用户手册

User Manual

MM32F103xx

32 位基于 ARM ® Cortex ® M3 核心的微控制器

版本：1.69_n

目录

插图

表格

1 存储器和总线构架

存储器和总线构架

1.1 系统构架

主系统由以下部分构成：

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

这些都是通过一个多级的 AHB 总线构架相互连接的，如图 1所示：

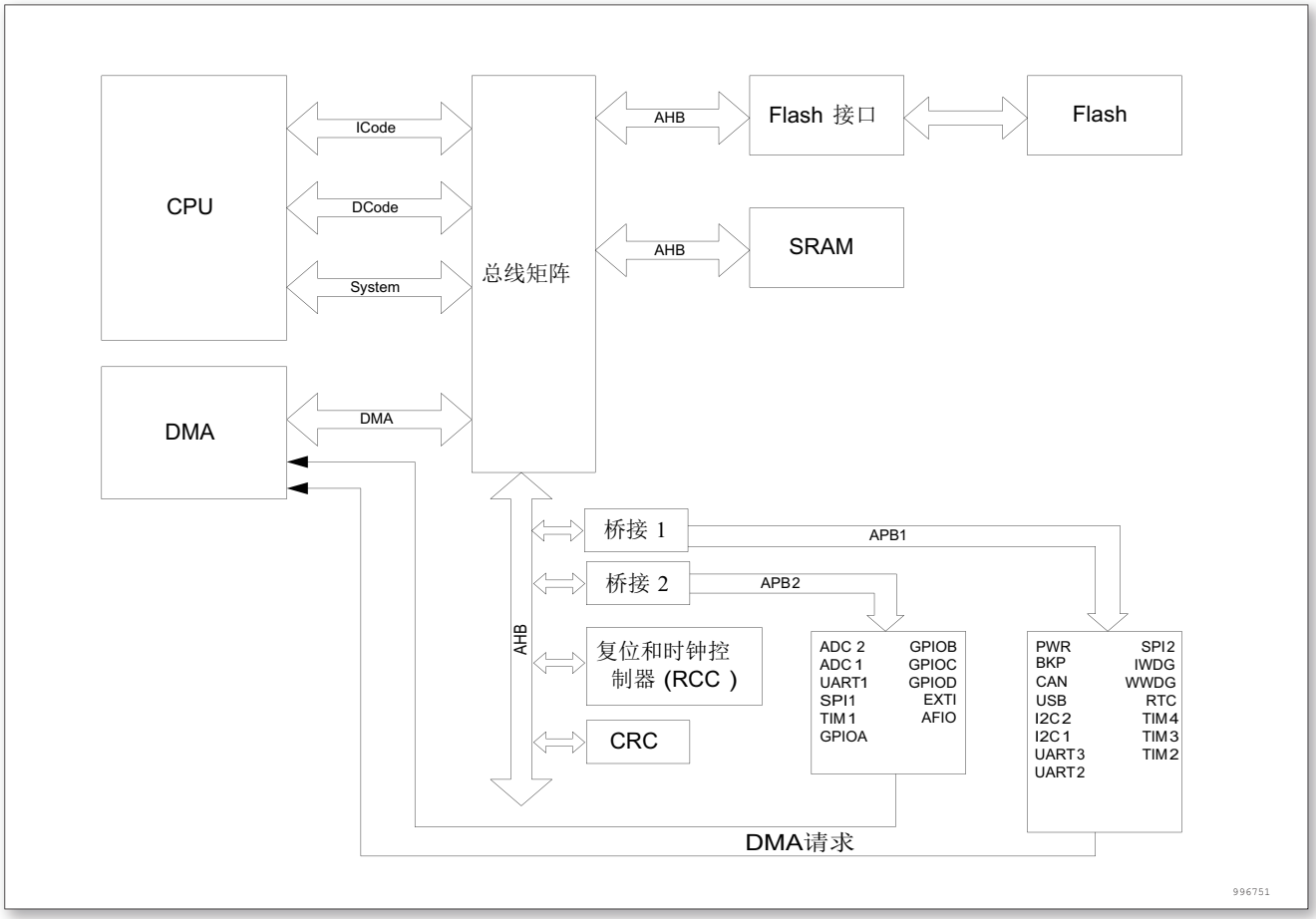


图 1. 系统架构

ICode 总线

该总线将 CPU 内核的指令总线与闪存指令接口相连接。指令预取在此总线上完成。

DCode 总线

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

系统总线

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

DMA 总线

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

总线矩阵 (BusMatrix)

总线矩阵管理着内核系统总线与 DMA 总线的访问仲裁，总线矩阵由主模块总线及从模块总线组成。

AHB 外设通过总线矩阵与系统总线相连，允许 DMA 访问。

AHB 到 APB 桥 (AHB2APB bridges - APB)

AHB 到 APB 桥在 AHB 与 APB 总线间提供同步连接。在每次复位之后，所有的外设时钟都关闭 (除了 SRAM 及 Flash 外)。在用一个外设前，你必须打开相应的 RCC_AHBENR、RCC_APB2ENR 或 RCC_APB1ENR 寄存器中时钟使能位。

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

1.2 存储器组织

1.2.1 介绍

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。详细请参考存储器映像和寄存器编址章节和外设章节。

1.2.2 存储器映像和寄存器编址

存储器映像请参考各外设对应章节中的存储器映像图。

下表给出了所有内置外设的起始地址。

表 1. 存储器映像

总线	编址范围	大小	外设	备注
Flash	0x0000 0000 - 0x0001 FFFF	128KB	主闪存存储器，系统存储器或是 SRAM 有赖于 BOOT 的配置	
	0x000 20000 - 0x07FF FFFF	~128KB	Reserved	
	0x0800 0000 - 0x0801 FFFF	128KB	Main Flash memory	

总线	编址范围	大小	外设	备注
Flash	0x0802 0000 - 0x0FFF FFFF	~128KB	Reserved	
	0x1000 0000 - 0x1000 1FFF	8KB	Reserved	
	0x1000 2000 - 0x1FFD FFFF	~256KB	Reserved	
	0x1FFE 0000 - 0x1FFE 01FF	0.5KB	Protect byte	
	0x1FFE 0200 - 0x1FFE 0FFF	3KB	Reserved	
	0x1FFE 1000 - 0x1FFE 1BFF	3KB	Security space	
	0x1FFE 1C00 - 0x1FFF F3FF	~256KB	Reserved	
	0x1FFF F400 - 0x1FFF F7FF	1KB	System memory	
	0x1FFF F800 - 0x1FFF F80F	16KB	Option bytes	
	0x1FFF F810 - 0x1FFF FFFF	~2KB	Reserved	
SRAM	0x2000 0000 - 0x2000 4FFF	20KB	SRAM	
	0x2000 5000 - 0x3FFF FFFF	~512MB	Reserved	
APB1	0x4000 0000 - 0x4000 03FF	1KB	TIM2	
	0x4000 0400 - 0x4000 07FF	1KB	TIM3	
	0x4000 0800 - 0x4000 0BFF	1KB	TIM4	
	0x4000 0C00 - 0x4000 27FF	7KB	Reserved	
	0x4000 2800 - 0x4000 2BFF	1KB	RTC	
	0x4000 2C00 - 0x4000 2FFF	1KB	WWDG	
	0x4000 3000 - 0x4000 33FF	1KB	IWWDG	
	0x4000 3400 - 0x4000 37FF	1KB	Reserved	
	0x4000 3800 - 0x4000 3BFF	1KB	SPI2	
	0x4000 3C00 - 0x4000 43FF	2KB	Reserved	
	0x4000 4400 - 0x4000 47FF	1KB	UART2	
	0x4000 4800 - 0x4000 4BFF	1KB	UART3	
	0x4000 4C00 - 0x4000 53FF	2KB	Reserved	
	0x4000 5400 - 0x4000 57FF	1KB	I2C1	
	0x4000 5800 - 0x4000 5BFF	1KB	I2C2	
	0x4000 5C00 - 0x4000 5FFF	1KB	USB	
	0x4000 6000 - 0x4000 63FF	1KB	Reserved	
	0x4000 6400 - 0x4000 67FF	1KB	CAN	
	0x4000 6800 - 0x4000 6BFF	1KB	Reserved	
	0x4000 6C00 - 0x4000 6FFF	1KB	后备寄存器 (BKP)	
	0x4000 7000 - 0x4000 73FF	1KB	电源控制 (PWR)	
	0x4000 7400 - 0x4000 77FF	34KB	Reserved	
	0x4000 7800 - 0x4000 FFFF	34KB	Reserved	
APB2	0x4001 0000 - 0x4001 03FF	1KB	AFIO	
	0x4001 0400 - 0x4001 07FF	1KB	EXTI	
	0x4001 0800 - 0x4001 0BFF	1KB	GPIOA	
	0x4001 0C00 - 0x4001 0FFF	1KB	GPIOB	
	0x4001 1000 - 0x4001 13FF	1KB	GPIOC	
	0x4001 1400 - 0x4001 17FF	1KB	GPIOD	

总线	编址范围	大小	外设	备注
APB2	0x4001 1800 - 0x4001 1BFF	1KB	Reserved	
	0x4001 1C00 - 0x4001 23FF	2KB	Reserved	
	0x4001 2400 - 0x4001 27FF	1KB	ADC1	
	0x4001 2800 - 0x4001 2BFF	1KB	ADC2	
	0x4001 2C00 - 0x4001 2FFF	1KB	TIM1	
	0x4001 3000 - 0x4001 33FF	1KB	SPI1	
	0x4001 3400 - 0x4001 37FF	1KB	Reserved	
	0x4001 3800 - 0x4001 3BFF	1KB	UART1	
	0x4001 3C00 - 0x4001 3FFF	1KB	Reserved	
	0x4001 4000 - 0x4001 43FF	1KB	Reserved	
	0x4001 4400 - 0x4001 47FF	1KB	Reserved	
	0x4001 4800 - 0x4001 4BFF	1KB	Reserved	
	0x4001 4C00 - 0x4001 7FFF	13KB	Reserved	
	0x4001 8000 - 0x4001 FFFF	32KB	Reserved	
AHB	0x4002 0000 - 0x4002 03FF	1KB	DMA	
	0x4002 0400 - 0x4002 0FFF	3KB	Reserved	
	0x4002 1000 - 0x4002 13FF	1KB	复位和时钟控制 (RCC)	
	0x4002 1400 - 0x4002 1FFF	3KB	Reserved	
	0x4002 2000 - 0x4002 23FF	1KB	Flash 接口	
	0x4002 2400 - 0x4002 2FFF	3KB	Reserved	
	0x4002 3000 - 0x4002 33FF	1KB	CRC	
	0x4002 3400 - 0x4002 43FF	4KB	Reserved	

1.3 内置的 SRAM

内置最大可到 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。SRAM 起始地址为 0x2000 0000。

- 数据总线上最大可到 20K 字节的 SRAM。可以被 CPU 或者 DMA 用最快的系统时钟且不插入任何等待进行访问。

1.4 闪存存储器概述

闪存存储器有两个不同存储区域：

- 主闪存存储块，它包括应用程序和用户数据区 (若需要时)
- 信息块，其包含四个部分：
  - 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 (System memory) －其包含 boot loader 代码。参见内置闪存存储器章节。

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。

1.5 启动配置 (Boot configuration)

可通过 BOOT0 及 BOOT1 脚的配置选择三种不同的启动模式，如下表所示。

表 2. 启动模式

启动模式选择		启动模式	说明
BOOT1	BOOT0		
x	0	主闪存存储器	主闪存存储器选为启动区域
0	1	系统存储器	系统存储器选为启动区域
1	1	内置 SRAM	内置 SRAM 选为启动区域

器件复位后，在 SYSCLK 的第 4 个上升沿锁存 BOOT0 和 BOOT1 的引脚值，用户可通过设置 BOOT1 和 BOOT0 来选择启动模式。

从待机模式唤醒时，CPU 会得新采样 BOOT0 及 BOOT1 的引脚值，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 0x0000 0004 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 (0x0000 0000)，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 F400) 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。

内嵌的自举程序

内嵌的自举程序存放在系统存储器，由厂家在生产时写入。该程序可以通过 UART1 对闪存进行重新编程。

2 嵌入式闪存 (FLASH)

嵌入式闪存 (FLASH)

2.1 闪存主要特性

- 高达 128K 字节闪存存储器

闪存接口的特性为：

- 带预取缓冲器的数据接口 (2 × 64 位)
- 选择字节加载器
- 闪存编程/擦除操作
- 访问/写保护
- 低功耗模式

2.2 闪存功能描述

2.2.1 闪存结构

闪存空间由 64 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) 分块，以扇区为单位设置写保护 (参见存储保护相关内容)。

表 3. Flash 模块结构

模块	名称	地址	大小 (字节)
主存储块	页 0	0x0800 0000 - 0x0800 03FF	1K
	页 1	0x0800 0400 - 0x0800 07FF	1K
	页 2	0x0800 0800 - 0x0800 0BFF	1K
	页 3	0x0800 0C00 - 0x0800 0FFF	1K
	…	…	…
	…	…	…
	页 28	0x0800 7000 - 0x0800 73FF	1K
	页 29	0x0800 7400 - 0x0800 77FF	1K
	页 30	0x0800 7800 - 0x0800 7BFF	1K
	页 31	0x0800 7C00 - 0x0800 7FFF	1K
	…	…	…
	…	…	…
	页 123	0x0801 EC00 - 0x0801 EFFF	1K
	页 124	0x0801 F000 - 0x0801 F3FF	1K
	页 125	0x0801 F400 - 0x0801 F7FF	1K
	页 126	0x0801 F800 - 0x0801 FBFF	1K

模块	名称	地址	大小 (字节)
主存储块	页 127	0x0801 FC00 - 0x0801 FFFF	1K
信息块	保护字节	0x1FFE 0000 - 0x1FFE 01FF	0.5K
	保密空间	0x1FFE 1000 - 0x1FFE 1BFF	3K
	系统存储器	0x1FFF F400 - 0x1FFF F7FF	1K
	选项字节	0x1FFF F800 - 0x1FFF F80F	16
闪存存储器接口寄存器	FLASH_ACR	0x4002 2000 - 0x4002 2003	4
	FLASH_KEYR	0x4002 2004 - 0x4002 2007	4
	FLASH_OPTKEYR	0x4002 2008 - 0x4002 200B	4
	FLASH_SR	0x4002 200C - 0x4002 200F	4
	FLASH_CR	0x4002 2010 - 0x4002 2013	4
	FLASH_AR	0x4002 2014 - 0x4002 2017	4
	保留	0x4002 2018 - 0x4002 201B	4
	FLASH_OBR	0x4002 201C - 0x4002 201F	4
	FLASH_WRPR	0x4002 2020 - 0x4002 2023	4

2.2.2 FLASH 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
- 潜伏期：等待位的个数，保证正确的读取。

取指

CPU 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

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

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打开的。只有在 SYSCLK 低于 24MHz，并且 AHB 时钟没有经过任何分频的条件下 (SYSCLK 必须等于 HCLK) 才可以开/关预取指缓冲区。通常情况下，预取指缓冲区在初始化过程中就已经决定好开关状态了，而当时 MCU 运行在内部 8MHz 的振荡器下。

注：当 AHB 时钟的预分频器不等于 1 时，预取指缓冲区必须打开访问潜伏期。

访问潜伏期

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。

2.2.3 Flash 写和擦除操作

嵌入式闪存支持在线编程以及在应用编程。

ICP 是指使用 SWD 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(在应用编程) 能够使用 MCU 支持的任何通信接口 (I/Os，USB，UART，I2C，SPI，等等) 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/ISP 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。该操作有下列 7 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)

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 1 = 0x45670123
- 写关键字 2 = 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。KEY1 出错会立即中断，KEY1 正确但 KEY2 错误时会在 KEY2 错的时候引发中断。

主闪存编程

主闪存一次可以编程 16 位。当 FLASH_CR 中的 PG 位为 1 时，直接对相应的地址写一个半字 (16 位)，就是一次编程操作。如果试图写别的长度而不是半字，将引起硬件错误中断。

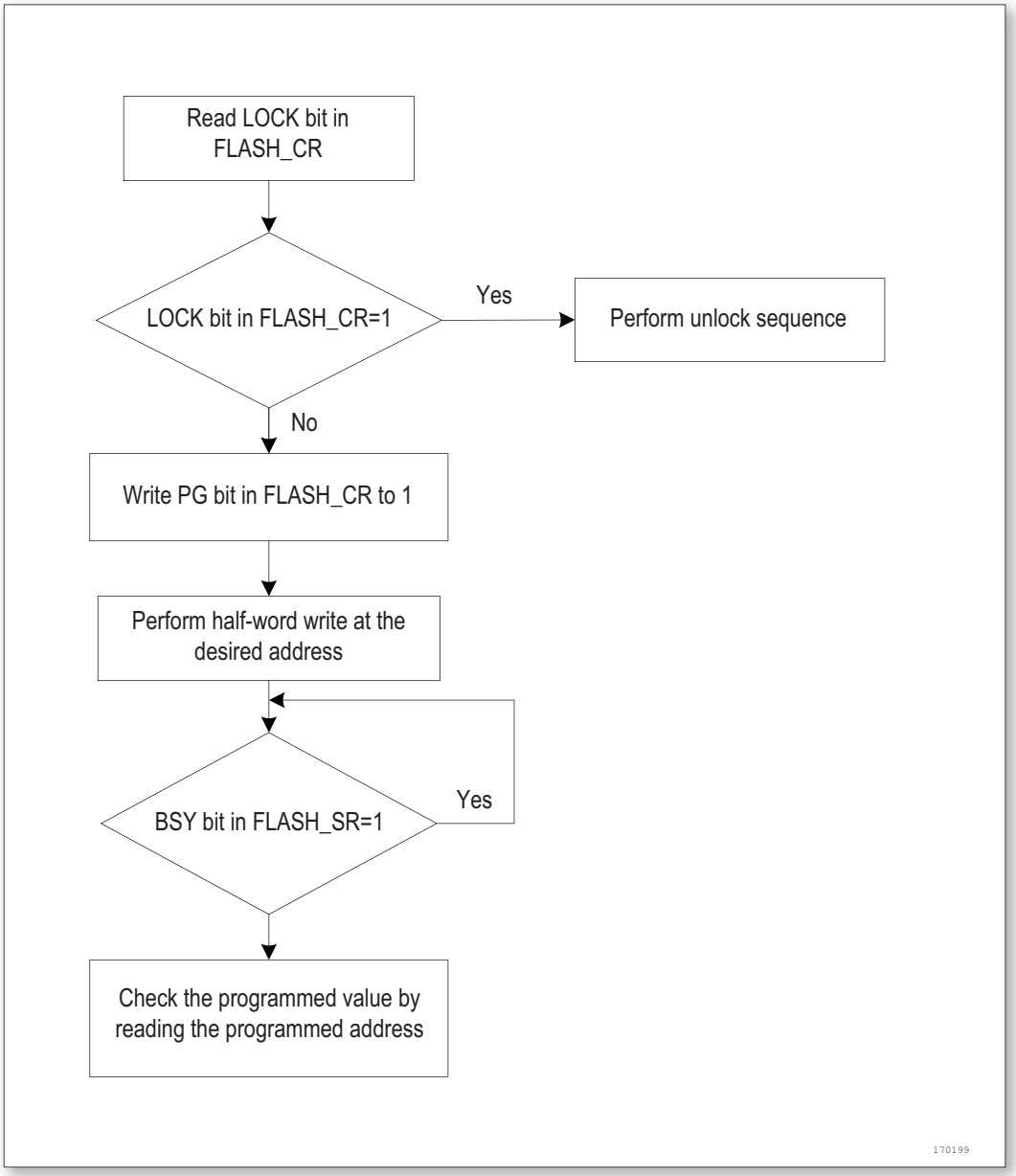


图 2. 编程流程

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 位上提示编程错误警告。

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警告。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，以确认上一操作已经结束
- 置 FLASH_CR 寄存器中的 PG 位
- 以半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
- 读数据以校验

注：当 FLASH_SR 中得 BSY 位为 1 的时候，这些寄存器不能写。

Flash 存储器擦除

Flash 存储器可以按页为单位擦除，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，以确认上一操作已经结束
- 置 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 FLASH_CR 寄存器中的 STRT 位为 1
- 等待 FLASH_SR 中的 BSY 归零
- 读取已擦除页以校验

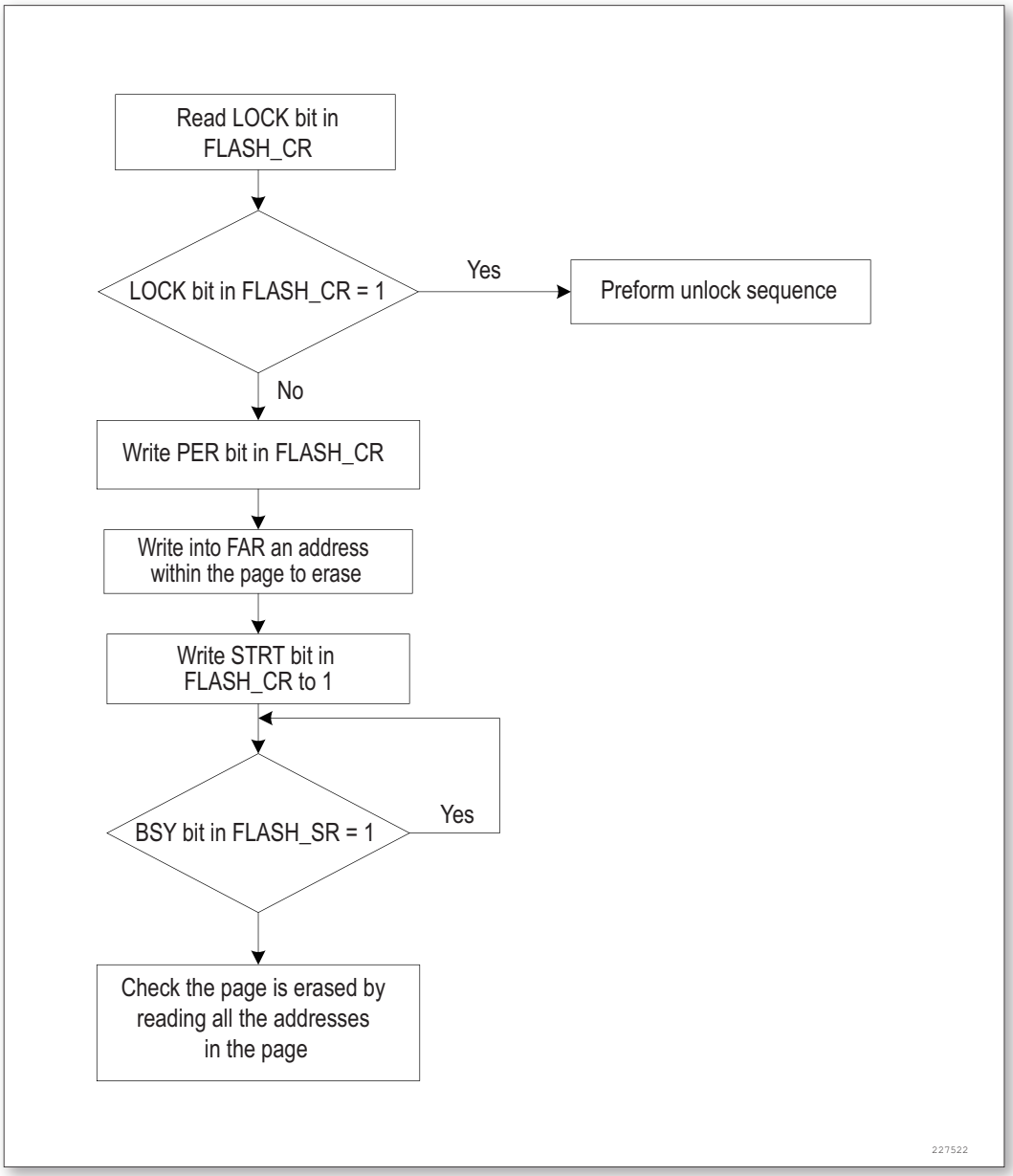


图 3. Flash 寄存器页擦除流程

整片擦除

可以用整片擦除命令一次擦除整个 Flash 用户区，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，以确认上一操作已经结束
- 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
- 读取全部页并校验

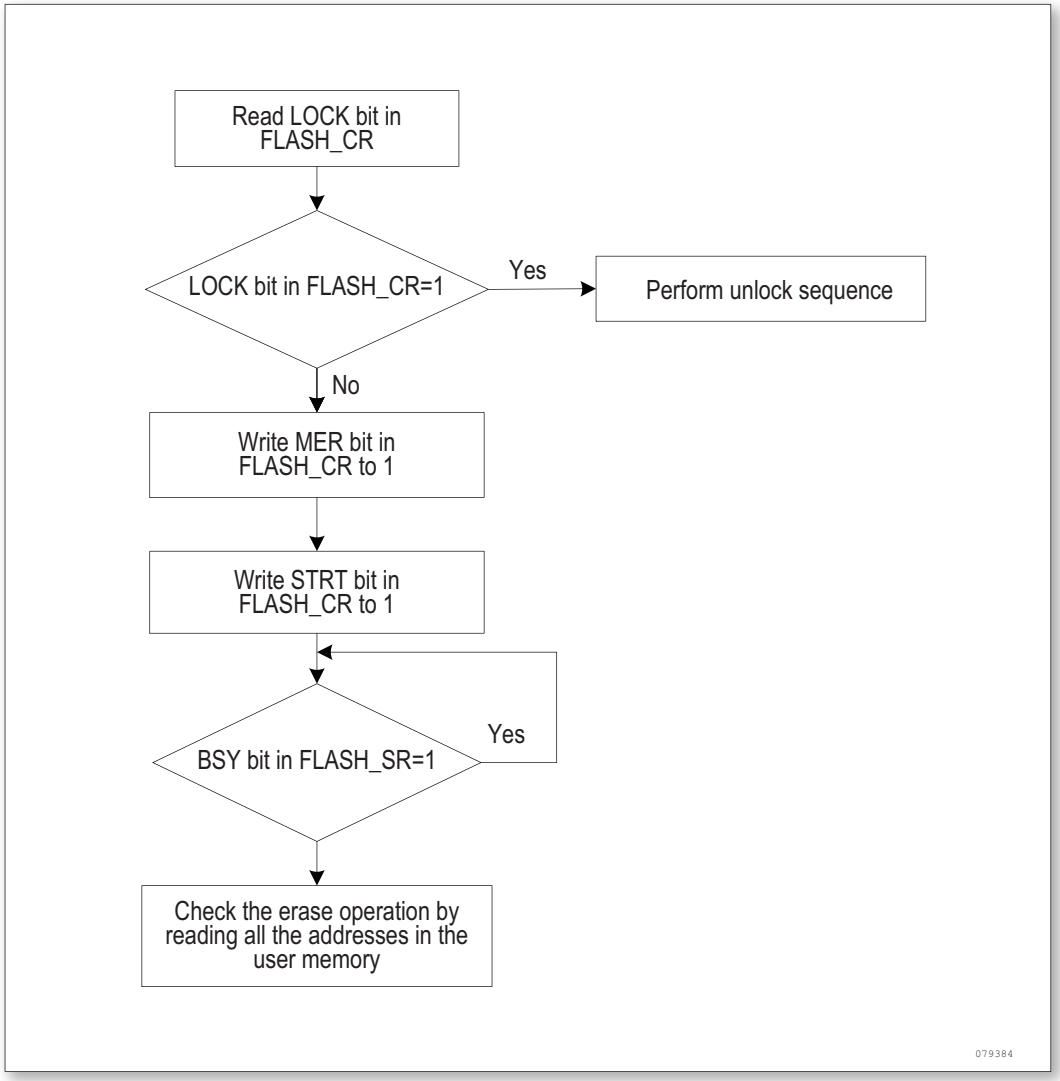


图 4. Flash 寄存器整片擦除流程

选项字节编程

选项字节的编程与常规用户地址不同，包括 2 个写保护，1 个硬件配置。解除 Flash 访问限制后，还需要对 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，再按半字单位写目标地址。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。

选项字节为 16 位数据，有效数据为低 8 位，而高 8 位为低 8 位的反码。在编程过程中，硬件会自动将高 8 位设置为低 8 位的反码，保证选项字节的写入值总是对的。步骤如下：

- 检查 FLASH_SR 寄存器中的 BSY 位，以确保上一操作结束
- 解锁 FLASH_CR 寄存器中的 OPTWRE 位
- 置 FLASH_CR 寄存器中 OPTPG 位为 1
- 写数据 (半字) 到目标地址
- 等待 BSY 位归零
- 读取并校验当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。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。

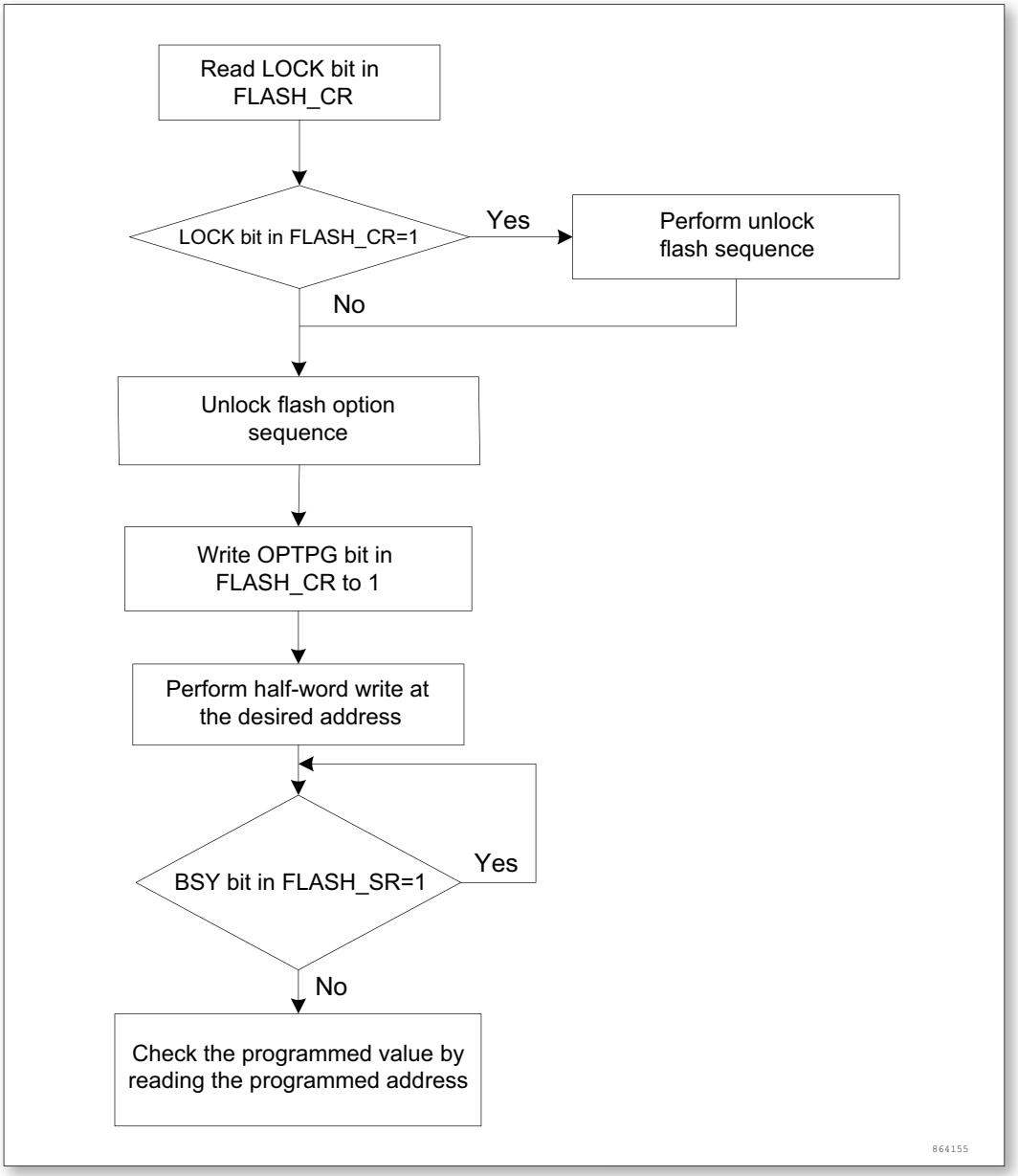


图 5. 选项字节编程流程

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，以确保上一操作结束
- 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

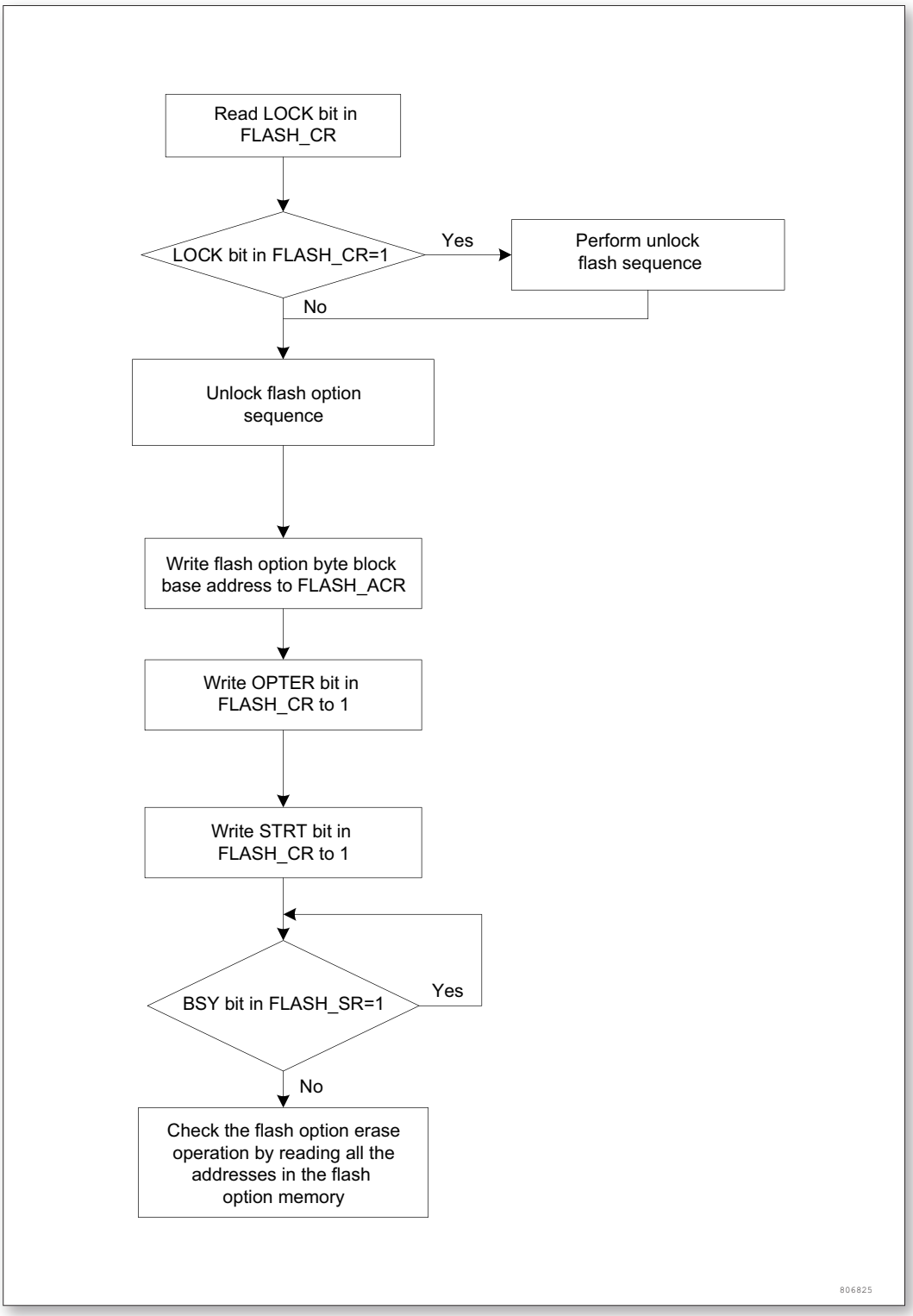


图 6. 选项字节擦除流程

2.3 存储保护

可以防范用户区 Flash 区的代码被不可信的代码读出，也可以防范在程序跑飞的时候对 Flash 的意外擦除，写保护的最小单位是一个扇区 (4 页)。

2.3.1 主空间写保护

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，随后的系统复位将加载新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。

解除保护

这种情况常见于用户自己的实现在程序中编程的启动程序:

- 使用闪存控制寄存器 (FLASH_CR) 的 OPTER 位擦除整个选项字节区域；
- 进行系统复位，重装载选项字节 (包含新的 WRP 字节)；写保护被解除。

使用这种方法，将解除除页 0 ~ 页 3 之外的整个主闪存模块的写保护，页 0~ 页 3 仍处于写保护。

2.3.2 选项字节的写保护

默认状态下，选项字节块始终是可以读且被写保护。要想对选项字节块进行写操作 (编程/擦除) 首先要在 OPTKEYR 中写入正确的键序列 (与上锁时一样)，随后允许对选项字节块的写操作，FLASH_CR 寄存器的 OPTWRE 位标示允许写，清除这位将禁止写操作。

2.4 Flash 中断

表 4. Flash 中断请求

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

2.5 选项字节说明

选项字节由用户根据应用的需要配置；例如：可以选择使用硬件模式的看门狗或软件的看门狗。

在选项字节中每个 32 位的字被划分为下述格式：

表 5. 选项字节格式

位 31 ~ 24	位 23 ~ 16	位 15 ~ 8	位 7 ~ 0
选项字节 1 的反码	选项字节 1	选项字节 0 的反码	选项字节 0

注：反码由硬件自动实现，软件写无效

选项字节块中选项字节的组织结构如下表所示。

选项字节可以从下表列出的存储器地址读出，或从选项字节寄存器 (FLASH_OBR) 读出。

注：新写入的选项字节 (用户的或读/写保护的)，在系统复位后才生效。

表 6. 选项字节结构

地址	[31：24]	[23：16]	[15：8]	[7：0]
0x1FFF F800	nUSER	USER		
0x1FFF F804	nData1	Data1	nData0	Data0
0x1FFF F808	nWRP1	WRP1	nWRP0	WRP0
0x1FFF F80C	nWRP3	WRP3	nWRP2	WRP2

表 7. 选项字节说明

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 (保存在 FLASH_OBR[9：2] 中)。这个字节用于配置下列功能： 选择看门狗事件：硬件或软件 进入停机 (STOP) 模式时的复位事件 进入待机模式时的复位事件 注：只使用位 [18：16]，不使用位 [23：19]。 位 18：nRST_STDBY 0：当进入待机模式时产生复位 1：进入待机模式时不产生复位 位 17：nRST_STOP 0：当进入停机 (STOP) 模式时产生复位 1：进入停机 (STOP) 模式时不产生复位 位 16：WDG_SW 0：硬件看门狗 1：软件看门狗
0x1FFF F804	Datax：2 个字节的用户数据 这个地址可以使用选项字节的编程方式编程。 位 [31：24]：nData1 位 [23：16]：Data1(存储在 FLASH_OBR[25：18]) 位 [15：8]：nData0 位 [7：0]：Data0(存储在 FLASH_OBR[17：10])

存储器地址	选项字节
0x1FFF F808	WRPx：闪存写保护选项字节 位 [31：24]：nWRP1 位 [23：16]：WRP1(存储在 FLASH_WRPR[15：8]) 位 [15：8]：nWRP0 位 [7：0]：WRP0(存储在 FLASH_WRPR[7：0])
0x1FFF F80C	WRPx：闪存写保护选项字节 位 [31：24]：nWRP3 位 [23：16]：WRP3(存储在 FLASH_WRPR[31：24]) 位 [15：8]：nWRP2 位 [7：0]：WRP2(存储在 FLASH_WRPR[23：16]) 选项字节 WRPx 中的每一个比特位用于保护主存储器中 4 个存储页： 0：实施写保护 1：不实施写保护 四个用户选项字节用于保护总共 128K 字节的主存器。 WRP0：第 0 ~ 31 页的写保护 WRP1：第 32 ~ 63 页的写保护 WRP2：第 64 ~ 95 页的写保护 WRP3：第 96 ~ 127 页的写保护

每次系统复位后，选项字节装载器 (OBL) 读出信息块的数据，并保存在选项字节寄存器 (FLASH_OBR) 中；每个选择位都在信息块中有它的反码位，在装载选择位时反码位用于验证选择位是否正确，如果有任何的差别，将产生一个选项字节错误标志 (OPTERR)。当发生选项字节错误时，对应的选项字节被强置为 0xFF。当选项字节和它的反码均为 0xFF 时 (擦除后的状态)，则关闭上述验证功能。

所有的选择位 (不包括它们的反码位) 用于配置该微控制器，CPU 可以读选项字节寄存器。

2.6 Flash 寄存器描述

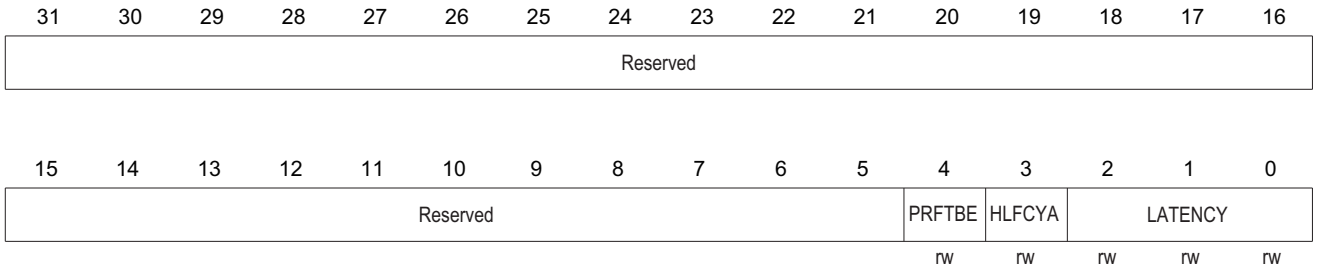
表 8. FLASH 寄存器概览

Offset	Acronym	Register Name	Reset	Section
0x00	FLASH_ACR	闪存访问控制寄存器	0x00000038	小节 2.6.1
0x04	FLASH_KEYR	FPEC 键寄存器	0xXXXXXXXX	小节 2.6.2
0x08	FLASH_OPTKEYR	闪存 OPTKEY 寄存器	0xXXXXXXXX	小节 2.6.3
0x0C	FLASH_SR	闪存状态寄存器	0x00000000	小节 2.6.4
0x10	FLASH_CR	闪存控制寄存器	0x00000080	小节 2.6.5
0x14	FLASH_AR	闪存地址寄存器	0x00000000	小节 2.6.6
0x1C	FLASH_OBR	选项字节寄存器	0x03FFFC1C	小节 2.6.7
0x20	FLASH_WRPR	写保护寄存器	0xFFFFFFFF	小节 2.6.8

2.6.1 闪存访问控制寄存器 (FLASH_ACR)

地址偏移：0x00

复位值：0x0000 0038

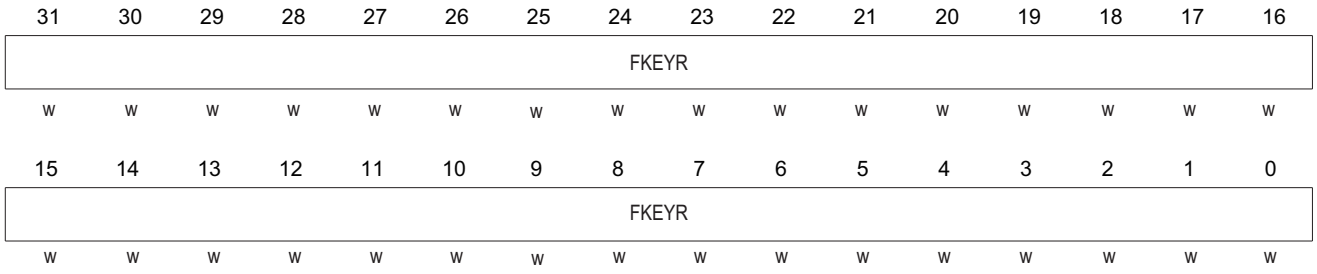


Bit	Field	Type	Reset	Description
31：5	Reserved		0x0000 0001	保留，Bit 5 始终读为 1。
4	PRFTBE	rw	0x01	预取缓冲区使能 (Prefetch buffer enable) 0：关闭预取缓冲区 1：启用预取缓冲区
3	HLFCYA	rw	0x01	闪存半周期访问使能 (Flash half cycle access enable) 0：禁止半周期访问 1：启用半周期访问
2：0	LATENCY	rw	0x00	时延 (Latency) 这些位表示 SYSCLK(系统时钟) 周期与闪存访问时间的比例。 000：零等待状态，当 0 < SYSCLK ≤ 24MHz 001：一个等待状态，当 24MHz < SYSCLK ≤ 48MHz 010：两个等待状态，当 48MHz < SYSCLK ≤ 72MHz 011：三个等待状态，当 72MHz < SYSCLK ≤ 96MHz

2.6.2 闪存访问控制寄存器 (FLASH_KEYR)

地址偏移：0x04

复位值：0xXXXX XXXX



Bit	Field	Type	Reset	Description
31：0	FKEYR	w	0xXXXX XXXX	FPEC 键 (Flash key) 这些位用于输入 FPEC 的解锁键。

注：所有这些位是只写的，读出时返回 0。

2.6.3 闪存 OPTKEY 寄存器 (FLASH_OPTKEYR)

地址偏移：0x08

复位值：0xXXXX XXXX

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

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

Bit	Field	Type	Reset	Description
31 : 0	OPTKEYR	w	0xXXXX XXXX	选择字节键 (Option byte key) 这些位用于输入选项字节的键以解除 OPTWRE。

注：所有这些位是只写的，读出时返回 0。

2.6.4 闪存状态寄存器 (FLASH_SR)

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
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved										EOP	WRPRTERR	Res.	PGERR	Res.	BSY
										rc_w1	rc_w1		rc_w1		r

Bit	Field	Type	Reset	Description
31 : 6	Reserved			保留，始终读为 0。
5	EOP	rc_w1	0x00	操作结束 (End of operation) 当闪存操作 (编程/擦除) 完成时，硬件设置这位为‘1’，写入‘1’可以清除这位状态。 注：每次成功的编程或擦除都会设置 EOP 状态。
4	WRPRTERR	rc_w1	0x00	写保护错误 (Write protection error) 试图对写保护的闪存地址编程时，硬件设置这位为‘1’，写入‘1’可以清除这位状态。
3	Reserved			保留，始终读为 0。

Bit	Field	Type	Reset	Description
2	PGERR	rc_w1	0x00	编程错误 (Programming error) 试图对内容不是‘0xFFFF’的地址编程时，硬件设置这位为‘1’，写入‘1’可以清除这位状态。 注：进行编程操作之前，必须先清除 FLASH_CR 寄存器的 STRT 位。
1	Reserved			保留，始终读为 0。
0	BSY	r	0x00	忙 (Busy) 该位指示闪存操作正在进行。在闪存操作开始时，该位被设置为‘1’；在操作结束或发生错误时该位被清除为‘0’。

2.6.5 闪存控制寄存器 (FLASH_CR)

地址偏移：0x10

复位值：0x0000 0080

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			EOPIE	Res.	ERRIE	OPTWRE	Res.	LOCK	STRT	OPTER	OPTPG	Res.	MER	PER	PG
			rw		rw	rc_w0		rw	rw	rw	rw		rw	rw	rw

Bit	Field	Type	Reset	Description
31：13	Reserved			保留，始终读为 0。
12	EOPIE	rw	0x00	允许操作完成中断 (End of operation interrupt enable) 该位允许在 FLASH_SR 寄存器中的 EOP 位变为‘1’时产生中断。 0：禁止产生中断 1：允许产生中断
11	Reserved			保留，始终读为 0。
10	ERRIE	rw	0x00	允许错误状态中断 (Error interrupt enable) 该位允许在发生 FPEC 错误时产生中断 (当 FLASH_SR 寄存器中的 PGERR/WRPRTERR 置为‘1’时)。 0：禁止产生中断 1：允许产生中断
9	OPTWRE	rc_w0	0x00	允许写选项字节 (Option byte write enable) 当该位为‘1’时，允许对选项字节进行编程操作。当在 FLASH_OPTKEYR 寄存器写入正确的键序列后，该位被置为‘1’。 软件写 0 可清除此位。
8	Reserved			保留，始终读为 0。

Bit	Field	Type	Reset	Description
7	LOCK	rw	0x01	锁 (Lock) 只能写‘1’。当该位为‘1’时表示 FPEC 和 FLASH_CR 被锁住。在检测到正确的解锁序列后，硬件自动清除此位为‘0’。 在一次不成功的解锁操作后，下次系统复位前，该位不能再被改变。
6	STRT	rw	0x00	开始 (Start) 当该位为‘1’时将触发一次擦除操作。该位只可由软件置为‘1’并在 BSY 变为‘1’时自动清‘0’。
5	OPTER	rw	0x00	擦除选项字节 (Option byte erase) 擦除选项字节。
4	OPTPG	rw	0x00	烧写选项字节 (Option byte programming) 对选项字节编程。
3	Reserved			保留，始终读为 0。
2	MER	rw	0x00	全擦除 (Mass erase) 选择擦除所有用户页。
1	PER	rw	0x00	页擦除 (Page erase) 选择擦除页。
0	PG	rw	0x00	编程 (Programming) 选择编程操作。

2.6.6 闪存地址寄存器 (FLASH_AR)

地址偏移：0x014

复位值：0x0000 0000

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

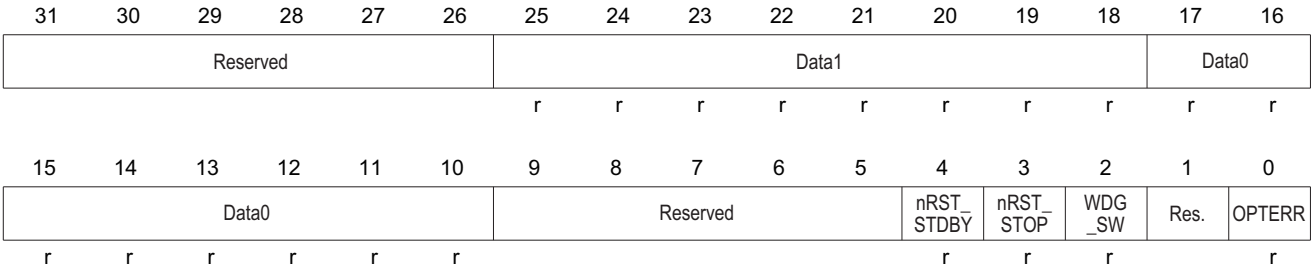
Bit	Field	Type	Reset	Description
31 : 0	FAR	w	0x0000 0000	用户选项字节 (Flash Address) 当进行编程时选择要编程的地址，当进行页擦除时选择要擦除的页。 注意：当 FLASH_SR 中的 BSY 位为‘1’时，不能写这个寄存器。

由硬件修改为当前/最后使用的地址。页擦除操作中，必须修改这个寄存器以指定要擦除的页。

2.6.7 选项字节寄存器 (FLASH_OBR)

地址偏移：0x1C

复位值：0x03FF FC1C



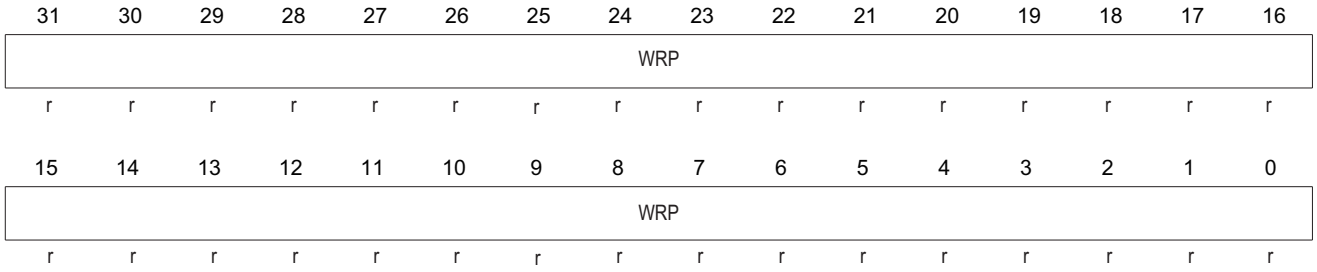
Bit	Field	Type	Reset	Description
31：26	Reserved			保留，始终读为 0。
25：18	Data1	r	0xFF	Data1
17：10	Data0	r	0xFF	Data0
9：5	Reserved			保留，始终读为 0。
4	nRST_STDBY	r	0x01	进入待机模式时的复位事件 0：当进入待机模式时产生复位 1：进入待机模式时不产生复位
3	nRST_STOP	r	0x01	进入停机模式时的复位事件 0：当进入停机 (STOP) 模式时产生复位 1：进入停机 (STOP) 模式时不产生复位
2	WDG_SW	r	0x01	选择看门狗事件 0：硬件看门狗 1：软件看门狗
1	Reserved			保留，始终读为 0。
0	OPTERR	r	0x00	选项字节错误 (Option byte error) 当该位为‘1’时表示选项字节和它的反码不匹配。 注意：该位为只读。

这个寄存器的复位数值与写入选项字节中的数值相关，OPTERR 位的复位值与加载选项字节时对选项字节和它的反码进行比较的结果相关。

2.6.8 写保护寄存器 (FLASH_WRPR)

地址偏移：0x20

复位值：0xFFFF FFFF



Bit	Field	Type	Reset	Description
31 : 0	WRP	r	0xFFFF FFFF	写保护 (Write protect) 该寄存器包含由 OBL 加载的写保护选项字节。 0：写保护生效 1：写保护失效 注意：这些位为只读。

3 循环冗余校验计算单元 (CRC)

循环冗余校验计算单元 (CRC)

3.1 CRC 简介

循环冗余校验 (CRC) 计算单元是根据固定的生成多项式得到任一 32 位全字的 CRC 计算结果。在其他的应用中，CRC 技术主要应用于核实数据传输或者数据存储的正确性和完整性。标准 EN/IEC60335-1 即提供了一种核实闪存存储器完整性的方法。CRC 计算单元可以在程序运行时计算出软件的标识，之后与在连接时生成的参考标识比较，然后存放在指定的存储器空间。

3.2 CRC 主要特征

- 使用 CRC-32(以太网) 多项式：0x4C11DB7
  $X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + X^8 + X^7 + X^5 + X^4 + X^2 + X + 1$
- 一个 32 位数据寄存器用于输入/输出。
- CRC 计算时间：4 个 AHB 时钟周期 (HCLK)
- 通用 8 位寄存器 (可用于存放临时数据)

下图为 CRC 计算单元框图

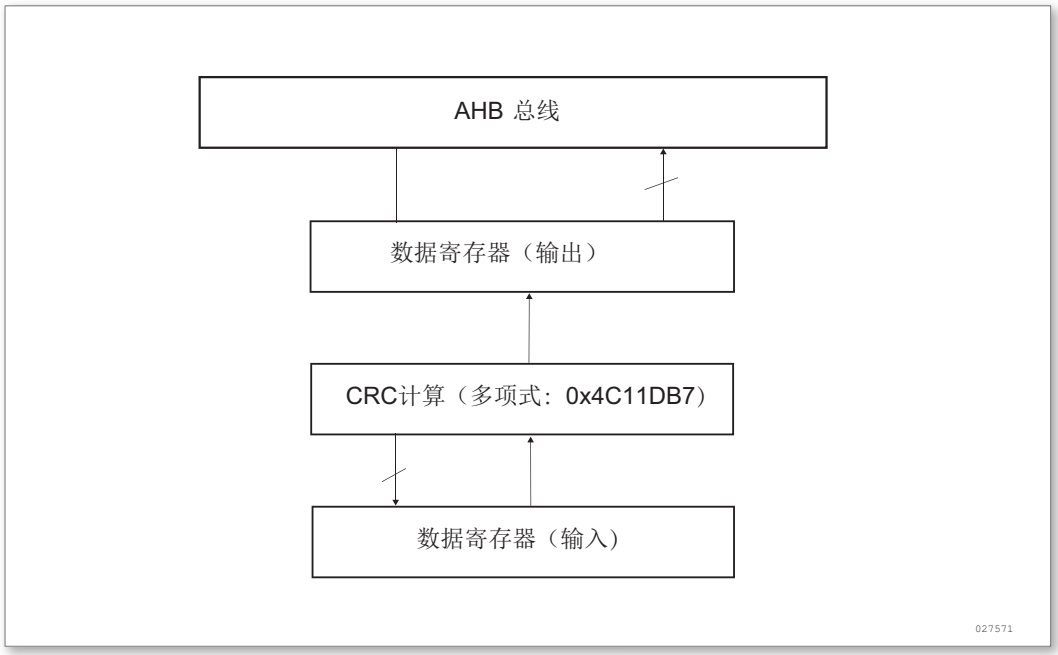


图 7. CRC 计算单元框图

3.3 CRC 功能介绍

CRC 计算单元含有 1 个 32 位数据寄存器：

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

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

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

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

3.4 CRC 寄存器

CRC 计算单元包括了 2 个数据寄存器和一个控制寄存器。

表 9. CRC 寄存器概览

Offset	Acronym	Register Name	Reset	Section
0x00	CRC_DR	CRC 数据寄存器	0xFFFFFFFF	小节 3.4.1
0x04	CRC_IDR	CRC 独立数据寄存器	0x00000000	小节 3.4.2
0x08	CRC_CTRL	CRC 控制寄存	0x00000000	小节 3.4.3

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

Bit	Field	Type	Reset	Description
31：0	DR	rw	0xFFFF FFFF	DR：数据寄存器位 (Data register bits) 写入 CRC 计算器的新数据时，作为输入寄存器 读取时返回 CRC 计算结果

3.4.2 CRC 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
Reserved															

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

Bit	Field	Type	Reset	Description
31 : 8	Reserved			始终读为 0。
7 : 0	IDR	rw	0x00	IDR：通用 8 位数据寄存器位 (General-purpose 8-bit data register bits) 可用于临时存放 1 字节的数据。 寄存器 CRC_CTRL 的 RESET 位产生的 CRC 复位对本寄存器没有影响。

注：此寄存器不参与 CRC 计算，可以存放任何数据。

3.4.3 CRC 控制寄存器 (CRC_CTRL)

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
Reserved															

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

Bit	Field	Type	Reset	Description
31 : 1	Reserved			始终读为 0。
0	RESET	w	0x00	RESET：复位 CRC 计算单元 (CRC reset) 设置数据寄存器为 0xFFFF FFFF。 只能对该位写‘1’，它由硬件自动清‘0’。

4 电源控制 (PWR)

电源控制 (PWR)

4.1 电源

芯片的工作电压 (V_{DD}) 为 2.0V ~ 5.5V。通过内置的电压调节器提供所需的 1.5V 电源。

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

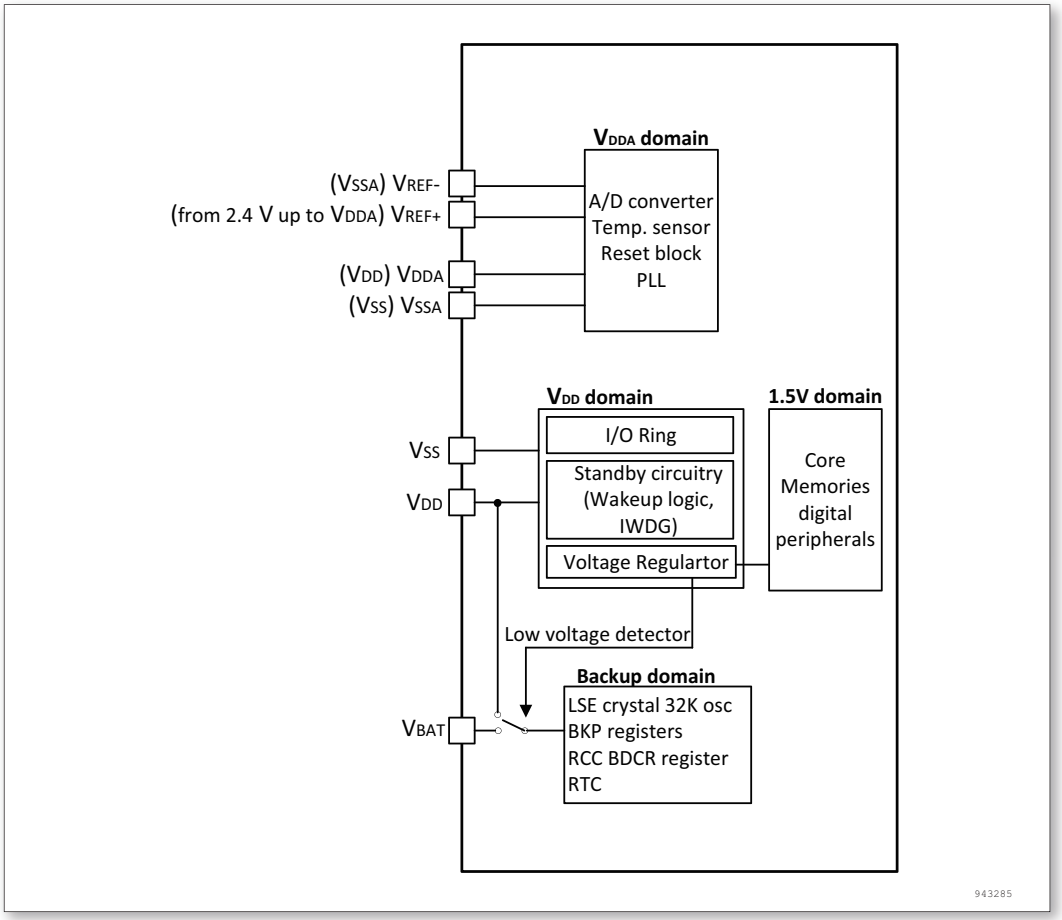


图 8. 电源框图

注：V_{DDA} 和 V_{SSA} 必须分别连到 V_{DD} 和 V_{SS}。

4.1.1 独立的 A/D 转换器供电和参考电压

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

- ADC 的电源引脚为 V_{DDA}
- 独立的电源地 V_{SSA}

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

4.1.2 电池备份区域

使用电池或其他电源连接到 V_{BAT} 脚上，当 V_{DD} 断电时，可以保存备份寄存器的内容和维持 RTC 的功能。

V_{BAT} 脚为 RTC、LSE 振荡器和 PC13 至 PC15 端口供电，可以保证当主电源被切断时 RTC 能继续工作。切换到 V_{BAT} 供电的开关，由复位模块中的掉电复位功能控制。

警告

在 V_{DD} 上升阶段 ($t_{RSTTEMPO}$) 或者探测到 PDR (掉电复位) 之后，V_{BAT} 和 V_{DD} 之间的电源开关仍会保持连接在 V_{BAT}。

如果在应用中没有外部电池，建议 V_{BAT} 在外部连接到 V_{DD} 并连接一个 100nF 的陶瓷滤波电容。

当备份区域由 V_{DD} 供电时，下述功能可用：

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

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

当后备区域由 V_{BAT} 供电时 (V_{DD} 掉电后模拟开关连到 V_{BAT})，可以使用下述功能：

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

4.1.3 电压调节器

复位后调节器总是使能的。根据应用方式它以 3 种不同的模式工作。

- 运转模式：调节器以正常功耗模式提供 1.5V 电源 (内核，内存和外设)。
- 停机模式：调节器以低功耗模式提供 1.5V 电源，以保存寄存器和 SRAM 的内容。
- 待机模式：调节器停止供电。除了备用电路和备份域外，寄存器和 SRAM 的内容全部丢失。

4.2 电源管理器

4.2.1 上电复位 (POR) 和掉电复位 (PDR)

芯片有一个完整的上电复位 (POR) 和掉电复位 (PDR) 电路，当供电电压达到 2.0V 时系统即能正常工作。

当 V_{DD}/V_{DDA} 低于指定的限位电压 V_{POR}/V_{PDR} 时，系统保持为复位状态，而无需外部复位电路。关于上电复位和掉电复位的细节请参考数据手册的电气特性部分。

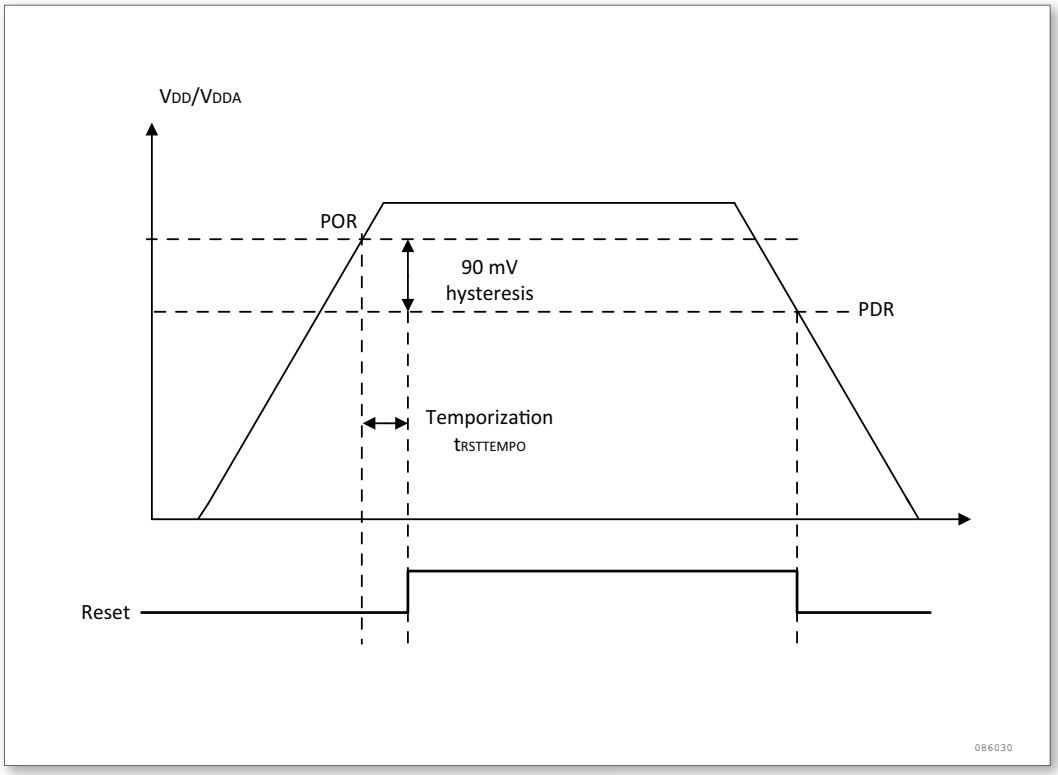


图 9. 上电复位和掉电复位的波形图

4.2.2 可编程电压监测器 (PVD)

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

通过设置 PVDE 位来使能 PVD。

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

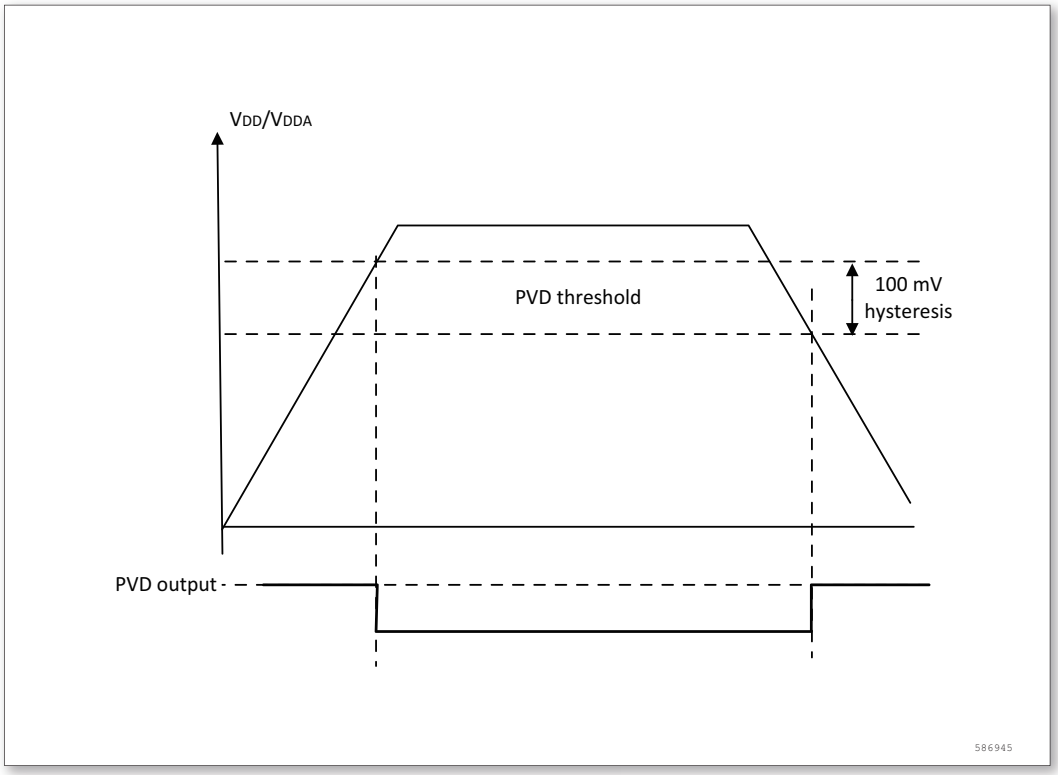


图 10. PVD 的阈值

4.3 低功耗模式

在系统或电源复位以后，微控制器处于运行状态。当 CPU 不需继续运行时，可以利用多种低功耗模式来节省功耗，例如等待某个外部事件时。用户需要根据最低电源消耗、最快速启动时间和可用的唤醒源等条件，选定一个最佳的低功耗模式。

芯片有三种低功耗模式：

- 睡眠模式 (CPU 停止，所有外设包括 CPU 的外设，如 NVIC、系统时钟 (SysTick) 等仍在运行)
- 停机模式 (所有的时钟都已停止，寄存器和 SRAM 的内容依然保存)
- 待机模式 (1.5V 电源关闭，除了备用电路和备份域外，寄存器和 SRAM 的内容全部丢失。)

此外，在运行模式下，可以通过以下方式中的一种降低功耗：

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

表 10. 低功耗模式一览

模式	进入	唤醒	对 1.5V 区域时钟的影响	对 V_{DD} 区域时钟的影响	电压调节器
睡眠 (SLEEP NOW 或 SLEEP ON EXIT)	WFI (Wait for Interrupt)	任一中断	CPU 时钟关，对其他时钟和 ADC 时钟无影响	无	开
	WFE (Wait for Event)	唤醒事件			
停机	PDDS 位 SLEEPDEEP 位 WFI 或 WFE	任一外部中断 (在外部中断寄存器中设置)	所有使用 1.5V 的区域的时钟都已关闭	PLL、HSI 和 HSE 的振荡器关闭	开
待机	PDDS SLEEPDEEP 位 WFI 或 WFE	WKUP 引脚的上升沿、RTC 闹钟事件、NRST 引脚上的外部复位、IWDG 复位			关

4.3.1 降低系统时钟

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

详见：时钟配置寄存器 (RCC_CFGR)

4.3.2 外部时钟的控制

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

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

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

4.3.3 睡眠模式

进入睡眠模式

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

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

在睡眠模式下，所有的 I/O 引脚都保持它们在运行模式时的状态。

关于如何进入睡眠模式，更多的细节参考表 11和表 12。

表 11. SLEEP NOW 模式

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

表 12. SLEEP ON EXIT 模式

SLEEP ON EXIT 模式	说明
进入	在以下条件下执行 WFI(Wait for Interrupt) 或 WFE(Wait for Event) 指令： - SLEEPDEEP = 0 - SLEEPONEXIT = 1 参考 CPU 系统控制寄存器。
退出	如果执行 WFI 进入睡眠模式：中断或清除 CPU 控制寄存器位 1 如果执行 WFE 进入睡眠模式：唤醒事件：参考唤醒事件管理
唤醒延时	无

4.3.4 停机模式

停机模式是在 CPU 的深睡眠模式基础上结合了外设的时钟控制机制，在停机模式下电压调节器可运行在正常模式。此时在 1.5V 供电区域的的所有时钟都被停止，PLL、HSI 和 HSE 振荡器的功能被禁止，SRAM 和寄存器内容被保留下来。

在停机模式下，所有的 I/O 引脚都保持它们在运行模式时的状态。

进入停机模式

关于如何进入停机模式，详见表 13。

可以通过对独立的控制位进行编程，可选择以下功能：

- 独立看门狗 (IWDG)：可通过写入看门狗的键寄存器或硬件选择来启动 IWDG。
- 实时时钟 (RTC)：通过备份域控制寄存器 (RCC_BDCR) 的 RTCEN 位来设置。
- 内部振荡器 (LSI 振荡器)：通过控制/状态寄存器 (RCC_CSR) 的 LSION 位来设置。
- 外部 32.768KHz 振荡器 (LSE)：通过备份域控制寄存器 (RCC_BDCR) 的 LSEON 位设置。

在停机模式下，如果在进入该模式前 ADC 和 DAC 没有被关闭，那么这些外设仍然消耗电流。通过设置寄存器 ADC_CR2 的 ADON 位和寄存器 DAC_CR 的 ENx 位为 0 可关闭这 2 个外设。其他没有使用的 GPIO 需要设置模拟输入，否则有电流消耗。

退出停机模式

关于如何退出停机模式，详见表表 13。

当一个中断或唤醒事件导致退出停机模式时，HSI 振荡器被选为系统时钟。时钟频率为 HSI 的 6 分频。

当电压调节器处于正常功耗模式下，系统从停机模式退出时，将会有一段额外的启动延时。

表 13. 停机模式

停机模式	说明
进入	在以下条件下执行 WFI(Wait for Interrupt) 或 WFE(Wait for Event) 指令： - 设置 CPU 系统控制寄存器中的 SLEEPDEEP 位 - 清除电源控制寄存器 (PWR_CR) 中的 PDDS 位 注：为了进入停机模式，所有的外部中断的请求位 (挂起寄存器 (EXTI_PEND)) 和 RTC 的闹钟标志都必须被清除，否则停机模式的进入流程将会被跳过，程序继续运行。
退出	在以下条件下执行 WFI(Wait for Interrupt) 指令： 任一外部中断引线被设置为中断模式 (相应的外部中断向量在 NVIC 中必须使能)。 参见中断向量表 在以下条件下执行 WFE(Wait for Event) 指令： 任一外部中断引线被设置为事件模式。参见唤醒事件管理。
唤醒延时	HSI 的唤醒时间

4.3.5 待机模式

待机模式可实现系统的最低功耗。该模式是在 CPU 深睡眠模式时关闭电压调节器。整个 1.5V 供电区域被断电。PLL、HSI 和 HSE 振荡器也被断电。SRAM 和寄存器内容丢失。只有备份的寄存器和待机电路维持供电。

进入待机模式

关于如何进入待机模式，详见表 14。

可以通过设置独立的控制位，选择以下待机模式的功能：

- 独立看门狗 (IWDG)：可通过写入看门狗的键寄存器或硬件选择来启动 IWDG。
- 实时时钟 (RTC)：通过备用区域控制寄存器 (RCC_BDCR) 的 RTCEN 位来设置。
- 内部振荡器 (LSI 振荡器)：通过控制/状态寄存器 (RCC_CSR) 的 LSION 位来设置。
- 外部 32.768KHz 振荡器 (LSE)：通过备用区域控制寄存器 (RCC_BDCR) 的 LSEON 位设置。

退出待机模式

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

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

关于如何退出待机模式，详见表表 14。

表 14. 待机模式

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

待机模式下的输入/输出端口状态

在待机模式下，所有的 I/O 引脚处于高阻态，除了以下的引脚：

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

调试模式

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

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

4.3.6 低功耗模式下的自动唤醒 (AWU)

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

- 低功耗 32.768KHz 外部晶振 (LSE)
  - 该时钟源提供了一个低功耗且精确的时间基准。(在典型情形下消耗小于 1μA)
- 低功耗内部振荡器 (LSI 振荡器)
  - 使用该时钟源，节省了一个 32.768KHz 晶振的成本。但是振荡器将少许增加电源消耗。

为了用 RTC 闹钟事件将系统从停机模式下唤醒，必须进行如下操作：

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

4.4 电源控制寄存器

表 15. 电源控制寄存器概览

Offset	Acronym	Register Name	Reset	Section
0x00	PWR_CR	电源控制寄存器	0x00000000	小节 4.4.1
0x04	PWR_CSR	电源控制/状态寄存器	0x00000000	小节 4.4.2

4.4.1 电源控制寄存器 (PWR_CR)

地址偏移：0x00

复位值：0x0000 0000(从待机模式唤醒时清除)

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

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

Bit	Field	Type	Reset	Description
31：13	Reserved			始终读为 0。
12：9	PLS	rw	0x00	PVD 电平选择 (PVD level selection) 这些位用于选择电源电压监测器的电压阀值 0000：1.8V 0100：3.0V 1000：4.2V 0001：2.1V 0101：3.3V 1001：4.5V 0010：2.4V 0110：3.6V 1010：4.8V 0011：2.7V 0111：3.9V 其他：保留 注：详细说明参见数据手册中的电气特性部分。
8	DBP	rw	0x00	取消后备区域的写保护 (domain write protection) 在复位后，RTC 和后备寄存器处于被保护状态以防意外写入。设置这位允许写入这些寄存器。 1 = 允许写入 RTC 和后备寄存器 0 = 禁止写入 RTC 和后备寄存器 注：如果 RTC 的时钟是 HSE/128，该位必须保持为‘1’。
7:5	Reserved			始终读为 0
4	PVDE	rw	0x00	电源电压监测器 (PVD) 使能 (Power voltage detector enable) 1 = 开启 PVD 0 = 禁止 PVD
3	CSBF	rw	0x00	清除待机位 (Clear standby flag) 始终读出为 0 1 = 清除 SBF 待机位 (写) 0 = 无功效
2	CWUF	rw	0x00	清除唤醒位 (Clear wakeup flag) 始终读出为 0 1 = 2 个系统时钟周期后清除 WUF 唤醒位 (写) 0 = 无功效

Bit	Field	Type	Reset	Description
1	PDDS	rw	0x00	掉电深睡眠 (Power down deepsleep) 1 = CPU 进入深睡眠时进入待机模式 0 = CPU 进入深睡眠时进入停机模式
0	Reserved			始终读为 0

4.4.2 电源控制/状态寄存器 (PWR_CSR)

地址偏移：0x04

复位值：0x0000 0000(从待机模式唤醒时不被清除)

与标准的 APB 读相比，读次寄存器需要额外的 APB 周期

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

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

Bit	Field	Type	Reset	Description
31 : 9	Reserved			始终读为 0。
8	EWUP	rw	0x00	使能 WKUP 引脚 (Enable WKUP pin) 1 = WKUP 引脚用于将 CPU 从待机模式唤醒，WKUP 引脚被强置为输入下拉的配置 (WKUP 引脚上的上升沿将系统从待机模式唤醒) 0 = WKUP 引脚为通用 I/O。WKUP 引脚上的事件不能将 CPU 从待机模式唤醒 注：在系统复位时清除这一位。
7:3	Reserved			始终读为 0
2	PVDO	rw	0x00	PVD 输出 (PVD output) 当 PVD 被 PVDE 位使能后该位才有效 1 = V_{DD}/V_{DDA} 低于由 PLS 选定的 PVD 阀值 0 = V_{DD}/V_{DDA} 高于由 PLS 选定的 PVD 阀值 注：在待机模式下 PVD 被停止。因此，待机模式后或复位后，直到设置 PVDE 位之前，该位为 0。
1	SBF	rw	0x00	待机标志 (Standby flag) 该位由硬件设置，并只能由 POR/PDR (上电/掉电复位) 或设置电源控制寄存器 (PWR_CR) 的 CSBF 位清除。 1 = 系统进入待机模式 0 = 系统不在待机模式

Bit	Field	Type	Reset	Description
0	WUF	rw	0x00	唤醒标志 (Wakeup flag) 该位由硬件设置，并只能由 POR/PDR (上电/掉电复位) 或设置电源控制寄存器 (PWR_CR) 的 CWUF 位清除。 1 = 在 WKUP 引脚上发生唤醒事件或出现 RTC 闹钟事件 0 = 没有发生唤醒事件 注：当 WKUP 引脚已经是高电平时，在 (通过设置 EWUP 位) 使能 WKUP 引脚时，会检测到一个额外的事件。

5 备份寄存器 (BKP)

备份寄存器 (BKP)

5.1 BKP 简介

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

此外，BKP 控制寄存器用来管理侵入检测和 RTC 校准功能。复位后，对备份寄存器和 RTC 的访问被禁止，并且备份域被保护以防止可能存在的意外的写操作。

执行以下操作可以使能对备份寄存器和 RTC 的访问。

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

5.2 BKP 特征

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

5.3 BKP 功能描述

5.3.1 侵入检测

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

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

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

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

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

5.3.2 RTC 校准

为方便测量，RTC 时钟可以经 64 分频输出到侵入检测引脚 TAMPER 上。通过设置 RTC 校验寄存器 (BKP_RTCCR) 的 CCO 位来开启这一功能。

通过配置 CAL[6：0] 位，此时钟可以最多减慢 121ppm。

5.4 BKP 寄存器描述

可以用半字 (16 位) 或字 (32 位) 的方式操作这些外设寄存器。

表 16. BKP 寄存器概览

Offset	Acronym	Register Name	Reset	Section
0x04 + 4 × (n - 1)	BKP_DRn	备份数据寄存器 x	0x00000000	小节 5.4.1
0x2C	BKP_RTCCR	RTC 时钟校准寄存器	0x00000000	小节 5.4.2
0x30	BKP_CR	备份控制寄存器	0x00000000	小节 5.4.3
0x34	BKP_CSR	备份控制/状态寄存器	0x00000000	小节 5.4.4

5.4.1 备份数据寄存器 n(BKP_DRn)(n = 1...10)

地址偏移：0x04 + 4 × (备份数据寄存器编号 - 1)

复位值：0x0000

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

Bit	Field	Type	Reset	Description
15 : 0	BKP	rw	0x0000	BKP：备份数据 这些位可以被用来写入用户数据。 注意：BKP_DRn 寄存器不会被系统复位、电源复位、从待机模式唤醒所复位。它们可以由备份域复位来复位或 (如果侵入检测引脚 TAMPER 功能被开启时) 由侵入引脚事件复位。

5.4.2 RTC 时钟校准寄存器 (BKP_RTCCR)

地址偏移：0x2C

复位值：0x0000

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

Bit	Field	Type	Reset	Description
15 :10	Reserved			始终读为 0。

Bit	Field	Type	Reset	Description
9	ASOS	rw	0x00	ASOS：闹钟或秒输出选择 (Alarm or second output selection) 当设置了 ASOE 位，ASOS 位可用于选择在 TAMPER 引脚上输出的是 RTC 秒脉冲还是闹钟脉冲信号。 0：输出 RTC 闹钟脉冲 1：输出秒脉冲 注：该位只能被后备区的复位所清除。
8	ASOE	rw	0x00	ASOE：允许输出闹钟或秒脉冲 (Alarm or second output enable) 根据 ASOS 位的设置，该位允许 RTC 闹钟或秒脉冲输出到 TAMPER 引脚上。 输出脉冲的宽度为一个 RTC 时钟的周期。设置了 ASOE 位时不能开启 TAMPER 的功能。 注：该位只能被后备区的复位所清除。
7	CCO	rw	0x00	CCO：校准时钟输出 (Calibration clock output) 0：无影响 1：此位置 1 可以在侵入检测引脚输出经 64 分频后的 RTC 时钟。当 CCO 位置 1 时，必须关闭侵入检测功能以避免检测到无用的侵入信号。 注：当 V_{DD} 供电断开时，该位被清除。
6 : 0	CAL	rw	0x00	CAL：校准值 (Calibration value) 校准值表示在每 2^{20} 个时钟脉冲内将有多少个时钟脉冲被跳过。这可以用来对 RTC 进行校准，以 1000000/(2^{20})ppm 的比例减慢时钟。 RTC 时钟可以被减慢 0~121ppm。

5.4.3 备份控制寄存器 (BKP_CR)

地址偏移：0x30

复位值：0x0000

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

Bit	Field	Type	Reset	Description
15 : 2	Reserved			始终读为 0。
1	TPAL	rw	0x00	TPAL：侵入检测 TAMPER 引脚有效电平 (TAMPER pin active level) 0：侵入检测 TAMPER 引脚上的高电平会清除所有数据备份寄存器 (如果 TPE 位为 1) 1：侵入检测 TAMPER 引脚上的低电平会清除所有数据备份寄存器 (如果 TPE 位为 1)

Bit	Field	Type	Reset	Description
0	TPE	rw	0x00	TPE：启动侵入检测 TAMPER 引脚 (TAMPER pin enable) 0：侵入检测 TAMPER 引脚作为通用 IO 口使用 1：开启侵入检测引脚作为侵入检测使用

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

5.4.4 备份控制/状态寄存器 (BKP_CSR)

地址偏移：0x34

复位值：0x0000

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

Bit	Field	Type	Reset	Description
15 : 10	Reserved			始终读为 0。
9	TIF	r	0x90	TIF：侵入中断标志 (Tamper interrupt flag) 当检测到有侵入事件且 TPIE 位为 1 时，此位由硬件置‘1’。通过向 CTI 位写‘1’来清除此标志位 (同时也清除了中断)。如果 TPIE 位被清除，则此位也会被清除。 0：无侵入中断 1：产生侵入中断 注意：仅当系统复位或由待机模式唤醒后才复位该位。
8	TEF	r	0x00	TEF：侵入事件标志 (Tamper event flag) 当检测到侵入事件时此位由硬件置‘1’。通过向 CTE 位写‘1’可清除此标志位。 0：无侵入事件 1：检测到侵入事件 注：侵入事件会复位所有的 BKP_DRn 寄存器。只要 TEF 为 1，所有的 BKP_DRn 寄存器就一直保持复位状态。当此位被置‘1’时，若对 BKP_DRn 进行写操作，写入的值不会被保存。
7 : 3	Reserved			始终读为 0。
2	TPIE	rw	0x00	TPIE：允许侵入 TAMPER 引脚中断 (TAMPER pin interrupt enable) 0：禁止侵入检测中断 1：允许侵入检测中断 (BKP_CR 寄存器的 TPE 位也必须被置‘1’) 注 1：侵入中断无法将系统内核从低功耗模式唤醒。 注 2：仅当系统复位或由待机模式唤醒后才复位该位。

Bit	Field	Type	Reset	Description
1	CTI	w	0x00	CTI：清除侵入检测中断 (Clear tamper interrupt) 此位只能写入，读出值为 0。 0：无效 1：清除侵入检测中断和 TIF 侵入检测中断标志
0	CTE	w	0x00	CTE：清除侵入检测事件 (Clear tamper event) 此位只能写入，读出值为 0。 0：无效 1：清除 TEF 侵入检测事件标志 (并复位侵入检测器)

6 复位和时钟控制 (RCC)

复位和时钟控制 (RCC)

6.1 复位

支持三种复位形式，分别为系统复位、上电复位和备份区域复位。

6.1.1 系统复位

系统复位将复位除时钟控制寄存器 CSR 中的复位标志、电源控制寄存器 CSR 中的待机和唤醒标志以及备份区域中的寄存器以外的所有寄存器。

当以下事件中的一件发生时，产生一个系统复位：

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

可通过查看 RCC_CSR 控制状态寄存器中的复位状态标志位识别复位事件来源。

软件复位

通过将 CPU 中断应用和复位控制寄存器中的 SYSRESETREQ 位置‘1’，可实现软件复位。

6.1.2 电源复位

当以下事件中之一发生时，产生电源复位：

1. 上电/掉电复位 (POR/PDR 复位)
2. 从待机模式中返回

电源复位将复位除了备份区域外的所有寄存器。

图 11复位源将最终作用于 RESET 管脚，并在复位过程中保持低电平。复位入口矢量被固定在地址 0x0000_0004。

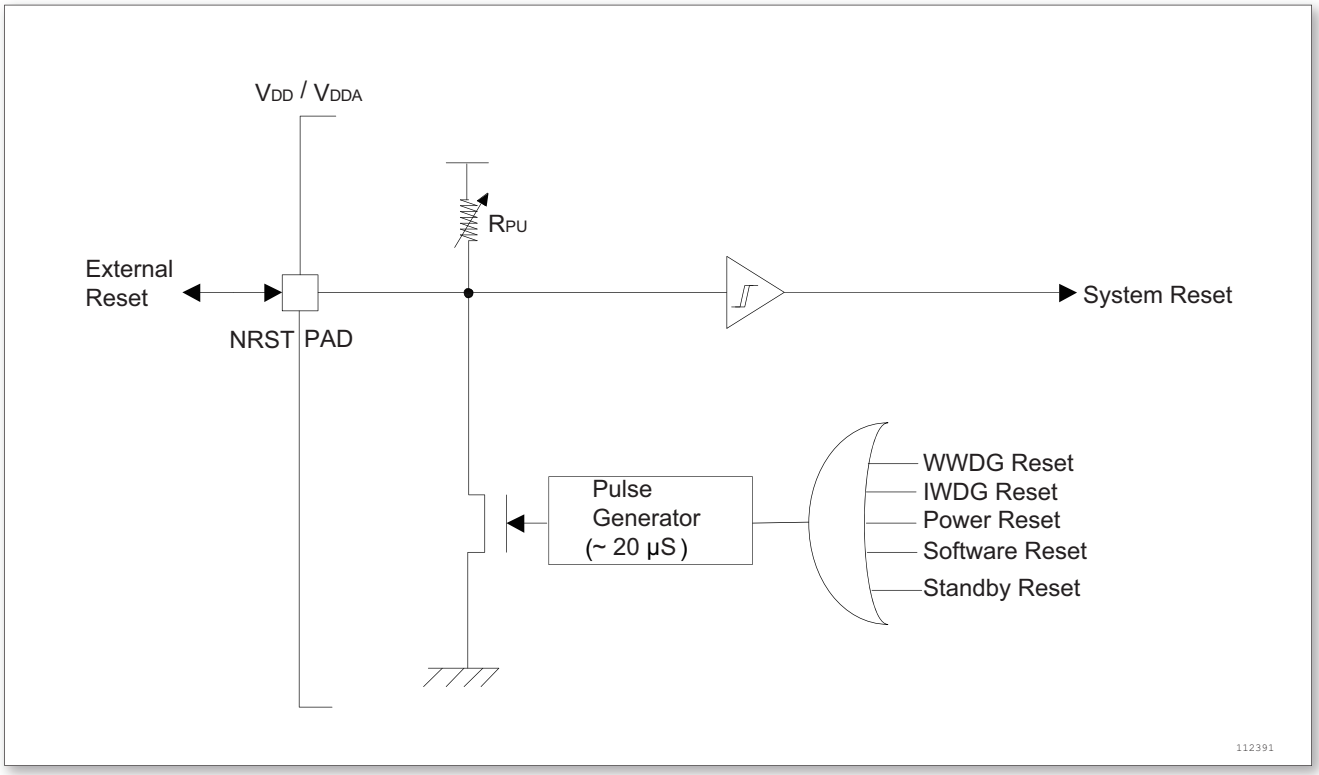


图 11. 复位电路

6.1.3 备份域复位

备份区域拥有专门的复位，它们只影响备份区域。备份区域复位可由设置备份区域控制寄存器 RCC_BDCR 中的 BDRST 位产生。

注：在 V_{BAT} 上电后且 BDRST 复位前，备份区域是未复位的状态，需要设置 BDRST 位复位备份区域。

6.2 时钟

三种不同的时钟源可被用来驱动系统时钟 (SYSCLK):

- HSI 振荡器时钟
- HSE 振荡器时钟
- PLL 时钟

这些设备有以下 2 种二级时钟源：

- 40KHz 低速内部振荡器，可以用于驱动独立看门狗和通过程序选择驱动 RTC。RTC 用于从停机/待机模式下自动唤醒系统。
- 32.768KHz 低速外部晶体也可用来通过程序选择驱动 RTC(RTCCLK)。

当不被使用时，任一个时钟源都可被独立地启动或关闭，由此优化系统功耗。

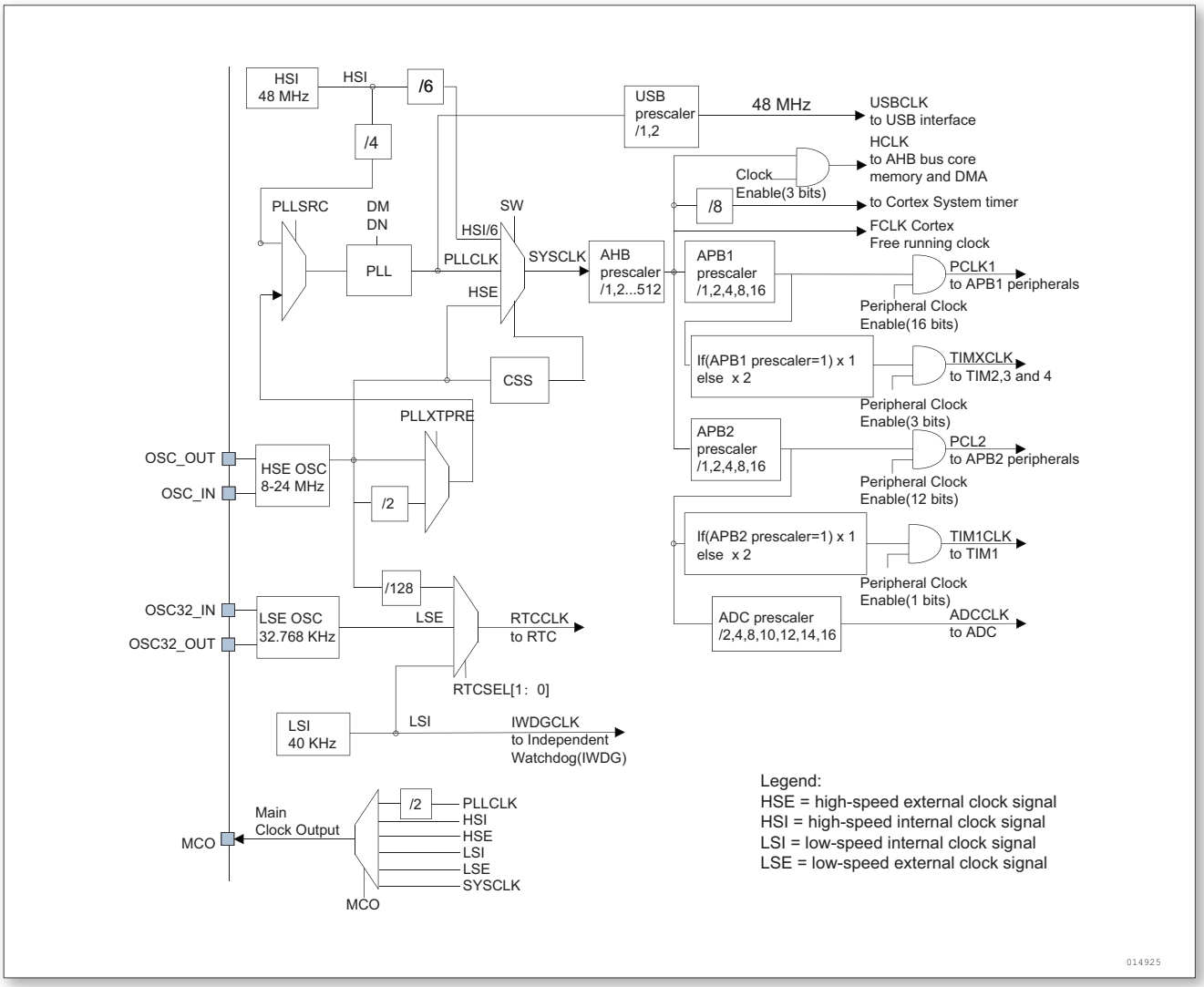


图 12. 时钟树

用户可通过多个预分频器配置 AHB、高速 APB(APB2) 和低速 APB(APB1) 域的频率。AHB 和 APB1，APB2 域的最大频率是 96MHz。

RCC 通过 AHB 时钟 8 分频后供给 CPU 系统定时器的 (SysTick) 外部时钟。通过对 SysTick 控制与状态寄存器的设置，可选择上述时钟或 AHB 时钟作为 SysTick 时钟。ADC 时钟由高速 APB2 时钟分频后获得。

定时器时钟频率分配由硬件按以下 2 种情况自动设置：

1. 如果相应的 APB 预分频系数是 1，定时器的时钟频率与所在 APB 总线频率一致。
2. 否则，定时器的时钟频率被设为与其相连的 APB 总线频率的 2 倍。

FCLK 是 CPU 的自由运行时钟。

6.2.1 HSE 时钟

高速外部时钟信号 (HSE) 由以下两种时钟源产生：

- HSE 外部晶体/陶瓷谐振器
- HSE 用户外部时钟

为了减少时钟输出的失真和缩短启动稳定时间，晶体/陶瓷谐振器和负载电容器必须尽可能

地靠近振荡器管脚。负载电容值必须根据所选择的振荡器来调整。

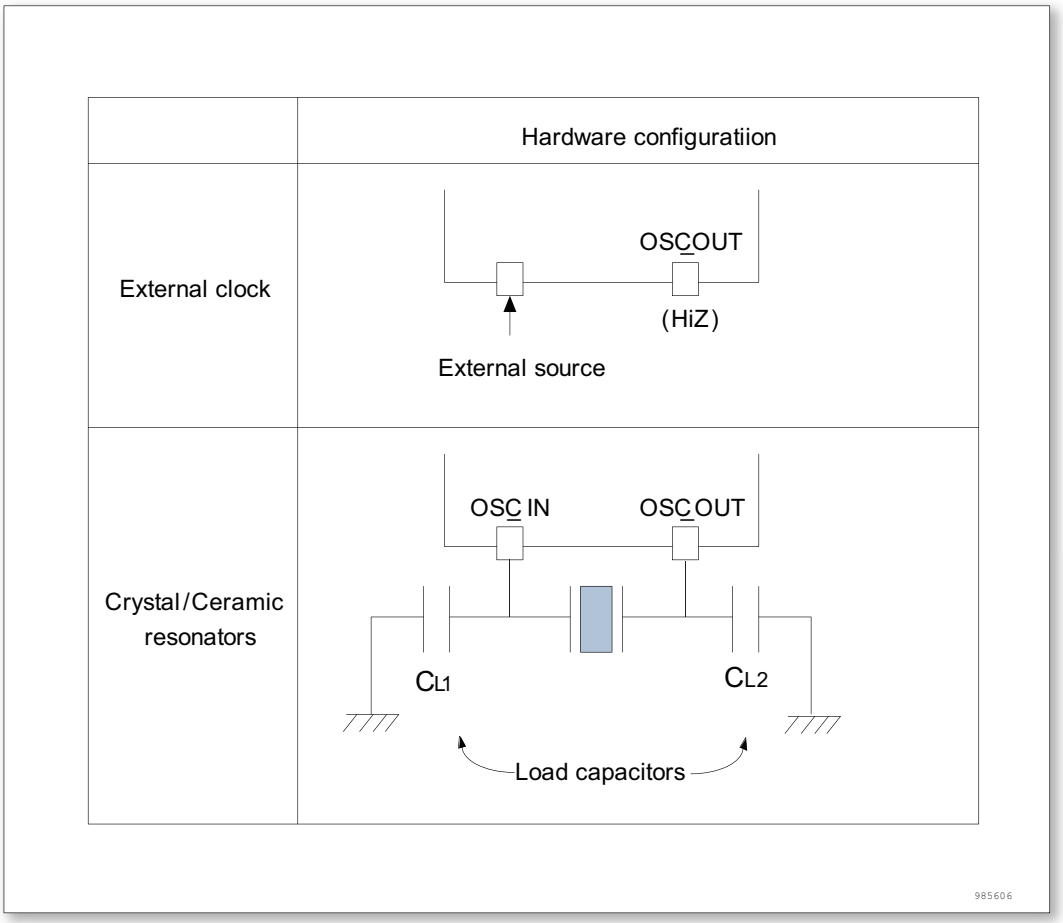


图 13. 时钟源

外部时钟源 (HSE 旁路)

在这个模式里，必须提供外部时钟。它的频率最高可达 24MHz。用户可通过设置在时钟控制寄存器中的 HSEBYP 和 HSEON 位来选择这一模式。外部时钟信号 (50% 占空比的方波、正弦波) 必须连到 OSC_IN 管脚，同时保证 OSC_OUT 管脚悬空。

外部晶体/陶瓷谐振器 (HSE 晶体)

外部振荡器可为系统提供更为精确的主时钟。相关的硬件配置可参考图 13，进一步信息可参考数据手册的电气特性部分。

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

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

6.2.2 HSI 时钟

HSI 时钟信号由内部 HSI 的振荡器产生，可直接作为系统时钟或作为 PLL 输入。HSI 振荡器能够在不需要任何外部器件的条件下提供系统时钟。它的启动时间比 HSE 晶体振荡器短。然而，即使在校准之后它的时钟频率精度仍较差。

校准

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

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

如果 HSE 晶体振荡器失效，HSI 时钟会被作为备用时钟源, 参考小节 6.2.7。

6.2.3 PLL

内部 PLL 可以用来倍频 HSI 振荡器的输出时钟或 HSE 晶体输出时钟。参考图 12和时钟控制寄存器。PLL 的设置 (选择 HSI 振荡器或 HSE 振荡器为 PLL 的输入时钟，和选择倍频因子) 必须在其被激活前完成。一旦 PLL 被激活，这些参数就不能被改动。

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

如果需要在应用中使用 USB 接口，PLL 必须被设置为输出 48 或 96MHz 时钟，用于提供 48MHz 的 USBCLK 时钟。

6.2.4 LSE 时钟

LSE 晶体是一个 32.768KHz 的低速外部晶体或陶瓷谐振器。它为实时时钟或者其他定时功能提供一个低功耗且精确的时钟源。

LSE 晶体通过在备份域控制寄存器 (RCC_BDCR) 里的 LSEON 位启动和关闭。在备份在备份域控制寄存器 (RCC_BDCR) 里的 LSERDY 指示 LSE 晶体振荡是否稳定。在启动阶段，直到这个位被硬件置‘1’后，LSE 时钟信号才被释放出来。如果在时钟中断寄存器里被允许，可产生中断请求。

外部时钟源 (LSE 旁路)

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

6.2.5 LSI 时钟

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

LSI 振荡器可以通过控制/状态寄存器 (RCC_CSR) 里的 LSION 位来启动或关闭。在控制/状态寄存器 (RCC_CSR) 里的 LSIRDY 位指示低速内部振荡器是否稳定。在启动阶段，直到这个位被硬件设置为‘1’后，此时钟才被释放。如果在时钟中断寄存器 (RCC_CIR) 里被允许，将产生 LSI 中断请求。

6.2.6 系统时钟 (SYSCLK) 选择

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

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

在时钟配置寄存器 (RCC_CFGR) 里的状态位指示哪个时钟已经准备好了，哪个时钟目前被用作系统时钟。

6.2.7 时钟安全系统 (CSS)

时钟安全系统可以通过软件被激活。一旦其被激活，时钟监测器将在 HSE 振荡器启动延迟后被使能，并在 HSE 时钟关闭后关闭。

如果 HSE 时钟发生故障，HSE 振荡器被自动关闭，时钟失效事件将被送到高级定时器 TIM1 的刹车输入端，并产生时钟安全中断 CSS，允许软件完成营救操作。此 CSS 中断连接到 CPU 的 NMI 中断。

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

如果 HSE 振荡器被直接或间接地作为系统时钟，(间接的意思是：它被作为 PLL 输入时钟，并且 PLL 时钟被作为系统时钟)，时钟故障将导致系统时钟自动切换到 HSI 振荡器，同时外部 HSE 振荡器被关闭。在时钟失效时，如果 HSE 振荡器时钟 (被分频或未被分频) 是用作系统时钟的 PLL 的输入时钟，PLL 也将被关闭。

6.2.8 RTC 时钟

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

LSE 时钟在备份域里，但 HSE 和 LSI 时钟不是。因此：

- 如果 LSE 被选为 RTC 时钟：
  - 只要 V_{BAT} 维持供电，尽管 V_{DD} 供电被切断，RTC 仍继续工作。
- 如果 LSI 被选为自动唤醒单元 (AWU) 时钟：详见小节 6.2.5。
  - 如果 V_{DD} 供电被切断，AWU 状态不能被保证
- 如果 HSE 时钟 128 分频后作为 RTC 时钟：
  - 如果 V_{DD} 供电被切断或内部电压调压器被关闭 (1.5V 域的供电被切断)，则 RTC 状态不确定。
  - 必须设置电源控制寄存器的 DBP 位 (取消后备区域的写保护) 为‘1’。

6.2.9 看门狗时钟

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

6.2.10 时钟输出

微控制器允许输出时钟信号到外部 MCO 管脚。

相应的 GPIO 端口寄存器必须被配置为相应功能。以下 6 个时钟信号可被选作 MCO 时钟：

- SYSCLK
- HSI
- HSE
- LSI
- LSE
- PLLCLK/2
  - 时钟的选择由时钟配置寄存器 (RCC_CFGR) 中的 MCO[2：0] 位控制。

6.3 RCC 寄存器堆与存储器映射描述

表 17. RCC 寄存器概览

Offset	Acronym	Register Name	Reset	Section
0x00	RCC_CR	时钟控制寄存器	0x0000FF01	小节 6.3.1
0x04	RCC_CFGR	时钟配置寄存器	0x00000000	小节 6.3.2
0x08	RCC_CIR	时钟中断寄存器	0x00000000	小节 6.3.3
0x0C	RCC_APB2RSTR	APB2 外设复位寄存器	0x00000000	小节 6.3.4
0x10	RCC_APB1RSTR	APB1 外设复位寄存器	0x00000000	小节 6.3.5
0x14	RCC_AHBENR	AHB 外设时钟使能寄存器	0x00000014	小节 6.3.6
0x18	RCC_APB2ENR	APB2 外设时钟使能寄存器	0x00000000	小节 6.3.7
0x1C	RCC_APB1ENR	APB1 外设时钟使能寄存器	0x00000000	小节 6.3.8
0x20	RCC_BDCR	备份域控制寄存器	0x00000000	小节 6.3.9
0x24	RCC_CSR	控制状态寄存器	0xXC000000	小节 6.3.10
0x40	RCC_SYSCFG	系统配置寄存器	0x00000003	小节 6.3.11

6.3.1 时钟控制寄存器 (RCC_CR)

地址偏移：0x00

复位值：0x0000 FF01

访问：无等待状态，字，半字和字节访问

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

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

Bit	Field	Type	Reset	Description
31：26	PLLMUL	rw	0x00	PLLMUL：PLL 倍频系数 (PLL multiplication factor)
25	PLLRDY	r	0x00	PLLRDY：PLL 时钟就绪标志 (PLL clock ready flag) PLL 锁定后由硬件置‘1’。 0：PLL 未锁定 1：PLL 锁定
24	PLLON	rw	0x00	PLLON：PLL 使能 (PLL enable) 由软件置‘1’或清零。 当进入待机和停机模式时，该位由硬件清零。当 PLL 时钟被用作或被选择将要作为系统时钟时，该位不能被清零。 0：PLL 关闭 1：PLL 使能
23	Reserved			始终读为 0。
22：20	PLLDIV	rw	0x00	PLLDIV：(PLL divider factor) PLL 配置公式：$F_{CLKO} = F_{REFIN} * N / M$ 其中：F_{CLKO} 是 PLL 输出频率，F_{REFIN} 是 PLL 输入参考时钟频率 M = PLLDIV + 1 N = PLLMUL + 1
19	CSSON	rw	0x00	CSSON：时钟安全系统使能 (Clock security system enable) 由软件置‘1’或清零以使能时钟监测器。 0：时钟监测器关闭 1：如果外部振荡器就绪，时钟监测器开启
18	HSEBYP	rw	0x00	HSEBYP：外部高速时钟旁路 (External high-speed clock bypass) 在调试模式下由软件置‘1’或清零来旁路外部晶体振荡器。只有外部振荡器关闭的情况下，才能写入该位。 0：外部振荡器没有旁路 1：外部外部晶体振荡器被旁路
17	HSERDY	r	0x00	HSERDY：外部高速时钟就绪标志 (External high-speed clock ready flag) 由硬件置‘1’来指示外部时钟已经稳定。 0：外部时钟没有就绪 1：外部时钟就绪
16	HSEON	rw	0x00	HSEON：外部高速时钟使能 (External high-speed clock enable) 由软件置‘1’或清零。 当进入待机和停机模式时，该位由硬件清零，关闭外部时钟。当外部时钟被用作或被选择将要作为系统时钟时，该位不能被清零。 0：HSE 振荡器关闭 1：HSE 振荡器开启

Bit	Field	Type	Reset	Description
15: 8	HSICAL	r	0xFF	HSICAL：内部高速时钟校准 (Internal high-speed clock calibration) 在系统启动时，这些位被自动初始化
7: 2	Reserved			始终读为 0。
1	HSIRDY	r	0x00	HSIRDY：内部高速时钟就绪标志 (Internal high-speed clock ready flag) 由硬件置‘1’来指示内部 8MHz 时钟已经稳定。在 HSION 位清零后，该位需要 6 个内部 8MHz 时钟周期清零。 0：内部 8MHz 时钟没有就绪 1：内部 8MHz 时钟就绪
0	HSION	rw	0x01	HSION：内部高速时钟使能 (Internal high-speed clock enable) 由软件置‘1’或清零。 当从待机和停机模式返回或用作系统时钟的外部时钟发生故障时，该位由硬件置‘1’来启动内部 8MHz 的振荡器。 当内部 8MHz 时钟被直接或间接地用作或被选择将要作为系统时钟时，该位不能被清零。 0：内部 8MHz 时钟关闭 1：内部 8MHz 时钟开启

6.3.2 时钟配置寄存器 (RCC_CFGR)

地址偏移：0x04

复位值：0x0000 0000

访问：无等待状态，字，半字和字节访问

只有当访问发生在时钟切换时，才会插入 1 或 2 个等待周期。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved					MCO			USBPRE		Reserved				PLLXTPRE	PLLSRC
					rw	rw	rw	rw	rw					rw	rw

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

Bit	Field	Type	Reset	Description
31 : 27	Reserved			始终读为 0

Bit	Field	Type	Reset	Description
26：24	MCO	rw	0x00	MCO：微控制器时钟输出 (Microcontroller clock output) 由软件置‘1’或清零。 00x：没有时钟输出； 010：LSI 时钟输出； 011：LSE 时钟输出； 100：系统时钟 (SYSCLK) 输出； 101：HSI 时钟输出； 110：HSE 时钟输出； 111：PLL 时钟 2 分频后输出。 注意： 1. 该时钟输出在启动和切换 MCO 时钟源时可能会被截断。 2. 系统时钟作为输出至 MCO 管脚时，请保证输出时钟频率不超过 50MHz(IO 口最高频率)
23：22	USBPRE	rw	0x00	USBPRE：USB 预分频 (USB prescaler) 由软件置‘1’或清‘0’来产生 48MHz 的 USB 时钟。在 RCC_APB1ENR 寄存器中使能 USB 时钟之前，必须保证该位已经有效。 00：PLL 时钟直接作为 USB 时钟 01：PLL 时钟 2 分频作为 USB 时钟 10：PLL 时钟 3 分频作为 USB 时钟 11：PLL 时钟 4 分频作为 USB 时钟
21：18	Reserved			始终读为 0
17	PLLXTPRE	rw	0x00	PLLXTPRE：HSE 分频器作为 PLL 输入 (HSE divider for PLL entry) 由软件置‘1’或清‘0’来分频 HSE 后作为 PLL 输入时钟。该位只有在 PLL 关闭时才可以被写入。 0：HSE 不分频 1：HSE2 分频
16	PLLSRC	rw	0x00	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。该位只有在 PLL 关闭时才可以被写入。 0：HSI 时钟 4 分频后作为 PLL 输入时钟 1：HSE 时钟作为 PLL 输入时钟
15: 14	Reserved			始终读为 0
13: 11	PPRE2	rw	0x00	PPRE2：APB2 分频系数 由软件置‘1’或清‘0’来控制高速 APB2 时钟 (PCLK2) 的预分频系数。 0xx：HCLK 不分频 100：HCLK 2 分频 101：HCLK 4 分频 110：HCLK 8 分频 111：HCLK 16 分频

Bit	Field	Type	Reset	Description
10: 8	PPRE1	rw	0x00	PPRE1：APB1 分频系数 由软件置‘1’或清‘0’来控制低速 APB1 时钟 (PCLK1) 的预分频系数。 0xx：HCLK 不分频 100：HCLK 2 分频 101：HCLK 4 分频 110：HCLK 8 分频 111：HCLK 16 分频
7: 4	HPRE	rw	0x00	HPRE：AHB 预分频 (AHB Prescaler) 由软件置‘1’或清‘0’来控制 AHB 时钟的预分频系数。 0xxx：SYSCLK 不分频 1000：SYSCLK 2 分频 1001：SYSCLK 4 分频 1010：SYSCLK 8 分频 1011：SYSCLK 16 分频 1100：SYSCLK 64 分频 1101：SYSCLK 128 分频 1110：SYSCLK 256 分频 1111：SYSCLK 512 分频 注意： 1. 当 AHB 时钟的预分频系数大于 1 时，必须开启预取缓冲器。详见读闪存存储器一节。
3: 2	SWS	r	0x00	SWS：系统时钟开关状态 (System clock switch status) 00：HSI 6 分频作为系统时钟； 01：HSE 作为系统时钟； 10：PLL 输出作为系统时钟； 11：不可用。
1: 0	SW	rw	0x00	SW：系统时钟开关 (System clock switch) 由软件置‘1’或清‘0’来选择系统时钟源。 在从停止或待机模式中返回时或直接或间接作为系统时钟的 HSE 出现故障时，由硬件强制选择 00：HSI 6 分频作为系统时钟； 01：HSE 作为系统时钟； 10：PLL 输出作为系统时钟； 11：不可用。

6.3.3 时钟中断寄存器 (RCC_CIR)

地址偏移：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
Reserved								CSSC	Reserved		PLL RDYC	HSE RDYC	HSI RDYC	LSE RDYC	LSI RDYC
								rc_w1			rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			PLL RDYIE	HSE RDYIE	HSI RDYIE	LSE RDYIE	LSI RDYIE	CSSF	Reserved		PLL RDYF	HSE RDYF	HSI RDYF	LSE RDYF	LSI RDYF
			rw	rw	rw	rw	rw	r			r	r	r	r	r

Bit	Field	Type	Reset	Description
31：24	Reserved			始终读为 0
23	CSSC	rc_w1	0x00	CSSC：清除时钟安全系统中断 (Clock security system interrupt clear) 由软件置‘1’来清除 CSSF 安全系统中断标志位 CSSF。 0：无作用 1：清除 CSSF 安全系统中断标志位
22：21	Reserved			始终读为 0
20	PLLRDYC	rc_w1	0x00	PLLRDYC：清除 PLL 就绪中断 (PLL ready interrupt clear) 由软件置‘1’来清除 PLL 就绪中断标志位 PLLRDYF。 0：无作用 1：清除 PLL 就绪中断标志位 PLLRDYF
19	HSERDYC	rc_w1	0x00	HSERDYC：清除 HSE 就绪中断 (HSE ready interrupt clear) 由软件置‘1’来清除 HSE 就绪中断标志位 HSERDYF。 0：无作用 1：清除 HSE 就绪中断标志位 HSERDYF
18	HSIRDYC	rc_w1	0x00	HSIRDYC：清除 HSI 就绪中断 (HSI ready interrupt clear) 由软件置‘1’来清除 HSI 就绪中断标志位 HSIRDYF。 0：无作用 1：清除 HSI 就绪中断标志位 HSIRDYF
17	LSERDYC	rc_w1	0x00	LSERDYC：清除 LSE 就绪中断 (LSE ready interrupt clear) 由软件置‘1’来清除 LSE 就绪中断标志位 LSERDYF。 0：无作用 1：清除 LSE 就绪中断标志位 LSERDYF
16	LSIRDYC	rc_w1	0x00	LSIRDYC：清除 LSI 就绪中断 (LSI ready interrupt clear) 由软件置‘1’来清除 LSI 就绪中断标志位 LSIRDYF。 0：无作用 1：清除 LSI 就绪中断标志位 LSIRDYF
15: 13	Reserved			始终读为 0

Bit	Field	Type	Reset	Description
12	PLLRDYIE	rw	0x00	PLLRDYIE：PLL 就绪中断使能 (PLL ready interrupt enable) 由软件置‘1’或清‘0’来使能或关闭 PLL 就绪中断。 0：PLL 就绪中断关闭 1：PLL 就绪中断使能
11	HSERDYIE	rw	0x00	HSERDYIE：HSE 就绪中断使能 (HSE ready interrupt enable) 由软件置‘1’或清‘0’来使能或关闭外部振荡器就绪中断。 0：HSE 就绪中断关闭 1：HSE 就绪中断使能
10	HSIRDYIE	rw	0x00	HSIRDYIE：HSI 就绪中断使能 (HSI ready interrupt enable) 由软件置‘1’或清‘0’来使能或关闭内部 8MHz 振荡器就绪中断。 0：HSI 就绪中断关闭 1：HSI 就绪中断使能
9	LSERDYIE	rw	0x00	LSERDYIE：LSE 就绪中断使能 (LSE ready interrupt enable) 由软件置‘1’或清‘0’来使能或关闭外部 32KHz 振荡器就绪中断。 0：LSE 就绪中断关闭 1：LSE 就绪中断使能
8	LSIRDYIE	rw	0x00	LSIRDYIE：LSI 就绪中断使能 (LSI ready interrupt enable) 由软件置‘1’或清‘0’来使能或关闭内部 40KHz 振荡器就绪中断。 0：LSI 就绪中断关闭 1：LSI 就绪中断使能
7	CSSF	r	0x00	CSSF：时钟安全系统中断标志 (Clock security system interrupt flag) 在外部振荡器时钟出现故障时，由硬件置‘1’。 由软件通过置‘1’CSSC 位来清除。 0：无 HSE 时钟失效产生的安全系统中断 1：HSE 时钟失效导致了时钟安全系统中断
6: 5	Reserved			始终读为 0
4	PLLRDYF	r	0x00	PLLRDYF：PLL 就绪中断标志 (PLL ready interrupt flag) 在 PLL 就绪且 PLLRDYIE 位被置‘1’时，由硬件置‘1’。 由软件通过置‘1’PLLRDYC 位来清除。 0：无 PLL 上锁产生的时钟就绪中断 1：PLL 上锁导致时钟就绪中断

Bit	Field	Type	Reset	Description
3	HSERDYF	r	0x00	HSERDYF：HSE 就绪中断标志 (HSE ready interrupt flag) 在外部低速时钟就绪且 HSERDYIE 位被置‘1’时，由硬件置‘1’。 由软件通过置‘1’HSERDYC 位来清除。 0：无外部振荡器产生的时钟就绪中断 1：外部振荡器导致时钟就绪中断
2	HSIRDYF	r	0x00	HSIRDYF：HSI 就绪中断标志 (HSI ready interrupt flag) 在内部高速时钟就绪且 HSIRDYIE 位被置‘1’时，由硬件置‘1’。 由软件通过置‘1’HSIRDYC 位来清除。 0：无内部 HSI 振荡器产生的时钟就绪中断 1：内部 HSI 振荡器导致时钟就绪中断
1	LSERDYF	r	0x00	LSERDYF：LSE 就绪中断标志 (LSE ready interrupt flag) 在外部低速时钟就绪且 LSERDYIE 位被置‘1’时，由硬件置‘1’。 由软件通过置‘1’LSERDYC 位来清除。 0：无外部 32KHz 振荡器产生的时钟就绪中断； 1：外部 32KHz 振荡器导致时钟就绪中断。
0	LSIRDYF	r	0x00	LSIRDYF：LSI 就绪中断标志 (LSI ready interrupt flag) 在内部低速时钟就绪且 LSIRDYIE 位被置‘1’时，由硬件置‘1’。 由软件通过置‘1’LSIRDYC 位来清除。 0：无内部 40KHz 振荡器产生的时钟就绪中断； 1：内部 40KHz 振荡器导致时钟就绪中断。

6.3.4 APB2 外设复位寄存器 (RCC_APB2RSTR)

地址偏移：0x0C

复位值：0x0000 0000

访问：无等待周期，字，半字和字节访问

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	UART1	Res.	SPI1	TIM1	ADC2	ADC1	Reserved		IOPE	IOPD	IOPC	IOPB	IOPA	Res.	AFIO
	rw		rw	rw	rw	rw			rw	rw	rw	rw	rw		rw

Bit	Field	Type	Reset	Description
31：15	Reserved			始终读为 0

Bit	Field	Type	Reset	Description
14	UART1	rw	0x00	UART1：UART1 复位 (UART1 reset) 由软件置‘1’或清‘0’。 0：无作用 1：复位 UART1
13	Reserved			始终读为 0
12	SPI1	rw	0x00	SPI1：SPI1 复位 (SPI1 reset) 由软件置‘1’或清‘0’。 0：无作用 1：复位 SPI1
11	TIM1	rw	0x00	TIM1：TIM1 定时器复位 (TIM1 timer reset) 由软件置‘1’或清‘0’。 0：无作用 1：复位 TIM1 定时器
10	ADC2	rw	0x00	ADC2:ADC2 接口复位 (ADC2 interface reset) 由软件置‘1’或清‘0’。 0：无作用 1：复位 ADC2 接口
9	ADC1	rw	0x00	ADC1:ADC1 接口复位 (ADC1 interface reset) 由软件置‘1’或清‘0’。 0：无作用 1：复位 ADC1 接口
8: 7	Reserved			始终读为 0
6	IOPE	rw	0x00	IOPE：IO 端口 E 复位 (IO port E reset) 由软件置‘1’或清‘0’。 0：无作用 1：复位 IO 端口 E
5	IOPD	rw	0x00	IOPD：IO 端口 D 复位 (IO port D reset) 由软件置‘1’或清‘0’。 0：无作用 1：复位 IO 端口 D
4	IOPC	rw	0x00	IOPC：IO 端口 C 复位 (IO port C reset) 由软件置‘1’或清‘0’。 0：无作用 1：复位 IO 端口 C
3	IOPB	rw	0x00	IOPB：IO 端口 B 复位 (IO port B reset) 由软件置‘1’或清‘0’。 0：无作用 1：复位 IO 端口 B
2	IOPA	rw	0x00	IOPA：IO 端口 A 复位 (IO port A reset) 由软件置‘1’或清‘0’。 0：无作用 1：复位 IO 端口 A

Bit	Field	Type	Reset	Description
1	Reserved			始终读为 0
0	AFIO	rw	0x00	AFIO：辅助功能 IO 复位 (Alternate function I/O reset) 由软件置‘1’或清‘0’。 0：无作用 1：复位辅助功能

6.3.5 APB1 外设复位寄存器 (RCC_APB1RSTR)

地址偏移：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
Reserved			PWR	BKP	Res.	CAN	Res.	USB	I2C2	I2C1	Reserved		UART3	UART2	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	SIP2	Reserved		WWDG	Reserved								TIM4	TIM3	TIM2

Bit	Field	Type	Reset	Description
31：29	Reserved			始终读为 0
28	PWR	rw	0x00	PWR：电源接口复位 (Power interface reset) 由软件置‘1’或清‘0’。 0：无作用 1：复位电源接口
27	BKP	rw	0x00	BKP：备份接口复位 (Backup interface reset) 由软件置‘1’或清‘0’。 0：无作用 1：复位备份接口
26	Reserved			始终读为 0
25	CAN	rw	0x00	CAN：CAN 复位 (CAN reset) 由软件置‘1’或清‘0’。 0：无作用 1：复位 CAN
24	Reserved			始终读为 0
23	USB	rw	0x00	USB：USB 复位 (USB reset) 由软件置‘1’或清‘0’。 0：无作用 1：复位 USB
22	I2C2	rw	0x00	I2C2：I2C2 复位 (I2C2 reset) 由软件置‘1’或清‘0’。 0：无作用 1：复位 I2C2

Bit	Field	Type	Reset	Description
21	I2C1	rw	0x00	I2C1：I2C1 复位 (I2C1 reset) 由软件置‘1’或清‘0’。 0：无作用 1：复位 I2C1
20：19	Reserved			始终读为 0
18	UART3	rw	0x00	UART3：UART3 复位 (UART3 reset) 由软件置‘1’或清‘0’。 0：无作用 1：复位 UART3
17	UART2	rw	0x00	UART2：UART2 复位 (UART2 reset) 由软件置‘1’或清‘0’。 0：无作用 1：复位 UART2
16：15	Reserved			始终读为 0
14	SPI2	rw	0x00	SPI2：SPI2 复位 (SPI2 reset) 由软件置‘1’或清‘0’。 0：无作用 1：复位 SPI2
13：12	Reserved			始终读为 0
11	WWDG	rw	0x00	WWDG：窗口看门狗复位 (Window watchdog reset) 由软件置‘1’或清‘0’。 0：无作用 1：复位窗口看门狗
10：3	Reserved			始终读为 0
2	TIM4	rw	0x00	TIM4：定时器 4 复位 (Timer4 reset) 由软件置‘1’或清‘0’。 0：无作用 1：复位 TIM4 定时器
1	TIM3	rw	0x00	TIM3：定时器 3 复位 (Timer3 reset) 由软件置‘1’或清‘0’。 0：无作用 1：复位 TIM3 定时器
0	TIM2	rw	0x00	TIM2：定时器 2 复位 (Timer2 reset) 由软件置‘1’或清‘0’。 0：无作用 1：复位 TIM2 定时器

6.3.6 AHB 外设时钟使能寄存器 (RCC_AHBENR)

地址偏移：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
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								AES	CRC	Res.	FLASH	Res.	SRAM	Res.	DMA
								rw	rw		rw		rw		rw

Bit	Field	Type	Reset	Description
31：8	Reserved			始终读为 0
7	AES	rw	0x00	AES：AES 时钟使能 (AES clock enable) 由软件置‘1’或清‘0’。 0：AES 时钟关闭 1：AES 时钟开启
6	CRC	rw	0x00	CRC：CRC 时钟使能 (CRC clock enable) 由软件置‘1’或清‘0’。 0：CRC 时钟关闭 1：CRC 时钟开启
5	Reserved			始终读为 0
4	FLASH	rw	0x01	FLASH：FLASH 时钟使能 (FLASH clock enable) 0：FLASH 时钟关闭 1：FLASH 时钟开启
3	Reserved			始终读为 0
2	SRAM	rw	0x01	SRAM：SRAM 时钟使能 (SRAM interface clock enable) 由软件置‘1’或清‘0’来开启或关闭睡眠模式时 SRAM 时钟。 0：睡眠模式时 SRAM 时钟关闭 1：睡眠模式时 SRAM 时钟开启
1	Reserved			始终读为 0
0	DMA	rw	0x00	DMA：DMA 时钟使能 (DMA clock enable) 由软件置‘1’或清‘0’。 0：DMA 时钟关闭 1：DMA 时钟开启

6.3.7 APB2 外设时钟使能寄存器 (RCC_APB2ENR)

地址偏移：0x18

复位值：0x0000 0000

访问：无等待周期，字，半字和字节访问

注：当外设时钟没有启动时，软件不能读出外设寄存器的数值

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	UART1	Res.	SPI1	TIM1	ADC2	ADC1	Reserved			IOPD	IOPC	IOPB	IOPA	Res.	AFIO
	rw		rw	rw	rw	rw				rw	rw	rw	rw		rw

Bit	Field	Type	Reset	Description
31：15	Reserved			始终读为 0
14	UART1	rw	0x00	UART1：UART1 时钟使能 (UART 1 clock enable) 由软件置‘1’或清‘0’。 0：UART1 时钟关闭 1：UART1 时钟开启
13	Reserved			始终读为 0
12	SPI1	rw	0x00	SPI1：SPI1 时钟使能 (SPI 1 clock enable) 由软件置‘1’或清‘0’。 0：SPI1 时钟关闭 1：SPI1 时钟开启
11	TIM1	rw	0x00	TIM1：TIM1 定时器时钟使能 (TIM1 Timer clock enable) 由软件置‘1’或清‘0’。 0：TIM1 定时器时钟关闭 1：TIM1 定时器时钟开启
10	ADC2	rw	0x00	ADC2：ADC2 接口时钟使能 (ADC2 interface clock enable) 由软件置‘1’或清‘0’。 0：ADC2 接口时钟关闭 1：ADC2 接口时钟开启
9	ADC1	rw	0x00	ADC1：ADC1 接口时钟使能 (ADC1 interface clock enable) 由软件置‘1’或清‘0’。 0：ADC1 接口时钟关闭 1：ADC1 接口时钟开启
8：6	Reserved			始终读为 0
5	IOPD	rw	0x00	IOPD：IO 端口 D 时钟使能 (I/O port D clock enable) 由软件置‘1’或清‘0’。 0：IO 端口 D 时钟关闭 1：IO 端口 D 时钟开启
4	IOPC	rw	0x00	IOPC：IO 端口 C 时钟使能 (I/O port C clock enable) 由软件置‘1’或清‘0’。 0：IO 端口 C 时钟关闭 1：IO 端口 C 时钟开启
3	IOPB	rw	0x00	IOPB：IO 端口 B 时钟使能 (I/O port B clock enable) 由软件置‘1’或清‘0’。 0：IO 端口 B 时钟关闭 1：IO 端口 B 时钟开启

Bit	Field	Type	Reset	Description
2	IOPA	rw	0x00	IOPA：IO 端口 A 时钟使能 (I/O port A clock enable) 由软件置‘1’或清‘0’。 0：IO 端口 A 时钟关闭 1：IO 端口 A 时钟开启
1	Reserved			始终读为 0
0	AFIO	rw	0x00	AFIO：辅助功能 IO 时钟使能 (Alternate function I/O clock enable) 由软件置‘1’或清‘0’。 0：辅助功能 IO 时钟关闭 1：辅助功能 IO 时钟开启

6.3.8 APB1 外设时钟使能寄存器 (RCC_APB1ENR)

地址偏移：0x1C

复位值：0x0000 0000

访问：无等待周期，字，半字和字节访问

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved			PWR	BKP	Res.	CAN	Res.	USB	I2C2	I2C1	Reserved		UART3	UART2	Res.
			rw	rw		rw		rw	rw	rw			rw	rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	SIP2	Reserved		WWDG	Reserved								TIM4	TIM3	TIM2
	rw			rw									rw	rw	rw

Bit	Field	Type	Reset	Description
31：29	Reserved			始终读为 0
28	PWR	rw	0x00	PWR：电源接口时钟使能 (Power interface clock enable) 由软件置‘1’或清‘0’。 0：电源接口时钟关闭 1：电源接口时钟开启
27	BKP	rw	0x00	BKP：备份接口时钟使能 (Backup interface clock enable) 由软件置‘1’或清‘0’。 0：备份接口时钟关闭 1：备份接口时钟开启
26	Reserved			始终读为 0
25	CAN	rw	0x00	CAN：CAN 时钟使能 (CAN clock enable) 由软件置‘1’或清‘0’。 0：CAN 时钟关闭 1：CAN 时钟开启
24	Reserved			始终读为 0

