CW32F020 用户手册

ARM® Cortex®-M0+ 32 位微控制器

版本号：Rev 1.0

武汉芯源半导体有限公司

声明

重要声明

本参考手册针对应用程序开发人员，它提供了关于如何使用 CW32F020 微控制器内存和外设的完整信息。仅适用于 CW32F020 设备。

免责声明

我们保留随时对文档内容进行更新的权利，文档内容如有改动，恕不另行通知。请在购买产品前到我们的官网上下载最新版本手册。

版权声明

技术支持

若您在使用过程中有任何意见或建议，请随时与我们联系。

网址：www.whxy.com

通信地址：湖北省武汉市东湖新技术开发区武大园三路 5 号

邮编：430070

目录

1 文档约定

1.1 排版约定

本文中的排版惯例是：

斜体 表示特殊术语或引文，或者提示注意项。

粗体 表示组件或信号名称，如用于无章节序号的标题说明、图表的标题等。

大写英文 用于具有特定技术含义的术语，并包含在词汇表中，如寄存器或位域名称。

彩色文本 表示链接。包括：

- URL 外部链接，如 www.whxy.com
- 交叉引用，本文的内部章节互相引用
- 内部链接，图、表或附录、词汇条目

1.2 寄存器协议

符号示例	描述
0x53CA	带‘0x’前缀的数字字符串表示十六进制数，数值为数字 0 ~ 9 或 A ~ F 大写英文
REGNAME[n]	REGNAME 的特定位，REGNAME 可以是寄存器或寄存器字段，例如，CR1[2] 指的是 CR1 寄存器 (字段) 的第 2 位
REGNAME[m:n]	REGNAME 的特定位，REGNAME 可以是寄存器或寄存器字段，例如，CR1[7:5] 指的是 CR1 寄存器 (字段) 的第 7 位 ~ 第 5 位
RW	可读可写，软件可以读写这些位域
RO	只读，软件只能读取这些位域
WO	只写，软件只能写入该位域，读取该位域时将返回无效数据
RW0	软件可以读写该位域，只能对该位域写入 0，写入 1 对该位域无影响
RW1	软件可以读写该位域，只能对该位域写入 1，写入 0 对该位域无影响
R0W1	软件读取该位域为 0，只能对该位域写入 1，写入 0 对该位域无影响
R1W0	软件读取该位域为 1，只能对该位域写入 0，写入 1 对该位域无影响
RFU	保留位域，请保持默认值
Word	一个字的数据长度为 32 bit
Half Word	一个半字的数据长度为 16 bit
Byte	一个字节的数据长度为 8 bit

2 系统和存储器概述

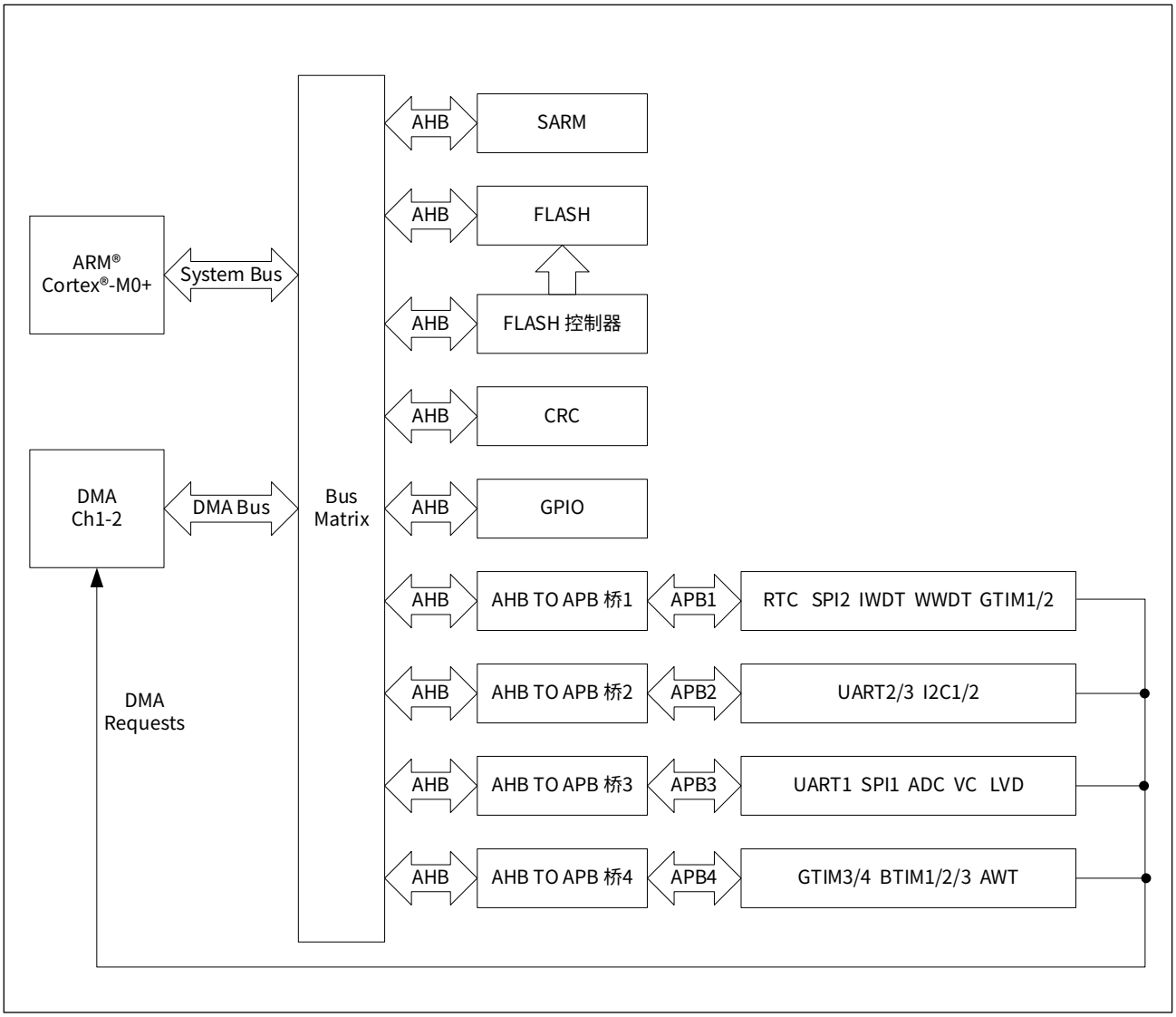
2.1 系统架构

CW32F020 微控制器系统包含：

- 2 个主设备：
  - ARM® Cortex®-M0+ 内核
  - DMA 直接内存存取
- 多个从设备：
  - 片上 SRAM
  - 片上 FLASH
  - FLASH 控制器
  - CRC 冗余计算单元
  - GPIO 端口
  - AHB 到 APB1 转换桥及 APB1 总线上所有设备
  - AHB 到 APB2 转换桥及 APB2 总线上所有设备
  - AHB 到 APB3 转换桥及 APB3 总线上所有设备
  - AHB 到 APB4 转换桥及 APB4 总线上所有设备

主从设备通过 AHB 总线矩阵进行连接，系统架构如下图所示：

图 2-1 系统架构



- 系统总线

  实现 M0+ 微处理器的外设总线和总线矩阵的连接。

- DMA 总线

  实现 DMA 的 AHB 总线和总线矩阵的连接。

- 总线矩阵

  管理 M0+ 微处理器和 DMA 对 FLASH、SRAM 以及所有外设的访问存取仲裁。仲裁控制采用轮询调度算法来对负载进行均衡处理，保证总线利用效率。

- AHB TO APB 桥 1/2/3/4

  提供 AHB 总线到 APB1/APB2/APB3/APB4 总线的完全同步的连接，即 AHB 和 APB 总线的桥接。

2.2 存储器组织

2.2.1 概述

CW32F020 内核为 32 位的 ARM® Cortex®-M0+ 微处理器，最大寻址空间为 4GB。芯片内置的程序存储器、数据存储器、各外设及端口寄存器被统一编址在同一个 4GB 的线性地址空间内。

存储器中字节组织为小端格式。一个字存储空间的最低字节数据为字的最低有效位，最高字节数据为最高有效位。

例：

将 0x1122 3344 存放在地址为 0x2000 0000 的存储器空间中，实际存放结果是：

0x20000000 字节存放 0x44，

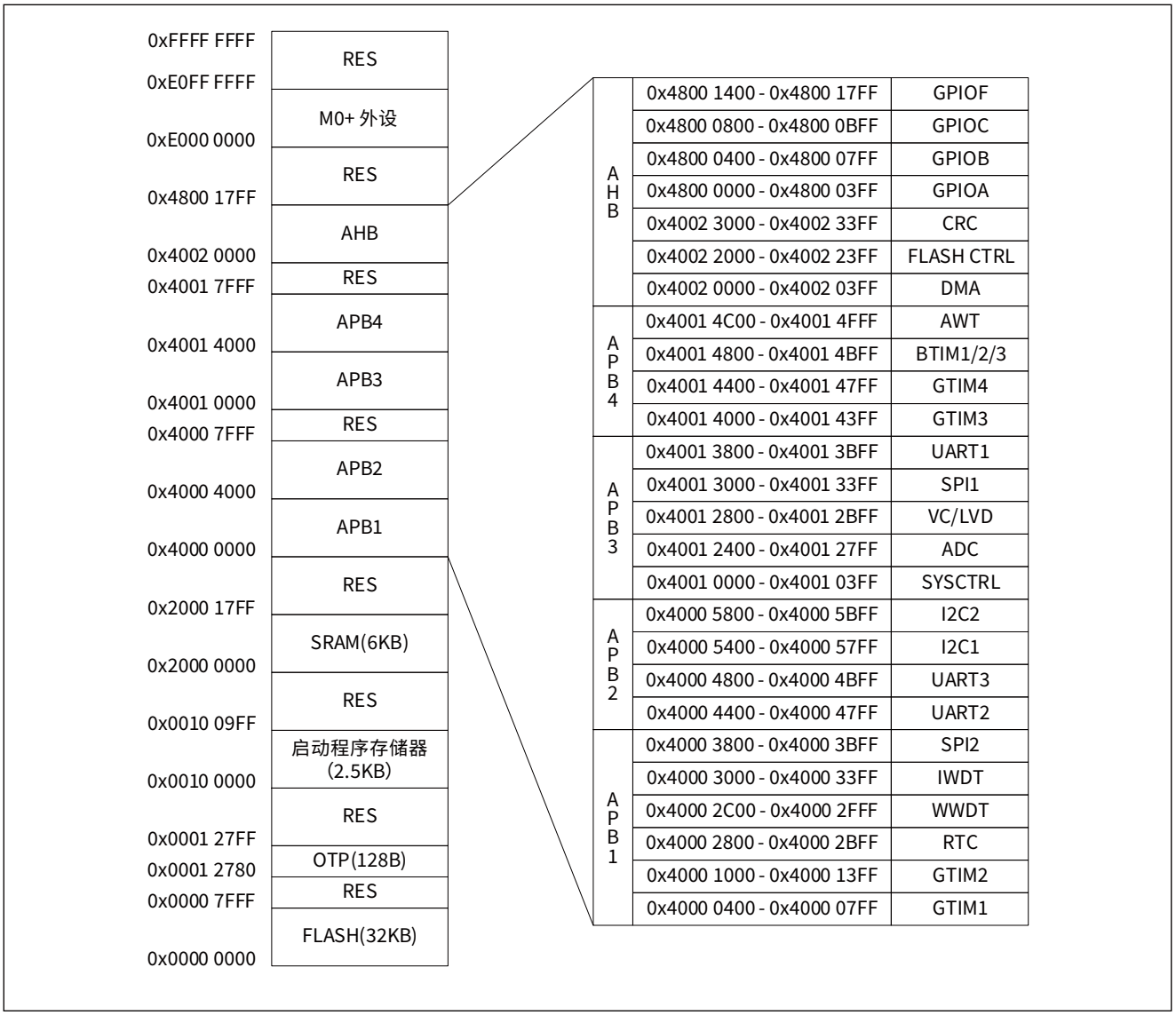
0x20000001 字节存放 0x33，

0x20000002 字节存放 0x22，

0x20000003 字节存放 0x11。

系统地址分配如下图所示，RES 为保留区域。

图 2-2 系统地址分配



2.2.2 存储器映射和寄存器边界地址

片上存储器及各外设的详细起始地址空间分配，如下表所示：

表 2-1 存储器和外设地址分配

设备或总线	边界地址	大小	对应外设
主 FLASH 存储器	0x0000 0000 - 0x0000 7FFF	32KB	主 FLASH
OTP 存储器	0x0001 2780 - 0x0001 27FF	128B	OTP
启动程序存储器	0x0010 0000 - 0x0010 09FF	2.5KB	BootLoader
SRAM 存储器	0x2000 0000 - 0x2000 1FFF	8KB	SRAM
APB1 外设	0x4000 0400 - 0x4000 07FF	1KB	GTIM1
	0x4000 1000 - 0x4000 13FF	1KB	GTIM2
	0x4000 2800 - 0x4000 2BFF	1KB	RTC
	0x4000 2C00 - 0x4000 2FFF	1KB	WWDT
	0x4000 3000 - 0x4000 33FF	1KB	IWDT
	0x4000 3800 - 0x4000 3BFF	1KB	SPI2
APB2 外设	0x4000 4400 - 0x4000 47FF	1KB	UART2
	0x4000 4800 - 0x4000 4BFF	1KB	UART3
	0x4000 5400 - 0x4000 57FF	1KB	I2C1
	0x4000 5800 - 0x4000 5BFF	1KB	I2C2
APB3 外设	0x4001 0000 - 0x4001 03FF	1KB	SYSCTRL
	0x4001 2400 - 0x4001 27FF	1KB	ADC
	0x4001 2800 - 0x4001 2BFF	1KB	VC/LVD
	0x4001 3000 - 0x4001 33FF	1KB	SPI1
	0x4001 3800 - 0x4001 3BFF	1KB	UART1
APB4 外设	0x4001 4000 - 0x4001 43FF	1KB	GTIM3
	0x4001 4400 - 0x4001 47FF	1KB	GTIM4
	0x4001 4800 - 0x4001 4BFF	1KB	BTIM1/2/3
	0x4001 4C00 - 0x4001 4FFF	1KB	AWT

设备或总线	边界地址	大小	对应外设
AHB 外设	0x4002 0000 - 0x4002 03FF	1KB	DMA
	0x4002 2000 - 0x4002 23FF	1KB	FLASH CTRL
	0x4002 3000 - 0x4002 33FF	1KB	CRC
	0x4800 0000 - 0x4800 03FF	1KB	GPIOA
	0x4800 0400 - 0x4800 07FF	1KB	GPIOB
	0x4800 0800 - 0x4800 0BFF	1KB	GPIOC
	0x4800 1400 - 0x4800 17FF	1KB	GPIOF
M0+ 外设	0xE000 0000 - 0xE00F FFFF	1MB	M0+ 内核外设

2.3 片上 SRAM 存储器

CW32F020 内部集成 8KB 的片上 SRAM，起始地址为 0x2000 0000。SRAM 支持以字节 (8bit)、半字 (16bit) 或全字 (32bit)3 种位宽进行访问，能够被 CPU 和 DMA 以最大的系统时钟频率进行访问，零等待延迟。

- 奇偶校验

  SRAM 支持奇偶校验功能，芯片上电后默认打开，用户不可配置。

  SRAM 的数据总线为 36bit，包括 32bit 数据位以及 4bit 奇偶校验位（每 8bit 数据位配有 1bit 的奇偶校验位），用来增加存储器的健壮性。

  奇偶校验位在 CPU 对 SRAM 进行写入时计算和存储，在 CPU 对 SRAM 进行读取时自动检查。系统检测到奇偶校验失败后，会产生对应的中断标志。

关于 SRAM 的详细介绍，请参见 6 RAM 存储器章节。

2.4 片上 FLASH 闪存存储器

片上 FLASH 闪存由两部分物理区域组成：主 FLASH 存储器和启动程序存储器。

- 主 FLASH 存储器，共 32KB，地址空间为 0x0000 0000 - 0x0000 7FFF。该区域主要用于存放应用程序代码和用户数据，用户可编程。
- 启动程序存储器，共 2.5KB，地址空间为 0x0010 0000 - 0x0010 09FF。该区域主要用于存储 BootLoader 启动程序，在芯片出厂时已编程，用户不可更改。

FLASH 控制器实现对 FLASH 的各种操作（擦除、写、读取），内部的预取缓存机制可加速 CPU 代码执行速度。

FLASH 支持以字节 (8bit)、半字 (16bit) 或全字 (32bit)3 种位宽进行访问，访问的最高频率为 24MHz，如果系统配置的 HCLK 时钟频率高于 24MHz，则必须通过 FLASH 控制寄存器 FLASH_CR2 的 WAIT 位域配置合理的响应等待时间，才能保证 FLASH 被正确访问。

关于 FLASH 的详细介绍，请参见 7 FLASH 存储器章节。

2.5 一次性可编程 OTP 存储器

芯片内部有 128B 的存储器空间为 OTP 区域，地址空间为 0x0001 2780 - 0x0001 27FF。该存储器区域只可通过 ISP 编程指令写入数据，且仅能写入 1 次，之后就只能被读取，不能擦除和写入。该区域主要用于存储不可更改的信息，如用户自定义的产品 UID、算法密钥等，在设备出厂时写入。

2.6 系统启动配置

设备支持 2 种不同的启动模式，通过 BOOT 引脚状态进行选择，如下表所示：

表 2-2 启动模式配置

启动模式配置	启动模式
BOOT = 0	从主 FLASH 存储器启动，运行用户程序。
BOOT = 1	从启动程序存储器启动，固定运行芯片的 BootLoader 程序，此时用户可通过 UART1 接口（PA13/PA14）利用 ISP 通信协议进行 FLASH 编程。

启动模式选择电路只在芯片解复位时刻采样 BOOT 引脚状态，因此在芯片解复位前用户必须根据需要设置好 BOOT 引脚的电平状态，以决定本次芯片复位后的启动模式。

系统启动完成之后，CPU 从存储器的 0x0000 0000 地址获取堆栈顶的地址，并从存储器的 0x0000 0004 指示的地址开始执行代码。

BootLoader 程序位于启动程序存储器区域，由设备提供商在生产时进行编程。用户可以通过 UART1（引脚为 PA13/PA14）利用 ISP 通信协议进行 FLASH 编程。

2.7 注意事项

CW32F020 在使用中需要注意如下事项：

- FLASH、SRAM 以及 GPIOx_ODR、CRC_DR 等少数寄存器支持 8bit/16bit/32bit 访问方式，其它外设只支持 32bit 访问方式。对不支持 8bit/16bit 访问方式的地址进行 8bit/16bit 访问会产生 HardFault 硬件错误异常。
- 图 2-2 系统地址分配中的 RES 区域均为保留的地址空间，无对应的物理设备，如果对这些保留地址空间进行访问，会产生 HardFault 硬件错误异常。
- 芯片复位后，除了 SYSTICK 和 SRAM 外的所有外设时钟都处于关闭状态，用户在使用这些外设前必须通过外设时钟使能控制寄存器（AHB 外设时钟使能控制寄存器 SYSCTRL_AHBEN、APB 外设时钟使能控制寄存器 SYSCTRL_APBEN1/2）打开对应外设的配置时钟和工作时钟。没有打开外设配置时钟会导致访问失败：写入数据失败，且不会产生 HardFault 硬件错误异常，读数据的结果不可用。没有打开外设的工作时钟，则外设主要功能失效。

3 电源控制（PWR）与功耗

3.1 概述

CW32F020 工作时需要两组电源供电：工作电源（VDD、VSS）和模拟电源（VDDA、VSSA）。用户可以选择工作电源和模拟电源使用同一电源，也可以使用两组不同电源，但是二者相差不能超过 0.3V，详细电源内容请参阅数据手册相关章节。

CW32F020 内嵌一路 1.5V 低压差 LDO 稳压器，为芯片内部数字电源域供电。

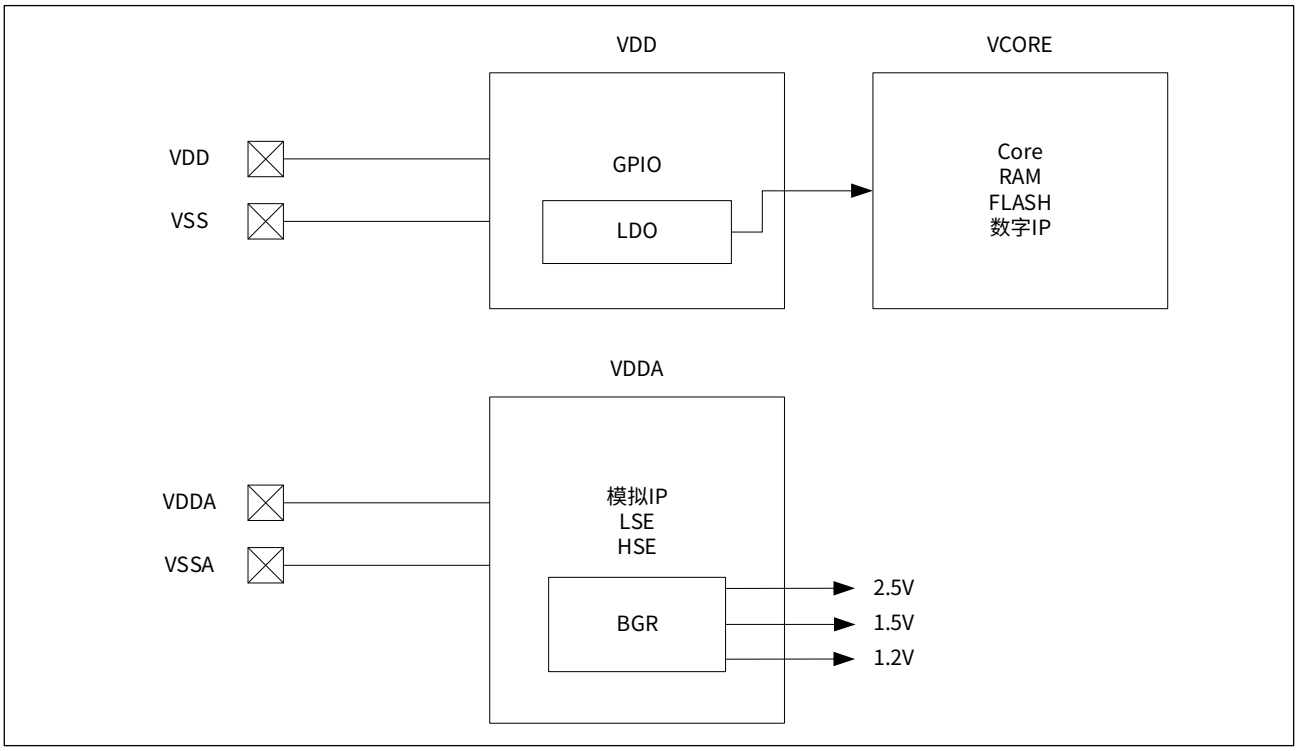
芯片内置参考电压生成器（BGR）电路，可为其他模拟模块提供参考电压。

注：

模拟电源电压与参考电压的压差须高于 0.3V。关于电压参考的详细参数请参阅数据手册相关章节。

芯片内部电源域的划分与分配，如下图所示：

图 3-1 系统电源分配



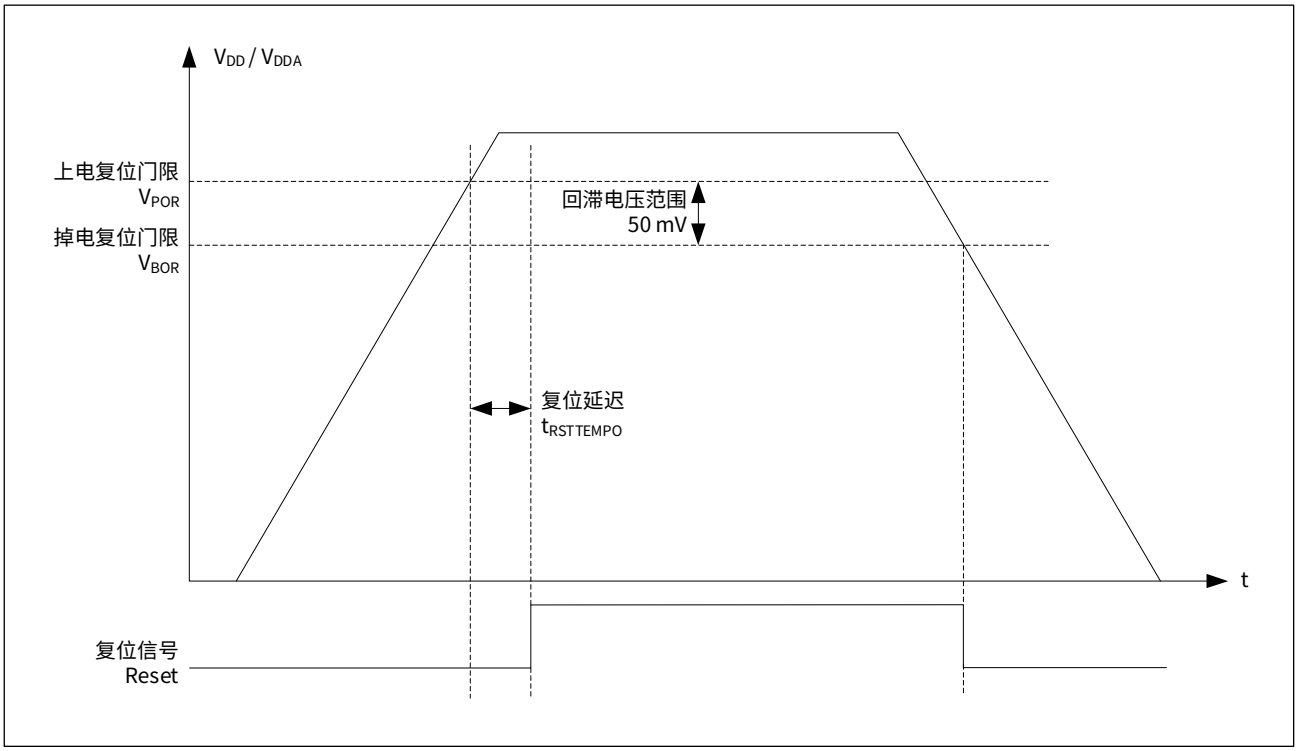
3.2 电源监控

上电复位（POR）/ 掉电复位（BOR）

CW32F020 集成了上电复位（POR）和掉电复位（BOR）电源监控电路，电源上电后始终处于工作状态。POR/BOR 同时监控 VDD 和 VDDA 电源电压，当监测到电源电压低于复位阈值（V_{BOR}）时，系统会进入复位状态。用户无需额外增加外部硬件复位电路。

电源上电启动以及电源跌落阶段的复位信号波形，如下图所示：

图 3-2 上电 / 掉电复位时序图



关于上电 / 掉电复位阈值的更多详细信息，请参阅本芯片的数据手册中的电气特征章节。

3.3 工作模式

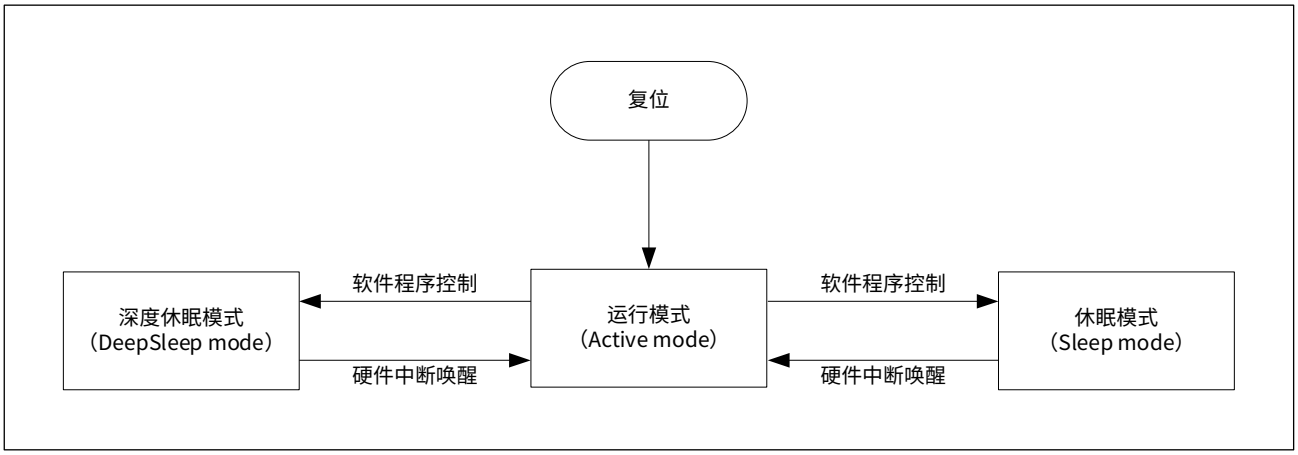
CW32F020 支持三种工作模式，由内嵌的电源管理模块自动完成电源的统一管理。三种工作模式是：

- 运行模式（Active mode）
- 休眠模式（Sleep mode）
- 深度休眠模式（DeepSleep mode）

电源上电后，系统自动进入运行模式。用户可通过软件程序，进入休眠或深度休眠两种低功耗运行状态；在低功耗运行状态时，可通过硬件中断触发唤醒机制，使系统返回到运行模式。

三种工作模式的转换机制，如下图所示：

图 3-3 工作模式转换机制



运行模式下 CPU 正常运行，所有模块用户均可正常使用。休眠模式下，CPU 停止运行，所有外设不受影响，所有 I/O 引脚保持状态不变。深度休眠模式下，CPU 停止运行，高速时钟（HSE、HSIOSC）自动关闭，低速时钟（LSE、LSI、RC10K、RC150K）保持原状态不变。

不同工作模式下 CPU 与时钟状态，如下表所示：

表 3-1 工作模式与 CPU 及时钟状态

工作模式	CPU 状态	高速时钟	低速时钟
运行模式（Active mode）	运行	ON/OFF	ON/OFF
休眠模式（Sleep mode）	停止	ON/OFF	ON/OFF
深度休眠模式（DeepSleep mode）	停止	OFF	ON

3.3.1 进入休眠模式或深度休眠模式

使用 M0+ 内核的 ARM 等待中断专用指令，WFI（Wait for Interrupt），配合 M0+ 内核的系统控制寄存器（SCR，System Control Register）的 SLEEPONEXIT 和 SLEEPDEEP 位域，可实现立即进入或退出（中断服务程序）时进入休眠模式或深度休眠模式。

- 立即进入

  执行 WFI 指令，MCU 将立即进入休眠模式（SLEEPDEEP 为 0 时）或深度休眠模式（SLEEPDEEP 为 1 时）

- 退出时进入

  将 SLEEPONEXIT 位置 1，当退出最低优先级的中断服务程序后，MCU 会进入休眠模式（SLEEPDEEP 为 0 时）或深度休眠模式（SLEEPDEEP 为 1 时），而不需执行 WFI 指令 。

表 3-2 进入休眠模式或深度休眠模式

进入方式	SLEEPONEXIT 位	进入条件	SLEEPDEEP 位	进入模式
立即进入	-	执行 WFI 指令	0	休眠模式
			1	深度休眠模式
退出时进入	1	退出最低优先级的中断服务程序后	0	休眠模式
			1	深度休眠模式

注 1：

在深度休眠模式下，系统将自动关闭高速时钟。如用户需要在深度休眠模式下使部分外设仍保持运行，则须在进入深度休眠模式前，启动相应的低速时钟并将该外设时钟设置为此低速时钟。

注 2：

在进入深度休眠模式之前，用户必须配置 HCLK 时钟频率小于或等于 4MHz，否则易造成内核损坏。

注 3：

若使能了VCx，必须等待VCx_SR.READY标志位置1后才可以进入深度休眠模式，否则无法进入深度休眠模式。

3.3.2 退出休眠模式或深度休眠模式

在休眠模式或深度休眠模式下，均可通过中断来唤醒 CPU，返回到运行模式。但是，值得注意的是，如果用户在中断服务程序中执行 WFI 命令进入休眠（包括深度休眠），则需要比此中断更高优先级的中断才能唤醒 CPU，因此，我们强烈建议用户在准备进入休眠前，应先处理完所有中断服务程序，并且清除所有中断请求和中断标志。

不同工作模式下，CPU 可响应的中断类型，如下表所示：

表 3-3 工作模式与中断源

中断号	中断源	运行模式	休眠模式	深度休眠模式
0	IWDT	Y	Y	Y
	WWDT	Y	Y	N
1	LVD	Y	Y	Y
2	RTC	Y	Y	Y
3	FLASH	Y	Y	N
	RAM	Y	Y	N
4	RCC	Y	Y	Y
5	GPIOA	Y	Y	Y
6	GPIOB	Y	Y	Y
7	GPIOC	Y	Y	Y
8	GPIOF	Y	Y	Y
9	DMA1	Y	Y	N
10	DMA2	Y	Y	N
12	ADC	Y	Y	N
14	VC1	Y	Y	Y
15	VC2	Y	Y	Y
16	GTIM1	Y	Y	N
17	GTIM2	Y	Y	N
18	GTIM3	Y	Y	N
19	GTIM4	Y	Y	N
20	BTIM1	Y	Y	N
21	BTIM2	Y	Y	N
22	BTIM3	Y	Y	N
23	I2C1	Y	Y	N
24	I2C2	Y	Y	N
25	SPI1	Y	Y	N

中断号	中断源	运行模式	休眠模式	深度休眠模式
26	SPI2	Y	Y	N
27	UART1	Y	Y	Y
28	UART2	Y	Y	Y
29	UART3	Y	Y	Y
30	AWT	Y	Y	Y
31	FAULT	Y	Y	Y

使用中断退出休眠模式，用户必须在进入休眠（包括深度休眠）前使能此中断的允许位。

中断唤醒退出休眠模式后，CPU 将立即进入此中断的中断服务程序。如果用户未设置此中断服务程序，且为立即进入休眠时：CPU 将继续执行进入休眠的 WFI 指令的下一条语句；而为退出时进入休眠时：继续执行最后进入的中断服务程序的下一条语句。一般情况下，基于系统可靠性考虑，强烈建议用户设置此中断的服务程序，并在中断服务程序中清除中断请求和中断标志。

中断唤醒退出深度休眠模式时，CPU 运行状态与退出休眠模式相同。

深度休眠模式下系统将自动关闭高速时钟，在退出深度休眠时，CW32F020 为用户额外增加了一种系统时钟选择，用户既可以选择继续使用进入深度休眠时使用的时钟，也可选择 HSI 作为系统时钟。配置系统控制寄存器 SYSCTRL_CR2 的 WAKEUPCLK 位域为 1，则在中断唤醒退出深度休眠模式后自动使用内部高速时钟 HSI 作为系统时钟，由于 HSI 时钟的恢复时间比 HSE 更快，从而可以加速系统唤醒。

3.3.3 工作模式与复位源

即使在休眠模式或深度休眠模式，CPU 亦可响应部分复位源。不同工作模式下 CPU 可响应的硬件复位或软件复位，如下表所示：

表 3-4 工作模式与复位源

复位源	运行模式	休眠模式	深度休眠模式
上电复位 / 掉电复位（POR/BOR）	Y	Y	Y
引脚输入复位 (NRST)	Y	Y	Y
LVD 复位	Y	Y	Y
IWDT 复位	Y	Y	Y
WWDT 复位	Y	Y	N
内核 LOCKUP 故障复位	Y	N	N
内核 SYSRESETREQ 复位	Y	N	N

3.4 低功耗应用

休眠模式下，CPU 停止运行，所有外设保持运行，包括 ARM® Cortex®-M0+ 内核外设，比如 NVIC、SysTick 等外设。休眠模式的功耗低于运行模式。

深度休眠模式下，CPU 停止运行，高速时钟关闭，低速时钟保持状态不变，部分外设可以配置为继续运行，NVIC 中断处理仍然工作。深度休眠模式的功耗远小于休眠模式。

用户可以通过以下方式降低系统运行功耗：

降低系统时钟频率

- 使用低频率的高速时钟 HSI、HSE 或低速时钟 LSI、LSE
- 通过编程预分频寄存器，降低 SYSCLK、HCLK、PCLK 的频率
  - 设置 SYSCTRL_CR0 寄存器的 SYSCLK 位域，选择适当的时钟源
  - 设置 SYSCTRL_CR0 寄存器的 HCLKPRS 位域，降低 HCLK 频率
  - 设置 SYSCTRL_CR0 寄存器的 PCLKPRS 位域，降低 PCLK 频率

关闭休眠期间不使用的时钟和外设

- AHB 总线时钟 HCLK 和 APB 总线时钟 PCLK，可以根据需要关闭
- 关闭与唤醒无关的外设的时钟
  - AHB 外设时钟使能控制寄存器，SYSCTRL_AHBEN
  - APB 外设时钟使能控制寄存器 1，SYSCTRL_APBEN1
  - APB 外设时钟使能控制寄存器 2，SYSCTRL_APBEN2

3.5 Cortex®-M0+ 内核系统控制寄存器 (SCB->SCR)

Address: 0xE000 ED10 Reset value: 0x0000 0000

位域	名称	权限	功能描述
31:5	RFU	-	保留位，请保持默认值
4	SEVONPEND	RW	设置为 1 时，中断的每次新的挂起都会产生一个事件，如果使用了 WFE 休眠，它可用于唤醒处理器。
3	RFU	-	保留位，请保持默认值
2	SLEEPDEEP	RW	设置为 1 时，执行 WFI 或 SLEEPONEXIT 为 1 且退出所有中断服务程序时进入深度休眠模式（DeepSleep mode）； 设置为 0 时，执行 WFI 或 SLEEPONEXIT 为 1 且退出所有中断服务程序时进入休眠模式（Sleep mode）
1	SLEEPONEXIT	RW	设置为 1 时，当退出所有中断服务程序时，处理器自动进入休眠模式 (或深度休眠模式)； 设置为 0 时，退出时进入休眠功能被禁止
0	RFU	-	保留位，请保持默认值

4 复位和时钟（RCC）

4.1 系统复位

CW32F020 支持以下 6 种系统复位：

- 上电复位 / 掉电复位（POR/BOR）
- 引脚输入复位（NRST）
- IWDT/WWDT 复位
- LVD 复位
- 内核 SYSRESETREQ 复位
- 内核 LOCKUP 故障复位

每种系统复位的复位范围如下表所示：

表 4-1 复位方式及范围

复位方式	复位范围
上电复位 / 掉电复位（POR/BOR）	整个 MCU
引脚输入复位（NRST）	整个 MCU（除 RTC 外）
IWDT/WWDT 复位	M0+ 内核 / 外设（除 RAM 控制器 /RTC 外）
LVD 复位	M0+ 内核 / 外设（除 LVD 控制部分 /RTC 外）
内核 SYSRESETREQ 复位	M0+ 内核（除 SWD 调试逻辑外）/ 外设（除 RAM 控制器 /LVD/RTC 外）
内核 LOCKUP 故障复位	M0+ 内核 / 外设（除 RAM 控制器 /LVD/RTC 外）

发生系统复位后，CPU 重新运行，大部分寄存器都被复位到默认值，程序从中断向量表的复位中断入口地址开始执行。

用户可通过系统复位标志寄存器 SYSCTRL_RESETFLAG 来查询本次系统复位的复位源。复位标志由硬件置位，软件清零。建议用户在读取标志后清除该寄存器相关标志位使标志位为 0，以避免在下次复位后发生混淆。

4.1.1 上电复位（POR）/ 掉电复位（BOR）

CW32F020 集成了专门的 POR 和 BOR 电路对电源电压进行监控，在电源电压低于安全范围时将芯片保持在复位状态，防止芯片在上电 / 掉电过程中误动作。为保证系统工作稳定，用户须保持电源电压在安全范围内。

POR 和 BOR 的具体功能请参见 3 电源控制（PWR）与功耗章节。

注：

为保证芯片解除复位后工作正常，须在电路设计上保证 VDD/VDDA 同时上下电。

4.1.2 引脚输入复位（NRST）

CW32F020具有专门的复位输入引脚，输入一定宽度的低电平信号会引起系统复位。芯片内部设计有专用防抖电路，短于 20μs 的低电平脉冲信号会被屏蔽。

复位输入引脚内置有上拉电阻，用户如需外接 RC 电路，须考虑内部上拉电阻的影响。内置上拉电阻的电路特性请参阅数据手册相关章节。

4.1.3 IWDT/WWDT 复位

CW32F020 集成有独立看门狗 (IWDT) 和窗口看门狗 (WWDT)，当 IWDT 或 WWDT 满足复位条件时，会产生复位信号引起系统复位。

IWDT 和 WWDT 复位详细功能请参见 15 独立看门狗定时器（IWDT）、16 窗口看门狗定时器（WWDT）章节。

4.1.4 LVD 低电压检测复位

CW32F020 内部集成低电压检测器 (LVD)，可将选定的监测电压和设定的 LVD 门限电压持续比较，当比较结果满足触发条件时，将产生 LVD 复位信号引起系统复位。

与 POR/BOR 复位方式相比，LVD 低电压检测复位的功能更强大，复位门限电压、监测电压源、脉冲滤波宽度和迟滞时间都可通过相关寄存器进行设置。

LVD 详细信息请参见 23 低电压检测器（LVD）章节。

4.1.5 内核 SYSRESETREQ 复位

内核 SYSRESETREQ 复位是软件复位，通过设置 ARM® Cortex®-M0+ 的应用中断和控制状态寄存器（AIRCR，Application Interrupt and Reset Control Register）的 SYSRESETREQ 位域来实现。应用程序设置该位为 1 则会产生内核 SYSRESETREQ 复位，从而实现软件复位。

4.1.6 内核 LOCKUP 故障复位

当 CPU 遇到严重异常（如读取到的指令无效、访问 FLASH 时位宽和目标地址不匹配），会将 PC 指针停在当前地址处锁定，并产生内核 LOCKUP 故障复位信号。

芯片上电后，LOCKUP 复位功能默认处于不使能状态，用户需要手动使能，通过设置系统控制寄存器 SYSCTRL_CR2 的 LOCKUP 位域为 1，即可使能 LOCKUP 复位功能。

4.2 外设复位

用户可使用软件将 CW32F020 内部的各种外设单独复位。外设复位可以让各外设的寄存器、状态机以及各种控制逻辑等，恢复到上电复位后的默认状态。用户通过设置 SYSCTRL_AHBRST、SYSCTRL_APBRST1、SYSCTRL_APBRST2 这 3 个寄存器来进行各外设模块的独立复位操作。

对于上电复位 / 掉电复位（POR/BOR），内部各外设模块已处于复位后的默认状态，可直接进行模块初始化配置，不需要再单独对各模块进行独立复位。

对于其它类型系统复位情形，部分外设可能保留复位前的工作状态。用户如需要将外设恢复到上电后的默认状态，应通过对应的复位寄存器执行外设复位操作后再使用。RTC 模块是典型的可能需要在复位后保留当前状态的外设，具体判断逻辑示例请参阅相关例程。

4.3 时钟及控制

4.3.1 概述

CW32F020 内置多路时钟产生电路，通过分频器和多路选择器产生各种不同频率的时钟给 CPU 及各外设使用。系统时钟树结构示意图如图 4-1 系统内部时钟树所示。

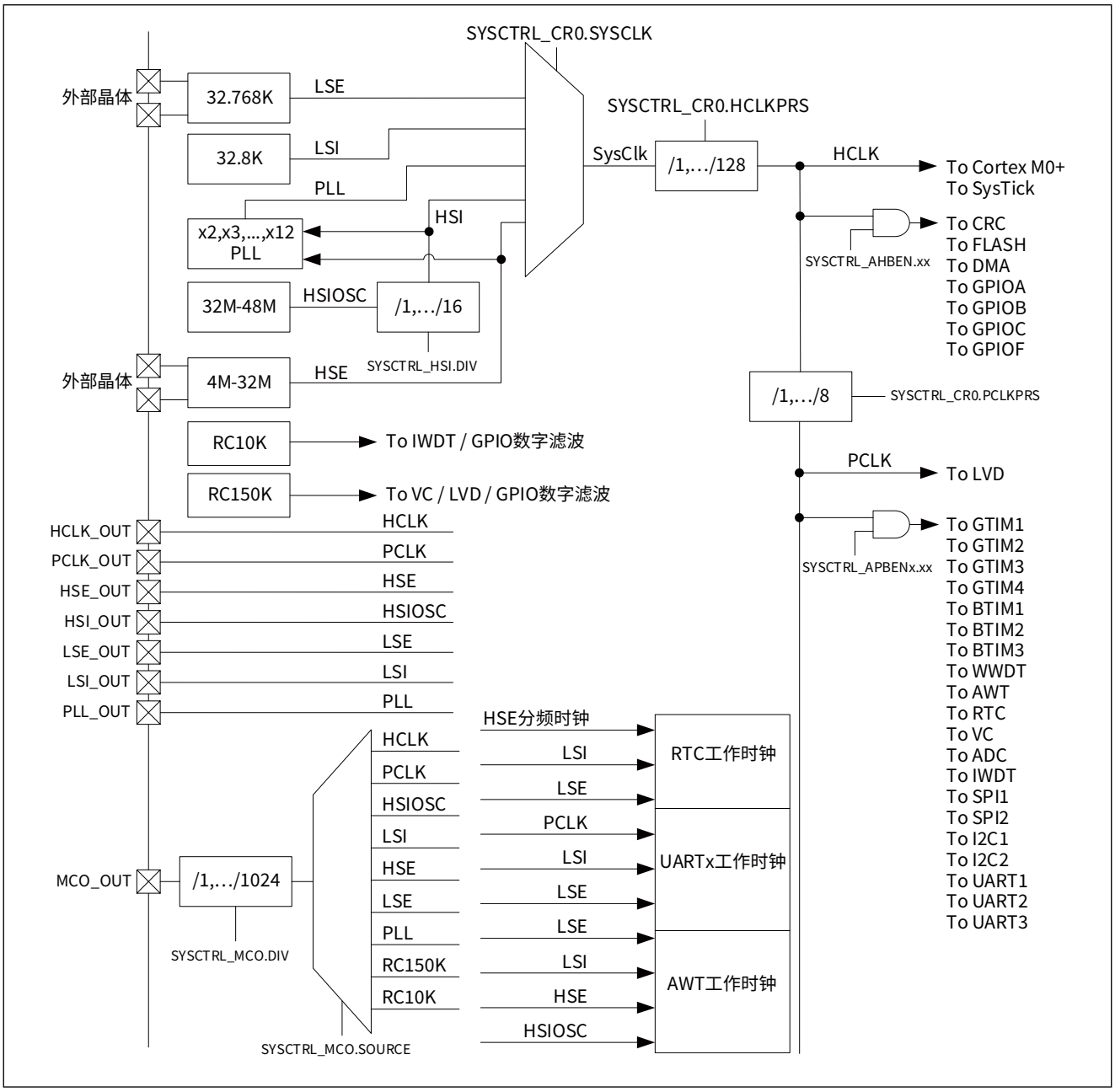
系统内部时钟 SysClk 经过分频为 CPU 内核提供高级高性能总线时钟 HCLK，HCLK 时钟经过分频为数字及模拟外设提供高级外设时钟 PCLK。

系统时钟 SYSCLK 有 5 个时钟源：

- 外部高速振荡器时钟（HSE）
- 外部低速振荡器时钟（LSE）
- HSI 时钟，由内部高速 RC 振荡器时钟（HSIOSC）经过分频产生
- 内部低速 RC 振荡器时钟（LSI）
- PLL 锁相环时钟，由 HSE 时钟或 HSI 时钟经锁相环 PLL 倍频（2~12 倍）产生

系统内部时钟树如下图所示：

图 4-1　系统内部时钟树



内部高速 RC 振荡器时钟 HSIOSC 经过分频器分频后产生 HSI 时钟，分频系数通过内置高频时钟控制寄存器 SYSCTRL_HSI 的 DIV 位域进行设置，有效分频系数为 1、2、4、6、8、10、12、14、16。

系统时钟 SysClk 可选 5 个时钟源：HSE、LSE、HSI、LSI、PLL，通过系统控制寄存器 SYSCTRL_CR0 的 SYSCLK 位域进行选择。

高级高性能总线时钟 HCLK，由系统时钟 SysClk 经过分频产生，作为 M0+ 内核、SysTick、DMA、FLASH、CRC、GPIO 等模块的配置时钟及工作时钟。分频系数通过系统控制寄存器 SYSCTRL_CR0 的 HCLKPRS 位域设置，有效分频系数为 2^n (n = 0~7)。

高级外设时钟 PCLK，由 HCLK 经过分频产生，作为定时器、SPI、I2C 等外设的配置时钟及工作时钟。分频系数通过系统控制寄存器 SYSCTRL_CR0 的 PCLKPRS 位域设置，有效分频系数为 2^n (n = 0~3)。

除作为系统时钟的 5 个时钟源外，CW32F020 还内置 2 个专用的低速时钟源：

- 内部 RC10K 时钟
  可作为 IWDT 模块计数时钟，以及 GPIO 端口中断输入信号的滤波时钟
- 内部 RC150K 时钟
  可作为 LVD 和 VC 的数字滤波模块的滤波时钟，以及 GPIO 端口中断输入信号的滤波时钟

4.3.2 系统时钟与工作模式

CW32F020 支持三种工作模式：运行模式（Active mode），休眠模式（Sleep mode）和深度休眠模式（DeepSleep mode）。

运行模式下 CPU 正常运行，所有模块用户均可正常使用。

休眠模式下，CPU 停止运行，各时钟振荡器及外设保持原状态不变。

深度休眠模式下，CPU 停止运行，高速时钟 HSE、HSIOSC 的振荡器以及 PLL 被自动关闭以节省功耗；低速时钟 LSE、LSI、RC10K、RC150K 的振荡器保持原状态不变；系统时钟 SysClk 及 HCLK、PCLK 时钟是否有效，取决于 SysClk 系统时钟的时钟源状态。

从深度休眠模式唤醒后，如果系统控制寄存器 SYSCTRL_CR2 的 WAKEUPCLK 位域配置为 1，则系统会自动使用 HSI 作为系统时钟的时钟源。如果 SYSCTRL_CR2.WAKEUPCLK 为 0，则系统会等待系统进入深度休眠前所使用的系统时钟源稳定后才开始运行；如果系统时钟的时钟源为 HSE 或者 PLL，时钟恢复速度及系统响应速度会比较慢。由于 HSI 启动速度很快，可快速响应用户需求，因此建议在进入深度休眠模式前切换系统时钟的时钟源为 HSI 时钟或者配置 SYSCTRL_CR2.WAKEUPCLK 为 1。

4.3.3 HSE 时钟

外部高速时钟（HSE）支持两种工作模式：

- 石英晶体 / 陶瓷谐振器，设置外置高频晶体控制寄存器 SYSCTRL_HSE 的 MODE 位域为 0，同时须配置 OSC_IN、OSC_OUT 引脚为模拟功能。
- 外部时钟输入，设置外置高频晶体控制寄存器 SYSCTRL_HSE 的 MODE 位域为 1，同时须配置 OSC_IN 引脚为数字输入功能。

