PY32T020-B 系列

32 位 ARM® Cortex®-M0+ 微控制器

参考手册

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

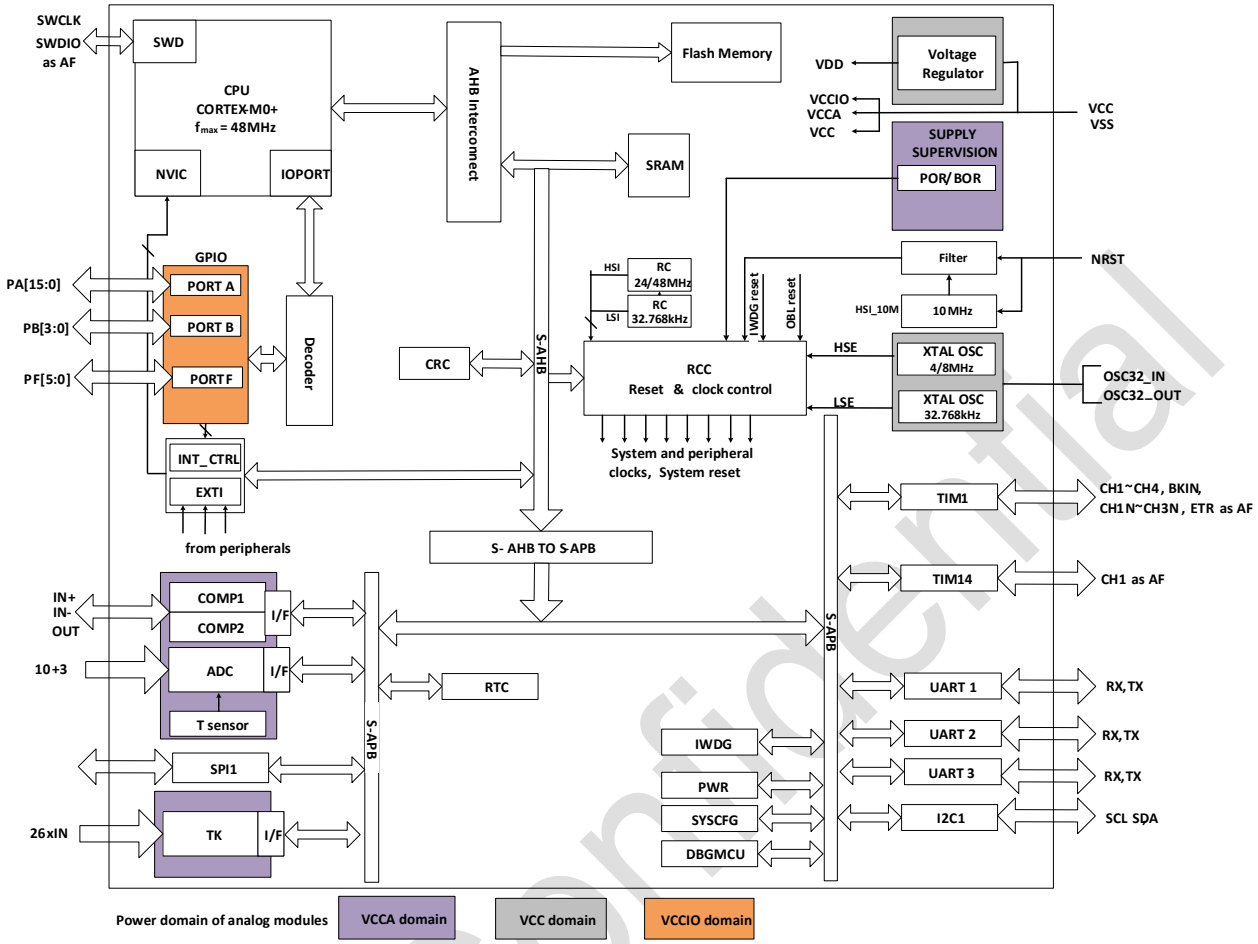


图 2-1 系统架构框图

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
  - 其他AHB slave （包括 AHB-to-APB bridge、CRC、EXTI、RCC）

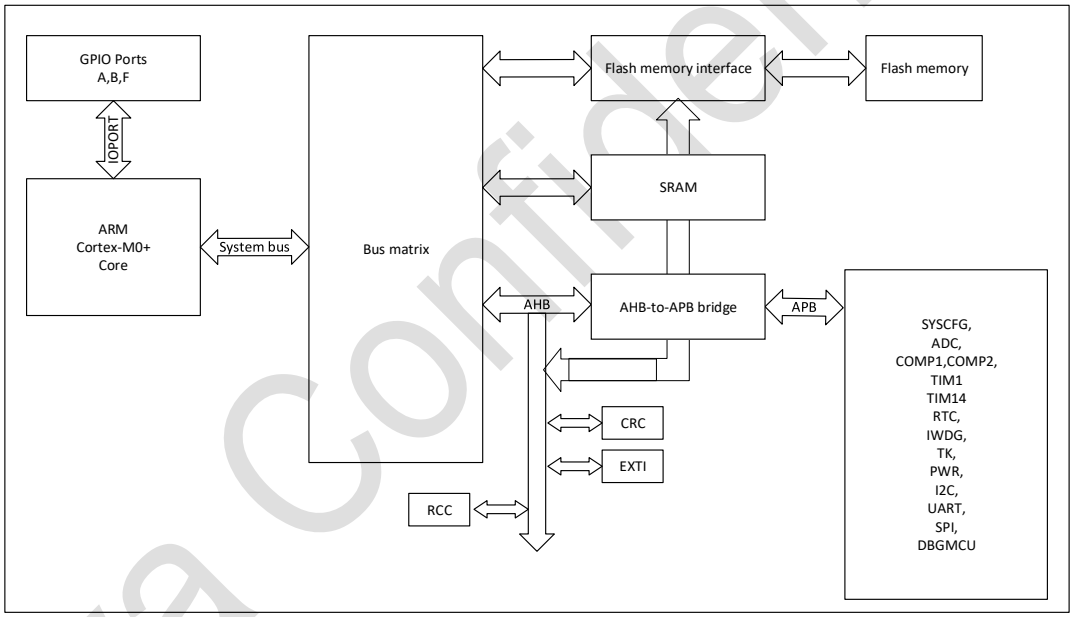


图 3-1 系统架构

- 系统总线

Cortex-M0+ 的系统总线连接到片上 AHB 总线，CPU通过该 AHB 总线访问 Flash、SRAM、CRC、EXTI、RCC 并通过 AHB-to-APB bridge 访问周边外设。

- AHB-to-APB bridge（APB）

AHB-to-APB bridge（APB）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. 存储器结构简介

程序存储器、数据存储器、寄存器和 IO ports 被统一编址在一个线性 4 GB 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 512 MB 的 Block 区域。

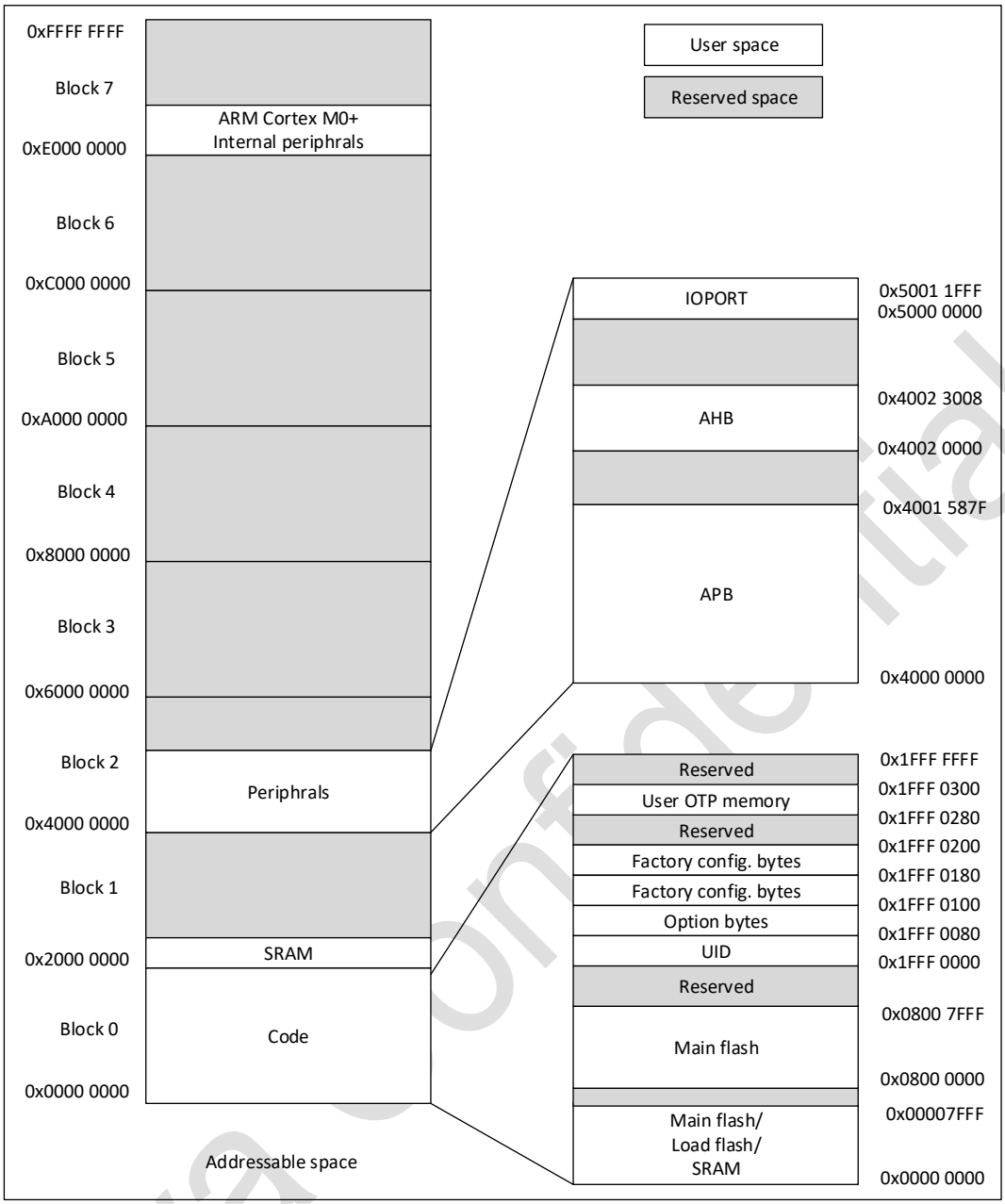


图 3-2 存储器映射

表 3-1 存储器地址

Type	Boundary Address	Size	Memory Area
SRAM	0x2000 1000-0x3FFF FFFF	-	保留
	0x2000 0000-0x2000 0FFF	4 KB	SRAM
Code	0x1FFF 0300-0x1FFF FFFF	-	保留
	0x1FFF 0280-0x1FFF 02FF	128 Bytes	User OTP memory
	0x1FFF 0180-0x1FFF 01FF	128 Bytes	Factory config. bytes
	0x1FFF 0100-0x1FFF 017F	128 Bytes	Factory config. bytes
	0x1FFF 0080-0x1FFF 00FF	128 Bytes	Option bytes
	0x1FFF 0000-0x1FFF 007F	128 Bytes	UID
	0x0800 8000-0x1FFE FFFF	-	保留
	0x0800 0000-0x0800 7FFF	32 KB	Main flash memory
	0x0000 8000-0x07FF FFFF	-	保留
	0x0000 0000-0x0000 7FFF	32 KB	根据 Boot 配置选择: 1. Main flash memory 2. Load flash 3. SRAM

表 3-2 外设寄存器地址

Bus	Boundary Address	Size	Peripheral (1)
	0xE000 0000-0xE00F FFFF	1 MB	M0+
IOPORT	0x5000 1800-0x5FFF FFFF	-	保留(1)
	0x5000 1400-0x5000 17FF	1 KB	GPIOF
	0x5000 0800-0x5000 13FF	-	保留(1)
	0x5000 0400-0x5000 07FF	1 KB	GPIOB
	0x5000 0000-0x5000 03FF	1 KB	GPIOA
AHB	0x4002 3400-0x4FFF FFFF	-	保留
	0x4002 3010-0x4002 33FF	1 KB	保留
	0x4002 3000-0x4002 300F		CRC
	0x4002 2400-0x4002 2FFF	-	保留
	0x4002 2000-0x4002 23FF	1 KB	Flash FMC
	0x4002 1C00-0x4002 1FFF	-	保留
	0x4002 1900-0x4002 1BFF	1 KB	保留
	0x4002 1800-0x4002 18FF		EXTI(2)
	0x4002 1400-0x4002 17FF	-	保留
	0x4002 1080-0x4002 13FF	1 KB	保留
	0x4002 1000-0x4002 107F		RCC(2)
	0x4002 0000-0x4002 0FFF	-	保留
APB	0x4001 5C00-0x4001 FFFF	-	保留
	0x4001 5880-0x4001 5BFF	1 KB	保留
	0x4001 5800-0x4001 587F		DBG
	0x4001 4800-0x4001 57FF	-	保留
	0x4001 4480-0x4001 47FF	1 KB	保留
	0x4001 4400-0x4001 447F		UART2

Bus	Boundary Address	Size	Peripheral (1)
	0x4001 3C00-0x4001 43FF	-	保留
	0x4001 3880-0x4001 3BFF	1 KB	保留
	0x4001 3800-0x4001 387F		UART3
	0x4001 3400-0x4001 37FF	-	保留
	0x4001 3080-0x4001 33FF	1 KB	保留
	0x4001 3000-0x4001 307F		SPI
	0x4001 2C80-0x4001 2FFF	1 KB	保留
	0x4001 2C00-0x4001 2C7F		TIM1
	0x4001 2800-0x4001 2BFF	-	保留
	0x4001 2400-0x4001 27FF	1 KB	ADC
	0x4001 0400-0x4001 23FF	-	保留
	0x4001 0220-0x4001 03FF	1 KB	保留
	0x4001 0200-0x4001 021F		CMP1/2
	0x4001 0000-0x4001 01FF		SYSCFG
	0x4000 7400-0x4000 FFFF	-	保留
	0x4000 7080-0x4000 73FF	1 KB	保留
	0x4000 7000-0x4000 707F		PWR(3)
	0x4000 5800-0x4000 6FFF	-	保留
	0x4000 5480-0x4000 57FF	1 KB	保留
	0x4000 5400-0x4000 547F		I^2C
	0x4000 4800-0x4000 53FF	-	保留
	0x4000 4480-0x4000 47FF	1 KB	保留
	0x4000 4400-0x4000 447F		UART1
	0x4000 3C00-0x4000 43FF	-	保留
	0x4000 3880-0x4000 3BFF	1 KB	保留
	0x4000 3800-0x4000 387F		TK
	0x4000 3400-0x4000 37FF	-	保留
	0x4000 3080-0x4000 33FF	1 KB	保留
	0x4000 3000-0x4000 307F		IWDG
	0x4000 2C00-0x4000 2FFF	-	保留
	0x4000 2880-0x4000 2BFF	1 KB	保留
	0x4000 2800-0x4000 287F		RTC
	0x4000 2400-0x4000 27FF	-	保留
	0x4000 2080-0x4000 23FF	1 KB	保留
	0x4000 2000-0x4000 207F		TIM14
	0x4000 0000-0x4000 1FFF	-	保留

1. 上表标注为保留的地址空间，无法写操作，读回为0，且产生hardfault。
2. 不仅支持32位 word访问，还支持half-word和Bytes访问。
3. 不仅支持32位 word访问，还支持half-word访问。

3.3. 嵌入式 SRAM

片内集成 4 KB SRAM。通过 bytes、half-word（16-bit）或者 word（32-bit）的方式可访问 SRAM。

3.4. Flash 存储器

Flash 存储器有两个不同的物理区域组成：

- Main flash block：32 KB（8K x 32 bits），用于存储用户程序和用户数据，另外可以根据客户配置可以设定最大 4 KB 作为 User bootloader 使用。
- Information 区域，0.75 KB(192 x 32 bits)，它包括以下部分：
  - Factory config. Bytes 0，128 Bytes，用于存放：
    存放Trimming 数据（含HSI triming数据）
  - Factory config. Bytes 1：128 Bytes，用于存放：
    芯片Trimming配置信息
  - UID：128 Bytes，用于存放芯片的UID
  - Option byte：128 Bytes，用于存放芯片硬件和存储保护的配置值
  - OTP Flash：128 Bytes，用于存放用户数据
    对 Flash main memory 的保护包括以下几种机制：
- 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)。写保护的最
- 小保护单位为 4 KB。
- Option byte 写保护，专门的解锁设计。
- SDK 保护。

3.5. 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	Boot memory size ==0	Boot memory size !=0
X	0	Main flash 启动	Main flash 启动
0	1	SRAM 启动	SRAM 启动
1	1	N/A	Load flash 启动

启动模式配置在系统复位后的第 4 个 SYSCLK 进行锁存。由用户按照上表决定选择哪种启动模式。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 user bootloader：user bootloader memory 对齐在启动存储器空间 0x0000 0000，但是仍然可以根据 user bootloader 大小设置从下列地址空间访问到。

表 3-4 Boot 配置

User Bootloader	访问地址
无	-
1 KB	0x0800 7C00~0x0800 7FFF
2 KB	0x0800 7800~0x0800 7FFF
3 KB	0x0800 7400~0x0800 7FFF
4 KB	0x0800 7000~0x0800 7FFF

- 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。

3.5.1. 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 SYSCFG 配置寄存器 1(SYSCFG_CFGR1)的 MEM_MODE 位选择决定。

4. 嵌入式 Flash

4.1. 闪存主要特性

Flash 存储器单位页大小为 128 Bytes，单位扇区大小为 4 KB。

Flash 存储器有两个不同的物理区域组成：

- Main flash block：32 KB（8K x 32 bits），用于存储用户程序和用户数据,另外可以根据客户配置可以设定最大 4 KB 作为 User bootloader 使用。
- Information 区域，0.75 KB(192 x 32 bits)

对 Flash main memory 的保护包括以下几种机制：

- 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)。写保护的最
- 小保护单位为 4 KB。
- Option byte 写保护，专门的解锁设计。
- SDK 保护。

4.2. 闪存功能介绍

4.2.1. 闪存结构

Flash 存储器由 32 位宽的存储单元组成，可以用作程序和数据的存储，page size 为 128 Bytes，sector size 为 4 KB。从功能上，Flash 存储器分为 Main flash 和 Information flash，前者容量是 32 KB，后者容量为 0.75 KB。

Page erase 和 sector erase 操作可以应用于 Main flash 中未被写保护的区域。

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 3-4 闪存结构及边界地址

Block	Sector	Page	Base address	Size
Main memory	Sector 0	Page 0-31	0x0800 0000-0x0800 0FFF	4 KB
	Sector 1	Page 32-63	0x0800 1000-0x0800 1FFF	4 KB
	Sector 2	Page 64-95	0x0800 2000-0x0800 2FFF	4 KB
	Sector 3	Page 96-127	0x0800 3000-0x0800 3FFF	4 KB
	Sector 4	Page128-159	0x0800 4000-0x0800 4FFF	4 KB
	Sector 5	Page160-191	0x0800 5000-0x0800 5FFF	4 KB
	Sector 6	Page192-223	0x0800 6000-0x0800 6FFF	4 KB
	Sector 7	Page224-255	0x0800 7000-0x0800 7FFF	4 KB
UID	-	Page 0	0x1FFF 0000-0x1FFF 007C	128 Bytes
option bytes	-	Page 1	0x1FFF 0080-0x1FFF 00FC	128 Bytes
Factory config0	-	Page 2	0x1FFF 0100-0x1FFF 017C	128 Bytes
Factory config1	-	Page 3	0x1FFF 0180-0x1FFF 01FC	128 Bytes

保留	-	Page 4	0x1FFF 0200-0x1FFF 027C	128 Bytes
USER OTP memory	-	Page 5	0x1FFF 0280-0x1FFF 02FC	128 Bytes

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。

取指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。

FLASH_ACR.0(LATENCY)位，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；当为 2，flash 读操作增加 3 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 进行写操作。

ICP：用来更新整个 Flash 存储器的内容，可以使用 SWD 协议或者 Bootloader，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，下载要写的数据到 flash 中。IAP 允许用户在应用运行时，再次写 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 Flash 写或者擦操作时，发生了复位，则 Flash 存储器的内容是不被保护的。

在写或者擦操作期间，任何读 Flash 的操作都会拖延总线。写或擦操作一结束，读操作就可以正确的进行。这也就意味着，当正在进行写或擦操作时，不能进行代码和数据的读取。

对于写和擦操作，必须打开 HSI（软件根据 HSI 频率配置对应的参数到擦写时间控制寄存器）。

通过以下 Flash 控制接口相关的寄存器，可以实现写和擦操作：

- 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 SDK address register (FLASH_SDKR)
- Flash boot control register（FLASH_BTCR）
- Flash write protection resister (FLASH_WRPR)
- FLASH sleep time config register (FLASH_STCR)
- 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（除了用作重新加载选项字节的 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，产生解锁时序，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 BSY 位被清零。

4.2.3.2. 闪存写操作

Flash 存储器每次以 32 位字为单位（进行半字或者字节操作会产生 hardfault）进行整个 page 的写操作。当 FLASH_CR 寄存器的 PG 位被置位，CPU 向 FLASH 存储器地址空间写 32 位数据时，写操作开始启动。任何非 32 位的写入将导致 hard fault 中断。

如果要写的 Flash 地址空间，是被 FLASH_WRPR 寄存器设置为保护的区域，则写操作会被忽略掉，同时 FLASH_SR 寄存器 WRPERR 位会被置位。另外，部分 Main flash 区域作为 Load flash 使用时，如果被选择为要 program 的区域，则 program 操作会被忽略掉，同时 FLASH_SR 寄存器 WRPERR 位也会被置位。写操作结束，FLASH_SR 寄存器的 EOP 位会被置位。

具体 Flash 写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 擦或者写操作，则软件读出该 Page 的 32 个字（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 个字的写操作（只接受 32 位的写）
6. 置位 FLASH_CR 寄存器的 PGSTRT
7. 写第 32 个字
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（当写操作已经成功，该位被置位），然后软件清零该位
10. 如果不再有写操作，则软件清除 PG 位

当上述步骤 7）成功执行，则写操作自动启动，同时 BSY 位被硬件置位。

4.2.3.3. 闪存擦除操作

Flash 存储器可以按照 page 进行擦操作，或者进行 sector 和 mass erase。

注：sector 和 mass erase 对 information memory 不起作用。

Page erase

当某个 page 被 WRP 保护，它是不会被擦的，此时 WRPERR 位被置位。另外，部分 Main flash 区域作为 Load flash 使用时，被选择的 page 不会被 erase 的，此时 WRPERR 位也会被置位。

当要进行 Page erase 操作时，要进行以下步骤：

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

Mass erase

Mass erase 用来对整片 Main flash 进行擦操作，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。

注：部分 Main flash 区域作为 Load flash 使用时，Mass erase 功能无效，不会产生 Mass erase 操作，并且 WRPERR 位也会被置位。

进行 Mass erase 的步骤如下：

- 检查 BSY 位，确认是否没有正在进行的 Flash 操作
- 向 FLASH_KEYR 寄存器依次写 KEY1,KEY2，解除 FLASH_CR 寄存器保护
- 置位 FLASH_CR 寄存器的 MER 位和 EOPIE 位
- 向 Flash 的任意 Main flash 空间写任意数据（32 位数据）
- 等待 BSY 位被清零
- 检查 EOP 标志位被置位
- 清零 EOP 标志

Sector erase

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

注：另外，部分 Main flash 区域作为 Load flash 使用时，如果选择 sector7 作为 erase 对象，sector7 不会被 erase 的，此时 WRPERR 位也会被置位。

进行 sector erase 的步骤如下：

- 检查 BSY 位，确认是否没有正在进行的 Flash 操作

- 向 FLASH_KEYR 寄存器依次写 KEY1、KEY2，解除 FLASH_CR 寄存器保护
- 置位 FLASH_CR 寄存器的 SER 位和 EOPIE 位
- 向该 sector 写任意数据
- 等待 BSY 位被清零
- 检查 EOP 标志位被置位
- 清零 EOP 标志

4.2.3.4. 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 进行写和擦的操作，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 写和擦时间控制寄存器进行正确的配置。

4.2.4. 闪存选项字节

4.2.5. Flash 选项字节描述

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 3-5 选项字节格式

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

选项字节的内容可以表 3-6 选项字节结构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 boot control 寄存器（FLASH_BTCR）
- FLASH WRP address 寄存器（FLASH_WRPR）

表 3-6 选项字节结构

地址	描述
0x1FFF 0080	闪存用户选项字节及其反码
0x1FFF 0084	BOR 配置及其反码
0x1FFF 0088	闪存 SDK 地址区域的选项字节及其反码
0x1FFF 008C	闪存 WRP 地址的选项字节及其反码
0x1FFF 0090	保留
0x1FFF 0094	保留
…	保留
…	保留
…	保留

0x1FFF 00FC	保留

4.2.5.1. Flash 用户选项的选项字节

Flash 地址：0x1FFF 0080

生产值：0x6D55 92AA

在上电复位（POR/BOR/OBL_LAUNCH）释放后,从 flash information memory 的选项字节区域读出相应的值，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
~ IWDG_STOP	~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
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
IWDG_STOP	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	~ IWDG_STOP	R	IWDG_STOP 的反码
30	~NRST_MODE	R	NRST_MODE 的反码
29	保留		
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	IWDG_STOP	R	设置 IWDG 在 stop 模式下定时器运行状态 0： 冻结定时器 1： 正常运行
14	NRST_MODE	R	0：复位 pin 作为 NRST 使用 1：复位 pin 作为普通 GPIO 使用
13	保留		-
12	IWDG_SW	R	0：硬件看门狗 1：软件看门狗
11：9	BOR_LEV[2:0]	R	001：BOR 上升阈值为 1.99 V，下降阈值位 1.88 V 010：BOR 上升阈值为 2.19 V，下降阈值位 2.10 V 011：BOR 上升阈值为 2.39 V，下降阈值位 2.30 V 100：BOR 上升阈值为 2.78 V，下降阈值位 2.69 V 101：BOR 上升阈值为 3.08 V，下降阈值位 2.99 V 110：BOR 上升阈值为 3.68 V，下降阈值位 3.58 V 111：BOR 上升阈值为 4.20 V，下降阈值位 4.08 V
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, 读保护不使能 非 0xAA：level 1, 读保护使能

4.2.5.2. Flash SDK 区域地址的选项字节

Flash 地址：0x1FFF 0084

生产值：0xFFF0 000F

在上电复位（POR/BOR/OBL_LAUNCH）释放后,从 flash information memory 的选项字节区域读出相应的值，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.	~SDK_END[3:0]				Res.	Res.	Res.	Res.	~SDK_STRT[3:0]			
				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.	SDK_END[3:0]				Res.	Res.	Res.	Res.	SDK_STRT[3:0]			
				R	R	R	R					R	R	R	R

Bit	Name	R/W	Function
31：28	保留		
27：24	~SDK_END[3:0]	R	SDK_END 的反码
23：20	保留		
19：16	~SDK_STRT[3:0]	R	SDK_STRT 的反码
15：12	保留		
11：8	SDK_END[3:0]	R	SDK area end address，每一位对应的 STEP 为 2 KB
7：4	保留		
3：0	SDK_STRT[3:0]	R	SDK area start address，每一位对应的 STEP 为 2 KB

4.2.5.3. 控制闪存启动的选项字节(Option byte for Flash boot control)

Flash 地址：0x1FFF 0088

生产值：0xFFFF 0000

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	~BOOT0	Res.	Res.	Res.	Res.	~SWD_MODE[1:0]		Res.	Res.	Res.	Res.	Res.	~BOOT_SIZE [2:0]		
R	R					R	R						R	R	R
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
nBOOT1	BOOT0.	Res.	Res.	Res.	Res.	SWD_MODE[1:0]		Res.	Res.	Res.	Res.	Res.	BOOT_SIZE [2:0]		
R	R					R	R						R	R	R

Bit	Name	R/W	Function
31	~nBOOT1	R	nBOOT1 的反码
30	~BOOT0	R	BOOT0 的反码
29：26	保留		
25:24	~SWD_MODE[1:0]	R	SWD_MODE[1:0]的反码
23：19	保留		
18：16	~BOOT_SIZE [2:0]	R	BOOT_SIZE 的反码
15	nBOOT1	R	nBOOT1，BOOT0 选择芯片启动模式
14	BOOT0	R	X0：MainFlash 启动 11：Load flash 启动 01：SRAM 启动
13：10	保留		
9:8	SWD_MODE[1:0]	R	选择 IO 口作为 GPIO 模式还是 SWD 模式 PF3 PF4 PA13 PA14 00：SWCLK SWDIO GPIO GPIO

			01：GPIO GPIO SWDIO SWCLK 10：GPIO SWDIO GPIO SWCLK 11：SWCLK GPIO SWDIO GPIO
7:3	保留		
2：0	BOOT_SIZE [2:0]	R	选择 Main flash 部分区域作为 Load flash 区使用 000：无 Load flash 区 001：1 KB（0x0800 7C00~0x0800 7FFF） 010：2 KB（0x0800 7800~0x0800 7FFF） 011：3 KB（0x0800 7400~0x0800 7FFF） 1xx：4 KB（0x0800 7000~0x0800 7FFF）

4.2.5.4. Flash WRP 地址的选项字节

Flash 地址：0x1FFF 008C

生产值：0xFF00 00FF

在上电复位（POR/BOR/OBL_LAUNCH）释放后,从 Flash information memory 的选项字节区域读出相应的值，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.	~WRP[7:0]							
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRP[7:0]							
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Bit	Name	R/W	Function
31：24	保留	-	-
23：16	~WRP	R	WRP 的反码
15：8	保留	-	-
7：0	WRP	R	0：sector[y]被保护 1：sector[y]无保护 y=0~7

4.2.6. 写 Flash 选项字节

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（注意：每次擦或写都要如此操作）。

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 KEY 时序错误时，会产生总线错误状态，并产生 Hard Fault 中断。

User option（information flash 的选项字节）可以通过软件写 FLASH_CR 寄存器的 OPTLOCK 位，被保护住，以防止不想要的擦/写操作。

如果软件置位 Lock 位，则 OPTLOCK 位也被自动置位。

修改用户的选项字节

选项字节的写操作，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. 向选项字节寄存器FLASH_OPTR/FLASH_SDKR/FLASH_WRPR写期望的值（1~4个字）（若非字对齐写入，则忽略该操作）
4. 置位OPTSTRT位
5. 向0x4002 2080写任意32位数据（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

任何对选项字节的改动，硬件都会先把选项字节对应的整个 page 擦掉，然后用 FLASH_OPTR、FLASH_SDKR 或者 FLASH_WRPR 寄存器的值，写到选项字节中。并且，硬件自动计算相应的反码，并把计算值写到选项字节的相应区域。

重新加载字节选项

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

选项字节的装载，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/PDR/BOR）

“装载选项字节”进行的操作是：对 information memory 区域的选项字节进行读操作，再把读出的数据存储在内部 option 寄存器中（FLASH_OPTR、FLASH_SDKR 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，这样选项字节的装载，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。在选项字节装载期间，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，则选项字节被复制到 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。option 寄存器维持默认值：

- 对于用户选项
  - 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[3:0]= 0x0，SDKR_END[3:0]=0xF，即所有 flash 空间都被设定为 SDK
- 对于 FLASH boot control option
  - nBOOT1，BOOT0位写成00（即选择 Main flash 作为启动区）
  - BOOT_SIZE位写成0（即无Load flash区域）
- 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 copy 到下面的 option 寄存器（软件可读可写）：
  - FLASH_OPTR
  - FLASH_SDKR
  - FLASH_BTCR
  - FLASH_WRPR

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

4.3. Flash 配置字节

芯片内的 Flash 的 information 区域的部分区间（共 2 个 page）作为 Factory config. byte 使用。

配置字节存放供软件读取信息：

- 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
- LSI 不同频率对应的 Trimming 值
- V_{REFBUF} 不同输出电压对应的 Trimming 值

表 3-7 配置字节信息

Page	Word	Address	描述
2	0	0x1FFF 0100	存放 HSI 24 MHz 频率选择控制及对应的 Trimming 值
	1	0x1FFF 0104	存放 HSI 48 MHz 频率选择控制及对应的 Trimming 值
	2	0x1FFF 0108	保留
	3	0x1FFF 010C	保留
	4	0x1FFF 0110	保留
	5	0x1FFF 0114	Normal TS Data
	6	0x1FFF 0118	High TS Data
	7	0x1FFF 011C	存放 HSI 24 MHz 频率下对应的 FLASH_TS0、FLASH_TS1、FLASH_TS3 寄存器的配置值
	8	0x1FFF 0120	存放 HSI 24 MHz 频率下对应的 FLASH_TS2P、FLASH_TPS3 寄存器的配置值
	9	0x1FFF 0124	存放 HSI 24 MHz 频率下对应的 FLASH_PERTPE 寄存器的配置值
	10	0x1FFF 0128	存放 HSI 24 MHz 频率下对应的 FLASH_SMERTPE 寄存器的配置值
	11	0x1FFF 012C	存放 HSI 24 MHz 频率下对应的 FLASH_PRGTPE、FLASH_PRETPE 寄存器的配置值
	12	0x1FFF 0130	存放 HSI 48 MHz 频率下对应的 FLASH_TS0、FLASH_TS1、FLASH_TS3 寄存器的配置值
	13	0x1FFF 0134	存放 HSI 48 MHz 频率下对应的 FLASH_TS2P、FLASH_TPS3 寄存器的配置值
	14	0x1FFF 0138	存放 HSI 48 MHz 频率下对应的 FLASH_PERTPE 寄存器的配置值
	15	0x1FFF 013C	存放 HSI 48 MHz 频率下对应的 FLASH_SMERTPE 寄存器的配置值

	16	0x1FFF 0140	存放 HSI 48 MHz 频率下对应的 FLASH_PRGTPE、FLASH_PRETPE 寄存器的配置值
	17	0x1FFF 0144	存放 LSI 32.768 kHz 频率对应的 Trimming 值
	18	0x1FFF 0148-0x1FFF 017C	保留
3	0	0x1FFF 0180-0x1FFF 0FFF	保留

4.3.1. HSI_TRIMMING_FOR_USER

Address: 0x1FFF 0100(24MHz)/0x1FFF 0104(48MHz)

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**
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。

4.3.2. FLASH_SLEEPTIME_CONFIG

Address: 0x1FFF 0114

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**
FLASH_SLEEPTIME[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
R	R	R	R	R	R	R	R								

软件需要从该地址读出数据，再写入 FLASH_STCR 寄存器对应的[15:8]。

4.3.3. HSI_24M/48M_EPPARA0

Address: 0x1FFF 011C(24 MHz)、0x1FFF 0130(48 MHz)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	TS1[9:0]										TS3[8:7]	
				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**
TS3[6:0]							TS0[8:0]								
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。

4.3.4. HSI_24M/48M_EPPARA1

Address: 0x1FFF 0120(24 MHz)、0x1FFF 0134(48 MHz)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	TPS3[11:0]											
				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**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	TS2P[8:0]								
							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_TS2P、FLASH_TPS3 寄存器，以实现对应 HSI 频率所需的擦写时间的配置。

4.3.5. HSI_24M/48M_EPPARA2

Address: 0x1FFF 0124(24 MHz)、0x1FFF 0138(48 MHz)

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	PERTPE[17:16]	
														R	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。

4.3.6. HSI_24M/48M_EPPARA3

Address: 0x1FFF 0128(24 MHz)、0x1FFF 013C(48 MHz)

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.	SMERTPE[17:16]	
														R	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。

4.3.7. HSI_24M/48M_EPPARA4

Address: 0x1FFF 012C(24 MHz)、0x1FFF 0140(48 MHz)

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	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。

4.3.8. LSI_32.768K_TRIMMING

Address: 0x1FFF 0144（32.768 kHz）

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	LSI_TRIM[8:0]								
							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 LSI_TRIM[8:0]，以实现 LSI 频率的更改。

4.4. Flash USER OTP memory bytes

芯片内的 Flash 的 information 区域的部分区间作为 Flash USER OTP memory bytes。

表 3-8 USER OTP memory bytes organization

Page	Word	Address	Contents
5	0	0x1FFF 0280	Bit[31:16]:存放用户数据 Bit[15:0]: USER OTP MEMORY_LOCK
	1	0x1FFF 0284	存放用户数据
	2	0x1FFF 0288	存放用户数据
	…	…	存放用户数据
	…	…	存放用户数据
	…	…	存放用户数据
	31	0x1FFF 02FC	存放用户数据

Page 配置在 information 区域，对本 Page 区域 program 和擦是按照 Main flash 的方法来处理。另外，Main flash 区域的 mass erase 对本区域无效。

设定 USER OTP MEMORY_LOCK 内容不会立刻更新，直到上电复位（POR/BOR/PDR），从会起到保护功能。

对本 Page Write 有如下保护。

表 3-9 Flash USER OTP memory bytes write protection status

USER OTP MEMORY_LOCK	Write protection
0xAA55	读：可以 program 和擦操作：不可以
除(0xAA55)之外的任何值	读、program 和擦操作：可以

4.5. 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，大小是 2 KB。
- 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 4 KB。
- 选项字节写保护，专门的解锁设计。

4.5.1. 闪存软件开发包(SDK)区域保护

对 Flash main memory 的保护包括以下几种机制：

起始地址

Flash memory base address + SDK_STRT[3:0] x 0x800(included)

结束地址

Flash memory base address + (SDK_END[3:0]+1) x 0x800(excluded)

当 SDK 保护有效时，不能使用 mer 进行 mass erase，若触发地址位于 sdk 区，则产生 hardfault，若触发地址不处于 SDK 区，则会置位 wrperr 错误标志位；当使用 ser 进行 sector erase 时，若触发地址位于 SDK 区，则产生 hardfault，若触发地址不处于 SDK 区，但所在 sector 中包含 sdk 区，则会置位 wrperr 错误标志位。

在保护生效状态下，对FLASH_SDKR寄存器解除保护时（写SDK_STRT[3:0]大于SDK_END[3:0]），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 option byte 中的 SDK option 的值（此时更新的值是 SDK 保护无效）。SDK protection 改写产生的 mass erase 也会把 Load flash 区域擦除掉。

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 option byte 中的 SDK option 装载到寄存器中。

4.5.2. Flash 读保护

通过设置 RDP 选项字节，并进行 POR/PDR/BOR 或者 OBL 复位装载新的 RDP 选项字节，可以激活读保护功能。RDP 保护 flash main memory。

如果通过 SWD 的 debug 仍在连接时，要设置读保护，需要进行上电复位而不是其他系统复位。

当 RDP 选项字节和补码成对正确存在于选项字节时，Flash main memory 会被保护。

表 3-10 Flash 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

Level 0: 无保护

对 Main flash 的读、写和擦操作是可能的，对选项字节也是可以进行任何操作。

Level 1: 读保护

当选项字节里的 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（从 Main flash 启动），可以对 Main flash、option byte 进行所有操作。
- Debug, 从 SRAM 启动:在 debug 模式，或者当从 SRAM 启动，Main flash 是不能被读访问的。

  在这些模式下，对 Main flash 读或者写访问产生一个 bus error，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 3-11 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) Boot From Load flash						Debug/ excuted From RAM		
			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
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

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 禁用，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 执行程序包括两种情况：一个是通设置 Boot 启动，另一个是从别的存储器 boot，程序跳转到 SRAM。

4.5.3. Flash 写保护（WRP）

Flash 可以被设置成写保护，以应对不想要的写操作。定义 WRP 寄存器每一位的控制粒度为 4 KB 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，则不允许进行擦或者写操作。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擦或者写操作，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。

4.5.4. Load flash 区域保护

当 Load flash 有效时，对被选择的区域进行擦写操作会被忽略掉，同时 FLASH_CR 寄存器 WRPRTERR 位也会被置位。

修改 FLASH_BTCR 的 BOOT_SIZE 设定硬件会对 Main flash 进行 mass erase 操作。改写产生的 mass erase 也会把 Load flash 区域擦除掉。

4.5.5. 选项字节写保护

缺省情况下，选项字节是可读，并进行写保护的。为获得对选项字节的擦或者写访问，需要向 OPTKEYR 寄存器写入正确的序列。

4.6. 闪存中断

表 3-12 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：

- 解锁 Flash memory 的 FLASH_CR 寄存器的序列错误
- 解锁 Flash 选项字节的写操作序列错误
- Flash 写操作未进行 32 位数据的对齐
- Flash 擦（含 page erase、sector erase 和 mass erase）操作未进行 32 位数据对齐
- 对选项字节寄存器的写操作未进行 32 位数据的对齐

4.7. 闪存寄存器描述

4.7.1. FLASH 访问控制寄存器 (FLASH_ACR)

偏移地址： 0x00

复位值： 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
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.	LATENCY[1:0]
														RW

Bit	Name	R/W	Reset Value	Function
31：2	保留	-	-	-
1:0	LATENCY	RW	0	Flash 读操作对应的等待状态： 00：Flash 读操作没有等待状态（AHB 时钟在 24 MHz 及以下） 01：Flash 读操作有 1 个等待状态，即每次读 Flash 需要两个 AHB 时钟周期（AHB 时钟在 48 MHz） 10：保留 11：保留

4.7.2. FLASH 密钥寄存器 (FLASH_KEYR)

偏移地址：0x08

复位值： 0x0000 0000

所有寄存器位是只写，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	32’h0	下面的值必须被连续的写入，才能解锁 FLASH_CR 寄存器，并使能了 Flash 的 program/erase 操作 KEY1: 0x4567 0123 KEY2: 0xCDEF 89AB

4.7.3. FLASH 选项密钥寄存器 (FLASH_OPTKEYR)

偏移地址： 0x0C

复位值： 0x0000 0000

所有寄存器位是只写，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	32’h0	下面的值必须被连续的写入，才能解锁 Flash 的 option 寄存器，并使能了 option byte 的 program/erase 操作 KEY1: 0x0819 2A3B KEY2: 0x4C5D 6E7F

4.7.4. Flash 状态寄存器 (FLASH_SR)

偏移地址： 0x10

复位值： 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31：17	保留			
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，参考 section Flash option byte programming。 软件写 1，清零。
14	保留			
13	USRLOCK	R	0	根据上电读 userdata 区域第一个 word 的低 16 位的值来指示 userdata 区域是否可写 0：读到的值不为 0xaa55， userdata 可写 1：读到的值为 0xaa55， userdata 不可写 该位仅能被上电复位所复位
12：5	保留			
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	保留			
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。

4.7.5. FLASH 控制寄存器(FLASH_CR)

偏移地址： 0x14

复位值： 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	0	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。当成功给出解锁时序后，该位被硬件清零，解锁了 FLASH_CR 寄存器。 【软件要在 program/erase 操作完成后，置位该位】 当不成功的解锁时序给出，该位仍然保持置位状态，直到下一次系统复位。
30	OPTLOCK	RS	0	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 有关的位被锁定住。当成功给出解锁时序后，该位被硬件清零，解锁了 FLASH_CR 寄存器。 【软件要在 program/erase 操作完成后，置位该位】 当不成功的解锁时序给出，该位仍然保持置位状态，直到下一次系统复位。
29：28	保留			
27	OBL_LAUNCH	RC_W1	0	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。
26	保留			
25	ERRIE	RW	0	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	0	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
23：20	保留			
19	PGSTRT	RW	0	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	保留			
17	OPTSTRT	RW	0	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 128 Bytes 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	保留			
11	SER	RW	0	4 KB 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： Sector erase 不会对 Flash information memory 起作用。 Sector erase 对设定为 WRP 的区域不起作用。
10：3	保留			
2	MER	RW	0	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	0	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	0	Program 操作 0：未选择 Flash 的 program 操作 1：选择 Flash 的 program 操作

4.7.6. FLASH 选项寄存器 (FLASH_OPTR)

偏移地址： 0x20

复位值： 0x0000 92AA。

在上电复位（POR/BOR/OBL_LAUNCH）释放后,从 Flash information memory 的选项字节区域读出相应的值，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
IWDG_STOP	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	保留			
15	IWDG_STOP	RW		设置 IWDG 在 stop 模式下定时器运行状态 0： 冻结定时器 1： 正常运行
14	NRST_MODE	RW		
13	保留			
12	IWDG_SW	RW		0：硬件 watchdog 1：软件 watchdog
11：9	BOR_LEV[2:0]	RW		001：BOR 上升阈值为 1.99 V，下降阈值位 1.88 V 010：BOR 上升阈值为 2.19 V，下降阈值位 2.10 V 011：BOR 上升阈值为 2.39 V，下降阈值位 2.30 V 100：BOR 上升阈值为 2.78 V，下降阈值位 2.69 V

				101：BOR 上升阈值为 3.08 V，下降阈值位 2.99 V 110：BOR 上升阈值为 3.68 V，下降阈值位 3.58 V 111：BOR 上升阈值为 4.20 V，下降阈值位 4.08 V
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

4.7.7. Flash SDK 地址寄存器 (FLASH_SDKR)

偏移地址： 0x24

复位值： 0x0000 000F

在上电复位（POR/BOR/OBL_LAUNCH）释放后,从 Flash information memory 的选项字节区域读出相应的值，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.	SA_END[3:0]				Res.	Res.	Res.	Res.	SA_STRT[3:0]			
				RW	RW	RW	RW					RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31：12	保留			
11：8	SDK_END[3:0]	RW		SDK area end addres，每一位对应的 STEP 为 2 KB
7：4	保留			
3：0	SDK_STRT[3:0]	RW		SDK area start address，每一位对应的 STEP 为 2 KB

4.7.8. Flash boot control （FLASH_BTCR）

偏移地址： 0x28

复位值： 0x0000 0000

在上电复位（POR/BOR/OBL_LAUNCH）释放后,从 Flash information memory 的选项字节区域读出相应的值，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	BOOT0	Res.	Res.	Res.	Res.	SWD_MODE[1:0]		Res.	Res.	Res.	Res.	Res.	BOOT_SIZE [2:0]		
RW	RW					RW	RW						RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31：16	保留			
15	NBOOT1	RW		nBOOT1，BOOT0 选择芯片启动模式 X0：MainFlash 启动 11：Load flash 启动 01：SRAM 启动
14	BOOT0			
13:10	保留			

9:8	SWD_MODE[1:0]	RW		选择 IO 口作为 gpio 模式还是 swd 模式 PF3 PF4 PA13 PA14 00：SWCLK SWDIO GPIO GPIO 01：GPIO GPIO SWDIO SWCLK 10：GPIO SWDIO GPIO SWCLK 11：SWCLK GPIO SWDIO GPIO
7:3	保留			
2：0	BOOT_SIZE [2:0]	RW		选择 Main flash block 部分区域作为 Load flash 区使用 000：无 Load flash 区 001：1 KB（0x0800 7C00~0x0800 7FFF） 010：2 KB（0x0800 7800~0x0800 7FFF） 011：3 KB（0x0800 7400~0x0800 7FFF） 1xx：4 KB（0x0800 7000~0x0800 7FFF）

4.7.9. FLASH 写保护地址寄存器 (FLASH_WRPR)

偏移地址： 0x2C

复位值： 0x0000 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.	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.	WRP[7：0]							
								RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:8	保留	-	-	保留
7	WRP[7]	RW	1	0：sector 7，有写保护，不允许进行 program 和 erase 1：sector 7，无写保护
6	WRP[6]	RW	1	0：sector 6，有写保护，不允许进行 program 和 erase 1：sector 6，无写保护
5	WRP[5]	RW	1	0：sector 5，有写保护，不允许进行 program 和 erase 1：sector 5，无写保护
4	WRP[4]	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[3]	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[2]	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[1]	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[0]	RW	1	0：sector 0，有写保护，不允许进行 program 和 erase

Bit	Name	R/W	Reset Value	Function
				1：sector 0，无写保护

4.7.10. FLASH 睡眠时间配置寄存器（FLASH_STCR)

偏移地址： 0x90

复位值： 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								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
31:16	保留	-	-	保留
15:8	SLEEP_TIME	RW	0x50	当系统时钟选择 LSI 或者 LSE 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 或者 LSE 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 注: 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 0x1E。
7:1	保留	-	-	保留
0	SLEEP_EN	RW	0	Flash 睡眠模式使能 1：Flash 睡眠使能 0：Flash 睡眠关闭

4.7.11. Flash TS0 寄存器(FLASH_TS0)

偏移地址： 0x100

复位值： 0x0000 00XX

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.	TS0								
							RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:9	保留	-	-	保留
8:0	TS0	RW	-	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。

Bit	Name	R/W	Reset Value	Function
				保存在 Flash 的如下地址内： 24 MHz 校准值存放地址：0x1FFF 011C 48 MHz 校准值存放地址：0x1FFF 0130

4.7.12. Flash TS1 寄存器(FLASH_TS1)

偏移地址： 0x104

复位值： 0x0000 01XX

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.	TS1									
						RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:10	保留	-	-	保留
9:0	TS1	RW	0x1XX	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： 24 MHz 校准值存放地址：0x1FFF 011C 48 MHz 校准值存放地址：0x1FFF 0130

4.7.13. Flash TS2P 寄存器(FLASH_TS2P)

偏移地址： 0x108

复位值： 0x0000 00B4

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	TS2P								
							RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:9	保留	-	-	保留
8:0	TS2P	RW	0xB4	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 0120 48MHz 校准值存放地址：0x1FFF 0134

4.7.14. Flash TPS3 寄存器(FLASH_TPS3)

偏移地址： 0x10C

复位值： 0x0000 06C0

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.	TPS3											
				RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:12	保留	-	-	保留
11:0	TPS3	RW	0x6C0	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 0120 48MHz 校准值存放地址：0x1FFF 0134

4.7.15. Flash TS3 寄存器(FLASH_TS3)

偏移地址： 0x110

复位值： 0x0000 00XX

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	TS3								
							RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:9	保留	-	-	保留
8:0	TS3	RW	0xXX	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： 24 MHz 校准值存放地址：0x1FFF 011C 48 MHz 校准值存放地址：0x1FFF 0130

4.7.16. Flash 页擦 TPE 寄存器(FLASH_PERTPE)

偏移地址： 0x114

复位值： 0x0001 4820

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.	PERTPE	
														RW	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
31:18	保留	-	-	保留
17:0	PERTPE	RW	0x14820	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： 24 MHz 校准值存放地址：0x1FFF 0124 48 MHz 校准值存放地址：0x1FFF 0138

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

偏移地址： 0x118

复位值： 0x0001 4820

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.	SMERTPE	
														RW	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
31:18	保留	-	-	保留
17:0	SMERTPE	RW	0x14820	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 0128 48MHz 校准值存放地址：0x1FFF 013C

4.7.18. FLASH PROGRAM TPE 寄存器(FLASH_PRGTPE)

偏移地址： 0x11C

复位值： 0x0000 5DC0

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	保留	-	-	保留
15:0	PRGTPE	RW	0x5DC0	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： 24 MHz 校准值存放地址：0x1FFF 012C 48 MHz 校准值存放地址：0x1FFF 0140

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

偏移地址： 0x120

复位值： 0x0000 12C0

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	保留	-	-	-
13：0	PRETPE	RW	0x12C0	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： 24 MHz 校准值存放地址：0x1FFF 012C 48 MHz 校准值存放地址：0x1FFF 0140

5. 电源控制

5.1. 电源框图

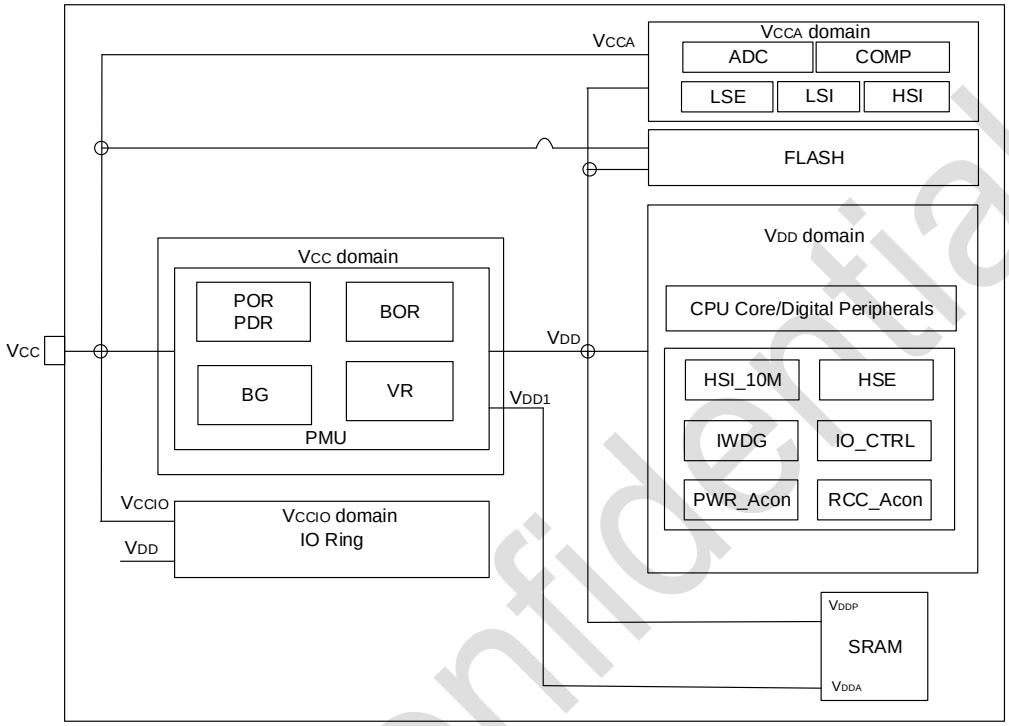


图 5-1 电源框图

表 5-1 电源框图

编号	电源	电源值	描述
1	V_{CC}	1.8 ~ 5.5 V	供电范围 1.8 ~ 5.5 V，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	V_{CCA}	1.8 ~ 5.5 V	给大部分模拟模块供电，来自于 V_{CC} PAD（也可设计单独电源 PAD）。
3	V_{CCIO}	1.8 ~ 5.5 V	给 IO 供电，来自于 V_{CC} PAD
4	V_{DDx} (V_{DD}/V_{DD1})	1.2 V ~ 1.0 V±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 1.2 V。当进入 Stop 模式时，根据软件配置，可以由 MR 或者 LPR 供电。

5.2. 电压调节器

芯片设计两种电压调节模式：

- 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。

V_{DD} 的电源根据芯片的工作模式，来自于 MR 或 LPR。

在芯片 Run 模式（正常运行状态）：

MR 保持工作，输出 1.2 V 电压，LPR 关闭。

软件通过配置可以决定 LDO 工作在 MR 模式，低功耗（Stop/Sleep）模式。在低功耗模式，BOR 根据上电加载或者寄存器配置工作或者不工作。

Stop 模式下涉及到唤醒功能，即 V_{DD} 的电源要从 LPR 切换到 MR，所以除了 LPR 要做到极低功耗外，切换时间也是重要的设计指标。

芯片系统要求 stop 模式总计唤醒时间（含 LPR 切换到 MR、HSI 唤醒、Flash 唤醒、CPU 唤醒）小于 6.5 us。

5.3. 动态电压值管理

动态电压值管理指的是对 VR 的输出 V_{DD} 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 1.2 V（V_{DD}），系统时钟频率可以运行在最快的 48 MHz 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。

5.4. 电源监控

5.4.1. 上电复位 (POR)/下电复位 (PDR)

芯片包含 Power-on reset（POR）、Brown-out reset（BOR）模块，提供电源相关复位。其中 POR 在所有 power 模式下都默认工作，BOR 需要使能后工作。POR/ BOR 低电平为复位。

在 BOR 使能后，在达到 V_{POR} 阈值后，POR 并不会立即释放，还需要 V_{DD} 达到 V_{BORFx} 规定的阈值。（V_{BORF} 的值在 option byte 中定义）。

5.4.2. 欠压复位 (BOR)

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。

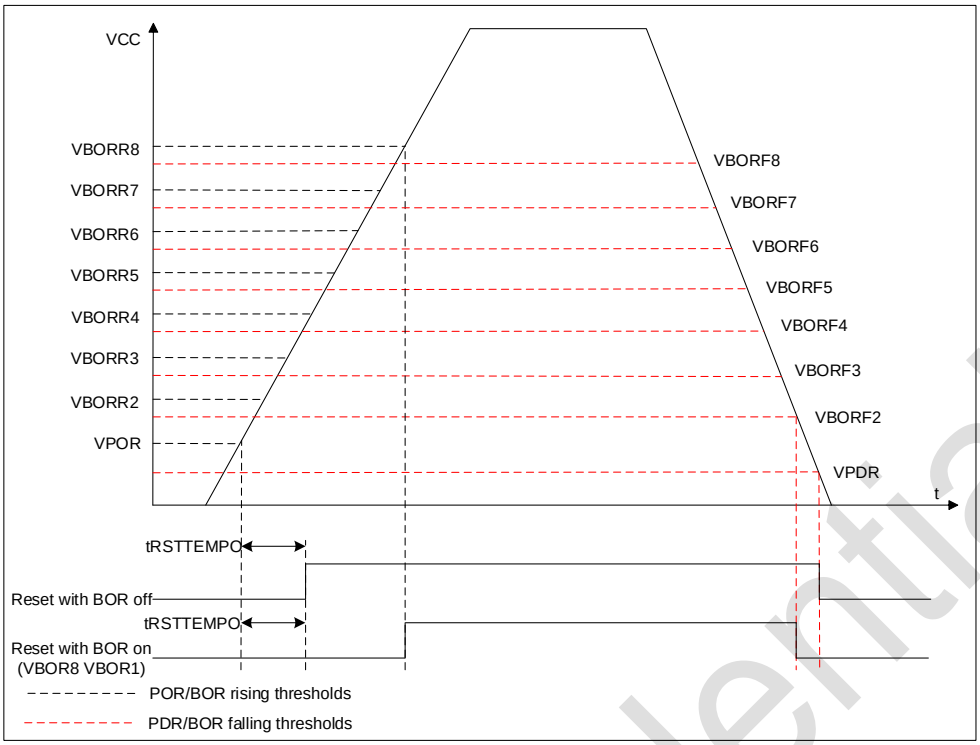


图 5-2 POR/PDR/BOR 阈值

5.5. 低功耗模式

芯片在正常的 Run 模式之外，有 4 个低功耗模式：

- **Sleep mode**：CPU HCLK 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**：LDO 进入 Low power 模式。该模式下 SRAM 和寄存器的内容保持， HSI 和 HSE 关闭，V_{DD} 域下大部分模块的时钟都被停掉。

在 Stop 模式，LSI 和 LSE 可以保持工作，RTC、IWDG 等可以保持工作。

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，但具备快速唤醒能力。

此外，正常 Run 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，Gating 掉外设时钟（系统时钟和模块时钟）

5.5.1. Run 模式

在 Run 模式下，为了降低功耗，可以采用：

- 降低系统时钟（SYSCLK/HCLK/PCLK）频率
- 只开启需要外设的时钟。

5.5.2. CPU 低功耗模式

- **进入**

CPU 通过 3 种方式进入低功耗模式：

- WFI
- WFE
- Cortex-M0+寄存器SLEEPONEXIT配置为1，并且从ISR返回。

■ **唤醒**

对于 V_{DD} 电压存在的低功耗模式，唤醒方式：

- WFI或者ISR返回后进入的低功耗模式，任何NVIC使能的外部中断都会唤醒；
- WFE进入的低功耗模式，唤醒事件产生时唤醒，唤醒事件包括：
- NVIC IRQ中断

当 CPU 控制寄存器 SEVONPEND=0 时：使能外设的中断寄存器+使能 CPU NVIC 寄存器；当从 WFE 恢复后，外设中断 pending 和 NVIC 外设 IRQ 通道 pending 位都必须清零；

当 CPU 控制寄存器 SEVONPEND=1 时：使能外设的中断寄存器+ CPU NVIC 寄存器（不管是否使能）；当从 WFE 恢复后，外设中断 pending 和 NVIC 外设 IRQ 通道 pending 位（如果使能）都必须清零。

■ **事件**

配置事件 EXTI。CPU 从 WFE 进入低功耗模式时，不需清零 EXTI 外设中断 pending 或者 NVIC IRQ 通道 pending 位。

5.5.3. Sleep 模式

在 Sleep 模式，CPU 时钟关闭(NVIC、SysTick 工作)，所有外设可以配置为工作（建议只使能必须工作的模块），外设的中断或者事件可以唤醒 CPU。

1. 硬件仅关闭CPU Core时钟，模块时钟由软件配置是否关闭
2. 模拟PMU工作在MR模式（PWR_CR1.LPR[1:0]=2’b00），数字V_{DD}电压为1.2 V

在 Sleep 模式，IO pin 的状态与 Run 模式相同。

选中的系统时钟源一直存在。

■ **进入**

- CPU的SLEEPDEEP=0，并且没有中断或者事件pending，则通过WFE或者WFI指令进入；
- CPU的SLEEPDEEP=0，且SLEEPONEXIT=1，没有中断pending时，从（最低优先级中断的）ISR返回后进入；

■ **唤醒**

- WFI或者ISR返回：使能中断；
- WFE，并且SEVONPEND=0：唤醒事件；
- WFE，并且SEVONPEND=1：中断（不管使能与否），或者唤醒事件；