Bit	Field	Type	Reset	Description
23	USB	rw	0x00	USB：USB 时钟使能 (USB clock enable) 由软件置‘1’或清‘0’。 0：USB 时钟关闭 1：USB 时钟开启
22	I2C2	rw	0x00	I2C2：I2C2 时钟使能 (I2C2 clock enable) 由软件置‘1’或清‘0’。 0：I2C 2 时钟关闭 1：I2C 2 时钟开启
20：19	Reserved			始终读为 0
18	UART3	rw	0x00	UART3：UART3 时钟使能 (UART3 clock enable) 由软件置‘1’或清‘0’。 0：UART3 时钟关闭 1：UART3 时钟开启
17	UART2	rw	0x00	UART2：UART2 时钟使能 (UART2 clock enable) 由软件置‘1’或清‘0’。 0：UART2 时钟关闭 1：UART2 时钟开启
16：15	Reserved			始终读为 0
14	SPI2	rw	0x00	SPI2：SPI2 时钟使能 (SPI2 clock enable) 由软件置‘1’或清‘0’。 0：SPI2 时钟关闭 1：SPI2 时钟开启
13：12	Reserved			始终读为 0
11	WWDG	rw	0x00	WWDG：窗口看门狗时钟使能 (Window watchdog clock enable) 由软件置‘1’或清‘0’。 0：窗口看门狗时钟关闭 1：窗口看门狗时钟开启
10：3	Reserved			始终读为 0
2	TIM4	rw	0x00	TIM4：定时器 4 时钟使能 (Timer4 clock enable) 由软件置‘1’或清‘0’。 0：定时器 4 时钟关闭 1：定时器 4 时钟开启
1	TIM3	rw	0x00	TIM3：定时器 3 时钟使能 (Timer3 clock enable) 由软件置‘1’或清‘0’。 0：定时器 3 时钟关闭 1：定时器 3 时钟开启
0	TIM2	rw	0x00	TIM2：定时器 2 时钟使能 (Timer2 clock enable) 由软件置‘1’或清‘0’。 0：定时器 2 时钟关闭 1：定时器 2 时钟开启

6.3.9 备份域控制寄存器器 (RCC_BDCR)

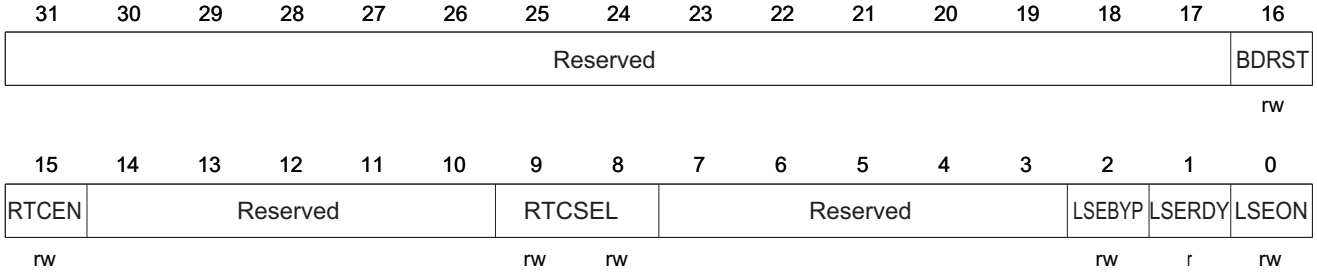
地址偏移：0x20

复位值：0x0000 0000, 只能由备份域复位有效复位

访问：0~3 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。

注：备份域控制寄存器中 (RCC_BDCR) 的 LSEON、LSEBYP、RTCSEL 和 RTCEN 位处于备份域。因此，这些位在复位后处于写保护状态，只有在电源控制寄存器 (PWR_CR) 中的 DBP 位置‘1’后才能对这些位进行改动。这些位只能由备份域复位清除。任何内部或外部复位都不会影响这些位。



Bit	Field	Type	Reset	Description
31：17	Reserved			始终读为 0
16	BDRST	rw	0x00	BDRST：备份域软件复位 (Backup domain software reset) 由软件置‘1’或清‘0’。 0：复位未激活 1：复位整个备份域
15	RTCEN	rw	0x00	RTCEN：RTC 时钟使能 (RTC clock enable) 由软件置‘1’或清‘0’。 0：RTC 时钟关闭 1：RTC 时钟开启
14：10	Reserved			始终读为 0
9：8	RTCSEL	rw	0x00	RTCSEL：RTC 时钟源选择 (RTC clock source selection) 由软件设置来选择 RTC 时钟源。一旦 RTC 时钟源被选定，直到下次后备域被复位，它不能在被改变。可通过设置 BDRST 位来清除。 00：无时钟 01：LSE 振荡器作为 RTC 时钟 10：LSI 振荡器作为 RTC 时钟 11：HSE 振荡器在 128 分频后作为 RTC 时钟
7：3	Reserved			始终读为 0
2	LSEBYP	rw	0x00	LSEBYP：外部低速时钟振荡器旁路 (External low-speed oscillator bypass) 在调试模式下由软件置‘1’或清‘0’来旁路 LSE。只有在外部 32KHz 振荡器关闭时，才能写入该位。 0：LSE 时钟未被旁路 1：LSE 时钟被旁路

Bit	Field	Type	Reset	Description
1	LSERDY	r	0x00	LSERDY：外部低速 LSE 就绪 (External low-speed oscillator ready) 由硬件置‘1’或清‘0’来指示是否外部 32KHz 振荡器就绪。在 LSEON 被清零后，该位需要 6 个外部低速振荡器的周期才被清零。 0：外部 32KHz 振荡器未就绪 1：外部 32KHz 振荡器就绪
0	LSEON	rw	0x00	LSEON：外部低速振荡器使能 (External low-speed oscillator enable) 由软件置‘1’或清‘0’ 0：外部 32KHz 振荡器关闭 1：外部 32KHz 振荡器开启

6.3.10 控制状态寄存器寄存器 (RCC_CSR)

地址偏移：0x24

复位值：0xXC00 0000

访问：0 ~ 3 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。

除复位标志位由系统复位清除，复位标志只能由电源复位清除。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	WWDG RSTF	IWDG RSTF	SFT RSTF	POR RSTF	PIN RSTF	Res.	RMVF	Reserved							
	r	r	r	r	r		rw								

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

Bit	Field	Type	Reset	Description
31	Reserved			始终读为 0
30	WWDGRSTF	r	0x0x	WDGRSTF：窗口看门狗复位标志 (Window watchdog reset flag) 在窗口看门狗复位发生时由硬件置‘1’。 由软件通过写 RMVF 位清除。 0：无窗口看门狗复位发生 1：发生窗口看门狗复位
29	IWDGRSTF	r	0x0x	IWDGRSTF：独立看门狗复位标志 (Independent watchdog reset flag) 在独立看门狗复位发生在 VDD 区域时由硬件置‘1’。 由软件通过写 RMVF 位清除。 0：无独立看门狗复位发生 1：发生独立看门狗复位

Bit	Field	Type	Reset	Description
28	SFTRSTF	r	0x0x	SFTRSTF：软件复位标志 (Software reset flag) 在软件复位发生时由硬件置‘1’。 由软件通过写 RMVF 位清除。 0：无软件复位发生 1：发生软件复位
27	PORRSTF	r	0x01	PORRSTF：上电/掉电复位标志 (POR/PDR reset flag) 在上电/掉电复位发生时由硬件置‘1’。 由软件通过写 RMVF 位清除。 0：无上电/掉电复位发生 1：发生上电/掉电复位
26	PINRSTF	r	0x01	PINRSTF：NRST 管脚复位标志 (PIN reset flag) 在 NRST 管脚复位发生时由硬件置‘1’。 由软件通过写 RMVF 位清除。 0：无 NRST 管脚复位发生 1：发生 NRST 管脚复位
25	Reserved			始终读为 0
24	RMVF	rw	0x00	RMVF：清除复位标志 (Remove reset flag) 由软件置‘1’来清除复位标志。 0：无作用 1：清除复位标志
23：2	Reserved			始终读为 0
1	LSIRDY	r	0x00	LSIRDY：内部低速时钟就绪 (Internal low-speed oscillator ready) 由硬件置‘1’或清‘0’来指示内部 40KHz 振荡器是否就绪。 在 LSION 清零后，3 个内部 40KHz 振荡器的周期后 LSIRDY 被清零。 0：内部 40KHz 振荡器时钟未就绪 1：内部 40KHz 振荡器时钟就绪
0	LSION	rw	0x00	LSION：内部低速振荡器使能 (Internal low-speed oscillator enable) 由软件置‘1’或清‘0’。 0：内部 40KHz 振荡器关闭 1：内部 40KHz 振荡器开启

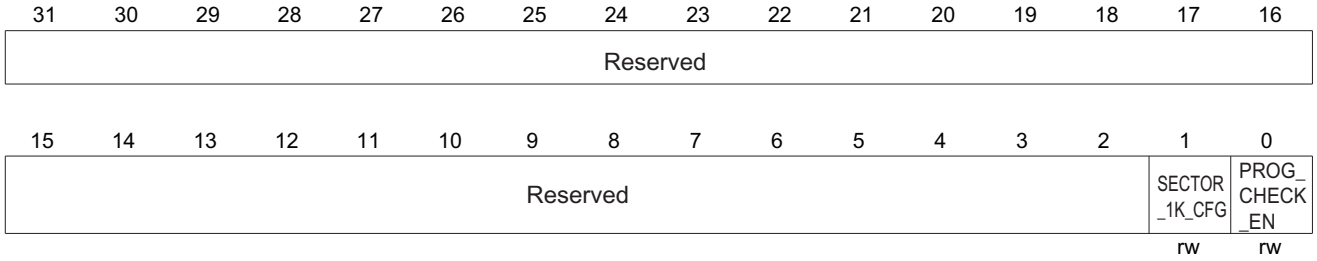
6.3.11 系统配置寄存器 (RCC_SYSCFG)

地址偏移：0x40

复位值：0x0000 0003

访问：0 ~ 3 等待周期，字，半字和字节访问

除复位标志位由系统复位清除，复位标志只能由电源复位清除。



Bit	Field	Type	Reset	Description
31：2	Reserved			始终读为 0
1	SECTOR_1K-_CFG	rw	0x01	SECTOR_1K_CFG ：Flash 页擦除时擦除的大小。 1：1K 字节 0：512 字节
0	PROG_-CHECK_EN	rw	0x01	PROG_CHECK_EN：写 Flash 时是否检查 Flash 内的数据是否是 FF。(硬件固定位 1 了) 1：检查 0：不检查

7 通用和复用功能 I/O(GPIO 和 AFIO)

通用和复用功能 I/O(GPIO 和 AFIO)

7.1 GPIO 功能描述

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

GPIO 端口的每个位可以由软件分别配置成多种模式。

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

每个 I/O 端口可以自由编程，然而必须按照 32 位字访问 I/O 端口寄存器 (不允许半字或字节访问)。

GPIOx_BSRR 和 GPIOx_BRR 寄存器允许对任何 GPIO 寄存器进行读/更改的独立访问；这样，在读更改访问之间产生 IRQ 不会发生危险。

下图给出了一个 I/O 端口位的基本结构。

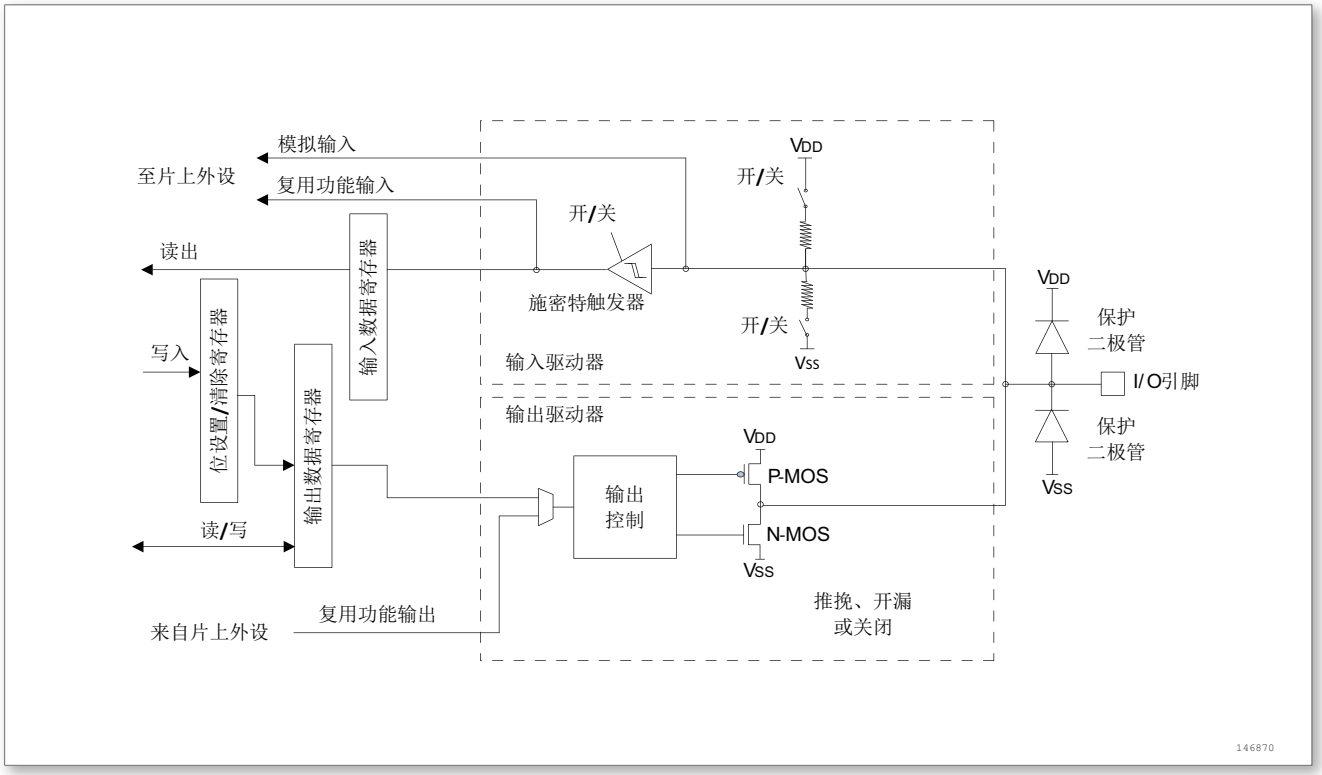


图 14. I/O 端口位的基本结构

表 18. 端口位配置表

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

表 19. 输出模式位

MODE[1：0]	意义
00	保留
01	输出

7.1.1 通用 I/O(GPIO)

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

复位后，JTAG 引脚被置于输入上拉或下拉模式：

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

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

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

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

7.1.2 单独的位设置或位清除

当 GPIOx_ODR 的个别位编程时，软件不需要禁止中断：

在单次 APB2 写操作里，可以只更改一个或多个位。

这是通过对‘置位/复位寄存器’(GPIOx_BSRR，复位是 GPIOx_BRR) 中想要更改的位写‘1’来实现的，没被选择的位将不被更改。

7.1.3 外部中断/唤醒线

所有端口都有外部中断能力。为了使用外部中断线，端口必须配置成输入模式。更多的关于外部中断的信息，参考 7.2节：外部中断/事件控制器 (EXTI)。

7.1.4 复用功能

使用默认复用功能前必须对端口位配置寄存器编程。

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

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

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

如果把端口配置成复用输出功能，则引脚和输出寄存器断开，并和片上外设的输出信号连接。如果软件把一个 GPIO 脚配置成复用输出功能，但是外设没有被激活，它的输出将不确定。

7.1.5 软件重新映射 I/O 复用功能

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

这时，复用功能就不再映射到它们的原始引脚上。

7.1.6 GPIO 锁定机制

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

7.1.7 输入配置

当 I/O 端口配置为输入时：

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

下图给出了 I/O 端口位的输入配置：

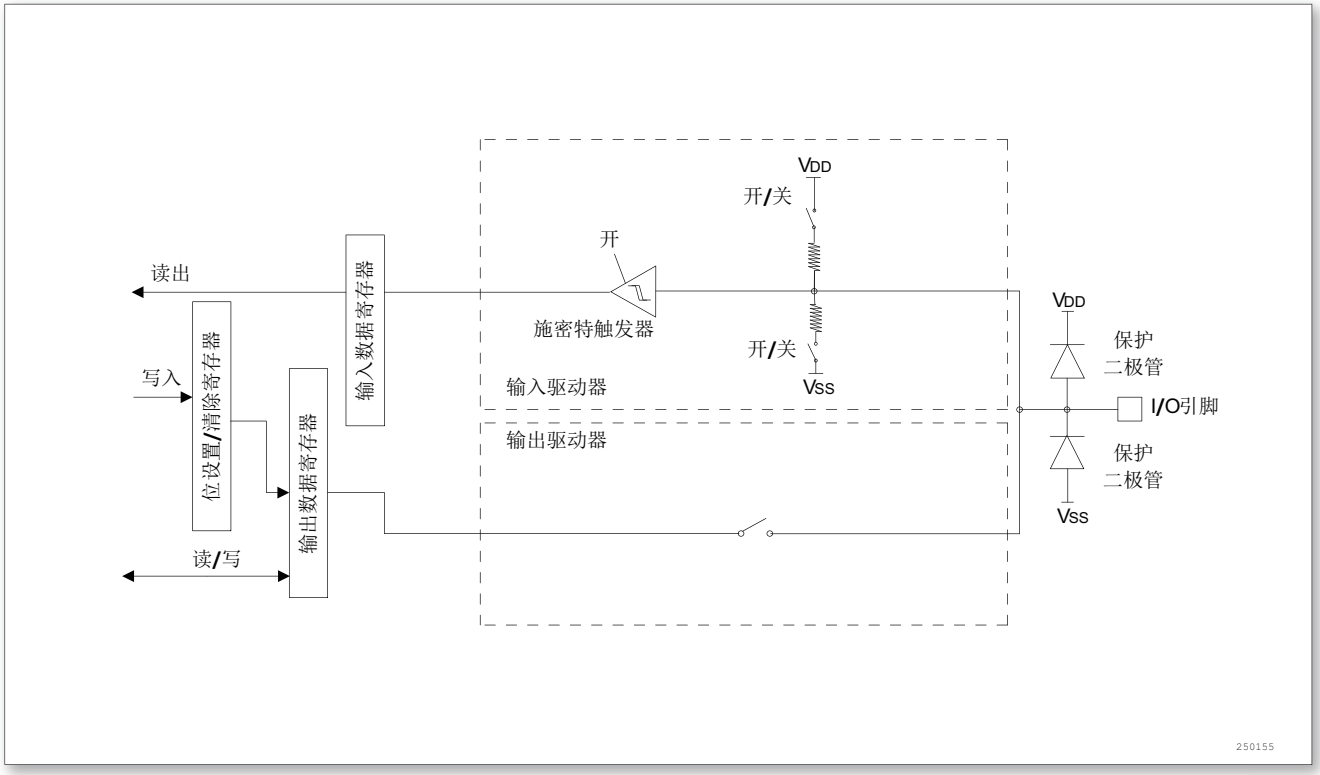


图 15. 输入浮空/上拉/下拉配置

7.1.8 输出配置

当 I/O 端口被配置为输出时：

- 输出缓冲器被激活
  - 开漏模式：输出寄存器上的‘0’激活 N-MOS，而输出寄存器上的‘1’将端口置于高阻状态 (P-MOS 从不被激活)
  - 推挽模式：输出寄存器上的‘0’激活 N-MOS，而输出寄存器上的‘1’将激活 P-MOS
- 施密特输入被激活
- 弱上拉和下拉电阻被禁止
- 出现在 I/O 脚上的数据在每个 APB2 时钟被采样到输入数据寄存器

- 在开漏模式时，对输入数据寄存器的访问可得到 I/O 状态
- 在推挽模式时，可对输出数据寄存器的读访问得到最后一次写的值。

下图给出了 I/O 端口位的输出配置：

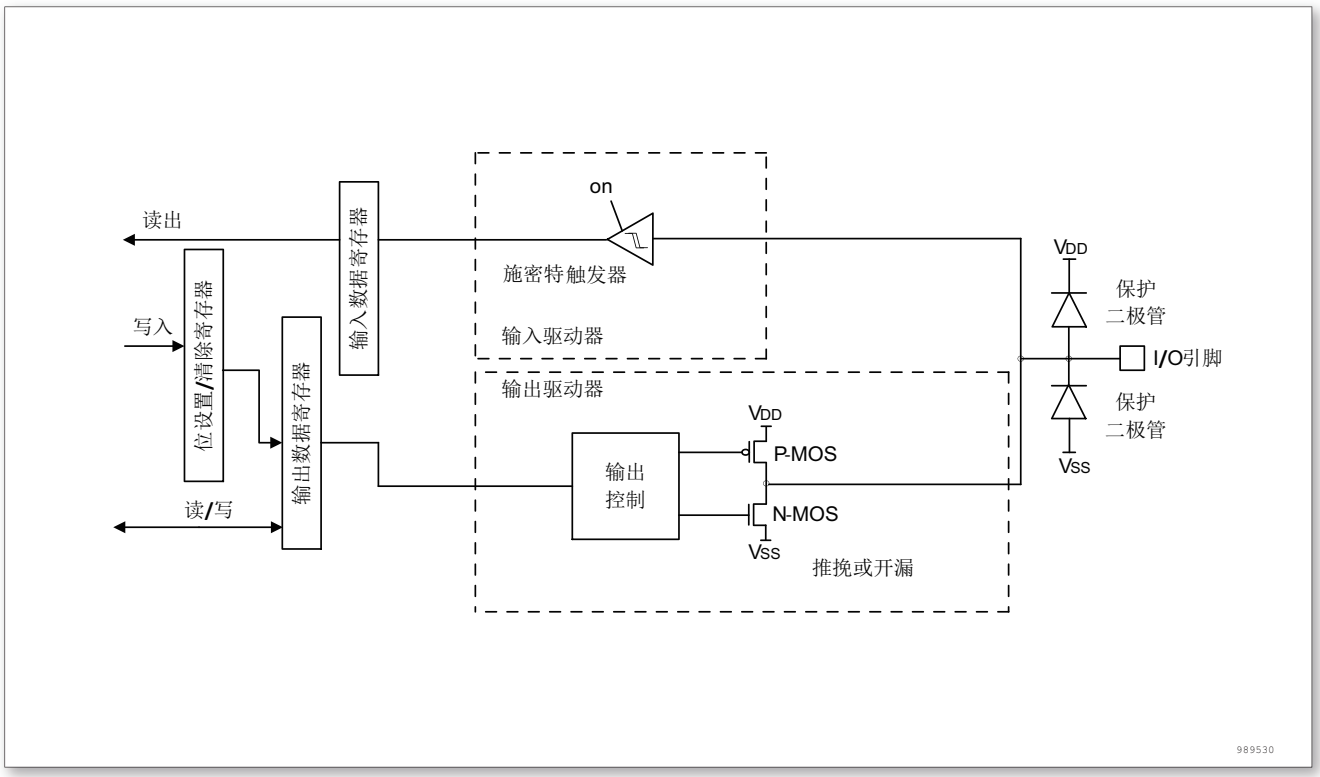


图 16. 输出配置

7.1.9 复用功能配置

当 I/O 端口被配置为复用功能时：

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

下图给出了 I/O 端口为复用功能的配置。详见 AFIO 寄存器描述。

一组复用功能 I/O 寄存器允许用户把一些复用功能重新映射到不同的引脚。

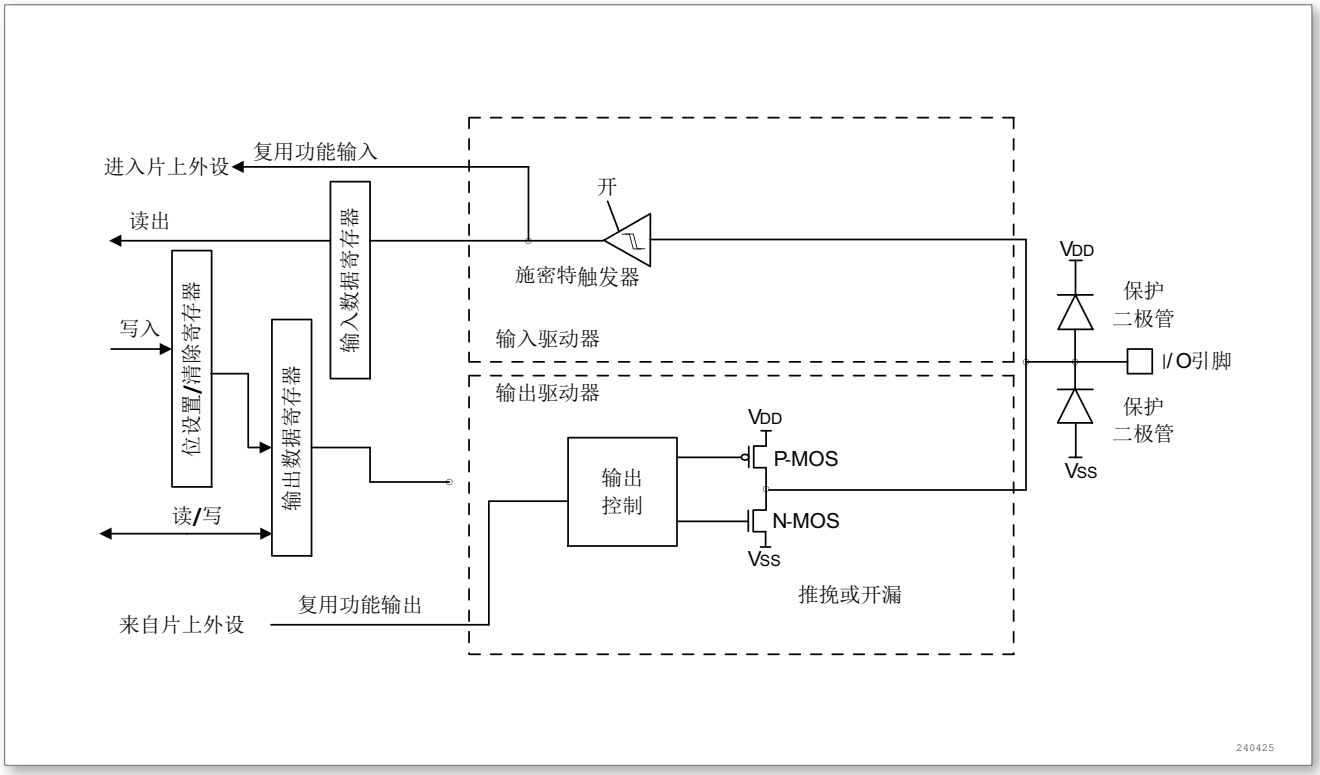


图 17. 复用功能配置

7.1.10 模拟输入配置

当 I/O 端口被配置成模拟输入配置时：

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

下图给出了 I/O 端口位的高阻抗输入配置：

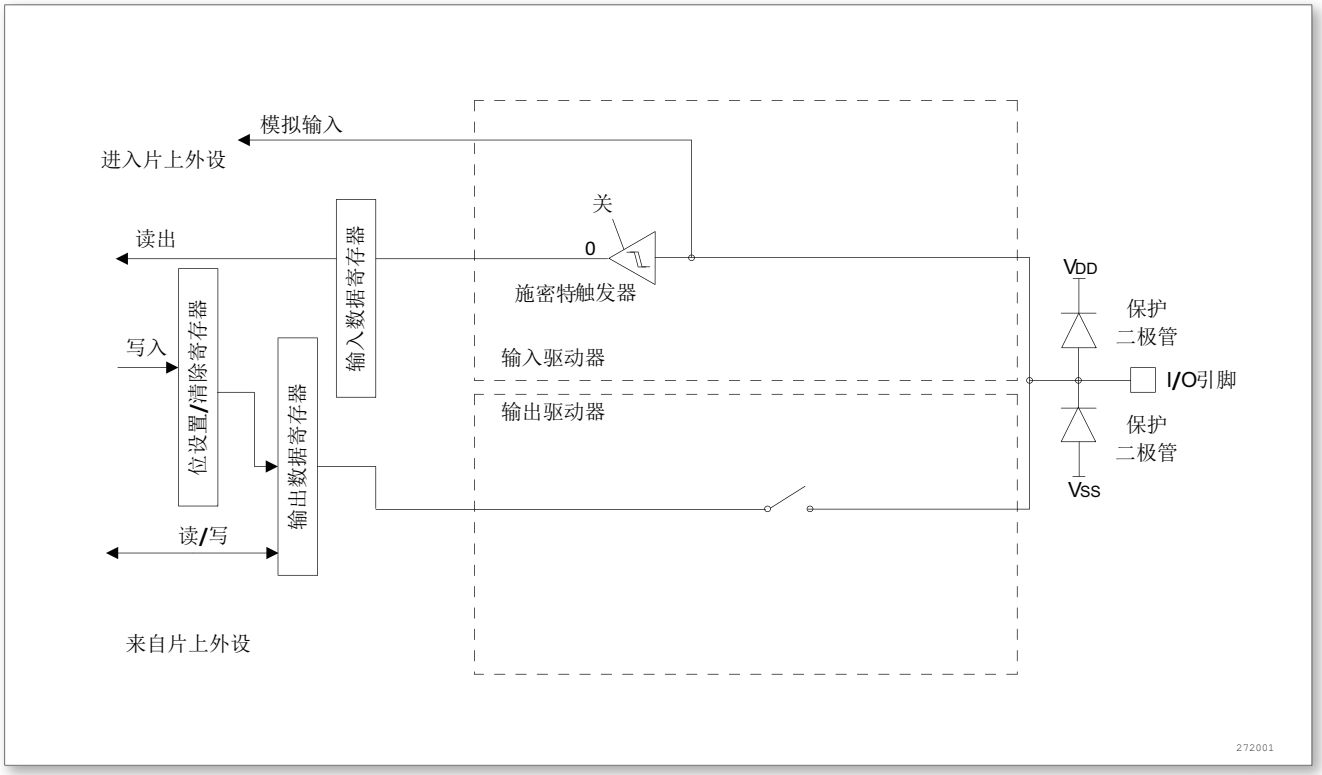


图 18. 高阻抗的模拟输入配置

7.1.11 外设的 GPIO 配置

下列表格列出了各个外设的引脚配置：

表 20. 高级定时器 TIM1

TIM1 引脚	配置	GPIO 配置
TIM1_CHx	输入捕获通道 x	浮空输入
	输出比较通道 x	推挽复用输出
TIM1_CHxN	互补输出通道 x	推挽复用输出
TIM1_BKIN	刹车输入	浮空输入
TIM1_ETR	外部触发时钟输入	浮空输入

表 21. 通用定时器 TIM2/3/4

TIM2/3/4 引脚	配置	GPIO 配置
TIM2/3/4_CHx	输入捕获通道 x	浮空输入
	输出比较通道 x	推挽复用输出
TIM2/3/4_ETR	外部触发时钟输入	浮空输入

表 22. UART

UART 引脚	配置	GPIO 配置
UARTx_TX	串口发送	推挽复用输出

UART 引脚	配置	GPIO 配置
UARTx_RX	串口接收	浮空输入或带上拉输入
UARTx_RTS	硬件流量控制	推挽复用输出
UARTx_CTS	硬件流量控制	浮空输入或带上拉输入

表 23. SPI

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

表 24. I2C

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

表 25. CAN

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

表 26. ADC

ADC 引脚	GPIO 配置
ADC	模拟输入

表 27. 其他 I/O 引脚

引脚	配置	GPIO 配置
TAMPER-RTC	RTC 输出	当配置 BKP_CP 和 BKP_RTCCR 寄存器时，由硬件强制设置
	侵入事件输入	
MCO	时钟输出	推挽复用输出

引脚	配置	GPIO 配置
EXTI 输入线	外部中断输入	浮浮空输入或带上拉输入或下拉输入

7.2 GPIO 寄存器描述

表 28. GPIO 寄存器概览

Offset	Acronym	Register Name	Reset	Section
0x00	GPIOx_CRL	端口配置低寄存器	0xXXXXXXXX	小节 7.2.1
0x04	GPIOx_CRH	端口配置高寄存器	0xXXXXXXXX	小节 7.2.2
0x08	GPIOx_IDR	端口输入数据寄存器	0x0000XXXX	小节 7.2.3
0x0C	GPIOx_ODR	端口输出数据寄存器	0x00000XXX	小节 7.2.4
0x10	GPIOx_BSRR	端口设置/清除寄存器	0x00000000	小节 7.2.5
0x14	GPIOx_BRR	端口位清除寄存器	0x00000000	小节 7.2.6
0x18	GPIOx_LCKR	端口配置锁定寄存器	0x00000000	小节 7.2.7

7.2.1 端口配置低寄存器 (GPIOx_CRL)(x = A..D)

偏移地址: 0x00

复位值: GPIOA_CRL: 0x4444 4848

GPIOB_CRL: 0x4466 6444

GPIOC_CRL: 0x0000 0000

GPIOD_CRL: 0x0000 4444

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

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

Bit	Field	Type	Reset	Description
31: 30 27: 26 23: 22 19: 18 15: 14 11: 10 7: 6 3: 2	CNFy	rw		端口 x 配置位 (0…7)(Port x configuration bits) 软件通过这些位配置相应的 I/O 端口，请参考表 18端口位配置表 在输入模式 (MODE = 00): 00: 模拟输入模式 01: 浮空输入模式 10: 上拉/下拉输入模式 11: 保留 在输出模式 (MODE > 00): 00: 通用推挽输出模式 01: 通用开漏输出模式 10: 复用功能推挽输出模式 11: 复用功能开漏输出模式
29: 28 25: 24 21: 20 17: 16 13: 12 9: 8 5: 4 1: 0	MODEy	rw		端口 x 的模式位 (y = 0…7)(Port x mode bits) 软件通过这些位配置相应的 I/O 端口，请参考表 18端口位配置表 00: 输入模式 (复位后的状态) 01: 输出模式 10: 保留 11: 保留

7.2.2 端口配置高寄存器 (GPIOx_CRH)(x = A..D)

偏移地址: 0x04

复位值: GPIOA_CRH: 0x6444 4484

GPIOB_CRH: 0x4444 8844

GPIOC_CRH: 0x4484 4444

GPIOD_CRH: 0x4444 4444

GPIOE_CRH: 0x4444 4444

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

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

Bit	Field	Type	Reset	Description
31: 30 27: 26 23: 22 19: 18 15: 14 11: 10 7: 6 3: 2	CNFy	rw		端口 x 配置位 (8…15)(Port x configuration bits) 软件通过这些位配置相应的 I/O 端口，请参考表 18端口位配置表 在输入模式 (MODE = 00): 00: 模拟输入模式 01: 浮空输入模式 10: 上拉/下拉输入模式 11: 保留 在输出模式 (MODE[1: 0] > 00): 00: 通用推挽输出模式 01: 通用开漏输出模式 10: 复用功能推挽输出模式 11: 复用功能开漏输出模式
29: 28 25: 24 21: 20 17: 16 13: 12 9: 8 5: 4 1: 0	MODEy	rw		端口 x 的模式位 (y = 8…15)(Port x mode bits) 软件通过这些位配置相应的 I/O 端口，请参考表 18端口位配置表 00: 输入模式 (复位后的状态) 01: 输出模式 10: 保留 11: 保留

7.2.3 端口输入数据寄存器 (GPIOx_IDR)(x = A..D)

偏移地址: 0x08

复位值: 0x0000 XXXX

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

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

Bit	Field	Type	Reset	Description
31: 16	Reserved			始终读为 0。
15: 0	IDRy	r	0xXXXX	端口输入数据 (y = 0..15)(Port input data) 这些位为只读只能以字 (16 位) 的形式读出，读出的值为对应的 I/O 口的状态。

7.2.4 端口输出数据寄存器 (GPIOx_ODR)(x = A..D)

偏移地址: 0x0C

复位值: GPIOA_ODR: 0x0204

GPIOB_ODR: 0x0400

GPIOC_ODR: 0x0000

GPIOD_ODR: 0x0000

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

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

Bit	Field	Type	Reset	Description
31: 16	Reserved			始终读为 0。
15: 0	ODRy	rw		端口输出数据 (y = 0..15)(Port output data) 这些位为可读并只能以字 (16 位) 的形式操作。 注：对 GPIOx_BSRR(x = A…E)，可以分别地对各个 ODR 位进行独立的设置/清除。

7.2.5 端口设置/清除寄存器 (GPIOx_BSRR)(x = A..D)

偏移地址: 0x10

复位值: 0x0000 0000

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

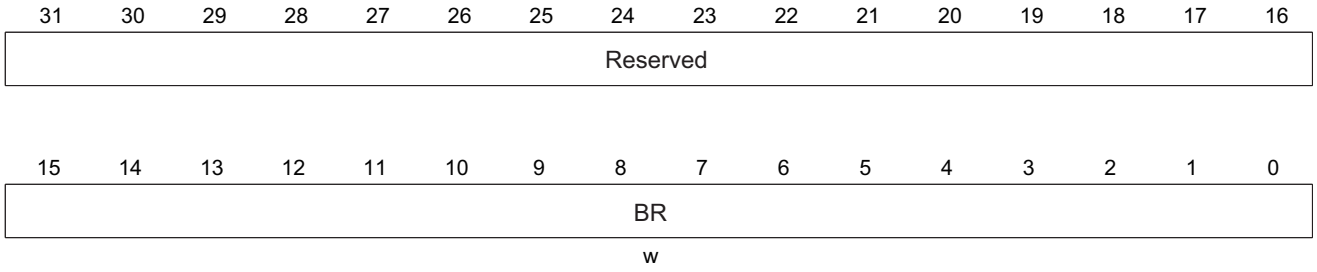
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BS15	BS14	BS13	BS12	BS11	BS10	BS9	BS8	BS7	BS6	BS5	BS4	BS3	BS2	BS1	BS0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bit	Field	Type	Reset	Description
31: 16	BRy	w	0x0000	清除端口 x 的位 y(y = 0…15)(Port x Reset bit y) 这些位只能写入并只能以字 (16 位) 的形式操作。 0: 对对应的 ODRy 位不产生影响 1: 清除对应的 ODRy 位为 0
15: 0	BSy	w	0x0000	设置端口 x 的位 y(y = 0..15)(Port x Set bit y) 这些位只能写入并只能以字 (16 位) 的形式操作。 0: 对应的 ODRy 位不产生影响 1: 设置对应的 ODRy 位为 1

7.2.6 端口位清除寄存器 (GPIOx_BRR)(x = A..D)

偏移地址: 0x14

复位值: 0x0000 0000



Bit	Field	Type	Reset	Description
31: 16	Reserved			始终读为 0
15: 0	BRy	w	0x0000	清除端口 x 的位 y(y = 0…15)(Port x Reset bit y) 这些位只能写入并只能以字 (16 位) 的形式操作。 0: 对应的 ODRy 位不产生影响 1: 清除对应的 ODRy 位为 0

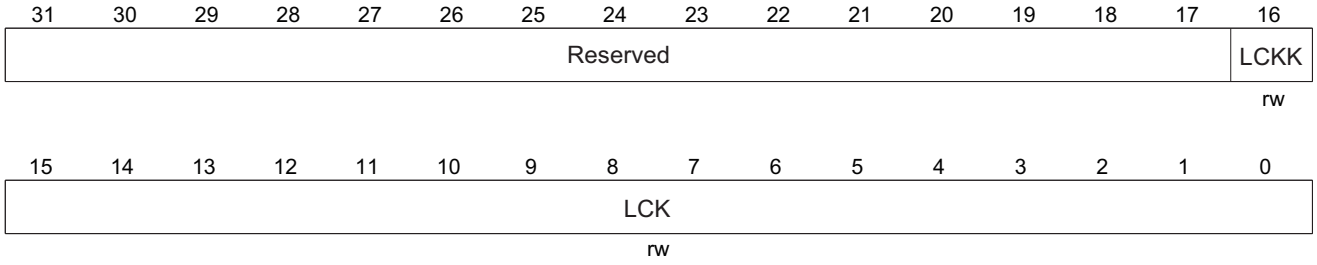
7.2.7 端口配置锁定寄存器 (GPIOx_LCKR)(x = A..D)

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

每个锁定位锁定控制寄存器 (CRL，CRH) 中相应的 4 个位。

地址偏移: 0x18

复位值: 0x0000 0000



Bit	Field	Type	Reset	Description
31: 17	Reserved			始终读为 0
16	LCKK	rw	0x00	锁键 (Lock key) 该位可随时读出，它只可通过锁键写入序列修改。 0: 端口配置锁键位激活 1: 端口配置锁键位被激活，下次系统复位前 GPIOx_LCKR 寄存器被锁住 锁键的写入序列: 写 1-> 写 0-> 写 1-> 读 0-> 读 1 最后一个读可省略，但可以用来确认锁键已被激活。 注：在操作锁键的写入序列时，不能改变 LCK 的值。操作锁键写入序列中的任何错误将不能激活锁键。

Bit	Field	Type	Reset	Description
15: 0	LCKy	rw	0x00	端口 x 的锁位 y(y = 0…15)(Port x Lock bit y) 这些位可读可写但只能在 LCKK 位为 0 时写入。 0: 不锁定端口的配置 1: 锁定端口的配置

7.3 复用功能 I/O 和调试配置（AFIO）

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

7.3.1 把 OSC32_IN/OSC32_OUT 作为 GPIO 端口 PC14/PC15

当 LSE 振荡器关闭时，LSE 振荡器引脚 OSC32_IN/OSC32_OUT 可以分别用作 GPIO 的 PC14/PC15，LSE 功能时钟优先于通用 I/O 的功能。

注：当关闭 1.5V 电压区（进入待机模式）或后备区域使用 VBAT 供电（不再有 VDD 供电）时，不能使用 PC14/PC15 的 GPIO 口功能。

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

外部振荡器引脚 OSC_IN/OSC_OUT 可以用作 GPIO 的 PD0/PD1，需先关闭内部高速时钟，再通过设置重复重映射和调试 I/O 配置寄存器（AFIO_MAPR）实现。注：外部中断/事件功能没有被重映射，在 36、48 和 64 脚的封装上，PD0 和 PD1 不能用来产生外部中断事件。

7.3.3 CAN 复用功能重映射

CAN 信号可以被映射到端口 A 和端口 B，如下表所示。

表 29. CAN 复用功能重映射

复用功能	CAN_REMAP = 00	CAN_REMAP = 10
CAN_RX	PA11	PB8
CAN_TX	PA12	PB9

7.3.4 JTAG/SWD 复用功能重映射

调试接口信号被映射到 GPIO 端口上，如下表所示。

表 30. 调试接口信号

复用功能	GPIO 端口
JTMS/SWDIO	PA13
JTCK/SWCLK	PA14
JTDI	PA15
JTDO/TRACESWO	PB3
JNTRST	PB4

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

表 31. 调试端口映像

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

7.3.5 定时器复用功能重映射

表 32. 定时器 3 复用功能重映像

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

表 33. 定时器 2 复用功能重映像

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

表 34. 定时器 1 复用功能重映像

复用功能映像	TIM1_REMAP =00（没有重映像）	TIM1_REMAP = 01（部分重映像）
TIM1_ETR	PA12	

TIM1_CH1	PA8	
TIM1_CH2	PA9	
TIM1_CH3	PA10	
TIM1_CH4	PA11	
TIM1_BKIN	PB12	PA6
TIM1_CH1N	PB13	PA7
TIM1_CH2N	PB14	PB0
TIM1_CH3N	PB15	PB1

7.3.6 UART 复用功能重映射

参见复用重映射和调试 I/O 配置寄存器 (AFIO_MAPR)

表 35. UART3 重映像

复用功能	**UART3_REMAP = 00（没有重映像）**	**UART3_REMAP = 01（部分重映像）**
UART3_TX	PB10	PC10
UART3_RX	PB11	PC11
UART3_CTS	PB13	
UART3_RTS	PB14	

表 36. UART1 重映像

复用功能	**UART1_REMAP= 0**	**UART1_REMAP= 1**
UART1_TX	PA9	PB6
UART1_RX	PA10	PB7

7.3.7 I2C1 复用功能重映射

参见复用重映射和调试 I/O 配置寄存器 (AFIO_MAPR)

表 37. I2C1 重映像

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

7.3.8 SPI 复用功能重映射

参见复用重映射和调试 I/O 配置寄存器 (AFIO_MAPR)

表 38. SPI1 重映像

复用功能	**SPI1_REMAP= 0**	**SPI1_REMAP= 1**
SPI1_NSS	PA4	PA15

SPI1_SCK	PA5	PB3
SPI1_MISO	PA6	PB4
SPI1_MOSI	PA7	PB5

7.4 AFIO 寄存器描述

表 39. AFIO 寄存器概览

Offset	Acronym	Register Name	Reset	Section
0x04	AFIO_MAPR	复用重映射和调试 I/O 配置寄存器	0x00000000	小节 7.4.1
0x08	AFIO_EXTICR1	外部中断配置寄存器 1	0x00000000	小节 7.4.2
0x0C	AFIO_EXTICR2	外部中断配置寄存器 2	0x00000000	小节 7.4.3
0x10	AFIO_EXTICR3	外部中断配置寄存器 3	0x00000000	小节 7.4.4
0x14	AFIO_EXTICR4	外部中断配置寄存器 4	0x00000000	小节 7.4.5

7.4.1 复用重映射和调试 I/O 配置寄存器 (AFIO_MAPR)

地址偏移: 0x04

复位值: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PD01_ REMAP	CAN_ REMAP		Res.	TIM3_ REMAP		TIM2_ REMAP		TIM1_ REMAP		UART3_ REMAP		Res.	UART1_ REMAP	I2C1_ REMAP	SPI1_ REMAP
rw	rw	rw		rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw

Bit	Field	Type	Reset	Description
31 : 27	Reserved			始终读为 0
26 : 24	SWJ_ CFG	w	0x00	串行线 JTAG 配置（Serial wire JTAG configuration） 这些位可由软件读写，用于配置 SWJ 和跟踪复用功能的 I/O 口。SWJ（串行线 JTAG）支持 JTAG 或 SWD 访问 CPU 的调试端口。系统复位后的默认状态是启用 SWJ 但没有跟踪功能。这种状态下可以通过 JTMS/JTCK 脚上的特定信号选择 JTAG 或 SW（串行线）模式。 000: 完全 SWJ（JTAG-DP + SW-DP）：复位状态 001: 完全 SWJ（JTAG-DP + SW-DP）但没有 JNTRST 010: 关闭 JTAG-DP，启动 SW-DP 100: 关闭 JTAG-DP，关闭 SW-DP 其他组合: 禁用
23 : 16	Reserved			始终读为 0

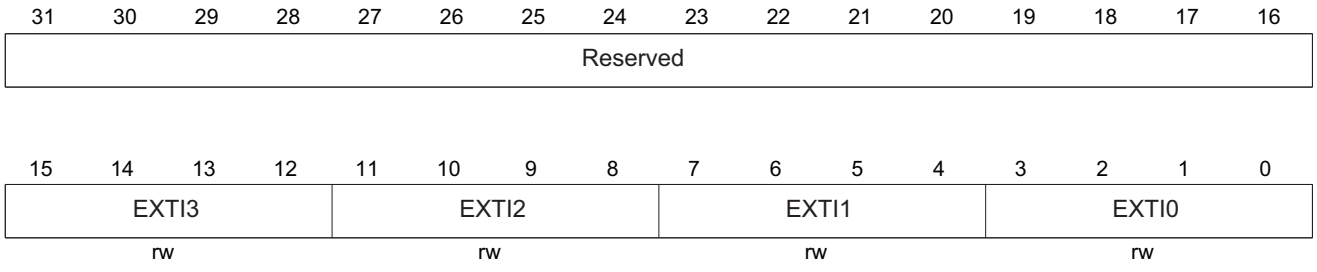
Bit	Field	Type	Reset	Description
15	PD01_ REMAP	rw	0x00	端口 D0/端口 D1 映像到 OSCIN/OSC_OUT（Port D0/Port D1 mapping on OSC_IN/OSC_OUT） 该位可由软件置‘1’或置‘0’。它控制 PD0 和 PD1 的 GPIO 功能映像。当不适用主振荡器 HSE 时（系统运行于内部的 8MHz 阻容振荡器）PD0 和 PD1 可以映像到 OSC_IN 和 OSC_OUT 引脚。此功能只能适用于 36、48 和 64 管脚的封装。 0: 不进行 PD0 和 PD1 的重映像 1: PD0 映像到 OSC_IN，PD1 映像到 OSC_OUT
14：13	CAN_ REMAP	rw	0x00	CAN 复用功能重映像（CAN alternate function remapping） 这些位可由软件置‘1’或置‘0’，控制复用功能 CAN_RX 和 CAN_TX 的重映像。 00: CAN_RX 映像到 PA11，CAN_TX 映像到 PA12 01: 未用组合 10: CAN_RX 映像到 PB8，CAN_TX 映像到 PB9 11: 未用组合
12	Reserved			始终读为 0
11：10	TIM3_ REMAP	rw	0x00	定时器 3 的重映像（TIM3 remapping） 该位可由软件置‘1’或置‘0’，控制 TIM3 的通道 1 4 映射到 GPIO 端口上。 00: 没有重映像（CH1/PA6, CH2/PA7, CH3/PB0, CH4/PB1） 01: 未用组合 10: 部分映像（CH1 /PB4，CH2/PB5，CH3/PB0，CH4/PB1） 11: 完全映像（CH1/PC6，CH2/PC7，CH3/PC8，CH4/PC9） 注: 重映像不影响在 PD2 上的 TIM3_ETR。
9：8	TIM2_ REMAP	rw	0x00	定时器 2 的重映像（TIM2 remapping） 该位可由软件置‘1’或置‘0’，控制 TIM2 的通道 1 4H 和外部触发（ETR）映射到 GPIO 端口上。 00: 没有重映像（CH1/ETR/PA0，CH2/PA1，CH3/PA2，CH4/PA3） 01: 部分映像（CH1/ETR/PA15，CH2/PB3，CH3/PA2，CH4/PA3） 10: 部分映像（CH1/ETR/PA0，CH2/PA1，CH3/PB10，CH4/PB11） 11: 完全映像（CH1/ETR/PA15，CH2/PB3，CH3/PB10，CH4/PB11）

Bit	Field	Type	Reset	Description
7 : 6	TIM1_ REMAP	rw	0x00	定时器 1 的重映像（TTIM1 remapping） 该位可由软件置‘1’或置‘0’，控制 TIM1 的通道 1 4、1N 至 3N、外部触发（ETR）和断线输入（BKIN）映射到 GPIO 端口上。 00: 没有重映像（ETR/PA12, CH1/PA8, CH2/PA9, CH3/PA10, CH4/PA11, BAKIN/PB12, CH1N/PB13, CH2.PB14, CH3N/PB15） 01: 部分映像（ETR/PA12，CH1/PA8，CH2/PA9，CH3/PA10，CH4/PA11，BAKIN/PA6，CH1N/PA7，CH2N/PB0，CH3N/PB1）
5 : 4	UART3_ REMAP	rw	0x00	UART3 的重映像（UART3 remapping） 这些位可由软件置‘1’或置‘0’，控制 UART3 的 RX，TX 复用功能在 GPIO 端口的映像。 00: 没有重映像（TX/PB10，RX/PB11） 01: 部分重映像（TX/PC10，RX/PC11）
3	Reserved		0x00	始终读为 0
2	UART1_ REMAP	rw	0x00	UART1 的重映像（UART1 remapping） 这些位可由软件置‘1’或置‘0’，控制 UART1 的 RX，TX 复用功能在 GPIO 端口的映像。 0: 没有重映像（TX/PA9，RX/PA10） 1: 重映像（TX/PB6，RX/PB7）
1	I2C1_ REMAP	rw	0x00	I2C1 的重映像 这些位可由软件置‘1’或置‘0’，控制 I2C1 的 SCL，SDA 复用功能在 GPIO 端口的映像。 0: 没有重映像（SCL/PB6，SDA/PB7） 1: 重映像（SCL/PB8，SDA/PB9）
0	SPI1_ REMAP	rw	0x00	SPI1 的重映像（SPI1 remapping） 这些位可由软件置‘1’或置‘0’，控制 SPI1 的 SCK，MISO，MOSI 复用功能在 GPIO 端口的映像。 0: 没有重映像（SCK/PA5，MISO/PA6，MOSI/PA7） 1: 重映像（SCK/PB3，MISO/PB4，MOSI/PB5）

7.4.2 外部中断配置寄存器 1(AFIO_EXTICR1)

偏移地址：0x08

复位值：0x0000 0000



Bit	Field	Type	Reset	Description
31：16	Reserved			始终读为 0
15：0	EXTI3， EXTI2， EXTI1， EXTI0	rw	0x00	EXTIx 配置（x = 0…3）（EXTI x configuration） 这些位可用于软件读写。用于选择 EXTIx 外部中断的输入源。 0000：PA[x] 管脚 0001：PB[x] 管脚 0010：PC[x] 管脚 0011：PD[x] 管脚（EXTI0-EXTI2）

7.4.3 外部中断配置寄存器 2(AFIO_EXTICR2)

偏移地址：0x0C

复位值：0x0000 0000

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

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

Bit	Field	Type	Reset	Description
31：16	Reserved			始终读为 0
15：0	EXTI7， EXTI6， EXTI5， EXTI4	rw	0x00	EXTIx 配置（x = 7…4）（EXTI x configuration） 这些位可用于软件读写。用于选择 EXTIx 外部中断的输入源。 0000：PA[x] 管脚 0001：PB[x] 管脚 0010：PC[x] 管脚

7.4.4 外部中断配置寄存器 3(AFIO_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
Reserved															

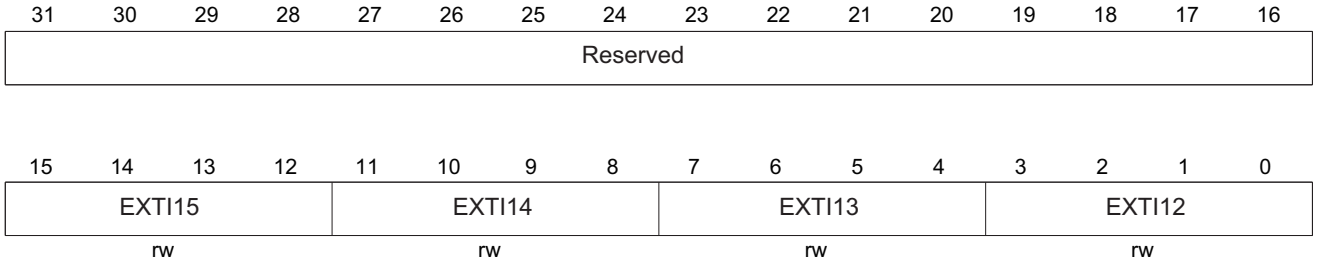
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXTI11				EXTI10				EXTI9				EXTI8			
rw				rw				rw				rw			

Bit	Field	Type	Reset	Description
31：16	Reserved			始终读为 0
15：0	EXTI11，EXTI10，EXTI9，EXTI8	rw	0x00	EXTIx 配置（x = 11…8）（EXTI x configuration） 这些位可用于软件读写。用于选择 EXTIx 外部中断的输入源。 0000：PA[x] 管脚 0001：PB[x] 管脚 0010：PC[x] 管脚

7.4.5 外部中断配置寄存器 1(AFIO_EXTICR4)

偏移地址：0x14

复位值：0x0000 0000



Bit	Field	Type	Reset	Description
31：16	Reserved			始终读为 0
15：0	EXTI15，EXTI14，EXTI13，EXTI12	rw	0x00	EXTIx 配置（x = 15…12）（EXTI x configuration） 这些位可用于软件读写。用于选择 EXTIx 外部中断的输入源。 0000：PA[x] 管脚 0001：PB[x] 管脚 0010：PC[x] 管脚

8 中断和事件 (EXTI)

中断和事件 (EXTI)

8.1 嵌套向量中断控制器

特征

- 中断都可屏蔽 (除了 NMI)
- 8 个可编程的优先等级 (使用了 3 位中断优先级)
- 低延迟的异常和中断处理
- 电源管理控制
- 系统控制寄存器的实现

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

嵌套向量中断控制器管理着包括核异常等中断。关于更多的异常和 NVIC 编程的说明请参考 CPU 技术参考手册。

8.1.1 系统嘀嗒 (SysTick) 校准值寄存器

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

8.1.2 中断和异常向量

下表列出了本系列产品的向量表。

表 40. 本系列产品的向量表

位置	优先级	优先级类型	名称	说明	地址
	-	-	-	保留	0x0000_0000
	-3	固定	Reset	复位	0x0000_0004
	-2	固定	NMI	不可屏蔽中断 RCC 时钟安全系统 (CSS) 联接到 NMI 向量	0x0000_0008
	-1	固定	硬件失效 (HardFault)	所有类型的失效	0x0000_000C
	0	可设置	存储管理 (MemManage)	存储器管理	0x0000_0010
	1	可设置	总线错误 (BusFault)	预取指失败，存储器访问失败	0x0000_0014
	2	可设置	错误应用 (UsageFault)	未定义的指令或非法状态	0x0000_0018
	-	-	-	保留	0x0000_001C ~ 0x0000_002B

位置	优先级	优先级类型	名称	说明	地址
	3	可设置	SVCall	通过 SWI 指令的系统服务调用	0x0000_002C
	4	可设置	调试监控 (DebugMonitor)	调试监控器	0x0000_0030
	-	-	-	保留	0x0000_0034
	5	可设置	PendSV	可挂起的系统服务	0x0000_0038
	6	可设置	SysTick	系统嘀嗒定时器	0x0000_003C
0	7	可设置	WWDG	窗口定时器中断	0x0000_0040
1	8	可设置	PVD	连到 EXTI16 的电源电压检测 (PVD) 中断	0x0000_0044
2	9	可设置	TAMPER	侵入检测中断	0x0000_0048
3	10	可设置	RTC	实时时钟 (RTC) 全局中断	0x0000_004C
4	11	可设置	FLASH	闪存全局中断	0x0000_0050
5	12	可设置	RCC	复位和时钟控制 (RCC) 中断	0x0000_0054
6	13	可设置	EXTI0	EXTI 线 0 中断	0x0000_0058
7	14	可设置	EXTI1	EXTI 线 1 中断	0x0000_005C
8	15	可设置	EXTI2	EXTI 线 2 中断	0x0000_0060
9	16	可设置	EXTI3	EXTI 线 3 中断	0x0000_0064
10	17	可设置	EXTI4	EXTI 线 4 中断	0x0000_0068
11	18	可设置	DMA1 通道 1	DMA1 通道 1 全局中断	0x0000_006C
12	19	可设置	DMA1 通道 2	DMA1 通道 2 全局中断	0x0000_0070
13	20	可设置	DMA1 通道 3	DMA1 通道 3 全局中断	0x0000_0074
14	21	可设置	DMA1 通道 4	DMA1 通道 4 全局中断	0x0000_0078
15	22	可设置	DMA1 通道 5	DMA1 通道 5 全局中断	0x0000_007C
16	23	可设置	DMA1 通道 6	DMA1 通道 6 全局中断	0x0000_0080
17	24	可设置	DMA1 通道 7	DMA1 通道 7 全局中断	0x0000_0084
18	25	可设置	ADC1_2	ADC1 和 ADC2 的全局中断	0x0000_0088
19	26	可设置	USB	USB 中断	0x0000_008C
20	-	-	-	保留	0x0000_0090
21	28	可设置	CAN_RX1	CAN 接收 1 中断	0x0000_0094
22	-	-	-	保留	0x0000_0098
23	30	可设置	EXTI9_5	EXTI 线 [9：5] 中断	0x0000_009C
24	31	可设置	TIM1_BRK	TIM1 断开中断	0x0000_00A0
25	32	可设置	TIM1_UP	TIM1 更新中断	0x0000_00A4
26	33	可设置	TIM1_TRG_COM	TIM1 触发和通信中断	0x0000_00A8
27	34	可设置	TIM1_CC	TIM1 捕获比较中断	0x0000_00AC
28	35	可设置	TIM2	TIM2 全局中断	0x0000_00B0
29	36	可设置	TIM3	TIM3 全局中断	0x0000_00B4
30	37	可设置	TIM4	TIM4 全局中断	0x0000_00B8
31	38	可设置	I2C1_EV	I2C1 事件中断	0x0000_00BC
32	-	-	-	保留	0x0000_00C0
33	40	可设置	I2C2_EV	I2C2 事件中断	0x0000_00C4

位置	优先级	优先级类型	名称	说明	地址
34	-	-	-	保留	0x0000_00C8
35	42	可设置	SPI1	SPI1 全局中断	0x0000_00CC
36	43	可设置	SPI2	SPI2 全局中断	0x0000_00D0
37	44	可设置	UART1	UART1 全局中断	0x0000_00D4
38	45	可设置	UART2	UART2 全局中断	0x0000_00D8
39	46	可设置	UART3	UART3 全局中断	0x0000_00DC
40	47	可设置	EXTI15_10	EXTI 线 [15：10] 中断	0x0000_00E0
41	48	可设置	RTCAlarm	连到 EXTI17 的 RTC 闹钟中断	0x0000_00E4
42	49	可设置	USB 唤醒	连到 EXTI18 的从 USB 待机唤醒中断	0x0000_00E8
43	-	-	Reset	保留	0x0000_00EC
44	-	-	Reset	保留	0x0000_00F0
45	-	-	Reset	保留	0x0000_00F4

8.2 外部中断/事件控制器 (EXTI)

外部中断和事件控制器 (EXTI) 管理外部和内部异步事件/中断，并生成相应的事件请求到 CPU/中断控制器和到电源管理的唤醒请求。

能产生事件/中断请求的边沿检测器。每个输入线可以独立地配置输入类型 (脉冲或挂起) 和对应的触发事件 (上升沿或下降沿或者双边沿都触发)。每个输入线都可以独立地被屏蔽。挂起寄存器保持着状态线的中断请求。

8.2.1 主要特征

EXTI 控制器的主要特性如下：

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

8.2.2 框图

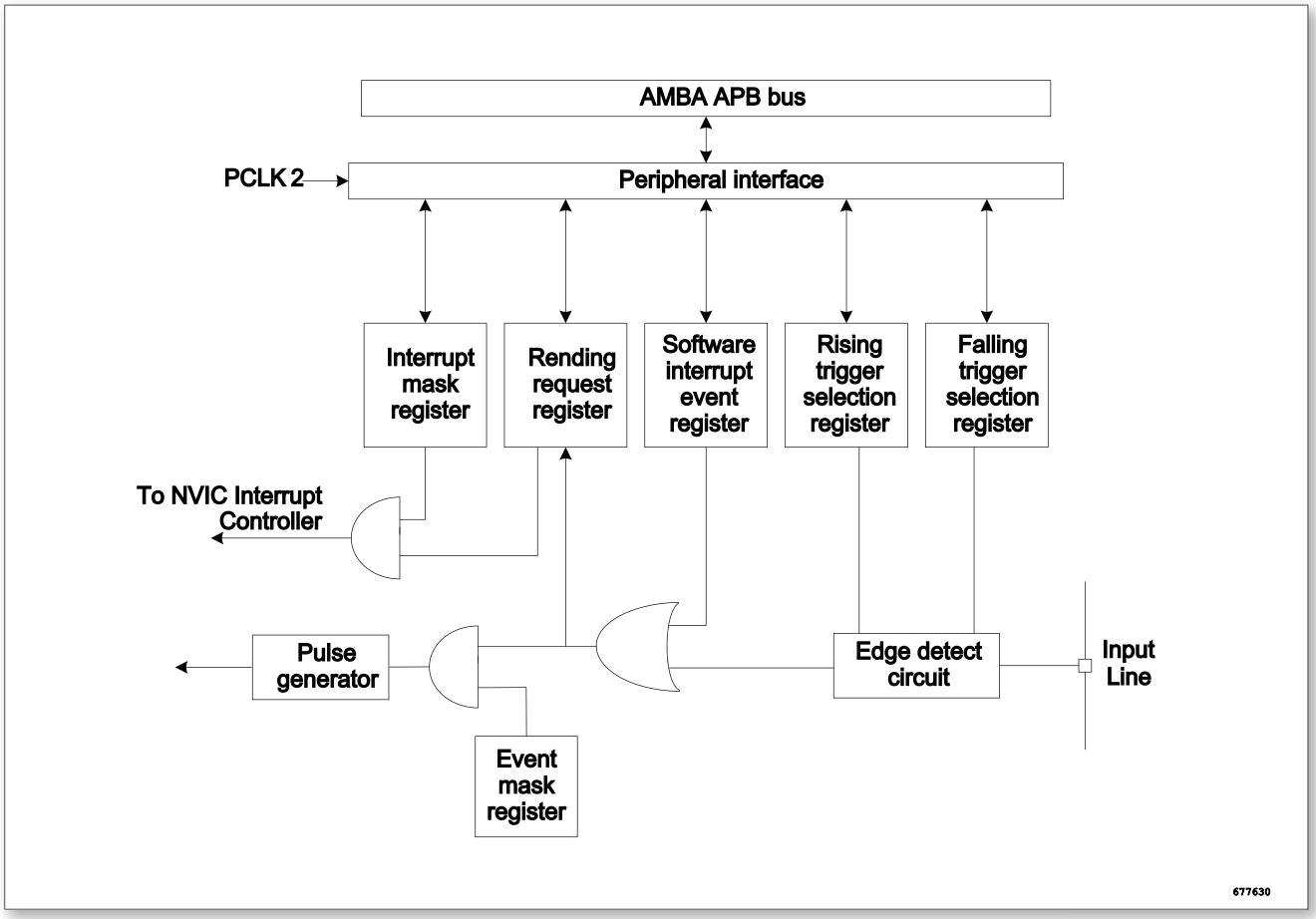


图 19. 外部中断/事件控制器框图

8.2.3 唤醒事件管理

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

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

使用外部 I/O 端口作为唤醒事件，请参见下节的功能说明。

8.2.4 功能说明

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

如果需要产生事件，必须先配置好并使能事件线。根据需要的边沿检测通过设置 2 个触发寄存器，同时在事件屏蔽寄存器的相应位写‘1’允许事件请求。当事件线上发生了需要的

边沿时，将产生一个事件请求脉冲，对应的挂起位不被置‘1’。

通过在软件中断/事件寄存器写‘1’，也可以通过软件产生中断/事件请求。

硬件中断选择

通过下面的过程来配置多个线路做为中断源：

- 配置中断线的屏蔽位 (EXTI_IMR)
- 配置所选中断线的触发选择位 (EXTI_RTSR 和 EXTI_FTSR)
- 配置对应到外部中断控制器 (EXTI) 的 NVIC 中断通道的使能和屏蔽位，使得中断线中的请求可以被正确地响应

硬件事件选择

通过下面的过程，可以配置多个线路为事件源：

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

软件中断/事件的选择

多个线路可以被配置成软件中断/事件线。下面是产生软件中断的过程：

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

8.2.5 外部中断/事件线路映像

通用 I/O 端口以下图的方式连接到 16 个外部中断/事件线上：

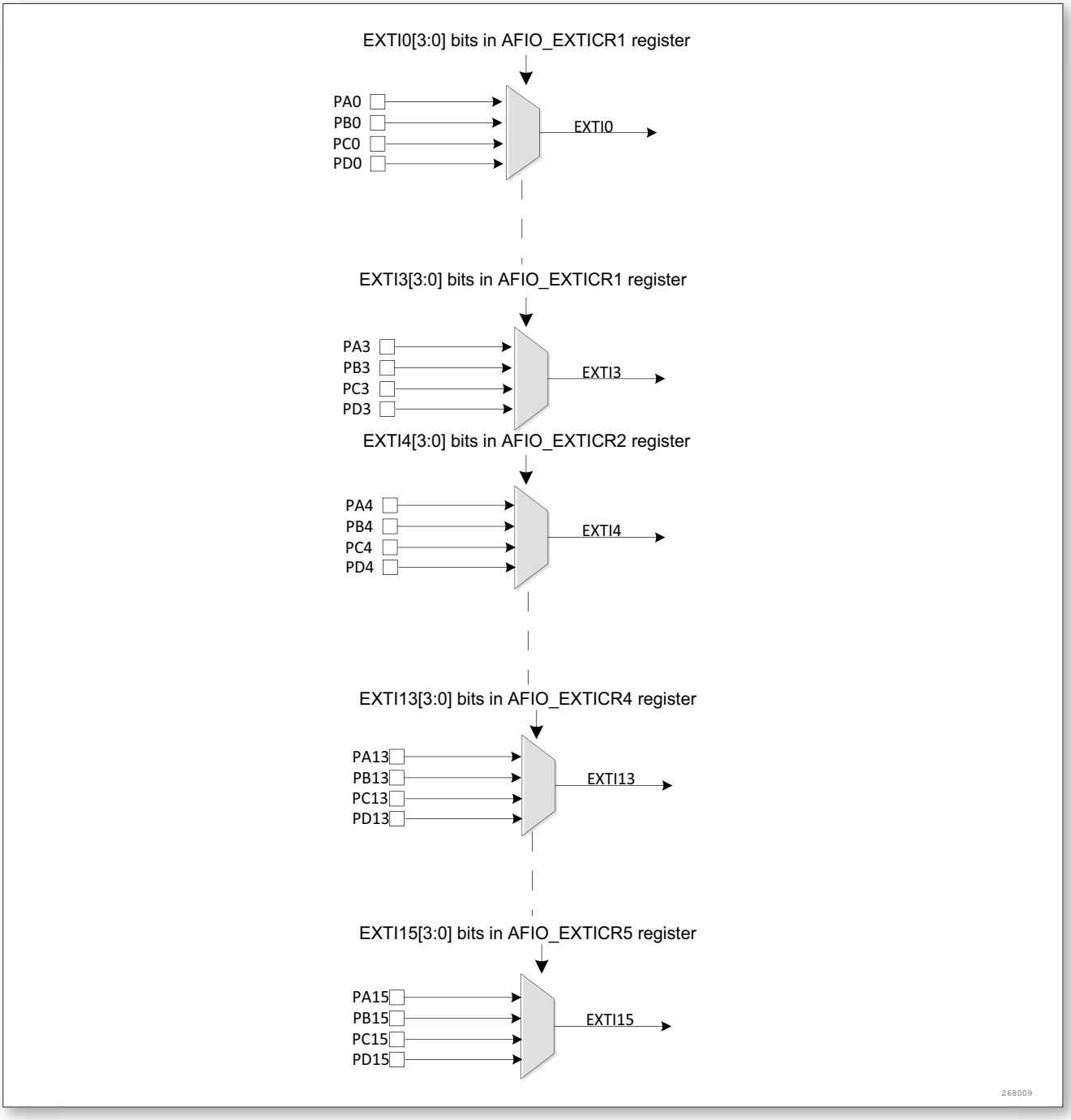


图 20. 外部中断通用 I/O 映像

注：上图对应的 GPIO 可能因实际芯片封装出现差异，以实际芯片为准。

另外其他的外部中断/事件控制器的连接如下：

- EXTI 线 16 连接到 PVD 输出
- EXTI 线 17 连接到 RTC 闹钟事件
- EXTI 线 18 连接到 USB 唤醒事件

8.3 EXTI 寄存器描述

表 41. EXTI 寄存器概览

Offset	Acronym	Register Name	Reset	Section
0x00	EXTI_IMR	中断屏蔽寄存器	0x00000000	小节 8.3.1
0x04	EXTI_EMR	事件屏蔽寄存器	0x00000000	小节 8.3.2
0x08	EXTI_RTSR	上升沿触发选择寄存器	0x00000000	小节 8.3.3
0x0C	EXTI_FTSR	下降沿触发选择寄存	0x00000000	小节 8.3.4
0x10	EXTI_SWIER	软件中断事件寄存器	0x00000000	小节 8.3.5
0x14	EXTI_PR	软件中断事件寄存	0x00000000	小节 8.3.6

8.3.1 中断屏蔽寄存器 (EXTI_IMR)

偏移地址：0x00

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved													IMR18	IMR17	IMR16
													rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IMR15	IMR14	IMR13	IMR12	IMR11	IMR10	IMR9	IMR8	IMR7	IMR6	IMR5	IMR4	IMR3	IMR2	IMR1	IMR0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
31：19	Reserved			始终读为 0。
18：0	IMRx	rw	0x00	线 x 上的中断屏蔽 (Interrupt Mask on line x) 1 = 开放来自线 x 上的中断请求 0 = 屏蔽来自线 x 上的中断请求

8.3.2 事件屏蔽寄存器 (EXTI_EMR)

偏移地址：0x04

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved													EMR18	EMR17	EMR16
													rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EMR15	EMR14	EMR13	EMR12	EMR11	EMR10	EMR9	EMR8	EMR7	EMR6	EMR5	EMR4	EMR3	EMR2	EMR1	EMR0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
31：19	Reserved			始终读为 0。

Bit	Field	Type	Reset	Description
18：0	EMRx	rw	0x00	线 x 上的事件屏蔽 (Event Mask on line x) 1 = 开放来自线 x 上的事件请求 0 = 屏蔽来自线 x 上的事件请求

8.3.3 上升沿触发选择寄存器 (EXTI_RTSR)

偏移地址：0x08

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved													TR18	TR17	TR16
													rw	rw	rw

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

Bit	Field	Type	Reset	Description
31：19	Reserved			始终读为 0。
18：0	TRx	rw	0x00	线 x 上的上升沿触发事件配置位 (Rising trigger event configuration bit of line x) 1 = 允许输入线 x 上的上升沿触发 (中断和事件) 0 = 禁止输入线 x 上的上升沿触发 (中断和事件)

8.3.4 下降沿触发选择寄存器 (EXTI_FTSR)

偏移地址：0x0C

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved													TR18	TR17	TR16
													rw	rw	rw

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

Bit	Field	Type	Reset	Description
31：19	Reserved			始终读为 0。
18：0	TRx	rw	0x00	线 x 上的下降沿触发事件配置位 (Falling trigger event configuration bit of line x) 1 = 允许输入线 x 上的下降沿触发 (中断和事件) 0 = 禁止输入线 x 上的下降沿触发 (中断和事件)

8.3.5 软件中断事件寄存器 (EXTI_SWIER)

偏移地址：0x10

复位值：0x0000 0000

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

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

Bit	Field	Type	Reset	Description
31：19	Reserved			始终读为 0。
18：0	SWIERx	rw	0x00	线 x 上的软件中断 (Software interrupt on line x) 当该位为‘0’时，写‘1’将设置 EXTI_PR 中相应的挂起位。如果在 EXTI_INTMASK 和 EXTI_EVNTMASK 中允许产生该中断，则此时将产生一个中断。 注：通过清除 EXTI_PEND 的对应位 (写入‘1’)，可以清除该位为‘0’。

8.3.6 软件中断事件寄存器 (EXTI_PR)

偏移地址：0x14

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved													PR18	PR17	PR16
													rc_w1	rc_w1	rc_w1

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PR15	PR14	PR13	PR12	PR11	PR10	PR9	PR8	PR7	PR6	PR5	PR4	PR3	PR2	PR1	PR0
rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

Bit	Field	Type	Reset	Description
31：19	Reserved			始终读为 0。
18：0	PRx	rc_w1	0x00	挂起位 (Pending bit) 1 = 发生了选择的触发请求 0 = 没有发生触发请求 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。

9 DMA 控制器 (DMA)

DMA 控制器 (DMA)

9.1 DMA 简介

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

DMA 控制器有 7 个通道，每个通道专门管理多个外设请求。

9.2 DMA 主要特征

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

下面为功能框图：

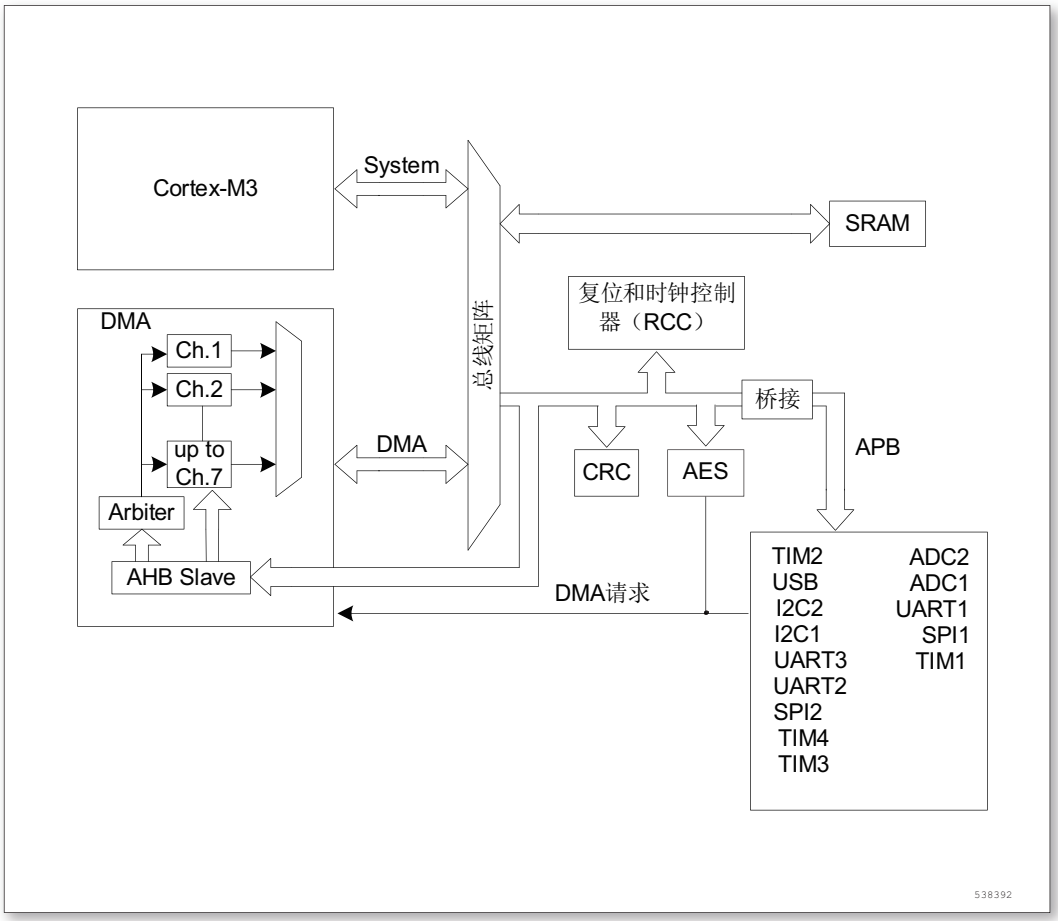


图 21. DMA 框图

9.3 功能描述

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

9.3.1 DMA 处理

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

总之，每个 DMA 传送由 3 个操作组成：

1. 从 DMA_CPARx 寄存器或者 DMA_CMARx 寄存器指定地址的存储器单元执行加载操作。
2. 存数据到 DMA_CPARx 寄存器或者 DMA_CMARx 寄存器指定地址的存储器单元。
3. 执行一次 DMA_CNDTRx 寄存器的递减操作。该寄存器包含未完成的操作数目。

9.3.2 仲裁器

仲裁器根据通道请求的优先级来启动外设/存储器的访问。优先权管理分 2 个阶段：

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

9.3.3 DMA 通道

每个通道都可以在有固定地址的外设寄存器和存储器地址之间执行 DMA 传输。DMA 传输的数据量是可编程的，最大达到 65536。包含要传输的数据项数量的寄存器，在每次传输后递减。

可编程数据量

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

指针增量

通过设置 DMA_CCRx 寄存器中 PINC 和 MINC 标志位，外设和存储器的指针在每次传输后可以有选择地完成自动增量。当设置为增量模式时，下一个要传输的地址将是前一个地址加上增量值，增量值取决与所选的数据宽度为 1、2 或 4。第一个传输的地址存放在 DMA_CPARx / DMA_CMARx 寄存器中。通道配置为非循环模式时，传输结束后 (即传输计数变为 0) 将不再产生 DMA 操作。

通道配置

下面是配置 DMA 通道 x 的过程 (x 代表通道号)：

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

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

循环模式

循环模式用于处理循环缓冲区和连续的数据传输 (如 ADC 的扫描模式)。在 DMA_CCRx 寄

存器中的 CIRC 位用于开启这一功能。当启动了循环模式，数据传输的数目变为 0 时，将会自动地被恢复成配置通道时设置的初值，DMA 操作将会继续进行。

存储器到存储器模式

DMA 通道的操作可以在没有外设请求的情况下进行，这种操作就是存储器到存储器模式。当设置了 DMA_CCRx 寄存器中的 MEM2MEM 位之后，在软件设置了 DMA_CCRx 寄存器中的 EN 位启动 DMA 通道时，DMA 传输将马上开始。当 DMA_CNDTRx 寄存器变为 0 时，DMA 传输结束。存储器到存储器模式不能与循环模式同时使用。

9.3.4 可编程的数据传输宽度，对齐方式和数据大小端

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

表 42. 可编程的数据传输宽度和大小端操作 (当 PINC = MINC = 1)

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

源端宽度	目标宽度	传输数目	源 (地址 / 数据)	传输操作	目标 (地址 / 数据)
32	16	4	0x0/B3B2B1B0 0x4/B7B6B5B4 0x8/BBBAB9B8 0xC/BFBEBDBC	1: 在 0x0 读 B3B2B1B0[31:0]，在 0x0 写 B1B0[15:0] 2: 在 0x4 读 B7B6B5B4[31:0]，在 0x2 写 B5B4[15:0] 3: 在 0x8 读 BBBAB9B8[31:0]，在 0x4 写 B9B8[15:0] 4: 在 0xC 读 BFBEBDBC[31:0]，在 0x6 写 BDBC[15:0]	0x0/B1B0 0x2/B5B4 0x4/B9B8 0x6/BDBC
32	32	4	0x0/B3B2B1B0 0x4/B7B6B5B4 0x8/BBBAB9B8 0xC/BFBEBDBC	1: 在 0x0 读 B3B2B1B0[31:0]，在 0x0 写 B3B2B1-B0[31:0] 2: 在 0x4 读 B7B6B5B4[31:0]，在 0x4 写 B7B6B5-B4[31:0] 3: 在 0x8 读 BBBAB9B8[31:0]，在 0x8 写 BBBAB9-B8[31:0] 4: 在 0xC 读 BFBEBDBC[31:0]，在 0xC 写 BFBEBD-BC[31:0]	0x0/B3B2B1B0 0x4/B7B6B5B4 0x8/BBBAB9B8 0xC/BFBEBDBC

操作一个不支持字节或半字写的 AHB 设备

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

- 当 HSIZE 为半字时，写入半字‘0xABCD’，DMA 将设置 HWDATA 总线为‘0xABCDABCD’。
- 当 HSIZE 为字节时，写入字节‘0xAB’，DMA 将设置 HWDATA 总线为‘0xABABABAB’。

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

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

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

9.3.5 错误管理

读写一个保留的地址区域，将会产生 DMA 传输错误。当在 DMA 读写操作时发生 DMA 传输错误时，硬件会自动地清除发生错误的通道所对应的通道配置寄存器 (DMA_CCRx) 的 EN 位，该通道操作被停止。此时，在 DMA_IFT 寄存器中对应该通道的传输错误中断标志位 (TEIF) 将被置位，如果在 DMA_CCRx 寄存器中设置了传输错误中断允许位，则将产生中断。

9.3.6 中断

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

表 43. DMA 中断请求

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

9.3.7 DMA 请求映像

DMA 控制器

从外设 TIMx、ADCx、USB、SPIx、I2Cx 和 UARTx 产生的 7 个请求，通过逻辑或输入到 DMA 控制器，这意味着同时只能有一个请求有效。参见下图的 DMA 请求映像。

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

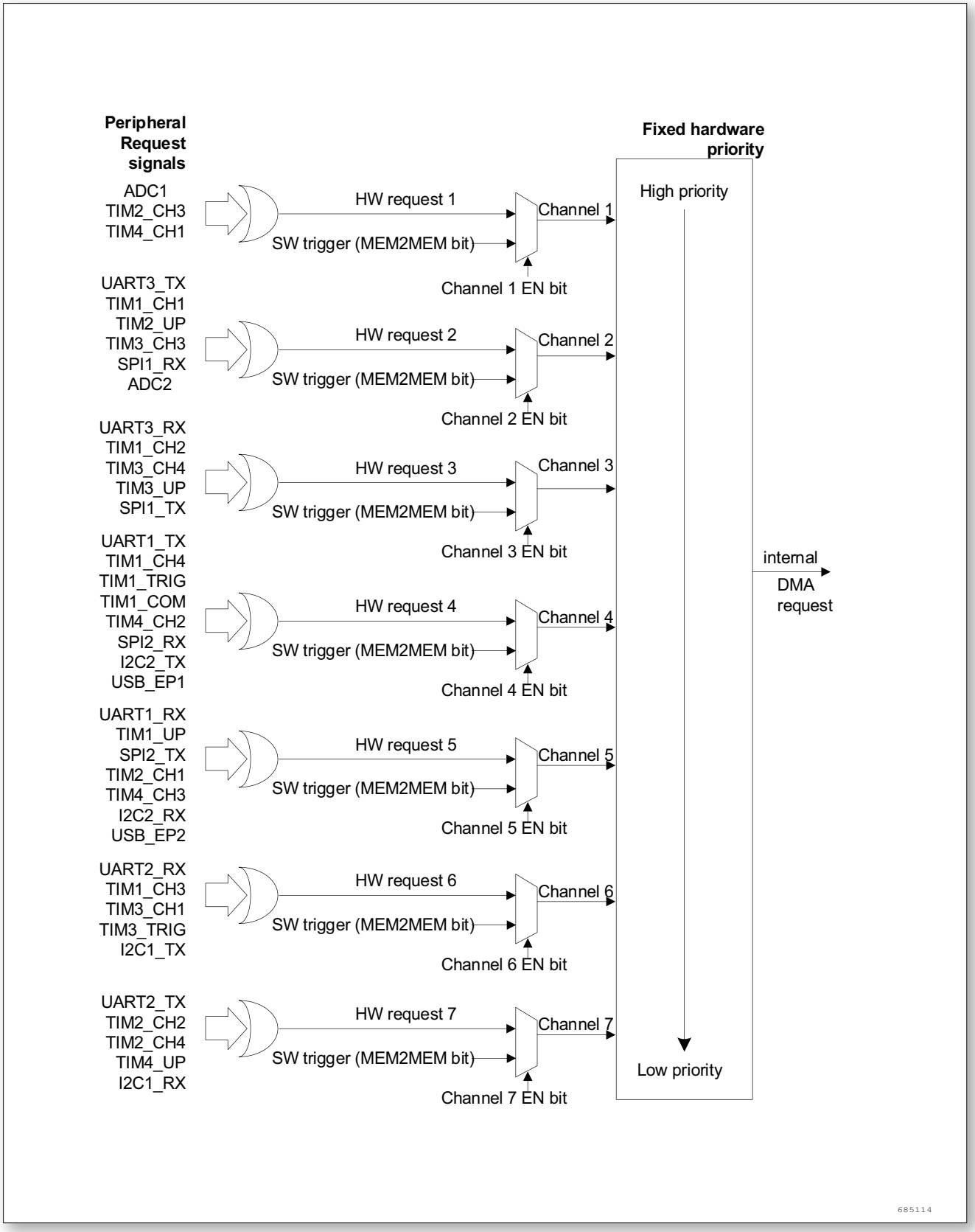


图 22. 外设 DMA 请求映射

表 44. 各个通道的 DMA 请求一览

外设	通道 1	通道 2	通道 3	通道 4	通道 5	通道 6	通道 7
ADC	ADC1	ADC2					
SPI		SPI1_RX	SPI1_TX	SPI2_RX	SPI2_TX		
UART		UART3_TX	UART3_RX	UART1_TX	UART1_RX	UART2_RX	UART2_TX
I2C				I2C2_TX	I2C2_RX	I2C1_TX	I2C1_RX
TIM1		TIM1_CH1	TIM1_CH2	TIM1_CH4 TIM1_TRIG TIM1_COM	TIM1_UP	TIM1_CH3	
TIM2	TIM2_CH3	TIM2_UP			TIM2_CH1		TIM2_CH2 TIM2_CH4
TIM3		TIM3_CH3	TIM3_CH4 TIM3_UP			TIM3_CH1 TIM3_TRIG	
TIM4	TIM4_CH1			TIM4_CH2	TIM4_CH3		TIM4_UP
USB				USB_EP1	USB_EP2		

9.4 DMA 寄存器描述

表 45. DMA 寄存器概览

Offset	Acronym	Register Name	Reset	Section
0x00	DMA_ISR	DMA 中断状态寄存器	0x00000000	小节 9.4.1
0x04	DMA_IFCR	DMA 中断标志清除寄存器	0x00000000	小节 9.4.2
0x08 + 20 × (n - 1)	DMA_CCRx	DMA 通道 x 配置寄存器	0x00000000	小节 9.4.3
0x0C + 20 × (n - 1)	DMA_CNDTRx	DMA 通道 x 传输数量寄存器	0x00000000	小节 9.4.4
0x10 + 20 × (n - 1)	DMA_CPARx	DMA 通道 x 外设地址寄存器	0x00000000	小节 9.4.5
0x14 + 20 × (n - 1)	DMA_CMARx	DMA 通道 x 存储器地址寄存器	0x00000000	小节 9.4.6

9.4.1 DMA 中断状态寄存器 (DMA_ISR)

偏移地址：0x00

复位值：0x0000 0000

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

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

Bit	Field	Type	Reset	Description
31 : 28	Reserved			保留，始终读为 0。

Bit	Field	Type	Reset	Description
27, 23, 19, 15, 11, 7, 3	TEIFx	r	0x00	通道 x 的传输错误标志 (x = 1 …7)(Channel x transfer error flag) 硬件设置这些位。在 DMA_IFCR 寄存器的相应位写入‘1’可以清除这里对应的标志位。 0：在通道 x 没有传输错误 (TE) 1：在通道 x 发生了传输错误 (TE)
26, 22, 18, 14, 10, 6, 2	HTIFx	r	0x00	通道 x 的半传输标志 (x = 1 …7)(Channel x half transfer flag) 硬件设置这些位。在 DMA_IFCR 寄存器的相应位写入‘1’可以清除这里对应的标志位。 0：在通道 x 没有半传输事件 (HT) 1：在通道 x 产生了半传输事件 (HT)
25, 21, 17, 13, 9, 5, 1	TCIFx	r	0x00	通道 x 的传输完成标志 (x = 1 …7)(Channel x transfer complete flag) 硬件设置这些位。在 DMA_IFCR 寄存器的相应位写入‘1’可以清除这里对应的标志位。 0：在通道 x 没有传输完成事件 (TC) 1：在通道 x 产生了传输完成事件 (TC)
24, 20, 16, 12, 8, 4, 0	GIFx	r	0x00	通道 x 的全局中断标志 (x = 1 …7)(Channel x global interrupt flag) 硬件设置这些位。在 DMA_IFCR 寄存器的相应位写入‘1’可以清除这里对应的标志位。 0：在通道 x 没有 TE、HT 或 TC 事件 1：在通道 x 产生了 TE、HT 或 TC 事件

9.4.2 DMA 中断标志清除寄存器 (DMA_IFCR)

偏移地址：0x04

复位值：0x0000 0000

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

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

Bit	Field	Type	Reset	Description
31：28	Reserved			保留，始终读为 0。

Bit	Field	Type	Reset	Description
27, 23, 19, 15, 11, 7, 3	CTEIFx	w	0x00	清除通道 x 的传输错误标志 (x = 1 …7)(Channel x transfer error clear) 这些位由软件设置和清除。 0：不起作用 1：清除 DMA_ISR 寄存器中的对应 TEIF 标志
26, 22, 18, 14, 10, 6, 2	CHTIFx	w	0x00	清除通道 x 的半传输标志 (x = 1 …7)(Channel x half transfer clear) 这些位由软件设置和清除。 0：不起作用 1：清除 DMA_ISR 寄存器中的对应 HTIF 标志
25, 21, 17, 13, 9, 5, 1	CTCIFx	w	0x00	清除通道 x 的传输完成标志 (x = 1 …7)(Channel x transfer complete clear) 这些位由软件设置和清除。 0：不起作用 1：清除 DMA_ISR 寄存器中的对应 TCIF 标志
24, 20, 16, 12, 8, 4, 0	CGIFx	w	0x00	清除通道 x 的全局中断标志 (x = 1 …7)(Channel x global interrupt clear) 这些位由软件设置和清除。 0：不起作用 1：清除 DMA_ISR 寄存器中的对应的 GIF、TEIF、HTIF 和 TCIF 标志

9.4.3 DMA 通道 x 配置寄存器 (DMA_CCRx) (x = 1…7)

偏移地址：0x08 + 20 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
Reserved															

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

Bit	Field	Type	Reset	Description
31：15	Reserved			保留，始终读为 0。
14	MEM2MEM	rw	0x00	存储器到存储器模式 (Memory to memory mode) 该位由软件设置和清除。 0：非存储器到存储器模式 1：启动存储器到存储器模式

Bit	Field	Type	Reset	Description
13：12	PL	rw	0x00	通道优先级 (Channel priority level) 这些位由软件设置和清除。 00：低 01：中 10：高 11：最高
11：10	MSIZE	rw	0x00	存储器数据宽度 (Memory size) 这些位由软件设置和清除。 00：8 位 01：16 位 10：32 位 11：保留
9：8	PSIZE	rw	0x00	外设数据宽度 (Peripheral size) 这些位由软件设置和清除。 00：8 位 01：16 位 10：32 位 11：保留
7	MINC	rw	0x00	存储器地址增量模式 (Memory increment mode) 该位由软件设置和清除。 0：不执行存储器地址增量操作 1：执行存储器地址增量操作
6	PINC	rw	0x00	外设地址增量模式 (Peripheral increment mode) 该位由软件设置和清除。 0：不执行外设地址增量操作 1：执行外设地址增量操作
5	CIRC	rw	0x00	循环模式 (Circular mode) 该位由软件设置和清除。 0：不执行循环操作 1：执行循环操作
4	DIR	rw	0x00	数据传输方向 (Data transfer direction) 该位由软件设置和清除。 0：从外设读 1：从存储器读
3	TEIE	rw	0x00	允许传输错误中断 (Transfer error interrupt enable) 该位由软件设置和清除。 0：禁止 TE 中断 1：允许 TE 中断
2	HTIE	rw	0x00	允许半传输中断 (Half transfer interrupt enable) 该位由软件设置和清除。 0：禁止 HT 中断 1：允许 HT 中断

Bit	Field	Type	Reset	Description
1	TCIE	rw	0x00	允许传输完成中断 (Transfer complete interrupt enable) 该位由软件设置和清除。 0：禁止 TC 中断 1：允许 TC 中断
0	EN	rw	0x00	通道开启 (Channel enable) 该位由软件设置和清除。 0：通道不工作 1：通道开启

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

偏移地址：0x0C + 20 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
Reserved															

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

Bit	Field	Type	Reset	Description
31：16	Reserved			保留，始终读为 0。
15：0	NDT	rw	0x0000	数据传输数量 (Number of data to transfer) 数据传输数量为 0 ~ 65535。这个寄存器只能在通道不工作 (DMA_CCRx 的 EN = 0) 时写入。通道开启后该寄存器变为只读，指示剩余的待传输的字节数目。寄存器内容在每次 DMA 传输后递减。数据传输结束后，寄存器的内容或者变为 0；或者当该通道配置为自动重加载模式时，寄存器的内容将被自动重新加载为之前配置时的数值。 当寄存器的内容为 0 时，无论通道是否开启，都不会发生任何数据传输。

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

偏移地址：0x10 + 20 x (通道编号 - 1)

复位值：0x0000 0000

当开启通道 (DMA_CCRx 的 EN = 1) 时不能写该寄存器。

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

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

Bit	Field	Type	Reset	Description
31：0	PA	rw	0x0000 0000	外设地址 (Peripheral address) 外设数据寄存器的基地址，作为数据传输的源或目标。 当 PSIZE = ‘01’(16 位)，不使用 PA[0] 位。操作自动地与半字地址对齐。 当 PSIZE = ‘10’(32 位)，不使用 PA[1：0] 位。操作自动地与字地址对齐。

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

偏移地址：0x14 + 20 x (通道编号 - 1)

复位值：0x0000 0000

当开启通道 (DMA_CCRx 的 EN = 1) 时不能写该寄存器。

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

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

Bit	Field	Type	Reset	Description
31：0	MA	rw	0x0000 0000	存储器地址 (Memory address) 存储器地址作为数据传输的源或目标。 当 MSIZE = ‘01’(16 位)，不使用 MA[0] 位。操作自动地与半字地址对齐。 当 MSIZE = ‘10’(32 位)，不使用 MA[1：0] 位。操作自动地与字地址对齐。

10 模拟/数字转换 (ADC)

模拟/数字转换 (ADC)

10.1 ADC 介绍

12 位 ADC 是逐次逼近式的模拟一数字转换器 (SAR A/D 转换器)。

A/D 转换器支持多种工作模式:单次转换和连续转换模式,并且可以选择通道自动扫描。A/D 转换的启动方式有软件设定、外部引脚触发以及各个定时器启动。

窗口比较器 (模拟看门狗) 允许应用程序检测输入电压是否超出了用户设定的高/低阈值值。

ADC 的输入时钟不得超过 15MHz，它是由 PCLK2 经分频产生。

10.2 ADC 主要特征

- 最高 12 位可编程分辨率的 SAR ADC，多达 16 路 (8 × 2) 外部输入通道，2 路内部通道 (ADC1 中用来测试温度，ADC2 中用来测试内部参考电压)
- ADC1 的内部信号源接内部温度传感器，ADC2 的内部信号源接内部 1.2V 参考电压
- 高达 1Msps 转换速率
- 支持多种工作模式：
  - 单次转换模式：A/D 转换在指定通道完成一次转换
  - 单周期扫描模式：A/D 转换在所有指定通道完成一个周期 (从低序号通道到高序号通道) 转换
  - 连续扫描模式：A/D 转换连续执行单周期扫描模式直到软件停止 A/D 转换
- 通道采样时间，分辨率可软件配置
- 支持 DMA 传输
- A/D 转换开始条件：
  - 软件启动
  - 外部触发启动
  - Timer 匹配
- 模拟看门狗，转换结果可和指定的值相比较，当转换值和设定值相匹配时，用户可设定是否产生中断请求

10.3 ADC 功能描述

下图显示了 ADC 框图

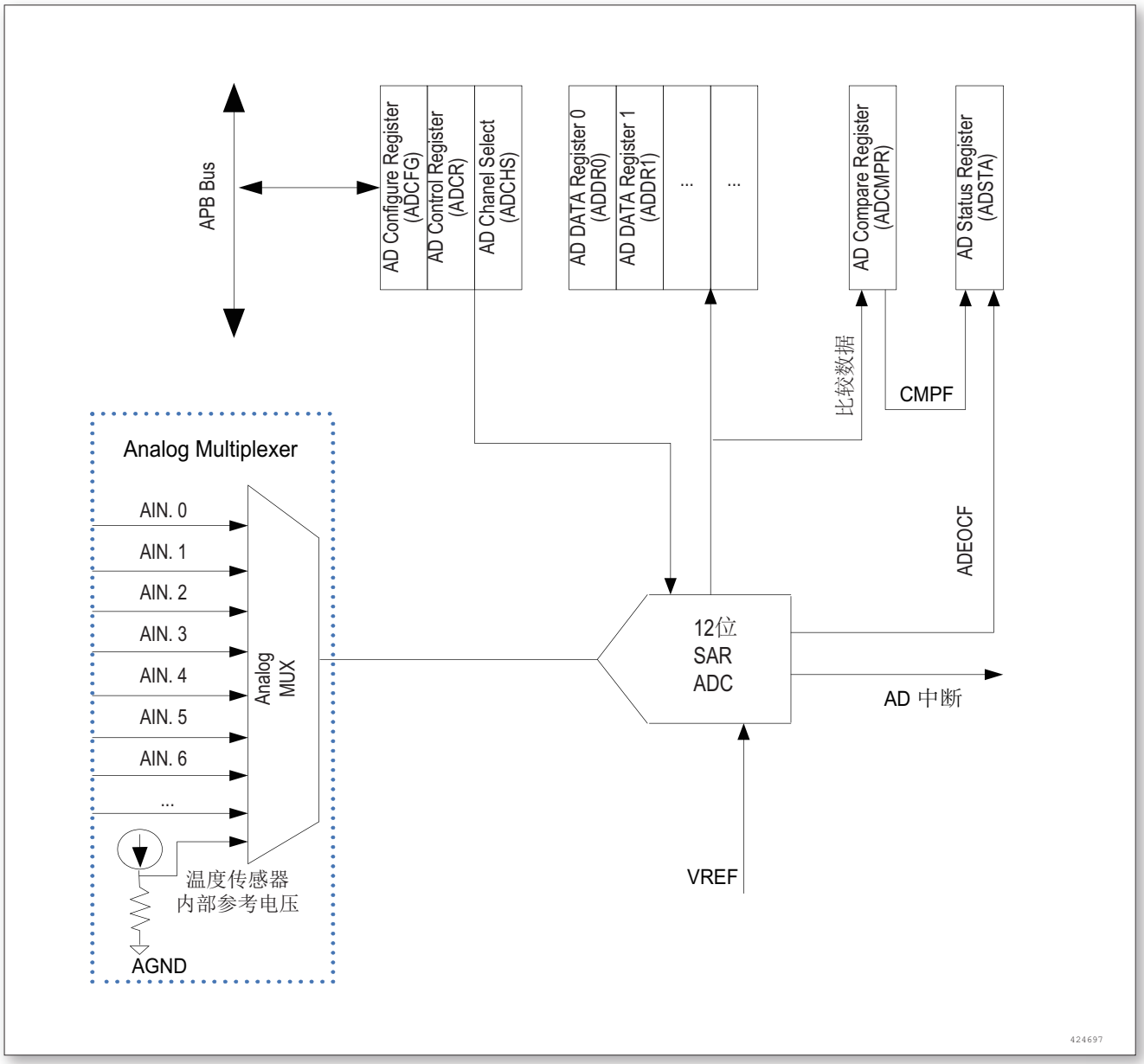


图 23. ADC 框图

10.3.1 ADC 开关控制

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

ADC 上电延迟一段时间后，设置 ADCR 寄存器的 ADST 位开始进行转换。

通过清除 ADST 位可以停止转换，清除 ADEN 位可置于断电模式。

10.3.2 通道选择

包含多路外部输入通道、内部温度传感器通道和内部 1.2V 参考电压通道。每个外部输入通道都有独立的使能位，可通过设置 ADCHS 寄存器的对应位来设置。

10.4 ADC 工作模式

10.4.1 单次转换模式

在单次转换模式下，A/D 转换相应通道上只执行一次，具体流程如下：

- 通过软件、外部触发输入及定时器溢出置位 ADCR 寄存器的 ADST，开始 A/D 转换。
- A/D 转换完成时，A/D 转换的数据值将存储于 A/D 的数据寄存器 ADDATA 和 ADDRn 中。
- A/D 转换完成时, 状态寄存器 ADSTA 的 ADIF 位置 1。若此时控制寄存器 ADCR 的 ADIE 位置 1，将产生 AD 转换结束中断请求。
- A/D 转换期间，ADST 位保持为 1。A/D 转换结束时，ADST 位自动清 0，A/D 转换器进入空闲模式。

注：在单次转换模式下，如果软件使能多于一个通道，序号最小的通道被转换，其他通道被忽略。

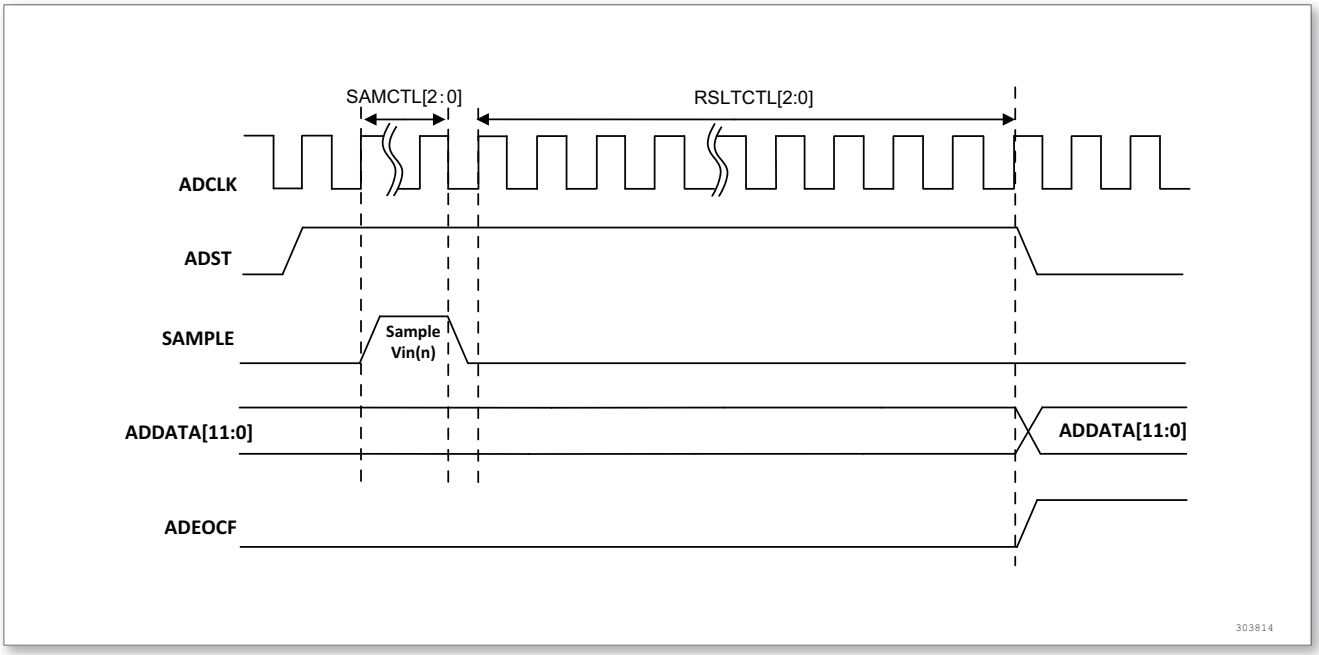


图 24. 单次转换模式时序图

10.4.2 单周期扫描模式

在单周期扫描模式下，将进行一次从被使能的最小序号通道向最大序号通道的 A/D 转换，操作步骤如下：

- 软件或外部触发置位 ADST 开始，从最小序号通道到最大序号通道的 A/D 转换。
- 每路 A/D 转换完成后，A/D 转换数值将有序装载到相应通道的数据寄存器中，ADIF 转换结束标志被设置，如果设置了转换结束中断，则在所有通道转换都完成后产生中断请求。
- 转换结束后，ADST 位自动清 0，A/D 转换器进入空闲状态。

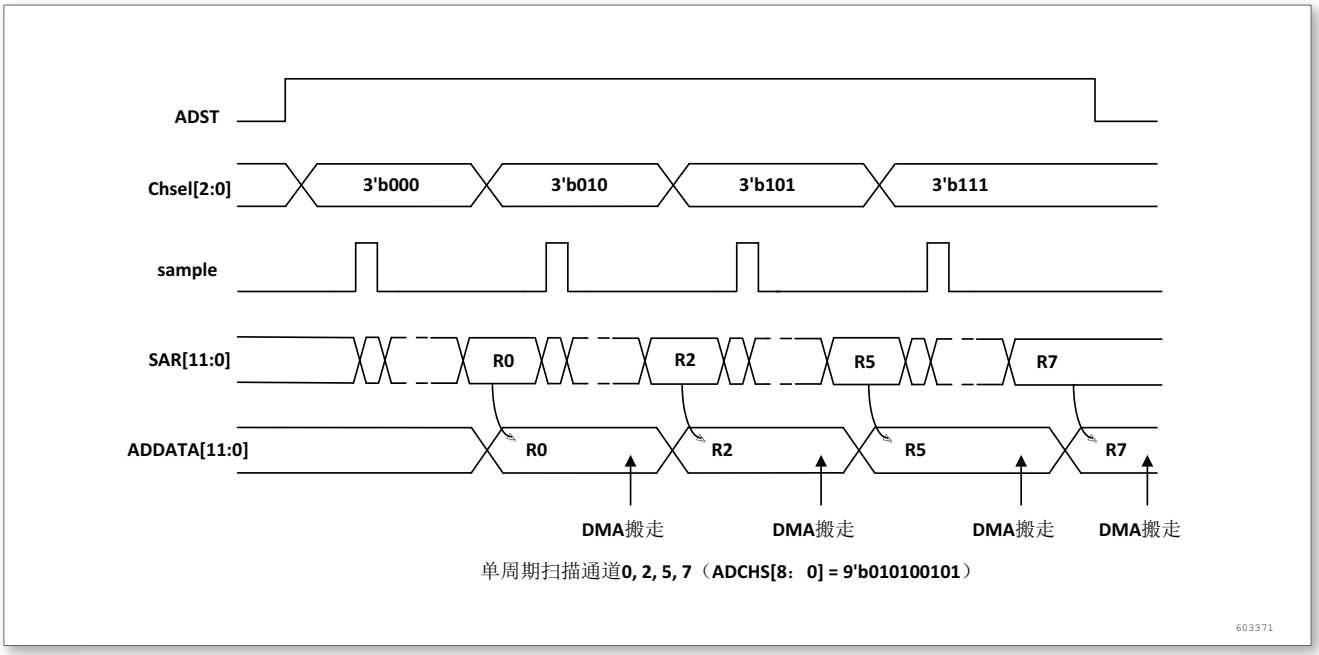


图 25. 单周期扫描下使能通道转换时序图

10.4.3 连续扫描模式

在连续扫描模式下，A/D 转换在 ADCHS 寄存器中的 CHENn 位被使能的通道上顺序进行，操作步骤如下：

- 软件或外部触发置位 ADST 开始，从最小序号通道到最大序号通道的 A/D 转换。
- 当所有通道的 A/D 转换完成一遍后，A/D 转换数值将有序装载到相应的数据寄存器中，ADIF 转换结束标志被设置，如果设置了转换结束中断，则在所有通道转换都完成后产生中断请求。
- 只要 ADST 位保持为 1，持续进行 A/D 转换。当 ADST 位被清 0，A/D 转换完成，A/D 转换器进入空闲状态。当 ADST 清 0，A/D 转换将完成当前转换。

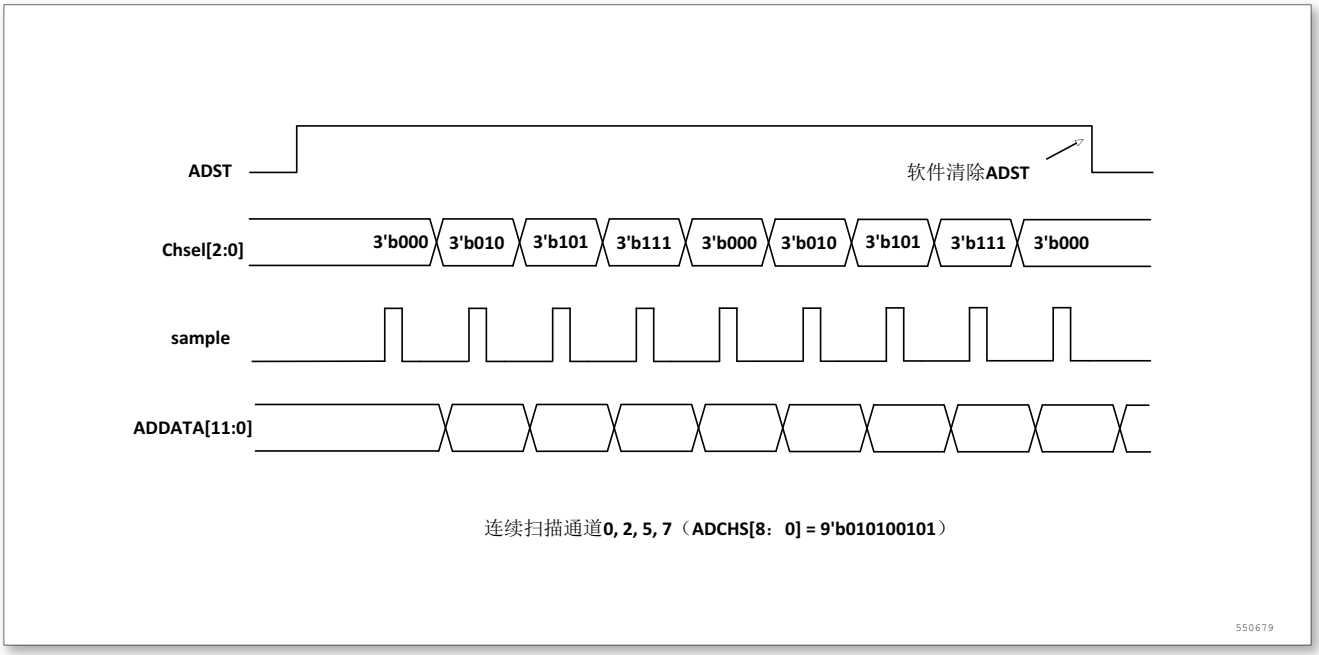


图 26. 连续扫描模式使能通道转换时序图

10.4.4 DMA 请求

单周期扫描和连续扫描时通道转换的值存储在各自通道的数据寄存 (ADDRn) 中，最近一次转换的结果也会保存在 ADDATA 寄存器中。DMA 传输时可以选择传输某个特定通道的数据，或者传输所有扫描通道的结果。

10.5 数据对齐

ADCR 寄存器中的 ALIGN 位选择转换后数据储存的对齐方式。数据可以左对齐或右对齐，如下图所示。

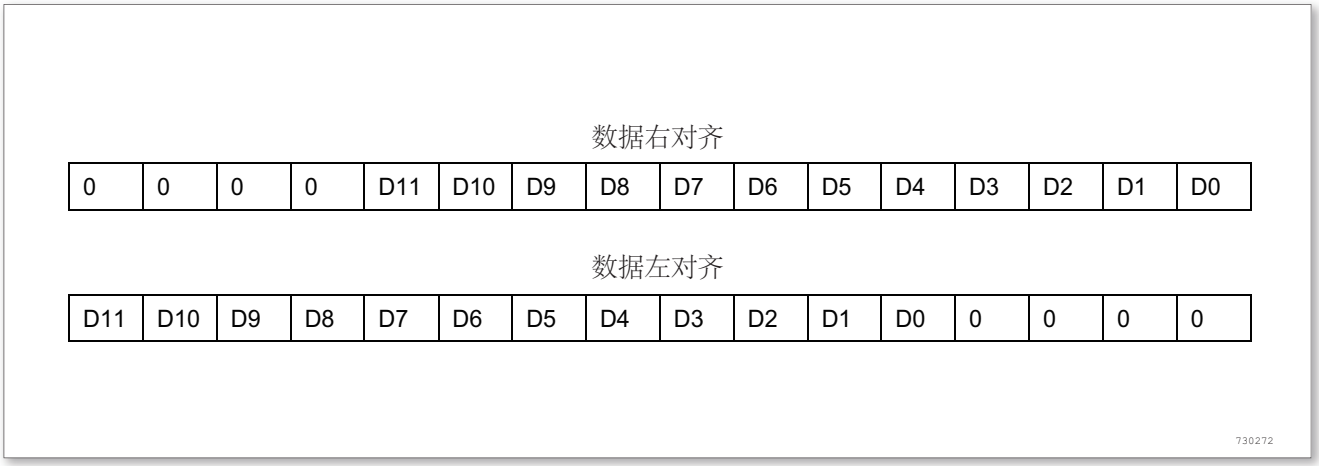


图 27. 数据对齐方式

10.5.1 可编程分辨率

ADC 转换有效位数可通过 ADC_CFG 寄存器中的 RSLTCTL[2：0] 位更改，以便加快数据转换速率，有效数据位是在 12 位数据高位对齐。

10.5.2 可编程采样时间

ADC 的时钟 ADCLK 由 PCLK2 分频得到，分频系数可通过设置 ADCFG 寄存器的 ADCPRE 位来确定，即 PCLK2 / (N+1) / 2 分频后作为 ADC 时钟。

设置 ADC 分辨率为 n 位 (n=8,9,10,11,12)，每个通道采样周期为 m，采样周期数目可以通过 ADC_CFG 寄存器中的 SAMCTL 位更改。

采样频率采样时间计算如下：

$F_{sample} = F_{ADCLK} / (m + n + 1.5)$。

例如：

当 ADCCLK = 15MHz，分辨率配置为 12Bit，采样时间为 1.5 周期

$F_{sample} = F_{ADCLK} / 15$。

T_{CONV} = 1.5 + 13.5 = 15 周期 = 1μs

10.6 外部触发转换

ADC 转换可以由外部事件触发 (例如定时器捕获，EXTI 线)。如果设置了 ADCR 寄存器的 TRGEN 位，就可以使用外部事件触发转换。通过设置 TRGSEL 位可以选择外部触发源。

具体的外部触发源选择情况，可以参考 AD 控制寄存器相关位的描述。

在触发信号产生后，延时 N 个 PCLK2 的时钟周期再开始采样。如果是触发扫描模式，只有第一个通道采样被延时，其余通道是在上一个采样结束后立即开始。

10.7 温度传感器

温度传感器可以用来测量器件周围的温度 (T_A)。

温度传感器在内部和 ADC 的内部信号源通道相连接，此通道把传感器输出的电压转换成数字值。当传感器不使用时，可设置寄存器相应位来单独关闭。

温度传感器输出电压随温度线性变化，由于生产过程的变化，温度变化曲线的偏移在不同芯片上会有不同。

内部温度传感器更适合于检测温度的变化，而不是测量绝对的温度。如果需要测量精确的温度，应该使用一个外置的温度传感器。

温度数值计算如下：

$T(°C) = (V_{SENSE} - V_{25}) / Avg\_Slope + 25$

V_{25}: 25°C 时的 V_{SENSE} 值

V_{SENSE}: 温度传感器当前的输出电压

$V_{SENSE} = Value * V_{dd} / 4096$ (Value 是 ADC 的转换结果数据)

Avg_Slope: 温度与 V_{SENSE} 曲线的平均斜率 (以 mV/°C 或 μV/°C 表示)

V_{25} 和 Avg_Slope 的典型值请参考数据手册温度传感器章节。

10.8 内部基准参考电压

ADC 的内部信号源通道连接了一个内部基准参考电压，大小为 1.2V，此通道把 1.2v 的参考电压输出转换为数字值。

内部参考电压有单独的始能位，可通过设置寄存器的相应位开启或关闭。

10.9 窗口比较器模式下 AD 转换结果监控

比较模式下提供了上限和下限两个比较寄存器。可通过软件设定 CMPCH 位选择监控通道。

当 CPMHDATA ≥ CPMLDATA 时，比较结果大于或等于 ADCMPR 寄存器的 CMPHDATA 指定值或者小于 CMPLDATA 指定值，状态寄存器 ADSTA 的 ADWIF 位置 1。

当 CPMHDATA < CPMLDATA 时，比较结果如果等于 CPMHDATA 指定值或者处于两个指定值之间，则状态寄存器 ADSTA 的 ADWIF 位置 1。如果控制寄存器 ADCR 的 ADWIE 置位，将产生中断请求。

10.10 ADC 寄存器描述

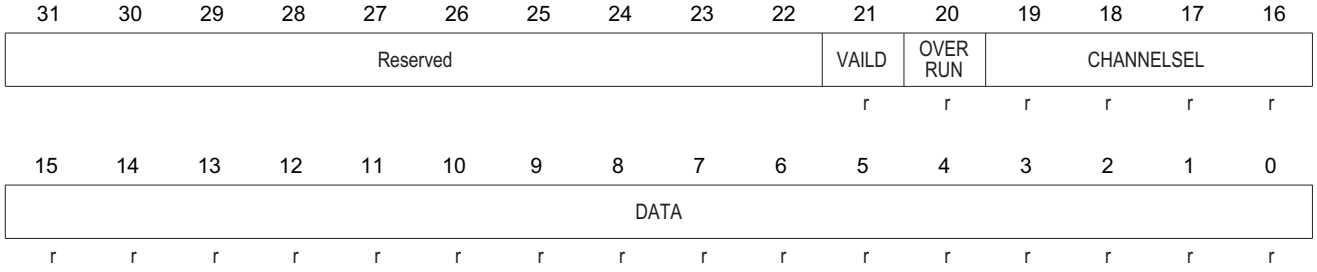
表 46. ADC 寄存器概览

Offset	Acronym	Register Name	Reset	Section
0x00	ADC_ADDATA	A/D 数据寄存器	0x00000000	小节 10.10.1
0x04	ADC_ADCFG	A/D 配置寄存器	0x00000000	小节 10.10.2
0x08	ADC_ADCR	A/D 控制寄存器	0x00000000	小节 10.10.3
0x0C	ADC_ADCHS	A/D 通道选择寄存器	0x00000000	小节 10.10.4
0x10	ADC_ADCMPR	A/D 窗口比较寄存器	0x00000000	小节 10.10.5
0x14	ADC_ADSTA	A/D 状态寄存器	0x00000000	小节 10.10.6
0x18 ~ 0x38	ADC_ADDR 0 ~ 8	A/D 数据寄存器	0x00000000	小节 10.10.7

10.10.1 A/D 数据寄存器 (ADC_ADDATA)

地址偏移：0x00

复位值：0x0000 0000



Bit	Field	Type	Reset	Description
31 : 22	Reserved			保留，始终读为 0。
21	VALID	r	0x00	有效标志位 (只读)(Valid flag) 1 = DATA[11：0] 位数据有效 0 = DATA[11：0] 位数据无效 相应模拟通道转换完成后，将该位置位，读 ADDATA 寄存器后，该位由硬件清除。
20	OVERRUN	r	0x00	数据覆盖标志位 (只读)(Overrun flag) 1 = DATA[11：0] 数据被覆盖 0 = DATA[11：0] 数据最近一次转换结果 新的转换结果装载至寄存器之前，若 DATA[11：0] 的数据没有被读取，OVERRUN 将置 1。读 ADDATA 寄存器后，该位由硬件清除。

Bit	Field	Type	Reset	Description
19：16	CHANNELSEL	r	0x00	该 4 位显示当前数据所对应的通道 (Channel selection) 0000 = 通道 0 的转换数据 0001 = 通道 1 的转换数据 0010 = 通道 2 的转换数据 0011 = 通道 3 的转换数据 0100 = 通道 4 的转换数据 0101 = 通道 5 的转换数据 0110 = 通道 6 的转换数据 0111 = 通道 7 的转换数据 1000 = ADC1 中是温度传感器的转换数据，ADC2 中是内部参考电压的转换数据) 其他：无效
15：0	DATA	r	0x00	12 位 A/D 转换结果 (Transfer data) 根据设置左对齐或者右对齐。

10.10.2 A/D 配置寄存器 (ADC_ADCFG)

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
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			SAMCTL			RSLTCTL			ADCPRE			Res.	TVSEN	ADWEN	ADEN
			rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw

Bit	Field	Type	Reset	Description
31：13	Reserved			保留，始终读为 0。
12：10	SAMCTL	rw	0x00	选择通道 x 的采样时间 (Channel x Sample time selection) 这些位用于独立地选择每个通道的采样时间。在采样周期中通道选择位必须保持不变。 000：1.5 周期　　100：41.5 周期 001：7.5 周期　　101：55.5 周期 010：13.5 周期　　110：71.5 周期 011：28.5 周期　　111：239.5 周期
9：7	RSLTCTL	rw	0x00	选择 ADCx 转换数据分辨率 (resolution) 000：12 位有效　　001：11 位有效 010：10 位有效　　011：9 位有效 100：8 位有效
6：4	ADCPRE	rw	0x00	ADC 预分频 (ADC prescaler) 由软件置‘1’或清‘0’来确定 ADC 时钟频率。 n：PCLK2 经过 2*(n+1) 分频后作为 ADC 时钟

Bit	Field	Type	Reset	Description
3	Reserved			保留，始终读为 0。
2	TVSEN	rw	0x00	温度传感器和内部参考电压使能控制位 (Temperature sensor enable) 1 = 使能 0 = 禁止 注：在 ADC1 中为温度传感器始能，在 ADC2 中为内部参考电压始能。
1	ADWEN	rw	0x00	A/D 窗口比较器使能 (ADC window comparison enable) 1 = A/D 窗口比较器使能 0 = A/D 窗口比较器禁用
0	ADEN	rw	0x00	A/D 转换使能 (ADC enable) 1 = 使能 0 = 禁用

10.10.3 A/D 控制寄存器 (ADC_ADCR)

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
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CMPCH				ALIGN	ADMD		ADST	SYNCEN	TRGSEL			DMAEN	TRGEN	ADWIE	ADIE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
31：16	Reserved			保留，始终读为 0。
15：12	CMPCH	rw	0x00	窗口比较通道选择 (Window comparison channel selection) 0000 = 选择比较通道 0 转换结果 0001 = 选择比较通道 1 转换结果 0010 = 选择比较通道 2 转换结果 0011 = 选择比较通道 3 转换结果 0100 = 选择比较通道 4 转换结果 0101 = 选择比较通道 5 转换结果 0110 = 选择比较通道 6 转换结果 0111 = 选择比较通道 7 转换结果 1000 = 选择温度传感器或内部参考电压 其他：无效
11	ALIGN	rw	0x00	数据对齐 (Data alignment) 0：右对齐 1：左对齐

Bit	Field	Type	Reset	Description
10：9	ADMD	rw	0x00	A/D 转换模式 (ADC mode) 00：单次转换 01：单周期扫描 10：连续扫描 当改变转换模式时，软件要先禁用 ADST 位。
8	ADST	rw	0x00	A/D 转换开始 (ADC start) 1 = 转换开始 0 = 转换结束或进入空闲状态 ADST 置位有下列两种方式： 在单次模式或者单周期模式下，转换完成后，ADST 将被硬件自动清除。 在连续扫描模式下，A/D 转换将一直进行，直到软件写‘0’到该位或系统复位。
7	SYNCEN	rw	0x00	两个 ADC 同步转换使能 0：两个 ADC 分别转换 1：作为从 ADC，与另一个 ADC 同步转化 注：两个 ADC 的 SYNCEN 不能同时设为 1。
6：4	TRGSEL	rw	0x00	外部触发源选择 (External trigger selection) ADC1 的触发配置如下： 000：TIM1_CC1 001：TIM1_CC2 010：TIM1_CC3 011：TIM2_CC2 100：TIM3_TRGO 101：TIM4_CC4 110：TIM3_CC1 111：EXTI 线 11 ADC2 的触发配置如下： 000：TIM1_TRGO 001：TIM1_CC4 010：TIM2_TRGO 011：TIM2_CC1 100：TIM3_CC4 101：TIM4_TRGO 110：TIM3_CC1 111：EXTI 线 15
3	DMAEN	rw	0x00	DMA 使能 (Direct memory access enable) 1 = DMA 请求使能 0 = DMA 禁止
2	TRGEN	rw	0x00	外部硬件触发源 (External trigger enable) 1 = 使用外部触发信号启动 A/D 转换 0 = 不用外部触发信号启动 A/D 转换

Bit	Field	Type	Reset	Description
1	ADWIE	rw	0x00	A/D 窗口比较器中断使能 (ADC window comparator interrupt enable) 1 = 使能 A/D 窗口比较器中断 0 = 禁用 A/D 窗口比较器中断
0	ADIE	rw	0x00	A/D 中断使能 (ADC interrupt enable) 1 = 使能 A/D 中断 0 = 禁用 A/D 中断 如果 ADIF 置位，A/D 转换结束后产生中断请求。

10.10.4 A/D 通道选择寄存器 (ADC_ADCHS)

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
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved							SENSOREN	CHEN7	CHEN6	CHEN5	CHEN4	CHEN3	CHEN2	CHEN1	CHEN0
							rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
31 : 9	Reserved			保留，始终读为 0。
8	SENSOREN	rw	0x00	传感器始能 (Sensor enable) 1 = 使能 0 = 禁用
7	CHEN7	rw	0x00	模拟输入通道 7 使能 (Analog input channel 7 enable) 1 = 使能 0 = 禁用
6	CHEN6	rw	0x00	模拟输入通道 6 使能 (Analog input channel 6 enable) 1 = 使能 0 = 禁用
5	CHEN5	rw	0x00	模拟输入通道 5 使能 (Analog input channel 5 enable) 1 = 使能 0 = 禁用
4	CHEN4	rw	0x00	模拟输入通道 4 使能 (Analog input channel 4 enable) 1 = 使能 0 = 禁用
3	CHEN3	rw	0x00	模拟输入通道 3 使能 (Analog input channel 3 enable) 1 = 使能 0 = 禁用

Bit	Field	Type	Reset	Description
2	CHEN2	rw	0x00	模拟输入通道 2 使能 (Analog input channel 2 enable) 1 = 使能 0 = 禁用
1	CHEN1	rw	0x00	模拟输入通道 1 使能 (Analog input channel 1 enable) 1 = 使能 0 = 禁用
0	CHEN0	rw	0x00	模拟输入通道 0 使能 (Analog input channel 0 enable) 1 = 使能 0 = 禁用

注: 如果通道使能都为 0，则通道 0 使能。

10.10.5 A/D 窗口比较寄存器 (ADC_ADCMPR)

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

Bit	Field	Type	Reset	Description
31 : 28	Reserved			保留，始终读为 0。
27 : 16	CMPHDATA	rw	0x00	比较数值上限 (Compare data high limit) 该 12 位数值将和指定通道的转换结果相比较。
15 : 12	Reserved			保留，始终读为 0。
11 : 0	CMPLDATA	rw	0x00	比较数值下限 (Compare data low limit) 该 12 位数值将和指定通道的转换结果相比较。

10.10.6 A/D 状态寄存器 (ADC_ADSTA)

地址偏移：0x14

复位值：0x0000 0000

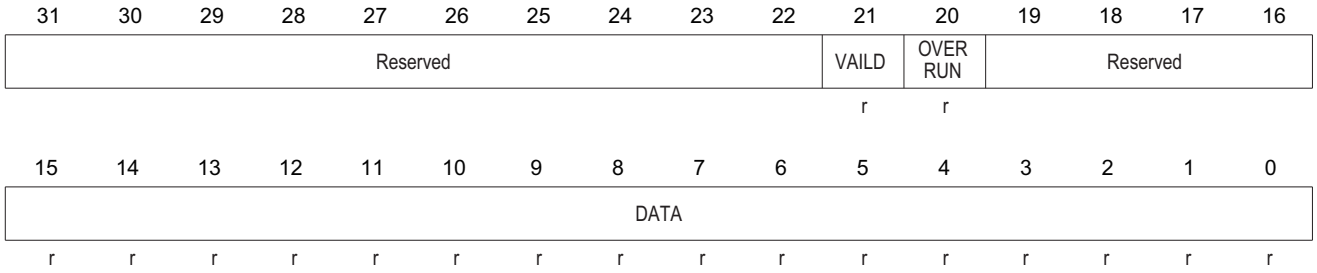
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved			OVERRUN									Reserved			VALID
			r	r	r	r	r	r	r	r	r				r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VALID								CHANNEL				Res.	BUSY	ADWIF	ADIF
r	r	r	r	r	r	r	r	r	r	r	r		r	rc_w1	rc_w1

