PY32F002A 系列参考手册

32 位 ARM® Cortex®-M0+ 微控制器

Puya Semiconductor (Shanghai) Co., Ltd.

目录

1. 寄存器描述中使用的缩写列表

缩写	描述
Read/Write (RW)	软件能读写此位
Read-only (R)	软件只能读此位
Write-only (W)	软件只能写此位，读此位将返回复位值
Read/Clear Write0 (RC_W0)	软件可以读此位，也可以通过写 0 清除此位，写 1 对此位无影响
Read/Clear Write1 (RC_W1)	软件可以读此位，也可以通过写 1 清除此位，写 0 对此位无影响
Read/Clear Write (RC_W)	软件可以通过写入寄存器来读取和清除该位，写入该位的值并不重要
Read/Clear by read (RC_R)	软件可以读取这个位。读取此位会自动将其清除为 0，写入此位不会影响位值
Read/Set by Read (RS_R)	软件可以读取这个位。读取此位会自动将其设置为 1，写入此位不会影响位值
Read/Set (RS)	软件可以读此位，也可以设置此位为 1，写 0 对此位无影响
Toggle (T)	软件可以通过写入 1 来切换此位，写入 0 无效
保留 (Res)	保留位，必须保持在重置值

2. 系统架构框图

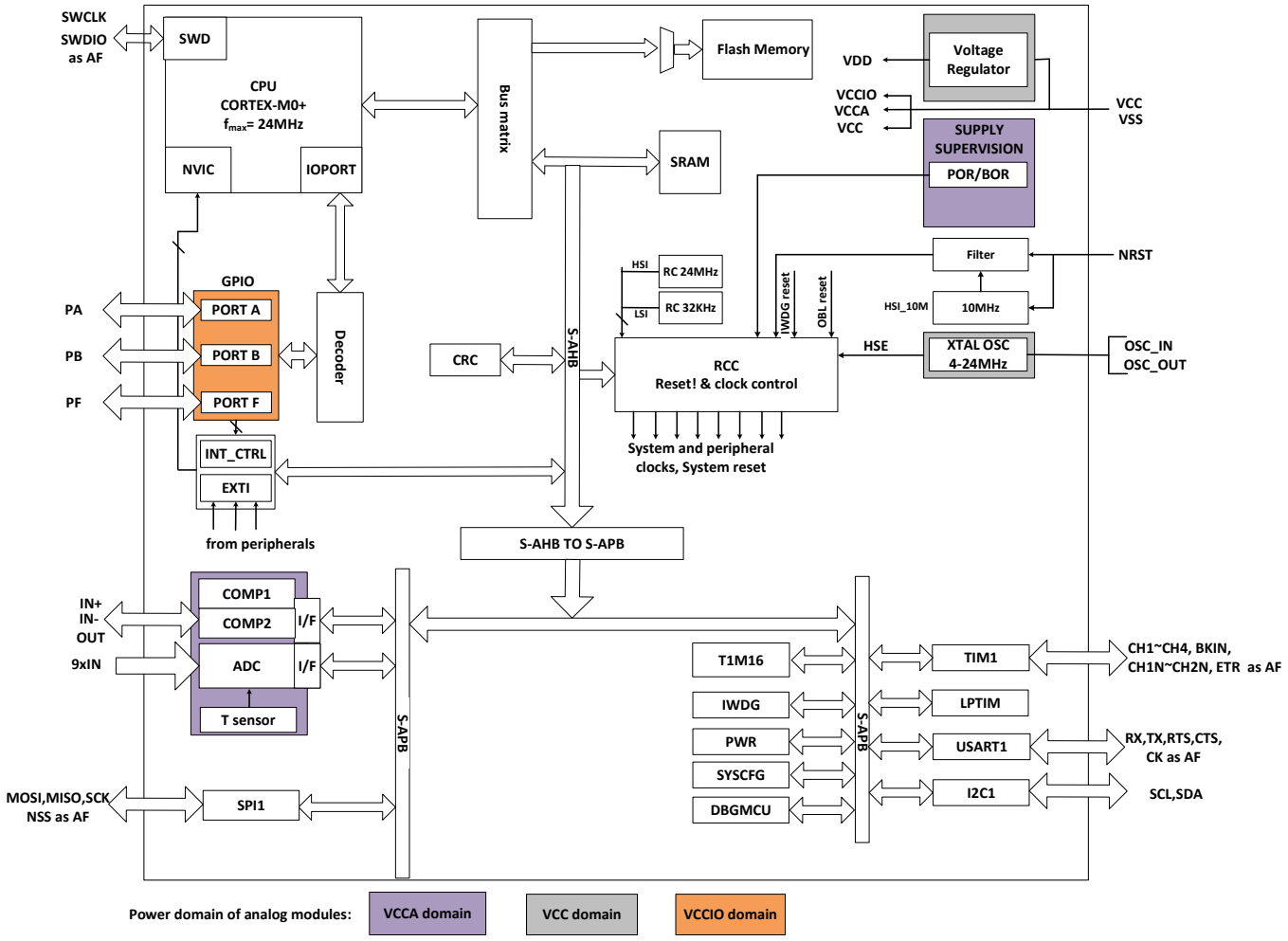


图 2-1 系统架构框图

3. 存储器和总线架构

3.1. 系统架构

系统由以下部分组成：

- 一个 Master
  - Cortex-M0+
- 三个 Slave
  - 内部 SRAM
  - 内部 Flash
  - 带 AHB-APB Bridge 的 AHB

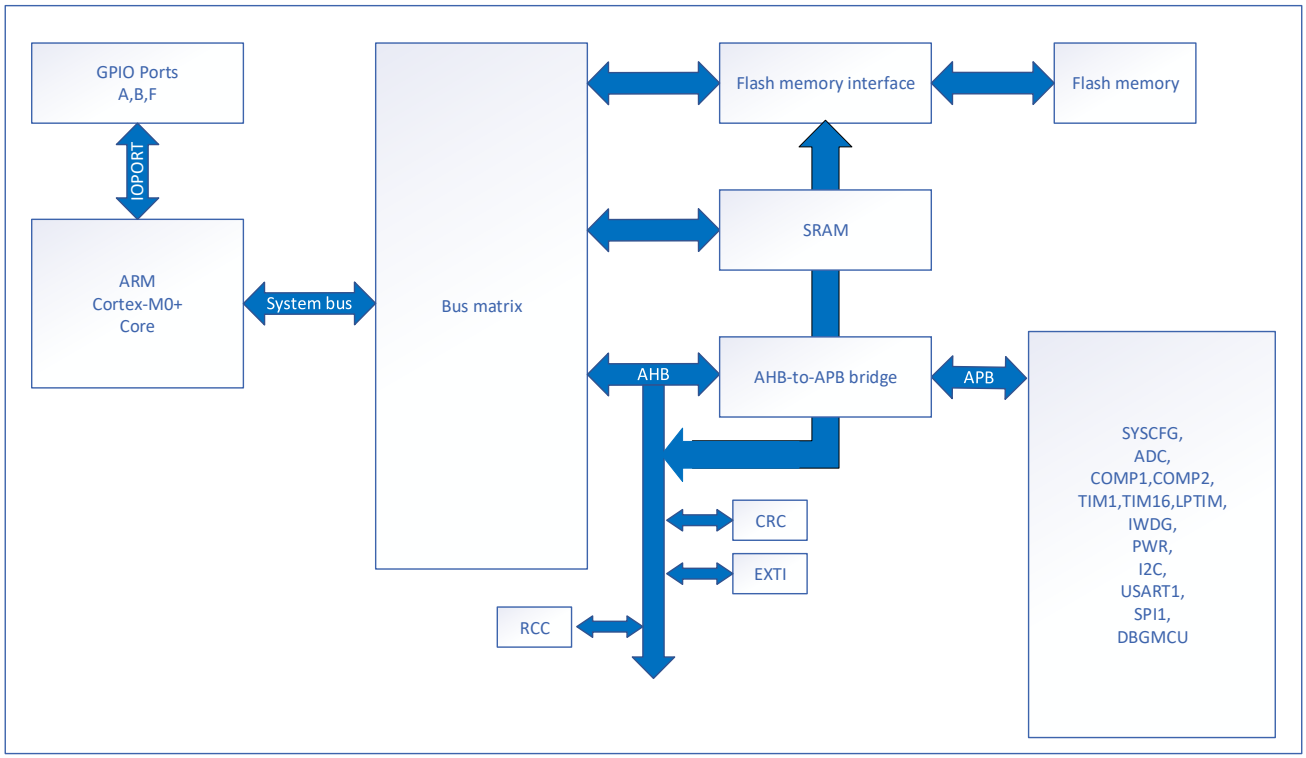


图 3-1 系统架构

- 系统总线

该总线把 Cortex-M0+的系统总线连接到 bus matrix，后者用来管理 CPU 的仲裁。

- 总线 Matrix

总线 Matrix 管理在 CPU 总线的仲裁。该仲裁使用 Round Robin 算法。总线 Matrix 由 Master（CPU）和 slaves（Flash memory、SRAM 和 AHB-to-APB bridge）。

- AHB-to-APB bridge（APB）

The AHB-to-APB bridge提供了在AHB和APB总线之间的同步连接到该Bridge的外设地址映射。

3.2. 存储器结构

3.3. 存储器结构简介

程序存储器、数据存储器、寄存器和 IO 端口被统一编址在一个线性 4-Gbytes 空间。该地址以小端编码形式存在（一个 word 中，最低字节分配在最低地址）。

整个寻址空间被划分成 8 个 512Mbyte 的 Block 区域。

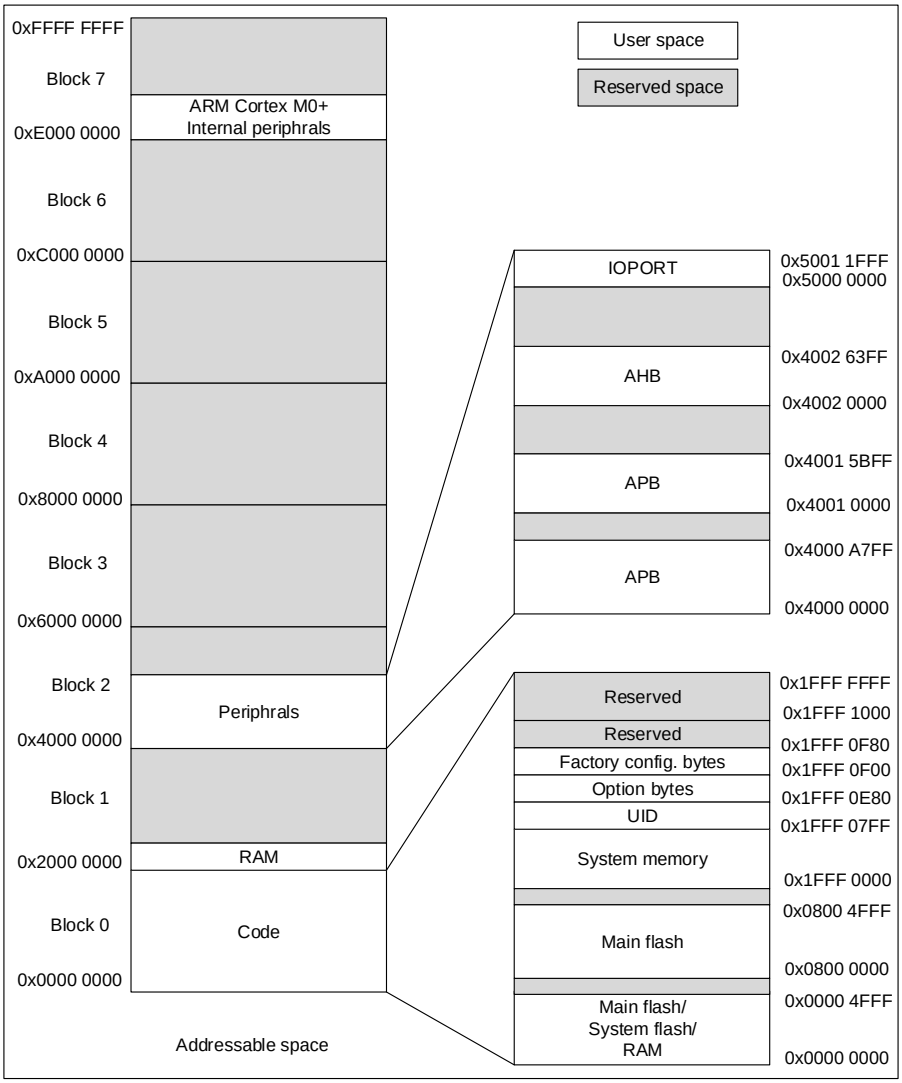


图 3-2 存储器映射

表 3-1 存储器地址

Type	Boundary Address	Size	Memory Area	Description
SRAM	0x2000 0C00-0x3FFF FFFF	512MBytes	Reserved	
	0x2000 0000-0x2000 0BFF	3KB	SRAM	根据硬件不同，SRAM 最大为 3KB
Code	0x1FFF 1000-0x1FFF FFFF	4KB	Reserved	
	0x1FFF 0F80-0x1FFF 0FFF	128Bytes	Reserved	
	0x1FFF 0F00-0x1FFF 0F7F	128Bytes	Factory config	存放 HSI triming 数据、flash 擦写时间配置参数

Type	Boundary Address	Size	Memory Area	Description
	0x1FFF 0E80-0x1FFF 0EFF	128Bytes	Option bytes	option bytes
	0x1FFF 0E00-0x1FFF 0E7F	128Bytes	UID	Unique ID
	0x1FFF 0000-0x1FFF 07FF	2KB	System memory	存放 boot loader
	0x0800 5000-0x1FFE FFFF	384MBytes	Reserved	
	0x0800 0000-0x0800 4FFF	20KB	Main flash memory	
	0x0000 5000-0x07FF FFFF	8MBytes	Reserved	
	0x0000 0000-0x0000 4FFF	20KB	根据 Boot 配置选择： 1）Main flash memory 2）System memory 3）SRAM	

Note:

上述空间除 0x1FFF 0E00-0x1FFF 0E7F 外，其余标注为 reserved 的空间，无法进行写操作，读为 0，且产生 response error。

表 3-2 外设寄存器地址

Bus	Boundary Address	Size	Peripheral
	0xE000 0000-0xE00F FFFF	1Mbytes	M0+
IOPORT	0x5000 1800-0x5FFF FFFF	256MBytes	Reserved(1)
	0x5000 1400-0x5000 17FF	1KB	GPIOF
	0x5000 1000-0x5000 13FF	1KB	Reserved
	0x5000 0C00-0x5000 0FFF	1KB	Reserved
	0x5000 0800-0x5000 0BFF	1KB	Reserved
	0x5000 0400-0x5000 07FF	1KB	GPIOB
	0x5000 0000-0x5000 03FF	1KB	GPIOA
AHB	0x4002 3400-0x4FFF FFFF		Reserved
	0x4002 300C-0x4002 33FF	1KB	Reserved
	0x4002 3000-0x4002 3008		CRC
	0x4002 2400-0x4002 2FFF		Reserved
	0x4002 2124-0x4002 23FF	1KB	Reserved
	0x4002 2000-0x4002 2120		Flash
	0x4002 1C00-0x4002 1FFF	3KB	Reserved
	0x4002 1888-0x4002 1BFF	1KB	Reserved
	0x4002 1800-0x4002 1884		EXTI（2）
	0x4002 1400-0x4002 17FF	1KB	Reserved
	0x4002 1064-0x4002 13FF	1KB	Reserved
	0x4002 1000-0x4002 1060		RCC（2）
	0x4002 0C00-0x4002 0FFF	1KB	Reserved
	0x4002 0040-0x4002 03FF	1KB	Reserved
	0x4002 0000-0x4002 003C		Reserved
APB	0x4001 5C00-0x4001 FFFF	32KB	Reserved
	0x4001 5880-0x4001 5BFF	1KB	Reserved
	0x4001 5800-0x4001 587F		DBG
	0x4001 4C00-0x4001 57FF	3KB	Reserved
	0x4001 4850-0x4001 4BFF	1KB	Reserved
	0x4001 4800-0x4001 484C		Reserved
	0x4001 4450-0x4001 47FF	1KB	Reserved
	0x4001 4400-0x4001 404C		TIM16

Bus	Boundary Address	Size	Peripheral
	0x4001 3C00-0x4001 43FF	2KB	Reserved
	0x4001 381C-0x4001 3BFF	1KB	Reserved
	0x4001 3800-0x4001 3018		USART1
	0x4001 3400-0x4001 37FF	1KB	Reserved
	0x4001 3010-0x4001 33FF	1KB	Reserved
	0x4001 3000-0x4001 300C		SPI1
	0x4001 2C50-0x4001 2FFF	1KB	Reserved
	0x4001 2C00-0x4001 2C4C		TIM1
	0x4001 2800-0x4001 2BFF	1KB	Reserved
	0x4001 270C-0x4001 27FF	1KB	Reserved
	0x4001 2400-0x4001 2708		ADC
	0x4001 0400-0x4001 23FF	8KB	Reserved
	0x4001 0220-0x4001 03FF	1KB	Reserved
	0x4001 0200-0x4001 021F		COMP1 and COMP2
	0x4001 0000-0x4001 01FF		SYSCFG
	0x4000 B400-0x4000 FFFF	19KB	Reserved
	0x4000 B000-0x4000 B3FF	1KB	Reserved
	0x4000 8400-0x4000 AFFF	11KB	Reserved
	0x4000 8000-0x4000 83FF	1KB	Reserved
	0x4000 7C28-0x4000 7FFF	1KB	Reserved
	0x4000 7C00-0x4000 7C24		LPTIM
	0x4000 7400-0x4000 7BFF	2KB	Reserved
	0x4000 7018-0x4000 73FF	1KB	Reserved
	0x4000 7000-0x4000 7014		PWR (3)
	0x4000 5800-0x4000 6FFF	6KB	Reserved
	0x4000 5434-0x4000 57FF	1KB	Reserved
	0x4000 5400-0x4000 5430		I2C
	0x4000 4800-0x4000 53FF	3KB	Reserved
	0x4000 441C-0x4000 47FF	1KB	Reserved
	0x4000 4400-0x4000 4418		Reserved
	0x4000 3C00-0x4000 43FF	1KB	Reserved
	0x4000 3800-0x4000 3BFF	1KB	Reserved
	0x4000 3400-0x4000 37FF	1KB	Reserved
	0x4000 3014-0x4000 33FF	1KB	Reserved
	0x4000 3000-0x4000 0010		IWDG
	0x4000 2C0C-0x4000 2FFF	1KB	Reserved
	0x4000 2C00-0x4000 2C08		Reserved
	0x4000 2830-0x4000 2BFF	1KB	Reserved
	0x4000 2800-0x4000 282C		Reserved
	0x4000 2400-0x4000 27FF	1KB	Reserved
	0x4000 2054-0x4000 23FF	1KB	Reserved
	0x4000 2000-0x4000 0050		Reserved
	0x4000 1800-0x4000 1FFF	2KB	Reserved
	0x4000 1400-0x4000 17FF	1KB	Reserved
	0x4000 1000-0x4000 13FF	1KB	Reserved
	0x4000 0800-0x4000 0FFF	2KB	Reserved
	0x4000 0450-0x4000 07FF	1KB	Reserved
	0x4000 0400-0x4000 044C		Reserved
	0x4000 0000-0x4000 03FF	1KB	Reserved

Note:

（1） 上表 AHB 标注为 Reserved 的地址空间，无法写操作，读回为 0，且产生 HardFault；APB 标注为 Reserved 的地址空间，无法写操作，读回为 0，不会产生 HardFault。

（2） 不仅支持 32bit word 访问，还支持 halfword 和 byte 访问。

（3） 不仅支持 32bit word 访问，还支持 halfword 访问。

3.4. 嵌入式 SRAM

片内最大集成 3KB SRAM。通过 bytes、half-word（16bit）或者 word（32bit）的方式可访问 SRAM。软件对设定范围外空间的读写操作，会产生 hard fault。

3.5. Flash 存储器

Flash 存储器有两个不同的物理区域组成：

- Main flash 区域，20KB，它包含应用程序和用户数据。
- Information 区域，2.7KB，它包括以下部分：
  - Factory config. bytes：128Bytes，用于存放：
    存放 trimming 数据（含 HSI triming 数据）、上电读校验码等。
  - UID：128Bytes，用于存放芯片的 UID
  - Option bytes：128Bytes，用于存放芯片硬件和存储保护的配置值
  - System memory（系统存储器）：2KB，用于存放 Boot loader

Flash 接口实现基于 AHB 协议的指令读取和数据访问，它也通过寄存器实现了 flash 的基本 program/erase 等操作。

3.6. Boot 模式

通过 BOOT0 pin 和 boot 配置位 nBOOT1（存放于 Option bytes 中），可选择三种不同的启动模式，如下表所示：

表 3-3 Boot 配置

Boot mode configuration		Mode
nBOOT1 bit	BOOT0 pin	
X	0	选择 Main flash 作为启动区
1	1	选择 System memory 作为启动区
0	1	选择 SRAM 作为启动区

在该 startup 延迟后，CPU 从地址 0x0000 0000 取堆栈顶的值，然后从启动存储器的 0x0000 0004 地址开始执行指令。取决于被选择的启动模式，Main flash、system 存储器或者 SRAM 按照如下进行访问：

- Boot from main flash：main flash 跟启动存储器空间的 0x0000 0000 对齐，但是仍然可以按其本来的存储器空间（0x0800 0000）进行访问。也就是说，Flash 空间可以从地址 0x0000 0000 或者 0x0800 0000 访问到。
- Boot from system memory：system memory 对齐在启动存储器空间 0x0000 0000，但是仍然可以从它本来的地址空间 0x1FFF 0000 访问到。
- Boot from SRAM：SRAM 对齐在启动存储器空间的 0x0000 0000，但是仍然可以通过 0x2000 0000 地址访问到。

3.6.1. 存储器物理映像

如果 boot mode 被选择，应用软件可以修改在程序空间可被访问的存储器。这个修改通过 SYS-CFG_CFGR1 寄存器的 MEM_MODE 位选择决定（详见 SYSCFG 章节）。

3.6.2. 内嵌的自举程序

Boot loader 在芯片生产阶段被写入，并存放在 system memory 中。它用来使用下面串行接口进行对 flash 存储器的再次写入：

- USART，对应 PA2/PA3

4. 嵌入式闪存

4.1. 闪存主要特性

- Main flash block: 最大 20KB(5k x 32bit)
- Information block: 2.7KB(0.675k x 32bit)
- Page size: 128Bytes
- Sector size: 4KB

闪存控制接口电路的主要特征如下：

- 闪存写和擦除
- 读保护
- 写保护

4.2. 闪存功能介绍

4.2.1. 闪存结构

Flash 存储器由 32bit 宽的存储单元组成，可以用作程序和数据的存储，Page 大小为 128 Bytes，Sector 大小为 4 KB。

从功能上，Flash 存储器分为 Main flash 和 information flash，前者容量最大是 20 KB，后者容量为 2.7 KB。

Page erase 操作可以应用于 Main flash。

如果没有写保护设置，则 Mass erase 可应用于 Main flash，否则不能应用于 Main flash。

表 4-1 闪存结构及边界地址

Block	扇区 sector	页 Page	Base address	Size
Main flash	Sector 0	Page 0-31	0x0800 0000-0x0800 0FFF	4KB
	Sector 1	Page 32-63	0x0800 1000-0x0800 1FFF	4KB
	Sector 2	Page 64-95	0x0800 2000-0x0800 2FFF	4KB
	Sector 3	Page 96-127	0x0800 3000-0x0800 3FFF	4KB
	Sector4	Page 128-159	0x0800 4000-0x0800 4FFF	4KB
System flash	Sector 16	Page 0-27	0x1FFF 0000-0x1FFF 0DFF	3.5KB
UID		Page 28	0x1FFF 0E00-0x1FFF 0E7F	128bytes
Option bytes		Page 29	0x1FFF 0E80-0x1FFF 0EFF	128bytes
Factory config		Page 30	0x1FFF 0F00-0x1FFF 0F7F	128bytes
Reserved		Page 31	0x1FFF 0F80-0x1FFF 0FFF	128bytes

4.2.2. 闪存读操作和访问延迟

Flash 可以被作为一个通用的存储器空间，被直接寻址访问。通过专门的读控制时序，可以对 flash 存储器的内容进行读取。

取址和数据访问都是通过 AHB 总线进行的。读操作可以被 FLASH_ACR 寄存器的 Latency 位控制，即读取 flash 增加一个或者不增加等待状态。当为 0，则不增加 flash 读操作的等待状态；当为 1，flash 读操作增加 1 个等待状态。该机制是为了匹配高速的系统时钟和相对低速的 flash 读取速度，而进行的专门设计。

4.2.3. 闪存写操作和擦除操作

通过 ICP（In-circuit programming）或者 IAP（In-application programming）可以对 flash 进行 program 操作。

ICP：用来更新整个 Flash 存储器的内容，可以使用 SWD 协议或者 boot loader，把用户应用装入 MCU 中。ICP 提供了快速和有效的设计迭代，并消除了不必要的包处理或者 socketing。

IAP：可以使用芯片支持的通讯接口，下载要 program 的数据到 flash 中。IAP 允许用户在应用运行时，再次 program flash 存储器。然后，此时 flash 存储器中已有了之前使用 ICP 编程进去的部分应用程序。

如果在进行闪存写和擦除操作时，发生了复位，则闪存存储器的内容是不被保护的。

在闪存写和擦除操作期间，任何读闪存的操作都会拖延总线。写或擦除操作一结束，读操作就可以正确的进行。这也就意味着，当正在进写和擦除操作时，不能进行代码和数据的读取。

对于写和擦除操作，必须打开 HSI。

通过以下控制接口相关的寄存器，可以实现写和擦除操作：

- Acess control register(FLASH_ACR)
- KEY register(FLASH_KEYR）
- Option byte key register (FLASH_OPTKEYR)
- Flash status register (FLASH_SR)
- Flash control register (FLASH_CR)
- Flash option register(FLASH_OPTR)
- Flash special area address register(FLASH_SAR)
- Flash write protection resister (FLASH_WRPR)
- Flash TS0 register(FLASH_TS0)
- Flash TS1 register(FLASH_TS1)
- Flash TS2P register(FLASH_TS2P)
- Flash TPS3 register(FLASH_TPS3)
- Flash TS3 register(FLASH_TS3)
- Flash page erase TPE register(FLASH_PERTPE)
- Flash sector/mass erase TPE register(FLASH_SMERTPE)
- Flash program TPE register(FLASH_PRGTPE)
- Flash pre-program TPE register(FLASH_PRETPE)

4.2.3.1. 闪存解锁

在复位后，flash 存储器会被保护，防止不想要的（比如电干扰引起的）写和擦除操作。写 FLASH_CR 寄存器是不被允许的（除了用作 reload option bytes 的 OBL_LAUNCH 位）。每次对 flash 的写和擦除操作，都必须通过写 FLASH_KEYR 寄存器，产生 Unlock 时序，启用 FLASH_CR 寄存器的访问。

具体步骤如下：

步骤 1：向 FLASH_KEYR 寄存器写入 KEY1=0x4567 0123

步骤 2：向 FLASH_KEYR 寄存器写入 KEY2=0xCDEF 89AB

任何错误的时序都会锁住 FLASH_CR 寄存器，直到下一次复位。在错误的 KEY 时序时，总线错误被发现，并产生 Hard Fault 中断。这样的错误包括第一个写周期的 KEY1 不匹配，或者 KEY1 匹配，但第二个写周期的 KEY2 不匹配。

FLASH_CR 寄存器可以通过软件写 FLASH_CR 寄存器的 LOCK 位被再次锁住。

另外，当 FLASH_SR 寄存器的 BSY 位被置位时，FLASH_CR 寄存器不能被写。此时，任何尝试进行写该寄存器（FLASH_CR）的操作会引起 AHB 总线的拖延，直到 BSY1 位被清零。

4.2.3.2. 闪存写操作

Flash 存储器每次以 32bit word 为单位（进行 half word 或者 byte 操作会产生 HardFault）进行整个 page 的 program 操作。当 FLASH_CR 寄存器的 PG 位被置位，CPU 向 FLASH 存储器地址空间写 32bit 数据时，program 操作开始启动。任何非 32bit 的写入将导致 hard fault 中断。

如果要 program 的 flash 地址空间，是被 FLASH_WRPR 寄存器设置为保护的区域，则 program 操作会被忽略掉，同时 FLASH_CR 寄存器 WRPRTERR 位会被置位。Program 操作的结束，FLASH_CR 寄存器的 EOP 位会被置位。

具体 flash program 的操作步骤如下所示：

1) 检查 FLASH_SR 寄存器的 BSY 位，判断是否当前没有正在继续的 flash 操作
2) 如果没有正在进行的 flash erase 或者 program 操作，则软件读出该 Page 的 32 个 word（如果该 page 已有数据存放，则进行该步骤，否则跳过该步骤）
3) 向 FLASH_KEYR 寄存器依次写 KEY1 和 KEY2，解除 FLASH_CR 寄存器的保护
4) 置位 FLASH_CR 寄存器的 PG 位和 EOPIE 位
5) 向目标地址进行第 1 到第 31 个 word 的 program 操作（只接受 32bit 的 program）
6) 置位 FLASH_CR 寄存器的 PGSTRT
7) 写第 32 个 word
8) 等待 FLASH_SR 寄存器的 BSY 位被清零
9) 检查 FLASH_SR 寄存器的 EOP 标志位（当 program 操作已经成功，该位被置位），然后软件清零该位
10) 如果不再有 program 操作，则软件清除 PG 位

当上述步骤 7）成功执行，则 program 操作自动启动，同时 BSY 位被硬件置位。

闪存擦除操作

Flash 存储器可以按照 page 进行 erase 操作，或者进行 sector 和 mass erase（sector 和 mass erase 对 information memory 不起作用）。

4.2.3.3. Page erase

当某个 page 被 WRP 保护，它是不会被 erase 的，此时 WRPERR 位被置位。当要进行 page erase 操作时，要进行以下步骤：

1) 检查 FLASH_SR 寄存器 BSY 位，确认没有正在进行的 flash 操作
2) 向 FLASH_KEYR 寄存器依次写 KEY1 和 KEY2，解除 FLASH_CR 寄存器的保护
3) 置位 FLASH_CR 寄存器的 PER 位和 EOPIE 位
4) 向该 page 写任意数据（必须 32bit 数据）
5) 等待 BSY 位被清零
6) 检查 EOP 标志位被置位
7) 清零 EOP 标志

4.2.3.4. 闪存片擦

Mass erase 用来对整片 main flash 进行擦除操作，但对 information 区不起作用。另外，当 WRP 被使能，mass erase 功能无效，不会产生 mass erase 操作，并且 WEPERR 位被置位。

进行 mass erase 的步骤如下：

1) 检查 BSY 位，确认是否没有正在进行的 Flash 操作
2) 向 FLASH_KEYR 寄存器依次写 KEY1,KEY2，解除 FLASH_CR 寄存器保护

3) 置位 FLASH_CR 寄存器的 MER 位和 EOPIE 位
4) 向 flash 的任意 main flash 空间写任意数据（32bit 数据）
5) 等待 BSY 位被清零
6) 检查 EOP 标志位被置位
7) 清零 EOP 标志

4.2.3.5. Sector erase

Sector erase 用来对 4KB 的 main flash 进行擦除操作，但对 information 区不起作用。另外，当某个 sector 被 WRP 保护，它是不会被擦除的，此时 WRPERR 位被置位。

进行 sector erase 的步骤如下：

1) 检查 BSY 位，确认是否没有正在进行的 Flash 操作
2) 向 FLASH_KEYR 寄存器依次写 KEY1、KEY2，解除 FLASH_CR 寄存器保护
3) 置位 FLASH_CR 寄存器的 SER 位和 EOPIE 位
4) 向该 sector 写任意数据
5) 等待 BSY 位被清零
6) 检查 EOP 标志位被置位
7) 清零 EOP 标志

4.2.3.6. 写和擦除时间配置

Flash 的 program 和 erase 的时间需要进行严谨的控制，否则会造成操作失败。如果需要对 Flash 进行 program 和 erase 的操作，需要根据 HSI 输出频率，参考 FLASH_TS0, FLASH_TS1, FLASH_TS2P, FLASH_TPS3, FLASH_TS3, FLASH_PERTPE, FLASH_SMERTPE, FLASH_PRGTPE, FLASH_PRETPE 的描述对 Flash program 和 erase 时间控制寄存器进行正确的配置。

4.3. 产品唯一身份标识码（UID）寄存器

唯一身份标识码典型应用场景：

- 用作序列号
- 对内部闪存编程时，将其用作密钥或加密原语以提高代码的安全性
- 激活安全自举过程等

产品唯一身份标识提供了一个对于任何设备都唯一的参考号码。

用户永远不能改变这些位。唯一身份标识符也可以以单字节/半字/字等不同方式读取，然后使用自定义算法连接起来。

基址：0x1FFF 0E00

表 4-2 UID 格式

偏移地址	描述	UID Bits							
		7	6	5	4	3	2	1	0
0	Lot Numer	Lot Number ASCII 码							
1	Lot Numer	Lot Number ASCII 码							
2	Lot Numer	Lot Number ASCII 码							
3	Lot Numer	Lot Number ASCII 码							
4	Wafer Number	Wafer Number							
5	Lot Numer	Lot Number ASCII 码							

偏移地址	描述	UID Bits							
		7	6	5	4	3	2	1	0
6	Lot Numer	Lot Number ASCII 码							
7	Lot Numer	Lot Number ASCII 码							
8	内部编码	内部编码							
9	Y 坐标低位	Y 坐标低位							
10	X 坐标低位	X 坐标低位							
11	X,Y 坐标高地址	Y 坐标高位				X 坐标高位			
12	固定码	0x78							
13	内部编码	内部编码							
14	内部编码	内部编码							
15	内部编码	内部编码							

4.4. Flash 选项字节

4.4.1. Flash 选项字

芯片内的 flash 的 information 区域的部分区间作为选项字节使用，用来存放芯片或者用户针对应用需要对硬件进行的配置。比如，看门狗可以选择为硬件或者软件模式。

为了数据的安全性，选项字节以正文及反码形式分别存储。

表 4-3 选项字节格式

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

选项字节的内容可以从表选项字节 organization 所述的存储器地址读到，也可以从以下选项字节的相关寄存器读到：

- FLASH user option 寄存器（FLASH_OPTR）
- FLASH SDK area address 寄存器（FLASH_SDKR）
- FLASH WRP address 寄存器（FLASH_WRPR）

表 4-4 选项字节结构

word 地址	描述
0x1FFF 0E80	Option byte for Flash User option and its complemented
0x1FFF 0E84	Option byte for Flash SDK area address and its complemented
0x1FFF 0E88	Reserved
0x1FFF 0E8C	Option byte for Flash WRP address and its complemented
0x1FFF 0E90	Reserved
0x1FFF 0E94	Reserved
…	Reserved
…	Reserved
…	Reserved
0x1FFF 0EFC	Reserved

- **Option byte for Flash User option**

Flash memory address: 0x1FFF 0E80

Production value:0x4155 BEAA

在上电复位（POR/BOR/OBL_LAUNCH）释放后,从 flash information memory 的 option bytes 区域读出相应的值，写入到该寄存器相应的 option bit。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
~nBOOT1	~NRST_	Res	~IWDG	~BOR_LEV[2:0]			~BOR_	~RDP[7:0]							

	MODE		_SW				EN								
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**
nBOOT1	NRST_ MODE	Res	IWDG _SW	BOR_LEV[2:0]			BOR_ EN	RDP[7:0]							
R	R		R	R	R	R	R	R	R	R	R	R	R	R	R

Bit	Name	R/W	Function
31	~nBOOT1	R	nBOOT1 的反码
30	~NRST_MODE	R	NRST_MODE 的反码
29	Reserved		
28	~IWDG_SW	R	IWDG_SW 的反码
27：25	~BOR_LEV[2:0]	R	BOR_LEV 的反码
24	~BOR_EN	R	BOR_EN 的反码
23：16	~RDP	R	RDP 的反码
15	nBOOT1	R	与 BOOT PIN 一起，选择芯片启动模式
14	NRST_MODE	R	0：仅复位输入 1：GPIO 功能
13	Reserved		
12	IWDG_SW	R	0：硬件 watchdog 1：软件 watchdog
11：9	BOR_LEV[2:0]	R	000：BOR 上升阈值为 1.8V，下降阈值位 1.7V 001：BOR 上升阈值为 2.0V，下降阈值位 1.9V 010：BOR 上升阈值为 2.2V，下降阈值位 2.1V 011：BOR 上升阈值为 2.4V，下降阈值位 2.3V 100：BOR 上升阈值为 2.6V，下降阈值位 2.5V 101：BOR 上升阈值为 2.8V，下降阈值位 2.7V 110：BOR 上升阈值为 3.0V，下降阈值位 2.9V 111：BOR 上升阈值为 3.2V，下降阈值位 3.1V
8	BOR_EN	R	BOR enable 0：BOR 不使能 1：BOR 使能，BOR_LEV 起作用
7：0	RDP	R	0xAA：level 0, read protection inactive 非 0xAA：level 1, read protection active

- **Option byte for flash SDK area address**

Flash memory address: 0x1FFF 0E84

Production value: 0xFF00 00FF

在上电复位（POR/BOR/OBL_LAUNCH）释放后,从 flash information memory 的 option bytes 区域读出相应的值，写入到该寄存器相应的 option bit。

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

Bit	Name	R/W	Function
31：16	Reserved		
28：24	Complemented SDK_END[4:0]	R	SDK_END 的反码
23：21	Reserved		
20：16	Complemented SDK_STRT[4:0]	R	SDK_STRT 的反码
15：13	Reserved		
12：8	SDK_END[4:0]	R	SDK area end address，每一位对应的 STEP 为 2KB
7：5	Reserved		
4：0	SDK_STRT[4:0]	R	SDK area start address，每一位对应的 STEP 为 2KB

- **Option byte for Flash WRP address**

Flash memory address: 0x1FFF 0E8C

Production value: 0x0000 FFFF

在上电复位（POR/BOR/OBL_LAUNCH）释放后,从 flash information memory 的 option bytes 区域读出相应的值，写入到该寄存器相应的 option bit。

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

Bit	Name	R/W	Function
31：16	Complemented WRP	R	WRP 的反码
15：0	WRP	R	0：sector[y]被保护 1：sector[y]无保护 y=0~15

4.4.2. Flash 选项字节写

复位后，FLASH_CR 寄存器中与选项字节相关的位是被写保护的。当对选项字节进行相关操作前，FLASH_CR 寄存器中的 OPTLOCK 位必须被清零。

以下步骤用来解锁该寄存器：

1) 通过解锁时序，解锁 FLASH_CR 寄存器的写保护
2) 向 FLASH_OPTKEYR 寄存器，写 OPTKEY1=0x0819 2A3B
3) 向 FLASH_OPTKEYR 寄存器，写 OPTKEY2=0x4C5D 6E7F

任何错误的时序都会锁住 FLASH_CR 寄存器，直到下一次复位。在错误的 KEY 时序时，总线错误被发现，并产生 Hard Fault 中断。

User option（information flash 的 option bytes）可以通过软件写 FLASH_CR 寄存器的 OPTLOCK 位，被保护住，以防止不想要的 erase/program 操作。

如果软件置位 Lock 位，则 OPTLOCK 位也被自动置位。

Modifying user option bytes

选项字节的写操作（program），跟对 Main flash 的操作不一样。为修改选项字节，需要进行如下步骤：

1) 用之前描述的步骤，清零 OPTLOCK 位
2) 检查 BSY 位，确认没有正在进行的 Flash 操作
3) 向 option bytes 寄存器 FLASH_OPTR/FLASH_SAR/FLASH_WRPR 写期望的值（1~3 个 word）
4) 置位 OPTSTRT 位
5) 向 main flash 0x4002 2080 地址写任意 32bit 数据（触发正式的写操作）
6) 等待 BSY 位被清零
7) 等待 EOP 拉高，软件清零

任何对 Option bytes 的改动，硬件都会先把 option byte 对应的整个 page erase 掉，然后用 FLASH_OPTR、FLASH_SAR 或者 FLASH_WRPR 寄存器的值，写到 option bytes 中。并且，硬件自动计算相应的补码，并把计算值写到 option bytes 的相应区域。

Option byte loading

在 BSY 位被清零后，所有新的 option bytes 被写入了 flash information 存储器中，但是未应用于芯片系统。对 option bytes 寄存器进行读操作，仍然返回上一次被装载的 option bytes 里的值。仅当他们（新值）被装载后，才对芯片系统起作用。

Option bytes 的装载，在以下两种情况下进行：

- 当 FLASH_CR 寄存器中的 OBL_LAUNCH 位被置位
- 在上电复位后（POR、BOR）

“装载 option bytes”进行的操作是：对 information memory 区域的 option bytes 进行读操作，再把读出的数据存储在内部 option 寄存器中（FLASH_OPTR、FLASH_SAR 和 FLASH_WRPR）。这些内部寄存器配置系统，并可以被软件读。置位 OBL_LAUNCH 位，产生了一个复位，这样 option bytes 的装载，才能在系统的复位下进行。

每个 option 位在它相同的双字地址（下一个 half word）有相应的补码。在 option bytes 装载期间，会对 option bit 和其补码的验证，这能确保装载被正确的进行了。

如果正补码匹配，则 option bytes 被复制到 option 寄存器中。

如果正补码不匹配，则 FLASH_SR 寄存器的 OPTVERR 状态位被置位。不匹配的值被写入 option 寄存器：

- 对于 user option
  - BOR_LEV 写成 000（最低阈值）
  - BOR_EN 位写成 0（BOR 不使能）
  - NRST_MODE 位写成 0（仅复位输入）
  - RDP 位写成 0xff（即 level 1）
  - 其余不匹配的值都写成 1
- 对于 SDK area option，SDKR_STRT[4:0]= 0x00，SDKR_END[4:0]=0x1F，即所有 flash 空间都被设定为 SDK
- 对于 WRP option，不匹配的值是缺省值“无保护”

在系统复位后，option bytes 的内容被复制到下面的 option 寄存器（软件可读可写）：

- FLASH_OPTR
- FLASH_SDKR
- FLASH_WRPR

这些寄存器也被用来修改 option bytes。如果这些寄存器不被用户修改，他们体现了系统 option 的状态。

4.5. Flash 配置字节

芯片内的 flash 的 information 区域的部分区间（共 1 个 page）作为 Factory config. byte 使用。

Page 0 存放供软件读取信息（仅有正码，无反码存放）：

- HSI 频率选择控制值，及对应的 Trimming 值
- 对应 HSI 不同频率的擦写时间配置参数值

表 4-5 Factory config. byte organization

Page	Word	Address	Contents
0	0	0x1FFF 0F00	RESERVED
	1	0x1FFF 0F04	存放 HSI 8MHz 频率选择控制及对应的 Trimming 值
	2	0x1FFF 0F08	RESERVED
	3	0x1FFF 0F0C	RESERVED
	4	0x1FFF 0F10	存放 HSI 24MHz 频率选择控制及对应的 Trimming 值
	5	0x1FFF 0F14	TS_CAL1， 低温温度传感器的校准值

	6	0x1FFF 0F18	TS_CAL2， 高温温度传感器的校准值
	7	0x1FFF 0F1C	RESERVED
	8	0x1FFF 0F20	RESERVED
	9	0x1FFF 0F24	RESERVED
	10	0x1FFF 0F28	RESERVED
	11	0x1FFF 0F2C	RESERVED
	12	0x1FFF 0F30	存放 HSI 8MHz 频率下对应的 FLASH_TSO、FLASH_TS1 寄存器的配置值
	13	0x1FFF 0F34	存放 HSI 8MHz 频率下对应的 FLASH_TS2P、FLASH_TPS3 寄存器的配置值
	14	0x1FFF 0F38	存放 HSI 8MHz 频率下对应的 FLASH_PERTPE 寄存器的配置值
	15	0x1FFF 0F3C	存放 HSI 8MHz 频率下对应的 FLASH_SMERTPE 寄存器的配置值
	16	0x1FFF 0F40	存放 HSI 8MHz 频率下对应的 FLASH_PRGTPE、FLASH_PRETPE 寄存器的配置值
	17	0x1FFF 0F44	RESERVED
	18	0x1FFF 0F48	RESERVED
	19	0x1FFF 0F4C	RESERVED
	20	0x1FFF 0F50	RESERVED
	21	0x1FFF 0F54	RESERVED
	22	0x1FFF 0F58	RESERVED
	23	0x1FFF 0F5C	RESERVED
	24	0x1FFF 0F60	RESERVED
	25	0x1FFF 0F64	RESERVED
	26	0x1FFF 0F68	RESERVED
	27	0x1FFF 0F6C	存放 HSI 24MHz 频率下对应的 FLASH_TSO、FLASH_TS1 寄存器的配置值
	28	0x1FFF 0F70	存放 HSI 24MHz 频率下对应的 FLASH_TS2P、FLASH_TPS3 寄存器的配置值
	29	0x1FFF 0F74	存放 HSI 24MHz 频率下对应的 FLASH_PERTPE 寄存器的配置值
	30	0x1FFF 0F78	存放 HSI 24MHz 频率下对应的 FLASH_SMERTPE 寄存器的配置值
	31	0x1FFF 0F7C	存放 HSI 24MHz 频率下对应的 FLASH_PRGTPE、FLASH_PRETPE 寄存器的配置值
1	0	0x1FFF 0F80-0x1FFF 0FFF	RESERVED
-	-	0x1FFF 0E20	1.2V 电压校准值，例如 vrefint_verify 电压值为 1.203V，那么 0x1FFF 0E23 地址写入 12，0x1FFF 0E22 地址写入 03

4.5.1. HSI_TRIMMING_FOR_USER

Address: 0x1FFF 0F00~0x1FFF 0F10

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

软件需要从该地址读出数据，再写入 RCC_ICSCR 寄存器对应的 HSI_FS[2:0]和 HSI_TRIM[12:0]，以实现 HSI 频率的更改。

4.5.2. 温度传感器的校准值

Address: 0x1FFF 0F14(30℃)、0x1FFF 0F18(85℃或 105℃)

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

软件需要从该地址读出数据。

4.5.3. HSI_8M/24M_EPPARA0

Address: 0x1FFF 0F30(8MHz)、0x1FFF 0F6C(24MHz)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	TS1[8:0]								
							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**

TS3[7:0]								TS0[7:0]							
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

软件需要根据需要设定的 HSI 时钟频率，选择从相应地址读出数据，再写入 FLASH_TS0、FLASH_TS1、FLASH_TS3 寄存器，以实现对应 HSI 频率所需的擦写时间的配置。

4.5.4. HSI_8M/24M_EPPARA1

Address: 0x1FFF 0F34(8MHz)、0x1FFF 0F70(24MHz)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	TPS3[10:0]										
					R	R	R	R	R	R	R		R	R	R
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res	Res	Res	Res	Res	Res	Res	Res	TS2P[7:0]							
								R	R	R	R	R	R	R	R

软件需要根据需要设定的 HSI 时钟频率，选择从相应地址读出数据，再写入 FLASH_TS2P、FLASH_TPS3 寄存器，以实现对应 HSI 频率所需的擦写时间的配置。

4.5.5. HSI_8M/24M_EPPARA2

Address: 0x1FFF 0F38(8MHz)、0x1FFF 0F74(24MHz)

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

软件需要根据需要设定的 HSI 时钟频率，选择从相应地址读出数据，再写入 FLASH_PERTPE 寄存器中，以实现对应 HSI 频率所需的擦写时间的配置。

4.5.6. HSI_8M/24M_EPPARA3

Address: 0x1FFF 0F3C(8MHz)、0x1FFF 0F78(24MHz)

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

软件需要根据需要设定的 HSI 时钟频率，选择从相应地址读出数据，再写入 FLASH_SMERTPE 寄存器中，以实现对应 HSI 频率所需的擦写时间的配置。

4.5.7. HSI_8M/24M_EPPARA4

Address: 0x1FFF 0F40(8MHz)、0x1FFF 0F7C(24MHz)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	PRETPE[13:0]													
					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**
PRGTPE[15:0]															
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

软件需要根据需要设定的 HSI 时钟频率，选择从相应地址读出数据，再写入 FLASH_PRGTPE 和 FLASH_PRETPE 寄存器中，以实现对应 HSI 频率所需的擦写时间的配置。

4.6. 闪存保护

对 Flash main memory 的保护包括以下几种机制：

- SDK（software design kit）的保护，用来对特定程序区的访问保护，粒度是 2KB。
- 读保护(RDP)，防止来自外部的访问。
- 写保护（WRP）控制，以防止不想要的写操作（由于程序存储器指针 PC 的混乱）。写保护的粒度设计为 4KB。
- Option byte 写保护，专门的解锁设计。

4.6.1. 闪存软件开发包(SDK)区域保护

被 FLASH_SDKR 寄存器保护的区域，遵循 table 15 的权限描述。

保护区域由 FLASH_SDKR 寄存器的 SDKR_STRT[4:0],SDKR_END[4:0]定义，每一个位对应 2KB。

Start address

FLASH memory base address + SDK_STRT[4:0] x 0x800(included)

End address

FLASH memory base address + (SDK_END[4:0]+1) x 0x800(excluded)

当 SDK_STRT[4:0]大于 SDK_END[4:0]时，SDK 保护无效；当 SDK_STRT[4:0]小于或等于 SDK_END[4:0]时，SDK 保护有效。

在保护生效状态下，对 FLASH_SDKR 寄存器解除保护时（写 SDK_STRT[4:0]大于 SDK_END[4:0]），硬件会先触发 mass erase（SDK 区域被保护的程序之前已经写入，通过 mass erase 起到了对 SDK 区域程序保护的作用），然后再更新 flash 字节选项中的 SDK option 的值（此时更新的值是 SDK 保护无效）。

此时，FLASH_SDKR 寄存器的内容不会更新，直到上电复位（POR/BOR/PDR）或者 OBL 复位，寄存器内容才会被从 flash 字节选项中的 SDK option 装载到寄存器中。

4.6.2. 闪存读保护

通过设置 RDP option byte，并进行系统复位(POR/BOR 或者 OPL 复位)装载新的 RDP option byte，可以激活读保护功能。RDP 保护 flah main memory、option byte、SRAM。

如果通过 SWD 的调试仍在连接时，读保护被设置，需要进行上电复位而不是系统复位。

当 RDP option byte 和补码成对正确存在于 option byte 时，Flash memory 会被保护。

表 4-6 闪存读保护状态

RDP byte value	RDP complemented byte value	Read protection level
0xAA	0x55	Level 0
除(0xAA 和 0x55)组合的任何值		Level 1

无论任何保护级别，System memory 都只能访问，不能进行写和擦除操作。

Level 0: no protection

对 main flash 的读、写和擦操作是可能的，对 option byte 也是可以进行任何操作。

Level 1: Read protection

当 option byte 里的 RDP 及其补码包含任何（0xAA、0x55）之外的组合，则 level 1 读保护生效，Level 1 是缺省的保护级别。

- User mode：在用户模式下执行的程序（boot from main flash），可以对 main flash、option byte 进行所有操作。
- Debug, boot from SRAM 以及 boot from system memory mode（Boot loader）:在调试模式，或者当从 SRAM 或者 system memory（Boot loader）启动，main flash 是不能被访问的。在这些模式下，对 main flash 读或者写访问产生一个总线错误，以及产生一个 hard fault 中断。

  当已处于 Level 1（0xAA 之外任何数），如果要修改为 Level 0（写 0xAA），硬件会对 main flash 进行

mass erase 操作。

表 4-7 访问状态与保护级别和执行模式的关系

Area	READ Protection level	SDK Area Protection level	Boot From Main Flash(CPU)						Debug/ excuted From RAM/ excuted From System memory					
			User execution (From Non SDK Area)			User execution (From SDK Area)								
			Read	Write	Erase	Read	Write	Erase	Read	Write	Erase			
Non SDK Area	0	Disable	Yes	Yes	Yes	N/A	N/A	N/A	Yes	Yes	Yes			
		Enable	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes			
SDK Area		Disable	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A			
		Enable	No	No	No	Yes	Yes	Yes	No	No	No			
Non SDK Area	1	Disable	Yes	Yes	Yes	N/A	N/A	N/A	No	No	No			
		Enable	Yes	Yes	Yes	Yes	Yes	Yes	No	No	No			
SDK Area		Disable	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A			
		Enable	No	No	No	Yes	Yes	Yes	No	No	No			
System memory	x	Disable	Yes	No	No	N/A	N/A	N/A	Yes	No	No			
		Enable	Yes	No	No	Yes	No	No	Yes	No	No			
Option bytes area	x	Disable	Yes	Yes	Yes	N/A	N/A	N/A	Yes	Yes	Yes			
		Enable	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes			
Factory bytes	x	Disable	Yes	No	No	N/A	N/A	N/A	Yes	No	No			
		Enable	Yes	No	No	Yes	No	No	Yes	No	No			
UID	x	Disable	Yes	No	No	N/A	N/A	N/A	Yes	No	No			
		Enable	Yes	No	No	Yes	No	No	Yes	No	No			

Note:

(1) 任何区域发出的 mass erase 指令都会 erase 掉 SDK 区。

(2) 任何对 level 1 修改为 level 0，都会触发硬件对 main flash 的 mass erase。

(3) N/A 的含义是当 SDK Area disable 掉，由于不存在 SDK Area，上表 SDK Area 不存在读出程序的情况，也不存在从其他区域读出程序对 SDK Area 访问的情况。

(4) 对于从 SRAM 或者 system memory 执行程序包括两种情况：一个是 Boot from，另一个是从别的存储器 boot，程序跳转到 SRAM 或者 system memory。

4.6.3. 闪存写保护

Flash 可以被设置成写保护，以应对不想要的写操作。定义 WRP 寄存器每 bit 的控制粒度为 4KB 的写保护（WRP）区域，即 1 个 sector 大小。具体参见 WRP 寄存器的描述。

当被 WRP 的区域被激活，则不允许进行 erase 或者 program 操作。相应的，即使只有一个区域被设定为写保护，则 mass erase 功能不起作用。

此外，如果尝试对设为写保护的区域进行 erase 或者 program 操作，则 FLASH_SR 寄存器的写保护错误标识（WRPERR）会被置位。

注：写保护仅对 main flash 起作用，对 system memory 不起作用。

4.6.4. 选项字节写保护

缺省情况下，Option bytes 是可读，并进行写保护的。为获得对 option bytes 的 erase 或者 program 访问，需要向 OPTKEYR 寄存器写入正确的序列。

4.7. 闪存中断

表 4-8 闪存中断请求

中断事件	事件标志	时间标志/中断清除方法	控制位使能
End of operation	EOP	Write EOP=1	EOPIE
Write protection	WRPERR	Write WRPERR=1	ERRIE

注：以下事件没有单独的中断标识，但会产生 Hard fault：

- Unlock flash memory 的 FLASH_CR 寄存器的序列错误
- Unlock flash option bytes 的写操作序列错误
- FLASH program 操作未进行 32 位数据的对齐
- Flash erase（含 page erase、sector erase 和 mass erase）操作未进行 32 位数据对齐
- 对 option byte 寄存器的写操作未进行 32 位数据的对齐

4.8. 闪存寄存器描述

4.8.1. FLASH 访问控制寄存器 (FLASH_ACR)

Address offset: 0x00

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31：1	Reserved			
0	LATENCY	RW	0	Flash 读操作对应的等待状态： 0：flash 读操作没有等待状态 1：flash 读操作有 1 个等待状态，即每次读 flash 需要两个系统时钟周期 芯片最大工作频率为 24M，建议配置为 0

4.8.2. FLASH 密钥寄存器 (FLASH_KEYR)

Address offset: 0x08

Reset value: 0x0000 0000

所有寄存器位是 write-only，读出返回 0。

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

Bit	Name	R/W	Reset Value	Function
31：0	KEY[31:0]	W	0x0000	下面的值必须被连续的写入，才能 unlock FLASH_CR 寄存器，并使能了 flash 的 program/erase 操作 KEY1: 0x4567 0123 KEY2: 0xCDEF 89AB

4.8.3. FLASH 选项密钥寄存器 (FLASH_OPTKEYR)

Address offset: 0x0C

Reset value: 0x0000 0000

所有寄存器位是 write-only，读出返回 0。

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

Bit	Name	R/W	Reset Value	Function
31：0	OPTKEY[31:0]	W	0x0000 0000	下面的值必须被连续的写入，才能 unlock flash 的 option 寄存器，并使能了 option byte 的 program/erase 操作 KEY1: 0x0819 2A3B KEY2: 0x4C5D 6E7F

4.8.4. FLASH 状态寄存器 (FLASH_SR)

Address offset: 0x10

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	BSY
															R
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
OPTV ERR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	WRP ERR	Res	Res	Res	EOP
RC_W1											RC_W1				RC_W1

Bit	Name	R/W	Reset Value	Function
31：17	Reserved			
16	BSY	R	0	Busy 位 该位表示 flash 的操作正在进行。该位在 flash 操作的开始被硬件置位，当操作完成或者错误产生时由硬件清零。
15	OPTVERR	RC_W1	0	Option and trimming bits loading validity error 当 option 和 trimming bit 及其反码不匹配时，硬件置位该位。装载不匹配的 option bytes，被强制成安全值。 软件写 1，清零。
14:5	Reserved			
4	WRPERR	RC_W1	0	Write protection error 当要被 program/erase 的地址处于被写保护的 flash 区域时（WRP），硬件置位该位。 写 1，清零该位。
3：1	Reserved			
0	EOP	RC_W1	0	当 flash 的 program/erase 操作成功完成，硬件置位。该位仅当如果 FLASH_CR 寄存器的 EOPIE 位使能才会被置位。 写 1，清零该位。

4.8.5. FLASH 控制寄存器(FLASH_CR)

Address offset: 0x14

Reset value: 0xC000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LOCK	OPTLOCK	Res	Res	OBL_LAUNCH	Res	ERRIE	EOPIE	Res	Res	Res	Res	PGSTRT	Res	OPTSTRT	Res
RS	RS			RC_W1		RW	RW					RW		RW	
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res	Res	Res	Res	SER	Res	Res	Res	Res	Res	Res	Res	Res	MER	PER	PG
				RW									RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31	Lock	RS		FLASH_CR Lock 位。 软件对该位只能置位。当置位后，FLASH_CR 寄存器被 Lock 住。当成功给出 unlock 时序后，该位被硬件清零，unlock 了 FLASH_CR 寄存器。 【软件要在 program/erase 操作完成后，置位该位】 当不成功的 unlock 时序给出，该位仍然保持置位状态，直到下一次系统复位。
30	OPTLOCK	RS		Option bytes Lock 位。 软件对该位只能置位。当置位后，FLASH_CR 寄存器中与 option bytes 有关的位被 Lock 住。当成功给出 unlock 时序后，该位被硬件清零，unlock 了 FLASH_CR 寄存器。 【软件要在 program/erase 操作完成后，置位该位】 当不成功的 unlock 时序给出，该位仍然保持置位状态，直到下一次系统复位。
29：28	Reserved			
27	OBL_LAUNCH	RC_W1		Force the option bytes loading。 当置位时，该位强制系统进行 option bytes 的重装载。该位仅当 option byte 装载被完成后被硬件清零。如果 OPT-LOCK 位被置位，该位不能被写. 0：Option byte loading 完成 1：产生 Option byte loading 请求，系统产生复位，进行 option byte 的重装载。
25	ERRIE	RW		Error interrupt enable 位，当 FLASH_SR 寄存器的 WRPERR 位被置位，如果该位使能，则产生中断请求。 0：无中断产生 1：有中断产生
24	EOPIE	RW		End of operation interrupt enable 当 FLASH_SR 寄存器的 EOP 位被置位，该位使能中断的产生。 0：EOP 中断关闭 1：EOP 中断使能
23：18	Reserved	RW		
19	PGSTRT	RW		Flash main memory 的 program 操作的启动位。 该位启动了 Flash main memory 的 program 操作，软件置位，在 FLASH_SR 寄存器的 BSY 位被清零后，硬件清零该位。
18	Reserved			
17	OPTSTRT	RW		Flash option bytes 修改的启动位 该位启动了对 option bytes 的修改。软件置位，在 FLASH_SR 寄存器的 BSY 位被清零后，硬件清零该位。 注意：当对 flash option bytes 进行修改时，硬件自动把整个 128Bytes 的 page 进行 erase 操作，再进行 program 操作，其中也包括自动进行补码的写入。
16:12	Reserved			
11	SER	RW		4kByte 的 Sector erase 操作 0：未选择 flash 的 sector erase 操作 1：选择 flash 的 sector erases 操作 注：

Bit	Name	R/W	Reset Value	Function
				1） Sector erase 不会对 flash information memory 起作用。 2） Sector erase 对设定为 WRP 的区域不起作用。
10：3	Reserved			
2	MER	RW		Mass erase 操作 0：未选择 flash 的 mass erase 操作 1：选择 flash 的 mass erases 操作 注： Mass erase 不会对 flash information memory 起作用。当有 WRP 设定时，Mass erase 不起作用
1	PER	RW		Page erase 操作 0：未选择 flash 的 page erase 操作 1：选择 flash 的 page erase 操作
0	PG	RW		Program 操作 0：未选择 flash 的 program 操作 1：选择 flash 的 program 操作

4.8.6. FLASH 选项寄存器 (FLASH_OPTR)

Address offset: 0x20

Reset value: 0x0000 xxxx。在上电复位（POR/BOR/OBL_LAUNCH）释放后,从 flash information memory 的 option bytes 区域读出相应的值，写入到该寄存器相应的 option bit。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
nBOOT1	NRST_MODE	Res	IWDG_SW	BOR_LEV[2:0]			BOR_EN	RDP[7:0]							
RW	RW		RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31：16	Reserved			
15	nBOOT1	RW		与 BOOT PIN 一起，选择芯片启动模式
14	NRST_MODE	RW		0：仅复位输入 1：GPIO：GPIO 功能
13	Reserved			
12	IWDG_SW	RW		0：硬件 watchdog 1：软件 watchdog
11：9	BOR_LEV[2:0]	RW		000：BOR 上升阈值为 1.8V，下降阈值位 1.7V 001：BOR 上升阈值为 2.0V，下降阈值位 1.9V 010：BOR 上升阈值为 2.2V，下降阈值位 2.1V 011：BOR 上升阈值为 2.4V，下降阈值位 2.3V 100：BOR 上升阈值为 2.6V，下降阈值位 2.5V 101：BOR 上升阈值为 2.8V，下降阈值位 2.7V 110：BOR 上升阈值为 3.0V，下降阈值位 2.9V 111：BOR 上升阈值为 3.2V，下降阈值位 3.1V
8	BOR_EN	RW		BOR enable 0：BOR 不使能 1：BOR 使能，BOR_LEV 起作用
7：0	RDP	RW		0xAA：level 0, read protection inactive 非 0xAA：level 1, read protection active

4.8.7. FLASH SDK 地址寄存器 (FLASH_SDKR)

Address offset: 0x24

Reset value: 32’b0000 0000 0000 0000 000X XXXX 000X XXXX。在上电复位（POR/BOR/OBL_LAUNCH）释放后,从 flash information memory 的 option bytes 区域读出相应的值，写入到该寄存器相应的 option bit。

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

Bit	Name	R/W	Reset Value	Function
31：13	Reserved			
12：8	SDK_END[4:0]	RW		SDK area end addres，每一位对应的 STEP 为 2KB
7：5	Reserved			
4：0	SDK_STRT[4:0]	RW		SDK area start address，每一位对应的 STEP 为 2KB

4.8.8. FLASH WRP 地址寄存器 (FLASH_WRPR)

Address offset: 0x2C

Reset value: 0x0000 XXXX

在上电复位（POR/BOR/OBL_LAUNCH）释放后,从 flash information memory 的 option bytes 区域读出相应的值，写入到该寄存器相应的 option bit。

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

Bit	Name	R/W	Reset Value	Function
31：5	Reserved			
4	WRP	RW	1	0：sector 4，有写保护，不允许进行 program 和 erase 1：sector 4，无写保护
3	WRP	RW	1	0：sector 3，有写保护，不允许进行 program 和 erase 1：sector 3，无写保护
2	WRP	RW	1	0：sector 2，有写保护，不允许进行 program 和 erase 1：sector 2，无写保护
1	WRP	RW	1	0：sector 1，有写保护，不允许进行 program 和 erase 1：sector 1，无写保护
0	WRP	RW	1	0：sector 0，有写保护，不允许进行 program 和 erase 1：sector 0，无写保护

4.8.9. FLASH 睡眠时间配置寄存器(FLASH_STCR)

Address offset: 0x90

Reset value: 0x0000 6400

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

Bit	Name	R/W	Reset Value	Function
31：8	Reserved			
15：8	SLEEP_TIME	RW	0x64	FLASH 睡眠时间计数(基于 HSI_10M 时钟的计数器）

Bit	Name	R/W	Reset Value	Function
				当系统时钟选择 LSI 时，为获得更优化的 Run 模式功耗，可选择使用该寄存器的功能（仅推荐在 LSI 为系统时钟时，使用该功能）。 当使能该功能时，每半个系统时钟低电平周期内 Flash 处于 Sleep 状态的时间宽度为： t_{HSI_10M} * SLEEP_TIME Note: t_{HSI_10M} 为 HSI_10M 的周期； 为确保 Flash 功能的正确，本寄存器最大设定值推荐设定为 0x28。
7：1	Reserved			
0	SLEEP_EN	RW	0	FLASH Sleep enable 1：enable flash sleep 0：disable flash sleep

4.8.10. FLASH TS0 寄存器 (FLASH_TS0)

Address offset: 0x100

Reset value: 0x0000 XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res	Res	Res	Res	Res	Res	Res	Res	TS0							
								RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31：8	Reserved			
7：0	TS0	RW	0xXXXX	软件通过读出存放在 information 区相应地址的数据，写入对应寄存器，以实现对应 HSI 频率所需的擦写时间的配置。 保存在 Flash 的如下地址内： 24MHz 校准值存放地址：0x1FFF 0F6C 8MHz 校准值存放地址：0x1FFF 0F30

4.8.11. FLASH TS1 寄存器 (FLASH_TS1)

Address offset: 0x104

Reset value: 0x0000 XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res	Res	Res	Res	Res	Res	Res	TS1								
							RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31：9	Reserved			
8：0	TS1	RW	0xXXXX	软件通过读出存放在 information 区相应地址的数据，写入对应寄存器，以实现对应 HSI 频率所需的擦写时间的配置。 保存在 Flash 的如下地址内： 24MHz 校准值存放地址：0x1FFF 0F6C 8MHz 校准值存放地址：0x1FFF 0F30

4.8.12. FLASH TS2P 寄存器 (FLASH_TS2P)

Address offset: 0x108

Reset value: 0x0000 XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res	Res	Res	Res	Res	Res	Res	Res	TS2P							
								RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31：8	Reserved			
7：0	TS2P	RW	0xXXXX	软件通过读出存放在 information 区相应地址的数据，写入对应寄存器，以实现对应 HSI 频率所需的擦写时间的配置。 保存在 Flash 的如下地址内： 24MHz 校准值存放地址：0x1FFF 0F70 8MHz 校准值存放地址：0x1FFF 0F34

4.8.13. FLASH TPS3 寄存器 (FLASH_TPS3)

Address offset: 0x10C

Reset value: 0x0000 XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res	Res	Res	Res	Res	TPS3										
					RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31：11	Reserved			
10：0	TPS3	RW	0xXXXX	软件通过读出存放在 information 区相应地址的数据，写入对应寄存器，以实现对应 HSI 频率所需的擦写时间的配置。 保存在 Flash 的如下地址内： 24MHz 校准值存放地址：0x1FFF 0F70 8MHz 校准值存放地址：0x1FFF 0F34

4.8.14. FLASH TS3 寄存器 (FLASH_TS3)

Address offset: 0x110

Reset value: 0x0000 XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res	Res	Res	Res	Res	Res	Res	Res	TS3							
															RW

Bit	Name	R/W	Reset Value	Function
31：8	Reserved			
7：0	TS3	RW	0xXXXX	软件通过读出存放在 information 区相应地址的数据，写入对应寄存器，以实现对应 HSI 频率所需的擦写时间的配置。 保存在 Flash 的如下地址内：

Bit	Name	R/W	Reset Value	Function
				24MHz 校准值存放地址：0x1FFF 0F6C 8MHz 校准值存放地址：0x1FFF 0F30

4.8.15. FLASH 页擦写（PAGE ERASE）TPE register (FLASH_PERTPE)

Address offset: 0x114

Reset value: 0x0001 XXXX

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

Bit	Name	R/W	Reset Value	Function
31：17	Reserved			
16：0	PERTPE	RW	0xXXXX	软件通过读出存放在 information 区相应地址的数据，写入对应寄存器，以实现对应 HSI 频率所需的擦写时间的配置。 保存在 Flash 的如下地址内： 24MHz 校准值存放地址：0x1FFF 0F74 8MHz 校准值存放地址：0x1FFF 0F38

4.8.16. FLASH SECTOR/MASS ERASE TPE 寄存器 (FLASH_SMERTPE)

Address offset: 0x118

Reset value: 0x0001 XXXX

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

Bit	Name	R/W	Reset Value	Function
31：17	Reserved			
16：0	SMERTPE	RW	0xXXXX	软件通过读出存放在 information 区相应地址的数据，写入对应寄存器，以实现对应 HSI 频率所需的擦写时间的配置。 保存在 Flash 的如下地址内： 24MHz 校准值存放地址：0x1FFF 0F78 8MHz 校准值存放地址：0x1FFF 0F3C

4.8.17. FLASH PROGRAM TPE register (FLASH_PRGTPE)

Address offset: 0x11C

Reset value: 0x0000 XXXX

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

Bit	Name	R/W	Reset Value	Function
31：16	Reserved			
15：0	PRGTPE	RW	0xXXXX	软件通过读出存放在 information 区相应地址的数据，写入对应寄存器，以实现对应 HSI 频率所需的擦写时间的配置。 保存在 Flash 的如下地址内： 24MHz 校准值存放地址：0x1FFF 0F7C 8MHz 校准值存放地址：0x1FFF 0F40

4.8.18. FLASH PRE-PROGRAM TPE 寄存器 (FLASH_PRETPE)

Address offset: 0x120

Reset value: 0x0000 XXXX

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

Bit	Name	R/W	Reset Value	Function
31：14	Reserved			
13：0	PRETPE	RW	0xXXXX	软件通过读出存放在 information 区相应地址的数据，写入对应寄存器，以实现对应 HSI 频率所需的擦写时间的配置。 保存在 Flash 的如下地址内： 24MHz 校准值存放地址：0x1FFF 0F7C 8MHz 校准值存放地址：0x1FFF 0F40

4.8.19. FLASH 寄存器映像

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	FLASH_ACR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LA-
	Reset value																																0
0x008	FLASH_KEYR	KEY[31:16]																KEY[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00C	FLASH_OPTKEYR	OPTKEY[31:16]																OPTKEY[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x010	FLASH_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BSY	OPTVER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRPER	Res.	Res.	Res.	EOP
	Reset value																0	0											0				0
0x014	FLASH_CR	LOCK	OPTLOCK	Res.	Res.	OBL_LAUNC	Res.	ERRIE	EOPIE	Res.	Res.	Res.	Res.	PGSTRT	Res.	OPTSTRT	Res.	Res.	Res.	Res.	Res.	SER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MER	PER	PG
	Reset value	0	0			0								0		0						0									0	0	0

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x20	FLASH_OPTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	nBOOT1	NRST_MODE	Res	IWDG_SW	BOR_LEV[2:0]			BOR_EN	RDP[7:0]							
	Reset value																	X	X		X	X	X	X	X	X	X	X	X	X	X	X	X
0x24	FLASH_SDKR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SA_END[4:0]					Res.	Res.	Res.	SA_STRT[4:0]				
	Reset value																				X	X	X	X	X				X	X	X	X	X
0x2C	FLASH_WRPR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRP[15:0]															
	Reset value																	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x90	FLASH_STCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SLEEP_TIME[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	SLEEP_E
	Reset value																	0	1	1	0	0	1	0	0								0
0x100	FLASH_TS0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TS0[7:0]							
	Reset value																									1	0	1	1	0	1	0	0
0x104	FLASH_TS1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TS1[8:0]							
	Reset value																								1	1	1	0	1	1	0	0	0
0x108	FLASH_TS2P	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TS2P[7:0]							
	Reset value																									1	0	1	1	0	1	0	0
0x10C	FLASH_TPS3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FLASH_TPS3[10:0]										
	Reset value																						1	1	0	1	1	0	0	0	0	0	0
0x110	FLASH_TS3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TS3[7:0]							
	Reset value																									1	0	1	1	0	1	0	0
0x114	FLASH_PERTPE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PERTPE	WRP[15:0]															
	Reset value																0	1	1	1	0	1	0	1	0	0	1	1	0	0	0	0	0
0x118	FLASH_SMERTPE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SMERTPE[16:0]																
	Reset value																0	1	1	1	1	1	1	0	1	0	0	1	0	0	0	0	0
0x11C	FLASH_PRGTPE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRGTPE[15:0]															
	Reset value																	1	0	0	0	1	1	0	0	1	0	1	0	0	0	0	0
0x120	FLASH_PRETPE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRETPE[13:0]													
	Reset value																			0	1	0	0	1	0	1	1	0	0	0	0	0	0

5. 电源控制

5.1. 电源

5.1.1. 电源框图

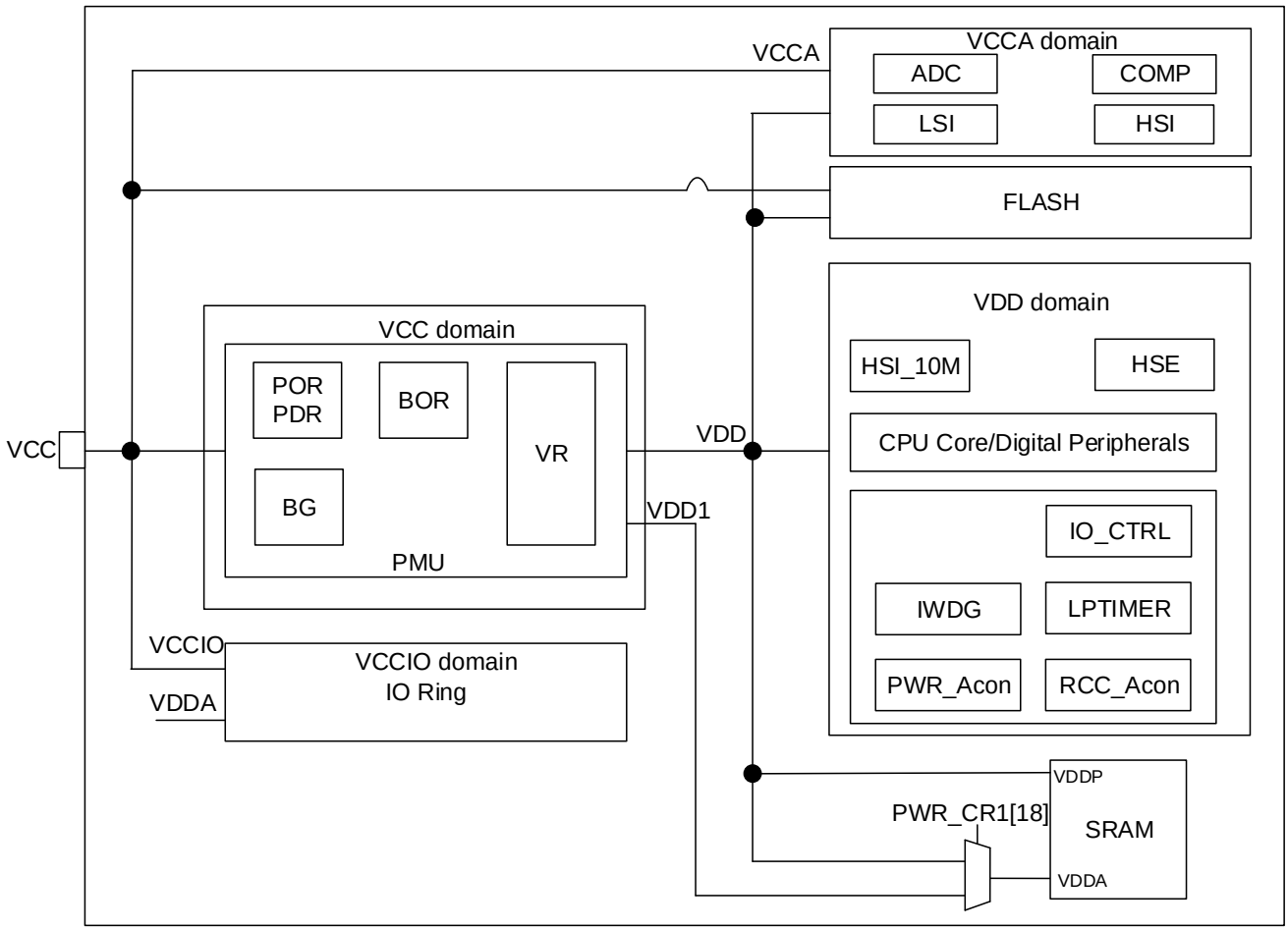


图 5-1 电源框图

表 5-1 电源框图

编号	电源	电源值	描述
1	VCC	1.7v~5.5v	通过电源管脚为芯片提供电源，其供电模块为：部分模拟电路。
2	VCCA	1.7v~5.5v	给大部分模拟模块供电，来自于 VCC PAD（也可设计单独电源 PAD）。
3	VCCIO	1.7v~5.5v	给 IO 供电，来自于 VCC PAD
4	VDD	1.2v/1.0v±10%	来自于 VR 的输出，为芯片内部主要逻辑电路、SRAM 供电。当 MR 供电时，输出 1.2v。当进入 stop 模式时，根据软件配置，可以由 MR 或者 LPR 供电，并根据软件配置决定 LPR 输出是 1.2v 或者 1.0v。

5.2. 电压调节器

芯片设计两个电压调节器：

- MR（Main regulator）在芯片正常运行状态时保持工作。
- LPR（low power regulator）在 stop 模式下，提供更低功耗的选择。

VDD 的电源根据芯片的工作模式，来自于 MR 或 LPR。

在芯片 run 模式，MR 保持工作，输出 1.2v 电压，LPR 关闭。

在 stop 模式，可由软件决定从 MR 或 LPR 供电。同样，由软件决定进入 stop 后，LPR 供电情况下的 VDD 是 1.2v 还是 1.0v。

5.3. 动态电压值管理

动态电压值管理指的是对 VR 的输出 VDD 电压进行调节，使芯片可以根据应用要求运行在不同的电压下，从而获得相应的性能及功耗。

本项目定义两种电压范围：

- **Range 1：高性能 Range**

MR 的输出为典型值 1.2V（VDD），系统时钟频率可以运行在最快的 24MHz 下。

- **Range 2：低功耗 Range**

只有当芯片处于 stop 模式时，才允许设定进入该 range，且该 range 只针对 LPR 起作用。

默认情况，LPR 的输出为典型值 1.2V（VDD），当置位寄存器 VOS 位时，芯片进入 stop 模式时，MR 切换成 LPR 供电（如果软件选择 stop 模式由 LPR 供电），且 LPR 被切换成典型值 1.0V（VDD）。此时，部分处于工作状态的逻辑电路（LPTIMER）可以运行在 LSI 下。

当芯片退出 stop 模式时，芯片恢复 MR 供电，VOS 位也由硬件清零。下次进入 stop 模式，如果想要获得更低的功耗，仍要软件置位 VOS 位，使芯片进入 stop 模式后的 LPR 供电为 1.0V。

5.4. 电源监控

5.4.1. 上电复位 (POR)/下电复位 (PDR)/欠压复位 (BOR)

芯片内设计 POR/PDR 模块，放在 VDD 电源域下，为芯片提供上电和下电复位。该模块在各种模式之下都保持工作。

除了 POR/PDR 外，还实现了 BOR（brown out reset）。BOR 仅可以通过 option byte，进行使能和关闭操作。

当 BOR 被打开时，BOR 的阈值可以通过 Option byte 进行选择，且上升和下降检测点都可以被单独配置。

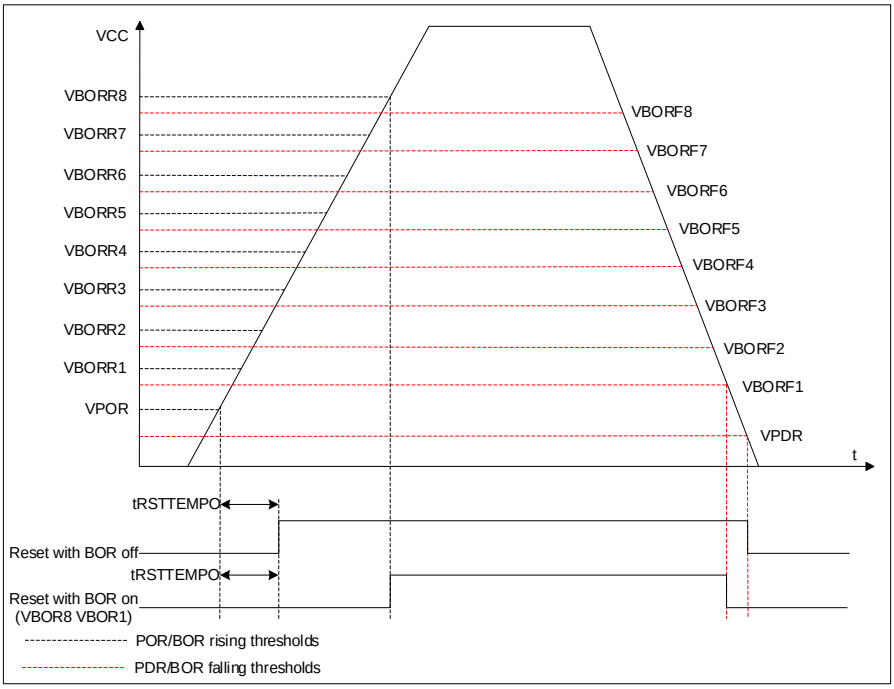


图 5-2 POR/PDR/BOR 阈值

6. 低功耗控制

缺省状态下，芯片在系统或者电源复位之后，进入正常运行 run 模式。当 CPU 不需要持续工作时，芯片可进入低功耗模式，例如，当等待外部事件时。软件可以在功耗、唤醒时间、唤醒源之间折中选择。

6.1. 低功耗模式

6.1.1. 低功耗模式介绍

芯片在正常的运行模式之外，有 2 个低功耗模式：

- **Sleep mode**：CPU 时钟关闭（NVIC，SysTick 等工作），外设可以配置为保持工作。（建议只使能必须工作的模块，在模块工作结束后关闭该模块）。
- **Stop mode**：该模式下 SRAM 和寄存器的内容保持，HSI 和 HSE 关闭，VDD 域下大部分模块的时钟都被停掉。

在 stop 模式，LSI 可以保持工作， LPTIMER 等可以保持工作。具体该模式下各模块的工作情况，参照表 6-2。

在 stop 模式下，对应的 VR 状态可由软件控制，设成 MR 或者 LPR 供电。当 LPR 供电时，芯片功耗大大降低，但唤醒时间较长；当保持 MR 供电的情况，芯片功耗较大，但具备几个周期的快速唤醒能力。

此外，正常运行模式下可以通过下述方法降低功耗：

- 降低系统时钟频率
- 对于不使用的外设，关掉外设时钟（系统时钟和模块时钟）

综上分析，本项目的低功耗模式转换图如下所述。

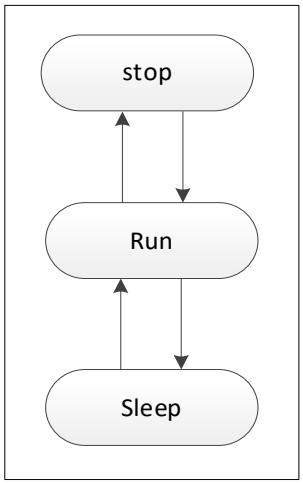


图 6-1 电源模式

6.1.2. 低功耗模式开关

表 6-1 低功耗模式开关

模式	进入	唤醒源	唤醒时钟	对时钟的影响	Voltage regulator	
					MR	LPR
Sleep (sleep-now or sleep-on-exit)	WFI or Return from ISR	任何中断	与进入 sleep 之前一样	CPU 时钟停止，对其他时钟和时钟源没影响。	开(1)	关
	WFE	唤醒事件				
Stop	SLEEPDEEP bit 1. WFI or 2. Return from ISR or	任何配置为唤醒的 EXTI Line	HSISYS HSI 保持进入 stop	HSI 关闭； HSE 关闭； LSI 可选择开或者关；	软件配置开关	软件配置开关，如果开，输

模式	进入	唤醒源	唤醒时钟	对时钟的影响	Voltage regulator	
					MR	LPR
	3. WFE Note: 系统时钟不能选择 LSI	（EXTI 寄存器配置）、IWDG、NRST	前的频率配置，不分频	LPTIMER、IWDG：由软件配置是否工作； 低功耗唤醒和部分 RCC 等模块保持工作； 其余模块的时钟关闭。		出电压 1.2/1.0v 可配置

注 1：软件要配置 VR 的状态为 MR 模式，才能进入 sleep 模式。

6.1.3. 各工作模式下的功能

表 6-2 各工作模式下的功能 (1)

Peripheral	Run	Sleep	Stop	
			VR@LPR or VR@MR	Wakeup ability
CPU	Y	-	-	-
Flash memory	Y	Y	- (2)	-
SRAM	Y	O (3)	- (4)	-
Brown-out reset (BOR)	Y	Y	O	O
HSI	O	O	-	-
HSE	O	O	-	-
LSI	O	O	O	-
HSE Clock Security System (CSS)	O	O	-	-
USART1	O	O	-	-
SPI1	O	O	-	-
ADC	O	O	-	-
COMP1/COMP2	O	O	O	O
Temperature sensor	O	O	-	-
Timers(TIM1/ TIM16)	O	O	-	-
LPTIM	O	O	O	O
IWDG	O	O	O	O
SysTick timer	O	O	-	-
CRC	O	O	-	-
GPIOs	O	O	O	O

注1. Y = Yes (使能)；O = Optional (默认关闭，可以软件使能)；- = Not available

注2. Flash 不下电，但无时钟提供，进入最低功耗状态。

注3. SRAM 的时钟可以被开或者关。

注4. SRAM 不下电，但无时钟提供，进入最低功耗状态。

6.2. Sleep mode

6.2.1. 进入 sleep mode

通过执行 WFI(wait for interrupt)或者 WFE(wait for event)指令，进入 sleep 模式。取决于 Cortex M0+的系统控制寄存器的 SLEEPONEXIT 位，有两种可选的进入 sleep 模式的机制。

- Sleep-now:如果 SLEEPONEXIT 位是 0，则执行 WFI 或者 WFE 后，立即进入 sleep 模式。
- Sleep-on-exit:如果 SLEEPONEXIT 位是 1，则当退出低优先级中断 ISR 时，进入 sleep 模式。

在 sleep 模式，所有的 IO pin 与 run 模式保持相同的状态。

6.2.2. 退出 sleep mode

如果用 WFI 进入 sleep 模式，被 NVIC 获得的任何外设中断可以把芯片从 sleep 模式唤醒。

如果用 WFE 进入 sleep 模式，当一个事件发生时，芯片退出 sleep 模式。唤醒事件可以通过以下方式产生：

- 在外设控制寄存器使能中断，而不是在 NVIC，并使能 Cortex M0+的 SEVONPEND 位。当芯片从 WFE 唤醒后继续执行时，外设中断 pending 位和外设 NVIC IRQ 通道 pending 位（在 NVIC 的中断清除 pending 寄存器）必须被清零。
- 或者，配置外部或者内部 EXTI line 为事件模式。当 CPU 从 WFE 唤醒后继续执行时，不必清除外设中断 pending 位，或者对应到事件 Line 的 NVIC IRQ 通道 pending 位没有被置位。

该模式具有最短的唤醒时间，并且没有在中断进入和退出浪费时间。

表 6-3 Sleep-now

Sleep-now mode	描述
进入模式	WFI 或者 WFE，并且： - SLEEPDEEP = 0 并且 - SLEEPONEXIT = 0
退出模式	如果通过 WFI 进入的睡眠模式，则退出方式是：中断。 如果通过 WFE 进入的睡眠模式，则退出方式是：唤醒事件。
唤醒延迟	无

表 6-4 Sleep-on-exit

Sleep-on-exit	描述
进入模式	WFI，并且： - SLEEPDEEP = 0 并且 - SLEEPONEXIT = 1
退出模式	中断
唤醒延迟	无

6.3. Stop 模式

Stop 模式是基于 Cortex-M0+的深度睡眠以及对外设时钟的门控，VR 可以被配置成 MR 或者 LPR 供电。在该模式下， HSI 和 HSE 被关闭，SRAM 和寄存器内容处于保持状态，LSI、LPTIMER、IWDG 可由软件配置是否工作，低功耗唤醒和部分 RCC 逻辑等保持工作，其余 VCORE 域的数字模块的时钟输入被关闭。

在 Stop 模式下，所有的 IO pin 保持跟运行模式相同的状态。

6.3.1. 进入 stop mode

为了进一步降低 stop 模式的功耗，配置 PWR_CR.LPR=1 时，VR 可以进入 LPR 供电。

如果正在进行 flash 的擦写操作，则 stop 模式的进入会被延迟，直到存储器访问结束（由软件读 FLASH_SR 寄存器的 BSY 位判断当前是否已完成擦、写操作）。

如果 APB 总线上的操作正在进行，则 stop 模式的进入也会被延迟，直到 APB 访问结束（由软件控制）。

6.3.2. 退出 stop mode

当通过中断或者唤醒事件退出 stop 模式时，HSI 被选择作为系统时钟。

在 stop 模式，如果 VR 处于 LPR 状态，则从 stop 模式唤醒有额外的稳定延迟。

在 stop 模式，如果 VR 处于 MR 状态，电流消耗会大，但唤醒时间会被减少。

表 6-5 stop mode

Stop mode	描述
进入模式	WFI(等待中断) 或者 WFE（等待事件），并且： - 配置设定： 1） 通过 PWR_CR 的 LPR 位，选择 VR 工作在 MR 或者 LPR 下 2） 通过 PWR_CR 的 VOS 位，选择 LPR 模式提供 1.2V 还是 1.0V 3） 通过 PWR_CR 的 SRAM_RETV 位，选择 SRAM 的 retention 电压 4） 通过 PWR_CR 的 MRRDY_TIME 和 FLS_SLPTIME 配置 MR 和 FLASH 的唤醒时间 - 置位 Cortex M0+的 SLEEPDEEP 位 **Note：**

Stop mode	描述
	为了进入 stop 模式，所有 EXTI line 的 pending 位（EXTI_PR 寄存器）、所有外设的中断 pending 位，必须被复位。否则，进入 stop 模式的流程将被忽略掉，程序继续执行。 如果应用需要在进入 stop 模式前关闭 HSE，系统时钟源必须首先切换到 HSI，然后清除 HSEON 位。 为使芯片功耗变化尽可能均衡，软件需要遵循逐步关闭的原则：逐步关闭各个模块的时钟，选择 HSI 作为系统时钟，关闭 HSE。 为缩短唤醒时间，在进入 stop 模式前，系统时钟应该配置为选择 HSI 高频时钟，RCC_CFGR 寄存器的 HPRE 设为 0，否则在唤醒后硬件切换时钟会消耗额外的时钟。
退出模式	如果使用 WFI 进入 stop 模式： - 任何被配置成中断模式的 EXTI line（相应的 EXTI 中断向量必须被在 NVIC 中使能） 如果使用 WFE 进入 stop 模式： - 任何被配置成事件模式的 EXTI line - CPU SEVONPEND 位置位情况下的中断 pending 位
唤醒延迟	LPR to MR wakeup time + HSI wakeup time + flash wakeup time

6.4. 降低系统时钟频率

在运行模式下，系统时钟的频率（SYSCLK, HCLK, PCLK）可以通过预分频寄存器配置分频去降低。这些预分频器也可以被用来在进入 sleep 模式前，降低外设的频率。

当系统运行在较低频率（32.768kHz），为获得更小的功耗，软件可以设定电压调节器（MR）的驱动能力配置位（PWR_CR1 寄存器的 BIAS_CR[3:0]），使 MR 自身的功耗大大降低。但要注意应该先降低系统时钟频率，后调整 MR 的驱动能力。反之，当要退出较低频率进入更高运行频率时，应该先调大 MR 的驱动能力，再改变系统时钟运行频率。

6.5. 外设时钟门控

在 run 模式，可以在任何时间停止单个外设和存储器的 AHB 时钟（HCLK）和 APB 时钟（PCLK），以降低功耗。

为了进一步降低在 sleep 模式的功耗，外设的时钟可以在执行 WFI 或者 WFE 指令之前被停掉。

6.6. 电源管理寄存器

该外设的寄存器可以通过 half-word 或者 word 访问。

6.6.1. 电源控制寄存器 1 (PWR_CR1)

Address offset: 0x00

Reset value: 0x0007 0000(reset by POR)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	HSION_CTRL	SRAM_RETV[2:0]		
												RW	RW	RW	RW
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res	LPR	FLS_SLPTIME[1:0]		MRRDY_TIME[1:0]		VOS	Res	Res	Res	Res	BIAS_CR_SEL	BIAS_CR[3:0]			
	RW	RW	RW	RW	RW	RW					RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:20	Reserved	-	-	Reserved

Bit	Name	R/W	Reset Value	Function
19	HSION_CTRL	RW	0	从 Stop 模式唤醒时，HSI 打开时间控制。 0：等待 MR 稳定后，使能 HSI； 1：与 VR 同时打开，即唤醒时立刻使能 HSI。
18:16	SRAM_RETV [2:0]	RW	111	Stop 模式下 SRAM retention 电压控制 000：Reserved 001：Reserved 010：Reserved 011：给 SRAM 提供 0.9V 电压 1xx：给 SRAM 提供 1.2V 或者 1.0V 电压（取决于 VOS bit）
15	Reserved			
14	LPR	RW	0	Low power regulator 0：Main regulator 工作在 stop 模式 1：Low power regulator 工作在 stop 模式
13:12	FLS_SLPTIME	RW	2’b00	Stop 模式唤醒时序中，在 HSI 稳定后，在 FLASH 操作前需要等待时间。 2’b00: 5us 2’b01: 2us 2’b10: 3us 2’b11: 0us 注：当该寄存器设置为 2’b11 时，表明唤醒后是从 SRAM 执行程序，而非 FLASH。并且程序保证在唤醒执行程序后不会在 3us 内访问 FLASH。
11:10	MRRDY_TIME	RW	2’b00	Stop 期间 VDD 电压为 LP-VR，唤醒时从 LP-VR 切换至稳定 Main-VR 的时间控制。 2’b00: 2us 2’b01: 3us 2’b10: 4us 2’b11: 5us
9	VOS	RW	0	Voltage scaling range selection 0：进入 stop 模式后, VDD=1.2V 1：进入 stop 模式后, VDD=1.0V
8:5	Reserved	-	-	Reserved
4	BIAS_CR_SEL	RW	0	用于选择 MR 偏置电流来自 BIAS_CR 寄存器的配置，还是来自 information memory 的 Factory config. bytes 区的加载 0：选择来自 Factory config. bytes 区的加载 1：选择来自 BIAS_CR 寄存器
3:0	BIAS_CR	RW	4’b0000	MR 偏置电流配置。 4’b0000: …..