注：CPU 接口并没有 SEVONPEND 信号输出，所以 PWR 模块并不会区分。

■ **唤醒 latency**

唤醒没有额外的等待时间。

■ **硬件实现**

在 Sleep 模式，硬件实现按照如下原则：

- 根据SLEEPING信号，控制关闭HCLK/PCLK；

- 外设IP时钟是否关闭由软件在进入Sleep模式前配置，硬件不会主动关闭外设时钟；
- 时钟选择由软件控制，硬件不会控制系统时钟频率及时钟源的选择；

5.5.4. Stop 模式

Stop 模式基于 CPU 的 DEEPSLEEP 模式，以及外设时钟门控。

V_{DD} 域的模块时钟均关闭，HSI 和 HSE 关闭。LSI 和 LSE 可以工作。

RTC 和 TAMP 保持工作。一些具备唤醒能力的外设（如 TK/I^2C）可以转换为 HSI 时钟去接收数据。此时 HSI 只响应外设请求。

Stop 模式：

- 硬件关闭高速HSE、HSI时钟
- 模拟PMU工作在LPR模式（PWR_CR1.LPR[1:0]=2’b01）。
- PWR_CR1.SRAM_RETVCTRL[1:0]=2’b00(PMU.SRAM_RETVSEL=1)时，SRAM电源V_{DDP}来源于V_{DD}
- PWR_CR1.SRAM_RETVCTRL[1:0]=2’b01(PMU.SRAM_RETVSEL=0)时，SRAM电源V_{DDA}来源于V_{DD}，SRAM电源V_{DDP}来源于V_{DD}．

Stop 模式，Main-VR 可以配置为关闭。

从 Stop 模式退出后，系统时钟为 HSISYS。

- **进入**
  - CPU的SLEEPDEEP=1，并且没有中断或者事件pending，则通过WFE或者WFI指令进入；
  - CPU的SLEEPDEEP=1，且SLEEPONEXIT=1，没有中断pending，时，从ISR返回后进入；
- **唤醒**
  - WFI或者ISR返回：使能为中断模式的EXTI(相应的EXTI中断向量必须在NVIC使能)。中断源可以为外部中断或者具有唤醒能力的外设的中断；
  - WFE，并且SEVONPEND=0：使能为事件模式的EXTI；
  - WFE，并且SEVONPEND=1：使能为中断模式的EXTI（即使相应的EXTI中断向量没有使能）；
  - 复位（POR/PinRST/IWDG reset/LSECSS;
- **唤醒 latency**

唤醒 latecny 取决于以下三者的最长时间：

- HSI时钟的唤醒时间
- Flash唤醒时间
- Main-VR唤醒的稳定时间
- 系统对唤醒时间有要求：5 us之内

5.5.5. 各工作模式下的功能

配置进入 stop 模式前，软件清零中断 pending 寄存器 EXTI_RPR1 和 EXTI_FPR1；

进入 stop 模式，硬件需要关闭 HSI/HSE 模拟时钟源；

Main-VR 工作还是 LP-VR 工作由软件配置；

唤醒源：

- GPIO0~15
- COMP1
- COMP2
- RTC
- IWDG复位
- POR复位
- nRST pin复位
- LSECSS
- I²C （I²C 唤醒进入I²C 地址匹配阶段如果来了其他唤醒（RTC/GPIO 等 直接唤醒源）可直接唤醒Stop模式）
- TK（TK 作为唤醒源，除了Sleep 可以中断唤醒外，其他模式唤醒流程是RTC唤醒PWR，PWR唤醒TK，TK 唤醒Stop模式，TK 作为唤醒源HSION_CTRL 必须为0）

表 5-2 各工作模式下的功能

Peripheral	Run	Sleep	Stop	
			VR@LPR or VR@MR	Wakeup ability
CPU Core	O	-	-	-
Flash memory	O	O	-	-
SRAM	O	O	-	-
Brown-out reset (BOR)	O	O	O	O
HSI	O	O	-	-
HSE	O	O	-	-
LSI	O	O	O	-
LSE	O	O	O	-
HSE Clock Security System (CSS)	O	O	-	-
LSE Clock Security System (CSS)	O	O	O	O
RTC	O	O	O	O
UART1/UART2/UART3	O	O	-	-
I²C	O	O	O	O
SPI	O	O	-	-
ADC	O	O	-	-
COMP1/COMP2	O	O	O(1)	O
OPA1	O	O	-	-
Temperature sensor	O	O	-	-
Timers(TIM1/TIM14)	O	O	-	-
IWDG	O	O	O	O
SysTick 定时器	O	O	-	-
CRC	O	O	-	-
TK	O	O	(1)	O(优先 RTC 唤醒 TK,然后 TK 唤醒 Stop)
GPIOs	O	O	O	O

1. COMP1、COMP2、TK模块，只有在VR@MR时才能工作。

5.6. 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。

5.6.1. 电源控制寄存器 1 (PWR_CR1)

偏移地址： 0x00

复位值： 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_CTRL		Res.
												RW	RW		
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
LPR[1:0]		FLS_SLP-TIME[1:0]		Res.	Res.		DBP	Res.							
RW		RW					RW								

Bit	Name	R/W	Reset Value	Function
31:20	保留	-	-	保留
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。
18:17	SRAM_RETV_CTRL[1:0]	RW	2’b01	00/01: Normal 电压选择； 10: Stop 模式下电压选择； 11:保留
16	保留	-	-	保留
15:14	LPR[1:0]	RW	2’b00	低功耗调节器 00：MR (Main regulator) 01：LPR (low power regulator) 10：保留 11：保留 注：Stop 模式唤醒后，硬件更新该寄存器为 00.
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: 5 us 2’b01: 2 us 2’b10: 3 us 2’b11: 0 us 注：Memory Boot 选择 SRAM 时，该寄存器设置为 2’b11/2’b01 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 3 us 内访问 Flash。
11:9	保留	-	-	保留
8	DBP	RW	0	RTC 写保护禁止 在复位后，RTC 处于写保护状态以防意外写入。要访问 RTC 该位必须设置为 1。

Bit	Name	R/W	Reset Value	Function
				0：禁止访问 RTC 1：可以访问 RTC
7:0	保留	-	-	保留

6. 复位

芯片内设计两种复位，分别是：电源复位和系统复位。

6.1. 复位源

6.1.1. 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）

电源复位包括如下几种：

- 模拟电路产生的 POR/ BOR，实现对 V_{CC} 的检测。当 V_{CC} 电压上升到 trigger 值时释放复位；当 V_{CC} 电压下降到某一 trigger 值时产生复位；
- 模拟电路产生的 PORI，实现对 VR 输出的检测。当 V_{DD} 电压上升到 trigger 值时释放复位；当 V_{DD} 电压下降到某一 trigger 值时产生复位；

6.1.2. 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）

通过检查 RCC_CSR 寄存器的复位标识位，可以识别复位源。

注: 当芯片进入 FBIST 模式，会对 CPU 进行复位。

6.1.2.1. NRST 管脚 (外部复位)

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）：

- 复位输入

输入：NRST pin 输入后，经过去毛刺电路（去毛刺可配置为禁止）后产生芯片的外部复位。

- GPIO

此时 NRST 可以作为标准 GPIO，不作为复位功能。复位由内部产生，并且内部产生的复位不会通过该 pin 输出。

注：上电复位后，NRST pin 默认配置为复位输入模式。

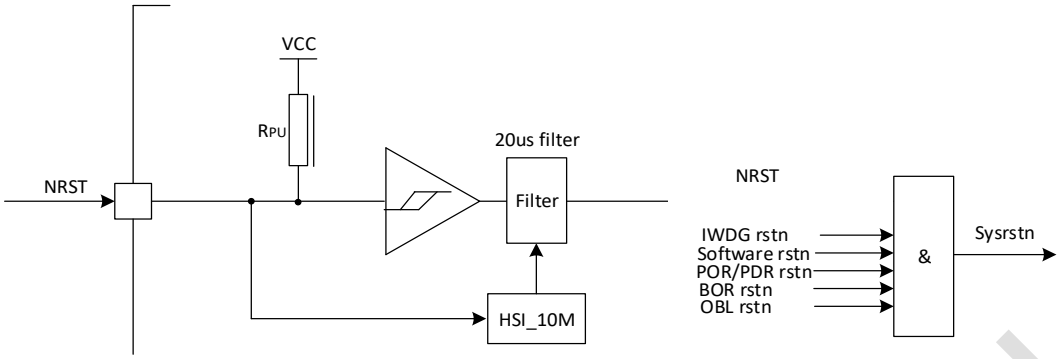


图 5-3 复位电路简化图

6.1.2.2. 看门狗复位

参见看门狗。该复位产生后，不需要重新进行读 Trimming 等操作。

6.1.2.3. 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。该复位产生后，不需要重新进行读 Trimming 等操作。

6.1.2.4. Option byte 加载复位

当 FLASH_CR.OBL_LAUNCH 位设为 1，则产生 option byte loader 复位。OBL_LAUNCH 用于软件启动加载 FLASH option 字节。

6.2. 复位源复位范围

下面为各个复位源对各寄存器的复位范围：

表 5-3 复位源复位范围

复位类型	复位源	复位寄存器	是否可屏蔽
系统复位	NRST 引脚低电平	除复位/时钟寄存器外的所有寄存器	×
	IWDG 复位		√
	软件复位 SYSRESETREQ		√
	Option byte 加载软复位		√
电源复位	PORE/PDR/BOR 复位	所有寄存器	×

6.3. 复位与系统功耗模式

系统中复位可能在任何工作模式中发生，有些复位只能在某些特定工作模式下发生。下表为复位与工作模式的关系：

表 5-4 复位源与功耗模式

复位类型	复位源	Run	Sleep	Stop
系统复位	NRST 引脚低电平（V_{CC}域）	√	√	√
	IWDG 溢出	√	√	√
	软件复位 SYSRESETREQ	√	×	×
	Option byte 加载软复位	√	×	×
电源复位	POR/PDR 复位（V_{CC}域）	√	√	√
	BOR 复位（V_{CC}域）	√	√	√

7. 时钟

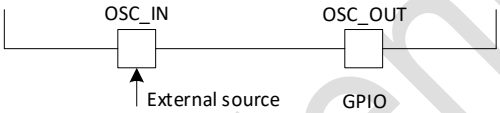
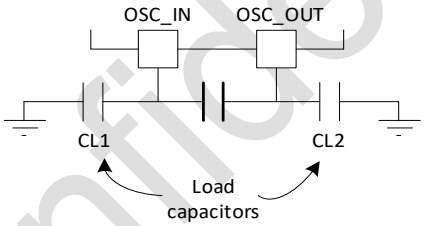
7.1. 时钟源

7.1.1. 外部高速时钟 HSE

外部高速时钟（HSE）来自两个来源：

- 通过外接 crystal（晶体），配合内部起振电路，产生 4 ~ 8 MHz 的时钟信号
- 直接从外部输入高速时钟源

表 7-1 HSI 时钟源

Clock source	Hardware configuration
外部时钟	
外接晶体	

7.1.1.1. 外接晶体

4 ~ 8 MHz 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。

7.1.1.2. 外接时钟源（HSE bypass）

该模式下，提供给了外部时钟源。软件通过 RCC_CR 的 HSEBYP 和 HSEON 位选择该模式。外部时钟源将会来自 OSC_IN pin，而 OSC_OUT pin 作为 GPIO 使用。

7.1.2. 外部低速时钟 LSE

外部 32.768 kHz OSC，用作低功耗时钟。
可以通过配置 LSE_DRIVER 在稳定时间和功耗之间做平衡。
与 HSE 来源类似，LSE 也有两个来源：

- 32.768 kHz XTAL+内部起振电路
- 通过 OSC_IN 输入的外部时钟（LSEBYP=1）

7.1.3. 内部高速时钟 HSI

内部 24/48 MHz RC 振荡器。相较于 XTAL OSC，RC OSC 功耗低，稳定时间短，但精度低。HSI 模拟模块计数稳定时间，即输出给数字的 HSI 时钟为稳定时钟。
上电复位后，在加载阶段，HSI 的校准和电压/温度 trimming 值以及中心频率加载到 RCC_ICSCR.HSI_TRIM 寄存器。

从 Stop 模式唤醒后，HSI 将作为系统时钟源。

HSI_10M

该时钟作为低精度时钟，用作 nRST pin 的滤波计数，以及 Flash 低速 Run 时低功耗处理。

7.1.4. 内部低速时钟 LSI

内部低速时钟，作为 RTC、IWDG 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 32.768 kHz。

7.2. 时钟树

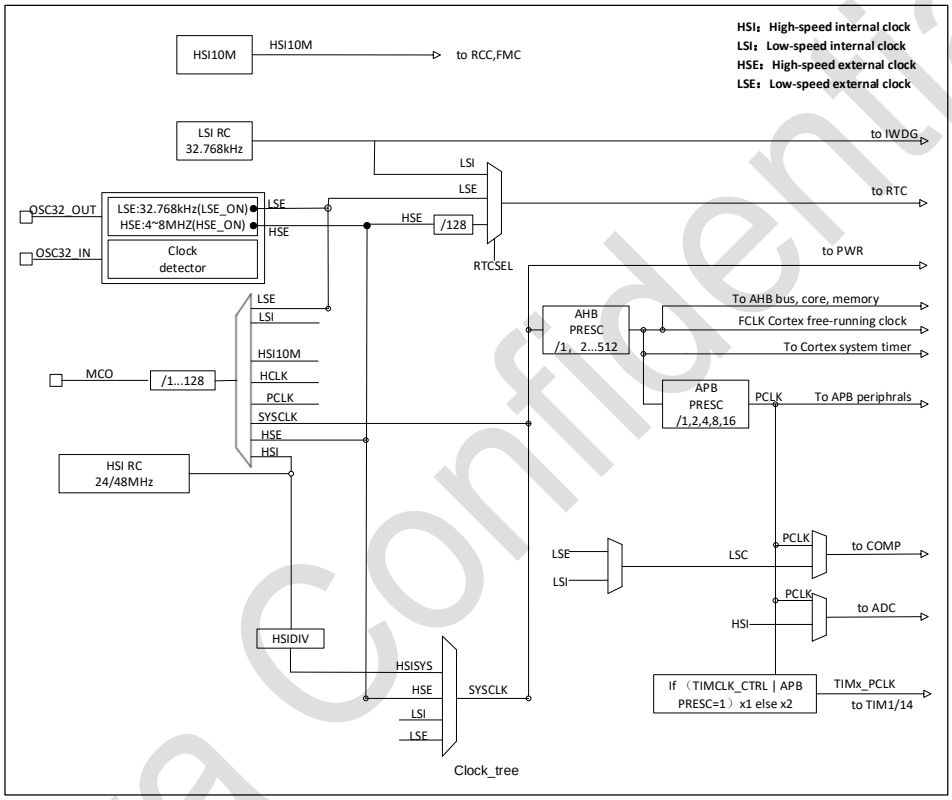


图 7-1 系统时钟结构图

7.3. 时钟安全系统 (CSS)

时钟安全主要包括如下方面：

- 时钟配置和状态安全
- 时钟源 HSE 安全
- 时钟源 LSE 安全

7.3.1.1. 时钟配置和状态安全

通过软件周期性地回读时钟配置和状态寄存器，获取系统当前时钟信息，并判断是否与预期一致。

7.3.1.2. 时钟源 HSE 监测

HSE 时钟检测系统（CSS）由 RCC_CR.HSE_CSSON 寄存器控制。当 HSE 时钟存在，并且稳定后，软件可以使能 CSS 检测。当 HSE 关闭后，CSS 功能由硬件自动关闭。

CSS 失败时：

- 关闭 HSE（HSEON 寄存器由硬件清零）,如果此时系统时钟时钟源选择 HSE，切换为 HSI。
- 时钟失败信号作为 TIM1 的刹车输入（在 Timer 内部有寄存器控制该功能是否使能）；
- 产生中断，并作为 NMI 中断，建议中断处理程序如下:
  - 写RCC_CICR.CSSC=1，清零HSE CSS标志寄存器；
  - 硬件将系统时钟源更换为HSI，同时软件可以配置系统时钟源选择非HSE时钟的其他时钟
  - 如果仍然选择HSE作为系统时钟源，需要先使能HSE（写RCC_CR.HSEON=1）

7.3.1.3. 时钟源 LSE 监测

时钟源 LSE 监测参考时钟为 LSI，在使能 LSE CSS 并且 LSE 稳定时，被监测时钟计数器 LSECAL（cal_count）和参考时钟监测计数器 LSICAL（ref_count）同时计数，LSICAL 递减计数，LSECAL 递增计数。当 LSICAL 递减到 0 后，判断 LSECAL 计数是否溢出。如果没有溢出，则表示在监测时间内 LSE 工作异常，可能 LSE 时钟停止或者频率大幅变化。

LSECSS 失败时：

- 关闭 LSE（LSEON 寄存器由硬件清零）；如果系统时钟选择 LSE，则切换为 LSI。
- 时钟失败信号作为 TIM1 的刹车输入（在 Timer 内部有寄存器控制该功能是否使能）；
- 产生中断，并作为 NMI 中断；
  - 写RCC_CICR.LSECSSC=1，清零LSE CSS标志寄存器；
  - 硬件将系统时钟源更换为LSI，同时软件可以配置系统时钟源选择非LSE时钟的其他时钟
  - 如果仍然选择LSE作为系统时钟源，需要先使能LSE（写RCC_CSR.LSEON=1）

7.4. 复位/时钟寄存器

该模块的寄存器可以用字(32 bits)、半字（16 bits）和字节（8 bits）访问。

7.4.1. 时钟控制寄存器 (RCC_CR)

偏移地址：0x00

复位值：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.	HSE CSSON	HSE BYP	HSE RDY	HSE ON
						R	RW					RS	RW	R	RW
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	HSIDIV[2:0]			HSI RDY	Res.	HSION	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
		RW			R		RW								

Bit	Name	R/W	Reset Value	Function
31:20	保留	-	-	保留
19	HSE_CSSON	RS	0	HSE 时钟安全系统使能。 当该位为 1 时，如果 HSE OSC ready 则硬件会使能时钟检测模块；如果 HSE 检测失败，则关闭时钟检测模块。

Bit	Name	R/W	Reset Value	Function
				0：时钟安全系统关闭（时钟检测关闭）； 1：时钟安全系统开启（如果 HSE 时钟稳定则时钟检测开启，否则关闭时钟检测）
18	HSEBYP	RW	0	HSE 屏蔽晶振，选择管脚输入时钟。 该位只有当 HSEON=0 时才能写. 0：HSE 晶振不屏蔽，外部高速时钟选择外部晶振； 1：HSE 晶振屏蔽，外部高速时钟选择外部管脚输入时钟源
17	HSERDY	R	0	HSE 晶振时钟 ready 标志。 该位由硬件置位表明 HSE 晶振稳定。 0：HSE 晶振没有 ready； 1：HSE 晶振 ready 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 注意：HSEON 和 LSEON 不能同时使用。
15:14	保留	-	-	保留
13:11	HSIDIV[2:0]	RW	0	HSI 时钟分频系数。 软件控制这些位设定 HSI 的分频系数，产生 HSISYS 时钟 000：1 001：2 010：4 011：8 100：16 101：32 110：64 111：128
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	保留	-	-	保留
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。 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	保留	-	-	保留

7.4.2. 内部时钟源校准寄存器 (RCC_ICSCR)

偏移地址：0x04

复位值： 0x0100_9100, 通过 POR/PDR/BOR 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.	LSI_TRIM[8:0]								
							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
31:25	保留	-	-	保留
24:16	LSI_TRIM	RW	9'h100	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 32.768 kHz 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'b100	HSI 频率选择: 100: 24 MHz 101: 48 MHz
12:0	HSI_TRIM	RW	13'h1100	时钟频率校准值。 上电后硬件用 HSI 4 MHz 的寄存器默认值，待 Trimming 时会把出厂信息写入该寄存器中。 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（24 / 48 MHz）下的校准。 保存在 Flash 的地址内: 24 MHz 校准值存放地址：0x1FFF 0100 48 MHz 校准值存放地址：0x1FFF 0104 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%。

7.4.3. 时钟配置寄存器 (RCC_CFGR)

偏移地址：0x08

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	MCOPRE[2:0]			MCOSEL[3:0]				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	保留	-	-	保留
30:28	MCOPRE[2:0]	RW	0	微处理器输出时钟（MCO）分频系数。 000: 1 001: 2

Bit	Name	R/W	Reset Value	Function
				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。
27:24	MCOSEL[3:0]	RW	0	MCO 输出时钟选择。 0000: no clock, MCO output disabled 0001: SYSCLK 0010: HSI10M 0011: HSI 0100: HSE 0110: LSI 0111: LSE 1000: HCLK 1001: PCLK 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	保留	-	-	保留
14:12	PPRE[2:0]	RW	0	APB 时钟分频系数。 PCLK 基于 HCLK 的分频系数。 0xx: 1 100: 2 101: 4 110: 8 111: 16
11:8	HPRE[3:0]	RW	0	AHB 时钟 HCLK 基于 SYSCLK 的分频系数。 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	保留	-	-	保留
5:3	SWS[2:0]	R	0	系统时钟源选择。 该位由硬件控制，表明系统时钟源的选择。 000: HSISYS 001: HSE 011: LSI 100: LSE Others: 保留
2:0	SW[2:0]	RW	0	系统时钟源选择。 该位由硬件和软件控制，表明系统时钟源的选择。 000: HSISYS 001: HSE 011: LSI 100: LSE Others: HSISYS 注意：系统时钟选择 HSE 时不能直接切换为 LSE，系统时钟选择 LSE 时不能直接切换为 HSE。可以选 HSISYS 或 LSI 做为 LSE 和 HSE 之间切换的中转。（像系统时钟选择

Bit	Name	R/W	Reset Value	Function
				HSE 切换为 LSE 的情况下，先 SW 切换 HSE 到 HSISYS，再关闭 HSEON 开启 LSEON，等 LSE ready 再 SW 切换为 LSE；LSE 切换为 HSE 同理) 硬件配置为 HSISYS 的情况： 1. MCU 从 stop 模式退出； 2. 软件配置为 001(HSE)，出现 HSE failure。

7.4.4. 外部时钟源控制寄存器 (RCC_ECSCR)

偏移地址：0x10

复位值： 0x0003_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.	LSE_STARTUP		Res.		LSE_DRIVER	
										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.	HSE_STARTUP		Res.		HSE_DRIVER
											RW				RW

Bit	Name	R/W	Reset Value	Function
31:22	保留	-	-	保留
21:20	LSE_STARTUP	RW	0x0	LSE 晶振稳定时间选择。 LSEBYP=0： 00：4096 个 LSE 时钟周期； 01：2048 个 LSE 时钟周期； 10：8192 个 LSE 时钟周期； 11：不计稳定时间，直接输出； LSEBYP=1： 00：2048 个 LSE 时钟周期； 01：1024 个 LSE 时钟周期； 10：4096 个 LSE 时钟周期； 11：不计稳定时间，直接输出；
19:18	保留	-	-	保留
17:16	LSE_DRIVER	RW	0x3	低速晶振驱动能力选择。 00：最弱驱动能力； 01：弱驱动能力； 10：默认驱动能力；（推荐） 11：最强驱动能力； 注：需要根据晶振特性、负载电容以及电路板的寄生参数选择适当的驱动能力。驱动能力越大则功耗越大，驱动能力越弱则功耗越小。
15:5	保留	-	-	保留
4:3	HSE_STARTUP	RW	0x0	HSE 稳定时间选择。 HSEBYP=0: 00：4096 个 HSE 时钟； 01：2048 个 HSE 时钟；

Bit	Name	R/W	Reset Value	Function
				10：8192 个 HSE 时钟； 11：不计稳定时间，直接输出； HSEBYP=1: 00：2048 个 HSE 时钟； 01：1024 个 HSE 时钟； 10：4096 个 HSE 时钟； 11：不计稳定时间，直接输出；
2::1	保留	-	-	保留
0	HSE_DRV	RW	0x0	HSE 驱动能力选择, default 0 0: gm 3.5 mA/V 1: gm 7.5 mA/V

7.4.5. 时钟中断使能寄存器 (RCC_CIER)

偏移地址：0x18

复位值：0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31:5	保留	-	-	保留
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	保留	-	-	保留
1	LSERDYIE	RW	0	LSE 时钟 ready 中断使能。 0：禁止； 1：使能；
0	LSIRDYIE	RW	0	LSI 时钟 ready 中断使能。 0：禁止； 1：使能；

7.4.6. 时钟中断标志寄存器 (RCC_CIFR)

偏移地址：0x1C

复位值：0x0000 0000

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

						CSSF					RDYF	RDYF		RDYF	RDYF
						R	R				R	R		R	R

Bit	Name	R/W	Reset Value	Function
31:10	保留	-	-	保留
9	LSECSSF	R	0	LSE 时钟安全系统（CSS）中断标志。 当硬件检测 LSE OSC 时钟失败时置位该寄存器。 0：LSE 时钟检测失败中断未产生； 1：LSE 时钟检测失败中断产生； 写 LSECSSC 寄存器 1 清零该位。
8	CSSF	R	0	HSE 时钟安全系统（CSS）中断标志。 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	保留	-	-	保留
4	HSERDYF	R	0	HSE 时钟 ready 中断标志。 当 HSE 时钟稳定并且 HSERDYIE=1 时，硬件置位该寄存器。 0：HSE 时钟 ready 中断未产生； 1：HSE 时钟 ready 中断产生； 写 HSERDYC 寄存器 1 清零该位。
3	HSIRDYF	R	0	HSI 时钟 ready 中断标志。 HSI 时钟稳定并且 HSIRDYIE=1 时，硬件置位该寄存器。 0：HSI 时钟 ready 中断未产生； 1：HSI 时钟 ready 中断产生； 写 HSIRDYC 寄存器 1 清零该位。
2	保留	-	-	保留
1	LSERDYF	R	0	LSERDY 时钟 ready 中断标志。 LSE 时钟稳定并且 LSERDYDIE=1 时，硬件置位该寄存器。 0：LSERDY 时钟 ready 中断未产生； 1：LSERDY 时钟 ready 中断产生； 写 LSERDYC 寄存器 1 清零该位。
0	LSIRDYF	R	0	LSIRDY 时钟 ready 中断标志。 LSI 时钟稳定并且 LSIRDYIE=1 时，硬件置位该寄存器。 0：LSIRDY 时钟 ready 中断未产生； 1：LSIRDY 时钟 ready 中断产生； 写 LSIRDYC 寄存器 1 清零该位。

7.4.7. 时钟中断清除寄存器 (RCC_CICR)

偏移地址：0x20

复位值：0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31:10	保留	-	-	保留
9	LSECSSC	W	0	LSE 时钟安全系统（CSS）中断标志清零。 0：No effect； 1：清零 LSECSSF 标志
8	CSSC	W	0	HSE 时钟安全系统（CSS）中断标志清零。 0：No effect； 1：清零 CSSF 标志
7:5	保留	-	-	保留
4	HSERDYC	W	0	HSE ready 中断标志清零。 0：No effect； 1：清零 HSERDYF 标志；
3	HSIRDYC	W	0	HSI ready 中断标志清零。 0：No effect； 1：清零 HSIRDYF 标志；
2	保留	-	-	保留
1	LSERDYC	W	0	LSE ready 中断标志清零。 0：No effect； 1：清零 LSERDYF 标志；
0	LSIRDYC	W	0	LSI ready 中断标志清零。 0：No effect； 1：清零 LSIRDYF 标志；

7.4.8. I/O 接口复位寄存器 (RCC_IOPRSTR)

偏移地址：0x24

复位值：0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
5	GPIOFRST	RW	0	I/O PortF 复位。 0：no effect； 1：PortF I/O 复位；
4:2	保留	-	-	保留
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；

7.4.9. AHB 外设复位寄存器 (RCC_AHBRSTR)

偏移地址：0x28

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
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	保留	-	-	保留
12	CRCRST	RW	0	CRC 模块复位。 0：No effect； 1：CRC 模块复位；
11:0	保留	-	-	保留

7.4.10. APB 外设复位寄存器 1（RCC_APBRSTR1）

偏移地址：0x2C

复位值=0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	PWR RST	DBG RST	Res.	Res.	Res.	Res.	Res	I2C RST	Res	Res.	UART2 RST	UART1 RST	Res.
			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.	RTCAP-BRST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
					RW										

该寄存器由软件置位和清零，软件置位后，该模块维持复位直到软件将复位清零。

Bit	Name	R/W	Reset Value	Function
31:29	保留	-	-	保留
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	保留	-	-	保留
21	I2C1RST	RW	0	I^2C1 模块复位。 0：no effect； 1：该模块复位；
20:19:	保留	-	-	保留
18	UART2RST	RW	0	UART2 模块复位。 0：no effect； 1：该模块复位；
17	UART1RST	RW	0	UART1 模块复位。 0：no effect； 1：该模块复位；
16:11	保留	-	-	保留
10	RTCAPBRST	RW	0	RTC 模块 APB 复位。 0：no effect； 1：该模块复位；
9:0	保留	-	-	保留

7.4.11. APB 外设复位寄存器 2（RCC_APBRSTR2）

偏移地址：0x30

复位值=0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	TKRST	Res.	Res.	COMP2RST	COMP1RST	ADCRST	Res.	Res.	Res.	Res.
						RW			RW	RW	RW				
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
TIM14RST	Res.	UART3RST	SPI1RST	TIM1RST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SYS-CFGRST
RW		RW	RW	RW											RW

该寄存器由软件置位和清零，软件置位后，该模块维持复位直到软件将复位清零。

Bit	Name	R/W	Reset Value	Function
31:26	保留	-	-	保留
25	TKRST	RW	0	TK 模块复位。 0：no effect； 1：该模块复位；
24:23	保留	-	-	保留
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；
19:17	保留	-	-	保留
15	TIM14RST	RW	0	TIM14 模块复位。 0：no effect； 1：该模块复位；
14	保留	-	-	保留
13	UART3RST	RW	0	UART3 模块复位。 0：no effect； 1：该模块复位；
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	保留	-	-	保留
0	SYSCFGRST	RW	0	SYSCFG 模块复位。 0：no effect； 1：该模块复位；

7.4.12. I/O Port 时钟使能寄存器（RCC_IOPENR）

偏移地址：0x34，

复位值=0x0000_0000

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

该寄存器由软件置位和清零。

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

7.4.13. AHB 外设时钟使能寄存器（RCC_AHBENR）

偏移地址：0x38，

复位值=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	RW								

该寄存器由软件置位和清零。

Bit	Name	R/W	Reset Value	Function
31:13	保留	-	-	保留
12	CRCEN	RW	0	CRC 模块时钟使能。 0：禁止； 1：使能；
11:10	保留	-	-	保留
9	SRAMEN	RW	1	SRAM 存储区时钟使能，针对 sleep 模式。 0：禁止； 1：使能；
8	FLASHEN	RW	1	Flash 接口模块时钟使能，针对 sleep 模式。 0：禁止； 1：使能；
7:0	保留	-	-	保留

7.4.14. APB 外设时钟使能寄存器 1（RCC_APBENR1）

偏移地址：0x3C，

复位值=0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	PWR EN	DBG EN	Res.	Res.	Res.	Res.	Res.	I2C EN	Res.	Res.	UART2 EN	UART1 EN	Res.
			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.	RTC APB EN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
					RW										

该寄存器由软件置位和清零。

Bit	Name	R/W	Reset Value	Function
31:29	保留	-	-	保留
28	PWREN	RW	0	Power 接口模块时钟使能。 0：禁止； 1：使能；
27	DBGEN	RW	0	DBG 模块时钟使能 0：禁止；

				1：使能；
26:22	保留	-	-	保留
21	I2C1EN	RW	0	I²C1 模块时钟使能。 0：禁止； 1：使能；
20:19	保留	-	-	保留
18	UART2EN	RW	0	UART2 模块时钟使能。 0：禁止； 1：使能；
17	UART1EN	RW	0	UART1 模块时钟使能。 0：禁止； 1：使能；
16:11	保留	-	-	保留
10	RTCAPBEN	RW	0	RTC 模块 APB 时钟使能。 0：禁止； 1：使能；
9:0	保留	-	-	保留

7.4.15. APB 外设时钟使能寄存器 2（RCC_APBENR2）

偏移地址：0x40，

复位值=0x0000_0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	TKEN	Res.	Res.	COMP2EN	COMP1EN	ADCEN	Res.	Res.	Res.	Res.
						RW			RW	RW	RW				
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
TIM14EN	Res.	UART3EN	SPI1EN	TIM1EN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SYS-CFGEN
RW		RW	RW	RW											RW

该寄存器由软件置位和清零。

Bit	Name	R/W	Reset Value	Function
31:26	保留	-	-	保留
25	TKEN	RW	0	TK 模块时钟使能。 0：禁止； 1：使能；
24:23	保留	-	-	保留
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；
19:16	保留	-	-	保留
15	TIM14EN	RW	0	TIM14 模块时钟使能。 0：禁止； 1：使能；
14	保留	-	-	保留
13	UART3EN	RW	0	UART3 模块时钟使能。 0：禁止； 1：使能；
12	SPI1EN	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	保留	-	-	保留
0	SYSCFGEN	RW	1	SYSCFG、COMP 和 V_{REFBUF} 模块时钟使能。 0：禁止； 1：使能；

7.4.16. 外设时钟配置寄存器（RCC_CCIPR）

偏移地址：0x54，

复位值=0x0000_0000

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

该寄存器由软件置位和清零。

Bit	Name	R/W	Reset Value	Function
31:24	保留	-	-	保留
23:20	ADC_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 0111：PCLK/2

				1000：HSI 1001：HSI/2 1010：HSI/4 1011：HSI/8 1100：HSI/16 1101：HSI/32 1110：HSI/64 1111：HSI/2 其他： 建议软件在 ADC 不使能且 ADC 配置为：ADCAL=0, ADSTART=0, ADSTP=0 and ADEN=0)时操作这些位。
19:13	保留	-	-	保留
12	IWDGSEL	RW	0	IWDG 内部时钟源选择。 0: LSI 1: LSE
11:10	保留	-	-	保留
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：1.在使能 COMP2_FR.FLTEN 之前先配置该寄存器选择时钟。 2.在选择 LSC 前先使能 COMP2 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：1.在使能 COMP1_FR.FLTEN 之前先配置该寄存器选择时钟。 2.在选择 LSC 前先使能 COMP1 时钟
7:1	保留	-	-	保留
0	TIMCLK_CTRL	RW	0	TIMER PCLK 频率控制。 0：TIMER PCLK 为系统 PCLK*2，但频率不会超过 HCLK； 1：TIMER PCLK 为系统 PCLK*1；

7.4.17. RTC domain 控制寄存器（RCC_BDCR）

偏移地址：0x5C，

复位值=0x0000_0000，由 POR/PDR/BOR 复位

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.	LSC SEL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BDRST
						RW									RW
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
RTC EN	Res.	Res.	Res.	RTC_HSEDIV_SEL[1:0]		RTCSEL [1:0]		Res.	LSECSSD	LSECS-SON	Res.	Res.	LSE BYP	LSE RDY	LSE ON
RW				RW		RW			RW	RW			RW	R	RW

Bit	Name	R/W	Reset Value	Function
31:26	保留	-	-	保留
25	LSCSEL	RW	0	低速时钟选择。 0: LSI 1: LSE
24:17	保留	-	-	保留
16	BDRST	RW	0	RTC 软复位。 0：no effect； 1：复位；
15	RTCEN	RW	0	RTC 时钟使能。 0：禁止； 1：使能； BDRST 软复位为 00
14:12	保留	-	-	保留
11:10	RTC_HSEDIV_SEL[1:0]	RW	0	RTC 时钟源选择为 HSE 时钟的分频 00：32 分频 01：128 分频 10：8 分频 11：32 分频 BDRST 软复位为 00
9:8	RTCSEL[1:0]	RW	0	RTC 时钟源选择。 00: No clock 01: LSE 10: LSI 11: HSE divided by RTC_HSEDIV_SEL[1:0]。 一旦 RTC 时钟源选择后不能再改变，除非以下情况： 1.RTC 复位为 00 2.选择为 LSE（LSECSSD=1）但没有 LSE 3.BDRST 软复位为 00
7	保留	-	-	保留
6	LSECSSD	R	0	LSE CSS(clock security system)检测失败。 该位由硬件置位，表明 CSS 检测 32.768KHz OSC（LSE）失败。 0：未检到 LSE 失败 1：检测 ISE 失败
5	LSECSSON	RW	0	LSE CSS 使能 0：禁止； 1：使能； 注：必须 LSEON=1 并且 LSERDY=1 后才能使能 LSECSSON。 一旦使能该位，不能再把该位禁止，除非 LSECSSD=1.
4:3	保留	-	-	保留
2	LSEBYP	RW	0	LSE OSC bypass 0: Not bypassed，低速外部时钟选择晶振； 1: Bypassed，低速外部时钟选择外部接口输入时钟；

				注：只有当外部 32.768kHz OSC 禁止（LSEON=0 并且 LSERDY=0）时才能写该位。
1	LSERDY	R	0	LSE OSC ready. 硬件配置该位为 1 表明 LSE 时钟 ready。
0	LSEON	RW	0	LSE OSC 使能。 0：禁止； 1：使能； 注意：LSEON 和 HSEON 不能同时使用。

7.4.18. 控制/状态寄存器（RCC_CSR）

偏移地址：0x60

复位值=0x0800_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	IWDG RSTF	SFT RSTF	POR RSTF	PIN RSTF	OBL RSTF	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.	BOR RSTF	NRST_ FLT-DIS	Res.	Res.	Res.	Res.	Res.	Res.	LSI RDY	LSION
						R	RW							R	RW

Reset by POR(flag bit), reset by system reset(LSION)，reset by system reset excluding NRST(NRST_FLTIDS)

该寄存器中复位标志位只能由 power reset 复位，其他由 system reset 复位。

Bit	Name	R/W	Reset Value	Function
31:30	保留	-	-	保留
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	PORRSTF	R	1	POR 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	保留	-	-	保留
23	RMVF	RW	0	软件置位后，会清零复位标志。
22:10	保留	-	-	保留
9	BORRSTF	R	0	BOR 复位标志。 RMVF 置 1 会清零该位。
8	PINRST_FLTDIS	RW	0	NRST 滤波禁止 0：使能 HSI_10M，且滤波 20us 宽度功能使能 1：滤波功能禁止，且 HSI_10M 保持关闭
7:2	保留	-	-	保留
1	LSIRDY	R	0	LSI OSC 稳定标志。 LSIRDY 由模拟 LSI 产生。

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）和软件使能 LSECSSON 时，硬件会对该位进行置位。

8. 通用 I/O（GPIO）

8.1. 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)。

8.2. GPIO 主要特性

- 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）
- 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 8 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

8.3. 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
- push-pull 输出，带上拉或者下拉
- 带上拉或者下拉的复用功能的 push-pull
- 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:

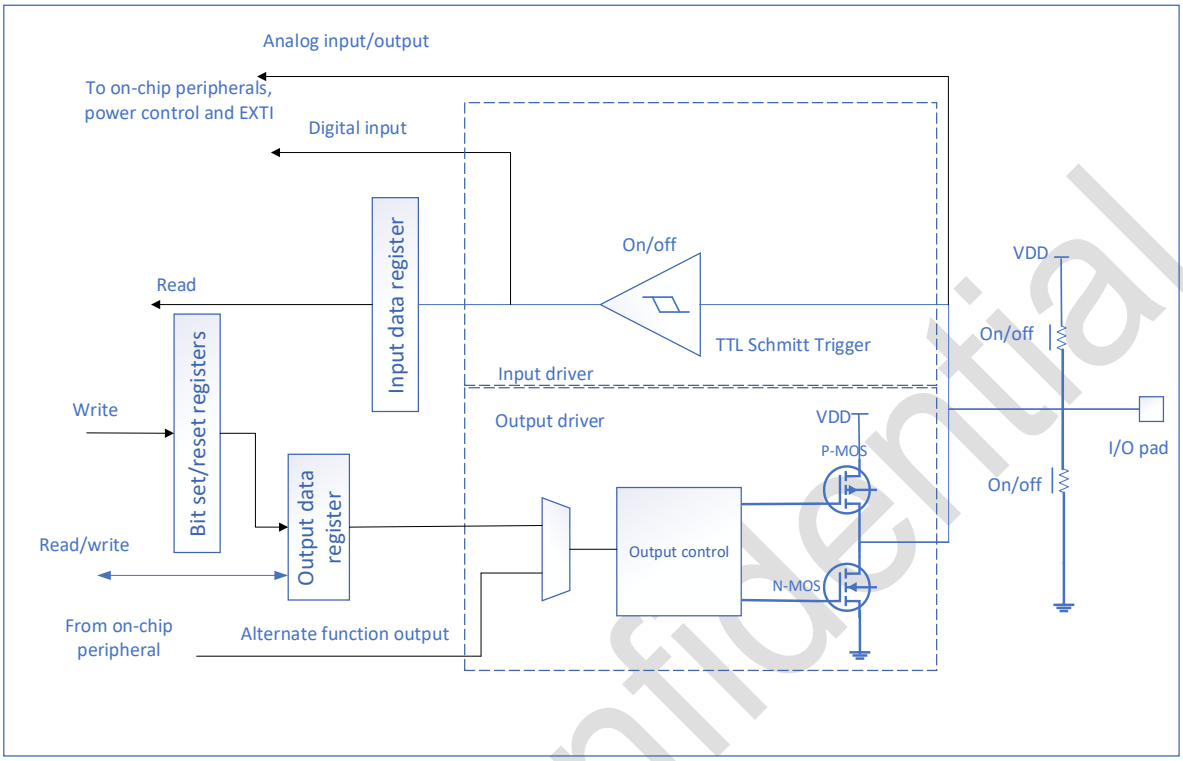


图 8-1 IO 端口位的基本结构

8.3.1. 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:

表 8-1 IO 端口复用模式

option[1:0]	PF3	PF4	PA13	PA14
0/0(default)	SWCLK	SWDIO	GPIO	GPIO
0/1	GPIO	GPIO	SWDIO	SWCLK
1/0	GPIO	SWDIO	GPIO	SWCLK
1/1	SWCLK	GPIO	SWDIO	GPIO

- PF3
  - Flash option byte未配置SWC时，默认为普通IO
  - Flash option byte配置SWC时, 默认为SWCLK下拉
- PA14
  - Flash option byte未配置SWC时，默认为普通IO
  - Flash option byte配置SWC时, 默认为SWCLK下拉
- PF4
  - Flash option byte未配置SWD时，默认为普通IO
  - Flash option byte配置SWD时, 默认为SWDIO上拉

- PA13
  - Flash option byte未配置SWD时，默认为普通IO
  - Flash option byte配置SWD时，默认为SWDIO上拉

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。有可能会使用推挽或者开漏模式的输出(只能驱动低电平，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。

8.3.2. 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 8 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
- 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，COMP功能均在ADC，COMP模块的寄存器中使能。当IO口用做ADC，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 和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。

8.3.3. 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 and 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

8.3.4. 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。

8.3.5. 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。

8.3.6. 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。

8.3.7. 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。

8.3.8. 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，并且需要使能输入触发。

8.3.9. 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

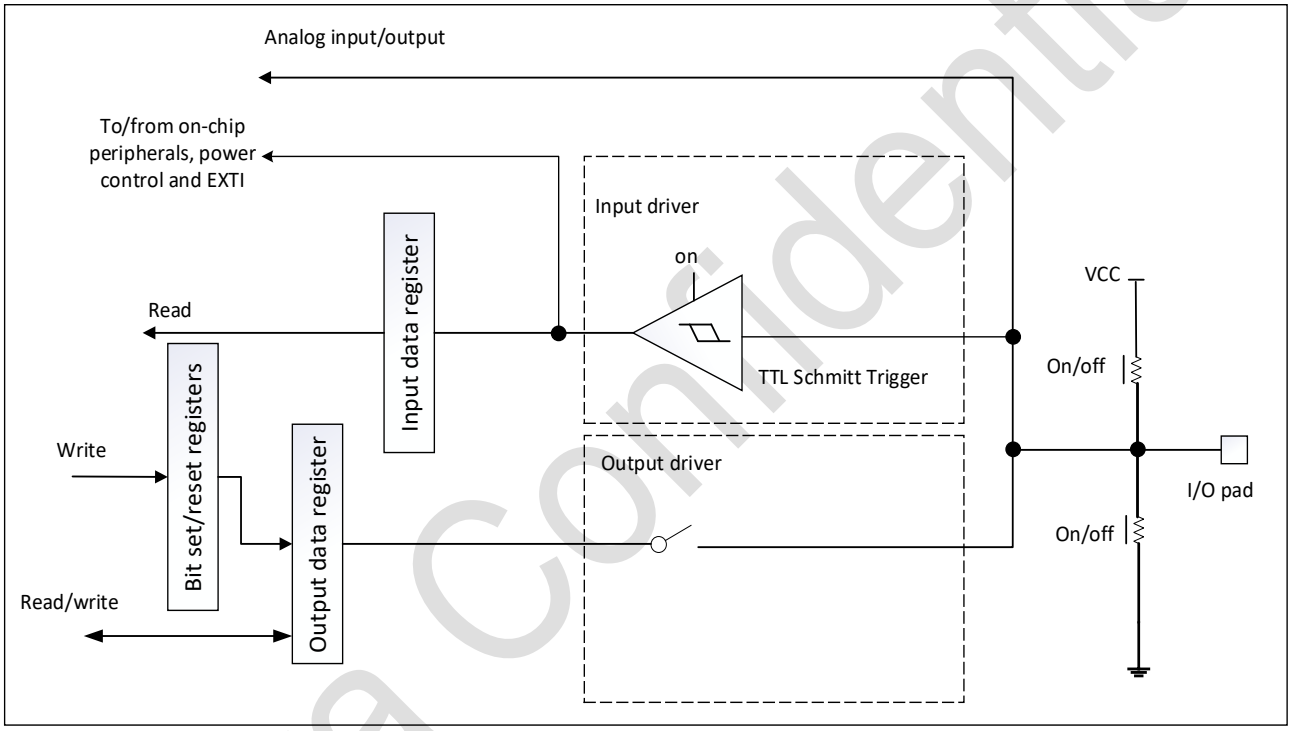


图 8-2 输入浮空/上拉/下拉配置

8.3.10. 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

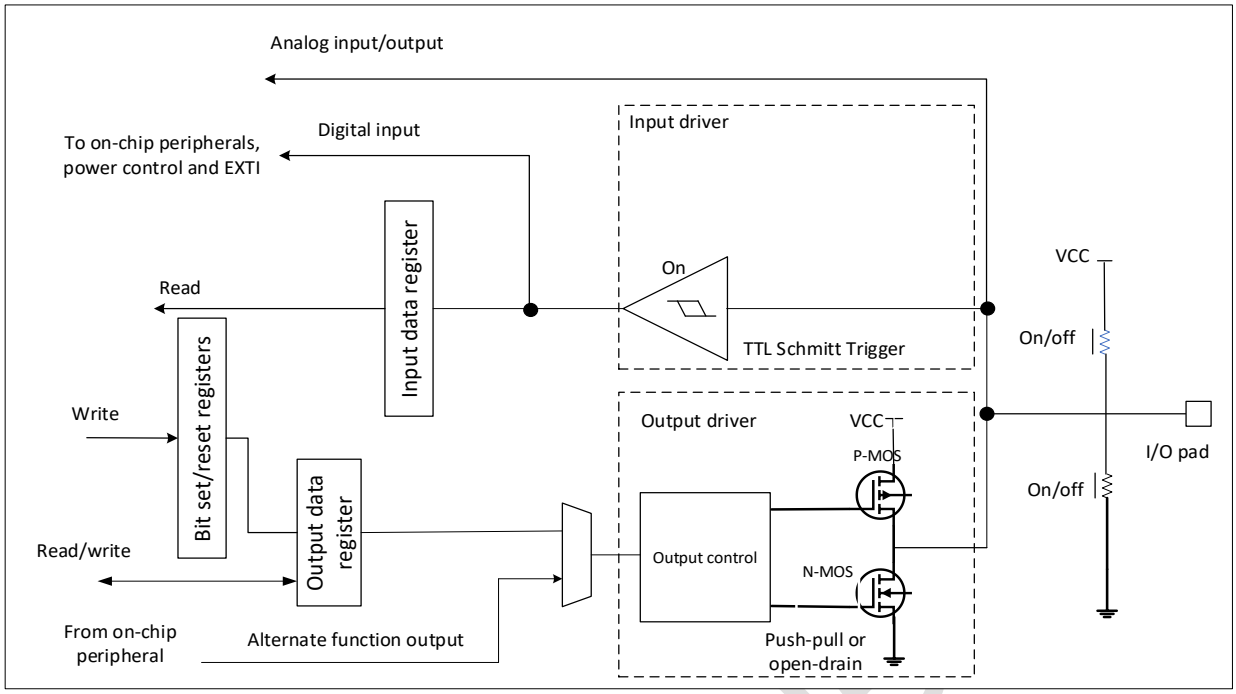


图 8-3 输出配置

8.3.11. 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

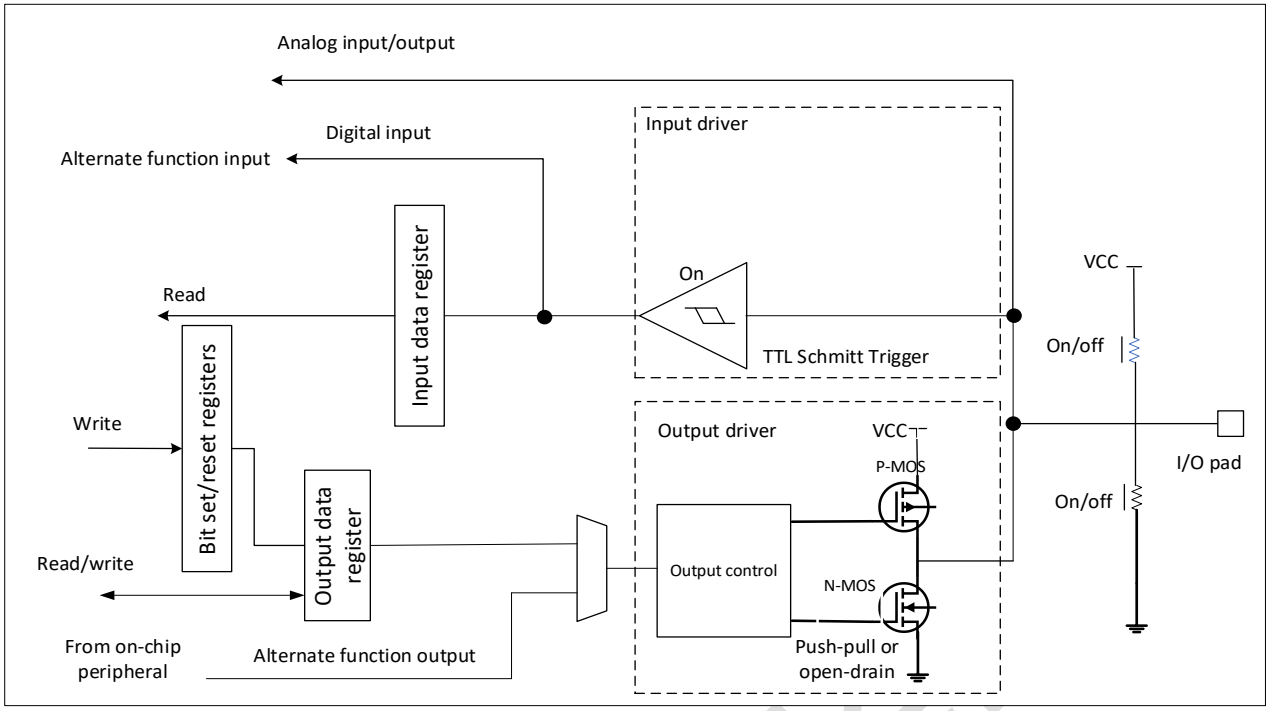


图 8-4 复用功能配置

8.3.12. 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'。

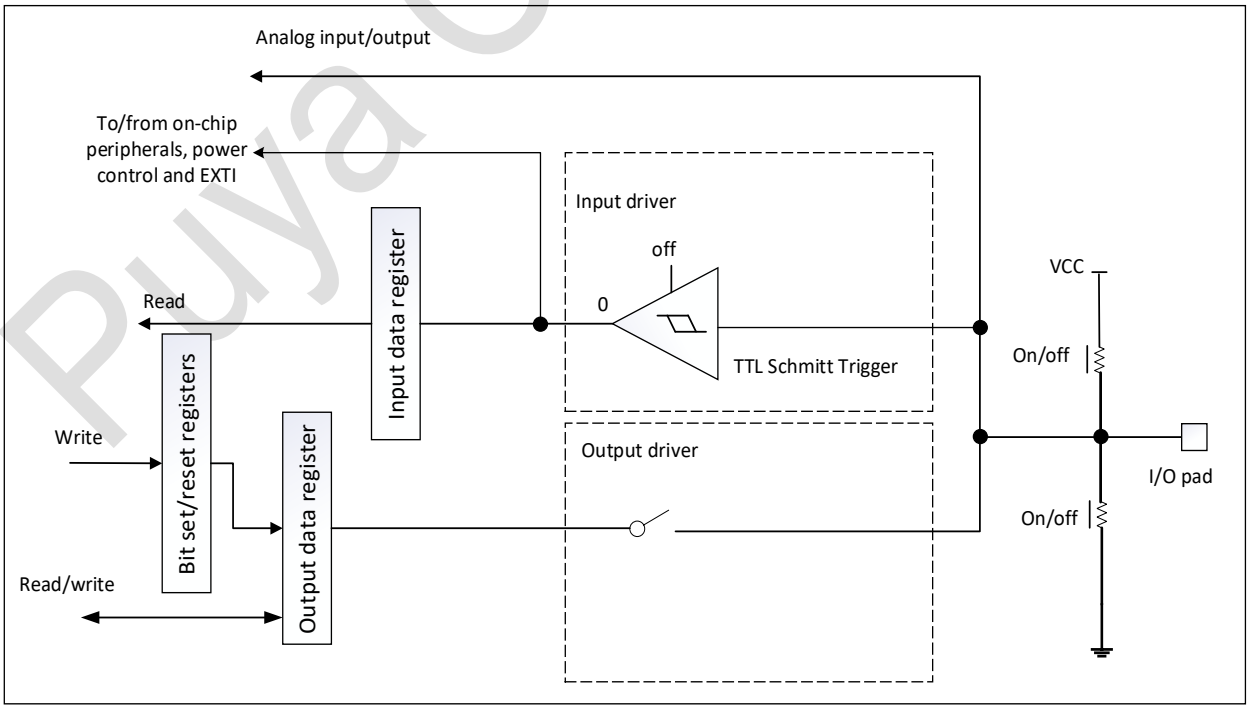


图 8-5 高阻抗模拟配置

8.3.13. 使用 LSE 管脚作为 GPIO

当 LSE 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 LSE 功能打开（RCC_BDCR 寄存器中设置 LSEON），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 或者 OSC32_IN 保留给时钟输入，而 OSC_OUT 或 OSC32_OUT 脚仍然可以用作正常 GPIO。

8.4. GPIO 寄存器

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

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

偏移地址： 0x00

复位值：

- GPIOA reset 值
  - Flash option byte配置11时：0xFBFF FFFF
  - Flash option byte配置10时：0xEFFF FFFF
  - Flash option byte配置01时：0xEBFF FFFF
  - Flash option byte配置00时：0xFFFF FFFF
- GPIOB reset 值
  - 0x0000 FFFF
- GPIOF reset 值
  - Flash option byte配置为11时：0x0000 0FBF
  - Flash option byte配置为10时：0x0000 0EFF
  - Flash option byte配置为01时：0x0000 0FFF
  - Flash option byte配置为00时：0x0000 0EBF

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															
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															

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)

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

偏移地址： 0x04

复位值： 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
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	保留	-	-	保留
15:0	OT[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

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

偏移地址： 0x08

复位值：

- GPIOA reset 值
  - Flash option byte配置11时：0x0C00 0000
  - Flash option byte配置10时：0x0000 0000
  - Flash option byte配置01时：0x0C00 0000
  - Flash option byte配置00时：0x0000 0000
- GPIOB reset 值
  - 0x0000 0000
- GPIOF reset 值
  - Flash option byte配置11时：0x0000 0300
  - Flash option byte配置10时：0x0000 0000
  - Flash option byte配置01时：0x0000 0000
  - Flash option byte配置00时：0x0000 0300

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：低速 01：中速 10：高速 11：非常高

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

偏移地址： 0x0C

复位值：

- GPIOA reset 值
  - Flash option byte配置11时：0x0400 0000
  - Flash option byte配置10时：0x2000 0000
  - Flash option byte配置01时：0x2400 0000
  - Flash option byte配置00时：0x0000 0000
- GPIOB reset 值
  - 0x0000 0000
- GPIOF reset 值
  - Flash option byte配置11时：0x0000 0080
  - Flash option byte配置10时：0x0000 0100
  - Flash option byte配置01时：0x0000 0000
  - Flash option byte配置00时：0x0000 0180

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: 上拉+下拉

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

偏移地址： 0x10

复位值： 0x0000 XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
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	保留	-	-	保留
15:0	IDy	R	**0000**	y = 15..0 这是只读的，读出值位对应 I/O 口的状态

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

偏移地址： 0x14

复位值： 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															
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	保留			
15:0	ODy[1:0]	RW	**0**	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。

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

偏移地址： 0x18

复位值： 0x0000 0000

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

8.4.8. GPIO 端口锁定配置寄存器 (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）

偏移地址： 0x1C

复位值： 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LCKK
															rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LCK1	LCK1	LCK1	LCK1	LCK1	LCK1	LCK	LCK	LCK	LCK	LCK	LCK	LCK	LCK	LCK	LCK0

5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	
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	保留			
16	LCKK	RW	**0**	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 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->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	**0**	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

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

偏移地址： 0x20

复位值： 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	AFSEL15[2:0]			Res.	AFSEL14[2:0]			Res.	AFSEL13[2:0]			Res.	AF-SEL12[2:0]		
	RW	RW	RW		RW	RW	RW		RW	RW	RW		RW	RW	RW
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	AFSEL11[3:0]			Res.	AFSEL10[2:0]			Res.	AFSEL9[2:0]			Res.	AFSEL8[2:0]		
	RW	RW	RW		RW	RW	RW		RW	RW	RW		RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31	保留			
30:28	AFSELy[2:0](y= 7 to 0)	RW	**3'h0**	软件可写这些位配置复用功能 I/O AFSELy 选择: 000:AF0 001:AF1 010:AF2 011:AF3 100:AF4 101:AF5 110:AF6 111:AF7
27	保留			
26:24	AFSELy[2:0](y= 7 to 0)	RW	**3'h0**	软件可写这些位配置复用功能 I/O AFSELy 选择: 000:AF0 001:AF1

Bit	Name	R/W	Reset Value	Function
				010:AF2 011:AF3 100:AF4 101:AF5 110:AF6 111:AF7
23	保留			
22:20	AFSELy[2:0](y= 7 to 0)	RW	**3'h0**	软件可写这些位配置复用功能 I/O AFSELy 选择: 000:AF0 001:AF1 010:AF2 011:AF3 100:AF4 101:AF5 110:AF6 111:AF7
19	保留			
18:16	AFSELy[2:0]（(y= 7 to 0)）	RW	**3'h0**	软件可写这些位配置复用功能 I/O AFSELy 选择: 000:AF0 001:AF1 010:AF2 011:AF3 100:AF4 101:AF5 110:AF6 111:AF7
15	保留			
14:12	AFSELy[2:0](y= 7 to 0)	RW	**3'h0**	软件可写这些位配置复用功能 I/O AFSELy 选择: 000:AF0 001:AF1 010:AF2 011:AF3 100:AF4 101:AF5 110:AF6 111:AF7
11	保留			
10:8	AFSELy[2:0](y= 7 to 0)	RW	**3'h0**	软件可写这些位配置复用功能 I/O AFSELy 选择: 000:AF0 001:AF1 010:AF2 011:AF3 100:AF4 101:AF5 110:AF6 111:AF7

Bit	Name	R/W	Reset Value	Function
11	保留			
10:8	AFSELy[2:0](y= 7 to 0)	RW	**3'h0**	软件可写这些位配置复用功能 I/O AFSELy 选择: 000:AF0 001:AF1 010:AF2 011:AF3 100:AF4 101:AF5 110:AF6 111:AF7
7	保留			
6:4	AFSELy[2:0](y= 7 to 0)	RW	**3'h0**	软件可写这些位配置复用功能 I/O AFSELy 选择: 000:AF0 001:AF1 010:AF2 011:AF3 100:AF4 101:AF5 110:AF6 111:AF7
3	保留			
2:0	AFSELy[2:0](y= 7 to 0)	RW	**3'h0**	软件可写这些位配置复用功能 I/O AFSELy 选择: 000:AF0 001:AF1 010:AF2 011:AF3 100:AF4 101:AF5 110:AF6 111:AF7

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

偏移地址： 0x24

复位值： 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	AFSEL15[2:0]			Res.	AFSEL14[2:0]			Res.	AFSEL13[2:0]			Res.	AF-SEL12[2:0]		
	RW	RW	RW		RW	RW	RW		RW	RW	RW		RW	RW	RW
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	AFSEL11[3:0]			Res.	AFSEL10[2:0]			Res.	AFSEL9[2:0]			Res.	AFSEL8[2:0]		
	RW	RW	RW		RW	RW	RW		RW	RW	RW		RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31	保留			
30:28	AFSELy[2:0] (y= 8 to 15)	RW		软件可写这些位配置复用功能 I/O

Bit	Name	R/W	Reset Value	Function
				AFSELy 选择: 000:AF0 001:AF1 010:AF2 011:AF3 100:AF4 101:AF5 110:AF6 111:AF7
27	保留			
26:24	AFSELy[2:0]（y= 8 to 15）	RW		软件可写这些位配置复用功能 I/O AFSELy 选择: 000:AF0 001:AF1 010:AF2 011:AF3 100:AF4 101:AF5 110:AF6 111:AF7
23	保留			
22:20	AFSELy[2:0]（y= 8 to 15）	RW		软件可写这些位配置复用功能 I/O AFSELy 选择: 000:AF0 001:AF1 010:AF2 011:AF3 100:AF4 101:AF5 110:AF6 111:AF7
19	保留			
18:16	AFSELy[2:0]（y= 8 to 15）	RW		软件可写这些位配置复用功能 I/O AFSELy 选择: 000:AF0 001:AF1 010:AF2 011:AF3 100:AF4 101:AF5 110:AF6 111:AF7
15	保留			
14:12	AFSELy[2:0]（y= 8 to 15）	RW		软件可写这些位配置复用功能 I/O AFSELy 选择: 000:AF0 001:AF1 010:AF2 011:AF3 100:AF4