Bit	Field	Type	Reset	Description
31：29	Reserved			保留，始终读为 0。
28：20	OVERRUN	r	0x00	通道 0 ~ 8 的数据覆盖标志位 (Overrun flag) 只读。
19：17	Reserved			保留，始终读为 0。
16：8	VALID	r	0x00	通道 0 ~ 8 的有效标志位 (Valid flag) 只读。
7：4	CHANNEL	r	0x00	当前转换通道 (Current conversion channel) 该 4 位在 BUSY = 1 时表示进行转换中的通道。BUSY = 0 时表示可进行下次转换的通道。
3	Reserved			保留，始终读为 0。
2	BUSY	r	0x00	忙/空闲 (Busy) 1 = A/D 转换器忙碌 0 = A/D 转换器空闲
1	ADWIF	rc_w1	0x00	比较标志位 (ADC window comparator interrupt flag) 选择的 A/D 转换通道，结果大于等于 ADCMPHR 或小于 ADCMPLR，该位置‘1’。 该标志位写‘1’清零。
0	ADIF	rc_w1	0x00	A/D 转换结束标志位 (ADC interrupt flag) 该位由硬件在通道组转换结束时设置，由软件清除。 1 = A/D 转换完成 0 = A/D 转换未完成 该标志位写‘1’清零。

10.10.7 A/D 数据寄存器 (ADC_ADDR0 ~ 8)

地址偏移：0x18 – 0x38

复位值：0x0000 0000



Bit	Field	Type	Reset	Description
31：22	Reserved			保留，始终读为 0。
21	VALID	r	0x00	有效标志位 (只读)(Valid flag) 1 = DATA[11：0] 位数据有效 0 = DATA[11：0] 位数据无效 相应模拟通道转换完成后，将该位置位，读 ADDATA 寄存器后，该位由硬件清除。

Bit	Field	Type	Reset	Description
20	OVERRUN	r	0x00	数据覆盖标志位 (只读)(Overrun flag) 1 = DATA [11：0] 数据被覆盖 0 = DATA [11：0] 数据最近一次转换结果 新的转换结果装载至寄存器之前，若 DATA[11：0] 的数据没有被读取，OVERRUN 将置‘1’，读 ADDATA 寄存器后，该位由硬件清除。
19：16	Reserved			保留，始终读为 0。
15：0	DATA	r	0x00	通道的 12 位 A/D 转换结果 (Transfer data) 根据设置左对齐或者右对齐。

11 高级控制定时器 (TIM1)

高级控制定时器 (TIM1)

11.1 TIM1 简介

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

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

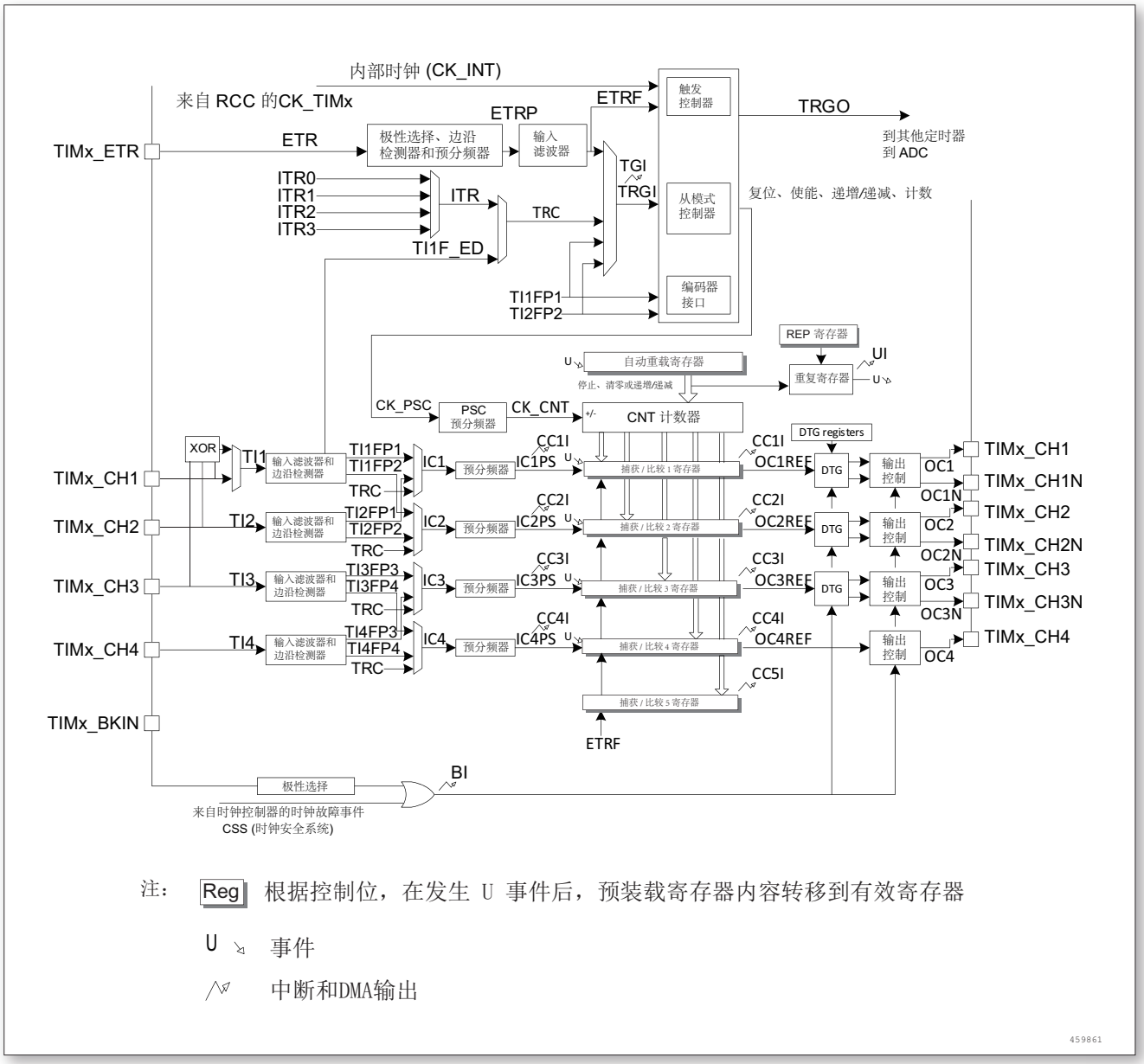
使用定时器预分频器和 RCC 时钟控制预分频器，可以实现脉冲宽度和波形周期从几个微秒到几个毫秒的调节。

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

11.2 主要特征

TIM1 定时器的功能包括：

- 16 位向上、向下、向上/下自动装载寄存器
- 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
- 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



注： Reg 根据控制位，在发生 U 事件后，预装载寄存器内容转移到有效寄存器

U 事件

中断和DMA输出

图 28. 高级控制定时器框图

11.3 功能描述

11.3.1 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：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 (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。

下面两个图分别给出了在预分频器运行时，更改计数器参数的例子。

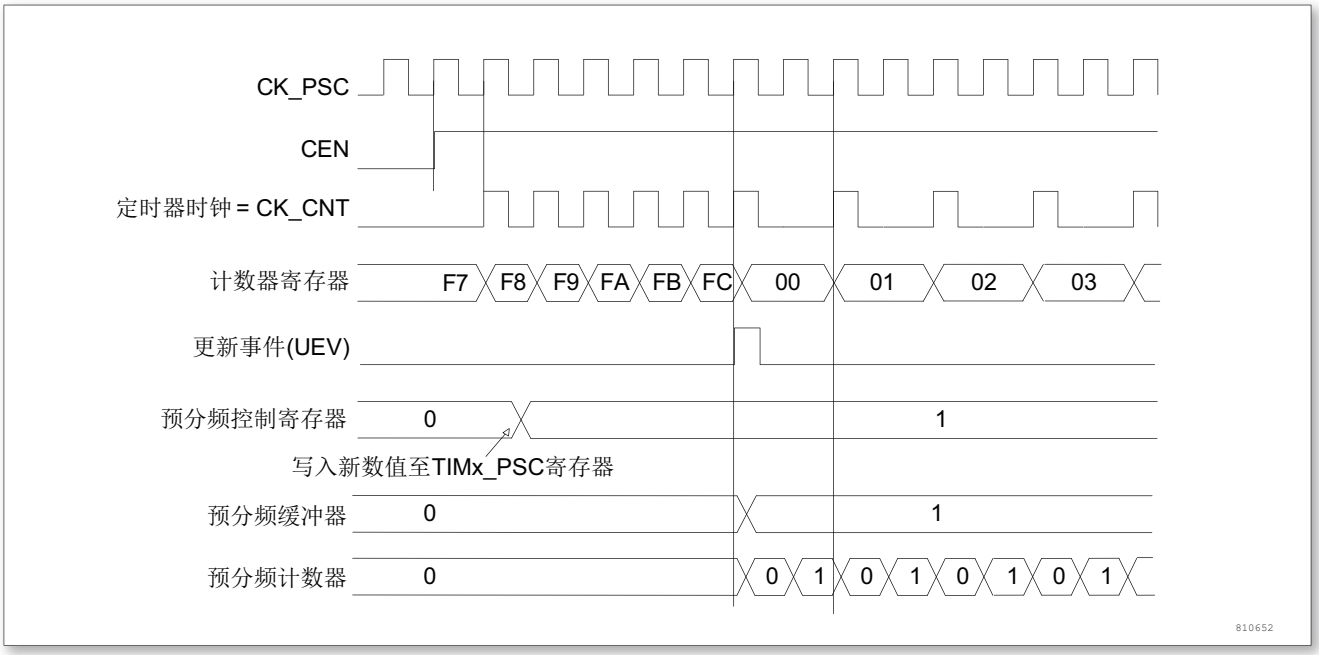


图 29. 当预分频器的参数从 1 变到 2 时，计数器的时序图

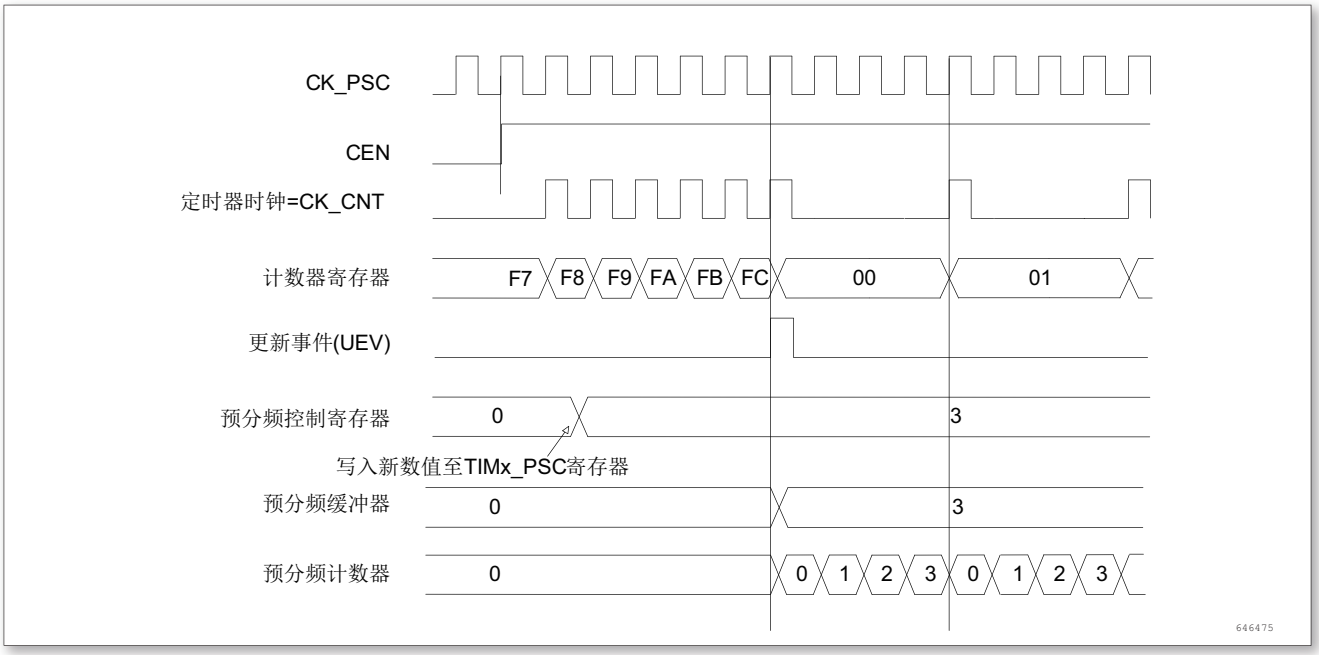


图 30. 当预分频器的参数从 1 变到 4 时，计数器的时序图

11.3.2 计数模式

向上计数模式

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

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

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

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

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

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

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

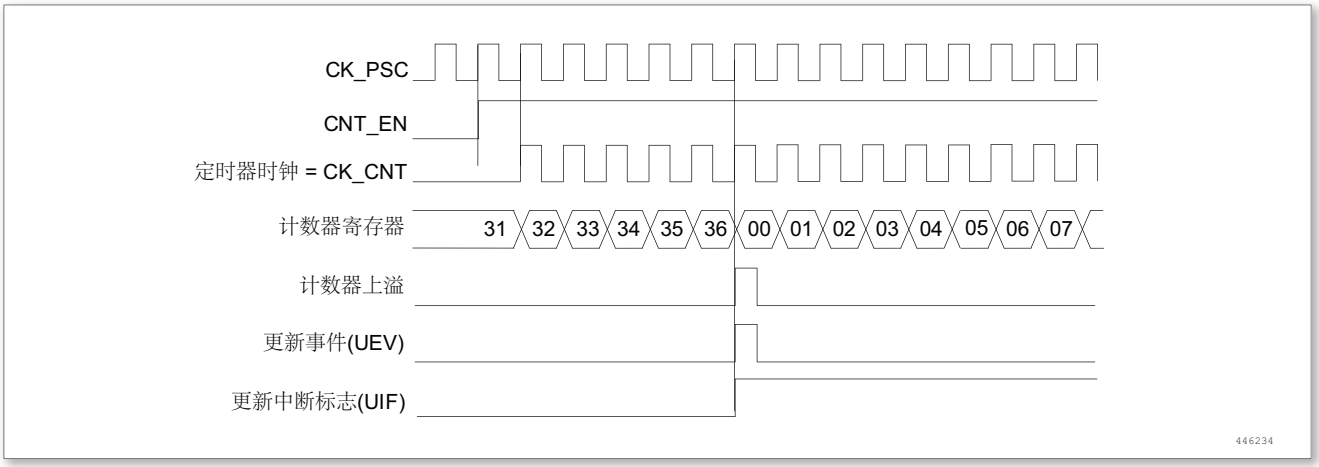


图 31. 计数器时序图，内部时钟分频因子为 1

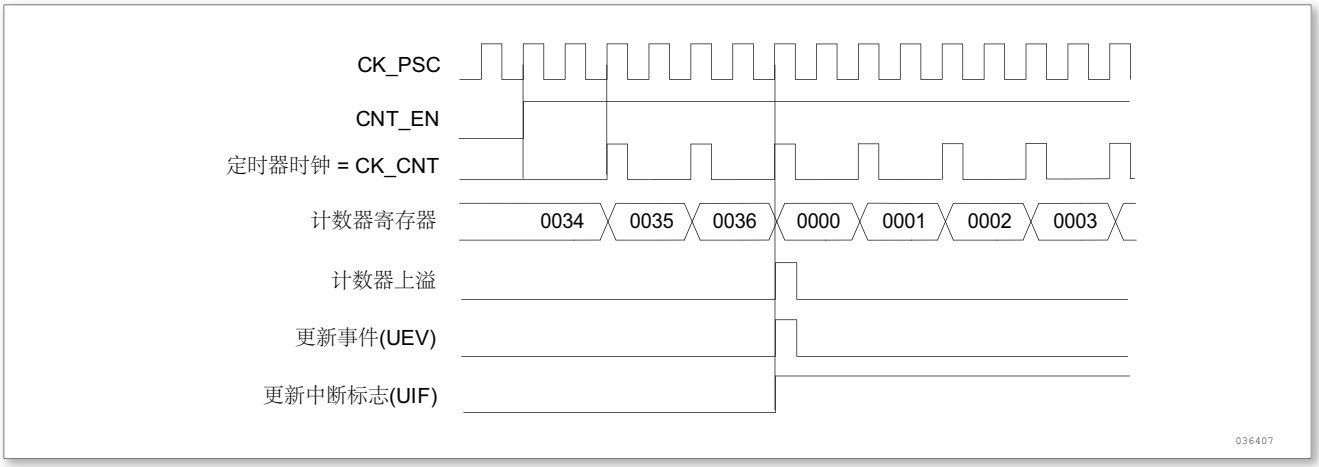


图 32. 计数器时序图，内部时钟分频因子为 2

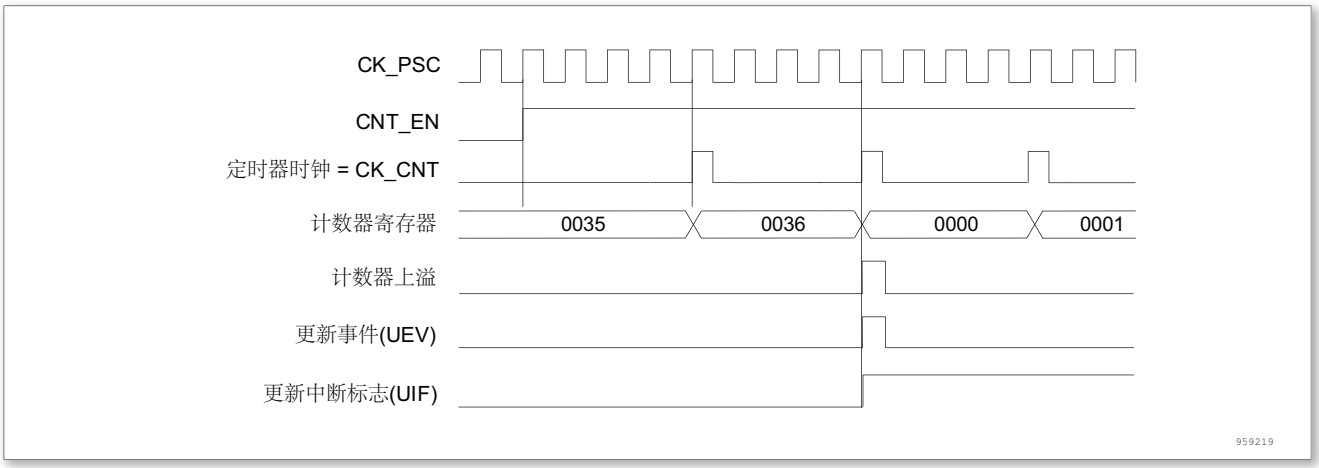


图 33. 计数器时序图，内部时钟分频因子为 4

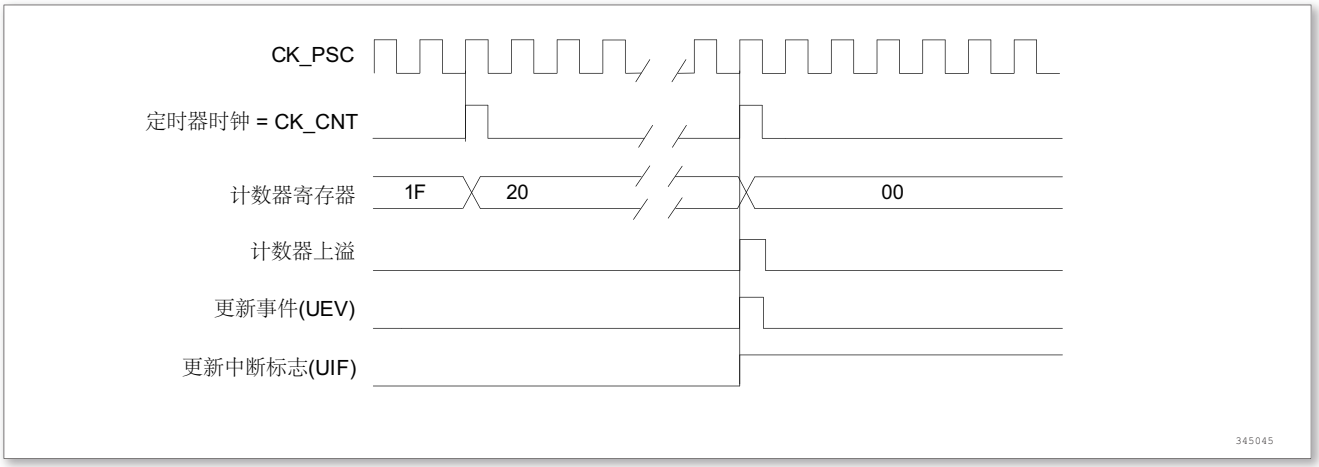


图 34. 计数器时序图，内部时钟分频因子为 N

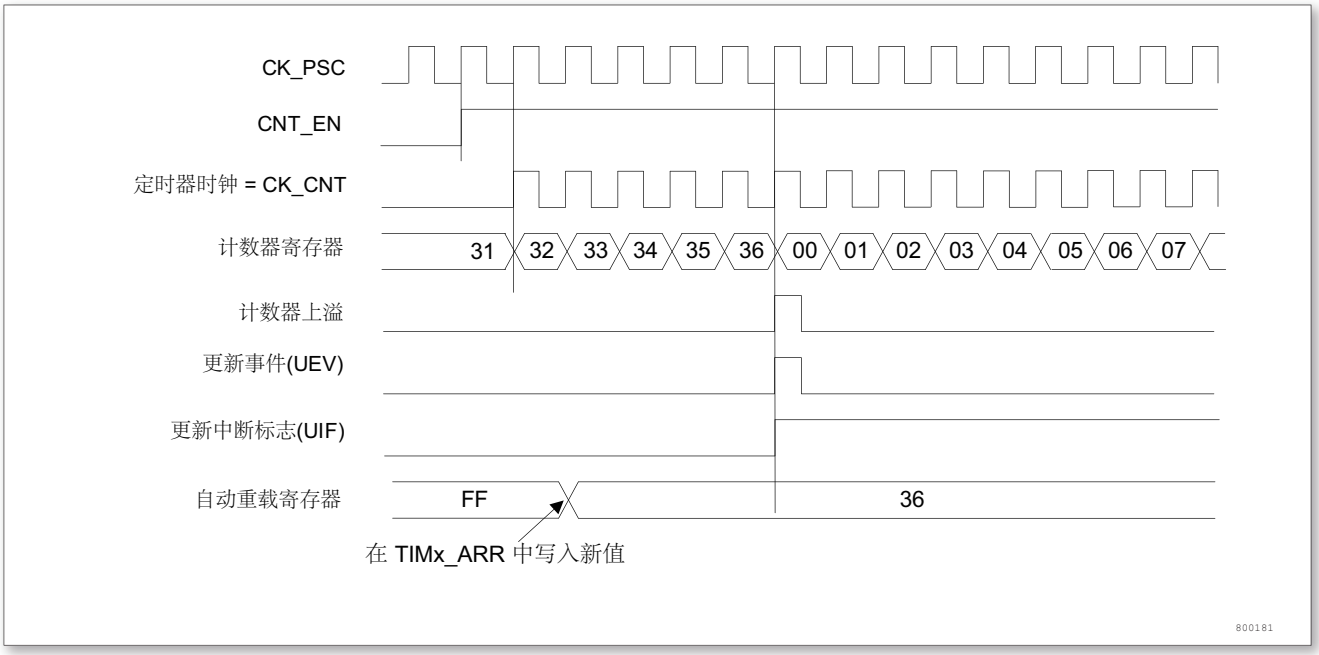


图 35. 计数器时序图，当 ARPE = 0 时的更新事件 (TIMx_ARR 没有预装入)

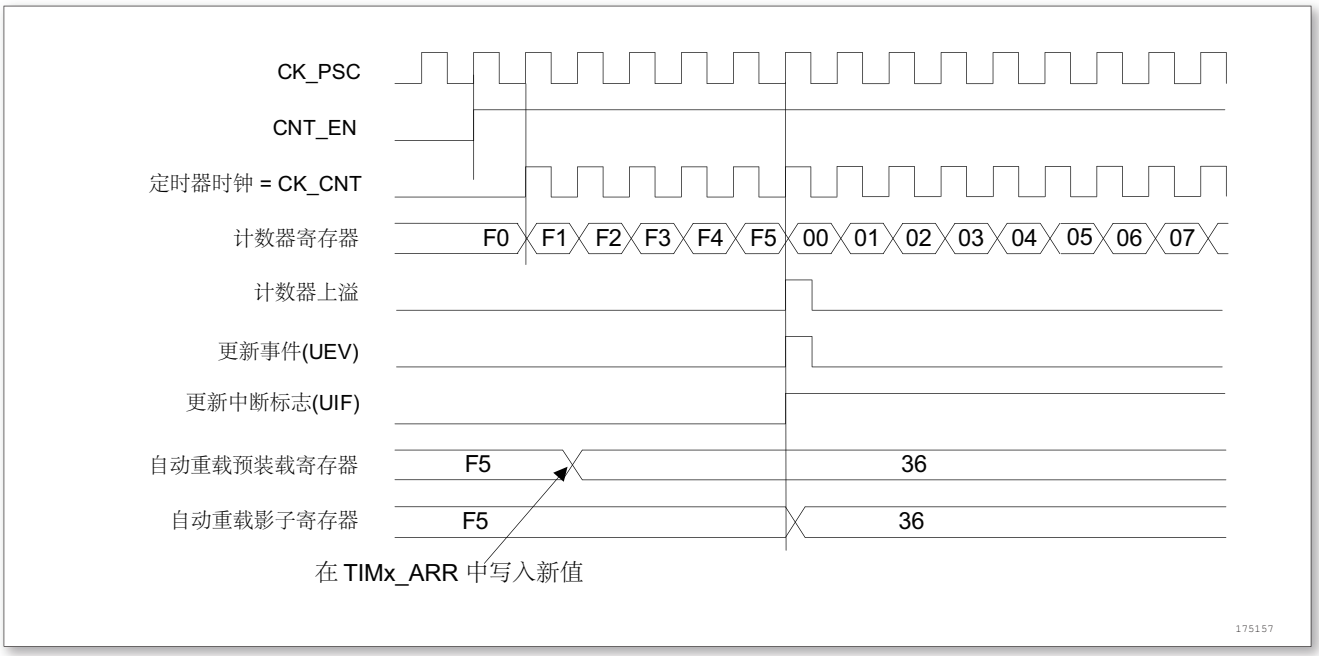


图 36. 计数器时序图，当 ARPE = 1 时的更新事件 (预装入了 TIMx_ARR)

向下计数模式

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

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

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

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

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

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

- 重复计数器被重置为 TIMx_RCR 寄存器中的内容。
- 预分频器的缓存器被加载为预装载的值 (TIMx_PSC 寄存器的值)。
- 当前的自动加载寄存器被更新为预装载值 (TIMx_ARR 寄存器中的内容)。

注：自动装载在计数器重载入之前被更新，因此下一个周期将是预期的值。

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

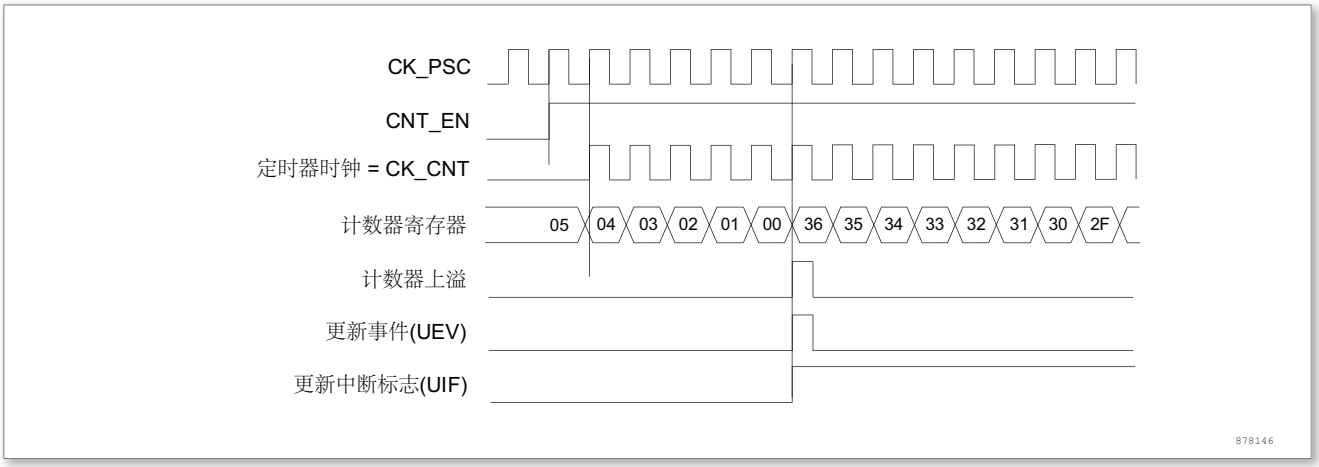


图 37. 计数器时序图，内部时钟分频因子为 1

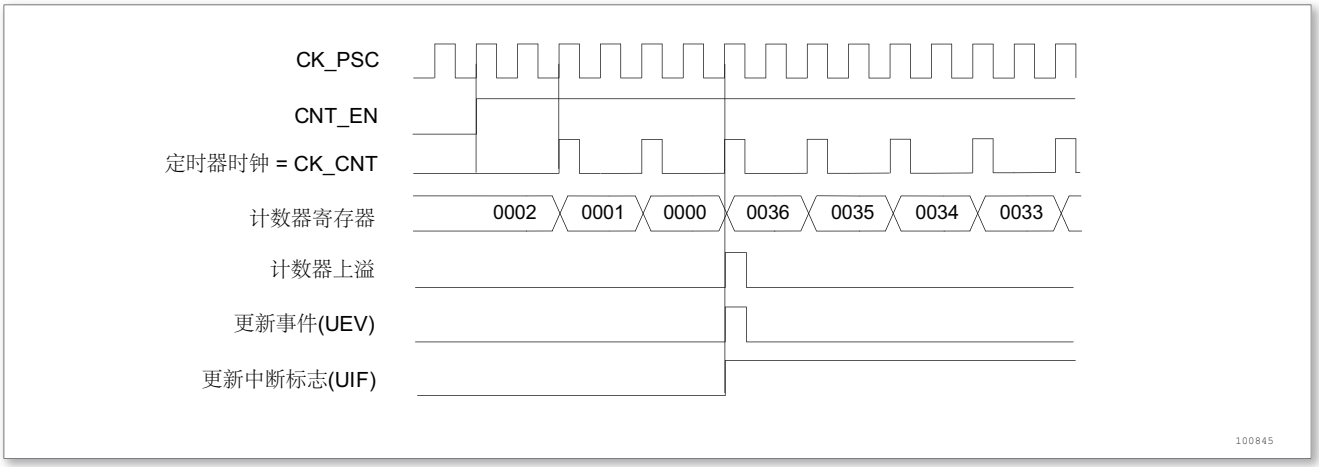


图 38. 计数器时序图，内部时钟分频因子为 2

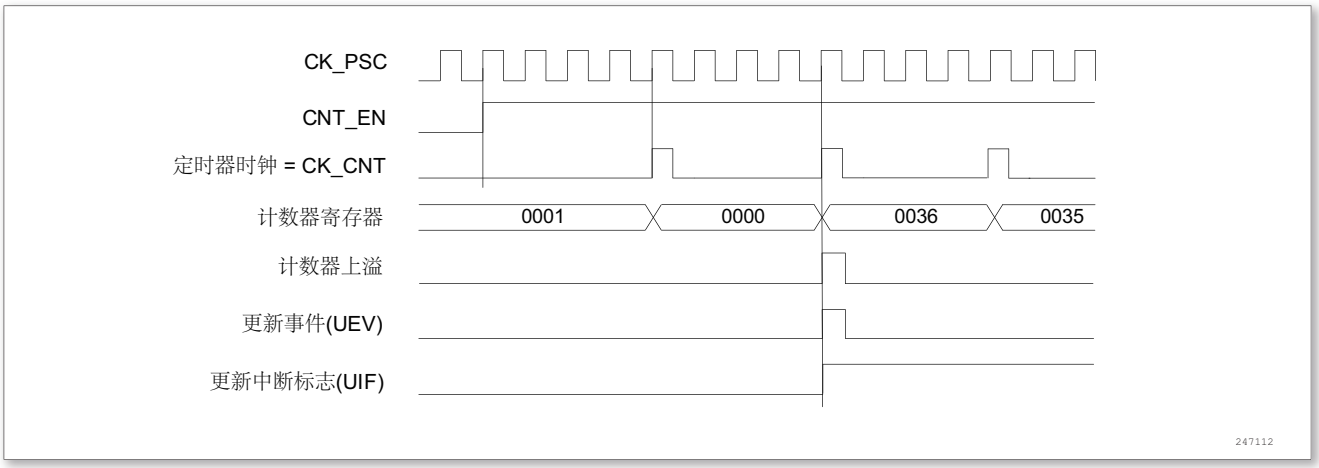


图 39. 计数器时序图，内部时钟分频因子为 4

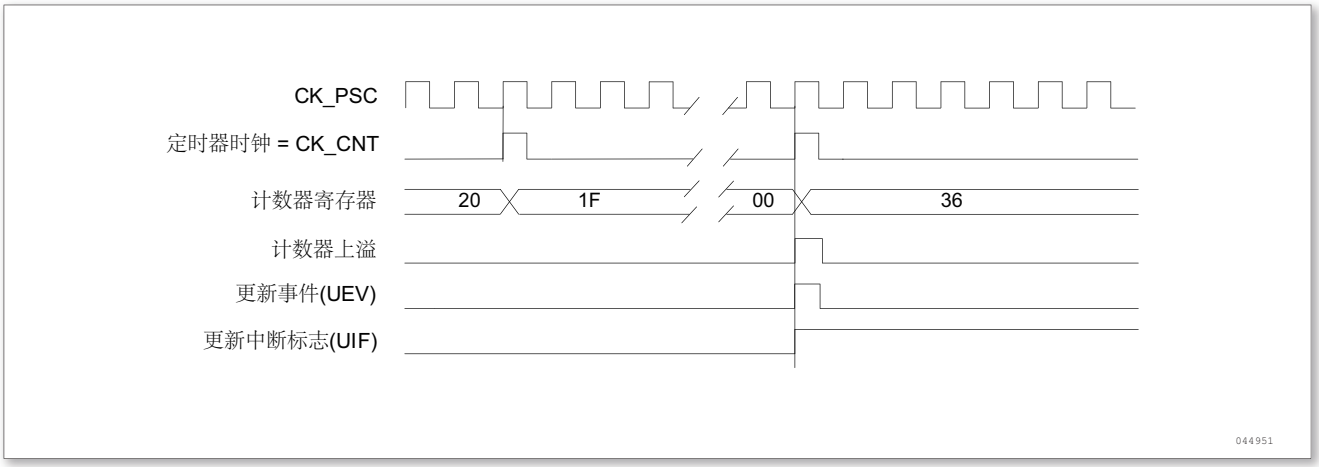


图 40. 计数器时序图，内部时钟分频因子为 N

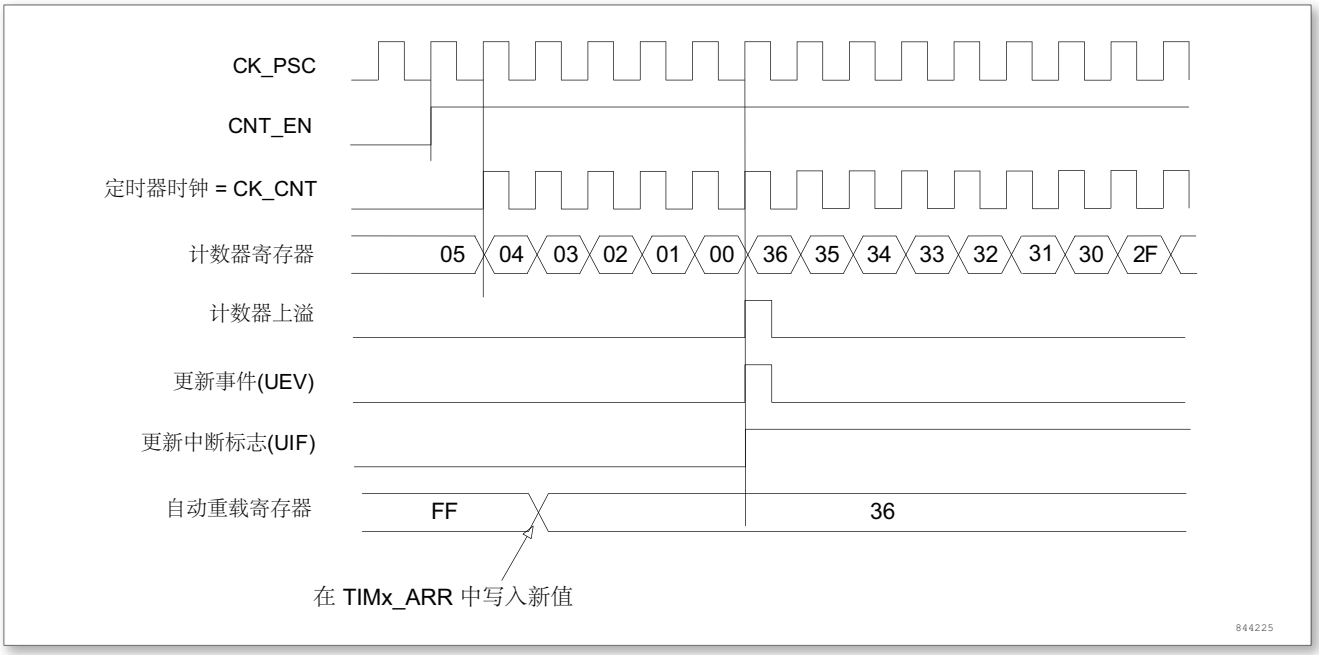


图 41. 计数器时序图，当没有使用重复计数器时的更新事件

中央对齐模式 (向上/向下计数)

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

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

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

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

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

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

- 重复计数器被重置为 TIMx_RCR 寄存器中的内容。
- 预分频器的缓存器被加载为预装载 (TIMx_PSC 寄存器) 的值。
- 当前的自动加载寄存器被更新为预装载值 (TIMx_ARR 寄存器中的内容)。

注：如果因为计数器溢出而产生更新，自动重装载将在计数器重载入之前被更新，因此下一个周期将是预期的值 (计数器被装载为新的值)。

以下是一些计数器在不同时钟频率下的操作的例子：

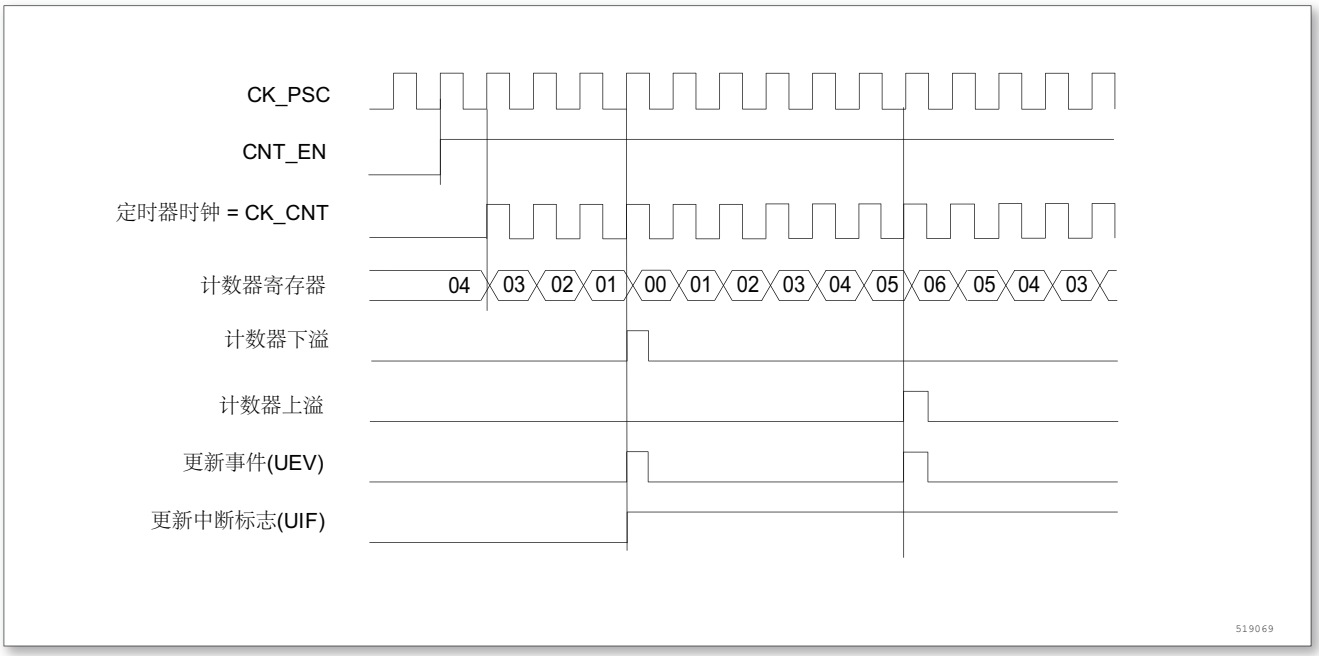


图 42. 计数器时序图，内部时钟分频因子为 1，TIMx_ARR = 0x6

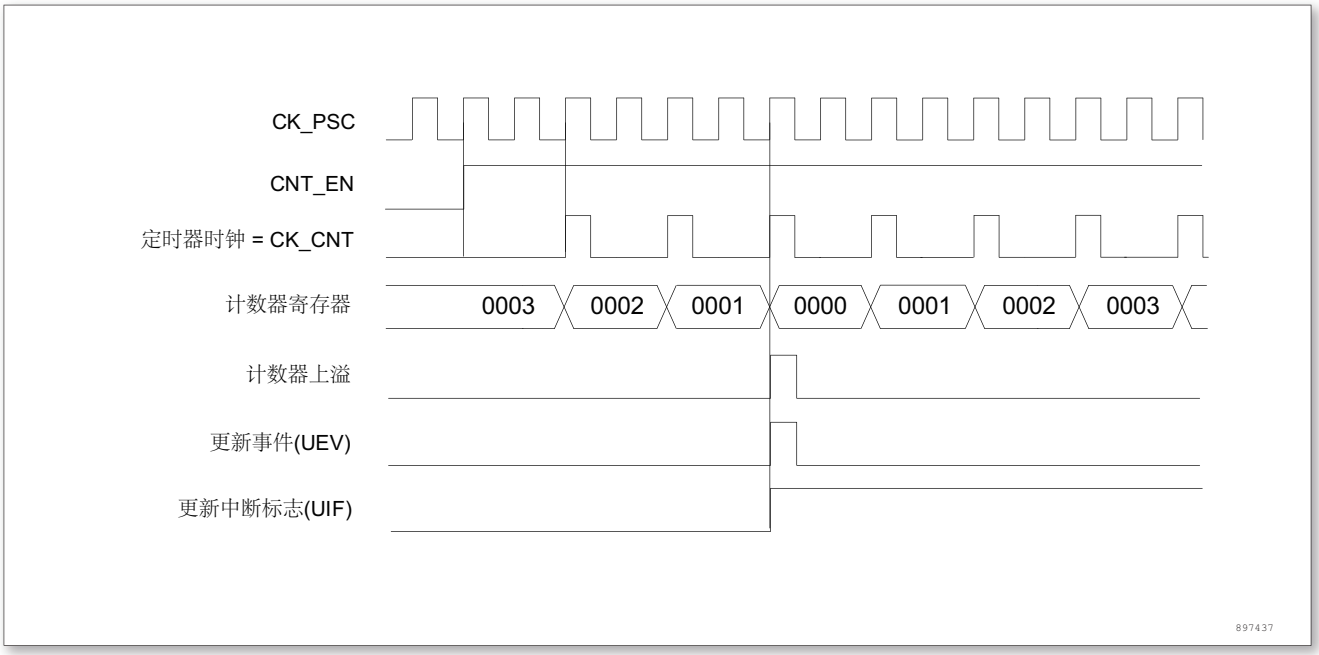
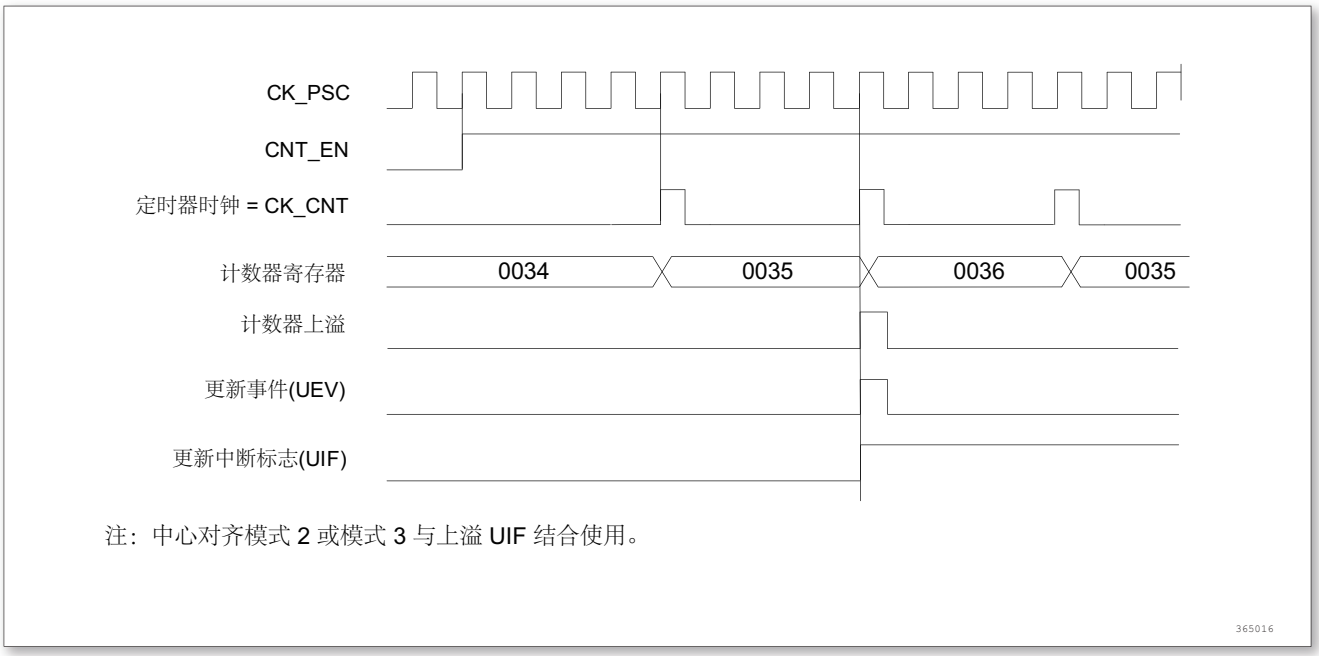


图 43. 计数器时序图，内部时钟分频因子为 2



注：中心对齐模式 2 或模式 3 与上溢 UIF 结合使用。

图 44. 计数器时序图，内部时钟分频因子为 4，TIMx_ARR = 0x36

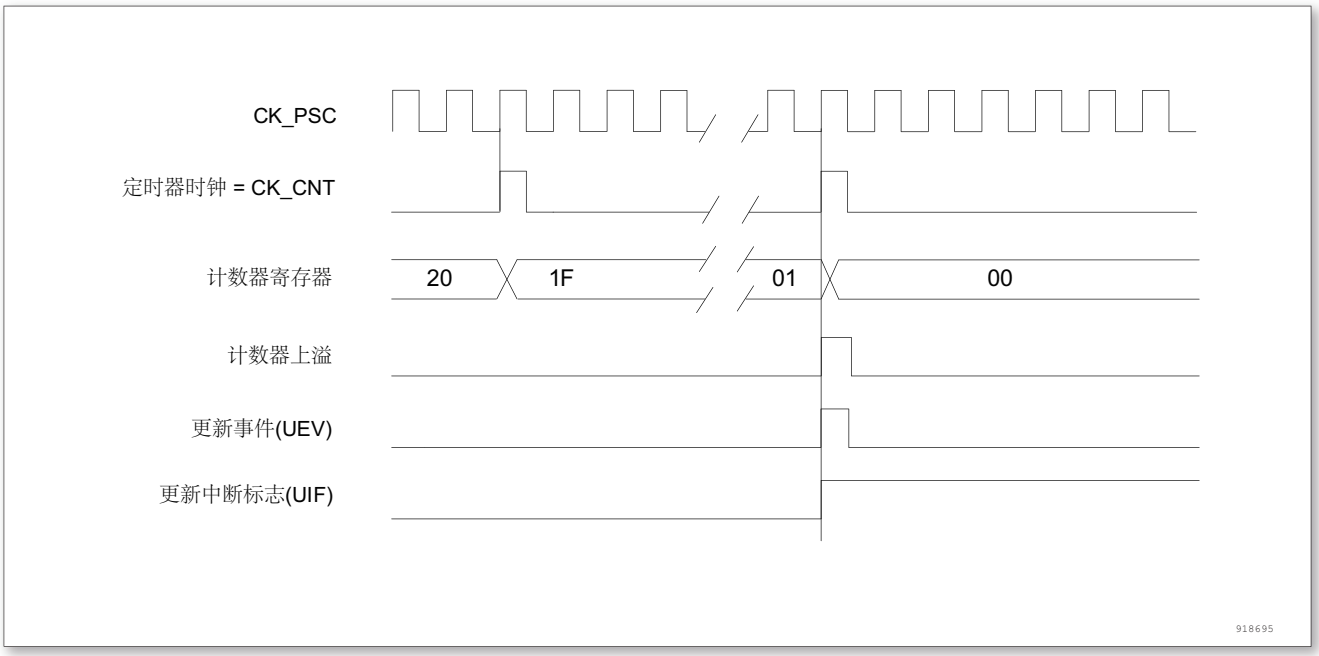


图 45. 计数器时序图，内部时钟分频因子为 N

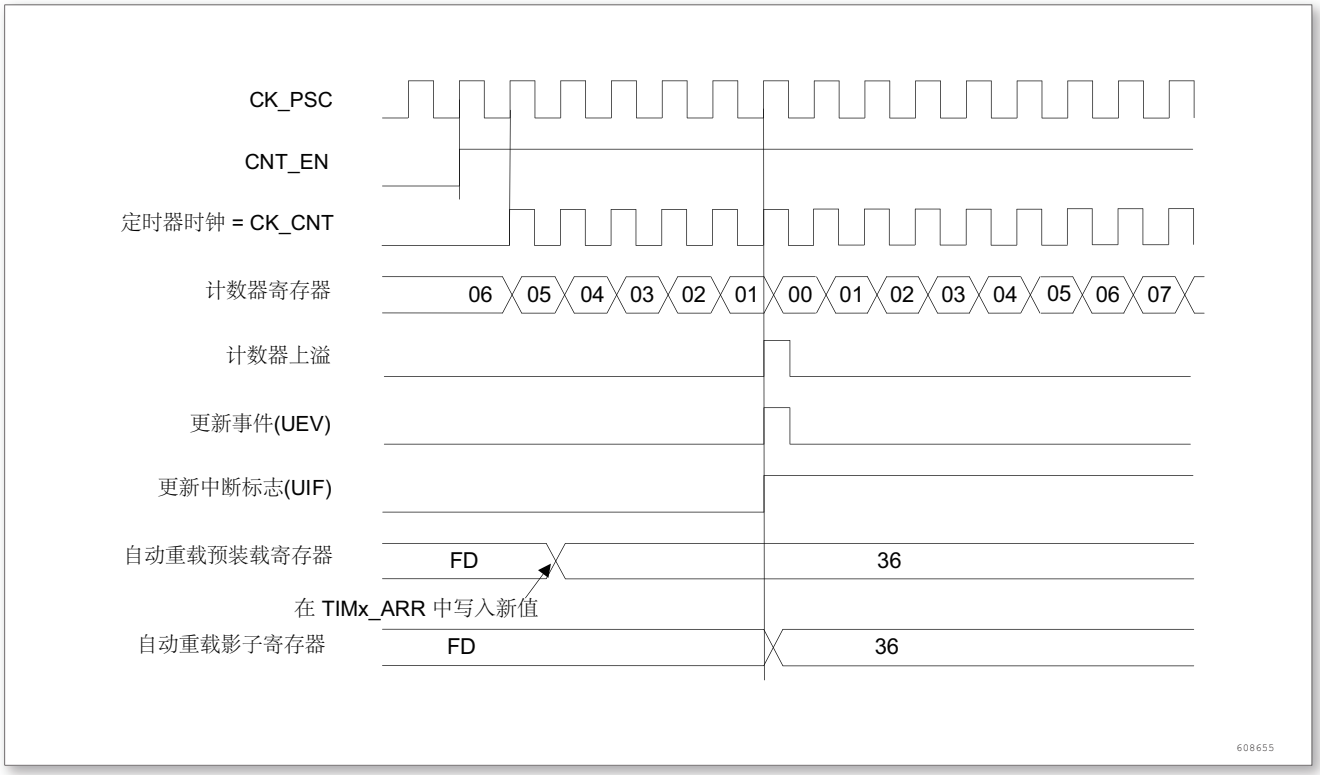


图 46. 计数器时序图，ARPE = 1 时的更新事件 (计数器下溢)

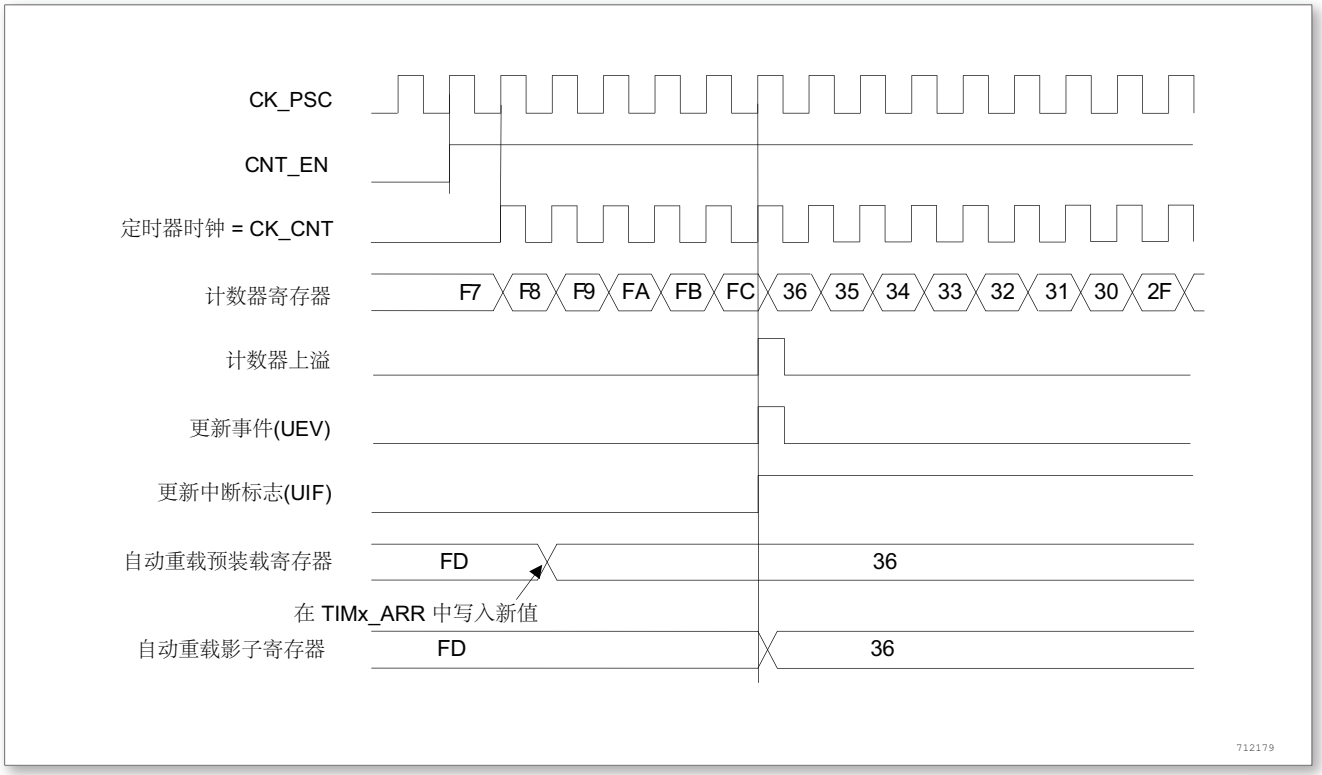


图 47. 计数器时序图，ARPE = 1 时的更新事件 (计数器溢出)

11.3.3 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 48)。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。

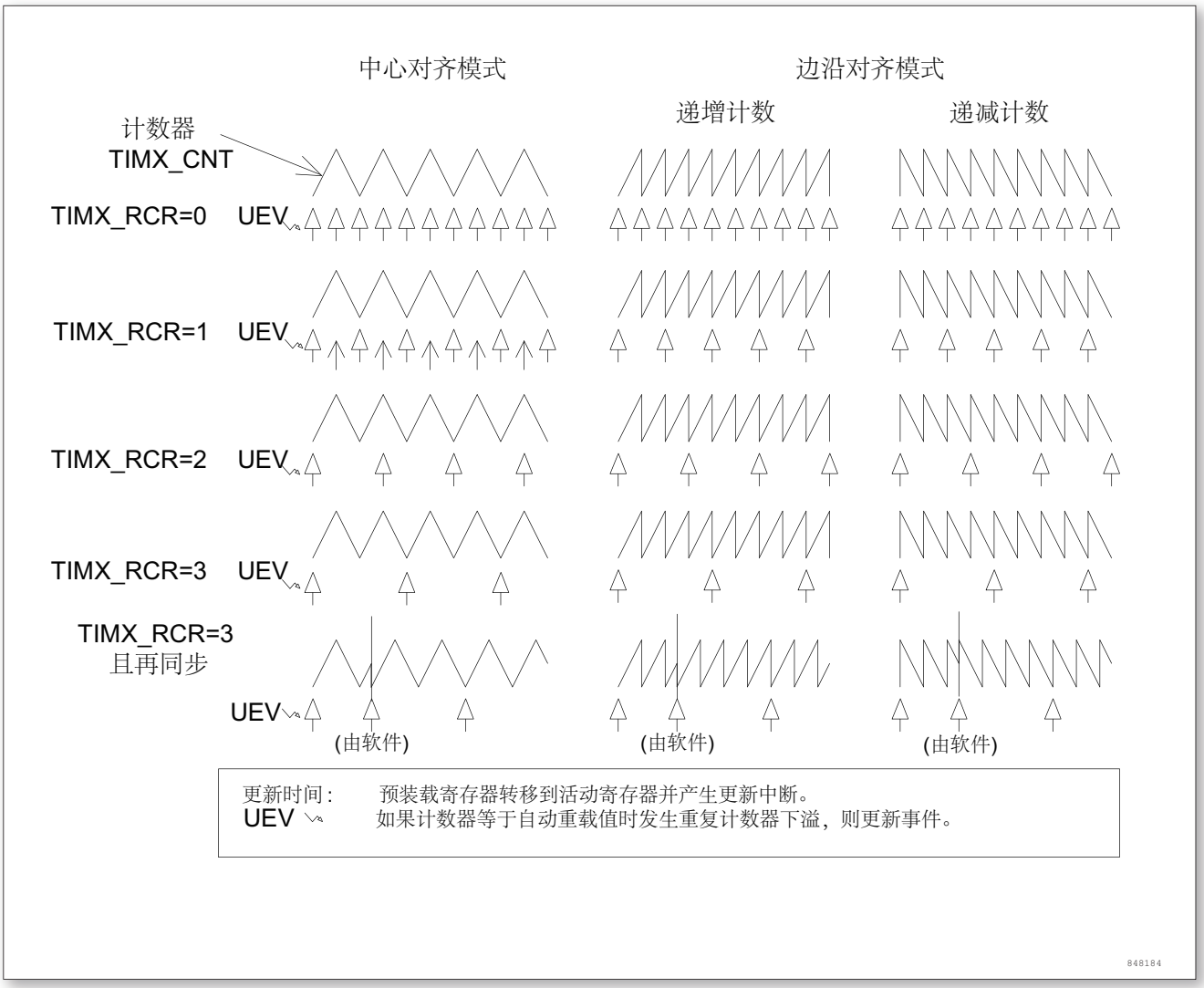


图 48. 不同模式下更新速率的例子，及 TIMx_RCR 的寄存器设置

11.3.4 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。

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)。当 CEN 位被写成 1 时，预分频器的时钟由内部时钟 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。

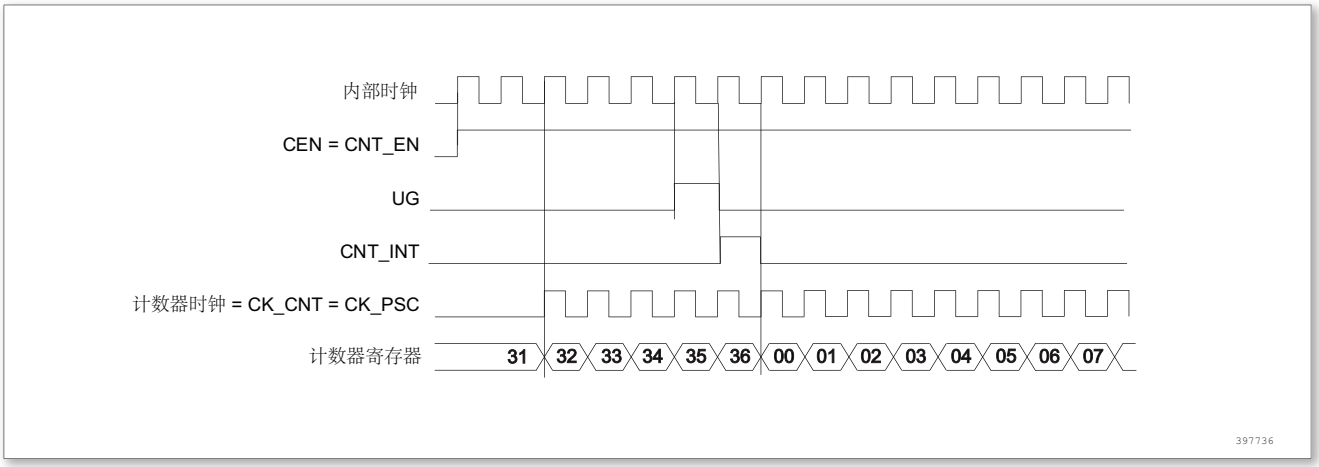


图 49. 一般模式下的控制电路，内部时钟分频因子为 1

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。

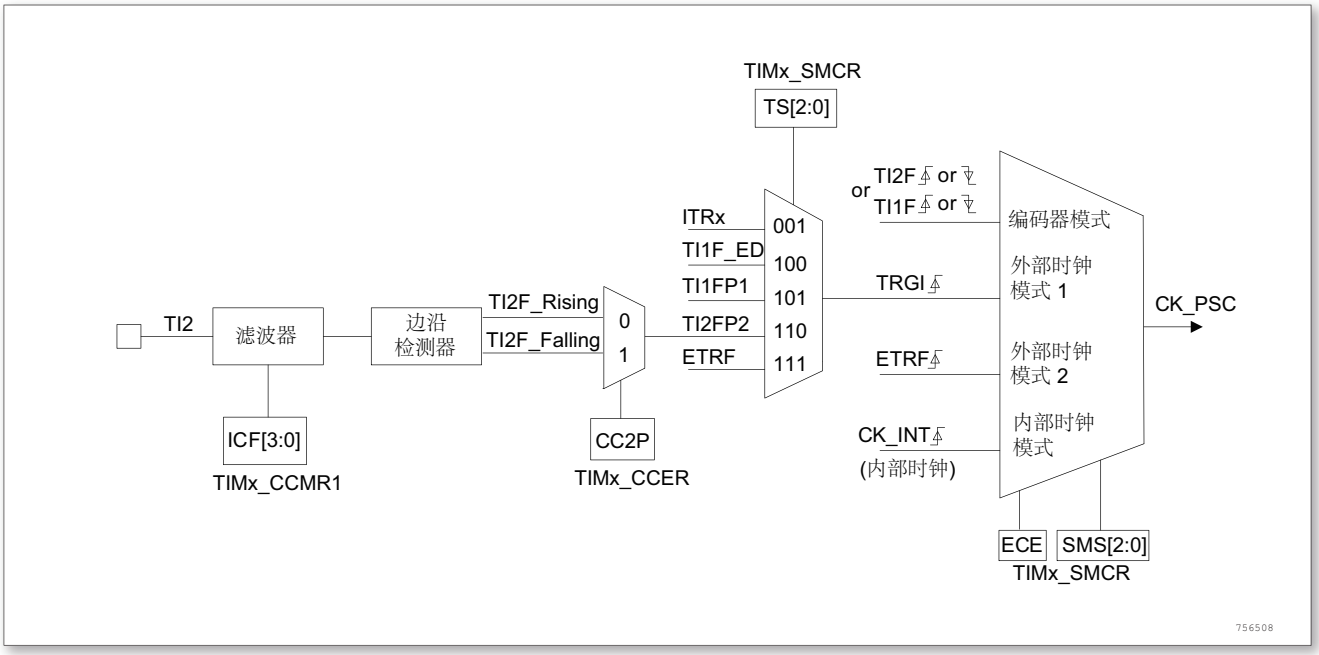


图 50. TI2 外部时钟连接例子

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)。
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。

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。

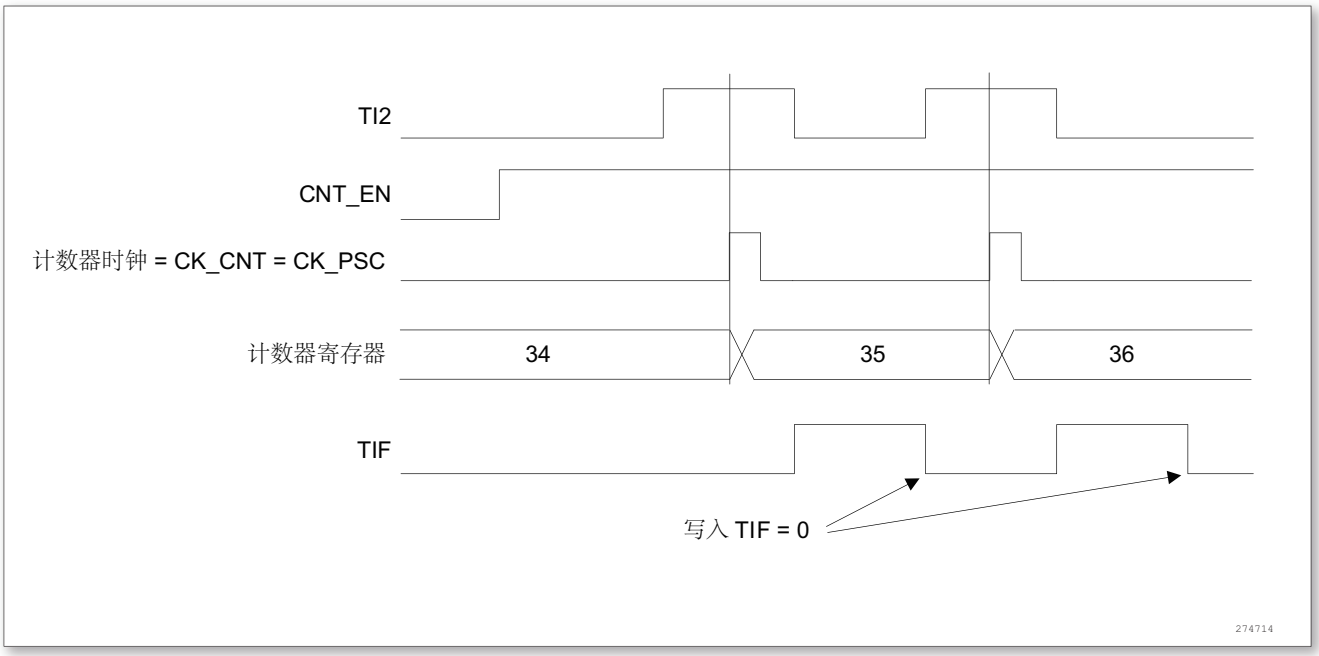


图 51. 外部时钟模式 1 下的控制电路

外部时钟源模式 2

选定此模式的方法为：令 TIMx_SMCR 寄存器中的 ECE = 1。

计数器能够在外部触发 ETR 的每一个上升沿或下降沿计数。

下图是外部触发输入的总体框图：

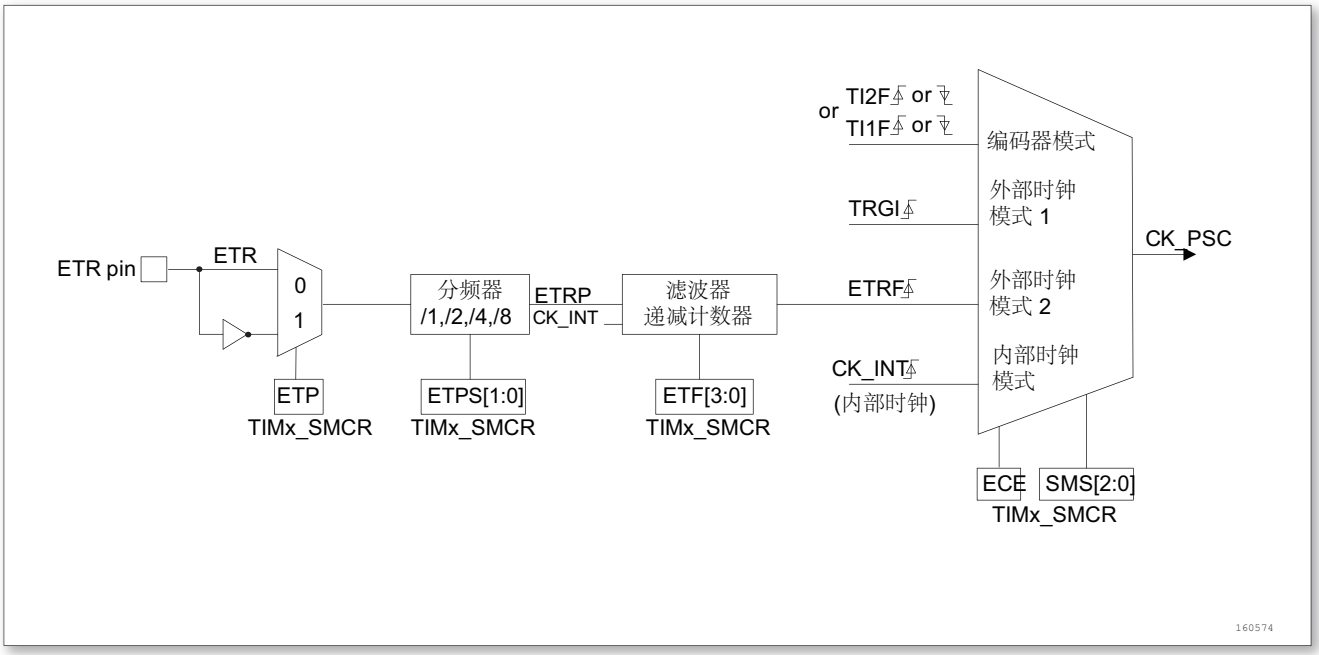


图 52. 外部触发输入框图

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：

- 本例中不需要滤波器，置 TIMx_SMCR 寄存器中的 ETF[3：0] = 0000
- 设置预分频器，置 TIMx_SMCR 寄存器中的 ETPS[1：0] = 01

- 选择 ETR 的上升沿检测，置 TIMx_SMCR 寄存器中的 ETP = 0
- 开启外部时钟模式 2，写 TIMx_SMCR 寄存器中的 ECE = 1
- 启动计数器，写 TIMx_CR1 寄存器中的 CEN = 1

计数器在每 2 个 ETR 上升沿计数一次。

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

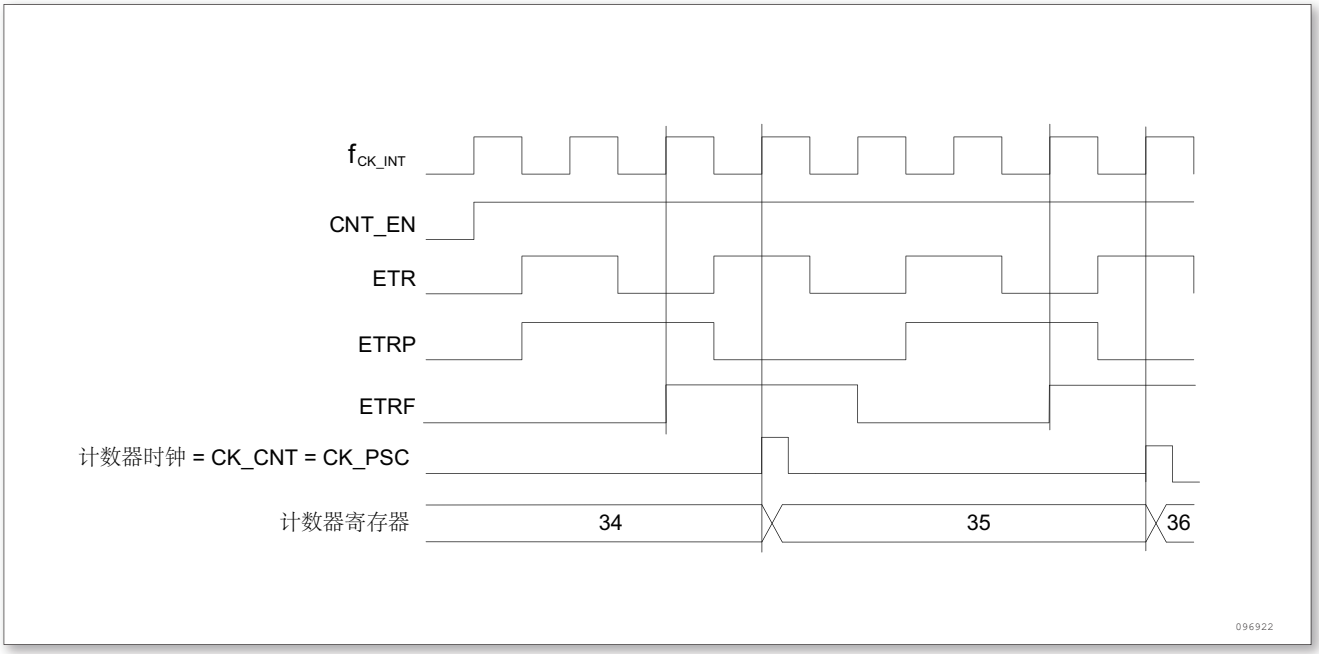


图 53. 外部时钟模式 2 下的控制电路

11.3.5 捕获/比较通道

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

图 54至图 57是一个捕获/比较通道概览。

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

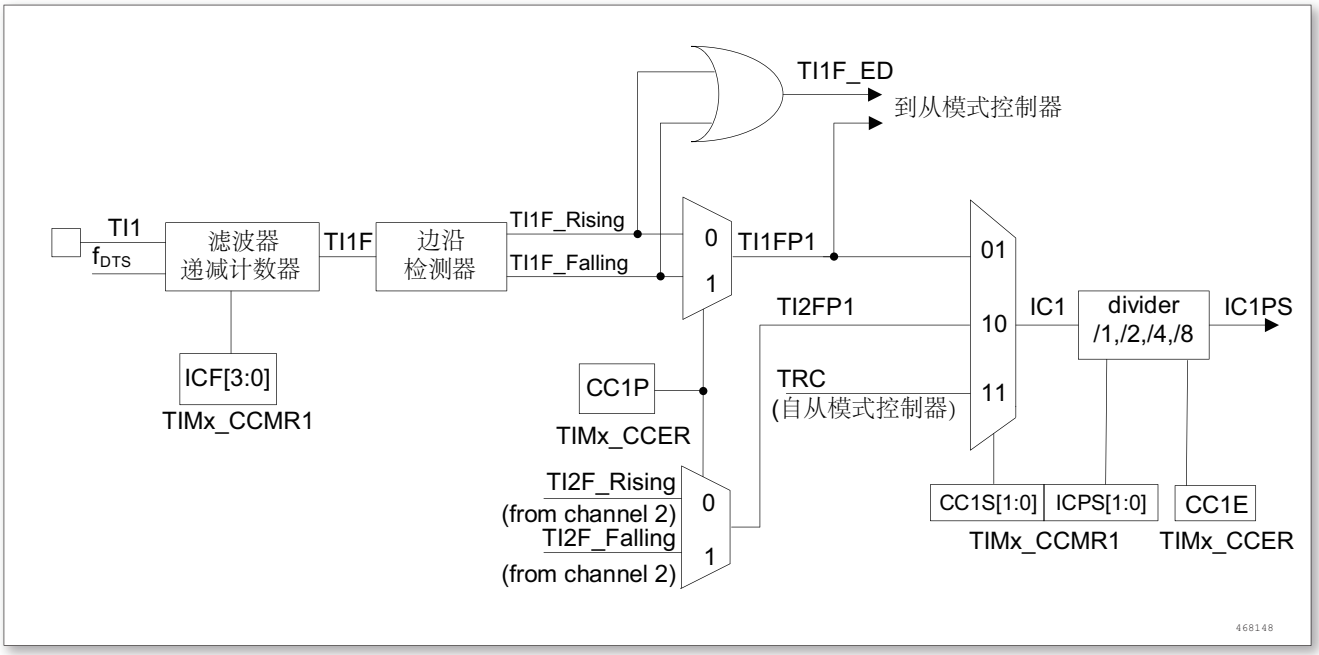


图 54. 捕获/比较通道 (如：通道 1 输入部分)

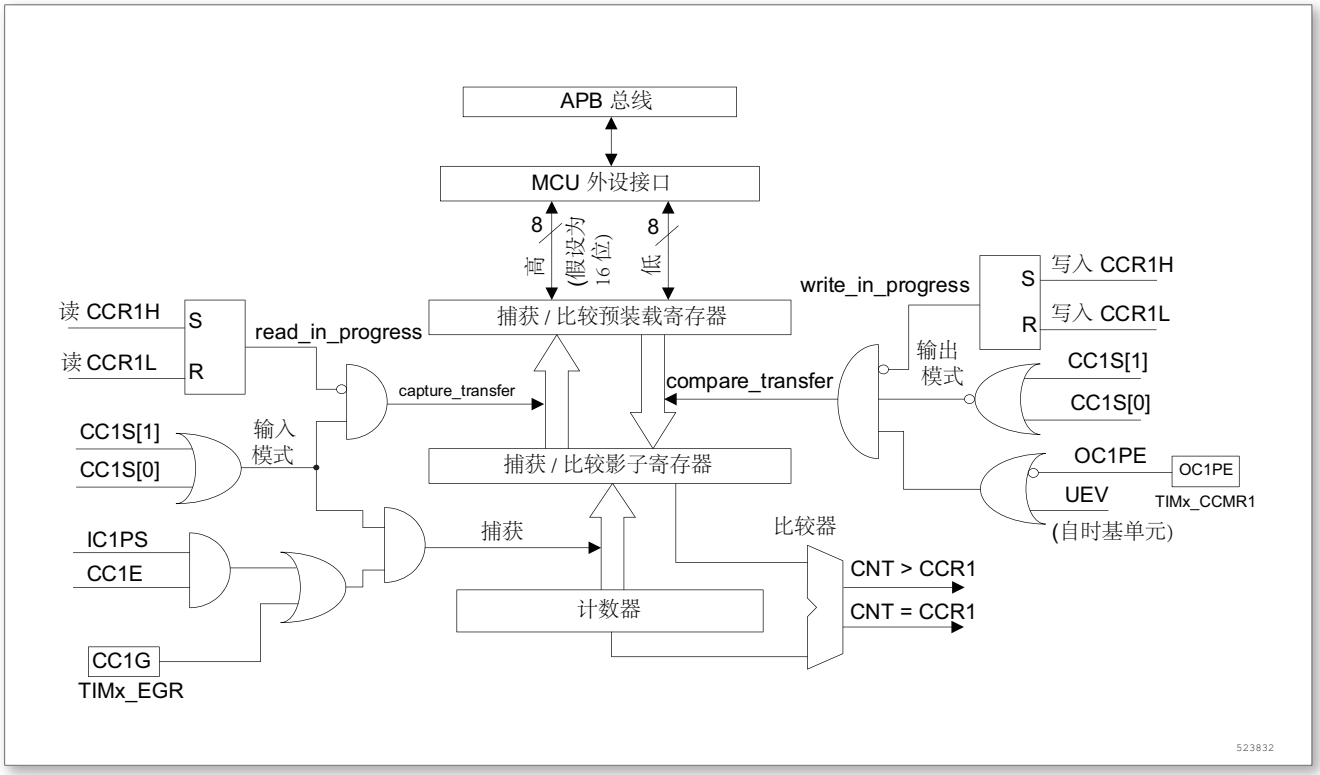


图 55. 捕获/比较通道 1 的主电路

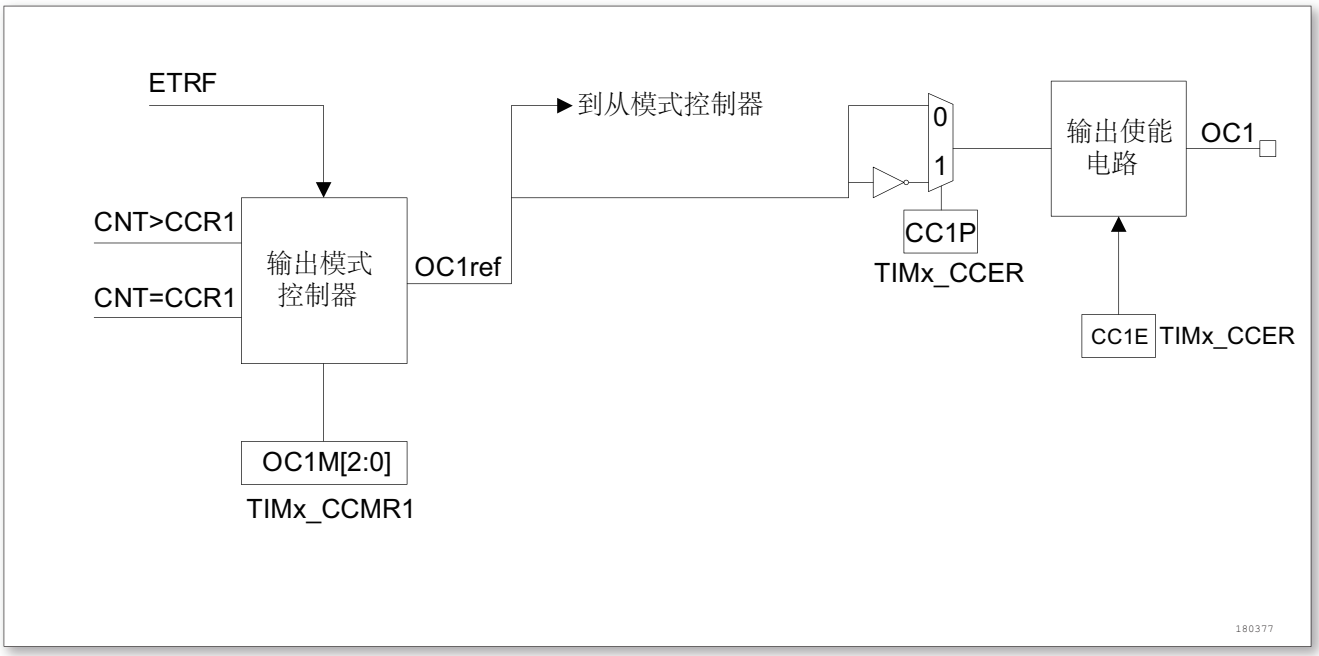


图 56. 捕获/比较通道的输出部分 (通道 1 至 3)

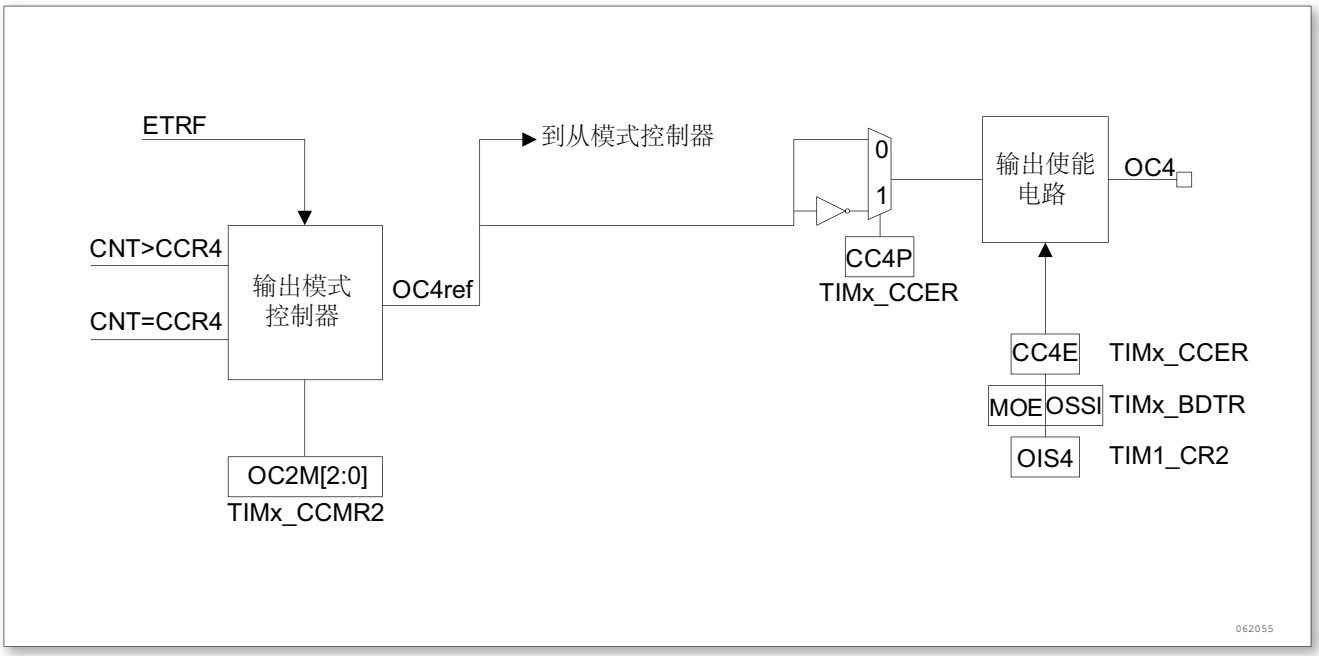


图 57. 捕获/比较通道的输出部分 (通道 4)

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。

11.3.6 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_CCMR1 寄存器中的 CC1S = 01，当 CC1S 不为 00 时，通道被配置为输入，并且 TIMx_CCR1 寄存器变为只读。
- 根据输入信号的特点，配置输入滤波器为所需的带宽 (即输入为 TIx 时，输入滤波器控制位是 TIMx_CCMRx 寄存器中的 ICxF 位)。假设输入信号在最多 5 个时钟周期的时间内抖动，我们须配置滤波器的带宽长于 5 个时钟周期；因此我们可以 (以 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：

- 当产生有效的电平转换时，计数器的值被传送到 TIMx_CCR1 寄存器。
- C1IF 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。

11.3.7 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。

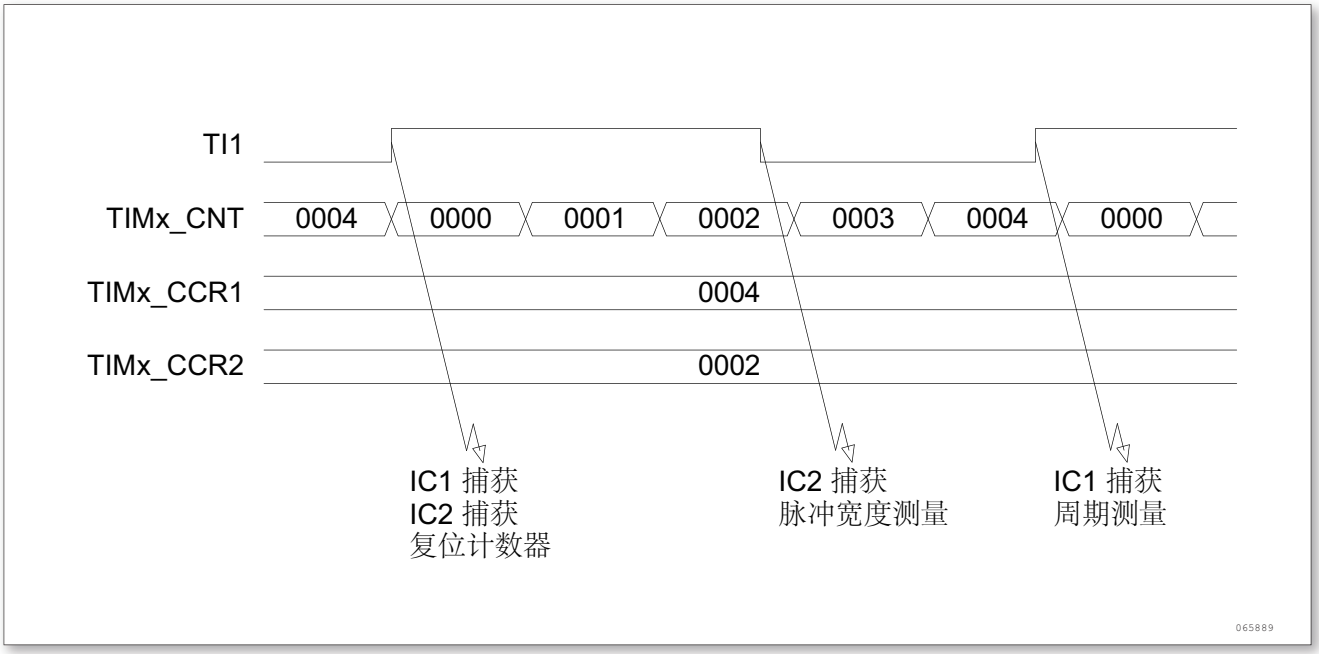


图 58. PWM 输入模式时序

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。

11.3.8 强制输出模式

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

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

例如：CCxP = 0(OCx 高电平有效)，则 OCx 被强置为高电平。

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

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

11.3.9 输出比较模式

此项功能是用来控制一个输出波形或者指示何时一段给定的时间已经到时。

当计数器与捕获/比较寄存器的内容相同时，输出比较功能做如下操作：

- 将输出比较模式 (TIMx_CCMRx 寄存器中的 OCxM 位) 和输出极性 (TIMx_CCER 寄存器中的 CCxP 位) 定义的值输出到对应的管脚上。在比较匹配时，输出管脚可以保持它的电平 (OCxM = 000)、被设置成有效电平 (OCxM = 001)、被设置成无有效电平 (OCxM = 010) 或进行翻转 (OCxM = 011)。

- 设置中断状态寄存器中的标志位 (TIMx_SR 寄存器中的 CCxIF 位)。
- 若设置了相应的中断屏蔽 (TIMx_DIER 寄存器中的 CCxIE 位)，则产生一个中断。
- 若设置了相应的使能位 (TIMx_DIER 寄存器中的 CCxDE 位，TIMx_CR2 寄存器中的 CCDS 位选择 DMA 请求功能)，则产生一个 DMA 请求。

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

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

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

输出比较模式的配置步骤：

- 选择计数器时钟 (内部，外部，预分频器)
- 将相应的数据写入 TIMx_ARR 和 TIMx_CCRx 寄存器中
- 如果要产生一个中断请求，设置 CCxIE 位
- 选择输出模式，例如：
  - 要求计数器与 CCRx 匹配时翻转 OCx 的输出管脚，设置 OCxM = 011
  - 置 OCxPE = 0 禁用预装载寄存器
  - 置 CCxP = 0 选择极性为高电平有效
  - 置 CCxE = 1 使能输出
- 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。

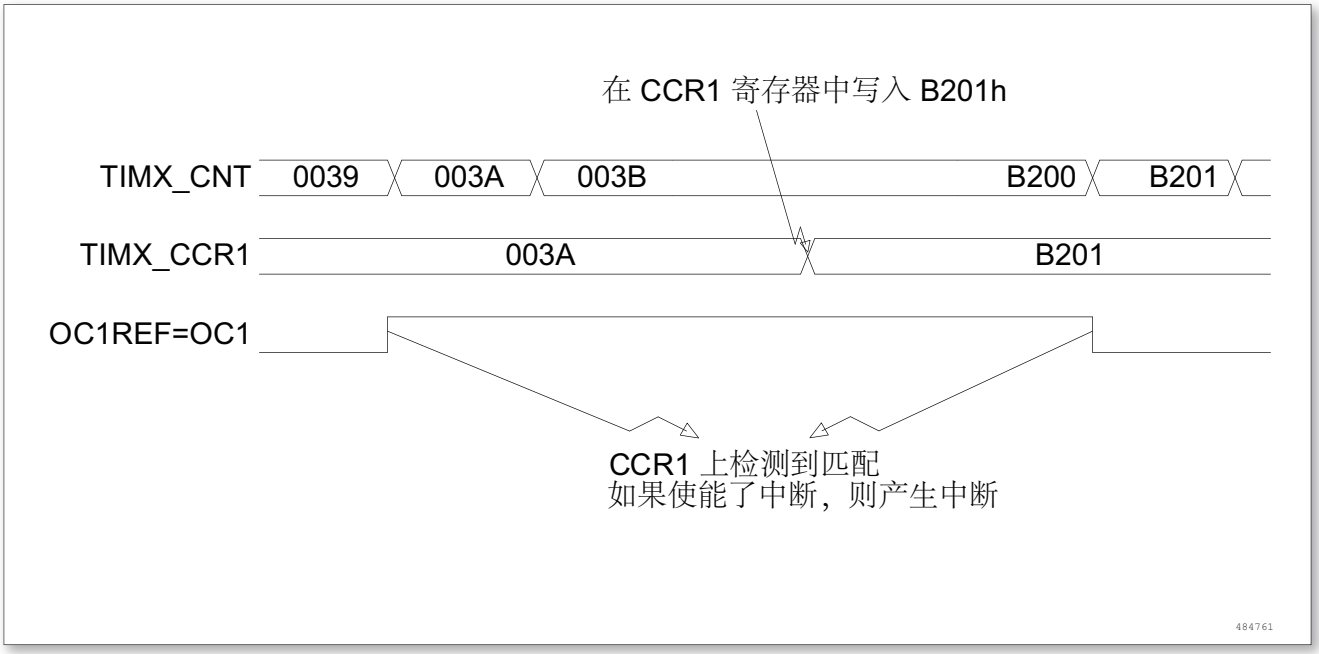


图 59. 输出比较模式，翻转 OC1

11.3.10 PWM 模式

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

在 TIMx_CCMRx 寄存器中的 OCxM 位写入‘110’(PWM 模式 1) 或‘111’(PWM 模式 2)，能够独立地设置每个 OCx 输出通道产生一路 PWM。必须通过设置 TIMx_CCMRx 寄存器的 OCxPE 位使能相应的预装载寄存器，最后还要设置 TIMx_CR1 寄存器的 ARPE 位使能自动重装载的预装载寄存器 (在向上计数或中心对称模式中)。

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

OCx 的极性可以通过软件在 TIMx_CCER 寄存器中的 CCxP 位设置，它可以设置为高电平有效或低电平有效。OCx 的输出使能通过 (TIMx_CCER 和 TIMx_BDTR 寄存器中) CCxE、CCxNE、MO E、OSSI 和 OSSR 位的组合控制。详见 TIMx_CCER 寄存器的描述。

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

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

PWM 边沿对齐模式

向上计数配置

当 TIMx_CR1 寄存器中的 DIR 位为低的时候执行向上计数。参看小节 11.3.2。

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，否则低。如果 TIMx_CCRx 中的比较值大于自动重装载值 (TIMx_ARR)，则 OCxREF 保持为‘1’。如果比较值为 0，则 OCxREF 保持为‘0’。图 60为 TIMx_ARR = 8 时边沿对齐的 PWM 波形实例。

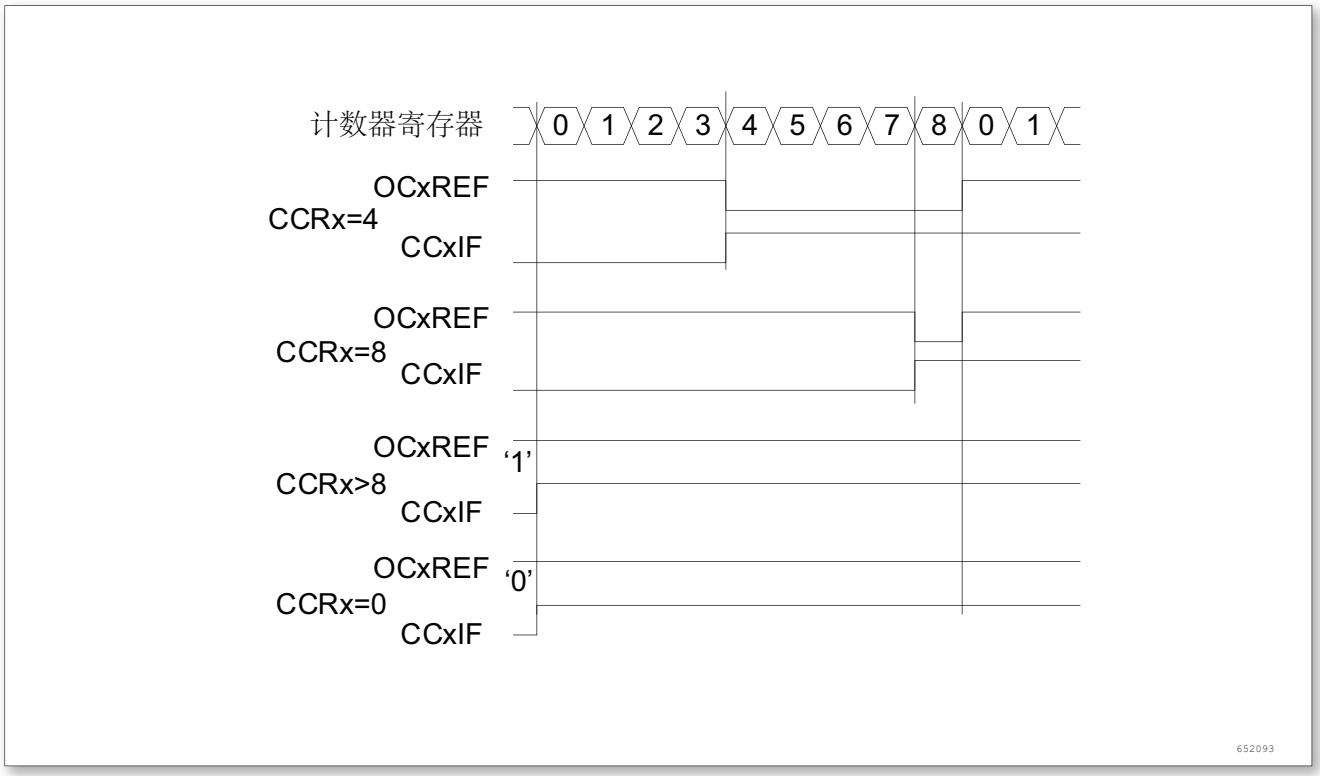


图 60. 边沿对齐的 PWM 波形 (ARR = 8)

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。参看小节 11.3.2。

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

PWM 中央对齐模式

当 TIMx_CR1 寄存器中的 CMS 位不为‘00’时为中央对齐模式 (所有其他的配置对 OCxREF/OCx 信号都有相同的作用)。根据不同的 CMS 位的设置，比较标志可以在计数器向上计数时被置 1、在计数器向下计数时被置 1、或在计数器向上和向下计数时被置‘1’。TIMx_CR1 寄存器中的计数方向位 (DIR) 由硬件更新，不要用软件修改它。参看小节 11.3.2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

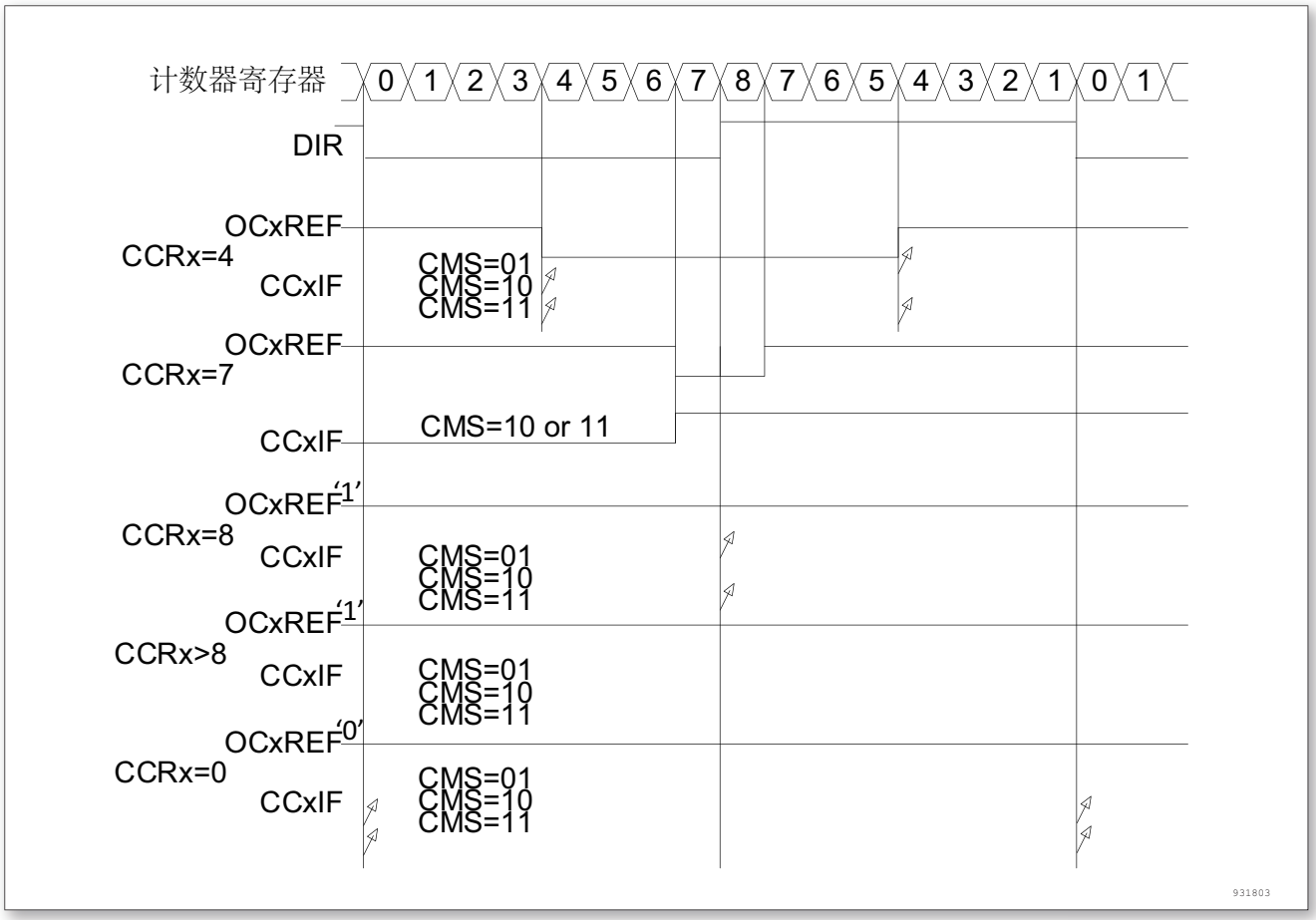


图 61. 中央对齐的 PWM 波形 (APR = 8)

使用中央对齐模式的提示：

- 进入中央对齐模式时，使用当前的上/下计数配置；这就意味着计数器向上还是向下计数

取决于 TIMx_CR1 寄存器中 DIR 位的当前值。此外，软件不能同时修改 DIR 和 CMS 位。

- 不推荐当运行在中央对齐模式时改写计数器，因为会产生不可预知的结果。特别地：
  - 如果写入计数器的值大于自动重加载的值 (TIMx_CNT > TIMx_ARR)，则方向不会被更新
  - 例如，如果计数器正在向上计数，它就会继续向上计数
  - 如果将 0 或者 TIMx_ARR 的值写入计数器，方向被更新，但不产生更新事件 UEV
- 使用中央对齐模式最保险的方法,就是在启动计数器之前产生一个软件更新 (设置 TIMx_EGR 位中的 UG 位)，不要在计数进行过程中修改计数器的值。

11.3.11 互补输出和死区插入

高级控制定时器 (TIM1) 能够输出两路互补信号，并且能够管理输出的瞬时关断和接通。这段时间通常被称为死区，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 50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)。

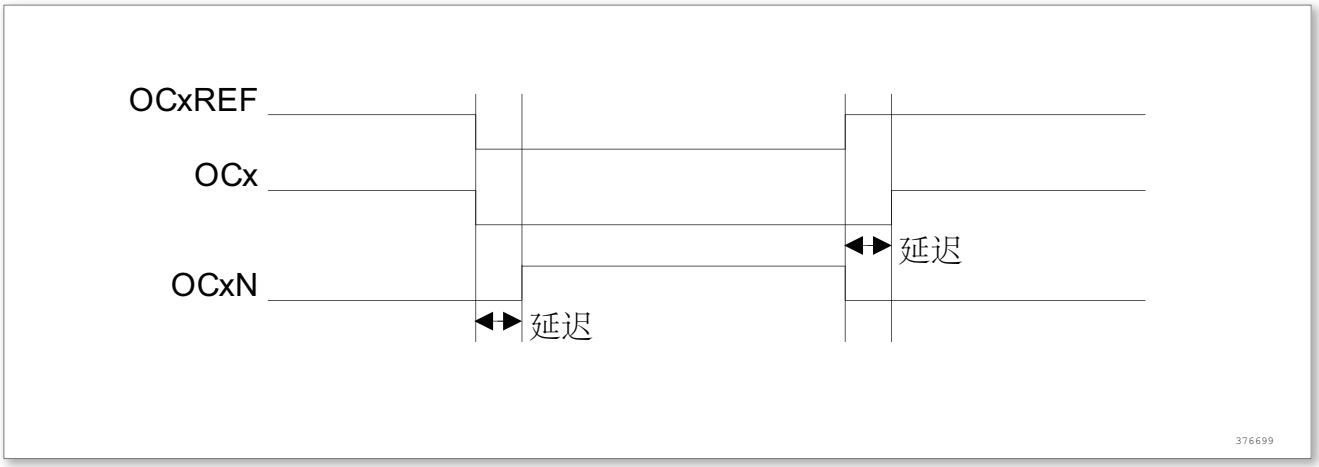


图 62. 带死区插入的互补输出

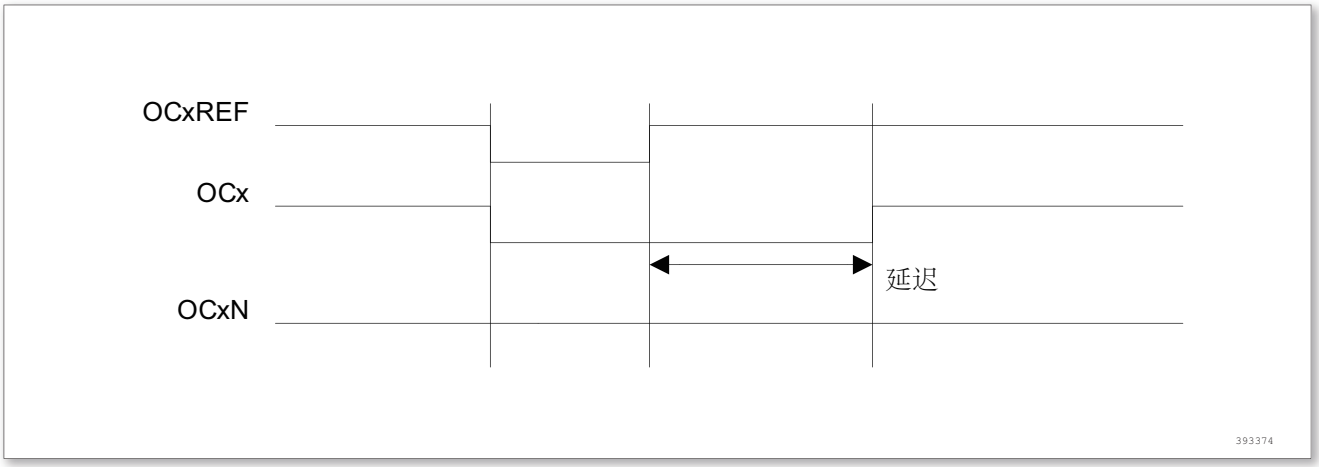


图 63. 死区波形延迟大于负脉冲

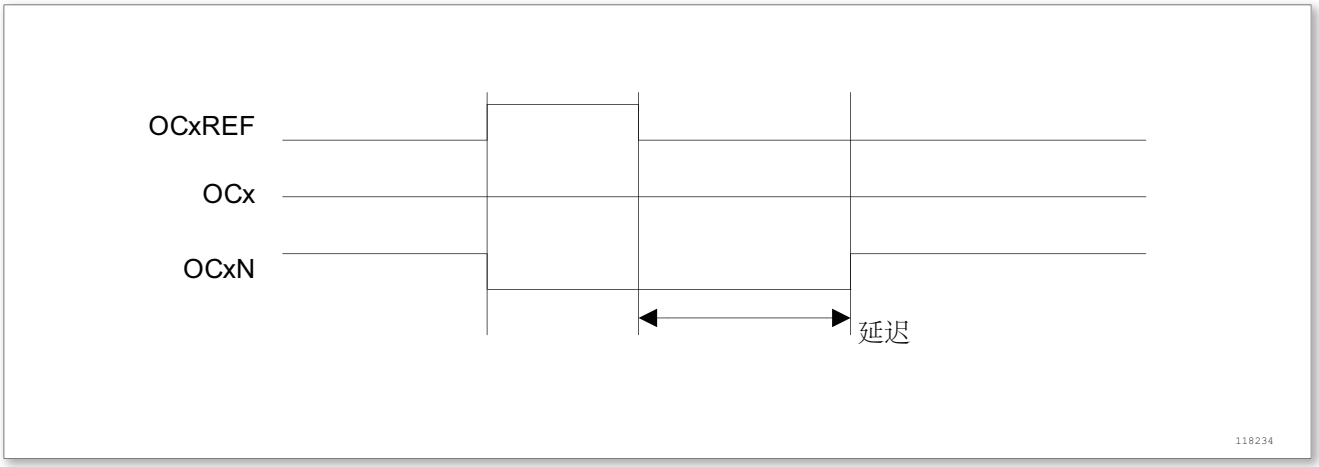


图 64. 死区波形延迟大于正脉冲

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。详见小节 11.4.18中的延时计算。

重定向 OCxREF 到 OCx 或 OCxN

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

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

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

11.3.12 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。详见寄存器表中带刹车功能的互补输出通道 OCx 和 OCxN 的控制位。

刹车源既可以是刹车输入管脚又可以是一个时钟失败事件。时钟失败事件由复位时钟控制器中的时钟安全系统产生。

系统复位后，刹车电路被禁止，MOE 位为低。设置 TIMx_BDTR 寄存器中的 BKE 位可以使能刹车功能。刹车输入信号的极性可以通过配置同一个寄存器中的 BKP 位选择。BKE 和 BKP 可以被同时修改。

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

当发生刹车时 (在刹车输入端出现选定的电平)，有下述动作：

- MOE 位被异步地清除，将输出置于无效状态、空闲状态或者复位状态 (由 OSSI 位选择)。这个特性在 MCU 的振荡器关闭时依然有效。
- 当 MOE = 0 时，每一个输出通道输出的电平由 TIMx_CR2 寄存器中的 OISx 位决定。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；或当 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，参看小节 11.4.8。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：

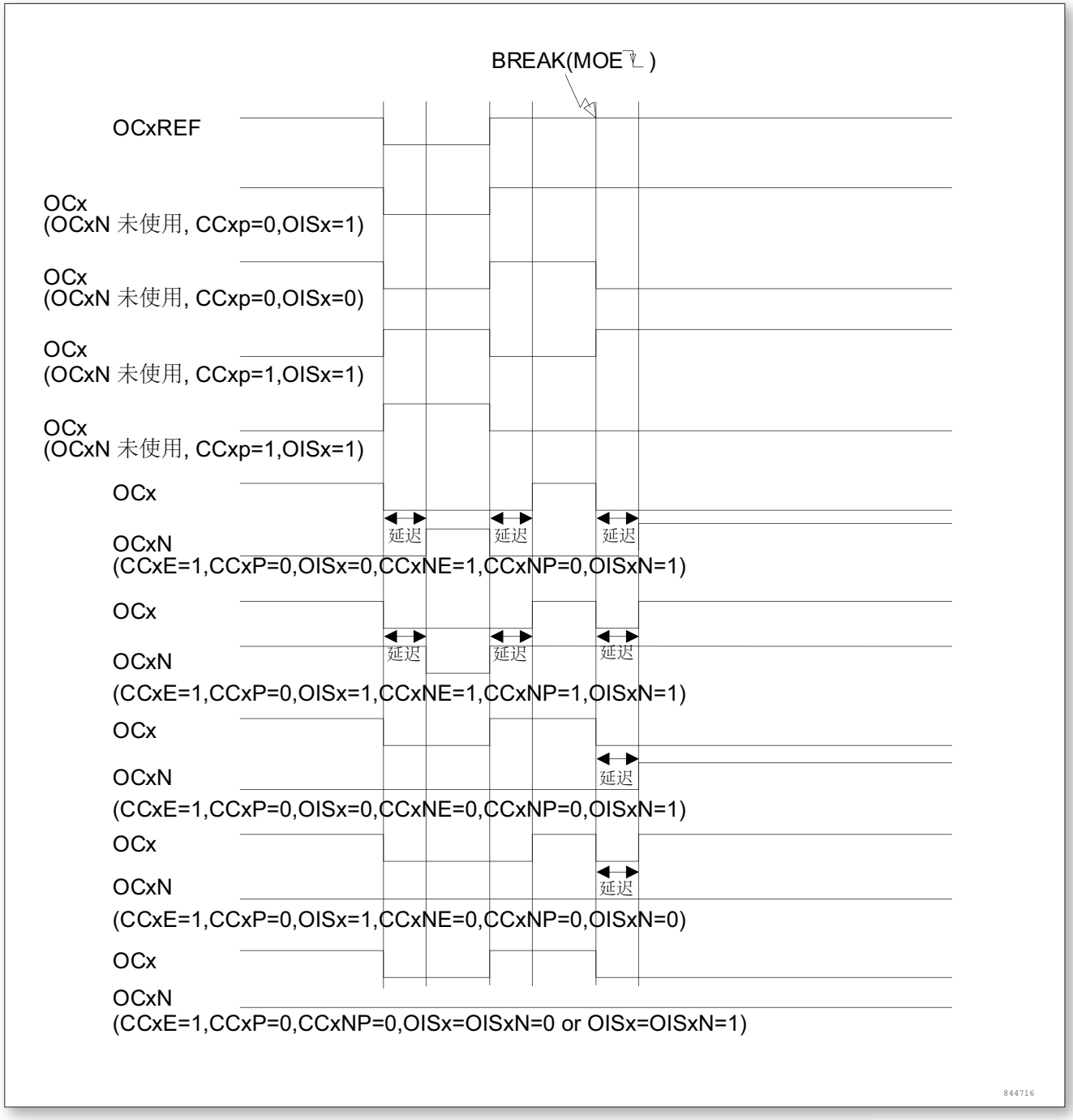


图 65. 响应刹车的输出

11.3.13 在外部事件时清除 OCxREF 信号

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

该功能只能用于输出比较和 PWM 模式，而不能用于强置模式。

例如，OCxREF 信号可以连到一个外部输入。这时，ETR 必须配置如下：

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

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

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

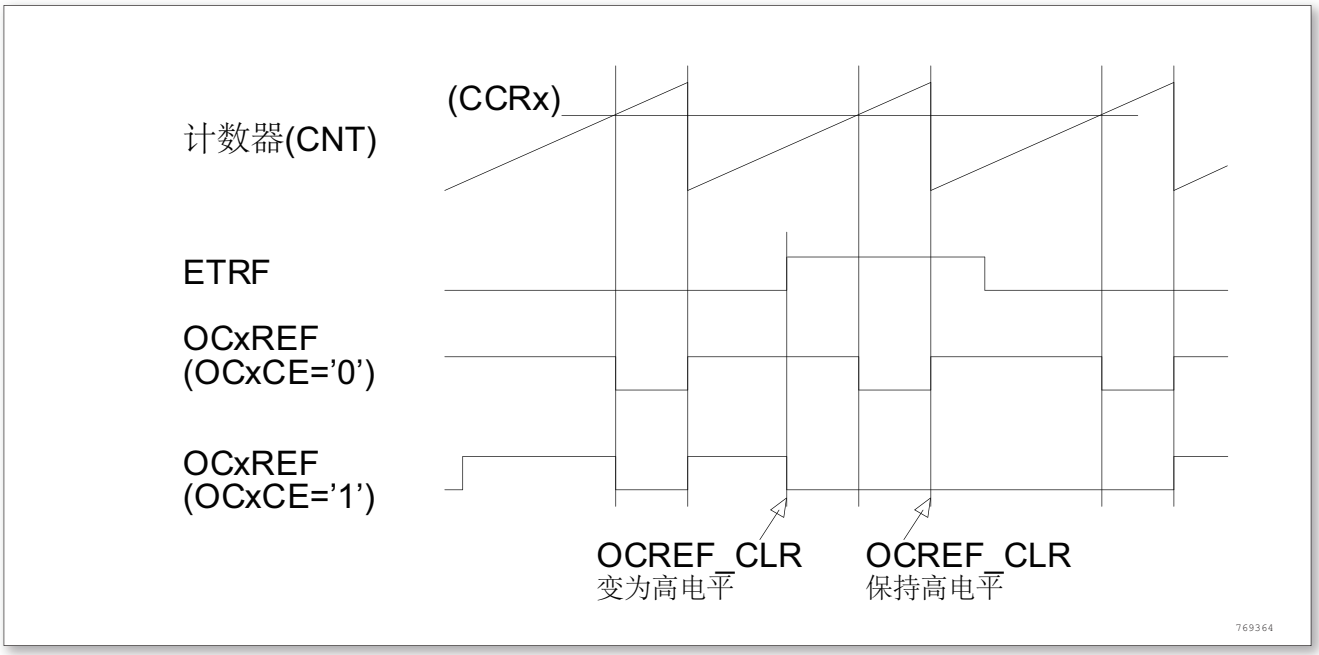


图 66. 清除 TIMx 的 OCxREF

11.3.14 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 (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。

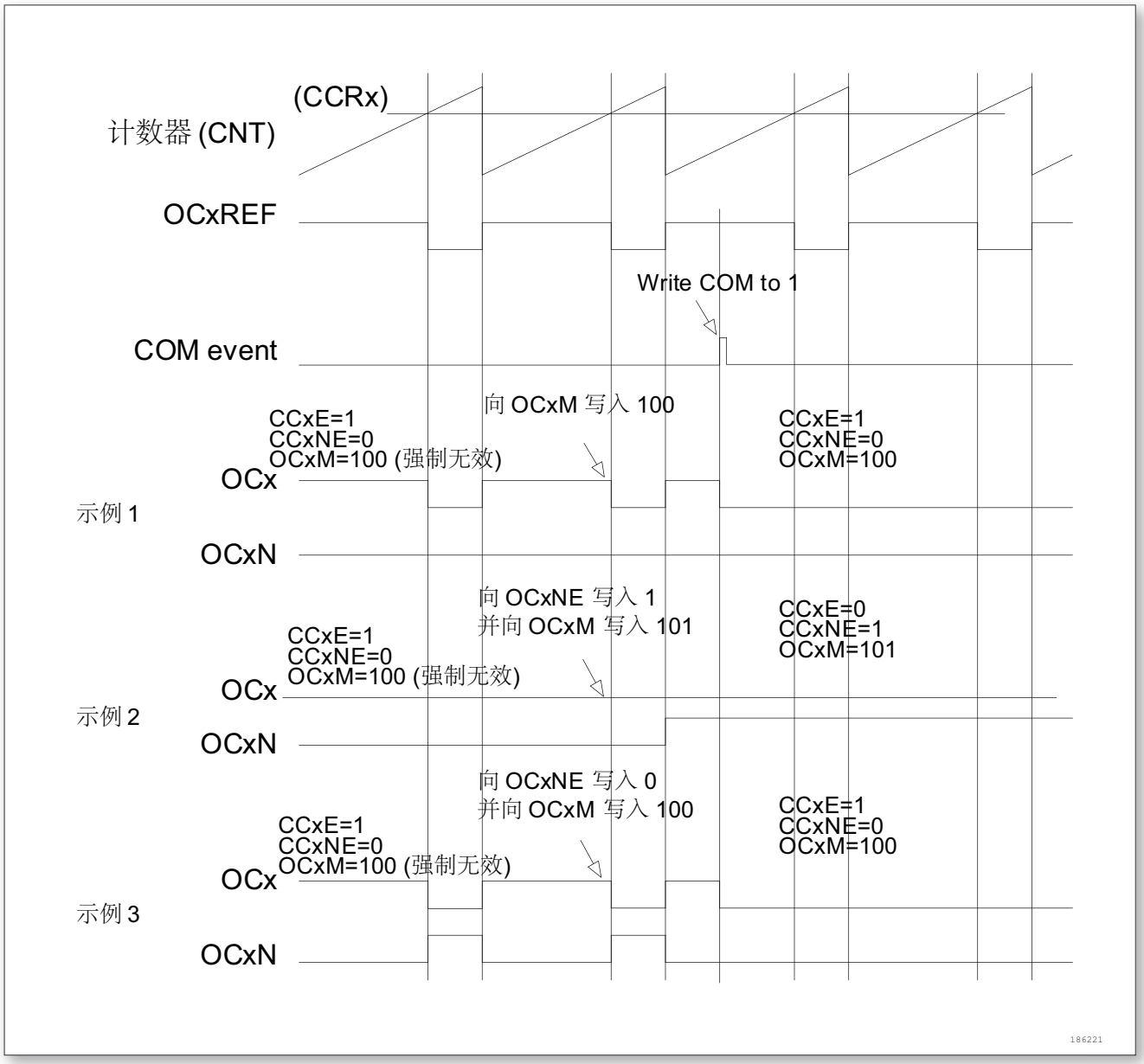


图 67. 产生六步 PWM，使用 COM 的例子 (OSSR = 1)

11.3.15 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

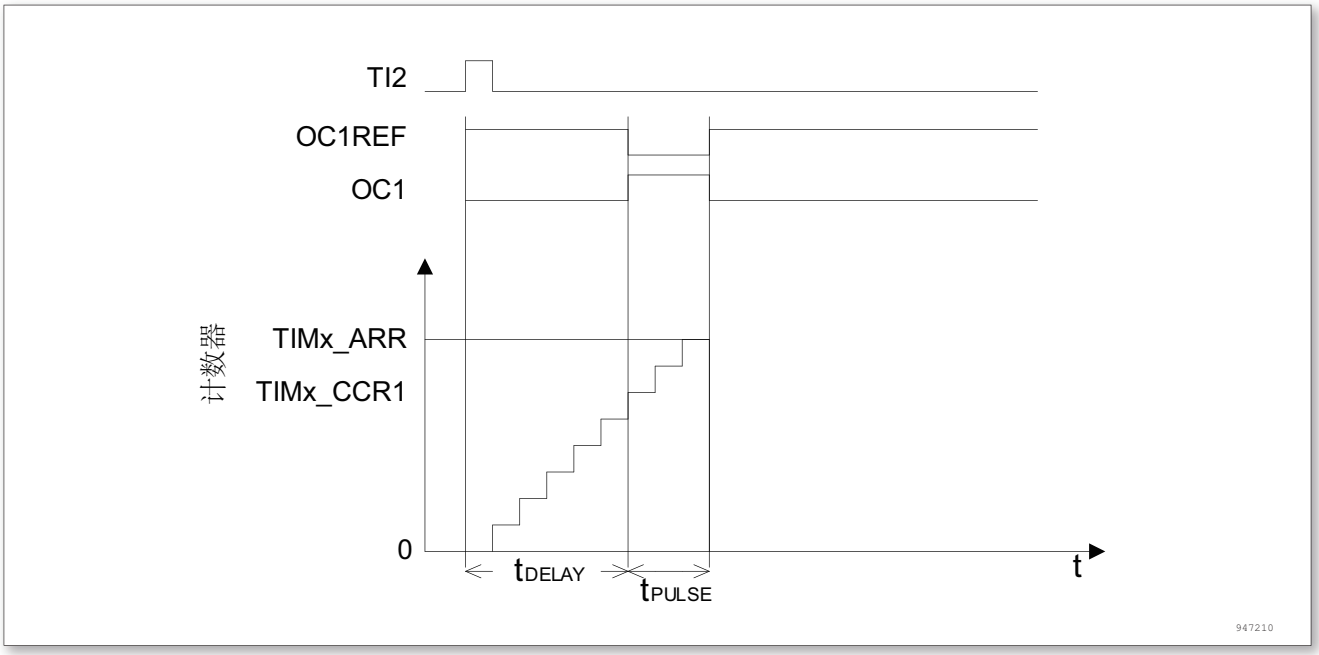


图 68. 单脉冲模式的例子

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

假定 TI2FP2 作为触发 1：

- 置 TIMx_CCMR1 寄存器中的 CC2S = 01，把 TI2FP2 映像到 TI2。
- 置 TIMx_CCER 寄存器中的 CC2P = 0，使 TI2FP2 能够检测上升沿。
- 置 TIMx_SMCR 寄存器中的 TS = 110，TI2FP2 作为从模式控制器的触发 (TRGI)。
- 置 TIMx_SMCR 寄存器中的 SMS = 110(触发模式)，TI2FP2 被用来启动计数器。

OPM 的波形由写入比较寄存器的数值决定 (要考虑时钟频率和计数器预分频器)

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

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

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

特殊情况：OCx 快速使能：

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

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

OCxREF(和 OCx) 直接响应激励而不再依赖比较的结果，输出的波形与比较匹配时的波形一样。OCxFE 只在通道配置为 PWM1 和 PWM2 模式时起作用。

11.3.16 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 47，假定计数器已经启动 (TIMx_CR1 寄存器中的 CEN = 1)，则计数器由每次在 TI1FP1 或 TI2FP2 上的有效跳变驱动。TI1FP1 和 TI2FP2 是 TI1 和 TI2 在通过输入滤波器和极性控制后的信号；如果没有滤波和变相，则 TI1FP1 = TI1；如果没有滤波和变相，则 TI2FP2 = TI2。根据两个输入信号的跳变顺序，产生了计数脉冲和方向信号。依据两个输入信号的跳变顺序，计数器向上或向下计数，同时硬件对 TIMx_CR1 寄存器的 DIR 位进行相应的设置。不管计数器是依靠 TI1 计数、依靠 TI2 计数或者同时依靠 TI1 和 TI2 计数，在任一输入端 (TI1 或者 TI2) 的跳变都会重新计算 DIR 位。

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

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

表 47. 计数方向与编码器信号的关系

有效边沿	相对信号的电平 (TITFP1 对应 TI2，TI2FP2 对应 TI1)	TI1FP1 信号		TI2FP2 信号	
		上升	下降	上升	下降
仅在 TI1 计数	高	向下计数	向上计数	不计数	不计数
	低	向上计数	向下计数	不计数	不计数
仅在 TI2 计数	高	不计数	不计数	向上计数	向下计数
	低	不计数	不计数	向下计数	向上计数
在 TI1 和 TI2 上计数	高	向下计数	向上计数	向上计数	向下计数
	低	向上计数	向下计数	向下计数	向上计数

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

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

- CC1S = ‘01’(TIMx_CCMR1 寄存器，IC1FP1 映射到 TI1)

- CC2S = ‘01’(TIMx_CCMR2 寄存器，IC2FP2 映射到 TI2)
- CC1P = ‘0’(TIMx_CCER 寄存器，IC1FP1 不反相，IC1FP1=TI1)
- C2P = ‘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)

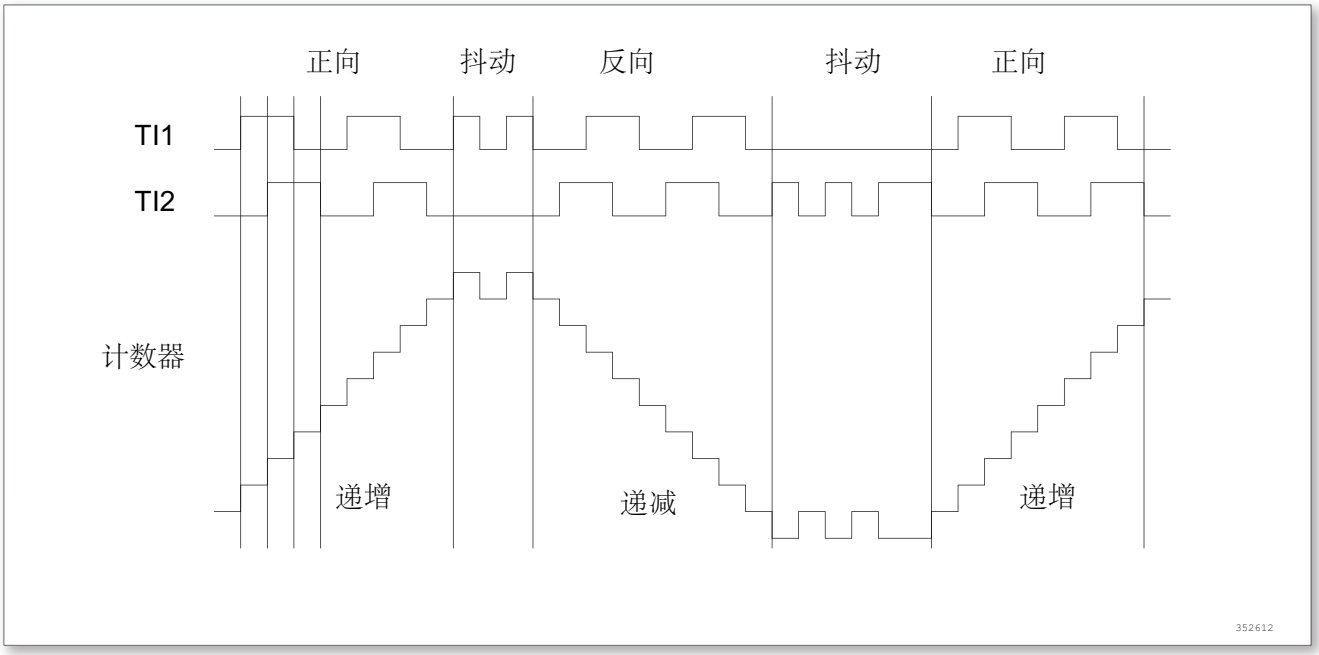


图 69. 编码器模式下的计数器操作实例

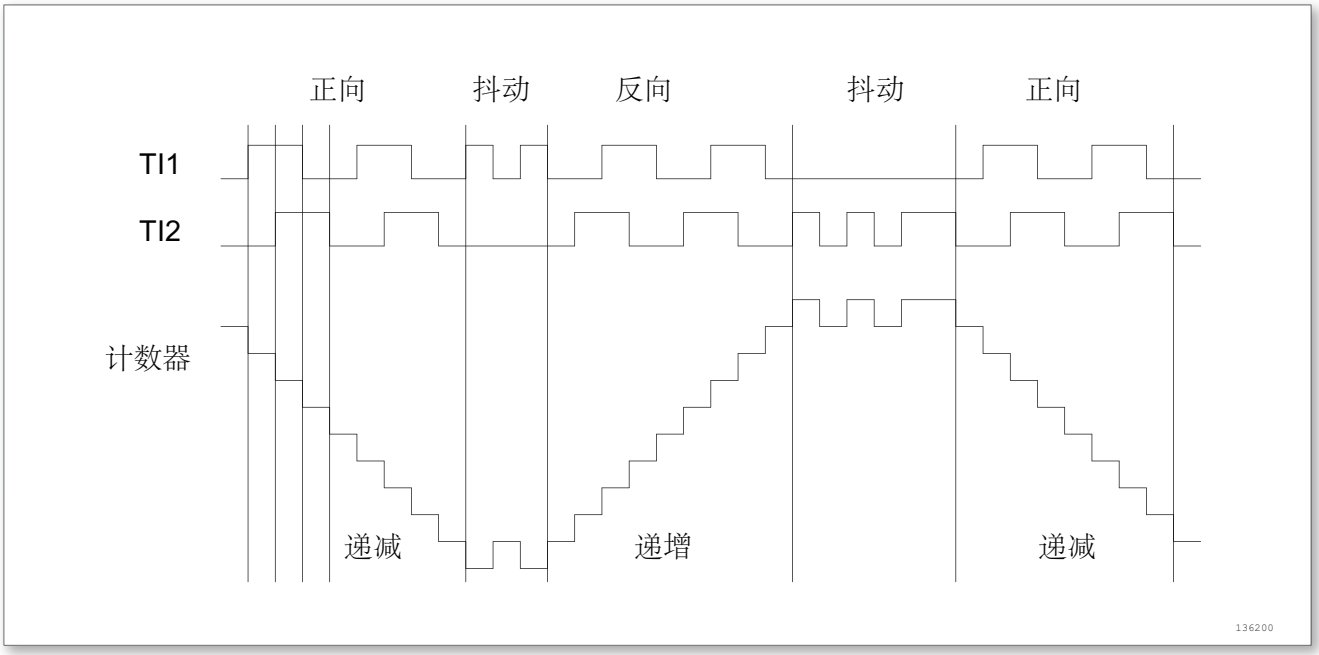


图 70. IC1FP1 反相的编码器接口模式实例

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

请求来读取它的值。

11.3.17 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。小节 11.3.18给出了此特性用于连接霍尔传感器的例子。

