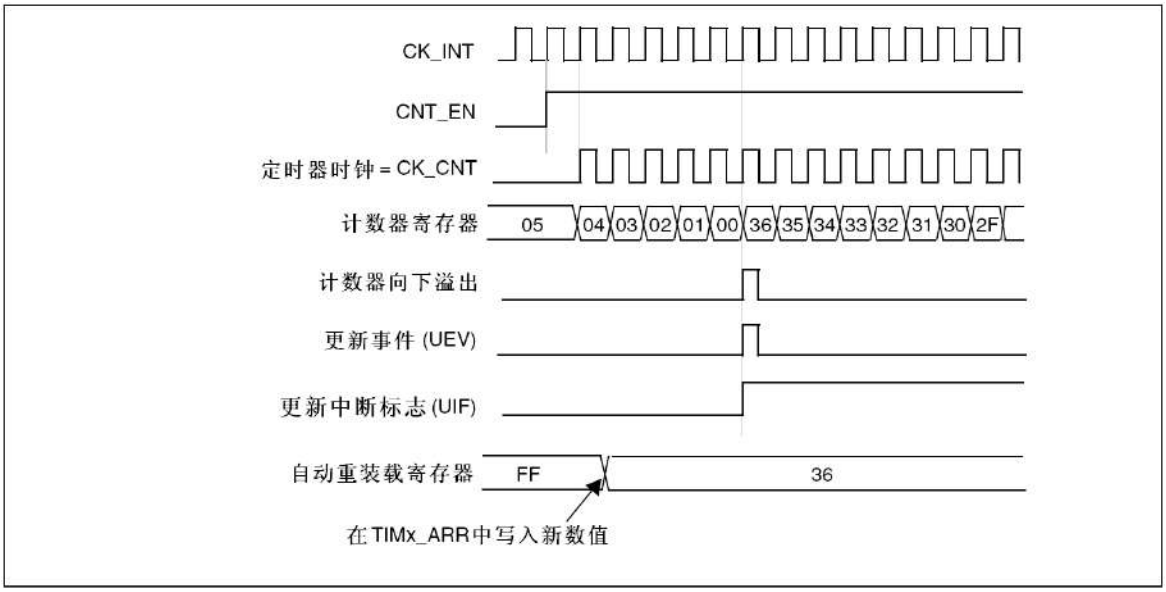


图 19-13 计数器时序图，当没有使用重复计数器时的更新事件



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。

可以在每次计数上溢和每次计数下溢时产生更新事件；也可以通过(软件或者使用从模式控制器) 设置 TIMx_EGR 寄存器中的 UG 位产生更新事件。然后，计数器重新从 0 开始计数，预分频器也 重新从 0 开始计数。

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

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

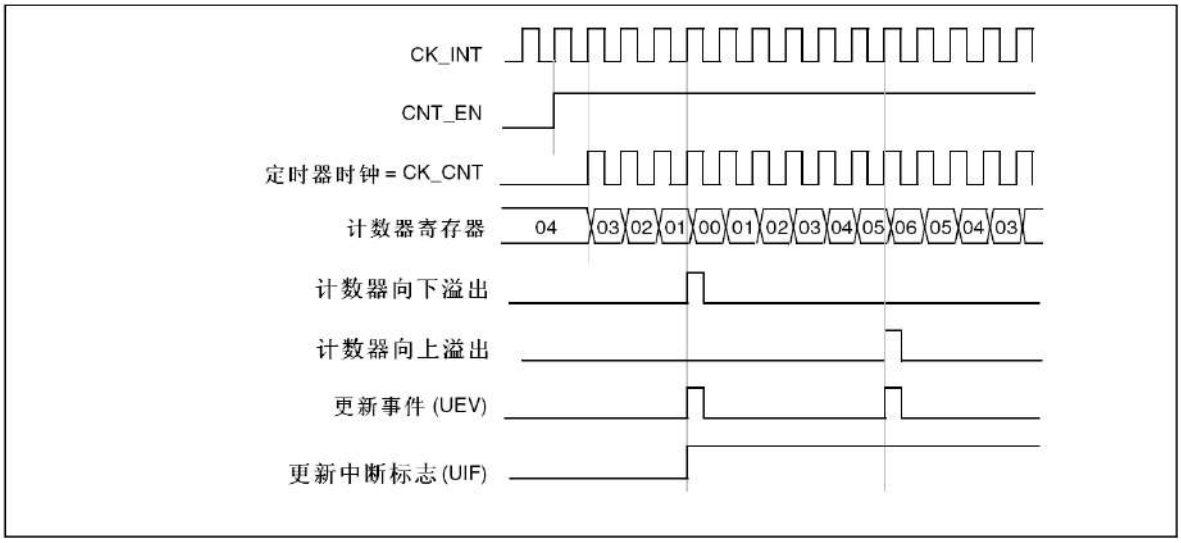
当发生更新事件时，所有的寄存器都被更新，并且(根据 URS 位的设置)更新标志位(TIMx_SR 寄 存器中

的 UIF 位)也被设置。

- 预分频器的缓存器被加载为预装载(TIMx_PSC 寄存器)的值。
- 当前的自动加载寄存器被更新为预装载值(TIMx_ARR 寄存器中的内容)。注：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 19-145 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 TIMx_CR1 章节)。

图 19-15 计数器时序图，内部时钟分频因子为 2

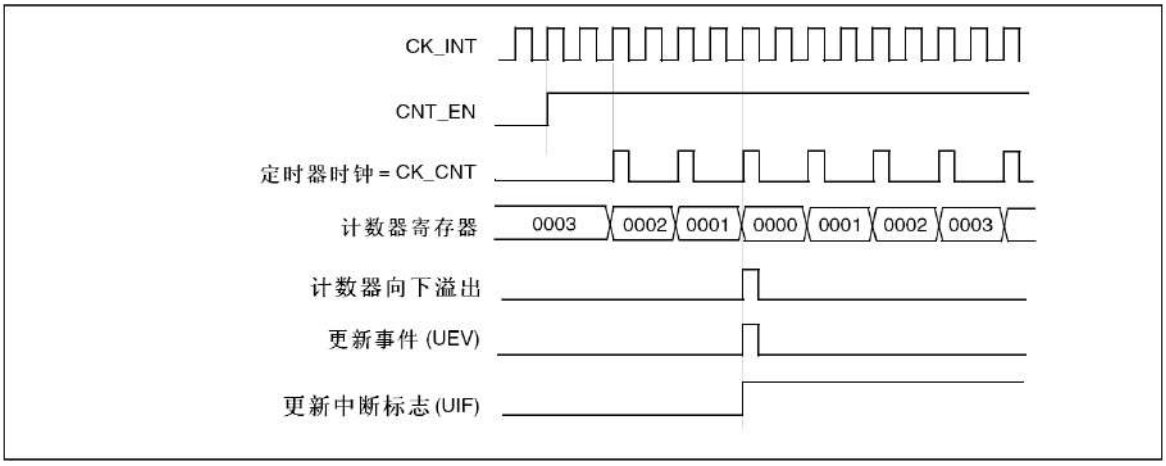


图 19-16 计数器时序图，内部时钟分频因子为 4，TIMx_ARR=0x36

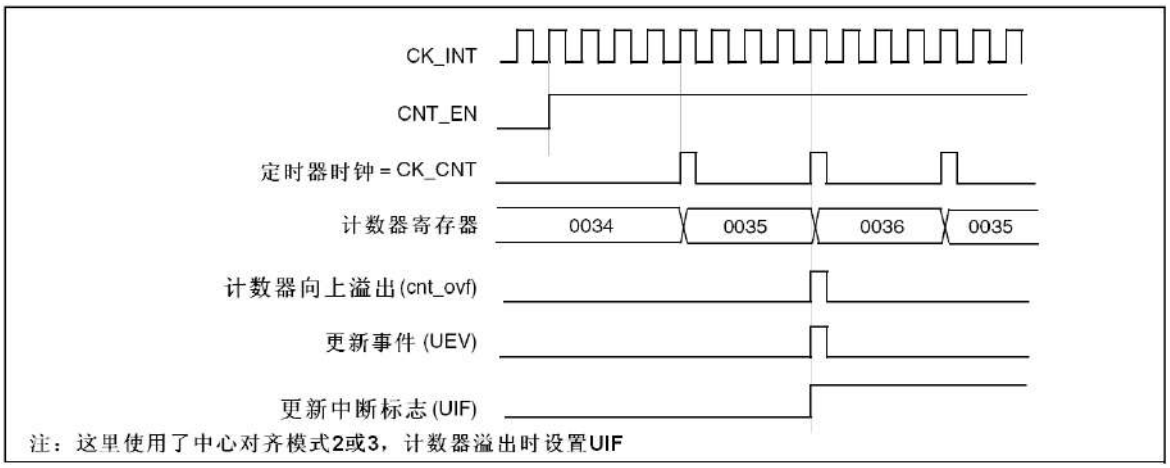


图 19-17 计数器时序图，内部时钟分频因子为 N

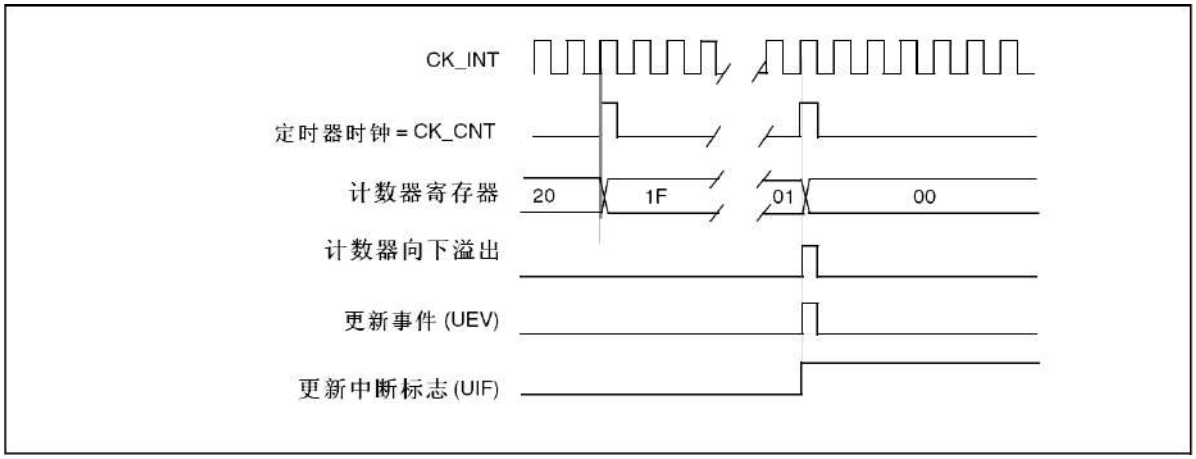


图 19-18 计数器时序图，ARPE=1 时的更新事件(计数器下溢)

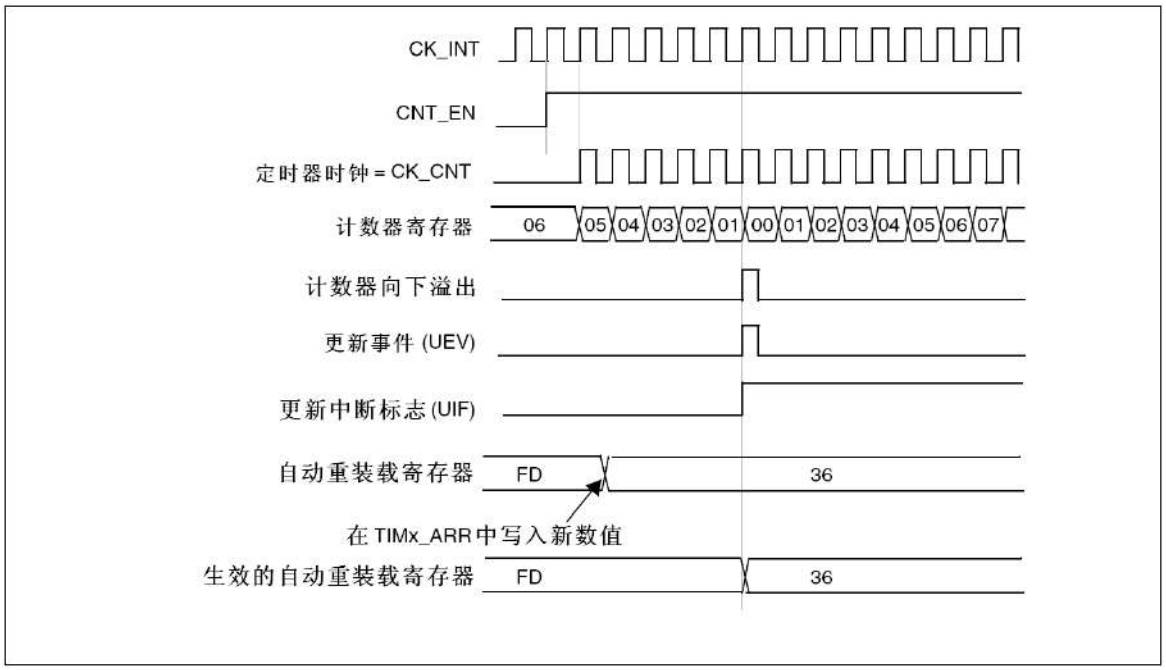
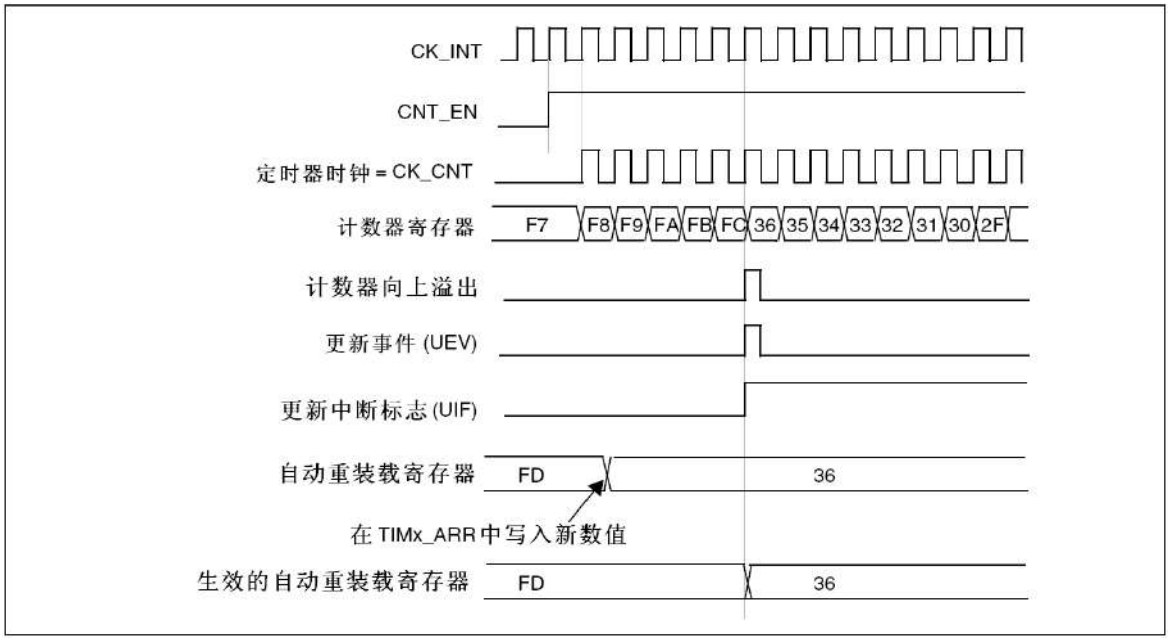


图 20 计数器时序图，ARPE=1 时的更新事件(计数器溢出)



19.3.3 时钟选择

计数器时钟可由下列时钟源提供：

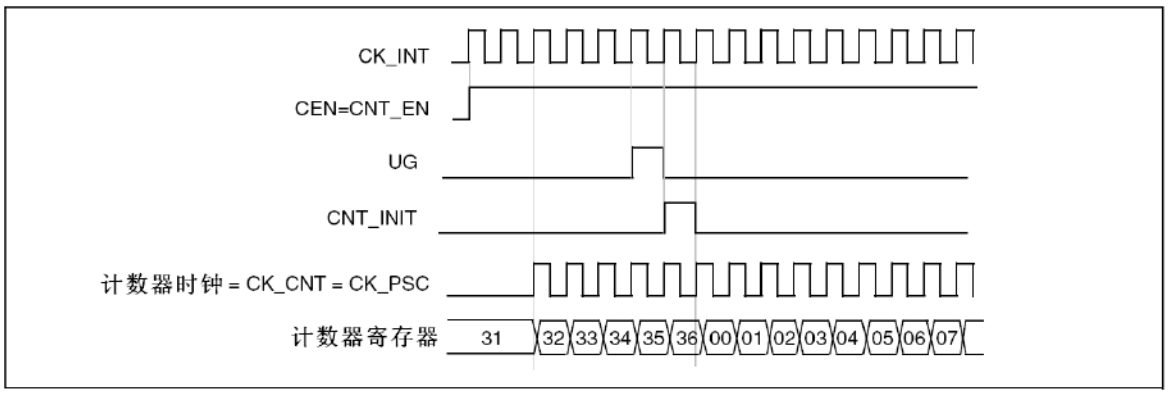
- 内部时钟(CK_INT)
- 外部时钟模式 1：外部输入脚(TIx)
- 外部时钟模式 2：外部触发输入(ETR)
- 内部触发输入(ITRx)：使用一个定时器作为另一个定时器的预分频器，如可以配置一个定时 器 Timer1 而作为另一个定时器 Timer2 的预分频器。参见定时器同步。

内部时钟源(CK_INT)

如果禁止了从模式控制器(TIMx_SMCR 寄存器的 SMS=000)，则 CEN、DIR(TIMx_CR1 寄存器) 和 UG 位(TIMx_EGR 寄存器)是事实上的控制位，并且只能被软件修改(UG 位仍被自动清除)。只要 CEN 位被写成’1’，预分频器的时钟就由内部时钟 CK_INT 提供。

下图显示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。

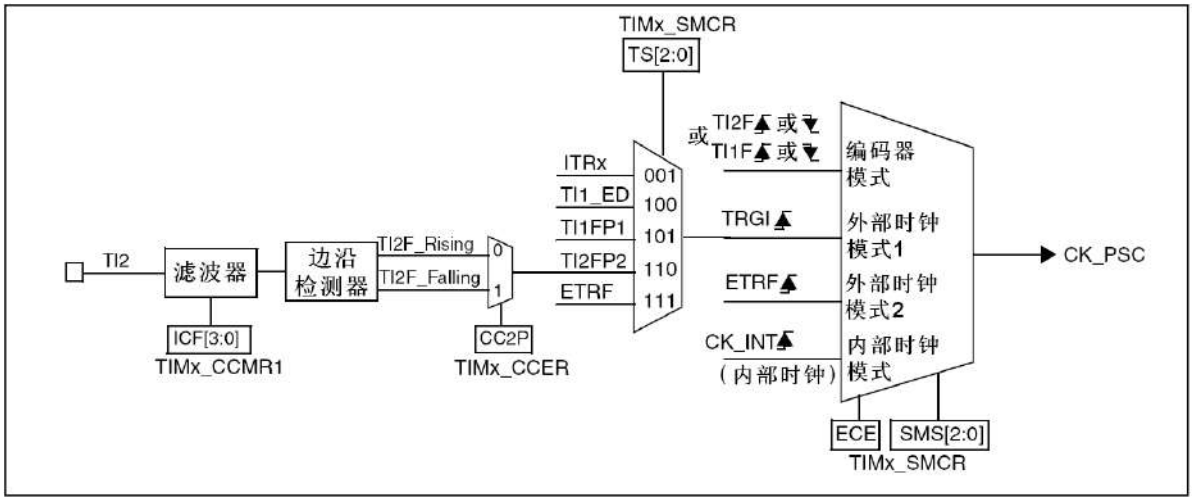
图 21 一般模式下的控制电路，内部时钟分频因子为 1



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 19-19 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)

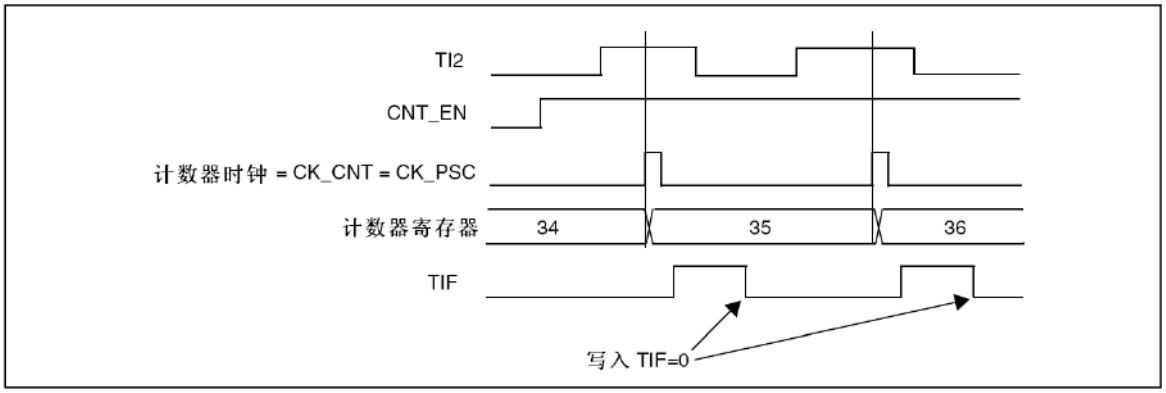
注：　　捕获预分频器不用作触发，所以不需要对它进行配置

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 23 外部时钟模式 1 下的控制电路



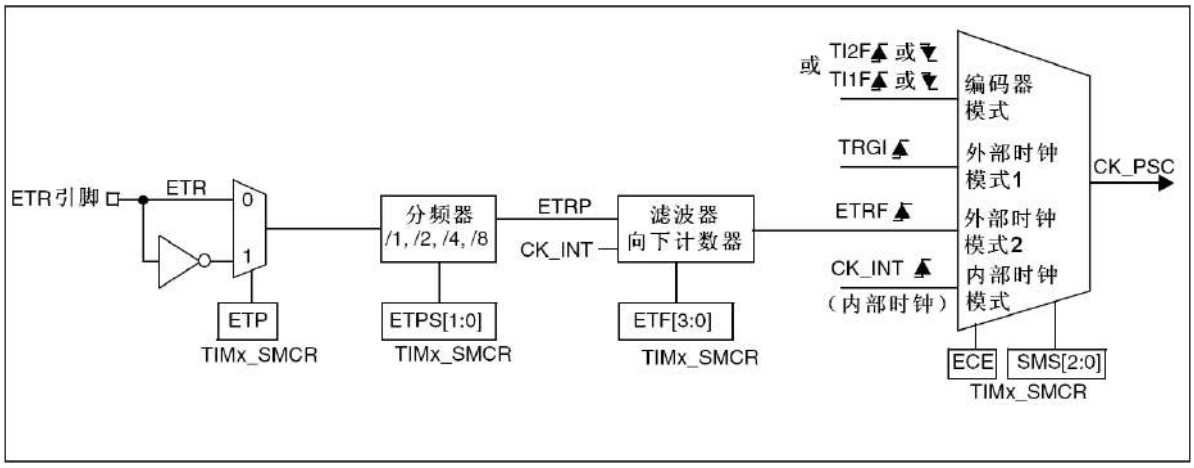
外部时钟源模式 2

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框图

图 24 外部触发输入框图



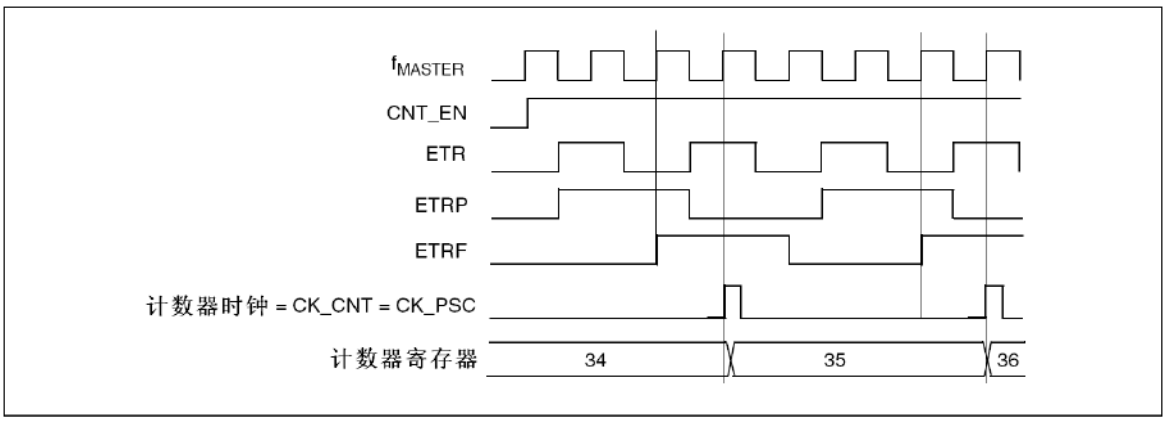
例如，要配置在 ETR 下每 2 个上升沿计数一次的向上计数器，使用下列步骤：

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

计数器在每 2 个 ETR 上升沿计数一次。

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

图 25 外部时钟模式 2 下的控制电路



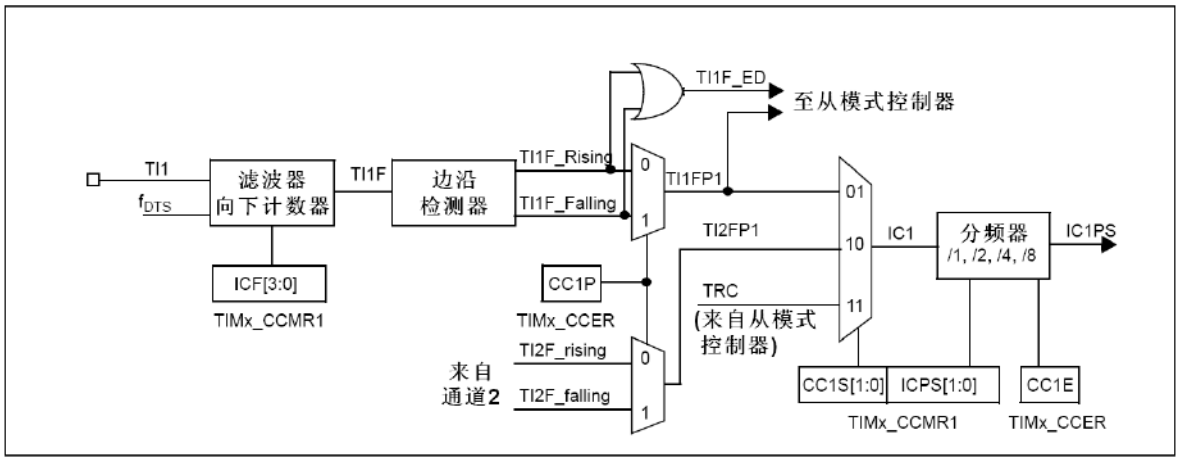
19.3.4 捕获/比较通道

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 边缘检测器产生一个信号(TIxFPx)，它可以作为从模式控制器的输入触发或者作为捕获控制。该 信号通过预分频进入捕获寄存器(ICxPS)。

图 26 捕获/比较通道(如：通道 1 输入部分)



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

图 27 捕获/比较通道 1 的主电路

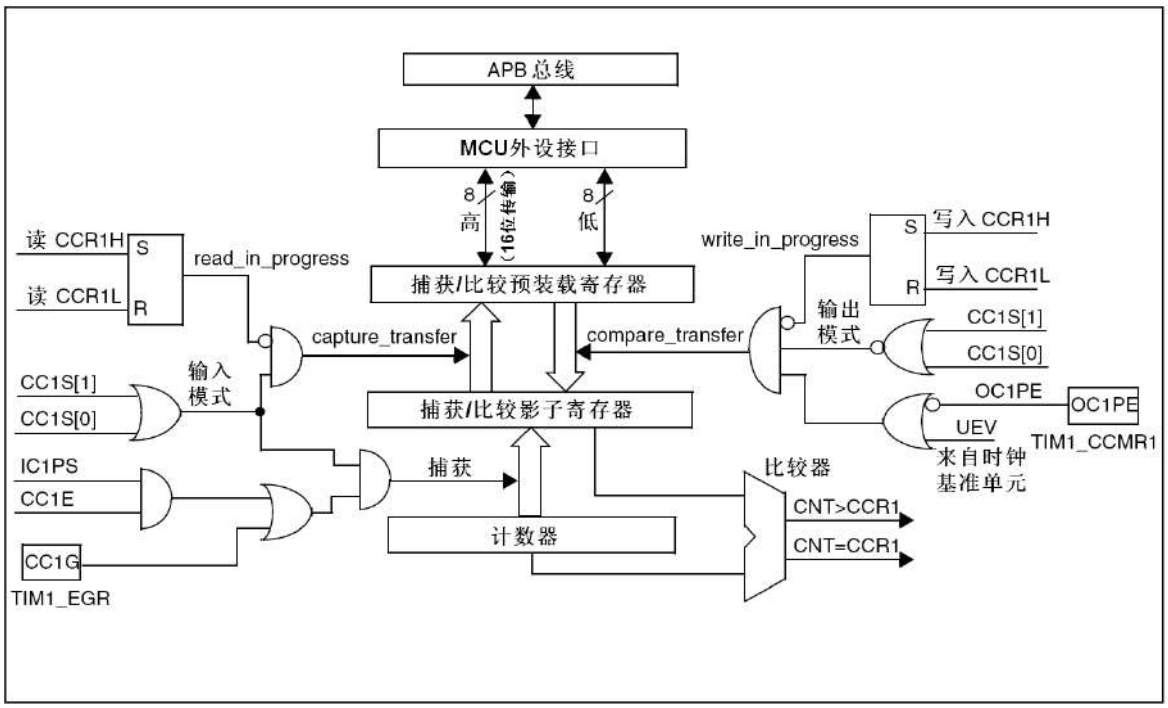
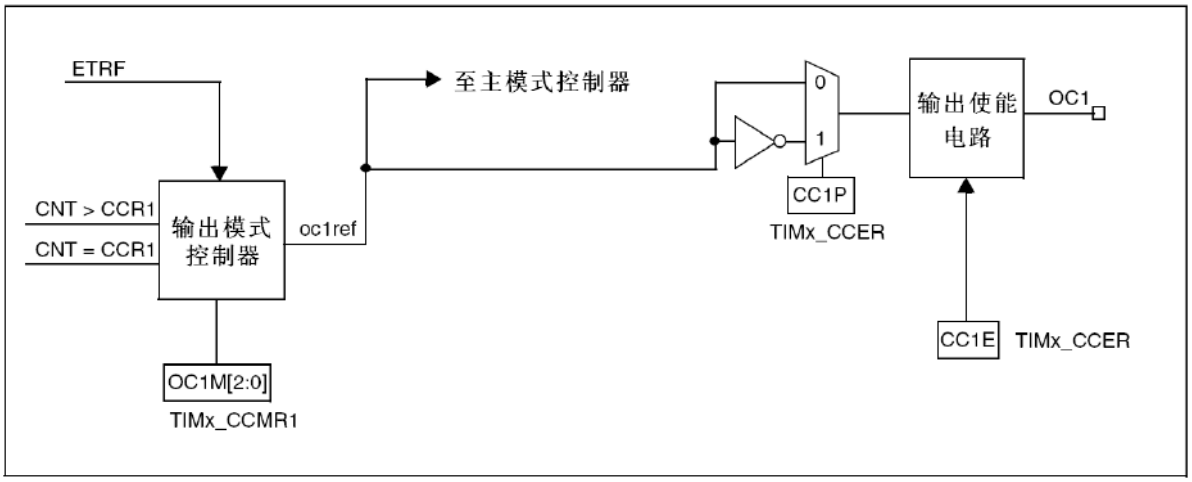


图 28 捕获/比较通道的输出部分(通道 1)



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

在捕获模式下，捕获发生在影子寄存器上，然后再复制到预装载寄存器中。

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

19.3.5 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'，如果使能 了中断或者 DMA 操作，则将产生中断或者 DMA 操作。如果捕获事件发生时 CCxIF 标志已经为 高，那么重复捕获标志 CCxOF(TIMx_SR 寄存器)被置'1'。写 CCxIF=0 可清除 CCxIF，或读取存 储在 TIMx_CCRx 寄存器中的捕获数据也可清除 CCxIF。写 CCxOF=0 可清除 CCxOF。

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

- 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，只要 CC1S 不为'00'，通道被配置为输入，并且 TM1_CCR1 寄存器变为只读。
- 根据输入信号的特点，配置输入滤波器为所需的带宽(即输入为 TIx 时，输入滤波器控制位是 TIMx_CCMRx 寄存器中的 ICxF 位)。假设输入信号在最多 5 个内部时钟周期的时间内抖动，我们须配置滤波器的带宽长于 5 个时钟周期。因此我们可以(以 fDTS 频率)连续采样 8 次，以确 认在 TI1 上一次真实的边沿变换，即在 TIMx_CCMR1 寄存器中写入 IC1F=0011。
- 选择 TI1 通道的有效转换边沿，在 TIMx_CCER 寄存器中写入 CC1P=0(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：

- 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。

19.3.6 PWM 输入模式

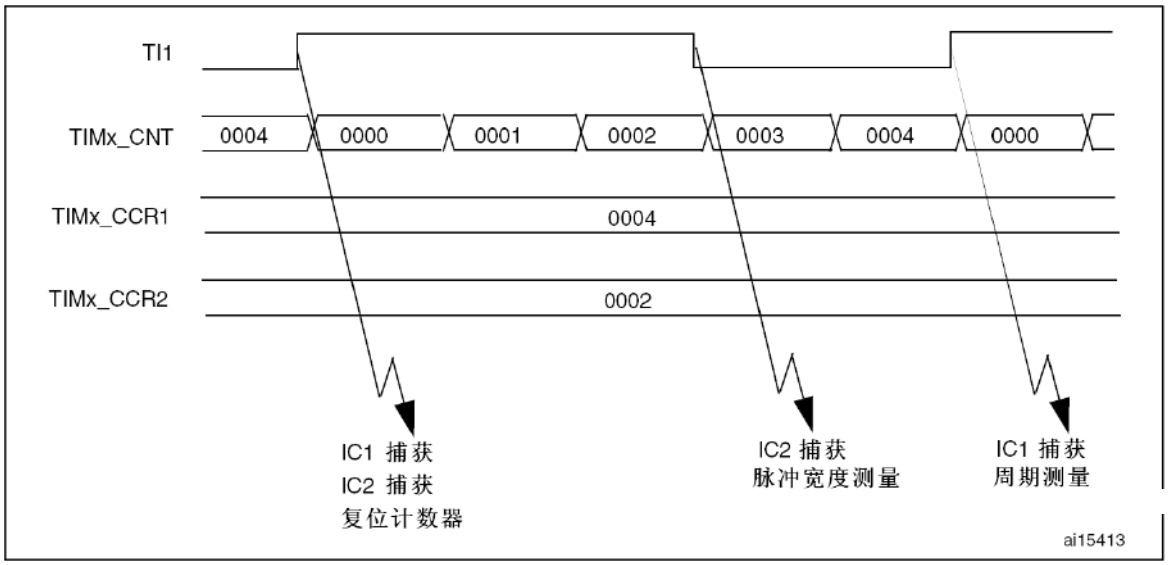
该模式是输入捕获模式的一个特例，除下列区别外，操作与输入捕获模式相同：

- 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 29 PWM 输入模式时序



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

19.3.7 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。

19.3.8 输出比较模式

此项功能是用来控制一个输出波形，或者指示一段给定的的时间已经到时。 当计数器与捕获/比较寄存器的内容相同时，输出比较功能做如下操作：

将输出比较模式(TIMx_CCMRx 寄存器中的 OCxM 位)和输出极性(TIMx_CCER 寄存器中的 CCxP 位)定义的值输出到对应的引脚上。在比较匹配时，输出引脚可以保持它的电平 (OCxM=000)、被设置成有效电平(OCxM=001)、被设置成无效电平(OCxM=010)或进行翻 转(OCxM=011)。

- 设置中断状态寄存器中的标志位(TIMx_SR 寄存器中的 CCxIF 位)。
- 若设置了相应的中断屏蔽(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。

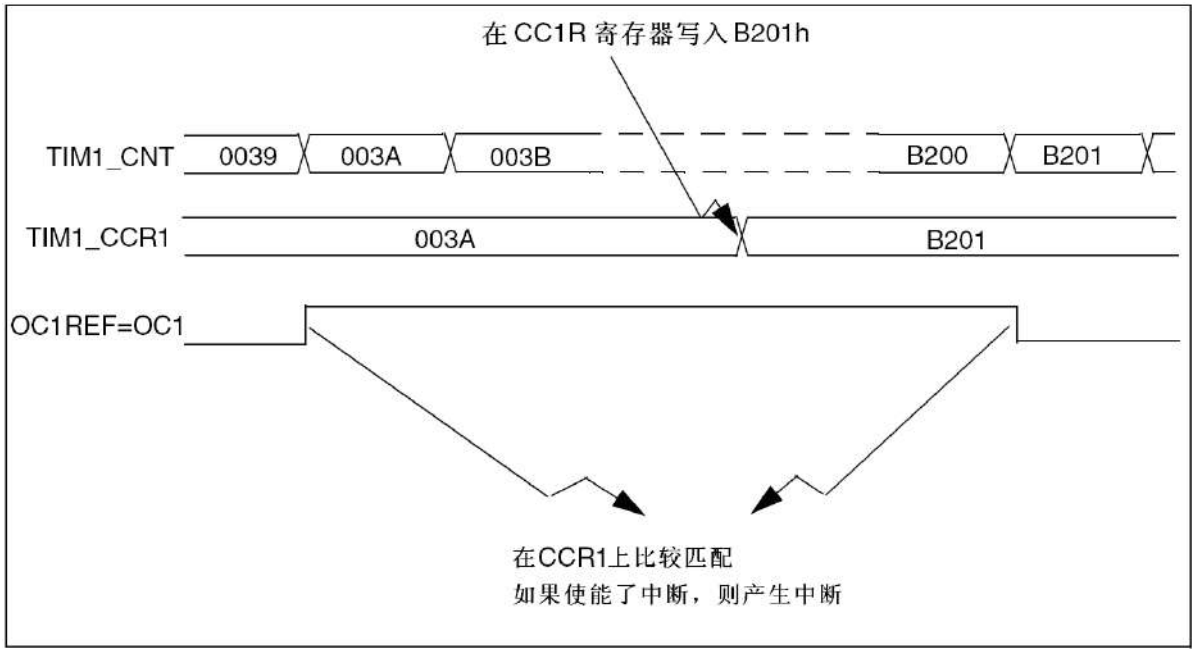
同步的精度可以达到计数器的一个计数周期。输出比较模式(在单脉冲模式下)也能用来输出一个 单脉冲。

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，例如当计数器 CNT 与 CCRx 匹配时翻转 OCx 的输出引脚，CCRx 预装载未 用，开启 OCx 输出且高电平有效，则必须设置 OCxM='011'、OCxPE='0'、CCxP='0'和 CCxE='1'。
5. 设置 TIMx_CR1 寄存器的 CEN 位启动计数器

TIMx_CCRx 寄存器能够在任何时候通过软件进行更新以控制输出波形，条件是未使用预装载寄 存器(OCxPE='0'，否则 TIMx_CCRx 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 19-20 输出比较模式，翻转 OC1



19.3.9 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，(在向上计数或中心对称模式中)使能 自动重装载的预装载寄存器。

仅当发生一个更新事件的时候，预装载寄存器才能被传送到影子寄存器，因此在计数器开始计 数之前，必须通过设置 TIMx_EGR 寄存器中的 UG 位来初始化所有的寄存器。 OCx 的极性可以通过软件在 TIMx_CCER 寄存器中的 CCxP 位设置，它可以设置为高电平有效或

低电平有效。TIMx_CCER 寄存器中的 CCxE 位控制 OCx 输出使能。详见 TIMx_CCERx 寄存器的描述。

在 PWM 模式(模式 1 或模式 2)下，TIMx_CNT 和 TIMx_CCRx 始终在进行比较，(依据计数器的计数 方向) 以确 定是否符合 TIMx_CCRx ≤ TIMx_CNT 或者 TIMx_CNT ≤ TIMx_CCRx 。然而 为了 与 OCREF_CLR 的功能(在下一个 PWM 周期之前，ETR 信号上的一个外部事件能够清除 OCxREF) 一致，OCxREF 信号只能在下述条件下产生：

- 当比较的结果改变，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。

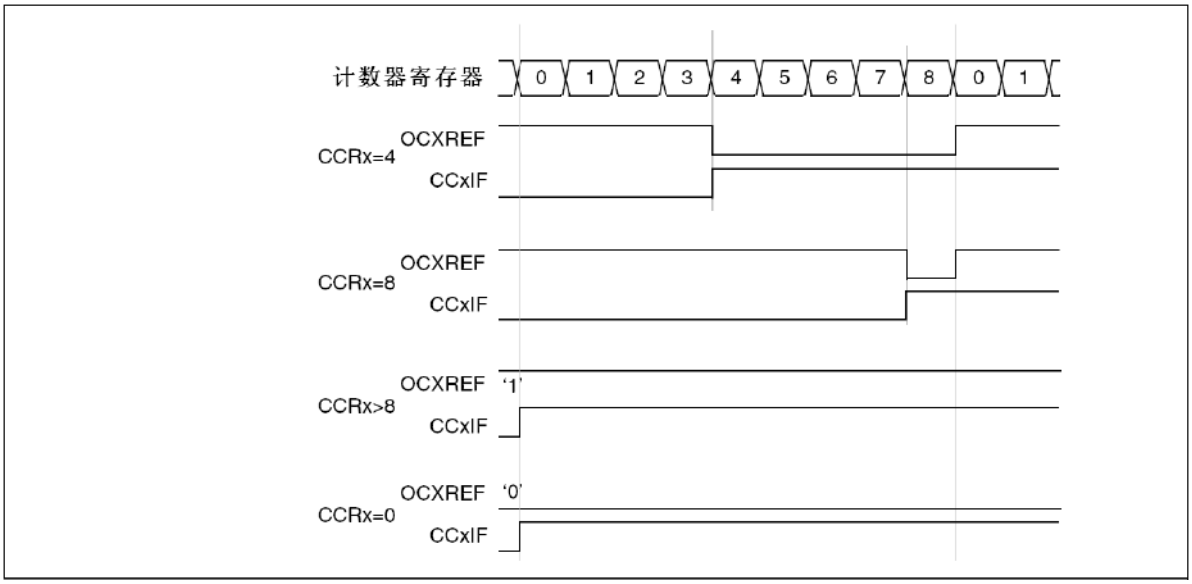
PWM 边沿对齐模式

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。参看章节计数器模式。

下面是一个 PWM 模式 1 的例子。当 TIMx_CNT<TIMx_CCRx 时 PWM 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 31 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。参看章节计数器模式节。

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。

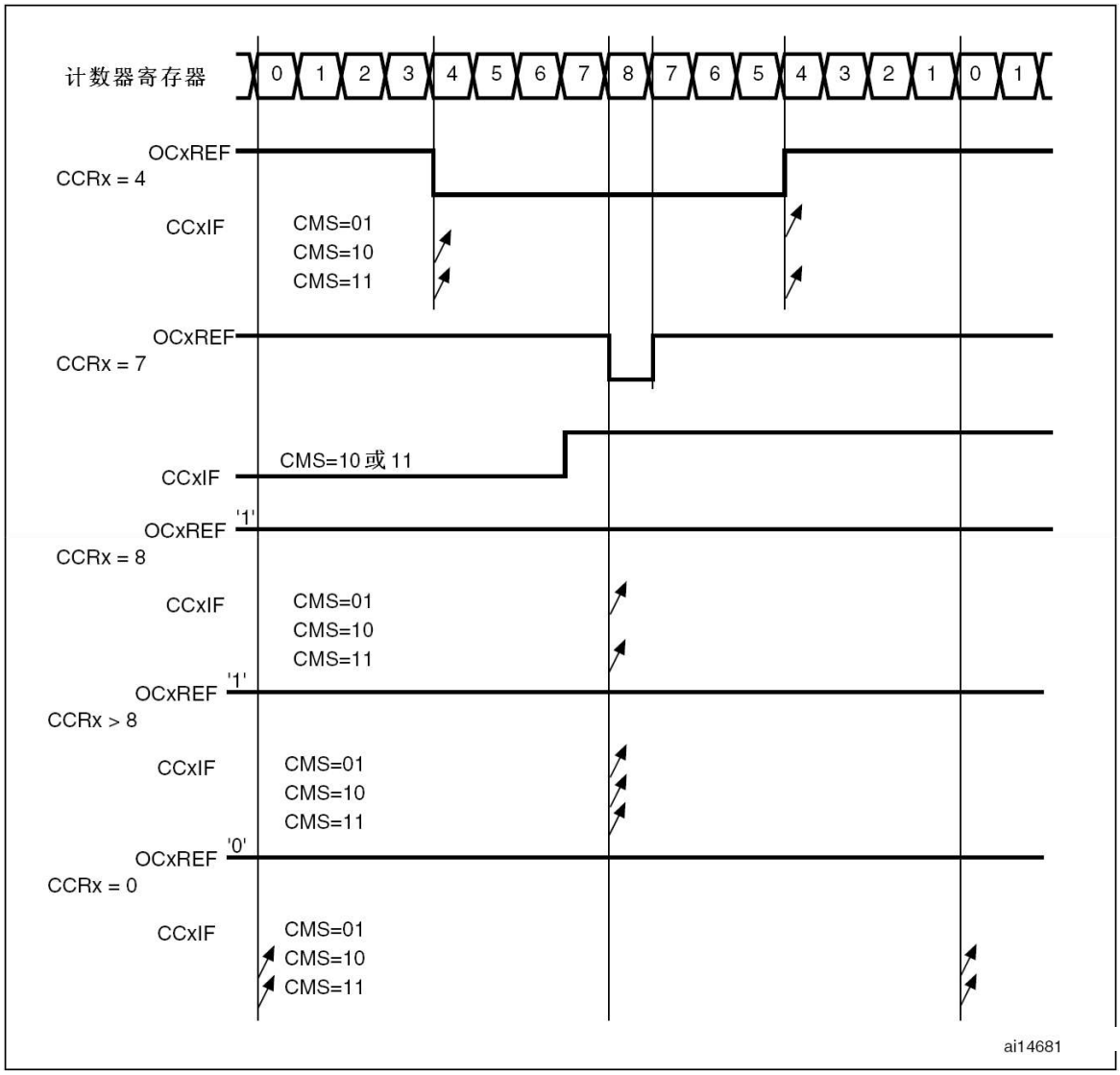
PWM 中央对齐模式

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

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 19-21 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，使用当前的向上/向下计数配置；这就意味着计数器向上还是向下计 数

取决于 TIMx_CR1 寄存器中 DIR 位的当前值。此外，软件不能同时修改 DIR 和 CMS 位。

- 不推荐当运行在中央对齐模式时改写计数器，因为这会产生不可预知的结果。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。

19.3.10 单脉冲模式

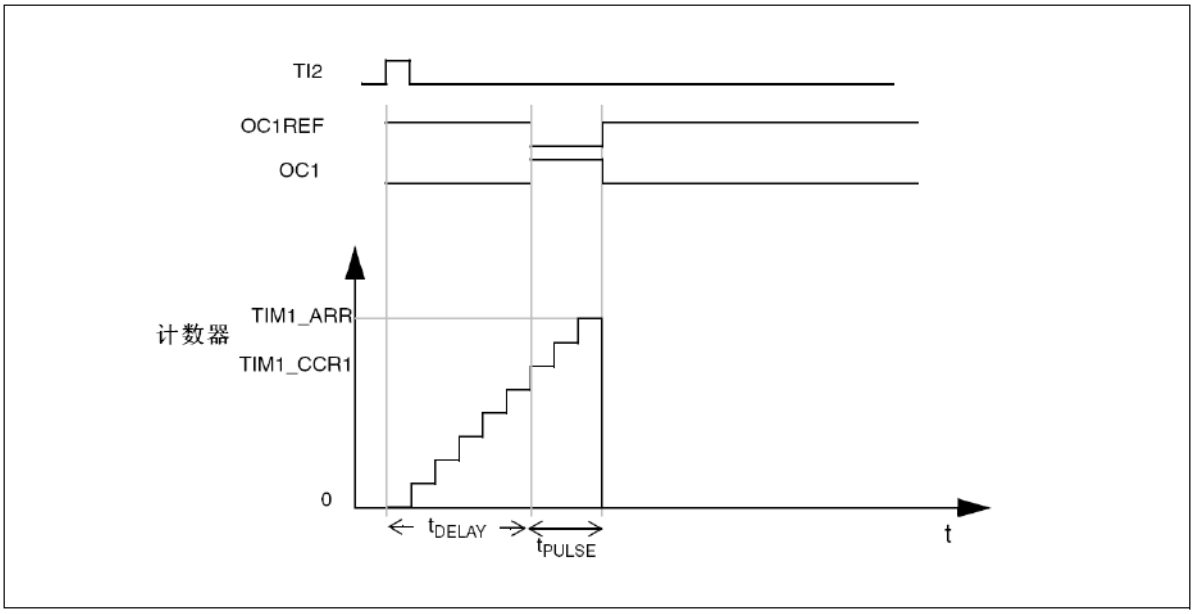
单脉冲模式(OPM)是前述众多模式的一个特例。这种模式允许计数器响应一个激励，并在一个 程序可控的延时之后，产生一个脉宽可程序控制的脉冲。

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 33 单脉冲模式的例子



例如，你需要在从 TI2 输入脚上检测到一个上升沿开始，延迟 tDELAY 之后，在 OC1 上产生一个长 度为 tPULSE 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)

- tDELAY 由写入 TIMx_CCR1 寄存器中的值定义。
- tPULSE 由自动装载值和比较值之间的差值定义(TIMx_ARR - TIMx_CCR1)。
- 假定当发生比较匹配时要产生从'0'到'1'的波形，当计数器到达预装载值时要产生一个从'1' 到'

0’的波形；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 tDELAY。

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

19.3.11 在外部事件时清除 OCxREF 信号

对于一个给定的通道，设置 TIMx_CCMRx 寄存器中对应的 OCxCE 位为’1’，能够用 ETRF 输入端 的高电平把 OCxREF 信号拉低，OCxREF 信号将保持为低直到发生下一次的更新事件 UEV。

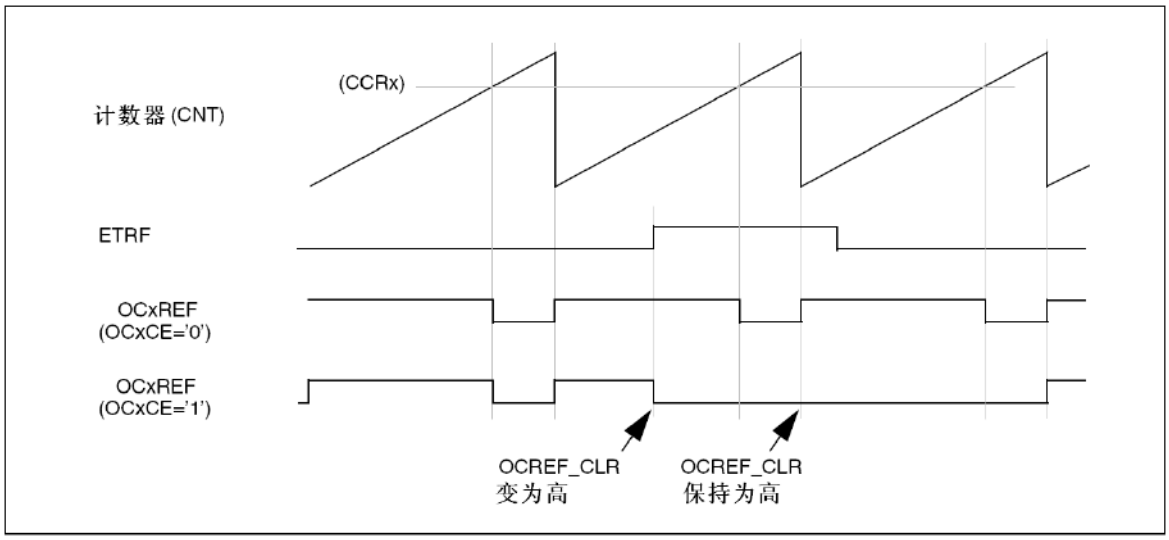
该功能只能用于输出比较和 PWM 模式，而不能用于强置模式。

例如，OCxREF 信号可以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 34 清除 TIMx 的 OCxREF



19.3.12 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 55，假定计数器已经启动(TIMx_CR1 寄存器中的 CEN='1')，计数器由每次在 TI1FP1 或 TI2FP2 上的有效跳变驱动。TI1FP1 和 TI2FP2 是 TI1 和 TI2 在通过输入滤波器和极性控制后的信号；如果没有滤波和变相，则 TI1FP1=TI1， TI2FP2=TI2。根据两个输入信号的跳变顺序，产生了计数脉冲和方向信号。依据两个输入信号 的跳变顺序，计数器向上或向下计数，同时硬件对 TIMx_CR1 寄存器的 DIR 位进行相应的设置。 不管计数器是依靠 TI1 计数、依靠 TI2 计数或者同时依靠 TI1 和 TI2 计数。在任一输入端(TI1 或者 TI2)的跳变都会重新计算 DIR 位。

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

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

表 35 计数方向与编码器信号的关系

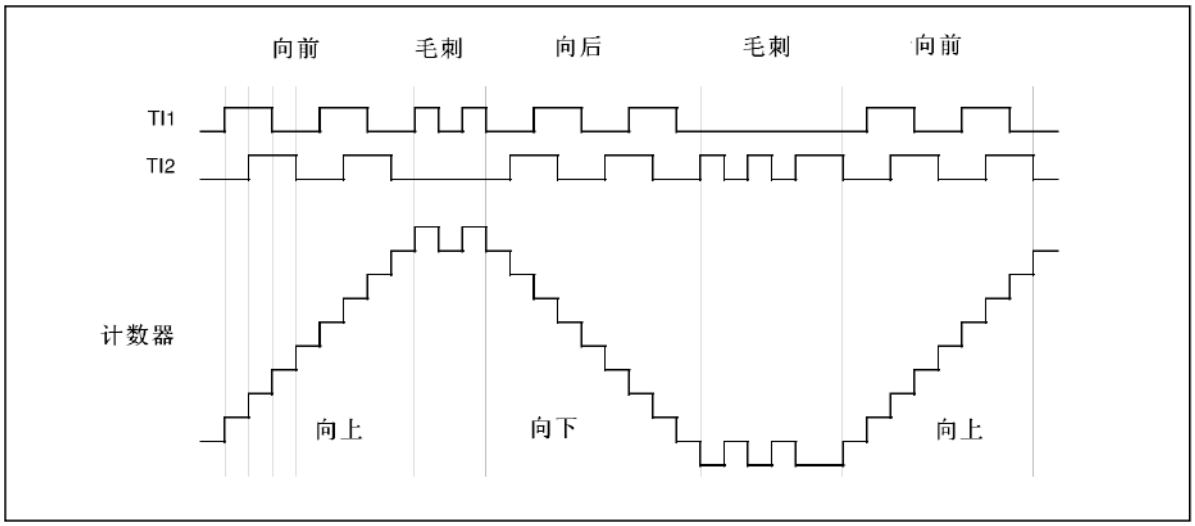
有效边沿	相对信号的电平(TI1FP1 对应 TITI2, TI2FP2 对应 TI1)	TI1FP1 信号		TI2FP2 信号	
		上升	下降	上升	下降
仅在 TI1 计数	高	向下计数	向上计数	不计数	不计数
	低	向上计数	向下计数	不计数	不计数
仅在 TI2 计数	高	不计数	不计数	向上计数	向下计数
	低	不计数	不计数	向下计数	向上计数
在 TI1 和 TI2 上计数	高	向下计数	向上计数	向上计数	向下计数
	低	向上计数	向下计数	向下计数	向上计数

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，一般会使用比较器 将编码器的差动输出转换到数字信号，这大大增加了抗噪声干扰能力。编码器输出的第三个信 号表示机械零点，可以把它连接到一个外部中断输入并触发一个计数器复位。

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

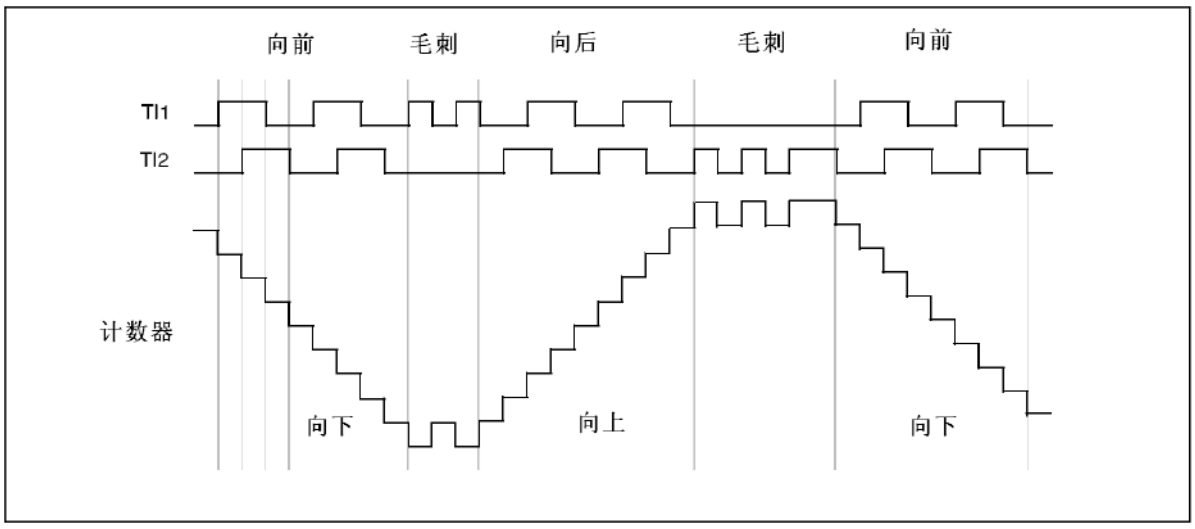
- CC1S='01' (TIMx_CCMR1 寄存器，IC1FP1 映射到 TI1)
- CC2S='01' (TIMx_CCMR2 寄存器，IC2FP2 映射到 TI2)
- CC1P='0' (TIMx_CCER 寄存器，IC1FP1 不反相，IC1FP1=TI1)
- CC2P='0' (TIMx_CCER 寄存器，IC2FP2 不反相，IC2FP2=TI2)
- SMS='011' (TIMx_SMCR 寄存器，所有的输入均在上升沿和下降沿有效).
- CEN='1' (TIMx_CR1 寄存器，计数器使能)

图 36 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 37 IC1FP1 反相的编码器接口模式实例



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

19.3.13 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 TIMx_BDTR 节给出了此特性用于连接霍尔传感器的例子。

19.3.14 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，还会产生一个更新事件 UEV；然后所有的预装载寄存器(TIMx_ARR， TIMx_CCRx)都会被更新。

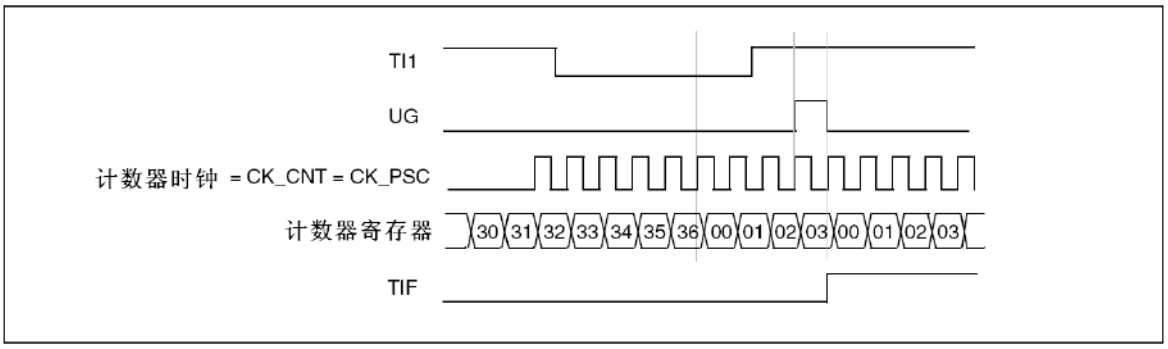
在下面的例子中，TI1 输入端的上升沿导致向上计数器被清零：

- 配置通道 1 以检测 TI1 的上升沿。配置输入滤波器的带宽(在本例中，不需要任何滤波器，因此保持 IC1F=0000)。触发操作中不使用捕获预分频器，所以不需要配置它。CC1S 位只选 择输入捕获源，即 TIMx_CCMR1 寄存器中 CC1S=01。置 TIMx_CCER 寄存器中 CC1P=0 以 确定极性(只检测上升沿)。
- 置 TIMx_SMCR 寄存器中 SMS=100，配置定时器为复位模式；置 TIMx_SMCR 寄存器中 TS=101，选择 TI1 作为输入源。
- 置 TIMx_CR1 寄存器中 CEN=1，启动计数器。

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

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

图 38 复位模式下的控制电路



从模式：门控模式

按照选中的输入端电平使能计数器。

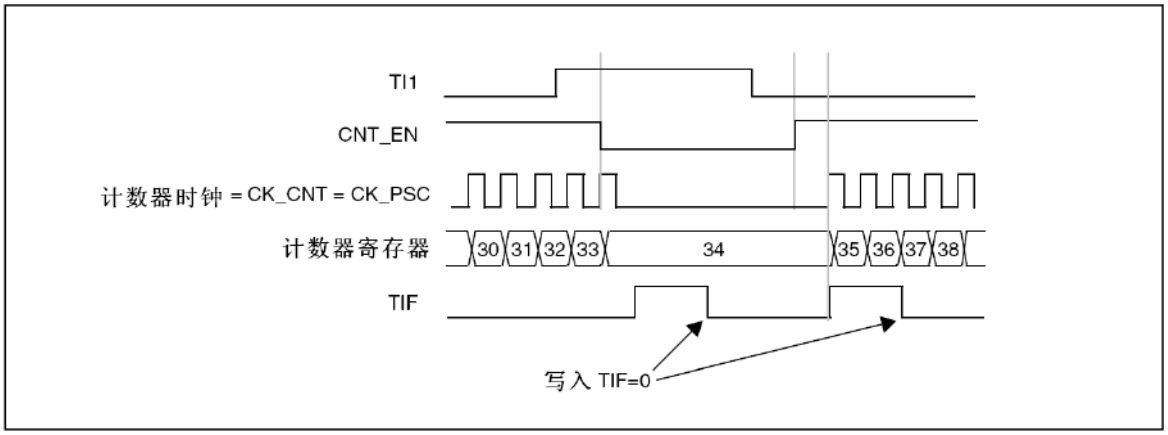
在如下的例子中，计数器只在 TI1 为低时向上计数：

- 配置通道 1 以检测 TI1 上的低电平。配置输入滤波器带宽(本例中，不需要滤波，所以保持 IC1F=0000)。触发操作中不使用捕获预分频器，所以不需要配置。CC1S 位用于选择输入 捕获源，置 TIMx_CCMR1 寄存器中 CC1S=01。置 TIMx_CCER 寄存器中 CC1P=1 以确定极 性(只检测低电平)。
- 置 TIMx_SMCR 寄存器中 SMS=101，配置定时器为门控模式；置 TIMx_SMCR 寄存器中 TS=101，选择 TI1 作为输入源。
- 置 TIMx_CR1 寄存器中 CEN=1，启动计数器。在门控模式下，如果 CEN=0，则计数器不能启动，不论触发输入电平如何。

只要 TI1 为低，计数器开始依据内部时钟计数，在 TI1 变高时停止计数。当计数器开始或停止时 都设置 TIMx_SR 中的 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 39 门控模式下的控制电路



从模式：触发模式

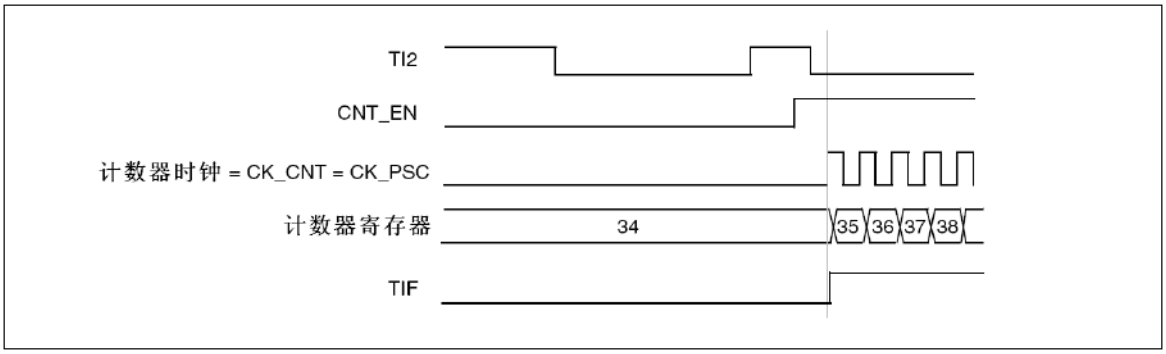
输入端上选中的事件使能计数器。

在下面的例子中，计数器在 TI2 输入的上升沿开始向上计数：

- 配置通道 2 检测 TI2 的上升沿。配置输入滤波器带宽(本例中，不需要任何滤波器，保持 IC2F =0000)。触发操作中不使用捕获预分频器，不需要配置。CC2S 位只用于选择输入捕 获源，置 TIMx_C CMR1 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 4019-22 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。

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

1. 通过 TIMx_SMCR 寄存器配置外部触发输入电路：

— ETF=0000：没有滤波

— ETPS=00：不用预分频器

— ETP=0：检测 ETR 的上升沿，置 ECE=1 使能外部时钟模式

2. 按如下配置通道 1，检测 TI 的上升沿：

— IC1F=0000：没有滤波

— 触发操作中不使用捕获预分频器，不需要配置

— 置 TIMx_CCMR1 寄存器中 CC1S=01，选择输入捕获源

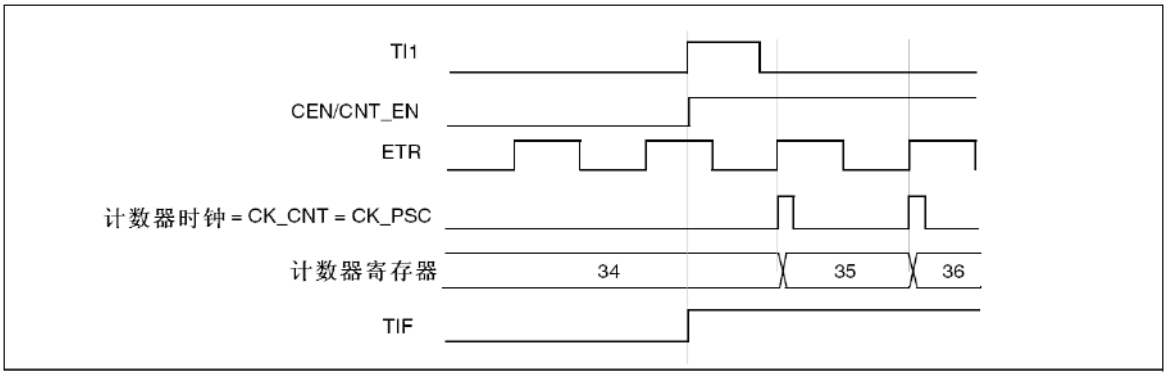
– 置 TIMx_CCER 寄存器中 CC1P=0 以确定极性(只检测上升沿)

3. 置 TIMx_SMCR 寄存器中 SMS=110，配置定时器为触发模式。置 TIMx_SMCR 寄存器中 TS=101，选择 TI1 作为输入源。

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

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

图 41 外部时钟模式 2＋触发模式下的控制电路



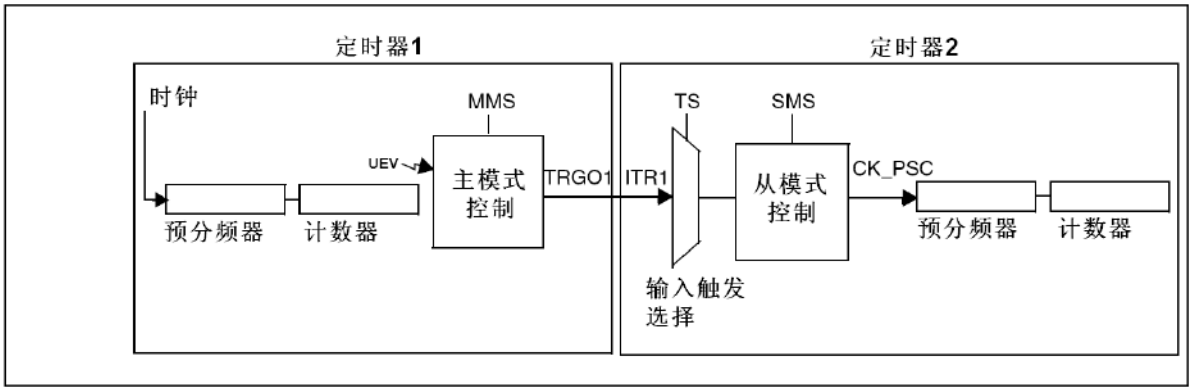
19.3.15 定时器同步

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 42 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。参考上图，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

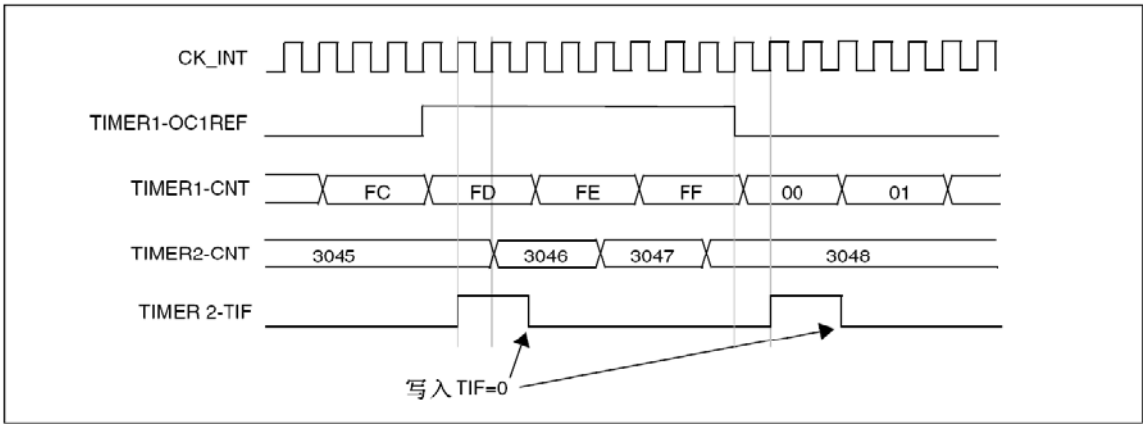
在这个例子中，定时器 2 的使能由定时器 1 的输出比较控制。参考图主/从定时器的例子的连接。只当定

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(fCK_CNT=fCK_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=000)
- 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 43 定时器 1 的 OC1REF 控制定时器 2

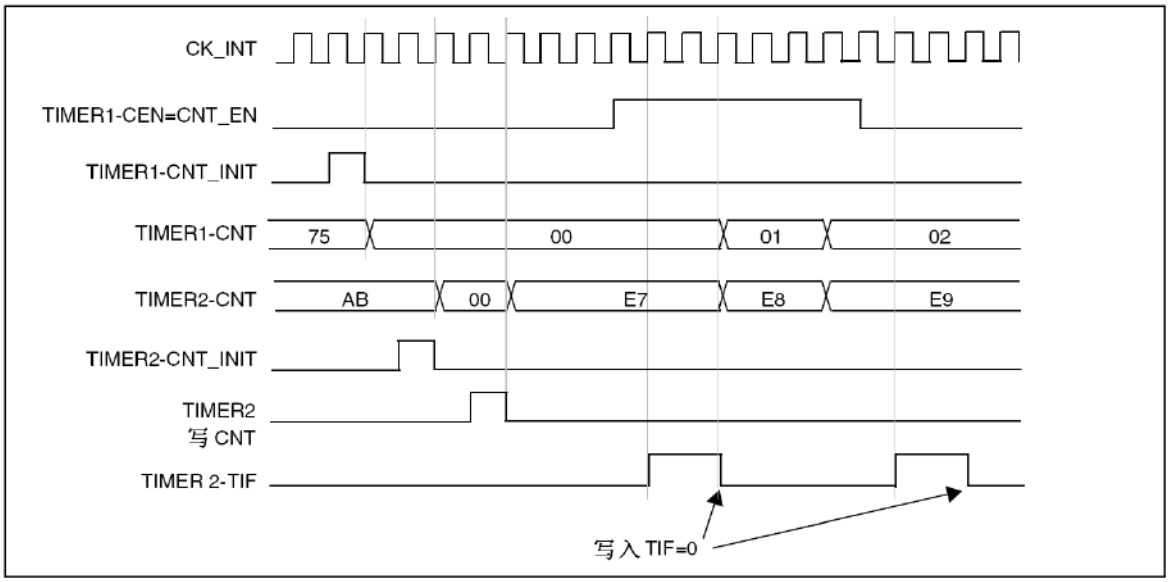


在上图的例子中，在定时器 2 启动之前，它们的计数器和预分频器未被初始化，因此它们从当 前的数值开始计数。可以在启动定时器 1 之前复位 2 个定时器，使它们从给定的数值开始，即在 定时器计数器中写入需要的任意数值。写 TIMx_EGR 寄存器的 UG 位即可复位定时器。

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 44 通过使能定时器 1 可以控制定时器 2

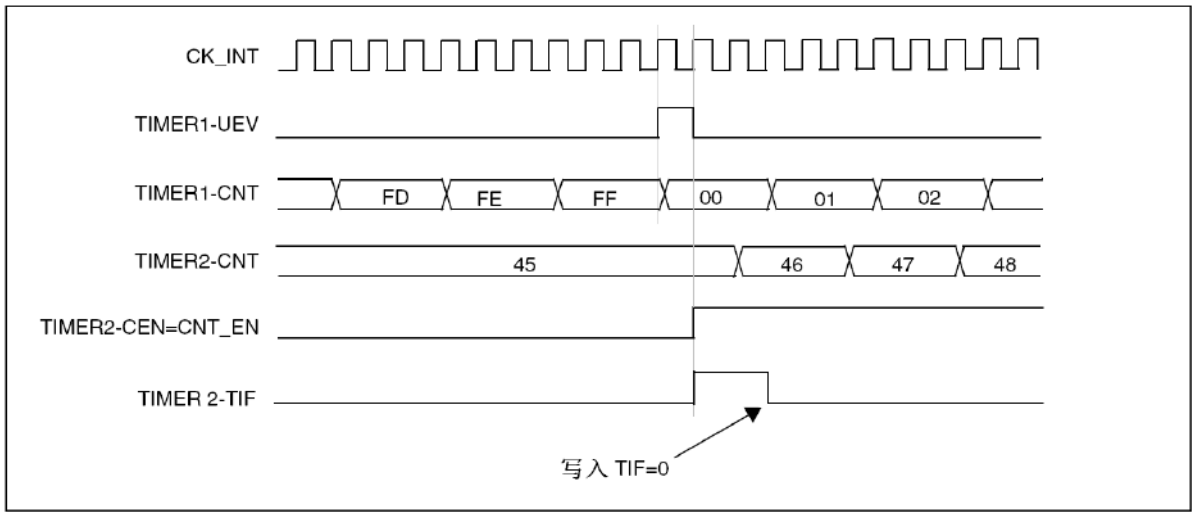


使用一个定时器去启动另一个定时器

在这个例子中，使用定时器 1 的更新事件使能定时器 2。参考图主/从定时器的例子的连接。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(fCK_CNT=fCK_INT/3)。

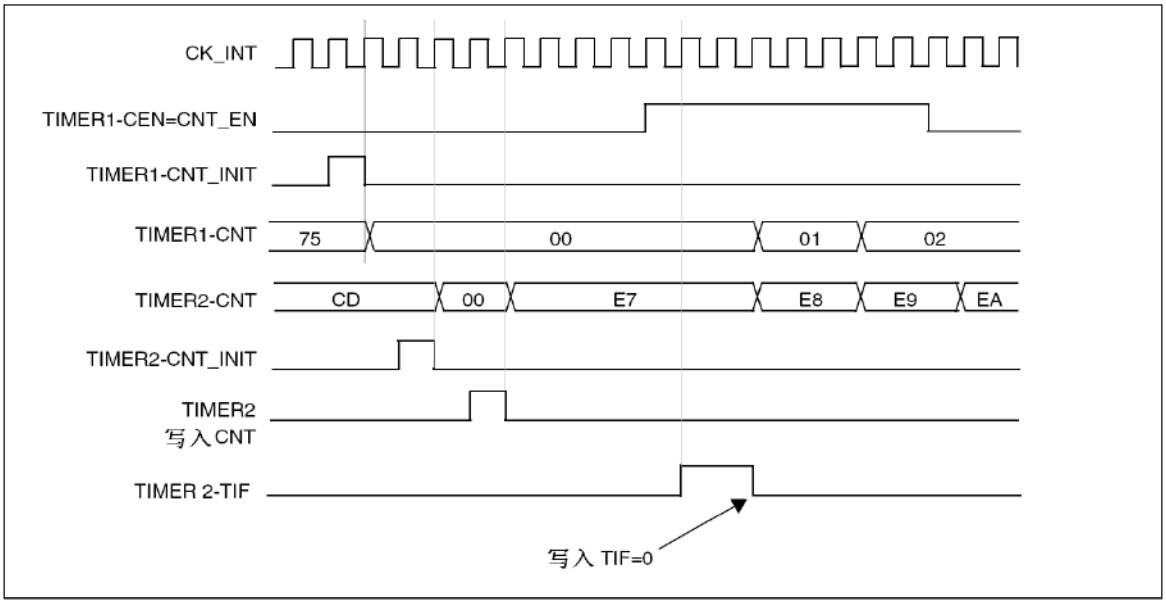
- 配置定时器 1 为主模式，送出它的更新事件(UEV)做为触发输出(TIM1_CR2 寄存器的 MMS=010)。
- 配置定时器 1 的周期(TIM1_ARR 寄存器)。
- 配置定时器 2 从定时器 1 获得输入触发(TIM2_SMCR 寄存器的 TS=000)
- 配置定时器 2 为触发模式(TIM2_SMCR 寄存器的 SMS=110)
- 置 TIM1_CR1 寄存器的 CEN=1 以启动定时器 1。
-

图 45 使用定时器 1 的更新触发定时器 2



在上一个例子中，可以在启动计数之前初始化两个计数器。下图显示在与 0 相同配置情况下， 使用触发模式而不是门控模式(TIM2_SMCR 寄存器的 SMS=110)的动作。

图 46 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。参考图主/从定时器的例子的连接，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

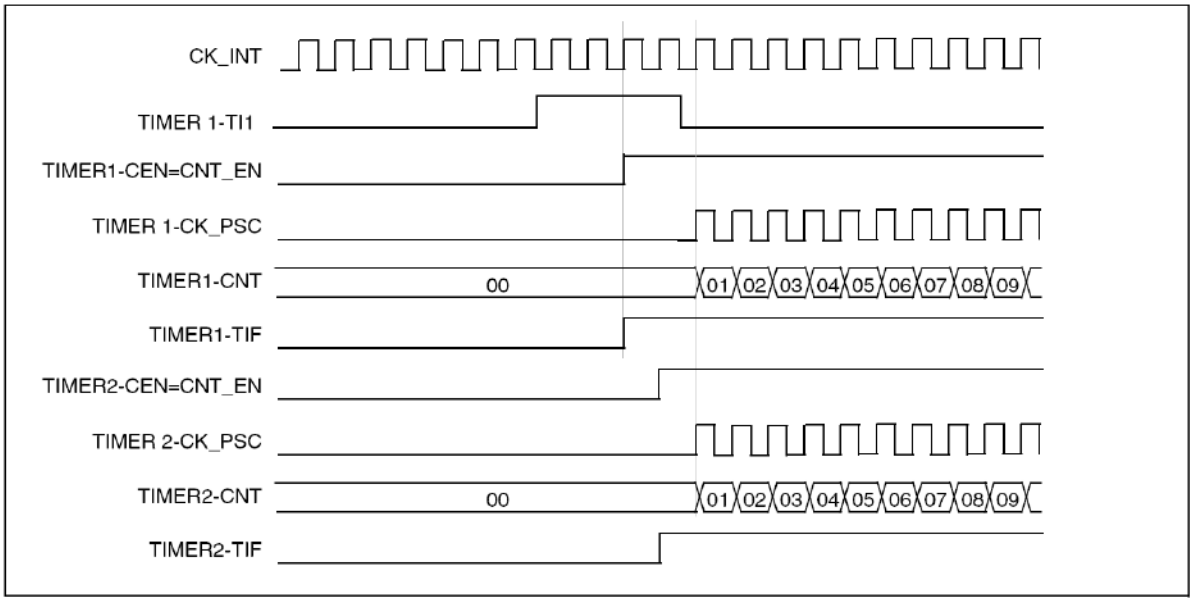
这个例子中当定时器 1 的 TI1 输入上升时使能定时器 1，使能定时器 1 的同时使能定时器 2，参见图主/从定时器的例子。为保证计数器的对齐，定时器 1 必须配置为主/从模式(对应 TI1 为从，对应定时器 2 为主)：

- 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 47 使用定时器 1 的 TI1 输入触发定时器 1 和定时器 2



19.3.16 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。详见随后章节： 支持定时器、看门狗、 bxCAN 和 I2C 的调试。

19.4 TIM2/3 寄存器

19.4.1 TIM2/3 控制寄存器 1 (TIMx_CR1)

偏移地址: 0x00

复位值: 0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CC4_TO_ADC	CC3_TO_ADC	Res.	Res.	Res.	Res.	CKD[1:0]		ARPE	CMS[1:0]		DIR	OPM	URS	UDIS	CEN
rw	rw					rw	rw	rw	rw	rw	rw	rw	rw		
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位 15	**CC4_TO_ADC**: CC4 触发信号选择 0：CC4_TO_ADC 输出信号来自 CH4 1：CC4_TO_ADC 输出信号来自 OC4REF
位 14	**CC3_TO_ADC**: CC3 触发信号选择 0：CC3_TO_ADC 输出信号来自 CH3 1：CC3_TO_ADC 输出信号来自 OC3REF
位 15：10	保留:必须保持复位值
位 9：8	**CKD[1:0]**: 时钟分频因子（Clock division） 定义在定时器时钟 (CK_INT)频率、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：tDTS = tCK_INT 01：tDTS = 2 x tCK_INT 10：tDTS = 4 x tCK_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（Auto-reload preload enable） 0: TIMx_ARR 寄存器没有缓冲 1: TIMx_ARR 寄存器有缓冲
位 6：5	**CMS[1:0]**:选择中央对齐模式 (Center-aligned mode selection) 00：边沿对齐模式。计数器依据方向位 (DIR) 向上或向下计数。 01：中央对齐模式 1。计数器交替地向上和向下计数。配置为输出的通道 (TIMx_CCMRx 寄存器中 CCxS=00) 的输出比较中断标志位，只在计数器向下计数时被设置。 10：中央对齐模式 2。计数器交替地向上和向下计数。配置为输出的通道 (TIMx_CCMRx 寄存器中 CCxS=00) 的输出比较中断标志位，只在计数器向上计数时被设置。 11：中央对齐模式 3。计数器交替地向上和向下计数。配置为输出的通道 (TIMx_CCMRx 寄存器中 CCxS=00) 的输出比较中断标志位，在计数器向上和向下计数时均被设置。
位 4	**DIR**: 方向 (Direction) 0：计数器向上计数 1：计数器向下计数
位 3	**OPM**: 单脉冲模式 (One pulse mode) 0：在发生更新事件时，计数器不停止； 1：在发生下一次更新事件 (清除 CEN 位)时，计数器停止。
位 2	**URS**: 更新请求源(Update request source),软件通过该位选择 UEV 事件的源 0：如果使能了更新中断或 DMA 请求，则下述任一事件产生更新中断或 DMA 请求： — 计数器溢出/下溢 — 设置 UG 位 — 从模式控制器产生的更新 1：如果使能了更新中断或 DMA 请求，则只有计数器溢出/下溢才产生更新中断或 DMA 请求。
位 1	**UDIS**:禁止更新(Update disable),软件通过该位允许/禁止 UEV 事件的产生 0：允许 UEV。更新(UEV)事件由下述任一事件产生： — 计数器溢出/下溢 — 设置 UG 位 — 从模式控制器产生的更新,具有缓存的寄存器被装入它们的预装载值 1：禁止 UEV。不产生更新事件，影子寄存器 (ARR、PSC、CCRx)保持它们的值。如果设置了 UG 位或从模式控制器发出了一个硬件复位，则计数器和预分频器被重新初始化。
位 0	**CEN:** 使能计数器(Counter enable) 0：禁止计数器； 1：使能计数器。

19.4.2 TIM2/3 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
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res	TI1S	MMS[2:0]			CCDS	Res.	Res	Res
								rw	rw	rw	rw	rw			
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位 15:8	保留:必须保持复位值
位 7	TI1S: TI1 选择 (TI1 selection) 0：TIMx_CH1 引脚连到 TI1 输入 1：TIMx_CH1、TIMx_CH2 和 TIMx_CH3 引脚经异或后连到 TI1 输入

位 6：4	MMS[1:0]: 主模式选择 (Master mode selection) 这 3 位用于选择在主模式下送到从定时器的同步信息 (TRGO)。可能的组合如下： 000：复位 – TIMx_EGR 寄存器的 UG 位被用于作为触发输出 (TRGO)。如果是触发输入产生的复位 (从模式控制器处于复位模式)，则 TRGO 上的信号相对实际的复位会有一个延迟。 001：使能 - 计数器使能信号 CNT_EN 被用于作为触发输出 (TRGO)。可用于同时启动多个定时器或控制在一段时间内使能从定时器。在门控模式下，计数器使能信号是 CEN 控制位和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。 010：更新 - 更新事件被选为触发输入 (TRGO)。例如，一个主定时器的时钟可以被用作一个从定时器的预分频器。 011：比较脉冲 - 在发生一次捕获或一次比较成功时，当要设置 CC1IF 标志时 (即使它已经为高)，触发输出送出一个正脉冲 (TRGO)。 100：比较 – OC1REF 信号被用于作为触发输出 (TRGO) 101：比较 - OC2REF 信号被用于作为触发输出(TRGO) 110：比较 - OC3REF 信号被用于作为触发输出(TRGO) 111：比较 - OC4REF 信号被用于作为触发输出(TRGO)
位 3	CCDS: 捕获 /比较的 DMA 选择 (捕捉 /比较 DMA selection) 0：当发生 CCx 事件时，送出 CCx 的 DMA 请求； 1：当发生更新事件时，送出 CCx 的 DMA 请求。
位 2:0	保留，必须始终为复位值

19.4.3 TIM2/3 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]			Res.	SMS[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位 15	ETP:外部触发极性(External trigger polarity) 该位选择是用 ETR 还是 ETR 的反相来作为触发操作 0：ETR 不反相，高电平或上升沿有效 1：ETR 被反相，低电平或下降沿有效
位 14	ECE: 外部时钟使能位(External clock enable)该位启用外部时钟模式 2 0：禁止外部时钟模式 2； 1：使能外部时钟模式 2。计数器由 ETRF 信号上的任意有效边沿驱动。
位 13:12	ETPS[1:0]: 外部触发预分频 (External trigger prescaler) 外部触发信号 ETRP 的频率最多是 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 (External trigger ?lter) 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。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 fSAMPLING=fDTS 采样 1000：采样频率 fSAMPLING=fDTS/8，N=6 0001：采样频率 fSAMPLING=fCK_INT，N=2 1001：采样频率 fSAMPLING=fDTS/8，N=8 0010：采样频率 fSAMPLING=fCK_INT，N=4 1010：采样频率 fSAMPLING=fDTS/16，N=5 0011：采样频率 fSAMPLING=fCK_INT，N=8 1011：采样频率 fSAMPLING=fDTS/16，N=6 0100：采样频率 fSAMPLING=fDTS/2，N=6 1100：采样频率 fSAMPLING=fDTS/16，N=8 0101：采样频率 fSAMPLING=fDTS/2，N=8 1101：采样频率 fSAMPLING=fDTS/32，N=5 0110：采样频率 fSAMPLING=fDTS/4，N=6 1110：采样频率 fSAMPLING=fDTS/32，N=6 0111：采样频率 fSAMPLING=fDTS/4，N=8 1111：采样频率 fSAMPLING=fDTS/32，N=8
位 7	MSM: 主/从模式(Master/slave mode) 0：无作用 1：触发输入(TRGI)上的事件被延迟了，以允许在当前定时器与它的从定时器间的完美同步(通过 TRGO)。这对要求把几个定时器同步到一个单一的外部事件时是非 常有用的。
位 6:4	TS[2:0]: 触发选择 (Trigger selection) 这 3 位选择用于同步计数器的触发输入。 000：内部触发 0(ITR0) 100：TI1 的边沿检测器 (TI1F_ED) 001：内部触发 1(ITR1) 101：滤波后的定时器输入 1(TI1FP1) 010：内部触发 2(ITR2) 110：滤波后的定时器输入 2(TI2FP2) 011：内部触发 3(ITR3) 111：外部触发输入 (ETRF)
位 3	保留，必须始终为复位值
位 2:0	SMS[2:0]: 从模式选择 (Slave mode selection) 当选择了外部信号，触发信号 (TRGI) 的有效边沿与选中的外部输入极性相关 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 – 根据另一个信号的输入电平，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 次更新寄存器。 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。

19.4.4 TIM2/3 DMA/中断允许寄存器 (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
Res.	TDE	Res.	CC4DE	CC3DE	CC2DE	CC1DE	UDE	Res.	TIE	Res	CC4IE	CC3IE	CC2IE	CC1IE	UIE
	rw	rw	rw	rw	rw	rw	rw		rw		rw	rw	rw	rw	rw
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位 15	保留，必须始终为复位值 .
位 14	TDE: 允许触发 DMA 请求 (Trigger DMA request enable) 0: 触发 DMA 请求禁止 1: 触发 DMA 请求允许
位 13	保留，必须始终为复位值 .
位 12	CC4DE: 捕捉 / 比较 4 DMA 请求使能 0: CC4 DMA 请求禁止 1: CC4 DMA 请求允许

位 11	CC3DE: 捕捉 / 比较 3 DMA 请求使能 0: CC3 DMA 请求禁止 1: CC3 DMA 请求允许
位 10	CC2DE: 捕捉 / 比较 2 DMA 请求使能 0: CC2 DMA 请求禁止 1: CC2 DMA 请求允许
位 9	CC1DE: 捕捉 / 比较 1 DMA 请求使能 0: CC1 DMA 请求禁止 1: CC1 DMA 请求允许
位 8	UDE: 更新 DMA 请求使能 0: 更新 DMA 请求禁止 1: 更新 DMA 请求允许
位 7	保留，必须始终为复位值 .
位 6	TIE: 触发中断使能 0: 触发中断禁止 1: 触发中断允许
位 5	保留，必须始终为复位值 .
位 4	CC4IE: 捕捉 / 比较 4 中断使能 0: CC4 中断禁止 1: CC4 中断允许
位 3	CC3IE: 捕捉 / 比较 3 中断使能 0: CC3 中断禁止 1: CC3 中断允许
位 2	CC2IE: 捕捉 / 比较 2 中断使能 0: CC2 中断禁止 1: CC2 中断允许
位 1	CC1IE: 捕捉 / 比较 1 中断使能 0: CC1 中断禁止 1: CC1 中断允许
位 0	UIE: 更新中断使能 0: 更新中断禁止 1: 更新中断允许

19.4.5 TIM2/3 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
Res.	Res.	Res.	CC4OF	CC3OF	CC2OF	CC1OF	Res.	Res.	TIF	Res.	CC4IF	CC3IF	CC2IF	CC1IF	UIF
			rc w0	rc w0	rc w0	rc w0			rc w0		rc w0	rc w0	rc w0	rc w0	rc w0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位 15:13	保留，必须始终为复位值 .
位 12	CC4OF: 捕捉 / 比较 4 重复捕捉标志
位 11	CC3OF: 捕捉 / 比较 3 重复捕捉标志
位 10	CC2OF: 捕捉 / 比较 2 重复捕捉标志
位 9	CC1OF: 捕捉 / 比较 1 重复捕捉标志 (Capture/Compare 1 overcapture flag) 仅当相应的通道被配置为输入捕获时，该标记可由硬件置 1。软件写 0 可清除该位。 0：无重复捕获产生 1：当 CC1IF 的状态已经为’1’，计数器的值被捕获到 TIMx_CCR1 寄存器。
位 8:7	保留，必须始终为复位值 .

位 6	TIF: 触发器中断标志 (Trigger interrupt ?ag) 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	保留，必须始终为复位值 .
位 4	CC4IF: 捕捉 /比较 4 中断标志
位 3	CC3IF: 捕捉/比较 3 中断标志
位 2	CC2IF: 捕捉/比较 2 中断标志
位 1	CC1IF: 捕捉 /比较 1 中断标志 (Capture/Compare 1 interrupt flag) **如果通道 CC1 配置为输出模式：** 当计数器值与比较值匹配时该位由硬件置 1，但在中心对称模式下除外。它由软件清’0’。 0：无匹配发生 1：TIMx_CNT 的 值 与 TIMx_CCR1 的 值 匹 配。 当 TIMx_CCR1 的 内 容 大 于 TIMx_APR 的内容时，在向上或向上 /下计数模式时计数器溢出，或向下计数模式时的计数器下溢条件下，CC1IF 位变高。 **如果通道 CC1 配置为输入模式：** 当捕获事件发生时该位由硬件置’1’，它由软件清’0’或通过读 TIMx_CCR1 清’0’。 0：无输入捕获产生； 1：计数器值被捕获至 TIMx_CCR1(在 IC1 上检测到与所选极性相同的边沿)。
位 0	UIF:更新中断标志(Update interrupt flag) 当产生更新事件时该位由硬件置’1’。它由软件清’0’。 0：无更新事件产生； 1：更新中断等待响应。当寄存器被更新时该位由硬件置’1’： — 若 TIMx_CR1 寄存器的 UDIS=0，当重复计数器数值上溢或下溢时(重复计数器=0 时产生更新事件)。 — 若 TIMx_CR1 寄存器的 URS=0、UDIS=0，当设置 TIMx_EGR 寄存器 的 UG=1 时产生更新事件，通过软件对计数器 CNT 重新初始化时。 — 若 TIMx_CR1 寄存器的 URS=0、UDIS=0，当计数器 CNT 被触发事件重新初始化时。

19.4.6 TIM2/3 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
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res	TG	Res	CC4G	CC3G	CC2G	CC1G	UG
								w	w	w	w	w	w	w	w
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位 15:7	保留，必须始终为复位值 .
位 6	TG: 触发产生（Trigger generation） 该位由软件置’1’，用于产生一个事件，由硬件自动清’0’。 0：无动作 1：TIMx_SR 中 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，必须始终为复位值 .
位 4	CC4G:捕捉/比较 4 发生
位 3	CC3G:捕捉/比较 3 发生
位 2	CC2G:捕捉/比较 2 发生

位 1	CC1G:捕捉/比较 1 发生(Capture/Compare 1 generation) 该位由软件置’1’，用于产生一个捕获 / 比较事件，由硬件自动清’0’。 0：无动作 1：在通道 1 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 (Update generation) 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 (寄存器) 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。

19.4.7 TIM2/3 捕捉/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
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

输出比较模式:

位 15	OC2CE: 输出比较 2 清 0 允许
位 14:12	OC2M[2:0]: 输出比较模式 2
位 11	OC2PE: 输出比较 2 预装允许
位 10	OC2FE: 输出比较 2 快速允许
位 9:8	CC2S[1:0]: 捕捉/比较 2 选择，该位定义通道的方向(输入/输出)，及输入信号的选择 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 允许 0: OC1REF 不受 ETRF 输入的影响； 1: 一旦检测到 ETRF 输入高电平，清除 OC1REF=0。

位 6：4	OC1M: 输出比较模式 1(Output Compare 1 mode)，该 3 位定义了输出参考信号 OC1REF 的动作，而 OC1REF 决定了 OC1、OC1N 的值。OC1REF 是高电平有效，而 OC1、OC1N 的有效电平取决于 CC1P、CC1NP 位。 000：冻结。输出比较寄存器 TIMx_CCR1 与计数器 TIMx_CNT 间的比较对 OC1REF 不起作用 001：匹配时设置通道 1 为有效电平。当计数器 TIMx_CNT 的值与捕获/比较寄存器 1(TIMx_CCR1)相同时，强制 OC1REF 为高。 010 ：匹配时设置通道 1 为无效电平。当计数器 TIMx_CNT 的值与捕获/比较寄存器 1(TIMx_CCR1)相同时，强制 OC1REF 为低。 011：翻转。当 TIMx_CCR1=TIMx_CNT 时，翻转 OC1REF 的电平。 100：强制为无效电平。强制 OC1REF 为低。 101：强制为有效电平。强制 OC1REF 为高。 110：PWM 模式 1－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，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。
位 3	OC1PE: 输出比较 1 预装允许(Output Compare 1 preload enable) 0：禁止 TIMx_CCR1 寄存器的预装载功能，可随时写入 TIMx_CCR1 寄存器，并且新写入的数值立即起作用。 1：开启 TIMx_CCR1 寄存器的预装载功能，读写操作仅对预装载寄存器操作，TIMx_CCR1 的预装载值在更新事件到来时被加载至当前寄存器中。
位 2	OC1FE: 输出比较 1 快速使能(Output Compare 1 fast enable) 该位用于加快 CC 输出对触发输入事件的响应。 0：CC1 的正常操作依赖于计数器与 CCR1 的值，即使工作于触发器状态。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: 捕捉/比较 1 选择 (Capture/Compare 1 selection) 这 2 位定义通道的方向(输入/输出)，及输入脚的选择： 00：CC1 通道被配置为输出 01：CC1 通道被配置为输入，IC1 映射在 TI1 上 10：CC1 通道被配置为输入，IC1 映射在 TI2 上 11：CC1 通道被配置为输入，IC1 映射在 TRC 上。此模式仅工作在内部触发器输入被选中时(由 TIMx_SMCR 寄存器的 TS 位选择)。

输入捕捉模式

位 15：12	IC2F[3:0]: 输入捕捉 2 滤波器
位 11：10	IC2PSC[1:0]: 输入捕捉 2 预分频器
位 9：8	CC2S[1:0]: 捕捉/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)。
位 7：4	IC1F[3:0]: 输入捕捉 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 fDTS 采样 1000：采样频率 fSAMPLING=fDTS/8，N=6 0001：采样频率 fSAMPLING=fCK_INT，N=2 1001：采样频率 fSAMPLING=fDTS/8，N=8 0010：采样频率 fSAMPLING=fCK_INT，N=4 1010：采样频率 fSAMPLING=fDTS/16，N=5 0011：采样频率 fSAMPLING=fCK_INT，N=8 1011：采样频率 fSAMPLING=fDTS/16，N=6 0100：采样频率 fSAMPLING=fDTS/2，N=6 1100：采样频率 fSAMPLING=fDTS/16，N=8 0101：采样频率 fSAMPLING=fDTS/2，N=8 1101：采样频率 fSAMPLING=fDTS/32，N=5 0110：采样频率 fSAMPLING=fDTS/4，N=6 1110：采样频率 fSAMPLING=fDTS/32，N=6 0111：采样频率 fSAMPLING=fDTS/4，N=8 1111：采样频率 fSAMPLING=fDTS/32，N=8

位 3:2	IC1PSC[1:0]: 输入捕捉 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]: 捕捉/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)。

19.4.8 TIM2/3 捕捉/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
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

输出比较模式:

位 15	OC4CE: 输出比较 4 清 0 允许
位 14:12	OC4M[2:0]: 输出比较模式 4
位 11	OC4PE: 输出比较 4 预装允许
位 10	OC4FE: 输出比较 4 快速允许
位 9:8	CC4S[1:0]: 捕捉/比较 4 选择，该位定义通道的方向(输入/输出)，及输入信号的选择 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 允许 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	OC3M: 输出比较 3 模式(Output Compare 3 mode)
位 3	OC3PE: 输出比较 3 预装允许(Output Compare 3 preload enable)
位 2	OC3FE: 输出比较 3 快速使能(Output Compare 3 fast enable)
位 1：0	CC3S[1:0]: 捕捉/比较 3 选择 (Capture/Compare 3 selection) 这 2 位定义通道的方向(输入/输出)，及输入脚的选择： 00：CC3 通道被配置为输出 01：CC3 通道被配置为输入，IC3 映射在 TI3 上 10：CC3 通道被配置为输入，IC3 映射在 TI4 上 11：CC3 通道被配置为输入，IC3 映射在 TRC 上。此模式仅工作在内部触发器输入被选中时(由 TIMx_SMCR 寄存器的 TS 位选择)。

输入捕捉模式

位 15：12	IC4F: 输入捕捉 4 滤波器

位 11：10	IC4PSC[1:0]: 输入捕捉 4 预分频器
位 9：8	CC4S: 捕捉/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)。
位 7：4	IC3F[3:0]: 输入捕捉 3 滤波器，这几位定义了 TI3 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 fDTS 采样 1000：采样频率 fSAMPLING=fDTS/8，N=6 0001：采样频率 fSAMPLING=fCK_INT，N=2 1001：采样频率 fSAMPLING=fDTS/8，N=8 0010：采样频率 fSAMPLING=fCK_INT，N=4 1010：采样频率 fSAMPLING=fDTS/16，N=5 0011：采样频率 fSAMPLING=fCK_INT，N=8 1011：采样频率 fSAMPLING=fDTS/16，N=6 0100：采样频率 fSAMPLING=fDTS/2，N=6 1100：采样频率 fSAMPLING=fDTS/16，N=8 0101：采样频率 fSAMPLING=fDTS/2，N=8 1101：采样频率 fSAMPLING=fDTS/32，N=5 0110：采样频率 fSAMPLING=fDTS/4，N=6 1110：采样频率 fSAMPLING=fDTS/32，N=6 0111：采样频率 fSAMPLING=fDTS/4，N=8 1111：采样频率 fSAMPLING=fDTS/32，N=8
位 3:2	IC3PSC: 输入捕捉 3 预分频器，这 2 位定义了 CC3 输入(IC3) 的预分频系数。一旦 CC3E=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	CC3S: 捕捉/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)。

19.4.9 TIM2/3 捕捉/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
Res.	Res.	CC4P	CC4E	Res.	Res.	CC3P	CC3E	Res.	Res.	CC2P	CC2E	Res.	Res.	CC1P	CC1E
		rw	rw			rw	rw			rw	rw			rw	rw
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位 15:14	保留，必须始终为复位值 .
位 13	CC4P: 捕捉/比较 4 输出极性
位 12	CC4E: 捕捉/比较 4 输出使能
位 11:10	保留，必须始终为复位值
位 9	CC3P: 捕捉/比较 3 输出极性
位 8	CC3E: 捕捉/比较 3 输出使能
位 7:6	保留，必须始终为复位值
位 5	CC2P: 捕捉/比较 2 输出极性
位 4	CC2E: 捕捉/比较 2 输出使能
位 3:2	保留，必须始终为复位值

位 1	CC1P: 捕捉 /比较 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：** CC1NP/CC1P 位选择在触发或捕捉模式下 TI1FP1 和 TI2FP1 的有效极性。 00: 非反相 /上升沿 电路作用于 TIxFP1 的 上升沿 (在复位、外部时钟或触发模式下的捕捉或触发操作),TIxFP1 非反相 01: 反想 /下降沿 电路作用于 TIxFP1 的 下降沿 (在复位、外部时钟或触发模式下的捕捉或触发操作),TIxFP1 反相 00: 保留不用 11: 非反相 /上升或下降沿 电路作用于 TIxFP1 的上升沿 和下降沿 (在复位、外部时钟或触发模式下的捕捉或触发操作), TIxFP1 非反相 (在门控模式)。在编码模式下不能使用此配置。
位 0	CC1E: 捕捉 /比较 1 输出使能 **CC1 通道配置为输出：** 0：关闭一 OC1N 禁止输出，因此 OC1N 的电平依赖于 MOE、OSSI、OSSR、OIS1、OIS1N 和 CC1E 位的值。 1：开启一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。 **CC1 通道配置为输入：** 本位用于决定是否一个定时器值的捕捉要装载到捕捉 / 比较寄存器 1(TIMx_CCR1)。 0: 捕捉禁止 1: 捕捉允许

19.4.10 TIM2/3 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
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位 15:0	CNT[15:0]: 计数器值

19.4.11 TIM2/3 预分频 (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
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位 15:0	PSC[15:0]: 预分频值 预分频器的值 (Prescaler value)计数器的时钟频率 (CK_CNT)等于 fCK_PSC/(PSC[15:0]+1)。 每次当更新事件产生时，PSC 的值被装入当前预分频器寄存器；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’。

19.4.12 TIM2/3 自动重装寄存器 (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
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位 15:0	ARR[15:0]: 自动重装载的值 (Prescaler value) ARR 包含了将要装载入实际的自动重装载寄存器的值。

19.4.13 TIM2/3 捕捉/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
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位 15:0	CCR1[15:0]: 捕捉 /比较通道 1 的值 **若 CC1 通道配置为输出：** CCR1 决定了装入当前捕获 /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。

19.4.14 TIM2/3 捕捉/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
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位 15:0	CCR2[15:0]: 捕捉 /比较通道 2 的值 **若 CC2 通道配置为输出：** CCR2 决定了装入当前捕获 /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 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。

19.4.15 TIM2/3 捕捉/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
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位 15:0	CCR3[15:0]: 捕捉 /比较通道 3 的值 **若 CC3 通道配置为输出：** CCR3 决定了装入当前捕获 /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 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。

19.4.16 TIM2/3 捕捉/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
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位 15:0	CCR4[15:0]: 捕捉 /比较通道 4 的值 **若 CC4 通道配置为输出：** CCR4 决定了装入当前捕获 /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 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。

19.4.17 TIM2/3 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
Res.			DBL[4:0]					Res.			DBA[4:0]				
			rw	rw	rw	rw	rw				rw	rw	rw	rw	rw
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位 15:13	保留，必须始终为复位值 .

位 12:8	DBL[4:0]: DMA 连续传送长度 (DMA burst length) 这 5 位定义了 DMA 在连续模式下的传送长度 (当对 TIMx_DMAR 寄存器进行读或写时，定时器则进行一次连续传送) 00000：1 次传输 00001：2 次传输 00010：3 次传输 .... 10001：18 次传输
位 7:5	保留，必须始终为复位值 .
位 4:0	DBA[4:0]: DMA 基地址 (DMA base address) 00000:TIM1_CR1 00001:TIM1_CR2 00010:TIM1_SMCR ……

19.4.18 TIM2/3 DMA 完全传送地址寄存器(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
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位 15:0	DMAB[15:0]: DMA 并发（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访问： （TIMx_CR1 地址） + （DBA + DMA 索引）x 4，其中： “TIMx_CR1 地址”是控制寄存器 1(TIMx_CR1)所在的地址； “DBA”是 TIMx_DCR 寄存器中定义的基地址； “DMA 索引”是由 DMA 自动控制的偏移量，它取决于 TIMx_DCR 寄存器中定义的 DBL。

19.4.19 TIM2/3 比较输出寄存器(TIMx_OR)

偏移地址: 0x50

复位值: 0x0000

TIM2:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res	Res.	OCREF_CLR_RMP		TI4_RMP		ETR_RMP		
									rw	rw	rw	rw	rw	rw	rw
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位 15:6	保留，必须始终为复位值
位 6:5	OCREF_CLR_RMP: 00: OCREF_CLR 连接 COMP1_OUT 01: OCREF_CLR 连接 COMP2_OUT 其他: OCREF_CLR 连接 COMP1_OUT|COMP2_OUT
位 4:3	TI4_RMP: 内部触发重定向 01: TIM TI4 连接 COMP2_OUT 10: TIM TI4 连接 COMP1_OUT 其他: TIM TI4 连接 PA3 或 PB11

位 2:0	ETR_RMP: TIMx ETR 重定向 011: TIM2 ETR 连接 HSI16 101: TIM2 ETR 连接 LSE 110：TIM2 ETR 连接 COMP2_OUT 111: TIM2 ETR 连接 COMP1_OUT 其他: TIM2 ETR 连接 PA0,PA5 或 PA15

TIM3:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res	Res.	OCREF_CLR_RMP		Res.	TI_RMP	Res.	ETR_RMP	
									rw	rw		rw		rw	rw
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位 15:6	保留，必须始终为复位值
位 6:5	OCREF_CLR_RMP: 00: OCREF_CLR 连接 COMP1_OUT 01: OCREF_CLR 连接 COMP2_OUT 其他: OCREF_CLR 连接 COMP1_OUT|COMP2_OUT
位 4	保留，必须始终为复位值
位 3	TI_RMP: 内部触发重定向 0: TIM3 TI1 连接 USB_SOF 1: TIM3 TI1 连接 PA6,PC6,PD2 或 PB4
位 2	保留，必须始终为复位值
位 1:0	ETR_RMP: TIM3 ETR 重定向 10: TIM3 ETR 连接 HSI8 其他: TIM3 ETR 连接 PD2

20 低功耗定时器 (LPTIM1/2/3)

20.1 简介

LPTIM 是一个低功耗的 16 位定时器。由于可以选择时钟源，LPTIM 能够在除待机和关机模式之外的所有电源模式下保持运行。同时 LPTIM 可以在没有内部时钟源的情况下运行，并在某些应用中作为“脉冲计数器”。此外，LPTIM 具有从低功耗模式中唤醒系统的能力，这使得它适合于以极低的功耗实现“超时功能”。

LPTIM 引入了一个灵活的时钟方案，该方案能够提供了所需的功能和性能，同时还能最大程度的降低功耗。

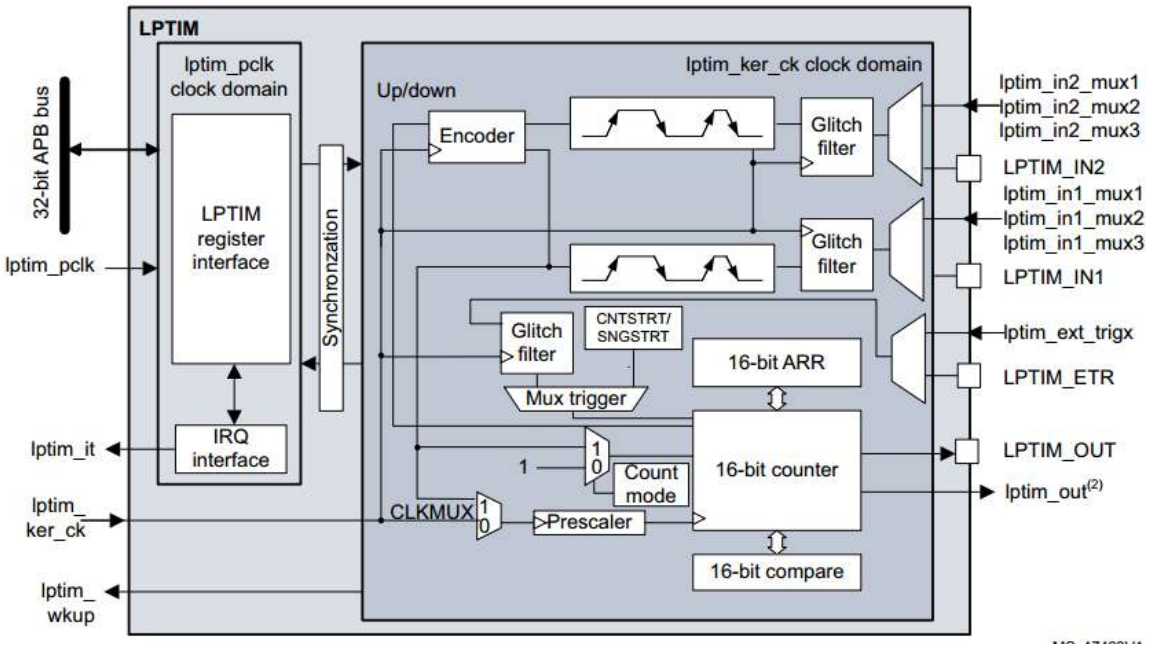
20.2 主要功能

- 16 位递增计数器
- 3bit 预分频器（1，2，4，8，16，32，64，128）
- 可选时钟源
- 16 位自动恢复寄存器
- 16 位比较寄存器
- 连续/单次模式
- 可选的软/硬件输入触发器
- 可编程数字过滤器
- 可配置输出模式：脉冲，PWM
- 可配置 I/O 极性
- 编码器模式

20.3 功能描述

20.3.1 LPTIM 框图

图 1 低功耗计时器框图



20.3.2 LPTIM 引脚和内部信号

下面的表分别提供了 LPTIM 引脚和内部信号的列表。

表 2 LPTIM 输入/输出引脚

名称	信号类型	说明
LPTIM_IN1	数字输入	LPTIM 输入 1
LPTIM_IN2	数字输入	LPTIM 输入 2
LPTIM_ETR	数字输入	LPTIM 外部触发 GPIO
LPTIM_OUT	数字输出	LPTIM 输出 GPIO

表 3 LPTIM 内部信号

名称	信号类型	说明
lptim_pclk	数字输入	LPTIM APB 时钟域
lptim_ker_ck	数字输入	LPTIM 内核时钟
lptim_in1_mux1	数字输入	内部 LPTIM 输入 1 连接到复用输入 1
lptim_in1_mux2	数字输入	内部 LPTIM 输入 1 连接到复用输入 2
lptim_in1_mux3	数字输入	内部 LPTIM 输入 1 连接到复用输入 3
lptim_in2_mux1	数字输入	内部 LPTIM 输入 2 连接到复用输入 1
lptim_in2_mux2	数字输入	内部 LPTIM 输入 2 连接到复用输入 2
lptim_in2_mux3	数字输入	内部 LPTIM 输入 2 连接到复用输入 3
lptim_ext_trigx	数字输入	LPTIM 外部触发器输入 x
lptim_out	数字输出	LPTIM 计数器输出
lptim_it	数字输出	LPTIM 中断
lptim_wakeup	数字输出	LPTIM 唤醒事件

20.3.3 LPTIM 输入和触发器映射

以下详细介绍 LPTIM 外部触发器和输入连接:

表 4 LPTIM1 外部触发连接

TRIGSEL	外部触发
lptim_ext_trig0	GPIO 引脚作为 LPTIM1_ETR 可选功能（alternate function）
lptim_ext_trig1	RTC ALARM A
lptim_ext_trig2	RTC ALARM B

lptim_ext_trig3	TAMP1 输入检测
lptim_ext_trig4	TAMP2 输入检测
lptim_ext_trig5	保留
lptim_ext_trig6	COMP1_OUT
lptim_ext_trig7	COMP2_OUT

表 5 LPTIM2 外部触发连接

TRIGSEL	外部触发
lptim_ext_trig0	GPIO 引脚作为 LPTIM2_ETR 可选功能（alternate function）
lptim_ext_trig1	RTC ALARM A
lptim_ext_trig2	RTC ALARM B
lptim_ext_trig3	TAMP1 输入检测
lptim_ext_trig4	TAMP2 输入检测
lptim_ext_trig5	保留
lptim_ext_trig6	COMP1_OUT
lptim_ext_trig7	COMP2_OUT

表 5 LPTIM3 外部触发连接

TRIGSEL	外部触发
lptim_ext_trig0	GPIO 引脚作为 LPTIM3_ETR 可选功能（alternate function）
lptim_ext_trig1	RTC ALARM A
lptim_ext_trig2	RTC ALARM B
lptim_ext_trig3	TAMP1 输入检测
lptim_ext_trig4	TAMP2 输入检测
lptim_ext_trig5	保留
lptim_ext_trig6	COMP1_OUT
lptim_ext_trig7	COMP2_OUT

表 6 LPTIM1 input1 连接

Lptim_in1_mux	LPTIM1 input 1 连接到
lptim_in1_mux0	GPIO 引脚作为 LPTIM1_IN1 可选功能（alternate function）
lptim_in1_mux1	COMP1_OUT
lptim_in1_mux2	没有连接
lptim_in1_mux3	没有连接

表 7 LPTIM1 input2 连接

Lptim_in2_mux	LPTIM1 input 2 连接到
lptim_in2_mux0	GPIO 引脚作为 LPTIM1_IN2 可选功能（alternate function）
lptim_in2_mux1	COMP2_OUT
lptim_in2_mux2	没有连接
lptim_in2_mux3	没有连接

表 8 LPTIM2 input1 连接

Lptim_in1_mux	LPTIM2 input 1 连接到
lptim_in1_mux0	GPIO 引脚作为 LPTIM2_IN1 可选功能（alternate function）
lptim_in1_mux1	COMP1_OUT
lptim_in1_mux2	COMP2_OUT
lptim_in1_mux3	COMP1_OUT

表 8 LPTIM2 input2 连接

Lptim_in2_mux	LPTIM2 input 2 连接到
lptim_in2_mux0	GPIO 引脚作为 LPTIM2_IN2 可选功能（alternate function）
lptim_in2_mux1	COMP2_OUT

lptim_in2_mux2	没有连接
lptim_in2_mux3	没有连接

表 8 LPTIM3 input1 连接

Lptim_in1_mux	LPTIM3 input 1 连接到
lptim_in1_mux0	GPIO 引脚作为 LPTIM3_IN1 可选功能（alternate function）
lptim_in1_mux1	COMP1_OUT
lptim_in1_mux2	没有连接
lptim_in1_mux3	没有连接

表 8 LPTIM3 input2 连接

Lptim_in2_mux	LPTIM3 input 2 连接到
lptim_in2_mux0	GPIO 引脚作为 LPTIM3_IN2 可选功能（alternate function）
lptim_in2_mux1	COMP3_OUT
lptim_in2_mux2	没有连接
lptim_in2_mux3	没有连接

20.3.4 LPTIM 复位和时钟

LPTIM 可以使用多个时钟源进行计时。

- APB 时钟
- LP 振荡器：LSI、LSE、HIS
- 外部时钟：LPTIM 可以使用外部输入的外部时钟信号进行计时（input1）。

当对外部时钟信号进行计时时，LPTIM 可能以以下两种配置之一运行:

- 第一个配置是对 LPTIM 外部信号计时时，内部时钟源选择 APB 或任何其他嵌入式振荡器(包括 LSE、LSI 和 HSI16)。
- 第二个配置是对 LPTIM 外部时钟信号计数，时钟源通过其外部 Input1 输入。这种配置是在进入低功耗模式后，用来实现超时功能或脉冲计数器功能。

通过配置 CKSEL 和 COUNTMODE 位可以选择 LPTIM 是使用外部时钟源或内部时钟源。

当配置为使用外部时钟源时，CKPOL 位用于选择外部时钟信号的边沿方式。当选定上下沿触发时，必须内部提供一个时钟信号，并且频率至少是外部时钟信号频率的四倍。

20.3.5 故障过滤

LPTIM 输入，无论是外部(映射到 GPIOs)或是内部(在芯片内其他嵌入式外围设备，例如嵌入式比较器)，都可以使用数字过滤器，以防止任何故障和噪声干扰引入 LPTIM 内部。这是为了防止误计数或触发。

为了保证过滤器的正常操作，需要首先配置 LPTIM 的内部时钟源，再使能数字滤波器。

数字滤波器分为两组：

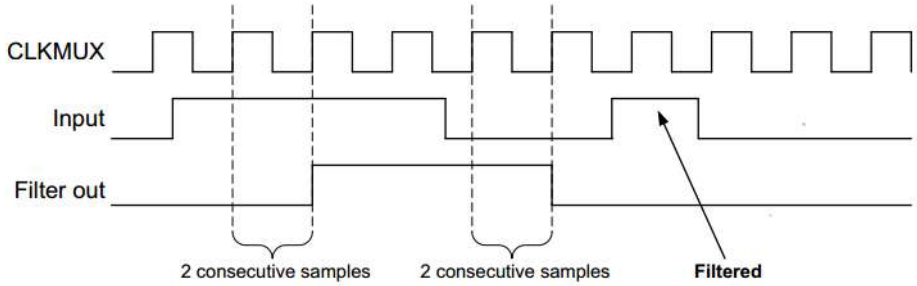
- 第一组数字滤波器保护 LPTIM 外部输入。数字滤波器的灵敏度由 CKFLT 位控制
- 第二组数字滤波器保护 LPTIM 内部输入。数字滤波器的灵敏度由 TRGFLT 位控制。

注：数字滤波器的灵敏度由组控制。不可能在同一组内分别配置每个数字滤波器灵敏度。

滤波器灵敏度作用于在 LPTIM 输入上，将输入信号电平转化位为有效信号。

下图显示了故障过滤器在 2 个连续的采样的时序。

图 故障过滤器时序图



注：如果没有提供内部时钟信号，则必须通过将 CKFLT 和 TRGFLT 位设置为“0”来关闭数字滤波器。在这种情况下，可以使用外部模拟滤波器来保护 LPTIM 的外部输入。

20.3.6 预分频

LPTIM 16 位计数器的前面有一个可配置预分频器。预分频比由 PRESC[2:0]位控制。

表 预分频比例

PRESC[2:0]	分频因子
000	/1
001	/2
010	/4
011	/8
100	/16
101	/32
110	/64
111	/128

20.3.7 多路触发

LPTIM 计数器可以通过软件启动，也可以在检测到 8 个触发器输入中的一个活动边缘之后启动。

TRIGEN[1:0]用来确定 LPTIM 触发源:

- 当 TRIGEN[1:0]为‘00’时，只要软件设置了 CNTSTRT 或 SNGSTRT 位中的一个，LPTIM 计数器就会立即启动。TRIGEN[1:0]剩下的三个可能值用于配置触发器输入使用的边沿。一旦检测到活动边沿，LPTIM 计数器就会立即启动。
- 当 TRIGEN[1:0]不为‘00’时，TRIGSEL[2:0]用于选择 8 个触发器源中的哪一个用于启动计数器。

外部触发器作为 LPTIM 的异步信号，在触发器检测沿之后，需要两个时钟周期延迟进行时钟同步，再启动。

如果一个新的触发事件在计时器已经启动时发生，它将被忽略(除非启用了超时功能)。

注：在设置 SNGSTRT/CNTSTRT 位之前，必须先使能计时器。当定时器被禁用时，这些位上的任何写操作都是无效的。

20.3.8 操作模式

LPTIM 有两种操作模式:

- 连续模式（coutinuous mode）：定时器从一个触发事件开始，并且从不停止，直到定时器被禁用。
- 单次模式(one-shot mode)：计时器从触发事件启动，当到达 ARR 值时停止。

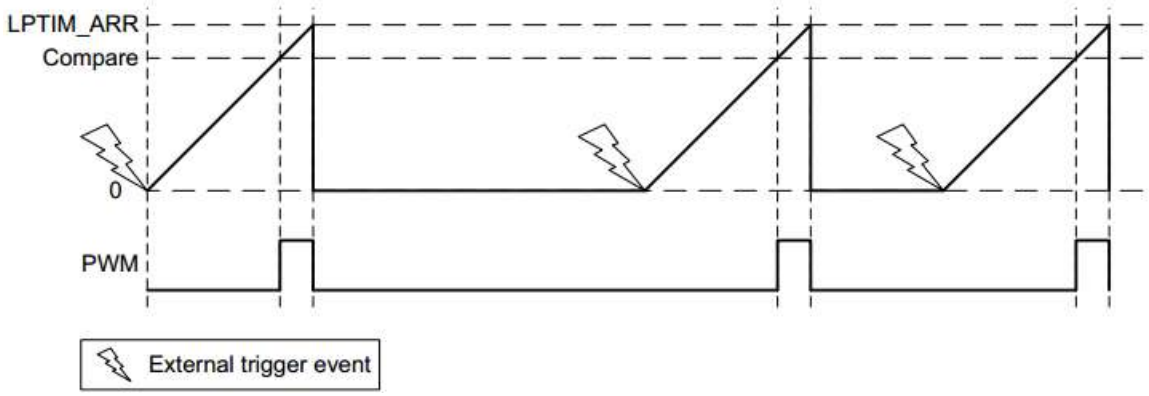
单次模式(one-shot mode)

要启用单次计数，必须设置 SNGSTRT 位。

当一个新的触发事件将重新启动计时器后。在计数值到达 ARR 之前发生的任何触发事件都将被丢弃。

如果选择了外部触发器，设置 SNGSTRT 位之后，当计数器寄存器停止之后，到来的每个外部触发器事件，都将启动计数器，进行新的单次模式计数，如下图所示。

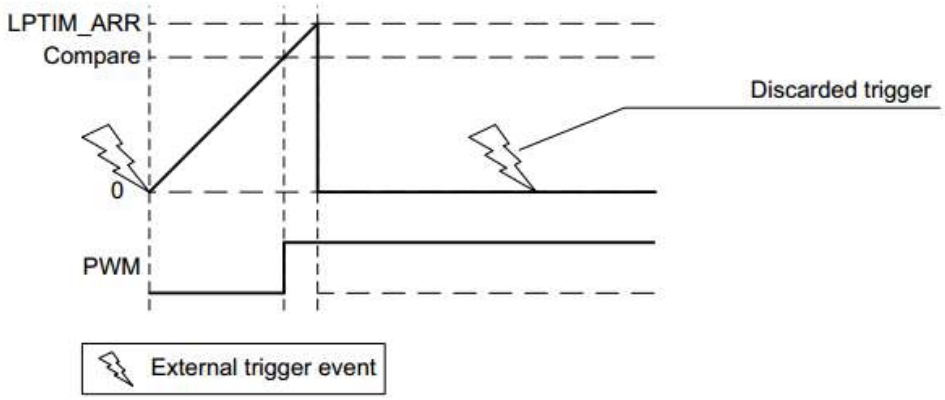
图　配置为单次模式，LPTIM 输出波形



一次模式（Set-once mode）启动：

需要注意，当设置 LPTIM_CFGR 寄存器中的 WAVE 位字段，将激活一次模式（Set-once mode）。在这种情况下，计数器只在第一个触发器之后启动一次，随后的任何触发器事件都被丢弃，如图下所示。

图 LPTIM 输出波形，单计数模式配置，一次启动(设置 WAVE 位)



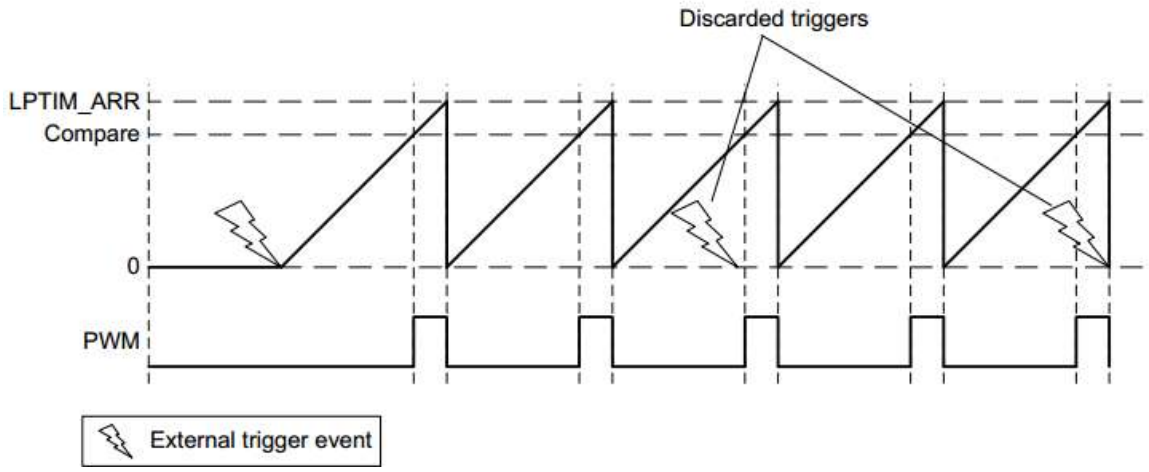
如果采用软件启动(TRIGEN[1:0] = ' 00 ')， SNGSTRT 设置后将启动一次计数。