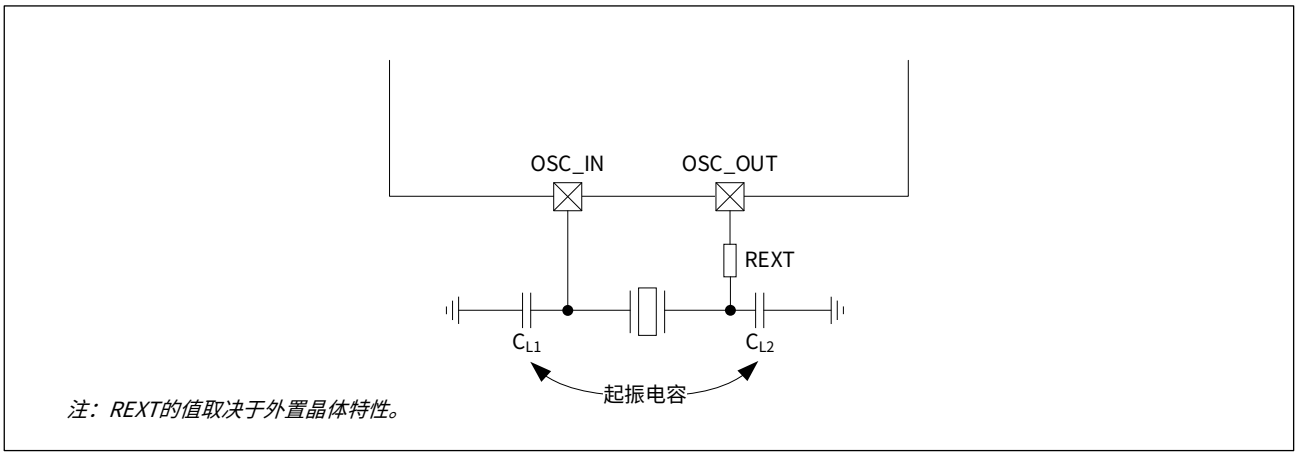
表 4-2 HSE 工作模式配置

工作模式	SYSCTRL_HSE.MODE	OSC_IN	OSC_OUT
石英晶体 / 陶瓷谐振器	0	模拟	模拟
外部时钟输入	1	数字输入	可作为通用 GPIO

石英晶体 / 陶瓷谐振器时钟模式

外置石英晶体 / 陶瓷谐振器接在 OSC_IN 和 OSC_OUT 两个引脚之间，配合起振电容，产生稳定的 4~32MHz 时钟信号，电路连接如下图所示：

图 4-2 HSE 外接石英晶体 / 陶瓷谐振器连接方式



石英晶体 / 陶瓷谐振器和起振电容须尽可能靠近芯片振荡器引脚放置，使得振荡器输出失真和启动时间达到最小。

起振电容取值大小须根据所选择的石英晶体 / 陶瓷谐振器进行调整，计算方法为：

$$C_L = (C_{L1} \times C_{L2}) / (C_{L1} + C_{L2}) + C_0$$

其中，

C_L 为晶体标称负载电容值；

C_0 为杂散电容值，跟电路板设计相关，一般可取值 4pF ~ 6pF；

C_{L1} 和 C_{L2} 为外接的 2 个起振电容，通常 C_{L1} 和 C_{L2} 相同。

如用户采用标称负载电容值为 12.5pF 的晶体谐振器，则晶体外接的 2 个起振电容常见取值为 15pF。

注：

由于晶体厂家众多，各厂家晶体特性各不相同，如果 HSE 振荡器输出时钟信号不理想，可联系晶体供货厂家协调处理。

为保证 HSE 振荡器的性能，需要根据外接晶体的标称频率来设置外置高频晶体控制寄存器 SYSCTRL_HSE 的 FREQRANGE 位域，如下表所示：

表 4-3 HSE 外接晶体频率设置

外接晶体标称频率	SYSCTRL_HSE.FREQRANGE
4MHz ~ 8MHz	00
8MHz ~ 16MHz	01
16MHz ~ 24MHz	10
24MHz ~ 32MHz	11

用户可以通过设置外置高频晶体控制寄存器 SYSCTRL_HSE 的 DRIVER 位域进行驱动能力调节，有四档驱动能力可选择，驱动能力越强功耗越大。用户可根据需要在时钟可靠性、启动时间以及低功率消耗三者之间取得平衡。

外部时钟输入模式

外部时钟输入模式下，外部时钟从 OSC_IN 引脚输入，OSC_OUT 引脚可以作为通用 GPIO 使用。输入的时钟信号可以是方波、正弦波或者三角波，占空比必须在 40% ~ 60% 之间，频率在 4 ~ 32MHz 之间。

HSE 两种工作模式下，HSE 时钟振荡器上电后均默认处于关闭状态，通过设置系统控制寄存器 SYSCTRL_CR1 的 HSEEN 位域为 1 启动。

HSE 振荡器启动后，当芯片内部时钟监控模块检测到一定数量的 HSE 时钟信号，则认为 HSE 时钟已稳定。检测时钟数量可通过外置高频晶体控制寄存器 SYSCTRL_HSE 的 WAITCYCLE 位域进行设置，如下表所示：

表 4-4 HSE 稳定检测时钟信号数量

SYSCTRL_HSE.WAITCYCLE	HSE 检测时钟数量
00	8192
01	32768
10	131072（默认值）
11	262144

通过外置高频晶体控制寄存器 SYSCTRL_HSE 的 STABLE 标志位，可确定 HSE 时钟的起振状态，STABLE 标志为 1 则 HSE 时钟已稳定，为 0 则表示 HSE 时钟还未稳定。

注：

HSE 振荡器应在启动前设置好所有参数，启动后禁止修改相关参数。

4.3.4 HSIOSC 时钟

HSIOSC 时钟由内部 RC 振荡器产生，不需要外部电路，比 HSE 时钟的成本低，启动速度快。HSIOSC 时钟频率固定为 48MHz，频率精度低于 HSE 时钟。

RC 振荡器输出时钟的频率受芯片加工过程、工作电压、环境温度等因素影响，CW32F020 提供了 HSIOSC 时钟频率校准功能，用户可通过设置内置高频时钟控制寄存器 SYSCTRL_HSI 的 TRIM 位域值来校准 HSIOSC 时钟频率，详情请参见 4.4.2 时钟校准。

HSIOSC 内部高速 RC 振荡器在芯片上电后，默认处于开启状态，用户可通过设置系统控制寄存器的 SYSCTRL_CR1 的 HSIEN 位域为 0 来关闭。如用户停止并重新启动了 HSIOSC 振荡器，可通过内置高频时钟控制寄存器 SYSCTRL_HSI 的 STABLE 标志位来确定 HSI 时钟是否稳定，STABLE 标志为 1 表示 HSIOSC 时钟已稳定，为 0 则表示 HSIOSC 时钟还未稳定。

注：

HSIOSC 振荡器应在启动前设置好所有参数，启动后禁止修改相关参数。

HSIOSC 时钟经过分频后输出 HSI 时钟，分频系数通过内置高频时钟控制寄存器 SYSCTRL_HSI 的 DIV 位域设置，有效分频系数为 1、2、4、6、8、10、12、14、16，上电后默认值为 6，所以 HSI 时钟默认频率为 8MHz。

4.3.5 LSE 时钟

外部低速时钟（LSE）支持两种工作模式：

- 石英晶体 / 陶瓷谐振器，设置外置低频晶体控制寄存器 SYSCTRL_LSE 的 MODE 位域为 0，同时配置 OSC32_IN、OSC32_OUT 引脚为模拟功能。
- 外部时钟输入，设置外置低频晶体控制寄存器 SYSCTRL_LSE 的 MODE 位域为 1，同时配置 OSC32_IN 引脚为数字输入功能。

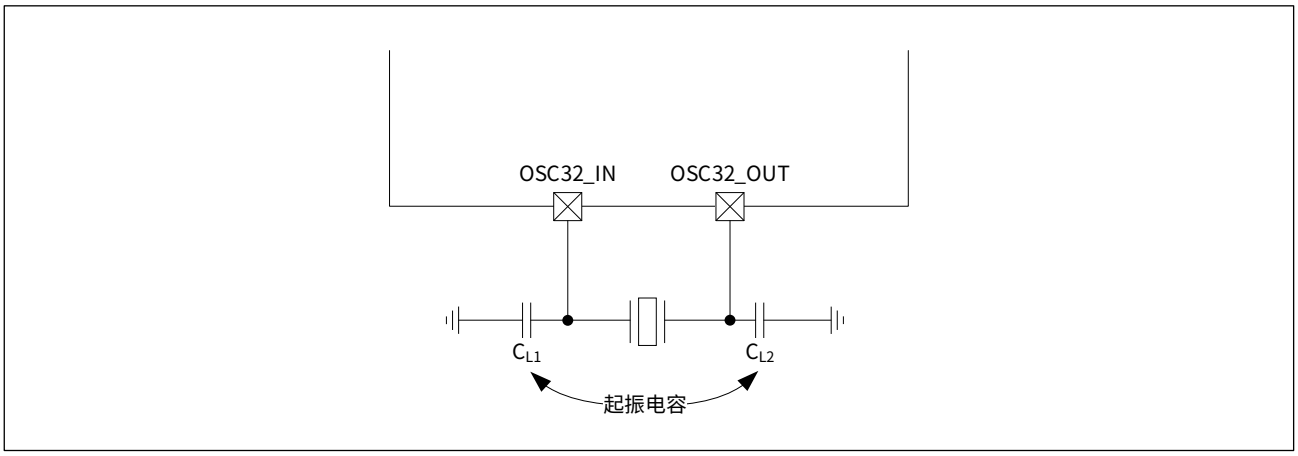
表 4-5 LSE 工作模式配置

工作模式	SYSCTRL_LSE.MODE	OSC32_IN	OSC32_OUT
石英晶体 / 陶瓷谐振器	0	模拟	模拟
外部时钟输入	1	数字输入	可作为通用 GPIO

石英晶体 / 陶瓷谐振器时钟模式

外部低频石英晶体 / 陶瓷谐振器接在 OSC32_IN 和 OSC32_OUT 两个引脚之间，配合起振电容，可产生最高 1MHz 的稳定时钟信号。LSE 通常作为系统实时时钟 RTC 的工作时钟，此时应选用频率为 32.768kHz 的石英晶体 / 陶瓷谐振器。外部电路连接如下图所示：

图 4-3 LSE 外接石英晶体 / 陶瓷谐振器连接方式



石英晶体 / 陶瓷谐振器和起振电容须尽可能靠近芯片振荡器引脚放置，使得振荡器输出失真和启动时间达到最小。

起振电容取值大小须根据所选择的石英晶体 / 陶瓷谐振器进行调整，计算方法为：

$$C_L = (C_{L1} \times C_{L2}) / (C_{L1} + C_{L2}) + C_0$$

其中，

C_L 为晶体标称负载电容值；

C_0 为杂散电容值，跟电路板设计相关，一般可取值 4pF ~ 6pF；

C_{L1} 和 C_{L2} 为外接的 2 个起振电容，通常 C_{L1} 和 C_{L2} 相同。

如用户采用标称负载电容值为 12.5pF 的晶体谐振器，则晶体外接的 2 个起振电容常见取值为 15pF。

注：

由于晶体厂家众多，各厂家晶体特性各不相同，如果 LSE 振荡器输出时钟信号不理想，可联系晶体供货厂家协调处理。

用户可以通过设置外置低频晶体控制寄存器 SYSCTRL_LSE 的 DRIVER 位域进行驱动能力调节，有四档驱动能力可选择，驱动能力越强功耗越大。通过设置外置低频晶体控制寄存器 SYSCTRL_LSE 的 AMP 位域可调节时钟信号幅度，有四档幅度可选择，幅度越大功耗越大。用户可根据需要在时钟可靠性、启动时间以及低功率消耗三者之间取得平衡。

外部时钟输入模式

外部时钟输入模式下，外部时钟从 OSC32_IN 引脚输入，OSC32_OUT 引脚可以作为通用 GPIO 使用。输入的时钟信号可以是方波、正弦波或者三角波，占空比必须在 45% ~ 55% 之间，频率最高为 1MHz。

LSE 两种工作模式下，LSE 时钟振荡器上电后均默认处于关闭状态，通过设置系统控制寄存器 SYSCTRL_CR1 的 LSEEN 位域为 1 来启动。

LSE 振荡器启动后，当芯片内部时钟监控模块检测到一定数量的 LSE 时钟信号，则认为 LSE 时钟已稳定。检测时钟数量可通过外置低频晶体控制寄存器 SYSCTRL_LSE 的 WAITCYCLE 位域进行设置，如下表所示：

表 4-6　LSE 稳定检测时钟信号数量

SYSCTRL_LSE.WAITCYCLE	LSE 检测时钟数量
00	256
01	1024
10	4096（默认值）
11	16384

通过外置低频晶体控制寄存器 SYSCTRL_LSE 的 STABLE 标志位，可确定 LSE 时钟的起振状态，STABLE 标志为 1 表示 LSE 时钟已稳定，为 0 则表示 LSE 时钟还未稳定。

注：

LSE 振荡器应在启动前设置好所有参数，启动后禁止修改相关参数。

LSE 振荡器具有锁定功能，锁定功能开启后将禁止软件对 LSE 时钟振荡器的使能位（SYSCTRL_CR1.LSEEN）进行清零操作。即，LSE 一旦开启，不能由软件关闭。锁定功能主要为满足部分特殊应用场景需求而设计，例如，对 RTC 时间丢失敏感的应用场景，可有效防止误操作导致的 RTC 时间丢失。该锁定功能默认关闭，通过设置系统控制寄存器 SYSCTRL_CR1 的 LSELOCK 位域为 1 启动。

4.3.6 LSI 时钟

LSI 时钟由内部低速 RC 振荡器产生，默认频率为 32.8kHz。内部低速 RC 振荡器不需要外部电路，比 LSE 时钟的成本低，但精度低于 LSE 时钟。

RC 振荡器输出时钟的频率受芯片加工过程、工作电压、环境温度等因素影响，CW32F020 提供了 LSI 时钟频率校准功能。用户可通过设置内置低频时钟控制寄存器 SYSCTRL_LSI 的 TRIM 位域值来校准 LSI 时钟频率，详情请参见 4.4.2 时钟校准。

LSI 内部低速 RC 振荡器默认处于关闭状态，通过设置系统控制寄存器 SYSCTRL_CR1 的 LSIEN 位域为 1 启动。LSI 振荡器启动后，芯片内部时钟监控模块检测到一定数量的 LSI 时钟信号，则认为 LSI 时钟已稳定。检测时钟数量可通过内置低频振荡器控制寄存器 SYSCTRL_LSI 的 WAITCYCLE 位域进行设置，如下表所示：

表 4-7 LSI 稳定检测时钟信号数量

SYSCTRL_LSI.WAITCYCLE	LSI 检测时钟数量
00	6（默认值）
01	18
10	66
11	258

通过内置低频时钟控制寄存器 SYSCTRL_LSI 的 STABLE 标志位，可确定 LSI 时钟是否稳定，STABLE 标志为 1 表示 LSI 时钟已稳定，为 0 则表示 LSI 时钟还未稳定。

注：

LSI 振荡器在启动前必须设置好所有参数，启动后禁止修改相关参数。

4.3.7 PLL 时钟

CW32F020 内部集成锁相环 PLL 电路，可对输入时钟源进行锁相倍频输出 PLL 时钟。用户可通过内置锁相环控制寄存器 SYSCTRL_PLL 的 SOURCE 位域选择 PLL 的输入参考时钟源，如下表所示：

表 4-8 PLL 时钟源选择

SYSCTRL_PLL.SOURCE	PLL 输入参考时钟源
00	HSE 振荡器时钟 [1]
01	HSE 引脚输入时钟 [1]
11	HSI 时钟 (HSIOSC 分频后的时钟)

注 1：

必须与 SYSCTRL_HSE.MODE 保持一致。

锁相环倍频系数通过内置锁相环控制寄存器 SYSCTRL_PLL 的 MUL 位域进行设置，可设置范围为 2 ~ 12，默认值为 8。

为保证锁相环的锁定收敛速度及输出时钟相噪性能，用户需根据实际的输入参考时钟频率和输出时钟频率分别设置 SYSCTRL_PLL.FREQIN 和 SYSCTRL_PLL.FREQOUT 位域的值。设置规则如下表所示：

表 4-9 锁相环输入参考时钟频率

输入参考时钟频率	SYSCTRL_PLL.FREQIN
4MHz ~ 6MHz	00
6MHz ~ 12MHz	01
12MHz ~ 20MHz	10
20MHz ~ 24MHz	11

表 4-10 锁相环输出时钟频率

需要输出时钟频率	SYSCTRL_PLL.FREQOUT
12MHz ~ 18MHz	00
18MHz ~ 24MHz	01
24MHz ~ 36MHz	10
36MHz ~ 48MHz	11
48MHz ~ 72MHz	1xx

PLL 默认处于关闭状态，通过设置系统控制寄存器 SYSCTRL_CR1 的 PLLEN 位域为 1 来启动。PLL 启动后，芯片内部时钟监控模块检测到一定数量的 PLL 时钟信号，则认为 PLL 时钟已稳定。检测时钟数量可通过内置锁相环控制寄存器 SYSCTRL_PLL 的 WAITCYCLE 位域进行设置，如下表所示：

表 4-11 PLL 稳定检测时钟信号数量

SYSCTRL_PLL.WAITCYCLE	PLL 检测时钟数量
000	128
001	256
010	512
011	1024（默认值）
100	2048
101	4096
110	8192
111	16384

通过内置锁相环控制寄存器 SYSCTRL_PLL 的 STABLE 标志位，可确定 PLL 时钟是否稳定，STABLE 标志为 1 表示 PLL 时钟已稳定，为 0 则表示 PLL 时钟还未稳定。

注：

PLL 在启动前必须设置好所有参数，启动后禁止修改相关参数。

修改 PLL 参数流程如下：

步骤 1：设置 SYSCTRL_CR1.PLLEN 为 0，关闭 PLL；

步骤 2：等待 SYSCTRL_PLL.STABLE 标志被系统硬件清零；

步骤 3：更改 PLL 的参数；

设置 SYSCTRL_CR1.PLLEN 为 1，启动 PLL；

步骤 4：等待 SYSCTRL_PLL.STABLE 标志被系统硬件置 1，标识 PLL 时钟已稳定。

4.3.8 SysClk 系统时钟

系统时钟 SysClk 可选 5 种时钟源，包括 HSE、LSE、HSI、LSI、PLL。

系统上电复位完成后默认选择 HSI 作为 SysClk 的时钟源，时钟频率默认值是 8MHz。

用户可通过系统控制寄存器 SYSCTRL_CR0 的 SYSCLK 位域选择系统时钟源，当选择某一时钟作为系统时钟源后，用户无法通过设置该时钟电路的使能位 SYSCTRL_CR1.xxxEN 为 0 来停止该时钟。

4.3.9 片内外设时钟控制

片内外设一般都需要有配置时钟和工作时钟，配置时钟用来响应 CPU 对外设寄存器的读写操作，工作时钟用于各外设的功能实现（如 UART 的传输时钟，定时器的计数时钟等）。

在使用外设前都必须打开外设的配置时钟和工作时钟，否则外设无法工作。通过设置 AHB 外设时钟使能控制寄存器 SYSCTRL_AHBEN、APB 外设时钟使能控制寄存器 SYSCTRL_APBEN1 和 SYSCTRL_APBEN2 的对应位为 1，打开对应外设的配置时钟和工作时钟。

RTC、UART、AWT、FLASH 等外设只打开配置时钟，工作时钟通过各模块的时钟源选择寄存器进行配置。

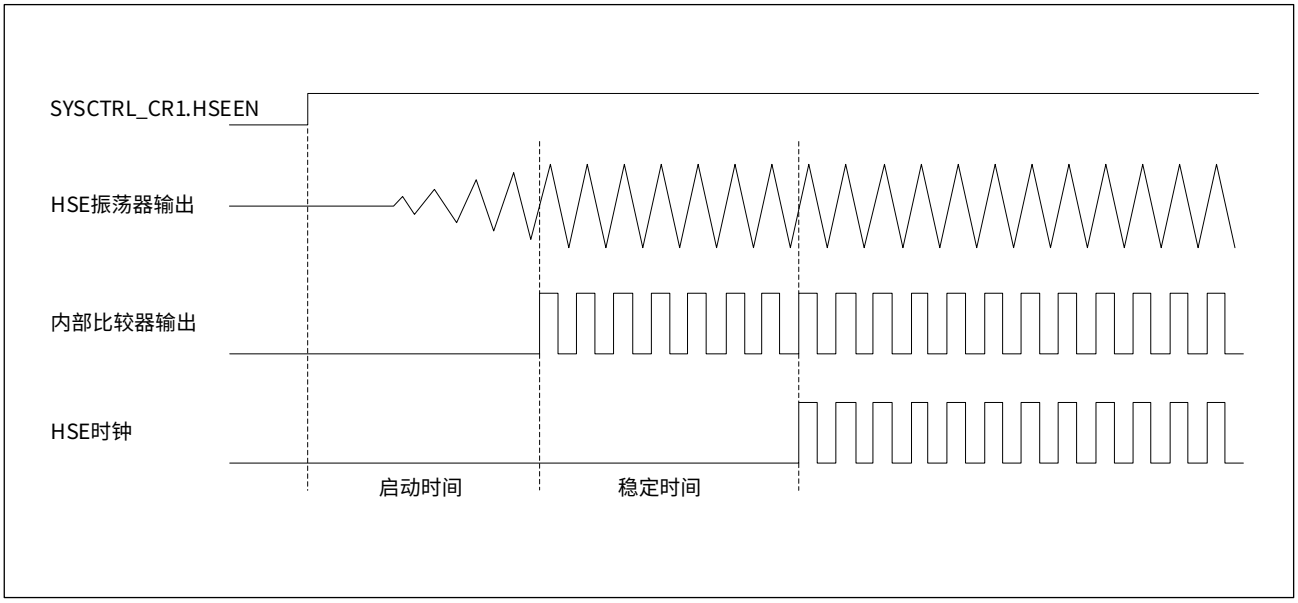
当外设不需要使用时，通过关闭外设的配置时钟和工作时钟禁止外设，能有效降低芯片功耗。

4.4 时钟启动、校准与状态检测

4.4.1 时钟启动

CW32F020 的时钟源启动过程类似，以 HSE 为例说明时钟启动后的稳定过程，如下图所示：

图 4-4 HSE 时钟启动过程



当设置 SYSCTRL_CR1.HSEEN 为 1 后，HSE 时钟振荡电路开始工作，但此时输出的时钟信号振幅很小。经过启动时间阶段后，输出时钟信号的振幅、占空比等可满足内部采样电路需求，进入稳定时间阶段。在稳定时间阶段，芯片内部时钟监控电路对 HSE 输出的时钟信号进行计数，当计数值达到设定的个数后，认为 HSE 时钟信号已稳定，HSE 时钟稳定标志位 SYSCTRL_HSE.STABLE 被置 1；如果在一定时间内未检测到 SYSCTRL_HSE.WAITCYCLE 个时钟信号则认为 HSE 振荡器起振失败。

其它时钟振荡器的时钟启动过程类似，但注意各时钟振荡器在稳定阶段检测时钟数量以及起振失败检测的检测时间和检测时钟数量等都不相同，具体参见各时钟振荡器相关描述。

4.4.2 时钟校准

时钟校准主要针对 HSIOSC 时钟和 LSI 时钟。通过调整振荡器的 TRIM 值来实现时钟频率校准。

HSIOSC 时钟校准

HSIOSC 安全工作范围为 32 ~ 48MHz，超过安全范围，芯片可能出现异常。芯片出厂时已预调好 48MHz 频点的校准参数，并存放在 FLASH 中。应用程序只需要将 FLASH 内的校准值读出并写入 SYSCTRL_HSI.TRIM 即可获得精准的 48MHz 时钟。48MHz 频率校准值存放地址为：0x0001 2600 – 0x0001 2601。如需其它频率的时钟则需要用户自行调整 SYSCTRL_HSI.TRIM 的值。

LSI 时钟校准

LSI 输出时钟频率范围为 32.8kHz±10%，如果将 LSI 的输出时钟频率调节到此范围外，则该时钟有可能异常。芯片出厂时已预调好 32.8kHz 频点的校准参数，并存放在 FLASH 中，应用程序只需要将 FLASH 内的校准值读出并写入 SYSCTRL_LSI.TRIM 即可获得精准的 32.8kHz 时钟。32.8kHz 频率校准值存放地址：0x0001 2602 - 0x0001 2603。如需其它频率的时钟则需要用户自行调整 SYSCTRL_LSI.TRIM 的值。

4.4.3 时钟状态检测

芯片具备时钟启动过程中的时钟稳定检测、起振失败检测，以及时钟运行中失效检测功能，并支持在当前选定的系统时钟源故障后自动进行时钟源切换。

4.4.3.1 时钟稳定检测

HSE、LSE、HSIOSC、LSI、PLL 这 5 种时钟源都支持时钟稳定检测功能，用户可通过对应时钟源的稳定标志位来确定时钟状态。时钟稳定标志在关闭时钟源时由硬件清 0，在时钟源启动并稳定后由硬件置 1。

注意时钟稳定标志只针对时钟启动过程而言，在时钟稳定运行过程中，检测到时钟运行失效不会影响该时钟稳定标志。

以 HSE 时钟源为例，针对 HSE 时钟稳定标志和时钟稳定中断标志，说明如下：

- SYSCTRL_HSE.STABLE

  硬件在关闭 HSE 时清零，硬件在检测到 HSE 时钟稳定时置位。

- SYSCTRL_ISR.HSESTABLE

  和 SYSCTRL_HSE.STABLE 一样，属于同一个信号。该信号虽然位于中断标志寄存器 SYSCTRL_ISR 中，但被硬件置 1 后并不会引起中断请求。

- SYSCTRL_ISR.HSERDY

  当硬件检测到时钟由不稳定状态变为稳定状态时（即 SYSCTRL_HSE.STABLE 标志位由 0 变为 1）置 1，为时钟稳定中断标志，用户可通过设置 SYSCTRL_ICR.HSERDY 为 0 清除该标志位。

4.4.3.2 时钟起振失败检测

CW32F020 支持外部时钟（HSE 和 LSE）起振失败检测功能。启动外部时钟后，时钟检测逻辑对所检测的时钟信号进行计数，在检测时间内检测到设定个数的时钟信号则时钟起振成功，否则起振失败。

HSE 起振失败检测的时钟数量通过 SYSCTRL_HSE.WAITCYCLE 设置，检测时间无需用户设置，由系统自动设定，如下表所示：

表 4-12 HSE 起振失败检测时间及检测时钟数量

SYSCTRL_HSE.WAITCYCLE	检测时间	检测时钟数量
00	65ms	8192
01	65ms	32768
10	65ms	131072
11	130ms	262144

LSE 起振失败检测的时钟数量通过 SYSCTRL_LSE.WAITCYCLE 设置，检测时间无需用户设置，由系统自动设定，如下表所示：

表 4-13 LSE 起振失败检测时间及检测时钟数量

SYSCTRL_LSE.WAITCYCLE	检测时间	检测时钟数量
00	1s	256
01	1s	1024
10	1s	4096
11	2s	16384

外部时钟（HSE 和 LSE）故障检测功能默认处于关闭状态，此时 HSE/LSE 起振失败不会产生相应的中断标志。通过设置 SYSCTRL_CR1.HSECCS 为 1 和 SYSCTRL_CR1.LSECCS 为 1 分别使能 HSE 和 LSE 外部时钟的故障检测功能。

使能外部时钟的故障检测功能后，如果 HSE 或者 LSE 起振失败，会产生起振失败中断标志（对应 SYSCTRL_ISR.HSEFAIL 或者 SYSCTRL_ISR.LSEFAIL 被置为 1）；如果中断使能（对应 SYSCTRL_IER.HSEFAIL 或者 SYSCTRL_IER.LSEFAIL 设置为 1），则 CPU 会执行中断服务程序进行时钟起振失败处理。

4.4.3.3 时钟运行中失效检测

CW32F020 支持外部时钟（HSE 和 LSE）运行中失效检测功能。在外部时钟稳定运行过程中，时钟检测逻辑持续以一定的检测周期对 HSE 和 LSE 时钟信号进行计数：在检测周期内检测到设定个数的时钟信号则运行正常，否则运行失效。

HSE 时钟的运行中失效检测周期通过 SYSCTRL_HSE.DETCNT 配置，实际时间为 SYSCTRL_HSE.DETCNT/f_{LSI}，检测时钟个数为 0x20000。考虑到时钟都有一定的偏差，为保证检测功能可靠，HSE 检测周期参数 HSE.DETCNT 的配置须留有一定的裕量，一般根据 HSE 的运行频率，配置为 8000/f_{HSE}（其中 f_{HSE} 为 HSE 时钟的频率，单位为 MHz）。

例：如果 HSE 时钟频率为 4MHz，则 SYSCTRL_HSE.DETCNT 应配置为 2000。

LSE 时钟的运行中失效检测周期不可配置，固定为 256 个 LSI 时钟周期，检测时钟个数为 128。

外部时钟故障检测功能默认处于关闭状态，此时 HSE/LSE 运行中失效不会产生相应的中断标志。通过设置 SYSCTRL_CR1.HSECCS 为 1 和 SYSCTRL_CR1.LSECCS 为 1，分别使能 HSE 和 LSE 时钟故障检测功能。HSE 或 LSE 时钟运行中失效检测还需要设置 SYSCTRL_CR1.LSIEN 为 1 使能内部低速时钟 LSI。

使能外部时钟的故障检测功能后，如果运行过程中检测到 HSE 或者 LSE 失效，则会产生运行中失效标志（对应 SYSCTRL_ISR.HSEFAULT 或者 SYSCTRL_ISR.LSEFAULT 被置 1），如果中断使能（对应 SYSCTRL_IER.HSEFAULT 或者 SYSCTRL_IER.LSEFAULT 设置为 1），则 CPU 会执行中断服务程序进行时钟运行中失效处理。

如果当前系统时钟来源为 HSE，则当 HSE 出现运行中失效时系统时钟源会自动切换到 HSI 时钟。

如果当前系统时钟来源为 LSE，则当 LSE 出现运行中失效时系统时钟源会自动切换到 HSI 时钟。

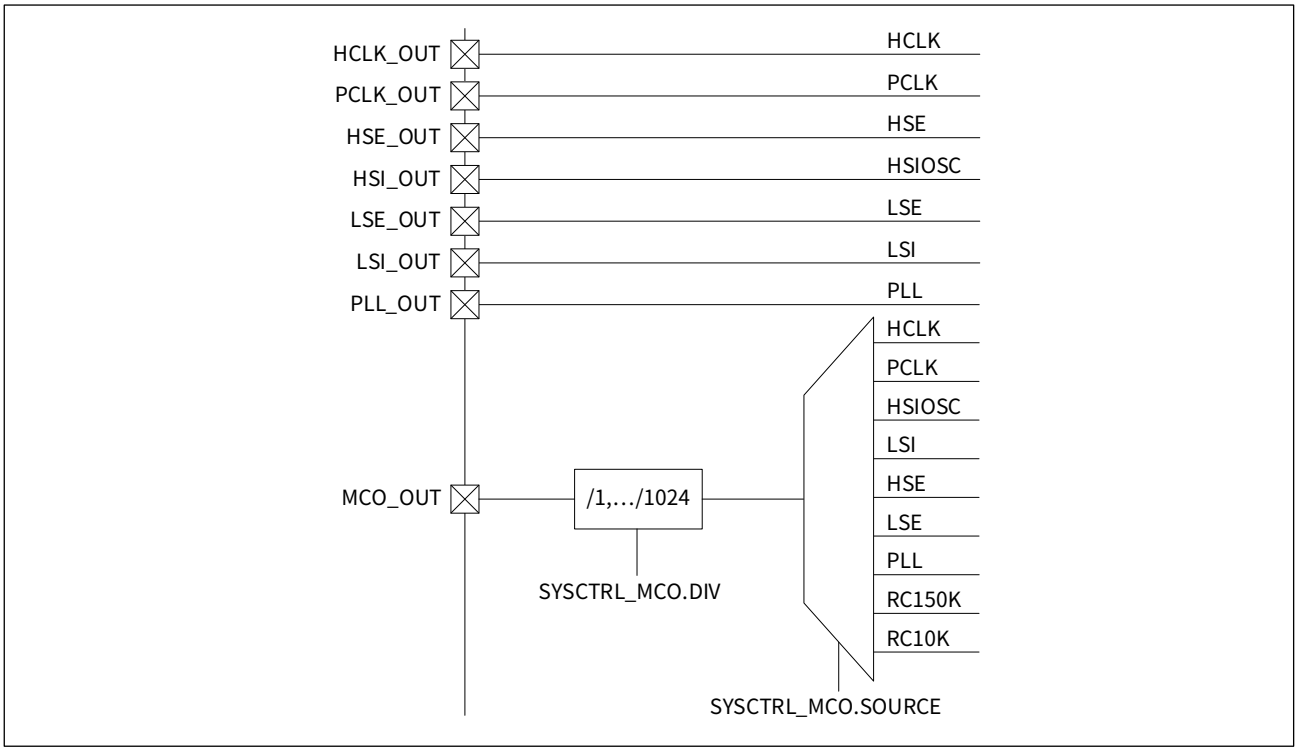
4.4.4 时钟验证与输出

CW32F020 支持将内部各种时钟信号输出到外部引脚。用户可利用该功能对当前系统的 CPU 运行频率、系统总线频率、外设工作频率等进行测量。

- HCLK_OUT 引脚
  输出 HCLK 时钟信号
- PCLK_OUT 引脚
  输出 PCLK 时钟信号
- HSE_OUT
  输出 HSE 时钟信号
- HSI_OUT
  输出 HSIOSC 时钟信号
- LSE_OUT
  输出 LSE 时钟信号
- LSI_OUT
  输出 LSI 时钟信号
- PLL_OUT
  输出 PLL 时钟信号
- MCO_OUT 引脚
  输出 HCLK/PCLK/HSIOSC/LSI/HSE/LSE/PLL/RC150K/RC10K 时钟信号，时钟输出到 MCO_OUT 引脚前可通过预分频器进行分频（有效分频系数为 1、2、8、64、128、256、512、1024），以便低带宽仪表能准确测量信号。

时钟输出框图如下图所示：

图 4-5 时钟输出



4.5 SysClk 系统时钟切换

系统时钟 SysClk 可选择 5 种时钟源，包括 HSE、LSE、PLL、HSI、LSI，通过对系统控制寄存器 SYSCTRL_CR0 的 SYSCLK 位域进行设置，可在不同时钟源之间进行切换。

主要应用场景包括：

- 时钟故障自动切换

  当检测到当前系统时钟源失效时，可快速安全切换到其它可用时钟源上，提高产品可靠性。

- 功耗管理

  在待机状态切换 SysClk 的时钟源为低速时钟降低系统功耗，在正常使用状态切换 SysClk 的时钟源为高速时钟来快速响应用户使用需求。

时钟源安全切换规则如下：

- HSE、LSE、HSI、LSI 四个时钟源中的任意两个之间可以相互切换
- PLL 只可与 HSE、HSI 这两个时钟进行相互切换

系统时钟切换操作必须按照下文所描述的 7 种时钟切换流程进行，否则可能出现异常。

注：

系统时钟切换时需要同步配置 FLASH 控制寄存器 FLASH_CR2.WAIT 读等待周期参数：系统时钟频率不大于 24MHz 则应设置 FLASH 控制寄存器 FLASH_CR2.WAIT 为 0；系统时钟频率大于 24MHz 则应设置 FLASH 控制寄存器 FLASH_CR2.WAIT 为 1；系统时钟频率大于 48MHz 则应设置 FLASH 控制寄存器 FLASH_CR2.WAIT 为 2。

4.5.1 标准的时钟切换流程

标准时钟切换操作流程如下：

步骤 1：如果新时钟源为 HSE 或者 LSE，则需要配置对应的外部引脚为适当的模式：接外部晶体时需要配置为模拟功能，接外部时钟输入时需要将 GPIO 设置为数字输入功能并使能端口的输入功能；

步骤 2：配置新时钟源的振荡参数，如驱动能力、振荡幅度、稳定周期等；

步骤 3：设置新时钟源振荡器使能位为 1；

步骤 4：根据当前时钟源和新时钟源两者中较高的频率，按 7 FLASH 存储器章节要求配置 FLASH_CR2.WAIT 读等待周期参数；

步骤 5：等待新时钟源输出稳定的频率（新时钟源的 STABLE 信号变为 1）；

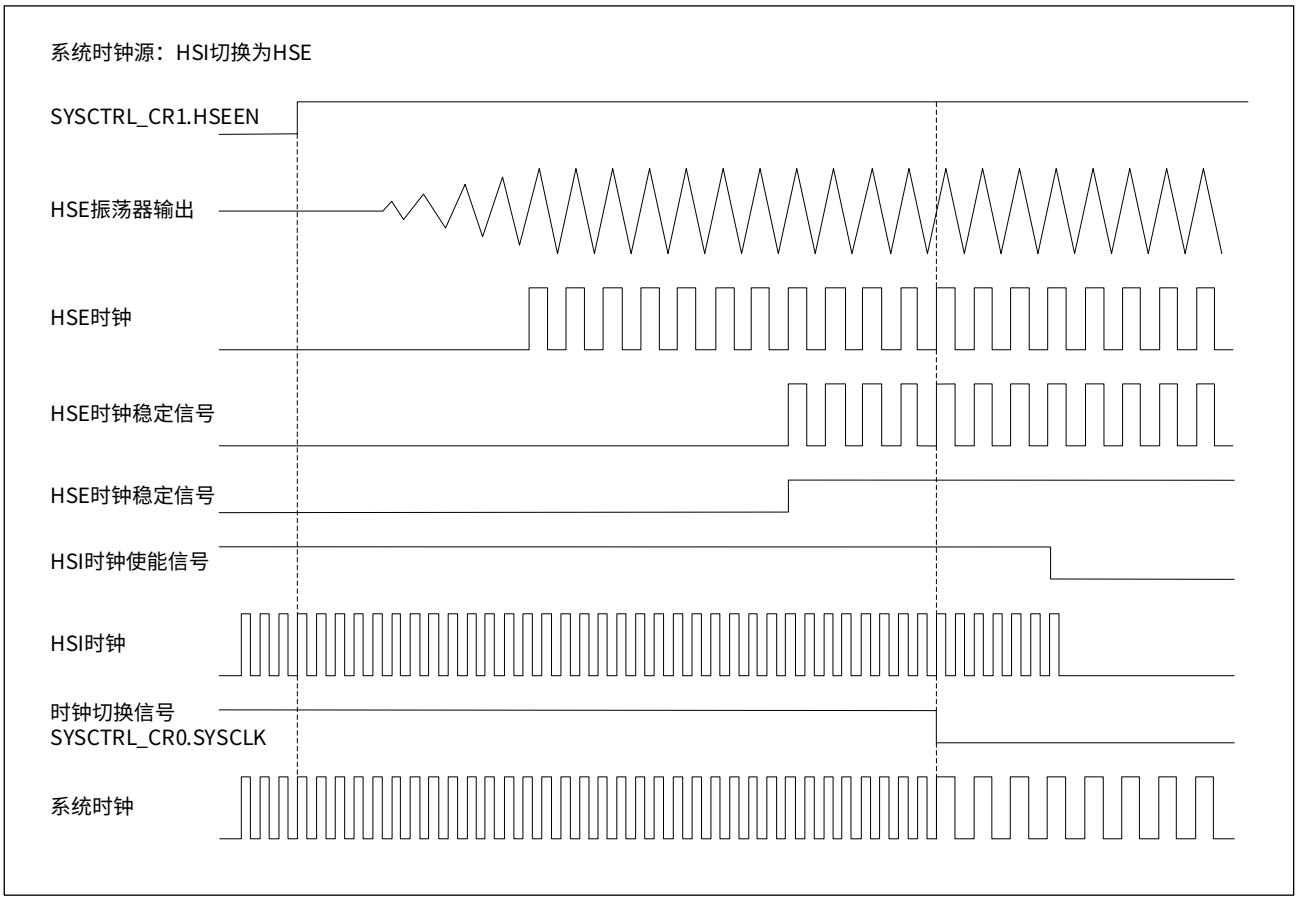
步骤 6：配置系统控制寄存器 SYSCTRL_CR0.SYSCLK，选择系统时钟 SysClk 的时钟源为新时钟源；

步骤 7：根据新时钟源的频率，按 7 FLASH 存储器章节要求配置 FLASH_CR2.WAIT 读等待周期参数；

步骤 8：关闭不再使用的时钟源。

以系统时钟从 HSI 切换为 HSE 为例，新旧时钟的切换过程如下图所示：

图 4-6 HSI 时钟切换为 HSE 时钟



4.5.2 HSI 时钟不同频率间切换流程

HSI 时钟不同频率切换不需要启动新时钟源，只需要改变 HSI 时钟预分频器的分频比即可，切换速度快，切换前后无需等待时钟稳定时间。切换操作流程如下：

步骤 1：根据当前时钟源和新时钟源两者中较高的频率，按 7 FLASH 存储器章节要求配置 FLASH_CR2.WAIT 读等待周期参数；

步骤 2：读取内置高频时钟控制寄存器 SYSCTRL_HSI 寄存器的值为 Value；

步骤 3：根据需要的目标频率，设置内置高频时钟控制寄存器 SYSCTRL_HSI.DIV 为对应的分频系数（注意 SYSCTRL_HSI.TRIM 域要保持不变）；

步骤 4：根据新时钟源的频率，按 7 FLASH 存储器章节要求配置 FLASH_CR2.WAIT 读等待周期参数。

4.5.3 从其它时钟到切换到 LSE 示例

切换操作流程如下：

步骤 1：设置 PC14 / PC15 引脚为模拟功能端口；

步骤 2：根据 LSE 振荡器外接晶体特性，配置新时钟源 LSE 的振荡参数，如驱动能力（通过外置低频晶体控制寄存器 SYSCTRL_LSE.DRIVER 设置，建议配置为 3）、振荡幅度 (通过外置低频晶体控制寄存器 SYSCTRL_LSE.AMP 设置，建议配置为 2 或 3)、稳定周期 (通过外置低频晶体控制寄存器 SYSCTRL_LSE.WAITCYCLE 设置，建议设置为 3) 等；

步骤 3：设置系统控制寄存器 SYSCTRL_CR1.LSEEN 为 1，使能 LSE 晶体振荡电路；

注：

SYSCTRL_CR1 寄存器具有 KEY 保护特性，写入的数据高 16bit 数据必须是 0x5A5A，否则无法写入。

步骤 4：循环查询并等待外置低频晶体控制寄存器 SYSCTRL_LSE.STABLE 稳定标志变为 1，即等待 LSE 晶体振荡电路输出稳定时钟；

步骤 5：设置系统控制寄存器 SYSCTRL_CR0.SYSCLK 为 4，将 SysClk 时钟来源切换为 LSE；

注：

SYSCTRL_CR0 寄存器具有 KEY 保护特性，写入的数据高 16bit 数据必须是 0x5A5A，否则无法写入。

步骤 6：按 7 FLASH 存储器章节要求配置 FLASH_CR2.WAIT 读等待周期参数；

步骤 7：设置系统控制寄存器 SYSCTRL_CR1.xxxEN 为 0，关闭原时钟。

4.5.4 从其它时钟到切换到 HSE 示例

切换操作流程如下：

步骤 1：设置 PF00/PF01 引脚为模拟功能端口；

步骤 2：根据 HSE 振荡器外接晶体特性，配置新时钟源 HSE 的振荡参数，如晶体工作频率（通过外置高频晶体控制寄存器 SYSCTRL_HSE.FREQRANGE 配置）、驱动能力（通过外置高频晶体控制寄存器 SYSCTRL_HSE.DRIVER 配置）、稳定周期 (通过外置高频晶体控制寄存器 SYSCTRL_HSE.WAITCYCLE 配置，建议设置为 3) 等；

步骤 3：设置系统控制寄存器 SYSCTRL_CR1.HSEEN 为 1，使能 HSE 晶体振荡电路；

注：

SYSCTRL_CR1 寄存器具有 KEY 保护特性，写入的数据高 16bit 数据必须是 0x5A5A，否则无法写入。

步骤 4：根据当前时钟和 HSE 两者中较高的频率，按 7 FLASH 存储器章节要求配置 FLASH_CR2.WAIT 读等待周期参数；

步骤 5：循环查询并等待外置高频晶体控制寄存器 SYSCTRL_HSE.STABLE 稳定标志变为 1 后，确保晶体振荡电路输出稳定时钟；

步骤 6：设置系统控制寄存器 SYSCTRL_CR0.SYSCLK 为 1，将 SysClk 时钟来源切换为 HSE；

注：

SYSCTRL_CR0 寄存器具有 KEY 保护特性，写入的数据高 16bit 数据必须是 0x5A5A，否则无法写入。

步骤 7：根据 HSE 的时钟频率，按 7 FLASH 存储器章节要求配置 FLASH_CR2.WAIT 读等待周期参数；

步骤 8：设置系统控制寄存器 SYSCTRL_CR1.xxxEN 为 0，关闭原时钟。

4.5.5 从其它时钟到切换到 LSI 示例

切换操作流程如下：

步骤 1：配置内置低频时钟控制寄存器 SYSCTRL_LSI.TRIM 及内置低频时钟控制寄存器 SYSCTRL_LSI.WAITCYCLE 为合适的值；

步骤 2：设置系统控制寄存器 SYSCTRL_CR1.LSIEN 为 1，使能 LSIRC 振荡电路；

注：

SYSCTRL_CR1 寄存器具有 KEY 保护特性，写入的数据高 16bit 数据必须是 0x5A5A，否则无法写入。

步骤 3：循环查询并等待内置低频时钟控制寄存器 SYSCTRL_LSI.STABLE 稳定标志变为 1，即等待 LSI 输出稳定时钟；

步骤 4：设置系统控制寄存器 SYSCTRL_CR0.SYSCLK 为 3，将 SysClk 时钟来源切换为 LSI；

注：

SYSCTRL_CR0 寄存器具有 KEY 保护特性，写入的数据高 16bit 数据必须是 0x5A5A，否则无法写入。

步骤 5：配置 FLASH_CR2.WAIT 读等待周期参数为 0；

步骤 6：设置系统控制寄存器 SYSCTRL_CR1.xxxEN 为 0，关闭原时钟。

4.5.6 从其它时钟到切换到 HSI 示例

切换操作流程如下：

步骤 1：配置内置高频时钟控制寄存器 SYSCTRL_HSI.TRIM、SYSCTRL_HSI.WAITCYCLE 以及 SYSCTRL_HSI.DIV 为合适的值；

步骤 2：设置系统控制寄存器 SYSCTRL_CR1.HSIEN 为 1，使能 HSIOSC 时钟振荡电路；

注：

SYSCTRL_CR1 寄存器具有 KEY 保护特性，写入的数据高 16bit 数据必须是 0x5A5A，否则无法写入。

步骤 3：循环查询并等待内置高频时钟控制寄存器 SYSCTRL_HSI.STABLE 稳定标志变为 1，即等待 HSIOSC 输出稳定时钟；

步骤 4：根据当前时钟和 HSI 两者中较高的频率，按 7 FLASH 存储器章节要求配置 FLASH_CR2.WAIT 读等待周期参数；

步骤 5：设置系统控制寄存器 SYSCTRL_CR0.SYSCLK 为 0，将 SysClk 时钟来源切换为 HSI；

注：

SYSCTRL_CR0 寄存器具有 KEY 保护特性，写入的数据高 16bit 数据必须是 0x5A5A，否则无法写入。

步骤 6：根据 HSI 的时钟频率，按 7 FLASH 存储器章节要求配置 FLASH_CR2.WAIT 读等待周期参数；

步骤 7：设置系统控制寄存器 SYSCTRL_CR1.xxxEN 为 0，关闭原时钟。

4.5.7 PLL 与 HSI 相互切换示例，PLL 的输入时钟源为 HSI

从 HSI 切换到 PLL 操作流程如下：

步骤 1：设置锁相环控制寄存器 SYSCTRL_PLL.SOURCE 为 3，选择 HSI 时钟作为 PLL 的输入时钟时钟源；

步骤 2：根据 HSIOSC 的输出时钟频率以及 SYSCTRL_HSI.DIV 位域的值，计算 HSI 时钟频率；根据 HSI 时钟频率值，配置锁相环输入频率范围 SYSCTRL_PLL.FREQIN；

步骤 3：根据需要 PLL 输出的时钟频率，配置锁相环倍频系数 SYSCTRL_PLL.MUL 及输出时钟频率范围 SYSCTRL_PLL.FREQOUT；

步骤 4：设置锁相环稳定周期 SYSCTRL_PLL.WAITCYCLE 为 7，选择最长的 PLL 稳定时间；

步骤 5：设置系统控制寄存器 PLL 使能位 SYSCTRL_CR1.PLLEN 为 1，使能 PLL 锁相环电路；

注：

SYSCTRL_CR1 寄存器具有 KEY 保护特性，写入的数据高 16bit 数据必须是 0x5A5A，否则无法写入。

步骤 6：根据 PLL 输出时钟频率，按 7 FLASH 存储器章节要求配置 FLASH_CR2.WAIT 读等待周期参数；

步骤 7：循环查询并等待锁相环控制寄存器 SYSCTRL_PLL.STABLE 稳定标志变为 1，即等待 PLL 输出稳定时钟；

步骤 8：设置系统控制寄存器 SYSCTRL_CR0.SYSCLK 为 2，将 SysClk 时钟来源切换为 PLL 时钟。

注：

1、SYSCTRL_CR0 寄存器具有 KEY 保护特性，写入的数据高 16bit 数据必须是 0x5A5A，否则无法写入。

2、时钟源切换过程中禁止更改高频时钟控制寄存器 SYSCTRL_HSI 的值。

从 PLL 切换到 HSI 操作流程如下：

本示例中，由于锁相环的输入参考时钟为 HSI，因此 HSI 时钟已经是处于稳定状态，故不需要先使能 HSIOSC 并等待 HSIOSC 及 HSI 时钟稳定，可直接切换。如果 PLL 的输入参考时钟是 HSE，则需要增加设置 HSIOSC 参数、使能 HSIOSC，然后等待 HSIOSC 以及 HSI 时钟稳定过程。

步骤 1：设置 SYSCTRL_CR0.SYSCLK 为 0，将 SysClk 时钟来源切换为 HSI；

注：

SYSCTRL_CR0 寄存器具有 KEY 保护特性，写入的数据高 16bit 数据必须是 0x5A5A，否则无法写入。

步骤 2：根据 HSI 时钟的频率，按 7 FLASH 存储器章节要求配置 FLASH_CR2.WAIT 读等待周期参数；

步骤 3：设置系统控制寄存器 SYSCTRL_CR1.PLLEN 为 0，关闭 PLL 时钟。

注：

1、SYSCTRL_CR1 寄存器具有 KEY 保护特性，写入的数据高 16bit 数据必须是 0x5A5A，否则无法写入。

2、时钟源切换过程中禁止更改高频时钟控制寄存器 SYSCTRL_HSI 的值。

4.5.8 PLL 与 HSE 相互切换示例，PLL 的输入时钟源为 HSE

从 HSE 切换到 PLL 操作流程如下：

步骤 1：设置锁相环控制寄存器 SYSCTRL_PLL.SOURCE 为 0，选择 HSE 时钟作为 PLL 输入时钟源；

步骤 2：根据 HSE 的输出时钟频率，通过锁相环控制寄存器 SYSCTRL_PLL.FREQIN 配置 PLL 输入频率范围；

步骤 3：根据需要的 PLL 的输出时钟频率，通过锁相环控制寄存器 SYSCTRL_PLL.MUL 及 SYSCTRL_PLL.FREQOUT 分别配置锁相环倍频系数和输出频率范围；