11.3.18 与霍尔传感器的接口

使用高级控制定时器 (TIM1) 产生 PWM 信号驱动马达时，可以用另一个通用 TIMx(TIM2、TIM3 或 TIM4) 定时器作为“接口定时器”来连接霍尔传感器，见图 71，3 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 54)。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 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。

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：

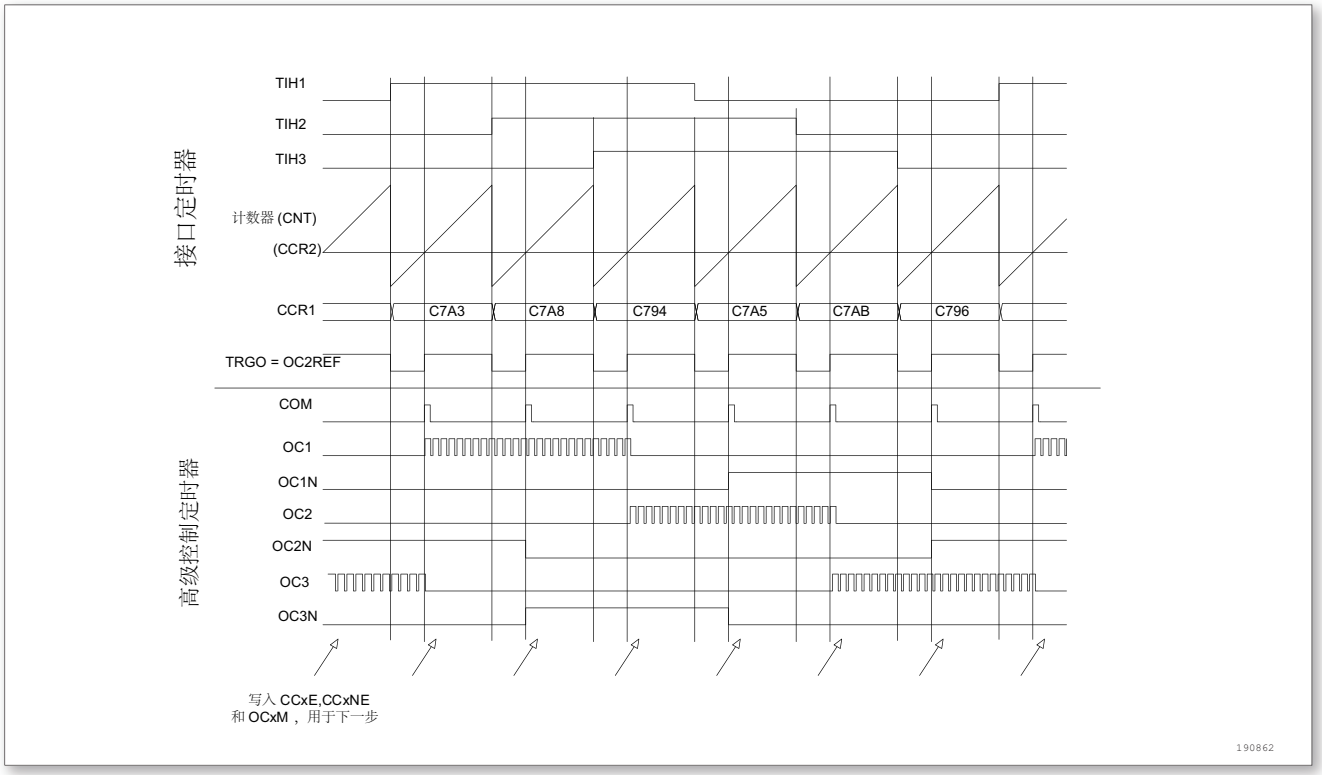


图 71. 霍尔传感器接口的实例

11.3.19 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。

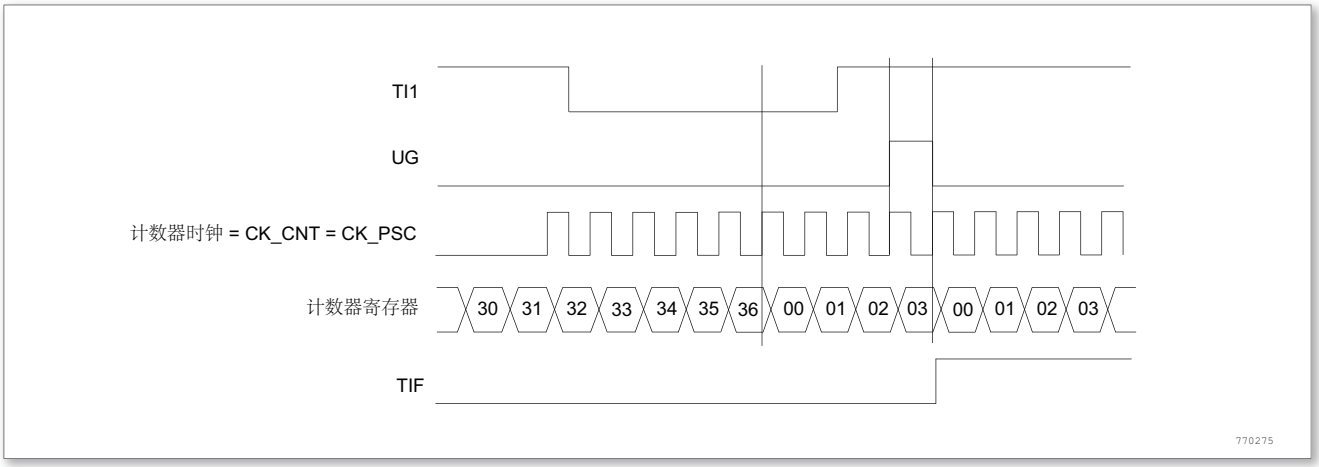


图 72. 复位模式下的控制电路

从模式：门控模式

计数器的使能依赖于选中的输入端的电平。

在如下的例子中，计数器只在 TI1 为低时向上计数：

- 配置通道 1 以检测 TI1 上的低电平。配置输入滤波器带宽 (本例中，不需要滤波，所以保持 IC1F = 0000)。触发操作中不使用捕获预分频器，所以不需要配置。CC1S 位用于选择输入捕获源，置 TIMx_CCMR1 寄存器中 CC1S = 01 。置 TIMx_CCER 寄存器中 CC1P = 1 以确定极性 (只检测低电平)。
- 置 TIMx_SMCR 寄存器中 SMS = 101，配置定时器为门控模式；置 TIMx_SMCR 寄存器中 TS = 101，选择 TI1 作为输入源。
- 置 TIMx_CR1 寄存器中 CEN = 1，启动计数器。在门控模式下，如果 CEN = 0，则计数器不能启动，不论触发输入电平如何。

只要 TI1 为低，计数器开始依据内部时钟计数，当 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。

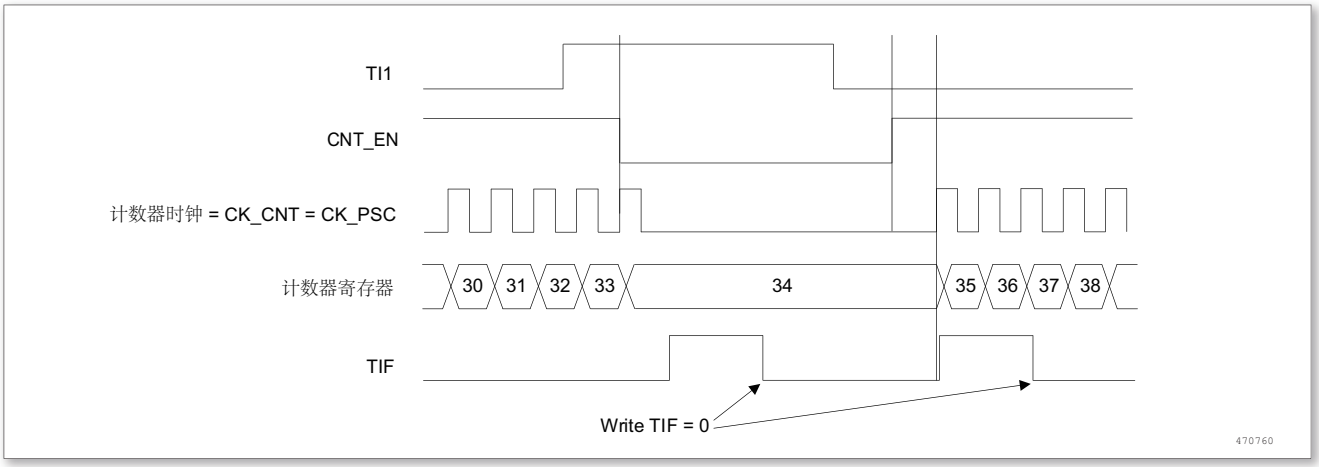


图 73. 门控模式下的控制电路

从模式：触发模式

计数器的使能依赖于选中的输入端上的事件。

在下面的例子中，计数器在 TI2 输入的上升沿开始向上计数：

- 配置通道 2 检测 TI2 的上升沿。配置输入滤波器带宽 (本例中，不需要任何滤波器，保持 IC2F = 0000)。触发操作中不使用捕获预分频器，不需要配置。CC2S 位只用于选择输入捕器中 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。

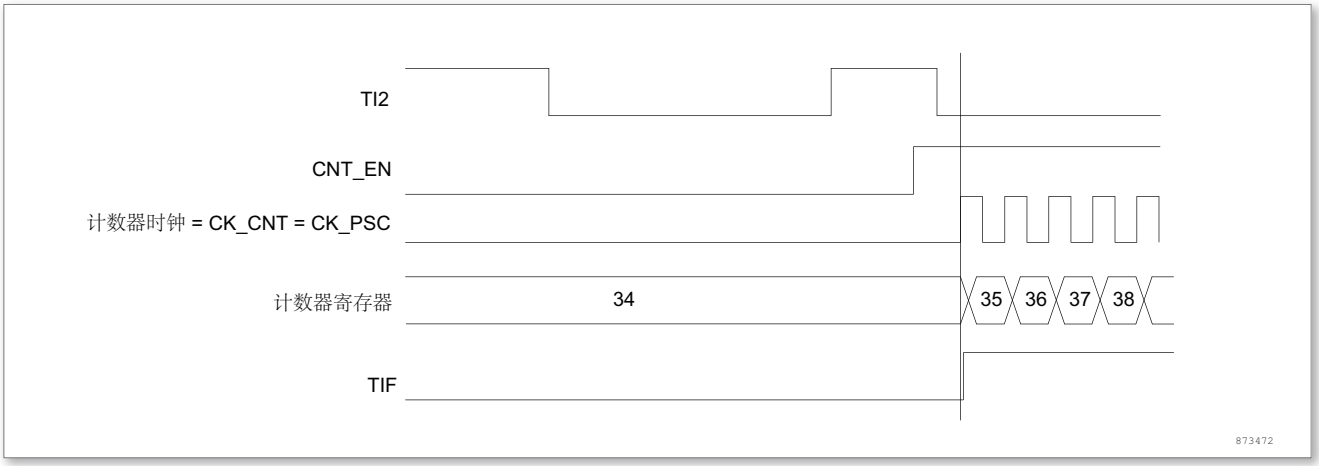


图 74. 触发器模式下的控制电路

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，当 TI1 上出现一个上升沿时，计数器即在 ETR 的每一个上升沿向上计数一次：

- 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)
- 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。

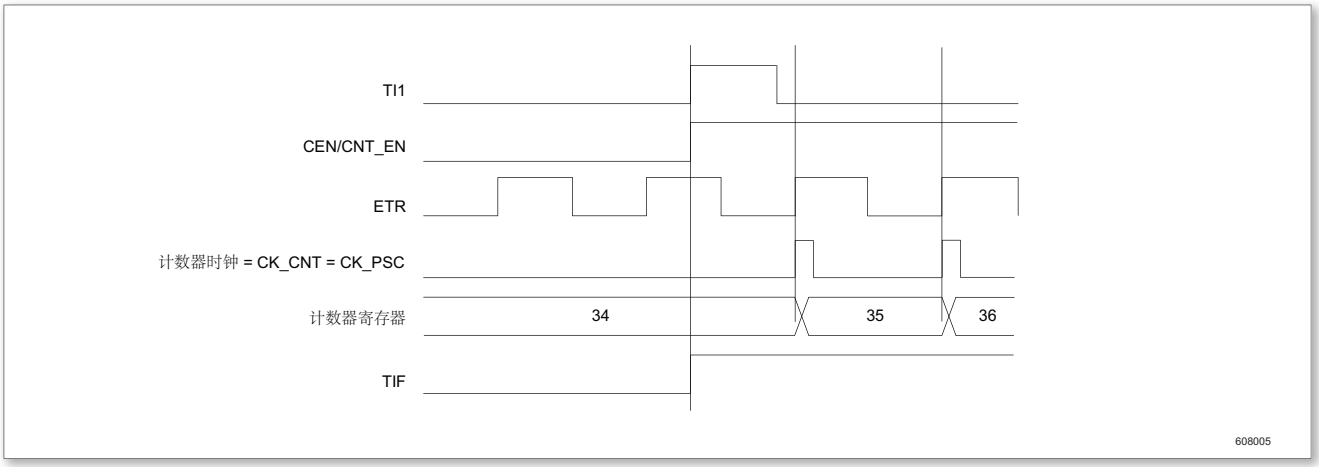


图 75. 外部时钟模式 2 ＋触发模式下的控制电路

11.3.20 定时器同步

所有 TIM 定时器在内部相连，用于定时器同步或链接。详见章节 TIM2/3/4。

11.3.21 调试模式

当微控制器进入调试模式时 (CPU 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。详见随后的调试章节。

11.4 寄存器描述

表 48. TIM1 寄存器概览

Offset	Acronym	Register Name	Reset	Section
0x00	TIMx_CR1	控制寄存器 1	0x00000000	小节 11.4.1
0x04	TIMx_CR2	控制寄存器 2	0x00000000	小节 11.4.2
0x08	TIMx_SMCR	从模式控制寄存器	0x00000000	小节 11.4.3
0x0C	TIMX_DIER	DMA/中断使能寄存器	0x00000000	小节 11.4.4
0x10	TIMx_SR	状态寄存器	0x00000000	小节 11.4.5
0x14	TIMx_EGR	事件产生寄存器	0x00000000	小节 11.4.6
0x18	TIMx_CCMR1	捕获/比较模式寄存器 1	0x00000000	小节 11.4.7
0x1C	TIMx_CCMR2	捕获/比较模式寄存器 2	0x00000000	小节 11.4.8
0x20	TIMx_CCER	捕获/比较使能寄存器	0x00000000	小节 11.4.9
0x24	TIMx_CNT	计数器	0x00000000	小节 11.4.10
0x28	TIMx_PSC	预分频率器	0x00000000	小节 11.4.11
0x2C	TIMx_ARR	自动装载寄存器	0x00000000	小节 11.4.12
0x30	TIMx_RCR	重复计数寄存器	0x00000000	小节 11.4.13
0x34	TIMx_CCR1	捕获/比较寄存器 1	0x00000000	小节 11.4.14
0x38	TIMx_CCR2	捕获/比较寄存器 2	0x00000000	小节 11.4.15
0x3C	TIMx_CCR3	捕获/比较寄存器 3	0x00000000	小节 11.4.16
0x40	TIMx_CCR4	捕获/比较寄存器 4	0x00000000	小节 11.4.17

Offset	Acronym	Register Name	Reset	Section
0x44	TIMx_BDTR	刹车和死区寄存器	0x00000000	小节 11.4.18
0x48	TIMx_DCR	DMA 控制寄存器	0x00000000	小节 11.4.19
0x4C	TIMx_DMAR	连续模式的 DMA 地址	0x00000000	小节 11.4.20

11.4.1 控制寄存器 1(TIMx_CR1)

偏移地址：0x00

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						CKD		ARPE	CMS		DIR	OPM	URS	UDIS	CEN
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
15：10	Reserved			保留，始终读为 0
9：8	CKD	rw	0x00	时钟分频因子 (Clock division) 这 2 位定义在定时器时钟 (CK_INT) 频率、死区时间和由死区发生器与数字滤波器 (ETR，TIx) 所用的采样时钟之间的分频比例。 00：$t_{DTS} = t_{CK\_INT}$ 01：$t_{DTS} = 2 \times t_{CK\_INT}$ 10：$t_{DTS} = 4 \times t_{CK\_INT}$ 11：保留，不要使用这个配置
7	ARPE	rw	0x00	自动重装载预装载允许位 (Auto-reload preload enable) 0：TIMx_ARR 寄存器没有缓冲 1：TIMx_ARR 寄存器被装入缓冲器
6：5	CMS	rw	0x00	选择中央对齐模式 (Center-aligned mode selection) 00：边沿对齐模式。计数器依据方向位 (DIR) 向上或向下计数 01：中央对齐模式 1。计数器交替地向上和向下计数。配置为输出的通道 (TIMx_CCMRx 寄存器中 CCxS = 00) 的输出比较中断标志位，只在计数器向下计数时被设置 10：中央对齐模式 2。计数器交替地向上和向下计数。配置为输出的通道 (TIMx_CCMRx 寄存器中 CCxS = 00) 的输出比较中断标志位，只在计数器向上计数时被设置 11：中央对齐模式 3。计数器交替地向上和向下计数。配置为输出的通道 (TIMx_CCMRx 寄存器中 CCxS = 00) 的输出比较中断标志位，在计数器向上和向下计数时均被设置 注：在计数器开启时 (CEN = 1)，不允许从边沿对齐模式转换到中央对齐模式。详见中央对齐的 PWM 波形

Bit	Field	Type	Reset	Description
4	DIR	rw	0x00	方向 (Direction) 0：计数器向上计数 1：计数器向下计数 ：当计数器配置为中央对齐模式或编码器模式时，该位为只读。
3	OPM	rw	0x00	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
2	URS	rw	0x00	更新请求源 (Update request source) 软件通过该位选择 UEV 事件的源。 0：如果允许产生更新中断或 DMA 请求，则下述任一事件产生一个更新中断或 DMA 请求： - 计数器溢出/下溢 - 设置 UG 位 - 从模式控制器产生的更新 1：如果允许产生更新中断或 DMA 请求，则只有计数器溢出/下溢才产生一个更新中断或 DMA 请求
1	UDIS	rw	0x00	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被缓存的寄存器被装入它们的预装载值。 1：禁止 UEV。不产生更新事件，影子寄存器 (ARR、PSC、CCRx) 保持它们的值。如果设置了 UG 位或从模式控制器发出了一个硬件复位，则计数器和预分频器被重新初始化
0	CEN	rw	0x00	允许计数器 (Counter enable) 0：禁止计数器 1：使能计数器。 注：在软件设置了 CEN 位后，外部时钟、门控模式和编码器模式才能工作。触发模式可以自动地通过硬件设置 CEN 位。

11.4.2 控制寄存器 2(TIMx_CR2)

偏移地址：0x04

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	OIS4	OIS3N	OIS3	OIS2N	OIS2	OIS1N	OIS1	TI1S	MMS			CCDS	CCUS	Res.	CCPC
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw

Bit	Field	Type	Reset	Description
15	Reserved			保留，始终读为 0

Bit	Field	Type	Reset	Description
14	OIS4	rw	0x00	输出空闲状态 4(OC4 输出)。参见 OIS1 位。
13	OIS3N	rw	0x00	输出空闲状态 3(OC3N 输出)。参见 OIS1N 位。
12	OIS3	rw	0x00	输出空闲状态 3(OC3 输出)。参见 OIS1 位。
11	OIS2N	rw	0x00	输出空闲状态 2(OC2N 输出)。参见 OIS1N 位。
10	OIS2	rw	0x00	输出空闲状态 2(OC2 输出)。参见 OIS1 位。
9	OIS1N	rw	0x00	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 注：已经设置了 LOCK(TIMx_BKR 寄存器) 级别 1、2 或 3 后，该位不能被修改。
8	OIS1	rw	0x00	输出空闲状态 1(OC1 输出)(Output Idle state 1) 0：当 MOE = 0 时，如果实现了 OC1N，则死区后 OC1 = 0 1：当 MOE = 0 时，如果实现了 OC1N，则死区后 OC1 = 1 注：已经设置了 LOCK(TIMx_BKR 寄存器) 级别 1、2 或 3 后，该位不能被修改。
7	TI1S	rw	0x00	TI1 选择 (TI1 selection) 0：TIMx_CH1 管脚连到 TI1 输入 1：TIMx_CH1、TIMx_CH2 和 TIMx_CH3 管脚经异或后连到 TI1 输入
6：4	MMS	rw	0x00	主模式选择 (Master mode selection) 这两位用于选择在主模式下送到从定时器的同步信息 (TRGO)。可能的组合如下： 000：复位-TIMx_EGR 寄存器的 UG 位被用于作为触发输出 (TRGO)。如果触发输入 (从模式控制器处于复位模式) 产生复位，则 TRGO 上的信号相对实际的复位会有一个延迟。 001：使能-计数器使能信号 CNT_EN 被用于作为触发输出 (TRGO)。有时需要在同一时间启动多个定时器或控制在一段时间内使能从定时器。计数器使能信号是通过 CEN 控制位和门控模式下的触发输入信号的逻辑或产生。当计数器使能信号受控于触发输入时，TRGO 上会有一个延迟，除非选择了主/从模式 (见 TIMx_SMCR 寄存器中 MSM 位的描述)。 010：更新-更新事件被选为触发输入 (TRGO)。例如，一个主定时器的时钟可以被用作一个从定时器的预分频器。 011：比较脉冲-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)

Bit	Field	Type	Reset	Description
3	CCDS	rw	0x00	捕获/比较的 DMA 选择 (Capture/compare DMA selection) 0：当发生 CCx 事件时，送出 CCx 的 DMA 请求 1：当发生更新事件时，送出 CCx 的 DMA 请求
2	CCUS	rw	0x00	捕获/比较控制更新选择 (Capture/compare control update selection) 0：如果捕获/比较控制位是预装载的 (CCPC = 1)，只能通过设置 COM 位更新它们 1：如果捕获/比较控制位是预装载的 (CCPC = 1)，可以通过设置 COM 位或 TRGI 上的一个上升沿更新它们 注：该位只对具有互补输出的通道起作用。
1	Reserved			保留，始终读为 0
0	CCPC	rw	0x00	捕获/比较预装载控制位 (Capture/compare preloaded control) 0：CCxE，CCxNE 和 OCxM 位不是预装载的 1：CCxE，CCxNE 和 OCxM 位是预装载的；设置该位后，它们只在设置了 COM 位后被更新 注：该位只对具有互补输出的通道起作用。

11.4.3 从模式控制寄存器 (TIMx_SMCR)

偏移地址：0x08

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ETP	ECE	ETPS		ETF				MSM	TS			Res.	SMS		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw

Bit	Field	Type	Reset	Description
15	ETP	rw	0x00	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
14	ECE	rw	0x00	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，计器由 ETRF 信号上的任意有效上升沿驱动 注 1：设置 ECE 位与选择外部时钟模式 1 并将 TRGI 连到 ETRF(SMS = 111 和 TS = 111) 具有相同功效。 注 2：下述从模式可以与外部时钟模式 2 同时使用：复位模式，门控模式和触发模式；但是，这时 TRGI 不能连到 ETRF(TS 位不能是 111)。 注 3：外部时钟模式 1 和外部时钟模式 2 同时被使能时，外部时钟的输入是 ETRF。

Bit	Field	Type	Reset	Description
13：12	ETPS	rw	0x00	外部触发预分频 (External trigger prescaler) 外部触发信号 ETRP 的频率必须最多是 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
11：8	ETF	rw	0x00	外部触发滤波 (External trigger filter) 这些位定义了对 ETRP 信号采样的频率和对 ETRP 数字滤波的带宽。实际上，数字滤波器是一个事件计数器，它记录到 N 个事件后会产生一个输出的跳变。 0000：无滤波器，以 fDTS 采样 0001：采样频率 $f_{SAMPLING}=f_{CK\_INT}$，N = 2 0010：采样频率 $f_{SAMPLING}=f_{CK\_INT}$，N = 4 0011：采样频率 $f_{SAMPLING}=f_{CK\_INT}$，N = 8 0100：采样频率 $f_{SAMPLING}=f_{DTS}/2$，N = 6 0101：采样频率 $f_{SAMPLING}=f_{DTS}/2$，N = 8 0110：采样频率 $f_{SAMPLING}=f_{DTS}/4$，N = 6 0111：采样频率 $f_{SAMPLING}=f_{DTS}/4$，N = 8 1000：采样频率 $f_{SAMPLING}=f_{DTS}/8$，N = 6 1001：采样频率 $f_{SAMPLING}=f_{DTS}/8$，N = 8 1010：采样频率 $f_{SAMPLING}=f_{DTS}/16$，N = 5 1011：采样频率 $f_{SAMPLING}=f_{DTS}/16$，N = 6 1100：采样频率 $f_{SAMPLING}=f_{DTS}/16$，N = 8 1101：采样频率 $f_{SAMPLING}=f_{DTS}/32$，N = 5 1110：采样频率 $f_{SAMPLING}=f_{DTS}/32$，N = 6 1111：采样频率 $f_{SAMPLING}=f_{DTS}/32$，N = 8
7	MSM	rw	0x00	主/从模式 (Master/slave mode) 0：无作用 1：触发输入 (TRGI) 上的事件被延迟了，以允许在当前定时器 (通过 TRGO) 与它的从定时器间的完美同步，这对要求把几个定时器同步到一个单一的外部事件时是非常有用的

Bit	Field	Type	Reset	Description
6：4	TS	rw	0x00	触发选择 (Trigger selection) 这 3 位选择用于同步计数器的触发输入。 000：内部触发 0(ITR0) 001：内部触发 1(ITR1) 010：内部触发 2(ITR2) 011：内部触发 3(ITR3) 100：TI1 的边沿检测器 (TI1F_ED) 101：滤波后的定时器输入 1(TI1FP1) 110：滤波后的定时器输入 2(TI2FP2) 111：外部触发输入 (ETRF) 更多有关 ITRx 的细节，参见下表。 注：这些位只能在未用到 (如 SMS = 000) 时被改变，以避免在改变时产生错误的边沿检测。
3	Reserved			保留，始终读为 0
2：0	SMS	rw	0x00	从模式选择 (Slave mode selection) 当选择了外部信号，触发信号 (TRGI) 的有效边沿与选中的外部输入极性相关 (见输入控制寄存器和控制寄存器的说明) 000：关闭从模式 - 如果 CEN = 1，则预分频器直接由内部时钟驱动。 001：编码器模式 1 - 根据 TI1FP1 的电平，计数器在 TI2FP2 的边沿向上/下计数。 010：编码器模式 2 - 根据 TI2FP2 的电平，计数器在 TI1FP1 的边沿向上/下计数。 011：编码器模式 3 - 根据另一个输入的电平，计数器在 TI1FP1 和 TI2FP2 的边沿向上/下计数。 100：复位模式 - 选中的触发输入 (TRGI) 的上升沿重新初始化计数器，并且产生一个更新寄存器的信号。 101：门控模式 - 当触发输入 (TRGI) 为高时，计数器的时钟开启。当触发输入变为低时，计数器停止 (但不复位)。计数器的启动和停止都是受控的。 110：触发模式 - 计数器在触发输入 TRGI 的上升沿启动 (但不复位)，只有计数器的启动是受控的。 111：外部时钟模式 1 - 选中的触发输入 (TRGI) 的上升沿驱动计数器。 注：如果 TI1F_EN 被选为触发输入 (TS = 100) 时，不要使用门控模式。这是因为，TI1F_ED 在每次 TI1F 变化时输出一个脉冲，然而门控模式是要检查触发输入的电平。

表 49. TIMx 内部触发连接

从定时器	ITR0	ITR1	ITR2	ITR3
TIM1	无	TIM2	TIM3	TIM4

从定时器	ITR0	ITR1	ITR2	ITR3
TIM2	TIM1	无	TIM3	无
TIM3	TIM1	TIM2	无	无
TIM4	TIM1	TIM2	TIM3	无

11.4.4 DMA/中断使能寄存器 (TIMX_DIER)

偏移地址：0x0C

复位值：0x0000

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

Bit	Field	Type	Reset	Description
15	Reserved			保留，始终读为 0
14	TDE	rw	0x00	允许触发 DMA 请求 (Trigger DMA request enable) 0：禁止触发 DMA 请求 1：允许触发 DMA 请求
13	COMDE	rw	0x00	允许 COM 的 DMA 请求 (COM DMA request enable) 0：禁止 COM 的 DMA 请求 1：允许 COM 的 DMA 请求
12	CC4DE	rw	0x00	允许捕获/比较 4 的 DMA 请求 (Capture/Compare 4 DMA request enable) 0：禁止捕获/比较 4 的 DMA 请求 1：允许捕获/比较 4 的 DMA 请求
11	CC3DE	rw	0x00	允许捕获/比较 3 的 DMA 请求 (Capture/Compare 3 DMA request enable) 0：禁止捕获/比较 3 的 DMA 请求 1：允许捕获/比较 3 的 DMA 请求
10	CC2DE	rw	0x00	允许捕获/比较 2 的 DMA 请求 (Capture/Compare 2 DMA request enable) 0：禁止捕获/比较 2 的 DMA 请求 1：允许捕获/比较 2 的 DMA 请求
9	CC1DE	rw	0x00	允许捕获/比较 1 的 DMA 请求 (Capture/Compare 1 DMA request enable) 0：禁止捕获/比较 1 的 DMA 请求 1：允许捕获/比较 1 的 DMA 请求
8	UDE	rw	0x00	允许更新的 DMA 请求 (Update DMA request enable) 0：禁止更新的 DMA 请求 1：允许更新的 DMA 请求
7	BIE	rw	0x00	允许刹车中断 (Break interrupt enable) 0：禁止刹车中断 1：允许刹车中断

Bit	Field	Type	Reset	Description
6	TIE	rw	0x00	触发中断使能 (Trigger interrupt enable) 0：禁止触发中断 1：使能触发中断
5	COMIE	rw	0x00	允许 COM 中断 (COM interrupt enable) 0：禁止 COM 中断 1：允许 COM 中断
4	CC4IE	rw	0x00	允许捕获/比较 4 中断 (Capture/Compare 4 interrupt enable) 0：禁止捕获/比较 4 中断 1：允许捕获/比较 4 中断
3	CC3IE	rw	0x00	允许捕获/比较 3 中断 (Capture/Compare 3 interrupt enable) 0：禁止捕获/比较 3 中断 1：允许捕获/比较 3 中断
2	CC2IE	rw	0x00	允许捕获/比较 2 中断 (Capture/Compare 2 interrupt enable) 0：禁止捕获/比较 2 中断 1：允许捕获/比较 2 中断
1	CC1IE	rw	0x00	允许捕获/比较 1 中断 (Capture/Compare 1 interrupt enable) 0：禁止捕获/比较 1 中断 1：允许捕获/比较 1 中断
0	UIE	rw	0x00	允许更新中断 (Update interrupt enable) 0：禁止更新中断 1：允许更新中断

11.4.5 状态寄存器 (TIMx_SR)

偏移地址：0x10

复位值：0x0000

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

Bit	Field	Type	Reset	Description
15：13	Reserved			保留，始终读为 0
12	CC4OF	rc_w0	0x00	捕获/比较 4 重复捕获标记 (Capture/Compare 4 overcapture flag) 参见 CC1OF 描述。
11	CC3OF	rc_w0	0x00	捕获/比较 3 重复捕获标记 (Capture/Compare 3 overcapture flag) 参见 CC1OF 描述。

Bit	Field	Type	Reset	Description
10	CC2OF	rc_w0	0x00	捕获/比较 2 重复捕获标记 (Capture/Compare 2 overcapture flag) 参见 CC1OF 描述。
9	CC1OF	rc_w0	0x00	捕获/比较 1 重复捕获标记 (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。写 0 可清除该位。 0：无重复捕获产生； 1：计数器的值被捕获到 TIMx_CCR1 寄存器时，CC1IF 的状态已经为 1。
8	Reserved			保留，始终读为 0
7	BIF	rc_w0	0x00	刹车中断标记 (Break interrupt flag) 当刹车输入有效时，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
6	TIF	rc_w0	0x00	触发器中断标记 (Trigger interrupt flag) 当发生触发事件 (当从模式控制器处于除门控模式外的其它模式时，在 TRGI 输入端检测到有效边沿，或或门控模式下的任一边沿) 时由硬件对该位置 1。它由软件清 0。 0：无触发器事件产生 1：触发器中断等待响应
5	COMIF	rc_w0	0x00	COM 中断标记 (COM interrupt flag) 当产生 COM 事件 (当捕获/比较控制位：CCxE、CCxNE、OCxM 已被更新) 时该位由硬件置 1。它由软件清 0。 0：无 COM 事件产生 1：COM 中断等待响应
4	CC4IF	rc_w0	0x00	捕获/比较 4 中断标记 (Capture/Compare 4 interrupt flag) 参考 CC1IF 描述。
3	CC3IF	rc_w0	0x00	捕获/比较 3 中断标记 (Capture/Compare 3 interrupt flag) 参考 CC1IF 描述。
2	CC2IF	rc_w0	0x00	捕获/比较 2 中断标记 (Capture/Compare 2 interrupt flag) 参考 CC1IF 描述。

Bit	Field	Type	Reset	Description
1	CC1IF	rc_w0	0x00	捕获/比较 1 中断标记 (Capture/Compare 1 interrupt flag) 如果通道 CC1 配置为输出模式： 当计数器值与比较值匹配时该位由硬件置‘1’，但在中心对称模式下除外 (参考 TIMx_CR1 寄存器的 CMS 位)。它由软件清‘0’。 0：无匹配发生 1：TIMx_CNT 的值与 TIMx_CCR1 的值匹配 如果通道 CC1 配置为输入模式： 当捕获事件发生时该位由硬件置‘1’，它由软件清 0 或通过读 TIMx_CCR1 清‘0’。 0：无输入捕获产生 1：计数器值已被捕获 (拷贝) 至 TIMx_CCR1(在 IC1 上检测到与所选极性相同的边沿)
0	UIF	rc_w0	0x00	更新中断标记 (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：更新事件等待响应。当寄存器被更新时该位由硬件置‘1’： - 若 TIMx_CR1 寄存器的 UDIS = 0，当 REP_CNT = 0 时产生更新事件 (重复向下计数器上溢或下溢时) - 若 TIMx_CR1 寄存器的 UDIS = 0、URS = 0，当 TIMx_EGR 寄存器的 UG = 1 时产生更新事件 (软件对计数器 CNT 重新初始化) - 若 TIMx_CR1 寄存器的 UDIS = 0、URS = 0，当计数器 CNT 被触发事件重初始化时产生更新事件。(参考同步控制寄存器的说明)

11.4.6 事件产生寄存器 (TIMx_EGR)

偏移地址：0x14

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								BG	TG	COMG	CC4G	CC3G	CC2G	CC1G	UG
								w	w	w	w	w	w	w	w

Bit	Field	Type	Reset	Description
15：8	Reserved			保留，始终读为 0

Bit	Field	Type	Reset	Description
7	BG	w	0x00	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
6	TG	w	0x00	产生触发事件 (Trigger generation) 该位由软件置‘1’，用于产生一个触发事件，由硬件自动清‘0’。 0：无动作 1：TIMx_SR 寄存器的 TIF = 1，若开启对应的中断和 DMA，则产生相应的中断和 DMA
5	COMG	w	0x00	捕获/比较事件，产生控制更新 (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 注：该位只对拥有互补输出的通道有效。
4	CC4G	w	0x00	产生捕获/比较 4 事件 (Capture/Compare 4 generation) 参考 CC1G 描述。
3	CC3G	w	0x00	产生捕获/比较 3 事件 (Capture/Compare 3 generation) 参考 CC1G 描述。
2	CC2G	w	0x00	产生捕获/比较 2 事件 (Capture/Compare 2 generation) 参考 CC1G 描述。
1	CC1G	w	0x00	产生捕获/比较 1 事件 (Capture/Compare 1 generation) 该位由软件置 1，用于产生一个捕获/比较事件，由硬件自动清 0。 0：无动作 1：在通道 CC1 上产生一个捕获/比较事件： 若通道 CC1 配置为输出： 设置 CC1IF=1，若开启对应的中断和 DMA，则产生相应的中断和 DMA。 若通道 CC1 配置为输入： 当前的计数器值被捕获至 TIMx_CCR1 寄存器，设置 CC1IF = 1，若开启对应的中断和 DMA，则产生相应的中断和 DMA。若 CC1IF 已经为 1，则设置 CC1OF = 1。

Bit	Field	Type	Reset	Description
0	UG	w	0x00	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。

11.4.7 捕获/比较模式寄存器 1(TIMx_CCMR1)

偏移地址：0x18

复位值：0x0000

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

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

输出比较模式：

Bit	Field	Type	Reset	Description
15	OC2CE	rw	0x00	输出比较 2 清 0 使能 (Output compare 2 clear enable)
14：12	OC2M	rw	0x00	输出比较 2 模式 (Output compare 2 mode)
11	OC2PE	rw	0x00	输出比较 2 预装载使能 (Output compare 2 preload enable)
10	OC2FE	rw	0x00	输出比较 2 快速使能 (Output compare 4 fast enable)
9：8	CC2S	rw	0x00	捕获/比较 2 选择 (Capture/Compare 2 selection) 该位定义通道的方向 (输入/输出)，及输入脚的选择： 00：CC2 通道被配置为输出 01：CC2 通道被配置为输入，IC2 映射在 TI2 上 10：CC2 通道被配置为输入，IC2 映射在 TI1 上 11：CC2 通道被配置为输入，IC2 映射在 TRC 上。此模式仅工作在内部触发器输入被选中时 (由 TIMx_SMCR 寄存器的 TS 位选择) 注：CC2S 仅在通道关闭时 (TIMx_CCER 寄存器的 CC2E = 0) 才是可写的。

Bit	Field	Type	Reset	Description
7	OC1CE	rw	0x00	输出比较 1 清 0 使能 (Output compare 1 clear enable) 0：OC1REF 不受 ETRF 输入的影响 1：当检测到 ETRF 输入高电平时，清除 OC1REF = 0
6：4	OC1M	rw	0x00	输出比较 1 模式 (Output compare 1 mode) 该 3 位定义了输出参考信号 OC1REF 的动作，而 OC1REF 决定了 OC1、OC1N 的值。OC1REF 是高电平有效，而 OC1、OC1N 的有效电平取决于 CC1P、CC1NP 位。 000：冻结。输出比较寄存器 TIMx_CCR1 与计数器 TIMx_CNT 间的比较对 OC1REF 不起作用 001：匹配时设置通道 1 为有效电平。当计数器 TIMx_CNT 的值与捕获/比较寄存器 1(TIMx_CCR1) 相同时，强制 OC1REF 为高 010：匹配时设置通道 1 为无效电平。当计数器 TIMx_CNT 的值与捕获/比较寄存器 1(TIMx_CCR1) 相同时，强制 OC1REF 为低 011：翻转。当 TIMx_CCR1=TIMx_CNT 时，翻转 OC1REF 的电平 100：强制为无效电平。强制 OC1REF 为低 101：强制为有效电平。强制 OC1REF 为高 110：PWM 模式 1 - 在向上计数时，当 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，当 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，当 TIMx_CNT < TIMx_CCR1 时通道 1 为无效电平，否则为有效电平；在向下计数时，当 TIMx_CNT > TIMx_CCR1 时通道 1 为有效电平，否则为无效电平 注 1：当 LOCK 级别设为 3(TIMx_BDTR 寄存器中的 LOCK 位)并且 CC1S = 00(该通道配置成输出) 时，该位不能被修改。 注 2：在 PWM 模式 1 或 PWM 模式 2 中，只有当比较结果改变了或在输出比较模式中从冻结模式切换到 PWM 模式时，OC1REF 电平才改变。

Bit	Field	Type	Reset	Description
3	OC1PE	rw	0x00	输出比较 1 预装载使能 (Output compare 1 preload enable) 0：禁止 TIMx_CCR1 寄存器的预装载功能，可随时写入 TIMx_CCR1 寄存器，并且新写入的数值立即起作用 1：开启 TIMx_CCR1 寄存器的预装载功能，读写操作仅对预装载寄存器操作，TIMx_CCR1 的预装载值在更新事件到来时被加载至当前寄存器中 注 1：当 LOCK 级别设为 3(TIMx_BDTR 寄存器中的 LOCK 位)并且 CC1S = 00(该通道配置成输出) 时，该位不能被修改。 注 2：仅在单脉冲模式下 (TIMx_CR1 寄存器的 OPM = 1)，可以在未确认预装载寄存器情况下使用 PWM 模式，否则其动作不确定。
2	OC1FE	rw	0x00	输出比较 1 快速使能 (Output compare 1 fast enable) 该位用于加快 CC 输出对触发输入事件的响应。 0：根据计数器与 CCR1 的值，CC1 正常操作，即使触发器是打开的。当触发器的输入有一个有效沿时，激活 CC1 输出的最小延时为 5 个时钟周期 1:输入到触发器的有效沿的作用就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
1：0	CC1S	rw	0x00	捕获/比较 1 选择 (Capture/Compare 1 selection) 这 2 位定义通道的方向 (输入/输出)，及输入脚的选择： 00：CC1 通道被配置为输出 01：CC1 通道被配置为输入，IC1 映射在 TI1 上 10：CC1 通道被配置为输入，IC1 映射在 TI2 上 11：CC1 通道被配置为输入，IC1 映射在 TRC 上，此模式仅工作在内部触发器输入被选中时 (由 TIMx_SMCR 寄存器的 TS 位选择) 注：CC1S 仅在通道关闭时 (TIMx_CCER 寄存器的 CC1E = 0)才是可写的。

输入捕获模式：

Bit	Field	Type	Reset	Description
15：12	IC2F	rw	0x00	输入捕获 2 滤波器 (Input capture 2 filter)
11：10	IC2PSC	rw	0x00	输入/捕获 2 预分频器 (Input capture 2 prescaler)

Bit	Field	Type	Reset	Description
9：8	CC2S	rw	0x00	捕获/比较 2 选择 (Capture/Compare 2 selection) 这 2 位定义通道的方向 (输入/输出)，及输入脚的选择： 00：CC2 通道被配置为输出 01：CC2 通道被配置为输入，IC2 映射在 TI2 上 10：CC2 通道被配置为输入，IC2 映射在 TI1 上 11：CC2 通道被配置为输入，IC2 映射在 TRC 上，此模式仅工作在内部触发器输入被选中时 (由 TIMx_SMCR 寄存器的 TS 位选择) 注：CC2S 仅在通道关闭时 (TIMx_CCER 寄存器的 CC2E = 0) 才是可写的。
7：4	IC1F	rw	0x00	输入捕获 1 滤波器 (Input capture 1 filter) 这几位定义了 TI1 输入的采样频率及数字滤波器长度。数字滤波器由一个事件计数器组成，它记录到 N 个事件后会产生一个输出的跳变： 0000：无滤波器，以 fDTS 采样 1000：采样频率 $f_{SAMPLING}=f_{DTS}/8$，N = 6 0001：采样频率 $f_{SAMPLING}=f_{CK\_INT}$，N = 2 1001：采样频率 $f_{SAMPLING}=f_{DTS}/8$，N = 8 0010：采样频率 $f_{SAMPLING}=f_{CK\_INT}$，N = 4 1010：采样频率 $f_{SAMPLING}=f_{DTS}/16$，N = 5 0011：采样频率 $f_{SAMPLING}=f_{CK\_INT}$，N = 8 1011：采样频率 $f_{SAMPLING}=f_{DTS}/16$，N = 6 0100：采样频率 $f_{SAMPLING}=f_{DTS}/2$，N = 6 1100：采样频率 $f_{SAMPLING}=f_{DTS}/16$，N = 8 0101：采样频率 $f_{SAMPLING}=f_{DTS}/2$，N = 8 1101：采样频率 $f_{SAMPLING}=f_{DTS}/32$，N = 5 0110：采样频率 $f_{SAMPLING}=f_{DTS}/4$，N = 6 1110：采样频率 $f_{SAMPLING}=f_{DTS}/32$，N = 6 0111：采样频率 $f_{SAMPLING}=f_{DTS}/4$，N = 8 1111：采样频率 $f_{SAMPLING}=f_{DTS}/32$，N = 8
3：2	IC1PSC	rw	0x00	输入/捕获 1 预分频器 (Input capture 1 prescaler) 这 2 位定义了 CC1 输入 (IC1) 的预分频系数。 当 CC1E = 0(TIMx_CCER 寄存器中) 时，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

Bit	Field	Type	Reset	Description
1：0	CC1S	rw	0x00	捕获/比较 1 选择 (Capture/compare 1 selection) 这 2 位定义通道的方向 (输入/输出)，及输入脚的选择： 00：CC1 通道被配置为输出 01：CC1 通道被配置为输入，IC1 映射在 TI1 上 10：CC1 通道被配置为输入，IC1 映射在 TI2 上 11：CC1 通道被配置为输入，IC1 映射在 TRC 上，此模式仅工作在内部触发器输入被选中时 (由 TIMx_SMCR 寄存器的 TS 位选择) 注：CC1S 仅在通道关闭时 (TIMx_CCER 寄存器的 CC1E = 0) 才是可写的。

11.4.8 捕获/比较模式寄存器 2(TIMx_CCMR2)

偏移地址：0x1C

复位值：0x0000

参看以上 CCMR1 寄存器的描述。

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

输出比较模式

Bit	Field	Type	Reset	Description
15	OC4CE	rw	0x00	输出比较 4 清 0 使能 (Output compare 4 clear enable)
14：12	OC4M	rw	0x00	输出比较 4 模式 (Output compare 4 mode)
11	OC4PE	rw	0x00	输出比较 4 预装载使能 (Output compare 4 preload enable)
10	OC4FE	rw	0x00	输出比较 4 快速使能 (Output compare 4 fast enable)
9：8	CC4S	rw	0x00	捕获/比较 4 选择 (Capture/Compare 4 selection) 该 2 位定义通道的方向 (输入/输出)，及输入脚的选择： 00：CC4 通道被配置为输出 01：CC4 通道被配置为输入，IC4 映射在 TI4 上 10：CC4 通道被配置为输入，IC4 映射在 TI3 上 11：CC4 通道被配置为输入，IC4 映射在 TRC 上，此模式仅工作在内部触发器输入被选中时 (由 TIMx_SMCR 寄存器的 TS 位选择) 注：CC4S 仅在通道关闭时 (TIMx_CCER 寄存器的 CC4E = 0) 才是可写的。
7	OC3CE	rw	0x00	输出比较 3 清‘0’使能 (Output compare 3 clear enable)
6：4	OC3M	rw	0x00	输出比较 3 模式 (Output compare 3 mode)

Bit	Field	Type	Reset	Description
3	OC3PE	rw	0x00	输出比较 3 预装载使能 (Output compare 3 preload enable)
2	OC3FE	rw	0x00	输出比较 3 快速使能 (Output compare 3 fast enable)
1：0	CC3S	rw	0x00	捕获/比较 3 选择 (Capture/Compare 3 selection) 这 2 位定义通道的方向 (输入/输出)，及输入脚的选择： 00：CC3 通道被配置为输出 01：CC3 通道被配置为输入，IC3 映射在 TI3 上 10：CC3 通道被配置为输入，IC3 映射在 TI4 上 11：CC3 通道被配置为输入，IC3 映射在 TRC 上，此模式仅工作在内部触发器输入被选中时 (由 TIMx_SMCR 寄存器的 TS 位选择) 注：CC3S 仅在通道关闭时 (TIMx_CCER 寄存器的 CC3E = 0) 才是可写的。

输入捕获模式

Bit	Field	Type	Reset	Description
15：12	IC4F	rw	0x00	输入捕获 4 滤波器 (Input capture 4 filter)
11：10	IC4PSC	rw	0x00	输入/捕获 4 预分频器 (Input capture 4 prescaler)
9：8	CC4S	rw	0x00	捕获/比较 4 选择 (Capture/Compare 4 selection) 这 2 位定义通道的方向 (输入/输出)，及输入脚的选择： 00：CC4 通道被配置为输出 01：CC4 通道被配置为输入，IC4 映射在 TI4 上 10：CC4 通道被配置为输入，IC4 映射在 TI3 上 11：CC4 通道被配置为输入，IC4 映射在 TRC 上，此模式仅工作在内部触发器输入被选中时 (由 TIMx_SMCR 寄存器的 TS 位选择) 注：CC4S 仅在通道关闭时 (TIMx_CCER 寄存器的 CC4E = 0) 才是可写的。
7：4	IC3F	rw	0x00	输入捕获 3 滤波器 (Input capture 3 filter)
3：2	IC3PSC	rw	0x00	输入/捕获 3 预分频器 (Input capture 3 prescaler)
1：0	CC3S	rw	0x00	捕获/比较 3 选择 (Capture/compare 3 selection) 这 2 位定义通道的方向 (输入/输出)，及输入脚的选择： 00：CC3 通道被配置为输出 01：CC3 通道被配置为输入，IC3 映射在 TI3 上 10：CC3 通道被配置为输入，IC3 映射在 TI4 上 11：CC3 通道被配置为输入，IC3 映射在 TRC 上，此模式仅工作在内部触发器输入被选中时 (由 TIMx_SMCR 寄存器的 TS 位选择) 注：CC3S 仅在通道关闭时 (TIMx_CCER 寄存器的 CC3E = 0) 才是可写的。

11.4.9 捕获/比较使能寄存器 (TIMx_CCER)

偏移地址：0x20

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		CC4P	CC4E	CC3 NP	CC3 NE	CC3P	CC3E	CC2 NP	CC2 NE	CC2P	CC2E	CC1 NP	CC1 NE	CC1P	CC1E
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
15：14	Reserved			保留，始终读为 0
13	CC4P	rw	0x00	输入/捕获 4 输出极性 (Capture/Compare 4 output polarity) 参考 CC1P 的描述。
12	CC4E	rw	0x00	输入/捕获 4 输出使能 (Capture/Compare 4 output enable) 参考 CC1E 的描述。
11	CC3NP	rw	0x00	输入/捕获 3 互补输出极性 (Capture/Compare 3 complementary output polarity) 参考 CC1NP 的描述。
10	CC3NE	rw	0x00	输入/捕获 3 互补输出使能 (Capture/Compare 3 complementary output enable) 参考 CC1NE 的描述。
9	CC3P	rw	0x00	输入/捕获 3 输出极性 (Capture/Compare 3 output polarity) 参考 CC1P 的描述。
8	CC3E	rw	0x00	输入/捕获 3 输出使能 (Capture/Compare 3 output enable) 参考 CC1E 的描述。
7	CC2NP	rw	0x00	输入/捕获 2 互补输出极性 (Capture/Compare 2 complementary output polarity) 参考 CC1NP 的描述。
6	CC2NE	rw	0x00	输入/捕获 2 互补输出使能 (Capture/Compare 2 complementary output enable) 参考 CC1NE 的描述。
5	CC2P	rw	0x00	输入/捕获 2 输出极性 (Capture/Compare 2 output polarity) 参考 CC1P 的描述。
4	CC2E	rw	0x00	输入/捕获 2 输出使能 (Capture/Compare 2 output enable) 参考 CC1E 的描述。
3	CC1NP	rw	0x00	输入/捕获 1 互补输出极性 (Capture/Compare 1 complementary output polarity) 0：OC1N 高电平有效 1：OC1N 低电平有效 注：当 LOCK 级别 (TIMx_BDTR 寄存器中的 LCCK 位) 设为 3 或 2 且 CC1S = 00(通道配置为输出) 时，该位不能被修改。

Bit	Field	Type	Reset	Description
2	CC1NE	rw	0x00	输入/捕获 1 互补输出使能 (Capture/Compare 1 complementary output enable) 0：关闭 - OC1N 禁止输出，因此 OC1N 的输出电平依赖于 MOE、OSSI、OSSR、OIS1、OIS1N 和 CC1E 位的值 1：开启 - OC1N 信号输出到对应的输出引脚，其输出电平依赖于 MOE、OSSI、OSSR、OIS1、OIS1N 和 CC1E 位的值
1	CC1P	rw	0x00	输入/捕获 1 输出极性 (Capture/Compare 1 output polarity) CC1 通道配置为输出： 0：OC1 高电平有效 1：OC1 低电平有效 CC1 通道配置为输入： 该位选择是 IC1 还是 IC1 的反相信号作为触发或捕获信号。 0：不反相：捕获发生在 IC1 的上升沿；当用作外部触发器时，IC1 不反相 1：反相：捕获发生在 IC1 的下降沿；当用作外部触发器时，IC1 反相
0	CC1E	rw	0x00	输入/捕获 1 输出使能 (Capture/Compare 1 output enable) CC1 通道配置为输出： 0：关闭 - OC1 禁止输出，因此 OC1 的输出电平依赖于 MOE、OSSI、OSSR、OIS1、OIS1N 和 CC1NE 位的值 1：开启 - OC1 信号输出到对应的输出引脚，其输出电平依赖于 MOE、OSSI、OSSR、OIS1、OIS1N 和 CC1NE 位的值 CC1 通道配置为输入： 该位决定了计数器的值是否能捕获入 TIMx_CCR1 寄存器。 0：捕获禁止 1：捕获使能

表 50. 带刹车功能的互补输出通道 OCx 和 OCxN 的控制位

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

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

注 1：与互补通道 OCx 和 OCxN 相连的外部 I/O 管脚的状态，取决于 OCx 和 OCxN 通道状态和 GPIO 寄存器。

2：如果 CCxE=0 且 CCxNE=0，输出禁止后 OCx 和 OCxN 为高阻。

11.4.10 计数器 (TIMx_CNT)

偏移地址：0x24

复位值：0x0000

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

Bit	Field	Type	Reset	Description
15：0	CNT	rw	0x0000	计数器的值 (Counter value)

11.4.11 预分频器 (TIMx_PSC)

偏移地址：0x28

复位值：0x0000

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

Bit	Field	Type	Reset	Description
15：0	PSC	rw	0x0000	预分频器的值 (Prescaler value) 计数器的时钟频率 (CK_CNT) 等于 f_{CK_PSC} /(PSC + 1)。 PSC 包含了每次当更新事件产生时，装入当前预分频器寄存器的值。更新事件包括计数器被 TIM_EGR 的 UG 位清‘0’或被工作在复位模式的从控制器清‘0’。

11.4.12 自动装载寄存器 (TIMx_ARR)

偏移地址：0x2C

复位值：0x0000

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

Bit	Field	Type	Reset	Description
15：0	ARR	rw	0x0000	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数值。 详细参考小节 11.3.1: 有关 ARR 的更新和动作。 当自动重装载的值为空时，计数器不工作。

11.4.13 重复计数寄存器 (TIMx_RCR)

偏移地址：0x30

复位值：0x0000

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

Bit	Field	Type	Reset	Description
15：8	Reserved			保留，始终读为 0。
7：0	REP	rw	0x00	重复计数器的值 (Repetition counter value) 开启了预装载功能后，这些位允许用户设置比较寄存器的更新速率 (即周期性地从预装载寄存器传输到当前寄存器)；如果允许产生更新中断，则会同时影响产生更新中断的速率。 每次向下计数器 REP_CNT 达到 0，会产生一个更新事件并且计数器 REP_CNT 重新从 REP 值开始计数。由于 REP_CNT 只有在周期更新事件 U_RC 发生时才重载 REP 值，因此对 TIMx_RCR 寄存器写入的新值只在下次周期更新事件发生时才起作用。 这意味着在 PWM 模式中，(REP+1) 对应着： - 在边沿对齐模式下，PWM 周期的数目 - 在中心对称模式下，PWM 半周期的数目

11.4.14 捕获/比较寄存器 1(TIMx_CCR1)

偏移地址：0x34

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
15：0	CCR1	rw	0x0000	捕获/比较 1 的值 (Capture/Compare 1 value) 若 CC1 通道配置为输出： CCR1 包含了装入当前捕获/比较 1 寄存器的值 (预装载值)。 如果在 TIMx_CCMR1 寄存器 (OC1PE 位) 中未选择预装载功能，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。

11.4.15 捕获/比较寄存器 2(TIMx_CCR2)

偏移地址：0x38

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR2															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
15：0	CCR2	rw	0x0000	捕获/比较 2 的值 (Capture/Compare 2 value) 若 CC2 通道配置为输出： CCR2 包含了装入当前捕获/比较 2 寄存器的值 (预装载值)。 如果在 TIMx_CCMR2 寄存器 (OC2PE 位) 中未选择预装载特性，写入的数值会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。

11.4.16 捕获/比较寄存器 3(TIMx_CCR3)

偏移地址：0x3C

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR3															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
15：0	CCR3	rw	0x0000	捕获/比较 3 的值 (Capture/Compare 3 value) 若 CC3 通道配置为输出： CCR3 包含了装入当前捕获/比较 3 寄存器的值 (预装载值)。 如果在 TIMx_CCMR3 寄存器 (OC3PE 位) 中未选择预装载特性，写入的数值会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。

11.4.17 捕获/比较寄存器 4(TIMx_CCR4)

偏移地址：0x40

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR4															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
15：0	CCR4	rw	0x0000	捕获/比较 4 的值 (Capture/Compare 4 value) 若 CC4 通道配置为输出： CCR4 包含了装入当前捕获/比较 4 寄存器的值 (预装载值)。 如果在 TIMx_CCMR4 寄存器 (OC4PE 位) 中未选择预装载特性，写入的数值会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。

11.4.18 刹车和死区寄存器 (TIMx_BDTR)

偏移地址：0x44

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MOE	AOE	BKP	BKE	OSSR	OSSI	LOCK		DTG							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bit	Field	Type	Reset	Description
15	MOE	rw	0x00	主输出使能 (Main output enable) 当刹车输入有效时，该位被硬件异步清‘0’。根据 AOE 位的设置值，该位可以由软件清‘0’或被自动置‘1’。它仅对配置为输出的通道有效。 0：禁止 OC 和 OCN 输出或强制为空闲状态 1：如果设置了相应的使能位 (TIMx_CCER 寄存器的 CCxE、CCxNE 位)，则开启 OC 和 OCN 输出 有关 OC/OCN 使能的细节，参见小节 11.4.9，捕获/比较使能寄存器 (TIMx_CCER)。
14	AOE	rw	0x00	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) 注：当 LOCK 级别 (TIMx_BDTR 寄存器中的 LOCK 位) 设为 1 时，该位不能被修改。
13	BKP	rw	0x00	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 注：当 LOCK 级别 (TIMx_BDTR 寄存器中的 LOCK 位) 设为 1 时，该位不能被修改。
12	BKE	rw	0x00	刹车功能使能 (Break enable) 0：禁止刹车输入 (BRK 及 BRK_ACTH) 1：开启刹车输入 (BRK 及 BRK_ACTH) 注：当 LOCK 级别 (TIMx_BDTR 寄存器中的 LOCK 位) 设为 1 时，该位不能被修改。
11	OSSR	rw	0x00	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。 参考 OC/OCN 使能的详细说明 (小节 11.4.9，捕获/比较使能寄存器 (TIMx_CCER))。 0：当定时器不工作时，禁止 OC/OCN 输出 (OC/OCN 使能输出信号 = 0) 1：当定时器不工作时，如果 CCxE = 1 或 CCxNE = 1，首先开启 OC/OCN 并输出无效电平，然后置 OC/OCN 使能输出信号 = 1 注：当 LOCK 级别 (TIMx_BDTR 寄存器中的 LOCK 位) 设为 2 时，该位不能被修改。

Bit	Field	Type	Reset	Description
10	OSSI	rw	0x00	空闲模式下‘关闭状态’选择 (Off-state selection for Idle mode) 该位用于当 MOE = 0 且通道设为输出时。 参考 OC/OCN 使能的详细说明 (小节 11.4.9，捕获/比较使能寄存器 (TIMx_CCER))。 0：当定时器不工作时，禁止 OC/OCN 输出 (OC/OCN 使能输出信号 = 0) 1：当定时器不工作时，当 CCxE = 1 或 CCxNE = 1 时，OC/OCN 首先输出其空闲电平，然后 OC/OCN 使能输出信号 = 1 注：当 LOCK 级别 (TIMx_BDTR 寄存器中的 LOCK 位) 设为 2 时，则该位不能被修改。
9：8	LOCK	rw	0x00	锁定设置 (Lock configuration) 该位为防止软件错误而提供写保护。 00：锁定关闭，寄存器无写保护 01：锁定级别 1，不能写入 TIMx_BDTR 寄存器的 DTG、BKE、BKP、AOE 位和 TIMx_CR2 寄存器的 OISx/OISxN 位 10：锁定级别 2，不能写入锁定级别 1 中的各位，也不能写入 CC 极性位 (当相关通道通过 CCxS 位设为输出时，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 (当相关通道通过 CCxS 位设为输出时，CC 控制位是 TIMx_CCMRx 寄存器的 OCxM/OCxPE 位) 注：在系统复位后，只能写一次 LOCK 位，当写入 TIMx_BDTR 寄存器时，其内容冻结直至复位。

Bit	Field	Type	Reset	Description
7：0	DTG	rw	0x00	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。假设 DT 表示其持续时间： DTG[7: 5] = 0xx: DT = (DTG[7: 0] + 1) × t_{dtg}, t_{dtg} = t_{DTS}; DTG[7: 5] = 10x: DT = (DTG[5: 0] + 1 + 64) × t_{dtg}, t_{dtg} = 2 × t_{DTS}; DTG[7: 5] = 110: DT = (DTG[4: 0] + 1 + 32) × t_{dtg}, t_{dtg} = 8 × t_{DTS}; DTG[7: 5] = 111: DT = (DTG[4: 0] + 1 + 32) × t_{dtg}, t_{dtg} = 16 × t_{DTS}; 例：若 t_{DTS} = 125ns(8MHz)，可能的死区时间为： 125ns 到 15875ns(步长时间为 125ns)， 16µs 到 31750ns(步长时间为 250ns)， 32µs 到 63µs(步长时间为 1µs)， 64µs 到 126µs(步长时间为 2µs)。 注：当 LOCK 级别 (TIMx_BDTR 寄存器中的 LOCK 位) 设为 1、2 或 3 时，不能修改这些位。

11.4.19 DMA 控制寄存器 (TIMx_DCR)

偏移地址：0x48

复位值：0x0000

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

Bit	Field	Type	Reset	Description
15：13	Reserved			保留，始终读为 0。

Bit	Field	Type	Reset	Description
12：8	DBL	w	0x00	DMA 连续传送长度 (DMA burst length) 这些位定义了 DMA 在连续模式下的传送长度 (当对 TIMx_DMAR 寄存器进行写操作时，定时器则进行一次连续传送)，即：定义传输的次数，传输可以是半字 (双字节) 或字节： 00000：1 次传输 00001：2 次传输 00010：3 次传输..... ..... 10001：18 次传输 例：我们考虑这样的传输：DBL = 7，DBA = TIM2_CR1 - 如果 DBL = 7，DBA = TIM2_CR1 表示待传输数据的地址，那么传输的地址由下式给出： (TIMx_CR1 的地址)+ DBA +(DMA 索引)，其中 DMA 索引 = DBL 其中 (TIMx_CR1 的地址)+ DBA 再加上 7，给出了将要写入或者读出数据的地址，这样数据的传输将发生在从地址 (TIMx_CR1 的地址)+ DBA 开始的 7 个寄存器。根据 DMA 数据长度的设置，可能发生以下情况： - 如果设置数据为半字 (16 位)，那么数据就会传输给全部 7 个寄存器。 - 如果设置数据为字节，数据仍然会传输给全部 7 个寄存器：第一个寄存器包含第一个 MSB 字节，第二个寄存器包含第一个 LSB 字节，以此类推。因此对于定时器，用户必须指定由 DMA 传输的数据宽度。
7：5	Reserved			保留，始终读为 0。
4：0	DBA	w	0x00	DMA 基地址 (DMA base address) 这些位定义了 DMA 在连续模式下的基地址 (当对 TIMx_DMAR 寄存器进行写操作时)，DBA 定义为从 TIMx_CR1 寄存器所在地址开始的偏移量： 00000：TIMx_CR1 00001：TIMx_CR2 00010：TIMx_SMCR .....

11.4.20 连续模式的 DMA 地址 (TIMx_DMAR)

偏移地址：0x4C

复位值：0x0000

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

Bit	Field	Type	Reset	Description
15：0	DMAB	w	0x0000	DMA 连续传送寄存器 (DMA register for burst accesses) 对 TIMx_DMAR 寄存器的写操作会导致对以下地址所在寄存器的存取操作： TIMx_CR1 地址 + DBA + DMA 索引，其中：‘TIMx_CR1 地址’是控制寄存器 1(TIMx_CR1) 所在的地址； ‘DBA’是 TIMx_DCR 寄存器中定义的基地址； ‘DMA 索引’是由 DMA 自动控制的偏移量，它取决于 TIMx_DCR 寄存器中定义的 DBL。

12 16 位通用定时器 (TIMx16 Bit)

16 位通用定时器 (TIMx16 Bit)

12.1 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)。

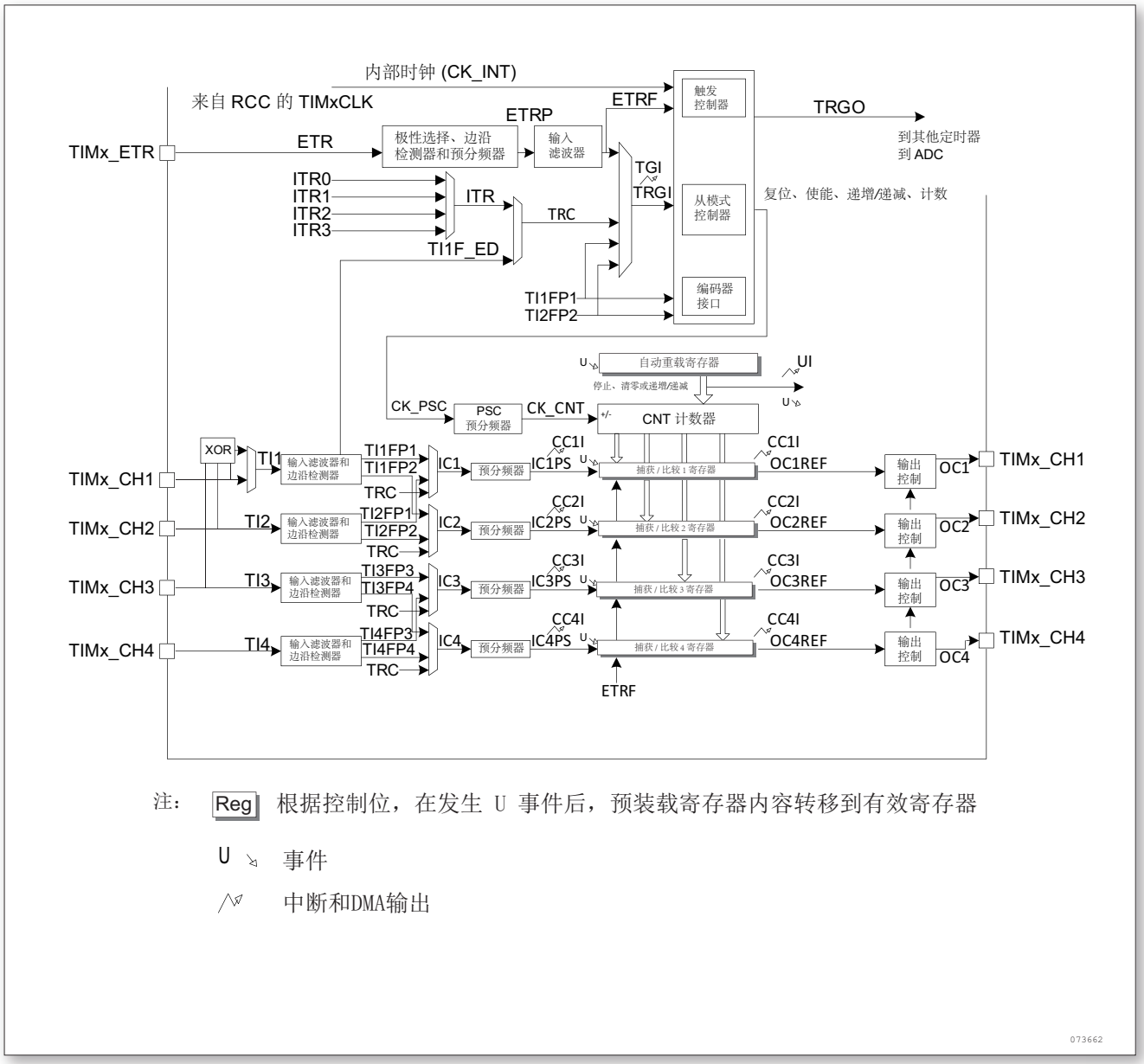
使用定时器预分频器和 RCC 时钟控制器预分频器，脉冲长度和波形周期可以在几个微秒到几个毫秒间调整。

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

12.2 TIMx 主要功能

通用 TIMx(TIM2TIM3TIM4) 定时器功能包括：

- 16 位向上、向下、向上/向下自动装载计数器
- 16 位可编程 (可以实时修改) 预分频器，计数器时钟频率的分频系数为 1 ~ 65536 之间的任意数值
- 4 个独立的通道
  - 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



注： Reg 根据控制位，在发生 U 事件后，预装载寄存器内容转移到有效寄存器

U 事件

中断和DMA输出

图 76. 通用定时器框图

12.3 TIMx 功能描述

12.3.1 时基单元

可编程通用定时器的主要部分是一个 16 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)。

预分频器描述

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

下面两个图分别给出了在预分频器运行时，更改计数器参数的例子。

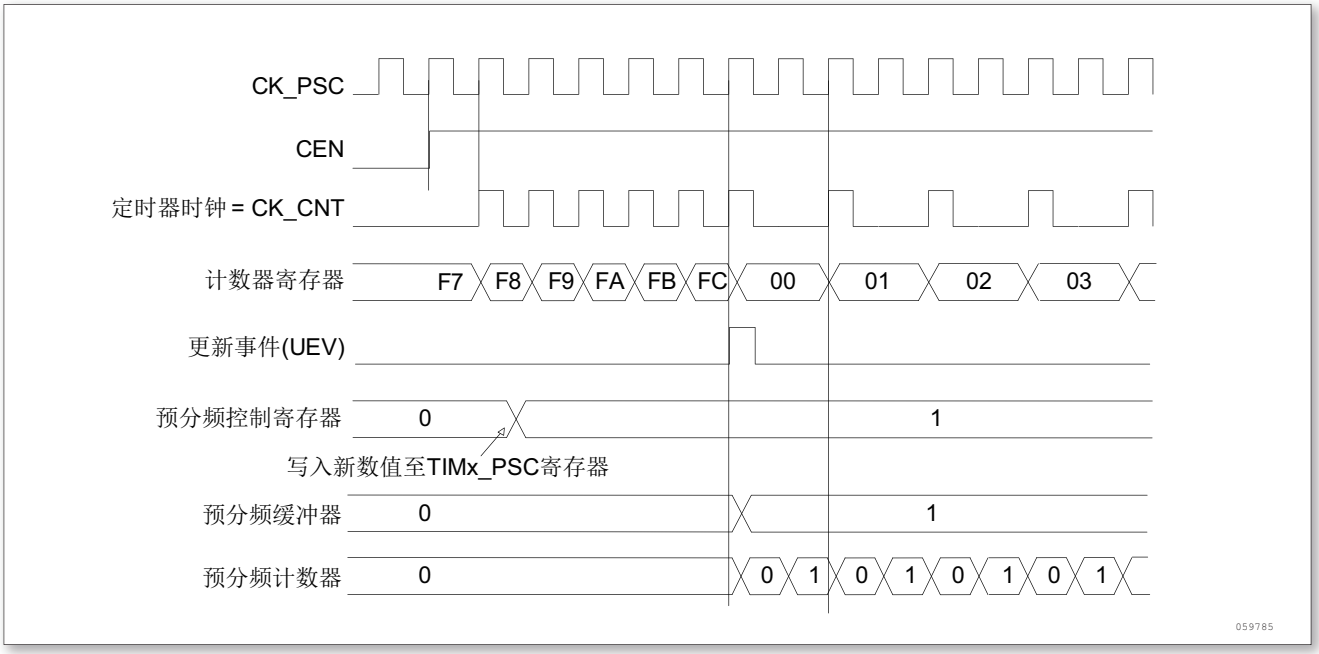


图 77. 当预分频器的参数从 1 变到 2 时，计数器的时序图

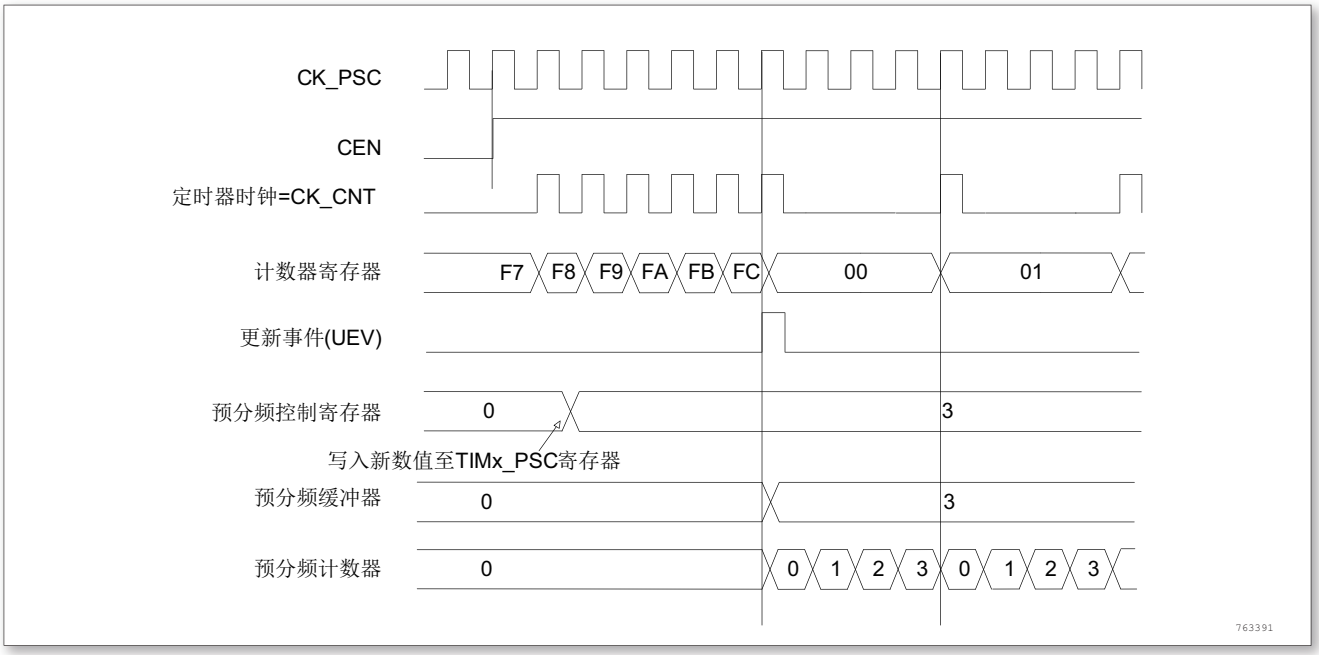


图 78. 当预分频器的参数从 1 变到 4 时，计数器的时序图

12.3.2 计数模式

向上计数模式

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

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

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

当发生一个更新事件时，所有的寄存器都被更新，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：

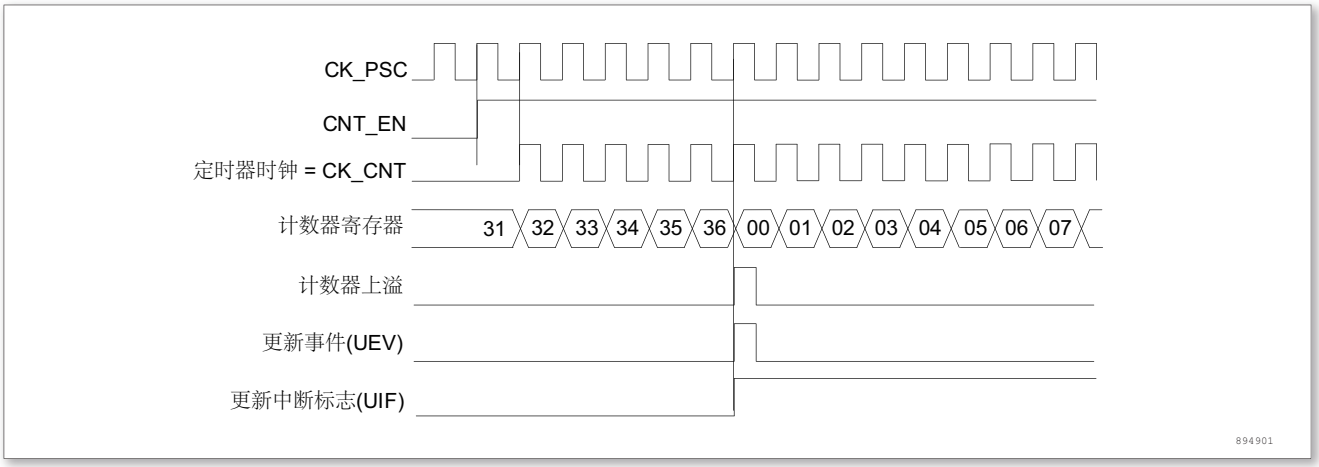


图 79. 计数器时序图，内部时钟分频因子为 1

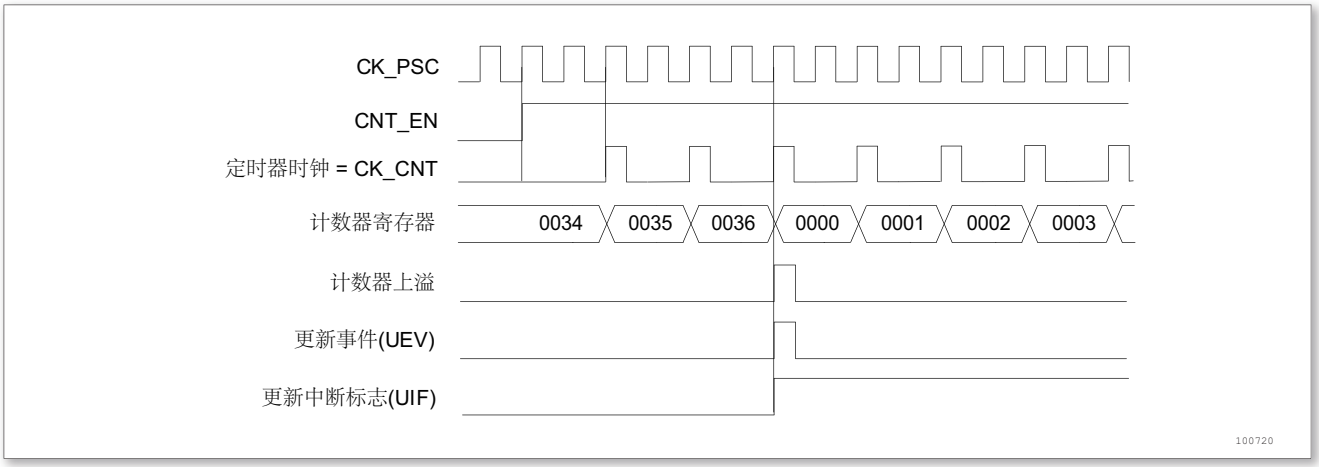


图 80. 计数器时序图，内部时钟分频因子为 2

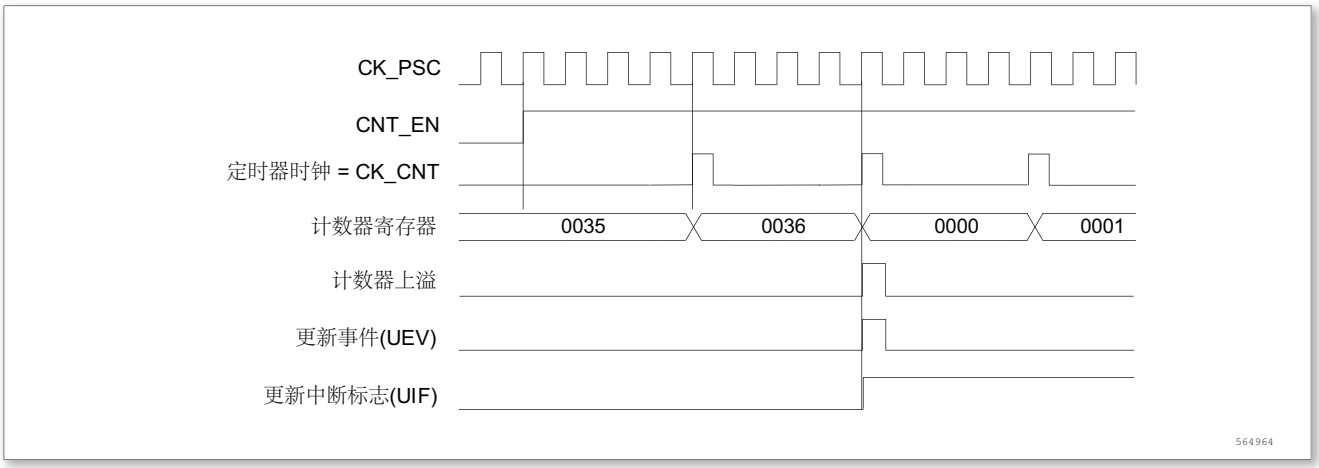


图 81. 计数器时序图，内部时钟分频因子为 4

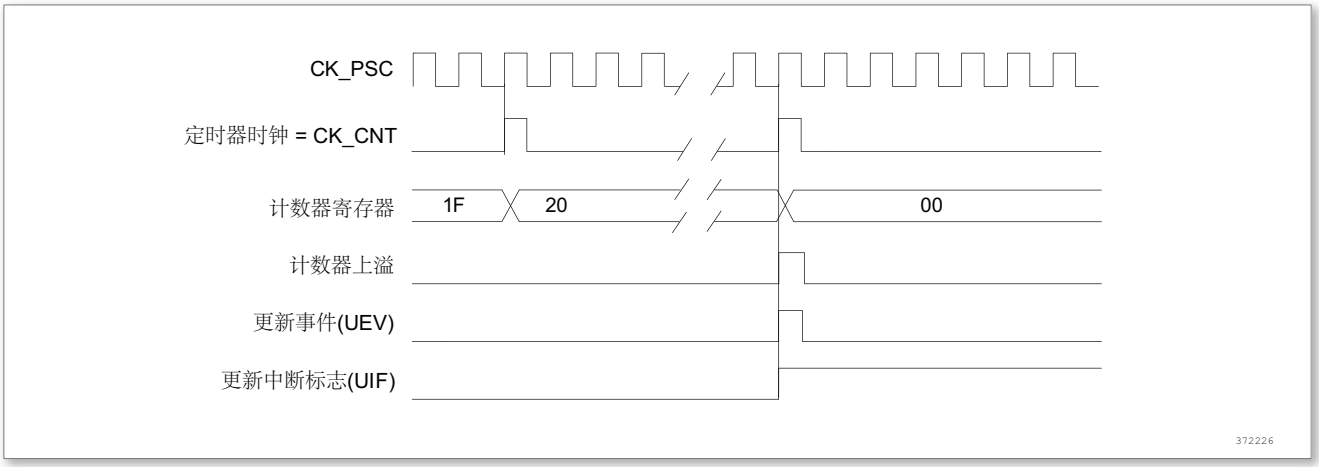


图 82. 计数器时序图，内部时钟分频因子为 N

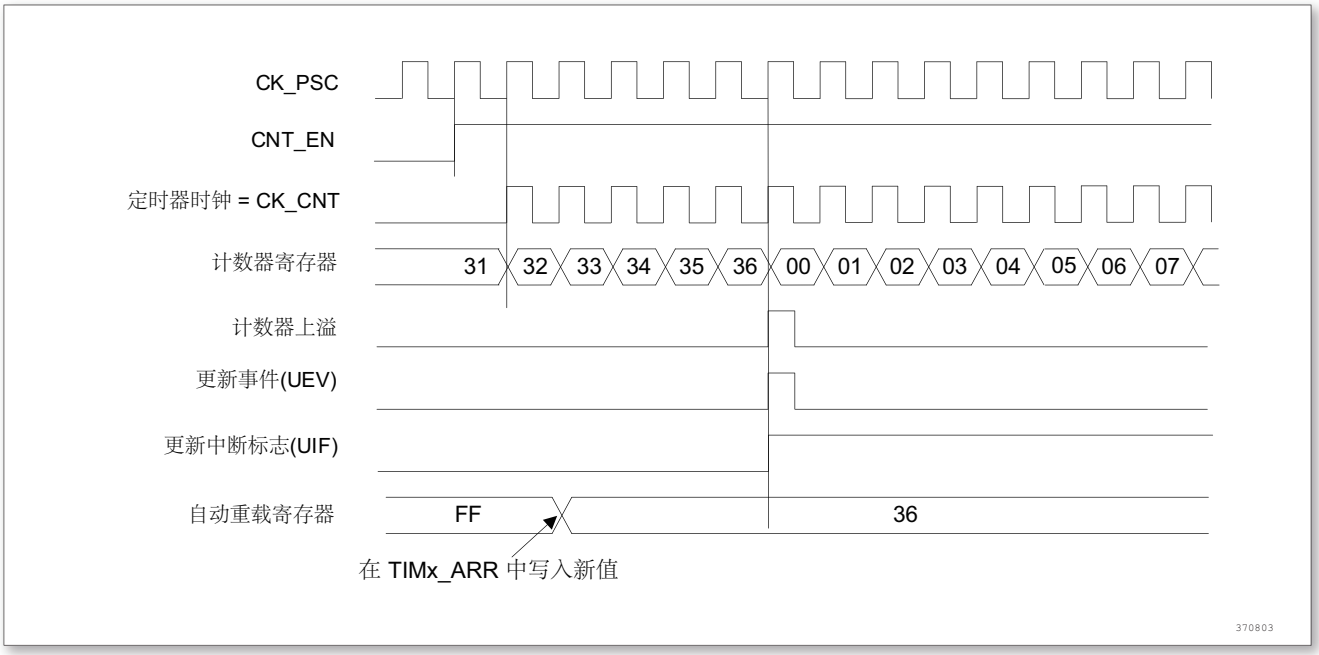


图 83. 计数器时序图，当 ARPE = 0 时的更新事件 (TIMx_ARR 没有预装入)

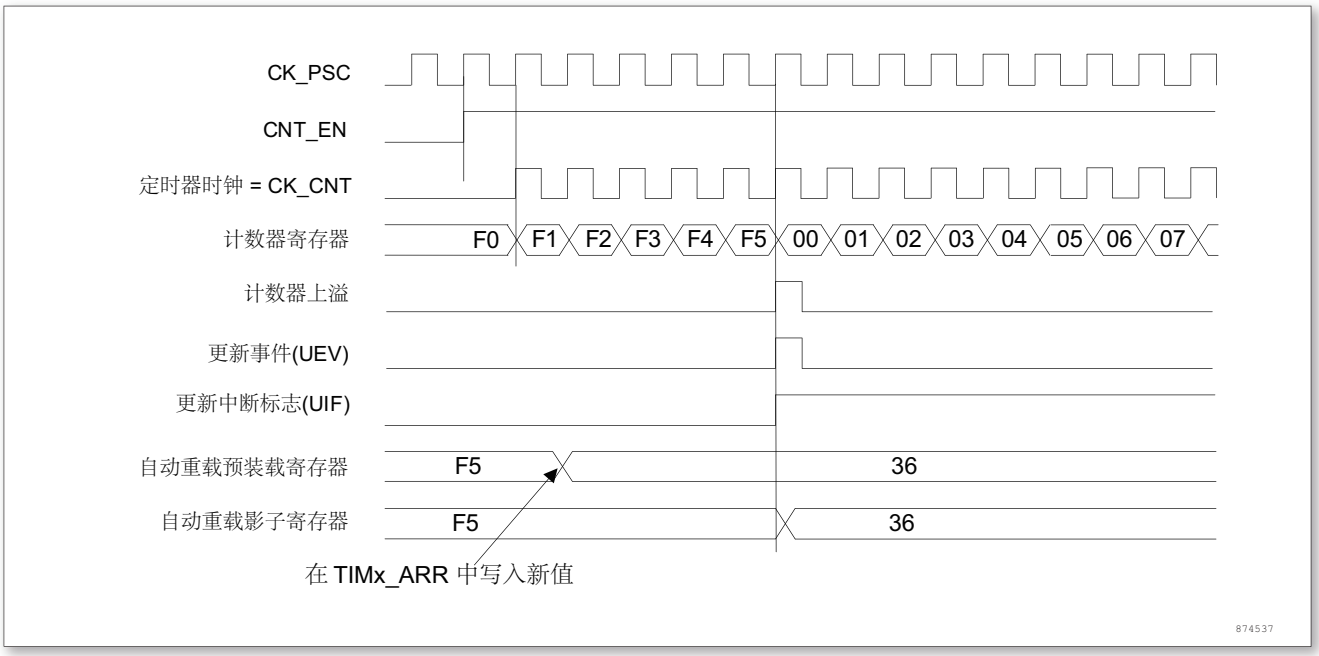


图 84. 计数器时序图，当 ARPE = 1 时的更新事件 (预装入了 TIMx_ARR)

向下计数模式

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

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

设置 TIMx_CR1 寄存器的 UDIS 位可以禁止 UEV 事件。这样可以避免向预装载寄存器中写入新值时更新影子寄存器。因此 UDIS 位被清为 0 之前不会产生更新事件。然而，计数器仍会从当前自动加载值重新开始计数，同时预分频器的计数器重新从 0 开始 (但预分频器的速率不能被修改)。

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

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

- 预分频器的缓存器被置入预装载寄存器的值 (TIMx_PSC 寄存器的值)。
- 当前的自动加载寄存器被更新为预装载值 (TIMx_ARR 寄存器中的内容)。

注：自动装载在计数器重载入之前被更新，因此下一个周期将是预期的值。

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

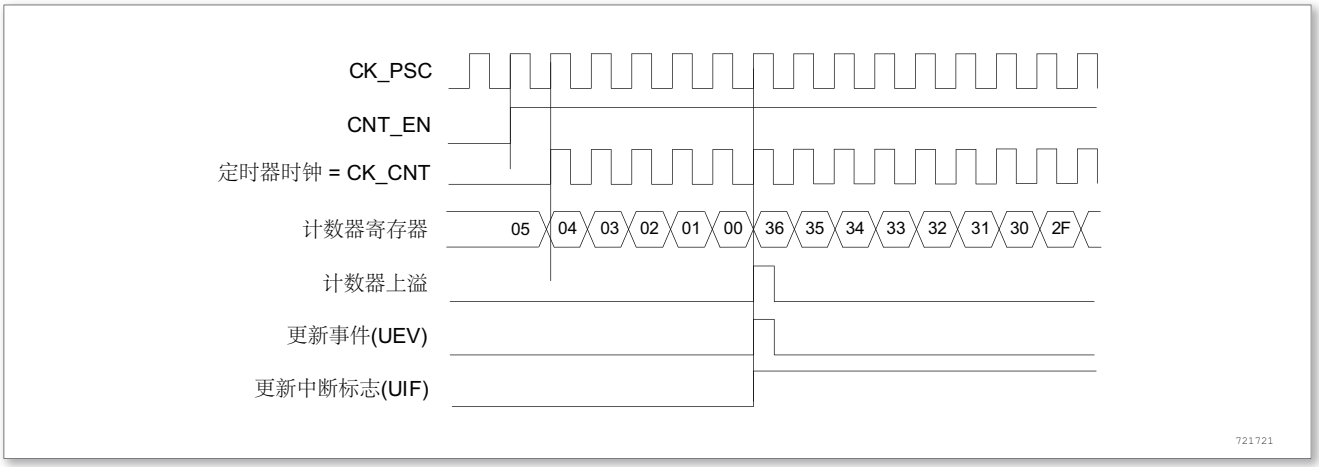


图 85. 计数器时序图，内部时钟分频因子为 1

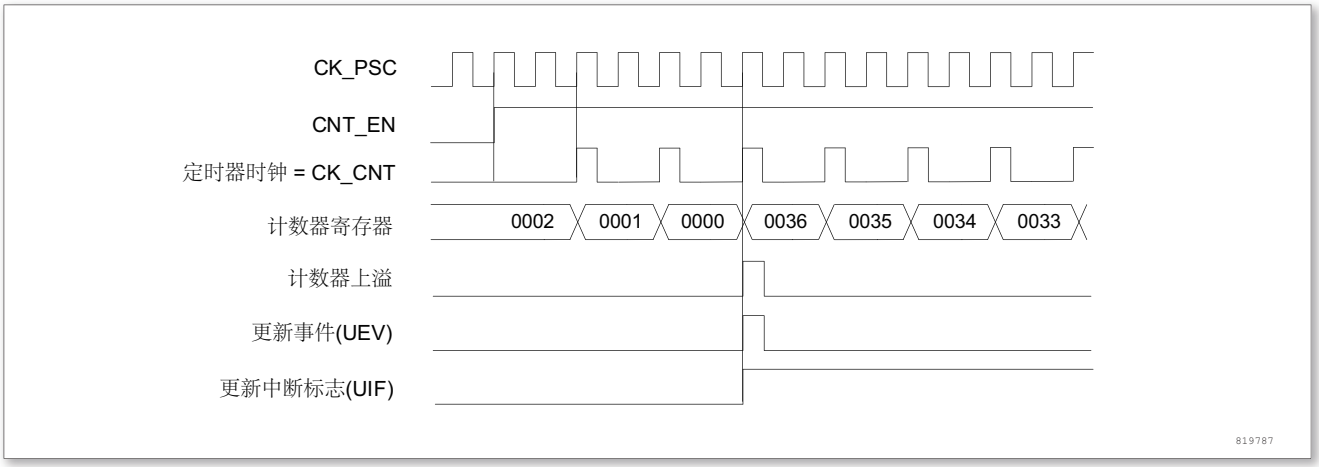


图 86. 计数器时序图，内部时钟分频因子为 2

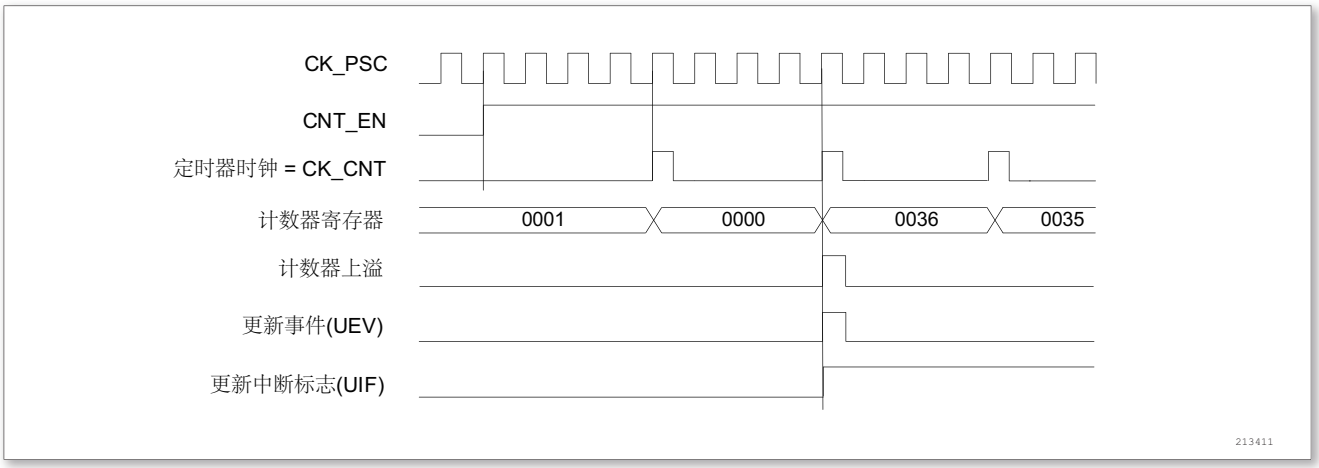


图 87. 计数器时序图，内部时钟分频因子为 4

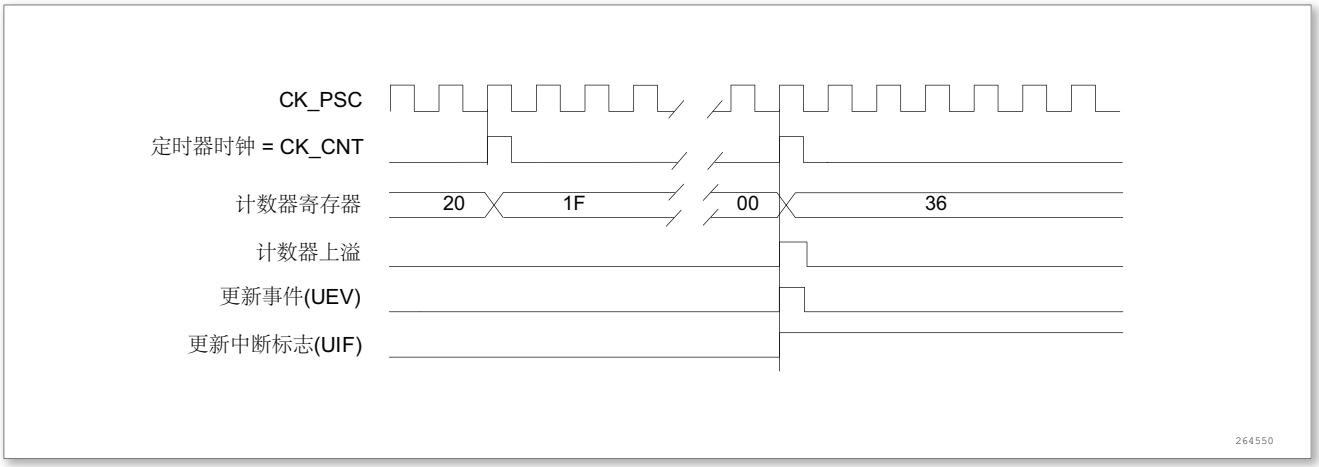


图 88. 计数器时序图，内部时钟分频因子为 N

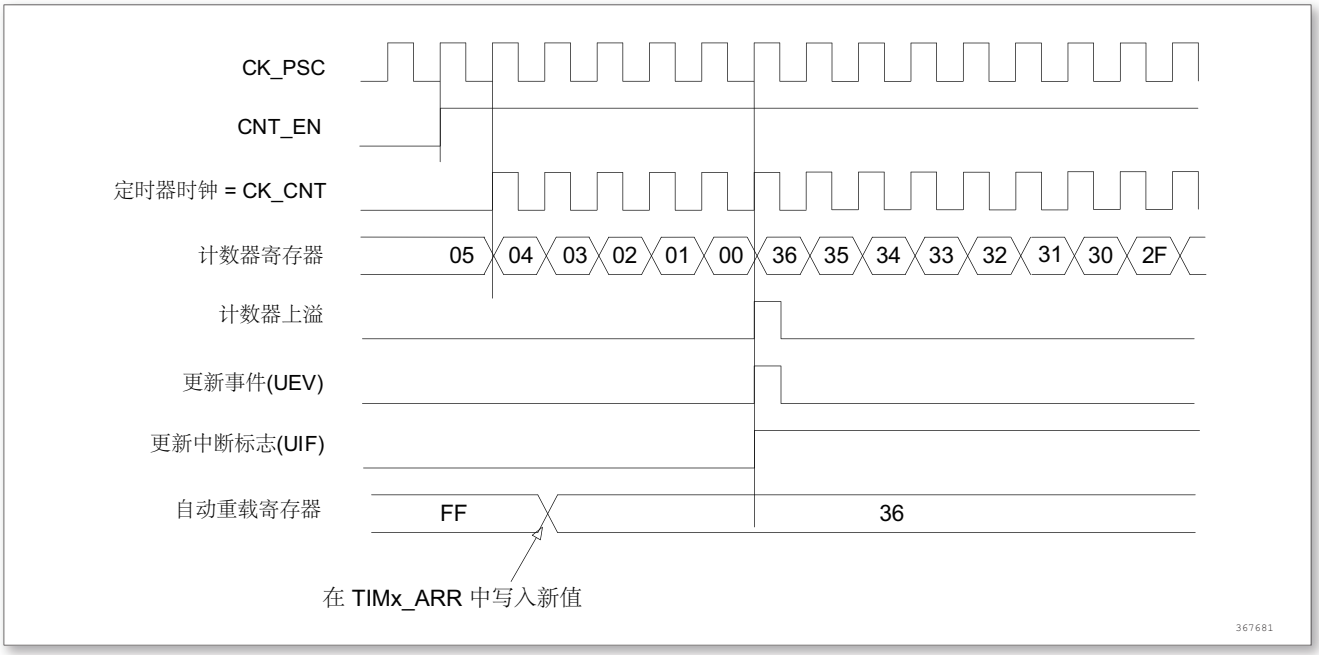


图 89. 计数器时序图，当没有使用重复计数器时的更新事件

中央对齐模式 (向上/向下计数)

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

在这个模式，不能写入 TIMx_CR1 中的 DIR 方向位。它由硬件更新并指示当前的计数方向。更新事件可以产生在每次计数溢出和每次计数下溢；也可以通过 (软件或者使用从模式控制器) 设置 TIMx_EGR 寄存器中的 UG 位产生，此时，计数器重新从 0 开始计数，预分频器也重新从 0 开始计数。

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

此外，如果设置了 TIMx_CR1 寄存器中的 URS 位 (选择更新请求)，设置 UG 位将产生一个更新事件 UEV 但不设置 UIF 标志 (因此不产生中断和 DMA 请求)，这是为了避免在发生

捕获事件并清除计数器时，同时产生更新和捕获中断。

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

- 预分频器的缓存器被加载为预装载 (TIMx_PSC 寄存器) 的值
- 当前的自动加载寄存器被更新为预装载值 (TIMx_ARR 寄存器中的内容)

注：如果因为计数器溢出而产生更新，自动重装载将在计数器重载入之前被更新，因此下一个周期将是预期的值 (计数器被装载为新的值)。

以下是一些计数器在不同时钟频率下的操作的例子：

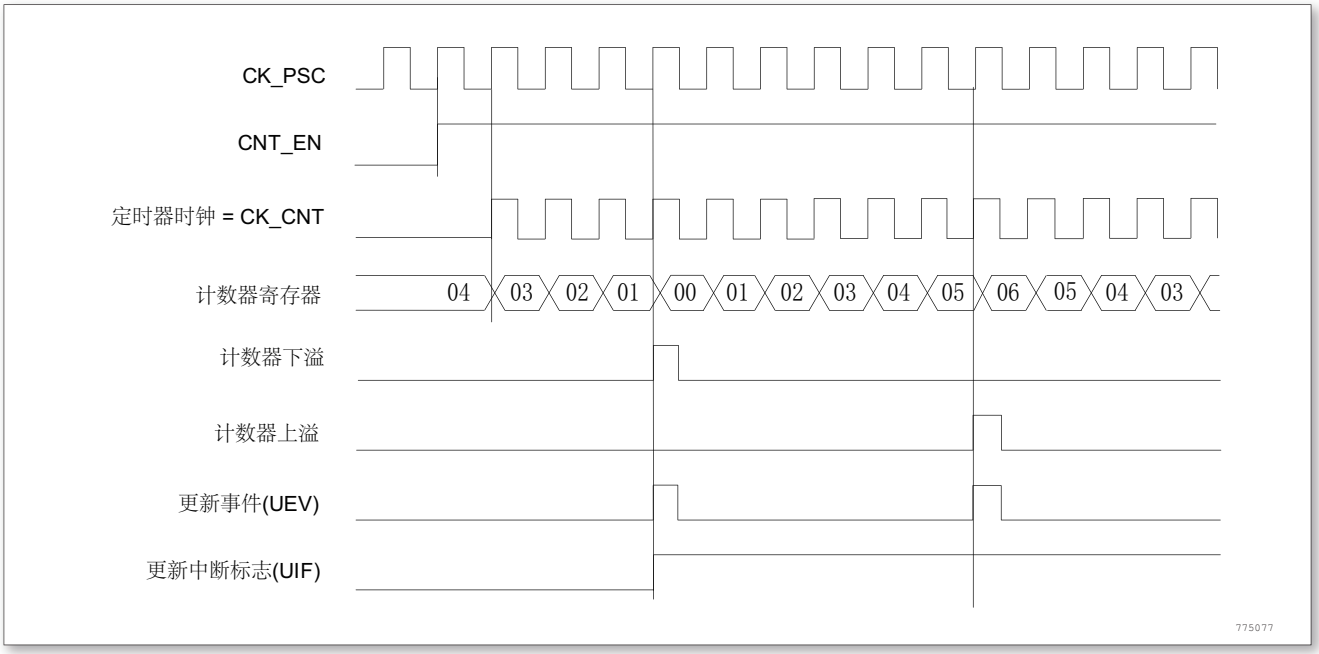


图 90. 计数器时序图，内部时钟分频因子为 1，TIMx_ARR = 0x6

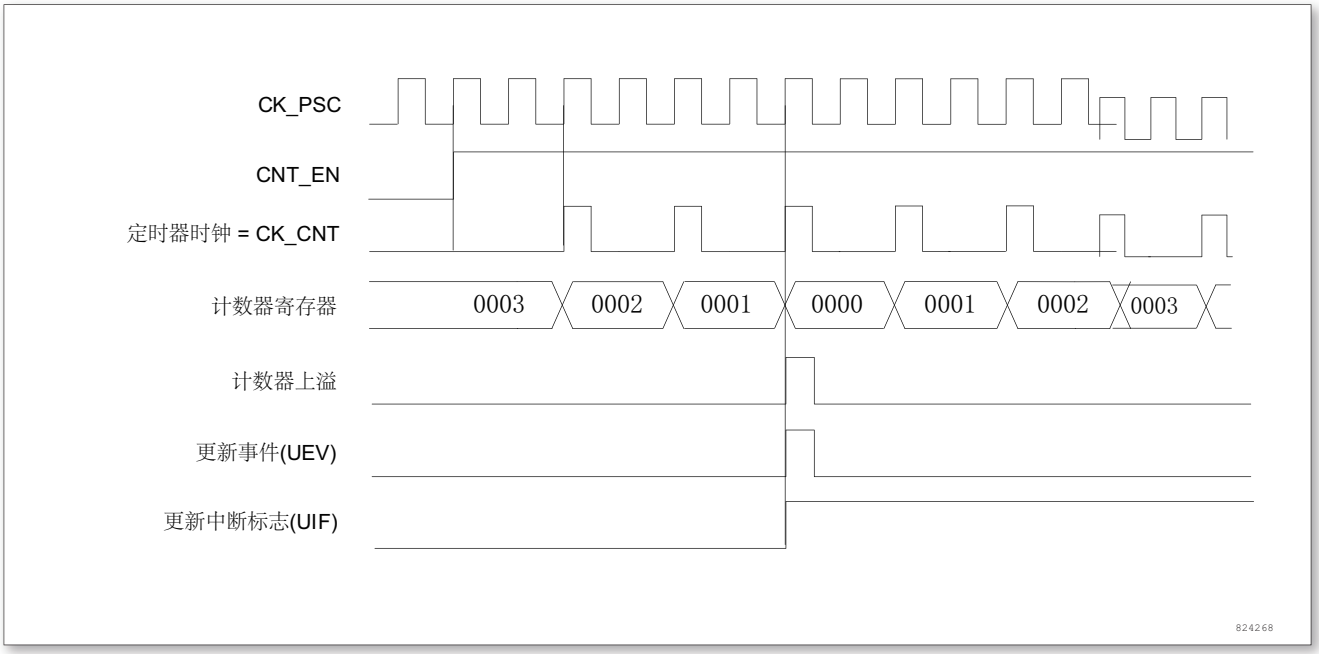
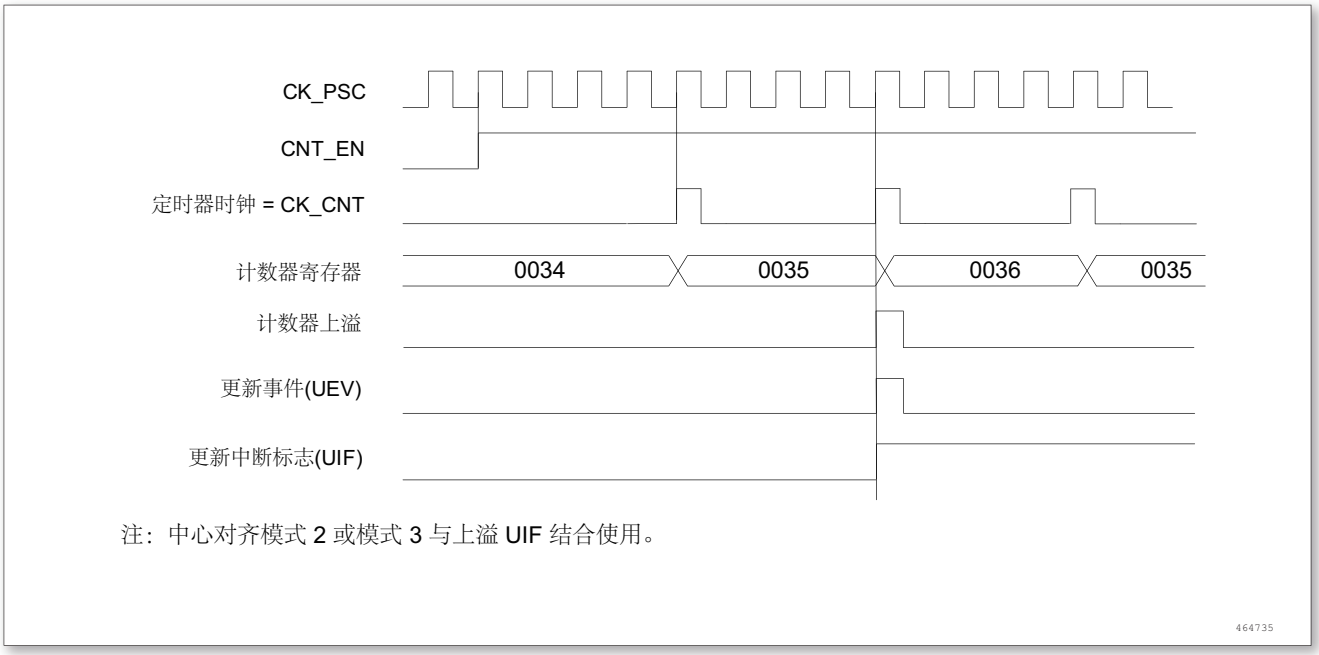


图 91. 计数器时序图，内部时钟分频因子为 2



注：中心对齐模式 2 或模式 3 与上溢 UIF 结合使用。

图 92. 计数器时序图，内部时钟分频因子为 4，TIMx_ARR = 0x36

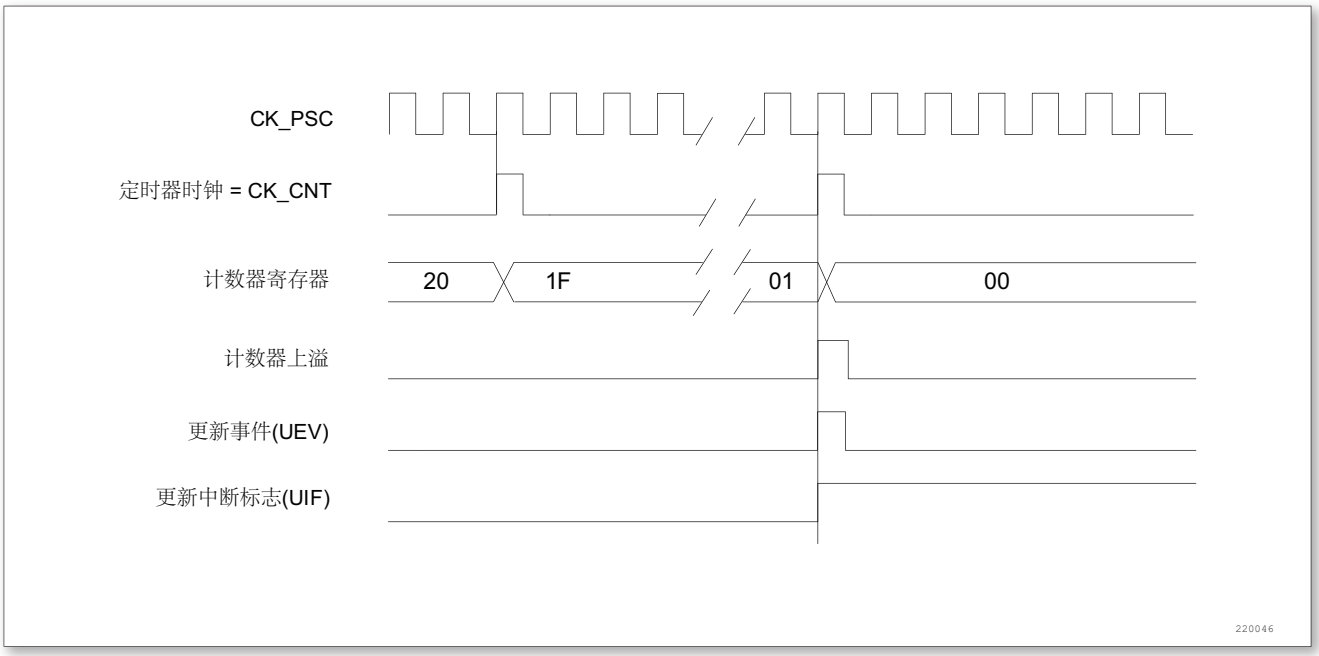


图 93. 计数器时序图，内部时钟分频因子为 N

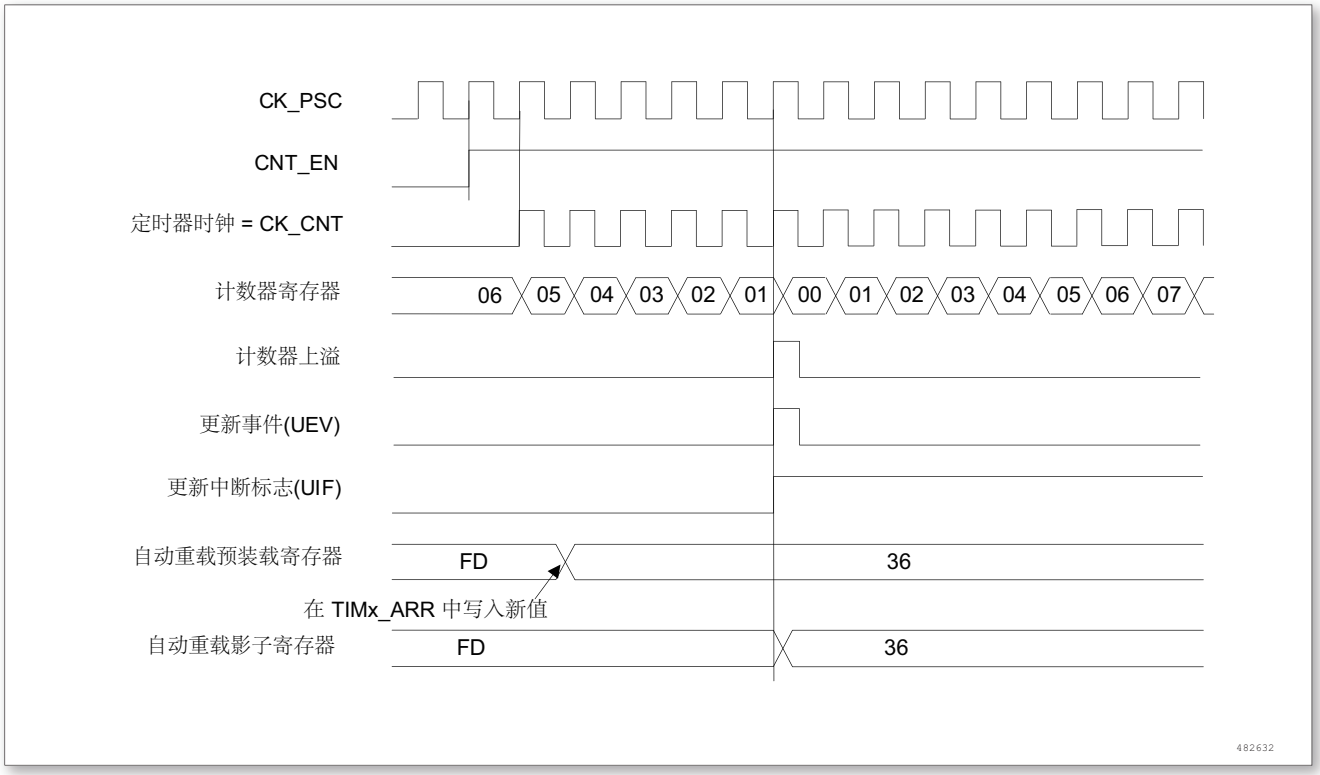


图 94. 计数器时序图，ARPE = 1 时的更新事件 (计数器下溢)

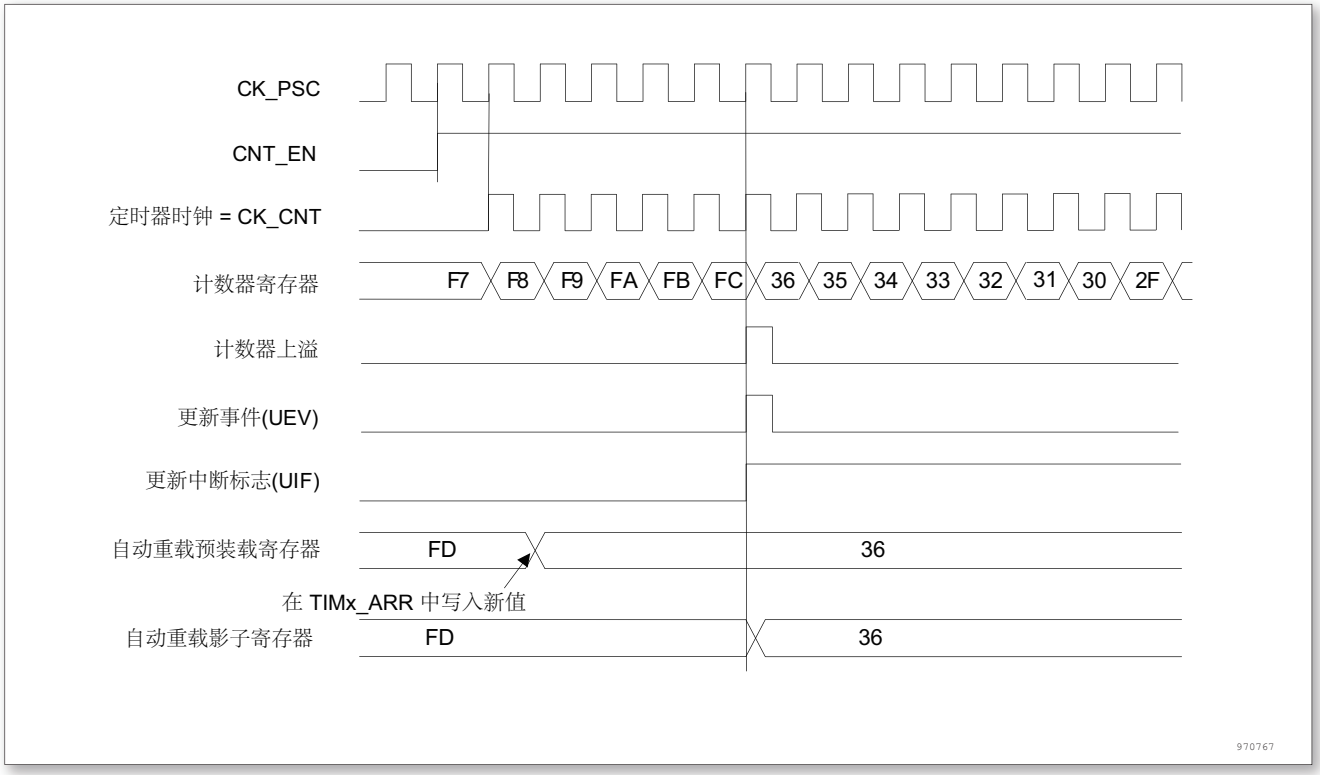


图 95. 计数器时序图，ARPE = 1 时的更新事件 (计数器溢出)

12.3.3 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 作为另一个定时器 Timer2 的预分频器。

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)。当 CEN 位被写成 1 时，预分频器的时钟由内部时钟 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。

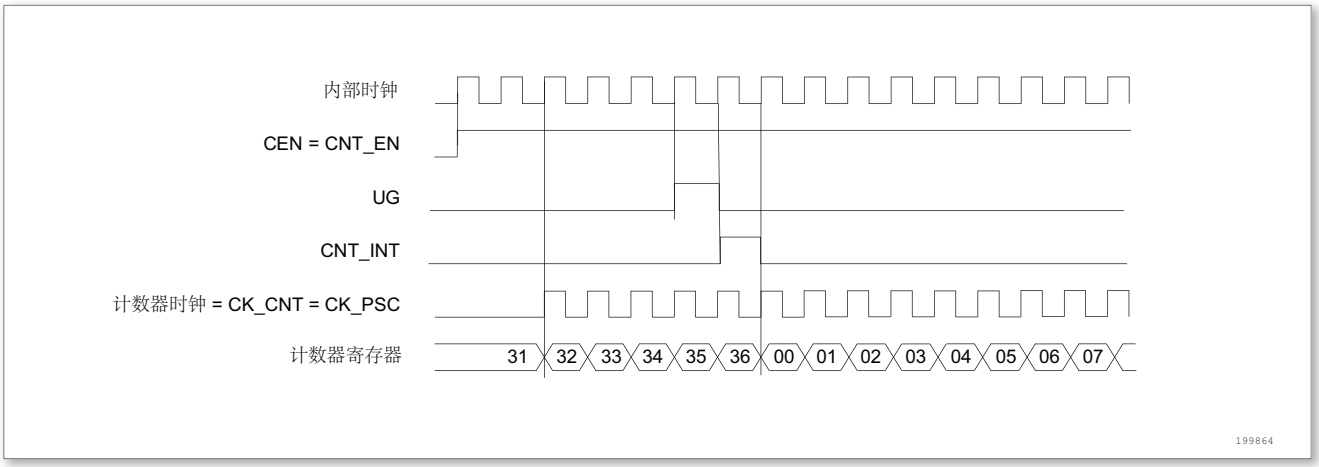


图 96. 一般模式下的控制电路，内部时钟分频因子为 1

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。

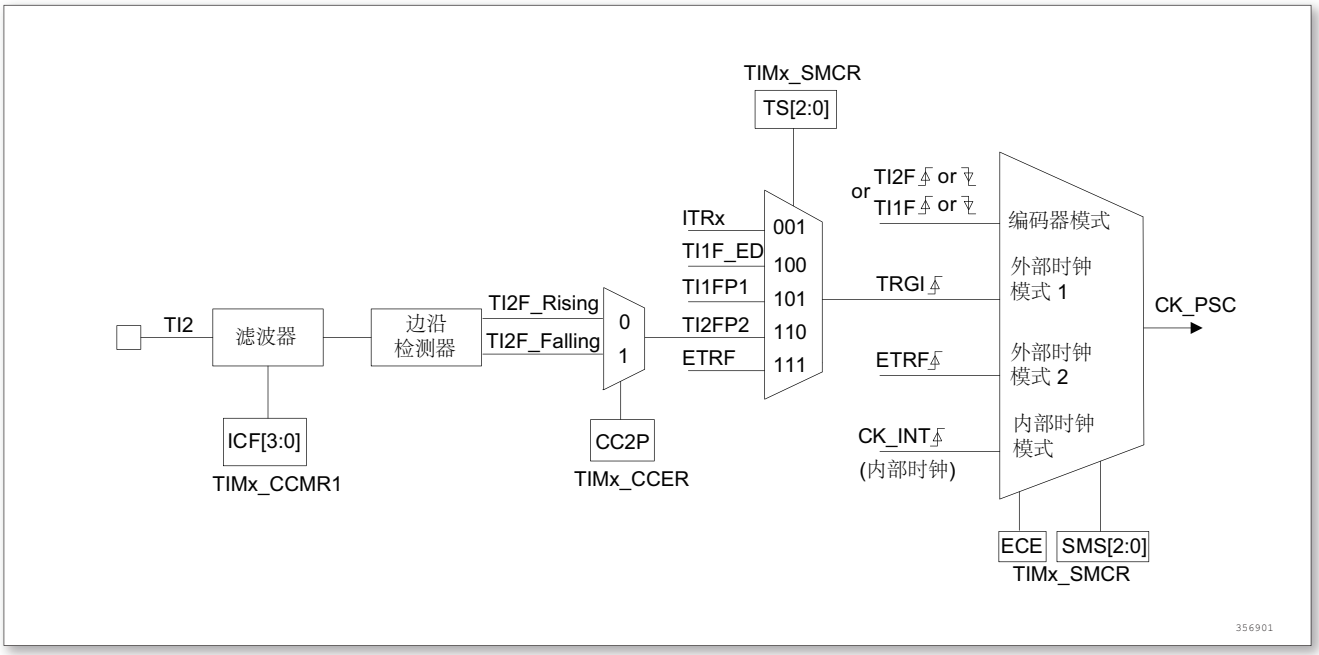


图 97. TI2 外部时钟连接例子

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。

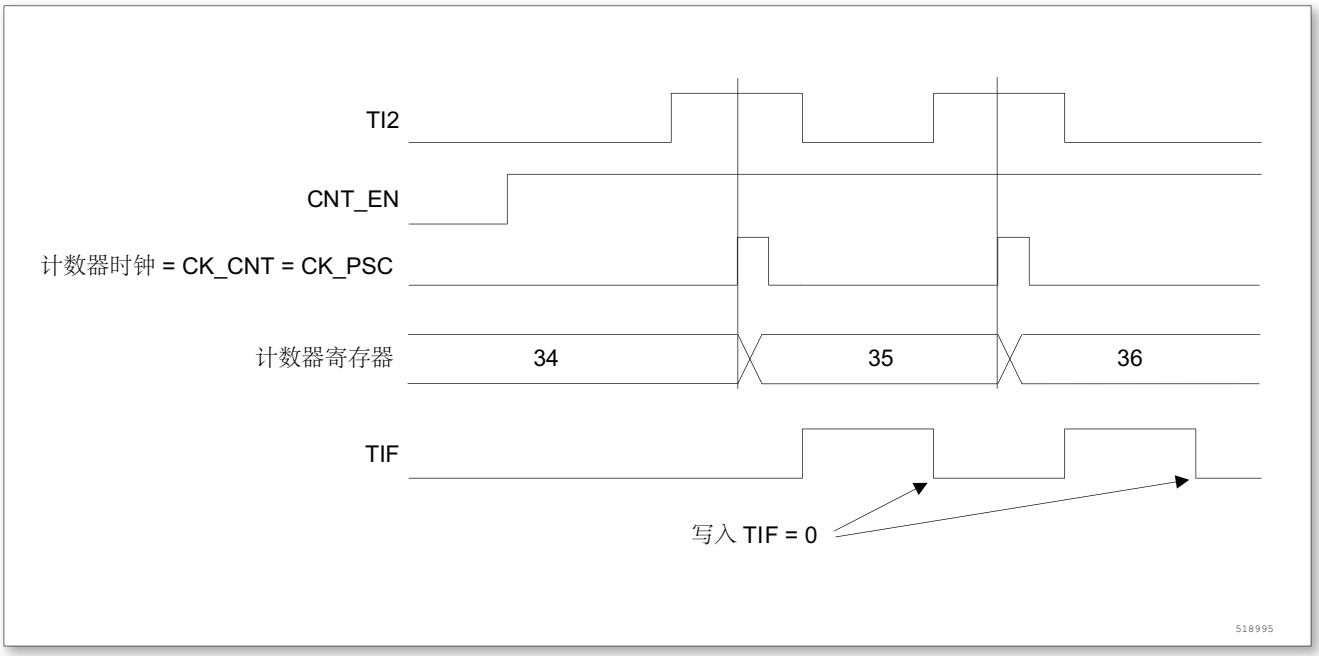


图 98. 外部时钟模式 1 下的控制电路

外部时钟源模式 2

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

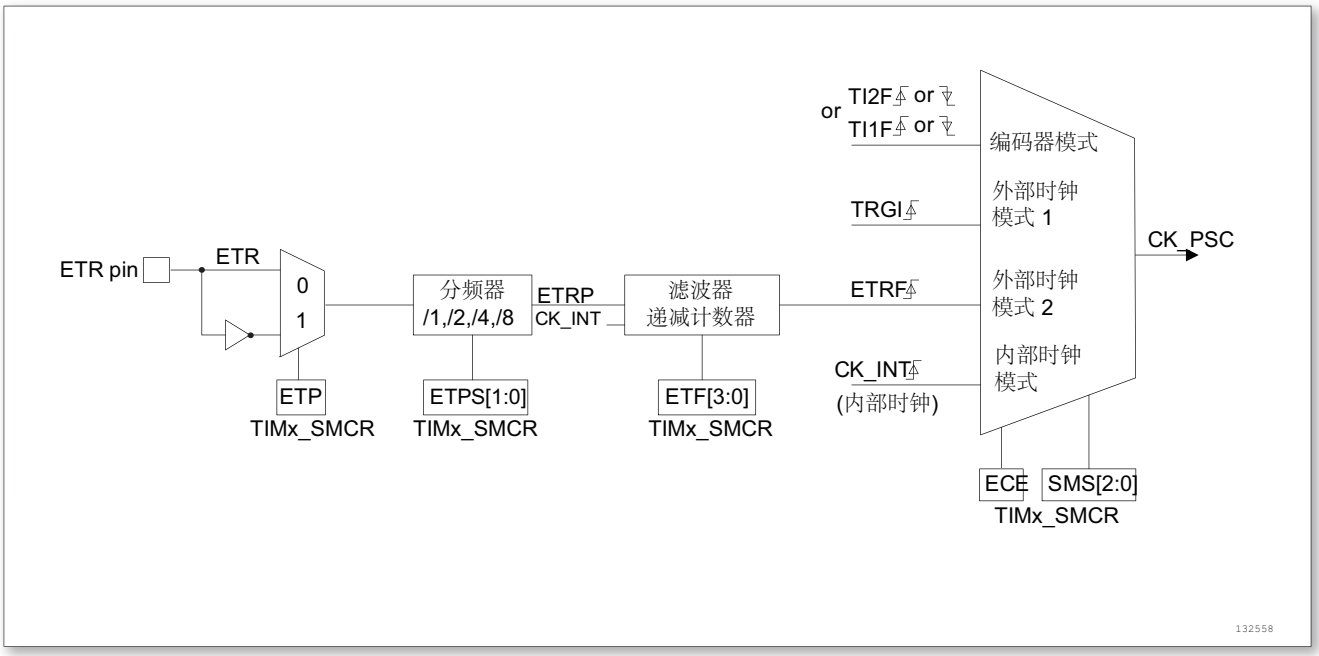


图 99. 外部触发输入框图

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。

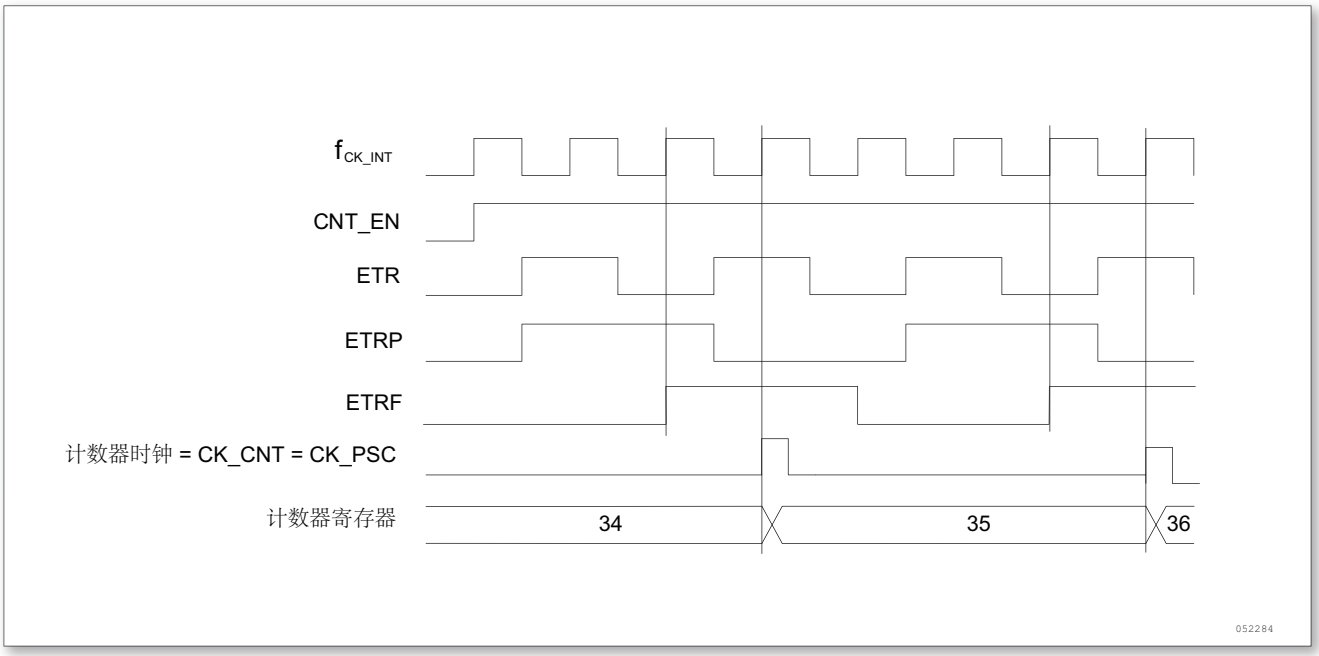


图 100. 外部时钟模式 2 下的控制电路

12.3.4 捕获/比较通道

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

下面几张图是一个捕获/比较通道概览。输入部分对相应的 TIx 输入信号采样，并产生一个滤波后的信号 TIxF。然后，一个带极性选择的边缘监测器产生一个信号 (TIxFPx)，它可以作为从模式控制器的输入触发或者作为捕获控制。该信号通过预分频进入捕获寄存器 (ICxPS)。