6.6.2. PWR 寄存器映像

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	PWR_CR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HSION_CTRL	SRAM_RETV[2:0]			Res.	LPR	FLS_SLP-TIME[1:0]		MRRD_TIME[1:0]		VOS	Res.	Res.	Res.	Res.	BIAS_CR_SEL	BIAS_CR[3:0]			
	Reset value													0	1	1	1		0	0	0	0	0	0	0				0	0	0	0	0

7. 复位

芯片内设计两种复位，分别是：电源复位和系统复位。

7.1. 复位源

7.1.1. 电源复位

电源复位把所有寄存器都复位掉，在以下几种情况下产生：

- 上下电复位（POR/PDR）
- 欠压复位（BOR）

7.1.2. 系统复位

系统复位把大部分寄存器置成复位值，一些特殊寄存器，如复位标识位寄存器，不会被系统复位。

当产生以下事件时，产生系统复位：

- NRST pin 的复位
- 独立看门狗复位(IWDG)
- SYSRESETREQ 软件复位
- option byte load 复位（OBL）
- 电源复位（POR/PDR、BOR）

通过检查 RCC_CSR 寄存器的复位标识位，可以识别复位源。

7.1.3. NRST 管脚 (external reset)

通过 option byte(NRST_MODE 位)的装载，NRST pin 可以被配置成下述模式（具体配置参见 option byte 描述）：

- Reset input

在该模式下，在 NRST pin 上任何有效的复位信号被传递到内部逻辑，但是芯片内部产生的复位在 NRST pin 上不输出。

在该配置模式下，GPIO 的 PF2 功能无效。

对 NRST pin 有滤毛刺处理，设计保证 NRST 最小要满足 40us 宽度，少于该宽度的信号将被滤除。

- GPIO

在该模式下，该 PIN 可以用作标准的 GPIO，即 PF2。Pin 上的 reset 功能无效。芯片复位只会由芯片内部产生，并且不能传递到 pin 上。

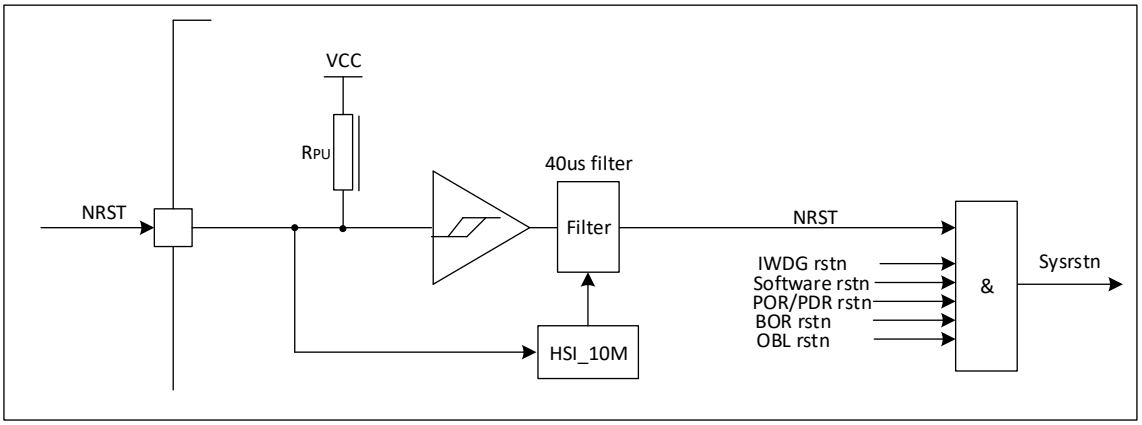


图 7-1 复位电路

7.1.4. 看门狗复位

参见 Independent watchdog。

7.1.5. 软件复位

通过置位 ARM M0+的中断和复位控制寄存器的 SYSRESETREQ 位，可实现软件复位。

7.1.6. Option byte loader 复位

软件通过配置 FLASH_CR.OBL_LAUNCH=1,产生 option byte load 复位, 从而启动 option byte 再次 load。

8. 时钟

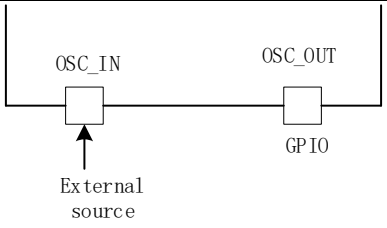
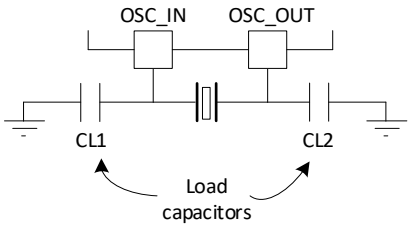
8.1. 时钟源

8.1.1. 外部高速时钟 HSE

外部高速时钟（HSE）来自两个来源：

- 通过外接晶体，配合内部起振电路，产生 4-24MHz 的时钟信号
- 直接从外部输入高速时钟源

表 8-1 HSE 时钟来源

时钟源	硬件配置
外部时钟	
外接晶体	

外接晶体

4-24MHz 的晶体具有非常高的精度。RCC_CR 的 HSERDY 标志位显示了 HSE 是否稳定。HSE 可以通过 HSEON 位进行开或者关。

外接时钟源（HSE bypass）

该模式下，外部时钟源直接提供给芯片。软件通过 RCC_CR 的 HSEBYP 和 HSEON 位选择该模式。外部时钟源将会通过 PF0 输入到芯片内部，PF1 作为 GPIO 使用。

8.1.2. 内部高速时钟 HSI

内部高速时钟，作为芯片系统时钟最重要的来源。HSI 时钟源的中心频率设计成 24MHz。

8.1.3. 内部低速时钟 LSI

内部低速时钟，作为 IWDG 和 LPTIM 的时钟，以及作为芯片低速运行时的系统时钟。该时钟中心频率设计在 32.768kHz。

8.2. 时钟树

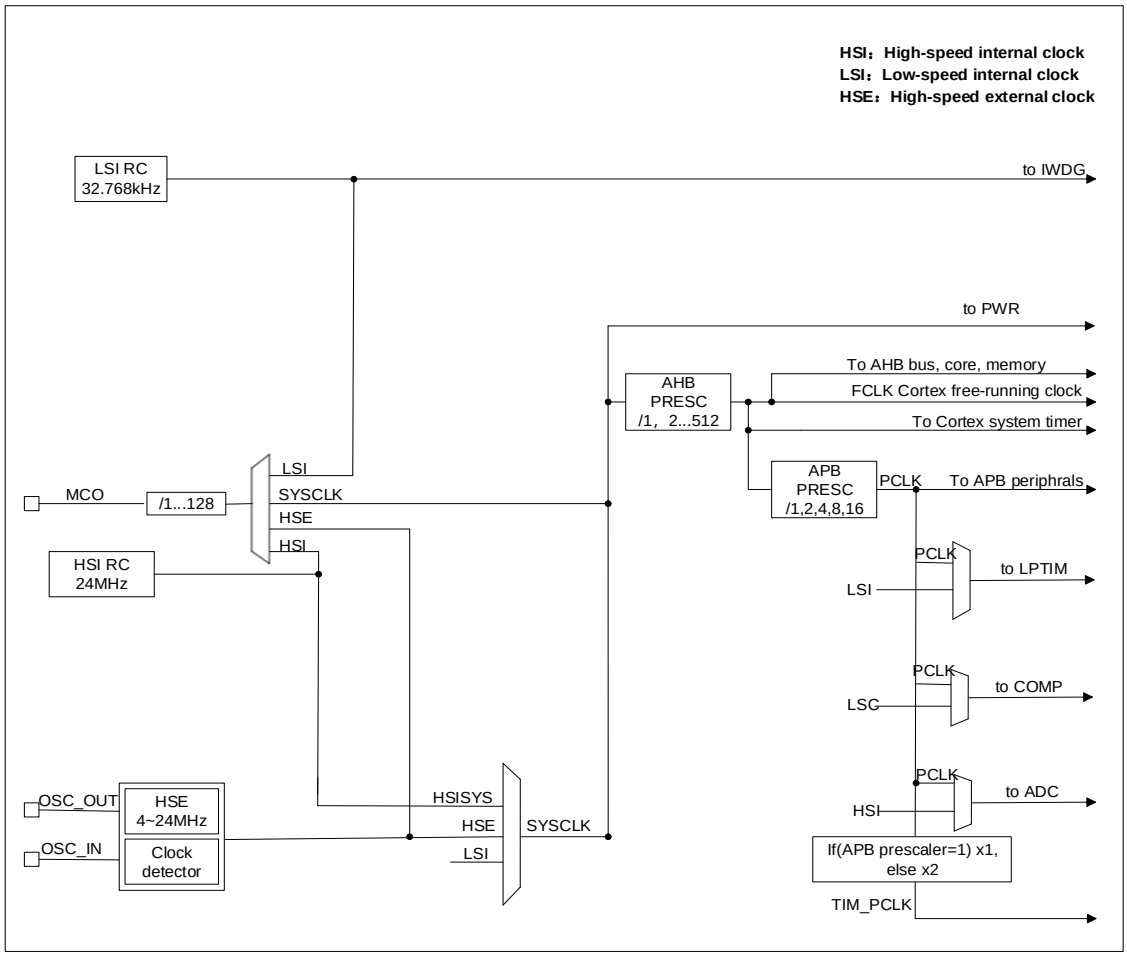


图 8-1 系统时钟结构图

8.3. 时钟安全系统 (CSS)

时钟安全系统可以被软件激活。在这种情况下，HSE 的启动延迟后，时钟检测功能被打开。当这个 HSE 被关闭后，时钟检测功能被关闭。

如果在 HSE 上发现时钟失效，HSE 会被自动关闭，时钟失效事件被送给 TIM1（高级 timer）的刹车输入端，并产生中断通知软件该失效（Clock Security System Interrupt CSSI），进而允许 MCU 进行拯救操作。CSSI 被链接到 Cortex-M0+的 NMI（Non-maskable interrupt）exception 向量。

Note: 一旦 CSS 被使能，并且如果 HSE 时钟失效，就会产生 CSS 中断，并自动产生一个 NMI。该 NMI 将不断执行，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_CICR）里的 CSSC 位来清除 CSS 中断。

如果 HSE 被直接或者间接的用作系统时钟，时钟失效将导致系统时钟自动切换到 HSI，同时关闭 HSE。

8.4. 输出时钟能力

为了方便板级应用，节省 BOM 成本，以及调试等的需求，需要芯片提供时钟输出功能。即把下表的 MCO 信号（并分频）通过 GPIO 的复用功能实现时钟输出功能。

表 8-2 输出时钟选择

时钟源	MCO 可输出的时钟源
HSI	√

时钟源	MCO 可输出的时钟源
SYSCLK	√
HSE	√
LSI	√

注意：当对 MCO 时钟源进行切换，以及选择 GPIO AF 功能为 MCO 的起始阶段，MCO 可能会产生毛刺，需要避开该段时间。

8.5. 复位/时钟寄存器

该模块的寄存器可以用 word(32bit)、half-word（16bit）和 byte（8bit）访问。

8.5.1. 时钟控制寄存器（RCC_CR）

Address offset:0x00

Reset value:0x0000 0100

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	CSS ON	HSE BYP	HSE RDY	HSE ON
						R	RW					RS	RW		
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res	Res	Res			HSI RDY	Res	HSION	Res	Res	Res	Res	Res	Res	Res	Res
					R	RW	RW								

Bit	Name	R/W	Reset Value	Function
31:20	Reserved	-	-	Reserved
19	HSE_CSSON	RS	0	时钟安全系统使能。 软件置位使能时钟安全系统。当该位被置位，如果 HSE ready 时，硬件会进行时钟检测，当发现时钟失效后，硬件 disable 掉时钟检测。 该位只能被置位，清零只能靠复位。 0：时钟安全系统 OFF（时钟检测 OFF） 1：时钟安全系统 ON（如果 HSE 稳定了，时钟检测 ON，否则 OFF）
18	HSEBYP	RW	0	旁路 HSE 外接 Crystal，选择管脚输入时钟。 软件置位和清零，bypass 掉外接 crystal 的情况，连接外部管脚输入时钟。外部时钟必须用 HSEON 使能。HSEBYP 位仅当 HSE 外接 crystal 不使能时才被置位。 0：HSE 外接 crystal 不被 bypass 掉 1：HSE 外接 crystal 被 bypass 掉，外接管脚输入时钟
17	HSERDY	R	0	HSE 时钟 ready 标志位 硬件置位，表明 HSE 稳定了。 0：HSE 没有 ready 1：HSE ready 了 注：当 HSEON 清零后，HSERDY 立即清零
16	HSEON	RW	0	HSE 时钟使能 软件可置位和清零。进入 stop 模式，硬件清零该位。如果 HSE 被直接或者间接用作系统时钟，则该位不能被复位。 0：HSE OFF 1：HSE ON
15:11	Reserved	-	-	Reserved
10	HSIRDY	R	0	HSI 时钟 ready 标志。 硬件置位表明 HSI OSC 稳定。该位只有当 HSION=1 时才有效。 0：HSI OSC not ready； 1：HSI OSC ready； 当 HSION 清零后，HSIRDY 立即拉低。
9	Reserved	-	-	Reserved
8	HSION	RW	1	HSI 时钟使能位。软件可以置位和清零该位。 当进入 stop 模式时，硬件清零该位，停止 HSI。

Bit	Name	R/W	Reset Value	Function
				当 HSI 被直接或者间接用作系统时钟（也当退出 stop 模式，或者 HSE 作为系统时钟，并产生失效时）。 0：HSI OFF 1：HSI ON
7:0	Reserved	-	-	Reserved

8.5.2. 内部时钟源校准寄存器（RCC_ICSCR）

Address offset:0x04

Reset value:0x00FF 10FF, reset by POR/BOR

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

Bit	Name	R/W	Reset Value	Function
31:28	Reserved		-	保留
27:26	LSI_STARTUP	RW	2’b00	内部低速时钟 LSI 稳定时间选择： 11：256 个 LSI 时钟周期 10：64 个 LSI 时钟周期 01：16 个 LSI 时钟周期 00：4 个 LSI 时钟周期
25	Reserved			
24:16	LSI_TRIM	RW	0x0FF	内部低速时钟频率校准。 上电后芯片硬件会把出厂信息（存放在 0x1FFF 0FA4）写入该寄存器中，使 LSI 可以输出精准的 32.768KHz 频率。 软件通过对该寄存器值进行改写，每增（减）1，使 LSI 的输出频率增（减）约 0.2%。
15:13	HSI_FS	RW	3’b000	HSI 频率选择： 001: 8MHz 100: 24MHz 其它:保留 上电后，默认配置为 8MHz。
12:0	HSI_TRIM	RW	0x10FF	时钟频率校准值。 软件通过读出存放在 information 区相应地址的数据，写入该寄存器，实现 HSI 特定输出频率下的校准。 保存在 Flash 的如下地址内： 24MHz 校准值存放地址：0x1FFF 0F10 8MHz 校准值存放地址：0x1FFF 0F04 通过向该寄存器写入校准值后，也可以该值为中心值，修改该寄存器数值，每增（减）1，则 HSI 的输出频率增（减）约 0.1%。

8.5.3. 时钟配置寄存器（RCC_CFGR）

Address offset:0x08

Reset value:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	MCOPRE[2:0]			Res	MCOSEL[2:0]			Res	Res	Res	Res	Res	Res	Res	Res
	RW				RW										
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res	PPRE[2:0]			HPRE[3:0]				Res	Res	SWS[2:0]			SW[2:0]		
	RW			RW						R			RW		

Bit	Name	R/W	Reset Value	Function
31	Reserved	-	-	Reserved

Bit	Name	R/W	Reset Value	Function
30:28	MCOPRE[2:0]	RW	0	MCO（microcontroller clock output）分频系数。软件控制这些位，设置 MCO 输出的分频系数： 000：1 001：2 010：4 011：8 100：16 101：32 110：64 111：128 推荐在 MCO 输出使能前，设置这些位。
27	Reserved	-	-	Reserved
26:24	MCOSEL[2:0]	RW	0	MCO 选择 000：没有时钟，MCO 输出不使能 001: SYSCLK 010: 保留 011: HSI 100: HSE 101: Reserved 110: LSI 111: Reserved 注：在时钟启动或者切换阶段可能会出现输出时钟不完整的情况。
23:15	Reserved	-	-	Reserved
14:12	PPRE[2:0]	RW	0	该位由软件控制。为了产生 PCLK 时钟，它设置 HCLK 的分频系数如下： 0xx：1 100：2 101：4 110：8 111：16
11:8	HPRE[3:0]	RW	0	AHB 时钟分频系数。 软件控制该位。为了产生 HCLK 时钟，它设置 SYSCLK 的分频系数如下： 0xxx: 1 1000: 2 1001: 4 1010: 8 1011: 16 1100: 64 1101: 128 1110: 256 1111: 512 为了保证系统正常工作，需要根据 VR 电源情况配置合适频率。 注：建议逐级切换分频系数。
7:6	Reserved	-	-	Reserved
5:3	SWS[2:0]	R	0	系统时钟切换状态位 这些位由硬件控制，表明当前哪个时钟源被用作系统时钟： 000: HSISYS 001: HSE 010: Reserved 011: LSI Others: Reserved
2:0	SW[2:0]	RW	0	系统时钟源选择位。 这些位由软件和硬件控制，用来选择系统时钟： 000: HSISYS 001: HSE 010: Reserved 011: LSI Others: Reserved 硬件配置为 HSISYS 的情况包括： 1）系统从 stop 模式退出

Bit	Name	R/W	Reset Value	Function
				2）软件配置 001(HSE)，出现 HSE failure（HSE 为系统时钟源）

8.5.4. 外部时钟源控制寄存器（RCC_ECSCR）

Address offset:0x10

Reset value: 0x0001_0000

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

Bit	Name	R/W	Reset Value	Function
31:18	Reserved	RES	-	保留
3:2	HSE_FREQ	RW	0x0	HSE 晶振工作频率。 00：HSE 关闭 01：4MHz~8MHz 10：8MHz ~16MHz 11：16MHz~24MHz
1:0	Reserved			

8.5.5. 时钟中断使能寄存器（RCC_CIER）

Address offset:0x18

Reset value:0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31:5	Reserved	-	-	Reserved
4	HSERDYIE	RW	0	HSE 时钟 ready 中断使能。 0：禁止 1：使能
3	HSIRDYIE	RW	0	HSI 时钟 ready 中断使能。 0：禁止 1：使能
2:1	Reserved	-	-	Reserved
0	LSIRDYIE	RW	0	LSI 时钟 ready 中断使能。 0：禁止 1：使能

8.5.6. 时钟中断标志寄存器（RCC_CIFR）

Address offset:0x1C

Reset value:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	Res	Res	Res	Res	CSSF	Res	Res	Res	HSE RDYF	HSI RDYF	Res	Res	LSI RDYF
							R				R	R			R

Bit	Name	R/W	Reset Value	Function
31:10	Reserved	-	-	Reserved
8	CSSF	R	0	HSE 时钟安全系统中断标识位。 当硬件检测 HSE OSC 时钟失败时置位该寄存器。 0：HSE 时钟检测失败中断未产生； 1：HSE 时钟检测失败中断产生； 写 CSSC 寄存器 1 清零该位。
7:5	Reserved	-	-	Reserved
4	HSERDYF	R	0	HSE ready 中断标识位 当 HSE 稳定并且 HSERDYIE 使能，该位由硬件置位。软件通过置位 HSERDYC 位，清零该位。 0：无由 HSE 引起的时钟 ready 中断 1：有由 HSE 引起的时钟 ready 中断
3	HSIRDYC	W	0	HSI ready 标志清零。 0：没影响。 1：清除 HSIRDYF 位。
2:1	Reserved	-	-	Reserved
0	LSIRDYF	R	0	LSI ready 中断标识位 当 LSI 稳定并且 LSIRDYIE 使能，该位由硬件置位。软件通过置位 LSIRDYC 位，清零该位。 0：无由 LSI 引起的时钟 ready 中断 1：有由 LSI 引起的时钟 ready 中断

8.5.7. 时钟中断清除寄存器（RCC_CICR）

Address offset:0x20

Reset value:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res	Res	Res	Res	Res	Res	Res	CSSC	Res	Res	Res	HSE RDYC	HSI RDYC	Res	Res	LSI RDYC
							W				W	W			W

Bit	Name	R/W	Reset Value	Function
31:9	Reserved	-	-	Reserved
8	CSSC	W	0	时钟安全中断清零位。 0：没影响。 1：清除 CSSF 标志位。
7:5	Reserved	-	-	Reserved
4	HSERDYC	W	0	HSE ready 标志清零。 0：没影响。 1：清除 HSERDYF 位。
3	HSIRDYC	W	0	HSI ready 标志清零。 0：没影响。 1：清除 HSIRDYF 位。
2:1	Reserved	-	-	Reserved
0	LSIRDYC	W	0	LSI ready 标志清零。 0：没影响。 1：清除 LSIRDYF 位。

8.5.8. I/O 接口复位寄存器（RCC_IOPRSTR）

Address offset:0x24

Reset value:0x0000 0000

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

										RW				RW	RW

Bit	Name	R/W	Reset Value	Function
31:6	Reserved	-	-	Reserved
5	GPIOFRST	RW	0	I/O PortF 复位。 0：no effect； 1：PortF I/O 复位
4:2	Reserved	-	-	Reserved
1	GPIOBRST	RW	0	I/O PortB 复位。 0：no effect； 1：PortB I/O 复位
0	GPIOARST	RW	0	I/O PortA 复位。 0：no effect； 1：PortA I/O 复位

8.5.9. AHB 外设复位寄存器（RCC_AHBRSTR）

Address offset:0x28

Reset value:0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31:13	Reserved	-	-	Reserved
12	CRCRST	RW	0	CRC 模块复位。 0：no effect； 1：CRC 模块复位；
11:0	Reserved	-	-	Reserved

8.5.10. APB 外设复位寄存器 1（RCC_APBRSTR1）

Address offset:0x2C

Reset value:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LPTIM RST	Res	Res	PWR RST	DBG RST	Res	Res	Res	Res	Res	I2C RST	Res	Res	Res	Res	Res
RW			RW	RW						RW					
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res

Bit	Name	R/W	Reset Value	Function
31	LPTIMRST	RW	0	LP Timer 模块复位。 0：no effect； 1：该模块复位；
30:29	Reserved	-	-	Reserved
28	PWRRST	RW	0	Power 接口模块复位。 0：no effect； 1：该模块复位；
27	DBGRST	RW	0	MCU Debug 模块复位。 0：no effect； 1：该模块复位；
26:22	Reserved	-	-	Reserved
21	I2CRST	RW	0	I2C1 模块复位。 0：no effect； 1：该模块复位；

Bit	Name	R/W	Reset Value	Function
20:0	Reserved	-	-	Reserved

8.5.11. APB 外设复位寄存器 2（RCC_APBRSTR2）

Address offset:0x30

Reset value:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	COMP2 RST	COMP1 RST	ADC RST	Res	Res	TIM16 RST	Res
									RW	RW	RW			RW	
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res	USART1 RST	Res	SPI1 RST	TIM1 RST	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	SYSCFG RST
	RW		RW	RW											RW

Bit	Name	R/W	Reset Value	Function
31:23	Reserved	-	-	Reserved
22	COMP2RST	RW	0	COMP2 模块复位。 0：no effect； 1：该模块复位；
21	COMP1RST	RW	0	COMP1 模块复位。 0：no effect； 1：该模块复位；
20	ADCRST	RW	0	ADC 模块复位。 0：no effect； 1：该模块复位；
19:18	Reserved	-	-	Reserved
17	TIM16RST	RW	0	TIM16 模块复位。 0：no effect； 1：该模块复位；
16:15	Reserved	-	-	Reserved
14	USART1RST	RW	0	USART1 模块复位。 0：no effect； 1：该模块复位；
13	Reserved	-	-	Reserved
12	SPI1RST	RW	0	SPI1 模块复位。 0：no effect； 1：该模块复位；
11	TIM1RST	RW	0	TIM1 模块复位。 0：no effect； 1：该模块复位；
10:1	Reserved	-	-	Reserved
0	SYSCFGRST	RWs	0	SYSCFG 模块复位。 0：no effect； 1：该模块复位；

8.5.12. I/O 接口时钟使能寄存器（RCC_IOPENR）

Address offset:0x34

Reset value:0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31:6	Reserved	-	-	Reserved
5	GPIOFEN	RW	0	I/O PortF 时钟使能。 0：时钟禁止； 1：时钟使能
4:2	Reserved	-	-	Reserved
1	GPIOBEN	RW	0	I/O PortB 时钟使能。 0：时钟禁止； 1：时钟使能
0	GPIOAEN	RW	0	I/O PortA 时钟使能。 0：时钟禁止； 1：时钟使能

8.5.13. AHB 外设时钟使能寄存器（RCC_AHBENR）

Address offset:0x38

Reset value:0x0000 0300

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res	Res	Res	CRC EN	Res	Res	SRA-MEN	FLASH EN	Res	Res	Res	Res	Res	Res	Res	Res
			RW				RW								

Bit	Name	R/W	Reset Value	Function
31:13	Reserved	-	-	Reserved
12	CRCEN	RW	0	CRC 模块时钟使能。 0：禁止 1：使能
11:10	Reserved	-	-	Reserved
9	SRAMEN	RW	1	在 sleep 模式下，SRAM 的时钟使能控制 0：在 sleep 模式该模块时钟关闭 1：在 sleep 模式该模块时钟使能 注：该位仅影响 sleep 模式该模块的时钟使能，在 run 模式，该模块时钟不会关闭
8	FLASHEN	RW	1	在 sleep 模式下，FLASH 的时钟使能控制 0：在 sleep 模式该模块时钟关闭 1：在 sleep 模式该模块时钟使能 注：该位仅影响 sleep 模式该模块的时钟使能，在 run 模式，该模块时钟不会关闭
7:0	Reserved	-	-	Reserved

8.5.14. APB 外设时钟使能寄存器 1（RCC_APBENR1）

Address offset:0x3C

Reset value:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LPTIM EN	Res	Res	PWR EN	DBG EN	Res	Res	Res	Res	Res	I2C EN	Res	Res	Res	Res	Res
RW			RW	RW											
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res

Bit	Name	R/W	Reset Value	Function
31	LPTIMEN	RW	0	LP Timer1 模块时钟使能。 0：禁止 1：使能
30:29	Reserved	-	-	Reserved
28	PWREN	RW	0	低功耗控制模块时钟使能。

Bit	Name	R/W	Reset Value	Function
				0：禁止 1：使能
27	DBGEN	RW	0	Debug 模块时钟使能。 0：禁止 1：使能
26:22	Reserved	-	-	Reserved
21	I2CEN	RW	0	I2C1 模块时钟使能。 0：禁止 1：使能
20:0	Reserved	-	-	Reserved

8.5.15. APB 外设时钟使能寄存器 2（RCC_APBENR2）

Address offset:0x40

Reset value:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	COMP2EN	COMP1EN	ADCEN	Res	Res	TIM16EN	Res
									RW	RW	RW			RW	
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res	USART1EN	Res	SPI1EN	TIM1EN	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	SYSCFGEN
	RW		RW	RW											RW

Bit	Name	R/W	Reset Value	Function
31:23	Reserved	-	-	Reserved
22	COMP2EN	RW	0	COMP2 模块时钟使能。 0：禁止 1：使能
21	COMP1EN	RW	0	COMP1 模块时钟使能。 0：禁止 1：使能
20	ADCEN	RW	0	ADC 模块时钟使能。 0：禁止 1：使能
19:18	Reserved	-	-	Reserved
17	TIM16EN	RW	0	TIM16 模块时钟使能。 0：禁止 1：使能
16:15	Reserved	-	-	Reserved
14	USART1EN	RW	0	USART1 模块时钟使能。 0：禁止 1：使能
13	Reserved	-	-	Reserved
12	SPIEN	RW	0	SPI1 模块时钟使能。 0：禁止 1：使能
11	TIM1EN	RW	0	TIM1 模块时钟使能。 0：禁止 1：使能
10:1	Reserved	-	-	Reserved
0	SYSCFGEN	RW	0	SYSCFG 模块时钟使能。 0：禁止 1：使能

8.5.16. 外设独立时钟配置寄存器（RCC_CCIPR）

Address offset:0x54

Reset value:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res		Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	LPTIM1SEL[1:0]		Res	Res
												RW	RW		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res		Res		Res		COMP2SEL	COMP1SEL	Res	Res	Res	Res	Res	Res	Res	Res
					RW	RW									

Bit	Name	R/W	Reset Value	Function
31:20	Reserved	-	-	Reserved
19:18	LPTIMSEL[1:0]	RW	2'b00	LPTIM1 内部时钟源选择。 00: PCLK 01: LSI 10: Reserved 11: Reserved 注：当选择 PCLK 时，LPTIM 的预分频系数系数要配置 2 分频及以上（通过 LPTIM_CFGR.PRESC 配置）。
17:10	Reserved	-	-	Reserved
9	COMP2SEL	RW	0	COMP2 模块时钟时钟源选择。 0: PCLK 1: LSC（RCC_BDCR.LSCOSEL 选择后的时钟） 注：在使能 FLTEN 之前先配置选择 LSC 时钟。
8	COMP1SEL	RW	0	COMP1 模块时钟时钟源选择。 0: PCLK 1: LSC（RCC_BDCR.LSCOSEL 选择后的时钟） 注：在使能 COMP2_FR2.FLTEN 之前先配置该寄存器选择时钟。
7:0	Reserved	-	-	Reserved

8.5.17. RCC_BDCR

Address:0x5C

Reset value:0x0000 0000，reset by POR/BOR

当 PWR_CR1.DBP 为 1 时，才允许写该寄存器。

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

Bit	Name	R/W	Reset Value	Function
31:25	Reserved	-	-	Reserved
24	LSCOEN	RW	0	低速时钟使能。 0：禁止 1：使能
23:0	Reserved	-	-	Reserved

8.5.18. 控制/状态寄存器 (RCC_CSR)

Address offset:0x60

Reset value:0x0000 0000

复位源如下：1）[30:25]：POR 复位；2）LSION：系统复位；3）NRST_FLTIDS 不会被 system reset 复位

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	IWDGRSTF	SFTRSTF	PWRRSTF	PINRSTF	OBLRSTF	Res	RMVF	Res	Res	Res	Res	Res	Res	Res

		R	R	R	R	R		RW							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res	Res	Res	Res	Res	Res	Res	NRST_FLT-DIS	Res	Res	Res	Res	Res	Res	LSIRDY	LSION
							RW							R	RW

Bit	Name	R/W	Reset Value	Function
31:30	Reserved			
29	IWDGRSTF	R	0	IWDG 复位标志。 RMVF 置 1 会清零该位。
28	SFTRSTF	R	0	软复位标志。 RMVF 置 1 会清零该位。
27	PWRRSTF	R	0	BOR/POR/PDR 复位标志。 RMVF 置 1 会清零该位。
26	PINRSTF	R	0	外部 NRST 管脚复位标志。 RMVF 置 1 会清零该位。
25	OBLRSTF	R	0	Option byte loader 复位标志。 RMVF 置 1 会清零该位。
24	Reserved			-
23	RMVF	RW	0	需通过软件置 1 来清零[30:25]的复位标志。
8	NRST_FLT-DIS	RW	0	NRST 滤波禁止 0：使能 HSI_10M，且滤波 40us 宽度功能使能 1：滤波功能禁止，且 HSI_10M 保持关闭
7:2	Reserved	-	-	Reserved
1	LSIRDY	R	0	LSI OSC 稳定标志。 0：LSI 未稳定 1：LSI 已稳定
0	LSION	RW	0	LSI OSC 使能。 0：禁止 1：使能 软件置位，软件清零。在硬件使能 IWDG（通过 option byte），硬件会对该位进行置位。

8.5.19. RCC 寄存器地址映像

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	RCC_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HSE_CSS	HSEBYP	HSERDY	HSEON	Res.	Res.	Res.			HSIRDY	Res.	HSION	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value													0	0	0	0						0	0	1								
0x04	RCC_ICSCR	Res.	Res.	Res.	Res.	LSI_STARTUP[1:0]		Res.	LSI_TRIM[8:0]									HSI_FS[2:0]			HSI_TRIM[12:0]												
	Reset value					0	0		0	1	1	1	1	1	1	1	1	0	0	0	1	0	0	0	0	1	1	1	1	1	1	1	1
0x08	RCC_CFGR	Res.	MCO-PRE[2:0]			Res.	MCO-SEL[2:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PPRE[2:0]			HPRE[3:0]				Res.	Res.	SWS[2:0]			SW[2:0]		
	Reset value		0	0	0		0	0	0										0	0	0	0	0	0	0			0	0	0	0	0	0

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x10	RCC_ECSCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HSE_FREQ[1:0]		Res.	Res.
	Reset value																													0	0		
0x14	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0x18	RCC_CIER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HSERDYIE	HSIRDYIE	Res.	Res.	Res.	LSIRDYIE
	Reset value																											0	0				0
0x1C	RCC_CIFR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CSSF	Res.	Res.	Res.	HSERDYIF	HSIRDYIF	Res.	Res.	LSIRDYIF
	Reset value																								0				0	0			0
0x20	RCC_CICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CSSC	Res.	Res.	Res.	HSERDYIC	HSIRDYIC	Res.	Res.	LSIRDYIC
	Reset value																								0				0	0			0
0x24	RCC_IOPRSTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GPIOFRST	Res.	Res.	Res.	GPIOBRST	GPIOARST
	Reset value																											0				0	0
0x28	RCC_AHBRSTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CRCRST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																				0												
0x2C	RCC_APBRSTR1	LPTIMRST	Res.	Res.	PWRRST	DBGRST	Res.	Res.	Res.	Res.	Res.	I2CRST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0			0	0						0																					
0x30	RCC_APBRSTR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	COMP2RST	COMP1RST	ADCRST	Res.	Res.	TIM16RST	Res.	Res.	USART1RST	Res.	SPI1RST	TIM1RST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SYSCFGRST
	Reset										0	0	0			0			0		0	0											0

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	value																																
0x34	RCC_IOPENR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GPIOFEN	Res.	Res.	Res.	GPIOBEN	GPIOAEN
	Reset value																											0				0	0
0x38	RCC_AHBENR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CRCEN	Res.	Res.	SRAMEN	FLASHE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																				0			1	1								0
0x3C	RCC_APBENR1	LPTIMEN	Res.	Res.	PWREN	DBGEN	Res.	Res.	Res.	Res.	Res.	I2CEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.		Res.	Res.	Res.	Res.
	Reset value	0			0	0						0				0						0	0					0				0	
0x40	RCC_APBENR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	COMP2EN	COMP1EN	ADCEN	Res.	Res.	TIM16EN	Res.	Res.	USART1EN	Res.	SPI1EN	TIM1EN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SYSCFGEN
	Reset value										0	0	0			0			0		0	0											0
0x44-0x50	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0x54	RCC_CCIPR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LPTIMSEL[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	COMP2SE	COMP1SE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value													0	0									0	0								
0x58	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0x5C	RCC_BDCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LSCOEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value								0																								
0x60	RCC_CSR	Res.	Res.	IWDGRSTF	SFTRSTF	PWRRSTF	PINRSTF	OBLRSTF	Res.	RMVF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NRST_FLT	Res.	Res.	Res.	Res.	Res.	LSIRDY	LSION
	Reset		0	0	0	0	0	0		0															0						0	0

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

9. 通用 I/O（GPIO）

9.1. 通用 IO 简介

每个 GPIO 端口有：

4 个 32 位配置寄存器(GPIOx_MODER,GPIOx_OTYPER,GPIOx_OSPEEDR, GPIOx_PUPDR)

2 个 32 位数据寄存器 (GPIOx_IDR 和 GPIOx_ODR)

1 个 32 位置位/复位寄存器(GPIOx_BSRR)

1 个 32 位锁定寄存器(GPIOx_LCKR)

2 个复用功能选择寄存器(GPIOx_AFRH 和 GPIOx_AFRL)。

9.2. 通用 IO 功能描述

- 输出状态：push-pull 或者 open drain + 上拉/下拉
- 数据寄存器(GPIOx_ODR)或者外设（复用功能输出）数据输出
- 每个 I/O 可进行速度选择
- 输入状态：floating，pull-up/down，analog
- 数据输入送给输入数据寄存器(GPIOx_IDR)或者外设（复用功能输入）
- 位置位/复位寄存器（GPIOx_BSRR），允许对 GPIOx_ODR 的位写访问
- 锁定机制 (GPIOx_LCKR)会冻结 I/O 口配置功能
- 模拟功能
- 复用功能选择寄存器（每个 IO 口最多 16 种复用功能）
- 单周期内快速翻转的能力
- 高度灵活的 I/O 多路选择功能，使得 I/O 口作为 GPIO，或者作为各种外设接口功能

9.3. 通用 IO 功能描述

每个 GPIO 的每个位，可以通过软件编程，进行几种模式的配置：

- 输入浮空
- 输入上拉
- 输入下拉
- 模拟输入
- 开漏输出，带上拉或者下拉
- 推挽输出，带上拉或者下拉
- 带上拉或者下拉的复用功能的推挽
- 带上拉或者下拉的复用的开漏

每个 I/O 口可以自由编程，然而 I/O 端口寄存器必须按 32 位字、半字或者字节被访问。GPIOx_BSRR 和 GPIOx_BRR 寄存器允许对任何 GPIOx_ODR 寄存器的读/更改的独立访问。这样，在读和更改访问之间产生 IRQ 时不会发生危险。

下图给出了一个 I/O 端口（1bit）的基本结构

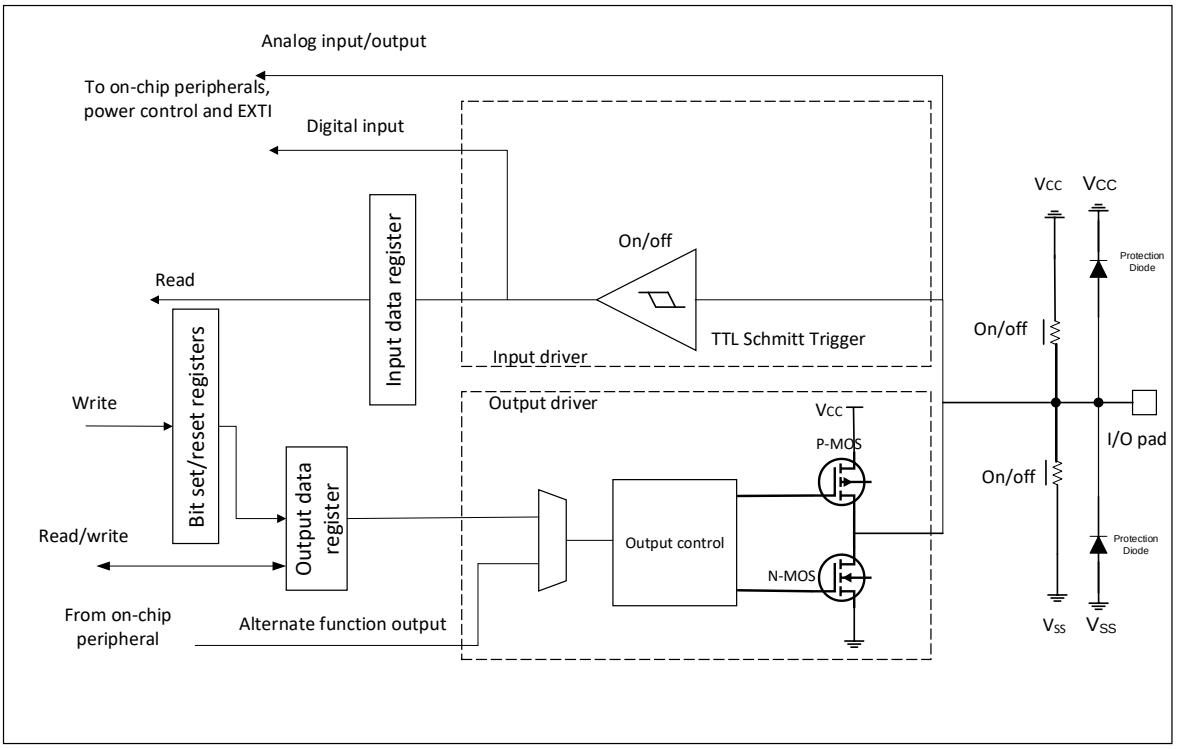


图 9-1 IO 端口位的基本结构

9.3.1. 通用 I/O(GPIO)

复位期间和复位后，复用功能未被激活，大多数 IO 被配置为模拟模式。

Debug 引脚默认被置于复用功能上拉或下拉模式：

—PA14-SWCLK：置于下拉模式

—PA13-SWDIO：置于上拉模式

Boot 引脚默认置于输入模式，下拉模式

—PF4-Boot：置于下拉模式

当管脚被配置为输出时，写入到输出数据寄存器（GPIOx_ODR）的值会输出到 I/O 上。有可能会使用 push-pull 或者开漏模式的输出(低电平是输出的，高电平是 HI-Z)。

输入数据寄存器（GPIOx_IDR）在每个 AHB 时钟会获取 I/O 脚上的电平。

所有的 GPIO 引脚都有内部的弱上拉和弱下拉电阻，可以通过 GPIOx_PUPDR 寄存器使能或者不使能该功能。

9.3.2. I/O 管脚复用功能多路选择和映射

设备 I/O 口通过多路选择器连着板级的外设/模块，一个外设每次可以通过复用功能连着一个 IO 口。这样可以避免同一个 IO 口上的可用外设不会出现冲突。

每个 I/O 口上的多路选择器多达 16 种复用功能输入（AF0 to AF7），可通过寄存器 GPIOx_AFRL (for pin 0 to 7) 和 GPIOx_AFRH (for pin 8 to 15)来配置。

- 刚复位后，多路选择器默认为 AF0。I/O 口的复用功能模式通过寄存器 GPIOx_MODER 配置
- 每个脚的复用功能分布在对应的数据手上有说明参见 2.3 节

除了这种灵活的多路选择器架构，每个外设还有复用功能可以分布在不同的 I/O 口上，以便在更小的封装上使用到的外设数量最优化。

用户按照如下说明去配置 IO：

- 调试功能：每次复位后，这些调试功能脚就是默认为调试器立即可用的复用功能脚
- GPIO：在 GPIOx_MODER 将对应 I/O 口配置为输出、输入或者模拟模式
- 外设复用功能：
  - 寄存器 GPIOx_AFRL 或者 GPIOx_AFRH 配置对应的 I/O 为复用功能 x(x=0…15)
  - 寄存器 GPIOx_OTYPER, GPIOx_PUPDR 和 GPIOX_OSPEEDER 分别配置类型，上拉/下拉以及输出速度
  - 寄存器 GPIOx_MODER 是配置对应 I/O 为复用功能
- 额外功能

  -无论 IO 口配置成任何模式，ADC 和 COMP 功能均在 ADC 和 COMP 模块的寄存器中使能。当 IO 口用做 ADC 或者 COMP 使用时，推荐通过寄存器 GPIOx_MODER 将该口配置为模拟模式

  -对于晶振额外功能，在相应的 PWR and RCC 模块寄存器里配置各自功能。这些配置比标准的 GPIO 配置具有更高优先级。

9.3.3. I/O 控制寄存器

每个 GPIO 口有四个 32 位内存映射控制寄存器(GPIOx_MODER, GPIOx_OTYPER, GPIOx_OSPEEDR 和 GPIOx_PUPDR)，可以配置多达 16 个 I/O 口。寄存器 GPIOx_MODER 用来选择 I/O 模式（输入、输出、复用、模拟）。寄存器 GPIOx_OTYPER 和 GPIOx_OSPEEDR 用来选择输出类型（推挽或开漏）和速度。寄存器 GPIOx_PUPDR 用来选择上拉/下拉不分 I/O 的方向。

9.3.4. I/O 数据寄存器

每个 GPIO 有 2 个 16 位内存映射的数据寄存器：输入和输出数据寄存器（GPIOx_IDR 和 GPIOx_ODR）。寄存器 GPIOx_ODR 保存了要输出的数据，可读可写。输入数据寄存器（GPIOx_IDR）用来保存 I/O 口上的电平状态，只读的。

9.3.5. I/O 数据按位处理

置位/复位寄存器(GPIOx_BSRR)是一个 32 位寄存器，可以将输出数据寄存器(GPIOx_ODR)的单独每位置位和复位。置位/复位寄存器位数是输出寄存器（GPIOx_ODR）的两倍。

GPIOx_ODR 的每一位对应 GPIOx_BSRR 的两个控制位：BS(i) and BR(i)。位 BS(i)置 1 可将 GPIOx_ODR 对应位置 1，位 BR(i)置 1 可将 GPIOx_ODR 对应位清 0。

寄存器 GPIOx_BSRR 任意位写 0 并不影响寄存器 GPIOx_ODR 对应的位。如果 GPIOx_BSRR 对某一位同时清 0 和置 1 操作，置 1 操作具有优先权。

使用寄存器 GPIOx_BSRR 改变寄存器 GPIOx_ODR 的对应位只有一次性的作用，并不会锁定寄存器 GPIOx_ODR 的位。寄存器 GPIOx_ODR 也可以直接访问。寄存器 GPIOx_BSRR 只是提供一种原子位操作处理方式。

当软件编程操作 GPIOx_ODR 的位时没必要关闭中断：在一个 AHB 写访问过程中有可能修改了一个或者多个位。

9.3.6. GPIO 锁定机制

寄存器 GPIOx_LCKR 通过一系列特殊写时序可以冻结 IO 的控制寄存器，包括 GPIOx_MODER,GPIOx_OTYPER,GPIOx_OSPEEDR,GPIOx_PUPDR,GPIOx_AFRL 和 GPIOx_AFRH。

一个特殊写/读时序可以操作寄存器 GPIOx_LCKR。当该寄存器的 Bit16 写入正确的时序，LCKR[15:0]写入值就可以锁定 I/O（在写入时序过程中，LCKR[15:0]写入值保持不变）。当在一个端口位上执行了锁定(LOCK)

程序，在下一次 MCU 或者外设复位之前，将不能再更改端口位的配置。GPIOx_LCKR 的每个位冻结控制寄存器（GPIOx_MODER、GPIOx_OTYPER、 GPIOx_OSPEEDR、GPIOx_PUPDR、GPIOx_AFRL and GPIOx_AFRH）对应的位。

LOCK 时序只能用字（32 位长）访问 GPIOx_LCKR 寄存器，因为 GPIOx_LCKR 位 16 设置的同时也会设置[15:0] 位。

9.3.7. I/O 复用功能输入/输出模式配置

每个 I/O 有两个寄存器可以用来配置复用功能输入/输出模式。用户根据应用需求将复用功能复用到 IO 口上。

使用寄存器 GPIOx_AFRL 和 GPIOx_AFRH 可以在每一个 GPIO 口多路选择许多可能的外设功能，因此应用让每个 I/O 选择其中一种功能。AF 选择信号对于复用功能输入和复用功能输出都是相同的，对于给定 I/O 的复用功能输入/输出可以选择单独的通道。

9.3.8. 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，端口必须禁止配置成模拟模式或者晶振管脚，并且输入触发需要使能。

9.3.9. I/O 输入配置

当 I/O 口配置为输入：

- 输出缓冲器不使能
- 施密特触发器输入使能
- 根据寄存器 GPIOx_PUPDR 配置可使能/不使能上下拉电阻
- 出现在 I/O 脚上的数据在每个 AHB 时钟被采样到输入数据寄存器
- 对输入数据寄存器的读访问可得到 I/O 状态

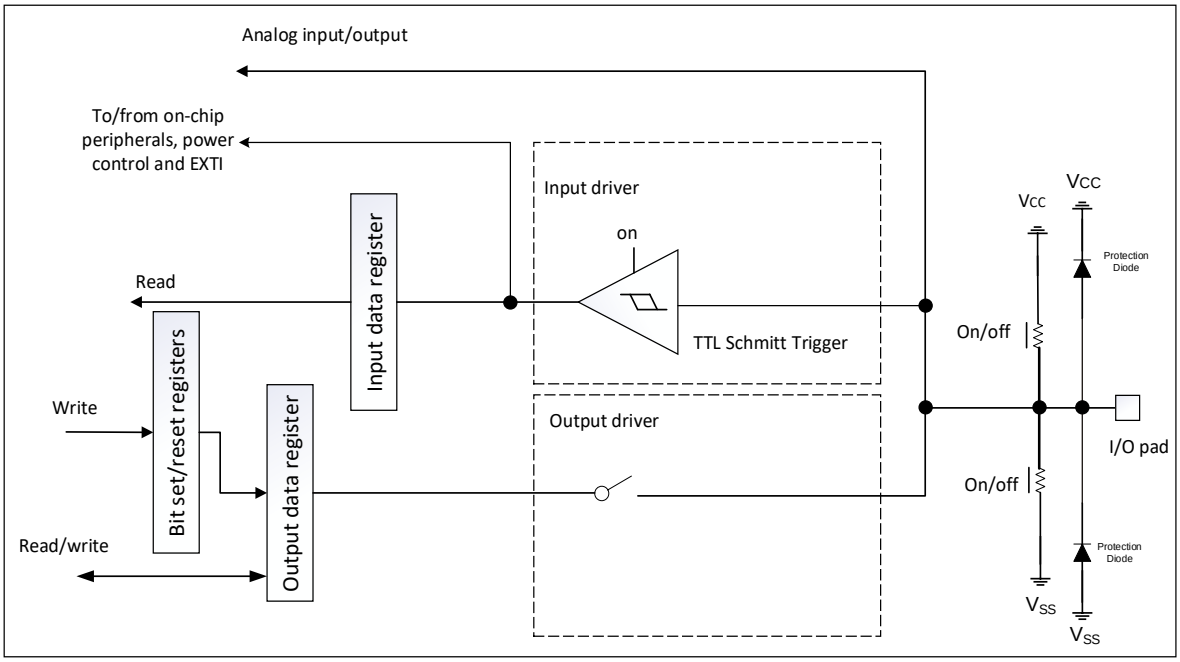


图 9-2 输入浮空/上拉/下拉配置

9.3.10. I/O 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(PMOS 从不被激活)。

  ─ 推挽模式：输出寄存器上的’0’激活 N-MOS，而输出寄存器上的’1’将激活 P-MOS。
- 施密特触发输入被激活
- 根据寄存器 GPIOx_PUPDR 配置可使能/不使能上下拉电阻
- 出现在 I/O 脚上的数据在每个 AHB 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
- 对输出数据寄存器的读访问得到后一次写的值

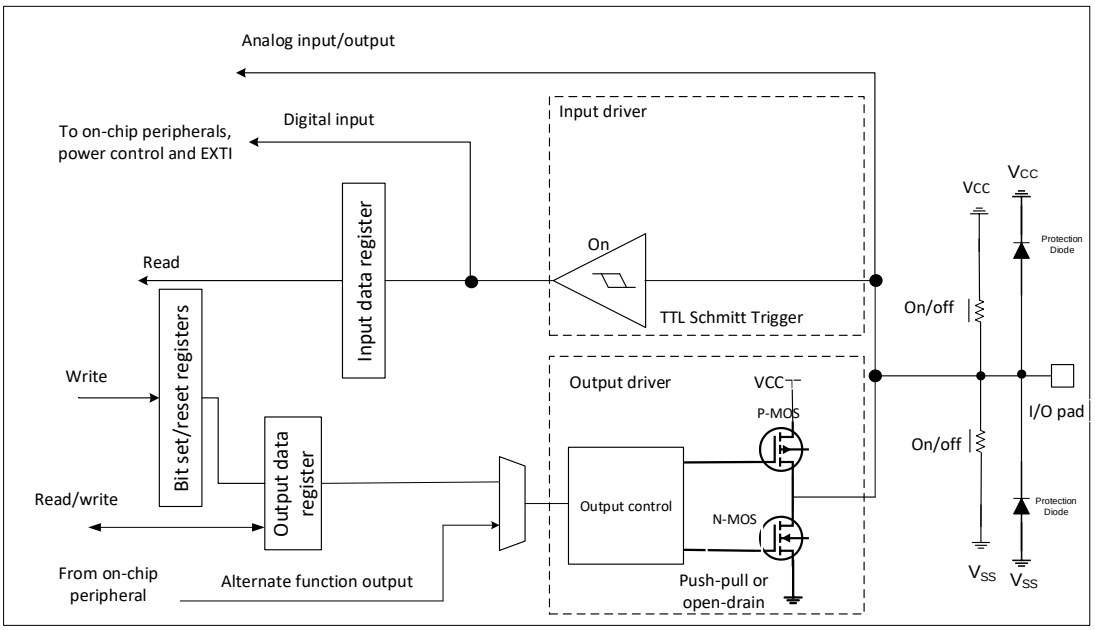


图 9-3 输出配置

9.3.11. 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
- 根据寄存器 GPIOx_PUPDR 配置可使能/不使能上下拉电阻
- 在每个 AHB 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
- 读输入数据寄存器时可得到 I/O 口状态

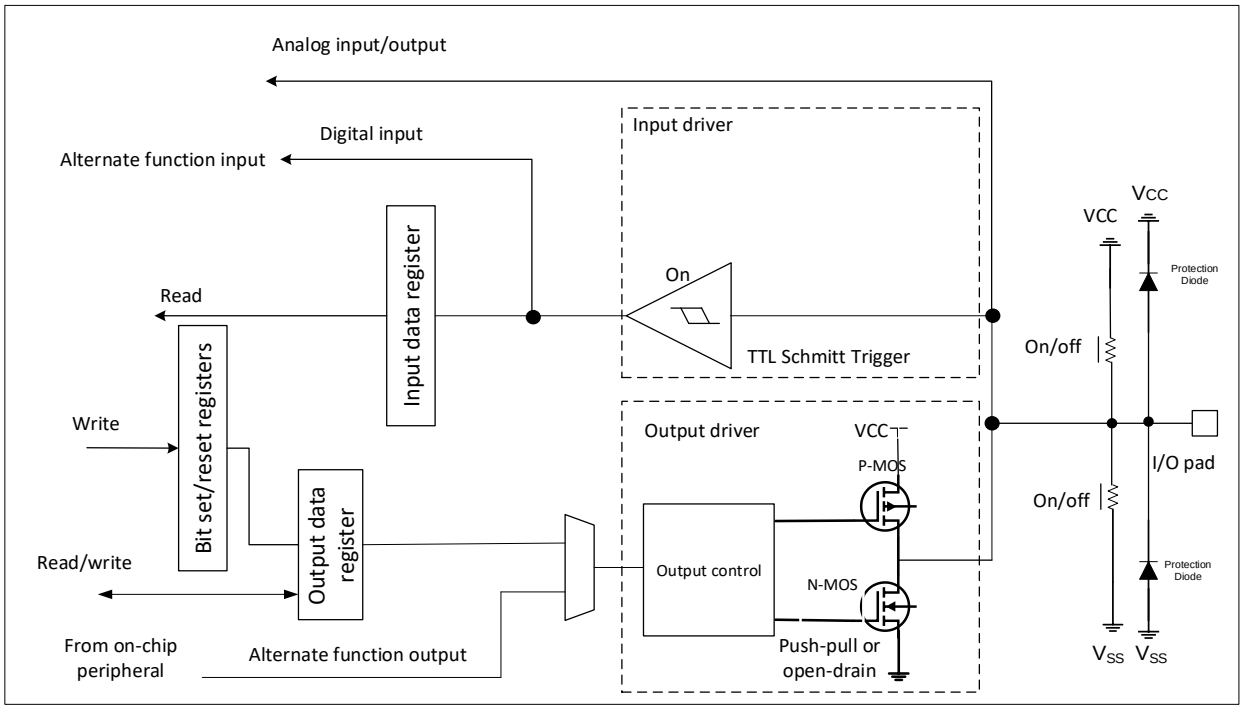


图 9-4 复用功能配置

9.3.12. 模拟配置

当 I/O 端口被配置为模拟配置时：

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

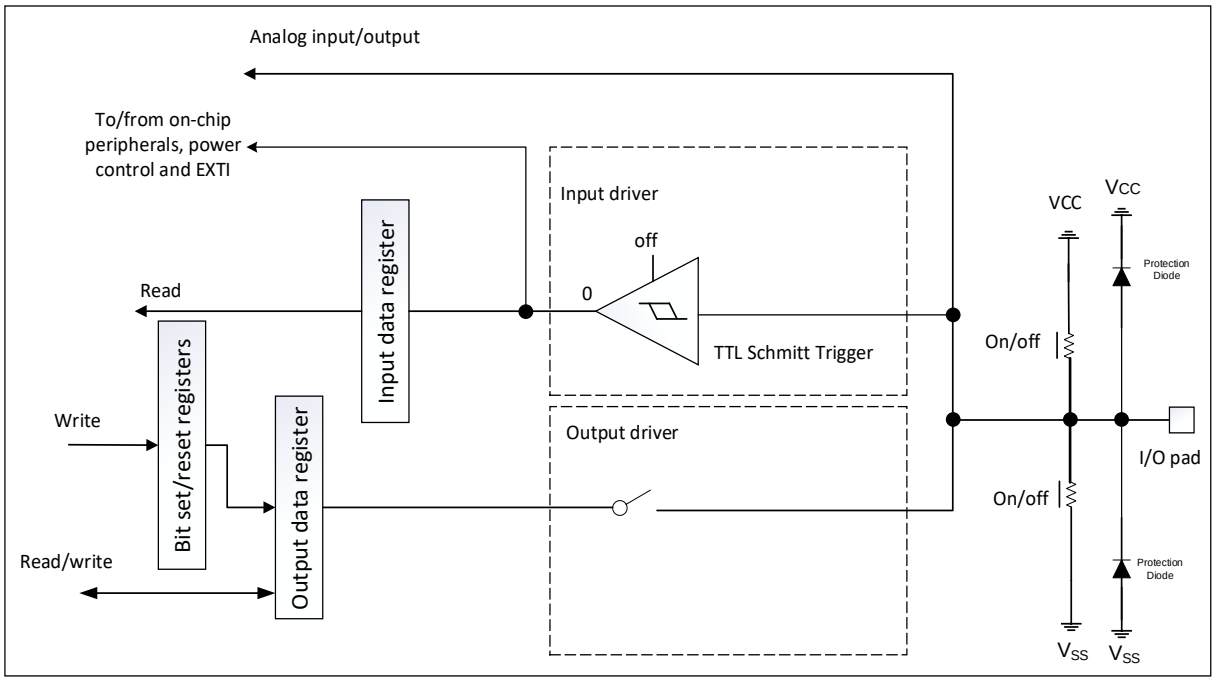


图 9-5 高阻抗模拟配置

9.3.13. 使用 HSE 管脚作为 GPIO

当 HSE 功能被关闭（复位后的默认），相应的管脚可以当作正常的 GPIO 用。

当 HSE 功能打开（RCC_CSR 寄存器中设置 HSEON），需要软件配置对应的端口为模拟端口。

当晶振配置为用户外部时钟模式，只有 OSC_IN 保留给时钟输入，而 OSC_OUT 脚仍然可以用作正常 GPIO。

9.4. GPIO 寄存器

所有 GPIO 相关寄存器都可进行 word、half word 和 byte 写操作。

9.4.1. GPIO 端口模式寄存器 (GPIOx_MODER) (x=A, B, F)

Address offset: 0x00

Reset value:

- 0xEBFF FFFF for GPIOA
- 0xFFFF FFFF for GPIOB
- 0xFFFF FCFF For GPIOF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MODE15[1:0]		MODE14[1:0]		MODE13[1:0]		MODE12[1:0]		MODE11[1:0]		MODE10[1:0]		MODE9[1:0]		MODE8[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MODE7[1:0]		MODE6[1:0]		MODE5[1:0]		MODE4[1:0]		MODE3[1:0]		MODE2[1:0]		MODE1[1:0]		MODE0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	Name	R/W	Reset Value	Function
31：0	MODEy[1:0]	RW		y = 15..0 软件通过这些位配置相应的 I/O 模式 00: 输入模式 01: 通用输出模式 10: 复用功能模式 11: 模拟模式(reset state)

9.4.2. GPIO 端口输出类型寄存器(GPIOx_OTYPER) (x = A, B, F)

Address offset: 0x04

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31:16	Reserved			
15:0	MODE[15:0]	RW		软件配置 I/O 的输出类型 0: 推挽输出 (复位状态) 1: 开漏输出

9.4.3. GPIO 端口输出速度寄存器(GPIOx_OSPEEDR) (x = A, B, F)

Address offset: 0x08

Reset value: 0x0C00 0000(for port A)

Reset value: 0x0000 0000(for other ports)

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

OSPEED15		OSPEED14		OSPEED13		OSPEED12		OSPEED11		OSPEED10		OSPEED9		OSPEED8	
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
OSPEED7		OSPEED6		OSPEED5		OSPEED4		OSPEED3		OSPEED2		OSPEED1		OSPEED0	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	Name	R/W	Reset Value	Function
31:0	OSPEEDy[1:0]	**RW**		Y = 15..0 软件配置 IO 口的输出速度 00：非常低 01：低速 10：高速 11：非常高

9.4.4. GPIO 端口上下拉寄存器(GPIOx_PUPDR) (x = A, B, F)

Address offset: 0x0C

Reset value:

0x2400 0000(for port A)

0x0000 0000(for port B)

0x0000 0200(for port F)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PUPD15[1:0]		PUPD14[1:0]		PUPD13[1:0]		PUPD12[1:0]		PUPD11[1:0]		PUPD10[1:0]		PUPD9[1:0]		PUPD8[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PUPD7[1:0]		PUPD6[1:0]		PUPD5[1:0]		PUPD4[1:0]		PUPD3[1:0]		PUPD2[1:0]		PUPD1[1:0]		PUPD0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	Name	R/W	Reset Value	Function
31:0	PUPDy [1:0]	**RW**		Y = 15..0 软件配置 I/O 口上拉或者下拉 00: 无上下拉 01: 上拉 10: 下拉 11: 保留

9.4.5. GPIO 端口输入数据寄存器(GPIOx_IDR) (x = A, B, F)

Address offset: 0x10

Reset value: 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
ID15	ID14	ID13	ID12	ID11	ID10	ID9	ID8	ID7	ID6	ID5	ID4	ID3	ID2	ID1	ID0
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bit	Name	R/W	Reset Value	Function
31:16	Reserved			
15:0	Idy	**R**		y = 15..0 这是只读的，读出值位对应 I/O 口的状态

9.4.6. GPIO 端口输出数据寄存器(GPIOx_ODR) (x = A, B, F)

Address offset: 0x14

Reset value: 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**
OD15	OD14	OD13	OD12	OD11	OD10	OD9	OD8	OD7	OD6	OD5	OD4	OD3	OD2	OD1	OD0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	Name	R/W	Reset Value	Function
31: 16	Reserved			
15: 0	Ody[1:0]	**RW**		y = 15..0 软件可读可写。 说明：对 GPIOx_BSRR or GPIOx_BRR registers. (x=A,B,F)，可以分别对各个 ODR 位进行独立的设置/清除。

9.4.7. GPIO 端口位设置/复位寄存器(GPIOx_BSRR) (x = A, B, F)

Address offset: 0x18

Reset value: 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	Name	R/W	Reset Value	Function
31:16	BRy	**W**		y = 15..0 软件可写，读出来返回值是 0 0: 对对应的 ODRy 位不产生影响 1: 清除对应的 ODRy 位 注：如果同时设置 Bsy 和 Bry 的对应位，Bsy 位起作用
15: 0	BSy	**W**		y = 15..0 软件可写， 读出来返回值是 0 0: 对对应的 ODRy 位不产生影响 1: 设置对应的 ODRy 位

9.4.8. GPIO 端口配置锁定寄存器(GPIOx_LCKR) (x = A, B, F)

当执行正确的写序列设置了 bit16（LCKK）时，该寄存器用来锁定端口位的配置。bit[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 LCKR[15:0]。当对相应的端口执行了 LOCK 序列后，在下次系统复位前将不能再更改端口位的配置。

注：特殊写时序用来写 GPIOx_LCKR 寄存器。在锁定时序中仅仅只有字访问可以被执行。

每个锁定位冻结一种特定的配置寄存器（控制和复用功能寄存器）

Address offset: 0x1C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	LCKK
															rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LCK15	LCK14	LCK13	LCK12	LCK11	LCK10	LCK9	LCK8	LCK7	LCK6	LCK5	LCK4	LCK3	LCK2	LCK1	LCK0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	Name	R/W	Reset Value	Function
31:17	Reserved			
16	LCKK	**RW**		该位可随时读出，它只能通过锁键写入序列修改 0: 端口配置锁键位未激活

Bit	Name	R/W	Reset Value	Function
				1: 端口配置锁键位被激活，下次系统复位前 GPIOx_LCKR 寄存器被锁定 LOCK key write sequence: 锁键的写入时序：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：在操作锁键的写入时序是，不能改变 LCK[15:0]的值。锁键时序的任何错误都会终止锁键被激活。对端口的任何一位首次锁键时序之后，读 LCKK 位都是返回 1,直接 MCU 复位或者外围复位。
15: 0	LCKy	**RW**		y = 15..0 这些位可读可写但只能在 LCKK 位为 0 是写入。 0: 不锁定端口的配置 1: 锁定端口配置

9.4.9. GPIO 复用功能寄存器（low）(GPIOx_AFRL) (x = A, B, F)

Address offset: 0x20

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AFSEL7[3:0]				AFSEL6[3:0]				AFSEL5[3:0]				AFSEL4[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AFSEL3[3:0]				AFSEL2[3:0]				AFSEL1[3:0]				AFSEL0[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	Name	R/W	Reset Value	Function
31:0	AFSELy[3:0]（(y= 7 to 0)）	**RW**		软件可写这些位配置复用功能 I/O AFSELy 选择: 0000:AF0　1000: AF8 0001:AF1　1001: AF9 0010:AF2　1010: AF10 0011:AF3　1011: AF11 0100:AF4　1100: AF12 0101:AF5　1101: AF13 0110:AF6　1110: AF14 0111:AF7　1111: AF15

9.4.10. GPIO 复用功能寄存器（high）(GPIOx_AFRH) (x = A, B, F)

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AFSEL15[3:0]				AFSEL14[3:0]				AFSEL13[3:0]				AFSEL12[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AFSEL11[3:0]				AFSEL10[3:0]				AFSEL9[3:0]				AFSEL8[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	Name	R/W	Reset Value	Function
31:0	AFSELy[3:0]（(y= 8 to 15)）	**RW**		软件可写这些位配置复用功能 I/O AFSELy 选择: 0000:AF0　1000: AF8 0001:AF1　1001: AF9 0010:AF2　1010: AF10 0011:AF3　1011: AF11 0100:AF4　1100: AF12 0101:AF5　1101: AF13 0110:AF6　1110: AF14 0111:AF7　1111: AF15

9.4.11. GPIO 端口位复位寄存器 (GPIOx_BRR) (x = A, B, F)

Address offset: 0x28

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res															
15	14	13	2	11	10	9	8	7	6	5	4	3	2	1	0
BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bit	Name	R/W	Reset Value	Function
31:16	Reserved			
15:0	Bry	**RW**		y = 15..0 这些位软件可写，读出来返回值是 0 0: 对对应的 Odry 位不产生影响 1: 清除对应的 Odry 位

9.4.12. GPIO 寄存器映像

Offset	Register	31 30	29 28	27 26	25 24	23 22	21 20	19 18	17 16	15 14	13 12	11 10	9 8	7 6	5 4	3 2	1 0
0x00	GPIOA_MODER	MODE15[1:0]	MODE14[1:0]	MODE13[1:0]	MODE12[1:0]	MODE11[1:0]	MODE10[1:0]	MODE9[1:0]	MODE8[1:0]	MODE7[1:0]	MODE6[1:0]	MODE5[1:0]	MODE4[1:0]	MODE3[1:0]	MODE2[1:0]	MODE1[1:0]	MODE0[1:0]
	Reset value	1 1	1 0	1 0	1 1	1 1	1 1	1 1	1 1	1 1	1 1	1 1	1 1	1 1	1 1	1 1	1 1
	GPIOB_MODER	MODE15[1:0]	MODE14[1:0]	MODE13[1:0]	MODE12[1:0]	MODE11[1:0]	MODE10[1:0]	MODE9[1:0]	MODE8[1:0]	MODE7[1:0]	MODE6[1:0]	MODE5[1:0]	MODE4[1:0]	MODE3[1:0]	MODE2[1:0]	MODE1[1:0]	MODE0[1:0]
	Reset value	0 0	0 0	0 0	0 0	0 0	0 0	0 0	1 1	1 1	1 1	1 1	1 1	1 1	1 1	1 1	1 1
	GPIOF_MODER	MODE15[1:0]	MODE14[1:0]	MODE13[1:0]	MODE12[1:0]	MODE11[1:0]	MODE10[1:0]	MODE9[1:0]	MODE8[1:0]	MODE7[1:0]	MODE6[1:0]	MODE5[1:0]	MODE4[1:0]	MODE3[1:0]	MODE2[1:0]	MODE1[1:0]	MODE0[1:0]
	Reset value	0 0	0 0	0 0	0 0	0 0	0 0	0 0	0 0	0 0	0 0	0 0	0 0	1 1	1 1	1 1	1 1
0x04	GPIOF_OTYPER (x=A, B, F)	Res. Res.	Res. Res.	Res. Res.	Res. Res.	Res. Res.	Res. Res.	Res. Res.	Res. Res.	OT15 OT14	OT13 OT12	OT11 OT10	OT9 OT8	OT7 OT6	OT5 OT4	OT3 OT2	OT1 OT0
	Reset value									0 0	0 0	0 0	0 0	0 0	0 0	0 0	0 0
0x08	GPIOA_OSPEEDR	OSPEED15[1:0]	OSPEED14[1:0]	OSPEED13[1:0]	OSPEED12[1:0]	OSPEED11[1:0]	OSPEED10[1:0]	OSPEED9[1:0]	OSPEED8[1:0]	OSPEED7[1:0]	OSPEED6[1:0]	OSPEED5[1:0]	OSPEED4[1:0]	OSPEED3[1:0]	OSPEED2[1:0]	OSPEED1[1:0]	OSPEED0[1:0]
	Reset value	0 0	0 0	1 1	0 0	0 0	0 0	0 0	0 0	0 0	0 0	0 0	0 0	0 0	0 0	0 0	0 0
0x08	GPIOx_OSPEEDR (x=B, F)	OSPEED15[1:0]	OSPEED14[1:0]	OSPEED13[1:0]	OSPEED12[1:0]	OSPEED11[1:0]	OSPEED10[1:0]	OSPEED9[1:0]	OSPEED8[1:0]	OSPEED7[1:0]	OSPEED6[1:0]	OSPEED5[1:0]	OSPEED4[1:0]	OSPEED3[1:0]	OSPEED2[1:0]	OSPEED1[1:0]	OSPEED0[1:0]
	Reset value	0 0	0 0	0 0	0 0	0 0	0 0	0 0	0 0	0 0	0 0	0 0	0 0	0 0	0 0	0 0	0 0
0x0C	GPIOA_PUPDR	PUPD15[1:0]	PUPD14[1:0]	PUPD13[1:0]	PUPD12[1:0]	PUPD11[1:0]	PUPD10[1:0]	PUPD9[1:0]	PUPD8[1:0]	PUPD7[1:0]	PUPD6[1:0]	PUPD5[1:0]	PUPD4[1:0]	PUPD3[1:0]	PUPD2[1:0]	PUPD1[1:0]	PUPD0[1:0]

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reset value	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	GPIOB_PUPDR	PUPD15[1:0]		PUPD14[1:0]		PUPD13[1:0]		PUPD12[1:0]		PUPD11[1:0]		PUPD10[1:0]		PUPD9[1:0]		PUPD8[1:0]		PUPD7[1:0]		PUPD6[1:0]		PUPD5[1:0]		PUPD4[1:0]		PUPD3[1:0]		PUPD2[1:0]		PUPD1[1:0]		PUPD0[1:0]	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	GPIOF_PUPDR	PUPD15[1:0]		PUPD14[1:0]		PUPD13[1:0]		PUPD12[1:0]		PUPD11[1:0]		PUPD10[1:0]		PUPD9[1:0]		PUPD8[1:0]		PUPD7[1:0]		PUPD6[1:0]		PUPD5[1:0]		PUPD4[1:0]		PUPD3[1:0]		PUPD2[1:0]		PUPD1[1:0]		PUPD0[1:0]	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0x10	GPIOx_IDR (x=A, B, F)	Res.		Res.		Res.		Res.		Res.		Res.		Res.		Res.		ID15	ID14	ID13	ID12	ID11	ID10	ID9	ID8	ID7	ID6	ID5	ID4	ID3	ID2	ID1	ID0
	Reset value																	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x14	GPIOx_ODR (x=A, B, F)	Res.		Res.		Res.		Res.		Res.		Res.		Res.		Res.		OD15	OD14	OD13	OD12	OD11	OD10	OD9	OD8	OD7	OD6	OD5	OD4	OD3	OD2	OD1	OD0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	GPIOx_BSRR (x=A, B, F)	BR15		BR14		BR13		BR12		BR11		BR10		BR9		BR8		BR7		BR6		BR5		BR4		BR3		BR2		BR1		BR0	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1C	GPIOx_LCKR (x=A, B, F)	Res.		Res.		Res.		Res.		Res.		Res.		Res.		Res.		LCKK															
	Reset value																	0															
0x20	GPIOx_AFRL (x=A, B, F)	AFSEL7 [3:0]				AFSEL6 [3:0]				AFSEL5 [3:0]				AFSEL4 [3:0]				AFSEL3 [3:0]				AFSEL2 [3:0]				AFSEL1 [3:0]				AFSEL0 [3:0]			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x24	GPIOx_AFRH (x=A, B, F)	AFSEL7 [3:0]				AFSEL6 [3:0]				AFSEL5 [3:0]				AFSEL4 [3:0]				AFSEL3 [3:0]				AFSEL2 [3:0]				AFSEL1 [3:0]				AFSEL0 [3:0]			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x28	GPIOx_BRR (x=A, B, F)	Res.		Res.		Res.		Res.		Res.		Res.		Res.		Res.		BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

10. 系统配置控制器(SYSCFG)

芯片内有一套配置寄存器，系统配置控制器的主要目的是：

- 使能或者不使能在某些 IO pin 上的 I2C fast Mode Plus
- 重映射位于代码区间开始区域的存储器
- 管理连接到 GPIO 的外部中断
- 管理鲁棒性特性

10.1. 系统配置寄存器

10.1.1. SYSCFG 配置寄存器 1(SYSCFG_CFGR1)

该寄存器用作存储器和控制特殊 IO 功能的具体配置。

有两位用作配置存储器地址 0x0000 0000 访问的种类。这两位用来选择软件的物理重映射，并 bypass 掉硬件 BOOT 选择。在复位后，这些位使用被实际 boot 模式配置的值。

Address offset：0x00

Reset value：0x0000 000x(x 是被实际 boot 模式配置选择的存储器模式)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	I2C_PF1_ANF	I2C_PF0_ANF	I2C_PB8_ANF	I2C_PB7_ANF	I2C_PB6_ANF	I2C_PA12_ANF	I2C_PA11_ANF	I2C_PA10_ANF	I2C_PA9_ANF	I2C_PA8_ANF	I2C_PA7_ANF	I2C_PA3_ANF	I2C_PA2_ANF	Res	Res
	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**
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	MEM_MODE[1:0]	
														RW	

Bit	Name	R/W	Reset Value	Function
31	Reserved	RW	-	可读可写
30: 18	I2C_IOx_ANF	RW	0	I2C 相关 IO 的模拟滤波使能控制 0：模拟滤波关闭 1：模拟滤波使能
17:2	Reserved	RW	0	可读可写
1:0	MEM_MODE[1:0]			Memory mapping 选择位 软件置位，软件清零。他们控制存储器的 0x0000 0000 地址的 mapping。在复位后，这些位采用实际实际启动模式配置值。 X0：Main flash, mapped 在 0x0000 0000 01：System flash , mapped 在 0x0000 0000 11：SRAM, mapped 在 0x0000 0000

10.1.2. SYSCFG 配置寄存器 2 (SYSCFG_CFGR2)

Address offset：0x18

Reset value：0x0000

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

Res	Res	Res	Res	Res	ETR_SRC_TIM1	Res	Res	COMP2_BRK_TIM16	COMP1_BRK_TIM16	COMP2_BRK_TIM1	COMP1_BRK_TIM1	Res	Res	LOCKUP_LOCK
					RW			RW	RW	RW	RW			RW