步骤 4：设置锁相环稳定周期 SYSCTRL_PLL.WAITCYCLE 为 7，选择最长的 PLL 稳定时间；

步骤 5：设置系统控制寄存器 SYSCTRL_CR1.PLLEN 为 1，使能 PLL 锁相环电路；

注：

SYSCTRL_CR1 寄存器具有 KEY 保护特性，写入的数据高 16bit 数据必须是 0x5A5A，否则无法写入。

步骤 6：循环查询并等待锁相环控制寄存器 SYSCTRL_PLL.STABLE 稳定标志变为 1，即等待 PLL 输出稳定时钟；

步骤 7：根据 PLL 的输出时钟频率，按 7 FLASH 存储器章节要求配置 FLASH_CR2.WAIT 读等待周期参数；

步骤 8：设置系统控制寄存器 SYSCTRL_CR0.SYSCLK 为 2，将 SysClk 时钟来源切换为 PLL 时钟。

注：

SYSCTRL_CR0 寄存器具有 KEY 保护特性，写入的数据高 16bit 数据必须是 0x5A5A，否则无法写入。

从 PLL 切换到 HSE 操作流程如下：

本示例中，由于锁相环的输入参考时钟为 HSE，因此 HSE 时钟已经是处于稳定状态，因此不需要先使能 HSE 并等待 HSE 稳定，可直接切换。如果 PLL 的输入参考时钟是 HSI，则需要增加设置 HSE 参数、使能 HSE，然后等待 HSE 时钟稳定过程。

步骤 1：设置系统控制寄存器 SYSCTRL_CR0.SYSCLK 为 1，将 SysClk 时钟来源切换为 HSE；

注：

SYSCTRL_CR0 寄存器具有 KEY 保护特性，写入的数据高 16bit 数据必须是 0x5A5A，否则无法写入。

步骤 2：根据 HSE 的输出时钟频率，按 7 FLASH 存储器章节要求配置 FLASH_CR2.WAIT 读等待周期参数；

步骤 3：设置系统控制寄存器 SYSCTRL_CR1.PLLEN 为 0，关闭 PLL 时钟。

注：

SYSCTRL_CR1 寄存器具有 KEY 保护特性，写入的数据高 16bit 数据必须是 0x5A5A，否则无法写入。

4.6 寄存器列表

SYSCTRL 基地址：SYSCTRL_BASE = 0x4001 0000

表 4-14 SYSCTRL 寄存器列表

寄存器名称	寄存器地址	寄存器描述
SYSCTRL_CR0	SYSCTRL_BASE + 0x00	系统控制寄存器 0
SYSCTRL_CR1	SYSCTRL_BASE + 0x04	系统控制寄存器 1
SYSCTRL_CR2	SYSCTRL_BASE + 0x08	系统控制寄存器 2
SYSCTRL_IER	SYSCTRL_BASE + 0x0C	系统中断使能控制寄存器
SYSCTRL_ISR	SYSCTRL_BASE + 0x10	系统中断标志寄存器
SYSCTRL_ICR	SYSCTRL_BASE + 0x14	系统中断标志清除寄存器
SYSCTRL_HSI	SYSCTRL_BASE + 0x18	内置高频时钟控制寄存器
SYSCTRL_HSE	SYSCTRL_BASE + 0x1C	外置高频晶体控制寄存器
SYSCTRL_LSI	SYSCTRL_BASE + 0x20	内置低频时钟控制寄存器
SYSCTRL_LSE	SYSCTRL_BASE + 0x24	外置低频晶体控制寄存器
SYSCTRL_PLL	SYSCTRL_BASE + 0x28	内置锁相环控制寄存器
SYSCTRL_DEBUG	SYSCTRL_BASE + 0x2C	调试状态定时器控制寄存器
SYSCTRL_AHBEN	SYSCTRL_BASE + 0x30	AHB 外设时钟使能控制寄存器
SYSCTRL_APBEN2	SYSCTRL_BASE + 0x34	APB 外设时钟使能控制寄存器 2
SYSCTRL_APBEN1	SYSCTRL_BASE + 0x38	APB 外设时钟使能控制寄存器 1
SYSCTRL_AHBRST	SYSCTRL_BASE + 0x40	AHB 外设复位控制寄存器
SYSCTRL_APBRST2	SYSCTRL_BASE + 0x44	APB 外设复位控制寄存器 2
SYSCTRL_APBRST1	SYSCTRL_BASE + 0x48	APB 外设复位控制寄存器 1
SYSCTRL_RESETFLAG	SYSCTRL_BASE + 0x4C	系统复位标志寄存器
SYSCTRL_GTIM1CAP	SYSCTRL_BASE + 0x50	通用定时器 1 输入捕捉来源配置寄存器
SYSCTRL_GTIM2CAP	SYSCTRL_BASE + 0x54	通用定时器 2 输入捕捉来源配置寄存器
SYSCTRL_GTIM3CAP	SYSCTRL_BASE + 0x58	通用定时器 3 输入捕捉来源配置寄存器
SYSCTRL_GTIM4CAP	SYSCTRL_BASE + 0x5C	通用定时器 4 输入捕捉来源配置寄存器
SYSCTRL_GTIMETR	SYSCTRL_BASE + 0x64	通用定时器 ETR 来源配置寄存器
SYSCTRL_TIMITR	SYSCTRL_BASE + 0x6C	定时器 ITR 来源配置寄存器
SYSCTRL_MCO	SYSCTRL_BASE + 0x70	系统时钟输出控制寄存器

4.7 寄存器描述

有关寄存器描述里所使用的缩写，请参见 1 文档约定章节。

4.7.1 SYSCTRL_CR0 系统控制寄存器 0

Address offset: 0x00 Reset value: 0x0000 0000

位域	名称	权限	功能描述
31:16	KEY	WO	仅当 KEY 为 0x5A5A 时，对该寄存器的写操作有效
15:8	RFU	-	保留位，请保持默认值
7:5	HCLKPRS	RW	配置 HCLK 的时钟来源 000：设置 HCLK 的时钟源为 SysClk 001：设置 HCLK 的时钟源为 SysClk / 2 010：设置 HCLK 的时钟源为 SysClk / 4 011：设置 HCLK 的时钟源为 SysClk / 8 100：设置 HCLK 的时钟源为 SysClk / 16 101：设置 HCLK 的时钟源为 SysClk / 32 110：设置 HCLK 的时钟源为 SysClk / 64 111：设置 HCLK 的时钟源为 SysClk / 128
4:3	PCLKPRS	RW	配置 PCLK 的时钟来源 00：设置 PCLK 的时钟源为 HCLK 01：设置 PCLK 的时钟源为 HCLK / 2 10：设置 PCLK 的时钟源为 HCLK / 4 11：设置 PCLK 的时钟源为 HCLK / 8
2:0	SYSCLK	RW	配置 SysClk 的时钟来源 000：设置 SysClk 的时钟源为 HSI 001：设置 SysClk 的时钟源为 HSE 010：设置 SysClk 的时钟源为 PLL 011：设置 SysClk 的时钟源为 LSI 100：设置 SysClk 的时钟源为 LSE

4.7.2 SYSCTRL_CR1 系统控制寄存器 1

Address offset: 0x04 Reset value: 0x0000 0001

位域	名称	权限	功能描述
31:16	KEY	WO	仅当 KEY 为 0x5A5A 时，对该寄存器的写操作有效
15:9	RFU	-	保留位，请保持默认值
8	CLKCCS	RW	请写入 1
7	HSECCS	RW	请写入 1 注 1：每 HSE.DETCNT 个 LSI 时钟持续时间内 HSE 输出的时钟个数小于 0x20000 个，则判定 HSE 运行中失效。 注 2：若不满足 HSE.WAITCYCLE 指定的条件，则判定 HSE 起振失败。
6	LSECCS	RW	请写入 1 注 1：每 256 个 LSI 时钟持续时间（约 7.8ms）内 LSE 输出的时钟个数小于 128 个则判定 LSE 运行中失效。 注 2：若不满足 LSE.WAITCYCLE 指定的条件，则判定 LSE 起振失败。
5	LSELOCK	RW	外部低速时钟 LSE 锁定控制 0：LSEEN 可置位可清零 1：LSEEN 只可置位
4	LSEEN	RW	外部低速时钟 LSE 使能控制 0：关闭 1：使能 注 1：当系统进入 DeepSleep，此低速时钟不会自动关闭。 注 2：仅上电复位可以复位该控制位。
3	LSIEN	RW	内部低速时钟 LSI 使能控制 0：关闭 1：使能 注：当系统进入 DeepSleep，此低速时钟不会自动关闭。
2	PLLEN	RW	内部高速时钟 PLL 使能控制 0：关闭 1：使能 注：当系统进入 DeepSleep，此高速时钟会自动关闭。
1	HSEEN	RW	外部高速时钟 HSE 使能控制 0：关闭 1：使能 注：当系统进入 DeepSleep，此高速时钟会自动关闭。
0	HSIEN	RW	内部高速时钟 HSIOSC 使能控制 0：关闭 1：使能 注：当系统进入 DeepSleep，此高速时钟会自动关闭。

4.7.3 SYSCTRL_CR2 系统控制寄存器 2

Address offset: 0x08 Reset value: 0x0000 0000

位域	名称	权限	功能描述
31:16	KEY	WO	仅当 KEY 为 0x5A5A 时，对该寄存器的写操作有效
15:4	RFU	-	保留位，请保持默认值
3	WAKEUPCLK	RW	DeepSleep 唤醒时，系统时钟的来源配置 0：保持原系统时钟来源 1：切换系统时钟来源为 HSI 8MHz，原系统时钟来源保持使能
2	LOCKUP	RW	Cortex-M0+ LockUp 功能配置 0：关闭 1：使能 注：使能该功能，则 CPU 读到无效指令时会复位 MCU。
1	SWDIO	RW	SWD 端口功能配置 0：PA14、PA13 作为 SWD 端口，GPIO 功能不可用 1：PA14、PA13 作为 GPIO 端口，SWD 功能不可用
0	RFU	-	保留位，请保持默认值

4.7.4 SYSCTRL_HSI 内置高频时钟控制寄存器

Address offset: 0x18 Reset value: 0x---- ----

位域	名称	权限	功能描述
31:16	RFU	-	保留位，请保持默认值
15	STABLE	RO	HSIOSC 时钟稳定状态位 0：HSIOSC 时钟尚未稳定 1：HSIOSC 时钟已经稳定
14:11	DIV	RW	HSI 时钟与 HSIOSC 时钟分频系数配置 0101：HSI = HSIOSC / 6(默认值) 0110：HSI = HSIOSC / 1 1000：HSI = HSIOSC / 2 1001：HSI = HSIOSC / 4 1011：HSI = HSIOSC / 8 1100：HSI = HSIOSC / 10 1101：HSI = HSIOSC / 12 1110：HSI = HSIOSC / 14 1111：HSI = HSIOSC / 16
10:0	TRIM	RW	时钟频率调整，更改该寄存器位的数值即可调整 HSIOSC 的振荡频率。TRIM 值每增加 1 则 HSIOSC 的振荡频率增加约 0.2%，总调整范围为 32 ~ 48MHz。FLASH 中已保存了 48MHz 的校准值，将 FLASH 内的校准值读出并写入 SYSCTRL_HSI.TRIM 即可获得精准的频率。 48.00M 校准值地址：0x0001 2600 - 0x0001 2601

4.7.5 SYSCTRL_LSI 内置低频时钟控制寄存器

Address offset: 0x20 Reset value: 0x---- ----

位域	名称	权限	功能描述
31:16	RFU	-	保留位，请保持默认值
15	STABLE	RO	LSI 时钟稳定状态位 0：LSI 时钟尚未稳定 1：LSI 时钟已经稳定
14:12	RFU	-	保留位，请保持默认值
11:10	WAITCYCLE	RW	内部低速时钟 LSI 稳定时间选择 00：6 个周期 01：18 个周期 10：66 个周期 11：258 个周期
9:0	TRIM	RW	内部低速时钟频率调整，FLASH 中保存了 1 组频率的校准值。将 FLASH 内的校准值读出并写入 SYSCTRL_LSI.TRIM 即可获得精准的频率。 32.8K 校准值地址：0x0001 2602 - 0x0001 2603

4.7.6 SYSCTRL_HSE 外置高频晶体控制寄存器

Address offset: 0x1C Reset value: 0x0007 FF22

位域	名称	权限	功能描述
31:20	RFU	-	保留位，请保持默认值
19	STABLE	RO	HSE 时钟稳定状态位 0：HSE 时钟尚未稳定 1：HSE 时钟已经稳定
18:8	DETCNT	RW	HSE 运行失效检测周期配置 考虑时钟误差，一般应设置 DETCNT 的值为 8000 / f_{HSE} 例：HSE 为 4MHz，则 DETCNT 应配置为 2000。 注：DETCNT 必须大于 0。
7	FLT	RW	外部高速时钟 HSE 滤波控制 0：关闭 HSE 滤波功能 1：使能 HSE 滤波功能，HSE 频率应不大于 8MHz 注：滤波器仅供高干扰场景使用。
6	MODE	RW	外部高速时钟 HSE 工作模式配置 0：HSE 工作于晶体模式，其输出时钟由晶体与内部电路振荡产生 1：HSE 工作于输入模式，其输出时钟来自 OSC_IN 引脚
5:4	WAITCYCLE	RW	HSE 稳定检测及起振失败检测时钟周期数量选择 00：等待 8192 个 HSE 时钟信号 01：等待 32768 个 HSE 时钟信号 10：等待 131072 个 HSE 时钟信号 11：等待 262144 个 HSE 时钟信号 注：当使能 HSECCS 后，起振失败检测电路会在限定时间内检测 HSE 时钟信号个数，四档限定时间分别为 65ms、65ms、65ms、130ms。
3:2	FREQRANGE	RW	外部晶体工作频率选择 00：4M~8M 01：8M~16M 10：16M~24M 11：24M~32M
1:0	DRIVER	RW	HSE 晶体驱动能力选择 00：最弱驱动能力 01：弱驱动能力 10：默认驱动能力 (推荐值) 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。驱动能力越强则功耗越大，驱动能力越弱则功耗越小。

4.7.7 SYSCTRL_LSE 外置低频晶体控制寄存器

Address offset: 0x24 Reset value: 0x0000 002B

位域	名称	权限	功能描述
31:16	RFU	-	保留位，请保持默认值
15	STABLE	RO	LSE 时钟稳定状态位 0：LSE 时钟尚未稳定 1：LSE 时钟已经稳定
14:7	RFU	-	保留位，请保持默认值
6	MODE	RW	外部低速时钟 LSE 工作模式配置 0：LSE 工作于晶体模式，其输出时钟由晶体与内部电路振荡产生 1：LSE 工作于输入模式，其输出时钟来自 OSC32_IN 引脚
5:4	WAITCYCLE	RW	LSE 稳定检测及起振失败检测时钟周期数量选择 00：等待 256 个 LSE 时钟信号 01：等待 1024 个 LSE 时钟信号 10：等待 4096 个 LSE 时钟信号 11：等待 16384 个 LSE 时钟信号 注：当使能 LSECCS 后，起振失败检测电路会在限定时间内检测 LSE 时钟信号个数，四档限定时间分别为 1s、1s、1s、2s。
3:2	AMP	RW	LSE 晶体振荡幅度配置 00：最小振幅 01：较小振幅 10：正常振幅（推荐值） 11：较大振幅
1:0	DRIVER	RW	LSE 晶体驱动能力配置 00：最小驱动能力 01：较小驱动能力 10：正常驱动能力 11：较强驱动能力（推荐值） 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。驱动能力越强则功耗越大，驱动能力越弱则功耗越小。

注：

仅上电复位可以复位该寄存器。

4.7.8 SYSCTRL_PLL 内置锁相环控制寄存器

Address offset: 0x28 Reset value: 0x0005 3483

位域	名称	权限	功能描述
31:20	RFU	-	保留位，请保持默认值
19:16	RFU	RW	调试控制位，请保持默认值
15	STABLE	RO	PLL 时钟稳定状态位 0：PLL 时钟尚未稳定 1：PLL 时钟已经稳定
14:12	WAITCYCLE	RW	PLL 稳定时间选择 000：128 个 PLL 时钟周期 001：256 个 PLL 时钟周期 010：512 个 PLL 时钟周期 011：1024 个 PLL 时钟周期 100：2048 个 PLL 时钟周期 101：4096 个 PLL 时钟周期 110：8192 个 PLL 时钟周期 111：16384 个 PLL 时钟周期
11:9	FREQOUT	RW	PLL 输出时钟频率范围配置 000：12M ~ 18M 001：18M ~ 24M 010：24M ~ 36M 011：36M ~ 48M 1xx：48M ~ 72M
8:4	MUL	RW	PLL 倍频系数配置 PLL 输出频率 = 输入频率 ×MUL MUL 取值为 0x02 ~ 0x0C
3:2	FREQIN	RW	PLL 输入时钟频率范围配置 00：4M ~ 6M 01：6M ~ 12M 10：12M ~ 20M 11：20M ~ 24M
1:0	SOURCE	RW	输入时钟来源选择 00：HSE 晶体生成的时钟 01：HSE 从引脚输入的时钟 11：HSI 时钟 (HSIOSC 分频后的时钟)

4.7.9 SYSCTRL_IER 系统中断使能控制寄存器

Address offset: 0x0C Reset value: 0x0000 0000

位域	名称	权限	功能描述
31:16	KEY	WO	仅当 KEY 为 0x5A5A 时，对该寄存器的写操作有效
15:9	RFU	-	保留位，请保持默认值
8	HSEFAULT	RW	HSE 运行失效中断使能控制 0：禁止 HSE 运行失效中断 1：使能 HSE 运行失效中断
7	LSEFAULT	RW	LSE 运行失效中断使能控制 0：禁止 LSE 运行失效中断 1：使能 LSE 运行失效中断
6	HSEFAIL	RW	HSE 起振失败中断使能控制 0：禁止 HSE 起振失败中断 1：使能 HSE 起振失败中断
5	LSEFAIL	RW	LSE 起振失败中断使能控制 0：禁止 LSE 起振失败中断 1：使能 LSE 起振失败中断
4	LSERDY	RW	LSE 稳定中断使能控制 0：禁止 LSE 稳定中断 1：使能 LSE 稳定中断
3	LSIRDY	RW	LSI 稳定中断使能控制 0：禁止 LSI 稳定中断 1：使能 LSI 稳定中断
2	PLLRDY	RW	PLL 稳定中断使能控制 0：禁止 PLL 稳定中断 1：使能 PLL 稳定中断
1	HSERDY	RW	HSE 稳定中断使能控制 0：禁止 HSE 稳定中断 1：使能 HSE 稳定中断
0	HSIRDY	RW	HSIOSC 稳定中断使能控制 0：禁止 HSIOSC 稳定中断 1：使能 HSIOSC 稳定中断

4.7.10 SYSCTRL_ISR 系统中断标志寄存器

Address offset: 0x10 Reset value: 0x0000 0801

位域	名称	权限	功能描述
31:16	RFU	-	保留位，请保持默认值
15	LSESTABLE	RO	LSE 时钟稳定状态位；硬件置位清零 0：LSE 时钟尚未稳定 1：LSE 时钟已经稳定
14	LSISTABLE	RO	LSI 时钟稳定状态位；硬件置位清零 0：LSI 时钟尚未稳定 1：LSI 时钟已经稳定
13	PLLSTABLE	RO	PLL 时钟稳定状态位；硬件置位清零 0：PLL 时钟尚未稳定 1：PLL 时钟已经稳定
12	HSESTABLE	RO	HSE 时钟稳定状态位；硬件置位清零 0：HSE 时钟尚未稳定 1：HSE 时钟已经稳定
11	HSISTABLE	RO	HSIOSC 时钟稳定状态位；硬件置位清零 0：HSIOSC 时钟尚未稳定 1：HSIOSC 时钟已经稳定
10:9	RFU	-	保留位，请保持默认值
8	HSEFAULT	RO	HSE 时钟运行失效标志位 0：HSE 时钟运行正常 1：HSE 时钟运行中停振
7	LSEFAULT	RO	LSE 时钟运行失效标志位 0：LSE 时钟运行正常 1：LSE 时钟运行中停振
6	HSEFAIL	RO	HSE 时钟起振失败标志位 0：HSE 时钟起振成功 1：HSE 时钟起振失败
5	LSEFAIL	RO	LSE 时钟起振失败标志位 0：LSE 时钟起振成功 1：LSE 时钟起振失败
4	LSERDY	RO	LSE 时钟稳定标志位 0：LSE 时钟尚未稳定 1：LSE 时钟已经稳定
3	LSIRDY	RO	LSI 时钟稳定标志位 0：LSI 时钟尚未稳定 1：LSI 时钟已经稳定
2	PLLRDY	RO	PLL 时钟稳定标志位 0：PLL 时钟尚未稳定 1：PLL 时钟已经稳定

位域	名称	权限	功能描述
1	HSERDY	RO	HSE 时钟稳定标志位 0：HSE 时钟尚未稳定 1：HSE 时钟已经稳定
0	HSIRDY	RO	HSIOSC 时钟稳定标志位 0：HSIOSC 时钟尚未稳定 1：HSIOSC 时钟已经稳定

4.7.11 SYSCTRL_ICR 系统中断标志清除寄存器

Address offset: 0x14 Reset value: 0x0000 01FF

位域	名称	权限	功能描述
31:9	RFU	-	保留位，请保持默认值
8	HSEFAULT	R1W0	HSE 时钟运行失效标志位清零控制 W0：清除 ISR 寄存器中的相应标志 W1：无功能
7	LSEFAULT	R1W0	LSE 时钟运行失效标志位清零控制 W0：清除 ISR 寄存器中的相应标志 W1：无功能
6	HSEFAIL	R1W0	HSE 时钟起振失败标志位清零控制 W0：清除 ISR 寄存器中的相应标志 W1：无功能
5	LSEFAIL	R1W0	LSE 时钟起振失败标志位清零控制 W0：清除 ISR 寄存器中的相应标志 W1：无功能
4	LSERDY	R1W0	LSE 时钟稳定标志位清零控制 W0：清除 ISR 寄存器中的相应标志 W1：无功能
3	LSIRDY	R1W0	LSI 时钟稳定标志位清零控制 W0：清除 ISR 寄存器中的相应标志 W1：无功能
2	PLLRDY	R1W0	PLL 时钟稳定标志位清零控制 W0：清除 ISR 寄存器中的相应标志 W1：无功能
1	HSERDY	R1W0	HSE 时钟稳定标志位清零控制 W0：清除 ISR 寄存器中的相应标志 W1：无功能
0	HSIRDY	R1W0	HSIOSC 时钟稳定标志位清零控制 W0：清除 ISR 寄存器中的相应标志 W1：无功能

4.7.12 SYSCTRL_AHBEN AHB 外设时钟使能控制寄存器

Address offset: 0x30 Reset value: 0x0000 0000

位域	名称	权限	功能描述
31:10	RFU	-	保留位，请保持默认值
9	GPIOF	RW	GPIOF 端口配置时钟及工作时钟使能控制 0：关闭 1：使能
8:7	RFU	-	保留位，请保持默认值
6	GPIOC	RW	GPIOC 端口配置时钟及工作时钟使能控制 0：关闭 1：使能
5	GPIOB	RW	GPIOB 端口配置时钟及工作时钟使能控制 0：关闭 1：使能
4	GPIOA	RW	GPIOA 端口配置时钟及工作时钟使能控制 0：关闭 1：使能
3	RFU	-	保留位，请保持默认值
2	CRC	RW	CRC 模块配置时钟及工作时钟使能控制 0：关闭 1：使能
1	FLASH	RW	FLASH 模块配置时钟使能控制 0：关闭 1：使能
0	DMA	RW	DMA 模块配置时钟及工作时钟使能控制 0：关闭 1：使能

4.7.13 SYSCTRL_APBEN1 APB 外设时钟使能控制寄存器 1

Address offset: 0x38 Reset value: 0x0000 0000

位域	名称	权限	功能描述
31:13	RFU	-	保留位，请保持默认值
12	I2C2	RW	I2C2 模块配置时钟及工作时钟使能控制 0：关闭 1：使能
11	I2C1	RW	I2C1 模块配置时钟及工作时钟使能控制 0：关闭 1：使能
10:9	RFU	-	保留位，请保持默认值
8	UART3	RW	UART3 模块配置时钟使能控制 0：关闭 1：使能
7	UART2	RW	UART2 模块配置时钟使能控制 0：关闭 1：使能
6	SPI2	RW	SPI2 模块配置时钟及工作时钟使能控制 0：关闭 1：使能
5	IWDT	RW	IWDT 模块配置时钟使能控制 0：关闭 1：使能
4	WWDT	RW	WWDT 模块配置时钟及工作时钟使能控制 0：关闭 1：使能
3	RTC	RW	RTC 模块配置时钟使能控制 0：关闭 1：使能
2	GTIM2	RW	GTIM2 模块配置时钟及工作时钟使能控制 0：关闭 1：使能
1	GTIM1	RW	GTIM1 模块配置时钟及工作时钟使能控制 0：关闭 1：使能
0	RFU	-	保留位，请保持默认值

4.7.14 SYSCTRL_APBEN2 APB 外设时钟使能控制寄存器 2

Address offset: 0x34 Reset value: 0x0000 8000

位域	名称	权限	功能描述
31:14	RFU	-	保留位，请保持默认值
13	AWT	RW	AWT 模块配置时钟使能控制 0：关闭 1：使能
12	BTIM	RW	BTIM1-3 模块配置时钟及工作时钟使能控制 0：关闭 1：使能
11	GTIM4	RW	GTIM4 模块配置时钟及工作时钟使能控制 0：关闭 1：使能
10	GTIM3	RW	GTIM3 模块配置时钟及工作时钟使能控制 0：关闭 1：使能
9	UART1	RW	UART1 模块配置时钟使能控制 0：关闭 1：使能
8	SPI1	RW	SPI1 模块配置时钟及工作时钟使能控制 0：关闭 1：使能
7:5	RFU	-	保留位，请保持默认值
4	VC	RW	VC 模块配置时钟使能控制 0：关闭 1：使能
3	RFU	RW	保留位，请保持默认值
2	ADC	RW	ADC 模块配置时钟及工作时钟使能控制 0：关闭 1：使能
1:0	RFU	-	保留位，请保持默认值

4.7.15 SYSCTRL_AHBRST AHB 外设复位控制寄存器

Address offset: 0x40 Reset value: 0x0000 0277

位域	名称	权限	功能描述
31:10	RFU	-	保留位，请保持默认值
9	GPIOF	RW	GPIOF 模块复位控制 0：模块处于复位状态 1：模块正常工作
8:7	RFU	-	保留位，请保持默认值
6	GPIOC	RW	GPIOC 模块复位控制 0：模块处于复位状态 1：模块正常工作
5	GPIOB	RW	GPIOB 模块复位控制 0：模块处于复位状态 1：模块正常工作
4	GPIOA	RW	GPIOA 模块复位控制 0：模块处于复位状态 1：模块正常工作
3	RFU	-	保留位，请保持默认值
2	CRC	RW	CRC 模块复位控制 0：模块处于复位状态 1：模块正常工作
1	FLASH	RW	FLASH 模块复位控制 0：模块处于复位状态 1：模块正常工作
0	DMA	RW	DMA 模块复位控制 0：模块处于复位状态 1：模块正常工作

4.7.16 SYSCTRL_APBRST1 APB 外设复位控制寄存器 1

Address offset: 0x48 Reset value: 0x0000 19FF

位域	名称	权限	功能描述
31:13	RFU	-	保留位，请保持默认值
12	I2C2	RW	I2C2 模块复位控制 0：模块处于复位状态 1：模块正常工作
11	I2C1	RW	I2C1 模块复位控制 0：模块处于复位状态 1：模块正常工作
10:9	RFU	-	保留位，请保持默认值
8	UART3	RW	UART3 模块复位控制 0：模块处于复位状态 1：模块正常工作
7	UART2	RW	UART2 模块复位控制 0：模块处于复位状态 1：模块正常工作
6	SPI2	RW	SPI2 模块复位控制 0：模块处于复位状态 1：模块正常工作
5	IWDT	RW	IWDT 模块复位控制 0：模块处于复位状态 1：模块正常工作
4	WWDT	RW	WWDT 模块复位控制 0：模块处于复位状态 1：模块正常工作
3	RTC	RW	RTC 模块复位控制 0：模块处于复位状态 1：模块正常工作
2	GTIM2	RW	GTIM2 模块复位控制 0：模块处于复位状态 1：模块正常工作
1	GTIM1	RW	GTIM1 模块复位控制 0：模块处于复位状态 1：模块正常工作
0	RFU	-	保留位，请保持默认值

4.7.17 SYSCTRL_APBRST2 APB 外设复位控制寄存器 2

Address offset: 0x44 Reset value: 0x0000 BF9D

位域	名称	权限	功能描述
31:14	RFU	-	保留位，请保持默认值
13	AWT	RW	AWT 模块复位控制 0：模块处于复位状态 1：模块正常工作
12	BTIM	RW	BTIM1-3 模块复位控制 0：模块处于复位状态 1：模块正常工作
11	GTIM4	RW	GTIM4 模块复位控制 0：模块处于复位状态 1：模块正常工作
10	GTIM3	RW	GTIM3 模块复位控制 0：模块处于复位状态 1：模块正常工作
9	UART1	RW	UART1 模块复位控制 0：模块处于复位状态 1：模块正常工作
8	SPI1	RW	SPI1 模块复位控制 0：模块处于复位状态 1：模块正常工作
7:5	RFU	-	保留位，请保持默认值
4	VC	RW	VC 模块复位控制 0：模块处于复位状态 1：模块正常工作
3	RFU	-	保留位，请保持默认值
2	ADC	RW	ADC 模块复位控制 0：模块处于复位状态 1：模块正常工作
1:0	RFU	-	保留位，请保持默认值

4.7.18 SYSCTRL_RESETFLAG 系统复位标志寄存器

Address offset: 0x4C Reset value: 0x---- ----

位域	名称	权限	功能描述
31:10	RFU	-	保留位，请保持默认值
9	SYSRESETREQ	RW0	Cortex-M0+ CPU SYSRESETREQ 软复位标志 0：未发生 CPU SYSRESETREQ 软复位 1：已发生 CPU SYSRESETREQ 软复位 写 0 清除，写 1 无效
8	LOCKUP	RW0	Cortex-M0+ CPU Lockup 复位标志 0：未发生 Lockup 复位 1：已发生 Lockup 复位 写 0 清除，写 1 无效
7	RFU	-	保留位，请保持默认值
6	RSTB	RW0	NRST 引脚复位标志 0：未发生引脚复位 1：已发生引脚复位 写 0 清除，写 1 无效
5	WWDT	RW0	WWDT 复位标志，需要软件初始化及清除 0：未发生 WWDT 复位 1：已发生 WWDT 复位 写 0 清除，写 1 无效
4	IWDT	RW0	IWDT 复位标志 0：未发生 IWDT 复位 1：已发生 IWDT 复位 写 0 清除，写 1 无效
3	LVD	RW0	LVD 复位标志 0：未发生 LVD 复位 1：已发生 LVD 复位 写 0 清除，写 1 无效
2:1	RFU	-	保留位，请保持默认值
0	POR	RW0	POR/BOR 复位标志 0：未发生 POR/BOR 复位 1：已发生 POR/BOR 复位 写 0 清除，写 1 无效

4.7.19 SYSCTRL_DEBUG 调试状态定时器控制寄存器

Address offset: 0x2C Reset value: 0x0000 07FF

位域	名称	权限	功能描述
31:11	RFU	-	保留位，请保持默认值
10	WWDT	RW	调试状态下，WWDT 计数功能配置 0：正常计数 1：暂停计数
9	IWDT	RW	调试状态下，IWDT 计数功能配置 0：正常计数 1：暂停计数
8	RTC	RW	调试状态下，RTC 计数功能配置 0：正常计数 1：暂停计数
7	RFU	-	保留位，请保持默认值
6	AWT	RW	调试状态下，AWT 计数功能配置 0：正常计数 1：暂停计数
5	BTIM123	RW	调试状态下，BTIM1 / BTIM2 / BTIM3 计数功能配置 0：正常计数 1：暂停计数
4	GTIM4	RW	调试状态下，GTIM4 计数功能配置 0：正常计数 1：暂停计数
3	GTIM3	RW	调试状态下，GTIM3 计数功能配置 0：正常计数 1：暂停计数
2	GTIM2	RW	调试状态下，GTIM2 计数功能配置 0：正常计数 1：暂停计数
1	GTIM1	RW	调试状态下，GTIM1 计数功能配置 0：正常计数 1：暂停计数
0	RFU	-	保留位，请保持默认值

4.7.20 SYSCTRL_GTIM1CAP 通用定时器 1 输入捕捉来源配置寄存器

Address offset: 0x50 Reset value: 0x0000 0000

位域	名称	权限	功能描述
31:15	RFU	-	保留位，请保持默认值
14:12	CH4	RW	GTIM1 的 CH4 输入捕捉来源配置 详见 GTIM4CAP[2:0]
11	RFU	-	保留位，请保持默认值
10:8	CH3	RW	GTIM1 的 CH3 输入捕捉来源配置 详见 GTIM4CAP[2:0]
7	RFU	-	保留位，请保持默认值
6:4	CH2	RW	GTIM1 的 CH2 输入捕捉来源配置 详见 GTIM4CAP[2:0]
3	RFU	-	保留位，请保持默认值
2:0	CH1	RW	GTIM1 的 CH1 输入捕捉来源配置 详见 GTIM4CAP[2:0]

4.7.21 SYSCTRL_GTIM2CAP 通用定时器 2 输入捕捉来源配置寄存器

Address offset: 0x54 Reset value: 0x0000 0000

位域	名称	权限	功能描述
31:15	RFU	-	保留位，请保持默认值
14:12	CH4	RW	GTIM2 的 CH4 输入捕捉来源配置 详见 GTIM4CAP[2:0]
11	RFU	-	保留位，请保持默认值
10:8	CH3	RW	GTIM2 的 CH3 输入捕捉来源配置 详见 GTIM4CAP[2:0]
7	RFU	-	保留位，请保持默认值
6:4	CH2	RW	GTIM2 的 CH2 输入捕捉来源配置 详见 GTIM4CAP[2:0]
3	RFU	-	保留位，请保持默认值
2:0	CH1	RW	GTIM2 的 CH1 输入捕捉来源配置 详见 GTIM4CAP[2:0]

4.7.22 SYSCTRL_GTIM3CAP 通用定时器 3 输入捕捉来源配置寄存器

Address offset: 0x58 Reset value: 0x0000 0000

位域	名称	权限	功能描述
31:15	RFU	-	保留位，请保持默认值
14:12	CH4	RW	GTIM3 的 CH4 输入捕捉来源配置 详见 GTIM4CAP[2:0]
11	RFU	-	保留位，请保持默认值
10:8	CH3	RW	GTIM3 的 CH3 输入捕捉来源配置 详见 GTIM4CAP[2:0]
7	RFU	-	保留位，请保持默认值
6:4	CH2	RW	GTIM3 的 CH2 输入捕捉来源配置 详见 GTIM4CAP[2:0]
3	RFU	-	保留位，请保持默认值
2:0	CH1	RW	GTIM3 的 CH1 输入捕捉来源配置 详见 GTIM4CAP[2:0]

4.7.23 SYSCTRL_GTIM4CAP 通用定时器 4 输入捕捉来源配置寄存器

Address offset: 0x5C Reset value: 0x0000 0000

位域	名称	权限	功能描述
31:15	RFU	-	保留位，请保持默认值
14:12	CH4	RW	GTIM4 的 CH4 输入捕捉来源配置 详见 GTIM4CAP[2:0]
11	RFU	-	保留位，请保持默认值
10:8	CH3	RW	GTIM4 的 CH3 输入捕捉来源配置 详见 GTIM4CAP[2:0]
7	RFU	-	保留位，请保持默认值
6:4	CH2	RW	GTIM4 的 CH2 输入捕捉来源配置 详见 GTIM4CAP[2:0]
3	RFU	-	保留位，请保持默认值
2:0	CH1	RW	GTIM4 的 CH1 输入捕捉来源配置 000：由 GPIOx_AFRH 和 GPIOx_AFRL 配置 001：UART1 的 RXD 信号 010：UART2 的 RXD 信号 011：UART3 的 RXD 信号 100：VC1 的比较输出信号 101：VC2 的比较输出信号 110：LVD 的输出信号

4.7.24 SYSCTRL_GTIMETR 通用定时器 ETR 来源配置寄存器

Address offset: 0x64 Reset value: 0x0000 0000

位域	名称	权限	功能描述
31:15	RFU	-	保留位，请保持默认值
14:12	GTIM4ETR	RW	GTIM4 的 ETR 来源配置 详见 GTIMETR[2:0]
11	RFU	-	保留位，请保持默认值
10:8	GTIM3ETR	RW	GTIM3 的 ETR 来源配置 详见 GTIMETR[2:0]
7	RFU	-	保留位，请保持默认值
6:4	GTIM2ETR	RW	GTIM2 的 ETR 来源配置 详见 GTIMETR[2:0]
3	RFU	-	保留位，请保持默认值
2:0	GTIM1ETR	RW	GTIM1 的 ETR 来源配置 000：由 GPIOx_AFRH 和 GPIOx_AFRL 配置 001：UART1 的 RXD 信号 010：UART2 的 RXD 信号 011：UART3 的 RXD 信号 100：VC1 的比较输出信号 101：VC2 的比较输出信号 110：LVD 的输出信号

4.7.25 SYSCTRL_TIMITR 定时器 ITR 来源配置寄存器

Address offset: 0x6C Reset value: 0x0000 0000

位域	名称	权限	功能描述
31:24	RFU	-	保留位，请保持默认值
23:21	BTIM3ITR	RW	BTIM3 的 ITR 来源配置 详见 TIMITR [5:3]
20:18	BTIM2ITR	RW	BTIM2 的 ITR 来源配置 详见 TIMITR [5:3]
17:15	BTIM1ITR	RW	BTIM1 的 ITR 来源配置 详见 TIMITR [5:3]
14:12	GTIM4ITR	RW	GTIM4 的 ITR 来源配置 详见 TIMITR [5:3]
11:9	GTIM3ITR	RW	GTIM3 的 ITR 来源配置 详见 TIMITR [5:3]
8:6	GTIM2ITR	RW	GTIM2 的 ITR 来源配置 详见 TIMITR [5:3]
5:3	GTIM1ITR	RW	GTIM1 的 ITR 来源配置 000：BTIM1 的溢出信号 001：BTIM2 的溢出信号 010：BTIM3 的溢出信号 011：GTIM1 的溢出信号 100：GTIM2 的溢出信号 101：GTIM3 的溢出信号 110：GTIM4 的溢出信号
2:0	RFU	-	保留位，请保持默认值

注：

不可配置 TIM 的 ITR 来源为自己的溢出信号。

4.7.26 SYSCTRL_MCO 系统时钟输出控制寄存器

Address offset: 0x70 Reset value: 0x0000 0000

位域	名称	权限	功能描述
31:7	RFU	-	保留位，请保持默认值
6:4	DIV	RW	MCO 输出分频控制 000：1 分频 001：2 分频 010：8 分频 011：64 分频 100：128 分频 101：256 分频 110：512 分频 111：1024 分频
3:0	SOURCE	RW	MCO 输出信号来源配置 0000：无输出 0001：HCLK 0010：PCLK 0011：HSIOSC 0100：LSI 0101：HSE 0110：LSE 0111：PLL 1000：RC150K 1001：RC10K

5 中断

5.1 概述

ARM® Cortex®-M0+ 内核的嵌套向量中断控制器 (NVIC)，用于管理中断和异常。NVIC 和处理器内核紧密相连，可以实现低延迟的异常和中断处理。

处理器支持最多 32 个中断请求 (IRQ) 输入，支持多个内部异常。

本章节只介绍了处理器的 32 个外部中断请求（IRQ0 ~ IRQ31），处理器内部异常的具体情况请参考“ARM® Cortex®-M0+Technical Reference Manual”与“ARM® v6-M Architecture Reference Manual”。

5.2 主要特性

- 16 个内部异常
- 32 个可屏蔽外部中断
- 4 个可编程的优先级
- 低延时的异常和中断处理
- 支持中断嵌套
- 中断向量表重映射

5.3 中断优先级

外部中断可设置 4 级优先级，最高优先级为“0”，最低优先级为“3”，默认值为“0”。

当处理器正在执行一个中断处理程序时，如果出现一个更高优先级的中断，那么这个中断就被抢占。如果出现的中断的优先级和正在处理的中断的优先级相同或更低，这个中断就不会被抢占，但是新中断的状态就变为挂起。如果多个挂起的中断具有相同的优先级，中断编号越小的挂起中断优先处理。例如，如果 IRQ[0] 和 IRQ[1] 均挂起时，并且两者的优先级相同，那么先处理 IRQ[0]。

5.4 中断向量表

ARM® Cortex®-M0+ 响应中断时，处理器自动从存储器的中断向量表中取出中断服务程序 (ISR) 的起始地址。中断向量表包括主栈指针（MSP）的初始值，内部异常和外部中断的服务程序入口地址。每个中断向量占用 1 个字（4 字节），中断向量的存储地址为向量编号乘以 4，如下表所示：

表 5-1 中断向量表

向量编号	外部中断号（IRQ#）	优先级	中断源	简介	地址
0	-	-	-	MSP 初始值	0x0000 0000
1	-	-3	Reset	复位向量	0x0000 0004
2	-	-2	NMI	不可屏蔽中断 [2]	0x0000 0008
3	-	-1	HardFault	硬件错误异常（fault）	0x0000 000C
4-10	-	-	-	保留	0x0000 0010~002B
11	-	可配置	SVCall	通过 SWI 指令调用的管理程序	0x0000 002C
12-13	-	-	-	保留	0x0000 0030~0037
14	-	可配置	PendSV	系统服务的可挂起请求	0x0000 0038
15	-	可配置	SysTick	系统滴答定时器	0x0000 003C
16	0	可配置	WDT[1]	窗口看门狗和独立看门狗中断	0x0000 0040
17	1	可配置	LVD	LVD 全局中断	0x0000 0044
18	2	可配置	RTC	RTC 全局中断	0x0000 0048
19	3	可配置	FLASHRAM[1]	FLASH/RAM 全局中断	0x0000 004C
20	4	可配置	RCC	RCC 全局中断 [3]	0x0000 0050
21	5	可配置	GPIOA	GPIOA 全局中断	0x0000 0054
22	6	可配置	GPIOB	GPIOB 全局中断	0x0000 0058
23	7	可配置	GPIOC	GPIOC 全局中断	0x0000 005C
24	8	可配置	GPIOF	GPIOF 全局中断	0x0000 0060
25	9	可配置	DMA1	DMA 通道 1 全局中断	0x0000 0064
26	10	可配置	DMA2	DMA 通道 2 全局中断	0x0000 0068
27	11	-	-	保留	0x0000 006C
28	12	可配置	ADC	ADC 全局中断	0x0000 0070
29	13	-	-	保留	0x0000 0074
30	14	可配置	VC1	VC1 全局中断	0x0000 0078
31	15	可配置	VC2	VC2 全局中断	0x0000 007C

向量编号	外部中断号（IRQ#）	优先级	中断源	简介	地址
32	16	可配置	GTIM1	GTIM1 全局中断	0x0000 0080
33	17	可配置	GTIM2	GTIM2 全局中断	0x0000 0084
34	18	可配置	GTIM3	GTIM3 全局中断	0x0000 0088
35	19	可配置	GTIM4	GTIM4 全局中断	0x0000 008C
36	20	可配置	BTIM1	BTIM1 全局中断	0x0000 0090
37	21	可配置	BTIM2	BTIM2 全局中断	0x0000 0094
38	22	可配置	BTIM3	BTIM3 全局中断	0x0000 0098
39	23	可配置	I2C1	I2C1 全局中断	0x0000 009C
40	24	可配置	I2C2	I2C2 全局中断	0x0000 00A0
41	25	可配置	SPI1	SPI1 全局中断	0x0000 00A4
42	26	可配置	SPI2	SPI2 全局中断	0x0000 00A8
43	27	可配置	UART1	UART1 全局中断	0x0000 00AC
44	28	可配置	UART2	UART2 全局中断	0x0000 00B0
45	29	可配置	UART3	UART3 全局中断	0x0000 00B4
46	30	可配置	AWT	AWT 全局中断	0x0000 00B8
47	31	可配置	FAULT	HSE/LSE 运行中失效中断 [4]	0x0000 00BC

注 1:

由于部分外设的中断复用一个 IRQ 中断源，用户在中断服务程序中应先检查中断标志位，以确定产生中断的外设。

注 2:

NMI 在 CW32F020 中未使用。

注 3:

HSE 、LSE 时钟信号起振失败和 LSI、LSE、HSIOSC、HSE、PLL 时钟信号稳定对应 RCC 全局中断。

注 4:

HSE 或 LSE 时钟信号在运行中失效对应 FAULT 中断。

5.5 中断相关寄存器

5.5.1 NVIC 中断使能和禁止使能

ARM® Cortex-M0+ 处理器支持最多 32 个外部中断源，分别对应中断使能设置寄存器 NIVC_ISER 的 32 个使能位，和中断使能清除寄存器 NVIC_ICER 的 32 个禁止位。将使能位置 1，允许中断；将禁止位置 1，禁止中断。

上文中 NVIC 中断使能仅针对处理器 NVIC 而言，外设的中断是否使能，还受相应外设的中断控制寄存器控制。

5.5.2 NVIC 中断挂起和清除挂起

在中断发生时，如果系统正在处理与之相同优先级或更高优先级的中断，系统将不会立即处理此中断，而是将中断的状态设置为挂起，保存在中断挂起状态寄存器中; 在处理器未进入此中断处理之前，如没有手动清除挂起状态，该状态将会一直保持有效。当处理器开始进入中断处理时，硬件会自动清除相应的中断挂起状态。

用户可通过设置中断挂起设置寄存器 NVIC_ISPR 的对应位，将此中断的状态设置为挂起状态，如果系统没有正在处理与之相同优先级或更高优先级的中断，此中断将被立即响应并处理。

用户可以通过设置中断挂起清除寄存器 NVIC_ICPR 的对应位，将此中断的状态设置为挂起清除状态。

5.5.3 NVIC 中断优先级

中断优先级控制寄存器 NVIC_IPR0 ~ NVIC_IPR7，用于设置 IRQ0~IRQ31 的中断优先级，每个中断源使用 8 位，在 CW32F020 中仅使用了高两位，最多可设置 4 个中断优先级。

注：

ARM® Cortex-M0+ 的中断优先级寄存器的设置应在中断使能之前，用户不可在中断使能之后改变中断优先级，这将导致不可预知的结果。

5.5.4 NVIC 中断屏蔽

在某些特殊场合，需要禁止所有中断，可以使用中断屏蔽寄存器 PRIMASK 实现。PRIMASK 只有最低 1 位有效，将此位置 1，除了 NMI 和硬件错误异常之外的所有外部中断和异常都被禁止；清 0 后，允许响应中断和异常。该位复位后默认为 0。

ARM® Cortex-M0+ 有专用的 ARM 指令用于修改 PRIMASK 寄存器，CPSIE i 和 CPSID i，详细请参考《ARM® v6-M Architecture Reference Manual》。

汇编指令示例参考：

```
CPSIE i    ; 清除 PRIMASK （使能中断）
CPSID i    ; 设置 PRIMASK （禁止中断）
```

C 语言（调用 CMSIS 设备驱动库）示例参考：

```
void __enable_irq(void);        // 清除 PRIMASK
void __disable_irq(void);       // 设置 PRIMASK
```

5.5.5 外设中断使能

外设模块一般都有各自的中断使能寄存器，在使用中断时，必须首先打开外设中断使能，同时参见表 5-1 中断向量表打开该中断源的 NVIC 中断使能。具体外设的中断使能，请参阅相关外设模块章节描述。

5.6 寄存器列表

NVIC 基地址：NVIC_BASE = 0xE000 E000

表 5-2 NVIC 寄存器列表

寄存器名称	寄存器地址	寄存器描述
NVIC_ISER	NVIC_BASE + 0x100	IRQ0~IRQ31 中断使能设置寄存器
NVIC_ICER	NVIC_BASE + 0x180	IRQ0~IRQ31 中断使能清除寄存器
NVIC_ISPR	NVIC_BASE + 0x200	IRQ0~IRQ31 中断挂起设置寄存器
NVIC_ICPR	NVIC_BASE + 0x280	IRQ0~IRQ31 中断挂起清除寄存器
NVIC_IPR0	NVIC_BASE + 0x400	IRQ0~IRQ3 中断优先级控制寄存器 0
NVIC_IPR1	NVIC_BASE + 0x404	IRQ4~IRQ7 中断优先级控制寄存器 1
NVIC_IPR2	NVIC_BASE + 0x408	IRQ8~IRQ11 中断优先级控制寄存器 2
NVIC_IPR3	NVIC_BASE + 0x40C	IRQ12~IRQ15 中断优先级控制寄存器 3
NVIC_IPR4	NVIC_BASE + 0x410	IRQ16~IRQ19 中断优先级控制寄存器 4
NVIC_IPR5	NVIC_BASE + 0x414	IRQ20~IRQ23 中断优先级控制寄存器 5
NVIC_IPR6	NVIC_BASE + 0x418	IRQ24~IRQ27 中断优先级控制寄存器 6
NVIC_IPR7	NVIC_BASE + 0x41C	IRQ28~IRQ31 中断优先级控制寄存器 7

5.7 寄存器描述

有关寄存器描述里所使用的缩写，请参见 1 文档约定章节。

5.7.1 NVIC_ISER 中断使能设置寄存器

Address offset: 0x100 Reset value: 0x0000 0000

位域	名称	权限	功能描述
31:0	SETIRQ	RW	设置使能外部中断 IRQ0 ~ IRQ31；写“1”置位，写“0”无效。 [0]：IRQ0 [1]：IRQ1 [2]：IRQ2 …… [31]：IRQ31 读出值表示当前中断使能状态

5.7.2 NVIC_ICER 中断使能清除寄存器

Address offset: 0x180 Reset value: 0x0000 0000

位域	名称	权限	功能描述
31:0	CLRIRQ	RW	设置禁止外部中断 IRQ0 ~ IRQ31；写“1”置位，写“0”无效。 [0]：IRQ0 [1]：IRQ1 [2]：IRQ2 …… [31]：IRQ31 读出值表示当前中断使能状态

5.7.3 NVIC_ISPR 中断挂起设置寄存器

Address offset: 0x200 Reset value: 0x0000 0000

位域	名称	权限	功能描述
31:0	SETPEND	RW	设置外部中断 IRQ0 ~ IRQ31 的挂起状态；写“1”置位，写“0”无效。 [0]：IRQ0 [1]：IRQ1 [2]：IRQ2 …… [31]：IRQ31 读出值表示当前中断挂起状态

5.7.4 NVIC_ICPR 中断挂起清除寄存器

Address offset: 0x280 Reset value: 0x0000 0000

位域	名称	权限	功能描述
31:0	CLRPEND	RW	清除外部中断 IRQ0 ~ IRQ31 的挂起状态; 写 “1” 清除，写 “0” 无效。 [0]：IRQ0 [1]：IRQ1 [2]：IRQ2 …… [31]：IRQ31 读出值表示当前中断挂起状态

5.7.5 NVIC_IPR0 中断优先级控制寄存器 0

Address offset: 0x400 Reset value: 0x0000 0000

位域	名称	权限	功能描述
31:30	IPRIRQ3	RW	中断 IRQ3 的优先级，00 优先级最高，11 优先级最低
29:24	RFU	-	保留位，请保持默认值
23:22	IPRIRQ2	RW	中断 IRQ2 的优先级，00 优先级最高，11 优先级最低
21:16	RFU	-	保留位，请保持默认值
15:14	IPRIRQ2	RW	中断 IRQ1 的优先级，00 优先级最高，11 优先级最低
13:8	RFU	-	保留位，请保持默认值
7:6	IPRIRQ0	RW	中断 IRQ0 的优先级，00 优先级最高，11 优先级最低
5:0	RFU	-	保留位，请保持默认值