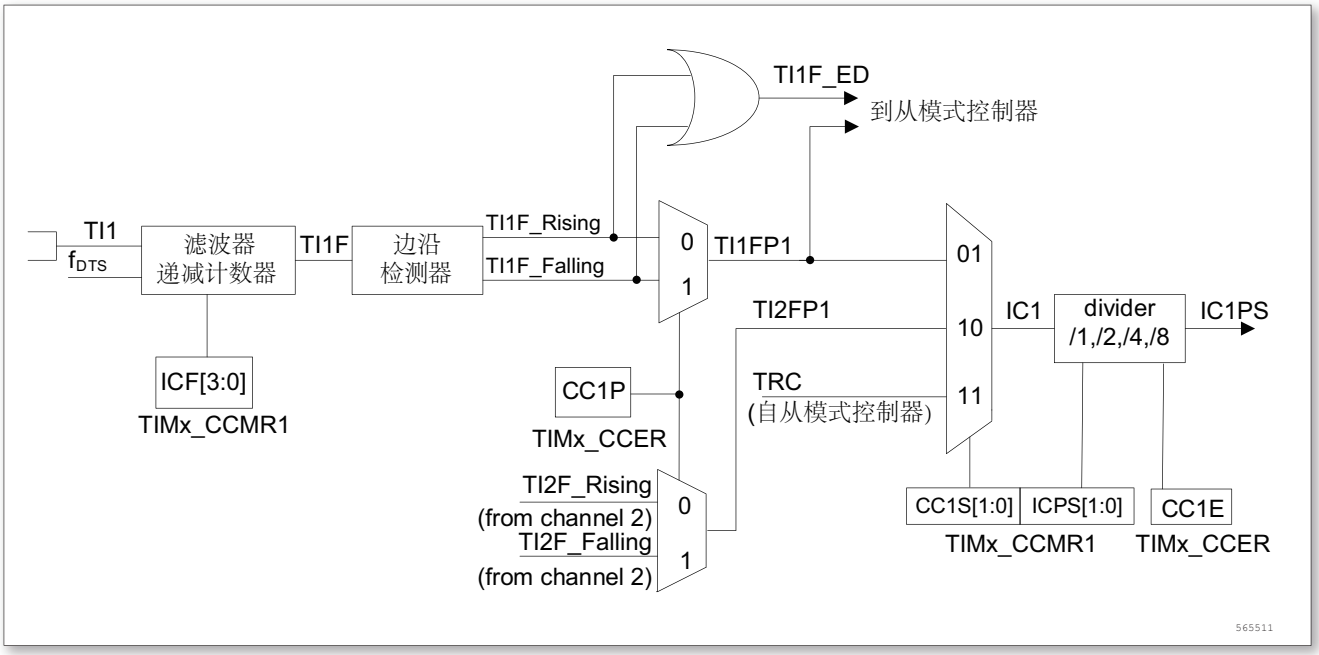


图 101. 捕获/比较通道 (如：通道 1 输入部分)

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。

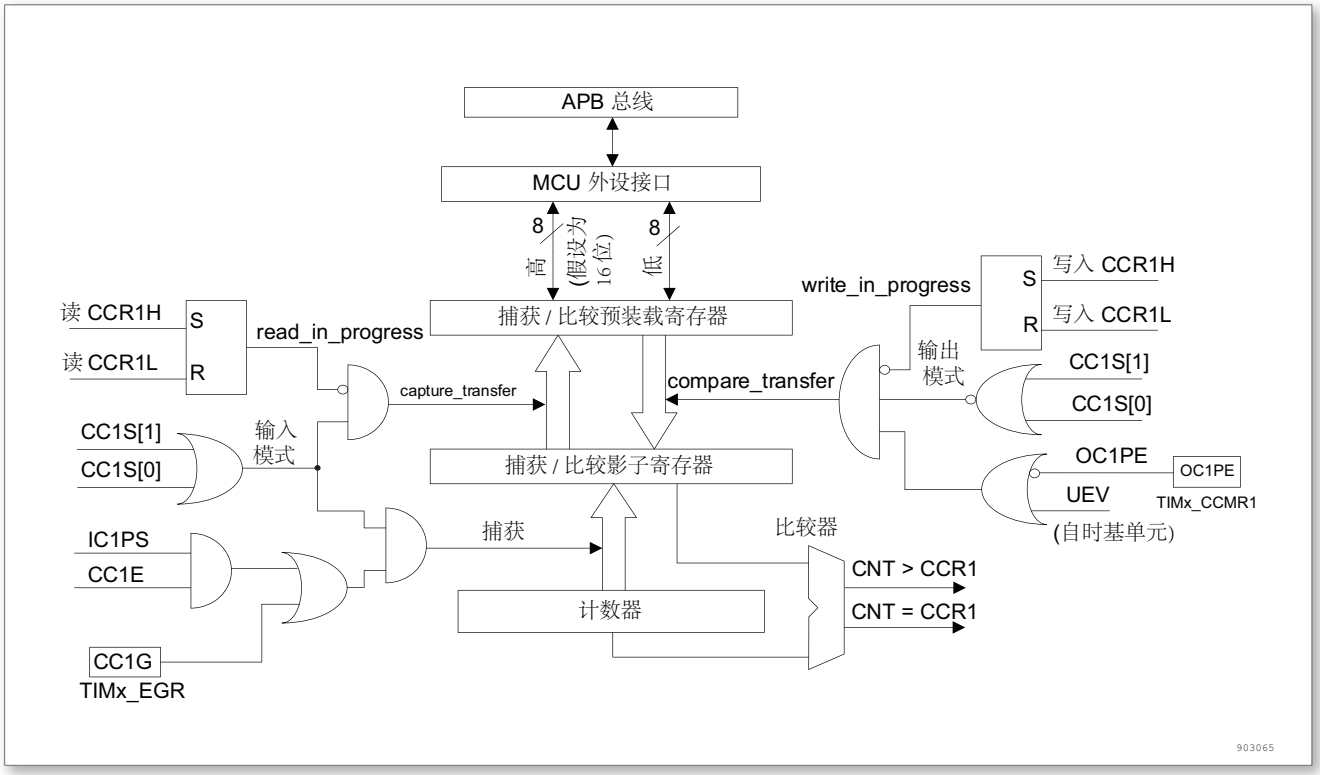


图 102. 捕获/比较通道 1 的主电路

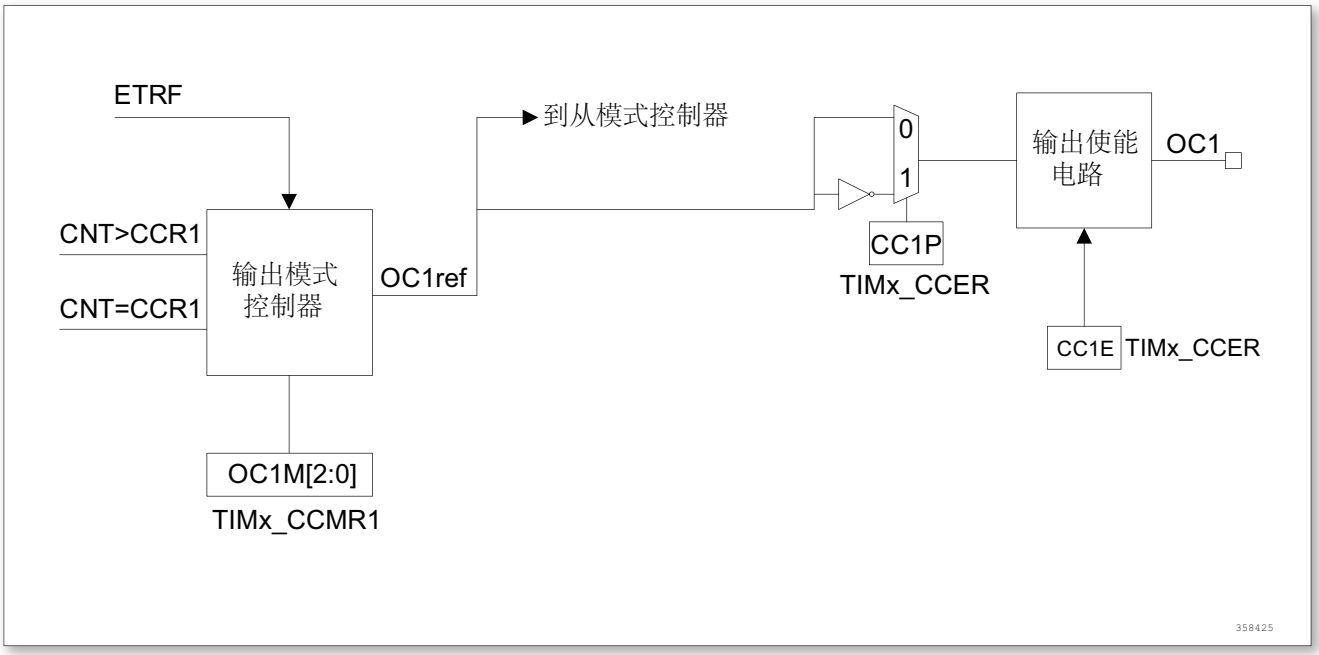


图 103. 捕获/比较通道的输出部分 (通道 1)

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。

12.3.5 输入捕获模式

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

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

- 选择有效输入端：TIMx_CCR1 必须连接到 TI1 输入，所以写入 TIMx_CCR1 寄存器中的 CC1S = 01，当 CC1S 不为 00 时，通道被配置为输入，并且 TM1_CCR1 寄存器变为只读。
- 根据输入信号的特点，配置输入滤波器为所需的带宽 (即输入为 TIx 时，输入滤波器控制位是 TIMx_CCMRx 寄存器中的 ICxF 位)。假设输入信号在最多 5 个时钟周期的时间内抖动，我们须配置滤波器的带宽长于 5 个时钟周期。因此我们可以 (以 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。

当一个输入捕获时:

- 当产生有效的电平转换时，计数器的值被传送到 TIMx_CCR1 寄存器。
- CC1IF 标志被设置 (中断标志)。当发生至少 2 个连续的捕获时，而 CC1IF 未曾被清除。
- CC1OF 也被置 1。
- 如设置了 CC1IE 位，则会产生一个中断。
- 如设置了 CC1DE 位，则还会产生一个 DMA 请求。

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

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

12.3.6 PWM 输入模式

该模式是输入捕获模式的一个特例，除下列区别外，操作与输入捕获模式相同：

- 两个 ICx 信号被映射同一个 TIx 输入
- 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。

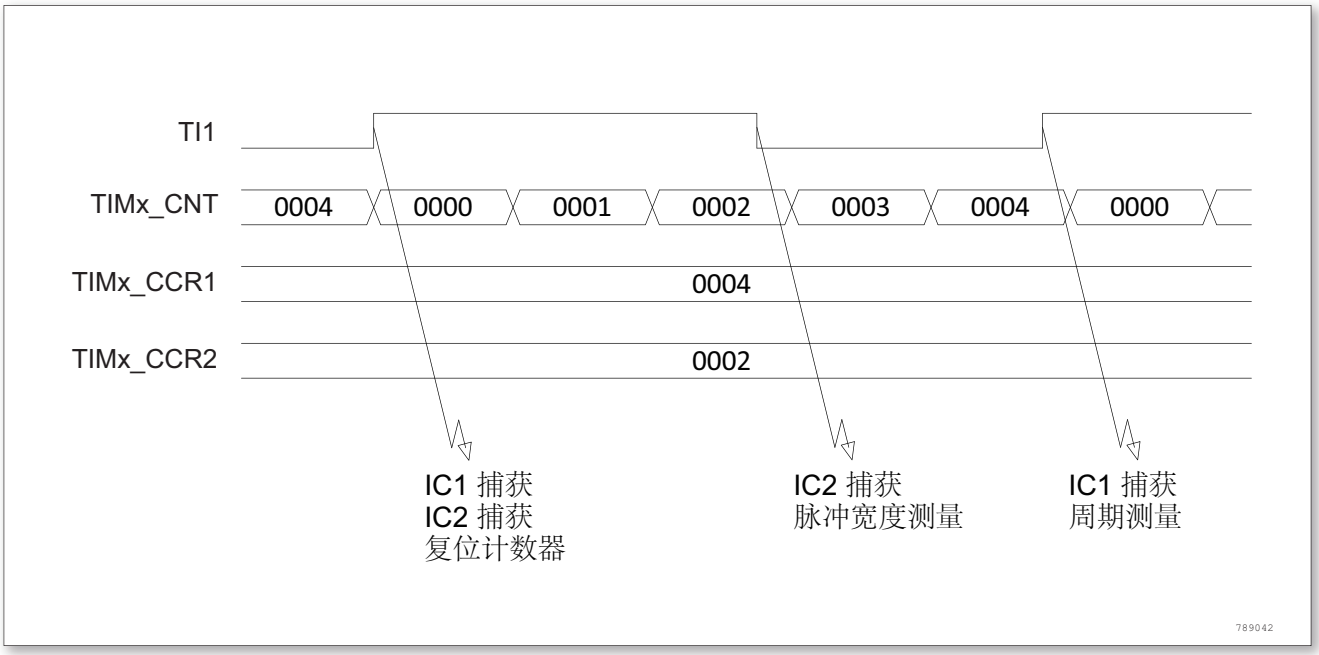


图 104. 捕获/比较通道的输出部分 (通道 1)

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。

12.3.7 强制输出模式

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。

12.3.8 输出比较模式

此项功能是用来控制一个输出波形或者指示何时一段给定的的时间已经到时。

当计数器与捕获/比较寄存器的内容相同时，输出比较功能做如下操作：

- 将输出比较模式 (TIMx_CCMRx 寄存器中的 OCxM 位) 和输出极性 (TIMx_CCER 寄存器中的 CCxP 位) 定义的值输出到对应的管脚上。在比较匹配时，输出管脚可以保持它

的电平 (OCxM = 000)、被设置成有效电平 (OCxM = 001)、被设置成无有效电平 (OCxM = 010) 或进行翻转 (OCxM = 011)。

- 设置中断状态寄存器中的标志位 (TIMx_SR 寄存器中的 CCxIF 位)。
- 若设置了相应的中断屏蔽 (TIMx_DIER 寄存器中的 CCXIE 位)，则产生一个中断。
- 若设置了相应的使能位 (TIMx_DIER 寄存器中的 CCxDE 位，TIMx_CR2 寄存器中的 CCDS 位选择 DMA 请求功能)，则产生一个 DMA 请求。

TIMx_CCMRx 中的 OCxPE 位选择 TIMx_CCRx 寄存器是否需要使用预装载寄存器。在输出比较模式下，更新事件 UEV 对 OCxREF 和 OCx 输出没有影响。同步的精度可以达到计数器的一个计数周期。输出比较模式 (在单脉冲模式下) 也能用来输出一个单脉冲。

输出比较模式的配置步骤：

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

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

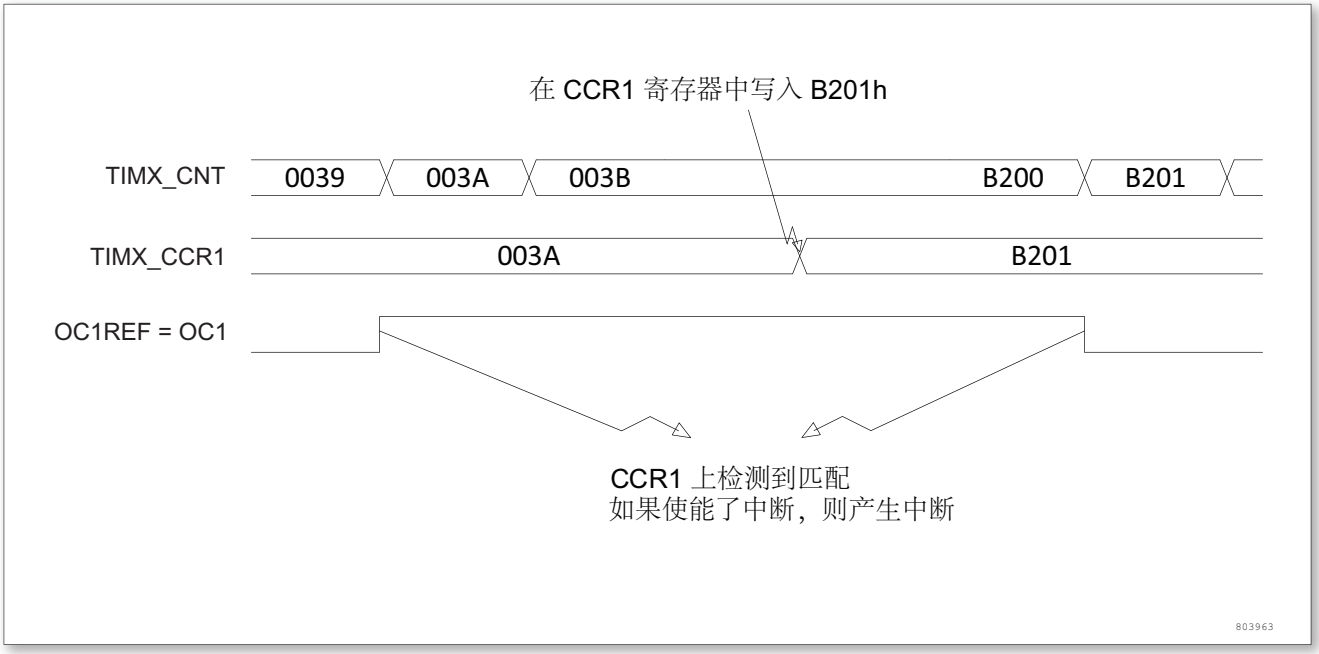


图 105. 输出比较模式，翻转 OC1

12.3.9 PWM 模式

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

在 TIMx_CCMRx 寄存器中的 OCxM 位写入‘110’(PWM 模式 1) 或‘111’(PWM 模式 2)，能够独立地设置每个 OCx 输出通道产生一路 PWM。必须设置 TIMx_CCMRx 寄存器

OCxPE 位以使能相应的预装载寄存器，最后还要设置 TIMx_CR1 寄存器的 ARPE 位使能自动重装载的预装载寄存器 (在向上计数或中心对称模式中)。

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

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

在 PWM 模式 (模式 1 或模式 2) 下，TIMx_CNT 和 TIM1_CCRx 始终在进行比较，(依据计数器的计数方向) 以确定是否符合 TIM1_CCRx ≤ TIM1_CNT 或者 TIM1_CNT ≤ TIM1_CCRx。然而为了与 OCREF_CLR 的功能 (在下一个 PWM 周期之前，ETR 信号上的一个外部事件能够清除 OCxREF) 一致，OCxREF 信号只能在下述条件下产生：

- 当比较的结果改变
- 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。

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。

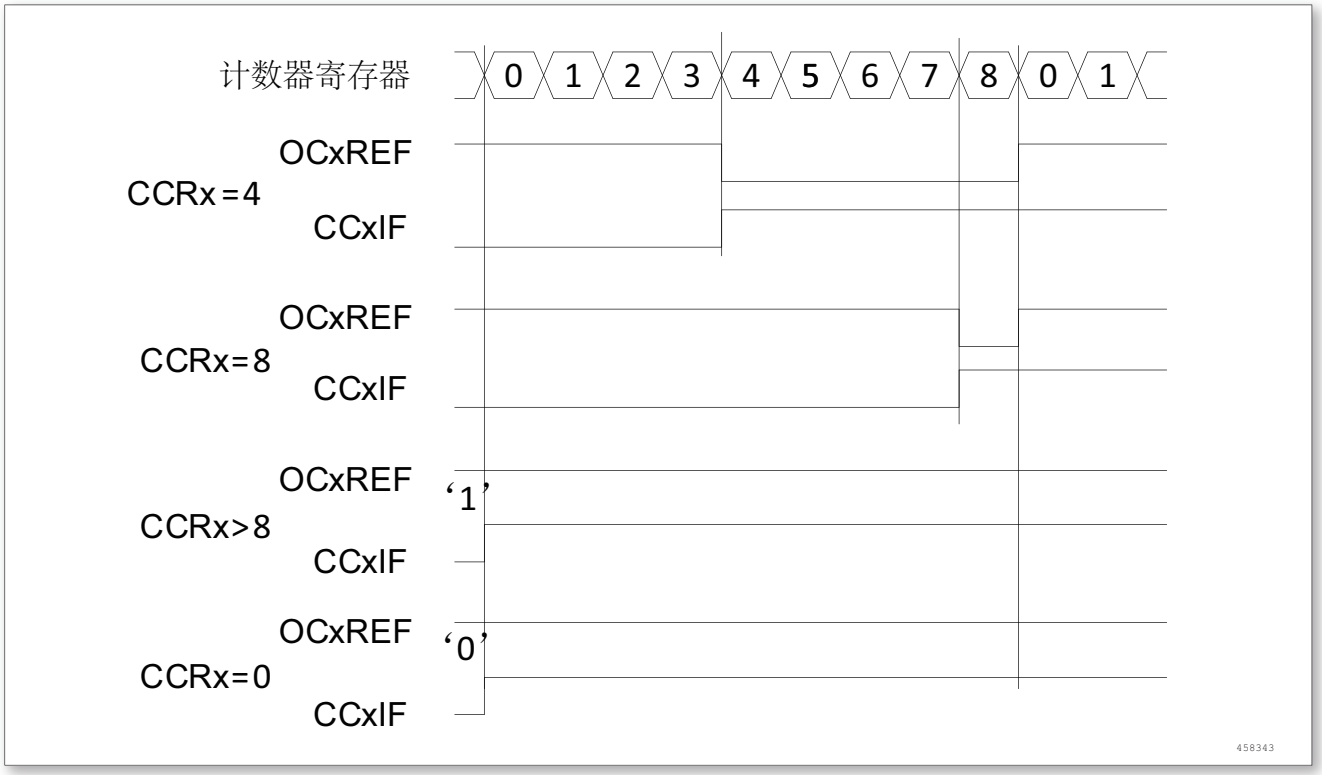


图 106. 边沿对齐的 PWM 波形 (ARR = 8)

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

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

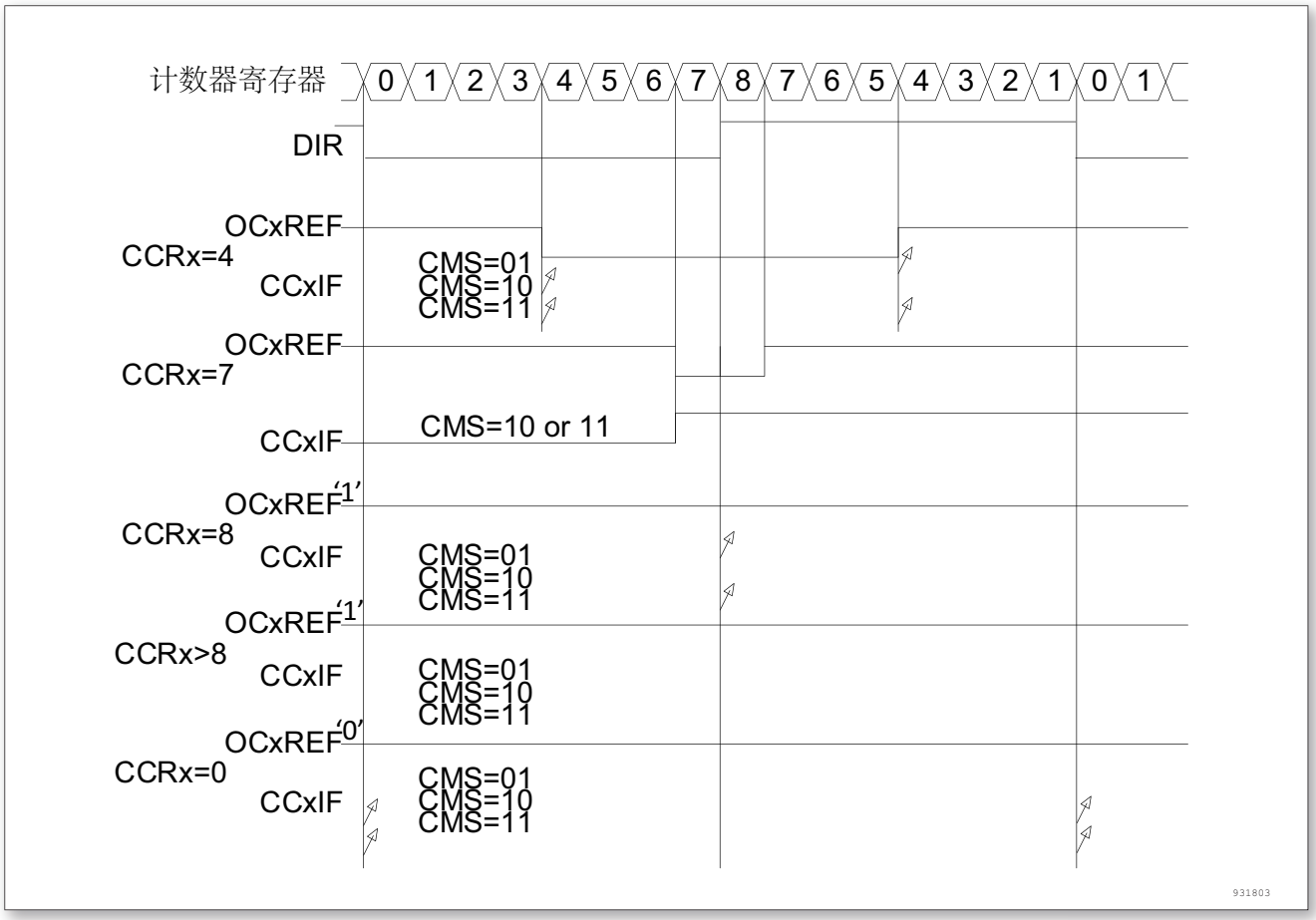


图 107. 中央对齐的 PWM 波形 (APR = 8)

使用中央对齐模式的提示：

- 进入中央对齐模式时，使用当前的上/下计数配置；这就意味着计数器向上还是向下计数取决于 TIMx_CR1 寄存器中 DIR 位的当前值。此外，软件不能同时修改 DIR 和 CMS 位。
- 不推荐当运行在中央对齐模式时改写计数器，因为会产生不可预知的结果。特别地：
  - 如果写入计数器的值大于自动重加载的值 (TIMx_CNT > TIMx_ARR)，则方向不会被更新。例如，如果计数器正在向上计数，它就会继续向上计数。
  - 如果将 0 或者 TIMx_ARR 的值写入计数器，方向被更新，但不产生更新事件 UEV
- 使用中央对齐模式最保险的方法，就是在启动计数器之前产生一个软件更新 (设置 TIMx_EGR 位中的 UG 位)，不要在计数进行过程中修改计数器的值。

12.3.10 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

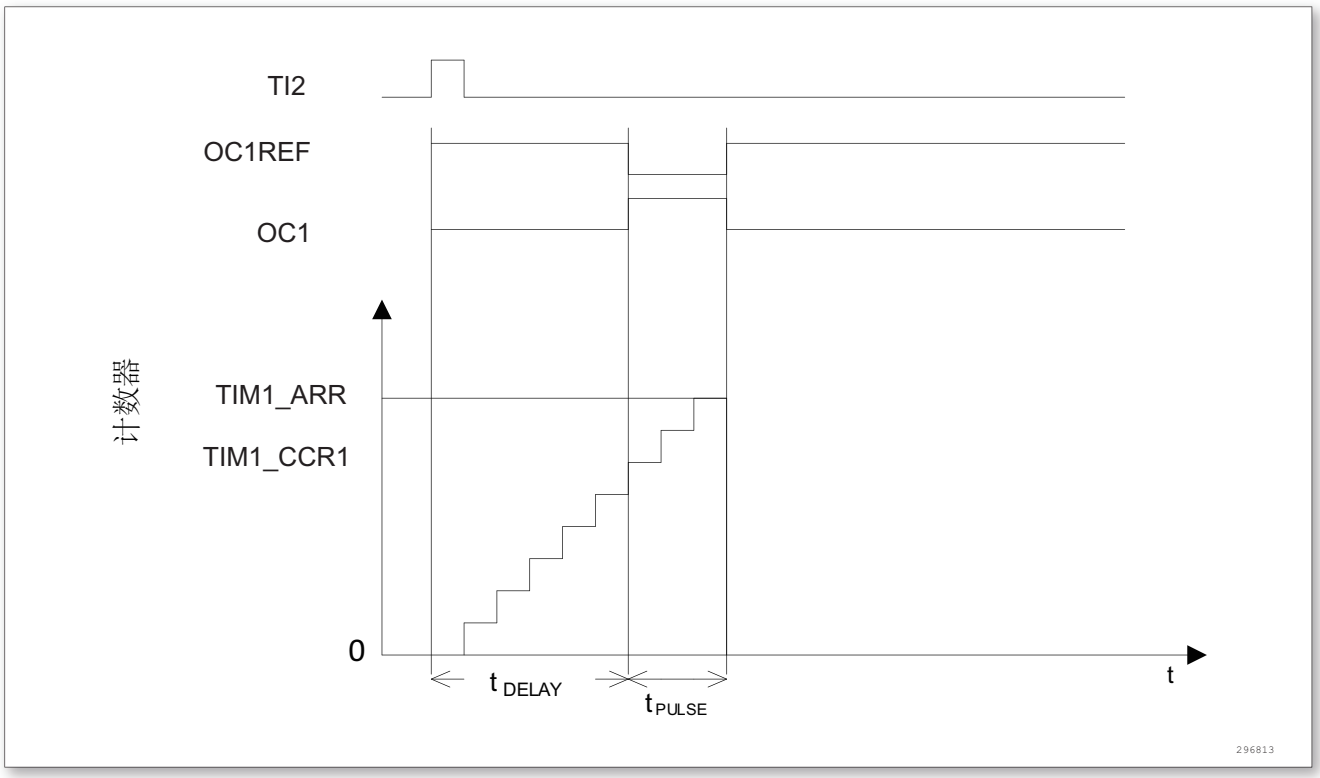


图 108. 单脉冲模式的例子

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

假定 TI2FP2 作为触发 1：

- 置 TIMx_CCMR1 寄存器中的 CC2S = 01，把 TI2FP2 映像到 TI2
- 置 TIMx_CCER 寄存器中的 CC2P = 0，使 TI2FP2 能够检测上升沿
- 置 TIMx_SMCR 寄存器中的 TS = 110，TI2FP2 作为从模式控制器的触发 (TRGI)
- 置 TIMx_SMCR 寄存器中的 SMS = 110(触发模式)，TI2FP2 被用来启动计数器

OPM 波形由写入比较寄存器的数值决定 (要考虑时钟频率和计数器预分频器)。

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

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

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

特殊情况：OCx 快速使能：

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

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

12.3.11 在外部事件时清除 OCxREF 信号

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

该功能只能用于输出比较和 PWM 模式，而不能用于强置模式。

例如，OCxREF 信号可以连到一个外部输入。这时，ETR 必须配置如下：

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

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

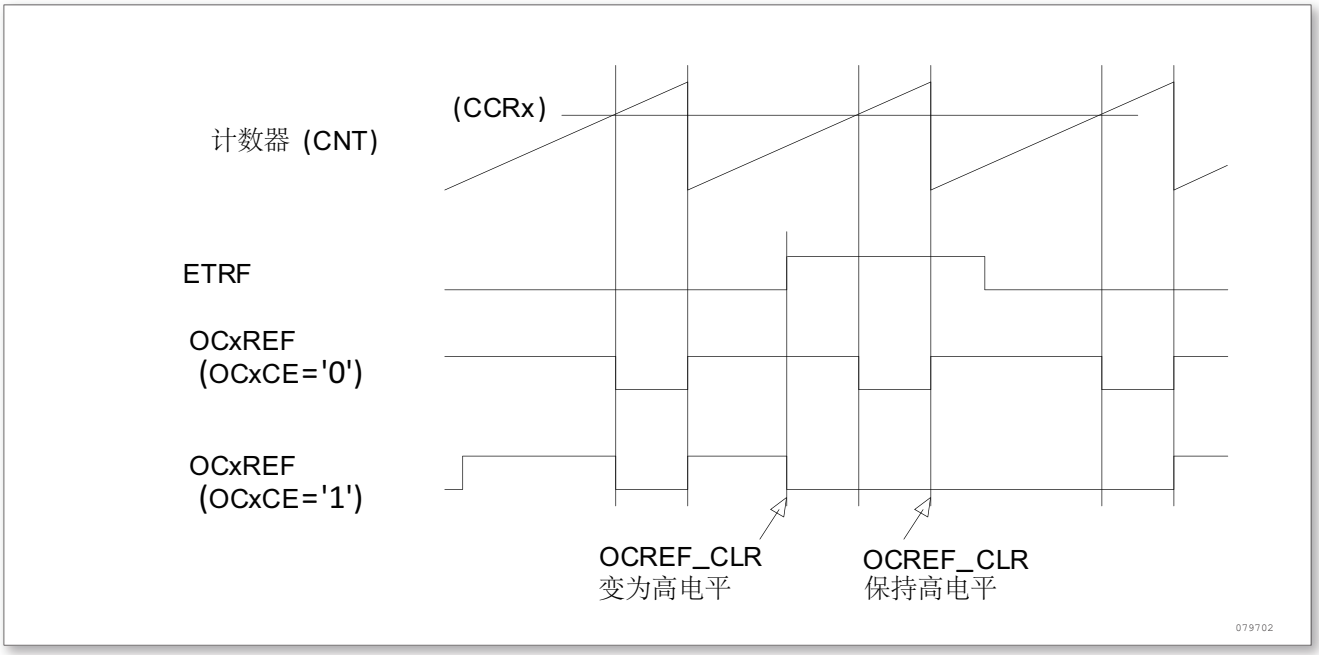


图 109. 清除 TIMx 的 OCxREF

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

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

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

表 51. 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		TI1FP2 信号	
		上升	下降	上升	下降
仅在 TI1 计数	高	向下计数	向上计数	不计数	不计数
	低	向上计数	向下计数	不计数	不计数
仅在 TI2 计数	高	不计数	不计数	向上计数	向下计数
	低	不计数	不计数	向下计数	向上计数
在 TI1 和 TI2 上计数	高	向下计数	向上计数	向上计数	向下计数
	低	向上计数	向下计数	向下计数	向上计数

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

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

- C1S = ‘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)

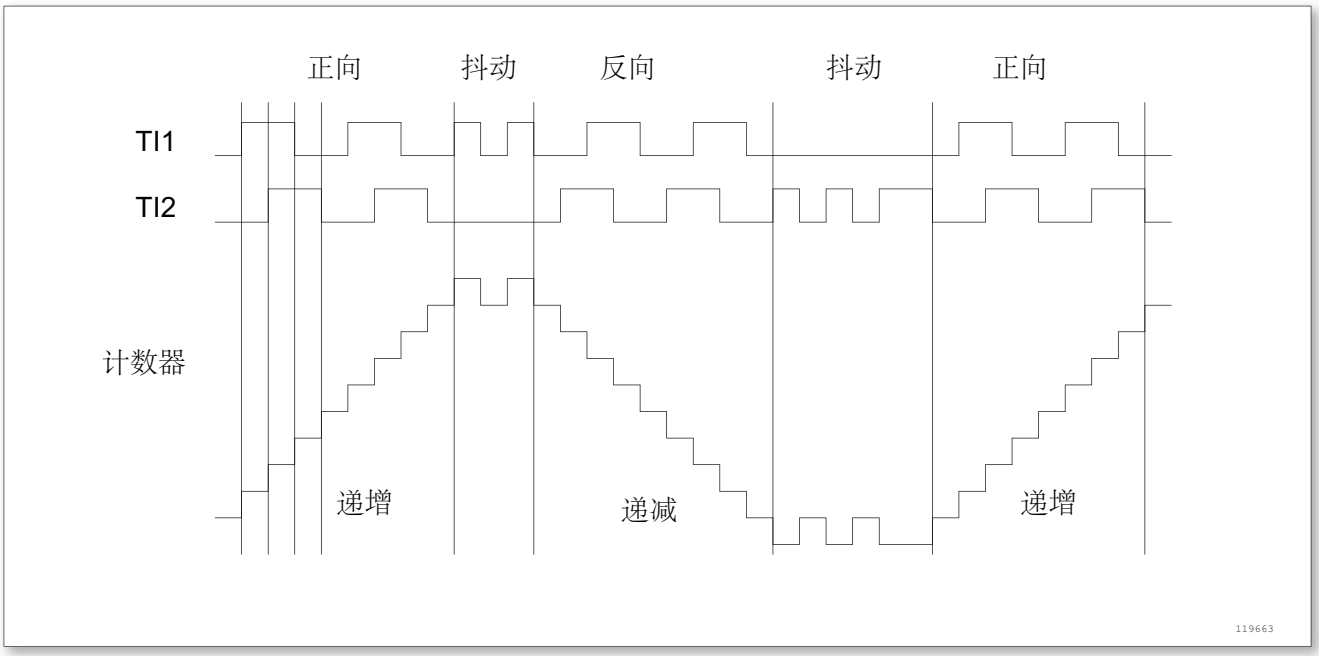


图 110. 编码器模式下的计数器操作实例

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)

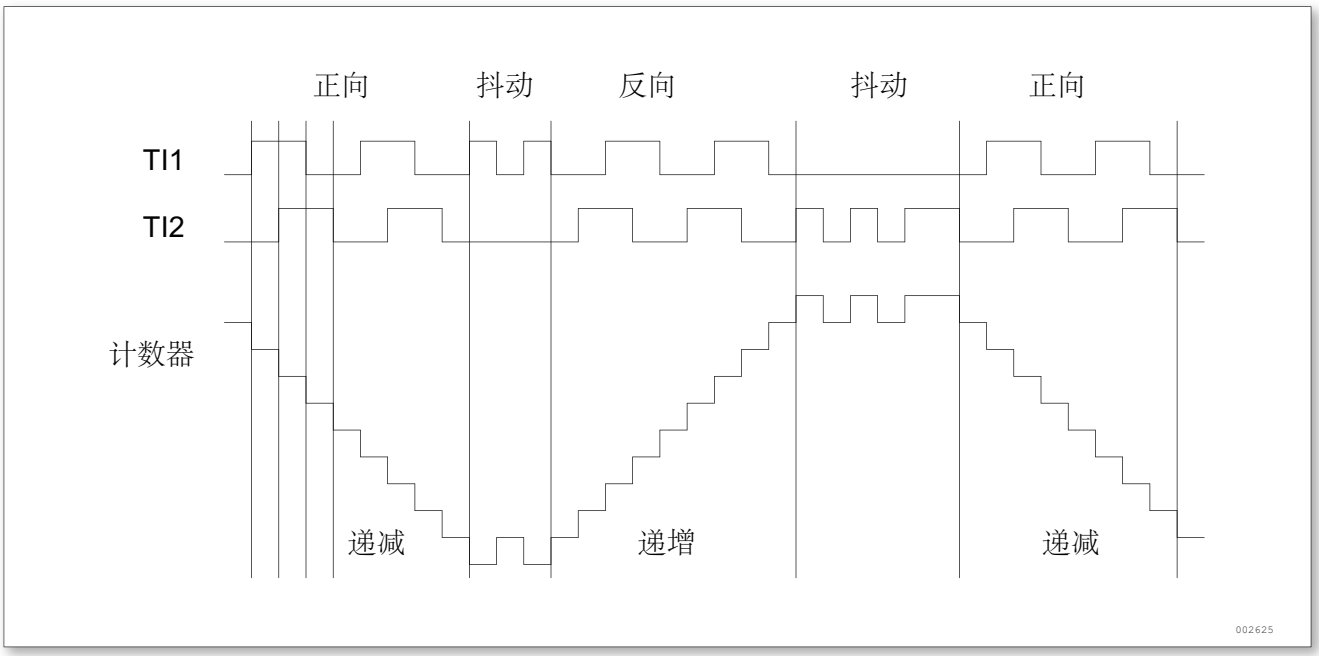


图 111. IC1FP1 反相的编码器接口模式实例

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

12.3.13 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。高级控制定时器章节给出了此特性用于连接霍尔传感器的例子。

12.3.14 定时器和外部触发的同步

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

从模式：复位模式

在发生一个触发输入事件时，计数器和它的预分频器能够重新被初始化；同时，如果 TIMx_CR1 寄存器的 URS 位为低，还产生一个更新事件 UEV；然后所有的预装载寄存器 (TIMx_ARR，TIMx_CCRx) 都被更新了。

- 在以下的例子中，TI1 输入端的上升沿导致向上计数器被清零
- 配置通道 1 以检测 TI1 的上升沿。配置输入滤波器的带宽 (在本例中，不需要任何滤波器，因此保持 IC1F = 0000)。触发操作中不使用捕获预分频器，所以不需要配置。CC1S 位只选择输入捕获源，即 TIMx_CCMR1 寄存器中 CC1S = 01。置 TIMx_CCER 寄存器中 CC1P = 0 以确定极性 (只检测上升沿)
- 置 TIMx_SMCR 寄存器中 SMS = 100，配置定时器为复位模式；置 TIMx_SMCR 寄存器中 TS = 101，选择 TI1 作为输入源
- 置 TIMx_CR1 寄存器中 CEN = 1，启动计数器

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

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

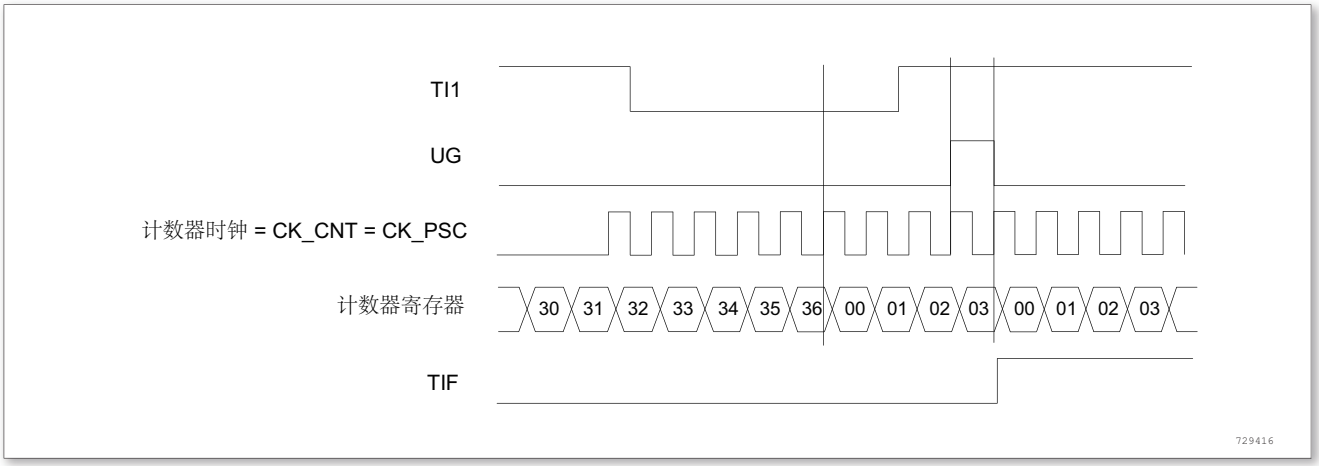


图 112. 复位模式下的控制电路

从模式：门控模式

计数器的使能依赖于选中的输入端的电平。

在如下的例子中，计数器只在 TI1 为低时向上计数：

- 配置通道 1 以检测 TI1 上的低电平。配置输入滤波器带宽 (本例中，不需要滤波，所以保持 IC1F = 0000)。触发操作中不使用捕获预分频器，所以不需要配置。CC1S 位用于选择输入捕获源，置 TIMx_CCMR1 寄存器中 CC1S = 01。置 TIMx_CCER 寄存器中 CC1P = 1 以确定极性 (只检测低电平)。
- 置 TIMx_SMCR 寄存器中 SMS = 101，配置定时器为门控模式；置 TIMx_SMCR 寄存器中 TS = 101，选择 TI1 作为输入源。
- 置 TIMx_CR1 寄存器中 CEN = 1，启动计数器。在门控模式下，如果 CEN = 0，则计数器不能启动，不论触发输入电平如何。

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

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

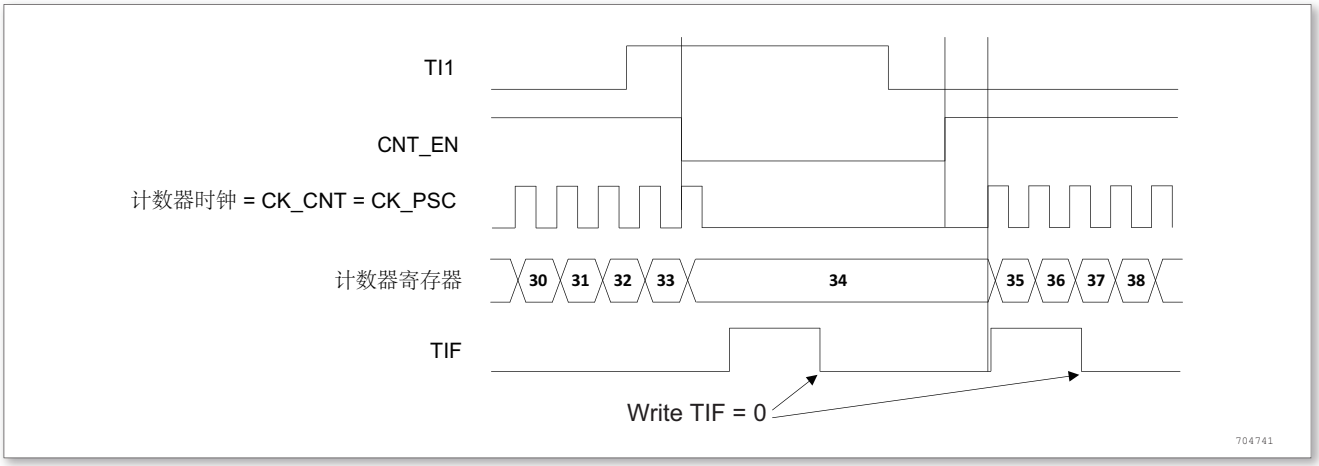


图 113. 门控模式下的控制电路

从模式：触发模式

计数器的使能依赖于选中的输入端上的事件。

在下面的例子中，计数器在 TI2 输入的上升沿开始向上计数：

- 配置通道 2 检测 TI2 的上升沿。配置输入滤波器带宽 (本例中，不需要任何滤波器，保持 IC2F = 0000)。触发操作中不使用捕获预分频器，不需要配置。CC2S 位只用于选择输入捕获源，置 TIMx_CCMR1 寄存器中 CC2S = 01。置 TIMx_CCER 寄存器中 CC1P = 1 以确定极性 (只检测低电平)。
- 置 TIMx_SMCR 寄存器中 SMS = 110，配置定时器为触发模式；置 TIMx_SMCR 寄存器中 TS = 110，选择 TI2 作为输入源。

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

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

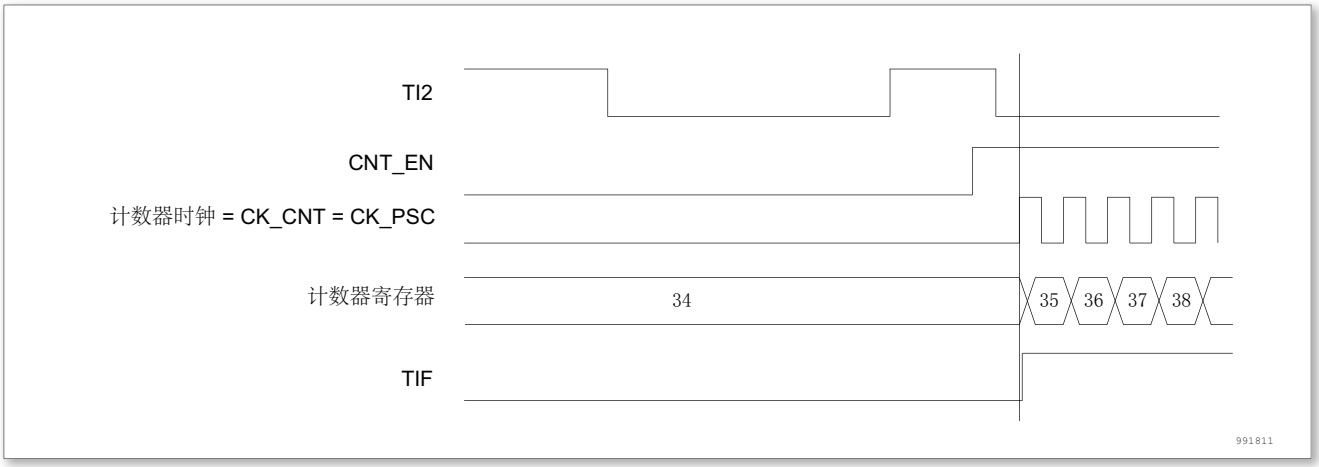


图 114. 触发器模式下的控制电路

从模式：外部时钟模式 2 + 触发模式

外部时钟模式 2 可以与另一种从模式 (外部时钟模式 1 和编码器模式除外) 一起使用。这时，ETR 信号被用作外部时钟的输入，在复位模式、门控模式或触发模式时可以选择另一个输入作为触发输入。不建议使用 TIMx_SMCR 寄存器的 TS 位选择 ETR 作为 TRGI。

在下面的例子中，当 TI1 上出现一个上升沿时，计数器便在 ETR 的每一个上升沿向上计数一次：

- 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)
- 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。

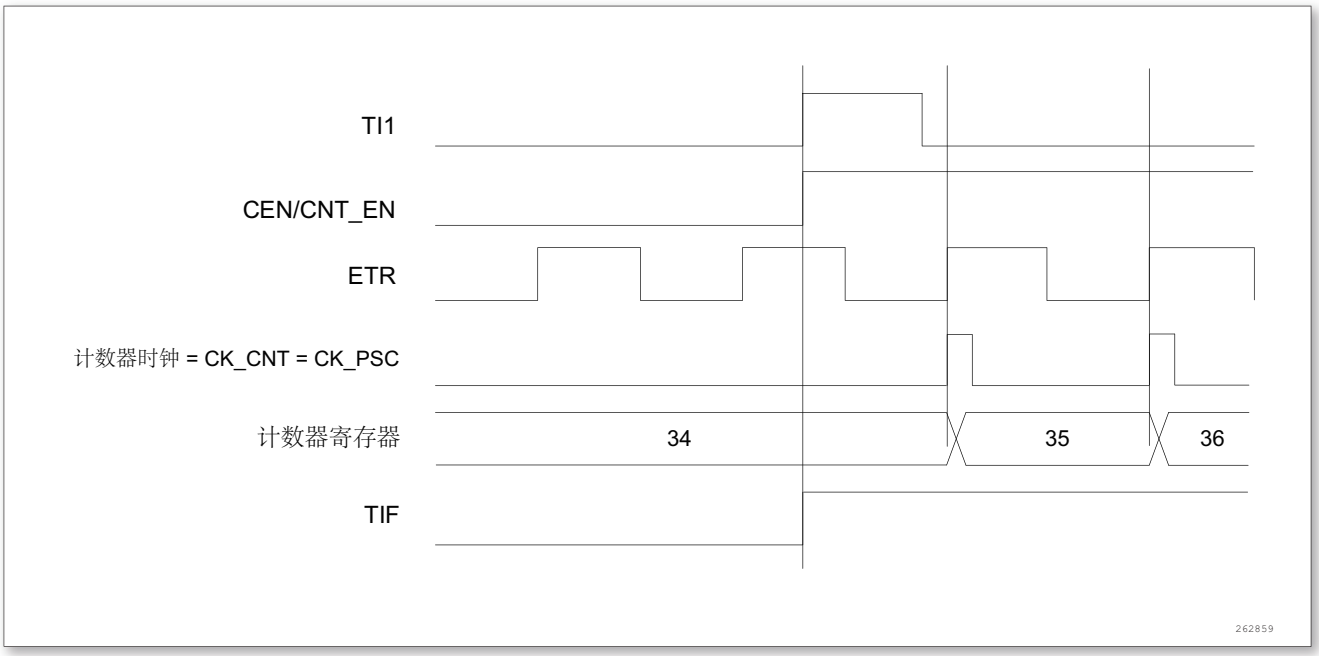


图 115. 外部时钟模式 2 + 触发模式下的控制电路

12.3.15 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

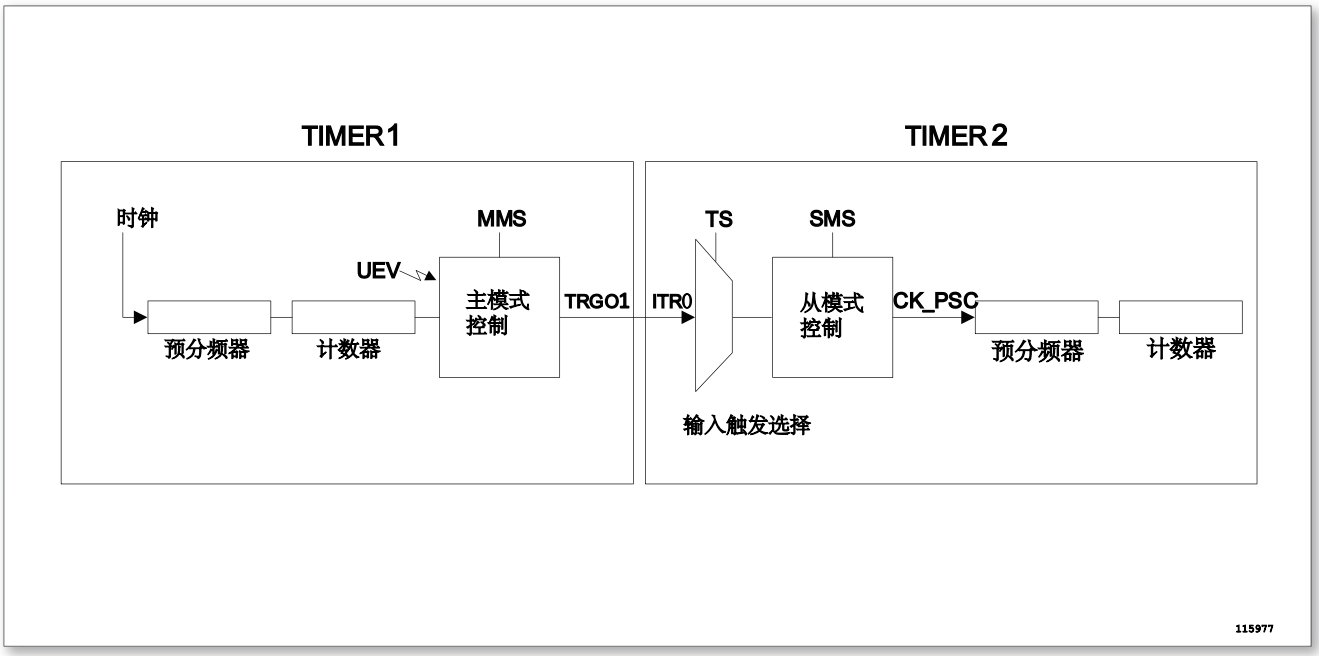


图 116. 主/从定时器的例子

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

- 配置定时器 1 为主模式，送出它的输出比较参考信号 (OC1REF) 为触发输出 (TIM1_CR2 寄存器的 MMS = 100)
- 配置定时器 1 的 OC1REF 波形 (TIM1_CCMR1 寄存器)
- 配置定时器 2 从定时器 1 获得输入触发 (TIM2_SMCR 寄存器的 TS = 001)
- 配置定时器 2 为门控模式 (TIM2_SMCR 寄存器的 SMS = 101)
- 置 TIM2_CR1 寄存器的 CEN = 1 以使能定时器 2
- 置 TIM1_CR1 寄存器的 CEN = 1 以使能定时器 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。

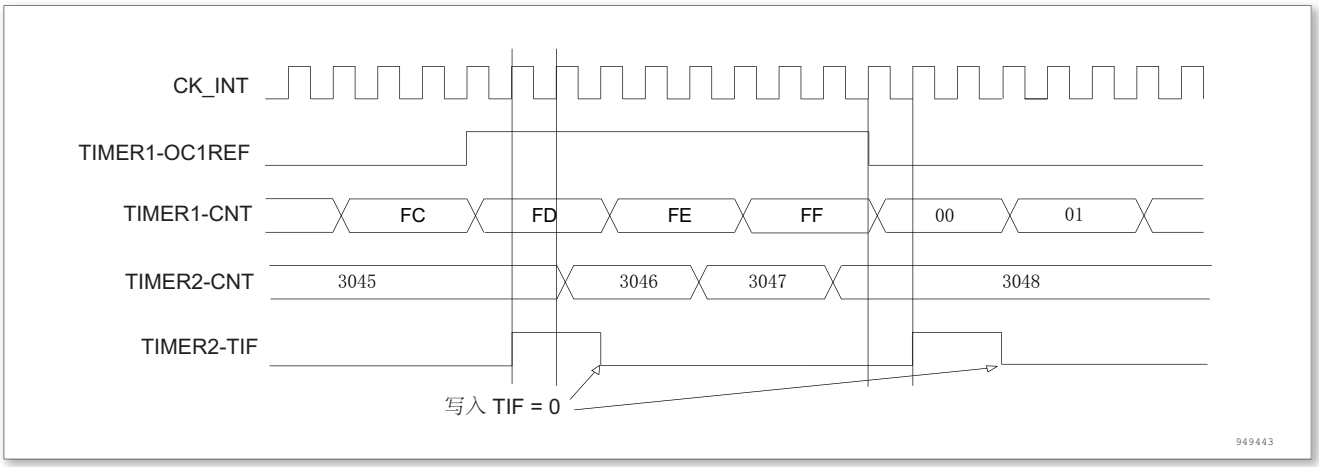


图 117. 定时器 1 的 OC1REF 控制定时器 2

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_CNT)，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。

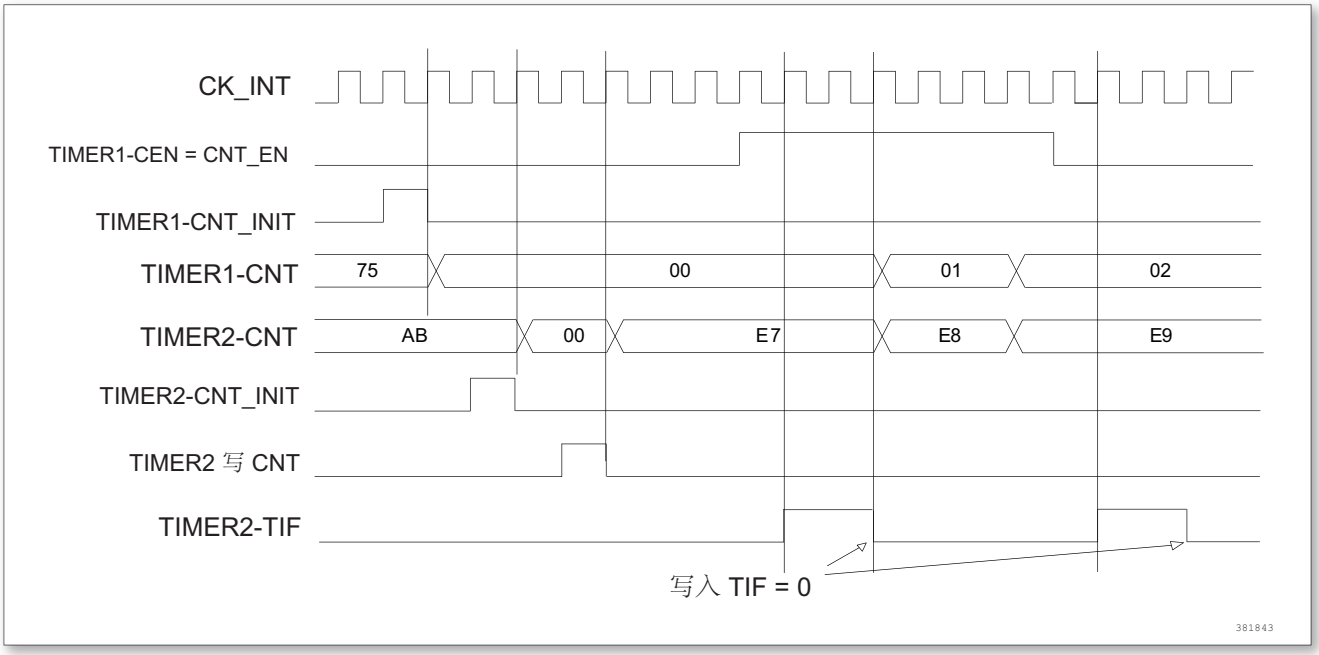


图 118. 通过使能定时器 1 可以控制定时器 2

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。参考下图。当定时器 1 产生更新事件时，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。

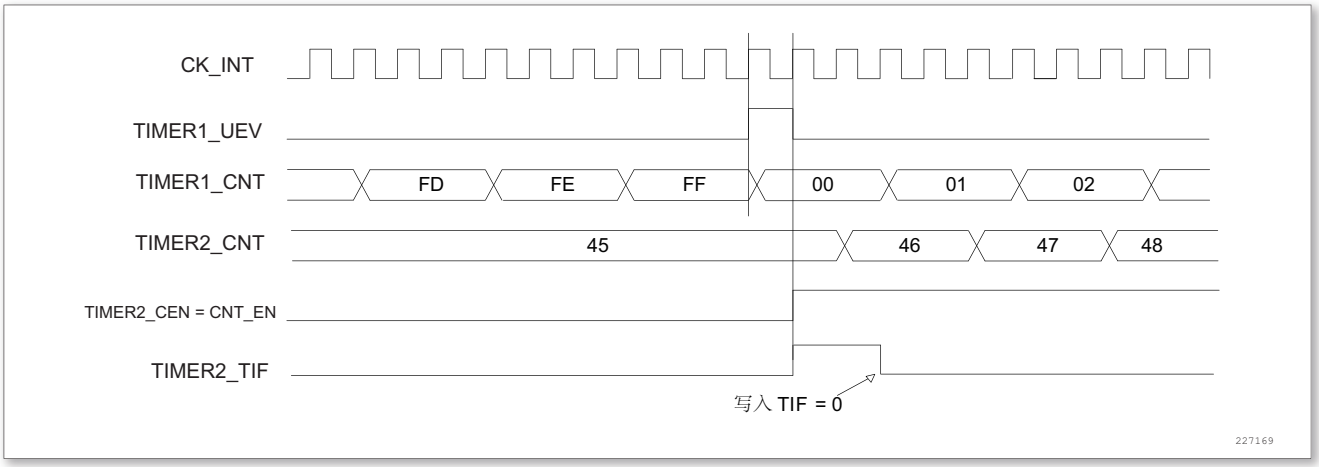


图 119. 使用定时器 1 的更新触发定时器 2

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。

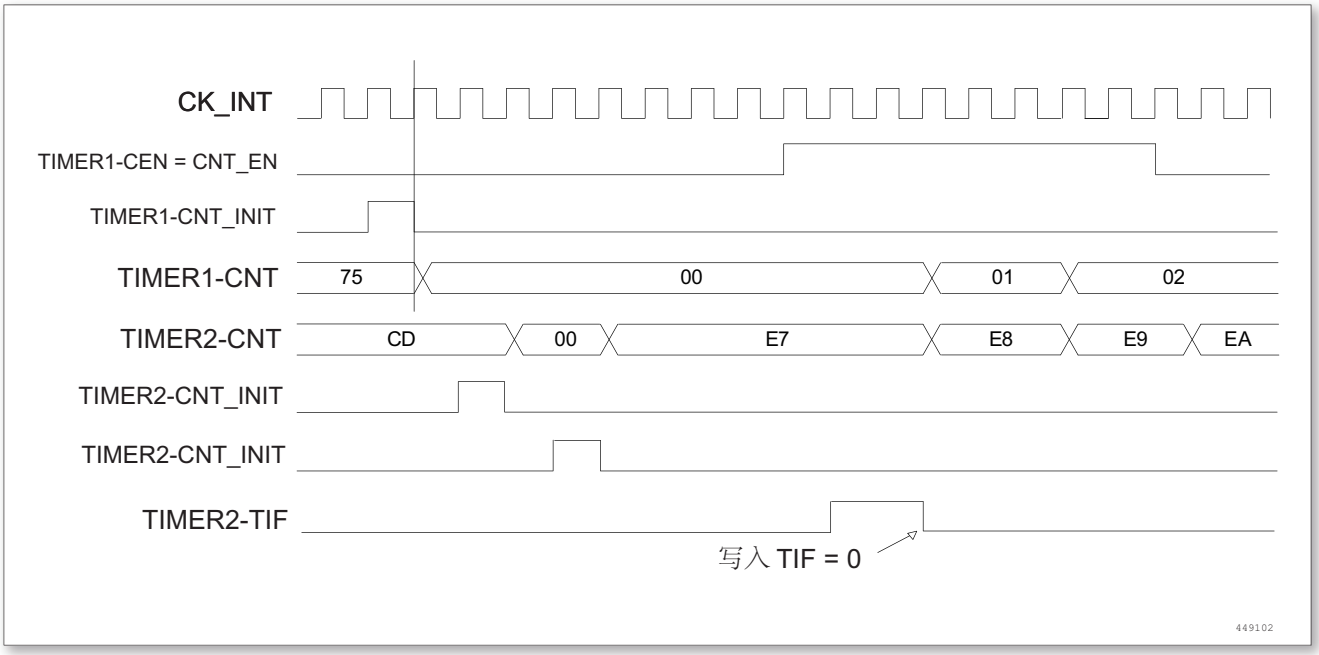


图 120. 利用定时器 1 的使能触发定时器 2

使用一个额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。配置如下：

- 配置定时器 1 为主模式，送出它的更新事件 UEV 作为触发输出 (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 的使能定时器 2。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，送出它的使能作为触发输出 (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 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。

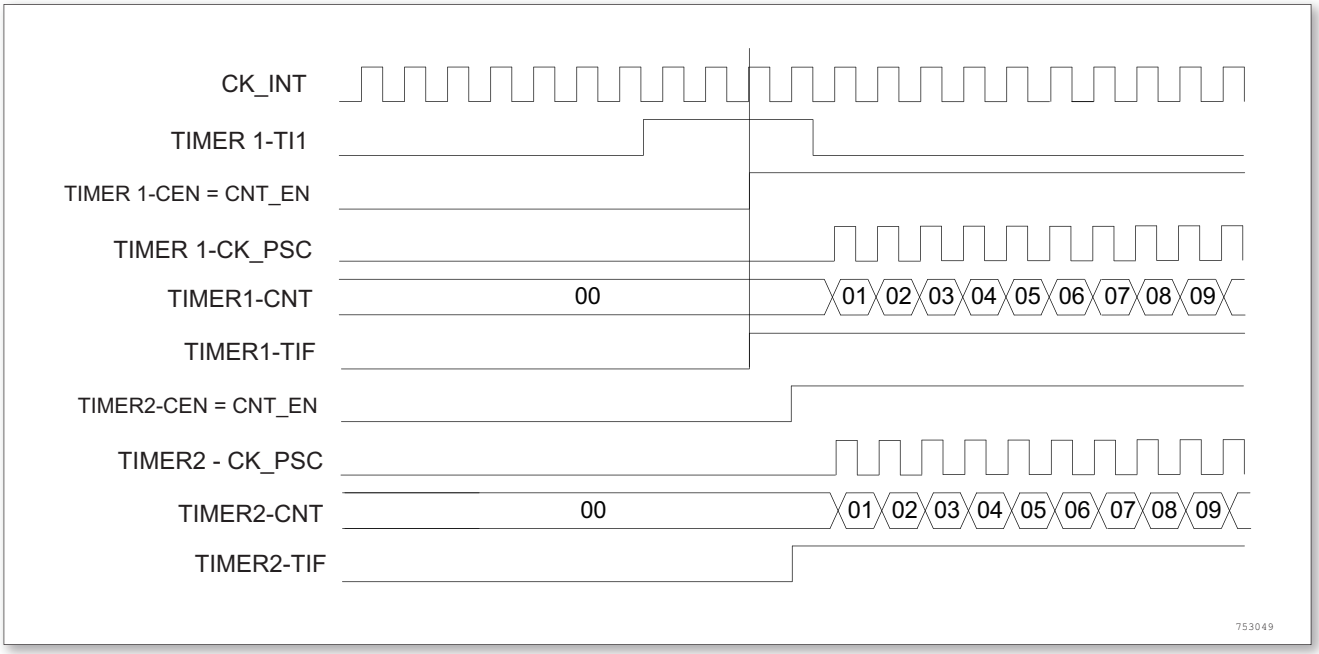


图 121. 使用定时器 1 的 TI1 输入触发定时器 1 和定时器 2

12.3.16 调试模式

当微控制器进入调试模式 (CPU 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。详见调试模块章节。

12.4 TIMx 寄存器描述

可以用半字 (16 位) 或字 (32 位) 的方式操作这些外设寄存器。

表 52. TIMx 寄存器概览

Offset	Acronym	Register Name	Reset	Section
0x00	TIMx_CR1	控制寄存器 1	0x00000000	小节 12.4.1
0x04	TIMx_CR2	控制寄存器 2	0x00000000	小节 12.4.2
0x08	TIMx_SMCR	从模式控制寄存器	0x00000000	小节 12.4.3
0x0C	TIMx_DIER	DMA/中断使能寄存器	0x00000000	小节 12.4.4
0x10	TIMx_SR	状态寄存器	0x00000000	小节 12.4.5
0x14	TIMx_EGR	事件产生寄存器	0x00000000	小节 12.4.6
0x18	TIMx_CCMR1	捕获/比较模式寄存器 1	0x00000000	小节 12.4.7
0x1C	TIMx_CCMR2	捕获/比较模式寄存器 2	0x00000000	小节 12.4.8
0x20	TIMx_CCER	捕获/比较使能寄存器	0x00000000	小节 12.4.9
0x24	TIMx_CNT	计数器	0x00000000	小节 12.4.10
0x28	TIMx_PSC	预分频器	0x00000000	小节 12.4.11
0x2C	TIMx_ARR	自动装载寄存器	0x00000000	小节 12.4.12
0x34	TIMx_CCR1	捕获/比较寄存器 1	0x00000000	小节 12.4.13
0x38	TIMx_CCR2	捕获/比较寄存器 2	0x00000000	小节 12.4.14
0x3C	TIMx_CCR3	捕获/比较寄存器 3	0x00000000	小节 12.4.15
0x40	TIMx_CCR4	捕获/比较寄存器 4	0x00000000	小节 12.4.16
0x48	TIMx_DCR	DMA 控制寄存器	0x00000000	小节 12.4.17
0x4C	TIMx_DMAR	连续模式的 DMA 地址	0x00000000	小节 12.4.18

12.4.1 控制寄存器 1(TIMx_CR1)

偏移地址：0x00

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						CKD		ARPE	CMS		DIR	OPM	URS	UDIS	CEN
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
15：10	Reserved			保留, 读为 0
9：8	CKD	rw	0x00	时钟分频因子 (Clock division) 这 2 位定义在定时器时钟 (CK_INT) 频率、死区时间和由死区发生器与数字滤波器 (ETR，TIx) 所用的采样时钟之间的分频比例。 00：$t_{DTS} = t_{CK\_INT}$ 01：$t_{DTS} = 2 \times t_{CK\_INT}$ 10：$t_{DTS} = 4 \times t_{CK\_INT}$ 11：保留，不要使用这个配置

Bit	Field	Type	Reset	Description
7	ARPE	rw	0x00	自动重装载预装载允许位 (Auto-reload preload enable) 0：TIMx_ARR 寄存器没有缓冲 1：TIMx_ARR 寄存器被装入缓冲器
6：5	CMS	rw	0x00	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。计数器交替地向上和向下计数。配置为输出的通道 (TIMx_CCMRx 寄存器中 CCxS = 00) 的输出比较中断标志位，只在计数器向上计数时被设置 11：中央对齐模式 3。计数器交替地向上和向下计数。计数器交替地向上和向下计数。配置为输出的通道 (TIMx_CCMRx 寄存器中 CCxS = 00) 的输出比较中断标志位，在计数器向上和向下计数时均被设置 注：在计数器开启时 (CEN = 1)，不允许从边沿对齐模式转换到中央对齐模式。
4	DIR	rw	0x00	方向 (Direction) 0：计数器向上计数 1：计数器向下计数 注：当计数器配置为中央对齐模式或编码器模式时，该位为只读。
3	OPM	rw	0x00	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
2	URS	rw	0x00	更新请求源 (Update request source) 软件通过该位选择 UEV 事件的源。 0：如果允许产生更新中断或 DMA 请求，则下述任一事件产生一个更新中断或 DMA 请求： - 计数器溢出/下溢 - 设置 UG 位 - 从模式控制器产生的更新 1：如果允许产生更新中断或 DMA 请求，则只有计数器溢出/下溢才产生一个更新中断或 DMA 请求

Bit	Field	Type	Reset	Description
1	UDIS	rw	0x00	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被缓存的寄存器被装入它们的预装载值。 1：禁止 UEV。不产生更新事件，影子寄存器 (ARR、PSC、CCRx) 保持它们的值。如果设置了 UG 位或从模式控制器发出了一个硬件复位，则计数器和预分频器被重新初始化
0	CEN	rw	0x00	允许计数器 (Counter enable) 0：禁止计数器 1：使能计数器。 注：在软件设置了 CEN 位后，外部时钟、门控模式和编码器模式才能工作。触发模式可以自动地通过硬件设置 CEN 位。

12.4.2 控制寄存器 2(TIMx_CR2)

偏移地址：0x04

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								TI1S	MMS			CCDS	Reserved		
								rw	rw	rw	rw	rw			

Bit	Field	Type	Reset	Description
15: 8	Reserved			保留, 读为 0
7	TI1S	rw	0x00	TI1 选择 (TI1 selection) 0：TIMx_CH1 管脚连到 TI1 输入 1：TIMx_CH1、TIMx_CH2 和 TIMx_CH3 管脚经异或后连到 TI1 输入

Bit	Field	Type	Reset	Description
6：4	MMS	rw	0x00	主模式选择 (Master mode selection) 这两位用于选择在主模式下送到从定时器的同步信息 (TRGO)。可能的组合如下： 000：复位-TIMx_EGR 寄存器的 UG 位被用于作为触发输出 (TRGO)。如果触发输入 (从模式控制器处于复位模式) 产生复位，则 TRGO 上的信号相对实际的复位会有一个延迟。 001：使能-计数器使能信号 CNT_EN 被用于作为触发输出 (TRGO)。有时需要在同一时间启动多个定时器或控制在一段时间内使能从定时器。计数器使能信号是通过 CEN 控制位和门控模式下的触发输入信号的逻辑或产生。当计数器使能信号受控于触发输入时，TRGO 上会有一个延迟，除非选择了主/从模式 (见 TIMx_SMCR 寄存器中 MSM 位的描述)。 010：更新-更新事件被选为触发输入 (TRGO)。例如，一个主定时器的时钟可以被用作一个从定时器的预分频器。 011：比较脉冲-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)
3	CCDS	rw	0x00	捕获/比较的 DMA 选择 (Capture/compare DMA selection) 0：当发生 CCx 事件时，送出 CCx 的 DMA 请求 1：当发生更新事件时，送出 CCx 的 DMA 请求
2: 0	Reserved			保留, 读为 0

12.4.3 从模式控制寄存器 (TIMx_SMCR)

偏移地址：0x08

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ETP	ECE	ETPS		ETF				MSM	TS			Res.	SMS		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw

Bit	Field	Type	Reset	Description
15	ETP	rw	0x00	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

Bit	Field	Type	Reset	Description
14	ECE	rw	0x00	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，计器由 ETRF 信号上的任意有效上升沿驱动 注 1：设置 ECE 位与选择外部时钟模式 1 并将 TRGI 连到 ETRF(SMS = 111 和 TS = 111) 具有相同功效。 注 2：下述从模式可以与外部时钟模式 2 同时使用：复位模式，门控模式和触发模式；但是，这时 TRGI 不能连到 ETRF(TS 位不能是 111)。 注 3：外部时钟模式 1 和外部时钟模式 2 同时被使能时，外部时钟的输入是 ETRF。
13：12	ETPS	rw	0x00	外部触发预分频 (External trigger prescaler) 外部触发信号 ETRP 的频率必须最多是 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
11：8	ETF	rw	0x00	外部触发滤波 (External trigger filter) 这些位定义了对 ETRP 信号采样的频率和对 ETRP 数字滤波的带宽。实际上，数字滤波器是一个事件计数器，它记录到 N 个事件后会产生一个输出的跳变。 0000：无滤波器，以 f_{DTS} 采样 0001：采样频率 $f_{SAMPLING}=f_{CK\_INT}$，N = 2 0010：采样频率 $f_{SAMPLING}=f_{CK\_INT}$，N = 4 0011：采样频率 $f_{SAMPLING}=f_{CK\_INT}$，N = 8 0100：采样频率 $f_{SAMPLING}=f_{DTS}/2$，N = 6 0101：采样频率 $f_{SAMPLING}=f_{DTS}/2$，N = 8 0110：采样频率 $f_{SAMPLING}=f_{DTS}/4$，N = 6 0111：采样频率 $f_{SAMPLING}=f_{DTS}/4$，N = 8 1000：采样频率 $f_{SAMPLING}=f_{DTS}/8$，N = 6 1001：采样频率 $f_{SAMPLING}=f_{DTS}/8$，N = 8 1010：采样频率 $f_{SAMPLING}=f_{DTS}/16$，N = 5 1011：采样频率 $f_{SAMPLING}=f_{DTS}/16$，N = 6 1100：采样频率 $f_{SAMPLING}=f_{DTS}/16$，N = 8 1101：采样频率 $f_{SAMPLING}=f_{DTS}/32$，N = 5 1110：采样频率 $f_{SAMPLING}=f_{DTS}/32$，N = 6 1111：采样频率 $f_{SAMPLING}=f_{DTS}/32$，N = 8

Bit	Field	Type	Reset	Description
7	MSM	rw	0x00	主/从模式 (Master/slave mode) 0：无作用 1：触发输入 (TRGI) 上的事件被延迟了，以允许在当前定时器 (通过 TRGO) 与它的从定时器间的完美同步，这对要求把几个定时器同步到一个单一的外部事件时是非常有用的
6：4	TS	rw	0x00	触发选择 (Trigger selection) 这 3 位选择用于同步计数器的触发输入。 000：内部触发 0(ITR0) 001：内部触发 1(ITR1) 010：内部触发 2(ITR2) 011：内部触发 3(ITR3) 100：TI1 的边沿检测器 (TI1F_ED) 101：滤波后的定时器输入 1(TI1FP1) 110：滤波后的定时器输入 2(TI2FP2) 111：外部触发输入 (ETRF) 更多有关 ITRx 的细节，参见下表。 注：这些位只能在未用到 (如 SMS = 000) 时被改变，以避免在改变时产生错误的边沿检测。
3	Reserved			保留, 读为 0

Bit	Field	Type	Reset	Description
2：0	SMS	rw	0x00	从模式选择 (Slave mode selection) 当选择了外部信号，触发信号 (TRGI) 的有效边沿与选中的外部输入极性相关 (见输入控制寄存器和控制寄存器的说明) 000：关闭从模式 - 如果 CEN = 1，则预分频器直接由内部时钟驱动。 001：编码器模式 1 - 根据 TI1FP1 的电平，计数器在 TI2FP2 的边沿向上/下计数。 010：编码器模式 2 - 根据 TI2FP2 的电平，计数器在 TI1FP1 的边沿向上/下计数。 011：编码器模式 3 - 根据另一个输入的电平，计数器在 TI1FP1 和 TI2FP2 的边沿向上/下计数。 100：复位模式 - 选中的触发输入 (TRGI) 的上升沿重新初始化计数器，并且产生一个更新寄存器的信号。 101：门控模式 - 当触发输入 (TRGI) 为高时，计数器的时钟开启。当触发输入变为低时，计数器停止 (但不复位)。计数器的启动和停止都是受控的。 110：触发模式 - 计数器在触发输入 TRGI 的上升沿启动 (但不复位)，只有计数器的启动是受控的。 111：外部时钟模式 1 - 选中的触发输入 (TRGI) 的上升沿驱动计数器。 注：如果 TI1F_EN 被选为触发输入 (TS = 100) 时，不要使用门控模式。这是因为，TI1F_ED 在每次 TI1F 变化时输出一个脉冲，然而门控模式是要检查触发输入的电平。

表 53. TIMx 内部触发连接

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

12.4.4 DMA/中断使能寄存器 (TIMx_DIER)

偏移地址：0x0C

复位值：0x0000

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

Bit	Field	Type	Reset	Description
15	Reserved			保留, 读为 0

Bit	Field	Type	Reset	Description
14	TDE	rw	0x00	允许触发 DMA 请求 (Trigger DMA request enable) 0：禁止触发 DMA 请求 1：允许触发 DMA 请求
13	Reserved			保留, 读为 0
12	CC4DE	rw	0x00	允许捕获/比较 4 的 DMA 请求 (Capture/Compare 4 DMA request enable) 0：禁止捕获/比较 4 的 DMA 请求 1：允许捕获/比较 4 的 DMA 请求
11	CC3DE	rw	0x00	允许捕获/比较 3 的 DMA 请求 (Capture/Compare 3 DMA request enable) 0：禁止捕获/比较 3 的 DMA 请求 1：允许捕获/比较 3 的 DMA 请求
10	CC2DE	rw	0x00	允许捕获/比较 2 的 DMA 请求 (Capture/Compare 2 DMA request enable) 0：禁止捕获/比较 2 的 DMA 请求 1：允许捕获/比较 2 的 DMA 请求
9	CC1DE	rw	0x00	允许捕获/比较 1 的 DMA 请求 (Capture/Compare 1 DMA request enable) 0：禁止捕获/比较 1 的 DMA 请求 1：允许捕获/比较 1 的 DMA 请求
8	UDE	rw	0x00	允许更新的 DMA 请求 (Update DMA request enable) 0：禁止更新的 DMA 请求 1：允许更新的 DMA 请求
7	Reserved			保留, 读为 0
6	TIE	rw	0x00	触发中断使能 (Trigger interrupt enable) 0：禁止触发中断 1：使能触发中断
5	Reserved			保留, 读为 0
4	CC4IE	rw	0x00	允许捕获/比较 4 中断 (Capture/Compare 4 interrupt enable) 0：禁止捕获/比较 4 中断 1：允许捕获/比较 4 中断
3	CC3IE	rw	0x00	允许捕获/比较 3 中断 (Capture/Compare 3 interrupt enable) 0：禁止捕获/比较 3 中断 1：允许捕获/比较 3 中断
2	CC2IE	rw	0x00	允许捕获/比较 2 中断 (Capture/Compare 2 interrupt enable) 0：禁止捕获/比较 2 中断 1：允许捕获/比较 2 中断

Bit	Field	Type	Reset	Description
1	CC1IE	rw	0x00	允许捕获/比较 1 中断 (Capture/Compare 1 interrupt enable) 0：禁止捕获/比较 1 中断 1：允许捕获/比较 1 中断
0	UIE	rw	0x00	允许更新中断 (Update interrupt enable) 0：禁止更新中断 1：允许更新中断

12.4.5 状态寄存器 (TIMx_SR)

偏移地址：0x10

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.			CC4OF	CC3OF	CC2OF	CC1OF	Res.		TIF	Res.	CC4IF	CC3IF	CC2IF	CC1IF	UIF
			rc_w0	rc_w0	rc_w0	rc_w0			rc_w0		rc_w0	rc_w0	rc_w0	rc_w0	rc_w0

Bit	Field	Type	Reset	Description
15：13	Reserved			保留, 读为 0
12	CC4OF	rc_w0	0x00	捕获/比较 4 重复捕获标记 (Capture/Compare 4 overcapture flag) 参见 CC1OF 描述。
11	CC3OF	rc_w0	0x00	捕获/比较 3 重复捕获标记 (Capture/Compare 3 overcapture flag) 参见 CC1OF 描述。
10	CC2OF	rc_w0	0x00	捕获/比较 2 重复捕获标记 (Capture/Compare 2 overcapture flag) 参见 CC1OF 描述。
9	CC1OF	rc_w0	0x00	捕获/比较 1 重复捕获标记 (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。写 0 可清除该位。 0：无重复捕获产生； 1：计数器的值被捕获到 TIMx_CCR1 寄存器时，CC1IF 的状态已经为 1。
8: 7	Reserved			保留, 读为 0
6	TIF	rc_w0	0x00	触发器中断标记 (Trigger interrupt flag) 当发生触发事件 (当从模式控制器处于除门控模式外的其它模式时，在 TRGI 输入端检测到有效边沿，或或门控模式下的任一边沿) 时由硬件对该位置 1。它由软件清 0。 0：无触发器事件产生 1：触发器中断等待响应
5	Reserved			保留, 读为 0

Bit	Field	Type	Reset	Description
4	CC4IF	rc_w0	0x00	捕获/比较 4 中断标记 (Capture/Compare 4 interrupt flag) 参考 CC1IF 描述。
3	CC3IF	rc_w0	0x00	捕获/比较 3 中断标记 (Capture/Compare 3 interrupt flag) 参考 CC1IF 描述。
2	CC2IF	rc_w0	0x00	捕获/比较 2 中断标记 (Capture/Compare 2 interrupt flag) 参考 CC1IF 描述。
1	CC1IF	rc_w0	0x00	捕获/比较 1 中断标记 (Capture/Compare 1 interrupt flag) 如果通道 CC1 配置为输出模式： 当计数器值与比较值匹配时该位由硬件置‘1’，但在中心对称模式下除外 (参考 TIMx_CR1 寄存器的 CMS 位)。它由软件清‘0’。 0：无匹配发生 1：TIMx_CNT 的值与 TIMx_CCR1 的值匹配 如果通道 CC1 配置为输入模式： 当捕获事件发生时该位由硬件置‘1’，它由软件清 0 或通过读 TIMx_CCR1 清‘0’。 0：无输入捕获产生 1：计数器值已被捕获 (拷贝) 至 TIMx_CCR1(在 IC1 上检测到与所选极性相同的边沿)
0	UIF	rc_w0	0x00	更新中断标记 (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：更新事件等待响应。当寄存器被更新时该位由硬件置‘1’： - 若 TIMx_CR1 寄存器的 UDIS = 0，当 REP_CNT = 0 时产生更新事件 (重复向下计数器上溢或下溢时) - 若 TIMx_CR1 寄存器的 UDIS = 0、URS = 0，当 TIMx_EGR 寄存器的 UG = 1 时产生更新事件 (软件对计数器 CNT 重新初始化) - 若 TIMx_CR1 寄存器的 UDIS = 0、URS = 0，当计数器 CNT 被触发事件重初始化时产生更新事件。(参考同步控制寄存器的说明)

12.4.6 事件产生寄存器 (TIMx_EGR)

偏移地址：0x14

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.									TG	Res.	CC4G	CC3G	CC2G	CC1G	UG
									w		w	w	w	w	w

Bit	Field	Type	Reset	Description
15：7	Reserved			保留, 读为 0

Bit	Field	Type	Reset	Description
6	TG	w	0x00	产生触发事件 (Trigger 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：TIMx_SR 寄存器的 TIF = 1，若开启对应的中断和 DMA，则产生相应的中断和 DMA
5	Reserved			保留, 读为 0
4	CC4G	w	0x00	产生捕获/比较 4 事件 (Capture/Compare 4 generation) 参考 CC1G 描述。
3	CC3G	w	0x00	产生捕获/比较 3 事件 (Capture/Compare 3 generation) 参考 CC1G 描述。
2	CC2G	w	0x00	产生捕获/比较 2 事件 (Capture/Compare 2 generation) 参考 CC1G 描述。
1	CC1G	w	0x00	产生捕获/比较 1 事件 (Capture/Compare 1 generation) 该位由软件置 1，用于产生一个捕获/比较事件，由硬件自动清 0。 0：无动作 1：在通道 CC1 上产生一个捕获/比较事件： 若通道 CC1 配置为输出： 设置 CC1IF=1，若开启对应的中断和 DMA，则产生相应的中断和 DMA。 若通道 CC1 配置为输入： 当前的计数器值被捕获至 TIMx_CCR1 寄存器，设置 CC1IF = 1，若开启对应的中断和 DMA，则产生相应的中断和 DMA。若 CC1IF 已经为 1，则设置 CC1OF = 1。
0	UG	w	0x00	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，并产生一个更新事件。注意预分频器的计数器也被清‘0’(但是预分频系数不变)。若在中心对称模式下或 DIR = 0(向上计数) 则计数器被清‘0’；若 DIR = 1(向下计数) 则计数器取 TIMx_ARR 的值。

12.4.7 捕获/比较模式寄存器 1(TIMx_CCMR1)

偏移地址：0x18

复位值：0x0000

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

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

输出比较模式：

Bit	Field	Type	Reset	Description
15	OC2CE	rw	0x00	输出比较 2 清 0 使能 (Output compare 2 clear enable)
14：12	OC2M	rw	0x00	输出比较 2 模式 (Output compare 2 mode)
11	OC2PE	rw	0x00	输出比较 2 预装载使能 (Output compare 2 preload enable)
10	OC2FE	rw	0x00	输出比较 2 快速使能 (Output compare 4 fast enable)
9：8	CC2S	rw	0x00	捕获/比较 2 选择 (Capture/Compare 2 selection) 该位定义通道的方向 (输入/输出)，及输入脚的选择： 00：CC2 通道被配置为输出 01：CC2 通道被配置为输入，IC2 映射在 TI2 上 10：CC2 通道被配置为输入，IC2 映射在 TI1 上 11：CC2 通道被配置为输入，IC2 映射在 TRC 上。此模式仅工作在内部触发器输入被选中时 (由 TIMx_SMCR 寄存器的 TS 位选择) 注：CC2S 仅在通道关闭时 (TIMx_CCER 寄存器的 CC2E = 0) 才是可写的。
7	OC1CE	rw	0x00	输出比较 1 清 0 使能 (Output compare 1 clear enable) 0：OC1REF 不受 ETRF 输入的影响 1：当检测到 ETRF 输入高电平时，清除 OC1REF = 0

Bit	Field	Type	Reset	Description
6：4	OC1M	rw	0x00	输出比较 1 模式 (Output compare 1 mode) 该 3 位定义了输出参考信号 OC1REF 的动作，而 OC1REF 决定了 OC1、OC1N 的值。OC1REF 是高电平有效，而 OC1、OC1N 的有效电平取决于 CC1P、CC1NP 位。 000：冻结。输出比较寄存器 TIMx_CCR1 与计数器 TIMx_CNT 间的比较对 OC1REF 不起作用 001：匹配时设置通道 1 为有效电平。当计数器 TIMx_CNT 的值与捕获/比较寄存器 1(TIMx_CCR1) 相同时，强制 OC1REF 为高 010：匹配时设置通道 1 为无效电平。当计数器 TIMx_CNT 的值与捕获/比较寄存器 1(TIMx_CCR1) 相同时，强制 OC1REF 为低 011：翻转。当 TIMx_CCR1=TIMx_CNT 时，翻转 OC1REF 的电平 100：强制为无效电平。强制 OC1REF 为低 101：强制为有效电平。强制 OC1REF 为高 110：PWM 模式 1 - 在向上计数时，当 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，当 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，当 TIMx_CNT < TIMx_CCR1 时通道 1 为无效电平，否则为有效电平；在向下计数时，当 TIMx_CNT > TIMx_CCR1 时通道 1 为有效电平，否则为无效电平 注 1：当 LOCK 级别设为 3(TIMx_BDTR 寄存器中的 LOCK 位) 并且 CC1S = 00(该通道配置成输出) 时，该位不能被修改。 注 2：在 PWM 模式 1 或 PWM 模式 2 中，只有当比较结果改变了或在输出比较模式中从冻结模式切换到 PWM 模式时，OC1REF 电平才改变。
3	OC1PE	rw	0x00	输出比较 1 预装载使能 (Output compare 1 preload enable) 0：禁止 TIMx_CCR1 寄存器的预装载功能，可随时写入 TIMx_CCR1 寄存器，并且新写入的数值立即起作用 1：开启 TIMx_CCR1 寄存器的预装载功能，读写操作仅对预装载寄存器操作，TIMx_CCR1 的预装载值在更新事件到来时被加载至当前寄存器中 注 1：当 LOCK 级别设为 3(TIMx_BDTR 寄存器中的 LOCK 位) 并且 CC1S = 00(该通道配置成输出) 时，该位不能被修改。 注 2：仅在单脉冲模式下 (TIMx_CR1 寄存器的 OPM = 1)，可以在未确认预装载寄存器情况下使用 PWM 模式，否则其动作不确定。

Bit	Field	Type	Reset	Description
2	OC1FE	rw	0x00	输出比较 1 快速使能 (Output compare 1 fast enable) 该位用于加快 CC 输出对触发输入事件的响应。 0：根据计数器与 CCR1 的值，CC1 正常操作，即使触发器是打开的。当触发器的输入有一个有效沿时，激活 CC1 输出的最小延时为 5 个时钟周期 1:输入到触发器的有效沿的作用就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
1：0	CC1S	rw	0x00	捕获/比较 1 选择 (Capture/Compare 1 selection) 这 2 位定义通道的方向 (输入/输出)，及输入脚的选择： 00：CC1 通道被配置为输出 01：CC1 通道被配置为输入，IC1 映射在 TI1 上 10：CC1 通道被配置为输入，IC1 映射在 TI2 上 11：CC1 通道被配置为输入，IC1 映射在 TRC 上，此模式仅工作在内部触发器输入被选中时 (由 TIMx_SMCR 寄存器的 TS 位选择) 注：CC1S 仅在通道关闭时 (TIMx_CCER 寄存器的 CC1E = 0) 才是可写的。

输入捕获模式：

Bit	Field	Type	Reset	Description
15：12	IC2F	rw	0x00	输入捕获 2 滤波器 (Input capture 2 filter)
11：10	IC2PSC	rw	0x00	输入/捕获 2 预分频器 (Input capture 2 prescaler)
9：8	CC2S	rw	0x00	捕获/比较 2 选择 (Capture/Compare 2 selection) 这 2 位定义通道的方向 (输入/输出)，及输入脚的选择： 00：CC2 通道被配置为输出 01：CC2 通道被配置为输入，IC2 映射在 TI2 上 10：CC2 通道被配置为输入，IC2 映射在 TI1 上 11：CC2 通道被配置为输入，IC2 映射在 TRC 上，此模式仅工作在内部触发器输入被选中时 (由 TIMx_SMCR 寄存器的 TS 位选择) 注：CC2S 仅在通道关闭时 (TIMx_CCER 寄存器的 CC2E = 0) 才是可写的。

Bit	Field	Type	Reset	Description
7：4	IC1F	rw	0x00	输入捕获 1 滤波器 (Input capture 1 filter) 这几位定义了 TI1 输入的采样频率及数字滤波器长度。数字滤波器由一个事件计数器组成，它记录到 N 个事件后会产生一个输出的跳变： 0000：无滤波器，以 fDTS 采样 1000：采样频率 $f_{SAMPLING}=f_{DTS}/8$，N = 6 0001：采样频率 $f_{SAMPLING}=f_{CK\_INT}$，N = 2 1001：采样频率 $f_{SAMPLING}=f_{DTS}/8$，N = 8 0010：采样频率 $f_{SAMPLING}=f_{CK\_INT}$，N = 4 1010：采样频率 $f_{SAMPLING}=f_{DTS}/16$，N = 5 0011：采样频率 $f_{SAMPLING}=f_{CK\_INT}$，N = 8 1011：采样频率 $f_{SAMPLING}=f_{DTS}/16$，N = 6 0100：采样频率 $f_{SAMPLING}=f_{DTS}/2$，N = 6 1100：采样频率 $f_{SAMPLING}=f_{DTS}/16$，N = 8 0101：采样频率 $f_{SAMPLING}=f_{DTS}/2$，N = 8 1101：采样频率 $f_{SAMPLING}=f_{DTS}/32$，N = 5 0110：采样频率 $f_{SAMPLING}=f_{DTS}/4$，N = 6 1110：采样频率 $f_{SAMPLING}=f_{DTS}/32$，N = 6 0111：采样频率 $f_{SAMPLING}=f_{DTS}/4$，N = 8 1111：采样频率 $f_{SAMPLING}=f_{DTS}/32$，N = 8
3：2	IC1PSC	rw	0x00	输入/捕获 1 预分频器 (Input capture 1 prescaler) 这 2 位定义了 CC1 输入 (IC1) 的预分频系数。 当 CC1E = 0(TIMx_CCER 寄存器中) 时，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
1：0	CC1S	rw	0x00	捕获/比较 1 选择 (Capture/compare 1 selection) 这 2 位定义通道的方向 (输入/输出)，及输入脚的选择： 00：CC1 通道被配置为输出 01：CC1 通道被配置为输入，IC1 映射在 TI1 上 10：CC1 通道被配置为输入，IC1 映射在 TI2 上 11：CC1 通道被配置为输入，IC1 映射在 TRC 上，此模式仅工作在内部触发器输入被选中时 (由 TIMx_SMCR 寄存器的 TS 位选择) 注：CC1S 仅在通道关闭时 (TIMx_CCER 寄存器的 CC1E = 0) 才是可写的。

12.4.8 捕获/比较模式寄存器 2(TIMx_CCMR2)

偏移地址：0x1C

复位值：0x0000

参看以上 CCMR1 寄存器的描述

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

输出比较模式

Bit	Field	Type	Reset	Description
15	OC4CE	rw	0x00	输出比较 4 清 0 使能 (Output compare 4 clear enable)
14：12	OC4M	rw	0x00	输出比较 4 模式 (Output compare 4 mode)
11	OC4PE	rw	0x00	输出比较 4 预装载使能 (Output compare 4 preload enable)
10	OC4FE	rw	0x00	输出比较 4 快速使能 (Output compare 4 fast enable)
9：8	CC4S	rw	0x00	捕获/比较 4 选择 (Capture/Compare 4 selection) 该 2 位定义通道的方向 (输入/输出)，及输入脚的选择： 00：CC4 通道被配置为输出 01：CC4 通道被配置为输入，IC4 映射在 TI4 上 10：CC4 通道被配置为输入，IC4 映射在 TI3 上 11：CC4 通道被配置为输入，IC4 映射在 TRC 上，此模式仅工作在内部触发器输入被选中时 (由 TIMx_SMCR 寄存器的 TS 位选择) 注：CC4S 仅在通道关闭时 (TIMx_CCER 寄存器的 CC4E = 0) 才是可写的。
7	OC3CE	rw	0x00	输出比较 3 清‘0’使能 (Output compare 3 clear enable)
6：4	OC3M	rw	0x00	输出比较 3 模式 (Output compare 3 mode)
3	OC3PE	rw	0x00	输出比较 3 预装载使能 (Output compare 3 preload enable)
2	OC3FE	rw	0x00	输出比较 3 快速使能 (Output compare 3 fast enable)
1：0	CC3S	rw	0x00	捕获/比较 3 选择 (Capture/Compare 3 selection) 这 2 位定义通道的方向 (输入/输出)，及输入脚的选择： 00：CC3 通道被配置为输出 01：CC3 通道被配置为输入，IC3 映射在 TI3 上 10：CC3 通道被配置为输入，IC3 映射在 TI4 上 11：CC3 通道被配置为输入，IC3 映射在 TRC 上，此模式仅工作在内部触发器输入被选中时 (由 TIMx_SMCR 寄存器的 TS 位选择) 注：CC3S 仅在通道关闭时 (TIMx_CCER 寄存器的 CC3E = 0) 才是可写的。

输入捕获模式

Bit	Field	Type	Reset	Description
15：12	IC4F	rw	0x00	输入捕获 4 滤波器 (Input capture 4 filter)
11：10	IC4PSC	rw	0x00	输入/捕获 4 预分频器 (Input capture 4 prescaler)
9：8	CC4S	rw	0x00	捕获/比较 4 选择 (Capture/Compare 4 selection) 这 2 位定义通道的方向 (输入/输出)，及输入脚的选择： 00：CC4 通道被配置为输出 01：CC4 通道被配置为输入，IC4 映射在 TI4 上 10：CC4 通道被配置为输入，IC4 映射在 TI3 上 11：CC4 通道被配置为输入，IC4 映射在 TRC 上，此模式仅工作在内部触发器输入被选中时 (由 TIMx_SMCR 寄存器的 TS 位选择) 注：CC4S 仅在通道关闭时 (TIMx_CCER 寄存器的 CC4E = 0) 才是可写的。
7：4	IC3F	rw	0x00	输入捕获 3 滤波器 (Input capture 3 filter)
3：2	IC3PSC	rw	0x00	输入/捕获 3 预分频器 (Input capture 3 prescaler)
1：0	CC3S	rw	0x00	捕获/比较 3 选择 (Capture/compare 3 selection) 这 2 位定义通道的方向 (输入/输出)，及输入脚的选择： 00：CC3 通道被配置为输出 01：CC3 通道被配置为输入，IC3 映射在 TI3 上 10：CC3 通道被配置为输入，IC3 映射在 TI4 上 11：CC3 通道被配置为输入，IC3 映射在 TRC 上，此模式仅工作在内部触发器输入被选中时 (由 TIMx_SMCR 寄存器的 TS 位选择) 注：CC3S 仅在通道关闭时 (TIMx_CCER 寄存器的 CC3E = 0) 才是可写的。

12.4.9 捕获/比较使能寄存器 (TIMx_CCER)

偏移地址：0x20

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.		CC4P	CC4E	Res.		CC3P	CC3E	Res.		CC2P	CC2E	Res.		CC1P	CC1E
		rw	rw			rw	rw			rw	rw			rw	rw

Bit	Field	Type	Reset	Description
15：14	Reserved			保留, 读为 0
13	CC4P	rw	0x00	输入/捕获 4 输出极性 (Capture/Compare 4 output polarity) 参考 CC1P 的描述。
12	CC4E	rw	0x00	输入/捕获 4 输出使能 (Capture/Compare 4 output enable) 参考 CC1E 的描述。
11：10	Reserved			保留, 读为 0
9	CC3P	rw	0x00	输入/捕获 3 输出极性 (Capture/Compare 3 output polarity) 参考 CC1P 的描述。

Bit	Field	Type	Reset	Description
8	CC3E	rw	0x00	输入/捕获 3 输出使能 (Capture/Compare 3 output enable) 参考 CC1E 的描述。
7：6	Reserved			保留, 读为 0
5	CC2P	rw	0x00	输入/捕获 2 输出极性 (Capture/Compare 2 output polarity) 参考 CC1P 的描述。
4	CC2E	rw	0x00	输入/捕获 2 输出使能 (Capture/Compare 2 output enable) 参考 CC1E 的描述。
3：2	Reserved			保留, 读为 0
1	CC1P	rw	0x00	输入/捕获 1 输出极性 (Capture/Compare 1 output polarity) CC1 通道配置为输出： 0：OC1 高电平有效 1：OC1 低电平有效 CC1 通道配置为输入： 该位选择是 IC1 还是 IC1 的反相信号作为触发或捕获信号。 0：不反相：捕获发生在 IC1 的上升沿；当用作外部触发器时，IC1 不反相 1：反相：捕获发生在 IC1 的下降沿；当用作外部触发器时，IC1 反相 注：当 LOCK 级别 (TIMx_BDTR 寄存器中的 LCCK 位) 设为 3 或 2 时，该位不能被修改。
0	CC1E	rw	0x00	输入/捕获 1 输出使能 (Capture/Compare 1 output enable) CC1 通道配置为输出： 0：关闭 - OC1 禁止输出，因此 OC1 的输出电平依赖于 MOE、OSSI、OSSR、OIS1、OIS1N 和 CC1NE 位的值 1：开启 - OC1 信号输出到对应的输出引脚，其输出电平依赖于 MOE、OSSI、OSSR、OIS1、OIS1N 和 CC1NE 位的值 CC1 通道配置为输入： 该位决定了计数器的值是否能捕获入 TIMx_CCR1 寄存器。 0：捕获禁止 1：捕获使能

表 54. 标准 Ocx 通道的输出控制位

CCxE 位	OCx 输出状态
0	禁止输出 (OCx = 0，OCx_EN = 0)
1	OCx = OCxREF + 极性，OCx_EN = 1

注：管脚连接到标准的 OCx 通道的外部 I/O 管脚的状态，取决于 OCx 通道状态和 GPIO 以及 AFIO 寄存器。

12.4.10 计数器 (TIMx_CNT)

偏移地址：0x24

复位值：0x0000

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

Bit	Field	Type	Reset	Description
15：0	CNT	rw	0x0000	计数器的值 (Counter value)

12.4.11 预分频器 (TIMx_PSC)

偏移地址：0x28

复位值：0x0000

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

Bit	Field	Type	Reset	Description
15：0	PSC	rw	0x0000	预分频器的值 (Prescaler value) 计数器的时钟频率 (CK_CNT) 等于 f_{CK_PSC} /(PSC + 1)。 PSC 包含了每次当更新事件产生时，装入当前预分频器寄存器的值。更新事件包括计数器被 TIM_EGR 的 UG 位清‘0’或被工作在复位模式的从控制器清‘0’。

12.4.12 自动装载寄存器 (TIMx_ARR)

偏移地址：0x2C

复位值：0x0000

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

Bit	Field	Type	Reset	Description
15：0	ARR	rw	0x0000	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数值。 详细参考小节 12.3.1：有关 ARR 的更新和动作。 当自动重装载的值为空时，计数器不工作。

12.4.13 捕获/比较寄存器 1(TIMx_CCR1)

偏移地址：0x34

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
15：0	CCR1	rw	0x0000	捕获/比较 1 的值 (Capture/Compare 1 value) 若 CC1 通道配置为输出： CCR1 包含了装入当前捕获/比较 1 寄存器的值 (预装载值)。 如果在 TIMx_CCMR1 寄存器 (OC1PE 位) 中未选择预装载功能，写入的数值会立即传输至当前寄存器中。否则只有当更新事件发生时，此预装载值才传输至当前捕获/比较 1 寄存器中。当前捕获/比较寄存器参与同计数器 TIMx_CNT 的比较，并在 OC1 端口上产生输出信号。 若 CC1 通道配置为输入： CCR1 包含了由上一次输入捕获 1 事件 (IC1) 传输的计数器值。

12.4.14 捕获/比较寄存器 2(TIMx_CCR2)

偏移地址：0x38

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR2															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
15：0	CCR2	rw	0x0000	捕获/比较 2 的值 (Capture/Compare 2 value) 若 CC2 通道配置为输出： CCR2 包含了装入当前捕获/比较 2 寄存器的值 (预装载值)。 如果在 TIMx_CCMR2 寄存器 (OC2PE 位) 中未选择预装载特性，写入的数值会立即传输至当前寄存器中。否则只有当更新事件发生时，此预装载值才传输至当前捕获/比较 2 寄存器中。当前捕获/比较寄存器参与同计数器 TIMx_CNT 的比较，并在 OC2 端口上产生输出信号。 若 CC2 通道配置为输入： CCR2 包含了由上一次输入捕获 2 事件 (IC2) 传输的计数器值。

12.4.15 捕获/比较寄存器 3(TIMx_CCR3)

偏移地址：0x3C

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR3															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
15：0	CCR3	rw	0x0000	捕获/比较 3 的值 (Capture/Compare 3 value) 若 CC3 通道配置为输出： CCR3 包含了装入当前捕获/比较 3 寄存器的值 (预装载值)。 如果在 TIMx_CCMR3 寄存器 (OC3PE 位) 中未选择预装载特性，写入的数值会立即传输至当前寄存器中。否则只有当更新事件发生时，此预装载值才传输至当前捕获/比较 3 寄存器中。当前捕获/比较寄存器参与同计数器 TIMx_CNT 的比较，并在 OC3 端口上产生输出信号。若 CC3 通道配置为输入： CCR3 包含了由上一次输入捕获 3 事件 (IC3) 传输的计数器值。

12.4.16 捕获/比较寄存器 4(TIMx_CCR4)

偏移地址：0x40

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR4															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
15：0	CCR4	rw	0x0000	捕获/比较 4 的值 (Capture/Compare 4 value) 若 CC4 通道配置为输出： CCR4 包含了装入当前捕获/比较 4 寄存器的值 (预装载值)。 如果在 TIMx_CCMR4 寄存器 (OC4PE 位) 中未选择预装载特性，写入的数值会立即传输至当前寄存器中。否则只有当更新事件发生时，此预装载值才传输至当前捕获/比较 4 寄存器中。当前捕获/比较寄存器参与同计数器 TIMx_CNT 的比较，并在 OC4 端口上产生输出信号。 若 CC4 通道配置为输入： CCR4 包含了由上一次输入捕获 4 事件 (IC4) 传输的计数器值。

12.4.17 DMA 控制寄存器 (TIMx_DCR)

偏移地址：0x48

复位值：0x0000

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

Bit	Field	Type	Reset	Description
15：13	Reserved			保留，始终读为 0。
12：8	DBL	w	0x00	DMA 连续传送长度 (DMA burst length) 这些位定义了 DMA 在连续模式下的传送长度 (当对 TIMx_DMAR 寄存器进行写操作时，定时器则进行一次连续传送)，即：定义传输的次数，传输可以是半字 (双字节) 或字节： 00000：1 次传输 00001：2 次传输 00010：3 次传输..... ..... 10001：18 次传输 例：我们考虑这样的传输：DBL = 7，DBA = TIM2_CR1 - 如果 DBL = 7，DBA = TIM2_CR1 表示待传输数据的地址，那么传输的地址由下式给出： (TIMx_CR1 的地址)+ DBA +(DMA 索引)，其中 DMA 索引 = DBL 其中 (TIMx_CR1 的地址)+ DBA 再加上 7，给出了将要写入或者读出数据的地址，这样数据的传输将发生在从地址 (TIMx_CR1 的地址)+ DBA 开始的 7 个寄存器。根据 DMA 数据长度的设置，可能发生以下情况： - 如果设置数据为半字 (16 位)，那么数据就会传输给全部 7 个寄存器。 - 如果设置数据为字节，数据仍然会传输给全部 7 个寄存器：第一个寄存器包含第一个 MSB 字节，第二个寄存器包含第一个 LSB 字节，以此类推。因此对于定时器，用户必须指定由 DMA 传输的数据宽度。
7：5	Reserved			保留，始终读为 0。
4：0	DBA	w	0x00	DMA 基地址 (DMA base address) 这些位定义了 DMA 在连续模式下的基地址 (当对 TIMx_DMAR 寄存器进行写操作时)，DBA 定义为从 TIMx_CR1 寄存器所在地址开始的偏移量： 00000：TIMx_CR1 00001：TIMx_CR2 00010：TIMx_SMCR .....