Bit	Name	R/W	Reset Value	Function
				101:AF5 110:AF6 111:AF7
11	保留			
10:8	AFSELy[2:0]（y= 8 to 15)	RW		软件可写这些位配置复用功能 I/O AFSELy 选择: 000:AF0 001:AF1 010:AF2 011:AF3 100:AF4 101:AF5 110:AF6 111:AF7
11	保留			
10:8	AFSELy[2:0]（y= 8 to 15)	RW		软件可写这些位配置复用功能 I/O AFSELy 选择: 000:AF0 001:AF1 010:AF2 011:AF3 100:AF4 101:AF5 110:AF6 111:AF7
7	保留			
6:4	AFSELy[2:0]（y= 8 to 15)	RW		软件可写这些位配置复用功能 I/O AFSELy 选择: 000:AF0 001:AF1 010:AF2 011:AF3 100:AF4 101:AF5 110:AF6 111:AF7
3	保留			
2:0	AFSELy[2:0]（y= 8 to 15)	RW		软件可写这些位配置复用功能 I/O AFSELy 选择: 000:AF0 001:AF1 010:AF2 011:AF3 100:AF4 101:AF5 110:AF6 111:AF7

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

偏移地址： 0x28

复位值： 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31:16	保留	-	-	保留
15:0	BRy	RW	**16'h0**	y = 15..0 这些位软件可写，读出来返回值是 0 0: 对对应的 ODy 位不产生影响 1: 清除对应的 ODy 位

9. 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：

- I^2C IO 相关功能
- IO Port 滤波使能控制
- 根据不同 boot 模式，映射初始程序区
- 管理 TIMERS ETR 以及是刹车输入

9.1. 系统配置寄存器

9.1.1. SYSCFG 配置寄存器 1(SYSCFG_CFGR1)

该寄存器用作存储器请求 remap 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 remap，并 by-pass 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。

偏移地址：0x00

复位值：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.	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.	ETR_SRC_TIM1[1:0]		Res.	Res.	Res.		TIM1_IC1_SRC		MEM_MODE[1:0]	
						RW	RW					RW	RW	RW	

Bit	Name	R/W	Reset Value	Function
31:10	保留	-	-	保留
9:8	ETR_SRC_TIM1[1:0]	RW	3’b00	TIMER1 ETR 输入源选择。 2’b00: ETR 来源于 GPIO； 2’b01: ETR 来源于 COMP1 输出； 2’b10: ETR 来源于 COMP2 输出； 2’b11: ETR 来源于 ADC 模拟看门狗输出；
7:6	保留	-	0	保留
3:2	TIM1_IC1_SRC	RW	00	TIM1 CH1 输入来源： 00,11：from TIM1_CH1 IO； 01：from COMP1; 10：from COMP2.
1:0	MEM_MODE[1:0]			内映映射选择位 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，这些位采用实际启动模式配置值。 X0：Main flash, 映射在 0x0000 0000 01：Load flash , 映射在 0x0000 0000 11：SRAM, 映射在 0x0000 0000

9.1.2. SYSCFG 配置寄存器 2 (SYSCFG_CFGR2)

偏移地址：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	COMP2_Ocref_CLR_TIM1	Res	COMP1_Ocref_CLR_TIM1	Res	Res	Res	Res	COMP2_BRK_TIM1	COMP1_BRK_TIM1	Res	Res	LOCKUP_LOCK
				RW		RW					RW	RW			RW

Bit	Name	R/W	Reset Value	Function
31:12	保留	-	-	保留
11	COMP2_Ocref_CLR_TIM1	RW	1’b0	1：COMP2 输出作为 TIM1 ocref_clr 输入； 0：COMP2 输出不作为 TIM1 ocref_clr 输入
10	保留			
9	COMP1_Ocref_CLR_TIM1	RW	1’b0	1：COMP1 输出作为 TIM1 ocref_clr 输入； 0：COMP1 输出不作为 TIM1 ocref_clr 输入
8:5	保留	-	-	-
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	保留	-	-	-
0	LOCKUP_LOCK	RW	0	Cortex-M0+ LOCKUP 位 lock 使能位。 0：Cortex-M0+ LOCKUP 位不与 TIM1 break input 相连； 1：Cortex-M0+ LOCKUP 位与 TIM1 break input 相连

9.1.3. GPIOA 滤波使能寄存器（PA_ENS）

偏移地址：0x10

复位值：0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31:16	保留	-	-	保留
15:0	PA_ENS[15:0]	RW	0x0000	Noise filter enable, active high

				0: noise filter bypassed 1: noise filter enabled

9.1.4. GPIOB 滤波使能寄存器（PB_ENS）

偏移地址：0x14

复位值：0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
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.	PB_ENS[3:0]			
												RW			

Bit	Name	R/W	Reset Value	Function
31:8	保留	-	-	保留
3:0	PB_ENS[3:0]	RW	0	Noise filter enable, active high 0: noise filter bypassed 1: noise filter enabled

9.1.5. GPIOF 滤波使能寄存器（PF_ENS）

偏移地址：0x18

复位值：0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
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.	PF_ENS[5:0]					
										RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:6	保留	-	-	保留
5:0	PF_ENS[5:0]	RW	0	Noise filter enable, active high 0: noise filter bypassed 1: noise filter enabled

9.1.6. I²C 类型 IO 配置寄存器（SYSCFG_IOCFG）

偏移地址：0x1C

复位值：0x0000_0000

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

Bit	Name	R/W	Reset Value	Function
31:24	保留	-	-	保留
23:22	PB_ENSEG[1:0]	RW	0	PB ENSEG 信号控制。用作 SEG 模式控制。 PB2:bit0 PB3:bit1 用户可通过修改 GPIOB_OSPEEDR 的配置并确定 PB2~3 的驱动电流
21:16	PA_ENSEG[5:0]	RW	0	PA ENSEG 信号控制。用作 SEG 模式控制。 PA0:bit0 PA1:bit1 PA5:bit2 PA6:bit3 PA7:bit4 PA8:bit5 用户可通过修改 GPIOA_OSPEEDR 的配置并确定 PA0~1，5~8 的驱动电流
15	保留	-	-	保留
14:13	PA_IODRVP[1:0]	RW	0	PA IODRVP 信号控制。用作 I2C_VCC 的输出驱动器功能设置。(PA2、7 对应 bit0~1)
12:8	PA_EHS[4:0]	RW	0	PA EHS 信号控制。用作 80mA LED IO 控制。(PA11~15 对应 bit0~4)
7:6	PF_PU_IIC[1:0]	RW	0	I2C_PU 类型 IO 上拉电阻控制使能。（PF3~4 对应 bit0~1）
5	保留	-	-	保留
4:2	PF_EIIC[2:0]	RW	0	PF EIIC 信号控制。用作_I2C 类型 IO。（PF2~4 对应 bit0~2）
1:0	PA_EIIC[1:0]	RW	0	PA EIIC 信号控制。用作_I2C 类型 IO。（PA8 对应 bit0,PA11 对应 bit1）

9.1.7. GPIOA 模拟 2 使能寄存器（PA_ANA2EN）

偏移地址：0x20

复位值：0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PA_ANA2EN[15:0]															
RW															

Bit	Name	R/W	Reset Value	Function
31:16	保留	-	-	保留
15:0	PA_ANA2EN[15:0]	RW	0	PAD_ANA2 enable, active high, default 0 0: PAD_ANA2 switch is off 1: PAD_ANA2 switch is on

9.1.8. GPIOB 模拟 2 使能寄存器（PB_ANA2EN）

偏移地址：0x24

复位值：0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
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.	PB_ANA2EN[3:0]			
												RW			

Bit	Name	R/W	Reset Value	Function
31:4	保留	-	-	保留
3:0	PB_ANA2EN[3:0]	RW	0	PAD_ANA2 enable, active high, default 0 0: PAD_ANA2 switch is off 1: PAD_ANA2 switch is on

9.1.9. GPIOF 模拟 2 使能寄存器（PF_ANA2EN）

偏移地址：0x28

复位值：0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
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.	PF_ANA2EN[5:0]					
										RW					

Bit	Name	R/W	Reset Value	Function
31:6	保留	-	-	保留
5:0	PF_ANA2EN[5:0]	RW	0	PAD_ANA2 enable, active high, default 0 0: PAD_ANA2 switch is off 1: PAD_ANA2 switch is on

9.1.10. GPIOF 模拟 2 使能寄存器（PF_ANA2EN）

偏移地址：0x28

复位值：0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
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.	PF_ANA2EN[5:0]					
										RW					

Bit	Name	R/W	Reset Value	Function
31:6	保留	-	-	保留
5:0	PF_ANA2EN[5:0]	RW	0	PAD_ANA2 enable, active high, default 0 0: PAD_ANA2 switch is off 1: PAD_ANA2 switch is on

9.1.11. SRAM_RETVSEL 寄存器（SRAM_RETVSEL）

偏移地址：0x2C

复位值：0x0000_0000

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

Bit	Name	R/W	Reset Value	Function
31:1	保留	-	-	保留
0	SRAM_RETVSEL	RW	0	SRAM retention voltage configuration signal SRAM_RETVSEL=1b'1 Vsram provide from V_{DD} SRAM_RETVSEL=1b'0 Vsram provide from V_{DD1}

9.1.12. GPIOA 上下拉电阻配置（PA_IORP）

偏移地址： 0x30

复位值： 0xFFFF_FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PA15_ IORP[1:0]		PA14_ IORP[1:0]		PA13_ IORP[1:0]		PA12 IORP[1:0]		PA11 IORP[1:0]		PA10_ IORP[1:0]		PA9 IORP[1:0]		PA8 IORP[1:0]	
RW		RW		RW		RW		RW		RW		RW		RW	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PA7_ IORP[1:0]		PA6_ IORP[1:0]		PA5 IORP[1:0]		PA4_ IORP[1:0]		PA3_ IORP[1:0]		PA2_ IORP[1:0]		PA1_ IORP[1:0]		PA0 IORP[1:0]	
RW		RW		RW		RW		RW		RW		RW		RW	

Bit	Name	R/W	Reset Value	Function
31：30	PA15_ IORP[1:0]	RW	2'b11	同 PA0_ IORP[1:0]
29：28	PA14_ IORP[1:0]	RW	2'b11	同 PA0_ IORP[1:0]
27：26	PA13_ IORP[1:0]	RW	2'b11	同 PA0_ IORP[1:0]
25：24	PA12_ IORP[1:0]	RW	2'b11	同 PA0_ IORP[1:0]
23：22	PA11_ IORP[1:0]	RW	2'b11	同 PA0_ IORP[1:0]
21：20	PA10_ IORP[1:0]	RW	2'b11	同 PA0_ IORP[1:0]
19：18	PA9_ IORP[1:0]	RW	2'b11	同 PA0_ IORP[1:0]
17：16	PA8_ IORP[1:0]	RW	2'b11	同 PA0_ IORP[1:0]
15：14	PA7_ IORP[1:0]	RW	2'b11	同 PA0_ IORP[1:0]
13：12	PA6_ IORP[1:0]	RW	2'b11	同 PA0_ IORP[1:0]
11：10	PA5_ IORP[1:0]	RW	2'b11	同 PA0_ IORP[1:0]
9：8	PA4_ IORP[1:0]	RW	2'b11	同 PA0_ IORP[1:0]
7：6	PA3_ IORP[1:0]	RW	2'b11	同 PA0_ IORP[1:0]
5：4	PA2_ IORP[1:0]	RW	2'b11	同 PA0_ IORP[1:0]
3：2	PA1_ IORP[1:0]	RW	2'b11	同 PA0_ IORP[1:0]
1：0	PA0_ IORP[1:0]	RW	2'b11	IO pull-up pull-down resistance config, default 11 00: pull-up/pull-down is open 01: pull-up/pull-down resistance 10 kΩ 10: pull-up/pull-down resistance 20 kΩ 11: pull-up/pull-down resistance 40 kΩ

9.1.13. GPIOB 上下拉电阻配置（PB_IORP）

偏移地址：0x34

复位值： 0x0000_00FF

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.	PB3_ IORP[1:0]		PB2_ IORP[1:0]		PB1_ IORP[1:0]		PB0_ IORP[1:0]	
								RW		RW		RW		RW	

Bit	Name	R/W	Reset Value	Function
31：8	保留	-	-	保留
7：6	PB3_ IORP[1:0]	RW	2’b11	同 PB0_ IORP[1:0]
5：4	PB2_ IORP[1:0]	RW	2’b11	同 PB0_ IORP[1:0]
3：2	PB1_ IORP[1:0]	RW	2’b11	同 PB0_ IORP[1:0]
1：0	PB0_ IORP[1:0]	RW	2’b11	IO pull-up pull-down resistance config, default 11 00: pull-up/pull-down is open 01: pull-up/pull-down resistance 10 kΩ 10: pull-up/pull-down resistance 20 kΩ 11: pull-up/pull-down resistance 40 kΩ

9.1.14. GPIOF 上下拉电阻配置（PF_IORP）

偏移地址：0x38

复位值： 0x0000_0FFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	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.	PF_ANA2EN[5:0]					
										RW					

Bit	Name	R/W	Reset Value	Function
31：12	保留	-	-	保留
11：10	PF5_ IORP[1:0]	RW	2’b11	同 PF0_ IORP[1:0]
9：8	PF4_ IORP[1:0]	RW	2’b11	同 PF0_ IORP[1:0]
7：6	PF3_ IORP[1:0]	RW	2’b11	同 PF0_ IORP[1:0]
5：4	PF2_ IORP[1:0]	RW	2’b11	同 PF0_ IORP[1:0]
3：2	PF1_ IORP[1:0]	RW	2’b11	同 PF0_ IORP[1:0]
1：0	PF0_ IORP[1:0]	RW	2’b11	IO pull-up pull-down resistance config, default 11 00: pull-up/pull-down is open 01: pull-up/pull-down resistance 10 kΩ 10: pull-up/pull-down resistance 20 kΩ 11: pull-up/pull-down resistance 40 kΩ

10. 中断和事件

10.1. 嵌套向量中断控制器(NVIC)

10.1.1. 主要特性

- 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。

10.1.2. 系统嘀嗒（SysTick）校准值寄存器

SysTick calibration 值被设为 6000，通过 SysTick 时钟置为 6 MHz（Max f_{HCLK}/8），给出了 1ms 的参考 time base。

在进入 sleep 或者 stop 低功耗前，需要关闭 systick 的中断。

10.1.3. 中断和异常向量

表 10-1 中断异常和向量表

Position	Priority	Types of Priority	Acronym	Description	Address
-	-	-	-	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	可设置	RTC	RTC 中断(combined EXTI lines 19)	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

Position	Priority	Types of Priority	Acronym	Description	Address
8	15	可设置	TK	TK int	0x0000_0060
9	16	-	保留	保留	0x0000_0068
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 刹车、更新、触发和 COM 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	-	保留	保留	0x0000_0084
18	25	-	保留	保留	0x0000_0088
19	26	可设置	TIM14	TIM14 全局中断	0x0000_008C
20	27	-	保留	保留	0x0000_0090
21	28	-	保留	保留	0x0000_0094
22	29	-	保留	保留	0x0000_0098
23	30	可设置	I^2C1	I^2C1 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	-	保留	保留	0x0000_00AC
28	35	-	保留	保留	0x0000_00B0
29	36	可设置	UART3	UART3 全局中断	0x0000_00B4
30	37	可设置	UART2	UART2 全局中断	0x0000_00B8
31	38	可设置	UART1	UART1 全局中断	0x0000_00BC

1. The grayed cells（the address less than 0x0000 0040） correspond to the Cortex®-M0+ interrupts.

10.2. 外部中断/事件控制器(EXTI)

扩展中断和事件控制器，通过 configurable（可配置）和 direct（直接事件）输入(Lines)，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（18 个 configurable 事件 Line 和 3 个 direct 事件 line）。

10.2.1. EXTI 主要特性

系统可以通过 GPIO 和指定模块（COMP/RTC/I2C/TK）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

10.2.2. EXTI 框图

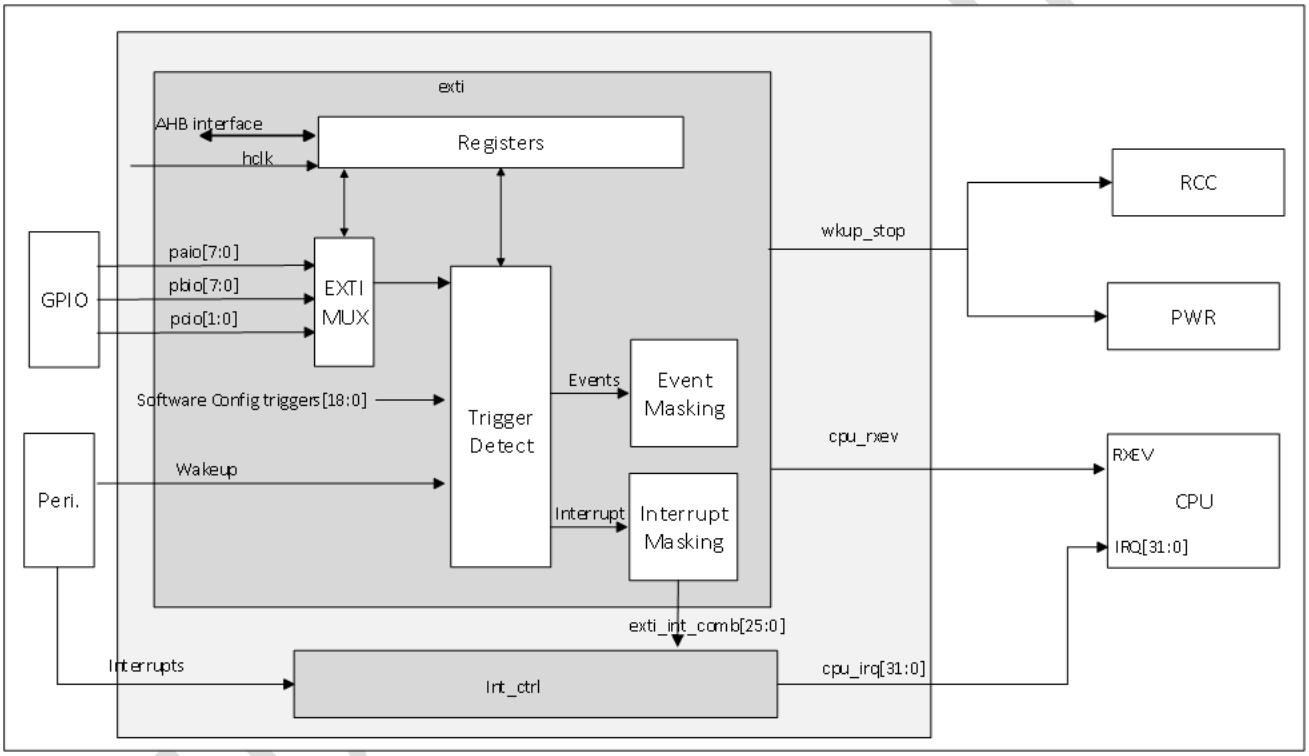


图 10-1 EXTI 框图

10.2.3. 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。

10.2.4. EXTI 可配置事件（configurable）触发唤醒

通过配置 EXTI_SWIER 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。

10.2.5. 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。

10.2.6. EXTI 选择器

GPIO 被用以下方式连接到 16 个外部中断/事件 line 上：

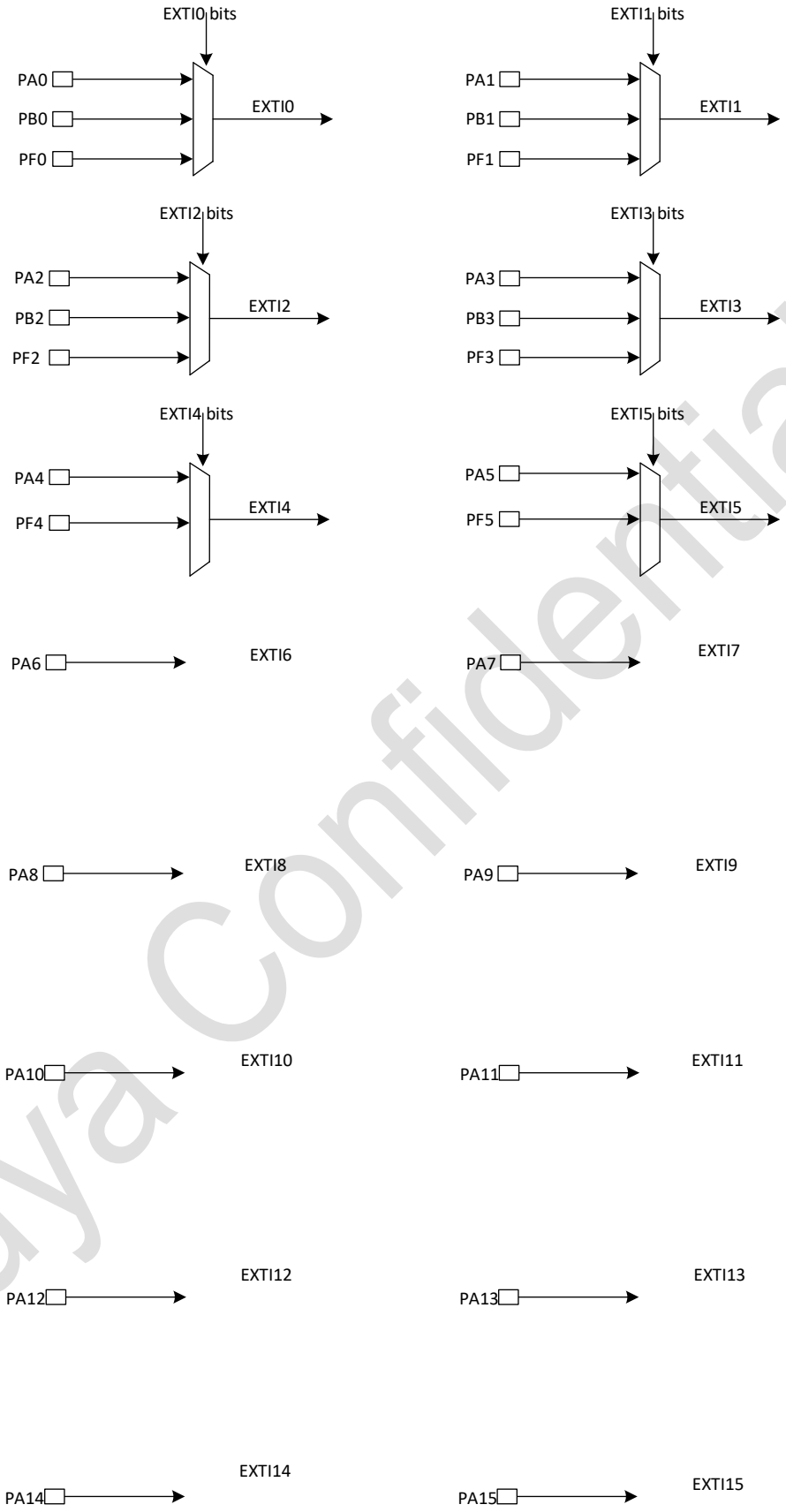


图 10-2 外部中断/事件 GPIO 映像

所有 line 连接内容如下表所示：

表 10-2 外部中断/事件 Line 连接

EXTI line	Line source	Line type
Line 0-15	GPIO	Configurable
Line 16	保留	-
Line 17	COMP 1 output	Configurable
Line 18	COMP 2 output	Configurable
Line 19	RTC	Direct
Line 20	保留	-
Line 21	保留	-
Line 22	保留	-
Line 23	I^2C(i2c_address match see i2c ip spec)	Direct
Line 24	保留	-
Line 25	保留	-
Line 26	保留	-
Line 27	保留	-
Line 28	TK(tk touch yes see tk ip spec)	Direct

10.3. EXTI 寄存器

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

10.3.1. 上升沿触发选择寄存器 (EXTI_RTSR)

偏移地址： 0x00

复位值： 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	RT18	RT17	Res
													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	保留	-	-	保留
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	保留	-	-	保留
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。

Bit	Name	R/W	Reset Value	Function
				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 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。

10.3.2. 下降沿触发选择寄存器 (EXTI_FTSR)

偏移地址： 0x04

复位值： 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	Res
													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	保留	-	-	保留
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	保留	-	-	保留
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。

10.3.3. 软件中断事件寄存器 (EXTI_SWIER)

偏移地址： 0x08

复位值： 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.	SW18	SW17	Res.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
													RW	RW	
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
31:19	保留	-	-	保留
18	SWI18	RW	0	Configurable 类型 EXTI line18 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）
17	SWI17	RW	0	Configurable 类型 EXTI line17 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.
16	保留	-	-	保留
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.
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。

Bit	Name	R/W	Reset Value	Function
				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 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

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）

10.3.4. 挂起寄存器(EXTI_PR)

偏移地址： 0x0C

复位值： 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.	PR18	PR17	Res.
													RC_W1	RC_W1	
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
PR15	PR14	PR13	PR12	PR11	PR10	PR9	PR8	PR7	PR6	PR5	PR4	PR3	PR2	PR1	PR0
RC_W1	RC_W1	RC_W1	RC_W1	RC_W1	RC_W1	RC_W1	RC_W1	RC_W1	RC_W1	RC_W1	RC_W1	RC_W1	RC_W1	RC_W1	RC_W1

Bit	Name	R/W	Reset Value	Function
31:19	保留	-	-	保留
18	PR18	RC_W1	0	Configurable 类型 EXTI line18 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；
17	PR17	RC_W1	0	Configurable 类型 EXTI line17 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；
16	保留	-	-	保留
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；

Bit	Name	R/W	Reset Value	Function
				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； 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；

Bit	Name	R/W	Reset Value	Function
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	RPIF2	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；

10.3.5. 外部中断选择寄存器 1 (EXTI_EXTICR1)

偏移地址：0x60

复位值：0x0000

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

Bit	Name	R/W	Reset Value	Function
23:18	保留	-	-	保留
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: 保留
15:10	保留	-	-	保留
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: 保留
7:2	保留	-	-	保留
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: 保留

10.3.6. 外部中断选择寄存器 2 (EXTI_EXTICR2)

偏移地址：0x64

复位值：0x0000

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

Bit	Name	R/W	Reset Value	Function
31:10	保留	-	-	保留
9:8	EXTI5[1:0]	RW	0	EXTI5 对应 GPIO port 选择。 2'b00: PA[5] pin 2'b01: PB[5] pin 2'b10: PF[5] pin 2'b11: 保留
7:2	保留	-	-	保留
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: 保留

10.3.7. 中断屏蔽寄存器(EXTI_IMR)

偏移地址：0x80

复位值： 0x1088 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.	Res.	IM28	Res.	Res.	Res.	Res.	IM23	Res.	Res.	Res.	IM19	IM18	IM17	Res.
			RW					RW				RW	RW	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
31:29	保留			
28	IM28	RW	1	EXTI line28 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
27:24	保留			
23	IM23	RW	1	EXTI line23 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
22:20	保留			
19	IM19	RW	1	EXTI line19 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
18	IM18	RW	0	EXTI line18 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
17	IM17	RW	0	EXTI line17 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
16	保留			
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

Bit	Name	R/W	Reset Value	Function
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。 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

10.3.8. 事件屏蔽寄存器(EXTI_EMR)

偏移地址： 0x84

复位值： 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	EM28	Res.	Res.	Res.	Res.	EM23	Res.	Res.	Res.	EM19	EM18	EM17	Res
			RW					RW				RW	RW	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
31:29	保留			
28	EM28	RW	0	EXTI line28 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
27:24	保留			
23	EM23	RW	0	EXTI line23 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
22:20	保留			
19	EM19	RW	0	EXTI line19 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
18	EM18	RW	0	EXTI line18 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
17	EM17	RW	0	EXTI line17 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
16	保留			
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

Bit	Name	R/W	Reset Value	Function
				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 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. 循环冗余校验(CRC)

11.1. 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。

在其他的应用中， CRC 技术主要应用于核实数据传输或者数据存储的正确性和完整性。

11.2. 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
- 通用的 8 位寄存器（可被用作临时存储）
- 计算时间：32 bits 数据 4 个 AHB 时钟

11.3. CRC 功能描述

11.3.1. CRC 框图

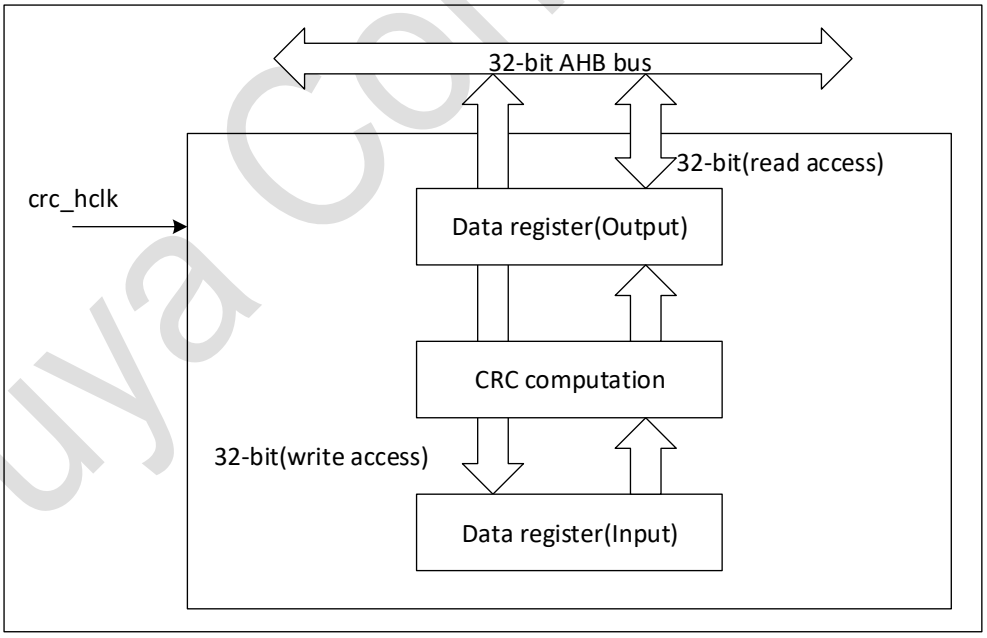


图 11-1 CRC 计算单元框图

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。

11.4. CRC 寄存器

11.4.1. 数据寄存器 (CRC_DR)

偏移地址：0x00

复位值：0xFFFF FFFF

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

11.4.2. 独立数据寄存器(CRC_IDR)

偏移地址：0x04

复位值：0x0000 0000

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

11.4.3. 控制寄存器(CRC_CR)

偏移地址：0x08

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
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	保留		-	
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. 模拟/数字转换(ADC)

12.1. 简介

芯片具有 1 个 12 位的 SAR-ADC。该模块最多支持 13 个转换通道，包括 10 个外部通道和 3 个内部通道。参考电压可选择片内精准电压（0.6 V、1.5 V、2.048 V、2.5 V）或 V_{CC} 电源电压。

内部通道包括 T_{S_VIN}，V_{REFINT}，$V_{CC}/3$。

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。

模拟看门狗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。

在采样结束，转换结束，连续转换结束，模拟看门狗时转换电压超出阈值时产生中断请求。

12.2. ADC 主要特性

- 高性能
  - 12 bits、10 bits、8 bits 和 6 bits 分辨率可配置
  - ADC 转换时间：1.33 us@12bit（12 MHz）
  - 自校准（软件启动）
  - 可编程的采样时间
  - 可编程的数据对齐模式（左对齐或者右对齐）
- 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
  - 自动延迟转换模式：防止以低频 PCLK 运行产生溢出
- 模拟输入通道
  - 10 个外部模拟输入通道：PA[7:0] 和 PB[1:0]
  - 1 个内部 temperature sensor 通道
  - 1 个内部参考电压通道（V_{REFINT}）
  - 1 个内部通道（$V_{CC}/3$）
- 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）
- 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
  - 非连续模式(discontinuous mode)：每次触发，转换被选择的通道 1 次
- 中断产生

- 在单个通道采样结束
- 在单个通道转换结束
- 在序列转换结束
- 模拟看门狗事件
- 溢出事件

■ 模拟看门狗

12.3. ADC 功能描述

12.3.1. ADC 框图

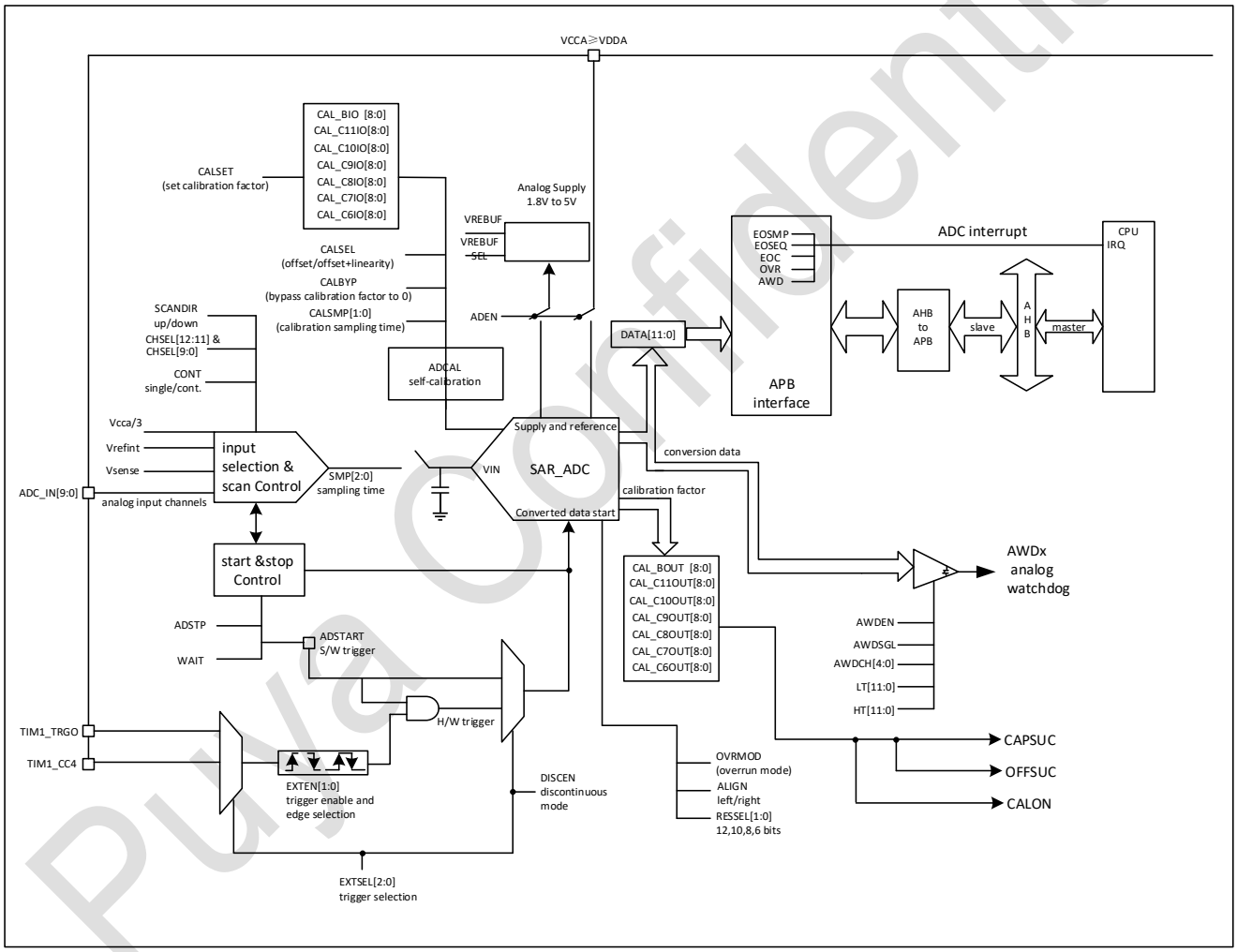


图 12-1 带模拟开关的 ADC 通道

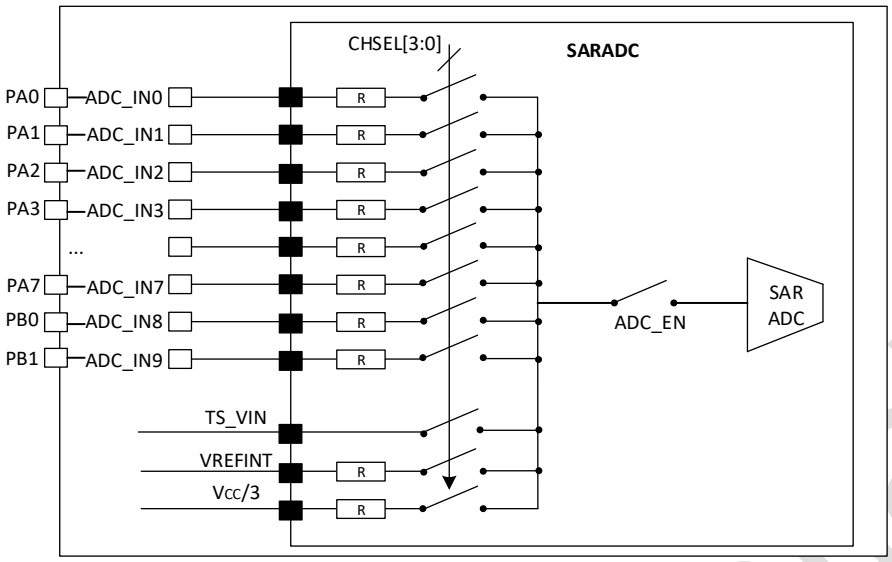


图 12-2 带模拟开关的 ADC 通道

12.3.2. 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。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。

较准操作包括软件较准。

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。校准完成后，可从 ADC_CALFACTOR 寄存器读出校准因子。校准因子会一直保持，直到产生系统复位。

当 ADC 的工作条件发生改变时（V_{CC} 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

12.3.3. 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)。

ADEN 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 ADSTART=0，ADEN=1，则可对 ADC_CR 中的 ADDIS 置 1 禁用 ADC。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)。

ADSTART、ADSTP、ADEN 和 ADDIS 寄存器的配置，符合以下原则：

- 软件在未配置 ADEN=1 时不能配置 ADSTART/ADSTP/ADDIS 任何一位为 1（硬件屏蔽），也即在配置 ADEN=1 时 ADSTART/ADSTP/ADDIS 必须都为 0，或者说在配置 ADSTART/ADSTP/ADDIS 为 1 时 ADEN=1；
- 在 ADSTART 为 0 时，软件无法置位 ADSTP（硬件屏蔽）；
- 在 ADEN=1 且 ADSTART 为 0 时，软件才能配置 ADDIS=1（硬件屏蔽）；
- 若 ADEN=1，则置位 ADDIS（ADSTART=0 为前提），会清零 ADEN；在 ADEN 清零后，ADDIS 也被清零；
- 软件将 ADEN 清零后再重新使能 ADEN 和 ADSTART 的间隔为 8 个 ADC_CLK 周期。

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。

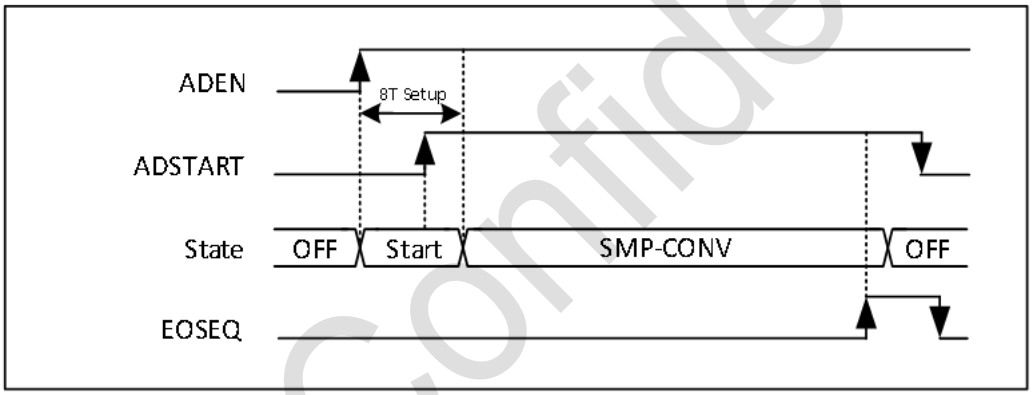


图 12-2 启用/禁用 ADC

12.3.4. 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。

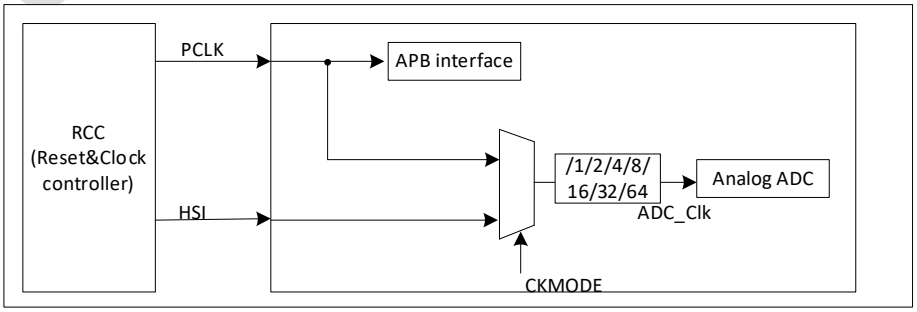


图 12-3 ADC 时钟方案

表 12-1 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	/	/

注:1. ADC_CLK>PCLK 时，建议打开 WAIT 模式；2. ADC_CLK>PCLK 时钟在不同转换精度的条件，ressel=00B ADC_CLK<4*PCLK;resel=01B ADC_CLK < 3PCLK;ressel=10B ADC_CLK < 3PCLK ; ressel = 11B ADC_CLK < 2PCLK

12.3.5. 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 开启且没有关闭请求挂起(ADEN=1)的情况下改写 ADC_CR 寄存器中的 ADSTART。

对于 ADC_IER、ADC_CFGRi、ADC_SMPR、ADC_TR 和 ADC_CCR 寄存器，软件必须在 ADC 开启 (ADEN = 1) 且无转换期间 (ADSTART = 0) 的情况下才能进行改写。ADC_CHSELR 是在 ADEN=0 且 ADSTART=0 的情况下改写。

ADC_CHSELR 是在 ADEN=0 且 ADSTART=0 的情况下改写。

软件必须在 ADC 开启且无挂起请求 (ADSTART = 1) 的情况下改写 ADC_CR 寄存器中的 ADSTP 位。

12.3.6. 通道选择 (CHSEL, SCANDIR)

共有 13 路复用通道：

- 10 个从 GPIO 引脚引入的模拟输入 (ADC_IN0...ADC_IN9)
- 3 个内部模拟输入(温度传感、内部参考电压和 V_{CC}/3)

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 12
- SCANDIR=1: 回退扫描：从通道 12 到通道 0

温度传感连接到 ADC_IN10（T_{S_VIN}） 通道，内部参考电压连接到 ADC1_IN11 通道（V_{REFINT}）。V_{CC}/3 连接到 ADC1_IN12。

12.3.7. 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

例如：

当 ADC_CLK = 12 MHz，分辨率为 12 位，且采样时间为 3.5 个 ADC 时钟周期：

t_{CONV} = （3.5 + 12.5） x ADC 时钟周期= 16 x ADC 时钟周期 =1.33 μs

EOSMP 标志位用来表明采样阶段的结束。

12.3.8. 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。

所有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。转换模式切换要先 disable aden(用 addis disable aden).

12.3.9. 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。转换模式切换要先 disable aden(用 addis disable aden).

12.3.10. 非连续转换模式 (DISCEN=1)

非连续转换模式由设置 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 中 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。转换模式切换要先 disable aden(用 addis disable aden).

12.3.11. 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 ADC 处于空闲时，该位可重新配置为 ADSTART=0 。

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。软件触发模式，START 控制位还保持 1 状态，再次软件触发是无效的。

12.3.12. 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。

$t_{ADC} = tS_{MPL} + t_{SAR}$ = [3.5 |min + 12.5 |12bit] x t_{ADC_CLK}

$t_{ADC} = t_{SMPL} + t_{SAR}$ = 291.7ns |min + 1041.625 ns |12bit = 1.33 µs |min (f_{ADC_CLK} = 12 MHz)

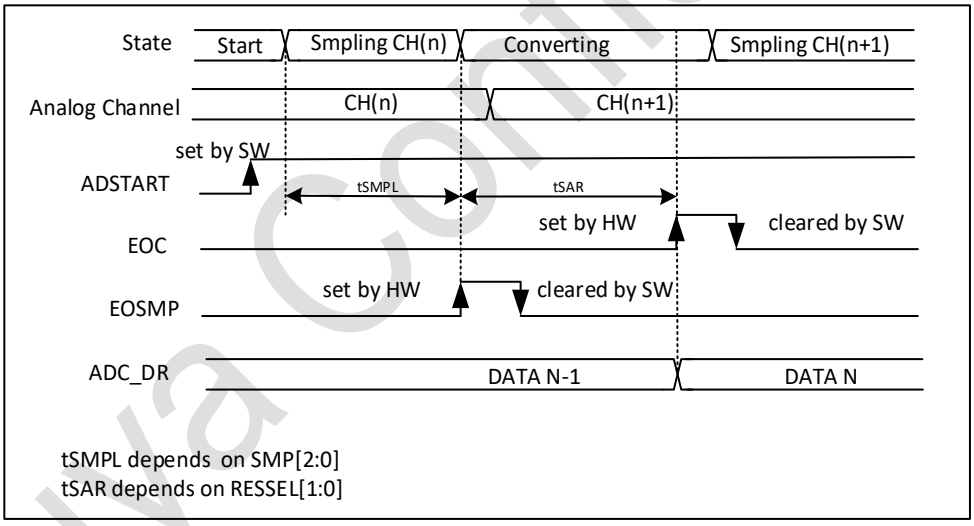


图 12-4 模数转换时序

12.3.13. 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。

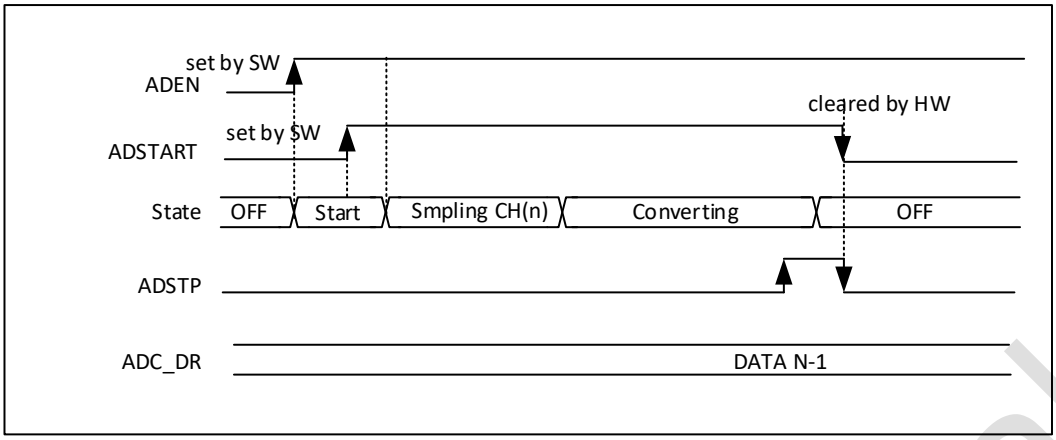


图 12-5 停止时序

12.3.14. 停止 ADEN(ADDIS)

用软件设置 ADC_CR 寄存器中的 ADDIS=1 可以清除 ADEN，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 ADDIS 由软件设置为 1(ADSTART=0)，ADEN 清零

一旦结束该过程 ADDIS 和 ADEN 位都由硬件清 0。

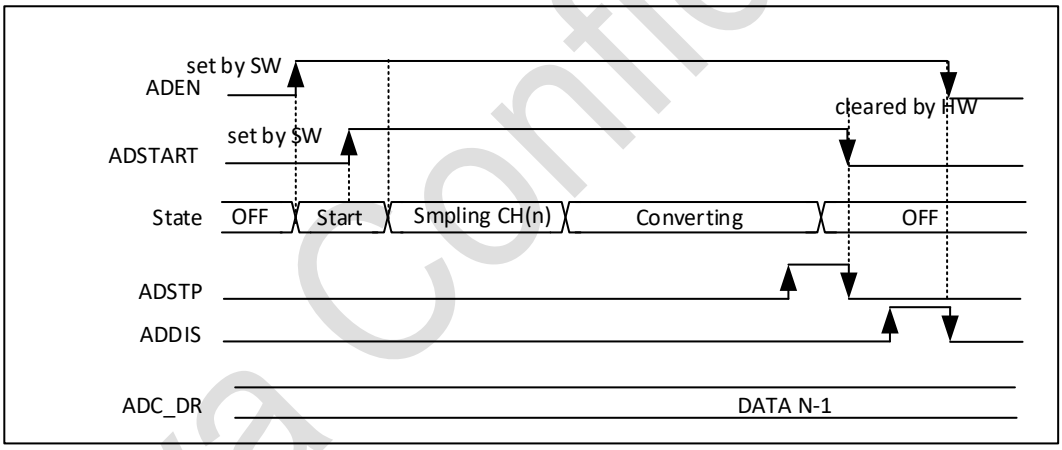


图 12-6 ADDSI 时序

12.3.15. 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、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。

表 12-2 外部触发

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 12-3 外部触发

Name	Source	EXTSEL[2:0]
EXT0	TIM1_TRGO	000
EXT1	TIM1_CC4	001
EXT2	TIM14_CC0	010
EXT3	保留	011
EXT4	保留	100
EXT5	保留	101
EXT6	保留	110
EXT7	保留	111

注： 在转换时外部触发源不能改变。

12.3.15.1. 快速转换模式

用降低转换分辨率来获取更快的转换时间 (tSAR) 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：

表 12-4 转换分辨率与转换时间

RESSEL [1:0]	t_{SAR} (ADC 时钟周期)	t_{SAR}(ns) @ f_{ADC} = 12MHz	t_{SMP} (ADC 时钟周期)	t_{ADC}(t_{SMP} = 3.5) (ADC 时钟周期)	t_{SAR}(ns) @ f_{ADC} = 12MHz
12	12.5	1041.6ns	3.5	16	1333.3ns
10	10.5	833.3ns	3.5	14	1166.2ns
8	8.5	708.3ns	3.5	12	1000ns
6	6.5	541.65ns	3.5	10	833.3ns

12.3.15.2. 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。

12.3.15.3. 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。

12.3.15.4. 采样时间图

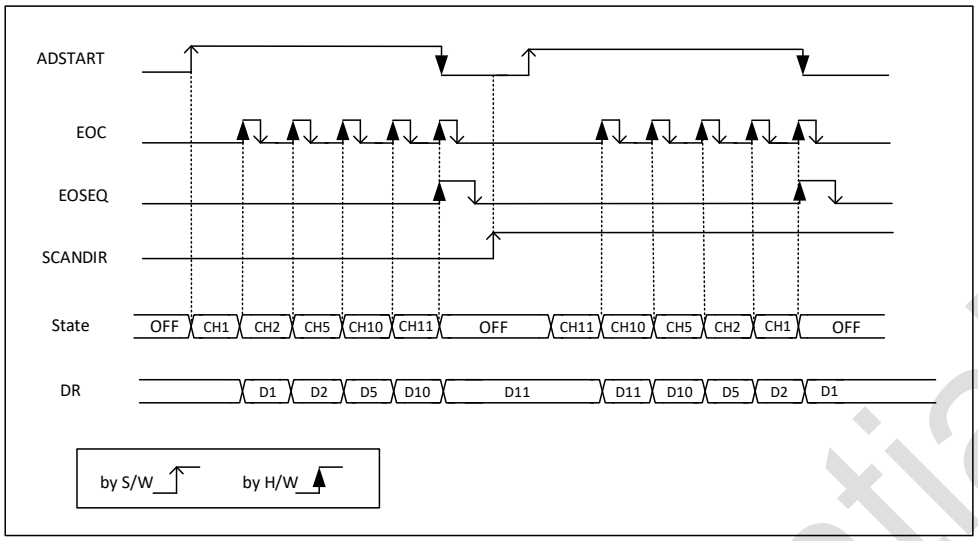


图 12-7 序列的单次转换, 软件触发

1. EXTEN=0x0, CONT=0
2. CHSEL=0x20601, WAIT=0
3. ADC_CLK 高分频的条件下，START 软件触发下一次转换，需要看 START 的状态（START:1->0 ,当前转换结束），下一次软件触发才能有效

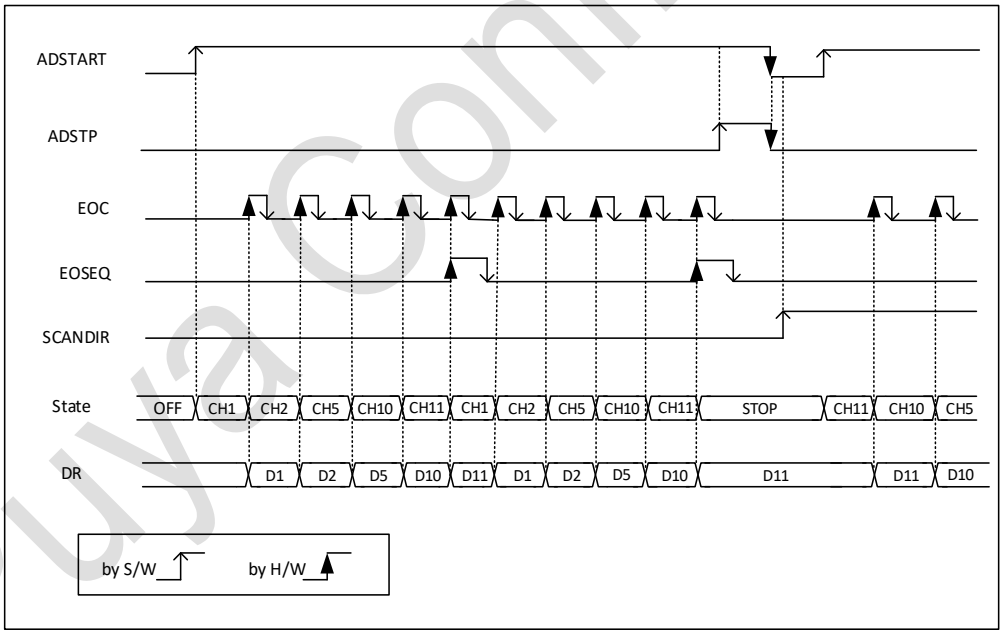


图 12-8 序列的连续转换, 软件触发

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

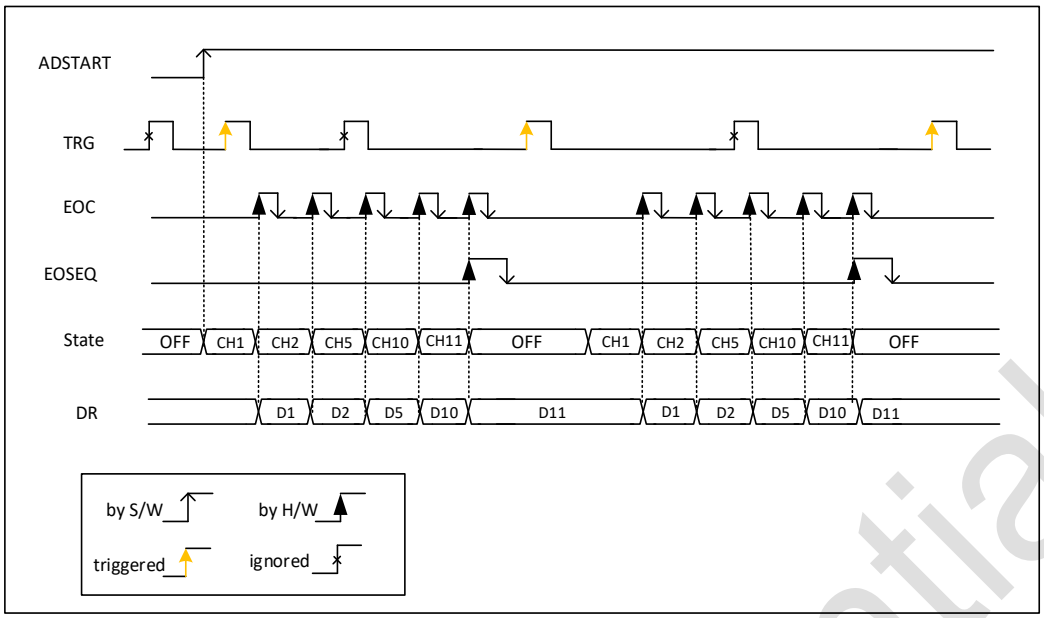


图 12-9 序列的单次转换, 硬件触发

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

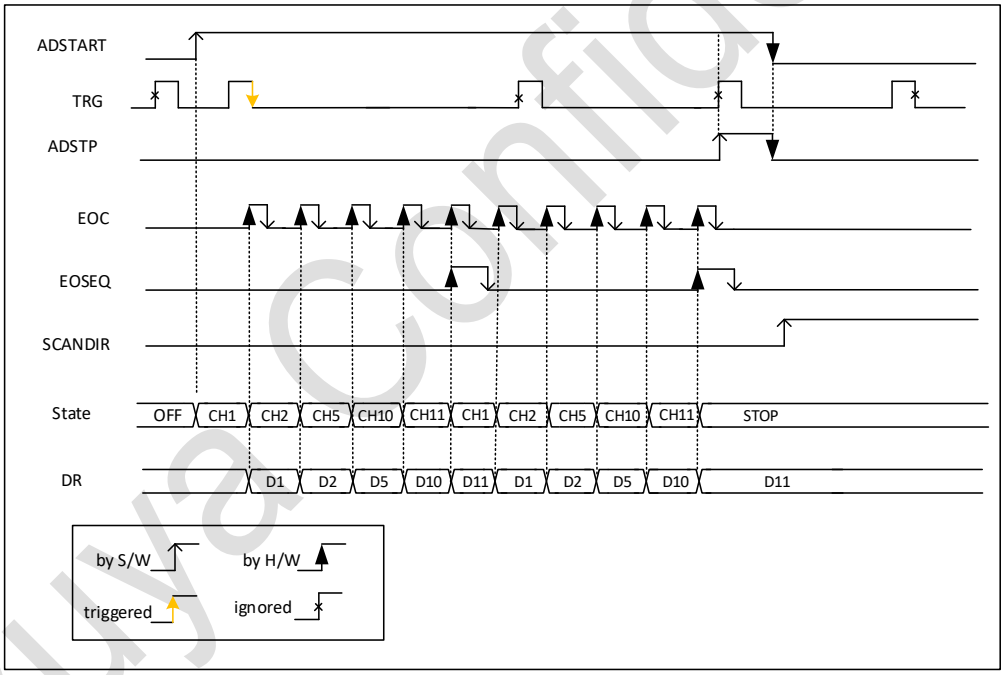


图 12-10 序列的连续转换，硬件触发

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

12.3.16. 数据管理

12.3.16.1. 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)。

表 12-5 数据寄存器和数据对齐

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			

12.3.16.2. 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（发生 OVR 之后，Read DR 不能清掉 OVR）。
- 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（发生 OVR 之后，Read DR 不能清掉 OVR）。

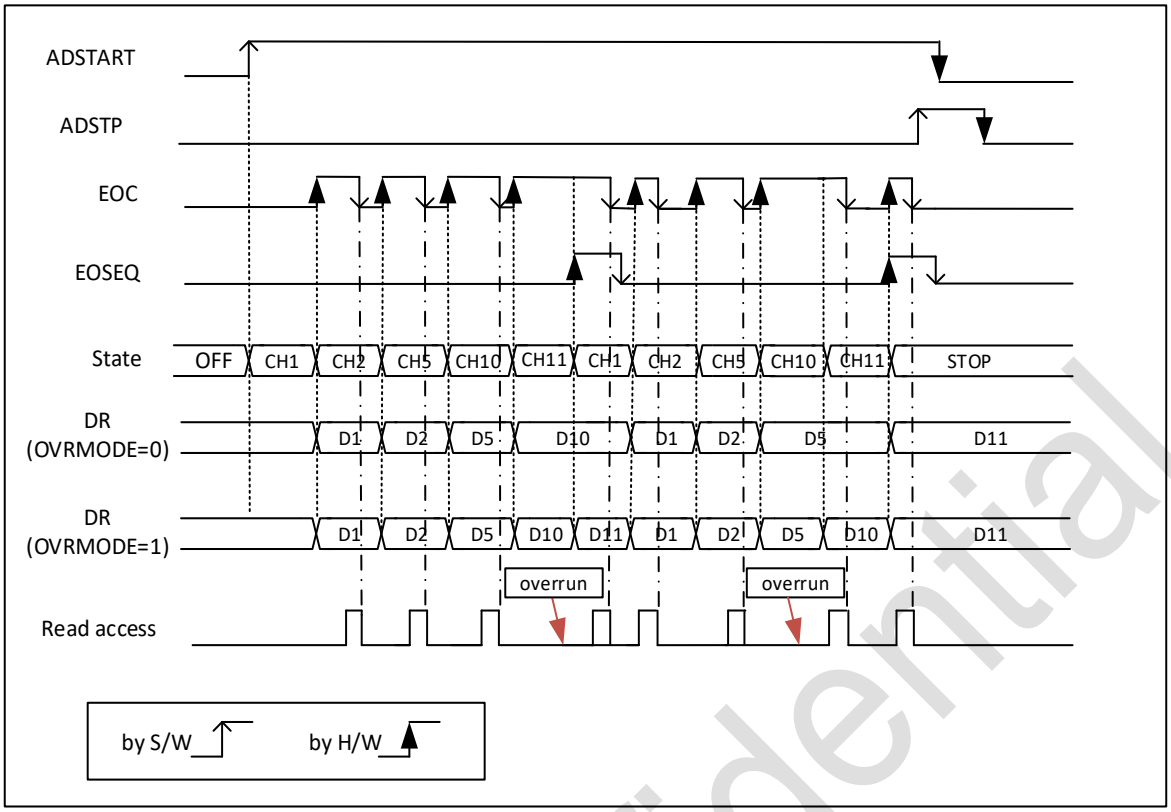


图 12-11 过载

12.3.17. 低功耗特性

12.3.17.1. 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。当 ADC_CLK 时钟>PCLK 时钟，建议打开 WAIT 。