5.7.6 NVIC_IPR1 中断优先级控制寄存器 1

Address offset: 0x404 Reset value: 0x0000 0000

位域	名称	权限	功能描述
31:30	IPRIRQ7	RW	中断 IRQ7 的优先级，00 优先级最高，11 优先级最低
29:24	RFU	-	保留位，请保持默认值
23:22	IPRIRQ6	RW	中断 IRQ6 的优先级，00 优先级最高，11 优先级最低
21:16	RFU	-	保留位，请保持默认值
15:14	IPRIRQ5	RW	中断 IRQ5 的优先级，00 优先级最高，11 优先级最低
13:8	RFU	-	保留位，请保持默认值
7:6	IPRIRQ4	RW	中断 IRQ4 的优先级，00 优先级最高，11 优先级最低
5:0	RFU	-	保留位，请保持默认值

5.7.7 NVIC_IPR2 中断优先级控制寄存器 2

Address offset: 0x408 Reset value: 0x0000 0000

位域	名称	权限	功能描述
31:30	IPRIRQ11	RW	中断 IRQ11 的优先级，00 优先级最高，11 优先级最低
29:24	RFU	-	保留位，请保持默认值
23:22	IPRIRQ10	RW	中断 IRQ10 的优先级，00 优先级最高，11 优先级最低
21:16	RFU	-	保留位，请保持默认值
15:14	IPRIRQ9	RW	中断 IRQ9 的优先级，00 优先级最高，11 优先级最低
13:8	RFU	-	保留位，请保持默认值
7:6	IPRIRQ8	RW	中断 IRQ8 的优先级，00 优先级最高，11 优先级最低
5:0	RFU	-	保留位，请保持默认值

5.7.8 NVIC_IPR3 中断优先级控制寄存器 3

Address offset: 0x40C Reset value: 0x0000 0000

位域	名称	权限	功能描述
31:30	IPRIRQ15	RW	中断 IRQ15 的优先级，00 优先级最高，11 优先级最低
29:24	RFU	-	保留位，请保持默认值
23:22	IPRIRQ14	RW	中断 IRQ14 的优先级，00 优先级最高，11 优先级最低
21:16	RFU	-	保留位，请保持默认值
15:14	IPRIRQ13	RW	中断 IRQ13 的优先级，00 优先级最高，11 优先级最低
13:8	RFU	-	保留位，请保持默认值
7:6	IPRIRQ12	RW	中断 IRQ12 的优先级，00 优先级最高，11 优先级最低
5:0	RFU	-	保留位，请保持默认值

5.7.9 NVIC_IPR4 中断优先级控制寄存器 4

Address offset: 0x410 Reset value: 0x0000 0000

位域	名称	权限	功能描述
31:30	IPRIRQ19	RW	中断 IRQ19 的优先级，00 优先级最高，11 优先级最低
29:24	RFU	-	保留位，请保持默认值
23:22	IPRIRQ18	RW	中断 IRQ18 的优先级，00 优先级最高，11 优先级最低
21:16	RFU	-	保留位，请保持默认值
15:14	IPRIRQ17	RW	中断 IRQ17 的优先级，00 优先级最高，11 优先级最低
13:8	RFU	-	保留位，请保持默认值
7:6	IPRIRQ16	RW	中断 IRQ16 的优先级，00 优先级最高，11 优先级最低
5:0	RFU	-	保留位，请保持默认值

5.7.10 NVIC_IPR5 中断优先级控制寄存器 5

Address offset: 0x414 Reset value: 0x0000 0000

位域	名称	权限	功能描述
31:30	IPRIRQ23	RW	中断 IRQ23 的优先级，00 优先级最高，11 优先级最低
29:24	RFU	-	保留位，请保持默认值
23:22	IPRIRQ22	RW	中断 IRQ22 的优先级，00 优先级最高，11 优先级最低
21:16	RFU	-	保留位，请保持默认值
15:14	IPRIRQ21	RW	中断 IRQ21 的优先级，00 优先级最高，11 优先级最低
13:8	RFU	-	保留位，请保持默认值
7:6	IPRIRQ20	RW	中断 IRQ20 的优先级，00 优先级最高，11 优先级最低
5:0	RFU	-	保留位，请保持默认值

5.7.11 NVIC_IPR6 中断优先级控制寄存器 6

Address offset: 0x418 Reset value: 0x0000 0000

位域	名称	权限	功能描述
31:30	IPRIRQ27	RW	中断 IRQ27 的优先级，00 优先级最高，11 优先级最低
29:24	RFU	-	保留位，请保持默认值
23:22	IPRIRQ26	RW	中断 IRQ26 的优先级，00 优先级最高，11 优先级最低
21:16	RFU	-	保留位，请保持默认值
15:14	IPRIRQ25	RW	中断 IRQ25 的优先级，00 优先级最高，11 优先级最低
13:8	RFU	-	保留位，请保持默认值
7:6	IPRIRQ24	RW	中断 IRQ24 的优先级，00 优先级最高，11 优先级最低
5:0	RFU	-	保留位，请保持默认值

5.7.12 NVIC_IPR7 中断优先级控制寄存器 7

Address offset: 0x41C Reset value: 0x0000 0000

位域	名称	权限	功能描述
31:30	IPRIRQ31	RW	中断 IRQ31 的优先级，00 优先级最高，11 优先级最低
29:24	RFU	-	保留位，请保持默认值
23:22	IPRIRQ30	RW	中断 IRQ30 的优先级，00 优先级最高，11 优先级最低
21:16	RFU	-	保留位，请保持默认值
15:14	IPRIRQ29	RW	中断 IRQ29 的优先级，00 优先级最高，11 优先级最低
13:8	RFU	-	保留位，请保持默认值
7:6	IPRIRQ28	RW	中断 IRQ28 的优先级，00 优先级最高，11 优先级最低
5:0	RFU	-	保留位，请保持默认值

6 RAM 存储器

6.1 概述

CW32F020 内部集成 8KB 嵌入式 RAM 供用户使用，用来存放程序执行过程中的各种数据。RAM 的起始地址为 0x2000 0000，数据在 RAM 中以小端模式存储，即最低字节地址空间存放数据的最低有效字节数据。

6.2 主要特性

- 支持以字节（8bit）、半字（16bit）或全字（32 bit）3 种位宽进行访问
- 零等待延迟，能够被 CPU 和 DMA 以最大的系统时钟频率进行访问
- 支持奇偶校验功能

6.3 RAM 存储器操作

用户可执行的 RAM 存储器操作包括：读操作、写操作。

对 RAM 的读写操作支持 8bit、16bit 和 32bit 三种位宽，用户程序可以通过直接访问绝对地址的方式完成读写，但要注意读写的数据位宽必须和对应地址边界对齐，否则读写操作无效，并会导致 HardFault 硬件错误异常。

6.3.1 读操作

读操作支持3种不同位宽，可采用直接访问绝对地址方式读取，但要注意读取的数据位宽必须和对应地址边界对齐。

代码示例：

8bit 读：

```
tempdata = * ( ( uint8_t * ) 0x2000 0001 ) ;
```

16bit 读：

```
tempdata = * ( ( uint16_t * ) 0x2000 0002 ) ;
```

32bit 读：

```
tempdata = * ( ( uint32_t * ) 0x2000 0004 ) .
```

6.3.2 写操作

写操作支持 3 种不同位宽，可采用直接访问绝对地址的方式写入数据，但要注意写入的数据位宽必须和对应地址边界对齐。

代码示例：

8bit 写：

```
* ( ( uint8_t * ) 0x2000 0001 ) = 0x12;
```

16bit 写：

```
* ( ( uint16_t * ) 0x2000 0002 ) = 0x1234;
```

32bit 写：

```
* ( ( uint32_t * ) 0x2000 0004 ) = 0x1234 5678.
```

6.4 奇偶校验功能

CW32F020 支持 RAM 的奇偶校验功能，上电后奇偶校验功能默认打开，用户不可配置。

每字节 RAM 数据实际存放在 9bit 的物理空间中，包括 8bit 数据位和 1bit 奇偶校验位。

CPU 在对 RAM 进行写入时，RAM 的奇偶校验单元会计算校验位并写入对应的校验位空间。在读取 RAM 数据时，数据连同校验位一起被读取，CPU 对数据进行每字节的奇偶校验，并将计算的校验位和读取的校验位进行比较，如果一致则说明 RAM 中数据正确，如果不一致则奇偶校验错误标志 RAM_ISR.PARITY 被置位，如果设置 RAM 奇偶校验出错中断使能控制位 RAM_IER.PARITY 为 1，CPU 会响应中断服务。用户程序可设置 RAM_ICR.PARITY 为 0 来清除奇偶校验错误标志。

用户可通过读取奇偶校验出错地址寄存器 RAM_ADDR，以获取发生奇偶校验错误的 RAM 地址。

6.5 寄存器列表

RAM 基地址：RAM_BASE = 0x4002 2400

表 6-1 RAM 寄存器列表

寄存器名称	寄存器地址	寄存器描述
RAM_IER	RAM_BASE + 0x00	中断使能控制寄存器
RAM_ADDR	RAM_BASE + 0x04	奇偶校验出错地址寄存器
RAM_ISR	RAM_BASE + 0x08	中断标志寄存器
RAM_ICR	RAM_BASE + 0x0C	中断标志清除寄存器

6.6 寄存器描述

有关寄存器描述里所使用的缩写，请参见 1 文档约定章节。

6.6.1 RAM_ADDR 奇偶校验出错地址寄存器

Address offset: 0x04 Reset value: 0x2000 0000

位域	名称	权限	功能描述
31:0	ADDR	RO	奇偶校验出错的 RAM 所在的地址

6.6.2 RAM_IER 中断使能控制寄存器

Address offset: 0x00 Reset value: 0x0000 0001

位域	名称	权限	功能描述
31:2	RFU	-	保留位，请保持默认值
1	PARITY	RW	RAM 奇偶校验出错中断使能控制 0：禁止 1：使能
0	EN	RO	RAM 奇偶校验使能标志 0：禁止 1：使能

6.6.3 RAM_ISR 中断标志寄存器

Address offset: 0x08 Reset value: 0x0000 0000

位域	名称	权限	功能描述
31:1	RFU	-	保留位，请保持默认值
0	PARITY	RO	奇偶校验错误标志 0：未发生奇偶校验错误 1：已发生奇偶校验错误

6.6.4 RAM_ICR 中断标志清除寄存器

Address offset: 0x0C Reset value: 0x0000 0001

位域	名称	权限	功能描述
31:1	RFU	-	保留位，请保持默认值
0	PARITY	R1W0	奇偶校验错误标志清除控制 W0：清除奇偶校验错误标志 W1：无功能

7 FLASH 存储器

7.1 概述

CW32F020 内部集成 32KB 嵌入式 FLASH 供用户使用，可用来存储应用程序和用户数据。芯片支持对 FLASH 存储器的读、擦除和写操作，支持擦写保护和读保护；芯片内置 FLASH 编程所需的高压 BOOST 电路，无须额外提供编程电压。

7.2 主要特性

- 支持以字节（8bit）、半字（16bit）或全字（32 bit）3 种位宽进行访问
- 支持读、擦除和写操作
- 支持擦写保护、读保护
- 支持缓存和预取功能
- 支持低功耗模式

7.3 FLASH 存储器组织

CW32F020 内部集成 32KB 用户可访问的 FLASH 存储器，按每页 512 字节进行分页管理，共 64 页，用户可以对 FLASH 进行整页擦除和逐字节编程操作。

除用户可访问的 32KB 的 FLASH 存储器外，CW32F020 还集成 2.5KB 的启动程序存储器，该存储器出厂时固化了 BootLoader 等代码，用户不可访问该存储器。

7.4 FLASH 存储器读等待周期配置

CW32F020 内部的 FLASH 存储器支持最快 24MHz 频率的操作时钟，当配置的 HCLK 频率大于 24MHz 时，需通过 FLASH 控制寄存器 FLASH_CR2 的 WAIT 位域来配置插入的等待 HCLK 周期个数，配置规则如下表所示：

表 7-1 FLASH 读等待周期配置

FLASH_CR2.WAIT	取指周期等待 HCLK 周期个数	HCLK 频率
000	1 个	<= 24MHz
001	2 个	> 24MHz，且 <= 48MHz
010	3 个	> 48MHz，且 <= 72MHz

注：

配置 FLASH 读等待周期时，必须先使能 FLASH 配置时钟。

7.5 FLASH 存储器操作

用户可执行的 FLASH 存储器操作包括：读操作、擦除、写（编程）操作。

对 FLASH 的读写操作支持 8bit、16bit 和 32bit 三种位宽，用户程序可以通过直接访问绝对地址的方式完成读写，但要注意读写的数据位宽必须和对应地址边界对齐，否则会导致 HardFault 硬件错误异常。

注：

在进行页擦除或写操作时，必须设置 FLASH_CR2. CACHE 和 FLASH_CR2. FETCH 为 0，以禁止缓存和预取指功能。

7.5.1 页擦除

FLASH 的页擦除操作的最小单位为 1 页，即 512 字节。页擦除操作完成后，该页所有地址空间的数据内容均为 0xFF。

如果对未解锁的 FLASH 页面进行页擦除操作，会操作失败，同时 FLASH_ISR.PAGELOCK 标志位会被硬件置位，如果设置 FLASH_IER.PAGELOCK 为 1，则 CPU 会执行对应的中断服务程序。用户可通过设置 FLASH_ICR. PAGELOCK 为 0 来清除 FLASH_ISR.PAGELOCK 中断标志。

如果程序在 FLASH 中运行，且对 PC（程序指针）所在页面的存储空间进行页擦除操作，会操作失败，同时 FLASH_ISR.PC 标志位会被硬件置位，如果设置 FLASH_IER.PC 为 1，则 CPU 会执行对应的中断服务程序。用户可通过设置 FLASH_ICR.PC 为 0 来清除 FLASH_ISR.PC 中断标志。

FLASH 页擦除操作步骤如下：

步骤 1：设置 FLASH_CR2. CACHE 和 FLASH_CR2. FETCH 为 0，禁止缓存和预取指；

注：

FLASH_CR2 寄存器具有 KEY 保护特性，写入的数据高 16bit 数据必须是 0x5A5A，否则无法写入。

步骤 2：设置 FLASH 操作模式为页擦除模式，向 FLASH 控制寄存器 FLASH_CR1 的工作模式位域 MODE 写入 0x02；

注：

FLASH_CR1 寄存器具有 KEY 保护特性，写入的数据高 16bit 数据必须是 0x5A5A，否则无法写入。

步骤 3：解锁待擦除页的擦除保护，参见表 7-2 FLASH_PAGELOCK 擦写锁定保护，向擦写锁定寄存器 FLASH_PAGELOCK 对应的锁定位域 LOCKx 写入 1；

注：

FLASH_PAGELOCK 寄存器具有 KEY 保护特性，写入的数据高 16bit 数据必须是 0x5A5A，否则无法写入。

步骤 4：对待擦除页内的任意地址写入任意数据，触发对该页的擦除操作。

注：

写入数据位宽需要和对应地址边界对齐。

代码示例：

```
*( ( uint8_t * ) 0x0000 0001 ) = 0x55;
```

步骤 5：等待擦除完成，查询 FLASH 忙标志位 FLASH_CR1.BUSY 变成 0；

步骤 6：如需擦除其它页，则重复步骤 3 至步骤 5；

步骤 7：擦除操作完成后，如需恢复对 FLASH 的锁定保护，需向擦写锁定寄存器 FLASH_PAGELOCK 对应的锁定位域 LOCKx 写入 0，启动锁定保护功能，保护 FLASH 内容不被误擦写；

步骤 8：设置 FLASH 操作模式为读模式，向 FLASH 控制寄存器 FLASH_CR1 的工作模式位域 MODE 写入 0x00；

步骤 9：设置 FLASH_CR2. CACHE 和 FLASH_CR2. FETCH 为 1，使能缓存和预取指。

7.5.2 写操作

基于嵌入式 FLASH 的特性，写操作只能将 FLASH 存储器中位数据由‘1’改写为‘0’，不能由‘0’改写为‘1’，因此在写数据之前先要对对应地址所在页进行擦除操作。

对 FLASH 写操作必须遵循以下三个原则：

- 不可对数据位内容为‘0’的地址写入
- 不可对锁定区域内的地址写入
- 不可对 PC（程序指针）所在的页的地址写入

CW32F020 在执行写操作时，会检查存储空间内容是否全为‘1’。FLASH 的 8bit、16bit、32bit 三种写操作位宽，需要检查的字节数分别为 1Byte、2Byte、4Byte。

如果对内部不全是‘1’的存储空间进行写入操作，会写入失败，同时 FLASH_ISR.PROG 标志位会被硬件置位，如果设置 FLASH_IER.PROG 为 1，则 CPU 会执行对应的中断服务程序。用户可通过设置 FLASH_ICR.PROG 为 0 来清除 FLASH_ISR.PROG 中断标志。

如果对未解锁的 FLASH 页面的存储空间进行写操作，会操作失败，同时 FLASH_ISR.PAGELOCK 标志位会被硬件置位，如果设置 FLASH_IER.PAGELOCK 为 1，则 CPU 会执行对应的中断服务程序。用户可通过设置 FLASH_ICR.PAGELOCK 为 0 来清除 FLASH_ISR.PAGELOCK 中断标志。

如果程序在 FLASH 中运行，且对 PC（程序指针）所在页面的存储空间进行写操作，会操作失败，同时 FLASH_ISR.PC 标志位会被硬件置位，如果设置 FLASH_IER.PC 为 1，则 CPU 会执行对应的中断服务程序。用户可通过设置 FLASH_ICR.PC 为 0 来清除 FLASH_ISR.PC 中断标志。

FLASH 写操作步骤如下：

步骤 1：设置 FLASH_CR2. CACHE 和 FLASH_CR2. FETCH 为 0，禁止缓存和预取指；

注：

FLASH_CR2 寄存器具有 KEY 保护特性，写入的数据高 16bit 数据必须是 0x5A5A，否则无法写入。

步骤 2：设置 FLASH 操作模式为写（编程）模式，向 FLASH 控制寄存器 FLASH_CR1 的工作模式位域 MODE 写入 0x01；

注：

FLASH_CR1 寄存器具有 KEY 保护特性，写入的数据高 16bit 数据必须是 0x5A5A，否则无法写入。

步骤 3：解锁待写页的擦除保护，参见表 7-2 FLASH_PAGELOCK 擦写锁定保护，向擦写锁定寄存器 FLASH_PAGELOCK 对应的锁定位域 LOCKx 写入 1；

注：

FLASH_PAGELOCK 寄存器具有 KEY 保护特性，写入的数据高 16bit 数据必须是 0x5A5A，否则无法写入。

步骤 4：将待写入的数据写入目标地址，触发 FLASH 的写入操作，注意写入数据位宽需要和对应地址边界对齐；

8bit 位宽代码示例：

```
*( ( uint8_t * ) 0x0000 0001 ) = 0x55;
```

16bit 位宽代码示例：

```
*( ( uint16_t * ) 0x0000 0002 ) = 0x1234;
```

32bit 位宽代码示例：

```
*( ( uint32_t * ) 0x0000 0004 ) = 0x1234 5678;
```

步骤 5：等待编程完成，查询等待 FLASH 忙标志位 FLASH_CR1.BUSY 变成 0；

步骤 6：如需写更多数据到当前已解除锁定区域的地址范围内，重复步骤 4 至步骤 5；如写入地址超出当前已解除锁定区域的地址范围，重复步骤 3 至步骤 5；

步骤 7：写操作完成后，如需恢复对 FLASH 的锁定保护，向擦写锁定寄存器 FLASH_PAGELOCK 对应的锁定位域 LOCKx 写入 0，启动锁定保护功能，保护 FLASH 内容不被误擦写；

步骤 8：设置 FLASH 操作模式为读模式，向 FLASH 控制寄存器 FLASH_CR1 的工作模式位域 MODE 写入 0x00；

步骤 9：设置 FLASH_CR2. CACHE 和 FLASH_CR2. FETCH 为 1，使能缓存和预取指。

7.5.3 读操作

CW32F020 对 FLASH 的读操作支持 3 种不同位宽，可采用直接访问绝对地址方式读取，注意读取的数据位宽必须和对应地址边界对齐。

8bit 位宽代码示例：

```
tempdata = *( ( uint8_t * ) 0x0000 0001 ) ;
```

16bit 位宽代码示例：

```
tempdata = *( ( uint16_t * ) 0x0000 0002 ) ;
```

32bit 位宽代码示例：

```
tempdata = *( ( uint32_t * ) 0x0000 0004 ) ;
```

7.6 FLASH 存储器保护

FLASH 存储器具有擦写保护和读保护功能。

擦写保护包括锁定页擦写保护和 PC 地址页擦写保护，处于保护状态的页面不能被擦写，可避免 FLASH 内容被意外改写。

读保护以整片 FLASH 为保护对象，不支持单页保护，可避免用户代码被非法读取。

7.6.1 擦写保护

CW32F020 通过设置擦写锁定寄存器 FLASH_PAGELOCK 来实现 FLASH 页面的擦写锁定，处于锁定状态的页面不能进行页擦除或写操作。

CW32F020 内部 FLASH 存储器被划分为 64 页，每 8 页对应擦写锁定寄存器 FLASH_PAGELOCK 的 1 个 LOCKx 锁定位。LOCKx 域共 8 位，可实现全部 64 页 FLASH 存储器的锁定保护。擦写锁定寄存器 FLASH_PAGELOCK 的各位域与 FLASH 锁定页面的对应关系如下表所示：

表 7-2 FLASH_PAGELOCK 擦写锁定保护

位	位域名称	锁定页面
7	LOCK7	Page56 - Page63
6	LOCK6	Page48 - Page55
5	LOCK5	Page40 - Page47
4	LOCK4	Page32 - Page39
3	LOCK3	Page24 - Page31
2	LOCK2	Page16 - Page23
1	LOCK1	Page8 - Page15
0	LOCK0	Page0 - Page7

对 FLASH 进行页擦除和写操作时，必须先通过设置对应的 FLASH_PAGELOCK.LOCKx 位域为 1 来解锁对应页面。解锁操作示例如下：

锁定 Page0-Page7 代码示例：

```
CW_FLASH->PAGELOCK = 0x5A5A 0000 | ( CW_FLASH->PAGELOCK & 0x0000 FFFE );
```

解锁 Page32-Page39 代码示例：

```
CW_FLASH->PAGELOCK = 0x5A5A 0000 | ( CW_FLASH->PAGELOCK | 0x0000 0010 );
```

注：

FLASH_PAGELOCK 寄存器具有 KEY 保护特性，写入的数据高 16bit 数据必须是 0x5A5A，否则无法写入。

对未解锁的 FLASH 页直接执行页擦除或者写操作，会操作失败并产生中断标志，请参见 7.5.1 页擦除和 7.5.2 写操作。

7.6.2 擦写 PC 页保护

CW32F020 的 FLASH 存储器支持擦写 PC 页保护功能。

当用户程序运行 FLASH 时，如果当前程序指针 PC 正好位于待擦写的 FLASH 地址页范围内，则该擦写操作失败，同时 FLASH_ISR.PC 标志位会被硬件置位，如果设置 FLASH_IER.PC 为 1，则 CPU 会执行对应的中断服务程序。用户可通过设置 FLASH_ICR.PC 为 0 来清除 FLASH_ISR.PC 中断标志。

7.6.3 读保护

CW32F020 支持 FLASH 读保护功能，设置读保护后，无法通过 ISP 或 SWD 方式对 FLASH 进行读取操作。读保护只支持整片 FLASH 保护，不支持按页保护。

读保护分为 4 个保护等级，当前保护等级可通过读取 FLASH 控制寄存器 FLASH_CR1 的安全位域 SECURITY 来获取，安全位域是只读属性，不能修改。读保护等级可通过 ISP 指令进行设定，请参阅 ISP 编程文档。

CPU 从 FLASH 中取指操作和程序对 FLASH 的读取操作，不受 FLASH 读保护功能影响。

FLASH 的读保护功能如下表所示：

表 7-3 FLASH 的读保护

保护等级	FLASH_CR1.SECURITY	功能描述
Level0	00	未设置读保护。 可以通过 SWD 或 ISP 方式对 FLASH 进行读取操作。
Level1	01	FLASH 内容不可通过 SWD 或 ISP 方式读取。 可通过 ISP 或 SWD 方式将保护等级降低到 Level0，降级之后 FLASH 中内容为全 0xFF，即处于整片擦除的空片状态。
Level2	10	FLASH 内容不可通过 SWD 或 ISP 方式读取。 仅可通过 ISP 方式将保护等级降低，但降级之后 FLASH 中内容为全 0xFF，即处于整片擦除的空片状态。
Level3	11	FLASH 内容不可通过 SWD 或 ISP 方式读取。 ISP 和 SWD 降级功能都被禁止。 在此保护等级下，芯片只能进行一次编程。

7.6.4 FLASH 存储器编程

CW32F020 内部集成 FLASH 存储器支持通过 SWD 或 ISP 方式进行 FLASH 的擦除和编程。SWD 方式是利用调试工具，通过 SWD 接口进行 FLASH 的擦除和编程。ISP 方式是通过芯片出厂前预装的 BootLoader 代码进行 FLASH 的擦除和编程。ISP 提供了快速和高效的在线擦除和编程方法，请参阅 ISP 编程文档。

7.7 注意事项

为正确操作 FLASH 和提高 FLASH 的访问效率及使用寿命，用户在编程应用时需要注意以下事项：

- 地址对齐要求

  地址边界对齐，即使用 16bit 位宽访问 FLASH 时的地址必须是偶地址，使用 32bit 位宽时的地址必须是 4 的倍数地址。

  正确地址对齐的代码示例：

  8bit 读取：

  ```
  tempdata = *( ( uint8_t * ) 0x0000 0001 ) ;
  ```

  16bit 读取：

  ```
  tempdata = *( ( uint16_t * ) 0x0000 0002 ) ;
  ```

  32bit 读取：

  ```
  tempdata = *( ( uint32_t * ) 0x0000 0004 ) ;
  ```

  错误地址对齐的代码示例：

  16bit 读取：

  ```
  tempdata = *( ( uint16_t * ) 0x0000 0001 ) ;
  ```

  32bit 读取：

  ```
  tempdata = *( ( uint32_t * ) 0x0000 0003 ) ;
  ```

- 操作完成标志查询

  当 CPU 从 FLASH 中取指并运行时，如果执行对 FLASH 的页擦除 / 写操作，CPU 会自动停止下一条指令存取，硬件自动等待擦写操作完成（FLASH_CR1.BUSY 状态位变成 0），故用户程序不必循环查询操作完成标志来判断操作是否完成。

  当 CPU 从 RAM 中取指并运行时，如果执行对 FLASH 的页擦除 / 写操作，在执行该操作的同时，CPU 会进行下一条指令存取，为保证程序下一条指令的执行正确性，用户程序必须在对 FLASH 擦写操作后循环查询 FLASH_CR1.BUSY 标志位，直到 FLASH_CR1.BUSY 标志位变成 0 后方可执行后续的任务。

- 使用寿命

  基于嵌入式 FLASH 的特性，FLASH 的操作次数和存储时间是有限的，用户在应用程序中应尽量避免频繁对某一页或某一地址的 FLASH 存储器进行擦写操作，以保证数据的可靠存储。具体寿命数据请参阅数据手册。

- 数据存储模式

  CW32F020 应用规定，数据在 FLASH 中以小端模式存储，即最低字节地址空间存放数据的最低有效字节数据。

- 预取和缓存配置

  CW32F020 的 FLASH 存储器带有预取缓存功能，可通过设置 FLASH_CR2.CACHE 为 1 和 FLASH_CR2.FETCH 为 1，分别使能 FLASH 的缓存和预取功能，提高 MCU 对 FLASH 的访问效率。

- 低功耗特性

  FLASH 具有低功耗特性，通过设置 FLASH_CR1.STANDBY 为 1，可以让 FLASH 在系统进入深度休眠模式后自动进入低功耗状态，保证产品在低功耗模式下具有更低功率消耗。

7.8 寄存器列表

FLASH 基地址：FLASH_BASE = 0x4002 2000

表 7-4 FLASH 寄存器列表

寄存器名称	寄存器地址	寄存器描述
FLASH_CR1	FLASH_BASE + 0x00	控制寄存器 1
FLASH_CR2	FLASH_BASE + 0x04	控制寄存器 2
FLASH_PAGELOCK	FLASH_BASE + 0x08	擦写锁定寄存器
FLASH_IER	FLASH_BASE + 0x20	中断使能寄存器
FLASH_ISR	FLASH_BASE + 0x24	中断标志寄存器
FLASH_ICR	FLASH_BASE + 0x28	中断标志清零寄存器

7.9 寄存器描述

有关寄存器描述里所使用的缩写，请参见 1 文档约定章节。

7.9.1 FLASH_CR1 控制寄存器 1

Address offset: 0x00 Reset value: 0x0000 0010

位域	名称	权限	功能描述
31:16	KEY	WO	仅当 KEY 为 0x5A5A 时，对该寄存器的写操作有效
15:8	RFU	-	保留位，请保持默认值
7:6	SECURITY	RO	当前保护等级 00：Level0，ISP 可读写，SWD 可读写 01：Level1，ISP 可降级，SWD 可降级；数据不可读出 10：Level2，ISP 可降级，SWD 无功能；数据不可读出 11：Level3，ISP 无功能，SWD 无功能；数据不可读出
5	BUSY	RO	擦写状态标志 0：擦写操作已完成 1：擦写操作未完成
4	STANDBY	RW	低功耗使能控制 0：当系统进入 DeepSleep 模式，FLASH 不进入低功耗模式 1：当系统进入 DeepSleep 模式，FLASH 进入低功耗模式 注：建议该值配置为 1，否则会导致 DeepSleep 模式功耗偏大
3:2	RFU	-	保留位，请保持默认值
1:0	MODE	RW	操作模式配置 00：读模式，Read 01：写模式，Program 10：页擦模式，PageErase 11：无效

7.9.2 FLASH_CR2 控制寄存器 2

Address offset: 0x04 Reset value: 0x0000 0000

位域	名称	权限	功能描述
31:16	KEY	WO	仅当 KEY 为 0x5A5A 时，对该寄存器的写操作有效
15:5	RFU	-	保留位，请保持默认值
4	CACHE	RW	缓存使能控制 0：禁止缓存 1：使能缓存 注：当 WAIT 不为 0 时，建议使能该功能
3	FETCH	RW	预取指使能控制 0：禁止预取指 1：使能预取指 注：建议使能该功能
2:0	WAIT	RW	FLASH 取指周期配置 000：1 个 HCLK 周期，适用于 HCLK <= 24MHz 001：2 个 HCLK 周期，适用于 24MHz < HCLK <= 48MHz 010：3 个 HCLK 周期，适用于 48MHz < HCLK <= 72MHz

7.9.3 FLASH_PAGELOCK 擦写锁定寄存器

Address offset: 0x08 Reset value: 0x0000 0000

位域	名称	权限	功能描述
31:16	KEY	WO	仅当 KEY 为 0x5A5A 时，对该寄存器的写操作有效
15:8	RFU	-	保留位，请保持默认值
7	LOCK7	RW	Page56 – Page63 擦写锁定配置 0：锁定，不可擦写 1：开放，可以擦写
6	LOCK6	RW	Page48– Page55 擦写锁定配置 0：锁定，不可擦写 1：开放，可以擦写
…	…	…	…
1	LOCK1	RW	Page8 – Page15 擦写锁定配置 0：锁定，不可擦写 1：开放，可以擦写
0	LOCK0	RW	Page0 – Page7 擦写锁定配置 0：锁定，不可擦写 1：开放，可以擦写

7.9.4 FLASH_IER 中断使能寄存器

Address offset: 0x20 Reset value: 0x0000 0000

位域	名称	权限	功能描述
31:5	RFU	-	保留位，请保持默认值
4	PROG	RW	编程错误中断使能控制 0：禁止 1：使能
3:2	RFU	-	保留位，请保持默认值
1	PAGELOCK	RW	擦写 PAGELOCK 锁定的页面中断使能控制 0：禁止 1：使能
0	PC	RW	擦写 PC 所在页面中断使能控制 0：禁止 1：使能

7.9.5 FLASH_ISR 中断标志寄存器

Address offset: 0x24 Reset value: 0x0000 0000

位域	名称	权限	功能描述
31:5	RFU	-	保留位，请保持默认值
4	PROG	RO	编程错误标志 0：当前待写入地址的每个字节的内容全是 0xFF 1：当前待写入地址的每个字节的内容不全是 0xFF
3:2	RFU	-	保留位，请保持默认值
1	PAGELOCK	RO	擦写 PAGELOCK 锁定的页面中断标志 0：当前擦写地址位于 PAGELOCK 锁定的页面之外 1：当前擦写地址位于 PAGELOCK 锁定的页面之内
0	PC	RO	擦写 PC 指针所在页面中断标志 0：当前擦写地址位于 PC 指针所在的页面之外 1：当前擦写地址位于 PC 指针所在的页面之内

7.9.6 FLASH_ICR 中断标志清除寄存器

Address offset: 0x28 Reset value: 0x0000 000F

位域	名称	权限	功能描述
31:5	RFU	-	保留位，请保持默认值
4	PROG	WO	编程错误标志清除 W0：清除编程错误标志 W1：无功能
3:2	RFU	-	保留位，请保持默认值
1	PAGELOCK	R1W0	擦写 PAGELOCK 锁定的页面中断标志清除 W0：清除擦写 PAGELOCK 锁定的页面中断标志 W1：无功能
0	PC	R1W0	擦写 PC 指针所在页面中断标志清除 W0：清除擦写 PC 指针所在页面中断标志 W1：无功能

8 直接内存访问（DMA）

8.1 概述

CW32F020 支持直接内存访问（DMA），无需 CPU 干预，即可实现外设和存储器之间、外设和外设之间、存储器和存储器之间的高速数据传输。DMA 控制器内部的优先级仲裁器，可实现 DMA 和 CPU 对外设总线控制权的仲裁，以及多 DMA 通道之间的调度执行。

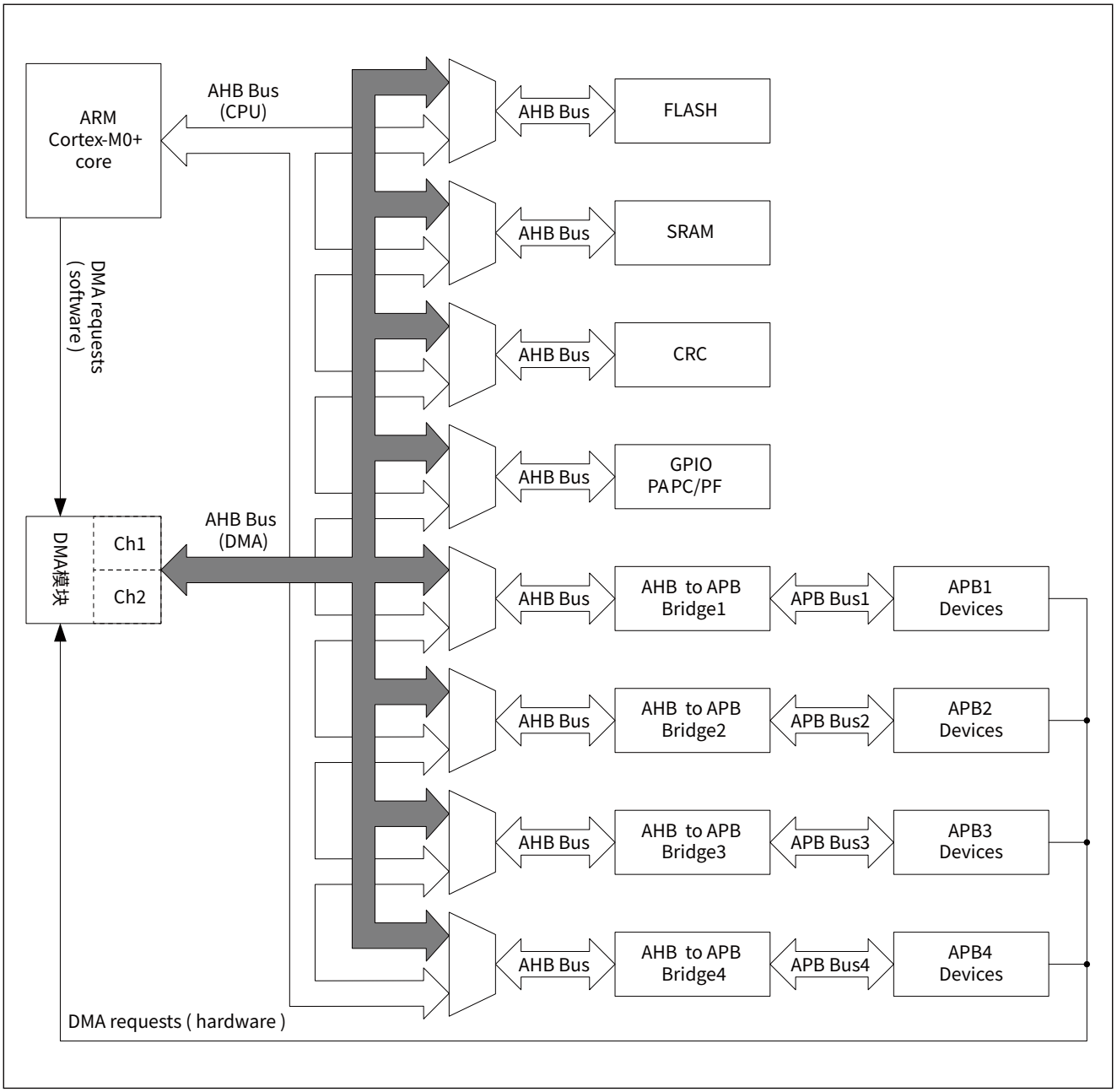
8.2 主要特性

- 2 条独立 DMA 通道
- 3 种数据传输宽度：8bit、16bit、32bit
- 4 种传输模式：软件 BLOCK、软件 BULK、硬件 BLOCK、硬件 BULK
- 源地址及目的地址类型：外设、存储器
- 地址增量方式：固定、自增
- 待传输数据块数量：1 ~ 65535
- 寻址范围：0x0000 0000 ~ 0xFFFF FFFF

8.3 功能框图

DMA 控制器的 AHB 总线通过总线矩阵和外设的 AHB 总线相连，实现 DMA 与外设间的数据互联，包括 FLASH、SRAM、CRC、GPIO 以及各 APB 设备等。DMA 控制器的功能框图如下图所示：

图 8-1 DMA 功能框图



- DMA 控制器

  2 条独立 DMA 通道，通道优先级和通道号绑定，通道号越小优先级越高，通道号越大优先级越低。

- Bus Matrix 总线矩阵

  CPU 系统总线和 DMA 总线接口均连接到总线矩阵上，通过总线矩阵切换和外设的连接。当 CPU 与 DMA 访问不同的 AHB 总线设备或 AHB 到 APB 桥时，数据传输可以同时进行；当 CPU 和 DMA 同时访问同一个 AHB 设备（AHB 到 APB 桥接器下不同设备也被认为是同一 AHB 设备）时，CPU 的优先级高于 DMA。

- AHB to APB 桥接器

  DMA 可以通过总线矩阵及 AHB 到 APB 桥接器实现对 APB 设备的访问。同一个 AHB 到 APB 桥接器下的所有

APB 设备共享同一 AHB 总线，因此被看作是同一个 AHB 设备。当 CPU 和 DMA 同时访问同一 AHB 到 APB 桥接器下的 APB 设备（无论是访问同一个 APB 设备还是不同的 APB 设备）时，CPU 的优先级高于 DMA。

8.4 DMA 传输模式

8.4.1 传输模式概述

DMA 支持 4 种传输模式：软件触发 BLOCK 传输模式，软件触发 BULK 传输模式，硬件触发 BLOCK 传输模式，硬件触发 BULK 传输模式。

- 软件和硬件触发

  部分外设支持硬件 DMA，当它们被配置为 DMA 通道的触发源时，可以产生 DMA 请求（DMA request），硬件触发启动 DMA 传输，无需 CPU 参与。不支持硬件 DMA 的外设，只能配置为软件触发启动 DMA 传输。

  触发方式由 DMA 触发源控制寄存器 DMA_TRIGy（y 是 DMA 通道号，y=1~2，下同）的 TYPE 位域来控制，TYPE 为 0 为软件触发方式，TYPE 为 1 则为硬件触发方式。

- BLOCK 和 BULK 传输模式

  BLOCK 和 BULK 传输模式的区别是，在传输过程中是否允许被更高优先级的传输请求（高优先级 DMA 或 CPU）打断。DMA 控制及状态寄存器 DMA_CSRy 的 TRANS 位域为 1 为 BLOCK 传输模式，为 0 则为 BULK 传输模式。

  BULK 传输模式时，会一次性完成所有数据的传输。 数据传输过程中不会插入传输间隙，传输过程也不会被 CPU 或者其它高优先级 DMA 通道传输请求打断。

  BLOCK 传输模式时，每传输完成 1 个数据块后会插入一个传输间隙，在该传输间隙内，由仲裁器进行传输优先级仲裁。如果有 CPU 或更高优先级的 DMA 通道要访问当前 DMA 通道所占用的外设，仲裁器会释放当前 DMA 通道对该设备的控制权，让 CPU 或者更高优先级的 DMA 通道优先访问，等完成后再继续之前被打断的 DMA 通道传输。

DMA 传输模式，如下表所示：

表 8-1 DMA 传输模式

传输模式	DMA_TRIGy.TYPE	DMA_CSRy.TRANS	是否可以被更高优先级 DMA 通道或 CPU 打断
软件 BLOCK	0	1	YES
软件 BULK	0	0	NO
硬件 BLOCK	1	1	YES
硬件 BULK	1	0	NO

DMA 传输过程

一次完整的 DMA 传输过程，用户需做以下设置：

步骤 1：设置传输模式，DMA_CSRy.TRANS 设置 1 为 BLOCK 传输模式，设置 0 则为 BULK 传输模式；

步骤 2：设置触发方式，DMA_TRIGy.TYPE 设置 0 为软件触发，设置 1 则为硬件触发方式；

步骤 3：配置 DMA 通道数据传输位宽，DMA_CSRy.SIZE，选择 8、16、32bit；

步骤 4：配置 DMA 通道传输数据块数量，DMA_CNTy.CNT，有效范围 1 ~ 65535；

步骤 5：配置 DMA 通道传输数据块大小，DMA_CNTy.REPEAT，请写入 1；

步骤 6：配置 DMA 传输源地址，DMA_SRCADDRy；

步骤 7：配置 DMA 传输目的地址，DMA_DSTADDRy；

步骤 8：配置 DMA 源地址增量方式，DMA_CSRy.SRCINC，固定或自动增加；

步骤 9：配置 DMA 目的地址增量方式，DMA_CSRy.DSTINC，固定或自动增加；

步骤 10：设置 DMA_CSRy.EN 为 1，使能对应的 DMA 通道；

步骤 11：对于软件触发 DMA 传输，设置 DMA_TRIGy.SOFTSRC 启动 DMA 传输，对于硬件触发 DMA 传输不需设置，而只需正确设置用于启动 DMA 的外设。

8.4.2 软件触发 BLOCK 传输模式

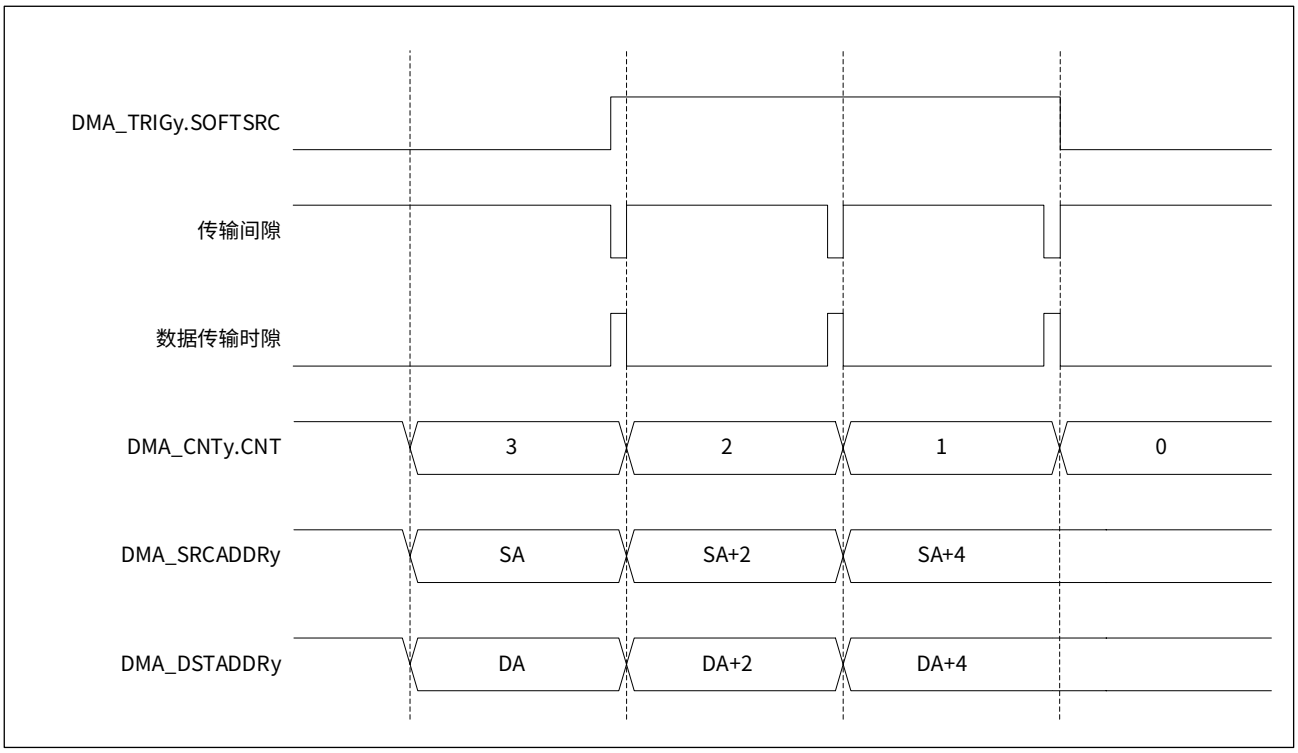
设置 DMA_TRIGy.SOFTSRC 为 1，将立即触发 DMA 通道进行数据传输。

DMA 通道每传输完成 1 个数据块，DMA 插入一个传输间隙，仲裁器在该传输间隙内进行传输优先级仲裁。如果在该传输间隙内，有 CPU 或更高优先级的 DMA 请求要访问当前 DMA 通道所占用的 AHB 设备，仲裁器会释放当前 DMA 通道对该设备的控制权，让 CPU 或者更高优先级的 DMA 通道优先访问；当 CPU 或更高优先级的 DMA 通道释放设备访问权后，本 DMA 通道未完成的数据传输将继续进行。

传输完成后 DMA_TRIGy.SOFTSRC 自动清零。

软件触发 BLOCK 传输模式，传输 3 个数据位宽为 16bit 的数据（DMA_CNTy.REPEAT = 1，DMA_CNTy.CNT = 3），其中，SA 代表源地址，DA 代表目标地址，源地址和目的地址均为地址自增模式，传输过程如下图所示：

图 8-2 软件触发 BLOCK 传输模式



8.4.3 软件触发 BULK 传输模式

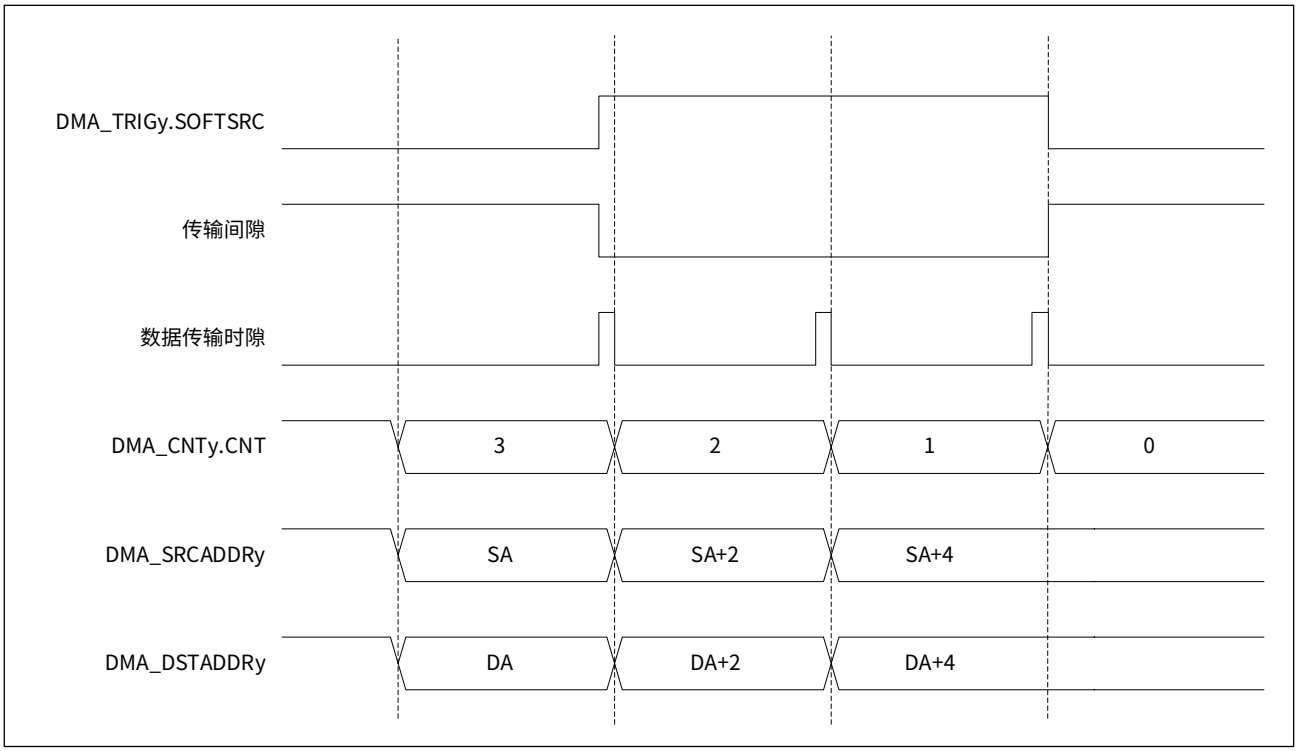
设置 DMA_TRIGy.SOFTSRC 为 1，将立即触发 DMA 进行数据传输，DMA 通道传输完 DMA_CNTy.CNT 个数据块后停止传输并释放外设占用权，传输完成后 DMA_TRIGy.SOFTSRC 自动清零。

软件触发 BULK 传输模式没有传输间隙，在 DMA_CNTy.CNT 个数据块传输完成之前，该 DMA 通道将一直占用外设，CPU 或更高优先级的 DMA 通道只能等待该 DMA 传输完成后才能访问该外设。

在用户应用中，建议每次不要传输太多的数据，否则将可能造成 CPU 在一段时间内完全不能访问被该 DMA 通道占用的外设。

软件触发 BULK 传输模式，传输 3 个数据位宽为 16bit 的数据（DMA_CNTy.REPEAT = 1，DMA_CNTy.CNT = 3），其中，SA 代表源地址，DA 代表目标地址，源地址和目的地址均为地址自增模式，传输过程如下图所示：