12.4.18 连续模式的 DMA 地址 (TIMx_DMAR)

偏移地址：0x4C

复位值：0x0000

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

Bit	Field	Type	Reset	Description
15：0	DMAB	w	0x0000	DMA 连续传送寄存器 (DMA register for burst accesses) 对 TIMx_DMAR 寄存器的写操作会导致对以下地址所在寄存器的存取操作： TIMx_CR1 地址 + DBA + DMA 索引，其中：‘TIMx_CR1 地址’是控制寄存器 1(TIMx_CR1) 所在的地址； ‘DBA’是 TIMx_DCR 寄存器中定义的基地址； ‘DMA 索引’是由 DMA 自动控制的偏移量，它取决于 TIMx_DCR 寄存器中定义的 DBL。

13 实时时钟 (RTC)

实时时钟 (RTC)

13.1 RTC 简介

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

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

系统复位后，对后备寄存器和 RTC 的访问被禁止，这是为了防止对后备区 (BKP) 的意外写操作。执行以下操作将使能对后备寄存器和 RTC 的访问。

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

13.2 主要特征

- 可编程的预分频系数：分频系数最高为 2^{20}
- 32 位的可编程计数器，用于较长时间段的测量
- 2 个分离的时钟：用于 APB1 接口的 PCLK1 和 RTC 时钟 (RTC 时钟的频率必须小于 PCLK1 时钟频率的四分之一以上)
- 可以选择以下三种 RTC 的时钟源
  - HSE 时钟除以 128
  - LSE 振荡器时钟
  - LSI 振荡器时钟
- 2 个独立的复位类型：
  - APB1 接口由系统复位
  - RTC 核心 (预分频器、闹钟、计数器和分频器) 只能由后备域复位
- 3 个专门的屏蔽中断：
  - 闹钟中断，用来产生一个软件可编程的闹钟中断
  - 秒中断，用来产生一个可编程的周期性中断信号 (最长可达 1 秒)
  - 溢出中断，指示内部可编程计数器溢出并返回为 0 的状态

13.3 功能描述

13.3.1 概述

RTC 由两个主要部分组成。参见下图。第一部分 (APB1 接口) 用来和 APB1 总线相连。此单元还包含一组 16 位寄存器，可以通过 APB 总线对其进行读写操作。APB1 接口由 APB1

总线时钟驱动，用来与 APB1 总线接口。

另一部分 (RTC 核心) 由一组可编程计数器组成，分成两个主要模块。第一个模块是 RTC 的预分频模块，它可编程产生最长为 1 秒的 RTC 时间基准 TR_CLK。RTC 的预分频模块包含了一个 20 位的可编程预分频器 (RTC 预分频器)。如果在 RTC_CR 寄存器中设置了相应的允许位，则在每个 TR_CLK 周期中 RTC 产生一个中断 (秒中断)。第二个模块是一个 32 位的可编程计数器，可被初始化为当前的系统时间。系统时间按 TR_CLK 周期累加与存储在 RTC_ALR 寄存器中的可编程时间比较，如果 RTC_CR 控制寄存器中设置了相应的允许位，比较匹配时将产生一个闹钟中断。

下图简化的 RTC 框图

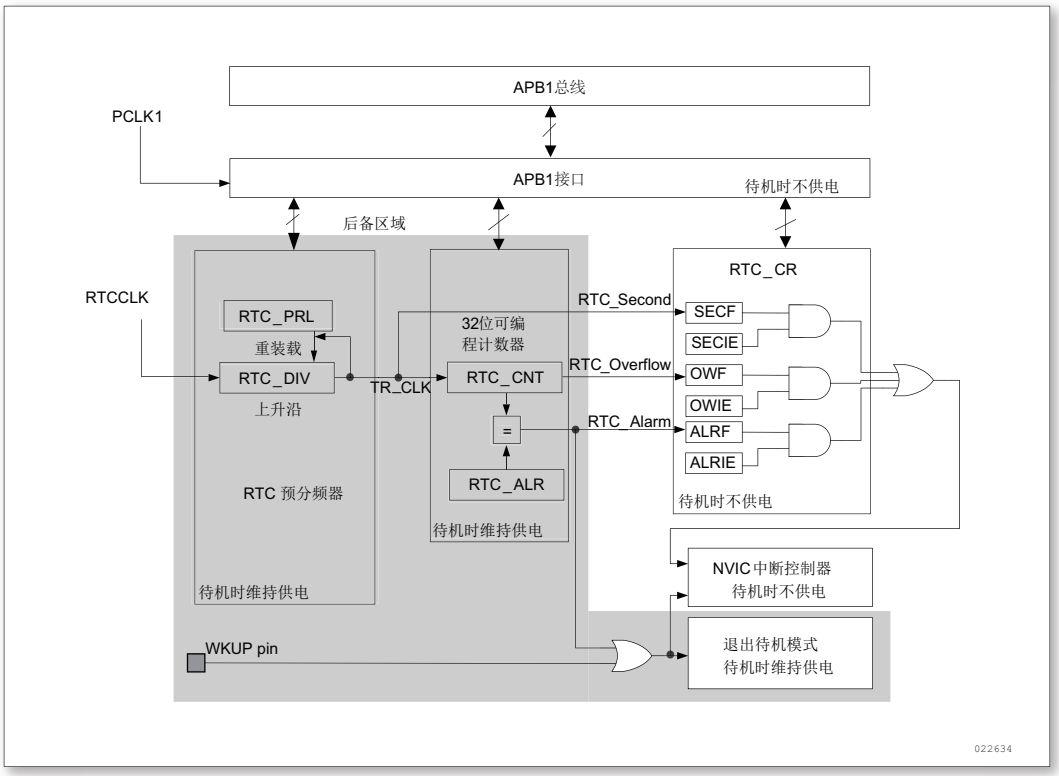


图 122. 实时时钟方框图

13.3.2 复位过程

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

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

13.3.3 读 RTC 寄存器

RTC 核完全独立于 RTC APB1 接口。

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

这意味着，如果 APB1 接口曾经被关闭，而读操作又是在刚刚重新开启 APB1 之后，则在第一次内部寄存器更新之前，从 APB1 上读出 RTC 寄存器数值可能被破坏了 (通常读到 0)。

下述几种情况下能够发生这种情形：

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

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

因此，若在读取 RTC 寄存器时，RTC 的 APB1 接口曾经处于禁止状态，则软件首先必须等待 RTC_CRL 寄存器中的 RSF 位 (寄存器同步标志) 被硬件置‘1’。

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

13.3.4 配置 RTC 寄存器

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

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

配置过程：

- 查询 RTOFF 位，直到 RTOFF 的值变为‘1’
- 置 CNF 值为‘1’，进入配置模式
- 对一个或多个 RTC 寄存器进行写操作
- 清除 CNF 标志位，退出配置模式
- 查询 RTOFF，直至 RTOFF 位变为‘1’以确认写操作已经完成
- 仅当 CNF 标志位被清除时，写操作才能进行，这个过程至少需要 3 RTCCLK 周期

13.3.5 RTC 标志的设置

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

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

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

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

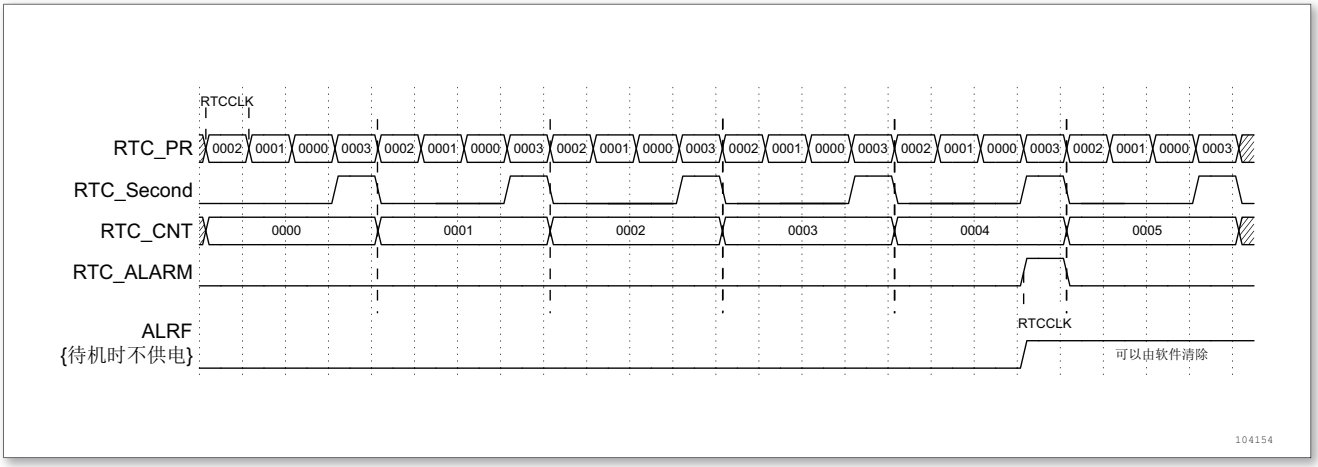


图 123. RTC 秒和闹钟波形图示例，PR = 0003，ALARM = 00004

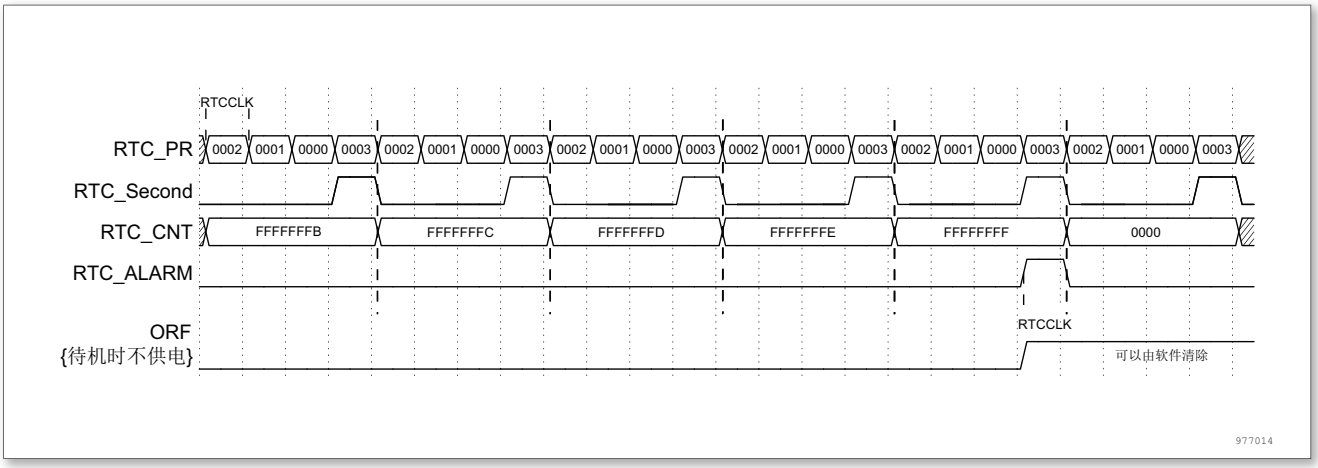


图 124. RTC 溢出波形图示例，PR = 0003

13.4 RTC 寄存器

表 55. RTC 寄存器概览

Offset	Acronym	Register Name	Reset	Section
0x00	RTC_CRH	RTC 控制寄存器高位	0x00000000	小节 13.4.1
0x04	RTC_CRL	控制寄存器低位	0x00000020	小节 13.4.2
0x08	RTC_PRLH	RTC 预分频装载寄存器高位	0x00000000	小节 13.4.3
0x0C	RTC_PRLL	RTC 预分频装载寄存器低位	0x00000000	小节 13.4.3
0x10	RTC_DIVH	RTC 预分频器分频因子寄存器高位	0x00000000	小节 13.4.4
0x14	RTC_DIVL	RTC 预分频器分频因子寄存器低位	0x00000000	小节 13.4.4
0x18	RTC_CNTH	RTC 计数器寄存器高位	0x00000000	小节 13.4.5
0x1C	RTC_CNTL	RTC 计数器寄存器低位	0x00000000	小节 13.4.5
0x20	RTC_ALRH	RTC 闹钟寄存器高位	0x0000FFFF	小节 13.4.6
0x24	RTC_ALRL	RTC 闹钟寄存器低位	0x0000FFFF	小节 13.4.6

13.4.1 RTC 控制寄存器 (RTC_CRH)

地址偏移量：0x00

复位值：0x0000

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

Bit	Field	Type	Reset	Description
15 : 3	Reserved			始终读为 0。
2	OWIE	rw	0x00	允许溢出中断位 (Overflow interrupt enable) 0：屏蔽 (不允许) 溢出中断 1：允许溢出中断
1	ALRIE	rw	0x00	允许闹钟中断 (Alarm interrupt enable) 0：屏蔽 (不允许) 闹钟中断 1：允许闹钟中断
0	SECIE	rw	0x00	允许秒中断 (Second interrupt enable) 0：屏蔽 (不允许) 秒中断 1：允许秒中断

这些位用来屏蔽中断请求。

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

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

13.4.2 RTC 控制寄存器低位 (RTC_CRL)

地址偏移量：0x04

复位值：0x0020

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved										RTOFF	CNF	RSF	OWF	ALRF	SECF
										r	rw	rc_w0	rc_w0	rc_w0	rc_w0

Bit	Field	Type	Reset	Description
15 : 6	Reserved			始终读为 0。
5	RTOFF	r	0x01	RTC 操作关闭 (RTC operation OFF) RTC 模块利用这位来指示对其寄存器进行的最后一次操作的状态，指示操作是否完成。若此位为‘0’，则表示无法对任何的 RTC 寄存器进行读写操作。此位为只读位。 0：上一次对 RTC 寄存器的写操作仍在进行 1：上一次对 RTC 寄存器的写操作已经完成

Bit	Field	Type	Reset	Description
4	CNF	rw	0x00	CNF：配置标志 (Configuration flag) 此位必须由软件置‘1’以进入配置模式，从而允许 RTC_CNTL/H、RTC_ALRL/H 或 RTC_PRLL/H 寄存器写入数据。只有当此位在被置‘1’并重新由软件清‘0’后，才会执行写操作。 0：退出配置模式 (开始更新 RTC 寄存器) 1：进入配置模式
3	RSF	rc_w0	0x00	寄存器同步标志 (Registers synchronized flag) 每当 RTC_CNT 寄存器和 RTC_DIV 寄存器由软件更新或清‘0’时，此位由硬件置‘1’。在 APB1 复位后，或 APB1 时钟停止后，此位必须由软件清‘0’。要进行任何的读操作之前，用户程序必须等待这位被硬件置‘1’，以确保 RTC_CNT、RTCALR 或 RTC_PRL 已经被同步。 0：寄存器尚未被同步 1：寄存器已经被同步
2	OWF	rc_w0	0x00	溢出标志 (Overflow flag) 当 32 位可编程计数器溢出时，此位由硬件置‘1’。如果 RTC_CRH 寄存器中的 OWIE = 1，则产生中断。此位只能由软件清‘0’。对此写‘1’无效 0：无溢出 1：32 位可编程计数器溢出
1	ALRF	rc_w0	0x00	闹钟标志 (Alarm flag) 当 32 位可编程计数器到达了 RTC_ALR 寄存器所设置的预定值，此位由硬件置‘1’。如果 RTC_CRH 寄存器中的 ALRIE = 1，则产生中断。此位只能由软件清‘0’。对此位写‘1’是无效的。 0：无闹钟 1：有闹钟
0	SECF	rc_w0	0x00	秒标志 (Second flag) 当 32 位可编程预分频器溢出时，此位由硬件置‘1’同时 RTC 计数器加 1。因此，此标志为分辨率可编程的 RTC 计数器提供了一个周期性信号 (通常为 1 秒)。如果 RTC_CRH 寄存器 SECIE = 1，则产生中断。此位只能由软件清除。对此位写‘1’是无效的。 0：秒标志条件不成立 1：秒标志条件成立

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

注：1. 任何标志位都将保持挂起状态，直到适当的 RTC_CR 请求位被软件复位，表示所请求的中断已经被接受。2. 在复位时禁止所有中断，无挂起的中断请求，可以对 RTC 寄存器进行写操作。3. 当 APB1 时钟不运行时，OWF、ALRF、SECF 和 RSF 位不被更新。4. OWF、ALRF、SECF 和 RSF

位只能由硬件置位，由软件来清零。5. 若 ALRF=1 且 ALRIE=1，则允许产生 RTC 全局中断。如果在 EXTI 控制寄存器中允许产生 EXTI 线 17 中断，则允许产生 RTC 全局中断和 RTC 闹钟中断。6. 若 ALRF=1，如果在 EXTI 控制器中设置了 EXTI 线 17 的中断模式，则允许产生 RTC 闹钟中断；如果在 EXTI 控制器中设置了 EXTI 线 17 的事件模式，则这条线上会产生一个脉冲 (不会产生 RTC 闹钟中断)。

13.4.3 RTC 预分频装载寄存器 (RTC_PRLH/RTC_PRLL)

RTC 预分频装载寄存器高位 (RTC_PRLH)

地址偏移量：0x08

复位值：0x0000

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

Bit	Field	Type	Reset	Description
15 : 4	Reserved			始终读为 0。
3 : 0	PRL	w	0x00	RTC 预分频器装载值高位 (RTC prescaler reload value high) 根据以下公式，这些位用来定义计数器的时钟频率： $f_{TR\_CLK} = f_{RTCCLK} /(PRL + 1)$ 注：不推荐使用 0 值，否则无法正确产生 RTC 中断和标志位。

RTC 预分频装载寄存器低位 (RTC_PRLL)

地址偏移量：0x0C

复位值：0x0000

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

Bit	Field	Type	Reset	Description
15 : 0	PRL	w	0x0000	RTC 预分频器装载值高位 (RTC prescaler reload value high) 根据以下公式，这些位用来定义计数器的时钟频率： $f_{TR\_CLK} = f_{RTCCLK} /(PRL + 1)$

注：如果输入时钟频率是 32.768KHz(f_{RTCCLK})，这个寄存器中写入 7FFFh 可获得周期为 1 秒的信号。

13.4.4 预分频器分频因子寄存器 (RTC_DIVH/RTC_DIVL)

在 TR_CLK 的每个周期里，RTC 预分频器中计数器的值都会被重新设置为 RTC_PRL 寄存器的值。用户可通过读取 RTC_DIV 寄存器，以获得预分频计数器的当前值，而不停止分频计数器的工作，从而获得精确的时间测量。此寄存器是只读寄存器，其值在 RTC_PRL 或 RTC_CNT 寄存器中的值发生改变后，由硬件重新装载。

RTC 预分频器分频因子寄存器高位 (RTC_DIVH)

地址偏移量：0x10

复位值：0x0000

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

Bit	Field	Type	Reset	Description
15 : 4	Reserved			始终读为 0。
3 : 0	DIV	r	0x00	RTC 时钟分频器分频因子高位 (RTC clock divider high)

RTC 预分频器分频因子寄存器低位 (RTC_DIVL)

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
DIV															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bit	Field	Type	Reset	Description
15 : 0	DIV	r	0x0000	RTC 时钟分频器分频因子高位 (RTC clock divider low)

13.4.5 RTC 计数器寄存器 (RTC_CNTH/RTC_CNTL)

RTC 核有一个 32 位可编程的计数器，可通过两个 16 位的寄存器访问。计数器以预分频器产生的 TR_CLK 时间基准为参考进行计数。RTC_CNT 寄存器用于存放计数器的计数值。他们受 RTC_CR 的位 RTOFF 写保护，仅当 RTOFF 值为‘1’时，允许写操作。在高或低寄存器 (RTC_CNTH 或 RTC_CNTL) 上的写操作，能够直接装载到相应的可编程计数器，并且重新装载 RTC 预分频器。在进行读操作时，直接返回计数器内的计数值 (系统时间)。

RTC 计数器寄存器高位 (RTC_CNTH)

偏移地址：0x18

复位值：0x0000

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

Bit	Field	Type	Reset	Description
15 : 0	CNT	rw	0x0000	RTC 计数器高位 (RTC counter high) 可通过读 RTC_CNTH 寄存器来获得 RTC 计数器当前值的高位部分。要对此寄存器进行写操作前，必须先进入配置模式。

RTC 计数器寄存器低位 (RTC_CNTL)

偏移地址：0x1C

复位值：0x0000

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

Bit	Field	Type	Reset	Description
15 : 0	CNT	rw	0x0000	RTC 计数器低位 (RTC counter low) 可通过读 RTC_CNTL 寄存器来获得 RTC 计数器当前值的低位部分。要对此寄存器进行写操作前，必须先进入配置模式。

13.4.6 闹钟寄存器 (RTC_ALRH/RTC_ALRL)

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

RTC 闹钟寄存器高位 (RTC_ALRH)

偏移地址：0x20

复位值：0xFFFF

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

Bit	Field	Type	Reset	Description
15 : 0	ALR	w	0xFFFF	RTC 闹钟值高位 (RTC alarm high) 此寄存器用来保护由软件写入的闹钟时间的高位部分。要对此寄存器进行写操作，必须先进入配置模式。

RTC 闹钟寄存器低位 (RTC_ALRL)

偏移地址：0x24

复位值：0xFFFF

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

Bit	Field	Type	Reset	Description
15 : 0	ALR	w	0xFFFF	RTC 计数器低位 (RTC counter low) 此寄存器用来保护由软件写入的闹钟时间的低位部分。要对此寄存器进行写操作，必须先进入配置模式。

14 独立看门狗 (IWDG)

独立看门狗 (IWDG)

14.1 (IWDG 简介)

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

独立看门狗（IWDG）由专门的低速时钟（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正在主程序外，能够完全独立工作，并且对时间精度要求低的场合。WWDG 最适合那些要求看门狗在精确计时窗口起作用的应用程序。

14.2 IWDG 主要性能

- 自由运行的递减计数器
- 时钟由独立的振荡器提供（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 0x0000 时产生复位。

14.3 IWDG 功能描述

下图为独立看门狗模块的功能框图。

在键寄存器（IWDG_KR）中写入 0xCCCC。开始启动独立看门狗；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（IWGD_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。

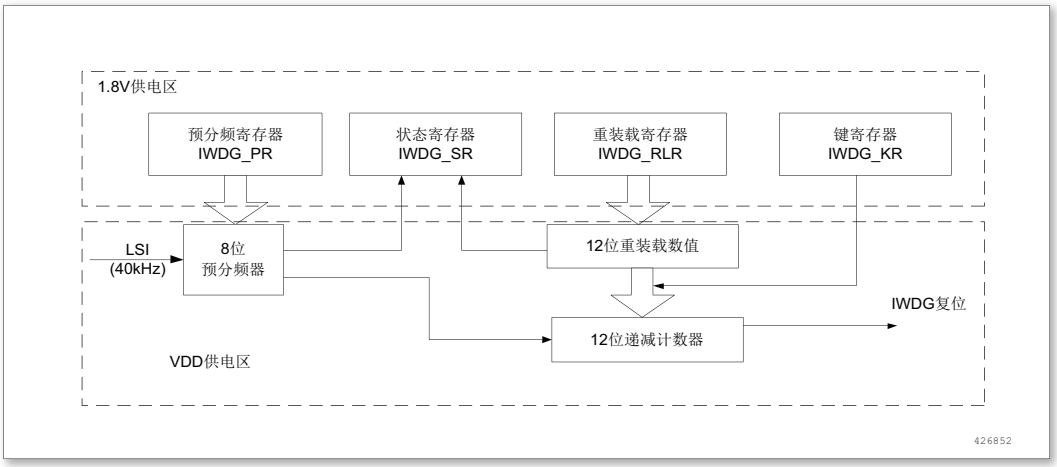


图 125. 独立看门狗框图

注：看门狗功能处于 V_{DD} 供电区，即在停机和待机模式时仍能正常工作。

表 56. 看门狗超时时间 (40KHz 的输入时钟 (LSI))

预分频系数	PR[2:0] 位	最短时间 RL[11:0]=0x000	最长时间
/4	0	0.1	409.6
/8	1	0.2	819.2
/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：这些时间是按照 40KHz 时钟给出。实际上，MCU 内部的振荡器频率会在 30KHz 到 60KHz 之间变化。

此外，即使振荡器的频率是精确的，确切的时序仍然依赖于 APB 接口时钟与振荡器时钟之间的相位差，因此总会有一个完整的振荡器周期是不确定的。

14.3.1 硬件看门狗

如果用户在选择字节中（请参考“嵌入式闪存”章节）启动了‘硬件看门狗’功能，在系统上电复位后，看门狗会自动开始运行；如果在计数器计数结束前，若软件没有向键寄存器写入相应的值，则系统会产生复位。

14.3.2 寄存器访问保护

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

状态寄存器指示预分频值和递减计数器是否正在被更新。

14.3.3 调试模式

当微控制器进入调试模式时（CPU 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。详见调试模块的章节。

14.4 IWDG 寄存器描述

表 57. IWDG 寄存器概览

Offset	Acronym	Register Name	Reset	Section
0x00	IWDG_KR	键寄存器	0x00000000	小节 14.4.1
0x04	IWDG_PR	预分频寄存器	0x00000000	小节 14.4.2
0x08	IWDG_RLR	重装载寄存器	0x00000FFF	小节 14.4.3
0x0C	IWDG_SR	状态寄存器	0x00000000	小节 14.4.4

14.4.1 键寄存器 (IWDG_KR)

偏移地址：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
Reserved															

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

Bit	Field	Type	Reset	Description
31 : 16	Reserved			始终读为 0。
15 : 0	KEY	w	0x0000	键值（只写寄存器，读出值为 0x0000）（Key value） 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。 写入 0xCCCC，启动看门狗工作。

14.4.2 预分频寄存器 (IWDG_PR)

偏移地址：0x04 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
Reserved															

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

Bit	Field	Type	Reset	Description
31：3	Reserved			始终读为 0。
2：0	PR	rw	0x00	预分频因子（Prescaler divider） 这些位具有写保护设置。通过设置这些位来选择计数器时钟的预分频因子。要改变预分频因子，IWDG_SR 寄存器的 PVU 位必须为 0。 000：预分频因子 = 4　　100：预分频因子 = 64 001：预分频因子 = 8　　101：预分频因子 = 128 010：预分频因子 = 16　　110：预分频因子 = 256 011：预分频因子 = 32　　111：预分频因子 = 256 注意：对此寄存器进行读操作，将从 V_{DD} 电压域返回预分频值。如果写操作正在进行，则读回的值可能是无效的。因此，只有对那个 IWDG_SR 寄存器的 PUV 位为 0 时，读出的值才有效。

14.4.3　重装载寄存器 (IWDG_RLR)

偏移地址：0x08

复位值：0x0000 0FFF（在待机模式复位）

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

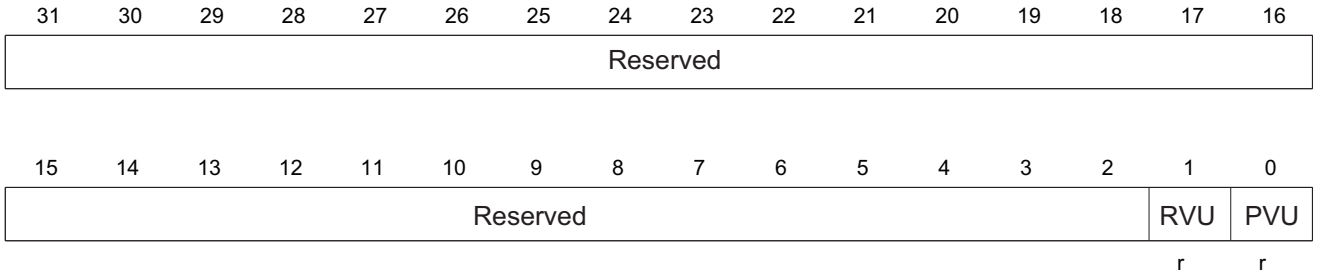
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				RL											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
31：12	Reserved			始终读为 0。
11：0	RL	rw	0xFFF	看门狗计数器重装载值（Watchdog counter reload value） 这些位具有写保护。用于定义看门狗计数器的重装载值，每当向 IWDG_KR 寄存器写入 0xAAAA 时，重装载值会被传送到计数器中。随后计数器从这个值开始递减计数。看门狗超时周期可通过次重装载值和时钟预分频值来计算。 注：对此寄存器进行读操作，将从 V_{DD} 电压域返回预分频值。如果写操作正在进行，则读回的值可能是无效的。因此，只有当 IWDG_SR 寄存器的 RUV 位为 0 时，读出的值才有效。

14.4.4　状态寄存器 (IWDG_SR)

偏移地址：0x0C

复位值：0x0000 0000（待机模式时不复位）



Bit	Field	Type	Reset	Description
31 : 2	Reserved			始终读为 0。
1	RVU	r	0x00	看门狗计数器重装载值更新（Watchdog counter reload value update） 此位由硬件置‘1’用来指示重装载值的更新正在进行中。当在 V_{DD} 域中的重装载更新结束后，此位由硬件清‘0’（最多需要 5 个 40KHz 的振荡器周期）重装载值只有在 RVU 位被清‘0’后才可更新。
0	PVU	r	0x00	看门狗预分频更新（Watchdog prescaler value update） 此位由硬件置‘1’用来指示预分频值的更新正在进行中。当在 V_{DD} 域中的预分频值更新结束后，此位由硬件清‘0’（最多需要 5 个 40KHz 的振荡器周期）预分频值只有在 RVU 位被清‘0’后才可更新。

注：如果在应用程序中使用多个重装载值或预分频值，则必须在 RVU 位被清除后才能重新改变预装载值，在 PVU 位被清除后才能重新改变预分频值。然而，在预分频和/或重装值更新后，不必等待 RVU 或 PVU 复位，可以继续执行下面的代码。（即使在低功耗模式下，次写操作仍会被继续执行完成）

15 窗口看门狗 (WWDG)

窗口看门狗 (WWDG)

15.1 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。

15.2 WWDG 主要特征

- 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。

15.3 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。

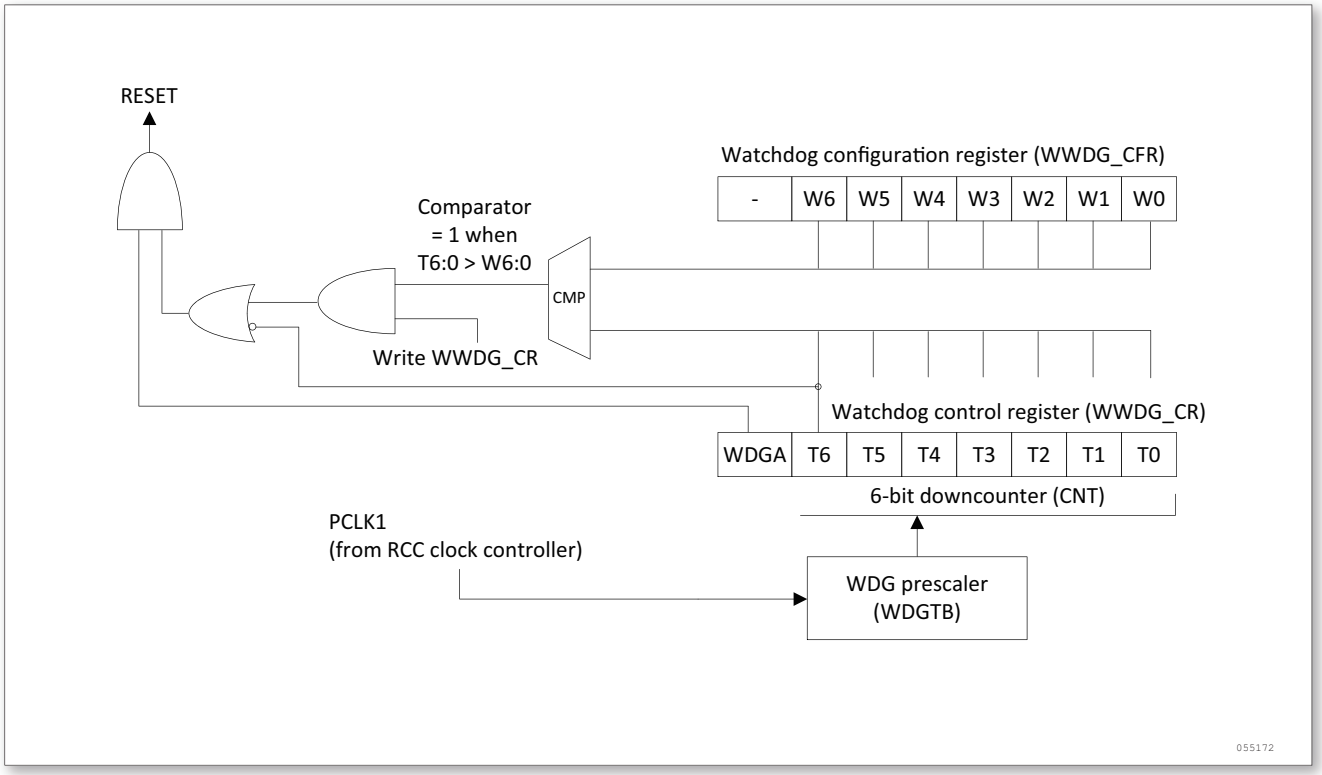


图 126. 看门狗框图

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

• 启动看门狗

在系统复位后，看门狗总是处于关闭状态，设置 WWDG_CR 寄存器的 WDGA 位能够开启看门狗，随后它不能再被关闭，除非发生复位。

• 控制递减计数器

递减计数器处于自由运行状态，即使看门狗被禁止，递减计数器仍继续递减计数。当看门狗被启用时，T6 位必须被设置，以防止立即产生一个复位。

T[5：0] 位包含了看门狗产生复位之前的计时数目；复位前的延时时间在一个最小值和一个最大值之间变化，这是因为写入 WWDG_CR 寄存器时，预分频值是未知的。

配置寄存器 (WWDG_CFR) 中包含窗口的上限值：要避免产生复位，递减计数器必须在其值小于窗口寄存器的数值并且大于 0x3F 时被重新装载，上图描述了窗口寄存器的工作过程。

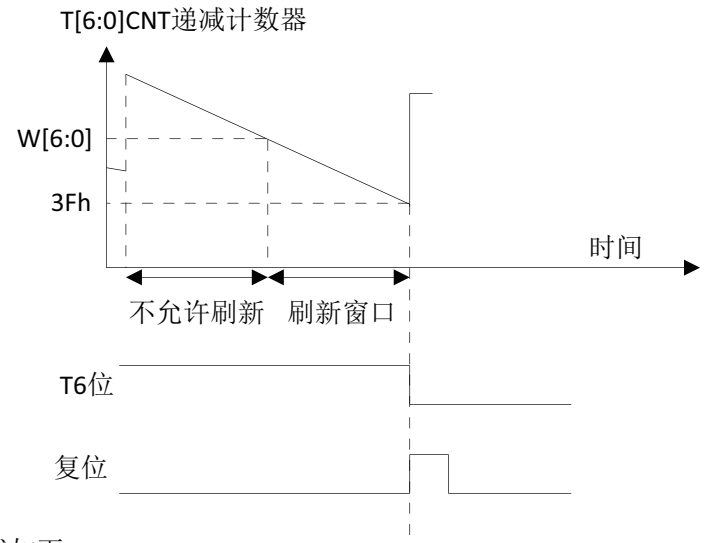
另一个重装载计数器的方法是利用早期唤醒中断 (EWI)。设置 WWDG_CFR 寄存器中的 WEI 位开启该中断。当递减计数器到达 0x40 时，则产生此中断，相应的中断服务程序 (ISR) 可以用来加载计数器以防止 WWDG 复位。在 WWDG_SR 寄存器中写‘0’可以清除该中断。

注：可以用 T6 位来产生一个软件复位 (WDGA 位被置位，T6 位清零)

15.4 如何编写看门狗超时程序

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

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



计算超时的公式如下：

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

其中：

T_{WWDG}：WWDG超时时间

T_{PCLK1}：APB1以mS为单位的时钟间隔

在PCLK1 = 36MHz时的最小-最大超时值

WDGTB	最小超时值	最大超时值
0	**113μS**	**7.28mS**
1	**227μS**	**14.56mS**
2	**455μS**	**29.12mS**
3	**910μS**	**58.25mS**

图 127. 窗口看门狗时序图

15.5 调试模式

当微控制器进入调试模式时 (CPU 核心停止)，根据调试模块中的 DBG_WWDG_STOP 配

置位的状态，WWDG 的计数器能够继续工作或停止。详见有关调试模块的章节。

15.6 WWDG 寄存器描述

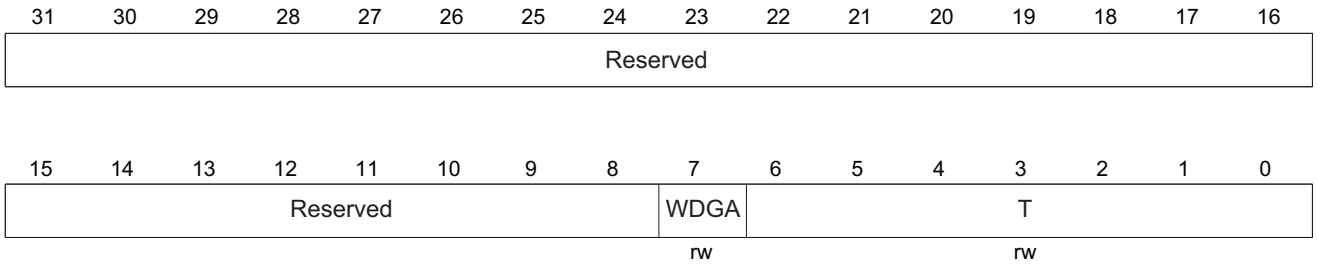
表 58. WWDG 寄存器概览

Offset	Acronym	Register Name	Reset	Section
0x00	WWDG_CR	控制寄存器	0x0000007F	小节 15.6.1
0x04	WWDG_CFGR	配置寄存器	0x0000007F	小节 15.6.2
0x08	WWDG_SR	状态寄存器	0x00000000	小节 15.6.3

15.6.1 控制寄存器 (WWDG_CR)

偏移地址：0x00

复位值：0x0000 007F

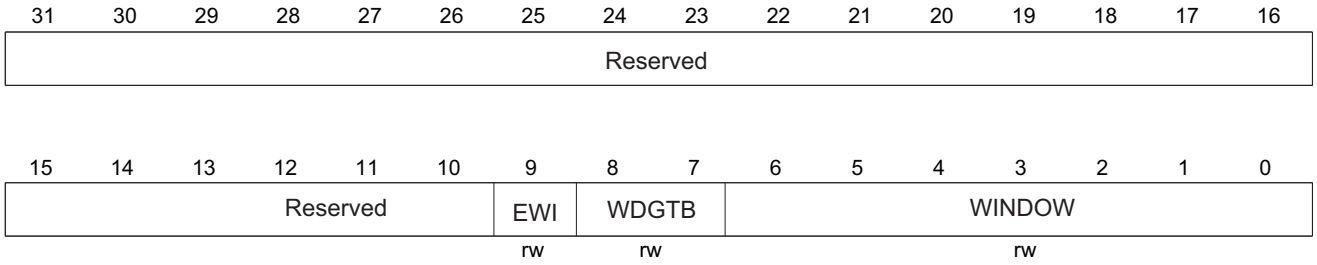


Bit	Field	Type	Reset	Description
31:8	Reserved			保留，始终读为 0
7	WDGA	rw	0x00	激活位 (Activation bit) 此位由软件置‘1’，但仅能由硬件在复位后清‘0’。当 WDGA = 1 时，看门狗可以产生复位。 0：禁止看门狗 1：启动看门狗
6:0	T	rw	0x7F	7 位计数器 (MSB 至 LSB)(7 - bit counter) 这些位用来存储看门狗的计数器值。每 ($4096x2^{WDGTB}$) 个 PCLK1 周期减 1。当计数器值从 40h 变为 3Fh 时 (T6 变成 0)，产生看门狗复位。

15.6.2 配置寄存器 (WWDG_CFGR)

偏移地址：0x04

复位值：0x0000 007F

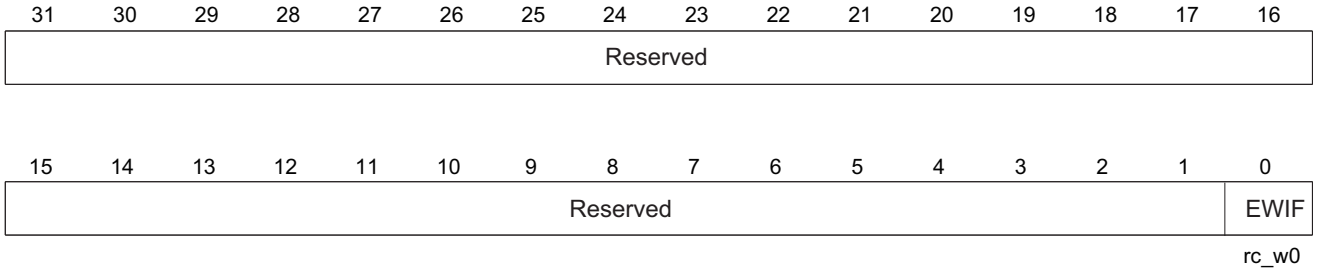


Bit	Field	Type	Reset	Description
31:10	Reserved			保留，始终读为 0
9	EWI	rw	0x00	提前唤醒中断 (Early wakeup interrupt) 此位若置‘1’，则当计数器值达到 40h，即产生中断。 此中断只能由硬件在复位后清除。
8:7	WDGTB	rw	0x00	时基 (Timer base) 预分频器的时基可根据如下修改： 00：CK 计时器时钟 (PCLK1 除以 4096) 除以 1 01：CK 计时器时钟 (PCLK1 除以 4096) 除以 2 10：CK 计时器时钟 (PCLK1 除以 4096) 除以 4 11：CK 计时器时钟 (PCLK1 除以 4096) 除以 8
6:0	WINDOW	rw	0x7F	7 位窗口值 (7-bit window value) 这些位包含了用来与递减计数器进行比较用的窗口值。

15.6.3 状态寄存器 (WWDG_SR)

偏移地址：0x08

复位值：0x0000 0000



Bit	Field	Type	Reset	Description
31:1	Reserved			保留，始终读为 0
0	EWIF	rc_w0	0x00	提前唤醒中断标志 (Early wakeup interrupt flag) 当计数器值达到 40h 时，此位由硬件置‘1’。它必须通过软件写‘0’来清除。 对此位写‘1’无效。若中断未被使能，此位也会被置‘1’。

16 USB 全速设备接口 (USB)

USB 全速设备接口 (USB)

16.1 USB 介绍

USB 外设实现了 USB2.0 全速总线和 APB1 总线间的接口。

USB 外设支持 USB 挂起/恢复操作，可以停止设备时钟实现低功耗。

16.2 USB 主要特征

- 符合 USB2.0 全速设备的技术规范
- 支持全速 12M 模式
- 一个控制传输端点和四个独立的通用端点用于中断传输和批量传输。
- 控制、批量以及中断传输最大可传输 64 字节的包。
- CRC（循环冗余校验）生成/校验，反向不归零（NRZI）编码/解码和位填充
- 支持 USB 挂起/恢复操作
- 支持 DMA 传输

下图是 USB 外设的方框图：

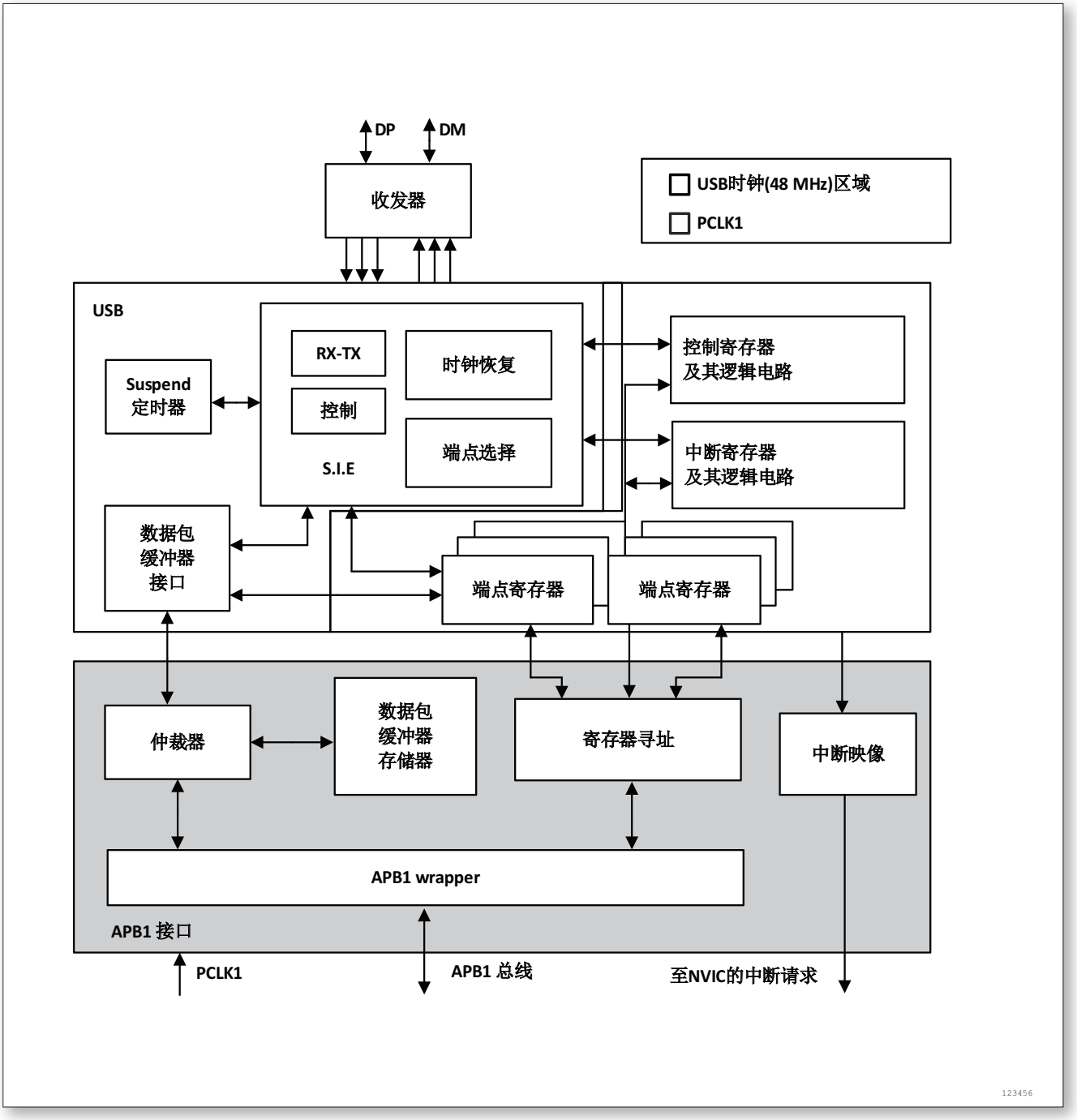


图 128. USB 方框图

16.3 USB 功能描述

USB 模块为 PC 主机和微控制器所实现的功能之间提供了符合 USB 规范的通信连接。PC 主机和微控制器之间的数据传输是通过数据缓冲区来完成的，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 64 字节缓冲区描述块，且该缓冲区是在 USB 模块内部，不能直接被 CPU 访问的。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：

- 主机请求的传输类型
- 哪个端点需要得到服务
- 产生如正在进行的是哪种类型的传输
- 端点的应答
- 传输是否完成

在任何不需要使用 USB 模块的时候，通过写 USB_POWER 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，不产生任何动态电流消耗，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，也可以通过软件设置直接唤醒 USB 系统，还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。

16.3.1 USB 功能模块描述

USB 模块包含 8 位宽的 320 字节大小的 FIFO，每个端点有 64 字节 FIFO。在 APB1 总线上，每个寄存器或者存储数据使用总线 32 位宽的低 8 位。

USB 模块包含一下几种寄存器：

- USB 寄存器。用于配置 USB 参数以及查询状态
- 端点状态寄存器
- 端点设置寄存器
- Setup 数据寄存器，存储 SETUP 包的数据。
- 端点数据控制寄存器，控制数据流。

APB1 总线或者 DMA 往数据端口写数据或者读数据，FIFO 数据个数就会增加或者递减。只有在端点接收和发送数据成功后，FIFO 指针才会变化。每个端点有一个 FIFO 的数据寄存器，作为写入或者读出 FIFO 的入口地址。每个端点还有专门的寄存器可以查询当前 FIFO 内有效数据个数。

只有端点 1 和端点 2 支持 DMA 传输。

16.4 编程中需要考虑的问题

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。

16.4.1 USB 传输概述

下面显示了包、事物以及传输三者之间的关系。

包（Packet）是 USB 系统中信息传输的基本单元，所有数据都是经过打包后在总线上传输的。

基本的 USB 包如下所示，如令牌包，数据包以及握手包。

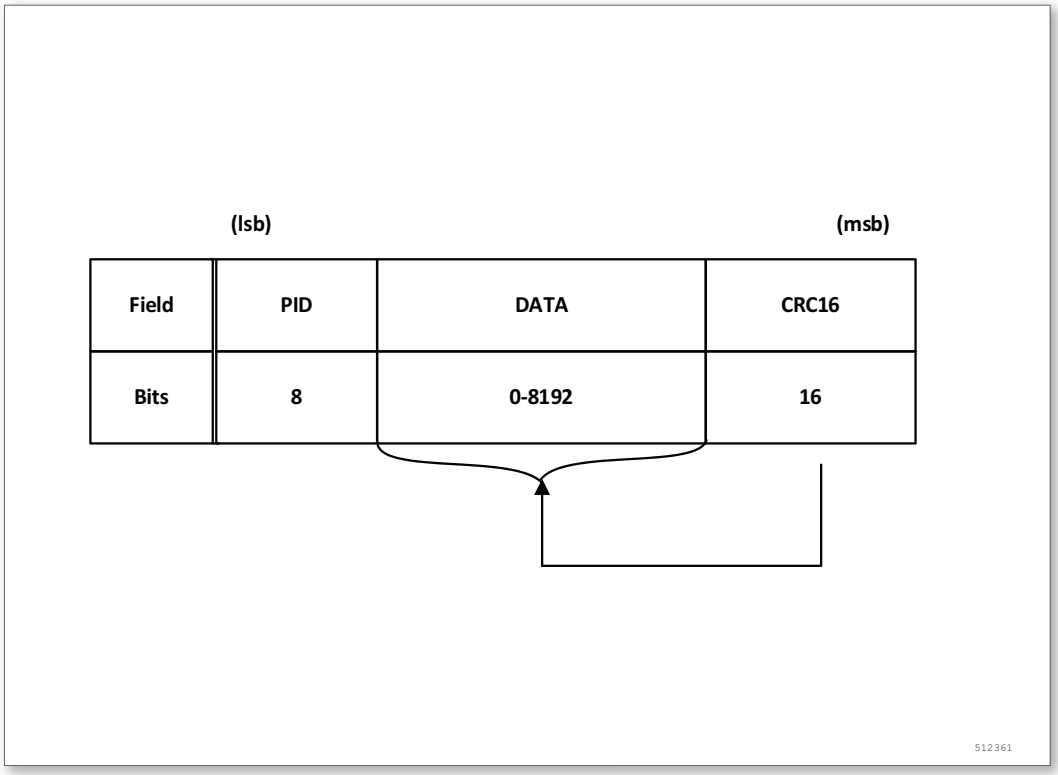


图 129. 包的基本格式

在 USB 上的数据信息的一次接收或发送的处理过程称为事务（Transaction）。事务处理的类型包括输入（IN）事务处理、输出（OUT）事务处理、设置（SETUP）事务处理等。

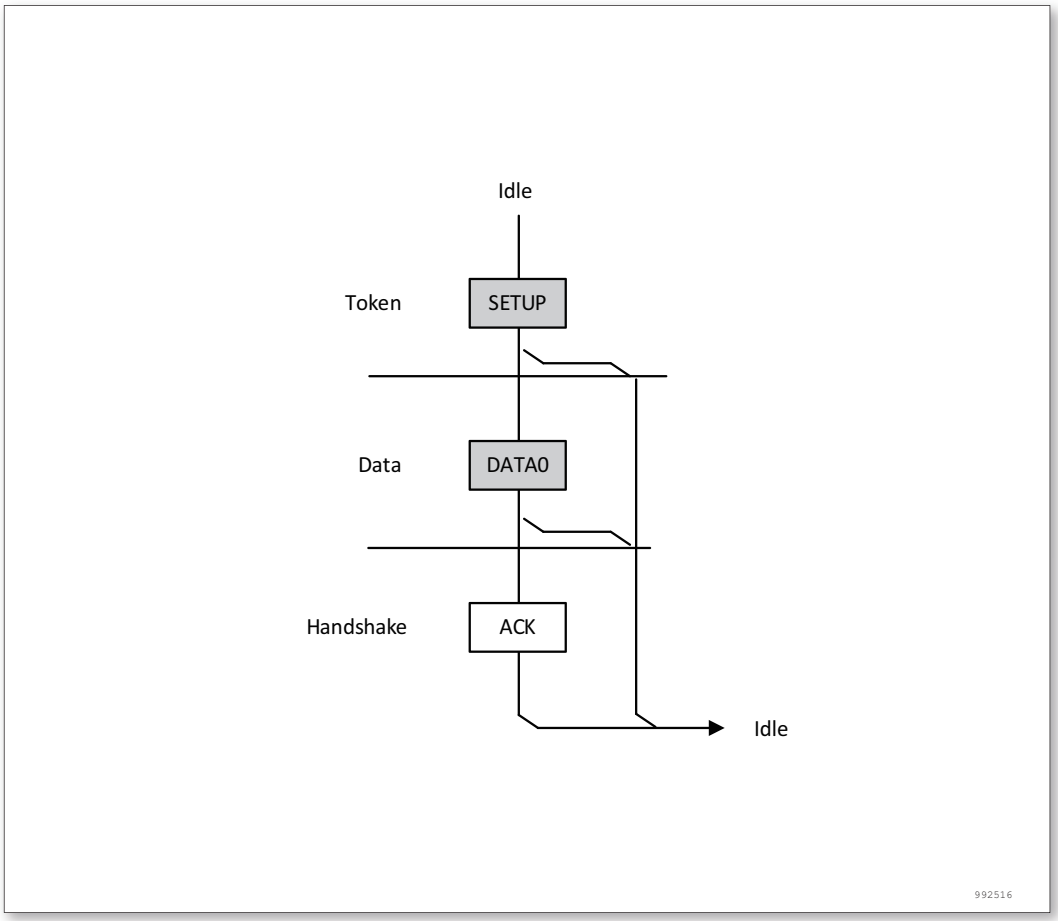


图 130. USB 事务

下图描述了一个端点的完整的传输过程。对于批量/控制传输，一个传输必须在上一次传输结束结束后开始。

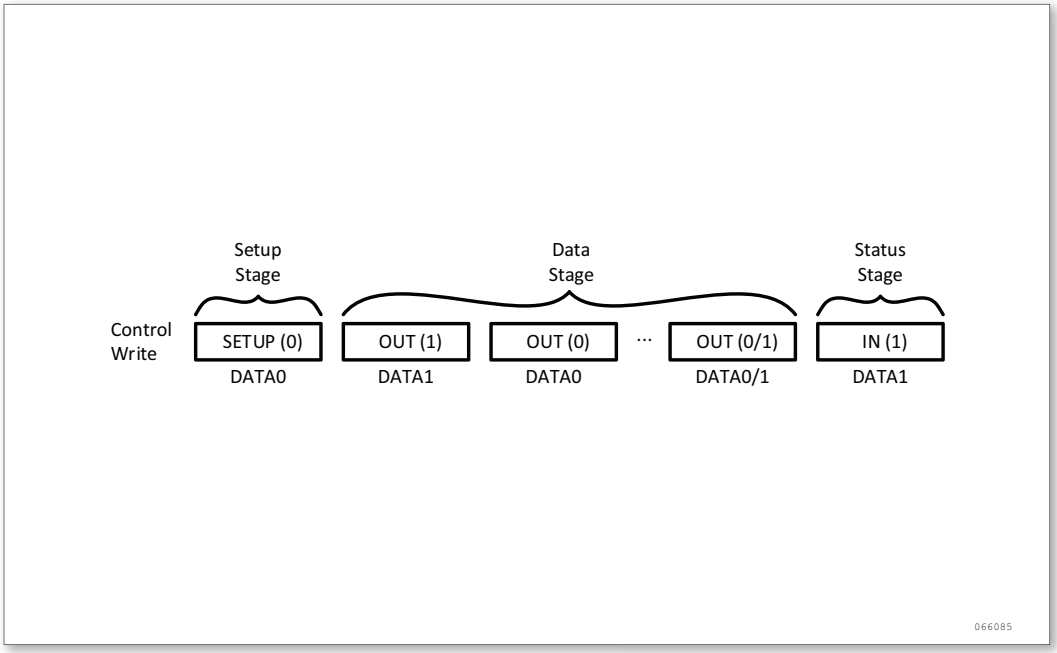


图 131. USB 传输

当 USB 控制器开始工作，USB 处于 IDLE 状态，然后等待总线命令。当接收到一个总线命令，如复位、设置等，就会产生一个中断，CPU 查询该命令类型。例如，如果命令是复位，CPU 就会清零和复位所有可编程的状态。如果是传输请求命令，CPU 就会写/读 USB 控制器 FIFO 中的数据。对于输入的批量传输或者控制传输，当 CPU 或者 DMA 准备好数据后，CPU 会发送一个 ACK 握手信号或者 STALL 握手信号来结束传输。对于批量传输输出，当数据放入到相应的 FIFO 后，USB 会自动发送 ACK 信号。

传输中当数据大小超过最大包的大小时，会将数据分割成超过一个包传输，否则只会传输一个包。

16.4.2 USB 枚举

主机以控制传输的方式，通过端点 0 对设备发送各种请求，设备收到主机发来的请求后恢复相应的信息，进行枚举操作。

系统上电复位后，首先配置 USB 控制器：

1. 设置端点使能位
2. 打开复位和端点中断
3. 打开连接位（USB_TOP 寄存器的 CONNECT 位）

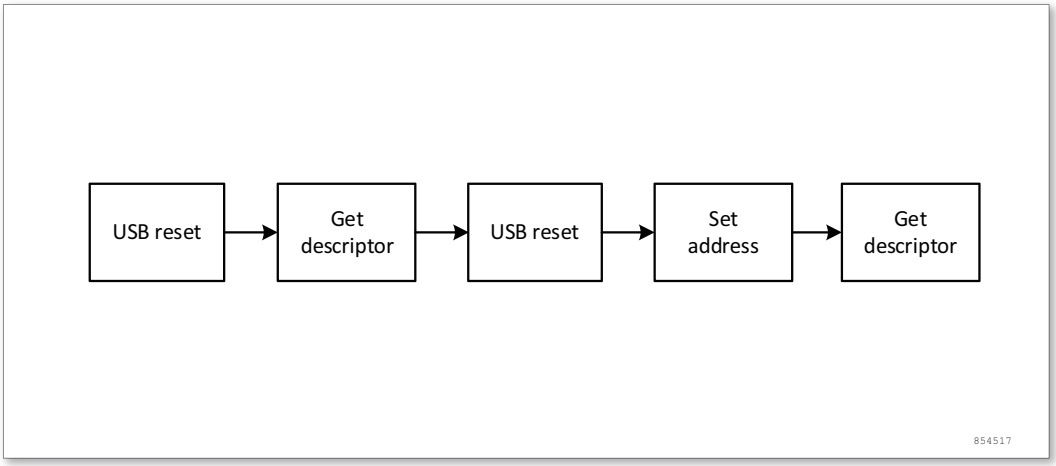


图 132. 枚举过程

当端点 0 接收到 SETUP 包，系统将会读取设置的 8 个字节数据从而决定下个步骤。对于 SetAddress 描述符，USB 控制器将会自动载入新地址。

枚举过程中，第一个来回的分析:

检测到设备，主机发总线复位。这个复位与 USB 上电复位和系统复位是不同的。这个是 SIE 根据总线状态通知用户的一种复位。设备产生复位中断，如何处理由设备固件程序决定。

主机发起第一个控制传输：

1. 主机 SETUP 包（发往地址 0 端点 0 ）、主机数据包（请求设备描述符）、设备握手包 ACK 。
   设备产生端点 0 数据输出中断，固件程序要根据数据包中的主机要求做好准备，这里是在端点 0 输入缓冲区准备好设备描述符。
2. 数据过程，主机先发一个 IN 令牌包、设备发一个数据包（这个数据已经准备好，SIE 收到 IN 令牌后，直接送到总线上，用户此时不干预）、主机发 ACK 包。

此时 SIE 产生端点 0 数据输入中断，表明主机已经取走了设备所准备的数据，用户也可以在该中断处理程序中作自己的处理。(SIE 指串行接口引擎，是所有 USB 控制器内部的‘核心’。SIE 负责处理底层协议，如填充位，CRC 生成和校验，并可发出错误报告。SIE 的主要任务是将低级信号转换成字节，以供控制器使用)

此时，主机只接受一次数据，最少 8 个字节。如果用户数据没有发完，又在控制端点输入缓冲区，准备了数据，主机也不理会。

3. 状态过程：主机发 OUT 包（通知设备要输出）、主机发 0 字节状态数据包（这个是 0 字节，表明自己收到设备描述符）、设备发握手 ACK 包。

   此时设备不会产生端点 0 数据输出中断，此时没有数据。

枚举过程中，第二个来回：设置地址。

第一个来回成功以后，主机再次复位总线。进入地址设置控制传输阶段。

1. 主机 SETUP 包（发往地址 0 端点 0 ）、主机数据包（请求设置地址）、设备握手包 ACK 。所以 SETUP 包后面都会跟一个表明主机 SETUP 目的的数据包，要么 GET ，要么 SET 。

   设备产生端点 0 数据输出中断，固件程序要根据数据包中的主机要求做好准备，这里是在根据主机发来的地址写入自己的地址控制寄存器。
2. 数据过程，本次传输没有数据。
3. 状态过程：主机发 IN 包（通知设备要返回数据）、设备发 0 字节状态数据包（表明地址设置已经成功）、主机发握手 ACK 包（地址设置已经生效）。

   此时设备不会产生端点 0 数据输入中断，此时没有数据。

枚举过程中，第三个来回：主机使用新地址获取完整的设备描述符。

主机采用新地址发起第一个控制传输：

1. 主机 SETUP 包（发往新的地址端点 0 ）、主机数据包（请求设备描述符）、设备握手包 ACK 。

   设备产生端点 0 数据输出中断，固件程序要根据数据包中的主机要求做好准备，这里是在端点 0 输入缓冲区准备好设备描述符。
2. 数据过程，主机先发一个 IN 令牌包、设备发一个数据包（这个数据已经准备好，SIE 收到 IN 令牌后，直接送到总线上，用户此时不干预）、主机发 ACK 包。

   此时 SIE 产生端点 0 数据输入中断，表明主机已经取走了设备所准备的数据，用户可以该中断处理程序中要做如下处理：如果一次没有将描述符送完，要再次将剩下的内容填充端点 0 输入缓冲区。

   第二次数据传输：主机再发一个 IN 令牌包、设备发一个数据包、主机发 ACK 包。

   此时 SIE 再次产生端点 0 数据输入中断，如果数据已经发完了。这里就不处理了。进入状态过程。
3. 状态过程：主机发 OUT 包（通知设备要输出）、主机发 0 字节状态数据包（表明自己收到设备描述符）、设备发握手 ACK 包。

16.4.3 USB 传输处理

系统需要设置一个无操作的循环来等待 USB 主机的请求。USB 主机的请求会设置一个中断位，系统通过不断查询中断位或者进入中断服务程序从而跳出循环。当系统检测到中断位时，跳到相应的子程序中处理。

下图是一个典型的 USB 传输的流程图：

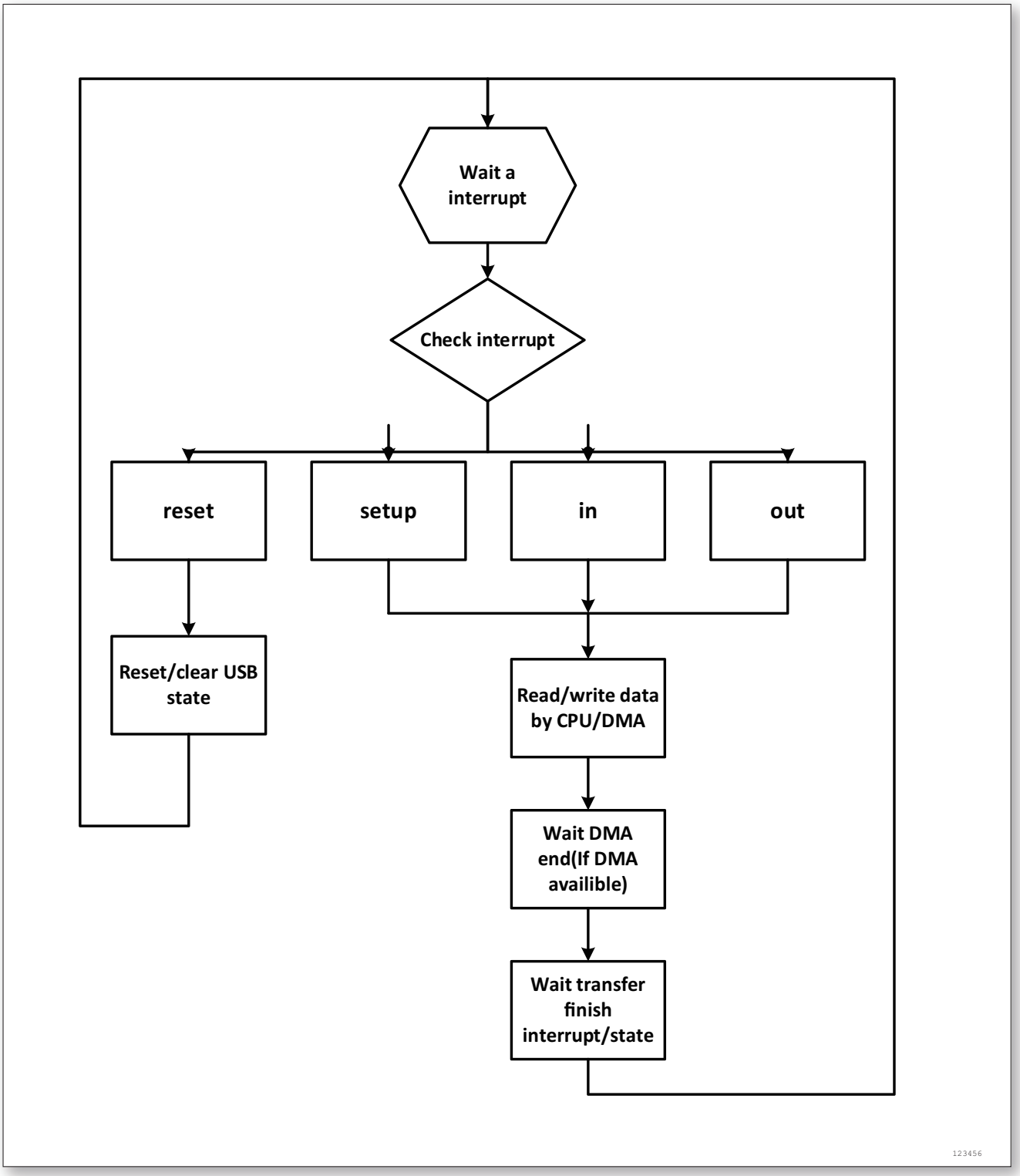


图 133. USB 传输流程图

16.4.4 IN 令牌包

当接收到一个主机发过来的 IN 请求时，如果收到 NACK 应答，USB 主机会重发 IN 请求直到该端点确认请求有效。如果接收到的地址和一个配置好的端点地址相符合的话，可以按照下面步骤：

1. 如果 FIFO 中的数据小于寄存器 EPX_CTRL[6:0] 中设置的大小，或者 EPX_CTRL[7]

未设为 1，USB 模块会自动发送 NACK，直到数据准备好。

2. CPU 收到 IN_NACK 状态。
3. CPU 把数据写入 FIFO。
4. CPU 在 EPX_CTRL 寄存器设置待发送的数据大小和发送使能。
5. USB 模块在收到下一个 IN 请求的时候自动发送 FIFO 中的数据发送。最后一个数据字节发送完成后，自动发送计算好的 CRC。
6. CPU 会收到 IN_ACK 状态，并且在发送结束后收到 END 状态。
7. 如果端点未使能，或者 EP_HALT 寄存器设置暂停，USB 模块会应答 IN_STALL 而不发送数据。

16.4.5 OUT 令牌包

当接收到一个主机发过来的 OUT 请求时，如果收到 NACK 应答，USB 主机会重发 OUT 请求直到该端点确认请求有效。如果接收到的地址和一个配置好的端点地址相符合的话，可以按照下面步骤：

1. 如果 FIFO 中的空间小于主机发过来的包的大小，USB 模块会自动发送 NACK，直到 CPU 把数据从 FIFO 中取走。
2. CPU 会收到 OUT_ACK 状态。
3. CPU 从 FIFO 读数据。
4. 如果端点未使能，或者 EP_HALT 寄存器设置暂停，USB 模块会应答 OUT_STALL 而不发送数据。

16.5 USB 寄存器描述

表 59. USB 寄存器概览

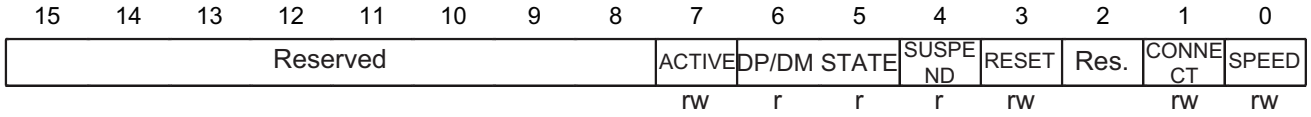
Offset	Acronym	Register Name	Reset	Section
0x00	USB_TOP	USB TOP 寄存器	0x00000002	小节 16.5.1
0x04	USB_INT_STATE	USB 中断状态寄存器	0x00000000	小节 16.5.2
0x08	EP_INT_STATE	USB 端点中断状态寄存器	0x00000000	小节 16.5.3
0x0C	EP0_INT_STATE	USB 端点 0 中断状态寄存器	0x00000000	小节 16.5.4
0x10	USB_INT_EN	USB 中断使能寄存器	0x00000000	小节 16.5.5
0x14	EP_INT_EN	USB 端点中断使能寄存器	0x00000000	小节 16.5.6
0x18	EP0_INT_EN	USB 端点 0 中断使能寄存器	0x00000000	小节 16.5.7
0x20~ 0x2C	EPX_INT_STATE	USB 端点 X 中断状态寄存器	0x00000000	小节 16.5.8
0x40~ 0x4C	EPX_INT_EN	USB 端点 X 中断使能寄存器	0x00000000	小节 16.5.9
0x60	USB_ADDR	USB 地址寄存器	0x00000000	小节 16.5.10
0x64	EP_EN	USB 端点使能寄存器	0x00000000	小节 16.5.11
0x78	TOG_CTRL1_4	USB 数据翻转控制寄存器	0x00000000	小节 16.5.12
0x80 ~ 0x9C	SETUPX	USB 传输包数据寄存器	0x00000000	小节 16.5.13
0xA0, 0xA4	PACKET_SIZEX	USB 传输包大小寄存器	0x00000040	小节 16.5.14
0x100 ~ 0x110	EPX_AVAIL	USB 端点 X 有效数据寄存器	0x00000000	小节 16.5.15
0x140 ~ 0x150	EPX_CTRL	USB 端点 X 控制寄存器	0x00000000	小节 16.5.16
0x160 ~ 0x170	EPX_FIFO	USB 端点 X FIFO 寄存器	0x00000000	小节 16.5.17

Offset	Acronym	Register Name	Reset	Section
0x184	EP_DMA	USB 端点 DMA 使能寄存器	0x00000000	小节 16.5.18
0x188	EP_HALT	USB 端点暂停寄存器	0x00000000	小节 16.5.19
0x1C0	USB_POWER	USB 功耗控制寄存器	0x00000003	小节 16.5.20

16.5.1 USB TOP 寄存器 (USB_TOP)

地址偏移：0x00

复位值：0x0000 0002



Bit	Field	Type	Reset	Description
15 : 8	Reserved			始终读为 0。
7	ACTIVE	rw	0x00	USB 总线活跃状态（USB bus is active） 0：USB 总线不活跃 1：USB 总线活跃
6 : 5	DP/DM STATE	r	0x00	USB DP/DM 线当前状态（Current USB DP/DM line state）
4	SUSPEND	r	0x00	USB suspend 状态（USB suspend state） 该位监控控制器状态与 APB_POWER 寄存器无关。 0：控制器处于工作状态 1：控制器处于挂起状态
3	RESET	rw	0x00	复位 USB 控制器的端点和 FIFO（Reset EP and FIFO in USB controller） 0：不复位 1：复位 注意，该位置位后需要软件清零。
2	Reserved			始终读为 0。
1	CONNECT	rw	0x01	USB 连接状态（USB connection） 0：断开连接 1：连接
0	SPEED	rw	0x00	设置 USB 速率 0：全速传输 1：低速传输

16.5.2 USB 中断状态寄存器 (USB_INT_STATE)

地址偏移：0x04

复位值：0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved											EP INTF	SOFF	RESUMF	SUS PENDF	RSTF
											r	rc_w1	rc_w1	rc_w1	rc_w1

Bit	Field	Type	Reset	Description
15 : 5	Reserved			始终读为 0。
4	EPINTF	r	0x00	端点中断标志位（EP interrupt received） 任何一个端点产生中断时，该位被置‘1’ 。具体参考 EP_INT_STATE 描述。该位只读。
3	SOFF	rc_w1	0x00	SOF 检测标志位（BUS received） 当 SOF 被检测到时，该位被置‘1’。写‘1’清除。
2	RESUMF	rc_w1	0x00	唤醒标志位（BUS resume received） 当 USB 总线被激活，该位被置‘1’。写‘1’清除。
1	SUSPENDF	rc_w1	0x00	USB 总线挂起标志位（BUS suspend received） 当检测到总线上挂起状态时，该位被置‘1’。写‘1’清除。
0	RSTF	rc_w1	0x00	USB 总线复位请求标志位（BUS reset received） 当总线的复位信号输入被检测到，该位被置‘1’。写‘1’清除。

16.5.3 USB 端点中断状态寄存器 (EP_INT_STATE)

地址偏移：0x08

复位值：0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved											EP4F	EP3F	EP2F	EP1F	EP0F
											r	r	r	r	r

Bit	Field	Type	Reset	Description
15 : 7	Reserved			始终读为 0。
4	EP4F	r	0x00	端点 4 中断标志位（EP4 interrupt received）
3	EP3F	r	0x00	端点 3 中断标志位（EP3 interrupt received）
2	EP2F	r	0x00	端点 2 中断标志位（EP2 interrupt received）
1	EP1F	r	0x00	端点 1 中断标志位（EP1 interrupt received）
0	EP0F	r	0x00	端点 0 中断标志位（EP0 interrupt received）

16.5.4 USB 端点 0 中断状态寄存器 (EP0_INT_STATE)

地址偏移：0x0C

复位值：0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								OUT-STALLF	OUT-ACKF	OUT-NACKF	IN-STALLF	IN-ACKF	IN-NACKF	END	SETUP
								rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

Bit	Field	Type	Reset	Description
15 : 8	Reserved			始终读为 0。
7	OUT-STALLF	rc_w1	0x00	OUT 包应答 STALL 标识（OUT-STALL received） 端点接收来自主机的 OUT 包后，控制器应答 STALL。在寄存器 EP_HALT[0] 设置为‘1’且 EP0_CTRL[7] 使能时，USB 控制器会应答 STALL。 写‘1’清除。
6	OUT-ACKF	rc_w1	0x00	OUT 包应答 ACK 标识（OUT-ACK received） 端点 0 接收来自主机的 OUT 包后，FIFO 有足够的空间，主机完成当前传输。USB 控制器自动应答 ACK。 写‘1’清除。
5	OUT-NACKF	rc_w1	0x00	OUT 包应答 NACK 标识（OUT-NACK received） 端点接收来自主机的 OUT 包后，但此时没有足够空间存储来自主机发送的数据。USB 控制器自动应答 NACK。 写‘1’清除。
4	IN-STALLF	rc_w1	0x00	IN 包应答 STALL 标识（IN-STALL received） 端点接收来自主机的 IN 包后，控制器应答 STALL。在寄存器 EP_HALT[0] 设置为‘1’且 EP0_CTRL[7] 使能时，USB 控制器会应答 STALL。 写‘1’清除。
3	IN-ACKF	rc_w1	0x00	IN 包应答 ACK 标识（IN-ACK received） 端点接收来自主机的 IN 包后，FIFO 中有足够的数据，主机完成当前传输。USB 控制器自动应答 ACK。 写‘1’清除。 端点接收来自主机的 IN 包后主机完成当前传输置位该位。
2	IN-NACKF	rc_w1	0x00	IN 包应答 NACK 标识（IN-NACK received） 端点接收来自主机的 IN 包后，但此时没有足够数据完成当前传输。USB 控制器自动应答 NACK。 写‘1’清除。
1	END	rc_w1	0x00	传输完成标识（Status stage finished） 端点传输完成时，该位被置‘1’。写‘1’清除。
0	SETUP	rc_w1	0x00	接收到 SETUP 包标识（SETUP packet received） 置位后可以从寄存器 SETUP0 ~ 7 读取 SETUP 包的内容。 写‘1’清除。

16.5.5 USB 中断使能寄存器 (USB_INT_EN)

地址偏移：0x10

复位值：0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								INT MASK	Reserved		EPIE	SOF IE	RESUM IE	SUSPE NDIE	RST IE
								rw			rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
15 : 8	Reserved			始终读为 0。
7	INTMASK	rw	0x00	中断屏蔽位 1，各中断寄存器中中断标志位受对应的中断使能寄存器中中断使能标志位的控制 0，各中断寄存器中中断标志位不受对应的中断使能寄存器中中断使能标志位的控制
6 : 5	Reserved			始终读为 0。
4	EPIE	rw	0x00	端点中断使能位（EP interrupt enable）
3	SOFIE	rw	0x00	SOF 检测中断使能位（SOF interrupt enable）
2	RESUMIE	rw	0x00	唤醒中断使能位（BUS resume interrupt enable）
1	SUSPENDIE	rw	0x00	USB 总线挂起中断使能位（BUS suspend interrupt enable）
0	RSTIE	rw	0x00	USB 总线复位中断使能位（BUS reset interrupt enable）

16.5.6 USB 端点中断使能寄存器 (EP_INT_EN)

地址偏移：0x14

复位值：0x0000 0000

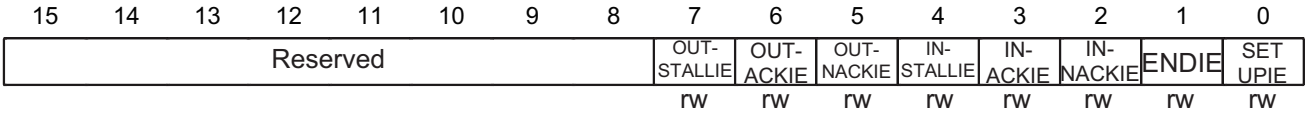
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved											EP4IE	EP3IE	EP2IE	EP1IE	EP0IE
											rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
15 : 5	Reserved			始终读为 0。
4	EP4IE	rw	0x00	端点 4 中断使能位（EP4 interrupt enable）
3	EP3IE	rw	0x00	端点 3 中断使能位（EP3 interrupt enable）
2	EP2IE	rw	0x00	端点 2 中断使能位（EP2 interrupt enable）
1	EP1IE	rw	0x00	端点 1 中断使能位（EP1 interrupt enable）
0	EP0IE	rw	0x00	端点 0 中断使能位（EP0 interrupt enable）

16.5.7 USB 端点 0 中断使能寄存器 (EP0_INT_EN)

地址偏移：0x18

复位值：0x0000 0000

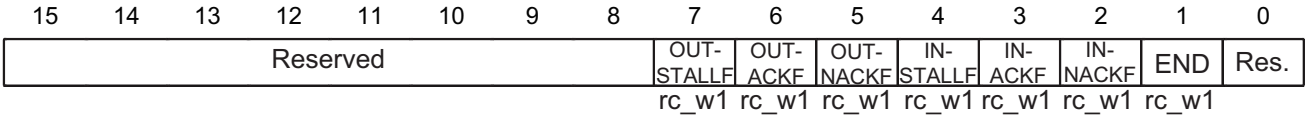


Bit	Field	Type	Reset	Description
15 : 8	Reserved			始终读为 0。
7	OUT-STALLIE	rw	0x00	OUT 包应答 STALL 中断使能位（OUT-STALL interrupt enable）
6	OUT-ACKIE	rw	0x00	OUT 包应答 ACK 中断使能位（OUT-ACK interrupt enable）
5	OUT-NACKIE	rw	0x00	OUT 包应答 NACK 中断使能位（OUT-NACK interrupt enable）
4	IN-STALLIE	rw	0x00	IN 包应答 STALL 中断使能位（IN-STALL interrupt enable）
3	IN-ACKIE	rw	0x00	IN 包应答 ACK 中断使能位（IN-ACK interrupt enable）
2	IN-NACKIE	rw	0x00	IN 包应答 NACK 中断使能位（IN-NACK interrupt enable）
1	ENDIE	rw	0x00	传输完成中断使能位（Status stage finished interrupt enable）
0	SETUPIE	rw	0x00	接收到 SETUP 包中断使能位（SETUP packet interrupt enable）

16.5.8 USB 端点 X 中断状态寄存器 (EPX_INT_STATE) (X = 1 ~ 4)

地址偏移：0x20 ~ 0x2C

复位值：0x0000 0000



Bit	Field	Type	Reset	Description
15 : 8	Reserved			始终读为 0。
7	OUT-STALLF	rc_w1	0x00	OUT 包应答 STALL 标识（OUT-STALL received） 端点接收来自主机的 OUT 包后，控制器应答 STALL。在寄存器 EP_HALT[x] 设置为‘1’且 EPx_CTRL[7] 使能时，USB 控制器会应答 STALL。 写‘1’清除。
6	OUT-ACKF	rc_w1	0x00	OUT 包应答 ACK 标识（OUT-ACK received） 端点 x 接收来自主机的 OUT 包后，FIFO 有足够的空间，主机完成当前传输。USB 控制器自动应答 ACK。 写‘1’清除。
5	OUT-NACKF	rc_w1	0x00	OUT 包应答 NACK 标识（OUT-NACK received） 端点接收来自主机的 OUT 包后，但此时没有足够空间存储来自主机发送的数据。USB 控制器自动应答 NACK。 写‘1’清除。

Bit	Field	Type	Reset	Description
4	IN-STALLF	rc_w1	0x00	IN 包应答 STALL 标识（IN-STALL received） 端点接收来自主机的 IN 包后，控制器应答 STALL。在寄存器 EP_HALT[x] 设置为‘1’且 EPx_CTRL[7] 使能时，USB 控制器会应答 STALL。 写‘1’清除。
3	IN-ACKF	rc_w1	0x00	IN 包应答 ACK 标识（IN-ACK received） 端点接收来自主机的 IN 包后，FIFO 中有足够的数据，主机完成当前传输。USB 控制器自动应答 ACK。 写‘1’清除。 端点接收来自主机的 IN 包后主机完成当前传输置位该位。
2	IN-NACKF	rc_w1	0x00	IN 包应答 NACK 标识（IN-NACK received） 端点接收来自主机的 IN 包后，但此时没有足够数据完成当前传输。USB 控制器自动应答 NACK。 写‘1’清除。
1	END	rc_w1	0x00	传输完成标识（Status stage finished） 端点传输完成时，该位被置‘1’。写‘1’清除。
0	Reserved			始终读为 0。

16.5.9 USB 端点 X 中断使能寄存器 (EPX_INT_EN) (X = 1 ~ 4)

地址偏移：0x40 ~ 0x4C

复位值：0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								OUT-STALLIE	OUT-ACKIE	OUT-NACKIE	IN-STALLIE	IN-ACKIE	IN-NACKIE	ENDIE	Res.
								rw	rw	rw	rw	rw	rw	rw	

Bit	Field	Type	Reset	Description
15 : 8	Reserved			始终读为 0。
7	OUT-STALLIE	rw	0x00	OUT 包应答 STALL 中断使能位（OUT-STALL interrupt enable）
6	OUT-ACKIE	rw	0x00	OUT 包应答 ACK 中断使能位（OUT-ACK interrupt enable）
5	OUT-NACKIE	rw	0x00	OUT 包应答 NACK 中断使能位（OUT-NACK interrupt enable）
4	IN-STALLIE	rw	0x00	IN 包应答 STALL 中断使能位（IN-STALL interrupt enable）
3	IN-ACKIE	rw	0x00	IN 包应答 ACK 中断使能位（IN-ACK interrupt enable）
2	IN-NACKIE	rw	0x00	IN 包应答 NACK 中断使能位（IN-NACK interrupt enable）
1	ENDIE	rw	0x00	传输完成中断使能位（Status stage finished interrupt enable）
0	Reserved			始终读为 0。

16.5.10 USB 地址寄存器 (USB_ADDR)

地址偏移：0x60

复位值：0x0000 0000

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

Bit	Field	Type	Reset	Description
15 : 7	Reserved			始终读为 0。
6 :0	ADDR	rw	0x00	USB 地址（USB address） 接收到主机发送的设置地址描述符时，硬件自动将地址装载至该寄存器中。 当接收到总线复位时硬件自动清除该寄存器的值。

16.5.11 USB 端点使能寄存器 (EP_EN)

地址偏移：0x64

复位值：0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved											EP4EN	EP3EN	EP2EN	EP1EN	EP0EN
											rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
15 : 5	Reserved			始终读为 0。
4	EP4EN	rw	0x00	使能端点 4（Enable End Point 4）
3	EP3EN	rw	0x00	使能端点 3（Enable End Point 3）
2	EP2EN	rw	0x00	使能端点 2（Enable End Point 2）
1	EP1EN	rw	0x00	使能端点 1（Enable End Point 1）
0	EP0EN	rw	0x00	使能端点 0（Enable End Point 0）

16.5.12 USB 数据翻转控制寄存器 (TOG_CTRL1_4)

地址偏移：0x78

复位值：0x0000 0000

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

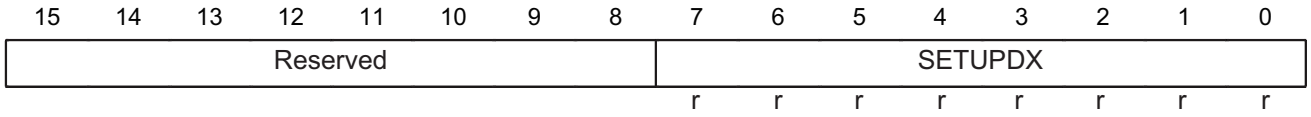
Bit	Field	Type	Reset	Description
15 : 8	Reserved			始终读为 0。

Bit	Field	Type	Reset	Description
7	DTOG4EN	w	0x00	端点 4 数据翻转使能位（Set End Point 4 enable） 0：不改变端点 4 的数据翻转位 1：将位 6 DTOG4 设置为端点 4 的数据翻转位
6	DTOG4	rw	0x00	端点 4 数据翻转位（Set End Point 4 Toggle） 0：DATA0 1：DATA1
5	DTOG3EN	w	0x00	端点 3 数据翻转使能位（Set End Point 3 enable） 0：不改变端点 3 的数据翻转位 1：将位 4 DTOG3 设置为端点 3 的数据翻转位
4	DTOG3	rw	0x00	端点 3 数据翻转位（Set End Point 3 Toggle） 0：DATA0 1：DATA1
3	DTOG2EN	w	0x00	端点 2 数据翻转使能位（Set End Point 2 enable） 0：不改变端点 2 的数据翻转位 1：将位 2 DTOG2 设置为端点 2 的数据翻转位
2	DTOG2	rw	0x00	端点 2 数据翻转位（Set End Point 2 Toggle） 0：DATA0 1：DATA1
1	DTOG1EN	w	0x00	端点 1 数据翻转使能位（Set End Point 1 enable） 0：不改变端点 1 的数据翻转位 1：将位 0 DTOG1 设置为端点 1 的数据翻转位
0	DTOG1	rw	0x00	端点 1 数据翻转位（Set End Point 1 Toggle） 0：DATA0 1：DATA1

16.5.13 USB 设置包数据寄存器 (SETUPX) (X = 0 ~ 7)

地址偏移：0x80 ~ 0x9C

复位值：0x0000 0000



Bit	Field	Type	Reset	Description
15 : 8	Reserved			始终读为 0。
7 : 0	SETUPDX	r	0x00	USB 设置包数据位（x = 0,1,2,..,7）（Setup Data X） 64 位 SETUP 数据，由硬件根据主机发送的数据自动设置。

16.5.14 USB 传输包大小寄存器 (PACKET_SIZEX)(X = 0 ~ 1)

地址偏移：0xA0, 0xA4

复位值：0x0000 0040

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

Bit	Field	Type	Reset	Description
15 : 8	Reserved			始终读为 0。
7 : 0	SIZE	rw	0x40	USB 最大传输包大小（USB DMA Max Packet Size） 最大可设置 64 字节。

16.5.15 USB 端点 X 有效数据寄存器 (EPX_AVAIL)

地址偏移：0x100 ~ 0x110

复位值：0x0000 0000

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

Bit	Field	Type	Reset	Description
15 : 8	Reserved			始终读为 0。
7 : 0	EPXAVIL	r	0x00	USB 端点 X FIFO 有效数据个数（EPX FIFO available data number）

16.5.16 USB 端点 X 控制寄存器 (EPX_CTRL)

地址偏移：0x140 ~ 0x150

复位值：0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								TRAN EN	TRANCOUNT						
								rw	rw	rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
15 : 8	Reserved			始终读为 0。
7	TRANEN	rw	0x00	USB 端点 X 传输使能位（EPX transfer enable） 如果该位设置为‘1’，端点 X 会在 IN 传输的后面应答位 6:0 中定义的数据个数，否则端点 X 应答 NACK。 如果端点 x FIFO 中没有足够的数据，端点 X 同样会应答 NACK。 如果端点 X HALT 使能位设置为‘1’，端点 X 就会自动应答 STALL。 传输结束该位会自动变为‘0’。

Bit	Field	Type	Reset	Description
6 : 0	TRANCOUNT	rw	0x00	端点 X 传输数量（EPX transfer counter） 端点 X 要传输的数据个数。数据保存在个端点的 FIFO 中，最大传输数量不能超过寄存器 PACKAGE_SIZE 定义的最大包的大小，最后一个包的传输数量可能小于最大包，甚至可以为零，意味着传输一个空包。

16.5.17 USB 端点 X FIFO 寄存器 (EPX_FIFO)

地址偏移：0x160 ~ 0x170

复位值：0x0000 0000

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

Bit	Field	Type	Reset	Description
15 : 8	Reserved			始终读为 0。
7 : 0	EPX_FIFO	rw	0x00	端点 X FIFO 数据端口（EPX FIFO port）

16.5.18 USB 端点 DMA 使能寄存器 (EP_DMA)

地址偏移：0x184

复位值：0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														DMA2EN	DMA1EN
														rw	rw

Bit	Field	Type	Reset	Description
15 : 2	Reserved			始终读为 0。
1	DMA2EN	rw	0x00	端点 2 DMA 使能位（EP2 DMA enable）
0	DMA1EN	rw	0x00	端点 1 DMA 使能位（EP1 DMA enable）

注：USB 控制器只支持端点 1 和端点 2 的 DMA 操作。

16.5.19 USB 端点暂停寄存器 (EP_HALT)

地址偏移：0x188

复位值：0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved											HALT4	HALT3	HALT2	HALT1	HALT0
											rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
15 : 5	Reserved			始终读为 0。
4	HALT4	rw	0x00	端点 4 暂停位（EP4 halt） 当该位设为‘1’，设备会在 IN/OUT 传输后自动响应 STALL。 当接收到令牌包时该位会被硬件自动清零。
3	HALT3	rw	0x00	端点 3 暂停位（EP3 halt） 当该位设为‘1’，设备会在 IN/OUT 传输后自动响应 STALL。 当接收到令牌包时该位会被硬件自动清零。
2	HALT2	rw	0x00	端点 2 暂停位（EP2 halt） 当该位设为‘1’，设备会在 IN/OUT 传输后自动响应 STALL。 当接收到令牌包时该位会被硬件自动清零。
1	HALT1	rw	0x00	端点 1 暂停位（EP1 halt） 当该位设为‘1’，设备会在 IN/OUT 传输后自动响应 STALL。 当接收到令牌包时该位会被硬件自动清零。
0	HALT0	rw	0x00	端点 0 暂停位（EP0 halt） 当该位设为‘1’，设备会在 IN/OUT 传输后自动响应 STALL。 当接收到令牌包时该位会被硬件自动清零。

16.5.20 USB 功耗控制寄存器 (USB_POWER)

地址偏移：0x1C0

复位值：0x0000 0003

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												WKUP	Res.	SUSP	SUSP EN
												rw		rw	rw

Bit	Field	Type	Reset	Description
15 : 4	Reserved			始终读为 0。
3	WKUP	rw	0x00	控制器从挂起状态唤醒 (Enable controller wake up from suspend state) 1：唤醒 0：不唤醒
2	Reserved			始终读为 0。
1	SUSP	rw	0x01	挂起位 (suspend) 1：正常工作模式 0：挂起模式
0	SUSPEN	rw	0x01	总线挂起使能位 (BUS suspend enable bit) 1：控制器根据位 1 的状态直接控制 USB 是否挂起 0：由控制器控制是否挂起信号

17 控制器局域网 (CAN)

控制器局域网 (CAN)

17.1 CAN 简介

它的设计目标是，以最小的 CPU 负荷来高效处理大量收到的报文。它也支持报文发送的优先级要求 (优先级特性可软件配置)。

对于安全紧要的应用，CAN 提供所有支持时间触发通信模式所需的硬件功能。

17.2 CAN 主要特点

- 支持 CAN 协议的 2.0 A 和 2.0 B
- 扩展的接收缓冲器 (64 字节、先进先出 FIFO)
- 同时支持 11 位和 29 位识别码
- 位速率可达 1Mbits/s
- PeliCAN 模式扩展功能
  - 可读/写访问的错误计数器
  - 可编程的错误报警限制
  - 最近一次错误代码寄存器
  - 对每一个 CAN 总线错误的中断
  - 具体控制位控制的仲裁丢失中断
  - 单次发送 (无重发)
  - 只听模式 (无确认、无活动的出错标志)
  - 软件位速率检测
  - 验收滤波器扩展 (4 字节代码，4 字节屏蔽)
  - 自身信息接收 (自接收请求)

17.3 CAN 控制器的总体描述

在当今的 CAN 应用中，CAN 网络的节点在不断增加，并且多个 CAN 常常通过网关连接起来，因此整个 CAN 网中的报文数量 (每个节点都需要处理) 急剧增加。除了应用层报文外，网路管理和诊断报文也被引入。

需要一个增强的过滤机制来处理各种类型的报文。

此外应用层任务需要更多 CPU 时间，因此报文接收所需的实时响应程度需要减轻。

接收 FIFO 的方案允许，CPU 花很长时间处理应用层任务而不会丢失报文。

构筑在底层 CAN 驱动程序上的高层协议软件，要求跟 CAN 控制器之间的高效的接口。

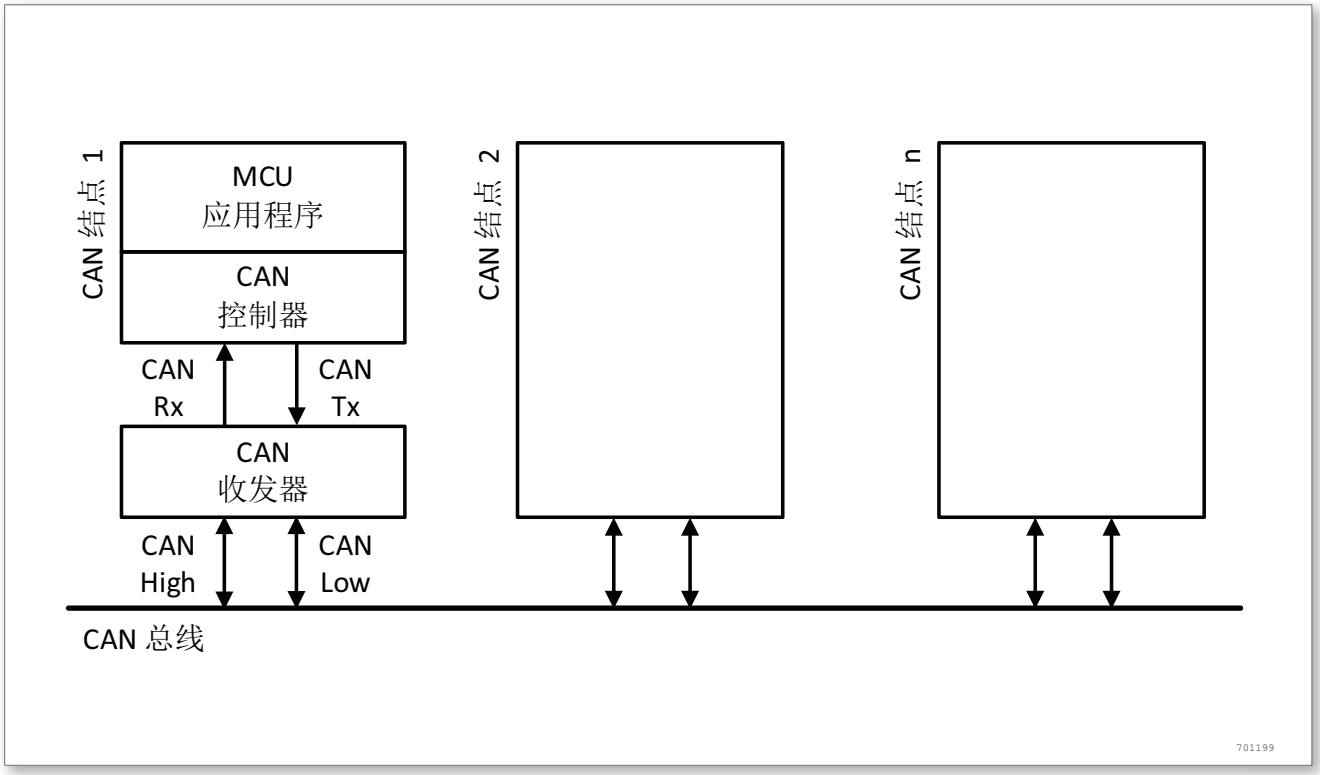


图 134. CAN 网络拓扑结构

17.3.1 CAN 2.0B 主动内核

CAN 模块可以完全自动地接收和发送 CAN 报文；且完全支持标准标识符 (11 位) 和扩展标识符 (29 位)。

17.3.2 CAN 方框图

CAN 结构框图如下:

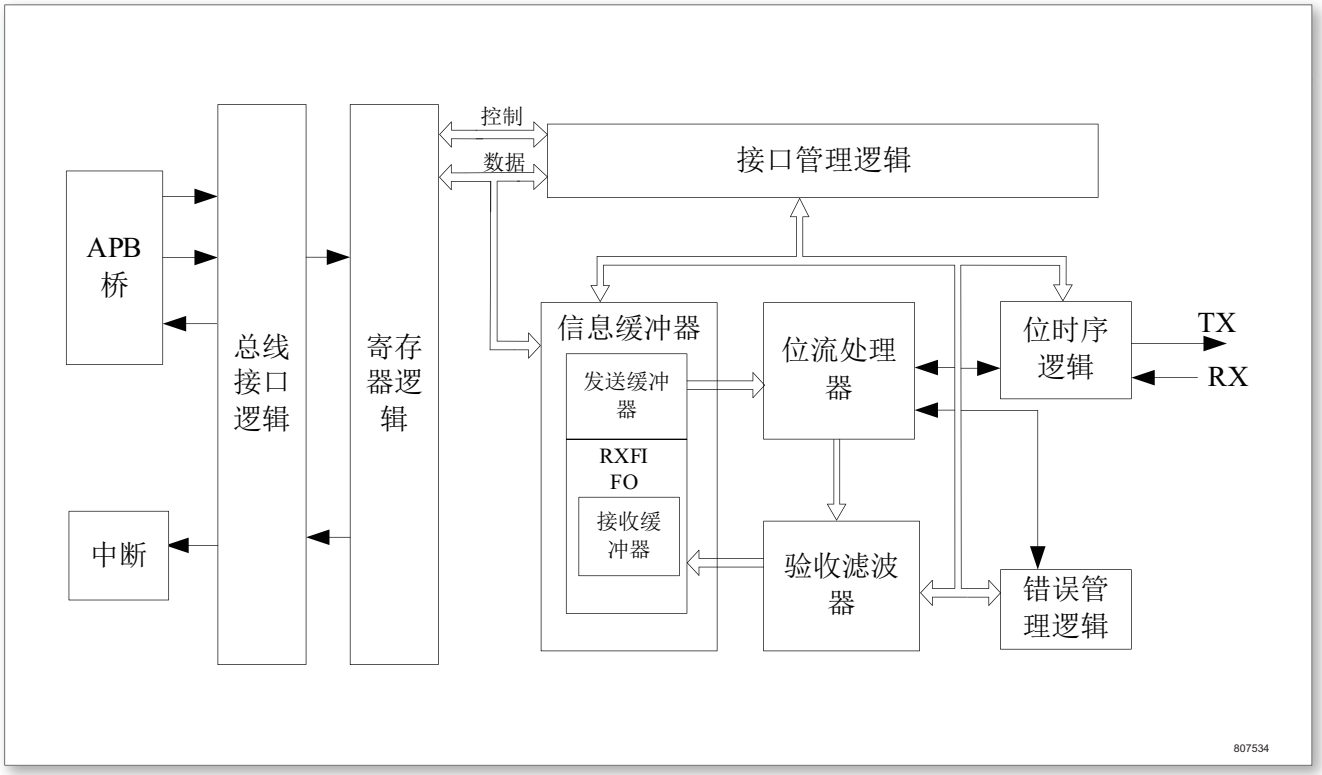


图 135. CAN 结构方框图

17.3.3 接口管理逻辑 (IML)

接口管理逻辑解释来自 CPU 的命令，控制 CAN 寄存器的寻址向主控制器提供中断信息和状态信息。

17.3.4 发送缓冲器 (TXB)

发送缓冲器是 CPU 和 BSP(位流处理器) 之间的接口，能够存储发送到 CAN 网络上的完整信息。缓冲器长 13 个字节，由 CPU 写入、BSP 读出。

17.3.5 接收缓冲器 (RXB,RXFIFO)

接收缓冲器是验收滤波器和 CPU 之间的接口用来储存从 CAN 总线上接收和接收的信息接收缓冲器 (RXB，13 个字节) 作为接收 FIFO(RXFIFO，长 64 字节) 的一个窗口，可被 CPU 访问。

CPU 在此 FIFO 的支持下可以在处理信息的时候接收其它信息。

17.3.6 验收滤波器 (ACF)

验收滤波器把它其中的数据和接收的识别码的内容相比较，以决定是否接收信息。在纯粹的接收测试中，所有的信息都保存在 RXFIFO 中。

17.3.7 位流处理器 (BSP)

位流处理器是一个在发送缓冲器、RXFIFO 和 CAN 总线之间控制数据流的程序装置。它还在 CAN 总线上执行错误检测、仲裁、填充和错误处理。

17.3.8 位时序逻辑 (BTL)

位时序逻辑监视串口的 CAN 总线和处理与总线有关的位时序。它在信息开头‘弱势-支配’的总线传输时同步 CAN 总线位流 (硬同步)，接收信息时再次同步下一次传送 (软同步)。BTL 还提供了可编程的时间段来补偿传播延迟时间、相位转换和定义采样点和一位时间内的采样次数。

17.3.9 错位管理逻辑 (EML)

EML 负责传送层模块的错误管制。它接收 BSP 的出错报告，通知 BSP 和 IML 进行错误统计。

17.4 CAN 工作模式

CAN 控制器有 2 个主要的工作模式：

- BasicCAN 模式
- PeliCAN 模式

系统复位时默认模式是 BasicCAN 模式。

PeliCAN 模式是新的操作模式，它能处理所有的 CAN 2.0B 规范的帧类型。而且它还提供一些增强功能使得应用于更宽的领域。

寄存器 CAN_CDR.7 定义了 CAN 模式。如果 CDR.7 是 0，CAN 控制器工作于 BasicCAN 模式。否则，CAN 控制器工作于 PeliCAN 模式。

17.4.1 Basic CAN 和 PeliCAN 模式的区别

在 Peli CAN 模式下，CAN 控制器有一个含很多功能的重组寄存器。Peli CAN 模式支持 CAN 2.0B 协议规定的所有功能 (29 字节的识别码)。下面是 PeliCAN 模式的主要新功能：

- 标准帧和扩展帧的接收和传送
- 接收 FIFO(64 字节)
- 在标准和扩展格式中都有单/双验收滤波器 (含屏蔽和代码寄存器)
- 读/写访问的错误计数器
- 可编程的错误限制报警
- 最近一次的误码寄存器
- 对每一个 CAN 总线错误的错误中断
- 仲裁丢失以及详细的位位置
- 一次性发送 (当错误或仲裁丢失时不重发)
- 只听模式 (CAN 总线监听，无应答，无错误标志)

17.5 CAN 功能描述

17.5.1 Basic CAN 模式

复位模式

复位模式即初始化模式。在硬件启动或总线状态设置为‘1’(总线关闭) 时，复位请求位 (CAN_CR.0) 被置为‘1’(当前)。如果这些位被软件访问，其值将发生变化，而且会影响内部时钟的下个上升沿。复位请求位的变化时内部分频时钟同步的读复位请求位能够反映

出这种同步状态。复位模式主要用于 CAN 通讯参数配置，在不同工作模式下内核对 CAN 寄存器的访问权限不同。

复位请求位被设为‘0’后 CAN 控制器将会等待：

a) 一个总线空闲信号 (11 个弱势位)，如果前一次复位请求是硬件复位或 CPU 初始复位。
b) 128 个总线空闲，如果前一次复位请求是 CAN 控制器在重新进入总线开启模式前初始化总线造成的；必须说明的是，如果复位请求位被置位，一些寄存器的值会被改变的。

工作模式

在复位模式完成后，软件应该让硬件进入正常模式，以便正常接收和发送报文。复位模式下，当向复位位传送了‘1 - 0’的下降沿时，CAN 控制器便返回工作模式，进行报文的发送和接收。

表 60. Basic CAN 模式寄存器权限分配表

Offset	段	工作模式		复位模式	
		读	写	读	写
00	控制	控制	控制	控制	控制
04		(FFH)	命令	(FFH)	命令
08		状态	–	状态	–
0C		(FFH)	–	中断	–
10		(FFH)	–	验收代码	验收代码
14		(FFH)	–	验收屏蔽	验收屏蔽
18		(FFH)	–	总线定时 0	总线定时 0
1C		(FFH)	–	总线定时 1	总线定时 1
20		(FFH)	–	–	–
24		测试	测试	测试	测试
28	发送缓冲器	识别码 (10 ~ 3)	识别码 (10 ~ 3)	(0xFF)	–
2C		识别码 (2~0) RTR 和 DLC	识别码 (2~0) RTR 和 DLC	(0xFF)	–
30		DATA1	DATA1	(0xFF)	–
34		DATA2	DATA2	(0xFF)	–
38		DATA3	DATA3	(0xFF)	–
3C		DATA4	DATA4	(0xFF)	–
40		DATA5	DATA5	(0xFF)	–
44		DATA6	DATA6	(0xFF)	–
48		DATA7	DATA7	(0xFF)	–
4C		DATA8	DATA8	(0xFF)	–
50	接收缓冲器	识别码 (10 ~ 3)	识别码 (10 ~ 3)	识别码 (10 ~ 3)	识别码 (10 ~ 3)
54		识别码 (2 ~ 0) RTR 和 DLC	识别码 (2 ~ 0) RTR 和 DLC	识别码 (2 ~ 0) RTR 和 DLC	识别码 (2 ~ 0) RTR 和 DLC
58		DATA1	DATA1	DATA1	DATA1

Offset	段	工作模式		复位模式	
		读	写	读	写
5C	接收缓冲器	DATA2	DATA2	DATA2	DATA2
60		DATA3	DATA3	DATA3	DATA3
64		DATA4	DATA4	DATA4	DATA4
68		DATA5	DATA5	DATA5	DATA5
6C		DATA6	DATA6	DATA6	DATA6
70		DATA7	DATA7	DATA7	DATA7
74		DATA8	DATA8	DATA8	DATA8
78		(FFH)	–	(FFH)	–
7C		时钟分频器	时钟分频器	时钟分频器	时钟分频器

注：‘(FFH)’代表读出数据全为 1，‘—’代表无写操作权限，其余表示可操作。偏移地址为 0x7C ‘时钟分频器’用于选择 BasicCAN 和 PeliCAN。

17.5.2 Peli CAN 模式

复位模式

复位模式即初始化模式。在硬件复位或总线状态位为‘1’(总线关闭) 时复位模式位被置为‘1’(当前)。如果通过软件访问这一位，值将发生变化且下一个内部时钟 (频率为外部振荡器的 1/2) 的上升沿有效。复位请求位的改变和内部分频时钟同步。读复位请求位能够反映出这种同步状态。复位模式位为‘0’后，CAN 控制器会等待：

1. 一个总线空闲信号 (11 个隐藏 (弱势) 位)，如果上一次复位是硬件复位或 CPU 初始复位。
2. 128 个总线空闲，如果上一次复位是 CAN 控制器在重新进入总线开启之前初始化复位。

工作模式

在复位模式完成后，软件应该让硬件进入正常模式，以便正常接收和发送报文。复位模式下，当检测到 CAN_MOD 寄存器的 RM 位出现了‘1 - 0’的下降沿，CAN 控制器便返回工作模式，进行报文的发送和接收。

自检测模式

此模式主要用于测试。设置自检测模式位 (CAN_MOD.2) 为 1，进入自检测模式。此模式可以检测所有节点，没有任何活动的节点使用自接收命令；即使没有应答，CAN 控制器也会成功发送。

只听模式

此主要用于测试。设置只听模式位 (CAN_MOD.1) 为 1 进入只听模式。这种工作模式使 CAN 控制器进入错误消极状态。信息传送是不可能的。以软件驱动的位速检测可使用只听模式。所有其它功能都能象在正常工作模式中一样使用。

在该模式中，CAN 控制器不能在 CAN 总线上写显性位。激活错误标志或超载标志不能都写，成功接收后的应答信号也不会给出。

注：在进入只听模式之前，必须进入复位模式。

表 61. Peli CAN 模式寄存器权限分配表

Offset	工作模式				复位模式	
	读		写		读	写
00	模式		模式		模式	模式
04	(00H)		命令		(00H)	命令
08	状态		–		状态	–
0C	中断		–		中断	–
10	中断使能		–		中断使能	中断使能
14	(00H)		–		(00H)	–
18	总线定时 0		–		总线定时 0	总线定时 0
1C	总线定时 1		–		总线定时 1	总线定时 1
20	保留		–		–	–
24	检测		检测		检测	检测
28	保留		–		保留	–
2C	仲裁丢失捕捉		–		仲裁丢失捕获	–
30	错误代码捕捉		–		错误代码捕捉	–
34	错误报警限制		–		错误报警限制	错误报警限制
38	RX 错误计数器		–		RX 错误计数器	RX 错误计数器
3C	TX 错误计数器		–		TX 错误计数器	TX 错误计数器
40	RX 帧信息 SFF	RX 帧错误 EFF	TX 帧错误 SFF	TX 帧错误 EFF	验收代码 0	验收代码 0
44	RX 识别码 1	RX 识别码 1	TX 识别码 1	TX 识别码 1	验收代码 1	验收代码 1
48	RX 识别码 2	RX 识别码 2	TX 识别码 2	TX 识别码 2	验收代码 2	验收代码 2
4C	RX 数据 1	RX 识别码 3	TX 数据 1	TX 识别码 3	验收代码 3	验收代码 3
50	RX 数据 2	RX 识别码 4	TX 数据 2	TX 识别码 4	验收屏蔽 0	验收屏蔽 0
54	RX 数据 3	RX 数据 1	TX 数据 3	TX 数据 1	验收屏蔽 1	验收屏蔽 1
58	RX 数据 4	RX 数据 2	TX 数据 4	TX 数据 2	验收屏蔽 2	验收屏蔽 2
5C	RX 数据 5	RX 数据 3	TX 数据 5	TX 数据 3	验收屏蔽 3	验收屏蔽 3
60	RX 数据 6	RX 数据 4	TX 数据 6	TX 数据 4	保留	–
64	RX 数据 7	RX 数据 5	TX 数据 7	TX 数据 5	保留	–
68	RX 数据 8	RX 数据 6	TX 数据 8	TX 数据 6	保留	–

Offset	工作模式				复位模式	
	读		写		读	写
6C	(FIFO RAM)	RX 数据 7	–	TX 数据 7	保留	–
70	(FIFO RAM)	RX 数据 8	–	TX 数据 8	保留	–
74	RX 信息计数器		–		RX 信息计数器	–
78	RX 缓冲器起始地址		–		RX 缓冲器起始地址	RX 缓冲器起始地址
7C	时钟分频器		时钟分频器		时钟分频器	时钟分频器
80	内部 RAM 地址 0(FIFO)		–		内部 RAM 地址 0	内部 RAM 地址 0
84	内部 RAM 地址 1(FIFO)		–		内部 RAM 地址 1	内部 RAM 地址 1
...	...		...		...	...
17C	内部 RAM 地址 63(FIFO)		–		内部 RAM 地址 63	内部 RAM 地址 63
180	内部 RAM 地址 64(TX 缓冲器)		–		内部 RAM 地址 64	内部 RAM 地址 64
...	...		...		...	...
1B0	内部 RAM 地址 76(TX 缓冲器)		–		内部 RAM 地址 76	内部 RAM 地址 76
1B4	内部 RAM 地址 77(空闲)		–		内部 RAM 地址 77	内部 RAM 地址 77
1B8	内部 RAM 地址 78(空闲)		–		内部 RAM 地址 78	内部 RAM 地址 78
1BC	内部 RAM 地址 79(空闲)		–		内部 RAM 地址 79	内部 RAM 地址 79
1C0	(00H)		–		(00H)	–
...	...		...		...	...
1FC	(00H)		–		(00H)	–

17.5.3 发送处理

根据 CAN 协议规范，报文的传输由 CAN 控制器独立完成。微控制器设置标识符，数据长度和待发送数据；然后对命令寄存器的‘发送请求’位置‘1’，来请求发送。当 CAN 控制器正在发送报文时，发送缓冲器被写锁定。所以在防止一个新报文到发送缓冲器之前，微控制器必须检查状态寄存器的‘发送缓冲器状态’标志 (TBS)。

设置命令位 CMR.0 和 CMR.1 会立即产生一次信息发送，当发送错误或仲裁丢失时是不会重发的 (单次发送)。只设置命令位 CMR.0 数据发送失败会重传。在自检测模式下，设置命令位 CMR.4 和 CMR.1 会立即产生一次自接收性质的信息发送。

中止

一个已经请求发送的报文，可以通过置位命令寄存器位的相应位执行‘中止发送’，通过对 CAN_CMR 寄存器中的 AT 位置‘1’，可以中止发送请求。

当 CPU 需要当前请求发送等待时，例如：先发送一条比较紧急的信息时。但当前正在处理的传送是不停止。要想知道源信息是否成功发送，可以通过传送完毕状态位来查看。不过，这应在发送缓冲器状态位置‘1’或产生发送中断后。

要注意的是，即使因为发送缓冲器状态位变为‘释放’而使信息被中止，也会产生发送中断。

如果前一条指令中发送请求被置为‘1’，它不能通过设置发送请求位为‘0’来取消，而应通过中止发送位为‘0’取消。

17.5.4 接收管理

接收到的报文由 CAN 控制器独立完成。收到的报文放在接收缓冲器。可以发送给微控制器的报文，由状态寄存器的接收缓冲器状态标志‘RBS’和接收中断标志‘RI’标出。

查询控制接收

微控制器读 CAN 控制器的状态寄存器，检查接收缓冲状态 (RBS) 查看是否收到一个报文。

当读到 RBS 位为 1，表示收到一个或多个报文，微控制器从 CAN 中取得报文，然后置位命令寄存器的响应标志位‘RRB’发送一个释放接收缓冲器命令。

中断控制接收

中断使能标志位位于 CAN 控制器寄存器里 (对于 BasicCAN 模式) 或位于中断使能寄存器里 (对于 PeliCAN 模式)。如果 CAN 控制器已接收一个报文，而且报文已经通过验收滤波器并放在接收 FIFO，那么会产生一个接收中断。进入中断服务程序，微控制器取走报文，然后置位命令寄存器的响应标志位‘RRB’发送一个释放接收缓冲器命令。

溢出

在接收 FIFO 满了但还接收其他的报文的时候就会导致溢出，同时置位状态寄存器中的数据超载状态位 (如果使能) 通知微控制器有数据溢出的情况，CMR.3 位置‘1’可清除溢出状态。

有效报文

根据 CAN 协议，当报文被正确接收 (直到 EOF 域的最后一位都没有错误)，且通过了标识符过滤，那么该报文被认为是有效报文。

CAN_CMR 寄存器中的 RRB 位是用于清除由数据溢出状态位指出的数据溢出情况。

17.5.5 标识符过滤

在 CAN 协议里，报文的标识符不代表节点的地址，而是跟报文的内容相关的。因此，发送者以广播的形式把报文发送给所有的接收者。节点在接收报文时－根据标识符的值－决定软件是否需要该报文；如果需要，拷贝到内部 BUFFER 里；如果不需要，报文就被丢弃且无需软件的干预。

独立的 CAN 控制器装配了一个多功能的验收滤波器，该滤波器允许自动检查标识符和数据字节。使用这些有效的滤波方法，可以防止对于某个节点无效的报文或报文组存储在接收缓冲器里。因此降低了微控制器的处理负载。

滤波器由验收代码寄存器和屏蔽寄存器根据给定算法来控制。接收到的数据会和验收代码寄存器中的值进行逐位比较。接收屏蔽寄存器定义与比较相关的位置 (0 = 相关，1 = 不相关)。只有收到报文的相应位与验收代码寄存器相应的位相同，报文才会被接收。

BasicCAN 模式里的验收滤波器

该滤波器是由两个寄存器-验收码寄存器 (ACR) 和验收屏蔽寄存器 (AMR) 控制。CAN 报文标识符的高 8 位和这些寄存器里值相比较。可以定义若干个组的标识符为被任何一个节点接收。

例子：验收码寄存器 (ACR) 包括：

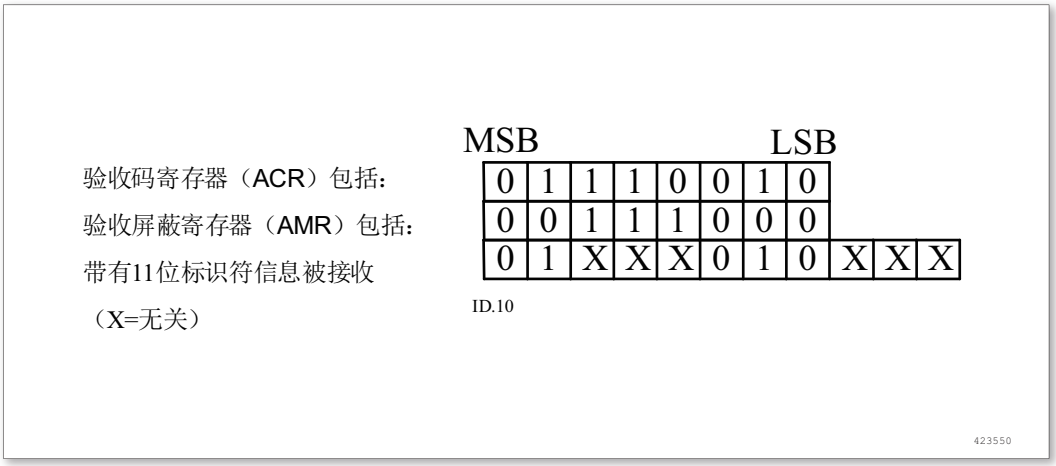


图 136. CAN 标识符接收示例

在验收屏蔽寄存器里是‘1’的位置上，标识符相应的位可以是任意值。这对于三个最低位也一样。因此在这个例子里可以接收 64 个不同的标识符。标识符其他的位必须等于验收代码寄存器相应位的值。

PeliCAN 模式里的验收滤波器

在验收滤波器的帮助下，只有当接收信息中的识别位和验收滤波器预定义的值相等时，CAN 控制器才允许将已接收信息存入 RXFIFO。PeliCAN 模式中，验收滤波器由验收代码寄存器 (ACRn) 和验收屏蔽寄存器 (AMRn) 定义。要接收的信息的位模式在验收代码寄存器中定义。相应的验收屏蔽寄存器允许定义某些位为‘不影响’(即可为任意值)。有两种不同的过滤模式可在模式寄存器中的位 3 选择：

- 单滤波器模式 (1)
- 双滤波器模式 (0)

单滤波器配置

这种滤波器配置可以定义一个长滤波器 (4 字节)。滤波器字节和信息字节之间位的对应关系取决于当前接收帧格式。

标准帧：如果接收的是标准帧格式的信息，在验收滤波中只使用前两个数据字节来存放包括 RTR 位的完整的识别码。如果由于置位 RTR 位而导致没有数据字节，或因为设置相应的数据长度代码而没有或只有一个数据字节，信息也会被接收的。对于一个成功接收的信息，所有单个位的比较后都必须发出接收信号。

注：ACR1 和 AMR1 的低四位是不用的，为了和未来产品兼容，这些位可通过设置 AMR1.3、AMR1.2、

AMR1.1、AMR1.0 为 1 而定为‘不影响’。

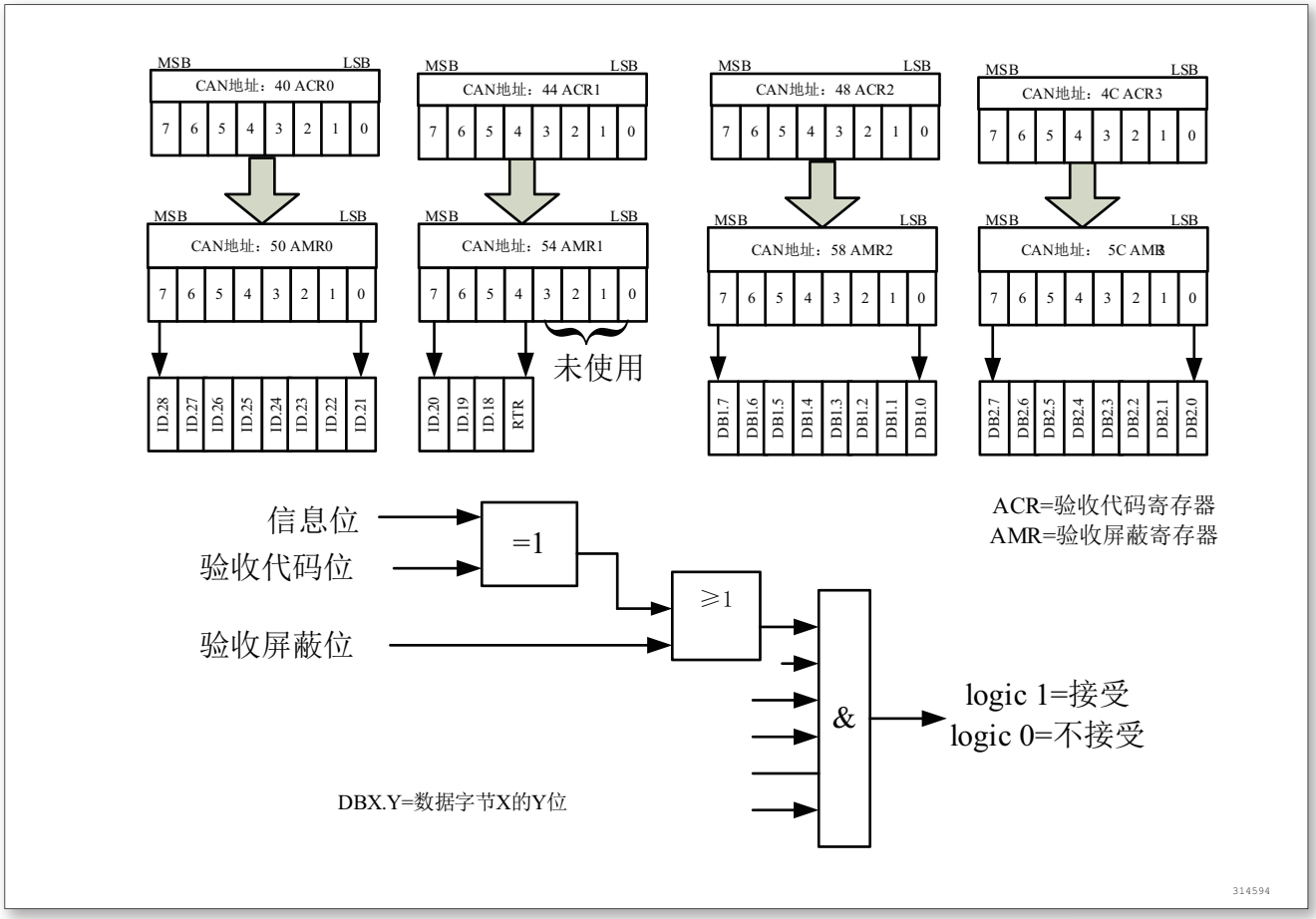


图 137. 接收标准结构信息时的单个滤波器配置

扩展帧：如果接收的信息是扩展帧格式的，包括 RTR 位的全部识别码将被接收过滤使用。为了成功接收信息，每个位都必须比较通过。

注：AMR3 的最低两位和 ACR3 是不用的。这些位应该通过置位 AMR3.1 和 AMR3.0 来定为“不影响”。

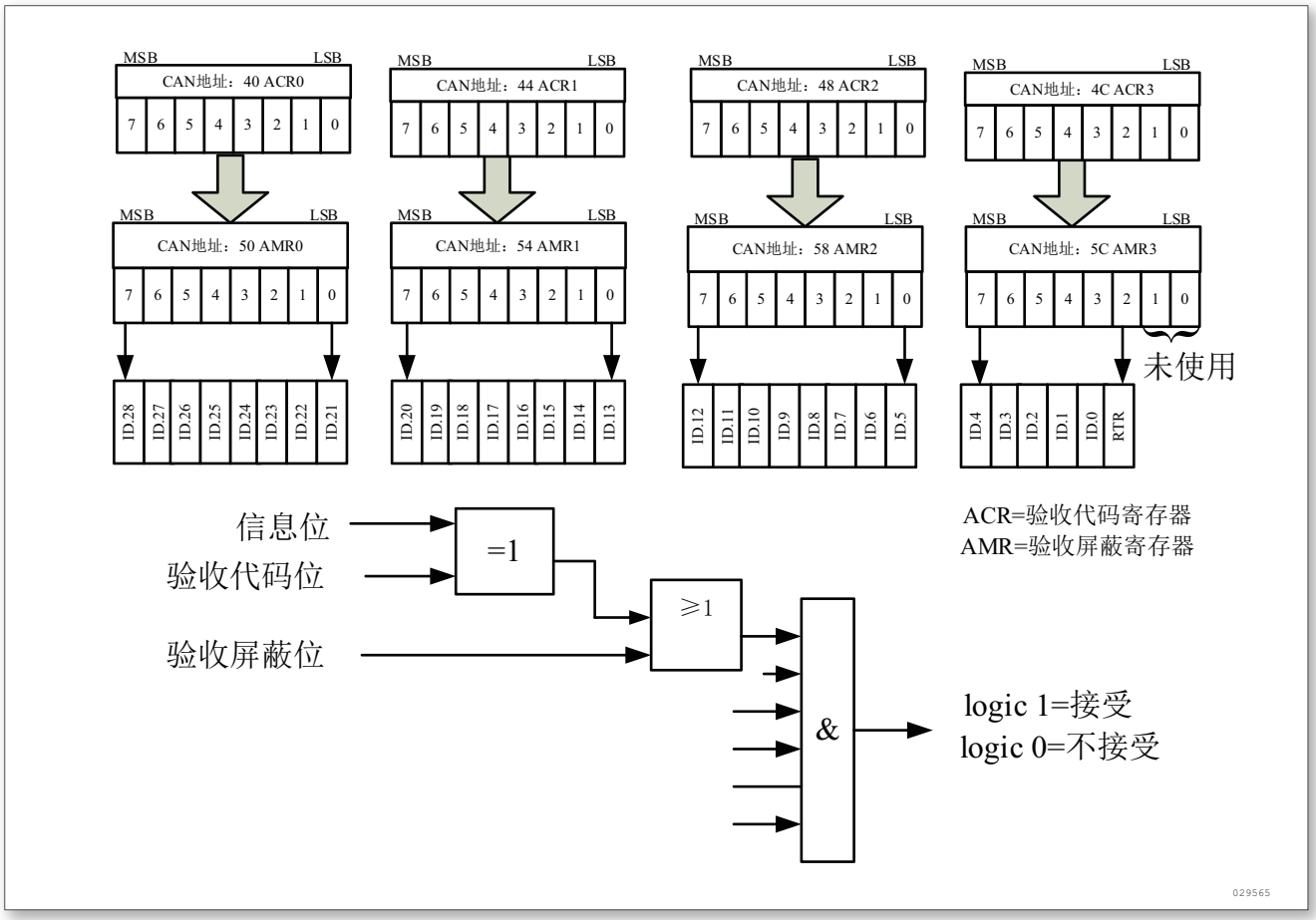


图 138. 单滤波器配置，接收扩展帧信息

双滤波器的配置

这种配置可以定义两个短滤波器。一条接收的信息要和两个滤波器比较来决定是否放入接收缓冲器中。至少有一个滤波器发出接收信号，接收的信息才有效。滤波器字节和信息字节之间位的对应关系取决于当前接收的帧格式。

标准帧：如果接收的是标准帧信息，被定义的两个滤波器是不一样的。第一个滤波器比较包括 RTR 位的整个标准识别码和信息的第一个数据字节。第二个滤波器只比较包括 RTR 位的整个标准识别码。

为了成功接收信息，所有单个位的比较时应至少有一个滤波器表示接收。RTR 位置位或数据长度代码是 0 时表示没有数据字节存在。无论怎样，只要从开始到 RTR 位的部分都被表示接收，信息就可以通过滤波器 1。

如果没有向滤波器请求数据字节过滤，AMR1 和 AMR3 的低四位必须被置为 ‘1’ (不影响)。当使用包括 RTR 位的整个标准识别码时，两个滤波器都同样工作。

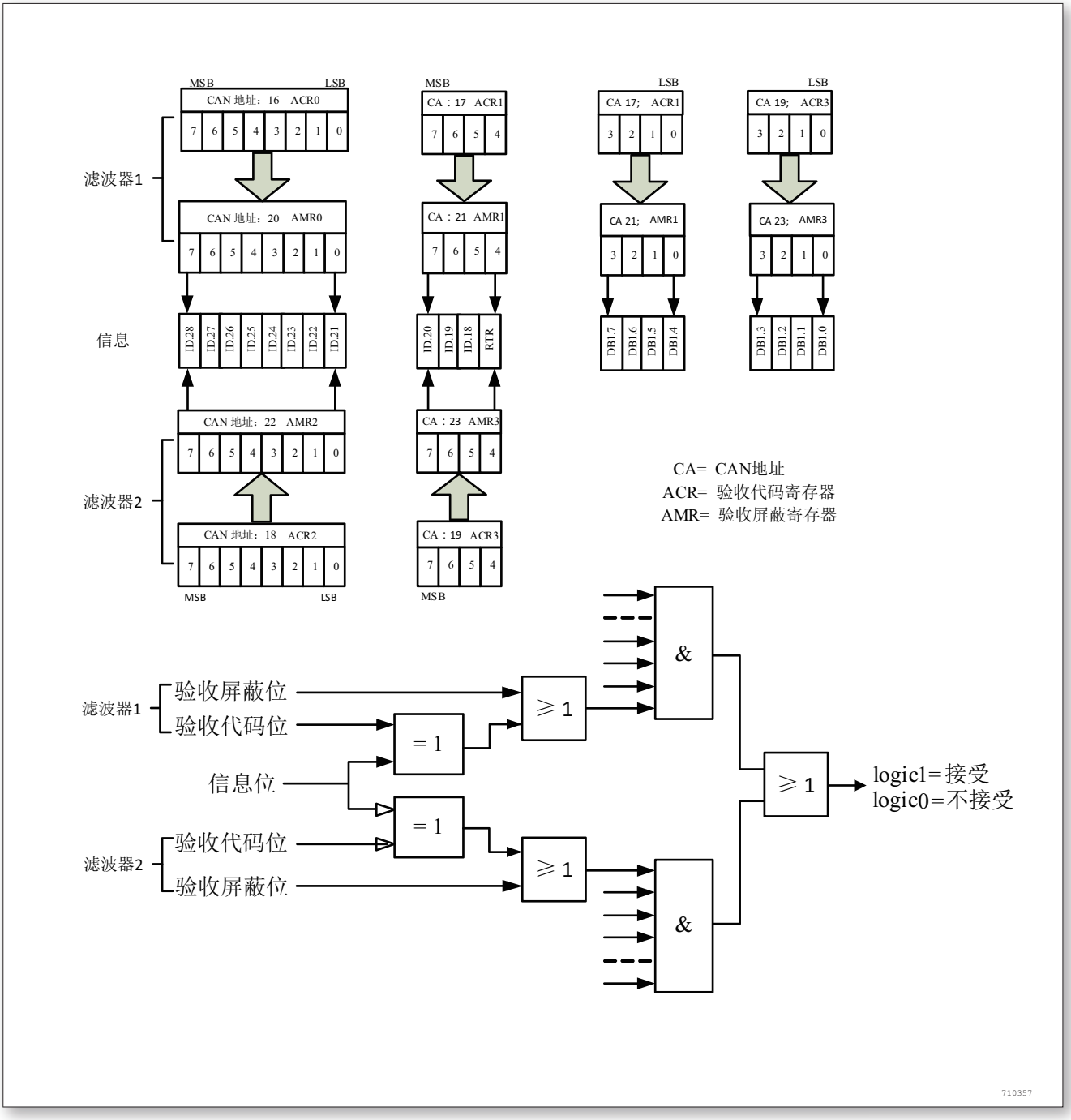


图 139. 接收标准结构信息时的双个滤波器配置

扩展帧：如果接收到扩展帧信息，定义的两个滤波器是相同的。两个滤波器都只比较扩展识别码的前两个字节。

为了能成功接收信息，所有单个位的比较时至少有一个滤波器表示接收。

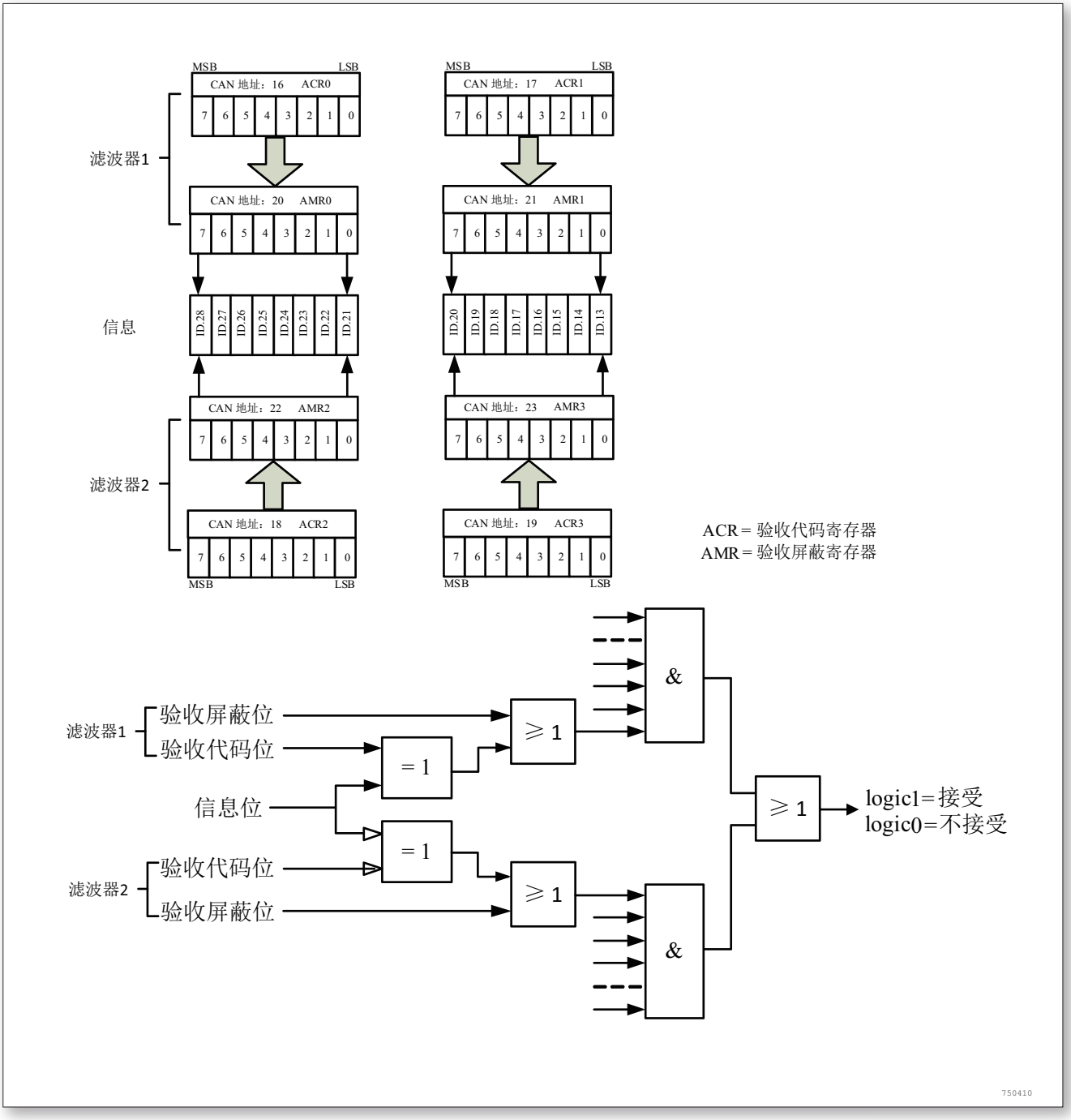


图 140. 双滤波器配置，接收扩展帧信息

例 1：假设下面 1 个标准帧报文要在 PeliCAN 模式里滤波，可以通过使用一个长滤波器完成 (单滤波器模式)。

验收代码寄存器 (ACRn) 和验收屏蔽寄存器 (AMRn) 包括：

n	0	1(高四位)	2	3
ACRn	01XX X010	XXXX	XXXX XXXX	XXXX XXXX
AMRn	0011 1000	1111	1111 1111	1111 1111
接收的报文 (ID28 ~ ID.18，RTR)	01xx x010 xxxx			

X’ = 不相关，‘x’ = 任意值，只使用了 ACR1 和 AMR1 的高四位。

例 2：假设下面 2 个有标准帧标识符的报文在标识符不用进一步译码就被接收。数据和远程帧必须被正确接收。数据字节不要求验收滤波。

报文 1：(ID.28)1011 1100 101(ID.18)

报文 2：(ID.28)1111 0100 101(ID.18)

使用单滤波模式可以接收到四个报文而不仅是要求的两个：

n	0	1(高四位)	2	3
ACRn	1X11 X100	101X	XXXX XXXX	XXXX XXXX
AMRn	0100 1000	0001	1111 1111	1111 1111
接收的报文 (ID28 ~ ID.18，RTR)	1011 0100 101x 1111 0100 101x (报文 2) 1011 1100 101x (报文 1) 1111 1100 101x			

(‘X’ = 不相关，‘x’ = 任意值，只使用了 ACR1 和 AMR1 的高四位。)

这个结果需要进一步解码才能满足接收两条信息的要求。

使用双滤波器可以得到正确的结果

	滤波器 1			滤波器 2	
n	0	1	3 低四位	2	3 高四位
ACRn	1011 1100	101X XXXX	... XXXX	1111 0100	101X ...
AMRn	0000 0000	0001 1111	... 1111	0000 0000	0001 ...
接收的信息 (ID28 ~ ID.18，RTR)	1011 1100 101X(报文 1)			1111 0100 101X(报文 2)	

(‘X’ = 不相关，‘x’ = 任意值)

报文 1 被滤波器 1 接收，报文 2 被滤波器 2 接收。如果报文至少被两个滤波器中的一个接收，报文就被存放到接收 FIFO。这种方法可满足于这种要求。

例 3：在这个例子里，使用一个长的验收滤波器过滤一组带有扩展标识符的报文。

n	0	1	2	3(高六位)
ACRn	1011 0100	1011 000X	1100 XXXX	0011 0XXX
AMRn	0000 0000	0001 0001	0000 1111	0000 0111
接收的信息 (ID28 ~ ID.18，RTR)	1011 0100 101x 000x 1100 xxxx 0011 0x			

(‘X’ = 不相关，‘x’ = 任意值，只使用了 ACR1 和 AMR1 的高六位。)

例 4：有些使用标准帧系统仅用 11 位标识符和头两个数据字节识别报文。如果报文超过 8 个数据字节，头两个数据字节定义为报文头和使用分段存储协议就会使用像这样的协议。

例如 DeviceNet。对于这种系统类型，CAN 控制器除了 11 位标识符和 RTR 位外，在单滤波器模式里能滤波两个数据字节，在双滤波器模式里能过滤一个数据字节 (除了 11 位标识符和 RTR 位)。

下面的例子显示了用双滤波器模式，在这种系统里有效地滤波报文：

	滤波器 1			滤波器 2	
n	0	1	3 低四位	2	3 高四位
ACRn	1110 1011	0010 1111	... 1001	1111 0100	XXX0 ...
AMRn	0000 0000	0000 0000	... 0000	0000 0000	1110 ...
接收的信息 (ID28 ~ ID.18，RTR)	1110 1011 0010 + 1111 ... 1001 标识符 RTR + 头一个数据字节			1111 0100 标识符	xxx0 RTR

(‘X’ = 不相关，‘x’ = 任意值)

- 滤波器 1 滤波的报文有：
  - 标识符‘11101011001’
  - RTR = ‘0’，也就是说是数据帧
  - 数据字节‘11111001’(这是指例如 DeviceNet：一个信息的所有段都被过滤)
- 滤波器 2 用来过滤一组 8 个报文，其中报文有:
  - 标识符‘11110100 000’到 11110100111’
  - RTR = ‘0’，也即是数据帧

17.5.6 报文存储

要在 CAN 总线上发送的数据被载入 CAN 控制器的存储区，这个存储区叫‘发送缓冲器’。从 CAN 总线上收到的数据也存储在 CAN 控制器的存储区，这个存储区叫‘接收缓冲器’。这些缓冲器包括 2，3 或 5 个字节的标识符和帧信息 (取决于模式和帧类型)，而最多可以包含 8 个数据字节。

BasicCAN 模式

缓冲器长达 10 个字节

- 2 个标识符字节
- 8 个数据字节

表 62. BasicCAN 模式里的 RX 和 TX 缓冲器

相对 CAN 偏移量		寄存器		组成与注释
TX (16 进制)	RX (16 进制)	TX	RX	
28	50	CAN_TXIDR1	CAN_RXIDR1	8 位标识符
2C	54	CAN_TXIDR2	CAN_RXIDR2	3 位标识符，1 位远程传输请求位，4 位数据长度代码，表示数据字节的数量
30	58	CAN_TXDR1	CAN_RXDR1	由数据长度代码表示，最多 8 个数据字节
34	5C	CAN_TXDR2	CAN_RXDR2	
38	60	CAN_TXDR3	CAN_RXDR3	

相对 CAN 偏移量		寄存器		组成与注释
TX (16 进制)	RX (16 进制)	TX	RX	
3C	64	CAN_TXDR4	CAN_RXDR4	由数据长度代码表示，最多 8 个数据字节
40	68	CAN_TXDR5	CAN_RXDR5	
44	6C	CAN_TXDR6	CAN_RXDR6	
48	70	CAN_TXDR7	CAN_RXDR7	
4C	74	CAN_TXDR8	CAN_RXDR8	