Bit	Name	R/W	Reset Value	Function
31:11	Reserved	-	-	-
10:9	ETR_SRC_TIM1[1:0]	RW	2'b00	TIMER1 ETR 输入源选择。 2'b00: ETR 来源于 GPIO 2'b01: ETR 来源于 COMP1 2'b10: ETR 来源于 COMP2 2'b11: ETR 来源于 ADC
8:7	Reserved	-	-	-
6	COMP2_BRK_TIM16	RW	0	COMP2 作为 TIMx break 输入使能。 0：COMP2 输出不作为 TIM16 break input 1：COMP2 输出作为 TIM16 break input
5	COMP1_BRK_TIM16	RW	0	COMP1 作为 TIMx break 输入使能。 0：COMP1 输出不作为 TIM16 break input 1：COMP1 输出作为 TIM16 break input
4	COMP2_BRK_TIM1	RW	0	COMP2 作为 TIMx break 输入使能。 0：COMP2 输出不作为 TIM1 break input 1：COMP2 输出作为 TIM1 break input
3	COMP1_BRK_TIM1	RW	0	COMP1 作为 TIMx break 输入使能。 0：COMP1 输出不作为 TIM1 break input 1：COMP1 输出作为 TIM1 break input
2:1	Reserved	-	-	-
0	LOCKUP_LOCK	RW		Cortex-M0+ LOCKUP 位的使能位 软件置位，系统复位清零。它可以使能和锁定 Cortex-M0+的 LOCKUP(HardFault)输出给 TIM1/TIM16 的刹车输入。 0：Cortex-M0+的 LOCKUP 输出不与 TIM1/TIM16 的刹车输入连接 1：Cortex-M0+的 LOCKUP 输出与 TIM1/TIM16 的刹车输入连接

10.1.3. SYSCFG 寄存器映像

Off-set	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8
0x00	SYS-CFG_CFGR1	Res.	I2C_PF1_ANF	I2C_PF0_ANF	I2C_PB8_ANF	I2C_PB7_ANF	I2C_PB6_ANF	I2C_PA12_ANF	I2C_PA11_ANF	I2C_PA10_ANF	I2C_PA9_ANF	I2C_PA8_ANF	I2C_PA7_ANF	I2C_PA3_ANF	I2C_PA2_ANF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0										
0x18	SYS-CFG_CFGR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ETR_SRC_TIM1[1:0]		Res.
	Reset value																						0		

11. 中断和事件

11.1. 嵌套向量中断控制器(NVIC)

11.1.1. 主要特性

- 最多 32 个可屏蔽的中断通道（不包括 16 个 CPU 的中断）
- 4 个可编程的优先级（2 位中断优先级）
- 低延迟的 exception 和中断处理
- 功耗管理控制
- 系统控制寄存器的实现

NVIC 和 CPU 接口是紧耦合的，这使得低延迟中断处理和后到达中断的高效处理成为可能。包括 CPU 的 exception，所有中断都被 NVIC 管理。

11.1.2. 系统嘀嗒（SysTick）校准值寄存器

系统嘀嗒校准值被设为 6000，通过 SysTick 时钟置为 6MHz（max f_{HCLK}/8），给出了 1ms 的参考 time base。

11.1.3. 中断和异常向量

位置	优先级	优先级类型	名称	说明	地址
-	-	-	-	保留	0x0000_0000
-	-3	固定	复位	复位	0x0000_0004
-	-2	固定	NMI_Handler	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	HardFualt_Handler	所有类型的失效	0x0000_000C
-	3	可设置	SVCall	通过 SWI 指令的系统服务调用	0x0000_002C
-	5	可设置	PendSV	可挂起的系统服务	0x0000_0038
	6		SysTick	系统嘀嗒定时器	0x0000_003C
0	7	-	保留	保留	0x0000_0040
1	8	-	保留	保留	0x0000_0044
2	9	-	保留	保留	0x0000_0048
3	10	可设置	Flash	Flash 全局中断	0x0000_004C
4	11	可设置	RCC	RCC 全局中断	0x0000_0050
5	12	可设置	EXTI0_1	EXTI line[1:0] interrupt	0x0000_0054
6	13	可设置	EXTI2_3	EXTI line[3:2] interrupt	0x0000_0058
7	14	可设置	EXTI4_15	EXTI line[15:4] interrupt	0x0000_005C
8	15	-	保留	保留	0x0000_0060
9	16	-	保留	保留	0x0000_0064
10	17	-	保留	保留	0x0000_0068
11	18	-	保留	保留	0x0000_006C
12	19	可设置	ADC_COMP	ADC and COMP interrupts (COMP combined with EXTI 17 & 18)	0x0000_0070
13	20	可设置	TIM1_BRK_UP_TRG_COM	TIM1 断开、更新、触发和通信中断	0x0000_0074
14	21	可设置	TIM1_CC	TIM1 捕获/比较中断	0x0000_0078
15	22	-	保留	保留	0x0000_007C
16	23	-	保留	保留	0x0000_0080
17	24	可设置	LPTIM1	LPTIM 中断	0x0000_0084
18	25	-	保留	保留	0x0000_0088
19	26	-	保留	保留	0x0000_008C
20	27	-	保留	保留	0x0000_0090

位置	优先级	优先级类型	名称	说明	地址
21	28	可设置	TIM16	TIM16 全局中断	0x0000_0094
22	29	-	保留	保留	0x0000_0098
23	30	可设置	I2C1	I2C1 全局中断	0x0000_009C
24	31	-	保留	保留	0x0000_00A0
25	32	可设置	SPI1	SPI1 全局中断	0x0000_00A4
26	33	-	保留	保留	0x0000_00A8
27	34	可设置	USART1	USART1 全局中断	0x0000_00AC
28	35	-	保留	保留	0x0000_00B0
29	36	-	-	-	0x0000_00B4
30	37	-	保留	保留	0x0000_00B8
31	38	-	保留	保留	0x0000_00BC

1. 灰色单元格（地址小于 0x0000 0040）对应于 Cortex-M0® +中断。

11.2. 外部中断/事件控制器(EXTI)

扩展中断和事件控制器，通过可配置和直接事件输入，管理着 CPU 和系统唤醒功能，并输出下述请求信号：

- 中断请求，送给 int_ctrl 模块产生 CPU 的 IRQ
- 事件请求，送给 CPU 的事件输入（RXEV）
- 唤醒请求，送给功耗管理控制模块

EXTI 唤醒请求允许系统从 stop 模式唤醒，中断请求和事件请求也可以在 run 模式使用。

EXTI 允许管理多达 21 个 configurable/direct 事件 line（19 个 configurable 事件 Line 和 2 个 direct 事件 line）。

11.2.1. EXTI 主要特性

- 系统可以通过 GPIO 和指定模块（LPTIM/COMP）输入事件唤醒
- Configurable 型事件（来自 I/O，或无状态 pending 位的外设，产生脉冲的外设）
  - ✓ 可选有效触发沿（上升沿/下降沿）
  - ✓ 中断挂起标志位
  - ✓ 独立中断和事件产生屏蔽位
  - ✓ 可软件触发
- Direct 型事件（具有关联标志和中断 pending 状态位的外设）
  - ✓ 固定的上升沿触发
  - ✓ 在 EXTI 模块里没有中断 pending 位
  - ✓ 独立中断和事件产生屏蔽位
  - ✓ 无软件触发
- IO 端口选择

11.2.2. EXTI 框图

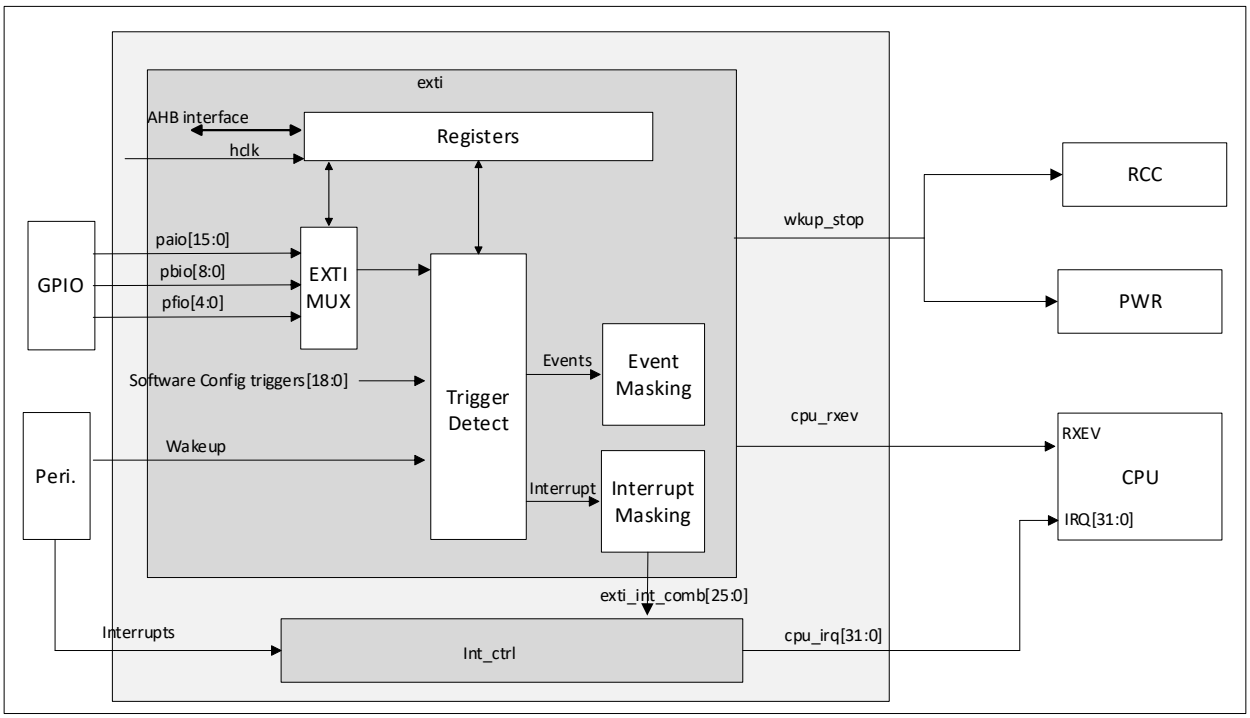


图 11-1 EXTI 框图

11.2.3. 外设和 CPU 的 EXTI 连接

在 stop 模式下能产生唤醒或者中断事件信号的外设，连接至 EXTI 模块。

- 产生一个脉冲的，或者外设内部没有中断状态位的唤醒信号，被连接到 EXTI 模块的 configurable line。此时 EXTI 模块产生一个中断挂起位（该位需要被清零），该 EXTI 中断会作为 CPU 的中断信号。
- 有关联的状态位（该位在外设被清零）的外设的中断和唤醒信号，连接到 EXTI 模块的唤醒触发信号线。
- 所有 GPIO port 输入到 EXTI MUX 模块，通过 configurable 的配置，允许选中后作为系统唤醒信号。

11.2.4. EXTI 可配置事件（configurable）触发唤醒

通过配置 EXTI_SWIER1 寄存器，软件可以触发唤醒功能。

有对应寄存器配置上升沿或者下降沿触发或者双沿触发 configurable 类型事件，硬件根据配置检测 configurable 类型事件输入信号，产生对应唤醒事件或者中断信号。

CPU 有专用中断屏蔽寄存器和事件屏蔽寄存器。事件使能后产生给 CPU 的事件。所有给 CPU 的事件‘或’运算后输出到 CPU 的唯一事件输入信号 rxev。

Configurable 类型事件有唯一的中断挂起请求寄存器，与 CPU 共享。挂起寄存器只有当 CPU 中断屏蔽寄存器（EXTI_IMR）配置为未屏蔽时才会置位。每一个 configurable 类型事件都会对应 CPU 外部中断信号（有些会复用到同一个 CPU 外部中断信号）。Configurable 类型事件中断需要 CPU 通过 EXTI_PR 寄存器确认（写 1 清零）。

注：当中断 pending 寄存器（EXTI_PR）有 bit 保持有效时（未清零），系统不能进入低功耗模式。

11.2.5. EXTI 直接类型事件输入唤醒

direct 类型事件会在 EXTI 模块产生中断，并会产生唤醒系统和 CPU 子系统的事件信号。CPU 在处理该种类型触发事件产生的中断时，要清零外设模块的中断状态位。

11.2.6. EXTI 选择器

GPIO 被用以下方式连接到 16 个外部中断/事件 line 上：

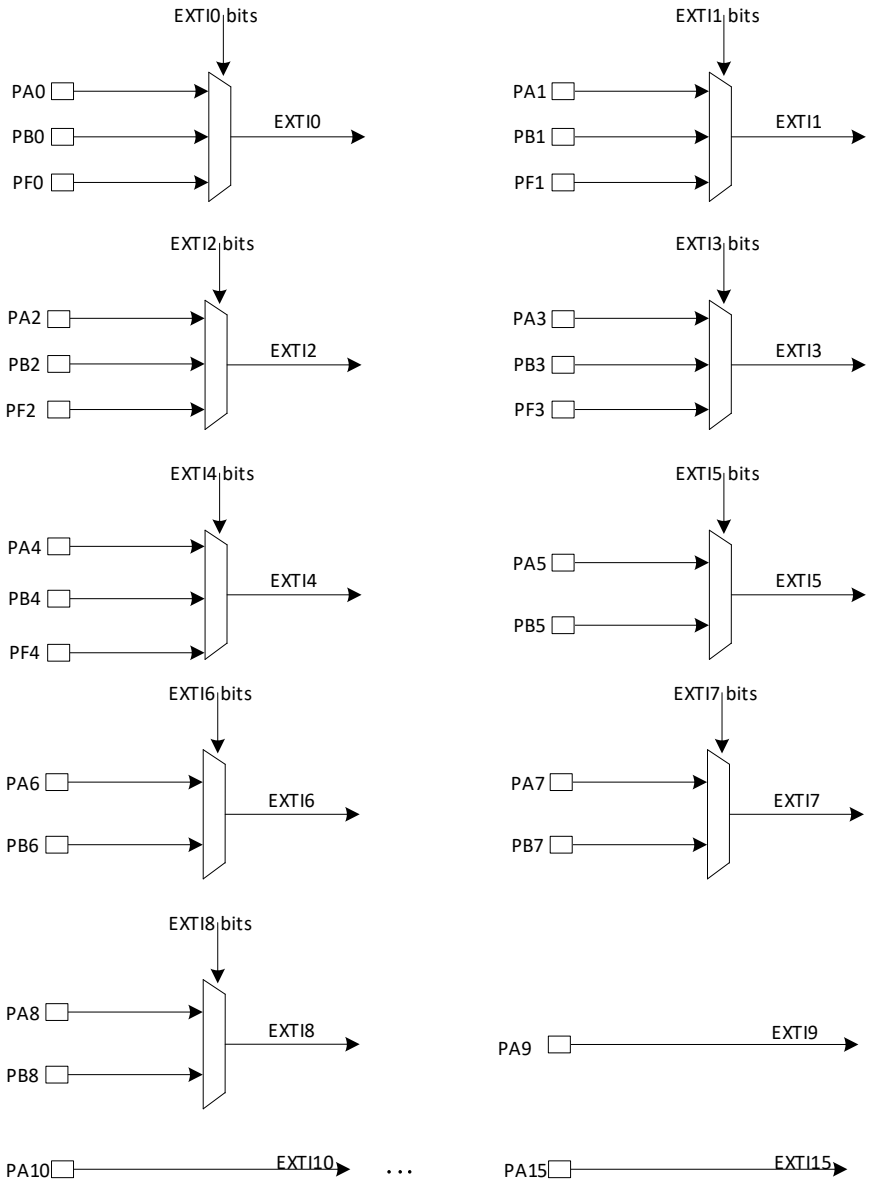


图 11-2 外部中断/事件 GPIO 映像

所有 line 连接内容如下表所示：

EXTI line	Line source	Line type
Line 0-15	GPIO	configurable
Line 16	Reserved	
Line 17	COMP 1 output	Configurable
Line 18	COMP 2 output	Configurable
Line 19	Reserved	
Line 20	Reserved	
Line 21	Reserved	
Line 22	Reserved	
Line 23	Reserved	
Line 24	Reserved	
Line 25	Reserved	
Line 26	Reserved	
Line 27	Reserved	
Line 28	Reserved	
Line 29	LPTIM	Direct

11.3. EXTI 寄存器

该外设的寄存器可以用 word(32bit)、half-word（16bit）和 byte（8bit）访问。

11.3.1. 上升沿触发选择寄存器 (EXTI_RTSR)

Address offset: 0x00

Reset value: 0x0000 0000

仅包含对 configurable 事件的寄存器控制位。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res		Res	Res	Res	Res	Res	Res	Res	Res	Res		RT18	RT17	RT16
													RW	RW	RW
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
RT15	RT14	RT13	RT12	RT11	RT10	RT9	RT8	RT7	RT6	RT5	RT4	RT3	RT2	RT1	RT0
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:19	Reserved			
18	RT18	RW	0	Configurable 类型 EXTI line18 上升沿触发配置。 0：禁止 1：使能
17	RT17	RW	0	Configurable 类型 EXTI line17 上升沿触发配置。 0：禁止 1：使能
16	RT16	RW	0	Configurable 类型 EXTI line16 上升沿触发配置。 0：禁止 1：使能
15	RT15	RW	0	Configurable 类型 EXTI line15 上升沿触发配置。 0：禁止 1：使能
14	RT14	RW	0	Configurable 类型 EXTI line14 上升沿触发配置。 0：禁止 1：使能
13	RT13	RW	0	Configurable 类型 EXTI line13 上升沿触发配置。 0：禁止 1：使能
12	RT12	RW	0	Configurable 类型 EXTI line12 上升沿触发配置。 0：禁止 1：使能
11	RT11	RW	0	Configurable 类型 EXTI line11 上升沿触发配置。 0：禁止 1：使能
10	RT10	RW	0	Configurable 类型 EXTI line10 上升沿触发配置。 0：禁止 1：使能
9	RT9	RW	0	Configurable 类型 EXTI line9 上升沿触发配置。 0：禁止 1：使能
8	RT8	RW	0	Configurable 类型 EXTI line8 上升沿触发配置。 0：禁止 1：使能
7	RT7	RW	0	Configurable 类型 EXTI line7 上升沿触发配置。 0：禁止 1：使能
6	RT6	RW	0	Configurable 类型 EXTI line6 上升沿触发配置。 0：禁止 1：使能
5	RT5	RW	0	Configurable 类型 EXTI line5 上升沿触发配置。 0：禁止 1：使能
4	RT4	RW	0	Configurable 类型 EXTI line4 上升沿触发配置。 0：禁止

Bit	Name	R/W	Reset Value	Function
				1：使能
3	RT3	RW	0	Configurable 类型 EXTI line3 上升沿触发配置。 0：禁止 1：使能
2	RT2	RW	0	Configurable 类型 EXTI line2 上升沿触发配置。 0：禁止 1：使能
1	RT1	RW	0	Configurable 类型 EXTI line1 上升沿触发配置。 0：禁止 1：使能
0	RT0	RW	0	Configurable 类型 EXTI line0 上升沿触发配置。 0：禁止 1：使能

configurable line 是边沿触发的，在这些 Line 上不能产生毛刺。如果在写 EXTI_RTSR 寄存器期间，configurable 中断线出现了上升沿，相关的 Pending 位不被置位。

在同一个 line 上可以同时设置上升和下降沿，在该情况下，两种边沿都会产生触发条件。

11.3.2. 下降沿触发选择寄存器 (EXTI_FTSR)

Address offset: 0x04

Reset value: 0x0000 0000

仅包含对 configurable 事件的寄存器控制位。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	FT18	FT17	FT16
													RW	RW	RW
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
FT15	FT14	FT13	FT12	FT11	FT10	FT9	FT8	FT7	FT6	FT5	FT4	FT3	FT2	FT1	FT0
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31：19	Reserved		-	
18	FT18	RW	0	Configurable 类型 EXTI line18 下降沿触发配置。 0：禁止 1：使能
17	FT17	RW	0	Configurable 类型 EXTI line17 下降沿触发配置。 0：禁止 1：使能
16	FT16	RW	0	Configurable 类型 EXTI line16 下降沿触发配置。 0：禁止 1：使能
15	FT15	RW	0	Configurable 类型 EXTI line15 下降沿触发配置。 0：禁止 1：使能
14	FT14	RW	0	Configurable 类型 EXTI line14 下降沿触发配置。 0：禁止 1：使能
13	FT13	RW	0	Configurable 类型 EXTI line13 下降沿触发配置。 0：禁止 1：使能
12	FT12	RW	0	Configurable 类型 EXTI line12 下降沿触发配置。 0：禁止 1：使能
11	FT11	RW	0	Configurable 类型 EXTI line11 下降沿触发配置。 0：禁止 1：使能
10	FT10	RW	0	Configurable 类型 EXTI line10 下降沿触发配置。 0：禁止 1：使能

Bit	Name	R/W	Reset Value	Function
9	FT9	RW	0	Configurable 类型 EXTI line9 下降沿触发配置。 0：禁止 1：使能
8	FT8	RW	0	Configurable 类型 EXTI line8 下降沿触发配置。 0：禁止 1：使能
7	FT7	RW	0	Configurable 类型 EXTI line7 下降沿触发配置。 0：禁止 1：使能
6	FT6	RW	0	Configurable 类型 EXTI line6 下降沿触发配置。 0：禁止 1：使能
5	FT5	RW	0	Configurable 类型 EXTI line5 下降沿触发配置。 0：禁止 1：使能
4	FT4	RW	0	Configurable 类型 EXTI line4 下降沿触发配置。 0：禁止 1：使能
3	FT3	RW	0	Configurable 类型 EXTI line3 下降沿触发配置。 0：禁止 1：使能
2	FT2	RW	0	Configurable 类型 EXTI line2 下降沿触发配置。 0：禁止 1：使能
1	FT1	RW	0	Configurable 类型 EXTI line1 下降沿触发配置。 0：禁止 1：使能
0	FT0	RW	0	Configurable 类型 EXTI line0 下降沿触发配置。 0：禁止 1：使能

Configurable line 是边沿触发的，在这些 Line 上不能产生毛刺。如果在写 EXTI_FTSR 寄存器期间，configurable line 出现了下降沿，相关的 Pending 位不被置位。

在同一个 line 上可以同时设置上升和下降沿，在该情况下，两种边沿都会产生触发条件。

11.3.3. 软件中断事件寄存器 (EXTI_SWIER)

Address offset: 0x08

Reset value: 0x0000 0000

仅包含对 configurable 事件的寄存器控制位。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
SW15	SW14	SW13	SW12	SW11	SW10	SW9	SW8	SW7	SW6	SW5	SW4	SW3	SW2	SW1	SW0
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31：16	Reserved		-	
15	SWI15	RW	0	Configurable 类型 EXTI line15 软件上升沿触发配置。 0：No effect 1：产生上升沿触发事件，进而产生中断 该位由硬件清零，读返回 0（硬件清零后）或者配置值（硬件清零前）
14	SWI14	RW	0	Configurable 类型 EXTI line14 软件上升沿触发配置。 0：No effect 1：产生上升沿触发事件，进而产生中断 该位由硬件自清。读返回 0.

Bit	Name	R/W	Reset Value	Function
13	SWI13	RW	0	Configurable 类型 EXTI line13 软件上升沿触发配置。 0：No effect 1：产生上升沿触发事件，进而产生中断 该位由硬件自清。读返回 0.
12	SWI12	RW	0	Configurable 类型 EXTI line12 软件上升沿触发配置。 0：No effect 1：产生上升沿触发事件，进而产生中断 该位由硬件清零，读返回 0（硬件清零后）或者配置值（硬件清零前）
11	SWI11	RW	0	Configurable 类型 EXTI line11 软件上升沿触发配置。 0：No effect 1：产生上升沿触发事件，进而产生中断 该位由硬件清零，读返回 0（硬件清零后）或者配置值（硬件清零前）
10	SWI10	RW	0	Configurable 类型 EXTI line10 软件上升沿触发配置。 0：No effect 1：产生上升沿触发事件，进而产生中断 该位由硬件清零，读返回 0（硬件清零后）或者配置值（硬件清零前）
9	SWI9	RW	0	Configurable 类型 EXTI line9 软件上升沿触发配置。 0：No effect 1：产生上升沿触发事件，进而产生中断 该位由硬件自清。读返回 0.
8	SWI8	RW	0	Configurable 类型 EXTI line8 软件上升沿触发配置。 0：No effect 1：产生上升沿触发事件，进而产生中断 该位由硬件清零，读返回 0（硬件清零后）或者配置值（硬件清零前）
7	SWI7	RW	0	Configurable 类型 EXTI line7 软件上升沿触发配置。 0：No effect 1：产生上升沿触发事件，进而产生中断 该位由硬件清零，读返回 0（硬件清零后）或者配置值（硬件清零前）
6	SWI6	RW	0	Configurable 类型 EXTI line6 软件上升沿触发配置。 0：No effect 1：产生上升沿触发事件，进而产生中断 该位由硬件清零，读返回 0（硬件清零后）或者配置值（硬件清零前）
5	SWI5	RW	0	Configurable 类型 EXTI line5 软件上升沿触发配置。 0：No effect 1：产生上升沿触发事件，进而产生中断 该位由硬件清零，读返回 0（硬件清零后）或者配置值（硬件清零前）
4	SWI4	RW	0	Configurable 类型 EXTI line4 软件上升沿触发配置。 0：No effect 1：产生上升沿触发事件，进而产生中断 该位由硬件清零，读返回 0（硬件清零后）或者配置值（硬件清零前）
3	SWI3	RW	0	Configurable 类型 EXTI line3 软件上升沿触发配置。 0：No effect 1：产生上升沿触发事件，进而产生中断 该位由硬件清零，读返回 0（硬件清零后）或者配置值（硬件清零前）
2	SWI2	RW	0	Configurable 类型 EXTI line2 软件上升沿触发配置。 0：No effect 1：产生上升沿触发事件，进而产生中断 该位由硬件清零，读返回 0（硬件清零后）或者配置值（硬件清零前）
1	SWI1	RW	0	Configurable 类型 EXTI line1 软件上升沿触发配置。 0：No effect 1：产生上升沿触发事件，进而产生中断

Bit	Name	R/W	Reset Value	Function
				该位由硬件清零，读返回 0（硬件清零后）或者配置值（硬件清零前）
0	SWI0	RW	0	Configurable 类型 EXTI line0 软件上升沿触发配置。 0：No effect 1：产生上升沿触发事件，进而产生中断 该位由硬件清零，读返回 0（硬件清零后）或者配置值（硬件清零前）

11.3.4. 挂起寄存器(EXTI_PR)

Address offset: 0x0C

Reset value: undefined

仅包含对 configurable 事件的寄存器控制位。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
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	Name	R/W	Reset Value	Function
31：16	Reserved	reserved	-	
15	PR15	RC_W1	0	Configurable 类型 EXTI line15 事件挂起标志。软件或者硬件产生上升沿/下降沿触发事件时，该位置位。软件写 1 清零。 0：未产生事件请求； 1：产生上升沿/下降沿/软件触发事件请求；
14	PR14	RC_W1	0	Configurable 类型 EXTI line14 事件挂起标志。软件或者硬件产生上升沿/下降沿触发事件时，该位置位。软件写 1 清零。 0：未产生事件请求； 1：产生上升沿/下降沿/软件触发事件请求；
13	PR13	RC_W1	0	Configurable 类型 EXTI line13 事件挂起标志。软件或者硬件产生上升沿/下降沿触发事件时，该位置位。软件写 1 清零。 0：未产生事件请求； 1：产生上升沿/下降沿/软件触发事件请求；
12	PR12	RC_W1	0	Configurable 类型 EXTI line12 事件挂起标志。软件或者硬件产生上升沿/下降沿触发事件时，该位置位。软件写 1 清零。 0：未产生事件请求； 1：产生上升沿/下降沿/软件触发事件请求；
11	PR11	RC_W1	0	Configurable 类型 EXTI line11 事件挂起标志。软件或者硬件产生上升沿/下降沿触发事件时，该位置位。软件写 1 清零。 0：未产生事件请求； 1：产生上升沿/下降沿/软件触发事件请求；
10	PR10	RC_W1	0	Configurable 类型 EXTI line10 事件挂起标志。软件或者硬件产生上升沿/下降沿触发事件时，该位置位。软件写 1 清零。 0：未产生事件请求； 1：产生上升沿/下降沿/软件触发事件请求；
9	PR9	RC_W1	0	Configurable 类型 EXTI line9 事件挂起标志。软件或者硬件产生上升沿/下降沿触发事件时，该位置位。软件写 1 清零。 0：未产生事件请求；

Bit	Name	R/W	Reset Value	Function
				1：产生上升沿/下降沿/软件触发事件请求；
8	PR8	RC_W1	0	Configurable 类型 EXTI line8 事件挂起标志。软件或者硬件产生上升沿/下降沿触发事件时，该位置位。软件写 1 清零。 0：未产生事件请求； 1：产生上升沿/下降沿/软件触发事件请求；
7	PR7	RC_W1	0	Configurable 类型 EXTI line7 事件挂起标志。软件或者硬件产生上升沿/下降沿触发事件时，该位置位。软件写 1 清零。 0：未产生事件请求； 1：产生上升沿/下降沿/软件触发事件请求；
6	PR6	RC_W1	0	Configurable 类型 EXTI line6 事件挂起标志。软件或者硬件产生上升沿/下降沿触发事件时，该位置位。软件写 1 清零。 0：未产生事件请求； 1：产生上升沿/下降沿/软件触发事件请求；
5	PR5	RC_W1	0	Configurable 类型 EXTI line5 事件挂起标志。软件或者硬件产生上升沿/下降沿触发事件时，该位置位。软件写 1 清零。 0：未产生事件请求； 1：产生上升沿/下降沿/软件触发事件请求；
4	PR4	RC_W1	0	Configurable 类型 EXTI line4 事件挂起标志。软件或者硬件产生上升沿/下降沿触发事件时，该位置位。软件写 1 清零。 0：未产生事件请求； 1：产生上升沿/下降沿/软件触发事件请求；
3	PR3	RC_W1	0	Configurable 类型 EXTI line3 事件挂起标志。软件或者硬件产生上升沿/下降沿触发事件时，该位置位。软件写 1 清零。 0：未产生事件请求； 1：产生上升沿/下降沿/软件触发事件请求；
2	PR2	RC_W1	0	Configurable 类型 EXTI line2 事件挂起标志。软件或者硬件产生上升沿/下降沿触发事件时，该位置位。软件写 1 清零。 0：未产生事件请求； 1：产生上升沿/下降沿/软件触发事件请求；
1	PR1	RC_W1	0	Configurable 类型 EXTI line1 事件挂起标志。软件或者硬件产生上升沿/下降沿触发事件时，该位置位。软件写 1 清零。 0：未产生事件请求； 1：产生上升沿/下降沿/软件触发事件请求；
0	PR0	RC_W1	0	Configurable 类型 EXTI line0 事件挂起标志。软件或者硬件产生上升沿/下降沿触发事件时，该位置位。软件写 1 清零。 0：未产生事件请求； 1：产生上升沿/下降沿/软件触发事件请求；

11.3.5. 外部中断选择寄存器 1 (EXTI_EXTICR1)

Address offset：0x60

Reset value：0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	EXTI3[1:0]		Res	Res	Res	Res	Res	Res	EXTI2[1:0]	
						RW	RW							RW	RW
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res	Res	Res	Res	Res	Res	EXTI1[1:0]		Res	Res	Res	Res	Res	Res	EXTI0[1:0]	
						RW	RW							RW	RW

Bit	Name	R/W	Reset Value	Function
31:21	Reserved	-	-	Reserved

Bit	Name	R/W	Reset Value	Function
25:24	EXTI3[1:0]	RW	0	EXTI3 对应 GPIO port 选择。 2'b00: PA[3] pin 2'b01: PB[3] pin 2'b10: PF[3] pin 2'b11: reserved
23:18	Reserved	-	-	Reserved
17:16	EXTI2[1:0]	RW	0	EXTI2 对应 GPIO port 选择。 2'b00: PA[2] pin 2'b01: PB[2] pin 2'b10: PF[2] pin 2'b11: reserved
15:10	Reserved	-	-	Reserved
9:8	EXTI1[1:0]	RW	0	EXTI1 对应 GPIO port 选择。 2'b00: PA[1] pin 2'b01: PB[1] pin 2'b10: PF[1] pin 2'b11: reserved
7:2	Reserved	-	-	Reserved
1:0	EXTI0[1:0]	RW	0	EXTI0 对应 GPIO port 选择。 2'b00: PA[0] pin 2'b01: PB[0] pin 2'b10: PF[0] pin 2'b11: reserved

11.3.6. 外部中断选择寄存器 2 (EXTI_EXTICR2)

Address offset：0x64

Reset value：0x0000

31	**30**	**29**	**28**	**27**	**26**	**25**	**24**	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Res	Res	Res	Res	Res	Res	Res	EXTI7	Res	Res	Res	Res	Res	Res	Res	EXTI6
							RW								RW
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res	Res	Res	Res	Res	Res	RW	EXTI5	Res	Res	Res	Res	Res	Res	EXTI4[1:0]	
							RW							RW	RW

Bit	Name	R/W	Reset Value	Function
31:25	Reserved	-	-	Reserved
24	EXTI7	RW	0	EXTI7 对应 GPIO port 选择。 0: PA[7] pin 1: PB[7] pin
23:18	Reserved	-	-	Reserved
17:16	EXTI6	RW	0	EXTI6 对应 GPIO port 选择。 0: PA[6] pin 1: PB[6] pin
15:9	Reserved	-	-	Reserved
8	EXTI5	RW	0	EXTI5 对应 GPIO port 选择。 0: PA[5] pin 1: PB[5] pin
7:2	Reserved	-	-	Reserved
1:0	EXTI4[1:0]	RW	0	EXTI4 对应 GPIO port 选择。 2'b00: PA[4] pin 2'b01: PB[4] pin 2'b10: PF[4] pin 2'b11: reserved

11.3.7. 外部中断选择寄存器 3 (EXTI_EXTICR3)

Address offset：0x68

Reset value：0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	EXTI8
															RW

Bit	Name	R/W	Reset Value	Function
31:1	Reserved	-	-	Reserved
0	EXTI8	RW	0	EXTI8 对应 GPIO port 选择。 0: PA[8] pin 1: PB[8] pin

11.3.8. 中断屏蔽寄存器（EXTI_IMR）

Address offset：0x80

Reset value:0x2008 0000

注意： Direct 类型 line 的中断 mask bit 默认为 1，即允许该 line；configurable line 的 mask 位，默认为 0，即屏蔽该 line。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	IM29	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
		RW													
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
IM15	IM14	IM13	IM12	IM11	IM10	IM9	IM8	IM7	IM6	IM5	IM4	IM3	IM2	IM1	IM0
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:30	Reserved			
29	IM29	RW	1	EXTI line29 作为中断唤醒 CPU 屏蔽控制。 0：中断唤醒屏蔽 1：中断唤醒未屏蔽
28:16	Reserved			
15	IM15	RW	0	EXTI line15 作为中断唤醒 CPU 屏蔽控制。 0：中断唤醒屏蔽 1：中断唤醒未屏蔽
14	IM14	RW	0	EXTI line14 作为中断唤醒 CPU 屏蔽控制。 0：中断唤醒屏蔽 1：中断唤醒未屏蔽
13	IM13	RW	0	EXTI line13 作为中断唤醒 CPU 屏蔽控制。 0：中断唤醒屏蔽 1：中断唤醒未屏蔽
12	IM12	RW	0	EXTI line12 作为中断唤醒 CPU 屏蔽控制。 0：中断唤醒屏蔽 1：中断唤醒未屏蔽
11	IM11	RW	0	EXTI line11 作为中断唤醒 CPU 屏蔽控制。 0：中断唤醒屏蔽 1：中断唤醒未屏蔽
10	IM10	RW	0	EXTI line10 作为中断唤醒 CPU 屏蔽控制。 0：中断唤醒屏蔽 1：中断唤醒未屏蔽
9	IM9	RW	0	EXTI line9 作为中断唤醒 CPU 屏蔽控制。 0：中断唤醒屏蔽 1：中断唤醒未屏蔽
8	IM8	RW	0	EXTI line8 作为中断唤醒 CPU 屏蔽控制。 0：中断唤醒屏蔽 1：中断唤醒未屏蔽
7	IM7	RW	0	EXTI line7 作为中断唤醒 CPU 屏蔽控制。 0：中断唤醒屏蔽 1：中断唤醒未屏蔽
6	IM6	RW	0	EXTI line6 作为中断唤醒 CPU 屏蔽控制。

Bit	Name	R/W	Reset Value	Function
				0：中断唤醒屏蔽 1：中断唤醒未屏蔽
5	IM5	RW	0	EXTI line5 作为中断唤醒 CPU 屏蔽控制。 0：中断唤醒屏蔽 1：中断唤醒未屏蔽
4	IM4	RW	0	EXTI line4 作为中断唤醒 CPU 屏蔽控制。 0：中断唤醒屏蔽 1：中断唤醒未屏蔽
3	IM3	RW	0	EXTI line3 作为中断唤醒 CPU 屏蔽控制。 0：中断唤醒屏蔽 1：中断唤醒未屏蔽
2	IM2	RW	0	EXTI line2 作为中断唤醒 CPU 屏蔽控制。 0：中断唤醒屏蔽 1：中断唤醒未屏蔽
1	IM1	RW	0	EXTI line1 作为中断唤醒 CPU 屏蔽控制。 0：中断唤醒屏蔽 1：中断唤醒未屏蔽
0	IM0	RW	0	EXTI line0 作为中断唤醒 CPU 屏蔽控制。 0：中断唤醒屏蔽 1：中断唤醒未屏蔽

11.3.9. 事件屏蔽寄存器(EXTI_EMR)

Address offset: 0x84

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	EM29	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
		RW													
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
EM15	EM14	EM13	EM12	EM11	EM10	EM9	EM8	EM7	EM6	EM5	EM4	EM3	EM2	EM1	EM0
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:30	Reserved			
29	EM29	RW	0	EXTI line29 作为事件唤醒 CPU 屏蔽控制。 0：事件唤醒屏蔽 1：事件唤醒未屏蔽
28:16	Reserved			
15	EM15	RW	0	EXTI line15 作为事件唤醒 CPU 屏蔽控制。 0：事件唤醒屏蔽 1：事件唤醒未屏蔽
14	EM14	RW	0	EXTI line14 作为事件唤醒 CPU 屏蔽控制。 0：事件唤醒屏蔽 1：事件唤醒未屏蔽
13	EM13	RW	0	EXTI line13 作为事件唤醒 CPU 屏蔽控制。 0：事件唤醒屏蔽 1：事件唤醒未屏蔽
12	EM12	RW	0	EXTI line12 作为事件唤醒 CPU 屏蔽控制。 0：事件唤醒屏蔽 1：事件唤醒未屏蔽
11	EM11	RW	0	EXTI line11 作为事件唤醒 CPU 屏蔽控制。 0：事件唤醒屏蔽 1：事件唤醒未屏蔽
10	EM10	RW	0	EXTI line10 作为事件唤醒 CPU 屏蔽控制。 0：事件唤醒屏蔽 1：事件唤醒未屏蔽
9	EM9	RW	0	EXTI line9 作为事件唤醒 CPU 屏蔽控制。 0：事件唤醒屏蔽

Bit	Name	R/W	Reset Value	Function
				1：事件唤醒未屏蔽
8	EM8	RW	0	EXTI line8 作为事件唤醒 CPU 屏蔽控制。 0：事件唤醒屏蔽 1：事件唤醒未屏蔽
7	EM7	RW	0	EXTI line7 作为事件唤醒 CPU 屏蔽控制。 0：事件唤醒屏蔽 1：事件唤醒未屏蔽
6	EM6	RW	0	EXTI line6 作为事件唤醒 CPU 屏蔽控制。 0：事件唤醒屏蔽 1：事件唤醒未屏蔽
5	EM5	RW	0	EXTI line5 作为事件唤醒 CPU 屏蔽控制。 0：事件唤醒屏蔽 1：事件唤醒未屏蔽
4	EM4	RW	0	EXTI line4 作为事件唤醒 CPU 屏蔽控制。 0：事件唤醒屏蔽 1：事件唤醒未屏蔽
3	EM3	RW	0	EXTI line3 作为事件唤醒 CPU 屏蔽控制。 0：事件唤醒屏蔽 1：事件唤醒未屏蔽
2	EM2	RW	0	EXTI line2 作为事件唤醒 CPU 屏蔽控制。 0：事件唤醒屏蔽 1：事件唤醒未屏蔽
1	EM1	RW	0	EXTI line1 作为事件唤醒 CPU 屏蔽控制。 0：事件唤醒屏蔽 1：事件唤醒未屏蔽
0	EM0	RW	0	EXTI line0 作为事件唤醒 CPU 屏蔽控制。 0：事件唤醒屏蔽 1：事件唤醒未屏蔽

11.3.10. EXTI 寄存器映像

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	EXTI_RTSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RT18	RT17	RT16	RT15	RT14	RT13	RT12	RT11	RT10	RT9	RT8	RT7	RT6	RT5	RT4	RT3	RT2	RT1	RT0
	Reset value														0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x004	EXTI_FTSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FT18	FT17	FT16	FT15	FT14	FT13	FT12	FT11	FT10	FT9	FT8	FT7	FT6	FT5	FT4	FT3	FT2	FT1	FT0
	Reset value														0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x008	EXTI_SWIER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWI15	SWI14	SWI13	SWI12	SWI11	SWI10	SWI9	SWI8	SWI7	SWI6	SWI5	SWI4	SWI3	SWI2	SWI1	SWI0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00C	EXTI_PR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PR15	PR14	PR13	PR12	PR11	PR10	PR9	PR8	PR7	PR6	PR5	PR4	PR3	PR2	PR1	PR0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10-0x	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
5C																																	
0x60	EXTI_EXTICR1	Res.	Res.	Res.	Res.	Res.	Res.	EXTI3[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	EXTI2[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	EXTI1[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	EXTI0[1:0]	
	Reset value							0	0							0	0							0	0							0	0
0x64	EXTI_EXTICR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EXTI7	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EXTI6	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EXTI5	Res.	Res.	Res.	Res.	Res.	Res.	EXTI4[1:0]	
	Reset value								0								0								0							0	0
0x68	EXTI_EXTICR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EXTI8
	Reset value																																0
0x6C-0x7C	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0x80	EXTI_IMR	Res.	Res.	IM29	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IM15	IM14	IM13	IM12	IM11	IM10	IM9	IM8	IM7	IM6	IM5	IM4	IM3	IM2	IM1	IM0
	Reset value			1														0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x84	EXTI_EMR	Res.	Res.	EM29	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EM15	EM14	EM13	EM12	EM11	EM10	EM9	EM8	EM7	EM6	EM5	EM4	EM3	EM2	EM1	EM0
	Reset value			0														0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

12. 循环冗余校验(CRC)

12.1. 简介

根据生成多项式，CRC 计算单元将输入的 32 位数据，运算产生一个 CRC 结果。

12.2. CRC 主要特点

- 使用 CRC-32（以太网）多项式：0x4C11DB7

$X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + X^8 + X^7 + X^5 + X^4 + X^2 + X + 1$

- 支持 32 位数据输入
- 单个输入/输出 32 数据和结果输出共用一个寄存器
- General purpose 的 8 位寄存器（可被用作临时存储）
- 计算时间：32bits 数据 4 个 AHB 时钟

12.3. CRC 功能描述

12.3.1. CRC 框图

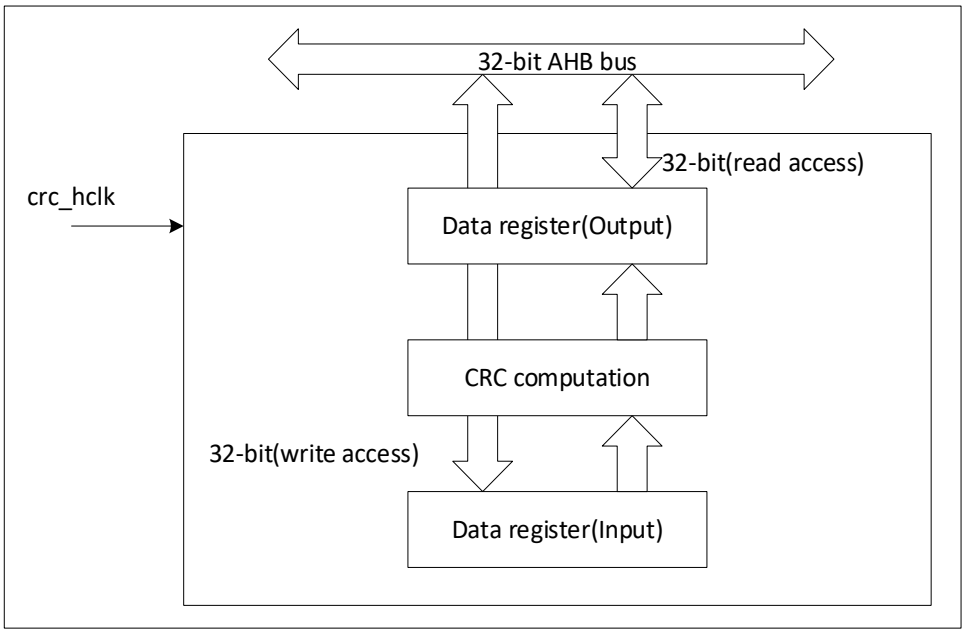


图 12-1 CRC 计算单元框图

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)。

当 CRC 正在计算时，写操作会被阻止，直到 CRC 计算结束。

可以通过设置寄存器 CRC_CR 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。

12.4. CRC 寄存器

12.4.1. 数据寄存器 (CRC_DR)

Address offset:0x00

Reset value:0xFFFF FFFF

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

Bit	Name	R/W	Reset Value	Function
31:0	DR	RW	32'hFFFFFFFF	数据寄存器。 当写新数据时，作为输入寄存器。当被读时，保持之前 CRC 计算结果。

12.4.2. 独立数据寄存器(CRC_IDR)

Address offset:0x04

Reset value:0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31:8	Reserved		-	
7:0	IDR[7:0]	RW	0	通用 8bit 数据寄存器 这些位用作一个字节的临时存储。该寄存器不会被 CRC_CR 寄存器的 RESET 位复位。

12.4.3. 控制寄存器(CRC_CR)

Address offset:0x08

Reset value:0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31:1	Reserved		-	
0	RESET		0	该位被软件置位，用来复位 CRC 计算单元。该位只能被置位，由硬件自动清零。

12.4.4. CRC 寄存器映像

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	CRC_DR	DR[31:0]																															
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x04	CRC_IDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IDR[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x08	CRC_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RESET
	Reset value																																0

13. 模拟/数字转换(ADC)

13.1. 简介

芯片具有 1 个 12 位的 SARADC（successive approximation analog-to-digital converter）。该模块共有 11 个要被测量的通道，包括 9 个外部通道和 2 个内部通道。

各通道的转换模式可以设定为单次、连续、扫描、不连续模式。转换结果存储在左对齐或者右对齐的 16 位数据寄存器中。

模拟 watchdog 允许应用检测是否输入电压超出了用户定义的高或者低阈值。

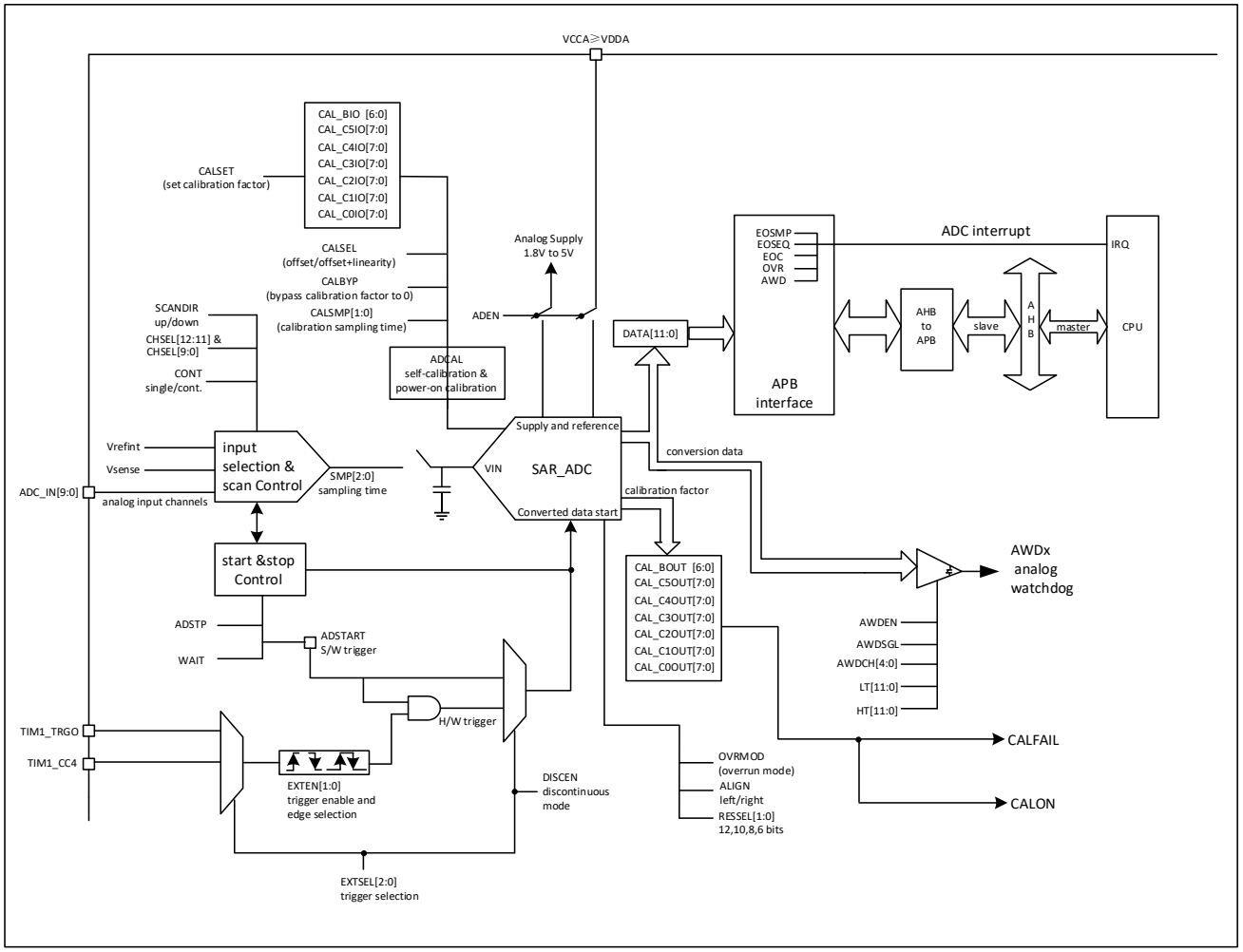
ADC 实现了在低频率下运行，可获得很低的功耗。

13.2. ADC 主要特性

- 高性能
  - 12bit、10bit、8bit 和 6bit 分辨率可配置
  - ADC 转换时间：ADC 转换时间：1.33 us@12bit（0.75 MSPS，3.5 T SMP，f_{ADC_CLK}=12 MHz）
  - 自校准
  - 可编程的采样时间
  - 可编程的数据对齐模式
- 低功耗
  - 为低功耗操作，降低 PCLK 频率，而仍然维持合适的 ADC 性能
  - 等待模式：防止以低频 PCLK 运行产生溢出
- 模拟输入通道
  - 9 个外部模拟输入通道：PA[7:0]和 PB[1]
  - 1 个内部 temperature sensor 通道
  - 1 个内部参考电压通道（V_{REFINT}）
- 转换操作启动可以通过
  - 软件启动
  - 可配置极性的硬件启动（TIM1 或者 GPIO）
- 转换模式
  - 单次模式(single mode)：可以转换 1 个单通道或者可以扫描一系列通道
  - 连续模式(continuous mode)：连续转换被选择的通道
  - 不连续模式(discontinuous mode)：每次触发，转换被选择的通道 1 次
- 中断产生
  - 在采样结束
  - 在转换结束
  - 在连续转换结束
  - 模拟看门狗事件
  - 溢出事件
- 模拟看门狗

13.3. ADC 功能描述

13.3.1. ADC 框图



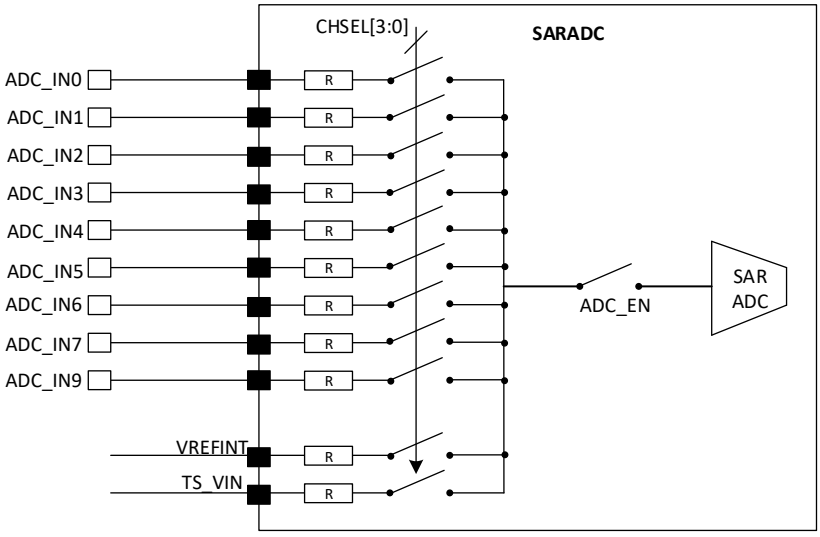


图 13-1 ADC channel with analog switch

13.3.2. 校准 (ADCAL)

该 ADC 具有校准功能。在校准期间，ADC 计算一个用于 ADC 内部的校准因子（ADC 断电后丢失）。在 ADC 校准期间、未完成校准前，应用不能使用 ADC 模块。

在使用 ADC 转换前，要进行校准操作。校准用于消除芯片和芯片之间的，由于工艺变化引起的 offset error。

校准操作包括上电校准及软件校准。

ADC 上电校准

上电后硬件会自动进行 ADC 校准。

ADC 软件校准

软件设置 ADCAL=1 可启动校准，校准只能在 ADC 未使能时 (ADEN=0)启动，且仅支持选择系统时钟作为 ADC 的时钟。当校准完成后，ADCAL 被硬件清 0。

当 ADC 的工作条件发生改变时（VCC 改变是 ADC offset 偏移的主要因素，温度改变次之），推荐进行再次校准操作。

校准的软件操作过程：

- 确认 ADEN=0、CKMODE 选择系统时钟
- 设置 ADCAL=1
- 等待到 ADCAL=0

13.3.3. ADC 开关控制 (ADEN)

芯片上电复位后，ADC 模块不使能，且处于断电模式 (ADEN=0)。

ADCEN 位用于控制位开启或关闭 ADC。

以下为启用 ADC 的过程：

1. 配 ADC_CR 寄存器的 ADEN 位为 1

ADC 转换也由设置 ADSRART 来启动或(如果触发启动)由外部触发事件来触发启动开启。

以下为禁用 ADC 的过程：

检查 ADC_CR 寄存器中的 ADSTART 是否为 0 以确保 ADC 不在转换过程中。若需要，可对 ADC_CR 寄存器中的 ADSTP 置 1 来停止正在进行的 ADC 转换，并等待 ADSTP 被硬件清 0(清 0 表示转换停止完成)。

警告：在 ADCAL 被硬件清除之后的 4 个 ADC 时钟期间并且 ADCAL=1 时，ADEN 位不能被置 1。

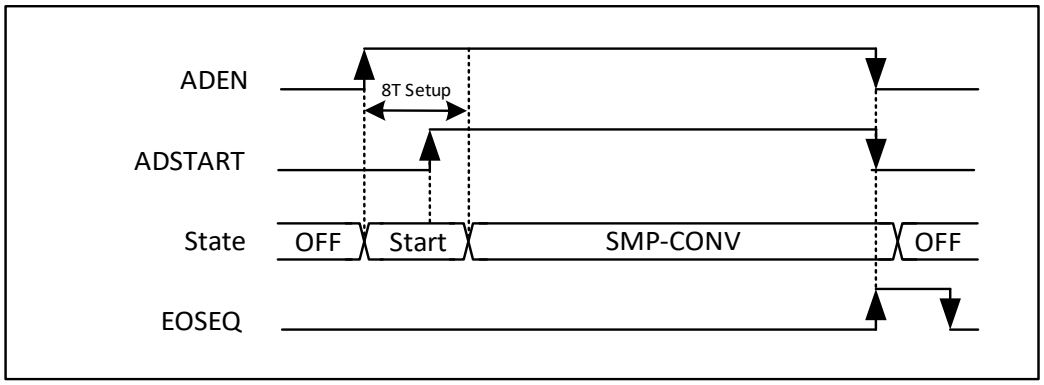


图 13-2 Enabling/disabling the ADC

13.3.4. ADC 时钟

ADC 具有双时钟域架构，ADC 时钟(ADC_CLK) 独立 APB 时钟(PCLK)。ADC_CLK 可由两种可能的时钟源产生。

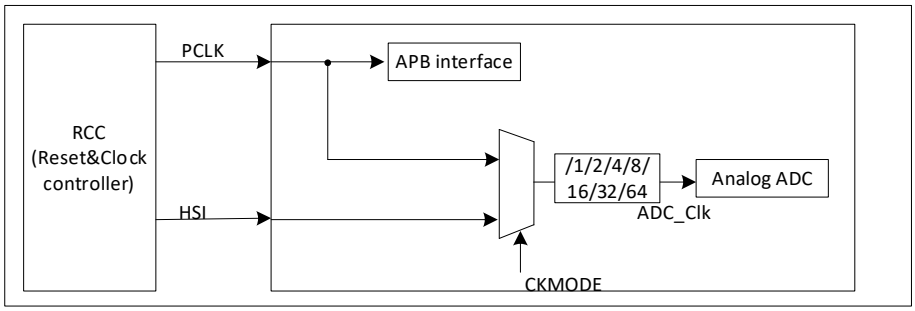


图 13-3 ADC clock scheme

表 13-1 触发器和转换开始之间的延迟

ADC clock source	CKMODE[3:0]	分频系数	Latency between the trigger event and the start of conversion（T 为时钟周期）
PCLK	0000	1	0
	0001	2	0
	0010	4	0
	0011	8	0
	0100	16	0
	0101	32	0
	0110	64	0
	0111	/	/
HSI	1000	1	0
	1001	2	0
	1010	4	0
	1011	8	0
	1100	16	0
	1101	32	0
	1110	64	0
	1111	/	/

13.3.5. 配置 ADC

软件必须在 ADC 禁止(ADEN 必须为 0) 的情况下改写 ADC_CR 寄存器中的 ADCAL 和 ADEN 位。软件必须在 ADC 开启(ADEN=1)的情况下改写 ADC_CR 寄存器中的 ADSTART。

对于以下这些 ADC_IER、ADC_CFGRi、ADC_SMPR、ADC_TR、ADC_CHSELR 和 ADC_CCR 寄存器，软件必须在无转换期间 (ADSTART = 0) 的情况下才能进行改写。

软件必须在 ADC 开启且 ADSTART = 1 的情况下改写 ADC_CR 寄存器中的 ADSTP 位。

13.3.6. 通道选择 (CHSEL, SCANDIR)

共有 11 路复用通道：

● 9 个从 GPIO 引脚引入的模拟输入 (ADC_IN0...ADC_IN9)

● 2 个内部模拟输入(温度传感、内部参考电压)

ADC 可以转换一个单一通道或自动扫描一个序列通道。

被转换的通道序列必须在通道选择寄存器 ADC_CHSELR 中编程选择：每个模拟输入通道有专门的一位选择位。

ADC 扫描的通道顺序由 ADC_CFGR1 中 SCANDIR 位的配置来决定：

● SCANDIR=0: 向前扫描: 从通道 0 到通道 11

● SCANDIR=1: 回退扫描：从通道 11 到通道 0

温度传感器,VREFINT 内部通道

温度传感连接到 ADC_IN10（TS_VIN） 通道，内部参考电压连接到 ADC1_IN11 通道（VREFINT）。

13.3.7. 可编程采样时间 (SMP)

在启动 ADC 转换之前，ADC 需要在被测电压源和内嵌采样电容间建立一个直接连接。采样时间必须足够长以便输入电压源对内嵌电容充电到输入电压的水平。

可编程采样时间根据输入电压的输入阻抗来调整转换速度。

ADC 采样输入电压所用的 ADC 时钟个数可用 ADC_SMPR 寄存器中的 SMP[2:0]位来进行修改。可编程采样时间对所有通道都通用。如有应用需求，则可用软件改变和适应不同通道间的采样时间。

总转换时间计算如下：

t_{CONV} = 采样时间 + (转换分辨率+0.5) x ADC 时钟周期

EOSMP 标志位用来表明采样阶段的结束。

13.3.8. 单次转换模式 (CONT=0, DISCEN=0)

单次转换模式下，ADC 执行一次序列转换，转换所有被选的通道。当 ADC_CFGR1 寄存器中的 CONT=0，DISCEN=0 时，ADC 为单次转换模式。

ADC 转换可由下述两种方法启动：

- 在 ADC_CR 寄存器中设置 ADSTART 位
- 硬件触发事件

在序列通道的转换期间，每次转换完成后：

- 转换的数据结果存放到 16 位寄存器 ADC_DR 中。
- EOC(转换结束标志)标志置位
- 若 EOCIE 位置位则产生一个中断。

通道序列转换完成后：

- EOSEQ(序列结束) 标志置位
- 若 EOSIE 位置位则产生一个中断

转换结束后，ADC 停止直到新的触发事件或 ADSTART 重新置位。

注：若转换单一通道，则可编程一个长度为 1 的一个转换序列。

13.3.9. 连续转换模式 (CONT=1)

在连续转换模式中，当软件或硬件触发事件产生，ADC 执行一个序列转换。转换所有的通道一次且自动重新开始执行相同的序列转换。当寄存器 ADC_CFGR1 中的 CONT=1 时， ADC 选择为连续转换模式。ADC 转换可由下述两种方法启动：

- 在 ADC_CR 寄存器中设置 ADSTART 位
- 硬件触发事件

  在序列通道的转换期间，每次转换完成后：
- 转换的数据结果存放到 16 位寄存器 ADC_DR 中
- EOC (转换结束标志)标志置位
- 若 EOCIE 位置位则产生一个中断。

通道序列转换完成后：

- EOSEQ(序列结束)标志置位
- 若 EOSEQIE 位置位则产生一个中断

一次序列转换结束后，ADC 立即重新转换相同的序列通道。

注：若转换单一通道，则可编程一个长度为 1 的一个转换序列。

ADC 不能同时处于 discontinuous 转换模式和 continuous 转换模式，在这种情况下 (DISCEN=1，CONT=1)，其表现为单次转换模式。

13.3.10. 非连续转换模式 (DISCEN=1)

该模式由设置 ADC_CFGR1 寄存器中的 DISCEN 位来开启。

在这个模式 (DISCEN=1)下，需要硬件或软件的触发事件去启动定义在一个序列中的每次转换。

相反，DISCEN=0 时，一个硬件或软件的触发事件，就可以启动定义在一个序列中的所有转换。

例如：

DISCEN=1, 需要转换的通道为：0, 3, 7, 10

— 1st 触发：通道 0 被转换且一个 EOC 事件产生

— 2nd 触发：通道 3 被转换且一个 EOC 事件产生

— 3rd 触发：通道 7 被转换且一个 EOC 事件产生

— 4th 触发：通道 10 被转换且产生 EOC 和 EOSEQ 事件

— 5th 触发：通道 0 被转换且一个 EOC 事件产生

— 6th 触发：通道 3 被转换且一个 EOC 事件产生

— ...

DISCEN=0, 需要转换的通道为：0, 3, 7, 10

— 1st 触发：整个完整的序列转换，依次为通道 0, 3, 7 和 10。

每次转换完成，产生一个 EOC 事件，转换到最后一个通道，除产生 EOC 外，还产生一个 EOSEQ 事件。

— 任何触发事件都会重新开始完整的序列转换。

注：让 ADC 同时处于连续模式和连续转换模式是不可能的事情， 在这种情况下(DISCEN =1，CONT=1)，其表现为单次转换模式。

13.3.11. 启动 ADC 转换 (ADSTART)

软件用设置 ADSTART=1 启动 ADC 转换。

当 ADSTART 设置，则转换：

- 当 EXTEN=0x0(软件触发) 时，立即开始
- 当 if EXTEN ≠ 0x0 时，在下一个所选择的硬件触发有效边沿开始

ADSTART 位也用于说明目前 ADC 转换操作是否正在进行。当 ADSTART=0 时，可重新配置 ADC，说明此时 ADC 处于空闲。

ADSTART 位可由硬件清除。

- 单次转换模式由软件触发 (CONT=0，EXTSEL=0x0)

— 在序列转换结束后 (EOSEQ=1)

- Discontinuous 转换模式由软件触发 (CONT=0，DISCEN=1, EXTSEL=0x0)

- 在转换结束后(EOC=1)

- 在所有的情况下(CONT=X，EXTSEL=X)

— 在软件调用并执行 ADSTP 过程后

注：在连续模式 (CONT=1) 下，ADSTART 位不能由 EOSEQ 引发的硬件清除，其原因是自动重新开始序列转换。当硬件触发选择为单次转换模式 (CONT=0 and EXTSEL =0x01), 则当 EOSEQ 标志设置后，ADSTART 不会被硬件清 0。这就避免了需要软件重新设置 ADSTART 位且要确保无硬件触发事件错过。

13.3.12. 转换时间

转换所用的时间由启动转换时间和与转换分辨率有关的逐次逼近时间组成。

$t_{ADC} = t_{SMPL} + t_{SAR} = [\ 3.5_{|min} + 12.5_{|12bit}] * t_{ADC\_CLK}$

$t_{ADC} = t_{SMPL} + t_{SAR} = \ 291.67ns_{|min} + 1041.67_{|12bit} = 1.33\ \mu s_{|min}$ (for f_{ADC_CLK} = 12 MHz)

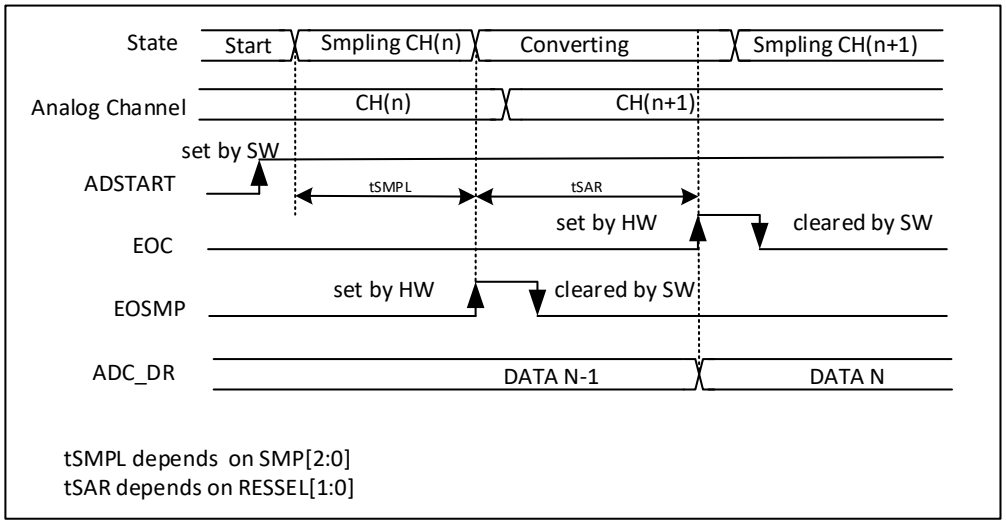


图 13-4 analog to digital conversion timing

13.3.13. 停止进行中的转换(ADSTP)

用软件设置 ADC_CR 寄存器中的 ADSTP=1 可以停上当前正在进行的转换，复位 ADC 的操作并让 ADC 进入空闲状态，为下次转换作好准备。

当 ADSTP 由软件设置为 1，任何当前的转换中止且转换结果丢弃(ADC_DR 寄存器不用当前的转换值进行更新)。

扫描序列也被中止并复位 (即重新启动 ADC 时会用新的序列进行转换)

一旦结束该过程 ADSTP 和 ADSTART 位都由硬件清 0。

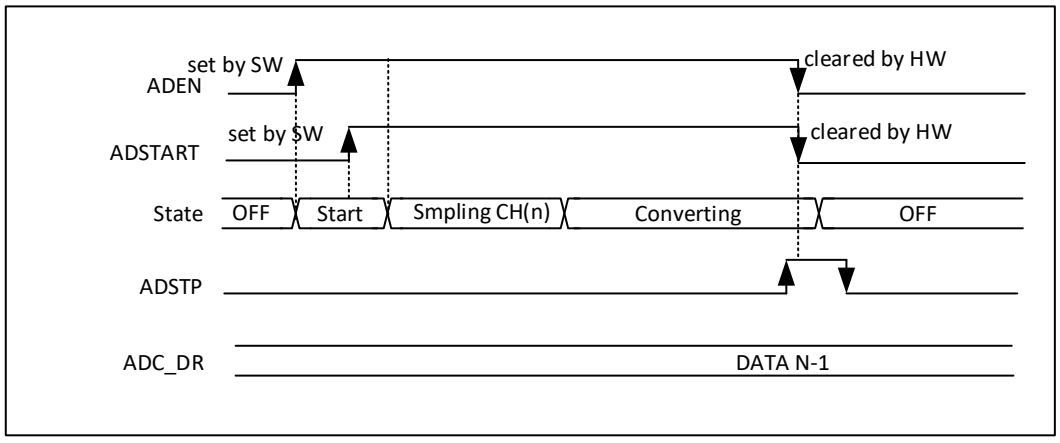


图 13-5 Stop timing

13.4. 外部触发转换和触发极性(EXTSEL, EXTEN)

一次转换或一个序列的转换可由软件或外部事件(例如：定时器捕、输入引脚) 触发。 若 EXTEN[1:0] ≠ “00”，则外部事件在其所选择的极性上可以用于触发转换。当软件设置 ADSTART=1 时，触发选择有效。

当正在进行 ADC 转换时，任何硬件触发都会被忽略。

当 ADSTART=0 时，任何硬件触发都会忽略。

Source	EXTEN[1:0]
触发检测禁止	00
在上升沿检测	01
在下降沿检测	10
在上升和下降沿检测	11

注：在转换时外部触发极性不能改变。EXTSEL[2:0] 控制位用于选择可触发转换的事件。

下表给出了规则转换可能的外部触发。软件源触发事件可由设置 ADC_CR 寄存器中的 ADSTART 位来产生。

表 13-2 外部触发

Name	source	EXTSEL[2:0]
EXT0	TIM1_TRGO	000
EXT1	TIM1_CC4	001
EXT2	Reserved	010
EXT3	Reserved	011
EXT4	Reserved	100
EXT5	Reserved	101
EXT6	Reserved	110
EXT7	Reserved	111

注： 在转换时外部触发源不能改变。

13.4.1. 快速转换模式

用降低转换分辨率来获取更快的转换时间 (t_{SAR}) 是可行的。转换分辨率可通过设置 ADC_CFGR1 寄存器中的 RES[1:0] 来配置为 12/10/8/6 位模式。当应用不需要高精度数据时，可用低的转换分辨率来加快转换时间。转换结果也是 12 位宽度且低位补 0。

分辨率模式减少逐次逼近的转换时间，如下表所示：

RESSEL [1:0]	t_{SAR} (ADC 时钟周期)	t_{SAR}(ns) @ fADC = 24MHz	t_{SMP} (ADC 时钟周期)	t_{ADC}(t_{SMP} = 3.5) (ADC 时钟周期)	t_{CONV}(ns) @ f_{ADC} = 24MHz
12	12.5	521ns	3.5	16	667ns
10	10.5	438ns	3.5	14	583ns
8	8.5	396ns	3.5	12	500ns
6	6.5	271ns	3.5	10	417ns

13.4.2. 转换结束/采样结束

ADC 通知应用每次转换结束 (EOC) 事件。

一旦在 ADC_DR 寄存器中的一个转换数据有效后，ADC 在 ADC_ISR 寄存器中设置 EOC 标志表明转换完成。当 ADC_IER 中的 EOCIE 置为 1 时，则会产生一个 EOC 中断。EOC 标志由软件写 1 清除或读 ADC_DR 寄存器来清除。

ADC 同样在 ADC_ISR 寄存器中给出采样阶段结束标志 EOSMP。EOSMP 标志可写 1 请除。当在 ADC_IER 寄存器中的 EOSMPIE 置为 1 后，则会产生一个 EOSMP 中断。

13.4.3. 序列转换结束 (EOSEQ flag)

ADC 通知应用每次序列转换结束 (EOSEQ) 事件。

一旦一个转换序列的最后一个通道转换数据有效后，ADC 在 ADC_ISR 寄存器中设置 EOSEQ 标志。当 ADC_IER 中的 EOSEQIE 位置 1 时，则会产生中断。EOSEQ 标志由软件写 1 清 0。

13.4.4. 采样时间图

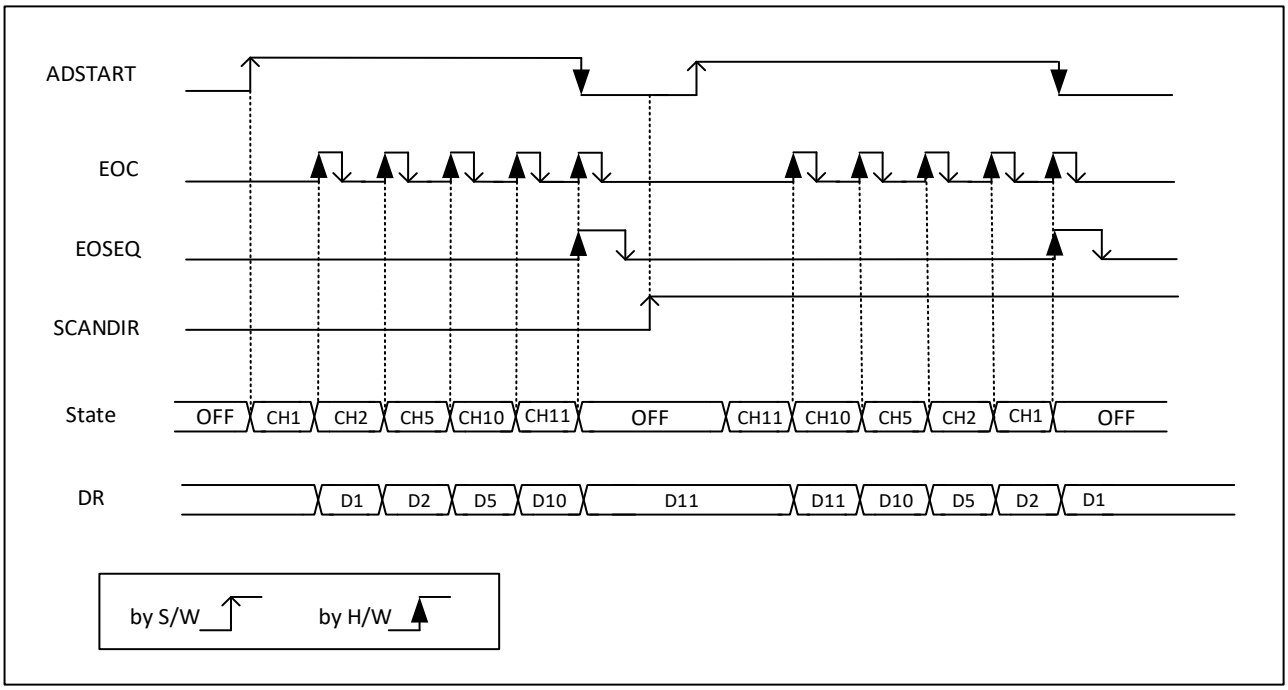


图 13-6 序列的单次转换, 软件触发

1. EXTEN=0x0, CONT=0
2. CHSEL=0x20601, WAIT=0, AUTOFF=0

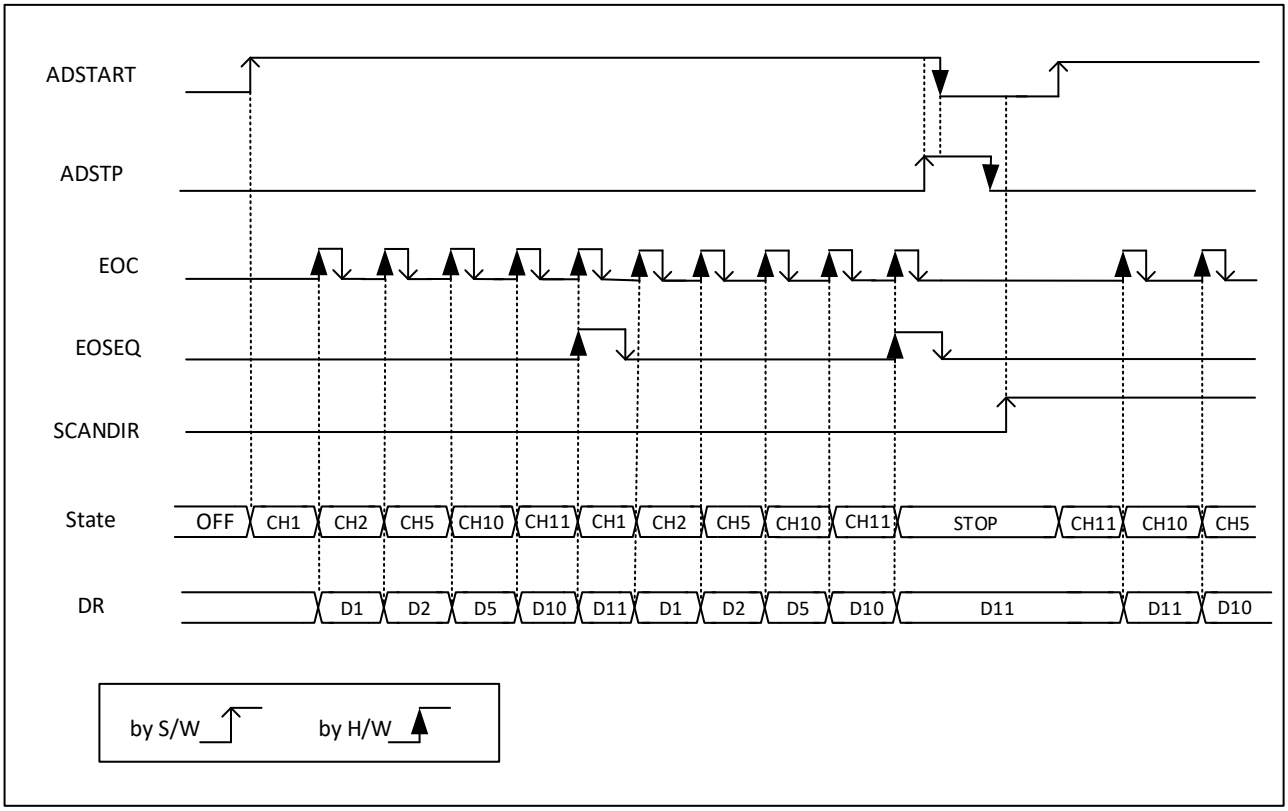


图 13-7 序列的连续转换, 软件触发

1. EXTEN=0x0, CONT=1,
2. CHSEL=0x20601, WAIT=0, AUTOFF=0

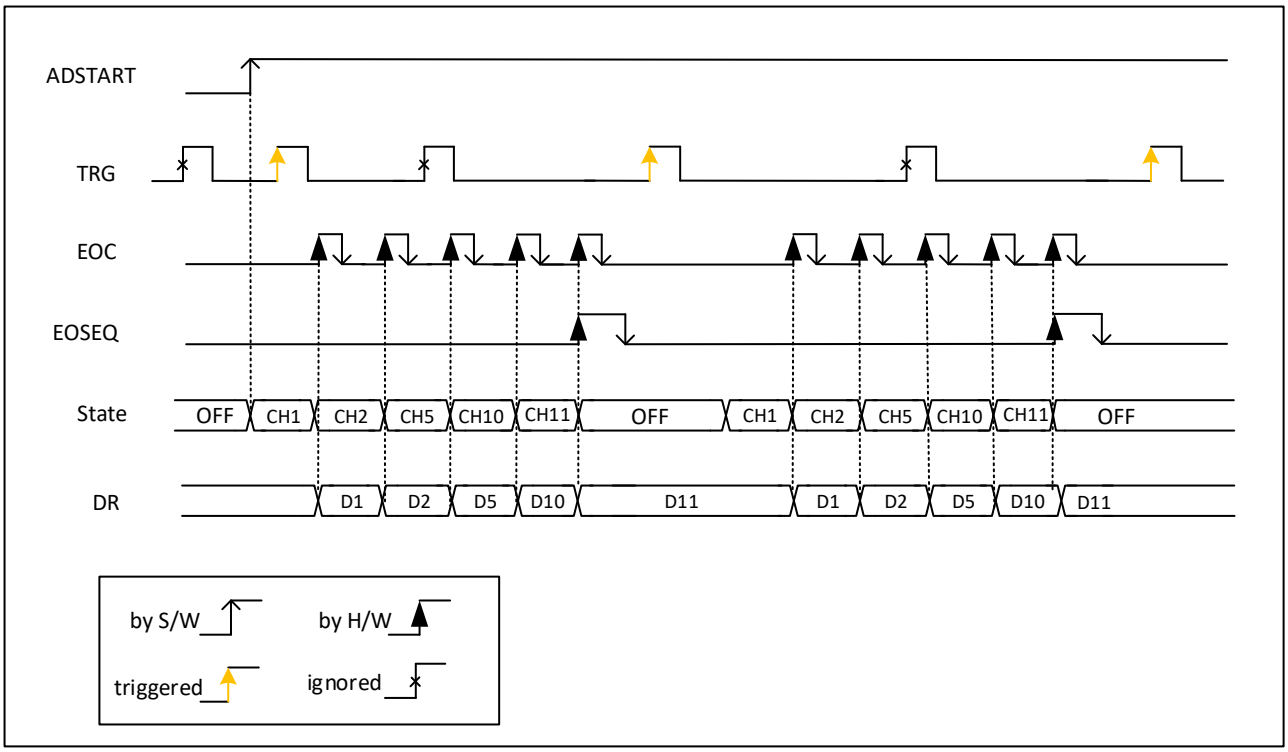


图 13-8 序列的单次转换, 硬件触发

1. EXTSEL=TRGx, EXTEN=0x1 (上升沿), CONT=0
2. CHSEL=0xF, SCANDIR=0, AUTDLY=0, AUTOFF=0

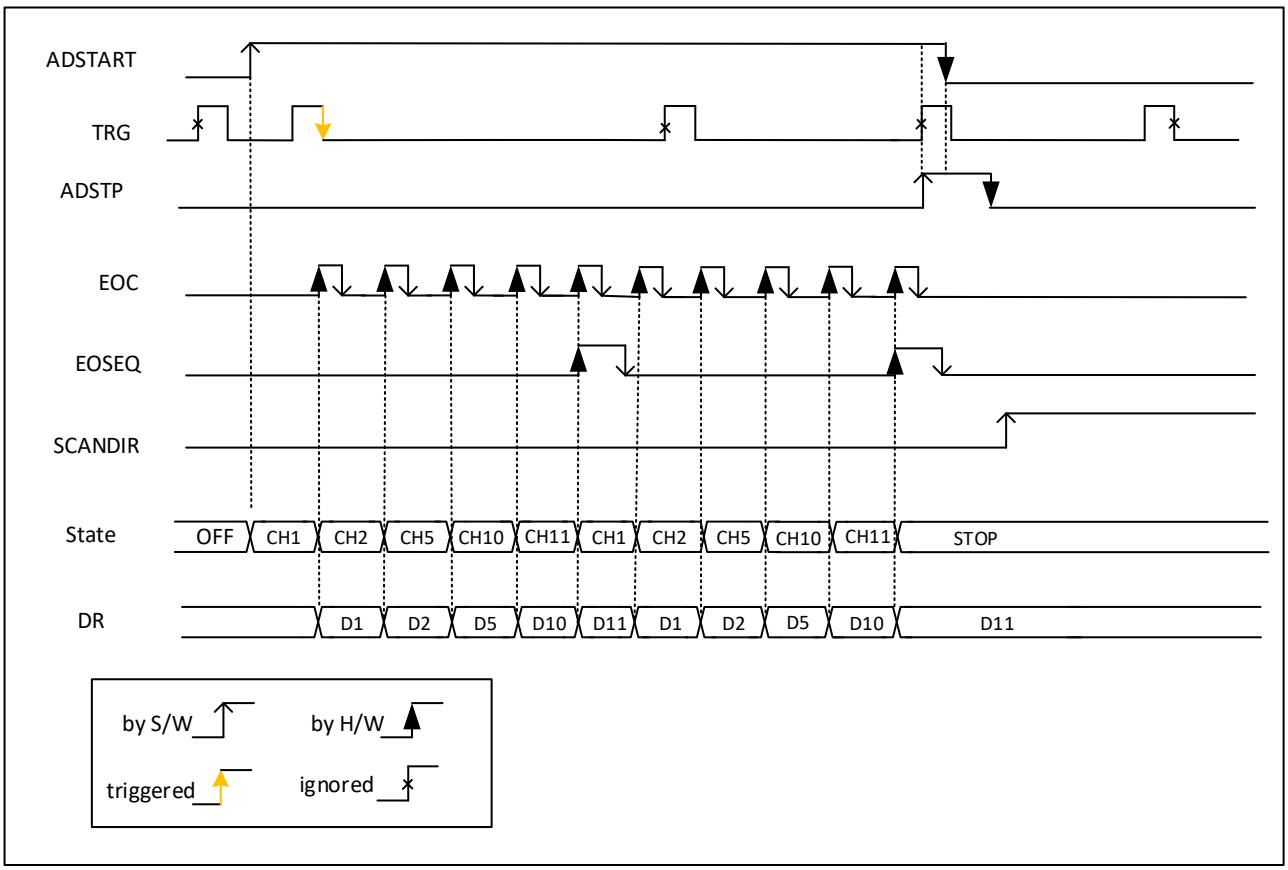


图 13-9 序列的连续转换，硬件触发

1. EXTSEL=TRGx, EXTEN=0x2 (下降沿), CONT=1
2. CHSEL=0xF, SCANDIR=0, WAIT=0, AUTOFF=0

13.5. 数据管理

13.5.1. 数据寄存器和数据对齐(ADC_DR, ALIGN)