连续模式（continous mode）

要实现连续计数，需要设置 CNTSTRT 位。

- 如果选择了外部触发，则设置 CNTSTRT 之后，外部触发事件将启动计数器进行连续计数。任何后续的外部触发事件都将被丢弃，如下图所示。
- 如果选择软件启动(TRIGEN[1:0] = ' 00 ')，设置 CNTSTRT 将启动计数器进行连续计数。

图　LPTIM 输出波形，连续模式



SNGSTRT 和 CNTSTRT 位只能在计时器使能后设置(ENABLE 位设置为' 1 ')。可以将单次模式更改为连续模式。

- 如果之前选择了连续模式，设置 SNGSTRT 将把 LPTIM 切换到单次模式。计数器(如果是活动的)一到达 ARR 就会停止。
- 如果之前选择了单次模式，设置 CNTSTRT 将把 LPTIM 切换到连续模式。计数器(如果是活动的)一到达 ARR 就会重新启动。

20.3.9 超时功能

超时功能可以实现，当检测到输入触发沿时重置 LPTIM 计数器的功能。

第一个触发事件将启动计时器，任何后续的触发事件将重置计数器，计时器将重新启动。

在实现低功耗模式下，超时值对应于比较值；如果在预期的时间范围内没有发生触发事件，比较匹配事件将唤醒 MCU。

20.3.10 波形产生

两个 16 位寄存器，LPTIM_ARR (autoreload register)和 LPTIM_CMP (compare register)，用于 LPTIM 输出几种不同的波形。

定时器可以生成以下波形:

- PWM 模式(PWM mode)：当 LPTIM_CNT 中的计数器值超过 LPTIM_CMP 中的比较值时，产生 LPTIM 输出。当 LPTIM_CNT 和 LPTIM_ARR 相等时，重置 LPTIM 输出。
- 单脉冲模式（One-pulse mode）：输出波形类似于只产生第一个脉冲的 PWM 模式，然后输出永久复位。
- 一次模式(set-once mode)：输出波形类似于单脉冲模式，除了输出保持在最后的信号电平(取决于输出配置的极性)。

上述模式要求 LPTIM_ARR 寄存器值严格大于 LPTIM_CMP 寄存器值。

LPTIM 输出波形可通过波形位配置:

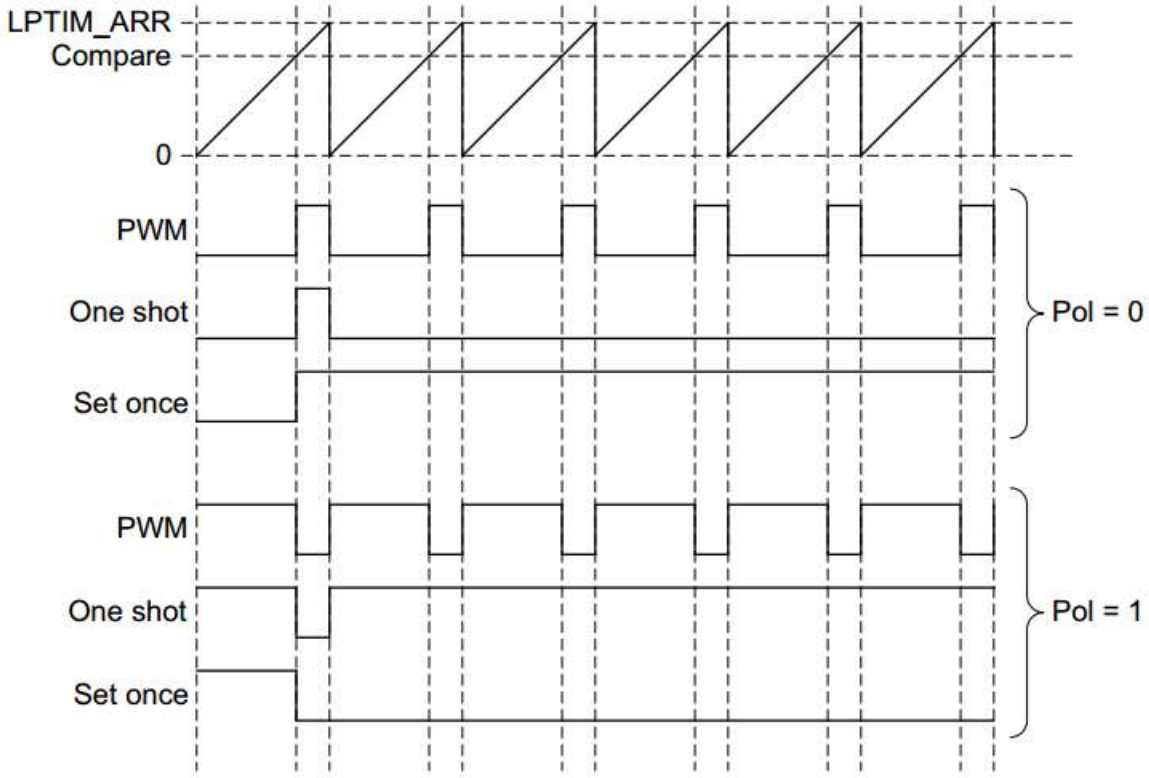
- 将 WAVE 位重新设置为“0”时，LPTIM 根据 CNTSTRT 或 SNGSTRT 的设置产生 PWM 波形或一个脉冲波形。
- 将 WAVE 位设置为“1”，LPTIM 产生一次模式的波形输出。

WAVPOL 位控制 LPTIM 输出极性。更改将立即生效，因此在重新配置极性之后，输出默认值将立即更改，甚至在计时器使能之前也是如此。

可以生成频率高达 LPTIM 时钟频率除以 2 的信号。

下图显示了 LPTIM 可以输出的三种波形。此外，它显示了 WAVPOL 位极性变化的影响。

图　波形产生



20.3.11 寄存器更新

LPTIM_ARR 寄存器和 LPTIM_CMP 寄存器在 APB 总线写操作之后立即更新，如果计时器已经启动，则在当前周期的结束后更新。

PRELOAD 位控制 LPTIM_ARR 和 LPTIM_CMP 寄存器如何更新:

- 当 PRELOAD 位设置为“0”时，LPTIM_ARR 和 LPTIM_CMP 寄存器设置后立即更新。
- 当 PRELOAD 位设置为“1”时，如果计时器已经启动，则在当前周期结束时更新 LPTIM_ARR 和 LPTIM_CMP 寄存器。

当 LPTIM APB 接口和 LPTIM 内核逻辑使用不同的时钟时，在 APB 写入后到 LPTIM 寄存器生效存在一些延迟。在此延迟期间，必须避免对这些寄存器进行任何额外的写入。

LPTIM_ISR 寄存器中的 ARROK 标志和 CMPOK 标志分别指示完成了对 LPTIM_ARR 寄存器和 LPTIM_CMP 寄存器的写操作的完成状态。

在对 LPTIM_ARR 寄存器或 LPTIM_CMP 寄存器进行写操作时，只有在第一个写操作完成后才能对同一个寄存器执行新的写操作。在 ARROK 标志或 CMPOK 标志被设置之前的任何连续写，都将导致不可预知的结果。

20.3.12 计数模式

LPTIM 计数器可以用来计算 LPTIM Input1 上的外部事件，也可以用来计算内部时钟周期。CKSEL 和 COUNTMODE 位选择用于计数器的源。

如果 LPTIM 被配置为计算 Input1 上的外部事件，计数器可以根据写入 CKPOL[1:0]位的值在上升沿、下降沿或上升/下降沿之后进行更新。

根据 CKSEL 和 COUNTMODE 值，可以选择以下计数模式:

- CKSEL=0：LPTIM 使用内部时钟源作为时钟进行计数
  - COUNTMODE = 0：LPTIM 配置为由内部时钟源进行计时，并将 LPTIM 计数器配置为在每个内部时钟脉冲之后进行更新。
  - COUNTMODE = 1：使用 LPTIM 的内部时钟对 LPTIM 外部输入 input1 进行采样。

因此，为了不遗漏任何事件，外部 Input1 信号的变化频率不应该超过提供给 LPTIM 的内部时钟的频率。此外，提供给 LPTIM 的内部时钟不能预分频 (PRESC[2:0] = 000)。

- CKSEL=1：LPTIM 使用外部时钟源作为时钟进行计数

COUNTMODE 值不起作用。

在这个配置中，LPTIM 不需要内部时钟源(除非启用了故障过滤器)。外部 Input1 上注入的信号用作 LPTIM 的系统时钟。此配置适用于未启用嵌入式振荡器的操作模式。

对于这种配置，LPTIM 计数器可以在 input1 时钟信号的上升沿或下降沿上更新，但不能同时在上升沿和下降沿上更新。

由于外部 Input1 上注入的信号也用于对 LPTIM 内核逻辑进行计时，所以在计数器开始计数之前会有一些初始延迟(在启用 LPTIM 之后)。更准确地说， Input1 上的前五个活动沿(在启用 LPTIM 之后)将丢失。

20.3.13 定时器使能

LPTIM_CR 寄存器中的 ENABLE 位用于启用/禁用 LPTIM 逻辑。在设置了 ENABLE 位之后，需要延迟两个时钟后起效。

只有在禁用 LPTIM 时，才必须修改 LPTIM_CFGR 和 LPTIM_IER 寄存器。

20.3.14 定时器计数器重置

为了将 LPTIM_CNT 寄存器的内容重置为零，可以采用以下两种重置流程:

- 同步重置机制：同步复位由 LPTIM_CR 寄存器中的 COUNTRST 位控制。
- 在将 COUNTRST 位设置为“1”之后，会复位 LPTIM 内核时钟域中逻辑。需要注意的是，在写操作之后，重置操作完成之前，一些时钟脉冲将会被忽略。因为 COUNTRST 位于 APB 时钟域和 LPTIM 计数器位于 LPTIM 内核时钟域,所有当 COUNTRST 写 1 操作后需要延迟 3 内核时钟用于同步复位信号。
- 异步重置机制：异步复位由 LPTIM_CR 寄存器中的 RSTARE 位控制。

当 RSTARE 位被设置为“1”时，对 LPTIM_CNT 寄存器的任何读操作都会将其内容置零。异步复位在没有 LPTIM 核心时钟的情况下被触发。例如，当 LPTIM Input1 用作外部时钟源时，只有在确保 LPTIM Input1 上不发生翻转时，才应用异步重置。

应该注意的是，为了可靠地读取 LPTIM_CNT 寄存器的内容，需要执行两个连续的读访问并进行比较。当两个读访问的值相等时，可以认为读访问是可靠的。不幸的是，当启用异步重置时，不可能读取两次 LPTIM_CNT 寄存器。

20.3.15 编码器模式

这种模式允许处理来自正交编码器的信号，用于检测旋转元件的角度位置。编码器模式作为一个简单的带方向选择功能的外部时钟，可以在 0 到 LPTIM_ARR 寄存器值之间连续计数(从 0 到 ARR，或者从 ARR 到 0，取决于方向)。因此，必须在启动之前配置 LPTIM_ARR 值。从两个外部输入信号 Input1 和 Input2 生成一个时钟信号，这两个信号之间的相位决定了计数方向。

编码器模式，只有在 LPTIM 采用内部时钟源时才可用。为了保证正常运行，Input1 和 Input2 输入的信号频率必须小于 LPTIM 内部时钟频率除以 4。

方向更改由 LPTIM_ISR 寄存器中的 Down 和 Up 两个标志指示。此外，通过使能 DOWNIE 位，可以在两个方向转向时都产生中断。

要使能编码器模式，ENC 位必须设置为“1”，且 LPTIM 必须配置为连续模式。

当编码器模式激活时，LPTIM 计数器会根据编码器的速度和方向自动增加，它的内容总是代表编码器的位置。计数方向由 Up 和 Down 标志表示，对应编码器转子的旋转方向。

根据使用 CKPOL[1:0]位配置的极性，可以实现不同的组合。下表总结了可能的组合，假设 Input1 和 Input2 不同时切换。

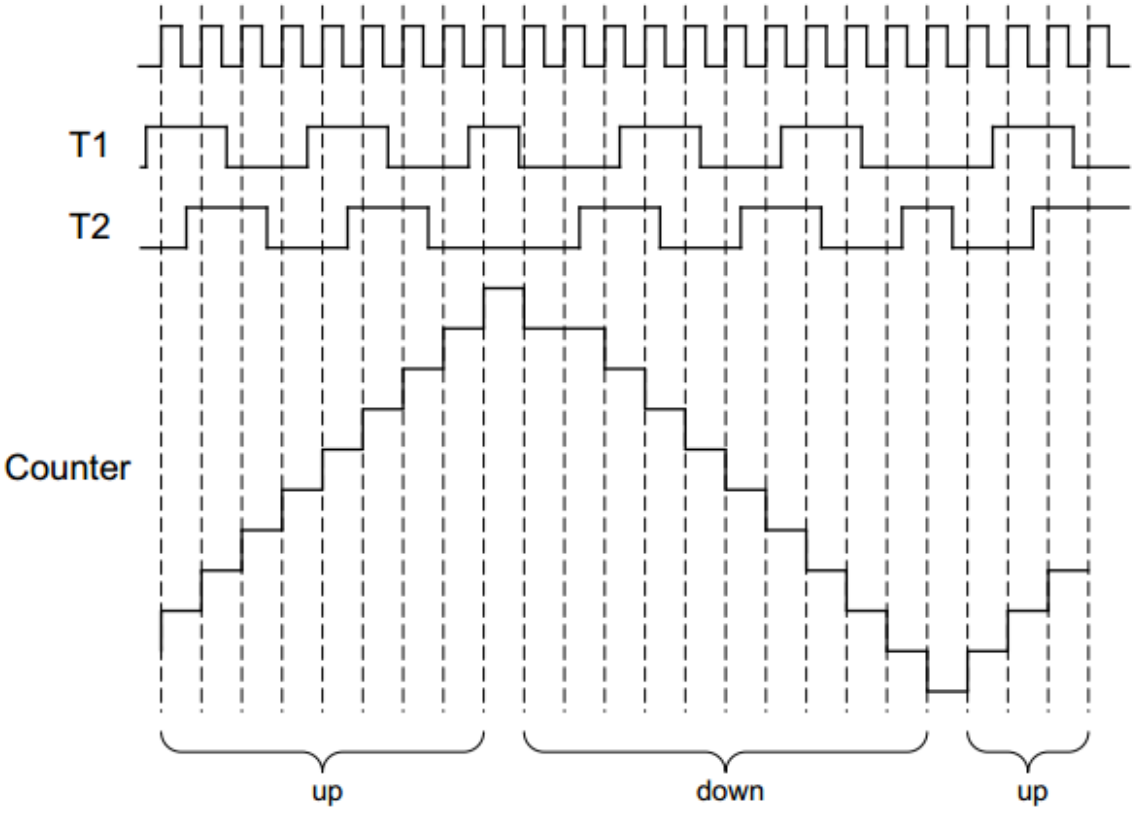
表　计数方向与编码器信号的关系

有效边沿	相对信号的电平 (TI1FP1 对应 TITI2, TI2FP2 对应 TI1)	Input1 信号		Input2 信号	
		上升	下降	上升	下降
上升沿	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
下降沿	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
上升沿和下降沿	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

下图显示了配置为上升沿和下降沿双边沿的编码器模式的计数序列。

注：在这种模式下，LPTIM 必须由内部时钟源进行计时，因此 CKSEL 位必须保持其重置值为“0”。此外，预分频值必须等于它的复位值 1 (PRESC[2:0]位必须是' 000 ')。

图　编码器模式计数序列



20.3.16 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_LPTIM_STOP 的设置， TIMx 计数器或者继续正常操作，或者停止。

20.4 LPTIM 低功耗模式

表 LPTIM 低功耗模式

模式	描述
Sleep	没有影响。LPTIM 中断导致设备退出睡眠模式。
Low-power run	没有影响
Low-power sleep	没有影响。LPTIM 中断导致设备退出低功耗睡眠模式。
Stop0/stop1	当 LPTIM 采用 LSE 或 LSI 时钟计时时没有影响。LPTIM 中断导致设备退出 Stop 0 和 Stop 1 模式。
Standby	LPTIM 被关闭，当退出待机或关机模式后必须重新初始化。
shutdown	

20.5 LPTIM 中断

通过 LPTIM_IER 寄存器配置哪些行为产生中断/唤醒事件:

- 比较匹配
- 自动重新加载匹配(在编码器模式下不考虑方向)
- 外部触发事件
- 自动重新加载寄存器写完成
- 比较寄存器写完成
- 方向改变(编码器模式)，可配置(上/下/两者)。

注：如果 LPTIM_IER 寄存器(中断使能寄存器)中的任何位被设置，并且 LPTIM_ISR 寄存器(状态寄存器)中相应的标志也被置位时，中断就不会产生。

表 中断事件

中断事件	描述
比较匹配	当计数器寄存器(LPTIM_CNT)与比较寄存器(LPTIM_CMP)相匹配时，将触发中断标志。
自动重新加载匹配	当计数器寄存器(LPTIM_CNT)与自动重新加载寄存器(LPTIM_ARR)内容相匹配时，将触发中断标志。
外部触发事件	当检测到外部触发事件时，将触发中断标志
自动重新加载寄存器写完成	当对 LPTIM_ARR 寄存器的写操作完成时，会触发中断标志。
比较寄存器写完成	当对 LPTIM_CMP 寄存器的写操作完成时，会触发中断标志。
方向改变	用于编码器模式。两个中断标志指示信号的方向改变： -UP 标志信号向上计数方向改变 -DOWN 标志信号向下计数方向改变

20.6 LPTIM1/2/3 寄存器

20.6.1 LPTIM 中断状态寄存器（LPTIM_ISR）

偏移地址: 0x00

复位值: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DOWN	UP	ARROK	CMPOK	EXTTRIG	ARRM	CMPM
									r	r	r	r	r	r	r
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位 15：7	保留:必须保持复位值
位 6	**DOWN**：计数方向由向上到向下 在编码器模式下，DOWN 标志在计数方向由向上到向下时，硬件设置。可以通过写 LPTIM_ICR 寄存器中的 DOWNCF 位为 1，来清除 DOWN 标志。
位 5	**UP**：计数方向由向下到向上 在编码器模式下，UP 标志在计数方向由向下到向上时，硬件设置。可以通过写 LPTIM_ICR 寄存器中的 UPCF 位为 1，来清除 UP 标志。
位 4	**ARROK**：自动重新加载寄存器写完成 ARROK 标志是为了通知应用程序，APB 总线对 LPTIM_ARR 寄存器的写操作已经完成，由硬件设置。可以通过写 LPTIM_ICR 寄存器中的 ARROKCF 位为 1，来清除 ARROK 标志。
位 3	**CMPOK**：比较寄存器写完成 CMPOK 标志是为了通知应用程序，APB 总线对 LPTIM_CMP 寄存器的写操作已经完成，由硬件设置。
位 2	**EXTTRIG**：外部触发事件 EXTTRIG 标志是为了通知应用程序，所选的外部触发器输入上出现了有效边沿时，由硬件设置。如果因为计时器已经启动而忽略触发，则不设置此标志。可以通过写 LPTIM_ICR 寄存器中的 EXTTRIGCF 位为 1，来清除 EXTTRIG 标志。
位 1	**ARRM**：自动匹配 当计数器寄存器(LPTIM_CNT)与自动重新加载寄存器(LPTIM_ARR)内容相匹配时，硬件设置。
位 0	**CMPM**：比较匹配 当计数器寄存器(LPTIM_CNT)与比较寄存器(LPTIM_CMP)相匹配时，硬件设置。

20.6.2 LPTIM 中断清除寄存器（LPTIM_ICR）

偏移地址：0x04

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DOWNCF	UPCF	ARROKCF	CMPOKCF	EXTTRIGCF	ARRMCF	CMPMCF
									w	w	w	w	w	w	w
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位 15：7	保留:必须保持复位值
位 6	**DOWNCF**：清除 DOWN 标志 写入 1 清除 LPTIM_ISR 寄存器中的 DOWN 标志。
位 5	**UPCF**：清除 UP 标志 写入 1 清除 LPTIM_ISR 寄存器中的 UP 标志。
位 4	**ARROKCF**：清除 ARROK 标志 写入 1 清除 LPTIM_ISR 寄存器中的 ARROK 标志。
位 3	**CMPOKCF**：清除 CMPOK 标志 写入 1 清除 LPTIM_ISR 寄存器中的 CMPOK 标志。
位 2	**EXTTRIGCF**：清除 EXTTRIG 标志 写入 1 清除 LPTIM_ISR 寄存器中的 EXTTRIG 标志。
位 1	**ARRMCF**：清除 ARRM 标志 写入 1 清除 LPTIM_ISR 寄存器中的 ARRM 标志。
位 0	**CMPMCF**：清除 CMPM 标志 写入 1 清除 LPTIM_ISR 寄存器中的 CMPM 标志。

20.6.3 LPTIM 中断使能寄存器（LPTIM_IER）

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
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DOWNIE	UPIE	ARROKIE	CMPOKIE	EXTTRIGIE	ARRMIE	CMPMIE
									rw	rw	rw	rw	rw	rw	rw
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位 15：7	保留:必须保持复位值
位 6	**DOWNIE**: 计数方向由向上到向下中断使能 0：DOWN 中断失能 1：DOWN 中断使能
位 5	**UPIE**: UP 中断使能 0：UP 中断失能 1：UP 中断使能
位 4	**ARROKIE**: ARROK 中断使能 0：ARROK 中断失能 1：ARROK 中断使能
位 3	**CMPOKIE**: CMPOK 中断使能 0：CMPOK 中断失能 1：CMPOK 中断使能
位 2	**EXTTRIGIE**: EXTTRIG 中断使能 0：EXTTRIG 中断失能 1：EXTTRIG 中断使能
位 1	**ARRMIE**: ARRM 中断使能 0：ARRM 中断失能 1：ARRM 中断使能
位 0	**CMPMIE**: CMPM 中断使能 0：CMPM 中断失能 1：CMPM 中断使能

20.6.4 LPTIM 配置寄存器（LPTIM_CFGR）

偏移地址: 0x0C

复位值: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	ENC	COUNTMODE	PRELOAD	WAVPOL	WAVE	TIMOUT	TRIGEN[1:0]		Res.
							rw	rw	rw	rw	rw	rw	rw	rw	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TRIGSEL[2:0]			Res.	PRESC[2:0]			Res.	TRGFLT[1:0]		Res.	CKFLT[1:0]		CKPOL[1:0]		CKSEL
rw	rw	rw		rw	rw	rw		rw	rw		rw	rw	rw	rw	rw
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位 31：25	保留:必须保持复位值
位 24	**ENC**: 编码器模式(encoder mode)使能 0：编码器模式失能 1：编码器模式使能

位 23	**COUNTMODE**: 计数器模式（counter mode）使能 COUNTMODE 用于选择 LPTIM 计数的时钟 0：计数采用内部时钟 1：计数器使用外部 input1 的时钟脉冲来计数
位 22	**PRELOAD**: 寄存器更新模式 控制 LPTIM_ARR 和 LPTIM_CMP 寄存器更新方式 0：寄存器在每个 APB 总线写访问之后更新 1：寄存器在当前 LPTIM 期间结束时更新
位 21	**WAVPOL**: 波形极性 控制输出极性 0：输出 LPTIM_ARR 和 LPTIM_CMP 寄存器之间的比较结果 1：输出 LPTIM_ARR 和 LPTIM_CMP 寄存器之间的比较结果取反
位 20	**WAVE**: 波形形状 0：关闭一次模式（Set-once mode），PWM 或单脉冲（one pulse）波形的选择取决于计时器的启动方式，CNTSTRT 为 PWM 模式；SNGSTRT 为单脉冲模式。 1：使能一次模式（Set-once mode）
位 19	**TIMOUT**: 使能超时 0：当计时器已经启动时到达的触发器事件将被忽略 1：当计时器已经启动时到达的触发事件将重置并重启计数器
位 18:17	**TRIGEN[1:0]**: 触发使能 TRIGEN 位控制 LPTIM 计数器是否由外部触发器启动。如果选择外部触发，触发边沿有三种可能的配置: 00：软件触发(计数启动由软件启动) 01：上升沿触发 10：下降沿触发 11：上升沿和下降沿触发
位 16	保留:必须保持复位值
位 15：13	**TRIGSEL[2:0]**: 触发选择 TRIGSEL 位选择 LPTIM 触发器事件的触发源: （参考 LPTIM 输入和触发器映射） 000：lptim_ext_trig0 001：lptim_ext_trig1 010：lptim_ext_trig2 011：lptim_ext_trig3 100：lptim_ext_trig4 101：lptim_ext_trig5 110：lptim_ext_trig6 111：lptim_ext_trig7
位 12	保留:必须保持复位值
位 11：9	**PRESC[2:0]**: 时钟分频 000：/1 001：/2 010：/4 011：/8 100：/16 101：/32 110：/64 111：/128
位 8	保留:必须保持复位值
位 7：6	**TRGFLT[1:0]**: 配置内部时钟的数字滤波器 TRGFLT 设置当内部时钟触发器上发生电平变化时，在将其视为有效的电平转换之前应检测到的连续相等样本的数量。只有内部时钟源时才能使用此功能。 00：任何触发都认为是有效的触发。 01：必须稳定至少 2 个时钟周期，才被认为是有效的触发。 10：必须稳定至少 4 个时钟周期，才被认为是有效的触发。 11：必须稳定至少 8 个时钟周期，才被认为是有效的触发。

位 5	保留:必须保持复位值
位 4：3	**CKFLT[1:0]**: 配置外部时钟的数字滤波器 CKFLT 设置当外部时钟信号发生电平变化时应检测到的连续相等的采样数，然后才将其视为有效的电平转换。只有内部时钟源时才能使用此功能。 00：任何外部时钟信号电平的变化都被认为是一个有效传输 01：外部时钟信号电平的变化必须在至少 2 个时钟周期内保持稳定，才能被认为是有效的传输。 10：外部时钟信号电平的变化必须在至少 4 个时钟周期内保持稳定，才能被认为是有效的传输。 11：外部时钟信号电平的变化必须在至少 8 个时钟周期内保持稳定，才能被认为是有效的传输。
位 2：1	**CKPOL[1:0]**: 时钟极性 如果 LPTIM 使用外部时钟计数，CKPOL 位用来配置计数器使用的边沿： 00：上升沿计数 如果在编码器模式下(设置 ENC 位)，编码器 sub-mode1 有效。 01：下降沿计数 如果在编码器模式下(设置 ENC 位)，编码器 sub-mode2 有效。 10：上升沿和下降沿计数 如果在编码器模式下(设置 ENC 位)，编码器 sub-mode3 有效。 11：无效值
位 0	**CKSEL**: 时钟源选择 0：LPTIM 采用内部时钟计数 1：LPTIM 采用外部时钟计数（input1）

注：LPTIM_CFGR 寄存器只能在禁用 LPTIM 时修改(ENABLE 为' 0 ')。

20.6.5 LPTIM 控制寄存器（LPTIM_CR）

偏移地址: 0x010

复位值: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RSTARE	COUNTRST	CNTSTRT	SNGSTRT	ENABLE
											rw	rw	rw	rw	rw
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位 15：5	保留:必须保持复位值
位 4	**RSTARE**:读复位 此位由软件设置和清除。当 RSTARE 设置为“1”时，任何对 LPTIM_CNT 寄存器的读访问都重置 LPTIM_CNT 寄存器内容。
位 3	**COUNTRST**: 计数器重置 此位由软件设置，由硬件清除。当设置为“1”时，将重置 LPTIM_CNT 寄存器。由于这个复位的同步性质，它只发生在同步延迟的 3 LPTimer 核心时钟周期(LPTimer 核心时钟可能不同于 APB 时钟)。 注：在硬件没有将 COUNTRST 设置为 0 之前，软件绝对将 COUNTRST 设置为 1。软件应该在确保 COUNTRST 为 0 时再设置为 1。
位 2	**CNTSTRT**: 计时器在连续模式下启动 此位由软件设置，由硬件清除。 在软件启动时(TRIGEN[1:0] = ' 00 ')，设置这个位将在连续模式下启动 LPTIM。 如果软件启动被禁用(TRIGEN[1:0]不为' 00 ')，设置这个位会在检测到外部触发时启动计时器。 如果在单脉冲模式下设置此位，则切换为连续模式。 这个位只能在 LPTIM 使能设置，并由硬件自动复位。

位 1	**SNGSTRT**:单模（single mode）启动 此位由软件设置，由硬件清除。 在软件启动时(TRIGEN[1:0] = ' 00 ')，设置此位将在单脉冲模式下启动 LPTIM。 如果软件启动被禁用(TRIGEN[1:0]不为' 00 ')，设置这个位会在检测到外部触发时启动计时器。 如果在连续模式时设置此位，那么 LPTIM 将在 LPTIM_ARR 和 LPTIM_CNT 寄存器下一次匹配时停止。 这个位只能在 LPTIM 使能后设置，并由硬件自动复位。
位 0	**ENABLE**:LPTIM 使能 0：LPTIM 禁用 1：LPTIM 使能

20.6.6 LPTIM 比较寄存器（LPTIM_CMP）

偏移地址: 0x014

复位值: 0x0000

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

位 15：0	**CMP[15:0]**: 比较值 注：LPTIM_CMP 寄存器只能在 LPTIM 使能后修改(ENABLE 设置为“1”)

20.6.7 LPTIM 自动装置寄存器（LPTIM_ARR）

偏移地址: 0x018

复位值: 0x0001

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
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位 15：0	**ARR[15:0]**: 自动重装载数值 注：LPTIM_ARR 寄存器只能在 LPTIM 使能后修改(ENABLE 设置为“1”)

20.6.8 LPTIM 计数器（LPTIM_CNT）

偏移地址: 0x01C

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
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位 15：0	**CNT[15:0]**: 计数器值 当 LPTIM 运行在异步时钟时，读取 LPTIM_CNT 寄存器可能会返回不可靠的值。因此，在这种情况下，需要执行两次连续的读访问，并检查两个返回值是否相同。只有在保证两次连续读访问的值相等时，才认为读访问是可靠的。

20.6.9 LPTIM 配置寄存器 2（LPTIM_CFGR2）

偏移地址：0x024

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IN2SEL[1:0]		Res.	Res.	IN1SEL[1:0]	
											rw	rw	rw	rw	rw
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位 15：6	保留:必须保持复位值
位 5:4	**IN2SEL[1:0]**: LPTIM input 2 选择 IN2SEL 位控制 LPTIM Input 2 多路复用器，它将 LPTIM Input 2 连接到一个可用的输入。 00: lptim_in2_mux0 01: lptim_in2_mux1 10: lptim_in2_mux2 11: lptim_in2_mux3
位 3：2	保留:必须保持复位值
位 1:0	**IN1SEL[1:0]**: LPTIM input1 选择 IN1SEL 位控制 LPTIM Input 1 多路复用器，它将 LPTIM Input1 连接到一个可用的输入。 00: lptim_in1_mux0 01: lptim_in1_mux1 10: lptim_in1_mux2 11: lptim_in1_mux3

21 独立看门狗 (IWDG)

21.1 简介

独立看门狗(IWDG)由专用的低速时钟(LSI)驱动，即使主时钟发生故障它也仍然有效。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。

21.2 IWDG 主要功能

- 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
- IWDG 计数器复位初始值可由 FLASH 选项字设置（请查阅 FLASH option 章节的描述），通过配置 Flash 选项字，可以保证当芯片复位后如果程序跑飞，IWDG 复位的时间间隔不会太长。
- 可以通过 Flash 选项字 LSI_LP_CTL 控制芯片进入 STOP 或者 STANDBY 模式后 LSI 的状态。通过配置 Flash 选项字，可以选择使芯片进入 STOP 或者 STANDBY mode 后，如果 LSION 设置为 0，则关掉 LSI；在芯片唤醒后，LSI 恢复成进模式之前的状态。如果不配置 Flash 选项字，则在使能 IWDG 后再进入 STOP 或者 STANDBY mode，系统一定会被 IWDG 周期唤醒。

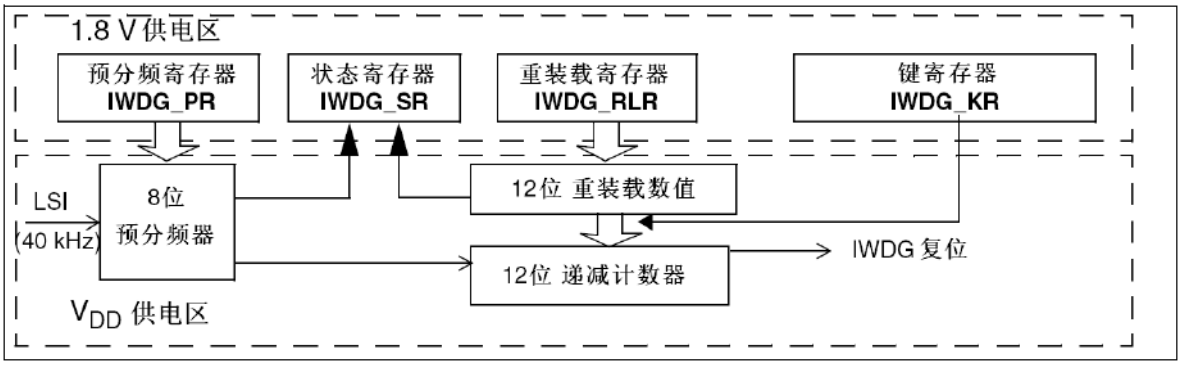
21.3 IWDG 功能描述

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，只要在键寄存器 IWDG_KR 中写入 0xAAAA， IWDG_RLR 中的值就会被重新加载到计数器，从而避免产生看门狗复位。

支持看门狗 window 窗口模式,详细的描述请参考 IWDG_WINR。

图 21-1 IWDG 框图



注意: 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 21-1 IWDG 超时时间表(40KHz 的输入时钟(LSI))

预分频系数	PR[2:0]位	最短时间(ms) RL[11:0] = 0x000	最长时间(ms) RL[11:0] = 0xFFF
/4	0	0.1	409.6
/8	1	0.2	819.2
/16	2	0.4	1638.4
/32	3	0.8	3276.8
/64	4	1.6	6553.6
/128	5	3.2	13107.2
/256	(6 或 7)	6.4	26214.4

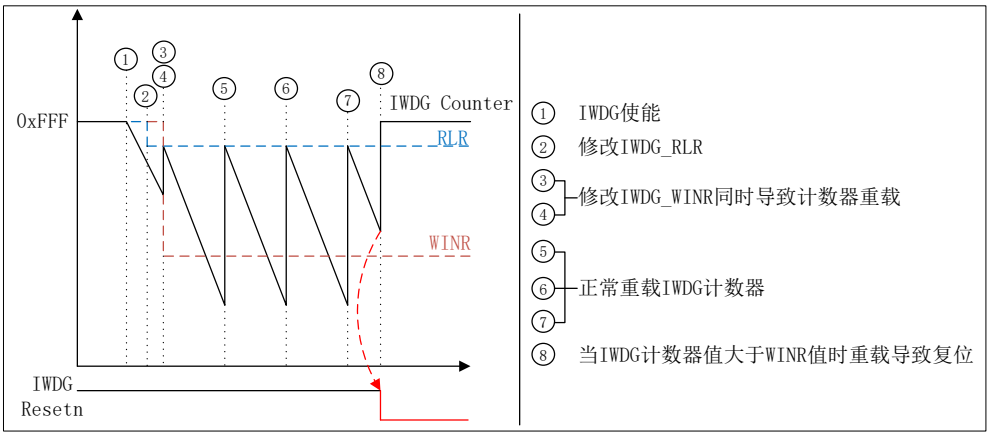
注意: 这些时间是按照 40kHz 时钟给出。实际上，MCU 内部的 RC 频率会在 30kHz 到 60kHz 之间变化。 此外，即使 RC 振荡器的频率是精确的，确切的时序仍然依赖于 APB 接口时钟与 RC 振荡器时钟 之间的相位差，因此总会有一个完整的 RC 周期是不确定的。

通过对 LSI 进行校准可获得相对精确的看门狗超时时间。

21.3.1 窗口选项

通过在 IWDG_WINR 寄存器中设置合适的窗口，IWDG 也可以用作窗口看门狗。当计数器值大于窗口寄存器(IWDG_WINR)中存储的值时，如果执行重载操作，则会产生复位。IWDG_WINR 的默认值为 0x0000FFFF，如果不更新此默认值，将会禁用窗口选项。窗口值一经更改，便会执行重载操作，以便将递减计数器的值复位为 IWDG_RLR 值，方便计算周期数以生成下一次重载。

图 21-2 IWDG 使能窗口选项时的工作说明



使能窗口选项时配置 IWDG 的流程:

1. 向 IWDG_KR 写入 0x0000CCCC 来使能 IWDG。
2. 向 IWDG_KR 写入 0x00005555 来使能寄存器访问。
3. 修改 IWDG_PR 的值为 0~7 中的值，配置 IWDG 的预分频器。
4. 写重载寄存器 IWDG_RLR。
5. 等待 IWDG_RLR 寄存器值更新完成(IWDG_SR = 0x0000 0000)。
6. 写窗口寄存器 IWDG_WINR，这个操作会导致 IWDG 计数器自动更新为 IWDG_RLR 中的值。

注意: 当 IWDG_SR 为 0x0000 0000 时，才能写 IWDT_WINR，否则 IWDG 计数器可能不会正确的重载为 IWDG_RLR 中的值。

不使能窗口选项时配置 IWDG 的流程:

1. 向 IWDG_KR 写入 0x0000CCCC 来使能 IWDG。
2. 向 IWDG_KR 写入 0x00005555 来使能寄存器访问。
3. 修改 IWDG_PR 的值为 0~7 中的值，配置 IWDG 的预分频器。
4. 写重载寄存器 IWDG_RLR。
5. 等待 IWDG_RLR 寄存器值更新完成(IWDG_SR = 0x0000 0000)。
6. 刷新计数器值为 IWDG_RLR 值(IWDG_KR = 0x0000 AAAA)。

21.3.2 硬件看门狗

如果用户在选择字节中启用了“硬件看门狗”功能，在系统上电复位后，看门狗会自动开始运行；如果在计数器计数结束前，若软件没有向键寄存器写入 0x0000AAAA 去使 IWDG 计数器重载，则系统会产生复位。

21.3.3 寄存器访问保护

IWDG_PR 和IWDG_RLR 寄存器具有写保护功能。要修改这两个寄存器的值，必须先向 IWDG_KR 寄存器中写入 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。

21.3.4 调试模式

当微控制器进入调试模式时(Cortex-M0 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。

21.4 IWDG 寄存器

21.4.1 关键字寄存器(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
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位 31:16	**保留:**必须保持复位值
位 15:0	**KEY[15:0]:** 键值。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 LSI 时钟章节) 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)。

21.4.2 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
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

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

位 31:3	**保留:**必须保持复位值
位 2:0	**PR[2:0]:** 预分频因子。这些位具有写保护设置(参照章节: “寄存器访问保护”), 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。具体的值与预分频因子对应关系如下: 000: 预分频因子 = 4 001: 预分频因子 = 8 010: 预分频因子 = 16 011: 预分频因子 = 32 100: 预分频因子 = 64 101: 预分频因子 = 128 110: 预分频因子 = 256 111: 预分频因子 = 256 ***注意: 对此寄存器进行读操作, 将从 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。***

21.4.3 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
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	RL[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1

位 31:12	**保留:**必须保持复位值
位 11:0	**RL[11:0]:** 看门狗计数器重载值。这些位具有写保护设置(参考章节: 26.3.2 “寄存器访问保护”)。RL 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。 看门狗的超时周期可以通过此重装载值和时钟预分频值来计算, 可以参考表: **IWDG 超时时间表 (40KHz 的输入时钟(LSI))。** 只有当 IWDG_SR 寄存器中的 RVU 位为 0 时, 才能对此寄存器进行修改。 注意: 对此寄存器进行读操作, 将从 VDD 电压域返回预分频值。如果写操作正在进行的话, 则读回的值可能是无效的。因此, 只有当 IWDG_SR 寄存器中的 RVU 位为 0 时, 读出的值才有效。

21.4.4 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)

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

														r	r
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位 31:2	**保留:**必须保持复位值
位 1	**RVU:** 看门狗计数器重装载值更新指示。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指示。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。

21.4.5 窗口寄存器(IWDG_WINR)

地址偏移: 0x10

复位值: 0x0000 0FFF(待机模式时复位)

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

位 31:12	**保留:**必须保持复位值
位 1	**WIN[11:0]:** 看门狗计数器窗口值。

22 系统窗口看门狗 (WWDG)

22.1 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。

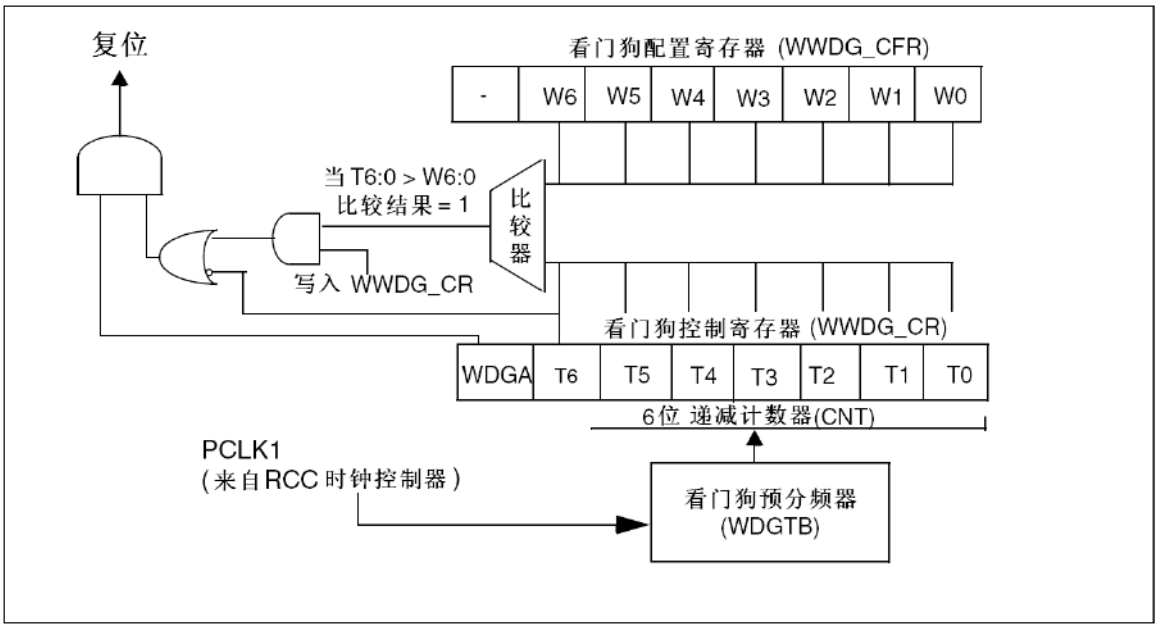
22.2 WWDG 主要特性

- 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。
- 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。

22.3 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 22-1 WWDG 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：

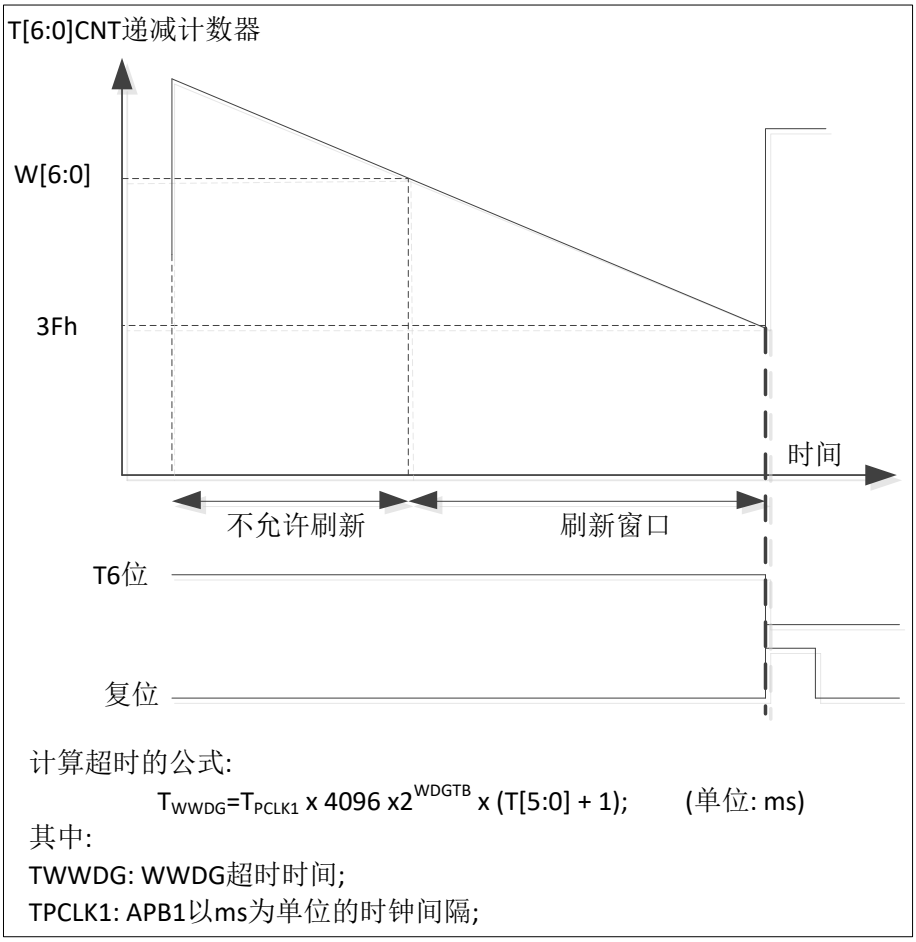
- 启动看门狗 在系统复位后，看门狗总是处于关闭状态，设置 WWDG_CR 寄存器的 WDGA 位能够开启看 门狗，随后它不能再被关闭，除非发生复位。
- 控制递减计数器 递减计数器处于自由运行状态，即使看门狗被禁止，递减计数器仍继续递减计数。当看门 狗被启用时，T6 位必须被设置，以防止立即产生一个复位。 T[5:0]位包含了看门狗产生复位之前的计时数目；复位前的延时时间在一个最小值和一个最 大值之间变化，这是因为写入 WWDG_CR 寄存器时，预分频值是未知的。 配置寄存器(WWDG_CFR) 中包含窗口的上限值：要避免产生复位，递减计数器必须在其值 小于窗口寄存器的数值并且大于 0x3F 时被重新装载，0 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。

注意: 可以用 T6 产生一个软件复位(设置 WDGA 位为'1', T 位为'0')。

22.4 如何编写看门狗超时程序

可以使用下图中提供的公式计算 WWDG 的超时时间(***警告: 当写入 WWDG 寄存器时, 请始终置 T6 位为'1'以避免立即产生一个复位***)。

图 22-2 WWDG 工作时间计算说明



计算超时的公式:

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

其中:

TWWDG: WWDG超时时间;

TPCLK1: APB1以ms为单位的时钟间隔;

表 22-1 在 PCLK1 = 36MHz 时的最小~最大超时时间表

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

22.5 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。

22.6 WWDG 寄存器

22.6.1 控制寄存器(WWDG_CR)

地址偏移: 0x00

复位值: 0x0000 007F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WDGA	T6	T5	T4	T3	T2	T1	T0
								rs	rw	rw	rw	rw	rw	rw	rw
0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1

位 31:8	**保留:**必须保持复位值
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(4096x2WDGTB)个 PCLK1 周期减 1。当计数器值从 40h 变为 3Fh 时(T6 变成 0)，产生看门狗复位。

22.6.2 配置寄存器(WWDG_CFR)

地址偏移: 0x04

复位值: 0x0000 007F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	EWI	WDGTB1	WDGTB0	W6	W5	W4	W3	W2	W1	W0
						rs	rw	rw	rw	rw	rw	rw	rw	rw	rw
0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1

位 31:10	**保留:**必须保持复位值
位 9	**EWI:** 提前唤醒中断。此位若置'1'，则当计数器值达到 40H 时即会产生中断。此中断只能由硬件在复位后清除。
位 8:7	**WDGTB[1:0]:** 预分频器的时基分频设置位。 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。

22.6.3 状态寄存器(WWDG_SR)

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

位 31:1	**保留:**必须保持复位值
位 1	**EWIF:** 提前唤醒中断标志。当计数器达到 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'。

23 实时时钟 (RTC)

23.1 简介

实时时钟(RTC)提供用于管理所有低功耗模式的自动唤醒单元。

实时时钟(RTC)是一个独立的BCD定时器/计数器。RTC提供具有可编程闹钟中断功能的日历时钟/日历。

RTC 还包含具有中断功能的周期性可编程唤醒标志。

两个 32 位寄存器包含二进码十进数格式(BCD)的秒、分钟、小时（12 或 24 小时制）、星期几、日期、月份和年份。此外，还可提供二进制格式的亚秒值。

系统可以自动将月份的天数补偿为 28、29（闰年）、30 和 31 天。并且还可以进行夏令时补偿。

其它 32 位寄存器还包含可编程的闹钟亚秒、秒、分钟、小时、星期几和日期。此外，还可以使用数字校准功能对晶振精度的偏差进行补偿。

RTC 域复位后，所有 RTC 寄存器都会受到保护，以防止可能的非正常写访问。

无论器件状态如何（运行模式、低功耗模式或处于复位状态），只要电源电压保持在工作范围内，RTC 便不会停止工作。

23.2 RTC 主要特性

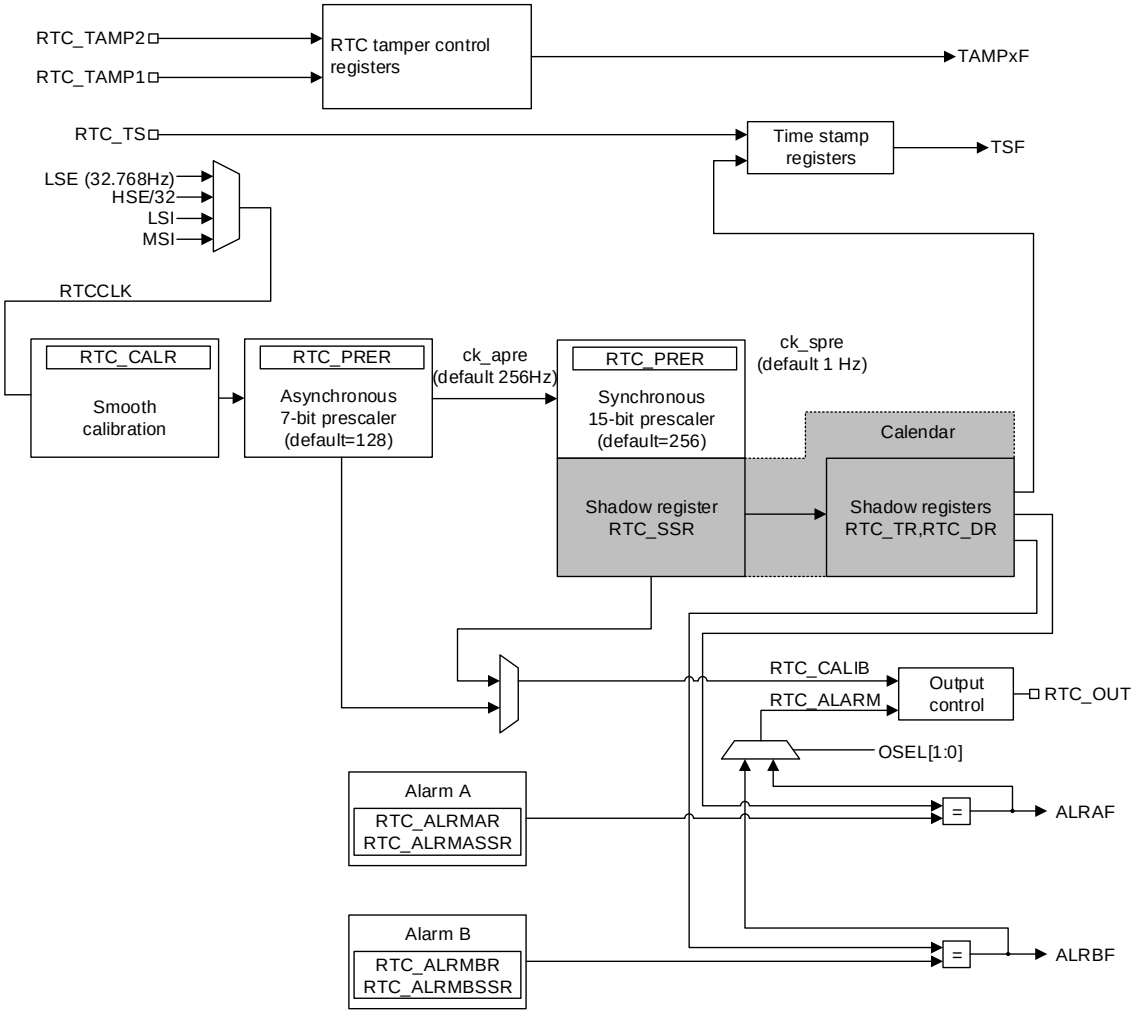
RTC 单元的主要特性如下：

- 包含亚秒、秒、分钟、小时（12/24 小时制）、星期几、日期、月份和年份的日历。
- 软件可编程的夏令时补偿。
- 具有中断功能的可编程闹钟。可通过任意日历字段的组合触发闹钟。
- 自动唤醒单元，可周期性地生成标志以触发自动唤醒中断。
- 参考时钟检测：可使用更加精确的第二时钟源（50 Hz 或 60 Hz）来提高日历的精确度。
- 利用亚秒级移位特性与外部时钟实现精确同步。
- 数字校准电路（周期性计数器调整）：精度为 0.95 ppm，在数秒钟的校准窗口中获得
- 用于事件保存的时间戳功能
- 带可配置过滤器和内部上拉的入侵检测事件
- 可屏蔽中断/事件：
  - 闹钟 A
  - 闹钟 B
  - 唤醒中断 (Wakeup interrupt)
  - 时间戳
  - 入侵检测
- 5 个备份寄存器。

23.3 RTC 功能说明

23.3.1 RTC 框图

图 23-1 RTC 框图



23.3.2 RTC 控制的 GPIO

23.3.3 时钟和预分频器

由时钟控制器从以下 4 种时钟中选择 RTC 时钟源（RTCCLK）：

- LSE 振荡器作为 RTC 时钟；
- LSI 振荡器作为 RTC 时钟；
- HSE 振荡器作为 RTC 时钟；
- MSI 振荡器作为 RTC 时钟。

一个可编程预分频器产生一个 1HZ 的时钟，用于更新日历。为最大限度地减少功耗，预分频器可分割成 2 个可编程预分频器。（见图 8-1 框图）：

- 通过控制 RTC_PRER 寄存器中 PREDIV_A 位来配置 7 位异步预分频器。

- 通过控制 RTC_PRER 寄存器中 PREDIV_S 位来配置 15 位同步预分频器。

注：当所有分频器被启用时，建议将异步预分频器的值调高以最大限度地减少功率消耗

异步预分频器的分频系数设为 128， 同步预分频器的分频系数设为 256， 从而得到一个基于 32.768 kHz LSE 频率的 1 Hz (ck_spre) 内部时钟频率。

最小分频系数为 1， 最大分频系数为 2^{22}， 相当于最大输入频率大约 4MHz。

f_{ck_apre} 满足：

$$f_{CK\_APRE} = \frac{f_{RTCCLK}}{PREDIV\_A + 1}$$

ck_apre 时钟为二进制 RTC_SSR 亚秒递减计数器提供时钟。 当值为 0 时， RTC_SSR 的值将被重置为 PREDIV_S 里的值。

f_{ck_spre} 满足：

$$f_{ck\_spre} = \frac{f_{RTCCLK}}{(PREDIV\_S + 1) \times (PREDIV\_A + 1)}$$

23.3.4 实时时钟和日历

RTC 的日历时间寄存器和日期寄存器是通过影子寄存器访问的，该影子寄存器与 PCLK (APB clock)同步。为避免同步等待时间，也可直接访问。

- RTC_SSR ： 亚秒
- RTC_TR ： 时间
- RTC_DR ： 日期

硬件会每两个 RTCCLK 时钟周期更新一次影子寄存器的日历值，并将 RTC_ISR 寄存器的 RSF 位置 1。停止或待机模式下则不作这个更新，当退出上述两种模式后，影子寄存器将在最多两个 RTCCLK 周期后执行更新操作。

默认情况下，应用程序通过访问影子寄存器获取日历寄存器的内容。如需直接访问日历寄存器，可通过设置 RTC_CR 寄存器的 BYPSHAD 控制位（默认为零）来实现。

在 BYPSHAD=0 模式下，如需读取 RTC_SSR, RTC_TR 或 RTC_DR 寄存器的内容，APB 时钟频率(fAPB)至少是 RTC 时钟频率(fRTCCLK)的 7 倍。

系统复位后，影子寄存器也将自动复位。

23.3.5 可编程闹钟

RTC 单元提供两个可编程闹钟，即闹钟 A 和闹钟 B。以下说明针对闹钟 A，但同样适用于闹钟 B。

可通过 RTC_CR 寄存器中的 ALRAE 位来使能可编程闹钟功能。如果日历亚秒、秒、分钟、小时、日期或日与闹钟寄存器 RTC_ALRMASSR 和 RTC_ALRMAR 中编程的值相匹配，则 ALRAF 标志会被置为 1。可通过 RTC_ALRMAR 寄存器的 MSKx 位以及 RTC_ALRMASSR 寄存器的 MASKSSx 位单独选择各日历字段。可通过 RTC_CR 寄存器中的 ALRAIE 位来使能闹钟中断。

注意：如果选择秒字段（RTC_ALRMAR 中的 MSK1 位复位），则 RTC_PRER 寄存器中设置的同步预分频器分频系数必须至少为 3，才能确保闹钟正确地运行。

闹钟 A 和闹钟 B（如果已通过 RTC_CR 寄存器中的位 OSEL[0:1]使能）可连接到 RTC_ALARM 输出。可通过 RTC_CR 寄存器的 POL 位配置 RTC_ALARM 输出极性。

23.3.6 周期性自动唤醒

周期性唤醒标志由 16 位可编程自动重载递减计数器生成。唤醒定时器范围可扩展至 17 位。

可通过 RTC_CR 寄存器中的 WUTE 位来使能此唤醒功能。

唤醒定时器的时钟输入可以是：

- 2、4、8 或 16 分频的 RTC 时钟(RTCCLK)。

当 RTCCLK 为 LSE(32.768kHz)时，可配置的唤醒中断周期介于 122μs 和 32s 之间，且分辨率低至 61μs。

ck_spre（通常为 1Hz 内部时钟）。

当 ck_spre 频率为 1Hz 时，可得到的唤醒时间为 1s 到 36h 左右，分辨率为 1 秒。这一较大的可编程时间范围分为两部分：

- WUCKSEL[2:1]=10 时为 1s 到 18h
- WUCKSEL[2:1]=11 时约为 18h 到 36h。在后一种情况下，会将 216 添加到 16 位计数器当前值。完成初始化序列后，定时器开始递减计数。在低功耗模式下使能唤醒功能时，递减计数保持有效。此外，当计数器计数到 0 时，RTC_ISR 寄存器的 WUTF 标志会置 1，并且唤醒寄存器会使用其重载值（RTC_WUTR 寄存器值）动重载。

之后必须用软件清零 WUTF 标志。

通过将 RTC_CR2 寄存器中的 WUTIE 位置 1 来使能周期性唤醒中断时，它会使器件退出低功耗模式。

如果已通过 RTC_CR 寄存器的位 OSEL[1:0]使能周期性唤醒标志，则该标志可连接到 RTC_ALARM 输出。可通过 RTC_CR 寄存器的 POL 位配置 RTC_ALARM 输出极性。系统复位以及低功耗模式（睡眠、停机和待机）对唤醒定时器没有任何影响。

23.3.7 RTC 初始化和配置

RTC 寄存器访问

RTC 寄存器为 32 位寄存器。除了当 BYPSHAD=0 时对日历影子寄存器执行的读访问之外，APB 接口会在访问 RTC 寄存器时引入 2 个等待周期。

RTC 寄存器写保护

系统复位后，可通过对 PWR_CR 寄存器中的 DBP 位清零来保护 RTC 寄存器以防止非正常的写访问。必须将 DBP 位置 1 才能使能 RTC 寄存器的写访问。

RTC 域复位后，所有 RTC 寄存器均受到写保护。通过向写保护寄存器(RTC_WPR)写入一个密钥来使能对 RTC 寄存器的写操作。

要解锁所有 RTC 寄存器（RTC_TAMPCR、RTC_BKPxR、RTC_OR 和 RTC_ISR[13:8]除外）的写保护，需要执行以下步骤：

1.将“0xCA”写入 RTC_WPR 寄存器。

2.将“0x53”写入 RTC_WPR 寄存器。

写入一个错误的关键字会再次激活写保护。

保护机制不受系统复位影响。

日历初始化和配置

要编程包括时间格式和预分频器配置在内的初始时间和日期日历值，需按照以下顺序操作：

1. 将 RTC_ISR 寄存器中的 INIT 位置 1 以进入初始化模式。在此模式下，日历计数器将停止工作并且其值可更新。
2. 轮询 RTC_ISR 寄存器中的 INITF 位。当 INITF 置 1 时进入初始化阶段模式。大约需要 2 个 RTCCLK 时钟周期（由于时钟同步）。
3. 要为日历计数器生成 1Hz 时钟，应编程 RTC_PRER 寄存器中的两个预分频系数。
4. 在影子寄存器（RTC_TR 和 RTC_DR）中加载初始时间和日期值，然后通过 RTC_CR 寄存器中的 FMT 位配置时间格式（12 或 24 小时制）。
5. 通过清零 INIT 位退出初始化模式。随后，自动加载实际日历计数器值，在 4 个 RTCCLK 时钟周期后重新开始计数。

当初始化序列完成之后，日历开始计数。

注：1.系统复位后，应用可读取 RTC_ISR 寄存器中的 INITS 标志，以检查日历是否已初始化。如果该标志为 0，表明日历尚未初始化，因为年份字段设置为其 RTC 域复位时的默认值(0x00)。

2.要在初始化之后读取日历，必须首先用软件检查 RTC_ISR 寄存器的 RSF 标志是否置 1。

夏令时

可通过 RTC_CR 寄存器的 SUB1H、ADD1H 和 BKP 位管理夏令时。

利用 SUB1H 或 ADD1H，软件只需单次操作便可在日历中减去或增加一个小时，无需执行整个初始化步骤。

此外，软件还可以使用 BKP 位来记录是否曾经执行过此操作。

编程闹钟

要对可编程的闹钟进行编程或更新，必须执行类似的步骤。以下步骤针对闹钟 A，但同样适用于闹钟 B。

1. 将 RTC_CR 中的 ALRAE 位清零以禁止闹钟 A。
2. 编程闹钟 A 寄存器(RTC_ALRMASSR/RTC_ALRMAR)。
3. 将 RTC_CR 寄存器中的 ALRAE 位置 1 以再次使能闹钟 A。

注：约 2 个 RTCCLK 时钟周期（由于时钟同步）后，将执行对 RTC_CR 寄存器的更改。

编程唤醒定时器

要配置或更改唤醒定时器的自动重载值（RTC_WUTR 中的 WUT[15:0]），需要按照以下顺序操作：

1. 清零 RTC_CR 中的 WUTE 以禁止唤醒定时器。
2. 轮询 RTC_ISR 中的 WUTWF，直到该位置 1，以确保可以访问唤醒自动重载定时器和 WUCKSEL[2:0]位。大约需要 2 个 RTCCLK 时钟周期（由于时钟同步）。
3. 编程唤醒自动重载值 WUT[15:0]，并选择唤醒时钟（RTC_CR 中的 WUCKSEL[2:0]位）。将 RTC_CR 寄存器中的 WUTE 位置 1 以再次使能定时器。唤醒定时器重新开始递减计数。由于时钟同步，WUTWF 位会在 WUTE 清零达 2 个 RTCCLK 时钟周期后清零。

23.3.8 读取日历

- **当 RTC_CR 寄存器中的 BYPSHAD 控制位清零时**

要正确读取 RTC 日历寄存器（RTC_SSR、RTC_TR 和 RTC_DR），APB1 时钟频率(f_{PCLK})必须等于或大于 RTC 时钟频率(f_{RTCCLK})的七倍。这可以确保同步机制行为的安全性。

如果 APB1 时钟频率低于 RTC 时钟频率的七倍，则软件必须读取日历时间寄存器和日期寄存器两次。这样，当两次读取的 RTC_TR 结果相同时，才能确保数据正确。否则必须执行第三次读访问。任何情况下，APB1 的时钟频率都不能低于 RTC 的时钟频率。

每次将日历寄存器中的值复制到 RTC_SSR、RTC_TR 和 RTC_DR 影子寄存器时，RTC_ISR 寄存器中的 RSF 位都会置 1。每两个 RTCCLK 周期执行一次复制。为确保这 3 个值一致，读取 RTC_SSR 或 RTC_TR 时会锁定高阶日历影子寄存器中的值，直到读取 RTC_DR。为避免软件对日历执行读访问的时间间隔小于 2 个 RTCCLK 周期：第一次读取日历之后必须通过软件将 RSF 清零，并且软件必须等待到 RSF 置 1 之后才可再次读取 RTC_SSR、RTC_TR 和 RTC_DR 寄存器。

从低功耗模式（停机模式或待机模式）唤醒之后，必须通过软件将 RSF 清零。之后，软件必须等待至 RSF 再次置 1 之后才可以读取 RTC_SSR、RTC_TR 和 RTC_DR 寄存器。

RSF 位必须在唤醒之后而不是进入低功耗模式之前进行清零。

系统复位之后，软件必须等待至 RSF 置 1 之后才可以读取 RTC_SSR、RTC_TR 和 RTC_DR 寄存器。实际上，系统复位会将影子寄存器复位为其默认值。

初始化之后，软件必须等待至 RSF 置 1 之后才可以读取 RTC_SSR、RTC_TR 和 RTC_DR 寄存器。

同步之后，软件必须等待至 RSF 置 1 之后才可以读取 RTC_SSR、RTC_TR 和 RTC_DR 寄存器。

- **当 RTC_CR 寄存器中的 BYPSHAD 控制位置 1 时（旁路影子寄存器）**

读取日历寄存器时会直接从日历计数器获取值，这样便无需等待至 RSF 位置 1。这对于从低功耗模式（停机模式或待机模式）退出后的情况特别有用，因为影子寄存器在这些模式下不更新。

当 BYPSHAD 位置 1 时，如果在对寄存器的两次读访问之间出现 RTCCLK 沿，则不同寄存器的结果彼此可能不一致。此外，如果在读操作期间出现 RTCCLK 沿，则可能导致其中一个寄存器的值不正确。软件必须分两次读取所有寄存器，然后将两次结果加以比较来确认数据是否一致和正确。此外，软件也可以只比较两次读取日历寄存器得到的结果的最低位。

注：当 BYPSHAD=1 时，读取日历寄存器的指令需要一个额外的 APB 周期才能完成。

23.3.9 复位 RTC

日历影子寄存器（RTC_SSR、RTC_TR 和 RTC_DR）以及 RTC 状态寄存器(RTC_ISR)的某些位通过所有可用的系统复位源复位为各自的默认值。

相反，以下寄存器则通过 RTC 域复位来复位为各自的默认值并且不受系统复位的影响：RTC 当前日历寄存器、RTC 控制寄存器(RTC_CR)、预分频器寄存器(RTC_PRER)、RTC 校准寄存器(RTC_CALR)、RTC 移位寄存器(RTC_SHIFTR)、RTC 时间戳寄存器（RTC_TSSSR、RTC_TSTR 和 RTC_TSDR）、RTC 入侵和复用功能配置寄存器(RTC_TAMPCR)、RTC 备份寄存器(RTC_BKPxR)、唤醒定时器寄存器(RTC_WUTR)、闹钟 A 和闹钟 B 寄存器（RTC_ALRMASSR/RTC_ALRMAR 和 RTC_ALRMBSSR/RTC_ALRMBR）以及选项寄存器(RTC_OR)。

此外，当由 LSE 提供时钟时，如果复位源并非 RTC 域复位源（有关不受系统复位影响的 RTC 时钟源列表的详细信息，请参见“复位和时钟控制器”的 RTC 时钟），则 RTC 将在系统复位时保持运行状态。发生 RTC 域复位时，RTC 会停止工作，并且所有 RTC 寄存器都会设置为各自的复位值。

23.3.10 RTC 同步

RTC 可与高精度的远程时钟同步。在读取亚秒字段后（RTC_SSR 或 RTC_TSSSR），即可计算远程时钟的时间与 RTC 之间的精准偏差。之后，可使用 RTC_SHIFTR 对 RTC 的时钟进行零点几秒的“平移”，经过调整后可消除此偏差。

RTC_SSR 包含同步预分频器计数器的值。这样，便可计算分辨率低至 1/(PREDIV_S+1)

秒的 RTC 的准确时间。因此，可通过增大同步预分频器的值(PREDIV_S[14:0])来提高分辨率。将 PREDIV_S 设置为 0x7FFF 时，可得到允许的最大分辨率（30.52μs，时钟频率为 32768Hz）。

但是，提高 PREDIV_S 意味着必须降低 PREDIV_A 才能将同步预分频器的输出维持在 1Hz。这样，异步预分频器的输出频率会增大，RTC 的动态功耗也会相应增加。

可以使用 RTC 平移控制寄存器(RTC_SHIFTR)对 RTC 进行微调。可以用大小为 1/(PREDIV_S+1)秒的分辨率对 RTC_SHIFTR 进行写操作，将时钟平移（延迟或提前）最长 1 秒。在这种平移操作中，会将 SUBFS[14:0]值加到同步预分频器计数器 SS[15:0]中：这将使时钟产生延迟。如果同时将 ADD1S 位置 1，则会增加一秒，与此同时减去的时间为零点几秒，因此将使时钟提前。

注：初始化平移操作前，用户必须检查确认 SS[15]=0，以确保不会发生上溢。

对 RTC_SHIFTR 寄存器执行写操作以启动平移操作时，硬件会将 SHPF 标志置 1 以指示平移操作挂起。完成平移操作时，硬件会将将该位清零。

注：该同步功能与参考时钟检测功能不兼容：当 REFCKON=1 时，固件不能对 RTC_SHIFTR 执行写操作。

23.3.11 RTC 参考时钟检测

RTC 日历更新可与参考时钟 RTC_REFIN（通常为市电频率，50Hz 或 60Hz）同步。

RTC_REFIN 参考时钟的精度应高于 32.768kHzLSE 时钟。使能 RTC_REFIN 检测时（将 RTC_CR 的 REFCKON 位置 1），日历仍由 LSE 提供时钟，而 RTC_REFIN 用于补偿不准确的日历更新频率(1Hz)。

每个 1Hz 时钟边沿都与最近的 RTC_REFIN 时钟边沿进行比较（如果在给定的时间窗口内发现一个边沿）。

在大多数情况下，两个时钟边沿恰好对齐。当 1Hz 时钟由于 LSE 时钟不精确而发生偏离时，RTC 会稍微偏移 1Hz 时钟，以便后续的 1Hz 时钟边沿能够对齐。利用这种机制，可使日历像参考时钟一样精确。

RTC 使用 32.768kHz 石英产生的 256Hz 时钟(ck_apre)检测是否存在参考时钟源。大约在日历每次更新时（每 1 秒钟），便会在时间窗口期间执行一次检测。检测到第一个参考时钟边沿时，该窗口等于 7 个 ck_apre 周期。随后的日历更新使用长度为 3 个 ck_apre 周期的较小窗口。

每次在窗口中检测到参考时钟时，都会行强制输出 ck_apre 时钟的异步预分频器进行重载。

当参考时钟与 1Hz 时钟对齐时，此操作不起作用，因为预分频器会在同一时刻重载。当时钟不对齐时，重载操作会微调后续的 1Hz 时钟边沿，使其与参考时钟对齐。

如果参考时钟停止（在 3 个 ck_apre 窗口内未出现参考时钟边沿），日历将仅根据 LSE 时钟进行连续更新。RTC 随后使用 ck_spre 边沿上居中的大检测窗口（7 个 ck_apre 周期）等待参考时钟。

使能 RTC_REFIN 检测后，必须将 PREDIV_A 和 PREDIV_S 设置为各自的默认值：

- PREDIV_A=0x007F
- PREVID_S=0x00FF

注：RTC_REFIN 时钟检测在待机模式下不可用

23.3.12 RTC 精密数字校准

RTC 频率可采用约 0.954ppm 的分辨率进行数字校准，校准范围为-487.1ppm 到+488.5ppm。使用一系列微调（增加和/或减少单独的 RTCCLK 脉冲）进行频率校正。这些微调的分布非常均匀，因此 RTC 的校准效果相当好，即使在短时间内持续观察也是如此。

当输入频率为 32768Hz 时，精密数字校准的周期约为 220 个 RTCCLK 脉冲或 32 秒。此周期由一个通过 RTCCLK 提供时钟信号的 20 位计数器 cal_cnt[19:0]维持。

精密数字校准寄存器(RTC_CALR)可指定 32 秒周期内要减少的 RTCCLK 时钟周期数：

- 将 CALM[0]置 1 时，32 秒周期内将只减少 1 个脉冲。
- 将 CALM[1]置 1 时，将减少两个周期。
- 将 CALM[2]置 1 时，将减少 4 个周期
- 依此类推，将 CALM[8]置 1 时，将减少 256 个时钟。

注：CALM[8:0](RTC_CALR)可指定 32 秒周期内要减少的 RTCCLK 脉冲数。将位 CALM[0]置 1 时，32 秒周期内将只减少 1 个脉冲（当 cal_cnt[19:0]=0x80000 时）；将 CALM[1]置 1 时，将减少 2 个周期（当 cal_cnt=0x40000 和 0xC0000 时）；将 CALM[2]置 1 时，将减少 4 个周期（当 cal_cnt=0x20000/0x60000/0xA0000/0xE0000 时）；依此类推，将 CALM[8]置 1 时，将减少 256 个时钟（当 cal_cnt=0xXX800 时）。

使用适当分辨率时，CALM 可使 RTC 频率减少最多 487.1ppm，而 CALP 可用于使频率增加 488.5ppm。将 CALP 置“1”，可每隔 211 个 RTCCLK 周期有效插入一个额外的 RTCCLK 脉冲，这意味着每 32 秒周期可增加 512 个时钟。

与 CALM 和 CALP 配合使用时，可在 32 秒周期内增加一个范围为-511 到+512RTCCLK 周期的偏差，对应的校准范围为-487.1ppm 到+488.5ppm，分辨率约为 0.954ppm。

若输入频率(FRTCCLK)已知，可通过以下公式计算有效校准频率(FCAL)：

FCAL=FRTCCLKx[1+(CALPx512-CALM)/(220+CALM-CALPx512)]

PREDIV_A<3 条件下的校准

当异步预分频器值（RTC_PRER 寄存器中的 PREDIV_A 位）小于 3 时，不能将 CALP 位置 1。如果 CALP 已置 1 并且 PREDIV_A 位的值小于 3，则会忽略 CALP，即假定 CALP 等于 0 而执行校准。

要在 PREDIV_A 小于 3 的条件下执行校准，应降低同步预分频器值(PREDIV_S)以便每秒内可加速 8 个 RTCCLK 时钟周期，这意味着每 32 秒可增加 256 个时钟周期。因此，仅使用 CALM 位，可在每 32 秒内有效增加 255 到 256 个时钟脉冲（对应的校准范围为 243.3ppm 到 244.1ppm）。

在标称 RTCCLK 频率 32768Hz 下，当 PREDIV_A 等于 1 时（分频系数为 2），应将 PREDIV_S 设置为

16379 而不是 16383（少 4）。唯一相关的其它情况是，当 PREDIV_A 等于 0 时，应将 PREDIV_S 设置为 32759 而不是 32767（少 8）。

如果以这种方式减少 PREDIV_S，则采用以下公式计算校准输入时钟的有效频率：

FCAL=FRTCCLKx[1+(256-CALM)/(220+CALM-256)]

在这种情况下，如果 RTCCLK 恰好为 32768.00Hz，则当 CALM[7:0]等于 0x100 时（CALM 范围的中值），说明设置正确。

验证 RTC 校准

通过测量 RTCCLK 的精确频率，计算正确的 CALM 和 CALP 值以确保 RTC 精度。此外，还为应用提供了一个可选的 1Hz 输出，用来测量和验证 RTC 精度。

如果在有限的间隔内测量 RTC 的精确频率，则会导致测量期间产生最多 2 个 RTCCLK 时钟周期的测量误差，具体取决于数字校准周期与测量周期的对齐方式。

但是，如果测量周期与校准周期的长度相同，则可以消除此测量误差。在这种情况下，观测到的唯一误差是由数字校准的分辨率导致的误差。

- 默认情况下，校准周期为 32 秒。

在此模式下，测量整个 32 秒内 1Hz 输出的精度，可确保测量误差在 0.477ppm 内（32 秒内为 0.5 个 RTCCLK 周期，受校准分辨率限制）。

- 可将 RTC_CALR 寄存器的 CALW16 位置 1，以强制 16 秒的校准周期。

此时，可在 16 秒内测量 RTC 精度，产生的最大误差为 0.954ppm（16 秒内为 0.5 个 RTCCLK 周期）。但是，由于校准分辨率降低，长期的 RTC 精度也会降到 0.954ppm：将 CALW16 置 1 时，CALM[0]位将始终保持为 0。

- 可将 RTC_CALR 寄存器的 CALW8 位置 1，以强制 8 秒的校准周期。

此时，可在 8 秒内测量 RTC 精度，产生的最大误差为 1.907ppm（8 秒内为 0.5 个 RTCCLK 周期）。长期的 RTC 精度也会降到 1.907ppm：将 CALW8 置 1 时，CALM[1:0]位将始终保持为 00。

动态重校准

当 RTC_ISR/INITF=0 时，可动态更新校准寄存器(RTC_CALR)，具体步骤如下：

1. 轮询 RTC_ISR/RECALPF（重新校准挂起标志）。
2. 如果该标志为 0，则可以根据需要向 RTC_CALR 写入新值。随后 RECALPF 位会被自动置为 1。
3. 新校准设置将在对 RTC_CALR 执行写操作之后的三个 ck_apre 周期内生效。

23.3.13 时间戳功能

将 RTC_CR 寄存器的 TSE 位置 1 可使能时间戳。

当在 RTC_TS 引脚上检测到时间戳事件时，日历会保存到时间戳寄存器（RTC_TSSSR、RTC_TSTR 和 RTC_TSDR）中。发生时间戳事件时，RTC_ISR 寄存器中的时间戳标志位(TSF)将置 1。

通过将 RTC_CR 寄存器中的 TSIE 位置 1，可在发生入侵检测事件时生成中断。

如果在时间戳标志(TSF)已置 1 的条件下检测到新的时间戳事件，则时间戳上溢标志(TSOVF)将置 1，而时间戳寄存器（RTC_TSTR 和 RTC_TSDR）将保持上一事件的结果。

注：在发生由同步过程引发的时间戳事件后，TSF 在 2 个 ck_apre 周期置为 1。

TSOVF 的产生中不存在延迟。也就是说如果两个时间戳事件接连发生，TSOVF 可能为“1”而 TSF 为“0”。因此，建议只在检测到 TSF 为“1”后再轮询 TSOVF。

注意：如果在 TSF 位清零后紧接着发生时间戳事件，则 TSF 和 TSOVF 位都将置 1。为防止在时间戳事件发生的同时屏蔽该事件，除非已将 TSF 位读为“1”，否则应用程序不得将“0”写入 TSF 位。

23.3.14 入侵检测

RTC_TAMPx 输入事件可设置为边沿检测或带过滤功能的电平检测。

RTC 备份寄存器

备份寄存器(RTC_BKPxR)处在 VDD 备份域中，当 VDD 电源被切断，他们仍由 VBAT 维持供电。当系统复位或设备在待机模式下被唤醒时，他们不会被复位。上电复位时，备份寄存器将被复位。

当产生一个入侵检测事件时，备份寄存器将被复位。

入侵检测初始化

所有 RTC_TAMPx 入侵检测输入均与 RTC_ISR2 寄存器的 TAMPxF 标志相关。所有输入可通过设置 RTC_TAFCR 寄存器中相应的 TAMPxE 位为“1”来使能。

入侵检测事件可将所有备份寄存器(RTC_BKPxR)内容清除。

通过设置 RTC_TAFCR 寄存器的 TAMPIE 位，当检测到一个入侵检测事件时便产生一个中断。

入侵事件时间戳

设置 TAMPTS 为“1”，所有入侵事件将产生一个时间戳。这种情况下，RTC_ISR 中无论是 TSF 位还是 TSOVF 位被置 1，和正常的时间戳事件发生的效果相同。设置 TSF 或 TSOVF，与其关联的的入侵标志寄存器 TAMPxF 将同时被设置。

入侵输入的边沿检测如果 TAMPFLT 位为“00”，根据相关 TAMPxTRG 位，当检测到上升沿或下降沿时，RTC_TAMPx 引脚将生成入侵检测事件。当边沿检测被选中后，RTC_TAMPx 输入的内部上拉电阻将被停用。

警告:为避免入侵检测事件丢失并及时地发现入侵检测事件,用于边沿检测的信号与相应TAMPxE 位执行逻辑与操作,以便在RTC_TAMPx 引脚启用钱检测到入侵检测事件。

- 当 TAMPxTRG=0：如果在启用入侵检测（通过设置 TAMPxE 位为 1）前 RTC_TAMPx 已经为高电平，一旦启用 RTC_TAMPx 输入，则会产生一个入侵事件（尽管在 TAMPxE 位置“1”后 RTC_TAMPx 输入中并没有出现上升沿）
- 当 TAMPxTRG=1：如果在启用入侵检测前，RTC_TAMPx 已经为低电平，一旦启用 RTC_TAMPx，则会产生一个入侵事件（尽管在设置 TAMPxE 位后 RTC_TAMPx 输入中并没有出现下降沿）。

在一个入侵事件被检测到并清除后，RTC_TAMPx 复用功能应该被禁止。然后在再次写入备份寄存器前重新启用 RTC_TAMPx 复用功能（通过设置 TAMPxE 位为 1）。这样可以阻止软件在 RTC_TAMPx 检测出仍有入侵事件发生时对备份寄存器进行写操作。这相当于对 RTC_TAMPx 复用功能输入进行电平检测。

注:当电源断开时，入侵检测功能仍然有效。为避免对备份寄存器产生不必要的复位，RTC_TAMPx 复用功能对应的引脚应该在片外连接到正确的电平。

针对 RTC_TAMPx 输入的带滤波功能电平检测将 TAMPFLT 设置为非“0”值，会启用带滤波功能的电平检测。当检测到 2 个、4 个或 8 个（根据 TAMPFLT 设置）连续样本的指定电平（TAMPxTRG 位决定）时，将产生一个入侵检测事件。

RTC_TAMPx 输入在其状态被采样前，通过 I/O 内部上拉电阻实现预充电，直至 TAMPPUDIS 设置为“1”后，停止该功能。预充电时间由 TAMPPRCH 位决定，允许 RTC_TAMPx 输入拥有更大的电容。

可通过调整 TAMPFREQ 来改变电平检测的采样频率，从而在入侵检测延迟与通过上拉电阻产生的电源消耗间进行取舍。

23.3.15 校准时钟输出

将 RTC_CR 寄存器中的 COE 位置 1 时，会在 RTC_CALIB 器件输出上提供一个参考时钟。如果 RTC_CR 寄存器中的 COSEL 位复位且 PREDIV_A=0x7F，则 RTC_CALIB 频率为 fRTCCLK/64。这相当于 RTCCLK 频率为 32.768kHz 时，512Hz 的校准输出。RTC_CALIB 占空比是不规则的：下降沿上存在轻微抖动。因

此推荐使用上升沿。

如果 COSEL 置 1 且“PREDIV_S+1”为 256 的非零整数倍（比如：PREDIV_S[7:0]=0xFF），则 RTC _CALIB 频率为 fRTCCLK/(256*(PREDIV_A+1))。这相当于 RTCCLK 频率为 32.768kHz 时，1Hz 的校准输出，其中预分频器为默认值（PREDIV_A=Ox7F、PREDIV_S=0xFF）。

注： 选择 RTC_CALIB 或 RTC_ALARM 输出时，RTC_OUT 引脚会自动配置为输出复用功能。

23.3.16 闹钟输出

RTC_CR 寄存器中的 OSEL[1:0]控制位用于激活闹钟复用功能输出 RTC_ALARM，以及选择输出的功能。这些功能可反映 RTC_ISR 寄存器中相应标志的内容。

输出的极性由 RTC_CR 中的 POL 控制位确定，这样当 POL 置 1 时会输出选定标志位的相反值。

闹钟复用功能输出

使用 RTC_OR 寄存器中的控制位 RTC_ALARM_TYPE 可将 RTC_ALARM 引脚配置为输出开漏或输出上拉。

注： 1.使能 RTC_ALARM 输出之后，其优先级高于 RTC_CALIB（与 COE 位无关，该为必须保持清零）。

2.选择 RTC_CALIB 或 RTC_ALARM 输出时，RTC_OUT 引脚会自动配置为输出复用功能。

23.3.17 RTC 低功耗模式

表 23-1 低功耗模式对 RTC 的作用

模式	说明
睡眠	无任何作用 RTC 中断可使器件退出睡眠模式。
停止	当 RTC 时钟源为 LSE 或 LSI 时，RTC 保持工作状态。RTC 闹钟、RTC 入侵事件、RTC 时间戳事件和 RTC 唤醒会使器件退出停机模式。
待机	当 RTC 时钟源为 LSE 或 LSI 时，RTC 保持工作状态。RTC 闹钟、RTC 入侵事件、RTC 时间戳事件和 RTC 唤醒会使器件退出待机模式。

23.3.18 RTC 中断

所有 RTC 中断均与 EXTI 控制器相连。

要使能 RTC 中断，需执行以下序列：

1. 将对应于 RTC 事件的 EXTI 线配置为中断模式并将其使能，然后选择上升沿有效。
2. 配置 NVIC 中的 RTCIRQ 通道并将其使能。
3. 将 RTC 配置为生成 RTC 中断。

表 23-2 中断控制位

中断事件	事件标志	使能控制位	退出睡眠模式	退出停止模式	退出待机模式
闹钟 A	ALRAF	ALRAIE	是	是(1)	是(1)
闹钟 B	ALRBF	ALRBE	是	是(1)	是(1)
RTC_TS 输入（时间戳）	TSF	TSIE	是	是(1)	是(1)
RTC_TAMP1 输入检测	TAMP1F	TAMPIE	是	是(1)	是(1)
RTC_TAMP2 输入检测	TAMP2F	TAMPIE	是	是(1)	是(1)
唤醒定时器中断	WUTF	WUTIE	是	是(1)	是(1)

1.仅当 RTC 时钟源为 LSE 或 LSI 时，才能从停机和待机模式唤醒。

23.4 RTC 寄存器

23.4.1 RTC 时间寄存器 (RTC_TR)

偏移地址： 0x00

复位值： 0x0000 0000

系统重置：BYPSHAD = 0 时为 0x00000000。BYPSHAD = 1 时不受影响。

注：RTC_TR 是日历时间影子寄存器，该寄存器只能在初始化模式下写入。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PM	HT		HU			
									R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	MNT			MNU				Res.	ST			SU			
	R/W	R/W	R/W	R/W	R/W	R/W	R/W		R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位 31:23	保留，始终保持复位值。
位 22	**PM:** AM/PM 标记 0: AM 或 24 小时制 1: PM
位 21:20	**HT:** 小时的十位（BCD 格式）
位 19:16	**HU:** 小时的个位（BCD 格式）
位 15	保留，始终读为 0。
位 14:12	**MNT:** 分钟的十位（BCD 格式）
位 11:8	**MNU:** 分钟的个位（BCD 格式）
位 6:4	**ST:** 秒的十位（BCD 格式）
位 3:0	**SU:** 秒的个位（BCD 格式）

23.4.2 RTC 日期寄存器 (RTC_DR)

偏移地址： 0x04

复位值： 0x0000 2101

系统重置： 当 BYPSHAD = 0 时为 0x00002101，当 BYPSHAD = 1 时不受影响。

注： RTC_DR 是日历日期影子寄存器。该寄存器只能在初始化模式下写入。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	YT				YU			
								rw	rw	rw	rw	rw	rw	rw	rw
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WDU			MT	MU				Res.	Res.	DT		DU			
rw	rw	rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw
0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	1

位 31:24	保留，始终保持复位值。
位 23:20	**YT:** 年的十位（BCD 格式）
位 21:20	**YU:** 年的个位（BCD 格式）

位 19:16	**YU:** 年的个位（BCD 格式）
位 15:13	**WDU:** 星期 000: 禁用 001: 星期一 … 111: 星期日
位 12	**MU:** 月的个位（BCD 格式）
位 11:8	**MU:** 月的个位（BCD 格式）
位 7:6	保留，始终读为 0。
位 5:4	**DT:** 日的十位（BCD 格式）
位 3:0	**DU:** 日的个位（BCD 格式）

23.4.3 RTC 控制寄存器 (RTC_CR)

偏移地址： 0x08

复位值： 0x0000 0000

系统复位： 不受影响

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	COE	OSEL		POL	COSEL	BKP	SUB1H	ADD1H
								rw	rw	rw	rw	rw	rw	w	w
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TSIE	WUTIE	ALRBIE	ALRAIE	TSE	WUTE	ALRBE	ALRAE	Res.	FMT	BYPSHAD	REFCKON	TSEDGE	WUCKSEL		
rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位 31:24	保留，始终保持复位值。
位 23	**COE:** 校准输出使能。 该位使能 RTC_CALIB 输出。 0: 校准输出禁用； 1: 校准输出启用。
位 22:21	**OSEL:** 输出选择。 该位用于选择 RTC_ALARM 输出关联的标志位。 00: 输出禁用 01: Alarm A 输出启用； 10: Alarm B 输出启用； 11: 保留
位 20	**POL:** 输出极性 (Output polarity) 该位用于配置 RTC_ALARM 输出的极性 0: 当 ALRAF/ALRBF/WUTF 置 1 时（取决于 OSEL[1:0]），该引脚为高电平 1: 当 ALRAF/ALRBF/WUTF 置 1 时（取决于 OSEL[1:0]），该引脚为低电平
位 19	**COSEL:** 校准输出选择 (Calibration output selection) 当 COE=1 时，该位可选择 RTC_CALIB 上输出的信号。 0: 校准输出为 512 Hz 1: 校准输出为 1 Hz 在 RTCCLK 为 32.768 kHz 且预分频器为其默认值（PREDIV_A=127 且 PREDIV_S=255）的条件下，这些频率有效。请参见第 27.3.15 节: 校准时钟输出。
位 18	**BKP:** 备份 (Backup) 用户可对此位执行写操作以记录是否已对夏令时进行更改。

位 17	**SUB1H:** 减少 1 小时（冬季时间更改）(Subtract 1 hour (winter time change)) 当该位在初始化模式以外的模式下置 1 时，如果当前小时不是 0，则日历时间将减少 1 小时。此位始终读为 0。 当前小时为 0 时，将此位置 1 没有任何作用。 0: 无任何作用 1: 将当前时间减少 1 小时。这可用于冬季时间更改。
位 16	**ADD1H:** 增加 1 小时（夏季时间更改）(Add 1 hour (summer time change)) 当该位在初始化模式以外的模式下置 1 时，日历时间将增加 1 小时。此位始终读为 0。 0: 无任何作用 1: 将当前时间增加 1 小时。这可用于夏季时间更改
位 15	**TSIE:** 时间戳中断使能 (Time-stamp interrupt enable) 0: 禁止时间戳中断 1: 使能时间戳中断
位 14	**WUTIE:** 唤醒定时器中断使能 (Wakeup timer interrupt enable) 0: 禁止唤醒定时器中断 1: 使能唤醒定时器中断
位 13	**ALRBIE:** 闹钟 B 中断使能 (Alarm B interrupt enable) 0: 禁止闹钟 B 中断 1: 使能闹钟 B 中断
位 12	**ALRAIE:** 闹钟 A 中断使能 (Alarm A interrupt enable) 0: 禁止闹钟 A 中断 1: 使能闹钟 A 中断
位 11	**TSE:** 时间戳使能 (timestamp enable) 0: 禁止时间戳 1: 使能时间戳
位 10	**WUTE:** 唤醒定时器使能 (Wakeup timer enable) 0: 禁止唤醒定时器 1: 使能唤醒定时器
位 9	**ALRBE:** 闹钟 B 使能 (Alarm B enable) 0: 禁止闹钟 B 1: 使能闹钟 B
位 8	**ALRAE:** 闹钟 A 使能 (Alarm A enable) 0: 禁止闹钟 A 1: 使能闹钟 A
位 7	保留，必须保持复位值。
位 6	**FMT:** 小时格式 (Hour format) 0: 24 小时/ 天格式 1: AM/PM 小时格式
位 5	**BYPSHAD:** 旁路影子寄存器 (Bypass the shadow registers) 0: 日历值（从 RTC_SSR、RTC_TR 和 RTC_DR 读取时）取自影子寄存器，该影子寄存器每两个 RTCCLK 周期更新一次。 1: 日历值（从 RTC_SSR、RTC_TR 和 RTC_DR 读取时）直接取自日历计数器。 注: 如果 APB1 时钟的频率低于 7 倍的 RTCCLK 频率，则必须将 BYPSHAD 置“1”。
位 4	**REFCKON:** RTC_REFIN 参考时钟检测使能（50 Hz 或 60 Hz）(RTC_REFIN reference clock detection enable (50 or 60 Hz)) 0: 禁止 RTC_REFIN 检测 1: 使能 RTC_REFIN 检测 注: PREDIV_S 必须为 0x00FF。
位 3	**TSEDGE:** 时间戳事件有效边沿 (Timestamp event active edge) 0: RTC_TS 输入上升沿生成时间戳事件 1: RTC_TS 输入下降沿生成时间戳事件 TSEDGE 发生更改时，必须复位 TSE 以避免将 TSF 意外置 1。

位 2:0	**WUCKSEL[2:0]:** 唤醒时钟选择 (Wakeup clock selection) 000: 选择 RTC/16 时钟 001: 选择 RTC/8 时钟 010: 选择 RTC/4 时钟 011: 选择 RTC/2 时钟 10x: 选择 ck_spre 时钟（通常为 1 Hz） 11x: 选择 ck_spre 时钟（通常为 1 Hz）并将 WUT 计数器值增加 2^{16}

23.4.4 RTC 初始化和状态寄存器 (RTC_ISR)

偏移地址： 0x08

复位值： 0x0000 0007

系统复位： 不受影响（INIT、INITF 和 RSF 位除外，它们在复位时被清零）

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RECALPF
															r
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TAMP2F	TAMP1F	TSOVF	TSF	WUTF	ALRBF	ALRAF	INIT	INITF	RSF	INITS	SHPF	WUTWF	ALABWF	ALRAWF
	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rw	r	rc_w0	r	r	r	r	r
0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1

位 31:17	保留，必须保持复位值
位 16	**RECALPF:** 重新校准挂起标志 (Recalibration pending Flag) 当软件对 RTC_CALR 寄存器执行写操作时，RECALPF 状态标志将自动置“1”，指示 RTC_CALR 寄存器已屏蔽。当采用新的校准设置时，该位恢复为“0”。
位 15	保留，必须保持复位时的值。
位 14	**TAMP2F:** RTC_TAMP2 检测标志 (RTC_TAMP2 detection flag) 在 RTC_TAMP2 输入上检测到入侵检测事件时，由硬件将此标志置 1。 该标志由软件写零清除
位 13	**TAMP1F:** RTC_TAMP1 检测标志 (RTC_TAMP1 detection flag) 在 RTC_TAMP1 输入上检测到入侵检测事件时，由硬件将此标志置 1。 该标志由软件写零清除
位 12	**TSOVF:** 时间戳溢出标志 (Time-stamp overflow flag) 当在 TSF 已置 1 的情况下发生时间戳事件时，由硬件将此标志置 1。 该标志由软件写零清除。建议仅在 TSF 位清零之后再检查并清零 TSOVF 位。否则，如果时间戳事件恰好在清零 TSF 位之前刚刚发生，则溢出事件可能会被漏掉。
位 11	**TSF:** 时间戳标志 (Time-stamp flag) 发生时间戳事件时，由硬件将此标志置 1。该标志由软件写零清除。
位 10	**WUTF:** 唤醒定时器标志 (Wakeup timer flag) 当唤醒自动重载计数器计数到 0 时，由硬件将此标志置 1。该标志由软件写零清除。 软件必须在 WUTF 再次置 1 的 1.5 个 RTCCLK 周期之前将该标志清零。
位 9	**ALRBF:** 闹钟 B 标志 (Alarm B flag) 当时间/ 日期寄存器（RTC_TR 和 RTC_DR）与闹钟 B 寄存器 (RTC_ALRMBR) 匹配时，由硬件将该标志置 1。 该标志由软件写零清除。
位 8	**ALRAF:** 闹钟 A 标志 (Alarm A flag) 当时间/ 日期寄存器（RTC_TR 和 RTC_DR）与闹钟 A 寄存器 (RTC_ALRMAR) 匹配时，由硬件将该标志置 1。 该标志由软件写零清除。

位 7	**INIT:** 初始化模式 (Initialization mode) 0: 自由运行模式 1: 初始化模式，用于编程时间和日期寄存器（RTC_TR 和 RTC_DR）以及预分频器寄存器 (RTC_PRER)。计数器停止计数，当 INIT 被复位后，计数器从新值开始计数。
位 6	**INITF:** 初始化标志 (Initialization flag) 当此位置 1 时，RTC 处于初始化状态，此时可更新事件、日期和预分频器寄存器。 0: 不允许更新日历寄存器 1: 允许更新日历寄存器
位 5	**RSF:** 寄存器同步标志 (Registers synchronization flag) 每次将日历寄存器的值复制到影子寄存器（RTC_SSRx、RTC_TRx 和 RTC_DRx）时，都会由硬件将此位置 1。在初始化模式下、平移操作挂起时 (SHPF=1) 或在旁路影子寄存器模式 (BYPSHAD=1) 下，该位由硬件清零。该位还可由软件清零。 在初始化模式下，该位可由软件或硬件清零。 0: 日历影子寄存器尚未同步 1: 日历影子寄存器已同步
位 4	**INITS:** 初始化状态标志 (Initialization status flag) 当日历年份字段不为 0 时（RTC 域复位状态），由硬件将该位置 1。 0: 日历尚未初始化 1: 日历已经初始化
位 3	**SHPF:** 平移操作挂起 (Shift operation pending) 0: 没有平移操作挂起 1: 某个平移操作挂起 只要通过对 RTC_SHIFTR 寄存器执行写操作来启动平移操作，此标志便由硬件置 1。执行完相应的平移操作后，此标志由硬件清零。对 SHPF 位执行写入操作不起作用。
位 2	**WUTWF:** 唤醒定时器写标志 (Wakeup timer write flag) 此位在 RTC_CR 中的 WUTE 位清零 2 个 RTCCLK 周期后由硬件置 1，在 WUTE 位置 1 后 2 个 RTCCLK 周期后清零。当 WUTE 位清零且 WUTWF 位置 1 时，可更改唤醒定时器的值。 0: 不允许更新唤醒定时器配置 1: 允许更新唤醒定时器配置
位 1	**ALRBWF:** 闹钟 B 写标志 (Alarm B write flag) 在 RTC_CR 寄存器中的 ALRBE 位置 0 之后，当闹钟 B 的值可更改时，由硬件将该位置 1。该位在初始化模式下由硬件清零。 0: 不允许更新闹钟 B 1: 允许更新闹钟 B
位 0	**ALRAWF:** 闹钟 A 写标志 (Alarm A write flag) 在 RTC_CR 寄存器中的 ALRAE 位置 0 后，当闹钟 A 的值可更改时，由硬件将该位置 1。该位在初始化模式下由硬件清零。 0: 不允许更新闹钟 A 1: 允许更新闹钟 A

23.4.5 RTC 预分频器寄存器 (RTC_PRER)

偏移地址： 0x10

复位值： 0x007F 00FF

系统复位： 不受影响

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREDIV_A						
									rw	rw	rw	rw	rw	rw	rw
0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1

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

位 31:23	保留，必须保持复位值
位 22:16	**PREDIV_A[6:0]**: 异步预分频系数 (Asynchronous prescaler factor) 下面是异步分频系数的公式: ck_apre 频率 = RTCCLK 频率/(PREDIV_A+1)
位 15	保留，必须保持复位值。
位 14:0	**PREDIV_S[14:0]**: 同步预分频系数 (Synchronous prescaler factor) 下面是同步分频系数的公式: ck_spre 频率 = ck_apre 频率/(PREDIV_S+1)

23.4.6 RTC 唤醒定时器寄存器 (RTC_WUTR)

仅当 RTC_ISR 中的 WUTWF 置 1 时才可对该寄存器执行写操作。

偏移地址： 0x14

复位值： 0x0000 FFFF

系统复位： 不受影响

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

位 31:16	保留，必须保持复位值
位 15:0	**WUT[15:0]**: 唤醒自动重载值位 (Wakeup auto-reload value bits) 当使能唤醒定时器时（WUTE 置 1），每 (WUT[15:0] + 1) 个 ck_wut 周期将 WUTF 标志置 1 一次。ck_wut 周期通过 RTC_CR 寄存器的 WUCKSEL[2:0] 位进行选择。 当 WUCKSEL[2] = 1 时，唤醒定时器变为 17 位，WUCKSEL[1] 等效为 WUT[16]，即要重载到定时器的最高有效位。 WUTF 第一次置 1 发生在 WUTE 置 1 之后 (WUT+1) 个 ck_wut 周期。禁止在 WUCKSEL[2:0]=011(RTCCLK/2)时将 WUT[15:0] 设置为 0x0000。

23.4.7 RTC 闹钟 A 寄存器 (RTC_ALRMAR)

仅当 RTC_ISR 中的 ALRAWF 置 1 时或在初始化模式下，才可以对该寄存器执行写操作。

偏移地址： 0x1C

复位值： 0x0000 0000

系统复位： 不受影响

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MSK4	WDSEL	DT		DU				MSK3	PM	HT		HU			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MSK2	MNT			MNU				MSK1	ST			SU			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位 31	**MSK4:** 闹钟 A 日期掩码 (Alarm A date mask) 0: 如果日期/ 日匹配，则闹钟 A 置 1 1: 在闹钟 A 比较中，日期/ 日无关
位 30	**WDSEL:** 星期几选择 (Week day selection) 0: DU[3:0] 代表日期的个位 1: DU[3:0] 代表星期几 DT[1:0] 为无关位。
位 29:28	**DT[1:0]:** 日期的十位（BCD 格式）(Date tens in BCD format)。
位 27:24	**DU[3:0]:** 日期的个位或日（BCD 格式）(Date units or day in BCD format)。
位 23	**MSK3:** 闹钟 A 小时掩码 (Alarm A hours mask) 0: 如果小时匹配，则闹钟 A 置 1 1: 在闹钟 A 比较中，小时无关
位 22	**PM:** AM/PM 符号 (AM/PM notation) 0: AM 或 24 小时制 1: PM
位 21:20	**HT[1:0]:** 小时的十位（BCD 格式）(Hour tens in BCD format)。
位 19:16	**HU[3:0]:** 小时的个位（BCD 格式）(Hour units in BCD format)。
位 15	MSK2: 闹钟 A 分钟掩码 (Alarm A minutes mask) 0: 如果分钟匹配，则闹钟 A 置 1 1: 在闹钟 A 比较中，分钟无关
位 14:12	**MNT[2:0]:** 分钟的十位（BCD 格式）(Minute tens in BCD format)。
位 11:8	**MNU[3:0]:** 分钟的个位（BCD 格式）(Minute units in BCD format)。
位 7	**MSK1:** 闹钟 A 秒掩码 (Alarm A seconds mask) 0: 如果秒匹配，则闹钟 A 置 1 1: 在闹钟 A 比较中，秒无关
位 6:4	**ST[2:0]:** 秒的十位（BCD 格式）(Second tens in BCD format)。
位 3:0	**SU[3:0]:** 秒的个位（BCD 格式）(Second units in BCD format)。

23.4.8 RTC 闹钟 B 寄存器 (RTC_ALRMBR)

仅当 RTC_ISR 中的 ALRBWF 置 1 时或在初始化模式下，才可以对该寄存器执行写操作。

偏移地址： 0x20

复位值： 0x0000 0000

系统复位： 不受影响

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MSK4	WDSEL	DT		DU				MSK3	PM	HT		HU			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MSK2	MNT			MNU				MSK1	ST			SU			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位 31	**MSK4:** 闹钟 B 日期掩码 (Alarm B date mask) 0: 如果日期和日匹配，则闹钟 B 置 1 1: 在闹钟 B 比较中，日期和日无关
位 30	**WDSEL:** 星期几选择 (Week day selection) 0: DU[3:0] 代表日期的个位 1: DU[3:0] 代表星期几 DT[1:0] 为无关位。
位 29:28	**DT[1:0]:** 日期的十位（BCD 格式）(Date tens in BCD format)
位 27:24	**DU[3:0]:** 日期个位或日（BCD 格式）(Date units or day in BCD format)

位 23	**MSK3:** 闹钟 B 小时掩码 (Alarm B hours mask) 0: 如果小时匹配，则闹钟 B 置 1 1: 在闹钟 B 比较中，小时无关
位 22	**PM:** AM/PM 符号 (AM/PM notation) 0: AM 或 24 小时制 1: PM
位 21:20	**HT[1:0]:** 小时的十位（BCD 格式）(Hour tens in BCD format)
位 19:16	**HU[3:0]:** 小时的个位（BCD 格式）(Hour units in BCD format)
位 15	**MSK2:** 闹钟 B 分钟掩码 (Alarm B minutes mask) 0: 如果分钟匹配，则闹钟 B 置 1 1: 在闹钟 B 比较中，分钟无关
位 14:12	**MNT[2:0]:** 分钟的十位（BCD 格式）(Minute tens in BCD format)
位 11:8	**MNU[3:0]:** 分钟的个位（BCD 格式）(Minute units in BCD format)
位 7	**MSK1:** 闹钟 B 秒掩码 (Alarm B seconds mask) 0: 如果秒匹配，则闹钟 B 置 1 1: 在闹钟 B 比较中，秒无关
位 6:4	**ST[2:0]:** 秒的十位（BCD 格式）(Second tens in BCD format)
位 3:0	**SU[3:0]:** 秒的个位（BCD 格式）(Second units in BCD format)

23.4.9 RTC 写保护寄存器 (RTC_WPR)

偏移地址： 0x24

复位值： 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

位 31:8	保留，必须保持复位值。
位 7:0	**KEY:** 写保护关键字 (Write protection key) 可通过软件对该字节执行写操作。读取该字节时，始终返回 0x00。 有关如何解锁 RTC 寄存器写保护的介绍。

23.4.10 RTC 亚秒寄存器 (RTC_SSR)

偏移地址： 0x28

复位值： 0x0000 0000

系统复位： 当 BYPSHAD = 0 时为 0x0000 0000。当 BYPSHAD = 1 时不受影响。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位 31:16	保留，必须保持复位值。
位 15:0	**SS:** 亚秒值 (Sub seconds value) SS[15:0] 是同步预分频器计数器的值。此亚秒值可根据以下公式得出：亚秒值 = (PREDIV_S - SS) / (PREDIV_S + 1) 注：仅当执行平移操作之后，SS 才能大于 PREDIV_S。在这种情况下，正确的时间/日期比 RTC_TR/RTC_DR 所指示的时间/日期慢一秒钟。

23.4.11 RTC 平移控制寄存器 (RTC_SHIFTR)

偏移地址： 0x2C

复位值： 0x0000 0000

系统复位： 不受影响

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADD1S	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
w															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	SUBFS														
	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位 31	**ADD1S：**增加一秒钟 (Add one second) 0：无任何作用 1：对时钟/ 日历增加一秒钟 此位为只写位且始终读为 0。当平移操作挂起（RTC_ISR 中的 SHPF=1）时，对该位执行写操作无作用。 此函数应与 SUBFS 配合使用（请参见下文介绍），以便有效地向原子操作机制的时钟添加亚秒值。
位 30:15	保留，必须保持复位值。
位 14:0	**SUBFS：**减少亚秒值 (Subtract a fraction of a second) 此位为只写位且始终读为 0。当平移操作挂起（RTC_ISR 中的 SHPF=1）时，对该位执行写操作无作用。 写入 SUBFS 的值将加到同步预分频器计数器中。由于该计数器递减计数，此操作可有效地从时钟减去（延迟）以下时间： 延迟（秒）= SUBFS / (PREDIV_S + 1) 当 ADD1S 函数与 SUBFS 结合使用时，可有效地将亚秒值增加到时钟（提前时钟），使时钟提前以下时间： 提前（秒）= (1 - (SUBFS / (PREDIV_S + 1)))。 注：对 SUBFS 执行写操作将使 RSF 清零。软件随后会等待至 RSF=1 以确定影子寄存器已更新为平移后的时间。

23.4.12 RTC 时间戳时间寄存器 (RTC_TSTR)

仅当 RTC_ISR 中的 TSF 置 1 时，该寄存器的内容才有效。当 TSF 位复位时，清零该寄存器。

偏移地址： 0x30

复位值： 0x0000 0000

系统复位： 不受影响

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PM	HT		HU			
									r	r	r	r	r	r	r
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	MNT			MNU				Res.	ST			SU			
	r	r	r	r	r	r	r		r	r	r	r	r	r	r
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位 31:23	保留，必须保持复位值。
位 22	**PM**：AM/PM 符号 (AM/PM notation) 0：AM 或 24 小时制 1：PM
位 21:20	**HT[1:0]**：小时的十位（BCD 格式）(Hour tens in BCD format)
位 19:16	**HU[3:0]**：小时的个位（BCD 格式）(Hour units in BCD format)
位 15	保留，必须保持复位值
位 14:12	**MNT[2:0]**：分钟的十位（BCD 格式）(Minute tens in BCD format)
位 11:8	**MNU[3:0]**：分钟的个位（BCD 格式）(Minute units in BCD format)
位 7	保留，必须保持复位值。
位 6:4	**ST[2:0]**：秒的十位（BCD 格式）(Second tens in BCD format)
位 3:0	**SU[3:0]**：秒的个位（BCD 格式）(Second units in BCD format)

23.4.13 RTC 时间戳日期寄存器 (RTC_TSDR)

仅当 RTC_ISR 中的 TSF 置 1 时，该寄存器的内容才有效。当 TSF 位复位时，清零该寄存器。

偏移地址： 0x34

复位值： 0x0000 0000

系统复位： 不受影响

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WDU			MT	MU				Res.	Res.	DT		DU			
r	r	r	r	r	r	r	r			r	r	r	r	r	r
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位 31:16	保留，必须保持复位值。
位 15:13	**WDU[1:0]**：星期几的个位 (Week day units)
位 12	**MT**：月份的十位（BCD 格式）(Month tens in BCD format)
位 11:8	**MU[3:0]**：月份的个位（BCD 格式）(Month units in BCD format)
位 7:6	保留，必须保持复位值。
位 5:4	**DT[1:0]**：日期的十位（BCD 格式）(Date tens in BCD format)
位 3:0	**DU[3:0]**：日期的个位（BCD 格式）(Date units in BCD format)

23.4.14 RTC 时间戳亚秒寄存器 (RTC_TSSSR)

仅当 RTC_ISR/TSF 置 1 时，该寄存器的内容才有效。当 RTC_ISR/TSF 位复位时，清零该寄存器。

偏移地址： 0x38

复位值： 0x0000 0000

系统复位： 不受影响

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

位 31:16	保留，必须保持复位值
位 15:0	**SS：**亚秒值 (Sub seconds value) 当发生时间戳事件时，SS[15:0] 是同步预分频器计数器的值。

23.4.15 RTC 校准寄存器 (RTC_CALR)

偏移地址： 0x3C

复位值： 0x0000 0000

系统复位： 不受影响

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CALP	CALW8	CALW16	Res.	Res.	Res.	Res.	CALM								
rw	rw	rw					rw	rw	rw	rw	rw	rw	rw	rw	rw
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位 31:16	保留，必须保持复位值
位 15	**CALP：**将 RTC 的频率增加 488.5 ppm (Increase frequency of RTC by 488.5 ppm) 0：不增加 RTCCLK 脉冲。 1：每 211 个脉冲有效插入一个 RTCCLK 脉冲（将频率增加 488.5ppm）。 此功能应与 CALM 结合使用，后者在高分辨率下会降低日历的频率。如果输入频率为 32768Hz，则在 32 秒窗口中增加的 RTCCLK 脉冲数按如下公式计算：(512*CALP)-CALM。
位 14	**CALW8：**使用 8 秒校准周期 (Use an 8-second calibration cycle period) 当 CALW8 置“1”时，选择 8 秒校准周期。 注：CALW8=“1”时，CALM[1:0] 将始终保持为“00”。
位 13	**CALW16：**使用 16 秒校准周期 (Use a 16-second calibration cycle period) 当 CALW16 置“1”时，选择 16 秒校准周期。如果 CALW8 = 1，则不能将该位置“1”。 注：当 CALW16=“1”时，CALM[0] 将始终保持为“0”。
位 12:9	保留，必须保持复位值

位 8:0	**CALM[8:0]：**负校准 (Calibration minus) 在 220 个 RTCCLK 脉冲内屏蔽 CALM 个脉冲（如果输入频率为 32768 Hz，则为 32 秒）来降低日历的频率。其分辨率为 0.9537 ppm。 要提高日历的频率，则应将此功能与 CALP 结合使用。

23.4.16 RTC 入侵和复用功能配置寄存器 (RTC_TAFCR)

偏移地址： 0x40

复位值： 0x0000 0000

系统复位： 不受影响

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PC15 MODE	PC15 VALUE	PC14 MODE	PC14 VALUE	PC13 MODE	PC13 VALUE	Res.	Res.
								rw	rw	rw	rw	rw	rw		
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TAMP PUDIS	TAMPPRCH		TAMPFLT		TAMPFREQ			TAMP TS	Res.	Res.	TAMP 2TRG	TAMP 2E	TAMPIE	TAMP 1TRG	TAMP 1E
rw	rw	rw	rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位 31:24	保留，必须保持复位值。
位 23	**PC15MODE：**PC15 模式 (PC15 mode) 0：PC15 由 GPIO 配置寄存器控制，因此 PC15 在待机模式下浮空。 1：LSE 禁用时，PC15 强制为推挽输出模式。
位 22	**PC15VALUE：**PC15 值 (PC15 value) 如果 LSE 禁用且 PC15MODE = 1, PC15VALUE 设置 PC15 的输出值。
位 21	**PC14MODE：**PC14 模式 (PC14 mode) 0: PC14 由 GPIO 配置寄存器控制，因此 PC14 在待机模式下浮空。 1: LSE 禁用时，PC14 强制为推挽输出模式
位 20	**PC14VALUE：**PC14 值 (PC14 value) 0：入侵 2 事件擦除备份寄存器。 1：入侵 2 事件不擦除备份寄存器。
位 19	**PC13MODE：**PC13 模式 (PC13 mode) 0: PC13 受 GPIO 配置寄存器约束，因此 PC13 在待机模式下浮空。 1: RTC 复用功能禁用时，PC13 强制为推挽输出模式。
位 18	**PC13VALUE：**RTC_ALARM 输出形式/PC13 值 (RTC_ALARM output type/PC13 value) 如果 PC13 用于输出 RTC_ALARM, PC13VALUE 设置输出： 0: RTC_ALARM 为开漏输出； 1: RTC_ALARM 为推挽输出 如果所有 RTC 复用功能禁用且 PC13MODE = 1, PC13VALUE 设置 PC13 输出值。
位 17:16	保留，必须保持复位时的值。
位 15	**TAMPPUDIS：**RTC_TAMPx 上拉禁止 (RTC_TAMPx pull-up disable) 该位决定在每次采样之前是否对每个 RTC_TAMPx 引脚都进行预充电。 0：采样之前对 RTC_TAMPx 引脚进行预充电（使能内部上拉） 1：禁止对 RTC_TAMPx 引脚进行预充电。

位 14:13	**TAMPPRCH[1:0]**：RTC_TAMPx 预充电持续时间 (RTC_TAMPx precharge duration) 这些位决定了在每次采样之前激活上拉的持续时间。TAMPPRCH 对每个 RTC_TAMPx 输入都有效。 0x0：1 个 RTCCLK 周期 0x1：2 个 RTCCLK 周期 0x2：4 个 RTCCLK 周期 0x3：8 个 RTCCLK 周期
位 12:11	**TAMPFLT[1:0]**：RTC_TAMPx 过滤器计数 (RTC_TAMPx filter count) 这些位决定了为激活入侵事件所需的指定电平 (TAMP*TRG) 上的连续采样次数。TAMPFLT 对每个 RTC_TAMPx 输入都有效。 0x0：在 RTC_TAMPx 输入转变为有效电平的边沿激活入侵事件（RTC_TAMPx 输入上无内部上拉）。 0x1：在有效电平上连续执行 2 次采样后激活入侵事件。 0x2：在有效电平上连续执行 4 次采样后激活入侵事件。 0x3：在有效电平上连续执行 8 次采样后激活入侵事件。
位 10:8	**TAMPFREQ[2:0]**：入侵采样频率 (Tamper sampling frequency) 决定对每个 RTC_TAMPx 输入进行采样时的频率。 0x0：RTCCLK/32768（RTCCLK = 32768 Hz 时为 1 Hz） 0x1：RTCCLK/16384（RTCCLK = 32768 Hz 时为 2 Hz） 0x2：RTCCLK/8192（RTCCLK = 32768 Hz 时为 4 Hz） 0x3：RTCCLK/4096（RTCCLK = 32768 Hz 时为 8 Hz） 0x4：RTCCLK/2048（RTCCLK = 32768 Hz 时为 16 Hz） 0x5：RTCCLK/1024（RTCCLK = 32768 Hz 时为 32 Hz） 0x6：RTCCLK/512（RTCCLK = 32768 Hz 时为 64 Hz） 0x7：RTCCLK/256（RTCCLK = 32768 Hz 时为 128 Hz）
位 7	**TAMPTS**：发生入侵检测事件时激活时间戳 (Activate timestamp on tamper detection event) 0：发生入侵检测事件时不保存时间戳 1：发生入侵检测事件时保存时间戳 即便 RTC_CR 寄存器中的 TSE=0，TAMPTS 仍有效。
位 6	**TAMP3TRG**：RTC_TAMP3 输入的有效电平 (Active level for RTC_TAMP3 input) 如果 TAMPFLT ≠ 00： 0：RTC_TAMP3 输入保持低电平会触发入侵检测事件。 1：RTC_TAMP3 输入保持高电平会触发入侵检测事件。如果 TAMPFLT = 00： 0：RTC_TAMP3 输入上升沿会触发入侵检测事件。 1：RTC_TAMP3 输入下降沿会触发入侵检测事件。
位 5	**TAMP3E**：RTC_TAMP3 检测使能 (RTC_TAMP3 detection enable) 0：禁止 RTC_TAMP3 输入检测 1：使能 RTC_TAMP3 输入检测
位 4	**TAMP2TRG**：RTC_TAMP2 输入的有效电平 (Active level for RTC_TAMP2 input) 如果 TAMPFLT != 00： 0：RTC_TAMP2 输入保持低电平会触发入侵检测事件。 1：RTC_TAMP2 输入保持高电平会触发入侵检测事件。如果 TAMPFLT = 00： 0：RTC_TAMP2 输入上升沿会触发入侵检测事件。 1：RTC_TAMP2 输入下降沿会触发入侵检测事件。
位 3	**TAMP2E**：RTC_TAMP2 输入检测使能 (RTC_TAMP2 input detection enable) 0：禁止 RTC_TAMP2 检测 1：使能 RTC_TAMP2 检测
位 2	**TAMPIE**：入侵中断使能 (Tamper interrupt enable) 0：禁止入侵中断 1：使能入侵中断 注：该位使能所有入侵引脚事件的中断，无论 TAMPxIE 电平如何。如果将该位清零，可以通过将 TAMPxIE 置 1 分别使能每个入侵事件中断。

位 1	**TAMP1TRG**：RTC_TAMP1 输入的有效电平 (Active level for RTC_TAMP1 input) 如果 TAMPFLT != 00 0：RTC_TAMP1 输入保持低电平会触发入侵检测事件。 1：RTC_TAMP1 输入保持高电平会触发入侵检测事件。如果 TAMPFLT = 00： 0：RTC_TAMP1 输入上升沿会触发入侵检测事件。 1：RTC_TAMP1 输入下降沿会触发入侵检测事件。
位 0	**TAMP1E**：RTC_TAMP1 输入检测使能 (RTC_TAMP1 input detection enable) 0：禁止 RTC_TAMP1 检测 1：使能 RTC_TAMP1 检测

23.4.17 RTC 闹钟 A 亚秒寄存器 (RTC_ALRMASSR)

仅当 RTC_CR 中的 ALRAE 复位时或在初始化模式下，才可以对该寄存器执行写操作。

偏移地址： 0x44

复位值： 0x0000 0000

系统复位： 不受影响

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	MASKSS				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
				rw	rw	rw	rw								
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

位 31:28	保留，必须保持复位值。
位 27:24	**MASKSS[3:0]**：屏蔽从此位开始的最高有效位 (Mask the most-significant bits starting at this bit) 0：不对闹钟 A 的亚秒进行比较。当秒单元递增时设置闹钟（假定其余字段均匹配）。 1：在闹钟 A 比较中，SS[14:1] 为无关位。仅比较 SS[0]。 2：在闹钟 A 比较中，SS[14:2] 为无关位。仅比较 SS[1:0]。 3：在闹钟 A 比较中，SS[14:3] 为无关位。仅比较 SS[2:0]。 ... 12：在闹钟 A 比较中，SS[14:12] 为无关位。比较 SS[11:0]。 13：在闹钟 A 比较中，SS[14:13] 为无关位。比较 SS[12:0]。 14：在闹钟 A 比较中，SS[14] 为无关位。比较 SS[13:0]。 15：所有 15 个 SS 位均进行比较，并且必须全部匹配才能激活闹钟。 同步计数器的溢出位（位 15）从不进行比较。仅当执行平移操作之后，此位才不为 0。
位 23:15	保留，必须保持复位值。
位 14:0	**SS[14:0]**：亚秒值 (Sub seconds value) 该值与同步预分频器计数器的内容进行比较以确定是否要激活闹钟 A。仅比较位 0 到 MASKSS-1。

23.4.18 RTC 闹钟 B 亚秒寄存器 (RTC_ALRMBSSR)

仅当 RTC_CR 中的 ALRBE 复位时或在初始化模式下，才可以对该寄存器执行写操作。

偏移地址： 0x48

复位值： 0x0000 0000

系统复位： 不受影响

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

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

位 31:28	保留，必须保持复位值。
位 27:24	**MASKSS[3:0]：**屏蔽从此位开始的最高有效位 (Mask the most-significant bits starting at this bit) 0x0：不对闹钟 B 的亚秒进行比较。当秒单元递增时设置闹钟（假定其余字段均匹配）。 0x1：在闹钟 B 比较中，SS[14:1] 为无关位。仅比较 SS[0]。 0x2：在闹钟 B 比较中，SS[14:2] 为无关位。仅比较 SS[1:0]。 0x3：在闹钟 B 比较中，SS[14:3] 为无关位。仅比较 SS[2:0]。 ... 0xC：在闹钟 B 比较中，SS[14:12] 为无关位。比较 SS[11:0]。 0xD：在闹钟 B 比较中，SS[14:13] 为无关位。比较 SS[12:0]。 0xE：在闹钟 B 比较中，SS[14] 为无关位。比较 SS[13:0]。 0xF：所有 15 个 SS 位均进行比较，并且必须全部匹配才能激活闹钟。 同步计数器的溢出位（位 15）从不进行比较。仅当执行平移操作之后，此位才不为 0。
位 23:15	保留，必须保持复位值。
位 14:0	**SS[14:0]：**亚秒值 (Sub seconds value) 该值与同步预分频器计数器的内容进行比较以确定是否要激活闹钟 B。仅比较位 0 到 MASKSS-1。

23.4.19 备份寄存器 (RTC_BKPxR)

偏移地址： 0x50 到 0x60

复位值： 0x0000 0000

系统复位： 不受影响

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

位 31:0	**BKP[31:0]** 应用可向/从这些寄存器写入/读取数据。 当器件以低功耗模式工作时，这些寄存器的内容保持有效。 发生入侵检测事件时该寄存器会被复位，并且只要 TAMPxF=1，该寄存器就一直保持复位。或者在禁止闪存读出保护时复位。

24 内部集成电路 (I2C) 接口

24.1 简介

I2C（内部集成电路）总线接口处理微控制器与串行 I2C 总线间的通信。它提供多主模式功能，可以控制所有 I2C 总线特定的序列、协议、仲裁和时序。它支持标准模式 (Sm)、快速模式 (Fm) 和超快速模式 (Fm+)。

它还与 SMBus（系统管理总线）和 PMBus（电源管理总线）兼容。

可使用 DMA 来减轻 CPU 的工作量。

24.2 I2C 主要特性

- 兼容 I2C 总线规范第 03 版：
  - 从模式和主模式
  - 多主模式功能
  - 标准速度模式（高达 100 kHz）
  - 快速模式（高达 400 kHz）
  - 超快速模式（高达 1 MHz）
  - 7 位和 10 位寻址模式
  - 多个 7 位从地址（2 个从设备地址寄存器， 1 个具有可配置的掩码位段）
  - 所有 7 位地址应答模式
  - 广播呼叫
  - 总线上的数据建立和保持时间可软件配置
  - 方便易用的事件管理
  - 可选的时钟延长
  - 软件复位
- 带 DMA 功能的 1 字节缓冲
- 可编程模拟和数字噪声滤波器

  还可额外提供以下特性，具体取决于产品实现（请参见第 *28.3* 节： *I2C* 特性实现）：
- 兼容 SMBus 规范第 2.0 版：
  - 具有 ACK 控制的硬件 PEC（数据包错误校验）生成和验证
  - 命令和数据应答控制
  - 支持地址解析协议 (ARP)
  - 支持主机和从设备
  - SMBus 报警
  - 超时和空闲条件检测
- 兼容 PMBus 第 1.1 版标准
- 独立时钟：选择独立时钟源可使 I2C 通信速度不受 PCLK 时钟频率更改的影响
- 地址匹配时从停止模式唤醒

24.3 I2C 功能说明

除了接收和发送数据之外，此接口还可以从串行格式转换为并行格式，反之亦然。中断由软件使能或禁止。该接口通过数据引脚 (SDA) 和时钟引脚 (SCL) 连接到 I2C 总线。它可以连接到标准速度（高达 100 kHz）、快速（高达 400 kHz）或超快速（高达 1 MHz） I2C 总线。