图 8-3 软件触发 BULK 传输模式



8.4.4 硬件触发 BLOCK 传输模式

DMA 通道对应的外设产生硬件触发请求信号时，DMA 会传输 1 个数据块，然后等待下一次硬件触发请求信号，直到 DMA 控制器中配置的数据量全部传输完毕。DMA 通道支持的硬件触发请求信号，请参考触发源控制寄存器 DMA_TRIGy 的 HARDSRC 位域。

表 8-2 硬件 DMA 触发传输模式外设触发源

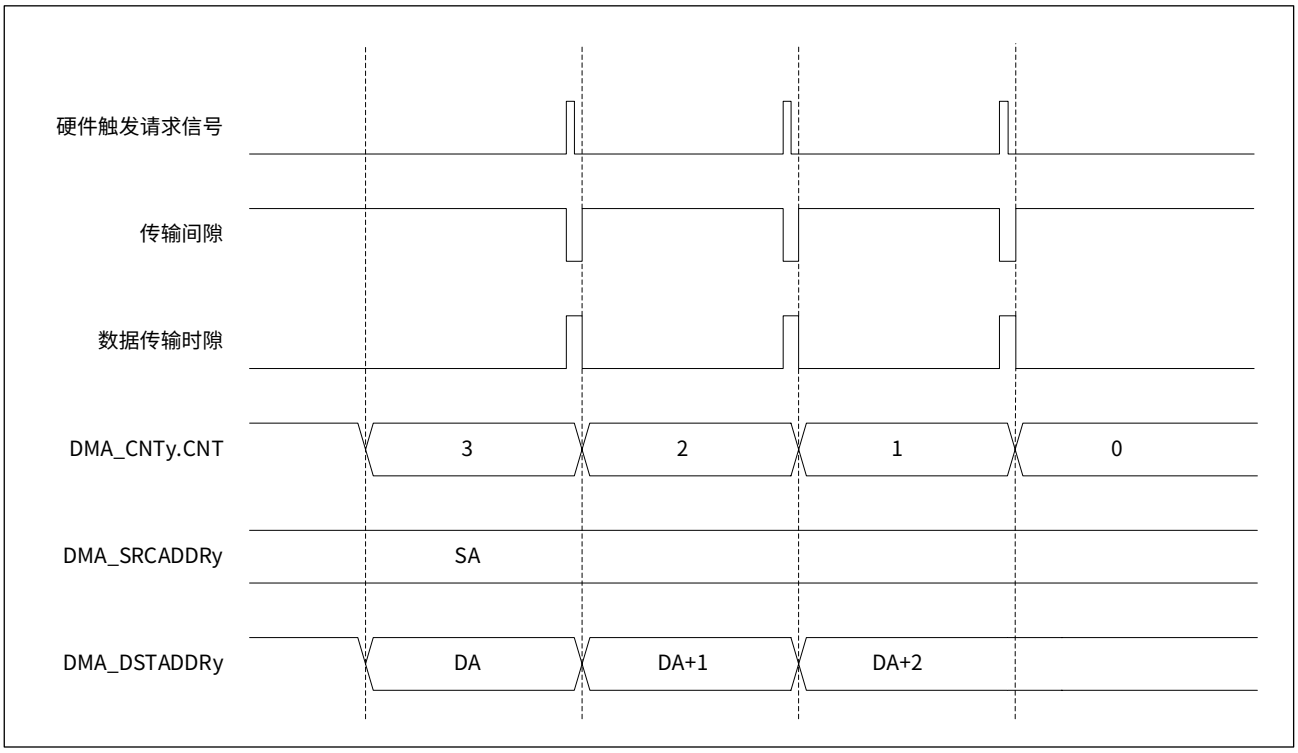
DMA_TRIGy.HARDSRC	DMA 硬件触发源
000000	UART1 接收 Buf 非空标志
000001	UART1 发送 Buf 空标志
000010	UART2 接收 Buf 非空标志
000011	UART2 发送 Buf 空标志
000100	UART3 接收 Buf 非空标志
000101	UART3 发送 Buf 空标志
000110	SPI1 接收 Buf 非空标志
000111	SPI1 发送 Buf 空标志
001000	SPI2 接收 Buf 非空标志
001001	SPI2 发送 Buf 空标志
001010	ADC 转换完成标志
001011	BTIM1 溢出中断标志
001100	BTIM1 触发中断标志
001101	BTIM2 溢出中断标志
001110	BTIM2 触发中断标志
001111	BTIM3 溢出中断标志
010000	BTIM3 触发中断标志
010011	GTIM1 溢出中断标志
010100	GTIM1 触发中断标志
010101	GTIM1 通道 1 比较捕获中断标志
010110	GTIM1 通道 2 比较捕获中断标志
010111	GTIM1 通道 3 比较捕获中断标志
011000	GTIM1 通道 4 比较捕获中断标志
011001	GTIM2 溢出中断标志
011010	GTIM2 触发中断标志
011011	GTIM2 通道 1 比较捕获中断标志
011100	GTIM2 通道 2 比较捕获中断标志

DMA_TRIGy.HARDSRC	DMA 硬件触发源
011101	GTIM2 通道 3 比较捕获中断标志
011110	GTIM2 通道 4 比较捕获中断标志
011111	GTIM3 溢出中断标志
100000	GTIM3 触发中断标志
100001	GTIM3 通道 1 比较捕获中断标志
100010	GTIM3 通道 2 比较捕获中断标志
100011	GTIM3 通道 3 比较捕获中断标志
100100	GTIM3 通道 4 比较捕获中断标志
100101	GTIM4 溢出中断标志
100110	GTIM4 触发中断标志
100111	GTIM4 通道 1 比较捕获中断标志
101000	GTIM4 通道 2 比较捕获中断标志
101001	GTIM4 通道 3 比较捕获中断标志
101010	GTIM4 通道 4 比较捕获中断标志

同软件 BLOCK 传输模式一样，硬件 BLOCK 传输模式每传输完成 1 个数据块，DMA 插入一个传输间隙，允许 CPU 或更高优先级的 DMA 通道传输请求抢占外设控制权。

硬件触发 BLOCK 传输模式，传输 3 个数据位宽为 8bit 的数据（DMA_CNTy.REPEAT = 1，DMA_CNTy.CNT = 3），其中，SA 代表源地址，DA 代表目标地址，源地址固定，目的地址自增，传输过程如下图所示：

图 8-4　硬件触发 BLOCK 传输模式



8.4.5 硬件触发 BULK 传输模式

DMA 通道对应的外设产生硬件触发请求信号时，将立即触发 DMA 进行数据传输，DMA 通道传输完 DMA_CNTy.CNT 个数据块后停止传输并释放外设占用权。

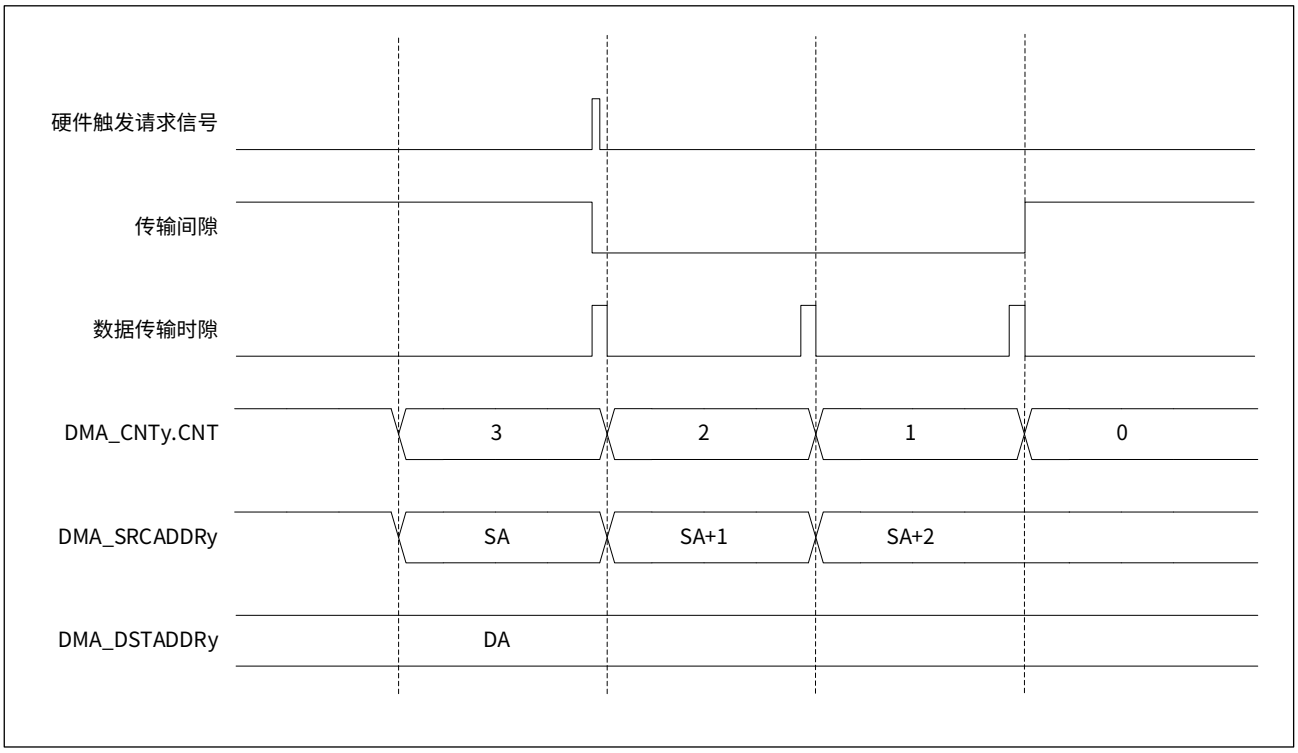
硬件触发 BULK 传输模式没有传输间隙，在 DMA_CNTy.CNT 个数据块传输完成之前，DMA 将一直占据外设访问权，CPU 或更高优先级的 DMA 通道只能等待该操作完成后才能访问该外设。

在用户应用中，建议每次不要传输太多的数据，否则将可能造成 CPU 在一段时间内完全不能访问被该 DMA 通道占用的外设。

DMA 通道支持的硬件触发请求信号，请参见表 8-2 硬件 DMA 触发传输模式外设触发源。

硬件触发 BULK 传输模式，传输 3 个数据位宽为 8bit 的数据（DMA_CNTy.REPEAT = 1，DMA_CNTy.CNT = 3），其中，SA 代表源地址，DA 代表目标地址，源地址自增，目的地址固定，传输过程如下图所示：

图 8-5 硬件触发 BULK 传输模式



8.5 DMA 其它配置

8.5.1 数据宽度

数据位宽可以设置为 8bit、16bit、32bit，通过 DMA_CSRy 寄存器的 SIZE 位域来设置。数据位宽应与 DMA 通道的源地址和目的地址的位宽完全一致。

8.5.2 数据块大小

数据块大小通过 DMA_CNTy 寄存器的 REPEAT 位域配置，建议写入 1。

注：

该位域值在每次传输完成后会自动变为 0，在下次 DMA 传输前需要再写入 1。

8.5.3 数据块数量

待传输的数据块数量可以设置为 1 ~ 65535，通过 DMA_CNTy 寄存器的 CNT 位域来配置。

8.5.4 通道优先级

2 个 DMA 通道的优先级和通道号绑定，通道号越小优先级越高，通道号越大优先级越低。

8.6 DMA 中断

DMA 通道在传输过程可产生 2 个中断标志：传输错误中断标志和传输完成中断标志。

不同 DMA 通道的中断各自独立，通过中断标志寄存器 DMA_ISR 可以获取各通道的中断标志。通道传输错误中断标志对应多个可能的产生原因，具体产生原因需查询 DMA_CSRy.STATUS 状态位，如下表所示：

表 8-3 DMA 中断标志状态位

DMA_CSRy.STATUS 状态位	DMA 通道 y 当前状态
000	初始状态
001	传输错误，传输地址超出寻址范围
010	传输错误，传输停止请求引起中止
011	传输错误，访问传输来源地址出错
100	传输错误，访问传输目的地址出错
101	传输完成

当传输错误中断标志位或传输完成中断标志位被置位后，如果允许对应的中断请求，CPU 会执行中断服务程序，退出中断服务程序之前应该清除对应的标志位，以避免重复进入中断服务程序。

DMA 中断触发条件、中断使能及中断标志位清除的方法如下表所示：

表 8-4 DMA 中断标志及清除方法

DMA 中断类型	触发条件	中断使能方式	中断标志位清除方式
传输错误	传输地址超出寻址范围； 或，外设请求中止 DMA 传输； 或，访问传输来源 / 目的地址出错	设置 DMA_CSRy.TEIE 为 1	设置 DMA_ICR.TEy 为 0
传输完成	数据传输全部正确完成	设置 DMA_CSRy.TCIE 为 1	设置 DMA_ICR.TCy 为 0

8.7 寄存器列表

DMA 基地址：DMA_BASE = 0x4002 0000

表 8-5 DMA 寄存器列表

寄存器名称	寄存器地址	寄存器描述
DMA_ISR	DMA_BASE + 0x00	中断标志寄存器
DMA_ICR	DMA_BASE + 0x04	中断标志清除寄存器
DMA_CSRy	DMA_BASE + 0x00 + 32×y	通道 y 控制及状态寄存器，y = 1~2
DMA_CNTy	DMA_BASE + 0x04 + 32×y	通道 y 传输数量寄存器，y = 1~2
DMA_SRCADDRy	DMA_BASE + 0x08 + 32×y	通道 y 传输源地址寄存器，y = 1~2
DMA_DSTADDRy	DMA_BASE + 0x0C + 32×y	通道 y 传输目的地址寄存器，y = 1~2
DMA_TRIGy	DMA_BASE + 0x10 + 32×y	通道 y 触发源控制寄存器，y = 1~2

8.8 寄存器描述

有关寄存器描述里所使用的缩写，请参见 1 文档约定章节。

8.8.1 DMA_ISR 中断标志寄存器

Address offset: 0x00 Reset value: 0x0000 0000

位域	名称	权限	功能描述
31:6	RFU	-	保留位，请保持默认值
5	TE2	RO	通道 2 传输错误标志 参见 TE1 说明
4	TC2	RO	通道 2 传输完成标志 参见 TC1 说明
3:2	RFU	-	保留位，请保持默认值
1	TE1	RO	通道 1 传输错误标志 0：传输正常 1：传输出错
0	TC1	RO	通道 1 传输完成标志 0：传输未完成 1：传输完成

8.8.2 DMA_ICR 中断标志清除寄存器

Address offset: 0x04 Reset value: 0xFFFF FFFF

位域	名称	权限	功能描述
31:6	RFU	-	保留位，请保持默认值
5	TE2	R1W0	通道 2 传输错误标志清除 参见 TE1 说明
4	TC2	R1W0	通道 2 传输完成标志清除 参见 TC1 说明
3:2	RFU	-	保留位，请保持默认值
1	TE1	R1W0	通道 1 传输错误标志清除 W0：清除通道 1 传输错误标志 W1：无功能
0	TC1	R1W0	通道 1 传输完成标志清除 W0：清除通道 1 传输完成标志 W1：无功能

8.8.3 DMA_CSRy 通道 y 控制及状态寄存器

Address offset: 0x00 + 32×y　　Reset value: 0x0000 0000（y = 1~2）

位域	名称	权限	功能描述
31:11	RFU	-	保留位，请保持默认值
10:8	STATUS	RW	通道 y 当前状态 000：初始状态 001：传输错误，传输地址超出寻址范围 010：传输错误，传输停止请求引起中止 011：传输错误，访问传输来源地址出错 100：传输错误，访问传输目的地址出错 101：传输完成
7:6	SIZE	RW	通道 y 传输数据位宽配置 00：8 比特 01：16 比特 10：32 比特
5	DSTINC	RW	通道 y 目的地址增量方式配置 0：目的地址固定 1：目的地址自增
4	SRCINC	RW	通道 y 源地址增量方式配置 0：源地址固定 1：源地址自增
3	TRANS	RW	通道 y 传输模式配置 0：批量传输（BULK） 1：块传输（BLOCK）
2	TEIE	RW	通道 y 传输错误中断使能控制 0：禁止 1：使能
1	TCIE	RW	通道 y 传输完成中断使能控制 0：禁止 1：使能
0	EN	RW	通道 y 使能控制 0：禁止 1：使能

8.8.4 DMA_TRIGy 通道 y 触发源控制寄存器

Address offset: 0x10 + 32×y　　Reset value: 0x0000 0000（y = 1~2）

位域	名称	权限	功能描述
31:8	RFU	-	保留位，请保持默认值
7:2	HARDSRC	RW	通道 y 硬件触发信号触发来源配置 000000：UART1 接收 Buf 非空标志 000001：UART1 发送 Buf 空标志 000010：UART2 接收 Buf 非空标志 000011：UART2 发送 Buf 空标志 000100：UART3 接收 Buf 非空标志 000101：UART3 发送 Buf 空标志 000110：SPI1 接收 Buf 非空标志 000111：SPI1 发送 Buf 空标志 001000：SPI2 接收 Buf 非空标志 001001：SPI2 发送 Buf 空标志 001010：ADC 转换完成标志 001011：BTIM1 溢出中断标志 001100：BTIM1 触发中断标志 001101：BTIM2 溢出中断标志 001110：BTIM2 触发中断标志 001111：BTIM3 溢出中断标志 010000：BTIM3 触发中断标志 010011：GTIM1 溢出中断标志 010100：GTIM1 触发中断标志 010101：GTIM1 通道 1 比较捕获中断标志 010110：GTIM1 通道 2 比较捕获中断标志 010111：GTIM1 通道 3 比较捕获中断标志 011000：GTIM1 通道 4 比较捕获中断标志 011001：GTIM2 溢出中断标志 011010：GTIM2 触发中断标志 011011：GTIM2 通道 1 比较捕获中断标志 011100：GTIM2 通道 2 比较捕获中断标志 011101：GTIM2 通道 3 比较捕获中断标志 011110：GTIM2 通道 4 比较捕获中断标志 011111：GTIM3 溢出中断标志 100000：GTIM3 触发中断标志 100001：GTIM3 通道 1 比较捕获中断标志 100010：GTIM3 通道 2 比较捕获中断标志 100011：GTIM3 通道 3 比较捕获中断标志 100100：GTIM3 通道 4 比较捕获中断标志 100101：GTIM4 溢出中断标志 100110：GTIM4 触发中断标志 100111：GTIM4 通道 1 比较捕获中断标志 101000：GTIM4 通道 2 比较捕获中断标志 101001：GTIM4 通道 3 比较捕获中断标志 101010：GTIM4 通道 4 比较捕获中断标志

位域	名称	权限	功能描述
1	SOFTSRC	RW	通道 y 软件触发控制 R0：传输已完成 R1：传输正在进行中 W0：无功能 W1：立即触发该通道启动传输
0	TYPE	RW	通道 y 触发类型配置 0：软件触发模式，由 SOFTSRC 信号启动传输 1：硬件触发模式，由 HARDSRC 所配置的触发源的触发信号启动传输

8.8.5 DMA_CNTy 通道 y 传输数量寄存器

Address offset: 0x04 + 32×y Reset value: 0x0000 0000（y = 1~2）

位域	名称	权限	功能描述
31:20	RFU	-	保留位，请保持默认值
19:16	REPEAT	RW	请写入 1
15:0	CNT	RW	配置通道 y 待传输的数据量为 CNT 传输过程中，每传输完成一个数据，该寄存器值自减 1

8.8.6 DMA_SRCADDRy 通道 y 传输源地址寄存器

Address offset: 0x08 + 32×y Reset value: 0x0000 0000（y = 1~2）

位域	名称	权限	功能描述
31:0	SRCADDR	RW	通道 y 传输源地址

注：

源地址不可以是 DMA 寄存器、FLASH 寄存器、RAM 寄存器。

8.8.7 DMA_DSTADDRy 通道 y 传输目的地址寄存器

Address offset: 0x0C + 32×y Reset value: 0x0000 0000（y = 1~2）

位域	名称	权限	功能描述
31:0	DSTADDR	RW	通道 y 传输目的地址

注：

目的地址不可以是 DMA 寄存器、FLASH 寄存器、RAM 寄存器。

9 通用输入输出端口（GPIO）

9.1 概述

GPIO 控制器实现芯片内部各类数字和模拟电路与物理引脚之间的联系。

GPIO 可配置为数字输入输出和模拟功能，支持外设功能复用，支持高电平、低电平、上升沿和下降沿 4 种中断源，可在深度休眠模式下通过外部中断唤醒 MCU 回到运行模式。

9.2 主要特性

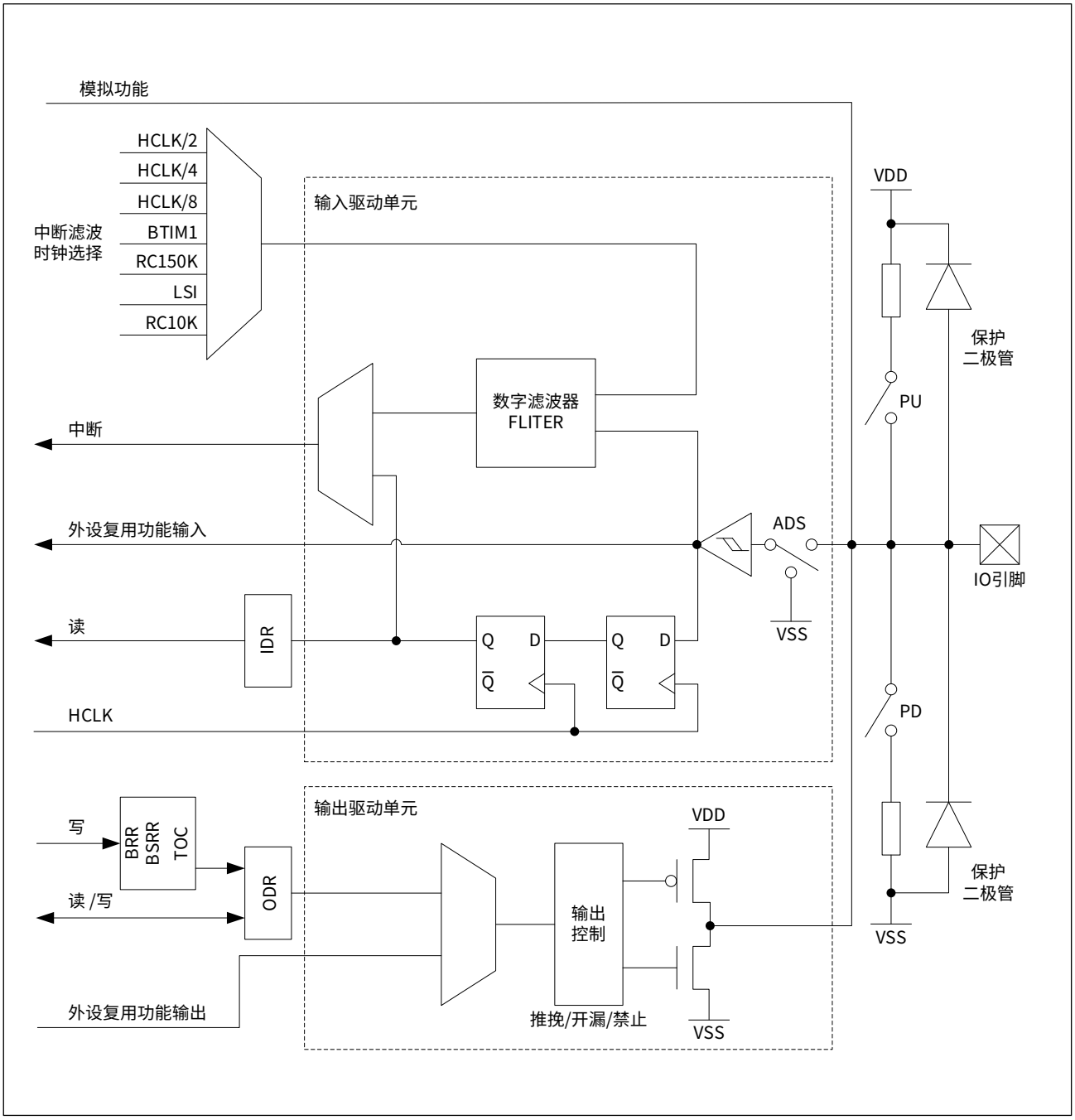
- 所有寄存器通过 AHB 总线接口读写
- 具有数字输入输出和模拟功能
- 数字输入输出支持普通 GPIO 和功能复用
- 模拟功能可作为 ADC、VC、LVD 的输入信号
- 支持内部多种时钟信号输出
- 数字输入支持内部上拉、下拉和高阻三种模式
- 数字输出支持推挽和开漏模式
- 数字输出两档驱动能力选择
- 数字输出两档响应速度选择
- 数字输出支持位置位，位清零，位翻转的原子位操作
- 中断功能支持高电平、低电平、上升沿、下降沿触发方式
- 中断具有数字滤波功能，可选择 7 种时钟源
- 支持在深度休眠模式下通过外部中断唤醒 MCU

9.3 功能描述

9.3.1 功能框图

GPIO 控制器的功能框图如下图所示：

图 9-1 GPIO 功能框图



9.3.2 数字输出

将模拟数字配置寄存器 GPIOx_ANALOG[y]（x 是 GPIO 端口号，x=A、B、C、F；y 是引脚号，y=0~15；下同）清零，配置相应的 GPIO 端口为数字功能；将输入输出方向寄存器 GPIOx_DIR[y] 清零，配置 GPIO 端口为输出模式。数字输出信号来源可以是：

- 输出数据寄存器 GPIOx_ODR
- 片内数字外设

在该模式下，可以继续配置输出模式、输出驱动能力、输出响应速度。

通过输出模式寄存器 GPIOx_OPENDRAIN 配置输出模式，可选择推挽输出或开漏输出。

对于需要较大电流输出能力的场合，可以通过输出驱动能力配置寄存器 GPIOx_DRIVER 来决定是否开启辅助驱动电路以适应更大电流输出。具体电流驱动能力数据请参阅数据手册相关章节。

对于需要较快响应速度的场合，可以通过输出速度寄存器 GPIOx_SPEED 来决定是否开启快速响应电路以适应更快的端口输出速度要求。

9.3.3 数字输入

将模拟数字配置寄存器 GPIOx_ANALOG[y] 清零，配置 GPIO 端口为数字功能；将输入输出方向寄存器 GPIOx_DIR[y] 置位，配置相应的 GPIO 端口为输入模式。数字输入信号可配置：

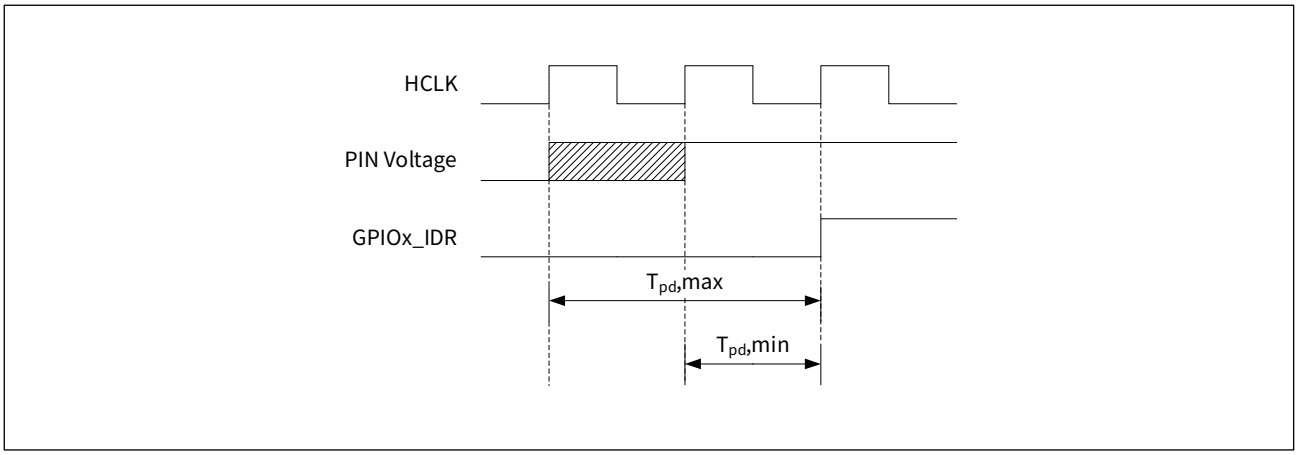
- 到达输入数据寄存器 GPIOx_IDR
- 到达片内数字外设
- 触发中断

在该模式下，数字输入信号通过 ADS 开关导入内部数字输入电路。

经施密特触发器确认电平状态后，可以直接被送往片内复用功能所指向的数字外设的输入，或者通过一个基于 HCLK 的同步器后，在输入数据寄存器 GPIOx_IDR[y] 上呈现。

GPIOx_IDR 寄存器的各位与其前面的锁存器组成了一个同步器，可以避免系统时钟变化的时间内引脚电平跳化而造成的信号不稳定，但是会产生一定的读取延迟。读端口引脚的同步时序如下图所示：

图 9-2 读端口引脚同步时序



在系统时钟上升沿之后的时钟周期，引脚电平信号会锁存在内部寄存器，如图中阴影部分所示，在下一次系统时钟上升沿之后，稳定的引脚电平信号被读取，再一个系统时钟上升沿时，数据被锁存到 GPIOx_IDR 寄存器中。信号延迟 T_{pd} 为 1~2 个系统时钟。

如果考虑将该输入信号用于触发中断，还可以启用内置的硬件滤波器电路。该滤波器电路是基于双 D 触发器同步器实现的，该同步器的时钟来源有 7 种，其中部分时钟源是低功耗模式特有的。例如，可以轻易的实现无软件干预的按键消抖操作。具体的时钟源选项及边沿 / 电平触发选项请参见 9.3.6 中断功能。

在该模式下，通过上拉电阻寄存器 GPIOx_PUR 和下拉电阻寄存器 GPIOx_PDR 可以单独选择打开或者关闭内部上拉和下拉功能。

9.3.4 模拟功能

对于配置有模拟功能的 GPIO 端口，可通过设置模拟数字配置寄存器 GPIOx_ANALOG[y] 为 1，打开 GPIO 模拟信号通道。在打开 GPIO 模拟信号通道时，端口的数字功能关闭，内部上拉、下拉均被断开，内部数字输入信号通过 ADS 开关被短接到 VSS，内部数字输出功能被禁止。

9.3.5 复用功能

通过复用功能寄存器 (GPIOx_AFRH 和 GPIOx_AFRL)，可以实现输入输出端口的复用功能。复用功能寄存器中每 4bit 的位域对应一个 GPIO 端口的复用功能选择，逻辑上可以选择多达 16 个输入输出信号目标。GPIO 复用功能具体定义如下表所示：

表 9-1 GPIO 复用功能设置

GPIOx_AFRL[4×y]	复用功能	GPIOx_AFRH[4×(z-8)]	复用功能
0000	GPIO	0000	GPIO
0001	AF1	0001	AF1
0010	AF2	0010	AF2
0011	AF3	0011	AF3
0100	AF4	0100	AF4
0101	AF5	0101	AF5
0110	AF6	0110	AF6
0111	AF7	0111	AF7

注：

y、z 为引脚号：y=0 ～ 7，对应 GPIOx_AFRL；z=8 ～ 15，对应 GPIOx_AFRH。

每一个 AF 对应的具体外设功能如下表所示：

表 9-2 GPIO 复用功能分配表

引脚名称	复用功能							
	AF0	AF1	AF2	AF3	AF4	AF5	AF6	AF7
PA00	GPIO	UART3_CTS	UART2_CTS	RTC_TAMP	VC1_OUT	SPI2_MISO	GTIM2_CH1	GTIM2_ETR
PA01	GPIO	UART3_RTS	UART2_RTS	I2C2_SCL	LVD_OUT	SPI2_MOSI	GTIM2_CH2	RTC_TAMP
PA02	GPIO	UART3_TXD	UART2_TXD	I2C2_SDA	VC2_OUT	SPI2_SCK	GTIM2_CH3	AWT_ETR
PA03	GPIO	UART3_RXD	UART2_RXD	GTIM2_CH2	PCLK_OUT	SPI2_CS	GTIM2_CH4	
PA04	GPIO		UART2_CTS	I2C2_SCL	HCLK_OUT	SPI1_CS	GTIM2_ETR	
PA05	GPIO	GTIM2_ETR	UART2_RTS	I2C2_SDA	BTIM2_TOGP	SPI1_SCK	GTIM2_CH1	
PA06	GPIO	GTIM3_CH1	UART2_TXD	VC1_OUT	BTIM2_TOGN	SPI1_MISO	GTIM1_CH1	
PA07	GPIO	GTIM4_CH1	UART2_RXD	VC2_OUT	BTIM1_TOGP	SPI1_MOSI	GTIM1_CH2	
PA08	GPIO		UART1_TXD	BTIM2_TOGN	MCO_OUT	LVD_OUT	GTIM3_ETR	
PA09	GPIO	UART3_TXD	UART1_RXD	I2C1_SCL	BTIM1_TOGP	SPI1_CS	GTIM3_CH1	
PA10	GPIO	UART3_RXD	UART1_CTS	I2C1_SDA	BTIM1_TOGN	SPI1_SCK	GTIM3_CH2	
PA11	GPIO	UART3_CTS	UART1_RTS	I2C2_SCL	VC1_OUT	SPI1_MISO	GTIM3_CH3	
PA12	GPIO	UART3_RTS	BTIM_ETR	I2C2_SDA	VC2_OUT	SPI1_MOSI	GTIM3_CH4	
PA13	GPIO		I2C1_SDA	UART1_RXD	UART2_RXD	I2C2_SCL	IR_OUT	
PA14	GPIO	UART3_TXD	I2C1_SCL	UART1_TXD	UART2_TXD	I2C2_SDA		

引脚名称	复用功能							
	AF0	AF1	AF2	AF3	AF4	AF5	AF6	AF7
PA15	GPIO	UART3_RXD	GTIM2_CH1	UART1_RXD	UART2_RXD	SPI1_CS	GTIM2_ETR	
PB00	GPIO	UART2_RXD	UART1_CTS	I2C2_SCL	BTIM1_TOGN	HSI_OUT	GTIM1_CH3	
PB01	GPIO	UART2_TXD	UART1_RTS	I2C2_SDA	GTIM4_TOGN	BTIM3_TOGP	GTIM1_CH4	
PB02	GPIO	UART2_CTS	UART1_TXD	HSE_OUT	GTIM4_TOGP	BTIM3_TOGN	GTIM1_ETR	
PB03	GPIO	UART3_RTS	GTIM2_CH2	UART1_CTS	UART2_TXD	SPI1_SCK	GTIM1_ETR	
PB04	GPIO	UART3_CTS	GTIM4_ETR	UART1_RTS	UART2_CTS	SPI1_MISO	GTIM1_CH1	
PB05	GPIO		GTIM3_CH4	AWT_ETR	UART2_RTS	SPI1_MOSI	GTIM1_CH2	
PB06	GPIO	UART3_TXD	GTIM3_CH3	I2C1_SCL	GTIM4_CH4	SPI2_MOSI	GTIM1_TOGN	
PB07	GPIO	UART3_RXD	GTIM3_CH2	I2C1_SDA	GTIM4_CH3	SPI2_MISO	GTIM1_TOGP	
PB08	GPIO	I2C1_SCL	GTIM3_CH1	UART1_TXD	GTIM4_CH2	SPI2_SCK	GTIM1_CH3	
PB09	GPIO	I2C1_SDA	GTIM4_CH1	UART1_RXD	IR_OUT	SPI2_CS	GTIM1_CH4	
PB10	GPIO	UART2_RTS	UART1_RXD	I2C1_SCL	I2C2_SCL	SPI2_SCK	GTIM2_CH3	
PB11	GPIO	LSI_OUT	GTIM4_ETR	I2C1_SDA	I2C2_SDA	BTIM_ETR	GTIM2_CH4	
PB12	GPIO	GTIM2_TOGN	GTIM4_CH4	LSE_OUT	SPI2_CS	SPI1_CS	GTIM1_TOGN	
PB13	GPIO	GTIM2_TOGP	GTIM4_CH3	I2C2_SCL	SPI2_SCK	SPI1_SCK	GTIM1_TOGP	
PB14	GPIO	GTIM2_CH1	GTIM4_CH2	I2C2_SDA	SPI2_MISO	SPI1_MISO	RTC_OUT	
PB15	GPIO	GTIM2_CH2	GTIM4_CH1	BTIM2_TOGP	SPI2_MOSI	SPI1_MOSI	RTC_1Hz	
PC13	GPIO	PLL_OUT	RTC_1Hz	UART1_CTS	RTC_OUT	BTIM_ETR	GTIM3_ETR	RTC_TAMP
PC14	GPIO	AWT_ETR	GTIM4_CH4	UART1_RTS	BTIM2_TOGP	SPI2_MISO	GTIM3_TOGN	GTIM3_CH1
PC15	GPIO	HSE_OUT	GTIM4_CH3	GTIM4_ETR	BTIM2_TOGN	SPI2_MOSI	GTIM3_TOGP	GTIM3_CH2
PF00	GPIO	AWT_ETR	GTIM4_CH2	I2C1_SDA	BTIM1_TOGP	SPI2_SCK	GTIM2_TOGN	GTIM3_CH3
PF01	GPIO	LSE_OUT	GTIM4_CH1	I2C1_SCL	BTIM1_TOGN	SPI2_CS	GTIM2_TOGP	GTIM3_CH4
PF06	GPIO	UART3_CTS	I2C1_SCL	GTIM4_TOGN	UART2_CTS	I2C2_SCL	GTIM3_TOGN	BTIM3_TOGP
PF07	GPIO	UART3_RTS	I2C1_SDA	GTIM4_TOGP	UART2_RTS	I2C2_SDA	GTIM3_TOGP	BTIM3_TOGN

9.3.6 中断功能

每个 GPIO 在设置为数字输入模式时，可作为外部中断信号源，产生中断的信号源可以设置为高电平、低电平、上升沿、下降沿 4 种。中断触发方式可组合使用，但共用同一个中断标志位。

中断触发后，中断标志寄存器 GPIOx_ISR 的对应位会被硬件置位，程序可通过查询 GPIOx_ISR 来确认产生中断的端口。通过中断标志清除寄存器 GPIOx_ICR[y]，可以清除对应的中断标志位。

内部的中断数字滤波器可对引脚上的输入信号进行数字滤波，提供了 7 种滤波时钟选择，如下表所示：

表 9-3 数字滤波时钟选择

GPIOx_FILTER.FLTCLK	数字滤波时钟频率
000	HCLK / 2
001	HCLK / 4
010	HCLK / 8
011	BTIM1 溢出
100	RC150K（约 150kHz）
101	LSI（约 32.8kHz）
110	RC10K (约 10kHz)

由于选择的滤波时钟周期范围宽广，用户可以轻易实现灵活的输入中断防抖功能。输入电平的变化如果未保持超过一个完整的滤波时钟周期，将不会通过硬件滤波器传达到内部中断触发电路。输入电平的变化如果保持超过两个完整的滤波时钟周期，则一定会通过硬件滤波器。

对于边沿触发类型，考虑到对触发沿的时间的敏感性，建议在中断数字滤波器配置寄存器 GPIOx_FILTER[y] 中关闭硬件滤波器功能，因为硬件滤波器在提升信号稳定性的同时，也会插入一定延迟。

当 CW32F020 工作于休眠模式 (Sleep mode) 或深度休眠模式（DeepSleep mode）时，仍可使用 GPIO 的外部中断功能，当产生外部中断后，可将芯片从休眠模式或深度休眠模式唤醒回到运行模式。

注：

同组 GPIOx.PINy 共用一个硬件滤波器时钟源选择寄存器，因此同组 GPIO 只能以相同的滤波时钟来过滤输入信号抖动。

9.3.7 其他功能

原子位操作

GPIO 控制器支持位置位、位清零和位翻转功能。

向 GPIO 位置位清零寄存器 GPIOx_BSRR[y] 或位清零寄存器 GPIOx_BRR[y] 写入 1，将直接改变输出数据寄存器 GPIOx_ODR 的对应位的状态，从而间接影响最终的输出电平，但不会影响该寄存器其它位的状态。

向 GPIO 位翻转寄存器 GPIOx_TOG[y] 写入 1，将使输出端口的电平状态发生翻转。

端口配置锁定

当配置锁定寄存器 GPIOx_LCKR 的对应位被设置为 1 后，配置寄存器的相应比特不可修改，包括如下配置寄存器：GPIOx_ANALOG、GPIOx_DIR、GPIOx_OPENDRAIN、GPIOx_SPEED、GPIOx_PUR、GPIOx_PDR、GPIOx_AFRH、GPIOx_AFRL、GPIOx_DRIVER、GPIOx_RISEIE、GPIOx_FALLIE、GPIOx_HIGHIE、 GPIOx_LOWIE。

用户可在 GPIO 初始化完成后，对重要端口的配置锁定寄存器相应位进行锁定，防止程序跑飞而对端口的异常操作。

例：

向 GPIOA_LCKR 写入 0x5A5A0201，解锁 GPIOA 除 PA09、PA00 之外的端口相关配置寄存器，同时，锁定 PA09 和 PA00 端口相关配置寄存器。

端口复位状态

上电或复位后，SWCLK（PA14）和 SWDIO（PA13）默认为数字上拉，BOOT（PF03）默认为数字功能。其他端口默认为模拟高阻输入（high resistance input），上拉或下拉均默认不打开。

9.4 编程示例

在配置 GPIO 端口时，必须先设置 SYSCTRL_AHBEN.GPIOx 为 1，使能对应的 GPIO 配置时钟及工作时钟，并向 GPIOx_LCKR 锁定寄存器写入‘0x5A5Aiiii’，以解锁 GPIO 相关配置寄存器，配置完成后，如有必要，可设置 GPIOx_LCKR 锁定寄存器，以保护设置内容不会被意外改写。

9.4.1 数字输出编程示例

步骤 1：设置 GPIOx_ANALOG.PINy 为 0，将端口配置为数字功能；

步骤 2：设置 GPIOx_DIR.PINy 为 0，将端口配置成输出；

步骤 3：配置 GPIOx_OPENDRAIN 寄存器，设置端口输出模式；

步骤 4：配置 GPIOx_DRIVER 寄存器，设置端口输出驱动能力；

步骤 5：配置 GPIOx_SPEED 寄存器，设置端口输出速度；

步骤 6：配置 GPIOx_ODR 寄存器，设置端口输出电平。

9.4.2 数字输入编程示例

步骤 1：设置 GPIOx_ANALOG.PINy 为 0，将端口配置为数字功能；

步骤 2：设置 GPIOx_DIR.PINy 为 1，将端口配置成输入；

步骤 3：配置 GPIOx_PUR 寄存器，选择是否使能内部上拉电阻；

步骤 4：配置 GPIOx_PDR 寄存器，选择是否使能内部下拉电阻；

步骤 5：读取 GPIOx_IDR 寄存器，读出端口输入电平。

9.4.3 模拟功能编程示例

步骤 1：设置 GPIOx_ANALOG.PINy 为 1，将端口配置为模拟功能。

9.4.4 复用功能编程示例

步骤 1：根据应用需求将端口配置成数字输出或数字输入，具体寄存器配置步骤请参见 9.4.1 数字输出编程示例、9.4.2 数字输入编程示例；

步骤 2：配置 GPIOx_AFRH 或 GPIOx_AFRL 寄存器，设置端口复用功能，请参见表 9-2 GPIO 复用功能分配表。

9.4.5 中断功能编程示例

步骤 1：将端口配置成数字输入，具体寄存器配置步骤请参见 9.4.2 数字输入编程示例；

步骤 2：配置 GPIOx_FILTER.FLTCLK，选择端口中断滤波时钟；

步骤 3：设置 GPIOx_FILTER.PINy 为 1，使能相应端口滤波时钟；

步骤 4：配置 NVIC 控制器，请参见 5 中断章节；

步骤 5：根据应用需求，配置 GPIOx_RISEIE、GPIOx_FALLIE、GPIOx_HIGHIE、GPIOx_LOWIE 寄存器，选择 GPIO 中断触发方式；

步骤 6：端口中断输入信号触发 GPIO 中断，执行中断服务函数。

9.5 寄存器列表

GPIOA 基地址：GPIOA_BASE = 0x4800 0000

GPIOB 基地址：GPIOB_BASE = 0x4800 0400

GPIOC 基地址：GPIOC_BASE = 0x4800 0800

GPIOF 基地址：GPIOF_BASE = 0x4800 1400

表 9-4 GPIO 寄存器列表

寄存器名称	寄存器地址	寄存器描述
GPIOx_DIR	GPIOx_BASE + 0x00	GPIOx 输入输出方向寄存器
GPIOx_OPENDRAIN	GPIOx_BASE + 0x04	GPIOx 输出模式寄存器
GPIOx_SPEED	GPIOx_BASE + 0x08	GPIOx 输出速度寄存器
GPIOx_PDR	GPIOx_BASE + 0x0C	GPIOx 下拉电阻寄存器
GPIOx_PUR	GPIOx_BASE + 0x10	GPIOx 上拉电阻寄存器
GPIOx_AFRH	GPIOx_BASE + 0x14	GPIOx 复用功能寄存器高段
GPIOx_AFRL	GPIOx_BASE + 0x18	GPIOx 复用功能寄存器低段
GPIOx_ANALOG	GPIOx_BASE + 0x1C	GPIOx 模拟数字配置寄存器
GPIOx_DRIVER	GPIOx_BASE + 0x20	GPIOx 输出驱动能力寄存器
GPIOx_RISEIE	GPIOx_BASE + 0x24	GPIOx 上升沿中断使能寄存器
GPIOx_FALLIE	GPIOx_BASE + 0x28	GPIOx 下降沿中断使能寄存器
GPIOx_HIGHIE	GPIOx_BASE + 0x2C	GPIOx 高电平中断使能寄存器
GPIOx_LOWIE	GPIOx_BASE + 0x30	GPIOx 低电平中断使能寄存器
GPIOx_ISR	GPIOx_BASE + 0x34	GPIOx 中断标志寄存器
GPIOx_ICR	GPIOx_BASE + 0x38	GPIOx 中断标志清除寄存器
GPIOx_LCKR	GPIOx_BASE + 0x3C	GPIOx 配置锁定寄存器
GPIOx_FILTER	GPIOx_BASE + 0x40	GPIOx 中断数字滤波器配置寄存器
GPIOx_IDR	GPIOx_BASE + 0x50	GPIOx 输入数据寄存器
GPIOx_ODR	GPIOx_BASE + 0x54	GPIOx 输出数据寄存器
GPIOx_BRR	GPIOx_BASE + 0x58	GPIOx 位清零寄存器
GPIOx_BSRR	GPIOx_BASE + 0x5C	GPIOx 位置位清零寄存器
GPIOx_TOG	GPIOx_BASE + 0x60	GPIOx 位翻转寄存器

9.6 寄存器描述

有关寄存器描述里所使用的缩写，请参见 1 文档约定章节。

9.6.1 GPIOx_DIR GPIO 输入输出方向寄存器 (x =A, B, C, F)

Address offset: 0x00 Reset value: 0x0000 FFFF(GPIOA)

0x0000 FFFF(GPIOB)

0x0000 E000(GPIOC)

0x0000 00CB(GPIOF)

位域	名称	权限	功能描述
31:16	RFU	-	保留位，请保持默认值
15:0	PINy y=0 ~ 15	RW	端口输入输出方向控制 0：将端口配置成输出 1：将端口配置成输入

9.6.2 GPIOx_OPENDRAIN GPIO 输出模式寄存器 (x =A, B, C, F)

Address offset: 0x04 Reset value: 0x0000 0000

位域	名称	权限	功能描述
31:16	RFU	-	保留位，请保持默认值
15:0	PINy y=0 ~ 15	RW	端口输出方式控制位 0：推挽输出 1：开漏输出

9.6.3 GPIOx_SPEED GPIO 输出速度寄存器 (x =A, B, C, F)

Address offset: 0x08 Reset value: 0x0000 0000

位域	名称	权限	功能描述
31:16	RFU	-	保留位，请保持默认值
15:0	PINy y=0 ~ 15	RW	端口输出速度控制位 0：低输出速度 1：高输出速度

9.6.4 GPIOx_PDR GPIO 下拉电阻寄存器 (x =A, B, C, F)

Address offset: 0x0C Reset value: 0x0000 0000

位域	名称	权限	功能描述
31:16	RFU	-	保留位，请保持默认值
15:0	PINy y=0 ~ 15	RW	端口下拉电阻使能控制 0：禁止下拉电阻 1：使能下拉电阻 注：当 GPIOx_PDR.PINy、GPIOx_PUR.PINy 同时置 1 时，端口状态为使能上拉电阻。

9.6.5 GPIOx_PUR GPIO 上拉电阻寄存器 (x =A, B, C, F)

Address offset: 0x10 Reset value: 0x0000 0000

位域	名称	权限	功能描述
31:16	RFU	-	保留位，请保持默认值
15:0	PINy y=0 ~ 15	RW	端口上拉电阻使能控制 0：禁止上拉电阻 1：使能上拉电阻 注：当 GPIOx_PDR.PINy、GPIOx_PUR.PINy 同时置 1 时，端口状态为使能上拉电阻。

9.6.6 GPIOx_AFRH GPIO 复用功能配置寄存器高段 (x =A, B, C, F)

Address offset: 0x14 Reset value: 0x0000 0000

位域	名称	权限	功能描述
31:28 27:24 23:20 19:16 15:12 11:8 7:4 3:0	AFRy y = 8 ~ 15	RW	端口数字复用功能控制 0000：GPIO 0001：AF1 0010：AF2 0011：AF3 0100：AF4 0101：AF5 0110：AF6 0111：AF7

9.6.7 GPIOx_AFRL GPIO 复用功能配置寄存器低段 (x =A, B, C, F)

Address offset: 0x18 Reset value: 0x0000 0000

位域	名称	权限	功能描述
31:28 27:24 23:20 19:16 15:12 11:8 7:4 3:0	AFRy y = 0 ~ 7	RW	端口数字复用功能控制 0000：GPIO 0001：AF1 0010：AF2 0011：AF3 0100：AF4 0101：AF5 0110：AF6 0111：AF7

9.6.8 GPIOx_ANALOG GPIO 模拟数字配置寄存器 (x =A, B, C, F)

Address offset: 0x1C Reset value: 0x0000 9FFF(GPIOA)

0x0000 FFFF(GPIOB)

0x0000 E000(GPIOC)

0x0000 00C3(GPIOF)

位域	名称	权限	功能描述
31:16	RFU	-	保留位，请保持默认值
15:0	PINy y=0 ~ 15	RW	端口模拟 / 数字功能配置 0：将端口配置为数字功能 1：将端口配置为模拟功能

9.6.9 GPIOx_DRIVER GPIO 输出驱动能力配置寄存器 (x =A, B, C, F)

Address offset: 0x20 Reset value: 0x0000 0000

位域	名称	权限	功能描述
31:16	RFU	-	保留位，请保持默认值
15:0	PINy y=0 ~ 15	RW	端口驱动能力控制 0：高驱动能力 1：低驱动能力

9.6.10 GPIOx_RISEIE GPIO 上升沿中断使能寄存器 (x =A, B, C, F)

Address offset: 0x24 Reset value: 0x0000 0000

位域	名称	权限	功能描述
31:16	RFU	-	保留位，请保持默认值
15:0	PINy y=0 ~ 15	RW	端口上升沿中断使能控制 0：禁止相应端口的上升沿中断 1：使能相应端口的上升沿中断

9.6.11 GPIOx_FALLIE GPIO 下降沿中断使能寄存器 (x =A, B, C, F)

Address offset: 0x28 Reset value: 0x0000 0000

位域	名称	权限	功能描述
31:16	RFU	-	保留位，请保持默认值
15:0	PINy y=0 ~ 15	RW	端口下降沿中断使能控制 0：禁止相应端口的下降沿中断 1：使能相应端口的下降沿中断