在每次转换结束(当 EOC 事件产生时)，转换的结果数据被存放到 16 位宽 ADC_DR 数据寄存器中。

ADC_DR 数据格式与所配置的数据对齐和转换分辨率有关。ADC_CFGR1 寄存器中的 ALIGN 位用于选择数据存储的对齐方式，数据可选为右对齐 (ALIGN=0) 或左对齐(ALIGN=1)。

ALIGN	RESSEL	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0X0	0X0				DATA[11:0]											
	0X1	0X0				DATA[9:0]										0X0	
	0X2	0X0				DATA[7:0]								0x0			
	0X3	0X0				DATA[6:0]						0X0					
1	0X0	DATA[11:0]												0X0			
	0X1	DATA[9:0]										0X0		0X0			
	0X2	DATA[7:0]								0x0				0X0			
	0X3	DATA[6:0]						0X0						0X0			

13.5.2. ADC 过载 (OVR, OVRMOD)

ADC 过冲标志(OVR) 是指一个缓冲区过冲事件，当转换好的数据未被 CPU 及时读取时，另一个转换数据已经有效时，就发生了 ADC 过冲。

若 EOC 还为‘1’的情况下，这时一个新的转换已经完成，那么 CPU 就会在 ADC_ISR 寄存器中的 OVR 标志被置位，表明 ADC 过冲。当 ADC_IER 寄存器中的 OVRIE 置位时，产生一个 ADC 过冲中断。

当过冲事件发生时，ADC 会继续操作并且继续转换除非软件决定停止并复位这个序列转换，可用软件设置 ADC_CR 寄存器中的 ADSTP 为 1 来停止 ADC 转换 OVR 标志可用软件写 1 清除。

当发生过冲事件时，可通过对 ADC_CFGR1 寄存器中的 OVRMOD 位来设置 ADC 数据寄存器中的数据是被保持还是被覆盖：

- OVRMOD=0

— 一个过冲事件保持数据寄存器的值防止被覆盖：之前的数据被保持，新的转换数据丢弃。若 OVR 保持为 1，则后续的转换会被执行但结果都被丢弃。

- OVRMOD=1

— 用最近一次的转换结果覆盖数据寄存器，先前未读的数据丢失。若 OVR 保持为 1，则后续的转换被执行且 ADC_DR 寄存器存放着最新转换的结果值。

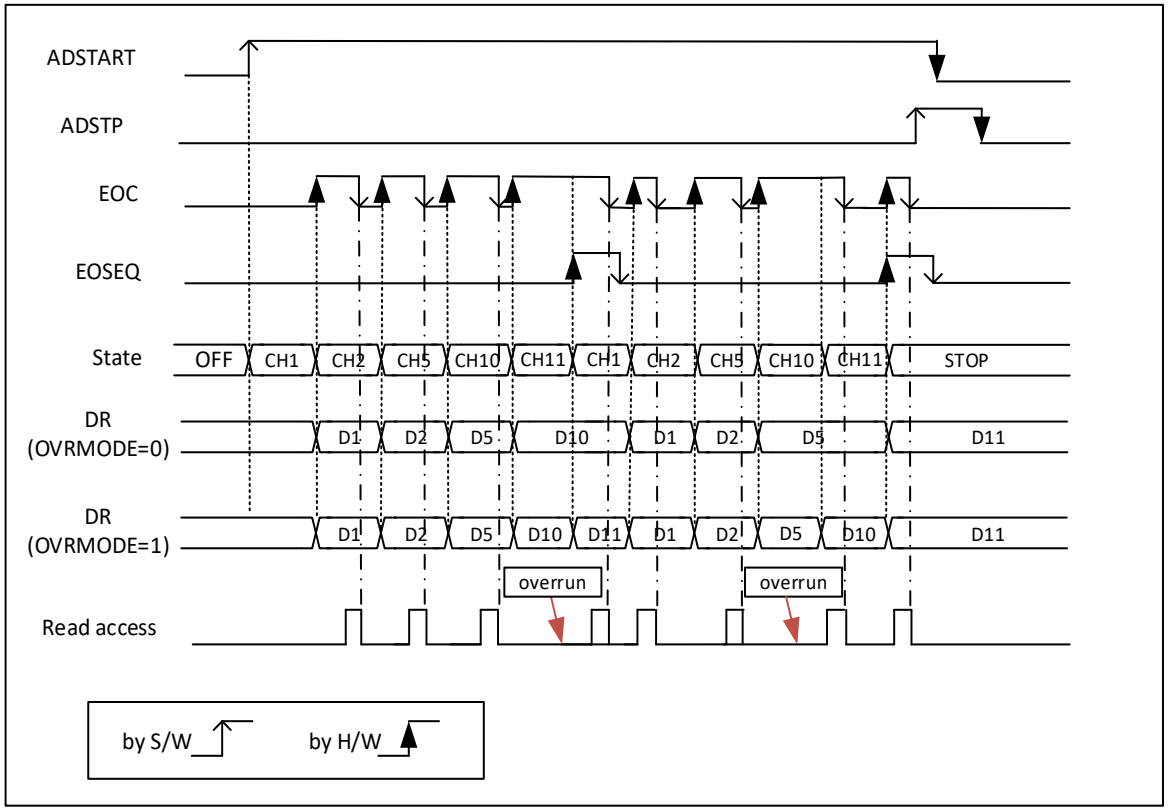


图 13-10 过载

13.5.3. 管理转换序列

若 ADC 的转换足够慢，转换序列可由软件来控制。这种情况下，软件应用 EOC 标志及其关联的中断去处理每个转换数据。当每次转换结束时，在 ADC_ISR 寄存器中的 EOC 位置位，此时可读 ADC_DR 寄存器的转换值。ADC_CFGR1 寄存器中的 OVRMOD 位可配为 0 来管理过冲事件。

13.5.4. 进行转换

存在着转换一个或多个通道且不用每次转换结果都要读取的应用。这种情况下， OVRMOD 位必须置为 1 且软件应忽略 OVR 标志。当 OVRMOD=1 时，过冲事件不能阻止 ADC 继续转换且 ADC_DR 寄存器中的数据一直为最后转换的数据。

13.6. 低功耗特性

13.6.1. 自动延迟转换模式

自动延迟转换模式可用于在低速运行时简化软件以及优化应用程序的性能，当然在这种模式下不容易产生 ADC 过冲的情况。

当在 ADC_CFGR1 寄存器中设置 WAIT 为 1 时，一个新的转换只有在刚才的 ADC 数据处理完后(比如 ADC_DR 寄存器中的数据被读取或 EOC 标志已被清除)才开始。这是一种自适应 ADC 速度和自适应系统读取 ADC 数据速度的方法。

注：当正在转换中或自动延迟产生的情况下，任一硬件产生的触发都会被忽略。

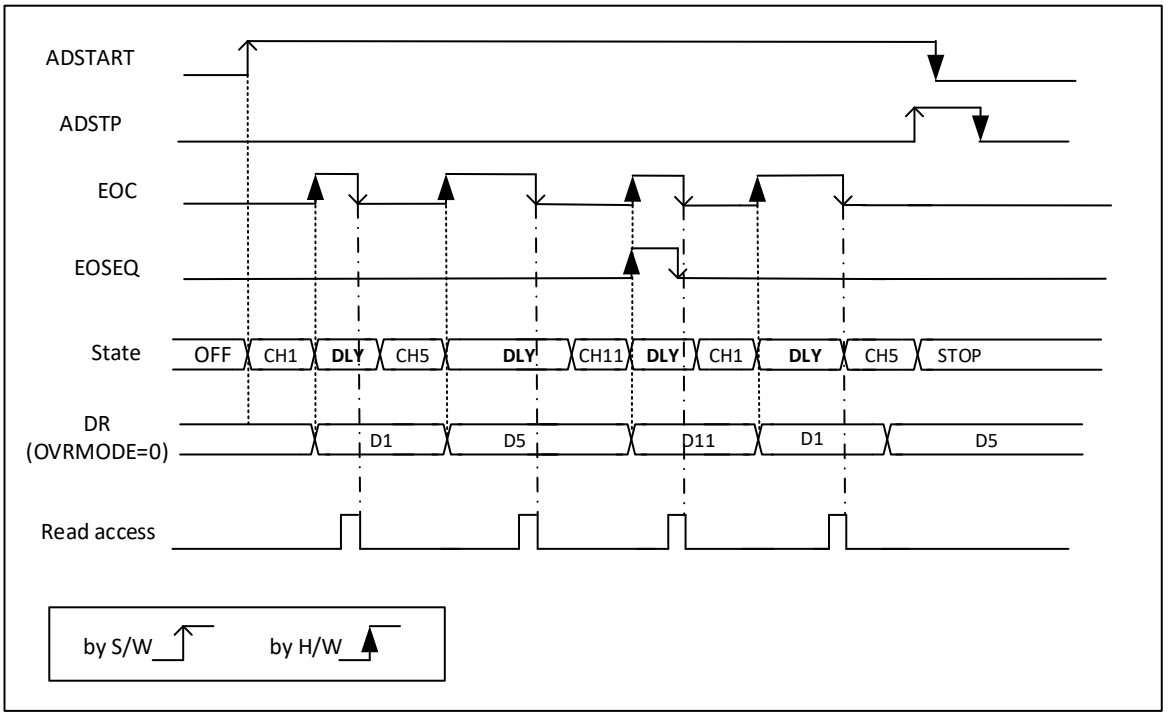


图 13-11 自动延迟转换模式

1. EXTEN=0x0, CONT=1
2. CHSEL=0x3, SCANDIR=0, AUTDLY=1, AUTOFF=0

13.7. 模拟看门狗

模拟看门狗的功能由在 ADC_CFGR1 寄存器中的 AWDEN 位置位来开启。它可用于监控所选的单一通道或所有使能通道所配置电压范围(窗口)。

如果模拟电压转换由 ADC 低于低阈值或高于高阈值时，AWD 模拟看门狗的状态位被置位。阈值被编程到最多具有 12 位有效数据的 ADC_HTR 和 ADC_LTR 16 位寄存器中。模拟看门狗中断可用设置 ADC_IER 寄存器中的 AWDIE 位来使能。AWD 标志位可用软件写 1 来清除。当转换的数据分辨率小于 12 位 (由 DRES[1:0] 位来决定), 被编程阀值的低位必须保持清零，因为内部转换数据的比较都是按左对齐全 12 位的方式进行比较。

表 13-3 模拟看门狗比较

Resolution bits	模拟看门狗比较：		说明
	原始转换数据，左对齐	阈值	
00: 12-bit	DATA[11:0]	LT[11:0] and HT[11:0]	
01: 10-bit	DATA[11:2],00	LT[11:0] and HT[11:0]	用户必须配置 LT[1:0]和 HT[1:0]为 00
10: 8-bit	DATA[11:4],0000	LT[11:0] and HT[11:0]	用户必须配置 LT[3:0]和 HT[3:0]为 0000
11: 6-bit	DATA[11:6],000000	LT[11:0] and HT[11:0]	用户必须配置 LT[5:0]和 HT[5:0]为 000000

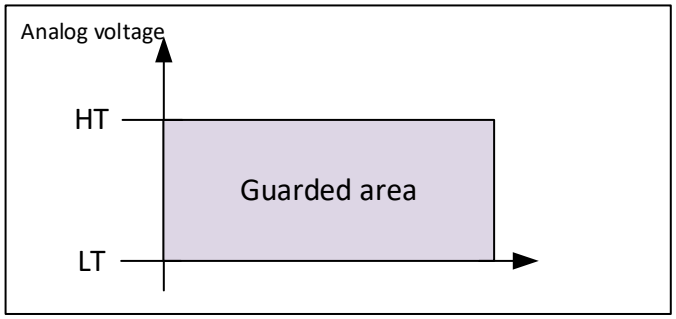


图 13-12 模拟看门狗保护区

表 13-4 模拟看门狗通道选择

Channels guarded by the analog watchdog	AWDSGL bit	AWDEN bit
None	x	0
All channels	0	1
Single channel	1	1

13.7.1. ADC_AWD_OUT 信号输出产生

模拟看门狗与一个内部硬件信号相关联，ADC_AWD_OUT 直接连接到片上定时器 TIM1 的 ETR 输入（外部触发）。

启用模拟看门狗时，将激活 ADC_AWD_OUT：

- 当经过 AWDCH 选择的通道转换超出程序阈值时，将设置 ADC_AWD_OUT。
- 在下一个经过 AWDCH 选择的通道的转换结束之后，ADC_AWD_OUT 在编程的阈值之内复位。如果下一个受保护的转换仍超出编程的阈值，则它将保持为 1。
- 禁用 ADC 时 ADC_AWD_OUT 也会复位。 请注意，停止转换（ADSTP 设置为 1）可能会清除 ADC_AWDx_OUT 状态。
- 未选择为模拟看门狗的通道，不影响 ADC_AWD_OUT 状态位。

AWD 标志由硬件设置并由软件复位：AWD 标志对 ADC_AWD_OUT 的生成没有影响（例如，如果软件未清除该标志，则 ADC_AWDx_OUT 可以切换，而 AWDx 标志保持为 1）。

ADC_AWD_OUT 信号由 PCLK 域生成。

AWD 比较在每次 ADC 转换结束时执行。

13.8. 温度传感器和内部参考电压

温度传感器可以用来测量器件的接点温度 (TJ)。

温度传感器内部连接到 ADC 输入通道，可用于转换传感器的电压值到一个数值。 温度传感器的采样时间必须大于 datasheet 给出的 Ts_temp 的最小值。当温度传感器没被使用时，传感器可以置于断电模式。

温度传感器输出电压跟温度成线性变化关系，但是跟工艺变量有关每颗芯片会有细微差别。为了提高这个准确度，每一颗的校准值会被产品测试单独给出并且保存在系统存储区域。

内部电压参考（VREFINT）提供一个稳定电压输出给 ADC 和比较器。

注： 必须设置 TSVREF 位来激活两个内部通道：温度传感器、VREFINT。

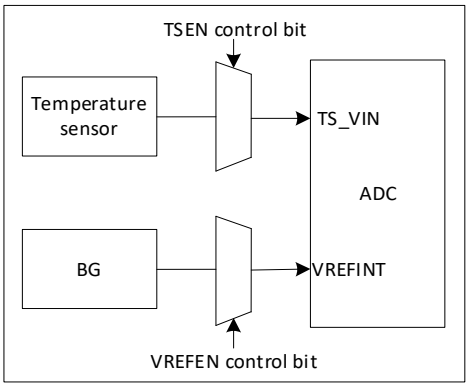


图 13-13 TS and VREFINT channel

读温度

如何用温度传感器：

1. 选择 ADC1_IN11 输入通道
2. 根据器件的规格书选择一个合适的采样时间
3. 在 ADC_CCR 寄存器中设置 TSEN 位用来唤醒从断电模式下的温度传感器
4. 用设置在 ADC_CR 寄存器中的 ADSTART 位 (也可用外部触发) 来启动 ADC 转换
5. 从 ADC_DR 寄存器中读取 VSENSE 转换数据
6. x6 版本(T_A = -40 ~ 85 ℃)产品用下列公式计数温度：

$$Temperature(in\ ℃) = \frac{85℃ - 30℃}{TS_{CAL2} - TS_{CAL1}} \times (TS_{DATA} - TS_{CAL1}) + 30℃$$

TS_{CAL2} 代表 85℃温度传感器的校准值，校准值存放地址：0x1FFF 0F18

TS_{CAL1} 代表 30℃温度传感器的校准值，校准值存放地址：0x1FFF 0F14

TS_{DATA} 是 ADC 转换的实际输出值

X7 版本(T_A = -40 ~ 105 ℃)产品用下列公式计数温度：

$$Temperature(in\ ℃) = \frac{105℃ - 30℃}{TS_{CAL2} - TS_{CAL1}} \times (TS_{DATA} - TS_{CAL1}) + 30℃$$

TS_{CAL2} 代表 105℃温度传感器的校准值，校准值存放地址：0x1FFF 0F18

TS_{CAL1} 代表 30℃温度传感器的校准值，校准值存放地址：0x1FFF 0F14

TS_{DATA} 是 ADC 转换的实际输出值

注： 传感器从断电模式下唤醒时到能正确输出 V_{SENSE} 要有一个启动时间，ADC 从上电后启动也有一个启动时间， 若要减少这个延时， 则需要同时设置 ADEN 和 TSEN 位。

利用内部的参考电压计算实际的 Vcc 电压

$$VREFINT = 1.2V = \frac{ADC\_DATAx}{4095} \times VCC$$

利用 Vcc 电压来计算 Vchannnel

$$VCHANNEL = \frac{ADC\_DATAx}{4095} \times VCC$$

VREFINT 固定值为 1.2V；

VCHANNEL 是通道电压；

ADC_DATA 是 ADC_DR 里面的转换数据；

4096 表示为 12 位。

13.9. ADC 中断

ADC 中断可由以下任一事件产生：

- 任何一次的转换结束 (EOC 标志)
- 序列转换结束 (EOS 标志)
- 当模拟看门狗检测发生 (AWD 标志)
- 当采样阶段结束发生 (EOSMP 标志)
- 当数据过冲发生 (OVR 标志)

独自的中断使能位用于灵活设置 ADC 中断

表 13-5 ADC 中断

中断事件	事件标志	使能控制
转换结束	EOC	EOCIE
序列转换结束	EOS	EOSIE
模拟看门狗状态置位	AWD	AWDIE
采样阶段结束	EOSMP	EOSMPIE
过冲	OVR	OVRIE

13.10. ADC 寄存器

13.10.1. ADC 中断和状态寄存器 (ADC_ISR)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res	Res	Res	Res	Res	Res	Res	Res	AWD	Res	Res	OVR	EOSEQ	EOC	EOSMP	Res
								rc_w1			rc_w1	rc_w1	rc_w1	rc_w1	

Bit	Name	R/W	Reset Value	Function
31:8	Re-served			
7	AWD	RC_W1	0	模拟看门狗 当转换电压值超过 ADC_LTR 和 ADC_HTR 寄存器编程的值时硬件置位。软件写 1 清零。 0：无模拟看门狗事件发生（或者软件已清除该事件标志） 1：模拟看门狗事件发生
6:5	Re-served			
4	OVR	RC_W1	0	ADC 过载 当过载发生时，硬件置位该位。当 EOC 标志已置起表明一次新的转换已完成。该位写 1 清 0 0：无过载发生（或软件已应答和清除该位） 1：过载已发生
3	EOSEQ	RC_W1	0	序列结束标志 CHSEL 位选择的序列转换结束时硬件置位该位。软件写 1 清 0 0：转换序列没有完成（或者软件已经应答和清除该标志） 1：转换序列完成

Bit	Name	R/W	Reset Value	Function
2	EOC	RC_W1	0	转换结束标志 当每个通道每次转换结果后新的数据结果可以从 ADC_DR 寄存器读到时，硬件置位该位。软件写 1 清 0 或读 ADC_DR 寄存器清 0 0：通道转换没有完成（或者软件已经应答和清除该标志） 1：通道转换已完成
1	EOSMP	RC_W1	0	采样结束标志，在每次转换的采样阶段结束时，硬件置位该位，软件写 1 清 0 0：不处在采样阶段结束时（或者软件已经应答和清除该标志） 1：采样阶段结束
0	Re-served			

13.10.2. ADC 中断使能寄存器 (ADC_IER)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res	Res	Res	Res	Res	Res	Res	Re.s	AWDIE	Res	Res	OVRIE	EOSEQIE	EOCIE	EOSMPIE	Res
								rw			rw	rw	rw	rw	

Bit	Name	R/W	Reset Value	Function
31:8	Reserved			
7	AWDIE	RW	0	模拟看门狗中断使能位 软件清除或置起模拟看门狗中断 0：模拟看门狗中断不使能 1：模拟看门狗中断使能
6:5	Reserved			
4	OVRIE	RW	0	ADC 过载中断使能位 软件清除或置起过载中断使能 0：ADC 过载中断不使能 1：ADC 过载中断使能
3	EOSEQIE	RW	0	序列结束中断使能位 软件清除或置起序列结束中断使能 0：序列结束中断不使能 1：序列结束中断使能
2	EOCIE	RW	0	转换结束中断使能位 软件清除或置起转换结束中断使能位 0：转换结束中断不使能 1：转换结束中断使能
1	EOSMPIE	RW	0	采样标志结束中断使能位 软件清除或置起转换采样标志结束中断位 0：采样标志结束中断不使能 1：采样标志结束中断使能
0	Reserved			

说明： 当 ADSTART=0 时(确保没有任何转换正在进行)软件可以写这些位

13.10.3. ADC 控制寄存器 (ADC_CR)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AD-CAL	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
rs															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	AD-STP	Res	AD-START	Res	ADEN
											rs		rs		rs

Bit	Name	R/W	Reset Value	Function
31	ADCAL	RS	0	ADC 校准启动，软件设置启动 ADC 校准，校正完成后硬件自动清 0
				0: 校准完成
				1: 写 1 校正 ADC，读为 1 表明校准正在进行
30:5	Reserved			
4	ADSTP	RS	0	ADC 停止转换命令
				软件置位停止和丢弃正在进行的转换（ADSTP 命令）
				当转换被丢弃并且准备接受新的转换命令时硬件回清除该位
				0: 没有正在进行的 ADC 停止转换命令
				1: 写 1 停止 ADC，读为 1 表明一个 ADSTP 命令正在进行中。
3	Reserved			
2	ADSTART	RS	0	ADC 启动命令
				软件置位该位启动 ADC 转换。根据 EXTEN[1：0]的配置来决定转换是软件立即启动，还是由硬件触发事件来启动.该位由硬件清除：
				– 在单次转换模式 (CONT=0, DISCEN=0), 选择软件驱动时(EXTEN=00)：转换完成标志结束时（EOSEQ 标志）
				– 在非连续转换模式 (CONT=0, DISCEN=1),当软件驱动时(EXTEN=00)：转换结束标志（EOC）
				– 其他情况下：执行 ADSTP 命令之后，同时 ADSTP 标志又被硬件清 0 之时
				0: 没有正在进行的 ADC 转换
				1: 写 1 启动 ADC，读为 1 表明 ADC 正在操作可能正在转换。
				Note: Software is allowed to set ADSTART only when ADEN=1 (ADC is enabled)
1	Reserved			
0	ADEN	RS	0	ADC 使能命令
				软件置位该位使能 ADC，ADC 将准备操作。
				0: 不使能 ADC (OFF state)
				1: 使能 ADC

13.10.4. ADC 配置寄存器 1 (ADC_CFGR1)

Address offset: 0x0C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	AWDCH				Res	Res	AWDEN	AWDSGL	Res	Res	Res	Res	Res	DIS-CEN
		RW	RW	RW	RW			RW	RW						RW
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**

Res	WAIT	CONT	OV-RMOD	EXTEN[1:0]	Res	EXTSEL			ALIGN	RES_SEL		SCA-DIR	Res	Res
	RW	RW	RW	RW		RW	RW	RW	RW	RW	RW	RW		

Bit	Name	R/W	Reset Value	Function
31:30	Reserved			
29:26	AWDCH[3:0]	RW	0000	模拟看门狗通道选择，软件可清除和设置该位。 模拟看门狗监测选择的输入通道 0000：ADC 模拟输入通道 0 0001：ADC 模拟输入通道 1 0010：ADC 模拟输入通道 2 …. 1011：保留 1011：ADC 模拟输入通道 11 1100：ADC 模拟输入通道 12 其他值：保留位 *说明：AWDCH[3:0]* 位配置的通道也需要设置到 CHSELR 寄存器里 仅当 ADSART=0 时（确保没有正在进行的转换）允许软件写这些位
25：24	Reserved			
23	AWDEN	RW	0	模拟看门狗使能 软件可设置和清除该位 0：不使能模拟看门狗 1：使能看门狗 仅当 ADSART=0 时（确保没有正在进行的转换）允许软件写这些位
22	AWDSGL	RW	0	在一个通道或者所有通道使能模拟看门狗 软件可设置和清除该位取使能模拟看门狗在 AWDCH[3：0]位设置的通道上或者所有通道 0：在所有通道上使能模拟看门狗 1：在一个通道上使能模拟看门狗 仅当 ADSART=0 时（确保没有正在进行的转换）允许软件写这些位
21：17	Reserved			
16	DISCEN	RW	0	非连续模式 软件可设置和清除该位，使能/不使能非连续模式 0：不使能非连续模式 1：使能非连续模式 不可能既使能非连续模式又使能连续模式；禁止设置 *DISCEN=1 和 CONT=1.* 仅当 ADSART=0 时（确保没有正在进行的转换）允许软件写这些位
15	Reserved			
14	WAIT	RW	0	等待转换模式 软件可设置和清除该位，使能/不使能等待转换模式 0：等待转换模式关闭 1：等待转换模式打开 仅当 ADSART=0 时（确保没有正在进行的转换）允许软件写这些位
13	CONT	RW	0	单次/连续转换模式 软件可设置和清除该位。如果置为 1，直到该为被清除，否则会一致发生转换 不可能既使能非连续模式又使能连续模式；禁止设置 *DISCEN=1 和 CONT=1.* 仅当 ADSART=0 时（确保没有正在进行的转换）允许软件写这些位
12	OVRMOD	RW	0	过载管理模式 软件可设置和清除该位，配置数据过载管理的方式 0：当过载发生时，ADC_DR 寄存器保留旧值 1：当过载发生时，ADC_DR 寄存器会被上一次转换结果覆盖掉 仅当 ADSART=0 时（确保没有正在进行的转换）允许软件写这些位
11：10	EXTEN[1:0]	RW	00	外部驱动使能和极性选择 软件可设置和清除该位，选择驱动极性和使能驱动 00：硬件驱动检测不使能（软件启动转换） 01：上升沿硬件驱动检测 10：下降沿硬件驱动检测

				11：上升沿和下降沿硬件驱动检测 仅当 ADSART=0 时（确保没有正在进行的转换）允许软件写这些位
9	Reserved			
8：6	EXTSEL[2:0]	RW	000	外部驱动选择 该位选择触发转换启动的外部事件 000：TRG0(TIM1_TRG0) 001：TRG1(TIM1_CC4) 010：TRG2(Reserved) 011：TRG3(Reserved) 100：TRG4(Reserved) 101：TRG5(Reserved) 110：TRG6(Reserved) 111：TRG7(Reserved)
5	ALIGN	RW	0	数据对齐 软件设置和清除该位选择右对齐或左对齐 0：右对齐 1：左对齐 仅当 ADSART=0 时（确保没有正在进行的转换）允许软件写这些位
4：3	RES-SEL[1:0]	RW	00	数据分辨率 软件设置该位选择转换分辨率 00：12 位 01：10 位 10：8 位 11：6 位 仅当 ADEN=0 时可软件操作这些位
2	SCANDIR	RW	0	扫描序列方向 软件可设置和清除该位，选择扫描序列方向 0：向上（从通道 0 到通道 11） 1：向下（从通道 11 到通道 0） 仅当 ADSART=0 时（确保没有正在进行的转换）允许软件写这些位
1:0	Reserved			

13.10.5. ADC 配置寄存器 2 (ADC_CFGR2)

Address offset: 0x10

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31:28	CKMODE [3:0]:	RW	0	ADC 时钟模式，软件可设置和清除该位，定义模拟 ADC 的时钟源 0000：PCLK 0001：PCLK/2 0010：PCLK/4 0011：PCLK/8 0100：PCLK/16 0101：PCLK/32 0110：PCLK/64 1000：HSI 1001：HSI/2 1010：HSI/4 1011：HSI/8 1100：HSI/16 1101：HSI/32 1110：HSI/64 其他： 仅当 ADC 不使能时 ADCAL=0, ADSTART=0, ADSTP=0 and ADEN=0)。软件被允许操作这些位
27:0	Reserved			

13.10.6. ADC 采样时间寄存器 (ADC_SMPR)

Address offset: 0x14

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31：3	Reserved			
2：0	SMP[2:0]	RW	000	采样时钟选择 软件可配置该位选择所有通道的采样时间 000：3.5ADC 时钟周期 001：5.5 ADC 时钟周期 010：7.5 ADC 时钟周期 011：13.5 ADC 时钟周期 100：28.5 ADC 时钟周期 101：41.5 ADC 时钟周期 110：71.5 ADC 时钟周期 111：239.5 ADC 时钟周期 仅当 ADSART=0 时（确保没有正在进行的转换）允许软件写这些位

13.10.7. ADC 看门狗阈值寄存器 (ADC_TR)

Address offset: 0x20

Reset value: 0x0FFF 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	HT											
				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**
Res	Res	Res	Res	LT											
				RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:28	Re-served			
27:16	HT[11:0]	RW	0xFFF	模拟看门狗高阈值 软件可配，定义模拟看门狗高阈值 仅当 ADSART=0 时（确保没有正在进行的转换）允许软件写这些位
15:12	Re-served			
11:0	LT[11:0]	RW	0x000	模拟看门狗低阈值 软件可配，定义模拟看门狗低阈值 仅当 ADSART=0 时（确保没有正在进行的转换）允许软件写这些位

13.10.8. ADC 通道选择寄存器 (ADC_CHSELR)

Address offset: 0x28

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31：14	Reserved		0	
13	Reserved	RW	0	该位可写可读，无实际功能
12	CHSEL12	RW	0	通道 12（VREFINT）选择使能 0：未选中该通道 1：选中该通道 仅当 ADSART=0 时（确保没有正在进行的转换）允许软件写该位
11	CHSEL11	RW	0	通道 11（TS）选择使能 0：未选中该通道 1：选中该通道 仅当 ADSART=0 时（确保没有正在进行的转换）允许软件写该位
10	Reserved	RW	0	该位可写可读，无实际功能
9：0	CHSELx	RW	0x0000	通道选择 软件可配置这些位，定义序列转换通道 0：不选择输入通道-x 1：选择输入通道-x 仅当 ADSART=0 时（确保没有正在进行的转换）允许软件写这些位

13.10.9. ADC 数据寄存器 (ADC_DR)

Address offset: 0x40

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31：16	Reserved			
15:0	DATA[15:0]	R	0x00	转换数据 该位时只读的。上次转换通道的转换结果放于此寄存器。 数据是左对齐或者右对齐的。

13.10.10. ADC 校准配置和状态寄存器(ADC_CCSR)

Address offset: 0x44

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CALON.	CALFAIL	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
R	RC_W1														
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	CALSMP[2:0]		CALSEL	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
		RW		RW											

Bit	Name	R/W	Reset Value	Function
31	CALON	R	0	Calibration flag，标志 ADC 校准正在进行。 1：ADC 校准正在进行 0：ADC 校准已结束或未启动 ADC 校准
30	CALFAIL	RC_W1	0	Calibration fail flag，显示了 ADC 校准是否成功，与 CALON 配合使用。 CALON=0、CALFAIL=1：ADC 校准失败 CALON=0、CALFAIL=0：ADC 校准成功 CALON=1、CALFAIL=0：正在校准 CALON=1、CALFAIL=1：无效状态 硬件置位，软件写 1 清零或软件写 ADCAL=1 时清零。
29:14	Reserved	-	0	-
13:12	CALSMP[2:0]	RW	0	Calibration sample time selection 根据以下信息，配置 calibration 的采样阶段的时钟周期个数： 00：2 个 ADC 时钟周期 01：4 个 ADC 时钟周期 10：8 个 ADC 时钟周期 11：1 个 ADC 时钟周期 校准时配置 SMP 的周期越长，校准结果更精确，但该配置会带来校准周期延长的问题
11	CALSEL	RW	0	Calibration 内容选择位，用于选择需要校准的内容 1：校准 OFFSET 以及线性度 0：只校准 OFFSET
10:0	Reserved	-	0	-

13.10.11. ADC 通用配置寄存器 (ADC_CCR)

Address offset: 0x308

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31：24	Reserved			
23	TSEN	RW	0	温度传感器使能位，软件可设置和清除该位，使能/不使能温度传感器 0：不使能 1：使能 仅当 ADSART=0 时（确保没有正在进行的转换）允许软件写这些位
22	VREFEN	RW	0	基准 Vrefint 使能位，软件可设置和清除该位，使能/不使能基准 Vrefint 0：不使能 1：使能 仅当 ADSART=0 时（确保没有正在进行的转换）允许软件写这些位
21：0	Reserved			

13.10.12. ADC 寄存器映像

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	ADC_ISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AWD	Res.	Res.	OVR	EOSEQ	EOC	EOSMP	Res.
	Reset value																									0			0	0	0	0	
0x04	ADC_IER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AWDIE	Res.	Res.	OVRIE	EOSE-	EOCIE	EOSMPI	Res.
	Reset value																									0			0	0	0	0	
0x08	ADC_CR	ADCAL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADSTP	Res.	AD-	Res.	ADEN
	Reset value	0																											0		0		0
0x0C	ADC_CFGR1	Res.	Res.	AWDCH [3:0]				Res.	Res.	AWDEN	AWDSGL	Res.	Res.	Res.	Res.	Res.	DISCEN	Res.	WAIT	CONT	OVRMOD	Res.	Res.	Res.	EXTSEL [2:0]			ALIGN	RES_SEL[1:0]		SCADIR	Res.	Res.
	Reset value			0	0	0	0			0	0						0		0	0	0				0	0	0	0	0	0	0	0	0
0x10	ADC_CFGR2	CKMODE [3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0	0	0	0																												
0x14	ADC_SMPR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SMP [2:0]		
	Reset value																														0		
0x20	ADC_TR	Res.	Res.	Res.	Res.	HT[11:0]												Res.	Res.	Res.	Res.	LT[11:0]											
	Reset					1	1	1	1	1	1	1	1	1	1	1	1					0	0	0	0	0	0	0	0	0	0	0	0

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	value																																
0x28	ADC_CHSELR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CHSEL12	CHSEL11	Res.	CHSEL9	CHSEL8	CHSEL7	CHSEL6	CHSEL5	CHSEL4	CHSEL3	CHSEL2	CHSEL1	CHSEL0
	Reset value																				0	0	0	0	0	0	0	0	0	0	0	0	0
0x40	ADC_DR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DATA[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x44	ADC_CCSR	CALON	CALFAIL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CALSMP[1:0]		CALSEL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0	0																	0	0	0											
0x308	ADC_CCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TSEN	VREFE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value									0	0																						

14. 比较器 (COMP)

14.1. 简介

芯片内集成 2 个通用比较器（general purpose comparators）COMP，分别是 COMP1 和 COMP2。这两个模块可以作为单独的模块，也可以与 timer 组合在一起使用。

比较器可以被如下使用：

- 被模拟信号触发，产生低功耗模式唤醒功能
- 模拟信号调节
- 当与来自 timer 的 PWM 输出连接时，Cycle by cycle 的电流控制回路

14.2. COMP 主要特性

- 每个比较器有可配置的正或者负输入，以实现灵活的电压选择
  - 多路 I/O pin
  - 电源 VCC
  - 温度传感器的输出
  - 内部参考电压和通过分压提供的 3 个分数值（1/4、1/2、3/4）
- 迟滞功能可配置
- 可编程的速度和功耗
- 输出可以被连接到 I/O 或者 timer 的输入作为触发
  - OCREF_CLR 事件（cycle by cycle 的电流控制）
  - 为快速 PWM shutdown 的刹车
- COMP1 和 COMP2 可以组合成 window COMP
- 每个 COMP 具有中断产生能力，用作芯片从低功耗模式（sleep 和 stop 模式）的唤醒（通过 EXTI）

14.3. COMP 功能描述

14.3.1. COMP 框图

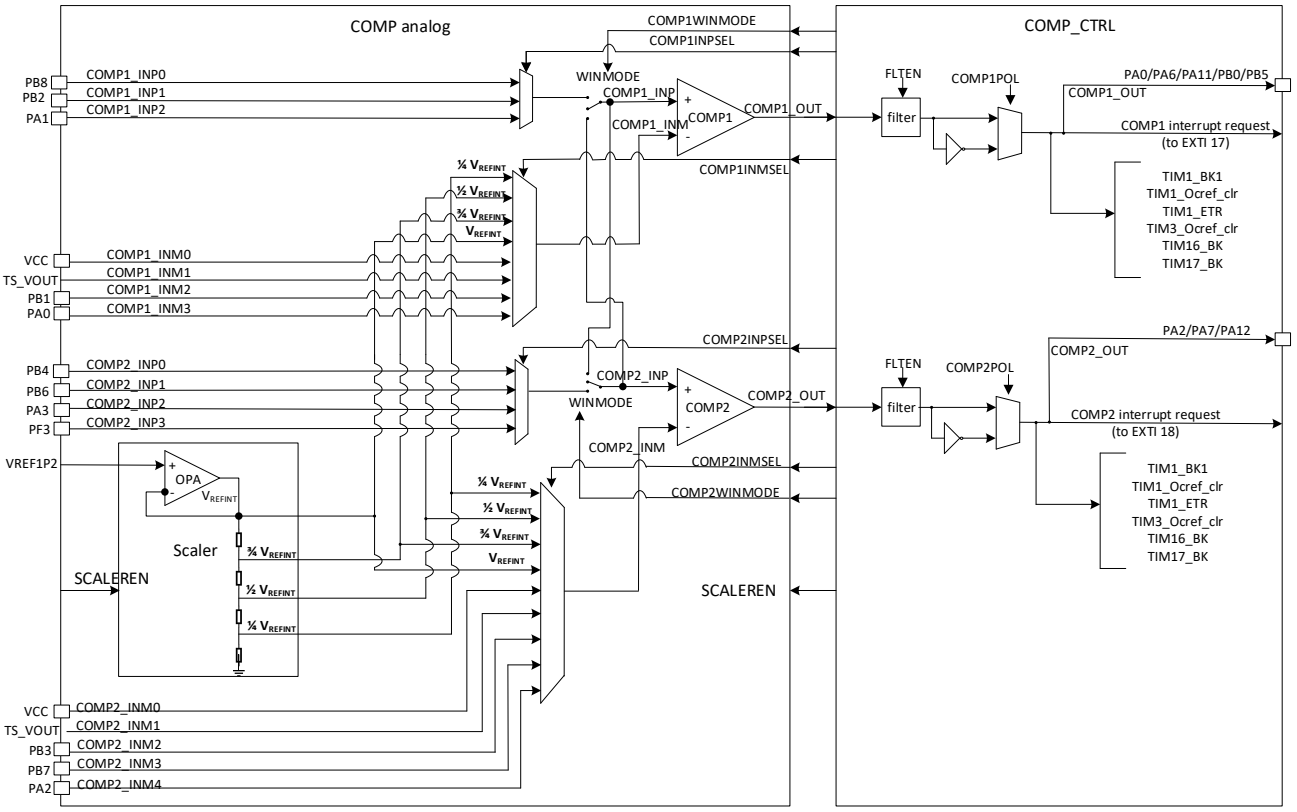


图 14-1 比较器架构框图

14.3.2. COMP 管脚和内部信号

用作比较器输入的 I/O，必须在 GPIO 寄存器中被配置成模拟模式。

比较器输出可以通过在 GPIO 的复用功能通道（alternate function）连接到 I/O pin。

输出也可以在内部连接到各种 timer 的输入，达到以下目的：

- 连接刹车输入时，PWM 信号的紧急 shut-down
- 使用 OCREF_CLR 输入的 Cycle-by-cycle 电流控制
- 时序测量的输入捕获

14.3.3. COMP 复位和时钟

COMP 模块有两个时钟源：

1） PCLK（APB clock），用于给配置寄存器提供时钟

2） COMP 时钟，用于模拟比较器输出后的电路（模拟输出的锁存电路、滤毛刺电路等）的时钟，可选择为 PCLK 或者 LSI。当需要在 stop 模式下工作时，选择 LSI。

COMP 模块的复位信号源有：

1）模拟比较器输出后电路（模拟输出的锁存电路、滤毛刺电路等）的复位，该复位信号包括 APB 复位源和 COMP 模块软件复位源（RCC_APBRSTR2.COMP1RST 和 RCC_APBRSTR2.COMP2RST）

14.3.4. COMP 锁定机制

比较器可以用在安全的用途，例如过流和温度保护。对于有特定功能性安全需求的应用，需要确保在寄存器访问出错和 PC（program counter）混乱时，比较器的程序不能被改写。

由此，比较器控制和状态寄存器可以被写保护（只读）。

如果寄存器的写完成，COMPx Lock 位要被置成 1，这使得整个寄存器变成只读，包括 COMPx Lock 位。

写保护仅能够被芯片的复位信号复位掉。

14.3.5. Window 比较器

Window 比较器的作用是监测模拟电压是否在低和高阈值范围内。

可以使用两个比较器创建 window 比较器。被监测的模拟电压同时连接到两个比较器的 non-inverting（+端）输入，高阈值和低阈值分别连接到两个比较器的 inverting 输入端（-端）。

通过使能 WINMODE 位，可以将两个比较器的 non-inverting（+输入端）连接到一起，起到节省一个 I/O pin 的作用。

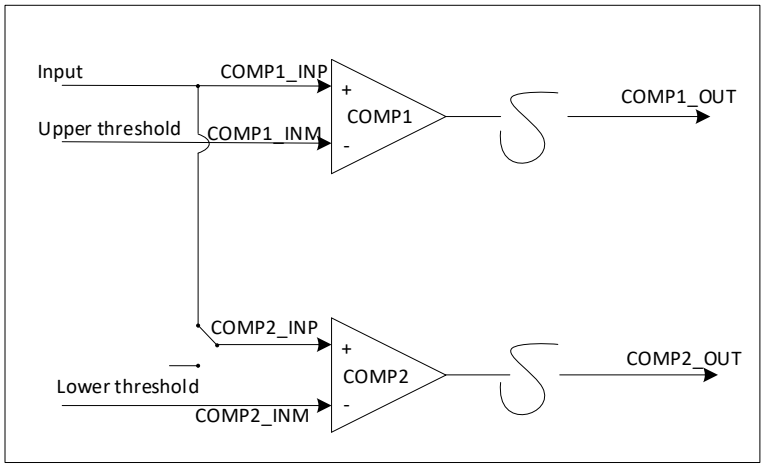


图 14-2 window comparator

14.3.6. 迟滞

为避免在噪声信号情况产生假的输出转换，比较器可以使能带有迟滞的功能（通过使能 COMP1_CSR 的 HYST 位，可同时打开 COMP1 和 COMP2 的迟滞功能）。

14.3.7. 功耗模式

比较器的功耗和传输延迟可以通过 COMPx_CSR 寄存器的 PWRMODE[1:0]位来选择不同模式，以实现在特定应用的最适合的 trade-off。可选的模式包括 high speed 和 medium speed 两种，相对而言 high speed mode 下的功耗更大，传输延迟也更小。注意，当进入 stop 之前，如果选择 PWR_CR2 寄存器 LPR=1（即选择用 low power regulator 供电），则需要首先设定 COMP 在 Medium speed(PWRMODE=01)。

此外，为降低功耗，APB 时钟和 COMP 时钟被 RCC_APBENR2.COMP1EN（和 RCC_APBENR2.COMP2EN）控制，软件可在使用 COMP 模块时，才使能该寄存器。

14.3.8. 比较器滤波

可以通过设定 COMP_FR 寄存器，使能 COMP 的输出滤波功能及相应的滤波宽度。注意该设定应在 COMP_EN 使能前完成。

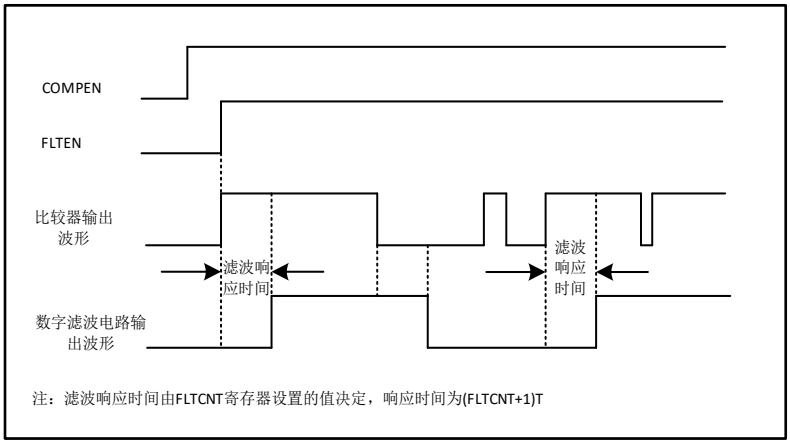


图 14-3 COMP 过滤器

14.3.9. COMP 中断

比较器输出在芯片内部连接到 EXTI 控制器（extended interrupts and events）。每个比较器有单独的 EXTI line（17 和 18），并能够产生中断或者事件。相同的机制被用作从低功耗的唤醒。

14.4. COMP 寄存器

14.4.1. COMP1 控制和状态寄存器(COMP1_CSR)

Address:0x00

Reset value:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LOCK	COMP_OUT	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	PWR-MODE[1:0]		Res	HYST
RW	R											RW	RW		RW
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PO-LAR-ITY	Res	Res	Res	WINMODE	Res	INPSEL[1:0]		INMSEL[3:0]				Res	Res	SCALER_EN	COMP1_EN
RW		-	-	RW	-	RW	RW	RW	RW	RW	RW			RW	RW

Bit	Name	R/W	Reset Value	Function
31	LOCK	RW	0	COMP1_CSR 寄存器 lock 软件置位，系统复位清零。当被置位，则会锁定 COMP1_CSR 寄存器的所有 32 位 0：未锁定，可读写整个寄存器 1：锁定，整个寄存器只读
30	COMP_OUT	R		COMP1 输出状态 该位只读，它反映了 COMP1 在经过极性选择的输出电平。
29：20	Reserved			
19：18	PWRMODE[1:0]	RW	0	COMP1 功耗模式选择 软件可读可写，选择了功耗和由此而来的 COMP1 的速度。高速模式下功耗更大，延迟更小 00：High speed 01：Medium speed 10：High speed 11：High speed 注：该 bit 不受 LOCK 功能控制。

17	Reserved			
16	HYST	RW	0	COMP1 和 COMP2 迟滞功能使能控制 0：迟滞功能关闭 1：迟滞功能使能
15	POLARITY	RW	0	COMP1 极性选择 软件可读可写（如果没有被锁定） 0：不反向 1：反向
14：12	Reserved			
11	WINMODE	RW	0	COMP1 输出选择（window 模式） 软件可读可写（如果没有被锁定） 0：信号被 INPSEL[1:0]选择 1：COMP2 的 COMP2_INP 信号 注意两个 COMP 的 WINMODE 模式不能同时使能。
10	Reserved			
9：8	INPSEL[1:0]	RW	00	00：PB8 01：PB2 10：PA1 11：Reserved
7：4	INMSEL[3:0]	RW	0000	0000：1/4 VREFINT 0001：1/2 VREFINT 0010：3/4 VREFINT 0011：VREFINT 0100：VCC 0101：TS 0110：PB1 0111：Reserved 1000：PA0 其他：1/4 VREFINT
3：2	Reserved			
1	SCALER_EN	RW	0	VREFINT 相关输入使能，当选择 VREFINT、3/4 VREFINT、1/2 VREFINT、1/4 VREFINT 中任何一个作为比较器输入时，都要打开该寄存器位。 0：不打开 SCALER 1：使能 SCALER
0	COMP1_EN	RW	0	COMP1 使能位 软件可读可写（如果没有被锁定） 0：Disable 1：Enable

14.4.2. COMP1 滤波寄存器(COMP1_FR)

Address offset:0x04

Reset value:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FLTCNT1[15:0]															
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	FLTEN1
															RW

Bit	Name	R/W	Reset Value	Function
31:16	FLTCNT1	RW	0x0	比较器 1 采样滤波计数器 采样时钟为 APB 或 LSI。滤波计数值可配置。采样次数达到滤波计数值时，结果一致输出。 采样计数周期=FLTCNT[15:0]
15:1	Reserved		0x0	
0	FLTEN1	RW	0x0	比较器 1 数字滤波功能配置 0：禁止数字滤波功能 1：使能数字滤波功能 Note: 该位必须在 COMP1_EN 为 0 时置位

14.4.3. COMP2 控制和状态寄存器(COMP2_CSR)

Address:0x10

Reset value:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LOCK	COMP_OUT	Res	Res	Res	Res	Res	Res					PWR-MODE[1:0]		Res	
RW	R											RW	RW		
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
PO-LAR-ITY	Res	Res	Res	WINMODE	Res	INPSEL[1:0]		INMSEL[3:0]				Res	Res	Res	COMP2_EN
RW		-	-	RW	-	RW	RW	RW	RW	RW	RW				RW

Bit	Name	R/W	Reset Value	Function
31	LOCK	RW	0	COMP2_CSR 寄存器 lock 软件置位，系统复位清零。当被置位，则会锁定 COMP2_CSR 寄存器的所有 32 位 0：未锁定，可读写整个寄存器 1：锁定，整个寄存器只读
30	COMP_OUT	R		COMP2 输出状态 该位只读，它反映了 COMP2 在经过极性选择的输出电平。
29：20	Reserved			
19：18	PWRMODE[1:0]	RW		COMP2 功耗模式选择 软件可读可写，选择了功耗模式和由此而来的 COMP2 的速度 00：High speed 01：Medium speed 10：High speed 11：High speed 注：该 bit 不受 LOCK 功能控制。
17：16	Reserved			
15	POLARITY	RW		COMP2 极性选择 软件可读可写（如果没有被锁定） 0：不反向 1：反向
14：12	Reserved			
11	WINMODE	RW		COMP2 不反向的输出选择（window 模式） 软件可读可写（如果没有被锁定） 0：信号被 INPSEL[1:0]选择 1：COMP2 的 COMP2_INP 信号 注意两个 COMP 的 WINMODE 模式不能同时使能。
10	Reserved			
9：8	INPSEL[1:0]	RW		COMP2 不反向输入的信号选择 软件可读可写（如果没有被锁定） 00：PB4 01：PB6 10：PA3 11：PF3
7：4	INMSEL[3:0]	RW		0000：1/4 VREFINT 0001：3/4 VREFINT 0010：1/2 VREFINT 0011：VREFINT 0100：VCC 0101：TS 0110：PB3 0111：PB7 1000：PA2 >1000：1/4 VREFINT

Bit	Name	R/W	Reset Value	Function
3：1	Reserved			
0	COMP2_EN	RW		COMP2 使能位 软件可读可写（如果没有被锁定） 0：Disable 1：Enable

14.4.4. COMP2 滤波寄存器(COMP2_FR)

Address:0x14

Reset value:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FLTCNT2[15:0]															
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	FLTEN2
															RW

Bit	Name	R/W	Reset Value	Function
31:16	FLTCNT2[15:0]	RW	0x0	比较器 2 采样滤波计数器 采样时钟为 APB 或 LSI。滤波计数值可配置。采样次数达到滤波计数值时，结果一致输出。 采样计数周期=FLTCNT[15:0]
15:1	Reserved		0x00	
0	FLTEN2	RW	0x0	比较器 2 数字滤波功能配置 0：禁止数字滤波功能 1：使能数字滤波功能 Note: 该位必须在 COMP2_EN 为 0 时置位

14.4.5. COMP 寄存器映像

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	COMP1_CSR	LOCK	COMP_OUT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PWR-MODE[1:0]		Res.	HYST	POLARITY	Res.	Res.	Res.	WINMODE	Res.	INPSEL[1:0]		INMSEL [3:0]				Res.	Res.	SCALER EN	COMP1 EN
	Reset value	0	0											0	0		0	0				0		0	0	0	0	0	0			0	0
0x004	COMP1_FR	FLTCNT1[15:0]																Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FLTEN1
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0																0
0x010	COMP2_CSR	LOCK	COMP_OUT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PWR-MODE[1:0]		Res.	HYST	POLARITY	Res.	Res.	Res.	WINMODE	Res.	INPSEL[1:0]		INMSEL [3:0]				Res.	Res.	SCALER EN	COMP2 EN
	Reset value	0	0											0	0		0	0				0		0	0	0	0	0	0			0	0
0x014	COMP2_FR	FLTCNT2[15:0]																Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FLTEN2

	Re-set valu e	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0														0

15. 高级控制定时器 (TIM1)

15.1. TIM1 简介

高级 timer（TIM1）由 16 位被可编程分频器驱动的自动装载计数器组成。它可以被用作各种场景，包括：输入信号（输入捕获）的脉冲长度测量，或者产生输出波形（输出比较、输出 PWM、带死区插入的互补 PWM）。

脉冲长度和波形周期可以使用定时器分频器和 RCC 时钟控制分频器，从微秒到毫秒的调制。高级 timer（TIM1）和通用（TIMx）timer 是完全独立的，不共享任何资源。他们可以同步起来。

15.2. TIM1 主要特性

- 16bit 向上、向下或者向上向下的自动重装载计数器
- 16bit 可编程分频器，允许对计数器的时钟频率进行 1 到 65535 的分频（on the fly）
- 多达 4 个独立的通道
  - 输入捕获
  - 输出比较
  - PWM 产生（边缘或者中心对齐模式）
  - 单脉冲模式输出
- 死区时间可编程的互补输出
- 使用外部信号控制定时器和定时器互连的同步电路
- 重复计数器，在计数指定周期数后，才更新时间寄存器
- 刹车输入可以将定时器的输出信号置为复位状态和已知状态
- 中断产生在以下事件
  - 更新：计数器向上、向下溢出，计数器初始化（通过软件或者内外部触发）
  - 触发事件
  - 输入捕获
  - 输出比较
  - 刹车输入
- 支持增量式的（正交）编码器和为定位用的霍尔传感器电路
- 触发输入作为外部时钟或者按周期的电流管理

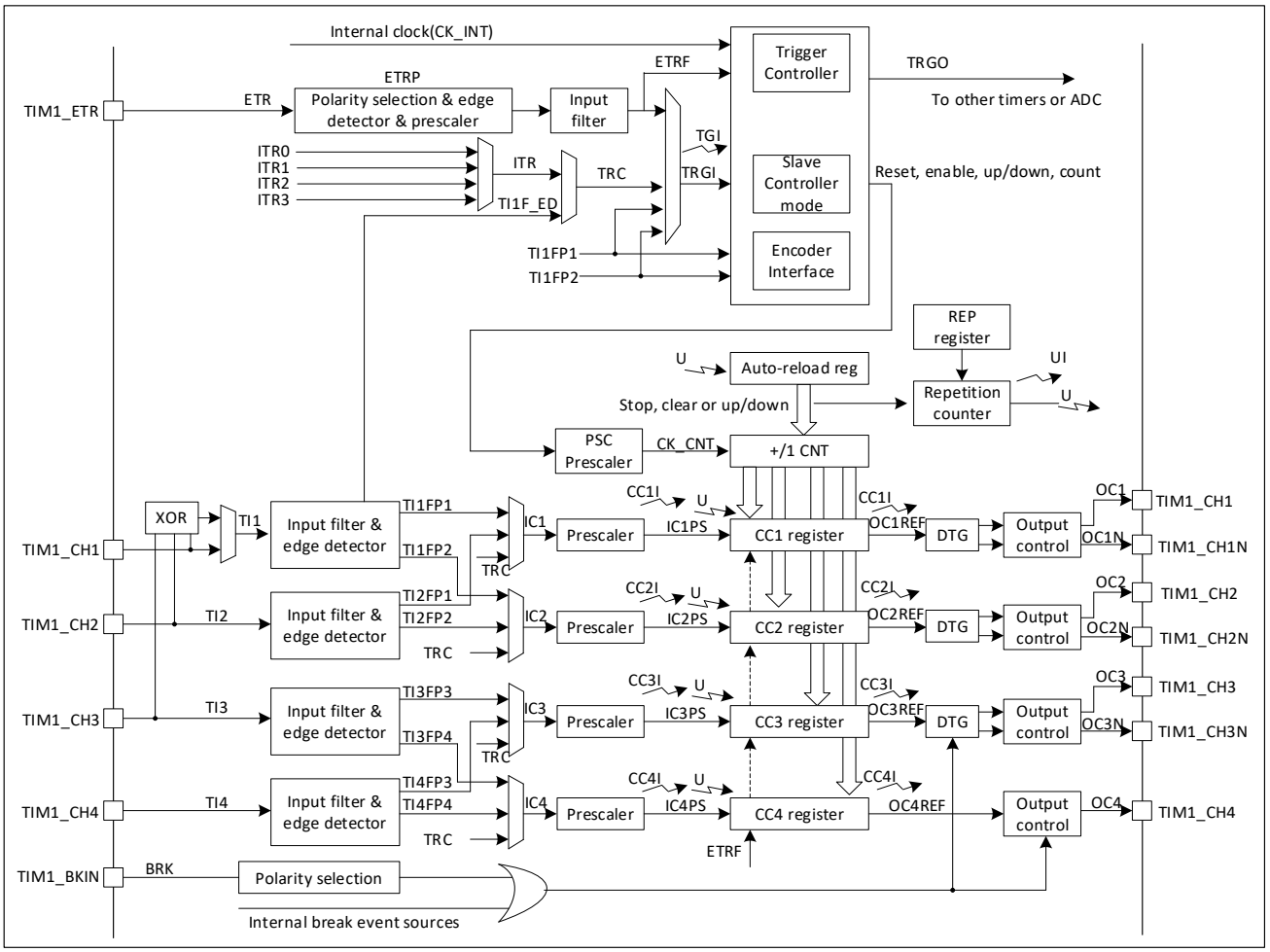


图 15-1 高级控制定时器架构框图

15.3. TIM1 功能描述

15.3.1. 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。

时基单元包括：

- 计数器寄存器（TIM1_CNT）
- 预分频寄存器（TIM1_PSC）
- 自动装载寄存器（TIM1_ARR）
- 重复计数寄存器（TIM1_RCR）

自动装载寄存器是预先装载的，写或者读自动重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。当计数器达到上溢出（向下计数器时的下溢条件）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。

计数器由预分频器的时钟输出 CK_CNT 驱动，仅当设置了计数器 TIM1_CR1 寄存器中的计数器使能位（CEN）时，CK_CNT 才有效。

注意，在设置了 TIM1_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按 1 到 65535 之间的任意值分频。它是基于一个（在 TIMx_PSC 寄存器中的）16 位寄存器控制的 16 位计数器。因为这个控制寄存器带有缓冲器，它能够在运行时被改变。新的预分频器的参数在下一次更新事件到来时被采用。

图 16-2 和图 16-3 给出了在预分频器运行时，更改计数器参数的例子。

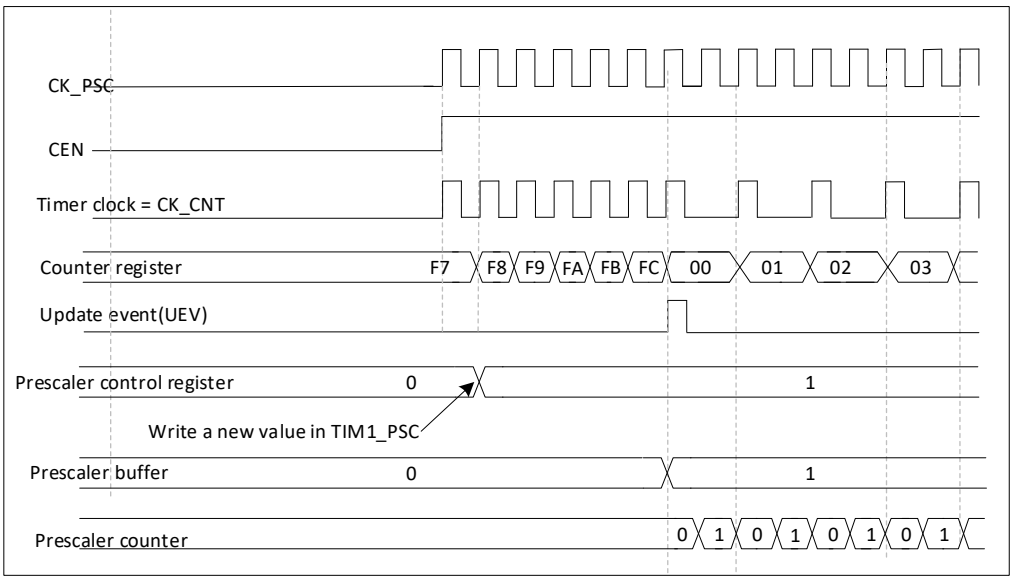


图 15-2 当预分频器的参数从 1 变到 2 时，计数器的时序图

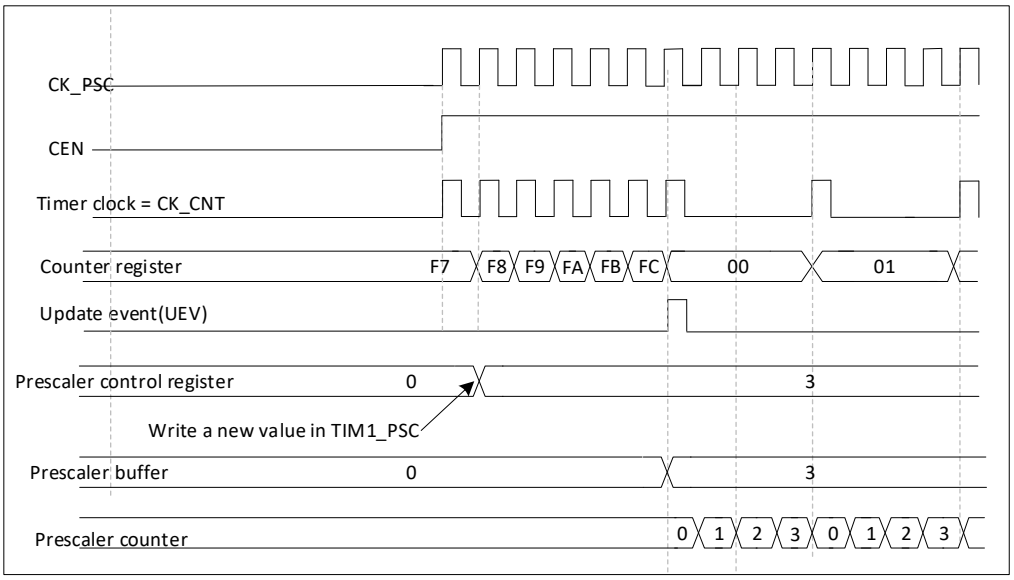


图 15-3 当预分频器的参数从 1 变到 4 时，计数器的时序图

15.3.2. 计数器模式

向上计数模式

向上计数模式，是从 0 到自动装载值的计数器，然后又从 0 重新开始计数，并产生一个计数的溢出事件。

如果重复计数器被使用，则在向上计数器重复几次（对重复计数器可编程）后，产生更新事件。否则，在每个计数溢出时，产生更新事件。

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

设置 TIMx_CR1 寄存器中的 UDIS 位，可以禁止更新事件；这样也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 位被清零之前，将不产生更新事件。即使这样，在应该产生更新事件时，计数器仍

会被清'0'，同时预分频器的计数也被请 0(但预分频器的数值不变)。此外，如果设置了 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。

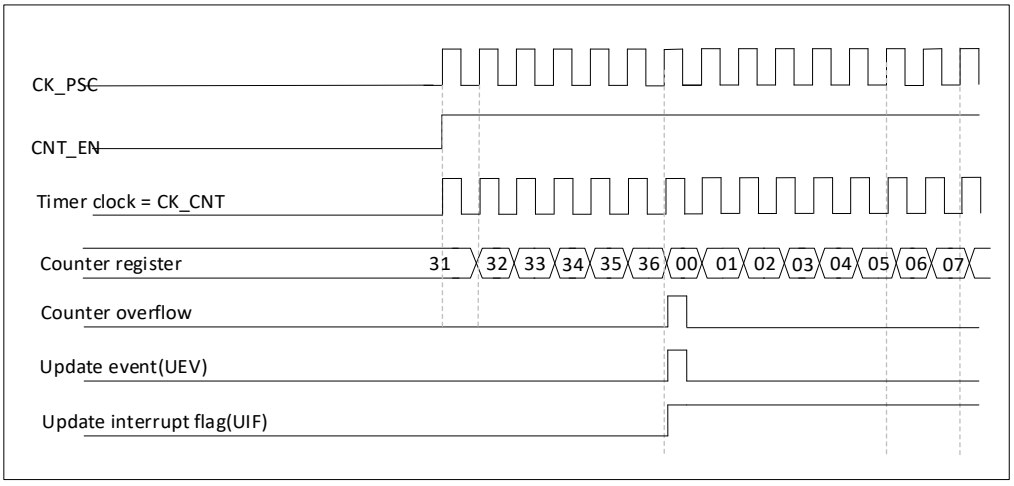


图 15-4 计数器时序图，内部时钟分频因子为 1

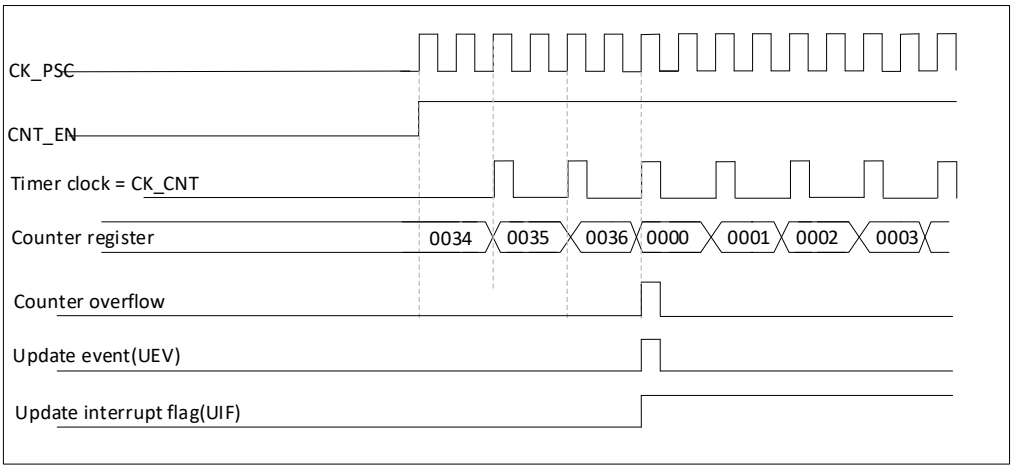


图 15-5 计数器时序图，内部时钟分频因子为 2

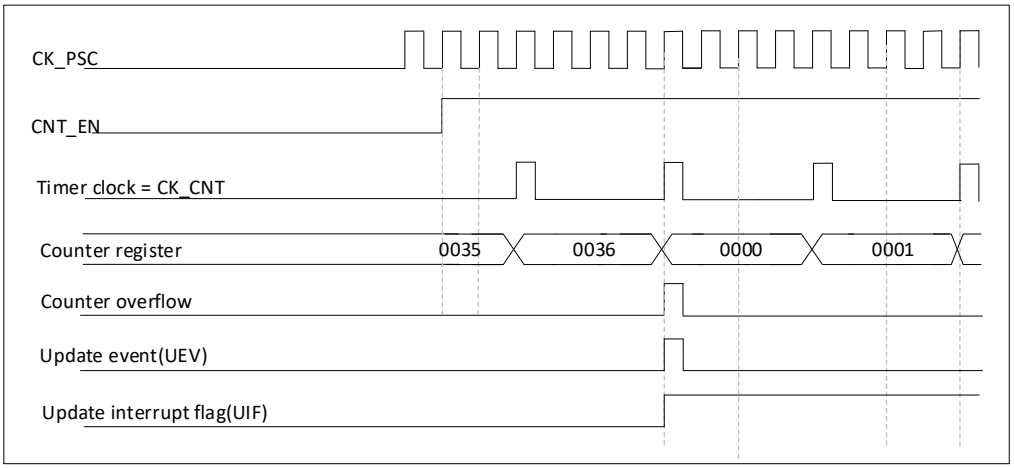


图 15-6 计数器时序图，内部时钟分频因子为 4

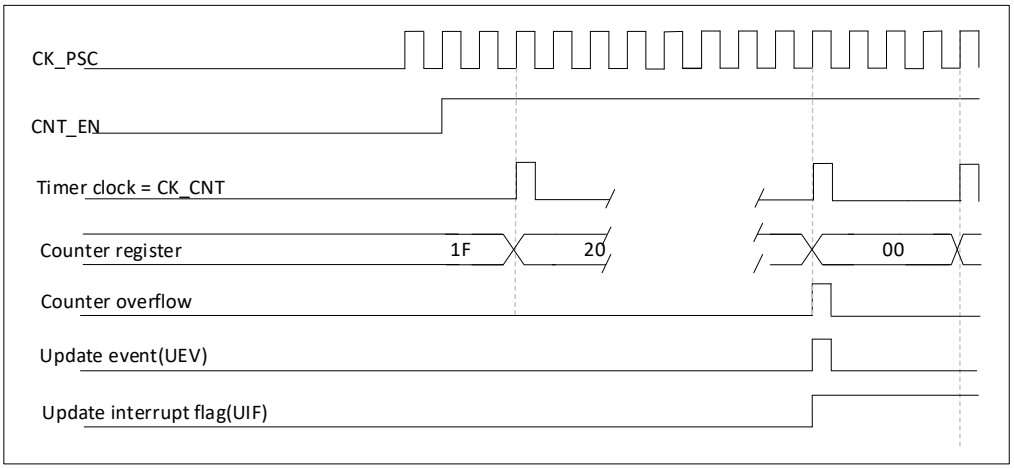


图 15-7 计数器时序图，内部时钟分频因子为 N

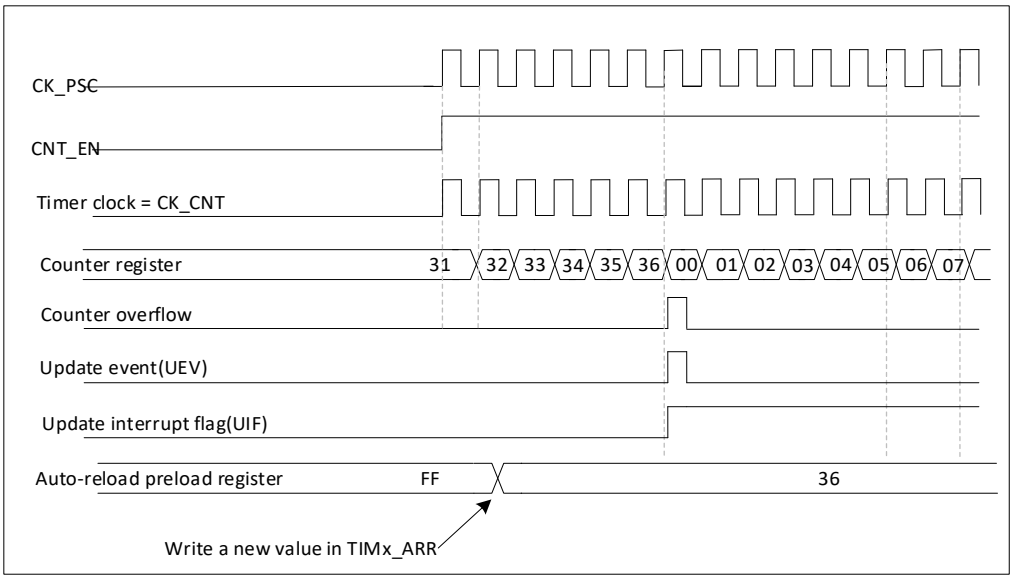


图 15-8 计数器时序图，当 ARPE=0 时的更新事件(TIM1_ARR 没有预装入)

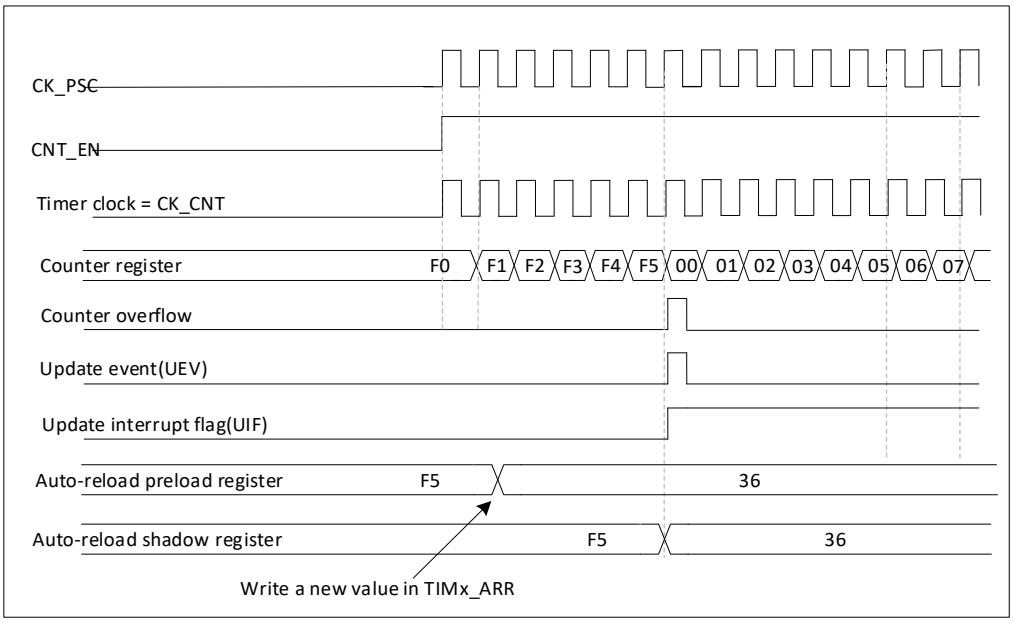


图 15-9 计数器时序图，当 ARPE=1 时的更新事件(预装入了 TIM1_ARR)

向下计数模式

向下计数模式，从自动装载的值开始向下计数到 0，然后重新开始从自动装载的值向下计数，并产生一个向下溢出事件。

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

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

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

此外，如果设置了 TIMx_CR1 寄存器中的 URS 位(选择更新请求)，设置 UG 位将产生一个更新事件 UEV 但不设置 UIF 标志(因此不产生中断请求)，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。

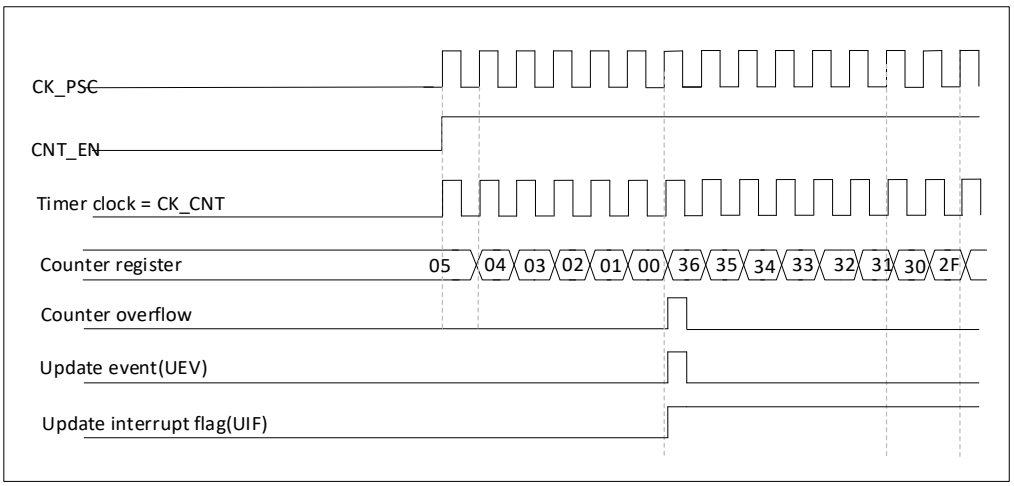


图 15-10 计数器时序图，内部时钟分频因子为 1

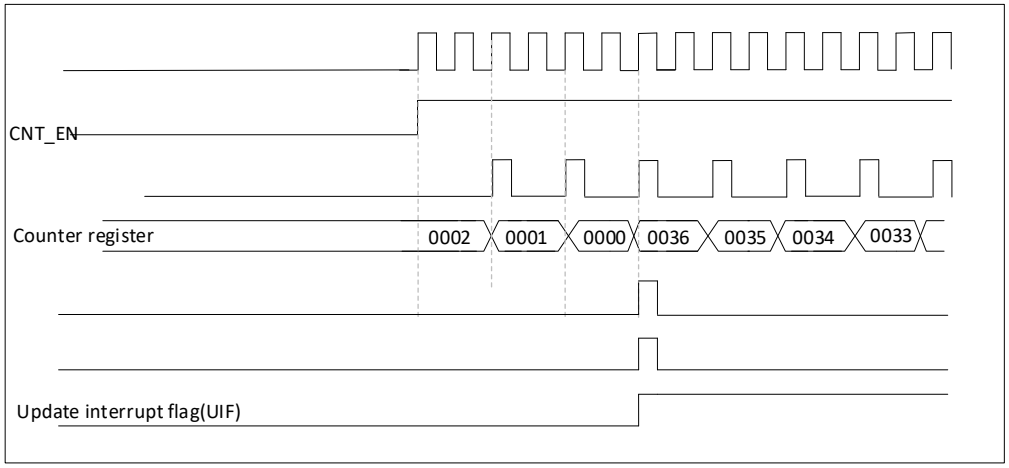


图 15-11 计数器时序图，内部时钟分频因子为 2

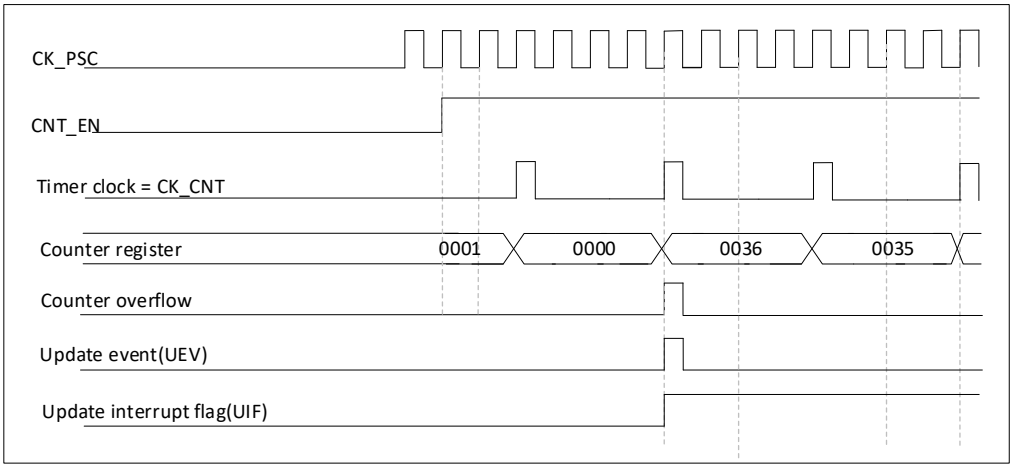


图 15-12 计数器时序图，内部时钟分频因子为 4

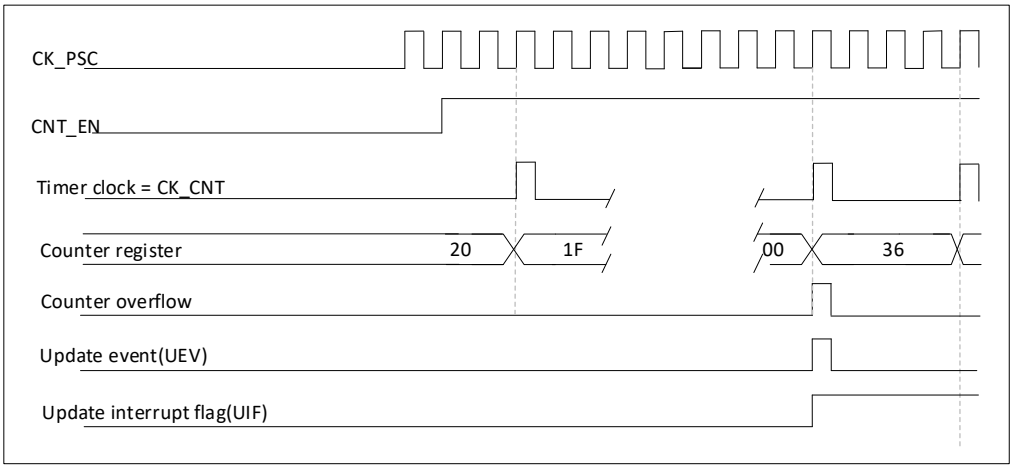


图 15-13 计数器时序图，内部时钟分频因子为 N

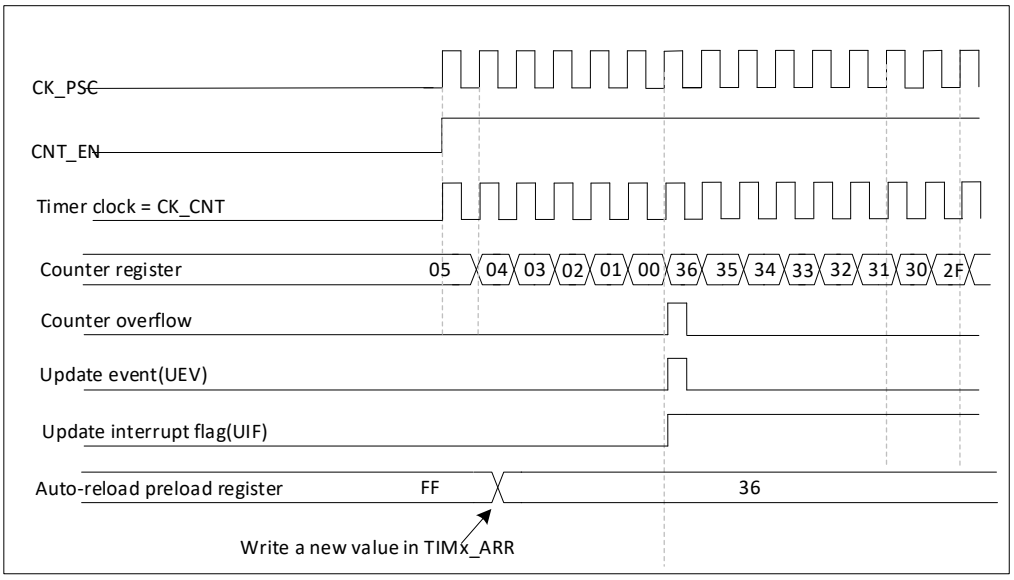


图 15-14 计数器时序图，当没有使用周期计数器时的更新事件

中央对齐模式 (向上/向下计数)

在中央对齐模式，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。

中央对齐模式在 TIMx_CR1 寄存器中的 CMS 不等于 0 时有效。通道在配置成输出模式时，输出比较中断标志将被置位，当：向下计数时（中央对齐模式 1，CMS=”01”）,向上计数时（中央对齐模式 2，CMS=”10”）向上向下计数（中央对齐模式 3，CMS=”11”）。

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

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

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

此外，如果设置了 TIMx_CR1 寄存器中的 URS 位(选择更新请求) ，设置 UG 位将产生一个更新事件 UEV 但不设置 UIF 标志(因此不产生中断请求)，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

  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)