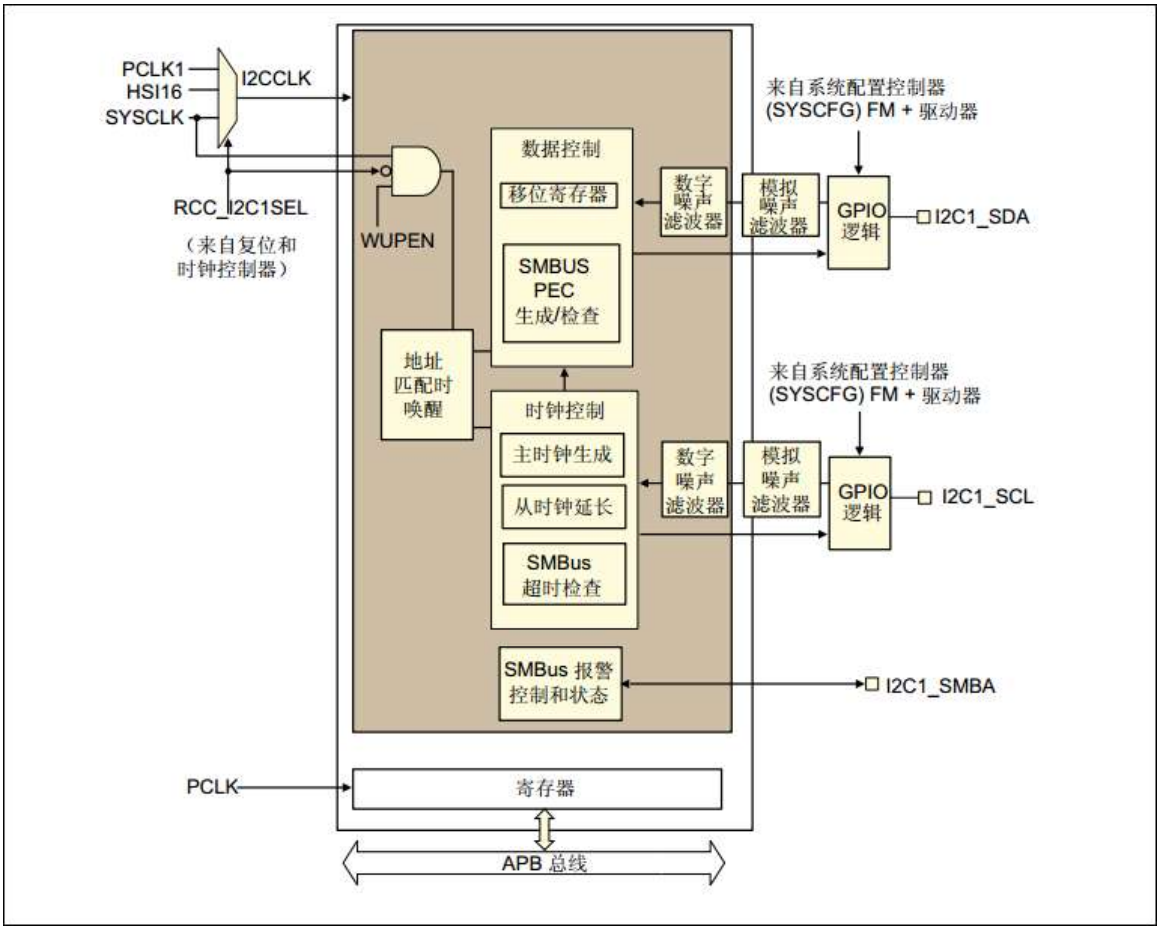
该接口也可通过数据引脚 (SDA) 和时钟引脚 (SCL) 连接到 SMBus。

如果支持 SMBus 功能：还可使用额外的可选 SMBus 报警引脚 (SMBA)。

24.3.1 I2C1 框图

I2C1 接口的框图如图 24-1 所示。

图 24-1 **I2C1/2** 框图



I2C1/2 的时钟由独立时钟源提供，这使得 I2C 能够独立于 PCLK 频率工作。

该独立时钟源可从以下三种时钟源中任选其一：

- PCLK1： APB1 时钟（默认值）
- HSI16：内部 16 MHz RC 振荡器
- SYSCLK：系统时钟

更多详细信息，请参见复位和时钟控制 *(RCC)*。

I2C1/2 I/O 支持 20 mA 输出电流驱动器，用于驱动超快速操作。将 SCL 和 SDA 的驱动能力控制位置 1 可使能该功能，这些位位于 *SYSCFG* 外设模式配置寄存器 *(SYSCFG_CFGR2)*。

24.3.2 I2C 时钟要求

I2C 内核的时钟由 I2CCLK 提供。

I2CCLK 周期 t_{I2CCLK} 必须遵循以下条件：

tI2CCLK < (tLOW - tfilters) / 4 且 tI2CCLK < tHIGH

其中：

t_{LOW}： SCL 低电平时间； t_{HIGH}： SCL 高电平时间

tfilters： 滤波器使能时，该值为模拟滤波器和数字滤波器引入的延时总和。

模拟滤波器延时最大值为 260 ns。数字滤波器延时为 DNF x t_{I2CCLK}。

PCLK 时钟周期 t_{PCLK} 必须遵循以下条件：

tPCLK < 4/3 tSCL

其中， t_{SCL}： SCL 周期

注意： 当 I2C 内核的时钟由 PCLK 提供时， PCLK 必须遵循 t_{I2CCLK} 的条件。

24.3.3 模式选择

该接口在工作时可选用以下四种模式之一：

- 从发送器
- 从接收器
- 主发送器
- 主接收器

默认情况下，它以从模式工作。接口在生成起始位后会自动由从模式切换为主模式，并在出现仲裁丢失或生成停止位时从主模式切换为从模式，从而实现多主模式功能。

通信流程

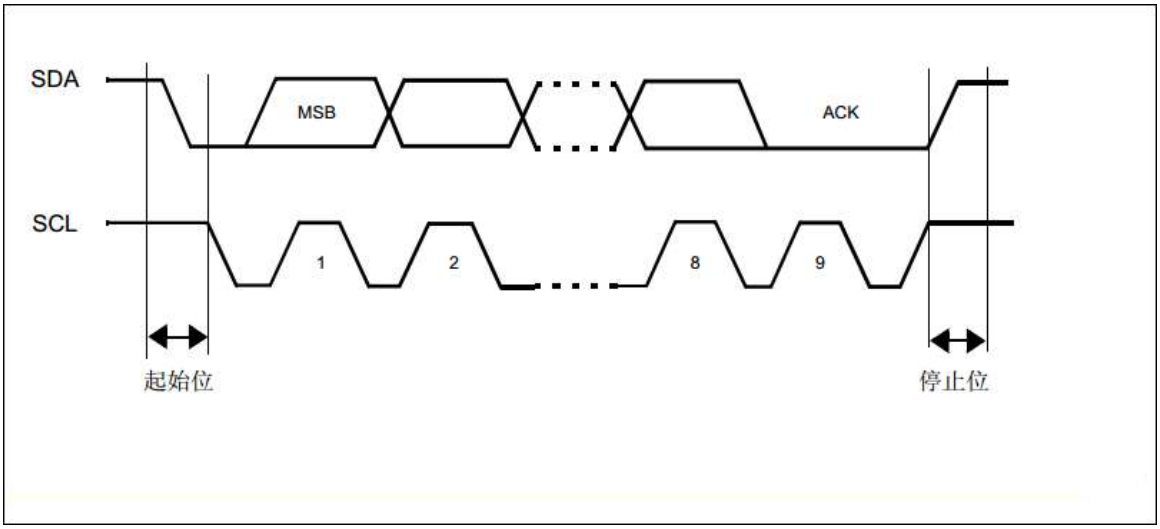
在主模式下， I2C 接口会启动数据传输并生成时钟信号。串行数据传输始终是在出现起始位时开始，在出现停止位时结束。起始位和停止位均在主模式下由软件生成。

在从模式下，该接口能够识别其自身地址（7 或 10 位）以及广播呼叫地址。广播呼叫地址检测可由软件使能或禁止。保留的 SMBus 地址也可由软件使能。

数据和地址均以 8 位字节传输， MSB 在前。起始位后紧随地址字节（7 位地址占据一个字节；10 位地址占据两个字节）。地址始终在主模式下传送。

在字节传输 8 个时钟周期后是第 9 个时钟脉冲，在此期间接收器必须向发送器发送一个应答位。请参见下图。

图 24-2 I2C 总线协议



应答位可由软件使能或禁止。 I2C 接口地址可通过软件进行选择。

24.3.4 I2C 初始化

使能和禁止外设

I2C 外设时钟必须在时钟控制器中进行配置和使能（请参见复位和时钟控制 *(RCC)*）。然后可通过将 I2C_CR1 寄存器中的 PE 位置 1 使能 I2C。

当禁止 I2C (PE=0) 时， I2C 将执行软件复位。更多详细信息，请参见软件复位。

噪声滤波器

通过将 I2C_CR1 寄存器中的 PE 位置 1 来使能 I2C 外设之前，如有必要，用户必须配置噪声滤波器。默认情况下， SDA 和 SCL 输入上集成了模拟噪声滤波器。该模拟滤波器符合 I2C 规范，此规范要求在快速模式和超快速模式下对脉宽在 50ns 以下的脉冲都要抑制。用户可通过将 ANFOFF 位置 1 来禁止该模拟滤波器，和/ 或通过配置 I2C_CR1 寄存器中的 DNF[3:0] 位来选择数字滤波器。

使能数字滤波器时， SCL 或 SDA 线的电平只有在电平稳定时间超过 DNF x I2CCLK 个周期后才会发生内部变化，从而可抑制的尖峰脉宽在 1 到 15 个 I2CCLK 周期可编程。

表 24-1 模拟滤波器与数字滤波器对比

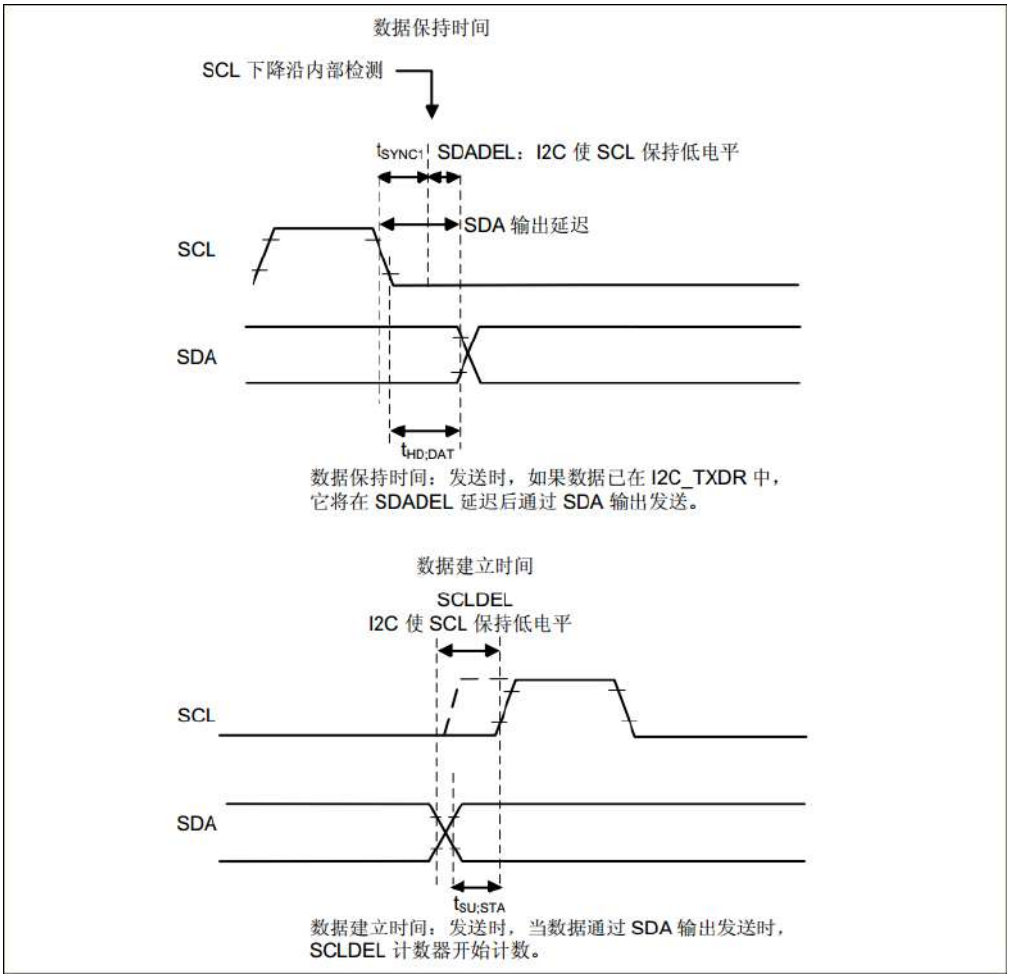
	模拟滤波器	数字滤波器
抑制尖峰的 脉冲宽度	≥ 50 ns	长度可编程为 1 到 15 个 I2C 外设时钟
优点	停止模式下仍可用	– 长度可编程：额外的滤波能力与标准要求 – 稳定长度
缺点	随温度、电压和 过程变化会发生变化	当使能数字滤波器后，无法在地址匹配时从停止模 式唤醒。

注意： 使能 I2C 时，不允许更改滤波器配置。

I2C 时序

必须配置时序，以便保证主模式和从模式下使用正确的数据保持和建立时间。配置方法是编程 I2C_TIMINGR 寄存器中的 PRESC[3:0]、 SCLDEL[3:0] 和 SDADEL[3:0] 位。

图 24-3 建立和保持时序



- 当内部检测到 SCL 下降沿时，会在发送 SDA 输出之前插入一段延时。该延时为 tSDADEL = SDADEL x tPRESC + tI2CCLK，其中 tPRESC = (PRESC+1) x tI2CCLK。TSDADEL 会影响保持时间 tHD;DAT。

  SDA 总输出延时为：

  tSYNC1 + {[SDADEL x (PRESC+1) + 1] x tI2CCLK }

  tSYNC1 持续时间取决于以下参数：

  - SCL 下降斜率
  - 模拟滤波器（使能时）引入的输入延时： $t_{AF(min)} < t_{AF} < t_{AF(max)}$ ns
  - 数字滤波器（使能时）引入的输入延时： $t_{DNF} = DNF \times t_{I2CCLK}$
  - SCL 与 I2CCLK 时钟建立同步而产生的延时（2 到 3 个 I2CCLK 周期）

为了桥接 SCL 下降沿的未定义区域，用户编程 SDADEL 时必须遵循以下条件：

$\{t_f\ (max) + t_{HD;DAT}\ (min) - t_{AF(min)} - [(DNF + 3) \times t_{I2CCLK}]\} / \{(PRESC + 1) \times t_{I2CCLK}\} \le SDADEL \le \{t_{HD;DAT}\ (max) - t_{AF(max)} - [(DNF+4) \times t_{I2CCLK}]\} / \{(PRESC + 1) \times t_{I2CCLK}\}$

注： 只有使能模拟滤波器时，公式中才包含 $t_{AF(min)}$ / $t_{AF(max)}$。有关 t_{AF} 值的信息，请参见器件数据手册。

标准模式、快速模式和超快速模式下的 tHD;DAT 最大值分别可达 3.45 µs、 0.9 µs 和 0.45 µs，但必须小于 tVD;DAT 最大值（差值为跳变时间）。只有器件未延长 SCL 信号的低电平周期 (tLOW) 时，才必须满足该最大值条件。如果时钟延长 SCL，数据必须在建立时间内保持有效，之后才能释放时钟。

SDA 上升沿通常为最坏情况，因此在这种情况下，上述公式变成如下形式：

SDADEL ≤{tVD;DAT (max) -tr (max) *-260 ns* - [(*DNF+4) x tI2CCLK*]} / {(PRESC +1) x tI2CCLK }。

注： NOSTRETCH=0 时会违反该条件，这是因为器件会根据 SCLDEL 值来延长 SCL 低电平时间，以保证建立时间。

有关 tf、 tr、 tHD;DAT 和 tVD;DAT 标准值的信息，请参见表 24-2。

- 在 tSDADEL 延时后，或在因数据未写入 I2C_TXDR 寄存器而导致从器件必须延长时钟的情况下发送 SDA 输出后， SCL 线会在建立时间内保持低电平。该建立时间为 tSCLDEL = (SCLDEL+1) x tPRESC，其中 tPRESC = (PRESC+1) x tI2CCLK。

tSCLDEL 会影响建立时间 tSU;DAT。

为了桥接 SDA 跳变（上升沿通常为最坏情况）的未定义区域，编程 SCLDEL 时必须遵循以下条件：

{[tr (max) + tSU;DAT (min)] / [(PRESC+1)] x tI2CCLK]} - 1 <= SCLDEL

有关 tr 和 tSU;DAT 标准值的信息，请参见表 24-2。

将使用的 SDA 和 SCL 跳变时间值就是应用中的值。使用最大值而非标准值会增加 SDADEL 和 SCL DEL 计算的约束条件，但能够确保任意应用的特性。

注： 在发送和接收模式下，对于每个时钟脉冲，检测到 *SCL* 下降沿后， *I2C* 主器件或从器件会至少在 *[(SDADEL+SCLDEL+1) x (PRESC+1) + 1] x tI2CCLK* 期间内延长 *SCL* 低电平时间。在发送模式下，如果 *SDADEL* 计数器计数结束后数据还未写入 *I2C_TXDR*，则 *I2C* 会继续延长 *SCL* 低电平时间，直到写入下一个数据。随后，会将新数据 *MSB* 发送到 *SDA* 输出，*SCLDEL* 计数器将开始计数，同时会继续延长 *SCL* 低电平时间以确保提供充足的数据建立时间。

如果从模式下 NOSTRETCH=1，则 SCL 不会延长。因此，编程 SDADEL 时还必须确保提供充足的建立时间。

表 24-2 I2C-SMBUS 规范数据建立和保持时间

符号	参数	标准模式(Sm)		快速模式(Fm)		超快速模式(Fm+)		SMBus		单位
		最小值	最大值	最小值	最大值	最小值	最大值	最小值	最大值	
t HD;DAT	数据保持时间	0	-	0	-	0	-	0.3		us
TVD;DAT	数据有效时间	-	3.45	-	0.9	-	0.45	-		
TSU;DAT	数据建立时间	250	-	100	50	250	ns			ns
tr	SDA 和 SCL 信号的上升时间	-	1000	300	-	120	-	1000		
tf	SDA 和 SCL 信号的下降时间	-	300	300	-	120	-	300		

此外，在主模式下，必须通过编程 I2C_TIMINGR 寄存器中的 PRESC[3:0]、SCLH[7:0] 和 SCLL[7:0] 位来配置 SCL 时钟的高电平和低电平。

- 当内部检测到 SCL 下降沿时，会在释放 SCL 输出之前插入一段延时。该延时为 tSCLL = (SCLL+1) x tPRESC，其中 tPRESC = (PRESC+1) x tI2CCLK。

tSCLL 会影响 SCL 低电平时间 tLOW。

- 当内部检测到 SCL 上升沿时，会在将 SCL 输出强制为低电平之前插入一段延时。该延时为 tSCLH = (SCLH+1) x tPRESC，其中 tPRESC = (PRESC+1) x tI2CCLK。tSCLH 会影响 SCL 高电平时间 tHIGH。

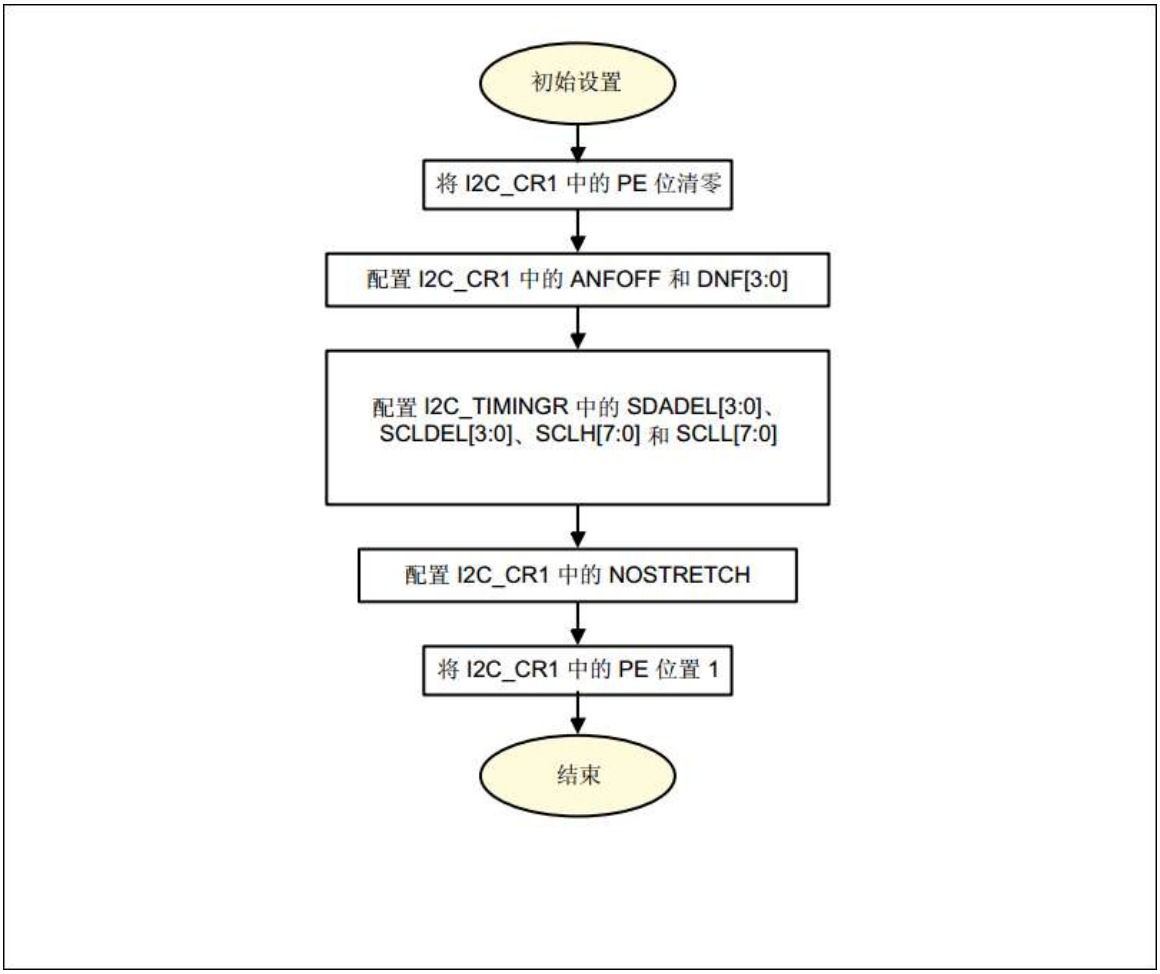
更多详细信息，请参见 *I2C* 主模式初始化。

注意： 使能 I2C 后，不允许更改时序配置。

此外，还必须在使能从设备前，对 NOSTRETCH 进行配置。更多详细信息，请参见 *I2C* 从模式初始化。

注意： 使能 I2C 后，不允许更改 NOSTRETCH 配置。

图 24-4 **I2C** 初始化流程图



24.3.5 软件复位

可通过将 I2C_CR1 寄存器中的 PE 位清零来执行软件复位。在这种情况下， I2C 线 SCL 和 SDA 被释放。内部状态机复位，通信控制位和状态位恢复为其复位值。配置寄存器不受影响。

下面列出了受影响的寄存器位：

1. I2C_CR2 寄存器： START、 STOP 和 NACK
2. I2C_ISR 寄存器：BUSY、TXE、TXIS、RXNE、ADDR、NACKF、TCR、TC、STOPF、BERR、 ARLO 和 OVR 支持 SMBus 功能时还会影响到以下寄存器位：
   - 1. I2C_CR2 寄存器： PECBYTE
   - 2. I2C_ISR 寄存器： PECERR、 TIMEOUT 和 ALERT

必须使 PE 保持低电平持续至少 3 个 APB 时钟周期，才能成功执行软件复位。写入以下软件序列可确保这一点： - 写入 PE=0 - 检查 PE=0 - 写入 PE=1

24.3.6 数据传输

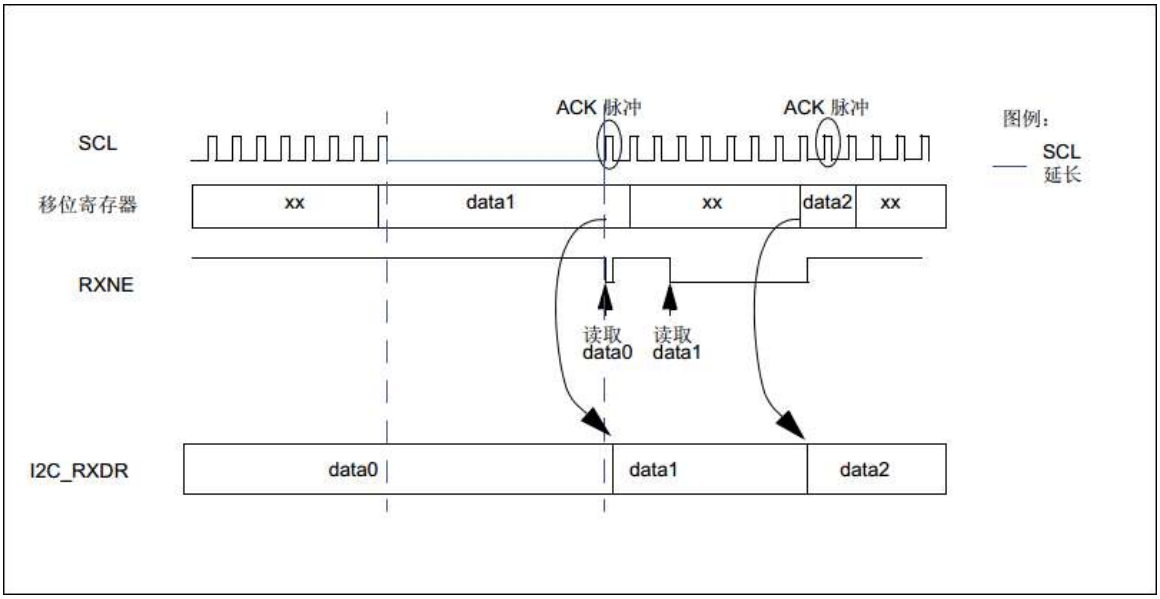
数据传输由发送和接收数据寄存器以及移位寄存器来管理。

接收

SDA 输入填充移位寄存器。在第 8 个 SCL 脉冲后（接收到完整的数据字节时），如果 I2C_RXDR

寄存器为空 (RXNE=0)，则移位寄存器的内容会复制到其中。如果 RXNE=1（意味着尚未读取上一次接收到的数据字节），则将延长 SCL 线的低电平时间，直到读取了 I2C_RXDR 为止。在第 8 个和第 9 个 SCL 脉冲之间（应答脉冲之前）插入一段延长的时间。

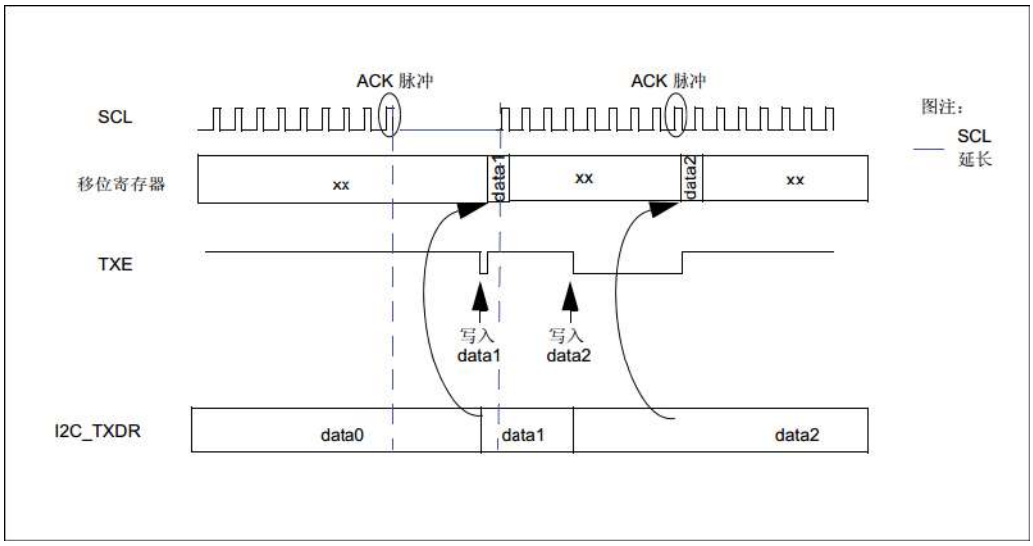
图 24-5 数据接收



发送

如果 I2C_TXDR 寄存器不为空 (TXE=0)，则其内容会在第 9 个 SCL 脉冲（应答脉冲）后复制到移位寄存器中。然后移位寄存器的内容会移出到 SDA 线上。如果 TXE=1（意味着 I2C_TXDR 内尚未写入任何数据），则将延长 SCL 线的低电平时间，直到写入了 I2C_TXDR 为止。在第 9 个 SCL 脉冲后进行延长。

图 24-6 数据发送



硬件传输管理

I2C 在硬件中内置了字节计数器，以便在下列各种模式下管理字节传输和结束通信：

- 主模式下生成 NACK、 STOP 和 ReSTART
- 从接收器模式下控制 ACK 是否发出
- SMBus 模式下生成/校验 PEC

字节计数器通常在主模式下使用。在从模式下，字节计数器默认为禁止状态，但可以通过软件来使能，方法是将 I2C_CR2 寄存器中的 SBC（从字节控制）位置 1。

待传输的字节数在 I2C_CR2 寄存器的 NBYTES[7:0] 位域中进行编程。如果待传输的字节数 (NBYTES) 大于 255，或者接收方希望控制是否对接收到的数据字节进行应答，则必须选择重载模式，方法是将 I2C_CR2 寄存器的 RELOAD 位置 1。在该模式下，完成 NBYTES 中所编程字节数的数据传输之后， TCR 标志将置 1，并且 TCIE 置 1 时将生成中断。只要 TCR 标志置 1， SCL 便会延长。当往 NBYTES 写入一个非零值时， TCR 由软件清零。

在往 NBYTE 中设置最后一次传输的字节数前，必须把 RELOAD 位清零。

当主模式下 RELOAD=0 时，可在以下 2 种模式下使用计数器：

- 自动结束模式（I2C_CR2 寄存器中的 AUTOEND =“1”）。在该模式下，一旦完成 NBYTES[7:0] 位域中所编程字节数的数据传输，主器件便会自动发送停止位。
- 软件结束模式（I2C_CR2 寄存器中的 AUTOEND =“0”）在该模式下，一旦完成 NBYTES[7:0] 位域中所编程字节数的数据传输， TC 标志将置 1，并且 TCIE 置 1 时将生成中断。只要 TC 标志置 1， SCL 信号便会延长，需要软件介入操作。当软件把 I2C_CR2 寄存器中的起始位或停止位置 1 时， TC 标志将被清零。当主器件要发送重复起始位时，必须使用该模式。

注意： 当 RELOAD 位置 1 时， AUTOEND 位将不起作用

表 24-3 I2C 配置表

功能	**SBC** 位	**RELOAD** 位	**AUTOEND** 位
主 Tx/Rx NBYTES + STOP	x	0	1
主 Tx/Rx + NBYTES + RESTART	x	0	0
从 Tx/Rx 接收的所有字节都要回复应答	0	x	x
具有 ACK 控制的从 Rx	1	1	x

24.3.7 从模式

I2C 从模式初始化

要在从模式下工作，用户必须至少使能一个从地址。可使用 I2C_OAR1 和 I2C_OAR2 这两个寄存器来编程自身从地址 OA1 和 OA2。

- OA1 既可配置为 7 位寻址模式（默认），也可通过将 I2C_OAR1 寄存器的 OA1MODE 位置 1 配置为 10 位寻址模式。

  通过将 I2C_OAR1 寄存器中的 OA1EN 位置 1 来使能 OA1。
- 如果需要额外的从地址，可配置第 2 个从地址 OA2。将 I2C_OAR2 寄存器的 OA2MSK[2:0] 位置 1 最多可屏蔽 7 个 OA2 LSB。因此，当 OA2MSK 配置为 1 到 6 时，将分别只有 OA2[7:2]、 OA2[7:3]、 OA2[7:4]、 OA2[7:5]、 OA2[7:6] 或 OA2[7] 与接收到的地址作比较。只要 OA2MSK 不等于 0， OA2 的地址比较器便会排除 I2C 保留地址 （0000 XXX 和 1111 XXX），这些地址将不会得到应答。如果 OA2MSK=7，接收到的所有 7 位地址（保留地址除外）均得到应答。 OA2 始终为 7 位地址。

  如果这些保留地址在 I2C_OAR1 或 I2C_OAR2 寄存器中进行了编程并且 OA2MSK=0，则它们可以在通过特定使能位使能后得到应答。

  通过将 I2C_OAR2 寄存器中的 OA2EN 位置 1 来使能 OA2。
- 通过将 I2C_CR1 寄存器中的 GCEN 位置 1 来使能广播呼叫地址。

当通过 I2C 的其中一个使能地址来寻址到该 I2C 设备时， ADDR 中断状态标志将置 1，并且 ADDRIE 位置 1 时将生成中断。

默认情况下，从器件使用其时钟延长功能（即必要时延长 SCL 信号的低电平时间）来为软件操作的执

行提供时机。如果主器件不支持时钟延长，则必须对 I2C 进行如下配置：将 I2C_CR1 寄存器中 NOSTRETCH 位置 1。

接收到 ADDR 中断后，如果使能多个地址，则用户必须读取 I2C_ISR 寄存器中的 ADDCODE[6:0] 位，以确定是哪个地址匹配。还必须检查 DIR 标志，以获悉传输方向。

带时钟延长的从模式 (NOSTRETCH = 0)

在默认模式下， I2C 从器件会在以下情况下延长 SCL 时钟：

- ADDR 标志置 1 时：接收到的地址与其中一个使能的从地址匹配。通过软件将 ADDRCF 位置 1 以清零 ADRR 标志时，将释放该时钟延展。
- 发送时，前一次数据传输已完成但 I2C_TXDR 寄存器中未写入任何新数据，或者 ADDR 标志清零 (TXE=1) 时未写入第一个数据字节。往 I2C_TXDR 寄存器中写入数据时，将释放该时钟延展。
- 接收时，尚未读取 I2C_RXDR 寄存器但新的数据接收已完成。读取 I2C_RXDR 时，将释放该时钟延展。
- 当从器件字节控制模式和重载模式（SBC=1 且 RELOAD=1）下 TCR = 1 时，这意味着最后一个数据字节已完成传输。通过向 NBYTES[7:0] 字段写入一个非零值以将 TCR 清零时，将释放该时钟延展。
- 在 SCL 下降沿检测之后， I2C 会延长 SCL 的低电平时间（不超过[(SDADEL+SCLDEL+1) x (PRESC+1) + 1] x t_{I2CCLK}）。

不带时钟延长的从模式 (NOSTRETCH = 1)

当 I2C_CR1 寄存器中的 NOSTRETCH = 1 时， I2C 从器件不会延长 SCL 信号。

- ADDR 标志置 1 时，不会延长 SCL 时钟。
- 发送时，必须在与发送数据对应的第一个 SCL 脉冲出现之前，向 I2C_TXDR 寄存器写入数据。否则，会发生下溢， I2C_ISR 寄存器中的 OVR 标志将置 1，如果 I2C_CR1 寄存器中的 ERRIE 位置 1，还将生成中断。当第一次数据发送开始而 STOPF 位仍置 1（尚未清零）时， OVR 标志也将置 1。因此，如果写入下一次传输要发送的第一个数据后才清零上一次传输的 STOPF 标志，则应提供 OVR 状态，甚至对于待发送的第一个数据也是如此。
- 接收时，必须在下一个数据字节的第 9 个 SCL 脉冲（ACK 脉冲）出现之前，从 I2C_RXDR 寄存器读取数据。否则，会发生上溢，I2C_ISR 寄存器中的 OVR 标志将置 1，如果 I2C_CR1 寄存器中的 ERRIE 位置 1，还将生成中断。

从器件字节控制模式

要在从接收模式下实现字节 ACK 控制，必须通过将 I2C_CR1 寄存器中的 SBC 位置 1 来使能从器件字节控制模式。这样符合 SMBus 标准。

要在从接收模式下实现字节 ACK 控制，必须选择重载模式 (RELOAD=1)。要控制每个字节，必须在 ADDR 中断子程序中将 NBYTES 初始化为 0x1，并在每接收一个字节后将 NBYTE 重载为 0x1。接收到字节后， TCR 位将置 1，从而延长 SCL 信号的第 8 个和第 9 个脉冲之间的低电平时间。用户可以从 I2C_RXDR 寄存器中读取数据，然后通过配置 I2C_CR2 寄存器中的 ACK 位来决定是否应答。通过将 NBYTES 编程为非零值来释放 SCL 延长：发送应答或不应答信号，然后可继续接收下一个字节。

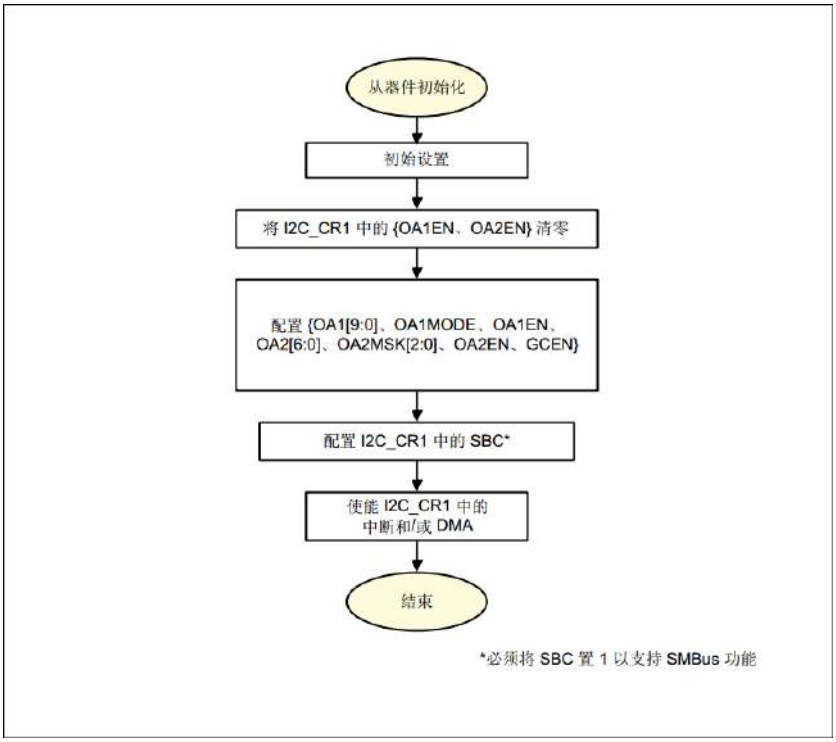
NBYTES 可加载大于 0x1 的值，在这种情况下，接收流在 NBYTES 个数据接收期间是连续的。

注： SBC 位只能在 I2C 被禁止时、从器件不被寻址时或 ADDR=1 时配置。

ADDR=1 或 TCR=1 时，可以更改 RELOAD 位的值。

注意： 从器件字节控制模式与 NOSTRETCH 模式不兼容。不允许在 NOSTRETCH=1 时将 SBC 位置 1。

图 24-7 从器件初始化流程图



从发送器

当 I2C_TXDR 寄存器为空时，将生成发送中断状态 (TXIS)。如果 I2C_CR1 寄存器中的 TXIE 位置 1，将生成中断。

I2C_TXDR 寄存器中写入待发送的下一个数据字节时， TXIS 位将被清零。

接收到 NACK 时， I2C_ISR 寄存器中的 NACKF 位将置 1，如果 I2C_CR1 寄存器中的 NACKIE 位置 1，还将生成中断。从器件自动释放 SCL 和 SDA 线，以使主器件执行停止或重复起始位的发送。收到 NACK 时， TXIS 位不会置 1。

当接收到停止位且 I2C_CR1 寄存器中的 STOPIE 位置 1 时， I2C_ISR 寄存器中的 STOPF 标志将置 1 并且会生成中断。在大多数应用中， SBC 位通常编程为“0”。在这种情况下，如果接收到从地址 (ADDR=1) 时 TXE = 0，用户可以选择发送 I2C_TXDR 寄存器的内容作为第一个数据字节，也可以选择通过将 TXE 位置 1 来刷新 I2C_TXDR 寄存器以编程新的数据字节。

在从器件字节控制模式 (SBC=1) 下，必须在地址匹配中断子程序 (ADDR=1) 中向 NBYTE 写入待发送数据的个数。在这种情况下，传输期间 TXIS 事件的数量对应于 NBYTES 中编程的值。

注意： 如果 NOSTRETCH=1，当 ADDR 标志置 1 时不会延长 SCL 时钟，因此用户无法在 ADDR 子程序中刷新 I2C_TXDR 寄存器的内容，从而编程第一个数据字节。必须在 I2C_TXDR 寄存器中预编程待发送的第一个数据字节：

- 该数据可以是前一个传输消息的最后一个 TXIS 事件中写入的数据。
- 如果该数据字节不是待发送的数据字节，可通过将 TXE 位置 1 来刷新 I2C_TXDR 寄存器，从而编程新的数据字节。必须仅在执行完这些操作后再清零 STOPF 位，以确保在地址应答之后，第一次数据传输开始之前执行这些操作。

如果第一次数据传输开始时 STOPF 仍置 1，则将生成下溢错误（OVR 标志置 1）。

如果需要 TXIS 事件（发送中断或发送 DMA 请求），用户必须将 TXE 位和 TXIS 位均置 1，以便生成 TXIS 事件。

图 24-8 **I2C** 从发送器的传输序列流程图 **(NOSTRETCH=0)**

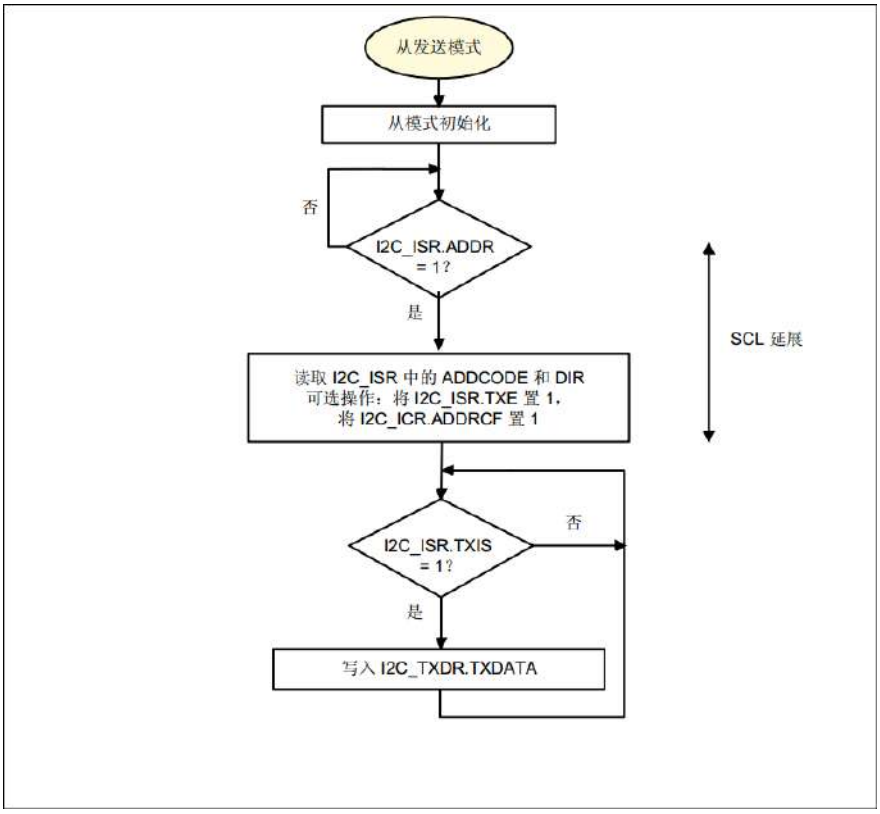


图 24-9 **I2C** 从发送器的传输序列流程图 **(NOSTRETCH=1)**

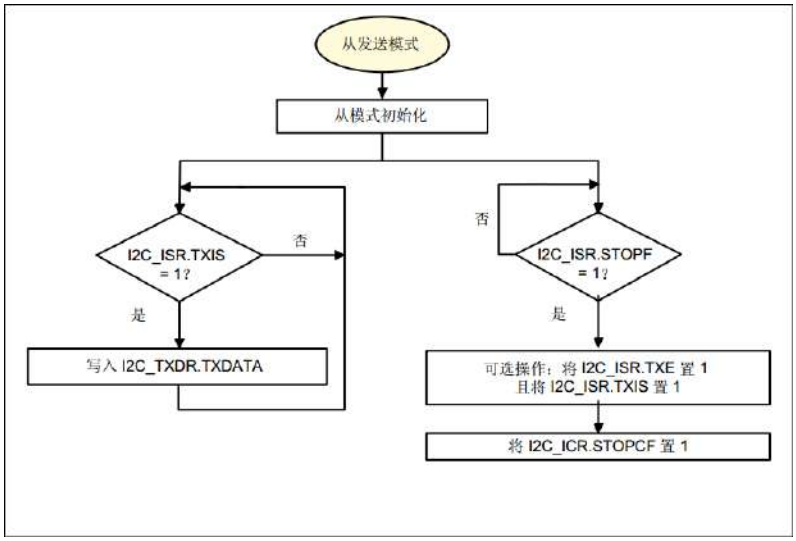
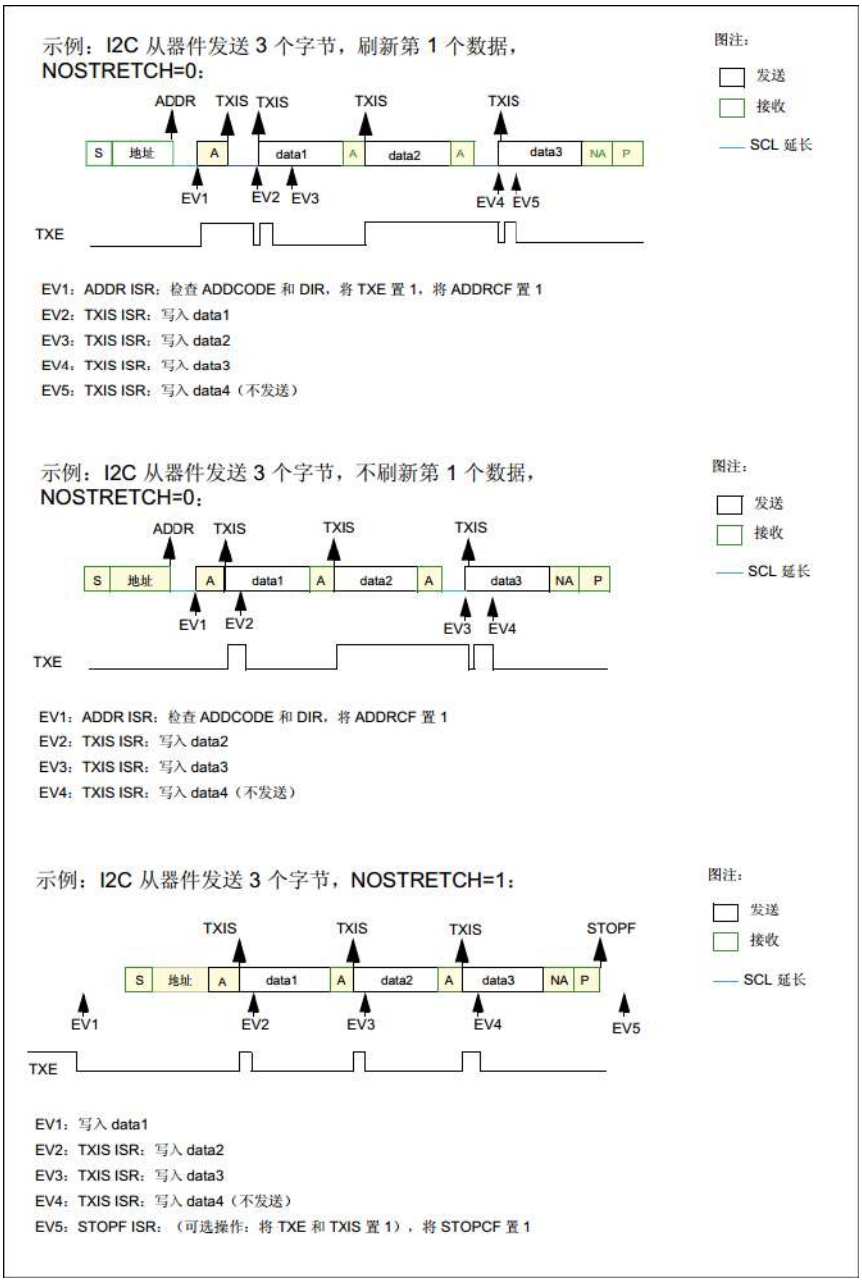


图 24-10 从发送器的传输总线图



从接收器

当 I2C_RXDR 满时， I2C_ISR 中的 RXNE 将置 1，如果 I2C_CR1 中的 RXIE 置 1，还将生成中断。读取 I2C_RXDR 时，将清零 RXNE。

接收到停止条件且 I2C_CR1 寄存器中的 STOPIE 置 1 时， I2C_ISR 中的 STOPF 将置 1 并且会生成中断。

图 24-11 从接收器的传输序列流程图 **(NOSTRETCH=0)**

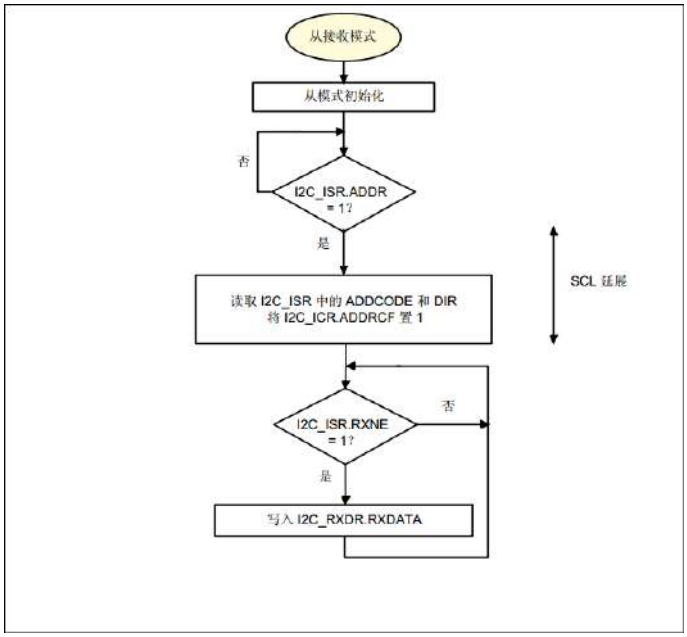


图 24-12 从接收器的传输序列流程图 **(NOSTRETCH=1)**

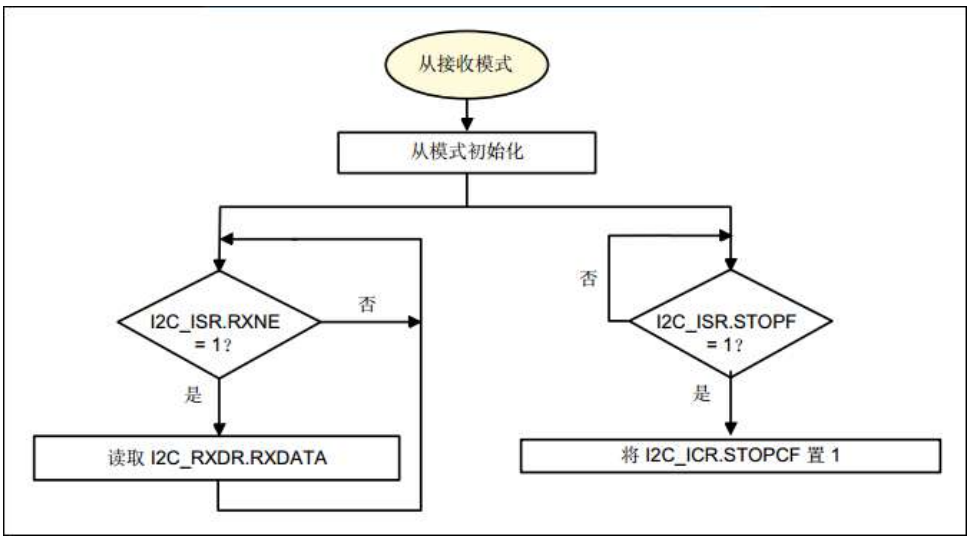
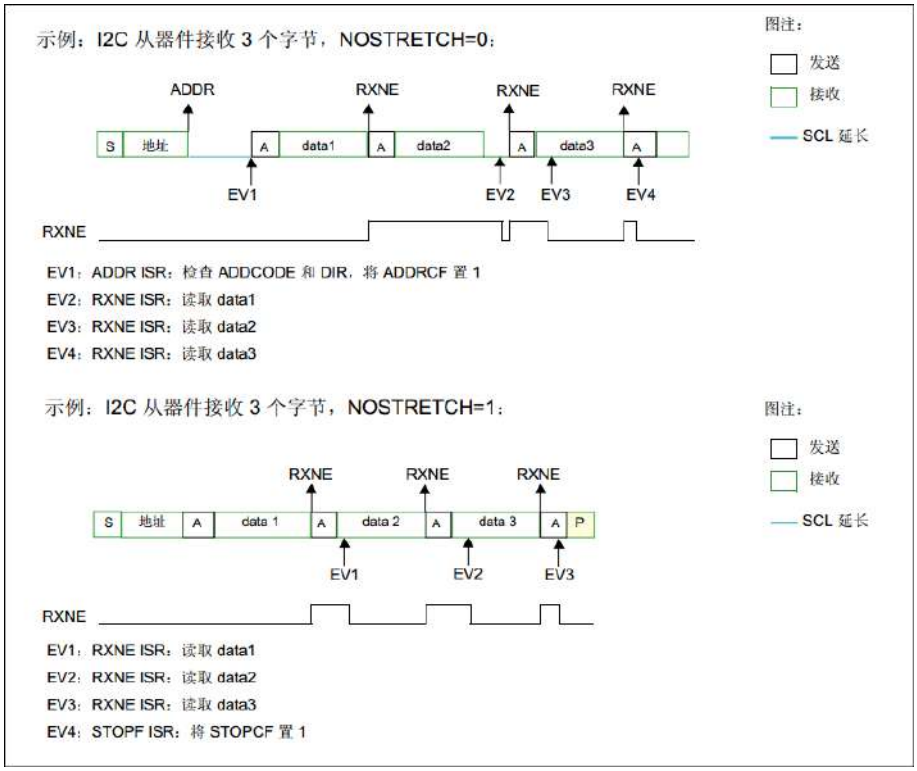


图 24-13 **I2C** 从接收器的传输总线图



24.3.8 主模式

I2C 主模式初始化

使能外设前，必须通过设置 I2C_TIMINGR 寄存器中的 SCLH 和 SCLL 位来配置 I2C 主时钟。

为了支持多主环境和从时钟延长， I2C 实现了时钟同步机制。

为了实现时钟同步，需执行以下操作：

- 使用 SCLL 计数器从 SCL 低电平内部检测开始对时钟的低电平进行计数。
- 使用 SCLH 计数器从 SCL 高电平内部检测开始对时钟的高电平进行计数。

I2C 经过 t_{SYNC1} 延时后检测其自身的 SCL 低电平，该延时取决于 SCL 下降沿、 SCL 输入噪声滤波器（模拟 + 数字）以及 SCL 与 I2CxCLK 时钟的同步。一旦 SCLL 计数器达到 I2C_TIMINGR 寄存器的 SCLL[7:0] 位中编程的值， I2C 便会将 SCL 释放为高电平。

I2C 经过 t_{SYNC2} 延时后检测其自身的 SCL 高电平，该延时取决于 SCL 上升沿、 SCL 输入噪声滤波器（模拟 + 数字）以及 SCL 与 I2CxCLK 时钟的同步。一旦 SCLH 计数器达到 I2C_TIMINGR 寄存器的 SCLH[7:0] 位中编程的值， I2C 便会使 SCL 变为低电平。

因此，主时钟周期为：

$t_{SCL} = t_{SYNC1} + t_{SYNC2} + \{[(SCLH+1) + (SCLL+1)] \times (PRESC+1) \times t_{I2CCLK}\}$

t_{SYNC1} 的持续时间取决于以下参数：

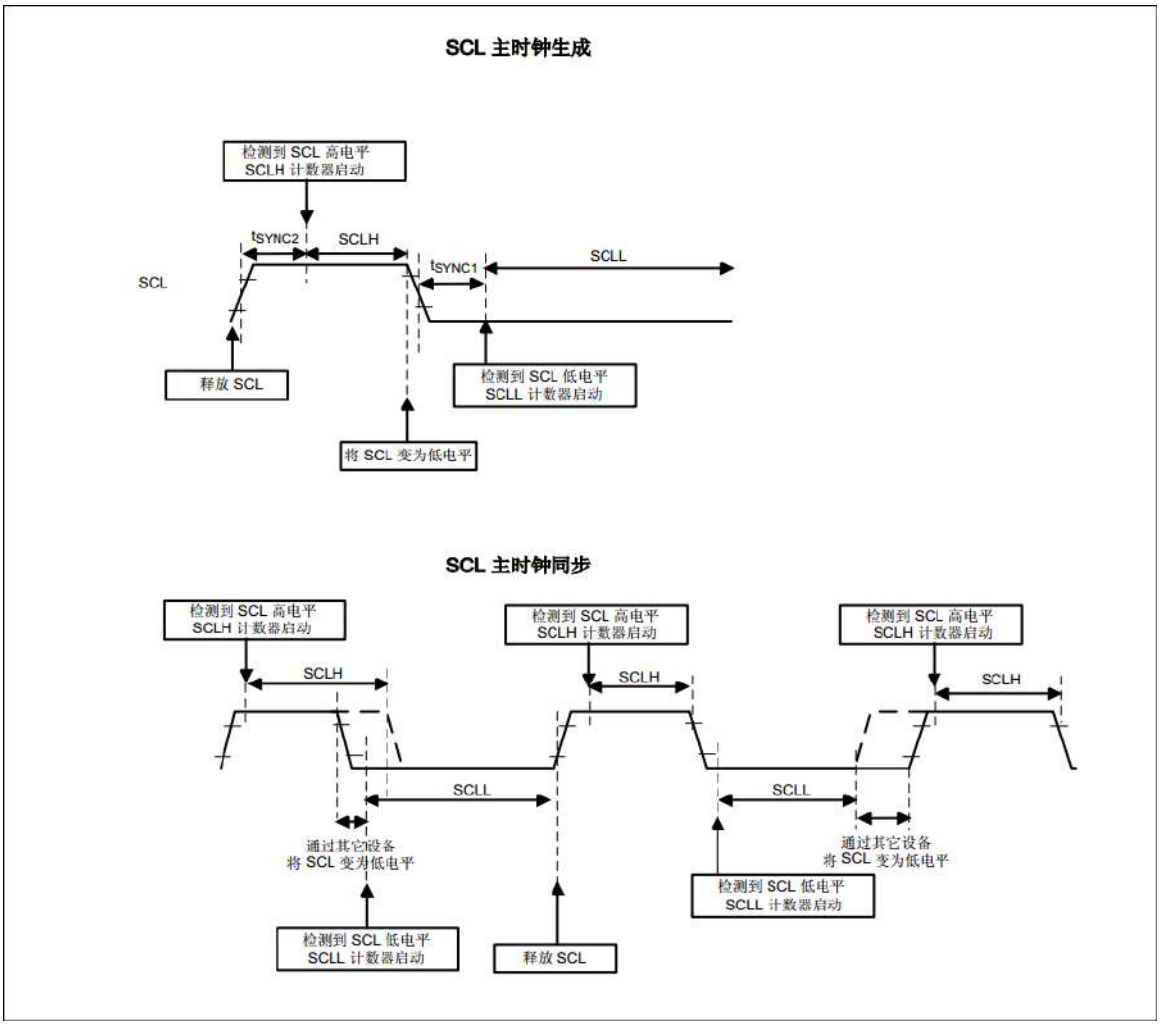
- SCL 下降斜率
- 模拟滤波器（使能时）引入的输入延时。
- 数字滤波器（使能时）引入的输入延时： DNF x t_{I2CCLK}
- SCL 与 I2CCLK 时钟建立同步而产生的延时（2 到 3 个 I2CCLK 周期）

t_{SYNC2} 的持续时间取决于以下参数：

- SCL 上升斜率

– 模拟滤波器（使能时）引入的输入延时。
– 数字滤波器（使能时）引入的输入延时： DNF x t_{I2CCLK}
– SCL 与 I2CCLK 时钟建立同步而产生的延时（2 到 3 个 I2CCLK 周期）

图 24-14 主时钟生成



注意： 为了符合 I2C 或 SMBus 规范，主时钟必须遵循下表中给出的时序：

表 24-4 **I2C-SMBUS** 规范时钟时序

符号	参数	标准模式（Sm）		快速模式（Fm）		超快速模式（Fm+）		SMBus		单位
		最小值	最大值	最小值	最大值	最小值	最大值	最小值	最大值	
f_{SCL}	SCL 时钟频率		100		400		1000		100	KHz
$t_{HD:STA}$	（重复）起始条件的保持时间	4.0	-	0.6		0.26		4.0	-	us
$t_{SU:STA}$	重复起始条件的建立时间	4.7	-	0.6		0.26		4.7	-	us
$t_{SU:STO}$	停止条件的建立时间	4.0	-	0.6		0.26	-	4.0	-	us
t_{BUF}	停止条件和起始条件之间的总线空闲时间	4.7	-	1.3		0.5	-	4.7	-	us

t_{LOW}	SCL 时钟的低电平周期	4.7	-	1.3		0.5	-	4.7	-	us
t_{HIGH}	SCL 时钟的高电平周期	4.0	-	0.6		0.26	-	4.0	50	us
t_r	SDA 和 SCL 信号的上升时间	-	1000	-	300		120	-	1000	us
t_f	SDA 和 SCL 信号的下降时间	-	300	-	300		120	-	300	us

注： SCLL 还用于生成 tBUF 和 tSU:STA 时序。

SCLH 还用于生成 tHD:STA 和 tSU:STO 时序。

有关 I2C_TIMINGR 设置与 I2CCLK 频率的示例，请参见 I2C_TIMINGR 寄存器配置示例。

主模式通信初始化（地址阶段）

要发起通信，用户必须在 I2C_CR2 寄存器中为寻址的从器件编程以下参数：

- 寻址模式（7 位或 10 位）： ADD10
- 待发送的从地址： SADD[9:0]
- 传输方向： RD_WRN
- 读取 10 位地址时： HEAD10R 位。必须对 HEAD10R 进行相应配置，以指示传输方向变化时必须发送完整的地址序列，还是只发送地址头。
- 待传输的字节数： NBYTES[7:0]。如果字节数等于或大于 255，则初始化时必须将 NBYTES[7:0] 填充为 0xFF。然后，用户必须将 I2C_CR2 寄存器中的 START 位置 1。START 位置 1 时，不允许更改上述所有位。

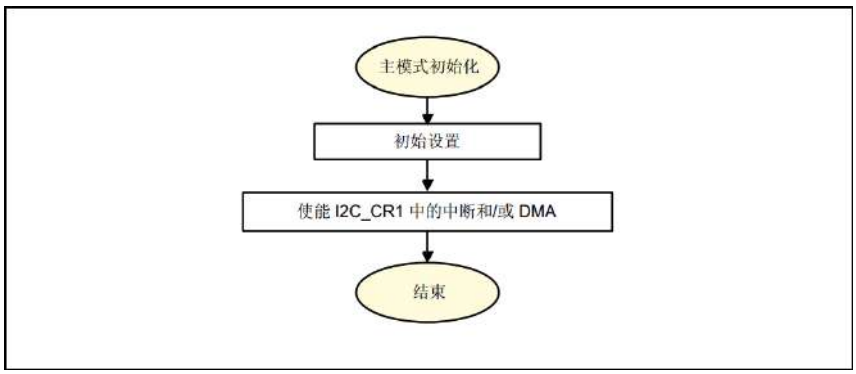
之后，当主器件检测到总线空闲 (BUSY = 0) 时，它会在经过 t_{BUF} 的延时后自动发送起始位，随后发出从器件地址。

仲裁丢失时，主器件将自动切换回从模式，如果作为从器件被寻址，还可对其自身地址进行应答。

注： 无论接收到的应答值为何，只要已在总线上发送从地址， START 位便会由硬件复位。如果仲裁丢失， START 位也会由硬件复位。如果当 START 位置 1 时， I2C 作为从器件 (ADDR=1)被寻址，则 I2C 将切换为从模式， START 位将在 ADDRCF 位置 1 时清零。

注： 该步骤同样适用于重复起始位。在这种情况下， BUSY=1。

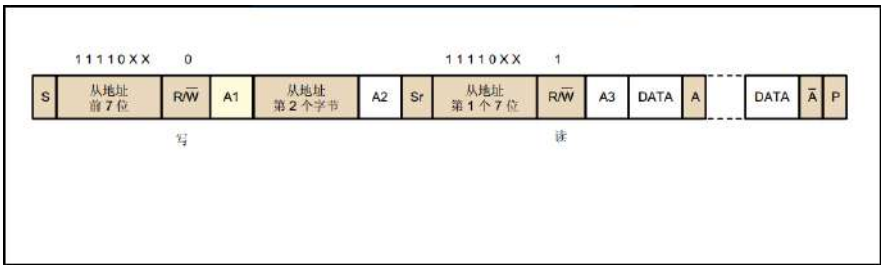
图 24-15 主模式初始化流程图



主接收器寻址 **10** 位地址从器件的初始化过程

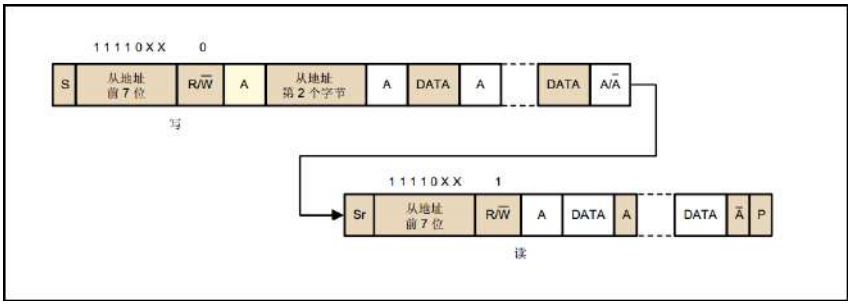
- 如果从地址采用 10 位格式，用户可选择将 I2C_CR2 寄存器中的 HEAD10R 位清零来发送完整的读序列。在这种情况下，主器件会在 START 位置 1 后自动发送以下完整序列：（重复）起始位 + 带写方向的从器件 10 位地址头字节 + 从器件地址第 2 个字节 +重复起始位 + 带读方向的从器件 10 位地址头字节

图 24-16 **10** 位地址读访问 **(HEAD10R=0)**



- 如果主器件对 10 位地址从器件进行寻址、向该从器件发送数据、然后再从该从器件读取数据，则必须首先完成主器件发送过程。然后，重复起始位置 1， 10 位从地址配置为 HEAD10R=1。在这种情况下，主器件发送以下序列：重复起始位 + 从地址 10 位头读取。

图 24-17 **10** 位地址读访问 **(HEAD10R=1)**



主发送器

写传输时，在发送完每个字节（即第 9 个 SCL 脉冲（接收到 ACK 时））后， TXIS 标志将置 1。

如果 I2C_CR1 寄存器中的 TXIE 位置 1， TXIS 事件将生成中断。当 I2C_TXDR 寄存器中写入待发送的下一个数据字节时，该标志将被清零。

传输期间的 TXIS 事件的数量对应于 NBYTES[7:0] 中编程的值。如果待发送的数据字节总数大于 255，则必须通过将 I2C_CR2 寄存器中的 RELOAD 位置 1 来选择重载模式。在这种情况下，当 NBYTES 数据传输完成时， TCR 标志将置 1，并且 SCL 线的低电平将被延展，直到 NBYTES[7:0] 被写入非零值。

收到 NACK 时， TXIS 标志不会置 1。

- 当 RELOAD=0 且 NBYTES 数据传输完成时：
  - 在自动结束模式 (AUTOEND=1) 下，将自动发送停止位。
  - 在软件结束模式 (AUTOEND=0) 下， TC 标志将置 1 且 SCL 线的低电平将被延展，以便执行以下软件操作：

  可通过将 I2C_CR2 寄存器中的 START 位置 1 并配置适当的从地址和待传输字节数来请求发送重复起始位。将 START 位置 1 会将 TC 标志清零，并在总线上发送起始位。

  可通过将 I2C_CR2 寄存器中的 STOP 位置 1 来请求停止位。将 STOP 位置 1 会将 TC 标志清零，并在总线上发送停止位。
- 如果接收到 NACK： TXIS 标志不会置 1，并且接收到 NACK 后会自动发送停止位。I2C_ISR 寄存器中的 NACKF 标志置 1，如果 NACKIE 位置 1，还将生成中断。

图 24-18 **I2C** 主发送器的传输序列流程图（**N≤255** 字节）

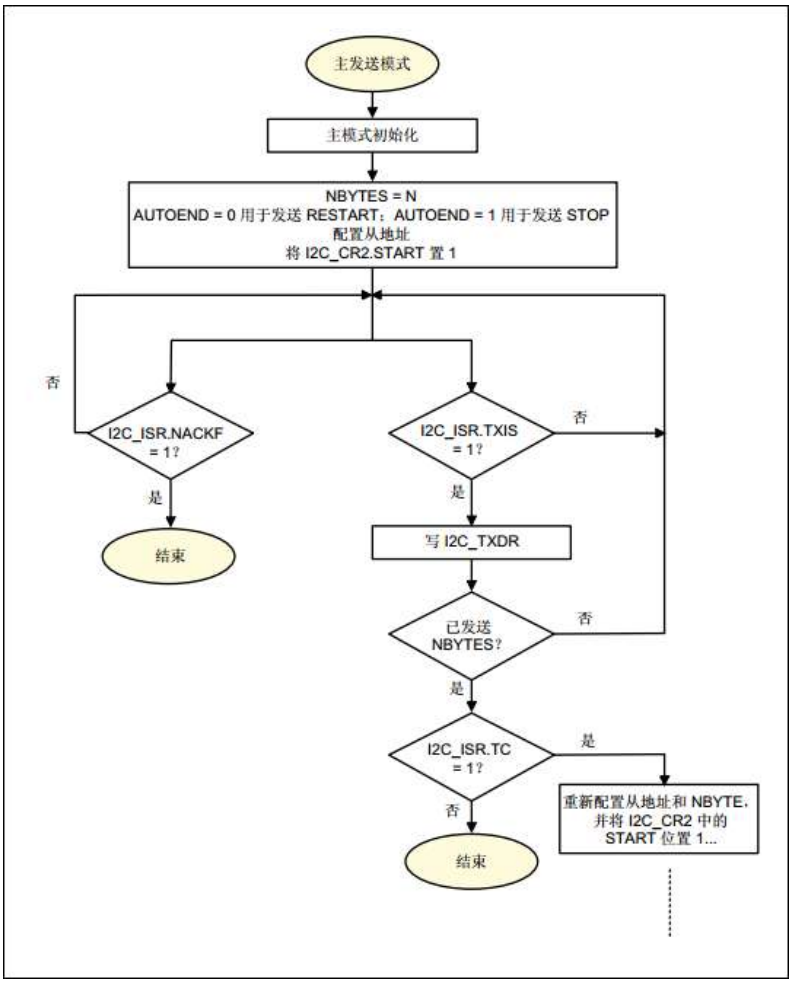


图 24-19 **I2C** 主发送器的传输序列流程图（**N>255** 字节）

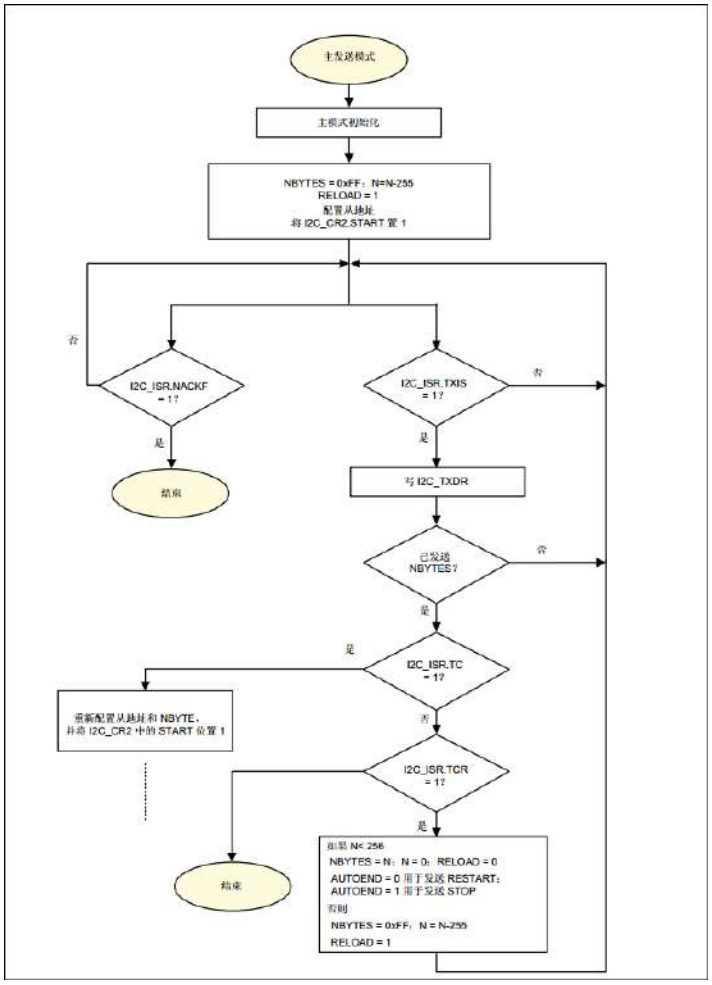
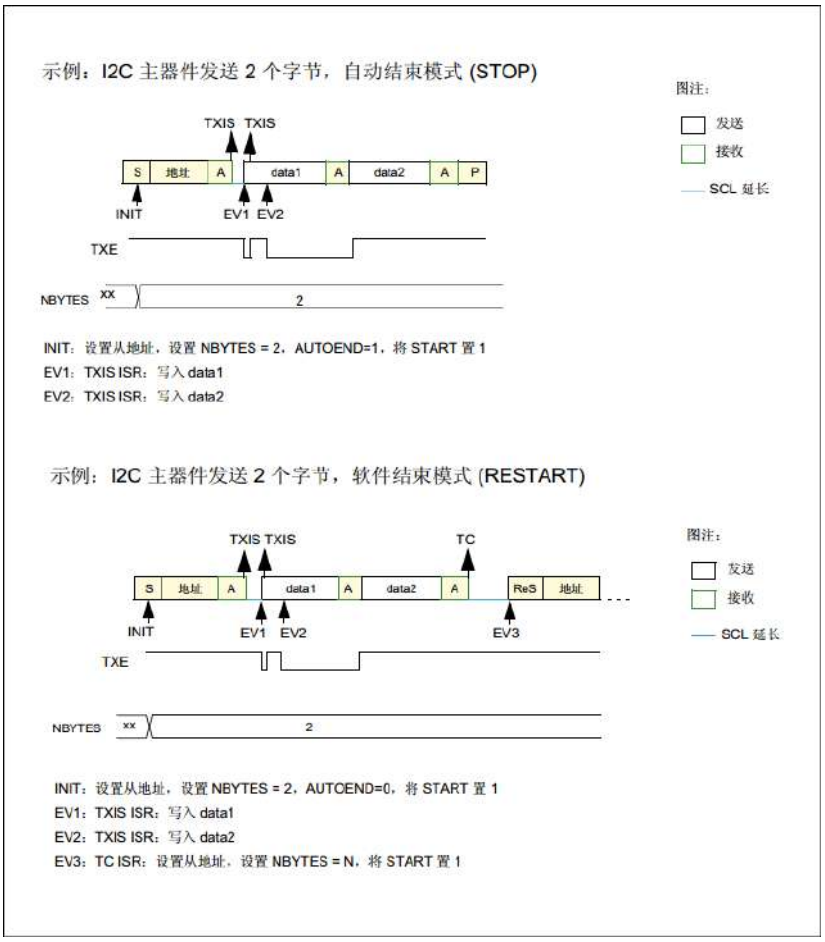


图 24-20 **I2C** 主发送器的传输总线图



主接收器

读传输时，在接收到每个字节（即第 8 个 SCL 脉冲）后， RXNE 标志将置 1。如果 I2C_CR1 寄存器中的 RXIE 位置 1， RXNE 事件将生成中断。读取 I2C_RXDR 时，将清零该标志。

如果待接收的数据字节总数大于 255，则必须通过将 I2C_CR2 寄存器中的 RELOAD 位置 1 来选择重载模式。在这种情况下，当 NBYTES[7:0] 数据传输完成时， TCR 标志将置 1，并且 SCL 线的低电平将被延展，直到 NBYTES[7:0] 被写入非零值。

- 当 RELOAD=0 且 NBYTES[7:0] 数据传输完成时：
  - 在自动结束模式 (AUTOEND=1) 下，接收到最后一个字节后，将自动发送 NACK 和停止位。
  - 在软件结束模式 (AUTOEND=0) 下，接收到最后一个字节后，将自动发送 NACK， TC 标志将置 1 且 SCL 线的低电平将被延展，以便执行以下软件操作：

可通过将 I2C_CR2 寄存器中的 START 位置 1 并配置适当的从地址和待传输字节数来请求发送重复起始位。将 START 位置 1 会将 TC 标志清零，并在总线上发送起始位，后跟从地址。

可通过将 I2C_CR2 寄存器中的 STOP 位置 1 来请求停止位。将 STOP 位置 1 会将 TC 标志清零，并在总线上发送停止位。

图 24-21 **I2C** 主接收器的传输序列流程图（**N≤255** 字节）

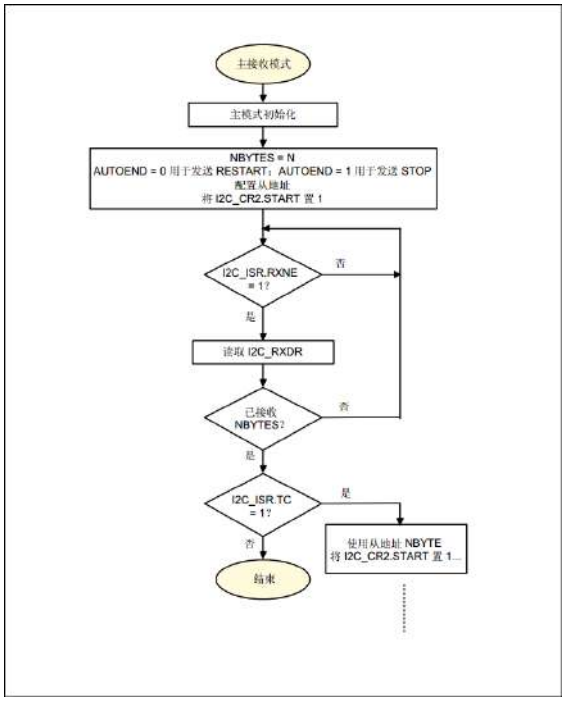


图 24-22 **I2C** 主接收器的传输序列流程图（**N>255** 字节）

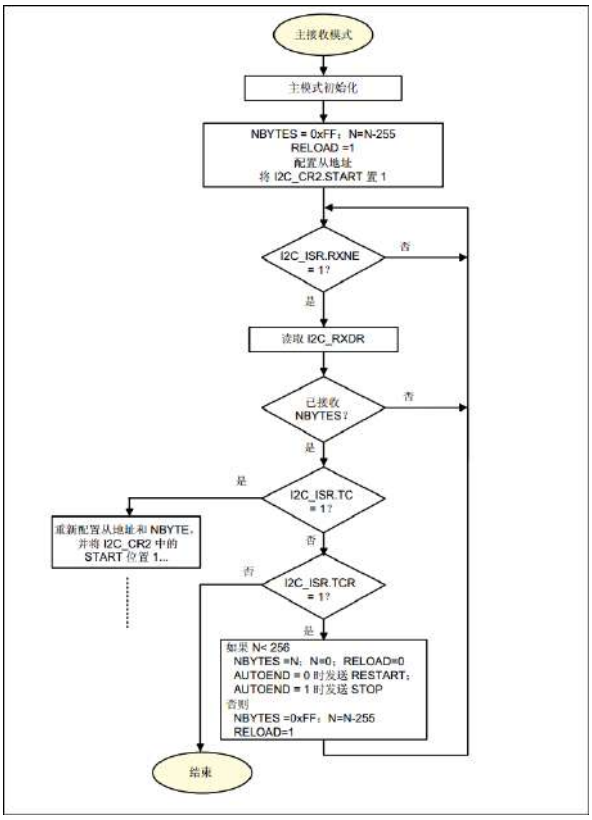
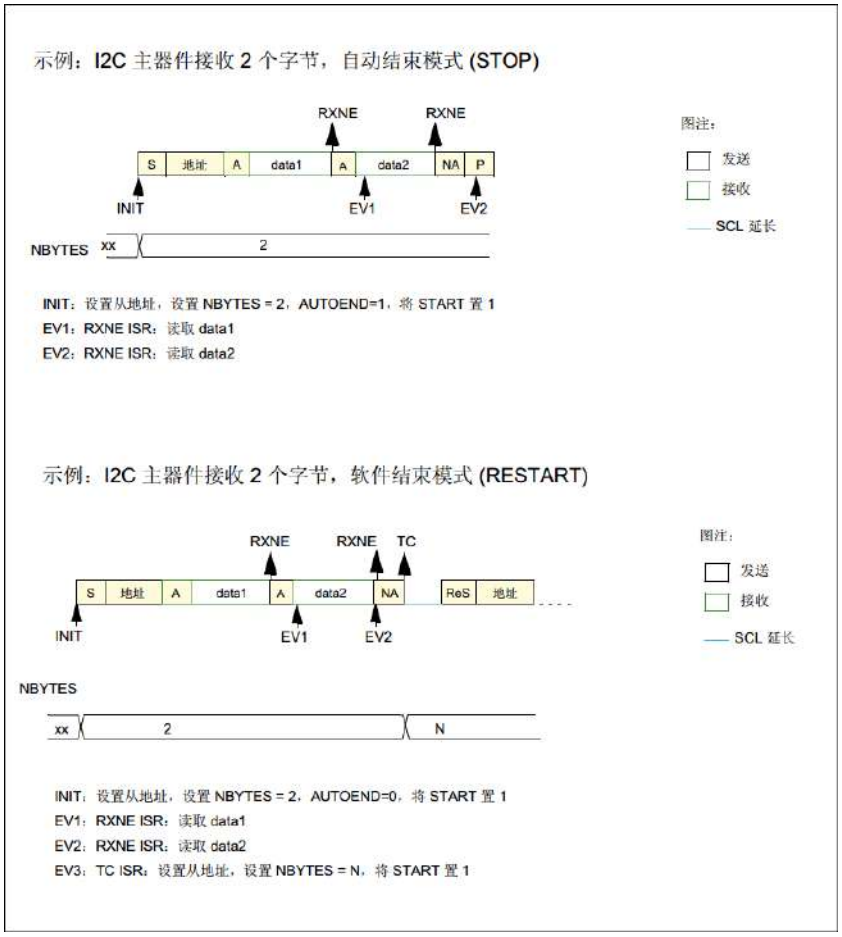


图 24-23 **I2C** 主接收器的传输总线图



24.3.9 寄存器配置示例

下文各表提供了相应示例，以介绍如何编程 I2C_TIMINGR 才能获得符合 I2C 规范的时序。

表 24-5 **f_{I2CCLK} = 8 MHz** 时的时序设置示例

参数	标准模式 **(Sm)**		快速模式 **(Fm)**	超快速模式 **(Fm+)**
	10 kHz	**100 kHz**	**400 kHz**	**500 kHz**
PRESC	1	1	0	0
SCLL	0xC7	0x13	0x9	0x6
t_{SCLL}	200x250 ns = 50 µs	20x250 ns = 5.0 µs	10x125 ns = 1250 ns	7x125 ns = 875 ns
SCLH	0xC3	0xF	0x3	0x3
t_{SCLH}	196x250 ns = 49 µs	16x250 ns = 4.0µs	4x125ns = 500ns	4x125 ns = 500 ns
t_{SCL}(1)	约 100 µs(2)	约 10 µs(2)	约 2500 ns(3)	约 2000 ns(4)
SDADEL	0x2	0x2	0x1	0x0
t_{SDADEL}	2x250 ns = 500 ns	2x250 ns = 500 ns	1x125 ns = 125 ns	0 ns
SCLDEL	0x4	0x4	0x3	0x1
t_{SCLDEL}	5x250 ns = 1250 ns	5x250 ns = 1250 ns	4x125 ns = 500 ns	2x125 ns = 250 ns

1. 由于 SCL 内部检测存在延时， SCL 周期 t_{SCL} 大于 t_{SCLL} + t_{SCLH}。为 t_{SCL} 提供的值仅用于举例说明。
2. t_{SYNC1} + t_{SYNC2} 最小值为 4 x t_{I2CCLK} = 500 ns。 t_{SYNC1} + t_{SYNC2} = 1000 ns 时的示例

3. t_{SYNC1} + t_{SYNC2} 最小值为 4 x t_{I2CCLK} = 500 ns。 t_{SYNC1} + t_{SYNC2} = 750 ns 时的示例
4. t_{SYNC1} + t_{SYNC2} 最小值为 4 x t_{I2CCLK} = 500 ns。 t_{SYNC1} + t_{SYNC2} = 655 ns 时的示例

表 24-6 **f_{I2CCLK} = 16 MHz** 时的时序设置示例

参数	标准模式 (Sm)		快速模式 (Fm)	超快速模式 (Fm+)
	10 kHz	**100 kHz**	**400 kHz**	**1000 kHz**
PRESC	3	3	1	0
SCLL	0xC7	0x13	0x9	0x4
t_{SCLL}	200 x 250 ns = 50 µs	20 x 250 ns = 5.0 µs	10 x 125 ns = 1250 ns	5 x 62.5 ns = 312.5 ns
SCLH	0xC3	0xF	0x3	0x2
t_{SCLH}	196 x 250 ns = 49 µs	16 x 250 ns = 4.0 µs	4 x 125 ns = 500 ns	3 x 62.5 ns = 187.5 ns
t_{SCL}(1)	约 100 µs(2)	约 10 µs(2)	约 2500 ns(3)	约 1000 ns(4)
SDADEL	0x2	0x2	0x2	0x0
t_{SDADEL}	2 x 250 ns = 500 ns	2 x 250 ns = 500 ns	2 x 125 ns = 250 ns	0 ns
SCLDEL	0x4	0x4	0x3	0x2
t_{SCLDEL}	5 x 250 ns = 1250 ns	5 x 250 ns = 1250 ns	4 x 125 ns = 500 ns	3 x 62.5 ns = 187.5 ns

1. 由于 SCL 内部检测存在延时， SCL 周期 tSCL 大于 tSCLL + tSCLH。为 tSCL 提供的值仅用于举例说明。
2. tSYNC1 + tSYNC2 最小值为 4 x tI2CCLK = 250 ns。tSYNC1 + tSYNC2 = 1000 ns 时的示例
3. tSYNC1 + tSYNC2 最小值为 4 x tI2CCLK = 250 ns。 tSYNC1 + tSYNC2 = 750 ns 时的示例
4. tSYNC1 + tSYNC2 最小值为 4 x tI2CCLK = 250 ns。 tSYNC1 + tSYNC2 = 500 ns 时的示例

24.3.10 特性

仅当支持 SMBus 功能时，才涉及本节内容。请参见 *I2C* 特性实现。

简介

系统管理总线 (SMBus) 是一个双线制接口，各器件可通过它在彼此之间或者与系统的其余部分进行通信。它以 I2C 的工作原理为基础。 SMBus 可针对系统和电源管理相关的任务提供控制总线。

该外设与 SMBUS 规范第 2.0 版兼容 (http://smbus.org)。

系统管理总线规范涉及三类器件。

- 从器件，用于接收或响应命令。
- 主器件，用于发出命令、生成时钟和中止传输。
- 主机，专用的主器件，可提供连接系统 CPU 的主接口。主机必须具有主 - 从器件功能，并且必须支持 SMBus 主机通知协议。系统中只允许存在一个主机。

该外设可配置为主器件或从器件，也可配置为主机。

SMBUS 以 I2C 规范第 2.1 版为基础。

总线协议

任何给定器件都有十一种可用命令协议。器件既可以在这十一种协议中任选其一，也可以使用全部十一种协议进行通信。这十一种协议分别为快速命令、发送字节、接收字节、写入字节、写入字、读取字节、读取字、过程调用、块读取、块写入以及块写入-块读取过程调用。

这些协议应通过用户软件实施。

有关这些协议的详细信息，请参见 SMBus 规范第 2.0 版 (http://smbus.org)。

地址解析协议 (ARP)

通过为各个从器件动态分配一个新的唯一地址可解决 SMBus 从地址冲突的问题。为了提供一种机制来针对地址分配隔离各个器件，各器件必须具有唯一的器件标识符 (UDID)。该 128 位数字由软件实现。

该外设支持地址解析协议 (ARP)。通过将 I2C_CR1 寄存器中的 SMBDEN 位置 1 来使能 SMBus 器件默认地址 (0b1100 001)。 ARP 命令应通过用户软件实现。

此外，还将在从模式下执行仲裁以支持 ARP。

有关 SMBus 地址解析协议的详细信息，请参见 SMBus 规范第 2.0 版(http://smbus.org)。

接收的命令和数据应答控制 SMBus 接收器必须能够对接收到的每个命令或数据进行否定应答。要在从模式下实现 ACK 控制，必须通过将 I2C_CR1 寄存器中的 SBC 位置 1 来使能从字节控制模式。更多详细信息，请参见第 *645* 页的“从器件字节控制模式”。

主机通知协议

该外设通过将 I2C_CR1 寄存器中的 SMBHEN 位置 1 来支持主机通知协议。在这种情况下，主机将应答 SMBus 主机地址 (0b0001 000)。

使用该协议时，器件作为主器件，而主机作为从器件。

SMBus 报警

器件支持 SMBus ALERT 可选信号。只具备从功能的器件可通过 SMBALERT# 引脚向主机发出信号，指示它想要通信。主机会处理该中断并通过报警响应地址 (0b0001 100) 同时访问所有 SMBALERT# 器件。只有那些将 SMBALERT# 拉到低电平的器件会应答报警响应地址。

如果配置为从器件 (SMBHEN=0)，则通过将 I2C_CR1 寄存器中的 ALERTEN 位置 1 来将 SMBA 引脚拉为低电平。这同时还会使能报警响应地址。

如果配置为主机 (SMBHEN=1)，则当 SMBA 引脚上检测到下降沿且 ALERTEN=1 时，

I2C_ISR 寄存器中的 ALERT 标志置 1。如果 I2C_CR1 寄存器中的 ERRIE 位置 1，将生成中断。当 ALERTEN=0 时，即使外部 SMBA 引脚为低电平， ALERT 线也将被视为高电平。

如果无需 SMBus ALERT 引脚，则当 ALERTEN=0 时， SMBA 引脚可用作标准 *GPIO*。

数据包错误校验

SMBus 规范中引入了数据包错误校验机制来提高可靠性和通信稳定性。数据包错误校验的实施方式是在每次消息传输结束时附加数据包错误代码 (PEC)。PEC 的计算方式是对所有消息字节（包括地址和读/写位）使用 CRC-8 多项式 $C(x) = x^8 + x^2 + x + 1$。

外设内置了硬件 PEC 计算器，可在接收到的字节与硬件计算的 PEC 不匹配时自动发送否定应答信号。

超时

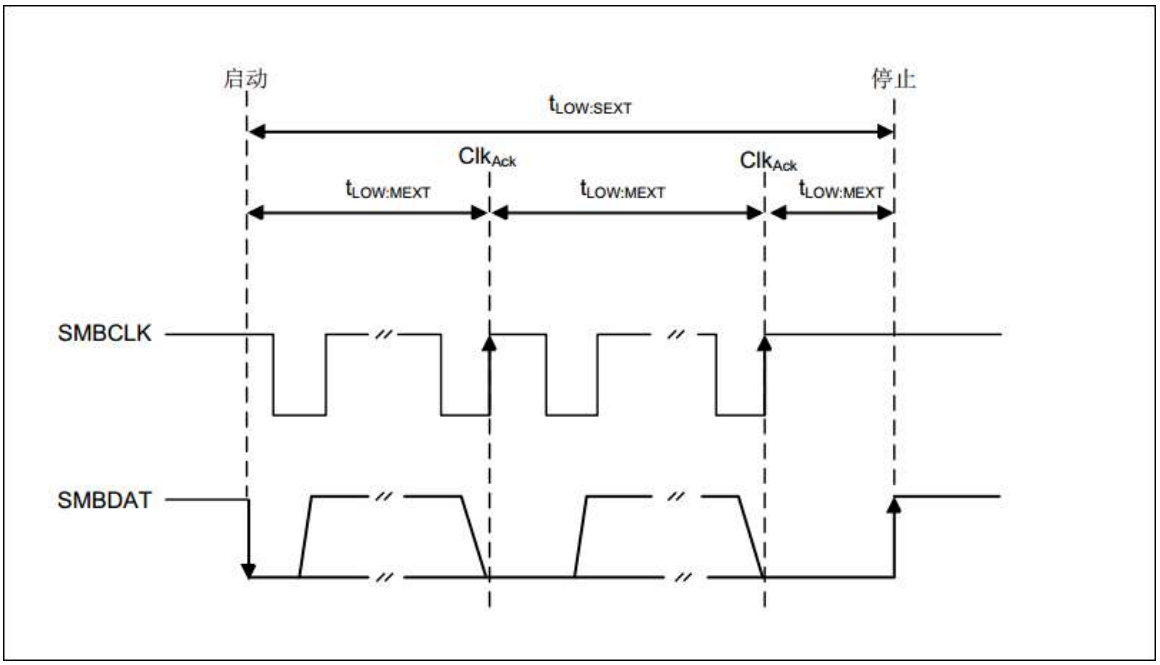
该外设内置了硬件定时器，以便符合 SMBus 规范第 2.0 版中定义的 3 个超时。

表 24-7 SMBus 超时规范

符号	参数	限值		单位
		最小值	最大值	
$t_{TIMEOUT}$	检测时钟低电平超时	25	35	ms
$t_{LOW:SEXT}$(1)	累积时钟低电平延长时间（从器件）	-	25	ms
$t_{LOW:MEXT}$(2)	累积时钟低电平延长时间（主器件）	-	10	ms

1. $t_{LOW:SEXT}$ 是一段累积时间，即给定从器件在一条消息的最初起始到停止期间时钟信号可延展的时间。其它从器件或主器件也可能延长时钟，进而导致时钟低电平总延长时间超过 $t_{LOW:SEXT}$。因此，测量该参数时该器件应该是全速主器件寻址的唯一器件。
2. $t_{LOW:MEXT}$ 是一段累积时间，即主器件在消息的每个字节（定义为 START 到 ACK、 ACK 到 ACK 或 ACK 到 STOP）内时钟信号可延展的时间。从器件或其它主器件也可能延长时钟，进而导致时钟低电平总时间超过 $t_{LOW:MEXT}$（针对给定字节）。因此，测量该参数时该全速主器件只寻址一个从器件。

图 24-24 **tLOW:SEXT** 和 **tLOW:MEXT** 的超时间隔



总线空闲检测

如果主器件检测到时钟和数据信号的高电平时间已达 t_{IDLE}（超过 $t_{HIGH,MAX}$），则认为总线空闲（请参见表 24-4）。

该时序参数已考虑如下情况：主器件已动态添加至总线，但可能尚未检测到 SMBCLK 或 SMBDAT 线上的状态转换。在这种情况下，主器件必须等待足够长的时间，以确定当前未进行传输。外设支持硬件总线空闲检测。

24.3.11 初始化

仅当支持 SMBus 功能时，才涉及本节内容。请参见 *I2C* 特性实现。

除了 I2C 初始化之外，还必须进行一些其它的特定初始化，以便执行 SMBus 通信：

接收的命令和数据应答控制（从模式）

SMBus 接收器必须能够对接收到的每个命令或数据进行否定应答。要在从模式下实现 ACK 控制，必须通过将 I2C_CR1 寄存器中的 SBC 位置 1 来使能从字节控制模式。

特定地址（从模式）

必要时应使能特定的 SMBus 地址。更多详细信息，请参见 “总线空闲检测”。• 通过将 I2C_CR1 寄存器中的 SMBDEN 位置 1 来使能 SMBus 器件默认地址 (0b1100001)。

- 通过将 I2C_CR1 寄存器中的 SMBHEN 位置 1 来使能 SMBus 主机地址 (0b0001 000)。
- 通过将 I2C_CR1 寄存器中的 ALERTEN 位置 1 来使能报警响应地址 (0b0001100)。

数据包错误校验

通过将 I2C_CR1 寄存器中的 PECEN 位置 1 来使能 PEC 的计算。然后，借助硬件字节计数器（I2C_CR2 寄存器中的 NBYTES[7:0]）来管理 PEC 传输。使能 I2C 之前，必须配置 PECEN 位。

PEC 传输由硬件字节计数器来管理，因此在从模式下连接 SMBus 时必须将 SBC 位置 1。当 PECBYTE 位置 1 且 RELOAD 位清零时，传输完 NBYTES-1 字节的数据后会传输 PEC。如果 RELOAD 置 1，PECBYTE 将不起作用。

注意： 使能 I2C 时，不允许更改 PECEN 配置。

表 24-8 带 **PEC** 的 **SMBUS** 配置

模式	**SBC** 位	**RELOAD** 位	**AUTOEND** 位	**PECBYTE** 位
主 Tx/Rx NBYTES + PEC+ STOP	x	0	1	1
主 Tx/Rx NBYTES + PEC + ReSTART	x	0	0	1
从 Tx/Rx + PEC	1	0	x	1

超时检测

将 I2C_TIMEOUTR 寄存器中的 TIMOUTEN 和 TEXTEN 位置 1 来使能超时检测。定时器必须按如下方式编程：即在 SMBus 规范第 2.0 版规定的时间最大值之前检测出超时情况。

- tTIMEOUT 检查

  要使能 $t_{TIMEOUT}$ 检查，必须将 12 位 TIMEOUTA[11:0] 位编程为定时器重载值，以检查 $t_{TIMEOUT}$ 参数。必须将 TIDLE 位配置为“0”，以检测 SCL 低电平超时。

  然后，通过将 I2C_TIMEOUTR 寄存器中的 TIMOUTEN 位置 1 来使能定时器。

  如果 SCL 的低电平持续时间超过 (TIMEOUTA+1) x 2048 x t_{I2CCLK}， I2C_ISR 寄存器中的 TIMEOUT 标志将置 1。

注意： TIMEOUTEN 位置 1 时，不允许更改 TIMEOUTA[11:0] 位和 TIDLE 位的配置。

- tLOW:SEXT 和 tLOW:MEXT 检查

  必须根据外设配置为主器件还是从器件来配置 TIMEOUTB 定时器，以便为从器件校验 $t_{LOW:SEXT}$，为主器件校验 $t_{LOW:MEXT}$。由于标准只规定了最大值，用户可以为这两个参数选择相同的值。

然后，通过将 I2C_TIMEOUTR 寄存器中的 TEXTEN 位置 1 来使能定时器。

如果 SMBus 外设延展 SCL 的累积时间超过 (TIMEOUTB+1) x 2048 x t_{I2CCLK}，并且达到 “总线空闲检测” 一节给出的超时间隔，则 I2C_ISR 寄存器中的 TIMEOUT 标志将置 1。

请参见表 24-10。

注意： TEXTEN 位置 1 时，不允许更改 TIMEOUTB 配置。

总线空闲检测

要使能 t_{IDLE} 检查，必须将 12 位 TIMEOUTA[11:0] 字段编程为定时器重载值，以获取 t_{IDLE} 参数。必须将 TIDLE 位配置为“1”，以检测 SCL 和 SDA 高电平超时。然后，通过将 I2C_TIMEOUTR 寄存器中的 TIMOUTEN 位置 1 来使能定时器。

如果 SCL 和 SDA 线的高电平持续时间超过 (TIMEOUTA+1) x 4 x t_{I2CCLK}， I2C_ISR 寄存器中的 TIMEOUT 标志将置 1。

请参见表 24-11 。

注意： TIMEOUTEN 置 1 时，不允许更改 TIMEOUTA 和 TIDLE 配置。

24.3.12 SMBus：I2C_TIMEOUTR 寄存器配置示例

仅当支持 SMBus 功能时，才涉及本节内容。请参见第 *28.3* 节： *I2C* 特性实现。

- 将 $t_{TIMEOUT}$ 的最大持续时间配置为 25 ms：

表 24-9 不同 **I2CCLK** 频率下的 **TIMEOUTA** 设置示例（最大 $t_{TIMEOUT}$ = **25 ms**）

fI2CCLK	TIMEOUTA[11:0] 位	TIDLE 位	TIMEOUTEN 位	tTIMEOUT
8 MHz	0x61	0	1	98 x 2048 x 125 ns = 25 ms
16 MHz	0xC3	0	1	196 x 2048 x 62.5 ns = 25 ms
32 MHz	0x186	0	1	391 x 2048 x 31.25 ns = 25 ms

- 将 $t_{LOW:SEXT}$ 和 $t_{LOW:MEXT}$ 的最大持续时间配置为 8 ms：

表 24-10 不同 **I2CCLK** 频率下的 **TIMEOUTB** 设置示例

fI2CCLK	TIMEOUTB[11:0] 位	TEXTEN 位	tLOW:EXT
8 MHz	0x1F	1	32 x 2048 x 125 ns = 8 ms
16 MHz	0x3F	1	64 x 2048 x 62.5 ns = 8 ms
32 MHz	0x7C	1	125 x 2048 x 31.25 ns = 8 ms

- 将 tIDLE 的最大持续时间配置为 50 µs

表 24-11 不同 **I2CCLK** 频率下的 **TIMEOUTA** 设置示例（最大 **tIDLE = 50 µs**）

fI2CCLK	TIMEOUTA[11:0] 位	TIDLE 位	TIMEOUTEN 位	tTIDLE
8 MHz	0x63	1	1	100 x 4 x 125 ns = 50 µs
16 MHz	0xC7	1	1	200 x 4 x 62.5 ns = 50 µs
32 MHz	0x18F	1	1	400 x 4 x 31.25 ns = 50 µs

24.3.13 SMBus 从模式

仅当支持 SMBus 功能时，才涉及本节内容。请参见第 *28.3* 节： *I2C* 特性实现。

除了 I2C 从模式传输管理（请参见第 *28.4.8* 节： *I2C* 从模式）之外，还提供了一些额外的软件流程图来支持 SMBus。

SMBus 从发送器

在 SMBus 模式下使用 IP 时，必须将 SBC 编程为“1”，以便在完成已编程数据字节数的传输后进行 PEC 传输。当 PECBYTE 位置 1 时， NBYTES[7:0] 中编程的字节数包含 PEC 传输。在这种情况下，总 TXIS 中断数为 NBYTES-1，如果主器件在完成 NBYTES-1 字节的数据传输后请求传输额外的字节，则将自动发送 I2C_PECR 寄存器的内容。

注意： 当 RELOAD 位置 1 时， PECBYTE 位将不起作用。

图 24-25 **SMBus** 从发送器的传输序列流程图（**N** 字节 **+ PEC**）

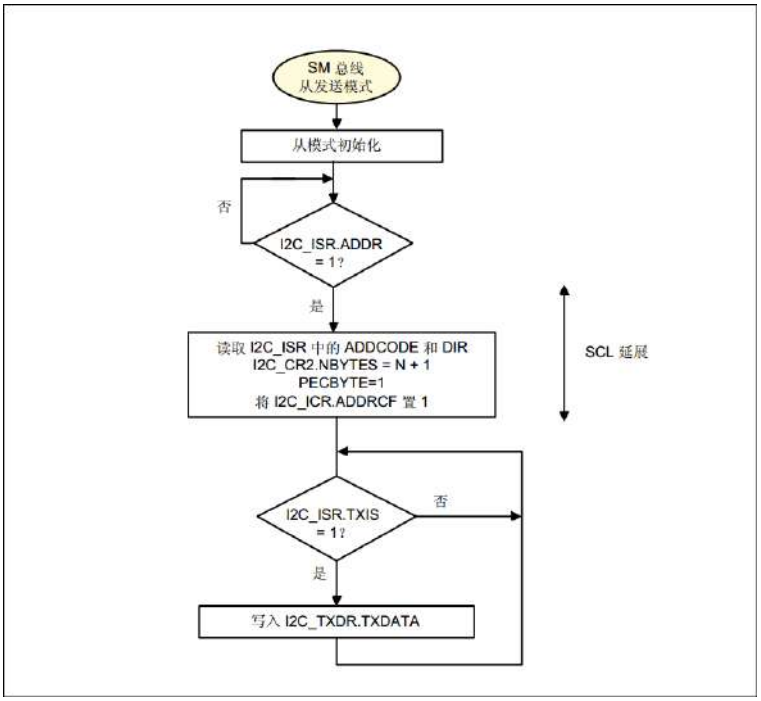
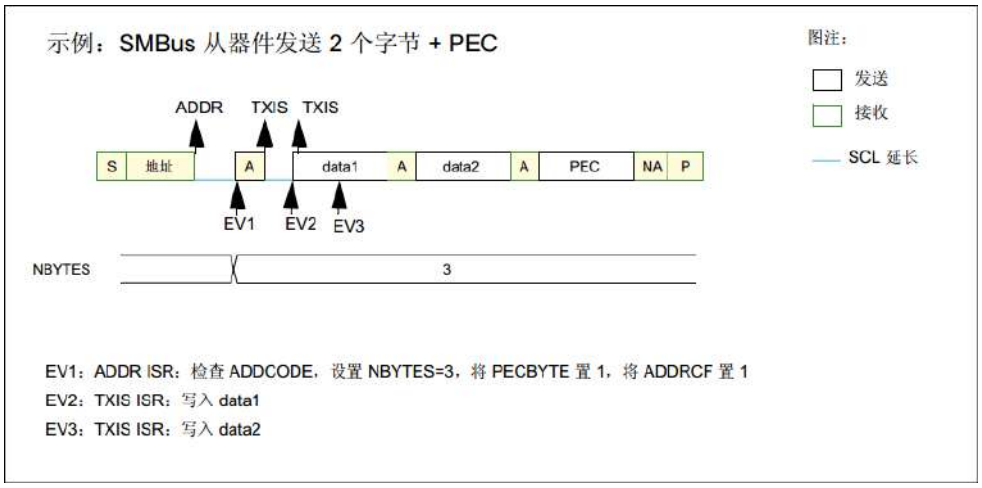


图 24-26 **SMBus** 从发送器的传输总线图 **(SBC=1)**



SMBus 从接收器

在 SMBus 模式下使用 I2C 时，必须将 SBC 编程为“1”，以便在完成已编程数据字节数的传输后进行 PEC 校验。要对每个字节进行 ACK 控制，必须选择重载模式 (RELOAD=1)。更多详细信息，请参见 “从器件字节控制模式”。要校验 PEC 字节，必须将 RELOAD 位清零并将 PECBYTE 位置 1。在这种情况下，当接收到 NBYTES-1 字节的数据后，接收的下一个字节将与内部 I2C_PECR 寄存器的内容作比较。如果比较不匹配，则将自动生成 NACK 信号；如果比较匹配，则将自动生成 ACK 信号，而与 ACK 位的值无关。 PEC 字节一经接收，便会像任何其它数据一样复制到 I2C_RXDR 寄存器中，并且 RXNE 标志将置 1。

当 PEC 不匹配时， PECERR 标志将置 1，如果 I2C_CR1 寄存器中的 ERRIE 位置 1，还将生成中断。

如果无需 ACK 软件控制，用户可编程 PECBYTE=1，在同一写操作下，将 NBYTES 编程为连续接收的字节数。接收到 NBYTES-1 字节的数据后，会将接收的下一个字节视为 PEC 进行校验。

注意： 当 RELOAD 位置 1 时， PECBYTE 位将不起作用。

图 24-27 **SMBus** 从接收器的传输序列流程图（**N** 字节 **+ PEC**）

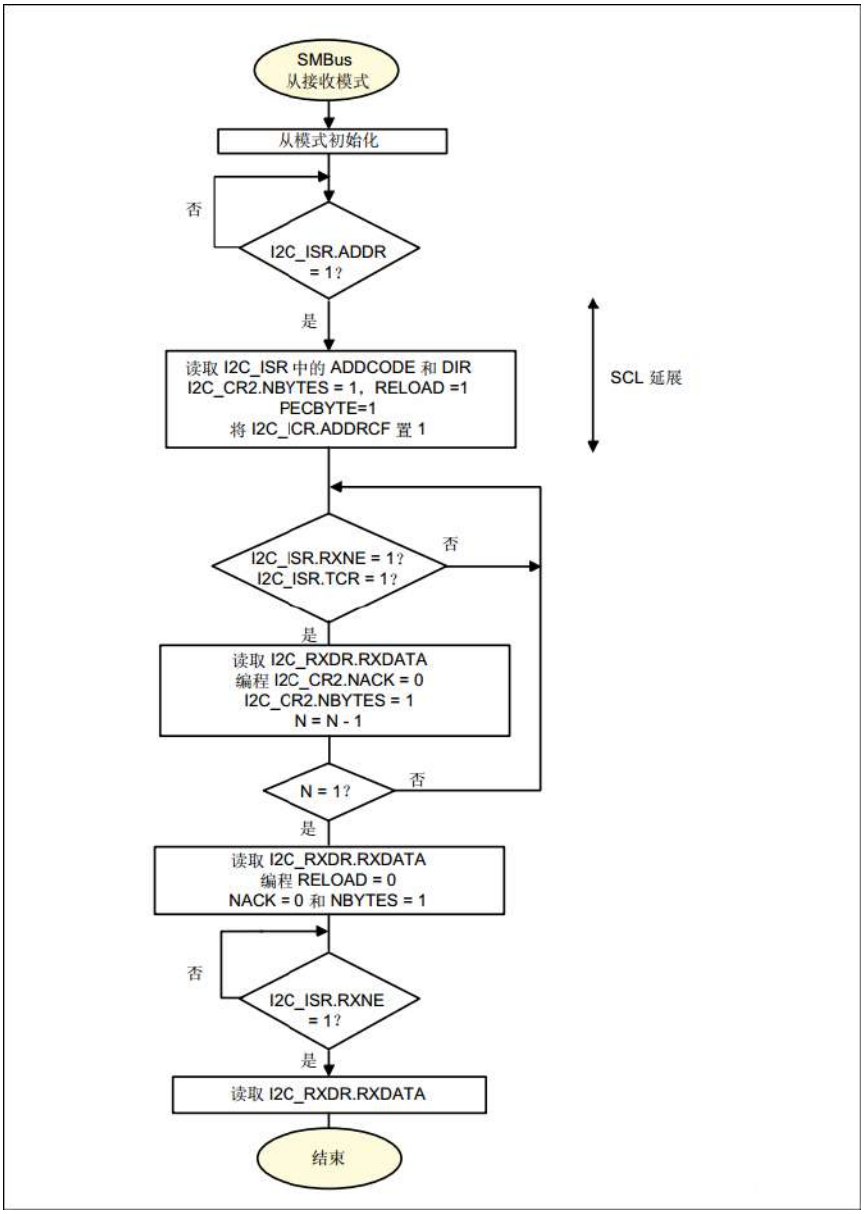
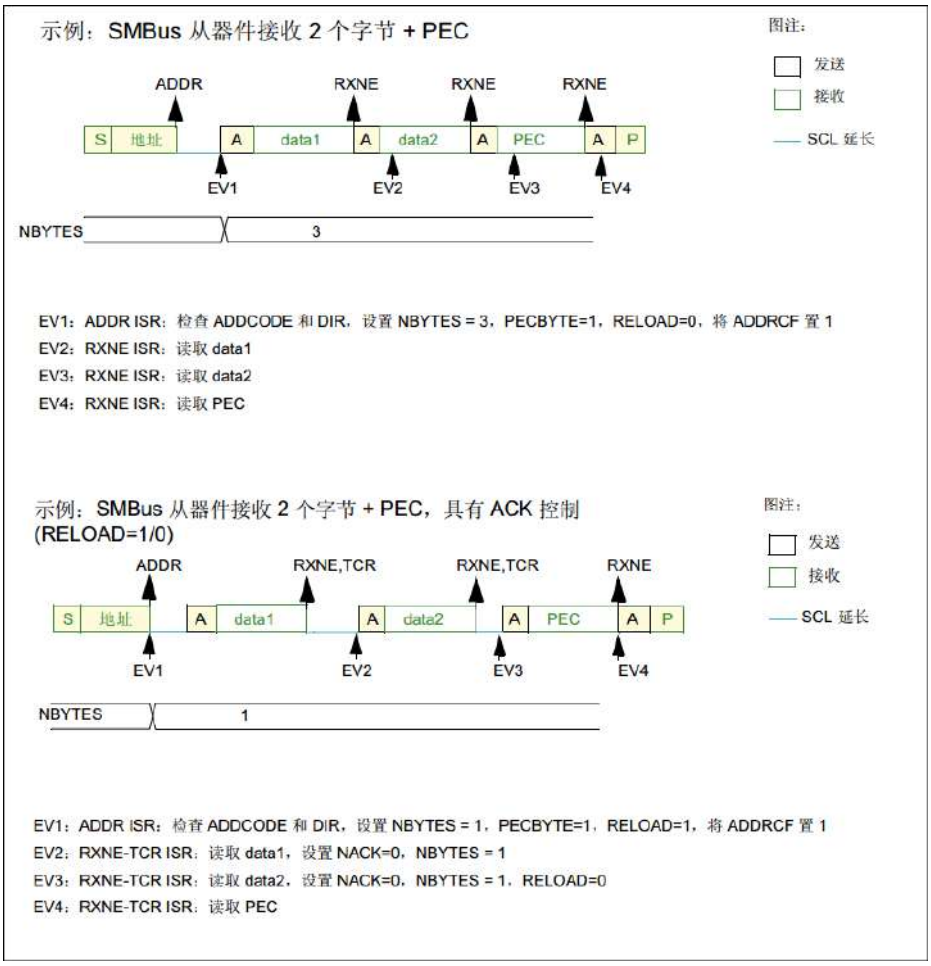


图 24-28 从接收器的总线传输图 **(SBC=1)**



仅当支持 SMBus 功能时，才涉及本节内容。请参见 *I2C* 特性实现。

除了 I2C 主模式传输管理（请参见 *I2C* 主模式）之外，还提供了一些额外的软件流程图来支持 SMBus。

SMBus 主发送器

当 SMBus 主器件想要发送 PEC 时，必须在 START 位置 1 前，将 PECBYTE 位置 1 并在 NBYTES[7:0] 字段中设置字节数。在这种情况下，总 TXIS 中断数为 NBYTES-1。因此，如果 PECBYTE 位在 NBYTES=0x1 时置 1，则将自动发送 I2C_PECR 寄存器的内容。

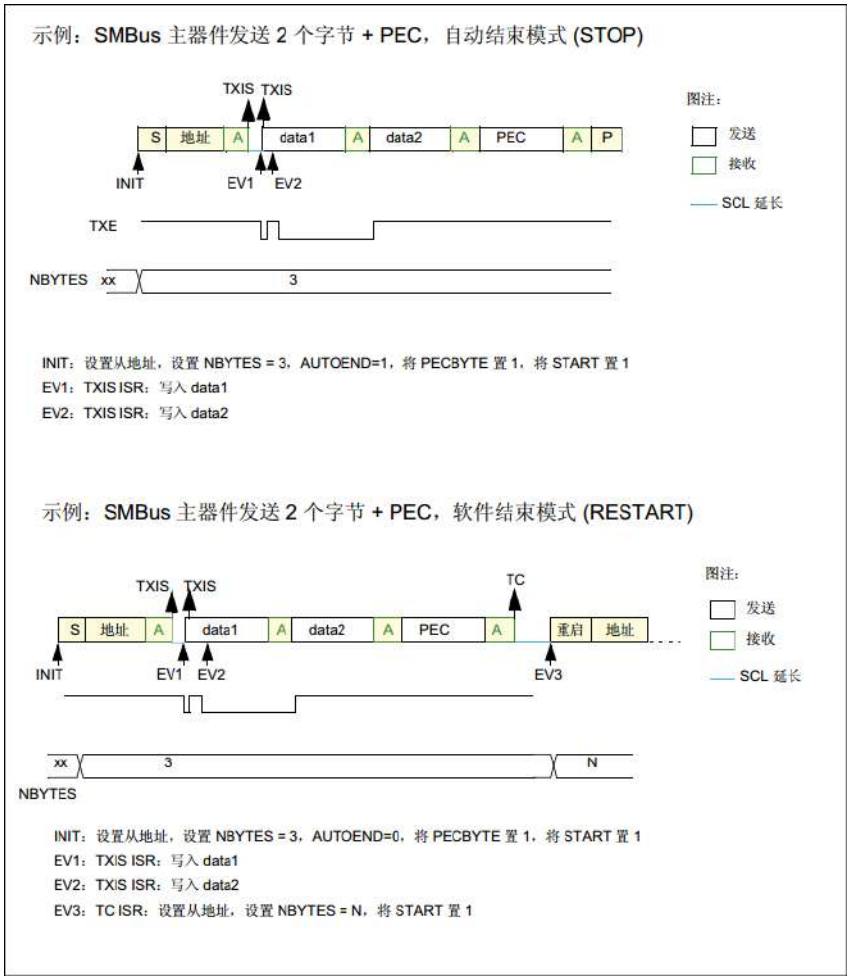
如果 SMBus 主器件想要在 PEC 后发送停止位，则应选择自动结束模式 (AUTOEND=1)。在这种情况下，传输 PEC 后将自动发送停止位。

如果 SMBus 主器件想要在 PEC 后发送重复起始位，则必须选择软件模式 (AUTOEND=0)。

在这种情况下，发送 NBYTES-1 字节的数据后，将发送 I2C_PECR 寄存器的内容， TC 标志将在传输完 PEC 之后置 1， SCL 线的低电平时间将延长。必须在 TC 中断子程序中设置重复起始位。

注意： 当 RELOAD 位置 1 时， PECBYTE 位将不起作用。

图 24-29 **SMBus** 主发送器的总线传输图



SMBus 主接收器

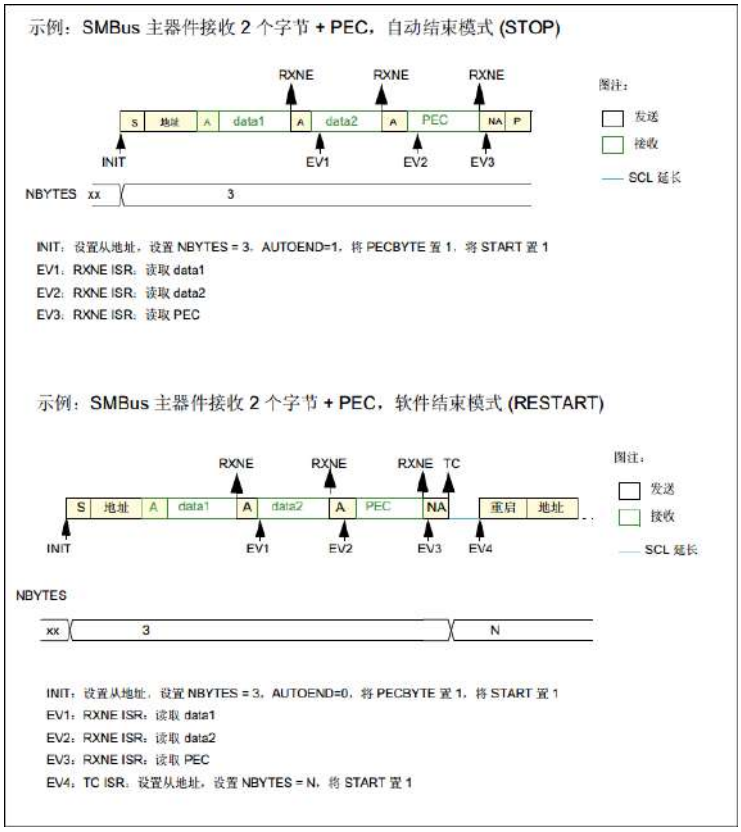
当 SMBus 主器件想要接收 PEC，并在传输结束后接收 STOP 时，可选择自动结束模式 (AUTOEND =1)。将 START 位置 1 之前，必须将 PECBYTE 位置 1 并设置从地址。在这种情况下，当接收到 NBY TES-1 字节的数据后，将自动使用 I2C_PECR 寄存器的内容对接收的下一个字节进行校验。 PEC 字节（其后跟有停止位）将得到 NACK 响应。

当 SMBus 主接收器想要接收 PEC 字节，并且在传输结束后接收重复起始位时，必须选择软件模式 (A UTOEND=0)。将 START 位置 1 之前，必须将 PECBYTE 位置 1 并设置从地址。

在这种情况下，当接收到 NBYTES-1 字节的数据后，将自动使用 I2C_PECR 寄存器的内容对接收的下一个字节进行校验。接收到 PEC 字节后， TC 标志将置 1， SCL 线的低电平时间将延长。可以在 TC 中断子程序中设置重复起始位。

注意： 当 RELOAD 位置 1 时， PECBYTE 位将不起作用。

图 24-30 **SMBus** 主接收器的总线传输图



24.3.14 地址匹配时从停止模式唤醒

仅当支持从停止模式唤醒功能时，才涉及本节内容。请参见： *I2C* 特性实现。

被寻址时， I2C 能够将 MCU 从停止模式唤醒（APB 时钟关断）。支持所有寻址模式。

将 I2C_CR1 寄存器中的 WUPEN 位置 1，可以使能从停止模式唤醒功能。对于 I2CCLK，必须选择 HSI 振荡器作为时钟源，以便从停止模式唤醒。

在停止模式中， HSI 关闭。当检测到 START 时， I2C 接口将 HSI 接通，并延长 SCL 低电平持续时间，直到 HSI 唤醒。HSI 随后用来接收地址。地址匹配的情况下， MCU 唤醒时间内， I2C 延长 SCL 的低电平持续时间。通过软件清除

ADDR 标志后，此延长操作被释放，传输正常进行。如果地址不匹配， HSI 再次关断， MCU 不被唤醒。

注： 如果 *I2C* 时钟是系统时钟，或者 *WUPEN = 0*，则接收到 *START* 后， *HSI* 振荡器不会接通。

只有 *ADDR* 中断能够唤醒 *MCU*。因此，当 *I2C* 以主器件身份或在 *ADDR* 标志置 *1* 后以被寻址从器件身份执行传输时，不要进入停止模式。将 *ADDR* 中断程序中的 *SLEEPDEEP* 位清零，然后仅在 *STOPF* 标志置 *1* 后将其置 *1*，可对此进行管理。

注意： 数字滤波器与从停止模式唤醒功能不兼容。如果 DNF 位不等于 0，则将 WUPEN 位置 1 将不起任何作用。

注意： 只有当 I2C 时钟源为 HSI 振荡器时，该功能才可用。

注意： 必须使能时钟延长 (NOSTRETCH=0) 才能确保从停止模式唤醒功能正常工作。

注意： 如果禁止从停止模式唤醒 (WUPEN=0)，则在进入停止模式前必须禁止 I2C 外设 (PE=0)。

24.3.15 错误条件

以下错误条件可能导致通信失败。

总线错误 (BERR)

当检测到起始位或停止位但不位于第 9N 个 SCL 时钟脉冲之后时，会检测到总线错误。当 SDA 边沿出现且 SCL 为高电平时，会检测到起始或停止位。

只有当 I2C 在传输过程中用作主器件或被寻址为从器件时（即未处于从模式下的地址阶段），才会将总线错误标志置 1。

在从模式下误检测到起始位或重复起始位时， I2C 会像接收到正确的起始位一样进入地址识别状态。

检测到总线错误时， I2C_ISR 寄存器中的 BERR 标志将置 1，如果 I2C_CR1 寄存器中的 ERRIE 位置 1，还将生成中断。

仲裁丢失 (ARLO)

当 SDA 线上发送高电平但在 SCL 上升沿却采样到低电平时，会检测到仲裁丢失。

- 在主模式下，将在地址阶段、数据阶段和数据应答阶段检测到仲裁丢失。在这种情况下， SDA 线和 SCL 线被释放，起始控制位由硬件清零，主器件自动切换为从模式。
- 在从模式下，将在数据阶段和数据应答阶段检测到仲裁丢失。在这种情况下，传输停止， SCL 和 SDA 线被释放。

检测到仲裁丢失时， I2C_ISR 寄存器中的 ARLO 标志将置 1，如果 I2C_CR1 寄存器中的 ERRIE 位置 1，还将生成中断。

上溢/下溢错误 (OVR)

当满足 NOSTRETCH=1 和以下条件时，将在从模式下检测到上溢或下溢错误：

- 在接收过程中接收到一个新的字节，但 RXDR 寄存器的值还未被读取。接收的新字节丢失，自动发送 NACK 来响应新字节。
- 在发送过程中：
  - 当 STOPF=1 且应发送第一个数据字节时。 TXE=0 时发送 I2C_TXDR 寄存器的内容，否则发送 0xFF。
  - 应发送一个新字节但尚未向 I2C_TXDR 寄存器写入数据时，将发送 0xFF。

检测到上溢或下溢错误时， I2C_ISR 寄存器中的 OVR 标志将置 1，如果 I2C_CR1 寄存器中的 ERRIE 位置 1，还将生成中断。

数据包错误校验错误 (PECERR)

仅当支持 SMBus 功能时，才涉及本节内容。请参见 *I2C 特性实现*。

当接收到的 PEC 字节与 I2C_PECR 寄存器的内容不匹配时，将检测到 PEC 错误。接收到错误的 PEC 后，将自动发送 NACK。

检测到 PEC 错误时， I2C_ISR 寄存器中的 PECERR 标志将置 1，如果 I2C_CR1 寄存器中的 ERRIE 位置 1，还将生成中断。

超时错误 (TIMEOUT)

仅当支持 SMBus 功能时，才涉及本节内容。请参见 *I2C 特性实现*。

满足以下任何条件均会出现超时错误：

- TIDLE=0 且 SCL 的低电平持续时间达到 TIMEOUTA[11:0] 位中定义的时间：这用于检测 SMBus 超时。
- TIDLE=1 且 SDA 和 SCL 的高电平持续时间达到 TIMEOUTA[11:0] 位中定义的时间：这用于检测总线空闲情况。
- 主器件的累积时钟低电平延长时间达到了 TIMEOUTB[11:0] 位中定义的时间（SMBus$_{LOW:MEXT}$ 参数）
- 从器件的累积时钟低电平延长时间达到了 TIMEOUTB[11:0] 位中定义的时间（SMBus$_{LOW:SEXT}$ 参数）

当在主模式下检测到超时时，将自动发送停止位。

当在从模式下检测到超时时，将自动释放 SDA 和 SCL 线。

检测到超时错误时， I2C_ISR 寄存器中的 TIMEOUT 标志将置 1，如果 I2C_CR1 寄存器中的 ERRIE 位置 1，还将生成中断。

报警 (ALERT)

仅当支持 SMBus 功能时，才涉及本节内容。请参见 *I2C* 特性实现。

当 I2C 接口配置为主机 (SMBHEN=1)、使能了报警引脚检测 (ALERTEN=1) 并且在 SMBA 引脚上检测到下降沿时， ALERT 标志将置 1。如果 I2C_CR1 寄存器中的 ERRIE 位置 1，将生成中断。