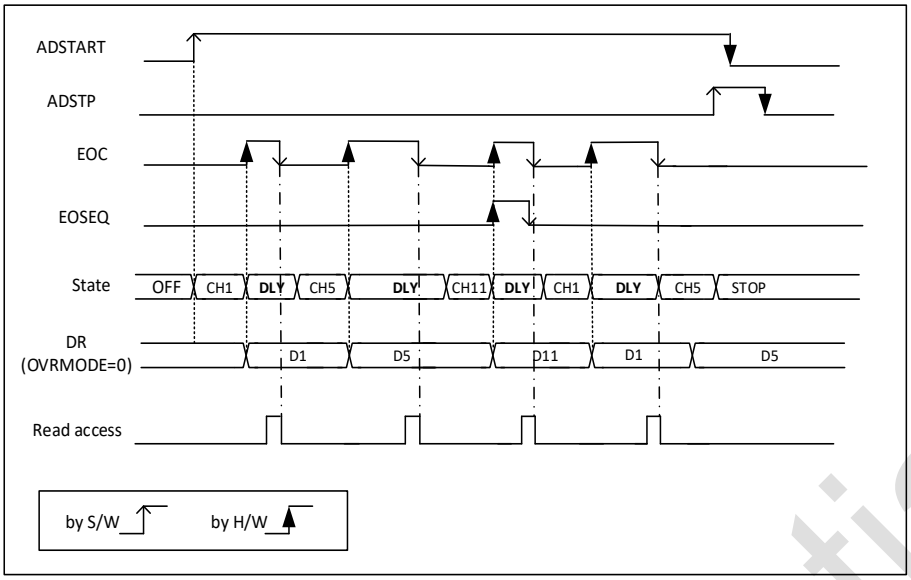


图 12-12 自动延迟转换模式

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

12.3.18. 模拟看门狗

AWD 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 12-6 模拟看门狗比较

分辨率位数	模拟看门狗之间的比较		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

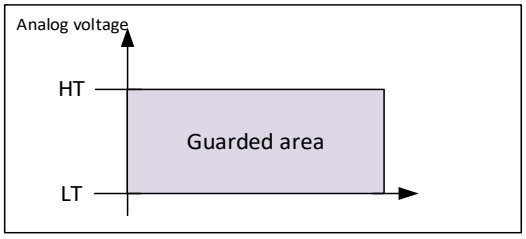


图 12-13 模拟看门狗保护区

表 12-7 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

12.3.18.1. 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（将 ADDIS 设置为 1 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。

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。

12.3.19. 温度传感器和内部参考电压

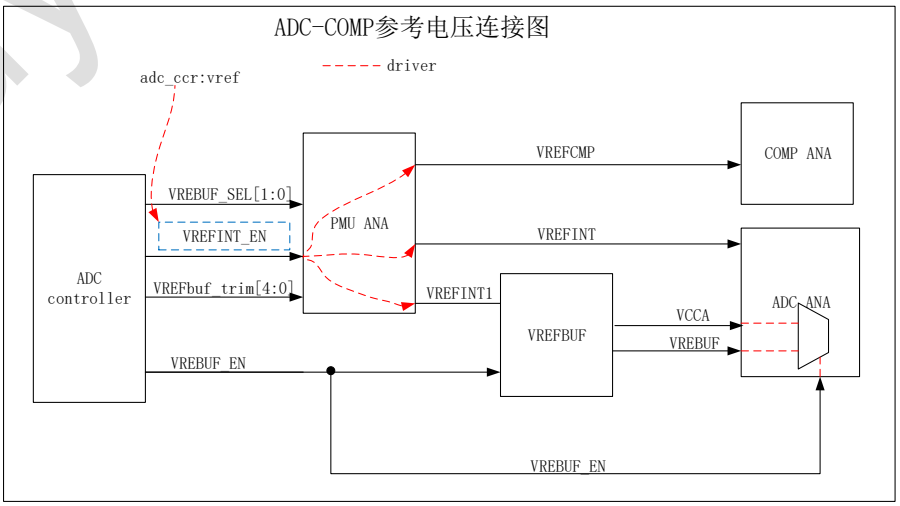


图 12-14 ADC 参考电压连接图

温度传感器可以用来测量器件的结点温度 (T_J)。

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 Tsamp_setup 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（V_{REFINT}）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、V_{REFINT}。

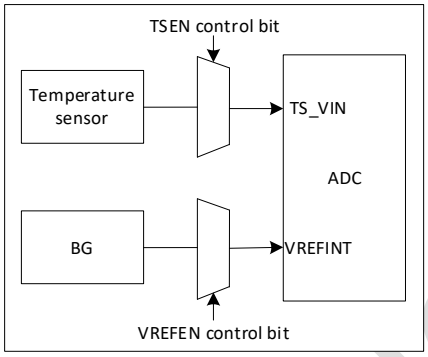


图 12-15 温度传感器和参考电压通道

如何用温度传感器读温度：

1. 选择 ADC1_IN8 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. 用下列公式计数温度：

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

TSCAL2 代表 85 ℃温度传感器的校准值

TSCAL1 代表 30 ℃温度传感器的校准值

TSDATA 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 VSENSE 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， 则需要在同一时间的设置 ADEN 和 TSEN 位。

利用内部的参考电压计算实际的 V_{CC} 电压

下面公式可以给出真实 V_{CC} 的电压值：

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

利用 Vcc 电压来计算 Vchannnel：

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

V_{REFINT} 固定值为 1.2 V；

$V_{CHANNEL}$ 是通道电压；

ADC_DATA 是 ADC_DR 里面的转换数据；

4096 表示为 12 位。

微控制的 V_{CC} 电源容易受影响或者不是很明确该值大小。内部电压参考（V_{REFINT}）以及在生产过程中 V_{CC} = 3.3 V ADC 获取的校准数据可以用来评估出真实的 V_{CC} 的电压水平。

下面公式可以给出真实 V_{DDA} 的电压值：

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

V_{REFINT} 固定值为 1.2 V；

$V_{CHANNEL}$ 是通道电压；

ADC_DATA 是 ADC_DR 里面的转换数据；

4096 表示为 12 位。

12.3.20. 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 12-8 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

12.4. ADC 寄存器

12.4.1. ADC 中断和状态寄存器 (ADC_ISR)

偏移地址： 0x00

复位值： 0x0000 0000

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

12.4.2. ADC 中断使能寄存器 (ADC_IER)

偏移地址： 0x04

复位值： 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res	Res	Res	Res	Res.	Res.	Res.
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	EOSE-QIE	EO-CIE	EOSMPIE	Res.
								rw			rw	rw	rw	rw	

Bit	Name	R/W	Reset Value	Function
31:8	保留	-	-	保留
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

Bit	Name	R/W	Reset Value	Function
				0：模拟看门狗中断不使能 1：模拟看门狗中断使能
6:5	保留	-	-	保留
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	保留	-	-	保留

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

12.4.3. ADC 控制寄存器 (ADC_CR)

偏移地址： 0x08

复位值： 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
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	AD-DIS	ADEN
											rs		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 注：软件写 1 不能写 0，硬件清零。
30:5	保留	-	-	保留

Bit	Name	R/W	Reset Value	Function
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会清除该位。 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。 软件写该位为 0 为无效操作。
3	保留	-	-	保留
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。 该位由硬件清除的情况： – 在单次转换模式（CONT=0，DISCEN=0)，选择软件驱动时(EXTEN=00)：序列转换完成时（EOSEQ 标志置位） – 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 标志置位） – 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。 注：软件只有当 ADEN=1 且 ADDIS=0 时才能配置 ADSTART=1. 软件写该位为 0 为无效操作。
1	ADDIS	RS	0	ADEN 禁止使能 软件置位禁止 ADC 并且 ADC 进入掉电。硬件清除该 bit，当 ADC 被禁止（ADEN 被硬件清零同时） 0：没有 ADDIS commond ongoing 1： 写 1 禁止 ADC，读 1 表示 ADDIS 指令正在执行 注：设置 ADDIS 为 1 有效只能在 ADEN=1 并且 ADSTART=0 时（确保没有转换进行）
0	ADEN	RS	0	ADC 使能命令。 软件置位该位使能 ADC，ADC 将准备操作。软件写该位为 0 为无效操作。 0: 不使能 ADC (OFF state) 1: 使能 ADC

12.4.4. ADC 配置寄存器 1 (ADC_CFGR1)

偏移地址： 0x0C

复位值： 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	AWDCH				Res	Res.	AWDEN	AWDSGL	Res.	Res	Re.	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	Res.	Res.	Res.	EXTSEL			ALIGN	RES_SEL		SCAN-DIR	Res.	Res.
	RW	RW	RW				RW	RW	RW	RW	RW	RW	RW		

Bit	Name	R/W	Reset Value	Function
31:30	保留	-	-	保留
29:26	AWDCH[3:0]	RW	0	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 …. 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	保留	-	-	保留
23	AWDEN	RW	0	模拟看门狗使能。 软件可设置和清除该位。 0：禁止模拟看门狗 1：使能模拟看门狗 注：仅当 ADSTART=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。 软件通过设置和清除该位，可以在 AWDCH[3:0]位设置的通道上或者所有通道上使能或者禁止模拟看门狗。 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（AWDCH[3:0]配置哪个通道） 注：仅当 ADSTART=0 时（确保没有正在进行的转换）允许软件写这些位
21:17	保留	-	-	保留
16	DISCEN	RW	0	非连续模式使能。 软件可设置和清除该位，使能/禁止非连续模式。 0：禁止非连续模式 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，即禁止设置 *DISCEN=1 和 CONT=1.* 注：仅当 ADSTART=0 时（确保没有正在进行的转换）允许软件写这些位。
15	保留	-	-	保留
14	WAIT	RW	0	等待转换模式。 软件可设置和清除该位，使能/禁止等待转换模式。 0：等待转换模式关闭 1：等待转换模式打开 注：仅当 ADSTART=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，直到该位被清除，否则在有触发发生时会一直发生转换。 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.* 注：仅当 ADSTART=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 注：仅当 ADSTART=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 注：仅当 ADSTART=0 时（确保没有正在进行的转换）允许软件写这些位。
9	保留	-	-	保留
8:6	EXTSEL[2:0]	RW	000	外部驱动选择(仅当 ADSART=0 时（确保没有正在进行的转换）允许软件写这些位) 该位选择触发转换启动的外部事件 000：TRG0(TIM1_TRGO) 001：TRG1(TIM1_CC4) 010：TRG2(TIM14_CC0) 011：TRG3(保留) 100：TRG4(保留) 101：TRG5(保留) 110：TRG6(保留) 111：TRG(保留)
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 注：仅当 ADSTART=0 时（确保没有正在进行的转换）允许软件写这些位
4:3	RESSEL[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 注：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 12） 1：向下（从通道 12 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	保留	-	-	保留

12.4.5. ADC 采样时间寄存器 (ADC_SMPR)

偏移地址： 0x14

复位值： 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.													SMP0		
													RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:3	保留	-	-	保留
2:0	SMP[2:0]	RW	0	采样时钟选择。 软件可配置该位选择通道 x 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 110：134.5 ADC 时钟周期 111：239.5 ADC 时钟周期 仅当 ADSTART=0 时（确保没有正在进行的转换）允许软件写这些位。

12.4.6. ADC 看门狗阈值寄存器 (ADC_TR)

偏移地址： 0x20

复位值： 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	保留	-	-	保留

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 ADSTART=0 时（确保没有正在进行的转换）允许软件写这些位。
15:12	保留	-	-	保留
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 ADSTART=0 时（确保没有正在进行的转换）允许软件写这些位。

12.4.7. ADC 通道选择寄存器 (ADC_CHSELR)

偏移地址： 0x28

复位值： 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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	CHSEL 10	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
31:13	保留	-	-	保留
12	CHSEL12	RW	0	通道 12（V_{CC}/3）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（V_{REFINT}）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	CHSEL10	RW	0	通道 10（T_S）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
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

12.4.8. ADC data register (ADC_DR)

偏移地址： 0x40

复位值： 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
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	保留	-	-	保留
15:0	DATA[15:0]	R	0	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。

12.4.9. ADC 校准配置和状态寄存器(ADC_CCSR)

偏移地址： 0x44

复位值： 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CALON.	CAP-SUC	OFFSUC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
R	RC_W1	RC_W1													
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CALSET	CAL-BYP	CALSMP		CALSEL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
RW	RW	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	CAPSUC	RC_W1	0	电容校准状态位。 表示 ADC 电容校准是否成功。硬件置 1；软件写 1 置 0； CALON=0, CALSEL=0,CAPSUC=1： 无效状态 CALON=0, CALSEL=0, CAPSUC=0： 未进行 CAPs 校准 CALON=0, CALSEL=1, CAPSUC =1： ADC CAPs 校准成功 CALON=0, CALSEL=1, CAPSUC =0： ADC CAPs 校准失败 注：不管 C11~C6 校准是否成功，电容校准仅校准一次。
29	OFFSUC	RC_W1	1’b0	Offset 校准状态位。 表示 ADC offset 校准是否成功。硬件置 1；软件写 1 置 0； CALON=0, CALSEL=0,OFFSUC=0： ADC OFFSET 校准失败 CALON=0, CALSEL=0, OFFSUC=1：ADC OFFSET 校准成功 CALON=0, CALSEL=1,OFFSUC=1： ADC OFFSET 校准成功 CALON=0, CALSEL=1, OFFSUC=0：ADC OFFSET 校准失败

Bit	Name	R/W	Reset Value	Function
28:16	保留	-	-	保留
15	CALSET	RS	0	Calibration factor selection 软件置位（在 ADCAL=0 之前，即校准之前），硬件清零。 1：设置 CAL_CXIN 数据作为最终的校准数据 0：关闭 CAL_CXIN 到 CAL_CXOUT 的通路，选择校准电路内部产生的结果。
14	CALBYP	RS	0	Calibration factor bypass 软件置位（在 ADCAL=0 之前，即校准之前），硬件清零。 1：屏蔽自动校准以及设置 CALSET 校准的输出结果到 CAL_CXOUT 0：选择自动校准或者设置 CALSET 校准的输出结果到 CAL_CXOUT
13:12	CALSMP	RW	0	Calibration sample time seletion 根据以下信息，配置 calibration 的采样阶段的时钟周期个数: 00：1 个 ADC 时钟周期 01：2 个 ADC 时钟周期 10：4 个 ADC 时钟周期 11：8 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	校准内容选择位，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	保留	-	-	保留

12.4.10. ADC 通用配置寄存器(ADC_CCR)

偏移地址： 0x308

复位值： 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31：24	保留	-	-	-
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 V_{refint} 使能位，软件可设置和清除该位，使能/不使能基准 V_{refint} 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
21:8	保留	-	-	-
7:6	Vrefbuff_sel	RW	2’b0	V_{REFBUF} output voltage select 00: 0.6 V 01:1.5 V 10: 2.048 V 11: 2.5 V
5	Vref_buffere	RW	1’b0	V_{REFBUF} enable 软件写 0 置 0，写 1 置 1， 0：disable V_{REFBUF} 1：enable V_{REFBUF}
4：0	保留	-	-	-

13. 比较器（COMP）

13.1. 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

13.2. 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 V_{CC} 和通过分压提供的 64 个分数值(1/64、2/64 … 64/64)
  - 内部参考电压 0.6 V、1.5 V、2.048 V 或 2.5 V，和通过分压提供的 64 个分数值(1/64、2/64 … 64/64)
- 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）

13.3. COMP 功能描述

13.3.1. COMP 框图

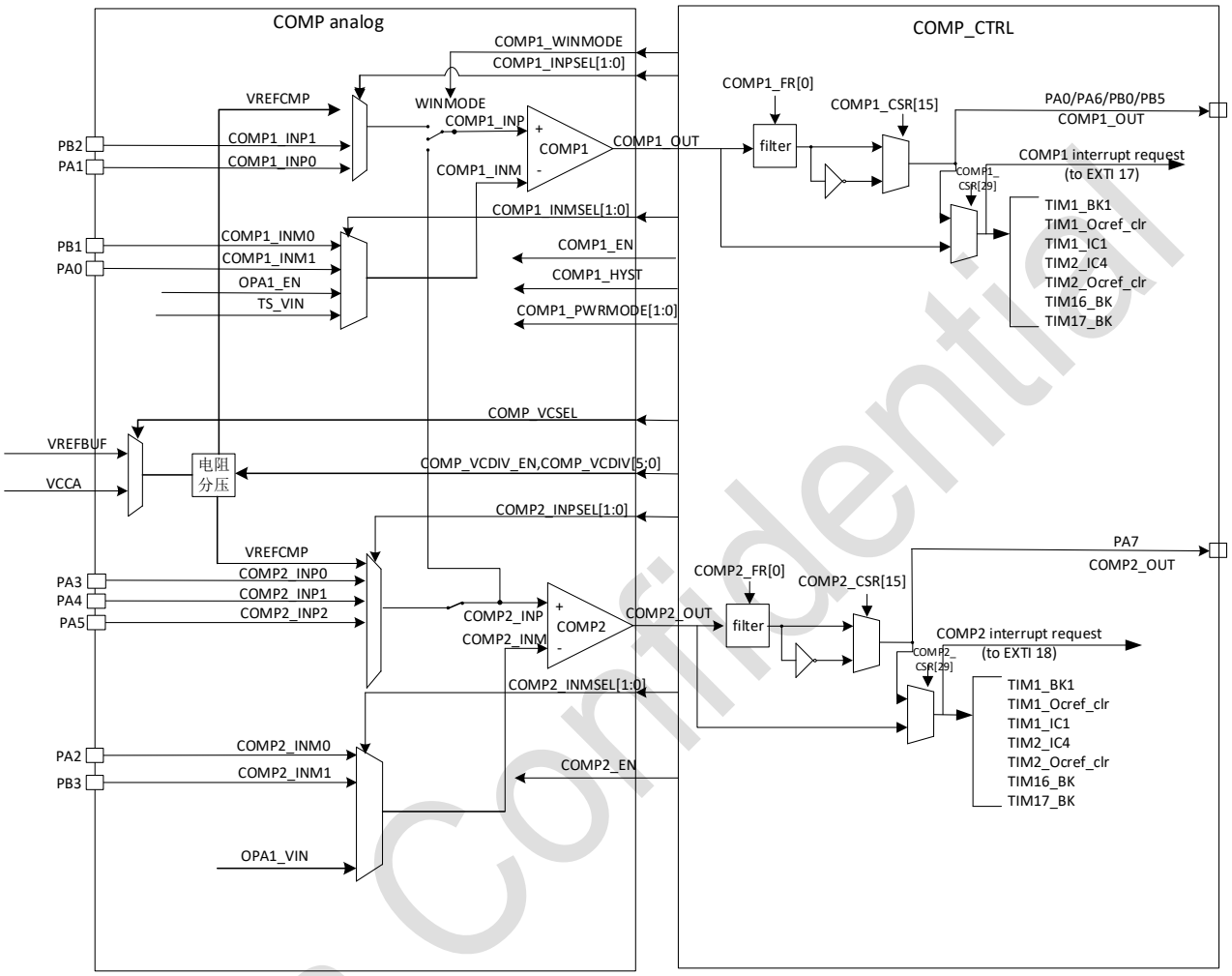


图 13-1 比较器架构框图

13.3.2. 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 的可选功能（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

可以在外部或内部重定向比较器输出。下表显示了比较器输出至其他 IP 的关系。

表 13-1 COMP 模块数字输出

COMP1 out	COMP 2 out
TIM1_BRK 刹车使能	TIM1_BRK 刹车使能

COMP1 out	COMP 2 out
TIM1_OREF_CLR 使能	TIM1_Ocref_clr 使能
TIM1_IC1 通道使能	TIM1_IC1 通道使能
TIM1_ETR 使能	TIM1_ETR 通道使能
作为 EXTI17 输入事件唤醒	作为 EXTI18 输入事件唤醒
输出到 PB3	输出到 PA2

13.3.3. COMP 复位和时钟

COMP 模块有两个时钟源：

- PCLK（APB clock），系统总线时钟
- 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、LSE 或者 LSI。当需要在 stop 模式下工作时，选择 LSE 或者 LSI。

注意：

COMP 模块的复位信号包含 APB 复位源和 COMP 模块软件复位源：

1)APB 复位，用于 COMP 寄存器的复位

2)COMP 软件复位，用于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

当 RCC_APBRSTR2 中的复位信号使能后，COMP 模块 PRESETn 和 COMP_RSETn 信号都会被复位。

13.3.4. 比较器锁装置

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，需要确保在假的寄存器访问和 PC（program counter）混乱时，比较器的编程不能被改写。

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。

13.3.5. 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。

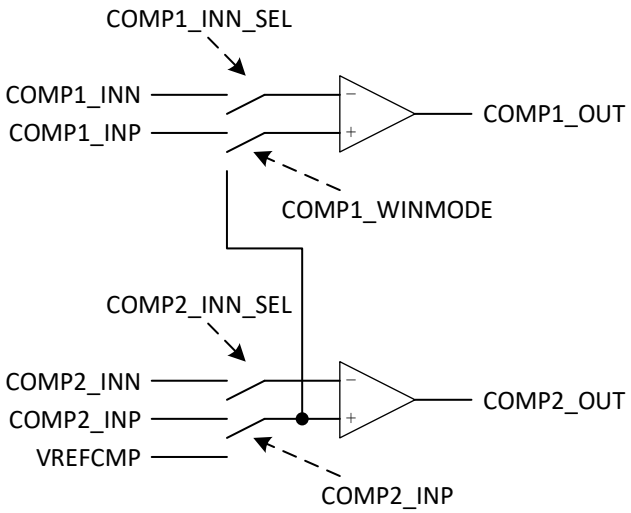


图 13-2 window comparator

13.3.6. 低功耗模式

- **Sleep Mode**：对 COMP 无影响，比较器中断可以使设备退出 Sleep 模式。
- **Stop Mode**：对 COMP 无影响，比较器中断可以使设备退出 Sleep 模式 Stop 模式。

13.3.7. 比较器滤波

如果芯片的工作环境恶劣，迟滞比较器的输出会出现噪声信号。使能数字滤波模块，则迟滞比较器的输出波形中脉宽小于 FRx.FLTCNT[15:0]设定时间的噪声信号都可以被滤除。禁止数字滤波模块，则数字滤波模块的输入输出信号相同。

注意：设置 COMP 滤波时间，开启滤波使能应在 COMP_EN 使能前完成。

滤波示意图如下：

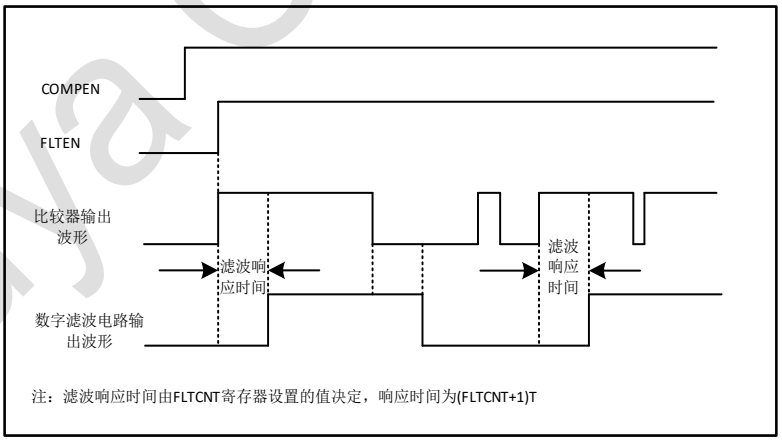


图 13-3 比较器滤波

13.3.8. 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，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。详细信息参见“NVIC”模块。

13.4. COMP 寄存器

13.4.1. COMP1 控制和状态寄存器(COMP1_CSR)

偏移地址：0x00

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	COMP_OUT	TIM_EXTI_OUT_SEL	Res.	COMP_VCSEL	COMP VCDIV_EN	COMP_VCDIV[5:0]						Res.	PWR-MODE	Res.	Res.
	R	RW		RW	RW	RW							RW		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PO-LARITY	Res.	Res.	Res.	WINMODE	Res.	INPSEL[1; 0]		INMSEL[1:0]		Res.	Res.	Res.	Res.	Res.	COMP1 EN
RW		-	-	RW	-	RW		RW							RW

Bit	Name	R/W	Reset Value	Function
31	保留	-	-	保留
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	TIM_EXTI_OUT_SEL	RW	0	COMP1 输出到 TIM/EXTI 选择。 0：不经过滤波，ana_out 直接输出到 TIM/EXTI 1：选择输出到 PAD 上的 comp_out 输出到 TIM/EXTI
28	保留	-	-	-
27	COMP_VCSEL	RW	0	V_{REFCMP} reference source selection. VR_{EFCMP} is enabled by VREFINT_EN. 0: V_{REFBUF} 1: V_{CC}, V_{REFINT} and V_{REFBUF} is not available at COMP_VCSEL=1.
26	COMP_VCDIV_EN	RW	0	V_{REFCMP} voltage divider enable signal High means enable V_{REFCMP} voltage divider
25:20	COMP_VCDIV[5:0]	RW	0	COMP1,COMP2 分压选择 000000: 1/64 Vref 000001: 2/64 Vref 000010: 3/64 Vref … 111110: 63/64 Vref 111111: Vref
19	保留	-	-	保留
18	PWRMODE	RW	0	COMP1 功耗模式选择 选择了功耗和由此而来的 COMP1 的速度 0：High speed 1：Medium speed
17:16	保留	-	-	保留
15	POLARITY	RW	0	COMP1 极性选择 软件可读可写 0：不反向 1：反向
14:12	保留	-	-	保留
11	WINMODE	RW	0	COMP1 window mode enable

Bit	Name	R/W	Reset Value	Function
				0: without window mode, COMP1 positive input is COMP1_INP 1: windows mode, COMP1 positive input is the same as COMP2 positive input
10	保留			
9:8	INPSEL[1:0]	RW	00	COMP1 non-inverting（+端）输入的信号选择 00: COMP1_INP 来自 PA1 01: COMP1_INP 来自 PB2 10: COMP1_INP 来自 V_{REFCMP} 11: 保留
7:6	INMSEL[1:0]	RW	00	COMP1 inverting（-端）输入选择 00: COMP1_INM 来自 PA0 01: COMP1_INM 来自 PB1 10: COMP1_INM 来自 TS_VIN 11: 保留
5:1	保留	-	-	保留
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

13.4.2. COMP1 滤波寄存器(COMP1_FR)

偏移地址：0x04

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FLTCNT1[15:0]															
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 或 LSE。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	保留	-	-	保留
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 注: 该位必须在 COMP1_EN 为 0 时置位

13.4.3. COMP2 控制和状态寄存器(COMP2_CSR)

偏移地址：0x10

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	COMP_OUT	TIM_EXTI_OUT_SEL	Res.	Res.	Res.	Res.				Res.	Res.	Res.	PWR-MODE	Res.	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.	Res.	Res.	INPSEL[1; 0]		INMSEL[1:0]		Res.	Res.	Res.	Res.	Res.	COMP2_EN
RW		-	-		-	RW		RW							RW

Bit	Name	R/W	Reset Value	Function
31	保留	-	-	保留
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	TIM_EXTI_OUT_SEL	RW	0	COMP2 输出到 TIM/EXTI 选择。 0：不经过滤波，ana_out 直接输出到 TIM/EXTI 1：选择输出到 PAD 上的 comp_out 输出到 TIM/EXTI
28:19	保留	-	-	保留
18	PWRMODE	RW	0	COMP2 功耗模式选择 选择了功耗模式和由此而来的 COMP2 的速度 0：High speed 1：Medium speed
17:16	保留	-	-	保留
15	POLARITY	RW	0	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
14:10	保留	-	-	保留
9:8	INPSEL[1:0]	RW	0	COMP2 non-inverting（+端）输入的信号选择 00: COMP2_INP 来自 PA2 01: COMP2_INP 来自 PB3 10: 保留 11: 保留
7:6	INMSEL[1:0]	RW	00	COMP2 inverting（-端）输入选择 00: COMP2_INM 来自 PA2 01: COMP2_INM 来自 PB3 10: 保留 11: 保留
5:1	保留	-	-	保留
0	COMP2_EN	RW	0	COMP2 使能位 软件可读可写（如果没有被锁定） 0：Disable 1：Enable

13.4.4. COMP2 滤波寄存器(COMP2_FR)

偏移地址：0x14

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
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	RW	0	比较器 2 采样滤波计数器 采样时钟为 APB 或 LSI 或 LSE。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	保留	-	-	保留
0	FLTEN2	RW	0	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 注: 该位必须在 COMP2_EN 为 0 时置位

14. 高级控制定时器(TIM1)

14.1. 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。

14.2. 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 65536 的分频（可以实时修改）
- 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，才更新定时器的寄存器
- 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

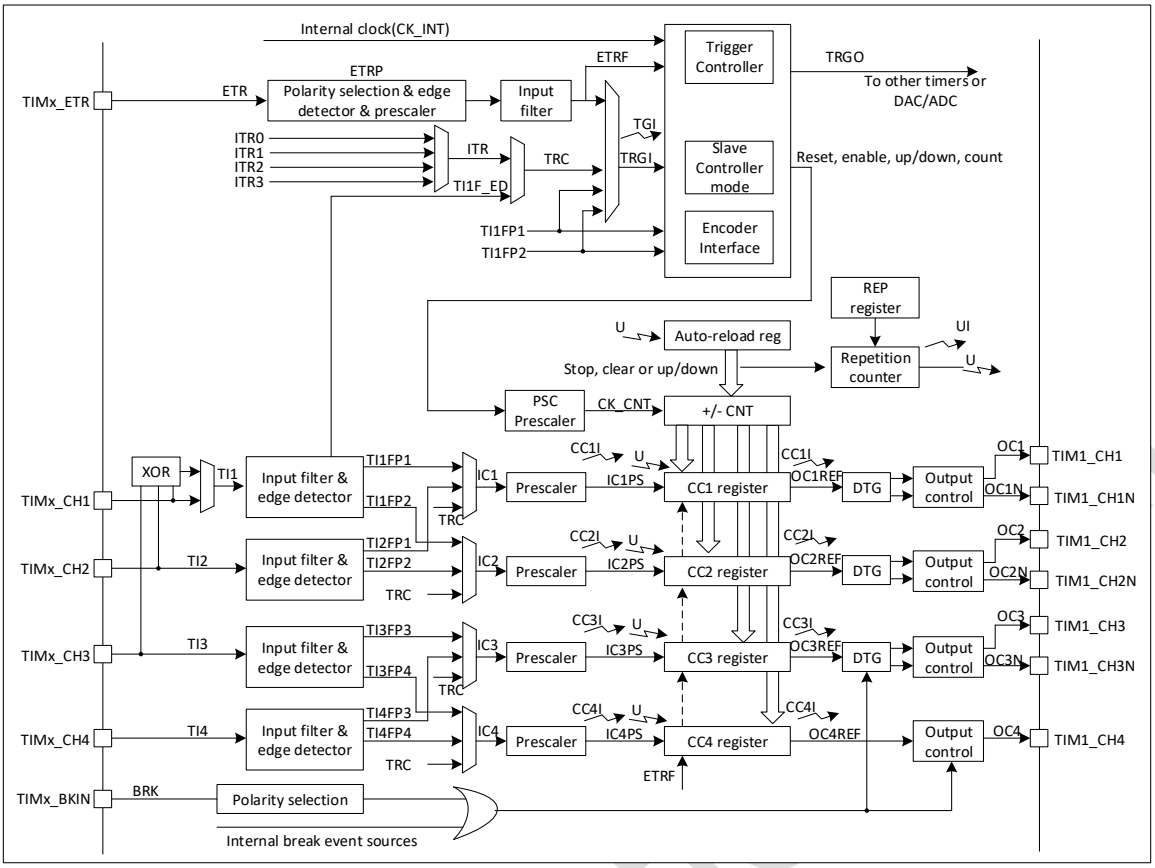


图 14-1 高级控制定时器架构框图

14.3. TIM1 功能描述

14.3.1. 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 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。

预分频器描述

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

下图给出了在预分频器运行时，更改计数器参数的例子。

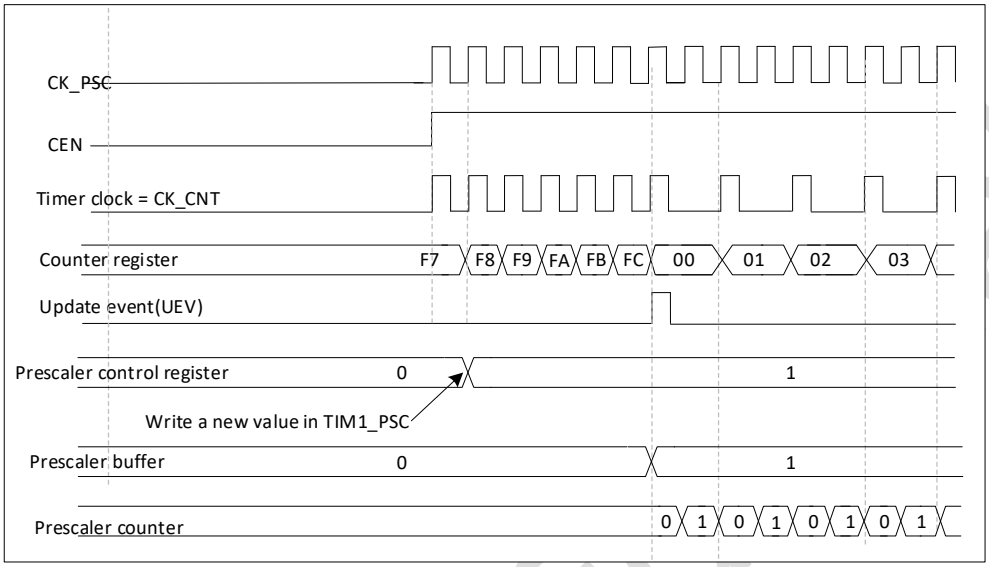


图 14-2 当预分频器的参数从 1 变到 2 时，计数器的时序图

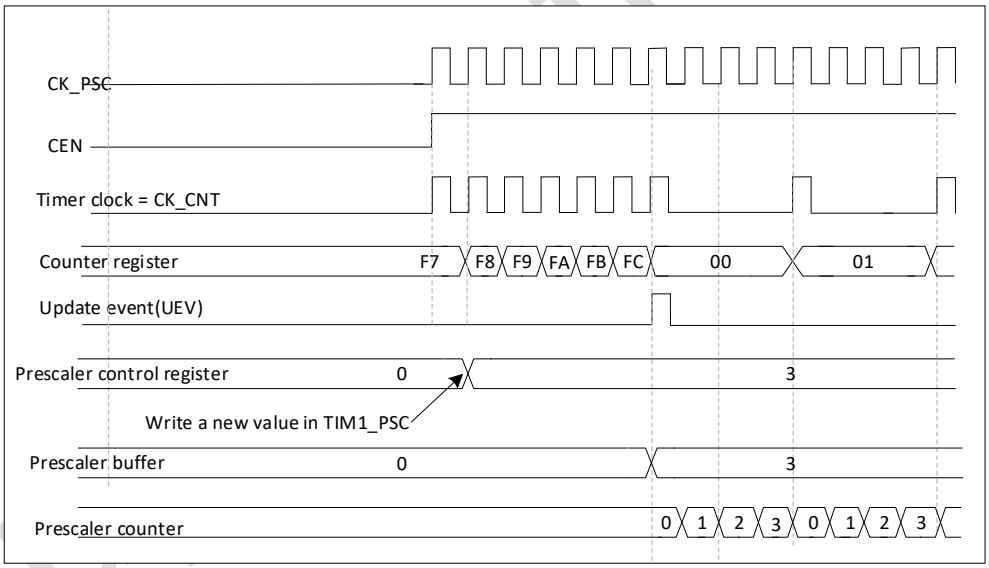


图 14-3 当预分频器的参数从 1 变到 4 时，计数器的时序图

14.3.2. 计数器模式

14.3.2.1. 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)。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。

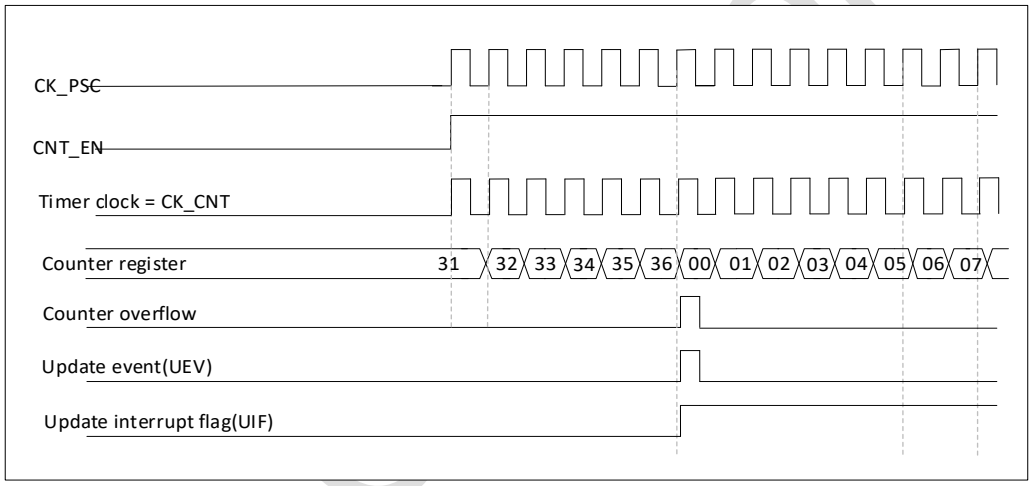


图 14-4 计数器时序图，内部时钟分频因子为 1

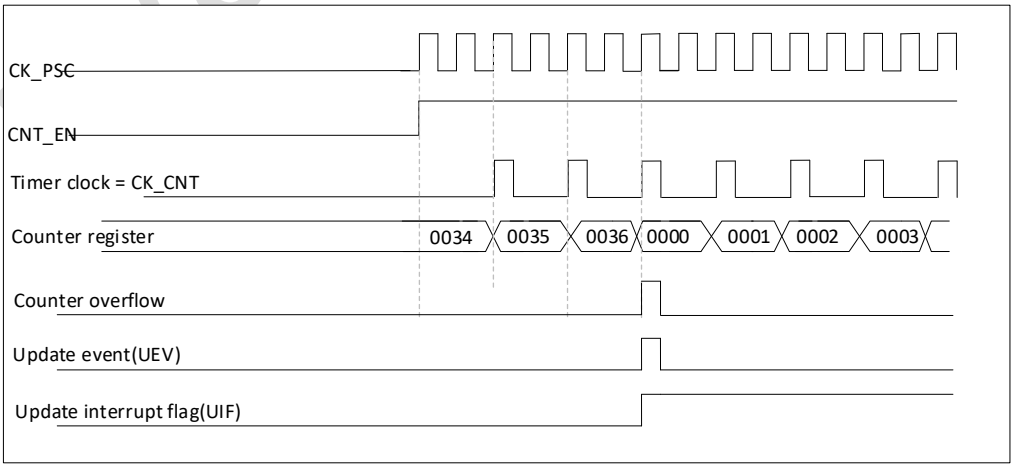


图 14-5 计数器时序图，内部时钟分频因子为 2

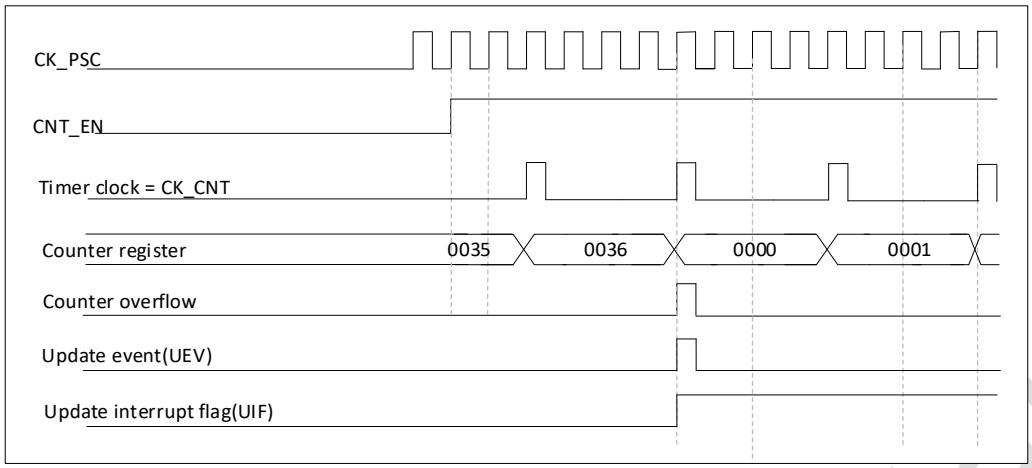


图 14-6 计数器时序图，内部时钟分频因子为 4

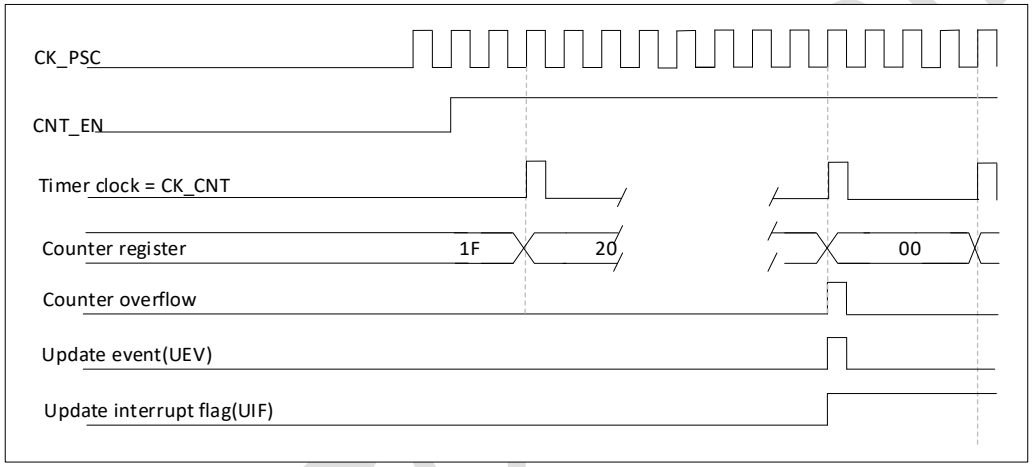


图 14-7 计数器时序图，内部时钟分频因子为 N

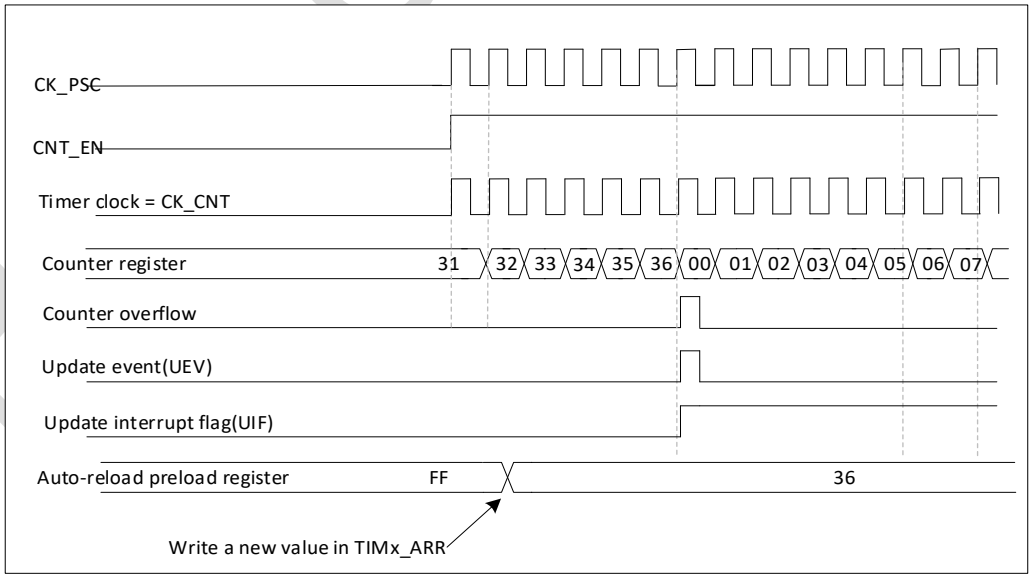


图 14-8 计数器时序图，当 ARPE=0 时的更新事件(TIM1_ARR 没有预装入)

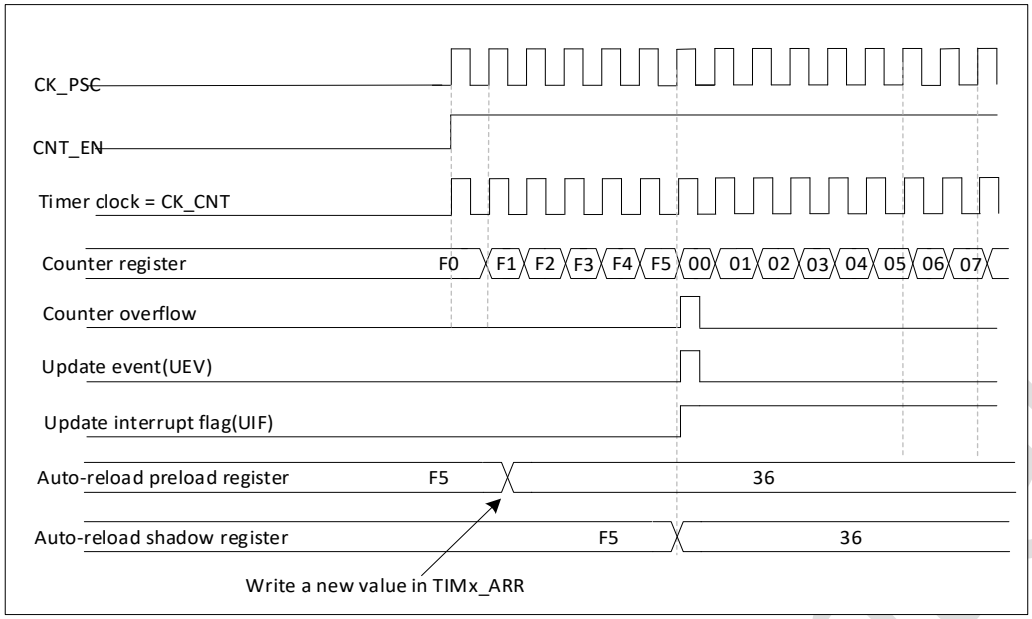


图 14-9 计数器时序图，当 ARPE=1 时的更新事件(预装入了 TIM1_ARR)

14.3.2.2. 向下计数模式

向下计数模式，计数器从自动装载的值(TIMx_ARR 的内容)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，那向下计数模式，计数器从自动装载的值(TIMx_ARR 的内容)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，那么更新事件(UEV)需要在下溢次数达到所配置的重复计数寄存器的值加一（即 TIMx_RCR+1）时才会产生；如果没有使用重复计数器（即 TIMx_RCR=0），那么每次计数下溢都会产生更新事件。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。

通过设置 TIMx_CR1 寄存器中的 UDIS 位，可以禁止更新事件；这样可以避免在向预装载寄存器中写入新值时更新影子寄存器。在 UDIS 位被清‘0’之前不会产生更新事件。即使这样，在应该产生更新事件时，计数器仍会从当前自动加载值重新开始计数，同时预分频器内部的计数器被清‘0’(但预分频系数不变)。此外，如果设置了 TIMx_CR1 寄存器中的 URS 位(选择更新请求)，设置 UG 位将产生一个更新事件 UEV 但不设置 UIF 标志(因此不产生中断)，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。么更新事件(UEV)需要在下溢次数达到所配置的重复计数寄存器的值加一（即 TIMx_RCR+1）时才会产生；如果没有使用重复计数器（即 TIMx_RCR=0），那么每次计数下溢都会产生更新事件。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。

通过设置 TIMx_CR1 寄存器中的 UDIS 位，可以禁止更新事件；这样可以避免在向预装载寄存器中写入新值时更新影子寄存器。在 UDIS 位被清‘0’之前不会产生更新事件。即使这样，在应该产生更新事件时，计数器仍会从当前自动加载值重新开始计数，同时预分频器内部的计数器被清‘0’(但预分频系数不变)。此外，如果设置了 TIMx_CR1 寄存器中的 URS 位(选择更新请求)，设置 UG 位将产生一个

更新事件UEV但不设置UIF标志(因此不产生中断)，这是为了避免在发生捕获事件并清除计数器时，同时产生更新和捕获中断。

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

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

注：自动装载在计数器重载入之前被更新，因此下一个周期将是预期的值。

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

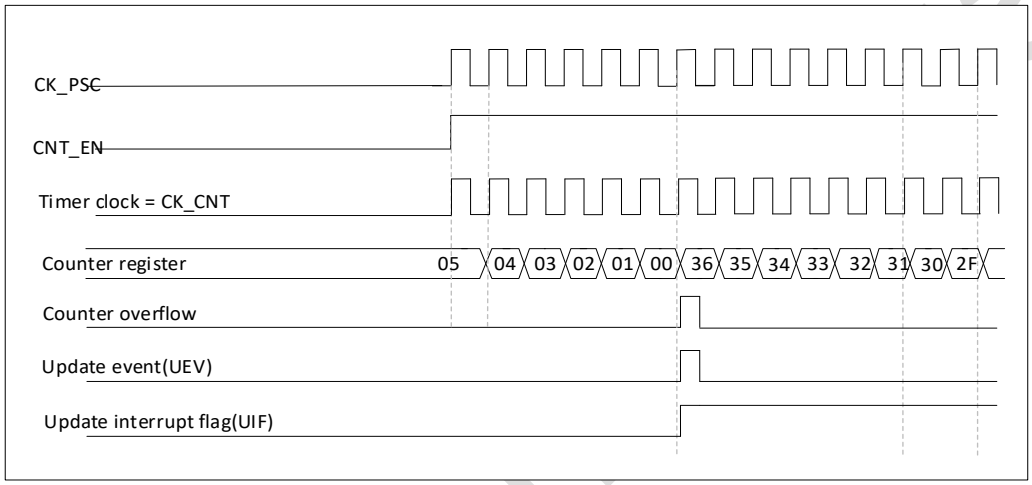


图 14-10 计数器时序图，内部时钟分频因子为 1

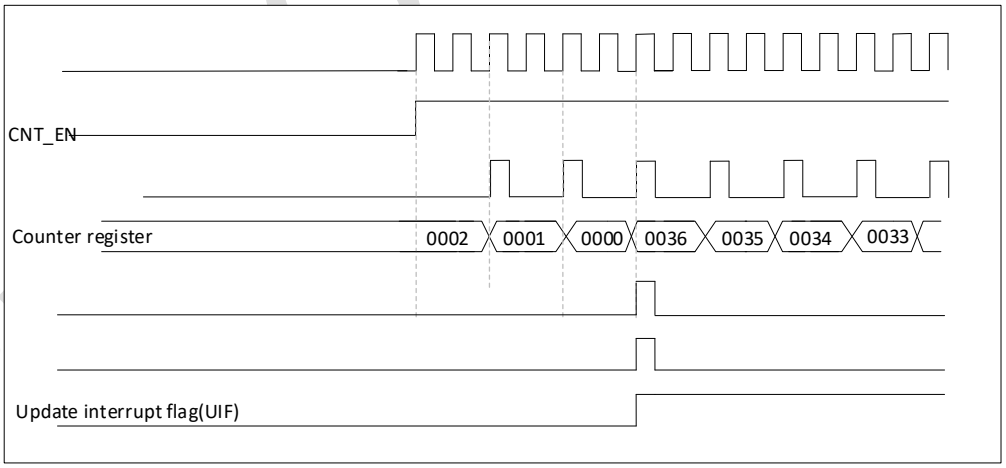


图 14-11 计数器时序图，内部时钟分频因子为 2

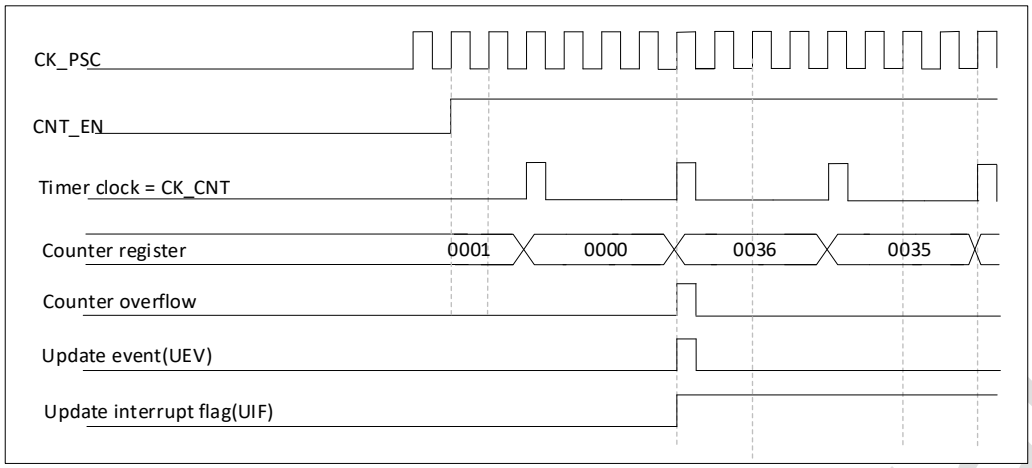


图 14-12 计数器时序图，内部时钟分频因子为 4

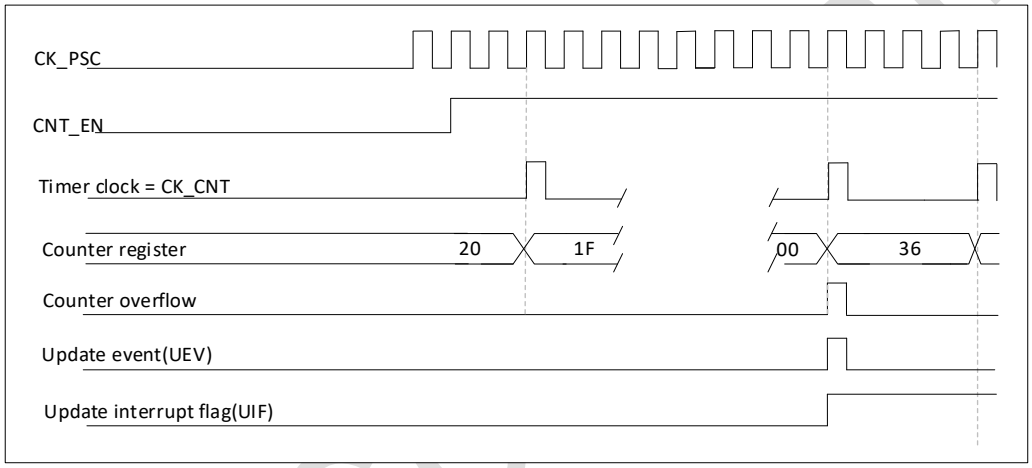


图 14-13 计数器时序图，内部时钟分频因子为 N

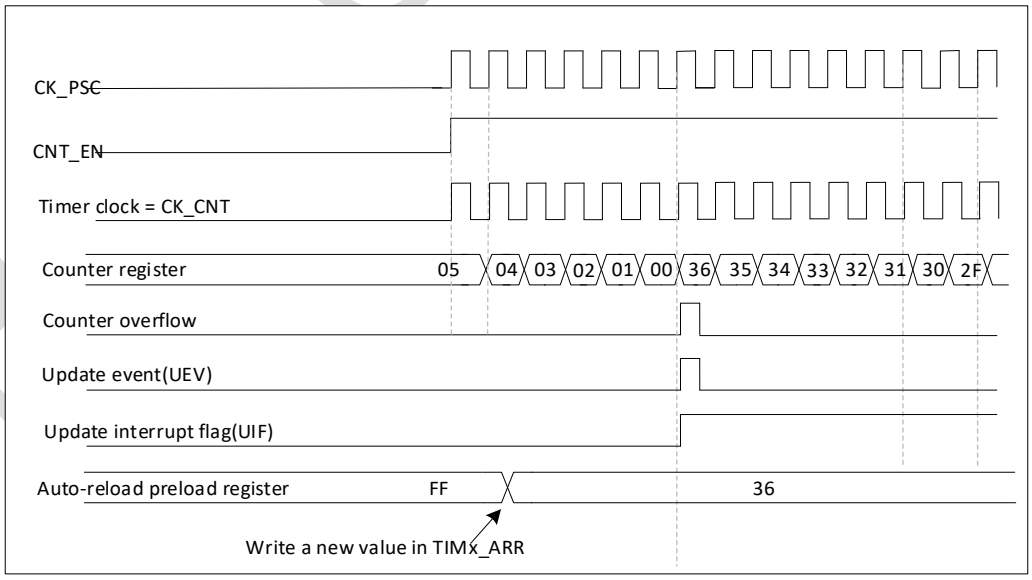


图 14-14 计数器时序图，当没有使用周期计数器时的更新事件

14.3.2.3. 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)，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)

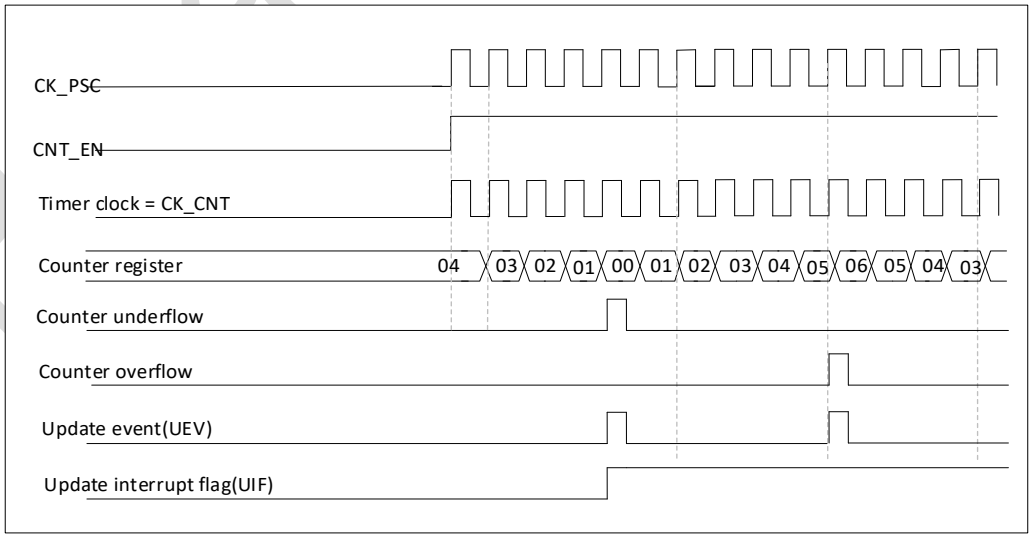


图 14-15 计数器时序图，内部时钟分频因子为 1，TIMx_ARR = 0x6

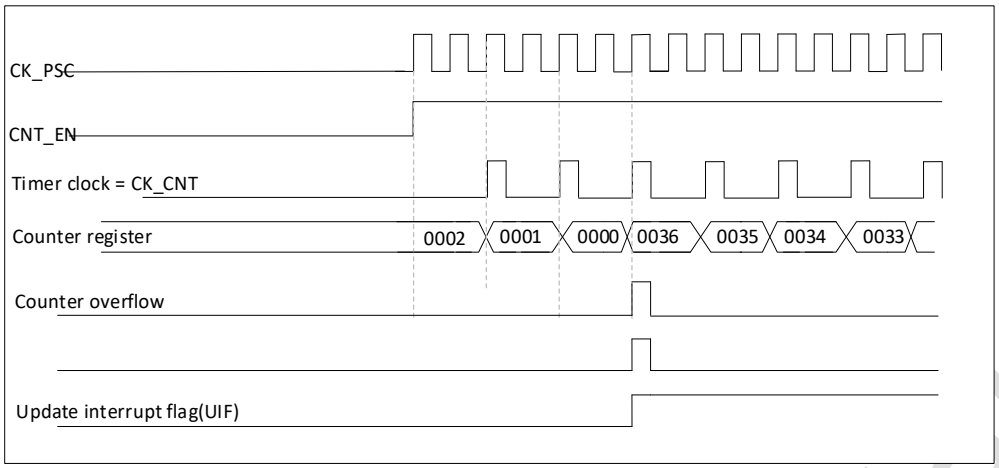


图 14-16 计数器时序图，内部时钟分频因子为 2, TIMx_ARR=0x36

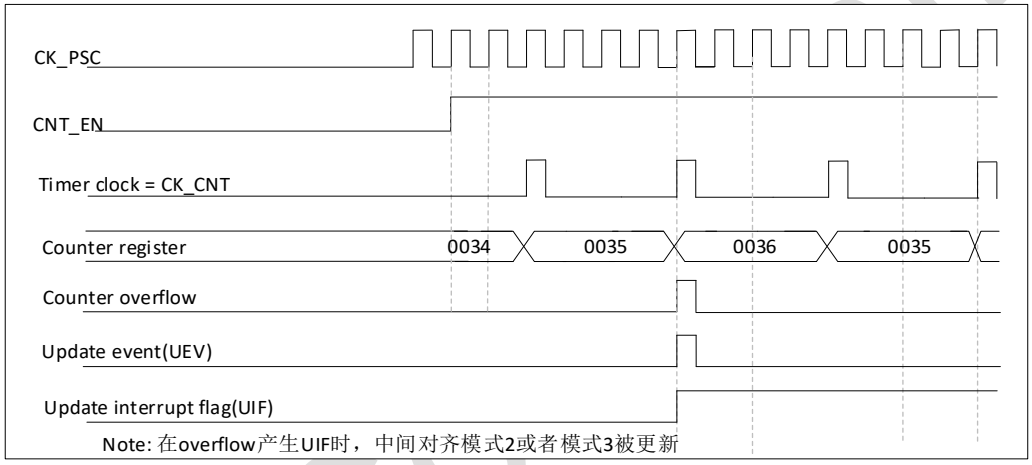


图 14-17 计数器时序图，内部时钟分频因子为 4, TIMx_ARR=0x36

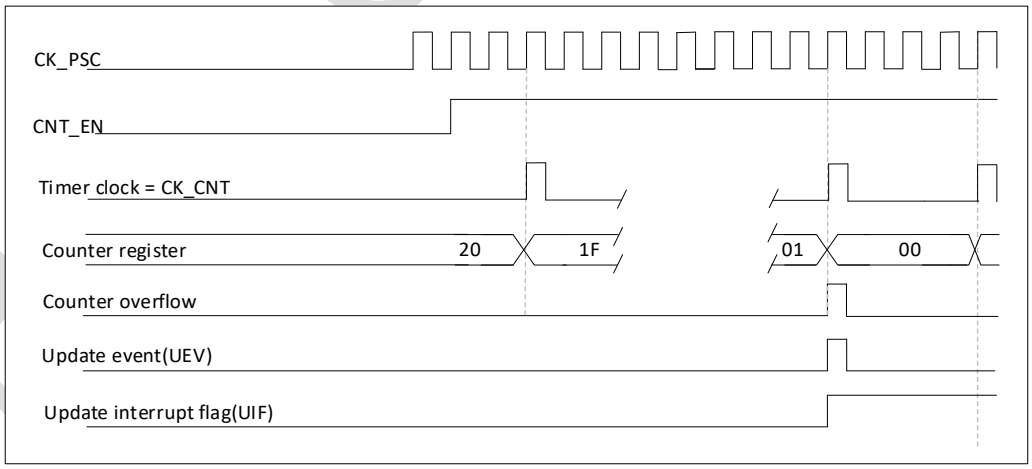


图 14-18 计数器时序图，内部时钟分频因子为 N, TIMx_ARR=0x36

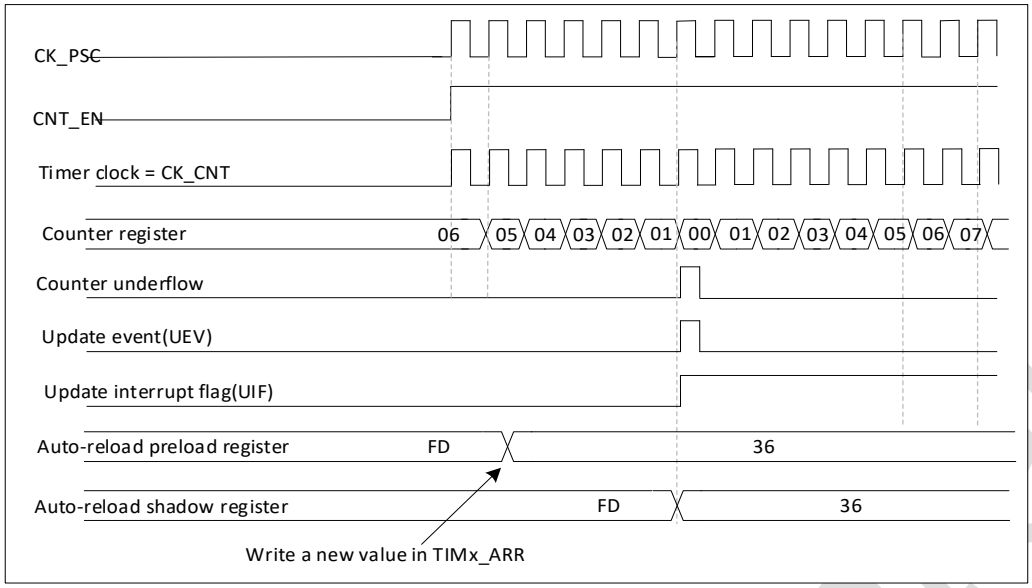


图 14-19 计数器时序图，ARPE=1 时的更新事件(计数器下溢)