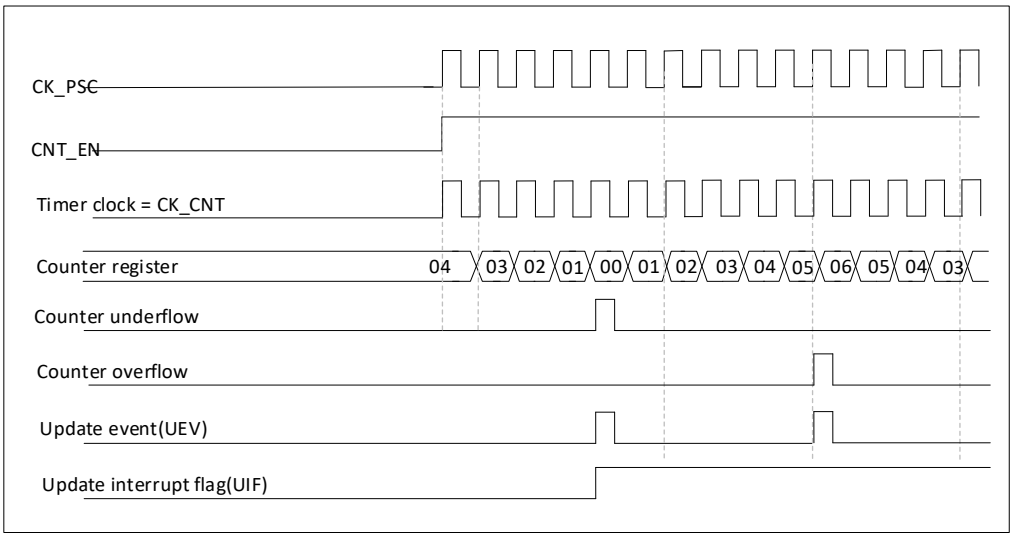


图 15-15 计数器时序图，内部时钟分频因子为 1，TIMx_ARR = 0x6

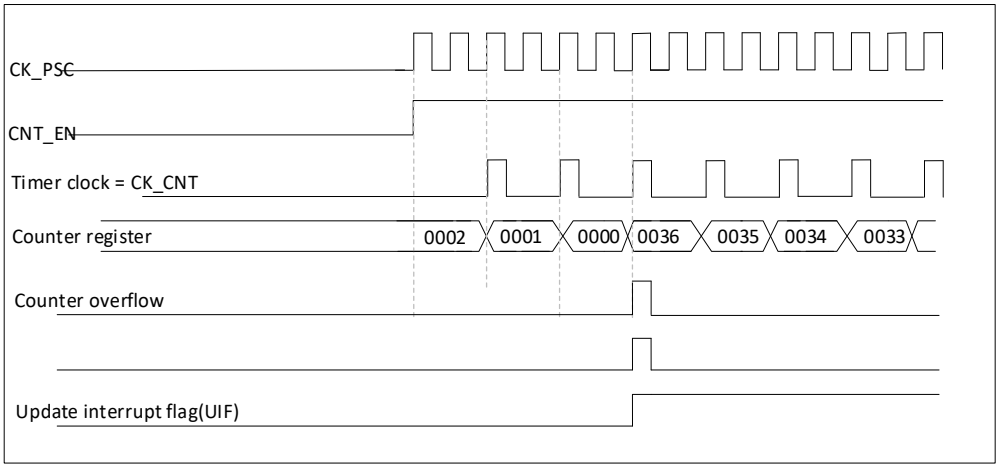


图 15-16 计数器时序图，内部时钟分频因子为 2, TIMx_ARR=0x36

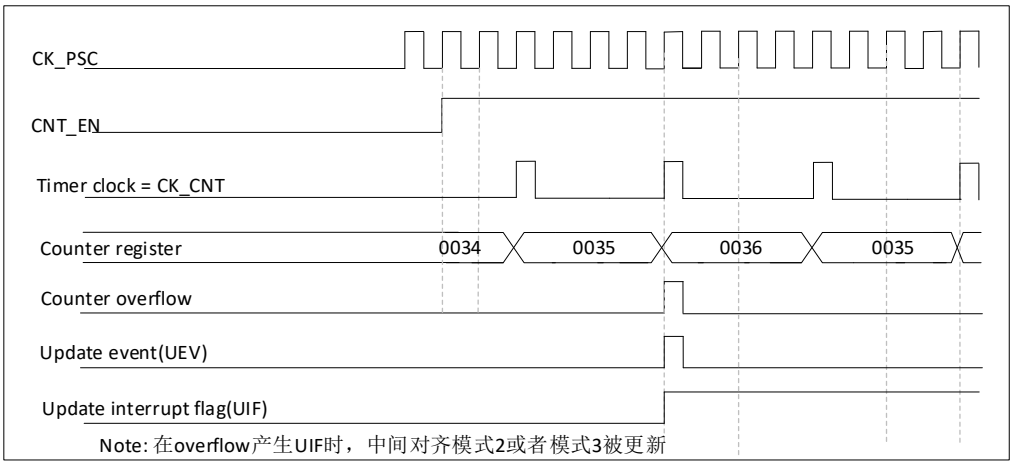


图 15-17 计数器时序图，内部时钟分频因子为 4, TIMx_ARR=0x36

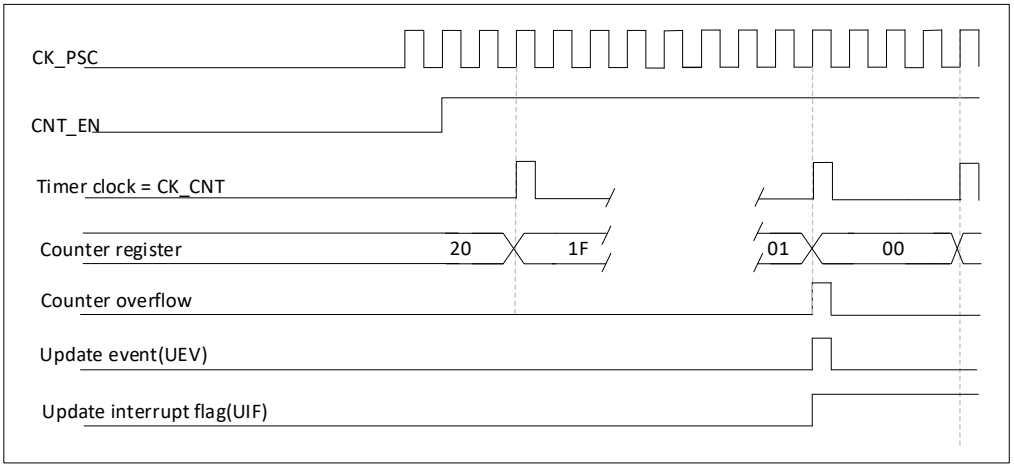


图 15-18 计数器时序图，内部时钟分频因子为 N

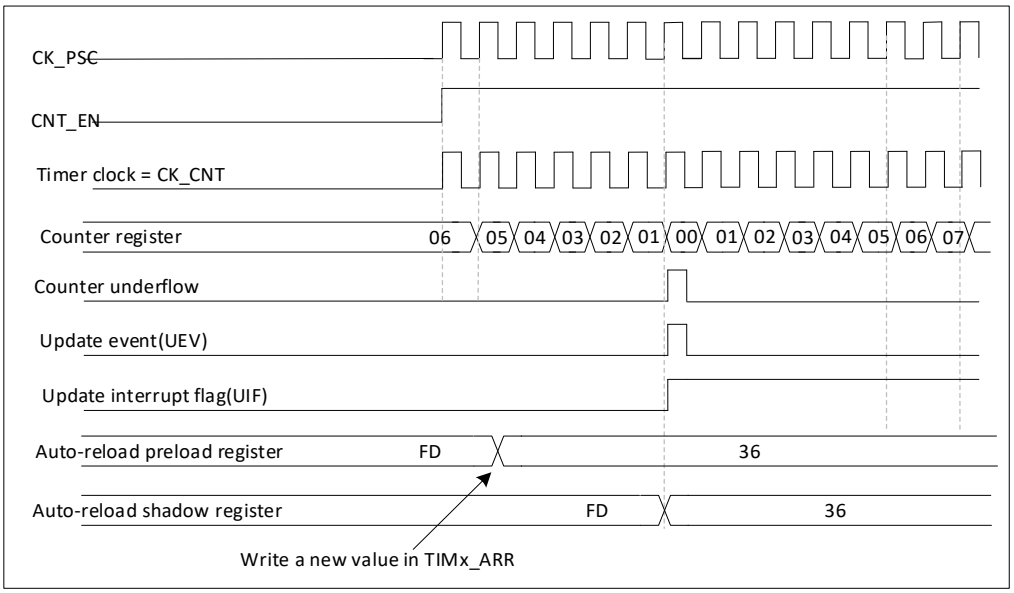


图 15-19 计数器时序图，ARPE=1 时的更新事件(计数器下溢)

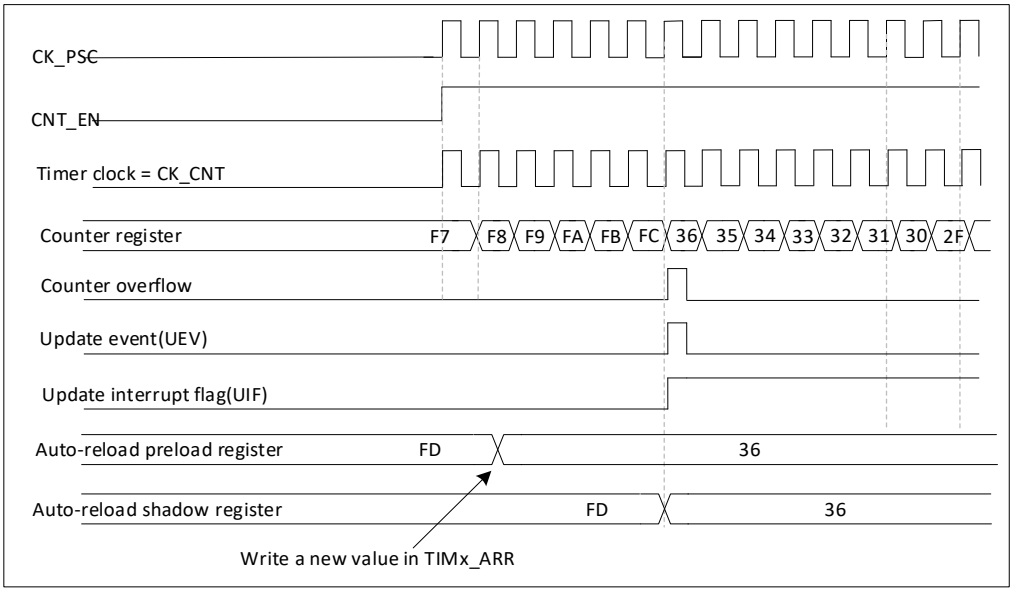


图 15-20 计数器时序图，ARPE=1 时的更新事件(计数器溢出)

15.3.3. 重复向下计数器

时基单元描述了关于计数器向上、向下溢出的更新事件如何产生的。它实际上仅当重复计数器计数到零才产生。这也是当产生 PMW 信号时很有用的。

这意味着在每 N 次计数上溢或下溢时，数据被从预装载寄存器传送到影子寄存器（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何一条件成立时递减：

- 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 的最大循环周期位 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。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如到重复计数器。

在中央对齐模式下，对于 RCR 的奇数值，取决于当 RCR 寄存器被写入和当计数器开始，出现上溢、或者下溢，则更新事件产生。如果在启动计数器之前写 RCR，在上溢时产生更新事件。例如，对于 RCR = 3 时，更新事件被产生在第 4 个上溢或者下溢事件（取决于 RCR 被写入的值）。

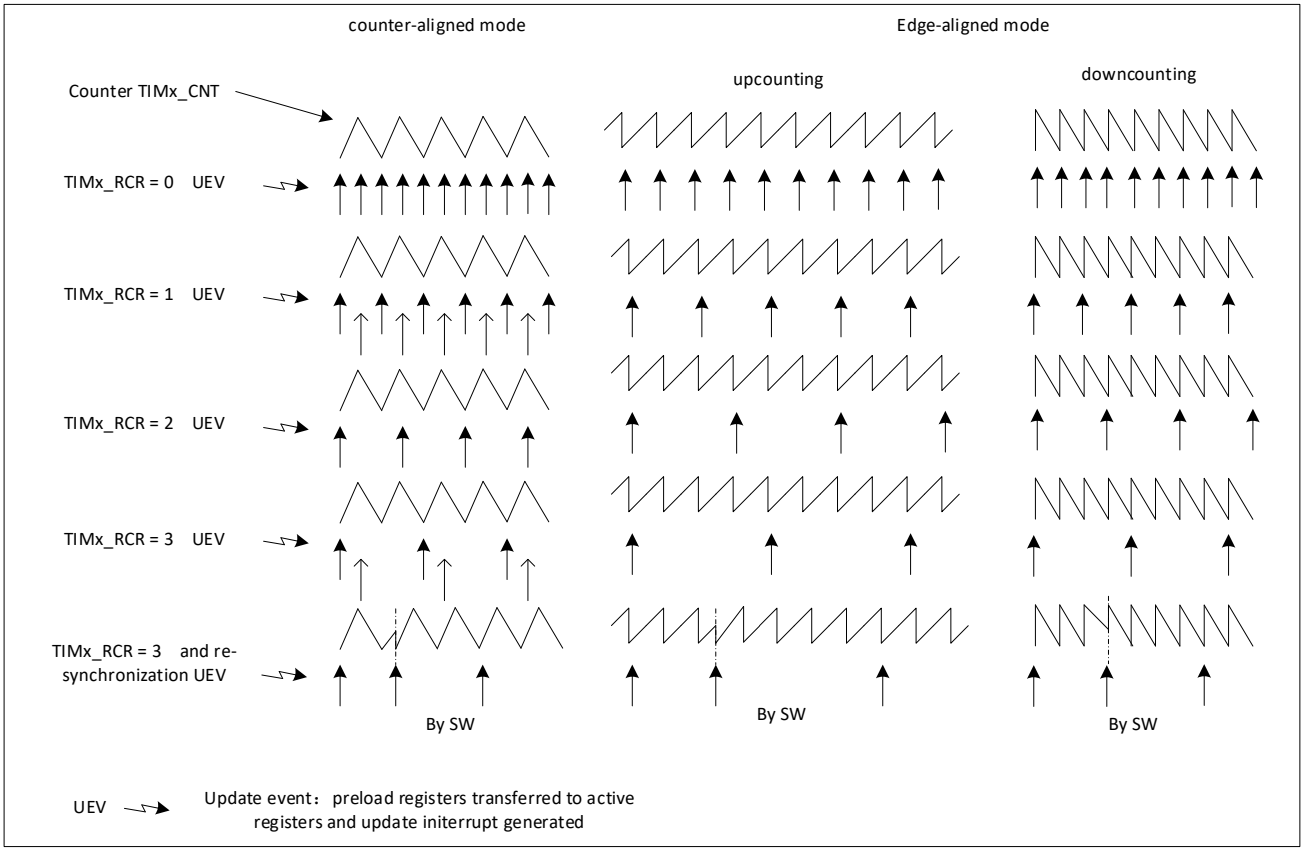


图 15-21 不同模式下更新速率的例子，及 TIM1_RCR 的寄存器设置

15.3.4. 时钟源

计数器的时钟可以由以下时钟源提供：

- 内部时钟（CK_INT）
- 外部时钟模式 1：外部输入引脚
- 外部时钟模式 2：外部触发输入 ETR
- 内部触发输入（ITRx）：使用一个定时器作为另一个定时器的预分频器。例如，可以配置一个定时器 Timer1 作为另一个定时器 Timer16 的预分频器。

内部时钟源（CK_INT）

如果从模式控制器被禁止，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。只要 CEN 位被写成 1，预分频器的时钟就由内部时钟 CK_INT 提供。

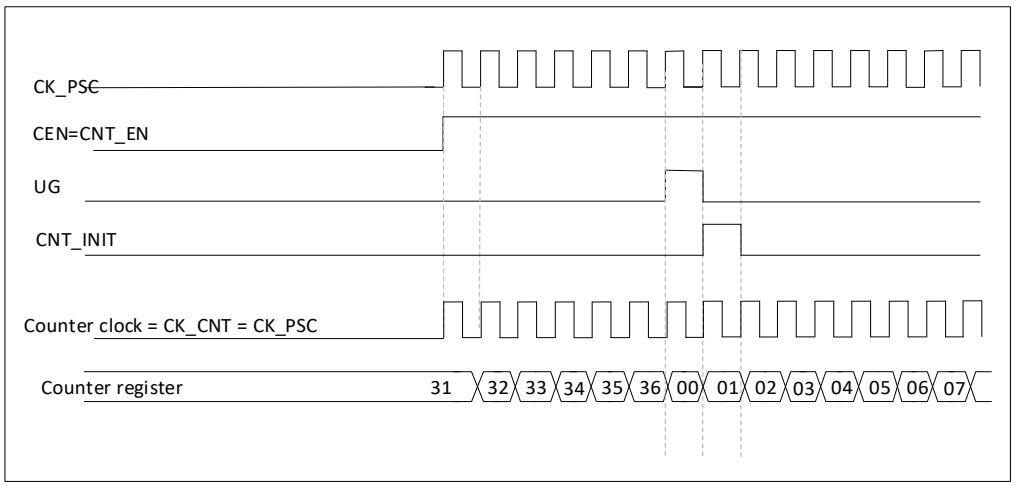


图 15-22 一般模式下的控制电路，内部时钟分频因子为 1

外部时钟源模式 1

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

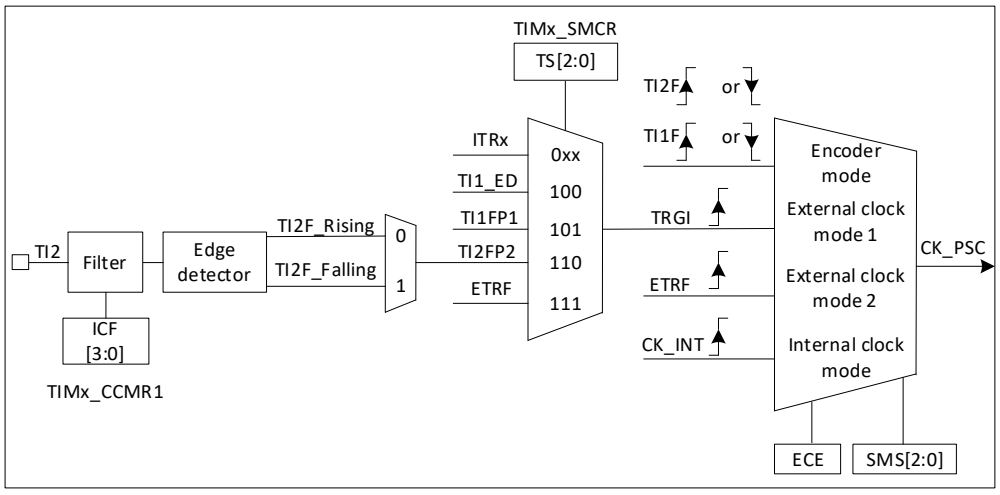


图 15-23 TI2 外部时钟连接例子

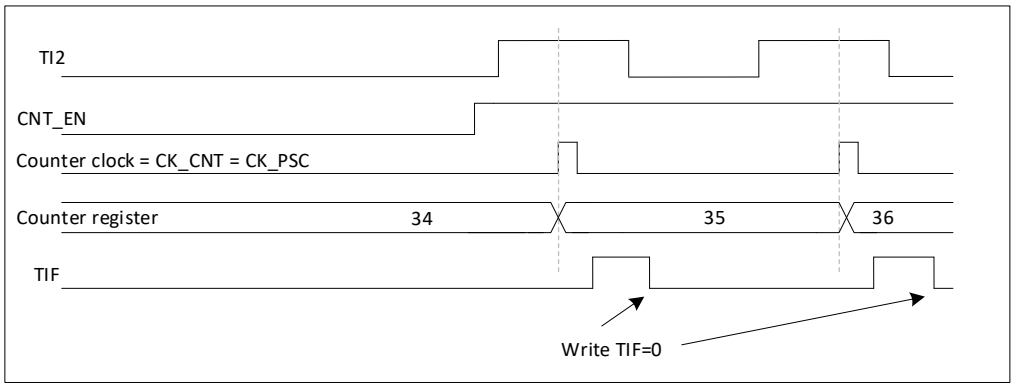


图 15-24 外部时钟模式 1 下的控制电路

外部时钟源模式 2

通过写 TIMx_SMCR 寄存器的 ECE 为 1，选定此模式。计数器能够在外部触发 ETR 的每一个上升沿或下降沿计数。

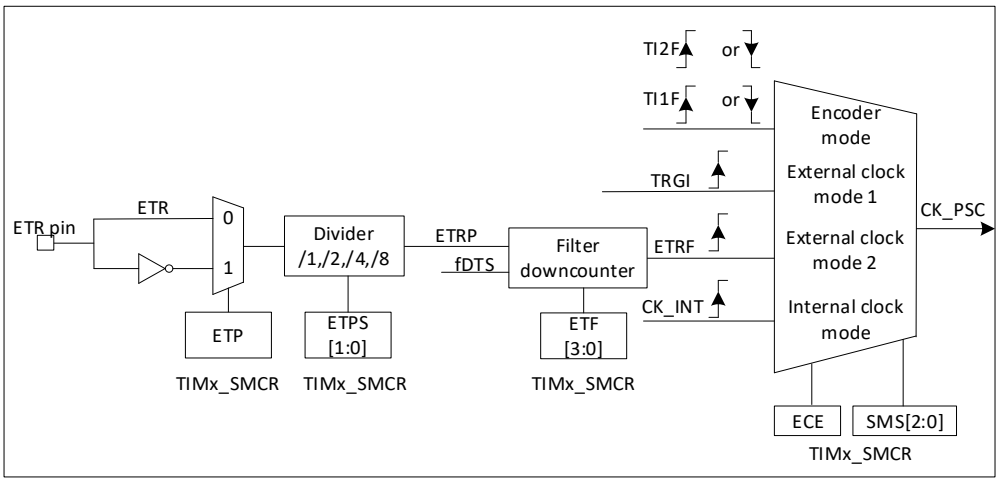


图 15-25 TI2 外部触发输入框图

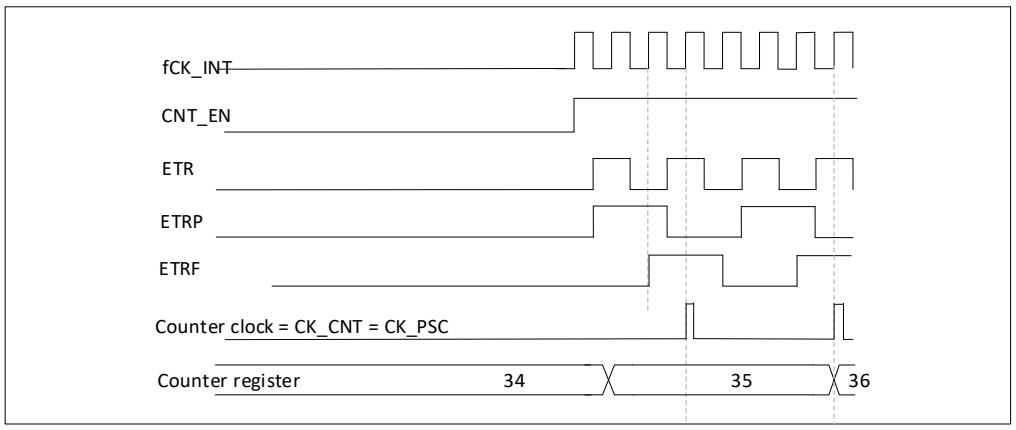


图 15-26 外部时钟模式 2 下的控制电路

15.3.5. 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（输入滤波、多路复用和预分频器），和输出部分（比较器和输出控制）。

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

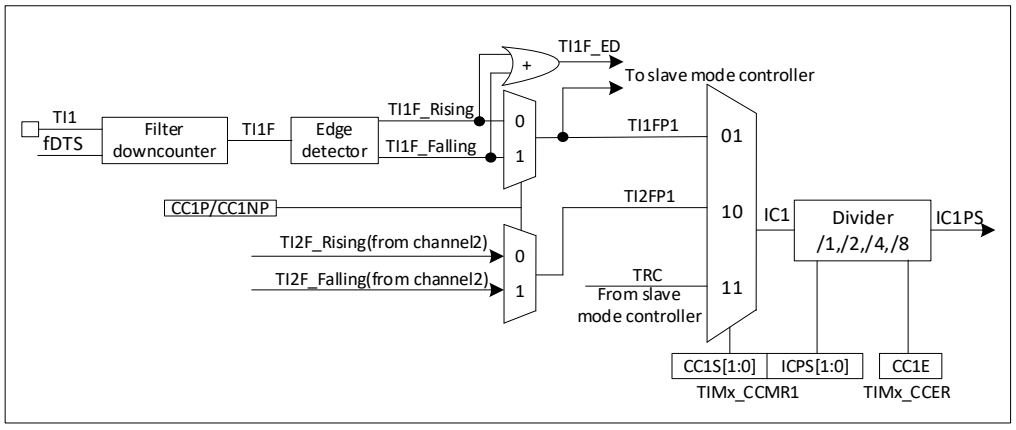


图 15-27 捕获/比较通道(如：通道 1 输入部分)

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

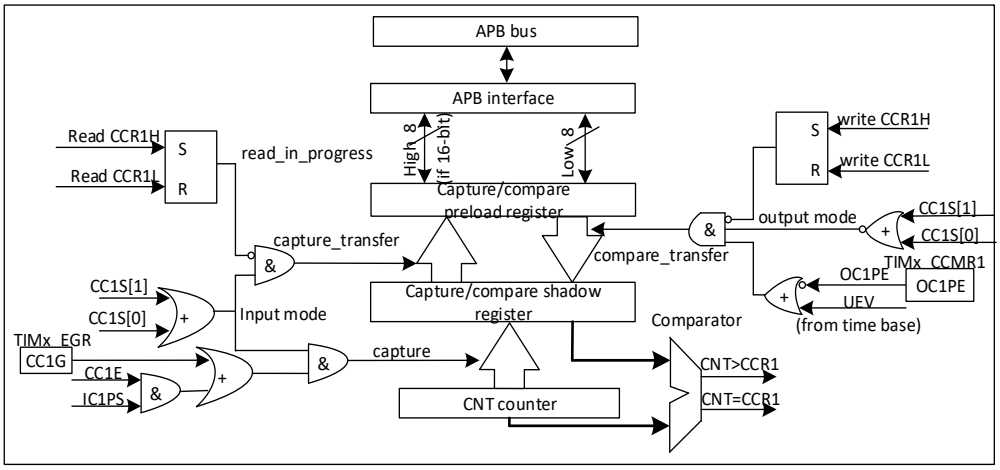


图 15-28 捕获/比较通道 1 的主电路

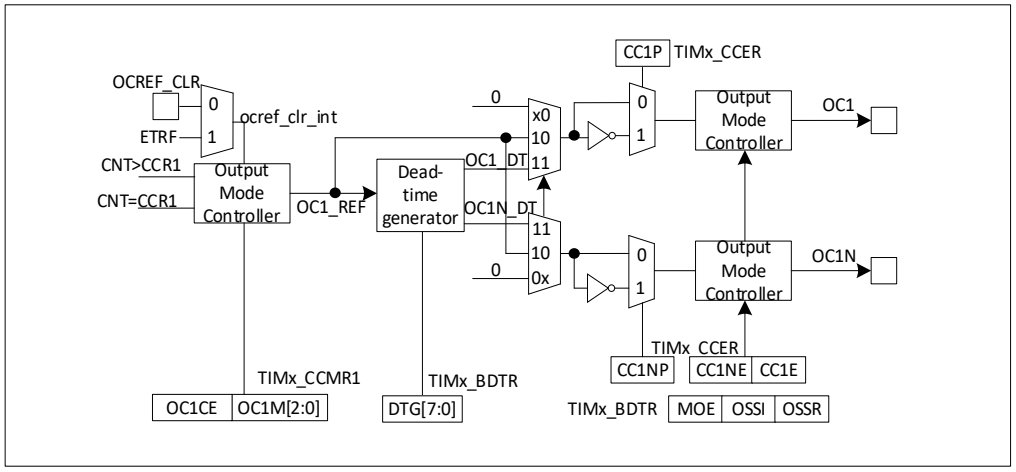


图 15-29 捕获/比较通道的输出部分(通道 1 至 3)

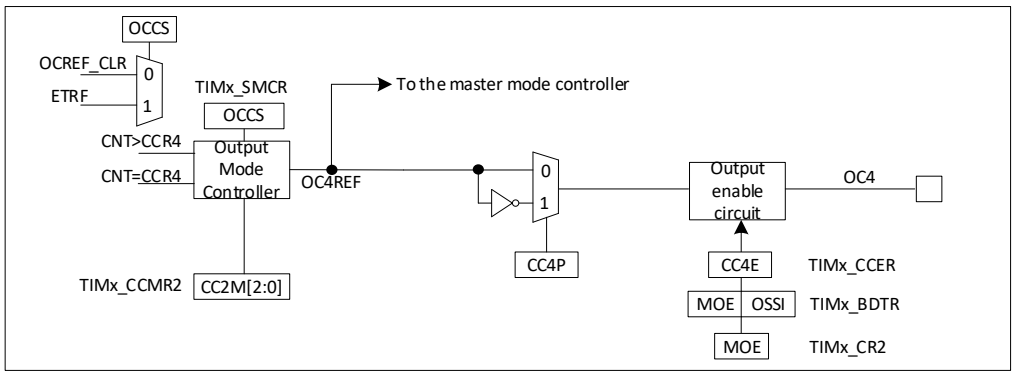


图 15-30 捕获/比较通道的输出部分(通道 4)

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

在捕获模式下，捕获发生在影子寄存器上，然后再复制到预装载寄存器中。

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

15.3.6. 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，如果中断操作被打开，则将产生中断请求。如果发生捕获事件时 CcxIF 标志已经为高，则重复捕获标志 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_CCMR1 必须连接到 TI1 输入，所以写入 TIMx_CCMR1 寄存器中的 CC1S=01，只要 CC1S 不为'00'，通道被配置为输入，并且 TIMx_CCR1 寄存器变为只读。
- 根据输入信号的特点，配置输入滤波器为所需的带宽(即输入为 Tix 时，输入滤波器控制位是 TIMx_CCMRx 寄存器中的 IcxF 位)。假设输入信号在最多 5 个内部时钟周期的时间内抖动，我们须配置滤波器的带宽长于 5 个时钟周期；因此我们可以(以 Fck_int 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。

当发生一个输入捕获时：

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

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请求。

15.3.7. 输入捕获模式（PWM input mode）

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 信号被映射到同一个 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，当需要测量输入到 TI1 上的 PWM 信号的长度(TIMx_CCR1 寄存器)和占空比(TIMx_CCR2 寄存器)时，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。

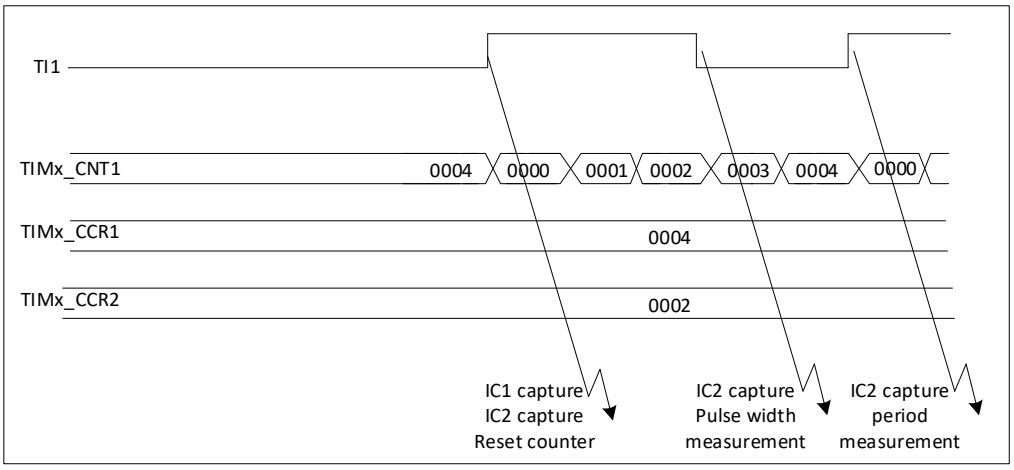


图 15-31 PWM 输入模式时序

15.3.8. 强置输出模式

在输出模式(TIMx_CCMRx 寄存器中 CCxS=00)下，输出比较信号(OCxREF 和相应的 OCx/OCxN)能够直接由软件强置为有效或无效状态，而不依赖于输出比较寄存器和计数器间的比较结果。 置 TIMx_CCMRx 寄存器中相应的 OCxM=101，即可强置输出比较信号(OCxREF/OCx)为有效状态。这样 OCxREF 被强置为高电平(OCxREF 始终为高电平有效)，同时 OCx 得到 CCxP 极性相反的信号。

例如：CCxP=0(OCx 高电平有效)，则 OCx 被强置为高电平。 置 TIMx_CCMRx 寄存器中的 OCxM=100，可强置 OCxREF 信号为低。

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

15.3.9. 输出比较模式

此项功能是用来控制一个输出波形，或者指示一段给定的时间已经到时。当计数器与捕获/比较寄存器的内容相同时，输出比较功能做如下操作：

- 将输出比较模式(TIMx_CCMRx 寄存器中的 OCxM 位)和输出极性(TIMx_CCER 寄存器中的 CCxP 位)定义的值输出到对应的引脚上。在比较匹配时，输出引脚可以保持它的电平(OCxM=000)、被设置成有效电平(OCxM=001)、被设置成无效电平(OCxM=010)或进行翻转(OCxM=011)。
- 设置中断状态寄存器中的标志位(TIMx_SR 寄存器中的 CcxIF 位)。
- 若设置了相应的中断屏蔽(TIMx_DIER 寄存器中的 CcxIE 位)，则产生一个中断。

TIMx_CCMRx 中的 OCxPE 位选择 TIMx_CCRx 寄存器是否需要使用预装载寄存器。 在输出比较模式下，更新事件 UEV 对 OCxREF 和 OCx 输出没有影响。

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

输出比较模式的配置步骤：

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

— 要求计数器与 CCRx 匹配时翻转 OCx 的输出引脚，设置 OCxM=011

— 置 OCxPE = 0 禁用预装载寄存器

— 置 CCxP = 0 选择极性为高电平有效

— 置 CcxE = 1 使能输出

2020. 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。

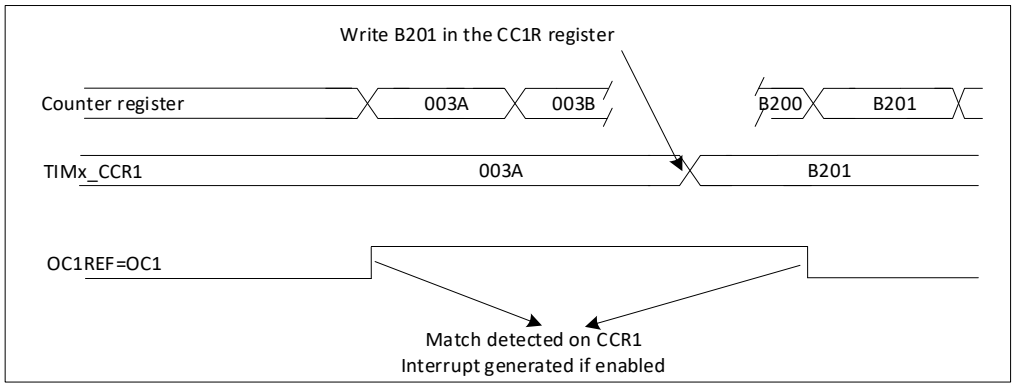


图 15-32 输出比较模式，翻转 OC1

15.3.10. PWM 模式

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

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

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

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

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

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

PWM 边沿对齐模式

- **向上计数配置**

当 TIMx_CR1 寄存器中的 DIR 位为低的时候执行向上计数。参看下面是一个 PWM 模式 1 的例子。当 TIMx_CNT<TIMx_CCRx 时，PWM 参考信号 OCxREF 为高，否则为低。如果 TIMx_CCRx 中的比较值大于自动重装载值(TIMx_ARR)，则 OCxREF 保持为'1'。如果比较值为 0，则 OCxREF 保持为'0'。 下图为 TIMx_ARR=8 时边沿对齐的 PWM 波形实例。

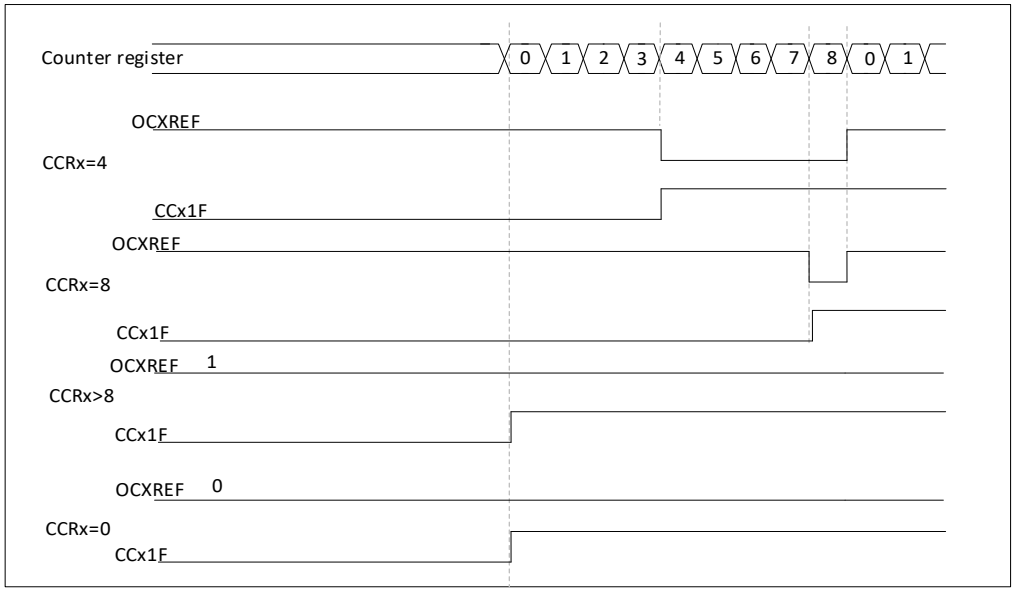


图 15-33 边沿对齐方式 PWM 输出，向上（ARR=8）

- **向下计数的配置**

当 TIMx_CR1 寄存器的 DIR 位为高时执行向下计数。

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

PWM 中央对齐模式

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

下图给出一些中央对齐的 PWM 波形的例子

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

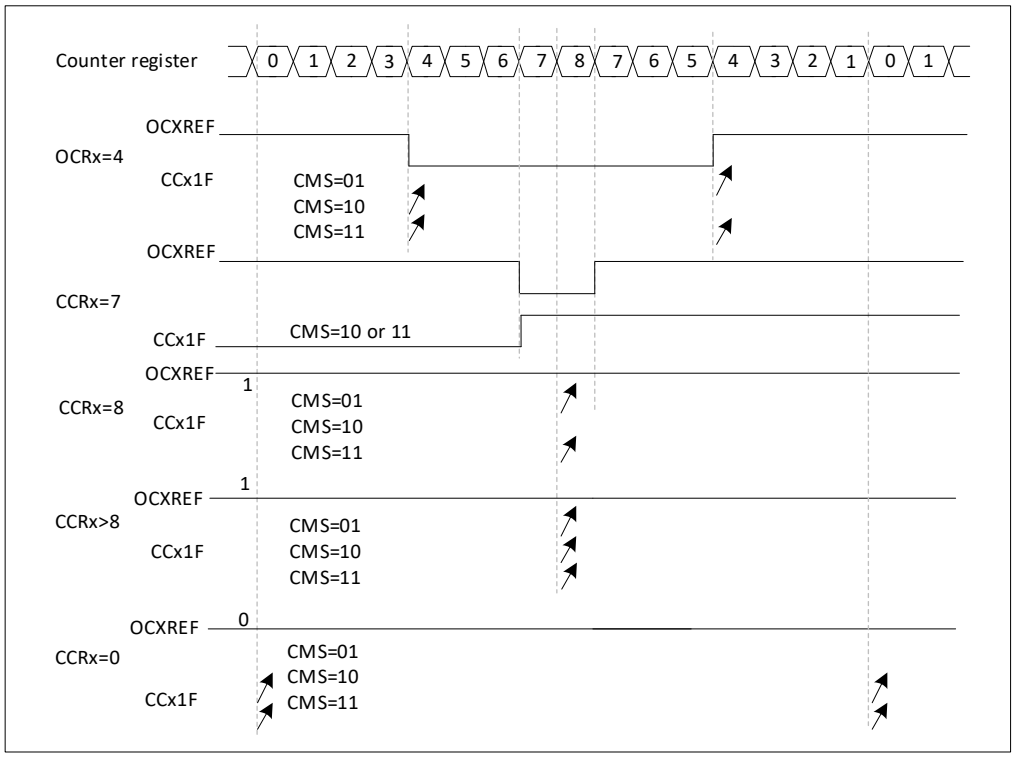


图 15-34 中央对齐的 PWM 波形(APR=8)

使用中央对齐模式的提示：

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

15.3.11. 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 xx 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 8 位的死区发生器 DTG[7:0]。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)

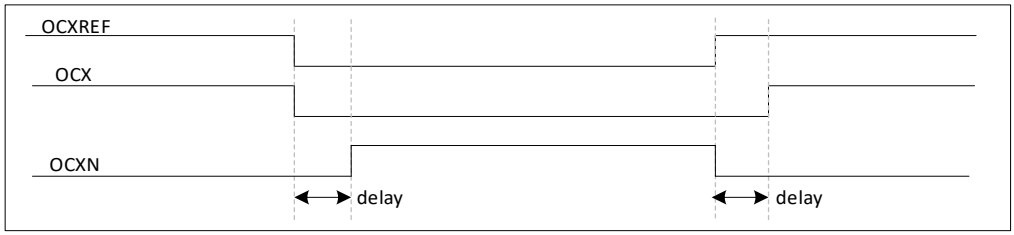


图 15-35 带死区插入的互补输出

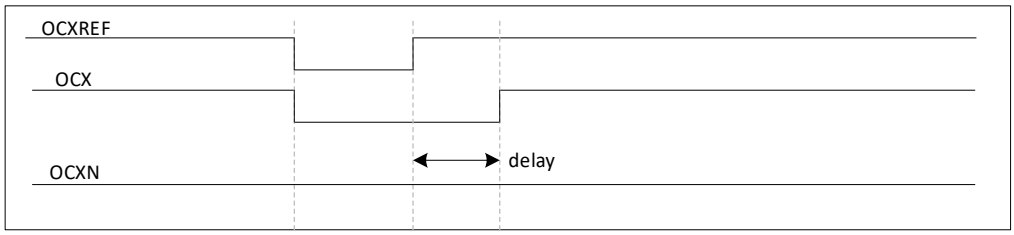


图 15-36 死区波形延迟大于负脉冲

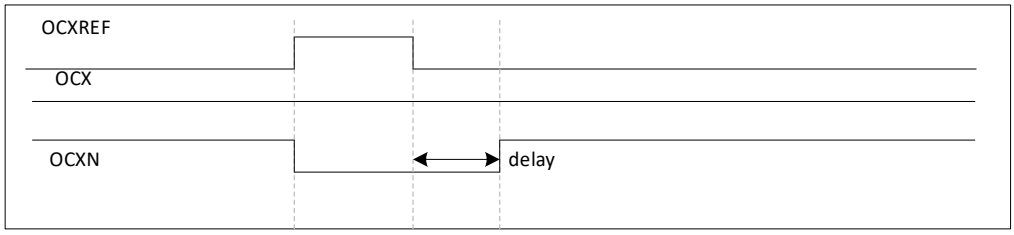


图 15-37 死区波形延迟大于正脉冲

每一个通道的死区延时都是相同的，是由 TIMx_BDTR 寄存器中的 DTG 位编程配置。

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。

15.3.12. 使用刹车功能

当使用刹车功能时，依据额外的控制位，输出使能信号和无效电平信号都会被修改。无论什么情况下，OCx 和 OCxN 输出不能在同一时间同时处于有效电平上。

刹车源既可以是刹车输入引脚，或者以下内部源：

- CPU LOCKUP 输出
- 由 CSS 监测产生的时钟 failure 事件

系统复位后，刹车电路被禁止，MOE 位为低。设置 TIMx_BDTR 寄存器中的 BKE 位可以使能刹车功能，刹车输入信号的极性可以通过配置同一个寄存器中的 BKP 位选择。BKE 和 BKP 可以同时被修改。当写入 BKE 和 BKP 位时，在真正写入之前会有 1 个 APB 时钟周期的延迟，因此需要等待一个 APB 时钟周期之后，才能正确地读回写入的位。

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

当发生刹车时(在刹车输入端出现选定的电平)，有下述动作：

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

刹车可以由 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。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。

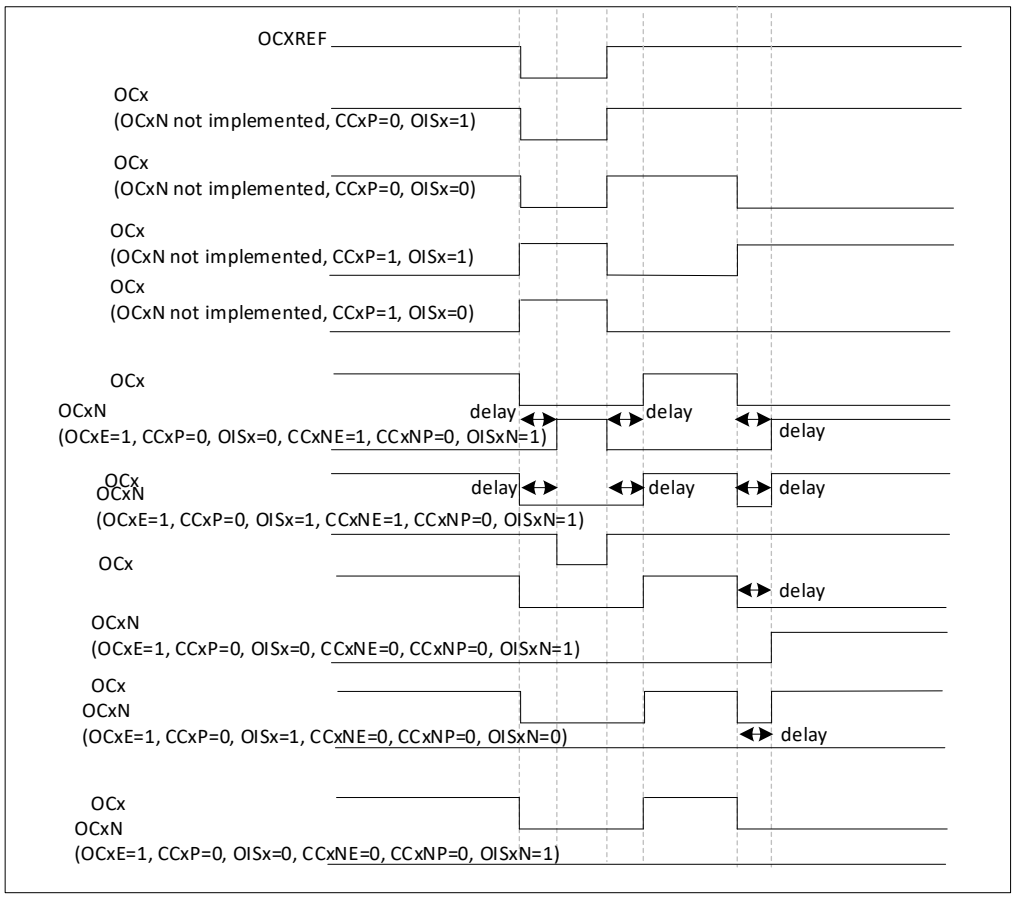


图 15-38 响应刹车的输出

15.3.13. 在外部事件时清除 OCxREF 信号

对于一个给定的通道，设置 TIMx_CCMRx 寄存器中对应的 OCxCE 位为 1，能够用 ETRF 输入端的高电平把 OCxREF 信号拉低，OCxREF 信号将保持为低电平，直到下一次的更新事件 UEV。

该功能只能用于输出比较和 PWM 模式，而不能用于强制模式。

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

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

2. 必须禁止外部时钟模式 2：TIMx_SMCR 寄存器中的 ECE=0。

3. 外部触发极性(ETP)和外部触发滤波器(ETF)可以根据需要配置。

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

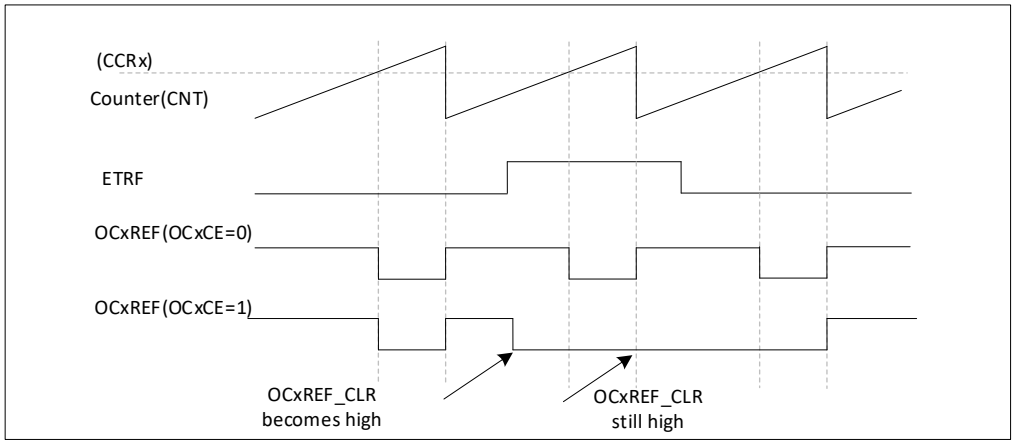


图 15-39 清除 TIM1 的 OCxREF

15.3.14. 六步 PWM 的产生

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

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

15.3.15. 单脉冲模式

单脉冲模式（OPM）是之前所述众多模式中的一个特例。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，产生一个脉宽可被程序控制的脉冲。

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 寄存器的 OPM 位将选择单脉冲模式，这样可以让计数器自动的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

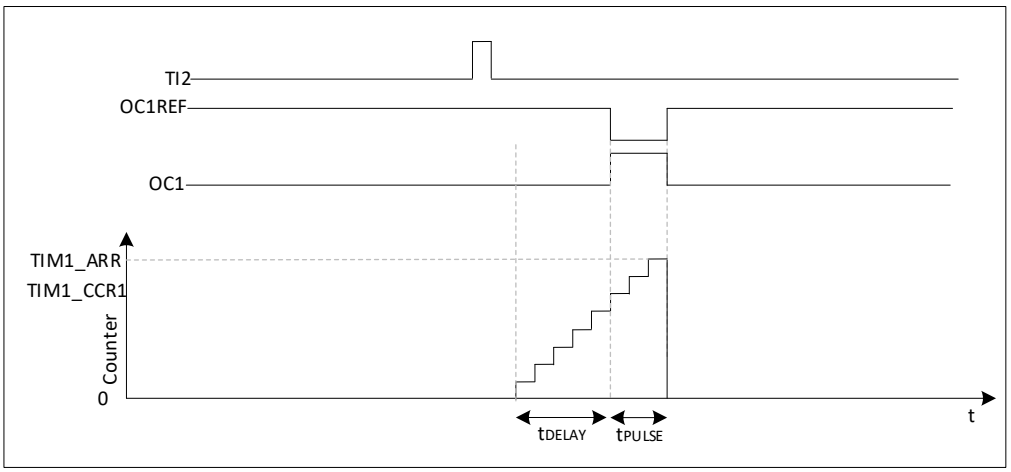


图 15-40 单脉冲模式的例子

例如，当需要在从 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。

使用 TI2FP2 作为触发 1:

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

OPM 的波形由写入比较寄存器的数值决定(要考虑时钟频率和计数器预分频器)

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

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

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

特殊情况：OCx 快速使能：

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

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

15.3.16. 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 table 35，假定计数器已经启动(TIMx_CR1 寄存器中的 CEN=1)，则计数器由每次在 TI1FP1 或 TI2FP2 上的有效跳变驱动。TI1FP1 和 TI2FP2 是 TI1 和 TI2 在通过输入滤波器和极性控制后的信号；如果没有滤波和变相，则 TI1FP1=TI1，TI2FP2=TI2。根据两个输入信号的跳变顺序，产生了计数脉冲和方向信号。依据两个输入信号的跳变顺序，计数器向上或向下计数，同时硬件对 TIMx_CR1 寄存器的 DIR 位进行相应的设置。不管计数器是依靠 TI1 计数、依靠 TI2 计数或者同时依靠 TI1 和 TI2 计数，在任一输入端(TI1 或者 TI2)的跳变都会重新计算 DIR 位。

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

动的修改，因此计数器的内容始终指示着编码器的位置。计数方向与相连的传感器旋转的方向对应。下表列出了所有可能的组合，假设 TI1 和 TI2 不同时变换。

表 15-1 计数方向与编码器信号的关系

Active edge	Level on opposite signal (TI1FP1 for TI2, TI2FP2 for TI1)	TI1FP1 signal		TI2FP2 signal	
		Rising	Falling	Rising	Falling
Counting on TI1 only	High	Down	Up	No count	No count
	Low	Up	Down	No count	No count
Counting on TI2 only	High	No count	No count	Up	Down
	Low	No count	No count	Down	Up
Counting on TI1 and TI2	High	Down	Up	Up	Down
	Low	Up	Down	Down	Up

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

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

- CC1S=’01’(TIMx_CCMR1 寄存器，TI1FP1 映射到 TI1)
- CC2S=’01’(TIMx_CCMR2 寄存器，TI1FP2 映射到 TI2)
- CC1P=’0’(TIMx_CCER 寄存器，TI1FP1 不反相，TI1FP1=TI1)
- CC2P=’0’(TIMx_CCER 寄存器，TI1FP2 不反相，TI1FP2=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)

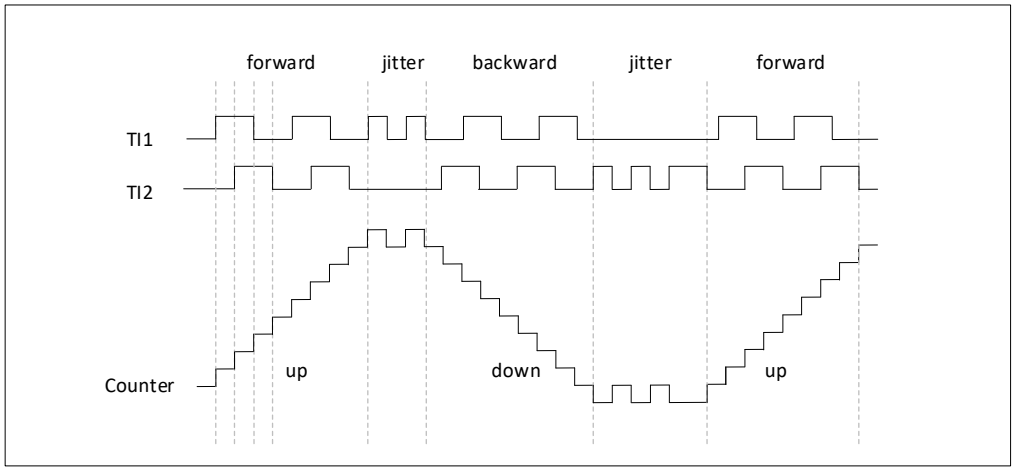


图 15-41 编码器模式下的计数器操作实例

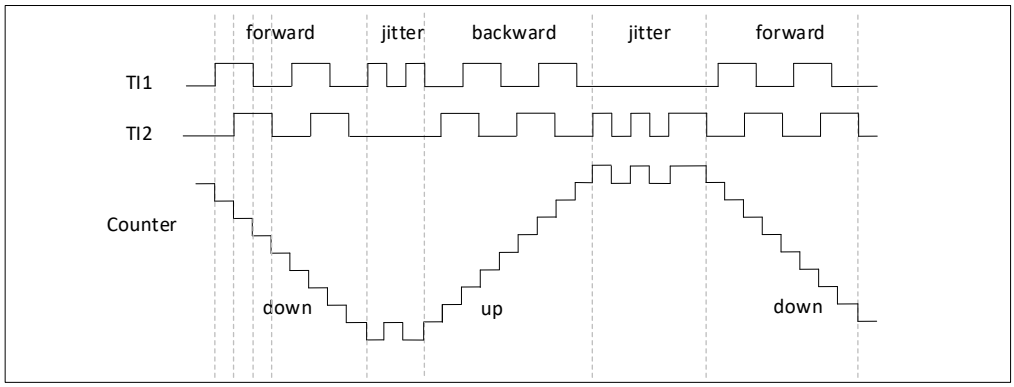


图 15-42 IC1FP1 反相的编码器接口模式实例

当定时器配置成编码器接口模式时，提供传感器当前的位置信息。使用第二个配置在捕获模式的定时器，可以测量两个编码器事件的间隔，获得动态的信息（速度、加速度、减速度）。指示机械零点的编码器输出可被用作此目的。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，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）。

15.3.17. 定时器输入异或功能

TIM_CR2 寄存器的 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。

15.3.18. TIM 和外部的触发同步

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)位的设置，产生一个中断请求。

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。

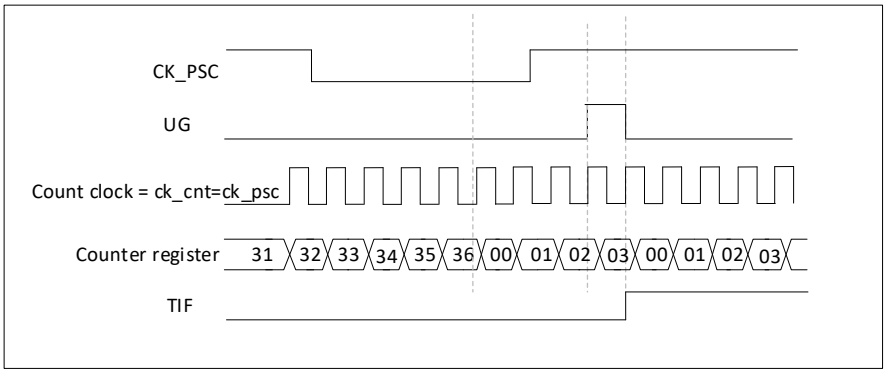


图 15-43 复位模式下的控制电路

从模式：门控模式

按照选中的输入端电平使能计数器。

在如下的例子中，计数器只在 TI1 为低时向上计数：

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

只要 TI1 为低，计数器开始依据内部时钟计数，一旦 TI1 变高则停止计数。当计数器开始或停止时都设置 TIMx_SR 中的 TIF 标置。

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

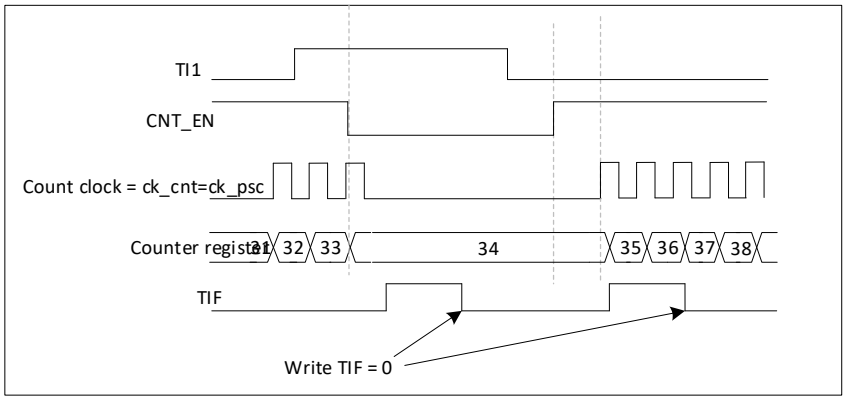


图 15-44 门控模式下的控制电路

输入端上选中的事件使能计数器。

在下面的例子中，计数器在 TI2 输入的上升沿开始向上计数：

- 配置通道 2 检测 TI2 的上升沿。配置输入滤波器带宽(本例中，不需要任何滤波器，保持 IC2F=0000)。触发操作中不使用捕获预分频器，不需要配置。CC2S 位只用于选择输入捕获源，置 TIMx_CCMR1 寄存器中 CC2S=01。置 TIMx_CCER 寄存器中 CC2P=1 以确定极性(只检测低电平)。
- 置 TIMx_SMCR 寄存器中 SMS=110，配置定时器为触发模式；置 TIMx_SMCR 寄存器中 TS=110，选择 TI2 作为输入源。

当 TI2 出现一个上升沿时，计数器开始在内部时钟驱动下计数，同时设置 TIF 标志。 TI2 上升沿和计数器启动计数之间的延时，取决于 TI2 输入端的重同步电路。

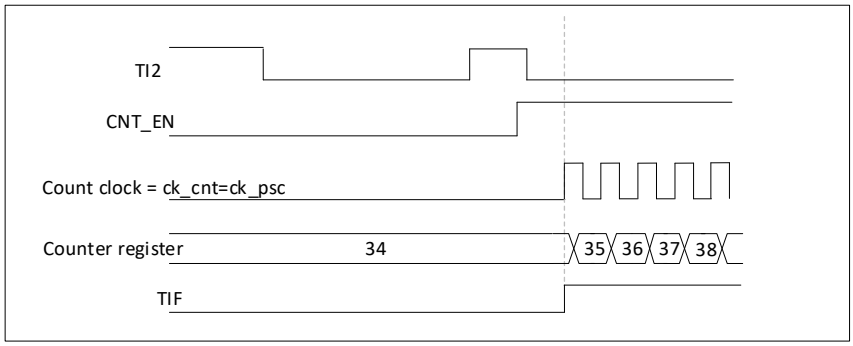


图 15-45 门控模式下的控制电路

从模式：外部时钟模式 2 + 触发模式

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

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

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

当 TI1 上出现一个上升沿时，TIF 标志被设置，计数器开始在 ETR 的上升沿计数。 ETR 信号的上升沿和计数器实际复位间的延时，取决于 ETRP 输入端的重同步电路。

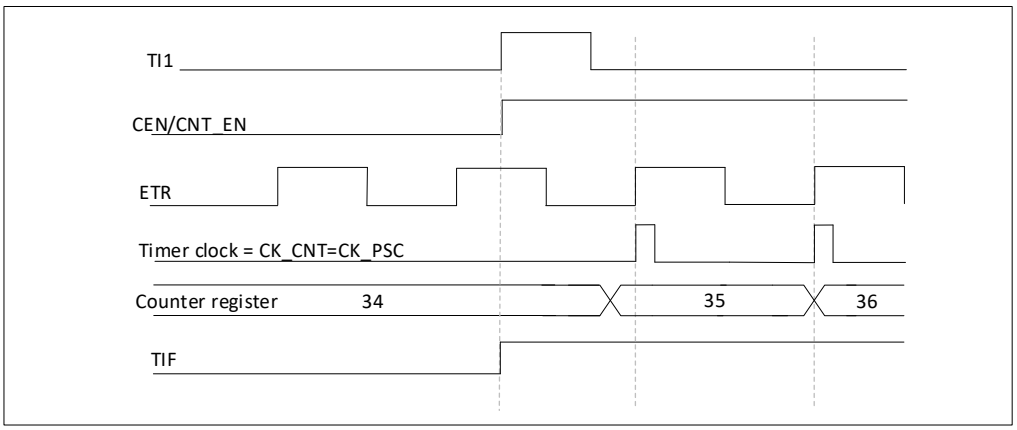


图 15-46 外部时钟模式 2＋触发模式下的控制电路

15.3.19. 定时器同步

TIM 定时器在内部相连，用于 timer 的同步或者链接功能。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、停止或时钟等操作。

15.3.20. 调试模式

当芯片进入调试模式时，根据 DBG 模块中 DBG_TIMx_STOP 的设置，TIMx 计数器可以继续正常工作或者停止工作。

15.4. TIM1 寄存器描述

15.4.1. TIM1 控制寄存器 1 (TIM1_CR1)

Address offset:0x00

Reset value:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res	Res	Res	Res	Res	Res	CKD[1:0]		ARPE	CMS[1:0]		DIR	OPM	URS	UDIS	CEN
-	-	-	-	-	-	RW		RW	RW		RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31：10	Reserved			
9:8	CKD[1:0]	RW	00	时钟分频因子 这 2 位定义在定时器时钟(CK_INT)频率，死区时间和由死区发生器与数字滤波器(ETR,Tix)所用的采样时钟之间的分频比例 00： tDTS = tCK_INT 01： tDTS = 2 x tCK_INT 10： tDTS = 4 x tCK_INT 11：保留，不要使用这个配置
7	ARPE	RW	0	自动重装载预装载允许位 0： TIM1_ARR 寄存器没有缓冲 1： TIM1_ARR 寄存器被装入缓冲器
6:5	CMS[1:0]	RW	00	选择中央对齐模式 00：边沿对齐模式。计数器依据方向位(DIR)向上或向下计数。 01：中央对齐模式 1。计数器交替地向上和向下计数。配置为输出的通道(TIM1_CCMRx 寄存器中 CCxS=00)的输出比较中断标志位，只在计数器向下计数时被设置。 10：中央对齐模式 2。计数器交替地向上和向下计数。计数器交替地向上和向下计数。配置为输出的通道(TIM1_CCMRx 寄存器中 CCxS=00)的输出比较中断标志位，只在计数器向上计数时被设置。 11：中央对齐模式 3。计数器交替地向上和向下计数。计数器交替地向上和向下计数。配置为输出的通道(TIM1_CCMRx 寄存器中 CCxS=00)的输出比较中断标志位，在计数器向上和向下计数时均被设置。 注：在计数器开启时(CEN=1)，不允许从边沿对齐模式转换到中央对齐模式。
4	DIR	RW	0	方向 0：计数器向上计数 1：计数器向下计数 注：当计数器配置为中央对齐模式或编码器模式时，该位为只读
3	OPM	RW	0	单脉冲模式 0：在发生更新事件时，计数器不停止

Bit	Name	R/W	Reset Value	Function
				1：在发生下一次更新事件(清除 CEN 位)时，计数器停止。
2	URS	RW	0	更新请求源 软件通过该位选择 UEV 事件的源 0：如果允许产生更新中断，则下述任一事件产生一个更新中断： – 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，则只有计数器溢出/下溢产生一个更新中 断
1	UDIS	RW	0	禁止更新 软件通过该位允许/禁止 UEV 事件的产生 0：允许 UEV。更新(UEV)事件由下述任一事件产生： – 计数器溢出/下溢 – 设置 UG 位 – 从模式控制器产生的更新 被缓存的寄存器被装入它们的预装载值。 1：禁止 UEV。不产生更新事件，影子寄存器(ARR,PSC,CCRx)保持它们的值。 如果设置了 UG 位或从模式控制器发出了一个硬件复位，则计数器和预分频器被重新初始化。
0	CEN	RW	0	允许计数器 0：禁止计数器 1：开启计数器 注：在软件设置了 CEN 位后，外部时钟、门控模式和编码器模式才能工作。触发模式可以自动地通过硬件设置 CEN 位。

15.4.2. TIM1 控制寄存器 2 (TIM1_CR2)

Address offset:0x04

Reset value:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res	OIS4	OIS3N	OIS3	OIS2N	OIS2	OIS1N	OIS1	TI1S	MMS[2:0]			Res	CCUS	Res	CCPC
-	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW		RW	-	RW

Bit	Name	R/W	Reset Value	Function
31：15	Reserved	-	0	保留，始终为 0
14	OIS4	RW		输出空闲状态 4(OC4 输出)。参见 OIS1 位。
13	OIS3N	RW	0	输出空闲状态 3(OC3N 输出)。参见 OIS1N 位
12	OIS3	RW	0	输出空闲状态 3(OC3 输出)。参见 OIS1 位。
11	OIS2N	RW	0	输出空闲状态 2(OC2N 输出)。参见 OIS1N 位。
10	OIS2	RW	0	输出空闲状态 2(OC2 输出)。参见 OIS1 位
9	OIS1N	RW	0	输出空闲状态 1(OC1N 输出)。 0：当 MOE=0 时，死区后 OC1N=0 1：当 MOE=0 时，死区后 OC1N=1 注：已经设置了 LOCK(TIM1_BKR 寄存器)级别 1、 2 或 3 后，该位不能被修改。
8	OIS1	RW	0	输出空闲状态 1(OC1 输出)。 0：当 MOE=0 时，如果实现了 OC1N，则死区后 OC1=0 1：当 MOE=0 时，如果实现了 OC1N，则死区后 OC1=1

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

15.4.3. TIM1 从模式控制寄存器 (TIM1_SMCR)

Address offset:0x08

Reset value:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
ETP	ECE	ETPS[1:0]		ETF[3:0]				MSM	TS[2:0]			OCCS	SMS[2:0]		
RW	RW	RW		RW				RW	RW			RW	RW		

Bit	Name	R/W	Reset Value	Function
31：16	Reserved			
15	ETP	RW	0	外部触发极性。该位选择是否 ETR 或者 ETR 的反向被用作触发操作。 0：ETR 不进行反向，高电平或者上升沿有效 1：ETR 反向，低电平或者下降沿有效
14	ECE	RW	0	外部时钟使能。这位使能外部时钟模式 2 0：外部时钟模式 2 不使能 1：外部时钟模式 2 使能，计数器工作在 ETRF 信号的有效沿
13：12	ETPS[1:0]	RW	00	外部触发预分频器。外部触发信号 ETRP 频率必须至多 TIM1CLK 频率的 1/4。一个预分频器可以被使能，以降低 ETRP 的频率。当输入快速外部时钟是有效的。 00：预分频器关闭 01：ETRP 频率的 2 分频 10：ETRP 频率的 4 分频 11：ETRP 频率的 8 分频
11：8	ETF[3:0]	RW	0000	外部触发滤波。这些位定义采样 ETRP 信号的频率和应用在 ETRP 的数字滤波长度。这个数字滤波由一个事件计数器组成，在改计数器里，N 个连续的事件被需要使输出的边沿有效。 0000：没有滤波器，在 fDTS 下采样 0001：fSAMPLING=fCK_INT, N=2 0010：fSAMPLING=fCK_INT, N=4 0011：fSAMPLING=fCK_INT, N=8 0100：fSAMPLING=fCK_INT/2, N=6 0101：fSAMPLING=fCK_INT/2, N=8 0110：fSAMPLING=fCK_INT/4, N=6 0111：fSAMPLING=fCK_INT/4, N=8 1000：fSAMPLING=fCK_INT/8, N=6 1001：fSAMPLING=fCK_INT/8, N=8 1010：fSAMPLING=fCK_INT/16, N=5 1011：fSAMPLING=fCK_INT/16, N=6 1100：fSAMPLING=fCK_INT/16, N=8 1101：fSAMPLING=fCK_INT/32, N=5 1110：fSAMPLING=fCK_INT/32, N=6 1111：fSAMPLING=fCK_INT/32, N=8 必须关注当 ETF[3:0] = 1 或者 2 或者 3 时，fDTS 被方程式中的 CK_INT 代替
7	MSM	RW	0	主/从模式 0：无作用 1：触发输入(TRGI)上的事件被延迟了，以允许在当前定时器(通过 TRGO)与它的当前定时器和从定时器间的同步（通过 TRGO）。这对要求把几个定时器同步到一个单一的外部事件时是非常有用的
6：4	TS[2:0]	RW	000	触发选择，这 3 位选择用于同步计数器的触发输入。 000：Reserved(ITR0) 001：Reserved(ITR1) 010：Reserved (ITR2) 011：Reserved (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) 注：为避免在信号转变时产生错误的边沿检测，必须在未使用这些位时修改它们
3	OCCS	RW	0	OCREF 清除选择位。该位用于选择 OCREF 的清除源。 0：OCREF_CLR_INT 连接到 OCREF_CLR 输入 1：OCREF_CLR_INT 连接到 ETRF
2：0	SMS[2:0]	RW	000	从模式选择。当选择了外部信号，触发信号(TRGI)的有效边沿与选中的外部输入极性相关(见输入控制寄存器和控制寄存器的说明)

Bit	Name	R/W	Reset Value	Function
				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 TI1FP2 的电平，计数器在 TI2FP1 的边沿向上/下计数。 010：编码器模式 2 根据 TI2FP1 的电平，计数器在 TI1FP2 的边沿向上/下计数。 011：编码器模式 3 模式 1 和模式 2 的综合 100：复位模式 选中的触发输入(TRGI)的上升沿重新初始化计数器，并且产生一个更新寄存器的信号。 101：门控模式 当触发输入(TRGI)为高时，计数器的时钟开启。一旦触发输入变为低，则计数器停止(但不复位)。计数器的启动和停止都是受控的。 110：触发模式 计数器在触发输入 TRGI 的上升沿启动(但不复位)，只有计数器的启动是受控的。 111：外部时钟模式 1 选中的触发输入(TRGI)的上升沿驱动计数器。 注：如果 TI1F_EN 被选为触发输入(TS=100)时，不要使用门控模式。这是因为，TI1F_ED 在每次 TI1F 变化时输出一个脉冲，然而门控模式是要检查触发输入的电平。

TIM1 内部触发连接

Slave TIM	ITR0(TS=000)	ITR1(TS=001)	ITR2(TS=010)	ITR3(TS=011)
TIM1	reserved	reserved	reserved	reserved

15.4.4. TIM1 中断使能寄存器 (TIM1_DIER)

Address offset:0x0C

Reset value:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res	Res	Res	Res	Res	Res	Res	Res	**BIE**	**TIE**	Res	**CC4IE**	**CC3IE**	**CC2IE**	**CC1IE**	**UIE**
-								RW	RW		RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31：8	Reserved			保留，一直为 0
7	BIE	RW	0	BIE：允许刹车中断 0：禁止刹车中断 1：允许刹车中断
6	TIE	RW	0	TIE：允许触发中断 0：禁止触发中断 1：允许触发中断
5	COMIE	RW	0	COMIE：允许 COM 中断 0：禁止 COM 中断 1：允许 COM 中断
4	CC4IE	RW	0	CC4IE：允许捕获/比较 4 中断 0：禁止捕获/比较 4 中断 1：允许捕获/比较 4 中断
3	CC3IE	RW	0	CC3IE：允许捕获/比较 3 中断 0：禁止捕获/比较 3 中断 1：允许捕获/比较 3 中断
2	CC2IE	RW	0	CC2IE：允许捕获/比较 2 中断 0：禁止捕获/比较 2 中断 1：允许捕获/比较 2 中断

Bit	Name	R/W	Reset Value	Function
1	CC1IE	RW	0	CC1IE：允许捕获/比较 1 中断 0：禁止捕获/比较 1 中断 1：允许捕获/比较 1 中断
0	UIE	RW	0	UIE：允许更新中断 0：禁止更新中断 1：允许更新中断

15.4.5. TIM1 状态寄存器(TIM1_SR)

Address offset:0x010

Reset value:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res	Res	Res	CC4OF	CC3OF	CC2OF	CC1OF	Res	BIF	TIF	COMIF	CC4IF	CC3IF	CC2IF	CC1IF	UIF
-	-	-	Rc_w0	Rc_w0	Rc_w0	Rc_w0	-	Rc_w0	Rc_w0	Rc_w0	Rc_w0	Rc_w0	Rc_w0	Rc_w0	Rc_w0

Bit	Name	R/W	Reset Value	Function
31：13	Reserved	-	0	保留，一直为 0
12	CC4OF	Rc_w0	0	捕获/比较 4 过捕获标记 参见 CC1OF 描述
11	CC3OF	Rc_w0	0	捕获/比较 3 过捕获标记 参见 CC1OF 描述
10	CC2OF	Rc_w0	0	捕获/比较 2 过捕获标记 参见 CC1OF 描述
9	CC1OF	Rc_w0	0	捕获/比较 1 过捕获标记 仅当相应的通道被配置为输入捕获时，该标记可由硬件置 1。写 0 可清除该位。 0：无过捕获产生； 1： CC1OF 置 1 时，计数器的值已经被捕获到 TIM1_CCR1 寄存器。
8	Res	Rc_w0	0	保留，始终读为 0。
7	BIF	Rc_w0	0	刹车中断标记 一旦刹车输入有效，由硬件对该位置 1。如果刹车输入无效，则该位可由软件清 0。 0：无刹车事件产生； 1：刹车输入上检测到有效电平。
6	TIF	Rc_w0	0	触发器中断标记 当发生触发事件（当从模式控制器处于除门控模式外的其它模式时,在 TRGI 输入端检测到有 效边沿，或或门控模式下的任一边沿）时由硬件对该位置 1。它由软件清 0。 0：无触发器事件产生； 1：触发器中断等待响应
5	COMIF	Rc_w0	0	COM 中断标记 一旦产生 COM 事件（当 CcxE、 CcxNE、 OCxM 已被更新）该位由硬件置 1。它由软件清 0。 0：无 COM 事件产生； 1： COM 中断等待响应
4	CC4IF	Rc_w0	0	捕获/比较 4 中断标记 参考 CC1IF 描述
3	CC3IF	Rc_w0	0	捕获/比较 3 中断标记 参考 CC1IF 描述
2	CC2IF	Rc_w0	0	捕获/比较 2 中断标记

Bit	Name	R/W	Reset Value	Function
				参考 CC1IF 描述
1	CC1IF	Rc_w0	0	捕获/比较 1 中断标记 如果通道 CC1 配置为输出模式： 当计数器值与比较值匹配时该位由硬件置 1，但在中心对称模式下除外(参考 TIM1_CR1 寄存器的 CMS 位)。它由软件清 0。 0：无匹配发生； 1： TIM1_CNT 的值与 TIM1_CCR1 的值匹配。 如果通道 CC1 配置为输入模式： 当捕获事件发生时该位由硬件置 1，它由软件清 0 或通过读 TIM1_CCR1 清 0。 0：无输入捕获产生； 1：输入捕获产生并且计数器值已装入 TIM1_CCR1(在 IC1 上检测到与所选极性相同的边沿)。 注：当 CEN 打开，该位也会被置位。
0	UIF	Rc_w0	0	更新中断标记 当产生更新事件时该位由硬件置 1。它由软件清 0。 0：无更新事件产生； 1：更新事件等待响应。当寄存器被更新时该位由硬件置 1： – 若 TIM1_CR1 寄存器的 UDIS=0，当 REP_CNT=0 时产生更新事件(重复向下计数器上溢或下溢时)； – 若 TIM1_CR1 寄存器的 UDIS=0、 URS=0，当 TIM1_EGR 寄存器的 UG=1 时产生更新事件(软件对 CNT 重新初始化)； – 若 TIM1_CR1 寄存器的 UDIS=0、 URS=0，当 CNT 被触发事件重初始化时产生更新事件。（参考从模式控制寄存器(TIM1_SMCR)）

15.4.6. TIM1 事件产生寄存器(TIM1_EGR)

Address offset:0x14

Reset value:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res	Res	Res	Res	Res	Res	Res	Res	BG	TG	COMG	CC4G	CC3G	CC2G	CC1G	UG
-	-	-	--	-	-	-	-	W	W	W	W	W	W	W	W

Bit	Name	R/W	Reset Value	Function
31：8	Reserved	-	0	保留，一直为 0
7	BG	W	0	产生刹车事件 该位由软件置 1，用于产生一个刹车事件，由硬件自动清 0。 0：无动作； 1：产生一个刹车事件。此时 MOE=0、 BIF=1。
6	TG	W	0	产生触发事件 该位由软件置 1，用于产生一个触发事件，由硬件自动清 0。 0：无动作； 1： TIM1_SR 寄存器的 TIF=1，若开启对应的中断，则产生相应的中断。
5	COMG	W	0	捕获/比较事件，产生控制更新 该位由软件置 1，由硬件自动清 0。 0：无动作； 1：当 CCPC=1，允许更新 CcxE、 CcxNE、 OCxM 位。

Bit	Name	R/W	Reset Value	Function
				注： 该位只对有互补输出的通道有效。
4	CC4G	W	0	产生捕获/比较 4 事件 参考 CC1G 描述
3	CC3G	W	0	产生捕获/比较 3 事件 参考 CC1G 描述
2	CC2G	W	0	产生捕获/比较 2 事件 参考 CC1G 描述
1	CC1G	W	0	产生捕获/比较 1 事件 该位由软件置 1，用于产生一个捕获/比较事件，由硬件自动清 0。 0：无动作； 1：在通道 CC1 上产生一个捕获/比较事件： 若通道 CC1 配置为输出： 设置 CC1IF=1，若开启对应的中断，则产生相应的中断。 若通道 CC1 配置为输入： 当前的计数器值捕获至 TIM1_CCR1 寄存器，设置 CC1IF=1，若开启对应的中断，则产生相应的中断。若 CC1IF 已经为 1，则设置 CC1OF=1。
0	UG	W	0	产生更新事件。该位由软件置 1，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)则计数器装载 TIM1_ARR 的值。

15.4.7. TIM1 捕获/比较模式寄存器 1(TIM1_CCMR1)

Address offset:0x18

Reset value:0x0000 0000

Output compare mode:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
OC2CE	OC2M[2:0]			OC2PE	CO2FE	CC2S[1:0]		OC1CE	OC1M[2:0]			OC1PE	OC1FE	CC1S[1:0]	
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31：16	Reserved	-		保留，一直为 0
15	OC2CE	RW	0	输出比较 2 清 0 使能
14:12	OC2M[2:0]	RW	000	输出比较 2 模式选择
11	OC2PE	RW	0	输出比较 2 预装载使能
10	OC2FE	RW	0	输出比较 2 快速使能
9:8	CC2S[1:0]	RW	00	捕获/比较 2 选择。 该位定义通道的方向（输入/输出），及输入脚的选择： 00：CC2 通道被配置为输出； 01：CC2 通道被配置为输入，IC2 映射在 TI2 上； 10：CC2 通道被配置为输入，IC2 映射在 TI1 上； 11：CC2 通道被配置为输入，IC2 映射在 TRC 上。此模式仅工作在内部触发器输入被选中时 （由 TIM1_SMCR 寄存器的 TS 位选择）。 注： CC2S 仅在通道关闭时(TIM1_CCER 寄存器的 CC2E=0)才是可写的。
7	OC1CE	RW	0	输出比较 1 清 0 使能 0： OC1REF 不受 ETRF 输入的影响； 1：一旦检测到 ETRF 输入高电平，清除 OC1REF=0。

Bit	Name	R/W	Reset Value	Function
6:4	OC1M[2:0]	RW	00	输出比较 1 模式 该位定义了输出参考信号 OC1REF 的动作，而 OC1REF 决定了 OC1、 OC1N 的值。 OC1REF 是高电平有效，而 OC1、 OC1N 的有效电平取决于 CC1P、 CC1NP 位。 000：冻结。输出比较寄存器 TIM1_CCR1 与计数器 TIM1_CNT 间的比较对 OC1REF 不起作用； 001：匹配时设置通道 1 为有效电平。当计数器 TIMx_CNT 的值与捕获 / 比较寄存器 1(TIMx_CCR1) 相同时，强制 OC1REF 为高。 010：匹配时设置通道 1 为无效电平。当计数器 TIMx_CNT 的值与捕获 / 比较寄存器 1(TIMx_CCR1)相同时，强制 OC1REF 为低。 011：翻转。当 TIM1_CCR1=TIM1_CNT 时，翻转 OC1REF 的电平。 100：强制为无效电平。强制 OC1REF 为低。 101：强制为有效电平。强制 OC1REF 为高。 110： PWM 模式 1－ 在向上计数时，一旦 TIM1_CNT<TIM1_CCR1 时通道 1 为有效电平，否则为无效电平；在向下计数时，一旦 TIM1_CNT>TIM1_CCR1 时通道 1 为无效电平 (OC1REF=0)，否则为有效电平(OC1REF=1)。 111： PWM 模式 2－ 在向上计数时，一旦 TIM1_CNT<TIM1_CCR1 时通道 1 为无效电平，否则为有效电平；在向下计数时，一旦 TIM1_CNT>TIM1_CCR1 时通道 1 为有效电平，否则为无效电平。 注 1：一旦 LOCK 级别设为 3(TIMx_BDTR 寄存器中的 LOCK 位)并且 CC1S=00(该通道配置成输出)则该位不能被修改。 注 2：在 PWM 模式 1 或 PWM 模式 2 中，只有当比较结果改变了或在输出比较模式中从冻结模式切换到 PWM 模式时， OC1REF 电平才改变。
3	OC1PE	RW	0	输出比较 1 预装载使能 0：禁止 TIM1_CCR1 寄存器的预装载功能，可随时写入 TIM1_CCR1 寄存器，且新值马上起作用。 1：开启 TIM1_CCR1 寄存器的预装载功能，读写操作仅对预装载寄存器操作， TIM1_CCR1 的预装载值在更新事件到来时被载入当前寄存器中。 注 1：一旦 LOCK 级别设为 3(TIMx_BDTR 寄存器中的 LOCK 位)并且 CC1S=00(该通道配置成输出)则该位不能被修改。 注 2：仅在单脉冲模式下，可以在未确认预装载寄存器情况下使用 PWM 模式，否则其动作不确定。
2	OC1FE	RW	0	输出比较 1 快速使能 该位用于加快 CC 输出对触发器输入事件的响应。 0：根据计数器与 CCR1 的值， CC1 正常操作，即使触发器是打开的。当触发器的输入有一个有效沿时，激活 CC1 输出的最小延时为 5 个时钟周期。 1：输入到触发器的有效沿的作用就象发生了一次比较匹配。因此， OC 被设置为比较电平而与比较结果无关。采样触发器的有效沿和 CC1 输出间的延时被缩短为 3 个时钟周期。 OCFE 的只在通道被配置成 PWM1 或 PWM2 模式时起作用。
1:0	CC1S[1:0]	RW	00	捕获/比较 1 选择。 这 2 位定义通道的方向（输入/输出），及输入脚的选择： 00：CC1 通道被配置为输出； 01：CC1 通道被配置为输入，IC1 映射在 TI1 上； 10：CC1 通道被配置为输入，IC1 映射在 TI2 上； 11：CC1 通道被配置为输入，IC1 映射在 TRC 上。此模式仅工作在内部触发器输入被选中时

Bit	Name	R/W	Reset Value	Function
				（由 TIM1_SMCR 寄存器的 TS 位选择）。 注： CC1S 仅在通道关闭时(TIM1_CCER 寄存器的 CC1E=0)才是可写的。

Input Capture mode:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
IC2F[3:0]				IC2PSC[1:0]		CC2S[1:0]		IC1F[3:0]				IC1PSC[1:0]		CC1S[1:0]	
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:16	Reserved	-		保留，一直为 0
15:12	IF2F	RW	0000	输入捕获 2 滤波器
11:10	IC2PSC[1:0]	RW	00	输入/捕获 2 预分频器
9:8	CC2S[1:0]	RW	0	捕获/比较 2 选择。 这 2 位定义通道的方向（输入/输出），及输入脚的选择： 00：CC2 通道被配置为输出； 01：CC2 通道被配置为输入，IC2 映射在 TI2 上； 10：CC2 通道被配置为输入，IC2 映射在 TI1 上； 11：CC2 通道被配置为输入，IC2 映射在 TRC 上。此模式仅工作在内部触发器输入被选中时 （由 TIM1_SMCR 寄存器的 TS 位选择）。 注： CC2S 仅在通道关闭时(TIM1_CCER 寄存器的 CC2E=0)才是可写的。
7:4	IC1F[3:0]	RW	0000	输入捕获 1 滤波器 这几位定义了 TI1 输入的采样频率及数字滤波器长度。数字滤波器由一个事件计数器组成， 需要 N 个连续事件来验证输出上的转换： 0000：无滤波器，以 fDTS 采样 1000：采样频率 fSAMPLING=fDTS/8， N=6 0001：采样频率 fSAMPLING=fCK_INT，N=2 1001：采样频率 fSAMPLING=fDTS/8，N=8 0010：采样频率 fSAMPLING=fCK_INT，N=4 1010：采样频率 fSAMPLING=fDTS/16，N=5 0011：采样频率 fSAMPLING=fCK_INT，N=8 1011：采样频率 fSAMPLING=fDTS/16，N=6 0100：采样频率 fSAMPLING=fDTS/2，N=6 1100：采样频率 fSAMPLING=fDTS/16，N=8 0101：采样频率 fSAMPLING=fDTS/2，N=8 1101：采样频率 fSAMPLING=fDTS/32，N=5 0110：采样频率 fSAMPLING=fDTS/4，N=6 1110：采样频率 fSAMPLING=fDTS/32，N=6 0111：采样频率 fSAMPLING=fDTS/4，N=8 1111：采样频率 fSAMPLING=fDTS/32，N=8
3:2	IC1PSC[1:0]	RW	00	输入/捕获 1 预分频器 这 2 位定义了 CC1 输入（IC1）的预分频系数。一旦 CC1E=0(TIM1_CCER 寄存器中)，则预分频器复位。 00：无预分频器，捕获输入口上检测到的每一个边沿都触发一次捕获； 01：每 2 个事件触发一次捕获； 10：每 4 个事件触发一次捕获； 11：每 8 个事件触发一次捕获。
1:0	CC1S[1:0]	RW	00	CC1S[1:0]： 捕获/比较 1 选择。 这 2 位定义通道的方向（输入/输出），及输入脚的选择： 00：CC1 通道被配置为输出； 01：CC1 通道被配置为输入，IC1 映射在 TI1 上；