24.3.16 DMA 请求

使用 **DMA** 进行发送

将 I2C_CR1 寄存器中的 TXDMAEN 位置 1 可以使能 DMA（直接存储器访问）进行发送。当 TXIS 位置 1 时，数据将从由 DMA 外设配置的 SRAM 区（请参见直接存储器访问控制器 *(DMA)*）装载进 I2C_TXDR 寄存器。

只有数据字节采用 DMA 进行传输。

- 在主模式下：初始化、从地址、方向、字节数和起始位均由软件编程（发送的从地址无法通过 DMA 传输）。当所有数据均通过 DMA 传输时，必须在起始位置 1 之前初始化 DMA。传输结束由 NBYTES 计数器来管理。请参见 “主发送器” 。
- 在从模式下：

– 当 NOSTRETCH=0 时，如果所有数据均通过 DMA 传输，则必须在地址匹配事件之前（或清零 ADDR 之前在 ADDR 中断子程序中）初始化 DMA。

– 当 NOSTRETCH=1 时，必须在地址匹配事件之前初始化 DMA。

- 支持 SMBus 时： PEC 传输由 NBYTES 计数器管理。请参见 “*SMBus* 从发送器” 和 “*SMBus* 主发送器” 。

注： 如果使用 DMA 进行发送，则无需使能 TXIE 位。

使用 DMA 进行接收

将 I2C_CR1 寄存器中的 RXDMAEN 位置 1 可以使能 DMA（直接存储器访问）进行接收。

当 RXNE 位置 1 时，数据将从 I2C_RXDR 寄存器装载进由 DMA 外设配置的 SRAM 区（请参见直接存储器访问控制器 *(DMA)*）。只有数据字节（包括 PEC）采用 DMA 进行传输。

- 在主模式下：初始化、从地址、方向、字节数和起始位均由软件编程。当所有数据均通过 DMA 传输时，必须在起始位置 1 之前初始化 DMA。传输结束由 NBYTES 计数器来管理。
- 在从模式下，当 NOSTRETCH=0 时，如果所有数据均通过 DMA 传输，则必须在地址匹配事件之前（或清零 ADDR 标志之前在 ADDR 中断子程序中）初始化 DMA。
- 如果支持 SMBus（请参见 *I2C* 特性实现）： PEC 传输由 NBYTES 计数器管理。请参见 “*SMBus* 从接收器” 和第 *676* 页的“*SMBus* 主接收器” 。

注： 如果使用 DMA 进行接收，则无需使能 RXIE 位。

24.3.17 调试模式

当微控制器进入调试模式时（内核停止）， SMBus 超时定时器会根据 DBG 模块中的 DBG_I2Cx_SMBUS_TIMEOUT 配置位选择继续正常工作还是停止工作。

24.3.18 I2C 低功耗模式

表 24-12 低功耗模式

模式	说明

睡眠	对 I2C 通信无影响 I2C 中断可使器件退出睡眠模式。
停止	I2C 模块的寄存器内容仍被保持。
待机	I2C 外设掉电，退出待机模式后必须重新初始化。

24.3.19 I2C 中断

下表给出了 I2C 中断请求列表。

表 24-13 **I2C** 中断请求

中断事件	事件标志	事件标志/ 中断清除方法	中断使能控制位
接收缓冲区非空 (Receive buffer not empty)	RXNE	读取 I2C_RXDR 寄存器	RXIE
发送缓冲区中断状态	TXIS	写入 I2C_TXDR 寄存器	TXIE
停止位检测中断标志	STOPF	写入 STOPCF=1	STOPIE
传输完成等待重载	TCR	写入 I2C_CR2（NBYTES[7:0] ≠ 0）	TCIE
传输完成	TC	写入 START=1 或 STOP=1	
地址匹配	ADDR	写入 ADDRCF=1	ADDRIE
接收到 NACK 应答	NACKF	写入 NACKCF=1	NACKIE
总线错误 (Bus error)	BERR	写入 BERRCF=1	ERRIE
仲裁丢失	ARLO	写入 ARLOCF=1	
上溢/下溢 (Overrun/Underrun)	OVR	写入 OVRCF=1	
PEC 错误	PECERR	写入 PECERRCF=1	
超时/tLOW 错误	TIMEOUT	写入 TIMEOUTCF=1	
SMBus 报警	ALERT	写入 ALERTCF=1	

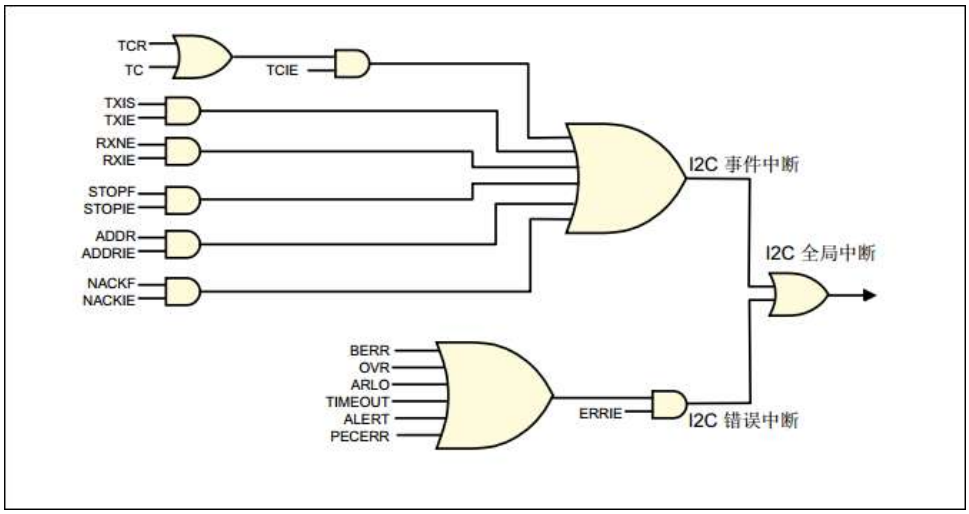
根据产品实现的不同，上述所有中断既可以共享同一个中断向量（I2C 全局中断），也可以分配到 2 个不同的中断向量上（I2C 事件中断和 I2C 错误中断）。

要使能 I2C 中断，需按照以下顺序操作：

1. 配置 NVIC 中的 I2C IRQ 通道并将其使能。
2. 配置 I2C 以生成中断。

I2C 唤醒事件连接到 EXTI 控制器（请参见 *EXTI* 寄存器）。

24-31 **I2C** 中断映射图



24.4 I2C 寄存器

24.4.1 控制寄存器 1（I2Cx_CR1）

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
Res.	Res	Res.	Res.	Res	Res	Res	Res	PECEN	ALERTEN	SMBDEN	SMBHEN	GCEN	WUPEN	NOSTRETCH	SBC
								rw	rw	rw	rw	rw		rw	rw
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXDMAEN	TXDMAEN	Res	ANFOFF	DNF				ERRIE	TCIE	STOPIE	NACKIE	ADDRIE	RXIE	TXIE	PE
rw	rw		rw	rw				rw	rw	rw	rw	rw	rw	rw	rw
0	0	0	0	0				0	0	0	0	0	0	0	0

位 31:24	保留，必须保持复位时的值。
位 23	PECEN： 启用 PEC 0: PEC 计算禁用 1: 启用 PEC 计算
位 22	ALERTEN： SMBus 通知使能 设备模式（SMBHEN = 0）： 0: 释放 SMBA 引脚为高，并禁用 NACK 之后的通知响应地址头： 0001100x。 1: 拉低 SMBA 引脚并启用 ACK 之后的通知响应地址头： 0001100x。 HOST 模式（SMBHEN = 1）： 0: SMBus 通知引脚（SMBA）不支持。 1: 支持 SMBus 通知引脚（SMBA）。
位 21	SMBDEN： SMBus 器件的默认地址启用 0: 禁用设备的默认地址。 地址 0b1100001x 会被 NACK。 1: 启用设备的默认地址。 地址 0b1100001x 会被 ACK。

位 20	SMBHEN： SMBus HOST 地址启用 0: 禁用 HOST 地址。 地址 0b0001000x 会被 NACK。 1: 启用 HOST 地址。 地址 0b0001000x 会被 ACK。
位 19	GCEN: 广播呼叫地址使能 0: 禁止广播呼叫。 地址 0b00000000 会被 NACK。 1: 启用广播呼叫。 地址 0b00000000 会被 ACK。
位 18	WUPEN： 从 STOP 模式唤醒 0: 禁止从 STOP 唤醒 1: 允许从 STOP 模式唤醒
位 17	NOSTRETCH： 禁止时钟延长 该位用于在从机模式中禁止时钟延长。 0: 允许时钟延长 1: 禁止时钟延长 该位用于在从机模式使能硬件字节控制。 0: 从机字节控制模式关闭 1: 从机字节控制模式使能
位 16	SBC： 从机字节控制置 1 时，I2C 的 SCL 和 SDA 线都被释放。 内部状态机和所有的状态为回到其复位值。控制位的内容被保留。
位 15	RXDMAEN：DMA 接收请求使能 0: 关闭 DMA 接收请求 1: 开启 DMA 接收请求
位 14	TXDMAEN： DMA 发送请求使能 0: 关闭 DMA 发送功能 1: 开启 DMA 发送功能
位 13	SWRST: 软件复位
位 12	ANFOFF： 模拟噪声滤波器关闭 0: 模拟噪声滤波器开启 1: 模拟噪声滤波器关闭
位 11：8	DNF[3:0]： 数字噪声滤波器 这些位用来配置 SDA 和 SCL 输入上的数字噪声滤波器。 数字滤波器会用 DNF[3:0]* tI2C_CLK 的长度来工作。 0000: 数字滤波器禁用 0001: 数字滤波器启用，并且滤波能力达到 1 个 tI2C_CLK …… 1111: 数字滤波器启用，其滤波能力达到 15 个 tI2C_CLK
位 7	ERRIE： 错误中断使能 0: 错误检测中断禁止 1: 错误检测中断使能
位 6	TCIE：传输完成中断使能 0: 传输完成中断禁用 1: 传输完成中断使能
位 5	STOPIE： STOP 检测中断使能 0: 停止检测（STOPF）中断禁止 1: 停止检测（STOPF）中断使能
位 4	收到 NACK 中断使能 0: （NACKF）收到中断禁用 1: （NACKF）收到中断允许
位 3	地址匹配中断使能（仅从机） 0: 地址匹配（ADDR）中断禁用 1: 启用地址匹配（ADDR）中断
位 2	接收中断使能 0: 接收（RXNE）中断禁止 1: 启用接收（RXNE）中断
位 1	TXIE： Tx 中断使能 0: 发送（TXIS）中断禁止 1: 发送（TXIS）中断使能

位 0	PE： 外设使能 0: 外设禁用 1: 外设使能

24.4.2 控制寄存器 2（I2Cx_CR2）

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
Res.	Res.	Res.	Res.	Res.	PECBYTE	AUTOEND	RELOAD	NBYTES[7:0]							
					rs	rw	rw	rw							
0	0	0	0	0	0	0	0	0							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NACK	STOP	START	HEAD10R	ADD10	RD_WRN	SADD[9:0]									
rs	rs	rs	rw	rw	rw	rw									
0	0	0	0	0	0	0									

位 31:27	保留，必须保持复位时的值。
位 26	PECBYTE： 包错误检查字节 这个位由软件置位，在 PEC 传输完后由硬件清零，或者在收到 STOP 条件后，在收到地址匹配事件后以及在 PE=0 或者 SWRST 被写为 1 的时候都会被硬件清零。 0: 没有 PEC 传送。 1: 要求发送/接收 PEC
位 25	AUTOEND：自动结束模式（主机模式）此位由软件置 1 和清零。 0: 软件结束模式： 当 NBYTES 个数据传输完毕后，TC 标志被置 1，SCL 被拉低。 1: 自动结束模式： NBYTES 个数据传输完后，会自动发送一个停止条件。
位 24	RELOAD： NBYTES 重装载模式 由软件设置和清除。 0: 在传输完 NBYTES 个字节后， 传输结束。 1: 传输 NBYTES 个字节后，传输并不结束（NBYTES 将被重装载）。 当 NBYTES 个数据传输完毕后，TC 标志被置 1，SCL 被拉低。
位 23:16	NBYTES[7:0]： 字节数 这里写入要发送 / 接收的字节数。 从机模式并且 SBC=0 的时候，这个地方的值无 所谓。
位 15	NACK： 产生 NACK （从机模式） 这个位由软件置位，在 NACK 发送后由硬件清零，或者在收到 STOP 条件后，在收 到地址匹配事件后以及在 PE=0 或者 SWRST 被写为 1 的时候都会被硬件清零。 0: 在当前字节收到后发送 ACK。 1: 当前字节接收到后发送一个 NACK。
位 14	STOP： 产生停止条件（主机模式） 该位由软件置 1,在检测到 STOP 条件或者 PE=0 或者 SWRST 被置 1 时由硬件清零。在主机模式下： 0: 不产生 STOP 条件。 1: 当前字节传输完后产生停止条件。
位 13	START： 产生起始条件 该位通过软件设置，在发送完一个起始条件和地址序列之后由硬件清零，或者由于 仲裁丢失、超时错误、PE=0 以及 SWRST 被置 1 等事件由硬件清零。 这个位也可 以由软件向 I2Cx_ICR 寄存器的 ADDRCF 位写 1 来清零。 0: 没有起始条件产生 1: 产生 START/RESTART 条件： －如果 I2C 已经是在主机模式下并且 AUTOEND = 0，RELOAD=0，并且 NBYTES 个字节发送完毕后设置此位会产生重复起始条件。 －否则只要总线空闲，设置此位将会立即产生一个起始条件。

位 12	HEAD10R： 10 位地址头只读方向（主接收器模式） 0: 主机发送完整的 10 位从机地址读序列： 起始 +2 字节 10 位地址（写方向）+ 重新起始 +10 位地址中的前 7 位（读方向）。 1: 主机只发送 10 位地址的前 7 位，跟着是读方向。
位 11	ADD10： 10 位地址模式（主机模式） 0: 主机按 7 位地址模式操作， 1: 主机按 10 位地址模式操作
位 10	RD_WRN： 传输方向（主机模式） 0: 主机请求一个写传输。 1: 主机请求一个读传输。
位 9:8	SADD [9:8]： 从机地址位 9:8（主机模式） 在 7 位地址模式下（ADD10 = 0）： 不用管这些个位 在 10 位地址模式下（ADD10 = 1）：这些位应写入要发送的从机地址位的 9:8
位 7:1	SADD [7:1]： 从机地址位 7:1（主机模式） 在 7 位地址模式下（ADD10 = 0）：这些位应写入要发送的 7 位从机地址位；在 10 位地址模式下（ADD10 = 1）： 这些位应写入要发送的从机地址位的 7:1
位 0	SADD0：从机地址的 0 位（主模式） 在 7 位地址模式下（ADD10 = 0）：不用管这个位；在 10 位地址模式下（ADD10=1）：这些位应写入要发送的从机地址位的位 0

24.4.3 本机地址 1 寄存器（I2Cx_OAR1）

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
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OA1EN	Res.	Res.	Res.	Res.	OA1MODE	OA1[9:8]		OA1[7:1]							OA1[0]
rw					rw	rw		rw							rw
0	0	0	0	0	0	0	0	0							0

位 31:16	保留，必须保持复位时的值。
位 15	OA1EN： 本机地址 1 启用 0: 禁用本机地址 1。接收到从机地址 OA1 后会用 NACK 回应。 1: 本机地址 1 启用 接收到从机地址 OA1 后会用 ACK 回应。
位 14:11	保留，必须保 持复位时的值。
位 10	OA1MODE 本机地址 1 的 10 位模式 0: 本机地址 1 是 7 位地址。 1: 本机地址 1 是一个 10 位的地址。
位 9:8	OA1[9:8]： 接口地址的 9:8 位 7 位地址模式：无所谓 10 位地址模式：地址的位 9:8
位 7:1	OA1[7:1]： 接口地址的 7:1
位 0	OA1 的[0]：接口地址的 0 位 7 位地址模式：无所谓 10 位地址模式：地址的 0 位

24.4.4 本机地址 2 寄存器（I2Cx_OAR2）

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
Res.	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OA2EN	Res	Res	Res	Res	OA2MSK[2:0]			OA2[7:1]							Res
rw					rw			rw							
0	0	0	0	0	0			0							0

位 31:16	保留，必须保持复位时的值。
位 15	OA2EN： 本机地址 2 启用 0: 禁用本机地址 2。 接收到从机地址 OA2 后会用 NACK 回应。 1: 本机地址 2 启用 接收到从机地址 OA2 后会用 ACK 回应。
位 14:11	保留，必须保 持复位时的值。
位 10:8	OA2MSK [2:0]： 本机地址 2 屏蔽 000: 没有屏蔽 001: OA2 [1] 被屏蔽掉，可忽略。 只有 OA2 [7:2] 参与比较。 010: OA2 [2:1] 被屏蔽掉，可忽略。 只有 OA2 [7:3] 参与比较。 011: OA2 [3:1] 被屏蔽掉，可忽略。 只有 OA2 [7:4] 参与比较。 100: OA2 [4:1] 被屏蔽掉，可忽略。 只有 OA2 [7:5] 参与比较。 101: OA2 [5:1] 被屏蔽掉，可忽略。 只有 OA2 [7:6] 参与比较。 110: OA2 [6:1] 被屏蔽掉，可忽略。 只有 OA2 [7] 参与比较。 111: OA2 [7:1] 被屏蔽掉，可忽略。 没有比较，所有的（除保留地址）收到的 7 位 地址都会用 ACK 回应。
位 7:1	OA2 [7:1]： 接口地址位 7:1
位 0	保留，必须保持复位时的值。

24.4.5 时序寄存器（I2Cx_TIMINGR）

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
PRESC[3:0]				Res	Res	Res	Res	SCLDEL[3:0]				SDADEL[3:0]			
rw								rw				rw			
0				0	0	0	0	0				0			

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

位 31:28	PRESC [3:0]： 时序预分频器
位 27:24	保留，必须保持复位时的值。
位 23:20	SCLDEL [3:0]：数据建立时间 此字段用于生成发送模式中 SDA 的沿和 SCL 上升沿之间的延迟 tSCLDEL。tSCLDEL = (SCLDEL+1) x tPRESC

位 19:16	SDADEL [3:0]：数据保持时间，此字段用于生成发送模式中 SCL 的下降沿和 SDA 的沿之间的延迟 tSDADEL。tSDADEL= SDADEL x tPRESC
位 15:8	SCLH[7:0]：SCL 高电平时间（主机模式） 此字段用于在主机模式下产生 SCL 的高电平时间。tSCLH = (SCLH+1) x tPRESC
位 7:0	SCLL [7:0]：SCL 低电平时间（主机模式） 此字段用于在主机模式下产生 SCL 的低电平时间。tSCLL = (SCLL+1) x tPRESC

24.4.6 超时寄存器（I2Cx_TIMEOUTR）

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
TEXTEN	Res.	Res.	Res.	TIMEOUTB											
rw				rw											
0	0	0	0	0											

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TIMEOUTEN	Res.	Res.	TIDLE	TIMEOUTA											
rw			rw	rw											
0	0	0	0	0											

位 31	TEXTEN： 外部时钟超时启用 0: 外部时钟超时检测被禁用 1: 外部时钟超时检测启用。
位 30:29	保留，必须保持复位时的值。
位 27:16	TIMEOUTB [11:0]： 总线超时 B 此字段用于配置累计时钟延长超时： 在主机模式下，要检测的累计主机时钟低延长时间 (tLOW:MEXT)。 在从机模式下，要检测的累积从机时钟低延长时间 (tLOW:SEXT)。 tTLOW:EXT= (TIMEOUTB+1) x 2048 x tI2C_CLK
位 15	TIMOUTEN： 时钟超时检测启用 0: SCL 超时检测被禁用 1: 启用 SCL 超时检测： 当 SCL 保持低的时间超过 tTIMEOUT (TIDLE=0) 或保持高的时 间超过 tIDLE (TIDLE=1)，会检测到一个超时错误 (TIMEOUT=1).
位 14:13	保留，必须保持复位时的值。
位 12	TIDLE： 空闲时钟超时检测 0: TIMEOUTA 用于检测 SCL 低电平超时 1: TIMEOUTA 用于同时检测 SCL 和 SDA 高电平超时（总线空闲条件）
位 11:0	TIMEOUTA [11:0]：总线超时 A 此字段用于配置： — 在 TIDLE=0 的时候，SCL 低超时条件 tTIMEOUT tTIMEOUT= (TIMEOUTA+1) x 2048 x tI2C_CLK — TIDLE=1 的时候，总线空闲条件 (SCL 和 SDA 同时为高) tIDLE= (TIMEOUTA+1) x4 x tI2C_CLK

24.4.7 中断和状态寄存器（I2Cx_ISR）

地址偏移: 0x18

复位值: 0x0000 0001

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

Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADDCODE[6:0]	DIR
								r	r
0	0	0	0	0	0	0	0	0	0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BUSY	Res.	ALERT	TIMEOUT	PECERR	OVR	ARLO	BERR	TCR	TC	STOPF	NACKF	ADDR	RXNE	TXIS	TXE
r	r	r	r	r	r	r	r	r	r	r	r	r	r	rs	rs
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

位 31:24	保留，必须保持复位时的值。
位 23:17	ADDCODE[6:0]： 匹配的地址码（从机模式） 这些位由地址匹配事件发生时所接收到的地址（ADDR= 1）来更新。在 10 位地址的情况下，ADDCODE 提供 10 位地址的头 2 位以后的地址。
位 16	DIR： 传输方向（从机模式） 此标志在地址匹配事件发生时（ADDR= 1）更新。 0: 写传输，从机进入接收模式。 1: 读传输，从机进入发送模式。
位 15	BUSY： 总线忙
位 14	保留，必须保持复位时的值。
位 13	ALERT： SMBus 通知 该标志在 SMBHEN=1（SMBus HOST 配置）及 ALERTEN=1 的条件下，在 SMBA 脚上检测到 SMBALERT 事件（下降沿）时，由硬件置 1。通过设置 ALERTCF 位，由软件清零。
位 12	TIMEOUT： 超时或 tLOW 检测标志 在超时或外部时钟超时发生时，该标志被硬件置 1。 通过设置 TIMEOUTCF 位，由软件清零。
位 11	PECERR： 接收中的 PEC 错误 该标志在收到的 PEC 值和 PEC 寄存器的内容不匹配时，由硬件置 1。收到错误的 PEC 后，会自动发送一个 NACK。这个位可以通过将 PECCF 位置 1，由软件清零。
位 10	OVR： 溢出 / 欠载 (从机模式) 此标志在从机模式下 NOSTRETCH = 1 时，发生溢出 / 欠载错误的时候，由硬件置 1。 通过设置 OVRCF 位，由软件清零。
位 9	ARLO： 仲裁丢失 该标志在总线仲裁丢失的情况下由硬件置 1。 通过设置 ARLOCF 位，由软件清零。
位 8	BERR： 总线错误 此标志在检测到错位的起始或者停止条件的时候由硬件置 1。通过设置 BERRCF 位， 由软件清零。
位 7	TCR： 传输完成重加载 该标志在 RELOAD=1， 并且 NBYTES 个数据发送完毕后， 由硬件置 1。 向 NBYTES 写入是一个非零的值时，TCR 标志由软件清除。
位 6	TC： 发送完毕（主机模式） 该标志在 RELOAD=0，AUTOEND=0，并且 NBYTES 个数据发送完毕后，由硬件置 1。 它在软件将 START 或 STOP 位置 1 的时候清零。
位 5	STOPF： 停止检测标志 该标志在外设参与传输的下列情况下，在总线上检测到一个停止条件时，由硬件置 1： — 作为主机，如果由外设生成一个停止条件的时候。 — 或作为从机，如果外设在这次传输之前被正确的寻址到了。 这个位可以通过将 STOPCF 位置 1，由软件清零。
位 4	NACKF： 收到 NACK 标志 该标志在一个字节传输后收到一个 NACK 的时候由硬件设置。 这个位可以通过将 NACKCF 位置 1，由软件清零。
位 3	ADDR： 地址匹配（从机模式） 这个位在收到的从机地址与其中一个有效的从机地址匹配的时候，由硬件置 1。 通 过设置 ADDRCF 位，由软件清零。
位 2	RXNE： 接收数据寄存器非空（接收） 此位在当接收到的数据被复制到 I2Cx_RXDR 寄存器，准备好被软件读取的时候由 硬件置位。 在读取 I2Cx_RXDR 时 RXNE 会被清除。

位 1	TXIS：发送中断状态（发送） 在 I2Cx_TXDR 寄存器为空的时候由硬件置 1，这时必须把要发的数据写到 I2Cx_TXDR 寄存器。下一个要发送的数据被写到 I2Cx_TXDR 寄存器的时候它会被清除。该位只在 NOSTRETCH = 1 的时候可以由软件写成 1，以生成一个 TXIS 事件（如果 TXIE =1 就有中断，如果 TXDMAEN=1 就有 DMA 请求）。
位 0	TXE：发送数据寄存器空（发送） 在 I2Cx_TXDR 寄存器为空的时候由硬件置 1。下一个要发送的数据被写到 I2Cx_TXDR 寄存器的时候它会被清除。 该位可通过软件写 1，以清空发送数据寄存器 I2Cx_TXDR。

24.4.8 中断清除寄存器（I2Cx_ICR）

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
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	ALERTCF	TIMOUTCF	PECCF	OVERCF	ARLOCF	BERRCF	Res.	Res.	STOPCF	NACKCF	ADDRCF	Res.	Res.	Res.
		w	w	w	w	w	w			w	w	w			
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位 31:14	保留，必须保持复位时的值。
位 13	ALERTCF： 通知标志清零 对这个位写 1，会清除 I2Cx_ISR 寄存器中的 ALERT 标志位。
位 12	TIMOUTCF： 超时检测标志清除 对这个位写 1，会清除 I2Cx_ISR 寄存器中的 TIMEOUT 标志位。
位 11	PECCF： PEC 错误标志清除 对这个位写 1，会清除 I2Cx_ISR 寄存器中的 PECERR 标志位。
位 10	OVRCF： 溢出 / 欠载标志清除 对这个位写 1，会清除 I2Cx_ISR 寄存器中的 OVR 标志位。
位 9	ARLOCF： 仲裁丢失标志清除 对这个位写 1，会清除 I2Cx_ISR 寄存器中的 ARLO 标志位。
位 8	BERRCF： 总线错误标志清除 对这个位写 1，会清除 I2Cx_ISR 寄存器中的 BERRF 标志位。
位 7:6	保留，必须保 持复位时的值。
位 5	STOPCF： 停止检测标志清除 对这个位写 1，会清除 I2Cx_ISR 寄存器中的 STOPF 标志位。
位 4	NACKCF： 收到 NACK 标志清除 对这个位写 1，会清除 I2Cx_ISR 寄存器中的 NACKF 标志位。
位 3	ADDRCF： 地址匹配标志清除 对这个位写 1，会清除 I2Cx_ISR 寄存器中的 ADDR 标志位。 对这个位写 1，还会 清除 I2C_CR2 寄存器中的 START 位。
位 2:0	保留，必须保持复位时的值。

24.4.9 PEC 寄存器（I2Cx_PECR）

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
Res.	Res	Res.	Res.	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

位31:8	保留，必须保持复位时的值。
位7:0	PEC [7:0] 包错误检查寄存器 当 PECEN = 1 时，此字段包含内部 PEC 结果。PEC 在 PE = 0 或 SWRST 被置 1 时，由硬件清零。

24.4.10 接收数据寄存器（I2Cx_RXDR）

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
Res.	Res	Res.	Res.	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

位31:8	保留，必须保持复位时的值。
位7:0	RXDATA [7:0] 8 位接收数据，从 I2C 总线接收的数据字节。

24.4.11 接收数据寄存器（I2Cx_TXDR）

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
Res.	Res	Res.	Res.	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

位31:8	保留，必须保持复位时的值。
位7:0	TXDATA [7:0] 8 位发送数据，发送到 I2C 总线的数据字节。

25 通用同步异步收发器 (USART)

25.1 简介

通用同步异步收发器 (USART) 能够灵活地与外部设备进行全双工数据交换，满足外部备对工业标准 NRZ 异步串行数据格式的要求。 USART 通过可编程波特率发生器提供了多种波特率。

它支持同步单向通信、半双工单线通信以及多处理器通信； 还支持 LIN（局域互连网络）、智能卡协议与 IrDA（红外线数据协会） SIR ENDEC 规范，以及调制解调器操作(CTS/RTS)。通过配置多个缓冲区使用 DMA（直接存储器访问）可实现高速数据通信。

25.2 USART 主要特性

- 全双工异步通信
- NRZ 标准格式（标记/空格）
- 可配置为 16 倍过采样或 8 倍过采样，因而为速度容差与时钟容差的灵活配置提供了可能
- 32 MHz 时钟频率和 8 倍过采样时，通用可编程收发波特率最高为 4 Mbps
- 双时钟域允许：
  - USART 功能和从停止模式唤醒
  - 方便的波特率编程，独立于 PCLK 重新编程
- 自动波特率检测
- 数据字长度可编程（7 位、 8 位或 9 位）
- 可编程的数据顺序，最先移位 MSB 或 LSB
- 停止位可配置（支持 1 个或 2 个停止位）
- 用于同步通信的同步模式和时钟输出
- 单线半双工通信
- 使用 DMA 实现连续通信
- 使用中央 DMA 在预留的 SRAM 缓冲区中收/发字节
- 发射器和接收器有单独的使能位
- 发送和接收的单独信号极性控制
- Tx/Rx 引脚配置可交换
- 调制解调器和 RS-485 收发器的硬件流控制
- 通信控制/错误检测标志
- 奇偶校验控制：
  - 发送奇偶校验位
  - 检查接收的数据字节的奇偶性
- 十四个具有标志位的中断源
- 多处理器通信

如果地址不匹配， USART 将进入静默模式。

- 从静默模式唤醒（通过空闲线检测或地址标记检测）

25.3 USART 扩展特性

- LIN 主模式同步断路发送功能和 LIN 从模式断路检测功能
  - 当 USART 被配置为 LIN 使用时，可生成 13 位断路和检测 10/11 位断路
- 正常模式下支持 3/16 位持续时间的 IrDA SIR 编解码器
- 智能卡模式
  - 对于 ISO/IEC 7816-3 标准中定义的智能卡，支持 T=0 和 T=1 异步协议
  - 智能卡工作模式下，支持 0.5 或 1.5 个停止位
- 支持 ModBus 通信
  - 超时功能
  - CR/LF 字符识别

25.4 USART 实现

表 25-1 **HK32L08x USART** 特性

USART 模式/特性(1)	USART1/2	UART4/5	LPUART1
调制解调器的硬件流控	X	X	X
使用 DMA 进行连续通信	X	X	X
多处理器通信	X	X	X
同步模式	X	-	-
智能卡模式	X	-	-
单线半双工通信	X	X	X
Ir SIR ENDEC 模块	X	-	-
LIN 模式	X	-	-
双时钟域和从停止模式唤醒	X	-	X
接收器超时中断	X	-	-
Modbus 通信	X	-	-
自动波特率检测	X	-	-
驱动器使能	X	X	X
USART 数据长度	7 位、 8 位和 9 位		

1. X = 支持

25.5 USART 功能说明

任何 USART 双向通信均需要至少两个引脚：接收数据输入引脚 (RX) 和发送数据输出引脚 (TX)：

- **RX**：接收数据输入引脚。
  串行数据输入引脚。过采样技术可区分有效输入数据和噪声，从而用于恢复数据。
- **TX**：发送数据输出引脚。
  如果关闭发送器，该输出引脚模式由其 I/O 端口配置决定。如果使能了发送器但没有待发送的数据，则 TX 引脚处于高电平。在单线和智能卡模式下，该 I/O 用于发送和接收数据。

正常 USART 模式下，通过这些引脚发送和接收串行数据。这些帧包括：

- 发送或接收前保持空闲线路
- 起始位
- 数据字（字长 7 位、 8 位或 9 位），最低有效位在前
- 用于指示帧传输已完成的 0.5 个、 1 个、 1.5 个、 2 个停止位
- USART 接口使用波特率发生器

- 状态寄存器 (USARTx_ISR)
- 接收和发送数据寄存器（USARTx_RDR、 USARTx_TDR）
- 波特率寄存器 (USARTx_BRR)
- 智能卡模式下的保护时间寄存器 (USARTx_GTPR)。

有关各个位的定义，请参见 *USART* 寄存器。

在同步模式和智能卡模式下连接时需要以下引脚：

- **CK**： 时钟输出。该引脚用于输出发送器数据时钟，以便按照 SPI 主模式进行同步发送（起始位和结束位上无时钟脉冲，可通过软件向最后一个数据位发送时钟脉冲）。 RX 上可同步接收并行数据。这一点可用于控制带移位寄存器的外设（例如 LCD 驱动器）。时钟相位和极性可通过软件编程。在智能卡模式下， CK 输出可向智能卡提供时钟。

在 RS232 硬件流控制模式下需要以下引脚：

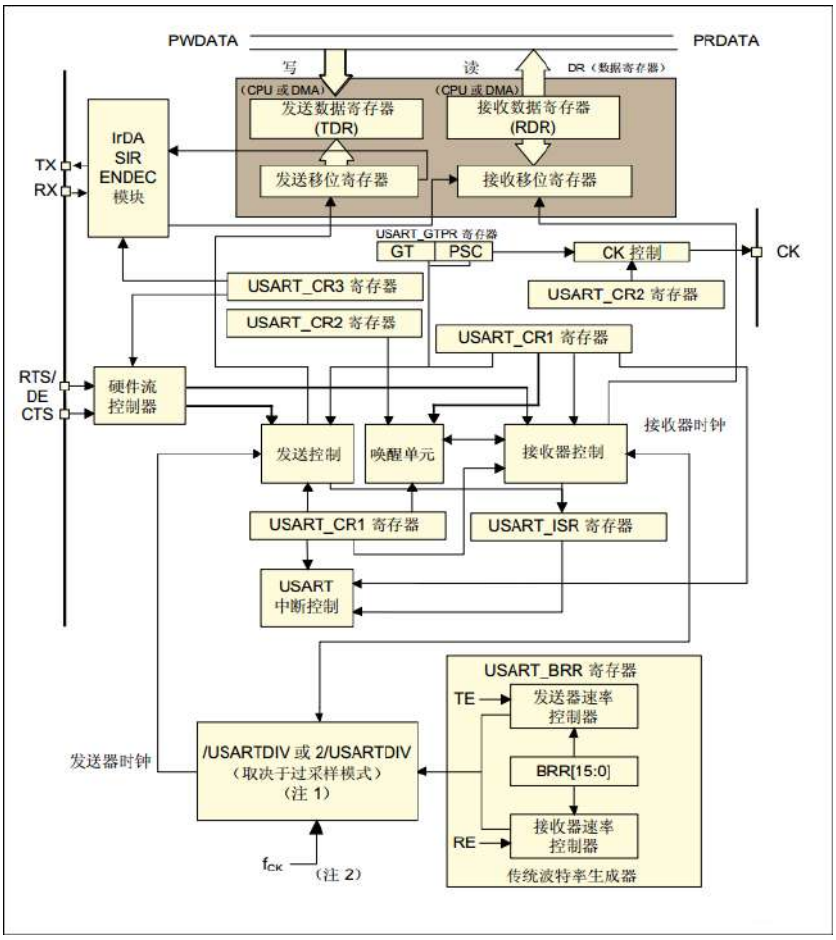
- **CTS**：“清除以发送”用于在当前传输结束时阻止数据发送（高电平时）
- **RTS**：“请求以发送”用于指示 USART 已准备好接收数据（低电平时）

在 RS485 硬件控制模式下需要以下引脚：

- **DE**：“驱动器使能”用于激活外部收发器的发送模式。

注： DE 和 RTS 共用同一个引脚。

图 25-1 **USART** 框图



1. 有关在 USARTx_BRR 寄存器中编码 USARTDIV 的详细信息，请参见第 *29.5.4* 节： *USART* 波特率生成。
2. fCK 可以是 fLSE、 fHSI、 fPCLK、 fSYS。

25.5.1 USART 字符说明

可通过对 USARTx_CR1 寄存器中的 M[1:0] 位进行编程来选择 7 位、 8 位或 9 位的字长（请参见图 25-2）。

- 7 位字符长度： M[1:0] = 10
- 8 位字符长度： M[1:0] = 00
- 9 位字符长度： M[1:0] = 01

注： 7 位数据长度模式下，不支持智能卡模式、 LIN 主模式和自动波特率（0x7F 帧和 0x55 帧检测）。只有某些 USART 支持 7 位模式。

在默认情况下，信号（TX 或 RX）在起始位工作期间处于低电平状态。在停止位工作期间处于高电平状态。

通过极性配置控制，可以单独针对每个信号对这些值取反。

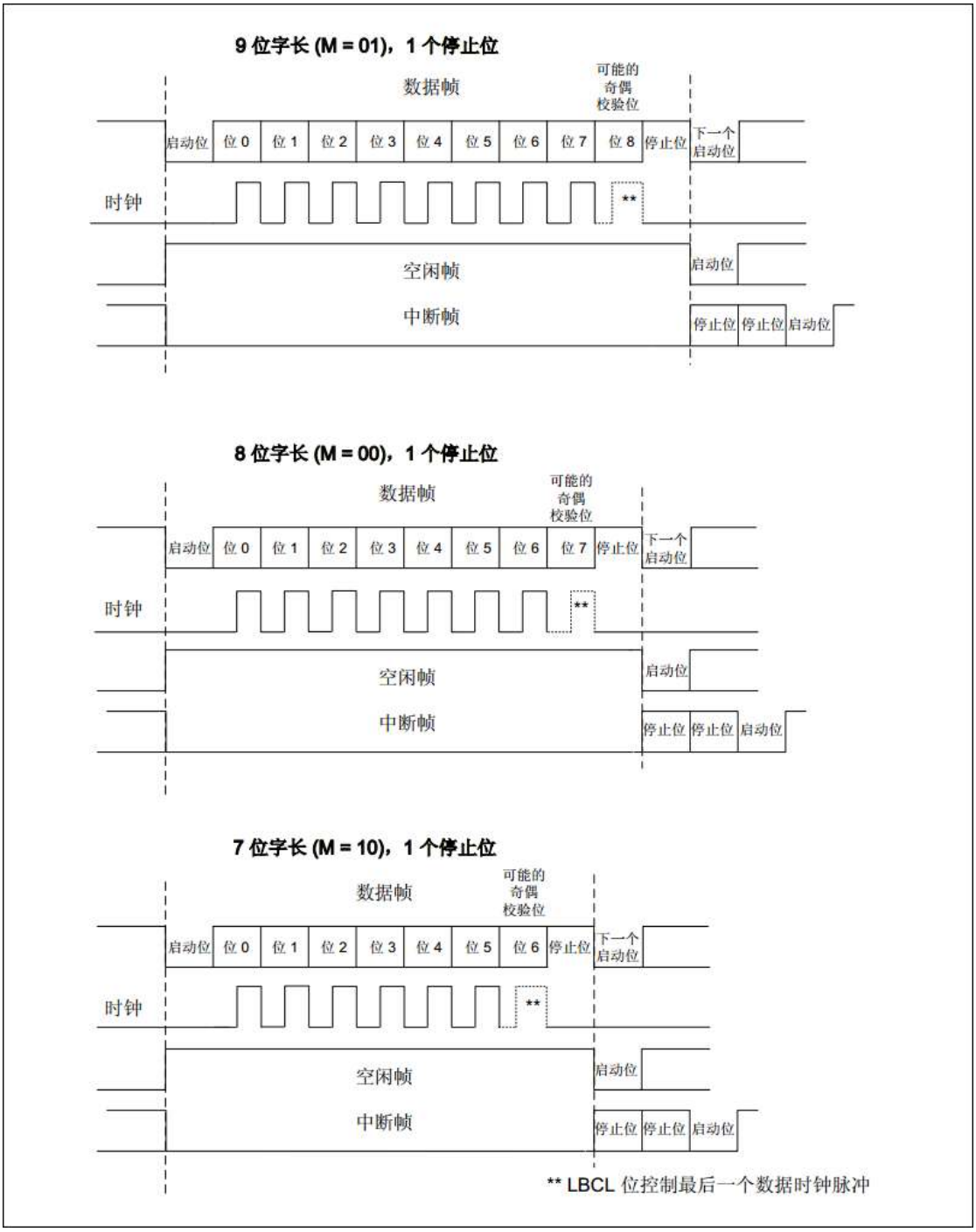
空闲字符可理解为整个帧周期内电平均为“1”（停止位的电平也是“1”）。

停止字符可理解为在一个帧周期内接收到的电平均为“0”。发送器在中断帧的末尾插入 2 个停止位。

发送和接收由通用波特率发生器驱动，发送器和接收器的使能位分别置 1 时将生成相应的发送时钟和接收时钟。

下面给出了各个块的详细信息。

图 25-2 字长编程



25.5.2 USART 发送器

发送器可发送 7 位、 8 位或 9 位的数据字，具体取决于 M 位的状态。要激活发送器功能，必须将发送使能位 (TE) 置 1。发送移位寄存器中的数据在 TX 引脚输出，相应的时钟脉冲在 CK 引脚输出。

字符发送

USART 发送期间，首先通过 TX 引脚移出数据的最低有效位（默认配置）。该模式下， USARTx_TDR 寄存器的缓冲区 (TDR) 位于内部总线和发送移位寄存器之间。

每个字符前面都有一个起始位，其逻辑电平在一个位周期内为低电平。字符由可配置数量的停止位终止。

USART 支持以下停止位： 0.5、 1、 1.5 和 2 个停止位。

注： 向 *USARTx_TDR* 中写入要发送的数据前，*TE* 位必须先置 *1*。

数据发送期间不应复位 *TE* 位。发送期间复位 *TE* 位会冻结波特率计数器，从而将损坏 *TX* 引脚上的数据。当前传输的数据将会丢失。

使能 *TE* 位后，将会发送空闲帧。

可配置的停止位

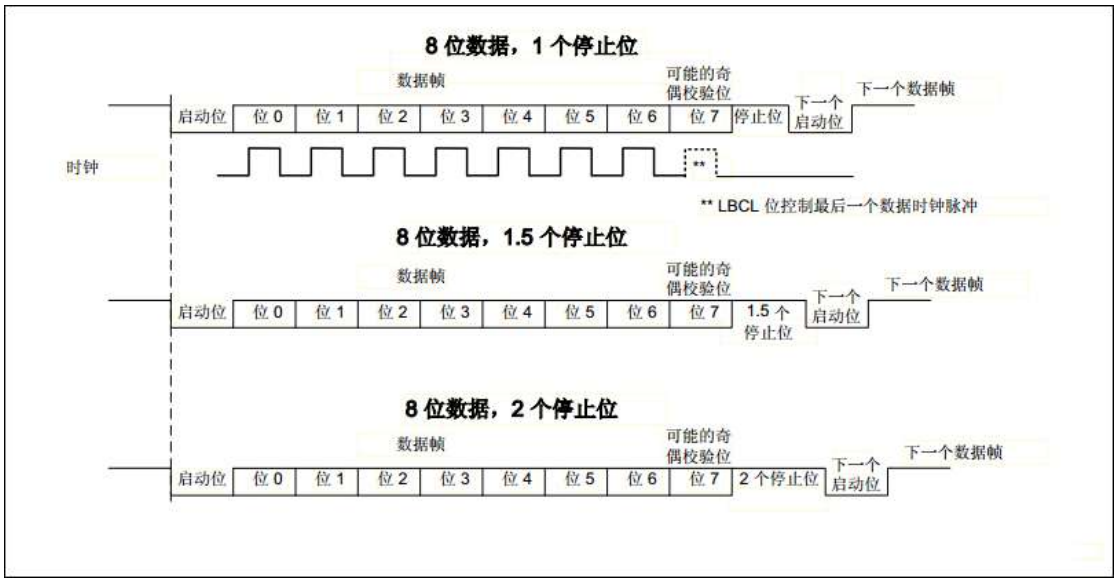
可以在控制寄存器 2 的位 13 和位 12 中编程将随各个字符发送的停止位的数量。

- ***1*** 个停止位： 这是停止位数量的默认值。
- ***2*** 个停止位： 正常 USART 模式、单线模式和调制解调器模式支持该值。
- ***1.5*** 个停止位： 用于智能卡模式。
- ***0.5*** 个停止位：在智能卡模式下接收数据时使用。

空闲帧发送将包括停止位。

中断发送是 10 个低电平位（M[1:0] = 00 时）、 11 个低电平位（M[1:0] = 01 时）或 9 个低电平位（M[1:0] = 10 时），然后是 2 个停止位。无法传送长中断（中断长度大于 9/10/11 个低电平位）。

图 25-3 可配置的停止位



字符发送步骤

1. 对 USARTx_CR1 中的 M 位进行编程以定义字长。
2. 使用 USARTx_BRR 寄存器选择所需波特率。
3. 对 USARTx_CR2 中的停止位数量进行编程。
4. 通过向 USARTx_CR1 寄存器中的 UE 位写入 1 使能 USART。
5. 如果将进行多缓冲区通信，请选择 USARTx_CR3 中的 DMA 使能 (DMAT)。按照多缓冲区通信中的解释说明配置 DMA 寄存器。
6. 将 USARTx_CR1 中的 TE 位置 1 以便在首次发送时发送一个空闲帧。
7. 在 USARTx_TDR 寄存器中写入要发送的数据（该操作将清零 TXE 位）。为每个要在单缓冲区模式下发送的数据重复这一步骤。
8. 向 USARTx_TDR 寄存器写入最后一个数据后，等待至 TC=1。这表明最后一个帧的传送已完成。禁止 USART 或进入暂停模式时需要此步骤，以避免损坏最后一次发送。

单字节通信

始终通过向发送数据寄存器写入数据将 TXE 位清零。

TXE 位由硬件置 1，它表示：

- 数据已从 USARTx_TDR 寄存器移到移位寄存器中且数据发送已开始。
- USARTx_TDR 寄存器为空。
- USARTx_TDR 寄存器中可写入下一个数据，而不会覆盖前一个数据。

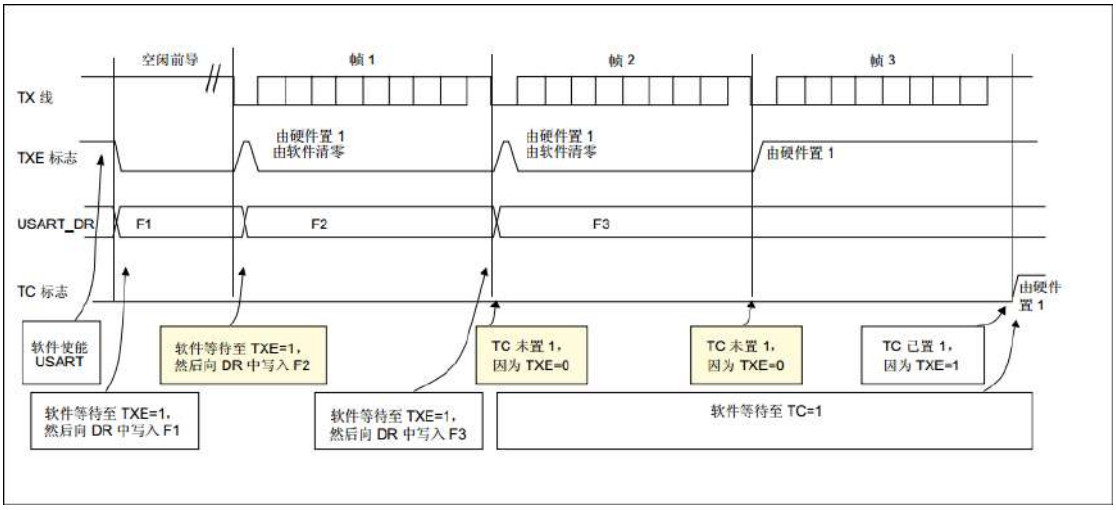
TXEIE 位置 1 时该标志位会生成中断。

发送时，要传入 USARTx_TDR 寄存器的写指令中存有 TDR 寄存器中的数据，接下来，该数据将在当前进行的发送结束时复制到移位寄存器中。未发送时，要传入 USARTx_TDR 寄存器的写指令将数据置于移位寄存器中，数据发送开始时， TXE 位置 1。

如果帧已发送（停止位后）且 TXE 位置 1， TC 位将变为高电平。如果 USARTx_CR1 寄存器中的 TCIE 位置 1，将生成中断。

向 USARTx_TDR 寄存器中写入最后一个数据后，必须等待至 TC=1，之后才可禁止 USART 或使微控制器进入低功率模式。

图 25-4 发送时的 **TC/TXE** 行为



中断字符

将 SBKRQ 位置 1 将发送一个中断字符。中断帧的长度取决于 M 位。

如果将“1”写入 SBKRQ 位，则当前字符发送完成后，将在 TX 线路上发送一个中断字符。

通过写操作将 SBKF 位置 1 并在中断字符发送完成时（发送中断字符后的停止位期间），此位由硬件复位。USART 在中断帧末尾的两位持续时间内插入一个逻辑“1”信号 (STOP)，以确保识别下个帧的起始位。

这种情况下，应用需要在之前插入的所有数据（包括尚未发送的数据）后发送中断字符，软件应等到 TXE 标志使能后将 SBKRQ 位置 1。

空闲字符

将 TE 位置 1 会驱动 USART 在第一个数据帧之前发送一个空闲帧。

25.5.3 USART 接收器

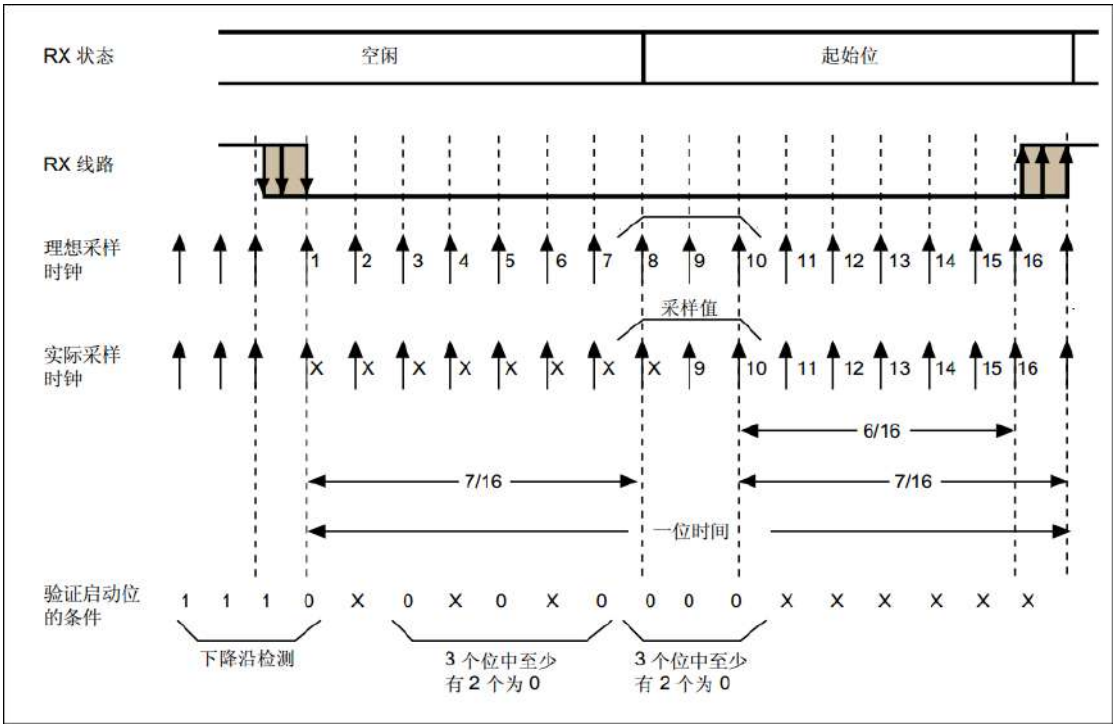
USARTx 可接收 7 位、 8 位或 9 位的数据字，具体取决于 USART_CR1 寄存器中的 M 位。

起始位检测

16 倍或 8 倍过采样时，起始位检测序列相同。

在 USART 中，识别出特定序列的采样时会检测起始位。此序列为：1 1 1 0 X 0 X 0X 0X 0 X0X 0。

图 25-5 **16** 倍或 **8** 倍过采样时的起始位检测



注： 如果序列不完整，起始位检测将中止，接收器将返回空闲状态（无标志位置 1）等待下降沿。

如果 3 个采样位均为 0（针对第 3 位、第 5 位和第 7 位进行首次采样时检测到这 3 位均为 0；针对第 8 位、第 9 位和第 10 位进行第二次采样时检测到这 3 位均为 0），可确认起始位（RXNE 标志位置 1， RXNEIE=1 时生成中断）。

满足以下条件时，可验证起始位（RXNE 标志位置 1，如果 RXNEIE=1 则生成中断）但 NF 噪声标志位置 1：

a) 对于两次采样， 3 个采样位中有 2 位为 0（针对第 3 位、第 5 位和第 7 位进行采样；针对第 8 位、第 9 位和第 10 位采样）

或

b) 如果其中一次采样时（对第 3 位、第 5 位和第 7 位进行采样或对第 8 位、第 9 位和第 10 位进行采样）， 3 个采样位中有 2 个为 0。

字符接收

USART 接收期间，首先通过 RX 引脚移入数据的最低有效位（默认配置）。该模式下， USARTx_RDR 寄存器的缓冲区 (RDR) 位于内部总线和接收移位寄存器之间。

字符接收步骤

1. 对 USARTx_CR1 中的 M 位进行编程以定义字长。
2. 使用波特率寄存器 USARTx_BRR 选择所需波特率
3. 对 USARTx_CR2 中的停止位数量进行编程。
4. 通过向 USARTx_CR1 寄存器中的 UE 位写入 1 使能 USART。
5. 如果将进行多缓冲区通信，请选择 USARTx_CR3 中的 DMA 使能 (DMAR)。按照多缓冲区通信中的解释说明配置 DMA 寄存器。
6. 将 RE 位 USARTx_CR1 置 1。这一操作将使能接收器开始搜索起始位。

接收到字符时：

- RXNE 位置 1。这表明移位寄存器的内容已传送到 RDR。也就是说，已接收到并可读取数

据（及其相应的错误标志）。

- 如果 RXNEIE 位置 1，则会生成中断。
- 如果接收期间已检测到帧错误、噪声错误或上溢错误，错误标志位可置 1。也可使用 RXNE 位将 PE 标志置 1。
- 在多缓冲区模式下，每接收到一个字节后 RXNE 均置 1，然后通过 DMA 对接收数据寄存器执行读操作清零。
- 在单缓冲区模式下，通过软件对 USARTx_RDR 寄存器执行读操作将 RXNE 位清零。

也可以通过将 USARTx_RQR 寄存器中的 RXFRQ 位置 1 将 RXNE 标志位清零。 RXNE 位必须在结束接收下一个字符前清零，以避免发生上溢错误。

中断字符

接收到中断字符时， USART 将会按照帧错误对其进行处理。

空闲字符

检测到空闲帧时，处理步骤与接收到数据的情况相同；如果 IDLEIE 位置 1，则会产生中断。

上溢错误

如果在 RXNE 未复位时接收到字符，则会发生上溢错误。 RXNE 位清零前，数据无法从移位寄存器传送到 RDR 寄存器。

每接收到一个字节后， RXNE 标志位都将置 1。当 RXNE 标志位是 1 时，如果在接收到下一个数据或尚未处理上一个 DMA 请求时，则会发生上溢错误。发生上溢错误时：

- ORE 位置 1。
- RDR 中的内容不会丢失。对 USARTx_RDR 执行读操作时可使用先前的数据。
- 移位寄存器将被覆盖。之后，上溢期间接收到的任何数据都将丢失。
- 如果 RXNEIE 位置 1 或 EIE 位置 1，则会生成中断。
- 通过设置 ICR 寄存器中的 ORECF 位复位 ORE 位。

注： *ORE* 位置 *1* 时表示至少 *1* 个数据丢失。存在两种可能：

- 如果 *RXNE=1*，则最后一个有效数据存储于接收寄存器 *RDR* 中并且可进行读取；
- 如果 *RXNE=0*，则表示最后一个有效数据已被读取，因此 *RDR* 中没有要读取的数据。接收到新（和丢失）数据的同时已读取 *RDR* 中的最后一个有效数据时，会发生该情况。

选择合适的过采样方法

当支持双时钟域和从停止模式唤醒时，时钟源可以是以下源之一： PCLK（默认值）、

LSE、 HSI16 或 SYSCLK。否则， USART 时钟源是 PCLK。

选择 LSE 或 HSI16 作为时钟源可允许 USART 在 MCU 处于低功耗模式时接收数据。需要时， US ART 可基于所接收的数据和唤醒模式选择来唤醒 MCU，以通过用软件读取 USARTx_RDR 寄存器的方式或通过 DMA 的方式传输已接收数据。

对于其它时钟源，系统必须激活才能进行 USART 通信。

接收器采用不同的用户可配置过采样技术，可以从噪声中提取有效数据。这可在最大通信速度与抗噪声/时钟误差性能之间实现平衡。

可通过编程 USARTx_CR1 寄存器中的 OVER8 位来选择采样方法，且采样时钟可以是波特率时钟的 16 倍或 8 倍（请参见图 25-6 和图 25-7）。

根据应用：

- 选择 8 倍过采样 (OVER8=1) 以获得更高的速度（高达 $f_{CK}/8$）。这种情况下接收器对时钟偏差的最大容差将会降低（请参见 *USART* 接收器对时钟偏差的容差）
- 选择 16 倍过采样 (OVER8=0) 以增加接收器对时钟偏差的容差。这种情况下，最大速度限制为最高 $f_{CK}/16$，其中 f_{CK} 为时钟源频率。

可通过编程 USARTx_CR3 寄存器中的 ONEBIT 位选择用于评估逻辑电平的方法。有两种选择：

• 在已接收位的中心进行三次采样，从而进行多数表决。这种情况下，如果用于多数表决的 3 次采样结果不相等， NF 位置 1。

• 在已接收位的中心进行单次采样

根据应用：

– 在噪声环境下工作时，请选择三次采样的多数表决法 (ONEBIT=0)；在检测到噪声时请拒绝数据（请参见表 26-1 采用 **16** 倍或 **8** 倍过采样时，在 **fCK = 32 MHz** 下编程波特率的误差计算），因为这表示采样过程中产生了干扰。

– 线路无噪声时请选择单次采样法 (ONEBIT=1) 以增加接收器对时钟偏差的容差（请参见 *USART* 接收器对时钟偏差的容差）。这种情况下 NF 位始终不会置 1。

帧中检测到噪声时：

• 在 RXNE 位的上升沿时 NF 位置 1。

• 无效数据从移位寄存器传送到 USARTx_RDR 寄存器。

• 单字节通信时无中断产生。然而，在 RXNE 位产生中断时，该位出现上升沿。多缓冲区通信时， USARTx_CR3 寄存器中的 EIE 位置 1 时将发出中断。

通过设置 ICR 寄存器中的 NFCF 位复位 NF 位。

注： *LIN*、智能卡和 *IrDA* 模式下不可采用 *8* 倍过采样。在这些模式下， *OVER8* 位由硬件强制清零。

图 25-6 **16** 倍过采样时的数据采样

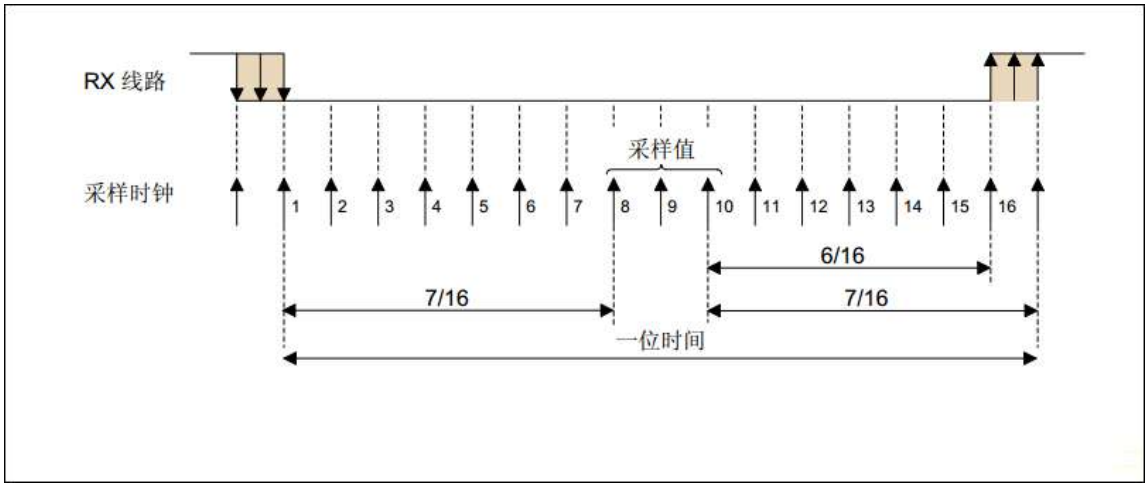


图 25-7 8 倍过采样时的数据采样

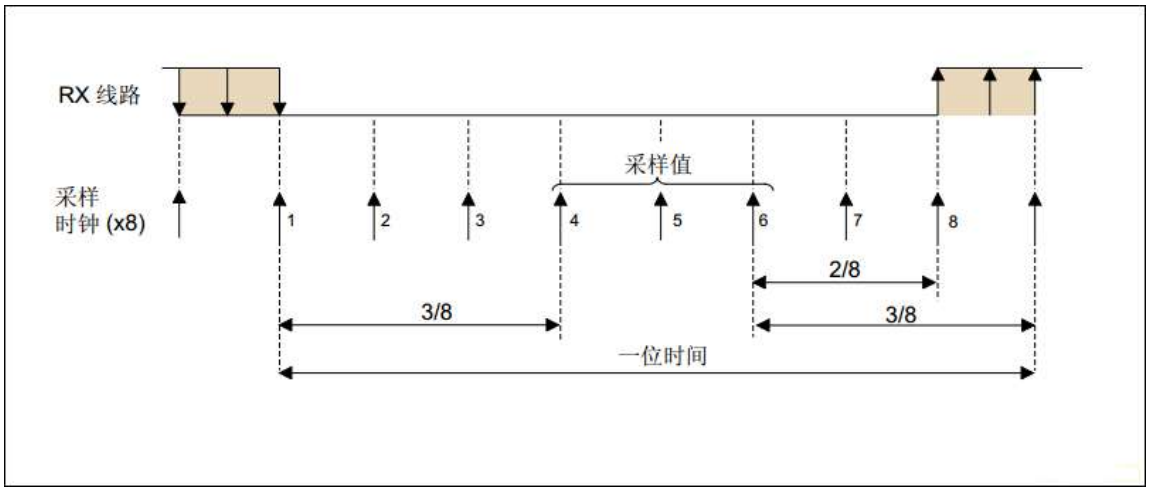


表 25-2 通过采样数据进行噪声检测

采样值	NE 状态	接收的位值
000	0	0
001	1	0
010	1	0
011	1	1
100	1	0
101	1	1
110	1	1
111	0	1

帧错误 (Framing error)

如果接收数据时未在预期时间内识别出停止位，从而出现同步失效或过度的噪声，则会检测到帧错误。

检测到帧错误时：

- FE 位由硬件置 1。
- 无效数据从移位寄存器传送到 USARTx_RDR 寄存器。
- 单字节通信时无中断产生。然而，在 RXNE 位产生中断时，该位出现上升沿。多缓冲区通信时， USARTx_CR3 寄存器中的 EIE 位置 1 时将发出中断。

通过将 1 写入 USARTx_ICR 寄存器中的 FECF 位来复位 FE 位。

接收期间可配置的停止位

可通过控制寄存器 2 中的控制位配置要接收的停止位的数量 - 可以是 1 或 2 个（正常模式下），也可以是 0.5 或 1.5 个（智能卡模式下）。

- *0.5* 个停止位（在智能卡模式下接收时）：不会对 *0.5* 个停止位进行采样。结果，选择 *0.5* 个停止位时，无法检测到帧错误和中断帧。
- *1* 个停止位：将在第 8、第 9 和第 10 次采样时对 1 个停止位进行采样。
- *1.5* 个停止位（智能卡模式）：在智能卡模式下发送时，设备必须检查数据是否正确发送。因此必须使能接收器块（USARTx_CR1 寄存器中的 RE =1）并检查停止位，以测试智能卡是否已检测到奇偶校验错误。发生奇偶校验错误时，智能卡会在采样时将数据信号强制为低电平，即 NACK 信号-，该信号被标记为帧错误。之后， FE 标志在 1.5 个停止位的末尾由 RXNE 置 1。在第 16、第 17 和第 18 次采样时对 1.5 个停止位进行采样（停止位采样开始后维持 1 个波特时钟周期）。 1.5 个停止位可分为 2 个部分： 0.5 个波特时钟周期（未发生任何动作），然后是 1 个正常的停止位周期（一半时间处进行采样）。更多详细信息，请参见 *USART* 智能卡模式。
- *2* 个停止位：采样 2 个停止位时在第 8、第 9 和第 10 次采样时对第一个停止位进行采样。如果在

第一个停止位期间检测到帧错误，则帧错误标志位将会置 1。发生帧错误时不检测第 2 个停止位。 RXNE 标志将在第一个停止位末尾时置 1。

25.5.4 USART 波特率生成

接收器和发送器（Rx 和 Tx）的波特率设置为 UART_BRR 寄存器中编程的相同值。

公式 **1**：适用于标准 **USART**（包括 **SPI** 模式）的波特率（**OVER8 = 0** 或 **1**）

16 倍过采样时的公式为：

$$\textbf{Tx/Rx}\ 波特\ = \frac{fck}{UARTDIV}$$

8 倍过采样时的公式为：

$$\textbf{Tx/Rx}\ 波特\ = \frac{2xfck}{UARTDIV}$$

公式 2：智能卡、 LIN 和 IrDA 模式 (OVER8 = 0) 下的波特率

智能卡、 LIN 和 IrDA 模式下只支持 16 倍过采样：

$$\textbf{Tx/Rx}\ 波特\ = \frac{fck}{UARTDIV}$$

USARTDIV 是一个存放在 USARTx_BRR 寄存器中的无符号定点数。

- 当 OVER8 = 0 时， BRR = USARTDIV。
- 当 OVER8 = 1 时
  - BRR[2:0] = USARTDIV[3:0]，右移 1 位。
  - BRR[3] 必须保持清零。
  - BRR[15:4] = USARTDIV[15:4]

注： 对 UART_BRR 执行写操作后，波特率计数器更新为波特率寄存器中的新值。因此，波特率寄存器的值不应在通信时发生更改。

16 倍或 *8* 倍过采样时， *USARTDIV* 必须大于或等于 *0d16*。

如何从 **USARTx_BRR** 寄存器中获取 **USARTDIV**

示例 **1**

要通过 fCK = 8 MHz 获得 9600 波特率。

• 16 倍过采样时：

USARTDIV = 8 000 000/9600

BRR = USARTDIV = 833d = 0341h

• 8 倍过采样时：

USARTDIV = 2 * 8 000 000/9600

USARTDIV = 1666,66 (1667d = 683h)

BRR[3:0] = 3h << 1 = 1h

BRR = 0x681

示例 **2**

要通过 fCK = 32 MHz 获得 921.6 K 波特率。

• 16 倍过采样时：

USARTDIV = 32 000 000/921 600

BRR = USARTDIV = 35d = 23h

• 8 倍过采样时：

USARTDIV = 2 * 32 000 000/921 600

USARTDIV = 70d = 46h

BRR[3:0] = USARTDIV[3:0] >> 1 = 6h >> 1 = 3h

BRR = 0x43

表 25-3 采用 **16** 倍或 **8** 倍过采样时，在 **fCK = 32 MHz** 下编程波特率的误差计算

波特率		**16** 倍过采样 **(OVER8 = 0)**			**8** 倍过采样 **(OVER8 = 1)**		
序号	所需值	实际值	**BRR**	误差 **% =**（计算值 - 所需值）**/** 所需波特率	实际值	**BRR**	误差 **%**
1	2.4 KBps	2.4 KBps	0x3415	0	2.4 KBps	0x6825	0
2	9.6 KBps	9.6 KBps	0xD05	0	9.6 KBps	0x1A05	0
3	19.2 KBps	19.19 KBps	0x683	0.02	19.2 KBps	0xD02	0
4	38.4 KBps	38.41 KBps	0x341	0.04	38.39 KBps	0x681	0.02
5	57.6 KBps	57.55 KBps	0x22C	0.08	57.6 KBps	0x453	0
6	115.2 KBps	115.1 KBps	0x116	0.08	115.11 KBps	0x226	0.08
7	230.4 KBps	230.21 KBps	0x8B	0.08	230.21 KBps	0x113	0.08
8	460.8 KBps	463.76 KBps	0x045	0.64	460.06 KBps	0x85	0.08
9	921.6 KBps	914.28 KBps	0x23	0.79	927.5 KBps	0x42	0.79
10	2 MBps	2 MBps	0x10	0	2 MBps	0x20	0
12	4MBps	4MBps	NA	NA	4MBps	0x10	0

1. CPU 时钟越低，特定波特率的精度就越低。可达到的波特率的上限可使用这些数据进行确定。

25.5.5 USART 接收器对时钟偏差的容差

仅当总时钟系统偏差小于 USART 接收器的容差时， USART 异步接收器才能正常工作。影响总偏差的因素包括：

- DTRA：发送器误差引起的偏差（其中还包括发送器本地振荡器的偏差）
- DQUANT：接收器的波特率量化引起的误差
- DREC：接收器本地振荡器的偏差
- DTCL：传输线路引起的偏差（通常是由于收发器所引起，它可能会在低电平到高电平转换时序与高电平到低电平转换时序之间引入不对称）

DTRA DQUANT DREC DTCL DWU + + + + USART < 接收器的容差

其中

DWU 是使用从停止模式唤醒时因采样点偏差产生的误差。

M[1:0] = 01 时：

M[1:0] = 00 时：

M[1:0] = 10 时：

tWUUSART 是检测到唤醒事件到时钟（由外设请求）与调压器均就绪的时间。

USART 接收器在表 25-4 中指定的最大容许偏差下可正确接收数据，具体取决于以下选择：

- 由 USARTx_CR1 寄存器中的 M 位定义的 9 位、 10 位或 11 位字符长度
- 由 USARTx_CR1 寄存器中的 OVER8 位定义的 8 倍或 16 倍过采样
- USARTx_BRR 寄存器的 BRR[3:0] 位等于或不等于 0000
- 使用 1 位或 3 位对数据进行采样，取决于 USARTx_CR3 寄存器中 ONEBIT 位的值

表 25-4 **BRR [3:0] = 0000** 时的 **USART** 接收器容差

M 位	OVER8 位=0		OVER8 位=1	
	ONEBIT=0	ONEBIT=1	ONEBIT=0	ONEBIT=1
00	3.75%	4.375%	2.50%	3.75%
01	3.41%	3.97%	2.27%	3.41%
10	4.16%	4.86%	2.77%	4.16%

注： 当接收的帧恰好包含 *10* 个（*M* 位 *= 00*）、 *11* 个（*M* 位 *= 01*）或 *9* 个（*M* 位 *= 10*）位持续时间的空闲帧时， 表 *126* 和表 *127* 中指定的数据可能与特例中的数据略微不同。

25.5.6 USART 自动波特率检测

USART 能够根据接收一个字符检测并自动设置 USARTx_BRR 寄存器的值。自动波特率检测在以下两种情况下非常有用：

- 事先不知道系统的通信速度
- 系统正在使用精确度相对较低的时钟源且该机制允许在不测量时钟偏差的情况下获得正确的波特率。

时钟源频率必须与预期通信速度匹配（16 倍过采样时，波特率介于 fCK/65535 与 fCK/16 之间； 8 倍过采样时，波特率介于 fCK/65535 与 fCK/8 之间）。

激活自动波特率检测前，必须先选择自动波特率检测模式。根据不同的字符模式，存在各种检测模式。

这些模式可通过 USARTx_CR2 寄存器中的 ABRMOD[1:0] 字段选择。在这些自动波特率模式下，波特率在同步接收数据期间被多次测量，每次测量的结果都与前一次进行比较。

这些模式包括：

- 模式 **0**：以 1 位开头的任意字符。这种情况下，USART 会测量起始位的持续时间（下降沿到上升沿）。
- 模式 **1**：以 10xx 位模式开头的任意字符。这种情况下， USART 会测量起始位和第一个数据位的持续时间。测量在下降沿到下降沿期间完成，可在信号斜率较小时确保较高的精度。
- 模式 **2**： 0x7F 字符帧（可以是 LSB 在前模式下的 0x7F 字符，也可以是 MSB 在前模式下的 0xFE 字符）。这种情况下，先在起始位结束时更新波特率 (BR)，然后在位 6 结束时更新波特率（根据从下降沿到下降沿执行的测量： BR6）。以 BR 对位 0 到位 6 进行采样，而以 BR6 对字符的其它位进行采样。
- 模式 **3**： 0x55 字符帧。这种情况下，先在起始位结束时更新波特率 (BR)，然后在位 0 结束时更新波特率（根据从下降沿到下降沿执行的测量： BR0），最后在位 6 结束时更新波特率 (BR6)。以 BR 对位 0 进行采样，以 BR0 对位 1 到位 6 进行采样，以 BR6 对字符的其它位进行采样。

同时，对 RX 线路的每个中间转换执行其它检查。如果 RX 上的转换与接收器（基于根据位 0 计算的波特率的接收器）未充分同步，则生成错误。

激活自动波特率检测前，必须先通过向 USARTx_BRR 寄存器写入非零的波特率值来初始化该寄存器。

通过将 USARTx_CR2 寄存器中的 ABREN 位置 1 来激活自动波特率检测。之后 USART 将等待 RX 线路上的第一个字符。通过将 USARTx_ISR 寄存器中的 ABRF 标志置 1 来指示自动波特率操作完成。如果线路繁忙，则无法保证正确的波特率检测。这种情况下， BRR 值可能会损坏， ABRE 错误标志位将置 1。如果通信速度不在自动波特率检测范围（位持续时间不在 16 个和 65536 个时钟周期（16 倍过采样时）之间，也不在 8 个和 65536 个时钟周期（8 倍过采样时）之间）内，也会出现这种情况。

RXNE 中断将指示操作结束。

稍后，可随时通过复位 ABRF 标志（通过写入 0）重新启动自动波特率检测。

注： 如果在自动波特率操作期间禁止 *USART (UE=0)*，则可能损坏 *BRR* 值。

25.5.7 使用 USART 进行多处理器通信

在多处理器通信中，以下位要保持清零：

- USART_CR2 寄存器中的 LINEN 位，
- USART_CR3 寄存器中的 HDSEL、 IREN 和 SCEN 位。

可以通过 USART 进行多处理器通信（多个 USART 连接在一个网络中）。例如，其中一个 USART 可以是主 USART，其 TX 输出与其它 USART 的 RX 输入相连接。其它 USART 为从 USART，其各自的 TX 输出在逻辑上通过与运算连在一起，并与主 USART 的 RX 输入相连接。

在多处理器配置中，理想情况下通常只有预期的消息接收方主动接收完整的消息内容，从而减少由所有未被寻址的接收器造成的冗余 USART 服务开销。

可通过静默功能将未被寻址的器件置于静默模式下。为了使用静默模式功能，必须将 USARTx_CR1 寄存器中的 MME 位置 1。

在静默模式下：

- 不得将接收状态位置 1。
- 禁止任何接收中断。
- USARTx_ISR 寄存器中的 RWU 位置 1。在某些情况下， RWU 可以由硬件或软件通过 USARTx_RQR 寄存器中的 MMRQ 位自动控制。

根据 USARTx_CR1 寄存器中 WAKE 位的设置， USART 可使用以下两种方法进入或退出静默模式：

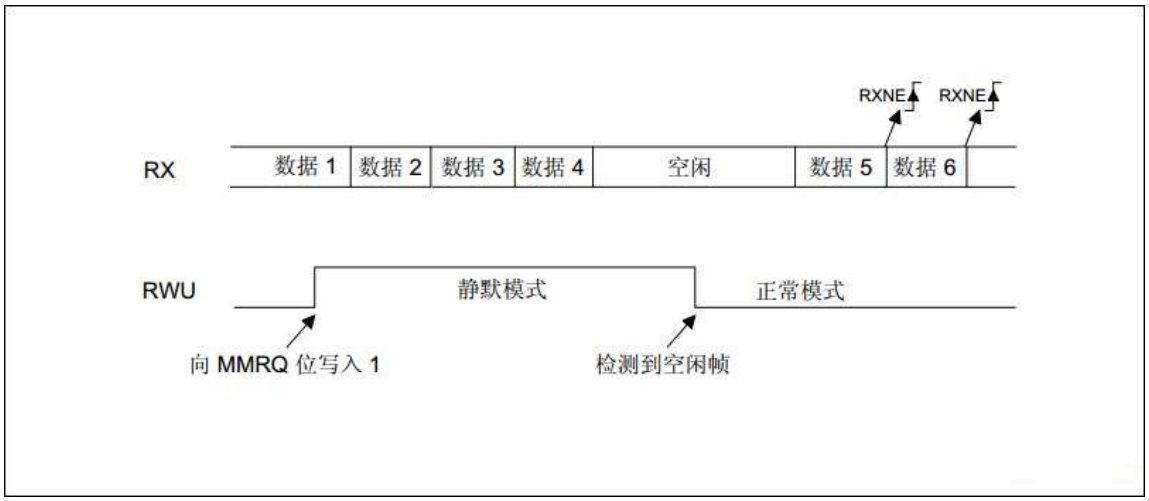
- 如果 WAKE 位被复位，则进行空闲线路检测，
- 如果 WAKE 位置 1，则进行地址标记检测。

空闲线路检测 **(WAKE=0)**

当向 MMRQ 位写入 1 且 RWU 位自动置 1 时， USART 进入静默模式。

当检测到空闲帧时，它会被唤醒。此时 RWU 位会由硬件清零，但 USARTx_ISR 寄存器中的 IDLE 位不会置 1。下图中给出了使用空闲线路检测时静默模式行为的示例。

图 25-8 使用空闲线路检测时的静默模式



注： 如果在 IDLE 字符已经过去时将 MMRQ 位置 1，将不会进入静默模式（RWU 未置 1）。

如果在线路处于空闲状态时激活 *USART*，在一个 *IDLE* 帧持续时间后（不只在接收一个字符帧后）会检测到空闲状态。

4 位*I*7 位地址标记检测 (WAKE=1)

在此模式下，如果字节的 MSB 为 1，则将这些字节识别为地址，否则将其识别为数据。在地址字节中，目标接收器的地址位于 4 个或 7 个 LSB 中。 7 位或 4 位地址检测通过 ADDM7 位来选择。接收器会将此 4 位*I*7 位字与其地址进行比较，该接收器的地址在 USARTx_CR2 寄存器的 ADD 位中进行设置。

注： 在 7 位和 9 位数据模式下，地址检测分别在 6 位和 8 位地址上完成（ADD[5:0] 和 ADD[7:0]）。

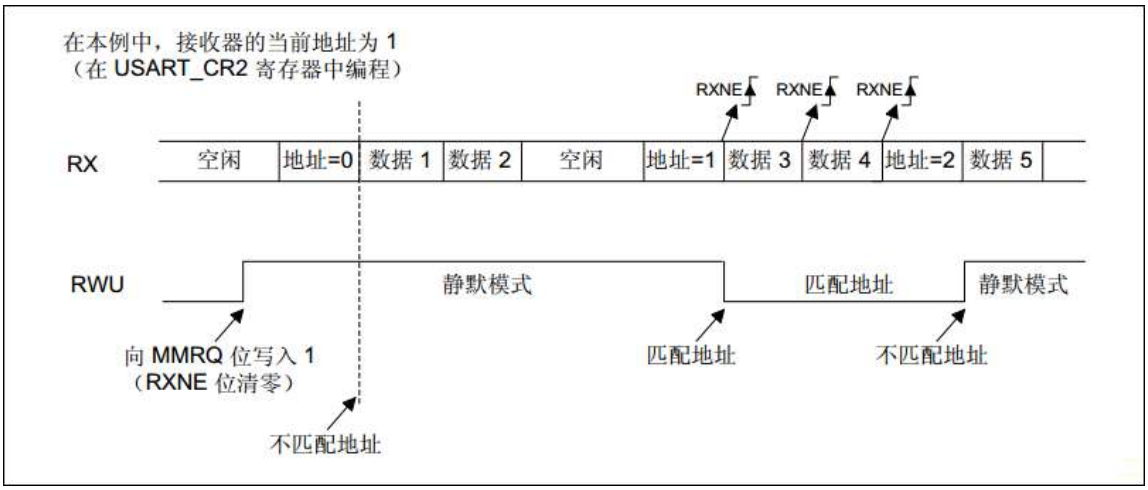
当接收到与其编程地址不匹配的地址字符时， USART 会进入静默模式。此时， RWU 位将由硬件置 1。 USART 进入静默模式后， RXNE 标志不会针对此地址字节置 1，也不会发出中断或 DMA 请求。

当向 MMRQ 位写入 1 时， USART 也会进入静默模式。这种情况下， RWU 位也自动置 1。

当接收到与编程地址匹配的地址字符时， USART 会退出静默模式。然后 RWU 位被清零，可以开始正常接收后续字节。由于 RWU 位已清零， RXNE 位会针对地址字符置 1。

图 *243* 中给出了使用地址标记检测时静默模式行为的示例。

图 25-9 使用地址标记检测时的静默模式



25.5.8 使用 USART 进行 Modbus 通信

USART 为 Modbus/RTU 和 Modbus/ASCII 协议的实现提供基本支持。 Modbus/RTU 是一个半双工块传输协议。该协议的控制部分（地址识别、块完整性控制和命令解析）必须用软件实现。

USART 为块结束检测提供基本支持，无需软件开销或其它资源。

Modbus/RTU

在此模式下，一个块的结束通过超过 2 个字符时间的“静默”（空闲线路）来识别。此功能通过可编程的超时功能实现。

超时功能和中断必须分别通过 USARTx_CR2 寄存器中的 RTOEN 位和 USARTx_CR1 寄存器中的 RTOIE 位激活。与 2 个字符时间（例如 22 x 位持续时间）的超时相对应的值必须在 RTO 寄存器中编程。如果在此期间接收线路空闲，则在接收到最后一个停止位后，将生成中断，同时通知软件当前块接收已完成。

Modbus/ASCII

在此模式下，块结束通过特定 (CR/LF) 字符序列识别。USART 通过字符匹配功能管理此机制。

通过在 ADD[7:0] 字段中编程 LF ASCII 码以及激活字符匹配中断 (CMIE=1)，软件可在接收到 LF 时获得通知并检查 DMA 缓存区中的 CR/LF。

25.5.9 USART 奇偶校验

将 USARTx_CR1 寄存器中的 PCE 位置 1，可以使能奇偶校验控制（发送时生成奇偶校验位，接收时进行奇偶校验检查）。根据 M 位定义的帧长度， 表 25-5 中列出了可能的 USART 帧格式。

表 25-5 帧格式

M 位	PCE 位	USART 帧(1)
00	0	| SB | 8 位数据 | STB |
00	1	| SB | 7 位数据 | PB | STB |
01	0	| SB | 9 位数据 | STB |
01	1	| SB | 8 位数据 | PB | STB |
10	0	| SB | 7 位数据 | STB |
10	1	| SB | 6 位数据 | PB | STB |

1. 图注： SB：起始位， STB：停止位， P：奇偶校验位。在数据寄存器中， PB 始终位于 MSB 位置（第 9 位、第 8 位或第 7 位，具体取决于 M 位的值）。

偶校验

对奇偶校验位进行计算，使帧和奇偶校验位中“1”的数量为偶数（帧由 6 个、7 个或 8 个 LSB 位组成，具体取决于 M 位的值）。

例如，如果数据=00110101 且 4 个位置 1，则在选择偶校验（USARTx_CR1 寄存器中的 PS 位 = 0）时，校验位为 0。

奇校验

对奇偶校验位进行计算，使帧和奇偶校验位中“1”的数量为奇数（帧由 6 个、7 个或 8 个 LSB 位组成，具体取决于 M 位的值）。

例如，如果数据=00110101 且 4 个位置 1，则在选择奇校验（USARTx_CR1 寄存器中的 PS 位 = 1）时，校验位为 1。

接收时进行奇偶校验检查

如果奇偶校验检查失败，则 USARTx_ISR 寄存器中的 PE 标志置 1；如果 USARTx_CR1 寄存器中 PEIE 位置 1，则会生成中断。通过软件将 1 写入 USARTx_ICR 寄存器中的 PECF 位来清零 PE 标志。

发送时的奇偶校验生成

如果 USARTx_CR1 寄存器中的 PCE 位置 1，则在数据寄存器中所写入数据的 MSB 位会进行传送，但是会由奇偶校验位进行更改（如果选择偶校验 (PS=0)，则“1”的数量为偶数；如果选择奇校验 (PS=1)，则“1”的数量为奇数）。

25.5.10 USART LIN（局域互连网络）模式

仅在支持 LIN 模式时才与本节相关。请参见 *USART* 实现。

通过将 USARTx_CR2 寄存器中的 LINEN 位置 1 来选择 LIN 模式。在 LIN 模式下，必须将以下位清零：

- USARTx_CR2 寄存器中的 STOP[1:0] 和 CLKEN 位，
- USARTx_CR3 寄存器中的 SCEN、 HDSEL 和 IREN 位。

LIN 发送

与正常的 USART 发送相比，在 LIN 主器件中发送时必须采用 *USART* 发送器中介绍的步骤，同时还具有以下区别：

- M 位清零以配置 8 位字长度。
- LINEN 位置 1 以进入 LIN 模式。此时，将 SBKRQ 位置 1 会发送 13 个“0”位作为断路字符。然后会发送值为“1”的 2 个位以进行下一启动检测。

LIN 接收

使能 LIN 模式后，将激活断路检测电路。该检测完全独立于正常的 USART 接收器。在空闲状态或某个帧期间，只要发生断路即可检测出来。

接收器（USARTx_CR1 寄存器中 RE=1）使能后，电路便开始监测启动信号的 RX 输入。检测起始位的方法与搜索断路字符或数据的方法相同。检测到起始位后，电路会对接下来的位进行采样，方法与数据采样相同（第 8、第 9 和第 10 次采样）。如果 10 个（USARTx_CR2 中 LBDL = 0 时）或 11 个（USARTx_CR2 中 LBDL=1 时）连续位均检测为“0”，且其后跟随分隔符，则 USARTx_ISR 寄存器中的 LBDF 标志将置 1。如果 LBDIE 位=1，则会生成中断。在验证断路前，会对分隔符进行检查，因为它表示 RX 线路已恢复到高电平。

如果在第 10 或第 11 次采样前已对“1”采样，则断路检测电路会取消当前检测，并重新搜索起始位。

如果禁止 LIN 模式 (LINEN=0)，接收器会作为正常的 USART 继续工作，不会再进行断路检测。

如果使能 LIN 模式 (LINEN=1)，只要发生帧错误（例如，在“0”处检测到停止位，这种情况可能出现在任何断路帧中），接收器即会停止，直到断路检测电路接收到“1”（断路字不完整时）或接收到分隔符（检测到断路时）为止。

图 25-10 **LIN** 模式下的断路检测（11 位断路长度——LBDL 位置 **1**）） 中显示了断路检测器状态机和断路标志的行为。

图 25-11 **LIN** 模式下的断路检测与帧错误检测中列出了断路帧的示例。

图 25-10 **LIN** 模式下的断路检测（**11** 位断路长度**——LBDL** 位置 **1**）

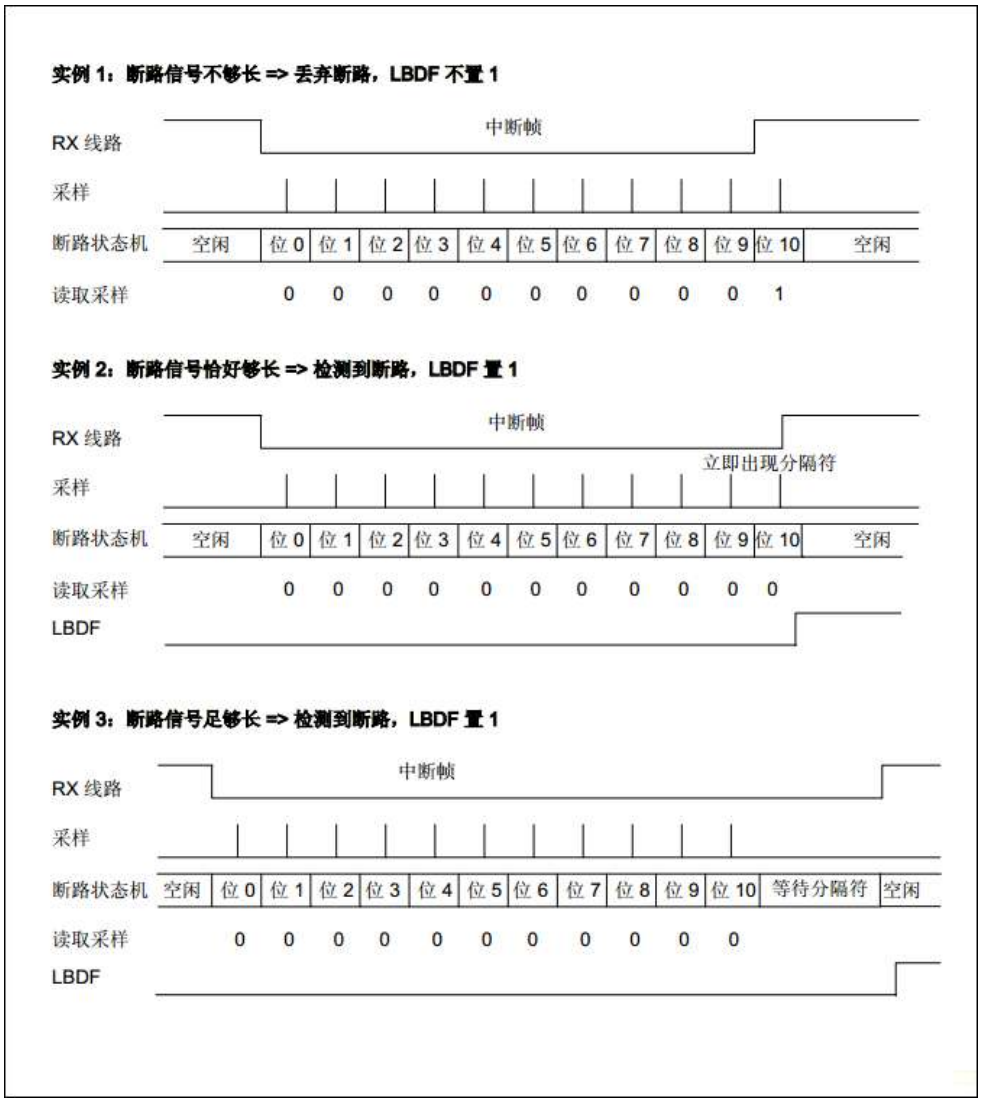
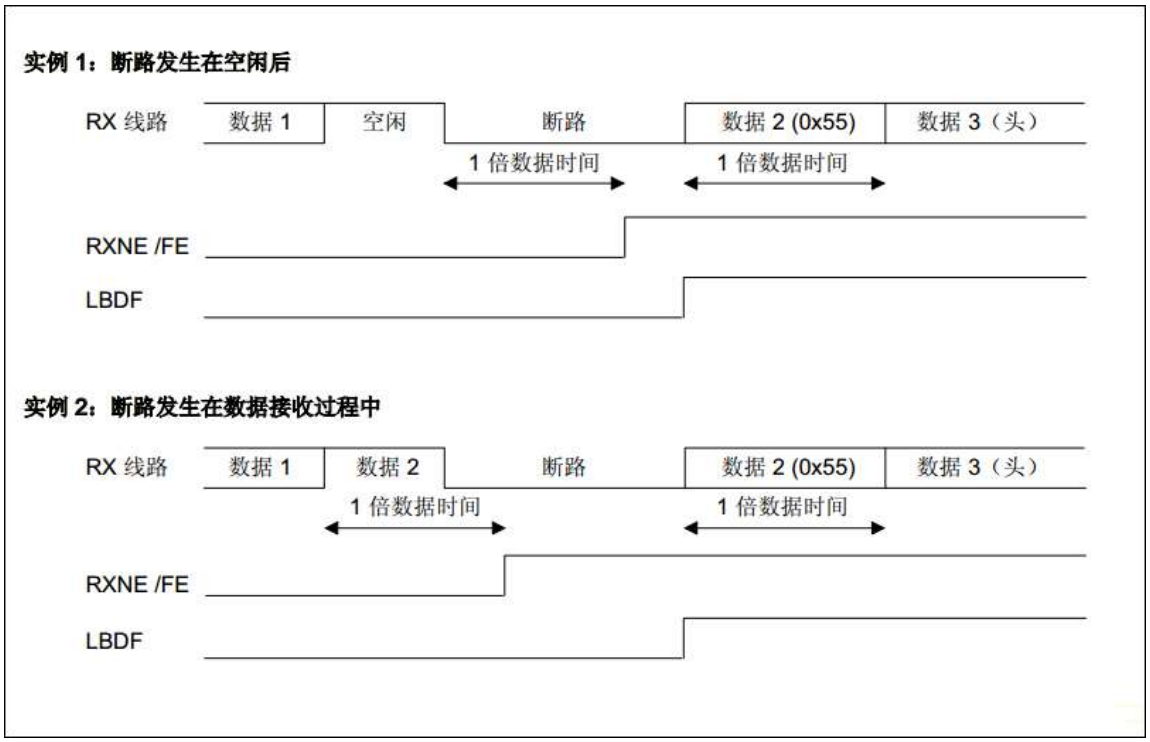


图 25-11 **LIN** 模式下的断路检测与帧错误检测



25.5.11 USART 同步模式

通过将 USARTx_CR2 寄存器中的 CLKEN 位写入 1 来选择同步模式。在同步模式下，必须将以下位清零：

- USARTx_CR2 寄存器中的 LINEN 位，
- USARTx_CR3 寄存器中的 SCEN、 HDSEL 和 IREN 位。

在此模式下， USART 可用于在主模式下控制双向同步串行通信。 CK 引脚是 USART 发送器时钟的输出。在起始位或停止位期间，不会向 CK 引脚发送时钟脉冲。在最后一个有效数据位（地址标记）期间，会（也可能不会）生成时钟脉冲，这取决于 USARTx_CR2 寄存器中 LBCL 位的状态。 USARTx_CR2 寄存器中的 CPOL 位用于选择时钟极性； USARTx_CR2 寄存器中的 CPHA 位用于选择外部时钟的相位（请参见图 25-12 图 25-13 和图 25-14）。

在空闲状态、报头模式和发送断路期间，外部 CK 时钟处于未激活状态。

USART 发送器在同步模式下的工作方式与异步模式下完全相同。但是由于 CK 与 TX 同步（根据 CPOL 和 CPHA），因此 TX 上的数据是同步的。

在此模式下， USART 接收器的工作方式与异步模式下不同。如果 RE=1，则数据在 CK 上采样（上升或下降沿，取决于 CPOL 和 CPHA），而不会进行任何过采样。此时必须确保建立时间和保持时间（取决于波特率： 1/16 位持续时间）。

注： *CK* 引脚可与 *TX* 引脚结合使用。因此，仅当使能发送器 *(TE=1)* 且正在发送数据时（数据寄存器 *USARTx_TDR* 已写入），才会提供时钟。这意味着，没有发送数据的情况下无法接收同步数据。

当 *USART* 被禁止时 *(UE=0)*，必须选择 *LBCL*、 *CPOL* 和 *CPHA* 位以确保时钟脉冲正常工作。

图 25-12 **USART** 同步发送示例

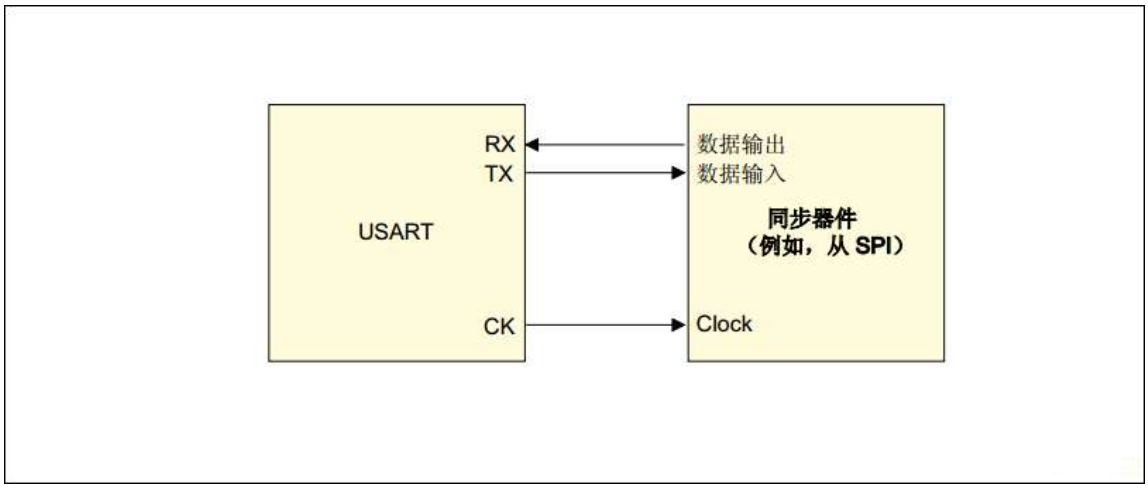


图 25-13 **USART** 数据时钟时序图 **(M 位 = 00)**

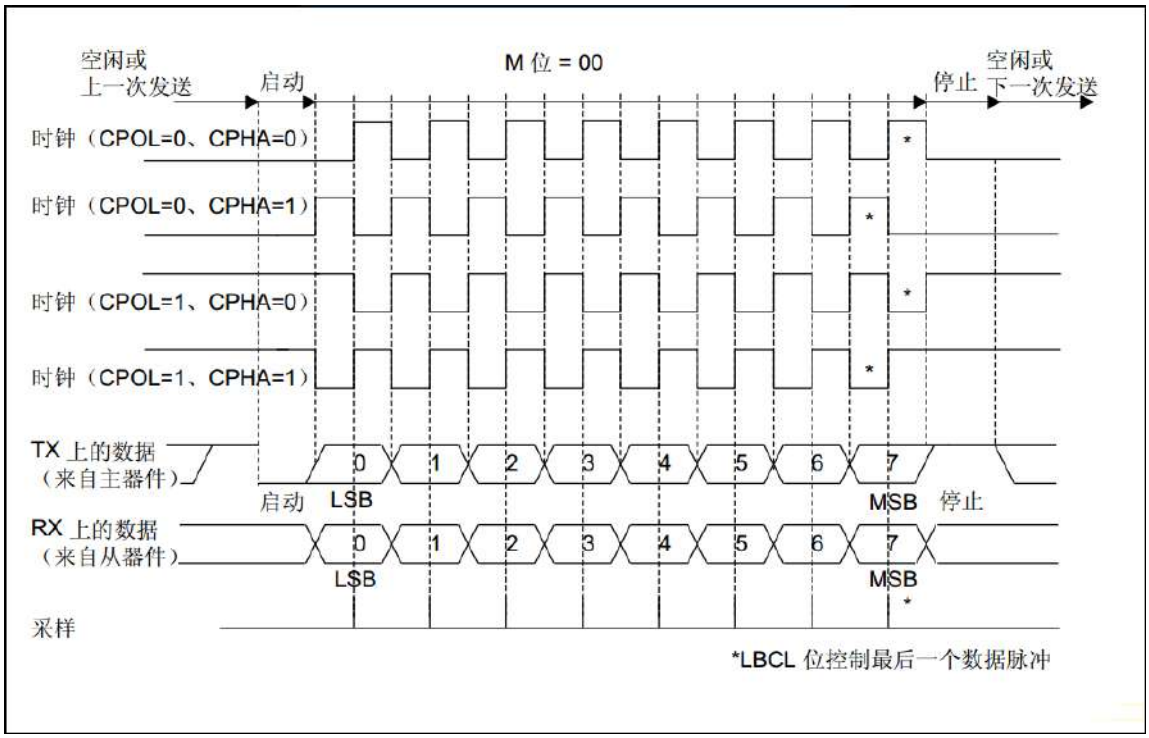


图 25-14 **USART** 数据时钟时序图 **(M** 位 **= 01)**

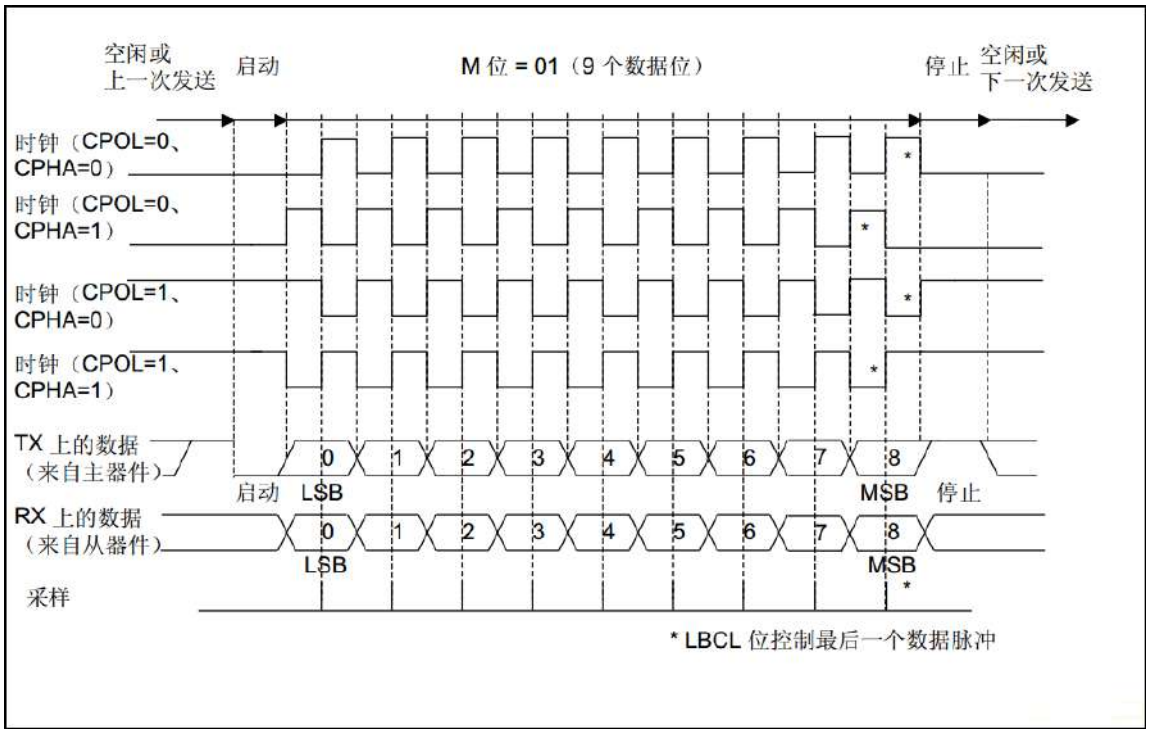
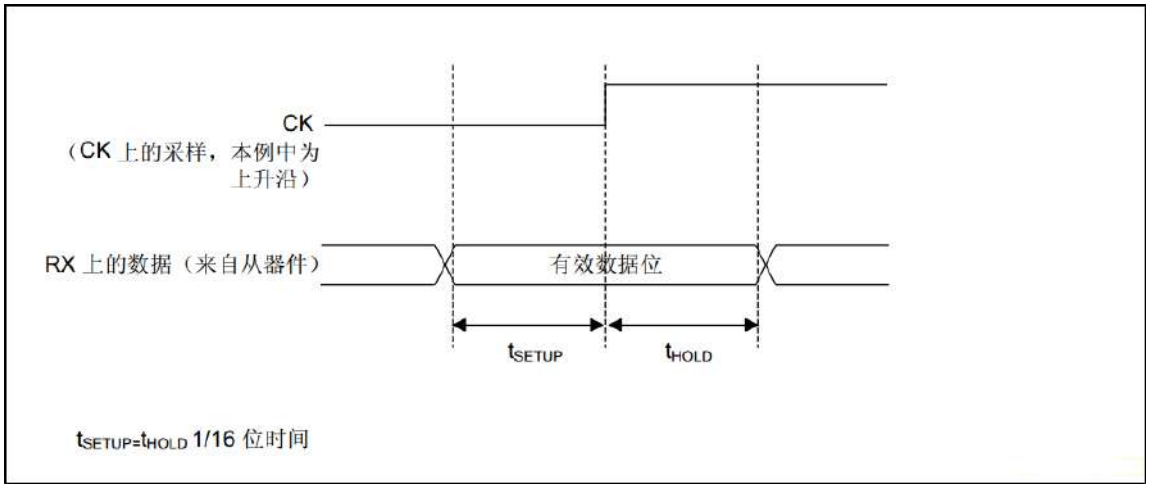


图 25-15 **RX** 数据建立**/**保持时间



注： 在智能卡模式下， CK 的功能有所不同。更多详细信息，请参见 USART 智能卡模式。

25.5.12 USART 单线半双工通信

通过将 USARTx_CR3 寄存器中的 HDSEL 位置 1 来选择单线半双工模式。在此模式下，必须将以下位清零：

- USARTx_CR2 寄存器中的 LINEN 位和 CLKEN 位，
- USARTx_CR3 寄存器中的 SCEN 和 IREN 位。

USART 可以配置为遵循单线半双工协议，其中 TX 和 RX 线路从内部相连接。使用 USARTx_CR3 寄存器中的控制位 HDSEL 可在半双工通信和全双工通信间进行选择。

一旦向 HDSEL 位写入 1：

- TX 和 RX 线路从内部相连接
- 不能再使用 RX 引脚
- 无数据传输时，TX 引脚始终处于释放状态。因此，它在空闲状态或接收过程中用作标准 I/O。这意味着，必须对 I/O 进行配置，以便将 TX 配置为复用功能开漏并外接上拉电阻。

除此之外，通信协议与正常 USART 模式下的通信协议相似。此线路上的任何冲突必须由软件管理（例如，使用中央仲裁器）。尤其要注意，发送过程永远不会被硬件封锁，只要数据是在 TE 位置 1 的情况下写入，发送就会持续进行。

25.5.13 USART 智能卡模式

仅在支持智能卡模式时才涉及本节内容。请参见 *USART 实现*。

通过将 USARTx_CR3 寄存器中的 SCEN 位置 1 选择智能卡模式。在智能卡模式下，必须将以下位清零：

- USARTx_CR2 寄存器中的 LINEN 位，
- USARTx_CR3 寄存器中的 HDSEL 和 IREN 位。

此外，可能需要将 CLKEN 位置 1，以便为智能卡提供时钟。

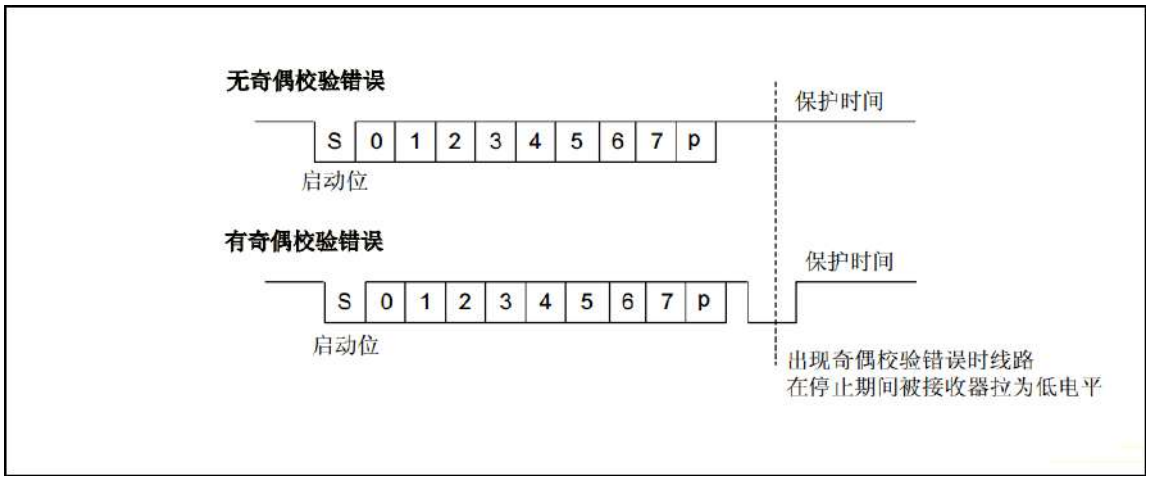
智能卡接口支持符合 ISO 7816-3 标准的异步协议智能卡。同时支持 T=0（字符模式）和 T=1（块模式）。USART 应如下所示进行配置：

- 8 个位加奇偶校验：其中字长度设置为 8 位， USARTx_CR1 寄存器中 PCE=1
- 1.5 个停止位：其中 USARTx_CR2 寄存器中 STOP=11。接收时也可以选择 0.5 个停止位。

在 T=0（字符）模式下，奇偶校验错误在保护时间周期内的每个字符结束时指示。

图 25-16 显示了有奇偶校验错误和无奇偶校验错误时数据线上情况的示例。

图 25-16 ISO 7816-3 异步协议



连接到智能卡时， USART 的 TX 输出会驱动一条双向线（它也由智能卡驱动）。必须将 TX 引脚配置为开漏引脚。

智能卡模式采用单线半双工通信协议。

- 从发送移位寄存器发送数据会经过至少 1/2 个时钟周期的延迟。正常工作时，已满的发送移位寄存器会在下一个波特时钟边沿开始移位。在智能卡模式下，此发送过程还会进一步经过 1/2 波特时钟周期的延迟。
- 发送时，如果智能卡检测到奇偶校验错误，它会通过将线路驱动为低电平 (NACK) 告知 USART 此状态。此 NACK 信号（将发送线拉低 1 个时钟周期）会导致发送器端（配置为 1.5

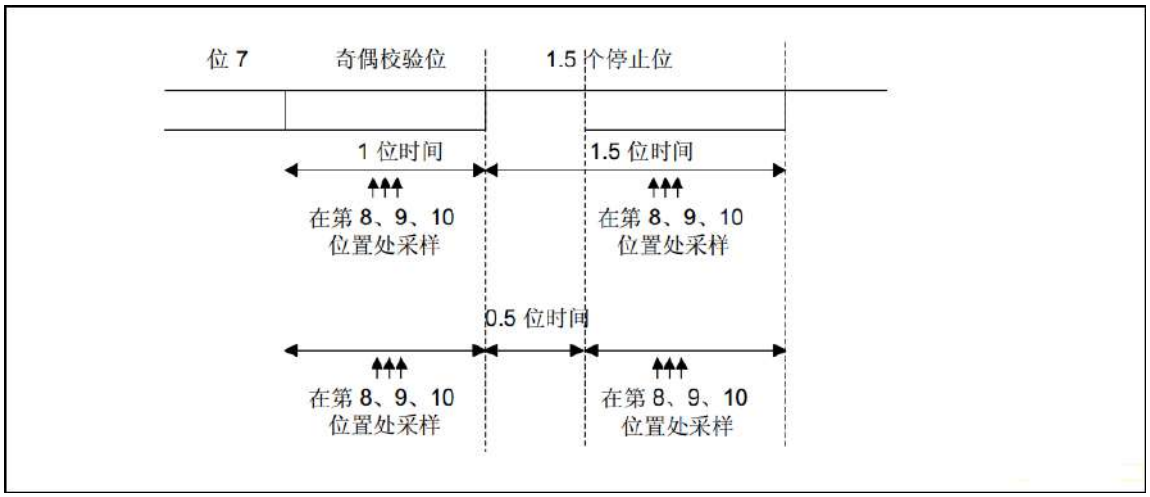
个停止位）出现帧错误。 USART 可根据协议自动重新发送数据。重试次数在 SCARCNT 位域中编程。如果经过编程次数的重试后， USART 继续收到 NACK， USART 会停止发送并将该错误以帧错误形式发出。可以使用 USARTx_RQR 寄存器中的 TXFRQ 位将 TXE 位置 1。

- 发送时智能卡自动重试：在 USART 检测 NACK 与重复字符的起始位之间插入 2.5 个波特率周期的延迟。接收最后一个重复字符后，立即将 TC 位置 1（无保护时间）。如果软件要重复此操作，必须确保标准要求的最短 2 个波特率周期。
- 如果在接收一个使用 1.5 个停止位编程的帧期间检测到奇偶校验错误，则在完成接收帧后，发送线会被拉低一个时钟周期。这是为了向智能卡指出发送到 USART 的数据尚未正确接收。如果 NACK 控制位置 1，接收器会向奇偶校验错误发送“NACK”信号；否则不会发送 NACK 信号（将在 T=1 模式下会使用）。如果接收到的字符错误，则不会激活 RXNE/接收 DMA 请求。根据协议规范，智能卡必须重新发送相同的字符。如果经过 SCARCNT 位域中指定的最大重试次数后接收到的字符仍然错误， USART 会停止发送 NACK 信号，并将错误以奇偶校验错误的形式发出。
- 接收时智能卡自动重试：如果 USART 向智能卡发送 NACK 信号，但智能卡不重复字符，则 BUSY 标志将保持置 1。
- 发送时， USART 会在两个连续字符之间插入保护时间（按照保护时间寄存器中编程的值）。由于保护时间在前一个字符的停止位后测量，因此必须将 GT[7:0] 寄存器编程为所需 CGT（字符保护时间，如 7816-3 规范所定义）减去 12（一个字符的持续时间）。
- 通过对保护时间寄存器进行编程，可以延迟 TC 标志的置位。正常工作时，当发送移位寄存器为空且没有新的发送请求出现时，会对 TC 标志进行置位。在智能卡模式下，空的发送移位寄存器会触发保护时间计数器，使其递增计数至保护时间寄存器中的值。在此期间， TC 标志被强制为低电平。当保护时间计数器达到设置值时， TC 置位为高电平。
- TC 标志的释放不受智能卡模式的影响。
- 如果在发送端检测到帧错误（由来自接收器的 NACK 信号引起），则发送端的接收块不会将 NACK 作为起始位进行检测。根据 ISO 协议，接收到的 NACK 信号的持续时间可以是 1 或 2 个时钟周期。
- 在接收端，如果检测到奇偶校验错误并发送了 NACK 信号，则接收端不会将 NACK 作为起始位进行检测。

注： 中断字符在智能卡模式下无效。带有帧错误的 0x00 数据被视为数据，而非中断。当翻转 TE 位时，不会发送空闲帧。空闲帧（在其它配置中进行了定义）在 ISO 协议中未进行定义。

图 25-17 详细介绍了 USART 如何对 NACK 信号采样。在本例中， USART 正在发送数据并配置了 1.5 个停止位。 USART 的接收部分已被使能，以检查数据的完整性和 NACK 信号。

图 25-17 使用 **1.5** 个停止位检测奇偶校验错误



USART 可以通过 CK 输出为智能卡提供时钟。在智能卡模式下，CK 仅通过一个 5 位预分频器由内部外设输入时钟提供。分频比在预分频器寄存器 USARTx_GTPR 中进行配置。CK 频率可在 $f_{CK/2}$ 到 $f_{CK/62}$ 之间进行编程，其中 f_{CK} 为外设输入时钟。

块模式(T=1)

在 T=1（块）模式下，通过将 UART_CR3 寄存器中的 NACK 位清零可停用奇偶校验错误发送。

在块模式下从智能卡请求读操作时，软件必须通过将 USART_CR2 寄存器中的 RTOEN 位置 1 使能接收器超时功能，并将 RTOR 寄存器中的 RTO 位域编程为 BWT（块等待时间）-11 值。如果此时间段后未从智能卡接收到应答，RTOF 标志将置 1，并生成超时中断（如果 USART_CR1 寄存器中的 RTOIE 位置 1）。如果该时间段后接收到第一个字符，则通过 RXNE 中断发出信号指示。

注：即使在使用 DMA 模式下的 USART 从块模式下的智能卡读取时，也必须使能 RXNE 中断。同时，只有在接收到第一个字节后才可使能 DMA。

接收到第一个字符（RXNE 中断）后，为允许自动检查两个连续字符间的最大等待时间，必须将 RTOR 寄存器中的 RTO 位域编程为 CWT（字符等待时间）-11 值。此时间以波特率时间单位表示。前一个字符结束后，如果智能卡未在小于 CWT 的时间段内发送新字符，USART 将通过 RTOF 标志和中断（当 RTOIE 位置 1 时）向软件指示此情况。

注：RTO 计数器遵循以下规则开始计数：

- STOP=00 时从停止位结束时开始计数。
- STOP=10 时从第二个停止位结束时开始计数。
- STOP=11 时从 STOP 位结束后开始计数 1 个位的持续时间。
- STOP=01 时从 STOP 位开始时开始计数。

按照智能卡协议定义，*BWT/CWT* 的值从最后一个字符开始（起始位）时定义。必须将 *RTO* 寄存器分别编程为 *BWT-11* 或 *CWT-11*，并考虑最后一个字符本身的长度。块长度计数器用于对 USART 接收到的所有字符进行计数。当 USART 进行发送(TXE=0)时，此计数器复位。块长度由智能卡在块的第三个字节（起始字段）中传达。必须将此值编程到 USARTx_RTOR 寄存器中的 BLEN 字段。使用 DMA 模式时，在块开始之前，必须将此寄存器字段编程为最小值(0x0)。对于该值，在接收到第四个字符后生成中断。软件必须读取 LEN 字段（第三个字节），其值必须从接收缓存区中读取。

在中断驱动接收模式下，块长度可以由软件或者通过编程 BLEN 值来检查。但是在块开始前，可以编程 BLEN 的最大值(0xFF)。接收到第三个字符后，将编程实际值。如果块使用 LRC（纵向冗余校验，1 个结尾字节），则 BLEN=LEN。如果块使用 CRC 机制（2 个结尾字节），则必须编程 BLEN=LEN+1。块总长度（包括起始字段、结尾字段和信息字段）等于 BLEN+4。块结束的信号通过 EOBF 标志和中断（EOBIE 位置 1 时）发送给软件。

如果块长度出现错误，则通过 RTO 中断（字符等待时间上溢）发送块结束信号。

注：错误检查代码(LRC/CRC)必须通过软件计算/验证。

正向约定和反向约定

智能卡协议定义了两种约定：正向约定和反向约定。

正向约定定义为：LSB 在前，逻辑位值 1 对应于线路的 H 状态，奇偶校验为偶校验。要使用此约定，必须编程以下控制位：MSBFIRST=0，DATAINV=0（默认值）。

反向约定定义为：MSB 在前，逻辑位值 1 对应于信号线路的 L 状态，奇偶校验为偶校验。要使用此约定，必须编程以下控制位：MSBFIRST=1，DATAINV=1。

注：将逻辑数据值取反（*0=H，1=L*）时，奇偶校验位将同样取反。

为识别智能卡约定，智能卡会将初始字符 TS 作为 ATR（复位应答）的第一个字符发送。TS 支持两种格式：LHHLLLLLLH 和 LHHLHHHLLH。

- (H)LHHLLLLLLH 建立反向约定：状态 L 编码为值 1，时间分量 2 传送最高有效位（MSB 在前）。按反向约定解码时，传送的字节等于“3F”。
- (H)LHHLHHHLLH 建立正向约定：状态 H 编码为值 1，时间分量 2 传送最低有效位（LSB 在前）。按正向约定解码时，传送的字节等于“3B”。

在 2 到 10 的九个时间分量中，如果有偶数个位设置为 1，则字符的奇偶校验正确。

由于 USART 不了解智能卡使用哪种约定，因此 USART 需要能够识别任意一种模式并相应

操作。模式识别不在硬件中完成，而是通过软件序列完成。此外，假设以正向约定配置 USART（默认）而智能卡以反向约定(TS=LHHLLLLLLH)应答，则 USART 接收到的字节将为“03”，奇偶校验将为奇校验。

因此，有两种方法可用于识别 TS 模式：

方法 1

将 USART 编程为标准智能卡模式/正向约定。这种情况下，TS 模式接收会向智能卡生成奇偶校验错误中断和错误信号。

- 奇偶校验错误中断通知软件智能卡未以正向约定正确应答。之后，软件重新将 USART 编程为反向约定
- 为响应错误信号，智能卡会重试同一 TS 字符，此时重新编程后的 USART 将正确接收该字符或者，为应答奇偶校验错误中断，软件可决定重新编程 USART，同时向智能卡生成新的复位命令，然后再次等待 TS。

方法 2

将 USART 编程为 9 位/无奇偶校验模式，无位反向。在此模式下，按如下方式接收两种 TS 模式中的任一种：

(H)LHHLLLLLLH=0x103->选择反向约定

(H)LHHLLLLLLH=0x13B->选择正向约定

软件根据这两种模式检查接收到的字符，如果其中任意一种匹配，则相应编程 USART 以接收下一个字符。

如果两种都未被识别，则可能复位智能卡以重新开始协商。

25.5.14 USART IrDA SIR ENDEC 模块

仅在支持 IrDA 模式时才涉及本节内容。请参见 *USART* 实现。

通过将 USARTx_CR3 寄存器中的 IREN 位置 1 来选择 IrDA 模式。在 IrDA 模式下，必须将以下位清零：

- USARTx_CR2 寄存器中的 LINEN、 STOP 和 CLKEN 位，
- USARTx_CR3 寄存器中的 SCEN 和 HDSEL 位。

IrDA SIR 物理层规定使用反相归零 (RZI) 调制方案，它以红外光脉冲表示逻辑 0（参见图 25-18）。

SIR 发送编码器用于调制 USART 发出的非归零 (NRZ) 位流。输出脉冲流会发送到外部输出驱动器和红外线 LED。 USART 支持的 SIR ENDEC 比特率最高为 115.2 Kbps。在正常模式下，所发送的脉冲宽度规定为一个位周期的 3/16。

SIR 接收解码器用于解调由红外探测器发出的归零位流，并将接收到的 NRZ 串行位流输出到 USART。在空闲状态下，解码器输入通常为高电平（标记状态）。发送编码器输出的极性与解码器输入相反。当解码器输入为低电平时，会检测到起始位。

- IrDA 是一个半双工通信协议。如果发送器忙（USART 正在向 IrDA 编码器发送数据时），则 IrDA 解码器会忽略 IrDA 接收线上的所有数据；如果接收器忙（USART 正在接收来自 IrDA 解码器的解码数据时），则 IrDA 不会对 USART 发送到 IrDA 的 TX 上的数据进行编码。接收数据时，应避免同时进行发送，因为这样做可能会破坏要发送的数据。
- 0 作为高电平脉冲发送，而 1 作为 0 发送。在正常模式下，脉冲宽度规定为所选位周期的 3/16（参见图 25-19）。
- SIR 解码器用于将兼容 IrDA 的接收信号转换为 USART 的位流。
- SIR 接收逻辑将高电平状态视为逻辑“1”，将低电平脉冲视为逻辑“0”。
- 发送编码器输出的极性与解码器输入相反。 SIR 输出在空闲时处于低电平状态。
- IIrDA 规范要求脉冲容忍值要大于 1.41 μs。可接受的脉冲宽度可通过寄存器设置。接收器端的干扰检测逻辑会滤除宽度小于 2 个 PSC 周期的脉冲（PSC 是在 USARTx_GTPR 中编程的预分频器值）。宽度小于 1 个 PSC 周期的脉冲都将被拒绝，但宽度大于 1 个而小于 2 个周期的脉冲可能被接受也可能被拒绝，而宽度大于 2 个周期的脉冲将被接受作为有效脉冲。当 PSC=0 时， IrDA 编码器/解码器不工作。
- 接收器能够与低功耗发送器进行通信。
- 在 IrDA 模式下， USARTx_CR2 寄存器中的 STOP 位必须配置为“1 个停止位”。

IrDA 低功耗模式

发送器

在低功耗模式下，脉冲宽度不再保持为位周期的 3/16。此时的脉冲宽度为低功耗波特率的

3 倍，最小可为 1.42 MHz。

通常此值是 1.8432 MHz (1.42 MHz < PSC < 2.12 MHz)。低功耗模式下的可编程分频器会对系统时钟进行分频，以达到此值。

接收器

在低功耗模式下接收与在正常模式下接收类似。为进行干扰检测， USART 应丢弃持续时间短于 1 个 PSC 周期的脉冲。只有当持续时间大于 2 个 IrDA 低功耗波特时钟周期（USARTx_GTPR 寄存器中的 PSC 值）时，才是有效低电平。

注： 宽度小于两个但大于一个 PSC 周期的脉冲可能被接受，也可能被拒绝。

接收器的建立时间应由软件进行管理。 *IrDA* 物理层规范规定发送和接收之间至少要经过 *10 ms* 的延迟（*IrDA* 是一个半双工协议）。

图 25-18 **IrDA SIR ENDEC——**框图

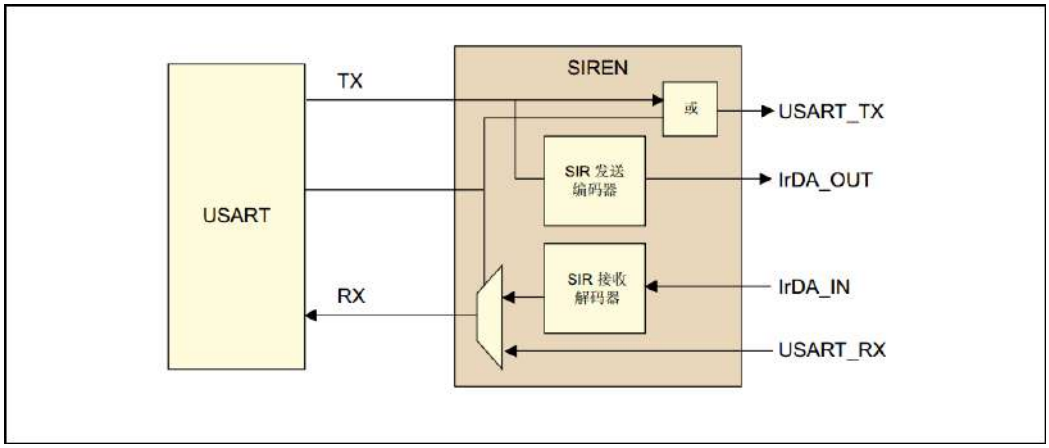
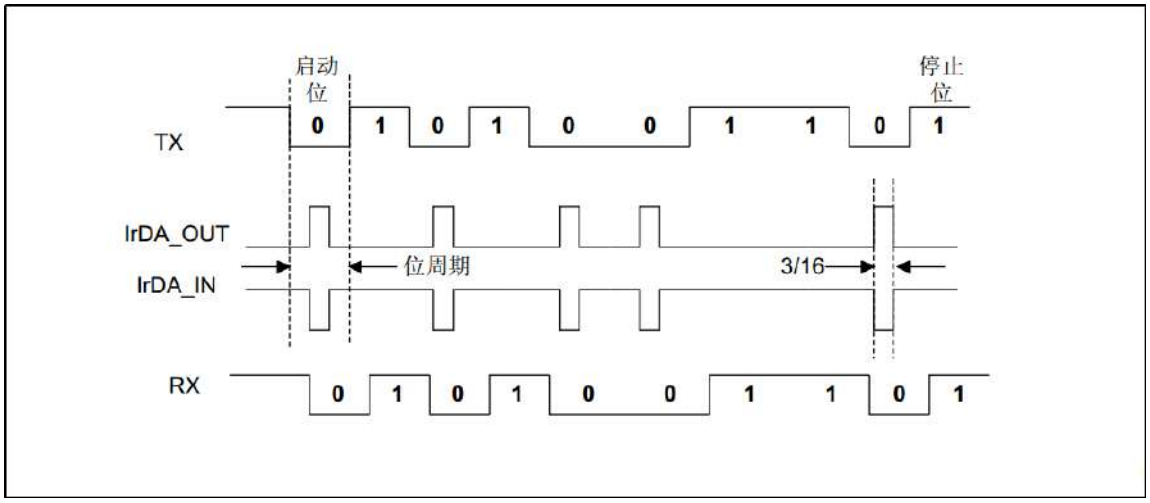