9.6.12 GPIOx_HIGHIE GPIO 高电平中断使能寄存器 (x =A, B, C, F)

Address offset: 0x2C Reset value: 0x0000 0000

位域	名称	权限	功能描述
31:16	RFU	-	保留位，请保持默认值
15:0	PINy y=0 ~ 15	RW	端口高电平中断使能控制 0：禁止相应端口的高电平中断 1：使能相应端口的高电平中断

9.6.13 GPIOx_LOWIE GPIO 低电平中断使能寄存器 (x =A, B, C, F)

Address offset: 0x30 Reset value: 0x0000 0000

位域	名称	权限	功能描述
31:16	RFU	-	保留位，请保持默认值
15:0	PINy y=0 ~ 15	RW	端口低电平中断使能控制 0：禁止相应端口的低电平中断 1：使能相应端口的低电平中断

9.6.14 GPIOx_ISR GPIO 中断标志寄存器 (x =A, B, C, F)

Address offset: 0x34 Reset value: 0x0000 0000

位域	名称	权限	功能描述
31:16	RFU	-	保留位，请保持默认值
15:0	PINy y=0 ~ 15	RO	端口中断状态标志 0：未检测到已使能的中断 1：已检测到已使能的中断

9.6.15 GPIOx_ICR GPIO 中断标志清除寄存器 (x =A, B, C, F)

Address offset: 0x38 Reset value: 0x0000 FFFF(GPIOA)

0x0000 FFFF(GPIOB)

0x0000 E000(GPIOC)

0x0000 00CB(GPIOF)

位域	名称	权限	功能描述
31:16	RFU	-	保留位，请保持默认值
15:0	PINy y=0 ~ 15	R1W0	端口中断状态标志清除 W0：清除相应的中断标志位 W1：无功能

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

Address offset: 0x3C Reset value: 0x0000 0000

位域	名称	权限	功能描述
31:16	KEY	WO	仅当 KEY 为 0x5A5A 时，对该寄存器的写操作有效
15:0	PINy y=0 ~ 15	RW	端口配置锁定 0：配置寄存器的相应比特可以修改 1：配置寄存器的相应比特不可修改 注：配置寄存器如下所示： GPIOx_ANALOG、GPIOx_DIR、GPIOx_OPENDRAIN、 GPIOx_SPEED、GPIOx_PDR、GPIOx_PUR、 GPIOx_AFRH、GPIOx_AFRL、GPIOx_DRIVER、 GPIOx_RISEIE、GPIOx_FALLIE、GPIOx_HIGHIE、 GPIOx_LOWIE

9.6.17 GPIOx_FILTER GPIO 中断数字滤波器配置寄存器 (x =A, B, C, F)

Address offset: 0x40 Reset value: 0x0000 0000

位域	名称	权限	功能描述
31:19	RFU	-	保留位，请保持默认值
18:16	FLTCLK	RW	端口中断滤波时钟选择 000：HCLK / 2 001：HCLK / 4 010：HCLK / 8 011：BTIM1 溢出 100：RC150K（约 150kHz） 101：LSI（约 32.8kHz） 110：RC10K（约 10kHz）
15:0	PINy y=0 ~ 15	RW	端口滤波时钟使能控制 0：禁止相应端口滤波时钟 1：使能相应端口滤波时钟 注：滤除宽度小于 FLTCLK 时钟宽度的脉冲

9.6.18 GPIOx_IDR GPIO 输入数据寄存器 (x =A, B, C, F)

Address offset: 0x50 Reset value: 0x0000 6000(GPIOA)

0x0000 0000(GPIOB)

0x0000 0000(GPIOC)

0x0000 0000(GPIOF)

位域	名称	权限	功能描述
31:16	RFU	-	保留位，请保持默认值
15:0	PINy y=0 ~ 15	RO	读出端口输入电平状态 0：端口为低电平 1：端口为高电平

9.6.19 GPIOx_ODR GPIO 输出数据寄存器 (x =A, B, C, F)

Address offset: 0x54 Reset value: 0x---- ----

位域	名称	权限	功能描述
31:16	RFU	-	保留位，请保持默认值
15:0	PINy y=0 ~ 15	RW	设置端口输出电平 0：设置端口输出低电平 1：设置端口输出高电平

9.6.20 GPIOx_BRR GPIO 端口位清零寄存器 (x =A, B, C, F)

Address offset: 0x58 Reset value: 0x0000 0000

位域	名称	权限	功能描述
31:16	RFU	-	保留位，请保持默认值
15:0	BRRy y=0 ~ 15	R0W1	端口位清零控制 0：不影响 GPIOx_ODR 寄存器相应的比特 1：设置 GPIOx_ODR 寄存器相应的比特为 0

9.6.21 GPIOx_BSRR GPIO 端口位置位清零寄存器 (x =A, B, C, F)

Address offset: 0x5C Reset value: 0x0000 0000

位域	名称	权限	功能描述
31:16	BRRy y=0 ~ 15	R0W1	端口位清零控制 0：不影响 GPIOx_ODR 寄存器相应的比特 1：设置 GPIOx_ODR 寄存器相应的比特为 0
15:0	BSSy y=0 ~ 15	R0W1	端口位置位控制 0：不影响 GPIOx_ODR 寄存器相应的比特 1：设置 GPIOx_ODR 寄存器相应的比特为 1 注：当 BRRy 与 BSSy 同时置 1 时，BSSy 具有更高优先级

9.6.22 GPIOx_TOG GPIO 端口位翻转寄存器 (x =A, B, C, F)

Address offset: 0x60 Reset value: 0x0000 0000

位域	名称	权限	功能描述
31:16	RFU	-	保留位，请保持默认值
15:0	PINy y=0 ~ 15	R0W1	端口位翻转控制 0：不影响 GPIOx_ODR 寄存器相应的比特 1：对 GPIOx_ODR 寄存器相应的比特取反

10 循环冗余校验（CRC）

10.1 概述

循环冗余校验 (CRC) 主要应用于核实数据传输或数据存储的正确性和完整性。CW32F020 内部集成 CRC 计算单元，支持采用多种 CRC 算法对输入数据进行 CRC 计算。

10.2 主要特性

- 3 种输入数据位宽：8bit、16bit、32bit
- 2 种多项式
  CRC-16 多项式 1：$x^{16} + x^{15} + x^2 + 1$
  CRC-16 多项式 2：$x^{16} + x^{12} + x^5 + 1$
- 8 种常用的算法
  基于多项式，初始值，结果异或值，输入 / 输出反转的组合

10.3 功能描述

CRC 单元通过对输入数据 (或输入数据的反转) 和选定的多项式值进行‘除’运算，得到的余数再进行反转或者不反转，以及异或处理，得到 CRC 计算结果。

CRC 单元在使用之前，需要设置 SYSCTRL_AHBEN.CRC 为 1，打开 CRC 单元的配置时钟及工作时钟，一般在系统初始化时进行设置。

10.3.1 算法模式

CW32F020 的 CRC 单元支持多种算法模式。不同的 CRC 算法，对应的多项式、初始值、输入数据反转、输出数据反转、结果异或值等参数不同，如下表所示：

表 10-1 CRC 算法模式

算法名称	多项式值	初始值	输入反转	输出反转	结果异或值
CRC16_IBM	0x8005	0x0000	True	True	0x0000
CRC16_MAXIM	0x8005	0x0000	True	True	0xFFFF
CRC16_USB	0x8005	0xFFFF	True	True	0xFFFF
CRC16_MODBUS	0x8005	0xFFFF	True	True	0x0000
CRC16_CCITT	0x1021	0x0000	True	True	0x0000
CRC16_CCITT_False	0x1021	0xFFFF	False	False	0x0000
CRC16_X25	0x1021	0xFFFF	True	True	0xFFFF
CRC16_XMODEM	0x1021	0x0000	False	False	0x0000

各参数含义如下：

- 多项式值

  多项式是码组的描述，如 CRC-16 多项式 2：$x^{16} + x^{12} + x^5 + 1$，对应的码组是 1 0001 0000 0010 0001。因为多项式码组的最高位固定为 1，且最高位的位置已知，因此一般将最高位 1 去掉后的码组称为多项式值，如 CRC-16 多项式 2 的值为 0001 0000 0010 0001，即 0x1021。

- 初始值

  在计算 CRC 校验值之前，CRC 寄存器的初始值。

- 输入数据反转

  即在计算开始前，将需要计算 CRC 校验值的数据进行高低序位反转，如数据位 1011，反转后为 1101。

- 输出数据反转

  在 CRC 计算结束后，与结果异或值进行异或之前，将计算值进行高低序位反转，如计算结果为 1011，反转后为 1101。

- 结果异或值

  在 CRC 计算结束后，得到的 CRC 计算值与结果异或值进行异或操作，就得到了最终的 CRC 校验值。

10.3.2 输入数据位宽

CRC 计算单元支持三种输入数据位宽：8bit、16bit、32bit。输入数据位宽和算法没有对应关系，但和写入的寄存器位宽要保持一致。为保证同样一组数据采用不同的输入数据位宽计算得到的 CRC 校验码相同，需要遵循如下原则：

- 每次输入的数据类型应与寄存器位宽一致
- 先输入低字节再输入高字节

例如，用户需要计算 0x00, 0x11, 0x22 , 0x33, 0x44, 0x55, 0x66, 0x77 这一组数据的 CRC 校验值，对应不同输入数据位宽示例如下：

1. 8bit 输入数据位宽时，写入顺序：0x00, 0x11, 0x22 , 0x33, 0x44, 0x55, 0x66, 0x77

   代码示例：

```
CW_CRC -> DR8 = 0x00 ;
CW_CRC -> DR8 = 0x11 ;
CW_CRC -> DR8 = 0x22 ;
CW_CRC -> DR8 = 0x33 ;
CW_CRC -> DR8 = 0x44 ;
CW_CRC -> DR8 = 0x55 ;
CW_CRC -> DR8 = 0x66 ;
CW_CRC -> DR8 = 0x77 ;
```

2. 16bit 输入数据位宽时，写入顺序：0x1100, 0x3322, 0x5544, 0x7766

   代码示例：

```
CW_CRC -> DR16 = 0x1100 ;
CW_CRC -> DR16 = 0x3322 ;
CW_CRC -> DR16 = 0x5544 ;
CW_CRC -> DR16 = 0x7766 ;
```

3. 32bit 输入数据位宽时，写入顺序： 0x3322 1100, 0x7766 5544

   代码示例：

```
CW_CRC -> DR32 = 0x3322 1100 ;
CW_CRC -> DR32 = 0x7755 6644 ;
```

10.4 编程示例

10.4.1 CRC16_CCITT 算法模式

步骤 1：设置 CRC_CR.MODE 为 0x04，选择 CRC16_CCITT 算法模式，硬件自动配置 CRC 寄存器初始值为 0x0000；

步骤 2：将待编码的原始数据依次写入数据寄存器 CRC_DR ，输入数据位宽可选择 8bit、16bit、32bit 。请参见 10.3.2 输入数据位宽中的代码示例；

步骤 3：读取 CRC_RESULT[15:0] 获取 CRC 校验值。

代码示例：

```
tempdata = CW_CRC -> RESULT ;
```

10.5 寄存器列表

CRC 基地址：CRC_BASE =0x4002 3000

表 10-2 CRC 寄存器列表

寄存器名称	寄存器地址	寄存器描述
CRC_CR	CRC_BASE + 0x00	控制寄存器
CRC_DR	CRC_BASE + 0x08	数据寄存器
CRC_RESULT	CRC_BASE + 0x0C	结果寄存器

10.6 寄存器描述

有关寄存器描述里所使用的缩写，请参见 1 文档约定章节。

10.6.1 CRC_CR 控制寄存器

Address offset: 0x00 Reset value: 0x0000 0004

位域	名称	权限	功能描述
31:4	RFU	-	保留位，请保持默认值
3:0	MODE	RW	CRC 算法模式配置 0000：CRC16_IBM 0001：CRC16_MAXIM 0010：CRC16_USB 0011：CRC16_MODBUS 0100：CRC16_CCITT 0101：CRC16_CCITT_False 0110：CRC16_X25 0111：CRC16_XMODEM

10.6.2 CRC_DR 数据寄存器

Address offset: 0x08 Reset value: 0x0000 0000

位域	名称	权限	功能描述
31:0	DR32	RW	数据写入寄存器，位宽 32bit 时使用
15:0	DR16	RW	数据写入寄存器，位宽 16bit 时使用
7:0	DR8	RW	数据写入寄存器，位宽 8bit 时使用

10.6.3 CRC_RESULT 结果寄存器

Address offset: 0x0C Reset value: 0x0000 FFFF

位域	名称	权限	功能描述
31:16	RFU	-	保留位，请保持默认值
15:0	RESULT16	RO	CRC16 计算结果

11 自动唤醒定时器（AWT）

11.1 概述

CW32F020 内部集成 1 个自动唤醒定时器 (AWT)，AWT 包含一个 16bit 向下计数器，并由一个可编程预分频器驱动。AWT 可选 5 种计数时钟源，可工作于定时模式或计数模式。当计数器时钟源为 LSE 或 LSI 时，AWT 可在深度休眠模式下保持运行，下溢出中断可唤醒 MCU 回到运行模式。

11.2 主要特性

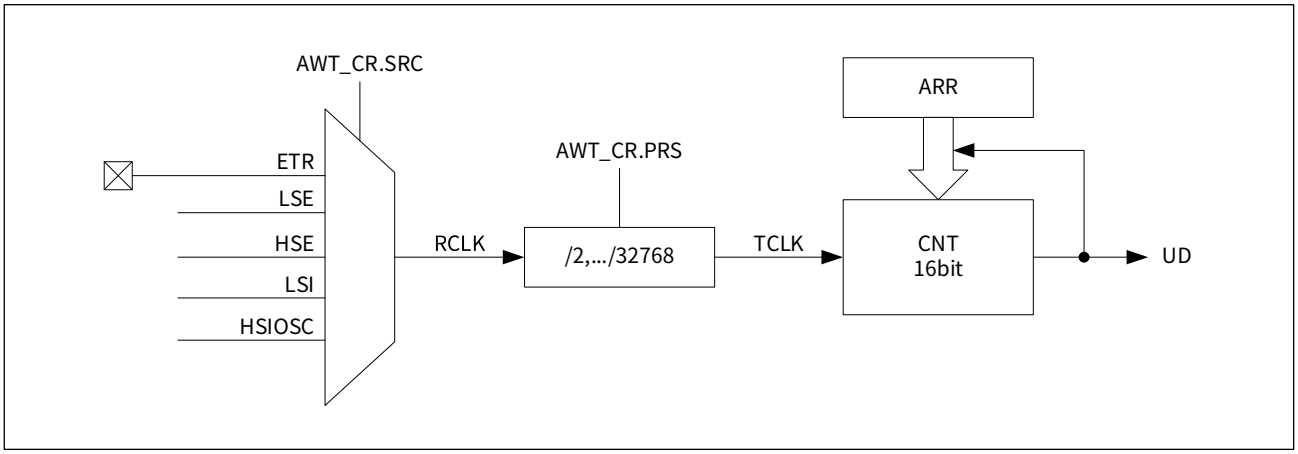
- 16bit 向下计数器
- 5 种工作时钟源：LSE、HSE、LSI、HSIOSC、ETR
- 可编程预分频器：2 ~ 32768 分频
- 支持深度休眠模式下保持运行，中断唤醒 MCU
- 计数时钟源为 LSI 时，唤醒周期为 60μs ~ 65536s

11.3 功能描述

11.3.1 功能框图

AWT 功能框图如下图所示：

图 11-1 AWT 功能框图



时钟源

AWT 的计数器时钟可选 5 种时钟源 RCLK：LSE、HSE、LSI、HSIOSC 以及外部 AWT_ETR 引脚输入的 ETR 信号，通过控制寄存器 AWT_CR 的 SRC 位域来选择。时钟源经预分频器分频后作为 AWT 的计数器时钟。

注：

时钟源选择必须在启动定时器之前完成，在定时器运行过程中禁止修改，否则会导致 AWT 工作异常。

预分频器

AWT 内置一个 4bit 预分频器，计数器时钟源 RCLK 经预分频器分频后驱动计数器计数。通过控制寄存器 AWT_CR 的 PRS 位域选择具体的分频系数，设置值和分频系数对应关系如下表所示：

表 11-1 AWT 计数时钟预分频系数

PRS	分频比	PRS	分频比	PRS	分频比	PRS	分频比
0x00	保留， 不可配置	0x04	16	0x08	256	0x0C	4096
0x01	2	0x05	32	0x09	512	0x0D	8192
0x02	4	0x06	64	0x0A	1024	0x0E	16384
0x03	8	0x07	128	0x0B	2048	0x0F	32768

注：

预分频系数的设置必须在启动定时器之前完成，在定时器运行过程中禁止修改，否则会导致 AWT 工作异常。

计数单元

计数单元的核心组件是一个 16bit 向下计数器 CNT 和一个 16bit 自动重载寄存器 ARR。

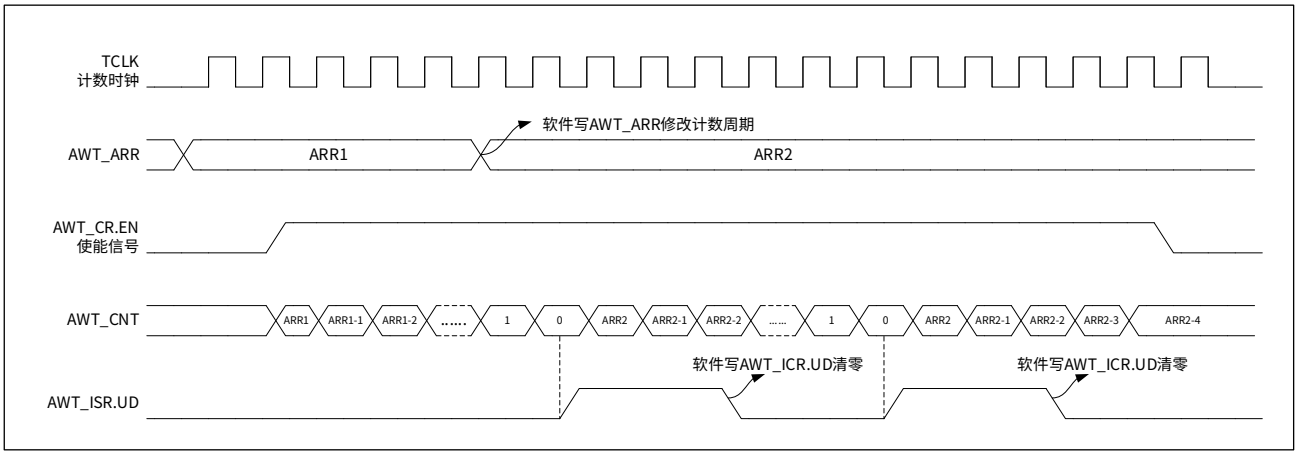
重载寄存器 ARR 用于设置定时或计数周期，定时或计数周期数为 ARR+1。计数器 CNT 用于递减计数，递减到 0 后下溢出，并从重载值 ARR 开始重新计数。

设置 AWT_CR.EN 为 1 使能 AWT，计数器 CNT 在计数时钟 TCLK 的驱动下从重载值 ARR 开始递减计数，当递减到 0 后产生下溢出，下溢出中断标志位 AWT_ISR.UD 将被硬件置位，计数器 CNT 重装载 AWT_ARR 寄存器的值，重新开始下一个周期的递减计数。

用户可随时设置重载值寄存器 AWT_ARR 的值，但不影响 CNT 当前计数值，新设置的值将在下个定时或者计数周期开始起作用。

AWT 计数时序图如下图所示：

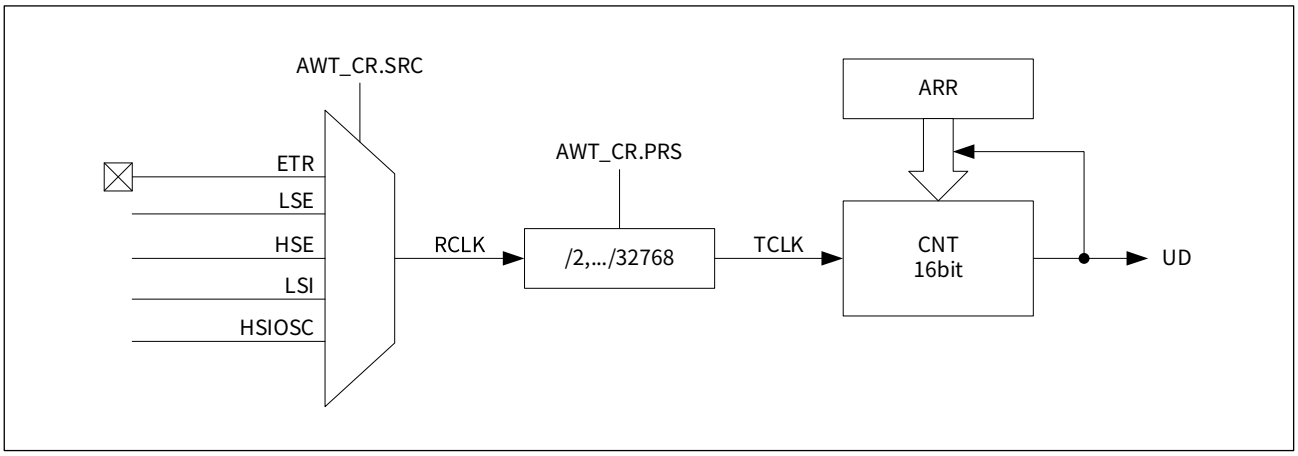
图 11-2 AWT 计数及中断标志



11.3.2 定时功能

AWT 定时功能一般用于延时或者产生固定间隔时间的时基信号。HSE、HSIOSC、LSE、LSI、ETR 都可以作为其时钟源，当选择 ETR 作为其时钟源时，需要确保 ETR 输入的信号为固定周期信号。AWT 定时模式的功能框图如下图所示：

图 11-3 定时功能框图



定时时间 T 计算公式：

$$T = (2^{PRS} / RCLK) \times (ARR + 1)$$

其中，RCLK 为计数器时钟源，PRS 为预分频系数，ARR 为重载值。

例：

当计数器时钟源 RCLK 为 LSE（时钟频率为 32768Hz）时，要求定时 200ms。

如果设置预分频系数 PRS 为 0x01，计算

$$T = 200\ ms = (2^1 / 32768) \times (ARR + 1)$$

则

ARR = 3275.8，最接近的整数是：3276 (0xCCC)

即需要设置重载值 ARR 为 0xCCC。

设置定时器使能控制位 AWT_CR.EN 为 1，重载值 ARR 被装载到计数器 CNT 中。此后每来 1 个 TCLK 时钟，CNT 计数值减 1。当计数值递减到 0 后产生下溢出，下溢出中断标志位 AWT_ISR.UD 被硬件置位。如果允许中断（设置 AWT_IER.UD 为 1），CPU 会响应中断服务程序，退出中断服务程序之前应设置 AWT_ICR.UD 为 0 来清除该标志位，以避免重复进入中断服务程序。

定时功能配置流程如下（当选择时钟源为 LSE/HSE/LSI/HSIOSC 时，步骤 1 和步骤 2 不需要执行）：

步骤 1：设置外设时钟使能控制寄存器 SYSCTRL_AHBEN 的相关位为 1，使能 AWT_ETR 对应 GPIO 端口的配置时钟及工作时钟；

步骤 2：设置 GPIO 复用功能寄存器 GPIOx_AFRH 和 GPIOx_AFRL 的相关位，配置对应引脚为 AWT 定时器的 AWT_ETR 功能；

步骤 3：设置外设时钟使能控制寄存器 SYSCTRL_APBEN2.AWT 为 1，打开 AWT 模块的配置时钟；

步骤 4：配置控制寄存器 AWT_CR.SRC，选择 AWT 计数时钟源；

步骤 5：配置控制寄存器 AWT_CR.PRS，选择 AWT 计数时钟分频系数；

步骤 6：设置中断使能寄存器 AWT_IER.UD 为 1，使能计数器 CNT 下溢出事件引发中断请求；

步骤 7：配置重载值寄存器 AWT_ARR，设置定时周期，具体配置请参考本节相关描述；

步骤 8：设置控制寄存器 AWT_CR.EN 为 1, 启动定时器开始计时；

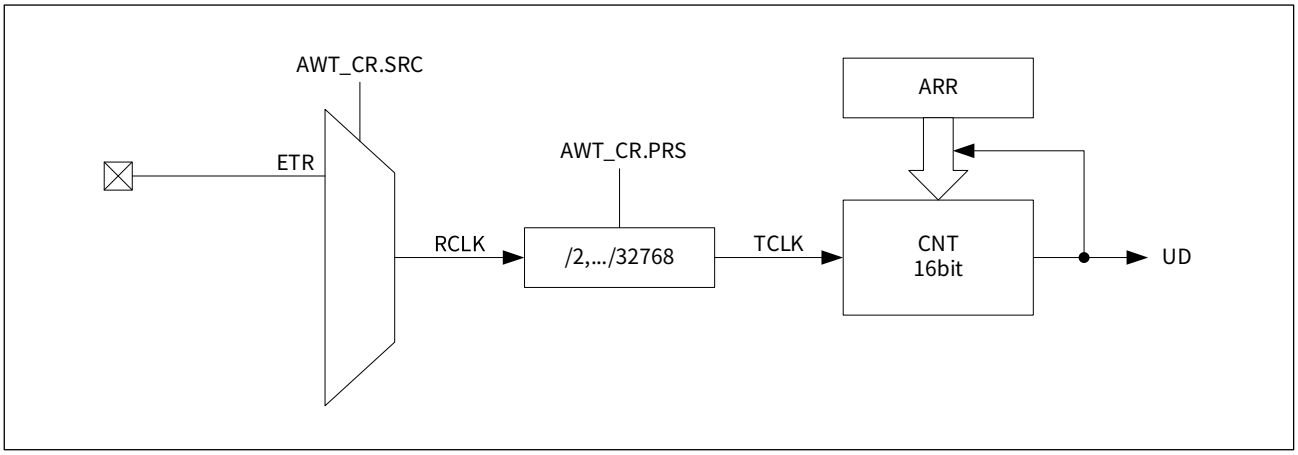
步骤 9：计数器 CNT 下溢出，进入中断服务函数，设置 AWT_ICR.UD 为 0 清除中断标志；

步骤 10：如果不希望继续定时，则设置控制寄存器 AWT_CR.EN 为 0 关闭定时器定时功能。

11.3.3 计数功能

AWT 计数功能用于测定某个事件发生的次数。计数模式选择 ETR 作为计数时钟源，用户将需要计数的外部信号通过 AWT_ETR 引脚（具体引脚请参阅数据手册引脚定义）输入。AWT 计数模式的功能框图如下图所示：

图 11-4 计数功能框图



通过 AWT_CR.SRC 选择计数时钟源，通过 AWT_CR.PRS 选择预分频系数，通过 AWT_ARR.ARR 配置计数周期，计数周期为 AWT_ARR+1。

设置定时器使能控制位 AWT_CR.EN 为 1，重载值 ARR 被装载到计数器 CNT 中。此后每来 1 个 TCLK 时钟，CNT 计数值减 1。当计数值递减到 0 后产生下溢出，下溢出中断标志位 AWT_ISR.UD 被硬件置位。如果允许中断（设置 AWT_IER.UD 为 1），CPU 会响应中断服务程序，退出中断服务程序之前应设置 AWT_ICR.UD 为 0 来清除该标志位，以避免重复进入中断服务程序。

计数功能配置流程如下：

步骤 1：设置外设时钟使能控制寄存器 SYSCTRL_AHBEN 的相关位为 1，使能 AWT_ETR 对应 GPIO 端口的配置时钟及工作时钟；

步骤 2：设置 GPIO 复用功能寄存器 GPIOx_AFRH 和 GPIOx_AFRL 的相关位，配置对应引脚为 AWT 定时器的 AWT_ETR 功能；

步骤 3：设置外设时钟使能控制寄存器 SYSCTRL_APBEN2.AWT 为 1，打开 AWT 模块的配置时钟；

步骤 4：设置控制寄存器 AWT_CR.SRC 为 0x04，选择 AWT 计数时钟源为 ETR；

步骤 5：配置控制寄存器 AWT_CR.PRS，选择 AWT 计数时钟分频系数；

步骤 6：设置中断使能寄存器 AWT_IER.UD 为 1，使能计数器 CNT 下溢出事件引发中断请求；

步骤 7：根据需要计数的个数 N，设置重载值寄存器 AWT_ARR 的值为 N-1；

步骤 8：设置控制寄存器 AWT_CR.EN 为 1, 启动计数器开始计数；

步骤 9：计数器 CNT 下溢出，进入中断服务函数，设置 AWT_ICR.UD 为 0 清除中断标志；

步骤 10：如果不希望继续计数，则设置 AWT 控制寄存器 AWT_CR.EN 为 0 关闭计数器计数功能。

11.4 低功耗模式

当计数器时钟源为 LSE 或 LSI 时，AWT 可在深度休眠模式下保持运行，下溢出中断可唤醒 MCU 回到运行模式。

当选择 LSE（32.768kHz）或 LSI（32.8kHz）作为计数器时钟源时，设置预分频器系数 AWT_CR.PRS 为 0x01，重载值 AWT_ARR 为 0，则最短定时周期为 60μs；设置预分频器系数 AWT_CR.PRS 为 0x0F，重载值 AWT_ARR 为 0xFFFF，则最长定时周期为 65536s。

因此，在低功耗定时模式下，唤醒周期可配置为 60μs ~ 65536s。

11.5 AWT 中断

当 16bit 计数器 CNT 的值递减到 0 后产生下溢出，下溢出中断标志位 AWT_ISR.UD 会被硬件置位。如果允许中断（设置中断使能寄存器 AWT_IER.UD 为 1），CPU 会响应中断服务程序，退出中断服务程序之前应设置中断标志清除寄存器 AWT_ICR.UD 为 0 来清除该标志位，以避免重复进入中断服务程序。

当选择 LSE 或 LSI 作为 AWT 工作时钟源时，AWT 可以在深度休眠模式下继续工作，此时 AWT 中断能唤醒 MCU 到运行模式。

11.6 调试支持

AWT 支持在调试模式下停止或继续计数，通过调试状态定时器控制寄存器 SYSCTRL_DEBUG 的 AWT 位域来设置。设置 SYSCTRL_DEBUG.AWT 为 1，则在调试状态时暂停 AWT 计数器计数；设置 SYSCTRL_DEBUG.AWT 为 0，则在调试状态时 AWT 计数器继续计数。

在调试状态，程序停止运行，如果不停止 AWT 计数器的计数，会很快产生 AWT_ISR.UD 中断标志，导致程序跳转到中断服务程序运行，影响其它功能调试，建议使能调试模式下 AWT 计数器停止计数功能。

11.7 寄存器列表

AWT 基地址：AWT_BASE = 0x4001 4C00

表 11-2 AWT 寄存器列表

寄存器名称	寄存器地址	寄存器描述
AWT_CR	AWT_BASE + 0x00	控制寄存器
AWT_ARR	AWT_BASE + 0x04	重载值寄存器
AWT_CNT	AWT_BASE + 0x08	计数值寄存器
AWT_IER	AWT_BASE + 0x10	中断使能寄存器
AWT_ISR	AWT_BASE + 0x14	中断标志寄存器
AWT_ICR	AWT_BASE + 0x1C	中断标志清除寄存器

11.8 寄存器描述

有关寄存器描述里所使用的缩写，请参见 1 文档约定章节。

11.8.1 AWT_CR 控制寄存器

Address offset: 0x00 Reset value: 0x0000 0000

位域	名称	权限	功能描述
31:11	RFU	-	保留位，请保持默认值
10:8	SRC	RW	计数时钟来源选择 000：HSIOSC 001：LSI 010：HSE 011：LSE 100：ETR 注：当时钟源为 LSI/LSE 时，定时器可在 DeepSleep mode 下工作。
7:4	PRS	RW	计数时钟预分频配置 0000：保留，不可配置 1000：256 分频 0001：2 分频 1001：512 分频 0010：4 分频 1010：1024 分频 0011：8 分频 1011：2048 分频 0100：16 分频 1100：4096 分频 0101：32 分频 1101：8192 分频 0110：64 分频 1110：16384 分频 0111：128 分频 1111：32768 分频
3	RFU	-	保留位，请保持默认值
2:1	MD	RW	请写入 11
0	EN	RW	定时器使能控制 0：禁止 1：使能 注：配置完成 AWT_CR[10:1] 后，才可将 EN 位域设置为 1。

11.8.2 AWT_ARR 重载值寄存器

Address offset: 0x04 Reset value: 0x0000 FFFF

位域	名称	权限	功能描述
31:16	RFU	-	保留位，请保持默认值
15:0	ARR	RW	计数器重载值

11.8.3 AWT_CNT 计数值寄存器

Address offset: 0x08 Reset value: 0x0000 0000

位域	名称	权限	功能描述
31:16	RFU	-	保留位，请保持默认值
15:0	CNT	RO	计数器计数值

11.8.4 AWT_IER 中断使能寄存器

Address offset: 0x10 Reset value: 0x0000 0000

位域	名称	权限	功能描述
31:4	RFU	-	保留位，请保持默认值
3	UD	RW	计数器下溢出中断使能配置 0：禁止 1：使能
2:0	RFU	-	保留位，请保持默认值

11.8.5 AWT_ISR 中断标志寄存器

Address offset: 0x14 Reset value: 0x0000 0000

位域	名称	权限	功能描述
31:4	RFU	-	保留位，请保持默认值
3	UD	RO	计数器下溢出中断标志 0：定时器未溢出 1：定时器已溢出
2:0	RFU	-	保留位，请保持默认值

11.8.6 AWT_ICR 中断标志清除寄存器

Address offset: 0x1C Reset value: 0x0000 003F

位域	名称	权限	功能描述
31:4	RFU	-	保留位，请保持默认值
3	UD	R1W0	计数器下溢出中断标志清除 W0：清除溢出标志 W1：无功能
2:0	RFU	-	保留位，请保持默认值

12 实时时钟（RTC）

12.1 概述

实时时钟（RTC）是一个专用的计数器 / 定时器，可提供日历信息，包括小时、分钟、秒、日、月份、年份以及星期。

RTC 具有两个独立闹钟，时间、日期可组合设定，可产生闹钟中断，并通过引脚输出；支持时间戳功能，可通过引脚触发，记录当前的日期和时间，同时产生时间戳中断；支持周期中断；支持自动唤醒功能，可产生中断并通过引脚输出；支持 1Hz 方波和 RTCOUT 输出功能；支持内部时钟校准补偿。

CW32F020 内置经独立校准的 32kHz 频率的 RC 时钟源，为 RTC 提供驱动时钟，RTC 可在深度休眠模式下运行，适用于要求低功耗的应用场合。

12.2 主要特性

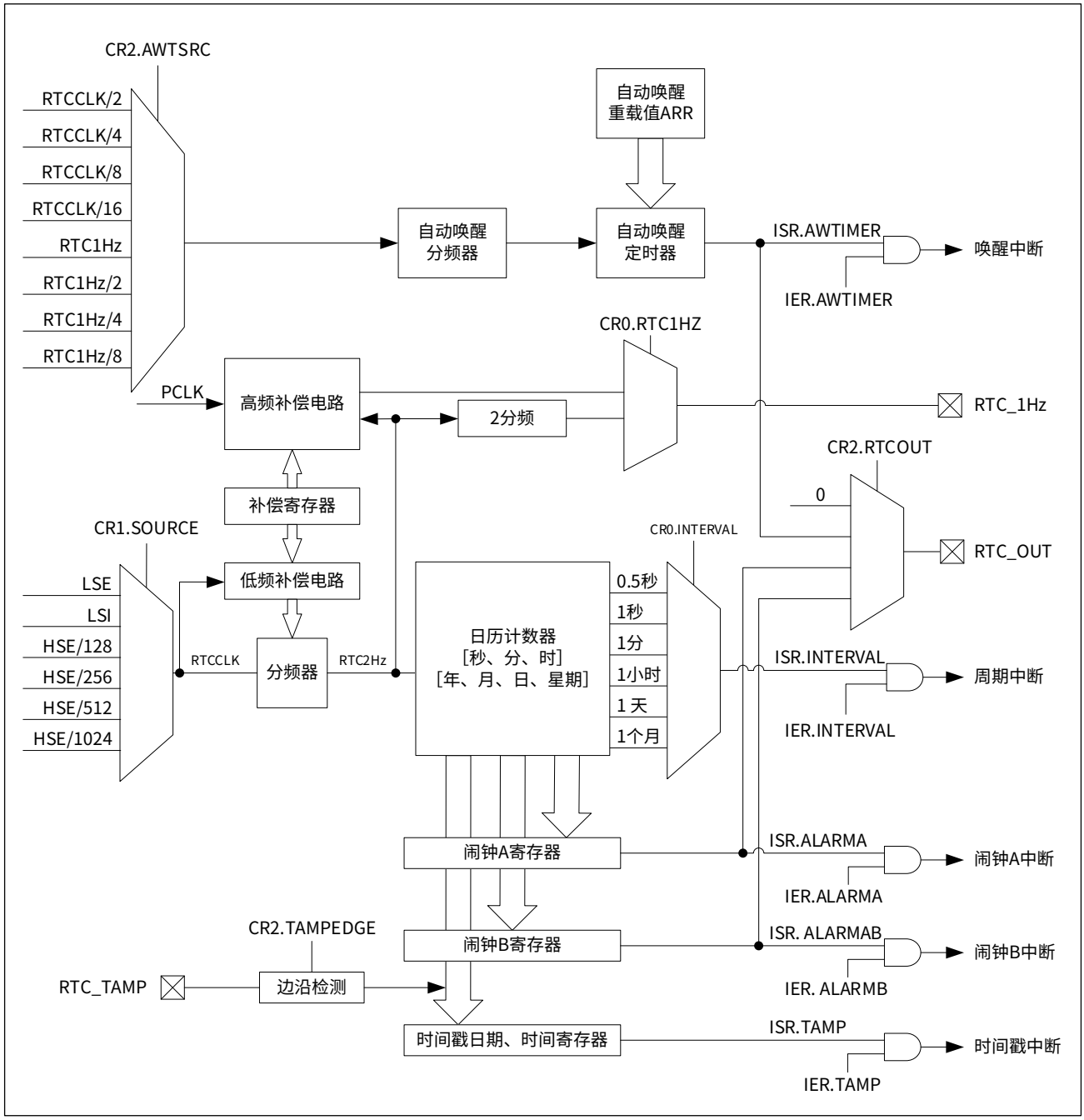
- 日历功能（BCD 码格式）
  - 秒、分、时 (支持 12 / 24 小时制)
  - 星期、日、月份和年份，支持自动闰年修正
- 中断功能
  - 闹钟 A、B 中断，可通过引脚输出
  - 周期中断，宽范围：0.5s ~ 1 个月
  - 自动唤醒中断，范围：61μs ~ 145h，可通过引脚输出
  - 时间戳中断
- 校准补偿功能
  - 最小时钟误差补偿步长，0.119ppm
  - 1Hz 方波输出引脚
- 寄存器锁定保护功能，防止意外修改
- 可工作于运行模式、休眠模式和深度休眠模式

12.3 功能描述

12.3.1 功能框图

实时时钟 (RTC) 的功能框图如下图所示：

图 12-1 RTC 功能框图



12.3.2 功能概述

实时时钟 (RTC) 主要由专用的高精度 RTC 定时器组成，时钟源可选择外部低速时钟 LSE 或内部低速时钟 LSI，当选择外部高速时钟 HSE 时，因精度受限只能用作一般定时 / 计数器。

时间寄存器 RTC_TIME 和日期寄存器 RTC_DATE，以 BCD 码格式分别记录当前的时间和日期值，在对其写入时会自动进行合法性检查，任何非法的时间或日期值将不能被写入，如 32 日、2A 时、61 秒、13 月等。

日期寄存器 RTC_DATE 中，YEAR 位域表示年，有效值 0 ~ 99；MONTH 位域表示月，有效值 1 ~ 12；DAY 位域表示日，有效值 1 ~ 31；WEEK 位域表示星期，有效值 0 ~ 6，其中 0 表示星期日，1 ~ 6 表示星期一至星期六。

时间寄存器 RTC_TIME 中，SECOND 位域表示秒，有效值 0 ~ 59；MINUTE 位域表示分，有效值 0 ~ 59；HOUR 位域代表小时，有效值为 1 ~ 12 或 0 ~ 23，可选择 12 小时制或 24 小时制。控制寄存器 RTC_CR0 的 H24 位域选择 12 或 24 小时制：

- H24 为 ‘1’ 时，选择 24 时制
- H24 为 ‘0’ 时，选择 12 时制，HOUR 位域的最高位代表 AM/PM（上午 / 下午）：
  - ‘0’ 表示 AM
  - ‘1’ 表示 PM

HOUR 位域值详细见下表：

表 12-1 12/24 小时制时 HOUR 位域值

12 时制（RTC_CR0.H24=0）				24 时制（RTC_CR0.H24=1）			
时间	HOUR 位域	时间	HOUR 位域	时间	HOUR 位域	时间	HOUR 位域
AM12 时	0x12	PM12 时	0x32	00 时	0x00	12 时	0x12
AM1 时	0x01	PM1 时	0x21	01 时	0x01	13 时	0x13
AM2 时	0x02	PM2 时	0x22	02 时	0x02	14 时	0x14
AM3 时	0x03	PM3 时	0x23	03 时	0x03	15 时	0x15
AM4 时	0x04	PM4 时	0x24	04 时	0x04	16 时	0x16
AM5 时	0x05	PM5 时	0x25	05 时	0x05	17 时	0x17
AM6 时	0x06	PM6 时	0x26	06 时	0x06	18 时	0x18
AM7 时	0x07	PM7 时	0x27	07 时	0x07	19 时	0x19
AM8 时	0x08	PM8 时	0x28	08 时	0x08	20 时	0x20
AM9 时	0x09	PM9 时	0x29	09 时	0x09	21 时	0x21
AM10 时	0x10	PM10 时	0x30	10 时	0x10	22 时	0x22
AM11 时	0x11	PM11 时	0x31	11 时	0x11	23 时	0x23

12.3.3 RTC 初始化设置

标准的 RTC 模块初始化过程，应包括以下步骤：

步骤 1：向 RTC_KEY 寄存器顺序写入 0xCA、0x53，解除 RTC 寄存器锁定；

步骤 2：配置 RTC_CR0.H24 位域，选择 12/24 小时制；

步骤 3：配置 RTC_CR1.SOURCE 位域，选择 RTC 时钟源；

注：

如选择 LSE 或 LSI，需要先使能和启动 LSE 或 LSI，并等待时钟稳定。

步骤 4：设置正确的时间和日期值，写入时间和日期寄存器；

步骤 5：配置需要的周期中断单元、自动唤醒单元、闹钟 A、闹钟 B；

步骤 6：配置 RTC_IER 寄存器，设置周期中断、自动唤醒中断、闹钟 A、闹钟 B 中断；

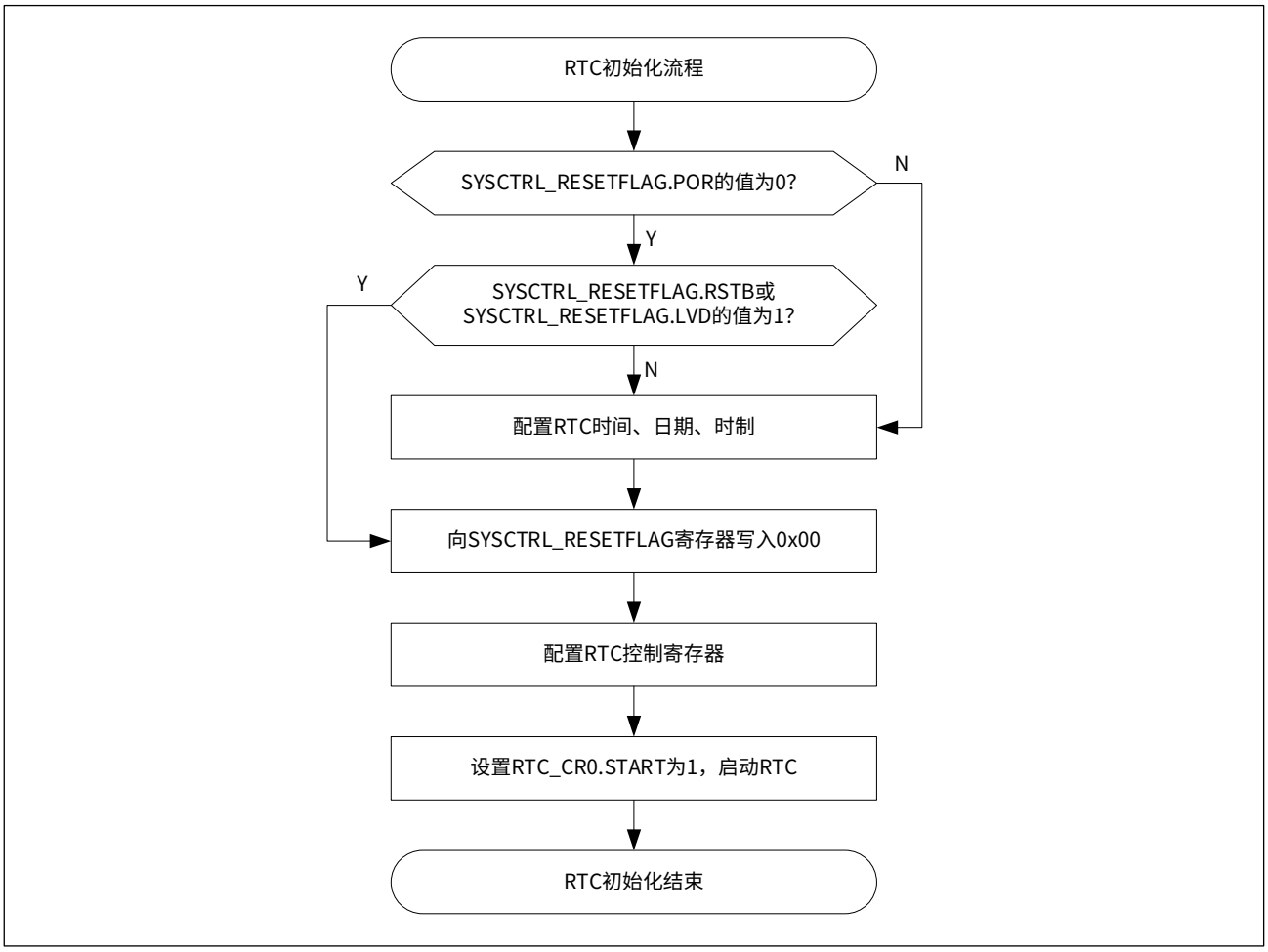
步骤 7：配置时间戳功能，配置时间戳中断；

步骤 8：RTC_CR0.START 位域置 1，启动 RTC 定时器；

步骤 9：向 RTC_KEY 寄存器顺序写入 0xCA、0x--，启动 RTC 寄存器锁定保护。

用户应用中，在对 RTC 进行初始化时，应先检查 RTC 是否已经运行以及 MCU 的复位状态（SYSCTRL_RESETFLAG）。如果 RTC 已经运行，则不再对 RTC 初始化，避免定时不准；如果不是上电复位（SYSCTRL_RESETFLAG.POR = 0 ），则不需要重置 RTC 时间和日期寄存器，只需重新配置 RTC 控制寄存器，并重新启动 RTC 即可。

图 12-2 RTC 初始化流程



12.3.4 寄存器锁定功能

CW32F020 上电复位后，除 RTC_KEY 和 RTC_ICR 之外的 RTC 寄存器都处于锁定保护状态，不可修改。向键值寄存器 RTC_KEY 依次写入 0xCA、0x53 关键字，解除 RTC 寄存器锁定保护；向 RTC_KEY 寄存器依次写入 0xCA、0x--，RTC 寄存器将被锁定。

12.3.5 寄存器访问操作

由于 RTC 计数器的运行时钟可能与 RTC 控制寄存器的时钟不同步，当 RTC 在运行时，为保证能正确访问寄存器内容，同时不影响 RTC 的运行，应通过对控制寄存器 RTC_CR1 的 WINDOW 位域和 ACCESS 位域的配合使用，保证对 RTC 寄存器的访问操作不影响 RTC 的计时精度。

在 RTC 已经运行的状态下访问 RTC 寄存器，需遵守以下操作规范：

步骤 1：循环检测 WINDOW 位是否为 1，若为 0，等待 10ms 再次检测，持续 1000 次，若 1000 次仍未检测到 1，则退出，返回操作未正确完成；

步骤 2：向 RTC_KEY 寄存器顺序写入 0xCA、0x53，解除 RTC 寄存器锁定；

步骤 3：向 ACCESS 写 1，设立访问标志；

步骤 4：执行正常 RTC 寄存器访问操作；

注：

此步骤可实现多条 RTC 寄存器访问指令，但是总执行时间不可超出 1s。

步骤 5：向 ACCESS 写 0，清除访问标志；

步骤 6：向 RTC_KEY 寄存器顺序写入 0xCA、0x--，启动 RTC 寄存器写保护。

注：

如用户需要更快速的读取 RTC 时间与日期信息，可选择连续读两次时间或日期寄存器，只要两次读出内容相同，即可认为读出的数据有效。

12.3.6 RTCOUT 输出

通用 GPIO 端口 PB14 和 PC13，可通过复用功能（PB14，AFR=6；PC13，AFR=4）配置为 RTC_OUT 输出功能。

通过设置控制寄存器 RTC_CR2 的 RTCOUT 位域，可选择 RTC_OUT 输出源，如下表所示：

表 12-2 RTC_OUT 输出配置

RTC_CR2.RTCOUT	RTC_OUT 输出
00	始终输出低电平
01	闹钟 A 匹配标志
10	闹钟 B 匹配标志
11	唤醒定时器溢出标志

12.3.7 1Hz 信号输出

控制寄存器 RTC_CR0 的 RTC1HZ 位域，用来设置 RTC_1Hz 端口输出 1Hz 时钟方波信号。通用 GPIO 端口 PB15 和 PC13，可通过复用功能（PB15，AFR=6；PC13，AFR=2）配置为 RTC_1Hz 输出功能，输出经过补偿或未经补偿的 1Hz 信号，如下表所示：

表 12-3 RTC_1Hz 输出配置

	RTC_CR0.RTC1HZ	RTC_CR1.SOURCE	RTC_1Hz 输出
关闭误差补偿（RTC_COMPEN.EN=0）	01 或 11	000	LSE 的 32768 分频信号
		010	LSI 的 32768 分频信号
开启误差补偿（RTC_COMPEN.EN=1）	01	-	使用 LSE 误差补偿的 1Hz 信号
	11[1]	-	使用 PCLK 误差补偿的 1Hz 信号

注 1：

当开启误差补偿，并设置 RTC_CR0.RTC1HZ 为 3 时，必须设置 RTC_COMPEN.FREQ 为 8，并配置 PCLK 时钟频率为 48MHz。

12.3.8 时钟误差补偿

RTC 内置时钟补偿电路，当 LSE 精度不能满足需求时，可对 RTC 时钟进行补偿，以实现更高精度的 RTC 定时。

RTC 时钟误差补偿寄存器 RTC_COMPEN 有三个位域，用于实现误差补偿：

- SIGN，补偿方向
  - 实际 LSE 时钟频率大于 32768Hz 时，SIGN 设为 0，向下补偿
  - 实际 LSE 时钟频率小于 32768Hz 时，SIGN 设为 1，向上补偿
- STEP，补偿周期，可选择 32、128、256 秒为一个补偿周期
  - 32 秒为一个补偿周期，可实现快速校准但是精度较低，补偿精度为 0.950ppm
  - 选择 128 时，补偿精度是 0.238ppm
  - 选择 256 时，调整过程最平滑，补偿精度是 0.119ppm
- COMP，补偿数值
  - 当 STEP 选择 32 时，COMP 有效范围是 1 ~ 511
  - 当 STEP 选择 128 时，COMP 有效范围是 1 ~ 2047
  - 当 STEP 选择 256 时，COMP 有效范围是 1 ~ 4095