Bit	Name	R/W	Reset Value	Function
				10：CC1 通道被配置为输入，IC1 映射在 TI2 上； 11：CC1 通道被配置为输入，IC1 映射在 TRC 上。此模式仅工作在内部触发器输入被选中时 （由 TIM1_SMCR 寄存器的 TS 位选择）。 注： CC1S 仅在通道关闭时(TIM1_CCER 寄存器的 CC1E=0)才是可写的。

15.4.8. TIM1 捕获/比较模式寄存器 2(TIM1_CCMR2)

Address offset:0x1C

Reset value:0x0000 0000

Output compare mode:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
OC4CE	OC4M[2:0]			OC4PE	CO4FE	CC4S[1:0]		OC3CE	OC3M[2:0]			OC3PE	OC3FE	CC3S[1:0]	
IC4F[3:0]				IC4PSC[1:0]				IC3F[3:0]				IC3PSC[1:0]			
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31：16	Reserved	-		保留，一直为 0
15	OC4CE	RW	0	输出比较 4 清 0 使能
14:12	OC4M[2:0]	RW	000	输出比较 4 模式
11	OC4PE	RW	0	输出比较 4 预装载使能
10	OC4FE	RW	0	输出比较 4 快速使能
9:8	CC4S[1:0]	RW	00	捕获/比较 4 选择。 该位定义通道的方向（输入/输出），及输入脚的选择： 00：CC4 通道被配置为输出； 01：CC4 通道被配置为输入，IC4 映射在 TI4 上； 10：CC4 通道被配置为输入，IC4 映射在 TI3 上； 11：CC4 通道被配置为输入，IC4 映射在 TRC 上。此模式仅工作在内部触发器输入被选中时（由 TIM1_SMCR 寄存器的 TS 位选择）。 注：CC4S 仅在通道关闭时(TIM1_CCER 寄存器的 CC4E=0)才是可写的。
7	OC3CE	RW	0	输出比较 3 清 0 使能
6:4	OC3M[2:0]	RW	00	输出比较 3 模式
3	OC3PE	RW	0	输出比较 3 预装载使能
2	OC3FE	RW	0	输出比较 3 快速使能
1:0	CC3S[1:0]	RW	00	捕获/比较 3 选择。 这 2 位定义通道的方向（输入/输出），及输入脚的选择： 00：CC3 通道被配置为输出； 01：CC3 通道被配置为输入，IC3 映射在 TI3 上； 10：CC3 通道被配置为输入，IC3 映射在 TI4 上； 11：CC3 通道被配置为输入，IC3 映射在 TRC 上。此模式仅工作在内部触发器输入被选中时 （由 TIM1_SMCR 寄存器的 TS 位选择）。 注：CC3S 仅在通道关闭时(TIM1_CCER 寄存器的 CC3E=0)才是可写的。

Input Capture mode:

Bit	Name	R/W	Reset Value	Function
31：16	Reserved	-		保留，一直为 0
15:12	IC4F	RW	0000	输入捕获 4 滤波器
11:10	IC4PSC	RW	00	输入/捕获 4 预分频器
9:8	CC4S	RW	00	捕获/比较 4 选择。

Bit	Name	R/W	Reset Value	Function
				这 2 位定义通道的方向（输入/输出），及输入脚的选择： 00：CC4 通道被配置为输出； 01：CC4 通道被配置为输入，IC4 映射在 TI4 上； 10：CC4 通道被配置为输入，IC4 映射在 TI3 上； 11：CC4 通道被配置为输入，IC4 映射在 TRC 上。此模式仅工作在内部触发器输入被选中时（由 TIM1_SMCR 寄存器的 TS 位选择）。 注： CC4S 仅在通道关闭时(TIM1_CCER 寄存器的 CC4E=0)才是可写的。
7:4	IC3F	RW	0000	输入捕获 3 滤波器
3:2	IC3PSC	RW	00	输入/捕获 3 预分频器
1:0	OC3S	RW	00	捕获/比较 3 选择。 这 2 位定义通道的方向（输入/输出），及输入脚的选择： 00：CC3 通道被配置为输出； 01：CC3 通道被配置为输入，IC3 映射在 TI3 上； 10：CC3 通道被配置为输入，IC3 映射在 TI4 上； 11：CC3 通道被配置为输入，IC1 映射在 TRC 上。此模式仅工作在内部触发器输入被选中时 （由 TIM1_SMCR 寄存器的 TS 位选择）。 注：CC3S 仅在通道关闭时(TIM1_CCER 寄存器的 CC3E=0)才是可写的。

15.4.9. TIM1 捕获/比较使能寄存器 (TIM1_CCER)

Address offset:0x20

Reset value:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res	Res	CC4P	CC4E	CC3NP	CC3NE	CC3P	CC3E	CC2NP	CC2NE	CC2P	CC2E	CC1NP	CC1NE	CC1P	CC1E
-	-	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31：14	Reserved	-	0	保留，一直为 0
13	CC4P	RW	0	输入/捕获 4 输出极性。参考 CC1P 的描述。
12	CC4E	RW	0	输入/捕获 4 输出使能。参考 CC1E 的描述。
11	CC3NP	RW	0	输入/捕获 3 互补输出极性。参考 CC1NP 的描述。
10	CC3NE	RW	0	输入/捕获 3 互补输出使能。参考 CC1NE 的描述。
9	CC3P	RW	0	输入/捕获 3 输出极性。参考 CC1P 的描述。
8	CC3E	RW	0	输入/捕获 3 输出使能。参考 CC1E 的描述。
7	CC2NP	RW	0	输入/捕获 2 互补输出极性。参考 CC1NP 的描述。
6	CC2NE	RW	0	输入/捕获 2 互补输出使能。参考 CC1NE 的描述。
5	CC2P	RW	0	输入/捕获 2 输出极性。参考 CC1P 的描述。
4	CC2E	RW	0	输入/捕获 2 输出使能。参考 CC1E 的描述。
3	CC1NP	RW	0	输入/捕获 1 互补输出极性 0： OC1N 高电平有效 1： OC1N 低电平有效 注：一旦 LOCK 级别(TIM1_BDTR 寄存器中的 LCCK 位)设为 3 或 2 且 CC1S=00(通道配置为输出)则该位不能被修改。
2	CC1NE	RW	0	输入/捕获 1 互补输出使能 0：关闭－ OC1N 禁止输出，因此 OC1N 的输出电平依赖于 MOE，OSSI，OSSR，OIS1，OIS1N，CC1E 位的值。 1：开启－ OC1N 信号输出到对应的输出引脚，其输出电平依赖于 MOE，OSSI， OSSR，OIS1，OIS1N，CC1E 位的值。
1	CC1P	RW	0	输入/捕获 1 输出极性

Bit	Name	R/W	Reset Value	Function
				CC1 通道配置为输出： 0：OC1 高电平有效 1：OC1 低电平有效 CC1 通道配置为输入： 该位选择是 IC1 还是 IC1 的反相信号作为触发或捕获信号。 0：不反相：捕获发生在 IC1 的上升沿；当用作外部触发器时，IC1 不反相。 1：反相：捕获发生在 IC1 的下降沿；当用作外部触发器时，IC1 反相。 注：一旦 LOCK 级别(TIM1_BDTR 寄存器中的 LCCK 位)设为 3 或 2，则该位不能被修改
0	CC1E	RW	0	输入/捕获 1 输出使能 CC1 通道配置为输出： 0：关闭－ OC1 禁止输出，因此 OC1 的输出电平依赖于 MOE、OSSI、OSSR、OIS1、OIS1N、CC1NE 位的值。 1：开启－ OC1 信号输出到对应的输出引脚，其输出电平依赖于 MOE、OSSI、 OSSR、OIS1、OIS1N、CC1NE 位的值。 CC1 通道配置为输入： 该位决定了计数器的值是否能捕获入 TIM1_CCR1 寄存器。 0：捕获禁止 1：捕获使能

Table 具有中断功能的互补 OCx 和 OCxN 通道的输出控制

Control bits					Output state	
MOE	OSSI	OSSR	CcxE	CcxNE	OCx output state	OCxN output state
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 + Polarity OCxN=OCxREF 异或 CCxNP, OCxN_EN=1
		0	1	0	OCxREF + Polarity OCx=OCREF 异或 CCxP, OCx_EN=1	输出禁止(与定时器断开)，OCxN=0, OCxN_EN=0
		0	1	1	OCREF + Polarity + dead-time OCx_EN=1	OCREF 的互补（not OC-REF）+ Polarity + dead-time 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	输出禁止(与定时器断开), OCx=CCxP, OCx_EN=1	OCxREF+Polarity OCxN=OCxREF xor CCxNP, OCxN_EN=1
		1	1	0	OCxREF+Polarity 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	OCREF+Polarity + dead-time OCx_EN=1	OCREF 的互补（not OC-REF）+ polarity + dead-time OCN_EN=1
0	0	X	0	0	输出禁止(与定时器断开), OCx=CCxP, OCx_EN=0	输出禁止(与定时器断开), OCxN=CCxNP, OCxN_EN=0
	0		0	1	输出禁止(与定时器断开) 异步的: OCx=CCxP, OCx_EN=0, OCxN=CCxNP, OCxN_EN=0 如果时钟存在：经过一个死区时间后，假设 OISx 与 OISxN 并不都对应 OCx 和 OCxN 的有效电平，OCx=OISx 和 OCxN=OISxN。	
	0		1	0		
	0		1	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） 异步的：OCx=CCxP, OCx_EN=1，OCxN=CCxNP, OCxN_EN=1 若时钟存在：经过一个死区时间后，假设 OISx 与 OISxN 并不都对应 OCx 和 OCxN 的有效电平，OCx=OISx 和 OCxN=OISxN	
	1		1	0		
	1		1	1		

15.4.10. TIM1 计算器(TIM1_CNT)

Address offset:0x24

Reset value:0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31：16	Reserved			保留，一直为 0
15:0	CNT[15:0]	RW	0	计数器的值

15.4.11. TIM1 预分频器 (TIM1_PSC)

Address offset:0x28

Reset value:0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31：16	Reserved			保留，一直为 0
15:0	PSC[15:0]	RW	0	预分频器的值 计数器的时钟频率（CK_CNT）等于 fCK_PSC/(PSC[15:0]+1)。 PSC 包含了当更新事件产生时装入当前预分频器寄存器的值；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。

15.4.12. TIM1 自动重新加载寄存器 (TIM1_ARR)

Address offset:0x2c

Reset value:0x0000 FFFF

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

Bit	Name	R/W	Reset Value	Function
31：16	Reserved			保留，一直为 0
15:0	ARR[15:0]	RW	0	自动重装载的值 ARR 包含了将要装载入实际的自动重装载寄存器的值。 当自动重装载的值为空时，计数器不工作。

15.4.13. TIM1 重复计数器寄存器(TIM1_RCR)

Address offset:0x30

Reset value:0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31：8	Reserved			保留，一直为 0
7:0	REP[7:0]	RW	0	周期计数器的值 开启了预装载功能后，这些位允许用户设置比较寄存器的更新速率（即周期性地从预装载寄 存器传输到当前寄存器）；如允许产生更新中断，则会同时影响产生更新中断的速率。 每次向下计数器 REP_CNT 达到 0，会产生一个更新事件并且计数器 REP_CNT 重新从 REP 值开始计数。由于 REP_CNT 只有在周期更新事件 U_RC 发生时才重载 REP 值，因此对 TIM1_RCR 寄存器写入的新值只在下次周期更新事件发生时才起作用。 这意味着在 PWM 模式中，(REP+1)对应着： — 在边沿对齐模式下，PWM 周期的数目； — 在中心对称模式下，PWM 半周期的数目；

15.4.14. TIM1 捕获/比较寄存器 1(TIM1_CCR1)

Address offset:0x34

Reset value:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CCR1[15:0]															
RW															

Bit	Name	R/W	Reset Value	Function
31：16	Reserved			保留，一直为 0
15：0	CCR1[15:0]	RW	0	捕获/比较 1 的值 若 CC1 通道配置为输出： CCR1 包含了装入当前捕获/比较 1 寄存器的值（预装载值）。 如果在 TIM1_CCMR1 寄存器(OC1PE 位)中未选择预装载特性，其始终装入当前寄存器中。 否则，只有当更新事件发生时，此预装载值才装入当前捕获/比较 1 寄存器中。 当前捕获/比较寄存器包含了与计数器 TIM1_CNT 比较的值，并且在 OC1 端口上输出信号。 若 CC1 通道配置为输入： CCR1 包含了由上一次输入捕获 1 事件（IC1）传输的计数器值。

15.4.15. TIM1 捕捉/比较寄存器 2(TIM1_CCR2)

Address offset:0x38

Reset value:0x0000 0000

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

Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CCR2[15:0]															
RW															

Bit	Name	R/W	Reset Value	Function
31：16	Reserved			保留，一直为 0
15:0	CCR2[15:0]	RW	0	捕获/比较 2 的值 若 CC2 通道配置为输出： CCR2 包含了装入当前捕获/比较 2 寄存器的值（预装载值）。 如果在 TIM1_CCMR2 寄存器(OC2PE 位)中未选择预装载特性，其始终装入当前寄存器中。 否则，只有当更新事件发生时，此预装载值才装入当前捕获/比较 2 寄存器中。 当前捕获/比较寄存器包含了与计数器 TIM1_CNT 比较的值，并且在 OC 端口上输出信号。 若 CC2 通道配置为输入： CCR2 包含了由上一次输入捕获 2 事件（IC2）传输的计数器值。

15.4.16. TIM1 捕获/比较寄存器 3 (TIM1_CCR3)

Address offset:0x3C

Reset value:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CCR3[15:0]															
RW															

Bit	Name	R/W	Reset Value	Function
31：16	Reserved			保留，一直为 0
15:0	CCR3[15:0]	RW	0	捕获/比较 3 的值 若 CC3 通道配置为输出： CCR3 包含了装入当前捕获/比较 3 寄存器的值（预装载值）。 如果在 TIM1_CCMR3 寄存器(OC3PE 位)中未选择预装载特性，其始终装入当前寄存器中。 否则，只有当更新事件发生时，此预装载值才装入当前捕获/比较 3 寄存器中。 当前捕获/比较寄存器包含了与计数器 TIM1_CNT 比较的值，并且在 OC 端口上输出信号。 若 CC3 通道配置为输入： CCR3 包含了由上一次输入捕获 3 事件（IC3）传输的计数器值。

15.4.17. TIM1 捕捉/比较寄存器 4(TIM1_CCR4)

Address offset:0x40

Reset value:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CCR4[15:0]															
RW															

Bit	Name	R/W	Reset Value	Function
31：16	Reserved			保留，一直为 0
15:0	CCR4[15:0]	RW	0	捕获/比较 4 的值 若 CC4 通道配置为输出： CCR4 包含了装入当前捕获/比较 4 寄存器的值（预装载值）。 如果在 TIM1_CCMR4 寄存器(OC4PE 位)中未选择预装载特性，其始终装入当前寄存器中。 否则，只有当更新事件发生时，此预装载值才装入当前捕获/比较 4 寄存器中。 当前捕获/比较寄存器包含了与计数器 TIM1_CNT 比较的值，并且在 OC 端口上输出信号。 若 CC4 通道配置为输入： CCR4 包含了由上一次输入捕获 4 事件（IC4）传输的计数器值。

15.4.18. TIM1 刹车和死区寄存器(TIM1_BDTR)

Address offset:0x44

Reset value:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
MOE	**AOE**	**BKP**	**BKE**	**OSSR**	**OSSI**	**LOCK[1:0]**		**DTG[7:0]**							
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31：16	Reserved	RW	0	保留，一直为 0
15	MOE	RW	0	主输出使能 一旦刹车输入有效，该位被硬件异步清 0。根据 AOE 位的值，可由软件清 0 或自动置 1。它仅对配置为输出通道有效。 0：禁止 OC 和 OCN 输出或强制为空闲状态； 1：如果设置了相应的使能位（TIM1_CCER 寄存器的 CcxE、 CcxNE 位），则开启 OC 和 OCN 输出。
14	AOE	RW	0	自动输出使能 0： MOE 只能被软件置 1； 1： MOE 能被软件置 1 或在下一个更新事件自动置 1（如果刹车输入无效）。 注：一旦 LOCK 级别(TIM1_BDTR 寄存器中的 LOCK 位)设为 1，则该位不能被修改。
13	BKP	RW	0	刹车输入极性 0：刹车输入低电平有效； 1：刹车输入高电平有效。 注：一旦 LOCK 级别(TIM1_BDTR 寄存器中的 LOCK 位)设为 1，则该位不能被修改。
12	BKE	RW	0	刹车功能使能 0：禁止刹车输入（BRK 及 BRK_ACTH）； 1：开启刹车输入（BRK 及 BRK_ACTH）。 注：一旦 LOCK 级别(TIM1_BDTR 寄存器中的 LOCK 位)设为 1，则该位不能被修改。
11	OSSR	RW	0	运行模式下“关闭状态”选择 该位用于当 MOE=1 且通道为互补输出时。没有互补输出的定时器中不存在 OSSR 位。 参考 OC/OCN 使能的详细说明（捕获/比较使能寄存器(TIM1_CCER)）。 0：当定时器不工作时，禁止 OC/OCN 输出（OC/OCN 使能输出信号=0）；

Bit	Name	R/W	Reset Value	Function
				1：当定时器不工作时，一旦 CcxE=1 或 CcxNE=1，开启 OC/OCN 输出并输出无效电平。 OC/OCN 使能输出信号=1。 注：一旦 LOCK 级别(TIM1_BDTR 寄存器中的 LOCK 位)设为 2，则该位不能被修改。
10	OSSI	RW	0	空闲模式下“关闭状态”选择 该位用于当 MOE=0 且通道设为输出时。 参考 OC/OCN 使能的详细说明（捕获/比较使能寄存器(TIM1_CCER)）。 0：当定时器不工作时，禁止 OC/OCN 输出（OC/OCN 使能输出信号=0）； 1：当定时器不工作时，一旦 CcxE=1 或 CcxNE=1，OC/OCN 首先输出其空闲电平。 OC/OCN 使能输出信号=1。 注：一旦 LOCK 级别(TIM1_BDTR 寄存器中的 LOCK 位)设为 2，则该位不能被修改。
9:8	LOCK[1:0]	RW	00	锁定设置 该位为防止软件错误而提供写保护。 00：锁定关闭，寄存器无写保护； 01：锁定级别 1，不能写入 TIM1_BDTR 寄存器的 DTG/BKE/BKP/AOE 位、 TIM1_CR2 寄存器的 OISx/OISxN 位； 10：锁定级别 2，不能写入锁定级别 1 中的各位，也不能写入 CC 极性位（一旦相关通道通过 CCxS 位设为输出，TIM1_CCER 寄存器的 CCxP/CCNxP 位）以及 OSSR/OSSI 位； 11：锁定级别 3，不能写入锁定级别 2 中的各位，也不能写入 CC 控制位（一旦相关通道通过 CCxS 位设为输出，TIM1_CCMRx 寄存器的 OCxM/OCxPE 位）； 注：在系统复位后，只能写一次 LOCK 位，一旦写入 TIM1_BDTR 寄存器，则其内容冻结直至复位。
7:0	DTG[7:0]	RW	0000 0000	死区发生器设置 这些位定义了插入互补输出之间的死区持续时间。假设 DT 表示其持续时间： DTG[7:5]=0xx => DT=DTG[7:0] × Tdtg， Tdtg = TDTS； DTG[7:5]=10x => DT=(64+DTG[5:0]) × Tdtg， Tdtg = 2 × TDTS； DTG[7:5]=110 => DT=(32+DTG[4:0]) × Tdtg， Tdtg = 8 × TDTS； DTG[7:5]=111 => DT=(32+DTG[4:0]) × Tdtg， Tdtg = 16 × TDTS； 例：若 TDTS = 125ns(8MHZ)，可能的死区时间为： 0 到 15875ns，若步长时间为 125ns； 16us 到 31750ns，若步长时间为 250ns； 32us 到 63us，若步长时间为 1us； 64us 到 126us，若步长时间为 2us； 注：一旦 LOCK 级别(TIM1_BDTR 寄存器中的 LOCK 位)设为 1、 2 或 3，则这些位不能被修改。

15.4.19. TIM1 寄存器映像

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	TIM1_CR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CKD[1:0]		ARPE	CMS[1:0]		DIR	OPM	URS	UDIS	CEN

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reset value																							0	0	0	0	0	0	0	0	0	0
0x04	TIM1_CR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OIS4	OIS3N	OIS3	OIS2N	OIS2	OIS1N	OIS1	TI1S	MMS [2:0]			Res.	CCUS	Res.	CCPC
	Reset value																		0	0	0	0	0	0	0	0	0	0	0		0		0
0x08	TIM1_SMCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ETP	ECE	ETPS[1:0]		ETF[3:0]				MSM	TS[2:0]			OCCS	SMS [2:0]		
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	TIM1_DIER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BIE	TIE	COMIE	CC4IE	CC3IE	CC2IE	CC1IE	UIE
	Reset value																									0	0	0	0	0	0	0	0
0x10	TIM1_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC4O	CC3O	CC2O	CC1O	Res.	BIF	TIF	COMIF	CC4IF	CC3IF	CC2IF	CC1IF	UIF
	Reset value																				0	0	0	0		0	0	0	0	0	0	0	0
0x14	TIM1_EGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BG	TG	COMG	CC4G	CC3G	CC2G	CC1G	UG
	Reset value																									0	0	0	0	0	0	0	0
0x18	TIM1_CCMR1(output compare mode)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC2CE	OC2M [2:0]			OC2PE	CO2FE	CC2S [1:0]		OC1CE	OC1M [2:0]			OC1PE	OC1FE	CC1S [1:0]	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	TIM1_CCMR1(Input Capture mode)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC2F[3:0]				IC2PSC [1:0]		CC2S [1:0]		IC1F[3:0]				IC1PSC [1:0]		CC1S [1:0]	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1C	TIM1_CCMR2(output capture mode)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC4CE	OC4M [2:0]			OC4PE	CO4FE	CC4S [1:0]		OC3CE	OC3M [2:0]			OC3PE	OC3FE	CC3S [1:0]	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1C	TIM1_CCMR2(Input	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC4F[3:0]				IC4PSC [1:0]		CC4S [1:0]		IC3F[3:0]				IC3PSC [1:0]		CC3S [1:0]	

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Capture mode)																																
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20	TIM1_CCER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC4P	CC4E	CC3NP	CC3NE	CC3P	CC3E	CC2NP	CC2NE	CC2P	CC2E	CC1NP	CC1NE	CC1P	CC1E
	Reset value																			0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x24	TIM1_CNT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CNT[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x28	TIM1_PSC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PSC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x2C	TIM1_ARR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARR[15:0]															
	Reset value																	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x30	TIM1_RCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REP[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x34	TIM1_CCR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR1[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x38	TIM1_CCR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR2[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x3C	TIM1_CCR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR3[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x40	TIM1_CCR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR4[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x44	TIM1_BDTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MOE	AOE	BKP	BKE	OSSR	OSSI	LOCK[1:0]		DTG[7:0]							
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

16. 基本定时器 (TIM16)

16.1. TIM16 主要特性

- 16 位自动装载向上计数器
- 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
- 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
- 如下时间发生时产生中断：
  - 更新：计数器上溢

16.2. TIM16 功能描述

16.2.1. 时基单元

这个可编程定时器的主要部分是一个带有自动重装载的 16 位的向上计数器，计数器的时钟通过一个预分频器得到。

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

时基单元包括：

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

自动装载寄存器是预先装载的，写或者读自动重装载寄存器（TIMx_ARR）将访问预装载寄存器（preload register）。根据 TIMx_CR1 寄存器中的自动装载使能位（ARPE）的设置，预装载寄存器的内容可以被一直或者在每次的更新事件 UEV 时传送到影子寄存器。当计数器达到上溢出并当 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。

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。

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

预分频描述：

预分频器可以将计数器的时钟按 1 到 65535 之间的任意值分频。它是基于一个（在 TIMx_PSC 寄存器中的）16 位寄存器控制的 16 位计数器。因为这个控制寄存器带有缓冲器，它能够在运行时被改变。新的预分频器的参数在下一次更新事件到来时被采用。

下图给出了在预分频器运行时，更改计数器参数的例子。

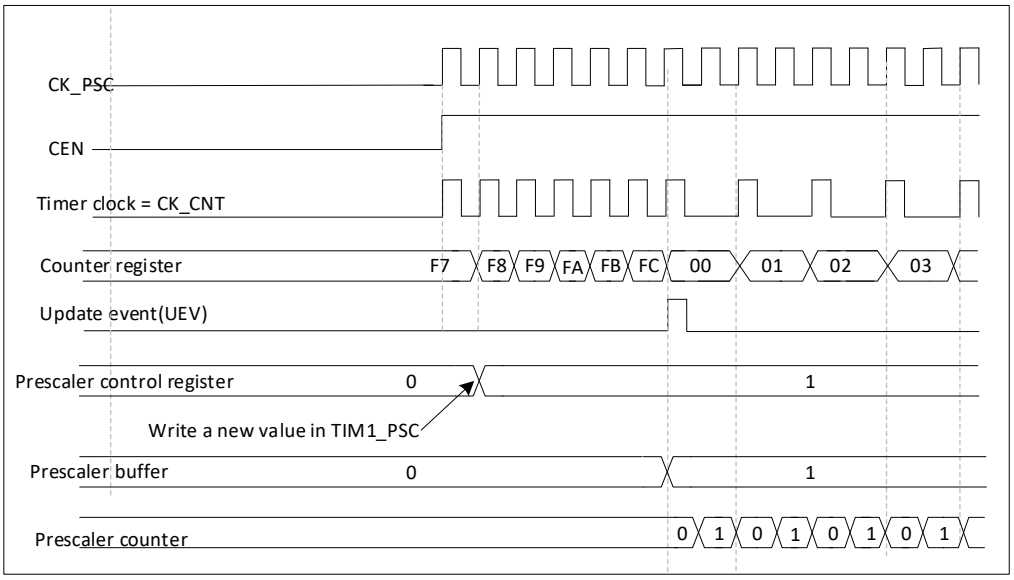


图 16-1 当预分频器的参数从 1 变到 2 时，计数器的时序图

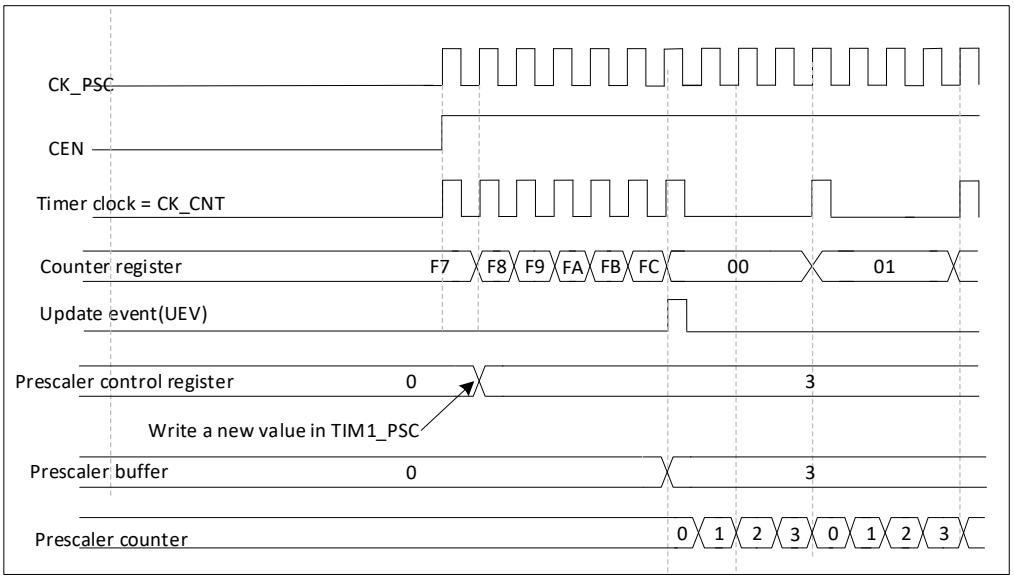


图 16-2 当预分频器的参数从 1 变到 4 时，计数器的时序图

16.2.2. 计数器模式

计数器从 0 计数到自动装载值（TIMx_ARR 寄存的值），然后又从 0 重新开始计数，并产生一个计数器溢出事件。

如果重复计数器被使用，则在向上计数器重复几次（对重复计数器可编程）后，产生更新事件。否则，在每个计数溢出时，产生更新事件。

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

设置 TIMx_CR1 寄存器中的 UDIS 位，可以禁止更新事件；这样也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 位被清零之前，将不产生更新事件。虽然如此，但是计数器依旧从 0 开始，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 TIMx_RCR 寄存器中的值重新装载。

- 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)。

下图显示了几个在不同频率下的计数器行为，当 TIMx_ARR=0X36。

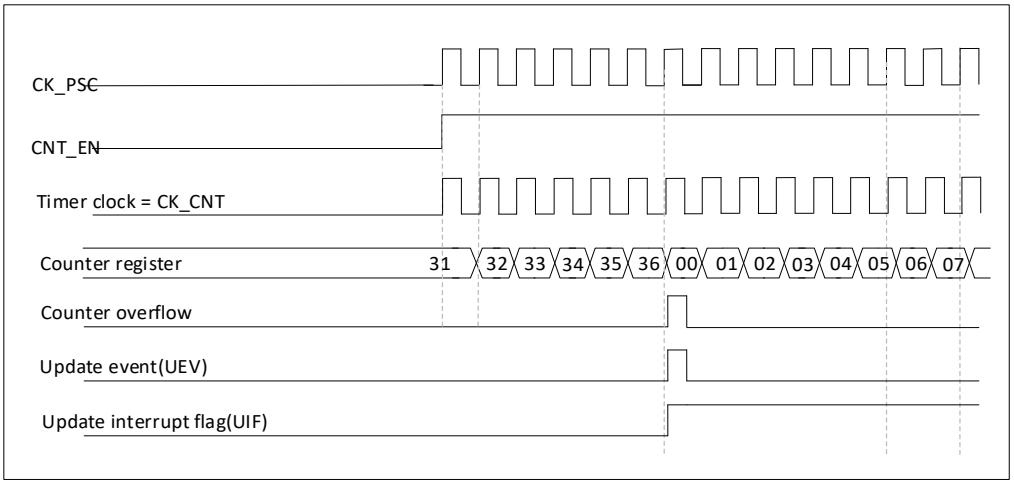


图 16-3 计数器时序图，内部时钟分频因子为 1

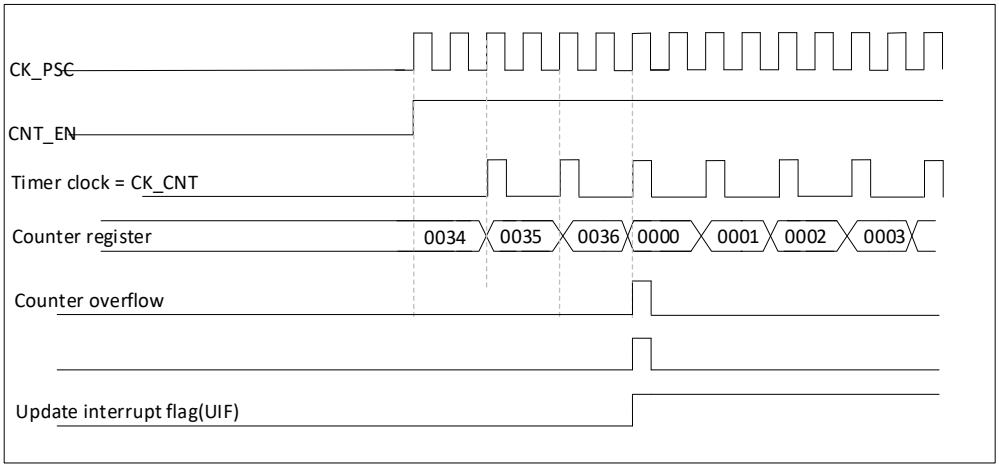


图 16-4 计数器时序图，内部时钟分频因子为 2

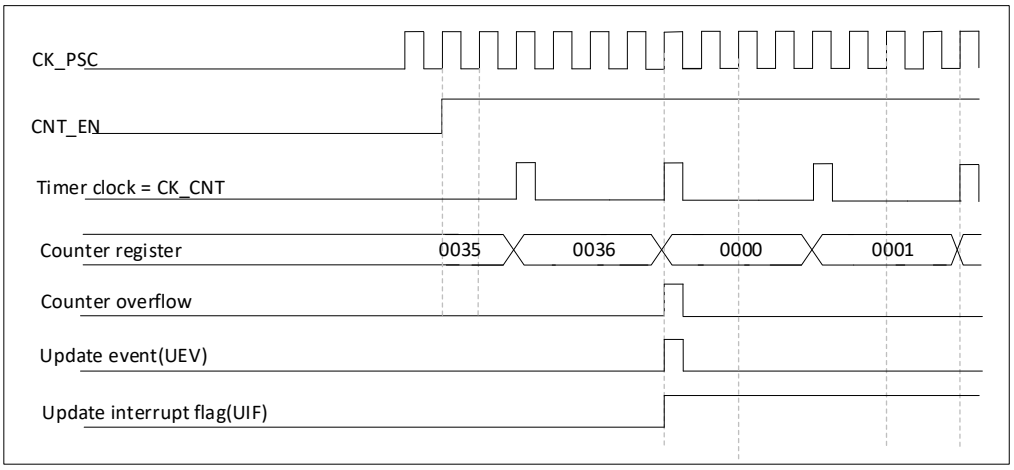


图 16-5 计数器时序图，内部时钟分频因子为 4

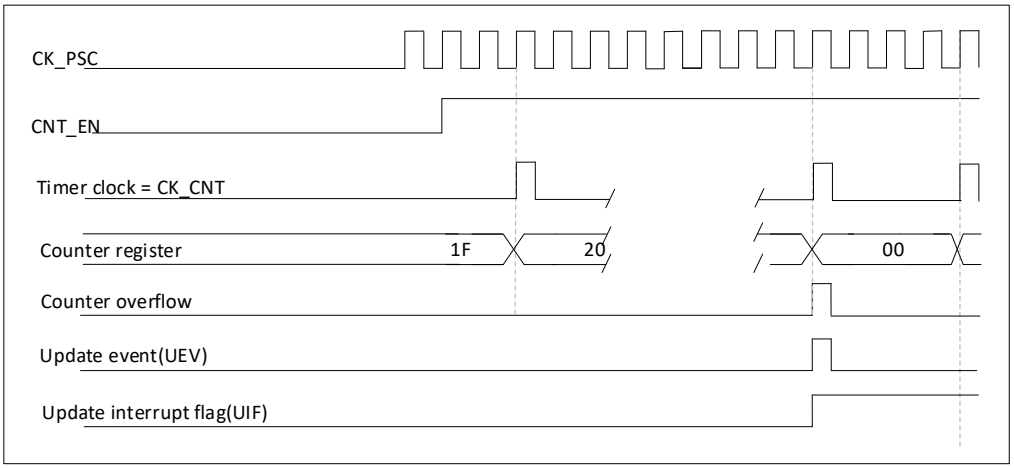


图 16-6 计数器时序图，内部时钟分频因子为 N

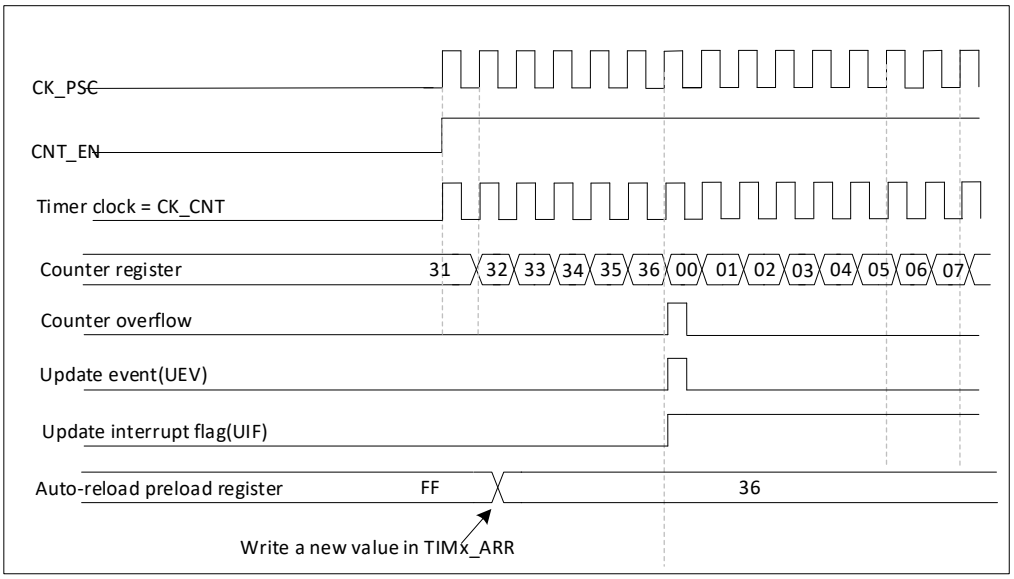


图 16-7 计数器时序图，当 ARPE=0 时的更新事件(TIMx_ARR 没有预装入)

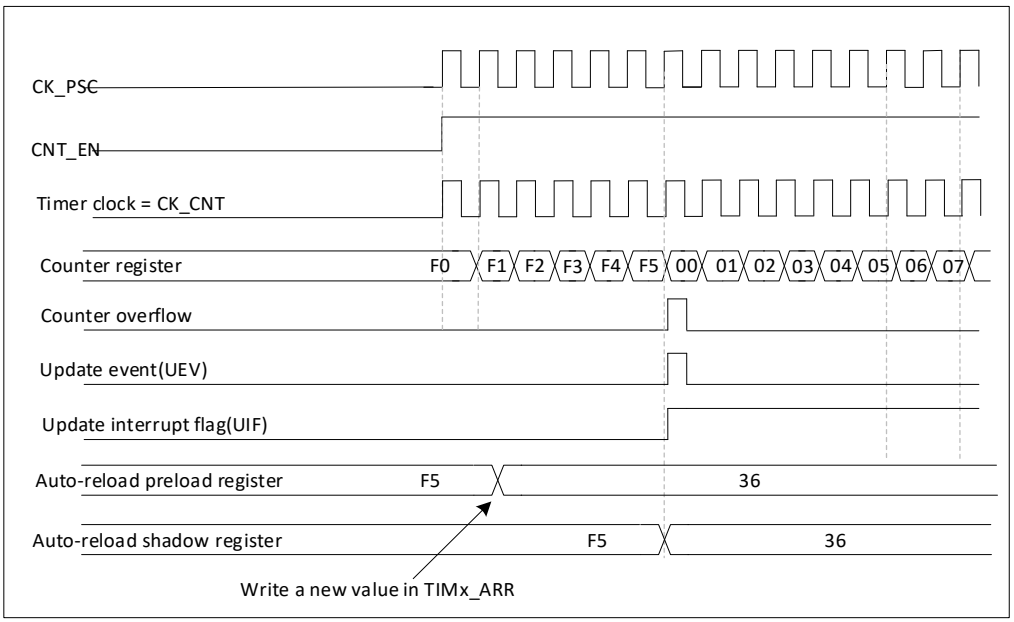


图 16-8 计数器时序图，当 ARPE=1 时的更新事件(预装入了 TIMx_ARR)

16.2.3. 重复计数器

时基单元描述了计数器上溢时更新事件（UEV）是如何产生的，然而实际上它只能在重复计数器达到 0 的时候产生。这个特性对产生 PWM 信号有用。

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

重复计数器，在向上计数器模式的每个计数上溢时递减。

重复计数器是自动重载的，重复速率是由 TIMx_RCR 寄存器的值定义。当更新事件由软件（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到重复计数器中。

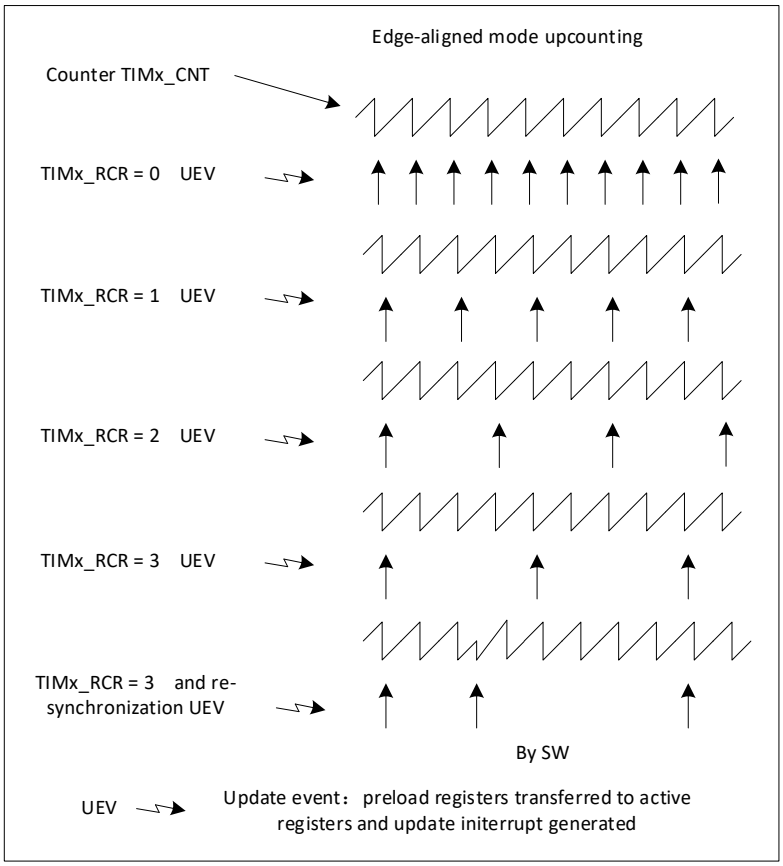


图 16-9 不同模式下更新速率的例子，及 TIMx_RCR 的寄存器设置

16.2.4. 时钟源

计数器的时钟由内部时钟（CK_INT）提供。TIMx_CR1 寄存器的 CEN 位和 TIMx_EGR 寄存器的 UG 位是实际的控制位（除了 UG 位被自动清除外），只能通过软件改变他们。一旦置 CEN 位为 1，内部时钟即向分频器提供时钟。

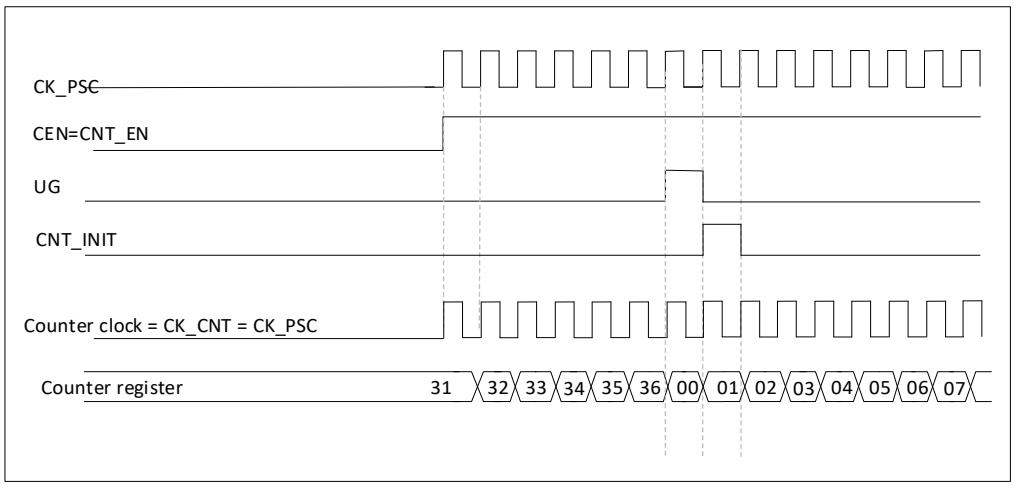


图 16-10 一般模式下的控制电路，内部时钟分频因子为 1

16.3. TIM16 寄存器

16.3.1. TIM16 控制寄存器 1 (TIMx_CR1)

Address offset:0x00

Reset value:0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31：10	Reserved	-	0	保留，一直读为 0
9:8	CKD[1:0]	RW	00	时钟分频因子 这 2 位定义在定时器时钟(CK_INT)频率，死区时间和由死区发生器与数字滤波器(ETR,Tix)所用的采样时钟之间的分频比例 00： tDTS = tCK_INT 01： tDTS = 2 x tCK_INT 10： tDTS = 4 x tCK_INT 11：保留，不要使用这个配置
7	ARPE	RW	0	自动重装载预装载允许位 0： TIM1_ARR 寄存器没有缓冲 1： TIM1_ARR 寄存器被装入缓冲器
6:3	Reserved	-	0	保留，一直读为 0
2	URS	RW	0	更新请求源 软件通过该位选择 UEV 事件的源 0：如果允许产生更新中断请求，则下述任一事件产生一个更新中断 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请求，则只有计数器溢出/下溢产生一个更新中断请求
1	UDIS	RW	0	禁止更新 软件通过该位允许/禁止 UEV 事件的产生

Bit	Name	R/W	Reset Value	Function
				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	0	允许计数器 0：禁止计数器 1：开启计数器 注：在软件设置了 CEN 位后，外部时钟、门控模式和编码器模式才能工作。触发模式可以自动地通过硬件设置 CEN 位。

16.3.2. TIM16 中断使能寄存器(TIM16_DIER)

Address offset:0x0C

Reset value:0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31：1	Reserved	-	0	保留，一直为 0
0	UIE	RW	0	UIE：允许更新中断 0：禁止更新中断 1：允许更新中断

16.3.3. TIM16 状态寄存器 (TIM16_SR)

Address offset:0x010

Reset value:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	UIF
															RC_W0

Bit	Name	R/W	Reset Value	Function
31：1	Reserved	-	0	保留，一直为 0
0	UIF	Rc_w0	0	更新中断标记 当产生更新事件时该位由硬件置 1。它由软件清 0。 0：无更新事件产生； 1：更新事件等待响应。当寄存器被更新时该位由硬件置 1： – 若 TIMx_CR1 寄存器的 UDIS=0，产生更新事件(计数器上溢)； – 若 TIMx_CR1 寄存器的 UDIS=0、 URS=0，当 TIMx_EGR 寄存器的 UG=1 时产生更新事件(软件对 CNT 重新初始化)；

16.3.4. TIM16 事件产生寄存器(TIM16_EGR)

Address offset:0x14

Reset value:0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31：1	Reserved	-	0	保留，一直为 0
0	UG	W	0	产生更新事件 该位由软件置 1，由硬件自动清 0。 0：无动作； 1：重新初始化计数器，并产生一个更新事件。注意预分频器的计数器也被清 0(但是预分频 系数不变)。若在中心对称模式下或 DIR=0(向上计数)则计数器被清 0，若 DIR=1(向下计数)则计数器取 TIMx_ARR 的值。

16.3.5. TIM16 计数器(TIM16_CNT)

Address offset:0x24

Reset value:0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31：16	Reserved			保留，一直为 0
15:0	CNT[15:0]	RW	0	计数器的值

16.3.6. TIM16 预分频器(TIM16_PSC)

Address offset:0x28

Reset value:0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31：16	Reserved			保留，一直为 0
15:0	PSC[15:0]	RW	0	预分频器的值 计数器的时钟频率（CK_CNT）等于 fCK_PSC/(PSC[15:0]+1)。 PSC 包含了当更新事件产生时装入当前预分频器寄存器的值；更新事件包括计数器被 TIMx_EGR 的 UG 位清 0 或被工作在复位模式的从控制器清 0。

16.3.7. TIM16 自动重装载寄存器 (TIM16_ARR)

Address offset:0x2c

Reset value:0x0000 FFFF

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

Bit	Name	R/W	Reset Value	Function
31：16	Reserved			保留，一直为 0
15:0	ARR[15:0]	RW	0xFFFF	自动重装载的值 ARR 包含了将要装载入实际的自动重装载寄存器的值。 当自动重装载的值为空时，计数器不工作。

16.3.8. TIM16 周期计数寄存器(TIM16_RCR)

Address offset:0x30

Reset value:0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31：8	Reserved			保留，一直为 0
7:0	REP[7:0]	RW	0	周期计数器的值 开启了预装载功能后，这些位允许用户设置比较寄存器的更新速率（即周期性地从预装载寄 存器传输到当前寄存器）；如允许产生更新中断，则会同时影响产生更新中断的速率。 每次向下计数器 REP_CNT 达到 0，会产生一个更新事件并且计数器 REP_CNT 重新从 REP 值开始计数。由于 REP_CNT 只有在周期更新事件 U_RC 发生时才重载 REP 值，因此对 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。

16.3.9. TIM16 寄存器映像

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	TIMx_CR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CKD[1:0]		ARPE	Res.	Res.	Res.	OPM	URS	UDIS	CEN
	Reset value																							0	0	0				0	0	0	0
0x00C	TIM16_DIER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UIE
	Reset value																																0
0x	TIM16_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UIF

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
10	Reset value																																0
0x14	TIM16_EGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UG
	Reset value																																0
0x24	TIM16_CNT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CNT[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x28	TIM16_PSC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PSC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x2C	TIM16_ARR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARR[15:0]															
	Reset value																	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x30	TIM16_RCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REP[7:0]							
	Reset value																									0	0	0	0	0	0	0	0

17. 低功耗定时器(LPTIM)

17.1. 简介

LPTIM 是一款 16 位定时器。LPTIM 将系统从低功耗模式中唤醒的能力使得它适合于实现低功耗应用。LPTIM 引入了一种灵活的时钟方案，可提供所需的功能和性能，同时将功耗降至最低。

17.2. LPTIM 主要特性

- 16 位向上计数器
- 3 位预分频器，具有 8 个可能的分频因子（1、2、4、8、16、32、64、128）
- 可选时钟
  - 内部时钟源: LSI 或 APB 时钟
- 16 Bit ARR 可重载寄存器
- 单次模式

17.3. 低功耗定时器（LPTIM）功能描述

17.3.1. LPTIM 框图

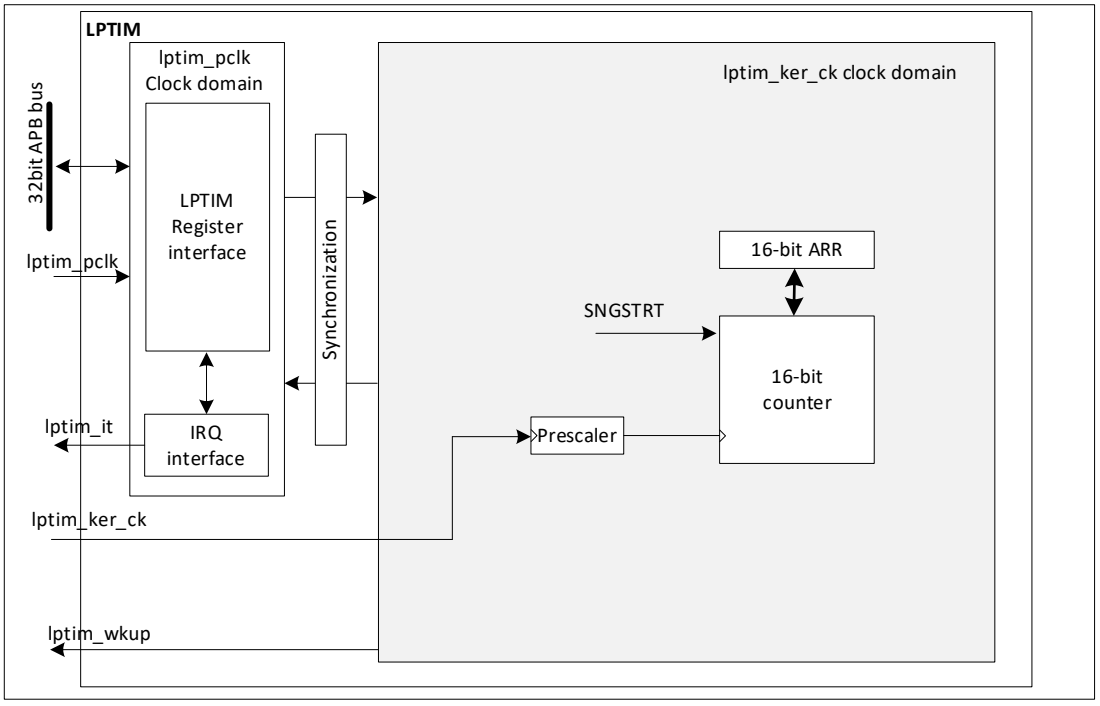


图 17-1 低功耗定时器框图

17.3.2. LPTIM 管脚和内部信号

表 17-1 LPTIM 内部信号

Names	Signal type	Description
lptim_pclk	Digital input	LPTIM APB clock domain
lptim_ker_ck	Digital input	LPTIM kernel clock
lptim_it	Digital output	LPTIM global interrupt

Names	Signal type	Description
lptim_wakeup	Digital output	LPTIM wakeup event

17.3.3. LPTIM 复位和时钟

LPTIM 可以使用多个时钟源进行计时。

通过 RCC 模块，可以使用内部时钟信号对其进行时钟控制（该时钟信号可以在 APB、LSI 源中进行选择）。

17.3.4. 预分频器

LPTIM 16 位计数器，由一个可配置的 2 次方预分频器控制驱动。预分频器分频比由 PRESC[2:0]控制。

下表列出了所有情况：

表 17-2 预分频系数

Programming	Dividing factor
000	/1
001	/2
010	/4
011	/8
100	/16
101	/32
110	/64
111	/128

注：当 lptim_ker_ck 选择 PCLK（通过 RCC_CCIPR.LPTIMSEL 配置）时，预分频系数要配置 2 分频及以上。

17.3.5. 工作模式

LPTIM 只有一种使用 timer 模式。

- **一次模式：**定时器从一个触发事件开始，当达到 ARR 值时停止。

  要使能单次计数，SNGSTRT 位必须置 1。

  一个新的触发事件将重新启动计时器。在计数器启动之后，并到达 ARR 之前的任何触发事件都将被忽略。

17.3.6. 寄存器更新

PRELOAD 位控制 LPTIM_ARR 寄存器的更新方式：

- 当 PRELOAD 位被复位为“0”：LPTIM_ARR 寄存器在任何写访问后立即更新。
- 当 PRELOAD 位被设为“1”时：如果定时器已经启动，则 LPTIM_ARR 将在当前周期结束时更新。

LPTIM APB 接口和 LPTIM Kernel 逻辑使用不同的时钟，因此在 APB 写入,和写入的值被应用到计数器比较器时，存在一定的延迟。在此延迟周期内，必须避免对这些寄存器进行任何额外的写操作。

17.3.7. 使能计时器

LPTIM_CR 寄存器中的 ENABLE 位用于使能/不使能 LPTIM 内核逻辑。置位 ENABLE 位后，需要延迟两个计数器时钟才能使能 LPTIM。

仅当 LPTIM 禁用时，才能修改 LPTIM_CFGR 和 LPTIM_IER 寄存器。

17.3.8. 计数器复位

为了将 LPTIM_CNT 寄存器的内容复位，提供复位机制：

异步复位机制：

异步复位由 LPTIM_CR 寄存器的 RSTARE 位控制。当该位被置为 1 时，任何读 LPTIM_CNT 寄存器的访问都将其内容复位为零。

应注意，为了可靠地读取 LPTIM_CNT 寄存器，必须进行 2 次读访问并比较其结果，结果一致，被则认为是读出值是可靠的。

需要注意的是：

- 使能异步复位时，第一次读会复位 LPTIM_CNT；第二次读才能读取 LPTIM_CNT 寄存器的计数结果。
- 在 LPTIM 计数时钟选择 PCLK/HSI 时，连续访问 2 次也不能保证读出值可靠。

17.3.9. 调试模式（debug mode）

当芯片进入 debug 模式，取决于 DBG 模块的 DBG_LPTIM_STOP 位的设定，LPTIM 或者继续正常工作，或者停止工作。

17.4. LPTIM 低功耗模式

表 17-3 LPTIM 不同低功耗模式的区别

模式	描述
Sleep	No effect. LPTIM interrupts cause the device to exit Sleep mode.
Stop	No effect when LPTIM is clocked by LSI. LPTIM interrupts cause the device to exit Stop.

17.5. LPTIM 中断

如果下列事件在 LPTIM_IER 寄存器内使能，则这些事件将生成中断/唤醒事件：

- 自动重新加载匹配

注意:如果在 LPTIM_ISR 寄存器（状态寄存器）中的相应标志置 1 后，LPTIM_IER 寄存器（中断使能寄存器）中的相应位被置 1，则不产生中断。

中断事件	描述
自动重载匹配	当计数器寄存器的内容(LPTIM_CNT)与自动重新加载寄存器的内容匹配(LPTIM_ARR)，中断标志置位

17.6. LPTIM 寄存器

17.6.1. LPTIM 中断和状态寄存器 (LPTIM_ISR)

Address offset：0x000

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31：2	Reserved	-	0	
1	ARRM	R	0	自动重载匹配 ARRM 由硬件设置，通知应用程序 LPTIM_CNT 寄存器值匹配 LPTIM_ARR 寄存器的值。向 LPTIM_ICR 寄存器的 ARRMCF 位写入 1 可清除 ARRM 标志
0	Reserved			

17.6.2. LPTIM 中断清除寄存器 (LPTIM_ICR)

Address offset：0x004

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31：2	Reserved	-	0	
1	ARRMCF	RW	0	自动重载匹配清除标志 向该位写入 1 可清除 LPTIM_ISR 寄存器中的 ARRM 标志
0	Reserved			

17.6.3. LPTIM 中断使能寄存器 (LPTIM_IER)

Address offset：0x008

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31：2	Reserved	-	0	
1	ARRMIE	RW	0	自动重载匹配中断使能 0:ARRM 中断禁用 1:ARRM 中断使能
0	Reserved			

17.6.4. LPTIM 配置寄存器 (LPTIM_CFGR)

Address offset：0x00C

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31:23	Reserved	-	0	
22	PRELOAD	RW	0	寄存器更新模式 预加载位控制 LPTIM_ARR 寄存器更新模式 0:每次 APB 总线写访问后更新寄存器 1:寄存器在当前 LPTIM 周期结束时更新
21:12	Reserved			

Bit	Name	R/W	Reset Value	Function
11:9	PRESC[2:0]	RW	0	时钟预分频器 PRESC 位配置预分频器分频系数。它可以是下列分部中的一个因素: 000:/1 001:/2 010:/4 011:/8 100:/16 101:/32 110:/64 111:/128
8:0	Reserved	-	0	保留，一直为 0

17.6.5. LPTIM 控制寄存器 (LPTIM_CR)

Address offset：0x010

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31:5	Reserved	-	0	
4	RSTARE	RW	0	读取后复位使能 此位由软件置 1 和清 0。当 RSTARE 设置为“1”时，对 LPTIM_CNT 的任何读取访问寄存器将异步重置 LPTIM_CNT 寄存器内容。
3:2	Reserved	-	0	
1	SNGSTRT	RW	0	LPTIM 启动单次模式。 该位由软件置位，由硬件清零。该位置 1 将以单脉冲模式启动 LPTIM。 注：仅当 LPTIM 使能时，此位才能置 1。它将由硬件自动复位。
0	ENABLE	RW	0	LPTIM 使能位,由软件设置和清零 0:LPTIM 禁用 1:LPTIM 使能

17.6.6. LPTIM 自动重装载寄存器 (LPTIM_ARR)

Address offset：0x018

Reset value: 0x0000 0001

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

Bit	Name	R/W	Reset Value	Function
31:16	Reserved	-	0	保留，一直为 0
15：0	ARR	RW	0x0001	自动重新加载值 ARR 是 LPTIM 的自动重载值 当 LPTIM 使能后才能更新该寄存器

17.6.7. LPTIM 计数寄存器 (LPTIM_CNT)

Address offset：0x01C

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31:16	Reserved	-	0	
15：0	CNT	R	0	计数器值 当 LPTIM 以异步时钟运行时，读取 LPTIM_CNT 寄存器可能返回不可靠的值。因此在这种情况下，有必要执行两次连续的读访问并验证返回的两个值是否相同。当两次连续读取访问的值相等时，可以认为读取访问是可靠的。

17.6.8. LPTIM 寄存器映像

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	LPTIM_ISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARRM	Res.
	Reset value																															0	
0x004	LPTIM_ICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARRMC	Res.
	Reset value																															0	
0x008	LPTIM_IER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARRMI	Res.
	Reset value																															0	
0x00C	LPTIM_CFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRELOAD	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRESC [2:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value										0											0	0	0									
0x010	LPTIM_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RSTARE	Res.	Res.	SNGSTR	ENABLE
	Reset value																												0			0	0
0x018	LPTIM_ARR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARR[15:0]															

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
0x1C	LPTIM_CNT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CNT[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

18. 独立看门狗（IWDG）

18.1. 简介

芯片内集成了一个 Independent watchdog（简称 IWDG），该模块具有高安全级别、时序精确及灵活使用的特点。IWDG 发现并解决由于软件失效造成的功能混乱，并在计数器达到指定的 timeout 值时触发系统复位。

IWDG 由 LSI 提供时钟，这样即使主时钟 Fail，也能保持工作。

IWDG 最适合需要 watchdog 作为主应用之外的独立过程，并且无很高的时序准确度限制的应用。

18.2. IWDG 主要特性

- Free-running 向下计数器
- 由 LSI 提供时钟（在 stop 模式也可以工作）
- 有条件的复位
  - 当向下计数器值为 0x000 复位

18.3. IWDG 功能描述

18.3.1. IWDG 框图

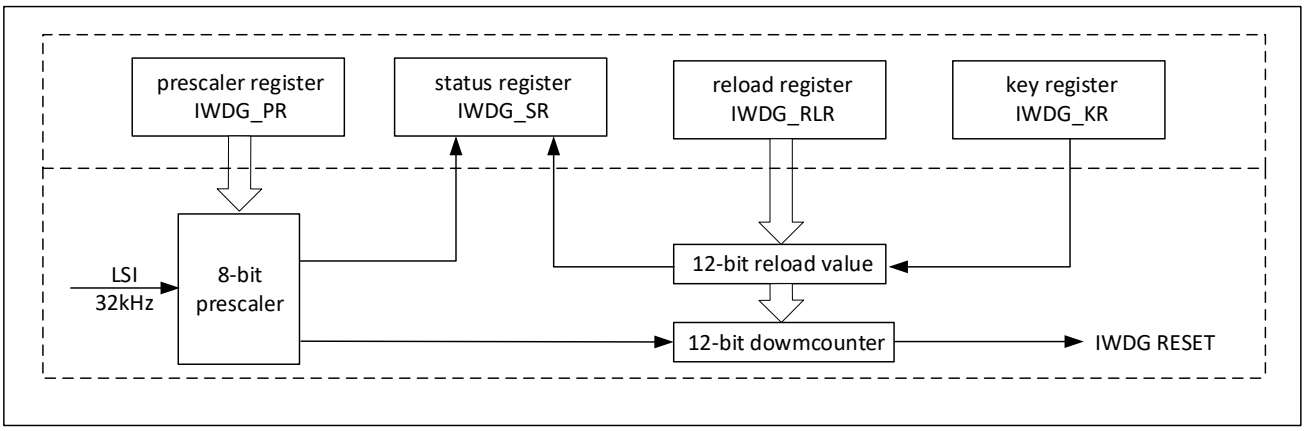


图 18-1 IWDG 框图

当通过向 IWDG 密钥寄存器(IWDG_KR)写 0x0000 CCCC，计数器开始从 0xFFF 向下计数。当到达计数最终值时（0x000），产生一个复位信号（IWDG 复位）。

不管何时，0x0000 AAAA 被写入 IWDG 密钥寄存器时，IWDG_RLR（reload 寄存器）的值被再次装载到计数器中，IWDG 不会产生复位。

一旦运行，则 IWDG 不能被停止。

18.3.2. 硬件看门狗

如果上电装载的选项字节（option bytes）设置了打开硬件 watchdog，则 IWDG 上电被自动使能，并且如果在计数器计数到终值之前，IWDG key 寄存器没被软件改写，则产生复位信号。

18.3.3. 硬件访问保护

对寄存器 IWDG 预分频、IWDG 重装载的写访问是被保护的。对这些寄存器的写其他数将破坏时序，如写 0x0000AAAA 加载，寄存器将被再次保护。

如果预分频寄存器、重装载寄存器的值正在更新，状态寄存器是会体现出来的。

18.3.4. 调试模式

本功能为系统支持 DBG_MCU 时才存在。

如果 CPU 进入调试模式，IWDG 继续计数还是进入 stop 模式，取决于 DBG 模块中 DBG_IWDG_STOP 的配置。

18.4. IWDG 寄存器

18.4.1. 密钥寄存器 (IWDG_KR)

Address offset:0x00

Reset value:0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31:16	Reserved	RES	-	Reserved
15:0	KEY[15:0]	W	0x00	Key 值。 软件必须以一定的时间间隔向该寄存器写入 0xAAAA，否则，当计数器计数到 0 时，看门狗会产生复位。 0x5555：表示允许访问 IWDG_PR、IWDG_RLR 寄存器； 0xCCCC：表示启动 IWDG（如果选择了硬件看门狗则不受此命令字限制）。

18.4.2. 预分频寄存器 (IWDG_PR)

Address offset:0x04

Reset value:0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31:3	Reserved	RES	-	Reserved
2:0	PR[2:0]	RW	0	预分频值。 通过配置该寄存器选择计数器时钟的预分频值。 要改变该寄存器，IWDG_SR 寄存器的 PVU 必须为 0。 000：4 分频； 001：8 分频； 010：16 分频； 011：32 分频； 100：64 分频； 101：128 分频； 110：256 分频； 111：256 分频；

18.4.3. 重装载寄存器 (IWDG_RLR)

Address offset:0x08

Reset value:0x0000 0FFF

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

Bit	Name	R/W	Reset Value	Function
31:12	Reserved	RES	-	Reserved
11:0	RL[11:0]	RW	0	IWDG 计数器重装载值。 当向 IWDG_KR 寄存器写入 0xAAAA 时，RL 值会传送到计数器中。随后计数器从这个值开始递减计数。看门狗超时周期可通过此 RL 值和时钟预分频值来计算。 只有当 IWDG_SR.RVU=0 时，才能对寄存器进行修改。

18.4.4. 状态寄存器 (IWDG_SR)

Address offset:0x0C

Reset value:0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31:2	Reserved	RES	-	Reserved
1	RVU	R	0	看门狗计数器重装值更新。 该位由硬件置 1，表明重装载值正在更新。当重装载值更新结束后，此位由硬件清零。
0	PVU	R	0	看门狗预分频值更新。 该位由硬件置 1，表明预分频值正在更新。当预分频值更新结束后，此位由硬件清零。

注：在更新 IWDG_PR、IWDG_SR.RLR 前，要分别等待 IWDG_PVU、IWDG_SR.RVU 为 0。但在更新 IWDG_PR、IWDG_RLR 后，不必再等待 IWDG_SR.PVU、IWDG_SR.RVU 为 0，可继续执行下面的代码。

18.4.5. IWDG 寄存器映像

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	IWDG_KR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	KEY[15:0]															
	Reset-set value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x004	IWDG_PR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PR[2:0]		
																															0	0	0
0x008	IWDG_RLR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RL[11:0]											
																						0	0	0	0	0	0	0	0	0	0	0	0
0x	IWDG	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RVU	PVU

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

19. I2C 接口

19.1. 介绍

I2C(inter-integrated circuit)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总线特定的顺序、协议、仲裁和时序。支持标准（Sm）、快速（Fm）。

19.2. I2C 主要特点

- Slave 和 master 模式
- 多主机功能：可以做 master，也可以做 slave
- 支持不同通讯速度
  - 标准模式（Sm）：高达 100kHz
  - 快速模式（Fm）：高达 400kHz
- 作为 Master
  - Clock 产生
  - Start 和 Stop 的产生
- 作为 slave
  - 可编程的 I2C 地址检测
  - Stop 位的发现
- 7 位寻址模式
- 通用广播（General call）
- 状态标志位
  - 发送/接收模式标志位
  - 字节传输完成标志位
  - I2C busy 标志位
- 错误标志位
  - 主机仲裁丢失
  - 地址/数据传输后的 ACK failure
  - Start/Stop 错误
  - Overrun/Underrun(时钟拉长功能禁止)
- 可选的时钟拉长功能
- 软件复位
- 模拟噪声滤波功能
- 可配置的 PEC（packet error checking）产生和验证
  - PEC 值可以在 Tx 模式下的最后一个 bytes 发送
  - 接收最后 byte 做 PEC 错误检查

19.3. I2C 功能描述

19.3.1. I2C 框图

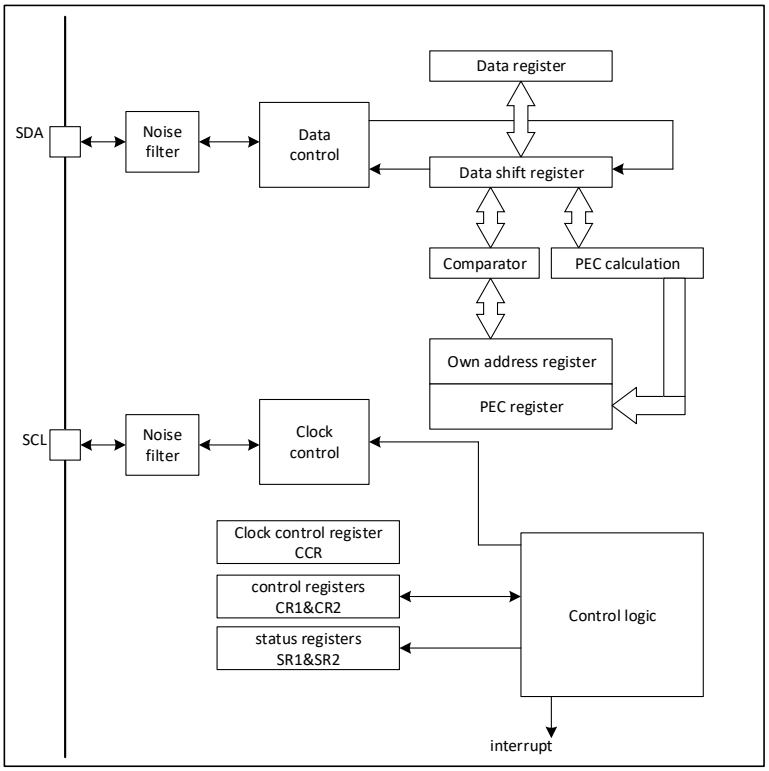


图 19-1 I2C 框图

19.3.2. 模式选择

I2C 支持以下四种模式：

- 从发送器模式 （Slave transmitter）
- 从接收器模式 （Slave receiver）
- 主发送器模式（Master transmitter）
- 主接收器模式（Master receiver）

默认模式为从模式。接口在生成起始条件后自动从从模式切换到主模式；当仲裁丢失或产生停止信号，则从主模式切换到从模式。允许多主机功能。

19.3.2.1. 通信流

作为 master，I2C 接口启动数据传输，并产生时钟信号。串行数据的传输总是以起始条件开始，并以停止条件结束。起始条件和停止条件都是在 master 模式下由软件控制产生。

作为 slave，I2C 接口能识别自己的地址(7 位)和 general call 地址。软件能够控制开启或禁止对 general call 地址的识别。

数据和地址按 8 位（字节）进行传输，高位在前。跟在 Start 条件后的 1 个字节是地址。地址只在 master 模式发送。

在一个字节传输的 8 个时钟后的第 9 个时钟期间，接收方必须回送一个应答位(ACK)给发送方。参加下图。

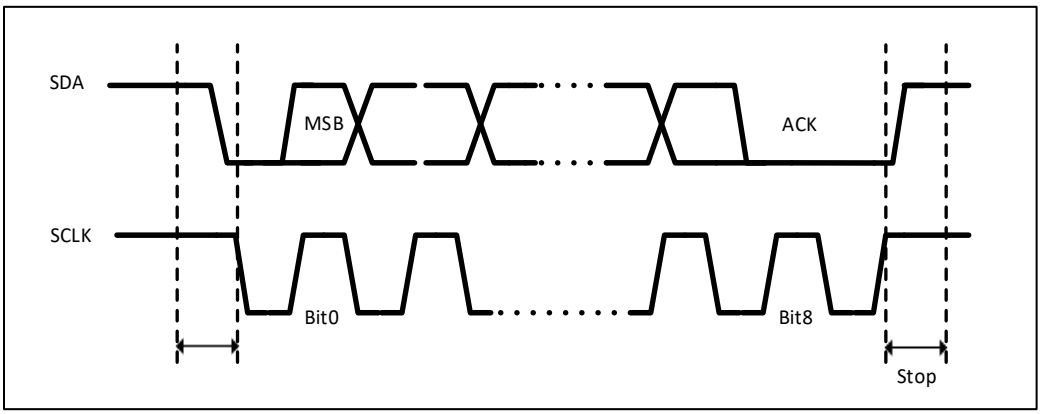


图 19-2 I2C 总线协议

软件可使能或不使能 ACK（应答）位。软件也可以选择 I2C 接口地址（7-bit 与/或 general call 地址）。

19.3.3. I2C 初始化

19.3.3.1. 使能/关闭 I2C 模块

I2C 的时钟模块先要通过 RCC_APBENR1 寄存器的 I2C_EN 位打开，然后通过设定 I2C_CR1 的 PE 位使能 I2C 模块。

19.3.3.2. I2C 时序设定

为满足在 master 和 slave 模式下，准确的数据 hold 和 setup 时间，需要进行 I2C 时序的设定。这是通过写 I2C_CCR 和 I2C_TRISE 寄存器实现的。

19.3.4. I2C 从模式

默认情况下，I2C 接口总是工作在 slave 模式。从 slave 模式切换到 master 模式，需要产生一个起始条件。

为了产生正确的时序，必须在 I2C_CR2 寄存器中设定该模块的输入时钟。输入时钟的频率必须至少是：

标准模式下为：2MHz

快速模式下为：4MHz

一旦检测到起始条件，在 SDA 线上接收到的地址，被送到 shift 寄存器，并与芯片的地址 OAR1 或者 general call 地址(如果 ENGC=1)相比较。

头段或地址不匹配：

I2C 接口将其忽略并等待另一个起始条件。

地址匹配：

I2C 接口产生以下时序：

- 如果 ACK 被软件置'1'，则产生一个应答脉冲
- 硬件置位 ADDR 位，如果设置了 ITEVTEN 位，则产生中断

在从模式下 TRA 位指示当前是处于接收器模式还是发送器模式。

19.3.4.1. 从发送器

在接收到地址并清除 ADDR 位后，（如果地址字节的最低位是 1）Slave 将数据（字节）从 DR 寄存器，经由内部 shift 寄存器发送到 SDA 上。

Slave 拉低 SCL，直到 ADDR 位被清除，并且待发送数据已写入 DR 寄存器。

当收到应答脉冲时：TxE 位被硬件置位，如果设置了 ITEVTEN 和 ITBUFEN 位，则产生一个中断。

如果 TxE 位被置位，但在下一个数据发送结束之前，没有新数据写入到 I2C_DR 寄存器，则 BTF 位被置位。Slave 拉低 SCL，直到 BTF 位被软件清零（读_SR1 之后，再写入 I2C_DR 寄存器）。

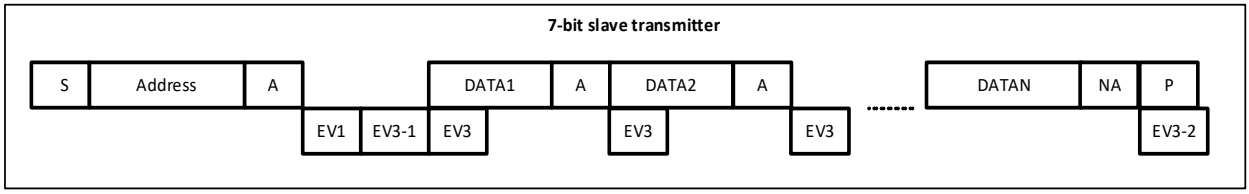


图 19-3 从发送器的传送序列图

Legend: S= Start（起始条件）, Sr= Repeated Start（重复的起始条件）, P= Stop（停止条件）, A= Acknowledge（响应）, NA= Non-acknowledge（不响应）, EVx= Event(ITEVFEN= 1 时产生中断)

EV1: ADDR=1, 通过先读 SR1 寄存器，再读 SR2 寄存器清零 ADDR 位

EV3-1：TxE=1, shift 寄存器 empty, 数据寄存器 empty, 向 DR 寄存器写 Data1

EV3：TxE=1, shift 寄存器不 empty, 数据寄存器 empty，向 DR 寄存器写（Data2）清零 TxE

EV3-2: AF=1; 软件向 AF 位写 0 清零该位

19.3.4.2. 从接收器

在接收到地址并清除 ADDR 后，（如果地址字节的最低位是 0）slave 将通过内部移位寄存器把从 SDA 线接收到的字节存进 DR 寄存器。I2C 接口在接收到每个字节后都执行下列操作：

- 如果设置了 ACK 位，则产生一个应答脉冲
- 硬件设置 RxNE=1。如果设置了 ITEVTEN 和 ITBUFEN 位，则产生一个中断。

如果 RxNE 被置位，并且在接收新的数据结束之前，DR 寄存器未被读出，则 BTF 位被置位，在清除 BTF（读出 I2C_SR1 之后再 I2C_DR 寄存器）之前，slave 一直拉低 SCL。(见下图)。

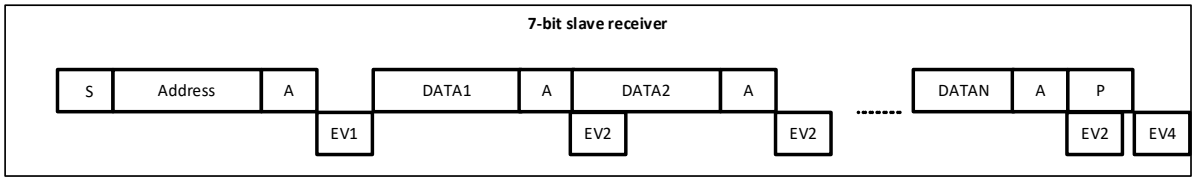


图 19-4 从接收器的传送序列图

Legend: S= Start, Sr= Repeated Start, P= Stop, A= Acknowledged, EVx= Event(with interrupt if ITEVFEN=1)

EV1：ADDR=1，通过先读 SR1，后读 SR2，实现 ADDR 的清零

EV2：RxNE=1, 读 DR 寄存器清零该位

EV4：STOPF=1, 通过先读 SR1 寄存器，后写 CR1 寄存器实现对该位的清零。

Note:

1) EV1 event 拉低 SCL，直到相应软件 sequence 的结束。
2) EV2 软件 sequence 必须在当前 byte 传输完成前即完成。
3) 当用户检查 SR1 寄存器内容后，应该对每个发现置位的标志位，进行完整的清除 sequence。比如 ADDR 和 STOPF 标志位，需要用以下 sequence：

   如果 ADDR=1，先读 SR，再读 SR2；如果 STOPF=1，先读 SR1，再写 CR1。

   这样做的目的是确保如果 ADDR 和 STOPF 两位都被发现置位，都能被清除掉。

19.3.4.3. 关闭通信

在传输完最后一个数据字节后，master 产生一个停止条件，slave 检测到该条件时：

- 硬件置位 STOPF，如果设置了 ITEVTEN 位，则产生一个中断。

通过先读 SR1，后写 CR1，实现对 STOPF 位的清零。（参见上图的 EV4）

19.3.5. I2C 主模式

在 Master 模式时，I2C 接口启动数据传输并产生时钟信号。串行数据传输总是以起始条件开始，并以停止条件结束。

当通过 START 位在总线上产生了起始条件，设备就进入了 master 模式。

以下是 master 模式所要求的操作顺序：

- 在 I2C_CR2 寄存器中设定该模块的输入时钟以产生正确的时序
- 配置时钟控制寄存器
- 配置上升时间寄存器
- 编程 I2C_CR1 寄存器启动外设
- 置 I2C_CR1 寄存器中的 START 位为 1，产生起始条件

  I2C 模块的输入时钟频率必须至少是：
- 标准模式下为：2MHz
- 快速模式下为：4MHz

19.3.5.1. 主机产生 clock

CCR 寄存器以上升沿或者下降沿，产生 SCL 的高电平和低电平。由于 slave 可能拉长 SCL 信号，在 SCL 上升沿产生后，master 在 TRISE 寄存器编程的时间到达时，检查来自总线的 SCL 信号。

— 如果 SCL 是低电平，意味着 slave 正在拉长 SCL 总线，高电平计数器停止计数，直到 SCL 被检测到高电平。这是为了确保 SCL 参数的最小高电平时间。

— 如果 SCL 是高电平，高电平计数器保持计数。

实际上，即使 slave 不拉长 SCK，从 SCL 上升沿产生，到 SCL 上升沿被发现，这样的反馈环路也是要花费些时间的。这个回路的时间与 SCL 的上升时间（SCL 的 VIH 数据检测）有关系，再加上 SCL 输入路径的模拟噪声滤波，以及芯片内部由于用 APB 时钟进行的 SCL 同步。反馈回路的最大时间编程在 TRISE 寄存器中，所以无论 SCL 上升时间如何，SCL 的频率保持稳定。

19.3.5.2. 开始条件

当 BUSY=0 时，设置 START=1，I2C 接口将产生一个 Start 条件，并切换至 master 模式(MSL 被置位)。

注：在 master 模式下设置 START 位，将在当前字节传输完后，由硬件产生一个 ReStart 条件。

一旦发出 Start 条件：

● SB 位被硬件置位，如果设置了 ITEVTEN 位，则会产生一个中断。

master 读 SR1 寄存器，再把 slave 地址写入 DR 寄存器。（Transfer sequence EV5）

19.3.5.3. 从机地址发送

slave 地址通过内部移位寄存器被送到 SDA 线上。

● 在 7 位地址模式时，送出一个地址字节。

该地址字节一旦被送出，

─ ADDR 位被硬件置位，如果设置了 ITEVTEN 位，则产生一个中断。

随后 Master 读 SR1 寄存器，跟着读 SR2 寄存器。

根据送出 slave 地址的最低位，master 决定进入发送器模式，还是进入接收器模式。

● 在 7 位地址模式时，

─ 要进入发送器模式，主设备发送从地址时置最低位为’0’。

─ 要进入接收器模式，主设备发送从地址时置最低位为’1’。

TRA 位指示主设备是在接收器模式还是发送器模式。

19.3.5.4. 主机发送

在发送了地址和清除了 ADDR 位后, 主设备 master 通过内部移位寄存器将字节从 DR 寄存器发送到 SDA 线上。

Master 等待，直到第一个数据字节被写入 DR 寄存器（参见 EV8_1）。

当收到 ACK 脉冲时，TxE 位被硬件置位，如果设置了 INEVFEN 和 ITBUFEN 位，则产生一个中断。

如果 TxE 被置位，且在上一次数据发送结束之前，没有写新的数据字节到 DR 寄存器，则 BTF 被硬件置位。在清除 BTF（读 I2C_SR1 之后，再写 I2C_DR 寄存器）之前，I2C 接口将保持 SCL 为低电平。

关闭通信

在 DR 寄存器中写入最后一个字节后，通过设置 STOP 位产生一个停止条件(见图的 EV8_2)，然后 I2C 接口将自动回到从模式(MLS 位清除)。

注：当 *TxE* 或 *BTF* 位置位时，停止条件应安排在出现 *EV8_2* 事件时。

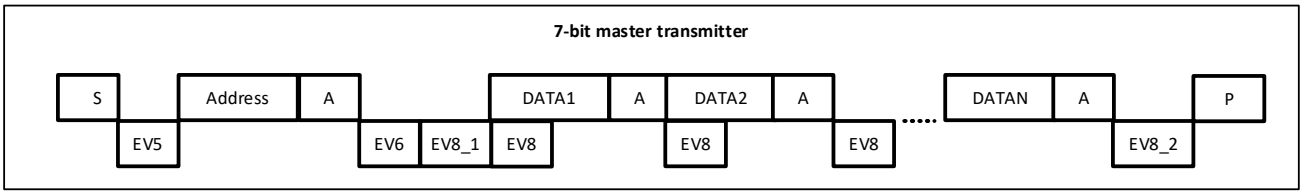


图 19-5 主发送器传送序列图

Legend: S= Start, Sr= Repeated Start, P= Stop, A= Acknowledge, EVx= Event(with interrupt if ITEVFEN= 1)

EV5：SB=1, 通过读 SR1，再向 DR 寄存器写数据，实现对该位的清零

EV6：ADDR=1，通过读 SR1，再读 SR2，实现对该位的清零

EV8_1：TxE=1, shift 寄存器 empty，数据寄存器 empty，向 DR 寄存器写 Data1

EV8：TxE=1, shift 寄存器不 empty，数据寄存器 empty，向 DR 寄存器写 Data2，该位被清零

EV8_2：TxE=1, BTF=1, 写 Stop 位寄存器，当硬件发出 Stop 位时，TxE 和 BTF 被清零

Note:

1- EV5, EV6, EV8_1 和 EV8_2 事件，拉长 SCL 的低电平，直到相应的软件 sequence 执行结束

2- EV8 软件 sequence 必须在当前字节发送完成前执行完毕。若 EV8 的软件 sequence 不能在当前传输的字节结束前完成，则推荐使用 BTF 代替 TxE，这产生的不利是减慢了通讯。

19.3.5.5. 主接收器

在发送地址和清除 ADDR 之后，I2C 接口进入主接收器模式。在此模式下，I2C 接口从 SDA 线接收数据字节，并通过内部移位寄存器送至 DR 寄存器。在每个字节后，I2C 接口依次执行以下操作：