图 25-19 **IrDA** 数据调制 **(3/16)——**正常模式



DMA 模式下的 USART 连续通信

USART 能够使用 DMA 进行连续通信。接收缓冲区和发送缓冲区的 DMA 请求是独立生成的。

注： 要确定是否支持 DMA 模式，请参见 USART 实现。如果不支持 DMA 模式，请按照 USART 发送器或 USART 接收器中的解释说明使用 USART。可以将 USARTx_ISR 寄存器中的 TXE/RXNE 标志清零，从而实现连续通信。

使用 DMA 进行发送

将 USARTx_CR3 寄存器中的 DMAT 位置 1 可以使能 DMA 模式进行发送。当 TXE 位置 1 时，可将数据从 SRAM 区（通过 DMA 外设，请参见直接存储器访问控制器 *(DMA)*）加载到 USARTx_TDR 寄存器。要映射一个 DMA 通道以进行 USART 发送，请按以下步骤操作（x 表示通道编号）：

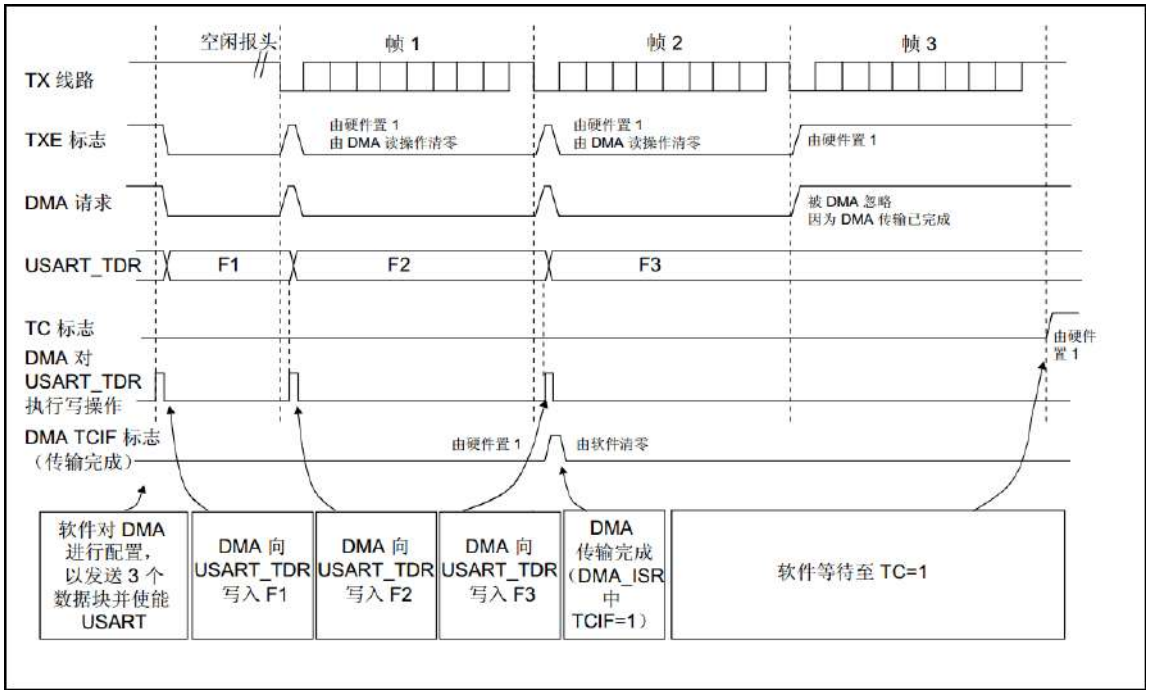
1. 在 DMA 控制寄存器中写入 USARTx_TDR 寄存器地址，将其配置为传输的目标地址。每次发生 TXE 事件后，数据都从存储器移动到此地址。
2. 在 DMA 控制寄存器中写入存储器地址，将其配置为传输的源地址。每次发生 TXE 事件后，数据都从这个存储区域加载到 USARTx_TDR 寄存器中。
3. 在 DMA 控制寄存器中配置要传输的总字节数。
4. 在 DMA 寄存器中配置通道优先级
5. 根据应用的需求，在完成一半或全部传输后产生 DMA 中断。
6. 通过将 USARTx_ICR 寄存器中的 TCCF 位置 1，将 USARTx_ISR 寄存器中的 TC 标志清零。
7. 在 DMA 寄存器中激活该通道。

当达到在 DMA 控制器中设置的数据传输量时，DMA 控制器会在 DMA 通道的中断向量上产生一个中

断。

在发送模式下，DMA 对所有要发送的数据执行了写操作（DMA_ISR 寄存器中的 TCIF 标志置 1）后，可以对 TC 标志进行监视，以确保 USART 通信已完成。在禁止 USART 或进入停止模式前必须执行此步骤，以避免损坏最后一次发送。软件必须等待直到 TC=1。 TC 标志在所有数据发送期间都保持清零状态，然后在最后一帧发送结束时由硬件置 1。

图 25-20 使用 **DMA** 进行发送



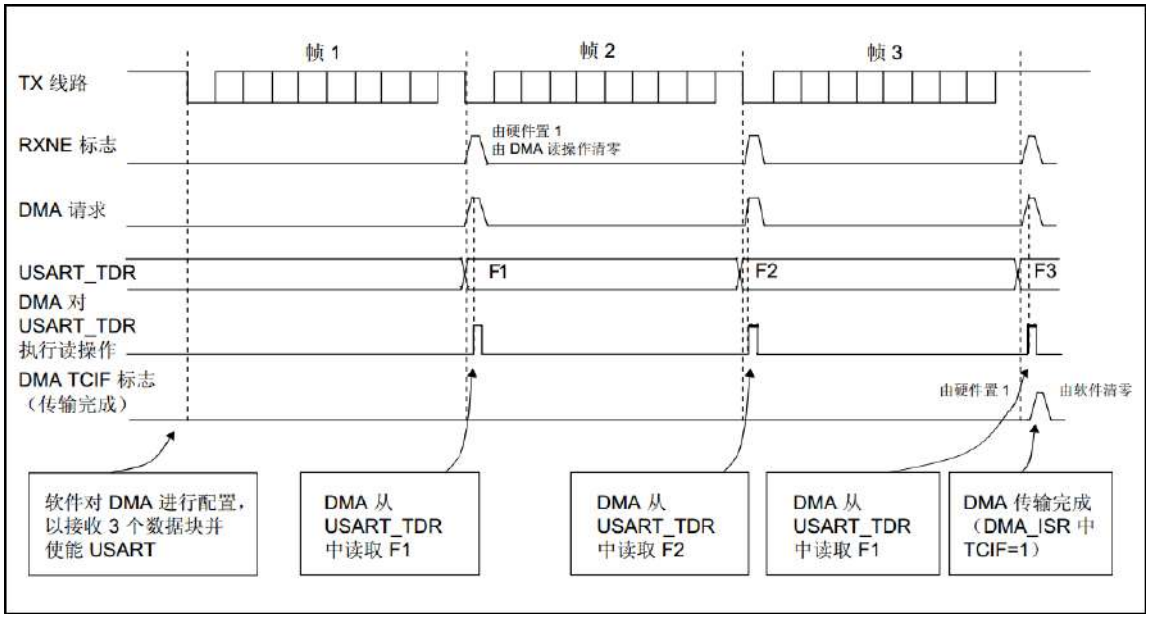
使用 DMA 进行接收

将 USARTx_CR3 寄存器中的 DMAR 位置 1 可以使能 DMA 模式进行接收。接收数据字节时，数据会从 USARTx_RDR 寄存器加载到 SRAM 区域中。要映射一个 DMA 通道以进行 USART 接收，请按以下步骤操作：

1. 在 DMA 控制寄存器中写入 USARTx_RDR 寄存器地址，将其配置为传输的源地址。每次发生 RXNE 事件后，数据都从此地址移动到存储器。
2. 在 DMA 控制寄存器中写入存储器地址，将其配置为传输的目标地址。每次发生 RXNE 事件后，数据都从 USARTx_RDR 寄存器加载到此存储区。
3. 在 DMA 控制寄存器中配置要传输的总字节数。
4. 在 DMA 控制寄存器中配置通道优先级。
5. 根据应用的需求，在完成一半或全部传输后产生中断。
6. 在 DMA 控制寄存器中激活该通道。

当达到在 DMA 控制器中设置的数据传输量时，DMA 控制器会在 DMA 通道的中断向量上产生一个中断。

图 25-21 使用 **DMA** 进行接收



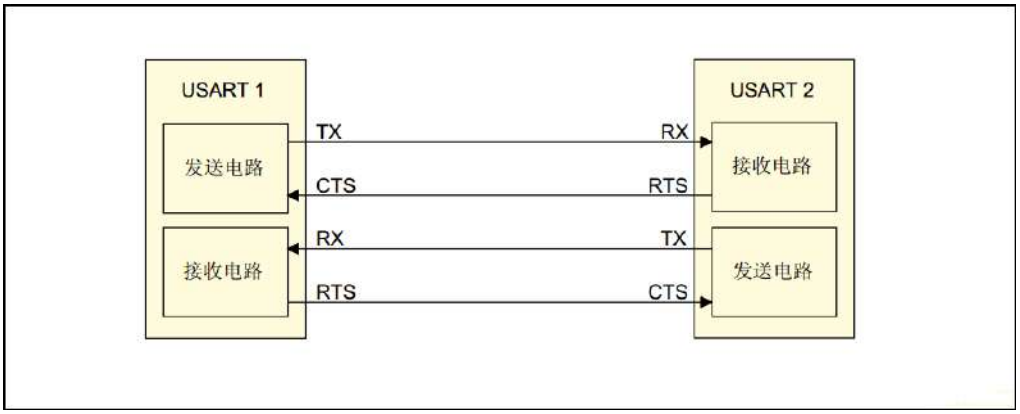
多缓冲区通信中的错误标志和中断生成

在多缓冲区通信中，如果事务中发生任何错误，都会在当前字节后放置错误标志。如果中断使能标志置 1，则会产生中断。在单字节接收过程中，与 RXNE 一同置位的帧错误、上溢错误和噪声标志具有单独的错误标志中断使能位（USARTx_CR3 寄存器中的 EIE 位）；如果该位置 1，在发生其中任何一个错误时，在当前字节后使能中断。

25.5.15 RS232 硬件流控制和 RS485 驱动器使能（使用 USART）

使用 CTS 输入和 RTS 输出可以控制 2 个器件间的串行数据流。 图 25-22 显示了在这种模式下如何连接 2 个器件：

图 25-22 **2** 个 **USART** 间的硬件流控制

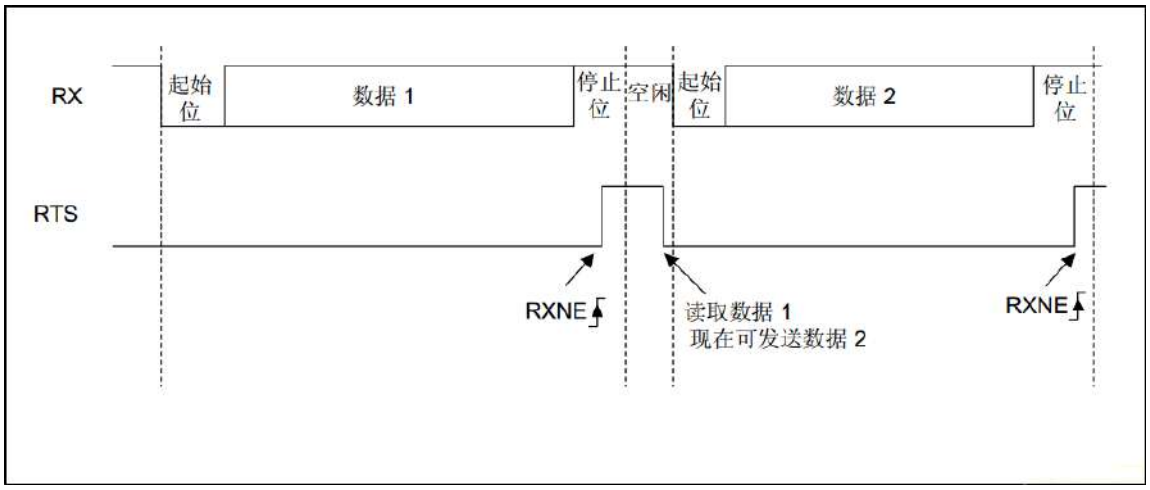


分别向 USARTx_CR3 寄存器中的 RTSE 位和 CTSE 位写入 1，可以分别使能 RS232 RTS 和 CTS 流控制。

RS232 RTS 流控制

如果使能 RTS 流控制 (RTSE=1)，只要 USART 接收器准备好接收新数据，便会将 RTS 变为有效（连接到低电平）。当接收寄存器已满时，会将 RTS 变为无效，表明发送过程会在当前帧结束后停止。 图 25-23 显示了在使能 RTS 流控制的情况下进行通信的示例。

图 25-23 **RS232 RTS** 流控制

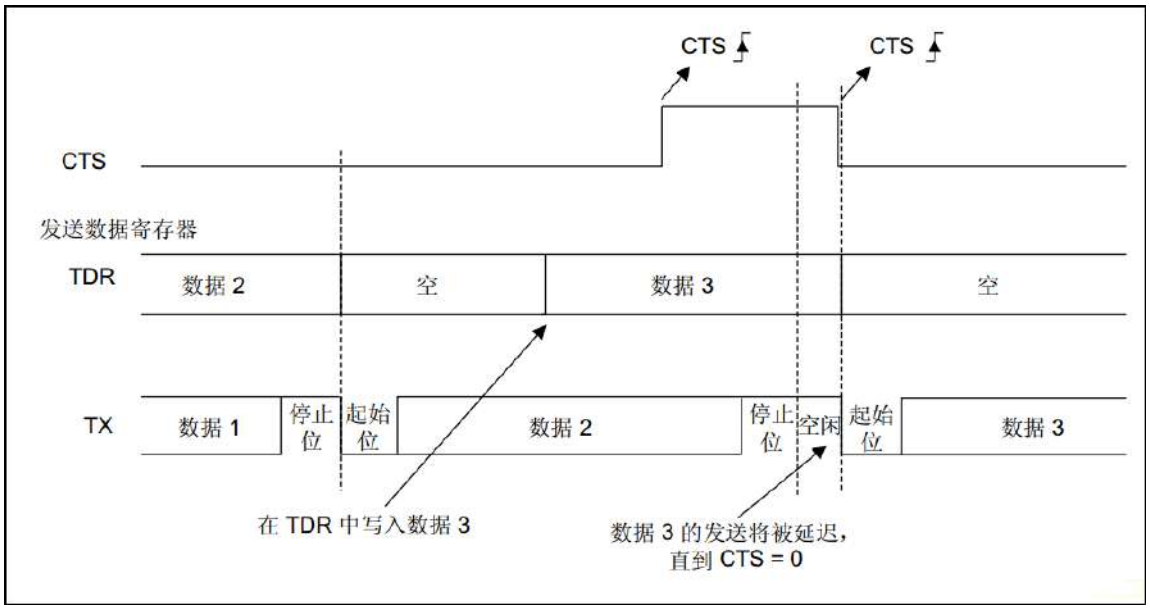


RS232 CTS 流控制

如果使能 CTS 流控制 (CTSE=1)，则发送器会在发送下一帧前检查 CTS。如果 CTS 有效（连接到低电平），则会发送下一数据（假设数据已准备好发送，即 TXE=0）；否则不会进行发送。如果在发送过程中 CTS 变为无效，则当前发送完成之后，发送器停止。

当 CTSE=1 时，只要 CTS 发生变化， CTSIF 状态位便会由硬件自动置 1。这指示接收器是否已准备好进行通信。如果 USARTx_CR3 寄存器中的 CTSIE 位置 1，则会产生中断。图 25-24 显示了在使能 CTS 流控制的情况下进行通信的示例。

图 25-24 **RS232 CTS** 流控制



注： 为正常运行，必须在当前字符结束前至少 3 个 USART 时钟源周期内使能 CTS。此外还应注意，当脉冲短于 2 个 PCLK 周期时无法将 CTSCF 标志置 1。

RS485 驱动器使能

驱动器使能功能可通过将 USARTx_CR3 控制寄存器中的 DEM 位置 1 使能。这样用户便可通过 DE（驱动器使能）信号激活外部收发器控制。使能时间为激活 DE 信号与 START 位开始间的时间。可以通过 USARTx_CR1 控制寄存器中的 DEAT [4:0] 位域编程使能时间。禁止时间为发送的消息中最后一个停止位结束与取消激活 DE 信号间的时间。可以通过 USARTx_CR1 控制寄存器中的 DEDT [4:0] 位域编程禁止时

间。 DE 信号的极性可使用 USARTx_CR3 控制寄存器中的 DEP 位配置。

在 USART 中， DEAT 和 DEDT 以采样时间单位表示（1/8 或 1/16 位持续时间，具体取决于过采样速率）。

使用 USART 从停止模式唤醒

当 UESM 位置 1 且 USART 时钟设置为 HSI 或 LSE 时， USART 能够将 MCU 从停止模式唤醒（请参见复位和时钟控制 (RCC) 部分）。

可使用标准 RXNE 中断将 MCU 从停止模式唤醒。在这种情况下，必须在进入停止模式前将 RXNEIE 位置 1。

也可通过 WUS 位域选择特定中断。

为了能够将 MCU 从停止模式唤醒，必须在进入停止模式前将 USARTx_CR1 控制寄存器中的 UESM 位置 1。

检测到唤醒事件后， WUF 标志会由硬件置 1 并在 WUFIE 位置 1 时生成一个唤醒中断。注： 在进入停止模式前，用户必须确保 *USART* 未在执行传输。 *BUSY* 标志无法确保运行接收期间始终不进入停止模式。

WUF 标志在检测到唤醒事件时置 *1*，而与 *MCU* 处于停止模式还是工作模式无关。

如果在初始化和使能接收器后立即进入停止模式，则必须校验 *REACK* 位以确保 *USART* 确实已使能。

当 *DMA* 用于接收时，它必须在进入停止模式前禁止，并在退出停止模式后重新使能。

从停止模式唤醒功能并非在所有模式下均可用。例如，该功能在 *SPI* 模式下不起作用，因为 *SPI* 仅在主模式下工作。

使用静默模式和停止模式

如果 USART 在进入停止模式前处于静默模式：

- 不得使用空闲检测时从静默模式唤醒，因为空闲检测无法在停止模式下工作。
- 如果使用地址匹配时从静默模式唤醒，则从停止模式唤醒的源也必须是地址匹配。如果 RXNE 标志在进入停止模式时置 1，则接口将在地址匹配时和从停止模式唤醒时保持静默模式。
- 如果 USART 配置为在 START 位检测时将 MCU 从停止模式唤醒， WUF 标志将置 1 但 RXNE 标志不置 1。

当 **USART** 时钟源为 **HSI** 时钟时，确定最大 **USART** 波特率允许从停止模式正确唤醒允许从停止模式正确唤醒的最大波特率取决于：

- 器件数据手册中提供的参数 tWUUSART
- *USART* 接收器对时钟偏差的容差中提供的 USART 接收器的容差。

举例来说： OVER8 = 0， M 位 = 10， ONEBIT = 1， BRR [3:0] = 0000。

在这些条件下，根据表 26-2， USART 接收器的容差为 4.86 %。

DTRA + DQUANT + DREC + DTCL + DWU < USART 接收器的容差

DWU max = tWUUSART / (11 x Tbit Min)

Tbit Min = tWUUSART / (11 x DWU max)

考虑一种理想情况：参数 DTRA、 DQUANT、 DREC 和 DTCL 为 0%， DWU 最大值为 4.86 %。实际上，我们至少需要考虑 HSI 不精确的情况。

假设 HSI 不精确度 = 1 %，则 tWUUSART = 8.1 µs（处于停止模式，主调压器处于运行模式，范围 1）：

DWU max = 4.86 % - 1 % = 3.86 %

Tbit min = 8.1 µs / (9 x 3.86 %) = 23.31 µs。

在这些情况下，允许从停止模式正确唤醒的最大波特率为 1/23.31 µs = 42 K 波特率。

25.6 USART 低功耗模式

表 25-6 低功耗模式对 USART 的影响

模式	说明
睡眠	无影响。 USART 中断可使器件退出睡眠模式。
低功耗运行	无影响。
低功耗睡眠	无影响。 USART 中断可使器件退出低功耗睡眠模式。
停止	当 UESM 位置 1 且 USART 时钟设置为 HSI16 或 LSE 时，USART 能够 将 MCU 从停止模式唤醒。 可使用标准 RXNE 中断将 MCU 从停止模式唤醒。
待机	USART 掉电，当器件退出待机模式后必须重新初始化 USART。

25.7 USART 中断

表 25-7 USART 中断请求

中断事件	事件标志	使能控制位
发送数据寄存器为空	TXE	TXEIE
CTS 中断	CTSIF	CTSIE
发送完成	TC	TCIE
接收数据寄存器不为空（已准备好读取数据）	RXNE	RXNEIE
检测到溢出错误	ORE	
检测到空闲线路	IDLE	IDLEIE
奇偶校验错误	PE	PEIE
LIN 断路	LBDF	LBDIE
多缓冲区通信中的噪声标志、溢出错误和帧错误。	NF 或 ORE 或 FE	EIE
字符匹配	CMF	CMIE
接收器超时	RTOF	RTOIE
块结束	EOBF	EOBIE
从停止模式唤醒	WUF(1)	WUFIE

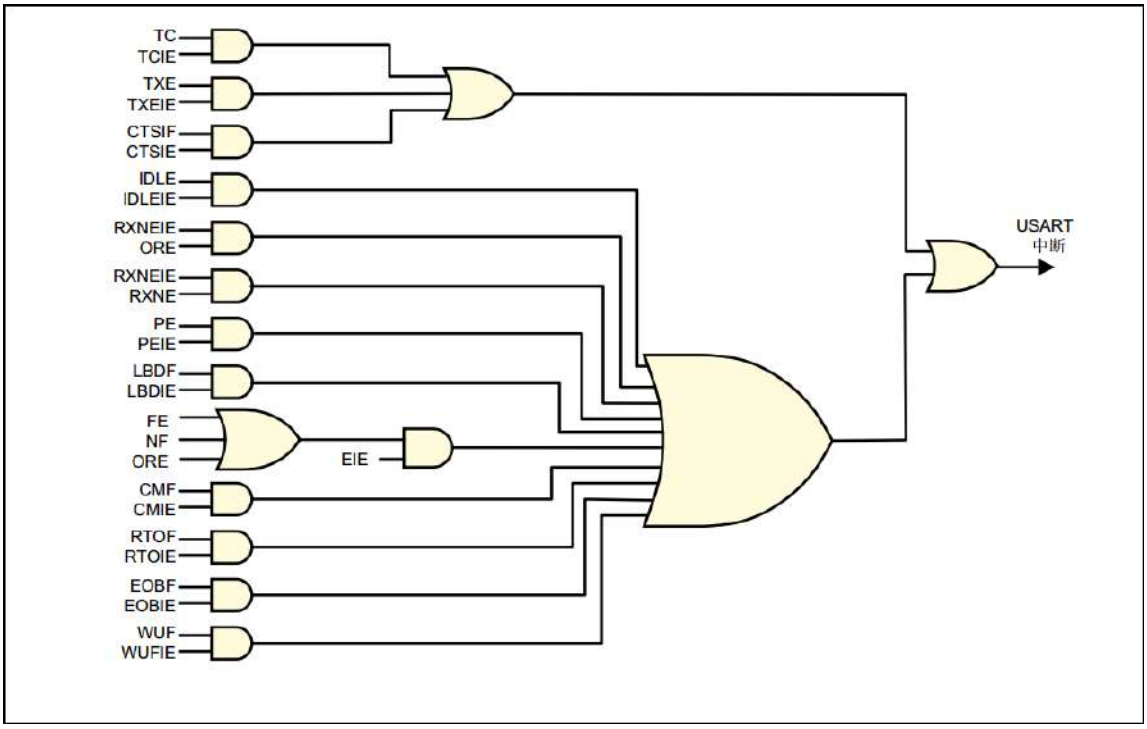
1. WUF 中断仅在停止模式下有效。

USART 中断事件被连接到相同的中断向量（请参见图 25-25）。

- 发送期间：发送完成、清除以发送、发送数据寄存器为空或帧错误（智能卡模式下）中断。
- 接收期间：空闲线路检测、上溢错误、接收数据寄存器不为空、奇偶校验错误、 LIN 断路检测、噪声标志、帧错误、字符匹配等。

如果相应的使能控制位置 1，则这些事件会生成中断。

图 25-25 **USART** 中断映射图



25.8 USART1/2 寄存器

25.8.1 控制寄存器 1（USARTx_CR1）

地址偏移: 0x00

复位值: 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	M1	EOBIE	RTOIE	DEAT[4:0]					DEDT[4:0]				
			rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
0	0	0	0	0		0	0	0	0	0	0	0	0	0	0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OVER8	CMIE	MME	M0	WAKE	PCE	PS	PEIE	TXEIE	TCIE	RXNEIE	IDLEIE	TE	RE	Res.	UE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位 31:29	保留，必须保持复位时的值
位 28	M1：word length M[1:0]=00,1 个起始位，8 个数据位，n 个停止位 M[1:0]=01,1 个起始位，9 个数据位，n 个停止位 M[1:0]=10,1 个起始位，7 个数据位，n 个停止位
位 27	EOBIE: 块尾中断使能 由软件置 1 和清零。 0: 中断禁止 1: 当 USART_ISR 寄存器中的 EOBF 标志被置 1 时引发 USART 中断

位 26	RTOIE: 接收超时中断接收超时中断使能 由软件置 1 和清零。 0: 中断禁止 1: 当 USART_ISR 寄存器中的 RTOF 标志被置 1 时引发 USART 中断。
位 25:21	DEAT[4:0]: 驱动使能提前时间 这个 5 位数字定义 DE（驱动器使能）信号激活和第一个发送字节的起始位的时间 间隔。 它以采样时间为单位（1/8 或者 1/16 位时间，由过采样率决定） 这个位域只能在 USART 未被使能的时候（UE=0）改写。
位 20:16	DEDT[4:0]: 驱动使能滞后时间 这个 5 位数字是一个发送消息的最后一个字节的停止位和释放 DE 信号之间的时间 间隔。 它以采样时间为单位（1/8 或者 1/16 位时间，由过采样率决定） 如果 USART_TDR 寄存器在 DEDT 时间内被改写，新的数据只会在 DEDT 和 DEAT 时间都过去之后才会被发送。 这个位域只能在 USART 未被使能的时候（UE=0）改写。
位 15	OVER8: 过采样模式 0: 16 倍过采样 1: 8 倍过采样 这个位域只能在 USART 未被使能的时候（UE=0）改写。
位 14	CMIE: 字符匹配中断使能 由软件置 1 和清零。 0: 中断禁止 1: 当 USART_ISR 寄存器中的 CMF 标志被置 1 时引发 USART 中断。
位 13	MME: 静默模式使能 这个位开启 USART 的静默模式功能。当置为 1 时，USART 可以在活动模式和静默 模式之间切换，和 WAKE 位的定义一样， 由软件置 1 和清零。 0: 接收器长期处于活动模式 1: 接收器可以在活动模式和静默模式间切换。
位 12	M: 字长 这个位决定串口字长。 由软件置 1 和清零。 0: 1 个起始位，8 位数据位，n 个停止位 1: 1 个起始位，9 个数据位，n 个停止位 这个位域只能在 USART 未被使能的时候（UE=0）改写。
位 11	WAKE: 接收器唤醒方式 这个位决定 USART 从静默模式唤醒的方式。 由软件置 1 和清零。 0: 空闲线 1: 地址标记 这个位域只能在 USART 未被使能的时候（UE=0）改写。
位 10	PCE: 校验控制使能 这个位选择硬件校验控制（产生和检测）功能。 当校验控制被打开，计算好的校验 位被插入到最高位（M=1 时是第九位，M=0 时是第八位），并检测接收数据的校验 位。由软件置 1 和清零。一旦这个位被置 1，在当前字节之后就激活了校验控制（在 收发的时候都有）。 0: 校验控制禁止 1: 校验控制使能 这个位域只能在 USART 未被使能的时候（UE=0）改写。
位 9	PS: 校验选择 这个位选择在校验生成和检测功能被打开的时候（PCE=1）使用奇校验还是使用偶 校验。 由软件置 1 和清零。 校验方式会在当前字节结束后生效。 0: 偶校验 1: 奇校验 这个位域只能在 USART 未被使能的时候（UE=0）改写。
位 8	PEIE: 校验错误中断使能 由软件置 1 和清零。 0: 中断禁止 1: 在 USART_ISR 寄存器中的 PE 被置 1 的时候会产生 USART 中断。
位 7	TXEIE: 发送寄存器空中断使能 由软件置 1 和清零。 0: 中断禁止 1: 在 USART_ISR 寄存器中的 TXE 被置 1 的时候会产生 USART 中断

位 6	TCIE: 发送完毕中断使能 由软件置 1 和清零。 0: 中断禁止 1: 在 USART_ISR 寄存器中的 TC 位被置 1 的时候会产生 USART 中断
位 5	RXNEIE: 接收寄存器非空中断使能 由软件置 1 和清零。 0: 中断禁止 1: 在 USART_ISR 寄存器中的 ORE 或者 RXNE 被置 1 的时候会产生 USART 中断。
位 4	IDLEIE: 空闲中断使能 由软件置 1 和清零。 0: 中断禁止 1: 在 USART_ISR 寄存器中的 IDLE 位被置 1 的时候会产生 USART 中断
位 3	TE: 发送器使能 这个位打开发送器。 由软件置 1 和清零。 0: 发送器关闭 1: 发送器打开 2: 当 TE 被置为 1 后，和发送开始之间有 1 个位的延迟时间。
位 2	RE: 接收器使能 这个位打开接收器。 由软件置 1 和清零。 0: 接收器被关闭 1: 接收器被打开并开始等待起始位
位 1	UESM: USART 在 Stop 模式下使能 当这个位为零，USART 不能够将 MCU 从 Stop 模式下唤醒。当这个位为 1 时， USART 可以将 MCU 从 Stop 模式唤醒，条件是 USART 的时钟选择位 HSI 或 LSE。由软件置 1 和清零。 0: USART 不能从 Stop 模式中唤醒 MCU。 1: USART 可以从 Stop 模式中唤醒 MCU。 当这个功能被打开，USART 的时钟源必 须为 HSI 或者是 LSE
位 0	UE: USART 使能 当这个位被清零，USART 的预分频器和输出都立即停止，并且当前的操作也被取消。 对 USART 的设置都不会丢，但 USART_ISR 中所有的状态标志都会被复位。 由软 件置 1 和清零。 0: USART 预分频器和输出关闭，低功耗模式 1: USART 开启

25.8.2 控制寄存器 2（USARx_CR2）

地址偏移: 0x04

复位值: 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADD[7:4]				ADD[3:0]				RTOEN	ABRMOD[1:0]		ABREN	MSBFIRST	DATAINV	TXINV	RXINV
rw				rw				rw	rw		rw	rw	rw	rw	rw
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWAP	LINEN.	STOP[1:0]		CLKEN	CPOL	CPHA	LBCL	Res.	LBDIE	LBDL	ADDM7	Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw				
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位 31:28	ADD[7:4]: USART 的节点地址 这个位域给出 USART 节点的地址或等待确认的字符码。 这用在多机通讯并且进入静默状态或者 Stop 模式的时候，用于 7 位地址标记的检测。 发送器发出的字符的最高位应该为 1。 也用在正常接收过程中的字符检测中，这时 不打开静默状态（例如，在 ModBus 协议下对块尾的检测）。 这个时候，整个收到 的 8 位字节与 ADD[7:0] 进行全面比较，如果匹配，将会引起 CMF 标志被硬件置起。 这个位域只能在接收器被关闭（RE=0）或者在 USART 被关闭的时候（UE=0）才 能改写。
位 27:24	ADD[3:0]: USART 的节点地址 这个位域给出 USART 节点的地址或等待确认的字符码。 这用在多机通讯并且进入静默状态或者 Stop 模式的时候，用于 7 位地址标记的检测。 这个位域只能在接收器被关闭（RE=0）或者在 USART 被关闭的时候（UE=0）才 能改写。
位 23	RTOEN: 接收器超时检测功能使能 由软件置 1 和清零。 0: 接收器超时检测功能关闭 1: 接收器超时检测功能开启 当这个功能开启，只要 RX 线发现空闲（不是接收）达到 RTOR 寄存器（接收超时 寄存器）中设置的时间长度后，USART_ISR 寄存器中的 RTOF 标志会被硬件置 1。
位 22:21	ABRMOD[1:0]: 自动波特率检测模式 由软件设置或清零 00: 对起始位的测量被用来检测波特率。 01: 使用下降沿对下降沿的测量。 （接收数据必须以单个的 1 作为开头，帧格式为 Start 1 0 xxxxxx） 10: 保留 11: 保留 这个位域只能在 ABREN=0 或者 USART 未被使能的时候（UE=0）改写。
位 20	ABREN: 自动波特率检测使能 由软件置 1 和清零。 0: 自动波特率检测被禁止。 1: 自动波特率检测被打开
位 19	MSBFIRST: 高位在前 由软件置 1 和清零。 0: 数据在发送和接收的时候，采用起始位在前，后面跟着第 0 位的顺序。 1: 数据在发送和接收的时候，采用起始位在前，后面跟着最高位（位 7 或者 8）的顺序。 这个位域只能在 USART 未被使能的时候（UE=0）改写。
位 18	DATAINV: 二进制数反向 由软件置 1 和清零。 0: 数据寄存器中的逻辑数据在发送和接收的时候，采用正 / 直接逻辑。 (1=H, 0=L) 1: 数据寄存器中的逻辑数据在发送和接收的时候，采用负 / 反向逻辑。 (1=L, 0=H). 校验位也一样反向。这个位域只能在 USART 未被使能的时候（UE=0）改写。
位 17	TXINV: TX 脚有效电平反向 由软件置 1 和清零。 0: TX 脚信号工作于标准逻辑电平 (VDD =1/idle, Gnd=0/mark) 1: TX 脚信号被反向。 ((VDD =0/mark, Gnd=1/idle). 这可以用于 TX 线上带有外部反 相器的时候。 这个位域只能在 USART 未被使能的时候（UE=0）改写。
位 16	RXINV: RX 脚有效电平反向 由软件置 1 和清零。 0: RX 脚信号工作于标准逻辑电平 (VDD =1/idle, Gnd=0/mark) 1: RX 脚信号被反向。 ((VDD =0/mark, Gnd=1/idle). 这可以用于 RX 线上带有外部反 相器的时候。 这个位域只能在 USART 未被使能的时候（UE=0）改写。
位 15	SWAP: 交换 TX/RX 引脚 由软件置 1 和清零。 0: TX/RX 引脚按照标准引脚分配来使用 1: TX 和 RX 的引脚功能交换使用。 这用于和其它 UART 口进行交叉互联的时候。 这个位域只能在 USART 未被使能的时候（UE=0）改写。

位 14	LINEN: LIN 模式使能 由软件置 1 和清零。 0: LIN 模式禁止 1: LIN 模式使能 LIN 模式打开后，可以具备发送 LIN 同步断开（13 个低位）的功能，用 USART_ CR1 寄存器的 SBKRQ 位实现，同时还具备 LIN 同步断开信号的检测功能。 这个位域只能在 USART 未被使能的时候（UE=0）改写。
位 13:12	STOP[1:0]: 停止位 这些位用来定制停止位的个数。 00: 1 个停止位： 01: 保留 10: 2 个停止位： 11: 1.5 个停止位 这个位域只能在 USART 未被使能的时候（UE=0）改写。
位 11	CLKEN: 时钟使能 这个位用来打开 SCLK 引脚的功能 0: SCLK 引脚被禁止 1: SCLK 引脚被使能 这个位域只能在 USART 未被使能的时候（UE=0）改写。
位 10	CPOL: 时钟极性 这个位允许用户选择同步模式下 SCLK 引脚上的时钟输出的极性。 它连同 CPHA 位 一起决定所需要的时钟 / 数据时序关系。 0: 在没有数据传输的时候保持低电平 1: 在没有数据传输的时候保持高电平 这个位域只能在 USART 未被使能的时候（UE=0）改写。
位 9	CPHA: 时钟相位 这个位允许用户选择同步模式下 SCLK 引脚上的时钟输出的相位。 它连同 CPOL 位 一起决定所需要的时钟 / 数据时序关系。（见图 238 和图 239） 0: 第一个时钟沿对准第一位数据 1: 第二和时钟沿对准第一位数据 这个位域只能在 USART 未被使能的时候（UE=0）改写。
位 8	LBCL: 末位时钟脉冲 这个位用来选择在同步模式下，SCLK 脚上传输最后一个位（MSB）的时候是否必 须输出时钟脉冲。 0: SCLK 脚上在传输末位数据的时候不输出时钟脉冲。 1: SCLK 脚上在传输末位数据的时候输出时钟脉冲。 警告：末位是第 8 位还是第 9 位，取决于 USART_CR1 寄存器中的 M 位的设置。 这个位域只能在 USART 未被使能的时候（UE=0）改写。
位 7	保留，必须保持复位时的值。
位 6	LBDIE: LIN 断开信号检测中断使能 断开中断屏蔽（利用断开分隔符来检测） 0: 中断禁止 1: 在 USART_ISR 寄存器中的 LBDF 被置 1 的时候会产生 USART 中断
位 5	LBDL: LIN 断开检测长度 这个位用来选择使用 11 位还是 10 位断开检测。 0: 10 位断开检测 1: 11 位断开检测 这个位域只能在 USART 未被使能的时候（UE=0）改写。
位 4	ADDM7:7 位地址检测或 4 位地址检测选择 这个位用来选择使用 4 位地址检测还是 7 位地址检测。 0: 4 位地址检测 1: 7 位地址检测（8 位数据模式下） 这个位域只能在 USART 未被使能的时候（UE=0）改写。
位 3:0	保留，必须保持复位时的值。

25.8.3 控制寄存器 3（USARTx_CR3）

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
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WUFIE	WUS		SCARCNT[2:0]			Res.
									rw	rw	rw	rw	rw	rw	rw
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DEP	DEM	DDRE	OVERDIS	ONEBIT	CTSIE	CTSE	RTSE	DMAT	DMAR	SCEN	NACK	HDSEL	IRLP	IREN	EIE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位 31:23	保留，必须保持复位时的值。
位 22	WUFIE: 从 Stop 模式唤醒中断使能 由软件置 1 和清零。 0: 中断禁止 1: 在 USART_ISR 寄存器中的 WUF 被置 1 的时候会产生 USART 中断
位 21:20	WUS[1:0]: 从 Stop 模式唤醒中断标志选择 这个位域指定激活 WUF 标志的事件。 00: 在发生地址匹配事件的时候激活 WUF 01: 保留 10: WUF 在检测到起始位的时候激活 11: WUF 在得到接受数据寄存器非空事件时激活 这个位域只能在 USART 未被使能的时候（UE=0）改写。
位 19:17	SCARCNT[2:0]: 智能卡模式重试计数器 这个位域指定智能卡模式中接收和发送的重试次数。在发送模式下，它指定的是在 产生发送错误（FE=1）之前自动重试发送的次数。 在接收模式下，它指定的是在产生接收错误事件前（RXNE 和 PE=1）所作的错误接 收的尝试的次数。 这个位域只能在 USART 未被使能的时候（UE=0）改写。 当 USART 被使能，这个位域只允许写成 0x0，这是为了避免盲目的自动重发数据。 0x0: 重发功能关闭 - 在发送模式下不进行自动重发操作。 0x1 to 0x7: 自动重传的尝试次数（在提示错误之前）
位 16	保留，必须保持复位时的值。
位 15	DEP: 驱动使能输出脚的极性选择 0: DE 信号高有效 1: DE 信号低有效 这个位域只能在 USART 未被使能的时候（UE=0）改写。
位 14	DEM: 驱动器使能模式 它允许用户通过 DE（驱动使能）信号来激活外部收发器的控制端。 0: DE 功能被禁止 1: DE 功能被打开。 DE 信号在 RTS 脚输出。 这个位域只能在 USART 未被使能的时候（UE=0）改写。
位 13	DDRE: 在接收错误的时候禁止 DMA 0: 在发生接收错误的时候不禁止 DMA。 相应的错误标志被置 1，但 RXNE 仍保持 零以阻止数据溢出覆盖。 作为结果，不会发起 DMA 请求，所以错误的数据不会被 传输（因为没有 DMA 请求），但下一个正确的数据会被传送。 （用于智能卡模式） 1: 在发生接收错误之后，DMA 被关闭。相应的错误标志会随着 RXNE 的置 1 而被置 1。 DMA 请求会被屏蔽，直到相关的错误标志被清零。 这意味着软件必须先禁止 DMA 请求（DMAR=0）或者在清零错误标志前先清零 RXNE 标志。 这个位域只能在 USART 未被使能的时候（UE=0）改写。

位 12	OVRDIS: 溢出检测禁止 这个位用于禁止对接收溢出现象的检测。 0: 当之前接收到的数据没有在新接收到数据之前读走时，会引起溢出错误标志 ORE 被硬件置 1。 1: 溢出检测功能关闭。 如果在新的接收数据到来时，RXNE 标志仍然是 1，但 ORE 标志还不是 1 时，新的数据会将 USART_RDR 中以前的内容覆盖掉。 这个位域只能在 USART 未被使能的时候（UE=0）改写。
位 11	ONEBIT: 单次采样方式使能 这个位允许用户选择采样方式。 当选择单次采样方式的时候，噪声监测标志（NF） 就被禁止了。 0: 三次采样方式 1: 单次采样方式 这个位域只能在 USART 未被使能的时候（UE=0）改写。
位 10	CTSIE: CTS 中断使能 0: 中断禁止 1: 在 USART_ISR 寄存器中的 CTSIF 被置 1 的时候会产生 USART 中断
位 9	CTSE: CTS 功能使能 0: 关闭 CTS 硬件流控 1: 打开 CTS 硬件流控，只有在 nCTS 输入上收到有效信号（被拉低）时才会发送数据。 如果在数据传送时 nCTS 输入变为无效，会在数据发送完成后再停。如果写数据到 发送寄存器的时候，nCTS 处于无效状态，那么这个数据的发送会被推迟直到 nCTS 信号变成有效。 这个位域只能在 USART 未被使能的时候（UE=0）改写。
位 8	RTSE: RTS 使能 0: 关闭 RTS 硬件流控 1: RTS 输出使能，只会在有接收空间的时候才请求下一个数据。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（拉低）。 这个位域只能在 USART 未被使能的时候（UE=0）改写。
位 7	DMAT: DMA 发送使能 该位由软件置 1 和清零 1: 为发送数据使能 DMA 模式 0: 关闭发送 DMA 模式
位 6	DMAR: DMA 接收使能 该位由软件置 1 和清零 1: 为接收数据使能 DMA 模式 0: 关闭接收 DMA 模式
位 5	SCEN: 智能卡模式使能 该位用于打开智能卡模式。 0: 关闭智能卡模式 1: 打开智能卡模式 这个位域只能在 USART 未被使能的时候（UE=0）改写。
位 4	NACK: 智能卡 NACK 发送使能 0: 出现校验错误的时候不发送 NACK 1: 出现校验错误的时候，发送 NACK 这个位域只能在 USART 未被使能的时候（UE=0）改写。
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 这个位域只能在 USART 未被使能的时候（UE=0）改写。
位 2	IRLP: IrDA 低功耗模式选择 用来选择 IrDA 的普通模式还是低功耗模式 0: 正常模式 1: 低功耗模式 这个位域只能在 USART 未被使能的时候（UE=0）改写。
位 1	IREN: 红外模式使能 由软件置 1 和清零。 0: 不使能红外模式 1: 使能红外模式 这个位域只能在 USART 未被使能的时候（UE=0）改写。

位 0	EIE: 错误中断使能 在允许帧错误，溢出错误或噪声错误产生中断请求时要打开这个开关。 0: 中断禁止 1: 当 USART_ISR 寄存器中的 FE=1 或 ORE=1 或 NF=1 时，会产生中断。

25.8.4 波特率寄存器（USARTx_BRR）

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
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

位 31:16	保留，必须保持复位时的值。
位 15:4	BRR[15:4]: USARTDIV 的整数部分 这 12 位定义 USART 分频器除法因子的整数部分 USARTDIV[15:4]
位 3:0	BRR[3:0]: USARTDIV 的小数部分 当 OVER8=1，BRR[3:0]= USARTDIV[3:0];当 OVER8=0，BRR[3:0]= USARTDIV[3:0]右移 1 位

25.8.5 保护时间和预分频器寄存器（USARTx_GTPR）

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
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位 31:16	保留，必须保持复位时的值。
位 15:8	GT[7:0]: 保护时间值 这个位域用来设置保护时间长度，以波特时钟为单位。 在智能卡模式下要用到。 在保护时间过去之后才会设置发送完成标志。 这个位域只能在 USART 未被使能的时候（UE=0）改写。

位 7:0	PSC[7:0]: 预分频器值 - 在红外低功耗和正常模式下： PSC[7:0] = IrDA 正常和低功耗模式波特率 对 USART 时钟源进行分频以获得低功耗模式下的频率： 时钟源按寄存器给定的值来分频（8 位有效）： 00000000: 保留 - 不要写这种无聊的值 00000001: 1 分频 00000010: 2 分频 ... - 智能卡模式： PSC[4:0]: 预分频器值 用于设定对 USART 时钟源的分频系数，得到智能卡时钟。按寄存器中的值（低 5 位有效）乘以 2 得到的值作为分频系数来分频： 00000: 保留 - 不要写这种无聊的值 00001: 2 分频 00010: 4 分频 00011: 6 分频 ... 这个位域只能在 USART 未被使能的时候（UE=0）改写。

25.8.6 接收超时寄存器（USARTx_RTOR）

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
BLEN[7:0]								RTO[23:16]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RTO[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位 31:24	BLEN[7:0]: 块长度 这个位域给出了智能卡模式 T=1 的接收时的块长度。 这个值等于 信息块字符数 + 结束部分（1-LEC/2-CRC）-1 。 例如： BLEN = 0 -> 0 个信息字符 + LEC BLEN = 1 -> 0 个信息字符 + CRC BLEN = 255 -> 254 个信息字符 +CRC（总共 256 个字符） 智能卡模式中， 当 TXE=0，会导致块长度计数器清零。 这个位域也可以在其它模式中使用。 这时，块长度计数器在 RE=0 的时候清零 和 / 或 在 EOBCF 位被写 1 的时候清零。
位 23:0	RTO[23:0]: 接收超时值 这个位域给出了接收超时的设置，单位是波特时钟的时长。 标准模式下，最后一个字节接收后，在整个 RTO 值规定的时长内，再也没有检测到 新的起始位的时候，RTOF 标志会被硬件置 1。 在智能卡模式中，这个值被用来实现 CWT 和 BWT。 详细信息参见智能卡相关章节。 这里，超时测量时从最后一个字节的其实位开始算的。

25.8.7 请求寄存器（USARTx_RQR）

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
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXFRQ	RXFRQ	MMRQ	SBKRQ	ABRRQ
											w_r0	w_r0	w_r0	w_r0	w_r0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位 31:5	保留，必须保持复位时的值。
位 4	TXFRQ: 发送数据清空请求 对这个位写 1 会直接令 TX 标志被硬件置 1 这里允许取消数据发送。 这个位只能在智能卡模式下使用，当数据由于所发生的错 误导致还没发出去，USART_ISR 寄存器的 FE 标志是 1 的时候用。 如果 USART 不支持智能卡模式，该位为保留位并由硬件强制为零。
位 3	RXFRQ: 接收数据清空请求 对这个位写 1 会直接令 RXNE 标志被硬件清零 这里允许直接丢弃还没有读的接收数据，免得被提示溢出错误。
位 2	MMRQ: 静默模式请求 对这个位写 1 将导致 USART 进入静默模式，同时清除 RWU 标志。
位 1	SBKRQ: 请求发送断开字符 对这个位写 1 会令 SBKF 标志置 1，并在发送状态机可用的时候向线路发出一个断开字符。
位 0	ABRRQ: 自动波特率检测请求 向这个位写 1 会清除 USART_ISR 中的 ABRF 标志，并在下一次数据接收的时候开 始一次自动波特率测量。

25.8.8 中断和状态寄存器（USARTx_ISR）

地址偏移: 0x1C

复位值: 0x00C0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REACK	TEACK	WUF	RWU	SBKF	CMF	BUSY
									r	r	r	r	r	r	r
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ABRF	ABRE	Res.	EOBF	RTOF	CTS	CTSIF	LBDF	TXE	TC	RXNE	IDLE	ORE	NF	FE	PE
r	r		r	r	r	r	r	r	r	r	r	r	r	r	r
0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0

位 31:23	保留，必须保持复位时的值。
位 22	REACK: 接收使能通知标志 这个位有硬件控制，当接收使能信号被 USART 读到的时候，会给出一个回执。这可以在进入 Stop 模式之前，用来确认 USART 是不是已经准备好接收了。
位 21	TEACK: 发送使能通知标志 这个位有硬件控制，当发送使能信号被 USART 读到的时候，会给出一个回执。 这可以在写 USART_CR1 寄存器的 TE=0 以产生一个空闲帧请求，跟着又将 TE 写 成 1 的时候，用于保证 TE=0 的最小周期的时候用。
位 20	WUF: 从 Stop 模式唤醒标志 当检测到唤醒事件时由硬件置 1。具体时间有 WUS 位域来定义。由软件向 USART_ICR 寄存器的 PECF 位写 1，可以清除这个标志。 如果 USART_CR3 寄存器的 WUF=1，会产生中断请求。

位 19	RWU: 接收器从静默模式唤醒 这个位表示 USART 处于静默模式时，当唤醒和静默模式切换时，由硬件清零和置 1。 静默模式控制顺序（地址还是空闲）用 USART_CR1 寄存器的 WAKE 位来选择。 如果选择由空闲信号唤醒，那这个位就只能由软件置 1 了，方法是向 USART_RQR 寄存器置 1 0: 接收器处于活动模式 1: 接收器处于静默模式
位 18	SBKF: 断开信号发送标志 这个位表示当要求发送一个断开字符时，由软件置 1,方法是向 USART_CR3 寄存 器的 SBKRQ 位写 1。 当断开字符的停止位传出后，由硬件自动清零。 0: 没发送断开字符 1: 断开字符将会被发送
位 17	CMF: 字符匹配标志 当收到一个字符和 ADD[7:0] 中设置的内容相同时，由硬件置 1。 由软件向 USART_ ICR 寄存器的 CMCF 位写 1，可以清除这个标志。 如果 USART_CR1 寄存器中的 CMIE 位是 1，就会产生中断请求。 0: 没发现字符匹配 1: 有发现字符匹配
位 16	BUSY: 忙标志 由硬件置 1 和清零。 当 RX 线在通讯时（成功检测到起始位），这个位被硬件置 1。 接收结束后（不管成功与否）会由硬件清零。 0: USART 处于空闲（没接收） 1: 正在接收数据
位 15	ABRF: 自动波特率检测标志 当自动波特率功能打开时（RXNE 也被置 1，如果 RXNEIE=1 产生中断）或者当自 动波特率操作不成功时（ABRE，RXNE 和 FE 也都被置 1 的时候），这个位被硬件 置 1。 开始一轮新的波特率检测（向 USART_RQR 寄存器的 ABRQ 写 1）的时候会被清除。
位 14	ABRE: 自动波特率检测错误 在波特率测量失败的时候（超量程或者字符比较失败）由硬件置 1。 由软件清零，方法是向 USART_CR3 寄存器的 ABRRQ 位写 1。
位 13	保留，必须保持复位时的值。
位 12	EOBF: 块结束标志 当一个完整的块被接收时由硬件置 1（例如 T=1 智能卡模式中）。当收到的字节数（从 块开始，包括序言部分）大于等于 BLEN+4 的时候完成检测。如果 USART_CR2 寄存器的 EOBIE=1，会产生中断请求。由软件向 USART_ICR 寄存器的 EOBCF 位写 1，可以清除这个标志。 0: 还没到块结束 1: 块结束（足够字节数）已经到了
位 11	RTOF: 接收超时标志 如果没有通讯的时间长度达到了 RTOR 寄存器中设定的超时值，这个位会被硬件置 1。 由软件向 USART_ICR 寄存器的 RTOCF 位写 1，可以清除这个标志。 如果 USART_CR2 寄存器的 RTOIE=1，会产生中断请求。 在智能卡模式中，这个超时相当于 CWT 或 BWT 计时。 0: 尚未超时 1: 已经达到超时
位 10	CTS: CTS 标志 该位由硬件置 1 和清零 这是 nCTS 输入脚状态的反向拷贝。 0: nCTS 线是高 1: nCTS 线是低
位 9	CTSIF: CTS 中断标志 如果 CTSE 位为 1,那么当 nCTS 输入状态变化的时候,由硬件置 1。由软件向 USART_ICR 寄存器的 CTSCF 位写 1，可以清除这个标志。 如果 USART_CR3 寄存器的 CTSIE=1，会产生中断请求。 0: nCTS 线的状态无变化 1: nCTS 线的状态有变化
位 8	LBDF: LIN 断开检测 当检测到 LIN 断开字符时由硬件置 1。由软件向 USART_ICR 寄存器的 LBDCF 位写 1， 可以清除这个标志。如果 USART_CR2 寄存器的 LBDIE=1，会产生中断请求。 0: 没检测到 LIN 断开字符 1: 检测到 LIN 断开字符

位 7	TXE: 发送数据寄存器空 当 USART_TDR 寄存器中的值被取到移位寄存器的同时，这个位被硬件置 1。再向 USART_TDR 寄存器写数据就会同时清掉这个位。TXE 标志还可以用其它方式清除，例如向 USART_RQR 寄存器的 TXFRQ 位写 1， 为了丢弃数据（仅于智能卡模式 T=0，传送失败的情形）。如果 USART_CR1 寄存器中的 TXEIE 位被置起时，则会产生中断。 0: 没有数据被传到移位寄存器 1: 有数据被传到移位寄存器，发送数据寄存器为空。
位 6	TC: 发送完成 在 TXE 为 1 的条件下，当数据发送完成的时候，这个位会被硬件置 1。如果 USART_CR1 寄存器中的 TCIE 位是 1，就会产生中断请求。向 USART_TDR 寄存器再次写入数据，或者向 USART_ICR 寄存器的 TCCF 位写 1，都可以清除这个标志。 如果 USART_CR1 寄存器中的 TCIE 位是 1，就会产生中断请求。 0: 发送未完成 1: 发送完成
位 5	RXNE: 接收数据寄存器非空 当接收移位寄存器的内容被传递到 USART_RDR 寄存器中时，这个位被硬件置 1。 读取 USART_TDR 寄存器的数据就会同时清掉这个位。RXNE 标志也可以通过对 USART_RQR 寄存器中的 RXFRQ 位写 1 来清除。如果 USART_CR1 寄存器中的 RXNEIE 位是 1，就会产生中断请求。 0: 没收到数据 1: 收到的数据已经可读
位 4	IDLE: 空闲线检测到 当检测到线路空闲时由硬件置 1。 如果 USART_CR1 寄存器中的 IDLEIE 位是 1，就 会产生中断请求。 由软件向 USART_ICR 寄存器的 IDLECF 位写 1，可以清除这个 标志。 0: 没有检测到线路空闲 1: 检测到线路空闲
位 3	ORE: 溢出错误 在 RXNE=1 的的条件下（也就是上次数据还没有读走），串口接收寄存器又接收好 了一个字节的数据并准备往 RDR 寄存器去转移的时候，会由硬件将这个位置 1。 由软件向 USART_ICR 寄存器的 ORECF 位写 1，可以清除这个标志。如果 USART_CR1 寄存器中的 RXNEIE 位或 EIE 位是 1，就会产生中断请求。 0: 没有溢出错误 1: 检测到溢出错误
位 2	NF: 噪声检测标志 当收帧的时候检测到噪声，这一位会由硬件置 1。 由软件向 USART_ICR 寄存器的 NFCF 位写 1，可以清除这个标志。 0: 没有检测到噪声 1: 检测到噪声
位 1	FE: 帧错误 当一个不同步现象、强噪声或一个断开符号被检测到的时候，这个位有硬件置 1。 由软件向 USART_ICR 寄存器的 FECF 位写 1，可以清除这个标志。 在智能卡模式 中发送数据时，当重发尝试的次数达到上限，由没有收到成功的回应（卡一直响应 NACK）的时候，这个位也会被硬件置 1。如果 USART_CR1 寄存器中的 EIE 位是 1，会产生中断请求。 0: 没有检测到帧错误 1: 有检测到帧错误或者有收到断开字符
位 0	PE: 校验错误标志 当在接收数据的时候发现校验错误，这个位会由硬件置 1。 由软件向 USART_ICR 寄存器的 PECF 位写 1，可以清除这个标志。如果 USART_CR1 寄存器中的 PEIE 位是 1，会产生中断请求。 0: 没有校验错误 1: 有校验错误

25.8.9 中断标志清除寄存器（USARTx_ICR）

地址偏移: 0x20

复位值: 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WUCF	Res.	Res.	CMCF	Res.
											w_r1			w_r1	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
			EOBCF	RTOCF		CTSCF	LBDCF		TCCF		IDLECF	ORECF	NCF	FECF	PECF
			w_r1	w_r1		w_r1	w_r1		w_r1		w_r1	w_r1	w_r1	w_r1	w_r1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位 31:21	保留，必须保持复位时的值。
位 20	WUCF: 从 Stop 模式唤醒标志的清除 对这个位写 1，会清除 USART_ISR 寄存器中的 WUF 标志位。
位 19:18	保留，必须保持复位时的值。 位 17 CMCF: 字符匹配标志的清除 对这个位写 1，会清除 USART_ISR 寄存器中的 CMF 标志位。 位 16:13 保留，必须保持复位时的值。
位 12	EOBCF: 块结束标志的清除 对这个位写 1，会清除 USART_ISR 寄存器中的 EOBF 标志位。
位 11	RTOCF: 接收超时标志的清除 对这个位写 1，会清除 USART_ISR 寄存器中的 RTOF 标志位。
位 10	保留，必须保持复位时的值。 位 9 CTSCF: CTS 标志的清除 对这个位写 1，会清除 USART_ISR 寄存器中的 CTSIF 标志位。
位 8	LBDCF: LIN 断开检测标志的清除 对这个位写 1，会清除 USART_ISR 寄存器中的 LBDF 标志位。
位 7	保留，必须保持复位时的值。
位 6	TCCF: 发送完成标志的清除 对这个位写 1，会清除 USART_ISR 寄存器中的 TC 标志位。 位 5 保留，必须保持复位时的值。
位 4	**IDLECF: 线路空闲检测标志的清除** **对这个位写 1，会清除 USART_ISR 寄存器中的 IDLE 标志位。**
位 3	ORECF: 溢出错误标志的清除 对这个位写 1，会清除 USART_ISR 寄存器中的 ORE 标志位。
位 2	NCF: 噪声检测标志的清除 对这个位写 1，会清除 USART_ISR 寄存器中的 NF 标志位。
位 1	FECF: 帧错误标志的清除 对这个位写 1，会清除 USART_ISR 寄存器中的 FE 标志位。
位 0	PECF: 校验错误标志的清除 对这个位写 1，会清除 USART_ISR 寄存器中的 PE 标志位。

25.8.10 数据接收寄存器（USARTx_RDR）

地址偏移: 0x24

复位值: undefined

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

							r	r	r	r	r	r	r	r	r
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位 31:9	保留，必须保持复位时的值。
位 8:0	RDR[8:0]: 接收数据的值包含所收到的字节。 RDR 寄存器提供输入移位寄存器和内部总线间的并行接口。 当接收数据时打开了校验位，读这个寄存器得到的最高位是校验位。

25.8.11 数据发送寄存器（USARTx_TDR）

地址偏移: 0x28

复位值: undefined

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

位 31:9	保留，必须保持复位时的值。
位 8:0	TDR[8:0]: 发送数据的值用于写入要发送的数据字节。 TDR 寄存器提供发送移位寄存器和内部总线间的并行接口。 当发送的时候设置了校验功能（USART_CR1 中的 PCE=1），向最高位（位 7 还是 位 8 取决于设置的字长）写入的信息是无效的，因为它总是要被校验位代替了之后 再去发送的。

26 通用异步收发器 (UART)

26.1 简介

通用同步异步收发器 (USART) 能够灵活地与外部设备进行全双工数据交换，满足外部设备对工业标准 NRZ 异步串行数据格式的要求。 USART 通过可编程波特率发生器提供了多种波特率。

它支持半双工单线通信以及多处理器通信，以及调制解调器操作 (CTS/RTS)。通过配置多个缓冲区使用 DMA（直接存储器访问）可实现高速数据通信。

26.2 UART 主要特性

- 全双工异步通信
- NRZ 标准格式（标记/空格）
- 双时钟域允许：
  - – UART 功能
  - – 方便的波特率编程，独立于 PCLK 重新编程
- 数据字长度可编程（7 位、 8 位或 9 位）
- 可编程的数据顺序，最先移位 MSB 或 LSB
- 停止位可配置（支持 1 个或 2 个停止位）
- 单线半双工通信
- 使用 DMA 实现连续通信
- 使用中央 DMA 在预留的 SRAM 缓冲区中收/发字节。
- 发射器和接收器有单独的使能位
- 发送和接收的单独信号极性控制
- Tx/Rx 引脚配置可交换
- 调制解调器和 RS-485 收发器的硬件流控制
- 传输检测标志：
  - 接收缓冲区已满
  - 发送缓冲区为空 (Transmit buffer empty)
  - BUSY 标志和发送结束标志
- 奇偶校验控制：
  - 发送奇偶校验位
  - 检查接收的数据字节的奇偶性
- 四个错误检测标志：
  - 上溢错误
  - 噪声检测
  - 帧错误
  - 奇偶校验错误
- 十四个具有标志位的中断源
- 多处理器通信

  如果地址不匹配， LPUART 将进入静默模式。
- 从静默模式唤醒（通过空闲线检测或地址标记检测）

26.3 UART 特性实现

参考 USART 特性实现。

26.4 UART 功能说明

任何 USART 双向通信均需要至少两个引脚：接收数据输入引脚 (RX) 和发送数据输出引脚(TX)：

- **RX**：接收数据输入引脚。

串行数据输入引脚。过采样技术可区分有效输入数据和噪声，从而用于恢复数据。

- **TX**：发送数据输出引脚。

如果关闭发送器，该输出引脚模式由其 I/O 端口配置决定。如果使能了发送器但没有待发送的数据，则 TX 引脚处于高电平。在单线和智能卡模式下，该 I/O 用于发送和接收数据。

正常 USART 模式下，通过这些引脚发送和接收串行数据。这些帧包括：

- 发送或接收前保持空闲线路
- 起始位
- 数据字（字长 7 位、 8 位或 9 位），最低有效位在前
- 用于指示帧传输已完成的 0.5 个、 1 个、 1.5 个、 2 个停止位
- USART 接口使用波特率发生器
- 状态寄存器 (USARTx_ISR)
- 接收和发送数据寄存器（USARTx_RDR、 USARTx_TDR）
- 波特率寄存器 (USARTx_BRR)
- 智能卡模式下的保护时间寄存器 (USARTx_GTPR)。

有关各个位的定义，请参见 *USART* 寄存器。

在同步模式和智能卡模式下连接时需要以下引脚：

- **CK**： 时钟输出。该引脚用于输出发送器数据时钟，以便按照 SPI 主模式进行同步发送（起始位和结束位上无时钟脉冲，可通过软件向最后一个数据位发送时钟脉冲）。 RX 上可同步接收并行数据。这一点可用于控制带移位寄存器的外设（例如 LCD 驱动器）。时钟相位和极性可通过软件编程。在智能卡模式下，CK 输出可向智能卡提供时钟。

在 RS232 硬件流控制模式下需要以下引脚：

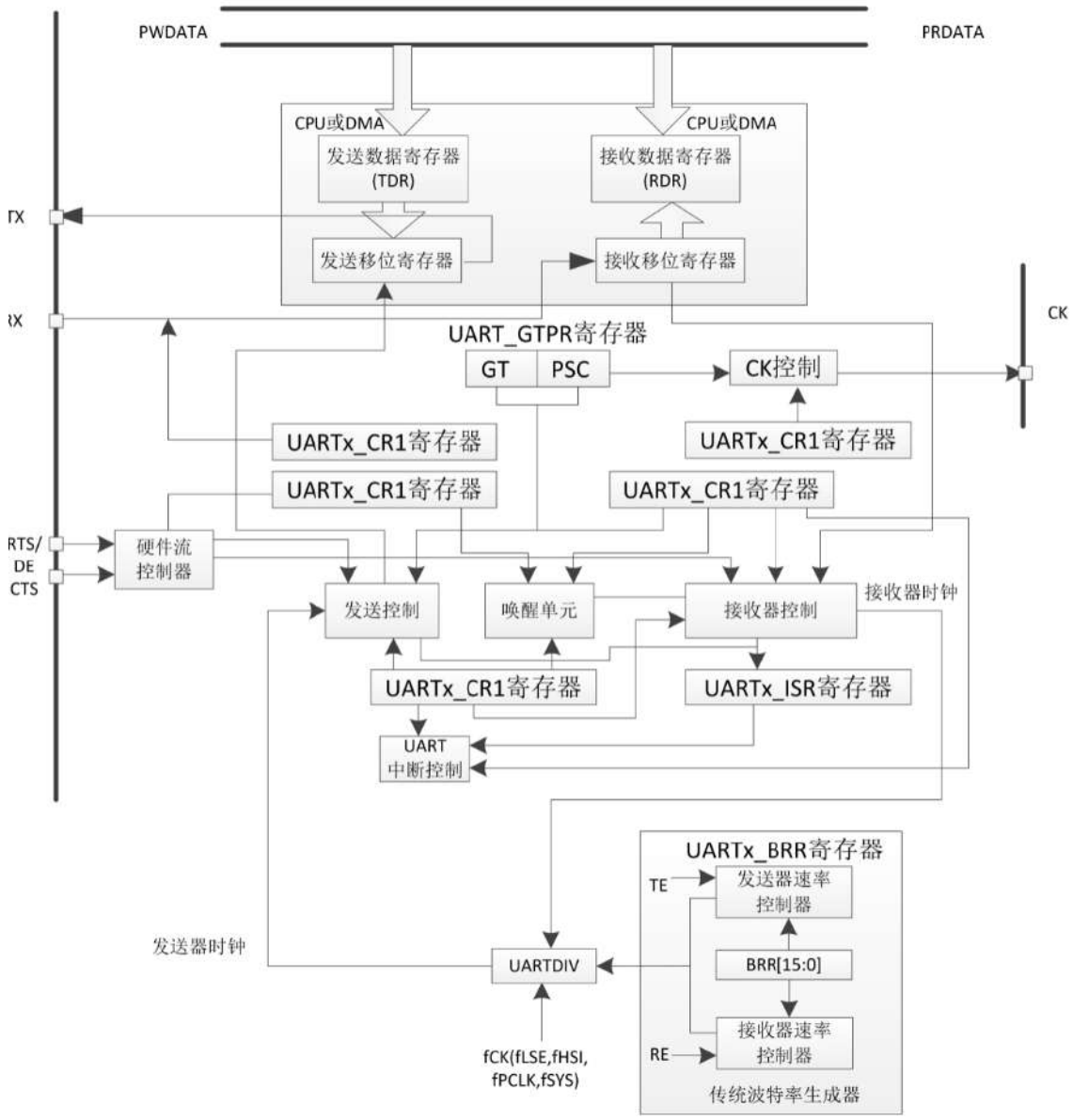
- **CTS**：“清除以发送”用于在当前传输结束时阻止数据发送（高电平时）
- **RTS**：“请求以发送”用于指示 USART 已准备好接收数据（低电平时）

在 RS485 硬件控制模式下需要以下引脚：

- **DE**：“驱动器使能”用于激活外部收发器的发送模式。

注： *DE* 和 *RTS* 共用同一个引脚。

图 26-1 USART 框图



26.4.1 UART 字符说明

可通过对 LPUART_CR1 寄存器中的 M[1:0] 位进行编程来选择 7 位、 8 位或 9 位的字长（请参见图 26-2）。

- 7 位字符长度： M[1:0] = 10
- 8 位字符长度： M[1:0] = 00
- 9 位字符长度： M[1:0] = 01

在默认情况下，信号（TX 或 RX）在起始位工作期间处于低电平状态。在停止位工作期间处于高电平状态。

通过极性配置控制，可以单独针对每个信号对这些值取反。

空闲字符可理解为整个帧周期内电平均为“1”。（停止位的电平也是“1”）。

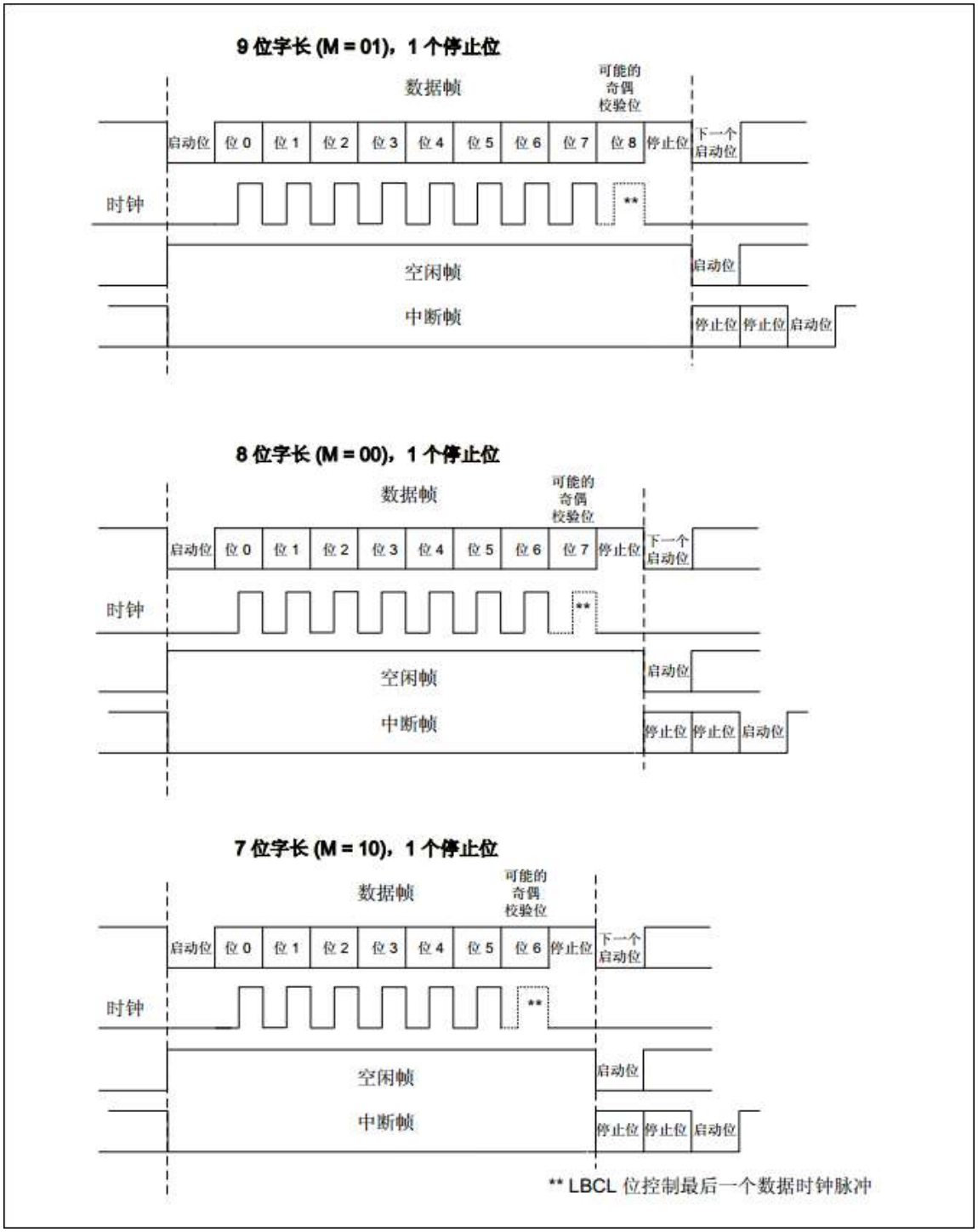
停止字符可理解为在一个帧周期内接收到的电平均为“0”。发送器在中断帧的末尾插入 2 个停止位。

发送和接收由通用波特率发生器驱动，发送器和接收器的使能位分别置 1 时将生成相应的发送时钟和接

收时钟。

下面给出了各个块的详细信息。

图 26-2 字长编程



26.4.2 UART 发送器

发送器可发送 7 位、 8 位或 9 位的数据字，具体取决于 M 位的状态。要激活发送器功能，必须将发送使能位 (TE) 置 1。发送移位寄存器中的数据输出到 TX 引脚。

字符发送

UART 发送期间，首先通过 TX 引脚移出数据的最低有效位（默认配置）。该模式下， UART_TDR

寄存器的缓冲区 (TDR) 位于内部总线和发送移位寄存器之间（请参见图 26-1）。

每个字符前面都有一个起始位，其逻辑电平在一个位周期内为低电平。字符由可配置数量的停止位终止。

UART 支持以下停止位： 1 个和 2 个停止位。

注： 向 *UART_TDR* 中写入要发送的数据前， *TE* 位必须先置 *1*。

数据发送期间不应复位 *TE* 位。发送期间复位 *TE* 位会冻结波特率计数器，从而将损坏 *TX* 引脚上的数据。当前传输的数据将会丢失。

使能 *TE* 位后，将会发送空闲帧。

可配置的停止位

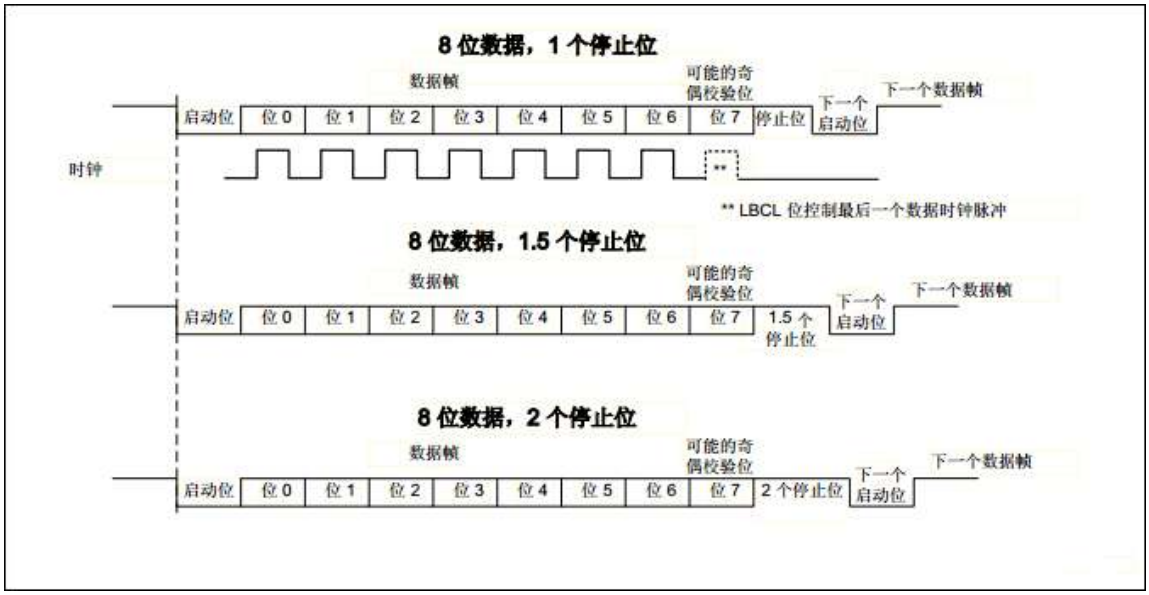
可以在控制寄存器 2 的位 13 和位 12 中编程将随各个字符发送的停止位的数量。

- ***1*** 个停止位： 这是停止位数量的默认值。
- ***2*** 个停止位： 正常 UART 模式、单线模式和调制解调器模式支持该值。

空闲帧发送将包括停止位。

中断发送是 10 个低电平位（M[1:0] = 00 时）、 11 个低电平位（M[1:0] = 01 时）或 9 个低电平位（M[1:0] = 10 时），然后是 2 个停止位。无法传送长中断（中断长度大于 9/10/11 个低电平位）。

图 26-3 可配置的停止位



字符发送步骤

1. 对 UART_CR1 中的 M 位进行编程以定义字长。
2. 使用 UART_BRR 寄存器选择所需波特率。
3. 对 UART_CR2 中的停止位数量进行编程。
4. 通过向 UART_CR1 寄存器中的 UE 位写入 1 使能 UART。
5. 如果将进行多缓冲区通信，请选择 UART_CR3 中的 DMA 使能 (DMAT)。按照多缓冲区通信中的解释说明配置 DMA 寄存器。
6. 将 UART_CR1 中的 TE 位置 1 以便在首次发送时发送一个空闲帧。
7. 在 UART_TDR 寄存器中写入要发送的数据（该操作将清零 TXE 位）。为每个要在单缓冲区模式下发送的数据重复这一步骤。
8. 向 UART_TDR 寄存器写入最后一个数据后，等待至 TC=1。这表明最后一个帧的传送已完成。禁止 UART 或进入暂停模式时需要此步骤，以避免损坏最后一次发送。

单字节通信

始终通过向发送数据寄存器写入数据将 TXE 位清零。

TXE 位由硬件置 1，它表示：

- 数据已从 UART_TDR 寄存器移到移位寄存器中且数据发送已开始。
- UART_TDR 寄存器为空。
- UART_TDR 寄存器中可写入下一个数据，而不会覆盖前一个数据。

TXEIE 位置 1 时该标志位会生成中断。

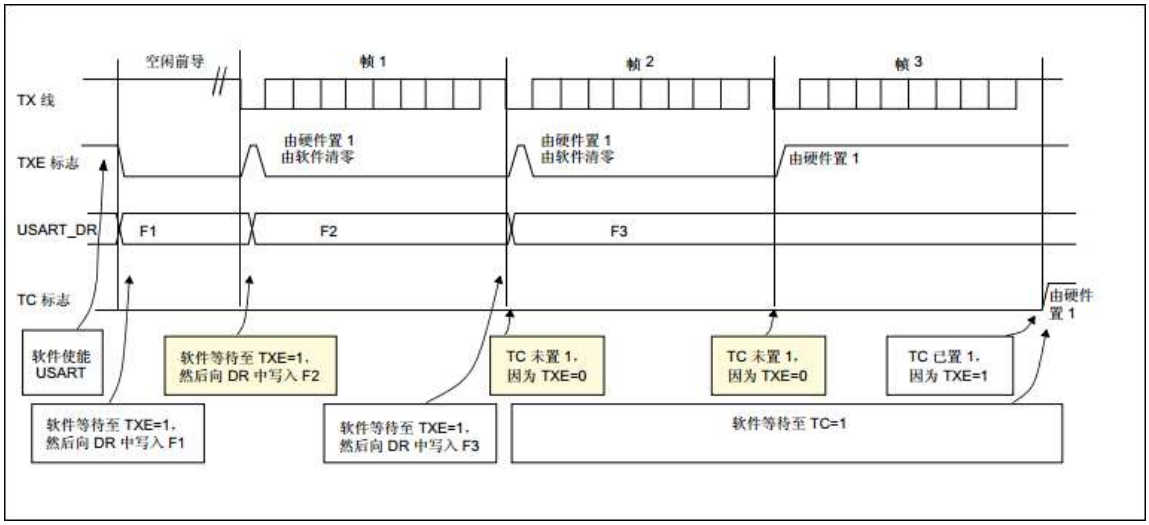
发送时，要传入 UART_TDR 寄存器的写指令中存有 TDR 寄存器中的数据，接下来，该数据将在当前进行的发送结束时复制到移位寄存器中。

未发送时，要传入 UART_TDR 寄存器的写指令将数据置于移位寄存器中，数据发送开始时， TXE 位置 1。

如果帧已发送（停止位后）且 TXE 位置 1， TC 位将变为高电平。如果 UART_CR1 寄存器中的 TCIE 位置 1，将生成中断。

向 UART_TDR 寄存器中写入最后一个数据后，必须等待至 TC=1，之后才可禁止 UART 或使微控制器进入低功率模式。

图 26-4 发送时的 TC/TXE 行为



中断字符

将 SBKRQ 位置 1 将发送一个中断字符。中断帧的长度取决于 M 位（请参见图 *261*）。如果将“1”写入 SBKRQ 位，则当前字符发送完成后，将在 TX 线路上发送一个中断字符。通过写操作将 SBKF 位置 1 并在中断字符发送完成时（发送中断字符后的停止位期间），此位由硬件复位。 LPUART 在中断帧末尾的两位持续时间内插入一个逻辑“1”信号 (STOP)，以确保识别下个帧的起始位。

这种情况下，应用需要在之前插入的所有数据（包括尚未发送的数据）后发送中断字符，软件应等到 TXE 标志使能后将 SBKRQ 位置 1。

空闲字符

将 TE 位置 1 会驱动 UART 在第一个数据帧之前发送一个空闲帧。

26.4.3 UART 接收器

UART 可接收 7 位、 8 位或 9 位的数据字，具体取决于 UART_CR1 寄存器中的 M 位。

起始位检测

在 UART 中，进行 START 位检测时，首先应在 Rx 线路上检测到下降沿，然后在起始位中间采样以确认电平是否仍为“0”。如果起始采样为“1”，则噪声错误标志 (NF) 置 1，

START 位将被丢弃，接收器将等待新的 START 位。否则，接收器将继续正常采样所有传入位。

字符接收

UART 接收期间，首先通过 RX 引脚移入数据的最低有效位（默认配置）。该模式下，UART_RDR 寄存器的缓冲区 (RDR) 位于内部总线和接收移位寄存器之间。

字符接收步骤

1. 对 UART_CR1 中的 M 位进行编程以定义字长。
2. 使用波特率寄存器 UART_BRR 选择所需波特率。
3. 对 UART_CR2 中的停止位数量进行编程。
4. 通过向 UART_CR1 寄存器中的 UE 位写入 1 使能 LPUART。
5. 如果将进行多缓冲区通信，请选择 UART_CR3 中的 DMA 使能 (DMAR)。按照多缓冲区通信中的解释说明配置 DMA 寄存器。
6. 将 RE 位 UART_CR1 置 1。这一操作将使能接收器开始搜索起始位。

接收到字符时

• RXNE 位置 1。这表明移位寄存器的内容已传送到 RDR。也就是说，已接收到并可读取数据（及其相应的错误标志）。

• 如果 RXNEIE 位置 1，则会生成中断。

• 如果接收期间已检测到帧错误、噪声错误或上溢错误，错误标志位可置 1。也可使用 RXNE 位将 PE 标志置 1。

• 在多缓冲区模式下，每接收到一个字节后 RXNE 均置 1，然后通过 DMA 对接收数据寄存器执行读操作清零。

• 在单缓冲区模式下，通过软件对 UART_RDR 寄存器执行读操作将 RXNE 位清零。也可以通过将 UART_RQR 寄存器中的 RXFRQ 位置 1 将 RXNE 标志位清零。

RXNE 位必须在结束接收下一个字符前清零，以避免发生上溢错误。

中断字符

接收到中断字符时， UART 将会按照帧错误对其进行处理。

空闲字符

检测到空闲帧时，处理步骤与接收到数据的情况相同；如果 IDLEIE 位置 1，则会产生中断。

上溢错误

如果在 RXNE 未复位时接收到字符，则会发生上溢错误。 RXNE 位清零前，数据无法从移位寄存器传送到 RDR 寄存器。

每接收到一个字节后， RXNE 标志位都将置 1。当 RXNE 标志位是 1 时，如果在接收到下一个数据或尚未处理上一个 DMA 请求时，则会发生上溢错误。发生上溢错误时：

- ORE 位置 1。
- RDR 中的内容不会丢失。对 UART_RDR 执行读操作时可使用先前的数据。
- 移位寄存器将被覆盖。之后，上溢期间接收到的任何数据都将丢失。
- 如果 RXNEIE 位置 1 或 EIE 位置 1，则会生成中断。
- 通过设置 ICR 寄存器中的 ORECF 位复位 ORE 位。

注： ORE 位置 1 时表示至少 1 个数据丢失。存在两种可能：

– 如果 *RXNE=1*，则最后一个有效数据存储于接收寄存器 *RDR* 中并且可进行读取；

– 如果 *RXNE=0*，则表示最后一个有效数据已被读取，因此 *RDR* 中没有要读取的数据。

接收到新（和丢失）数据的同时已读取 *RDR* 中的最后一个有效数据时，会发生该情况。

选择时钟源

通过复位和时钟控制系统 (RCC) 选择时钟源。时钟源必须在使能 UART（通过将 UE 位置 1）前选择。必须遵循以下两个条件选择时钟源：

- 可在低功耗模式下使用 UART
- 通信速度。

时钟源频率为 fCK。

当支持双时钟域和从停止模式唤醒功能时，时钟源可以是以下其中之一： fPCLK（默认值）、fLSE、 fHSI 或 fSYS。否则， UART 时钟源是 fPCLK。

选择 fLSE 或 fHSI 作为时钟源可允许 UART 在 MCU 处于低功耗模式时接收数据。必要时， UART 可基于所接收的数据和唤醒模式的选择来唤醒 MCU，以通过用软件读取 UART_RDR 寄存器的方式或通过 DMA 的方式传输已接收数据。

对于其它时钟源，系统必须激活才能进行 UART 通信。

时钟源还决定通信速度范围（尤其是最大通信速度）。

接收器在尽可能靠近波特周期中间的位置采样每个传入波特。每个传入波特仅进行一次采样。

注： 不进行数据噪声检测。

帧错误 **(Framing error)**

如果接收数据时未在预期时间内识别出停止位，从而出现同步失效或过度的噪声，则会检测到帧错误。

检测到帧错误时：

- FE 位由硬件置 1。
- 无效数据从移位寄存器传送到 UART_RDR 寄存器。
- 单字节通信时无中断产生。然而，在 RXNE 位产生中断时，该位出现上升沿。多缓冲区通信时， UART_CR3 寄存器中的 EIE 位置 1 时将发出中断。

通过将 1 写入 UART_ICR 寄存器中的 FECF 位来复位 FE 位。

接收期间可配置的停止位

可通过控制寄存器 2 中的控制位配置要接收的停止位的数量 - 可以是 1 个或 2 个（正常模式下）。

- ***1*** 个停止位：将在第 8、第 9 和第 10 次采样时对 1 个停止位进行采样。
- ***2*** 个停止位：将在第 2 个停止位的中间对 2 个停止位进行采样。 RXNE 和 FE 标志在此采样期间（例如，在第 2 个停止位期间）置 1。发生帧错误时不检测第 1 个停止位。

26.4.4 UART 波特率生成器

接收器和发送器（Rx 和 Tx）的波特率设置为 UART_BRR 寄存器中编程的相同值。

16 倍过采样时的公式为：

$$\textbf{Tx/Rx}\ \text{波特} = \frac{\text{fck}}{\text{UARTDIV}}$$

8 倍过采样时的公式为：

$$\textbf{Tx/Rx}\ \text{波特} = \frac{2\text{xfck}}{\text{UARTDIV}}$$

注： 对 UART_BRR 执行写操作后，波特率计数器更新为波特率寄存器中的新值。因此，波特率寄存器的值不应在通信时发生更改。

16 倍或 *8* 倍过采样时， *USARTDIV* 必须大于或等于 *0d16*。

表 26-1 采用 **16** 倍或 **8** 倍过采样时，在 **fCK = 32 MHz** 下编程波特率的误差计算

波特率		**16** 倍过采样 **(OVER8 = 0)**			**8** 倍过采样 **(OVER8 = 1)**		
序号	所需值	实际值	**BRR**	误差 **% =**（计算值 - 所需值）**/** 所需波特率	实际值	**BRR**	误差 **%**
1	2.4 KBps	2.4 KBps	0x3415	0	2.4 KBps	0x6825	0
2	9.6 KBps	9.6 KBps	0xD05	0	9.6 KBps	0x1A05	0
3	19.2 KBps	19.19 KBps	0x683	0.02	19.2 KBps	0xD02	0

4	38.4 KBps	38.41 KBps	0x341	0.04	38.39 KBps	0x681	0.02
5	57.6 KBps	57.55 KBps	0x22C	0.08	57.6 KBps	0x453	0
6	115.2 KBps	115.1 KBps	0x116	0.08	115.11 KBps	0x226	0.08
7	230.4 KBps	230.21 KBps	0x8B	0.08	230.21 KBps	0x113	0.08
8	460.8 KBps	463.76 KBps	0x045	0.64	460.06 KBps	0x85	0.08
9	921.6 KBps	914.28 KBps	0x23	0.79	927.5 KBps	0x42	0.79
10	2 MBps	2 MBps	0x10	0	2 MBps	0x20	0
12	4MBps	4MBps	NA	NA	4MBps	0x10	0

1. CPU 时钟越低，特定波特率的精度就越低。可达到的波特率的上限可使用这些数据进行确定。

26.4.5 UART 接收器对时钟偏差的容差

仅当总时钟系统偏差小于 USART 接收器的容差时， USART 异步接收器才能正常工作。影响总偏差的因素包括：

- DTRA：发送器误差引起的偏差（其中还包括发送器本地振荡器的偏差）
- DQUANT：接收器的波特率量化引起的误差
- DREC：接收器本地振荡器的偏差
- DTCL：传输线路引起的偏差（通常是由于收发器所引起，它可能会在低电平到高电平转换时序与高电平到低电平转换时序之间引入不对称）

DTRA DQUANT DREC DTCL DWU + + + + USART < 接收器的容差

LPUART 接收器在表 *133* 中指定的最大容许偏差下可正确接收数据：

- 由 LPUARTx_CR1 寄存器中的 M 位定义的 7 位、 8 位或 9 位字符长度
- 1 个或 2 个停止位

表 26-2 **BRR[3:0]** 不等于 **0000** 时的 **UART** 接收器容差

M 位	**768 ≤ BRR <1024**	**1024 ≤ BRR < 2048**	**2048 ≤ BRR < 4096**	**4096 ≤ BRR**
8 位 (M=00)，1 个停止位	1.82%	2.56%	3.90%	4.42%
9 位 (M=01)，1 个停止位	1.69%	2.33%	2.53%	4.14%
7 位 (M=10)，1 个停止位	2.08%	2.86%	4.35%	4.42%
8 位 (M=00)，2 个停止位	2.08%	2.86%	4.35%	4.42%
9 位 (M=01)，2 个停止位	1.82%	2.56%	3.90%	4.42%
7 位 (M=10)，2 个停止位	2.34%	3.23%	4.92%	4.42%

26.4.6 使用 UART 进行多处理器通信

可以通过 UART 进行多处理器通信（多个 UART 连接在一个网络中）。例如，其中一个 UART 可以是主器件，其 TX 输出与其它 UART 的 RX 输入相连接。其它 UART 为从 UART，其各自的 TX 输出在逻辑上通过与运算连在一起，并与主 UART 的 RX 输入相连接。

在多处理器配置中，理想情况下通常只有预期的消息接收方主动接收完整的消息内容，从而减少由所有未被寻址的接收器造成的冗余 UART 服务开销。

可通过静默功能将未被寻址的器件置于静默模式下。为了使用静默模式功能，必须将 UART_CR1 寄存器中的 MME 位置 1。

在静默模式下：

- 不得将接收状态位置 1。
- 禁止任何接收中断。
- UART_ISR 寄存器中的 RWU 位置 1。在某些情况下， RWU 可以由硬件或软件通过 UART_RQR 寄存器中的 MMRQ 位自动控制。

根据 UART_CR1 寄存器中 WAKE 位的设置， UART 可使用以下两种方法进入或退出静音模式：

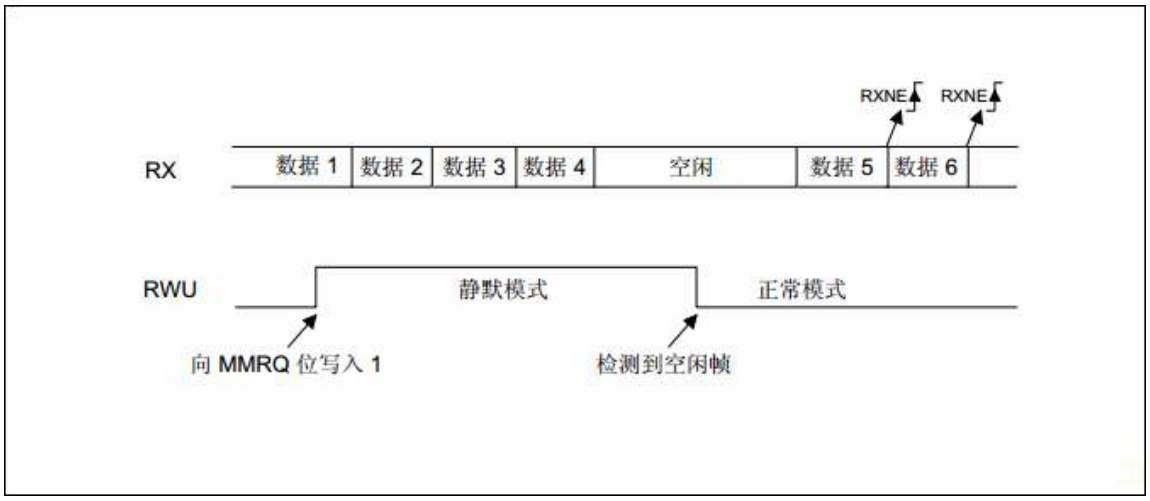
- 如果 WAKE 位被复位，则进行空闲线路检测，
- 如果 WAKE 位置 1，则进行地址标记检测。

空闲线路检测 (WAKE=0)

当向 MMRQ 位写入 1 且 RWU 位自动置 1 时， UART 进入静默模式。

当检测到空闲帧时，它会被唤醒。此时 RWU 位会由硬件清零，但 UART_ISR 寄存器中的 IDLE 位不会置 1。

图 26-5 使用空闲线路检测时的静默模式



注： 如果在 IDLE 字符已经过去时将 MMRQ 位置 1，将不会进入静默模式（RWU 未置 1）。

如果在线路处于空闲状态时激活 *UART*，在一个 *IDLE* 帧持续时间后（不只在接收一个字符帧后）会检测到空闲状态。

4 位*I*7 位地址标记检测 (WAKE=1)

在此模式下，如果字节的 MSB 为 1，则将这些字节识别为地址，否则将其识别为数据。在地址字节中，目标接收器的地址位于 4 个或 7 个 LSB 中。 7 位或 4 位地址检测通过 ADDM7 位来选择。接收器会将此 4 位/7 位字与其地址进行比较，该接收器的地址在 UART_CR2 寄存器的 ADD 位中进行设置。

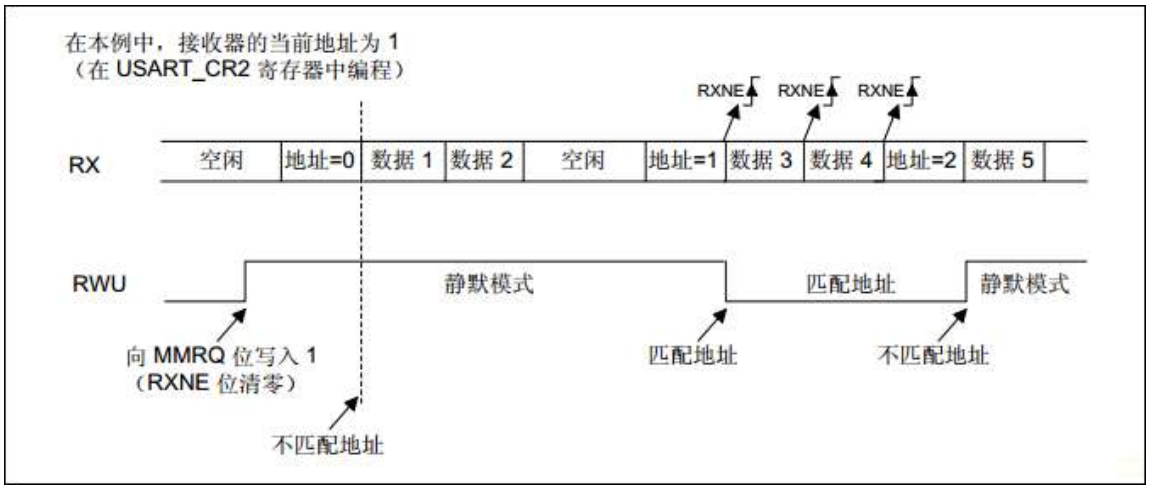
注：在 *7* 位和 *9* 位数据模式下，地址检测分别在 *6* 位和 *8* 位地址上完成（*ADD[5:0]* 和 *ADD[7:0]*）。

当接收到与其编程地址不匹配的地址字符时， UART 会进入静默模式。此时， RWU 位将由硬件置 1。 UART 进入静默模式后， RXNE 标志不会针对此地址字节置 1，也不会发出中断或 DMA 请求。

当向 MMRQ 位写入 1 时， UART 也会进入静默模式。这种情况下， RWU 位也自动置 1。

当接收到与编程地址匹配的地址字符时， UART 会退出静默模式。然后 RWU 位被清零，可以开始正常接收后续字节。由于 RWU 位已清零， RXNE 位会针对地址字符置 1。

图 26-6 使用地址标记检测时的静默模式



26.4.7 UART 校验控制

将 UART_CR1 寄存器中的 PCE 位置 1，可以使能奇偶校验控制（发送时生成奇偶校验位，接收时进行奇偶校验检查）。根据 M 位定义的帧长度， 下表中列出了可能的 UART 帧格式。

表 26-3 帧格式

M 位	PCE 位	LPUART 帧(1)
00	0	| SB | 8 位数据 | STB |
00	1	| SB | 7 位数据 | PB | STB |
01	0	| SB | 9 位数据 | STB |
01	1	| SB | 8 位数据 | PB | STB |
10	0	| SB | 7 位数据 | STB |
10	1	| SB | 6 位数据 | PB | STB |

1. 图注： SB：起始位， STB：停止位， P：奇偶校验位。
2. 在数据寄存器中， PB 始终位于 MSB 位置（第 9 位、第 8 位或第 7 位，具体取决于 M 位的值）。

偶校验

对奇偶校验位进行计算，使帧和奇偶校验位中“1”的数量为偶数（帧由 6 个、 7 个或 8 个 LSB 位组成，具体取决于 M 位的值）。

例如，如果数据=00110101 且 4 个位置 1，则在选择偶校验（LPUART_CR1 寄存器中的 PS 位 = 0）时，校验位为 0。

奇校验

对奇偶校验位进行计算，使帧和奇偶校验位中“1”的数量为奇数（帧由 6 个、 7 个或 8 个 LSB 位组成，具体取决于 M 位的值）。

例如，如果数据=00110101 且 4 个位置 1，则在选择奇校验（LPUART_CR1 寄存器中的 PS 位 = 1）时，校验位为 1。

接收时进行奇偶校验检查

如果奇偶校验检查失败，则 UART_ISR 寄存器中的 PE 标志置 1；如果 LPUART_CR1 寄存器中 PEIE 位置 1，则会生成中断。通过软件将 1 写入 LPUART_ICR 寄存器中的 PECF 位来清零 PE 标志。

发送时的奇偶校验生成

如果 UARTx_CR1 寄存器中的 PCE 位置 1，则在数据寄存器中所写入数据的 MSB 位会进行传送，但

是会由奇偶校验位进行更改（如果选择偶校验 (PS=0)，则“1”的数量为偶数；如果选择奇校验 (PS=1)，则“1”的数量为奇数）。

26.4.8 使用 UART 单线半双工通信

通过将 LPUART_CR3 寄存器中的 HDSEL 位置 1 来选择单线半双工模式。在此模式下，必须将以下位清零：

- LPUART_CR2 寄存器中的 LINEN 和 CLKEN 位，
- LPUART_CR3 寄存器中的 SCEN 和 IREN 位。

LPUART 可以配置为遵循单线半双工协议，其中 TX 和 RX 线路从内部相连接。使用 LPUART_CR3 中的控制位 HDSEL 可在半双工通信和全双工通信间进行选择。

一旦向 HDSEL 位写入 1：

- TX 和 RX 线路从内部相连接
- 不能再使用 RX 引脚
- 无数据传输时， TX 引脚始终处于释放状态。因此，它在空闲状态或接收过程中用作标准 I/O。这意味着，必须对 I/O 进行配置，以便将 TX 配置为复用功能开漏并外接上拉电阻。

除此之外，通信协议与正常 LPUART 模式下的通信协议相似。此线路上的任何冲突必须由软件管理（例如，使用中央仲裁器）。尤其要注意，发送过程永远不会被硬件封锁，只要数据是在 TE 位置 1 的情况下写入，发送就会持续进行。

注： 在 *LPUART* 中，当进行 *1* 个停止位配置时， *RXNE* 标志在停止位中间置 *1*。

26.4.9 使用 UART 在 DMA 模式下进行连续通信

LPUART 能够使用 DMA 进行连续通信。接收缓冲区和发送缓冲区的 DMA 请求是独立生成的。

注： 按照第 *30.4.3* 节中的说明使用 *LPUART*。可以将 *UART_ISR* 寄存器中的 *TXE/ RXNE* 标志清零，从而实现连续通信。

使用 **DMA** 进行发送

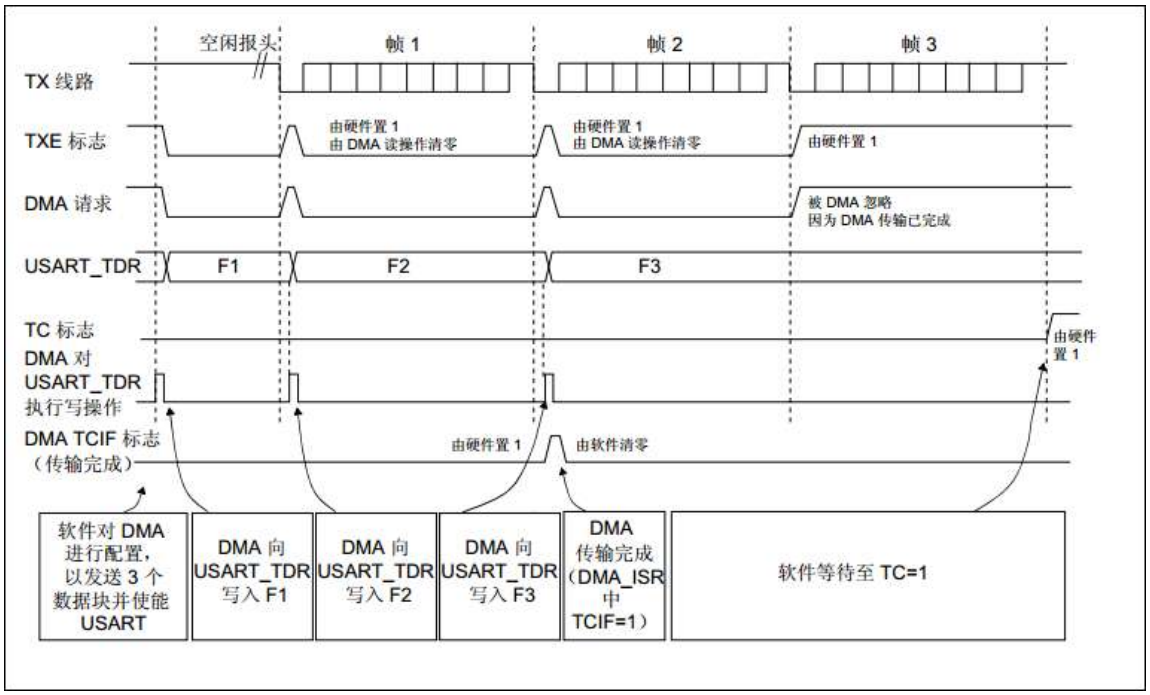
将 LPUART_CR3 寄存器中的 DMAT 位置 1 可以使能 DMA 模式进行发送。当 TXE 位置 1 时，可将数据从 SRAM 区（通过 DMA 外设，请参见直接存储器访问控制器 *(DMA)*）加载到 LPUART_TDR 寄存器。要映射一个 DMA 通道以进行 LPUART 发送，请按以下步骤操作（x 表示通道编号）：

1. 在 DMA 控制寄存器中写入 UART_TDR 寄存器地址，将其配置为传输的目标地址。每次发生 TXE 事件后，数据都从存储器移动到此地址。
2. 在 DMA 控制寄存器中写入存储器地址，将其配置为传输的源地址。每次发生 TXE 事件后，数据都从这个存储区域加载到 LPUART_TDR 寄存器中。
3. 在 DMA 控制寄存器中配置要传输的总字节数。
4. 在 DMA 寄存器中配置通道优先级
5. 根据应用的需求，在完成一半或全部传输后产生 DMA 中断。
6. 通过将 LPUART_ICR 寄存器中的 TCCF 位置 1，将 UART_ISR 寄存器中的 TC 标志清零。
7. 在 DMA 寄存器中激活该通道。

当达到在 DMA 控制器中设置的数据传输量时， DMA 控制器会在 DMA 通道的中断向量上产生一个中断。

在发送模式下， DMA 对所有要发送的数据执行了写操作（DMA_ISR 寄存器中的 TCIF 标志置 1）后，可以对 TC 标志进行监视，以确保 UART 通信已完成。在禁止 UART 或进入停止模式前必须执行此步骤，以避免损坏最后一次发送。软件必须等待直到 TC=1。 TC 标志在所有数据发送期间都保持清零状态，然后在最后一帧发送结束时由硬件置 1。

图 26-7 使用 **DMA** 进行发送



使用 **DMA** 进行接收

将 LPUART_CR3 寄存器中的 DMAR 位置 1 可以使能 DMA 模式进行接收。接收数据字节时，数据会从 LPUART_RDR 寄存器加载到通过 DMA 外设配置的 SRAM 区域中（请参见直接存储器访问控制器 *(DMA)*）。

要映射一个 DMA 通道以进行 LPUART 接收，请按以下步骤操作：1. 在 DMA 控制寄存器中写入 LPUART_RDR 寄存器地址，将其配置为传输的源地址。每次发生 RXNE 事件后，数据都从此地址移动到存储器。

2. 在 DMA 控制寄存器中写入存储器地址，将其配置为传输的目标地址。每次发生 RXNE 事件后，数据都从 LPUART_RDR 寄存器加载到此存储区。

3. 在 DMA 控制寄存器中配置要传输的总字节数。

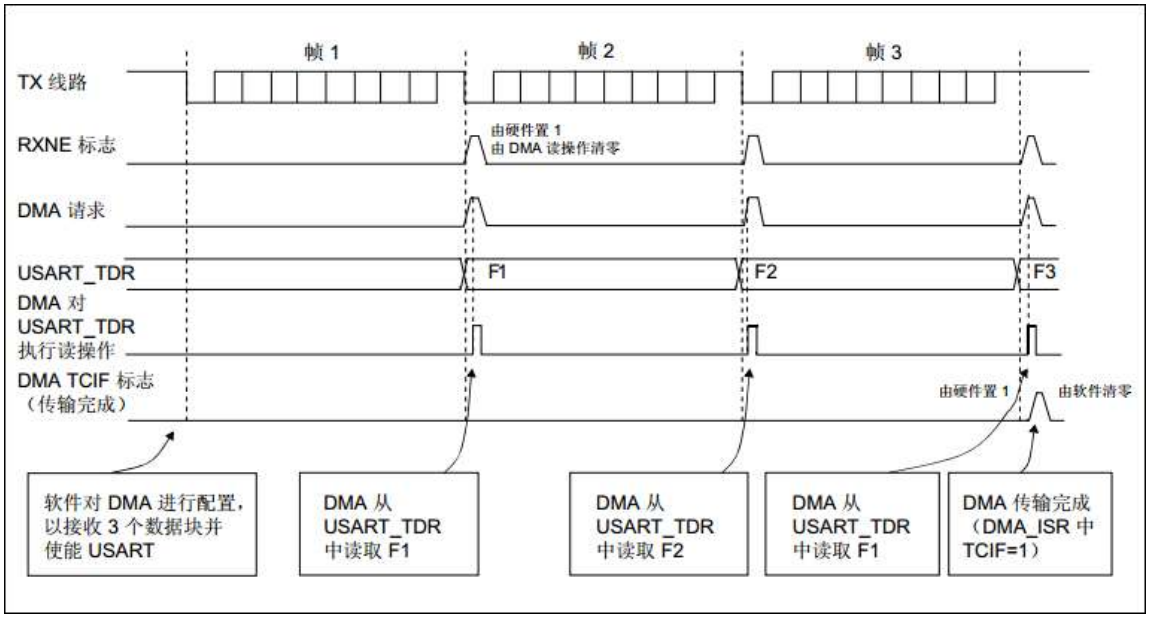
4. 在 DMA 控制寄存器中配置通道优先级。

5. 根据应用的需求，在完成一半或全部传输后产生中断。

6. 在 DMA 控制寄存器中激活该通道。

当达到在 DMA 控制器中设置的数据传输量时，DMA 控制器会在 DMA 通道的中断向量上产生一个中断。

图 26-8 使用 **DMA** 进行接收



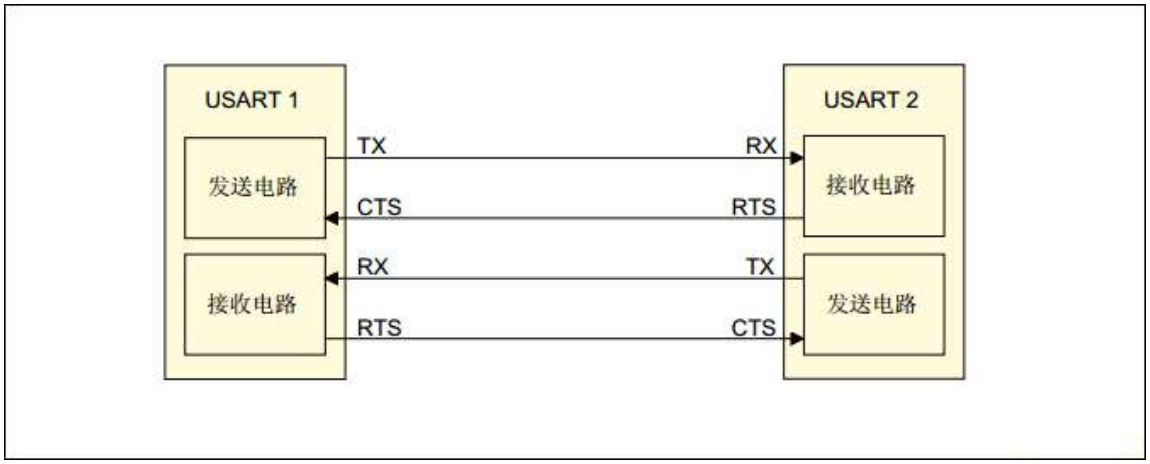
多缓冲区通信中的错误标志和中断生成

在多缓冲区通信中，如果事务中发生任何错误，都会在当前字节后放置错误标志。如果中断使能标志置 1，则会产生中断。在单字节接收过程中，与 RXNE 一同置位的帧错误、上溢错误和噪声标志具有单独的错误标志中断使能位（UART_CR3 寄存器中的 EIE 位）；如果该位置 1，在发生其中任何一个错误时，在当前字节后使能中断。

26.4.10 RS232 硬件流控制和 RS485 驱动器使能（使用 UART）

使用 CTS 输入和 RTS 输出可以控制 2 个器件间的串行数据流。图 26-9 显示了在这种模式下如何连接 2 个器件：

图 26-9 **2** 个 **USART** 间的硬件流控制

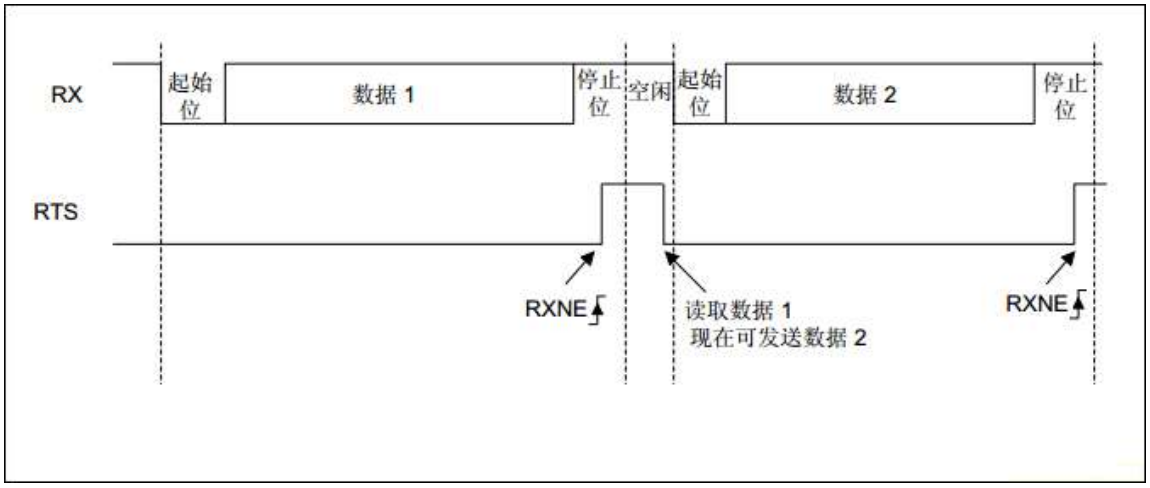


分别向 LPUART_CR3 寄存器中的 RTSE 位和 CTSE 位写入 1，可以分别使能 RS232 RTS 和 CTS 流控制。

RS232 RTS 流控制

如果使能 RTS 流控制 (RTSE=1)，只要 LPUART 接收器准备好接收新数据，便会将 RTS 变为有效（连接到低电平）。当接收寄存器已满时，会将 RTS 变为无效，表明发送过程会在当前帧结束后停止。图 26-10 显示了在使能 RTS 流控制的情况下进行通信的示例。

图 26-10 **RS232 RTS** 流控制



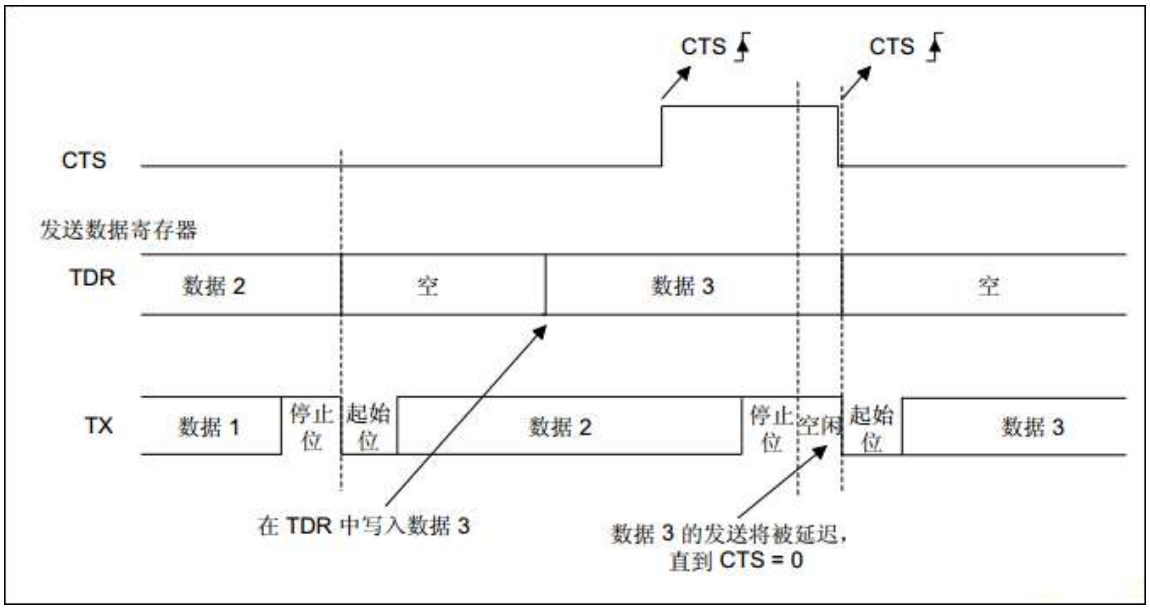
RS232 CTS 流控制

如果使能 CTS 流控制 (CTSE=1)，则发送器会在发送下一帧前检查 CTS。如果 CTS 有效（连接到低电平），则会发送下一数据（假设数据已准备好发送，即 TXE=0）；否则不会进行发送。如果在发送过程中 CTS 变为无效，则当前发送完成之后，发送器停止。

当 CTSE=1 时，只要 CTS 发生变化， CTSIF 状态位便会由硬件自动置 1。这指示接收器是否已准备好进行通信。如果 LPUART_CR3 寄存器中的 CTSIE 位置 1，则会产生中断。

图 26-11 显示了在使能 CTS 流控制的情况下进行通信的示例。

图 26-11 **RS232 CTS** 流控制



注：为正常运行，必须在当前字符结束前至少 *3* 个 *LPUART* 时钟源周期内使能 *CTS*。此外还应注意，当脉冲短于 *2* 个 *PCLK* 周期时无法将 *CTSCF* 标志置 *1*。

RS485 驱动器使能

驱动器使能功能可通过将 LPUART_CR3 控制寄存器中的 DEM 位置 1 使能。这样用户便可通过 DE（驱动器使能）信号激活外部收发器控制。使能时间为激活 DE 信号与 START 位开始间的时间。可以通过 LPUART_CR1 控制寄存器中的 DEAT [4:0] 位域编程使能时间。

禁止时间为发送的消息中最后一个停止位结束与取消激活 DE 信号间的时间。可以通过 LPUART_CR1

控制寄存器中的 DEDT [4:0] 位域编程使能时间。 DE 信号的极性可使用 LPUART_CR3 控制寄存器中的 DEP 位配置。

在 LPUART 中， DEAT 和 DEDT 用 USART 时钟源 (fCK) 周期表示：

- 驱动器使能有效时间 =
  – (1 + (DEAT x P)) x fCK（如果 P <> 0）
  – (1 + DEAT) x fCK（如果 P = 0）
- 驱动器使能禁止时间 =
  – (1 + (DEDT x P)) x fCK（如果 P <> 0）
  – (1 + DEDT) x fCK（如果 P = 0）

P = BRR[14:11]

26.5 UART 中断

表 26-4 **UART** 中断请求

中断事件	事件标志	使能控制位
发送数据寄存器为空	TXE	TXEIE
CTS 中断	CTSIF	CTSIE
发送完成	TC	TCIE
接收数据寄存器不为空（已准备好读取数据）	RXNE	RXNEIE
检测到溢出错误	ORE	
检测到空闲线路	IDLE	IDLEIE
奇偶校验错误	PE	PEIE
多缓冲区通信中的噪声标志、溢出错误和帧错误	NF 或 ORE 或 FE	EIE
字符匹配	CMF	CMIE

26.6 UART3/4 寄存器

26.6.1 控制寄存器 1（UARTx_CR1）

偏移地址：0x00
复位值 ：0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	M1	Res.	Res.	DEAT[4:0]					DEDT[4:0]				
			rw			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
			0			0	0	0	0	0	0	0	0	0	0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OVER8	CMIE	MME	M0	WAKE	PCE	PS	PEIE	TXEIE	TCIE	RXNEIE	IDLEIE	TE	RE		UE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw
0	0	0	0	0	0	0	0	0	0	0	0	0	0		0

位 31:29	保留，必须保持复位值
位 28	**M1** 字长 (Word length) 此位和位 12 (M0) 用于确定字长度。此位由软件置 1 或清零。 M[1:0] = 00： 1 个起始位， 8 个数据位， n 个停止位 M[1:0] = 01： 1 个起始位， 9 个数据位， n 个停止位 M[1:0] = 10： 1 个起始位， 7 个数据位， n 个停止位 只有在禁止 UART (UE=0) 时才能写入此位。

位 27:26	保留，必须保持复位值
位 25:21	**DEAT[4:0]**：驱动器使能使能时间 (Driver Enable assertion time) 该 5 位值用于定义激活 DE（启动器使能）信号与起始位开始间的时间。以 UCLK（UART 时钟）时钟周期数表示。有关更多详细信息，请参见 RS485 驱动器使能段落。 只有在禁止 UART (UE=0) 后才能写入此位域。
位 20:16	**DEDT[4:0]**：驱动器使能禁止时间 (Driver Enable de-assertion time) 该 5 位值用于定义发送的消息中最后一个停止位结束与取消激活 DE（驱动器使能）信号间的时间。以 UCLK（USART 时钟）时钟周期数表示。有关更多详细信息，请参见 RS485 驱动器使能段落。 如果在 DEDT 时间内对 UART_TDR 寄存器执行写操作，则新数据仅在经过 DEDT 和 DEAT 时间后才会发送。 只有在禁止 UART (UE=0) 后才能写入此位域。
位 15	保留，必须保持复位值
位 14	**CMIE**：字符匹配中断使能 (Character match interrupt enable) 此位由软件置 1 和清零。 0：禁止中断 1：如果 UART_ISR 寄存器中的 CMF 位置 1，则生成 UART 中断。
位 13	**MME**：静默模式使能 (Mute mode enable) 此位用于激活 LPUART 的静默模式功能。此位置 1 时， LPUART 可按 WAKE 位定义的方式在活动模式与静默模式之间切换。此位由软件置 1 和清零。 0：接收器永久处于活动模式 1：接收器可在静默模式和活动模式之间切换。
位 12	**M0**：字长 (Word length) 此位和位 28 (M1) 用于确定字长度。此位由软件置 1 或清零。请参见位 28 (M1) 的说明。 只有在禁止 UART (UE=0) 时才能写入此位。
位 11	**WAKE**：接收器唤醒方法 (Receiver wakeup method) 此位用于确定 UART 静默模式的唤醒方法。此位由软件置 1 或清零。 0：空闲线路 1：地址标记 只有在禁止 UART (UE=0) 后才能写入此位域。
位 10	**PCE**：奇偶校验控制使能 (Parity control enable) 该位选择硬件奇偶校验控制（生成和检测）。使能奇偶校验控制时，计算出的奇偶校验位被插入到 MSB 位置（如果 M=1，则为第 9 位；如果 M=0，则为第 8 位），并对接收到的数据检查奇偶校验位。此位由软件置 1 和清零。一旦该位置 1， PCE 在当前字节的后面处于活动状态（在接收和发送时）。 0：禁止奇偶校验控制 1：使能奇偶校验控制 只有在禁止 UART (UE=0) 后才能写入此位域。
位 9	**PS**：奇偶校验选择 (Parity selection) 该位用于在使能奇偶校验生成/检测（PCE 位置 1）时选择奇校验或偶校验。此位由软件置 1 和清零。将在当前字节的后面选择奇偶校验。 0：偶校验 1：奇校验 只有在禁止 UART (UE=0) 后才能写入此位域。
位 8	**PEIE**： PE 中断使能 (PE interrupt enable) 此位由软件置 1 和清零。 0：禁止中断 1：当 UART_ISR 寄存器中的 PE=1 时，生成 UART 中断
位 7	**TXEIE**：中断使能 (interrupt enable) 此位由软件置 1 和清零。 0：禁止中断 1：当 UART_ISR 寄存器中的 TXE=1 时，生成 UART 中断
位 6	**TCIE**：传输完成中断使能 (Transfer complete interrupt enable) 此位由软件置 1 和清零。 0：禁止中断 1：当 UART_ISR 寄存器中的 TC=1 时，生成 UART 中断

位 5	**RXNEIE**： RXNE 中断使能 (RXNE interrupt enable) 此位由软件置 1 和清零。 0：禁止中断 1：当 UART_ISR 寄存器中的 ORE=1 或 RXNE=1 时，生成 UART 中断
位 4	**IDLEIE**： IDLE 中断使能 (IDLE interrupt enable) 此位由软件置 1 和清零。 0：禁止中断 1：当 UART_ISR 寄存器中的 IDLE=1 时，生成 UART 中断
位 3	**TE**：发送器使能 (Transmitter enable) 该位使能发送器。此位由软件置 1 和清零。 0：禁止发送器 1：使能发送器 注： 传送期间 *TE* 位上的“*0*”脉冲（“*0*”后紧跟的是“*1*”）会在当前字的后面发送一个报头（空闲线路）。为生成空闲字符， *TE* 不能立即写入 *1*。为确保所需的持续时间，软件可轮询 *UART_ISR* 寄存器中的 *TEACK* 位。 当 *TE* 置 *1* 时，在发送开始前存在 *1* 位的时间延迟。
位 2	**RE**：接收器使能 (Receiver enable) 该位使能接收器。此位由软件置 1 和清零。 0：禁止接收器 1：使能接收器并开始搜索起始位
位 0	**UE**： LPUART 使能 (CTS enable) 此位清零后， LPUART 预分频器和输出将立即停止，并丢弃当前操作。 LPUART 的配置保留，但 LPUART_ISR 中的所有状态标志将被复位。此位由软件置 1 和清零。 0：禁止 LPUART 预分频器和输出，低功耗模式 1：使能 LPUART 注： 为进入低功耗模式而不在线路上生成错误，之前必须复位 *TE* 位，并且软件必须等待 *LPUART_ISR* 寄存器中的 *TC* 位置 *1* 后才能复位 *UE* 位。 *UE = 0* 时也会复位 *DMA* 请求，因必须在复位 *UE* 位前禁止 *DMA* 通道。

26.6.2 控制寄存器 2（UARTx_CR2）

偏移地址：0x04

复位值 ：0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADD[7:4]				ADD[3:0]				Res.	Res.	Res.	Res.	MSBFIRST	DATAINV	TXINV	RXINV
rw				rw								rw	rw	rw	rw
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWAP	Res.	STOP[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADDM7	Res.	Res.	Res.	Res.
rw		rw	rw								rw				
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位 31:28	**ADD[7:4]**：LPUART 节点的地址(Address of the UART node) 此位域用于指定 LPUART 节点的地址或要识别的字符代码。 此位域在多处理器通信时于静默模式或停止模式下使用，以通过 7 位地址标记检测进行唤醒。发送器发送字符的 MSB 应为 1。此位域还可用于正常接收和静默模式无效时的字符检测（例如，Modbus 协议中的块结束检测）。这种情况下，接收到的整个字符（8 位）将与 ADD[7:0] 值进行比较，如果匹配， CMF 标志将置 1。 仅在禁止接收 (RE = 0) 或禁止 UART (UE=0) 时才能写入该位域。
位 23:20	保留，必须保持复位值