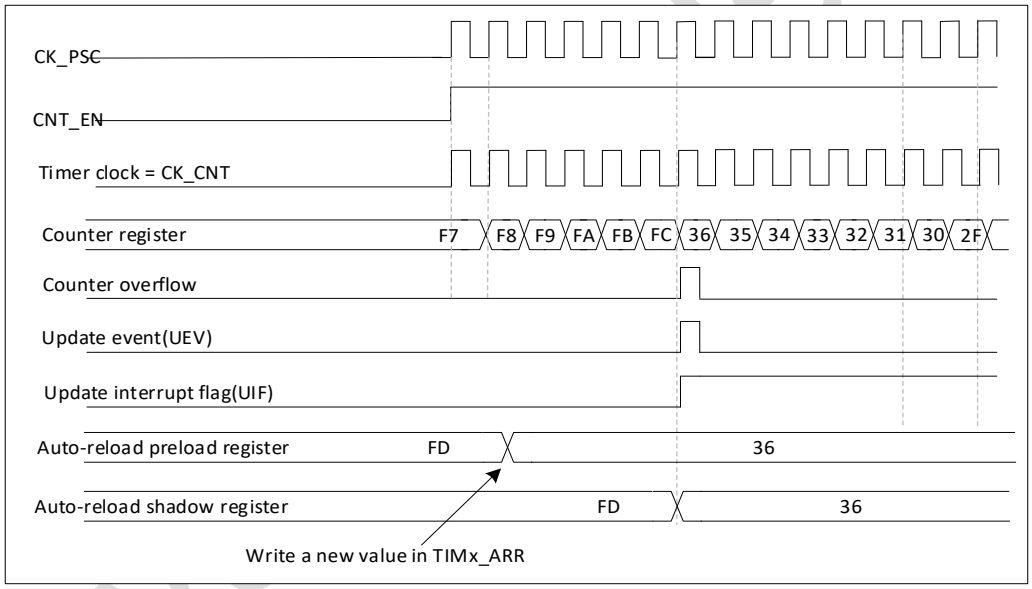


图 14-20 计数器时序图，ARPE=1 时的更新事件(计数器溢出)

14.3.3. 重复计数器

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 PWM 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到重复计数器。

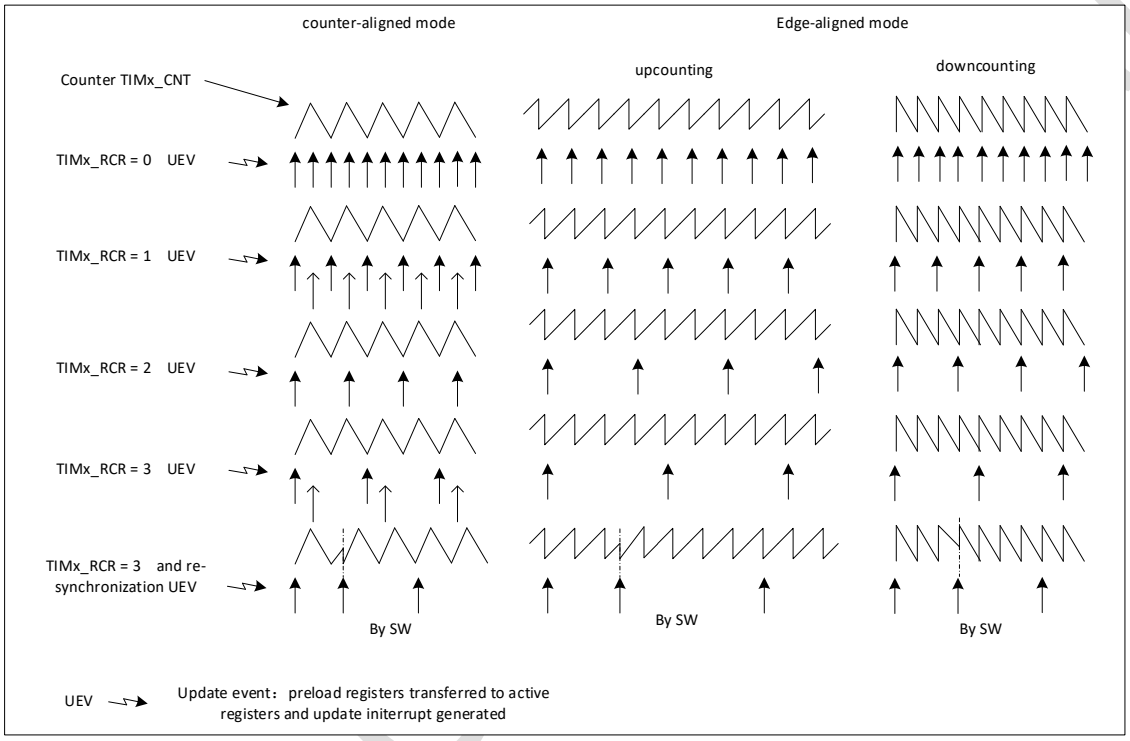


图 14-21 不同模式下更新速率的例子，及 TIMx_RCR 的寄存器设置

14.3.4. 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 Timer3 的预分频器。

14.3.4.1. 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。

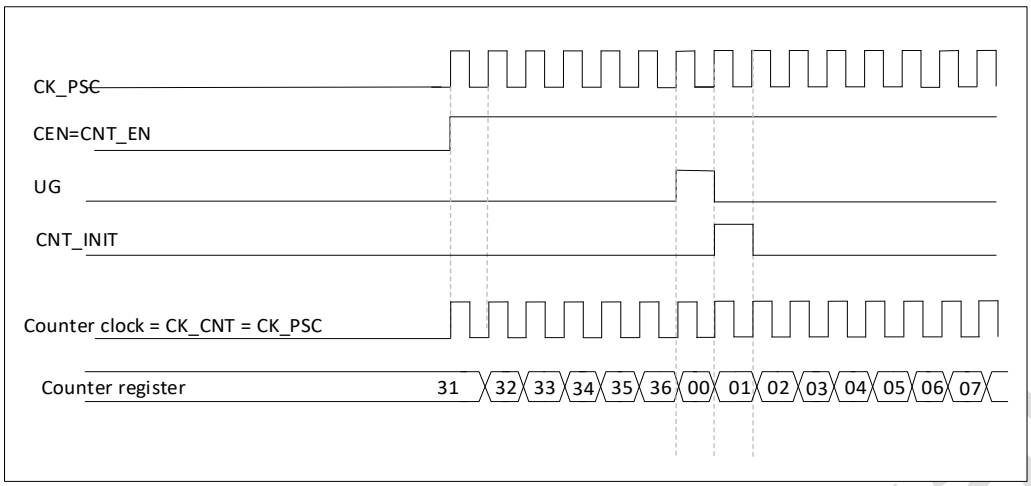


图 14-22 一般模式下的控制电路，内部时钟分频因子为 1

14.3.4.2. 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。

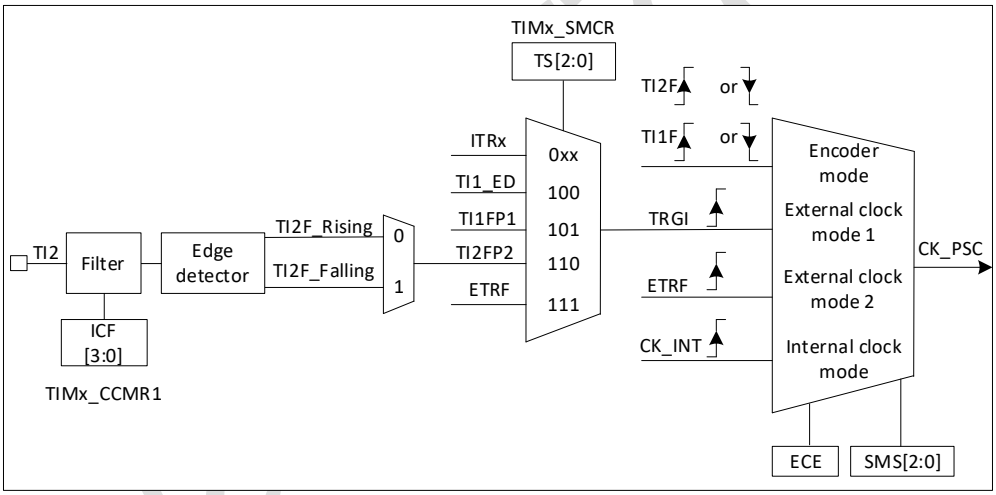


图 14-23 TI2 外部时钟连接例子

例如，要配置计数器在 T12 输入端的上升沿向上计数，使用下列步骤：

1.配置 TIMx_CCMR1 寄存器 CC2S=01，使得通道 2 检测 TI2 输入端的上升沿；

2.配置 TIMx_CCMR1 寄存器的 IC2F[3:0]，选择输入滤波器带宽（如果不需要滤波器， 保持 IC2F=0000)；

3.配置 TIMx_CCER 寄存器的 CC2P=0，选定上升沿极性；

4.配置 TIMx_SMCR 寄存器的 SMS=111，选择定时器为外部时钟模式 1；

5.配置 TIMx_SMCR 寄存器中的 TS=110，选定 TI2 作为触发输入源；

6.设置 TIMx_CR1 寄存器的 CEN=1，启动计数器。

注： 捕获预分频器不用作触发，所以不需要对它进行配置

当上升沿出现在 TI2，计数器计数一次，且 TIF 标志被设置。

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

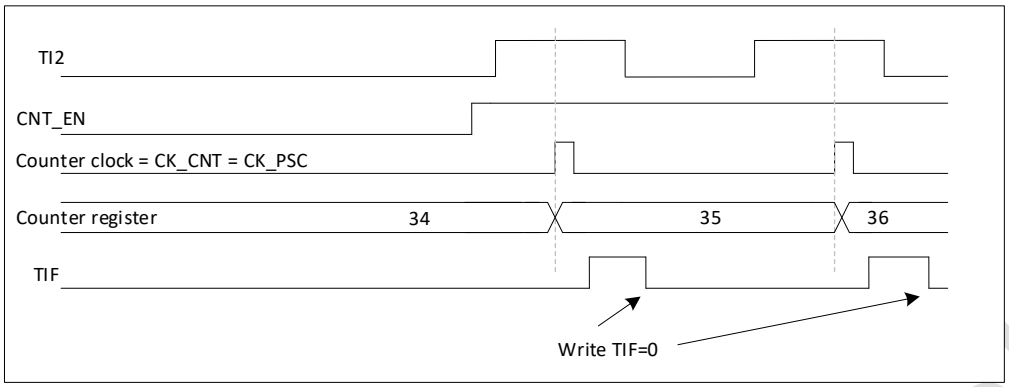


图 14-24 外部时钟模式 1 下的控制电路

14.3.4.3. 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。

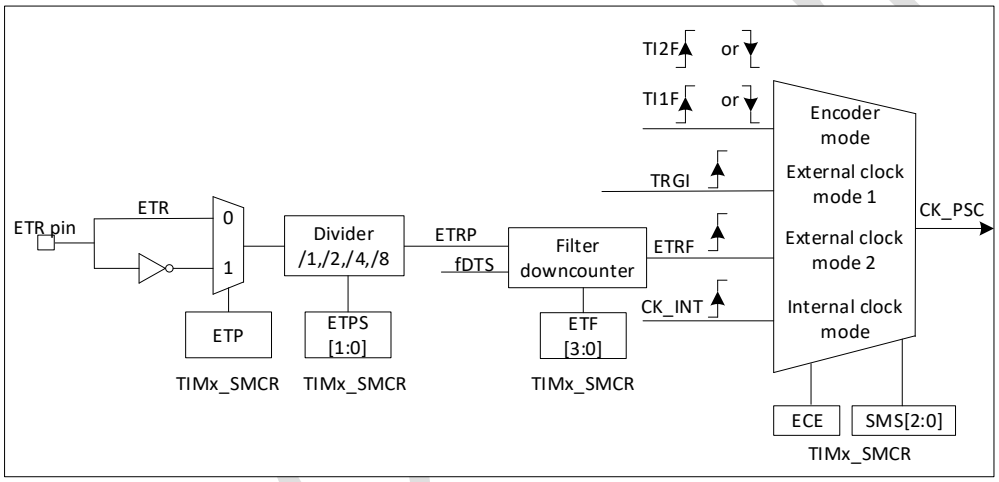


图 14-25 TI2 外部触发输入框图

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

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

2.设置预分频器，置 TIMx_SMCR 寄存器中的 ETPS[1:0]=01；

3.选择 ETR 输入端的上升沿，置 TIMx_SMCR 寄存器中的 ETP=0；

4.开启外部时钟模式 2，写 TIMx_SMCR 寄存器中的 ECE=1；

5.启动计数器，写 TIMx_CR1 寄存器中的 CEN=1；

计数器在每 2 个 ETR 上升沿计数一次。

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

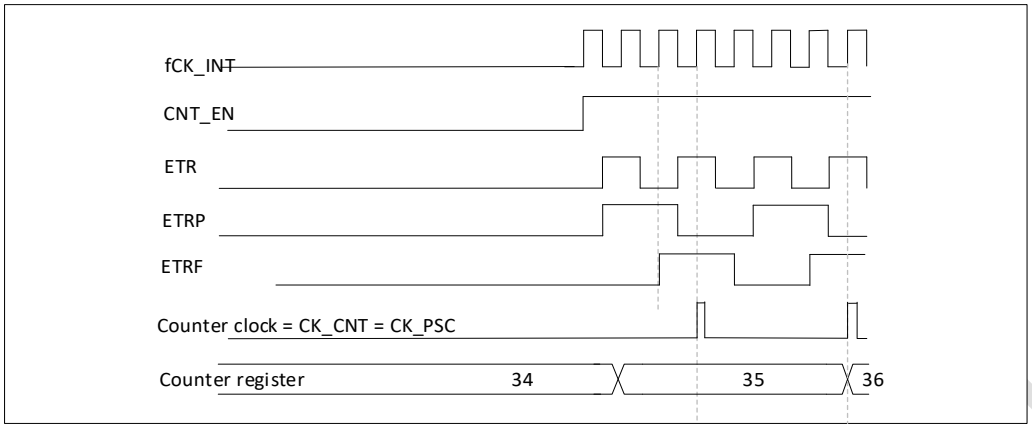


图 14-26 外部时钟模式 2 下的控制电路

14.3.5. 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）。

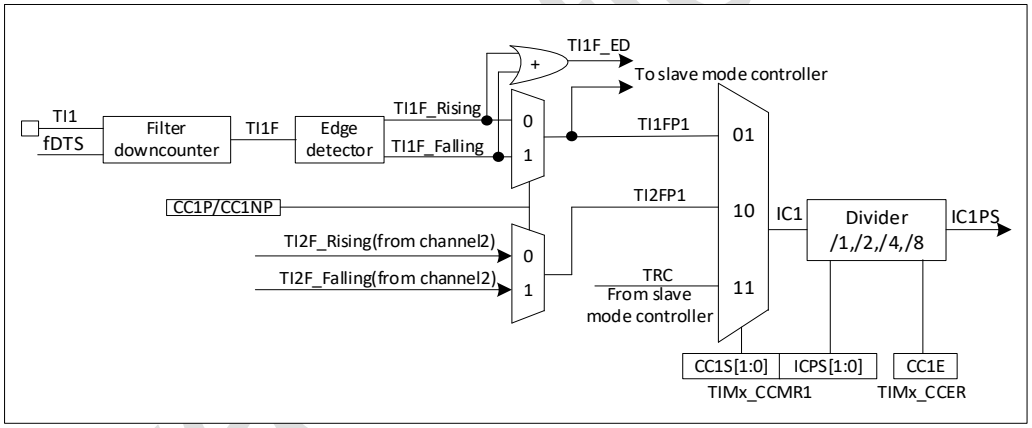


图 14-27 捕获/比较通道(如：通道 1 输入部分)

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。

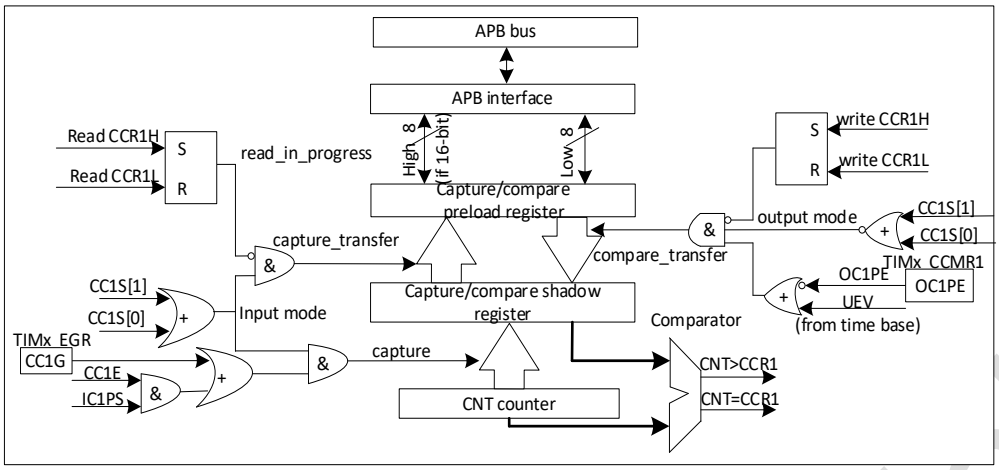


图 14-28 捕获/比较通道 1 的主电路

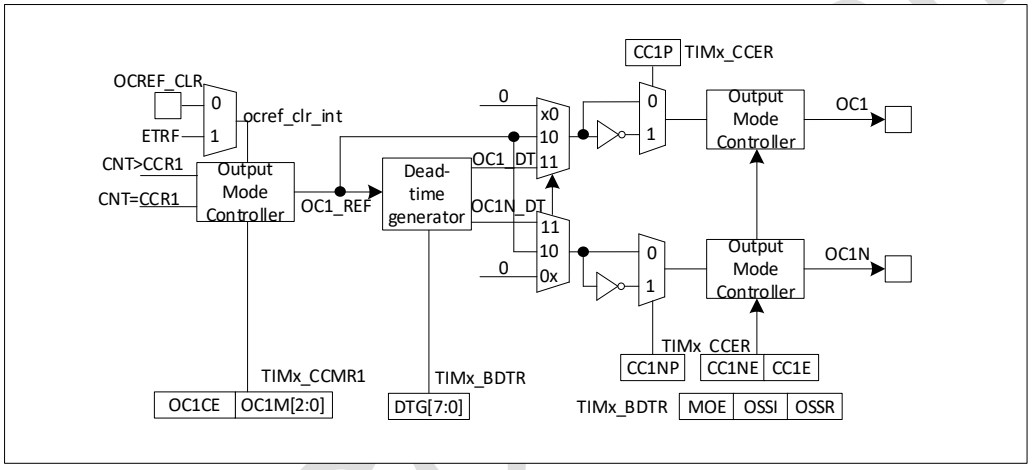


图 14-29 捕获/比较通道的输出部分(通道 1 至 3)

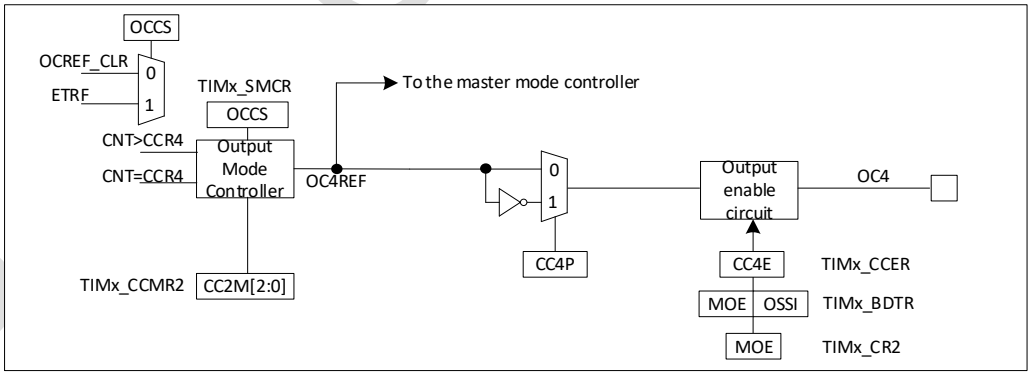


图 14-30 捕获/比较通道的输出部分(通道 4)

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。

14.3.6. 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。
- 如果需要，通过设置 TIMx_DIER 寄存器中的 CC1IE 位允许相关中断请求。
- 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。

14.3.7. 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 TI2FP2 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。

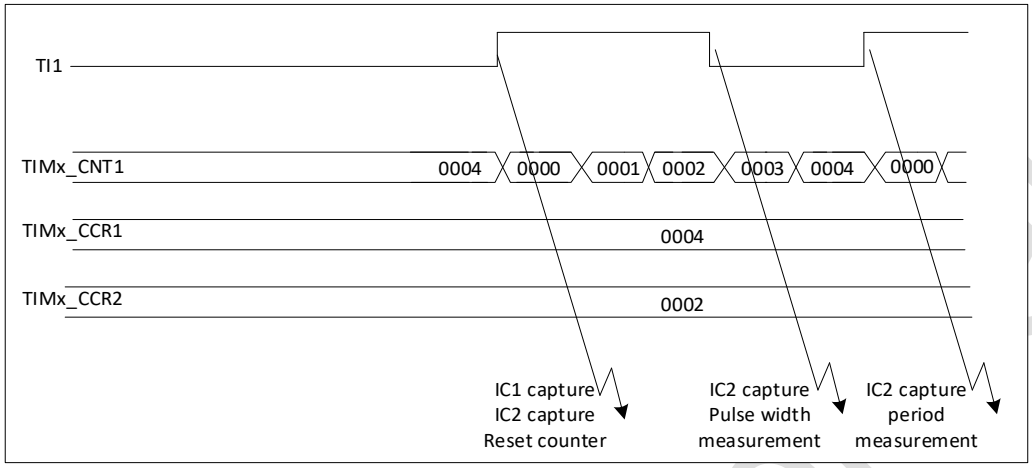


图 14-31 PWM 输入模式时序

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

14.3.8. 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。这将会在下面的输出比较模式一节中介绍。

14.3.9. 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

5. 设置 TIMx_CR1 寄存器的 CEN 位启动计数器

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

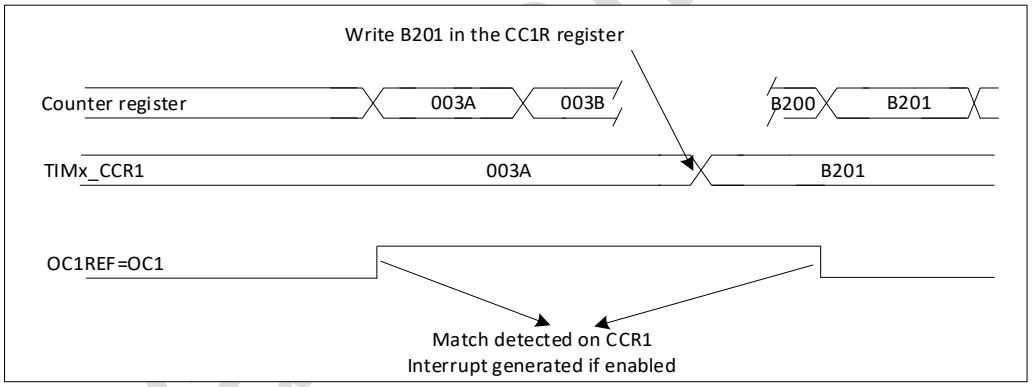


图 14-32 输出比较模式，翻转 OC1

14.3.10. 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。

14.3.10.1. 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。

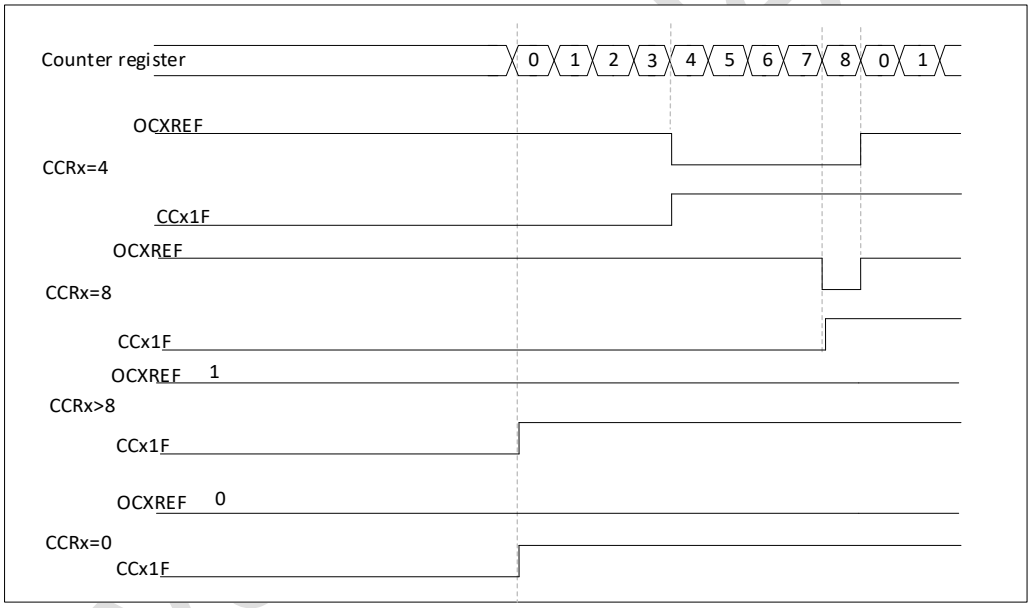


图 14-33 边沿对齐方式 PWM 输出，向上（ARR=8）

- **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。

14.3.10.2. 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

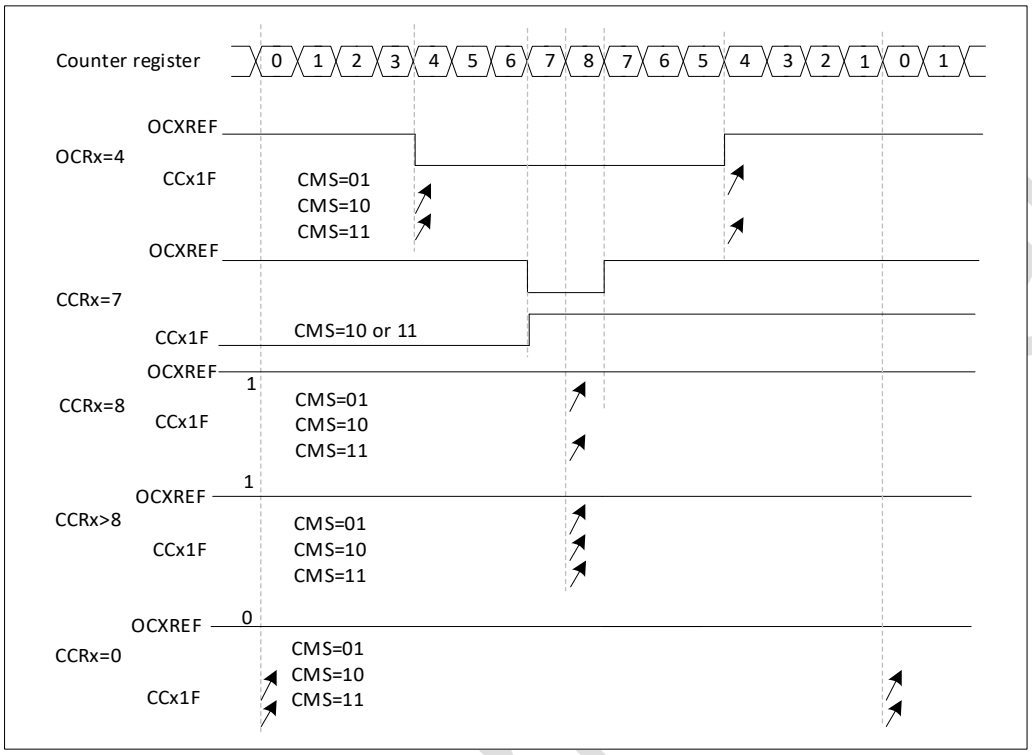


图 14-34 中央对齐的 PWM 波形(APR=8)

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。

14.3.11. 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]，通过配置 TIMx_BDTR 寄存器配置。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)

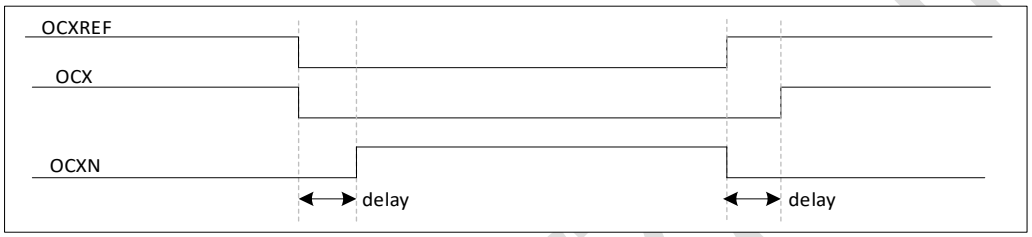


图 14-35 带死区插入的互补输出

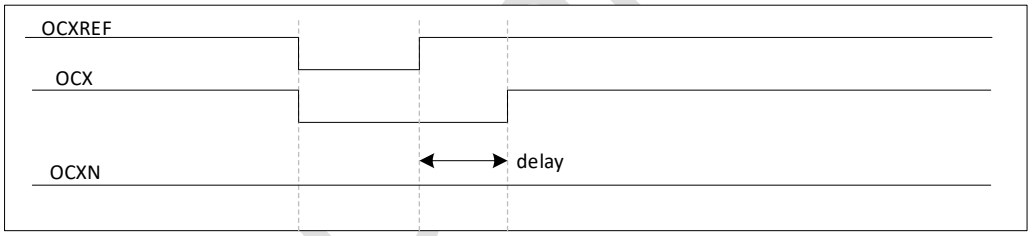


图 14-36 死区波形延迟大于负脉冲

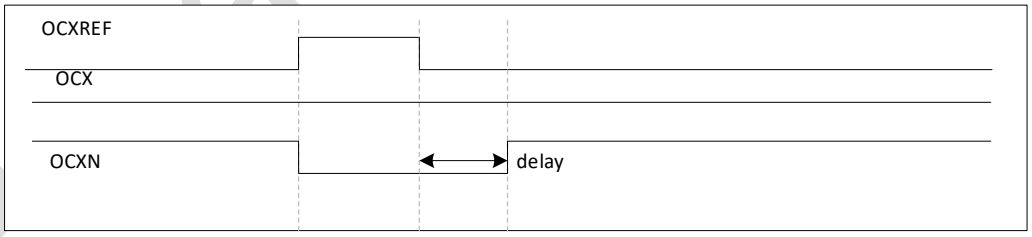


图 14-37 死区波形延迟大于正脉冲

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。

14.3.12. 使用刹车功能

当使用刹车功能时，依据相应的控制位(TIMx_BDTR 寄存器中的 MOE、OSSI 和 OSSR 位，TIMx_CR2 寄存器中的 OISx 和 OISxN 位)，输出使能信号和无效电平都会被修改。但无论何时，OCx 和 OCxN 输出不能在同一时间同时处于有效电平上。详见表 5-2 刹车功能的互补输出通道 OCx 和 OCxN 的控制位。刹车源既可以是刹车输入引脚又可以是以下事件：

- CPU LOCKUP 输出
- SRAM 奇偶校验错误信号
- 由 CSS 监测产生的时钟 failure 事件
- 来自比较器的输出

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。刹车也可以通过软件设置 TIMx_EGR 寄存器中的 BG 位来产生。

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。

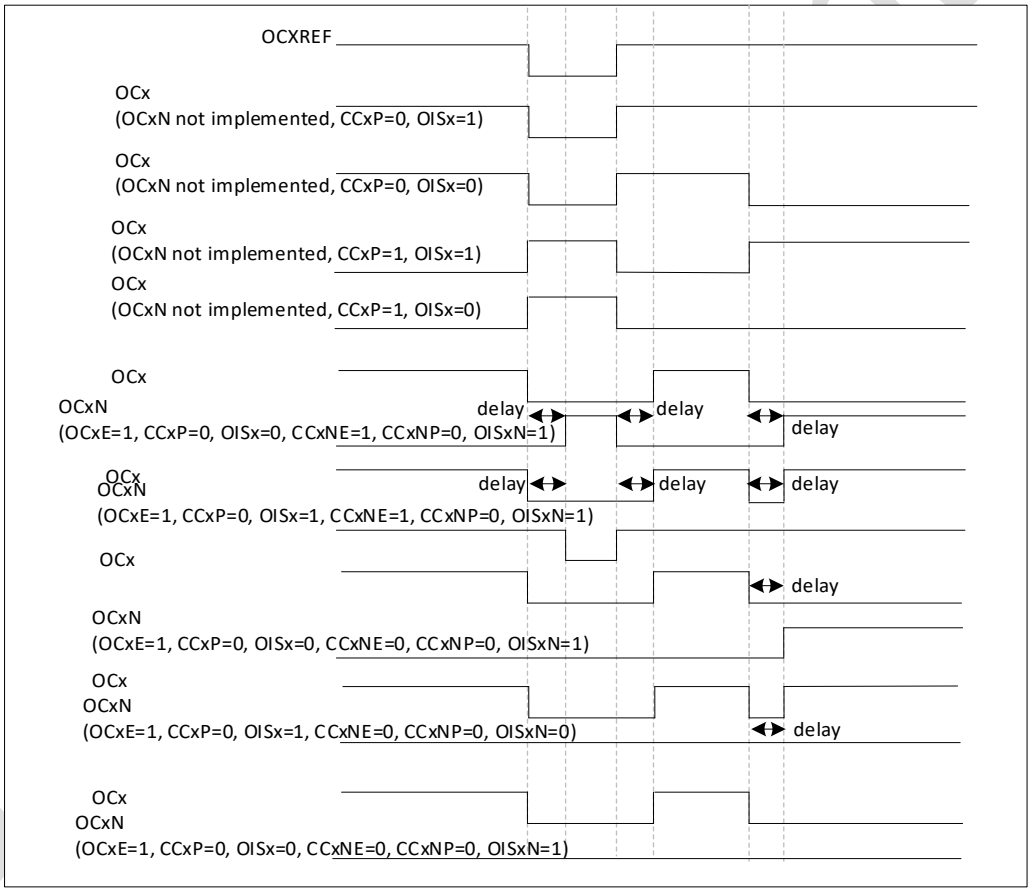


图 14-38 响应刹车的输出

14.3.13. 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'，能够用用 OCREF_CLR_INPUT 输入端的高电平把 OCxREF 信号拉低，OCxREF 信号将保持为低直到发生下一次计数溢出所产生的更新事件 UEV。该功能只能用于输出比较和 PWM 模式，而不能用于强置模式。

而 OCREF_CLR_INPUT 可以通过配置 TIMx_SMCR 寄存器中的 OCCS 位，在 OCREF_CLR 和 ETRF(ETR 滤波后)之间选择。

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。

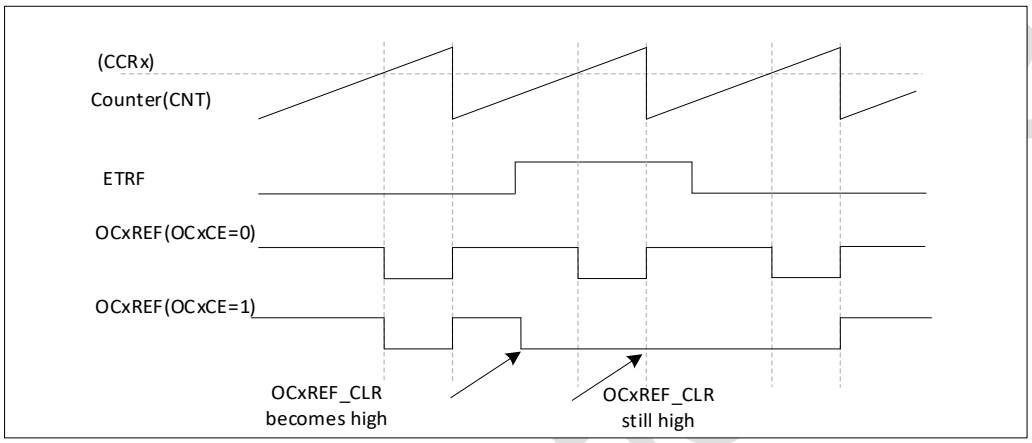


图 14-39 清除 TIM1 的 OCxREF

注：如果 PWM 占空比为 100% （若 CCRx>ARR），则在下一次计数器溢出时再次启用 OCxREF。

14.3.14. 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 OCxM、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。

下图显示当发生 COM 事件时，三种不同配置下 OCx 和 OCxN 输出。

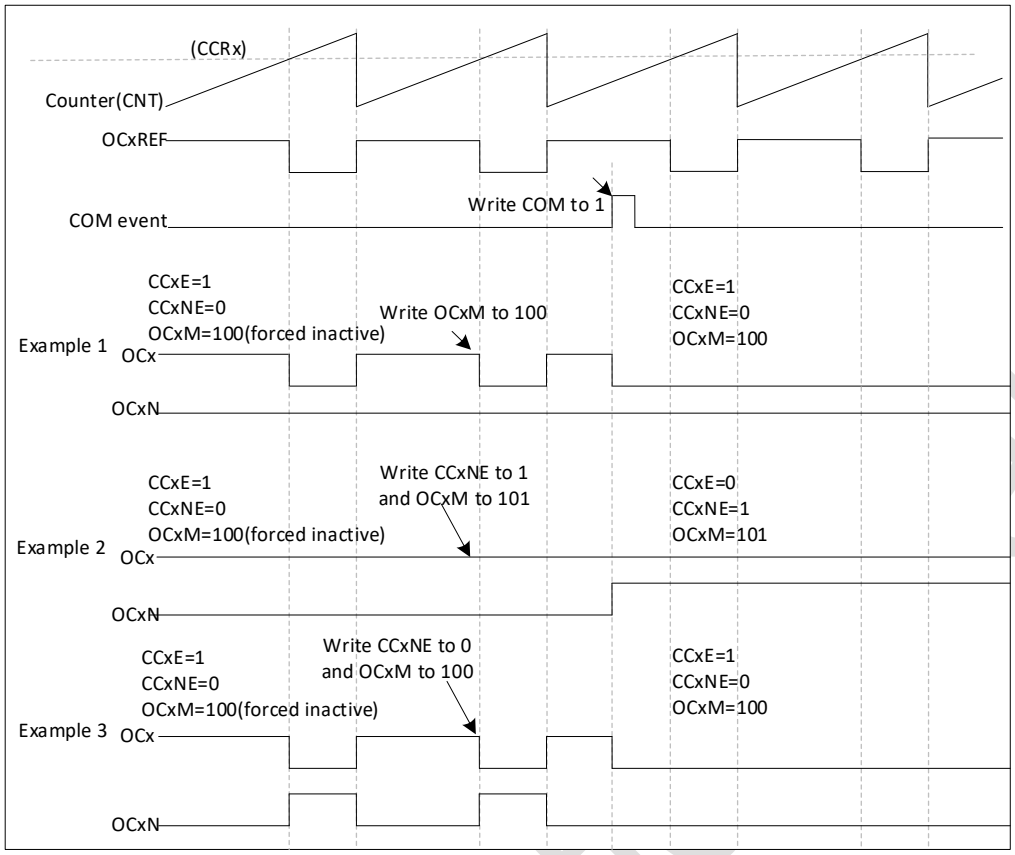


图 14-40 六步产生，COM 的例子(OSSR=1)

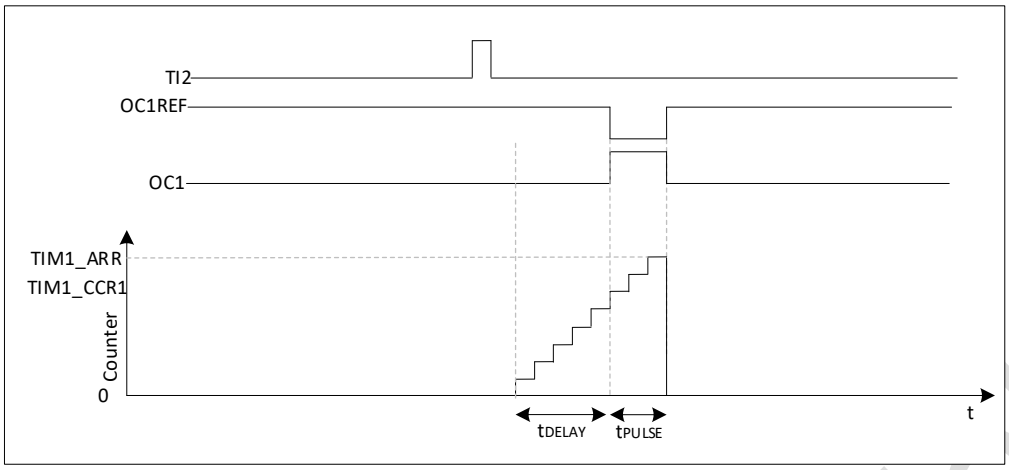
14.3.15. 单脉冲模式

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地在产生下一次计数溢出时停止。

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



7 图 14-41 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 tDELAY 之后，在 OC1 上产生一个长度为 tPULSE 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。

14.3.16. 可再触发单脉冲模式(OPM)

该模式允许计数器响应激励启动并产生长度可编程的脉冲，与上节所述的不可再触发单脉冲模式有以下区别：

- 触发一发生，脉冲就开始（无可编程延迟）。
- 如果在前一个触发所产生的脉冲完成之前发生新的触发，则延长脉冲。

要使用可重触发的单脉冲模式，计时器必须处于从模式，TIMx_SMCR 寄存器中的位 SMS[3:0]=“1000”（组合模式-复位+触发），OCxM[3:0]位设置为“1000”或“1001”（可重新触发 OPM 模式 1 或 2）。

如果此时计时器配置为递增计数模式，则相应的 CCRx 必须设置为 0（ARR 寄存器设置脉冲长度）。如果计时器配置为递减计数模式，CCRx 必须大于或等于 ARR。

注：出于兼容性原因，OCxM[3:0]和 SMS[3:0]位字段被分成两部分，最高有效位与 3 个最低有效位位置不连续。

该模式不得与中心对齐计数模式一起使用。TIMx_CR1 中必须有 CMS[1:0]=00。

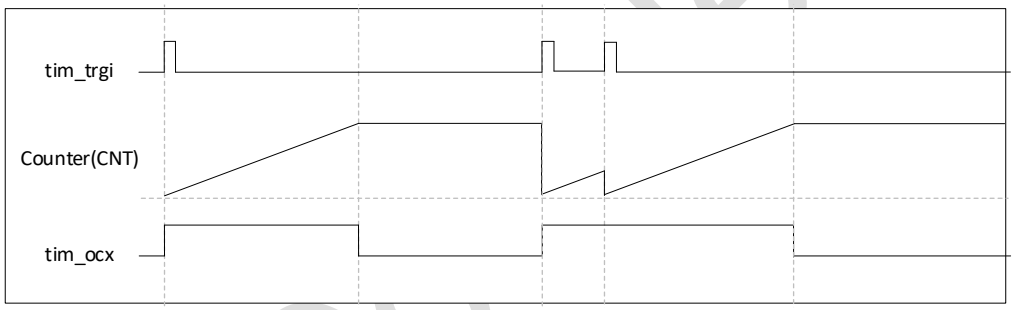


图 14-42 可重新触发单脉冲模式的示例

14.3.17. 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。参见表 14-1，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 14-1 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，TI2FP2 映射到 TI2)
- CC1P=’0’(TIMx_CCER 寄存器，TI1FP1 不反相，TI1FP1=TI1)
- CC2P=’0’(TIMx_CCER 寄存器，TI2FP2 不反相，TI2FP2=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)

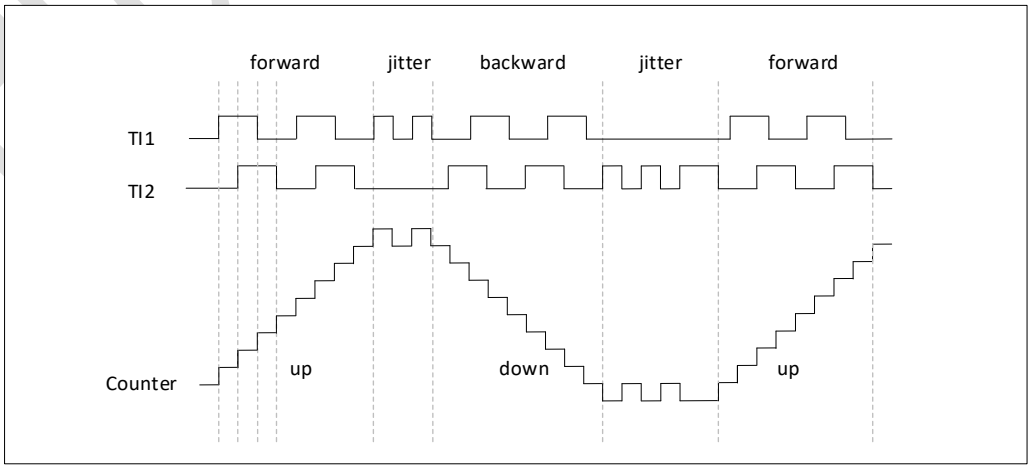


图 14-43 编码器模式下的计数器操作实例

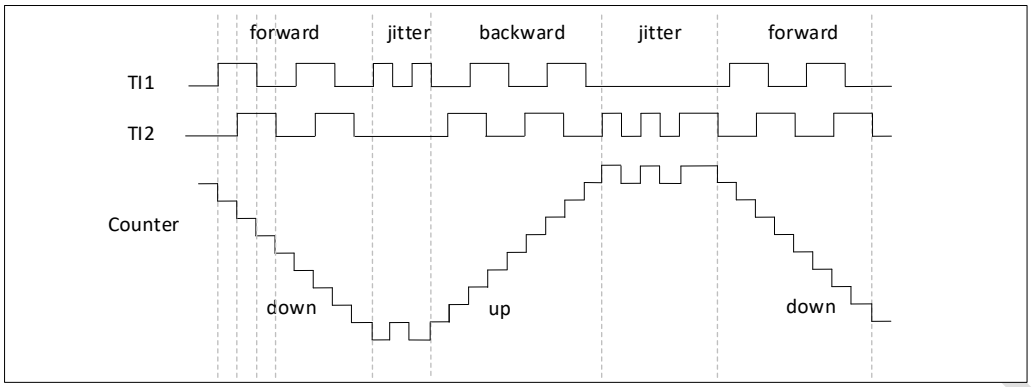


图 14-44 TI1P1 极性反转的编码器接口模式示例

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）。

14.3.18. 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。

14.3.19. 与霍尔传感器的接口

使用高级定时器（TIM1）产生 PWM 信号驱动马达时，可以使用另一个通用 timer 作为“接口定时器”来连接霍尔传感器。3 个定时器输入脚（CC1、CC2、CC3）通过一个异或门连接到 TI1 输入通道（通过设置 TIMx_CR2 寄存器中的 TI1S 位来选择），“接口定时器”捕获这个信号。

从模式控制器被配置到复位模式，从输入是 TI1F_ED。每当 3 个输入之一变化时，计数器重新从 0 开始计数。这样产生一个由霍尔输入端的任何变化而触发的时间基准。

接口定时器上的捕获/比较通道 1 配置为捕获模式，捕获信号为 TR。捕获值反映了两个输入变化间的时间延迟，给出了马达速度的信息。

接口定时器可以用来在输出模式产生一个脉冲，这个脉冲可以（通过触发一个 COM 事件）用于改变高级定时器 TIM1 各个通道的属性，而高级定时器产生 PWM 信号驱动马达。因此接口定时器通道必须编程为一个指定的延迟（输出比较或 PWM 模式）之后产生一个正脉冲，这个脉冲通过 TRGO 输出被送到高级定时器 TIM1。

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

- 置 TIMx_CR2 寄存器的 TI1S 位为’1’，配置三个定时器输入逻辑或到 TI1 输入。
- 时基编程：置 TIMx_ARR 为其最大值(计数器必须通过 TI1 的变化清零)。设置预分频器得到一个最大的计数器周期，它长于传感器上的两次变化的时间间隔。

- 设置通道 1 为捕获模式(选中 TRC)：置 TIMx_CCMR1 寄存器中 CC1S=01，如果需要，还可以设置数字滤波器。
- 设置通道 2 为 PWM2 模式，并具有要求的延时：置 TIMx_CCMR1 寄存器中的 OC2M=111 和 CC2S=00。
- 选择 OC2REF 作为 TRGO 上的触发输出：置 TIMx_CR2 寄存器中的 MMS=101。

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

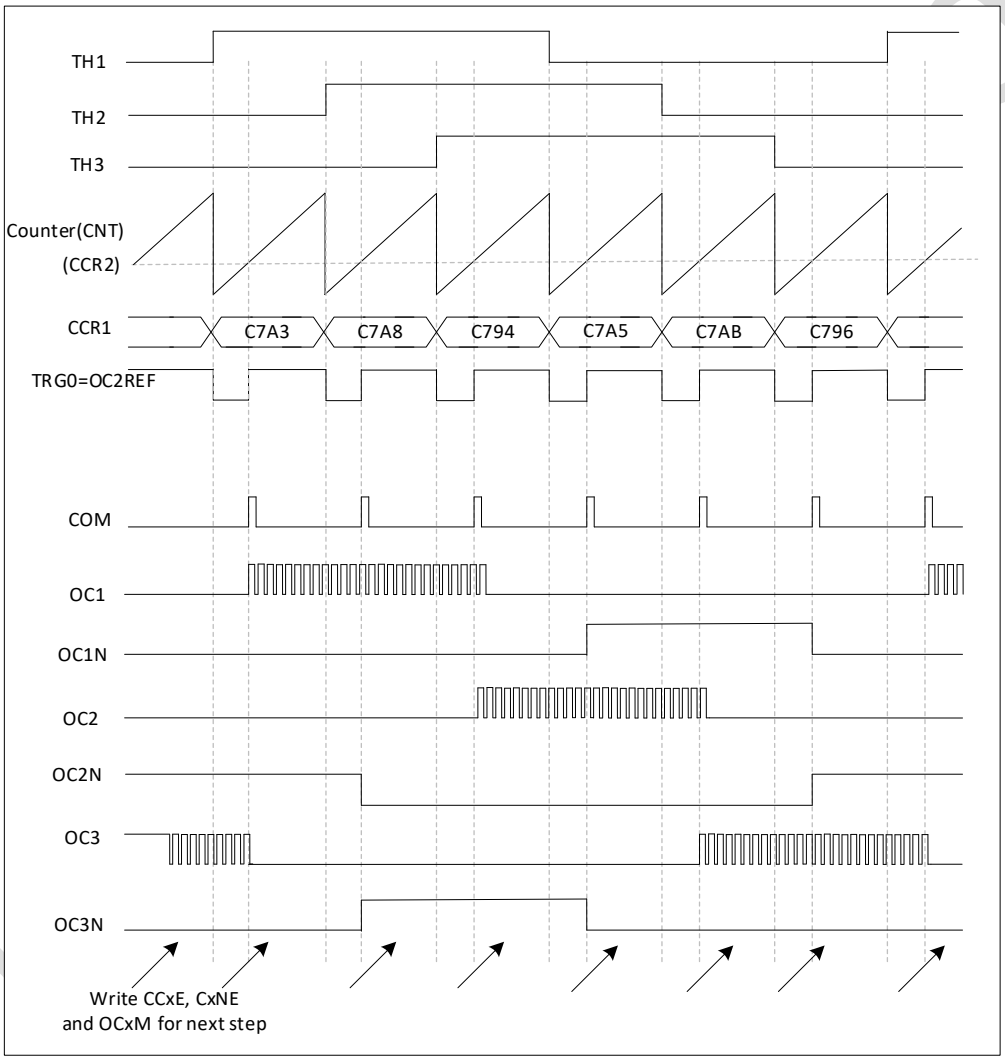


图 14-45 霍尔传感器接口的实例

14.3.20. 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，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。

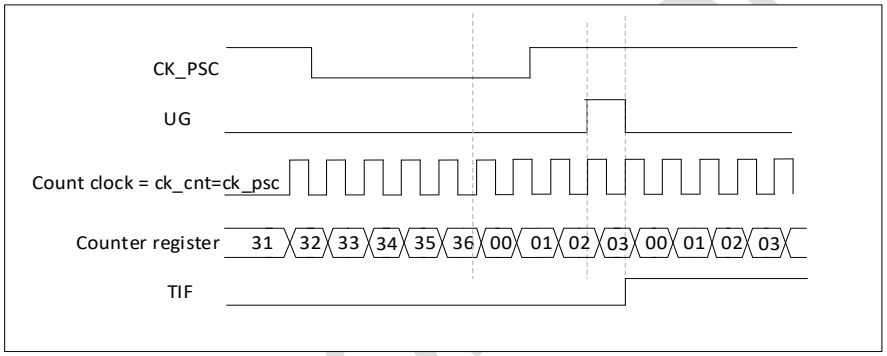


图 14-46 复位模式下的控制电路

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。

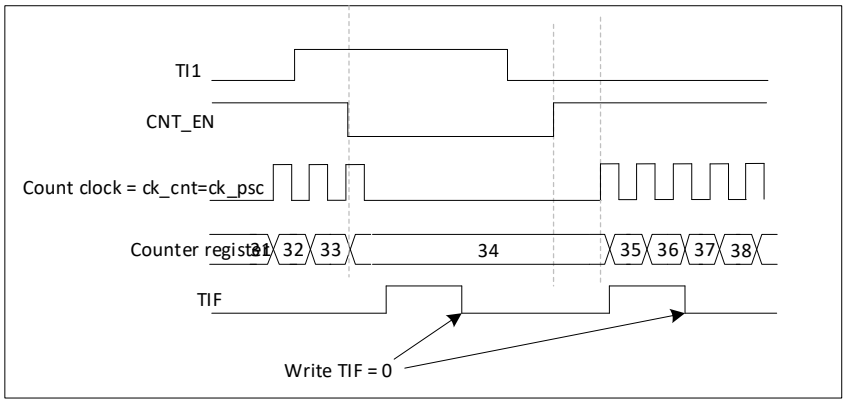


图 14-47 门控模式下的控制电路

从模式：触发模式

输入端上选中的事件使能计数器。

在下面的例子中，计数器在 TI2 输入的上升沿开始向上计数：

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

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

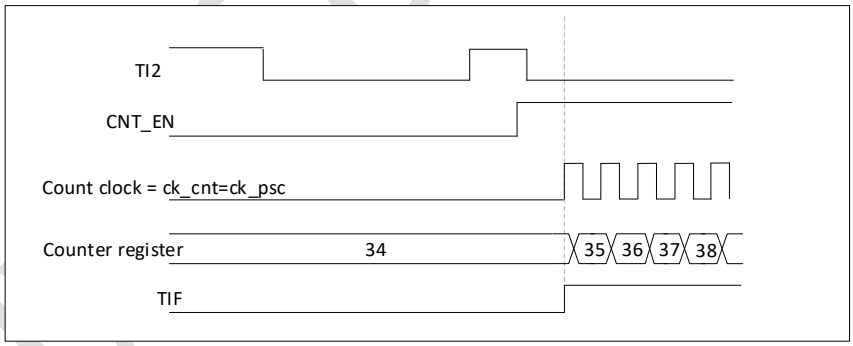


图 14-48 门控模式下的控制电路

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。

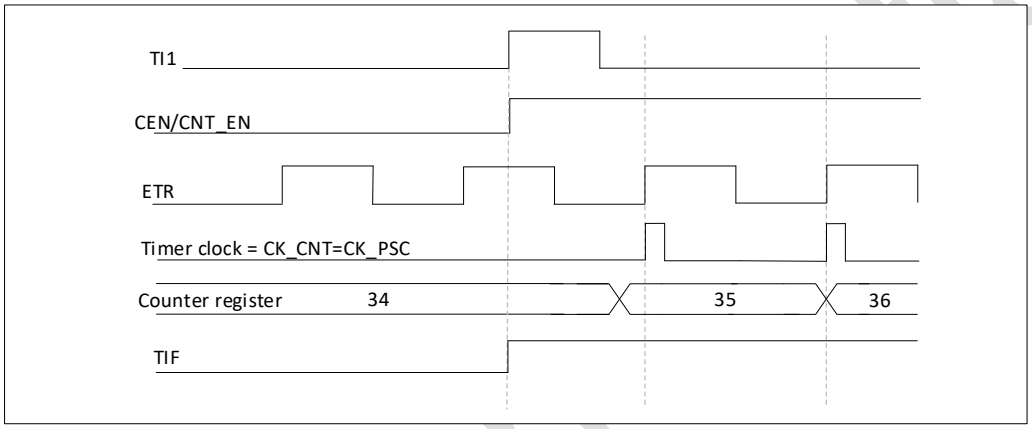


图 14-49 外部时钟模式 2 + 触发模式下的控制电路

14.3.21. 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。

14.4. TIM1 寄存器描述

14.4.1. TIM1 控制寄存器 1 (TIM1_CR1)

偏移地址：0x00

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
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	保留	-	-	保留
9:8	CKD[1:0]	RW	00	时钟分频因子

Bit	Name	R/W	Reset Value	Function
				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 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

Bit	Name	R/W	Reset Value	Function
				– 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。