- 如果 ACK 位被置位，发出一个应答脉冲。
- 硬件设置 RxNE=1，如果设置了 INEVFEN 和 ITBUFEN 位，则会产生一个中断。

如果 RxNE 位被置位，并且在接收新数据结束前， DR 寄存器中的数据没有被读走，硬件将设置 BTF=1，在清除 BTF 之前 I2C 接口将保持 SCL 为低电平；读出 I2C_SR1 之后再读出 I2C_DR 寄存器将清除 BTF 位。

关闭通信

Method 1:该方法的应用场景是：当 I2C 被设成应用程序中最高优先级的中断

Master 在从 Slave 接收到最后一个字节后，发送一个 NACK。接收到 NACK 后，Slave 释放对 SCL 和 SDA 线的控制。Master 就可以发送一个 Stop/Restart 条件。

1） 为了在收到最后一个字节后产生一个 NACK 脉冲，在读倒数第二个数据字节之后(在倒数第二个 RxNE 事件之后)必须清除 ACK 位。

2） 为了产生一个停止/重起始条件，软件必须在读倒数第二个数据字节之后(在倒数第二个 RxNE 事件之后)设置 STOP/START 位。

3） 当接收单个字节时，关闭应答和停止条件的产生位要刚好在 EV6 之后(EV6_1 时，清除 ADDR 之后)。
在产生了停止条件后，I2C 接口自动回到从模式(MSL 位被清除)。

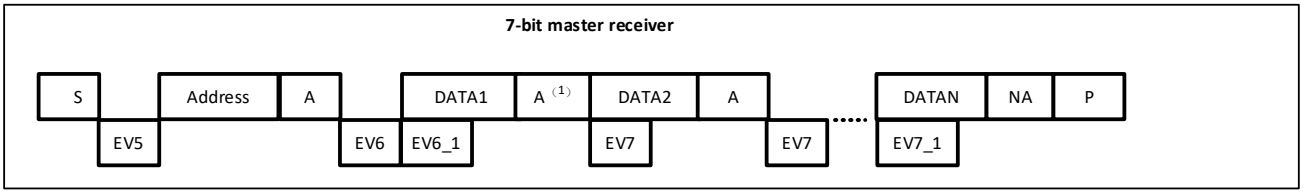


图 19-6 方法 1: 主模式发送时的时序

Legend: S= Start, Sr= Repeated Start, P= Stop, A= Acknowledge, EVx= Event(with interrupt if ITEVFEN= 1)

EV5：SB=1，读 SR1，再写 DR 寄存器，该位被清零

EV6：ADDR=1，读 SR1，再读 SR2，该位被清零

EV6_1：无相关的标志事件，仅用作 1 个字节的接收。

EV7：RxNE=1，读 DR 寄存器，该位被清零

EV7_1：RxNE=1, 读 DR 寄存器，写 ACK=0 并置位 STOP

1） 如果是单个字节接收，则上述标注为（1）的地方会是 NA

2） EV5, EV6 事件，拉长 SCL 的低电平，直到相应的软件 sequence 执行结束

3） EV7 软件 sequence 必须在当前字节发送完成前执行完毕。在 EV7，软件 sequence 不能在当前传输的字节传输完成前，被管理。推荐使用 BTF 代替 RXNE，这产生的不利是减慢了通讯。

4） EV6_1 或者 EV7_1 的软件 sequence 必须在当前字节传输的 ACK 之前完成。

Method 2: 这个方法的应用场景是：I2C 的中断在应用中不是最高优先级，或者使用查询方式

用这个方法，DataN-2 没有被读，因此在 DataN-1 之后，通讯被拉长（RxNE 和 BTF 都被置位）。然后，在读 DR 寄存器的 DataN-2 前，清 ACK 位，以确保在 DataN ACK 之前被清掉。在此之后，在读 DataN-2 之后，置位 STOP/START 位，并读 DataN-1。在 RxNE 置位后，读 DataN

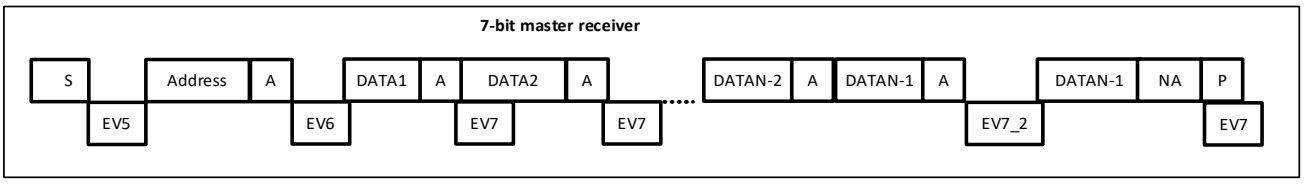


图 19-7 方法 2: N>2 时主模式发送时的时序

Legend: S= Start, Sr= Repeated Start, P= Stop, A= Acknowledge, EVx= Event(with interrupt if ITEVFEN= 1)

EV5：SB=1, 先读 SR1 寄存器，再写 DR 寄存器，清零该位

EV6：ADDR，先读 SR1，再读 SR2，清零该位

EV7：RxNE=1, 读 DR 寄存器清零该位

EV7_2：BTF=1, DataN-2 存在 DR 寄存器中，DataN-1 存在 shift 寄存器中，写 ACK=0，读 DR 寄存器中的 DataN-2。置位 STOP，读 DataN-1

Note:

1） EV5, EV6 事件，拉长 SCL 的低电平，直到相应的软件 sequence 执行结束。

2） EV7 软件 sequence 必须在当前字节发送完成前执行完毕。在 EV7，软件 sequence 不能在当前传输的字节传输完成前，被管理。推荐使用 BTF 代替 RXNE，这产生的不利是减慢了通讯。

●当 3 个字节要被读走：

— RxNE=1 ＝＞ Nothing(DataN-2 not read)。

— DataN-1 received

— BTF=1, shift 和 data 寄存器都是 full：DR 寄存器存放了 DataN-2，shift 寄存器存放了 DataN-1 ＝＞ SCL 拉低：总线上没有其他要被接收的数据

— 清零 ACK 位

— 读 DR 寄存器中的 DataN-2 ＝＞ 这将启动 shift 寄存器对 DataN 的接收

— DataN 接收完成（with a NACK）

— 写 START 或者 STOP 位

— 读 DataN-1

— RxNE=1

— 读 DataN

以上流程是针对 N > 2 的描述。1 个字节和 2 个字节的接收，要用不同的处理方式，参见以下描述：

● 2 个字节接收的情况

— 置位 POS 和 ACK 位

— 等待 ADDR 置位

— 清零 ADDR 位

— 清零 ACK 位

— 等待 BTF 被置位

— 写 STOP 位

— 读 DR 两次

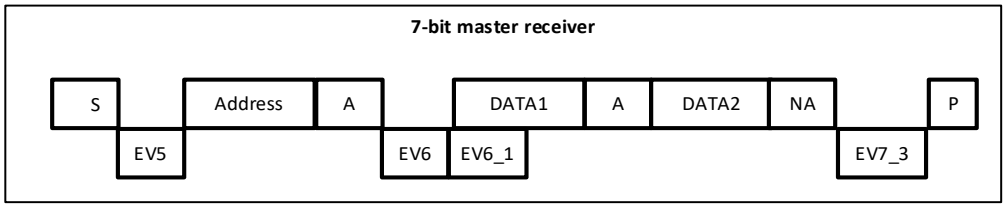


图 19-8 方法 2: N=2 时主模式发送时的时序

Legend: S= Start, Sr= Repeated Start, P= Stop, A= Acknowledge, EVx= Event(with interrupt if ITEVFEN= 1)

EV5：SB=1, 先读 SR1 寄存器，再写 DR 寄存器，清零该位

EV6：ADDR=1，先读 SR1 寄存器，后读 SR2 寄存器，清零 ADDR 位

EV6_1：无相关的标志位事件。在 EV6 后，也就是地址被清零后，ACK 应该被清零

EV7_3：BTF=1，写 STOP=1，之后读两次 DR（Data1 和 Data2）

Note:

1) EV5, EV6 事件，拉长 SCL 的低电平，直到相应的软件 sequence 执行结束

2) EV6_1 的软件 sequence 必须在当前字节传输的 ACK 之前完成

● 单个字节接收的情况

— 在 ADDR 事件里，清零 ACK 位

— 清零 ADDR

— 写 STOP 或者 START 位

— 在 RxNE 标志置位后，读数据

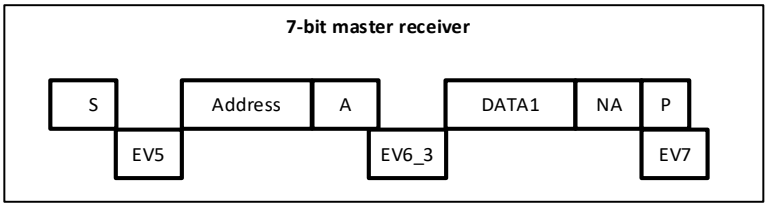


图 19-9 方法 2: N=1 时主模式发送时的时序

Legend: S= Start, Sr= Repeated Start, P= Stop, A= Acknowledge, EVx= Event(with interrupt if ITEVFEN= 1)

EV5：SB=1, 先读 SR1 寄存器，再写 DR 寄存器，清零该位

EV6_3：ADDR=1，写 ACK=0。先读 SR1 寄存器，后读 SR2 寄存器，清零 ADDR 位。在 ADDR 被清零后，写 STOP=1

EV7：RxNE=1, 读 DR 寄存器清零该位

Note:

EV5 事件会拉长 SCL 的低电平，直到相应的软件 sequence 执行结束。

19.3.6. 错误状态

19.3.6.1. 总线错误

在一个地址或数据字节传输期间，当 I2C 接口检测到一个外部的停止或起始条件则产生总线错误。此时：

● BERR 位被置位为'1'；如果设置了 ITERREN 位，则产生一个中断；

● 在 slave 模式：数据被丢弃，硬件释放总线：

– 如果是错误的 Start 条件，slave 认为是一个 Restart，并等待地址或停止条件。

– 如果是错误的 Stop 条件，slave 按正常的停止条件操作，同时硬件释放总线。

● 在 master 模式：硬件不释放总线，同时不影响当前的传输状态。此时由软件决定是否要中止当前的传输。

19.3.6.2. 应答失败(AF)

当接口检测到一个无应答位时，产生应答错误。此时：

● AF 位被置位，如果设置了 ITERREN 位，则产生一个中断

● 当发送器接收到一个 NACK 时，必须复位通讯：

– 如果是处于 slave 模式，硬件释放总线。

– 如果是处于 master 模式，软件必须生成一个停止条件或者 repeated start。

19.3.6.3. 仲裁丢失 (ARLO)

当 I2C 接口检测到仲裁丢失时产生仲裁丢失错误，此时：

● ARLO 位被硬件置位，如果设置了 ITERREN 位，则产生一个中断

● I2C 接口自动回到从模式(MSL 位被清除)。当 I2C 接口丢失了仲裁，则它无法在同一个传输中响应它的从地址，但它可以在赢得总线的 master 发送 repeated start 条件之后响应

● 硬件释放总线

19.3.6.4. 过载 overrun/欠载 underrun (OVR)

在 slave 模式下，如果禁止时钟延长，I2C 接口正在接收数据时，当它已经接收到一个字节(RxNE=1)，但在 DR 寄存器中前一个字节数据还没有被读出，则发生 overrun 错误。

此时：

● 最后接收的数据被丢弃

● 在 overrun 错误时，软件应清除 RxNE 位，发送器应该重新发送最后一次发送的字节

在 slave 模式下，如果禁止时钟延长，I2C 接口正在发送数据时，在下一个字节的时钟到达之前，新的数据还未写入 DR 寄存器(TxE=1)，则发生欠载错误。此时：

● 在 DR 寄存器中的前一个字节将被重复发出

● 用户应该确定在发生 underrun 错时，接收端应丢弃重复接收到的数据。发送端应按 I2C 总线标准在规定的时间更新 DR 寄存器

在发送第一个字节时，必须在清除 ADDR 之后且在第一个 SCL 上升沿之前写入 DR 寄存器；如果不能做到这点，则接收方应该丢弃第一个数据。

19.3.7. SDA/SCL 控制

● 如果允许时钟延长：

— 发送器模式：如果 TxE=1 且 BTF=1：I2C 接口在传输前保持时钟线为低，以等待软件读取 SR1，然后把数据写进数据寄存器(DR 和 shift 寄存器都是空的)。

— 接收器模式：如果 RxNE=1 且 BTF=1：I2C 接口在接收到数据字节后保持时钟线为低，以等待软件读 SR1，然后读数据寄存器 DR(DR 和 shift 寄存器都是满的)。

● 如果在 slave 模式中禁止时钟延长：

— 如果 RxNE=1，在接收到下个字节前 DR 还没有被读出，则发生 overun。接收到的最后一个字节丢失。

— 如果 TxE=1，在必须发送下个字节之前却没有新数据写进 DR，则发生 underrun。相同的字节将被重复发出。

— 硬件未实现对写冲突的控制。

19.4. I2C 中断

表 19-1 I2C 中断请求

中断事件	事件标志	开启控制位
起始位已发送(Master)	SB	ITEVTEN
地址已发送(Master) 或 地址匹配(Slave)	ADDR	
已收到停止(Slave)	STOPF	
数据字节传输完成	BTF	
接收缓冲区非空	RxNE	ITEVTEN 和 ITBUFEN
发送缓冲区空	TxE	
总线错误	BERR	ITERREN
仲裁丢失(Master)	ARLO	
响应失败	AF	
过载/欠载	OVR	
PEC 错误	PECERR	

19.5. I2C 寄存器

寄存器可以 half-word 或者 word 访问。

19.5.1. I2C 控制寄存器 1 (I2C_CR1)

Address offset:0x00

Reset value:0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWRST	Res	Res	Res	POS	ACK	STOP	START	NOSTRETCH	ENGC	Res	Res	Res	Res	Res	PE
RW				RW	RW	RW	RW	RW	RW						RW

Bit	Name	R/W	Reset Value	Function
15	SWRST	RW	0	软件复位。 当被置位时，I2C 处于复位状态。在复位释放前，要确保 I2C 的引脚被释放，总线是空闲状态。 0：I2C 模块不处于复位状态 1：I2C 模块处于复位状态 注：该位可以用于 error 或 locked 状态时重新初始化 I2C。如 BUSY 位为 1，在总线上又没有检测到停止条件时。
14:12	Reserved	RES	-	Reserved
11	POS	RW	0	ACK/PEC 位置（用于数据接收），软件可置位/清零该寄存器，或 PE=0 时由硬件清零。 0：ACK 位控制当前移位寄存器内正在接收的字节的(N)ACK。PEC 位表明当前移位寄存器内的字节是 PEC 1：ACK 位控制在移位寄存器里接收的下一个字节的(N)ACK。PEC 位表明在移位寄存器里接收的下一个字节是 PEC 注：POS 位只能用在 2 字节的接收配置中，必须在接收数据之前配置。 为了 NACK 第 2 个字节，必须在清除 ADDR 之后清除 ACK 位。 为了检测第 2 个字节的 PEC，必须在配置了 POS 位之后，ADDR stretch 事件时设置 PEC 位。
10	ACK	RW	0	应答使能。软件可置位/清零该寄存器，或 PE=0 时由硬件清零。 0：无应答返回 1：在接收到一个字节后返回一个应答。（匹配的地址或数据）
9	STOP	RW	0	停止条件产生，软件可以置位/清零该寄存器，或者当检测到停止条件时，由硬件清除；当检测到超时错误时，硬件置位。 在主模式下： 0：无停止条件产生 1：在当前字节传输或在当前起始条件发出后产生停止条件 在从模式下： 0：无停止条件产生 1：在当前字节传输后释放 SCL 和 SDA 线
8	START	RW	0	起始条件产生。 软件可置位/清零该寄存器，或当起始条件发出后或 PE=0 时由硬件清零。 主模式： 0：无起始条件产生 1：重复产生起始条件 从模式： 0：无起始条件产生 1：当总线空闲时，产生起始条件（并由硬件自动切换到 master mode）
7	NOSTRETCH	RW	0	禁止时钟延长（Slave）。 当 ADDR 或 BTF 标志被置位时，该位用于 slave 禁止时钟延长，直到被软件复位。 0：允许时钟延长 1：禁止时钟延长
6	ENGC	RW	0	广播呼叫使能。 0：禁止广播呼叫。以 NACK 响应地址 00h 1：允许广播呼叫。以 ACK 响应地址 00h

Bit	Name	R/W	Reset Value	Function
5:1	Reserved	RES	-	Reserved
0	PE	RW	0	I2C 模块使能。 0：禁止 1：I2C 使能 注：如果清除该位时通讯正在进行，在当前通讯结束后，I2C 模块被禁用并返回空闲状态。 由于在通讯结束后 PE=0，所有的位被清除。 在主模式下，通讯结束之前，绝不能清除该位。

注：当设置了 STOP/START/PEC 位，在硬件清除这个位之前，软件不要执行任何对 I2C_CR1 的写操作；否则有可能会第 2 次设置 STOP/START/PEC 位。

19.5.2. I2C 控制寄存器 2 (I2C_CR2)

Address offset:0x04

Reset value:0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	Res	Res	IT-BUFEN	ITEV-TEN	ITER-REN	Res	Res	FREQ[5:0]					
					RW	RW	RW			RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
15:11	Reserved	RES	-	Reserved
10	ITBUFEN	RW	0	缓冲器中断使能。 0：当 TxE=1 或 RxNE=1 时，不产生中断 1：当 TxE=1 或 RxNE=1 时，产生事件中断
9	ITEVTEN	RW	0	事件中断使能。 0：禁止 1：允许事件中断 在下列条件下，将产生该中断： ● SB=1（主模式）； ● ADDR=1（主/从模式） ● STOPF=1（从模式） ● BTF=1，但没有 TxE 或 RxNE 事件 ● 如果 ITBUFFEN=1，TxE 事件为 1 ● 如果 ITBUFEN=1，RxNE 事件为 1
8	ITERREN	RW	0	出错中断使能。 0：禁止出错中断； 1：允许出错中断； 在下列条件下，将产生该中断： ● BERR=1 ● ARLO=1 ● AF=1 ● OVR=1 ● PECERR=1
7:6	Reserved	RES	-	Reserved
5:0	FREQ	RW	0	I2C 模块时钟频率。 必须用 APB 时钟频率的值配置该寄存器，以产生与 I2C 协议兼容的数据 setup 和 hold 时间。 最小允许可设定的频率是 4MHz（标准模式，即 100k）、12MHz（400k），最大频率是芯片最高的 APB 时钟频率。 000000：禁止 000001：禁止 001000：8MHz ….. 011000：24MHz 其它：禁止。

19.5.3. I2C 自身地址寄存器 1 (I2C_OAR1)

Address offset:0x08

Reset value:0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	Res	Res	Res	Res		ADD[7:1]							Res
									RW	RW	RW	RW	RW	RW	

Bit	Name	R/W	Reset Value	Function
14:8	Reserved	RES	-	Reserved
7:1	ADD[7:1]	RW	0	接口地址的 7~1 位。
0	reserved			

19.5.4. I2C 数据寄存器 (I2C_DR)

Address offset:0x10

Reset value:0x0000

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

Bit	Name	R/W	Reset Value	Function
15:8	Reserved	RES	-	Reserved
7:0	DR[7:0]	RW	0	8 位数据寄存器，芯片内部实际是两个独立的 buffer 共用一个地址，分别用于存放接收到的数据（RX_DR）、放置要发送到总线的数据（TX_DR）。 **发送器模式：** 当写一个字节至 DR 寄存器时（实际写到 TX_DR），自动启动数据传输。一旦传输开始（TxE=1），如果能及时把下一个需传输的数据写入 DR 寄存器，I2C 模块将保持连续的数据流。 **接收器模式：** 接收到的字节被拷贝到 DR 寄存器（实际是 RX_DR）（RxNE=1）。在接收到下一个字节（RxNE=1）之前读出数据寄存器，即可实现连续的数据接收。 注： 1) 在 slave 模式下，地址不会被 copy 进数据寄存器 DR 2) 硬件不处理写冲突（如果 TxE=0，仍能写入数据寄存器） 3) 如果在处理 ACK 脉冲时发生 ARLO 事件，接收到的字节不会被 copy 到数据寄存器里，因此不能读到

19.5.5. I2C 状态寄存器(I2C_SR1)

Address offset:0x14

Reset value:0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	PECERR	OVR	AF	ARLO	BERR	TxE	RxNE	Res	STOPF	Res	BTF	ADDR	SB
			RC_W0	RC_W0	RC_W0	RC_W0	RC_W0	R	R		R		R	R	R

Bit	Name	R/W	Reset Value	Function
15:13	Reserved	RES	-	Reserved
12	PECERR	RC_W0	0	在接收时发生 PEC 错误。 0：无 PEC 错误，接收到 PEC 后返回 ACK（如果 ACK=1） 1：有 PEC 错误，接收到 PEC 后返回 NACK（不管 ACK 为何值） 该位由软件写 0 清除，或当 PE=0 时由硬件清除。

Bit	Name	R/W	Reset Value	Function
11	OVR	RC_W0	0	过载/欠载标志。 0：无过载/欠载； 1：出现过载/欠载。 当 NOSTRETCH=1 时，在从模式下该位被硬件置位； 在接收模式中当收到一个新的字节时（包括 ACK 应答脉冲），数据寄存器里的内容还未被读出，则新接收的字节将丢失。 在发送模式中当要发送一个新的字节时，却没有新的数据写入数据寄存器，同样的字节将被发送两次。 该位由软件写 0 清除，或当 PE=0 时由硬件清除。 注：如果数据寄存器的写操作发生时间非常接近 SCL 的上升沿，发送的数据是不确定的，并发生保持时间错误。
10	AF	RC_W0	0	应答失败标志。 0：没有应答失败； 1：应答失败。 当没有返回应答时，硬件将置位该寄存器。 该位由软件写 0 清除，或当 PE=0 时由硬件清除。
9	ARLO	RC_W0	0	仲裁丢失（主模式）。 0：没有检测到仲裁丢失； 1：检测到仲裁丢失。 当接口失去对总线的控制给另一个主机时，硬件将置位该寄存器。 该位由软件写 0 清除，或在 PE=0 时由硬件清除。 在 ARLO 事件之后，I2C 接口自动切换回从模式（M/SL=0）。
8	BERR	RC_W0	0	总线出错标志。 0：无起始或者停止条件出错； 1：起始或者停止条件出错。 当接口检测到错误的起始或者停止条件，硬件将该位置 1。 该位由软件写 0 清除，或者在 PE=0 时由硬件清除。
7	TxE	R	0	数据寄存器为空（发送时）标志。 0：数据寄存器非空； 1：数据寄存器为空。 在发送数据时，数据寄存器为空时该位被置 1，在发送地址阶段不设置该位。 软件写数据到 DR 寄存器可清除该位，或在发生一个起始或停止条件后，或当 PE=0 时由硬件自动清除。 如果收到一个 NACK，或下一个要发送的字节时 PEC（PEC=1），该位不被置位。 注：在写入第 1 个要发送的数据后，或设置了 BTF 时写入数据，都不能清除 TxE 位，因为此时数据寄存器为空。
6	RxNE	R	0	数据寄存器非空（接收时）标志。 0：数据寄存器为空； 1：数据寄存器非空。 在接收时，当数据寄存器不为空，置位该寄存器。在接收地址阶段，该寄存器不置位。 软件对数据寄存器的读写操作会清除该寄存器，或当 PE=0 时由硬件清除。 注：当设置了 BTF 时，读取数据不能清除 RxNE 位，因为此时数据寄存器仍为满。
5	Reserved	RES	-	Reserved

Bit	Name	R/W	Reset Value	Function
4	STOPF	R	0	停止条件检测位（从模式）。 0：没有检测到停止位； 1：检测到停止条件。 在一个应答之后（如果 ACK=1），当从设备在总线上检测到停止条件时，硬件将该位置 1。 软件读取 I2C_SR1 寄存器后，对 I2C_CR1 寄存器的写操作将清除该位，或当 PE=0 时，硬件清除该位。 注：在收到 NACK 后，STOPF 位不会置位。
3	reserved			
2	BTF	R	0	字节传输结束标志位。 0：字节传输未完成 1：字节传输成功~~结束~~ 在下列情况下硬件将置位该寄存器（当 slave 模式，NOSTRETCH=0 时；master 模式，与 NOSTRETCH 无关）： — 接收时，当收到一个新字节（包括 ACK 脉冲）且数据寄存器还未被读取（RxNE=1）。 — 发送时，当一个新数据应该被发送，且数据寄存器还未被写入新的数据（TxE=1）。 软件读取 I2C_SR1 寄存器后，对数据寄存器的读或写操作将清除该位；或发送一个起始或停止条件后，或当 PE=0 时，由硬件清除。 注： 在收到一个 NACK 后，BTF 位不会被置位。
1	ADDR	R	0	地址已被发送（主模式）/地址匹配（从模式）。 软件读取 I2C_SR1 寄存器后，再读 I2C_SR2 寄存器将清除该位；当 PE=0 时，由硬件清除。 **地址匹配（Slave）：** 0：地址不匹配或没有收到地址； 1：收到的地址匹配。 当收到的从地址与 OAR 寄存器或 general call 地址匹配，硬件将置位该位。 **Note:** 在 slave 模式下，推荐进行完整的清零 sequece，即在 ADDR 被置位后，先读 SR1 寄存器，再读 SR2 寄存器。 **地址已发送（Master）：** 0：地址发送没有结束； 1：地址发送结束。 7 位地址时，当收到 ACK byte 后置位。 **Note:** 在收到 NACK 后，该寄存器不会被置位。
0	SB	R	0	起始位标志（主模式）。 0：未发送起始条件； 1：起始条件已发送； —当发送起始条件时，置位该寄存器。 —软件读取 I2C_SR1 寄存器后，对数据寄存器的写操作将清除该位；或当 PE=0 时，由硬件清除。

19.5.6. I2C 状态寄存器 2 (I2C_SR2)

Address offset:0x18

Reset value:0x0000

Note: 即使 ADDR 标志位在读 I2C_SR1 寄存器后被置位，在读 I2C_SR1 之后再读 I2C_SR2 寄存器，也会清零 ADDR 标志位。因此，仅在发现 I2C_SR1 寄存器的 ADDR 位被置位或者 STOPF 位被清零时，I2C_SR2 寄存器才必须被读。

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res								Res	Res	Res	GEN-CALL	Res	TRA	BUSY	MSL
R	R	R	R	R	R	R	R				R		R	R	R

Bit	Name	R/W	Reset Value	Function
15:7	reserved			
4	GENCALL	R	0	广播呼叫地址（从模式）。 0：未收到广播呼叫地址； 1：当 ENGC=1 时，收到广播呼叫的地址。 当产生一个停止条件或一个重复的起始条件时，或 PE=0 时，硬件清除该寄存器。
3	Reserved	RES	-	Reserved
2	TRA	R	0	发送/接收标志。 0：接收到数据 1：数据已发送 在整个地址传输阶段的结尾，该寄存器根据地址字节的 R/W 位来设定。 当检测到停止条件（STOPF=1），或者重复的起始条件、或者总线仲裁丢失（ARLO=1），或当 PE=0 时，硬件清除该寄存器。
1	BUSY	R	0	总线忙标志。 0：在总线上无数据通讯 1：在总线上正在极性数据通讯 当检测到 SDA 或 SCL 为低电平时，硬件置位。 当检测到一个停止条件时，硬件清零。 该寄存器指示当前正在进行的总线通讯，当接口被禁用（PE=0）时该信息仍然被更新。
0	MSL	R	0	主从模式。 0：slave 1：master —当接口处于主模式（SB=1）时，硬件置位； —当总线上检测到一个停止条件（STOPF=1）、仲裁丢失（ARLO=1）、或当 PE=0 时，硬件清零。

19.5.7. I2C 时钟控制寄存器(I2C_CCR)

Address offset:0x1C

Reset value:0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
F/S	DUTY	Res	Res	CCR[11:0]											
RW	RW			RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
15	F/S	RW	0	I2C 主模式选择。 0：标准模式 1：快速模式
14	DUTY	RW	0	快速模式时的占空比。 0：快速模式下：$T_{low}/T_{high}=2$ 1：快速模式下：$T_{low}/T_{high}=16/9$
13:12	Reserved	RES	-	Reserved
11:0	CCR[11:0]	RW	0	快速/标准模式下的时钟控制分频系数（主模式）。 该分频系数用于设置主模式下的 SCL 时钟。 ● 标准模式： ✓ T_{high}=CCR x Tpclk ✓ T_{low} =CCR x Tpclk ● 快速模式： ✓ DUTY=0: T_{high}=CCR x Tpclk

Bit	Name	R/W	Reset Value	Function
				T_{low} =2 x CCR x Tpclk ✓ DUTY=1(为达到 400KHz): T_{high}=9 x CCR x Tpclk T_{low} =16 x CCR x Tpclk 注： 1. 允许设定的最小值为 0x04，在快速 DUTY 模式下允许的最小值为 0x01 2. $T_{high}=t_{r(SCL)}+t_{w(SCLH)}$ 3. $T_{low}=t_{r(SCL)}+t_{w(SCLL)}$ 4. 这些延时没有过滤器 5. 只有当 PE=0 时才能配置该寄存器； 6. f_{CK} 应当是 10MHz 的整数倍，这样可以正确产生 400kHz 的快速时候在哪个

19.5.8. I2C TRISE 寄存器 (I2C_TRISE)

Address offset:0x20

Reset value:0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	TRISE[5:0]					
										RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
15:6	Reserved	RES	-	Reserved
5:0	TRISE	RW	0	在快速/标准模式下的最大上升时间（主模式）。这些位应该提供在 master mode 下， SCL 反馈回路的最大持续时间。这样做的目的是无论 SCL 上升沿持续时间多少，SCL 都能保持一个稳定的频率。 这些位必须设置为 I2C 总线规范里给出的最大的 SCL 上升时间，增长步幅为 1。 例如：标准模式中最大允许 SCL 上升时间为 1000ns。如果在 I2C_CR2 寄存器中 FREQ[5:0] 中的值等于 0x08，Tpclk=125ns，则 TRISE 中配置为 0x09（1000ns/125ns = 8 + 1 = 9）。 滤波器的值也可以加到 TRISE 内。 如果结果不为整数，则将整数部分写入 TRISE，以确保 t_{HIGH} 参数。 注：当 PE=0 时才能设置该寄存器。

19.5.9. I2C 寄存器映像

Offset	Register	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	I2C_CR1	SWRST	Res.	Res.	Res.	POS	ACK	STOP	START	NOS-TRETCH	ENGC	ENPEC	Res.	Res.	Res.	Res.	PE
	Reset value	0				0	0	0	0	0	0	0					0
0x04	I2C_CR2	Res.	Res.	Res.	Res.	Res.	IT-BUFEN	ITEV-TEN	ITER-REN	Res.	Res.	FREQ[5:0]					
	Reset value						0	0	0			0	0	0	0	0	0
0x08	I2C_OAR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADD[7:1]							Res.
	Reset value									0	0	0	0	0	0	0	
0x10	I2C_DR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DR[7:0]							
	Reset value									0	0	0	0	0	0	0	0
0x14	I2C_SR1	Res.	Res.	Res.	PECERR	OVR	AF	ARLO	BERR	TXE	RXNE	Res.	STOPF	Res.	BTF	ADDR	SB
	Reset value				0	0	0	0	0	0	0		0		0		0

Offset	Register	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x18	I2C_SR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GEN-CALL	Res.	TRA	BUSY	MSL
	Reset value												0		0	0	0
0x1C	I2C_CCR	F/S	DUTY	Res.	Res.	CCR[11:0]											
	Reset value	0	0			0	0	0	0	0	0	0	0	0	0	0	0
0x20	I2C_TRISE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRISE[5:0]					
												0	0	0	0	1	0

20. 通用同步异步收发器 (USART)

20.1. 介绍

通用同步异步收发器(USART)提供了一种灵活的方法与使用工业标准NRZ异步串行数据格式的外部设备之间进行全双工数据交换。USART利用分数波特率发生器提供宽范围的波特率选择。

它支持同步单向通信和半双工单线通信。它还允许多处理器通信。

20.2. USART 主要特性

- 全双工异步通信
- NRZ 标准格式
- 可配置 16 倍或者 8 倍过采样，增加在速度和时钟容忍度的灵活性
- 发送和接收共用的可编程波特率
- 自动波特率检测
- 可编程的数据长度 8 位或者 9 位
- 可配置的停止位（1 或者 2 位）
- 同步模式和为同步通讯的时钟输出功能
- 单线半双工通讯
- 独立的发送和接收使能位
- 硬件流控制
- 检测标志
  - 接收 buffer 满
  - 发送 buffer 空
  - 传输结束
- 奇偶校验控制
  - 发送校验位
  - 对接收数据进行校验
- 带标志的中断源
  - CTS 改变
  - 发送寄存器空
  - 发送完成
  - 接收数据寄存器满
  - 检测到总线空闲
  - 溢出错误
  - 帧错误
  - 噪音操作
  - 检测错误
- 多处理器通信
  - 如果地址不匹配，则进入静默模式
- 从静默模式唤醒：通过空闲检测和地址标志检测

20.3. USART 功能描述

USART 接口通过三个引脚与其他设备连接在一起。任何 USART 双向通信至少需要两个脚：接收数据输入(RX)和发送数据输出(TX)。

RX：接收数据串行输。通过过采样技术来区别数据和噪音，从而恢复数据。

TX：发送数据输出。当发送器被禁止时，输出引脚恢复到它的 I/O 端口配置。当发送器被激活，并且不发送数据时，TX 引脚处于高电平。在单线模式里，此 I/O 口被同时用于数据的发送和接收。

- 总线在发送或接收前应处于空闲状态
- 一个起始位
- 一个数据字(8 或 9 位)，最低有效位在前
- 1、2 个的停止位，由此表明数据帧的结束
- 使用分数波特率发生器：12 位整数和 4 位小数的表示方法
- 一个状态寄存器(USART_SR)
- 数据寄存器(USART_DR)
- 一个波特率寄存器(USART_BRR)，12 位的整数和 4 位小数

在同步模式中需要下列引脚：

CK：发送器时钟输出。

此引脚输出用于同步传输的时钟，(在 Start 位和 Stop 位上没有时钟脉冲，软件可选地，可以在最后一个数据位送出一个时钟脉冲)。数据可以在 RX 上同步被接收。这可以用来控制带有移位寄存器的外部设备(例如 LCD 驱动器)。时钟相位和极性都是软件可编程的。

下列引脚在硬件流控模式中需要：

- **nCTS:** 清除发送，若是高电平，在当前数据传输结束时阻断下一次的数据发送。
- **nRTS:** 发送请求，若是低电平，表明 USART 准备好接收数据。

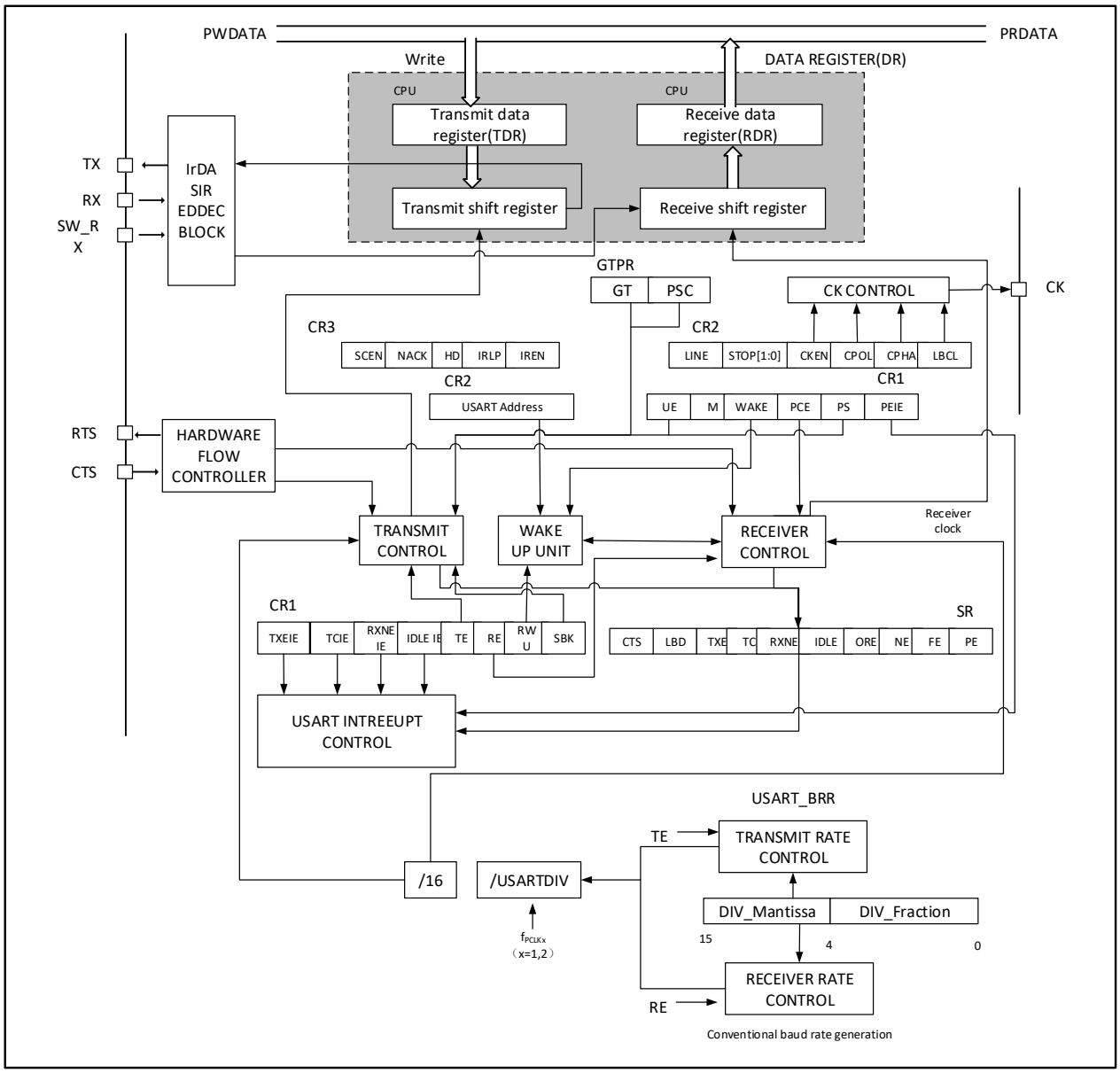


图 20-1 USART 框图

20.3.1. USART 特征描述

字长可以通过编程 USART_CR1 寄存器中的 M 位，选择成 8 或 9 位。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')。在断开符号结束时，发送器再插入 1 或 2 个停止位('1')来应答起始位。

发送和接收由一共用的波特率发生器驱动，当发送器和接收器的使能位分别置位时，分别为其产生时钟。

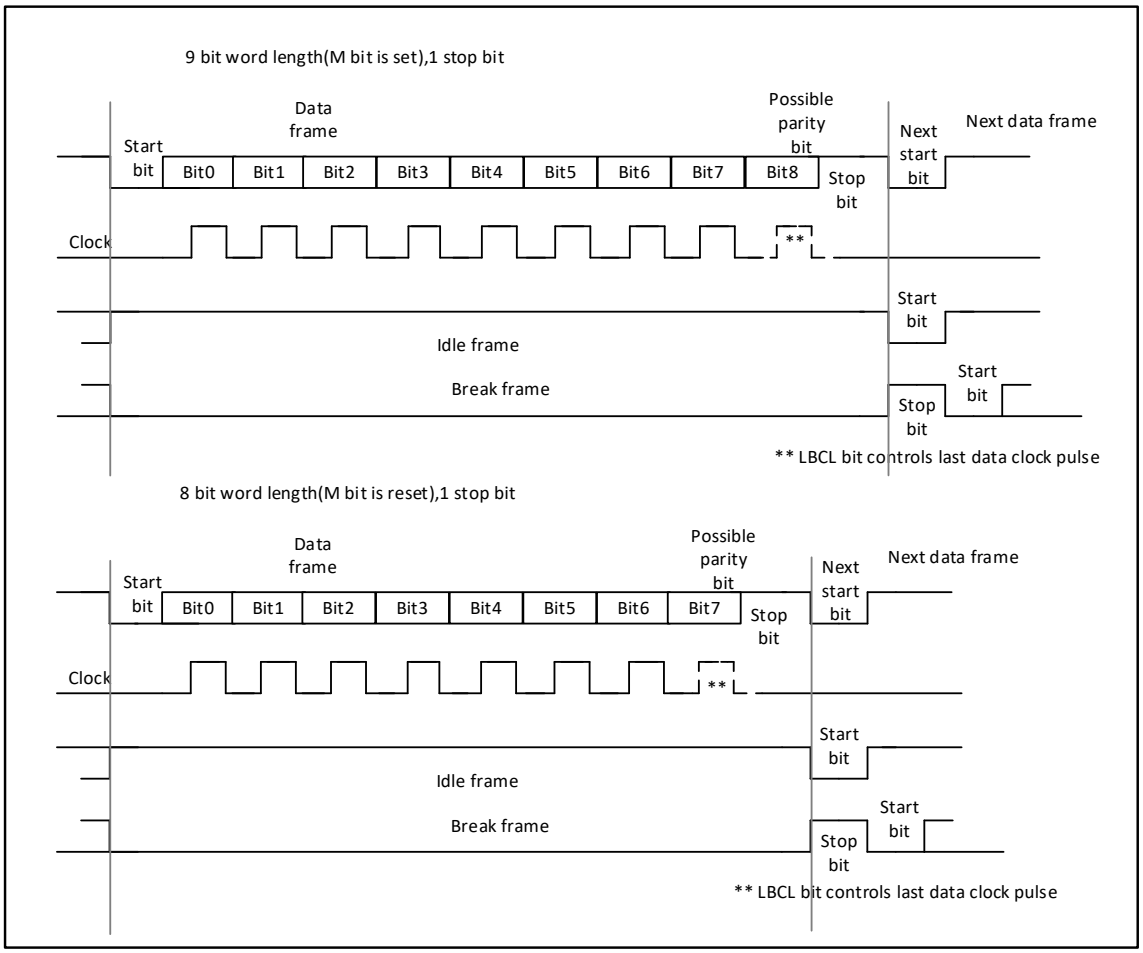


图 20-2 字长设置

20.3.2. 发送器

发送器根据 M 位的状态发送 8 位或 9 位的数据字。当发送使能位(TE)被设置时，发送移位寄存器中的数据在 TX 脚上输出，相应的时钟脉冲在 CK 脚上输出。

20.3.2.1. 字符发送

在 USART 发送期间，在 TX 引脚上首先移出数据的最低有效位。在此模式 USART_DR 寄存器包含了一个内部总线和发送移位寄存器之间的缓冲器。

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。USART 支持多种停止位的配置：1 和 2 个停止位。

注：

在数据传输期间不能复位 *TE* 位，否则将破坏 *TX* 脚上的数据，因为波特率计数器停止计数。正在传输的当前数据将丢失。

TE 位被激活后将发送一个空闲帧。

20.3.2.2. 可配置的停止位

随每个字符发送的停止位的位数可以通过控制寄存器 2 的位 13、12 进行编程。

1) 1 个停止位：停止位位数的默认值。

2) 2 个停止位：可用于常规 USART 模式、单线模式以及调制解调器模式。

空闲帧包括了停止位。

断开帧是 10 位低电平，后跟停止位(当 m=0 时)；或者 11 位低电平，后跟停止位(m=1 时)。不可能传输更长的断开帧(长度大于 10 或者 11 位)。

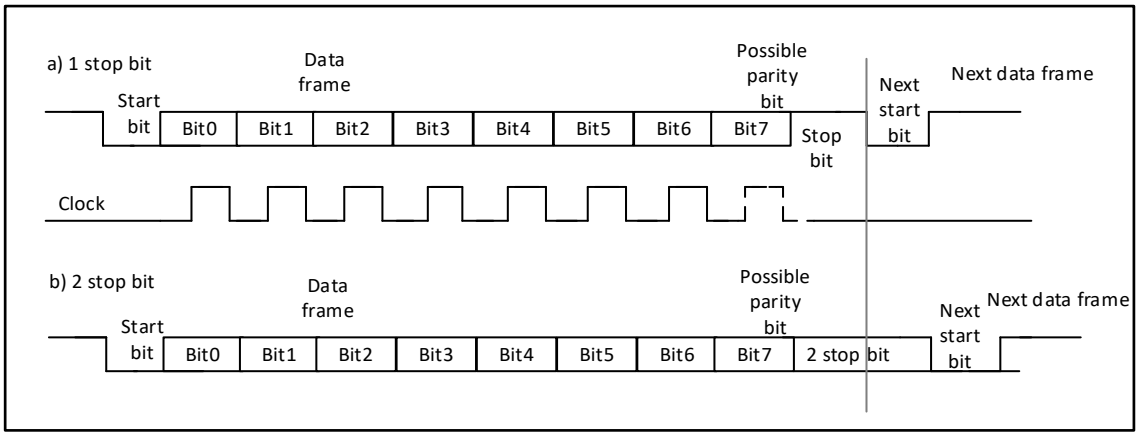


图 20-3 配置停止位

配置步骤：

1) 通过在 USART_CR1 寄存器上置位 UE 位来激活 USART

2) 编程 USART_CR1 的 M 位来定义字长。

3) 在 USART_CR2 中编程停止位的位数。

4) 利用 USART_BRR 寄存器选择要求的波特率。

5) 设置 USART_CR1 中的 TE 位，发送一个空闲帧作为第一次数据发送。

6) 把要发送的数据写进 USART_DR 寄存器(此动作清除 TXE 位)。在只有一个缓冲器的情况下，对每个待发送的数据重复步骤 7。

7) 在 USART_DR 寄存器中写入最后一个数据字后，要等待 TC=1，它表示最后一个数据帧的传输结束。当需要关闭 USART 或需要进入停机模式之前，需要确认传输结束，避免破坏最后一次传输。

20.3.2.3. 单字节通信

清零 TXE 位总是通过对数据寄存器的写操作来完成的。TXE 位由硬件来设置，它表明：

- 数据已经从 TDR 移送到移位寄存器，数据发送已经开始
- TDR 寄存器被清空
- 下一个数据可以被写进 USART_DR 寄存器而不会覆盖先前的数据

如果 TXEIE 位被设置，此标志将产生一个中断。

如果此时 USART 正在发送数据，对 USART_DR 寄存器的写操作把数据存进 TDR 寄存器，并在当前传输结束时把该数据复制进移位寄存器。

如果此时 USART 没有在发送数据，处于空闲状态，对 USART_DR 寄存器的写操作直接把数据放进移位寄存器，数据传输开始，TXE 位立即被置起。

当一帧发送完成时(停止位发送后)并且设置了 TXE 位，TC 位被置起，如果 USART_CR1 寄存器中的 TCIE 位被置起时，则会产生中断。

在 USART_DR 寄存器中写入了最后一个数据字后，在关闭 USART 模块之前或设置微控制器进入低功耗模式(详见下图)之前，必须先等待 TC=1。

使用下列软件过程清除 TC 位：

1．读一次 USART_SR 寄存器；

2．写一次 USART_DR 寄存器。

注：*TC* 位也可以通过软件对它写*'0'*来清除。此清零方式只推荐在多缓冲器通信模式下使用。

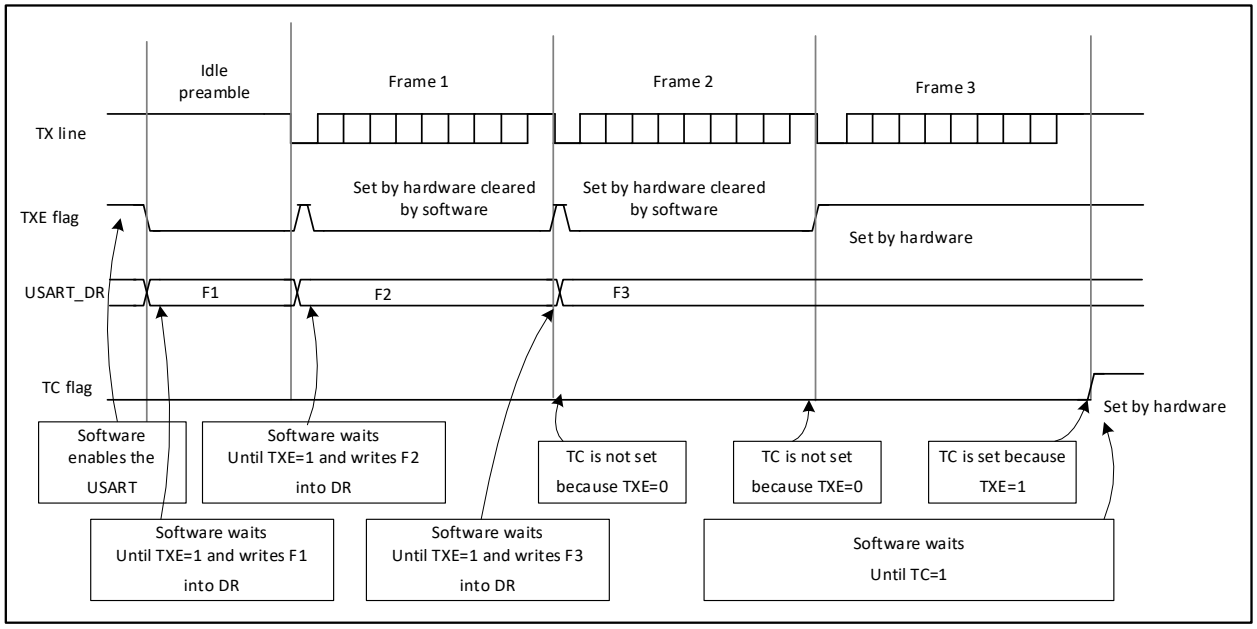


图 20-4 传输时 TC/TXE 的状态

20.3.2.4. 断开符号

设置 SBK 可发送一个断开符号。断开帧长度取决 M 位。如果设置 SBK=1，在完成当前数据发送后，将在 TX 线上发送一个断开符号。断开字符发送完成时(在断开符号的停止位时)SBK 被硬件复位。USART 在最后一个断开帧的结束处插入一逻辑'1'，以保证能识别下一帧的起始位。

注意：如果在开始发送断开帧之前，软件又复位了 SBK 位，断开符号将不被发送。如果要发送两个连续的断开帧， SBK 位应该在前一个断开符号的停止位之后置起。

20.3.2.5. 空闲符号

置位 TE 将使得 USART 在第一个数据帧前发送一空闲帧。

20.3.3. 接收器

USART 可以根据 USART_CR1 的 M 位接收 8 位或 9 位的数据字。

20.3.3.1. 开始位检测

在 USART 中，如果辨认出一个特殊的采样序列，那么就认为侦测到一个起始位。该序列为： 1 1 1 0 X 0 X 0 X 0 0 0 0

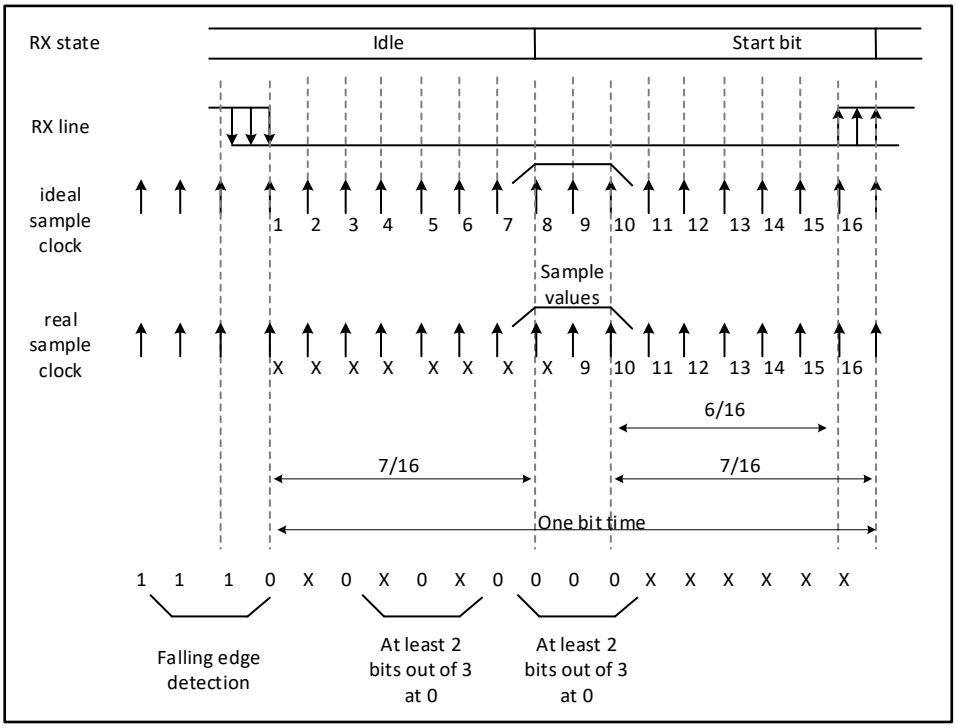


图 20-5 开始位检测

如果该序列不完整，那么接收端将退出起始位侦测并回到空闲状态*(*不设置标志位*)*等待下降沿。如果 *3* 个采样点都为*'0'(*在第 *3*、*5*、*7* 位的第一次采样，和在第 *8*、*9*、*10* 的第二次采样都为*'0')*，则确认收到起始位，这时设置 *RXNE* 标志位，如果 *RXNEIE=1*，则产生中断。

如果两次 *3* 个采样点上仅有 *2* 个是*'0'(*第 *3*、*5*、*7* 位的采样点和第 *8*、*9*、*10* 位的采样点*)*，那么起始位仍然是有效的，但是会设置 *NE* 噪声标志位。如果不能满足这个条件，则中止起始位的侦测过程，接收器会回到空闲状态*(*不设置标志位*)*。

如果有一次 *3* 个采样点上仅有 *2* 个是*'0'(*第 *3*、*5*、*7* 位的采样点或第 *8*、*9*、*10* 位的采样点*)*，那么起始位仍然是有效的，但是会设置 *NE* 噪声标志位。

20.3.3.2. 字符接收

在 USART 接收期间，数据的最低有效位首先从 RX 脚移进。在此模式里， USART_DR 寄存器包含的缓冲器位于内部总线和接收移位寄存器之间。

配置步骤：

1. 将 USART_CR1 寄存器的 UE 置 1 来激活 USART。
2. 编程 USART_CR1 的 M 位定义字长
3. 在 USART_CR2 中编写停止位的个数
4. 利用波特率寄存器 USART_BRR 选择希望的波特率。
5. 设置 USART_CR1 的 RE 位。激活接收器，使它开始寻找起始位。

当一字符被接收到时：

● RXNE 位被置位。它表明移位寄存器的内容被转移到 RDR。换句话说，数据已经被接收并且可以被读出(包括与之有关的错误标志)。

● 如果 RXNEIE 位被设置，产生中断。

● 在接收期间如果检测到帧错误，噪音或溢出错误，错误标志将被置起

●

● 在单缓冲器模式里，由软件读 USART_DR 寄存器完成对 RXNE 位清除。 RXNE 标志也可以通过对它写 0 来清除。RXNE 位必须在下一字符接收结束前被清零，以避免溢出错误。

注意： 在接收数据时，*RE* 位不应该被复位。如果 *RE* 位在接收时被清零，当前字节的接收被丢失。

20.3.3.3. 断开符号

当接收到一个断开帧时，USART 像处理帧错误一样处理它。

20.3.3.4. 空闲符号

当一空闲帧被检测到时，其处理步骤和接收到普通数据帧一样，但如果 IDLEIE 位被设置将产生一个中断。

20.3.3.5. 溢出错误

如果 RXNE 还没有被复位，又接收到一个字符，则发生溢出错误。数据只有当 RXNE 位被清零后才能从移位寄存器转移到 RDR 寄存器。 RXNE 标记是接收到每个字节后被置位的。如果下一个数据已被收到， RXNE 标志仍是置起的，溢出错误产生。

当溢出错误产生时：

● ORE 位被置位。

● RDR 内容将不会丢失。读 USART_DR 寄存器仍能得到先前的数据。

● 移位寄存器中以前的内容将被覆盖。随后接收到的数据都将丢失。

● 如果 RXNEIE 位被设置，中断产生。

● 顺序执行对 USART_SR 和 USART_DR 寄存器的读操作，可复位 ORE 位

注意：当 *ORE* 位置位时，表明至少有 *1* 个数据已经丢失。有两种可能性：

● 如果 RXNE=1，上一个有效数据还在接收寄存器 RDR 上，可以被读出。

● 如果 RXNE=0，这意味着上一个有效数据已经被读走，RDR 已经没有东西可读。当上一个有效数据在 RDR 中被读取的同时又接收到新的(也就是丢失的)数据时，此种情况可能发生。在读序列期间(在 USART_SR 寄存器读访问和 USART_DR 读访问之间)接收到新的数据，此种情况也可能发生。

20.3.3.6. 噪音错误

使用过采样技术(同步模式除外)，通过区别有效输入数据和噪音来进行数据恢复。

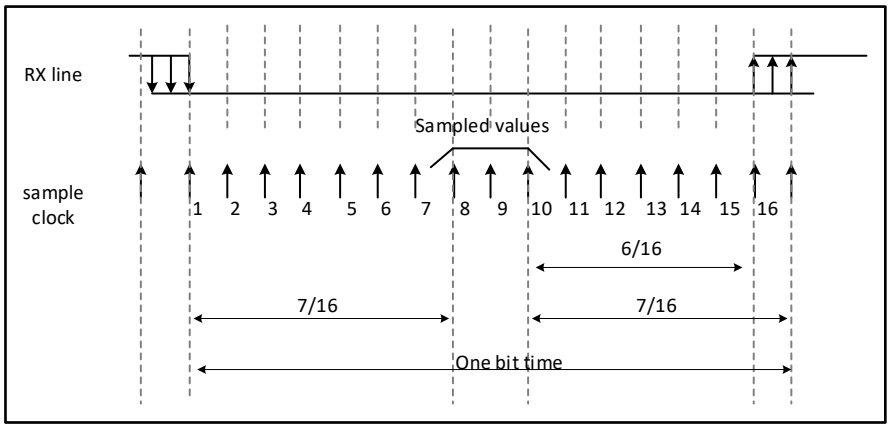


图 20-6 检测噪声的数据采样

表 20-1 检测噪声的数据采样

采样值	NE 状态	接收的位值	数据有效性
000	0	0	Valid
001	1	0	Not Valid
010	1	0	Not Valid
011	1	1	Not Valid
100	1	0	Not Valid
101	1	1	Not Valid
110	1	1	Not Valid
111	0	1	Valid

当在接收帧中检测到噪音时：

- 在 RXNE 位的上升沿设置 NE 标志。
- 无效数据从移位寄存器传送到 USART_DR 寄存器。
- 在单个字节通信情况下，没有中断产生。然而，因为 NE 标志位和 RXNE 标志位是同时被设置， RXNE 将产生中断。在多缓冲器通信情况下，如果已经设置了 USART_CR3 寄存器中 EIE 位，将产生一个中断。先读出 USART_SR，再读出 USART_DR 寄存器，将清除 NE 标志位。

20.3.3.7. 帧错误

当以下情况发生时检测到帧错误：

由于没有同步上或大量噪音的原因，停止位没有在预期的时间上接和收识别出来。

当帧错误被检测到时：

- FE 位被硬件置起
- 无效数据从移位寄存器传送到 USART_DR 寄存器。
- 在单字节通信时，没有中断产生。然而，这个位和 RXNE 位同时置起，后者将产生中断。在多缓冲器通信情况下，如果 USART_CR3 寄存器中 EIE 位被置位的话，将产生中断。

顺序执行对 USART_SR 和 USART_DR 寄存器的读操作，可复位 FE 位。

20.3.3.8. 接收期间的可配置的停止位

接收期间的可配置的停止位被接收的停止位的个数可以通过控制寄存器 2 的控制位来配置，在正常模式时，可以是 1 或 2 个。

- 1 个停止位：对 1 个停止位的采样在第 8，第 9 和第 10 采样点上进行。
- 2 个停止位：对 2 个停止位的采样是在第一停止位的第 8，第 9 和第 10 个采样点完成的。如果第一个停止位期间检测到一个帧错误，帧错误标志将被设置。第二个停止位不再检查帧错误。在第一个停止位结束时 RXNE 标志将被设置。

20.3.4. 分数波特率的产生

分数波特率的产生接收器和发送器的波特率在 USARTDIV 的整数和小数寄存器中的值应设置成相同。

Tx / Rx 波特率＝*fCK* /(16**USARTDIV*)

这里的 *fCK* 是给外设的时钟 USARTDIV 是一个无符号的定点数。这 12 位的值设置在 USART_BRR 寄存器。

注：在写入 *USART_BRR* 之后，波特率计数器会被波特率寄存器的新值替换。因此，不要在通信进行中改变波特率寄存器的数值。

如何从 USART_BRR 寄存器值得到 USARTDIV

例 1：

如果 DIV_Mantissa = 27，DIV_Fraction = 12 (USART_BRR=0x1BC),

则：

Mantissa (USARTDIV) = 27

Fraction (USARTDIV) = 12/16 = 0.75

所以 USARTDIV = 27.75

例 2：

要求 USARTDIV = 25.62,

就有：

DIV_Fraction = 16*0.62 = 9.92

最接近的整数是： 10 = 0x0A

DIV_Mantissa = mantissa (25.620) = 25 = 0x19

于是， USART_BRR = 0x19A

例 3：

要求 USARTDIV = 50.99

就有：

DIV_Fraction = 16*0.99 = 15.84

最接近的整数是：16 = 0x10 => DIV_frac[3:0]溢出 => 进位必须加到小数部分

DIV_Mantissa = mantissa (50.990 + 进位) = 51 = 0x33

于是：USART_BRR = 0x330，USARTDIV=51

波特率		F_{PCLK}=36MHz			F_{PCLK}=72MHz		
序号	Kbps	实际	置于波特率寄存器中的值	误差（%）	实际	置于波特率寄存器中的值	误差（%）
1	2.4	2.400	937.5	0%	2.4	1875	0%
2	9.6	9.600	234.375	0%	9.6	468.75	0%
3	19.2	19.2	117.1875	0%	19.2	234.375	0%
4	57.6	57.6	39.0625	0%	57.6	78.125	0%
5	115.2	115.384	19.5	0.15%	115.2	39.0625	0%
6	230.4	230.769	9.75	0.16%	230.769	19.5	0.16%
7	460.8	461.538	4.875	0.16%	461.538	9.75	0.16%
8	921.6	923.076	2.4375	0.16%	923.076	4.875	0.16%
9	2250	2250	1	0%	2250	2	0%
10	4500	不可能	不可能	不可能	4500	1	0%

注： *CPU* 的时钟频率越低，则某一特定波特率的误差也越低。可以达到的波特率上限可以由这组数据得到。

20.3.5. USART 接收器容忍度

只有当整体的时钟系统地变化小于 USART 异步接收器能够容忍的范围，USART 异步接收器才能正常地工作。影响这些变化的因素有：

- DTRA：由于发送器误差而产生的变化(包括发送器端振荡器的变化)
- DQUANT：接收器端波特率取整所产生的误差
- DREC：接收器端振荡器的变化
- DTCL：由于传输线路产生的变化(通常是由于收发器在由低变高的转换时序，与由高变低转换时序之间的不一致性所造成)。

需要满足： DTRA + DQUANT + DREC + DTCL < USART 接收器的容忍度。

对于正常接收数据，USART 接收器的容忍度等于最大能容忍的变化，它依赖于下述选择：

- 由 USART_CR1 寄存器的 M 位定义的 10 或 11 位字符长度
- 是否使用分数波特率产生

Table 20-1 Tolerance of the USART receiver when DIV_Fraction is 0

M bit	OVR8=0		OVR8=1	
	Consider NF an error	No think NF is an error	Consider NF an error	No think NF is an error
0	1.81%	1.75%	2.04%	2.06%
1	1.81%	1.75%	1.84%	1.81%

Table 20-2 Tolerance of the USART receiver when DIV_Fraction is different from 0

M bit	OVR8=0		OVR8=1	
	Consider NF an error	No think NF is an error	Consider NF an error	No think NF is an error
0	1.81%	1.75%	1.91%	1.81%
1	1.79%	1.75%	1.66%	1.64%

20.3.6. USART 自动波特率检测

USART 能够基于一个字符的接收，检测并自动设定 USARTx_BRR 寄存器的值。自动波特率检测在以下场景是有用处的：

1） 系统通讯速率提前未确认

2） 系统正在使用相对低精度的时钟源，该机制允许在不测量时钟偏差的情况下，获得正确的波特率

时钟源频率必须与被期望的通讯速率兼容。（必须选择 16 倍过采样，并从 fCK/65535 和 fCK/16 之间选择）

在激活自动波特率检测之前，自动波特率检测模式必须被选择（通过 USARTx_CR3 寄存器的 ABRMOD[1:0]位被选择）。基于不同的字符模式，有各种模式。

在这些自动波特率模式下，波特率在同步数据接收期间会被测量几次，每次测量都会跟之前进行对比。

这些模式是：

MODE 0: 任何以 1 开始的字符。在这种情况下，USART 测量起始位的宽度（下降沿到上升沿）

MODE 1: 任何以 10xx 位开始的字符。在这种情况下，USART 测量起始位和第一个数据位的宽度。该测量在下降沿到下降沿之间进行，以确保在慢速信号上升情况下更好的精度。

另一个对每个 RX 的 transition 的检查也会并行进行。如果 RX 上的 transition 没有与接收器充分的同步（接收器基于 bit 0 计算的波特率），则会产生错误。

在激活自动波特率检测之前，USARTx_BRR 寄存器必须通过写一个 non-zero 波特率值进行初始化。

通过置位 USARTx_CR3 寄存器的 ABREN 位，可以激活自动波特率检测功能。USART 将等待 RX 上的第一个字符。置位 USARTx_ISR 寄存器的 ABRF 标志，显示了自动波特率检测的完成。如果通讯线上是有噪声的，则自动波特率检测的正确进行不能被保证。在这种情况下，BRR 的值可能被破坏，ABRE 错误标志位将被置位。如果通讯速度与自动波特率检测范围不兼容（位宽不在 16 和 65535 个时钟周期之间@16 位过采样），ABRE 错误也会发生。

RXNE 中断将显示了操作的完成。在以后的任何时刻，自动波特率检测可能通过复位 ABRF 标志（通过写 0）再次启动。

Note：如果在自动波特期间，清零 UE，BRR 值可能被破坏。

20.3.7. 多处理器通信

通过 USART 可以实现多处理器通信(将几个 USART 连在一个网络里)。例如某个 USART 设备可以是主，它的 TX 输出和其他 USART 从设备的 RX 输入相连接；USART 从设备各自的 TX 输出逻辑地与在一起，并且和主设备的 RX 输入相连接。

在多处理器配置中，我们通常希望只有被寻址的接收者才被激活，来接收随后的数据，这样就可以减少由未被寻址的接收器的参与带来的多余的 USART 服务开销。

未被寻址的设备可启用其静默功能置于静默模式。在静默模式里：

- 任何接收状态位都不会被设置。
- 所有接收中断被禁止。
- USARTx_CR1 寄存器中的 RWU 位被置 1。RWU 可以被硬件自动控制或在某个条件下由软件写入。

根据 USARTx_CR1 寄存器中的 WAKE 位状态，USARTx 可以用二种方法进入或退出静默模式。

- 如果 WAKE 位被复位：进行空闲总线检测。
- 如果 WAKE 位被设置：进行地址标记检测。

20.3.7.1. 空闲总线检测(WAKE=0)

当 RWU 位被写 1 时， USART 进入静默模式。当检测到一空闲帧时，它被唤醒。然后 RWU 被硬件清零，但是 USART_SR 寄存器中的 IDLE 位并不置起。 RWU 还可以被软件写 0。下图给出利用空闲总线检测来唤醒和进入静默模式的一个例子

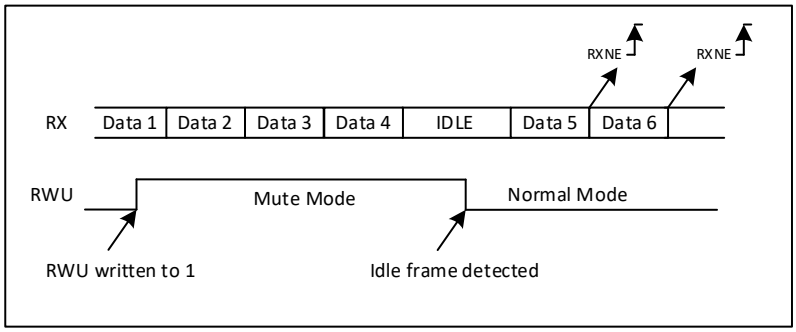


图 20-7 利用空闲总线检测的静默模式

20.3.7.2. 地址标记（Address mark）检测 (WAKE=1)

在这个模式里，如果 MSB 是 1，该字节被认为是地址，否则被认为是数据。在一个地址字节中，目标接收器的地址被放在 4 个 LSB 中。这个 4 位地址被接收器同它自己地址做比较，接收器的地址被编程在 USART_CR2 寄存器的 ADD。

如果接收到的字节与它的编程地址不匹配时，USART 进入静默模式。此时，硬件设置 RWU 位。

接收该字节既不会设置 RXNE 标志也不会产生中断请求，因为 USART 已经在静默模式。

当接收到的字节与接收器内编程地址匹配时，USART 退出静默模式。然后 RWU 位被清零，随后的字节被正常接收。收到这个匹配的地址字节时将设置 RXNE 位，因为 RWU 位已被清零。

当接收缓冲器不包含数据时(USART_SR 的 RXNE=0)，RWU 位可以被写 0 或 1。否则，该次写操作被忽略。下图给出利用地址标记检测来唤醒和进入静默模式的例子。

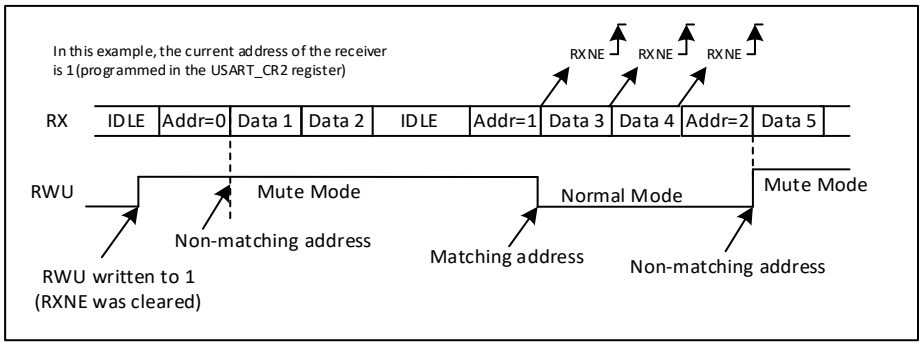


图 20-8 利用地址标记检测的静默模式

20.3.7.3. 校验控制

设置 USART_CR1 寄存器上的 PCE 位，可以使能奇偶控制(发送时生成一个奇偶位，接收时进行奇偶校验)。根据 M 位定义的帧长度，可能的 USART 帧格式列在下表中。

表 20-2 帧格式

M bit	PCE bit	USART fram
0	0	SB—8 bit data—STB
0	1	SB—7 bit data—PB—STB
1	0	SB—9 bit data—STB
1	1	SB—8 bit data—PB—STB

在用地址标记唤醒设备时，地址的匹配只考虑到数据的 *MSB* 位，而不用关心校验位。*(MSB* 是数据位中最后发出的，后面紧跟校验位或者停止位*)*

20.3.7.4. 偶校验

校验位使得一帧中的 7 或 8 个 LSB 数据以及校验位中'1'的个数为偶数。

例如：数据=00110101，有 4 个'1'，如果选择偶校验(在 USARTx_CR1 中的 PS＝0)，校验位将是'0'。

20.3.7.5. 奇校验

此校验位使得一帧中的 7 或 8 个 LSB 数据以及校验位中'1'的个数为奇数。

例如：数据=00110101，有 4 个'1'，如果选择奇校验(在 USARTx_CR1 中的 PS＝1)，校验位将是'1'。

20.3.7.6. 传输模式

如果 USARTx_CR1 的 PCE 位被置位，写进数据寄存器的数据的 MSB 位被校验位替换后发送出去(如果选择偶校验偶数个'1'，如果选择奇校验奇数个'1')。如果奇偶校验失败，USART_SR 寄存器中的 PE 标志被置'1'，并且如果 USART_CR1 寄存器的 PEIE 在被预先设置的话，中断产生。

20.3.8. USART 同步模式

通过写 USART_CR2 寄存器的 CLKEN 位为 1，选择同步模式。在同步模式里，下列位必须保持清零状态：

● USART_CR3 寄存器中的 HDSEL 位

USART 允许用户以主模式方式控制双向同步串行通信。CK 脚是 USART 发送器时钟的输出。在起始位和停止位期间，CK 脚上没有时钟脉冲。根据 USART_CR2 寄存器中 LBCL 位的状态，决定在最后一个有效数据位期间产生或不产生时钟脉冲。USART_CR2 寄存器的 CPOL 位允许用户选择时钟极性，USART_CR2 寄存器上的 CPHA 位允许用户选择外部时钟的相位。

在总线空闲期间，实际数据到来之前以及发送断开符号的时候，外部 CK 时钟不被激活。

同步模式时，USART 发送器和异步模式里工作一模一样。但是因为 CK 是与 TX 同步的(根据 CPOL 和 CPHA)，所以 TX 上的数据是随 CK 同步发出的。

同步模式的 USART 接收器工作方式与异步模式不同。如果 RE=1，数据在 CK 上采样(根据 CPOL 和 CPHA 决定在上升沿还是下降沿)，不需要任何的过采样。但必须考虑建立时间和持续时间(取决于波特率，1/16 位时间)。

注意：

CK 脚同 *TX* 脚一起联合工作。因而，只有在使能了发送器*(TE＝1)*，并且发送数据时*(*写入数据至 *USART_DR* 寄存器*)*才提供时钟。这意味着在没有发送数据时是不可能接收一个同步数据的。

LBCL,CPOL 和 *CPHA* 位的正确配置，应该在发送器和接收器都被禁止时；当使能了发送器或接收器时，这些位不能被改变。

建议在同一条指令中设置 *TE* 和 *RE*，以减少接收器的建立时间和保持时间。

USART 只支持主模式：它不能用来自其他设备的输入时钟接收或发送数据*(CK* 永远是输出*)*。

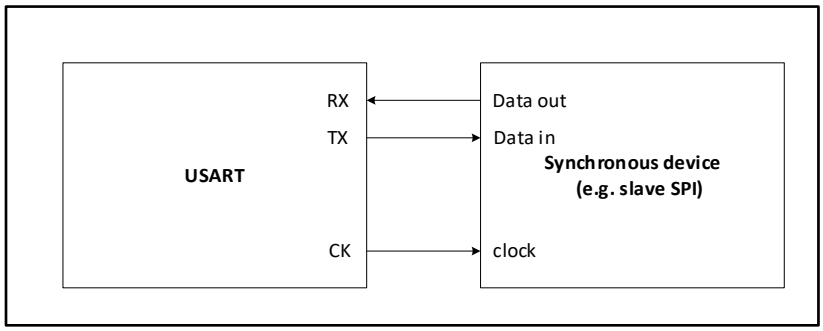


图 20-9 USART 同步传输的例子

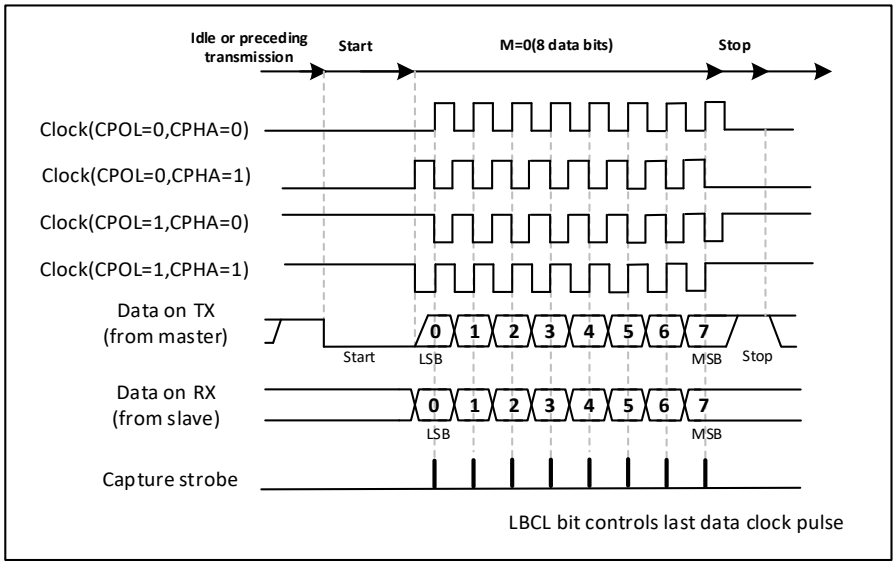


图 20-10 USART 数据时钟时序示例(M=0)

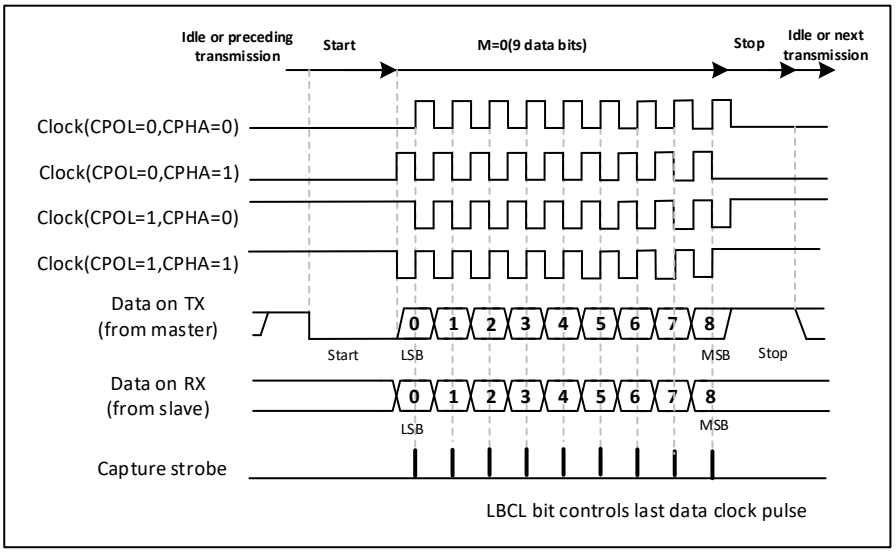


图 20-11 USART 数据时钟时序示例(M=1)

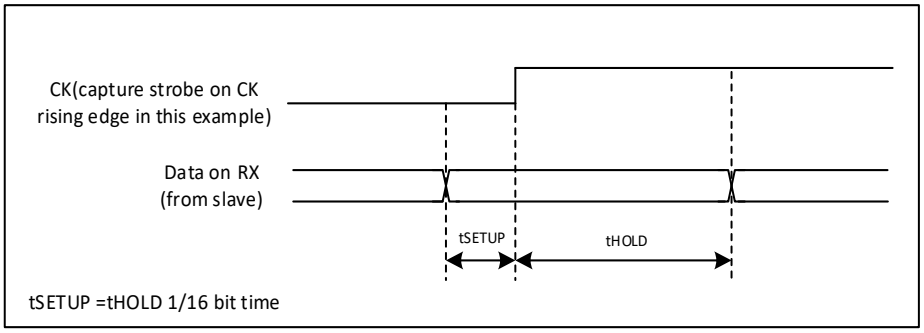


图 20-12 RX 数据采样/保持时间

20.3.9. 单线半双工通信

单线半双方模式通过设置 USARTx_CR3 寄存器的 HDSEL 位选择。在这个模式里，下面的位必须保持清零状态：

- USARTx_CR2 寄存器的 CLKEN 位

USART 可以配置成遵循单线半双工协议。在单线半双工模式下，TX 和 RX 引脚在芯片内部互连。使用控制位”HALF DUPLEX SEL”(USARTx_CR3 中的 HDSEL 位)选择半双工和全双工通信。

当 HDSEL 为’1’时

- RX 不再被使用
- 当没有数据传输时， TX 总是被释放。因此，它在空闲状态的或接收状态时表现为一个标准 I/O 口。这就意味该 I/O 在不被 USART 驱动时，必须配置成悬空输入(或开漏的输出高)。

除此以外，通信与正常 USART 模式类似。由软件来管理线上的冲突(例如通过使用一个中央仲裁器)。特别的是，发送从不会被硬件所阻碍。当 TE 位被设置时，只要数据一写到数据寄存器上，发送就继续。

20.3.10. 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。

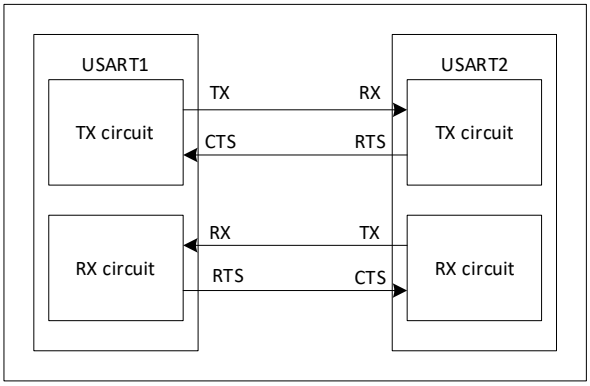


图 20-13 两个 USART 间的硬件流控制

20.3.10.1. RTS 流控制

如果 RTS 流控制被使能(RTSE=1)，只要 USART 接收器准备好接收新的数据，nRTS 就变成有效(接低电平)。当接收寄存器内有数据到达时，nRTS 被释放，由此表明希望在当前帧结束时停止数据传输。

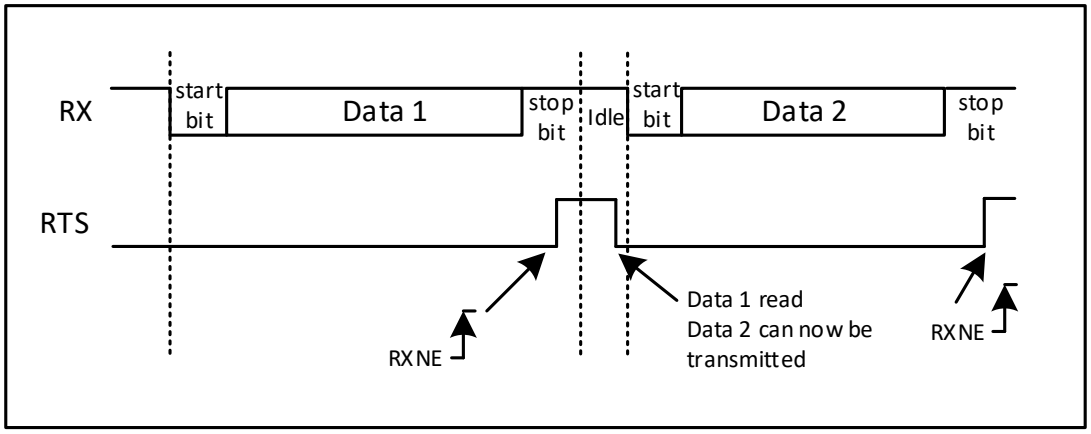


图 20-14 RTS 流控制

20.3.10.2. CTS 流控制

如果 CTS 流控制被使能(CTSE=1)，发送器在发送下一帧前检查 nCTS 输入。如果 nCTS 有效(被拉成低电平)，则下一个数据被发送(假设那个数据是准备发送的，也就是 TXE=0)，否则下一帧数据不被发出去。若 nCTS 在传输期间被变成无效，当前的传输完成后停止发送。

当 CTSE=1 时，只要 nCTS 输入一变换状态，硬件就自动设置 CTSIF 状态位。它表明接收器是否准备好进行通信。如果设置了 USART_CT3 寄存器的 CTSIE 位，则产生中断。

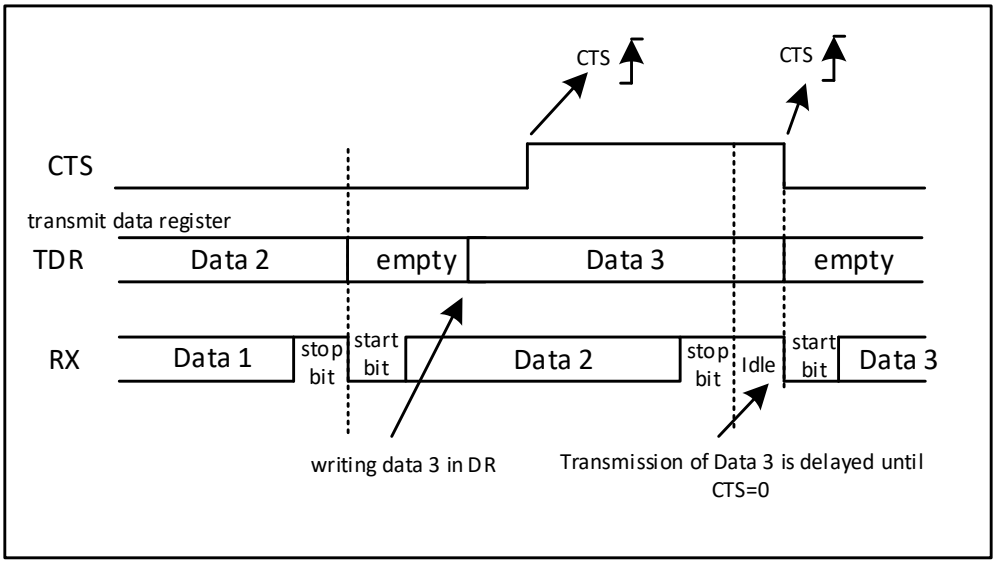


图 20-15 CTS 流控制

20.4. USART 中断请求

序号	中断事件	事件标志	使能位	发送/接收
1	发送数据寄存器空	TXE	TXEIE	发送
2	CTS（Clear to Send）中断	CTSIF	CTSIE	发送
3	传送完成	TC	TCIE	发送
4	接收寄存器非空（读数据准备好）	RXNE	RXNEIE	接收
5	Overrun 错误	ORE		接收
6	空闲帧	IDLE	IDLEIE	接收
7	奇偶校验错误	PE	PEIE	接收