位 19	**MSBFIRST**：最高有效位在前 (Most significant bit first) 此位由软件置 1 和清零。 0：发送/接收数据时位 0 在前，后跟起始位。 1：发送/接收数据时 MSB（位 7/8/9）在前，后跟起始位。 只有在禁止 UART (UE=0) 后才能写入此位域。
位 18	**DATAINV**： 二进制数据反向 (Binary data inversion) 此位由软件置 1 和清零。 0：按正/正向逻辑发送/接收数据寄存器中的逻辑数据。（1=H， 0=L） 1：按负/反向逻辑发送/接收数据寄存器中的逻辑数据。（1=L， 0=H）。奇偶校验位也取反。 只有在禁止 UART (UE=0) 后才能写入此位域。
位 17	**TXINV**： TX 引脚有效电平反向 (TX pin active level inversion) 此位由软件置 1 和清零。 0： TX 引脚信号使用标准逻辑电平（V_{DD} = 1/空闲， Gnd = 0/标记）工作。 1：对 TX 引脚信号值取反。（V_{DD} = 0/标记， Gnd=1/空闲）。 允许在 TX 线路上使用外部反相器。 只有在禁止 UART (UE=0) 后才能写入此位域。
位 16	**RXINV**： RX 引脚有效电平反向 (RX pin active level inversion) 此位由软件置 1 和清零。 0： RX 引脚信号使用标准逻辑电平（V_{DD} = 1/空闲， Gnd = 0/标记）工作。 1：对 RX 引脚信号值取反。（V_{DD} = 0/标记， Gnd=1/空闲）。 允许在 RX 线路上使用外部反相器。 只有在禁止 UART (UE=0) 后才能写入此位域。
位 15	**SWAP**： 交换 TX/RX 引脚 (Swap TX/RX pins) 此位由软件置 1 和清零。 0：按标准引脚排列定义使用 TX/RX 引脚 1：交换 TX 和 RX 引脚功能。允许在与另一个 UART 的交叉连接时工作。 只有在禁止 UART (UE=0) 后才能写入此位域。
位 14	保留，必须保持复位值
位 13:12	**STOP[1:0]**：停止位 (STOP bits) 这些位用于编程停止位。 00： 1 个停止位 01：保留 10： 2 个停止位 11：保留 只有在禁止 UART (UE=0) 后才能写入此位域。
位 11:5	保留，必须保持复位值
位 4	**ADDM7**： 7 位地址检测/4 位地址检测 (7-bit Address Detection/4-bit Address Detection) 此位用于选择 4 位地址检测或 7 位地址检测。 0： 4 位地址检测 1： 7 位地址检测（在 8 位数据模式下） 只有在禁止 UART (UE=0) 时才能写入该位 注： 在 *7* 位和 *9* 位数据模式下，地址检测分别在 *6* 位和 *8* 位地址上完成（*ADD[5:0]* 和 *ADD[7:0]*）。
位 3:0	保留，必须保持复位值。

26.6.3 控制寄存器 3（UARTx_CR3）

偏移地址：0x08

复位值 ：0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DEP	DEM	DDRE	OVERDIS	ONEBIT	CTSIE	CTSE	RTSE	DMAT	DMAR	Res.	Res.	HDSEL	Res.	Res.	EIE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw			rw			rw
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位 31:16	保留，必须保持复位值。
位 15	**DEP**：驱动器使能极性选择 (Driver enable polarity selection) 0： DE 信号高电平有效。 1： DE 信号低电平有效。 只有在禁止 UART (UE=0) 时才能写入此位。
位 14	**DEM**：驱动器使能模式 (Driver enable mode) 此位用于通过 DE 信号激活外部收发器控制。 0：禁止 DE 功能。 1：使能 DE 功能。 DE 信号在 RTS 引脚上输出。 只有在禁止 UART (UE=0) 时才能写入此位。
位 13	**DDRE**：接收出错时的 DMA 禁止 (DMA Disable on Reception Error) 0：接收出错时不禁止 DMA。相应的错误标志置 1，但 RXNE 保持为 0 以防止上溢。因此，将不使能 DMA 请求，从而不会传送错误数据（无 DMA 请求），但会传送接收到的下一个正确数据。 1：接收出错后禁止 DMA。相应的错误标志以及 RXNE 均置 1。屏蔽 DMA 请求，直到错误标志清零。这意味着软件必须首先禁止 DMA 请求 (DMAR = 0) 或者将 RXNE 清零，然后再将错误标志清零。 只有在禁止 UART (UE=0) 时才能写入此位。 注： 接收错误包括：奇偶校验错误、帧错误或噪声错误。
位 12	**OVRDIS**：上溢禁止 (Overrun Disable) 此位用于禁止接收上溢检测。 0：接收新数据前未读取已接收的数据时，上溢错误标志 ORE 置 1。 1：禁止上溢功能。如果在 RXNE 标志仍置 1 时接收到新数据，则 ORE 标志不会置 1，且新接收的数据会覆盖 LPUART_RDR 寄存器之前的内容。 只有在禁止 UART (UE=0) 时才能写入此位。 注： 此控制位用于检查通信流而不会读取数据。
位 11	**ONEBIT**：一个采样位方法使能 (One sample bit method enable) 该位允许用户选择采样方法。选择一个采样位方法后，将禁止噪声检测标志 (NF)。 0：三个采样位方法 1：一个采样位方法 只有在禁止 USART (UE=0) 时才能写入此位。 注： *ONEBIT* 功能仅适用于数据位，不适用于起始位。
位 10	**CTSIE**： CTS 中断使能 (CTS interrupt enable) 0：禁止中断 1：当 USARTx_ISR 寄存器中的 CTSIF=1 时，生成中断 注： 如果不支持硬件流控制功能，该位保留并由硬件强制清零。
位 9	**CTSE**： CTS 使能 (CTS enable) 0：禁止 CTS 硬件流控制 1：使能 CTS 模式，仅当 CTS 输入有效（连接到 0）时才发送数据。如果在发送数据时使 CTS 输入无效，会在停止之前完成发送。如果使 CTS 无效时数据已写入数据寄存器，则将延迟发送，直到 CTS 有效。 只有在禁止 UART (UE=0) 时才能写入该位
位 8	**RTSE**： RTS 使能 (RTS enable) 0：禁止 RTS 硬件流控制 1：使能 RTS 输出，仅当接收缓冲区中有空间时才会请求数据。发送完当前字符后应停止发送数据。可以接收数据时使 RTS 输出有效（拉至 0）。 只有在禁止 UART (UE=0) 时才能写入此位。

位 7	**DMAT**： DMA 使能发送器 (DMA enable transmitter) 此位由软件置 1/复位。 1：针对发送使能 DMA 模式 0：针对发送禁止 DMA 模式
位 6	**DMAR**： DMA 使能接收器 (DMA enable receiver) 此位由软件置 1/复位。 1：针对接收使能 DMA 模式 0：针对接收禁止 DMA 模式
位 5:4	保留，必须保持复位值
位 3	**HDSEL**：半双工选择 (Half-duplex selection) 选择单线半双工模式 0：未选择半双工模式 1：选择半双工模式 只有在禁止 UART (UE=0) 时才能写入此位。
位 2:1	保留，必须保持复位值。
位 0	**EIE**：错误中断使能 (Error interrupt enable) 如果发生帧错误、上溢错误或出现噪声标志（LPUART_ISR 寄存器中 FE = 1 或 ORE = 1 或 NF = 1），则需要使用错误中断使能位来使能中断生成。 0：禁止中断 UART_ISR 寄存器中的 FE=1 或 ORE=1 或 NF=1 时生成中断。

26.6.4 波特率寄存器（UARTx_BRR）

偏移地址：0x0C

复位值 ：0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

26.6.5 请求寄存器（UARTx_RQR）

偏移地址：0x10

复位值 ：0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXFRQ	MMRQ	SBKRQ	Res.
												w_r0	w_r0	w_r0	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位 31:23	保留，必须保持复位值。

位 3	**RXFRQ**：接收数据刷新请求 (Receive data flush request) 向该位写入 1 时会将 RXNE 标志清零。 这可以丢弃数据而不对其执行读取操作，并避免发生上溢。
位 2	**MMRQ**：静默模式请求 (Mute mode request) 向此位写入 1 可将 LPUART 置于静默模式，并将 RWU 标志复位。
位 1	**SBKRQ**：发送断路请求 (Send break request) 向此位写入 1 可将 SBKF 标志置 1 并在发送设备可用后立即请求在线路上发送 BREAK。 注： 这种情况下，应用需要在之前插入的所有数据（包括尚未发送的数据）后发送中断字符，软件应等到 *TXE* 标志使能后将 *SBKRQ* 位置 *1*。
位 0	保留，必须保持复位值

26.6.6 中断和状态寄存器（UARTx_ISR）

偏移地址：0x14

复位值 ：0000_00C0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REACK	TEACK	Res.	RWU	SBKF	CMF	BUSY
									r	r		r	r	r	r
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	CTS	CTSIF	Res.	TXE	TC	RXNE	IDLE	ORE	NF	FE	PE
					r	r		r	r	r	r	r	r	r	r
0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0

位 31:23	保留，必须保持复位值。
位 22	**REACK**：接收使能确认标志 (Receive enable acknowledge flag) UART 采用接收使能值时，通过硬件将此位置 1/复位。 此位可用于验证 UART 是否准备好在进入停止模式前接收数据。 注： 如果 *UART* 不支持从停止模式唤醒功能，则此位保留并由硬件强制清零。
位 21	**TEACK**：发送使能确认标志 (Transmit enable acknowledge flag) UART 采用发送使能值时，通过硬件将此位置 1/复位。 通过写入 TE=0 生成空闲帧请求，然后在 LPUART_CR1 寄存器中写入 TE=1 以遵循 TE=0 最短周期时，可使用此位。
位 20	保留，必须保持复位值。
位 19	**RWU**： 接收器从静默模式唤醒 *(Receiver wakeup from Mute mode)* 此位指示 LPUART 是否处于静默模式。当识别出唤醒/静默序列时，此位由硬件清零/置 1。 静默模式控制序列（地址或 IDLE）通过 LPUART_CR1 寄存器中的 WAKE 位选择。 当选择 IDLE 模式下唤醒时，该位只能通过用软件向 LPUART_RQR 寄存器中的 MMRQ 位写 1 的方式置 1。 0：接收器处于活动模式 1：接收器处于静默模式 注： 如果 *UART* 不支持从停止模式唤醒功能，则此位保留并由硬件强制清零
位 18	**SBKF**：发送断路标志 (Send break flag) 此位指示已请求发送断路字符。通过将 1 写入 UART_CR3 寄存器中的 SBKRQ 位，此位由软件置 1。此位在断路发送的停止位期间由硬件自动复位。 0：不发送断路字符 1：将发送断路字符

位 17	**CMF**：字符匹配标志 (Character match flag) 接收到由 ADD[7:0] 定义的字符后由硬件将此位置 1。通过向 LPUART_ICR 寄存器中的 CMCF 写入 1，此位由软件清零。 如果 UART_CR1 寄存器中 CMIE=1，则会生成中断。 0：未检测到字符匹配 1：检测到字符匹配
位 16	**BUSY**：忙标志 (Busy flag) 此位由硬件置 1 和复位。当 RX 线路上正在进行通信（成功检测到起始位）时有效。在接收结束（成功或失败）时复位。 0： LPUART 处于空闲状态（无接收） 1：正在接收
位 15:11	保留，必须保持复位值。
位 10	**CTS**： CTS 标志 (CTS flag) 此位由硬件置 1/复位。此位是对 CTS 输入引脚的状态取反。 0： CTS 线置 1 1： CTS 线复位 注： 如果不支持硬件流控制功能，该位保留并由硬件强制清零。
位 9	**CTSIF**： CTS 中断标志 (CTS interrupt flag) 如果 CTSE 位置 1，当 CTS 输入切换时，此位由硬件置 1。通过将 1 写入 LPUART_ICR 寄存器中的 CTSCF 位，此位由软件清零。 如果 LPUART_CR3 寄存器中 CTSIE=1，则会生成中断。 0： CTS 状态线上未发生变化 1： CTS 状态线上发生变化 注： 如果不支持硬件流控制功能，该位保留并由硬件强制清零。
位 8	保留，必须保持复位值。
位 7	**TXE**：发送数据寄存器为空 (Transmit data register empty) 当 UART_TDR 寄存器的内容已传输到移位寄存器时，此位由硬件置 1。通过对 UART_TDR 寄存器执行写入操作将该位清零。 如果 UART_CR1 寄存器中 TXEIE 位 = 1，则会生成中断。 0：数据未传输到移位寄存器 1：数据传输到移位寄存器 注： 单缓冲区发送期间使用该位。
位 6	**TC**：发送完成 (Transmission complete) 如果已完成对包含数据的帧的发送并且 TXE 置 1，则此位由硬件置 1。如果 UART_CR1 寄存器中 TCIE = 1，则会生成中断。通过向 UART_ICR 寄存器中的 TCCF 写入 1 或向 UART_TDR 寄存器执行写操作，此位由软件清零。 如果 UART_CR1 寄存器中 TCIE = 1，则会生成中断。 0：传送未完成 1：传送已完成 注： 如果 *TE* 位复位且无任何发送正在进行， *TC* 位会立即置 *1*。
位 5	**RXNE**：读取数据寄存器不为空 (Read data register not empty) 当 RDR 移位寄存器的内容已传输到 LPUART_RDR 寄存器时，此位由硬件置 1。通过对 UART_RDR 寄存器执行读入操作将该位清零。也可以通过将 UART_RQR 寄存器中的 RXFRQ 位置 1 将 RXNE 标志位清零。 如果 UART_CR1 寄存器中 RXNEIE = 1，则会生成中断。 0：未接收到数据 1：已准备好读取接收到的数据
位 4	**IDLE**：检测到空闲线路 (IDLE line detected) 检测到空闲线路时，此位由硬件置 1。如果 UART_CR1 寄存器中 IDLEIE=1，则会生成中断。通过向 UART_ICR 寄存器中的 IDLECF 写入 1，此位由软件清零。 0：未检测到空闲线路 1：检测到空闲线路 注： 直到 *RXNE* 位已置 *1* 时（即，当出现新的空闲线路时） *IDLE* 位才会被再次置 *1*。使能静默模式 *(MME=1)* 后，如果 *UART* 未静默 *(RWU=0)*，则 *IDLE* 置 *1*，无论是否通过 *WAKE* 位选择了静默模式。如果 *RWU=1*， *IDLE* 不置 *1*。

位 3	**ORE**：溢出错误 (Overrun error) 在 RXNE = 1 的情况下，当移位寄存器中当前正在接收的数据准备好传输到 RDR 寄存器时，此位由硬件置 1。通过向 UART_ICR 寄存器中的 ORECF 写入 1，此位由软件清零。如果 UART_CR1 寄存器中 RXNEIE=1 或 EIE = 1，则会生成中断。 0：无溢出错误 1：检测到溢出错误 注： 当此位置 *1* 时， *RDR* 寄存器的内容不会丢失，但移位寄存器会被覆盖。 *EIE* 位置 *1* 后，如果在多缓冲区通信中 *ORE* 标志置 *1*，则会生成中断。 *UART_CR3* 寄存器中的 *OVRDIS* 位置 *1* 时，此位将被永久强制清零（无上溢检测）。
位 2	**NF**： START 位噪声检测标志 (START bit Noise detection flag) 当在接收的帧的 START 位上检测到噪声时，此位由硬件置 1。通过向 UART_ICR 寄存器中的 NFCF 写入 1，此位由软件清零。 0：未检测到噪声 1：检测到噪声 注： 该位不会生成中断，因为该位出现的时间与本身生成中断的 *RXNE* 位出现的时间相同。*EIE* 位置 *1* 后，如果在多缓冲区通信中 *NF* 标志置 *1*，则会生成中断。
位 1	**FE**：帧错误 (Framing error) 当检测到去同步化、过度的噪声或中断字符时，此位由硬件置 1。通过向 UART_ICR 寄存器中的 FECF 写入 1，此位由软件清零。 如果 UART_CR1 寄存器中 EIE = 1，则会生成中断。 0：未检测到帧错误 1：检测到帧错误或中断字符
位 0	**PE**：奇偶校验错误 (Parity error) 当在接收器模式下发生奇偶校验错误时，此位由硬件置 1。通过向 UART_ICR 寄存器中的 PECF 写入 1，此位由软件清零。 如果 UART_CR1 寄存器中 PEIE = 1，则会生成中断。 0：无奇偶校验错误 1：奇偶校验错误

26.6.7 中断标志清除寄存器（UARTx_ICR）

偏移地址：0x18

复位值 ：0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CMCF	Res.
														w	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	CTSCF	Res.	Res.	TCCF	Res.	IDLECF	ORECF	NCF	FECF	PECF
						w			w		rw	rw	w	w	w
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位 31:18	保留，必须保持复位值。
位 17	**CMCF**：字符匹配清零标志 (Character match clear flag) 将 1 写入此位时， LPUART_ISR 寄存器中 CMF 标志将清零。
位 16:10	保留，必须保持复位值。
位 9	**CTSCF**： CTS 清零标志 (CTS clear flag) 将 1 写入此位时， LPUART_ISR 寄存器中 CTSIF 标志将清零。
位 8:7	保留，必须保持复位值。
位 6	**TCCF**：发送完成清零标志 (Transmission complete clear flag) 将 1 写入此位时， LPUART_ISR 寄存器中 TC 标志将清零。

位 5	保留，必须保持复位值。
位 4	**IDLECF**：检测到空闲线路清零标志 (Idle line detected clear flag) 将 1 写入此位时， LPUART_ISR 寄存器中 IDLE 标志将清零。
位 3	**ORECF**：上溢错误清零标志 (Overrun error clear flag) 将 1 写入此位时， LPUART_ISR 寄存器中 ORE 标志将清零。
位 2	**NCF**：检测到噪声清零标志 (Noise detected clear flag) 将 1 写入此位时， LPUART_ISR 寄存器中 NF 标志将清零。
位 1	**FECF**：帧错误清零标志 (Framing error clear flag) 将 1 写入此位时， LPUART_ISR 寄存器中 FE 标志将清零。
位 0	**PECF**：奇偶校验错误清零标志 (Parity error clear flag) 将 1 写入此位时， LPUART_ISR 寄存器中 PE 标志将清零。

26.6.8 数据接收寄存器 （UARTx_RDR）

偏移地址：0x1C

复位值 ：未定义

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

位 31:9	保留，必须保持复位值。
位 8:0	**RDR[8:0]**：接收数据值 (Receive data value) 包含接收到的数据字符。 RDR 寄存器在输入移位寄存器和内部总线之间提供了并行接口。 在使能奇偶校验位的情况下进行接收时，从 MSB 位中读取的值为接收到的奇偶校验位。

26.6.9 数据发送寄存器 （UARTx_TDR）

偏移地址：0x20

复位值 ：未定义

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

位 31:9	保留，必须保持复位值。

位 8:0	**TDR[8:0]**：发送数据值 (Transmit data value) 包含要发送的数据字符。 TDR 寄存器在内部总线和输出移位寄存器之间提供了并行接口（请参见图 *235*）。 在使能奇偶校验位的情况下（LPUART_CR1 寄存器中的 PCE 位被置 1）进行发送时，由于 MSB 的写入值（位 7 或位 8，具体取决于数据长度）会被奇偶校验位所取代，因此该值不起任何作用。 注： 只能在 *TXE=1* 时写入此寄存器。

27 低功耗通用异步接收器 (LPUART)

27.1 简介

低功耗通用异步接收器（LPUART）是一种 UART，允许有限功耗下进行全双工 UART 通信。仅需 32.768 kHz LSE 时钟即可进行高达 9600 baud/s 的 UART 通信。当 LPUART 由与 LSE 时钟不同的时钟源计时，可以达到更高的波特率。

即使当微控制器处于停止模式，能耗极低时，LPUART 也会等待 UART 帧的到来。LPUART 包含所有必要的硬件支持，使在最小功耗下可以进行异步串行通信。

它支持半双工单线通信和调制解调器操作 (CTS/RTS)，还支持多处理器通信。DMA（直接存储器访问）可用于数据发送/接收。

27.2 LPUART 主要特性

- 全双工异步通信
- NRZ 标准格式（标记/空格）
- 使用 32.768 kHz 时钟源时波特率可编程为 300 波特/s 到 9600 波特/s。使用高频时钟源可实现更高的波特率
- 双时钟域允许：
  - UART 功能和从停止模式唤醒
  - 方便的波特率编程，独立于 PCLK 重新编程
- 数据字长度可编程（7 位、8 位或 9 位）
- 可编程的数据顺序，最先移位 MSB 或 LSB
- 停止位可配置（支持 1 个或 2 个停止位）
- 单线半双工通信
- 使用 DMA 实现连续通信
- 使用中央 DMA 在预留的 SRAM 缓冲区中收/发字节。
- 发射器和接收器有单独的使能位
- 发送和接收的单独信号极性控制
- Tx/Rx 引脚配置可交换
- 调制解调器和 RS-485 收发器的硬件流控制
- 传输检测标志：
  - 接收缓冲区已满
  - 发送缓冲区为空
  - BUSY 标志和发送结束标志
- 奇偶校验控制：
  - 发送奇偶校验位
  - 检查接收的数据字节的奇偶性
- 四个错误检测标志：
  - 上溢错误
  - 噪声检测
  - 帧错误

- 奇偶校验错误
- 十四个具有标志位的中断源
- 多处理器通信

  如果地址不匹配，LPUART 将进入静默模式。
- 从静默模式唤醒（通过空闲线检测或地址标记检测）

27.3 LPUART 特性实现

LPUART 支持的特性有：

- 调制解调器的硬件流控
- 使用 DMA 进行连续通信
- 多处理器通信
- 单线半双工通信
- 双时钟域和从停止模式唤醒
- 驱动器使能
- USART 数据长度 7/8/9 位

27.4 LPUART 功能说明

任何 LPUART 双向通信均需要至少两个引脚：接收数据输入引脚(RX)和发送数据输出引脚(TX)：

- **RX**：接收数据输入引脚。

串行数据输入引脚。

- **TX**：发送数据输出引脚。

如果关闭发送器，该输出引脚模式由其 I/O 端口配置决定。如果使能了发送器但没有待发送的数据，则 TX 引脚处于高电平。在单线模式下，该 I/O 用于发送和接收数据。

正常 LPUART 模式下，通过这些引脚以帧的形式发送和接收串行数据：

- 发送或接收前保持空闲线路
- 起始位
- 数据字（7 位、8 位或 9 位），最低有效位在前
- 用于指示帧传输已完成的 1 个、2 个停止位
- LPUART 接口使用波特率发生器
- 状态寄存器(LPUART_ISR)
- 接收和发送数据寄存器（LPUART_RDR、LPUART_TDR）
- 波特率寄存器(LPUART_BRR)

有关各个位的定义，请参见第 5.25.7 节：LPUART 寄存器

在 RS232 硬件流控制模式下需要以下引脚：

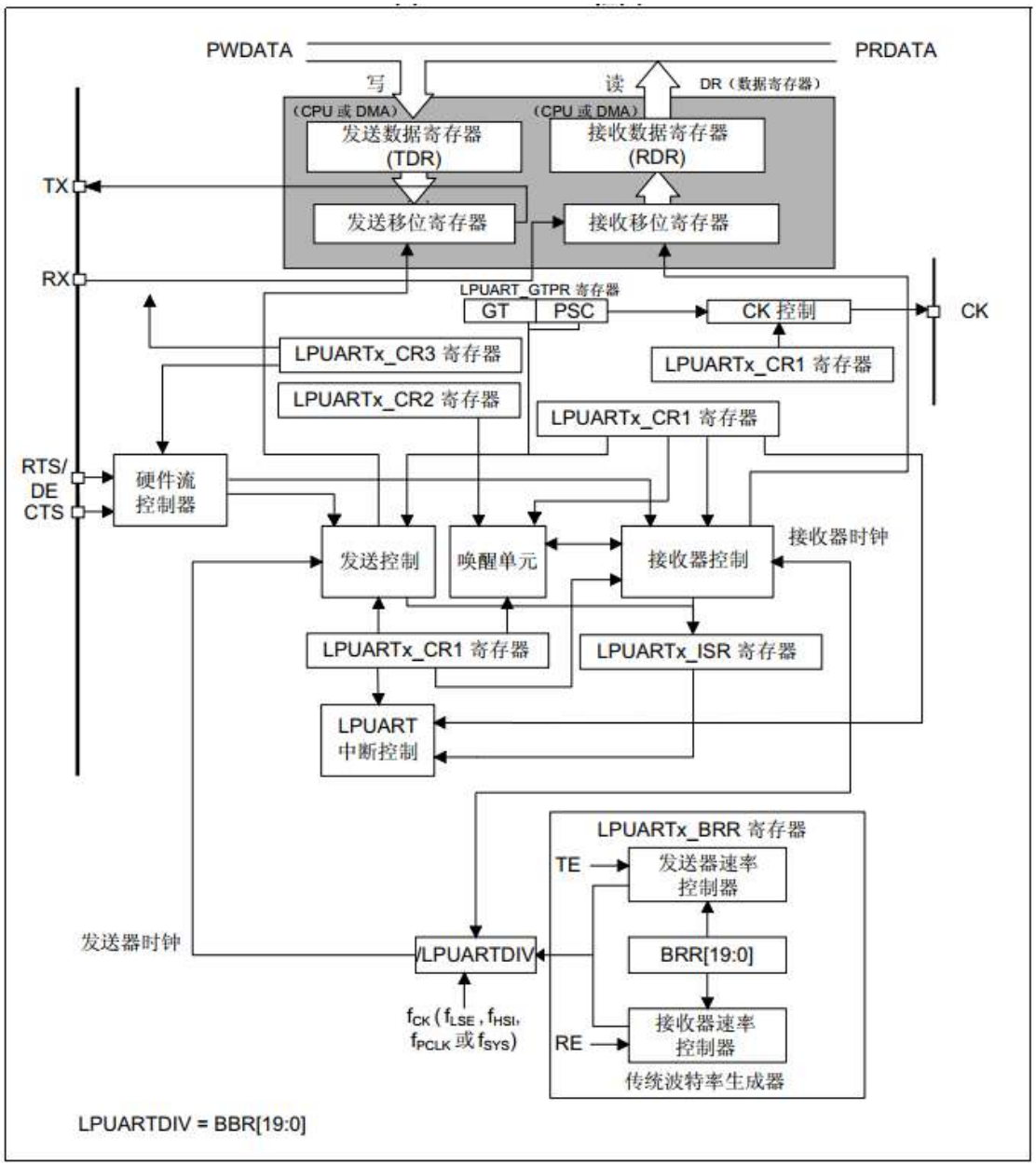
- **CTS**：“清除以发送”用于在当前传输结束时阻止数据发送（高电平时）
- **RTS**：“请求以发送”用于指示 LPUART 已准备好接收数据（低电平时）。

在 RS485 硬件控制模式下需要以下引脚：

- **D E**：“驱动器使能”用于激活外部收发器的发送模式。

注：DE 和 RTS 共用同一个引脚

图 27-1 LPUART 框图



27.4.1 LPUART 字符说明

可通过对 LPUART_CR1 寄存器中的 M[1:0]位进行编程来选择 7 位、8 位或 9 位的字长（图 27-2 字长编程）。

- 7 位字符长度：M[1:0] = 10
- 8 位字符长度：M[1:0] = 00
- 9 位字符长度：M[1:0] = 01

在默认情况下，信号（TX 或 RX）在起始位工作期间处于低电平状态。在停止位工作期间处于高电平状态。

通过极性配置控制，可以单独针对每个信号对这些值取反。

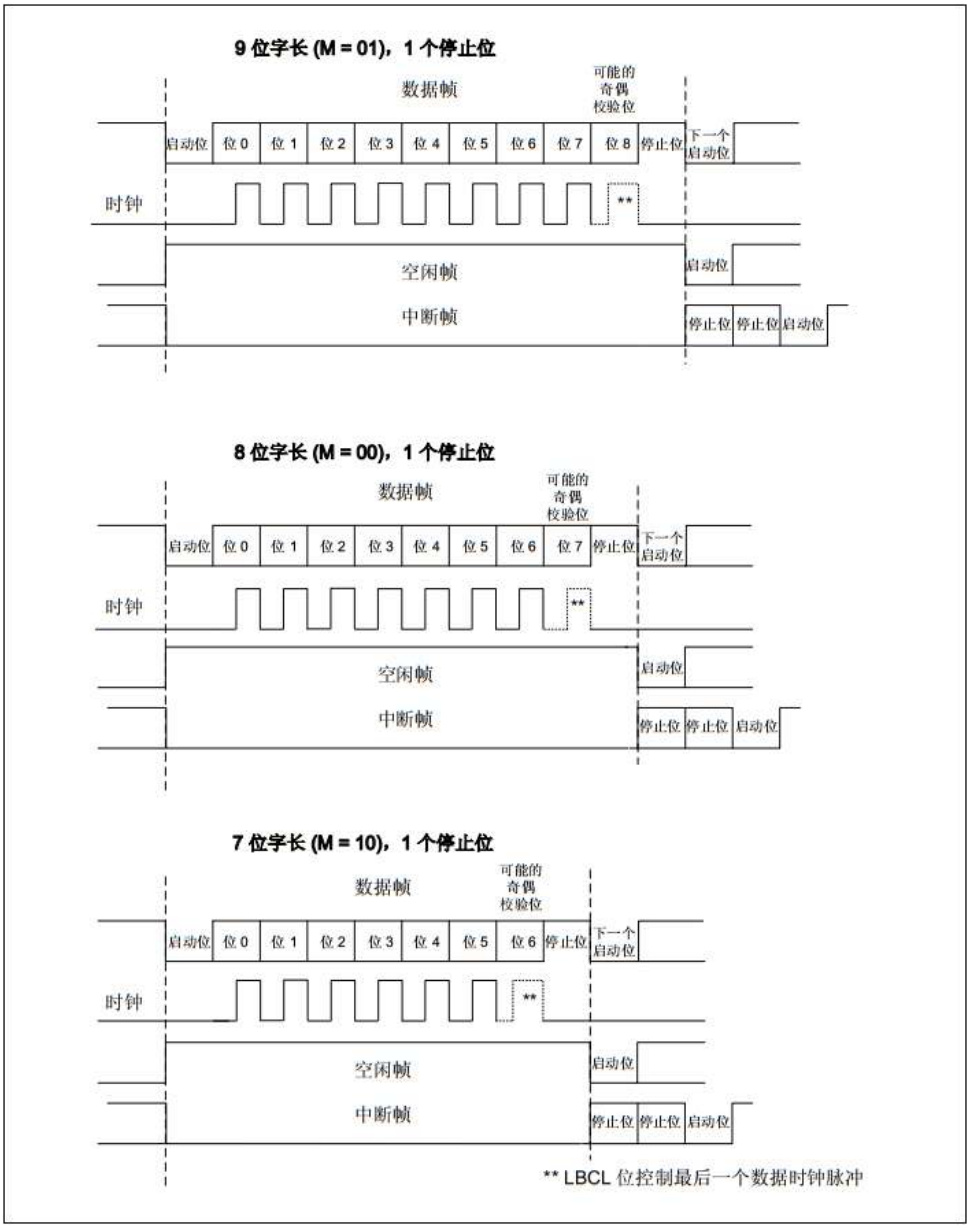
空闲字符可理解为整个帧周期内电平均为“1”。（停止位的电平也是“1”）。

停止字符可理解为在一个帧周期内接收到的电平均为“0”。发送器在中断帧的末尾插入 2 个停止位。

发送和接收由通用波特率发生器驱动，发送器和接收器的使能位分别置 1 时将生成相应的发送时钟和接收时钟。

下面给出了各个块的详细信息。

图 27-2 字长编程



27.4.2 LPUART 发送器

发送器可发送 7 位、8 位或 9 位的数据字，具体取决于 M 位的状态。要激活发送器功能，必须将发送使能位(TE)置 1。发送移位寄存器中的数据输出到 TX 引脚。

字符发送

LPUART 发送期间，首先通过 TX 引脚移出数据的最低有效位（默认配置）。该模式下，LPUART_TDR 寄存器的缓冲区(TDR)位于内部总线和发送移位寄存器之间（图 27-1 LPUART 框图）。

每个字符前面都有一个起始位，其逻辑电平在一个位周期内为低电平。字符由可配置数量的停止位终止。

LPUART 支持以下停止位：1 个和 2 个停止位。

注：向 LPUART_TDR 中写入要发送的数据前，TE 位必须先置 1。

数据发送期间不应复位 TE 位。发送期间复位 TE 位会冻结波特率计数器，从而将损坏 TX 引脚上的数据。当前传输的数据将会丢失。

使能 TE 位后，将会发送空闲帧。

可配置的停止位

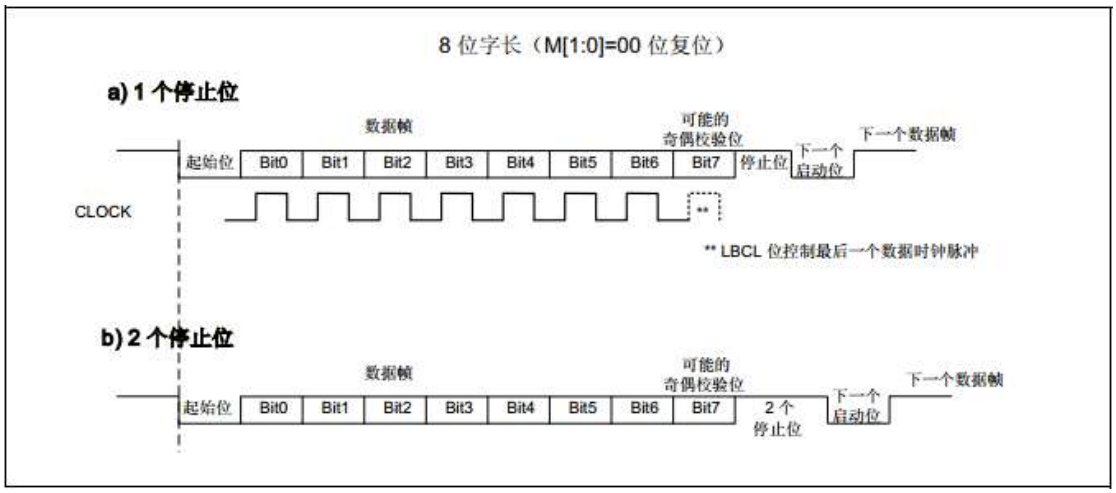
可以在控制寄存器 2 的位 13 和位 12 中编程将随各个字符发送的停止位的数量。

- **1 个停止位：** 这是停止位数量的默认值。
- **2 个停止位：** 正常 LPUART 模式、单线模式和调制解调器模式支持该值。

空闲帧发送将包括停止位。

中断发送是 10 个低电平位（M[1:0] = 00 时）、 11 个低电平位（M[1:0] = 01 时）或 9 个低电平位（M[1:0] = 10 时），然后是 2 个停止位。无法传送长中断（中断长度大于 9/10/11 个低电平位）。

图 27-3 可配置的停止位



字符发送步骤

1. 对 LPUART_CR1 中的 M 位进行编程以定义字长。
2. 使用 LPUART_BRR 寄存器选择所需波特率。
3. 对 LPUART_CR2 中的停止位数量进行编程。
4. 通过向 LPUART_CR1 寄存器中的 UE 位写入 1 使能 LPUART。
5. 如果将进行多缓冲区通信，请选择 LPUART_CR3 中的 DMA 使能 (DMAT)。按照多缓冲区通信中的解释说明配置 DMA 寄存器。
6. 将 LPUART_CR1 中的 TE 位置 1 以便在首次发送时发送一个空闲帧。
7. 在 LPUART_TDR 寄存器中写入要发送的数据（该操作将清零 TXE 位）。为每个要在单缓冲区模式下发送的数据重复这一步骤。
8. 向 LPUART_TDR 寄存器写入最后一个数据后，等待至 TC=1。这表明最后一个帧的传送已完成。禁止 LPUART 或进入暂停模式时需要此步骤，以避免损坏最后一次发送。单字节通信始终通过向发送数据寄存器写入数据将 TXE 位清零。

TXE 位由硬件置 1，它表示：

- 数据已从 LPUART_TDR 寄存器移到移位寄存器中且数据发送已开始。

- LPUART_TDR 寄存器为空。
- LPUART_TDR 寄存器中可写入下一个数据，而不会覆盖前一个数据。

TXEIE 位置 1 时该标志位会生成中断。

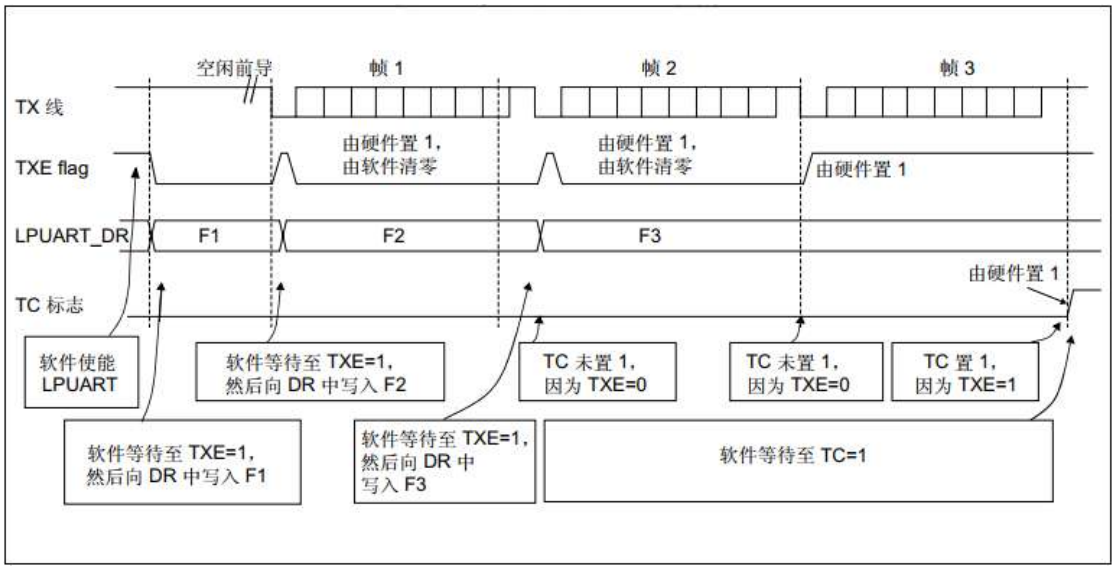
发送时，要传入 LPUART_TDR 寄存器的写指令中存有 TDR 寄存器中的数据，接下来，该数据将在当前进行的发送结束时复制到移位寄存器中。

未发送时，要传入 LPUART_TDR 寄存器的写指令将数据置于移位寄存器中，数据发送开始时，TXE 位置 1。

如果帧已发送（停止位后）且 TXE 位置 1， TC 位将变为高电平。如果 LPUART_CR1 寄存器中的 TCIE 位置 1，将生成中断。

向 LPUART_TDR 寄存器中写入最后一个数据后，必须等待至 TC=1，之后才可禁止 LPUART 或使微控制器进入低功率模式（图 27-4 发送时的 TC/TXE 行为）。

图 27-4 发送时的 TC/TXE 行为



中断字符

将 SBKRQ 位置 1 将发送一个中断字符。中断帧的长度取决于 M 位（图 27-2 字长编程）。

如果将“1”写入 SBKRQ 位，则当前字符发送完成后，将在 TX 线路上发送一个中断字符。通过写操作将 SBKF 位置 1 并在中断字符发送完成时（发送中断字符后的停止位期间），此位由硬件复位。LPUART 在中断帧末尾的两位持续时间内插入一个逻辑“1”信号 (STOP)，以确保识别下个帧的起始位。

这种情况下，应用需要在之前插入的所有数据（包括尚未发送的数据）后发送中断字符，软件应等到 TXE 标志使能后将 SBKRQ 位置 1。

空闲字符

将 TE 位置 1 会驱动 LPUART 在第一个数据帧之前发送一个空闲帧。

27.4.3 LPUART 接收器

LPUART 可接收 7 位、8 位或 9 位的数据字，具体取决于 LPUART_CR1 寄存器中的 M 位。

起始位检测

在 LPUART 中，进行 START 位检测时，首先应在 Rx 线路上检测到下降沿，然后在起始位 中间采样以确认电平是否仍为“0”。如果起始采样为“1”，则噪声错误标志 (NF) 置 1， START 位将被丢弃，接收器将等待新的 START 位。否则，接收器将继续正常采样所有传入位。

字符接收

LPUART 接收期间，首先通过 RX 引脚移入数据的最低有效位（默认配置）。该模式下，LPUART_R

DR 寄存器的缓冲区 (RDR) 位于内部总线和接收移位寄存器之间。

字符接收步骤

1. 对 LPUART_CR 中的 M 位进行编程以定义字长。
2. 使用波特率寄存器 LPUART_BRR 选择所需波特率。
3. 对 LPUART_CR2 中的停止位数量进行编程。
4. 通过向 LPUART_CR1 寄存器中的 UE 位写入 1 使能 LPUART。
5. 如果将进行多缓冲区通信，请选择 LPUART_CR3 中的 DMA 使能 (DMAR)。按照多缓冲区通信中的解释说明配置 DMA 寄存器。
6. 将 RE 位 LPUART_CR1 置 1。这一操作将使能接收器开始搜索起始位。

接收到字符时

- RXNE 位置 1。这表明移位寄存器的内容已传送到 RDR。也就是说，已接收到并可读取数据（及其相应的错误标志）。
- 如果 RXNEIE 位置 1，则会生成中断。
- 如果接收期间已检测到帧错误、噪声错误或上溢错误，错误标志位可置 1。也可使用 RXNE 位将 PE 标志置 1。
- 在多缓冲区模式下，每接收到一个字节后 RXNE 均置 1，然后通过 DMA 对接收数据寄 存器执行读操作清零。
- 在单缓冲区模式下，通过软件对 LPUART_RDR 寄存器执行读操作将 RXNE 位清零。也可以通过将 LPUART_RQR 寄存器中的 RXFRQ 位置 1 将 RXNE 标志位清零。 RXNE 位必须在结束接收下一个字符前清零，以避免发生上溢错误。

中断字符

接收到中断字符时，LPUART 将会按照帧错误对其进行处理。

空闲字符

检测到空闲帧时，处理步骤与接收到数据的情况相同；如果 IDLEIE 位置 1，则会产生中断。

上溢错误

如果在 RXNE 未复位时接收到字符，则会发生上溢错误。RXNE 位清零前，数据无法从移位 寄存器传送到 RDR 寄存器。

每接收到一个字节后，RXNE 标志位都将置 1。当 RXNE 标志位是 1 时，如果在接收到下一个 数据或尚未处理上一个 DMA 请求时，则会发生上溢错误。发生上溢错误时：

- ORE 位置 1。
- RDR 中的内容不会丢失。对 LPUART_RDR 执行读操作时可使用先前的数据。
- 移位寄存器将被覆盖。之后，上溢期间接收到的任何数据都将丢失。
- 如果 RXNEIE 位置 1 或 EIE 位置 1，则会生成中断。
- 通过设置 ICR 寄存器中的 ORECF 位复位 ORE 位。

注：ORE 位置 1 时表示至少 1 个数据丢失。存在两种可能：

- 如果 RXNE=1，则最后一个有效数据存储于接收寄存器 RDR 中并且可进行读取；
- 如果 RXNE=0，则表示最后一个有效数据已被读取，因此 RDR 中没有要读取的数据。接收到新（和丢失）数据的同时已读取 RDR 中的最后一个有效数据时，会发生该情况。

选择时钟源

通过复位和时钟控制系统(RCC)选择时钟源。时钟源必须在使能 LPUART（通过将 UE 位置 1）前选择。

必须遵循以下两个条件选择时钟源：

- 可在低功耗模式下使用 LPUART
- 通信速度。

时钟源频率为 fCK。

当支持双时钟域和从停止模式唤醒功能时，时钟源可以是以下其中之一：fPCLK（默认值）、fLSE、fHSI 或 fSYS。否则，LPUART 时钟源是 fPCLK。

选择 fLSE 或 fHSI 作为时钟源可允许 LPUART 在 MCU 处于低功耗模式时接收数据。必要时，LPUART 可基于所接收的数据和唤醒模式的选择来唤醒 MCU，以通过用软件读取 LPUART_RDR 寄存器的方式或通过 DMA 的方式传输已接收数据。

对于其它时钟源，系统必须激活才能进行 LPUART 通信。时钟源还决定通信速度范围（尤其是最大通信速度）。接收器在尽可能靠近波特周期中间的位置采样每个传入波特。每个传入波特仅进行一次采样。

注： 不进行数据噪声检测。

帧错误 (Framing error)

如果接收数据时未在预期时间内识别出停止位，从而出现同步失效或过度的噪声，则会检测到帧错误。

检测到帧错误时：

- FE 位由硬件置 1。
- 无效数据从移位寄存器传送到 LPUART_RDR 寄存器。
- 单字节通信时无中断产生。然而，在 RXNE 位产生中断时，该位出现上升沿。多缓冲区通信时，LPUART_CR3 寄存器中的 EIE 位置 1 时将发出中断。

通过将 1 写入 LPUART_ICR 寄存器中的 FECF 位来复位 FE 位。

接收期间可配置的停止位

可通过控制寄存器 2 中的控制位配置要接收的停止位的数量 - 可以是 1 个或 2 个（正常模式下）。

- 1 个停止位： 将在第 8、第 9 和第 10 次采样时对 1 个停止位进行采样。
- 2 个停止位： 将在第 2 个停止位的中间对 2 个停止位进行采样。RXNE 和 FE 标志在此采样期间（例如，在第 2 个停止位期间）置 1。发生帧错误时不检测第 1 个停止位。

27.4.4 LPUART 波特率生成

接收器和发送器（Rx 和 Tx）的波特率设置为 LPUART_BRR 寄存器中编程的相同值。

接收器和发送器（Rx 和 Tx）的波特率设置为 LPUART_BRR 寄存器中编程的相同值。

$$\text{Tx/Rx 波特} = \frac{256*\text{fck}}{\text{LPUARTDIV}}$$

LPUARTDIV 在 LPUART_BRR 寄存器中编码。

注： 对 LPUART_BRR 执行写操作后，波特率计数器更新为波特率寄存器中的新值。因此，波特率寄存器的值不应在通信时发生更改。

禁止在 LPUART_BRR 寄存器中写入小于 0x300 的值。

fck 必须在[3x 波特率，4096x 波特率]范围内。

LPUART 时钟源为 LSE 时可达到的最大波特率为 9600 波特。当 LPUART 由与 LSE 时钟不同的时钟源驱动时，可以达到更高的波特率。例如，如果 USART 时钟源是系统时钟（最大为 32 MHz）,可达到的最大波特率为 10M 波特。

表 27-1 fck = 32,768 KHz 时编程的波特率的错误计算

波特率		fCK = 32,768 KHz		
序号	所需值	实际值	波特率寄存器中编程的值	误差 % = (计算值 - 所需值) / 所需波特率
1	300Bps	300Bps	0x6D3A	0
2	600Bps	600Bps	0x369D	0
3	1200Bps	1200.087Bps	0x1B4E	0.007
4	2400Bps	2400.17Bps	0xDA7	0.007
5	4800Bps	4801.72Bps	0x6D3	0.035
6	9600Bps	9608.94Bps	0x369	0.093

表 27-2 fck = 32MHz 时编程的波特率的错误计算

波特率	fCK = 32MHz		
所需值	实际值	波特率寄存器中编程的值	误差 % = (计算值 - 所需值) / 所需波特率
9600Bps	9608.94Bps	9600,004	D0555
19200	19200.030	682AA	0.0001
38400	38400.06	34155	0.0001
57600	57600.09	22B8E	0.0001
115200	115200.18	115C7	0.0001
230400	230403.60	8AE3	0.0015
460800	460820.16	4571	0.004
921600	921692.17	22B8	0.01
4000000	4000000.00	800	0
10000000	10002442.00	333	0.024

27.4.5 LPUART 接收器对时钟偏差的容差

仅当总时钟系统偏差小于 LPUART 接收器的容差时， LPUART 异步接收器才能正常工作。

影响总偏差的因素包括：

- DTRA：发送器误差引起的偏差（其中还包括发送器本地振荡器的偏差）
- DQUANT：接收器的波特率量化引起的误差
- DREC：接收器本地振荡器的偏差
- DTCL：传输线路引起的偏差（通常是由于收发器所引起，它可能会在低电平到高电平转换时序与高电平到低电平转换时序之间引入不对称）

DTRA + DQUANT + DREC + DTCL + DWU < LPUART 接收器容差

其中

DWU 是使用从停止模式唤醒时因采样点偏差产生的误差。

M[1:0] = 01 时：

$$\mathrm{DWU} = \frac{t_{WULPUART}}{11 \times T_{bit}}$$

M[1:0] = 00 时：

$$\mathrm{DWU} = \frac{t_{WULPUART}}{10 \times T_{bit}}$$

M[1:0] = 10 时：

$$\mathrm{DWU} = \frac{t_{WULPUART}}{9 \times T_{bit}}$$

$t_{WULPUART}$是检测到唤醒事件到时钟（由外设请求）与调压器均就绪的时间。

LPUART 接收器在表 27-3 BRR[3:0] 不等于 0000 时的 LPUART 接收器容差中指定的最大容许偏差下可正确接收数据：

- 由 LPUARTx_CR1 寄存器中的 M 位定义的 7 位、 8 位或 9 位字符长度
- 1 个或 2 个停止位

表 27-3 BRR[3:0] 不等于 0000 时的 LPUART 接收器容差

M 位	768 ≤ BRR <1024	1024 ≤ BRR < 2048	2048 ≤ BRR < 4096	4096 ≤ BRR
8 位 (M=00)， 1 个停止位	1.82%	2.56%	3.90%	4.42%
9 位 (M=01)， 1 个停止位	1.69%	2.33%	2.53%	4.14%
7 位 (M=10)， 1 个停止位	2.08%	2.86%	4.35%	4.42%
8 位 (M=00)， 2 个停止位	2.08%	2.86%	4.35%	4.42%
9 位 (M=01)， 2 个停止位	1.82%	2.56%	3.90%	4.42%

7 位 (M=10)，2 个停止位	2.34%	3.23%	4.92%	4.42%

注： 当接收的帧恰好包含 10 个（M 位 = 00）、 11 个（M 位 = 01）或 9 个（M 位 = 10）位持续时间的空闲帧时，表 27-3 BRR[3:0] 不等于 0000 时的 LPUART 接收器容差中指定的数据可能与特例中的数据略微不同。

27.4.6 使用 LPUART 进行多处理器通信

可以通过 LPUART 进行多处理器通信（多个 LPUART 连接在一个网络中）。例如，其中一个 LPUART 可以是主器件，其 TX 输出与其它 LPUART 的 RX 输入相连接。其它 USART 为从 USART，其各自的 TX 输出在逻辑上通过与运算连在一起，并与主 USART 的 RX 输入相连接。

在多处理器配置中，理想情况下通常只有预期的消息接收方主动接收完整的消息内容，从而减少由所有未被寻址的接收器造成的冗余 LPUART 服务开销。

可通过静默功能将未被寻址的器件置于静默模式下。为了使用静默模式功能，必须将 LPUART_CR1 寄存器中的 MME 位置 1。

在静默模式下：

- 不得将接收状态位置 1。
- 禁止任何接收中断。
- LPUART_ISR 寄存器中的 RWU 位置 1。在某些情况下， RWU 可以由硬件或软件通过 LPUART_RQR 寄存器中的 MMRQ 位自动控制。

根据 LPUART_CR1 寄存器中 WAKE 位的设置， LPUART 可使用以下两种方法进入或退出静音模式：

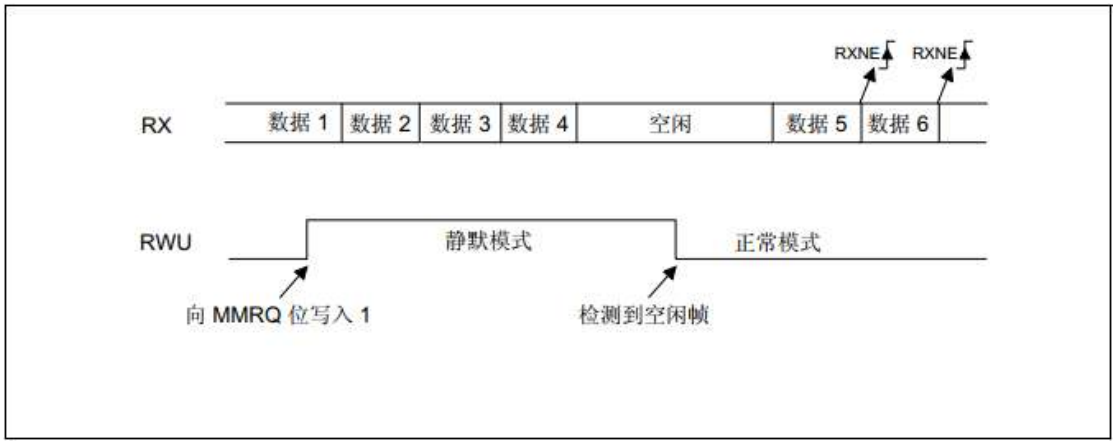
- 如果 WAKE 位被复位，则进行空闲线路检测，
- 如果 WAKE 位置 1，则进行地址标记检测。

空闲线路检测 (WAKE=0)

当向 MMRQ 位写入 1 且 RWU 位自动置 1 时， LPUART 进入静默模式。

当检测到空闲帧时，它会被唤醒。此时 RWU 位会由硬件清零，但 LPUART_ISR 寄存器中的 IDLE 位不会置 1。 图 27-5 使用空闲线路检测时的静默模式中给出了使用空闲线路检测时静默模式行为的示例。

图 27-5 使用空闲线路检测时的静默模式



注： 如果在 IDLE 字符已经过去时将 MMRQ 位置 1，将不会进入静默模式（RWU 未置 1）。如果在线路处于空闲状态时激活 LPUART，在一个 IDLE 帧持续时间后（不只在接收一个字符帧后）会检测到空闲状态。

4 位/7 位地址标记检测 (WAKE=1)

在此模式下，如果字节的 MSB 为 1，则将这些字节识别为地址，否则将其识别为数据。在地址字节中，目标接收器的地址位于 4 个或 7 个 LSB 中。 7 位或 4 位地址检测通过 ADDM7 位来选择。接收器会将此 4 位/7 位字与其地址进行比较，该接收器的地址在 LPUART_CR2 寄存器的 ADD 位中进行设置。

注： 在 7 位和 9 位数据模式下，地址检测分别在 6 位和 8 位地址上完成（ADD[5:0] 和 ADD[7:0]）。

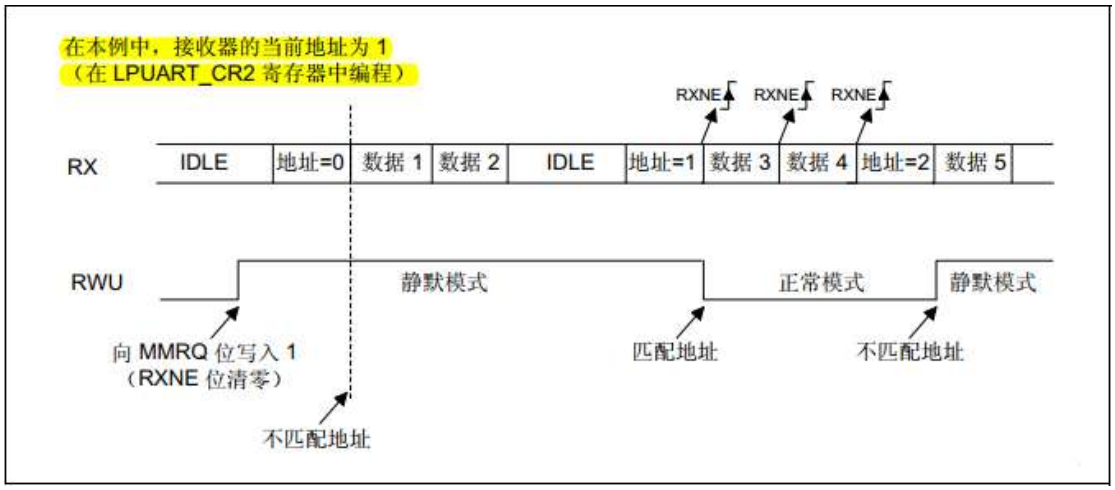
当接收到与其编程地址不匹配的地址字符时， LPUART 会进入静默模式。此时，RWU 位将由硬件置 1。

LPUART 进入静默模式后，RXNE 标志不会针对此地址字节置 1，也不会发出中断或 DMA 请求。

当向 MMRQ 位写入 1 时，LPUART 也会进入静默模式。这种情况下，RWU 位也自动置 1。当接收到与编程地址匹配的地址字符时，LPUART 会退出静默模式。然后 RWU 位被清零，可以开始正常接收后续字节。由于 RWU 位已清零，RXNE 位会针对地址字符置 1。

图 27-6 使用地址标记检测时的静默模式 中给出了使用地址标记检测时静默模式行为的示例

图 27-6 使用地址标记检测时的静默模式



27.4.7 LPUART 奇偶控制

将 LPUART_CR1 寄存器中的 PCE 位置 1，可以使能奇偶校验控制（发送时生成奇偶校验位，接收时进行奇偶校验检查）。根据 M 位定义的帧长度， 表 27-4 帧格式中列出了可能的 LPUART 帧格式

表 27-4 帧格式

M 位	PCE 位	LPUART 帧(1)
00	0	| SB | 8 位数据 | STB |
00	1	| SB | 7 位数据 | PB | STB |
01	0	| SB | 9 位数据 | STB |
01	1	| SB | 8 位数据 | PB | STB |
10	0	| SB | 7 位数据 | STB |
10	1	| SB | 6 位数据 | PB | STB |

1. 图注： SB：起始位， STB：停止位， P：奇偶校验位。

偶校验

对奇偶校验位进行计算，使帧和奇偶校验位中“1”的数量为偶数（帧由 6 个、 7 个或 8 个 LSB 位组成，具体取决于 M 位的值）。

例如，如果数据=00110101 且 4 个位置 1，则在选择偶校验（LPUART_CR1 寄存器中的 PS 位 = 0）时，校验位为 0。

奇校验

对奇偶校验位进行计算，使帧和奇偶校验位中“1”的数量为奇数（帧由 6 个、 7 个或 8 个 LSB 位组成，具体取决于 M 位的值）。

例如，如果数据=00110101 且 4 个位置 1，则在选择奇校验（LPUART_CR1 寄存器中的 PS 位 = 1）时，校验位为 1。

接收时进行奇偶校验检查

如果奇偶校验检查失败，则 LPUART_ISR 寄存器中的 PE 标志置 1；如果 LPUART_CR1 寄存器中 PEIE 位置 1，则会生成中断。通过软件将 1 写入 LPUART_ICR 寄存器中的 PECF 位来清零 PE 标志。

发送时的奇偶校验生成

如果 LPUARTx_CR1 寄存器中的 PCE 位置 1，则在数据寄存器中所写入数据的 MSB 位会进行传送，但是会由奇偶校验位进行更改（如果选择偶校验 (PS=0)，则“1”的数量为偶数；如果选择奇校验 (PS=1)，则“1”的数量为奇数）。

27.4.8 使用 LPUART 单线半双工通信

通过将 LPUART_CR3 寄存器中的 HDSEL 位置 1 来选择单线半双工模式。在此模式下，必须将以下位清零：

- LPUART_CR2 寄存器中的 LINEN 和 CLKEN 位，
- LPUART_CR3 寄存器中的 SCEN 和 IREN 位。

LPUART 可以配置为遵循单线半双工协议，其中 TX 和 RX 线路从内部相连接。使用 LPUART_CR3 中的控制位 HDSEL 可在半双工通信和全双工通信间进行选择。

一旦向 HDSEL 位写入 1：

- TX 和 RX 线路从内部相连接
- 不能再使用 RX 引脚
- 无数据传输时， TX 引脚始终处于释放状态。因此，它在空闲状态或接收过程中用作标准 I/O。这意味着，必须对 I/O 进行配置，以便将 TX 配置为复用功能开漏并外接上拉电阻。

除此之外，通信协议与正常 LPUART 模式下的通信协议相似。此线路上的任何冲突必须由软件管理（例如，使用中央仲裁器）。尤其要注意，发送过程永远不会被硬件封锁，只要数据是在 TE 位置 1 的情况下写入，发送就会持续进行。

注： 在 LPUART 中，当进行 1 个停止位配置时， RXNE 标志在停止位中间置 1。

27.4.9 使用 LPUART 在 DMA 模式下进行连续通信

LPUART 能够使用 DMA 进行连续通信。接收缓冲区和发送缓冲区的 DMA 请求是独立生成的。

注： 按照第 5.21.4.3 节中的说明使用 LPUART。可以将 LPUART_ISR 寄存器中的 TXE/ RXNE 标志清零，从而实现连续通信

使用 DMA 进行发送

将 LPUART_CR3 寄存器中的 DMAT 位置 1 可以使能 DMA 模式进行发送。当 TXE 位置 1 时，可将数据从 SRAM 区（通过 DMA 外设，请参见第 246 页上的第 11 节：直接存储器访问控制器 (DMA)）加载到 LPUART_TDR 寄存器。要映射一个 DMA 通道以进行 LPUART 发送，请按以下步骤操作（x 表示通道编号）：

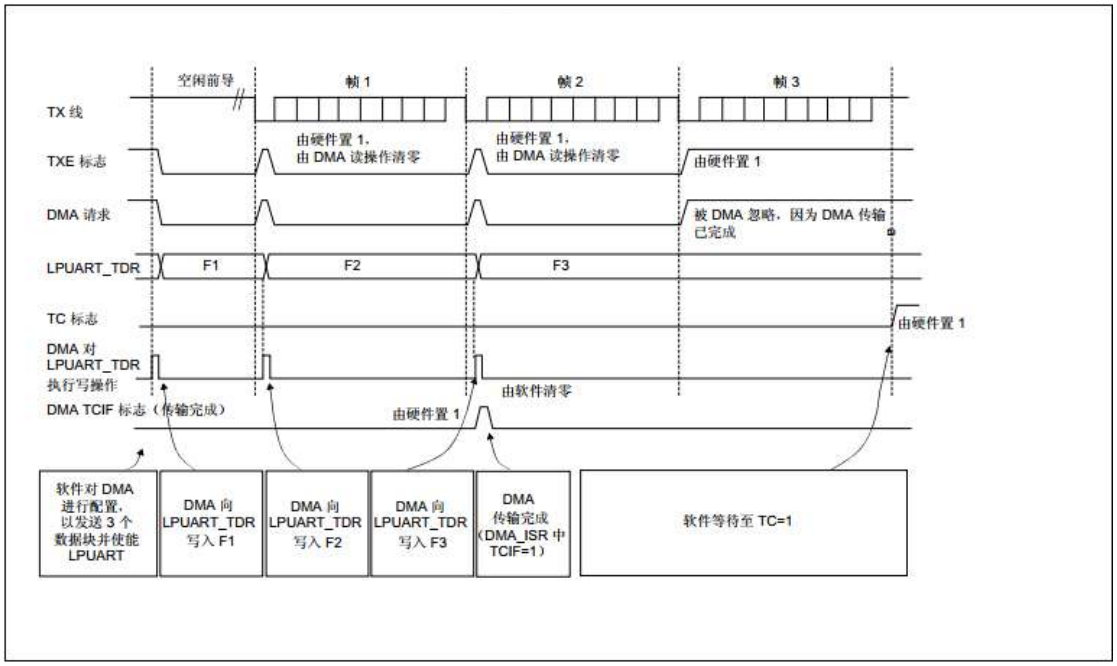
1. 在 DMA 控制寄存器中写入 LPUART_TDR 寄存器地址，将其配置为传输的目标地址。每次发生 TXE 事件后，数据都从存储器移动到此地址。
2. 在 DMA 控制寄存器中写入存储器地址，将其配置为传输的源地址。每次发生 TXE 事件后，数据都从这个存储区域加载到 LPUART_TDR 寄存器中。
3. 在 DMA 控制寄存器中配置要传输的总字节数。
4. 在 DMA 寄存器中配置通道优先级
5. 根据应用的需求，在完成一半或全部传输后产生 DMA 中断。
6. 通过将 LPUART_ICR 寄存器中的 TCCF 位置 1，将 LPUART_ISR 寄存器中的 TC 标志清零。
7. 在 DMA 寄存器中激活该通道。

当达到在 DMA 控制器中设置的数据传输量时， DMA 控制器会在 DMA 通道的中断向量上产生一个中断。

在发送模式下， DMA 对所有要发送的数据执行了写操作（DMA_ISR 寄存器中的 TCIF 标志置 1）后，可以对 TC 标志进行监视，以确保 LPUART 通信已完成。在禁止 LPUART 或进入停止模式前必须执行此步骤，以避免损坏最后一次发送。软件必须等待直到 TC=1。 TC 标志在所有数据发送期间都保持清零状态，

然后在最后一帧发送结束时由硬件置 1。

图 27-7 使用 DMA 进行发送



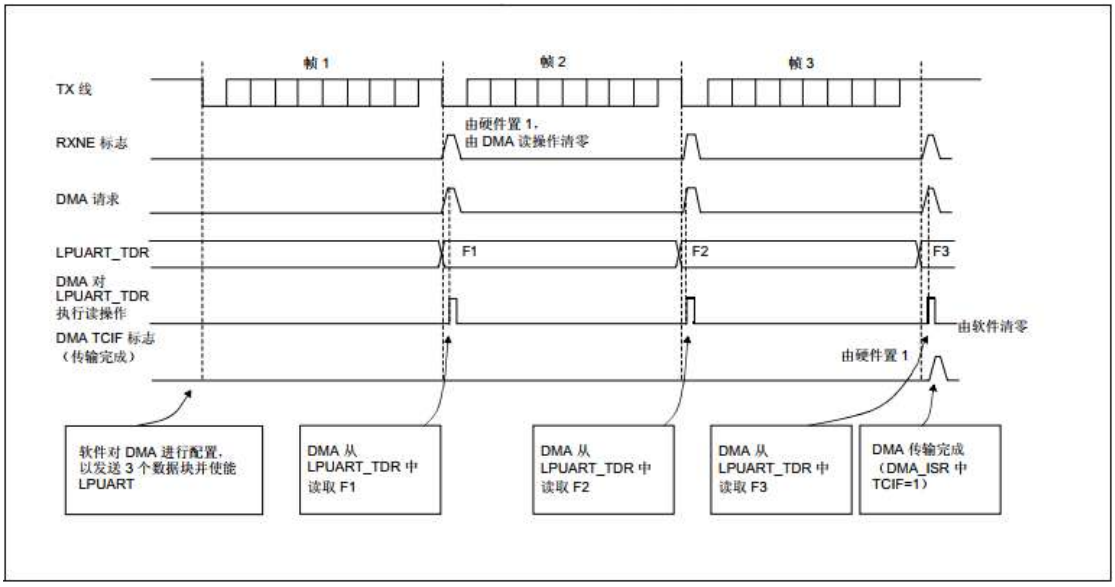
使用 DMA 进行接收

将 LPUART_CR3 寄存器中的 DMAR 位置 1 可以使能 DMA 模式进行接收。接收数据字节时，数据会从 LPUART_RDR 寄存器加载到通过 DMA 外设配置的 SRAM 区域中（请参见第 246 页上的第 11 节：直接存储器访问控制器 (DMA)）。要映射一个 DMA 通道以进行 LPUART 接收，请按以下步骤操作：

1. 在 DMA 控制寄存器中写入 LPUART_RDR 寄存器地址，将其配置为传输的源地址。每次发生 RXNE 事件后，数据都从此地址移动到存储器。
2. 在 DMA 控制寄存器中写入存储器地址，将其配置为传输的目标地址。每次发生 RXNE 事件后，数据都从 LPUART_RDR 寄存器加载到此存储区。
3. 在 DMA 控制寄存器中配置要传输的总字节数。
4. 在 DMA 控制寄存器中配置通道优先级。
5. 根据应用的需求，在完成一半或全部传输后产生中断。
6. 在 DMA 控制寄存器中激活该通道。

当达到在 DMA 控制器中设置的数据传输量时，DMA 控制器会在 DMA 通道的中断向量上产生一个中断。

图 27-8 使用 DMA 进行接收



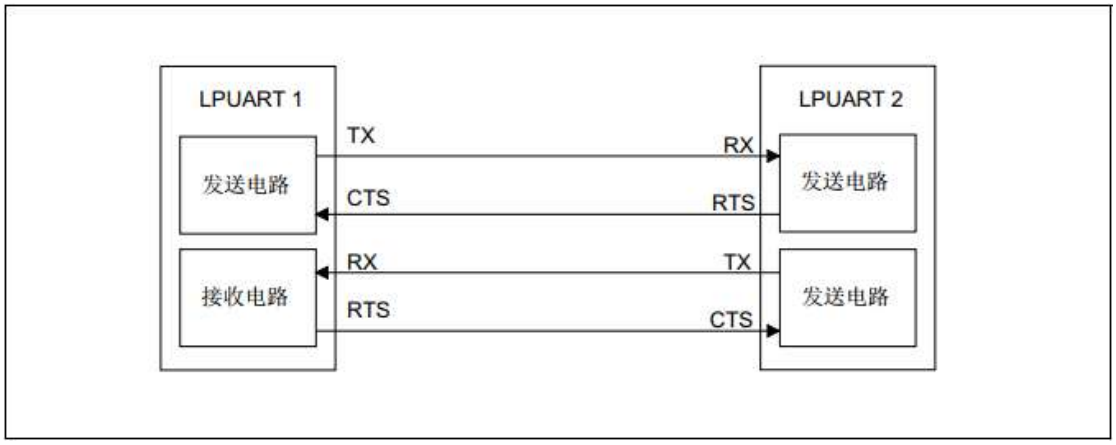
多缓冲区通信中的错误标志和中断生成

在多缓冲区通信中，如果事务中发生任何错误，都会在当前字节后放置错误标志。如果中断使能标志置 1，则会产生中断。在单字节接收过程中，与 RXNE 一同置位的帧错误、上溢错误和噪声标志具有单独的错误标志中断使能位（LPUART_CR3 寄存器中的 EIE 位）；如果该位置 1，在发生其中任何一个错误时，在当前字节后使能中断。

27.4.10 RS232 硬件流控制和 RS485 驱动器使用

使用 CTS 输入和 RTS 输出可以控制 2 个器件间的串行数据流。 图 27-9 2 个 LPUART 间的硬件流控制 显示了在这种模式下如何连接 2 个器件：

图 27-9 2 个 LPUART 间的硬件流控制

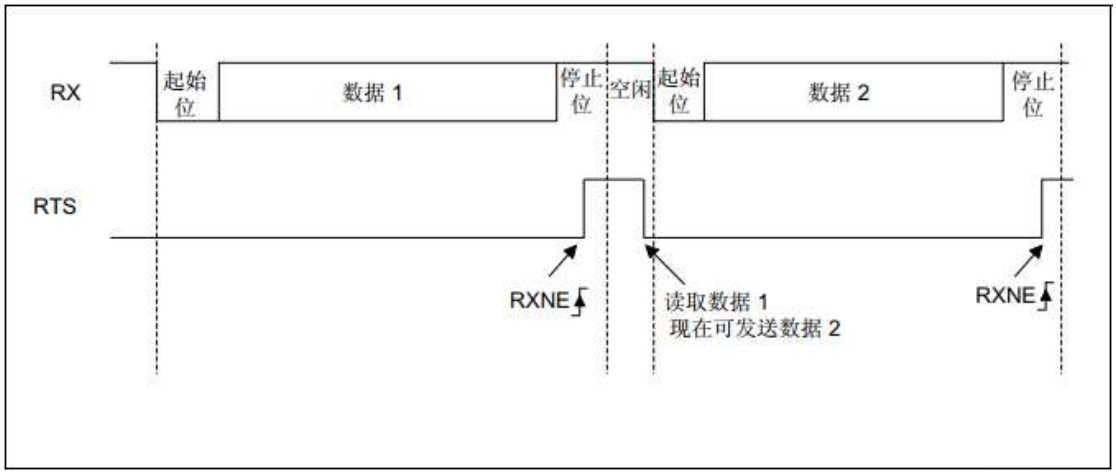


分别向 LPUART_CR3 寄存器中的 RTSE 位和 CTSE 位写入 1，可以分别使能 RS232 RTS 和 CTS 流控制。

RS232 RTS 流控制

如果使能 RTS 流控制 (RTSE=1)，只要 LPUART 接收器准备好接收新数据，便会将 RTS 变为有效（连接到低电平）。当接收寄存器已满时，会将 RTS 变为无效，表明发送过程会在当前帧结束后停止。 图 27-10 RS232 RTS 流控制显示了在使能 RTS 流控制的情况下进行通信的示例。

图 27-10 RS232 RTS 流控制



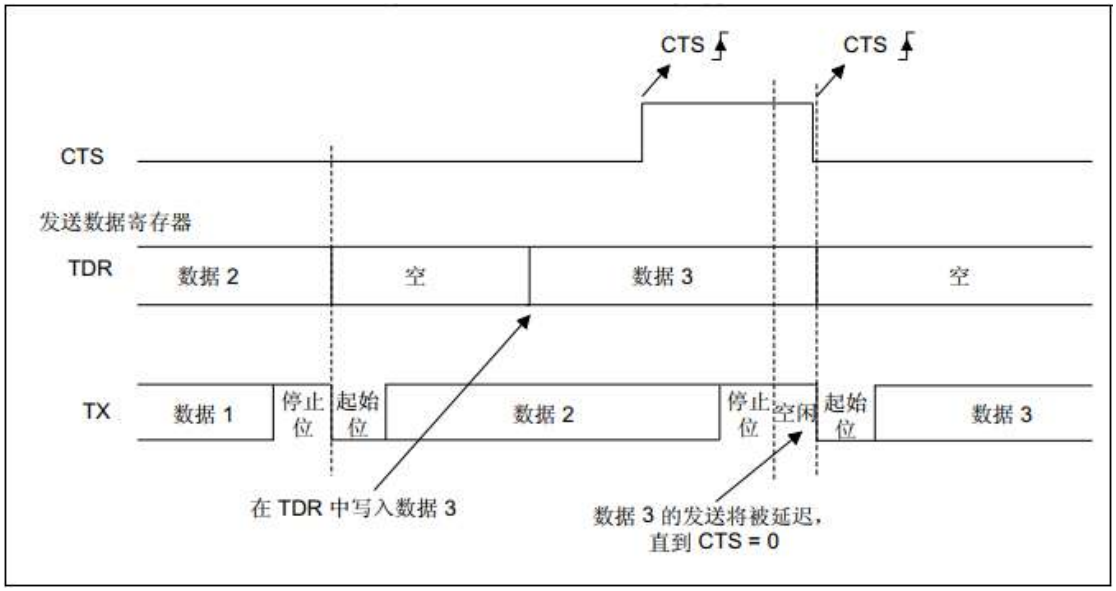
RS232 CTS 流控制

如果使能 CTS 流控制 (CTSE=1)，则发送器会在发送下一帧前检查 CTS。如果 CTS 有效（连接到低电平），则会发送下一数据（假设数据已准备好发送，即 TXE=0）；否则不会进行发送。如果在发送过程中 CTS 变为无效，则当前发送完成之后，发送器停止。

当 CTSE=1 时，只要 CTS 发生变化， CTSIF 状态位便会由硬件自动置 1。这指示接收器是否已准备好进行通信。如果 LPUART_CR3 寄存器中的 CTSIE 位置 1，则会产生中断。

图 27-11 RS232 CTS 流控制显示了在使能 CTS 流控制的情况下进行通信的示例。

图 27-11 RS232 CTS 流控制



注：为正常运行，必须在当前字符结束前至少 3 个 LPUART 时钟源周期内使能 CTS。此外还应注意，当脉冲短于 2 个 PCLK 周期时无法将 CTSCF 标志置 1。

RS485 驱动器使能

驱动器使能功能可通过将 LPUART_CR3 控制寄存器中的 DEM 位置 1 使能。这样用户便可通过 DE（驱动器使能）信号激活外部收发器控制。使能时间为激活 DE 信号与 START 位开始间的时间。可以通过 LPUART_CR1 控制寄存器中的 DEAT [4:0] 位域编程使能时间。禁止时间为发送的消息中最后一个停止位

结束与取消激活 DE 信号间的时间。可以通过 LPUART_CR1 控制寄存器中的 DEDT [4:0] 位域编程使能时间。 DE 信号的极性可使用 LPUART_CR3 控制寄存器中的 DEP 位配置。

在 LPUART 中， DEAT 和 DEDT 用 USART 时钟源 (fCK) 周期表示：

- 驱动器使能有效时间 =
  - （1+ DEAT）x fCK
- 驱动器使能禁止时间 =
  - （1+DEDT）x fCK

27.4.11 使用 LPUART 从停止模式唤醒

当 UESM 位置 1 且 LPUART 时钟设置为 HSI 或 LSE 时， LPUART 能够将 MCU 从停止模式唤醒（请参见复位和时钟控制 (RCC) 部分）。

可使用标准 RXNE 中断将 MCU 从停止模式唤醒。在这种情况下，必须在进入停止模式前将 RXNEIE 位置 1。也可通过 WUS 位域选择特定中断。

为了能够将 MCU 从停止模式唤醒，必须在进入停止模式前将 LPUART_CR1 控制寄存器中的 UESM 位置 1。检测到唤醒事件后，WUF 标志会由硬件置 1 并在 WUFIE 位置 1 时生成一个唤醒中断。

有关代码示例，请参见 A.18.1： LPUART 接收器配置代码示例和 A.18.2： LPUART 接收字节代码示例。

注： 在进入停止模式前，用户必须确保 LPUART 未在执行传输。 BUSY 标志无法确保运行接收期间始终不进入停止模式。

WUF 标志在检测到唤醒事件时置 1，而与 MCU 处于停止模式还是工作模式无关。

如果在初始化和使能接收器后立即进入停止模式，则必须校验 REACK 位以确保 LPUART 确实已使能。当 DMA 用于接收时，它必须在进入停止模式前禁止，并在退出停止模式后重新使能。

从停止模式唤醒功能并非在所有模式下均可用。例如，该功能在 SPI 模式下不起作用，因为 SPI 仅在主模式下工作

使用静默模式和停止模式

如果 LPUART 在进入停止模式前处于静默模式：

- 不得使用空闲检测时从静默模式唤醒，因为空闲检测无法在停止模式下工作。
- 如果使用地址匹配时从静默模式唤醒，则从停止模式唤醒的源也必须是地址匹配。如果 RXNE 标志在进入停止模式时置 1，则接口将在地址匹配时和从停止模式唤醒时保持静默模式。
- 如果 LPUART 配置为在 START 位检测时将 MCU 从停止模式唤醒， WUF 标志将置 1 但 RXNE 标志不置 1。

当 LPUART 时钟源为 HSI 时钟时，确定最大 LPUART 波特率可使从停止模式正确地唤醒

允许从停止模式正确唤醒的最大波特率取决于：

- 器件数据手册中提供的参数 tWULPUART（从停止模式唤醒）
- 第 5.21.4.5 节：LPUART 接收器对时钟偏差的容差中提供的 LPUART 接收器的容差。

举例来说：M 位 = 01 个、2 个停止位，BRR ≥ 4096。

在这些条件下，根据表 27-3 BRR[3:0] 不等于 0000 时的 LPUART 接收器容差， LPUART 接收器的容差为 4.42 %。

DTRA + DQUANT + DREC + DTCL + DWU < LPUART 接收器的容差

DWU max = $t_{WULPUART}$ / (11 x Tbit Min)

Tbit Min = $t_{WULPUART}$ / (11 x DWU max)

考虑一种理想情况：参数 DTRA、DQUANT、DREC 和 DTCL 为 0%，则 DWU 最大值为 4.42 %。实际上，我们至少需要考虑 HSI 不精确的情况。

假设 HSI 不精确度 = 1 %，则 $t_{WULPUART}$ = 8.1 µs（处于停止模式，主调压器处于运行模式，范围

1）：

DWU max = 4.42 % - 1 % = 3.42 %

Tbit min = 8.1 μs / (11 x 3.42 %) = 2.5 μs。

在这些情况下，可使从停止模式正确唤醒的最大波特率为 1/ 21.5 μs = 46 K 波特。

27.5 LPUART 低功耗模式

表 27-5 低功耗模式对 LPUART 的影响

模式	说明
睡眠	无影响。USART 中断可使器件退出睡眠模式。
低功耗运行	无影响。
低功耗睡眠	无影响。USART 中断可使器件退出低功耗睡眠模式。
停止	当 UESM 位置 1 且 LPUART 时钟设置为 HSI16 或 LSE 时，LPUART 能够将 MCU 从停止模式唤醒。 可使用标准 RXNE 或 WUF 中断将 MCU 从停止模式唤醒。
待机	LPUART 掉电，当器件退出待机模式后必须重新初始化 USART。

27.6 LPUART 中断

表 27-6 LPUART 中断请求

中断事件	事件标志	使能控制位
发送数据寄存器为空	TXE	TXEIE
CTS 中断	CTSIF	CTSIE
发送完成	TC	TCIE
接收数据寄存器不为空（已准备好读取数据）	RXNE	RXNEIE
检测到溢出错误	ORE	
检测到空闲线路	IDLE	IDLEIE
奇偶校验错误	PE	PEIE
多缓冲区通信中的噪声标志、溢出错误和帧错误	NF 或 ORE 或 FE	EIE
字符匹配	CMF	CMIE
从停止模式唤醒	WUF(1)	WUFIE

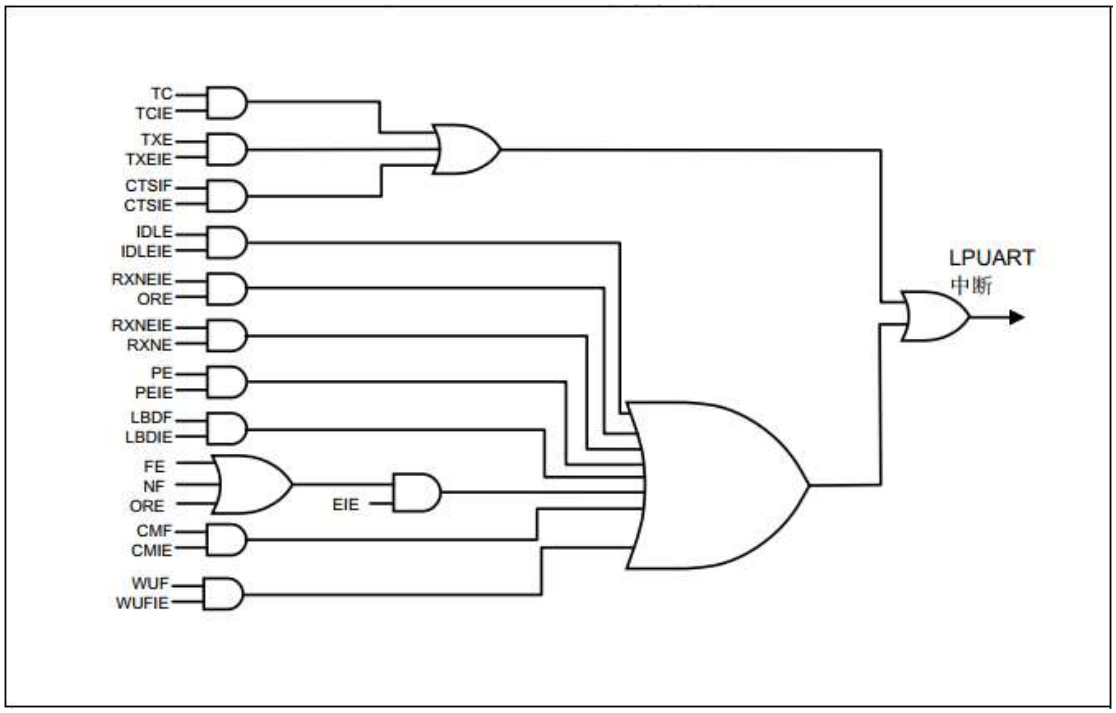
1. WUF 中断仅在停止模式下有效。

LPUART 中断事件被连接到相同的中断向量（请参见图 27-12 LPUART 中断映射图）。

- 发送期间：发送完成、清除以发送、发送数据寄存器为空或帧错误中断。
- 接收期间：空闲线路检测、上溢错误、接收数据寄存器不为空、奇偶校验错误、噪声标志、帧错误、字符匹配等。

如果相应的使能控制位置 1，则这些事件会生成中断。

图 27-12 LPUART 中断映射图



27.7 LPUART1 寄存器

27.7.1 控制寄存器 1 (LPUART_CR1)

偏移地址：0x00

复位值：0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	M1	Res.	Res.	DEAT[4:0]					DEDT[4:0]				
			rw			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
			0			0	0	0	0	0	0	0	0	0	0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	CMIE	MME	M0	WAKE	PCE	PS	PEIE	TXEIE	TCIE	RXNEIE	IDLEIE	TE	RE	UESM	UE
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位 31:29	保留，必须保持复位值
位 28	**M1**：字长 (Word length) 此位和位 12 (M0) 用于确定字长度。此位由软件置 1 或清零。 M[1:0] = 00：1 个起始位，8 个数据位，n 个停止位 M[1:0] = 01：1 个起始位，9 个数据位，n 个停止位 M[1:0] = 10：1 个起始位，7 个数据位，n 个停止位 只有在禁止 LPUART (UE=0) 时才能写入此位。
位 27	保留，必须保持复位值
位 26	保留，必须保持复位值

位 25:21	**DEAT[4:0]**：驱动器使能使能时间 (Driver Enable assertion time) 该 5 位值用于定义激活 DE（启动器使能）信号与起始位开始间的时间。以 UCLK（USART 时钟）时钟周期数表示。有关更多详细信息，请参见 RS485 驱动器使能段落。 只有在禁止 LPUART (UE=0) 后才能写入此位域。
位 20:16	**DEDT[4:0]**：驱动器使能禁止时间 (Driver Enable de-assertion time) 该 5 位值用于定义发送的消息中最后一个停止位结束与取消激活 DE（驱动器使能）信号间的时间。以 UCLK（USART 时钟）时钟周期数表示。有关更多详细信息，请参见 RS485 驱动器使能段落。如果在 DEDT 时间内对 LPUART_TDR 寄存器执行写操作，则新数据仅在经过 DEDT 和 DEAT 时间后才会发送。只有在禁止 LPUART (UE=0) 后才能写入此位域。
位 15	保留，必须保持复位值
位 14	**CMIE**：字符匹配中断使能 (Character match interrupt enable) 此位由软件置 1 和清零。 0：禁止中断 1：如果 LPUART_ISR 寄存器中的 CMF 位置 1，则生成 LPUART 中断。
位 13	**MME**：静默模式使能 (Mute mode enable) 此位用于激活 LPUART 的静默模式功能。此位置 1 时，LPUART 可按 WAKE 位定义的方式在活动模式与静默模式之间切换。此位由软件置 1 和清零。 0：接收器永久处于活动模式 1：接收器可在静默模式和活动模式之间切换。
位 12	**M0**：字长 (Word length) 此位和位 28 (M1) 用于确定字长度。此位由软件置 1 或清零。请参见位 28 (M1) 的说明。只有在禁止 LPUART (UE=0) 时才能写入此位。
位 11	**WAKE**：接收器唤醒方法 (Receiver wakeup method) 此位用于确定 LPUART 静默模式的唤醒方法。此位由软件置 1 或清零。 0：空闲线路 1：地址标记 只有在禁止 LPUART (UE=0) 后才能写入此位域。
位 10	**PCE**：奇偶校验控制使能 (Parity control enable) 该位选择硬件奇偶校验控制（生成和检测）。使能奇偶校验控制时，计算出的奇偶校验位被插入到 MSB 位置（如果 M=1，则为第 9 位；如果 M=0，则为第 8 位），并对接收到的数据检查奇偶校验位。此位由软件置 1 和清零。一旦该位置 1，PCE 在当前字节的后面处于活动状态（在接收和发送时）。 0：禁止奇偶校验控制 1：使能奇偶校验控制 只有在禁止 LPUART (UE=0) 后才能写入此位域。
位 9	**PS**：奇偶校验选择 (Parity selection) 该位用于在使能奇偶校验生成/检测（PCE 位置 1）时选择奇校验或偶校验。此位由软件置 1 和清零。将在当前字节的后面选择奇偶校验。 0：偶校验 1：奇校验 只有在禁止 LPUART (UE=0) 后才能写入此位域。
位 8	**PEIE**：PE 中断使能 (PE interrupt enable) 此位由软件置 1 和清零。 0：禁止中断 1：当 LPUART_ISR 寄存器中的 PE=1 时，生成 LPUART 中断
位 7	**TXEIE**：中断使能 (interrupt enable) 此位由软件置 1 和清零。 0：禁止中断 1：当 LPUART_ISR 寄存器中的 TXE=1 时，生成 LPUART 中断
位 6	**TCIE**：传输完成中断使能 (Transfer complete interrupt enable) 此位由软件置 1 和清零。 0：禁止中断 1：当 LPUART_ISR 寄存器中的 TC=1 时，生成 LPUART 中断

位 5	**RXNEIE**：RXNE 中断使能 (RXNE interrupt enable) 此位由软件置 1 和清零。 0：禁止中断 1：当 LPUART_ISR 寄存器中的 ORE=1 或 RXNE=1 时，生成 LPUART 中断
位 4	**IDLEIE**：IDLE 中断使能 (IDLE interrupt enable) 此位由软件置 1 和清零。 0：禁止中断 1：当 LPUART_ISR 寄存器中的 IDLE=1 时，生成 LPUART 中断
位 3	**TE**：发送器使能 (Transmitter enable) 该位使能发送器。此位由软件置 1 和清零。 0：禁止发送器 1：使能发送器 注： 传送期间 TE 位上的“0”脉冲（“0”后紧跟的是“1”）会在当前字的后面发送一个报头（空闲线路）。为生成空闲字符，TE 不能立即写入 1。为确保所需的持续时间，软件可轮询 LPUART_ISR 寄存器中的 TEACK 位。当 TE 置 1 时，在发送开始前存在 1 位的时间延迟。
位 2	**RE**：接收器使能 (Receiver enable) 该位使能接收器。此位由软件置 1 和清零。 0：禁止接收器 1：使能接收器并开始搜索起始位
位 1	**UESM**：停止模式下的 LPUART 使能 (LPUART enable in Stop mode) 当此位清零时，LPUART 无法将 MCU 从停止模式唤醒。 当此位置 1 时，LPUART 能够将 MCU 从停止模式唤醒，前提是 LPUART 时钟选择为 HSI 或 LSE（在 RCC 中）。 此位由软件置 1 和清零。 0：LPUART 无法将 MCU 从停止模式唤醒。 1：LPUART 能够将 MCU 从停止模式唤醒。当该功能激活时，LPUART 的时钟源必须是 HSI 或 LSE（请参见复位和时钟控制 (RCC) 部分）。 注： 建议在进入停止模式前将 UESM 位置 1，并在退出停止模式时将其清零。
位 0	**UE**：LPUART 使能 (CTS enable) 此位清零后，LPUART 预分频器和输出将立即停止，并丢弃当前操作。LPUART 的配置保留，但 LPUART_ISR 中的所有状态标志将被复位。此位由软件置 1 和清零。 0：禁止 LPUART 预分频器和输出，低功耗模式 1：使能 LPUART 注： 为进入低功耗模式而不在线路上生成错误，之前必须复位 TE 位，并且软件必须等待 LPUART_ISR 寄存器中的 TC 位置 1 后才能复位 UE 位。UE = 0 时也会复位 DMA 请求，因必须在复位 UE 位前禁止 DMA 通道。

27.7.2 控制寄存器 2 (LPUART_CR2)

偏移地址：0x04

复位值：0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADD[7:4]				ADD[3:0]				Res.	Res.	Res.	Res.	MSBI RST	DATAINV	TXINV	RXIN V
rw	rw	rw	rw	rw	rw	rw	rw					rw	rw	rw	rw
0	0	0	0	0	0	0	0					0	0	0	0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWAP	Res.	STOP[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADDM7	Res.	Res.	Res.	Res.
rw		rw	rw								rw				
0		0	0								0				

位 31:28	**ADD[7:4]**：LPUART 节点的地址 (Address of the USART node) 此位域用于指定 LPUART 节点的地址或要识别的字符代码。 此位域在多处理器通信时于静默模式或停止模式下使用，以通过 7 位地址标记检测进行唤醒。发送器发送字符的 MSB 应为 1。此位域还可用于正常接收和静默模式无效时的字符检测（例如，Modbus 协议中的块结束检测）。这种情况下，接收到的整个字符（8 位）将与 ADD[7:0]值进行比较，如果匹配，CMF 标志将置 1。仅在禁止接收 (RE = 0) 或禁止 LPUART (UE=0) 时才能写入该位域
位 27:24	**ADD[3:0]**：LPUART 节点的地址 (Address of the USART node) 此位域用于指定 LPUART 节点的地址或要识别的字符代码。 此位域在多处理器通信时于静默模式或停止模式下使用，以通过地址标记检测进行唤醒。 仅在禁止接收 (RE = 0) 或禁止 LPUART (UE=0) 时才能写入该位域
位 23:20	保留，必须保持复位值
位 19	**MSBFIRST**：最高有效位在前 (Most significant bit first) 此位由软件置 1 和清零。 0：发送/接收数据时位 0 在前，后跟起始位。 1：发送/接收数据时 MSB（位 7/8/9）在前，后跟起始位。 只有在禁止 LPUART (UE=0) 后才能写入此位域。
位 18	**DATAINV**：二进制数据反向 (Binary data inversion) 此位由软件置 1 和清零。 0：按正/正向逻辑发送/接收数据寄存器中的逻辑数据。（1=H，0=L） 1：按负/反向逻辑发送/接收数据寄存器中的逻辑数据。（1=L，0=H）。奇偶校验位也取反。 只有在禁止 LPUART (UE=0) 后才能写入此位域。
位 17	**TXINV**：TX 引脚有效电平反向 (TX pin active level inversion) 此位由软件置 1 和清零。 0：TX 引脚信号使用标准逻辑电平（VDD = 1/空闲，Gnd = 0/标记）工作。 1：对 TX 引脚信号值取反。（VDD = 0/标记，Gnd=1/空闲）。 允许在 TX 线路上使用外部反相器。 只有在禁止 LPUART (UE=0) 后才能写入此位域。
位 16	**RXINV**：RX 引脚有效电平反向 (RX pin active level inversion) 此位由软件置 1 和清零。 0：RX 引脚信号使用标准逻辑电平（VDD = 1/空闲，Gnd = 0/标记）工作。 1：对 RX 引脚信号值取反。（VDD = 0/标记，Gnd=1/空闲）。 允许在 RX 线路上使用外部反相器。 只有在禁止 LPUART (UE=0) 后才能写入此位域。
位 15	**SWAP**：交换 TX/RX 引脚 (Swap TX/RX pins) 此位由软件置 1 和清零。 0：按标准引脚排列定义使用 TX/RX 引脚 1：交换 TX 和 RX 引脚功能。允许在与另一个 UART 的交叉连接时工作。 只有在禁止 LPUART (UE=0) 后才能写入此位域。
位 14	保留，必须保持复位值
位 13:12	**STOP[1:0]**：停止位 (STOP bits) 这些位用于编程停止位。 00：1 个停止位 01：保留 10：2 个停止位 11：保留 只有在禁止 LPUART (UE=0) 后才能写入此位域。
位 11:5	保留，必须保持复位值
位 4	**ADDM7**：7 位地址检测/4 位地址检测 (7-bit Address Detection/4-bit Address Detection) 此位用于选择 4 位地址检测或 7 位地址检测。 0：4 位地址检测 1：7 位地址检测（在 8 位数据模式下） 只有在禁止 LPUART (UE=0)时才能写入该位 注：在 7 位和 9 位数据模式下，地址检测分别在 6 位和 8 位地址上完成（ADD[5:0]和 ADD[7:0]）。
位 3:0	保留，必须保持复位值。

27.7.3 控制寄存器 3 (LPUART_CR3)

偏移地址：0x08

复位值：0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WUFIE	WUS[2:0]		Res.	Res.	Res.	Res.
									rw	rw	rw				
									0	0	0				

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DEP	DEM	DDRE	OVRDIS	Res.	CTSIE	CTSE	RTSE	DMAT	DMAR	Res.	Res.	HDSEL	Res.	Res.	EIE
rw	rw	rw	rw		rw	rw	rw	rw	rw			rw			rw
0	0	0	0		0	0	0	0	0			0			0

位 31:23	保留，必须保持复位值。
位 22	**WUFIE**：从停止模式唤醒中断使能 (Wakeup from Stop mode interrupt enable) 此位由软件置 1 和清零。 0：禁止中断 1：当 LPUART_ISR 寄存器中的 WUF=1 时，生成 LPUART 中断 注： WUFIE 必须在进入停止模式前置 1。WUF 中断仅在停止模式下有效。如果 LPUART 不支持从停止模式唤醒功能，则此位保留并由硬件强制清零。
位 21:20	**WUS[1:0]**：从停止模式唤醒中断标志选择 (Wakeup from Stop mode interrupt flag selection) 该位域用于指定激活 WUF（从停止模式唤醒标志）的事件。 00：WUF 在地址匹配时激活（按 ADD[7:0] 和 ADDM7 所定义） 01：保留。 10：WUF 在起始位检测时激活 11：WUF 在 RXNE 时激活。 只有在禁止 LPUART (UE=0) 后才能写入此位域。 注： 如果 LPUART 不支持从停止模式唤醒功能，则此位保留并由硬件强制清零。
位 19:16	保留，必须保持复位值。
位 15	**DEP**：驱动器使能极性选择 (Driver enable polarity selection) 0：DE 信号高电平有效。 1：DE 信号低电平有效。 只有在禁止 LPUART (UE=0) 时才能写入此位。
位 14	**DEM**：驱动器使能模式 (Driver enable mode) 此位用于通过 DE 信号激活外部收发器控制。 0：禁止 DE 功能。 1：使能 DE 功能。DE 信号在 RTS 引脚上输出。 只有在禁止 LPUART (UE=0) 时才能写入此位。
位 13	**DDRE**：接收出错时的 DMA 禁止 (DMA Disable on Reception Error) 0：接收出错时不禁止 DMA。相应的错误标志置 1，但 RXNE 保持为 0 以防止上溢。因此，将不使能 DMA 请求，从而不会传送错误数据（无 DMA 请求），但会传送接收到的下一个正确数据。 1：接收出错后禁止 DMA。相应的错误标志以及 RXNE 均置 1。屏蔽 DMA 请求，直到错误标志清零。这意味着软件必须首先禁止 DMA 请求 (DMAR = 0) 或者将 RXNE 清零，然后再将错误标志清零。只有在禁止 LPUART (UE=0) 时才能写入此位。 注： 接收错误包括：奇偶校验错误、帧错误或噪声错误。

位 12	**OVRDIS**：上溢禁止 (Overrun Disable) 此位用于禁止接收上溢检测。 0：接收新数据前未读取已接收的数据时，上溢错误标志 ORE 置 1。 1：禁止上溢功能。如果在 RXNE 标志仍置 1 时接收到新数据，则 ORE 标志不会置 1，且新接收的数据会覆盖 LPUART_RDR 寄存器之前的内容。只有在禁止 LPUART (UE=0) 时才能写入此位。 注：此控制位用于检查通信流而不会读取数据。
位 11	保留，必须保持复位值。
位 10	**CTSIE**：CTS 中断使能 (CTS interrupt enable) 0：禁止中断 1：当 LPUART_ISR 寄存器中 CTSIF = 1 时，生成中断
位 9	**CTSE**：CTS 使能 (CTS enable) 0：禁止 CTS 硬件流控制 1：使能 CTS 模式，仅当 CTS 输入有效（连接到 0）时才发送数据。如果在发送数据时使 CTS 输入无效，会在停止之前完成发送。如果使 CTS 无效时数据已写入数据寄存器，则将延迟发送，直到 CTS 有效。 只有在禁止 LPUART (UE=0) 时才能写入该位
位 8	**RTSE**：RTS 使能 (RTS enable) 0：禁止 RTS 硬件流控制 1：使能 RTS 输出，仅当接收缓冲区中有空间时才会请求数据。发送完当前字符后应停止发送数据。可以接收数据时使 RTS 输出有效（拉至 0）。 只有在禁止 LPUART (UE=0) 时才能写入此位。
位 7	**DMAT**：DMA 使能发送器 (DMA enable transmitter) 此位由软件置 1/复位。 1：针对发送使能 DMA 模式 0：针对发送禁止 DMA 模式
位 6	**DMAR**：DMA 使能接收器 (DMA enable receiver) 此位由软件置 1/复位。 1：针对接收使能 DMA 模式 0：针对接收禁止 DMA 模式
位 5:4	保留，必须保持复位值。
位 3	**HDSEL**：半双工选择 (Half-duplex selection) 选择单线半双工模式 0：未选择半双工模式 1：选择半双工模式 只有在禁止 LPUART (UE=0) 时才能写入此位。
位 2:1	保留，必须保持复位值。
位 0	**EIE**：错误中断使能 (Error interrupt enable) 如果发生帧错误、上溢错误或出现噪声标志（LPUART_ISR 寄存器中 FE = 1 或 ORE = 1 或 NF = 1），则需要使用错误中断使能位来使能中断生成。 0：禁止中断 LPUART_ISR 寄存器中的 FE=1 或 ORE=1 或 NF=1 时生成中断。

27.7.4 波特率寄存器 (LPUART_BRR)

只有在禁止 LPUART (UE=0) 时才能写入此寄存器。

偏移地址：0x0C

复位值：0x0000

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

rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位 31:20	保留，必须保持复位值。
位 19:0	**BRR[19:0]** 注： 禁止在 LPUART_BRR 寄存器中写入小于 0x300 的值。如果 LPUARTx_BRR 必须> = 0x300 且 LPUART_BRR 为 20 位，则使用高 fck 值生成高波特率时应十分谨慎。fck 必须在[3 x 波特率，4096 x 波特率] 范围内。

27.7.5 请求寄存器 (LPUART_RQR)

偏移地址：0x18

复位值：0x0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXFRQ	MMRQ	SBKRQ	Res.
												w	w	w	
												0	0	0	

位 31:4	保留，必须保持复位值
位 3	**RXFRQ**：接收数据刷新请求 (Receive data flush request) 向该位写入 1 时会将 RXNE 标志清零。 这可以丢弃数据而不对其执行读取操作，并避免发生上溢。
位 2	**MMRQ**：静默模式请求 (Mute mode request) 向此位写入 1 可将 LPUART 置于静默模式，并将 RWU 标志复位。
位 1	**SBKRQ**：发送断路请求 (Send break request) 向此位写入 1 可将 SBKF 标志置 1 并在发送设备可用后立即请求在线路上发送 BREAK。 注： 这种情况下，应用需要在之前插入的所有数据（包括尚未发送的数据）后发送中断字符，软件应等到 TXE 标志使能后将 SBKRQ 位置 1。
位 0	保留，必须保持复位值

27.7.6 中断和状态寄存器 (LPUART_ISR)

偏移地址：0x1C

复位值：0x00C0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REACK	TEACK	WUF	RWU	SBKF	CMF	BUSY
									r	r	r	r	r	r	r
									0	0	0	0	0	0	0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	CTS	CTSIF	Res.	TXE	TC	RXNE	IDLE	ORE	NF	FE	PE
					r	r		r	r	r	r	r	r	r	r
					0	0		1	1	0	0	0	0	0	0

位 31:23	保留，必须保持复位值。
位 22	**REACK**：接收使能确认标志 (Receive enable acknowledge flag) LPUART 采用接收使能值时，通过硬件将此位置 1/复位。 此位可用于验证 LPUART 是否准备好在进入停止模式前接收数据。 注： 如果 LPUART 不支持从停止模式唤醒功能，则此位保留并由硬件强制清零。
位 21	**TEACK**：发送使能确认标志 (Transmit enable acknowledge flag) LPUART 采用发送使能值时，通过硬件将此位置 1/复位。 通过写入 TE=0 生成空闲帧请求，然后在 LPUART_CR1 寄存器中写入 TE=1 以遵循 TE=0 最短周期时，可使用此位。
位 20	**WUF**：从停止模式唤醒标志 (Wakeup from Stop mode flag 当检测到唤醒事件时，此位由硬件置 1。事件通过 WUS 位域定义。通过向 LPUART_ICR 寄存器中的 WUCF 写入 1，此位由软件清零。 如果 LPUART_CR3 寄存器中 WUFIE=1，则会生成中断。 注：当 UESM 清零时，WUF 标志也清零。WUF 中断仅在停止模式下有效。如果 LPUART 不支持从停止模式唤醒功能，则此位保留并由硬件强制清零。
位 19	**RWU**：接收器从静默模式唤醒(Receiver wakeup from Mute mode) 此位指示 LPUART 是否处于静默模式。当识别出唤醒/静默序列时，此位由硬件清零/置 1。 静默模式控制序列（地址或 IDLE）通过 LPUART_CR1 寄存器中的 WAKE 位选择。 当选择 IDLE 模式下唤醒时，该位只能通过用软件向 LPUART_RQR 寄存器中的 MMRQ 位写 1 的方式置 1。 0：接收器处于活动模式 1：接收器处于静默模式 注： 如果 LPUART 不支持从停止模式唤醒功能，则此位保留并由硬件强制清零。
位 18	**SBKF**：发送断路标志 (Send break flag) 此位指示已请求发送断路字符。通过将 1 写入 LPUART_CR3 寄存器中的 SBKRQ 位，此位由软件置 1。此位在断路发送的停止位期间由硬件自动复位。 0：不发送断路字符 1：将发送断路字符
位 17	**CMF**：字符匹配标志 (Character match flag) 接收到由 ADD[7:0] 定义的字符后由硬件将此位置 1。通过向 LPUART_ICR 寄存器中的 CMCF 写入 1，此位由软件清零。 如果 LPUART_CR1 寄存器中 CMIE=1，则会生成中断。 0：未检测到字符匹配 1：检测到字符匹配
位 16	**BUSY**：忙标志 (Busy flag) 此位由硬件置 1 和复位。当 RX 线路上正在进行通信（成功检测到起始位）时有效。在接收结束（成功或失败）时复位。 0：LPUART 处于空闲状态（无接收） 1：正在接收
位 15:11	保留，必须保持复位值。
位 10	**CTS**：CTS 标志 (CTS flag) 此位由硬件置 1/复位。此位是对 CTS 输入引脚的状态取反。 0：CTS 线置 1 1：CTS 线复位 注： 如果不支持硬件流控制功能，该位保留并由硬件强制清零。
位 9	**CTSIF**：CTS 中断标志 (CTS interrupt flag) 如果 CTSE 位置 1，当 CTS 输入切换时，此位由硬件置 1。通过将 1 写入 LPUART_ICR 寄存器中的 CTSCF 位，此位由软件清零。 如果 LPUART_CR3 寄存器中 CTSIE=1，则会生成中断。 0：CTS 状态线上未发生变化 1：CTS 状态线上发生变化 注： 如果不支持硬件流控制功能，该位保留并由硬件强制清零。
位 8	保留，必须保持复位值。

位 7	**TXE**：发送数据寄存器为空 (Transmit data register empty) 当 LPUART_TDR 寄存器的内容已传输到移位寄存器时，此位由硬件置 1。通过对 LPUART_TDR 寄存器执行写入操作将该位清零。 如果 LPUART_CR1 寄存器中 TXEIE 位 = 1，则会生成中断。 0：数据未传输到移位寄存器 1：数据传输到移位寄存器 注： 单缓冲区发送期间使用该位。
位 6	**TC**：发送完成 (Transmission complete) 如果已完成对包含数据的帧的发送并且 TXE 置 1，则此位由硬件置 1。如果 LPUART_CR1 寄存器中 TCIE = 1，则会生成中断。通过向 LPUART_ICR 寄存器中的 TCCF 写入 1 或向 LPUART_TDR 寄存器执行写操作，此位由软件清零。 如果 LPUART_CR1 寄存器中 TCIE = 1，则会生成中断。 0：传送未完成 1：传送已完成 注： 如果 TE 位复位且无任何发送正在进行，TC 位会立即置 1。
位 5	**RXNE**：读取数据寄存器不为空 (Read data register not empty) 当 RDR 移位寄存器的内容已传输到 LPUART_RDR 寄存器时，此位由硬件置 1。通过对 LPUART_RDR 寄存器执行读入操作将该位清零。也可以通过将 LPUART_RQR 寄存器中的 RXFRQ 位置 1 将 RXNE 标志位清零。 如果 LPUART_CR1 寄存器中 RXNEIE = 1，则会生成中断。 0：未接收到数据 1：已准备好读取接收到的数据
位 4	**IDLE**：检测到空闲线路 (IDLE line detected) 检测到空闲线路时，此位由硬件置 1。如果 LPUART_CR1 寄存器中 IDLEIE=1，则会生成中断。通过向 LPUART_ICR 寄存器中的 IDLECF 写入 1，此位由软件清零。 0：未检测到空闲线路 1：检测到空闲线路 注： 直到 RXNE 位已置 1 时（即，当出现新的空闲线路时）IDLE 位才会被再次置 1。使能静默模式(MME=1) 后，如果 LPUART 未静默(RWU=0)，则 IDLE 置 1，无论是否通过 WAKE 位选择了静默模式。如果 RWU=1，IDLE 不置 1。
位 3	**ORE**：溢出错误 (Overrun error) 在 RXNE = 1 的情况下，当移位寄存器中当前正在接收的数据准备好传输到 RDR 寄存器时，此位由硬件置 1。通过向 LPUART_ICR 寄存器中的 ORECF 写入 1，此位由软件清零。 如果 LPUART_CR1 寄存器中 RXNEIE=1 或 EIE = 1，则会生成中断。 0：无溢出错误 1：检测到溢出错误 注： 当此位置 1 时，RDR 寄存器的内容不会丢失，但移位寄存器会被覆盖。EIE 位置 1 后，如果在多缓冲区通信中 ORE 标志置 1，则会生成中断。LPUART_CR3 寄存器中的 OVRDIS 位置 1 时，此位将被永久强制清零（无上溢检测）。
位 2	**NF**：START 位噪声检测标志 (START bit Noise detection flag) 当在接收的帧的 START 位上检测到噪声时，此位由硬件置 1。通过向 LPUART_ICR 寄存器中的 NFCF 写入 1，此位由软件清零。 0：未检测到噪声 1：检测到噪声 注： 该位不会生成中断，因为该位出现的时间与本身生成中断的 RXNE 位出现的时间相同。EIE 位置 1 后，如果在多缓冲区通信中 NF 标志置 1，则会生成中断。
位 1	**FE**：帧错误 (Framing error) 当检测到去同步化、过度的噪声或中断字符时，此位由硬件置 1。通过向 LPUART_ICR 寄存器中的 FECF 写入 1，此位由软件清零。 如果 LPUART_CR1 寄存器中 EIE = 1，则会生成中断。 0：未检测到帧错误 1：检测到帧错误或中断字符

位 0	**PE**：奇偶校验错误 (Parity error) 当在接收器模式下发生奇偶校验错误时，此位由硬件置 1。通过向 LPUART_ICR 寄存器中的 PECF 写入 1，此位由软件清零。 如果 LPUART_CR1 寄存器中 PEIE = 1，则会生成中断。 0：无奇偶校验错误 1：奇偶校验错误

27.7.7 中断标志清零寄存器 (LPUART_ICR)

偏移地址：0x20

复位值：0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WUCF	Res.	Res.	CMCF	Res.
											w			w	
											0			0	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	CTSCF	Res.	Res.	TCCF	Res.	IDLECF	ORECF	NCF	FECF	PECF
						w			w		w	w	w	w	w
						0			0		0	0	0	0	0

位 31:21	保留，必须保持复位值。
位 20	**WUCF**：从停止模式唤醒清零标志 (Wakeup from Stop mode clear flag) 将 1 写入此位时，LPUART_ISR 寄存器中 WUF 标志将清零。 注： 如果 LPUART 不支持从停止模式唤醒功能，则此位保留并由硬件强制清零。
位 19:18	保留，必须保持复位值。
位 17	**CMCF**：字符匹配清零标志 (Character match clear flag) 将 1 写入此位时，LPUART_ISR 寄存器中 CMF 标志将清零。
位 16:10	保留，必须保持复位值。
位 9	**CTSCF**：CTS 清零标志 (CTS clear flag) 将 1 写入此位时，LPUART_ISR 寄存器中 CTSIF 标志将清零。
位 8:7	保留，必须保持复位值。
位 6	**TCCF**：发送完成清零标志 (Transmission complete clear flag) 将 1 写入此位时，LPUART_ISR 寄存器中 TC 标志将清零。
位 5	保留，必须保持复位值。
位 4	**IDLECF**：检测到空闲线路清零标志 (Idle line detected clear flag) 将 1 写入此位时，LPUART_ISR 寄存器中 IDLE 标志将清零。
位 3	**ORECF**：上溢错误清零标志 (Overrun error clear flag) 将 1 写入此位时，LPUART_ISR 寄存器中 ORE 标志将清零。
位 2	**NCF**：检测到噪声清零标志 (Noise detected clear flag) 将 1 写入此位时，LPUART_ISR 寄存器中 NF 标志将清零。
位 1	**FECF**：帧错误清零标志 (Framing error clear flag) 将 1 写入此位时，LPUART_ISR 寄存器中 FE 标志将清零。
位 0	**PECF**：奇偶校验错误清零标志 (Parity error clear flag) 将 1 写入此位时，LPUART_ISR 寄存器中 PE 标志将清零。

27.7.8 接收数据寄存器 (LPUART_RDR)

偏移地址：0x24

复位值：未定义

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

Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

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

位 31:9	保留，必须保持复位值。
位 8:0	**RDR[8:0]**：接收数据值 (Receive data value) 包含接收到的数据字符。 RDR 寄存器在输入移位寄存器和内部总线之间提供了并行接口在使能奇偶校验位的情况下进行接收时，从 MSB 位中读取的值为接收到的奇偶校验位。

27.7.9 发送数据寄存器 (LPUART_TDR)

偏移地址：0x28

复位值：未定义

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

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

位 31:9	保留，必须保持复位值。
位 8:0	**TDR[8:0]**：发送数据值 (Transmit data value) 包含要发送的数据字符。 TDR 寄存器在内部总线和输出移位寄存器之间提供了并行接口 在使能奇偶校验位的情况下（LPUART_CR1 寄存器中的 PCE 位被置 1）进行发送时，由于 MSB 的写入值（位 7 或位 8，具体取决于数据长度）会被奇偶校验位所取代，因此该值不起任何作用。 注： 只能在 TXE=1 时写入此寄存器。

28 蜂鸣器 (BEEPER)

28.1 简介

Beep 蜂鸣器内置超低功耗 7-bit 定时器，工作时钟可配置为 1~48MHz 外部高速时钟或 128kHz 片内 LSI 慢速时钟；定时器使用 down-count 的方式计数，可输出 1、2、4、8kHz 频率脉冲。

在 MCU 停机（Stop）模式下，Beep 可继续工作并可定时触发 ADC 采样；定时触发 ADC 采样的频率为 Beeper 蜂鸣输出脉冲频率的 1024 分之 1。例如 Beep 当前输出的蜂鸣脉冲为 1kHz，那么定时触发 ADC 采样的频率为 1kHz/1024 ≈ 0.98Hz（周期约为 1.02 秒）。

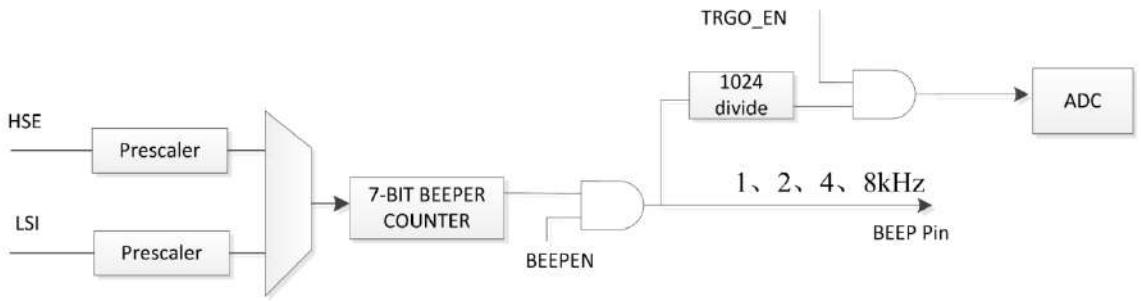
28.2 BEEP 主要特性

- 超低功耗 7 位向下计数器
- 时钟源可选
- 1、2、4、8kHz 频率脉冲输出
- 停止模式下，定时触发 ADC 采样

28.3 BEEP 功能说明

28.3.1 BEEP 框图

图 28-1BEEP 框图



28.3.2 TRGO

在 MCU 停机（Stop）模式下，Beep 可继续工作并可定时触发 ADC 采样；定时触发 ADC 采样的频率为 Beeper 蜂鸣输出脉冲频率的 1024 分之 1。例如 Beep 当前输出的蜂鸣脉冲为 1kHz，那么定时触发 ADC 采样的频率为 1kHz/1024 ≈ 0.98Hz（周期约为 1.02 秒）。

28.4 BEEPER 寄存器

28.4.1 配置寄存器（BEEP_CFGR）

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

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

位 31:5	保留
位 4:3	**TRGO_PRE[1:0]**:选择输出 TRGO 定时信号的预分频时钟，TRGO 定时信号的频率为预分频时钟信号的 1024 分频。 00: 1kHz@LSI 时钟源（时钟源 128 分频） 01: 2kHz@LSI 时钟源（时钟源 64 分频） 10: 4kHz@LSI 时钟源（时钟源 32 分频） 11: 8kHz@LSI 时钟源（时钟源 16 分频）
位 2:1	**BEEP_FREQ**: 00: 1kHz@LSI/HSE 时钟源（时钟源 128 分频） 01: 2kHz@LSI/HSE 时钟源（时钟源 64 分频） 10: 4kHz@LSI/HSE 时钟源（时钟源 32 分频） 11: 8kHz@LSI/HSE 时钟源（时钟源 16 分频）
位 0	**BEEP_CKSEL**: 0:选择 128KHz LSI 作为 BEEP 的计时时钟 1:选择 HSE 作为 BEEP 的计时时钟

28.4.2 控制寄存器（BEEP_CR）

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

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

位 31	**CR_WBUSY**: (只读寄存器) 0: CR 寄存器未被 APB 总线写操作 1: CR 寄存器正在被 APB 总线写操作 （当 CR_WBUSY=1 时，软件禁止再次对 CR 寄存器进行写操作）
位 30:2	保留

位 1	**TRGO_EN**: 0：关闭 TRGO 的输出（TRGO 信号保持 0 电平输出），并复位 10-bit 的 TRGO counter 计数器 1：开启 TRGO 的输出 （当 CR_WBUSY=1 时，对 TRGO_EN 寄存器的写操作将无效。）
位 0	**BEEP_EN**: 0:关闭 Beeper,并复位 7-bit 的 Beeper counter 计数器和 10-bit 的 TRGO counter 计数器 1:开启 Beeper (当 CR_WBUSY=1 时，对 BEEP_EN 寄存器的写操作将无效。)

29 通用串行总线全速设备接口 (USB)

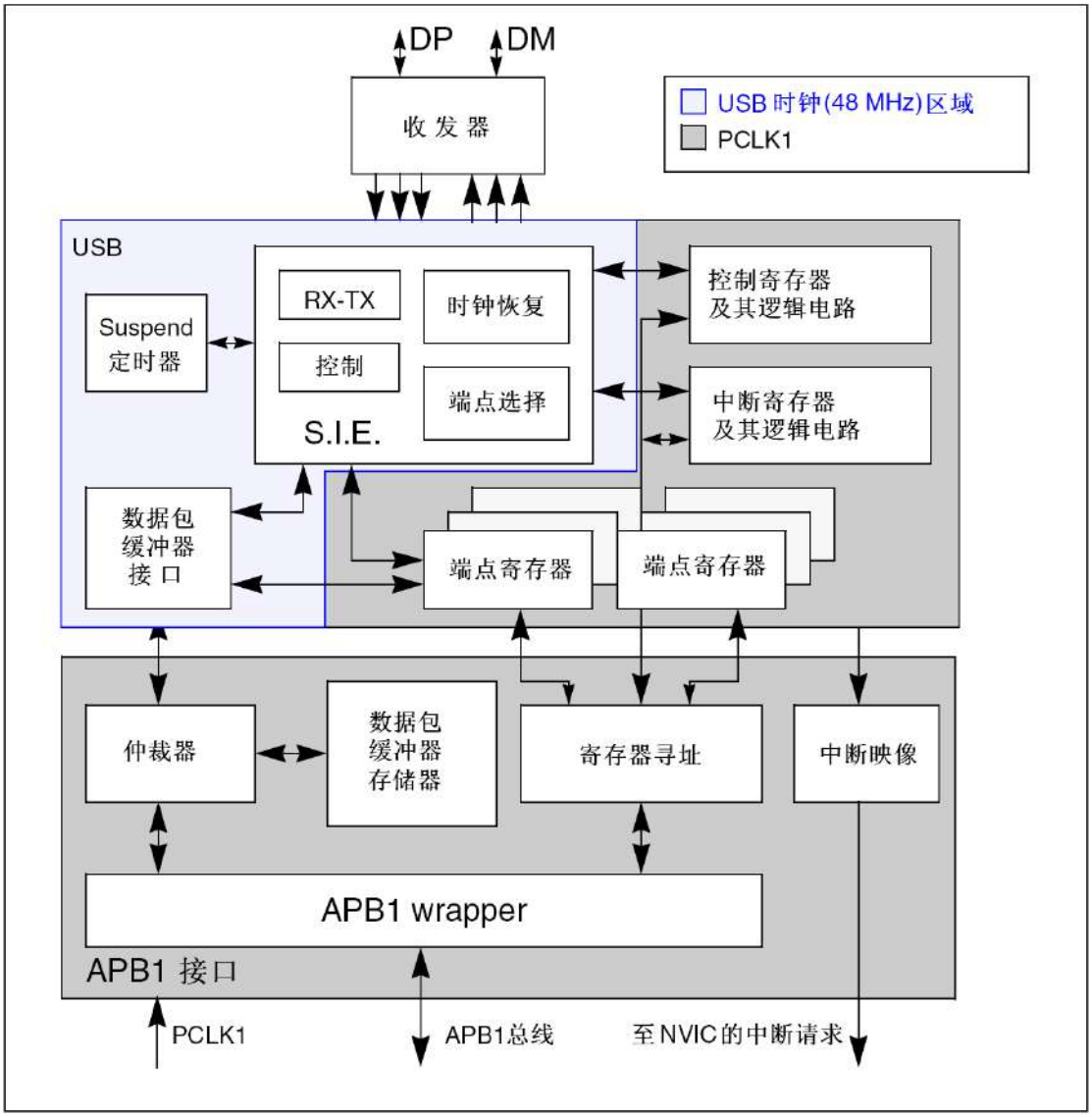
29.1 USB 简介

USB 外设实现了 USB2.0 全速总线和 APB1 总线间的数据接口。
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。

29.2 USB 主要特片

- 符合 USB 2.0 全速设备的技术规范。
- 可配置 1 到 8 个 USB 端点。
- 512 字节的专用数据包缓冲区(SRAM)。
- 带有 CRC(循环冗余校验)生成/校验单元，反向不归零(NRZI)编码/解码和位填充单元。
- 支持同步传。
- 批量/同步端点支持双缓冲区模式。
- 支持 USB 设备挂起/恢复操作。
- 帧锁定时钟脉冲生成。
- USB_DP 线上内置上拉电阻。
- 支持无外部晶体的 USB 应用。
- 注意: 在不使用 USB 外设的情况下，用户可以将 USB 的 512Bytes 数据缓冲区(实际是 1K Bytes)用于其他用途。

图 29-1 USB 设备框图



29.3 USB 功能描述

USB 模块提供了一个在 USB 主机与 MCU 之间的符合 USB 规范的通信接口。USB 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。

29.3.1 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：

(1)串行接口控制器(SIE)：该模块包括的功能有：帧头同步域的识别，位填充，CRC 的产生和校验，PID 的验证/产生，和握手分组处理等。它与 USB 收发器交互，利用分组缓冲接口提供的虚拟缓冲区存储局部数据。它也根据 USB 事件，和类似于传输结束或一个包正确接收等与端点相关事件生成信号，例如帧首(Start of Frame)，USB 复位，数据错误等等，这些信号用来产生中断。

(2)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。

(3)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。

(4)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。

(5)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。

(6)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：

(1)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。

(2)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。

(3)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。

(4)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。

(5)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。

29.4 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。

29.4.1 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 事件所驱动的。

29.4.2 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份，这点需要特别处理。此位能打开为端点收发器供电的内部参考电压。由于打开内部电压需要一段启动时间(数据手册中的 $t_{STARTUP}$)，在此期间内 USB 收发器处于不确定状态，所以在设置 CNTR 寄存器的 PDWN 后必须等待一段时间后，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符，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 复位的流程来处理(参见下段)。

29.4.2.1 USB 复位(RESET 中断)

发生 USB 复位时，USB 模块进入前面章节中描述过的系统复位状态: 所有端点的通信都被禁止(USB 模块不会响应任何分组)。在 USB 复位后，USB 模块被使能，同时地址为 0 的默认控制端点(端点 0)也需要被使能。这可以通过配置 USB_DADDR 寄存器的 EF 位，EP0R 寄存器和相关的分组缓冲区来实现。在 USB 设备枚举阶段,主机将给设备分配一个唯一的地址，这个地址必须写入 USB_DADDR 寄存器的 ADD[6:0]位中，同时配置其它所需的端点。

当复位中断产生时，应用程序必须在中断产生后的 10ms 之内使能端点 0 的传输。

29.4.2.2 分组缓冲区的结构和用途

每个双向端点都可以接收或发送数据。接收到的数据存储在该端点指定的专用缓冲区内，而另一个缓冲区则用于存放待发送的数据。对这些缓冲区的访问由分组缓冲区接口模块实现，它提出缓冲区访问请求，并待待确认信息后返回。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，另一半周期保证 USB 模块的访问。这样，微控制器和 USB 模块对分组缓冲区的访问如同对一个双端口 SRAM 的访问，即使微控制器连续访问缓冲区，也不会产生访问冲突。

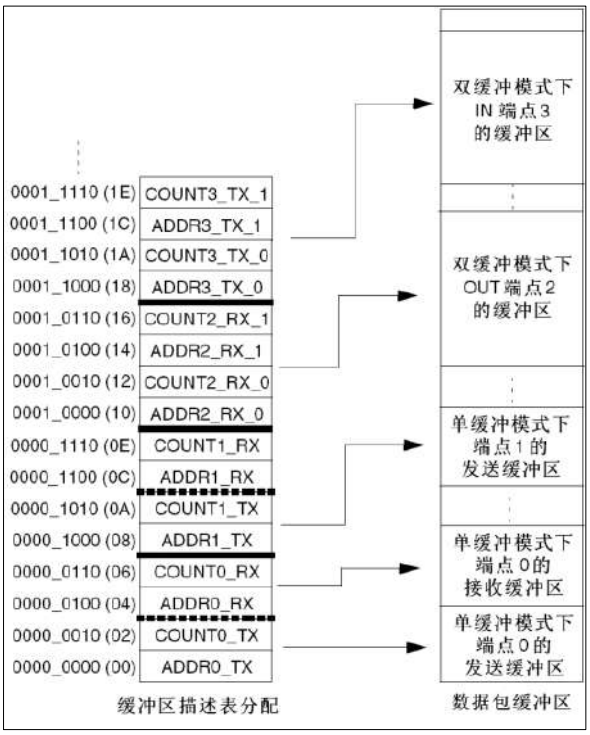
USB 模块使用固定的时钟，按照 USB 标准，此时钟频率固定为 48MHz。APB1 总线的时钟可以大于或者小于这个频率。

注意: 为满足 USB 数据传输率和分组缓冲区接口的系统需求，APB1 总线时钟的频率必须大于 8MHz，以避免数据缓冲区溢出或不满 。

每个端点对应于两个分组缓冲区(一般一个用于发送，另一个用于接收)。这些缓冲区可以位于整个分组存储区的任何位置，因此它们的地址和长度都定义在缓冲区描述表中，而缓冲区描述表也同样位于分组缓冲区中，其地址由 USB_BTABLE 寄存器确定。

缓冲区描述表的每个表项都关联到一个端点寄存器，它由 4 个 16 位的字组成，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')。后面的"缓冲区描述表"章节将详细介绍缓冲区描述表表项。如果是非同步非双缓冲的单向端点，只需要一个分组缓冲区(IN 端点只需要一个发送方向上的分组缓冲区，OUT 端点只需一个接收方向上的分组缓冲区)。其它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它用途。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: "同步传输"和"双缓冲端点"章节)。下图描述了缓冲区描述表项和分缓冲区区域的关系。

图 29-2 缓冲区描述表项和缓冲区区域关系



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缓冲区区域以外的其它缓冲区的内容。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，其它的丢掉，即发生了所谓的缓冲区举出异常。

29.4.2.3 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 模块能找到需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 位确定发送数据的长度。作为接收方，需要设置 STAT_RX 位来使能端点，并设置 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 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。

29.4.2.4 IN 分组(用于数据发送)

当接收到一个 IN 令牌分组时，如果接收到的地址和一个配置好的端点地址相符合的话，USB 模块将会根据缓冲区描述表的表项访问相应的 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"段落)。在 IN 分组传输完毕后，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 将会被发送。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'，使端点变为无效，CTR_TX 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'(置端点为有效状态)，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 部分详细说明。当接收到一个 OUT 或 SETUP 分组时，如果地址和某个有效端点的地址相匹配，

USB 模块将访问缓冲区描述表，找到与该端点相关的 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 值自动递减，COUNT 值自动递增。当检测到数据分组的结束信号时， USB 模块校验收到 CRC 的正确性。如果传输中没有任何错误发生，则发送 ACK 握手分组到主 机。即使发生 CRC 错误或者其他类型的错误(位填充，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。

29.4.2.5 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。

29.4.3 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，则须使用两个 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_R

X(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 29-1 双缓冲批量端点缓冲区标识定义表

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_BUG	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 29-2 双缓冲批量端点的缓冲区使用标识表

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	无(1)	ADDRn_TX_0 / COUNTn_TX_0
	1	1	无(1)	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	无(1)	ADDRn_RX_0 / COUNTn_RX_0
	1	1	无(1)	ADDRn_RX_0 / COUNTn_RX_0

端点处于 NAK 状态

可以通过以下方式设置一个双缓冲批量端点:

将 USB_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 64)，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。

29.4.4 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 29-3 同步端点的缓冲区使用标识

端点类型	DTOG 位值	USB 模块使用的缓冲区	应用程序使用的缓冲区
IN 端点	0	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。

29.4.5 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(此项选)。

清零 USB_CNTR 寄存器的 FSUSP 位。

USB_FNR 寄存器的 RXDP 和 RXDM 位可以用来判断是什么触发唤醒事件，如表 66 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 29-4 唤醒事件检测表

[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。

29.5 USB 寄存器

29.5.1 通用寄存器

这组寄存器用于定义 USB 模块的工作模式，中断的处理，设备的地址和读取当前帧的编号。

29.5.1.1 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	PMAOVRM	ERRM	WKUPM	SUSPM	RESETM	SOFM	ESOFM	Res.	Res.	Res.	RESUME	FSUSP	LP_MODE	PDWN	FRES
rw	rw	rw	rw	rw	rw	rw	rw				rw	rw	rw	rw	Rw
0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

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 7:5	**Res:** 保留位
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：进入低功耗模式
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。

29.5.1.2 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	PMAOVR	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				r	r	r	r	r
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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。

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此特填充错误，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(因为同步传输不支持重传机制)因此数据可能会丢失。此位应用程序可读可写，但仅有写 0 时有效，写 1 无效。
位 13	**ERR:** 出错。在 USB 发生以下错误时，该位由硬件置位: 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、数据或者 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 模块和主机在发生这些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(连接线松动，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 CNTR 寄存器的 LP_MODE 位将被清零，同时 USB_WAKEUP 被激活，通知设备的其他部份(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，但仅写 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：**帧首标识。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	**ESOFM:** 期望帧首标识。此位在 USB 模块未收到期望的 SOF 分组时由硬件置位。主机应该每毫秒都发送 SOF 分组，但如果 USB 模块没有收到，挂起定时器将触发此中断。如果连续发生 3 次 ESOF 中断，也就是连续 3 次未收到 SOF 分组，将产生 SUSP 中断。即使在挂起定时器未被锁定时发生 SOF 分组丢失，此位也会被置位。此位应用程序可读可写，但只有写 0 有效，写 1 无效。
位 7:5	**Res:** 保留位
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 为'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 为 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 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相关的 USB_EPnR 的寄存器位，因为它表示中断挂起的方向。此位应用程序只读。
位 3:0	**EP_ID[3:0]:** 端点标识符。这些位将由硬件根据已生成中断请求的端点编号写入。如果有多个端点事务挂起，则硬件将写入与优先级最高的端点相关的端点标识符，优先级的定义方式如下：定义两个端点组，同步端点和双缓冲批量端点的优先级最高，其他端点次之。如果来自同一组的多个端点请求一个中断，则 USB_ISTR 寄存器中的 EP_ID 位将按照请求端点寄存器的编号升序分配，即 EP0R 的优先级最高，EP1R 次之，依此类推。应用软件可按照此优先级方案为每个端点分配一个寄存器，以便对同时发生的多个端点请求进行适当排序。这些位为只读位。

29.5.1.3 USB 帧数寄存器(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
0	0	0	0	X	X	X	X	X	X	X	X	X	X	X	X

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。当 ESOF 事件发生时，硬件会将丢失的 SOF 分组的数目写入这些位。如果再次收到 SOF 分组，硬件会清除此位。
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。此部份发生 SOF 中断时更新。

29.5.1.4 USB 设备地址寄存器(USB_DADDR)

偏移地址：0x4C

复位值：0x0000

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

位 15:8	保留。
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。ADD[6:0]包含了设备地址。如果此位为 0，USB 模块将停止工作，忽略所有寄存器的设置，不响应任何 USB 通信。

位 6:0	**ADD[6:0]:** 设备地址。这些位记录了 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)必须和 USB 令牌分组中的地址令牌匹配，才能在指定的端点进行正确的 USB 传输。

29.5.1.5 USB 分组缓冲区描述表地址寄存器(USB_BTABLE)

偏移地址：0x50

复位值：0x0000

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

位 15:3	**BTABLE[15:3]:** 缓冲区描述表起始地址。这些位记录了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(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的地址和大小令牌。
位 2:0	保留位。由硬件置 0.