14.4.2. TIM1 控制寄存器 2 (TIM1_CR2)

偏移地址：0x04

复位值：0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
				0：当 MOE=0 时，如果实现了 OC1N，则死区后 OC1=0 1：当 MOE=0 时，如果实现了 OC1N，则死区后 OC1=1 注：已经设置了 LOCK(TIM1_BKR 寄存器)级别 1、 2 或 3 后，该位不能被修改。
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)。 注意： 1.从定时器和 ADC 的时钟必须先被使能以接收主定时器的信号，并在接收时不要改变。 2.若主从定时器不在同一总线上，主模式应该配置为能被从定时器采到的宽度。
3	保留	-	-	保留
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	保留	-	-	保留

Bit	Name	R/W	Reset Value	Function
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。

14.4.3. TIM1 从模式控制寄存器 (TIM1_SMCR)

偏移地址：0x08

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
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	保留	-	-	保留
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：禁止外部时钟模式 2； 1：使能外部时钟模式 2。计数器由 ETRF 信号上的任意有效边沿驱动。 注 1：设置 ECE 位与选择外部时钟模式 1 并将 TRGI 连到 ETRF(SMS=111 和 TS=111)具有相同功效。 注 2：下述从模式可以与外部时钟模式 2 同时使用：复位模式，门控模式和触发模式；但是，这时 TRGI 不能连到 ETRF(TS 位不能是‘111’)。 注 3：外部时钟模式 1 和外部时钟模式 2 同时被使能时，外部时钟的输入是 ETRF
13:12	ETPS[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。当输入较快的外部时钟时，可以使用预分频降低 ETRP 的频率。 00：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。这些位定义了对 ETRP 信号采样的频率和对 ETRP 数字滤波的带宽。实际上，数字滤波器是一个事件计数器，它记录到 N 个事件后会产生一个输出的跳变。 0000：没有滤波器，在 fDTS 下采样 0001：fSAMPLING=fCK_INT, N=2

Bit	Name	R/W	Reset Value	Function
				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：保留(ITR0) 注：这些位只能在未用到(如 SMS=000)时被改变，以避免在改变时产生错误的边沿检测
3	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) 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 TI2FP2 的电平，计数器在 TI2FP1 的边沿向上/下计数。 若 SMS[3]=0: 010：编码器模式 2 根据 TI2FP1 的电平，计数器在 TI2FP2 的边沿向上/下计数。 011：编码器模式 3 根据其他输入的电平，计数器在 TI1FP1 和 TI2FP2 的边沿向上/下计数。 100：复位模式 选中的触发输入(TRGI)的上升沿重新初始化计数器，并且产生一个更新寄存器的信号。 101：门控模式

Bit	Name	R/W	Reset Value	Function
				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。 若 SMS[3]=1，SMS[2:0]必须配置为 0. 000：组合“复位+触发”模式 - 所选触发输入（tim_trgi）的上升沿重新初始化计数器，生成寄存器更新并启动计数器。 注：在编码器模式下，不要使用 uev 作为 trgo 输出信号，（即 mms 不能配置为 010）

表格 14-1 TIM1 内部触发连接

Slave TIM	ITR0(TS=000)
TIM1	TIM14

14.4.4. TIM1 中断使能寄存器 (TIM1_DIER)

偏移地址：0x0C

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
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	COMIE	CC4IE	CC3IE	CC2IE	CC1IE	UIE
-								RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31：8	保留	-	-	保留
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

Bit	Name	R/W	Reset Value	Function
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
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

14.4.5. TIM1 状态寄存器(TIM1_SR)

偏移地址：0x010

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	IC4IF	IC3IF	IC2IF	IC1IF	IC4IR	IC3IR	IC2IR	IC1IR
-	-	-	-	-	-	-	-	RC_W0	RC_W0	RC_W0	RC_W0	RC_W0	RC_W0	RC_W0	RC_W0
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	CC1F	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
31：24	保留	-	-	保留
23	IC4IF	RC_W0	0	下降沿捕获 4 标志 参见 IC1IF 描述。
22	IC3IF	RC_W0	0	下降沿捕获 3 标志 参见 IC1IF 描述。
21	IC2IF	RC_W0	0	下降沿捕获 2 标志 参见 IC1IF 描述。
20	IC1IF	RC_W0	0	下降沿捕获 1 标志 仅当相应的通道被配置为输入捕获时且由下降沿触发捕获事件，该标记可由硬件置 1。它由软件清‘0’或通过读 TIMx_CCR1 清‘0’。 0：无重复捕获产生； 1：发生下降沿捕获事件。
19	IC4IR	RC_W0	0	上升沿捕获 4 标志 参见 IC1IR 描述。
18	IC3IR	RC_W0	0	上升沿捕获 3 标志 参见 IC1IR 描述。
17	IC2IR	RC_W0	0	上升沿捕获 2 标志 参见 IC1IR 描述。
16	IC1IR	RC_W0	0	上升沿捕获 1 标志

Bit	Name	R/W	Reset Value	Function
				仅当相应的通道被配置为输入捕获时且由上升沿触发捕获事件，该标记可由硬件置 1。它由软件清‘0’或通过读 TIMx_CCR1 清‘0’。 0：无重复捕获产生； 1：发生上升沿捕获事件。
15:13	保留	-	-	保留
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	保留	-	-	保留
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 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

Bit	Name	R/W	Reset Value	Function
				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)。
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)）

14.4.6. TIM1 事件产生寄存器(TIM1_EGR)

偏移地址：0x14

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
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	保留	-	-	保留
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，若开启对应的中断，则产生相应的中断。
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。

Bit	Name	R/W	Reset Value	Function
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。 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。

14.4.7. TIM1 捕获/比较模式寄存器 1(TIM1_CCMR1)

偏移地址：0x18

复位值：0x0000 0000

输出比较模式：

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	保留	-	-	保留
15	OC2CE	RW	0	输出比较 2 清 0 使能
14:12	OC2M[2:0]	RW	0	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

Bit	Name	R/W	Reset Value	Function
9:8	CC2S[1:0]	RW	0	捕获/比较 2 选择。 该位定义通道的方向（输入/输出），及输入脚的选择: 00：CC2 通道被配置为输出; 01：CC2 通道被配置为输入，IC2 映射在 TI2 上; 10：CC2 通道被配置为输入，IC2 映射在 TI1 上; 11：CC2 通道被配置为输入，IC2 映射在 TRC 上。此模式仅工作在内部触发器输入被选中时 （由 TIM1_SMCR 寄存器的 TS 位选择）。 注： CC2S 仅在通道关闭时(TIM1_CCER 寄存器的 CC2E=0)才是可写的。
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。
6:4	OC1M[2:0]	RW	0	输出比较 1 模式 该位定义了输出参考信号 OC1REF 的动作，而 OC1REF 决定了 OC1、 OC1N 的值。 OC1REF 是高电平有效，而 OC1、 OC1N 的有效电平取决于 CC1P、 CC1NP 位。 000：冻结。输出比较寄存器 TIM1_CCR1 与计数器 TIM1_CNT 间的比较对 OC1REF 不起作 用; 001：匹配时设置通道 1 为有效电平。当计数器 TIMx_CNT 的值与捕获/比较寄存器 1(TIMx_CCR1)相同时，强制 OC1REF 为高。 010：匹配时设置通道 1 为无效电平。当计数器 TIMx_CNT 的值与捕获/比较寄存器 1(TIMx_CCR1)相同时，强制 OC1REF 为低。 011：翻转。当 TIM1_CCR1=TIM1_CNT 时，翻转 OC1REF 的电平。 100：强制为无效电平。强制 OC1REF 为低。 101：强制为有效电平。强制 OC1REF 为高。 110： PWM 模式 1 - 在向上计数时，一旦 TIM1_CNT<TIM1_CCR1 时通道 1 为有效电平，否则为无效电平；在向下计数时，一旦 TIM1_CNT>TIM1_CCR1 时通道 1 为无效电平(OC1REF=0)，否则为有效电平(OC1REF=1)。 111： PWM 模式 2 - 在向上计数时，一旦 TIM1_CNT<TIM1_CCR1 时通道 1 为无效电平，否则为有效电平；在向下计数时，一旦 TIM1_CNT>TIM1_CCR1 时通道 1 为有效电平，否则为无效电平。 注 1：一旦 LOCK 级别设为 3(TIMx_BDTR 寄存器中的 LOCK 位)并且 CC1S=00(该通道配置成输出)则该位不能被修改。 注 2：在 PWM 模式 1 或 PWM 模式 2 中，只有当比较结果改变了或在输出比较模式中从冻结模式切换到 PWM 模式时， OC1REF 电平才改变。
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。

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

输入捕获模式:

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	保留	-	-	保留
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）。

Bit	Name	R/W	Reset Value	Function
				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， 它记录到 N 个事件后会产生一个输出的跳变： 0000：无滤波器，以 f_{DTS} 采样 1000：采样频率 $f_{SAMPLING}=f_{DTS}/8$， N=6 0001：采样频率 $f_{SAMPLING}=f_{CK\_INT}$，N=2 1001：采样频率 fSAMPLING=fDTS/8，N=8 0010：采样频率 $f_{SAMPLING}=f_{CK\_INT}$，N=4 1010：采样频率 fSAMPLING=fDTS/16，N=5 0011：采样频率 $f_{SAMPLING}=f_{CK\_INT}$，N=8 1011：采样频率 $f_{SAMPLING}=f_{DTS}/16$，N=6 0100：采样频率 $f_{SAMPLING}=f_{DTS}/2$，N=6 1100：采样频率 $f_{SAMPLING}=f_{DTS}/16$，N=8 0101：采样频率 $f_{SAMPLING}=f_{DTS}/2$，N=8 1101：采样频率 $f_{SAMPLING}=f_{DTS}/32$，N=5 0110：采样频率 $f_{SAMPLING}=f_{DTS}/4$，N=6 1110：采样频率 $f_{SAMPLING}=f_{DTS}/32$，N=6 0111：采样频率 $f_{SAMPLING}=f_{DTS}/4$，N=8 1111：采样频率 $f_{SAMPLING}=f_{DTS}/32$，N=8
3:2	IC1PSC[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； 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。

14.4.8. TIM1 捕获/比较模式寄存器 2(TIM1_CCMR2)

偏移地址：0x1C

复位值：0x0000 0000

输出比较模式:

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	保留	-	-	保留
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。

输入捕获模式:

Bit	Name	R/W	Reset Value	Function
31：16	保留	-	-	保留
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。 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；

Bit	Name	R/W	Reset Value	Function
				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。

14.4.9. TIM1 捕获/比较使能寄存器 (TIM1_CCER)

偏移地址：0x20

复位值：0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
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 CC1 通道配置为输出： 0：OC1 高电平有效 1：OC1 低电平有效 CC1 通道配置为输入： CC1NP/CC1P 位选择作为触发或捕获信号的 TI1FP1 和 TI2FP1 的极性。 00：不反相/上升沿： TIxFP1 上升沿有效（捕获、复位模式下触发、外部时钟或触发模式下）； TIxFP1 不反相（门控模式、编码器模式）。 01：反相/下降沿： TIxFP1 下降沿有效（捕获、复位模式下触发、外部时钟或触发模式下）； TIxFP1 反相（门控模式、编码器模式）。 10：保留，不要使用这个配置。 11：不反相/双沿 TIxFP1 上升和下降沿都有效（捕获、复位模式下触发、外部时钟或触发模式下）； TIxFP1 不反相（门控模式）。这个配置不能应用于编码器模式下。 注： 1.对于互补输出通道，这一位是预载的。如果 TIMx_CR2 寄存器中的 CCPC 位被设置，那么 CC1P 的实际有效位只有在 com 事件发生时才会加载预载值。 2.一旦 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 0：捕获使能 注： 对于互补输出通道，这一位是预载的。如果 TIMx_CR2 寄存器中的 CCPC 位被设置，那么 CC1E 的实际有效位只有在 com 事件发生时才会加载预载值。

表格 14-2 具有中断功能的互补 OCx 和 OCxN 通道的输出控制

控制位					输出状态	
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		

特别的，当死区有效时，输出总是按照死区内输出的规则输出，尽管此时 moe 可能已经被软件或硬件的方式置为 1。如果一个通道的 2 个输出都没有使用(CCxE = CCxNE = 0)，那么 OISx, OISxN, CCxP 和 CCxNP 都必须清零。

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

14.4.10. TIM1 计数器寄存器(TIM1_CNT)

偏移地址：0x24

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
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	保留	-	-	保留
15:0	CNT[15:0]	RW	0	计数器的值

14.4.11. TIM1 预分频器寄存器 (TIM1_PSC)

偏移地址：0x28

复位值：0x0000 0000

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

14.4.12. TIM1 自动重新加载寄存器 (TIM1_ARR)

偏移地址：0x2C

复位值：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	保留	-	-	保留
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。

14.4.13. TIM1 重复计数器寄存器(TIM1_RCR)

偏移地址：0x30

复位值：0x0000 0000

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

14.4.14. TIM1 捕获/比较寄存器 1(TIM1_CCR1)

偏移地址：0x34

复位值：0x0000 0000

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

14.4.15. TIM1 捕获/比较寄存器 2(TIM1_CCR2)

偏移地址：0x38

复位值：0x0000 0000

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

14.4.16. TIM1 捕获/比较寄存器 3 (TIM1_CCR3)

偏移地址：0x3C

复位值：0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
				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。

14.4.17. TIM1 捕获/比较寄存器 4(TIM1_CCR4)

偏移地址：0x40

复位值：0x0000 0000

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

14.4.18. TIM1 刹车和死区寄存器(TIM1_BDTR)

偏移地址：0x44

复位值：0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
				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（12.5.9 节，捕获/比较使能寄存器(TIM1_CCER)）。 0：当定时器不工作时，禁止 OC/OCN 输出（OC/OCN 使能输出信号=0）； 1：当定时器不工作时，一旦 CCxE=1 或 CCxNE=1，开启 OC/OCN 输出并输出无效电平。 OC/OCN 使能输出信号=1。 注：一旦 LOCK 级别(TIM1_BDTR 寄存器中的 LOCK 位)设为 2，则该位不能被修改。
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（12.5.9 节，捕获/比较使能寄存器(TIM1_CCER)）。 0：当定时器不工作时，禁止 OC/OCN 输出（OC/OCN 使能输出信号=0）； 1： 当 定 时器不 工 作时 ， 一旦 CCxE=1 或 CCxNE=1，OC/OCN 首 先 输出 其 空闲 电 平。 OC/OCN 使能输出信号=1。 注：一旦 LOCK 级别(TIM1_BDTR 寄存器中的 LOCK 位)设为 2，则该位不能被修改。
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；

Bit	Name	R/W	Reset Value	Function
				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. 通用定时器 (TIM14)

15.1. TIM14 简介

通用定时器 TIM14 由可编程预分频器驱动的 16 位自动装载计数器构成。

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

使用定时器预分频器，脉冲长度和波形周期可以在几个微秒到几个毫秒间调整。

TIM14 定时器都是完全独立的，没有互相共享任何资源。

15.2. TIM14 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
- 1 个独立通道，作为：
  - 输入捕获
  - 输出比较
  - PWM 生成（边缘对齐模式）
  - 单脉冲输出
- 如下事件发生时产生中断
  - 更新：计数器向上溢出，计数器初始化(通过软件)
  - 输入捕获
  - 输出比较

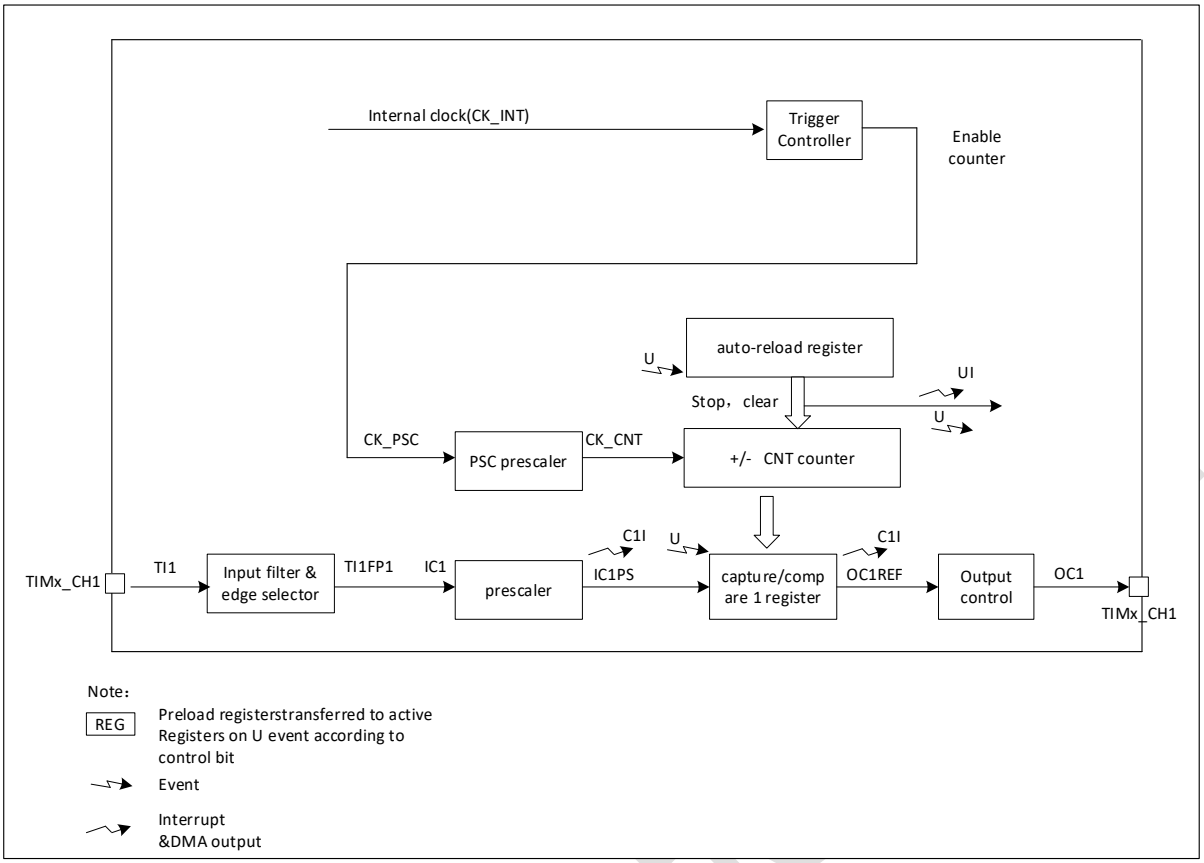


图 15-1 TIM14 架构框图

15.3. TIM14 功能描述

15.3.1. 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（TIM14_CNT）
- 预分频寄存器（TIM14_PSC）
- 自动重装载寄存器（TIM14_ARR）

自动重装载寄存器是预先装载的，写或者读自动重装载寄存器将访问预装载寄存器。根据在 TIM14_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一直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并当 TIM14_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 TIM14_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 TIM14_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 65536 之间的任意值分频。它是基于一个（在 TIM14_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。

下图给出了在计数器运行时，更改预分频器参数的例子。

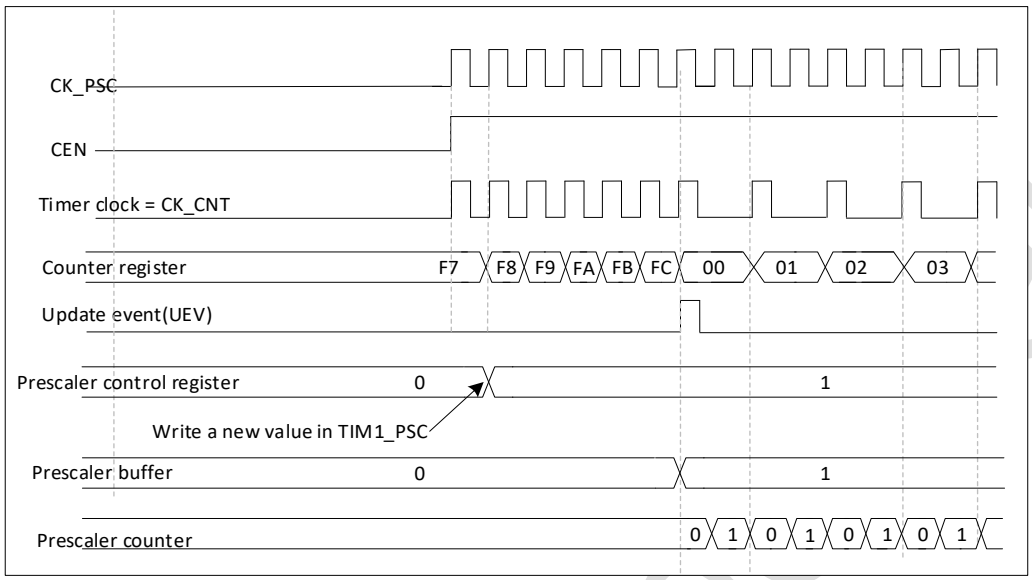


图 15-2 当预分频器的参数从 1 变到 2 时，计数器的时序图

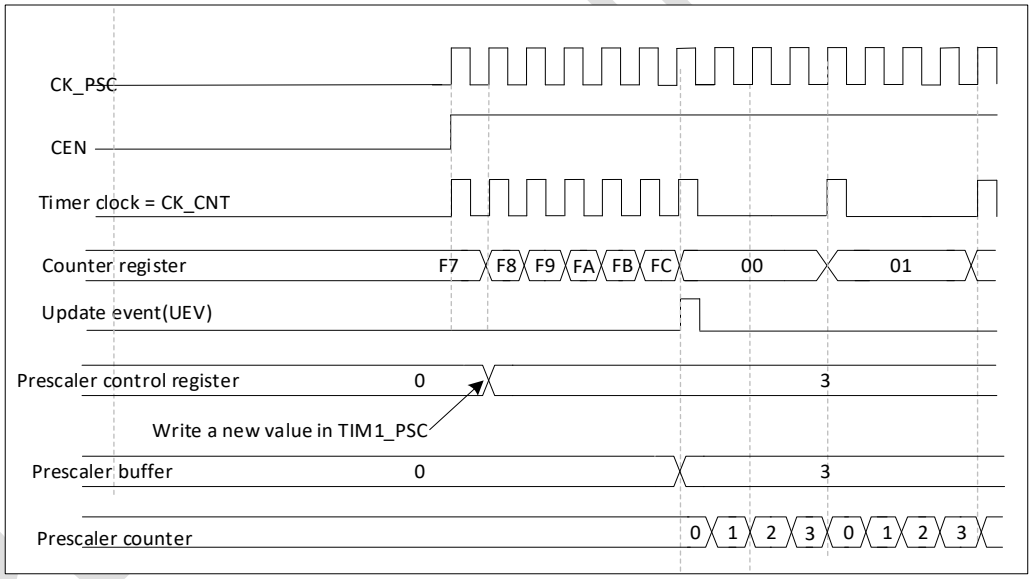


图 15-3 当预分频器的参数从 1 变到 4 时，计数器的时序图

15.3.2. 计数模式

向上计数模式

计数器从 0 计数到自动装载值（TIM14_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。

每个计数溢出时，产生更新事件。在 TIM14_EGR 寄存器中(通过软件方式)设置 UG 位也同样可以产生一个更新事件。

设置 TIM14_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清 0(但预分频器的数值不变)。此外，如果设置了 TIM14_CR1 寄存器中的 URS 位(选择更新请求)，设置 UG 位将产生一个更新事件 UEV，但硬件不设置 UIF 标志(即不产生中断或)。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(TIM14_SR 寄存器中的 UIF 位)。

- 自动装载影子寄存器被重新置入预装载寄存器的值(TIM14_ARR)。
- 预分频器的缓冲区被置入预装载寄存器的值(TIM14_PSC 寄存器的内容)。

下面的例程显示了几个在不同频率下的计数器行为，当 TIMx_ARR=0X36.

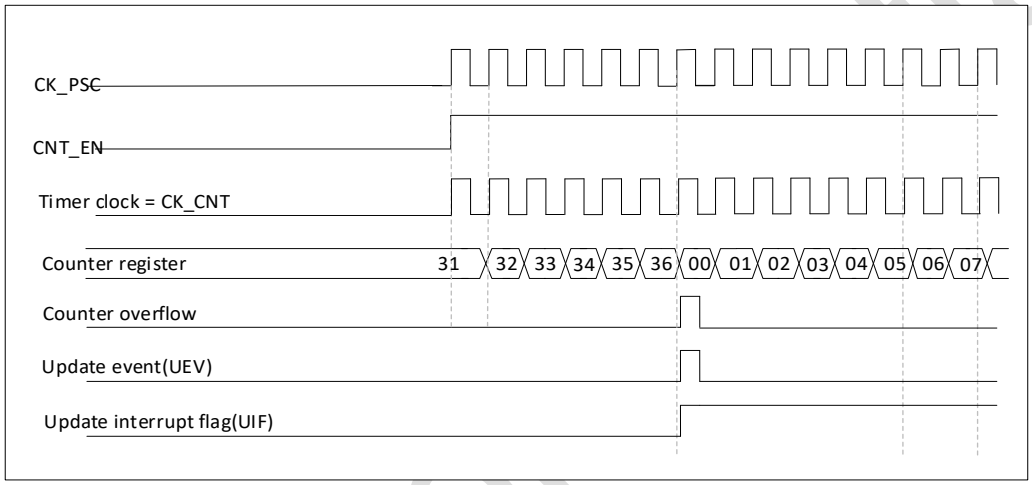


图 15-4 计数器时序图，内部时钟分频因子为 1

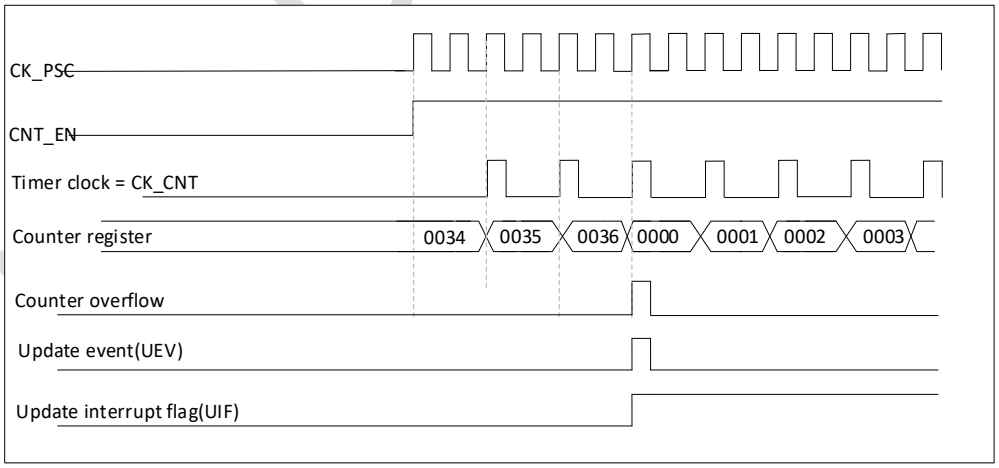


图 15-5 计数器时序图，内部时钟分频因子为 2

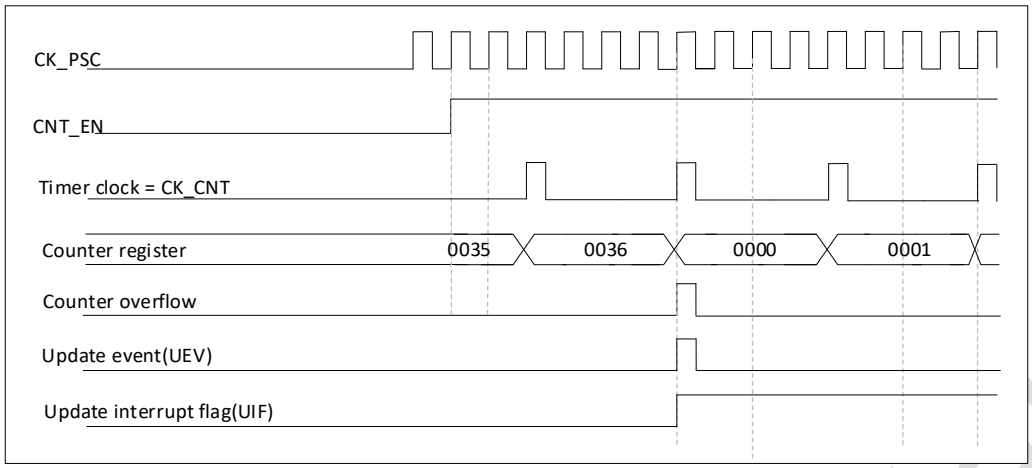


图 15-6 计数器时序图，内部时钟分频因子为 4

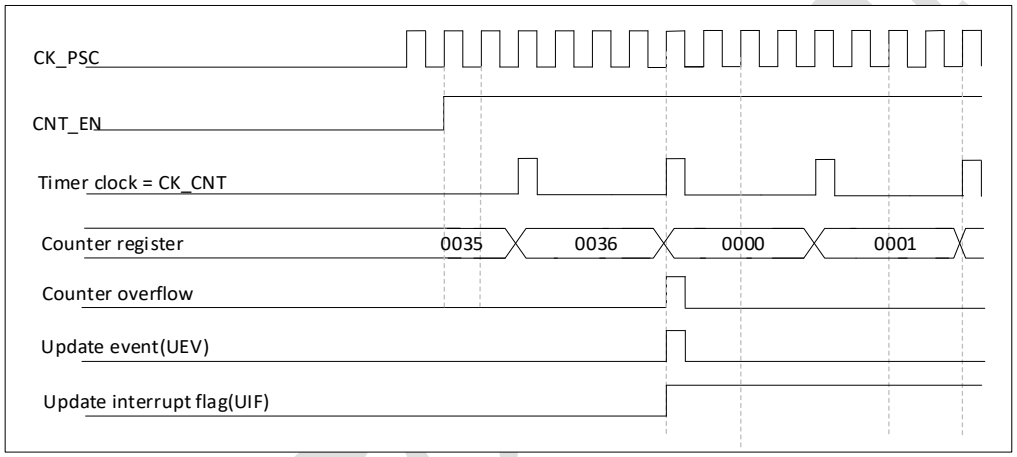


图 15-7 计数器时序图，内部时钟分频因子为 N

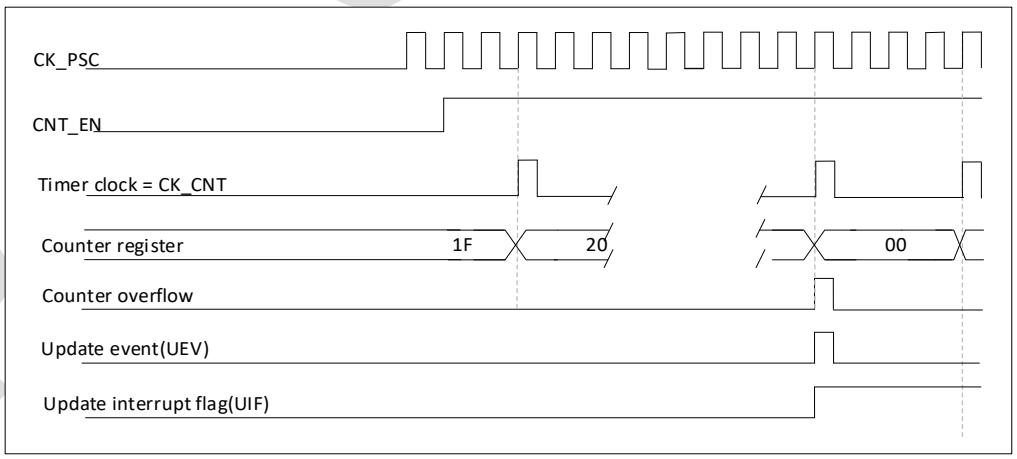


图 15-8 计数器时序图，当 ARPE=0 时的更新事件(TIMx_ARR 没有预装入)

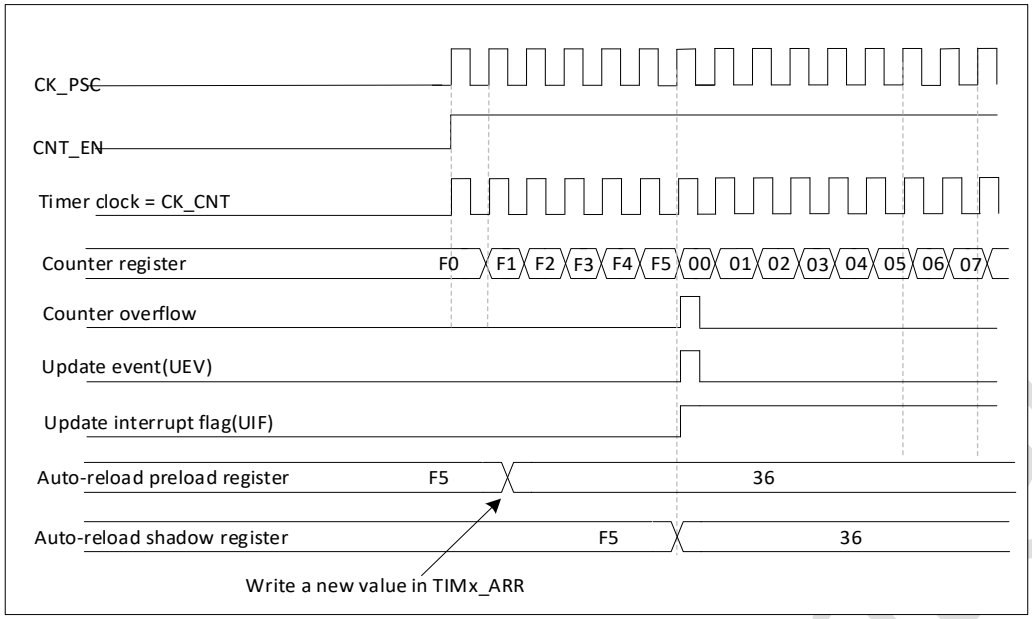


图 15-9 计数器时序图，当 ARPE=1 时的更新事件(预装入了 TIMx_ARR)

15.3.3. 时钟源

计数器的时钟由内部时钟（CK_INT）提供。TIMx_CR1 寄存器的 CEN 位和 TIM14_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。

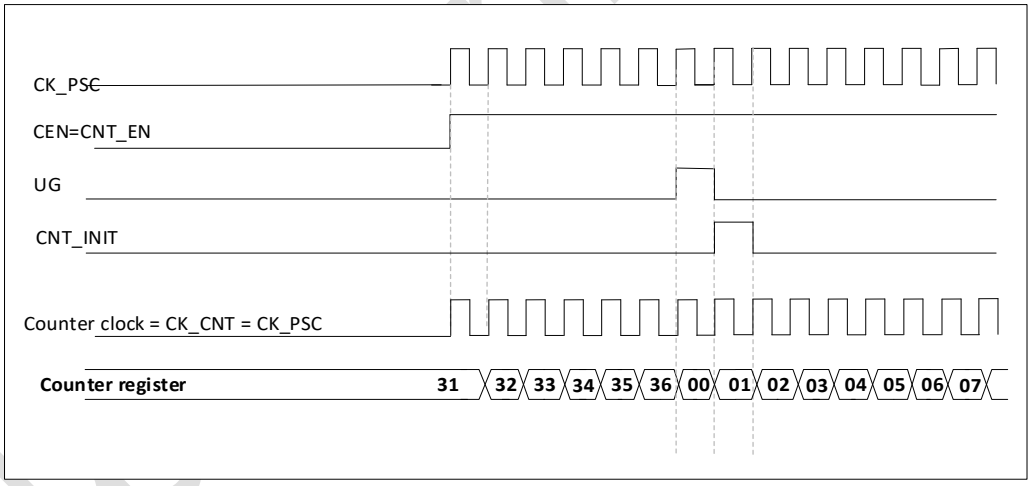


图 15-10 一般模式下的控制电路，内部时钟分频因子为 1

15.3.4. 捕获/比较通道

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

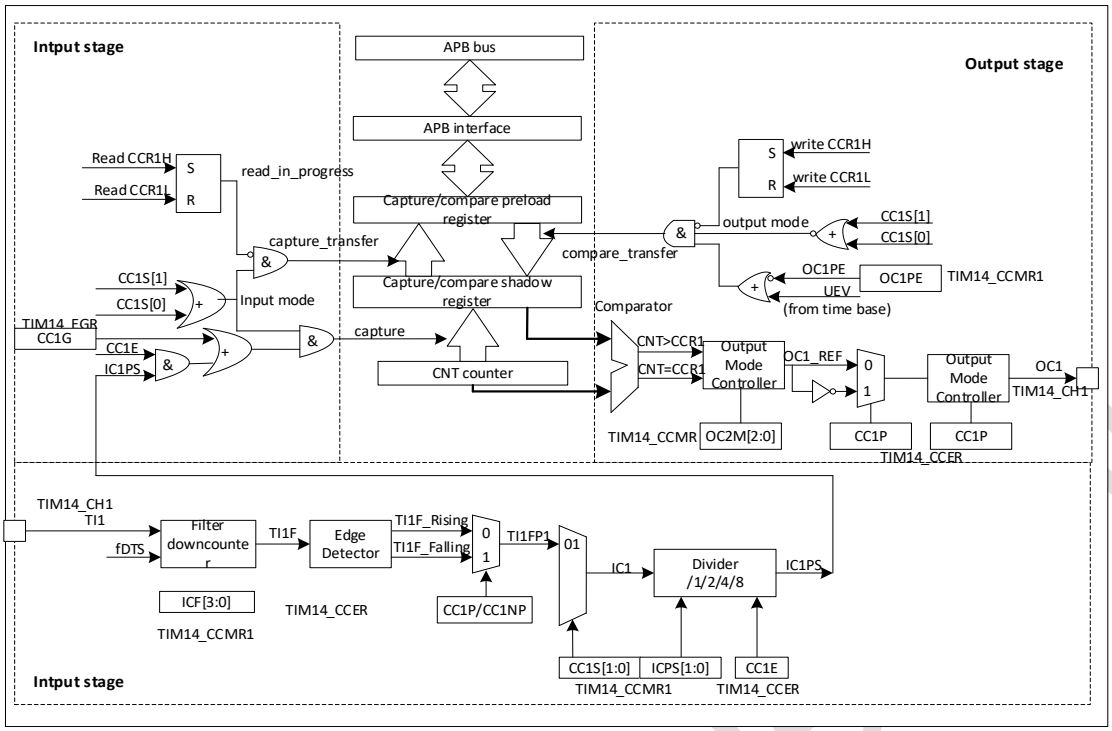


图 15-11 TIM14 框图

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

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

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。

15.3.5. 输入捕获模式

在输入捕获模式下，当检测到 Icx 信号相应的边沿后，计数器的当前值被锁存到捕获/比较寄存器（TIM14_CCRx）中。当捕获事件发生时，相应的 CcxIF 标志（TIM14_SR 寄存器）被置 1。如果捕获事件发生时，CcxIF 标志已经为高，那么重复捕获标志 CcxOF（TIMx_SR 寄存器）被置 1。写 CcxIF 可清除 CcxIF，或读取存储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 TIM14_CCR1 寄存器中，步骤如下：

- 选择有效输出端：TIM14_CCR1 必须连接到 TI1 输入，所以写入 TIM14_CCMR1 寄存器中的 CC1S=01，只要 CC1S 不为 00 时，通道被配置为输入，并且 TIM14_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宽度（即输入为 Tix 时，输入滤波器控制位是 TIM14_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，我们必须配置滤波器的宽度长于 5 个时钟周期。因此，我们可(以 fDTS 频率)连续采样 8 次，已确认在 TI1 上一次真是的边沿变化，然后再 TIM14_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）（和 CC1NP=0）
- 配置输入预分频器。在这个例子中，我们希望捕获发生在每一个有效的电平转换时刻，因此预分频器被禁止（写 TIMx_CCMR1 寄存器的 IC1PS=00）。
- 设置 TIMx_CCER 寄存器的 CC1E=1，允许捕获计数器的值到捕获寄存器中。
- 如果需要，通过设置 TIMx_DIER 寄存器中的 CC1E 位允许相关中断请求。

当发生一个输入捕获时：

- 产生有效的电平转换时，计数器的值被传送到 TIM14_CCR1 寄存器。
- CC1IF 标志被设置（中断标志）。当发生至少 2 个连续捕获时，而 CC1IF 未曾被清除， CC1OF 也被置 1.
- 如设置了 CC1E 位，则会产生一个中断请求。

为了处理中断溢出，建议在读出中断溢出标志之前，读取数据。这是为了避免丢失在读出中断溢出标志之后和读取数据之前可能产生的中断溢出信息。

注：输入捕获中断请求能通过软件设置在 TIMx_EGR 中相应的 CCxG 位来产生。

15.3.6. 强置输出模式

在该模式下（TIM14_CCMRx 寄存器中 CCxS bits = 00）下，输出比较信号（OCxREF 和相应的 Ocx）能够直接由软件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。

写 TIM14_CCMRx 寄存器中相应的 OcxM=101，即可强制输出比较信号（OCxREF/Ocx）为有效状态。这样 OCxREF 被强置为高电平（OCxREF 始终为高电平有效），同时 Ocx 得到 CCxP 极性位相反的值。

例如：CCxP=0(Ocx 高电平有效)，则 Ocx 被强置为高电平。

置 TIM14_CCMRx 寄存器中的 OcxM=100，可强置 OCxREF 信号为低。

该模式下，在 TIM14_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。

15.3.7. 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，输出比较工作做如下操作：

- 将输出比较模式(TIM14_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(TIM14_DIER 寄存器中的 CcxIE 位)，则产生一个中断。

TIM14_CCMRx 中的 OcxPE 位选择 TIM14_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。

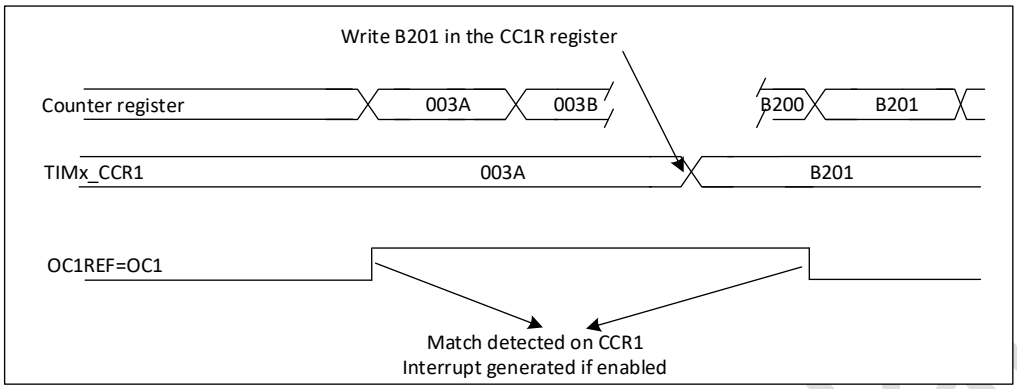


图 15-12 Output compare mode, toggle on OC1

15.3.8. 脉冲宽度调节（PWM）模式

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

在 TIM14_CCMRx 寄存器中的 OCxM 位写入“110”（PWM 模式 1）或“111”（PWM 模式 2），能够独立地设置每个 OCx 输出通道产生一路 PWM。必须设置 TIM14_CCMRx 寄存器 OCxPE 位以使能相应的预装载寄存器，最后还要设置 TIM14_CR1 寄存器中的 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 TIM14_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 TIM14_CCER 寄存器中的 CCxP 位设置，它可以设置为高电平有效或者低电平有效。TIM14_CCER 寄存器中的 CCxE 位控制 OCx 输出使能。

在 PWM 模式（模式 1 或者模式 2），TIM14_CNT 和 TIM14_CCRx 始终在进行比较，以确定是否符合 TIM14_CNT ≤ TIM14_CCRx。

定时器仅当计数器是向上计数时才能够产生边沿对齐模式的 PWM。

PWM 边沿对齐模式

下面是一个 PWM 模式 1 的例子。当 TIM14_CNT<TIMx_CCRx 时，PWM 参考信号 OCxREF 为高，否则为低。如果 TIM14_CCRx 中的比较值大于自动重装载值(TIM14_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。

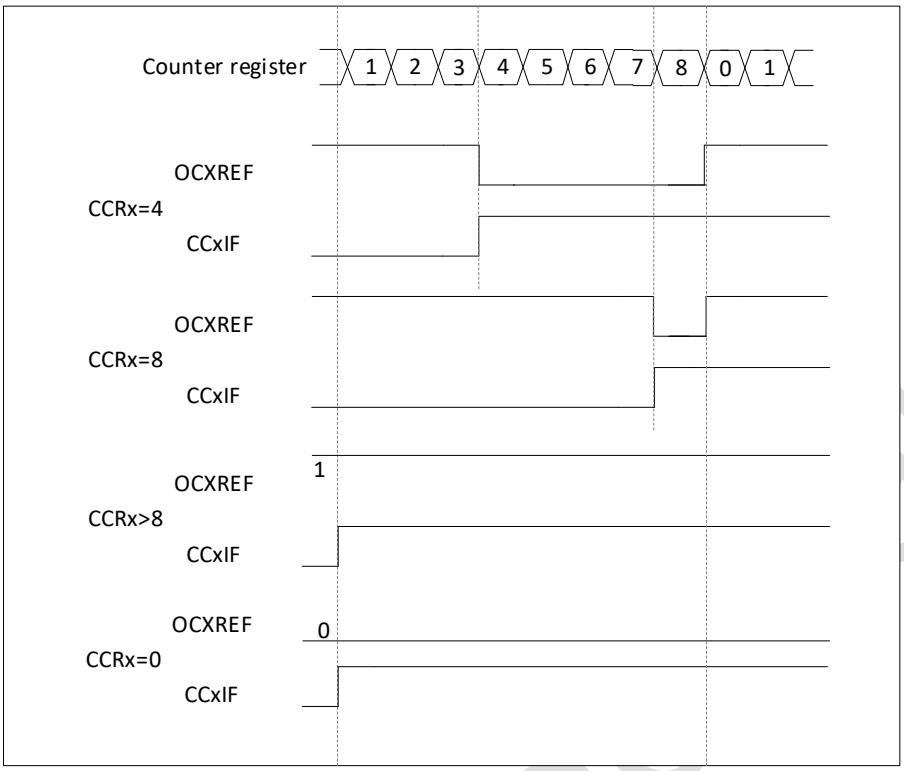


图 15-13 边沿对齐的 PWM 波形(ARR=8)

15.3.9. 单脉冲模式

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

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

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

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

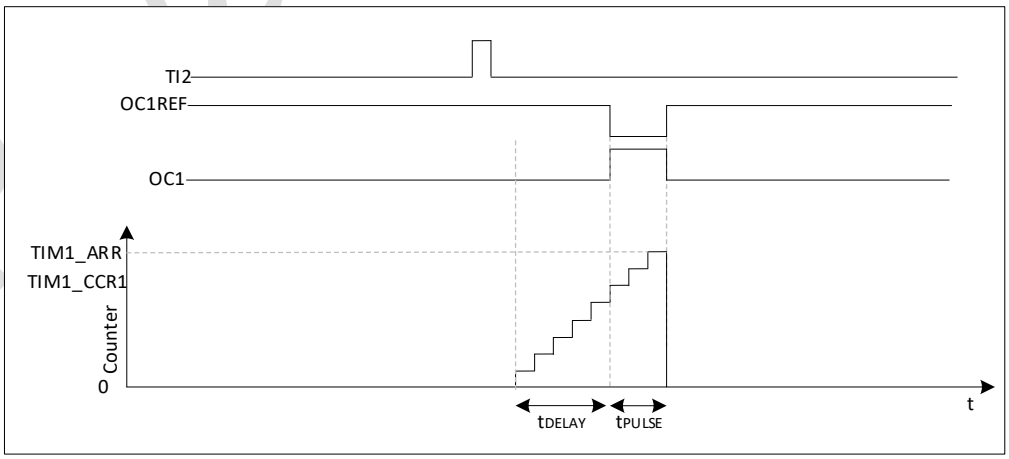


图 15-14 **Example of Single pulse mode**

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

假定 TI2FP2 作为触发:

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

OPM 的波形由写入比较寄存器的数值决定(要考虑时钟频率和计数器预分频器)

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

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

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

15.3.10. 同步模式

所有 TIMx 定时器在内部相连，用于定时器同步或链接。当一个定时器处于主模式时，它可以对另一个处于从模式的定时器的计数器进行复位、启动、停止或提供时钟等操作。

15.3.11. 调试模式

当芯片进入调试模式（M0+停止），根据 DBG 模块中 DBG_TIMx_STOP 的设置，TIMx 计数器或者继续正常操作，或者停止。

15.4. TIM14 寄存器

15.4.1. TIM14 控制寄存器 1 (TIM14_CR1)

偏移地址：0x00

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
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	OPM	URS	UDIS	CEN
-	-	-	-	-	-	RW		RW	-		-	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31：10	保留	-	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，所用的采样时钟之间的分频比例

Bit	Name	R/W	Reset Value	Function
				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： TIM14_ARR 寄存器没有缓冲 1： TIM14_ARR 寄存器被装入缓冲器
6:4	保留	-	0	保留，一直读为 0
3	OPM	RW	0	单脉冲模式 (One pulse mode) 0：在发生更新事件时，计数器不停止； 1：在发生下一次更新事件(清除 CEN 位)时，计数器停止。
2	URS	RW	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 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 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. TIM14 中断使能寄存器 (TIM14_DIER)

偏移地址：0x0C

复位值：0x0000 0000

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

Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.														CC1IE	UIE
-														RW	RW

Bit	Name	R/W	Reset Value	Function
31:2	保留	-	-	保留
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

15.4.3. TIM14 状态寄存器(TIM14_SR)

偏移地址：0x010

复位值：0x0000 0000

31	**30**	**29**	**28**	**27**	**26**	**25**	**24**	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	IC1IF	Res	Res	Res	IC1IR
-	-	-	-	-	-	-	-	-	-	-	RC_W0	-	-	-	RC_W0
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res						**CC1OF**	**Res**							**CC1F**	**UIF**
-						RC_W0	-							RC_W0	RC_W0

Bit	Name	R/W	Reset Value	Function
31:21	保留	-	-	保留
20	IC1IF	RC_W0	0	下降沿捕获 1 标志 仅当相应的通道被配置为输入捕获时且由下降沿触发捕获事件，该标记可由硬件置 1。它由软件清‘0’或通过读 TIMx_CCR1 清‘0’。 0：无重复捕获产生； 1：发生下降沿捕获事件。
19:17	Res.	-	0	保留，一直为 0
16	IC1IR	RC_W0	0	上升沿捕获 1 标志 仅当相应的通道被配置为输入捕获时且由上升沿触发捕获事件，该标记可由硬件置 1。它由软件清‘0’或通过读 TIMx_CCR1 清‘0’。 0：无重复捕获产生； 1：发生上升沿捕获事件。
15:10	保留	-	-	保留
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：CC1IF 置 1 时，计数器的值已经被捕获到 TIM14_CCR1 寄存器。
8:2	保留	-	-	保留
1	CC1IF	Rc_w0	0	捕获/比较 1 中断标记

Bit	Name	R/W	Reset Value	Function
				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，它由软件清 0。 0：无匹配发生； 1：TIM14_CNT 的值与 TIM14_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 TIM14_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 TIM14_CCR1(在 IC1 上检测到与所选极性相同的边沿)。
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)；

15.4.4. TIM14 事件产生寄存器(TIM14_EGR)

偏移地址：0x14

复位值：0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31:2	保留	-	-	保留
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，则产生相应的中断请求。 **若通道 CC1 配置为输入：** 当前的计数器值捕获至 TIM14_CCR1 寄存器，设置 CC1IF=1，若开启对应的中断，则产生相应的中断请求。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，由硬件自动清 0。 0：无动作；

				1：重新初始化计数器，并产生一个寄存器的更新事件。注意预分频器的计数器也被清 0(但是预分频系数不变)。

15.4.5. TIM14 捕获/比较模式寄存器 1(TIM14_CCMR1)

偏移地址：0x18

复位值：0x0000 0000

输出比较模式:

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	OC1M[2:0]			OC1PE	Res	CC1S[1:0]	
-								-	RW	RW	-	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:7	保留	-	-	保留
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 TIMx_CNT 间的比较对 OC1REF 不起作 用; 001： 匹配时设置通道 1 为有效电平。当计数器 TIMx_CNT 的值与捕获/比较寄存器 1(TIMx_CCR1)相同时，强制 OC1REF 为高。 010： 匹配时设置通道 1 为无效电平。当计数器 TIMx_CNT 的值与捕获/比较寄存器 1(TIMx_CCR1)相同时，强制 OC1REF 为低。 011： 翻转。当 TIMx_CCR1=TIMx_CNT 时，翻转 OC1REF 的电平。 100：强制为无效电平。强制 OC1REF 为低。 101：强制为有效电平。强制 OC1REF 为高。 110： PWM 模式 1 - 在向上计数时，一旦 TIMx_CNT<TIMx_CCR1 时通道 1 为有效电平，否则为无效电平；在向下计数时，一旦 TIMx_CNT>TIMx_CCR1 时通道 1 为无效电平(OC1REF=0)，否则为有效电平(OC1REF=1)。 111： PWM 模式 2 一旦 TIMx_CNT<TIMx_CCR1 时，通道 1 为无效电平，否则为有效电平。 注：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 TIM14_CCR1 寄存器的预装载功能，可随时写入 TIM14_CCR1 寄存器，且新值马上起作用。

Bit	Name	R/W	Reset Value	Function
				1：开启 TIM14_CCR1 寄存器的预装载功能，读写操作仅对预装载寄存器操作，TIM14_CCR1 的预装载值在更新事件到来时被载入当前寄存器中。
2	保留	-	-	保留
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：保留； 11：保留。 注： CC1S 仅在通道关闭时(TIM14_CCER 寄存器的 CC1E=0)才是可写的。

输入捕获模式:

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								IC1F[3:0]				IC1PSC[1:0]		CC1S[1:0]	
-								RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:8	保留	-	-	保留
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， 它记录到 N 个事件后会产生一个输出的跳变： 0000：无滤波器，以 fDTS 采样 1000：采样频率 fSAMPLING=fDTS/8，N=6 0001：采样频率 fSAMPLING=fCK_INT， N=2 1001：采样频率 fSAMPLING=fDTS/8， N=8 0010：采样频率 fSAMPLING=fCK_INT， N=4 1010：采样频率 fSAMPLING=fDTS/16， N=5 0011：采样频率 fSAMPLING=fCK_INT， N=8 1011：采样频率 fSAMPLING=fDTS/16， N=6 0100：采样频率 fSAMPLING=fDTS/2， N=6 1100：采样频率 fSAMPLING=fDTS/16， N=8 0101：采样频率 fSAMPLING=fDTS/2， N=8 1101：采样频率 fSAMPLING=fDTS/32， N=5 0110：采样频率 fSAMPLING=fDTS/4， N=6 1110：采样频率 fSAMPLING=fDTS/32， N=6 0111：采样频率 fSAMPLING=fDTS/4， N=8 1111：采样频率 fSAMPLING=fDTS/32， N=8
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；

Bit	Name	R/W	Reset Value	Function
				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； 10：保留 11：保留 注： CC1S 仅在通道关闭时(TIM14_CCER 寄存器的 CC1E=0)才是可写的。

15.4.6. TIM14 捕获/比较使能寄存器(TIM14_CCER)

偏移地址：0x20

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
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	CC1NP	Res	CC1P	CC1E
												RW		RW	RW

Bit	Name	R/W	Reset Value	Function
31:4	保留	-	-	保留
3	CC1NP	RW	0	输入/捕获 1 互补输出极性 CC1 通道配置成输出： CC1NP 必须保持 0. CC1 通道配置成输入： CC1NP 和 CC1P 联合使用来定义 TI1FP1 极性（参考 CC1P 描述）
2	保留	-	-	保留
1	CC1P	RW	0	输入/捕获 1 输出极性 CC1 通道配置为输出： 0：OC1 高电平有效 1：OC1 低电平有效 CC1 通道配置为输入： CC1NP/CC1P 两位选择是 TI1FP1 还是 TI2FP1 的极性信号作为捕获信号。 00：不反相/上升沿：捕获发生在 TixFP1 的上升沿(捕获，复位触发，外部时钟或触发模式)； 01：反相/下降沿：捕获发生在 TixFP1 的下降沿(捕获，复位触发，外部时钟或触发模式)； 10：保留，无效配置。 11：不反向，双边沿。
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

Bit	Name	R/W	Reset Value	Function
				1：开启 - OC1 信号输出到对应的输出引脚 **CC1 通道配置为输入：** 该位决定了计数器的值是否能捕获入 TIMx_CCR1 寄存器。 0：捕获禁止 1：捕获使能

CcxE 位	OCx output State
0	输出禁止（OCx=0,OCx_EN=0）
1	OCx=OCxREF+Polarity,OCx_EN=1

15.4.7. TIM14 计数器(TIM14_CNT)

偏移地址：0x24

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
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	保留	-	-	保留
15:0	CNT[15:0]	RW	0	计数器的值

15.4.8. TIM14 预分频器(TIM14_PSC)

偏移地址：0x28

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
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	保留	-	-	保留
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 f_{CK_PSC}/(PSC[15:0]+1)。 PSC 包含了当更新事件产生时装入当前预分频器寄存器的值；更新事件包括计数器 被 TIM_EGR 的 UG 位清 0 或被工作在复位模式的从控制器清 0。

15.4.9. TIM14 自动重装载寄存器 (TIM14_ARR)

偏移地址：0x2C

复位值：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	保留	-	-	保留
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。 详细参考 12.4.1：时基单元有关 ARR 的更新和动作。 当自动重装载的值为空时，计数器不工作。

15.4.10. TIM14 捕获/比较寄存器 1(TIM14_CCR1)

偏移地址：0x34

复位值：0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31:16	保留	-	-	保留
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 TIMx_CNT 比较的值，并且在 OC1 端口上输出信号。 若 CC1 通道配置为输入： CCR1 包含了由上一次输入捕获 1 事件（IC1）传输的计数器值。

15.4.11. TIM14 选项寄存器(TIMx_OR)

偏移地址：0x50

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res															
-															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res														TI1_RMP	
-														RW	RW

Bit	Name	R/W	Reset Value	Function
31:2	保留	-	-	保留
1:0	TI1_RMP	RW	0	定时器输入 1 重映射

Bit	Name	R/W	Reset Value	Function
				通过软件置位和清零。 00:TIM14 通道 1 连接到 GPIO,具体参考数据手册的复用功能。 01：TIM14 通道 1 连接到 RTCCLK. 10: TIM14 通道 1 连接到 HSE/32 时钟 11： TIM14 通道 1 连接到 MCU 时钟输出（MCO）.这个配置是通过 RCC_CFG 寄存器的 MCO[2:0]的设置来决定的。

16. 实时时钟(RTC)

16.1. 简介

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

RTC 模块和时钟配置系统(RCC_BDCR 寄存器)处于保护区域，即在系统复位后，RTC 的设置和时间维持不变。

系统复位后，对RTC的访问被禁止，这是为了防止对RTC的意外写操作。执行以下操作将使能RTC的访问：

- 设置寄存器 RCC_APBENR1 的 PWREN，使能电源和时钟
- 设置寄存器 PWR_CR 的 DBP 位，使能对 BDTR 寄存器和 RTC 的访问。

16.2. 主要特性

- 可编程的预分频系数：分频系数最高为 220
- 32 位的可编程计数器，可用于较长时间段的测量
- 2 个单独的时钟：用于 APB1 接口的 PCLK1 和 RTC 时钟(RTC 时钟的频率必须小于 PCLK1 时钟频率的四分之一以上)
- 可以选择以下三种 RTC 的时钟源：
  - HSE 时钟除以 128
  - LSI 振荡器时钟
  - LSE 振荡器时钟
- 2 个独立的复位类型：
  - APB1 接口由系统复位；
  - RTC 核心(预分频器、闹钟、计数器和分频器)只能由备份域复位
- 3 个专门的可屏蔽中断：
  - 闹钟中断，用来产生一个软件可编程的闹钟中断
  - 秒中断，用来产生一个可编程的周期性中断信号(最长可达 1 秒)
  - 溢出中断，指示内部可编程计数器溢出并回转为 0 的状态

16.3. RTC 功能描述

16.3.1. 寄存器复位

RTC_PRL，RTC_ALR，RTC_CNT 和 RTC_DIV 和 BKP_RTCCR 寄存器仅能通过备份域复位信号复位(上电复位或 BDCR rtcrst)。其他系统寄存器（CRH、CRL）由系统复位或 rtcapb_sw_rst 或电源复位进行异步复位。

16.3.2. 读 RTC 寄存器

RTC 核完全独立于 RTC APB1 接口。

软件通过 APB1 接口访问 RTC 的预分频值、计数器值和闹钟值。但是，相关的可读寄存器只在与 RTC 时钟的上升沿同步到 RTC APB1 时钟后的信号有效时更新。RTC 标志也是如此的。

这意味着，如果 APB1 接口曾经被关闭，而读操作又是在刚刚重新开启 APB1 之后，则在第一次的内部寄存器更新之前，从 APB1 上读出的 RTC 寄存器数值可能被破坏了(通常读到 0)。下述几种情况下能够发生这种情形：

发生系统复位或电源复位

系统刚从待机模式唤醒

系统刚从停机模式唤醒(这种情况计数值只是不会更新，因为 CPU 不工作，系统时钟停止，但 RTC 正常计数，计数值不会同步到 V_{DD} 区)

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

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

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

说明：

1. CPU可读的寄存器包括RTC_CR，RTC_CNT和RTC_DIV；
2. RTC_CR寄存器为RTC_PCLK域，CPU任何时候读能读到稳定值；
3. RTC_CNT和RTC_DIV来源于RTC_CLK域。在RTC工作后，RTC_DIV寄存器在每个RTC_CLK的上升沿更新；RTC_CNT和来源于RTC_CLK时钟域的标志位同样采用跟RTC_DIV寄存器同样的更新信号，虽然这样不是每次更新时RTC_CNT的值都改变；
4. RSF实现在RTC_PCLK域，在RTC_CLK同步到RTC_PCLK后的脉冲信号有效时置位；
5. RSF仅控制RTC_CNT和RTC_DIV的读取时机（硬件不会控制）