PeliCAN 模式

这些缓冲器是 13 个字节长

- 1 字节帧信息
- 2 个或 4 个标识符字节 (标准帧或扩展帧)
- 最多 8 个数据字节

发送缓冲器

发送缓冲器的全部列表见下图。请务必分清标准帧格式 (SFF) 和扩展帧格式 (EFF) 配置。发送缓冲器允许定义长达 8 个数据字节发送信息。

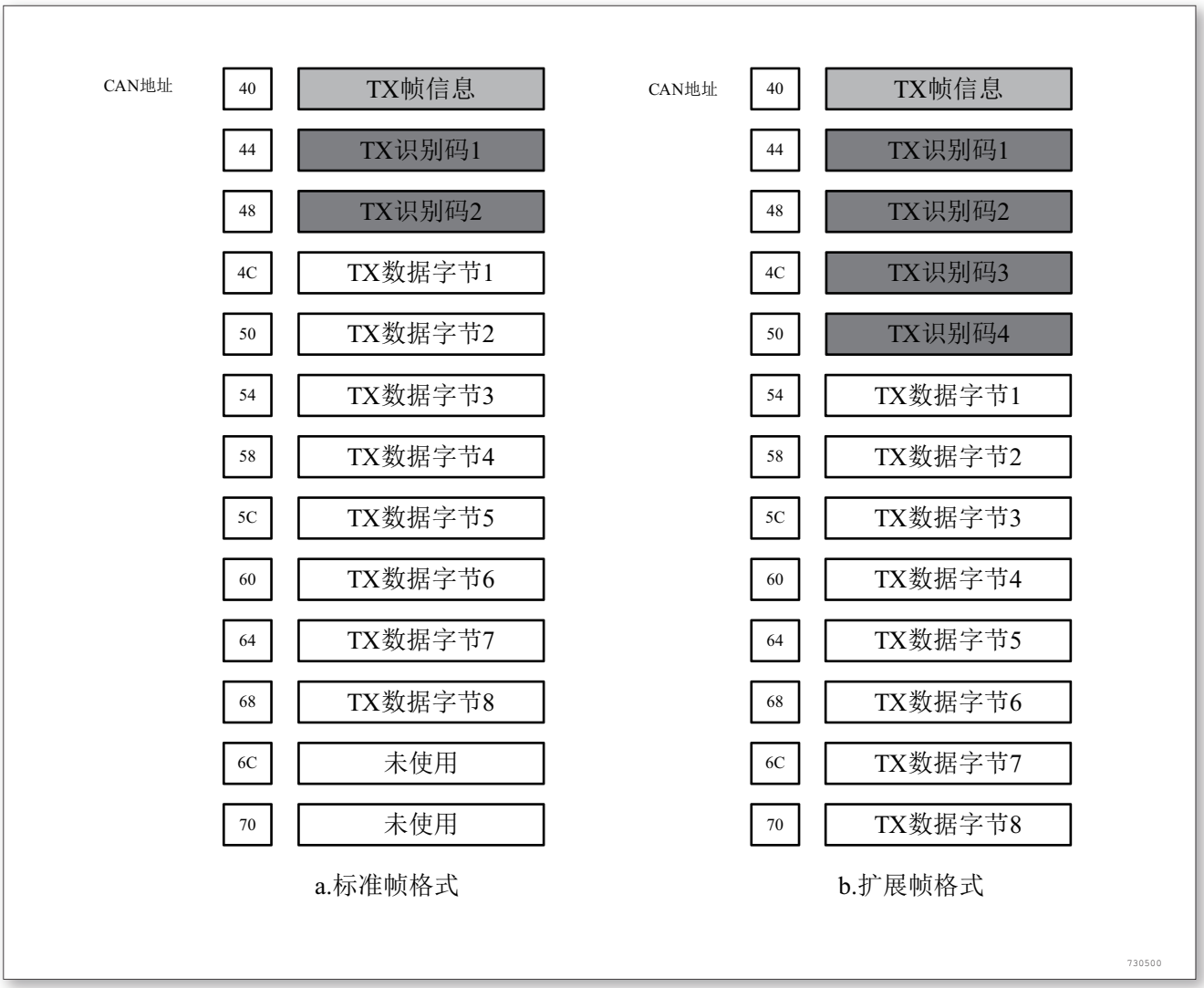


图 141. 标准帧和扩展帧格式配置在发送缓冲器的列表

帧信息中的 FF 位决定 CAN 控制器将要发送扩展帧格式还是标准帧格式。

接收缓冲器

接收缓冲器的列表与发送缓冲器很相似。接收缓冲器是 RXFIFO 的可访问部分，位于 CAN 地址的 40 ~ 70。每条信息都分为描述区和数据区。

注：在帧信息字节中的接收字节长度代码代表实际发送的数据长度代码，它有可能大于 8(取决于发送器)。无论如何，最大接收数据字节数是 8。这一点在读接收缓冲器中的信息时应当考虑。

见下图，RXFIFO 共有 64 个信息字节的空间。一次可以存储多少条信息取决于数据的长度。如果 RXFIFO 中没有足够的空间来存储新的信息，CAN 控制器会产生数据溢出条件，此时信息有效且接收检测为肯定。发生数据溢出情况时，已部分写入 RXFIFO 的信息将被删除。这种情况可以通过状态寄存器和数据超限中断 (中断允许) 反应到 CPU。

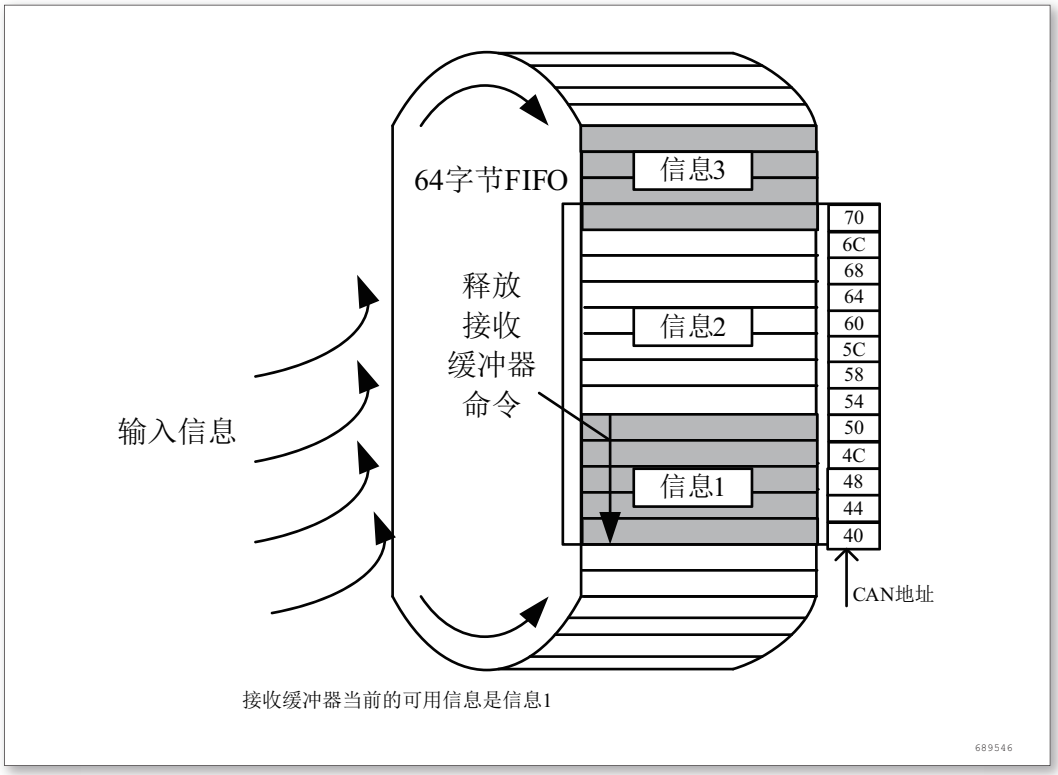


图 142. 标准帧和扩展帧格式配置在发送缓冲器的列表

17.5.7 出错管理

基于错误计数器的值，每个 CAN 控制器能够在三种错误状态之一中工作：错误激活、错误认可或总线离线。如果错误计数器的值都在 0 ~ 127 之间，CAN 控制器是错误激活的。此时产生错误激活标志 (6 个显性位)。如果一个错误计数器的值在 128 ~ 255 之间，CAN 控制器是错误认可的。此时，在检测到错误前，产生认可错误标志 (6 个隐性位)。如果发送错误计数器的值高于 255，则到达总线离线状态。在这种状态下，自动置位复位请求，CAN 控制器对总线没有影响。总线离线状态只能在微控制器用命令‘复位请求 = 0’退出。这将启动总线离线恢复定时器，发送错误计数器计数 128 个总线释放信号。计数结束后，两个错误计数器都是 0，器件再次处于错误激活状态。

错误计数器

如上面描述，CAN 的错误状态和发送错误计数器和接收错误计数器的值直接相关。

为了仔细研究错误界定，支持 CAN 控制器的增强的错误分析功能，CAN 控制器提供可读的错误计数器。另外，在复位模式，允许对于两个错误计数器进行写访问。

出错中断

有三个中断源来向微处理器发出错的状态。每个中断都能在中断使能寄存器里分别使能。

- 总线出错中断：在 CAN 总线上检测到任何一个错误都会产生中断
- 出错警告中断: 如果超过出错警告界限，产生出错警告中断。而且它在 CAN 控制器进入总线离线状态和再次之前再一次进入错误激活状态也会产生这个中断。CAN 控制器的出错警告界限在复位模式中可编程。复位后的默认值是 96。
- 错误认可中断: 如果错误状态从错误激活变成错误认可或相反，将产生错误认可中断。

错误码捕捉

CAN 控制器可以执行在 CAN2.0B 规范定义的所有错误界定。每个 CAN 控制器处理错误的整个过程是完全自动的。但是，为了向用户提供某个错误的详细信息，CAN 控制器提供了错误代码捕捉功能。无论什么时候发生 CAN 总线错误，它都会强制产生相应的总线出错中断。同时，当前位的位置被捕捉入错误代码捕捉寄存器。在主控制器将捕捉的数据读出前，它都会被保存在寄存器中。然后捕捉机制再次激活。寄存器可以内容区分四种错误类型：格式出错、填充出错、位出错和其他错误。如下图表示，寄存器还另外表明在错误是在报文的接收还是发送期间发生。这个寄存器中的五个位表示 CAN 帧内错误的位位置，更多信息参考下面的表和数据表。

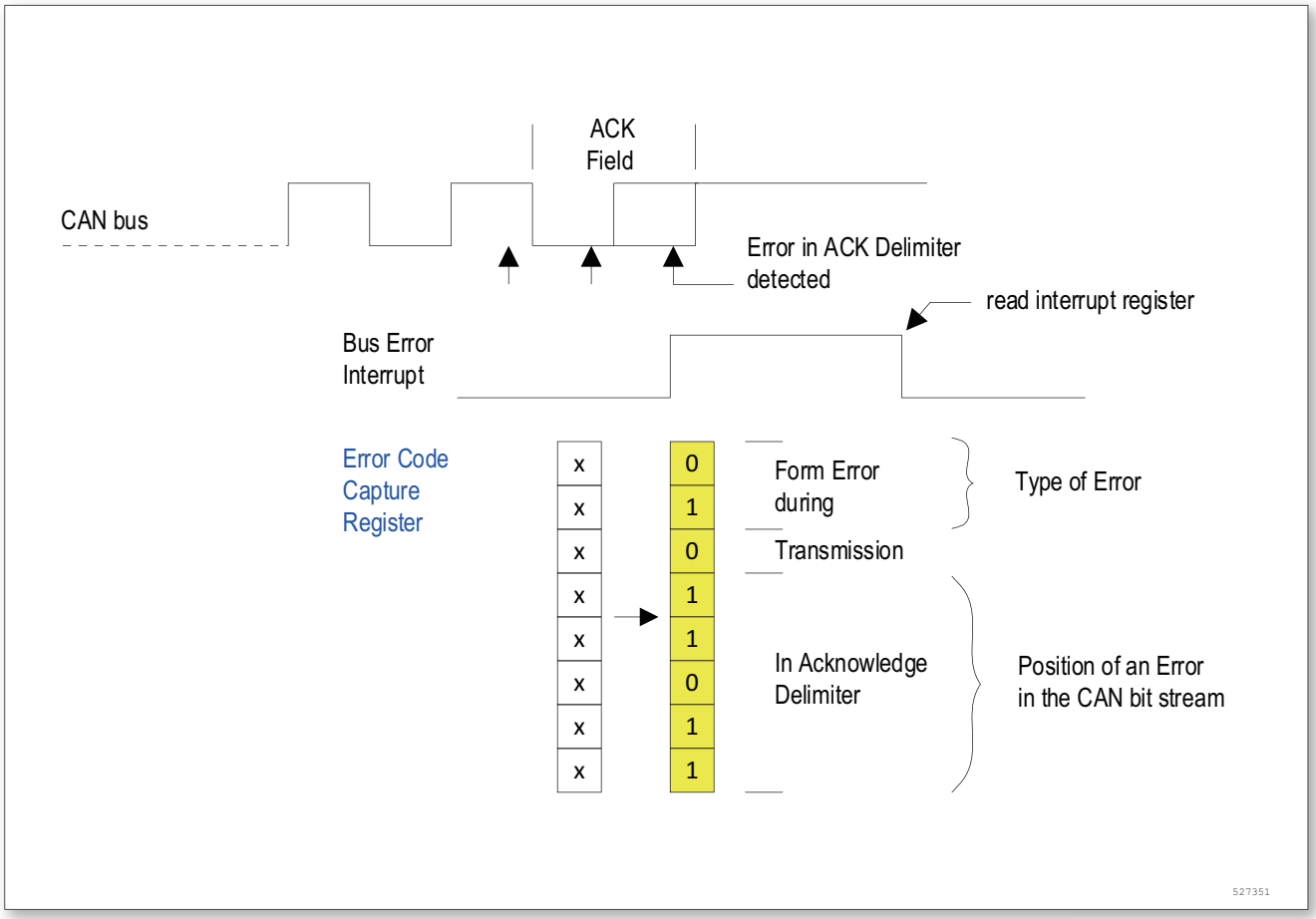


图 143. 错误码捕捉功能举例

CAN 规范定义了：CAN 总线上的每个位只有特殊类型的错误。下面两张显示了 CAN 报文发送和接收期间可能出现的所有错误。左边的部分包括位置和错误的类型，这些由错误码捕捉寄存器捕捉。每张表的右边部分是将错误码转换成上层的错误描述，可以直接从寄存器内容知道其含义。通过使用这些表格，能得到有关错误计数器的变化和在器件发送和接收管脚的错误状态的更多信息。使用这些表时，例如在错误分析软件里，可以详细地分析每个错误状态。关于 CAN 错误类型和位置的信息能用于错误统计和系统维护或在系统优化器件进行纠正。

表 63. 接收时可能出现的错误

CAN 位流里的错误位置	错误类型	RX 错误计数	错误码捕捉	描述
标识符 SRR、IDE 和 RTR 位 保留位 数据长度码 数据场 CRC 序列	填充	+1	收到 5 个电平相同的连续位	–
CRC 定界符	格式 填充	+1 +1	RX = 显性 收到超过 5 个电平相同的连续的位	位必须是隐性
应答位	位	+1	RX = 显性，或检测到 CRC 错误	临界的总线定时或总线长度 CRC 序列不正确
应答定界符	格式	+1	RX = 显性，或检测到 CRC 错误	临界的总线定时或总线长度 CRC 序列不正确
帧结束	格式 其他	+1 ±0	RX = 头六位是显性 RX = 最后一位的显性	– 反应：发出超载标志，如果发送器重新发送，数据可能重复
间隔	其他	±0	RX = 显性	反应：接收器发出超载标志
激活错误标志	位	+8	TX = 显性，但 RX = 隐性	不能写显性位
容许的显性位	其他	+8	TX = 显性，但 RX = 隐性	不能写显性位
错误定界符	格式 其他	+8	TX = 显性，但 RX = 隐性	不能写显性位
超载标志	位	+8	TX = 显性，但 RX = 隐性	不能写显性位

表 64. 发送时可能出现的错误

CAN 位流里的错误位置	错误类型	TX 错误计数	错误码捕捉	描述
帧开始	位	+8	TX = 显性，但 RX = 隐性	不能写显性位
标识符	位 填充	+ 8 ± 0	TX = 显性，但 RX = 隐性 TX = 隐性，但 RX = 显性	不能写显性位 –
SRR 位	位 填充	+ 8 ± 0	TX = 显性，但 RX = 隐性 TX = 隐性，但 RX = 显性	不能写显性位 –
IDE 和 RTR 位	位 填充	+ 8 ± 8	TX = 显性，但 RX = 隐性 TX = 隐性，但 RX = 显性	不能写显性位 –
保留位 数据长度码 数据场 CRC 序列	位	+8	TX = 显性，但 RX = 隐性	不能写显性位

CAN 位流里的错误位置	错误类型	TX 错误计数	错误码捕捉	描述
CRC 定界符	格式	+8	RX = 显性	位必须是隐性
应答隙	其他 其他	+ 8 ± 0	RX = 隐性 (错误激活) RX = 隐性 (错误认可)	没有应答 没有应答，节点可能单独在总线上
应答应界符	格式	+8	RX = 显性	临界的总线定时或总线长度
帧结束	格式 其他	+8 +8	RX = 头六位是显性 RX = 最后一位是显性位	– 帧已经被一些节点接收，再次发送可能导致接收器里数据重复
间隔	其他	±0	RX = 显性	来自于“旧”CAN 控制器的超载标志
激活错误标志 过载标志	位	+8	TX = 显性，但 RX = 隐性	不能写显性位
允许显性位	格式	+8	RX = 在激活错误标志或过载标志后有超过 7 个显性位	–
错误定界符	格式 其他	+8 ±0	RX = 头七位是显性位 RX = 定界符的最后一位是显性位	–
认可错误标志	其他	+ 8	RX = 显性 (错误认可)	没有收到应答，节点不是单独在总线上。

离线恢复

如果发送错误计数器的值高于 255，则达到总线离线状态。总线状态位被置为‘1’。在这种状态下，自动置位复位请求位，CAN 控制器对总线没有影响。在错误中断允许的情况下，会产生一个错误中断。这种状态会持续到 CPU 清除复位请求位。所有这些完成后，CAN 控制器将会等待协议规定的最小时间 (128 个总线空闲信号)。

17.5.8 位时间特性

位时间特性逻辑通过采样来监视串行的 CAN 总线，并且通过跟帧起始位的边沿进行同步，及通过跟后面的边沿进行重新同步，来调整其采样点。

它的操作可以简单解释为，如下所述把名义上的每位的时间分为 3 段：

- 同步段 ($t_{SYNCSEG}$)：通常期望位的变化发生在该时间段内。其值固定为 1 个时间单元 (1 x t_{CAN})。
- 时间段 1(t_{TSEG1})：定义采样点的位置。它包含 CAN 标准里的 PROP_SEG 和 PHASE_SEG1。其值可以编程为 1 到 16 个时间单元，但也可以被自动延长，以补偿因为网络中不同节点的频率差异所造成的相位的正向漂移。
- 时间段 2(t_{TESG2}): 定义发送点的位置。它代表 CAN 标准里的 PHASE_SEG2。其值可以编程为 1 到 8 个时间单元，但也可以被自动缩短以补偿相位的负向漂移。

CAN 系统时钟 t_{SCL} 的周期是可编程的，而且决定了相应的位时序。

CAN 系统时钟由如下公式计算：t_{SCL} = 2 × t_{CLK} ×(BRP+ 1)。这里 t_{CLK} = APB1 的频率周期同步跳跃宽度 (SJW) 定义了，在每位中可以延长或缩短多少个时间单元的上限。为了补偿在不同总线控制器的时钟振荡器之间的相位偏移，任何总线控制器必须在当前传送的相关信号边沿重新同步。同步跳转宽度定义了每一位周期可以被重新同步缩短或延长的时钟周期的最大数目

t_{SJW} = t_{SCL} ×(SJW + 1) 时间段 1(TSEG1) 和时间段 2(TSEG2) 决定了每一位的时钟数目和采样点的位置，这里：

$t_{SYNCSEG}$ = 1 × t_{SCL}

t_{TESG1} = t_{SCL} ×(TESG1+ 1)

t_{TESG2} = t_{SCL} ×(TESG2+ 1)

有效跳变被定义为，当 CAN 自己没有发送隐性位时，从显性位到隐性位的第 1 次转变。如果在时间段 1(t_{TSEG1}) 而不是在同步段 ($t_{SYNCSEG}$) 检测到有效跳变，那么 t_{TSEG1} 的时间就被延长最多 SJW 那么长，从而采样点被延迟了。

相反如果在时间段 2(t_{TSEG2}) 而不是在 $t_{SYNCSEG}$ 检测到有效跳变，那么 t_{TSEG2} 的时间就被缩短最多 SJW 那么长，从而采样点被提前了。

为了避免软件的编程错误，对位时间特性寄存器 (CAN_BTR) 的设置，只能在 CAN 处于初始化状态下进行。

CAN 波特率 = APB1 /(2*(BRP + 1)*(TSEG1+ 1+ TSEG2+ 1+ 1))；

17.5.9 仲裁丢失

仲裁丢失时，会产生相应的仲裁丢失中断 (中断允许)。同时，位流处理器的当前位位置被捕捉送入仲裁丢失捕捉寄存器。一直到用户通过软件读这个值，寄存器中的内容都不会变。随后，捕捉机制又被激活了。

读中断寄存器时，中断寄存器中相应的中断标志位被清除。直到仲裁丢失捕捉寄存器被读一次之后，新的仲裁丢失中断才有效。

下图为仲裁丢失位解释：

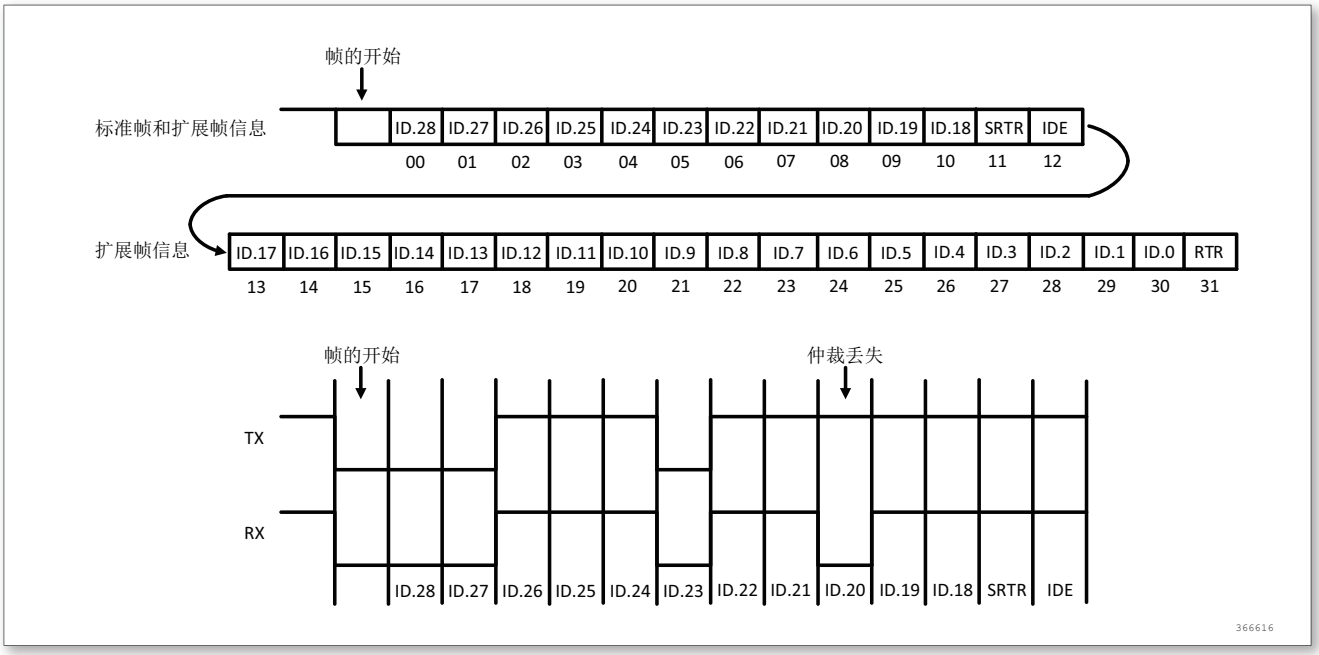


图 144. 仲裁丢失解释举例

17.5.10 CAN 中断

BasicCAN 模式里有 5 个中断：

- 接收中断
  当接收 FIFO 不空和接收中断使能 (CAN_CR 寄存器位 1) 时产生。CAN_IR 寄存器 RI 位被置‘1’。
- 发送中断
  发送缓冲器状态从 0 变为 1(释放) 和发送中断使能 (CAN_CR 寄存器位 2) 时产生。
- 错误中断
  错误中断使能 (CAN_CR 寄存器位 3) 时，错误状态位或总线状态位的变化会置位此位。
- 数据溢出中断
  大概数据溢出中断使能位 (CAN_CR 寄存器位 4) 被置‘1’时向数据溢出状态位‘0 - 1’跳变。

PeliCAN 模式里有 8 个不同的中断：

- 接收中断
  接收 FIFO 不空且中断寄存器的 RIE 位被置位时产生中断。
- 发送中断
  发送缓冲器状态从‘0 - 1’(释放) 跳变且中断寄存器的 TIE 位被置位时产生中断。
- 错误报警中断
  错误状态位和总线状态位的改变和中断寄存器的 EIE 位被置位时产生中断。
- 数据溢出中断
  数据溢出状态位有‘0 - 1’跳变且中断寄存器的 DOIE 位被置位时产生中断。
- 错误消极中断
  当 CAN 控制器到达错误消极状态 (至少一个错误计数器超过协议规定的值 127) 或从错误消极状态又进入错误活动状态以及中断寄存器的 EPIE 位被置位时产生中断。
- 仲裁丢失中断

当 CAN 控制器丢失仲裁，变为接收器和中断使能寄存器的 ALIE 为被置位时产生中断。

- 总线错误中断

  当 CAN 控制器检测到总线错误且中断使能寄存器中的 BEIE 被置位时产生中断。

17.6 CAN 寄存器描述

表 65. CAN 寄存器概览

Offset	Acronym	Register Name	Reset	Section	Note
0x00	CAN_MOD	CAN 模式寄存器	0x00000001	小节 17.6.1	仅 PeliCAN 模式
0x00	CAN_CR	CAN 控制寄存器	0x00000001	小节 17.6.2	仅 BasicCAN 模式
0x04	CAN_CMR	CAN 命令寄存器	0x000000XX	小节 17.6.3	复位值： BasicCAN 模式：0x00FF PeliCAN 模式：0x0000
0x08	CAN_SR	CAN 状态寄存器	0x00000000	小节 17.6.4	–
0x0C	CAN_IR	CAN 中断寄存器	0x00000000	小节 17.6.5	–
0x10	CAN_IER	CAN 中断使能寄存器	0x00000000	小节 17.6.6	仅存在 PeliCAN 模式
0x10	CAN_ACR	CAN 验收代码寄存器	0x000000XX	小节 17.6.7	BasicCAN 模式
0x14	CAN_AMR	CAN 验收屏蔽寄存器	0x000000XX	小节 17.6.8	BasicCAN 模式
0x18	CAN_BTR0	CAN 总线定时 0	0x00000000	小节 17.6.9	–
0x1C	CAN_BTR1	CAN 总线定时 1	0x00000000	小节 17.6.10	–
0x28	CAN_TXID0	CAN 发送识别码寄存器 0	0x000000XX	小节 17.6.11	仅存在 BasicCAN 模式，复位模式为 0xFF
0x2C	CAN_TXID1	CAN 发送识别码寄存器 1	0x000000XX	小节 17.6.12	仅存在 BasicCAN 模式，复位模式为 0xFF
0x2C	CAN_ALC	CAN 仲裁丢失捕捉寄存器	0x00000000	小节 17.6.13	仅存在 PeliCAN 模式
0x30	CAN_ECC	CAN 错误代码捕捉	0x00000000	小节 17.6.14	仅存在 PeliCAN 模式
0x34	CAN_EWLR	CAN 错误报警限制寄存器	0x00000096	小节 17.6.15	仅存在 PeliCAN 模式
0x38	CAN_RXERR	CAN RX 错误计数寄存器	0x00000000	小节 17.6.16	仅存在 PeliCAN 模式
0x3C	CAN_TXERR	CAN TX 错误计数寄存器	0x00000000	小节 17.6.17	仅存在 PeliCAN 模式
0x40	CAN_SFF	CAN 发送帧信息寄存器	0x000000XX	小节 17.6.18	仅存在 PeliCAN 模式
0x44	CAN_TXID0	CAN 发送识别码寄存器 0	0x000000XX	小节 17.6.19	仅存在 PeliCAN 模式
0x48	CAN_TXID1	CAN 发送识别码寄存器 1	0x000000XX	小节 17.6.20	仅存在 PeliCAN 模式
0x4C	CAN_TXDATA0	CAN 发送数据寄存器 0	0x000000XX	小节 17.6.21	仅存在 PeliCAN 模式
0x50	CAN_TXDATA1	CAN 发送数据寄存器 1	0x000000XX	小节 17.6.22	仅存在 PeliCAN 模式
0x7C	CAN_CDR	CAN 时钟分频寄存器	0x00000000	小节 17.6.23	–

17.6.1 CAN 模式寄存器 (CAN_MOD)

仅 PeliCAN 模式

偏移地址：0x00

复位值：0x0000 0001

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.												AFM	STM	LOM	RM
												rw	rw	rw	rw

Bit	Field	Type	Reset	Description
31 : 4	Reserved			保留，始终读为 0。
3	AFM	rw	0x00	验收滤波器模式 (Acceptance filter mode) 1：单；选择单个验收滤波器 (32 位长度) 0：双；选择两个验收滤波器 (每个有 16 位激活)
2	STM	rw	0x00	自检测模式 (Self test mode) 1：自检测；此模式可以检测所有节点，没有任何活动的节点使用自接收命令；即使没有应答，CAN 控制器也会成功发送 0：正常模式；成功发送时必须应答信号
1	LOM	rw	0x00	只听模式 (Listen only mode) 1：只听；这种模式中，即使成功接收信息，CAN 控制器也不向总线发应答信号；错误计数器停止在当前值 0：正常模式
0	RM	rw	0x01	复位模式 (Reset mode) 1：复位；检测到复位模式位被置位，中止当前正在接收/发送的信息，进入复位模式 0：正常；复位模式位接收到‘1-0’的跳变后，CAN 控制器回到工作模式

17.6.2 CAN 控制寄存器 (CAN_CR)

仅 BasicCAN 模式：

偏移地址：0x00

复位值：0x0000 0001

控制寄存器的内容是用于改变 CAN 控制器的行为的。这些位可以被微控制器设置或置位，微控制器可以对控制寄存器进行读/写操作。

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.											OIE	EIE	TIE	RIE	RR
											rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
31：5	Reserved			保留，始终读为 0。
4	OIE	rw	0x00	溢出中断使能 (Overflow interrupt enable) 1：使能；如果置位数据溢出位，微控制器接收溢出中断信号 0：禁止；微控制器不从 CAN 控制器接收溢出中断信号
3	EIE	rw	0x00	错误中断使能 (Error interrupt enable) 1：使能；如果出错或总线状态改变，微控制器接收错误中断信号 0：禁止；微控制器不从 CAN 控制器接收错误中断信号
2	TIE	rw	0x00	发送中断使能 (Transmit interrupt enable) 1：使能；当信息被成功发送或发送缓冲器又被访问时 (例如，中止发送命令后)，微控制器接收 CAN 控制器发出的一个发送中断信号 0：禁止；微控制器不从 CAN 控制器接收发送中断信号
1	RIE	rw	0x00	接收中断使能 (Receive interrupt enable) 1：使能；信息被无错误接收时，CAN 控制器发出的一个接收中断信号到微控制器 0：禁止；微控制器不从 CAN 控制器接收接收中断信号
0	RR	rw	0x01	复位请求 (Reset request) 1：当前；CAN 控制器检测到复位请求后，中止当前发送/接收的信息，进入复位模式 0：空缺；复位请求位接收到一个下降沿后。CAN 控制器回到工作模式

17.6.3 CAN 命令寄存器 (CAN_CMR)

偏移地址：0x04

复位值：BasicCAN 模式：0x0000 00FF

PeliCAN 模式：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.											SRR	CDO	RRB	AT	TR
											w	w	w	w	w

Bit	Field	Type	Reset	Description
31：5	Reserved			保留
4	SRR	w		PeliCAN 模式： SRR：自接收请求 (Self reset request) 1：当前信息可被同时发送和接收 0：(空缺)

Bit	Field	Type	Reset	Description
3	CDO	w		清除数据溢出 (Clear data overrun) 1：清除；清除数据溢出状态位 0：无动作 清除数据溢出这个命令位是用来清除由数据溢出状态位指出的数据溢出情况的。如果数据溢出位被置位，不会产生数据溢出中断。释放接收缓冲器命令的同时是可以发出清除数据溢出命令的。
2	RRB	w		释放接收缓冲器 (Release receive buffer) 1：释放；接收缓冲器中存放信息的内存空间将被释放 0：无动作 读接收缓冲器之后，微控制器可以通过设置释放接收缓冲器位为 1 来释放 RXFIFO 中当前信息的内存空间。这可能会导致接收缓冲器中的另一条信息立即有效。这样会再产生一次接收中断 (使能条件下)。如果没有其它可用信息，就不会再产生接收中断，接收缓冲器状态位被清除。
1	AT	w		中止传输 (Abort transmission) 1：当前；如果不是在处理过程中，等待处理的发送请求将取消 0：空缺；无动作 中止传送位是在 CPU 要求当前传送暂停时使用的，例如，传送一条紧急信息。正在进行的传送是不停止的。要查看原始信息是否被成功发送，可以通过传送成功状态位来检测。不过，这必须在发送缓冲器状态位为 1(释放) 或发送中断产生的情况下才能实现。
0	TR	w		发送请求 (Transmission request) 1：当前；信息被发送 0：空缺；无动作 如果发送请求在前面的命令中被置位。它就不可以通过直接设置为 0 来取消它了。不过，可以通过设置中止发送位为 0 来取消。

17.6.4 CAN 状态寄存器 (CAN_SR)

偏移地址：0x08

复位值：0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.								BS	ES	TS	RS	TCS	TBS	DOS	RBS
								rw	rw	rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
31：8	Reserved			保留，始终读为 0。
7	BS	rw	0x00	BS：总线状态 (Bus status) 1：总线关闭；CAN 控制器退出总线活动 0：总线开启；CAN 控制器加入总线活动 当传输错误计数器超过限制 (255)(总线状态位置‘1’ - 总线关闭)，CAN 控制器就会将复位请求位置‘1’(当前)，在错误中断允许的情况下，会产生一个错误中断。这种状态会持续直到 CPU 清除复位请求位。所有这些完成之后，CAN 控制器将会等待协议规定的最小时 (128 个总线空闲信号)。总线状态位被清除后 (总线开启)，错误状态位被置为‘0’(ok)，错误计数器复位且产生一个错误中断 (中断允许)
6	ES	rw	0x00	出错状态 (Error status) 1：出错；至少出现一个错误计数器满或超过 CPU 报警限制 0：ok；两个错误计数器都在报警限制以下 根据 CAN 2.0B 协议说明，在接收或发送时检测到错误会影响错误计数。当至少有一个错误计数器满或超出 CPU 警告限制 (96) 时，错误状态位被置位。在允许情况下，会产生错误中断。
5	TS	rw	0x00	发送状态 (Transmit status) 1：发送；CAN 控制器在传送信息 0：空闲；没有要发送的信息 如果接收状态位和发送状态位都是 0，则 CAN 总线是空闲的。
4	RS	rw	0x00	接收状态 (Receive status) 1：接收；CAN 控制器正在接收信息 0：空闲；没有正在接收的信息 如果接收状态位和发送状态位都是 0，则 CAN 总线是空闲的。
3	TCS	rw	0x00	发送完毕状态 (Transmission complete status) 1：完毕；最近一次发送请求被成功处理 0：未完毕；当前发送请求未处理完毕 无论何时发送请求位被置为‘1’，发送完毕位都会被置为‘0’(未完毕)。发送完毕位的‘0’会一直保持到信息被成功发送。
2	TBS	rw	0x00	发送缓冲器状态 (Transmit buffer status) 1：释放；CPU 可以向发送缓冲器写信息 0：锁定；CPU 不能访问发送缓冲器；有信息正在等待发送或正在发送 如果 CPU 在发送缓冲器状态位是 0(锁定) 时试图写发送缓冲器，则写入的字节被拒绝接收且会在无任何提示的情况下丢失。

Bit	Field	Type	Reset	Description
1	DOS	rw	0x00	数据溢出状态 (Data overrun status) 1：溢出；信息丢失，因为 RXFIFO 中没有足够的空间来存储它 0：空缺；自从最后一次清除数据溢出命令执行无数据溢出发生 当要被接收的信息成功的通过验收滤波器后 (例如，仲裁后之初)，CAN 控制器需要在 RXFIFO 中用一些空间来存储这条信息的描述符。因此必须有足够的空间来存储接收的每一个数据字节。如果没有足够的空间存储信息，信息将会丢失且只向 CPU 提示数据溢出情况。如果这个接收到的信息除了最后一位之外都无错误，信息有效。
0	RBS	rw	0x00	接收缓冲器状态 (Receive buffer status) 1：满；RXFIFO 中有可用信息 0：空；无可用信息 在读 RXFIFO 中的信息且用释放接收缓冲器命令来释放内存空间之后，这一位被清除。如果 FIFO 中还有可用信息，此位将在下一位的时限 (t_{SCL}) 中被重新设置。

17.6.5 CAN 中断寄存器 (CAN_IR)

偏移地址：0x0C

复位值：BasicCAN 模式：0x0000 00E0

PeliCAN 模式：0x0000 0000

中断寄存器允许中断源的识别。当寄存器的一位或多位被置位时中断就被激活了。中断寄存器对微控制器来说是只读存储器。

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.								BEI	ALI	EPI	Res.	DOI	EI	TI	RI
								r	r	r		r	r	r	r

Bit	Field	Type	Reset	Description
31 : 8	Reserved			保留，始终读为 0。
7	BEI	r		Basic 模式： 保留，读出值为 1 PeliCAN 模式： BEI：总线错误中断 (Bus error interrupt) 1：置位；当 CAN 控制器检测到总线错误且中断使能寄存器中的 BEIE 被置位时此位被置位 0：复位

Bit	Field	Type	Reset	Description
6	ALI	r		Basic 模式： 保留，读出值为 1 PeliCAN 模式： ALI：仲裁丢失中断 (Arbitration lost interrupt) 1：置位；当 CAN 控制器丢失仲裁，变为接收器和中断使能寄存器的 ALIE 为被置位时，此位被置位 0：复位
5	EPI	r		Basic 模式： 保留，读出值为 1 PeliCAN 模式： EPI：错误消极中断 (Error passive interrupt) 1：置位；当 CAN 控制器到达错误消极状态 (至少一个错误计数器超过协议规定的值 127) 或从错误消极状态又进入错误活动状态以及中断寄存器的 EPIE 位被置位时此位被置‘1’ 0：复位
4	Reserved			保留，始终读为 0。
3	DOI	r		数据溢出中断 (Data overrun interrupt) 1：设置；当数据溢出中断使能位被置为‘1’时向数据溢出状态位‘0-1’跳变，此位被置位 0：复位；微控制器的任何读访问将清除此位 溢出中断位 (中断允许情况下) 和溢出状态位是同时被置位的。
2	EI	r		错误中断 (Error interrupt) 1：置位；错误中断使能时，错误状态位或总线状态位的变化会置位此位 0：复位；微控制器的任何读访问将清除此位
1	TI	r		发送中断 (Transmit interrupt) 1：置位；发送缓冲器状态从 0 变为 1(释放) 和发送中断使能时，置位此位 0：复位；微控制器的任何读访问将清除此位
0	RI	r		接收中断 (Receive interrupt) 1：置位；当接收 FIFO 不空和接收中断使能时置位此位 0：复位；微控制器的任何读访问将清除此位 接收中断位 (中断允许时) 和接收缓冲器状态位是同时置位的。 必须说明的是接收中断位在读的时候被清除，即使 FIFO 中还有其它可用信息。当释放接收缓冲器命令执行后，接收缓冲器中还有其它可用信息，接收中断 (中断允许时) 会在下一个 t_{SCL} 被重置。

17.6.6 CAN 中断使能寄存器 (CAN_IER)

仅存在 PeliCAN 模式

偏移地址：0x10

复位值：0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.								BEIE	ALIE	EPIE	Res.	DOIE	EIE	TIE	RIE
								rw	rw	rw		rw	rw	rw	rw

Bit	Field	Type	Reset	Description
31 : 8	Reserved			保留，始终读为 0。
7	BEIE	rw	0x00	总线错误中断使能 (Bus error interrupt enable) 1：使能；如果检测到总线错误，则 CAN 控制器请求相应的中断 0：禁止
6	ALIE	rw	0x00	仲裁丢失中断使能 (Arbitration lost interrupt enable) 1：使能；如果 CAN 控制器已丢失了仲裁，则请求相应的中断 0：禁止
5	EPIE	rw	0x00	错误消极中断使能 (Error passive interrupt enable) 1：使能；若 CAN 控制器的错误状态改变 (从消极到活动或反之)，则请求相应的中断 0：禁止
4	Reserved			保留，始终读为 0。
3	DOIE	rw	0x00	数据溢出中断使能 (Data overrun interrupt enable) 1：使能；如果数据溢出状态位被置位 (见状态寄存器)，CAN 控制器请求相应的中断 0：禁止
2	EIE	rw	0x00	错误报警中断使能 (Error interrupt enable) 1：使能；如果错误或总线状态改变 (见状态寄存器)，CAN 控制器请求相应的中断 0：禁止
1	TIE	rw	0x00	发送中断使能 (Transmit interrupt enable) 1：使能；当信息被成功发送或发送缓冲器又可访问 (例如，中止发送命令后) 时，CAN 控制器请求相应的中断 0：禁止
0	RIE	rw	0x00	接收中断使能 (Receive interrupt enable) 1：使能；当接收缓冲器状态是‘满’时，CAN 控制器请求相应的中断 0：禁止

17.6.7 CAN 验收代码寄存器 (CAN_ACR)

BasicCAN 模式：CAN_ACR

偏移地址：0x10

复位值：0x0000 00XX

在验收滤波器的帮助下，CAN 控制器能够允许 RXFIFO 只接收同识别码和验收滤波器中预设值相一直的信息。验收滤波器通过验收代码寄存器和验收屏蔽寄存器来定义。

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.								AC							
								rw	rw	rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
31：8	Reserved			保留，始终读为 0。
7：0	AC	rw	0xXX	(Acceptance code) 复位请求位被置高 (当前) 时，这个寄存器是可以访问 (读/写) 的。如果一条信息通过了验收滤波器的测试而且接收缓冲器有空间，那么描述符和数据将被分别顺次写入 RXFIFO。当信息被正确的接收完毕就会： 接收状态位置高 (满) 接收中断使能位置高 (使能) 接收中断置高 (产生中断) 验收代码位 (AC.7-AC.0) 和信息识别码的高 8 位 (ID.10-ID.3) 相等，且与验收屏蔽位 (AM.7-AM.0) 的相应位相或为 1。即如果满足以下方程的描述，则被接收： $[(ID.10-ID.3)\equiv(AC.7-AC.0)]\vee(AM.7-AM.0)\equiv11111111$

PeliCAN 模式：有四个验收代码寄存器分别是 CAN_ACR0，CAN_ACR1，CAN_ACR2，CAN_ACR3

CAN_ACR0：偏移地址：0x40 复位值：0x0000 00XX

CAN_ACR1：偏移地址：0x44 复位值：0x0000 00XX

CAN_ACR2：偏移地址：0x48 复位值：0x0000 00XX

CAN_ACR3：偏移地址：0x4C 复位值：0x0000 00XX

注：详细说明见的标识符过滤中 peliCAN 模式介绍。

17.6.8 CAN 验收屏蔽寄存器 (CAN_AMR)

BasicCAN 模式：CAN_AMR

偏移地址：0x14

复位值：0x0000 00XX

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.								AM							
								rw	rw	rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
31 : 8	Reserved			保留，始终读为 0。
7 : 0	AM	rw	0xXX	(Acceptance mask) 如果复位请求位置高 (当前) 这个寄存器可以被访问 (读/写)。验收屏蔽寄存器定义验收代码寄存器的相应位对验收滤波器是‘相关的’或‘无影响的’(即可为任意值)。

PeliCAN 模式：有四个验收屏蔽寄存器分别是 CAN_AMR0，CAN_AMR1，CAN_AMR2，CAN_AMR3

CAN_AMR0：偏移地址：0x50

复位值：0x0000 00XX

CAN_AMR1：偏移地址：0x54

复位值：0x0000 00XX

CAN_AMR2：偏移地址：0x58

复位值：0x0000 00XX

CAN_AMR3：偏移地址：0x5C

复位值：0x0000 00XX

注：详细说明见的标识符过滤中 peliCAN 模式介绍。

17.6.9 CAN 总线定时 0(CAN_BTR0)

偏移地址：0x18

复位值：0x0000 0000

总线定时寄存器 0 定义了波特率预设值 (BRP) 和同步跳转宽度 (SJW) 的值。复位模式有效时这个寄存器是可以被访问 (读/写) 的。

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.								SJW		BRP					
								rw	rw	rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
31 : 8	Reserved			保留，始终读为 0。
7 : 6	SJW	rw	0x00	同步跳转宽度 (Synchronization jump width) 为了补偿在不同总线控制器的时钟振荡器之间的相位偏移，任何总线控制器必须在当前传送的相关信号边沿重新同步。同步跳转宽度定义了每一位周期可以被重新同步缩短或延长的时钟周期的最大数目。 $t_{SJW} = t_{SCL} \times (SJW + 1)$
5 : 0	BRP	rw	0x00	波特率预设值 (Baud rate prescaler) CAN 系统时钟 t_{SCL} 的周期是可编程的而且决定了相应的位时序。CAN 系统时钟由如下公式计算 $t_{SCL} = 2 \times t_{CLK} \times (BRP + 1)$ 这里 $t_{CLK} = APB1$ 的时钟周期

17.6.10 CAN 总线定时 1(CAN_BTR1)

偏移地址：0x1C

复位值：0x0000 0000

总线定时寄存器 1 定义了每个位周期的长度、采样点的位置和在每个采样点的采样数目。在复位模式中，这个寄存器可以被读/写访问。

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.								SAM	TESG2			TESG1			
								rw	rw	rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
31 : 8	Reserved			保留，始终读为 0。
7	SAM	rw	0x00	采样 (Sampling) 1：三倍；总线采样三次；建议在低/中速总线 (A 和 B 级) 上使用，这对过滤总线上的毛刺波是有益的 0：单倍；总线采样一次；建议使用在高速总线上 (SAE C 级)
6 : 0	TSEG2 TSEG1	rw	0x00	时间段 1(Time segment 1) 和时间段 2(Time segment 2) TSEG1 和 TSEG2 决定了每一位的时钟数目和采样点的位置，这里： $t_{SYNCSEG}$ = 1 x t_{SCL} t_{TSEG1} = t_{SCL} × (8 × TSEG1.3 + 4 × TSEG1.2 + 2 × TSEG1.1 + TSEG1.0 + 1) t_{TSEG2} = t_{SCL} × (4 × TSEG2.2 + 2 × TSEG2.1 + TSEG2.1 + 1)

17.6.11 CAN 发送识别码寄存器 0(CAN_TXID0)

在 BasicCAN 模式：

偏移地址：0x28

复位值：0x0000 00XX

注：复位模式为 0xFF

发送识别码寄存器 0 定义发送帧的类型与数据长度。仅在工作模式下这个寄存器可以被读/写访问。

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.								ID10	ID9	ID8	ID7	ID6	ID5	ID4	ID3
								rw	rw	rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
31：8	Reserved			保留，始终读为 0。
7：0	IDx	rw	0xXX	CAN 识别码 10 ~ 3(CAN identifier byte 10 ~ 3) 注：在 Peli 模式下，位 [3：0] 为 DLC[3：0]

17.6.12 CAN 发送识别码寄存器 1(CAN_TXID1)

仅存在 BasicCAN 模式：

偏移地址：0x2C

复位值：0x0000 00XX

注：复位模式为 0xFF

发送识别码寄存器 1 定义发送帧的类型与数据长度。仅在工作模式下这个寄存器可以被读/写访问。

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.								ID2	ID1	ID0	RTR	DLC3	DLC2	DLC1	DLC0
								rw	rw	rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
31：8	Reserved			保留，始终读为 0。
7：5	IDx	rw	0x0X	CAN 识别码 2 ~ 0(CAN identifier byte 2 ~ 0)
4	RTR	rw	0x0X	帧格式 (Remote transmission request) 1：远程；CAN 将发送远程帧 0：数据；CAN 将发送数据帧

Bit	Field	Type	Reset	Description
3 : 0	DLCx	rw	0x0X	发送数据区长度 0 ~ 8(Data length code 0 ~ 8)

仅存在 BasicCAN 模式：

偏移地址：0x30 ~ 0x4C

复位值：0x0000 0000

发送数据寄存器 CAN_TXDR0 ~ 7。仅在工作模式下这个寄存器可以被读/写访问。接收缓冲与发送缓冲数据格式一致。

17.6.13 CAN 仲裁丢失捕捉寄存器 (CAN_ALC)

仅存在 PeliCAN 模式：

偏移地址：0x2C

复位值：0x0000 0000

Bit	Field	Type	Reset	Description
31 : 5	Reserved			保留，始终读为 0。
4 : 0	BITNO	rw	0x00	值和功能参考下表 (Bit number) 仲裁丢失时，会产生相应的仲裁丢失中断 (中断允许)。同时，位流处理器的当前位位置被捕捉送入仲裁丢失捕捉寄存器。一直到用户通过软件读这个值，寄存器中的内容都不会变。随后，捕捉机制又被激活了。 读中断寄存器时，中断寄存器中相应的中断标志位被清除。直到仲裁丢失捕捉寄存器被读一次之后，新的仲裁丢失中断才有效。

表 66. 仲裁丢失捕捉寄存器的 bit 4 - bit 0 的功能

位					十进制值	功能
ALC.4	ALC.3	ALC.2	ALC.1	ALC.0		
0	0	0	0	0	0	仲裁丢失在识别码的 bit1
0	0	0	0	1	1	仲裁丢失在识别码的 bit2
0	0	0	1	0	2	仲裁丢失在识别码的 bit3
0	0	0	1	1	3	仲裁丢失在识别码的 bit4
0	0	1	0	0	4	仲裁丢失在识别码的 bit5

位					十进制值	功能
ALC.4	ALC.3	ALC.2	ALC.1	ALC.0		
0	0	1	0	1	5	仲裁丢失在识别码的 bit6
0	0	1	1	0	6	仲裁丢失在识别码的 bit7
0	0	1	1	1	7	仲裁丢失在识别码的 bit8
0	1	0	0	0	8	仲裁丢失在识别码的 bit9
0	1	0	0	1	9	仲裁丢失在识别码的 bit10
0	1	0	1	0	10	仲裁丢失在识别码的 bit11
0	1	0	1	1	11	仲裁丢失在 SRTR 位；注 2
0	1	1	0	0	12	仲裁丢失在 IDE 位
0	1	1	0	1	13	仲裁丢失在识别码的 bit12；注 3
0	1	1	1	0	14	仲裁丢失在识别码的 bit13；注 3
0	1	1	1	1	15	仲裁丢失在识别码的 bit14；注 3
1	0	0	0	0	16	仲裁丢失在识别码的 bit15；注 3
1	0	0	0	1	17	仲裁丢失在识别码的 bit16；注 3
1	0	0	1	0	18	仲裁丢失在识别码的 bit17；注 3
1	0	0	1	1	19	仲裁丢失在识别码的 bit18；注 3
1	0	1	0	0	20	仲裁丢失在识别码的 bit19；注 3
1	0	1	0	1	21	仲裁丢失在识别码的 bit20；注 3
1	0	1	1	0	22	仲裁丢失在识别码的 bit21；注 3
1	0	1	1	1	23	仲裁丢失在识别码的 bit22；注 3
1	1	0	0	0	24	仲裁丢失在识别码的 bit23；注 3
1	1	0	0	1	25	仲裁丢失在识别码的 bit24；注 3
1	1	0	1	0	26	仲裁丢失在识别码的 bit25；注 3
1	1	0	1	1	27	仲裁丢失在识别码的 bit26；注 3
1	1	1	0	0	28	仲裁丢失在识别码的 bit27；注 3
1	1	1	0	1	29	仲裁丢失在识别码的 bit28；注 3
1	1	1	1	0	30	仲裁丢失在识别码的 bit29；注 3
1	1	1	1	1	31	仲裁丢失在 RTR 位；注 3

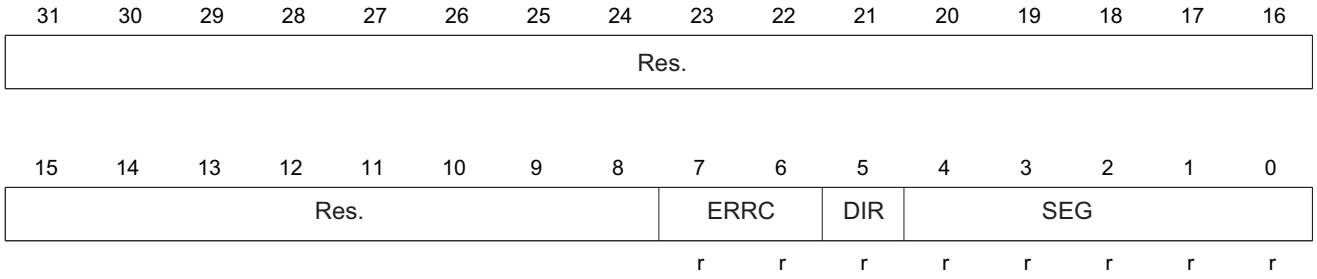
注：仲裁丢失的二进制编码结构位的号码。标准帧信息的 RTR 位。只使用于扩展帧信息。

17.6.14 CAN 错误代码捕捉寄存器 (CAN_ECC)

仅存在 PeliCAN 模式：

偏移地址：0x30

复位值：0x0000 0000



Bit	Field	Type	Reset	Description
31 : 8	Reserved			保留，始终读为 0。
7 : 6	ERRC	r	0x00	错误代码 (Error code) 00：位错 01：格式错 10：填充错 11：其它错误
5	DIR	r	0x00	方向 (Direction) 1：RX；接收时发生的错误 0：TX；发送时发生的错误
4 : 0	SEG	r	0x00	段 (Segment) 00010：ID.28-ID.21 00011：帧开始 00100：SRTR 位 00101：IDE 位 00110：ID.20-ID.18 00111：ID.17-ID.13 01000：CRC 序列 01001：保留位 0 01010：数据区 01011：数据长度代码 01100：RTR 位 01101：保留位 1 01110：ID.4-ID.0 01111：ID.12-ID.5 10001：活动错误标志 10010：中止 10011：支配 (控制) 位误差 10110：消极错误标志 10111：错误定义符 11000：CRC 定义符 11001：应答通道 11010：帧结束 11011：应答定义符 11100：溢出标志 其他：保留

注：位的设置反映了当前结构段的不同错误事件。

总线发生错误时被迫产生相应的错误中断 (中断允许时)。同时，位流处理器的当前位置被捕捉送入错误代码捕捉寄存器。其内容直到用户通过软件读出时都是不变的。读出后，捕捉机制又被激活了。访问中断寄存器期间，中断寄存器中相应的中断标志位被清除。新的总线中断直到捕捉寄存器被读出一次才可能有效。

17.6.15 CAN 错误报警限制寄存器 (CAN_EWLR)

仅存在 PeliCAN 模式：

偏移地址：0x34

复位值：0x0000 0096

错误报警限制在这个寄存器中被定义。默认值 (复位值) 是 0096 。复位模式中，此寄存器对 CPU 来说是可读/写的。工作模式中是只读的。

注：只有之前进入复位模式，EWLR 才有可能被改变。直到复位模式被再次取消后，才有可能发生错误状态的改变 (见状态寄存器) 和由新的寄存器内容引起的错误报警中断。

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.								EWL							
								rw	rw	rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
31 : 8	Reserved			保留，始终读为 0。
7 : 0	EWL	rw	0x96	可编程的错误限制报警 (Programmable error warning limit) 当只要一个错误计数器超过错误限制编程值，错误状态位被置位。

17.6.16 CAN RX 错误计数寄存器 (CAN_RXERR)

仅存在 PeliCAN 模式：

偏移地址：0x38

复位值：0x0000 0000

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

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

Bit	Field	Type	Reset	Description
31 : 8	Reserved			保留，始终读为 0。
7 : 0	RXERR	rw	0x00	RX 错误计数寄存器 (RX error counter register) 反应了接收错误计数器的当前值。硬件复位后寄存器被初始化为 0。工作模式中，对 CPU 来说是只读的。只有在复位模式中才可以写访问此寄存器。 如果发生总线关闭，RX 错误计数器就被初始化为 0。总线关闭期间，写这个寄存器是无效的。 注意：只有之前进入复位模式，才有可能由 CPU 迫使 RX 错误计数器发生改变。直到复位模式被取消后，错误状态的改变 (见状态寄存器)、错误报警和由新的寄存器内容引起的错误中断才可能有效。

17.6.17 CAN TX 错误计数寄存器 (CAN_TXERR)

仅存在 PeliCAN 模式：

偏移地址：0x3C

复位值：0x0000 0000

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

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

Bit	Field	Type	Reset	Description
31 : 8	Reserved			保留，始终读为 0。

Bit	Field	Type	Reset	Description
7 : 0	TXERR	rw	0x00	TX 错误计数寄存器 (TX error counter register) 反映了发送错误计数器的当前值。 工作模式中，这个寄存器对 CPU 是只读内存。复位模式中才可以写访问这个寄存器。硬件复位后，寄存器被初始化为 0。如果总线关闭，TX 错误计数器被初始化为 127 来计算总线定义的最小时间 (128 个总线空闲信号)。这段时间里读 TX 错误计数器将反映出总线关闭恢复的状态信息。 如果总线关闭是激活的，写访问 TXERR 的 0 ~ 254 单元会清除总线关闭标志，复位模式被清除后控制器会等待一个 11 位的连续隐藏 (弱势) 位 (总线空闲)。 向 TXERR 写入 255 会初始化 CPU 驱动的总线关闭事件。只有之前进入复位模式，才有可能发生 CPU 引起的 TX 错误计数器内容的改变。直到复位模式被再次取消，错误或总线状态的改变 (见状态寄存器)、错误报警和由新的寄存器内容引起的错误中断才有可能有效。离开复位模式后，就象总线错误引起的一样，给出新的 TX 计数器内容且总线关闭被同样的执行。这意味着重新进入复位模式，TX 错误计数器被初始化到 127，RX 计数器被清 0，所有的相关状态和中断寄存器位被置位。 复位模式的清除将会执行协议规定的总线关闭恢复序列 (等待 128 个总线空闲信号)。 如果在总线关闭恢复 (TXERR > 0) 之前又进入复位模式，总线关闭保持有效且 TXERR 被锁定。

17.6.18 CAN 发送帧信息寄存器 (CAN_SFF)

仅存在 PeliCAN 模式：

偏移地址：0x40

复位值：0x0000 00XX

发送帧信息寄存器设置送帧的类型与数据长度。仅在工作模式下这个寄存器可以被读/写访问，写时为发送寄存器，读时为接收寄存器，数据结构相同。

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.								FF	RTR	Res.		DLC.3	DLC.2	DLC.1	DLC.0
								rw	rw			rw	rw	rw	rw

Bit	Field	Type	Reset	Description
31 : 8	Reserved			保留，始终读为 0。

Bit	Field	Type	Reset	Description
7	FF	rw	0x0X	帧格式 (Frame format) 1：EFF；CAN 将发送/接收扩展帧格式 0：SFF；CAN 将发送/接收标准帧格式
6	RTR	rw	0x0X	帧格式 (Remote transmission request) 1：远程；CAN 将发送/接收远程帧 0：数据；CAN 将发送/接收数据帧
5：4	Reserved			保留。
3：0	DLC	rw	0x0X	数据区长度 (Data length code bit) 发送数据区长度 0 ~ 8。

17.6.19 CAN 发送识别码寄存器 0(CAN_TXID0)

仅存在 PeliCAN 模式：

偏移地址：0x44

复位值：0x0000 00XX

发送识别码寄存器 0 设置帧的标识符。仅在工作模式下这个寄存器可以被读/写访问，写时为发送寄存器，读时为接收寄存器，数据结构相同。

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.								ID28	ID27	ID26	ID25	ID24	ID23	ID22	ID21
								rw	rw	rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
31：8	Reserved			保留，始终读为 0。
7：0	IDx	rw	0xXX	CAN 标识符 ID28 ~ ID21(Identifier bit 28-21) 标准帧时，与 ID20 ~ ID18 组成 11 位标识符。

17.6.20 CAN 发送识别码寄存器 1(CAN_TXID1)

仅存在 PeliCAN 模式：

偏移地址：0x48

复位值：0x0000 00XX

发送识别码寄存器 1 设置帧的标识符。仅在工作模式下这个寄存器可以被读/写访问，写时为发送寄存器，读时为接收寄存器，数据结构相同。

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.								ID20	ID19	ID18	ID17	ID16	ID15	ID14	ID13
								rw	rw	rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
31 : 8	Reserved			保留，始终读为 0。
7 : 5	IDx	rw	0x0X	CAN 标识符 ID20 ~ ID18(Identifier bit 20-18)
4 : 0	IDx	rw	0x0X	标准帧: 无意义 扩展帧: CAN 标识符 ID17 ~ ID13(Identifier bit 17-13)

17.6.21 CAN 发送数据寄存器 0(CAN_TXDATA0)

仅存在 PeliCAN 模式：

偏移地址：0x4C

复位值：0x0000 00XX

发送数据寄存器 0，用于 CAN 数据发送存储。仅在工作模式下这个寄存器可以被读/写访问，写时为发送寄存器，读时为接收寄存器，数据结构相同。

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.								DATA0							
								rw	rw	rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
31 : 8	Reserved			保留，始终读为 0。
7 : 0	DATA0	rw	0xXX	CAN 发送数据 0(CAN transmit data 0) 扩展帧 ID12 ~ 5

17.6.22 CAN 发送数据寄存器 1(CAN_TXDATA1)

仅存在 PeliCAN 模式：

偏移地址：0x50

复位值：0x0000 00XX

发送数据寄存器 1，用于 CAN 数据发送存储。仅在工作模式下这个寄存器可以被读/写访问，写时为发送寄存器，读时为接收寄存器，数据结构相同。

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.								DATA1							
								rw	rw	rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
31 : 8	Reserved			保留，始终读为 0。
7 : 0	DATA1	rw	0xXX	CAN 发送数据 1(CAN transmit data 1) 扩展帧 ID4 ~ 0，低 3 位无意义

仅存在 PeliCAN 模式：

偏移地址：0x54 ~ 0x70

复位值：0x0000 00XX

以上偏移地址均为 CAN 数据寄存器，扩展帧时 CAN 发送数据 0 储存在 DATA2 余下数据依次类推。这些寄存器写时为发送寄存器，读时为接收寄存器，数据结构相同。

17.6.23 CAN 时钟分频寄存器 (CAN_CDR)

偏移地址：0x7C

复位值：0x0000 0000

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

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

Bit	Field	Type	Reset	Description
31 : 8	Reserved			保留，始终读为 0。
7	MODE	rw	0x00	CAN 模式 (CAN mode) 如果 MODE 是 0，CAN 控制器工作于 BasicCAN 模式。否则，CAN 控制器工作于 PeliCAN 模式。只有在复位模式中是可以写的。
6 : 0	Reserved			保留，始终读为 0。

18 串行外设接口 (SPI)

串行外设接口 (SPI)

18.1 SPI 简述

SPI 接口广泛用于不同设备之间的板级通讯，如扩展串行 Flash，ADC 等。许多 IC 制造商生产的器件都支持 SPI 接口。

SPI 允许 MCU 与外部设备以全双工、同步、串行方式通信。应用软件可以通过查询状态或 SPI 中断来通信。

18.2 主要特征

- 完全兼容 Motorola 的 SPI 规格
- 支持 DMA 请求
- 在 3 根线上支持全双工同步传输
- 16 位的可编程波特率生成器
- 支持主机模式和从机模式
- 8 个字节的接收/发送 FIFO
- SPI 作为主机模式下 SPI 的时钟最快可高达 pclk/2（pclk 为 APB 时钟），作为从机模式下 SPI 的时钟最快可高达 pclk/4
- 可编程的时钟极性和相位
- 可编程的数据顺序，MSB 在前或者 LSB 在前
- 支持一个主机多个从机操作
- 支持 1 ~ 32 位的数据位长度同时发送和接收
- 除了 8 位数据收发，其余 1 ~ 32 位数据收发只支持 LSB 模式，不支持 MSB 模式。
- 支持各 8 个对应配置数据位（Data size）的发送缓冲器和接收缓冲器
- 中断驱动操作
  - 发送缓冲为空
  - 发送缓冲和发送移位寄存器同时为空
  - 接收缓冲为满
  - 接收缓冲上溢
  - 主模式下接收到指定字节数
  - 发送端下溢
  - 接收到有效字节

18.3 SPI 功能描述

18.3.1 概述

SPI 的方框图见下图

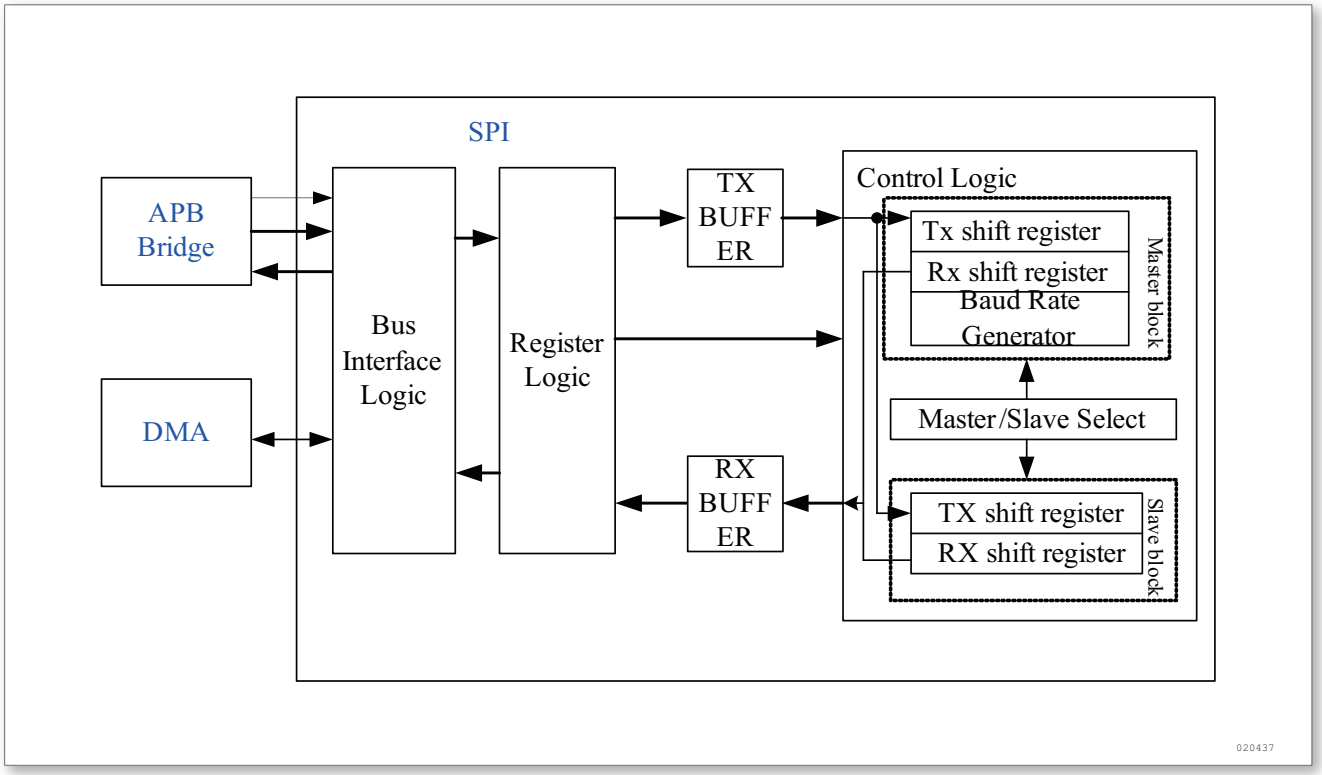


图 145. SPI 框图

SPI 支持接收和发送 1 ~ 32 位数据同时进行。SPI 可以被配置为从模式或者在一个主机环境下配置为主模式。可以通过配置时钟极性 CPOL 和相位 CPHA 选择四种可能的时序关系。可编程的数据顺序，MSB 在前或者 LSB 在前。

发送和接收部分使用相同的时钟。数据在时钟的上升沿或者下降沿输出，在 SCLK 相反的有效沿锁存数据。因为 SPI 是用于交换数据，因此数据必须在转移结束后读取，即使数据不是有效数据。在 SPI 模式下，主机和与其通信的从机的时钟相位和极性必须相同。

通常 SPI 通过 4 个管脚与外部器件相连：

- MISO：主设备输入/从设备输出管脚。该管脚在从模式下发送数据，在主模式下接收数据。
- MOSI：主设备输出/从设备输入管脚。该管脚在主模式下发送数据，在从模式下接收数据。
- SCK：串口时钟，作为主设备的输出，从设备的输入。
- NSS：从设备选择。这是一个可选的管脚，用来选择主/从设备。它的功能是用来作为‘片选管脚’，让主设备可以单独地与特定从设备通讯，避免数据线上的冲突。从设备的 NSS 管脚可以由主设备当作一个标准的 IO 来驱动。一旦被使能，NSS 管脚也可以作为输出管脚，并在 SPI 设置为主模式时拉低；此时，所有 NSS 管脚连接到主设备 NSS 管脚的 SPI 设备，会检测到低电平。

下图是一个单主和单从设备互连的例子。

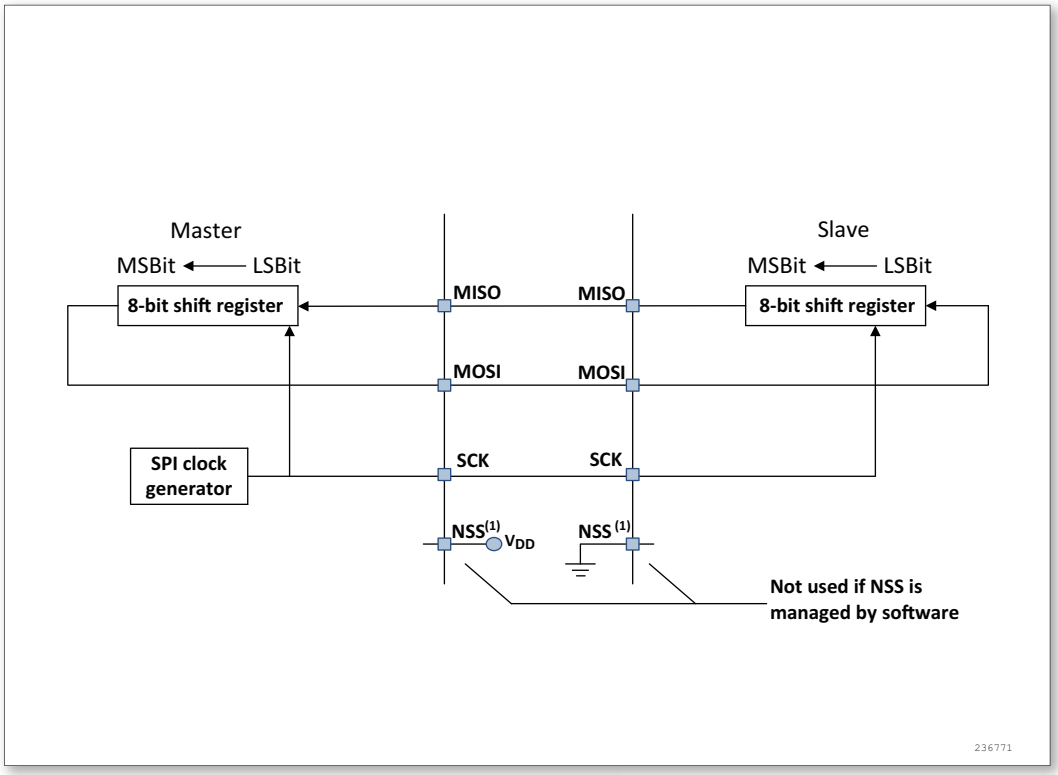


图 146. 单主和单从应用

MOSI 脚相互连接，MISO 脚相互连接。这样，数据在主和从之间串行地传输（MSB 位在前）。

通信总是由主设备发起。主设备通过 MOSI 脚把数据发送给从设备，从设备通过 MISO 引脚回传数据。这意味全双工通信的数据输出和数据输入是用同一个时钟信号同步的；时钟信号由主设备通过 SCK 脚提供。

时钟信号的相位和极性

SPI_CCTL 寄存器的 CPOL 和 CPHA 位，能够组合成四种可能的时序关系。CPOL（时钟极性）位控制无数据传输期间 SCK 时钟的空闲状态电平，此位对主模式和从模式下的设备都有效。如果 CPOL 被清‘0’，SCK 引脚在空闲状态保持低电平，即两次传输之间为低电平；如果 CPOL 被置‘1’，SCK 引脚在空闲状态保持高电平，即两次传输之间为高电平。

如果 CPHA（时钟相位）位被置‘1’，第一个数据位在 SCK 时钟的第一个时钟边沿被锁存（CPOL 位为 1 时就是下降沿，CPOL 位为 0 时就是上升沿），同时对被接收的第一个数据位进行采样。SPI 在传输的第一个 SCK 时钟转换时改变串行数据（此时时钟向空闲状态的反方向变动），在下一个边沿捕捉数据。

如果 CPHA（时钟相位）位被清‘0’，第一个数据位在 SCK 时钟的第二个时钟边沿被锁存（CPOL 位为 0 时就是下降沿，CPOL 位为 1 时就是上升沿），同时对被接收的第一个数据位进行采样。SPI 在传输的第一个 SCK 时钟转换时捕捉串行数据（此时时钟向空闲状态的反方向变动），数据在下一个边沿改变。

CPOL 时钟极性和 CPHA 时钟相位的组合选择数据捕捉的时钟边沿。图 147显示了 SPI 传输的 4 种 CPHA 和 CPOL 位组合。此图可以解释为主设备和从设备的 SCK 脚、MISO 脚、MOSI 脚直接连接的主或从时序图。

高速传输

针对高速传输模式下对板级延时的敏感，在 SPI_CCTL 寄存器中由 TXEDGE 和 RXEDGE 控制位对发送相位和接收采样进行时间调整。

- 在从模式下，TXEDGE 为 1 时，发送数据立即发送到数据总线，用于高速模式时（SPBRG = 4）；为 0 时，发送数据在一个有效时钟边沿后发送到数据总线，用于低速模式时（SPBRG ＞ 4）。
- 在主模式下，RXEDGE 为 1 时，在传输数据位的中间采样数据；为 0 时，在传输数据位的尾时钟沿采样数据（用于高速模式）

1. 在改变 CPOL/CPHA 位之前，必须清除 SPIEN 位将 SPI 禁止。
2. 主和从必须配置成相同的时序模式。
3. SCK 的空闲状态必须和 SPI_CCTL 寄存器指定的极性一致（CPOL 为 1 时，空闲时应上拉 SCK 为高电平；CPOL 为 0 时，空闲时应下拉 SCK 为低电平）。

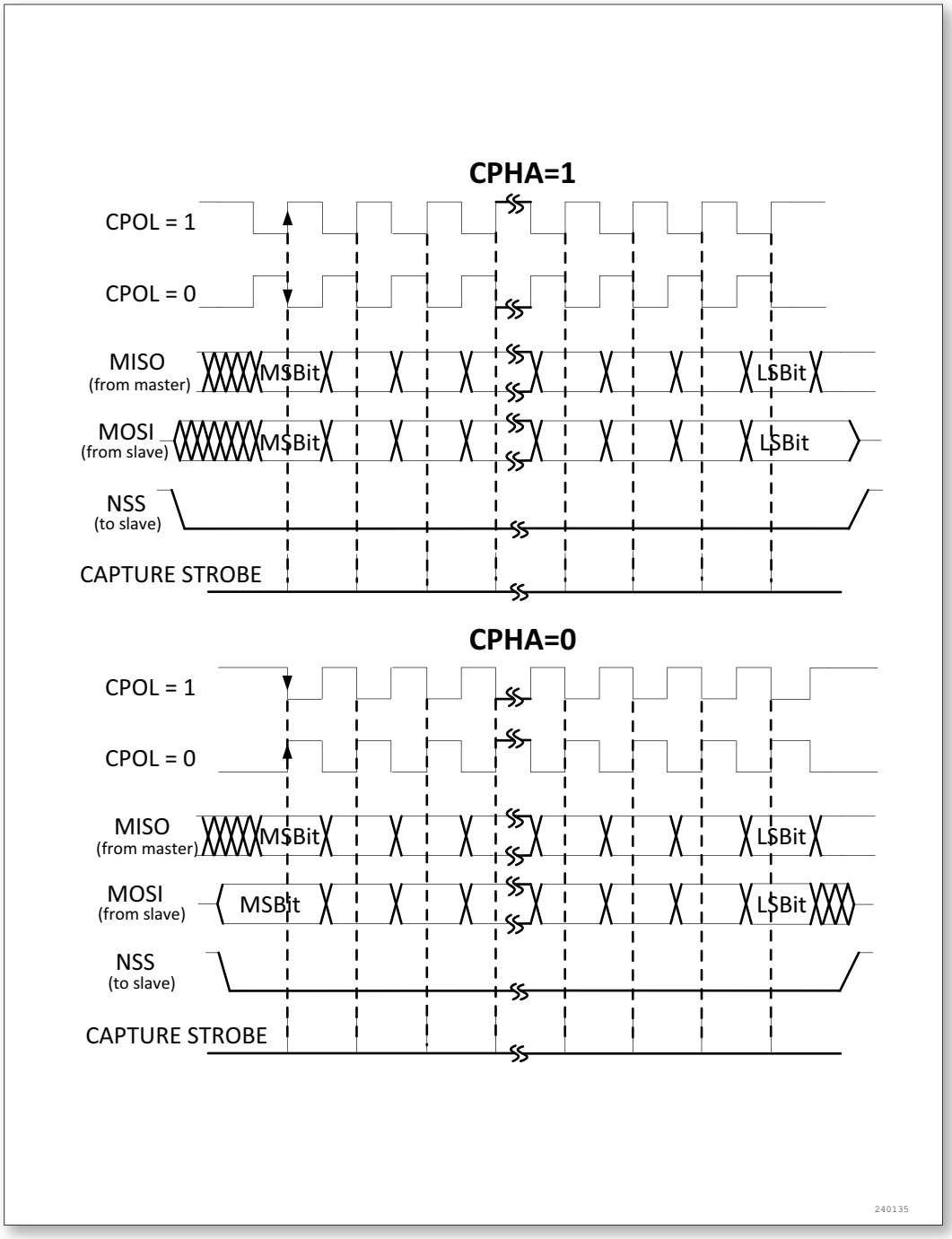


图 147. 数据时钟时序图

数据帧格式

根据 SPI_CCTL 寄存器中的 LSBFE 位，输出数据位时可以 MSB 在先也可以 LSB 在先。根据 SPI_CCTL 寄存器的 SPILEN 位，每个数据帧可以是 7 位或者 8 位。所选择的数据帧格式对发送和/或接收都有效。

另外设置寄存器 SPI_EXTCTL，可以配置数据帧长度为 1 ~ 32 位。使用此配置时需要配置：SPI_GCTL 寄存器的 DW8_32 位为‘0’，且 SPI_CCTL 寄存器的 LSBFE 位配置为‘1’，SPILEN 位配置为‘1’。在配合 DMA 数据传输是需要将 DMA 的数据长度配置为 8bit。

18.3.2 SPI 从模式

在从配置里，SCK 引脚用于接收到从主设备来的串行时钟。SPI_SPBRG 寄存器中的设置不影响数据传输速率。

配置步骤

1. 设置 SPILEN 位以定义数据帧格式为 7 位或者 8 位
2. 选择 CPOL 和 CPHA 位来定义数据传输和串行时钟之间的相位关系。为保证正确的数据传输，从设备和主设备的 CPOL 和 CPHA 位必须配置成相同的方式。
3. 帧格式（MSB 在前还是 LSB 在前取决于 SPI_CCTL 寄存器中的 LSBFE 位）必须和主设备相同。
4. 清除 MDOE 位，设置 SPIEN 位，使相应引脚工作于 SPI 模式下。在这个配置里，MOSI 引脚是数据输入，MISO 引脚是数据输出。

数据发送过程

在写操作中，数据字被并行地写入发送缓冲器。

当从设备收到时钟信号，并且在 MOSI 引脚上出现第一个数据位时，发送过程开始，第一个位被发送出去。余下的位被装进移位寄存器。当发送缓冲器中的数据传输到移位寄存器时，SPI_INTSTAT 寄存器里的 TX_INTF 标志被设置。如果设置了 SPI_INTEN 寄存器上的 TXIEN 位，将会产生中断。

数据接收过程

对于接收方，当数据接收完成时：

- 移位寄存器中的数据传送到接收缓冲器，SPI_INTSTAT 寄存器中的 RX_INTF 标志被设置。
- 如果设置了 SPI_INTEN 寄存器中的 RXIEN 位，则产生中断。

在最后一个采样时钟边沿后，RXNE 位被置‘1’，移位寄存器中接收到的数据字节被传送到接收缓冲器。当读 SPI_RXREG 寄存器时，SPI 设备返回这个值。

18.3.3 SPI 主模式

在主配置时，串行时钟在 SCK 脚产生。

配置步骤

1. 通过 SPI_SPBRG 寄存器定义串行时钟波特率。
2. 选择 CPOL 和 CPHA 位，定义数据传输和串行时钟间的相位关系。
3. 设置 SPILEN 位来定义 8 或 7 位数据帧格式。
4. 配置 SPI_CCTL 寄存器的 LSBFE 位定义帧格式。
5. 如果只接收而不发送数据，配置 SPI_RNDNR 寄存器，定义需要接收的字节数。
6. 必须设置 MDOE 和 SPIEN 位。

在这个配置中，MOSI 脚是数据输出，而 MISO 脚是数据输入，NSS 是从设备选择信号输出。

数据发送过程

当一字节写进发送缓冲器时，发送过程开始。在发送第一个数据位时，数据字被并行地（通

过内部总线）传入移位寄存器，而后串行地移出到 MOSI 脚上；MSB 在先还是 LSB 在先，取决于 SPI_CCTL 寄存器中的 LSBFE 位。数据从发送缓冲器传输到移位寄存器时 TX_INTF 标志将被置位，如果设置 SPI_INTEN 寄存器中的 TXIEN 位，将产生中断。

数据接收过程

对于接收器来说，当数据传输完成时：

- 移位寄存器中的数据传送到接收缓冲器，SPI_INTSTAT 寄存器中的 RX_INTF 标志被设置。
- 如果设置了 SPI_INTEN 寄存器中的 RXIEN 位，则产生中断。

在最后一个采样时钟边沿后，RXNE 位被置‘1’，移位寄存器中接收到的数据字节被传送到接收缓冲器。当读 SPI_RXREG 寄存器时，SPI 设备返回这个值。

如果只接收而不发送数据，在接收完 RXDNR 定义的字节数，RXMATCH_INTF 位被置‘1’，表示所有的数据接收完毕，主模式下不再发送时钟信号。

18.3.4 状态标志

为了软件操作的方便，应用程序可以通过 4 个当前状态标志和 7 个中断状态标志来监控 SPI 总线的状态。当前状态标志是只读，由硬件自动置位和清除。中断状态标志位在事件发生时置位，并在中断使能时产生 CPU 中断，由软件清除。

SPI 内部分别有一个 8 字节的发送缓冲和接收缓冲，根据 SPI_GCTL 的 DW8_32 位的设置，CPU 每次可以读写 1 或者 4 个字节。根据 DW8_32 的设置，发送和接收缓冲分别有一个字节或者一个有效数据的状态标志。

表 67. SPI 状态

分类	状态标志	缓冲器和信号状态
中断状态	TX_INTF	根据 DW8_32 设置，至少有一个有效数据的空间，能完成一次发送数据寄存器的写操作
	RX_INTF	根据 DW8_32 设置，至少有一个有效数据的数据，能完成一次接收数据寄存器的读操作
	UNDERRUN_INTF	发送缓冲器空且重复发送
	RXOERR_INTF	接收缓冲器非空且被覆盖
	RXMATCH_INTF	非空，最后一个数据传送到接收缓冲中
	RXFULL_INTF	接收缓冲器满，不能再接收新的数据
	TXEPT_INTF	发送缓冲器空，不能再发送
当前状态	RXAVL_4BYTE	接收缓冲器有超过 4 字节有效数据
	TXFULL	发送缓冲器满
	TXEPT	发送缓冲器空
	RXAVL	接收缓冲器非空，至少还能接收一个字节

当 SPI_GCTL 寄存器的 TXTLF 为 00 时，发送缓冲器有大于等于 1 个空闲数据空间时 TX_INTF 置位；TXTLF 为 01 时，发送缓冲器有超过一半的空闲空间时 TX_INTF 置位。

当 SPI_GCTL 寄存器的 RXTLF 为 00 时，接收缓冲器有大于等于 1 个有效数据时，RX_INTF 置位；RXTLF 为 01 时，接收缓冲器有超过一半的有效数据时 RX_INTF 置位。

18.3.5 波特率设置

波特率是生成的 SCLK 的频率，一般是 PCLK 的分频。BRG 是一个 16 位的波特率发生器。SPBREG 寄存器控制 16 位计数器的计数周期。

提供期望的波特率和 f_{pclk}（APB 模块的频率），使用下表所示的公式计算出的值近似数赋值给 SPBRG 寄存器。其中下表中的 X 等于 SPBRG 寄存器的值（2 ~ 65535）。

表 68. 波特率公式

模式	公式
SPI 模式	波特率 = f_{pclk}/X

18.3.6 利用 DMA 的 SPI 通信

为了达到最大通信速度，需要及时往 SPI 发送缓冲器填数据，同样接收缓冲器中的数据也必须及时读走以防止溢出。为了方便高速率的数据传输，SPI 实现了一种采用简单的请求/应答的 DMA 机制。

当 SPI_GCTL 寄存器上的 DMAEN 位被设置时，SPI 模块可以发出 DMA 传输数据的请求。发送缓冲器和接收缓冲器的 DMA 请求都由 DMAEN 使能。

- 发送时，当 SPI_GCTL 寄存器的 TXTLF 为 00 时，发送缓冲器有大于等于 1 个空闲数据空间时即进行 DMA 传输请求；TXTLF 为 01 时，发送缓冲器有超过一半的空闲空间时即进行 DMA 请求。每次请求只进行一次 DMA 传输。每次 DMA 传输数据大小以及发送缓冲器每个数据大小由 DW8_32 为决定。
- 接收时，当 SPI_GCTL 寄存器的 RXTLF 为 00 时，接收缓冲器有大于等于 1 个有效数据时即进行 DMA 传输请求；RXTLF 为 01 时，接收缓冲器有超过一半的有效数据时即进行 DMA 请求。每次请求只进行一次 DMA 传输。每次 DMA 传输数据大小以及接收缓冲器每个数据大小由 DW8_32 为决定。

18.4 寄存器堆和存储器映射描述

表 69. SPI 寄存器概览

Offset	Acronym	Register Name	Reset	Section
0x00	SPI_TXREG	发送数据寄存器	0x00000000	小节 18.4.1
0x04	SPI_RXREG	接收数据寄存器	0x00000000	小节 18.4.2
0x08	SPI_CSTAT	当前状态寄存器	0x00000001	小节 18.4.3
0x0C	SPI_INTSTAT	中断状态寄存器	0x00000000	小节 18.4.4
0x10	SPI_INTEN	中断使能寄存器	0x00000000	小节 18.4.5
0x14	SPI_INTCLR	中断清除寄存器	0x00000000	小节 18.4.6
0x18	SPI_GCTL	全局控制寄存器	0x00000004	小节 18.4.7
0x1C	SPI_CCTL	通用控制寄存器	0x00000008	小节 18.4.8
0x20	SPI_SPBRG	波特率发生器	0x00000002	小节 18.4.9
0x24	SPI_RXDNR	接收数据个数寄存器	0x00000001	小节 18.4.10
0x28	SPI_NSSR	从机片选寄存器	0x000000FF	小节 18.4.11

Offset	Acronym	Register Name	Reset	Section
0x2C	SPI_EXTCTL	数据控制寄存器	0x00000008	小节 18.4.12

18.4.1 发送数据寄存器（SPI_TXREG）

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

Bit	Field	Type	Reset	Description
31:0	TXREG	rw	0x0000 0000	发送数据寄存器（Transmit data register） 有效数据位由 DW8_32 控制。 0：只有低 8 位有效 1：TXREG[31：0] 都有效

18.4.2 接收数据寄存器（SPI_RXREG）

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

Bit	Field	Type	Reset	Description
31:0	RXREG	r	0x0000 0000	接收数据寄存器（Receive data register） 有效数据位由 DW8_32 控制。 0：只有低 8 位有效 1：RXREG[31：0] 都有效 该寄存器可读不可写。

18.4.3 当前状态寄存器（SPI_CSTAT）

偏移地址：0x08

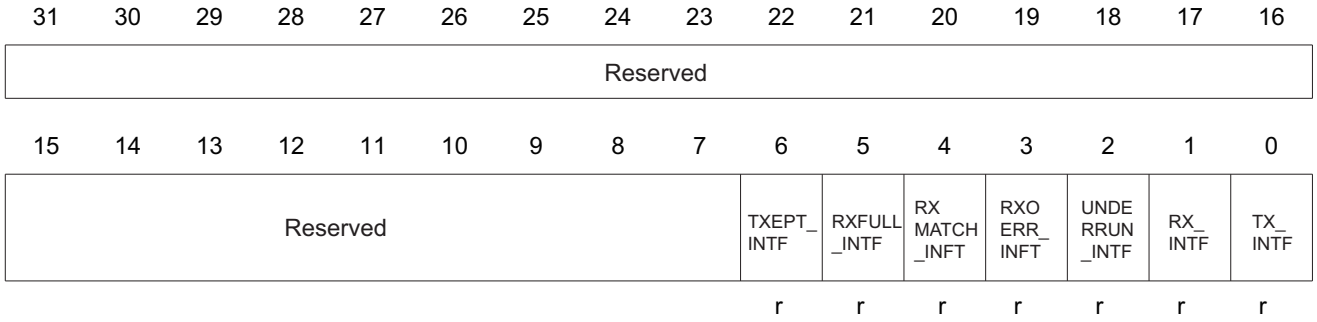
复位值：0x0000 0001

Bit	Field	Type	Reset	Description
31:12	Reserved			始终读为 0。
11:8	RXFADDR	r	0x00	当前接收缓冲器中有效数据个数
7:4	TXFADDR	r	0x00	当前发送缓冲器中有效数据个数
3	RXAVL_4BYTE	r	0x00	接收缓冲器中有效数据达到 4 个字节标志位（Receive available 4 byte data message） 1：接收缓冲器中有超过 4 个字节 0：接收缓冲器中数据小于 4 个字节
2	TXFULL	r	0x00	发送缓冲器满标志位（Transmitter FIFO full status bit） 1：发送缓冲器满 0：发送缓冲器未满
1	RXAVL	r	0x00	接收有效字节数据信息位（Receive available byte data message） 当接收端缓冲器接收了一个完整字节的数据时置位该位。 1：接收端缓冲器已经接收了一个有效字节数据 0：接收端缓冲器空 该位只读，由硬件自动置位和清除。
0	TXEPT	r	0x01	发送端空位（Transmitter empty bit） 1：发送端缓冲器和发送移位寄存器为空 0：发送端不为空 该位只读，由硬件自动置位和清除。

18.4.4 中断状态寄存器（SPI_INTSTAT）

偏移地址：0x0C

复位值：0x0000 0000



Bit	Field	Type	Reset	Description
31:7	Reserved			始终读为 0。
6	TXEPT_INTF	r	0x00	发送端空中断标志位（Transmitter empty interrupt flag bit） 硬件自动置位，写 INTCLR 寄存器 TXEPT_ICLR 位清除。 1：发送端缓冲器和 TX 移位寄存器为空 0：发送端不为空 注意：该位是中断状态信号，TXEPT 是状态信号。
5	RXFULL_INTF	r	0x00	接收端缓冲器满中断标志位（RX FIFO full interrupt flag bit） 硬件自动置位，写 INTCLR 寄存器 RXFULL_ICLR 位清除。 1：RX 缓冲器满 0：RX 缓冲器未满
4	RXMATCH_ INTF	r	0x00	接收指定字节数中断标志位（Receive data match the RXDNR number, the receive process will be completed and generate the interrupt） 硬件自动置位，写 INTCLR 寄存器 RXMATCH_ICLR 位清除。 1：接收了 RXDNR 寄存器指定的字节数 0：未完成 RXDNR 寄存器指定的字节数
3	RXOERR_ INTF	r	0x00	接收端溢出错误中断标志位（Receive overrun error interrupt flag bit） 硬件自动置位，写 INTCLR 寄存器 RXOERR_ICLR 位清除。 1：溢出错误 0：没有溢出错误
2	UNDERRUN_ INTF	r	0x00	SPI 从机模式下溢标志位（SPI underrun interrupt flag bit） 硬件自动置位，写 INTCLR 寄存器 UNDERRUN_ICLR 位清除。 1：下溢错误 0：没有下溢错误

Bit	Field	Type	Reset	Description
1	RX_INTF	r	0x00	接收端数据有效中断标志位（Receive data available interrupt flag bit） 硬件自动置位，写 INTCLR 寄存器 RX_ICLR 位清除。 当接收端缓冲器接收了一个完整字节数据。 1：接收端缓冲器有有效字节数据 0：接收端缓冲器空
0	TX_INTF	r	0x00	发送缓冲器有效中断标志位（发送了一个字节的数据）（Transmit FIFO available interrupt flag bit） 硬件自动置位，发送缓冲器不为空自动清零。 1：发送端缓冲器有效发送 0：发送端缓冲器无效发送

18.4.5 中断使能寄存器（SPI_INTEN）

偏移地址：0x10

复位值：0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved									TXEPT_IEN	RXFULL_IEN	RX MATCH_IEN	RXO ERR_IEN	UNDE RRUN_IEN	RX_IEN	TX_IEN
									rw	rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
31:7	Reserved			始终读为 0。
6	TXEPT_IEN	rw	0x00	发送端空中断使能位（Transmit empty interrupt enable bit） 1：中断使能 0：禁止中断
5	RXFULL_IEN	rw	0x00	接收端缓冲器满中断使能位（Receive FIFO full interrupt enable bit） 1：中断使能 0：禁止中断
4	RXMATCH_IEN	rw	0x00	接收指定字节数中断使能位（Receive data complete interrupt enable bit） 1：中断使能 0：禁止中断
3	RXOERR_IEN	rw	0x00	接收端溢出错误中断使能位（Overrun error interrupt enable bit） 1：中断使能 0：禁止中断

Bit	Field	Type	Reset	Description
2	UNDERRUN_IEN	rw	0x00	SPI 从机模式下溢中断使能位（SPI 从机模式）（Transmitter underrun interrupt enable bit（SPI slave mode only）） 1：中断使能 0：禁止中断
1	RX_IEN	rw	0x00	接收端数据中断使能位（Receive FIFO interrupt enable bit） 1：中断使能 0：禁止中断
0	TX_IEN	rw	0x00	发送缓冲器空中断使能位（Transmit FIFO empty interrupt enable bit） 1：中断使能 0：禁止中断

18.4.6 中断清除寄存器（SPI_INTCLR）

偏移地址：0x14

复位值：0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved									TXEPT_ICLR	RXFULL_ICLR	RX MATCH _ICLR	RXO ERR_ ICLR	UNDE RRUN _ICLR	RX_ ICLR	TX_ ICLR
									w	w	w	w	w	w	r

Bit	Field	Type	Reset	Description
31:7	Reserved			始终读为 0。
6	TXEPT_ICLR	w	0x00	发送端空中断清除位（Transmitter empty interrupt clear bit） 1：中断清除 0：中断没有清除
5	RXFULL_ICLR	w	0x00	接收端缓冲器满中断清除位（Receiver buffer full interrupt clear bit） 1：中断清除 0：中断没有清除
4	RXMATCH_ICLR	w	0x00	接收指定字节数中断清除位（Receive completed interrupt clear bit） 1：中断清除 0：中断没有清除

Bit	Field	Type	Reset	Description
3	RXOERR_ICLR	w	0x00	接收端溢出错误中断清除位（Overrun error interrupt clear bit） 1: 中断清除 0: 中断没有清除
2	UNDERRUN_ICLR	w	0x00	SPI 从机模式下溢中断清除位（SPI 从机模式）（Transmitter underrun interrupt clear bit（SPI slave mode only）） 1：中断清除 0：中断没有清除
1	RX_ICLR	w	0x00	接收端数据中断清除位（Receive interrupt clear bit） 1：中断清除 0：中断没有清除
0	TX_ICLR	r	0x00	发送缓冲器空中断清除位（Transmitter FIFO empty interrupt clear bit） 1：中断清除 0：中断没有清除

18.4.7 全局控制寄存器（SPI_GCTL）

偏移地址：0x18

复位值：0x0000 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				DW8_32	NSS	DMAEN	TXTLF		RXTLF		RXEN	TXEN	MODE	INTEN	SPIEN
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
31:12	Reserved			始终读为 0。
11	DW8_32	rw	0x00	发送和接收数据寄存器有效数据选择（Valid byte or double-word data select signal） 0：只有低 8 位有效 1：32 位数据都有效 注：不管是通过 CPU 还是 DMA 都必须用指定数据格式访问。
10	NSS	rw	0x00	硬件或软件控制主模式下的 NSS 输出（NSS select signal that from software or hardware） 0：由 NSSR 寄存器值控制 1：进行数据传输时硬件自动控制

Bit	Field	Type	Reset	Description
9	DMAEN	rw	0x00	接收和发送的 DMA 模式使能（DMA access mode enable） 0：DMA 模式禁止 1：DMA 模式使能
8:7	TXTLF	rw	0x00	发送缓冲器触发 DMA 请求的边沿选择（TX FIFO trigger level bit） 00：发送缓冲器有大于等于 1 个空闲数据空间时即进行 DMA 请求或发送中断请求 01：发送缓冲器有超过一半的空闲空间时即进行 DMA 请求或发送中断请求 1x：保留 注：当 DW8_32 为 0 时，一个数据空间代表 1 个字节；为 1 时，一个数据空间代表 4 字节。
6:5	RXTLF	rw	0x00	接收缓冲器触发 DMA 请求的边沿选择（RX FIFO trigger level bit） 00：接收缓冲器有大于等于 1 个有效数据时即进行 DMA 请求或接收中断请求 01：接收缓冲器有超过一半的有效数据时即进行 DMA 请求或接收中断请求 1x：保留 注：当 DW8_32 为 0 时，一个有效数据代表 1 个字节；为 1 时，一个有效数据代表 4 字节。
4	RXEN	rw	0x00	接收使能位（Receive enable bit） 1：接收使能 0：接收禁止。同时可以清空 RX 缓冲器 注意：当 SPI 只工作在主机接收模式时，txen 必须设置为 0。
3	TXEN	rw	0x00	发送使能位（Transmit enable bit） 1：发送使能 0：发送禁止。同时可以清空 TX 缓冲器 注意：当在主机模式下发送和接收同时发生。
2	MODE	rw	0x01	主机模式位（Master mode bit） 1：主机模式（由内部 BRG 产生串行时钟） 0：从机模式（串行时钟来自外部主机）
1	INTEN	rw	0x00	SPI 中断使能位（SPI interrupt enable bit） 1：使能 SPI 中断 0：禁止 SPI 中断
0	SPIEN	rw	0x00	SPI 选择位（SPI select bit） 0：SPI 禁止（复位状态） 1：SPI 使能

18.4.8 通用控制寄存器（SPI_CCTL）

偏移地址：0x1C

复位值：0x0000 0008

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved										TX EDGE	RX EDGE	SPI LEN	LSB FE	CPOL	CPHA
										rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
31:6	Reserved			始终读为 0。
5	TXEDGE	rw	0x00	发送数据相位调整位（从模式）（Transmit data edge select） 1：发送数据立即发送到数据总线 可用于高速模式时（SPBRG = 4）。 0：发送数据在一个有效时钟边沿后发送到数据总线 可用于低速模式时（SPBRG > 4）。
4	RXEDGE	rw	0x00	接收数据采样时钟沿选择位（主模式）（Receive data edge select） 1：在传输数据位的尾时钟沿采样数据（用于高速模式） 0：在传输数据位的中间采样数据
3	SPILEN	rw	0x01	SPI 数据宽度位（SPI character length bit） 该位在 DW8_32 置位后（DW8_32=0）配置后起作用。 1：8 位数据（缺省） 0：7 位数据
2	LSBFE	rw	0x00	LSBFE：LSB 在前使能位（LSI first enable bit） 1：数据传输或接收最低位在前 0：数据传输或接收最高位在前
1	CPOL	rw	0x00	时钟极性标志位（Clock polarity select bit） 1：时钟在空闲状态为高电平（两次传输之间） 0：时钟在空闲状态为低电平（两次传输之间）
0	CPHA	rw	0x00	时钟相位选择位（Clock phase select bit） 1：第一个数据位采样从第一个时钟边沿开始 0：第一个数据位采样从第二个时钟边沿开始

18.4.9 波特率发生器（SPI_SPBRG）

偏移地址：0x20

复位值：0x0000 0002

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

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

Bit	Field	Type	Reset	Description
31:16	Reserved			始终读为 0。
15:0	SPBRG	rw	0x0002	SPI 波特率控制寄存器用于产生波特率（SPI baud rate control register for baud rate） 波特率公式： 波特率 = f_{pclk}/SPBRG （f_{pclk}/是 APB 时钟频率） 注意：不要往该寄存器写 0 和 1。

18.4.10 接收数据个数寄存器（SPI_RXDNR）

偏移地址：0x24

复位值：0x0000 0001

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

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

Bit	Field	Type	Reset	Description
31:16	Reserved			始终读为 0。
15:0	RXDNR	rw	0x0001	该寄存器用于存储下次接收过程需要接收字节的个数（The register is used to hold a count of to be received bytes in next receive process） 该寄存器的值在 SPI 为主机接收模式下有效。缺省值是 1。 该寄存器值通过 MCU 写值改变。 注意：不要往该寄存器写‘0’值。

18.4.11 从机片选寄存器（SPI_NSSR）

偏移地址：0x28

复位值：0x0000 00FF

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

Bit	Field	Type	Reset	Description
15:1	Reserved			
0	NSS	rw	0x01	主模式下片选输出信号。低有效，从模式下该位无效（Chip select output signal in Master mode）。 0：从器件被选中 1：从器件未选中

18.4.12 数据控制寄存器（SPI_EXTCTL）

偏移地址：0x2C

复位值：0x0000 0008

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

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

Bit	Field	Type	Reset	Description
31:5	Reserved			始终读为 0。
4：0	EXTLEN	rw	0x08	控制 SPI 数据长度 0 0000：32 bit 0 0001：1 bit 0 0010：2 bit 0 0011：3 bit … 1 1100：28 bit 1 1101：29 bit 1 1110：30 bit 1 1111：31 bit 注：仅当 SPI_GCTL 寄存器 DW8_32 位为‘0’时有效，且 SPI_CCTL 寄存器的 LSBFE 位必须配置为‘1’，SPILEN 位也必须配置为‘1’。

19 I2C 接口 (I2C)

I2C 接口 (I2C)

19.1 I2C 简介

I2C（芯片间）总线接口连接微控制器和串行 I2C 总线。它提供多主机功能，控制所有 I2C 总线特定的时序、协议、仲裁和定时。

I2C 总线是一个两线串行接口，其中两线位串行数据（SDA）和串行时钟（SCL）线在连接到总线器件间传递信息。每个器件都有一个唯一的地址识别，而且都可以作为一个发送或接收器。除了发送器和接收器外，器件在执行数据传输时也可以被看做是主机或者从机。主机是初始化总线的数据传输并产生允许传输的时钟信号的器件。此时，任何被寻址的器件都被认为是从机。

I2C 可以工作在标准模式（数据传输速率为 0 ~ 100 Kbps），快速模式（数据传输速率最大为 400 Kbps）。

19.2 I2C 主要特征

- 并行总线 I2C 总线协议转换器
- 半双工同步操作
- 支持主从模式
- 支持 7 位地址和 10 位地址
- 支持标准模式 100 Kbps，快速模式 400 Kbps
- 产生 Start、Stop、重新发 Start、应答 Acknowledge 信号检测
- 在主模式下只支持一个主机
- 分别有 2 字节的发送和接收缓冲
- 在 SCLI 和 SDAI 上增加了无毛刺电路
- 支持 DMA 操作
- 支持中断和查询操作

19.3 I2C 协议

19.3.1 起始和停止条件

当总线处于空闲状态时，SCL 和 SDA 同时被外部上拉电阻拉为高电平。当主机启动数据传输时，必须先产生一个起始条件。在 SCL 线是高电平时，SDA 线从高电平向低电平切换表示起始条件。当主机结束传输时要发送停止条件。在 SCL 线是高电平，SDA 线由低电平向高电平切换表示停止条件。下图显示了起始和停止条件的时序图。数据传输过程中，当 SCL 为 1 时，SDA 必须保持稳定。

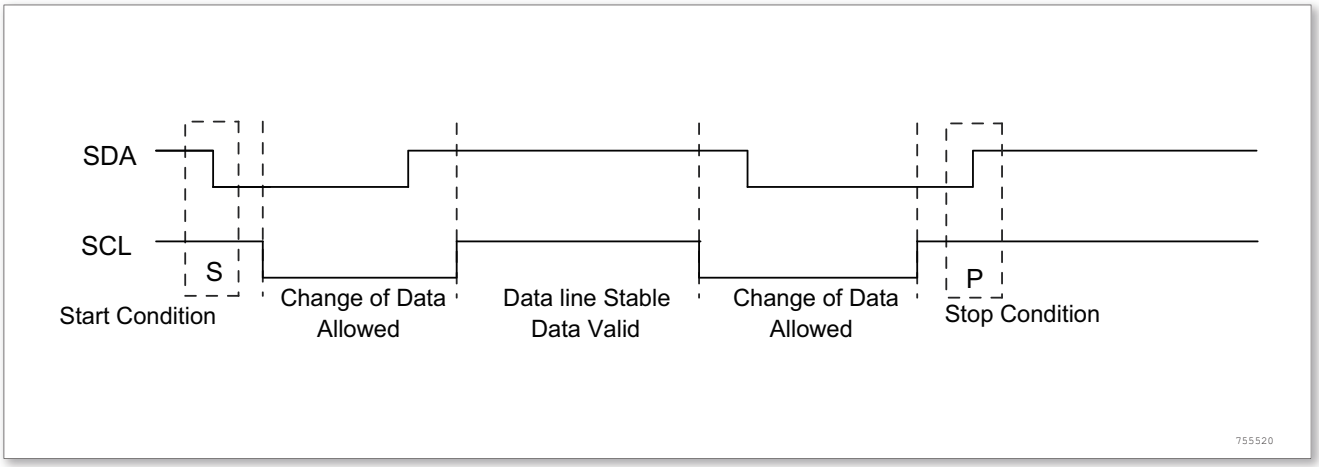


图 148. 起始和停止条件

19.3.2 从机寻址协议

I2C 有两种地址格式：7 位的地址格式和 10 位的地址格式。

7 位的地址格式

下图中显示在起始条件（S）后发送的一个字节的前 7 位（bit 7：1）为从机地址，最低位（bit 0）是数据方向位，当 bit 0 为 0，表示主机写数据到从机，1 表示主机从从机读数据。

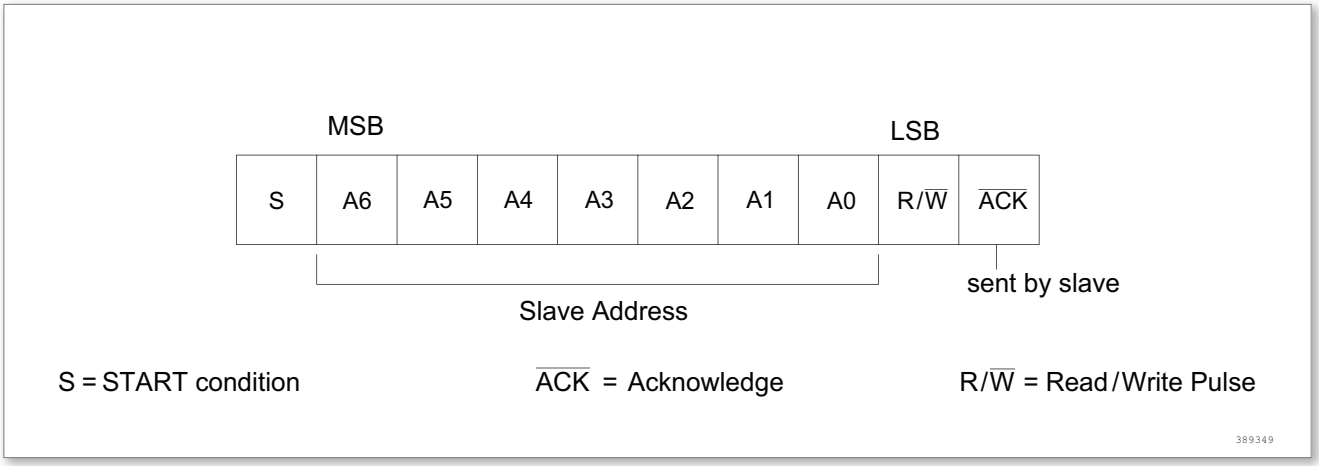


图 149. 7 位的地址格式

10 位的地址格式

在 10 位的地址格式中，发送 2 个字节来传输 10 位地址。发送的第一个字节的位的描述如下：第一个 5 位（bit 7：3）用于告示从机接下来是 10 位的传输。第一个字节的后两个字节（bit 2：1）位从机地址的 bit 9：8，最低位（bit 0）是数据方向位（R/W）。传输的第二个字节为 10 位地址的低八位。

具体如下图所示：

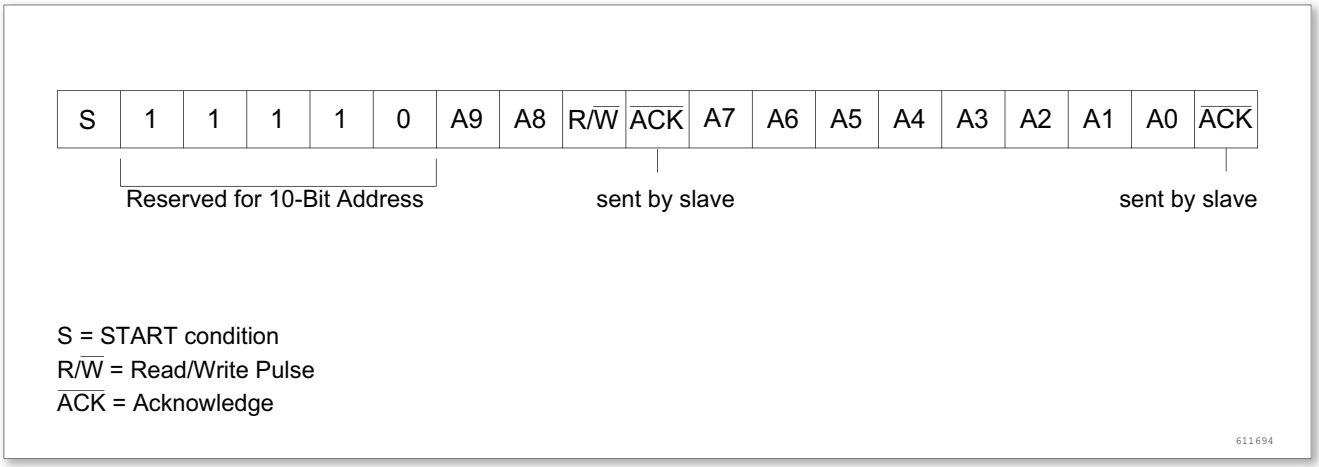


图 150. 10 位的地址格式

下表定义了 I2C 首字节的特殊用途和保留地址：

表 70. I2C 首字节

从机地址	R/W 位	描述
0000 000	0	广播呼叫地址。I2C 将数据放入接收缓冲，并产生一个广播呼叫中断
0000 000	1	起始字节
0000 001	X	CBUS 地址。I2C 接口忽略该访问
0000 010	X	保留
0000 011	X	保留
0000 1xx	X	保留
1111 1xx	X	保留
1111 0xx	X	10 位从机寻址

19.3.3 发送和接收协议

主机初始化数据传输并且从总线上发送或接收数据，作为主发送或者主接收。从机响应主机的请求来发送或接收数据，作为从发送或从接收器。

主发送和从接收

所有数据都是以字节格式传输，且不限制每次传输的字节数。当主机发送完地址和 R/W 位或者主机发送一个字节的数据到从机上，从接收器必须产生一个响应信号（ACK）。当从接收器不能产生 ACK 响应信号，主机将会产生一个停止条件中止传输。从机不能响应时，必须释放 SDA 为高电平才能使得主机产生停止条件。

当主发送器传输数据如下图所示，从接收器在接收到的每个字节后产生一个 ACK 来响应主发送器。

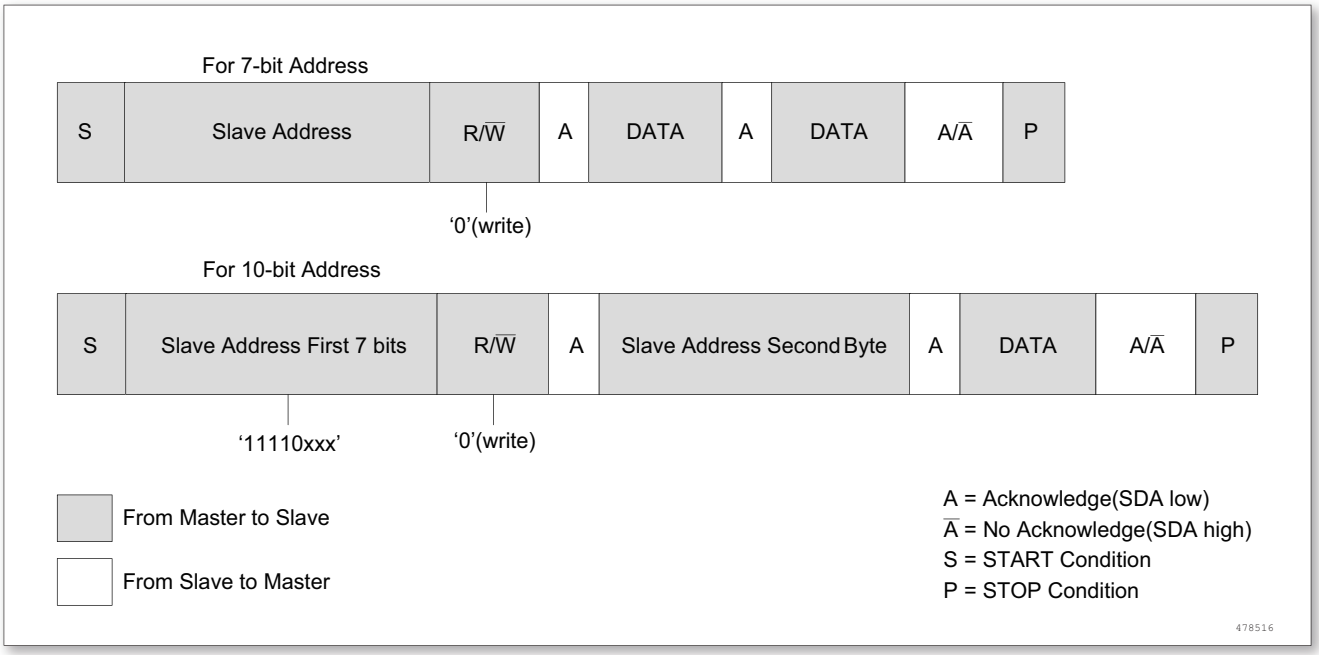


图 151. 主发送协议

主接收和从发送

当主机接收数据如下图所示，主机必须在每次接收到一个字节数据后响应从发送器，除了最后一个字节。通过这种方式，主接收器能通知从发送器是否是最后一个字节。从发送器在检测到 NACK 时必须释放 SDA，这样主机可以产生停止条件。

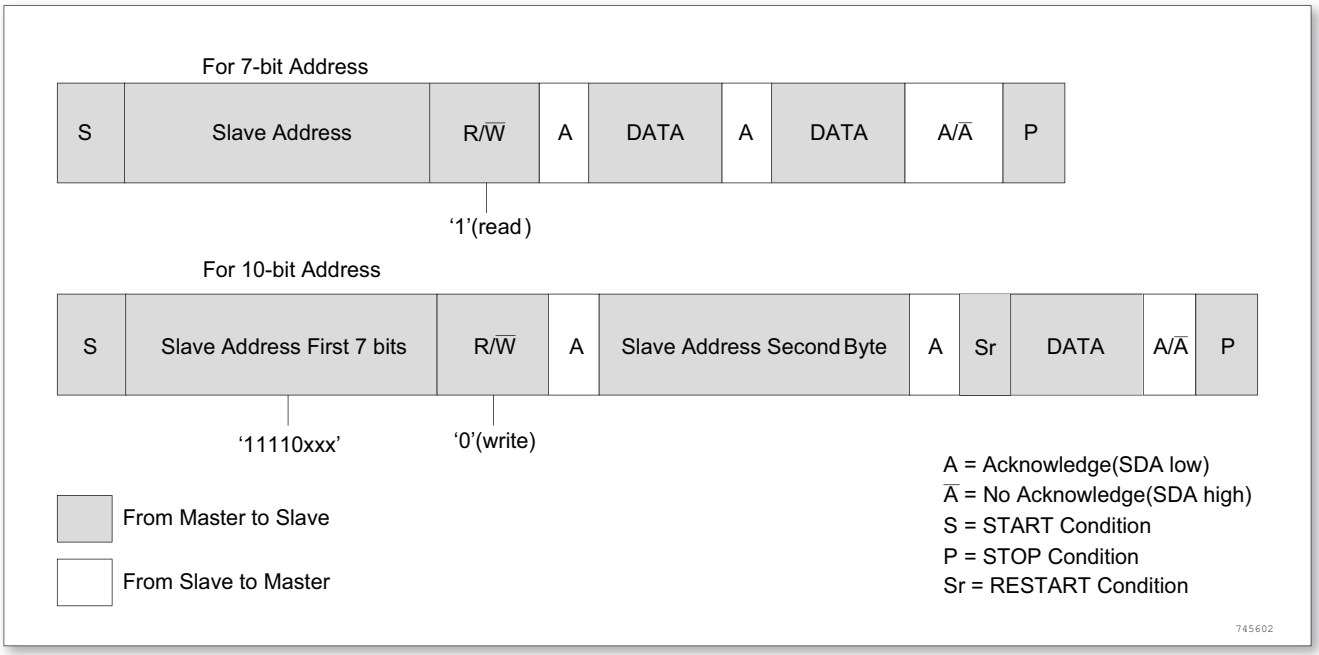


图 152. 主接收协议

当主机不想产生停止条件而释放总线，可以产生一个重复起始条件。重复起始条件与起始条件相同，只是它是在 ACK 后产生。工作在主机模式下，I2C 接口可以使用不同的传输方向与相同的从机通信。

起始字节传输协议

起始字节传输协议是用来给没有专用的 I2C 硬件模块的系统使用。当 I2C 模块作为主机时，在每次传输开始可以给需要的从机产生起始字节输出。

该协议由 7 个 0 以及一个 1 组成，如下图所示。处理器可以在地址阶段用低速采样 0 来查询总线。一旦检测到 0，处理器可以从低速采样切换到主机的正常速率。

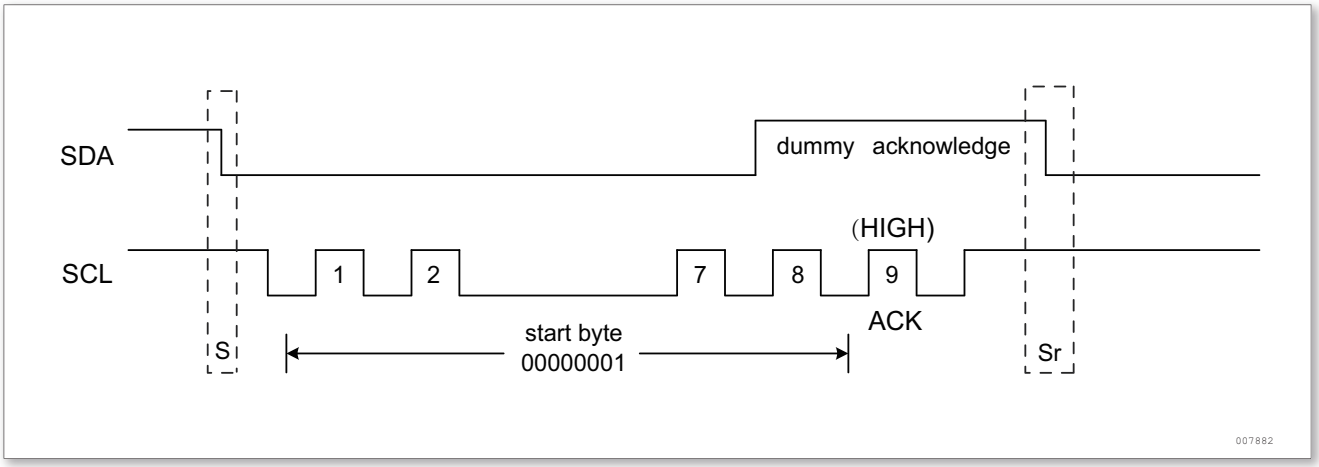


图 153. 起始字节传输

起始字节程序流程如下：

1. 主机产生一个起始条件
2. 主机发送起始字节（0000 0001）
3. 主机发送 ACK 时钟脉冲（ACK）
4. 没有从机响应 ACK 信号
5. 主机产生重复起始条件（RESTART）

硬件 I2C 接收器不需要响应开始字节，因为这是一个保留地址，而且地址会在 RESTART 后复位。

19.3.4 发送缓冲管理以及起始、停止和重复起始条件产生

当工作在主机模式，每当发送为空时 I2C 模块就在总线上产生一个停止条件。如果重复起始产生功能使能（RESTART = 1），则传输方向从读变为写或者写变为读时产生重复起始条件。如果没有使能重复起始条件，则会在停止条件后产生一个起始条件。

下图显示了 DR 寄存器的位。

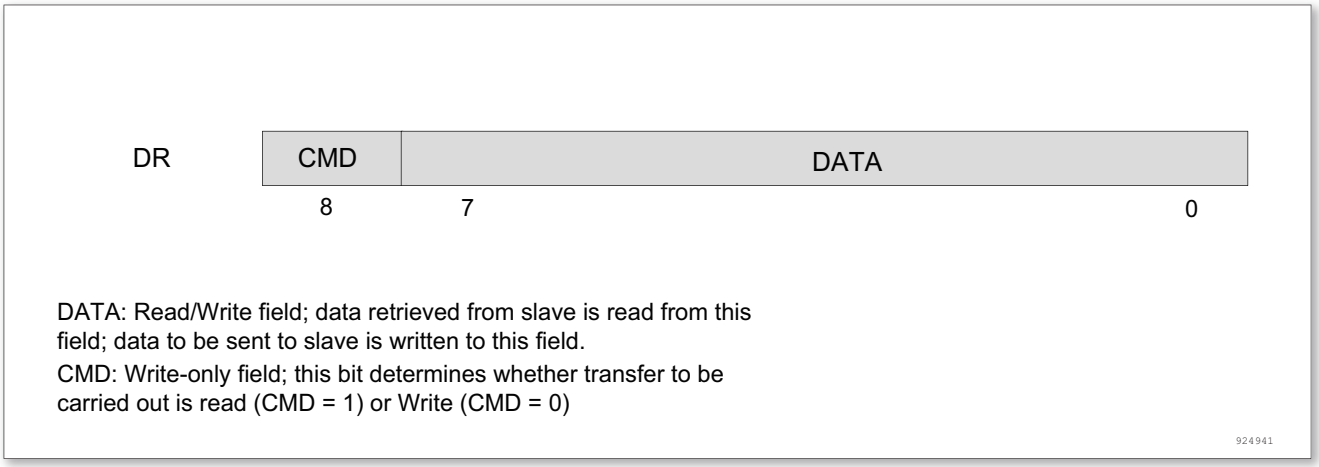


图 154. DR 寄存器

下面的时序图描述了 I2C 模块工作在主发送模式下 Tx FIFO 变为空时行为。

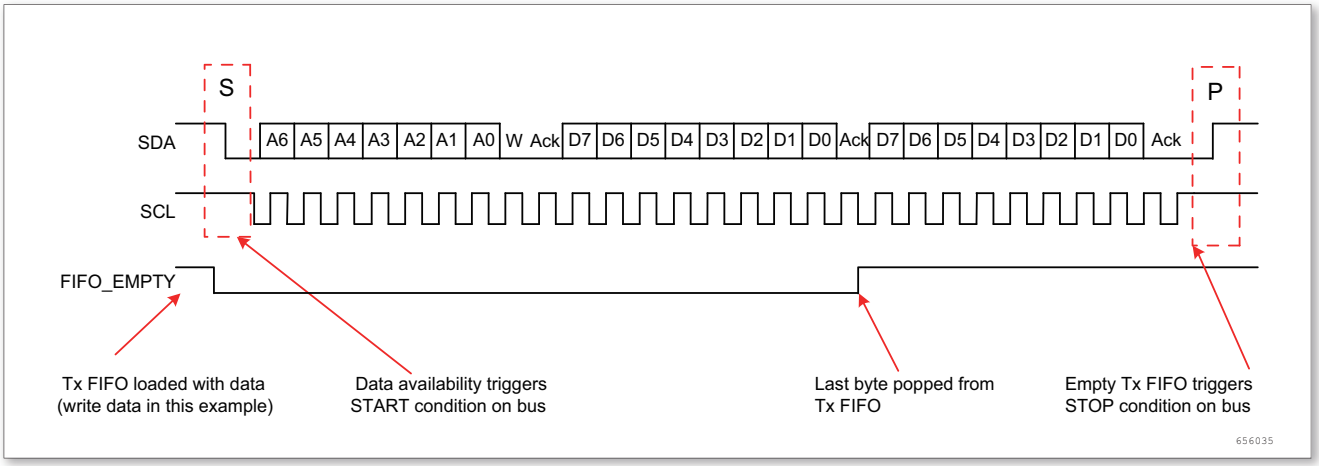


图 155. 主发送 - Tx FIFO 为空

下面的时序图描述了 I2C 模块工作在主接收模式下当 Tx FIFO 变为空时行为。

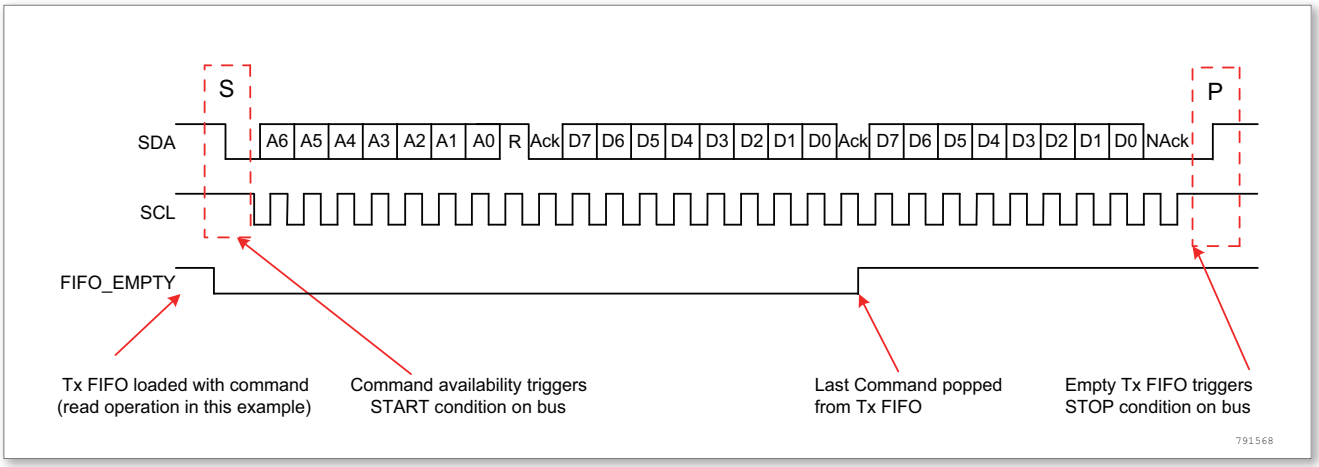


图 156. 主接收 - Tx FIFO 为空

19.3.5 多个主机仲裁

I2C 总线是一个多主机的总线。仲裁是一个在有多个主机同时尝试控制总线，但只允许其中一个控制总线并使报文不被破坏的过程。一旦其中一个主机已经控制了总线，那么直到

该主机发送一个停止条件并且将总线释放为空闲状态时，其他主机才能控制总线。

当 SCL 线是高电平时，仲裁在 SDA 线发生。如果两个或多个主机尝试发送信息到总线，在其他主机都产生”0” 的情况下，首先产生一个”1” 的主机将丢失仲裁。丢失仲裁的主机可以继续产生时钟脉冲直到字节传输结束。如果每个主机都尝试寻址相同的器件，仲裁会继续在数据阶段进行。

检测到丢失仲裁后，I2C 接口会停止产生 SCL 信号。

下图显示了两个主机的仲裁的总线时序

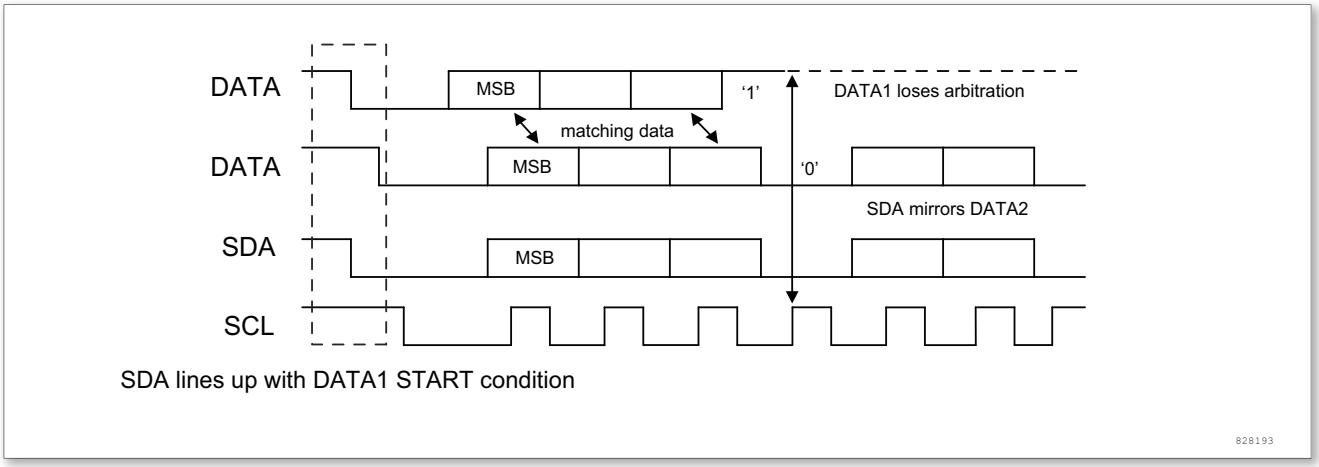


图 157. 多个主机仲裁

19.3.6 时钟同步

当两个或多个主机试图同时在总线传输信息时，他们必须仲裁和同步 SCL 时钟。所有的主机产生自己的时钟来传输消息。数据只在时钟的高电平有效。时钟同步是通过 SCL 信号的线‘与’连接进行的。当主机把 SCL 时钟变成 0，主机会计算 SCL 低电平的时间，在下一个时钟周期开始把 SCL 时钟变成 1。但是，假如另一个主机把 SCL 保持为 0，那么这个主机会进入等待状态直到 SCL 时钟变为 1。

所有的主机会计算它们的高电平时间，最短高电平时间的主机会把 SCL 变为 0。接下来主机会计算低电平时间，最长低电平时间的主机会强制其他主机进入等待状态。这样就产生一个同步后的 SCL 时钟，如下图所示。