29.5.2 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)必须为 0。不同的端点必须使用不同的端点号，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。

29.5.2.1 USB 端点 n 寄存器(USB_EPnR, n=0..7)

地址偏移：0x00 至 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		EP_KINK	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
0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

当 USB 模块收到 USB 总线复位信号或 CNTR 寄存器的 FRES 位置位时，USB 模块将会复位，同时除了 CTR_RX 和 CTR_TX 以外，该寄存器的其它位也会复位。CTR_RX 和 CTR_TX 状态不发生成变是为了避免丢失紧跟在 USB 复位事件后的数据包通知。每个端点都有自己的 USB_EPnR 寄存器，其中 n 是端点编号(标识符)。

由于在读操作和写操作之间，一些位会被硬件置 1，并将在 CPU 有时间检测变化之前由下一次写操作修改，因此应避免这类寄存器上的读-修改-写周期。为此，受该问题影响的所有位都有一个"不变量"值，无需修改时必须使用该值。建议通过加载指令来修改这类寄存器，即向只能通过硬件修改的所有位写入其"不变量"值。

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 USB_CNTR 寄存器中的 CTRM 位置位，相应的中断会产生。收到的是 OUT 分组还是 SETUP 分组可以通过下面的 SETUP 位来确定。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(请参考章节: “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(请参考章节: “同步传输”)而不需进行翻转。同步传输不需要握手分组，因此硬件在接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(对于非控制传输端点，初始化是必须的)，或者翻转此位用于特殊用途。
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，下表接收状态编码列出了端点的所有状态。当一次正确的 OUT 或 SETUP 数据传输完成后(CTR_RX=1)，硬件会自动设置此位为 NAK 状态，使应用程序有足够的时间在处理完当前传输的数据后，响应下一个数据分组。 对于双缓冲批量端点，由于使用特殊的传输流量控制策略，因此根据使用的缓冲区状态控制传输状态(请参考章节: 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_TYPE[1:0]:** 端点类型。此位用于指示端点当前的类型，所有的端点类型都在***端点类型编码表***中列出。所有的 USB 设备都必需包 含一个地址为 0 的控制端点，如果需要可以有其他地址的控制端点。只有控制端点才会有 SETUP 传输，其他类型的端点无视此类传输。SETUP 传输不能以 NAK 或 STALL 分组响应，如果控制端点在收到 SETUP 分组时处于 NAK 状态，USB 模块将不响应分组，就会出现接收错误。如果控制端点处于 STALL 状态，SETUP 分组会被正确接收，数据会被正确传输，并产生一个正确传输完成的中断。控制端点的 OUT 分组安装普通端点的方式处理。 批量端点和中断端点的处理方式非常类似，仅在对 EP_KIND 位的处理上有差别。 同步端点的用法请参考章节：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***端点特殊类型定义表***。 DBL_BUF：应用程序设置此位能使能批量端点的双缓冲功能。详见章节：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。此位由硬件在一个正确的 IN 分组传输完成后置位。如果 CTRM 位已被置位，会产生相应的中 断。应用程序需要在处理完该事件后清除此位。在 IN 分组结束时，如果主机响应 NAK 或 STALL 则此位不会被置位，因为数据传输没有成功。 此位应用程序可读可写，但写 0 有效，写 1 无效。
位 6	**DTOG_TX:** 发送数据翻转位。对于非同步端点，此位用于指示下一个要传输的数据分组的 Toggle 位(0＝DATA0，1=DATA1)。在一个成功传输的数据分组后，如果 USB 模块接收到主机发送的 ACK 分组，就会翻转此位。对于控制端点，USB 模块会在收到正确的 SETUP PID 后置位此位。对于双缓冲端点，此位还可用于支持分组缓冲区交换(请参考章节：双缓冲端点)。对于同步端点，由于只传送 DATA0，因此该位只用于支持分组缓冲区交换(请参考章节：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]:** 用于指示发送端点的状态。此位用于标识端点的当前状态，表发送状态编码列出了所有的状态。应用程序也可以翻转这些位来切换发送端点的状态。在正确完成一次 IN 分组传输后(CTR_TX=1)，硬件会自动设置此位为 NAK 状态，保证应用程序有足够的时间准备好数据响应后续的数据传输。 对于双缓冲批量端点，由于使用特殊的传输流量控制策略，是根据缓冲区的状态控制传输的状态的(请参考章节: “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 NAK，则 USB 模块后续的操作是未知的。
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 29-5 接收状态编码表

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 29-6 端点类型编码表

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 29-7 端点特殊类型定义表

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 29-8 发送状态编码表

STAT_TX[1:0]	描述
00	DISABLED：端点忽略所有的发送请求。
01	STALL：端点以 STALL 分组响应所有的发送请求。
10	NAK：端点以 NAK 分组响应所有的发送请求。
11	VALID：端点可用于发送。

29.5.3 缓冲区描述表

虽然缓冲区描述表位于分组缓冲区内，但仍可将它看作是特殊的寄存器，用以配置 USB 模块和微控制器内核共享的分组缓冲区的地址和大小。由于 APB1 总线按 32 位寻址，所以所有的分组缓 冲区地址都使用 32 位对齐的地址，而不是 USB_BTABLE 寄存器和缓冲区描述表所使用的地址。

以下介绍两种地址表示方式：一种是应用程序访问分组缓冲区时使用的，另一种是相对于 USB 模块的本地地址。供应用程序使用的分组缓冲区地址需要乘以 2 才能得到缓冲区在微控制器中的 真正地址。分组缓冲区的首地址为 0x4000 6000。下面将描述与 USB_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章节：系统复位和上电复位。

29.5.3.1 发送缓冲区地址寄存器 n(USB_ADDRn_TX, n=0..7)

地址偏移: [USB_BTABLE] + n x 16

USB 地址地址: [USB_BTABLE] + n x 8

复位值: 0x0000

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

位 15:1	**ADDRn_TX:** 发送缓冲区地址。这些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	**保留:** 硬件写 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'。

29.5.3.2 发送数据字节数寄存器 n(USB_COUNTn_TX，其中 n 是端点号)

地址偏移: [USB_BTABLE] + n x 16 + 4

USB 地址地址: [USB_BTABLE] + n x 8 + 2

复位值: 0x0000

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

位 15:10	**保留位:** 由于 USB 模块支持的最大数据分组为 1024 个字节，所以 USB 模块忽略这些位。
位 9:0	**COUNTn_TX:** 发送数据字节数。这些位记录了收到下一个 IN 分组时要发送的数据的字节数。

注意: 双缓冲区批量端点和同步传输端点有两个 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
Res.	Res.	Res.	Res.	Res.	Res.	COUNTn_TX_1[9:0]									
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	COUNTn_TX_0[9:0]									
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

29.5.3.3 接收缓冲区地址寄存器 n(USB_ADDRn_RX，其中 n 是端点号)

地址偏移: [USB_BTABLE] + n x 16 + 8

USB 地址地址: [USB_BTABLE] + n x 8 +4

复位值: 0x0000

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

位 15:1	**ADDRn_RX:** 接收缓冲区地址。这些位记录了收到下一个 OUT 或 SETUP 分组时，用于保存数据的缓冲区的起始地址。
位 0	**保留:** 硬件写 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'。

29.5.3.4 接收数据字节数寄存器 n(USB_COUNTn_RX，其中 n 为端点号)

偏移地址: [USB_BTABLE] + n x16 + 12

USB 本地地址: [USB_BTABLE] + n x8 + 6

复位值: 0x0000

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
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

该寄存器用于存放接收分组时需要使用到的两个参数。最高的标志位定义了接收分组缓冲区的大小，以便 USB 模块检测缓冲区的溢出。低标志位则用于 USB 模块记录实际接收到的字节数。由于有效位数的限制，缓冲区的大小由分配到的存储区块数表示，而存储区块的大小则由所需的缓冲区大小决定。缓冲区的大小在设备枚举过程中定义，由端点描述符的参数 maxPacketSize 表述。(具体信息请参考"USB 2.0 协议规范")

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 32－1024 字节，符合 USB 协议定义的最大分组长度限制。
位 14:10	**NUM_BLOCK[4:0]:** 存储区块的数目。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缓冲区大小。具体请参考后面的: "分组缓冲区大小定义"表。
位 9:0	**COUNTn_RX[9:0]:** 接收到的字节数。这些位由 USB 模块写入，用以记录端点收到的和 USB_EPnR 相关的最新的 OUT 或 SETUP 分组的实际字节数。

注意: 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_RX_0，具体内结构如下:

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
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
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
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

表 29-9 分级缓冲区大小定义表

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

30 高级加密标准硬件加速器 (AES)

30.1 简介

AES 硬件加速器可用于通过AES 算法对数据进行加密和解密。加密处理器完全兼容下列标准：

联邦信息处理标准出版物（FIPS PUB 197，2001 年11 月26 日）规定的高级加密标准(AES) 加速器会使用长度为 128 位、192 位或 256 位的密钥对 128 位块进行加密和解密，还可执行密钥生成。为了最大限度地减少使用相同密钥处理多个数据块时CPU 或DMA 执行的写操作，会将加密或解密密钥存储在内部寄存器中。

HK32L08X 系列形芯片只支持电子密码本(ECB) 模式。

AES 支持对传入和传出数据进行DMA 传输（需要2 个DMA 通道）。

使用 TRNG 输出的随机数来随机化 AES 时钟，加强 AES 的安全性.

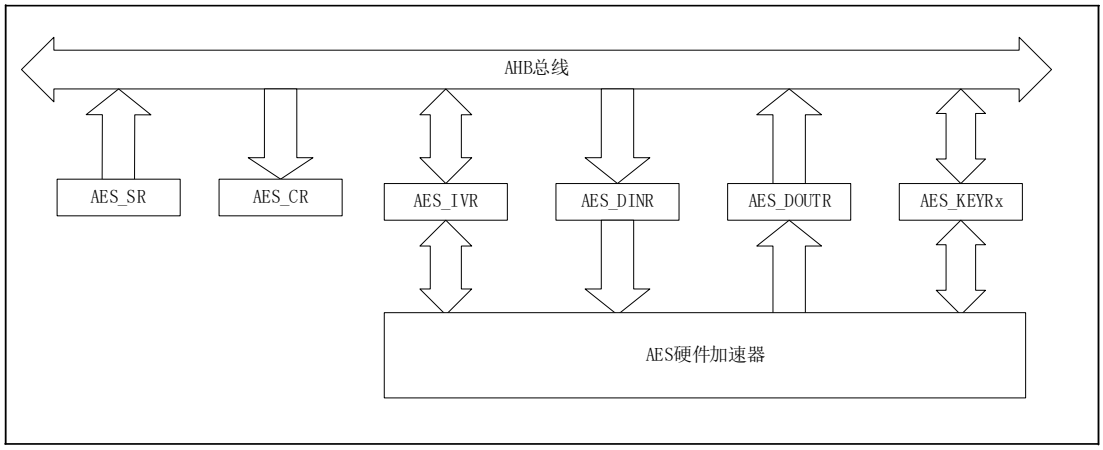
30.2 AES 主要特性

- 使用 AES Rijndael 模块密码算法进行加密/ 解密
- AES 加密 / 解密算法符合 NIST FIPS 197 标准
- 内部 256位寄存器用于存储加密或生成密钥（8 个 32 位寄存器）
- 支持电子密码本 (ECB)
- 解密密钥生成
- 128 位数据块处理
- 128 位、192 位或者 256 位密钥长度可配
- 加解密计算时间
  - 128 位秘钥：57 个时钟周期;
  - 192 位秘钥：67 个时钟周期;
  - 256 位秘钥：77 个时钟周期;
- 解密秘钥准备时间
  - 128 位秘钥：57 个时钟周期;
  - 192 位秘钥：67 个时钟周期;
  - 256 位秘钥：77 个时钟周期;
- 1 个 32 位输入缓冲器和 1 个 32 位输出缓冲器。
  - 寄存器访问仅支持 32 位数据宽度。
- 采用自动数据流控制，支持直接存储器访问 (DMA)（使用 2 个通道，分别用于传入数据和传出数据）

30.3 AES 功能说明

下图显示了AES 加速器的框图。

图 30-1 AES 系统框图



AES 加速器密钥使用长度为 128 位、192 位或 256 位可配；数据长度为 128 位。

它提供 4 种工作模式：

模式 1：使用 AES_KEYRx 寄存器中存储的秘钥进行加密。

模式 2：ECB 解密秘钥准备。在进行模式 3 的 ECB 解密运算之前，需要先使用模式 2 把解密秘钥准备好。在模式 2 执行完成后解密秘钥自动存储在 AES_KEYRx 中。

模式 3：使用存储在 AES_KEYRx 中的秘钥进行解密。

模式 4：使用存储在 AES_KERx 中的秘钥进行单次 ECB 解密运算。(自动完成解密秘钥准备)

通过对 AES_CR 寄存器中的 MODE[1:0] 位进行编程来选择工作模式。仅当 AES 已禁止（AES_CR 寄存器中的位 EN=0）时，才允许切换模式。使能 AES 之前，必须存储 KEY 寄 存器 (AES_KEYRx)。

使能（位 EN=1）后，AES 处于输入阶段，等待软件针对模式 1、模式 3 或模式 4 将输入数据写入 AES_DINR（4 个字）。根据所选模式，数据对应于明文消息或者是密文消息。连续两次对 AES_DINR 寄存器进行写操作时为了在两次数据交替之间，向 AES 处理器发送密钥，将自动插入一个等待周期。

对于模式 2，会在将 AES_CR 寄存器中的 EN 位置 1 后立即开始密钥生成过程。在使能 AES 之前，需要将加密的密钥加载到 AES_KEYRx 寄存器中。密钥生成过程结束时（CFF 标志置 1），会将生成密钥存储在 AES_KEYRx 寄存器中，并由硬件禁止 AES。在该模式下，不得在 AES 已使能的情况下读取 AES_KEYRx 寄存器，应等到 CCF 标志由硬件置 1 后才能读取。

计算阶段完成后，会立即将 AES_SR 寄存器中的状态标志 CCF（计算完成标志）置 1。如 果 AES_CR 寄存 器中的 CCFIE=1，可 产 生 中 断。随 后，软 件 可 从 AES_DOUTR 寄存 器（对于模式 1、模式 3、模式 4）或 AES_KEYRx 寄存器（若选择模式 2）中读回数据。

在输入和输出阶段，软件必须连续读取或写入数据字节（处于模式 2 时除外），但 AES 允许每个读取或写入操作之间存在延迟（示例：如果在此时为另一中断提供服务）。

如果检测到未预期的读取或写入操作，则会将 AES_SR 寄存器中的 RDERR 和 WRERR 标志置 1。如果 AES_CR 寄存器中的 ERRIE 位置 1，可产生中断。检测到错误后，AES 不会禁止，而是会继续照常处理。

可随时通过将 AES_xCR 寄存器中的 EN 位置 0 来重新初始化 AES。随后，可通过将 EN 置 1 重新启动 AES，此时，AES 会等待第一个输入数据字节写入（处于模式 2 下时除外，此时在 EN 位置 1 后，会立即从 AES_KEYRx 寄存器中存储的值开始密钥生成过程）。

30.4 加密和生成密钥

AES_KEYRx 寄存器用于存储加密或解密密钥。这 8 个寄存器使用小端模式配置进行组织：必须将密钥的 32 位 LSB 载入寄存器 AES_KEYR0 中，以此类推，必须将 256 位密钥的 32 位 MSB 载入 AES_KEYR7。

AES 禁止时（AES_CR 寄存器中的EN = 0），必须将加密或解密密钥存储在这些寄存器中。密钥的字节顺序是固定的。

在模式 2（密钥生成）中，需要将加密密钥载入到AES_KEYRx 中。然后必须使能AES。计算阶段结束时，生成密钥会自动存储在 AES_KEYRx 寄存器中，并会覆盖之前的加密密钥。生成密钥可用时，将由硬件禁止AES。

在模式 4（密钥生成 + 解密）中，AES_KEYRx 寄存器只包含加密密钥。生成密钥会在内部计算，不会对这些寄存器执行任何写操作。

30.5 电子密码本 (ECB)

消息会划分到各个块中，并对各个块分别进行独立加解密。

下图 ECB 加密模式和图 ECB 解密模式分别介绍了用于加密和解密的电子密码本算法的原理。

图 30-2 ECB 加密模式

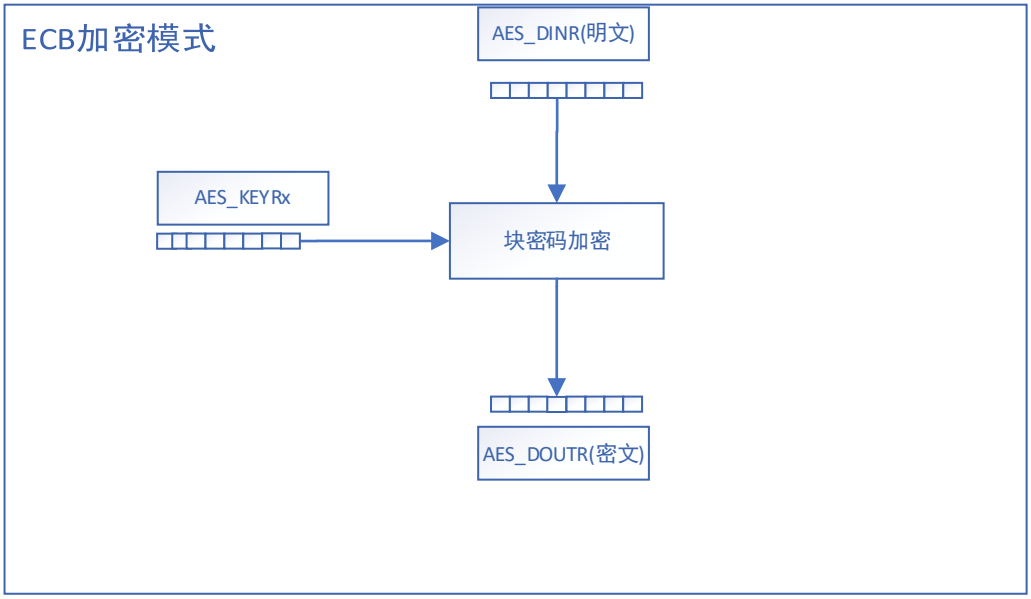
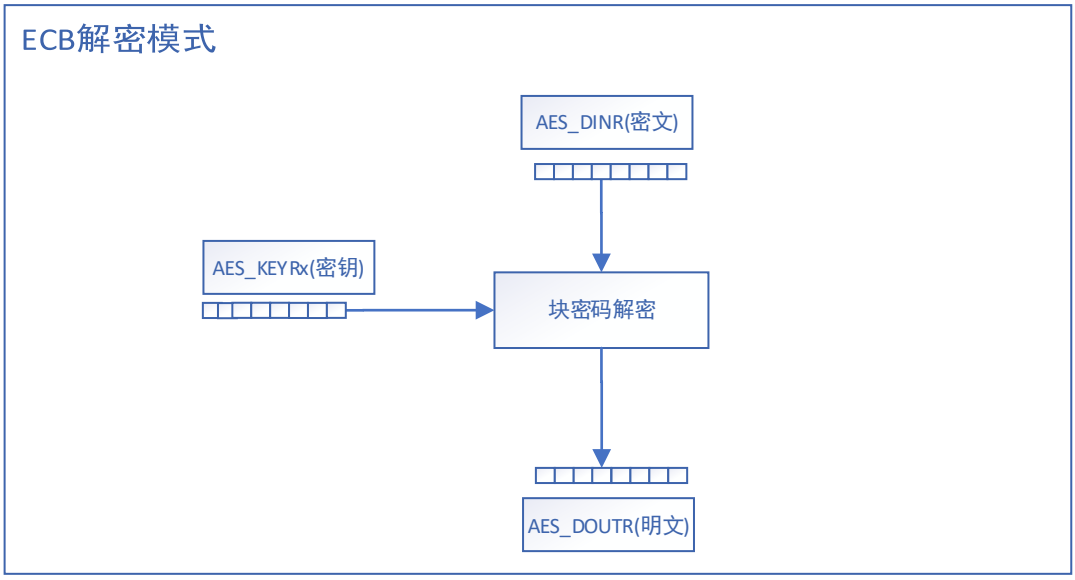


图 30-3 ECB 解密模式

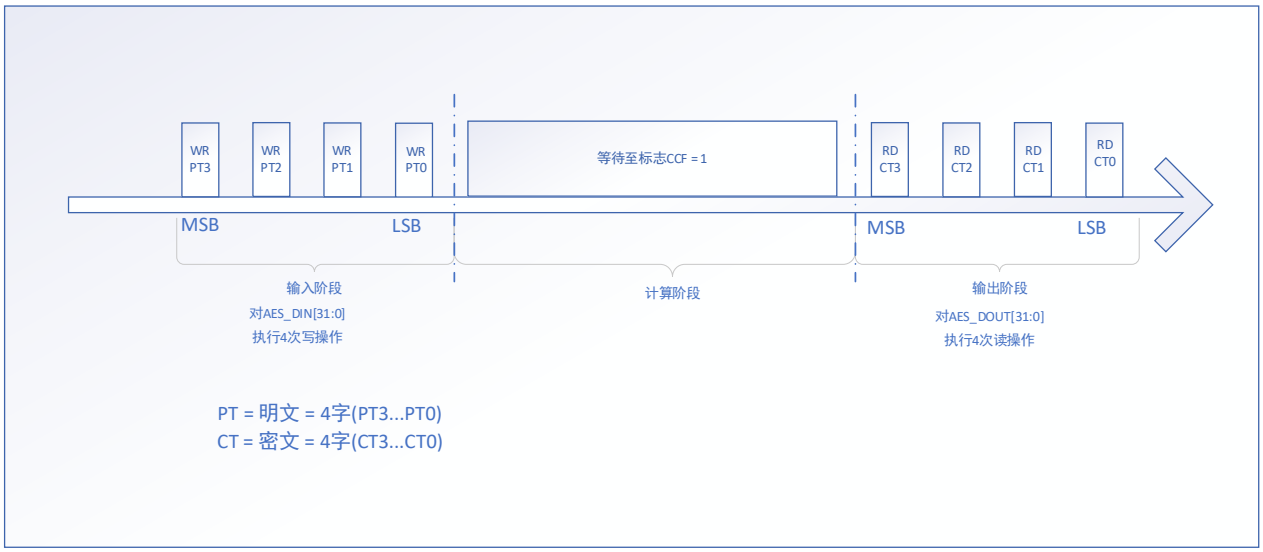


30.6 工作模式

30.6.1 模式 1：加密

1.通过将 AES_CR 寄存器中的EN 位置 0 禁止 AES。
2.通过在 AES_CR 寄存器中编程 MODE[1:0]=00 来配置模式 1。
3.通过将AES_CR 寄存器中的EN 位置 1 使能AES。
4.对 AES_DINR 寄存器执行 4 次写入操作，以输入明文（MSB 优先），如下图模式 1：加密所示。
5.等待 AES_SR 寄存器中的 CCF 标志置 1。
6.对 AES_DOUTR 寄存器执行 4 次读取操作，以获取密文（MSB 优先），如下图模式 1：加密所示。
7.重复执行步骤 4、5、6，处理使用相同加密密钥的所有块。

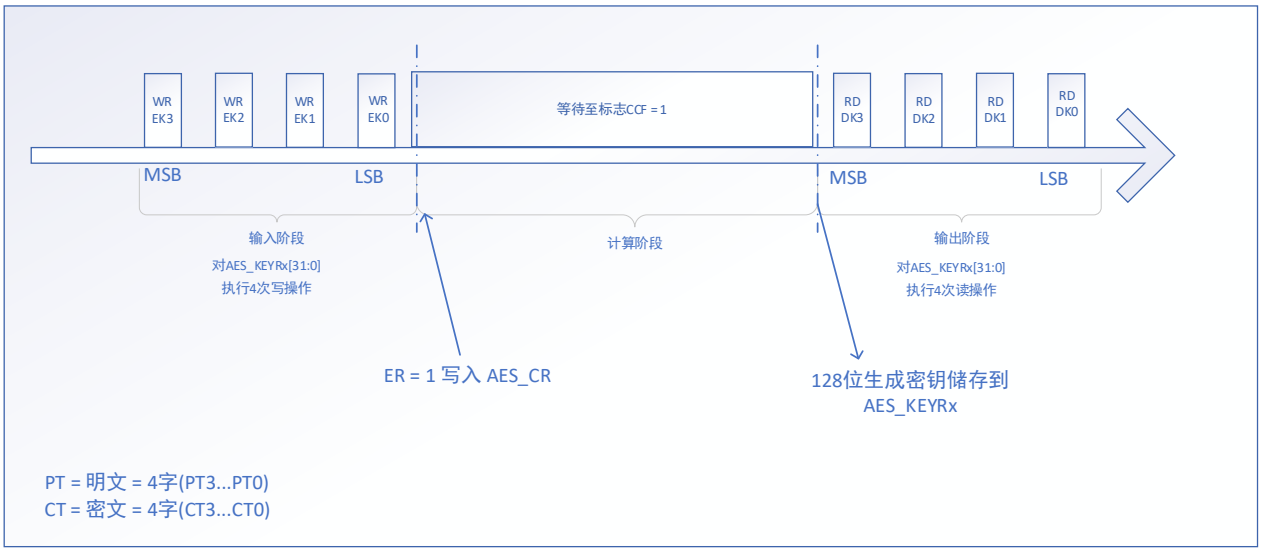
图 30-4 模式 1：加密



30.6.2 模式 2：密钥生成

1.通过将 AES_CR 寄存器中的 EN 位置 0 禁止 AES。
2.通过在 AES_CR 寄存器中编程 MODE[1:0]=01 配置模式 2。
3.将加密密钥写入到 AES_KEYRx 寄存器中，以获取生成密钥。
4.通过将 AES_CR 寄存器中的 EN 位置 1 使能 AES。
5.等待 AES_SR 寄存器中的CCF 标志置 1。
6.生成密钥会自动放入 AES_KEYRx 寄存器。如有需要，可读取 AES_KEYRx 寄存器获得解密密钥。AES 由硬件禁止。要重新开始计算生成密钥，请重复步骤 3、4、5、6。

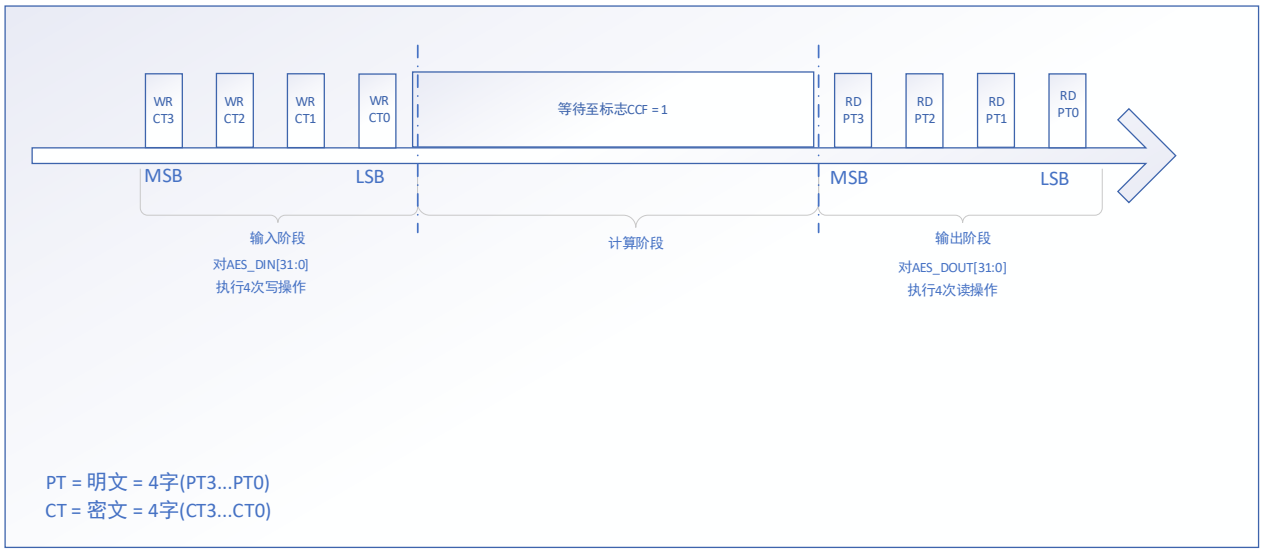
图 30-5 模式 2：密钥生成



30.6.3 模式 3：解密

1. 通过将AES_CR 寄存器中的EN 位置 0 禁止AES。
2. 通过在 AES_CR 寄存器中编程 MODE[3:0]=10 来配置模式 3。
3. 将解密密钥写入 AES_KEYRx 寄存器（如果已使用模式 2，密钥生成将生成密钥存储到 AES_KEYRx 寄存器中，则可略过此步）。
4. 通过将AES_CR 寄存器中的EN 位置 1 使能AES。
5. 对 AES_DINR 寄存器执行 4 次写入操作，以输入密文（MSB 优先），如下图。
6. 等待 AES_SR 寄存器中的CCF 标志置 1。
7. 对 AES_DOUTR 寄存器进行 4 次读取操作，以获取明文（MSB 优先），如下图。
8. 重复执行步骤 5、6、7，处理使用相同加密密钥（存储在 AES_KEYRx 寄存器中）的块。

图 30-6 模式 3：解密

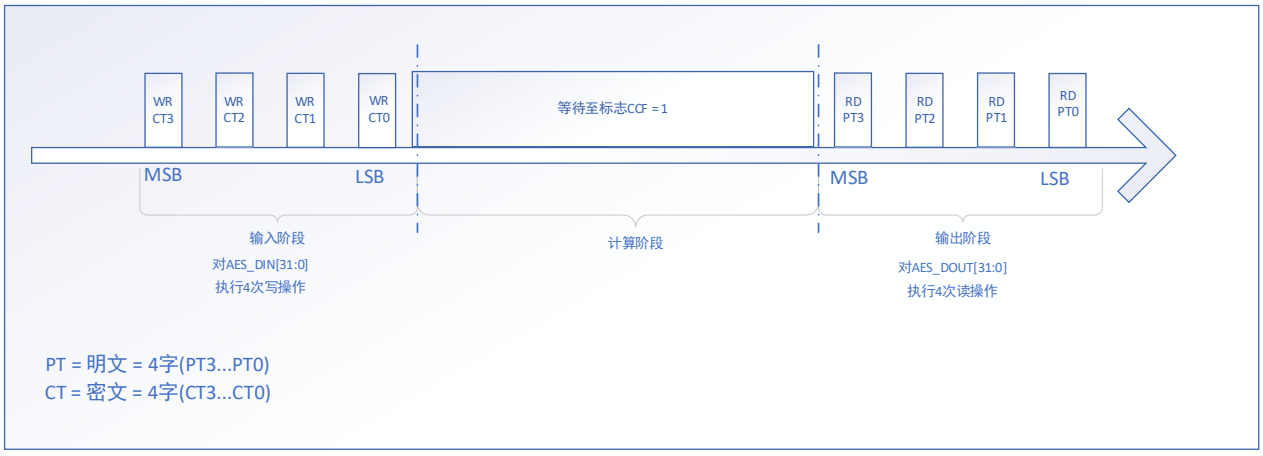


30.6.4 模式 4：单次解密

1.通过将 AES_CR 寄存器中的EN 位置 0 禁止AES。
2.通过在 AES_CR 寄存器中编程 MODE[1:0]=11 配置模式 4。
3.将加密密钥写入AES_KEYRx 寄存器。
4.通过将 AES_CR 寄存器中的 EN 位置 1 使能AES。
5.对 AES_DINR 寄存器执行 4 次写入操作，以输入密文（MSB 优先），如下图所示。
6.等待 AES_SR 寄存器中的 CCF 标志置 1。
7.对 AES_DOUTR 寄存器进行 4 次读取操作，以获取明文（MSB 优先），如下图所示。
8.重复执行步骤 5、6、7，处理所有使用相同加密密钥的块。

注：运行单次解密运算后，AES_KEYRx 会被替换为解密秘钥，因此，在调用单次解密运算时，每次都需要重新输入秘钥到AES_KEYRx 寄存器，即使本次使用的秘钥和上一次使用的秘钥相同。

图 30-7 模式 4：密钥生成和解密



30.7 AES DMA 接口

AES 加速器提供连接 DMA 控制器的接口。

必须对DMA 进行配置才能传输字。

可将AES 关联到两个不同的DMA 请求通道：

- 输入的DMA 请求通道：如果将AES_CR 寄存器中的DMAINEN 位置1，每次AES 要将字写入 AES_DINR 寄存器时，都会在INPUT 阶段发起DMA 请求(AES_IN)。DMA 通道必须配置为存储器到外设模式，数据大小为 32 位。

输出的DMA 请求通道：如果将DMAOUTEN 位置 1 ，每次AES 要从AES_DOUTR 寄存器读取字时，都会在OUTPUT 阶段发起DMA 请求(AES_OUT)。DMA 通道必须配置为外设到存储器模式，数据大小为 32 位。

会为每个阶段使能四个 DMA 请求，下图 “输入阶段的 DMA 请求和数据传输 (AES_IN)”和下图 “输出阶段的 DMA 请求和数据传输 (AES_OUT)”对此进行了介绍。

DMA 请求会一直产生，赶到 AES 被禁用。因此，128 位数据块处理结束时的数据输出阶段之后，AES 会自动切换到下一个数据块的新数据输入阶段（如果存在）。

*注：*对于模式 2（密钥生成），可通过软件使用 *CPU* 访问 *AES_KEYRx* 寄存器。不会为此提供 *DMA* 通道。因此，*AES_CR* 寄存器中的 *DMAINEN* 位和 *DMAOUTEN* 位在该模式下不起作用。

当 *DMAOUTEN = 1* 时，*CCF* 标志不起作用，这种情况下软件不需要读取该标志。如果应用需要禁止 *AES* 来取消 *DMA* 管理并为数据输入或数据输出阶段使用 *CPU* 访问，此位可能保持为 1，必须由软件清零。

图 30-8 输入阶段的 **DMA** 请求和数据传输**(AES_IN)**

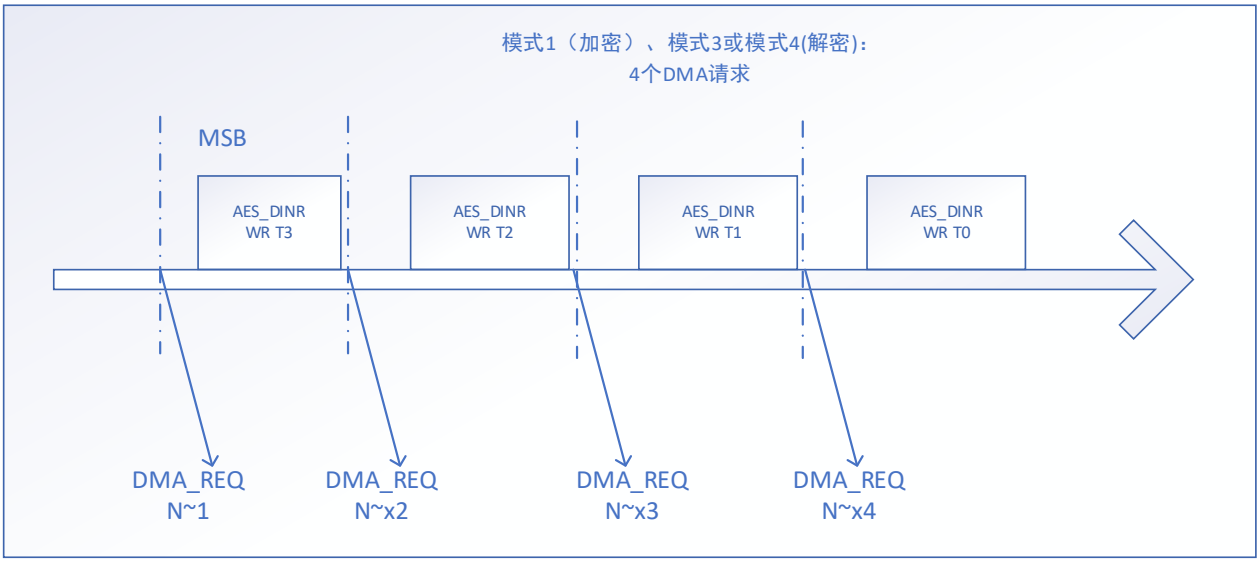
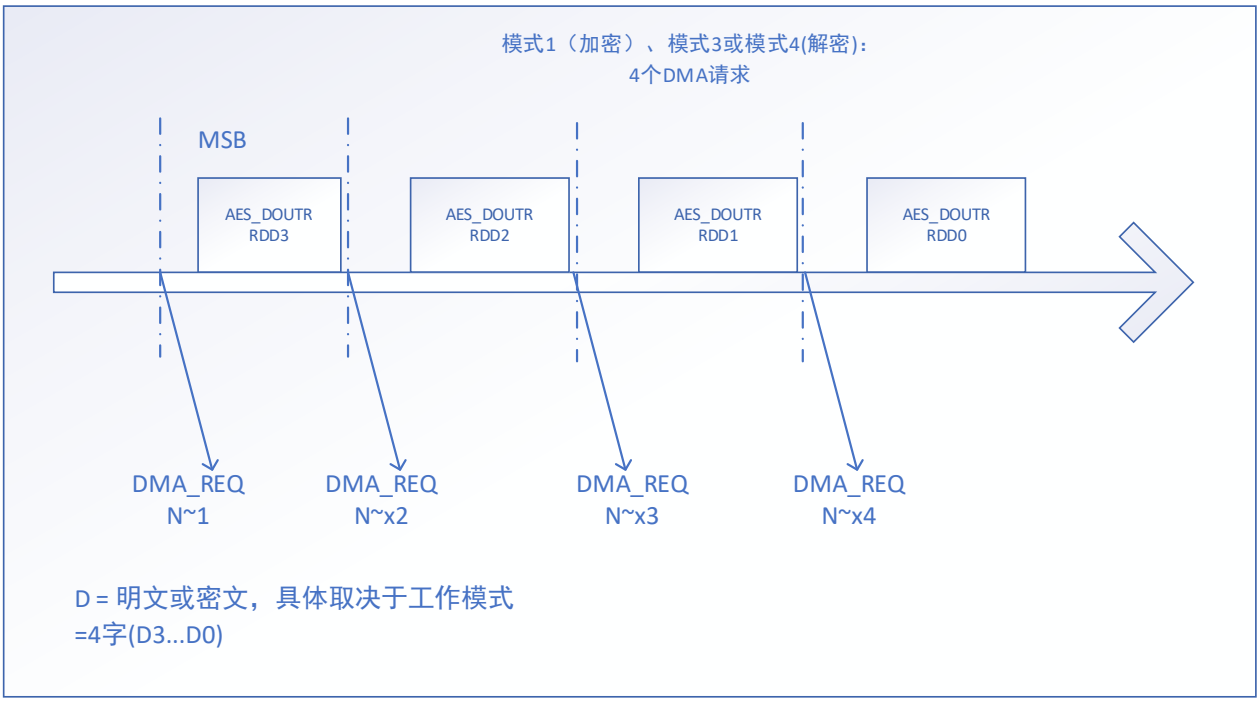


图 30-9 输出阶段的**DMA** 请求**(AES_OUT)**



30.8 错误标志

如果在计算阶段或输入阶段检测到未预期的读取操作，则会将AES_SR 寄存器中的RDERR 标志置 1。

如果在输出阶段或计算阶段检测到未预期的写入操作，则会将 AES_SR 寄存器中的WRERR 标志置 1。

可通过将AES_CR 寄存器中的相应位置 1 的方式将标志清零（CCFC 位可将CCF 标志清零，ERRC 位可将WERR 和RDERR 标志清零）。

如果之前已将AES_CR 寄存器中的ERRIE 位置 1，则当其中一个错误标志置 1 时，可生成 中断。

如果检测到错误，则 AES 不会被硬件禁止并继续照常进行处理。

30.9 处理时间

下表列出了每种工作模式下处理 128 位块需要的时间。

表 30-1 处理时间（时钟周期数）

工作模式		输入阶段	计算阶段	输出阶段	总计
模式1：加密	128 位秘钥	8	57	4	69
	192 位秘钥	10	67	4	81
	256 位秘钥	12	77	4	93
模式2：解密密钥生成	128 位秘钥	4	23	-	27
	192 位秘钥	6	27	-	33
	256 位秘钥	8	31	-	39
模式3：解密	128 位秘钥	4	57	4	65
	192 位秘钥	6	67	4	77

	256 位秘钥	8	77	4	89
模式4：单次解密	128 位秘钥	8	82	4	94
	192 位秘钥	10	96	4	110
	256 位秘钥	12	110	4	126

30.10 AES 中断

一共有三种情况可能产生 AES 中断：

- 计算完成
- 读取错误
- 写入错误

表 30-2 AES 中断请求

中断事件	事件标志	使能控制位
AES 计算完成标志	CCF	CCFIE
AES 读取错误标志	RDERR	ERRIE
AES 写入错误标志	WRERR	ERRIE

每一个中断请求源可以从 AES_SR 寄存器读得。

30.11 挂起与上下文恢复

AES 挂起：

在 AES 运算的过程中，如果高优先级的中断发生，并且该中断服务程序也需要使用 AES 时，需要把当前的 AES 挂起。为了保证计算的正确性，需要在一个计算完成后才挂起 AES。

如果使用了 DMA，清除 AES. DMAINEN。

等待 AES_SR. CCF 变高。

如果使用了 DMA，清除 AES. DMAOUTEN。DMAOUTEN 清除后硬件依然会完成 DMA 读 DOUT 的操作。如果没有使用 DMA，连续读 4 次 AES_DOUTR 寄存器并把结果压栈。

写 CCFC 寄存器清除 CCF 标志。

把 AES_CR. EN 清零使 AES 关闭。

连续 4 次读取 AES_DINR 并压栈。

读取 AES_CR, AES_CR2, AES_KEYRx 寄存器并压栈。

上下文恢复：

把 AES_CR. EN 置 0 关闭 AES。

恢复 AES_CR，AES_CR2 的配置。

恢复 AES_KEYRx 配置。

恢复 AES_DOUTR。

恢复 AES_DINR。

把 AES_CR. EN 置 1 打开 AES。

把 AES_CR2. CCF_SET 和 AES_CR2. INT_RESUME 置 1。

30.12 AES128 寄存器

30.12.1 AES 控制寄存器 (AES_CR)

偏移地址：0x00

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	DMAOUTEN	DMAINEN	ERRIE	CCFIE	ERRC	CCFC	Res.	Res.	MODE[1:0]		Res.	Res.	EN
r	r	r	rw	rw	rw	rw	rw	rw	r	r	rw	rw	r	r	rw
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位31:13	**Res:**读为 0
位12	**DMAOUTEN:**数据输出阶段使能 DMA 管理(Enable DMA management of data output phase) 0：禁止 DMA（在数据输出阶段） 1：使能 DMA（在数据输出阶段） 如果 DMAOUTEN 位置 1，则会在模式 1、3 或 4 下为输出数据阶段生成 DMA 请求。此位在模式 2（解密密钥生成）下不起作用。
位11	**DMAINEN:**数据输入阶段使能 DMA 管理(Enable DMA management of data input phase) 0：禁止 DMA（在数据输入阶段） 1：使能 DMA（在数据输入阶段） 如果 DMAINEN 位置 1，则会在模式 1、3 或 4 下为输入数据阶段生成 DMA 请求。此位在模式 2 （解密密钥生成）下不起作用。
位10	**ERRIE:**错误中断使能 (Error interrupt enable) 0：禁止错误中断 1：使能错误中断 如果 RDERR 或 WRERR 中至少有一个标志置 1，则会生成中断。
位9	**CCFIE:**CCF 标志中断使能 (CCF flag interrupt enable) 0：禁止 CCF 中断 1：使能 CCF 中断 如果 CCF 标志置 1，则会生成中断。
位8	**ERRC:**错误清零 (Error clear) 向该位写入 1 会将 RDERR 和 WRERR 标志清零。 此位始终读为 0
位 7	**CCFC:**计算完成标志清零 (Computation Complete Flag Clear) 向该位写入 1 时会将 CCF 标志清零。 此位始终读为 0。
位 4:3	**MODE[1:0]:** AES 工作模式 (AES operating mode) 00：模式 1：加密 01：模式 2：解密密钥生成 10：模式 3：解密 11：模式 4：单次密钥生成 +解密 仅可在 AES 已禁用的情况下更改工作模式。禁止在 AES 已使能的情况下写入这些位，以免出现不 可预期的 AES 行为。

位 0	**EN:** AES 使能 (AES enable) 0：禁 AES 1：使能 AES 可随时通过将此位清零的方式重新初始化 AES：EN 置 1 后，AES 准备好开始处理新块。如果在模式 2（解密密钥生成）下完成 AES 计算，则由硬件清零此位。

30.12.2 AES 控制寄存器 (AES_CR2)

偏移地址：0x80

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCF_SET	INT_RESUM	RANDOM	Res.	Res.	KEY_SIZE[1:0]	
r	r	r	r	r	r	r	r	r	rw	rw	rw	r	r	rw	rw
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位31:7	**Res**：读为 0
位 6	**CCF_SET:**强制置位 ASE_SR.CCF 写 1：把 AES_SR.CCF 置 1。 写 0：无作用。 读此位始终为 0。 用于在中断返回使恢复上下文。
位 5	**INT_RESUME:** 指示 AES 从中断服务程序恢复 当 INT_RESUME 被置 1 后，会被硬件清零，当 AES 开始恢复执行运算会立即清除 INT_RESUME 位。
位 4	**RANDOM_CLK_EN:**TRNG 输出的随机数来随机化 AES 时钟 1：使用 TRNG 输出的随机数来随机化 AES 时钟，加强 AES 的安全性 0：不随机化 AES 时钟
位 3:2	**Res:**读为 0
位 1:0	**KEY_SIZE[1:0]：** 秘钥长度配置寄存器 00：128 位秘钥长度 01：192 位秘钥长度 10：256 位秘钥长度 11：无效

30.12.3 AES 状态寄存器 (AES_SR)

偏移地址：0x04

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRERR	RDERR	CCF
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位 31:3	**Res**：读为 0
位 2	**WRERR**：写入错误标志 (Write error flag) 如果检测到 AES_DINR 寄存器发生了未预期的写入操作（计算阶段或数据输出阶段），则由硬件将 此位置 1。如果 AES_CR 寄存器中的 ERRIE 位之前已置 1，将生成中断。此标志对 AES 无影响， 即使 WRERR 置 1，AES 也会继续运行。 通过将 AES_CR 寄存器中的 ERRC 位置 1，此位由软件清零。 0：未检测到写入错误 1：检测到写入错误
位 1	**RDERR**：读取错误标志 (Read error flag) 如果检测到 AES_DOUTR 寄存器发生了未预期的读取操作（计算阶段或数据输入阶段），则会通过 硬件将此位置 1。如果 AES_CR 寄存器中的 ERRIE 位之前已置 1，将生成中断。此标志对 AES 无 影响，即使 RDERR 置 1，AES 也会继续运行。 通过将 AES_CR 寄存器中的 ERRC 位置 1，此位由软件清零。 0：未检测到读取错误 1：检测到读取错误
位 0	**CCF**：计算完成标志 (Computation complete flag) 当计算完成时，该位由硬件置 1。如果 AES_CR 寄存器中的 CCFIE 位之前已置 1，将生成中断。也可以通过向 AES_CR2.CCF_SET 写 1，置位 CCF 标志寄存器。 通过向 AES_CR 寄存器中的 CCFC 位写入 1 清零。 0：计算未完成 1：计算已完成

30.12.4 AES 状态寄存器 (AES_SR2)

偏移地址：0x94

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

位 31:5	**Res**：读为 0
位 4	**FST_CAL**:状态由硬件更新，当 AES_EN 为 0 时，把 FST_CAL 置 0；如果 AES_EN 使能后，第一个运算还未完成，硬件把 FST_CAL 置 1，完成第一个运算后硬件会自动把 FST_CAL 清零。 1：CR.AES_EN 使能后的第一个运算还未完成 0： CR.AES_EN 使能后的第一个运算已经完成或者 AES_EN 无效
位 3: 2	**INPUT_CNT**:状态由硬件更新，标识已经输入了多少个字的数据到 DINR 00：表示还未输入或者已经完成输入 4 个字的数据 01：表示输入了 1 个字到 DINR 10：表示输入了 2 个字到 DINR 11：表示输入了 3 个字到 DINR 从中断恢复的时候，可以通过 INPUT_CNT 的值来保证恢复现场。

30.12.5 AES 数据输入寄存器 (AES_DINR)

偏移地址：0x08

复位值：0x0000 0000

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

位 31:0	**DINR**：数据输入寄存器 (Data Input Register) 输入阶段必须对该寄存器执行 4 次写入操作： – 在模式 1（加密）下，必须从 MSB 到 LSB 写入 4 个代表明文的字。 – 在模式 2（密钥生成）下，不会使用该寄存器。 – 在模式 3（解密）和模式 4（单次密钥生成 + 解密）下，必须从 MSB 到 LSB 写入 4 个代表密文的字。 注：该寄存器必须通过 32 位数据宽度进行访问。

30.12.6 AES 数据输出寄存器 (AES_DOUTR)

偏移地址：0x0C

复位值：0x0000 0000

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

位 31:0	**DOUTR:**数据输出寄存器(Data output register)。 该寄存器为只读。 将 CCF 标志（计算完成标志）置 1 后，对该数据寄存器读取 4 次可得到 128 位输出结果： - 在模式 1（加密）下，必须从 MSB 到 LSB 读取 4 个代表密码文本的字。 - 在模式 2（密钥生成）下，不需要读取该寄存器。 - 在模式 3（加密）和模式 4（单次密钥生成 + 解密）下，读取的 4 个字代表从 MSB 到 LSB 的明文。 注：该寄存器必须通过 32 位数据宽度进行访问。

30.12.7 AES 密钥寄存器 0 (AES_KEYR0) (LSB: key [31:0])

偏移地址：0x10

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEYR0[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEYR0[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位 31:0	**KEYR0**：AES 密钥寄存器 (AES key register) (LSB key [31:0]) 必须在将 AES_CR 寄存器中的 EN 位置 1 之前写入该寄存器。 在模式 1（加密）、模式 2（密钥生成）和模式 4（单次密钥生成 + 解密）下，要写入的值代表从 LSB 开 始的加密密钥，也就是 Key [31:0]。 在模式 3（解密）下，要写入的值代表从 LSB 开始的解密密钥，也就是 Key [31:0]。如果在该解密 模式下向寄存器写入加密密钥，则在 AES 使能前读取该寄存器将返回加密值。在 CCF 标志置 1 之 后读取该寄存器将返回生成密钥。 在 AES 已使能时读取该寄存器会返回不可预测的值。 注：在模式 4（单次生成密钥 + 解密）下，该寄存器不包含生成密钥，但始终包含加密密钥值。

30.12.8 AES 密钥寄存器 1 (AES_KEYR1) (Key[63:32])

偏移地址：0x14

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEYR1[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEYR1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位 31:0	**KEYR1**：AES 密钥寄存器 (AES key register) (key [63:32]) 请参见 AES_KEYR0 的说明。

30.12.9 AES 密钥寄存器 2 (AES_KEYR2) (Key [95:64])

偏移地址：0x18

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEYR2[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEYR2[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位 31:0	**KEYR2**：AES 密钥寄存器 (AES key register) (key[95:64]) 请参见 AES_KEYR0 的说明。

30.12.10 AES 密钥寄存器 3 (AES_KEYR3) (key[127:96])

偏移地址：0x1C

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEYR3[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEYR3[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位 31:0	**KEYR3**[31:0]：AES 密钥寄存器 (AES key register) (key[127:96]) 请参见 AES_KEYR0 的说明。

30.12.11 AES 密钥寄存器 4 (AES_KEY4) (key[159:128])

偏移地址：0x84

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEYR4[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEYR4[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位 31:0	**KEYR4**[31:0]：AES 密钥寄存器 (AES key register) (key[159:128]) 请参见 AES_KEYR0 的说明。

30.12.12 AES 密钥寄存器 5 (AES_KEY5) (key[191:160])

偏移地址：0x88

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEYR5[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEYR5[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位 31:0	**KEYR5**[31:0]：AES 密钥寄存器 (AES key register) (key[191:160]) 请参见 AES_KEYR0 的说明。

30.12.13 AES 密钥寄存器 6 (AES_KEY6) (key[223:192])

偏移地址：0x8C

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEYR6[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEYR6[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位 31:0	**KEYR6**[31:0]：AES 密钥寄存器 (AES key register) (key[223:192]) 请参见 AES_KEYR0 的说明。

30.12.14 AES 密钥寄存器 7 (AES_KEY7) (MSB: key[255:224])

偏移地址：0x90

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEYR7[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEYR7[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位 31:0	**KEYR7**[31:0]：AES 密钥寄存器 (AES key register) (MSB key[255:224]) 请参见 AES_KEYR0 的说明。

31 随机数发生器 (TRNG)

31.1 简介

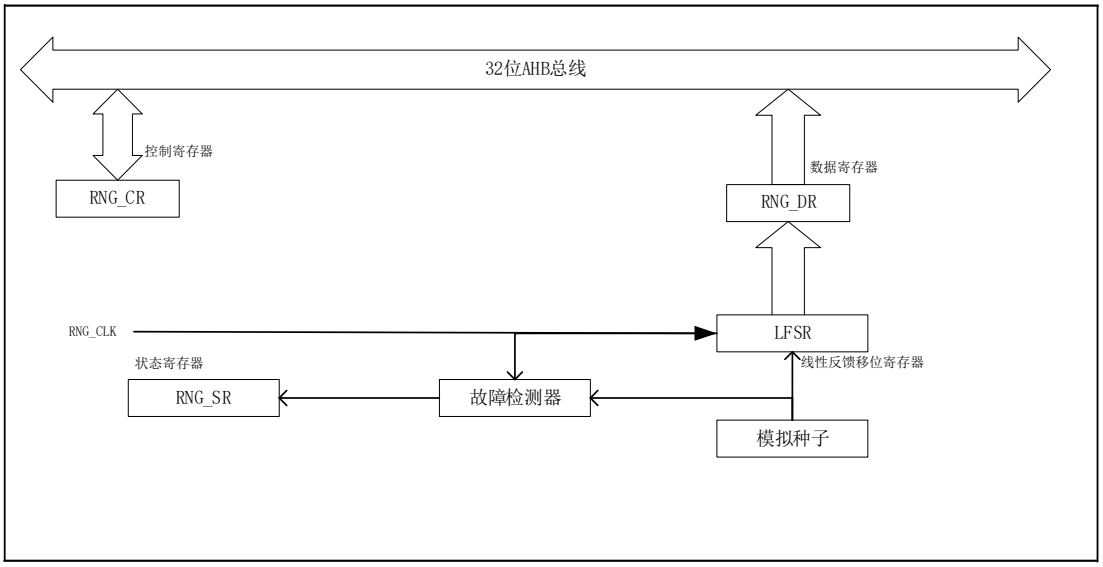
TRNG 处理器是一个以连续模拟噪声为基础的随机数发生器，在主机读数时提供一个 32 位的随机数。TRNG 已通过 FIPS PUB 140-2（2001 年 10 月 10 日）测试，成功率达 99%。

31.2 TRNG 主要特性

- 提供由模拟量发生器产生的 32 位随机数
- 两个连续随机数的间隔为 40 个 RNG_CLK 时钟信号周期
- 通过监视 RNG 熵来标识异常行为（产生稳定值，或产生稳定的值序列）
- 可被禁止以降低功耗

31.3 TRNG 功能说明

图 31-1 显示了 RNG 框图。



随机数发生器采用模拟电路实现。此电路产生馈入线性反馈移位寄存器 (RNG_LFSR) 的种子，用于生成 32 位随机数。

该模拟电路由几个环形振荡器组成，振荡器的输出进行异或运算以产生种子。RNG_LFSR 由专用时钟 (RNG_CLK) 按恒定频率提供时钟信息，因此随机数质量与 HCLK 频率无关。当将大量种子引入 RNG_LFSR 后，RNG_LFSR 的内容会传入数据寄存器 (RNG_DR)。

同时，系统会监视模拟种子和专用时钟 RNG_CLK。状态位（RNG_SR 寄存器中）指示何时在种子上出现异常序列，或指示何时 RNG_CLK 时钟频率过低。检测到错误时生成中断。

31.4 操作

要运行 RNG，请按以下步骤操作：

1. 如果需要，使能中断（为此，将 RNG_CR 寄存器中的 IE 位置 1）。准备好随机数时或出现错误时生成中断。

2. 通过将 RNG_CR 寄存器中的 RNGEN 位置 1 使能随机数产生。这会激活模拟部分、RNG_LFSR 和错误检测器。

3. 每次中断时，检查确认未出现错误（RNG_SR 寄存器中的 SEIS 位应为 0），并且随机数已准备就绪（RNG_SR 寄存器中的 DRDY 位为 1）。然后即可读取 RNG_DR 寄存器中的内容。

按照 FIPS PUB（联邦信息处理标准出版物）140-2 的要求，将 RNGEN 位置 1 后产生的第一个随机数不应使用，但应保存起来，与产生的下一个随机数进行比较。随后产生的每个随机数都需要与产生的上一个随机数进行比较。如果任何一对进行比较的数字相等，则测试失败（连续随机数发生器测试）。

31.5 错误管理

如果 SEIS 位的值为 1（种子错误）出现种子错误时，只要 SECS 位为 1，就会中断随机数产生。如果 RNG_DR 寄存器中有可用随机数，不能使用该随机数，因为它可能没有足够的熵。

应执行以下操作：将 SEIS 位清零，然后将 RNGEN 位清零并置 1，以便重新初始化和重新启动 RNG。

31.6 TRNG 寄存器

31.6.1 RNG 控制寄存器 (RNG_CR)

偏移地址：0x00

复位值：0x0000_6000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OP_CYCLE							Res.	Res.	Res.	Res.	Res.	IE	RNGEN	Res.	MODE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0

位 31:16	**Res**: 必须保持复位值
位 15:9	**OP_CYCLE**：产生一个随机数的累加时钟周期 初始值：48 个周期 不建议小于 48 个周期，最小可以 33 个周期 注：用于配置 LFSR 的运行周期，按理论运行周期越长，数据随机性越好。
位 3	**IE**：中断使能 (Interrupt enable) 0：禁止 RNG 中断 1：使能 RNG 中断。只要 RNG_SR 寄存器中 DRDY=1 或 SEIS=1，就会挂起中断。
位 2	**RNGEN**：随机数发生器使能 (Random number generator enable) 0：禁止随机数发生器,模拟随机源电源关闭，逻辑时钟关闭。 1：使能随机数发生器。
位 1	**Res**: 必须保持复位值
位 0	**MODE**：工作模式配置 0：生成一个随机数后，关闭模拟模块，等待下次启动。 1：生成一个随机数后，不关闭模拟模块，等待下次启动。 注：如果随机数产生要快速度，建议配置为 1，但会带来写些功耗。

31.6.2 RNG 状态寄存器 (RNG_SR)

偏移地址：0x04

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEIS	Res.	Res.	Res.	SECS	Res.	DRDY
rw	rw	rw	rw	rw	rw	rw	rw	rw	rcw0	rcw0	rw	rw	r	r	r
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位 31:7	**Res**：必须保持复位值
位 6	**SEIS**：种子错误中断状态 (Seed error interrupt status) 此位与 SECS 同时设置，通过向其写入 0 来清零，写 1 无效。 0：未检测到错误序列 1：检测到以下错误序列之一： – 超过 64 个连续位具有相同值（0 或 1） – 超过 32 个连续交替的 0 和 1 (0101010101...01) 如果 RNG_CR 寄存器中 IE = 1，则会挂起中断。
位 4:3	**Res**: 必须保持复位值
位 2	**SECS**：种子错误当前状态 (Seed error current status) 0：目前未检测到错误序列。如果 SEIS 位置 1，则意味着已检测到错误序列并已恢复正常。 1：检测到以下错误序列之一： – 超过 64 个连续位具有相同值（0 或 1） – 超过 32 个连续交替的 0 和 1 (0101010101...01)
位 0	**DRDY**：数据就绪 (Data ready) 0：RNG_DR 寄存器尚未有效，无可用随机数据 1：RNG_DR 寄存器包含有效随机数据 *注： 如果 RNG_CR 寄存器中 IE = 1，则会挂起中断。* *读取 RNG_DR 寄存器后，此位恢复到 0，直到计算出新的有效值。*

31.6.3 RNG 数据寄存器 (RNG_DR)

偏移地址：0x08

复位值：0x0000 0000

RNG_DR 寄存器是只读寄存器，在读取时提供 32 位随机数值。读取后，此寄存器在最多 40 个 RNG_CLK 时钟周期后，提供新的随机数值。在读取 RNDATA 值之前，软件必须检查 DRDY 位是否已置 1。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RNDATA															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

位 31:4	**RNDATA**：随机数据 (Random data) 32 位随机数据。 DRDY=1 时，更新该数据

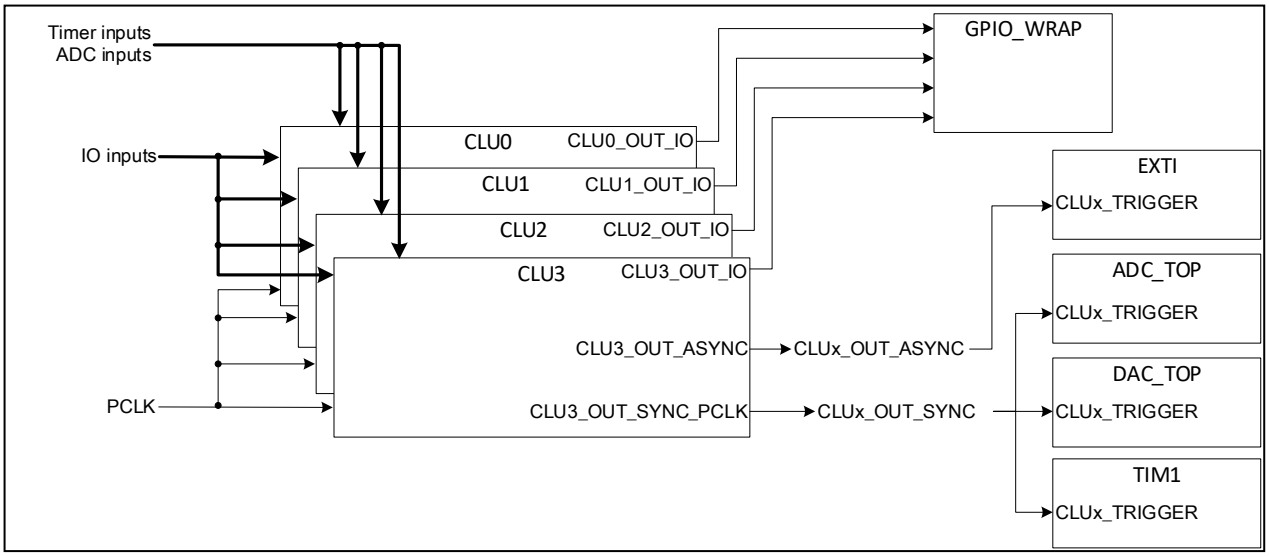
32 可配置逻辑单元（CLU）

32.1 简介

可配置逻辑（CL）模块提供了多个用户编程的数字逻辑块，这些块无需 CPU 干预即可运行。它由四个专用的独立可配置逻辑单元（CLU）组成，它们支持用户可编程的异步和同步布尔逻辑运算。大量内部和外部信号可用作每个 CLU 的输入，并且输出可路由到端口 I / O 引脚，或直接路由至选择外围输入。

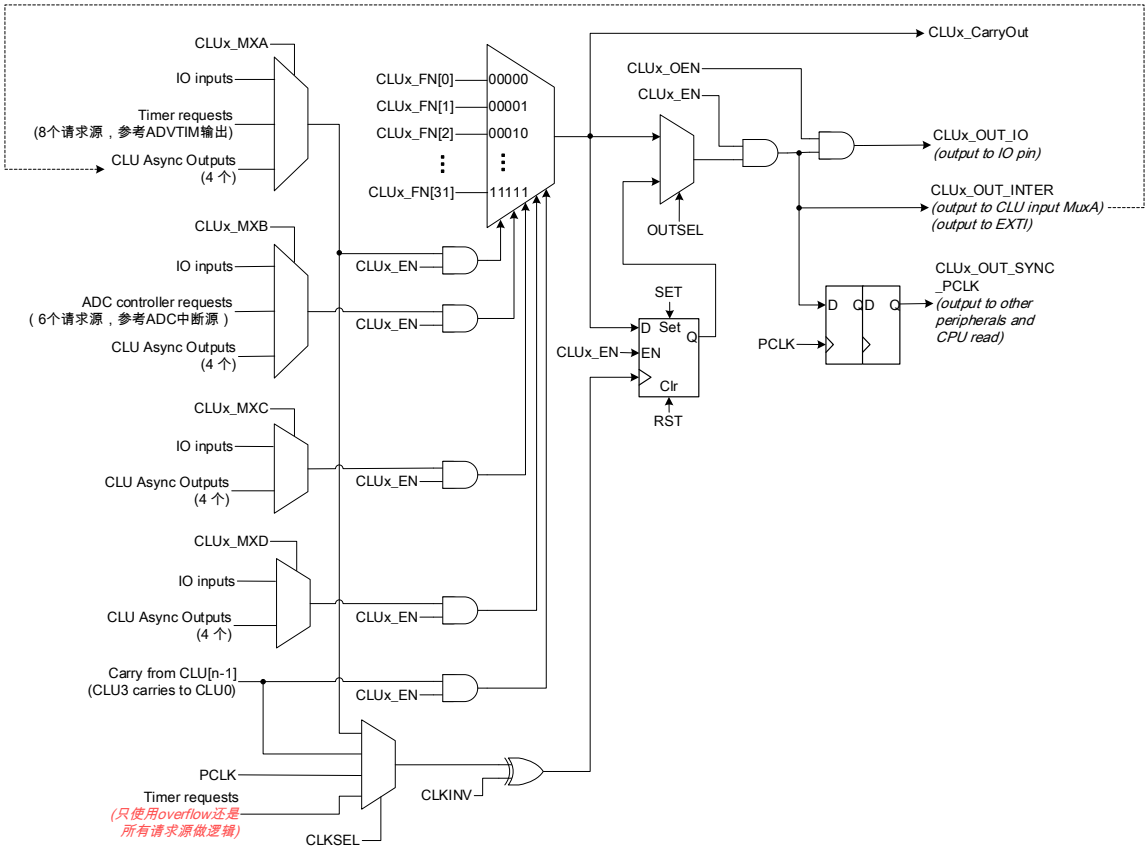
注: 如图：CLU 送给 GPIO 的信号没有经过 PCLK 同步

图 32-1 可配置逻辑概括框图



CLU 经过 PCLK 同步后的信号会 trigger ADC、DAC 和 timer1，同时也会送给 EXTI 产生中断

图 32-2 内部逻辑框图



特征

- 四个可配置逻辑单元（CLU），具有直接引脚和内部逻辑连接
- 每个单元支持 256 个不同的组合逻辑功能（AND，OR，XOR，多路复用等），并包括一个用于同步操作的时钟触发器
- 设备可以同步或异步操作
- 可以级联在一起以执行更复杂的逻辑功能
- 可以与 UART 和 SPI 等串行外设或定时器和 PCA 通道等定时外设配合使用
- 可用于同步和触发多个片上资源（ADC，DAC，定时器等）
- 异步输出可用于从低功耗状态唤醒

32.2 功能说明

32.2.1 配置步骤

初始化步骤：

1、 在 CLUx_MX 中选择 LUT 的 A、B、C 和 D 输入；
2、 使用 CLUnFN 选择 LUT 功能；
3、 通过 CLUx_CTL 配置 CLU；
4、 如果为 CLU 选择了 D 触发器输出(OUTSEL = 1)，建议也将 RST=1 设置为了将 D 触发器输出重置为 0；
5、 通过将 CLU_CTL 中的 CLUx_EN 位置 1 使能 CLU，固件可以通过在 CLU_CTL 中设置多个位来同事启用多个 CLU；

6、 如果需要直接引脚输出，则固件可以通过将 CLUx_CTL 中的 OEN 位置 1 来使能输出。

32.2.2 输入多路复用器选择

每个 CLU 有四个主逻辑输入（A、B、C 和 D）和进位输入（Carry_In）。A、B、C 和 D 输入由 CLUnMX 寄存器中的 MXA、MXB、MXC 和 MXD 字段选择，并且可以是许多不同的内部和外部信号之一。 选择另一个 CLU 输出作为输入时，将使用该 CLU 的异步输出，从而可以实现更复杂的布尔逻辑功能。

进位输入 C 是前一个 CLU 的 LUT 输出。例如，CLU1 上的进位输入是 CLU0 的 LUT 输出。 CLU0 的进位输入是 CLU3 的 LUT 输出。

CLU 输入的引脚输入未与 SYSCLK 同步。 CLU 输入的其他内部外设（例如定时器）也已进行 SYSCLK 同步，因为这些外设已进行 SYSCLK 同步。捕获模式下的定时器的脉冲宽度至少应为 1 SYSCLK，以确保捕获沿。但是，仍然有可能捕获到更窄的脉冲。因此，包含 CLU 的固件不能依赖于计时器不捕获小于 1 SYSCLK 周期宽的脉冲。

表 32-1 CLUx_MXA 输入选择

CLUx_MXA[3:0]	CLU0	CLU1	CLU2	CLU3
0000	CLU0_OUT_INTER	CLU0_OUT_INTER	CLU0_OUT_INTER	CLU0_OUT_INTER
0001	CLU1_OUT_INTER	CLU1_OUT_INTER	CLU1_OUT_INTER	CLU1_OUT_INTER
0010	CLU2_OUT_INTER	CLU2_OUT_INTER	CLU2_OUT_INTER	CLU2_OUT_INTER
0011	CLU3_OUT_INTER	CLU3_OUT_INTER	CLU3_OUT_INTER	CLU3_OUT_INTER
0100	TIM1_BRK	TIM1_BRK	TIM1_BRK	TIM1_BRK
0101	TIM1_CC1	TIM1_CC1	TIM1_CC1	TIM1_CC1
0110	TIM1_CC2	TIM1_CC2	TIM1_CC2	TIM1_CC2
0111	TIM1_CC3	TIM1_CC3	TIM1_CC3	TIM1_CC3
1000	TIM1_CC4	TIM1_CC4	TIM1_CC4	TIM1_CC4
1001	TIM1_COM	TIM1_COM	TIM1_COM	TIM1_COM
1010	TIM1_TRIG	TIM1_TRIG	TIM1_TRIG	TIM1_TRIG
1011	TIM1_UP	TIM1_UP	TIM1_UP	TIM1_UP
1100	PF0	PF0	PF1	PA0
1101	PF1	PA4	PA4	PA6
1110	PA0	PA8	PA14	PA8
1111	PA1	PB3	PB9	PA14

表 32-2 CLUx_MXB 输入选择

CLUx_MXB[3:0]	CLU0	CLU1	CLU2	CLU3
0000	CLU0_OUT_INTER	CLU0_OUT_INTER	CLU0_OUT_INTER	CLU0_OUT_INTER
0001	CLU1_OUT_INTER	CLU1_OUT_INTER	CLU1_OUT_INTER	CLU1_OUT_INTER
0010	CLU2_OUT_INTER	CLU2_OUT_INTER	CLU2_OUT_INTER	CLU2_OUT_INTER
0011	CLU3_OUT_INTER	CLU3_OUT_INTER	CLU3_OUT_INTER	CLU3_OUT_INTER
0100	ADC1_RDY	ADC1_RDY	ADC1_RDY	ADC1_RDY
0101	ADC1_EOSMP	ADC1_EOSMP	ADC1_EOSMP	ADC1_EOSMP
0110	ADC1_EOC	ADC1_EOC	ADC1_EOC	ADC1_EOC
0111	ADC1_EOSEQ	ADC1_EOSEQ	ADC1_EOSEQ	ADC1_EOSEQ
1000	ADC1_OVR	ADC1_OVR	ADC1_OVR	ADC1_OVR
1001	ADC1_AWD	ADC1_AWD	ADC1_AWD	ADC1_AWD

1010	PA2	PF1	PA0	PB7
1011	PA3	PA5	PA7	PA4
1100	PA4	PA9	PA10	PB2
1101	PA5	PB4	PB4	PA12
1110	PA6	PA6	PF0	PB5
1111	PA7	PA10	PA3	PF0

表 32-3 CLUx_MXC 输入选择

CLUx_MXC[3:0]	CLU0	CLU1	CLU2	CLU3
0000	CLU0_OUT_INTER	CLU0_OUT_INTER	CLU0_OUT_INTER	CLU0_OUT_INTER
0001	CLU1_OUT_INTER	CLU1_OUT_INTER	CLU1_OUT_INTER	CLU1_OUT_INTER
0010	CLU2_OUT_INTER	CLU2_OUT_INTER	CLU2_OUT_INTER	CLU2_OUT_INTER
0011	CLU3_OUT_INTER	CLU3_OUT_INTER	CLU3_OUT_INTER	CLU3_OUT_INTER
0100	PB0	PB5	PB2	PA2
0101	PB1	PA7	PA13	PB0
0110	PB2	PA11	PB7	PA10
0111	PA8	PB6	PA6	PB3
1000	PA9	PA0	PA9	PB8
1001	PA10	PB0	PB3	PA1
1010	PA11	PA12	PA2	PA7
1011	PA12	PB7	PB1	PA9
1100	PA13	PA1	PA12	PA15
1101	PA14	PB1	PB6	PB9
1110	Reserved	Reserved	Reserved	Reserved
1111	Reserved	Reserved	Reserved	Reserved

表 32-4 CLUx_MXD 输入选择

CLUx_MXD[3:0]	CLU0	CLU1	CLU2	CLU3
0000	CLU0_OUT_INTER	CLU0_OUT_INTER	CLU0_OUT_INTER	CLU0_OUT_INTER
0001	CLU1_OUT_INTER	CLU1_OUT_INTER	CLU1_OUT_INTER	CLU1_OUT_INTER
0010	CLU2_OUT_INTER	CLU2_OUT_INTER	CLU2_OUT_INTER	CLU2_OUT_INTER
0011	CLU3_OUT_INTER	CLU3_OUT_INTER	CLU3_OUT_INTER	CLU3_OUT_INTER
0100	PA15	PA13	PA5	PA5
0101	PB3	PB9	PA8	PA13
0110	PB4	PA2	PA15	PB6
0111	PB5	PB2	PB8	PF1
1000	PB6	PA14	PA1	PA3
1001	PB7	PB8	PB0	PB1
1010	PB9	PA3	PA11	PA11
1011	PB8	PA15	PB5	PB4
1100	Reserved	Reserved	Reserved	Reserved
1101	Reserved	Reserved	Reserved	Reserved

1110	Reserved	Reserved	Reserved	Reserved
1111	Reserved	Reserved	Reserved	Reserved

32.2.3 CLU 输出配置

每个 CLU 都向系统提供一个异步和一个同步（同步到 SYSCLK）输出。固件可随时通过读取 CLOUT0 寄存器来读取同步输出。 CLU 输出可以直接从 LUT 导出，也可以从 CLUnCF 中 OUTSEL 位控制的锁存 D 型触发器输出导出。当 CLU 被禁用（CLEN0 中的 CnEN 为 0）时，其两个输出都将保持逻辑 0。

D 触发器时钟可以由四个源之一配置，由 CLUnCF 中的 CLKSEL 字段选择。可以使用 CLKINV 位将触发器时钟反相。每个 CLU 具有以下用于为其触发器计时的选项：

- CARRY_IN
- MXA_INPUT
- SYSCLK
- TIMER

32.2.4 LUT 配置

每个 CLU 中的布尔逻辑功能由 LUT 确定，可以通过对寄存器 CLUnFN 中的 FNSEL 字段进行编程来更改。LUT 被实现为 8 输入多路复用器。FNSEL 的位映射到 8 个多路复用器输入，并且 LUT 的输出通过 A，B，C，D 和 CARRY_IN 输入的组合进行选择。

表 32-5 LUT Truth Table

A INPUT	B INPUT	C INPUT	D INPUT	CARRY_IN	LUT OUTPUT
0	0	0	0	0	FNSEL.0
0	0	0	0	1	FNSEL.1
0	0	0	1	0	FNSEL.2
0	0	0	1	1	FNSEL.3
0	0	1	0	0	FNSEL.4
0	0	1	0	1	FNSEL.5
0	0	1	1	0	FNSEL.6
0	0	1	1	1	FNSEL.7
0	1	0	0	0	FNSEL.8
0	1	0	0	1	FNSEL.9
0	1	0	1	0	FNSEL.10
0	1	0	1	1	FNSEL.11
0	1	1	0	0	FNSEL.12
0	1	1	0	1	FNSEL.13
0	1	1	1	0	FNSEL.14
0	1	1	1	1	FNSEL.15
1	0	0	0	0	FNSEL.16
1	0	0	0	1	FNSEL.17
1	0	0	1	0	FNSEL.18
1	0	0	1	1	FNSEL.19
1	0	1	0	0	FNSEL.20
1	0	1	0	1	FNSEL.21
1	0	1	1	0	FNSEL.22

1	0	1	1	1	FNSEL.23
1	1	0	0	0	FNSEL.24
1	1	0	0	1	FNSEL.25
1	1	0	1	0	FNSEL.26
1	1	0	1	1	FNSEL.27
1	1	1	0	0	FNSEL.28
1	1	1	0	1	FNSEL.29
1	1	1	1	0	FNSEL.30
1	1	1	1	1	FNSEL.31

32.3 CLU 寄存器

32.3.1 可配置逻辑单元控制寄存器（CLU_CTL）

偏移地址： 0x00

复位值： 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLU3_EN	CLU2_EN	CLU1_EN	CLU0_EN
												rw	rw	rw	rw
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位 31:4	保留，必须保持为复位值
位 3	CLU3_EN: 可配置逻辑单元 3 使能位（Configurable logic unit 3 enable bit） 0: CLU3 失能 1: CLU3 使能
位 2	CLU2_EN: 可配置逻辑单元 2 使能位（Configurable logic unit 2 enable bit） 0: CLU2 失能 1: CLU2 使能
位 1	CLU1_EN: 可配置逻辑单元 1 使能位（Configurable logic unit 1 enable bit） 0: CLU1 失能 1: CLU1 使能
位 0	CLU0_EN: 可配置逻辑单元 0 使能位（Configurable logic unit 0 enable bit） 0: CLU0 失能 1: CLU0 使能

32.3.2 可配置逻辑单元同步输出（CLU_OUT_SYNC）

偏移地址： 0x04

复位值： 0x0000 0000

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

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLU3_OUT_SYNC	CLU2_OUT_SYNC	CLU1_OUT_SYNC	CLU0_OUT_SYNC
												r	r	r	r
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位 31:4	保留，必须保持为复位值
位 3	CLU3_OUT_SYNC: 可配置逻辑单元 3 同步输出位（Configurable logic unit 3 synchronous output bit） 输出的状态是与 SYSCLK 同步的
位 2	CLU2_OUT_SYNC: 可配置逻辑单元 2 同步输出位（Configurable logic unit 2 enable bit） 输出的状态是与 SYSCLK 同步的
位 1	CLU1_OUT_SYNC: 可配置逻辑单元 1 同步输出位（Configurable logic unit 1 enable bit） 输出的状态是与 SYSCLK 同步的
位 0	CLU0_OUT_SYNC: 可配置逻辑单元 0 同步输出位（Configurable logic unit 0 enable bit） 输出的状态是与 SYSCLK 同步的

32.3.3 可配置逻辑单元 0 多路复用通道选择（CLU0_MX）

偏移地址： 0x10

复位值： 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CLU0_MXD				CLU0_MXC				CLU0_MXB				CLU0_MXA			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位 31:16	保留，必须保持为复位值
位 15:12	CLU0_MXD: 选择到 CLU0 多路复用通道 D 的输入源（Select the input source to CLU0 multiplex channel D） 参考 CLUx_MXD 表
位 11:8	CLU0_MXC: 选择到 CLU0 多路复用通道 C 的输入源（Select the input source to CLU0 multiplex channel C） 参考 CLUx_MXC 表
位 7:4	CLU0_MXB: 选择到 CLU0 多路复用通道 B 的输入源（Select the input source to CLU0 multiplex channel B） 参考 CLUx_MXB 表
位 3:0	CLU0_MXA: 选择到 CLU0 多路复用通道 A 的输入源（Select the input source to CLU0 multiplex channel A） 参考 CLUx_MXA 表

32.3.4 可配置逻辑单元 0 功能选择（CLU0_FN）

偏移地址： 0x14

复位值： 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CLU0_FN[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CLU0_FN[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位 31:0	CLU0_FN: 针对 CLU0 的功能选择 （Function select for CLU0） 设定值参考 FN 表

32.3.5 可配置逻辑单元 0 控制寄存器（CLU0_CTL）

偏移地址： 0x18

复位值： 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLU0_OUTSEL	CLU0_OEN	Res.	CLU0_SET	CLU0_RST	CLU0_CLKINV	CLU0_CLKSEL	
								rw	rw		rw	rw	rw	rw	rw
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位 31:8	保留，必须保持为复位值
位 7	CLU0_OUTSEL: CLU0 输出选择（CLU0 Output Select） 0: 选择 DFF 作为 CLU0 的输出 1: 选择功能多路器作为 CLU0 的输出
位 6	CLU0_OEN: 使能 CLU0 输出到 IO 口（CLU Port Output Enable） 0: 失能 1: 使能
位 5	保留，必须保持为复位值
位 4	CLU0_SET: 设置 CLU0 输出（Set CLU0） 0: 不影响 1: 设置 DFF 输出为 1
位 3	CLU0_RST: 复位 CLU0 输出（Reset CLU0） 0: 不影响 1: 设置 DFF 输出为 0
位 2	CLU0_CLKINV: CLU0 D 触发器时钟翻转（CLU0 D flip-flop Clock Invert） 0: 不影响 1: 翻转时钟信号
位 1:0	CLU0_CLKSEL: CLU0 D 触发器时钟选择（CLU0 D flip-flop Clock Selection） 00: 选择进位输入作为时钟信号 01: 选择 MXA 输出作为时钟信号 10: 选择 SYSCLK 作为时钟信号 11: 选择定时器作为时钟信号

32.3.6 可配置逻辑单元 1 多路复用通道选择（CLU1_MX）

偏移地址： 0x20

复位值： 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CLU1_MXD				CLU1_MXC				CLU1_MXB				CLU1_MXA			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位 31:16	保留，必须保持为复位值
位 15:12	CLU1_MXD: 选择到 CLU1 多路复用通道 D 的输入源（Select the input source to CLU1 multiplex channel D） 参考 CLUx_MXD 表
位 11:8	CLU1_MXC: 选择到 CLU1 多路复用通道 C 的输入源（Select the input source to CLU1 multiplex channel C） 参考 CLUx_MXC 表
位 7:4	CLU1_MXB: 选择到 CLU1 多路复用通道 B 的输入源（Select the input source to CLU1 multiplex channel B） 参考 CLUx_MXB 表
位 3:0	CLU1_MXA: 选择到 CLU1 多路复用通道 A 的输入源（Select the input source to CLU1 multiplex channel A） 参考 CLUx_MXA 表

32.3.7 可配置逻辑单元 1 功能选择（CLU1_FN）

偏移地址： 0x24

复位值： 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CLU1_FN[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CLU1_FN[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位 31:0	CLU1_FN: 针对 CLU1 的功能选择 （Function select for CLU1） 设定值参考 FN 表

32.3.8 可配置逻辑单元 1 控制寄存器（CLU1_CTL）

偏移地址： 0x28

复位值： 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLU1_OUTSEL	CLU1_OEN	Res.	CLU1_SET	CLU1_RST	CLU1_CLKINV	CLU1_CLKSEL	
								rw	rw		rw	rw	rw	rw	rw
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位 31:8	保留，必须保持为复位值
位 7	CLU1_OUTSEL: CLU1 输出选择（CLU1 Output Select） 0: 选择 DFF 作为 CLU1 的输出 1: 选择功能多路器作为 CLU1 的输出
位 6	CLU1_OEN: 使能 CLU1 输出到 IO 口（CLU Port Output Enable） 0: 失能 1: 使能
位 5	保留，必须保持为复位值
位 4	CLU1_SET: 设置 CLU1 输出（Set CLU1） 0: 不影响 1: 设置 DFF 输出为 1
位 3	CLU1_RST: 复位 CLU1 输出（Reset CLU1） 0: 不影响 1: 设置 DFF 输出为 0
位 2	CLU1_CLKINV: CLU1 D 触发器时钟翻转（CLU1 D flip-flop Clock Invert） 0: 不影响 1: 翻转时钟信号
位 1:0	CLU1_CLKSEL: CLU1 D 触发器时钟选择（CLU1 D flip-flop Clock Selection） 00: 选择进位输入作为时钟信号 01: 选择 MXA 输出作为时钟信号 10: 选择 SYSCLK 作为时钟信号 11: 选择定时器作为时钟信号

32.3.9 可配置逻辑单元 2 多路复用通道选择（CLU2_MX）

偏移地址： 0x30

复位值： 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CLU2_MXD				CLU2_MXC				CLU2_MXB				CLU2_MXA			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位 31:16	保留，必须保持为复位值
位 15:12	CLU2_MXD: 选择到 CLU2 多路复用通道 D 的输入源（Select the input source to CLU2 multiplex channel D） 参考 CLUx_MXD 表

位 11:8	CLU2_MXC: 选择到 CLU2 多路复用通道 C 的输入源（Select the input source to CLU2 multiplex channel C） 参考 CLUx_MXC 表
位 7:4	CLU2_MXB: 选择到 CLU2 多路复用通道 B 的输入源（Select the input source to CLU2 multiplex channel B） 参考 CLUx_MXB 表
位 3:0	CLU2_MXA: 选择到 CLU2 多路复用通道 A 的输入源（Select the input source to CLU2 multiplex channel A） 参考 CLUx_MXA 表

32.3.10 可配置逻辑单元 2 功能选择（CLU2_FN）

偏移地址： 0x34

复位值： 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CLU2_FN[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CLU2_FN[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位 31:0	CLU2_FN: 针对 CLU2 的功能选择 （Function select for CLU2） 设定值参考 FN 表

32.3.11 可配置逻辑单元 2 控制寄存器（CLU2_CTL）

偏移地址： 0x38

复位值： 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLU2_OUTSEL	CLU2_OEN	Res.	CLU2_SET	CLU2_RST	CLU2_CLKINV	CLU2_CLKSEL	
								rw	rw		rw	rw	rw	rw	rw
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位 31:8	保留，必须保持为复位值
位 7	CLU2_OUTSEL: CLU2 输出选择（CLU2 Output Select） 0: 选择 DFF 作为 CLU2 的输出 1: 选择功能多路器作为 CLU2 的输出
位 6	CLU2_OEN: 使能 CLU2 输出到 IO 口（CLU2 Port Output Enable） 0: 失能 1: 使能
位 5	保留，必须保持为复位值

位 4	CLU2_SET: 设置 CLU2 输出（Set CLU2） 0: 不影响 1: 设置 DFF 输出为 1
位 3	CLU2_RST: 复位 CLU2 输出（Reset CLU2） 0: 不影响 1: 设置 DFF 输出为 0
位 2	CLU2_CLKINV: CLU2 D 触发器时钟翻转（CLU2 D flip-flop Clock Invert） 0: 不影响 1: 翻转时钟信号
位 1:0	CLU2_CLKSEL: CLU2 D 触发器时钟选择（CLU2 D flip-flop Clock Selection） 00: 选择进位输入作为时钟信号 01: 选择 MXA 输出作为时钟信号 10: 选择 SYSCLK 作为时钟信号 11: 选择定时器作为时钟信号

32.3.12 可配置逻辑单元 3 多路复用通道选择（CLU3_MX）

偏移地址： 0x40

复位值： 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CLU3_MXD				CLU3_MXC				CLU3_MXB				CLU3_MXA			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位 31:16	保留，必须保持为复位值
位 15:12	CLU3_MXD: 选择到 CLU3 多路复用通道 D 的输入源（Select the input source to CLU3 multiplex channel D） 参考 CLUx_MXD 表
位 11:8	CLU3_MXC: 选择到 CLU3 多路复用通道 C 的输入源（Select the input source to CLU3 multiplex channel C） 参考 CLUx_MXC 表
位 7:4	CLU3_MXB: 选择到 CLU3 多路复用通道 B 的输入源（Select the input source to CLU3 multiplex channel B） 参考 CLUx_MXB 表
位 3:0	CLU3_MXA: 选择到 CLU3 多路复用通道 A 的输入源（Select the input source to CLU3 multiplex channel A） 参考 CLUx_MXA 表

32.3.13 可配置逻辑单元 3 功能选择（CLU3_FN）

偏移地址： 0x44

复位值： 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CLU3_FN[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CLU3_FN[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位 31:0	CLU3_FN: 针对 CLU3 的功能选择 （Function select for CLU3） 设定值参考 FN 表

32.3.14 可配置逻辑单元 3 控制寄存器（CLU3_CTL）

偏移地址： 0x48

复位值： 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLU3_OUTSEL	CLU3_OEN	Res.	CLU3_SET	CLU3_RST	CLU3_CLKINV	CLU3_CLKSEL	
								rw	rw		rw	rw	rw	rw	rw
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位 31:8	保留，必须保持为复位值
位 7	CLU3_OUTSEL: CLU3 输出选择（CLU3 Output Select） 0: 选择 DFF 作为 CLU3 的输出 1: 选择功能多路器作为 CLU3 的输出
位 6	CLU3_OEN: 使能 CLU3 输出到 IO 口（CLU Port Output Enable） 0: 失能 1: 使能
位 5	保留，必须保持为复位值
位 4	CLU3_SET: 设置 CLU3 输出（Set CLU3） 0: 不影响 1: 设置 DFF 输出为 1
位 3	CLU3_RST: 复位 CLU3 输出（Reset CLU3） 0: 不影响 1: 设置 DFF 输出为 0
位 2	CLU3_CLKINV: CLU3 D 触发器时钟翻转（CLU3 D flip-flop Clock Invert） 0: 不影响 1: 翻转时钟信号
位 1:0	CLU3_CLKSEL: CLU3 D 触发器时钟选择（CLU3 D flip-flop Clock Selection） 00: 选择进位输入作为时钟信号 01: 选择 MXA 输出作为时钟信号 10: 选择 SYSCLK 作为时钟信号 11: 选择定时器作为时钟信号

33 串行外设接口 (SPI)

33.1 简介

SPI/I²S 接口可用于使用 SPI 协议或 I2S 音频协议与外部器件进行通信。SPI 或 I2S 模式可通过软件进行选择。器件复位后默认选择 SPI 模式。

串行外设接口 (SPI) 协议支持与外部器件进行半双工、全双工和单工同步串行通信。该接口可配置为主模式，在这种情况下，它可为外部从器件提供通信时钟 (SCK)。该接口还能够在多主模式配置下工作。

集成电路内置音频总线 (I2S) 也是同步串行通信接口。它能够在从模式或主模式下作为接收器或发送器工作。

它可满足四种不同音频标准的要求，包括 Philips I2S 标准、MSB 和 LSB 对齐标准以及 PCM 标准。

33.2 SPI 主要特性

- 主模式或从模式操作
- 基于三条线的全双工同步传输
- 基于双线的半双工同步传输，其中一条可作为双向数据线
- 基于双线的单工同步传输，其中一条可作为单向数据线
- 8 位到 16 位传输帧格式选择
- 多主模式功能
- 8 个主模式波特率预分频器，可达 fPCLK/2。
- 从模式频率可达 fPCLK/2。
- 对于主模式和从模式都可通过硬件或软件进行 NSS 管理：动态切换主/从操作
- 可编程的时钟极性和相位
- 可编程的数据顺序，最先移位 MSB 或 LSB
- 可触发中断的专用发送和接收标志
- SPI 总线忙状态标志
- 支持 SPI Motorola 模式
- 用于确保可靠通信的硬件 CRC 功能：
  - 在发送模式下可将 CRC 值作为最后一个字节发送
  - 根据收到的最后一个字节自动进行 CRC 错误校验
- 可触发中断的主模式故障和上溢标志
- CRC 错误标志
- 具有 DMA 功能的单字节/字收发缓冲器：发送和接收请求

33.3 SPI 扩展特性

- 支持 SPI TI 模式

33.4 I2S 功能

- 半双工通信（仅作为发送器或接收器）

- 主模式或从模式操作
- 8 位可编程线性预分频器，可实现精确的音频采样频率（从 8 kHz 到 192 kHz）
- 数据格式可以是 16 位、 24 位或 32 位
- 数据包帧由音频通道固定为 16 位（可容纳 16 位数据帧）或 32 位（可容纳 16 位、24 位、32 位数据帧）
- 可编程的时钟极性（就绪时的电平状态）
- 从发送模式下的下溢标志、接收模式下的上溢标志（主模式和从模式），以及接收和发送模式下的帧错误标志（仅从模式）
- 发送和接收使用同一个 16 位数据寄存器
- 支持的 I2S 协议：
  - 2S Philips 标准
  - MSB 对齐标准（左对齐）
  - LSB 对齐标准（右对齐）
  - PCM 标准（在 16 位通道帧或扩展为 32 位通道帧的 16 位数据帧上进行短帧和长帧同步）
- 数据方向始终为 MSB 在前
- 用于发送和接收的 DMA 功能（16 位宽）
- 可输出主时钟以驱动外部音频元件。比率固定为 256 × Fs（其中 Fs 为音频采样频率）

33.5 SPI/I2S 实现

本手册介绍了 SPI1 和 SPI2 中实现的所有特性。

表 33-1 **HK32L08x SPI** 实现

SPI 特性(1)	SPI1	SPI2
硬件 CRC 计算	X	X
I2S 模式	-	X
TI 模式	X	X

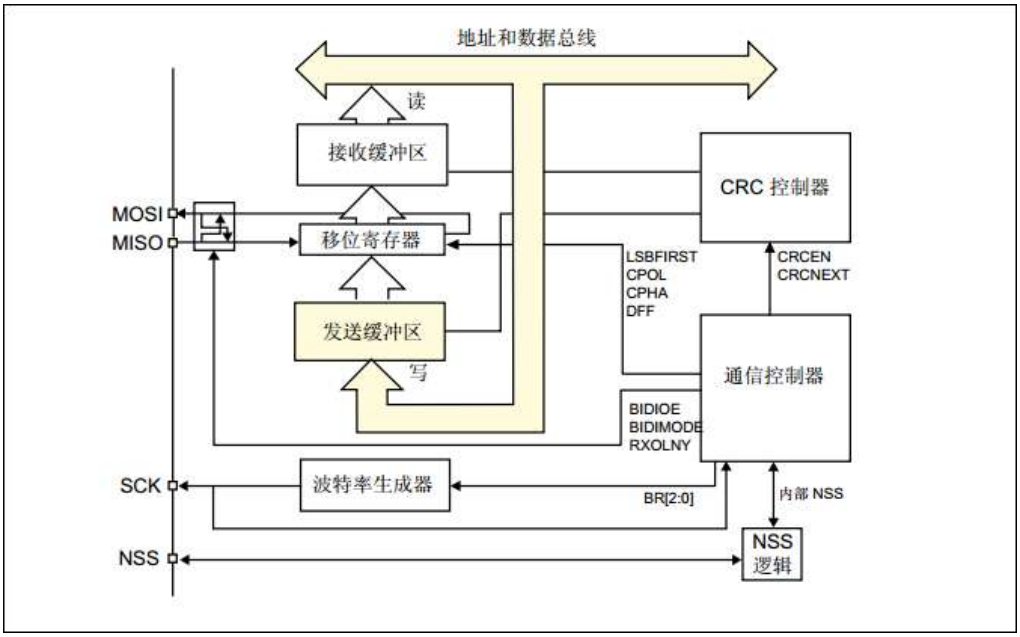
1. X = 支持。

33.6 SPI 功能说明

概述

SPI 支持在 MCU 与外部器件之间进行同步串行通信。应用软件可通过轮询状态标志或使用专用 SPI 中断对通信进行管理。 SPI 的主要组件及其交互方式如以下框图所示图 33-1。

图 33-1 **SPI** 框图



四个 I/O 引脚专用于与外部器件进行 SPI 通信。

- **MISO**： 主输入 / 从输出数据。通常情况下，此引脚用于在从模式下发送数据和在主模式下接收数据。
- **MOSI**：主输出 / 从输入数据。通常情况下，此引脚用于在主模式下发送数据和在从模式下接收数据。
- **SCK**： SPI 主器件的串行时钟输出引脚以及 SPI 从器件的串行时钟输入引脚。
- **NSS**： 从器件选择引脚。根据 SPI 和 NSS 设置，该引脚可用于：

– 选择单个从器件以进行通信
– 同步数据帧或
– 检测多个主器件之间是否存在冲突

详细信息，请参见多主器件通信。

SPI 总线支持一个主器件与一个或多个从器件之间进行通信。该总线至少由两条线构成——一条用于时钟信号，另一条用于同步数据传输。其它信号可以根据 SPI 节点间的数据交换及其从器件选择信号管理进行添加。

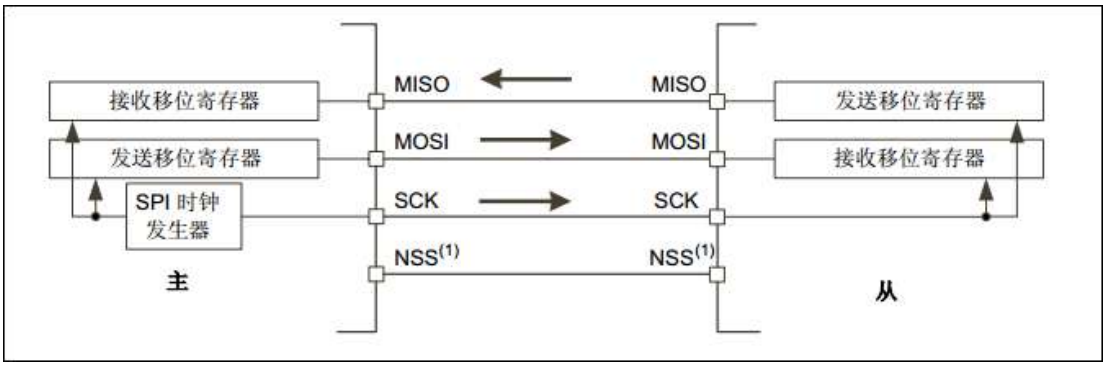
33.6.1 一个主器件和一个从器件之间的通信

SPI 支持 MCU 基于目标器件和应用要求使用不同的配置进行通信。这些配置使用 2 条或 3 条线（通过软件 NSS 管理），也可以使用 3 条或 4 条线（通过硬件 NSS 管理）。通信始终由主器件发起。

全双工通信

默认情况下， SPI 配置为进行全双工通信。在这种配置下，主器件和从器件的移位寄存器通过 MOSI 和 MISO 引脚之间的两条单向线连接。在 SPI 通信过程中，数据随主器件提供的 SCK 时钟边沿同步移位。主器件通过 MOSI 线将待发送的数据发送给从器件，通过 MISO 线从从器件接收数据。当数据帧传输完成时（所有位均移出），主器件和从器件之间即完成信息交换。

图 33-2 全双工单个主器件/单个从器件应用

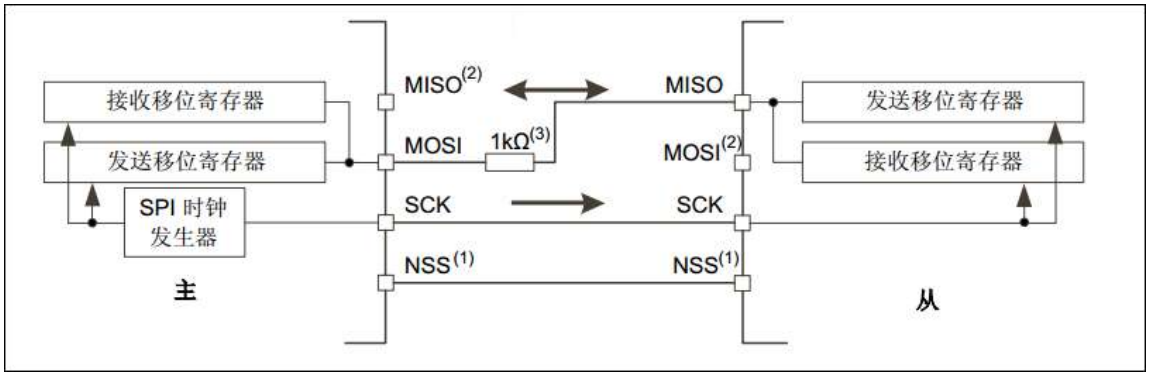


1. NSS 引脚可用于在主器件和从器件之间提供硬件控制流。外设也可选择不使用这些引脚。之后，必须在内部为主器件和从器件处理硬件控制流。有关更多详细信息，请参见从器件选择 *(NSS)* 引脚管理。

半双工通信

通过将 SPIx_CR1 寄存器的 BIDIMODE 位置 1， SPI 可采用半双工模式进行通信。在这种配置下，使用一条交叉连接线将主器件和从器件的移位寄存器连接起来。在此通信过程中，数据随 SCK 时钟边沿在移位寄存器之间进行移位，传输方向由主器件和从器件通过各自 SPIx_CR1 寄存器中的 BDIOE 位进行选择。在这种配置下，主器件的 MISO 引脚和从器件的 MOSI 引脚空闲，可在其它应用中用作 GPIO。

图 33-3 半双工单个主器件/单个从器件应用



1. NSS 引脚可用于在主器件和从器件之间提供硬件控制流。外设也可选择不使用这些引脚。之后，必须在内部为主器件和从器件处理硬件控制流。有关更多详细信息，请参见从器件选择 *(NSS)* 引脚管理。
2. 在这种配置下，主器件的 MISO 引脚和从器件的 MOSI 引脚可用作 GPIO。
3. 当以双向模式工作的两个节点间的通信方向不是同步变化时，会出现临界情况，新发送器访问共用数据线，而前一个发送器仍保持线路上的相反值（值取决于 SPI 配置和通信数据）。两个节点会出现冲突，在共用线上短暂提供相反的输出电平，直到下一个节点也相应地改变其方向设置。建议此模式下在 MISO 和 MOSI 引脚之间插入串行电阻以在这种情况下保护输出并限制电流在二者之间流过。

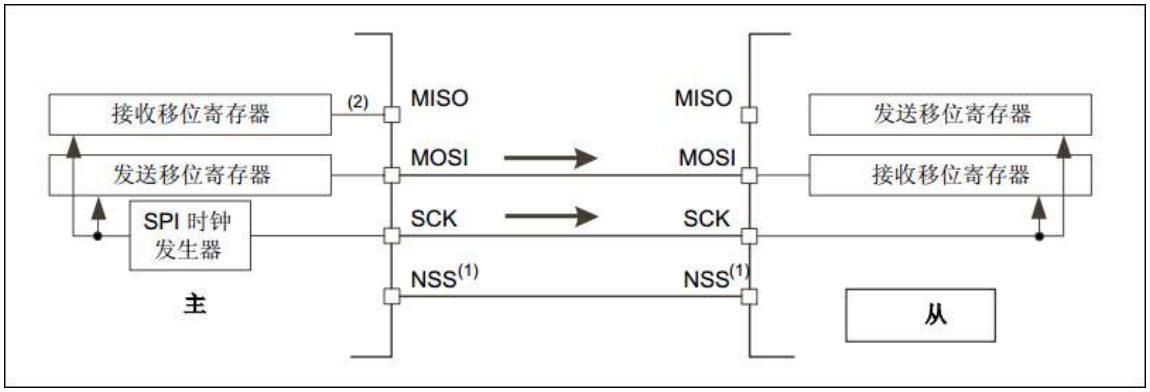
单工通信

通过 SPIx_CR2 寄存器中的 RXONLY 位将 SPI 设置为只发送模式或只接收模式，可使 SPI 以单工模式进行通信。在这种配置下，仅使用一条线在主器件和从器件的移位寄存器之间进行传输。其余 MISO 和 MOSI 引脚对不用于通信，可用作标准 GPIO。

- 只发送模式 **(RXONLY=0)**： 配置设置与全双工设置相同。应用必须忽略在未使用的输入引脚上捕获的信息。该引脚可以用作标准 GPIO。
- 只接收模式 **(RXONLY=1)**：应用可通过将 RXONLY 位置 1 来禁止 SPI 输出功能。在从器件配置下，

MISO 输出被禁止，该引脚可用作 GPIO。当从器件选择信号有效时，从器件继续从 MOSI 引脚接收数据（请参见多主器件通信）。基于数据缓冲区的配置产生接收数据事件。在主器件配置下， MOSI 输出被禁止，该引脚可用作 GPIO。只要 SPI 处于使能状态，便不断生成时钟信号。停止时钟的唯一方式是将 RXONLY 位或 SPE 位清零，直至来自 MISO 引脚的传入模式结束，然后基于相应配置填充数据缓冲区结构。

图 33-4 单工单个主器件/单个从器件应用（主器件为只发送模式/从器件为只接收模式）



1. NSS 引脚可用于在主器件和从器件之间提供硬件控制流。外设也可选择不使用这些引脚。之后，必须在内部为主器件和从器件处理硬件控制流。有关更多详细信息，请参见第 *31.3.5* 节：从器件选择 *(NSS)* 引脚管理。

2. 在发送器 Rx 移位寄存器的输入上捕获意外输入信息。标准只发送模式下必须忽略与发送器接收流相关的所有事件（例如 OVF 标志）。

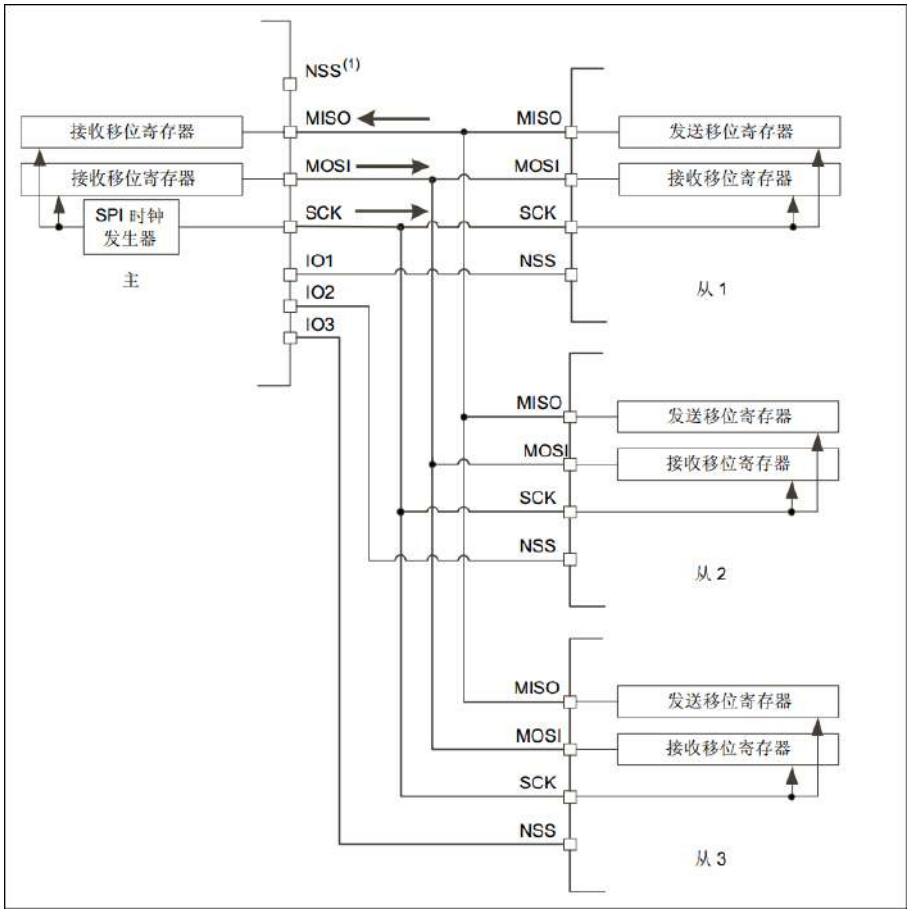
3. 在这种配置下，两个 MISO 引脚均可用作 GPIO。

注： 任何单工通信均可以通过半双工通信的一种变型来替换，该变型中设置的数据传输方向不变（在 BDIO 位保持不变的同时，双向模式处于使能状态）。

33.6.2 标准多从器件通信

在具有两个或多个独立从器件的配置下，主器件使用 GPIO 引脚来管理每个从器件的片选线（请参见图 33-5）。主器件必须通过拉低与从器件 NSS 输入相连的 GPIO 的电平来单独选择一个从器件。执行该操作后，便建立了标准主器件与专用从器件之间的通信。

图 33-5 主器件和三个独立的从器件



1. 此配置的主器件侧不使用 NSS 引脚。该引脚必须在内部管理 (SSM=1， SSI=1) 以避免任何 MODF 错误。2. 由于从器件的 MISO 引脚连在一起，所有从器件 MISO 引脚的 GPIO 配置必须设置为复用功能开漏（请参见时钟中断清零寄存器 *(RCC_CICR)*）。

33.6.3 多主器件通信

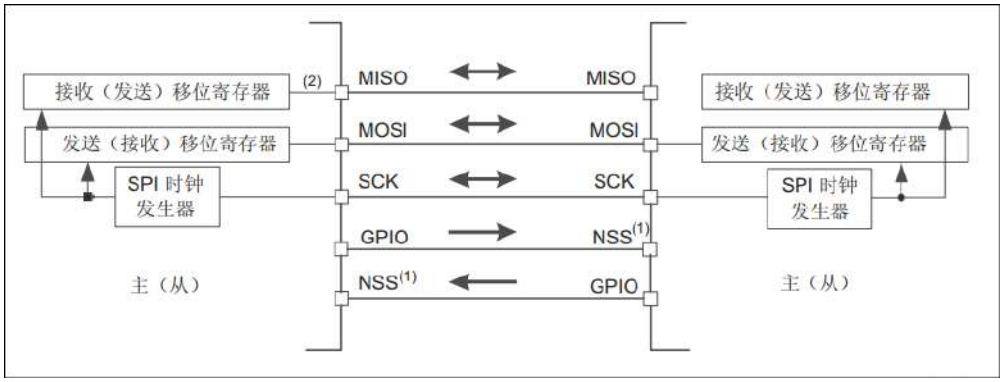
如果 SPI 总线未用于多主功能，用户可使用内置功能来检测尝试同时控制总线的两个节点间是否存在潜在冲突。对于该检测， NSS 引脚配置为硬件输入模式。

由于此时只有一个结点可将其输出施加到公用数据线上，因此无法连接超过两个以此模式工作的 SPI 节点。

当节点无效时，默认情况下均保持从模式。一旦一个节点要接管对总线的控制，它会将自身切换到主模式，然后通过专用 GPIO 引脚向其它节点的从器件选择输入施加有效电平。会话完成后，有效的从器件选择信号将被释放，控制总线的节点会短暂切换回被动从模式，等待下一个会话开始。

如果两个节点同时发出各自的控制请求，则会出现总线冲突（请参见模式故障 MODF 事件）。随后，用户可应用某个简单的仲裁过程（例如，在两个节点上施加不同的预定义超时来推迟下一个尝试） 。

图 33-6 多主器件应用



1. 在两个节点上， NSS 引脚配置为硬件输入模式。当无效节点配置为从器件时，其有效电平将使能 MISO 线输出控制。

33.6.4 从器件选择 (NSS) 引脚管理

在从模式下， NSS 用作标准的“片选”输入，使从器件与主器件进行通信。在主模式下，NSS 可用作输出或输入。用作输入时，可防止多主模式总线冲突；用作输出时，可驱动单个从器件的从器件选择信号。

可以使用 SPIx_CR1 寄存器中的 SSM 位设置硬件或软件从器件选择管理：

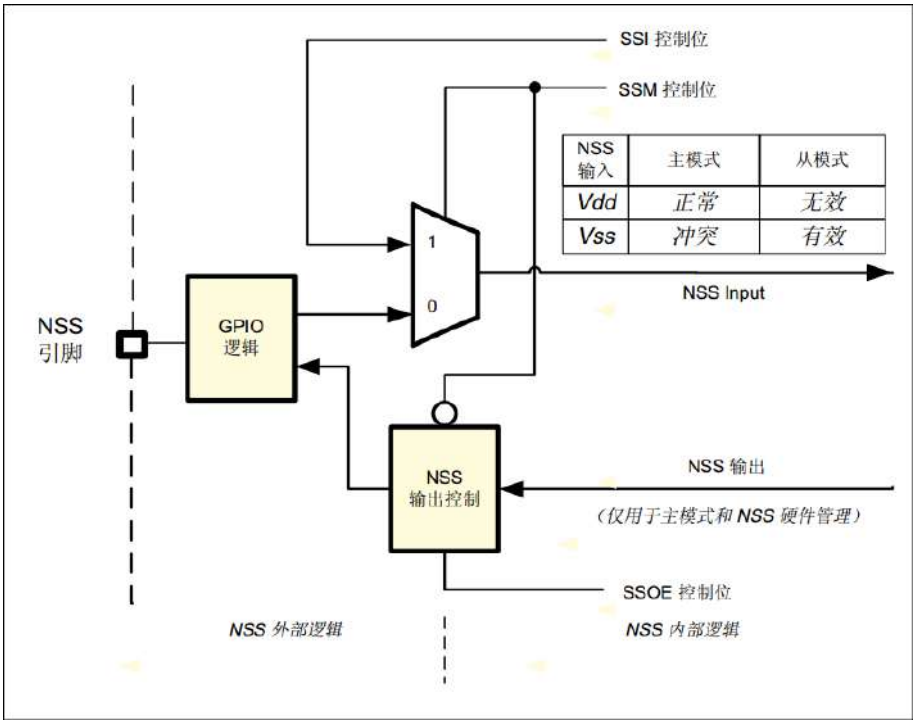
• 软件 **NSS** 管理 **(SSM = 1)**：在这种配置下，由 SPIx_CR1 寄存器中的 SSI 位的值内部驱动从器件选择信息。外部 NSS 引脚空闲，可供其它应用使用。

• 硬件 **NSS** 管理 **(SSM = 0)**：在这种情况下，可行的配置有两种：所用配置取决于 NSS 输出配置（SPIx_CR1 寄存器中的 SSOE 位）。

– **NSS** 输出使能（**SSM=0** 且 **SSOE = 1**）：仅在将 MCU 设置为主器件时才使用该配置。 NSS 引脚由硬件管理。只要在主模式下使能 SPI (SPE=1)， NSS 信号便会被驱动为低电平，并且会一直保持低电平状态，直至禁止 SPI (SPE =0)。

– **NSS** 输出禁止（**SSM=0** 且 **SSOE = 0**）：如果微控制器在总线上用作主器件，此配置可实现多主模式功能。如果在该模式下将 NSS 引脚拉至低电平， SPI 将进入主模式故障状态，器件将在从模式下自动进行重新配置。在从模式下， NSS 引脚用作标准的“片选”输入，当 NSS 线为低电平时将选择从器件。

图 33-7 硬件/软件从器件选择管理



33.6.5 通信格式

SPI 通信过程中，将同时执行接收和发送操作。串行时钟 (SCK) 对数据线上的信息的移位和采样进行同步。通信格式取决于时钟相位、时钟极性和数据帧格式。为了能够在彼此间进行通信，主器件和从器件必须遵循相同的通信格式。

时钟相位和极性控制

通过 SPIx_CR1 寄存器中的 CPOL 和 CPHA 位，可以用软件选择四种可能的时序关系。CPOL（时钟极性）位控制不传输任何数据时时钟的空闲状态值。此位对主器件和从 器件都有作用。如果复位 CPOL，SCK 引脚在空闲状态处于低电平。如果将 CPOL 置 1， SCK 引脚在空闲状态处于高电平。

如果将 CPHA 位置 1,则会在 SCK 引脚的第二个边沿捕获传输的第一个数据位(如果复位 CPOL 位，则为下降沿；如果将 CPOL 位置 1，则为上升沿）。即，在每次出现该时钟边沿时锁存数据。如果将 CPHA 位复位，则会在 SCK 引脚的第一个边沿捕获传输的第一个数据位（如果将 CPOL 位置 1，则为下降沿；如果将 CPOL 位复位，则为上升沿）。即，在每次出现该时钟边沿时锁存数据。

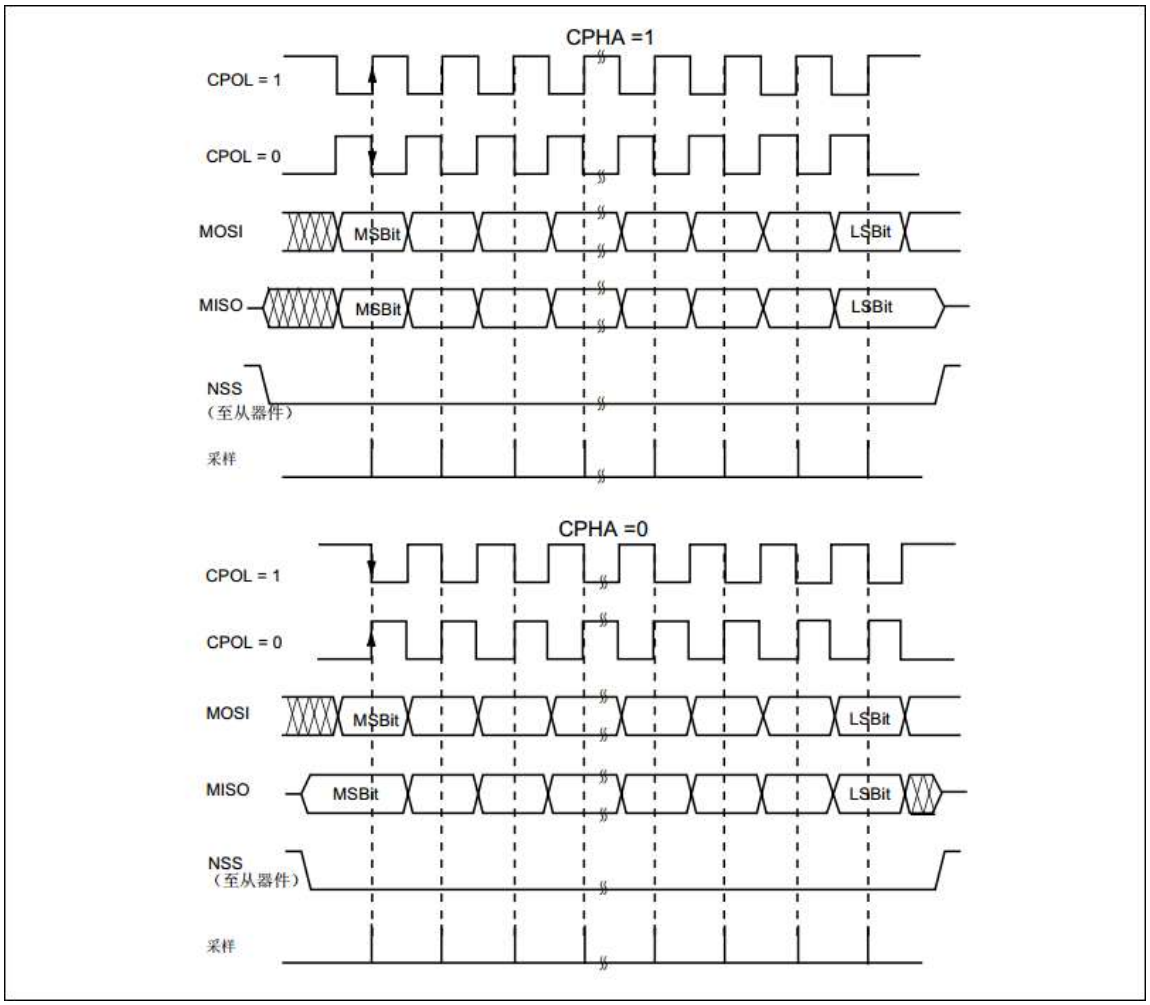
CPOL（时钟极性）和 CPHA（时钟相位）位的组合用于选择数据捕获时钟边沿。

图 33-8 给出了在 CPHA 和 CPOL 位的四种组合下的 SPI 全双工传输。

注： 在切换 *CPOL/CPHA* 位之前，必须通过复位 *SPE* 位来禁止 *SPI*。

SCK 的空闲状态必须与 *SPIx_CR1* 寄存器中选择的极性相对应（如果 *CPOL=1*，则上拉 *SCK*；如果 *CPOL=0*，则下拉 *SCK*）。

图 33-8 数据时钟时序图



注： 数据位的顺序取决于 LSBFIRST 位的设置。

数据帧格式

SPI 移位寄存器可设置为以 MSB 在前或 LSB 在前的方式移出数据，具体取决于 LSBFIRST 位的值。每个数据帧的长度均为 8 位或 16 位，具体取决于使用 SPI_CR1 寄存器中的 DFF 位编程的数据长度。所选的数据帧格式适用于发送和接收。