16.3.3. 配置 RTC 寄存器

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

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

配置过程：

1. 查询RTOFF位，直到RTOFF的值变为'1'；（表明前面配置已经完成）
2. 置CNF值为1，进入配置模式；（此时RTOFF位仍为1，保证CPU在写寄存器时RTOFF=1）
3. 对一个或多个RTC寄存器进行写操作；（RTOFF在此过程仍为1，RTC_CLK域寄存器此步骤操作写入buffer寄存器）
4. 清除CNF标志位，退出配置模式；（硬件检测到CNF清零后，开始执行上一步骤中的写寄存器操作，开始写入RTC_CLK域的寄存器；同时RTOFF被清零）
5. 查询RTOFF，直至RTOFF位变为'1'以确认写操作已经完成。（buffer寄存器写入RTC_CLK域后置位RTOFF）

注：仅当 CNF 标志位被清除时，写操作才能进行，这个过程至少需要 3 个 RTC_CLK 周期。(在清除 CNF 标志位后 3 个 RTC_CLK 不能重新启动配置，否则会出现配置错误（此时通过 RTOFF=0 控制）

注：

1. 在此过程中，CPU写寄存器时RTOFF=1；
2. CPU的写周期为从CNF=1到CNF=0，配置其他寄存器在这两个操作之间；
3. 先将CNF写1后将CNF清零，这个操作清零RTOFF；只写CNF=0或者写CNF=1后不清零不会清零RTOFF；
4. 先将CNF写1后将CNF清零，这个操作启动buffer寄存器写RTC_CLK域寄存器的过程；
5. RTOFF实现在RTC_PCLK域；

16.3.4. RTC 标志设置

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

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

在计数器的值到达闹钟寄存器的值加 1(RTC_ALR+1)之前的 RTC 时钟周期中，设置 RTC_Alarm 和 RTC 闹钟标志(ALRF)。

对 RTC 闹钟寄存器（RTC_ALR）的写操作必须使用下述过程之一与 RTC 秒标志同步：

1. 使用RTC闹钟中断，并在中断处理程序中修改RTC闹钟寄存器（RTC_ALR）和/或RTC计数器寄存器（RTC_CNT）。
2. 等待RTC控制寄存器中的SECF位被设置，再更改RTC闹钟寄存器（RTC_ALR）和/或RTC计数器寄存器（RTC_CNT）。

16.3.5. RTC 时序

RTC 秒和闹钟时序如下图所示：

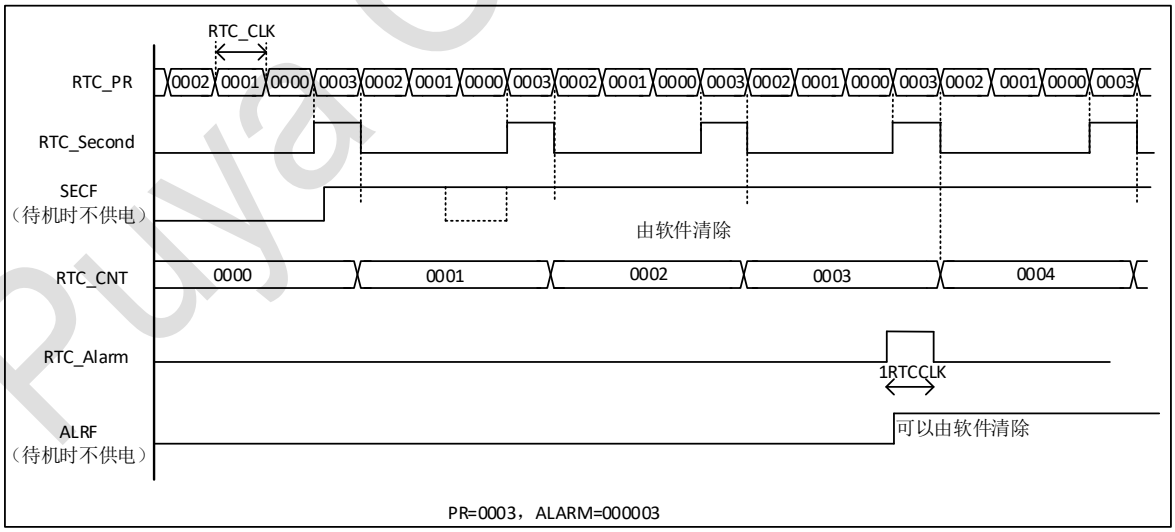


图 16-1 RTC 秒和闹钟时序图

SECF 和 ALRF 为 RTC_PCLK 域信号，分别采样 RTC_CLK 域 RTC_Sectond 和 RTC_Alarm 信号产生。

RTC 溢出时序如下图所示：

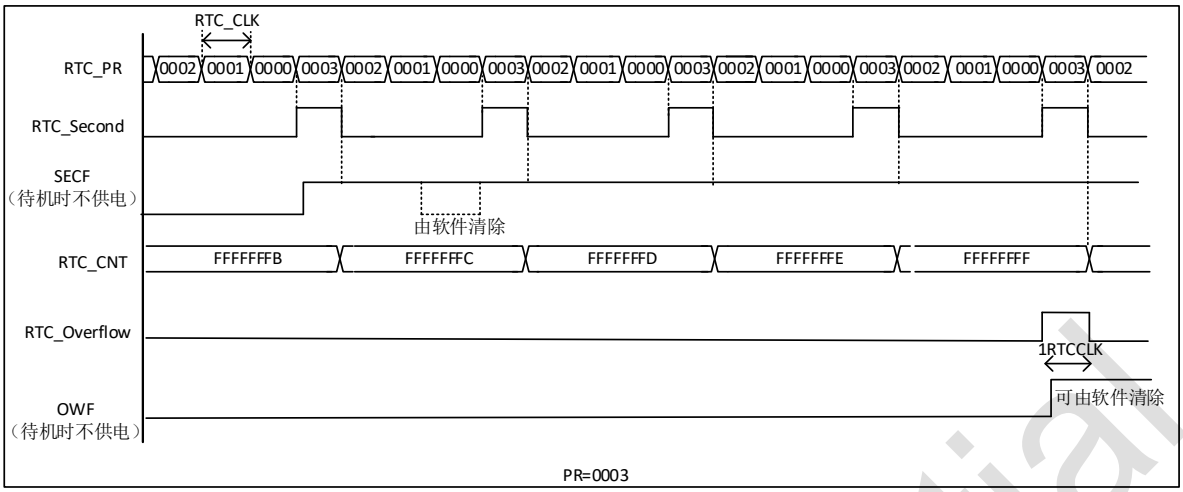


图 16-2 RTC 溢出时序图

SECF 和 OWF 为 RTC_PCLK 域信号，分别采样 RTC_CLK 域 RTC_Sectond 和 RTC_Overflow 号产生。

16.3.6. RTC 校准

为了测量目的，RTC 时钟的 64 分频可以输出到 IO 引脚上（PF5）。该功能是通过置位 CCO 位（RTCCR 寄存器）实现的。

通过配置 CAL[6:0] 位，时钟可以被减慢到 121 个 PPM。

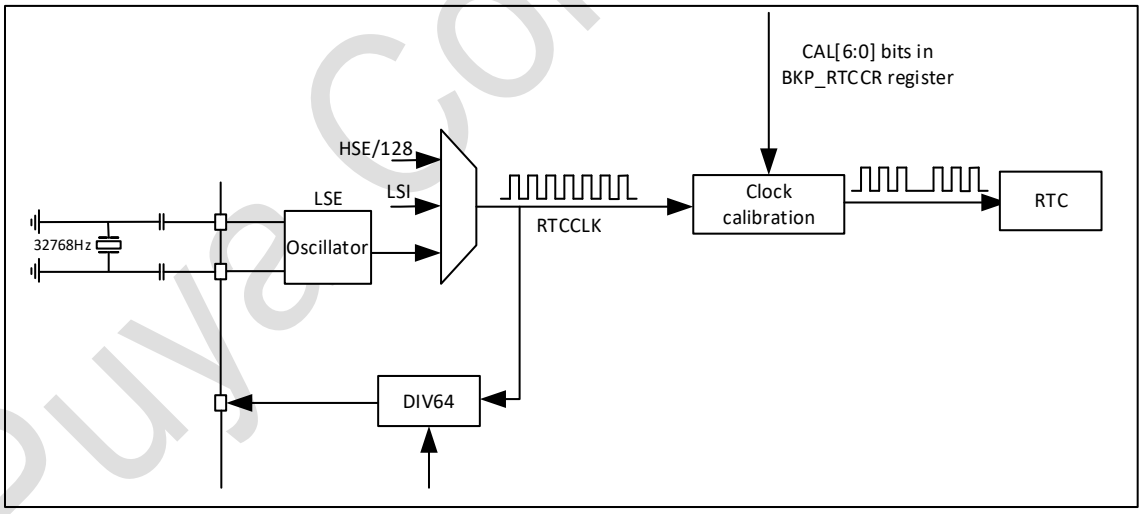


图 16-3 RTC 校准图

16.4. RTC 寄存器

16.4.1. RTC 控制寄存器 (RTC_CRH)

偏移地址：0x00

复位值：0x0000

该寄存器由系统复位复位。

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	TK_OWIE	TK_AL-RIE	TK_SECIE	OWIE	ALR IE	SEC IE
										RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:3	保留	-	-	保留
5	TK_OWIE	RW	0	TK 溢出中断允许位 0：不允许 TK 溢出中断 1：允许 TK 溢出中断
4	TK_ALRIE	RW	0	TK 闹钟中断允许位 0：不允许 TK 闹钟中断 1：允许 TK 闹钟中断
3	TK_SECIE	RW	0	TK 秒中断允许位 0：不允许 TK 秒中断 1：允许 TK 秒中断
2	OWIE	RW	0	溢出中断允许位 0：不允许溢出中断 1：允许溢出中断
1	ALRIE	RW	0	闹钟中断允许位 0：不允许闹钟中断 1：允许闹钟中断
0	SECIE	RW	0	秒中断允许位 0：不允许秒中断 1：允许秒中断

这些位用于屏蔽中断请求。注意：在复位后，所有中断是未使能的，所以在初始化后，写 RTC 寄存器以确保没有正在挂起的中断请求是可能的。但是当外设正在完成前一次的写操作（RTOFF=0）时，是不能写 RTC_CRH 寄存器的。

该控制寄存器控制着 RTC 的功能。某些位必须使用专门的配置流程才能进行写操作。

16.4.2. RTC 控制寄存器 (RTC_CRL)

偏移地址：0x04

复位值：0x0020

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	RTOFF	CNF	RSF	OWF	ALRF	SECF
										R	RW	RC_W0	RC_W0	RC_W0	RC_W0

Bit	Name	R/W	Reset Value	Function
31:6	保留	-	-	保留
5	RTOFF	R	1	RTC 操作关闭 (RTC operation OFF)，该位只读。 RTC 模块利用该位来指示对其寄存器进行的最后一次操作的状态（指示操作是否完成）。

Bit	Name	R/W	Reset Value	Function
				若此位为'0'，则表示无法对任何的 RTC 寄存器进行写操作。 0：上一次对 RTC 寄存器的写操作仍在进行 1：上一次对 RTC 寄存器的写操作已经完成
4	CNF	RW	0	配置标志 (Configuration flag) 此位必须由软件置'1'以进入配置模式，从而允许向 RTC_CNT、RTC_ALR 或 RTC_PRL 寄存器写入新值。只有当此位在被置'1'，并重新由软件清'0'后，才会执行写操作。参见《Configurating RTC registers》 0：退出配置模式(开始更新 RTC 寄存器) 1：进入配置模式
3	RSF	RC_W0	0	寄存器同步标志 (寄存器 s synchronized flag)当 RTC_CNT 寄存器和 RTC_DIV 寄存器更新时，硬件置'1'该位，软件清零该位。 在 APB1 复位后，或 APB1 时钟停止后，此位必须由软件清'0'。 要进行任何的读操作之前，用户程序必须等待该位被硬件置'1'，以确保 RTC_CNT、RTC_ALR 或 RTC_PRL 已经被同步。 0：寄存器尚未被同步 1：寄存器已经被同步
2	OWF	RC_W0	0	溢出标志 (Overflow flag) 当 32 位可编程计数器溢出时，此位由硬件置'1'。如果 RTC_CRH 寄存器中 OWIE=1，则产生中断。 此位只能由软件清'0'，写'1'无效。 0：无溢出； 1：32 位可编程计数器溢出
1	ALRF	RC_W0	0	闹钟标志 (Alarm flag) 当 32 位可编程计数器达到 RTC_ALR 寄存器所设置的预定值，此位由硬件置'1'。如果 RTC_CRH 寄存器中 ALRIE=1，则产生中断。此位只能由软件清'0'，写'1'无效。 0：无闹钟； 1：有闹钟。
0	SECF	RC_W0	0	秒标志 (Second flag) 当 32 位可编程预分频器溢出时，此位由硬件置'1'，同时 RTC 计数器加 1。 因此，此标志为分辨率可编程的 RTC 计数器提供一个周期性的信号(通常为 1 秒)。如果 RTC_CRH 寄存器中 SECIE=1，则产生中断。此位只能由软件清除，写'1'无效。 0：秒标志条件不成立 1：秒标志条件成立

RTC 的功能是被该控制寄存器控制的。当外设正在继续上一次写操作时（RTOFF=0），是不能写 RTC_CR 寄存器的。

注：

1. 任何标志位都将保持挂起状态，直到适当的RTC_CR请求位被软件复位，表示所请求的中断已经被接受。
2. 在复位时禁止所有中断，无挂起的中断请求，可以对RTC寄存器进行写操作。
3. 当APB1时钟不运行时， OWF、 ALRF、 SECF和RSF位不被更新。
4. OWF、 ALRF、 SECF和RSF位只能由硬件置位，由软件来清零。
5. 若ALRF=1且ALRIE=1，则允许产生RTC全局中断。如果在EXTI控制器中允许产生EXTI线 17中断，则允许产生RTC全局中断和RTC闹钟中断。
6. 若ALRF=1，如果在EXTI控制器中设置了EXTI线 17的中断模式，则允许产生RTC闹钟中断；如果在EXTI控制器中设置了EXTI线 17的事件模式，则这条线上会产生一个脉冲(不会产生RTC闹钟中断)。

16.4.3. RTC 重装载寄存器高位 (RTC_PRLH)

PRL 寄存器保持 RTC 预分频器周期性的计数值。该寄存器是被 RTC_CR 寄存器的 RTOFF 位写保护的，只有 RTOFF=1，才允许 CPU 进行写操作。

偏移地址：0x08

复位值：0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	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	PRL[19:16]			
												W	W	W	W

Bit	Name	R/W	Reset Value	Function
31：4	保留	-	-	保留
3：0	PRL[19:16]	W	0	RTC 预分频装载值高位 (RTC prescaler reload value high)根据以下公式，这些位用来定义计数器的时钟频率： fTR_CLK = fRTCCLK/(PRL[19:0]+1) 注：不推荐使用 0 值，否则无法正确的产生 RTC 中断和标志位

16.4.4. RTC 重装载寄存器低位 (RTC_PRLL)

偏移地址：0x0C

复位值：0x8000

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**
PRL[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	保留	-	-	保留

Bit	Name	R/W	Reset Value	Function
15：0	PRL	W	0x8000	RTC 预分频装载值高位 (RTC prescaler reload value high) 根据以下公式，这些位用来定义计数器的时钟频率： fTR_CLK = fRTCCLK/(PRL[19:0]+1) 注：不推荐使用 0 值，否则无法正确的产生 RTC 中断和标志位

16.4.5. RTC 预分频分频因子寄存器高位 (RTC_DIVH)

在每个 TR_CLK 周期，RTC_PRL 寄存器的值被重装载到 RTC 预分频计数器里。用户可以通过读取 RTC_DIV 寄存器，以获得预分频计数器的当前值，而不停止分频计数器的工作，从而获得精确的时间测量。

该寄存器只读属性，当 RTC_PRL 或者 RTC_CNT 寄存器的值发生任何变化，该寄存器值将由硬件重新装载。

偏移地址：0x10

复位值：0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
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	RTC_DIV[19:16]			
												R	R	R	R

Bit	Name	R/W	Reset Value	Function
31：4	保留	-	-	保留
3：0	RTC_DIV[19:16]	R	0	RTC 时钟分频器余数高位。

16.4.6. RTC 预分频分频因子寄存器低位 (RTC_DIVL)

偏移地址：0x14

复位值：0x8000

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[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	保留	-	-	保留
15：0	DIV[15:0]	R	0x8000	RTC 时钟分频器

16.4.7. RTC 计数寄存器高位 (RTC_CNTH)

RTC 模块有个 32 位可编程的计数器，该寄存器通过两个 16 位的寄存器访问，计数基于预分频器产生的 TR_CLK 时间基准为参考进行计数。

RTC_CNT 寄存器保持该计数器的计数值。寄存器是被写保护的，仅当 RTOFF=1 时才能进行写操作。对高 16 位的 RTC_CNTH 或者低 16 位的 RTC_CNTL 寄存器进行写操作，直接装载到相应的可编程计数器里，并重装载了 RTC 分频。当读操作发生，返回计数器的当前值（系统时间）。

偏移地址：0x18

复位值：0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RTC_CNT[31:16]															
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	保留	-	-	保留
15：0	RTC_CNT[31:16]	RW	0x0000	RTC 内核计数器的高 16 位 当读 RTC_CNTH 寄存器时，返回 RTC 计数器寄存器的当前值的高 16 位。只有进入配置模式才能对该寄存器进行写操作。

16.4.8. RTC 计数寄存器低位 (RTC_CNTL)

偏移地址：0x1C

复位值：0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31：16	保留	-	-	保留
15：0	RTC_CNT[15:0]	RW	0x0000	RTC 内核计数器低 16 位 当读 RTC_CNTL 寄存器时，返回 RTC 计数器寄存器当前值的低 16 位。只有进入配置模式才能对该寄存器进行写操作。

16.4.9. RTC 闹钟寄存器高位 (RTC_ALRH)

当可编程计数器（计数）达到存储在 RTC_ALR 寄存器的 32 位值时，并产生 alarm 中断请求。该寄存器是被 RTOFF 位写保护的，只有 RTOFF=1，才允许写访问。

偏移地址：0x20

复位值：0xFFFF

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
RTC_ALR[31:16]															
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	保留	-	-	保留
15：0	ALR[31:16]	RW	0xFFFF	RTC alarm 高 16 位

				软件可写 Alarm 时间的高 16 位。写该寄存器必须进入配置模式。

16.4.10. RTC 闹钟寄存器高位 (RTC_ALRL)

偏移地址：0x24

复位值：0xFFFF

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**
RTC_ALR[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
31：16	保留	-	-	保留
15：0	ALR[15:0]	RW	0xFFFF	RTC alarm 低 16 位 软件可写 Alarm 时间的低 16 位。写该寄存器必须进入配置模式。

16.4.11. RTC 时钟校准及输出配置寄存器(BKP_RTCCR)

Address offset：0x2C

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	ASOS	ASOE	CCO	CAL[6:0]						
						RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31：10	保留	-	-	保留
9	ASOS	RW	0	秒/闹钟脉冲输出选择位 当 ASOE 位被置位，ASOS 位可以被用作选择 Pin 上输出是 RTC second pulse 还是 Alarm pulse 信号 0：RTC Alarm pulse 信号 1：RTC second pulse 信号
8	ASOE	RW	0	秒/闹钟脉冲输出使能位 当置位该位，则由 ASOS 位决定 pin 上输出是 RTC second pulse 还是 Alarm pulse 信号。
7	CCO	RW	0	校准时钟输出 当 ASOE 没有置位时，可以置位 CCO 输出 RTC 时钟的 64 分频 0：无作用 1：当该位被置位，pin 上输出 RTC clock 的 64 分频。
6：0	CAL[6:0]	RW	0	校准值

Bit	Name	R/W	Reset Value	Function
				该值显示了每 2^{20} 个时钟可忽略的时钟脉冲个数，这允许 RTC 进行校准，以 1000000/2^{20}PPM 的 step 减慢时钟。 RTC clock 可以被减慢的从 0 到 121PPM。

写 RTCCR 寄存器是对其缓存寄存器的读写，需要通过 cnf 相关的配置流程将缓存寄存器的值加载到实际起作用的寄存器中，才能影响对应的结果。

17. 独立看门狗（IWDG）

17.1. 简介

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 可用来检测和解决由软件错误引起的故障，并在计数器达到指定的超时值时（TIMEOUT）触发系统复位。

独立看门狗(IWDG)由专用的低速时钟(LSI)驱动，即使主时钟发生故障它也仍然有效。

IWDG 最适合应用于那些需要看门狗作为一个在主程序之外，能够完全独立工作，并且对时间精度要求较低的场合。

17.2. IWDG 主要特性

- Free-running 向下计数器
- 时钟由独立的 RC 振荡器提供(可在 STOP 和 STANDBY 模式下工作)
- 支持 CPU 配置 IWDG 模块启动后，RCC 模块自动使能 LSI 作为 IWDG 时钟。
- 看门狗被激活后，则在计数器计数至 0x000 时产生复位

17.3. IWDG 功能描述

17.3.1. IWDG 框图

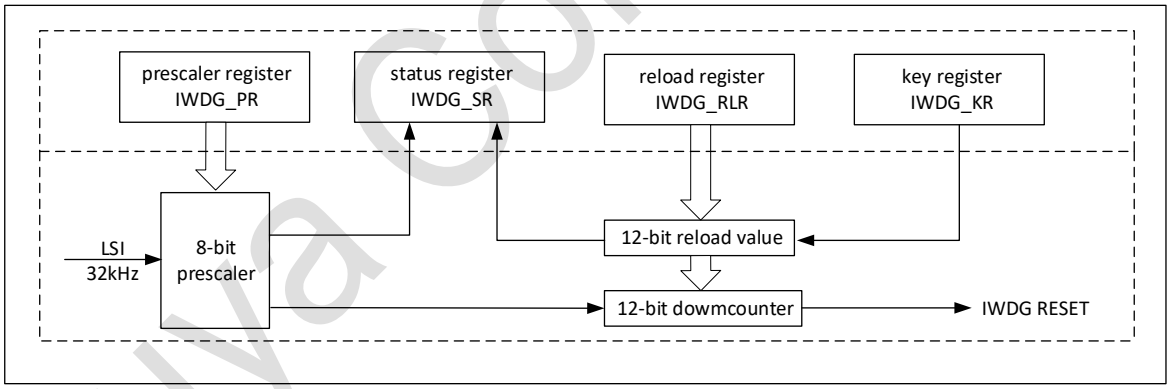


图 17-1 IWDG 框图

注： 看门狗功能处于 V_{DD} 供电区，即在停机和待机模式时仍能正常工作。

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

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

表 17-1 IWDG 超时时间 32.768 kHz 的输入时钟(LSI)

预分频	PR[2:0]	最小溢出值	最大溢出值	单位
/4	0	0.122	499.712	ms
/8	1	0.244	999.424	

/16	2	0.488	1998.848	
/32	3	0.976	3997.696	
/64	4	1.952	7995.392	
/128	5	3.904	15990.784	
/256	6 or 7	7.808	31981.568	

注：这些时间是按照 32.768 kHz 时钟给出。实际上， MCU 内部的 RC 频率会在 30 kHz 到 60kHz 之间变化。此外，即使 RC 振荡器的频率是精确的，确切的时序仍然依赖于 APB 接口时钟与 RC 振荡器时钟之间的相位差，因此总会有一个完整的 RC 周期是不确定的。通过对 LSI 进行校准可获得相对精确的看门狗超时时间。

17.3.2. 硬件看门狗

如果上电装载的 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 寄存器没有被软件改写，则产生复位信号

17.3.3. 硬件访问保护

对寄存器 IWDG Prescaler、IWDG Reload 和 IWDG Window 的写访问是被保护的。为了修改他们，用户必须先向 IWDG Key 寄存器写 0x0000 5555。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 Prescaler 寄存器、Reload 寄存器、Window 寄存器的值正在更新，状态寄存器是会体现出来的。

17.3.4. Debug 模式和 Stop 模式

调试模式功能为系统支持 DBG_MCU 时才存在。如果 CPU 进入调试模式，IWDG 继续正常计数还是进入 stop 模式，取决于 DBG 模块中 DBG_IWDG_STOP 的配置。

Stop 模式为在 option byte 中有 IWDG_STOP 位，可以控制住 CPU 在进入 DEEPSLEEP 模式时。IWDG 继续正常计数还是暂停计数，取决于 FMC 中的 option byte 里的 IWDG_STOP 的配置。

Option byte 中关于 IWDG_STOP 的配置如下：

设置 IWDG 在 Stop 模式下定时器运行状态

0：冻结定时器

1： 正常运行

Option byte 中默认 IWDG_STOP 为“ 1 ”

17.4. IWDG 寄存器

17.4.1. 密钥寄存器 (IWDG_KR)

偏移地址：0x00

复位值：0x0000 0000

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

W

Bit	Name	R/W	Reset Value	Function
31:16	保留	-	-	保留
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）。

17.4.2. 预分频寄存器 (IWDG_PR)

偏移地址：0x04

复位值：0x0000 0000

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

17.4.3. 重装载寄存器 (IWDG_RLR)

偏移地址：0x08

复位值：0x0000 0FFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	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	保留	-	-	保留
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。 此外需要注意，当向 RLR 寄存器写入重装载值后需要等待三个 LSI 时钟才能读出。

17.4.4. 状态寄存器 (IWDG_SR)

偏移地址：0x0C

复位值：0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31:2	保留	-	-	保留
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. I²C 接口

18.1. 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 总线特定的时序、协议、仲裁和定时。支持标准（Sm）、快速（Fm）、快速增强模式（Fm+）。

18.2. 主要特性

- 从机和主机模式
- 多主机功能：可以做主机，也可以做从机
- 支持不同通讯速度
  - 标准模式（Sm）：高达 100 kHz
  - 快速模式（Fm）：高达 400 kHz
  - 快速增强模式（Fm+）：1 MHz
- 作为主机
  - 时钟的产生
  - 起始和停止条件的产生
- 作为从机
  - 可编程的 I²C 地址检测
  - 停止条件的检测
- 7 位寻址模式
- 支持广播呼叫（General call）功能
- 状态标志位
  - 发送/接收模式标志位
  - 字节传输完成标志位
  - I²C 总线忙标志位
- 错误标志位
  - 主机仲裁丢失
  - 地址/数据传输后的 ACK 失败
  - 起始/停止条件错误
  - 过载（overrun）/欠载（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
- 支持地址匹配时 stop 唤醒
  - 支持可配置的唤醒 timeout 时间

— 支持可配置的 timeout 是否唤醒

18.3. I²C 功能描述

18.3.1. I²C 框图

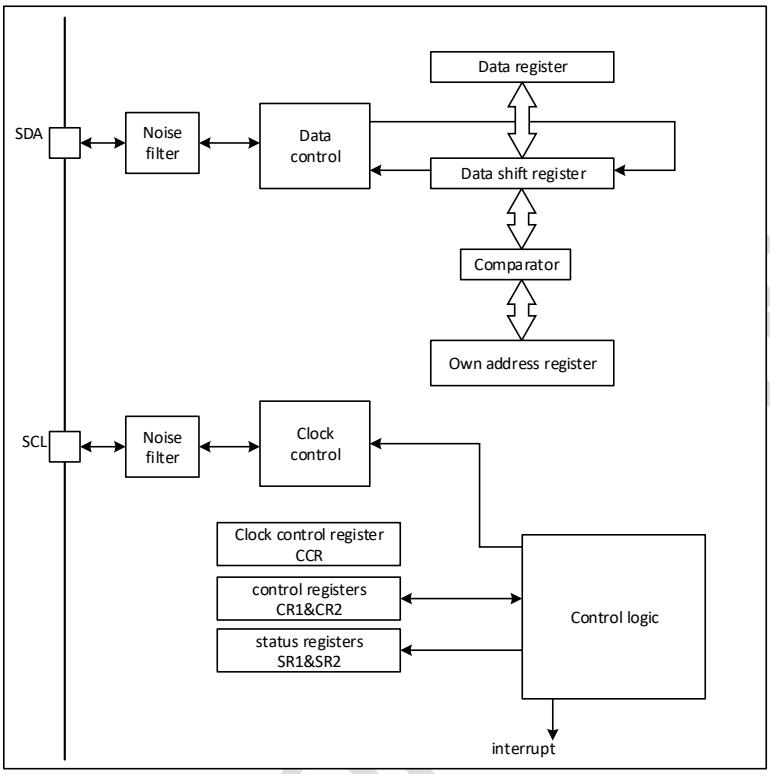


图 18-1 I2C 框图

18.3.2. 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。

18.3.2.1. 通信流

作为主机，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主机模式下由软件控制产生。

作为从机，I²C 接口能识别自己的地址(7 位)和广播呼叫地址。软件能够控制开启或禁止对广播呼叫地址的识别。

数据和地址按 8 位（字节）进行传输，高位在前。跟在起始条件后的 1 个字节是地址。地址只在主机模式发送。

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，见下图。

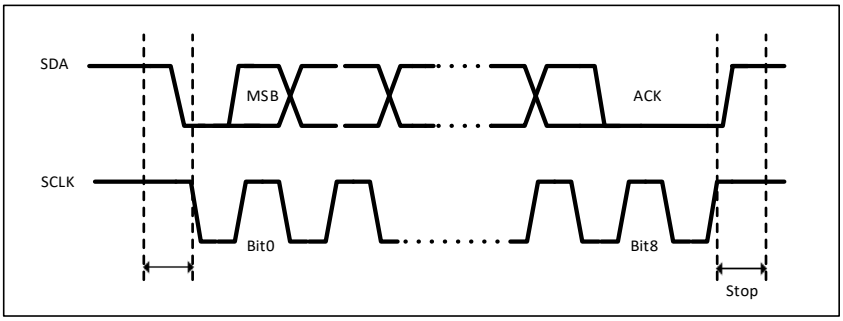


图 18-2 I2C 总线协议

软件可使能或禁用应答（ACK）位，也可以选择 I2C 接口地址（7 位地址或广播呼叫地址）。

18.3.3. I²C 初始化

18.3.3.1. 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。

18.3.3.2. I²C 时序设定

数据信号（SDA）的保持和建立时间要求满足 I²C 标准协议，需要对 I²C 时序的进行设定。这是通过写 I2C_CCR 和 I2C_TRISE 寄存器实现的。

18.3.4. I²C 从模式

默认情况下，I²C 接口总是工作在从机模式。从机模式切换到主机模式，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：4 MHz
- 快速模式下为：8 MHz
- 快速增强模式下为：16 MHz

一旦检测到起始条件，在 SDA 线上接收到的地址，被送到移位寄存器，并与芯片的地址 OAR1 或者广播呼叫地址(如果 ENGC=1)相比较。

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。

18.3.4.1. 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）从机将数据（字节）从 DR 寄存器，经由内部移位寄存器发送到 SDA 上。

从机拉低 SCL，直到 ADDR 位被清除，并且待发送数据已写入 DR 寄存器（参考 EV1、EV3）。

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。从机拉低 SCL，直到 BTF 位被软件清零（读 I2C_SR1 之后，再写入 I2C_DR 寄存器）。

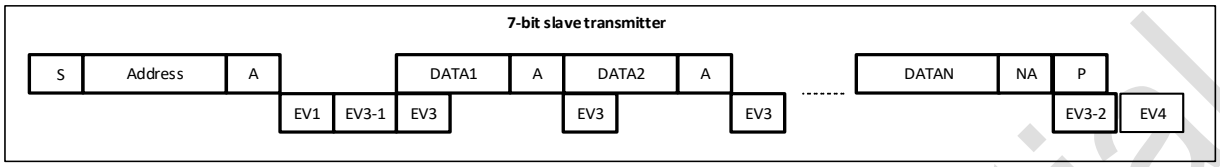


图 18-3 从发送器的传送序列图

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, 移位寄存器为空, 数据寄存器为空, 向 DR 寄存器写 Data1

EV3： TxE=1, 移位寄存器不为空, 数据寄存器为空，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

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。

注:

1.EV1 和 EV3_1 事件拉低 SCL，直到对应的软件序列结束。

2.EV3 的软件序列必须在当前字节传输结束之前完成

18.3.4.2. 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）从机将通过内部移位寄存器把从 SDA 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 之后再读入 I2C_DR 寄存器）之前，从机一直拉低 SCL。(见下图)。

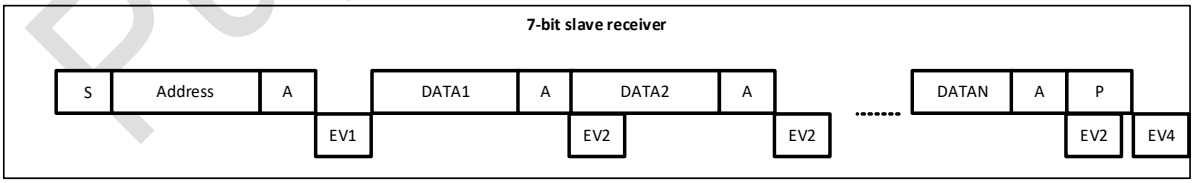


图 18-4 从接收器的传送序列图

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），EVx= Event(ITEVFEN= 1 时产生中断)

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。

注:

1. EV1事件拉低SCL，直到相应软件序列结束。
2. EV2软件序列必须在当前字节传输完成之前完成。
3. 当用户检查SR1寄存器内容后，应该对每个置位的标志位，进行完整的清除序列。比如ADDR和STOPF标志位，需要用以下序列：

如果 ADDR=1，先读 SR1，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 都被置位，都能被发现并且清除掉。

18.3.4.3. 关闭通信

在传输完最后一个数据字节后，主机产生一个停止条件，从机检测到该条件时：

- 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）

18.3.4.4. 低功耗唤醒

在芯片处于 stop 状态下，可以通过 I²C 接收地址，若地址匹配则可以唤醒芯片，反之则返回之前状态。

1）实现以上功能需要在进入 stop 之前使能 I2C_CR1 的 PE 和 WUPEN，并且在 RCC 中使能 I²C 的时钟。

2）Stop 唤醒过程还支持可配置的 timeout 时间，可通过设置 I2C_WT 进行配置，若使能 ITERREN，则在产生 timeout 之后唤醒芯片，若不使能 ITERREN，则在产生 timeout 之后返回之前状态。

18.3.5. I²C 主模式

在主机模式时，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。

- 以下是主机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 I2C_CCR 寄存器
- 配置 I2C_TRISE 寄存器
- 编程 I2C_CR1 寄存器中的 PE 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：4 MHz
- 快速模式下为：8 MHz
- 快速增强模式下为：16 MHz

18.3.5.1. 主机产生时钟

CCR 寄存器以上升沿计数，产生 SCL 的高电平和低电平。由于从机可能拉长 SCL 信号，在 SCL 上升沿产生后，主机在 TRISE 寄存器所设置的时间到达时，检查来自总线的 SCL 信号。

如果 SCL 是低电平，意味着从机正在拉长 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，即使从机不拉长 SCL，从 SCL 上升沿产生，到 SCL 上升沿被发现，这样的反馈回路也是要花费些时间的。这个回路的时间与 SCL 的上升时间有关系，再加上 SCL 输入路径的模拟噪声滤波延时，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。

18.3.5.2. 开始条件

当 BUSY=0 时，设置 START=1，I2C 接口将产生一个起始条件，并切换至主机模式(MSL 被置位)。

注：在 master 模式下设置 START 位，将在当前字节传输完后，由硬件产生一个重开始条件。

一旦发出起始条件：

- SB 位被硬件置位，如果设置了 ITEVTEN 位，则会产生一个中断。

主机读 SR1 寄存器，再把从机地址写入 DR 寄存器。

18.3.5.3. 从机地址发送

主机将从机的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。

随后主机读 SR1 寄存器，再读 SR2 寄存器。

根据送出从机地址的最低位，主机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。

18.3.5.4. 主机发送

在发送了地址和清除了 ADDR 位后，主设备主机通过内部移位寄存器将数据字节从 DR 寄存器发送到 SDA 线上。

主机等待，直到第一个数据字节被写入 DR 寄存器。

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，然后 I2C 接口将自动回到从模式(MSL 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。

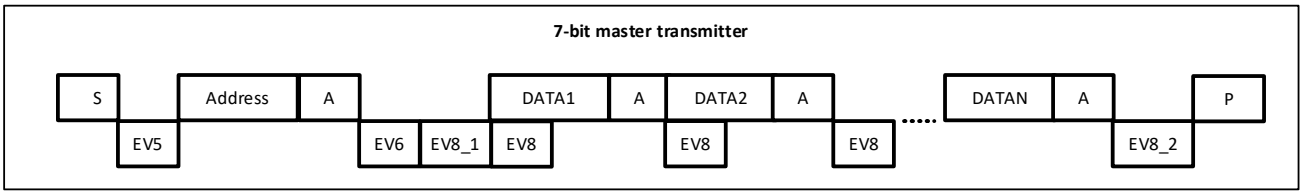


图 18-5 主发送器传送序列图

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），EVx= Event(ITEVFEN= 1 时产生中断)

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, 移位寄存器为空，数据寄存器为空，向 DR 寄存器写 Data1

EV8：TxE=1, 移位寄存器不为空，数据寄存器为空，向 DR 寄存器写 Data2，该位被清零

EV8_2：TxE=1, BTF=1, 写停止位寄存器，当硬件发出停止位时，TxE 和 BTF 被清零

注:

1）EV5, EV6, EV8_1 和 EV8_2 事件，拉长 SCL 的低电平，直到相应的软件序列执行结束

2）EV8 软件序列必须在当前字节发送完成前执行完毕。若 EV8 的软件序列不能在当前传输的字节结束前完成，则推荐使用 BTF 代替 TxE。

18.3.5.5. 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 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

方法 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

主机在从从机接收到最后一个字节后，发送一个 NACK。接收到 NACK 后，从机释放对 SCL 和 SDA 线的控制。从机就可以发送一个停止/重开始条件。

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)设置停止/开始位。
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)。
4. 在产生了停止条件后，I2C接口自动回到从模式(MSL位被清除)。

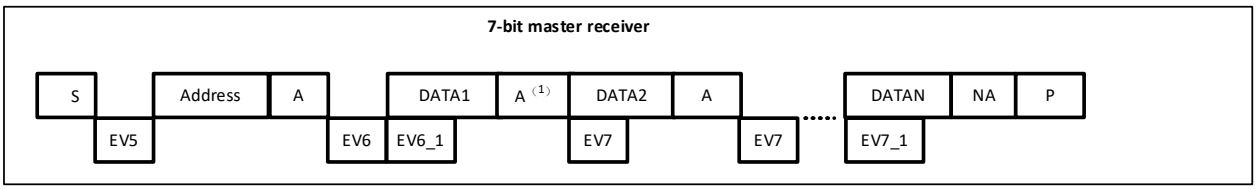


图 18-6 方法 1: 主模式发送时的时序

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），EVx= Event(ITEVFEN= 1 时产生中断)

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，直到相应的软件序列执行结束
3. EV7软件序列必须在当前字节发送完成前执行完毕。在EV7，软件序列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。
4. EV6_1或者EV7_1的软件序列必须在当前字节传输的ACK之前完成。

方法 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 ACK 位在 DataN ACK 之前被清掉。在此之后，在读 DataN-2 之后，置位停止/开始位，并读 DataN-1。在 RxNE 置位后，读 DataN。

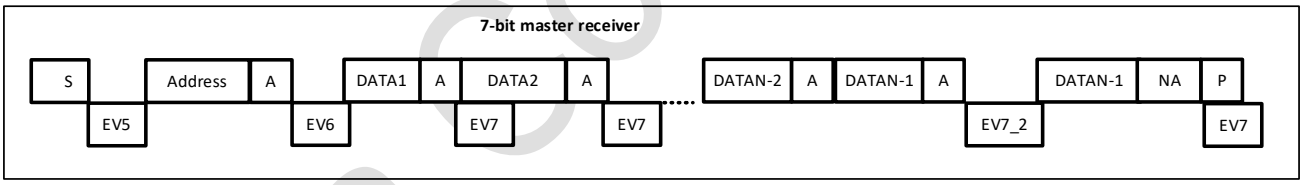


图 18-7 方法 2: N>2 时主模式发送时的时序

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），EVx= Event(ITEVFEN= 1 时产生中断)

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 存在移位寄存器中，写 ACK=0，读 DR 寄存器中的 DataN-2。置位 STOP，读 DataN-1

注:

1. EV5, EV6事件，拉长SCL的低电平，直到相应的软件序列执行结束
2. EV7软件序列必须在当前字节发送完成前执行完毕。在EV7，软件序列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。

当 3 个字节要被读走：

— RxNE=1 = > DataN-2 没有读。

- DataN-1 接收
- BTF=1, 移位和数据寄存器都是满的：DR 寄存器存放了 DataN-2，移位寄存器存放了 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 = > 这将启动移位寄存器对 DataN 的接收
- DataN 接收完成（发送 NACK 条件）
- 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

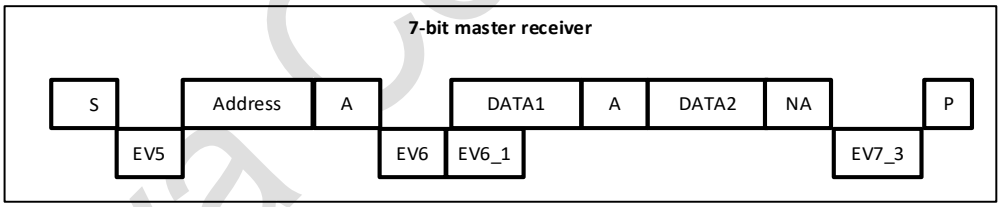


图 18-8 方法 2: N=2 时主模式发送时的时序

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）, EVx= Event(ITEVFEN= 1 时产生中断)

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）

注:

1. EV5, EV6 事件，拉长 SCL 的低电平，直到相应的软件序列执行结束
2. EV6_1 的软件序列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

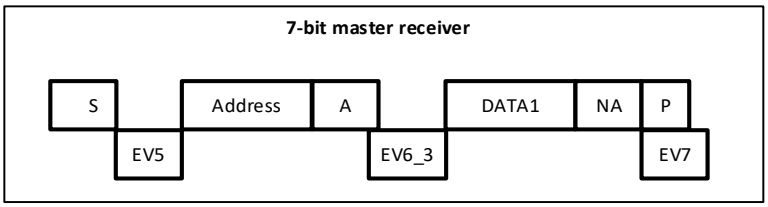


图 18-9 方法 2: N=1 时主模式发送时的时序

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）, EVx= Event(ITEVFEN= 1 时产生中断)

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

注:

EV5,EV6，EV8_1 和 EV8_2 事件会拉长 SCL 的低电平，直到相应的软件序列执行结束。

18.3.6. 错误状态

18.3.6.1. 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；

在从机模式：数据被丢弃，硬件释放总线：

- 如果是错误的起始条件，从机认为是一个重开始，并等待地址或停止条件
- 如果是错误的停止条件，从机按正常的停止条件操作，同时硬件释放总线

在主机模式：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。

18.3.6.2. 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：

- 如果是处于从机模式，硬件释放总线。
- 如果是处于主机模式，软件必须生成一个停止条件或者重开始。

18.3.6.3. 仲裁丢失 (ARLO)

当 I^2C 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。

I^2C 接口自动回到从模式(MSL 位被清除)。当 I^2C 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主机发送重开始条件之后响应硬件释放总线

18.3.6.4. 过载 overrun /欠载 underrun(OVR)

在从机模式下，如果禁止时钟延长，I²C接口正在接收数据时，当它已经接收到一个字节(RxNE=1)，但在 DR 寄存器中前一个字节数据还没有被读出，则发生过载错误。

此时：

最后接收的数据被丢弃在过载错误时，软件应清除 RxNE 位，发送器应该重新发送最后一次发送的字节

在从机模式下，如果禁止时钟延长，I2C 接口正在发送数据时，在下一个字节的时钟到达之前，新的数据还未写入 DR 寄存器(TxE=1)，则发生欠载错误。此时：

在 DR 寄存器中的前一个字节将被重复发出

用户应该确定在发生欠载运行错误时，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。

18.3.7. 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 和移位寄存器都是空的)。
- 接收器模式：如果 RxNE=1 且 BTF=1：I2C 接口在接收到数据字节后保持时钟线为低，以等待软件读 SR1，然后读数据寄存器 DR(DR 和移位寄存器都是满的)。
- 如果在从机模式中禁止时钟延长：
- 如果 RxNE=1，在接收到下个字节前 DR 还没有被读出，则发生过载运行。接收到的最后一个字节丢失。
- 如果 TxE=1，在必须发送下个字节之前却没有新数据写进 DR，则发生欠载运行。相同的字节将被重复发出。
- 硬件未实现对重复写冲突的控制。

18.4. I²C 中断

表 18-1 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	

中断事件	事件标志	开启控制位
过载/欠载	OVR	
超时/Tlow 错误	TIMEOUT	

18.5. I²C 寄存器

寄存器可以半字或者字访问。

18.5.1. I²C 控制寄存器 1 (I2C_CR1)

偏移地址：0x00

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWRST	Res.	Res.	Res.	POS	ACK	STOP	START	NOSTRETCH	ENGC	Res.	Res.	Res.	WUP.	Res.	PE
RW				RW	RW	RW	RW	RW	RW				RW		RW

Bit	Name	R/W	Reset Value	Function
31:16	保留	-	-	保留
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：该位可以用于错误或锁住状态时重新初始化 I²C。如 BUSY 位为 1，在总线上又没有检测到停止条件时。
14:12	保留	-	-	保留
11	POS	RW	0	ACK 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。 1：ACK 位控制在移位寄存器里接收的下一个字节的(N)ACK。 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。
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。

Bit	Name	R/W	Reset Value	Function
				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主机模式）
7	NOSTRETCH	RW	0	禁止时钟延长（从机）。 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
5:3	保留	-	-	保留
2	WUPEN	RW	0	从模式唤醒低功耗模式使能 0：禁止从模式唤醒低功耗模式 1：使能从模式唤醒低功耗模式 注： 如果不支持从模式唤醒低功耗模式功能，该位保留并由硬件强制为零。
1	保留	-	-	保留
0	PE	RW	0	I^2C 模块使能。 0：禁止 1：I^2C 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 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 位。

18.5.2. I^2C 控制寄存器 2 (I2C_CR2)

偏移地址：0x04

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	ITBUFEN	ITEVTEN	ITERREN	Res.	FREQ[6:0]						
					RW	RW	RW		RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:11	保留	-	-	保留
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： 1.SB=1（主模式） 2.ADDR=1（主/从模式） 3.STOPF=1（从模式） 4.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： 1.BERR=1 2.ARLO=1 3.AF=1 4.OVR=1 5.TIMEOUT=1；
7	保留	-	-	保留
6:0	FREQ	RW	0	I^2C 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 协议兼容的数据建立和保持时间。 最小允许可设定的频率是 4MHz，最大频率是芯片最高的 APB 时钟频率。 0000000：禁止 0000001：禁止 0000010：禁止 0000011：禁止 0000100：4MHz ….. 0100100：36MHz ….. 0110000：48MHz 大于 1001000：禁止。

18.5.3. I^2C 自身地址寄存器 1 (I2C_OAR1)

偏移地址：0x08

复位值：0x0000

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

Bit	Name	R/W	Reset Value	Function
31:8	保留	-	-	保留
7:1	ADD[7:1]	RW	0	接口地址的 7~1 位。
0	保留	-	-	保留

18.5.4. I²C 数据寄存器 (I2C_DR)

偏移地址：0x10

复位值：0x0000

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

Bit	Name	R/W	Reset Value	Function
31:8	保留	-	-	保留
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) 在从机模式下，地址不会被写进数据寄存器 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写到数据寄存器里，因此不能读到

18.5.5. I²C 状态寄存器(I2C_SR1)

偏移地址：0x14

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	TIMEOUT	Res.	Res	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
31:15	保留	-	-	保留
14	TIMEOUT	RC_W0	0	超时或 Tlow 错误。 0：无超时错误； 1：I²C 模式低功耗唤醒时可通过设置 I2C_WT 设置 timeout 时间。 该位由软件写 0 清除，或当 PE=0 时由硬件清除。
13:12	保留	RES	-	保留
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。

Bit	Name	R/W	Reset Value	Function
				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，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	保留	-	-	保留
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。
3	保留	-	-	保留
2	BTF	R	0	字节传输结束标志位。 0：字节传输发送未完成 1：字节传输发送成功结束 在下列情况下硬件将置位该寄存器（当从机模式，NOSTRETCH=0 时；主机模式，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）。

Bit	Name	R/W	Reset Value	Function
				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（从机）：** 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 寄存器或广播呼叫地址匹配，硬件将置位该位。 注: 在从机模式下，推荐进行完整的清零序列，即在 ADDR 被置位后，先读 SR1 寄存器，再读 SR2 寄存器。 **地址已发送（主机）：** 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。 注: 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。

18.5.6. I²C 状态寄存器 2 (I2C_SR2)

偏移地址：0x18

复位值：0x0000

注: 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

Bit	Name	R/W	Reset Value	Function
31:5	保留	-	-	保留
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	保留	-	-	保留
2	TRA	R	0	发送/接收标志。 0：接收到数据 1：数据已发送

Bit	Name	R/W	Reset Value	Function
				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：从模式 1：主模式 —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。

18.5.7. I²C 时钟控制寄存器(I2C_CCR)

偏移地址：0x1C

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
F/S	DUTY	F+	Res.	CCR[11:0]											
RW	RW	RW		RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:16	保留	-	-	保留
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	F+	RW	0	I2C F+主模式选择。 0：标准模式或者快速模式，具体选择哪种由 bit15 决定； 1：快速增强模式； 注：在配置该寄存器为 1 时，不关心 bit15 的值；
12	保留	-	-	保留
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： —Thigh=CCR x Tpclk —Tlow =CCR x Tpclk

Bit	Name	R/W	Reset Value	Function
				■ 快速模式: —DUTY=0: Thigh=CCR x Tpclk Tlow =2 x CCR x Tpclk —DUTY=1(为达到 400KHz): Thigh=9 x CCR x Tpclk Tlow =16 x CCR x Tpclk ■ 快速增强模式时: —DUTY=0: Thigh=3*CCR x Tpclk Tlow =5 x CCR x Tpclk —DUTY=1(为达到 1MHz): Thigh=2 x CCR x Tpclk Tlow =3 x CCR x Tpclk 注: ■ 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 ■ Thigh=tr(SCL)+tw(SCLH) ■ Tlow=tr(SCL)+tw(SCLL) ■ 这些延时没有过滤器 ■ 只有当 PE=0 时才能配置该寄存器; ■ f_{CK} 应当是 10 MHz 的整数倍，这样可以正确产生 400 kHz 的快速时候在哪个

注：bit15 和 bit13 结合起来扩展模式如下表所示：

F+	F/S	模式
0	0	标准模式
0	1	快速模式
1	0	快速增强模式
1	1	快速增强模式

18.5.8. I²C TRISE 寄存器 (I2C_TRISE)

偏移地址：0x20

复位值： 0x0082

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	THOLDDATA_SEL	THOLDDATA					TRISE[6:0]						
			RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:13	保留	-	-	保留
12	THOLDDATA_SEL	RW	0	数据保持时间选择 0：默认硬件计算数据保持时间 1：通过 THLDDATA 配置数据保持时间
11:7	THLDDATA	RW	1	在快速/标准/快速增强模式下的最大的数据保持时间(发送模式) 这些位是用于数据发送模式下，保证数据保持时间的最小保持时间。

Bit	Name	R/W	Reset Value	Function
				例如：标准模式允许的最大 SDA 下降时间位 300ns。如果在 I2C_CR2 寄存器种 FREQ[6:0]中的值等于 0x08，Tpclk =125ns，则 TRISE 中配置为 0x03（300ns/125ns = 2.4 +1 ） 如果结果不为整数，则将整数部分写入 THLDDATA，以确保配置。
6:0	TRISE	RW	0x2	在快速/标准模式下的最大上升时间（主模式）。 这些位应该提供在主机模式下， 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[6:0]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 tHIGH 参数。 注：当 PE=0 时才能设置该寄存器。

18.5.9. I²C Wakeup Time 寄存器 (I2C_WT)

偏移地址：0x24

复位值：0x0008

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	CNT		DIV	
												RW		RW	

Bit	Name	R/W	Reset Value	Function
15:4	保留	RES	-	保留
3:2	CNT	RW	0x2	timeout 的计数个数 00：2 01：8 10：32 11：128 注：timout 时间的计算公式为(DIV/FREQ)*CNT us 例：DIV 选择 11 即 1024，FREQ 为 8 M，CNT 选择 00 即 2，则 timeout 时间为(1024/8)*2=256us 注：PCLK 为 72 M 时，timeout 时间为 3.5us-1817.6us PCLK 为 4M 时，timeout 时间为 64us-32768us

1:0	DIV	RW	00	PCLK 的分频系数，分频后的时钟用于计数 stop 唤醒时的 timeout 时间 00：128 01：256 10：512 11：1024

19. 串行外接口 (SPI)

19.1. 简介

SPI 接口可以配置为支持 SPI 协议，SPI 接口默认工作在 SPI 方式，可以通过软件把功能从 SPI 模式切换到 I²S 模式。

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，还可使用 CRC 校验的可靠通信。

I²S 也是一种 3 引脚的同步串行接口通讯协议。它支持四种音频标准，包括飞利浦 I²S 标准， MSB 和 LSB 对齐标准，以及 PCM 标准。它在半双工通讯中，可以工作在主和从 2 种模式下。当它作为主设备时，通过接口向外部的从设备提供时钟信号。

19.2. 主要特性

- Master 或者 Slave 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 24 M）
- 从模式频率（最大为 24 M）
- 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
- 可引起中断的主模式故障、过载
- 2 个 32 位 Rx 和 Tx FIFOs

19.3. SPI 功能描述

19.3.1. 概述

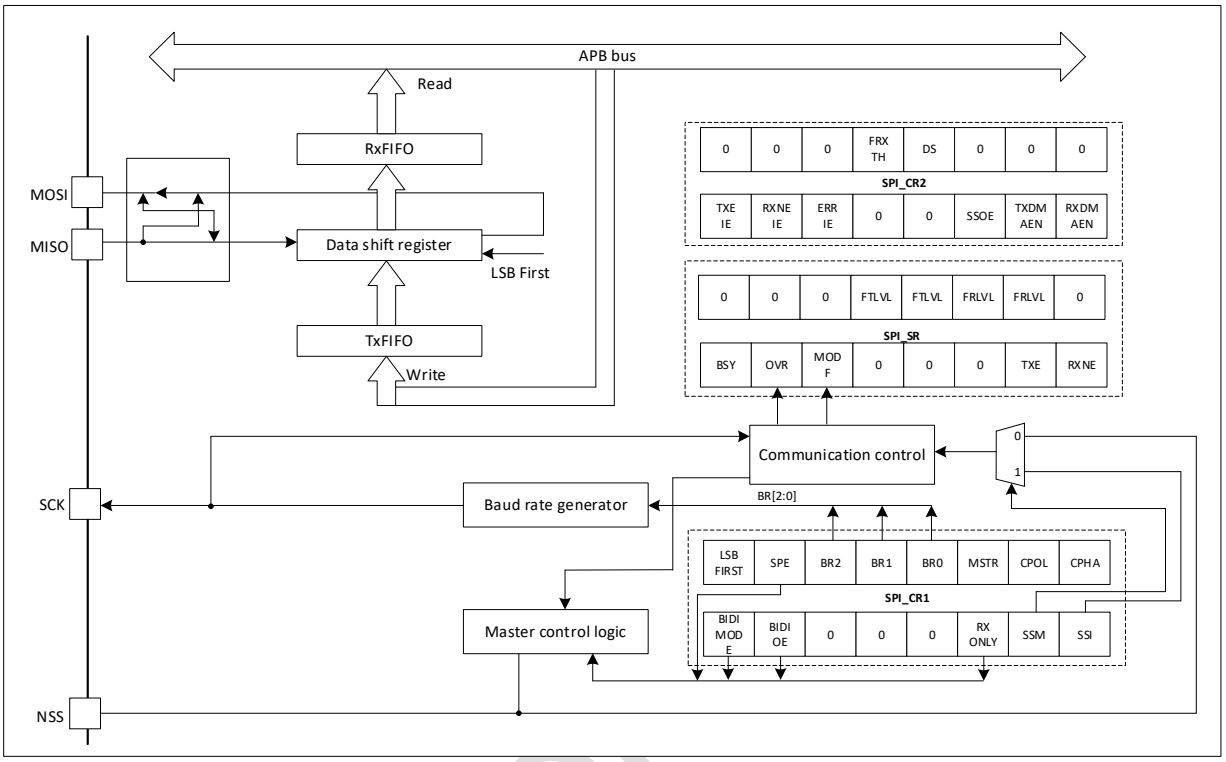


图 19-1 SPI 框图

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。

19.3.2. 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。

19.3.2.1. 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。

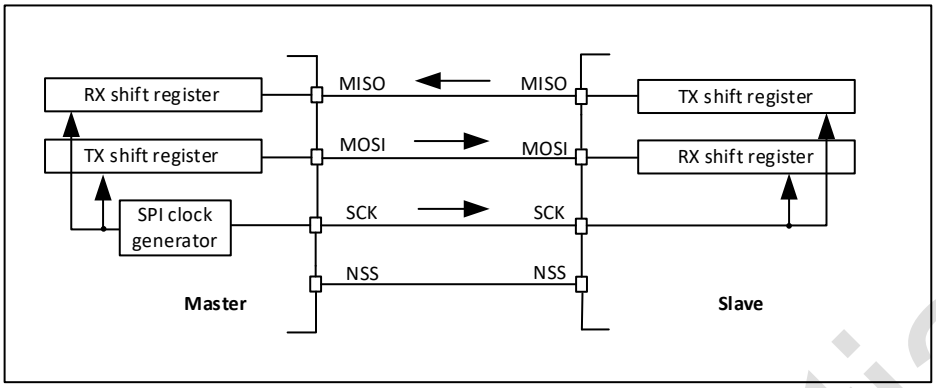


图 19-2 全双工单主机/单从机应用

19.3.2.2. 半双工通信

通过设定 BIDIMODE bit（SPI_CR1 寄存器），SPI 可以工作在 half-duplex 模式。在这种配置下，用 1 根数据线完成 master 和 slave 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，master 的 MISO pin 和 slave 的 MOSI pin 被释放作为 GPIO 给其他应用使用。

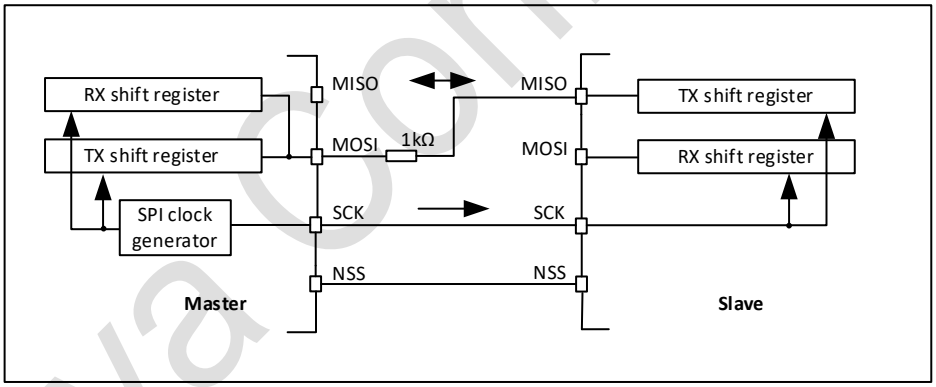


图 19-3 半双工单主机/单从机应用

NSS pin 可以被使用在 master 和 slave 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。

在该配置下，master 的 MISO pin 和 slave 的 MOSI pin 可以用作 GPIO。

19.3.2.3. 单工通信

通过使用 RXONLY（SPI_CR2 寄存器），设定 SPI 在 transmit-only 或者 receive-only，使 SPI 工作在 simplex 模式下。在这个配置下，在 master 和 slave 的 shift 寄存器之间只使用 1 根线。另一对 MISO 和 MOSI pin 不被使用，可以被释放成 GPIO。

- 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 disable SPI 输出功能。
  - 在 Slave 模式配置下，MISO 输出被 disable，该 pin 被用作 GPIO。当他的 slave select 信号有效时，Slave 继续从 MOSI pin 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 Master 模式配置下，MOSI 输出被 disable，pin 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 bit 或者 SPE bit，等待，直到来自 MISO pin 的输入 pattern 完成，并填入数据 buffer。