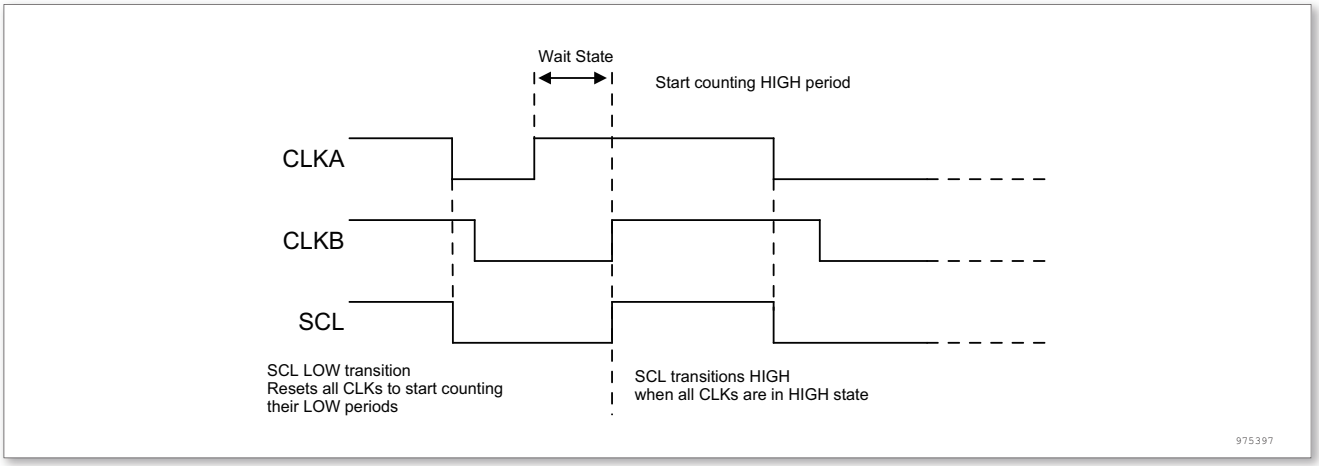


图 158. 多个主机时钟同步

19.4 I2C 工作模式

I2C 接口可以以下述 4 种方式中的一种运行：

- 从发送器模式
- 从接收器模式
- 主发送器模式
- 主接收器模式

注：I2C 接口模块只能工作在主机模式或者从机模式，但不能同时工作在两种模式下。因此要确保寄存器 CR 中位 6（DISSLAVE）和位 0（MASTER）不能分别设置为 0 和 1（或者分别为 1 和 0）。

I2C 功能框图如下：

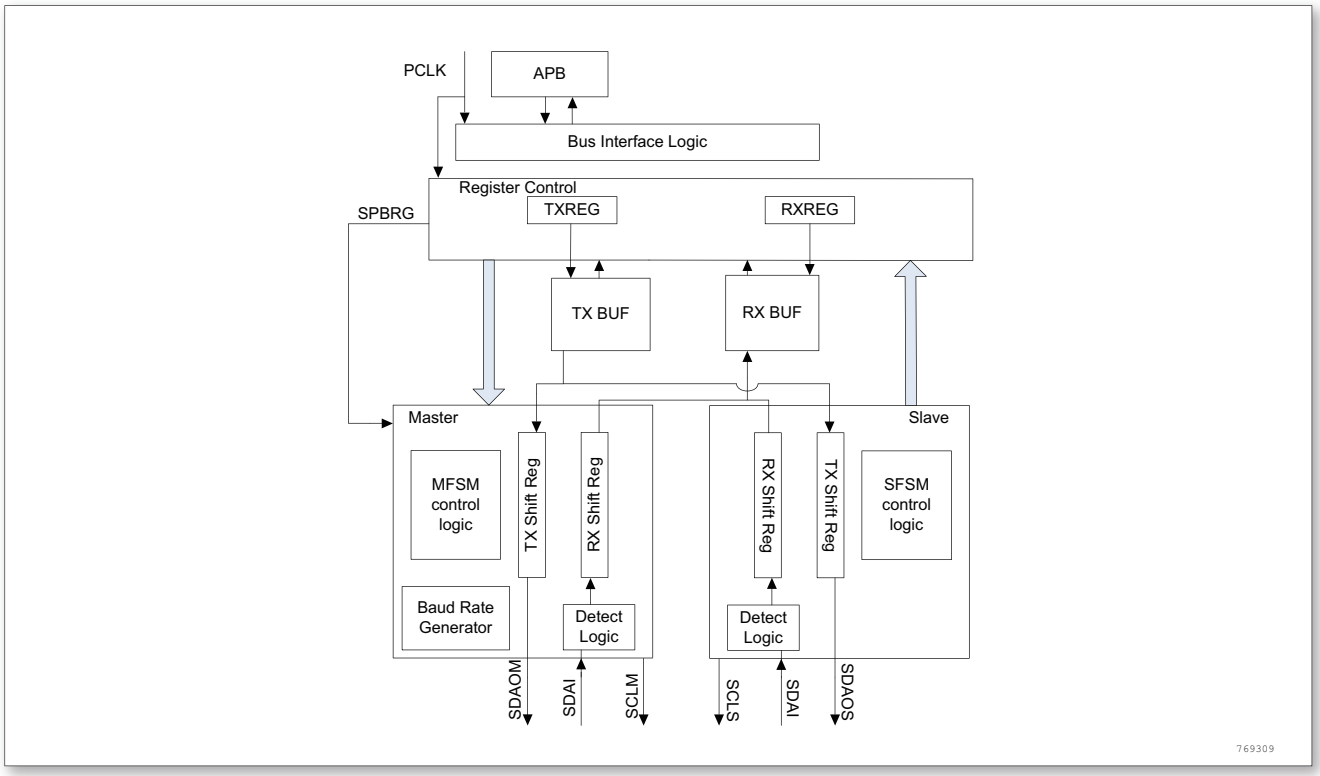


图 159. I2C 功能框图

19.4.1 从模式

下面介绍从模式的程序流程图

初始化配置

1. 写 0 到 ENR 寄存器位 0 禁止 I2C。
2. 通过初始化 SAR 寄存器来配置从机地址。该地址为 I2C 接口所响应的地址。
3. 配置 CR 寄存器指定地址格式（设置 bit 3 来选择 7 位或 10 位地址格式）。写 0 到寄存器 CR 寄存器的位 6（DISSLAVE）和写 0 到位 0（MASTER）。
4. 写 1 到 ENR 寄存器中位 0 来使能 I2C 接口模块。

从发送的单字节操作

当 I2C 接口被其他 I2C 主机寻址并请求数据的时候，I2C 接口工作在从发送模式，步骤如

下：

1. 其他 I2C 主机器件初始化 I2C 传输，发送地址与 SAR 寄存器中的从机地址匹配。
2. I2C 接口响应发送的地址，识别传输的方向是工作在从发送模式。
3. I2C 接口产生 RD_REQ 中断（寄存器 RAWISR 位 5），并且将 SCL 线拉低。总线一直处于等待状态直到软件响应。

如果 RD_REQ 中断被屏蔽（寄存器 IMR[5] = 0），建议 CPU 定期查询 RAWISR 寄存器。

1. RAWISR 位 5 置位等效于产生了一个 RD_REQ 中断。
2. 软件必须满足 I2C 传输的要求。
3. 时间间隔通常在 10 个 SCL 时钟周期左右。例如，对于 400kbps，时间间隔是 25us。
4. 如果在接收读请求之前 Tx FIFO 仍然有数据，I2C 接口就会产生一个 TX_ABRT 中断（RAWISR[6]），清空 Tx FIFO 中的数据。
5. 软件写数据到 DR 寄存器（其中位 8 设置为 0）。
6. 软件必须先清除 RAWISR 寄存器 RD_REQ 和 TX_ABRT 中断（分别为 bit5，6）
7. I2C 接口释放 SCL，并发送数据字节。
8. 主机器件发送重复起始条件控制总线或者发送停止条件释放总线。

从接收的单字节操作

当其他主机器件寻址 I2C 接口并且发送数据，I2C 接口工作在从接收模式，步骤如下：

1. 其他 I2C 主机器件初始化 I2C 传输，发送地址与 SAR 寄存器中的从机地址匹配。
2. I2C 接口响应发送的地址，识别传输的方向是工作在从接收模式。
3. I2C 接口收到主机发送的数据并将数据存储在接收缓冲中。
4. I2C 接口产生 RX_FULL 中断（RAWISR[2]）. 如果 RX_FULL 中断被屏蔽（IMR[2] = 0），建议软件定期查询 SR 寄存器中。读到 SR 寄存器位 3（RFNE）为 1 时等效于 RX_FULL 中断产生。
5. 软件通过读 DR 寄存器中的 bit 7:0 来获得接收到的数据。
6. 主机器件发送重复起始条件控制总线或者发送停止条件释放总线。

从机的块传输操作

标准的 I2C 协议中，所有的数据处理都是单个字节的处理，程序通过写一个字节到从机的 Tx FIFO 响应主机的读请求。当一个从机（从发送）接收到主机（主接收）的读请求（RD_REQ）时，最少有一个数据放到从发送的 Tx FIFO。这个 I2C 接口模块可以处理 x FIFO 中有多个数据，所以接下来的读请求不需要再产生中断来取数据。最终，这极大的减少了因为每次数据中断导致等待时间。

该模式仅存在当 I2C 接口作为从发送模式。如果主机发送响应从发送传输的数据，从机的 TX FIFO 中没有数据，I2C 接口将拉低 I2C 总线的 SCL 线直到读请求中断（RD_REQ）产生并且 TX FIFO 的数据准备好后才释放 SCL 线。

如果 RX_REQ 中断被屏蔽（ISR[5] = 0），软件可以定期查询读 RAWISR 寄存器。当读到 RAWISR[5] 返回为 1 等效于产生了 RX_REQ 中断。

RD_REQ 中断由于读请求产生，像中断一样必须退出中断服务程序（ISR）时清除。在中断服务程序中（ISR）可以写一个或多个字节的数据到 TX FIFO。在这些字节传输给主机的过程中，如果主机响应了最后一个字节，从机将必须再次产生 RD_REQ 中断请求。这是因为主机要求更多的数据。

如果主机接收了来自 I2C 接口的 n 字节，但是程序写到 Tx FIFO 中的数据个数大于 n，从机在完成要求的 n 字节的数据发送后，将会清空 Tx FIFO 并且忽略额外的字节。

19.4.2 主模式

初始化配置

1. 通过设置 ENR[0] = 0 来禁止 I2C 接口
2. 配置 CR 寄存器的 bit 2:1 设置 I2C 工作的速率模式（标准模式、快速模式）。同时确保 bit 6（DISSLAVE）为 1，且 bit 0（MASTER）为 1。
3. 往 TAR 寄存器写入 I2C 器件地址。设置该寄存器可配置为广播地址或起始字节命令。
4. 置位 ENR[0] 使能 I2C 接口。
5. 将传输的数据以及传输方向写入到 DR 寄存器中。如果在使能 I2C 接口之前配置了 DR 寄存器，数据和命令都会丢失，这是因为在 I2C 接口禁止的情况下缓冲是清空的。

以上的步骤将会使得 I2C 接口产生一个起始条件并发送地址字节数据到 I2C 总线上。

主发送和主接收

I2C 接口支持读写的动态切换。当发送数据时，写数据到 I2C RX/TX 数据缓冲和命令寄存器的低字节中（DR），配置 CMD 位为 0 产生写操作。接下来的读命令，不需要设置 DR 寄存器的低字节，只需要确保 CMD 位为 1。如果发送 FIFO 为空，I2C 模块拉低 SCL 直到下个命令写入到发送 FIFO 中。

程序流程图

下面的流程图为 I2C 接口作为主机的程序示例：

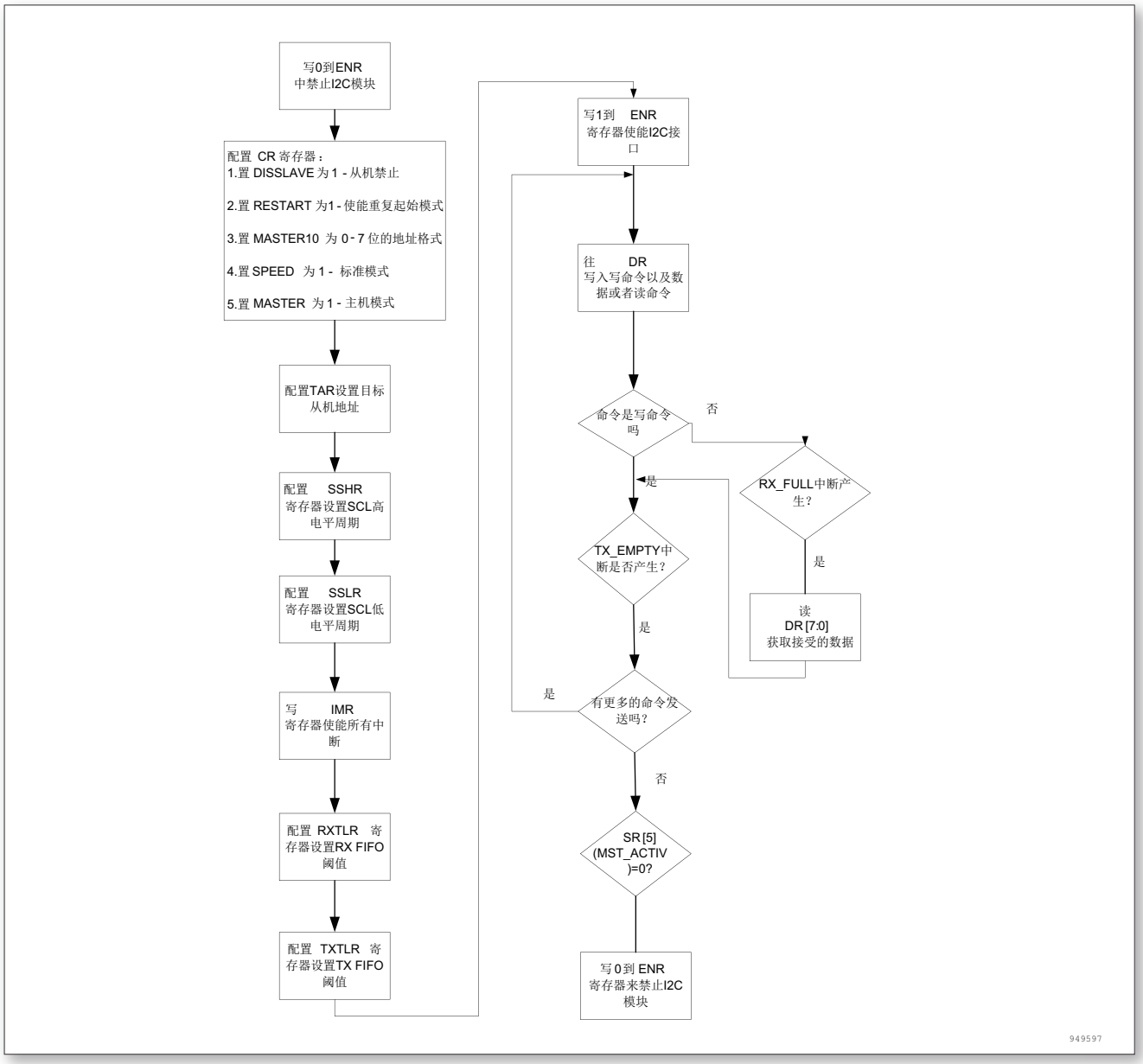


图 160. I2C 接口主机流程图

19.4.3 I2C 中止传输

ENR 寄存器中的 ABRT 控制位允许软件在完成 TX FIFO 中传输命令之前放弃 I2C 总线。作为 ABORT 请求的响应，I2C 模块发出一个停止条件到 I2C 总线，同时清空 TX FIFO。中止传输操作值允许在主模式下。

程序流程

1. 停止往 Tx FIFO（DR）中写新的命令
2. 如果工作在 DMA 模式中，置 TDMAE = 0 禁止发送 DMA。
3. 置 ENR 寄存器 ABRT 位为 1
4. 等待 TX_ABRT 中断

19.5 利用 DMA 通信

I2C 接口支持用 DMA 来发送和接收数据。通过设置 DMA 寄存器中的对应位可以单独开启 DMA 发送或者 DMA 接收。发送时数据寄存器变空或接收时数据寄存器变满，则产生 DMA 请求。DMA 请求必须在当前字节传输结束之前被响应。

利用 DMA 发送

通过设置 DMA 寄存器的 TXEN 位可以激活 DMA 发送模式。为 I2C 分配好 DMA 通道后，当发送数据时，DMA 控制器会将数据从预置的存储区装载进 DR 寄存器。

利用 DMA 接收

通过设置 DMA 寄存器的 RDMAE 位可以激活 DMA 接收模式。为 I2C 分配好 DMA 通道后，当每次接收到数据字节时，DMA 控制器会将数据从 DR 寄存器中传送到预置的存储区。

19.6 I2C 中断

下表列出了 I2C 的中断位以及它们的设置和清除方式。部分位由硬件置位并由软件清除；另一部分位由硬件置位和清除。

表 71. 中断位的置位和清除

中断位	硬件置位/软件清除	硬件置位和清除
GEN_CALL	√	x
START_DET	√	x
STOP_DET	√	x
ACTIVITY	√	x
RX_DONE	√	x
TX_ABRT	√	x
RD_REQ	√	x
TX_EMPTY	x	√
TX_OVER	√	x
RX_FULL	x	√
RX_OVER	√	x
RX_UNDER	√	x

下图描述了中断寄存器中，中断位被硬件置位和软件清除的操作

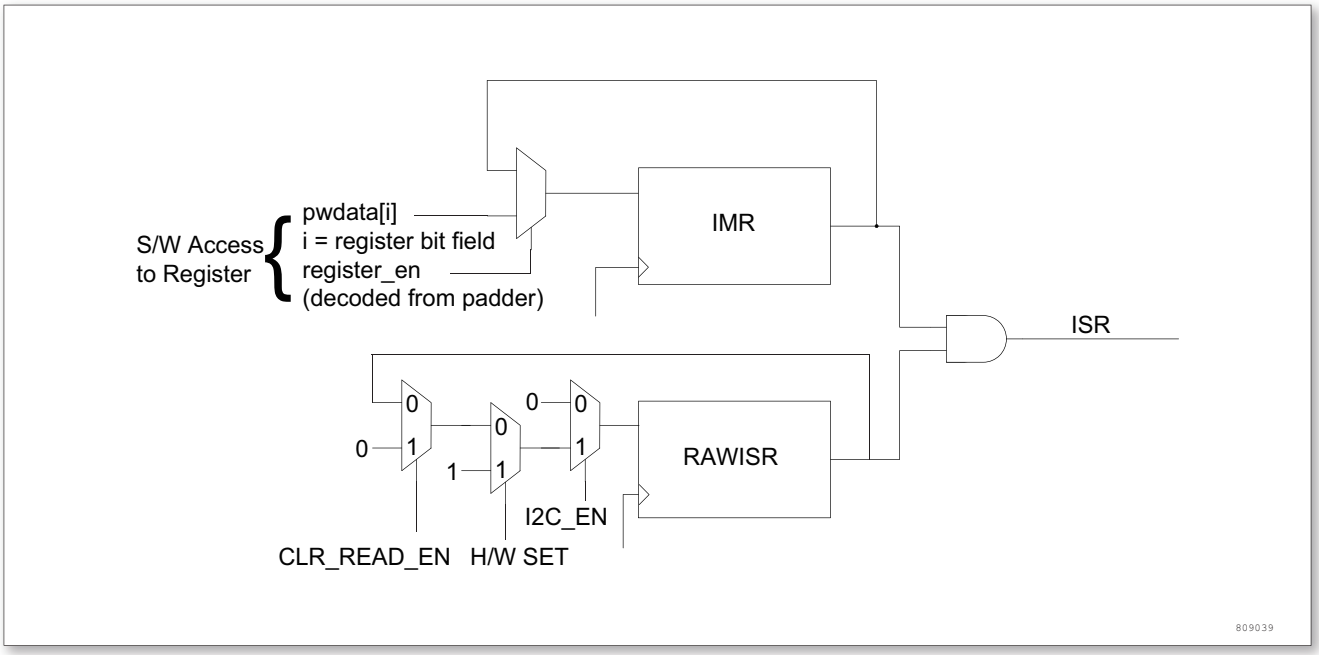


图 161. 中断机制

19.7 I2C 寄存器描述

表 72. I2C 寄存器描述概览

Offset	Acronym	Register Name	Reset	Section
0x00	I2C_CR	I2C 控制寄存器	0x00000011	小节 19.7.1
0x04	I2C_TAR	I2C 目标地址寄存器	0x00000055	小节 19.7.2
0x08	I2C_SAR	I2C 从机地址寄存器	0x00000055	小节 19.7.3
0x10	I2C_DR	I2C 数据命令寄存器	0x00000001	小节 19.7.4
0x14	I2C_SSHR	标准模式 I2C 时钟高电平计数寄存器	0x00000190	小节 19.7.5
0x18	I2C_SSLR	标准模式 I2C 时钟低电平计数寄存器	0x000001D6	小节 19.7.6
0x1C	I2C_FSHR	快速模式 I2C 时钟高电平计数寄存器	0x00000036	小节 19.7.7
0x20	I2C_FSLR	快速模式 I2C 时钟低电平计数寄存器	0x00000082	小节 19.7.8
0x2C	I2C_ISR	I2C 中断状态寄存器	0x00000000	小节 19.7.9
0x30	I2C_IMR	I2C 中断屏蔽寄存器	0x000008FF	小节 19.7.10
0x34	I2C_RAWISR	I2C RAW 中断寄存器	0x00000000	小节 19.7.11
0x38	I2C_RXTLR	I2C 接收阈值	0x00000000	小节 19.7.12
0x3C	I2C_TXTLR	I2C 发送阈值	0x00000000	小节 19.7.13
0x40	I2C_ICR	I2C 组合和独立中断清除寄存器	0x00000000	小节 19.7.14
0x44	I2C_RX_UNDER	I2C 清除 RX_UNDER 中断寄存器	0x00000000	小节 19.7.15

Offset	Acronym	Register Name	Reset	Section
0x48	I2C_RX_OVER	I2C 清除 RX_OVER 中断寄存器	0x00000000	小节 19.7.16
0x4C	I2C_TX_OVER	I2C 清除 TX_OVER 中断寄存器	0x00000000	小节 19.7.17
0x50	I2C_RD_REQ	I2C 清除 RD_REQ 中断寄存器	0x00000000	小节 19.7.18
0x54	I2C_TX_ABRT	I2C 清除 TX_ABRT 中断寄存器	0x00000000	小节 19.7.19
0x58	I2C_RX_DONE	I2C 清除 RX_DONE 中断寄存器	0x00000000	小节 19.7.20
0x5C	I2C_ACTIV	I2C 清除 ACTIVITY 中断寄存器	0x00000000	小节 19.7.21
0x60	I2C_STOP	I2C 清除 STOP_DET 中断寄存器	0x00000000	小节 19.7.22
0x64	I2C_START	I2C 清除 START_DET 中断寄存器	0x00000000	小节 19.7.23
0x68	I2C_GC	I2C 清除 GEN_CALL 中断寄存器	0x00000000	小节 19.7.24
0x6C	I2C_ENR	I2C 使能寄存器	0x00000000	小节 19.7.25
0x70	I2C_SR	I2C 状态寄存器	0x00000006	小节 19.7.26
0x74	I2C_TXFLR	I2C 发送缓冲水平寄存器	0x00000000	小节 19.7.27
0x78	I2C_RXFLR	I2C 接收缓冲水平寄存器	0x00000000	小节 19.7.28
0x7C	I2C_HOLD	I2C SDA 保持时间寄存器	0x00000001	小节 19.7.29
0x88	I2C_DMA	I2C DMA 控制寄存器	0x00000000	小节 19.7.30
0x94	I2C_SETUP	I2C SDA 建立时间寄存器	0x00000064	小节 19.7.31
0x98	I2C_GCR	I2C 广播呼叫 ACK 寄存器	0x00000001	小节 19.7.32

19.7.1 I2C 控制寄存器（I2C_CR）

偏移地址：0x00

复位值：0x0011

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved							EMPINT	STOPINT	DISSLAVE	REPEN	MASTER10	SLAVE10	SPEED		MASTER
							rw	rw	rw	rw	r	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
15 : 9	Reserved			始终读为 0。
8	EMPINT	rw	0x00	该位控制 TX_EMPTY 中断产生，细节参考 IC_RAW_INTR_STAT 寄存器。
7	STOPINT	rw	0x00	在从机模式下，是否产生 STOP_DET 中断。 1: 当地址匹配时才产生 STOP_DET 中断 0: 无论地址是否匹配，都产生 STOP_DET 中断 该位仅适用于从机模式 注：广播地址寻址时，如果该位置位，从机不产生 STOP_DET 中断。STOP_DET 中断仅当发送的地址与从机地址匹配时产生。

Bit	Field	Type	Reset	Description
6	DISSLAVE	rw	0x00	该位控制 I2C 接口从机禁止（This bit controls whether I2C has its slave disabled） 0：从机使能 1：从机禁止
5	REPEN	rw	0x00	当作为主机时该位控制是否发送 RESTART 条件（Determines whether RESTART conditions may be sent when acting as a master） 0：禁止 1：使能 当 RESTART 禁止，I2C 接口作为主机时不能执行以下功能： 发送起始字节 组合格式模式下改变传输方向 10 位的地址格式的读操作 替换 RESTART 条件为先发送停止条件再发送起始条件。 如果上述操作执行会置位 IC_RAW_INTR_STAT 寄存器的位 6（TX_ABRT）
4	MASTER10	r	0x01	I2C 作为主机时的地址格式（Address mode when acting as a master） 0：7 位的地址格式 1：10 位的地址格式
3	SLAVE10	rw	0x00	当作为从机时，该位控制响应 10 位或者 7 位地址（When acting as a slave, this bit controls whether the I2C responds to 7- or 10-bit addresses） 0：7 位的寻址地址。I2C 接口忽略处理 10 位的寻址。对于 7 位寻址，仅比较 IC_SAR 寄存器的低 7 位 1：10 位的寻址地址。I2C 仅响应 10 位的寻址，接收地址与 IC_SAR 的 10 位比较
2：1	SPEED	rw	0x00	该两位控制 I2C 接口工作的速率模式（These bits control at which speed the I2C operates） 该设置仅当 I2C 接口工作在主机模式下有效。 1：标准模式 (0 ~ 100Kbps) 2：快速模式 (400Kbps)
0	MASTER	rw	0x01	该位控制主机模式（This bit controls whether the I2C master is enabled） 0：主机禁止 1：主机使能

DISSLAVE(bit 6) 和 MASTER(bit 0) 配置如下表所列：

表 73. DISSLAVE(bit 6) 和 MASTER(bit 0) 配置

DISSLAVE CR[6]	MASTER CR[0]	状态
0	0	从机器件
0	1	配置错误
1	0	配置错误
1	1	主机器件

19.7.2 I2C 目标地址寄存器（I2C_TAR）

偏移地址：0x04

复位值：0x0055

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				SPECIAL	GC	ADDR									
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
15：12	Reserved			始终读为 0。
11	SPECIAL	rw	0x00	该位指示软件执行的是否是特殊命令（广播呼叫或者起始字节命令） 0：忽略第 10 位 GC，正常使用 ADDR 位 1：执行特殊 I2C 命令如 GC 位描述
10	GC	rw	0x00	如果位 11 置位，该位显示 I2C 执行的是广播呼叫还是起始字节（If bit 11(SPECIAL) is set to 1, then this bit indicates whether a General Call or START byte command is to be performed by the I2C） 0：广播呼叫地址。发送广播呼叫地址时只能执行写操作。I2C 接口一直工作在广播地址模式下直到 SPECIAL(bit 11) 的值被清零 1：起始字节命令
9：0	ADDR	rw	0x55	主操作的目标地址（This is the target address for any master transaction） 当发送一个广播地址，这些位就可以忽略。 要产生开始字节的命令，CPU 只需要对这些位写一次。

19.7.3 I2C 从机地址寄存器（I2C_SAR）

偏移地址：0x08

复位值：0x0055

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

Bit	Field	Type	Reset	Description
15：10	Reserved			始终读为 0。
9：0	ADDR	rw	0x55	当 I2C 接口工作在从机模式下，这些存储从机地址对于 7 位的地址格式，ADDR 只有 [6:0] 有效。

19.7.4 I2C 数据命令寄存器（I2C_DR）

偏移地址：0x10

复位值：0x0001

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved					RESTART	STOP	CMD	DAT							
					w	w	w	rw	rw	rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
15：11	Reserved			始终读为 0。
10	RESTART	w	0x00	发送或接收之前，是否产生一个 RESTART 信号 仅在 IC_EMPTYFIFO_HOLD_MASTER_EN 的配置为‘1’时有效。 1：如果 RESTART 信号为”1’，数据接收或发送（根据 CMD 的值）前产生一个 RESTART 的信号，无论前一个命令是否改变数据的传输方向。如果 IC_RESTART_EN 信号为‘0’，STOP 信号将紧跟 START 信号 0：如果 RESTART 信号为‘1’，仅在前一个命令改变传输方向时才产生 RESTART 信号。如果 RESTART 信号为‘0’，STOP 信号将紧跟 START 信号
9	STOP	w	0x00	STOP：发送或接收之后，是否产生一个 STOP 信号 仅在 IC_EMPTYFIFO_HOLD_MASTER_EN 的配置为”1”时有效。 1：当前字节之后产生一个 STOP 信号，无论 Tx FIFO 是否为空。如果 Tx FIFO 不为空，主机立即发出一个新的传输及总线仲裁信号 0：当前字节之后不产生一个 STOP 信号，无论 Tx FIFO 是否为空。主机继续当前传输（发送或接收数据根据 CMD 的值）。如果 Tx FIFO 为空，主机将拉低 SCL 挂起总线直至 Tx FIFO 收到新数据
8	CMD	w	0x00	控制在主模式下执行读或写操作 1：读 0：写 当一个命令进入 TX FIFO，该位用于区分读写命令。在从接收模式下，该位写值操作被忽略。在从发送模式下，写 0 表示 DR 寄存器的数据准备发送。
7：0	DAT	rw	0x01	I2C 总线待发送或者接收到的数据

19.7.5 标准模式 I2C 时钟高电平计数寄存器（I2C_SSHR）

偏移地址：0x14

复位值：0x0190

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

Bit	Field	Type	Reset	Description
15 : 0	CNT	rw	0x0190	I2C 接口标准模式下 SCL 时钟高电平周期 注意：该寄存器可配置值在 6 和 65525 之间，这是由于 I2C 接口使用了一个 16 位的计数器，该计数器值等于 SSHR + 10 时标志 I2C 总线处于空闲状态。

19.7.6 标准模式 I2C 时钟低电平计数寄存器（I2C_SSLR）

偏移地址：0x18

复位值：0x01D6

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

Bit	Field	Type	Reset	Description
15 : 0	CNT	rw	0x01D6	I2C 接口标准模式下 SCL 时钟低电平周期 最小值为 8。

19.7.7 快速模式 I2C 时钟高电平计数寄存器（I2C_FSHR）

偏移地址：0x1C

复位值：0x0036

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

Bit	Field	Type	Reset	Description
15 : 0	CNT	rw	0x0036	I2C 接口快速模式下 SCL 时钟高电平周期 当 I2C 工作在标准模式下该寄存器为只读且返回值为 0。 最小值为 6。

19.7.8 快速模式 I2C 时钟低电平计数寄存器（I2C_FSLR）

偏移地址：0x20

复位值：0x0082

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

Bit	Field	Type	Reset	Description
15：0	CNT	rw	0x0082	I2C 接口快速模式下 SCL 时钟低电平周期 当 I2C 工作在标准模式下该寄存器为只读且返回值为 0。 最小值为 8。

19.7.9 I2C 中断状态寄存器（I2C_ISR）

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
Reserved		D	RESTART	GC	START	STOP	ACTIV	RX_DONE	TX_ABRT	RD_REQ	TX_EMPTY	TX_OVER	RX_FULL	RX_OVER	RX_UNDER
		r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bit	Field	Type	Reset	Description
15：14	Reserved			始终读为 0。
13：0	ISR	r	0x0000	具体每位描述可以参考 RAWISR 寄存器

19.7.10 I2C 中断屏蔽寄存器（I2C_IMR）

偏移地址：0x30

复位值：0x08FF

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				GC	START	STOP	ACTIV	RX_DONE	TX_ABRT	RD_REQ	TX_EMPTY	TX_OVER	RX_FULL	RX_OVER	RX_UNDER
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
15：12	Reserved			始终读为 0。
11：0	IMR	rw	0x08FF	每一位屏蔽与 ISR 对应位。

19.7.11 I2C RAW 中断寄存器（I2C_RAWISR）

偏移地址：0x34

复位值：0x0000

RAWISR 与 ISR 寄存器的区别在于前者不会被屏蔽。

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				GC	START	STOP	ACTIV	RX_DONE	TX_ABRT	RD_REQ	TX_EMPTY	TX_OVER	RX_FULL	RX_OVER	RX_UNDER
				r	r	r	r	r	r	r	r	r	r	r	r

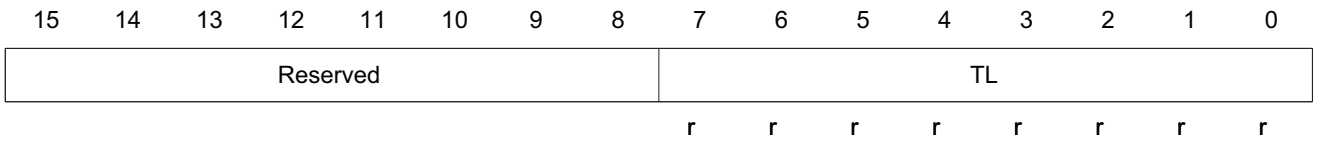
Bit	Field	Type	Reset	Description
15 : 12	Reserved			始终读为 0。
11	GC	r	0x00	广播呼叫（General call） 接收到广播呼叫地址时置位。 禁止 I2C 接口或者当 CPU 读 GC 寄存器时清零。I2C 将接收的数据存储在接收缓冲中。
10	START	r	0x00	起始条件检测（Start condition detection） 无论 I2C 接口工作在主机或者从机，一旦检测到 I2C 接口上起始或者重复起始条件即置位该位
9	STOP	r	0x00	停止条件检测（Stop condition detection） 该位状态依据 CR 寄存器的 STOPINT 的状态 当 STOPINT = 0 无论 I2C 接口工作在主机或者从机，一旦检测到 I2C 接口上停止条件时即置位该位。从机模式下，无论寻址是否匹配都会产生一个 STOP 中断 当 STOPINT = 1 在主机模式下（MASTER = 1），该位显示 I2C 接口是否发生停止条件 在从机模式下（MASTER = 0），仅当从机地址匹配成功时产生一个 STOP 中断。
8	ACTIV	r	0x00	I2C 接口激活，该位用于捕捉 I2C 模块的活动状态 置位后只能由以下四种方式清零： 禁止 I2C 接口 读 ACTIV 寄存器 读 ICR 寄存器 系统复位 一旦置位后，只能由上述方式清零，即使 I2C 处于空闲状态，该位也仍然保持为高直到被清零。
7	RX_DONE	r	0x00	从发送结束（Transmit done） 当 I2C 作为从发送时，如果发送一个字节的数据后主机没有响应，将会置位该位。 该情况发生在传输的最后一个字节，表示传输结束。

Bit	Field	Type	Reset	Description
6	TX_ABRT	r	0x00	发送中止（Transmit abort） 当 I2C 接口作为发送机时，不能发送完缓冲中的数据时置位。 注意：发送中止会将 I2C 接口中的接收和发送缓冲清空。发送缓冲会处于刷新状态直到读 TX_ABRT 寄存器。一旦该读操作执行后，发送就可以接收 APB 总线上的新的数据。
5	RD_REQ	r	0x00	读请求（Read request） 当 I2C 作为从机，其他主机试图从 I2C 接口读取数据时置位。 I2C 接口会使总线保持等待状态（SCL = 0）直到中断被处理。这就意味着 I2C 接口作为从机时被其他主机寻址成功且要求发送数据。处理器必须响应该中断然后写入数据到 DR 寄存器中。当处理器读 RD_REQ 寄存器该位清零。
4	TX_EMPTY	r	0x00	发送缓冲空（Transmit buffer empty） 该位状态取决于 CR 寄存器中的 EMPINT 状态： 当 EMPINT = 0，发送缓冲为空时置位 当 EMPINT = 1，发送缓冲为空且内部移位寄存器结束时置位 当发送缓冲非空时由硬件自动清零。
3	TX_OVER	r	0x00	发送缓冲过载（Transmit buffer over） 发送缓冲满时处理器写入新的数据导致溢出时置位。
2	RX_FULL	r	0x00	接收缓冲非空（Receive buffer not empty） 当接收缓冲非空时置位。 当接收缓冲为空时由硬件清零。
1	RX_OVER	r	0x00	接收缓冲过载（Receive buffer over） 接收缓冲满且有收到新的数据时置位。此时 I2C 接口会响应，但新的数据会丢失。
0	RX_UNDER	r	0x00	接收缓冲欠载（Receive buffer under） 当 RX FIFO 为空时处理器读 DR 寄存器时置位。

19.7.12 I2C 接收阈值（I2C_RXTLR）

偏移地址：0x38

复位值：0x0000



Bit	Field	Type	Reset	Description
15 : 8	Reserved			始终读为 0。
7 : 0	TL	r	0x00	接收 FIFO 阈值（Receive FIFO threshold level） 控制 RX_FULL 中断触发。

19.7.13 I2C 发送阈值（I2C_TXTLR）

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
Reserved								TL							
								r	r	r	r	r	r	r	r

Bit	Field	Type	Reset	Description
15 : 8	Reserved			始终读为 0。
7 : 0	TL	r	0x00	接收 FIFO 阈值（Receive FIFO threshold level） 控制 TX_EMPTY 中断触发。

19.7.14 I2C 组合和独立中断清除寄存器（I2C_ICR）

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
Reserved															ICR
															r

Bit	Field	Type	Reset	Description
15 : 1	Reserved			始终读为 0。
0	ICR	r	0x00	读该寄存器将会清除所有组合中断、独立中断 该位不清除硬件可自动清除的中断，仅清除软件可清除中断。

19.7.15 I2C 清除 RX_UNDER 中断寄存器（I2C_RX_UNDER）

偏移地址：0x44

复位值：0x0000

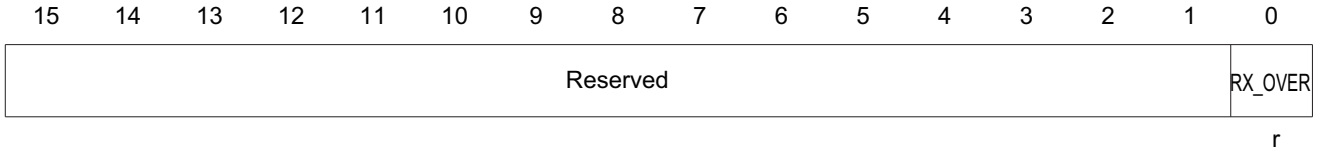
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															RX_UNDER
															r

Bit	Field	Type	Reset	Description
15 : 1	Reserved			始终读为 0。
0	RX_UNDER	r	0x00	读该寄存器清零 RX_UNDER 中断（RAWISR[0]）

19.7.16 I2C 清除 RX_OVER 中断寄存器（I2C_RX_OVER）

偏移地址：0x48

复位值：0x0000

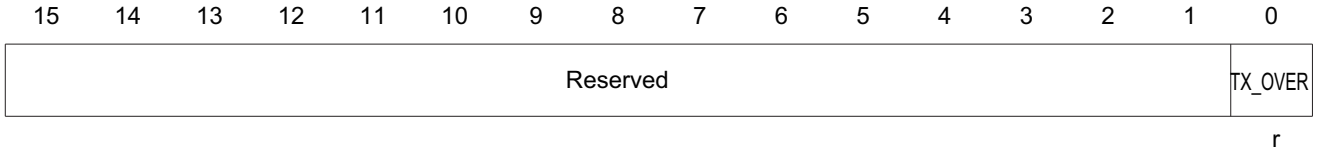


Bit	Field	Type	Reset	Description
15 : 1	Reserved			始终读为 0。
0	RX_OVER	r	0x00	读该寄存器清零 RX_OVER 中断（RAWISR[1]）

19.7.17 I2C 清除 TX_OVER 中断寄存器（I2C_TX_OVER）

偏移地址：0x4C

复位值：0x0000

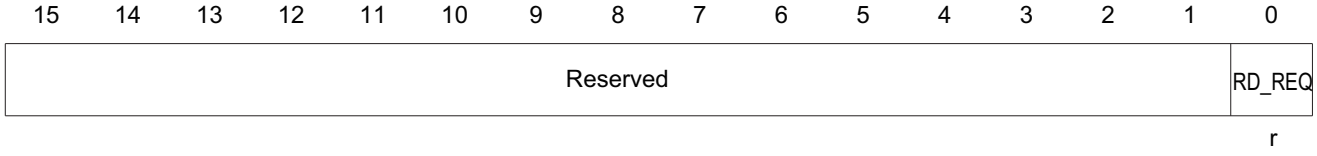


Bit	Field	Type	Reset	Description
15 : 1	Reserved			始终读为 0。
0	TX_OVER	r	0x00	读该寄存器清零 TX_OVER 中断（RAW_ISR[3]）

19.7.18 I2C 清除 RD_REQ 中断寄存器（I2C_RD_REQ）

偏移地址：0x50

复位值：0x0000



Bit	Field	Type	Reset	Description
15 : 1	Reserved			始终读为 0。
0	RD_REQ	r	0x00	读该寄存器清零 RD_REQ 中断（RAW_ISR[5]）

19.7.19 I2C 清除 TX_ABRT 中断寄存器（I2C_TX_ABRT）

偏移地址：0x54

复位值：0x0000

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

Bit	Field	Type	Reset	Description
15：1	Reserved			始终读为 0。
0	TX_ABRT	r	0x00	读该寄存器清零 TX_ABRT 中断（RAWISR6]） 同时也将 TX FIFO 从刷新/复位状态中释放，以便接收写入的数据。

19.7.20 I2C 清除 RX_DONE 中断寄存器（I2C_RX_DONE）

偏移地址：0x58

复位值：0x0000

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

Bit	Field	Type	Reset	Description
15：1	Reserved			始终读为 0。
0	RX_DONE	r	0x00	读该寄存器清零 RX_DONE 中断（RAWISR[7]）

19.7.21 I2C 清除 ACTIVITY 中断寄存器（I2C_ACTIV）

偏移地址：0x5C

复位值：0x0000

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

Bit	Field	Type	Reset	Description
15：1	Reserved			始终读为 0。

Bit	Field	Type	Reset	Description
0	ACTIV	r	0x00	如果 I2C 总线不活动则读该寄存器清零 ACTIV 中断（RAWISR[8]） 如果 I2C 仍然活动，那么 ACTIV 中断将继续置位。当 I2C 模块禁止或者在 I2C 总线不再活动时该位由硬件清零。可以通过读该寄存器得到 RAWISR 中的 ACTIV（bit 8）的状态。

19.7.22 I2C 清除 STOP_DET 中断寄存器（I2C_STOP）

偏移地址：0x60

复位值：0x0000

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

Bit	Field	Type	Reset	Description
15：1	Reserved			始终读为 0。
0	STOP	r	0x00	读该寄存器清零 STOP 中断（RAWISR[9]）

19.7.23 I2C 清除 START_DET 中断寄存器（I2C_START）

偏移地址：0x64

复位值：0x0000

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

Bit	Field	Type	Reset	Description
15：1	Reserved			始终读为 0。
0	START	r	0x00	读该寄存器清零 START 中断（RAWISR[10]）

19.7.24 I2C 清除 GEN_CALL 中断寄存器（I2C_GC）

偏移地址：0x68

复位值：0x0000

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

Bit	Field	Type	Reset	Description
15：1	Reserved			始终读为 0。
0	GC	r	0x00	读该寄存器清零 GC 中断（RAWISR[11]）

19.7.25 I2C 使能寄存器（I2C_ENR）

偏移地址：0x6C

复位值：0x0000

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

Bit	Field	Type	Reset	Description
15：2	Reserved			始终读为 0。
1	ABORT	rw	0x00	I2C 传输中止（I2C transfer abort） 0：中止没有发生或者已经结束 1：中止操作正在进行 I2C 模块作为主机时置位时可以软件中止 I2C 的传输。一旦置位不能立即清除。置位后 I2C 模块控制逻辑会在完成当前传输之后产生一个 STOP 条件和清空发送缓冲，中止操作之后产生 TX_ABRT 中断。 该 ABORT 位会在中止操作结束后自动清零。
0	ENABLE	rw	0x00	I2C 模块使能（I2C mode enable） 0：禁止 I2C 模块（发送和接收缓冲保持擦除状态） 1：使能 I2C 模块

19.7.26 I2C 状态寄存器（I2C_SR）

偏移地址：0x70

复位值：0x0006

该寄存器只读，指示当前传输和缓冲状态，状态位不产生中断。

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved									SLV_ACTIV	MST_ACTIV	RFF	RFNE	TFE	TFNE	ACTIV
									r	r	r	r	r	r	r

Bit	Field	Type	Reset	Description
15：7	Reserved			始终读为 0。
6	SLV _ACTIV	r	0x00	从机状态机活动状态位（Slave FSM activity status） 0：从机状态机处于 IDLE 状态，所以 I2C 从机部分不活动 1：从机状态机不处于 IDLE 状态，所以 I2C 从机部分活动

Bit	Field	Type	Reset	Description
5	MST _ACTIV	r	0x00	主机状态机活动状态位（Master FSM activity status） 0：主机状态机处于 IDLE 状态，所以 I2C 主机部分不活动 1：主机状态机不处于 IDLE 状态，所以 I2C 主机部分活动
4	RFF	r	0x00	接收缓冲满（Receive FIFO completely full） 0：接收缓冲未满 1：接收缓冲满
3	RFNE	r	0x00	接收缓冲非空（Receive FIFO not empty） 0：接收缓冲空 1：接收缓冲非空
2	TFE	r	0x01	发送缓冲空（Transmit FIFO completely empty） 0：发送缓冲非空 1：发送缓冲空
1	TFNF	r	0x01	发送缓冲未满（Transmit FIFO not full） 0：发送缓冲满 1：发送缓冲未满
0	ACTIV	r	0x00	I2C 位活动状态（I2C activity status） MST_ACTIV 位与 SLV_ACTIV 位相或的结果。

19.7.27 I2C 发送缓冲水平寄存器（I2C_TXFLR）

偏移地址：0x74

复位值：0x0000

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

Bit	Field	Type	Reset	Description
15 : 2	Reserved			始终读为 0。
1 : 0	CNT	r	0x00	发送缓冲中有效数据个数（0 ~ 2）

19.7.28 I2C 接收缓冲水平寄存器（I2C_RXFLR）

偏移地址：0x78

复位值：0x0000

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

Bit	Field	Type	Reset	Description
15 : 2	Reserved			始终读为 0。
1 : 0	CNT	r	0x00	接收缓冲中有效数据个数（0 ~ 2）

19.7.29 I2C SDA 保持时间寄存器（I2C_HOLD）

偏移地址：0x7C

复位值：0x0000 0001

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

Bit	Field	Type	Reset	Description
31 : 24	Reserved			始终读为 0。
23 : 16	RX_HOLD	r	0x00	当 I2C 器件作为接收时，SDA 保持时间，单位为 APB1 系统时钟周期
15 : 0	TX_HOLD	r	0x01	当 I2C 器件作为发送时，SDA 保持时间，单位为 APB1 系统时钟周期

19.7.30 I2C DMA 控制寄存器（I2C_DMA）

偏移地址：0x88

复位值：0x0000

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

Bit	Field	Type	Reset	Description
15 : 2	Reserved			始终读为 0。
1	TXEN	rw	0x00	发送 DMA 使能（Transmit DMA enable） 0：发送 DMA 禁止 1：发送 DMA 使能
0	RXEN	rw	0x00	接收 DMA 使能（Receive DMA enable） 0：接收 DMA 禁止 1：接收 DMA 使能

19.7.31 I2C SDA 建立时间寄存器（I2C_SETUP）

偏移地址：0x94

复位值：0x0064

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

Bit	Field	Type	Reset	Description
15：8	Reserved			始终读为 0。
7：0	CNT	rw	0x64	SDA 建立时间（SDA setup） 如果有要求建议延迟时间为 1000nS，APB1 时钟频率为 10MHZ 时，建议该寄存器设为 11。该寄存器最小值为 2。

19.7.32 I2C 广播呼叫 ACK 寄存器（I2C_GCR）

偏移地址：0x98

复位值：0x0001

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

Bit	Field	Type	Reset	Description
15：1	Reserved			始终读为 0。
0	GC	rw	0x01	广播呼叫 ACK（ACK general call） 1：接收到广播呼叫后响应 ACK 0：接收到广播呼叫后不响应，也不产生中断

20 通用异步收发器 (UART)

通用异步收发器 (UART)

20.1 UART 简介

通用异步收发器 (UART) 提供了一种灵活的方法与使用工业标准 NRZ 异步串行数据格式的外部设备之间进行全双工数据交换。UART 利用分数波特率发生器提供宽范围的波特率选择。调制解调器 (CTS/RTS) 操作。

使用多缓冲器配置的 DMA 方式，可以实现高速数据通信。

20.2 UART 主要特征

- 支持异步方式下 RS-232S 协议，符合工业标准 16550
- 支持 DMA 请求
- 全双工异步操作
- 分数波特率发生器系统
- 发送和接收共用的可编程波特率
- 单独分开的发送和接收缓冲寄存器
- 内置 1 字节发送和 32 字节接收缓冲
- 发送和接收数据低位在前
- 一个起始位开始，后面接数据位，输出的数据长度可为 5 位、6 位、7 位、8 位，最后为停止位。另外可选择是否有加奇偶校验位，奇偶校验位在数据位之后停止位之前。
- 支持硬件奇数或者偶数校验产生和侦测
- 线断开产生和侦测
- 支持硬件自动流控制
- 支持下面中断源：
  - 发送端 BUFFER 空
  - 接收端数据有效
  - 接收缓冲缓存溢出
  - 帧错误
  - 奇偶校验错误
  - 接收断开帧

20.3 UART 功能概述

任何 UART 双向通信至少需要两个脚：接收数据输入 (RX) 和发送数据输出 (TX)。

RX: 接收数据串行输入。通过过采样技术来区别数据和噪音，从而恢复数据。

TX: 发送数据输出。当发送器被禁止时，输出引脚恢复到它的 I/O 端口配置。当发送器被激活，并且不发送数据时，TX 引脚处于高电平。

- 总线在发送或接收前应处于空闲状态
- 一个起始位
- 一个数据字 (5, 6, 7 或 8 位)，最低有效位在前
- 1 或 2 个的停止位，由此表明数据帧的结束
- 使用分数波特率发生器——16 位整数和 4 位小数的表示方法。

下列引脚在硬件流控模式中需要:

- nCTS: 清除发送，若是高电平，在当前数据传输结束时阻断下一次的数据发送。
- nRTS: 发送请求，若是低电平，表明 UART 准备好接收数据。

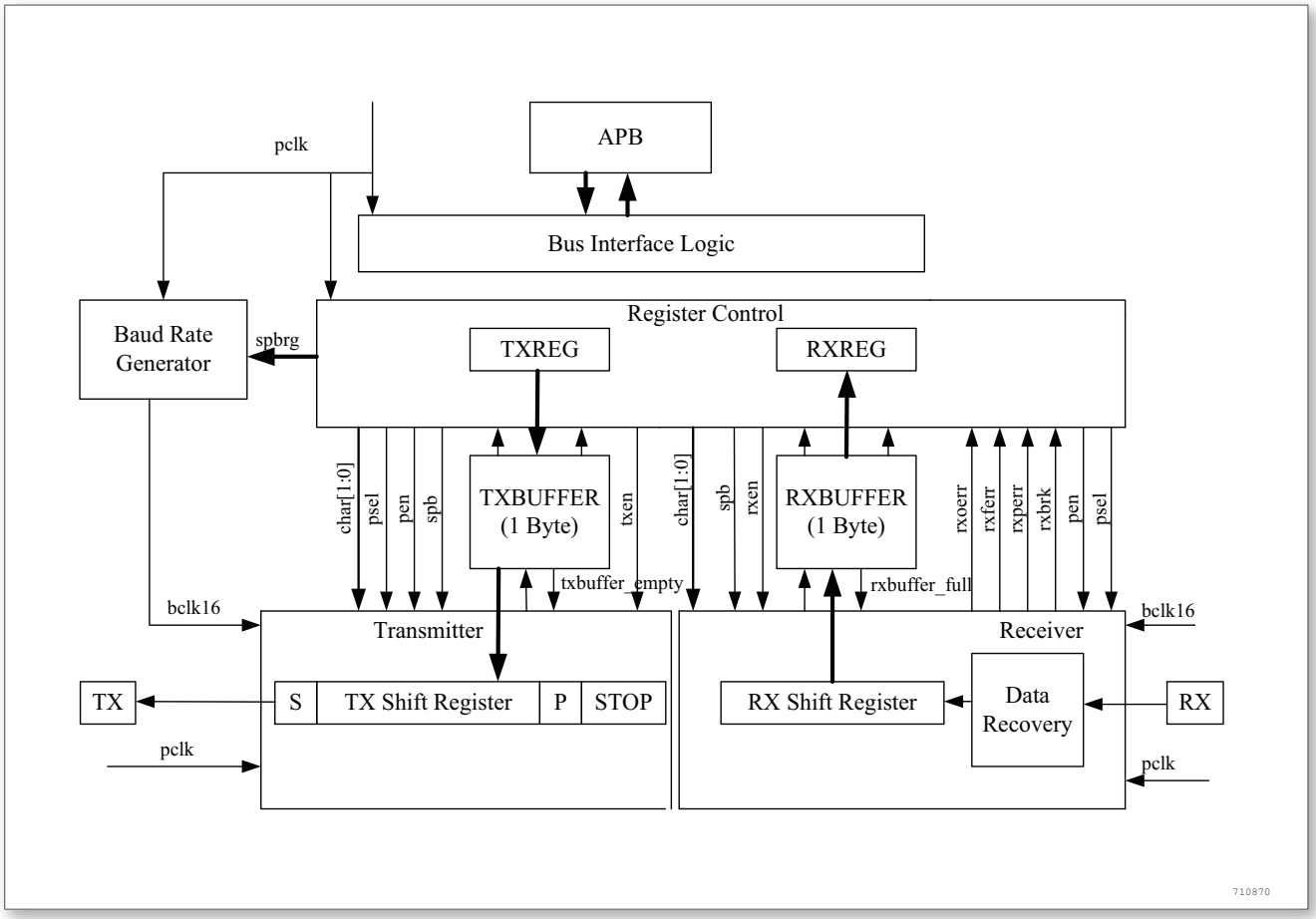


图 162. UART 方框图

20.3.1 UART 特性描述

字长可以通过编程 UART_CCR 寄存器中的 CHAR 位，选择 5 ~ 8 位。在起始位期间，TX 脚处于低电平，在停止位期间处于高电平。

空闲符号被视为完全由'1' 组成的一个完整的数据帧，后面跟着包含了数据的下一帧的开始位 ('1' 的位数也包括了停止位的位数)。

断开符号被视为在一个帧周期内全部收到'0'(包括停止位期间，也是'0')。在断开帧结束时，发送器再插入 1 或 2 个停止位 ('1') 来应答起始位。

发送和接收由一个共用的波特率发生器驱动，当发送器和接收器的使能位分别置位时，分别为其产生时钟。

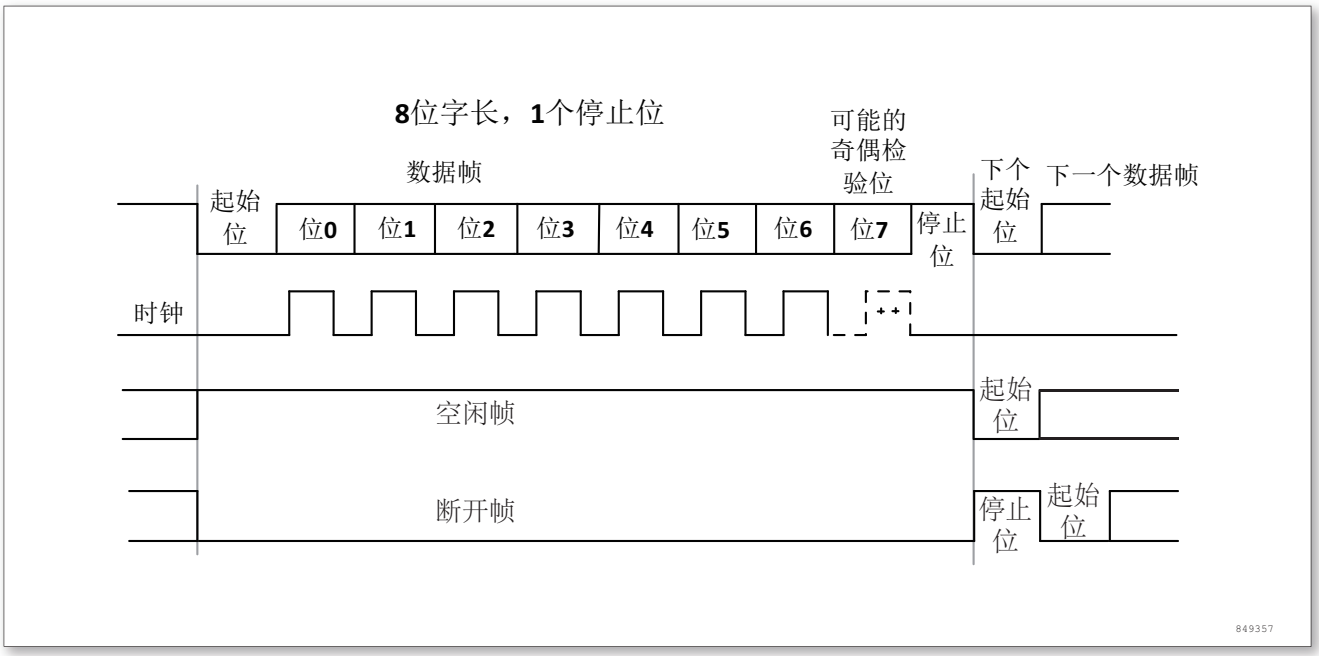


图 163. UART 时序

20.3.2 发送器

发送器根据 CHAR 位的状态发送 5 ~ 8 位的数据字。当发送使能位（TXEN）被设置时，发送移位寄存器中的数据在 TX 脚上输出。

字符发送

在 UART 发送期间，在 TX 引脚上首先移出数据的最低有效位。在此模式里，UART_TDR 寄存器包含了一个内部总线和发送移位寄存器之间的缓冲器。

每个字符之前都有一个低电平的起始位；之后跟着的停止位，其数目可配置。

在数据传输期间不能复位 TXEN 位，否则将破坏 TX 脚上的数据，因为波特率计数器停止计数。正在传输的当前数据将丢失。

可配置的停止位

随每个字符发送的停止位的位数可以通过 SPB 位进行编程。

断开帧是 10 位低电平，后跟停止位；或者 11 位低电平，后跟停止位。接收到断开帧会置位中断状态寄存器的 RXBRK_INTF 位。

配置步骤

1. 通过在 UART_GCR 寄存器上置位 UARTEN 位来激活 UART。
2. 编程 UART_CCR 的 CHAR 位来定义字长。
3. 在 UART_CCR 中 SPB 编程停止位的位数。
4. 设置 UART_GCR 中的 TXEN 位。
5. 利用 UART_BRR 寄存器选择要求的波特率。
6. 把要发送的数据写进 UART_TDR 寄存器（此动作清除 TX_INTF 位）。在只有一个缓冲器的情况下，对每个待发送的数据重复步骤 6。

单字节通信

清零 TX_INTF 位总是通过对数据寄存器的写操作来完成的。TX_INTF 位由硬件来设置，它表明：

- 数据已经从 TDR 移送到移位寄存器，数据发送已经开始
- TDR 寄存器被清空
- 下一个数据可以被写进 UART_TDR 寄存器而不会覆盖先前的数据。

如果 TXIEN 位被设置，此标志将产生一个中断。如果此时 UART 正在发送数据，对 UART_TDR 寄存器的写操作把数据存进 TDR 寄存器，并在当前传输结束时把该数据复制进移位寄存器。

如果此时 UART 没有在发送数据，处于空闲状态，对 UART_TDR 寄存器的写操作直接把数据放进移位寄存器，数据传输开始，TX_INTF 位立即被置起。同时 UART_CSR 的 TXBUF_EMPTY 也会置起。当一帧发送完成时（停止位发送后），同时没有往 UART_TDR 写入新的数据（TDR 寄存器为空），TXC 会置位，表示所有的传输都已经完成。

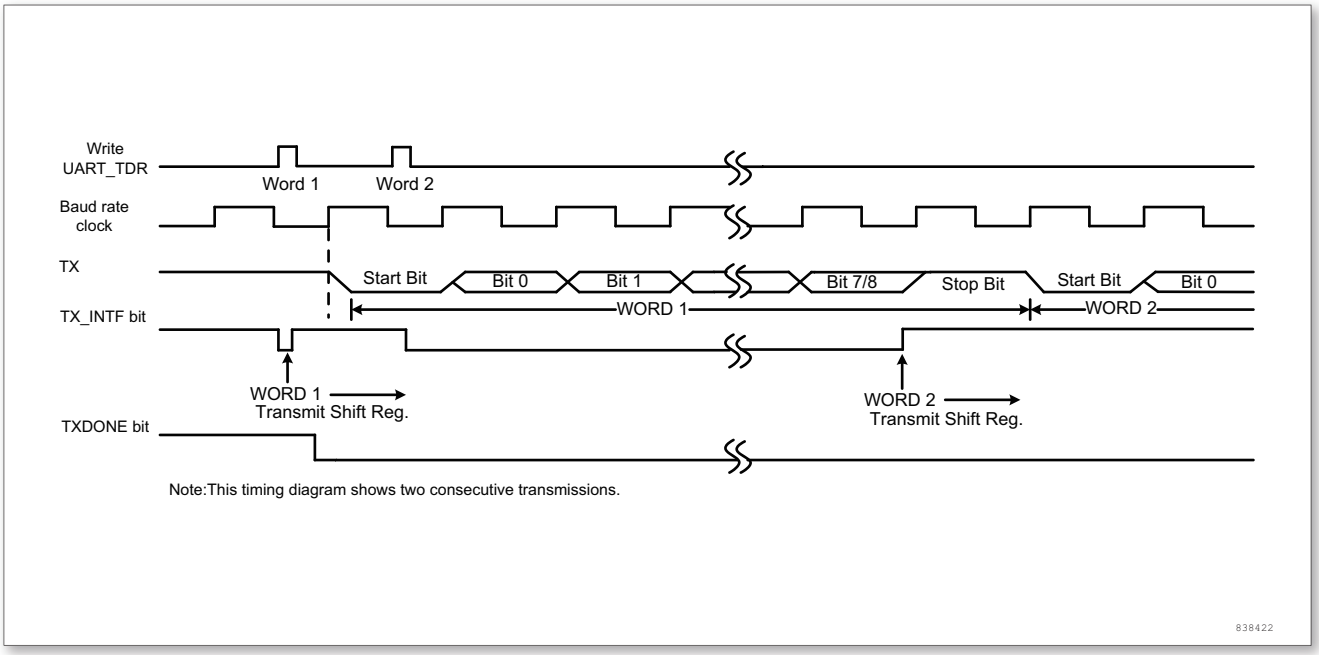


图 164. 发送时状态位变化

断开符号

设置 BRK 可发送一个断开符号。如果设置 BRK=1，在完成当前数据发送后，将在 TX 线上发送一个断开符号。断开字符发送完成时（在断开符号的停止位时）软件必须设置 BRK = 0。UART 在最后一个断开帧的结束处插入一逻辑‘1’，以保证能识别下一帧的起始位。

20.3.3 接收器

字符接收

在 UART 接收期间，数据的最低有效位首先从 RX 脚移进。在此模式里，UART_RDR 寄存器包含的缓冲器位于内部总线和接收移位寄存器之间。

配置步骤：

1. 将 UART_GCR 寄存器的 UARTEN 置‘1’来激活 UART。

2. 编程 UART_CCR 的 CHAR 位定义字长。
3. 在 UART_CCR 中 SPB 编程停止位的位数。
4. 利用 UART_BRR 寄存器选择要求的波特率。
5. 设置 UART_GCR 的 RXEN 位。激活接收器，使它开始寻找起始位。

当一字符被接收到时，

- RX_INTF 位被置位。它表明移位寄存器的内容被转移到 RDR。换句话说，数据已经被接收并且可以被读出（包括与之有关的错误标志）。
- 如果 RXIEN 位被设置，产生中断。
- 在接收期间如果检测到帧错误，或溢出错误，错误标志将被置起。
- 软件读 UART_RDR 寄存器。RX_INTF 位必须在下一字符接收结束前被清零。

在接收数据时，RXEN 位不应该被复位。如果 RXEN 位在接收时被清零，当前字节的接收被丢失。

断开符号

当接收到一个断开帧时，UART 会置位 RXBRK_INTF 中断。

溢出错误

如果在 UART_RDR 没有读出前又接收到一个字符，则发生溢出错误。

当溢出错误产生时：

- RXOERR_INTF 位被置位。
- RDR 内容将不会丢失。读 UART_RDR 寄存器仍能得到先前的数据。
- 移位寄存器中以前的内容将被覆盖。随后接收到的数据都将丢失。
- 如果 RXOERREN 位被设置，中断产生。

帧错误

当停止位没有在预期的时间上接收和识别出来时检测到帧错误。当帧错误被检测到时：

- RXFERR_INTF 位被硬件置起。
- 无效数据不会从移位寄存器传送到 UART_RDR 寄存器。
- 如果 RXFERREN 位被设置，中断产生。

20.3.4 分数波特率发生器

设置 BRR 和 FRA 寄存器，可设置相应波特率，参考如下公式：

$$f_{baudrate} = \frac{f_{PCLK}}{16 \times UARTDIV}$$

$$UARTDIV = BRR + \frac{FRA}{16}$$

得：

$$f_{baudrate} = \frac{f_{PCLK}}{16 \times BRR + FRA}$$

BRR 寄存器最小值为 4。

20.3.5 采样

由于异步操作没有单独的时钟，接收器需要一个同步于接收器方法。为了能够在接收引脚‘RX’获得正确的字符数据，UART 有一个检测电路。UART 采用 16 倍数据波特率‘bclk16’的时钟进行采样 RX 引脚的数据，每个数据有 16 个时钟采样，取中间第 7, 8, 9 的下降沿的采样值。

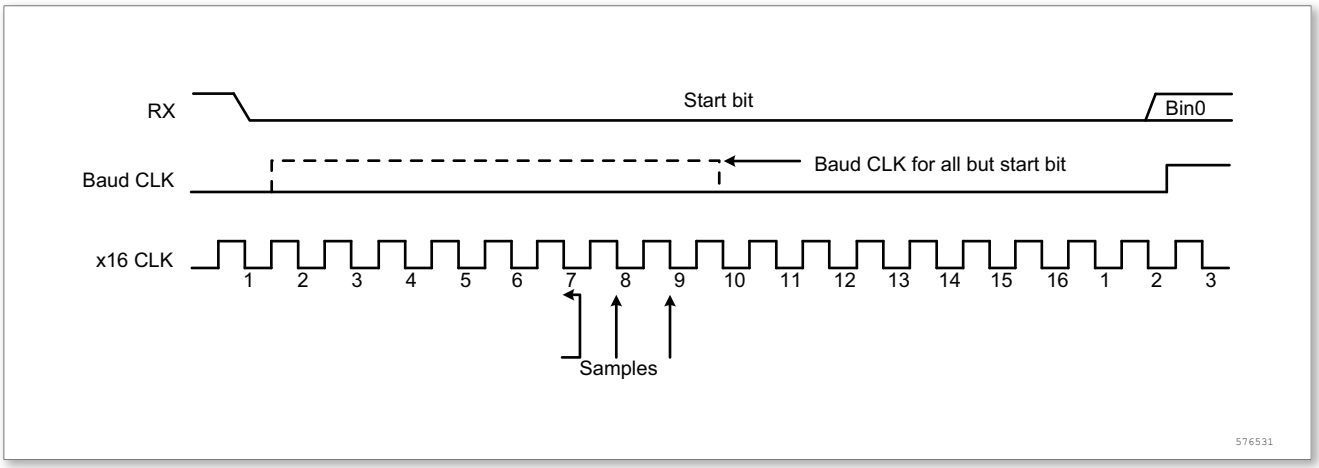


图 165. RX 引脚采样方案

20.3.6 校验控制

奇偶控制 (发送时生成一个奇偶位，接收时进行奇偶校验) 可以通过设置 UART_CCR 寄存器上的 PEN 位而激活。如果奇偶校验出错，无效数据不会从移位寄存器传送到 UART_RDR 寄存器。

偶校验：校验位使得一帧中的数据以及校验位中‘1’的个数为偶数。

例如：数据 = 00110101，有 4 个‘1’，如果选择偶校验（在 UART_CCR 中的 PSEL = 1），校验位将是‘0’。

奇校验：此校验位使得一帧中的数据以及校验位中‘1’的个数为奇数。

例如：数据 = 00110101，有 4 个‘1’，如果选择奇校验（在 UART_CCR 中的 PSEL = 0），校验位将是‘1’。

传输模式: 如果 UART_CCR 的 PEN 位被置位，奇偶校验位将在数据寄存器数据发送完毕后发出 (如果选择偶校验偶数个‘1’，如果选择奇校验奇数个‘1’)。如果奇偶校验失败，UART_ISR 寄存器中的 RXPERR_INTF 标志被置‘1’，并且如果 RXPERREN 在被预先设置的话，中断产生。

20.3.7 硬件流控制

利用 nCTS 输入和 nRTS 输出可以控制 2 个设备间的串行数据流。下图表明在这个模式里如何连接 2 个设备。

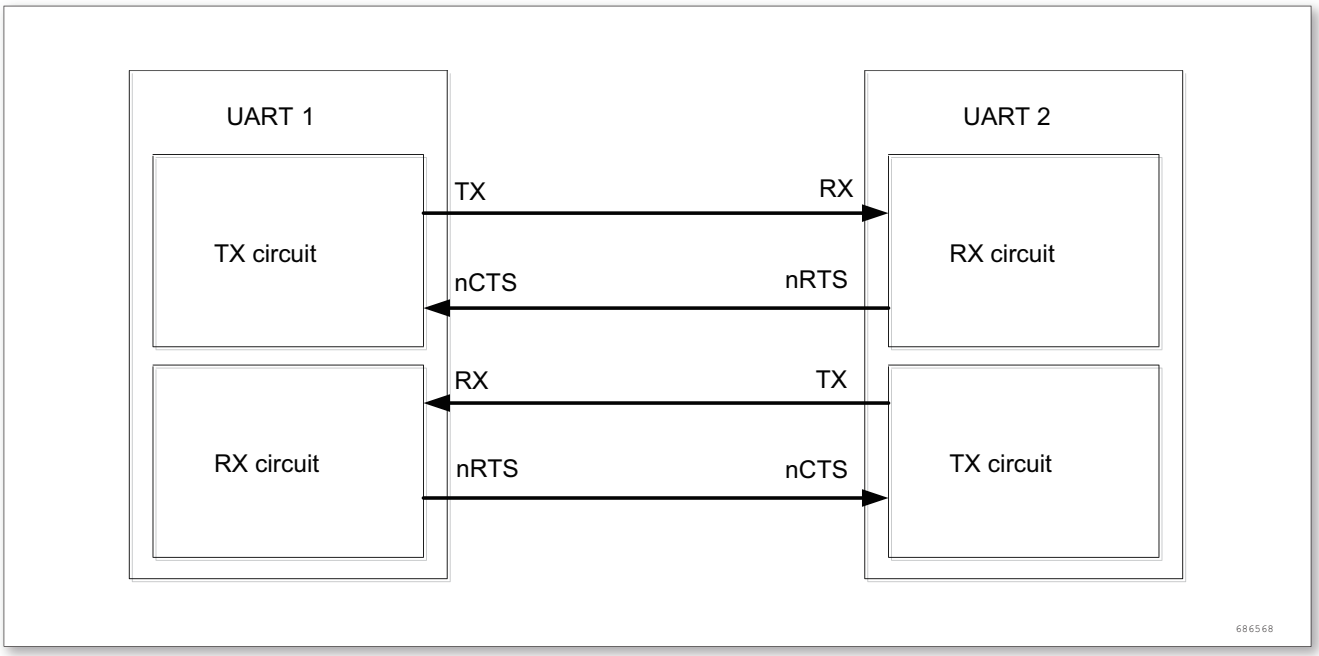


图 166. 两个 UART 间的硬件流控制

通过将 UART_GCR 中的 AUTOFLOWEN 置位，可以使能 RTS 和 CTS 流控制。

RTS 流控制

如果 RTS 流控制被使能，只要 UART 接收器准备好接收新的数据，nRTS 就变成有效 (接低电平)。当接收寄存器内有数据到达时，nRTS 被释放，由此表明希望在当前帧结束时停止数据传输。下图是一个启用 RTS 流控制的通信的例子。

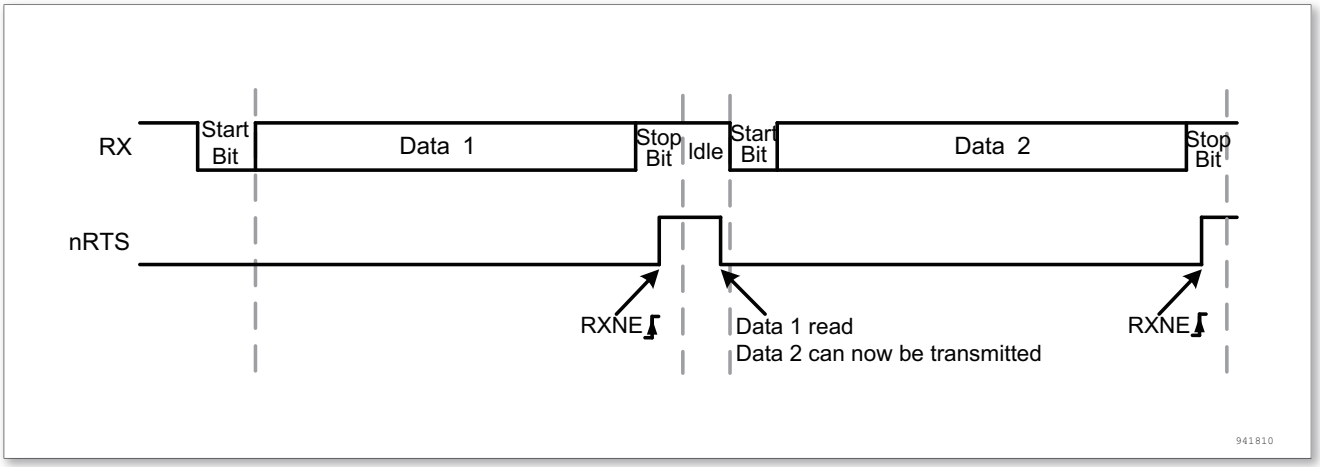


图 167. RTS 流控制

CTS 流控制

如果 CTS 流控制被使能，发送器在发送下一帧前检查 nCTS 输入。如果 nCTS 有效 (被拉成低电平)，则下一个数据被发送 (假设那个数据是准备发送的)，否则下一帧数据不被发出去。若 nCTS 在传输期间被变成无效，当前的传输完成后停止发送。下图是一个 CTS 流控制被启用的通信的例子。

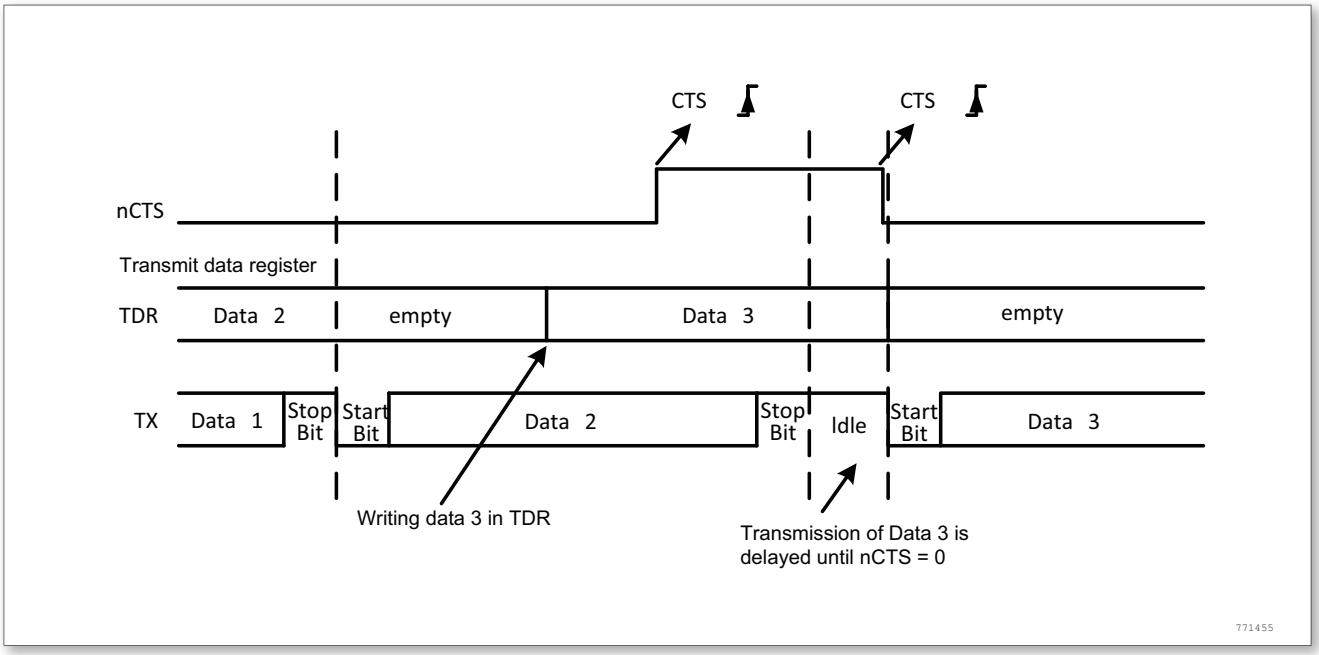


图 168. CTS 流控制

20.3.8 利用 DMA 通信

UART 可以利用 DMA 进行通信。

利用 DMA 发送

使用 DMA 进行发送时，首先在 DMA 控制寄存器上将 UART_TDR 寄存器的地址配置成 DMA 传输的目的地址，将存储器地址配置成 DMA 传输的源地址，并配置传输的数据量。通过设置 UART_GCR 寄存器的 DMAMODE 位来激活 DMA 模式。当 TXEN 位被置‘1’时，DMA 就从指定的 SRAM 区传送数据到 UART_TDR 寄存器。

利用 DMA 接收

使用 DMA 进行接收时，首先在 DMA 控制寄存器上将 UART_RDR 寄存器的地址配置成 DMA 传输的源地址，将存储器地址配置成 DMA 传输的目的地址，并配置传输的数据量。通过设置 UART_GCR 寄存器的 DMAMODE 位来激活 DMA 模式。当 RXEN 位使能时，每接收到一个字节，DMA 就把数据从 UART_RDR 寄存器传送到指定的 SRAM 区。

20.4 UART 中断请求

表 74. UART 中断请求

中断事件	中断状态	使能位
发送缓冲空	TX_INTF	TXIEN
接收到有效数据	RX_INTF	RXIEN
接收溢出错误	RXOERR_INTF	RXOERREN
奇偶校验错误	RXPERR_INTF	RXPERREN
帧错误	RXFERR_INTF	RXFERREN
接收断开帧	RXBRK_INTF	RXBRKEN

如果设置了对应的中断使能控制位，这些设置就可以产生各自对应的中断。

20.5 UART 寄存器描述

表 75. UART 寄存器概览

Offset	Acronym	Register Name	Reset	Section
0x00	UART_TDR	UART 发送数据寄存器	0x00000000	小节 20.5.1
0x04	UART_RDR	UART 接收数据寄存器	0x00000000	小节 20.5.2
0x08	UART_CSR	UART 当前状态寄存器	0x00000009	小节 20.5.3
0x0C	UART_ISR	UART 中断状态寄存器	0x00000000	小节 20.5.4
0x10	UART_IER	UART 中断使能寄存器	0x00000000	小节 20.5.5
0x14	UART_ICR	UART 中断清除寄存器	0x00000000	小节 20.5.6
0x18	UART_GCR	UART 全局控制寄存器	0x00000000	小节 20.5.7
0x1C	UART_CCR	UART 通用控制寄存器	0x00000030	小节 20.5.8
0x20	UART_BRR	UART 波特率寄存器	0x00000001	小节 20.5.9
0x24	UART_FRA	UART 分数波特率寄存器	0x00000000	小节 20.5.10

20.5.1 UART 发送数据寄存器（UART_TDR）

偏移地址：0x00

复位值：0x0000 0000

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

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

Bit	Field	Type	Reset	Description
31 : 8	Reserved			始终读为 0。
7 : 0	TXREG	rw	0x00	发送数据寄存器（Transmit data register）

20.5.2 UART 接收数据寄存器（UART_RDR）

偏移地址：0x04

复位值：0x0000 0000

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

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

Bit	Field	Type	Reset	Description
31：8	Reserved			始终读为 0。
7：0	RXREG	r	0x00	接收数据寄存器（Receive data register） 该寄存器只读

20.5.3 UART 当前状态寄存器（UART_CSR）

偏移地址：0x08

复位值：0x0000 0009

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												TXBUF_EMPTY	TXFULL	RXAVL	TXC
												r	r	r	r

Bit	Field	Type	Reset	Description
31：4	Reserved			始终读为 0。
3	TXBUF_ EMPTY	r	0x01	发送缓冲空标识位（Transmit buffer empty flag bit） 1：发送缓冲为空 0：发送缓冲不为空
2	TXFULL	r	0x00	发送缓冲满满标志位（Transmit buffer full flag bit） 1：发送缓冲为满 0：发送缓冲不满
1	RXAVL	r	0x00	接收有效字节数据标识位（Receive valid data flag bit） 当接收缓冲接收了一个完整字节的数据时置位该位。 1：接收缓冲接收了一个完整有效的字节数据 0：接收缓冲为空
0	TXC	r	0x01	发送结束标识位（Transmit complete flag bit） 1：发送缓冲和发送移位寄存器都为空 0：发送不为空

20.5.4 UART 中断状态寄存器（UART_ISR）

偏移地址：0x0C

复位值：0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved									RXBRK_INTF	RXFERR_INTF	RXPERR_INTF	RXOERR_INTF	Res.	RX_INTF	TX_INTF
									r	r	r	r		r	r

Bit	Field	Type	Reset	Description
31：7	Reserved			始终读为 0。
6	RXBRK_INTF	r	0x00	UART 接收断开帧中断标志位（Receive frame break interrupt flag bit） 在异常停止位后 RX 引脚在一段时间内接收到 10 个或大于 10 位的低电平。 1：检测断开帧 0：没有断开帧
4	RXPERR_INTF	r	0x00	奇偶校验错误中断标志位（Parity error interrupt flag bit） 1：检测到奇偶校验错误 0：没有奇偶校验错误
3	RXOERR_INTF	r	0x00	接收溢出错误中断标志位（Receive overflow error interrupt flag bit） 仅当 autoflowen=0 时置位。 1：接收溢出错误 0：没有溢出错误
2	Reserved			始终读为 0。
1	RX_INTF	r	0x00	接收有效数据中断标志位（Receive valid data interrupt flag bit） 当接收缓冲接收了一个完整字节的数据时置位该位。 1：接收缓冲有效字节数据 0：接收缓冲为空
0	TX_INTF	r	0x00	发送缓冲空中断标志位（Transmit buffer empty interrupt flag bit） 1：发送缓冲空 0：发送缓冲不为空

20.5.5 UART 中断使能寄存器（UART_IER）

偏移地址：0x10

复位值：0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved									RXBRKEN	RXFERREN	RXPERREN	RXOERREN	Res.	RXIEN	TXIEN
									rw	rw	rw	rw		rw	rw

Bit	Field	Type	Reset	Description
31：7	Reserved			始终读为 0。
6	RXBRKEN	rw	0x00	UART 接收断开帧中断使能位（Receive frame break interrupt enable bit） 1：中断使能 0：中断禁止
5	RXFERREN	rw	0x00	帧错误中断使能位（Frame error interrupt enable bit） 1：中断使能 0：中断禁止
4	RXPERREN	rw	0x00	奇偶校验错误中断使能位（Parity error interrupt enable bit） 1：中断使能 0：中断禁止
3	RXOERREN	rw	0x00	接收溢出错误中断使能位（Receive overflow error interrupt enable bit） 1：中断使能 0：中断禁止
2	Reserved			始终读为 0。
1	RXIEN	rw	0x00	接收缓冲中断使能位（Receive buffer interrupt enable bit） 1：中断使能 0：中断禁止
0	TXIEN	rw	0x00	发送缓冲空中断使能位（Transmit buffer empty interrupt enable bit） 1：中断使能 0：中断禁止

20.5.6 UART 中断清除寄存器（UART_ICR）

偏移地址：0x14

复位值：0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved									RX BRK CLR	RXF ERR CLR	RXP ERR CLR	RXO ERR CLR	Res.	RXICLR	TXICLR
									w	w	w	w		w	w

Bit	Field	Type	Reset	Description
31：7	Reserved			始终读为 0。
6	RXBRKCLR	w	0x00	UART 接收断开帧中断清除位（Receive frame break interrupt clear bit） 1：中断清除 0：中断没有清除
5	RXFERRCLR	w	0x00	帧错误中断使能位（Frame error interrupt enable bit） 1：中断清除 0：中断没有清除
4	RXPERRCLR	w	0x00	奇偶校验错误中断清除位（Parity error interrupt clear bit） 1：中断清除 0：中断没有清除
3	RXOERRCLR	w	0x00	接收溢出错误中断清除位（Receive overflow error interrupt clear bit） 1：中断清除 0：中断没有清除
2	Reserved			始终读为 0。
1	RXICLR	w	0x00	接收中断清除位（Receive interrupt clear bit） 1：中断清除 0：中断没有清除
0	TXICLR	w	0x00	发送缓冲空中断清除位（Transmit buffer empty interrupt clear bit） 1：中断清除 0：中断没有清除

20.5.7 UART 全局控制寄存器（UART_GCR）

偏移地址：0x18

复位值：0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved											TXEN	RXEN	AUTO FLOW EN	DMA MODE	UARTEN
											rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
31：5	Reserved			始终读为 0。
4	TXEN	rw	0x00	发送使能位（Enable transmit） 1：发送使能 0：发送禁止。可以清除 TX BUFFER
3	RXEN	rw	0x00	接收使能位（Enable receive） 1：接收使能 0：接收禁止。可以清除 RX BUFFER.
2	AUTO FLOWEN	rw	0x00	自动流控制使能位（Automatic flow control enable bit） 1：自动流控制使能 0：自动流控制禁止
1	DMAMODE	rw	0x00	DMA 方式选择位（DMA mode selection bit） 1：选择 DMA 方式 0：选择正常方式
0	UARTEN	rw	0x00	UART 模块选择位（UART mode selection bit） 1：UART 模块使能 0：UART 模块禁止

20.5.8 UART 通用控制寄存器（UART_CCR）

偏移地址：0x1C

复位值：0x0000 0030

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved										CHAR		BRK	SPB	PSEL	PEN
										rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
31：6	Reserved			始终读为 0。

Bit	Field	Type	Reset	Description
5：4	CHAR	rw	0x03	UART 数据宽度位（UART width bit） 00: 5 位　　01: 6 位 10: 7 位　　11: 8 位
3	BRK	rw	0x00	UART 发送断开帧（UART transmit frame break） 1：串行强制输出逻辑‘0’（断开帧） 0：禁止断开
2	SPB	rw	0x00	停止位选择（Stop bit selection） 设置发送停止位位数。接收器通常测查一个停止位. 1：2 个停止位（5 位数据位时，不使用 SPB 设置，停止位强制为 1 位） 0：1 个停止位
1	PSEL	rw	0x00	校验选择位（Parity selection bit） 当校验使能后，该位用于选择是采用偶校验还是奇校验。 1：偶校验 0：奇校验
0	PEN	rw	0x00	校验使能位（Parity enable bit） 1：发送接收使能校验 0：禁止校验

20.5.9　UART 波特率寄存器（UART_BRR）

偏移地址：0x20

复位值：0x0000 0001

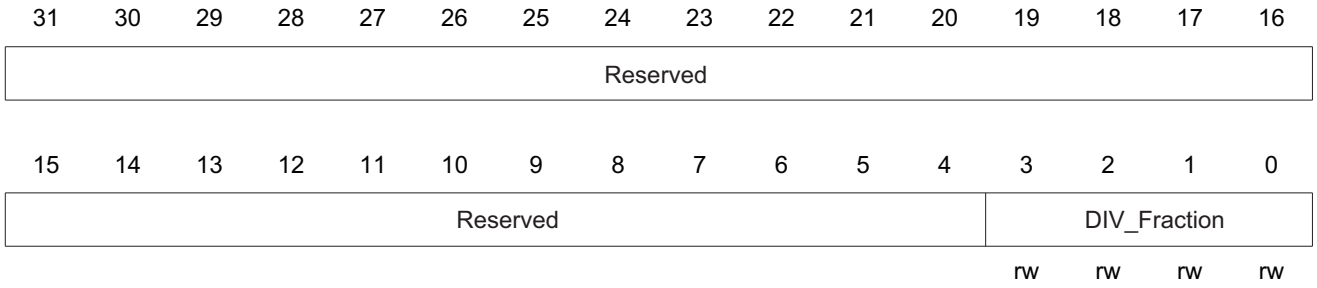
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DIV_Mantissa															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
31：16	Reserved			始终读为 0。
15：0	DIV_Mantissa	rw	0x0001	UARTDIV 的整数部分 这 16 位定义了 UART 分频器除法因子（UARTDIV）的整数部分。 DIV_Mantissa 最小值为 4

20.5.10 UART 分数波特率寄存器（UART_FRA）

偏移地址：0x24

复位值：0x0000 0000



Bit	Field	Type	Reset	Description
31 : 4	Reserved			始终读为 0。
3 : 0	DIV_Fraction	rw	0x00	UARTDIV 的小数部分 这 4 位定义了 UART 分频器除法因子（UARTDIV）的小数部分。

21 器件电子签名 (Device)

器件电子签名 (Device)

电子签名存放在闪存存储器模块的系统存储区域，可以通过 JTAG、SWD 或者 CPU 读取。它所包含的芯片识别信息在出厂时编写，用户固件或者外部设备可以读取电子签名，用以自动匹配不同配置的微控制器。

21.1 存储器容量寄存器

21.1.1 产品唯一身份标识寄存器 (96 位)

产品唯一的身份标识非常合适：

- 用来作为序列号 (例如 USB 字符序列号或者其他的终端应用)。
- 用来作为密码，在编写闪存时，将此唯一标识与软件加解密算法结合使用，提高代码在闪存存储器的安全性。
- 用来激活带安全机制的自举过程。

96 位的产品唯一身份标识所提供的参考号码对任意一个系列微控制器，在任何情况下都是唯一的。用户在何种情况下，都不能修改这个身份标识。

这个 96 位的产品唯一身份标识，按照用户不用的用法，可以以字节 (8 位) 为单位读取，也可以以半字 (16 位) 或者全字 (32 位) 读取。

21.2 UID 寄存器描述

表 76. 存储器容量寄存器描述概览

Offset	Acronym	Register Name	Reset	Section
0x00	UID1	唯一标识码	0xXXXXXXXX	小节 21.2.1
0x02	UID2	唯一标识码	0xXXXXXXXX	小节 21.2.2
0x04	UID3	唯一标识码	0xXXXXXXXX	小节 21.2.3
0x08	UID4	唯一标识码	0xXXXXXXXX	小节 21.2.4

21.2.1 唯一标识码 (UID1)

基地址：0x1FFF F7E8

地址偏移：0x00

只读，其值在出厂时编写

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

Bit	Field	Type	Reset	Description
15 :0	U_ID	r		U_ID：唯一身份标志 15：0 位 (15：0 unique ID bits) 这个域的数值也预留作为未来的其他功能。

21.2.2 唯一标识码 (UID2)

地址偏移：0x02

只读，其值在出厂时编写

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

Bit	Field	Type	Reset	Description
15 :0	U_ID	r		U_ID：唯一身份标志 31：16 位 (31：16 unique ID bits) 这个域的数值也预留作为未来的其他功能。

21.2.3 唯一标识码 (UID3)

地址偏移：0x04

只读，其值在出厂时编写

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

Bit	Field	Type	Reset	Description
31 :0	U_ID	r		U_ID：唯一身份标志 63：32 位 (63：32 unique ID bits) 这个域的数值也预留作为未来的其他功能。

21.2.4 唯一标识码 (UID4)

地址偏移：0x08

只读，其值在出厂时编写

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

Bit	Field	Type	Reset	Description
31：0	U_ID	r		U_ID：唯一身份标志 95：64 位 (95：64 unique ID bits) 这个域的数值也预留作为未来的其他功能。

22 调试支持 (DBG)

调试支持 (DBG)

22.1 概述

该系列内核内含硬件调试模块，支持复杂的调试操作。硬件调试模块允许内核在取指（指令断点）或访问数据（数据断点）时停止。内核停止时，内核的内部状态和系统的外部状态都是可以查询的。完成查询后，内核和外设可以被复原，程序将继续执行。

当该系列微控制器连接到调试器并开始调试时，调试器将使用内核的硬件调试模块进行调试操作。

支持两种调试接口：

- 串行接口
- JTAG 调试接口

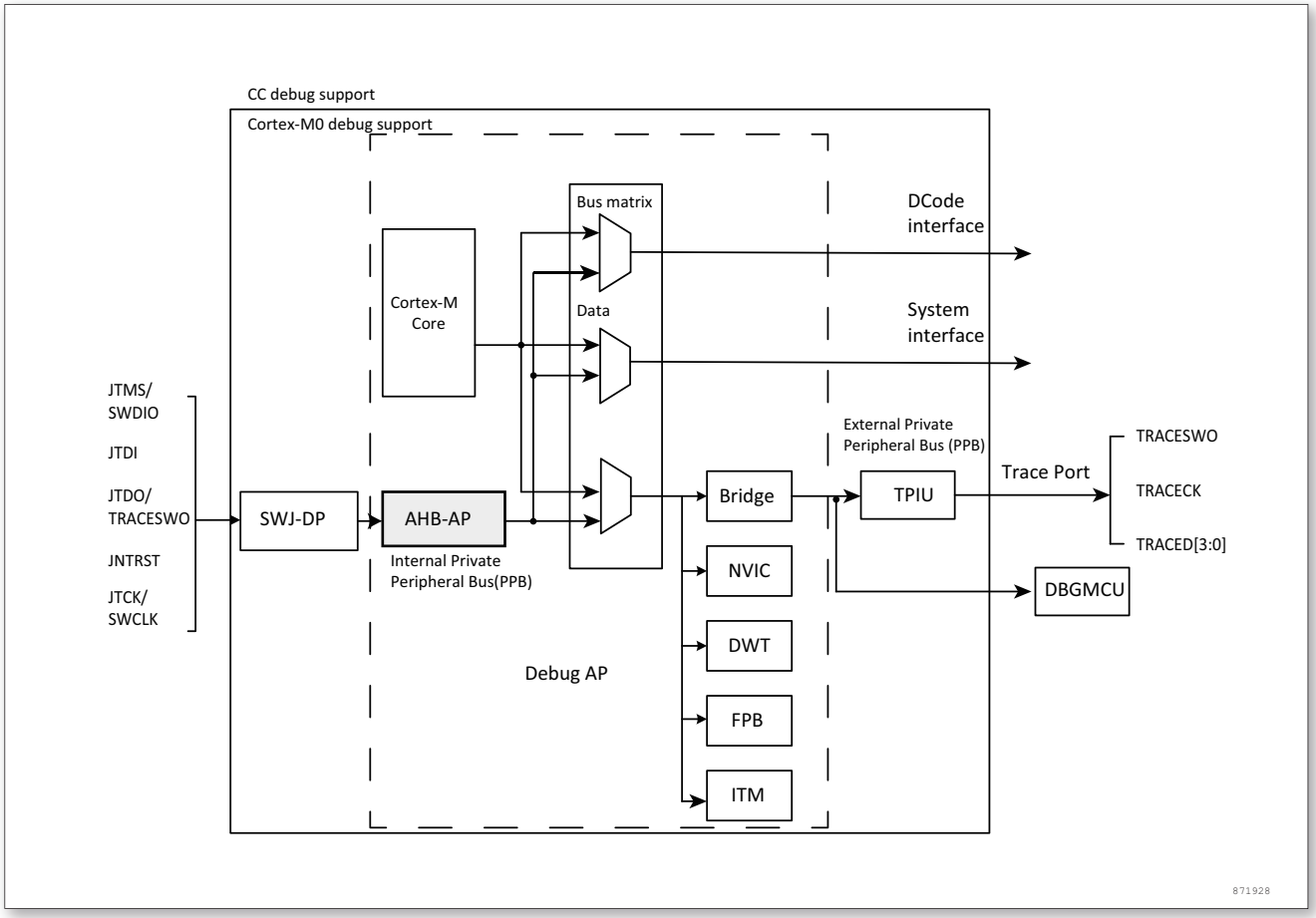


图 169. MM32 系列级别和 CPU 级别的调试框

CPU 内核提供集成的片上调试功能。它由以下部分组成：

- SWJ-DP：串行/JTAG 调试端口

- AHP-AP：AHB 访问端
- ITM：执行跟踪单元
- FPB：闪存指令断点
- DWT：数据触发
- TPUI：跟踪单元接口

22.2 SWJ 调试端口（serial wire and JTAG）

该芯片内核集成了串行/JTAG 调试接口（SWJ-DP）。这是标准的 CoreSight 调试接口，包括 JTAG-DP 接口（5 个引脚）和 SW-DP 接口（2 个引脚）。

- JTAG 调试接口（JTAG-DP）为 AHP-AP 模块提供 5 针标准 JTAG 接口。
- 串行调试接口（SW-DP）为 AHP-AP 模块提供 2 针（时钟 + 数据）接口。

在 SWJ-DP 接口中，SW-DP 接口的 2 个引脚和 JTAG 接口的 5 个引脚中的一些是复用的。

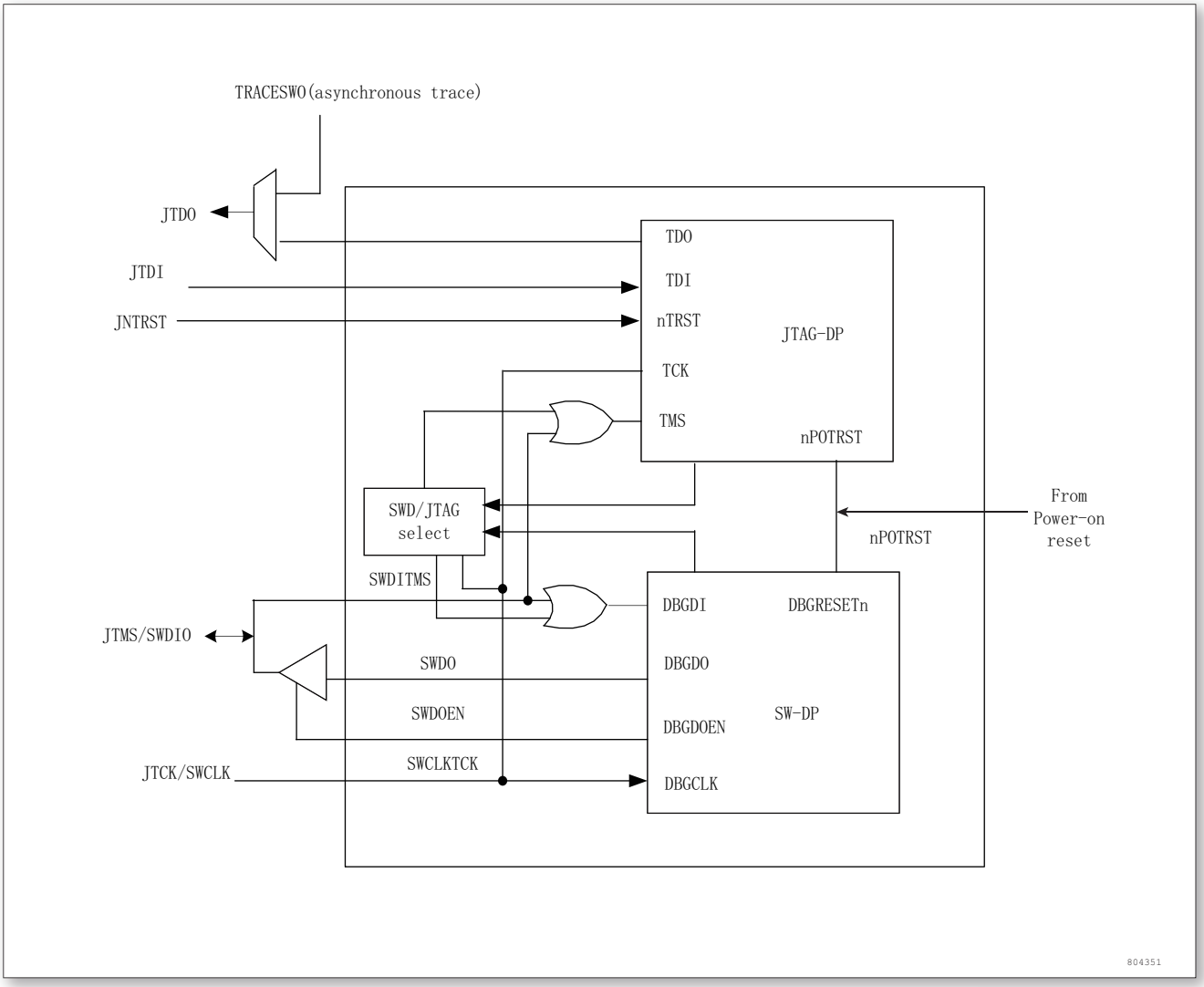


图 170. SWJ 调试端口

上面的图显示异步跟踪输出脚（TRACESWO）和 TDO 是复用的，因此异步跟踪功能只能在 SW-DP 调试接口上实现，不能在 JTAG-DP 调试接口上实现。

22.2.1 JTAG-DP 和 SW-DP 切换的机制

该芯片支持 SW-DP 和 JTAG-DP 两种调试接口，JTAG 调试接口是默认的调试接口。

如果调试器想要切换到 SW-DP ，必须在 TMS/TCK 上输出一指定的 JTAG 序列（分别映射到 SWDIO 和 SWCLK），该序列禁止 JTAG-DP ，并激活 SW-DP。该方法可以只通过 SWCLK 和 SWDIO 两个引脚来激活 SW-DP 接口。

指定的序列是：

- 输出超过 50 个 TCK 周期的 TMS（SWDIO）= 1 信号
- 出 16 个 TMS（SWDIO）信号 0111100111100111（MSB）
- 输出超过 50 个 TCK 周期的 TMS（SWDIO）= 1 信号

22.3 引脚分布和调试端口脚

该芯片微控制器的不同封装有不同的有效引脚数。因此，某些与引脚相关的功能可能随封装而不同。

22.3.1 SWJ 调试端口脚

该芯片 5 个普通 I/O 口可用作 SWJ-DP 接口引脚。这些引脚在所有的封装里都存在。

表 77. SWJ 调试端口管脚

SWJ-DP 端口引脚名称	JTAG 调试接口		SW 调试接口		引脚分配
	类型	描述	类型	调试功能	
JTMS/SWDIO	输入	JTAG 模式选择	输入/输出	串行数据输入/输出	PA13
JTCK/SWCLK	输入	JTAG 时钟	输入	串行时钟	PA14
JTDI	输入	JTAG 数据输入	-	-	PA15
JTDO/TRACESWO	输出	JTAG 数据输出	-	跟踪时为 TRACESWO 信号	PB3
JNTRST	输入	JTAG 模块复位	-	-	PB4

22.3.2 灵活的 SWJ-DP 脚分配

复位（SYSRESETn 或 PORESETn）以后，属于 SWJ-DP 的所有 5 个引脚都立即被初始化为可被调试器使用的专用引脚（注意，并没有初始化跟踪输出脚，除非调试器对此脚进行定义）。

然而，该系列微控制器可以用 AFIO_MAPR 寄存器来禁止 SWJ-DP 接口的部分或所有引脚的功能，这些专用引脚将被释放以用作普通 I/O 口。此寄存器被映射到和 CPU 系统总线相连接的 APB 桥上。对此寄存器的设置将由用户代码而不是调试器完成。

3 个控制位用来配置 SWJ-DP 接口的引脚，这 3 个位在系统复位时复位。

- AFIO_MAPR（微控制器中的地址是 0x40010004）
- 读：APB，无等待状态
- 写：APB，如果 AHB-APB 桥的写缓冲器满了，则一个等待状态

位 26：24=SWJ_CFG[2：0] 由软件置位和复位，这 3 位用来设置分配给 SWJ 调试接口的专用引脚数目，目的是在使用不同的调试接口时能释放尽可能多的引脚用作普通 I/O 口。

复位后的初始值是 000（所有引脚都设置为 JTAG-DP 接口专用引脚），同时只能置位 3 个

位中的一个（禁止同时设置一个以上的位）。

表 78. 灵活的 SWJ_DP 管脚分配

SWJ-CFG[2:0]	配置为调试专用的引脚	SWJ 接口的 I/O 口分配				
		PA13/JTMS /SWDIO	PA14/JTCK /SWCLK	PA15/JTDI	PB3/JTDO	PB4/JNTRST
000	所有的 SWJ 引脚（JTAG-DP + SW-DP）复位状态	专用	专用	专用	专用	专用
001	所有的 SWJ 引脚（JTAG-DP + SW-DP）除了 JNTRST 引脚	专用	专用	专用	专用	释放
002	JTAG-DP 接口禁止，SW-DP 接口允许	专用	专用	释放		
100	JTAG-DP 接口和 SW-DP 接口都禁止	释放				
其他	禁止					

当 APB 桥的写缓冲区满了时，在写 AFIO_MAPR 寄存器时需要多用一个 APB 周期。这是因为 JTAGSW 脚的释放需要 2 个 APB 周期，以保证输入内核的 nTRST 和 TCK 信号的平稳。

- 周期 1：输入 1 / 0 的 JTAGSW 信号到内核（nTRST，TDI 和 TMS 为 1，TCK 为 0）。
- 周期 2：GPI/O 控制器获得 SWJTAG I/O 引脚的控制信号（如对方向，上拉/下拉，施密特触发等的控制）。

22.3.3 JTAG 脚上的内部上拉和下拉

保证 JTAG 的输入引脚不是悬空的是非常必要的，因为他们直接连接到 D 触发器控制着调试模式。必须特别注意 SWCLK/TCK 引脚，因为他们直接连接到一些 D 触发器的时钟端。

为了避免任何未受控制的 I/O 电平，该芯片在 JTAG 输入脚上嵌入了内部上拉和下拉。

- JINTRST：内部上拉
- JTDI：内部上拉
- JTMS/SWDIO：内部上拉
- TCK/SWCLK：内部下拉

一旦 JTAG I/O 被用户代码释放，GPIO 控制器再次取得控制。这些 I/O 口的状态将恢复到复位时的状态。

- JINTRST：带上拉的输入
- JTDI：带上拉的输入
- JTMS/SWDIO：带上拉的输入
- TCK/SWCLK：带下拉的输入
- JTDO：浮动输入

软件可以把这些 I/O 口作为普通的 I/O 口使用。

JTAG IEEE 标准建议对 TDI，TMS 和 nTRST 上拉，而对 TCK 没有特别的建议。但在芯片中，JTCK 引脚带有下拉。内嵌的上拉和下拉使芯片不再需要外加外部电阻。

22.3.4 利用串行接口并释放不用的调试脚作为普通 I/O 口

为了利用串行调试接口来释放一些普通 I/O 口，用户软件必须在复位后设置 SWJ_CFG=010，从而释放 PA15，PB3 和 PB4 用做普通 I/O 口。

在调试时，调试器进行以下操作：

- 在系统复位时，所有 SWJ 引脚被分配为专用引脚（JTAG-DP + SW-DP）。
- 在系统复位状态下，调试器发送指定 JTAG 序列，从 JTAG-DP 切换到 SW-DP。
- 仍然在系统复位状态下，调试器在复位地址处设置断点。
- 释放复位信号，内核停止在复位地址处。
- 从这里开始，所有的调试通信将使用 SW-DP 接口，其他 JTAG 引脚可以由用户代码改配为普通 I/O 口。

对于用户软件设计，应注意：

在复位后，这些专用引脚仍然处于带上拉的输入（nTRST，TMS，TDI），带下拉的输入（TCK），和输出（TDO）状态，并持续一段时间，直到用户代码释放这些引脚

当这些引脚被配置成专用引脚时（JTAG 或者 SW 或者 TRACE），修改相应的普通 I/O 口配置寄存器是无效的。

22.4 JTAG TAP 连接

微控制器内部串联了两个 JTAG TAP。边界扫描 TAP 专门用来进行测试（IR 寄存器为 5 比特位宽）和 TAP（IR 寄存器为有 4 比特位宽）。

为了访问 TAP 对芯片进行调试，必须：

- 首先，必须将 BYPASS 指令移位输入 TMC TAP。
- 其次，在移位输入 IR 时，每个扫描链包含 9 个比特位（=5+4），对于不用的 TAP，必须输入 BYPASS 指令
- 移位输入数据时，不用的 TAP 处于 BYPASS 模式下，因此数据扫描链需要额外添加一位比特位。

重要：一旦使用了指定的 JTAG 序列选择了串行调试接口，TMC TAP 自动被禁止（JTMS 被强制为高）。

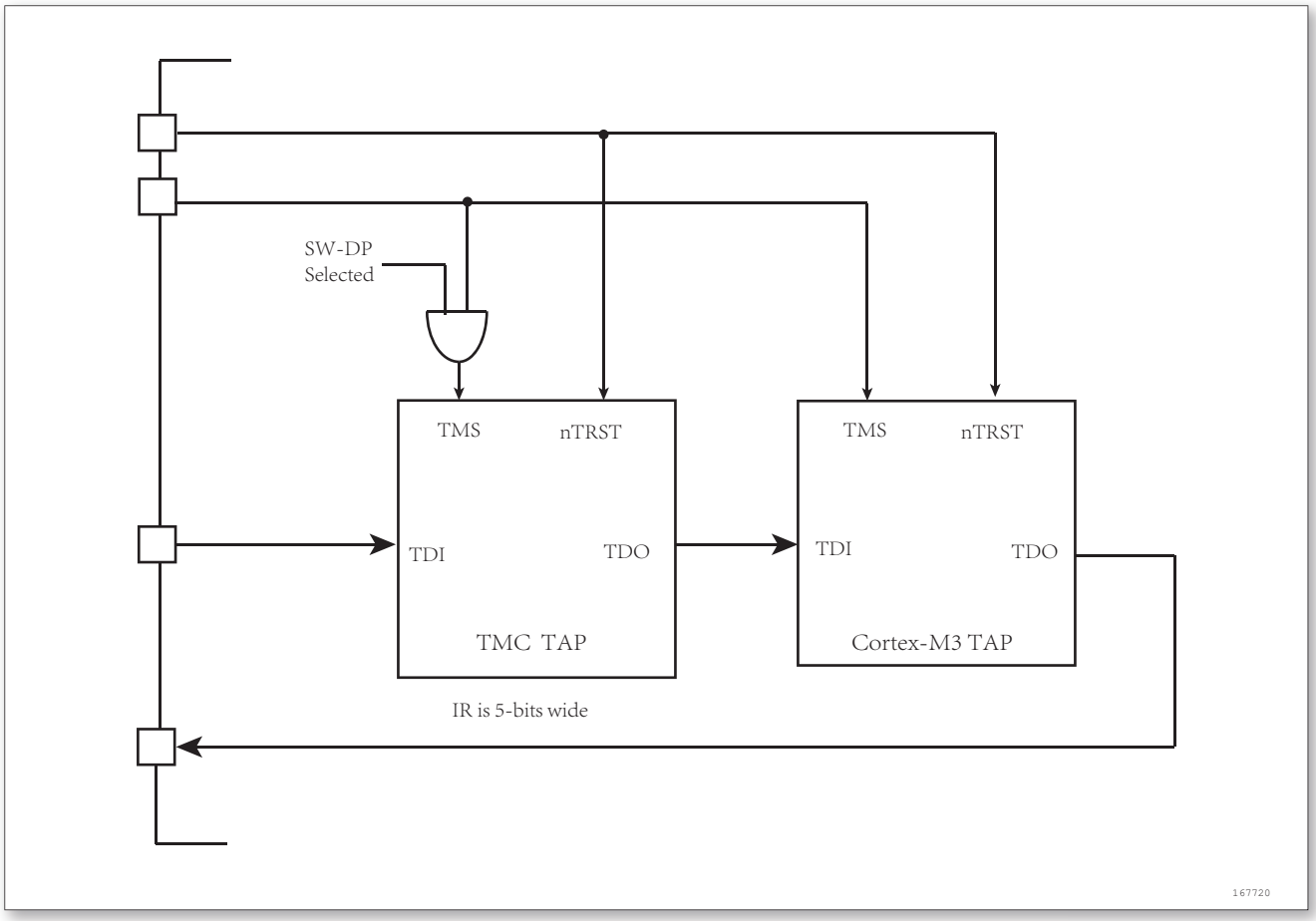


图 171. JTAG TAP 连接

22.5 ID 代码和锁定机制

在芯片内部有多个 ID 编码。

22.5.1 微控制器设备 ID 编码

微控制器内含一个 MCU ID 编码。这个 ID 定义了 MCU 的部件号和硅片版本。它是 DBG_MCU 的一个组成部分，并且映射到外部 PPB 总线上。使用 JTAG 调试口（4 ~ 5 个引脚）或 SW 调试口（2 个引脚）或通过用户代码都可以访问此编码。

DBGMCU_IDCODE

地址：0x40007080 只支持 32 位访问

只读 =0xXXXXXXXX，其中 X 为内容不确定的位

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

位 31: 0	DEV_ID：设备识别编码（Device identifier）

22.5.2 边界扫描 TAP JTAG ID 编码

芯片的边界扫描 TAP 集成了 JTAG ID 编码。

22.5.3 CPU JTAG TAP

CPU JTAG TAP 有一个 JTAG ID 编码。这个 ID 编码是 CPU 默认的，且没有被修改过，只能通过 JTAG 调试口访问。

22.5.4 Cortex JEDEC-106 ID 代码

CPU 有一个 JEDEC-106 ID 编码。它位于映射到内部 PPB 总线地址为 0xE00F F000_0xE00F FFFF 的 4KB ROM 表中。

下表是该系列的各个 ID 编码：

表 80. 芯片 ID 编码

ID 名	芯片
DEV_ID	0xCC88 xxx3
CPU TAP JTAG ID	0x4BA0 0477
CPU TAP SW ID	0x2BA0 1477

22.6 JTAG 调试端口

标准的 JTAG 状态机是通过一个 4 比特位的指令寄存器（IR）和 5 个数据寄存器实现的。

表 81. JTAG 调试端口数据寄存器

IR(3:0)	数据寄存器	描述
1111	BYPASS[1 比特位]	
1110	IDCODE[32 比特位]	ID 编码寄存器 0x4BA0 0477

IR(3:0)	数据寄存器	描述
1010	DPACC[35 比特位]	调试接口寄存器 初始化调试端口，并允许访问调试接口寄存器 输入数据时： Bits34：3 = DATA[31：0]：对应写操作的 32 位数据位（32-bit data to transfer for a write request） Bits2：1 = A[3：2]：调试接口寄存器的 2 位地址值（2-bit address of a debug port register） Bit0 = RnW：读操作（1）或写操作（0） 输出数据时： Bits34：3 = DATA[31：0]：前一次读操作的 32 位数据结果（32-bit data which is read following a read request） Bits2：0 = ACK[2：0]：3 比特位的应答（3-bit Acknowledge） 010 = 成功/失败 001 = 等待其他 = 未定义
1011	APACC[35 比特位]	存取接口寄存器] 初始化存取接口并允许访问存取接口寄存器 输入数据时： Bits34：3 = DATA[31：0]：对应写操作的 32 位数据位（32-bit data to shift in for a write request） Bits2：1 = A[3：2]：2 比特位地址（2-bit address（sub-address AP registers）（AP 寄存器的部分地址） Bit0 = RnW：读操作（1）或写操作（0）（Read request（1）or write request（0）） 输出数据时： Bits34：3 = DATA[31：0]：前一次读操作的 32 位数据结果（32-bit data which is read following a read request） Bits2：0 = ACK[2：0]：3 比特位的应答（3-bit Acknowledge） 010 = 成功/失败 001 = 等待其他 = 未定义 关于 AP 寄存器请参考 AHB-AP 章节，这些寄存器的地址由以下部分组成：A[3：2] 移位值 A[3：2]。 DP SELECT 寄存器的当前值。
1000	ABORT [35 比特位]	中止寄存器 Bits 31：1 未定义 Bit 0 = DAPABORT：写 1 产生一个 DAP 中止（write 1 to generate a DAP abort）

表 82. A[3：2] 定义的 32 位调试接口寄存器地址

IR(3:0)	数据寄存器	描述
地址	A(3:2) 值	描述
0x0	00	未定义

IR(3:0)	数据寄存器	描述
0x4	01	DP CTRL/STAT 寄存器 请求一个系统或调试的上电操作 配置 AP 访问的操作模式 控制比较，校验操作 读取一些状态位（溢出，上电响应）
0x8	10	DP SELECT 寄存器：用来选择当前的访问端口和有效的 4 字长寄存器窗口 Bits31：24：APSEL 选择当前 AP（select the current AP） Bits23：8：未定义 Bits7：4：APBANKSEL：在当前 AP 上选择 4 字长寄存器窗口（select the active 4-words register window on the current AP） Bits3：0：未定义
0xC	11	DP RDBUFF 寄存器：用来使调试器获得前一次操作的最终结果（不用再请求一个新的 JTAG-DP 操作）

22.7 SW 调试端口

22.7.1 SW 协议介绍

此同步串行协议使用 2 个引脚：

- SWCLK：从主机到目标的时钟信号
- SWDIO：双向数据信号

协议允许读写 2 个寄存器组（DPACC 和 APACC 寄存器组）。

数据位按 LSB 传输。

由于 SWDIO 为双向口，该引脚需有上拉（建议使用 100K 电阻）。

按协议每次 SWDIO 方向改变时，需插入一个转换时间。在该期间内主机和目标都不驱动此信号线。转换时间的默认值是 1 个比特，但可以通过配置 SWCLK 频率来调节。

22.7.2 SW 协议序列

每个序列由 3 个阶段组成：

- 主机发送包请求（8 位）
- 目标发送确认相应（3 位）
- 主机或目标发送数据（33 位）

表 83. 请求包（8 比特位）

比特位	名称	描述
0	起始	必须为 1
1	APnDP	0：访问 DP 1：访问 AP
2	RnW	0：写请求 1：读请求
4:3	A（3：2）	DP 或 AP 寄存器的地址

比特位	名称	描述
5	Parity	前面比特位的校验位
6	Stop	0
7	Park	不能由主机驱动，由于有上拉，目标永远读为 1

有关 DPACC 和 APACC 寄存器描述的详细资料，请参考 CPU 技术参考手册。

包请求后总是跟一个（缺省为 1 位）转换时间，此时主机和目标都不驱动线路。

表 84. 请求包（3 比特位）

比特位	名称	描述
0..2	ACK	001：失败 010：等待 100：成功

当 ACK 为失败或等待，或者是一个回复读操作的 ACK，此 ACK 后有一个转换时间。

表 85. 请求包（33 比特位）

比特位	名称	描述
0..31	WDATA/RDATA	写或读的数据
32	Parity	32 位数据的奇偶校验位

读操作的数据传输操作后有一个转换时间。

22.7.3 SW-DP 状态机（Reset，idle states，ID code）

SW-DP 状态机有一个内部 ID 编码用来识别 SW-DP，它遵守 JEP-106 标准。

在调试器读这个 ID 编码之前，SW-DP 的状态机是不工作的。

- SW-DP 状态机将处于 RESET 状态，在上电复位后，或 DP 从 JTAG 切换到 SWD 后，或有超过 50 个周期的高电平。
- 当状态机处于 RESET 状态时，如果有至少 2 个周期的低电平，状态机将切换到 IDLE 状态。
- 当状态机处于 RESET 状态后，必须首先进入 IDLE 状态，并执行一个读 DP-SW ID 寄存器的操作。否则，调试器在执行其他传输时，只能获得一个失败的 ACK 响应。

22.7.4 DP 和 AP 读/写访问

- 对 DP 的读操作没有传递性：调试器将直接获得数据（如果 ACK = 成功），或者等待（如果 ACK= 等待）。
- 对 AP 的读操作具有传递性。这意味着前一次读操作的结果只能在下一次操作时获得。如果下一次的操作不是对 AP 的访问，则必须读 DP-RDBUFF 寄存器来获得上一次读操作的结果。
- DP-CTRL/STAT 寄存器的 READOK 标志位会在每次 AP 读操作和 RDBUFF 读操作后更新，以通知调试器 AP 的读操作是否成功。

- SW-DP 具有写缓冲区（DP 和 AP 都有写缓冲），这使得其他传输在进行时，仍然可以接受写操作。如果写缓冲区满，调试器将获得一个等待的 ACK 响应。读 IDCODE 寄存器，读 CTRL/STAT 寄存器和写 ABORT 寄存器操作在写缓冲区满时仍被接受。
- 由于 SWCLK 和 HCLK 的异步性，需要在写操作后（在奇偶校验位后）插入 2 个额外的 SWCLK 周期，以确保内部写操作正确完成。这两个额外的时钟周期需要在线路为低时插入（IDLE 状态下）。这个操作步骤在写 CTRL/STAT 寄存器以提出一个上电请求时尤其重要，否则下一个操作（在内核上电后才有效的操作）会立即执行，这将导致失败。

22.7.5 SW-DP 寄存器

当 APnDP=0 时，可以访问以下这些寄存器。

A（3：2）	读/写	SELECT 寄存器的 CTRLSEL 位	寄存器	描述
00	读		IDCODE	固定为 0x1BA0 1477（用于识别 SW-DP）。
00	写		ABORT	
01	读/写	0	DP-CTRL/STAT	请求一个系统或调试的上电操作； 配置 AP 访问的操作模式； 控制比较，校验操作； 读取一些状态位（溢出，上电响应）。
01	读/写	1	WIRE CONTROL	配置串行通信物理层协议（如转换时间长度等）。
10	读		READ RESEND	允许从一个错误的调试传输中恢复数据而不用重复最初的 AP 传输。
10	写		SELECT	选择当前的访问端口和有效的 4 字长寄存器窗口。
11	读/写		READ BUFFER	由于 AP 的访问具有传递性（当前 AP 读操作的结果会在下次 AP 传输时传出），因此这个寄存器非常必要。这个寄存器会从 AP 捕获上一次读操作的数据结果，因此可以获得数据而不必再启动一个新的 AP 传输。

22.7.6 SW-AP 寄存器

当 APnDP=1 时，可以访问

AP 寄存器的访问地址由以下两部分组成：

- A[3：2] 的值
- DP SELECT 寄存器的当前值

22.8 MCU 调试模块（MCUDBG）

MCU 调试模块协助调试器提供以下功能：

- 低功耗模式
- 在断点时提供定时器，看门狗的时钟控制
- 对跟踪脚分配的控制

22.8.1 低功耗模式的调试支持

使用 WFI 和 WFE 可以进入低功耗模式。MCU 支持多种低功耗模式，分别可以关闭 CPU 时钟，或降低 CPU 的能耗。内核不允许在调试期间关闭 FCLK 或 HCLK。这些时钟对于调试操作是必要的，因此在调试期间，它们必须工作。MCU 使用一种特殊的方式，允许用户在低功耗模式下调试代码。

为实现这一功能，调试器必须先设置一些配置寄存器来改变低功耗模式的特性。

- 在睡眠模式下，调试器必须先置位 DBGMCU_CR 寄存器的 DBG_SLEEP 位。这将为 HCLK 提供与 FCLK（由代码配置的系统时钟）相同的时钟。
- 停机模式下，调试器必须先置位 DBG_STOP 位。这将激活内部振荡器，在停机模式下为 FCLK 和 HCLK 提供时钟。

22.8.2 支持定时器、看门狗

在产生断点时，有必要根据定时器和看门狗的不同用途选择计数器的工作模式：

- 在产生断点时，计数器继续计数。这在输出 PWM 控制电机时常常要用到。
- 在产生断点时，计数器停止计数。这对于看门狗的计数器是必需的。

22.8.3 调试 MCU 配置寄存器

此寄存器允许在调试状态下配置 MCU。包括：

- 支持低功耗模式
- 支持定时器和看门狗的计数器
- 分配跟踪引脚

DBGMCU_CR 寄存器被映射到外部 APB 总线，基地址为 0x4000 7084。寄存器由 POR-SET 异步复位（不被系统复位所复位）。当内核处于复位状态下时，调试器可写该寄存器。

DBGMCU_CR 寄存器被映射到外部 APB 总线，基地址为 0x4001 3404。寄存器由 POR-SET 异步复位（不被系统复位所复位）。当内核处于复位状态下时，调试器可写该寄存器。

如果调试器不支持这些特性，用户软件仍可写这些寄存器。

22.9 DBG 寄存器描述

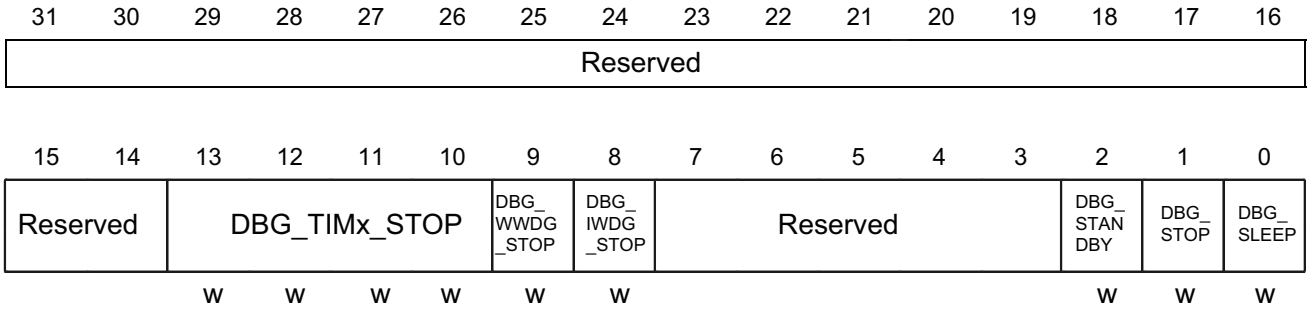
表 87. DBG 寄存器概览

Offset	Acronym	Register Name	Reset	Section
0x00	DBG	DBG 控制寄存器	0x00000000	小节 22.9.1

22.9.1 DBG 控制寄存器 (DBG_CR)

地址：0x40007084 只支持 32 位访问

POR 复位：0x0000 0000（不被系统复位所复位）



Bit	Field	Type	Reset	Description
31:14	Reserved			始终读为 0。
13:10	DBG_TIMx_STOP	w	0x00	当内核进入调试状态时计数器停止工作 x = 4..1（TIMx counter stopped when core is halted） 0：选中定时器的计数器仍然正常工作 1：选中定时器的计数器停止工作
9	DBG_WWDG_STOP	w	0x00	当内核进入调试状态时调试窗口看门狗停止工作（Debug window watchdog stopped when core is halted） 0：窗口看门狗计数器仍然正常工作 1：窗口看门狗计数器停止工作
8	DBG_IWDG_STOP	w	0x00	当内核进入调试状态时看门狗停止工作（Debug independent watchdog stopped when core is halted） 0：看门狗计数器仍然正常工作 1：看门狗计数器停止工作
7:3	Reserved			始终读为 0。
2	DBG_STANDBY	w	0x00	调试待机模式（Debug Standby mode） 0：（FCLK 关，HCLK 关）整个数字电路部分都断电。从软件的观点看，退出 STANDBY 模式与复位是一样的（除了一些状态位指示了微控制器刚从 STANDBY 状态退出）par 1：（FCLK 开，HCLK 开）数字电路部分不下电，FCLK 和 HCLK 时钟由内部 RL 振荡器提供时钟。另外，微控制器通过产生系统复位来退出 STANDBY 模式和复位是一样的。
1	DBG_STOP	w	0x00	调试停机模式（Debug Stop mode） 0：（FCLK 关，HCLK 关）在停机模式时，时钟控制器禁止一切时钟（包括 HCLK 和 FCLK）。当从 STOP 模式退出时，时钟的配置和复位之后的配置一样（微控制器由 8MHz 的内部振荡器（HSI）提供时钟）。因此，软件必须重新配置时钟控制系统启动 PLL，晶振等。 1：（FCLK 开，HCLK 开）在停机模式时，FCLK 和 HCLK 时钟由内部振荡器提供。当退出停机模式时，软件必须重新配置时钟系统启动 PLL，晶振等（与配置此比特位为 0 时的操作一样）。

Bit	Field	Type	Reset	Description
0	DBG_SLEEP	w	0x00	调试睡眠模式（Debug Sleep mode） 0：（FCLK 开，HCLK 关）在睡眠模式时，FCLK 由原先已配置好的系统时钟提供，HCLK 则关闭。由于睡眠模式不会复位已配置好的时钟系统，因此从睡眠模式退出时，软件不需要重新配置时钟系统。 1：（FCLK 开，HCLK 开）在睡眠模式时，FCLK 和 HCLK 时钟都由原先配置好的系统时钟提供。

23 修改记录

修改记录

表 88. 修改记录

日期	版本	内容
2020/02/24	Rev1.69	1. UART 传输部分描述
2020/02/13	Rev1.68	修改时钟树 MCO 时钟源
2019/09/03	Rev1.67	修改 DMA 通道描述
2019/08/06	Rev1.66	修改存储器和总线架构处笔误
2019/07/23	Rev1.65	SPI_EXTCTL 仅当 DW8_32 位为‘0’时有效
2019/07/16	Rev1.64	UART BRR 寄存器最小值为 4
2019/05/09	Rev1.63	修改端口模式配置 高级定时器中的刹车和死区寄存器 BDTR 的死区发生器设置位 DTG 纠正
2019/04/16	Rev1.62	修改 adc 计算公式
2019/01/10	Rev1.61	修改 PWR 描述
2018/12/22	Rev1.61	TIMX_16bit 功能描述部分输入捕获小节中将“TIMx_CCR1 寄存器中的 CC1S = 01”修改为“TIMx_CCMR1 寄存器中的 CC1S = 01”。 TIM1/8 将文中的“捕捉”都替换为“捕获”。 TIM1/8 功能描述部分输入捕获小节中将“TIMx_CCR1 寄存器中的 CC1S = 01”修改为“TIMx_CCMR1 寄存器中的 CC1S=01”。 TIM1/8 TIMx 定时器和外部触发的同步部分将错误的书写“IMx_CR1”修改为“TIMx_CR1”。 UART_BRR 中 DIV_Mantiss 不能为 0。
2018/12/20	Rev1.60	UART_CSR 修改为 UART 当前状态寄存器