33.6.6 SPI 配置

主器件和从器件的配置步骤几乎相同。对于具体的模式设置，请遵从相应章节的内容。若要对标准通信进行初始化，请执行以下步骤：

1. 对相应的 GPIO 寄存器执行写操作：将 MOSI、 MISO 和 SCK 引脚配置为 GPIO。

2. 对 SPI_CR1 寄存器执行写操作：

a) 通过 BR[2:0] 位配置串行时钟波特率。

b) 配置 CPOL 位和 CPHA 位组合，从数据与时钟四组时序定义中选择一种定义。

c) 通过配置 RXONLY 或 BIDIMODE 和 BIDIOE 来选择单工或半双工模式（RXONLY 和 BIDIMODE 不可同时置 1）。

d) 配置 LSBFIRST 位以定义帧格式（注： 2）。

e) 如果需要 CRC，请配置 CRCEN 和 CRCEN 位（SCK 时钟信号处于空闲状态时）。

f) 配置 SSM 和 SSI（注： 2）。

g) 配置 MSTR 位（在多主模式 NSS 配置下，如果主器件配置为防止发生 MODF 错误，则应避免 NSS 上出现状态冲突）。

h) 将 DFF 位置 1 以配置数据帧格式（8 位或 16 位）。

3. 对 SPI_CR2 寄存器执行写操作：

a) 配置 SSOE（注： 1 和 2）。

b) 如果需要 TI 协议，请将 FRF 位置 1。

4. 对 SPI_CRCPR 寄存器执行写操作：需要时配置 CRC 多项式。

5. 对相应的 DMA 寄存器执行写操作：如果使用 DMA 数据流，请在 DMA 寄存器中配置 SPI Tx 和 Rx 专用的 DMA 数据流。

注： *(1)* 从模式下无需此步骤。

(2) TI 模式下无需此步骤。

(3) 从模式下无需此步骤，但从器件在 *TI* 模式下工作时除外。

使能 SPI 的步骤

建议在主器件发送时钟前使能 SPI 从器件。否则，数据传输可能会不正常。从器件的数据寄存器必须包含待发送的数据才能开始与主器件通信（在通信时钟的第一个边沿；如果时钟信号连续，则是在正在进行的通信结束前）。使能 SPI 从器件前， SCK 信号必须稳定为所选极性对应的空闲状态电平。

在全双工（或任何只发送模式）下，使能 SPI 后，主器件开始通信，待发送的数据将写入 Tx 缓冲区。

在任何主器件只接收模式（RXONLY=1 或 BIDIMODE=1 且 BIDIOE=0）下，使能 SPI 后，主器件立即开始通信且时钟立即开始运行。

从器件接收到来自主器件的正确时钟信号时，它将开始通信。在 SPI 主器件启动传输前，从软件必须写入待发送的数据。

有关如何处理 DMA 的详细信息，请参见使用 *DMA*（直接存储器寻址）进行通信。

33.6.7 数据发送和接收过程

接收和发送缓冲区

在接收过程中，数据收到后，先存储到内部接收缓冲区中；而在发送过程中，先将数据存储到内部发送缓冲区中，然后发送数据。对 SPI_DR 寄存器的读访问将返回接收缓冲值，而对 SPI_DR 寄存器的写访问会将写入的数据存储到发送缓冲区中。

发送缓冲区处理

在第一个位传输期间，数据帧从发送缓冲区加载到移位寄存器。各个位随后从移位寄存器以串行方式移出到专用输出引脚，具体取决于 LSBFIRST 位设置。当数据从发送缓冲区传送到移位寄存器时， TXE 标志（发送缓冲区为空）置 1。该标志表示内部发送缓冲区已准备好加载接下来的数据。如果 SPI_CR2 寄存器中的 TXEIE 位置 1，可产生中断。通过对 SPI_DR 寄存器执行写操作将 TXE 位清零。

如果在前一次帧传输仍在进行时将要发送的下一个数据存储到发送缓冲区，则可保持连续的发送流。如果软件在 TXE 未置 1 时写入发送缓冲区，则等待发送的数据将被覆盖。

接收缓冲区处理

将数据从移位寄存器传输到接收缓冲区时， RXNE 标志（接收缓冲区非空）会在最后一个采样时钟边沿置 1。它表示已准备好从 SPI_DR 寄存器中读取数据。如果 SPI_CR2 寄存器中的 RXNEIE 位置 1，可产生中断。读取 SPI_DR 寄存器即会使 RXNE 位清零。

如果器件未将因发送前一个数据字节而产生的 RXNE 位清零，则在下一个值缓冲时会产生溢出条件。OVR 位置 1 并在 ERRIE 位置 1 时生成一个中断。

另一种管理数据交换的方式是使用 DMA（请参见 *CRS* 主要特性）。

序列处理

当前数据帧传输正在进行时， BSY 位将置 1。当时钟信号连续运行时， BSY 标志在主器件侧的两个

数据帧间保持置 1。不过，在从器件侧，它将在每个数据帧传输之间变为 0 并持续至少一个 SPI 时钟周期。

对于某些配置，可以在最后一次数据传输期间使用 BSY 标志来等待传输完成。

当主器件侧配置只接收模式时，无论是半双工（BIDIMODE=1 且 BIDIOE=0）还是单工配置（BIDIMODE=0 且 RXONLY=1），主器件都会在 SPI 使能后立即启动接收序列。随后，时钟信号由主器件提供，且直至主器件禁止 SPI 或只接收模式才会停止。在此之前，主器件会连续接收数据帧。

当主器件能够以连续模式（SCK 信号连续）提供所有交互时，任何时候都必须根据从器件能力来处理数据流及其内容。必要时，主器件必须降低通信速度，提供较慢的时钟或带有足够延时的单独帧或数据段。请注意，在 SPI 模式下工作时不存在从器件的下溢错误信号，来自从器件的数据始终由主器件处理，即使从器件无法及时正确地准备数据也是如此。从器件最好使用 DMA，尤其是数据帧较短而总线速率较高时。

在多从器件系统中，每个序列必须通过 NSS 脉冲进行控制，从而只选择其中一个从器件进行通信。在单个从器件系统中，不必使用 NSS 来控制从器件。但是，可使用 NSS 脉冲将从器件与每个数据传输序列的开始同步。 NSS 可通过软件或硬件进行管理（请参见多主器件通信）。

有关主器件/全双工和从器件/全双工模式下的连续传输的说明，请参见图 33-10 和图 33-9。

图 33-9 主/全双工模式（BIDIMODE=0 且 RXONLY=0）下的 TXE/RXNE/BSY 时序（在连续传输的情况下）

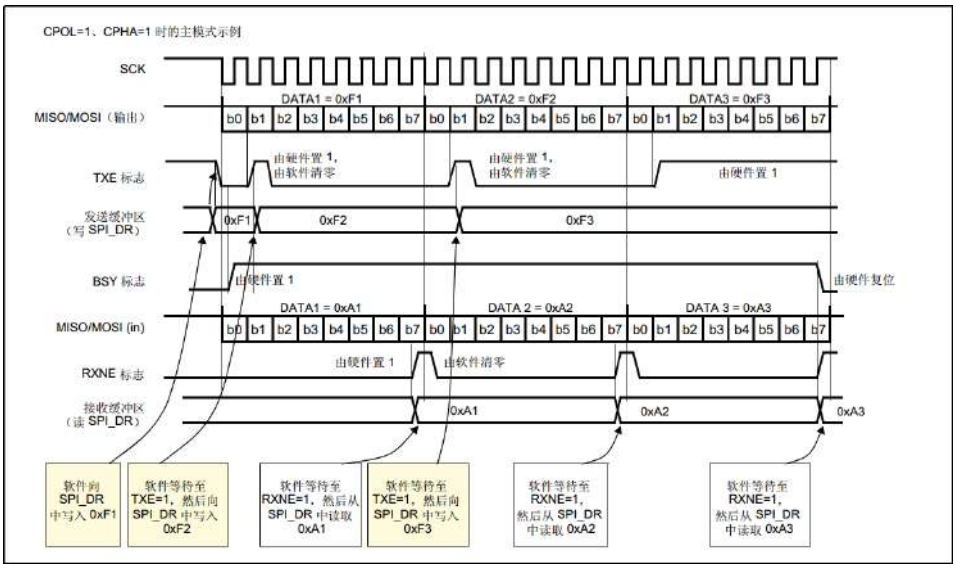
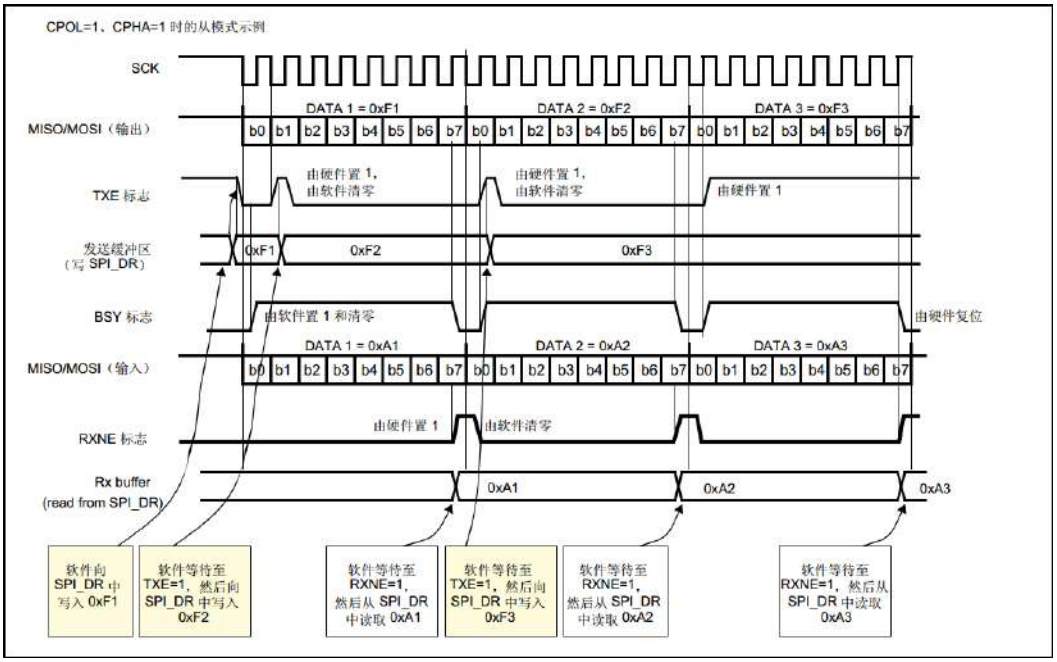


图 33-10 从器件/全双工模式（**BIDIMODE=0** 且 **RXONLY=0**）下的 **TXE/RXNE/BSY** 时序（在连续传输的情况下）



33.6.8 禁止 SPI 的步骤

当禁止 SPI 时，必须按照本段中介绍的禁止步骤进行操作。当外设时钟停止时，在系统进入低功耗模式前做到这一点是十分重要的。否则会损坏正在进行的交互。在某些模式下，禁止步骤是停止所进行的连续通信的唯一方式。

当处于全双工或只发送模式下的主器件停止提供待发送的数据时，可结束任何交互。在这种情况下，时钟在最后一个数据传输后停止。

标准禁止步骤通过轮询 BSY 状态以及 TXE 标志来检查发送会话是否完全结束。还可以在必须识别正在处理的传输是否结束的特定情况下完成这种检查，例如：

- 当 NSS 信号由任意 GPIO 切换管理且主器件必须为从器件提供 NSS 脉冲结束时，或者
- 最后一个数据帧或 CRC 帧传输仍在外设总线中处理时，来自 DMA 的传输数据流完成。

正确的禁止步骤如下（使用只接收模式时除外）：

1. 等待 RXNE=1 以接收最后的数据。
2. 等待至 TXE=1，然后等待至 BSY=0，再关闭 SPI。
3. 读取接收的数据。

注： 在不连续通信期间，在对 *SPI_DR* 寄存器执行写操作与 *BSY* 位置 *1* 之间有 *2* 个 *APB* 时钟周

期的延迟。因此，写入最后的数据后，必须先等待 *TXE* 位置 *1*，然后等待 *BSY* 位清零。

某些只接收模式的正确禁止步骤如下：

1. 当最后一个数据帧正在处理时，通过在特定时间窗口内禁止 SPI (SPE=0) 来中断接收流。
2. 等待至 BSY=0（最后一个数据帧已处理完）。
3. 读取接收的数据。

注： 要停止连续的接收序列，必须在接收最后一个数据帧时遵循特定的时间窗口。该时间窗口在第一个位采样时开始，在最后一个位的传输开始前结束。

33.6.9 使用 DMA（直接存储器寻址）进行通信

为了以最大速度工作并且为了促进避免上溢所需的数据寄存器读/写过程， SPI 提供了 DMA 功能，该功能采用了简单的请求/应答协议。

将 SPIx_CR2 寄存器中的使能位 TXE 或 RXNE 置 1 时，将请求 DMA 访问。必须向发送缓冲区和接收缓冲区发出单独的请求。

- 在发送过程中，每次 TXE 位置 1 都会发出 DMA 请求。然后， DMA 将对 SPIx_DR 寄存器执行写操作。
- 在接收过程中，每次 RXNE 位置 1 都会发出 DMA 请求。然后， DMA 将对 SPIx_DR 寄存器执行读操作。

有关 DMA 发送和接收波形的说明，请参见图 33-11 和图 33-12。

当 SPI 仅用于发送数据时，可以只使能 SPI Tx DMA 通道。在这种情况下， OVR 标志会置 1，因为未读取接收的数据。当 SPI 仅用于接收数据时，可以只使能 SPI Rx DMA 通道。

在发送模式下， DMA 写入所有要发送的数据（DMA_ISR 寄存器中的 TCIF 标志置 1）后，可以对 BSY 标志进行监视，以确保 SPI 通信已完成。在禁止 SPI 或进入停止模式前必须执行此步骤，以避免损坏最后一次发送。软件必须首先等待 TXE=1，再等待 BSY=0。

通过 DMA 开始通信时，为防止 DMA 通道管理引发错误事件，必须按顺序执行以下步骤：

1. 如果使用 DMA Rx，通过 SPI_CR2 寄存器中的 RXDMAEN 位来使能 DMA 接收缓冲区。
2. 如果使用数据流，通过 DMA 寄存器来使能 Tx 和 Rx 的 DMA 数据流。
3. 如果使用 DMA Tx，通过 SPI_CR2 寄存器中的 TXDMAEN 位来使能 DMA 发送缓冲区。
4. 通过将 SPE 位置 1 使能 SPI。

要关闭通信，必须按顺序执行以下步骤：

1. 如果使能了 DMA，通过 DMA 寄存器来禁止 Tx 和 Rx 的 DMA 数据流。
2. 通过后续 SPI 禁止步骤来禁止 SPI。
3. 如果使用 DMA Tx 和/或 DMA Rx，通过将 SPI_CR2 寄存器中的 TXDMAEN 和 RXDMAEN 位清零来禁止 DMA 发送缓冲区和接收缓冲区。

图 33-11 使用 **DMA** 进行发送

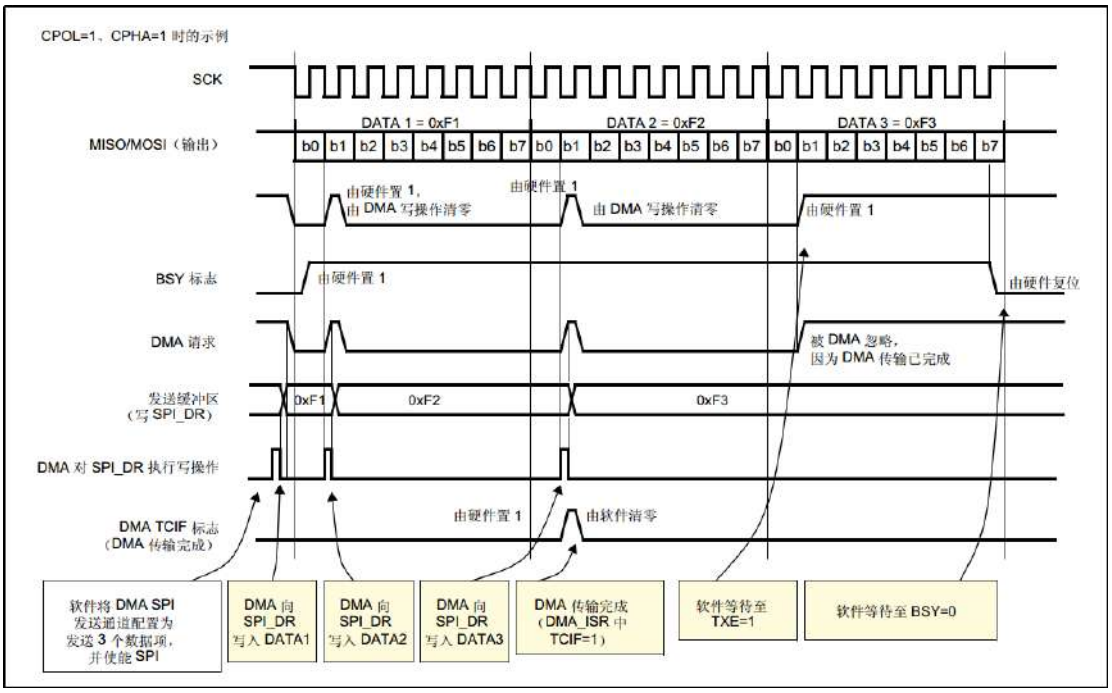
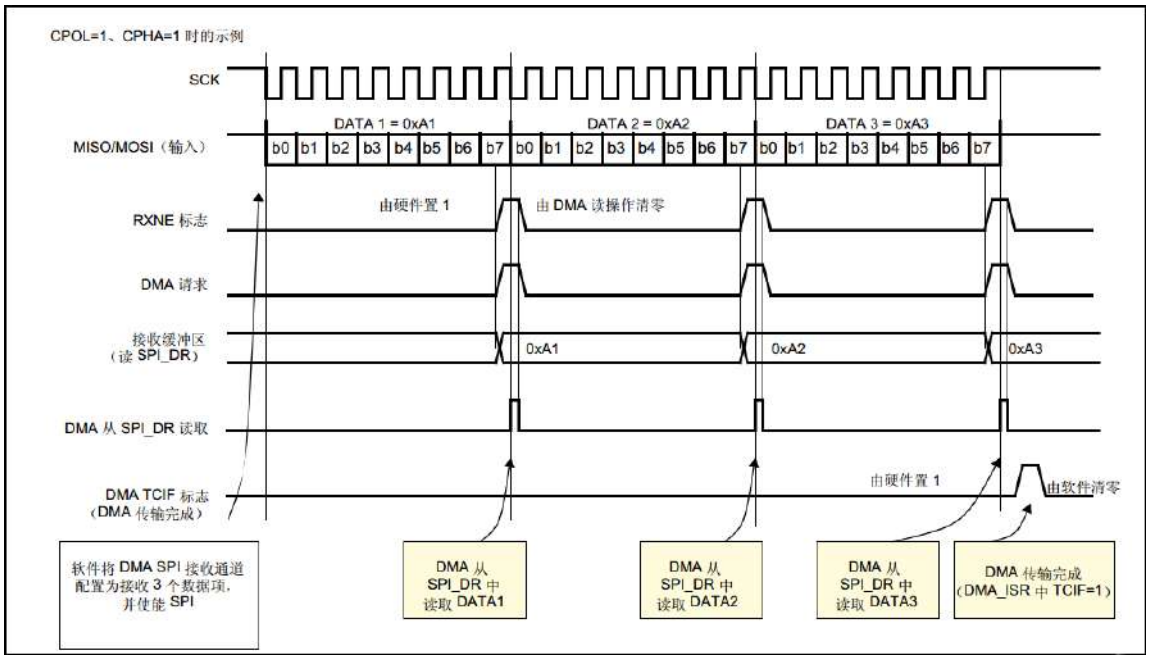


图 33-12 使用 **DMA** 进行接收



33.6.10 SPI 状态标志

应用可通过三种状态标志监视 SPI 总线的状态。

发送缓冲区为空 **(TXE)**

TXE 标志置 1 时，表示发送缓冲区为空，可以将待发送的下一个数据加载到缓冲区中。对 SPI_DR 寄存器执行写操作时，将清零 TXE 标志。

接收缓冲区非空 **(RXNE)**

RXNE 标志置 1 时，表示接收缓冲区中存在有效的已接收数据。通过对 SPI_DR 寄存器执行读取操作将该位清零。

忙标志 **(BSY)**

BSY 标志由硬件置 1 和清零（写入此标志没有任何作用）。

当 BSY 置 1 时，表示 SPI 上正在进行数据传输（SPI 总线繁忙）。在主模式下的双向通信接收模式（MSTR=1 且 BDM=1 且 BDOE=0）有一个例外情况， BSY 标志在接收过程中保持置 0。

在某些模式下，可使用 BSY 标志来检测传输是否结束，从而避免在进入低功耗模式前禁止 SPI 外设时钟时或通过软件管理 NSS 脉冲结束时破坏最后一次传输。

BSY 标志还可用于避免在多主模式系统中发生写冲突。

在以下任意一种条件下， BSY 标志将清零：

- 正确禁止 SPI 时
- 在主模式下检测到故障时（MODF 位置 1）
- 在主模式下，完成了数据发送并且不准备发送任何新数据时
- 在从模式下， BSY 标志在各传输之间的至少一个 SPI 时钟周期内置为“0”时。

注： 建议始终使用 *TXE* 和 *RXNE* 标志（而非 *BSY* 标志）来处理数据发送或接收操作。

33.6.11 SPI 错误标志

如果以下其中一个错误标志置 1 且已通过将 ERRIE 位置 1 使能了中断，则将生成 SPI 中断。

上溢标志 **(OVR)**

如果从器件在接收缓冲区中前一个帧的读操作尚未完成时（RXNE 标志置 1）完成下一个数据帧的接收，则会出现溢出条件。

在这种情况下， Rx 缓冲区的内容不会更新为接收的新数据。对 SPI_DR 寄存器执行的读操作将返回先前接收的帧。后续发送的所有其它数据均将丢失。

要将 OVR 位清零，应首先对 SPI_DR 寄存器执行读访问，然后再对 SPI_SR 寄存器执行读访问。

模式故障 **(MODF)**

当主器件的内部 NSS 信号（NSS 硬件模式下为 NSS 引脚， NSS 软件模式下为 SSI 位）被拉低时，将发生模式故障。这会自动将 MODF 位置 1。主模式故障会在以下几方面影响 SPI 接口：

- 如果 ERRIE 位置 1， MODF 位将置 1，并生成 SPI 中断。
- SPE 位清零。这将关闭器件的所有输出，并禁止 SPI 接口。
- MSTR 位清零，从而强制器件进入从模式。

使用以下软件序列将 MODF 位清零：

1. 在 MODF 位置 1 时，对 SPIx_SR 寄存器执行读或写访问。
2. 然后，对 SPIx_CR1 寄存器执行写操作。

为避免包含多个 MCU 的系统中发生多从模式冲突，必须在 MODF 位清零序列期间将 NSS 引脚拉高。在该清零序列后，可以将 SPE 和 MSTR 位恢复到原始状态。安全起见，硬件不允许在 MODF 位置 1 时将 SPE 和 MSTR 位置 1。在从器件中， MODF 位不可置 1，但由前一次多主模式冲突引起时除外。

CRC 错误 **(CRCERR)**

当 SPIx_CR1 寄存器中的 CRCEN 位置 1 时，此标志用于验证接收数据的有效性。如果移位寄存器中接收的值与 SPIx_RXCRC 的值不匹配，SPIx_SR 寄存器中的 CRCERR 标志将置 1。该标志由软件清零。

TI 模式帧格式错误 **(FRE)**

如果 SPI 在从模式下工作，并配置为符合 TI 模式协议，则在通信进行期间出现 NSS 脉冲时，将检测到 TI 模式帧格式错误。出现此错误时， SPIx_SR 寄存器中的 FRE 标志将置 1。

发生错误时不会禁止 SPI，但会忽略 NSS 脉冲，并且 SPI 会等待下一个 NSS 脉冲，然后再开始新的

传输。由于错误检测可能导致丢失两个数据字节，因此数据可能会损坏。

读取 SPIx_SR 寄存器时，将清零 FRE 标志。如果 ERRIE 位置 1，则检测到 NSS 错误时将生成中断。在这种情况下，由于无法保证数据的一致性，应禁止 SPI，并在重新使能从 SPI 后，由主器件重新发起通信。

33.7 SPI 特性

33.7.1 TI 模式

主模式下的 **TI** 协议

SPI 接口与 TI 协议兼容。可以使用 SPIx_CR2 寄存器的 FRF 位来配置 SPI，以兼容此协议。

时钟极性和相位都被强制为遵循 TI 协议，和 SPIx_CR1 中的设置无关。 NSS 管理也特定于 TI 协议，在这种情况下，无法通过 SPIx_CR1 和 SPIx_CR2 寄存器（SSM、 SSI 和 SSOE）来对 NSS 管理进行配置。

在从模式下， SPI 波特率预分频器用于控制在当前传输完成时 MISO 引脚切换为高阻态的时刻（请参见图 *284*）。可以使用任意波特率，因此可以非常灵活地确定此时刻。但是，波特率通常设置为外部主时钟波特率。 MISO 信号变为高阻态的延时 (trelease) 取决于内部重新同步以及通过 SPIx_CR1 寄存器的 BR[2:0] 位设置的波特率值。具体公式如下：

$$\frac{t_{baud_{rate}}}{2} + 4xt_{pclk} < t_{release} < \frac{t_{baud_{rate}}}{2} + 6xt_{pclk}$$

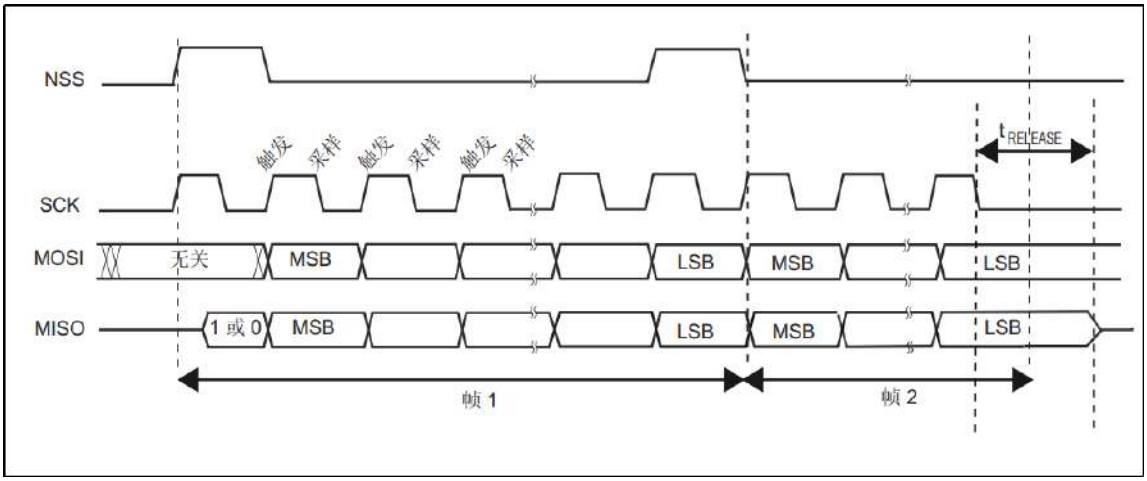
如果从器件在数据帧传输期间检测到错位的 NSS 脉冲， TIFRE 标志将置 1。

此特性不适用于 Motorola SPI 通信（FRF 位为 0）。

注： 要在从器件发送器模式下使用错误中断 *(ERRIE = 1)* 检测 *TI* 帧错误，必须通过将 *SPI_CR1* 寄存器中的 *BIDIMODE* 和 *BIDIOE* 置 *1* 来将 *SPI* 配置为双线单向模式。当 *BIDIMODE* 置为 *0* 时， *OVR* 将置 *1*，因为始终不会读取数据寄存器从而始终生成错误中断；而当 *BIDIMODE* 置 *1* 时，不会接收数据，也不会将 *OVR* 置 *1*。

图 33-13 给出了选择 *TI* 模式时的 *SPI* 通信波形。

图 33-13 TI 模式传输



33.7.2 CRC 计算

为检查发送数据和接收数据的可靠性，使用两个独立的 CRC 计算器（作用于发送和接收数据流）。 S

PI 提供 CRC8 或 CRC16 计算，具体取决于通过 DFF 位选择的数据格式。 CRC 通过 SPI_CRCPR 寄存器中编程的多项式连续计算。

CRC 原理

在使能 SPI (SPE = 1) 前，通过将 SPIx_CR1 寄存器中的 CRCEN 位置 1 来使能 CRC 计算。使用值为奇数的可编程多项式对每个位计算 CRC 值。在由 SPIx_CR1 寄存器中的 CPHA 位和 CPOL 位定义的采样时钟边沿进行计算。所计算的 CRC 值在数据块末尾自动进行校验，以及针对由 CPU 或 DMA 管理的传输进行校验。当检测到所接收数据内部计算的 CRC 与发送器发送的 CRC 不匹配时， CRCERR 标志将置 1 以指示数据损坏错误。 CRC 计算的正确处理步骤取决于 SPI 配置和所选的传输管理。

注： 多项式值只应为奇数。不支持任何偶数值。

CPU 管理的 **CRC** 传输

通信开始后将一直持续到必须发送或接收 SPIx_DR 寄存器中的最后一个数据帧时。之后，SPIx_CR1 寄存器中的 CRCNEXT 位必须置 1，以指示当前处理的数据帧传输后将处理 CRC 帧传输。 CRCNEXT 位必须在最后一个数据帧传输结束前置 1。在 CRC 传输期间，CRC 计算将冻结。

与任何其它数据帧一样，接收的 CRC 存储在接收缓冲区内。

CRC 帧的传输通常在数据序列结束时再传输一个数据帧。

接收最后一个 CRC 数据后，将执行自动校验，将接收的值与 SPIx_RXCRC 寄存器中的值进行比较。软件必须校验 SPIx_SR 寄存器中的 CRCERR 标志，以确定数据传输是否损坏。软件通过向 CRCERR 标志写入“0”来将其清零。

接收 CRC 后， CRC 值存储到接收缓冲区中，且必须在 SPIx_DR 寄存器中进行读取，以将 RXNE 标志清零。

DMA 管理的 **CRC** 传输

当使能的 SPI 通信支持 CRC 通信和 DMA 模式时，在通信结束时会自动发送和接收 CRC（在只接收模式下读取 CRC 数据时除外）。 CRCNEXT 位不是一定要通过软件来处理。 SPI 发送 DMA 通道计数器必须设置为要发送的数据帧数，其中不包括 CRC 帧。在接收器侧，接收的 CRC 值在传输结束时通过 DMA 自动处理，但用户必须注意刷新 SPI_DR 中接收的 CRC 帧（因为该信息始终加载到其中）。

如果传输过程中出现损坏，则在数据和 CRC 传输结束时， SPIx_SR 寄存器中的 CRCERR 标志将置 1。

复位 **SPIx_TXCRC** 和 **SPIx_RXCRC** 值

当使能 CRC 计算时， SPIx_TXCRC 和 SPIx_RXCRC 值自动清零。

当 SPI 配置为从模式并且 CRC 功能已使能时，即使 NSS 引脚上为高电平，也会进行 CRC 计算。例如，在多从模式环境下可能出现这种情况，此时通信主器件会交替寻址从器件。

在禁止从器件（NSS 上为高电平）到使能新的从器件（NSS 上为低电平）的时间内，应在主器件和从器件两端同时将 CRC 值清零，以重新同步主从双方的 CRC 计算。

要将 CRC 清零，请按以下步骤操作：

1. 禁止 SPI
2. 将 CRCEN 位清零
3. 使能 CRCEN 位
4. 使能 SPI

注： 当 *SPI* 处于从模式时，只要 *CRCEN* 位置 *1*，无论 *SPE* 位的值如何，只要存在 *SCK* 从输入时钟， *CRC* 计算器就开始工作。为了避免 *CRC* 计算错误，软件必须仅在时钟稳定（处于稳态）时才能使能 *CRC* 计算。当 *SPI* 接口配置为从模式时， *NSS* 内部信号需要在数据阶段和 *CRC* 阶段之间保持低电平。

33.8 SPI 中断

在 SPI 通信过程中，中断可由以下事件产生：

- 发送缓冲区准备就绪，可以装载数据
- 接收缓冲区中接收了数据
- 主模式故障
- 上溢错误
- TI 帧格式错误

中断可分别进行使能和禁止。

表 33-2 **SPI** 中断请求

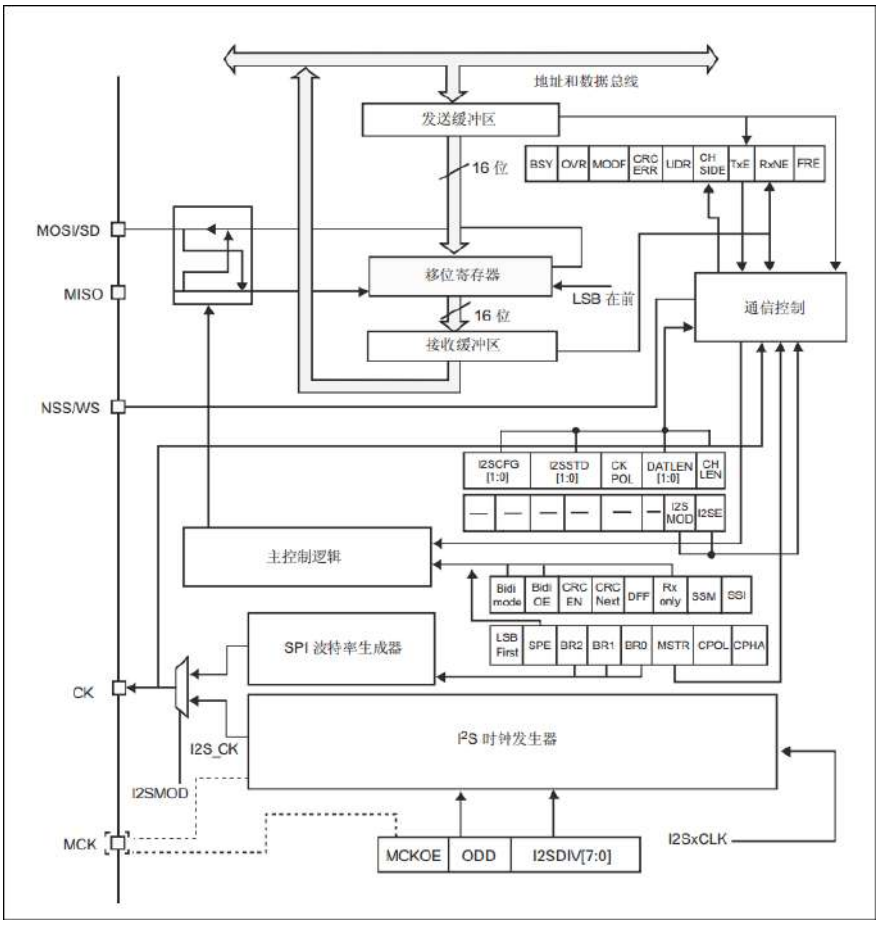
中断事件	事件标志	使能控制位
发送缓冲区准备就绪，可以装载数据	TXE	TXEIE
接收缓冲区中接收了数据	RXNE	RXNEIE
主模式故障	MODF	ERRIE
上溢错误	OVR	
CRC 错误	CRCERR	
TI 帧格式错误	FRE	

33.9 I2S 功能说明

33.9.1 I2S 一般说明

I2S 的框图如图 33-14 所示。

图 33-14 **I2S** 框图



1. MCK 映射到 MISO 引脚上。

当使能 I2S 功能（将 SPIx_I2SCFGR 寄存器中的 I2SMOD 位置 1）后， SPI 可用作音频 I2S 接口。此接口使用几乎与 SPI 相同的引脚、标志和中断。

I2S 与 SPI 共用以下三个引脚：

• SD：串行数据（映射到 MOSI 引脚），用于发送或接收两个时分复用的数据通道上的数据（仅半双工模式）。

• WS：字选择（映射到 NSS 引脚），是主模式下的数据控制信号输出以及从模式下的数据控制信号输入。

• CK：串行时钟（映射到 SCK 引脚），是主模式下的串行时钟输出以及从模式下的串行时钟输入。

当某些外部音频设备需要使用主时钟输出时，可以使用其它引脚：

• MCK：当 I2S 配置为主模式（并且 SPIx_I2SPR 寄存器中的 MCKOE 位置 1）时，使用主时钟（单独映射）输出此附加时钟，该时钟以 256 × fs 的预配置频率生成，其中 fs 为音频信号采样频率。

I2S 在主模式下使用自身的时钟发生器生成通信时钟。此时钟发生器也是主时钟输出的源。

在 I2S 模式下可以使用两个额外的寄存器。一个是时钟发生器配置寄存器 SPIx_I2SPR，另一个是通用 I2S 配置寄存器 SPIx_I2SCFGR（音频标准、从/ 主模式、数据格式、数据包帧、时钟极性等）。

在 I2S 模式下不使用 SPIx_CR1 寄存器和所有 CRC 寄存器。同样，也不使用 SPIx_CR2 寄存器中的 SSOE 位以及 SPIx_SR 中的 MODF 和 CRCERR 位。

I2S 使用相同的 SPI 寄存器 (SPIx_DR) 进行 16 位数据传输。

33.9.2 支持的音频协议

三线总线仅需要处理通常在左右两个通道上时分复用的音频数据。但是，只有一个 16 位寄存器进行发送和接收。所以，需由软件将与每个通道对应的值写入数据寄存器，以及从数据寄存器中读取数据，并通过检查 SPIx_SR 寄存器中的 CHSIDE 位来识别对应的通道。始终先发送左通道数据，而后再发送右通道数据（对于 PCM 协议来说， CHSIDE 没有意义）。

数据和帧格式组合有四种，可采用下列格式发送数据：

- 将 16 位数据封装在 16 位帧中
- 将 16 位数据封装在 32 位帧中
- 将 24 位数据封装在 32 位帧中
- 将 32 位数据封装在 32 位帧中

当使用 32 位数据包中的 16 位数据时，前 16 位 (MSB) 为有效位， 16 位 LSB 被强制清零，无需任何软件操作或 DMA 请求（只需一个读/写操作）。

如果应用程序首选 DMA，则 24 位和 32 位数据帧需要对 SPIx_DR 寄存器执行两次 CPU 读取或写入操作，或者需要两次 DMA 操作。对于 24 位数据帧，硬件会在低 8 位填充 8 个 0 扩展到 32 位。

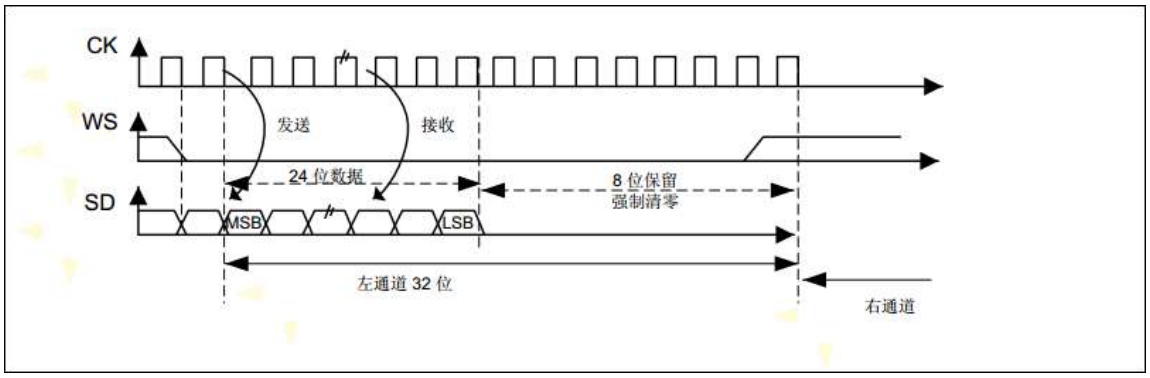
对于所有数据格式和通信标准而言，始终会先发送最高有效位（MSB 优先）。

I2S 接口支持四种音频标准，可使用 SPIx_I2SCFGR 寄存器中的 I2SSTD[1:0] 和 PCMSYNC+位对其进行配置。

I2S Philips 标准

使用 WS 信号来指示当前正在发送的数据所属的通道。该信号从当前通道数据的第一个位(MSB) 之前的一个时钟开始有效。

图 33-15 **I2S Philips 协议波形（16/32 位全精度， CPOL = 0）**

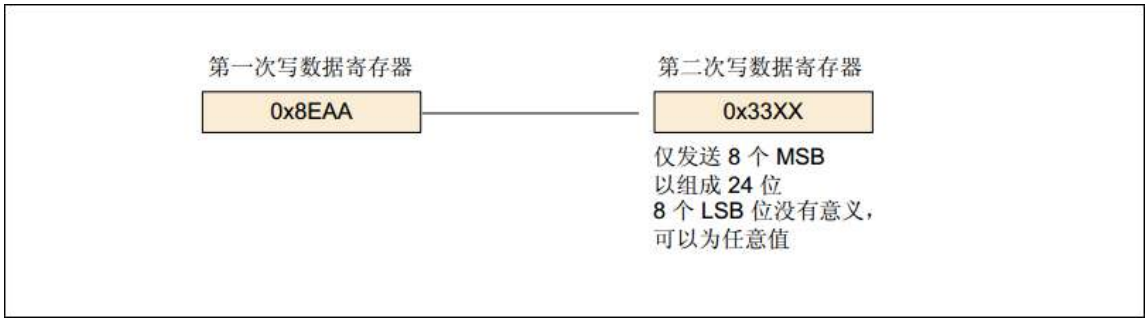


该模式需要对 SPIx_DR 寄存器执行两次写入或读取操作。

- 在发送模式下：

如果需要发送 0x8EAA33（24 位）：

图 33-16 发送 **0x8EAA33**

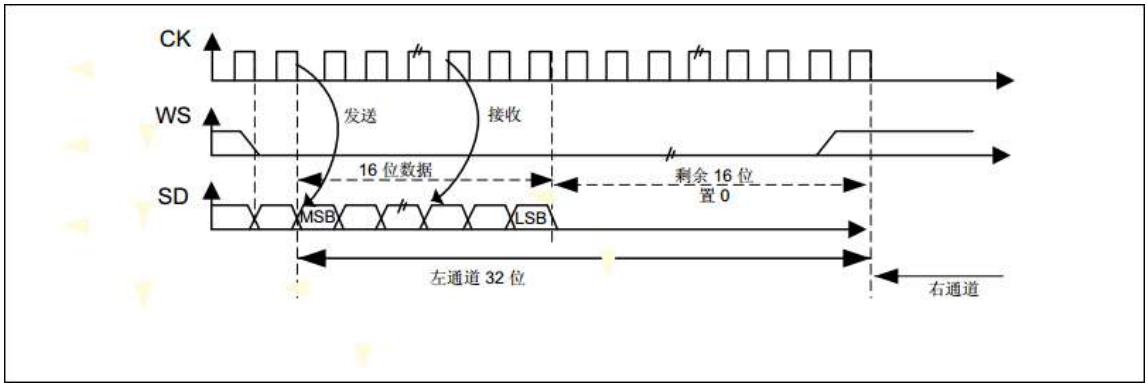


- 在接收模式下：

如果接收数据 0x8EAA33：

图 33-17 接收 **0x8EAA33**

图 33-18 **I2S Philips 标准（16 位扩展为 32 位数据包帧， CPOL = 0）**



如果在 I2S 配置阶段选择将 16 位数据帧扩展到 32 位通道帧，则只需要访问一次 SPIx_DR 寄存器。扩展到 32 位中高半字（16 位 MSB）被硬件置为 0x0000。

如果要发送的数据或已接收的数据为 0x76A3（0x76A30000 扩展为 32 位），则需要执行图 33-19 中显示的操作。

图 33-19 **16 位数据帧扩展到 32 位通道帧的示例**

发送时，每次将 MSB 写入 SPIx_DR， TXE 标志就会置 1，并在中断使能的情况下触发中断，以将要发送的新数据加载到 SPIx_DR 寄存器。即使硬件填充的低 16 位 0x0000 还未发送，也会如此，因为低 16 位是由硬件发送。

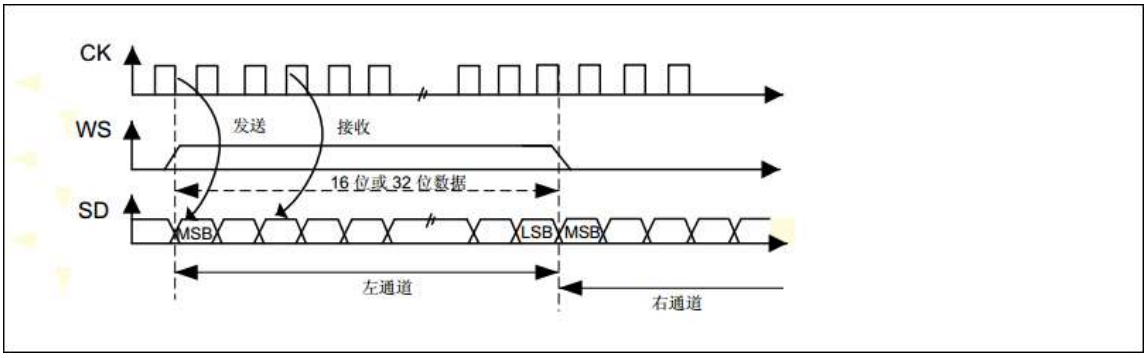
接收时，接收到第一个半字（高 16 位），则硬件将 RXNE 标志置 1，并在中断使能的情况下触发中断。

这样，就延长了两个写入或读取操作之间的时间间隔，从而可防止出现下溢或上溢情况（具体取决于数据传输方向）。

MSB 对齐标准

此标准同时生成 WS 信号和第一个数据位（即 MSBit）。

图 33-20 **MSB** 对齐的 **16** 位或 **32** 位全精度长度， **CPOL = 0**



发送方在时钟信号的下降沿改变数据；接收方在上升沿读取数据。

图 33-21 **MSB** 对齐的 **24** 位帧长度， **CPOL = 0**

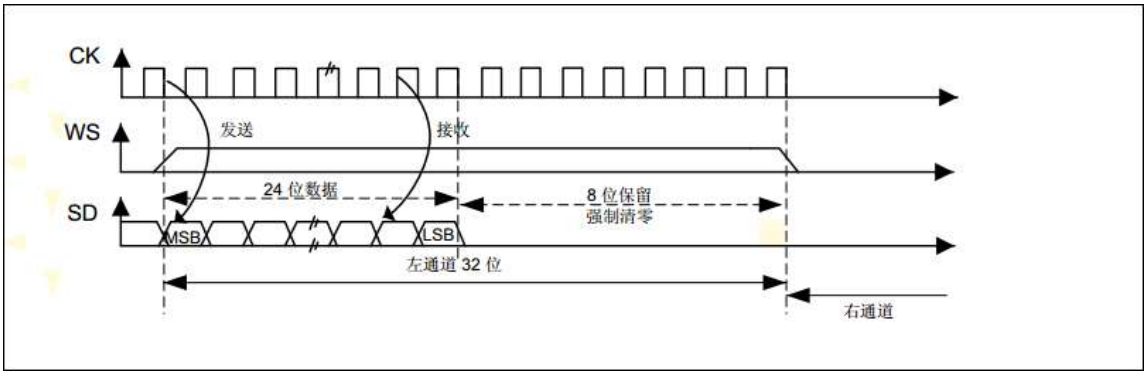
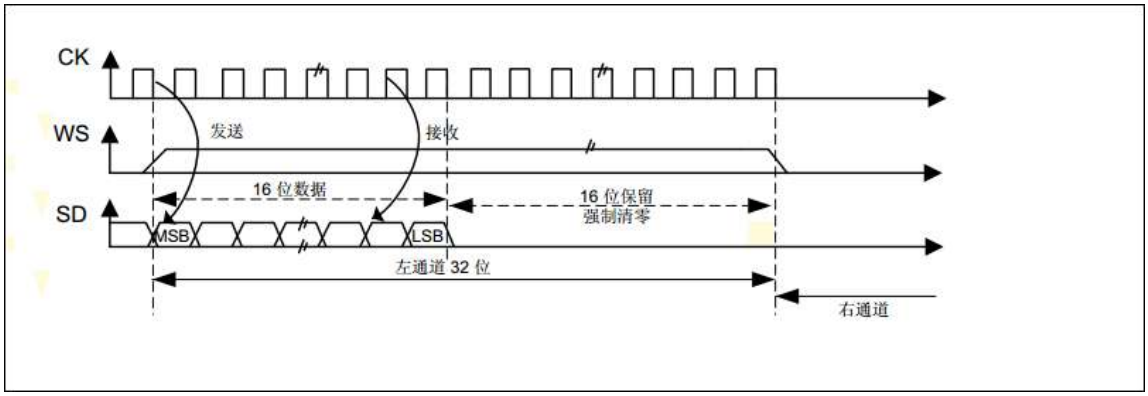


图 33-22 扩展为 **32** 位数据包帧的 **MSB** 对齐的 **16** 位， **CPOL = 0**



LSB 对齐标准

该标准与 MSB 对齐标准类似（对于 16 位和 32 位全精度帧格式，没有任何不同）。

图 33-23 **LSB** 对齐的 **16** 位或 **32** 位全精度， **CPOL = 0**

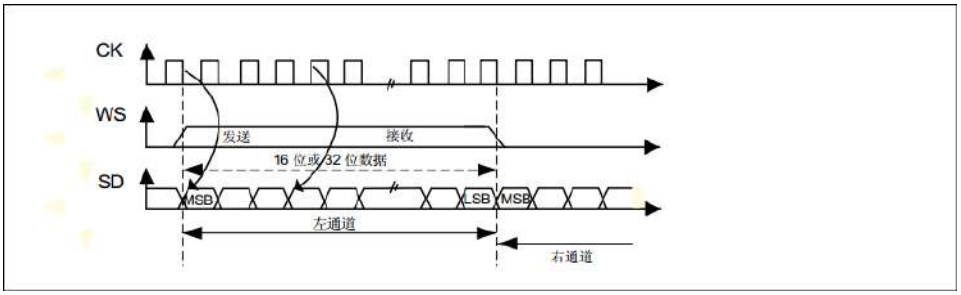
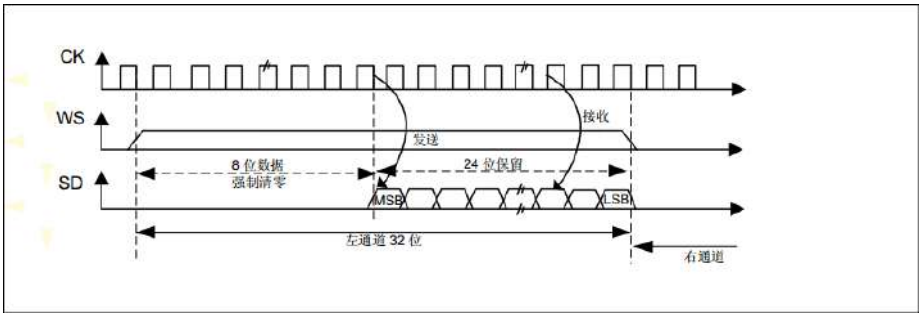


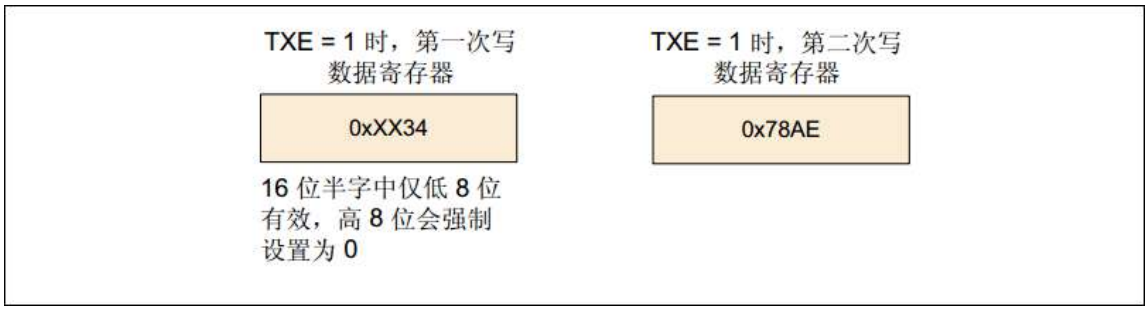
图 33-24 **LSB** 对齐的 **24** 位帧长度， **CPOL = 0**



• 在发送模式下：

如果需要发送数据 0x3478AE，则需要通过软件或 DMA 对 SPIx_DR 寄存器执行两次写入操作。下面给出了这些操作。

图 33-25 发送 **0x3478AE** 所需的操作



• 在接收模式下：

如果接收到数据 0x3478AE，则在每个 RXNE 事件时需要对 SPIx_DR 寄存器执行两次连续的读取操作。

图 33-26 接收 **0x3478AE** 时所需的操作

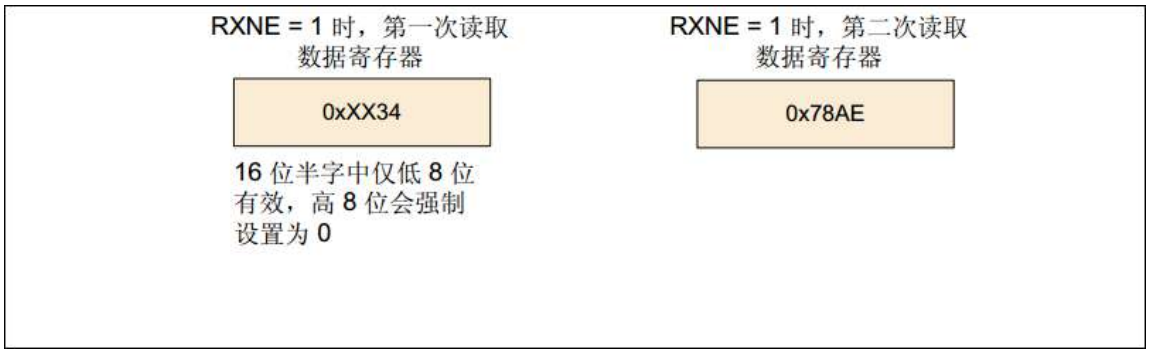
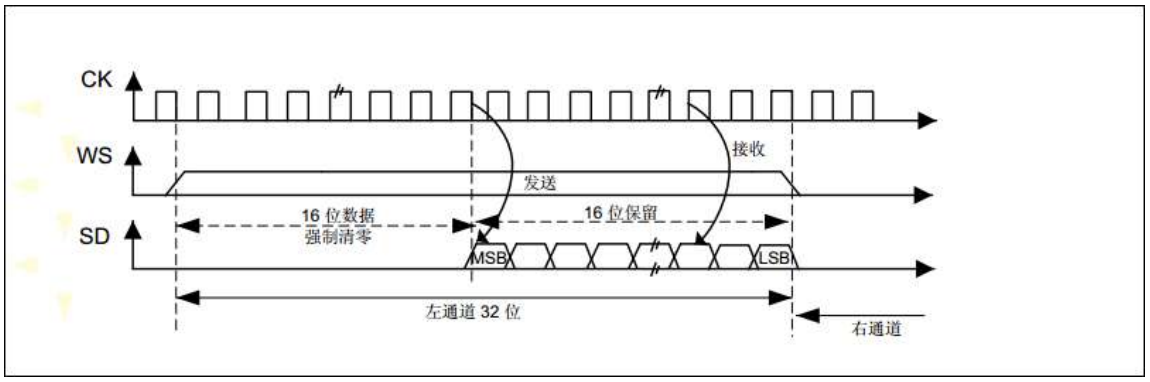


图 33-27 扩展为 **32** 位数据包帧的 **LSB** 对齐的 **16** 位， **CPOL = 0**



如果在 I2S 配置阶段选择将 16 位数据帧扩展到 32 位通道帧，则只需要访问一次 SPIx_DR 寄存器。扩展到 32 位中高半字（16 位 MSB）被硬件置为 0x0000。在这种情况下，其对应于半字 MSB。

如果要发送的数据或已接收的数据为 0x76A3（0x0000 76A3 扩展为 32 位），则需要执行图 33-28 中显示的操作。

图 33-28 **16** 位数据帧扩展到 **32** 位通道帧的示例

在发送模式下，发生 TXE 事件时，应用程序需要写入要发送的数据（此例中，为 0x76A3）。

首先发送 0x000 字段（扩展到 32 位）。有效数据 (0x76A3) 发送到 SD 后， TXE 标志会被再次置 1。

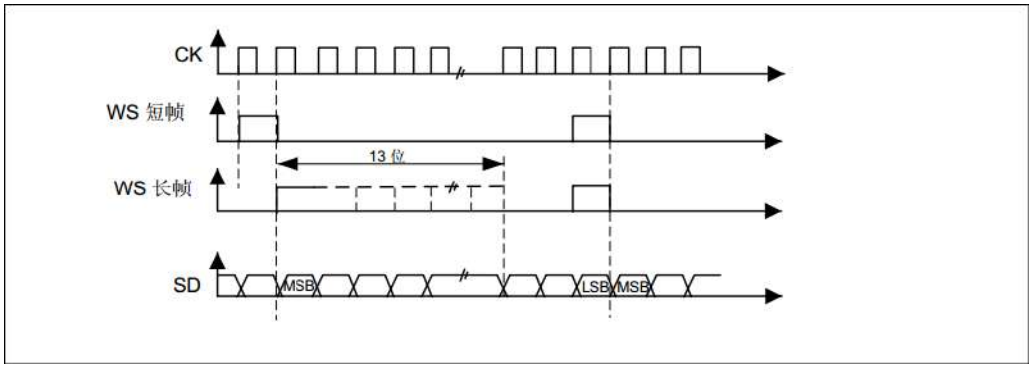
在接收模式下，当接收到有效半字后（而非 0x0000 字段），即会置位 RXNE。

这样，就延长了两个写入或读取操作之间的时间间隔，以防止出现下溢或上溢情况。

PCM 标准

对于 PCM 标准，无需使用通道信息。可使用两种 PCM 模式（短帧和长帧），并且可使用 SPIx_I2SCFGR 寄存器中的 PCMSYNC 位来配置。

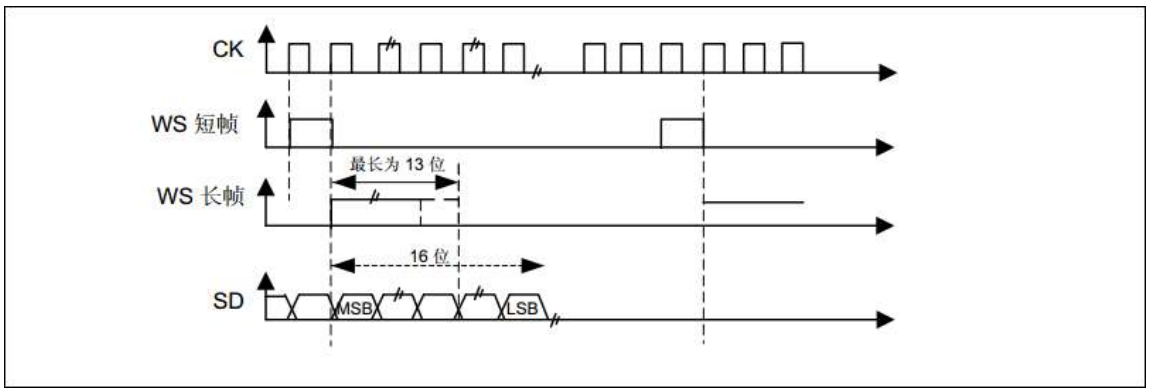
图 33-29 **PCM** 标准波形（**16** 位）



对于长帧同步，在主模式下会将 WS 信号持续 13 个周期。

对于短帧同步， WS 同步信号的持续时间仅为一个周期。

图 33-30 **PCM** 标准波形（**16** 位扩展到 **32** 位数据包帧）



注： 对于两种模式（主 / 从模式）和两种同步（短 / 长同步），即使在从模式下，也需要指定两组连续数据（以及两个同步信号）之间位的个数（SPIx_I2SCFGR 寄存器中的 DATLEN 位和 CHLEN 位）。

33.9.3 时钟发生器

I2S 比特率用来确定 I2S 数据线上的数据流和 I2S 时钟信号频率。

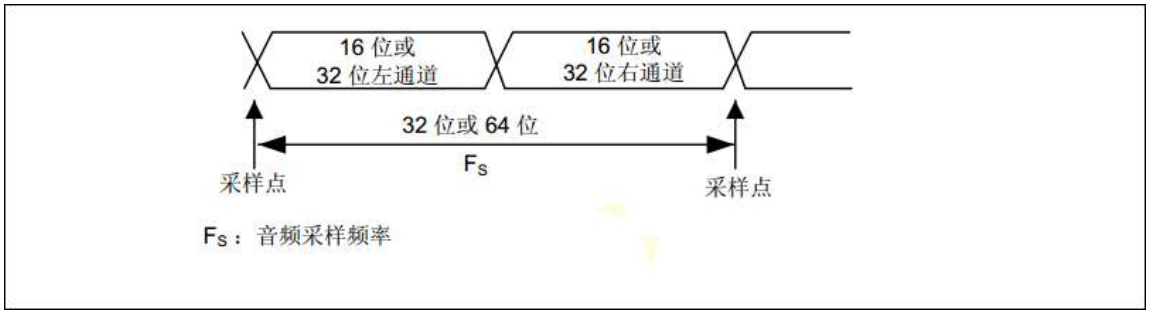
I2S 比特率 = 每个通道的位数 × 通道数 × 音频采样频率

对于 16 位双通道音频， I2S 比特率的计算公式如下：

I2S 比特率 = 16 × 2 × fs

如果数据包为 32 位宽，则 I2S 比特率 = 32 x 2 x fs。

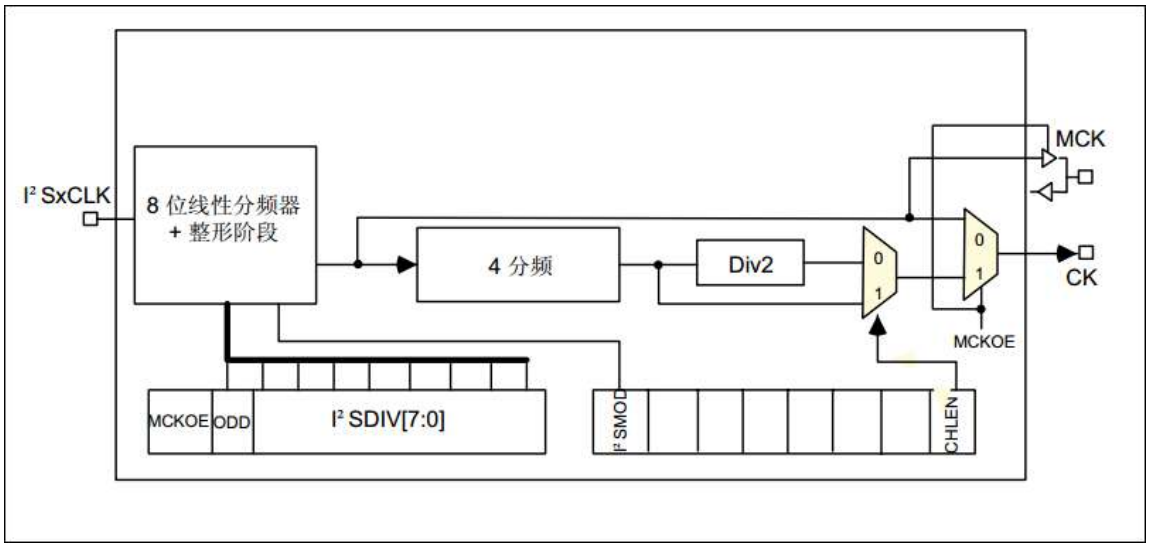
图 33-31 音频采样频率定义



配置主模式时，需要正确地对线性分频器进行设置，以便采用所需的音频频率进行通信。

图 33-32 展示了通信时钟架构。 I2Sx 时钟始终为系统时钟。

图 33-32 **I2S** 时钟发生器架构



1. 其中 x = 2。

音频采样频率可以是 192 KHz、 96 kHz、 48 kHz、 44.1 kHz、 32 kHz、 22.05 kHz、16 kHz、11.025 kHz 或 8 kHz（或此范围内的任何其它值）。为达到所需频率，需要根据以下公式对线性分频器进行编程：

输出主时钟（SPIx_I2SPR 寄存器中的 MCKOE 置 1)时：

fs = I2SxCLK / [(16*2)*((2*I2SDIV)+ODD)*8)]（通道帧宽度为 16 位时）

fs = I2SxCLK / [(32*2)*((2*I2SDIV)+ODD)*4)]（通道帧宽度为 32 位时）

禁止主时钟输出（MCKOE 位清零）时：

fs = I2SxCLK / [(16*2)*((2*I2SDIV)+ODD)]（通道帧宽度为 16 位时）

fs = I2SxCLK / [(32*2)*((2*I2SDIV)+ODD)]（通道帧宽度为 32 位时）

表 33-3 提供了针对不同时钟配置的示例精度值。

注： 还可以采用其它配置以达到更好的时钟精度。

表 33-3 使用标准 **8 MHz HSE** 时的音频频率精度

SYSCLK (MHz)	数据长度	I2SDIV	I2SODD	MCLK	目标 fs (Hz)	实际 fs (kHz)	误差
32	16	5	0	无	96000	100	4.1667%
32	32	2	0	无	96000	100	4.1667%
32	16	10	1	无	48000	47.619	0.7937%

32	32	5	0	无	48000	50	4.1667%
32	16	11	1	无	44100	43.478	1.4098%
32	32	5	1	无	44100	45.454	3.0715%
32	16	15	1	无	32000	32.258	0.8065%
32	32	8	0	无	32000	31.25	2.3430%
32	16	22	1	无	22050	22.222	0.7811%
32	32	11	1	无	22050	21.739	1.4098%
32	16	31	1	无	16000	15.873	0.7937%
32	32	15	1	无	16000	16.129	0.8065%
32	16	45	1	无	11025	10.989	0.3264%
32	32	22	1	无	11025	11.111	0.7811%
32	16	62	1	无	8000	8	0.0000%
32	32	31	1	无	8000	7.936	0.7937%
32	16	2	0	有	32000	31.25	2.3430%
32	32	2	0	有	32000	31.25	2.3430%
32	16	3	0	有	22050	20.833	5.5170%
32	32	3	0	有	22050	20.833	5.5170%
32	16	4	0	有	16000	15.625	2.3428%
32	32	4	0	有	16000	15.625	2.3428%
32	16	5	1	有	11025	11.363	3.0715%
32	32	5	1	有	11025	11.363	3.0715%
32	16	8	0	有	8000	7.812	2.3428%
32	32	8	0	有	8000	7.812	2.3428%

33.9.4 I2S 主模式

I2S 可配置为主模式。这意味着将在 CK 引脚输出串行时钟，在 WS 引脚生成字选信号。主时钟 (MCK) 可以输出，也可以不输出，具体由 SPIx_I2SPR 寄存器中的 MCKOE 位控制。

步骤

1. 设置 SPIx_I2SPR 寄存器的 I2SDIV[7:0] 位，以定义串行时钟波特率，从而达到相应的音频采样频率。 SPIx_I2SPR 寄存器的 ODD 位也需要设置。

2. 设置 CKPOL 位，定义时钟在空闲时的电平状态。如果需要为外部 DAC/ADC 音频组件提供主时钟 MCK，则将 SPIx_I2SPR 寄存器的 MCKOE 位置 1（I2SDIV 和 ODD 值应根据 MCK 输出的状态进行计算。有关详细信息，请参见时钟发生器）。

3. 将 SPIx_I2SCFGR 寄存器中的 I2SMOD 位置 1 以激活 I2S 功能，通过 I2SSTD[1:0] 和 PCMSYNC 位选择 I2S 标准，通过 DATLEN[1:0] 位选择数据长度并通过配置 CHLEN 位选择每个通道的位数。此外，通过 SPIx_I2SCFGR 寄存器的 I2SCFG[1:0] 位选择 I2S 主模式和方向（发送器或接收器）。

4. 如果需要，通过对 SPIx_CR2 寄存器执行写操作来选择所有可能的中断源和 DMA 功能。

5. SPIx_I2SCFGR 寄存器的 I2SE 位必须置 1。

WS 和 CK 配置为输出模式。如果 SPIx_I2SPR 的 MCKOE 位置 1，则 MCK 也是输出。

发送序列

将半字写入发送缓冲区后，发送序列随即开始。

假设写入发送缓冲区的第一个数据对应于左通道数据。数据从发送缓冲区传输到移位寄存器时， TXE 置 1，并且必须将对应于右通道的数据写入发送缓冲区。 CHSIDE 标志指示将发送的数据对应的通道。 TXE 标志置 1 时， CHSIDE 标志有意义，因为该标志在 TXE 变为高电平时进行更新。

一个完整帧表示先进行左通道数据发送再进行右通道数据发送。不存在仅发送左通道的部分帧。

首位发送期间，数据按半字并行加载到 16 位移位寄存器中，然后以串行方式移位并输出到 MOSI/SD 引脚（MSB 在前）。每次数据从发送缓冲区传输到移位寄存器后， TXE 标志都将置 1，如果 SPIx_CR2 寄存器的 TXEIE 位置 1，将产生中断。

有关各种 I2S 标准模式的写操作的更多详细信息，请参见支持的音频协议。

为确保连续进行音频数据发送，必须在当前数据发送结束前将下一个要发送的数据写入 SPIx_DR 寄存器。

要通过将 I2SE 清零来关闭 I2S，必须等待 TXE = 1 且 BSY = 0。

接收序列

此工作模式与发送模式相同，只有第 3 点存在不同（请参见 *I2S* 主模式所述的步骤），即通过 I2SCFG[1:0] 位设置主器件接收模式。

无论数据或通道长度如何，音频数据始终按 16 位数据包进行接收。这意味着，每当接收缓冲区满时，RXNE 标志即置 1，并且如果 SPIx_CR2 寄存器的 RXNEIE 位置 1，还将产生中断。所接收的右通道或左通道的音频值可通过一次或两次接收操作进入接收缓冲区，具体取决于数据和通道长度配置。

读取 SPIx_DR 寄存器即会使 RXNE 位清零。

CHSIDE 在每次接收后进行更新。它由 I2S 单元所产生的 WS 信号触发。

有关各种 I2S 标准模式中读操作的更多详细信息，请参见支持的音频协议。

如果在先前收到的数据尚未读取时又接收到新数据，将产生上溢错误并将 OVR 标志置 1。

如果 SPIx_CR2 寄存器的 ERRIE 位置 1，将产生中断以指示该错误。

要关闭 I2S，需要执行特定操作来确保 I2S 正确完成传输周期而不启动新的数据传输。该序列取决于数据和通道长度的配置，以及所选的音频协议模式。在以下情况下：

• 32 位通道长度上扩展的 16 位数据长度（DATLEN = 00 且 CHLEN = 1），使用 LSB 对齐模式 (I2SSTD = 10)

a) 等待倒数第二个 RXNE = 1 (n – 1)

b) 然后等待 17 个 I2S 时钟周期（使用软件循环）

c) 禁止 I2S (I2SE = 0)

• 32 位通道长度上扩展的 16 位数据长度（DATLEN = 00 且 CHLEN = 1），使用 MSB 对齐、 I2S 或 PCM 模式（分别为 I2SSTD = 00、 I2SSTD = 01 或 I2SSTD = 11）

a) 等待最后一个 RXNE

b) 然后等待 1 个 I2S 时钟周期（使用软件循环）

c) 禁止 I2S (I2SE = 0)

• 对于 DATLEN 和 CHLEN 的所有其它组合，无论通过 I2SSTD 位选择何种音频模式，都将执行以下序列来关闭 I2S：

a) 等待倒数第二个 RXNE = 1 (n – 1)

b) 然后等待一个 I2S 时钟周期（使用软件循环）

c) 禁止 I2S (I2SE = 0)

注： 传输期间， *BSY* 标志保持低电平。

33.9.5 I2S 从模式

对于从配置而言， I2S 可配置为发送模式或接收模式。

此工作模式所遵循的规则与 I2S 主模式配置基本相同。在从模式下， I2S 接口不产生时钟。

时钟和 WS 信号从 I2S 接口所连接的外部主器件输入。这样，用户便不需要配置时钟。

应遵循如下配置步骤：

1. 将 SPIx_I2SCFGR 寄存器的 I2SMOD 位置 1 以选择 I2S 模式，通过 I2SSTD[1:0] 位选择 I2S 标准，通过 DATLEN[1:0] 位选择数据长度并通过配置 CHLEN 位选择帧中每个通道的位数。此外，通过 SPIx_I2SCFGR 寄存器的 I2SCFG[1:0] 位选择从器件的模式（发送或接收）。

2. 如果需要，通过对 SPIx_CR2 寄存器执行写操作来选择所有可能的中断源和 DMA 功能。

3. SPIx_I2SCFGR 寄存器的 I2SE 位必须置 1。

发送序列

当外部主器件发送时钟并且通过 NSS_WS 信号请求传输数据时，发送序列开始。必须首先使能从器件，然后外部主器件才能开始通信。主器件开始通信前，还必须加载 I2S 数据寄存器。

对于 I2S、 MSB 对齐和 LSB 对齐模式，要写入数据寄存器的第一个数据项对应于左通道的数据。通信开始时，数据从发送缓冲区传输到移位寄存器。 TXE 标志随即置 1，以请求将右通道的数据写入 I2S 数据寄存器。

CHSIDE 标志指示将发送的数据对应的通道。与主发送模式相比，在从模式下， CHSIDE 由来自外部主器件的 WS 信号触发。这意味着，从器件需要首先为发送第一个数据做好准备，然后主器件才能产生时钟。 WS 置位意味着首先发送左通道数据。

注：必须要在主器件发出的第一个时钟出现在 *CK* 线上至少 *2* 个 *PCLK* 周期之前置位 *I2SE*。

首位发送期间，数据按半字从内部总线并行加载到 16 位移位寄存器中，然后以串行方式移位并输出到 MOSI/SD 引脚（MSB 在前）。每次数据从发送缓冲区传输到移位寄存器后，TXE 标志都将置 1，如果 SPIx_CR2 寄存器的 TXEIE 位置 1，将产生中断。

请注意，仅当 TXE 标志为 1 时，才可以尝试向发送缓冲区写入数据。

有关各种 I2S 标准模式中写操作的更多详细信息，请参见支持的音频协议。

为确保连续进行音频数据发送，必须在当前数据发送结束前将下一个要发送数据写入 SPIx_DR 寄存器。如果在数据尚未写入 SPIx_DR 寄存器时下一个数据通信的首个时钟边沿到来，下溢标志将置 1 并可能产生中断。通过这种方式，软件可以获知所传输的数据不正确。如果 SPIx_CR2 寄存器的 ERRIE 位置 1，则当 SPIx_SR 寄存器中的 UDR 标志变为 1 时，将产生中断。这种情况下，必须关闭 I2S 并从左通道开始重新启动数据传输。

要通过将 I2SE 位清零来关闭 I2S，必须等待 TXE = 1 且 BSY = 0。

接收序列

此工作模式与发送模式相同，只有第 1 点存在不同（请参见 *I2S* 从模式所述的步骤），即通过 SPIx_I2SCFGR 寄存器的 I2SCFG[1:0] 位设置从器件接收模式。

无论数据长度或通道长度如何，音频数据始终按 16 位数据包进行接收。这意味着，每当接收缓冲区填满时， SPIx_SR 寄存器中的 RXNE 标志即置 1，并且如果 SPIx_CR2 寄存器的 RXNEIE 位置 1，还将产生中断。所接收的右通道或左通道的音频值可能通过一次或两次接收操作进入接收缓冲区，具体取决于数据长度和通道长度配置。

每次接收要从 SPIx_DR 寄存器读取的数据时， CHSIDE 标志都将更新。该标志由外部主器件所管理的外部 WS 线路触发。

读取 SPIx_DR 寄存器即会使 RXNE 位清零。

有关各种 I2S 标准模式的读操作的更多详细信息，请参见支持的音频协议。

如果在先前收到的数据尚未读取时又接收到新数据，将产生上溢错误， OVR 标志将置 1。如果 SPIx_CR2 寄存器的 ERRIE 位置 1，将产生中断以指示该错误。

要在接收模式下关闭 I2S，必须在接收到最后一个 RXNE = 1 后立即将 I2SE 清零。

注： 外部主器件应能够通过音频通道以 *16* 位或 *32* 位数据包发送/接收数据。

33.9.6 I2S 状态标志

应用程序可通过三个状态标志来全面监视 I2S 总线的状态。

忙标志 **(BSY)**

BSY 标志由硬件置 1 和清零（写入此标志没有任何作用）。该标志表示 I2S 通信层的状态。

BSY 置 1 时，表示 I2S 正在忙于通信。在主接收模式 (I2SCFG = 11) 中， BSY 标志的情况例外，该标志在接收期间仍保持低电平。

如果软件需要禁止 I2S，可使用 BSY 标志检测传输是否结束。这可以避免损坏最后传输的数据。为此，必须严格遵循下述步骤。

在传输开始时（I2S 处于主接收器模式时除外），将 BSY 标志置 1。

出现以下情况时， BSY 标志被硬件清零：

- 传输完成时（主发送模式除外，在该模式下通信是连续的）
- 禁止 I2S 时

当通信连续时：

- 在主发送模式下， BSY 标志在所有传输期间均保持高电平
- 在从模式下， BSY 标志在每次传输之间变为低电平并持续一个 I2S 时钟周期

注： 请勿使用 *BSY* 标志处理每次数据发送或接收，最好改用 *TXE* 标志和 *RXNE* 标志。

发送缓冲区为空 **(TXE)**

如果此标志置 1，表示发送缓冲区为空，可将要发送的下一个数据加载到其中。发送缓冲区已包含要发送的数据时， TXE 标志复位。禁止 I2S（I2SE 位复位）时，该标志也会复位。

接收缓冲区非空 **(RXNE)**

此标志置 1 时，表示接收缓冲区中存在有效的已接收数据。读取 SPIx_DR 寄存器时，该标志复位。

通道方向 **(CHSIDE)**

在发送模式下，此标志将在 TXE 变为高电平时进行刷新。此标志指示 SD 上要传输的数据所属的通道。如果在从发送模式下发生下溢错误事件，此标志将不可靠，在恢复通信前需要关闭并重新开启 I2S。

在接收模式下，该标志将在 SPIx_DR 中接收数据时进行刷新。它表示接收的数据所属的通道。请注意，如果发生错误（例如 OVR），此标志将失去意义，应通过禁止并重新使能 I2S（根据需要更改配置）来复位。

此标志在 PCM 标准中没有意义（短帧和长帧模式）。

当 SPIx_SR 中的 OVR 或 UDR 标志置 1，并且 SPIx_CR2 中的 ERRIE 位也置 1 时，将产生中断。中断源被清除后，可通过读取 SPIx_SR 状态寄存器来将此中断清除。

33.9.7 I2S 错误标志

I2S 单元共有三个错误标志。

下溢标志 **(UDR)**

在从发送模式下，如果在软件尚未将任何值加载到 SPIx_DR 之前出现第一个数据发送时钟，此标志将置 1。 SPIx_I2SCFGR 寄存器中的 I2SMOD 位置 1 时，可以使用此标志。如果 SPIx_CR2 寄存器中的 ERRIE 位置 1，可产生中断。

UDR 位通过 SPIx_SR 寄存器上的读操作进行清零。

上溢标志 **(OVR)**

如果在尚未从 SPIx_DR 寄存器读取上一个数据时又接收到新数据，此标志将置 1。因此，传入的数据将丢失。如果 SPIx_CR2 寄存器中的 ERRIE 位置 1，可产生中断。

这种情况下，将不会用接收到的新数据更新接收缓冲区的内容。对 SPIx_DR 寄存器执行的读操作将返回先前正确接收的数据。主器件后续发送的所有其它半字都将丢失。

要将 OVR 位清零，应首先对 SPIx_DR 寄存器执行读操作，然后再对 SPIx_SR 寄存器进行读访问。

帧错误标志 **(FRE)**

仅当 I2S 配置为从模式时，此标志才可由硬件置 1。如果外部主器件没有按照从器件期望的那样改变 WS 线，则此标志将置 1。如果失去同步，要从此状态中恢复并将外部主器件与 I2S 从器件重新同步，需执行下列步骤：

1. 禁止 I2S。
2. 在 WS 线上检测到正确的电平时将其重新使能（WS 线在 I2S 模式下为高电平，在 MSB 对齐、 LSB 对齐或 PCM 模式下为低电平）。

主器件与从器件之间的同步失效可能是由于 SCK 通信时钟或 WS 帧同步信号线上存在噪音干扰。如果 ERRIE 位置 1，可产生错误中断。读取状态寄存器时，同步失效标志 (FRE) 由软件清零。

33.9.8 I2S 中断

表 33-4 为 I2S 中断的列表。

表 33-4 **I2S 中断请求**

中断事件	事件标志	使能控制位
发送缓冲区为空	TXE	TXEIE
接收缓冲区非空	RXNE	RXNEIE
上溢错误	OVR	ERRIE
下溢错误	UDR	
帧错误	FRE	

33.9.9 DMA 特性

在 I2S 模式下， DMA 的工作方式与在 SPI 模式下完全相同。除了由于不存在数据传输保护机制， I2S 模式下没有 CRC 功能外，没有其它差别。

33.10 SPI 寄存器

33.10.1 SPI 控制寄存器 1（SPIx_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
BIDIMODE	BIDIOE	CRCEN	CRCNEXT	CRCL	RXONLY	SSM	SSI	LSBFIRST	SPE	BR[2:0]			MSTR	CPOL	CPHA
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位 15	BIDIMODE： 双向数据模式使能 0: 选择 2 线的单向数据模式 1: 选择单线双向数据模式
位 14	BIDIOE： 双向模式输出使能 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)。
位 13	CRCEN： 硬件 CRC 计算使能 0: CRC 计算禁用 1: CRC 计算使能
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。
位 11	CRCL： CRC 长度 由软件设置和清零，用于选择 CRC 长度。 0: 8 位 CRC 长度 1: 16 位 CRC 长度

位 10	RXONLY： 只接收 和 BIDIMODE 位一起决定在“2 线单向”模式下数据的输出方向。 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: 输出禁止 (只收模式)；
位 9	SSM： 软件从机管理 当 SSM 被置位时，NSS 引脚上的电平由 SSI 位的值决定。 0: 禁止软件从设备管理； 1: 启用软件从设备管理
位 8	SSI： 内部从设备选择 此位只有当 SSM 位为 1 的时候才有效果。它决定了 NSS 上的电平，在 NSS 引脚 上的 I/O 操作无效。
位 7	LSBFIRST： 帧格式 0: 先发送 MSB； 1: 先发送 LSB。
位 6	SPE：SPI 使能 0: 禁止 SPI 设备； 1: 开启 SPI 设备。
位 5:3	BR[2:0]：波特率控制 000: fPCLK/2 001: fPCLK/4 010: fPCLK/8 011: fPCLK/16 100: fPCLK/32 101: fPCLK/64 110: fPCLK/128 111: fPCLK/256
位 2	MSTR：主设备选择 0: 配置为从设备 1: 配置为主设备
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。
位 0	CPHA：时钟相位 0: 第一个时钟沿对准第一位数据 1: 第二和时钟沿对准第一位数据

33.10.2 SPI 控制寄存器 2（SPIx_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
Res.	LDMA_TX	LDMA_RX	FRXTH	DS[3:0]				TXEIE	RXNEIE	ERRIE	RFR	NSSP	SSOE	TXDMAEN	RXDMAEN
rw	rw	rw	rw	rw				rw	rw	rw	rw	rw	rw	rw	rw
0	0	0	0	0				0	0	0	0	0	0	0	0

位 15	保留，必须保持复位时的值。

位 14	LDMA_TX：最后一次 DMA 发送 此位是在数据打包模式中，用来定义 DMA 数据传输的总数是奇数还是偶数。 它只有在 SPIx_CR2 寄存器的 TXDMAEN 位被设置，并且打包模式已开启（数据长度小于等于 8 位和写入访问 SPIx_DR 是 16 位宽）时有意义。它必须在 SPI 被禁止（SPIx_ CR1 寄存器的 SPE = 0）时写入。 0: 传输的数据数量为偶数 1: 传输的数据数量为奇数
位 13	LDMA_RX： 最后一次 DMA 接收 此位是在数据打包模式中，用来定义 DMA 数据接收的总数是奇数还是偶数。 它只 有在 SPIx_CR2 寄存器的 RXDMAEN 位被设置，并且打包模式已开启（数据长度小 于等于 8 位和写入访问 SPIx_DR 是 16 位宽）时有意义。它必须在 SPI 被禁止（SPIx_ CR1 寄存器的 SPE = 0）时写入。 0: 传输的数据数量为偶数 1: 传输的数据数量为奇数
位 12	FRXTH： FIFO 接收门限 此位用来设置触发 RXNE 事件时的 RXFIFO 的阈值。 0: 如果 FIFO 的存储水平大于或等于 1/2（16 位），产生 RXNE 事件。 1: 如果 FIFO 的存储水平大于或等于 1/4（8 位），产生 RXNE 事件。
位 11:8	DS[3:0] 数据位宽 这些位配置 SPI 传输数据的位宽： 0000: 不使用 0001: 不使用 0010: 不使用 0011: 4 位 0100: 5 位 0101: 6 位 0110: 7 位 0111: 8 位 1000: 9 位 1001: 10 位 1010: 11 位 1011: 12 位 1100: 13 位 1101: 14 位 1110: 15 位 1111: 16 位 如果软件试图写一个“不使用”的值，他们会被迫赋值“0111”（8 位）。
位 7	TXEIE： TX 缓冲器空中断使能 0: TXE 中断屏蔽 1: TXE 中断没有被屏蔽。 用于在 TXE 标志置 1 的时候产生一个中断请求。
位 6	RXNEIE： RX 缓冲区非空中断使能 0: RXNE 中断屏蔽 1: TXE 中断没有被屏蔽。 用于在 RXNE 标志置 1 的时候产生一个中断请求。
位 5	ERRIE： 错误中断使能 这个位控制在出现错误事件（CRCERR，OVR，SPI 模式中的 MODF，TI 模式中的 FRE 和 UDR，I2S 模式中的 OVR）时是否产生中断。 0: 错误中断屏蔽 1: 错误中断使能
位 4	FRF： 帧格式 0: SPI Motorola 模式 1：SPI TI 模式
位 3	NSSP： NSS 脉冲管理 该位仅在主模式下使用。它允许 SPI 在连续传输时，两个数据传输之间产生一个 NSS 脉冲。 在单个数据传输的情况下，它会在传输结束后将 NSS 脚强制为高电平。在 CPHA=1 或 FRF=1 的时候，这个位没有什么意义。 0: 没有 NSS 脉冲 1: 产生 NSS 脉冲

位 2	SSOE： SS 输出使能 0: 在主模式下 SS 输出被禁用，SPI 接口可以工作在多主机的配置下。 1: SPI 接口启用的同时在主模式下启用 SS 输出。SPI 接口不能在多主环境下工作。
位 1	TXDMAEN： Tx 缓冲 DMA 使能 当此位被设置，TXE 标志被设置时，产生一个 DMA 请求。 0: Tx 缓冲 DMA 禁止 1: Tx 缓冲 DMA 使能
位 0	RXDMAEN：Rx 缓冲 DMA 使能 当此位被设置，RXNE 标志被设置时，产生一个 DMA 请求。 0: Rx 缓冲 DMA 禁止 1: Rx 缓冲 DMA 使能

33.10.3 SPI 状态寄存器（SPIx_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
Res.			FTLVL[1:0]		FRLVL[2:0]		FRE	BSY	OVR	MODF	CRCERR	UDR	CHSIDE	TXE	RXNE
			r		r		r	r	r	r	rc_w0	r	r	r	r
0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

位 15:13	保留，必须保持复位时的值。
位 12:11	FTLVL [1:0]： FIFO 发送存储水平，由硬件设置或清零 00: FIFO 空 01: 1/4 FIFO 10: 1/2 FIFO 11: FIFO 满（当 FIFO 门限大于 1/2 时认为是满）
位 10:9	FRLVL [1:0]： FIFO 接收存储水平 由硬件设置或清零 00: FIFO 空 01: 1/4 FIFO 10: 1/2 FIFO 11: FIFO 满
位 8	FRE： TI 帧格式错误 0: 没发生帧格式错误 1: 发生了一个帧格式错误
位 7	BSY： 忙标志 0: SPI（或 I2S）不忙 1: （SPI 或 I2S）通信忙或发送缓冲区不为空 由硬件设置和清除。
位 6	OVR： 溢出标志 0: 没有发生溢出 1: 发生溢出 此标志由硬件置位，由软件序列复位。
位 5	MODF： 模式故障 0: 无模式故障发生 1: 模式故障发生 此标志由硬件置位，由软件序列复位。
位 4	CRCERR： CRC 错误标志 0: 收到的 CRC 值和 SPIx_RXCRCR 的值是匹配的 1: 收到的 CRC 值和 SPIx_RXCRCR 值不匹配 此标志由硬件置位，由软件清零。
位 3	UDR：欠载标志 0: 没有欠载发生 1: 欠载发生 此标志由硬件置位，由软件序列复位。

位 2	CHSIDE： 通道标志 0: 需要传输或者接收左声道； 1: 需要传输或者接收右声道；
位 1	TXE： 发送缓冲区为空标志 0: Tx 缓冲区非空 1: TX 缓冲器空
位 0	RXNE： 接收缓冲区非标志 0: RX 缓冲区空 1: Rx 缓冲非空

33.10.4 SPI 数据寄存器（SPIx_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															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位 15:0	DR[15:0]: 数据寄存器 待发送或者已经收到的数据，数据寄存器作为 Rx 和 Tx FIFOs 的接口。 当读取数据寄存器时，RXFIFO 会 被访问，而写入数据寄存器则会访问 TXFIFO。

33.10.5 SPI 的 CRC 多项式寄存器（SPIx_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															
0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1

位 15:0	CRCPOLY [15:0]： CRC 多项式寄存器 该寄存器包含 CRC 计算多项式。根据需要，可以配置成另一个多项式。

33.10.6 SPI 接收 CRC 寄存器（SPIx_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															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位 15:0	RXCRC [15:0]： RX CRC 寄存器 当启用 CRC 计算功能，RxCRC [15:0]位包含根据收到的字节计算出来的 CRC 值。 当 SPIx_CR1 寄存器的 CRCEN 位被写为 1 的时候，这个寄存器被复位。CRC 计算使用 SPI_CRCPR 中的多项式。 当数据帧格式被设置为 8 位（SPIx_CR1 的 CRCL 位为 0）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（SPIx_CR1 的 CRCL 位为 1）时，寄存器中的所有 16 位都参与计算，并且按照 CRC16 的方法进行；

33.10.7 SPI 发送 CRC 寄存器（SPIx_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															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位 15:0	TxCRC[15:0]: Tx CRC 寄存器 当启用 CRC 计算功能，TxCRC [7:0]位包含根据发送的字节计算出来的 CRC 值。 当 SPIx_CR1 寄存器的 CRCEN 位被写为 1 的时候，这个寄存器被复位。CRC 计算使用 SPI_CRCPR 中的多项式。 当数据帧格式被设置为 8 位（SPIx_CR1 中的 CRCL 位为 0）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（SPIx_CR1 的 CRCL 位为 1）时，寄存器中的所有 16 位都参与计算，并且按照 CRC16 的方法进行；

33.10.8 SPIx_I2S 配置寄存器（SPIx_I2SCFGR）

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
Res.	Res.	Res.	Res.	I2SMOD	I2SE	I2SCFG		PCMSYNC	Res.	I2SSTD		CKPOL	DATLEN		CHLEN
				rw	rw	rw	rw	rw		rw		rw	rw		rw
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位 15:12	保留： 由硬件强制为 0
位 11	I2SMOD： I2S 模式选择 0: 选中 SPI 模式 1: 选中 I2S 模式
位 10	I2SE： I2S 使能 0: I2S 的外设被禁用 1: I2S 外设被使能
位 9:8	I2SCFG： I2S 配置模式 00: 从机 - 发送 01: 从机 - 接收 10: 主机 - 发送 11: 主机 - 接收

位 7	PCMSYNC： PCM 帧同步 0: 短帧同步 1: 长帧同步
位 6	保留： 由硬件强制为 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
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；
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
位 0	CHLEN： 声道长度（每个音频通道的数据位数） 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。

33.10.9 SPIx_I2S 预分频寄存器（SPIx_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
Res.	Res.	Res.	Res.	Res.	Res.	MCKOE	ODD	I2SDIV							
						rw	rw	rw							
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

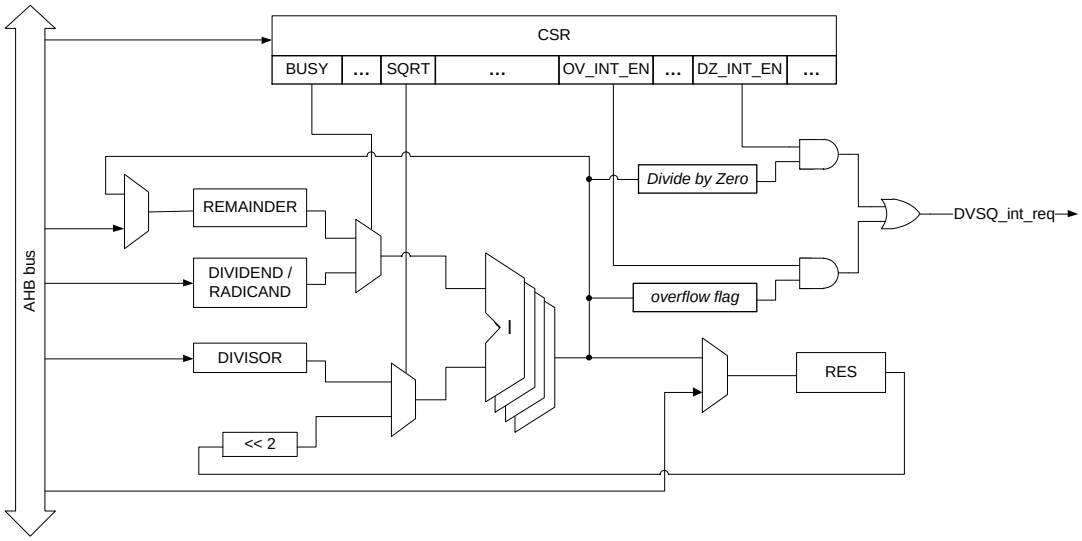
位 15:10	保留： 由硬件强制为 0
位 9	MCKOE： 主时钟输出使能 0: 主时钟输出被禁用 1: 主时钟输出启用
位 8	ODD： 奇系数预分频 0: 实际分频系数 = I2SDIV *2； 1: 实际分频系数 = （I2SDIV *2）+1
位 7:0	I2SDIV： I2S 线性预分频器 不允许设置 I2SDIV [7:0] =0 或者 I2SDIV [7:0] =1 的值。

34 除法开方运算器 (DIVSQRT)

34.1 简介

DVSQ 加速器支持 32/32 的无符号除法(SDIV)和带符号除法(UDIV)，和 32 位无符号数开方运算。除法运算同时支持 MOD 操作，运算完成后，商和余数同时更新到相应的寄存器中供软件读取。

- 支持 32 位带符号数和无符号数除法，支持 32 位开方运算。在同一时刻，DVSQ 加速器不能同时支持除法和开方运算，只能在两个中选取一个执行。32/32 有符号/无符号整数除法(同时获取商和余数)。无符号开方运算，可以通过软件选择高精度开方运算。
- 流水线设计，每个时钟完成 2bits 运算。
- 运算时间根据运算数据不同而改变。
- 支持除零中断和溢出中断。
  - 结构如下图



34.2 除法操作描述

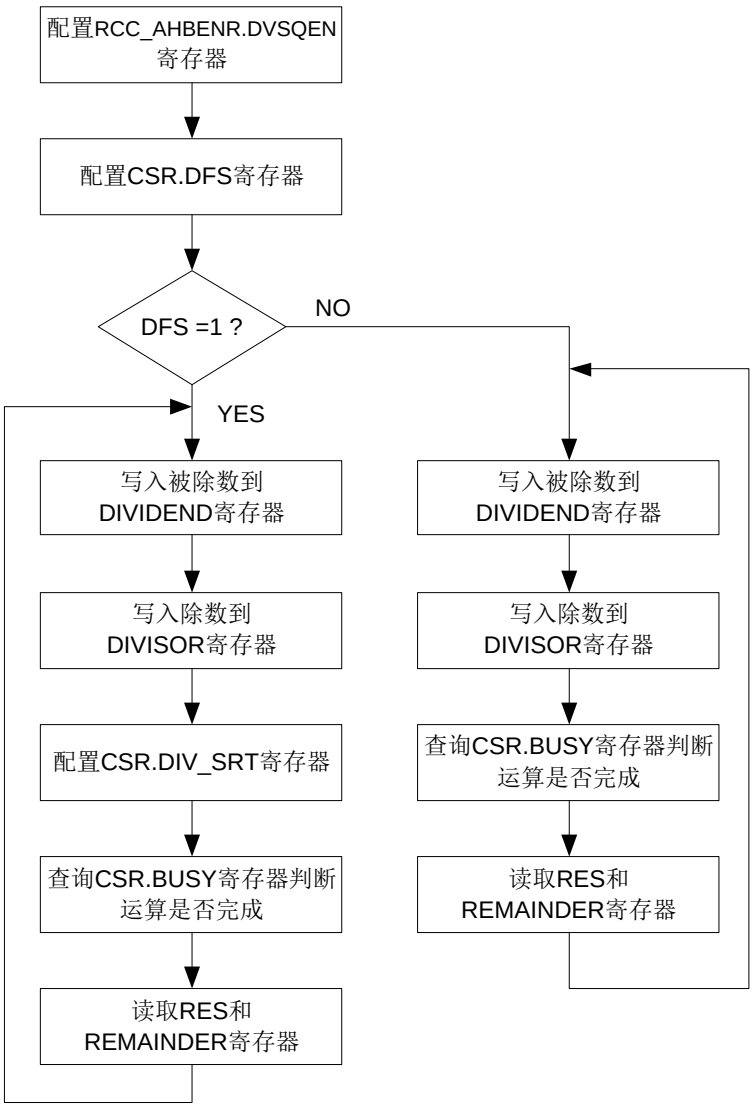
除法操作

结果寄存器 RES 和余数寄存器 REMAINDER 中保持的值始终为补码格式。如果执行无符号除法运算，则 RES 寄存器和 REMAINDER 寄存器中的值都为正数。如果执行带符号数除法，则 RES 寄存器和 REMAINDER 寄存器的符号位由输入的操作数决定：

- 如果被除数和除数的符号位不同，则商为负数；否则商为正数。
- 余数的符号位始终和除数的符号位相同。

操作流程

可以通过快速启动和非快速启动和非快速启动两种方式启动除法运算。除法运算的操作流程示意图为：



34.3 除法运行时间

DVSQ 加速器通过被除数的数据决定运算时间，以尽快得到运算结果。具体运算时间如下表，其中运算时间定义为从运算开始到得到运算结果的时钟周期数，也就是 CSR.BUSY 为高的持续时间。

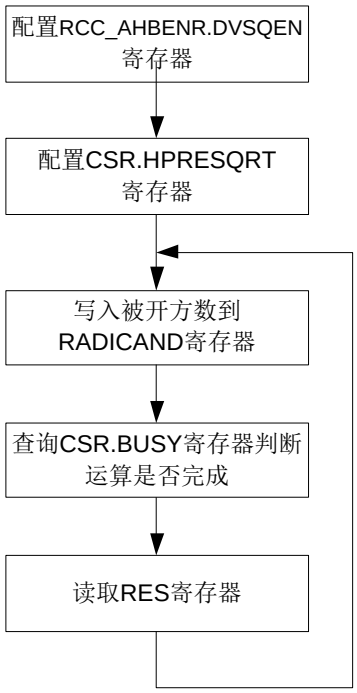
被除数的绝对值	运算时间[周期数]
(01,1x)xx_xxxx_xxxx_xxxx__xxxx_xxxx_xxxx_xxxx	17
00(01,1x)_xxxx_xxxx_xxxx__xxxx_xxxx_xxxx_xxxx	16
0000_(01,1x)xx_xxxx_xxxx__xxxx_xxxx_xxxx_xxxx	15
0000_00(01,1x)_xxxx_xxxx__xxxx_xxxx_xxxx_xxxx	14
0000_0000_(01,1x)xx_xxxx__xxxx_xxxx_xxxx_xxxx	13
0000_0000_00(01,1x)_xxxx__xxxx_xxxx_xxxx_xxxx	12
0000_0000_0000_(01,1x)xx__xxxx_xxxx_xxxx_xxxx	11
0000_0000_0000_00(01,1x)__xxxx_xxxx_xxxx_xxxx	10

0000_0000_0000_0000_(01,1x)xx_xxxx_xxxx_xxxx	9
0000_0000_0000_0000_0000_(01,1x)xx_xxxx_xxxx	8
0000_0000_0000_0000_0000_(01,1x)xx_xxxx_xxxx	7
0000_0000_0000_0000_0000_00(01,1x)_xxxx_xxxx	6
0000_0000_0000_0000_0000_0000_(01,1x)xx_xxxx	5
0000_0000_0000_0000_0000_0000_00(01,1x)_xxxx	4
0000_0000_0000_0000_0000_0000_0000_(01,1x)xx	3
0000_0000_0000_0000_0000_0000_0000_00(01,1x)	2
0000_0000_0000_0000_0000_0000_0000_0000	1

34.4 开方操作描述

DVSQ 加速器只能进行无符号是开方，因此进行开方运算的时候 RADICAND 和 RES 中的值都是无符号数。

操作流程



34.5 开方运行时间

DVSQ 加速器根据被开方数决定运算时间，同时 CSR.HPRESQRT 也会影响开方运算时间。具体时间如下表，其中运算时间定义为从运算开始到得到运算结果的时钟周期数，也就是 CSR.BUSY 为高的持续时间。

当 CSR.HPRESQRT 为 0 时：：

被开方数	运算时间[周期数]
(01,1x)xx_xxxx_xxxx_xxxx__xxxx_xxxx_xxxx_xxxx	17
00(01,1x)_xxxx_xxxx_xxxx__xxxx_xxxx_xxxx_xxxx	16

0000_(01,1x)xx_xxxx_xxxx__xxxx_xxxx_xxxx_xxxx	15
0000_00(01,1x)_xxxx_xxxx__xxxx_xxxx_xxxx_xxxx	14
0000_0000_(01,1x)xx_xxxx__xxxx_xxxx_xxxx_xxxx	13
0000_0000_00(01,1x)_xxxx__xxxx_xxxx_xxxx_xxxx	12
0000_0000_0000_(01,1x)xx__xxxx_xxxx_xxxx_xxxx	11
0000_0000_0000_00(01,1x)__xxxx_xxxx_xxxx_xxxx	10
0000_0000_0000_0000_(01,1x)xx_xxxx_xxxx_xxxx	9
0000_0000_0000_0000_0000_(01,1x)xx_xxxx_xxxx	8
0000_0000_0000_0000_0000_(01,1x)xx_xxxx_xxxx	7
0000_0000_0000_0000_0000_00(01,1x)_xxxx_xxxx	6
0000_0000_0000_0000_0000_0000_(01,1x)xx_xxxx	5
0000_0000_0000_0000_0000_0000_00(01,1x)_xxxx	4
0000_0000_0000_0000_0000_0000_0000_(01,1x)xx	3
0000_0000_0000_0000_0000_0000_0000_00(01,1x)	2
0000_0000_0000_0000_0000_0000_0000_0000	1

当 CSR.HPRESQRT 为 1 时：

被开方数	运算时间[周期数]
(01,1x)xx_xxxx_xxxx_xxxx__xxxx_xxxx_xxxx_xxxx	33
00(01,1x)_xxxx_xxxx_xxxx__xxxx_xxxx_xxxx_xxxx	32
0000_(01,1x)xx_xxxx_xxxx__xxxx_xxxx_xxxx_xxxx	31
0000_00(01,1x)_xxxx_xxxx__xxxx_xxxx_xxxx_xxxx	30
0000_0000_(01,1x)xx_xxxx__xxxx_xxxx_xxxx_xxxx	29
0000_0000_00(01,1x)_xxxx__xxxx_xxxx_xxxx_xxxx	28
0000_0000_0000_(01,1x)xx__xxxx_xxxx_xxxx_xxxx	27
0000_0000_0000_00(01,1x)__xxxx_xxxx_xxxx_xxxx	26
0000_0000_0000_0000_(01,1x)xx_xxxx_xxxx_xxxx	25
0000_0000_0000_0000_0000_(01,1x)xx_xxxx_xxxx	24
0000_0000_0000_0000_0000_(01,1x)xx_xxxx_xxxx	23
0000_0000_0000_0000_0000_00(01,1x)_xxxx_xxxx	22
0000_0000_0000_0000_0000_0000_(01,1x)xx_xxxx	21
0000_0000_0000_0000_0000_0000_00(01,1x)_xxxx	20
0000_0000_0000_0000_0000_0000_0000_(01,1x)xx	19
0000_0000_0000_0000_0000_0000_0000_00(01,1x)	18
0000_0000_0000_0000_0000_0000_0000_0000	1

34.6 中断

DVSQ 加速器内部的两个中断源共用芯片中的同一个中断号，当系统检测到中断请求后需要通过读 CSR 寄存器来判断是除零中断还是溢出中断。在同一时刻，除零中断和溢出中断只可能有一个发生。

带符号数除法溢出中断：

- 可以通过配置 CSR.OV_INT_EN 使能或关闭。
- 当带符号数除法被除数为 0x8000_0000，除数为 0xFFFF_FFFF 时，中断请求会被硬件置位。

- 本中断情况可以通过软件或硬件清除：
  - 软件配置 CSR.OV_FLAG 为 0。
  - 开始下一次除法或开方运算。

除 0 中断：

- 可以通过配置 CSR.DZ_INT_EN 使能或关闭。
- 当除数为 0 时，中断请求会被硬件置位。
- 本中断请求可以通过软件或硬件清除：
  - 软件配置 CSR.DZ_FLAG 为 0。
  - 开始下一次除法或开方运算。

34.7 注意事项

除法操作：

- 因为 DIVIDEND 寄存器和 DIVISOR 寄存器的值在运算过程中不会被硬件改变，所以软件通过字节或半字写这两个寄存器的时候需要小心。比如软件字节写 DIVIDEND[7：0]后，DIVIDEND 寄存器中的高 24 位为上一次除法运算写入的值，低 8 位为新写入的值。
- 当除法配置为快速启动后，无论是字节写、半字写还是字写 DIVISOR 寄存器都会使除法运算开始。

开方运算：

- 无论是字节写、半字写还是字写 DIVISOR 寄存器都会使除法运算开始。

结果读取：

- 如果在 DVSQ 还没有完成运算的时候访问 RES 和 REMAINDER 寄存器，将会使总线处于等待状态，在等待状态中也不能相应中断。软件可以通过轮询 CSR.BUSY 的状态来判断 RES 和 REMAINDER 的值是否已经就绪。

如果在 DVSQ 还没有完成运算的时候访问 DIVIDEND/DIVISOR/RADICAND 寄存器也会使总线处于等待状态。

34.8 DIVSQRT 寄存器

DIVSQRT 的寄存器基址为 0x40030000.

34.8.1 被除数寄存器 (DIVIDEND)

软件配置除法运算所需的被除数值。如果在 DVSQ 加速器处于工作态的时候对 DIVIDEND 寄存器进行读写访问，则总线将处于等待状态直到 DVSQ 加速器的操作完成。DIVIDEND 寄存器只能被软件写访问更新，硬件运算不会更新 DIVIDEND 寄存器。

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

位 31:0	**DIVIDEND:**被除数寄存器 读操作：读出寄存器里存储的值。 写操作：更新除法运算的被除数。

34.8.2 除数寄存器 (DIVISOR)

软件配置除法运算所需的除数值。当 CSR.DFS = 0 时，如果软件写 DIVISOR 寄存器会立即使能除法运算。如果在 DVSQ 加速器处于工作态的时候对 DIVISOR 寄存器进行读写访问，则总线将处于等待状态直到 DVSQ 加速器的操作完成。DIVISOR 寄存器只能被软件写访问更新，硬件运算不会更新 DIVISOR 寄存器。

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

位 31:0	**DIVISOR**:除数寄存器 读操作：读出寄存器里存储的值。 写操作：更新除法运算的被除数。

34.8.3 控制和状态寄存器(CSR)

本寄存器控制除法和开方运算工作，并返回 DVSQ 加速器的工作状态。CSR 寄存器的值会被 DVSQ 加速器硬件更新，软件可以轮询 CSR 寄存器来判断除法和开方运算是否已经完成。如果在 DVSQ 加速器处于工作态的时候对 CSR 寄存器进行写操作，则总线将处于等待状态直到 DVSQ 加速器的操作完成。

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
BUSY	DIV	SQRT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r	r	r	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HPRESQRT	OV_FLAG	OV_INT_EN	DZ_FLAG	DZ_INT_EN	DFS	UNSIGN_DIV	DIV_SRT
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位 31	**BUSY**: DVSQ 加速器工作状态标志 0:DVSQ 加速器处于空闲状态。 1:DVSQ 加速器处于工作状态，表示有除法或开方运算正在工作。 [bit31]BUSY 信号与 [bit30]DIV 和 [bit29]SQRT 结合起来可以标识 DVSQ 加速器的工作状态，具体当 [BUSY, DIV, SQRT] 为不同值时表示的含义为： 000: 标识 DVSQ 加速器处于空闲态，并且还没有执行过除法或开方操作。 001: 标识 DVSQ 加速器处于空闲态，并且 DVSQ 加速器完成的上一个运算为开方运算。 010: 标识 DVSQ 加速器处于空闲态，并且 DVSQ 加速器完成的上一个运算为除法运算。 101: 标识 DVSQ 加速器处于工作态，并且正在进行的是开方运算。 110: 标识 DVSQ 加速器处于工作态，并且正在进行的是除法运算。 [BUSY, DIV, SQRT]的值的其余组合没有定义。
位 30	**DIV:** 除法运算标志 0:表示 DVSQ 加速器进行的上一个运算或者当前运算不是除法运算。 1:表示 DVSQ 加速器进行的上一个运算或者当前运算为除法运算。
位 29	**SQRT**: 开方运算标志 0:表示 DVSQ 加速器进行的上一个运算或者当前运算不是开方运算。 1:表示 DVSQ 加速器进行的上一个运算或者当前运算为开方运算。
位 7	**HPRESQRT**: 高精度开方运算选择寄存器 写操作： 0:DVSQ 执行普通精度开方运算。 1:DVSQ 执行高精度开方运算。 读操作：读出寄存器里存储的值。 注：见本表格说明部分。
位 6	**OV_FLAG**: 带符号数除法溢出标志 写操作： 0:清除 OV_FLAG 标志。 1:无效操作。 读操作： 0：上一次除法运算没有发生溢出。 1：上一次除法运算溢出。 在进行带符号数除法的时候，如果被除数为 0x8000_0000，同时除数为 0XFFFF_FFFF，则结果超出了 32 位补码的表示范围。在这种情况下，DVSQ 加速器置位 OV_FLAG，并且返回的商为 0x8000_0000，余数为 0x0000_0000。 OV_FLAG 可以被软件清除，如果发生带符号数除法溢出，OV_FLAG 一直保持为高，直到被软件清除，或者 DVSQ 加速器开始下一次除法或开方运算。
位 5	**OV_INT_EN**: 带符号数除法溢出中断使能 写操作： 0:关闭带符号数除法溢出中断。 1:使能带符号数除法溢出中断。 读操作：读出寄存器里存储的值。
位 4	**DZ_FLAG:** 除数为 0 标志 写操作： 0:清除 DZ_FLAG 标志。 1:无效操作。 读操作： 0:上一次运算的除数不为 0。 1:上一次运算的除数为 0。 无论是带符号数除法还是不带符号数除法，当除数为‘0’时，DVSQ 加速器都会把 DZ_FLAG 置位为 1,并且返回的商和余数都为 0x0000_0000。 DZ_FLAG 可以被软件清除，如果发生除数为‘0’的情况，DZ_FLAG 保持为高，直到被软件清除，或者 DVSQ 加速器开始下一次除法或开方运算。

位 3	**DZ_INT_EN**: 除数为 0 中断使能 写操作： 0:关闭除数为 0 中断。 1:使能除数为 0 中断。 读操作：读出寄存器里存储的值。
位 2	**DFS**: 快速启动除法运算关闭 写操作： 0:使能快速启动除法运算。 1:关闭快速启动除法运算。 读操作：读出寄存器里存储的值。 DVSQ 加速器支持两种方式启动除法运算。默认方式为‘快速启动’，如果对 DIVISOR 进行写操作，就立即启动除法运算。另一种启动方式为：软件置位 CSR[DIV_SRT]寄存器后才开始除法运算。由 DFS 位选择使用哪一种启动方式。
位 1	**UNSIGN_DIV**: 无符号数除法 写操作： 0:执行带符号数除法。 1:执行无符号是除法。 读操作：读出寄存器里存储的值。
位 0	**DIV_SRT**: 开始除法运算 写操作： 0:无效操作。 1:如果 **CSR[DFS]** = 1，则开始除法运算；否则不进行任何下一步动作。 读操作：一直返回为 0。

说明：

软件可以使用 Q notation 来增加开方运算的精度。关于 Q notation 的具体描述请参考 Wikipedia。

如果使用 Q notation 来进行开方运算，无符号 Qm.n 记号法需要 m+n 位作为被开方数(m+n = 32)，产生的结果为 Q(m/2).(n/2)格式。这占用 n 位被开方数作为小数位 (n < 31)，因此如果软件需要高精度开方结果的时候，将比较大的影响被开方数的表示范围。

DVSQ 加速器提供了一种同时满足高精度开方和被开方数表示范围的方法，当 HPRESQRT 被置位为 1 时，DVSQ 加速器内部附加 32 位‘0’到被开方数寄存器后面，即：RADICAND ([RADICAND.0x00000000])。然后把这个新的数作为被开方数带入开方运算单元进行运算。这时被开方数为 Q32.32，开方结果为 Q16.16。这样被开方数的结果不会受到影响，并且同时达到高精度开方的要求。如果 HPRESQRT 被设置为‘0’，则被开方数的小数位由软件决定，并且满足 m+n=32。

34.8.4 被开方数寄存器(RADICAND)

软件通过写 RADICAND 寄存器配置开方运算的被开方数。如果在 DVSQ 加速器处于工作态的时候对 RADICAND 寄存器进行读写访问，则总线将处于等待状态直到 DVSQ 加速器的操作完成。RADICAND 寄存器只能被软件写访问更新，硬件运算不会更新 RADICAND 寄存器。

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

位 31:0	**RADICAND**:被开方数寄存器读操作：读出寄存器里存储的值。 写操作：更新开方运算所需的被开方数。 读操作：读出寄存器里存储的值。

34.8.5 结果寄存器 (RES)

RES 寄存器存储除法运算的商或开方运算的平方根结果。在 DVSQ 加速器完成除法或开方运算后会自动更新 RES 寄存器。如果在 DVSQ 加速器处于工作态的时候对 RES 寄存器进行读写访问，则总线将处于等待状态直到 DVSQ 加速器的操作完成。在 DVSQ 寄存器进行运算的时候有可能被系统中断服务程序打断，软件需要保存 DVSQ 加速器的上下文。因此 RES 寄存器和 REMAINDER 寄存器可以被软件写操作更新。

偏移地址：0x10

复位值：0x0000_0000

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

位 31:0	**RES**:结果寄存器 写操作：更新结果寄存器中的值。 读操作：读出寄存器里存储的值。

34.8.6 余数寄存器(REMAINDER)

REMAINDER 寄存器保存除法运算的余数结果。DVSQ 加速器进行除法运算或开方运算的时候都会自动更新 REMAINDER 寄存器。如果在 DVSQ 加速器处于工作态的时候对 REMAINDER 寄存器进行读写访问，则总线将处于等待状态直到 DVSQ 加速器的操作完成。

偏移地址：0x14

复位值：0x0000_0000

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

位 31:0	**REMAINDER**:余数寄存器 写操作：更新结果寄存器中的值。 读操作：读出寄存器里存储的值。

35 调试支持 (DBG)

35.1 简介

HK32L08x产品集成了一个MCU ID Code,这个ID是用来区分HK MCU不同的partnumber和die版本。这个ID可以通过SWD接口或者用户软件来读取。

35.2 DBGMCU 寄存器

35.2.1 MCU 器件 ID 代码 (DBGMCU_IDCODE)

地址： 0x40015800

仅支持 32 位访问。只读

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

位 31:16	**REV_ID:** 版本标识符 (Revision identifier) 0x1000: 版本编号 1.0 0x2000: 版本编号 2.0
位 15:12	保留，必须保持为复位值。
位 11:0	**DEV_ID:** 器件标识符 (Device identifier) 复位后从 flash 加载。 该字段指定设备 ID: 0x506

35.2.2 调试 MCU 配置寄存器 (DBG_CR)

偏移地址： 0x04

仅支持 32 位访问

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_STANDBY	DBG_STOP	DBG_SLEEP
													r	r	r
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位 31:3	保留，必须保持为复位值。
位 2	**DBG_STANDBY:** 调试待机模式 (Debug Standby mode) 0：（FCLK=关闭， HCLK=关闭）将整个数字部分断电。 从软件角度来看，退出待机模式相当于取复位向量（指示 MCU 从待机状态恢复的几个状态位除外） 1：（FCLK=开启， HCLK=开启）在这种情况下，数字部分不断电， FCLK 和 HCLK 由仍保持激活状态的内部 RC 振荡器提供。此外， MCU 会在待机模式期间产生系统复位，这样退出待机模式相当于取复位向量
位 1	**DBG_STOP:** 调试停止模式 (Debug Stop mode) 0：（FCLK=关闭， HCLK=关闭）在停机模式下，时钟控制器禁止所有时钟（包括 HCLK 和 FCLK）。当退出停止模式时，时钟配置与 RESET 后的时钟配置相同。因此，软件必须重新编程时钟控制器以使能 PLL 和晶振等。 1：（FCLK=开启， HCLK=开启）在这种情况下，进入停机模式时， FCLK 和 HCLK 由在停机模式下仍保持激活状态的内部 RC 振荡器提供。退出停机模式时，软件必须重新编程时钟控制器以使能 PLL 和晶振等（操作步骤与在 DBG_STOP=0 时相同）
位 0	**DBG_SLEEP:** 调试睡眠模式 (Debug Sleep mode) 0：在睡眠模式下， FCLK 由原先在 HCLK 禁止时通过软件配置好的系统时钟驱动。时钟控制器配置不复位，保持先前设置的状态。因此，退出睡眠模式时，软件不需要重新配置时钟控制器。 1：在这种情况下，进入睡眠模式时，将为 HCLK 和 FCLK 馈送相同的时钟（软件先前配置的系统时钟）。

35.2.3 调试 MCU APB1 冻结寄存器(DBG_APB1_FZ)

当 MCU 处于调试状态下时， DBG_APB1_FZ 寄存器用于配置以下 APB 外设：

- 定时器时钟计数器冻结
- I2C SMBUS 超时冻结
- 系统窗口看门狗和独立看门狗计数器冻结支持。

偏移地址： 0x08

仅支持 32 位访问

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DBG_LPTIM1_STOP	DBG_LPTIM2_STOP	DBG_LPTIM3_STOP	Res.	Res.	Res.	Res.	Res.	Res.	DBG_I2C2_SMBUS_TIMEOUT	DBG_I2C1_SMBUS_TIMEOUT	Res.	Res.	Res.	Res.	Res.
rw	rw	rw							rw	rw					
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	DBG_IWDG_STOP	DBG_WWDG_STOP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_TIM3_STOP	DBG_TIM2_STOP
			rw	rw										rw	rw
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位 31	**DBG_LPTIM1_STOP:** 内核停止时 LPTIM1 计数器停止 (LPTIM1 counter stopped when core is halted) 0：即使内核停止，仍然馈送 LPTIM1 计数器时钟 1：内核停止时， LPTIM1 计数器时钟停止

位 30	**DBG_LPTIM2_STOP:** 内核停止时 LPTIM2 计数器停止 (LPTIM2 counter stopped when core is halted) 0：即使内核停止，仍然馈送 LPTIM2 计数器时钟 1：内核停止时， LPTIM2 计数器时钟停止
位 29	**DBG_LPTIM3_STOP:** 内核停止时 LPTIM3 计数器停止 (LPTIM3 counter stopped when core is halted) 0：即使内核停止，仍然馈送 LPTIM3 计数器时钟 1：内核停止时， LPTIM3 计数器时钟停止
位 28:23	保留，必须保持复位值。
位 22	**DBG_I2C2_SMBUS_TIMEOUT:** 内核停止时停止 I2C2 SMBUS 超时模式 (I2C2 SMBUS timeout mode stopped when core is halted) 0：行为方式与正常模式下相同 1：冻结 I2C1 SMBUS 超时
位 21	**DBG_I2C1_SMBUS_TIMEOUT:** 内核停止时停止 I2C1 SMBUS 超时模式 (I2C1 SMBUS timeout mode stopped when core is halted) 0：行为方式与正常模式下相同 1：冻结 I2C1 SMBUS 超时
位 20:13	保留，必须保持为复位值。
位 12	**DBG_IWDG_STOP:** 内核停止时停止调试独立看门狗 (Debug independent watchdog stopped when core is halted) 0：即使内核停止，独立看门狗计数器时钟仍继续工作 1：内核停止时，独立看门狗计数器时钟停止
位 11	**DBG_WWDG_STOP:** 内核停止时停止调试窗口看门狗 (Debug window watchdog stopped when core is halted) 0：即使内核停止，窗口看门狗计数器时钟仍继续工作 1：内核停止时，窗口看门狗计数器时钟停止
位 10:2	保留，必须保持为复位值。
位 1	**DBG_TIM3_STOP:** 内核停止时 TIM3 计数器停止 (TIM3 counter stopped when core is halted) 0：即使内核停止，仍然馈送 TIM3 计数器时钟 1：内核停止时， TIM3 计数器时钟停止
位 0	**DBG_TIM2_STOP:** 内核停止时 TIM2 计数器停止 (TIM2 counter stopped when core is halted) 0：即使内核停止，仍然馈送 TIM2 计数器时钟 1：内核停止时，TIM2 计数器时钟停止

35.2.4 调试 MCU APB2 冻结寄存器 (DBG_APB2_FZ)

偏移地址： 0x0c

仅支持 32 位访问。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	DBG_TIM1_STOP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
				rw											
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位 31:12	保留，必须保持复位值。
位 11	**DBG_TIM1_STOP:** 内核停止时 TIM1 计数器停止 (TIM21 counter stopped when core is halted) 0：即使内核停止，仍然馈送 TIM1 计数器时钟 1：内核停止时，TIM1 计数器时钟停止

位 10:0	保留，必须保持复位值。

36 设备电子签名

除非特别说明，否则本部分适用于所有 HK32L08x 器件。
电子签名存储在 Flash 模块的系统存储区中，可以使用 JTAG/SWD 或 CPU 对其进行读取。
它包含出厂前编程的标识数据，这些标识数据允许用户固件或其它外部设备将其接口与 HK32L08x 微控制器的特性自动匹配。

36.1 存储器大小寄存器

36.1.1 Flash 大小寄存器

Flash size register
基址： 0x1FF8 007C
只读 = 0xXXXX，其中 X 是出厂前编程的

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

位 15:0	**F_SIZE:** Flash 大小 (Flash memory size) 此字段中存储的值指示器件的 Flash 大小（用 KB 表示）。 示例： 0x0040 = 64 KB。

36.2 唯一设备 ID 寄存器（96 位）

唯一设备标识符最适合：

- 用作序列号
- 在对内部 Flash 进行编程前将唯一 ID 与软件加密原语和协议结合使用时用作安全密钥以提高 Flash 中代码的安全性
- 激活安全自举过程等

96 位的唯一设备标识符提供了一个对于任何设备和任何上下文都唯一的参考号码。用户永远不能改变这些位。
96 位的唯一设备标识符也可以以单字节/半字/字等不同方式读取，然后使用自定义算法连接起来。

基址： 0x1FF8 0050
偏移地址： 0x00
只读 = 0xXXXX XXXX，其中 X 是出厂前编程的

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
U_ID(31:16)															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
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
X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

位 31:0	**FU_ID(31:24):** WAF_NUM[7:0] 晶圆编号（8 位无符号数） **U_ID(23:0)**: LOT_NUM[55:32] 批号（ASC Ⅱ代码）

偏移地址 ： 0x04

只读 = 0xXXXX XXXX，其中 X 是出厂前编程的

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
U_ID(63:48)															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
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
X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

位 31:0	**U_ID(63:32)**: LOT_NUM[31:0] 批号（ASC Ⅱ代码）

偏移地址： 0x14

只读 = 0xXXXX XXXX，其中 X 是出厂前编程的

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
U_ID(95:80)															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
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
X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

位 31:0	**U_ID(95:64):**95:64 唯一 ID 位

37 重要提示

在未经深圳市航顺芯片技术研发有限公司同意下不得以任何形式或途径修改本公司产品规格和数据表中的任何部分以及子部份。深圳市航顺芯片技术研发有限公司在以下方面保留权利：修改数据单和/或产品、停产任一产品或者终止服务不做通知；建议顾客获取最新版本的相关信息，在下定订单前进行核实以确保信息的及时性和完整性。所有的产品都依据订单确认时所提供的销售合同条款出售，条款内容包括保修范围、知识产权和责任范围。

深圳市航顺芯片技术研发有限公司保证在销售期间，产品的性能按照本公司的标准保修。公司认为有必要维持此项保修，会使用测试和其他质量控制技术。除了政府强制规定外，其他仪器的测量表没有必要进行特殊测试。

顾客认可本公司的产品的设计、生产的目的不涉及与生命保障相关或者用于其他危险的活动或者环境的其他系统或产品中。出现故障的产品会导致人身伤亡、财产或环境的损伤（统称高危活动）。人为在高危活动中使用本公司产品，本公司据此不作保修，并且不对顾客或者第三方负有责任。

深圳市航顺芯片技术研发有限公司将会提供与现在一样的技术支持、帮助、建议和信息，（全部包括关于购买的电路板或其他应用程序的设计，开发或调试）。特此声明，对于所有的技术支持、可销性或针对特定用途，及在支持技术无误下，电路板和其他应用程序可以操作或运行的，本公司将不作任何有关此类支持技术的担保，并对您在使用这项支持服务不负任何法律责任。

X-ON Electronics

Largest Supplier of Electrical and Electronic Components

Click to view similar products for 32-bit Microcontrollers - MCU ***category:***

Click to view products by HK ***manufacturer:***

Other Similar products are found below :

MCF51AC256AVFUE MCF51AC256BCFUE MCF51AC256BVFUE MB91F464AAPMC-GSE2 R5S726B0D216FP#V0 MB91F248PFV-GE1 MB91243PFV-GS-136E1 SAK-TC1782F-320F180HR BA TC364DP64F300WAAKXUMA1 R5F566NNDDFP#30 R5F566NNDDFC#30 R5F566NNDDBD#20 MC96F8216ADBN A96G181HDN A96G140KNN A96G174FDN A31G213CL2N A96G148KNN A96G174AEN AC33M3064TLBN-01 V3s T3 A40i-H V526 A83T R11 V851s A133 V833 F1C100S T3L T507 A33 A63 T113-i H616 V853 V533 V536-H A64-H V831 V3LP T113-S3 F1C200S F133-A R128-S2 ADUCM360BCPZ128-TR APT32S003F8PT AT32F435VMT7 AT32F435CGT7