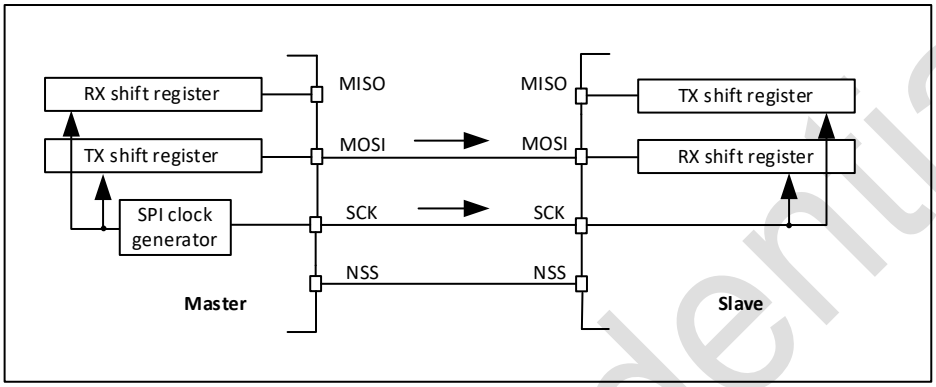


图 19-4 单工单从机/单主机应用

(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。

19.3.3. 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。

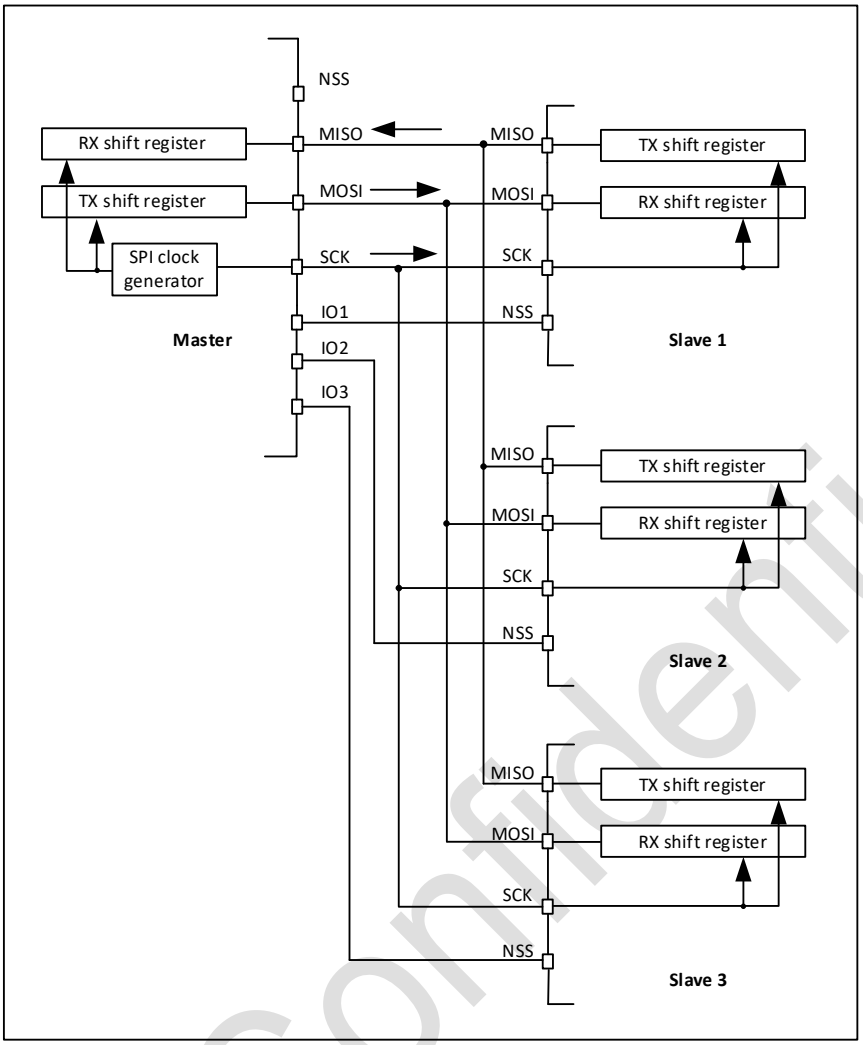


图 19-5 主机与三个独立的从机通信

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。

19.3.4. 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）

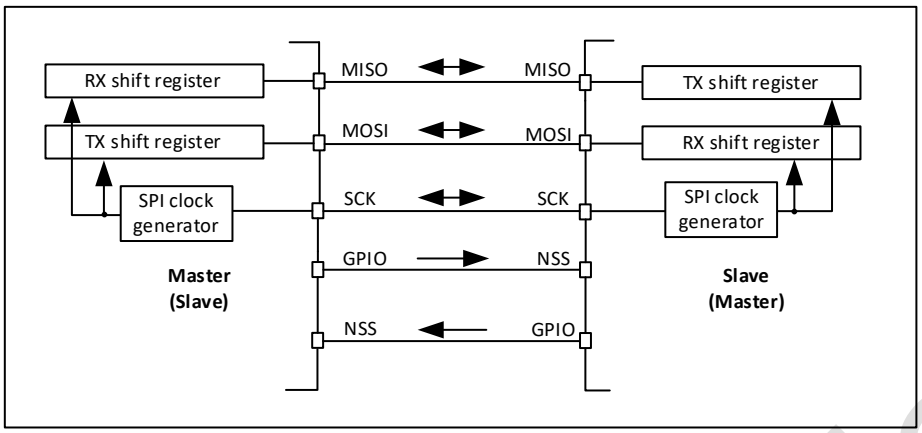


图 19-6 Multi-主机 application

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。

19.3.5. 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。
  - 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。
  - 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。

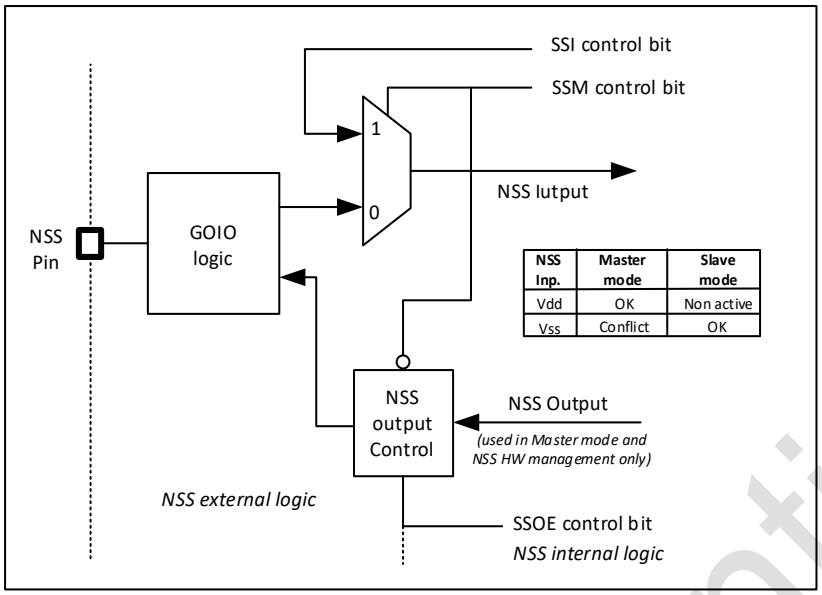


图 19-7 Hardware/software slave select management

19.3.6. 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。

19.3.6.1. 时钟相位和极性控制

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。

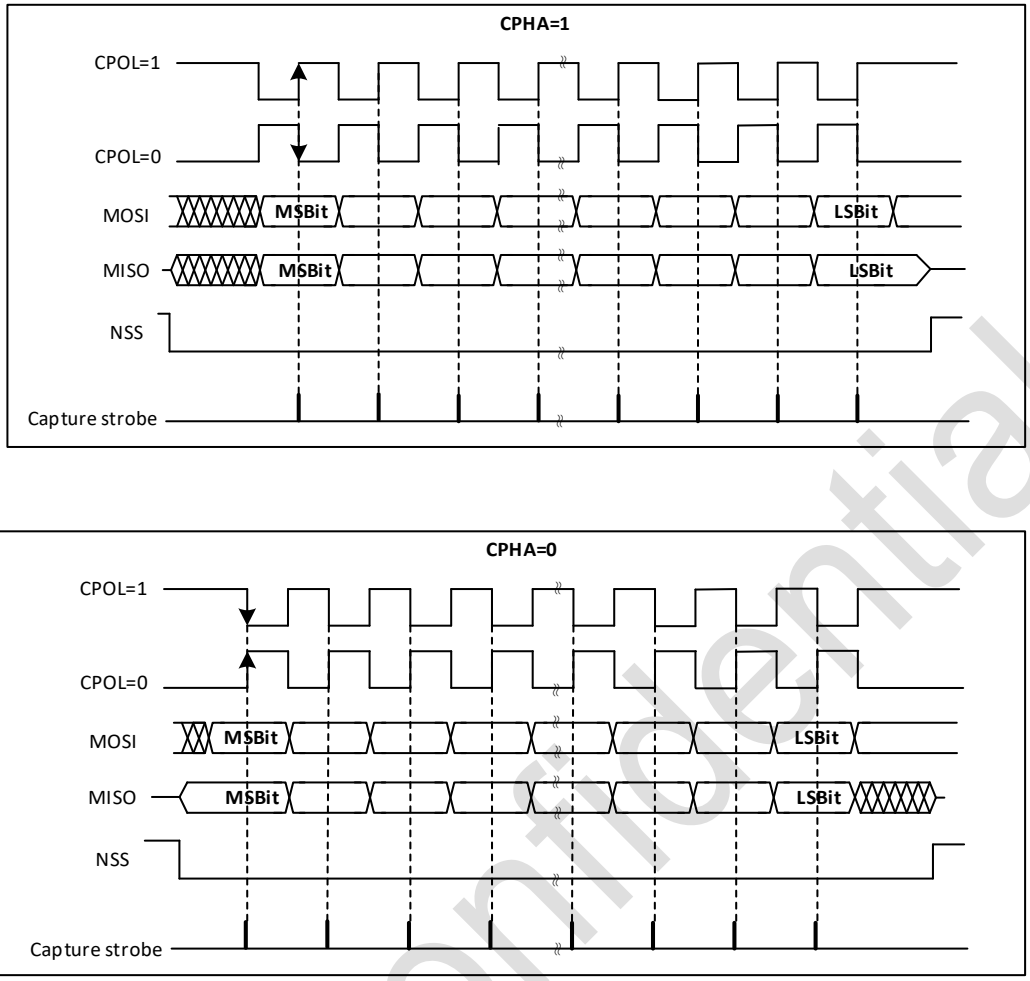


图 19-8 数据时钟时序图

数据 bit 的顺序取决于 LSBFIRST bit 的设定。

19.3.6.2. 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。

19.3.7. 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

19.3.8. 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。

19.3.9. 数据传输和接收流程

19.3.9.1. RXFIFO 和 TXFIFO

SPI 所有数据通讯都通过深度为 2，宽度为 16bit（当数据帧设置为 8bit 时，宽度为 8bit）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 size（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， FTLVL 和 FRLVL 位显示了两个 FIFO 当前的占用级别。

对 SPI_DR 寄存器的都访问必须通过 RXNE 事件管理。当数据存储器在 RXFIFO, 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 大于 1 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 2 个数据帧

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 master 端的两个数据帧之间，BSY 标志保持置位。但在 slave 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。

某些应用场景下，当向 TXFIFO 中写入数据，可以通过设置 CLRTXFIFO 位，清空 TXFIFO 数据，从而重新往 TXFIFO 里写新的数据重新进行通信。

19.3.9.2. 序列处理

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，且 maser 的 TXFIFO 里有任何数据，sequence 开始并继续进行。时钟信号被 master 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 被使能和 receive-only 模式被激活，master 就立即开始发送。Master 一直会提供时钟，直到 master 停止了 SPI 或者 receive-only 模式。Master 连续的接收数据。

当 master 能够以连续的模式（SCK 信号是连续的），提供所有通讯，master 必须要考虑 slave 处理数据流的能力。当有必要时，master 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 master 或者 slave 来说，没有 underflow 错误信号，来自于 slave 的数据通常被 master 交互和处理（即使 slave 不能及时准备好数据）。

每个 sequence 都必须被 NSS 脉冲包住，同时在多 slave 系统中选择要进行通讯的其中的一个 slave。在一个单 slave 系统，没有必要用 NSS 去控制 slave。

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 bit 被采样，并且整个数据帧被存在 RXFIFO 中。

19.3.9.3. 禁用 SPI 的步骤

当 SPI 被 disable 掉，必须按照特定的 disable 流程。对于系统 disable 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（disable），正在进行的交互会被破坏。在一些模式下，disable 流程是唯一停止连续通讯的办法。

全双工或者 transmit-only 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 packing mode（当交互奇数个数的数据帧，以放置一些 dummy 字节交互）。在这些模式下，SPI 被 disable 之前，用户必须使用标准的 disable 流程。

当 SPI 被 disable 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 receive-only 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 disable 流程。

当 SPI 被 disable，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 disable 时，RXFIFO 是空的（使用正确的 disable 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 disable 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 disable 流程是（receive-only 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 receive-only 模式，正确的 disable 流程是：

1. 在最后一个数据帧传输过程中，通过disable 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）

19.3.9.4. 数据压缩

当 frame size= 8, 任何 16-bit 的读或者写访问时，都会自动使用 data packing。在这种情况下，双数据 frame 会被并行处理。首先，SPI 使用存储在被访问 word 低位的 pattern，然后是存在高位的。

下图提供了 data packing 处理过程。在发送方的单个 16-bit 访问后，两个数据 frame 被发送。在接收方，如果 RXFIFO 阈值被是 16 bits（FRXTH=0），则该 sequence 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 frame。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 sequence 的最后一个数据 frame 是足够的。为了产生 RXNE 事件，对于奇数个数据 frame，对于接收的最后一个数据帧，接收方必须改变 Rx_FIFO 阈值。

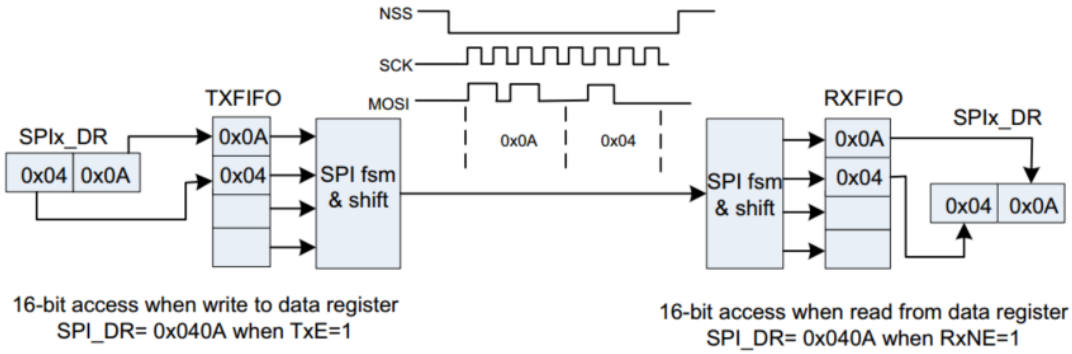


图 19-9 Packing data in FIFO for transmission and reception

19.3.10. 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。

19.3.10.1. 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。

19.3.10.2. 接收缓冲非空（Rx buffer not empty）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。

19.3.10.3. 忙（Busy）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 disable 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

注: 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。

19.3.11. 错误标志

19.3.11.1. 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。

19.3.11.2. 过载模式

当数据被主机或者从机接收，并且 RXFIFO 没有足够的空间存接收到的数据时，产生 over 正常运行情况。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 over 正常运行情况发生，新收到的数据不会 overwrite 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。

19.3.12. SPI 中断

表 19-1 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

19.3.13. 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。

19.3.13.1. SPI 控制寄存器 1 (SPI_CR1)

偏移地址：0x00

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BI-DIMODE	BIDIOE	Res	Res	DFF	RXONLY	SSM	SSI	LSBFIRST	SPE	BR[2:0]			MSTR	CPOL	CPHA
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。
13:12	保留	-	-	保留
11	DFF	RW	0	数据帧格式 0：使用 8 位数据帧格式进行发送/接收； 1：使用 16 位数据帧格式进行发送/接收。 注：只有当SPI禁止(SPE=0)时，才能写该位，否则出错。
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）

Bit	Name	R/W	Reset Value	Function
				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。 注：当通信在进行时不能改变该位的值，若软件在通信进行时修改通信将发生错误。
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：$f_{PCLK}/256$ 注：当通信在进行时不能改变该位的值，若软件在通信进行时修改通信将发生错误。
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。 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

Bit	Name	R/W	Reset Value	Function
				通讯进行时不能改变该寄存器的值。

19.3.13.2. SPI 控制寄存器 2 (SPI_CR2)

偏移地址：0x04

复位值：0x0700

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	Res	Res	Res	Res	Res	TXEIE	RXNEIE	ERRIE	CLRTXFIFO	Res	SSOE	Res	Res
								RW	RW	RW	RW		RW		

Bit	Name	R/W	Reset Value	Function
31:8	保留	-	-	保留
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	CLRTXFIFO	RW	0	清空 TXFIFO 软件置位，硬件复位 0：没作用 1：清空 TXFIFO 注：只有当SPI 禁止(SPE=0)时，才能写该位，否则无效。
3	保留	-	-	保留
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	保留	-	-	保留

注:

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：
3. 如果通讯的数据帧数为 1 帧数据，则需要将该位置为 1
4. 如果通讯的数据帧数为大于 1 帧数据，则将该位清为 0

19.3.13.3. SPI 状态寄存器 (SPI_SR)

偏移地址：0x08

复位值：0x0002

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	Res	FTLVL	Res	FRLVL	Res	BSY	OVR	MODF	Res	UDR	Res	TXE	RXNE

				R		R		R	R	R		R		R	R

Bit	Name	R/W	Reset Value	Function
31:12	保留	-	-	保留
11	FTLVL	R	0	FIFO 发送 level。 硬件置位，硬件清零 0：FIFO 空 1：FIFO 已满（FIFO 阈值大于 1/2 时视为 FULL）
10	保留	-	-	保留
9	FRLVL	R	0	FIFO 接收 level。硬件置位，硬件清零 0：FIFO 空 1：FIFO 满 注意：这些位在 I²S 模式和 带 CRC 校验的 SPI 仅接收模式下不使用。
8	保留	-	-	保留
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。 该位由硬件置位或者复位。
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	保留	-	-	保留
3	UDR	R	0	下溢标志位。 0：未发生下溢； 1：发生下溢错误。 硬件置位，软件序列清零。
2	保留	-	-	保留
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。 0：接收缓冲非空 1：接收缓冲为空

19.3.13.4. SPI 数据寄存器 (SPI_DR)

偏移地址：0x0C

复位值：0x0000

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

20. 通用异步收发器 (UART)

UART 是一种可编程通用异步收发器。该组件是一个符合 AMBA 2.0 的高级外围总线（APB）从设备。

20.1. UART 主要特性

- AMBA APB 接口
- 支持 5/6/7/8/9 位串行数据
- 支持 1/2 位 STOP 位（5 位数据时：1/1.5 位 STOP）
- 支持发送地址/数据
- 支持固定奇偶校验
- 支持 break 帧
- 起始位错误检测
- 支持可编程分数波特率：可编程串行数据波特率，计算如下：波特率=（串行时钟频率）/（16*除数）
- 支持 SWAP 功能
- 支持大小端切换 MSBFIRST 功能

20.2. 功能描述

20.2.1. UART（RS232）串行协议

因为 UART 和所选设备之间的串行通信是异步的，所以在串行数据中添加了额外的位（开始和停止）来指示开始和结束。利用这些比特可以使两个设备同步。这种由起始位和停止位组成的串行数据结构被称为一个字符，如下图所示。

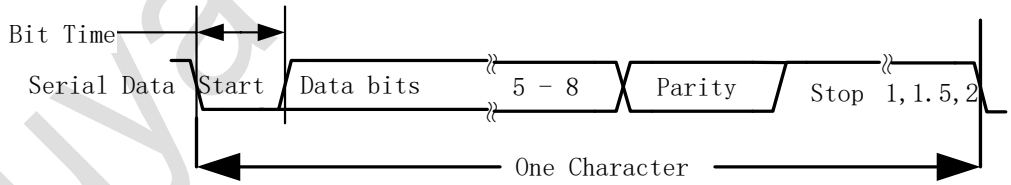


图 20-1 串行数据格式

一个额外的奇偶校验位可以添加到串行字符。该位出现在字符结构中的最后一个数据位之后和停止位之前，以便为 UART 提供对接收到的数据执行简单错误检查的能力。

UART 线路控制寄存器（ 寄存器说明”中的“CR1”部分）用于控制串行字符特性。数据字的各个位在起始位之后发送，从最低有效位（LSB）开始。它们后面是可选的奇偶校验位，后面是停止位，可以是 1、1.5 或 2。

注：UART 实现的 STOP 位持续时间可能会更长，原因如下：

- 在某些配置的字符之间插入的空闲时间

传输中的所有比特都是在完全相同的持续时间内传输的；这方面的例外是当使用 1.5 个停止位时的半停止位。该持续时间被称为比特周期或比特时间；一位时间等于十六个波特时钟。

为了确保线路的稳定性，一旦检测到起始位，接收器就在位时间的大约中点对串行输入数据进行采样。因为每个比特传输的波特时钟的确切数量是已知的，所以计算采样的中点并不困难；即在起始位的中点采样之后每十六个波特时钟。

与串行输入去抖动一起，这种采样有助于避免错误起始位的检测。短故障通过去抖动被过滤掉，并且在线路上没有检测到转换。如果故障足够宽，可以通过去抖动来避免滤波，则会检测到下降沿。然而，只有当线路在半个采样周期后再次被采样为低时，才检测到起始位

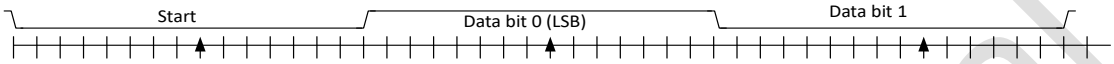


图 20-2 接收器串行数据采样点

作为 16550 标准的一部分，一个可选的波特时钟参考输出信号（baudout_n）为需要它的接收设备提供定时信息。UART 的波特率由单时钟实现中的串行时钟 sclk 或 pclk 以及分频锁存寄存器（BRR）控制。

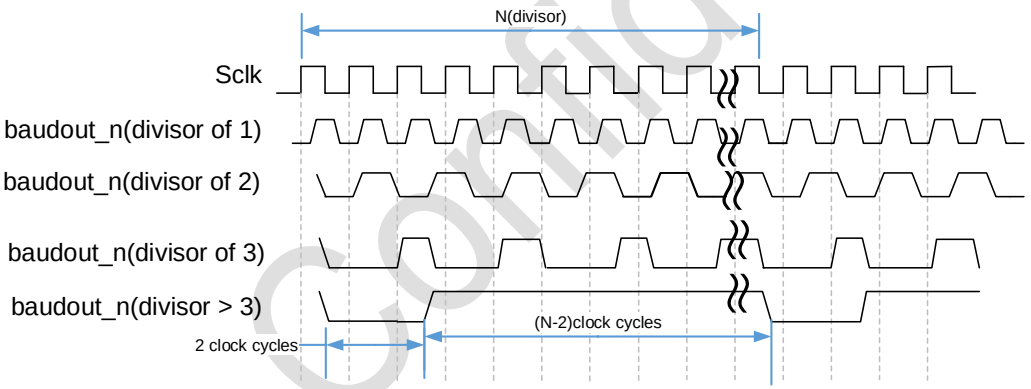


图 20-3不同除数值的baudout_n输出的时序图。

20.2.2. 9 位数据传输

UART 在发送和接收模式下都可以被配置为具有 9 位数据传输。字符中的第 9 位出现在字符的第 8 位之后和奇偶校验位之前。下图显示了字符的串行传输，其中 D8 表示第 9 位，还显示了 9 位模式下的一般串行传输。（此为正常小端模式，如果是大端模式则调转 9 位数据位的顺序）

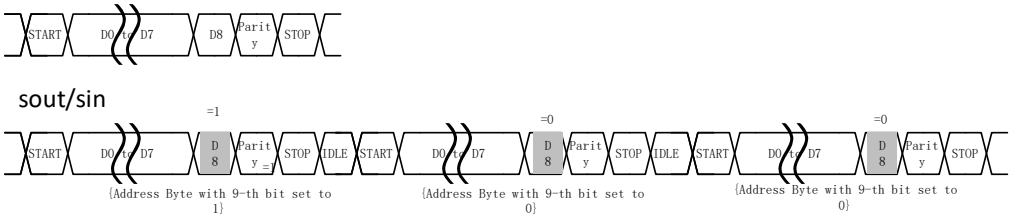


图 20-4 9-Bit 数据传输特性

通过启用 9 位数据传输模式，UART 可以用于多点系统，其中一个主设备连接到系统中的多个从设备。master 与其中一个 slave 交流。当主设备想要将数据块传输到从设备时，它首先发送一个地址字节来识别目标从设备。

地址/数据字节之间的区分是基于输入字符中的第 9 位来完成的。如果第 9 位设置为 0，则该字符表示一个数据字节。如果第 9 位设置为 1，则该字符表示地址字节。所有从系统将地址字节与它们自己的地址进行比较，并且只有目标从系统（其中地址匹配）能够从主系统接收数据。主机开始向目标从机发送数据字节。未寻址的从系统忽略传入的数据，直到接收到新的地址字节。

请注意一个地址后面跟着 2 个数据字节。地址字节在第 9 位（D8）设置为 1 的情况下输出，而数据字节在第九位（D8）设置为 0 的情况下发出。奇偶校验位是一个可选字段。

用于 9 位数据传输的 UART 的配置执行以下操作：

1. CR3[0]位用于启用或禁用9位数据传输。
2. 在接收的情况下，CR1[1]比特用于在基于硬件和软件的地址匹配之间进行选择。
3. CR3[2]位用于在发送的情况下使能发送地址。
4. CR3[3]位用于在基于硬件和软件的地址传输之间进行选择。
5. TAR和RAR寄存器分别用于发送地址和匹配接收到的地址。
6. TDR、RDR寄存器为9位寄存器，用于在9位模式下进行数据传输。
7. SR[8]位用于指示地址接收中断。

20.2.2.1. 发送模式

UART 支持两种类型的传输模式：

- 传输模式 0（当（CR3[3]）设置为 0 时）
- 传输模式 1（当（CR3[3]）设置为 1 时）

20.2.2.2. 传输模式 0

在传输模式 0 中，地址被编程在传输地址寄存器（TAR）中，数据被写入传输保持寄存器（TDR）。TDR 寄存器的第 9 位在此模式中不适用。

下图说明了基于 SEND_ADDR（CR3[2]）、TDR 空条件的地址和数据传输。

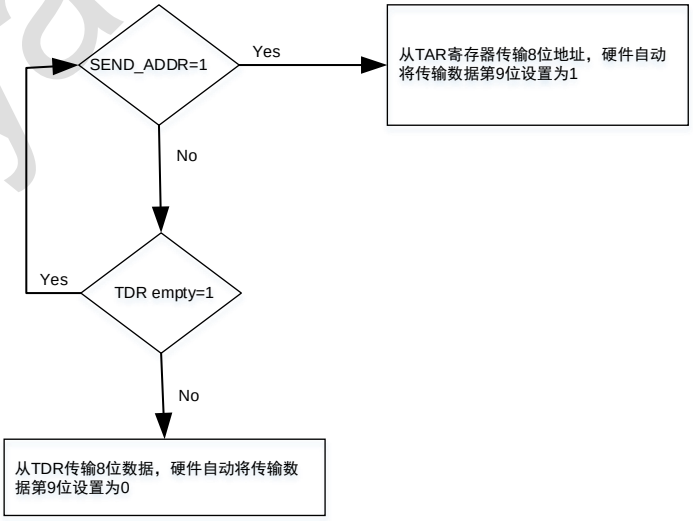


图 20-5 自动地址传输流程图

要向其传输数据的目标从机地址被编程在 TAR 寄存器中。必须启用 SEND_ADDR（CR3[2]）位来传输串行 UART 线上 TAR 寄存器中的目标从机地址，其中第 9 个数据位设置为 1，表示正在发送地址到从机。地址字符开始在 uart 线上传输后，UART 清除 SEND_ADDR 位。

传输到目标从机所需的数据通过传输保持寄存器（TDR）进行编程。数据在 UART 线上传输，第 9 个数据位设置为 0，表示正在发送数据到从机。

20.2.2.3. 传输模式 1

在传输模式 1 中，TDR 寄存器为 9 位宽，地址和数据都通过 TDR 寄存器编程。UART 不区分地址和数据。SEND_ADDR（CR3[2]）位和传输地址寄存器（TAR）不适用于此模式。根据是否必须发送地址/数据，软件必须用 1/0 写入第 9 位。

表 20-1 传输关系表

M_E	TX_MODE	SEND_ADDR	使用场景
0	X	X	发 8 位 TDR 发数据
1	0	0	发“0+8 位 TDR 数据“
1	0	1	发“1+8 位 TAR 地址”
1	1	X	发 “9 位 TDR 数据”

注：该表说明作为发送方可以使用的几种发送场景和对应的配置。

20.2.2.4. 接收模式

UART 支持两种接收模式：

- 硬件地址匹配接收模式（当 ADDR_Match（CR3[1]）设置为 1 时）
- 软件地址匹配接收模式（当 ADDR_Match（CR3[1]）设置为 0 时）

20.2.2.5. 硬件地址匹配接收模式

在硬件地址匹配接收模式中，如果接收字符的第 9 位设置为 1，则 UART 将接收字符与在接收地址寄存器（RAR）中编程的地址进行匹配：

如果接收到的地址与 RAR 寄存器中的编程地址匹配成功，则随后接收数据字节。

如果地址匹配失败，则 UART 控制器丢弃数据字符，直到接收到匹配的地址为止。

下图显示了基于地址匹配功能的数据字节接收流程图。

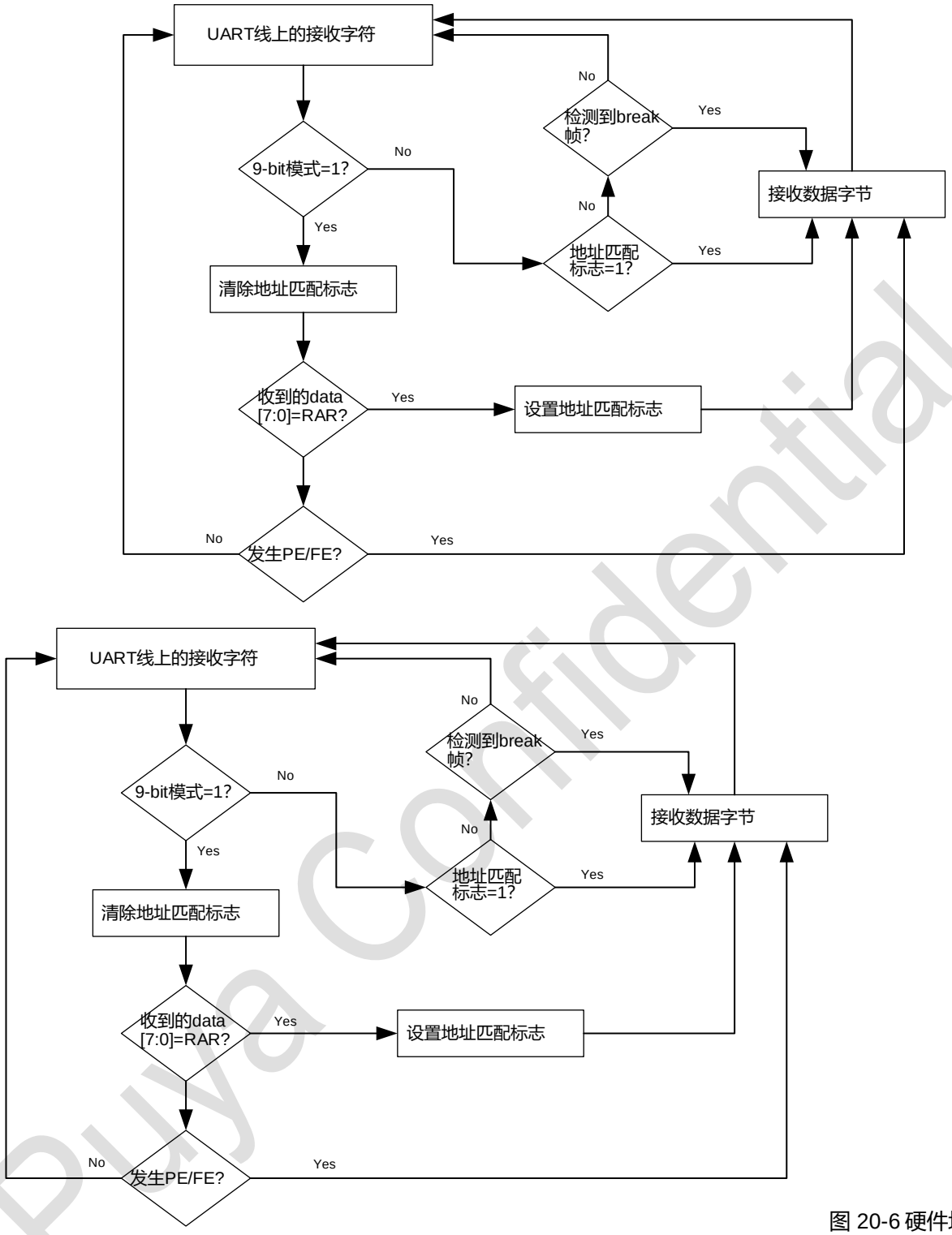


图 20-6 硬件地址匹配接收模式

UART 接收字符，而不管第 9 位数据是否设置为 1。如果接收到的字符的第 9 位被设置为 1，则它清除内部地址匹配标志，然后将接收到的 8 位字符信息与 RAR 寄存器中编程的地址进行比较。

如果接收到的地址字符与 RAR 寄存器中编程的地址匹配成功，则地址匹配标志被设置为 1，并且接收到的字符推送到 RDR 寄存器，并且 SR 寄存器中的 ADDR_RCVD 位被设置为指示地址已被接收，随后的数据字节（接收字符的第 9 位设置为 0）被推送到或 RDR。

如果地址与 RAR 寄存器匹配失败并且（奇偶校验或者接收的地址字符中发现帧错误）的情况下，则接收的地址字符串仍然被推送到 RDR 寄存器，ADDR_RCVD 和 PE/FE 错误位被设置为 1。

如果接收到任何中断字符，UART 将其视为一个特殊字符，并将其推送到 RDR 寄存器，而不考虑地址匹配标志。

20.2.2.6. 软件地址匹配接收模式

在这种操作模式中，UART 不执行与 RAR 寄存器的接收地址字符（第 9 位数据设置为 1）的地址匹配。UART 始终接收 9 位数据，推进 RDR 寄存器。每当接收到地址字节并通过线路状态寄存器中的 ADDR_RCVD 位指示时，必须由用户自行比较地址。

20.2.3. 波特率

UART 的波特率由 pclk 和分频锁存寄存器（BRR）控制。

波特率由以下因素决定：

- 串行时钟工作频率（pclk）
- 所需波特率
- 波特率生成器除数值 divisor（由 BRR 寄存器组成）
- 可接受的波特率误差%error

计算波特率的方程式如下：

波特率=串行时钟工作频率/（16 DIVISOR）　　——等式（1）*

其中，DIVISOR–用于对 BRR 进行编程的数字（十六进制）。

串行时钟频率–UART 的 sclk 或 pclk 引脚处的频率。

根据等式（1），DIVISOR 可以计算为：DIVISOR=串行时钟频率/（16*波特率）

同样从等式（1）中，还可以示出：

串行时钟频率= 波特率* 16* 除数

波特率和选定波特率定之间的误差如下所示：

百分比错误=（|波特率-选定波特率|）/波特率* 100%

表 20-2 设置波特率时的误差计算

波特率		f_{pclk} = 4 MHZ			f_{pclk} = 24 MHZ			f_{pclk} = 48 MHZ		
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%	实际	置于波特率寄存器中的值	误差%
1	2.4	2.404	104	0.16%	2.4	625	0.00%	2.4	1250	0.00%
2	9.6	9.615	26	0.16%	9.615	156	0.16%	9.615	312	0.16%
3	19.2	19.231	13	0.16%	19.231	78	0.16%	19.231	156	0.16%
4	57.6	62.5	4	8.51%	57.692	26	0.16%	57.692	52	0.16%
5	115.2	125	2	8.51%	115.385	13	0.16%	115.385	26	0.16%
6	230.4	250	1	8.51%	250	6	8.51%	230.769	13	0.16%
7	460.8	不可能	不可能	不可能	500	3	8.51%	500	6	8.51%
8	921.6	不可能	不可能	不可能	1500	1	62.76 %	1000	3	8.51%
9	2250	不可能	不可能	不可能	不可能	不可能	不可能	3000	1	33.33%
10	4500	不可能	不可能	不可能	不可能	不可能	不可能	不可能	不可能	不可能

注:

1. 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。

2.在配置时钟的时候，由于波特率时钟是在系统时钟上进行分频的，但得到的时钟频率与实际时钟频率会有误差。配置的时钟误差需要在 2%以内，才可以正常工作。

20.2.4. UART 接收器容忍时钟的变化

只有当整体的时钟系统地变化小于 UART 异步接收器能够容忍的范围， UART 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 < UART 接收器的容忍度

对于正常接收数据， UART 接收器的容忍度等于最大能容忍的变化为 96%-105%。

20.2.5. 中断

UART 中断输出信号（intr）的断言——只要几个优先中断类型中的一个被启用并激活，就会发生中断。

当发生中断时，主机可访问 SR 寄存器判定中断类型。

以下中断类型可以通过 CR2 寄存器启用：

- 接收器错误
- 接收器数据可用
- 发送保持寄存器为空（在可编程 TDR 中断模式下）
- 忙检测指示

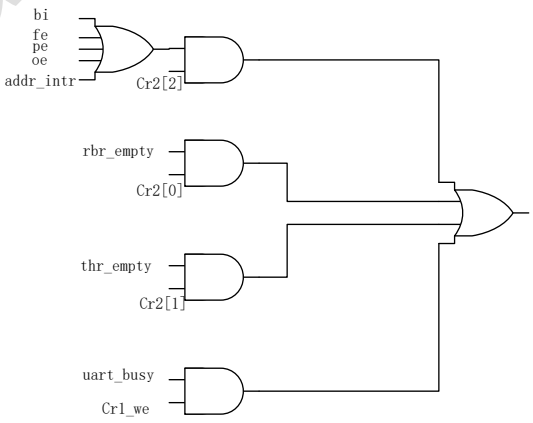


图 20-7 UART 中断映射图

这些中断类型在下表中有详细说明。

表 20-3 中断控制功能

Interrupt ID	Interrupt Set and Reset Functions

Bit 3	Bit 2	Bit 1	Bit 0	Priority Level	Interrupt Type	Interrupt Source	Interrupt Reset Control
0	0	0	1	–	None	None	–
0	1	1	0	Highest	Receiver line status	溢出/奇偶校验/帧错误、中断或地址接收中断	对应位写 1 清 0
0	1	0	0	Second	Received data available	接收器数据可用	读取接收器缓冲寄存器
0	0	1	0	Third	Transmit holding register empty	发送保持寄存器为空	写入 TDR
0	1	1	1	Fifth	Busy detect indication	UART 繁忙时（BUSY[0] 设置为 1），master 尝试写入 CR1 寄存器。	对应位写 1 清 0

20.3. UART 寄存器

20.3.1. 状态寄存器 (UART_DR)

偏移地址：0x00

复位值： 0x00000000

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

Bit	Name	R/W	Reset Value	Function
31:9	保留	-	-	保留
8:0	DR	RW	9'b0	接收/发送数据寄存器。一共是由两个寄存器组成（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。 RDR 寄存器是 UART 模式下串行输入端口（sin）上接收到的数据字节。只有状态寄存器（SR）中的接收非空（RXNE）位被设置时，该寄存器中的数据才有效。 必须在下一个数据到达之前读取 RDR 中的数据，否则它将被覆盖，从而导致 over-run 错误。 TDR 寄存器是在 UART 模式下串行输出端口（sout）上传输的数据。 软件保证只有当 TDR 空（TDRE）位（SR[5]）被设置时，数据才能再次写入 TDR。 如果 TDRE 被设置，则向 TDR 写入单个字符将清除 TDRE。在再次设置 TDRE 之前对 TDR 的任何额外写入都会导致 TDR 数据被重写。 仅当 CR3[3] = 1 时，第 9 位才适用。

20.3.2. 波特率寄存器 (UART_BRR)

偏移地址：0x04

复位值：0x0000_0000

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

Bit	Name	R/W	Reset Value	Function
31:16	保留	-	-	保留
15:0	BRR	RW	16'b0	该寄存器构成 16 位 divisor，divisor 包含 UART 的波特率除数。 输出波特率等于串行时钟（pclk）频率除以波特率除数值的十六倍，如下所示：波特率=（串行时钟频率）/（16*divisor）。 注：当除数锁存寄存器（BRR）设置为零时，波特时钟被禁用，不会发生串行通信。 此外，一旦设置了 BRR，在发送或接收数据之前，应等待至少 8 个时钟周期通过。

20.3.3. 状态寄存器 (UART_SR)

偏移地址：0x10

复位值：0x0000 0060

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.	BUSY_ERR	BUSY	ADDR_RCVD	Res.	TXE	TDRE	BRI	FE	PE	ORE	RXNE

Bit	Name	R/W	Reset Value	Function
31:9	保留	-	-	保留
10	BUSY_ERR	r_w1	1’b0	Busy Detect Error 检测到 busy 时的误操作。 错误标志：UART 繁忙时（SR[9]设置为 1），master 尝试写入 CR1 寄存器。 数值： ■ 0x0：无 busy 误操作错误。 ■ 0x1： UART 繁忙时（SR[9]设置为 1），master 尝试写入 CR1 寄存器。 该位写 1 清零
9	BUSY	ro	1'b0	UART Busy. UART 忙 这表示串行传输正在进行中，清除时表示 UART 处于空闲或非活动状态。

Bit	Name	R/W	Reset Value	Function
				在以下任何一种情况下，此位都将设置为 1（忙）： -串行接口上正在进行传输 -串行接口上正在进行接收 -传输 TDR 中存在的数据，波特除数为非零（BRR 不等于 0） -接收 RDR 中存在数据 注意：即使可能从另一个设备发送了新字符，UART 忙位也可能被清除。也就是说，如果 UART 在 TDR 和 RDR 中没有数据，并且没有正在进行的传输，并且新字符的起始位刚刚到达 UART。这是由于直到位周期的中间才看到有效的开始，并且该持续时间取决于已编程的波特除数。 数值： 0x0（IDLE）：UART 处于空闲或非活动状态: 0x1（忙）：UART 忙（主动传输数据）
8	ADDR_RCVD	r_w1	1'b0	地址接收。 如果启用 9 位数据模式（CR3[0] = 1），则该位用于指示接收数据的第 9 位被设置为 1。该位还可以用于指示输入字符是地址还是数据。 ■ 1=表示字符为地址。 ■ 0=表示字符为数据。 读取 SR 清除 9BIT。 注意：用户需要确保在下一个地址字节到达之前清除中断（该位写 1 清 0）。 如果清除中断有延迟，则软件将无法区分多个地址相关的中断。 0x1（RCVD_1）：传入字符为地址 0x0（RCVD_0）：传入字符为数据
7	保留	-	-	保留
6	TXE	ro	1'b1	发送为空。 每当当前没有发送数据且 TDR 为空的时候，就会设置此位。 0x0（DISABLED）：发送不为空 0x1（ENABLED）：发送为空
5	TDRE	ro	1'b1	发送保持寄存器空。 该位指示 TDR 为空。 每当数据从 TDR 传输到发送器移位寄存器且没有新数据写入 TDR 时，该位被设置。如果启用了 TDR 中断，这也会导致发生 TDR 中断。 0x0（DISABLED）：TDR 非空 0x1（ENABLED）：TDR 为空
4	BRI	r_w1	1'b0	break 中断位 这用于指示在串行输入数据上检测到中断序列。

Bit	Name	R/W	Reset Value	Function
				如果在 UART 模式下，每当串行输入 sin 保持在逻辑“0”状态的时间超过起始时间+数据位+奇偶校验位+停止位的总和时，就会设置它。 对该位写 1 将清除 BRI 位。 0x0（NO_BREAK）： 未检测到 break 序列 0x1（BREAK）： 检测到 break 序列
3	FE	r_w1	1'b0	帧错误位。 这用于指示接收机中出现帧错误。当接收机在接收的数据中没有检测到有效的 STOP 位时，就会发生帧错误。 应该注意的是，如果发生 break，也将设置帧错误（FE）位（SR[3]），如 break 中断（BRI）位（SR[4]）所示。之所以会发生这种情况，是因为 break 字符通过将 sin 输入保持到逻辑 0 的时间长于字符的持续时间而隐式地生成帧错误。 对该位写 1 将清除 FE 位。 0x0（NO_FRAMING_ERROR）： 无帧错误 0x1（FRAMING_ERROR）： 帧错误
2	PE	r_w1	1'b0	奇偶校验错误位。 如果设置了奇偶校验使能（PEN）位（CR1[3]），则这用于指示接收器中奇偶校验错误的发生。 应该注意的是，如果发生了 break，在这种情况下，如果奇偶校验生成和检测被启用（CR1[3] = 1）并且奇偶校验被设置为奇数（CR1[4] = 0），则 PE 也会被设置。 对该位写 1 将清除 PE 位。 0x0（NO_PARITY_ERROR）： 无奇偶校验错误 0x1（PARITY_ERROR）： 奇偶校验错误
1	ORE	r_w1	1'b0	溢出错误位。 这用于指示溢出错误的发生。 如果在读取以前的数据之前接收到新的数据字符，则会发生这种情况。 当新字符在从 RDR 读取前一个字符之前到达接收器时，设置 ORE 位。当这种情况发生时，RDR 中的数据将被覆盖。 对该位写 1 将清除 ORE 位。 0x0（NO_OVER_RUN_ERROR）： 无溢出错误 0x1（OVER_RUN_ERROR）： 溢出错误
0	RXNE	ro	1'b0	数据就绪位。 这用于指示接收器在 RDR 中至少包含一个字符。 读取 RDR 时，此位被清除。 0x0（NOT_READY）： 数据未就绪

Bit	Name	R/W	Reset Value	Function
				0x1（READY）：数据就绪

20.3.4. 控制寄存器 1 (UART_CR1)

偏移地址：0x14

复位值： 0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res	Res	Res	Res	Res	Res	MSBFIRST	SWAP		SBK	SP	PS	PCE	STOP	M	
						rw	rw		rw	rw	rw	rw	rw	rw	

Bit	Name	R/W	Reset Value	Function
31：10	保留	-	-	保留
9	MSBFIRST	rw	1'b0	MSB first。 0：start bit 后，收发第 0 位数据； 1：start bit 后，收发第 5/6/7/8/9 位数据； 在数据传输过程中，不能修改这个位。
8	SWAP	rw	1'b0	TX/RX pin 互换。 0：TX/RX pin 按照标准 pinout 定义； 1：TX/RX pin 互换。此时用作 cross-wired 连接其他 UART 时。
7	保留	-	-	保留
6	SBK	rw	1'b0	break 控制位。 这用于将 break 发送到接收设备。如果设置为 1，串行输出将强制进入间隔（逻辑 0）状态。sout 线被强制为低电平，直到 SBK 位被清除。 0x0（DISABLED）：释放串行输出以进行数据传输 0x1（ENABLED）：串行输出被迫进入间隔状态
5	SP	rw	1'b0	固定奇偶校验。 仅当 UART 不忙时可写（BUSY[0]为 0）。此位用于强制固定奇偶校验值。当 PCE、PS 和 Stick Parity 设置为 1 时，奇偶校验位被发送并检查为逻辑 0。如果 PCE 和 Stick Parity 被设置为 1，并且 PS 是逻辑 0，则奇偶校验位被发送并被检查为逻辑 1。如果此位设置为 0，则 Stick Parity（固定奇偶校验）被禁用。 0x0（DISABLED）：固定奇偶校验已禁用 0x1（ENABLED）：固定奇偶校验已启用
4	PS	rw	1'b0	偶数奇偶校验选择。 仅当 UART 不忙时可写（BUSY[0]为零）。当奇偶校验被启用（PCE 设置为 1）时，这用于在偶数和奇数奇偶校验之间进行选择。如果设置为

Bit	Name	R/W	Reset Value	Function
				1，则传输或检查偶数个逻辑“1”。如果设置为零，则传输或检查奇数个逻辑“1”。 0x0（ODD_PARITY）：奇校验 0x1（EVEN_PRITY）：偶校验
3	PCE	rw	1'b0	奇偶校验启用。 仅当 UART 不忙时可写（BUSY[0]为零）。该位用于分别启用和禁用发送和接收的串行字符中的奇偶校验生成和检测。 0x0（DISABLED）：禁用奇偶校验 0x1（ENABLED）：启用奇偶校验
2	STOP	rw	1'b0	停止位的数量。 仅当 UART 不忙时可写（BUSY[0]为零）。这用于选择外围设备将发送和接收的每个字符的停止位数。如果设置为零，则在串行数据中传输一个停止位。 如果设置为 1 并且数据位设置为 5（CR1[1:0]设置为 0），则发送一个半停止位。否则，传输两个停止位。 注意，无论所选择的停止位的数量如何，接收器将仅检查第一个停止位。 注：由于某些配置的字符之间插入的空闲时间和传输方向上的波特时钟除数值，UART 实现的 STOP 位持续时间可能会更长； 0x0（STOP_1BIT）：1 个停止位 0x1（STOP_1_5BIT_OR_2BIT）：1.5/2 个停止位
1:0	M	rw	2'b0	数据长度选择。 仅当 UART 不忙时可写（BUSY[0]为零）。当 CR3 中的 M_E 设置为 0 时，此寄存器用于选择外围设备将发送和接收的每个字符的数据位数。 0x0（CHAR_5BITS）：每个字符 5 个数据位 0x1（CHAR_6BITS）：每个字符 6 个数据位 0x2（CHAR_7BITS）：每个字符 7 个数据位 0x3（CHAR_8BITS）：每个字符 8 个数据位

20.3.5. 控制寄存器 2 (UART_CR2)

偏移地址：0x18

复位值： 0x0000_0008

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.	BUSY-ERRIE	LSIE	TDREIE	RXNEIE
												RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:27	保留	-	-	保留
3	BUSYERRIE	RW	1’b1	启用 BUSYERR 状态中断。这用于启用/禁用 BUSYERR 状态中断生成。UART 繁忙时（SR[9] 设置为 1），master 尝试写入 CR1 寄存器。优先级最低=4。 0x0（DISABLED）：禁用 BUSYERR 状态中断 0x1（ENABLED）：启用 BUSYERR 状态中断
2	LSIE	RW	1'b0	启用接收器线路状态中断。这用于启用/禁用接收器线路状态中断的生成。这是最高优先级的中断。 该中断标志是 PE、FE、OE、BI、ADDR_RCV 中断标志的组合。 0x0（DISABLED）：禁用接收器线路状态中断 0x1（ENABLED）：启用接收器线路状态中断
1	TDREIE	RW	1'b0	启用传输保持寄存器空中断。这用于启用/禁用传输保持寄存器空中断的生成。这是优先级第三高的中断。 0x0（DISABLED）：禁用传输空中断 0x1（ENABLED）：启用传输空中断
0	RXNEIE	RW	1'b0	启用接收数据可用中断。这用于启用/禁用接收数据可用中断。这些是优先级第二高的中断。 0x0（DISABLED）：禁用接收数据中断 0x1（ENABLED）：启用接收数据中断

20.3.6. 控制寄存器 3 (UART_CR3)

偏移地址：0x1c

复位值： 0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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.	TX_MODE	SEND_ADDR	ADDR_MATCH	M_E
												RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31：4	保留	-	-	保留
3	TX_MODE	RW	1'b0	传输模式控制位。该比特用于控制 9-bit 数据传输期间的传输模式的类型。 数值：■ 0x1（TX_MODE_1）：在这种操作模式下，传输保持寄存器（TDR）为 9 位宽。 用户需要确保 TDR 寄存器的地址/数据写入正确。 地址：第 9 位设置为 1

Bit	Name	R/W	Reset Value	Function
				数据：第 9 位数设置为 0 注意：传输地址寄存器（TAR）不适用于此操作模式。 ■ 0x0（TX_MODE_0）：在此操作模式中，传输保持寄存器（TDR）的宽度为 8 位。 用户需要将地址编程到传输地址寄存器（TAR）中，并将数据编程到 TDR 寄存器中。 ■ SEND_ADDR 位用作控制旋钮，指示 UART 何时发送地址。
2	SEND_ADDR	RW	1'b0	发送地址控制位。此位用作控制旋钮，供用户在传输模式下确定何时发送地址。注：1.在发出地址字符后，该位由硬件自动清除。用户不应将此位编程为 0。 2.此字段仅在 M_E 位设置为 1 且 TX_MODE 设置为 0 时适用。 数值：■ 0x1（SEND_ADDR_1）：9 位字符内容：第 9 位设置为 1，其余 8 位将与“传输地址寄存器”中正在编程的内容相匹配。 ■ 0x0（SEND_ADDR_0）：9 位字符内容来自 TDR 寄存器
1	ADDR_MATCH	RW	1'b0	地址匹配模式。此位用于在接收期间启用地址匹配功能。 ■ 在地址匹配模式下，UART 将等待第 9 位设置为 1 的传入字符。并进一步检查地址是否与“接收地址匹配寄存器”中编程的地址匹配。如果匹配，则后续字符将被视为有效数据，UART 开始接收数据。 ■ 在正常模式下，UART 将开始接收数据，9 位字符将形成。用户负责读取数据并区分 b/n 地址和数据。 注意：仅当 M_E 设置为 1 时，此字段才适用。 数值：■ 0x0（DISABLE）：正常模式■ 0x1（启用）：地址匹配模式
0	M_E	RW	1'b0	Enable_9_bit 此位用于启用用于发送和接收传输的 9 位数据

20.3.7. 接收地址寄存器（UART_RAR）

地址偏移：0x20

复位值：0x00000000

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.	RAR							
								RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:8	保留	-	-	保留
7:0	RAR	RW	8'b0	这是接收模式下的地址匹配寄存器。如果在输入字符中第 9 位被设置为 1，则将对照该寄存器值检查剩余的 8 位。如果匹配成功，则第 9 位设置为 0 的后续字符将被视为数据字节，直到接收到下一个地址字节。 注：1.仅当“ADDR_MATCH”（CR3[1]）和“M_E”（CR3[0]）位设置为 1 时，此寄存器才适用。 2. RAR 可以在任何时间点被编程。但是，当任何接收正在进行时，用户不得更改此寄存器值。

20.3.8. 发送地址寄存器（UART_TAR）

地址偏移：0x24

复位值：0x00000000

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.	TAR							
								RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:8	保留	-	-	保留
7:0	TAR	RW	8'b0	这是传输模式下的地址匹配寄存器。 如果 M_E（CR3[0]）位被启用，并且“SEND_ADDR”（CR3[2]）位被设置为 1，则 UART 将发送第 9 位设置为 1 的 9 位字符，剩余的 8 位地址将从该寄存器发送。 注：1.此寄存器仅用于发送地址。正常数据应通过编程 TDR 寄存器发送。 2.在 UART 串行通道上开始发送地址后，硬件将自动清除“send_ADDR”位

21. 触控按键

21.1. 介绍

本模块是一个 26 通道的高灵敏度触摸按键传感器，可实现隔空触摸、接近式感应等触摸应用。其抗干扰性能优越，可通过动态 CS 10 V 测试。独特的低功耗设计可使触摸在省电模式下，多按键唤醒芯片整体功耗低于 8 μA。本公司提供了完整的智能化触摸开发套件，结合本公司提供的触摸库和配套工具可快速完成各种触摸应用的开发和调试。

21.2. TouchKey 主要特性

- 最多可支持 26 个触摸按键
- Cmod 电容可选择内置或外接
- 支持两种 TK_SW 模式：PWM 模式和 PRS 模式，PRS 模式多项式可灵活设置
- 支持触摸补偿功能，补偿模式和电压可选
- 支持多通道并联
- 支持软件模式及硬件模式
- 支持低功耗正常和异常唤醒，多按键唤醒芯片整体功耗可低于 8 μA

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。Debug 扩展允许 CPU 停止在一给定指令（breakpoint）或者在数据访问（watchpoint）。当停止时，CPU core 的内部状态和系统的外部状态可能被检查。一旦检查结束，CPU Core 和系统可能被恢复，并继续程序的执行。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）

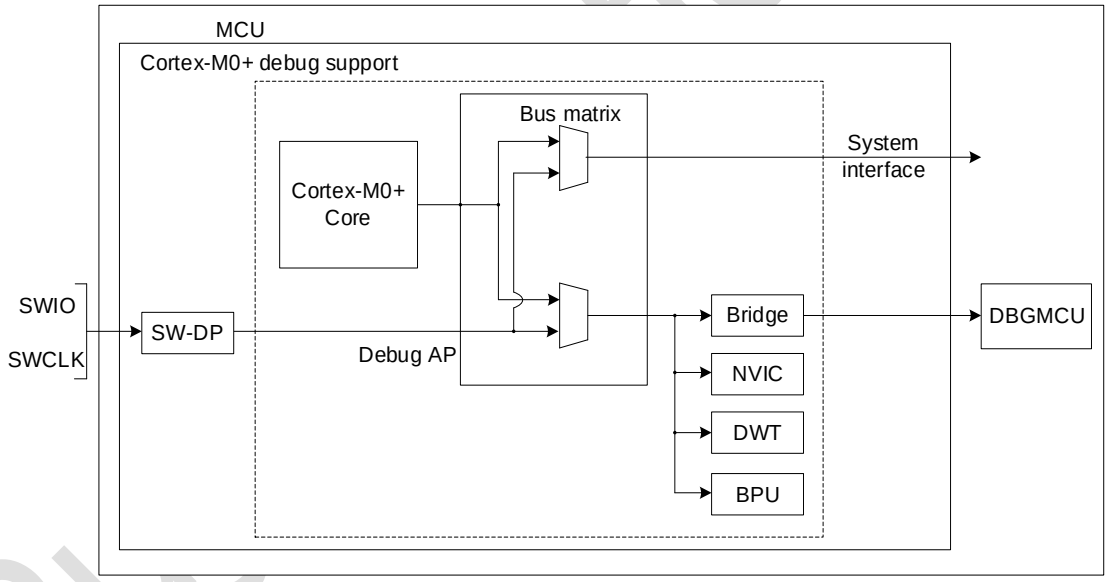


图 22-1DBG 框图

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 pin 有两个，可以作为 GPIO 的 alternate functions 用，这两个 pin，在所有封装形式都可见。

表 22-1 SWD 调试端口

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
SWDCLK	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 100 kΩ 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（8 bits）
- 芯片发送的应答响应（3 bits）

- 主机或者芯片的数据发送阶段（33 bits）

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 代码是默认的 ARM 代码，并被置位 0x0BC11477（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。

表 22-5 SW_DP 寄存器

A[3:2]	R/W	CTRLSEL 位或者 SELECT 寄存器	寄存器	注
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

表 22-6 SW-AP 寄存器

Address	A[3:2] value	Description
0x0	00	保留
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）
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：保留 -Bit 7:4：APBANKSEL：在当前 AP,选择 active 4 个 word 寄存器窗口 -Bit 3:0：保留
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）

22.4.7. 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 22-7 内核调试寄存器

寄存器	描述
DHCSR	32 位的调试控制和状态寄存器
DCRSR	17 位的内核寄存器调试选择寄存器
DHCDR	32 位的内核寄存器调试数据寄存器
DEMCR	32 位异常调试和监视控制寄存器

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

22.5. 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。

22.5.1. 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。

22.6. 数据观察点 DWT (Data Watchpoint)

Cortex-M0 DWT 实现提供了 2 个 watchpoint 寄存器。

22.6.1. DWT 功能

处理器的断点实现基于 PC 的断点功能。

22.6.2. 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。

22.7. 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（sleep、stop）
- 对 timer、watchdog 和 RTC 在 breakpoint 期间的时钟控制

22.7.1. 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。此模式不会对标准调试特性施加任何限制。
- 在 Stop 模式：debugger host 必须提前设置 DBG_STOP 位。这样，系统在 STOP 模式保持 FCLK 和 HCLK 存在。

22.7.2. 支持定时器、看门狗和 RTC

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。

22.8. DBG 寄存器

22.8.1. DBG 设备 ID 代码寄存器(DBG_IDCODE)

偏移地址： 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[31:16]															
R															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
TBD[15:0]															
R															

Bit	Name	R/W	Reset Value	Function
31：0	DBG_IDCODE[31:0]	R	0x0044 0020	MCU 的 ID 编码寄存器

22.8.2. 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。

偏移地址： 0x04

复位值： 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	DBG_SLEEP
														RW	RW

Bit	Name	R/W	Reset Value	Function
31：2	保留	-	-	保留

Bit	Name	R/W	Reset Value	Function
1	DBG_STOP	RW	0	Debug stop 模式。 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 HSI 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	DBG_SLEEP	RW	0	调试睡眠模式。 0：（FCLK 开，HCLK 关）。在睡眠模式，FCLK 由原先配置好的系统时钟提供，HCLK 关闭。由于睡眠模式不会复位已配置好的时钟系统，因此从睡眠模式退出后，软件不需要重新配置时钟。 1：（FCLK 开，HCLK 开）。在睡眠模式，FCLK 和 HCLK 时钟都由原先配置好的系统时钟提供。

22.8.3. DBG APB 冻结寄存器 1 (DBG_APB_FZ1)

该寄存器用来配置 timer、RTC、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。

偏移地址： 0x08

Power on 复位值： 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31：13	保留	-	-	保留
12	DBG_IWDG_STOP	RW	0	当 CPU 核处于 halt 状态时，控制 IWDG 的计数时钟。 0：时钟使能； 1：时钟关闭；
11	保留	-	-	保留
10	DBG_RTC_STOP	RW	0	当 CPU 核处于 halt 状态时，控制 RTC 的计数时钟。 0：时钟使能； 1：时钟关闭；
9:0	保留	-	-	保留

22.8.4. DBG APB 冻结寄存器 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。

偏移地址： 0x0C

Power on 复位值： 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
													RW	RW	
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
DBG_ TIM14_STOP	Res	Res	Res	DBG_ TIM1_STOP	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
RW				RW											

Bit	Name	R/W	Reset Value	Function
31:16	保留	-	-	保留
15	DBG_TIM14_STOP	RW		当 CPU 核处于 halt 状态时，控制 TIM14 的计数时钟。 0：时钟使能； 1：时钟关闭；
14:12	保留	-	-	保留
11	DBG_TIM1_STOP	RW		当 CPU 核处于 halt 状态时，控制 TIM1 的计数时钟。 0：时钟使能； 1：时钟关闭；
10:0	保留	-	-	保留

23. 更新历史

版本	日期	更新记录
V0.5	2024.09.06	初始版本
V0.6	2024.09.12	更新 Flash/PWR 章节

Puya Semiconductor Co., Ltd.