时钟校准操作步骤如下：

步骤 1：关闭误差补偿功能，设置 RTC_COMPEN.EN 为 0；

步骤 2：配置 GPIO 端口（PB15 和 PC13）为 RTC_1Hz 输出，输出未补偿的 1Hz 信号；

步骤 3：依 12.3.3 RTC 初始化设置，配置 LSE 为 RTC 时钟源；

步骤 4：RTC_CR0.START 位域置 1，启动 RTC；

步骤 5：使用高精度频率计对 RTC_1Hz 引脚输出的 1Hz 信号进行测量，记录实际测量频率 f_0；

补偿值 COMP 的计算方法：

假设测量值 f_0= 0.9999888Hz，则 RTC 时钟实际频率 f_z：

$$f_z = 32768 \times 0.9999888 \approx 32767.63Hz$$

频率误差 $_{\Delta}f = 32768 - 32767.63 = 0.37$

选择补偿周期为 256，校准精度最高，计算出补偿数值：

$$COMP = {}_{\Delta}f \times STEP = 0.37 \times 256 = 94.72 \approx 95 = 0x5F$$

步骤 6：按 12.3.5 寄存器访问操作，配置 RTC_COMPEN 补偿寄存器，此例中 SIGN 位域写 1，STEP 位域写 10，COMP 位域为 0x5F；

步骤 7：写 RTC_COMPEN.EN 为 1，启动时钟误差补偿。

注：

步骤 6 和 7 也可合并，直接按半字操作，向 RTC_COMPEN 写入 0xE05F。

12.3.9 闹钟 A 和闹钟 B

RTC 支持 2 个独立闹钟（闹钟 A 和闹钟 B），可在一周内任意时刻产生闹钟事件，并产生闹钟中断，同时将闹钟匹配事件通过外部 RTC_OUT 引脚输出。设置控制寄存器 RTC_CR2 的 ALARMAEN 和 ALARMBEN 位域为 1，可分别单独使能闹钟 A 和闹钟 B。

通过设置闹钟 A、B 控制寄存器（RTC_ALARMA 和 RTC_ALARMB）的时、分、秒匹配控制位 HOUREN、MINUTEEN、SECONDEN 和时、分、秒计数值 HOUR、MINUTE、SECOND，可设定闹钟在‘xx 时 xx 分 xx 秒’，或‘xx 分 xx 秒’或‘xx 时 xx 分’或‘xx 时’等多种组合产生闹钟事件；闹钟星期使能控制位 WEEKMASK，可选择一周中的任意一天产生闹钟事件，bit0 代表星期日，bit1 ~ 6 代表星期一至星期六。

采用 12 或 24 小时制，闹钟控制寄存器 RTC_ALARMx(x=A,B) 的设置值可能不同，示例如下表：

表 12-4 定时闹钟举例

定时要求	时制	RTC_ALARMx 值	闹钟时刻
每天早上 6：30 闹钟	12/24 时制	0x7F06 3000	06：30：00
每工作日 7：00 闹钟	12/24 时制	0x3E07 0000	07：00：00
每小时的 50 分发生一次	12/24 时制	0x7F80 5000	xx：50：00
每分钟的 10 秒发生一次	12/24 时制	0x7F80 8010	xx：xx：10
每周一，中午 12 点每分钟的 5 秒发生一次	12 时制	0x0232 8005	12PM：xx：05
每周一，中午 12 点每分钟的 5 秒发生一次	24 时制	0x0212 8005	12：xx：05
每周日，晚上 12 点 30 分每秒钟发生一次	12 时制	0x0112 3080	12AM：30：xx
每周日，晚上 0 点 30 分每秒钟发生一次	24 时制	0x0100 3080	00：30：xx

当发生闹钟事件时，对应的闹钟匹配标志位 RTC_ISR.ALARMx 会被硬件置 1，如果设置了闹钟中断使能位 RTC_IER.ALARMx 为 1，将产生中断请求。

闹钟 A 或闹钟 B 的匹配标志可通过 RTC_OUT 引脚输出，请参见 12.3.6 RTCOUT 输出。

12.3.10 周期中断功能

RTC 内置周期中断模块，可产生固定周期的中断信号。定时周期通过控制寄存器 RTC_CR0 的 INTERVAL 位域来配置，如下表所示：

表 12-5 定时周期配置

RTC_CR0. INTERVAL	周期中断的周期
000	不产生周期中断
001	0.5 秒，每 1 秒和 0.5 秒时刻产生
010	1 秒，秒计数器发生变化时产生
011	1 分钟，分计数器发生变化时产生
100	1 小时，时计数器发生变化时产生
101	1 天，日计数器发生变化时产生
11x	1 个月，月计数器发生变化时产生

通过 RTC 中断使能寄存器 RTC_IER 的 INTERVAL 位域，可设置周期中断是否产生中断请求。

12.3.11 自动唤醒功能

自动唤醒定时器是一个 16 位可编程自动重载减法计数器，可选择时钟源，定时范围为：61μs ~ 145h。当计数器溢出时，可产生自动唤醒中断，并将溢出标志通过 RTC_OUT 引脚输出。

设置控制寄存器 RTC_CR2 的 AWTEN 位域为 1 使能自动唤醒功能，该功能专为低功耗应用场合而设计，可工作于 MCU 的全部工作模式。

RTC 控制寄存器 RTC_CR2 的 AWTSRC 位域，用于选择自动唤醒定时器的计数时钟源，如下表所示：

表 12-6 自动唤醒定时器时钟源选择

RTC_CR2.AWTSRC	周期中断的时钟源
000	RTCCLK/2（~ 16384Hz）
001	RTCCLK/4（~ 8192Hz）
010	RTCCLK/8（~ 4096Hz）
011	RTCCLK/16（~ 2048Hz）
100	RTC1Hz/1（~ 1Hz）[1]
101	RTC1Hz/2（~ 0.5Hz）[1]
110	RTC1Hz/4（~ 0.25Hz）[1]
111	RTC1Hz/8（~ 0.125Hz）[1]

注 1：选择 RTC1Hz 为时钟源时，RTC 必须运行（RTC_CR0.START = 1）

自动唤醒定时器计数周期由计数时钟源和重载寄存器 RTC_AWTARR 决定，定时时长计算公式为：

自动唤醒定时器定时周期 = （RTC_AWTARR+1）/ 唤醒定时器时钟频率

最短定时：(0+1) / 16384Hz = 61μs

最长定时：(65535+1) / 0.125Hz = 524288s = 8738min ≈ 145.63h

通过 RTC 中断使能寄存器 RTC_IER 的 AWTIMER 位域，可选择自动唤醒定时器溢出时是否产生中断请求。自动唤醒定时器的溢出标志可通过 RTC_OUT 引脚输出，请参见 12.3.6 RTCOUT 输出。

12.3.12 时间戳功能

RTC 支持时间戳功能，即通过 RTC_TAMP 引脚触发，将当前时间和日期分别保存到时间戳日期寄存器 RTC_TAMPDATE 和时间戳时间寄存器 RTC_TAMPTIM，同时可产生时间戳中断。

控制寄存器 RTC_CR2 的 TAMPEDGE 位域用来选择触发时间戳的信号是上升沿还是下降沿有效，RTC_CR2 寄存器的 TAMPEN 位域用于使能时间戳功能。

用户可灵活选择触发引脚 RTC_TAMP，并需配置该引脚为数字输入和复用功能，如下表所示：

表 12-7 RTC_TAMP 引脚配置

GPIO	AFR
PA00	0x03
PA01	0x07
PC13	0x07

当发生时间戳事件时，时间戳事件标志位 RTC_ISR.TAMP 会被置 1，如果设置了时间戳中断使能位 RTC_IER.TAMP 为 1，将产生中断请求。

如果发生第一次时间戳事件后，未软件清除 RTC_ISR.TAMP 标志位，又产生了第二次时间戳事件，时间戳溢出标志位 RTC_ISR.TAMPOV 会被置 1，如果设置了时间戳溢出中断使能位 RTC_IER.TAMPOV 为 1，将产生中断请求。

12.3.13 RTC 中断

RTC 支持 6 个中断源：唤醒中断、闹钟 A 中断、闹钟 B 中断、周期中断、时间戳中断和时间戳溢出中断。

中断的开启由中断使能寄存器 RTC_IER 控制，对应位域写入‘1’允许中断，写‘0’禁止中断；中断发生时会硬件自动置位中断标志寄存器 RTC_ISR 的对应位；通过向中断标志清除寄存器 RTC_ICR 的对应位写入‘0’，可清除相应的中断标志。

用户应用程序中，启用 RTC 中断功能后，所有 RTC 中断事件发生时都会从 RTC 中断入口（RTC_IRQHandler）调用同一个中断服务程序，用户应当在中断服务程序中查询 RTC 中断标志寄存器 RTC_ISR，以确定引发中断的具体 RTC 中断源。

在中断事件处理完成后，退出中断服务程序之前，必须向 RTC 中断标志清除寄存器 RTC_ICR 的对应位写入‘0’，以清除中断标志，避免重复进入中断服务程序。

12.4 寄存器列表

RTC 基地址：RTC_BASE = 0x4000 2800

表 12-8 RTC 寄存器列表

寄存器名称	寄存器地址	寄存器描述
RTC_KEY	RTC_BASE + 0x00	键值寄存器
RTC_CR0	RTC_BASE + 0x04	控制寄存器 0
RTC_CR1	RTC_BASE + 0x08	控制寄存器 1
RTC_CR2	RTC_BASE + 0x0C	控制寄存器 2
RTC_COMPEN	RTC_BASE + 0x10	时钟误差补偿寄存器
RTC_DATE	RTC_BASE + 0x14	日期寄存器
RTC_TIME	RTC_BASE + 0x18	时间寄存器
RTC_ALARMA	RTC_BASE + 0x1C	闹钟 A 控制寄存器
RTC_ALARMB	RTC_BASE + 0x20	闹钟 B 控制寄存器
RTC_TAMPDATE	RTC_BASE + 0x24	时间戳日期寄存器
RTC_TAMPTIME	RTC_BASE + 0x28	时间戳时间寄存器
RTC_AWTARR	RTC_BASE + 0x2C	唤醒定时器重载寄存器
RTC_IER	RTC_BASE + 0x30	中断使能寄存器
RTC_ISR	RTC_BASE + 0x34	中断标志寄存器
RTC_ICR	RTC_BASE + 0x38	中断标志清除寄存器

12.5 寄存器描述

有关寄存器描述里所使用的缩写，请参见 1 文档约定章节。

12.5.1 RTC_KEY 键值寄存器

Address offset: 0x00 Reset value: 0x0000 0000

位域	名称	权限	功能描述
31:8	RFU	-	保留位，请保持默认值
7:0	KEY	WO	键值寄存器 向该寄存器依次写入“0x00CA”，“0x0053”，方可修改 RTC 模块的寄存器。 向该寄存器依次写入“0x00CA”，“0x----”，RTC 模块除 KEY、ICR 以外寄存器不可修改。 注：“0x----”为除“0x0053”之外的任意值。

12.5.2 RTC_CR0 控制寄存器 0

Address offset: 0x04 Reset value: 0x0000 0000

位域	名称	权限	功能描述
31:8	RFU	-	保留位，请保持默认值
7	START	RW	RTC 计时器运行控制 0：停止 RTC 计时器 1：启动 RTC 计时器
6:5	RTC1HZ	RW	RTC_1HZ 输出配置 01：采用 LSE 时钟补偿的 1Hz 信号 11：采用 PCLK 时钟补偿的 1Hz 信号 注 1：若 RTC_COMPEN.EN 为 0，则输出 LSE 或 LSI 的 32768 分频。 注 2：若 RTC1HZ 为 11，则需正确配置 RTC_COMPEN. FREQ。
4	RFU	-	保留位，请保持默认值
3	H24	RW	计时时制设置 0：12 小时制 1：24 小时制
2:0	INTERVAL	RW	定时周期设置 000：不产生定时周期 001：0.5 秒，在秒计时器发生变化和 0.5 秒时刻时产生 010：1 秒钟，在秒计时器发生变化时产生 011：1 分钟，在分计时器发生变化时产生 100：1 小时，在时计时器发生变化时产生 101：1 天， 在天计时器发生变化时产生 11x：1 月， 在月计时器发生变化时产生

12.5.3 RTC_CR1 控制寄存器 1

Address offset: 0x08 Reset value: 0x0000 0000

位域	名称	权限	功能描述
31:11	RFU	-	保留位，请保持默认值
10:8	SOURCE	RW	计时时钟 RTCCLK 来源配置 000：外部低频时钟 LSE 010：内置低频时钟 LSI 100：外部高速时钟 HSE，128 分频 101：外部高速时钟 HSE，256 分频 110：外部高速时钟 HSE，512 分频 111：外部高速时钟 HSE，1024 分频
7:2	RFU	-	保留位，请保持默认值
1	WINDOW	RO	RTC 寄存器访问窗口状态 0：非读写窗口，不建议对寄存器进行读写 1：读写窗口，可以对寄存器进行读写
0	ACCESS	RW	RTC 计时相关寄存器访问控制 0：正常计数模式 1：写入 / 读出模式 注：需要对寄存器读写访问时需置位该控制位，并在 1 秒的时间内完成写入 / 读出操作，然后将该位清 0。若 ACCESS 置 1 的时间超过 1 秒，可能造成计时误差。

12.5.4 RTC_CR2 控制寄存器 2

Address offset: 0x0C Reset value: 0x0000 0000

位域	名称	权限	功能描述
31:11	RFU	-	保留位，请保持默认值
10	ALARMBEN	RW	闹钟 B 使能控制 0：禁止 1：使能
9	ALARMAEN	RW	闹钟 A 使能控制 0：禁止 1：使能
7	AWTEN	RW	唤醒定时器使能控制 0：禁止唤醒定时器 1：使能唤醒定时器
6	TAMPEN	RW	时间戳使能控制 0：禁止时间戳功能 1：使能时间戳功能
5:4	RTCOUT	RW	RTC_OUT 输出配置 00：输出低电平 01：闹钟 A 匹配标志 10：闹钟 B 匹配标志 11：唤醒定时器溢出标志
3	TAMPEDGE	RW	时间戳信号边沿配置 0：在 RTC_TAMP 引脚的上升沿记录当前时间 1：在 RTC_TAMP 引脚的下降沿记录当前时间
2:0	AWTSRC	RW	唤醒定时器计数时钟来源配置 000：RTCCLK / 2 ~ 16384Hz 001：RTCCLK / 4 ~ 8192Hz 010：RTCCLK / 8 ~ 4096Hz 011：RTCCLK / 16 ~ 2048Hz 100：RTC1Hz / 1 ~ 1Hz 101：RTC1Hz / 2 ~ 0.5Hz 110：RTC1Hz / 4 ~ 0.25Hz 111：RTC1Hz / 8 ~ 0.125Hz 注：0.125Hz ~ 1Hz 需要使能 RTC

12.5.5 RTC_COMPEN 时钟误差补偿寄存器

Address offset: 0x10 Reset value: 0x0000 0000

位域	名称	权限	功能描述
31:20	RFU	-	保留位，请保持默认值
19:16	FREQ	RW	采用 PCLK 对 RTC 进行补偿时，PCLK 时钟的频率 请写入 0x08，并配置 PCLK 时钟的频率为 48MHz
15	EN	RW	时钟误差补偿使能控制 0：禁止时钟误差补偿 1：使能时钟误差补偿
14	SIGN	RW	时钟补偿方向 0：增加计数值（LSE 振荡频率大于 32768Hz） 1：减小计数值（LSE 振荡频率小于 32768Hz）
13:12	STEP	RW	补偿步长选择 00：0.950ppm，补偿率为 COMP[08:0] / 32768 / 32 01：0.238ppm，补偿率为 COMP[10:0] / 32768 / 128 10：0.119ppm，补偿率为 COMP[11:0] / 32768 / 256 注：补偿精度为补偿步长的一半
11:0	COMP	RW	补偿值 可补偿范围为 ±488.0ppm

12.5.6 RTC_DATE 日期寄存器

Address offset: 0x14 Reset value: 0x0000 0000

位域	名称	权限	功能描述
31:27	RFU	-	保留位，请保持默认值
26:24	WEEK	RW	周计数值 BCD 码格式，有效值范围为 0-6 0 代表周日，1 ~ 6 代表周一到周六
23:16	YEAR	RW	年计数值 BCD 码格式，有效值范围为 00-99
15:8	MONTH	RW	月计数值 BCD 码格式，有效值范围为 1 ~ 12
7:0	DAY	RW	日计数值 BCD 码格式，有效值范围为 1 ~ 28，1 ~ 29，1 ~ 30，1 ~ 31

12.5.7 RTC_TIME 时间寄存器

Address offset: 0x18 Reset value: 0x0012 0000

位域	名称	权限	功能描述
31:22	RFU	-	保留位，请保持默认值
21:16	HOUR	RW	时计数值 24 小时制，时计数 BCD 码有效值范围为 0-23 12 小时制，时计数 BCD 码有效值范围为 1 ~ 12(AM)、21 ~ 32(PM) bit21 为 0 代表 AM，为 1 代表 PM
15	RFU	-	保留位，请保持默认值
14:8	MINUTE	RW	分计数值 BCD 码格式，有效值范围为 0 ~ 59
7	RFU	-	保留位，请保持默认值
6:0	SECOND	RW	秒计数值 BCD 码格式，有效值范围为 0 ~ 59

12.5.8 RTC_ALARMA 闹钟 A 控制寄存器

Address offset: 0x1C Reset value: 0x0012 0000

位域	名称	权限	功能描述
31	RFU	-	保留位，请保持默认值
30:24	WEEKMASK	RW	闹钟星期使能控制 bit0 代表周日，bit1 ~ bit6 分别代表周一到周六 相应的 bit 为 1，则使能该闹钟
23	HOUREN	RW	闹钟小时位屏蔽控制 0：该闹钟需检查小时位计数值 1：该闹钟与小时位计数值无关
22	RFU	-	保留位，请保持默认值
21:16	HOUR	RW	闹钟小时计数值，BCD 格式
15	MINUTEEN	RW	闹钟分钟位屏蔽控制 0：该闹钟需检查分钟位计数值 1：该闹钟与分钟位计数值无关
14:8	MINUTE	RW	闹钟分钟计数值，BCD 格式
7	SECONDEN	RW	闹钟秒钟位屏蔽控制 0：该闹钟需检查秒钟位计数值 1：该闹钟与秒钟位计数值无关
6:0	SECOND	RW	闹钟秒钟计数值，BCD 格式

12.5.9 RTC_ALARMB 闹钟 B 控制寄存器

Address offset: 0x20 Reset value: 0x0012 0000

位域	名称	权限	功能描述
31	RFU	-	保留位，请保持默认值
30:24	WEEKMASK	RW	闹钟星期使能控制 bit0 代表周日，bit1 ~ bit6 分别代表周一到周六 相应的 bit 为 1，则使能该闹钟
23	HOUREN	RW	闹钟小时位屏蔽控制 0：该闹钟与小时位计数值无关 1：该闹钟需检查小时位计数值
22	RFU	-	保留位，请保持默认值
21:16	HOUR	RW	闹钟小时计数值，BCD 格式
15	MINUTEEN	RW	闹钟分钟位屏蔽控制 0：该闹钟与分钟位计数值无关 1：该闹钟需检查分钟位计数值
14:8	MINUTE	RW	闹钟分钟计数值，BCD 格式
7	SECONDEN	RW	闹钟秒钟位屏蔽控制 0：该闹钟与秒钟位计数值无关 1：该闹钟需检查秒钟位计数值
6:0	SECOND	RW	闹钟秒钟计数值，BCD 格式

12.5.10 RTC_TAMPDATE 时间戳日期寄存器

Address offset: 0x24 Reset value: 0x0000 0000

位域	名称	权限	功能描述
31:16	RFU	-	保留位，请保持默认值
15:13	WEEK	RO	发生时间戳信号时的周计数值
12:8	MONTH	RO	发生时间戳信号时的月计数值
7:6	RFU	-	保留位，请保持默认值
5:0	DAY	RO	发生时间戳信号时的日计数值

12.5.11 RTC_TAMPTIME 时间戳时间寄存器

Address offset: 0x28 Reset value: 0x0000 0000

位域	名称	权限	功能描述
31:22	RFU	-	保留位，请保持默认值
21:16	HOUR	RO	发生时间戳信号时的时计数值
15	RFU	-	保留位，请保持默认值
14:8	MINUTE	RO	发生时间戳信号时的分计数值
7	RFU	-	保留位，请保持默认值
6:0	SECOND	RO	发生时间戳信号时的秒计数值

12.5.12 RTC_AWTARR 唤醒定时器重载值

Address offset: 0x2C Reset value: 0x0000 FFFF

位域	名称	权限	功能描述
31:16	RFU	-	保留位，请保持默认值
15:0	ARR	RW	唤醒定时器重载值

12.5.13 RTC_IER 中断使能寄存器

Address offset: 0x30 Reset value: 0x0000 0000

位域	名称	权限	功能描述
31:7	RFU	-	保留位，请保持默认值
6	INTERVAL	RW	定时周期中断使能控制 0：禁止 1：使能
5	RFU	-	保留位，请保持默认值
4	TAMPOV	RW	时间戳溢出中断使能控制 0：禁止 1：使能
3	TAMP	RW	时间戳中断使能控制 0：禁止 1：使能
2	AWTIMER	RW	唤醒定时器中断使能控制 0：禁止 1：使能
1	ALARMB	RW	闹钟 B 中断使能控制 0：禁止 1：使能
0	ALARMA	RW	闹钟 A 中断使能控控制 0：禁止 1：使能

12.5.14 RTC_ISR 中断标志寄存器

Address offset: 0x34 Reset value: 0x0000 0000

位域	名称	权限	功能描述
31:7	RFU	-	保留位，请保持默认值
6	INTERVAL	RO	周期性定时完成标志 0：未发生周期定时完成事件 1：已发生周期定时完成事件
5	RFU	-	保留位，请保持默认值
4	TAMPOV	RO	时间戳溢出标志 0：未发生时间戳溢出事件 1：已发生时间戳溢出事件
3	TAMP	RO	时间戳事件标志 0：未发生时间戳事件 1：已发生时间戳事件
2	AWTIMER	RO	唤醒定时器定时完成标志 0：未发生唤醒定时器定时完成事件 1：已发生唤醒定时器定时完成事件
1	ALARMB	RO	闹钟 B 匹配标志 0：未发生闹钟 B 匹配事件 1：已发生闹钟 B 匹配事件
0	ALARMA	RO	闹钟 A 匹配标志 0：未发生闹钟 A 匹配事件 1：已发生闹钟 A 匹配事件

12.5.15 RTC_ICR 中断标志清除寄存器

Address offset: 0x38 Reset value: 0x0000 007F

位域	名称	权限	功能描述
31:7	RFU	-	保留位，请保持默认值
6	INTERVAL	R1W0	周期性定时完成标志清除 W0：清除周期性定时完成标志 W1：无功能
5	RFU	-	保留位，请保持默认值
4	TAMPOV	R1W0	时间戳溢出标志清除 W0：清除时间戳溢出标志 W1：无功能
3	TAMP	R1W0	时间戳事件标志清除 W0：清除时间戳事件标志 W1：无功能
2	AWTIMER	R1W0	唤醒定时器定时完成标志清除 W0：清除唤醒定时器定时完成标志 W1：无功能
1	ALARMB	R1W0	闹钟 B 匹配标志清除 W0：清除闹钟 B 匹配标志 W1：无功能
0	ALARMA	R1W0	闹钟 A 匹配标志清除 W0：清除闹钟 A 匹配标志 W1：无功能

13 基本定时器（BTIM）

13.1 概述

CW32F020 内部集成 3 个基本定时器 (BTIM)，每个 BTIM 完全独立且功能完全相同，各包含一个 16bit 自动重装载计数器并由一个可编程预分频器驱动。BTIM 支持定时器模式、计数器模式、触发启动模式和门控模式 4 种工作模式，支持溢出事件触发中断请求和 DMA 请求。得益于对触发信号的精细处理设计，使得 BTIM 可以由硬件自动执行触发信号的滤波操作，还能令触发事件产生中断和 DMA 请求。

13.2 主要特性

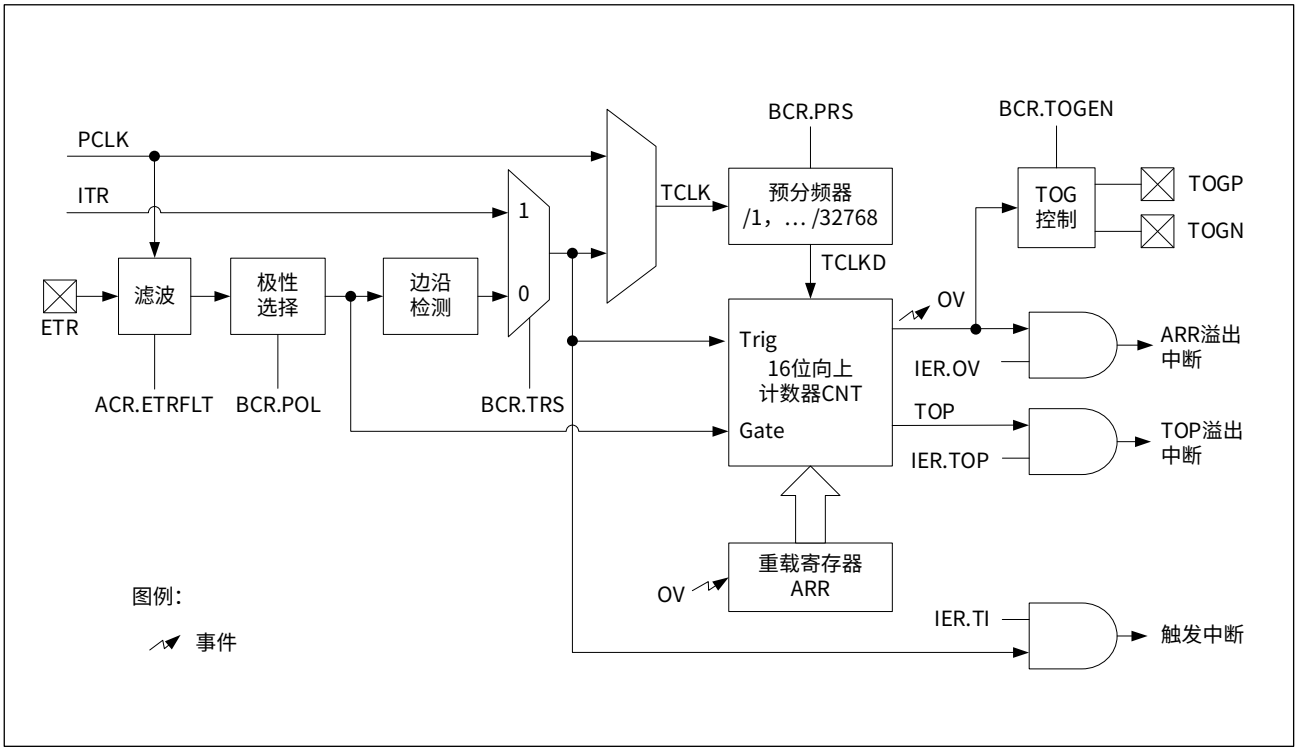
- 16bit 自动重载向上计数器
- 可编程预分频器支持 1，2，4，8，…，32768 分频
- 支持单次计数模式和连续计数模式
- 对内部 ITR 或外部 ETR 信号的计数功能
- 内部 ITR 或外部 ETR 信号触发启动计数
- 由外部 ETR 输入信号控制的门控功能
- 灵活的外部 ETR 信号滤波处理
- 溢出事件触发数字输出电平翻转
- 内部级联 ITR 和外部互联 ETR
- 计数器溢出触发中断 /DMA 请求
- 内部 ITR 或外部 ETR 信号触发中断 /DMA 请求

13.3 功能描述

13.3.1 功能框图

BTIM 的功能框图如下图所示：

图 13-1 BTIM 功能框图



13.3.1.1 滤波单元

BTIM 支持对外部 BTIM_ETR 引脚输入的信号进行滤波处理。滤波器以一定的采样频率对 ETR 信号进行采样，当连续采样到 N 个相同电平时信号有效，否则信号无效，以此滤除高频杂波信号。

滤波单元的采样时钟为 PCLK 或 PCLK 的分频，通过高级控制寄存器 BTIMx_ACR 的 ETRFLT 位域可以选择 PCLK 的分频比及采样点个数 N。

注：

由于硬件是以 PCLK 时钟或 PCLK 时钟的分频对外部 BTIM_ETR 引脚输入的信号进行采样的，因此频率高于实际采样频率二分之一的输入信号存在被漏采样的可能。

下面举例说明 BTIMx_ACR.ETRFLT 的配置和截止频率的关系。

例：

当 PCLK 时钟频率为 24MHz 时，设置 BTIMx_ACR.ETRFLT 为 0x05（即 F_{sample} = PCLK / 4 且 N = 6），

则 F_{sample} = 6MHz，N = 6

连续采样到 6 个有效电平，需要的时间为：

$$6 \times T_{sample} = 1\mu s$$

因此，0.5MHz 频率以上的输入信号将被滤除。

13.3.1.2 极性选择单元

当 BTIM 配置为门控模式时，极性选择单元用来选择外部 BTIM_ETR 引脚输入的信号是高电平有效还是低电平有效，具体通过基本控制寄存器 BTIMx_BCR 的 POL 位域来选择。

当设置 BTIMx_BCR.POL 为 0 时，ETR 信号为高电平时有效；当设置 BTIMx_BCR.POL 为 1 时，ETR 信号为低电平时有效。

13.3.1.3 边沿检测单元

当 BTIM 配置为触发启动模式时，边沿检测单元用来选择外部 BTIM_ETR 引脚输入的信号的边沿触发时机，具体通过基本控制寄存器 BTIMx_BCR 的 POL 位域来选择。

当设置 BTIMx_BCR.POL 为 0 时，ETR 信号为上升沿时有效；当设置 BTIMx_BCR.POL 为 1 时，ETR 信号为下降沿时有效。

13.3.1.4 预分频器

预分频器以 2 的倍数为系数对计数器时钟源 TCLK 分频，最大分频系数为 32768。预分频系数由基本控制寄存器 BTIMx_BCR 的 PRS 位域配置，分频系数配置如下表所示：

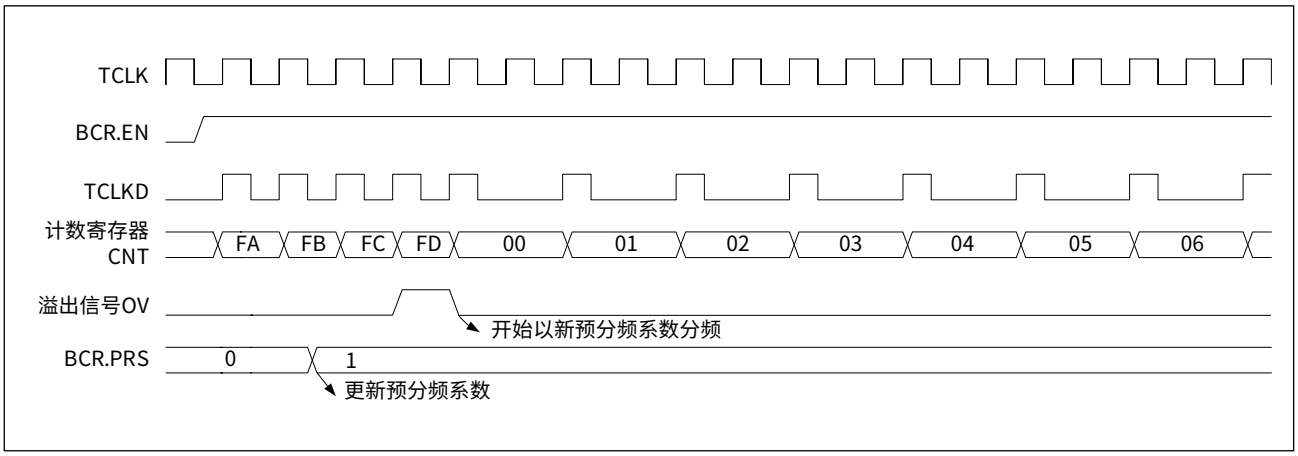
表 13-1 预分频器分频系数配置表

PRS	分频比	PRS	分频比	PRS	分频比	PRS	分频比
0x00	1	0x04	16	0x08	256	0x0C	4096
0x01	2	0x05	32	0x09	512	0x0D	8192
0x02	4	0x06	64	0x0A	1024	0x0E	16384
0x03	8	0x07	128	0x0B	2048	0x0F	32768

在定时器运行过程中允许修改 BTIMx_BCR.PRS，但新的预分频系数不会立即生效，当定时器发生计数溢出或者运行控制位 BTIMx_BCR.EN 由 0 变为 1 时，新的预分频系数生效。

在运行过程中改变预分频系数示例如下图所示：

图 13-2 预分频系数从 1 变到 2 的计数器时序图 (ARR=0xFD)



13.3.1.5 计数单元

计数单元的核心组件是一个 16bit 向上计数器 CNT 和一个 16bit 自动重载寄存器 ARR。

在定时器运行过程中允许修改重载寄存器 ARR，且 ARR 的值将立即生效。当计数器 CNT 的值小于 ARR 的值时，将当前的 ARR 值修改为小于当前 CNT 值后，不会产生 ARR 溢出信号 OV，计数器 CNT 将一直向上计数到 0xFFFF 后，产生 TOP 溢出信号，并从 0x0000 重新开始计数。

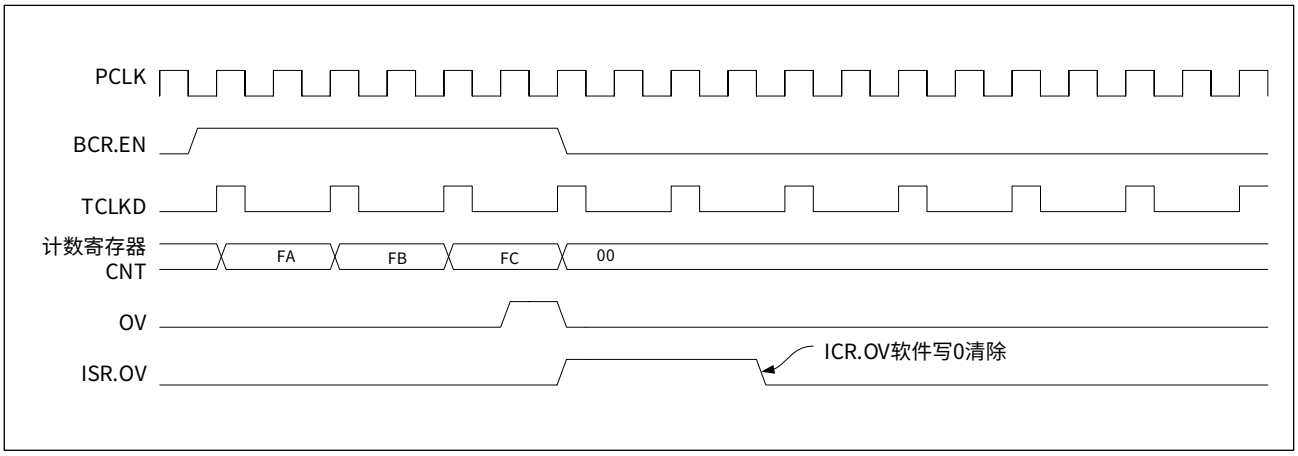
计数器可工作在单次计数或连续计数模式下，通过基本控制寄存器 BTIMx_BCR 的 ONESHOT 位域来选择。

单次计数模式

设置 BTIMx_BCR.ONESHOT 为 1，使定时器工作在单次计数模式下。设置 BTIMx_BCR.EN 为 1 使能 BTIMx，计数器 CNT 在计数时钟 TCLKD 的驱动下累加计数。当计数值到达重载值 ARR 后产生溢出信号 OV（溢出信号 OV 保持一个 PCLK 周期，然后自动清除），OV 信号被清除时，计数器 ARR 溢出标志位 BTIMx_ISR.OV 被硬件置位，同时计数器停止计数，BTIMx_BCR.EN 被硬件自动复位。

下图是单次计数模式示例：

图 13-3 单次计数模式 (PRS=0x01，ARR=0xFC)

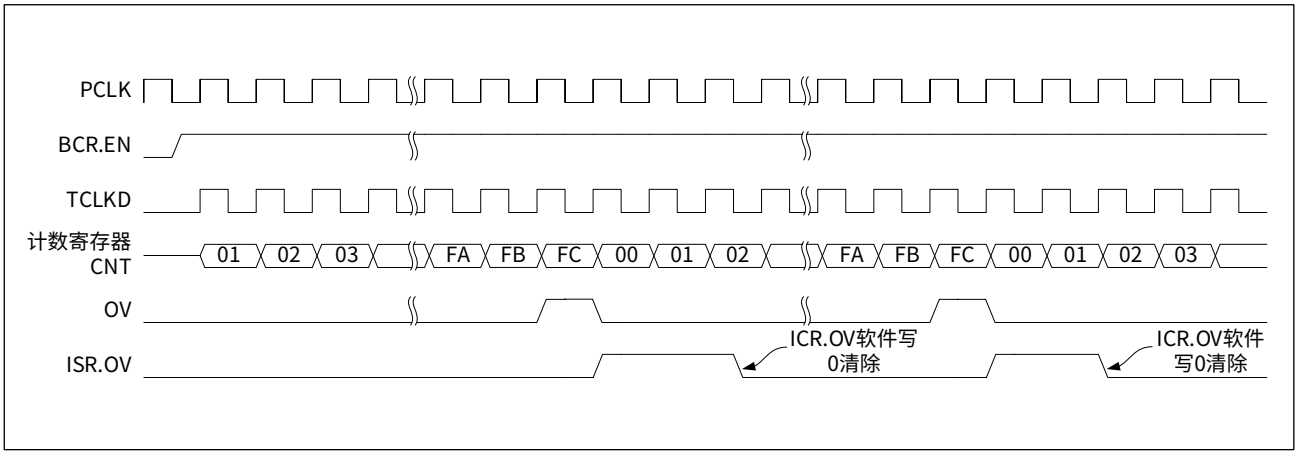


连续计数模式

设置 BTIMx_BCR.ONESHOT 为 0，使定时器工作在连续计数模式下。设置 BTIMx_BCR.EN 为 1 使能 BTIMx，计数器 CNT 在计数时钟 TCLKD 的驱动下累加计数。当计数值到达重载值 ARR 后产生溢出信号 OV（溢出信号 OV 保持一个 PCLK 周期，然后自动清除）。当计数值从 ARR 变为 0 时，计数器 ARR 溢出标志位 BTIMx_ISR.OV 被硬件置位，计数器开始下一个周期的累加计数。

下图是连续计数模式示例：

图 13-4 连续计数模式 (PRS=0x00，ARR=0xFC)



13.3.1.6 翻转输出单元

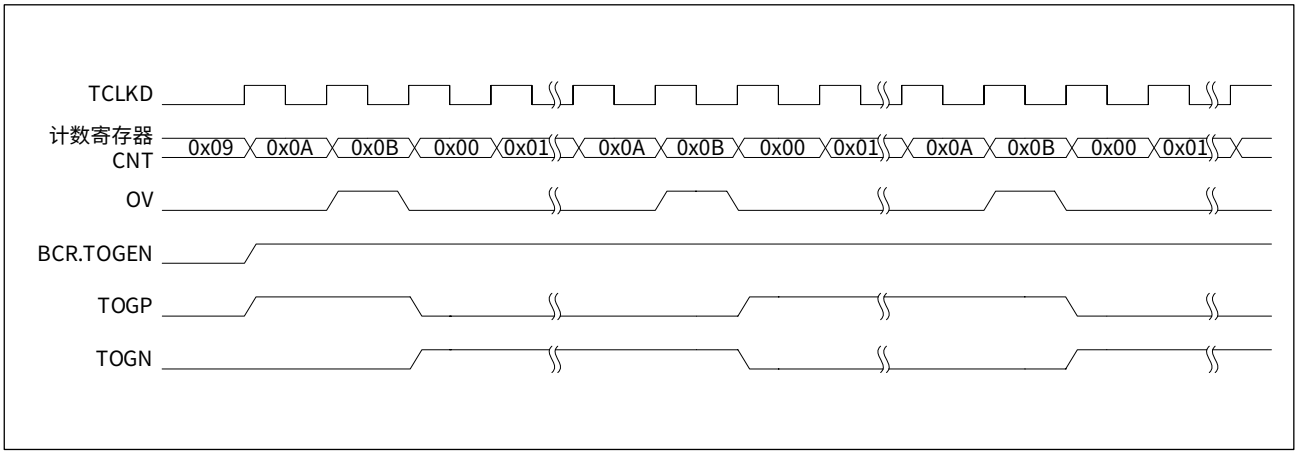
翻转输出单元可通过 ARR 溢出信号 OV 控制外部 BTIMx_TOGP 和 BTIMx_TOPN 引脚输出翻转信号。

当设置 BTIMx_BCR.TOGEN 为 0 时，BTIMx_TOGP 和 BTIMx_TOPN 引脚均输出低电平。

当设置 BTIMx_BCR.TOGEN 为 1 时，BTIMx_TOGP 和 BTIMx_TOPN 引脚输出电平相反的信号（BTIMx_TOGP 默认电平为高电平）；当计数器 ARR 溢出时 (BTIMx_ISR.OV 为 1)，BTIMx_TOGP 和 BTIMx_TOPN 引脚输出电平将翻转。

下图所示为连续计数模式下，BTIMx_TOGP 和 BTIMx_TOPN 引脚电平翻转输出示意图：

图 13-5 电平翻转输出示意图（连续计数模式，ARR=0x0B）



BTIM 支持的 TOGP/TOGN 引脚如下表所示，在应用时需要复用对应的 GPIO 引脚。

表 13-2 BTIM 翻转输出引脚

TOGP/TOGN	引脚	AFR
BTIM1_TOGP	PA07/PA09/PF00	0x04
BTIM1_TOGN	PA10/PB00/PF01	0x04
BTIM2_TOGP	PA05/PC14	0x04
BTIM2_TOGP	PB15	0x03
BTIM2_TOGN	PA06/PC15	0x04
BTIM2_TOGN	PA08	0x03
BTIM3_TOGP	PB01	0x05
BTIM3_TOGP	PF06	0x07
BTIM3_TOGN	PB02	0x05
BTIM3_TOGN	PF07	0x07

13.3.2 工作模式

BTIM 支持 4 种工作模式：定时器模式、计数器模式、触发启动模式和门控模式。通过基本控制寄存器 BTIMx_BCR 的 MODE 位域来配置。

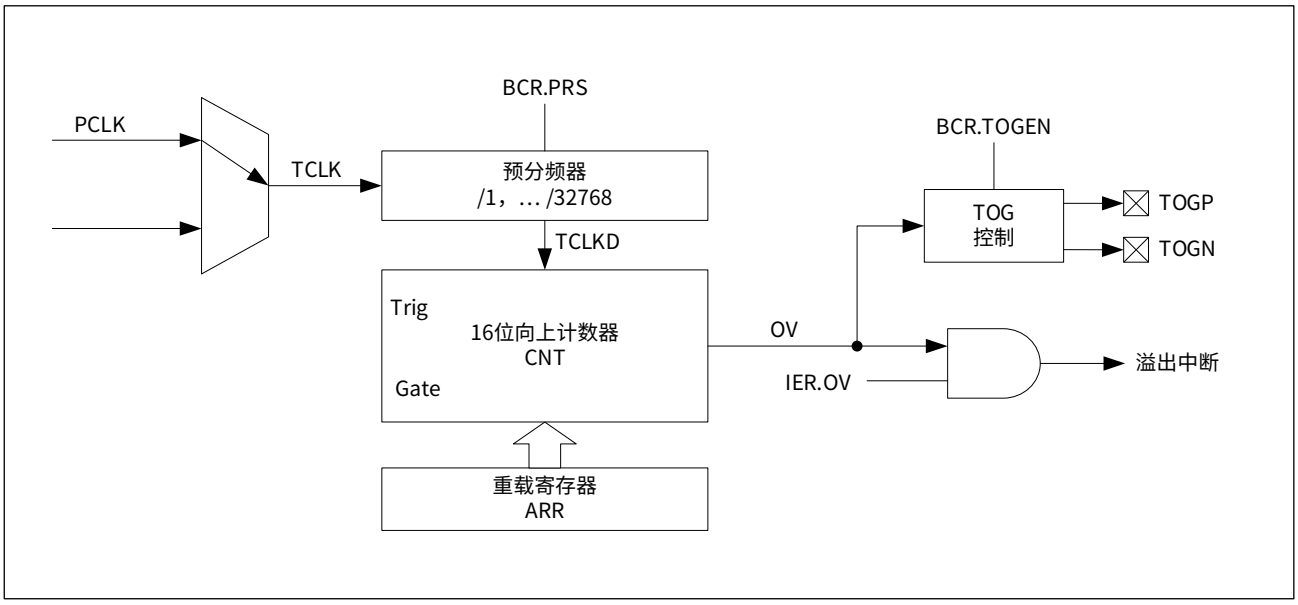
表 13-3 BTIM 工作模式

基本定时器	MODE 位域值	工作模式	描述
BTIMx_BCR (x=1,2,3)	00	定时器模式	时钟源为 PCLK
	01	计数器模式	计数源为 TRS 信号
	10	触发启动模式	时钟源为 PCLK，TRS 信号触发计数器启动
	11	门控模式	时钟源为 PCLK，ETR 输入信号作为门控信号

13.3.2.1 定时器模式

定时器模式主要用于产生固定时间间隔的时基信号，在该模式下，计数器时钟源为内部系统时钟 PCLK，经预分频器分频后得到计数时钟 TCLKD，来驱动计数器 CNT 计数。定时器模式功能框图如下图所示：

图 13-6 定时器模式框图



设置 BTIMx_BCR.MODE 为 0x00，使 BTIMx 工作于定时器模式。设置 BTIMx_BCR.EN 为 1 使能 BTIMx，计数器 CNT 将在计数时钟 TCLKD 的驱动下累加计数。当计数值到达重载值 ARR 后产生溢出，计数器 ARR 溢出标志位 BTIMx_ISR.OV 被硬件置位，如果允许中断 (设置中断使能寄存器 BTIMx_IER.OV 为 1)，CPU 将响应中断服务程序。退出中断服务程序之前，应设置中断标志清除寄存器 BTIMx_ICR.OV 为 0 以清除该标志。

在实际应用中，系统时钟 PCLK 的频率是已知的，通过合理设置预分频器系数 PRS 可以对固定时长的时钟进行计数，配合对重载值 ARR 的设置及计数器 ARR 溢出中断标志位的使用，可以精确获得某一特定时长，从而达到定时的目地。

定时时间 T 计算公式：

$$T = (2^{PRS} / PCLK) \times (ARR + 1)$$

其中，PCLK 为计数器时钟源，PRS 为预分频系数，ARR 为重载值。

例：

当计数器时钟源 PCLK 的频率为 24MHz 时，要求定时 100ms。

如果设置预分频系数 PRS 为 0x08，计算：

$$T = 100ms = (2^{8} / 24MHz) \times (ARR + 1)$$

则

ARR = 9374 (0x249E)

即需要设置重载值 ARR 为 0x249E。

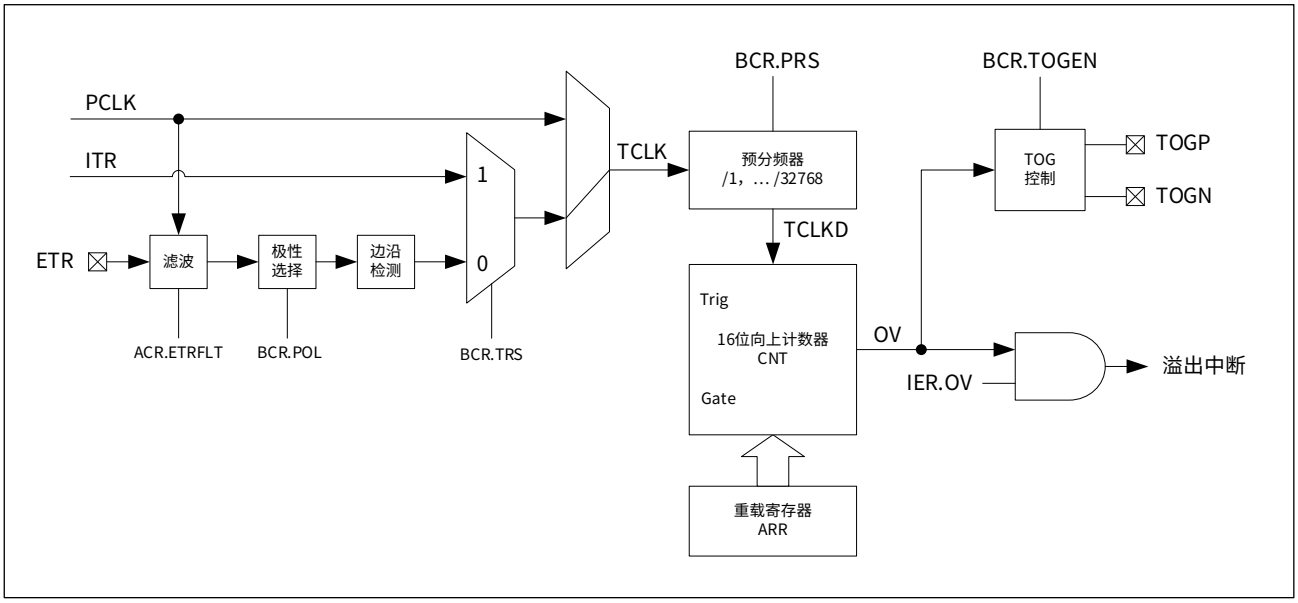
13.3.2.2 计数器模式

计数器模式主要用于测定某个事件发生的次数，可选择对内部 ITR 或外部 BTIM_ETR 引脚输入的信号进行计数，具体通过基本控制寄存器 BTIMx_BCR 的 TRS 位域来选择。

当设置 BTIMx_BCR.TRS 为 1 时，计数源为内部 ITR 信号（具体 ITR 来源请参见表 13-4 ITR 来源配置）；当设置 BTIMx_BCR.TRS 为 0 时，计数源为外部 BTIM_ETR 引脚输入的信号，选择该计数源时，可通过 BTIMx_ACR.ETRFLT 配置滤波器，通过 BTIMx_BCR.POL 选择对 ERT 信号的上升沿或下降沿进行计数。

计数器模式功能框图如下图所示：

图 13-7 计数器模式框图



设置 BTIMx_BCR.MODE 为 0x01，使 BTIMx 工作于计数器模式。设置 BTIMx_BCR.EN 为 1 使能 BTIMx，计数器 CNT 对分频后的 TCLKD 信号进行累加计数。当计数值到达重载值 ARR 后产生溢出，计数器 ARR 溢出标志位 BTIMx_ISR.OV 被硬件置位，如果允许中断 (设置中断使能寄存器 BTIMx_IER.OV 为 1)，CPU 将响应中断服务程序。退出中断服务程序之前，应设置中断标志清除寄存器 BTIMx_ICR.OV 为 0 以清除该标志。