8	多处理器通讯时，噪声、overrun 和帧错误	NR/ORE/FE	EIE	接收

所有 USART 中断共用同一个中断向量。

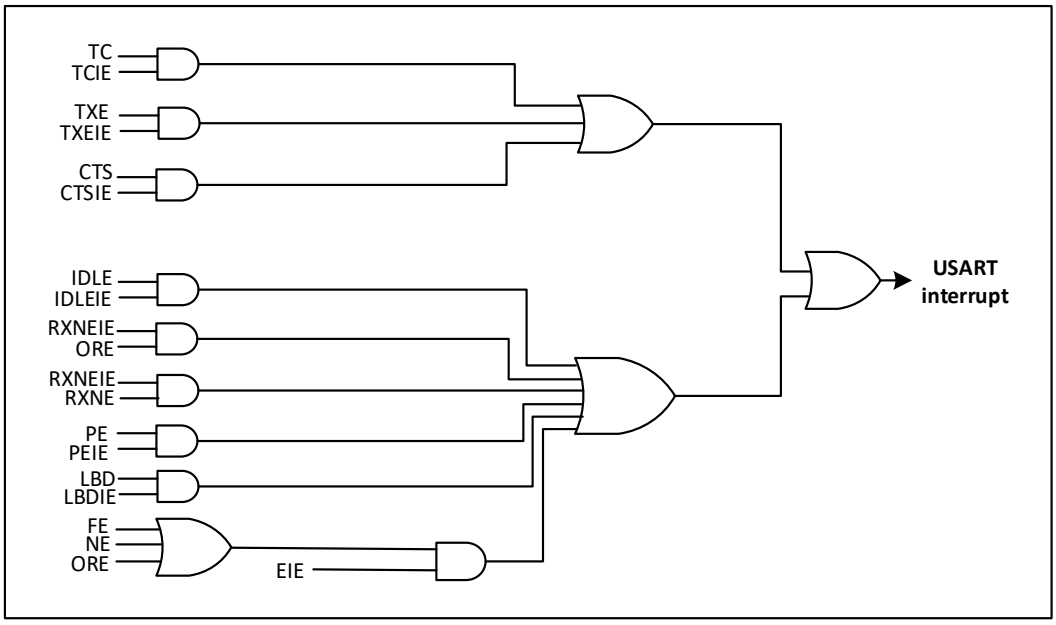


图 20-16 USART 中断映像图

20.5. USART 寄存器

20.5.1. 状态寄存器 (USART_SR)

Address offset:0x00

Reset value:0x0000 00C0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	ABRRQ	ABRE	ABRF	CTS	Res	TXE	TC	RXNE	IDLE	ORE	NE	FE	PE
			W	R	R	RC_W0		R	RC_W0	RC_W0	R	R	R	R	R

Bit	Name	R/W	Reset Value	Function
31:13	Reserved	RES	-	Reserved
12	ABRRQ	W	0	自动波特率请求。 该位写 1 会复位 ABRF 标志位，并且请求下一帧的自动波特率检测。
11	ABRE	R	0	自动波特率错误标志。 当自动波特率检测出错（波特率超出范围或者字符比较错误）时，硬件置位该寄存器。 软件通过写 1 到 ABRRQ 寄存器清零该位。
10	ABRF	R	0	自动波特率检测标志。 当自动波特率设置（同时设置 RXNE=1，当中断使能后产生中断），或者自动波特率检测操作出错（ABRE=1，RXNE=1，FE=1）时该位由硬件置 1。 软件通过写 1 到 USART_RQR 寄存器的 ABRRQ 位清零该位。
9	CTS	RC_W0	0	CTS 标志。

Bit	Name	R/W	Reset Value	Function
				当 CTS 输入 toggle，别 CTSE=1 时，该寄存器为 1.软件写 0 清零。当 CTSIE=1 时，产生 CTS 中断。 0：CTS line 值未改变 1：CTS line 值改变
8	reserved			
7	TXE	R	1	传输寄存器空标志。 当 USART_DR 寄存器数据传送到移位寄存器，硬件置位该寄存器。当 TXEIE=1 时，产生中断。 写 USART_DR 寄存器会清零该位。 0：数据未传送到移位寄存器 1：数据传送到移位寄存器
6	TC	RC_W0	1	传送完成标志。 传送数据帧完成后，且 TXE=1，则硬件置位该寄存器。TCIE=1 时产生中断。 软件先读 USART_SR 寄存器然后写 USART_DR 寄存器会清零该位（针对多处理器通讯）。软件同时可以写 0 清零。 0：传送未完成 1：传送完成
5	RXNE	RC_W0	0	读数据寄存器不空标志。 当移位寄存器值传送到 USART_DR 寄存器，硬件置位该寄存器。 软件读 USART_DR 寄存器、或者写 0 清零该位。 当 RXNEIE=1 时，产生中断。 0：未收到数据 1：接收数据准备好
4	IDLE	R	0	空闲标志。 检测 IDLE line，硬件置位该寄存器。当 IDLEIE=1 时产生中断。 软件先读 USART_SR 寄存器后读 USART_DR 寄存器可以清零该位。 0：未检测到 IDLE line 1：检测到 IDLE line
3	ORE	R	0	Overrun 错误标志。 当 RXNE=1 时，在移位寄存器中接收到的数据正准备转移到 RDR 寄存器时，硬件设置该位。 软件先读 USART_SR 寄存器后读 USART_DR 寄存器可以清零该位。 当 RXNEIE=1 时，产生中断。 0：未产生 Overrun 错误 1：产生 Overrun 错误 注：该寄存器置位时，RDR 寄存器内容不会丢失，但移位寄存器内容被覆盖。 当 EIE=1 时，产生 ORE 中断。
2	NE	R	0	噪声错误标志。 在数据帧接收到噪声时，硬件置位该寄存器。 软件先读 USART_SR 寄存器后读 USART_DR 寄存器可以清零该位。 0：未检测到噪声错误 1：检测到噪声错误 注：当 RXNE 与 NE 同时产生时，NE=1 时不产生中断，而在设置 RXNE 标志时产生中断。在多缓冲器通讯模式下，当 EIE=1 时 NE=1 会产生中断。
1	FE	R	0	帧错误标志。 当检测到不同步、过多的噪声或中止字符时，由硬件设置此位。 软件先读 USART_SR 寄存器后读 USART_DR 寄存器可以清零该位。

Bit	Name	R/W	Reset Value	Function
				0：未检测到帧错误 1：检测到帧错误或 break 字符 注：当 RXNE 与 FE 同时产生时，FE=1 时不产生中断，而在设置 RXNE 标志时产生中断。如果当前传输的数据既产生了帧错误，又产生了过载错误，硬件还是会继续该数据的传输，并且只设置 ORE 标志位。在多缓冲器通讯模式下，当 EIE=1 时 FE=1 会产生中断。
0	PE	R	0	校验值错误。 当接收时校验值错误时，硬件置位该寄存器。 软件先读 USART_SR 寄存器后读 USART_DR 寄存器可以清零该位。但软件在清该位前必须等待 RXNE=1. 当 PEIE 时，产生中断。 0：未产生奇偶校验错误 1：产生奇偶校验错误

20.5.2. 数据寄存器 (USART_DR)

Address offset: 0x04

Reset value: undefined

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

Bit	Name	R/W	Reset Value	Function
31：9	Reserved	RES	-	Reserved
8：0	DR[8:0]	RW	undefined	接收/发送数据寄存器。 取决于读还是写操作，前者是接收到的数据，后者是发送的数据。 DR 寄存器物理上由两个寄存器组成（一个是发送的 TDR，一个是接收的 RDR），所以 DR 寄存器实现了读和写的两个功能。 TDR 寄存器在内部总线和输出移位寄存器之间提供了并行的接口，RDR 寄存器在输入移位寄存器和内部总线之间提供了并行接口。 当奇偶校验使能打开进行发送操作时，写 MSB 位（bit7 或者 bit8）是无效的，因为已被校验位代替了。 当奇偶校验使能打开进行接收操作时，读出的 MSB 位是被接收到的校验位。

20.5.3. 波特率寄存器 (USART_BRR)

Address offset:0x08

Reset value:0x0000_0000

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

在自动波特率检测模式，硬件更新该寄存器。

Bit	Name	R/W	Reset Value	Function
31：16	Reserved	RES	-	Reserved
15：4	DIV_Mantissa[15:4]	RW	0	12bit 整数
3：0	DIV_Fraction[3:0]	RW	0	4bit 小数

20.5.4. 控制寄存器 1 (USART_CR1)

Address offset:0x0C

Reset value: 0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	UE	M	WAKE	PCE	PS	PEIE	TXEIE	TCIE	RXNEIE	IDLEIE	TE	RE	RWU	SBK
		RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31：14	Reserved	RES	-	Reserved
13	UE	RW	0	USART 使能。当该位清零后，USART 模块会立即停止当前操作。该位由软件置位和清零。 0：USART prescaler 和 output 禁止，low-power 模式 1：USART 使能 软件需要等待 USART_ISR.TC 置位后，才能清零 UE 位，进入低功耗模式；
12	M	RW	0	0：1 start bit，8 data bits，n stop bit 1：1 start bit，9 data bit，n stop bit
11	WAKE	RW	0	接收唤醒方式。 从 mute 模式唤醒方式。由软件置位或者清零。 0：Idle line 唤醒 1：地址唤醒
10	PCE	RW	0	奇偶校验控制。 0：奇偶校验禁止 1：奇偶校验使能 奇偶校验位：9bit 的第 9 位；8bit 的第 8 位。
9	PS	RW	0	奇偶校验选择。由软件置位和清零。 0：偶校验 1：奇校验
8	PEIE	RW	0	PE 中断使能。由软件置位和清零。 0：禁止 1：PE 中断使能
7	TXEIE	RW	0	TXE 中断使能。由软件置位和清零。 0：禁止 1：TXE 中断使能
6	TCIE	RW	0	传送结束中断使能。由软件置位和清零。 0：禁止 1：TC 中断使能
5	RXNEIE	RW	0	RXNE 中断使能；由软件置位和清零。 0：禁止 1：ORE 或者 RXNE 中断使能
4	IDLEIE	RW	0	IDLE 中断使能。由软件置位和清零。 0：禁止 1：IDLE 中断使能
3	TE	RW	0	传送使能。 0：传送禁止 1：传送使能
2	RE	RW	0	接收使能。 0：接收禁止

Bit	Name	R/W	Reset Value	Function
				1：接收使能，开始检测 start 位
1	RWU	RW	0	接收唤醒。 该位表明 USART 是否为 mute 模式。 当接收到 mute 模式序列，该寄存器置位；如果接收到唤醒序列，该寄存器清零。具体哪种唤醒序列（地址或者 IDLE）由寄存器 USART_CR1.WAKEbit 控制。 0：接收器为工作模式 1：接收器为静默模式 注 1：在设置该位进入 mute 模式前，USART 要已经先接收了一个数据字节，否则在 mute 模式下，不能被 idle 总线检测唤醒。 注 2：当配置成地址标记检测唤醒（WAKE=1），在 RXNE 被置位时，不能用软件修改 RWU 位。
0	SBK	RW	0	发送 break 帧。 软件置位该寄存器，发送 break 字节。Break 帧的 stop 位发送后，硬件清零该寄存器。 0：不发送 break 字节 1：发送 break 字节

20.5.5. 控制寄存器 2 (USART_CR2)

Address offset:0x10

Reset value: 0x0000_0000

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

Bit	Name	R/W	Reset Value	Function
31：14	Reserved	RES	-	Reserved
13	STOP	RW	0	Stop 位配置。 0：1 stop bit； 1：2 stop 位；
12	Reserved			
11	CLKEN	RW	0	CK pin 使能。 0：禁止； 1：CK pin 使能； 不支持同步模式时，该位保留。
10	CPOL	RW	0	时钟极性。 同步模式，CK pin 输出时钟极性。 0：传输窗外，CK pin 为稳定低值； 1：传输窗外，CK pin 为稳定高值；
9	CPHA	RW	0	该位在同步模式下用于选择 CK pin 输出时钟的相位。它与 CPOL 位一起工作，以产生所需的时钟/数据关系。 0：第一个时钟传输是首个数据捕获沿； 1：第二个时钟传输是首个数据捕获沿；
8	LBCL	RW	0	最后一位数据的时钟脉冲是否在 CK pin 输出。 0：最后一位数据的时钟脉冲不在 CK pin 输出； 1：最后一位数据的时钟脉冲在 CK pin 输出；
7:4	Reserved	RES	-	Reserved
3:0	ADD[3:0]	RW	4’b0	USART 地址。 该寄存器用于多处理器 mute 模式，用作 4bit 地址唤醒时的地址。

20.5.6. 控制寄存器 3 (USART_CR3)

Address offset:0x14

Reset value: 0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	ABR-MOD[1:0]		ABR EN	OVER8	CTSIE	CTSE	RTSE	Res	Res	Res	Res	HDSEL	Res	Res	EIE
	RW		RW	RW	RW	RW	RW					RW			RW

Bit	Name	R/W	Reset Value	Function
31：15	Reserved	RES	-	Reserved
14：13	ABRMOD[1:0]	RW	2’b0	自动波特率检测模式。 00：从 start 位开始测量波特率 01：下降沿到下降沿测量 10： Reserved 11： Reserved 当 ABREN=0 或者 UE=0 时，该寄存器只写。
12	ABREN	RW	0	自动波特率使能。 0：禁止 1：自动波特率使能
11	OVER8	RW	0	Oversampling 模式。 0：Oversampling by 16 1：Oversampling by 8 该位仅在 UE=0 时可被写。
10	CTSIE	RW	0	CTS 中断使能。 0：禁止； 1：CTSIF 中断使能；
9	CTSE	RW	0	CTS 使能。 0：CTS 硬件流控制禁止； 1：CTS 模式使能。当有当 CTS 输入为 0 时，才会传输数据。此时，当数据写入数据寄存器后，要等待 CTS 有效后才会启动传输。
8	RTSE	RW	0	RTS 使能。 0：RTS 硬件流控制禁止； 1：RTS 输出使能，只有当接收 buffer 未满时才会请求下一个数据。当前数据发送完成后，发送操作暂停。如果可以接收数据了，将 RTS 置为有效（0）。
7:4	reserved			
3	HDSEL	RW	0	半双工选择。 0：非半双工模式； 1：半双工模式选择；
2:1				
0	EIE	RW	0	错误中断使能。 0：禁止； 1：帧错误 FE、overrun 错误 ORE、噪声 NF 中断使能。

20.5.7. USART 寄存器映像

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	USART_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ABRRQ	ABRE	ABRF	CTS	Res.	TXE	TC	RXNE	IDLE	ORE	NE	FE	PE

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reset value																				0	0	0	0		1	1	0	0	0	0	0	0
0x04	USART_DR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DR[8:0]								
	Reset value																								0	0	0	0	0	0	0	0	0
0x08	USART_BRR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIV_Mantissa[11:0]												DIV_Faction[3:0]			
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	USART_CR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UE	M	WAKE	PCE	PS	PEIE	TXEIE	TCIE	RXNEIE	IDLEIE	TE	RE	RWU	SBK
	Reset value																			0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	USART_CR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STOP	Res.	CLKEN	CPOL	CPHA	LBCL	Res.	Res.	Res.	Res.	ADD[3:0]			
	Reset value																			0		0	0	0	0					0	0	0	0
0x14	USART_CR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ABR-MOD[1:0]		ABREN	OVER8	CTSIE	CTSE	RTSE	Res.	Res.	Res.	Res.	HDSEL	Res.	Res.	EIE
	Reset value																		0	0	0	0	0	0	0					0			0
0x18	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

21. 串行外设接口 (SPI)

本项目设计实现了 1 个 SPI 模块。

21.1. 简介

串行外设接口(SPI)允许芯片与外部设备以半双工、全双工、单工同步的串行方式通信。此接口可以被配置成主模式，并为外部从设备提供通信时钟(SCK)。接口还能以多主配置方式工作。

它可用于多种用途，包括使用一条双向数据线的双线单工同步传输。

21.2. SPI 主要特征

- 支持 SPI 主机模式和 SPI 从机模式
- 3 线全双工同步传输
- 2 线半双工同步传输（有双向数据线）
- 2 线单工同步传输（无双向数据线）
- 8 位或者 16 位传输帧选择
- 支持多主模式
- 8 个主模式波特率预分频系数（最大为 $f_{PCLK}/4$）
- 从模式频率（最大为 $f_{PCLK}/4$）
- 主模式和从模式下均可以由软件或硬件进行 NSS 管理：主/从操作模式的动态改变
- 可编程的时钟极性和相位
- 可编程的数据顺序，MSB 在前或 LSB 在前
- 可触发中断的专用发送和接收标志
- SPI 总线忙状态标志
- Motorola 模式
- 可引起中断的主模式故障、过载

21.3. SPI 功能描述

21.3.1. 概述

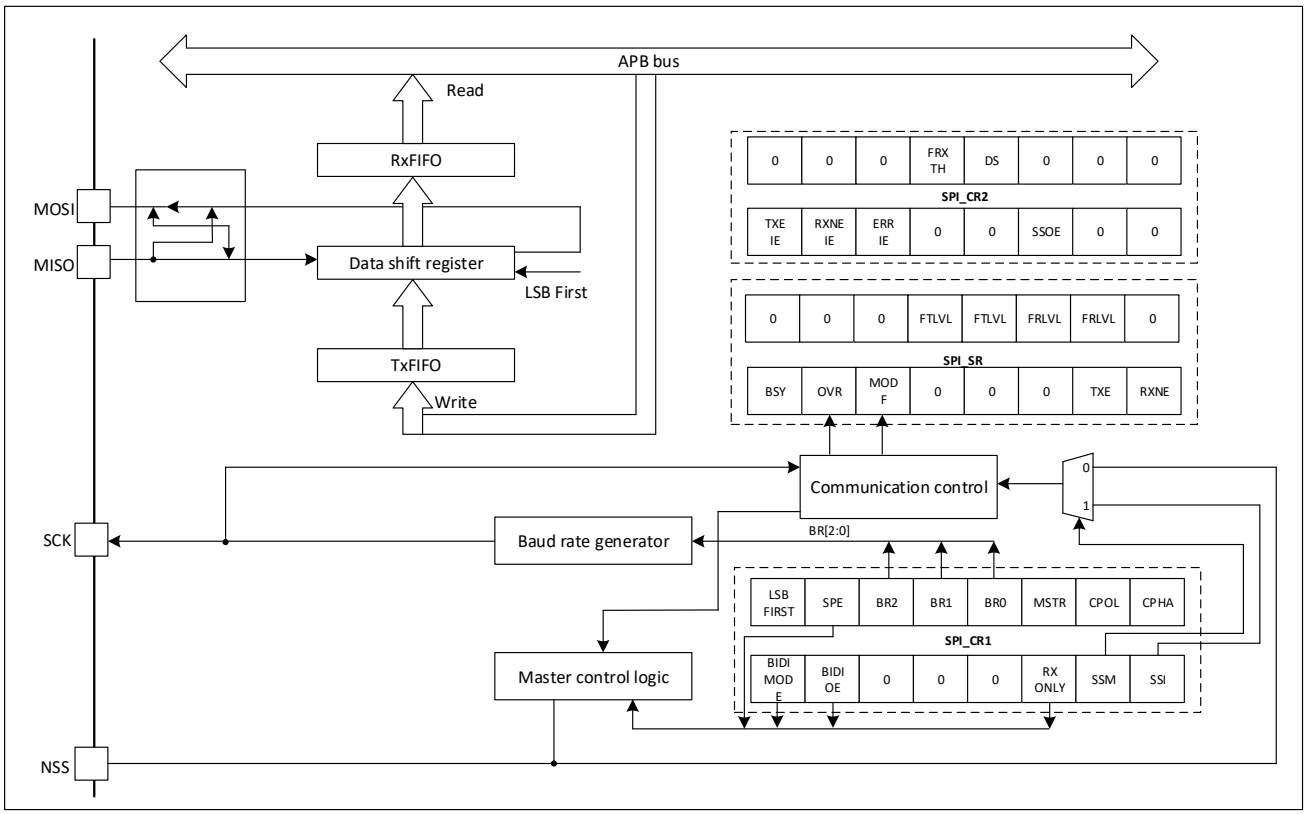


图 21-1 SPI 框图

SPI 通过 4 个引脚与外部器件相连：

MISO：主设备输入/从设备输出引脚。该引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/从设备输入引脚。该引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。取决于 SPI 和 NSS 的设定，该 pin 可以用作：

- 选择要通讯的从机
- 同步数据帧
- 发现多主机间的冲突

SPI 总线允许在一个主机和一个或者多个从机之间的通讯。总线由至少两根线组成：一个是时钟，另一个是被同步传输的数据。根据应用场景，可以选择增加另外一根数据线和从机 NSS 信号。

21.3.2. 单主机和单从机通信

针对不同的应用场景，SPI 可以使用几种不同的配置进行通讯。这些配置使用 2 线、3 线（软件 NSS management）或者 4 线（硬件 NSS management）。通讯通常都被主机启动。

21.3.2.1. 全双工通信

缺省情况，SPI 被配置成全双工通讯。在这种配置下，主机和从机的 shift 寄存器，在 MOSI 和 MISO 之间，使用两个单向的线连到一起。在 SPI 通讯期间，数据在主机提供的时钟沿同步的被移位。主机通过 MOSI 发送数据，从 MISO 接收来自从机的数据。当数据帧传输完成（所有 bit 被 shift 完成），在主机和从机之间的信息就被交互过了。

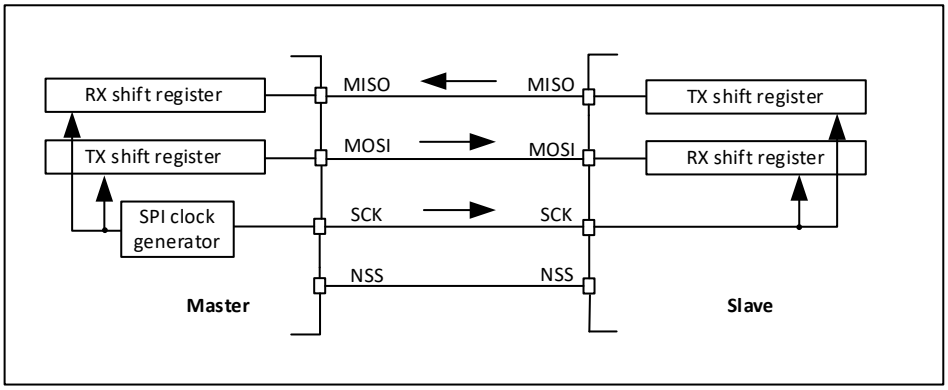


图 21-2 全双工单主机/单从机应用

21.3.2.2. 半双工通信

通过设定 BIDIMODE bit（SPI_CR1 寄存器），SPI 可以工作在半双工模式。在这种配置下，用 1 根数据线完成主机和从机 shift 寄存器的连接。在通讯过程中，在 SCK 的时钟沿，数据在两个 shift 寄存器之间以 BIDIOE（SPI_CR1 寄存器）选择的方向，同步移位。在该配置下，主机的 MISO 和从机的 MOSI 被释放作为通用端口给其他应用使用。

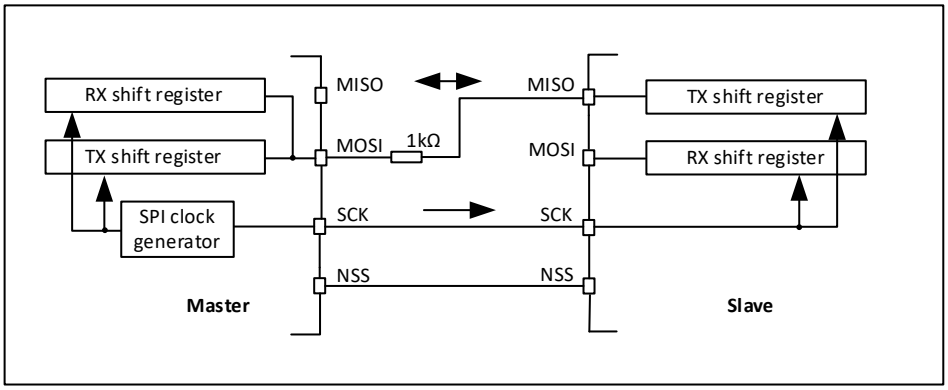


图 21-3 半双工单主机/单从机应用

NSS 可以被使用在主机和从机之间进行硬件控制流。可选的，NSS 也可以不使用。然后该流程就要内部处理。

21.3.2.3. 单工通信

通过使用 RXONLY（SPI_CR1 寄存器），设定 SPI 在只发送模式或者只接收模式，使 SPI 工作在单线模式下。在这个配置下，在主机和从机的 shift 寄存器之间只使用 1 根线。另一对 MISO 和 MOSI 不被使用，可以被释放成通用端口。

- 只发送模式（RXONLY=0）：配置与全双工相同。应用忽略在未使用的端口上的信息。这个端口可以被用作标准的 GPIO。
- 只接收模式（RXONLY=1）：通过置位 RXONLY，应用可以禁能 SPI 输出功能。在从机配置，MISO 输出被禁能，该端口被用作 GPIO。当他的从机 NSS 信号有效时，从机继续从 MOSI 接收数据。接收到的数据事件是否发生取决于数据 buffer 的配置。在主机配置下，MOSI 输出被禁能，该端口可以用作 GPIO。只要 SPI 被使用，时钟信号就被连续的产生。停止时钟的唯一方法是清零 RXONLY 或者 SPE，直到来自 MISO 的输入完成。

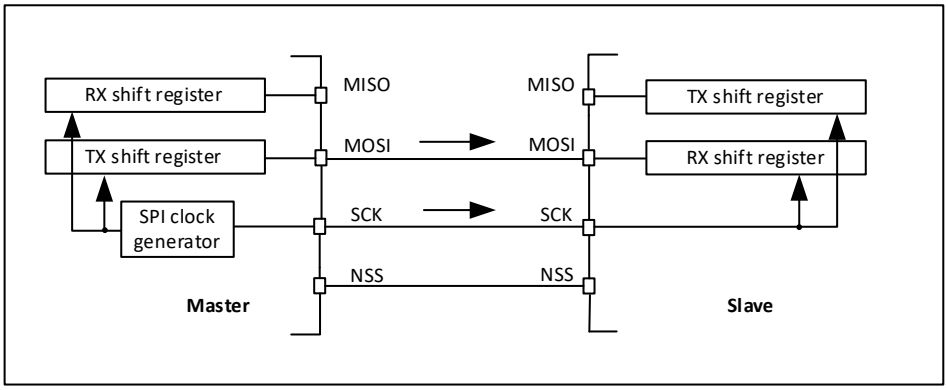


图 21-4 单工单从机/单主机应用

(主设备处于仅发送模式/从设备处于仅接收模式)

（1）在主机和从机之间可以使用 NSS 进行硬件控制流。可选的，NSS 也可以不使用。然后该流程就要内部处理。

（2）在 Rx shift 寄存器的输入捕获意外的输入信息。在标准的 transmit-only 模式下，所有与传输接收相关的事件都被必须被忽略。

（3）在该配置下，两边的 MISO pin 都被用作 GPIO。

通过用传输方向的设定（在 BIDIOE bit 未发生改变时，双向模式被使能），任何 simplex 通讯可以被 half-duplex 通讯代替。

21.3.3. 多从机通信

在一个有两个或者更多独立从机的配置里，主机为每个从机，使用 GPIO 来管理 NSS。主机必须通过拉低连接从机 NSS 选择某个从机。当完成这个，标准的主机和专门的从机通讯就被建立了。

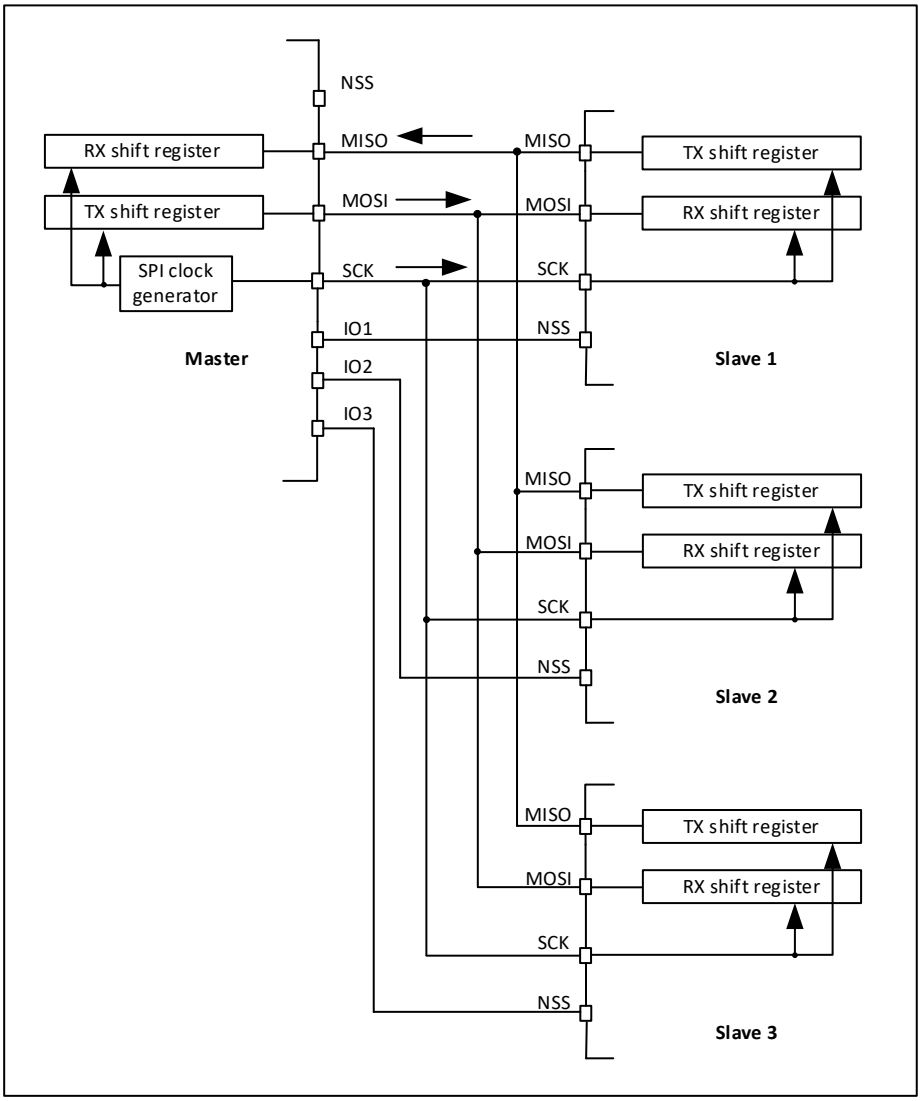


图 21-5 主机与三个独立的从机通信

NSS 在这种配置下在主机端未被使用。必须通过 SSM=1, SSI=1 来防止任何 MODF 错误。

由于从机的 MISO 连接到一起，所有从机必须把他们 MISO 的 GPIO 配置作为 AF open-drain。

21.3.4. 多主机通信

除非 SPI 总线不是被设计成具备多主机功能，否则用户可以使用其内嵌 feature，该 feature 可以发现两个试图同时控制总线的节点存在的潜在冲突。当需要用到这种检测时，要使用配置为硬件输入模式的 NSS pin。

在这种模式下有两个以上 SPI 节点的连接是不可能的，因为单次只有一个节点可以在公共数据线上传输。

当节点无效，缺省下两个都保持从机模式。一旦节点要控制总线，它自己切换成主机模式，并把有效的电平经过专门的 GPIO 给予其余节点的从机 select input。在该进程完成后，有效的从机 select 信号被释放，控制总线的节点暂时返回 passive mode，并等待新的进程开始。

如果两个节点在同一时间都给出控制请求，总线冲突事件就会产生（查看 MODF 事件）。然后用户可以应用一些简单的仲裁过程（例如：通过提前定义的给两个节点的不同的超时推迟下一次尝试）

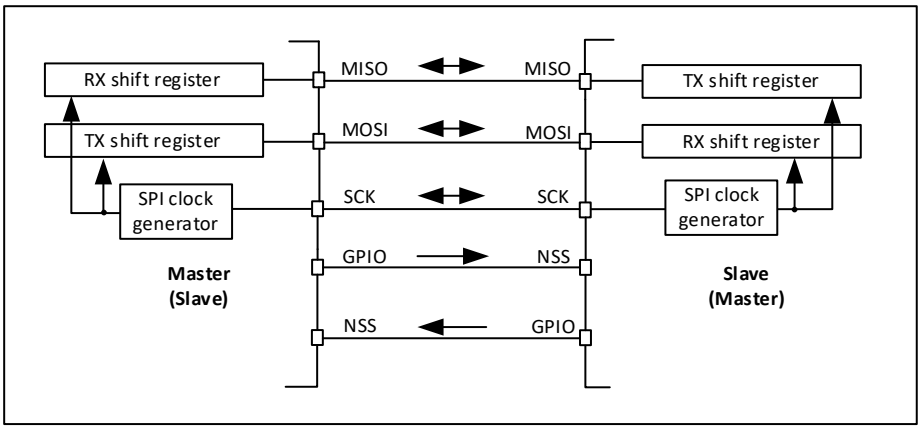


图 21-6 Multi-主机 application

NSS 在两个节点都被配置成硬件输入模式。他的有效电平使能了 MISO 输出控制，而 passive node 被配置成从机。

21.3.5. 从选择(NSS)脚管理

在从机 mode，NSS 作为标准的片选输入，使从机能与主机通讯。在主机 mode，NSS 既可以作为输出又可以作为输入。当作为输入时，它可以防止多主机的总线冲突，当作为输出时，它可以驱动单个从机的从机选择信号。

通过 SPI_CR1 寄存器的 SSM bit，可以选择硬件或者软件从机 management：

- 软件 NSS management（SSM=1）：在这个配置下，从机 select 信号被内部的 SSI bit（SPI_CR1 寄存器）值驱动。外部 NSS pin 被释放给其他应用使用。
- 硬件 NSS management（SSM=0）：在这个情况下，有几个可能的配置。
  1） NSS 输出使能（SSM=0, SSOE=1）：这个配置仅在作为主机时使用。硬件管理 NSS pin。当 SPI 一在主机模式被使能（SPE=1），NSS 信号就被拉低并保持低电平，直到 SPI 被 disable（SPE=0）。在多主机应用中，SPI 不能进行这种 NSS 配置。
  2） NSS 输出 disable（SSM=0, SSOE=0）：如果 MCU 在总线上作为主机，这个配置允许进行多主机能力。如果 NSS pin 此时被拉低，SPI 进入主机 mode fault 状态，芯片自动被重配置为从机模式。在从机模式，NSS pin 作为标准的片选输入，当 NSS 为低时，从机被选中。

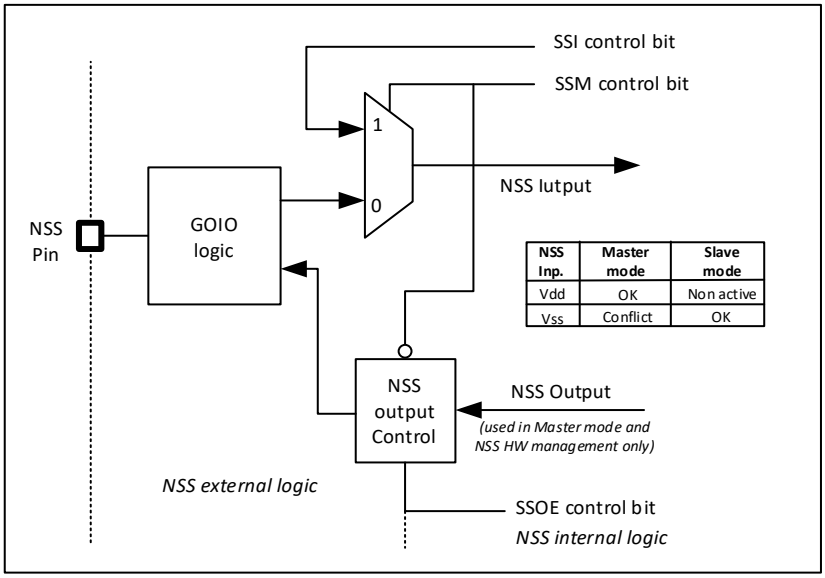


图 21-7 hardware/software slave select management

21.3.6. 通讯格式

在 SPI 通讯期间，接收和发送操作同时进行。SCK（serial clock）将数据线上的信息移位和采样操作同步。通讯格式取决于时钟相位、时钟极性和数据帧格式。为了能够进行通讯，主机和从机必须遵循相同的通讯格式。

21.3.6.1. Clock phase and polarity controls

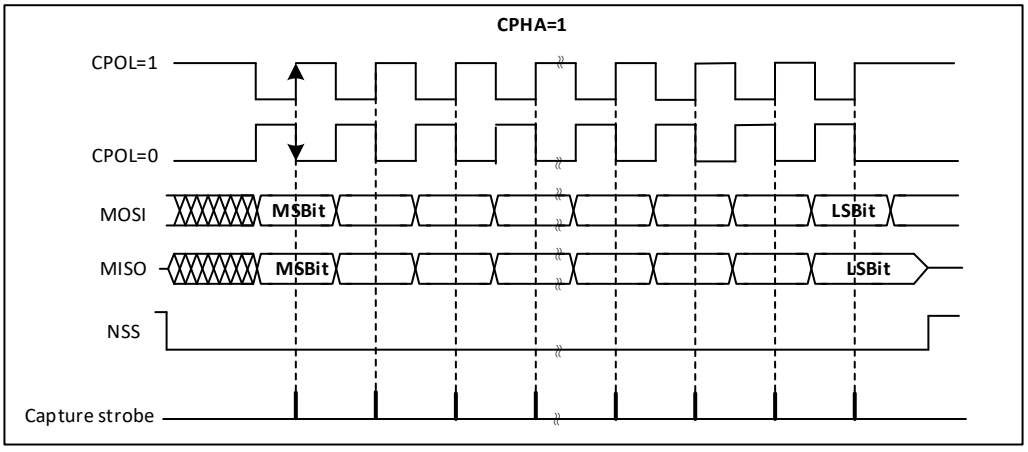
通过 CPOL 和 CPHA bit（SPI_CR1 寄存器），软件可以配置 4 种可能的时序。CPOL（clock polarity）控制当没有数据传输时的 clock 的 IDLE 状态。该位对主机和从机都有影响。如果 CPOL 被复位，SCK pin 有低电平的状态。如果 CPOL 被置位，SCK pin 有高电平的 IDLE 状态。

如果 CPHA 被置位，SCK 的第二个边沿捕获传输的第一个数据位（如果 CPOL 被复位，是下降沿，否则是上升沿）。在时钟变化类型的出现，数据被锁存。如果 CPHA 被复位，SCK 的第一个边沿捕获第一个传输的数据位（如果 CPOL 被置位，是下降沿，否则是上升沿）。在该时钟变化类型出现时，数据被锁存。

CPOL 和 CPHA 的组合选择了数据捕获时钟边沿。

在 CPOL/CPHA 改变之前，SPI 必须被 disable（SPE=0）。

SCK 的 IDLE 状态必须对应被 SPI_CR1 寄存器选择的极性。



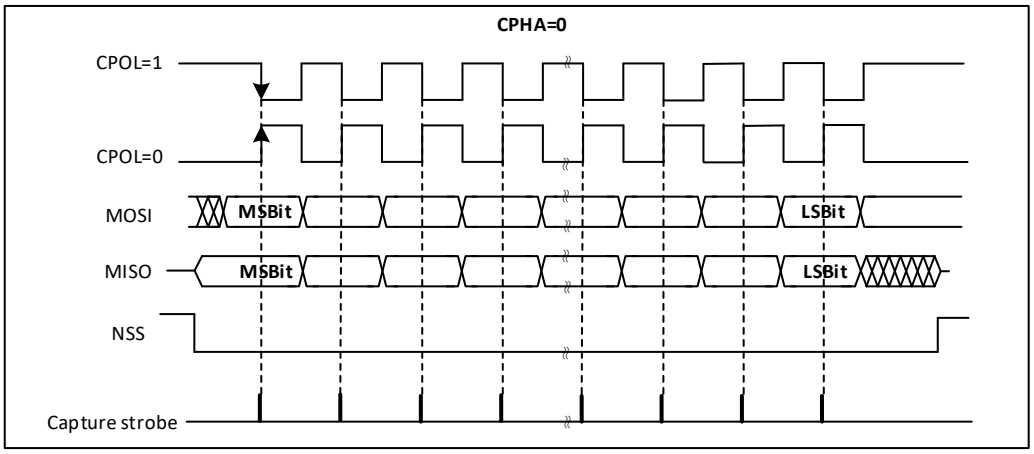


图 21-8 数据时钟时序图

数据 bit 的顺序取决于 LSBFIRST bit 的设定。

21.3.6.2. 数据帧格式

通过 LSBFIRST bit(SPI_CR1 寄存器)，SPI shift 寄存器可以设定为 MSB-FIRST 或者 LSB-FIRST。通过使用 DS bit(SPI_CR2 寄存器)，选择数据帧的位数。可选择为 8 位或者 16 位长度，该设置对于发送和接收都适用。

21.3.7. SPI 配置

对于主机和从机，SPI 的配置流程几乎一样。对于具体的模式建立，遵循专门的章节介绍。当进行标准的通讯，进行以下步骤：

1. 写相关的 GPIO 寄存器：配置 MOSI、MISO 和 SCK pin
2. 写 SPI_CR1 寄存器
   1） 通过 BR[2:0]配置时钟波特率（从机模式不需要）
   2） 配置 CPOL 和 CPHA
   3） 通过 RXONLY 或者 BIDIMODE 和 BIDIOE（RXONLY 和 BIDIMODE 不能同时有效），选择 simplex 或者 half-duplex 模式
   4） 配置 LSBFIRST
   5） 配置 SSM 和 SSI
   6） 配置 MSTR bit（在多主机 NSS 配置中，如果主机被配置防止 MODF 错误，要避免 NSS 的冲突状态）
3. 写 SPI_CR2 寄存器
   1） 配置 DS bit，选择数据帧位数
   2） 配置 SSOE（从机模式不需要）
   3） 配置 FRXTH bit。RXFIFO 阈值必须与对 SPI_DR 寄存器访问的位数对齐

21.3.8. SPI 使能流程

推荐在主机发送时钟之前使能 SPI 从机。如果不这样处理，不期望的数据传输可能会发生。从机的数据寄存器必须在开始与主机通讯之前，已经包含要被发送的数据（或者在通讯时钟的第一个沿，或者如果时钟信号是连续的情况，要在正在进行的通讯结束之前）。SCK 信号必须在 SPI 从机被使能之前，固定在 IDLE 状态 level（相应的被选择极性）。

Full-duplex 模式（或者 transmit-only），当 SPI 被使能并且 TXFIFO 不空，或者向 TXFIFO 进行下一个写， 主机开始通讯。

在任何主机 receive-only 模式（RXONLY=1，或者 BIDIMODE=1 且 BIDIOE=0），在 SPI 被使能后，主机开始通讯，时钟立即被提供。

21.3.9. 数据传输和接收流程

21.3.9.1. RXFIFO and TXFIFO

SPI 所有数据通讯都通过 32-bit 的 FIFO。该特性使 SPI 能够以连续数据流进行工作，并防止由于 CPU 来不及处理数据导致的通讯问题。发送和接收有独立的 FIFO，叫做 TXFIFO 和 RXFIFO。这些 FIFO 被用在所有的 SPI 模式。

FIFO 的处理取决于多种参数，包括：数据交换模式（全双工、半双工）、数据帧格式、访问 FIFO 数据寄存器的大小（8 位还是 16 位）。

读 SPI_SR 寄存器会得到最早存放在 RXFIFO 中还未被读走的数据结果。写 SPI_DR 寄存器，会在 FIFO 发送队列的最后位置，存入被写的数据。读访问必须通常与 RXFIFO 阈值对齐，该阈值是通过 SPI_CR2 寄存器的 FRXTH 位配置的。FTLVL[1:0]和 FRLVL[1:0]位显示了两个 FIFO 当前的占用级别。

对 SPI_DR 寄存器的访问必须通过 RXNE 事件管理。当数据存储器在 RXFIFO 并且达到阈值（被 FRXTH 位定义的），该事件被触发。当 RXNE 被清零，RXFIFO 就被认为是空的。

相似地，写要发送的数据帧，通过 TXE 事件管理。当 TXFIFO Level 小于或者等于总容量的一半时，该事件就会被触发。否则，TXE 被清零，并且 TXFIFO 被认为是满的。

用这样的方式，RXFIFO 可以存 4 个数据帧，而 TXFIFO 仅可以存 3 个数据帧（当数据帧格式不大于 8bit）。这样的差别，可以防止以下混乱情况的发生：3 个 8-bit 数据帧已经存在 TXFIFO，而此时软件要向 TXFIFO 以 16-bit（CPU 数据宽度）模式再写入数据。

TXE 和 RXNE 事件都都可以通过查询、中断方式处理。

当 RXFIFO 满时，如果下一个数据被接收，则 overrun 事件产生。Overrun 事件可以通过查询和中断的方式处理。

被置位的 BSY 位显示了 1 个当前数据帧的通讯正在进行。当时钟信号连续的提供，在主机端的两个数据帧之间，BSY 标志保持置位。但在从机端的每个数据帧传输之间，BSY 会保持最小 1 个 SPI Clock 宽度的低电平。

21.3.9.2. Sequence handling

一些数据帧可以通过 single sequence 传递来完成一条信息。当发送被使能，当 maser 的 TXFIFO 里有任何数据，sequence 开始并继续进行。时钟信号被主机连续的提供，直到 TXFIFO 空，然后停止等待额外的数据。

在 receive-only 模式，即 half-duplex（BIDIMODE=1, BIDIOE=0）或者 simplex 模式（BIDIMODE=0，RXONLY=1），在 SPI 被使能并且 receive-only 模式被激活的时候，主机就立即开始接收。主机一直会提供时钟并连续地接收数据，直到主机停止了 SPI 或者关闭了 receive-only 模式。

当主机能够以连续的模式（SCK 信号是连续的），提供所有通讯，主机必须要考虑从机处理数据流的能力。当有必要时，主机必须降低通讯速度，并提供或者更慢的时钟，或者分开的帧，或者重组 delay 的数据包。要注意的是，对于主机或者从机来说，没有 underflow 错误信号，来自于从机的数据通常被主机交互和处理（即使从机不能及时准备好数据）。

每个序列都必须被 NSS 脉冲包住，同时在多从机系统钟选择要进行通讯的其中的一个从机。在一个单从机系统，没有必要用 NSS 去控制从机，但通常也最好提供脉冲，使从系统与每个数据序列的开头同步。NSS 可以由软件和硬件两种方式管理。

当 BSY 被置位，它显示了正在进行的数据帧交互。当专门的帧交互被完成时，RXNE 标志置位。最后一个位被采样，并且整个数据帧被存在 RXFIFO 中。

21.3.9.3. 禁能 SPI 的步骤

当 SPI 被禁能掉，必须按照特定的禁能流程。对于系统禁能 SPI 的流程是很重要的，因为此后应用上，外设时钟会被停掉，系统进入低功耗模式。这种情况下（禁能），正在进行的交互会被破坏。在一些模式下，禁能流程是唯一停止连续通讯的办法。

全双工或者仅传输模式下，主机可以当停止提供要发送的数据时完成交互。在这种情况下，在最后的数据交互后，时钟被停止。要额外注意打包模式（当交互奇数个数的数据帧，以放置一些虚拟字节交互）。在这些模式下，SPI 被禁能之前，用户必须使用标准的禁能流程。当 SPI 被禁能在主机发送时，如果此时一个帧交互正在进行，或者下一个数据帧存在 TXFIFO 中，SPI 的功能是不能被保证的。

当主机处在任何仅接收模式，停止连续时钟的唯一方法是停止外设（SPE=0）。该模式下，要进行专门的 SPI 禁能流程。

当 SPI 被禁能，接收到未读走的数据存放在 RXFIFO 中，这些数据必须在下一次 SPI 使能要开始新的序列之前被处理掉。为防止有未读的数据，要确保当 SPI 被禁能时，RXFIFO 是空的（使用正确的禁能流程，或者通过用软件复位以初始化所有的 SPI 寄存器）。

标准的禁能流程是基于 BSY 状态，并查看 FTLVL[1:0]，以确保传输彻底完成。也可以通过特定别的检查来鉴别正在进行交互的结束，例如：

- 当 NSS 信号被软件管理，主机要向从机提供正确的 NSS 脉冲。 或者
- 当完成来自 FIFO 的交互数据流时，此时最后的数据帧仍在传输过程中。

正确的禁能流程是（仅接收模式除外）：

1. 等待 FTLVL[1:0]=00（没有数据要发送）
2. 等待 BSY=0（最后的数据被处理完成）
3. Disable SPI（SPE=0）
4. 读数据，直到 FRLVL[1:0]=00（读所有接收到的数据）

对于特定只接收模式，正确的禁能流程是：

1. 在最后一个数据帧传输过程中，通过禁能 SPI（SPE=0），打断接收流程
2. 等待 BSY=0（最后的数据帧已被处理）
3. 读数据，直到 FRLVL[1:0]=00（读所有接收到的数据）

21.3.9.4. 数据打包

当 frame size= 8, 任何 16-bit 的读或者写访问时，都会自动使用数据打包。在这种情况下，双数据帧会被并行处理。首先，SPI 使用存储在被访问字低位的模式，然后是存在高位的。

下图提供了数据打包处理过程。在发送方的单个 16-bit 访问后，两个数据帧被发送。在接收方，如果 RXFIFO 阈值被是 16 bits（FRXTH=0），则该序列会在 RXNE 事件就立即产生。作为对 RXNE 事件的响应，接收方通过一个 16-bit 读 SPI_DR 寄存器，访问了 2 个数据帧。在接收端，RxFIFO 阈值的设定和接下来的读访问必须保持对齐，否则数据会丢失。

在发送端，用 8-bit 访问方式写奇数序列的最后一个数据帧是足够的。为了产生 RXNE 事件，对于奇数个数据帧，对于接收的最后一个数据帧，接收方必须改变 Rx_FIFO 阈值。

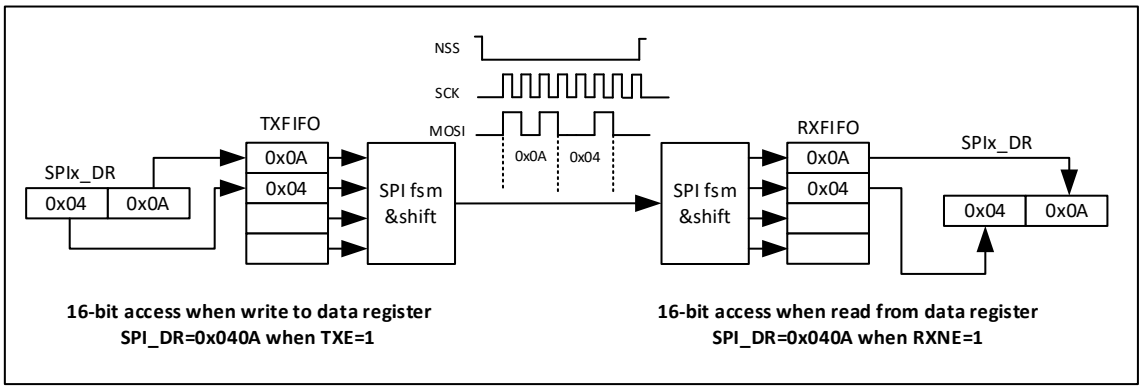


图 21-9 用于传输和接收的先进先出封装数据

21.3.9.5. 通信图

本节介绍一些典型的时序，这些时序对于查询、中断都是有效的。为了简化，假定 LSBFIRST=0, CPOL=0。

1. 当 NSS 有效，SPI 被使能，从机开始控制 MISO；当 NSS 被释放或者 SPI 关闭时从机失去对 MISO 的控制。对于从机，在传输开始前必须提供充足的时间给主机，以便提前准备数据。

   在主机端，仅在 SPI 被使能时，SPI 外设会控制 MOSI 和 SCK 信号（也包括 NSS 信号）。如果 SPI 被 disable，SPI 外设就从 GPIO 断开，因此在这些线上的电平值取决于 GPIO 的设定。
2. 在主机端，如果通讯是连续的，则 BSY 在帧之间保持有效。在从机端，BSY 信号在数据帧之间通常变低至少一个时钟周期。
3. 只有当 TXFIFO 是满的，TXE 信号才被清零。
4. 在 TXEIE 被置位后，产生 TXE 中断。当 TXE 信号有效时，开始向 TxFIFO 传输数据，直到 TxFIFO 变满。
5. 。这个标志在 SPI 交互完成之前，一直都为高。
6. 在 Data packed mode, TxE 和 RxNE 事件是成对出现的，每个读/写 FIFO 的访问是 16bit 宽（until the number of data frames are even）。如果 TxFIFO 是 3/4 full，FTLVL 状态停在 FIFO full level。这就是为什么最后一个奇数 frame 不能在 TxFIFO 变成 1/2 full 之前存储。该数据 frame 以 8-bit 的访问方式（软件）存储在 TxFIFO 中。
7. 为了接收 packed mode 的最后一个奇数 data frame，当最后一个数据 frame 被处理时，Rx 阈值必须被改变成 8-bit（或者是软件置位 FRXTH=1）。

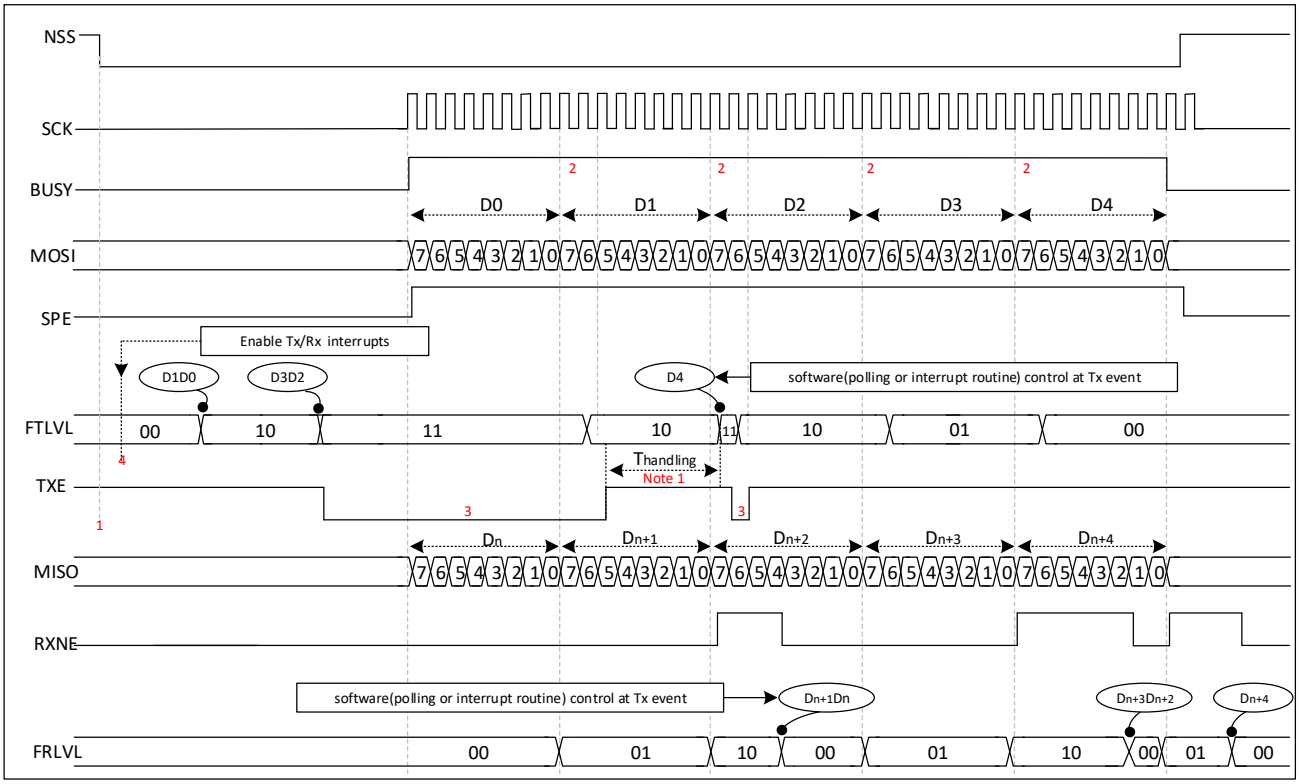


图 21-10 主机全双工通信图(bit frame=8, FRXTH=0)

Note1：$T_{handling}$ 即 cpu 写数据到 Tx fifo 所用的时间

21.3.10. 状态标志

应用程序通过 3 个状态标志可以完全监控 SPI 总线的状态。

21.3.10.1. 发送缓冲区空标志(TXE)

当 TXFIFO 有足够的空间存放要发送的数据时，TXE 标志位被置位。TXE 标志位与 TXFIFO level 有关。该标志位变高并保持高电平，直到 TXFIFO level 小于等于 1/2 FIFO 深度才会被硬件清零。如果 TXEIE（SPI_CR2）被置位，则会产生中断请求。当 TXFIFO level 大于 1/2，该位被自动清零。

21.3.10.2. 接收缓冲非空标志(RXNE)

取决于 FRXTH 位（SPI_CR2）的值，RXNE 标志位才会被置位：

- 如果 FRXTH 为 1，RXNE 变高并保持高电平，直到 RXFIFO level 大于或者等于 1/4(8-bit)。
- 如果 FRXTH 为 0，RXNE 变高并保持高电平，直到 RXFIFO level 大于或者等于 1/2(16-bit)。

如果 RXNEIE 位（SPI_CR2）被置位，则产生中断。

当上述条件不再成立，则 RXNE 被硬件自动清零。

21.3.10.3. 忙标志(BSY)

BSY 标志由硬件设置与清除(写入此位无效果)，此标志表明 SPI 通信层的状态。

当它被设置为'1'时，表明 SPI 正忙于通信，但有一个例外：在主模式的双向接收模式下(MSTR=1、BDM=1 并且 BDOE=0)，在接收期间 BSY 标志保持为低。

在软件要关闭 SPI 模块并进入停机模式(或关闭设备时钟)之前，可以使用 BSY 标志检测传输是否结束，这样可以避免破坏最后一次传输，因此需要严格按照下述过程执行。

BSY 标志还可以用于在多主机系统中避免写冲突。

除了主模式的双向接收模式(MSTR=1、BDM=1 并且 BDOE=0)，当传输开始时， BSY 标志被置'1'。

以下情况该标志将被清除为'0'：

● 当 SPI 被正确的禁能掉

● 主机模式，当产生 MODF=1

● 主机模式，当传输完成，不再有效数据要发送

● 从机模式，在每个数据传输之间，BSY 标志置为 0，并保持至少一个 SPI 时钟周期

Note: 不要使用 BSY 标志处理每个数据发送和接收。使用 TXE 和 RXNE 更合适。

21.3.11. 错误标志

21.3.11.1. 主模式失效(MODF)

主模式失效（MODF）仅发生在：当 NSS 作为输入信号（SSOE=0），NSS 引脚硬件模式管理下，主设备的 NSS 脚被拉低；或者在 NSS 引脚软件模式管理下，SSI 位被置为'0'时。此时，MODF 位被自动置位。主模式失效对 SPI 设备有以下影响：

● MODF 位被置为'1'，如果设置了 ERRIE 位，则产生 SPI 中断；

● SPE 位被清为'0'。这将停止一切输出，并且关闭 SPI 接口；

● MSTR 位被清为'0'，因此强迫此设备进入从模式。

下面的步骤用于清除 MODF 位：

1. 当 MODF 位被置为'1'时，执行一次对 SPI_SR 寄存器的读或写操作；

2. 然后写 SPI_CR1 寄存器。

在有多个 MCU 的系统中，为了避免出现多个从设备的冲突，必须先拉高该主设备的 NSS 脚，再对 MODF 位进行清零。在完成清零之后， SPE 和 MSTR 位可以恢复到它们的原始状态。

出于安全的考虑，当 MODF 位为'1'时，硬件不允许设置 SPE 和 MSTR 位。

通常配置下，从设备的 MODF 位不能被置为'1'。然而，在多主配置里，一个设备可以在设置了 MODF 位的情况下，处于从设备模式；此时，MODF 位表示可能出现了多主冲突。中断程序可以执行一个复位或返回到默认状态来从错误状态中恢复。

21.3.11.2. 过载模式

当数据被主机或者从机接收，并且接收缓冲区没有足够的空间存接收到的数据时，产生 overrun 情况。如果软件没有足够的时间读走以前接收到的数据（RXFIFO 中存放），该情况就会发生。

当 overrun 情况发生，新收到的数据不会改写以前存放在 RXFIFO 的数据。接收到的新数据被忽略，并且所有接下来发送的数据丢失。

依次读出 SPI_DR 寄存器和 SPI_SR 寄存器可将 OVR 清除。

21.3.12. SPI 中断

表 21-1 SPI 中断请求

中断事件	事件标志	使能控制位
TXFIFO 等待被装载	TXE	TXEIE
数据接收到 RXFIFO 中	RXNE	RXNEIE
主模式失效事件	MODF	ERRIE
溢出错误	OVR	ERRIE

21.4. SPI 寄存器

SPI 对应的寄存器可以进行 16-bit 和 32-bit 访问，DR 寄存器支持 32-bit、16-bit 和 8-bit 访问。

21.4.1. SPI 控制寄存器 1 (SPI_CR1)

Address offset:0x00

Reset value:0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BI-DIMODE	BIDIOE	Res	Res	Res	RXONLY	SSM	SSI	LSBFIRST	SPE	BR[2:0]			MSTR	CPOL	CPHA
RW	RW				RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
15	BIDIMODE	RW	0	双向数据模式使能。 0：“双线单向”模式 1：“单线双向”模式
14	BIDIOE	RW	0	双向模式输出使能。 与 BIDIMODE 位一起配置“单线双向”模式下数据的输出方向。 0：输出禁止（只收模式） 1：输出使能（只发模式） “单线”在主设备端位 MOSI 引脚，在从设备端为 MISO 引脚。
13:11	Reserved	RES	-	Reserved
10	RXONLY	RW	0	仅接收控制。 该位和 BIDIMODE 位一起决定在“双线单向”模式下的传输方向。在多个从设备的配置中，在未被访问的从设备上该位置 1，使得只有被访问的从设备才有输出，因而不会造成数据线上有数据冲突。 0：全双工（发送和接收） 1：禁止输出（只接收模式）
9	SSM	RW	0	软件从设备管理。 当 SSM 置位，NSS 引脚上的电平由 SSI 位的值决定。 0：禁止软件从设备管理 1：使能软件从设备管理
8	SSI	RW	0	内部从设备选择。 该寄存器只有当 SSM=1 时才有效。该寄存器决定了 NSS 上的电平，在 NSS 引脚上的 I/O 操作无效。
7	LSBFIRST	RW	0	帧格式。 0：先发送 MSB 1：先发送 LSB 通讯进行时不能改变该寄存器的值。
6	SPE	RW	0	SPI 使能。 0：禁止 SPI 1：使能 SPI
5:3	BR[2:0]	RW	0	波特率控制。 000：$f_{PCLK}/2$ 001：$f_{PCLK}/4$ 010：$f_{PCLK}/8$ 011：$f_{PCLK}/16$ 100：$f_{PCLK}/32$ 101：$f_{PCLK}/64$ 110：$f_{PCLK}/128$ 111：$f_{PCLK}/256$ 通讯进行时不能改变该寄存器的值。 注：从机模式下，最快波特率仅支持 $f_{PCLK}/4$。
2	MSTR	RW	0	主设备选择。 0：配置为从设备 1：配置为主设备 通讯进行时不能改变该寄存器的值。
1	CPOL	RW	0	时钟极性。

Bit	Name	R/W	Reset Value	Function
				0：空闲状态时，SCK 保持低电平 1：空闲状态时，SCK 保持高电平 通讯进行时不能改变该寄存器的值。
0	CPHA	RW	0	时钟相位。 0：数据采样从第一个时钟边沿开始 1：数据采样从第二个时钟边沿开始 通讯进行时不能改变该寄存器的值。

21.4.2. SPI 控制寄存器 2 (SPI_CR2)

Address offset:0x04

Reset value:0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SLVFM	Res	Res	FRXTH	DS	Res	Res	Res	TXEIE	RXNEIE	ERRIE	Res	Res	SSOE	Res	Res
RW			RW	RW	RW	RW	RW	RW	RW	RW			RW		

Bit	Name	R/W	Reset Value	Function
31:16	Reserved		-	Reserved
15	SLVFM	RW	0	从机 fast mode enable 0：从机 normal mode，从机模式支持最快 SPI clock 的速度小于 pclk/4 1：从机 fast mode，可支持从机模式下 SPI clock 速度可到 pclk/4 注：当 SPI clock 的速度小于 pclk/4 时，一定不能设定该寄存器位。
14:13	Reserved		-	Reserved
12	FRXTH	RW	0	FIFO 接收阈值 该位用来设定触发 RXNE 事件的 RXFIFO 阈值。 0：如果 FIFO level 大于或者等于 1/2（16-bit），产生 RXNE 1：如果 FIFO level 大于或者等于 1/4（8-bit），产生 RXNE
11	DS	RW	0	SPI 传输数据长度 0：8-bit 数据帧传输 1：16-bit 数据帧传输
10:8	Reserved			
7	TXEIE	RW	0	发送缓冲区空中断使能 0：禁止 TXE 中断 1：使能 TXE 中断。TXE=1 时产生中断请求。
6	RXNEIE	RW	0	接收缓冲区非空中断使能 0：禁止 RXNE 中断 1：使能 RXNE 中断。RXNE=1 时产生中断请求。
5	ERRIE	RW	0	错误中断使能。 0：禁止错误中断 1：使能错误中断。当 CRCERR、OVR 或 MODF 为 1 时，产生中断请求。
4:3	Reserved	RES	-	Reserved
2	SSOE	RW	0	SS 输出使能。 0：禁止在主模式下 SS 输出，该设备可以工作在多主设备模式 1：开启主模式下 SS 输出，该设备不能工作在多主设备模式。
1:0	Reserved	RES	-	Reserved

Note:

FRXTH 与 DS 搭配共计有 4 种组合方式，但限制软件的流程使用如下：

1. 如果配置了 DS=F (即传输数据长度为 16)，则该位应为 0
2. 如果配置了 DS=7 (即传输数据长度为 8)，则需要区分如下两种情况：

1） 如果通讯的数据帧数为 1 帧数据，则需要将该位置为 1

2） 如果通讯的数据帧数为大于 1 帧数据，则将该位清为 0

21.4.3. SPI 状态寄存器 (SPI_SR)

Address offset:0x08

Reset value:0x0002

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	FTLVL [1:0]		FRLVL [1:0]		Res	BSY	OVR	MODF	Res	Res	Res	TXE	RXNE
			R	R	R	R		R	R	R				R	R

Bit	Name	R/W	Reset Value	Function
31:13	Reserved	RES	-	Reserved
12:11	FTLVL	R	0	FIFO 发送 level。硬件置位，硬件清零 00：FIFO 空 01：1/4 FIFO 10：1/2 FIFO 11：FIFO full(当 FIFO 阈值大于 1/2，即认为是满)
10:9	FRLVL	R	0	FIFO 接收 level。硬件置位，硬件清零 00：FIFO 空 01：1/4 FIFO 10：1/2 FIFO 11：FIFO 满
7	BSY	R	0	忙标志。 0：SPI 不忙； 1：SPI 处于通讯，或者发送缓冲非空。
6	OVR	R	0	溢出标志。 0：无溢出错误 1：产生溢出错误 该寄存器由硬件置位，或者软件序列复位（上溢和下溢序列不同）。
5	MODF	R	0	模式错误。 0：无模式错误 1：出现模式错误 该寄存器由硬件置位，或者软件序列复位。
4:2	Reserved		0	
1	TXE	R	1	发送缓冲空。 0：发送缓冲非空 1：发送缓冲为空
0	RXNE	R	0	接收缓冲非空。 1：接收缓冲非空 0：接收缓冲为空

21.4.4. SPI 数据寄存器 (SPI_DR)

Address offset:0x0C

Reset value:0x0000

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

Bit	Name	R/W	Reset Value	Function
15:0	DR[15:0]	RW	0	数据寄存器。 要发送或者接收到的数据。

Bit	Name	R/W	Reset Value	Function
				数据寄存器作为 RxFIFO 和 TxFIFO 的接口。当要读数据，实际访问 RxFIFO，而要写数据，实际访问 TxFIFO。 Note：取决于 DS 位（数据帧宽度选择），数据发送或者接收是 8-bit 或者 16-bit。 对于 8-bit 数据帧，数据寄存器是基于 right-aligned 的 8-bit 数据进行发送和接收的。当在接收模式，DR[15:8]硬件置为 0。 对于 16-bit 数据帧，数据寄存器是 16-bit 的，整个 DR[15:0]都用作发送和接收。

21.4.5. SPI 寄存器映像

Offset	Register	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	SPI_CR1	BI-DIMODE	BIDIOE	Res.	Res.	Res.	RXONLY	SSM	SSI	LSBFIRST	SPE	BR[2:0]			MSTR	CPOL	CPHA
	Reset value	0	0				0	0	0	0	0	0	0	0	0	0	0
0x04	SPI_CR2	SLVFM	Res.	Res.	FRXTH	DS	Res.	Res.	Res.	TXEIE	RXNEIE	ERRIE	Res.	Res.	SSOE	Res.	Res.
	Reset value	0	0			0				0	0	0			0		
0x08	SPI_SR	Res.	Res.	Res.	FTLVL[1:0]		FRLVL[1:0]		Res.	BSY	OVR	MODEF	Res.	Res.	Res.	TXE	RXNE
	Reset value				0	0	0	0		0	0	0				1	0
0x0C	SPI_DR	DR[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

22. 调试支持

22.1. 概况

本芯片基于 Cortex-M0+ CPU，该 CPU Core 包含高级 debug 硬件扩展功能。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。

调试功能在由调试主机在连接和调试 MCU 时使用，调试的接口是 serial wire。在 M0+ CPU Core 中的调试功能是一套 ARM CoreSight Design kit。

M0+提供了集成的片上调试支持，由以下部分组成：

- SW-DP：serial wire
- BPU：Break point unit
- DWT：Data watchpoint trigger

调试支持也包括了本芯片的调试集成功能：

- 灵活的调试引脚分配，SWIO@PA13、SWCLK@PA14
- MCU 调试盒（支持低功耗模式，控制外设时钟等

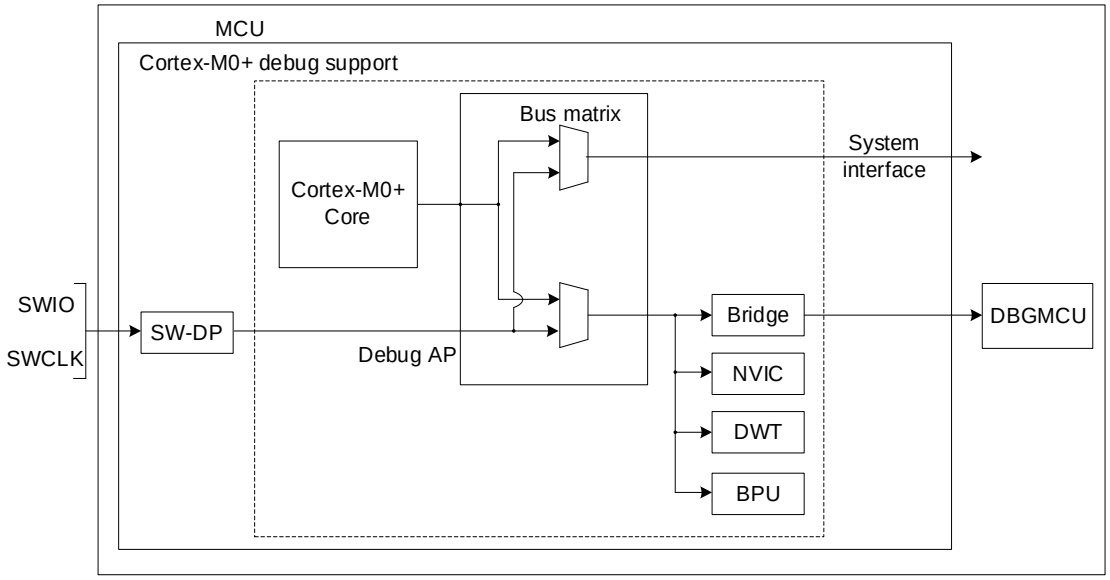


图 22-1 DBG 框图

22.2. 引脚分布和调试端口脚

22.2.1. SWD 调试端口

调试功能相关的端口有两个，在所有封装形式都可见。

表 22-1 DBG 框图

SW-DP 端口引脚名称	SW 调试接口		引脚分配
	类型	调试功能	
SWDIO	输入/输出	串行数据输入/输出	PA13
SWCLK	输入	串行时钟	PA14

22.2.2. 灵活的 SW-DP 脚分配

在芯片复位后（系统复位或者上电复位），用作 SW-DP 的端口被分配作为专门被调试主机立即使用的 pin。

然而，芯片提供了关闭 SWD 端口的可能性，并释放该端口作为 GPIO 用。

22.2.3. SWD 脚上的内部上拉和下拉

一旦 SWD 端口被软件释放，则 GPIO 控制器控制了这两个端口。GPIO 控制寄存器的复位状态把 IO 置为同等的状态：

- SWDIO：input pull-up
- SWCLK：input pull-down

片内的上拉和下拉电阻为外围节省了增加电阻的需求。

22.3. ID 代码和锁定机制

芯片内存放 ID code。推荐 Keil、IAR 等工具使用该 ID Code（位于 0x4001 5800 地址）锁住调试。

芯片上电后，硬件读取 flash 的 factory config. byte 的 0x1FFF 0FF8 地址，装载到 DBG_IDCODE 寄存器中。

22.4. SWD 调试端口

22.4.1. SWD 协议介绍

这是个同步的串行通讯协议，使用以下两个端口：

- SWCLK：来自主机给芯片的 clock 信号
- SWDIO：双向数据信号

该协议允许两个 bank 的寄存器（DPACC 寄存器和 APACC 寄存器）被读和写入。数据位是按照在线上的 LSB-first 传输。对于 SWDIO 的双向管理，线上必须在板级上拉（推荐 100k 欧的电阻）。

在协议中每次 SWDIO 方向的改变，转向时间被插入在线上既没有被主机，也没有被芯片驱动的情况。缺省状态下，这个转向时间是 1 个位的时间，然而整个可以通过配置 SWCLK 频率来调整。

22.4.2. SWD 协议序列

每个序列由以下阶段组成：

- 主机发送的包请求（8bits）
- 芯片发送的应答响应（3bits）
- 主机或者芯片的数据发送阶段（33bits）

表 22-2 请求包(8-bits)

比特位	名称	描述
0	Start	必须为“1”
1	ApnDP	0：DP 访问 1：AP 访问
2	RnW	0：写请求 1：读请求
4:3	A[3:2]	DP 或者 AP 寄存器的地址区域
5	Parity	以前位的校验位
6	Stop	0
7	Park	没有被主机驱动。由于上拉属性，会被芯片读出 1。

通常转向时间（缺省为 1bit）跟随着包请求，此时主机和芯片都没有驱动信号线。

表 22-3 ACK 响应（3bits）

比特位	名称	描述
[2:0]	ACK	001：FAULT 010：WAIT 100：OK

如果一个读操作或者如果 1 个 wait 或者 FAULT 应答被接收到，则转向时间必须跟随 ACK 响应。

表 22-4 DATA 传输（33bits）

比特位	名称	描述
[31:0]	WDATA 或者 RDATA	写或者读数据
32	校验位	对[31:0]的奇偶校验位

如果是个读操作时，转向时间必须跟随着数据传输。

22.4.3. SW-DP 状态机(reset, idle states, ID code)

SW-DP 的状态机有个定义了 SW-DP 的内部 ID 代码。它遵循 JEP-106 标准。这个 ID 代码是缺省的 ARM 代码，并被置位 0x0BB11477（对应 Cortex-M0）。

22.4.4. DP and AP 读/写访问

- 读 DP 的操作不会被 posted: 芯片响应可以被立即（ACK=OK），或者可以被延迟（ACK=WAIT）
- 读 AP 的操作被 posted：这意味着访问的结果被返回到下一次传输。如果下一次要进行的访问不是 AP 访问，则 DP-RDBUFF 寄存器必须被地呼出获得该结果。

  DP-CTRL/STAT 寄存器的 READOK 标志在每个 AP 读访问或者 RDBUFF 读请求（知道是否 AP 读访问是成功的）时被更新。
- SW-DP 实现了写 buffer（对于 DP 和 AP 写），这甚至当其他操作仍未完成时，接收一个写操作。如果写 buffer 满了，芯片应答响应是“WAIT”。IDCODE 读、CTRL/STAT 读或者 ABORT 写，是例外（甚至当如果写 buffer 是满的）
- 由于 SWCLK 和 HCLK 是异步时钟，在写操作后（校验位之后）需要两个额外的 SWCLK 周期，用来确保写的内部有效性。当驱动信号线为低时，这几个周期应该被应用。

  当为上电请求写 CTRL/STAT 时，以上尤其重要。如果下个操作（需要上电）立即出现，则会 fail。

22.4.5. SW-DP 寄存器

当 ApnDP=0 时，可以访问这些寄存器。

A[3:2]	R/W	CTRLSEL 位或者 SELECT 寄存器	Register	Notes
00	Read		IDCODE	
00	Write		ABORT	
01	Read/Write	0	DP-CTRL/STAT	
01	Read/Write	1	WIRE CONTROL	
10	Read		READ RESEND	
10	Write		SELECT	
11	Read/Write		READ BUFFER	

22.4.6. SW-AP 寄存器

Address	A[3:2] value	Description
0x0	00	Reserved
0x4	01	DP CTRL/STAT 寄存器，用作 ■ 请求一个系统或者调试的 power-up ■ 为 AP 访问配置传输操作 ■ 控制被 pushed 比较和被 pushed 验证操作 ■ 读一些状态标志（溢出、power-up 应答）

Address	A[3:2] value	Description
0x8	10	DP SELECTRION 寄存器：用作选择当前访问端口和 active 4 个 word 的寄存器在窗口。 ■ Bit 31:24：APSEL：选择当前 AP ■ Bit 23:8：reserved ■ Bit 7:4：APBANKSEL：在当前 AP,选择 active 4 个 word 寄存器窗口 ■ Bit 3:0：reserved
0xC	11	DP RDBUFF 寄存器：用于提供调试者在一个操作序列后，得到最终的结果（不用请求新的 JTAG-DP 操作）

22.5. 内核调试

通过 core debug 寄存器，可以访问 Core debug。Debug 访问这些寄存器是通过 debug 访问端口。他由下面四个寄存器组成

表 22-5 内核调试寄存器

寄存器	描述
DHCSR	32bit Debug halting control and status register
DCRSR	17bit Debug Core register selector register
DHCDR	32bit debug Core register Data register
DEMCR	32bit debug exception and monitor control register

这些寄存器不会被系统复位，复位掉。他们进会被上电复位，复位掉。为了在复位时 Hart，需要：

- 调试和例外监视控制寄存器的 bit0（VC_CORRESET），被使能
- 调试停止控制和状态寄存器，被使能

22.6. BPU 断点单元(Break Point Unit)

Cortex-M0+ BPU 实现提供了 4 个断点寄存器。BPU 是一套 ARMv7-M 的 flash 补丁和断点（FPB）Block（Cortex-M3 & Cortex-M4）。

22.6.1. BPU 功能

处理器断点实现基于 PC 的断点功能。

参考 ARMv6-M ARM 和 ARM Coresight Components Technical Reference Manual，以获得更多关于 BPU Coresight 的身份寄存器和他们的地址和访问种类。

22.7. 数据观察点 DWT (Data Watchpoint)

Cortex-M0 DWT 实现提供了 2 个 watchpoint 寄存器。

22.7.1. DWT 功能

处理器的断点实现基于 PC 的断点功能。

22.7.2. DWT 程序计数器样本寄存器

实现数据 watchpoint 单元的处理器，也实现了 ARMv6-M 可选的 DWT Program Counter Sample register(DWT_PCSR)。该寄存器允许调试者周期性的采样 PC，而不用停止处理器。这个机制提供了粗粒度分析。

CORTEX-M0+ DWT_PCSR 记录了通过了条件代码的指令和未通过的指令。

22.8. MCU 调试模块 (DBGMCU)

MCU debug component 帮助调试者提供以下支持：

- 低功耗模式

- 对 timer、watchdog 在 breakpoint 期间的时钟控制

22.8.1. 低功耗模式的调试支持

为进入低功耗模式，要执行 WFI 或者 WFE 指令。MCU 进入低功耗模式，或者是将 CPU Clock 停止掉，或者是减少 CPU 的功耗。

CPU 不允许在 debug 期间，停掉 FCLK 或者 HCLK。由于这些是调试者连接的需要，在一个调试期间，他们必须保持开启。MCU 集成了特殊的方法，允许用户在低功耗模式下调试软件。

因此，调试者主机必须先置某些调试配置寄存器的内容，以改变低功耗行为：

- 在 sleep 模式：FCLK 和 HCLK 仍然有效。相应的，该模式不能引起任何对于标准调试功能的限制。
- 在 stop 模式：DBG_STOP 位必须被调试者提前置位。

22.8.2. 支持定时器、看门狗、bxCAN 和 I2C 的调试

在一个 breakpoint 期间，是有必要选择 timer 的计数器和 watchdog 要怎样的行为：

- 他们可以继续在 breakpoint 里计数。例如，这是当一个 PWM 正在控制电机时通常被需要的。
- 他们可以停下来在 breakpoint 内部计数。这是 watchdog 的特性决定的。

22.9. DBG 寄存器

22.9.1. DBG 设备 ID 代码寄存器(DBG_IDCODE)

Address offset: 0x00

仅支持 32-bit 地址访问，只读。

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

Bit	Name	R/W	Reset Value	Function
31：0	待定义	R		

22.9.2. 调试 MCU 配置寄存器 (DBGMCU_CR)

该寄存器配置在 debug 状态下的 MCU 低功耗模式。

该寄存器会被上电复位进行异步复位（不是系统复位）。它可以在系统复位下被调试者进行写操作。

如果调试者主机不支持该功能，对于软件使用者来说，写这些寄存器仍然是可能的。

Address offset: 0x04

Reset value: 0x0000 0000（不会被系统复位进行复位）

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

Bit	Name	R/W	Reset Value	Function
31：2	Reserved			
1	DBG_STOP	RW	0	Debug stop 模式。

Bit	Name	R/W	Reset Value	Function
				0：（FCLK=off，HCLK=off）。在 STOP 模式，HCLK 和 FCLK 都会关闭。当从 STOP 模式退出时，时钟配置与上电复位后相同（系统时钟为 HSI）。随后，软件需要重新配置时钟控制器。 1：（FCLK=on，HCLK=on）。当进入 STOP 模式，HSI 不会关闭，FCLK 和 HCLK 由 I 产生。当退出 STOP 模式，如果需要改变时钟控制，软件需要重新配置。
0	Reserved			

22.9.3. DBG APB freeze register 1 (DBG_APB_FZ1)

该寄存器用来配置 timer、IWDG 在 debug 下的时钟。该寄存器被上电复位进行异步复位（不是系统复位）。它可以被调试者在系统复位下进行写。

Address offset: 0x08

Power on Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31	DBG_LPTIM_STOP	RW	0	当 CPU 停止时，LPTIM 的计数器时钟控制位 0：使能 1：不使能
30：13	Reserved			
12	DBG_IWDG_STOP	RW	0	当 CPU 停止时，IWDG 计数器的时钟控制位 0：使能 1：不使能
11：1	Reserved			
0	reserved			

22.9.4. DBG APB freeze register 2(DBG_APB_FZ2)

该寄存器用来配置 timer 在 debug 下的时钟控制。该寄存器被上电复位进行异步复位（不是系统复位）。它可以被调试者在系统复位下进行写。

Address offset: 0x0C

Power on Reset value: 0x0000 0000

仅支持 32-bit 地址访问，只读。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	DBG_ TIM16_STOP	Res
														RW	
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res	Res	Res	Res	DBG_ TIM1_STOP	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
				RW											

Bit	Name	R/W	Reset Value	Function
31：18	Reserved			
17	DBG_TIM16_STOP			当 CPU 停止时，TIM16 计数器的时钟控制位

Bit	Name	R/W	Reset Value	Function
				0：使能 1：不使能
16：12	Reserved			
11	DBG_TIM1_STOP			当 CPU 停止时，TIM1 计数器的时钟控制位 0：使能 1：不使能
10：0	Reserved			

22.9.5. DBG 寄存器映像

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	DBG_ID-CODE	TBD	TBD	TBD	TBD	TBD	TBD	TBD	TBD	TBD	TBD	TBD	TBD	TBD	TBD	TBD	TBD	TBD	TBD	TBD	TBD	TBD	TBD	TBD	TBD	TBD	TBD	TBD	TBD	TBD	TBD	TBD	TBD
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x004	DBG_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_STO	Res.
	Reset value																															0	
0x008	DBG_APB_FZ1	DBG_LPTIM_STO	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_IWDG_STO	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0																			0												
0x00C	DBG_APB_FZ2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_TIM16_ST	Res.	Res.	Res.	Res.	Res.	DBG_TIM1_STO	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value															0						0											

23. 版本历史

版本	日期	更新记录
V1.0	2022.10.11	初版
V1.1	2022.10.21	更新功能描述
V1.2	2022.12.20	更新功能描述
V1.3	2023.01.18	更新功能描述
V1.4	2025.08.08	更新USART接收器容忍度 增加LPTIM内部时钟源选择注意事项

Puya Semiconductor Co., Ltd.