计数器模式的具体寄存器配置流程请参见 13.5 编程示例。

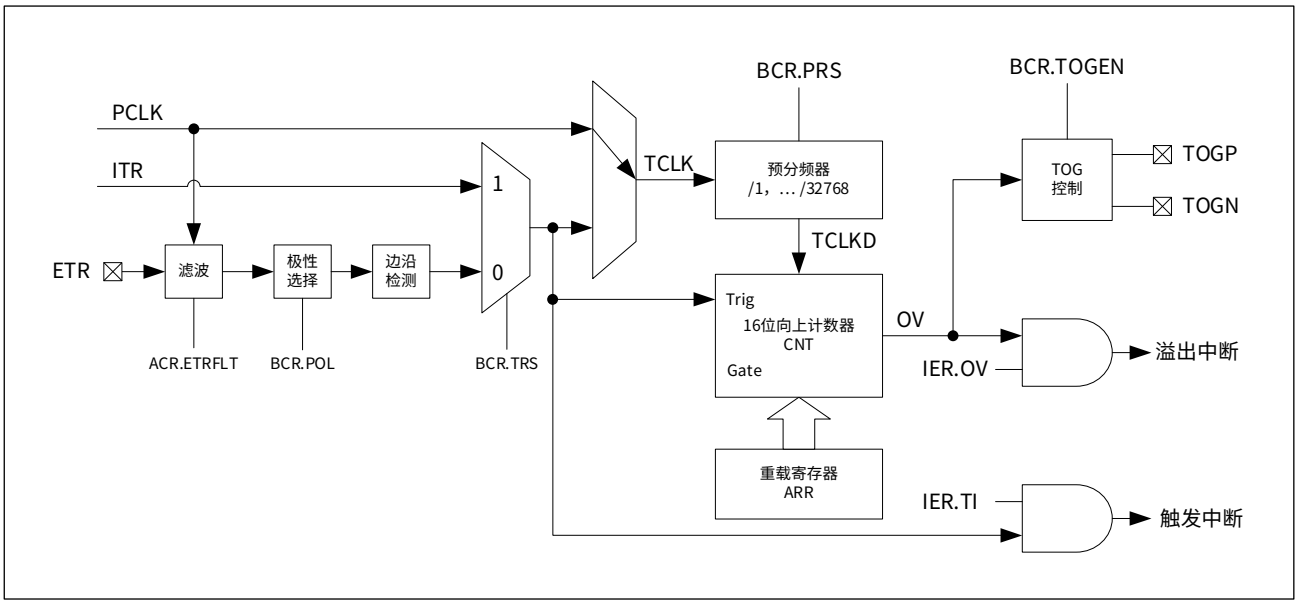
13.3.2.3 触发启动模式

设置 BTIMx_BCR.MODE 为 0x02，使 BTIMx 工作于触发启动模式。在该模式下，设置 BTIMx_BCR.EN 为 1 或触发信号有效时，将启动计数器 CNT 对内部时钟 PCLK 经预分频器分频后的 TCLKD 信号进行计数。

触发信号可选择内部 ITR 信号或外部 BTIM_ETR 引脚输入的信号，具体通过基本控制寄存器 BTIMx_BCR 的 TRS 位域来选择。当设置 BTIMx_BCR.TRS 为 1 时，触发源为内部 ITR 信号（具体 ITR 来源请参见表 13-4 ITR 来源配置）；当设置 BTIMx_BCR.TRS 为 0 时，触发源为外部 BTIM_ETR 引脚输入的信号，选择该触发源时，可通过 BTIMx_ACR.ETRFLT 配置滤波器，通过 BTIMx_BCR.POL 选择 ERT 信号是上升沿或是下降沿有效。

触发启动模式功能框图如下图所示：

图 13-8 触发启动模式框图



当检测到有效的触发信号时，将产生以下影响：

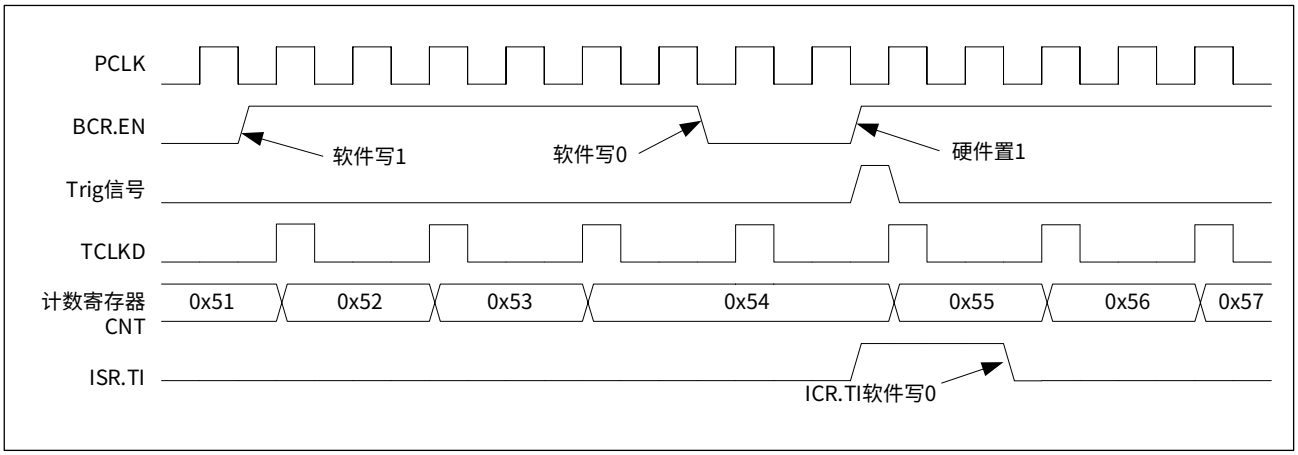
1. BTIMx_BCR.EN 被硬件置位；
2. 触发标志位 BTIMx_ISR.TI 被硬件置位；
3. 计数器启动，开始计数。

计数器启动后，计数器从初始值开始向上计数，当计数值到达重载值 ARR 后产生溢出，计数器 ARR 溢出标志位 BTIMx_ISR.OV 被硬件置位，如果允许中断 (设置中断使能寄存器 BTIMx_IER.OV 为 1)，CPU 将响应中断服务程序。退出中断服务程序之前，应设置中断标志清除寄存器 BTIMx_ICR.OV 为 0 以清除该标志。

在任意时候设置运行控制位 BTIMx_BCR.EN 为 0 后，计数器立即暂停计数。再次设置运行控制位 BTIMx_BCR.EN 为 1 或者触发信号有效时，计数器按前一次的设置继续计数。

触发启动模式时序图如下图所示：

图 13-9　触发启动模式时序图（分频比为 2，PRS=0x01）



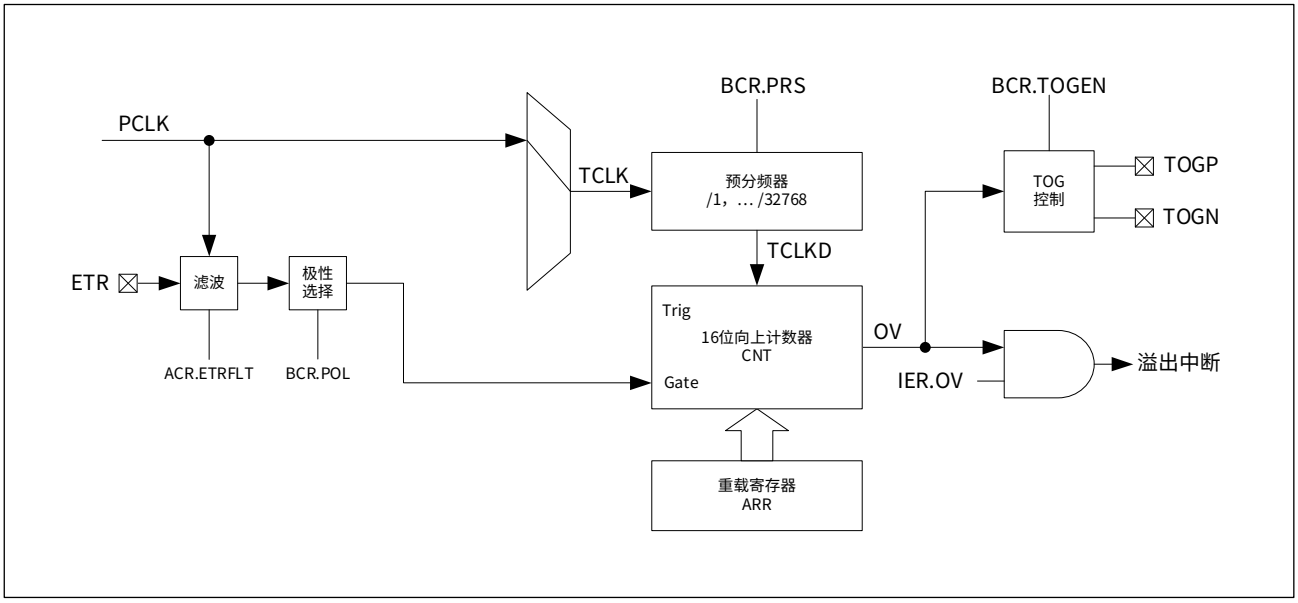
13.3.2.4　门控模式

设置 BTIMx_BCR.MODE 为 0x03，使 BTIMx 工作于门控模式。在该模式下，设置 BTIMx_BCR.EN 为 1 且门控信号有效时，将启动计数器 CNT 对内部时钟 PCLK 经预分频器分频后的 TCLKD 信号进行计数。

门控信号固定为外部 BTIM_ETR 引脚输入的信号，设置基本控制寄存器 BTIMx_BCR 的 POL 位域可选择门控信号是高电平或低电平为有效电平，还可通过高级控制寄存器 BTIMx_ACR 的 ETRFLT 位域配置是否滤波及滤波的带宽。

门控模式功能框图如下图所示：

图 13-10　门控模式框图

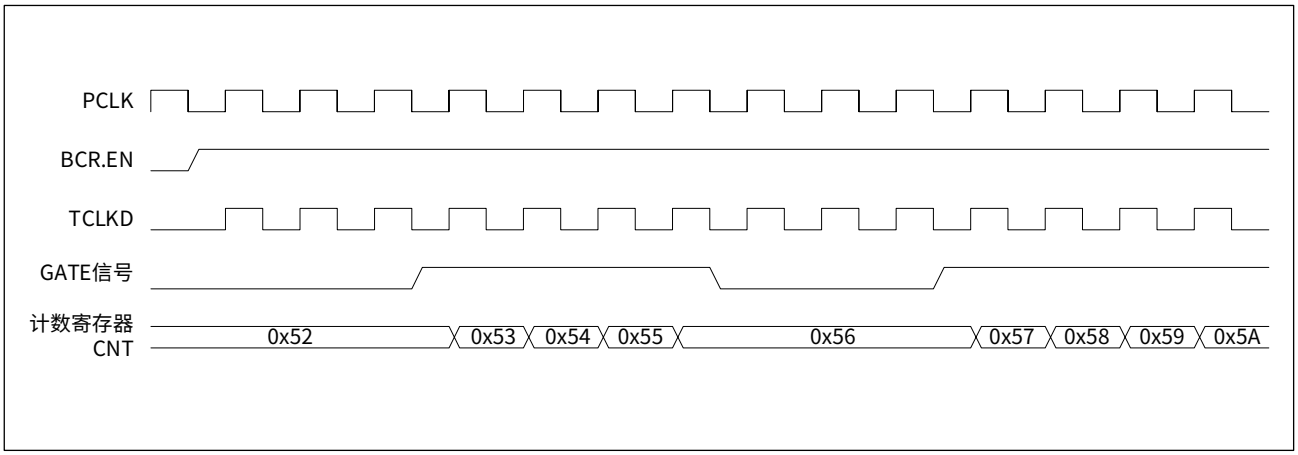


计数器启动后，计数器从初始值开始向上计数，当计数值到达重载值 ARR 后产生溢出，计数器 ARR 溢出标志位 BTIMx_ISR.OV 被硬件置位，如果允许中断 (设置中断使能寄存器 BTIMx_IER.OV 为 1)，CPU 将响应中断服务程序。退出中断服务程序之前，应设置中断标志清除寄存器 BTIMx_ICR.OV 为 0 以清除该标志。

在任意时候设置运行控制位 BTIMx_BCR.EN 为 0 或门控信号无效时，计数器立即暂停计数。再次设置运行控制位 BTIMx_BCR.EN 为 1 且门控信号有效时，计数器按前一次的设置继续计数。

门控模式时序图如下图所示：

图 13-11 门控模式时序图（PRS=0x00，POL=0x00）

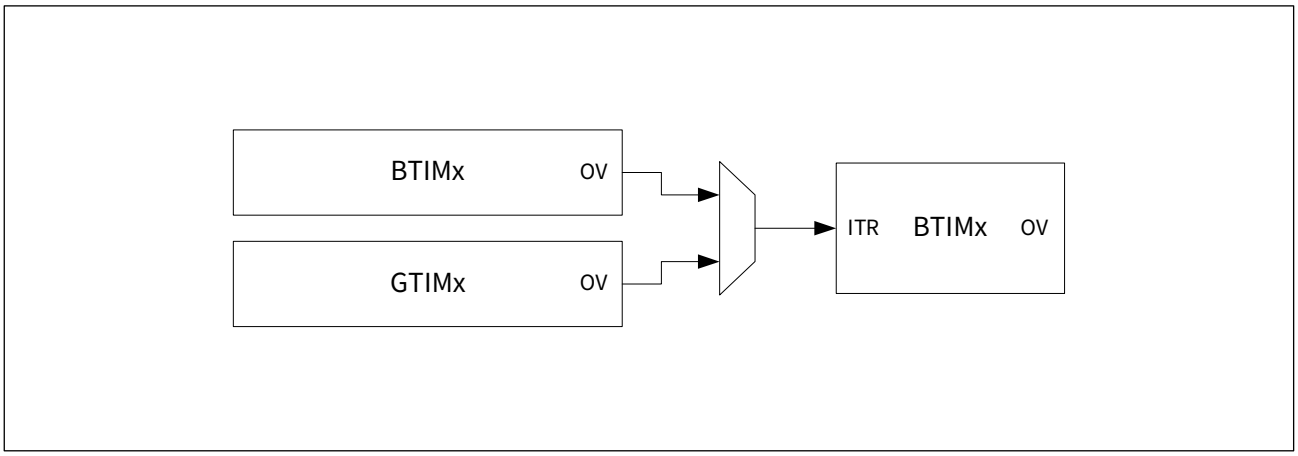


13.3.3 内部级联 ITR

定时器可以级联使用，从而增加定时器位宽，如两个 16 位的定时器级联可作为一个 32 位的定时器使用，也可以通过 ITR 级联实现对计数时钟源 PCLK 进行 1 ~ 65535 之间任意数分频。

BTIM 两级级联使用示意图如下图所示：

图 13-12 两级级联 ITR



级联使用时，第一级定时器的计数时钟源可以是 PLCK 或外部 ETR 引脚输入的信号，后级定时器的计数时钟源为前一级定时器的 ARR 溢出信号 OV。

定时器的 ITR 来源，可通过定时器 ITR 来源配置寄存器 SYSCTRL_TIMITR 来选择，具体配置如下表所示：

表 13-4 ITR 来源配置

SYSCTRL_TIMITR 位域	值	ITR 来源
BTIM3ITR BTIM2ITR BTIM1ITR GTIM4ITR GTIM3ITR GTIM2ITR GTIM1ITR	000	BTIM1 的溢出信号
	001	BTIM2 的溢出信号
	010	BTIM3 的溢出信号
	011	GTIM1 的溢出信号
	100	GTIM2 的溢出信号
	101	GTIM3 的溢出信号
	110	GTIM4 的溢出信号

注：

定时器的 ITR 来源不能选择自身的溢出信号 OV。

13.3.4 外部互联 ETR

BTIM 的 ETR 信号来源只能是外部 BTIM_ETR 引脚，且所有 BTIM 共用一个 ETR 信号，由 GPIO 复用功能寄存器 (CPIOx_AFRH 和 CPIOx_AFRL) 进行配置，参见下表：

表 13-5 外部 BTIM_ETR 引脚复用

ETR	引脚	AFR
BTIM_ETR	PA12	0x02
BTIM_ETR	PB11	0x05
BTIM_ETR	PC13	0x05

13.4 调试支持

BTIM 支持在调试模式下停止或继续计数，通过调试状态定时器控制寄存器 SYSCTRL_DEBUG 的 BTIM123 位域来设置。

设置 SYSCTRL_DEBUG.BTIM123 为 1，则在调试状态时暂停 BTIM1、BTIM2 和 BTIM3 的计数器计数；

设置 SYSCTRL_DEBUG.BTIM123 为 0，则在调试状态时 BTIM1、BTIM2 和 BTIM3 的计数器继续计数。

13.5 编程示例

13.5.1 定时器模式编程示例

步骤 1：设置 SYSCTRL_APBEN2.BTIM 为 1，打开 BTIM1、BTIM2 和 BTIM3 的配置时钟及工作时钟；

注：

BTIM1、BTIM2 和 BTIM3 共用 SYSCTRL_APBEN2.BTIM 位。

步骤 2：设置 BTIMx_BCR.MODE 为 0x00，使 BTIMx 工作于定时器模式；

步骤 3：配置 BTIMx_BCR.ONESHOT，选择单次或连续计数模式。配置为 0，则为连续计数模式，配置为 1，则为单次计数模式；

步骤 4：配置 BTIMx_BCR.PRS，选择预分频器的分频比；

步骤 5：配置 BTIMx_ARR，设置 BTIMx 计数溢出时间；

步骤 6：设置 BTIMx_CNT 为 0x0000，以便计数；

步骤 7：如果希望 ARR 溢出事件引发中断，设置 BTIMx_IER.OV 为 1；

步骤 8：设置 BTIMx_BCR.EN 为 1，启动 BTIMx；

步骤 9：如果中断产生，设置 BTIMx_ICR.OV 为 0 清除中断标志。

13.5.2 计数器模式编程示例

13.5.2.1 对外部 ETR 信号计数

对外部 BTIM_ETR 引脚输入的信号计数，步骤如下：

步骤 1：设置 SYSCTRL_AHBEN.GPIOx 为 1，使能 BTIM_ETR 引脚对应的 GPIO 端口的配置时钟及工作时钟；

步骤 2：设置 GPIO 复用功能寄存器 (GPIOx_AFRH 和 GPIOx_AFRL) 的相关位，将 ETR 的来源配置到所需的硬件引脚上；

步骤 3：设置 SYSCTRL_APBEN2.BTIM 位为 1，打开 BTIM1、BTIM2 和 BTIM3 的的配置时钟及工作时钟；

注：

BTIM1、BTIM2 和 BTIM3 共用 SYSCTRL_APBEN2.BTIM 位。

步骤 4：配置 BTIMx_BCR.MODE 为 0x01，使 BTIMx 工作于计数器模式；

步骤 5：配置 BTIMx_BCR.ONESHOT，选择单次或连续计数模式；配置为 0，则为连续计数模式，配置为 1，则为单次计数模式；

步骤 6：配置 BTIMx_BCR.TRS 为 0，选择计数源为外部 BTIM_ETR 引脚输入的信号；

步骤 7：配置 BTIMx_ACR.ETRFLT，选择是否滤波或滤波的带宽；

步骤 8：配置 BTIMx_BCR.POL，选择 ETR 信号计数边沿；配置为 0，则上升沿计数，配置为 1，则下降沿计数；

步骤 9：配置 BTIMx_BCR.PRS，选择预分频器的分频比；

步骤 10：配置 BTIMx_ARR，设置 BTIMx 计数溢出时间；

步骤 11：如果希望 ARR 溢出事件引发中断，设置 BTIMx_IER.OV 为 1；

步骤 12：设置 BTIMx_BCR.EN 为 1，启动 BTIMx；

步骤 13：如果中断产生，设置 BTIMx_ICR.OV 为 0 清除中断标志。

13.5.2.2 对内部 ITR 信号计数

步骤 1：设置 SYSCTRL_APBEN2.BTIM 位为 1，打开 BTIM1、BTIM2 和 BTIM3 的的配置时钟及工作时钟；

注：

BTIM1、BTIM2 和 BTIM3 共用 SYSCTRL_APBEN2.BTIM 位。

步骤 2：配置 BTIMx_BCR.MODE 为 0x01，选择 BTIMx 工作在计数器模式下；

步骤 3：配置 BTIMx_BCR.ONESHOT，选择单次或连续计数模式；配置为 0，则为连续计数模式，配置为 1，则为单次计数模式；

步骤 4：配置 BTIMx_BCR.TRS 为 1，选择计数信号来自内部 ITR 信号；

步骤 5：配置 SYSCTRL_TIMITR 寄存器，选择相应定时器的 ITR 来源；

步骤 6：配置 BTIMx_BCR.PRS，选择预分频器的分频比；

步骤 7：配置 BTIMx_ARR，选择 BTIMx 计数溢出时间；

步骤 8：如果希望 ARR 溢出事件引发中断，配置 BTIMx_IER.OV 为 1；

步骤 9：设置 BTIMx_BCR.EN 为 1，启动 BTIMx；

步骤 10：如果中断产生，设置 BTIMx_ICR.OV 为 0 清除中断标志。

13.5.3 触发启动模式编程示例

步骤 1：设置 SYSCTRL_APBEN2.BTIM 位为 1，打开 BTIM1、BTIM2 和 BTIM3 的的配置时钟及工作时钟；

注：

BTIM1、BTIM2 和 BTIM3 共用 SYSCTRL_APBEN2.BTIM 位。

步骤 2：配置 BTIMx_BCR.TRS，选择触发源；配置为 0，则触发信号来自外部 BTIM_ETR 引脚输入信号，配置为 1，则触发信号来自内部级联 ITR 信号；

步骤 3：如果触发信号来自 BTIM_ETR 引脚输入，可配置 BTIMx_ACR.ETRFLT，选择是否滤波或滤波的带宽；

步骤 4：如果触发信号来自 BTIM_ETR 引脚输入，配置 BTIMx_BCR.POL，选择 ETR 信号触发边沿。配置为 0，则上升沿计数启动，配置为 1，则下降沿计数启动；

步骤 5：配置 BTIMx_BCR.PRS，选择预分频器的分频比；

步骤 6：配置 BTIMx_BCR.ONESHOT，选择单次或连续计数模式；配置为 0，则为连续计数模式，配置为 1，则为单次计数模式；

步骤 7：如果希望触发事件引发中断，设置 BTIMx_IER.IT 为 1；

步骤 8：如果希望 ARR 溢出事件引发中断，设置 BTIMx_IER.OV 为 1；

步骤 9：配置 BTIMx_ARR，选择 BTIMx 计数溢出时间；

步骤 10：配置 BTIMx_BCR.MODE 为 0x10，选择 BTIMx 工作在触发启动模式下；

步骤 11：根据需要决定是否写 1 到 BTIMx_BCR.EN，以立即启动 BTIMx 进行计数；

步骤 12：如果中断产生，需要根据中断产生的事件对 BTIMx_ICR 寄存器的相应位写 0 清除中断标志。

注：

应先设置触发信号的极性，再设置定时器的模式，否则会引起误触发。

13.5.4 门控模式编程示例

以外部 BTIM_ETR 引脚输入的信号作为门控信号，控制计数器计数，操作步骤如下：

步骤 1：设置 SYSCTRL_AHBEN.GPIOx 为 1，使能 BTIM_ETR 引脚对应的 GPIO 端口的配置时钟及工作时钟；

步骤 2：设置 GPIO 复用功能寄存器 (GPIOx_AFRH 和 GPIOx_AFRL) 的相关位，将 ETR 的来源配置到所需的硬件引脚上；

步骤 3：设置相应 GPIO 引脚的 GPIOx_ANALOG 寄存器的对应位为 0，配置为数字功能；

步骤 4：设置相应 GPIO 引脚的 GPIOx_DIR 寄存器的对应位为 1，将端口配置成输入；

步骤 5：设置 SYSCTRL_APBEN2.BTIM 位为 1，打开 BTIM1、BTIM2 和 BTIM3 的的配置时钟及工作时钟；

注：

BTIM1、BTIM2 和 BTIM3 共用 SYSCTRL_APBEN2.BTIM 位。

步骤 6：配置 BTIMx_BCR.MODE 为 0x03，选择 BTIMx 工作在门控模式下；

步骤 7：配置 BTIMx_ACR.ETRFLT，选择是否滤波或滤波的带宽；

步骤 8：配置 BTIMx_BCR.POL，选择 ETR 信号极性；配置为 0，门控模式高电平有效，配置为 1，门控模式低电平有效；

步骤 9：配置 BTIMx_BCR.PRS，选择预分频器的分频比；

步骤 10：配置 BTIMx_BCR.ONESHOT，选择单次或连续计数模式；配置为 0，为连续计数模式，配置为 1，为单次计数模式；

步骤 11：如果希望 ARR 溢出事件引发中断，配置 BTIMx_IER.OV 为 1；

步骤 12：配置 BTIMx_ARR 寄存器，选择 BTIMx 计数溢出时间；

步骤 13：设置 BTIMx_BCR.EN 为 1，启动 BTIMx；

步骤 14：如果中断产生，设置 BTIMx_ICR.OV 为 0 清除中断标志。

13.6 寄存器列表

BTIM1 基地址：BTIM1_BASE = 0x4001 4800

BTIM2 基地址：BTIM2_BASE = 0x4001 4900

BTIM3 基地址：BTIM3_BASE = 0x4001 4A00

表 13-6 BTIM 寄存器列表

寄存器名称	寄存器地址	寄存器描述
BTIMx_ARR	BTIMx_BASE + 0x00	重载寄存器
BTIMx_CNT	BTIMx_BASE + 0x04	计数寄存器
BTIMx_ACR	BTIMx_BASE + 0x0C	高级控制寄存器
BTIMx_BCR	BTIMx_BASE + 0x10	基本控制寄存器
BTIMx_IER	BTIMx_BASE + 0x14	中断使能寄存器
BTIMx_ISR	BTIMx_BASE + 0x18	中断标志寄存器
BTIMx_ICR	BTIMx_BASE + 0x1C	中断标志清除寄存器
BTIMx_DMA	BTIMx_BASE + 0x20	DMA 触发寄存器

13.7 寄存器描述

有关寄存器描述里所使用的缩写，请参见 1 文档约定章节。

13.7.1 BTIMx_BCR 基本控制寄存器

Address offset: 0x10 Reset value: 0x0000 0000

位域	名称	权限	功能描述
31:15	RFU	-	保留位，请保持默认值
14:11	PRSSTATUS	RO	预分频器当前正在使用的分频系数 定时器溢出或 EN 从 0 变成 1 时会从 PRS 更新该值
10:7	PRS	RW	预分频器分频系数配置 0：DIV1 4：DIV16 8：DIV256 12：DIV4096 1：DIV2 5：DIV32 9：DIV512 13：DIV8192 2：DIV4 6：DIV64 10：DIV1024 14：DIV16384 3：DIV8 7：DIV128 11：DIV2048 15：DIV32768
6	TOGEN	RW	TOG 引脚输出使能控制 0：TOGP、TOGN 输出电平均为 0 1：TOGP、TOGN 输出电平相反的信号
5	ONESHOT	RW	单次 / 连续计数模式控制 0：连续计数模式 1：单次计数模式
4	POL	RW	外部引脚输入的 ETR 信号极性选择 0：ETR 正向 (触发模式上升沿有效，门控模式高电平有效) 1：ETR 反向 (触发模式下降沿有效，门控模式低电平有效)
3	TRS	RW	触发源选择 0：ETR 引脚输入的信号 1：ITR，详见 13.3.3 内部级联 ITR
2:1	MODE	RW	基本定时模式配置 00：定时器模式，计数时钟源为 PCLK 01：计数器模式，计数时钟源为 TRS 信号 10：触发启动模式，计数时钟源为 PCLK，TRS 信号触发计数器启动 11：门控模式，计数时钟源为 PCLK，ETR 引脚输入信号作为门控
0	EN	RW	定时器运行控制 0：定时器停止 1：定时器运行

13.7.2 BTIMx_ACR 高级控制寄存器

Address offset: 0x0C Reset value: 0x0000 0000

位域	名称	权限	功能描述
31:7	RFU	-	保留位，请保持默认值
6:4	ETRFLT	RW	ETR 引脚输入信号滤波配置；采样时钟为 PCLK 或 PCLK 分频，采样点个数为 N，连续 N 个电平有效。 000：无滤波 001：F_{sample} = PCLK， N=2 010：F_{sample} = PCLK， N=4 011：F_{sample} = PCLK， N=6 100：F_{sample} = PCLK/4， N=4 101：F_{sample} = PCLK/4， N=6 110：F_{sample} = PCLK/16， N=4 111：F_{sample} = PCLK/16， N=6
3:0	RFU	-	保留位，请保持默认值

13.7.3 BTIMx_ARR 重载寄存器

Address offset: 0x00 Reset value: 0x0000 FFFF

位域	名称	权限	功能描述
31:16	RFU	-	保留位，请保持默认值
15:0	ARR	RW	定时器重载值

13.7.4 BTIMx_CNT 计数寄存器

Address offset: 0x04 Reset value: 0x0000 0000

位域	名称	权限	功能描述
31:16	RFU	-	保留位，请保持默认值
15:0	CNT	RW	定时器计数值

13.7.5 BTIMx_IER 中断使能寄存器

Address offset: 0x14 Reset value: 0x0000 0000

位域	名称	权限	功能描述
31:3	RFU	-	保留位，请保持默认值
2	TOP	RW	计数器 TOP 溢出中断使能控制 0：禁止 TOP 溢出中断 1：使能 TOP 溢出中断
1	TI	RW	触发中断使能控制 0：禁止触发中断 1：使能触发中断
0	OV	RW	计数器 ARR 溢出中断使能控制 0：禁止 ARR 溢出中断 1：使能 ARR 溢出中断

13.7.6 BTIMx_ISR 中断标志寄存器

Address offset: 0x18 Reset value: 0x0000 0000

位域	名称	权限	功能描述
31:3	RFU	-	保留位，请保持默认值
2	TOP	RO	计数器 TOP 溢出标志 0：计数器未 TOP 溢出 1：计数器已 TOP 溢出
1	TI	RO	触发标志 0：未发生触发事件 1：已发生触发事件
0	OV	RO	计数器 ARR 溢出标志 0：计数器未 ARR 溢出 1：计数器已 ARR 溢出

13.7.7 BTIMx_ICR 中断标志清除寄存器

Address offset: 0x1C Reset value: 0x0000 0007

位域	名称	权限	功能描述
31:3	RFU	-	保留位，请保持默认值
2	TOP	R1W0	计数器 TOP 溢出标志清除 W0：清除计数器 TOP 溢出标志 W1：无功能
1	TI	R1W0	触发标志清除 W0：清除触发标志 W1：无功能
0	OV	R1W0	计数器 ARR 溢出标志清除 W0：清除计数器 ARR 溢出标志 W1：无功能

13.7.8 BTIMx_DMA DMA 触发寄存器

Address offset: 0x20 Reset value: 0x0000 0000

位域	名称	权限	功能描述
31:2	RFU	-	保留位，请保持默认值
1	TRS	RW	计数器被触发启动时同步触发 DMA 使能控制 0：无功能 1：计数器被触发启动时同步触发 DMA
0	OV	RW	计数器 ARR 溢出触发 DMA 使能控制 0：无功能 1：计数器 ARR 溢出触发 DMA

14 通用定时器（GTIM）

14.1 概述

CW32F020 内部集成 4 个通用定时器 (GTIM)，每个 GTIM 完全独立且功能完全相同，各包含一个 16bit 自动重装载计数器并由一个可编程预分频器驱动。GTIM 支持定时器模式、计数器模式、触发启动模式和门控模式 4 种基本工作模式，每组带 4 路独立的捕获 / 比较通道，可以测量输入信号的脉冲宽度（输入捕获）或者产生输出波形（输出比较和 PWM）。

14.2 主要特性

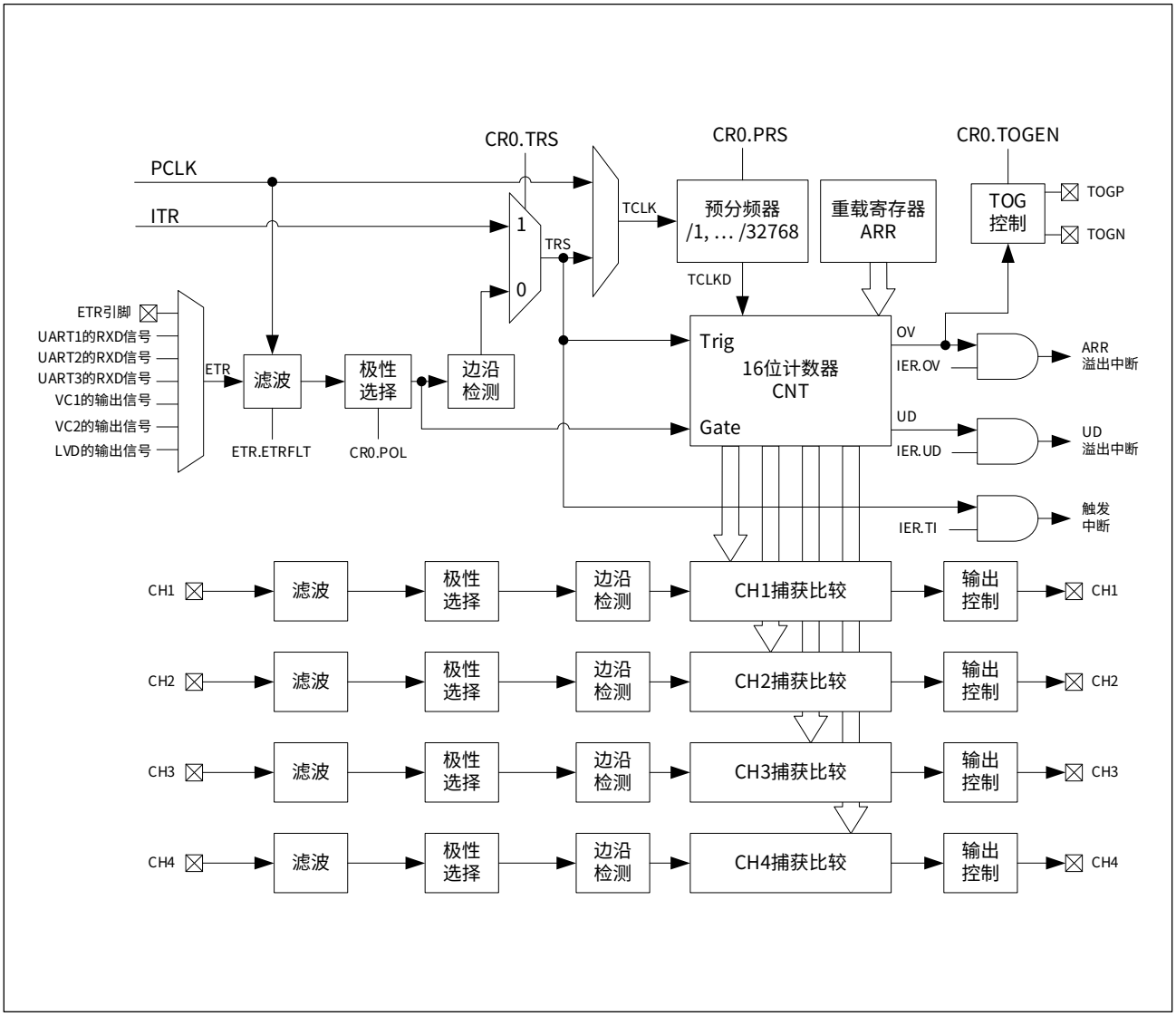
- 16bit 自动重载计数器
- 可编程预分频器支持 1，2，4，8，…，32768 分频
- 支持单次计数模式和连续计数模式
- 对内部 ITR 或 ETR 输入信号的计数功能
- 内部 ITR 或 ETR 输入信号触发启动计数
- 由 ETR 输入信号控制的门控功能
- 4 路独立输入捕获和输出比较通道
- 支持针对定位的增量 (正交) 编码器
- 溢出事件触发数字输出电平翻转
- 内部级联 ITR 和片内外设互联 ETR
- 多种事件发生时产生中断 /DMA
  - 计数器计数溢出
  - 计数器被触发启动
  - 输入捕获
  - 输出比较

14.3 功能描述

14.3.1 功能框图

GTIM 的功能框图如下图所示：

图 14-1 GTIM 功能框图



14.3.1.1 预分频器

预分频器以 2 的倍数为系数对计数器时钟源 TCLK 分频，最大分频系数为 32768。预分频系数由控制寄存器 GTIMx_CR0 的 PRS 位域配置，分频系数配置如下表所示：

表 14-1 预分频器分频系数配置表

PRS	分频比	PRS	分频比	PRS	分频比	PRS	分频比
0x00	1	0x04	16	0x08	256	0x0C	4096
0x01	2	0x05	32	0x09	512	0x0D	8192
0x02	4	0x06	64	0x0A	1024	0x0E	16384
0x03	8	0x07	128	0x0B	2048	0x0F	32768

在定时器运行过程中允许修改 GTIMx_CR0.PRS，但新的预分频系数不会立即生效，当定时器发生计数溢出或者运行控制位 GTIMx_CR0.EN 由 0 变为 1 时，新的预分频系数生效。

14.3.1.2 计数单元

计数单元的核心组件是一个 16bit 的计数器 CNT 和一个 16bit 自动重载寄存器 ARR。

在定时器运行过程中允许修改重载寄存器 ARR，且 ARR 的值将立即生效。计数器 CNT 的正确计数范围为 0 到重载值 ARR 之间，因此设置重载寄存器 ARR 时，应保证 CNT 的值小于 ARR 寄存器的设置值。

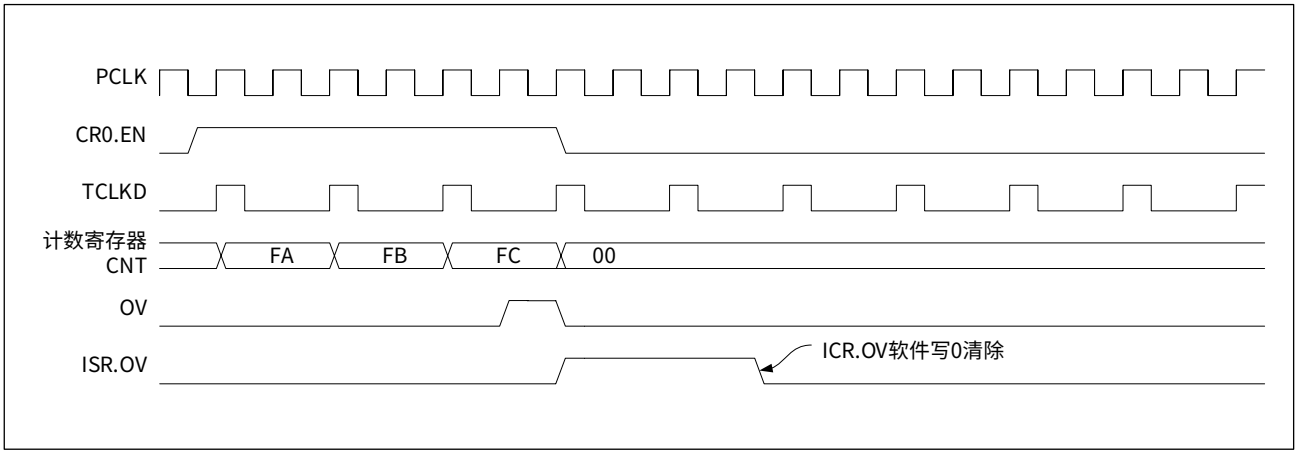
计数器可工作在单次计数或连续计数模式下，通过控制寄存器 GTIMx_CR0 的 ONESHOT 位域来选择。

单次计数模式

设置 GTIMx_CR0.ONESHOT 为 1，使定时器工作在单次计数模式下。设置 GTIMx_CR0.EN 为 1 使能 GTIMx，计数器 CNT 在 TCLKD 时钟的驱动下累加计数。当计数值到达重载值 ARR 后产生溢出信号 OV（溢出信号 OV 保持一个 PCLK 周期，然后自动清除），同时计数器停止计数，GTIMx_CR0.EN 被硬件自动复位。

下图是单次计数模式示例：

图 14-2 单次计数模式（PRS=0x01，ARR=0xFC）

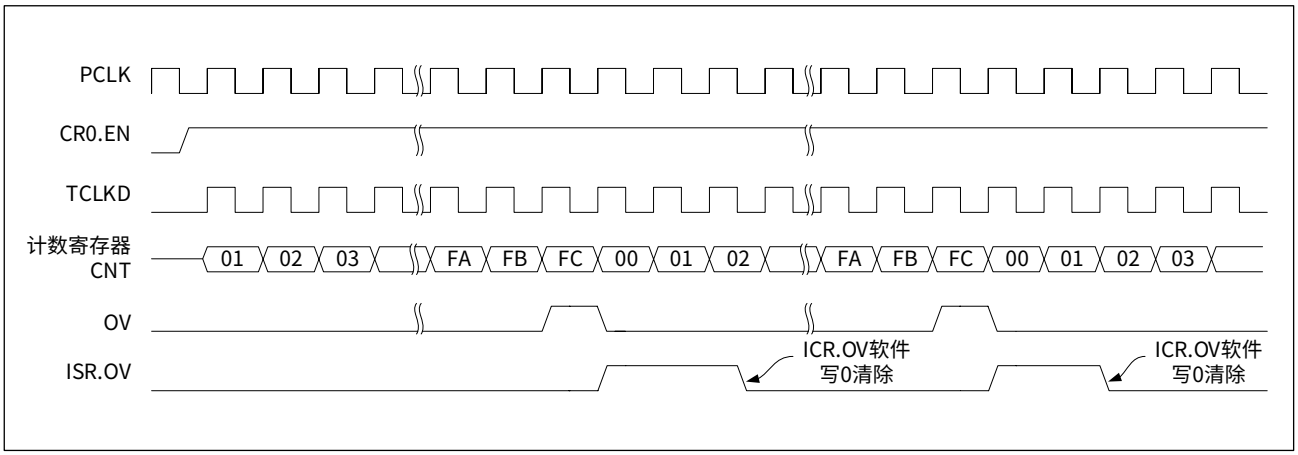


连续计数模式

设置 GTIMx_CR0.ONESHOT 为 0，使定时器工作在连续计数模式下。设置 GTIMx_CR0.EN 为 1 使能 GTIMx，计数器 CNT 在 TCLKD 时钟的驱动下累加计数。当计数值到达重载值 ARR 后产生溢出信号 OV（溢出信号 OV 保持一个 PCLK 周期，然后自动清除）。当计数值从 ARR 变为 0 时，计数器 ARR 溢出标志位 GTIMx_ISR.OV 被硬件置位，计数器开始下一个周期的累加计数，直到设置 GTIMx_CR0.EN 为 0 后才停止计数。

下图是连续计数模式示例：

图 14-3　连续计数模式 (PRS=0x00，ARR=0xFC)



编码计数模式

当寄存器位 GTIMx_CR0.ENCMODE 不为 0x00 时，计数器工作在编码计数模式，此时，计数器的计数方向由硬件指定，可以向上计数，也可以向下计数。计数器 CNT 根据 GTIMx_ISR.DIR 自动加 1 或减 1。每当加计数到 ARR 时，计数器将产生上溢出信号 OV，并再次自动从 0 开始加计数；每当减计数到 0 时，计数器将产生下溢出信号 UD，并从 0xFFFF 开始减计数。

14.3.1.3 输入控制单元

ETR 输入信号和 4 个比较输入信号 CH1 ~ CH4 都具有各自的输入控制单元。输入控制单元由滤波器、极性选择和边沿检测组成。

滤波器

滤波器采用数字滤波方式，以一定频率对输入信号进行采样，当连续采样到 N 个相同电平时信号有效，否则信号无效，以此滤除高频杂波信号。

注：

由于硬件是以 PCLK 时钟或 PCLK 时钟的分频对输入信号进行采样的，因此频率高于实际采样频率二分之一的输入信号存在被漏采样的可能。

ETR 输入信号的滤波器由外部触发控制寄存器 GTIMx_ETR.ETRFLT 配置采样时钟频率和采样点的个数。比较输入 CHy 的滤波器由控制寄存器 GTIMx_CR1.CHyFLT 配置采样时钟频率和采样点的个数（y = 1，2，3，4）。

极性选择

极性选择用于选择是否对输入信号进行反相。ETR 输入信号通过控制寄存器 GTIMx_CR0.POL 选择是否反相，应用于门控模式。比较输入 CHy 的输入信号通过控制寄存器 GTIMx_CR1.CHyPOL 选择是否反相。

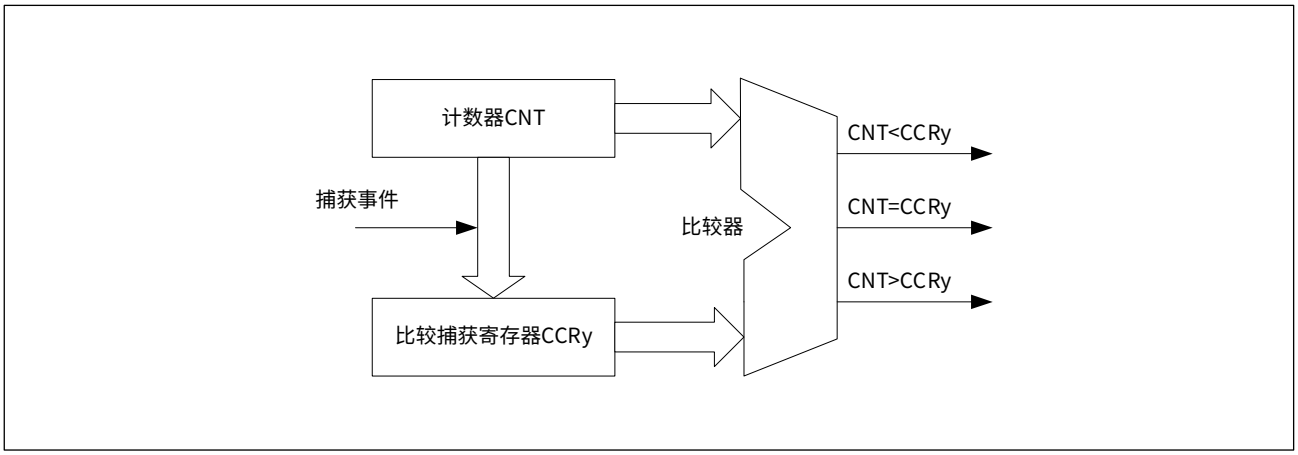
边沿检测

边沿检测用于检测输入信号的跳变以及跳变的方向是上升还是下降。

14.3.1.4 捕获比较通道

捕获比较通道由一个比较捕获寄存器和一个比较器组成，如下图所示：

图 14-4 捕获比较通道示意图



当进行输入捕获时，一旦相应的通道有捕获事件发生，计数器 CNT 的当前值立即传入比较捕获寄存器 CCRy 中保存，如果对应的捕获比较中断使能，将产生一个中断请求。

当进行输出比较时，计数器 CNT 的值一直和比较捕获寄存器 CCRy 的值比较，当比较结果为设定要求时，可以产生信号控制输出发生变化。如果相应的捕获比较中断使能，将产生一个中断请求。

14.3.1.5 输出控制单元

输出控制单元用于比较匹配时，控制输出端口的波形。通过比较捕获控制寄存器 GTIMx_CMMR 的 CCyM 位域配置对应通道的比较输出模式，具体配置及应用请参见 14.3.4 输出比较功能。

14.3.1.6 翻转输出单元

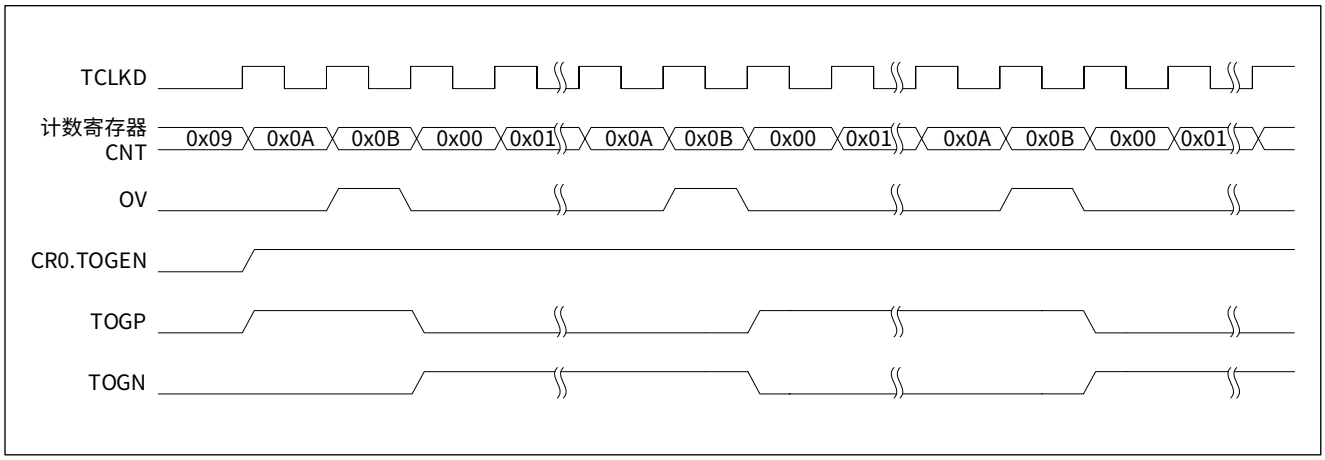
翻转输出单元可通过 ARR 溢出信号 OV 控制外部 GTIMx_TOGP 和 GTIMx_TOPN 引脚输出翻转信号。

当设置 GTIMx_CR0.TOGEN 为 0 时，GTIMx_TOGP 和 GTIMx_TOPN 引脚均输出低电平。

当设置 GTIMx_CR0.TOGEN 为 1 时，GTIMx_TOGP 和 GTIMx_TOPN 引脚输出电平相反的信号（GTIMx_TOGP 默认电平为高电平）；当计数器 ARR 溢出时 (GTIMx_ISR.OV 为 1)，GTIMx_TOGP 和 GTIMx_TOPN 引脚输出电平将翻转。

下图所示为连续计数模式下，GTIMx_TOGP 和 GTIMx_TOPN 引脚电平翻转输出示意图：

图 14-5 电平翻转输出示意图（连续计数模式，ARR=0x0B）



GTIM 支持的 TOGP/TOGN 引脚如下表所示：

表 14-2 GTIM 翻转输出引脚

TOGP/TOGN	引脚	AFR
GTIM1_TOGP	PB07/PB13	0x06
GTIM1_TOGN	PB06/PB12	0x06
GTIM2_TOGP	PB13	0x01
GTIM2_TOGP	PF01	0x06
GTIM2_TOGN	PB12	0x01
GTIM2_TOGN	PF00	0x06
GTIM3_TOGP	PC15/PF07	0x06
GTIM3_TOGN	PC14/PF06	0x06
GTIM4_TOGP	PB02	0x04
GTIM4_TOGP	PF07	0x03
GTIM4_TOGN	PB01	0x04
GTIM4_TOGN	PF06	0x03

14.3.2 基本工作模式

GTIM 支持 4 种基本工作模式：定时器模式、计数器模式、触发启动模式和门控模式。通过控制寄存器 0(GTIMx_CR0)的 MODE 位域来配置。

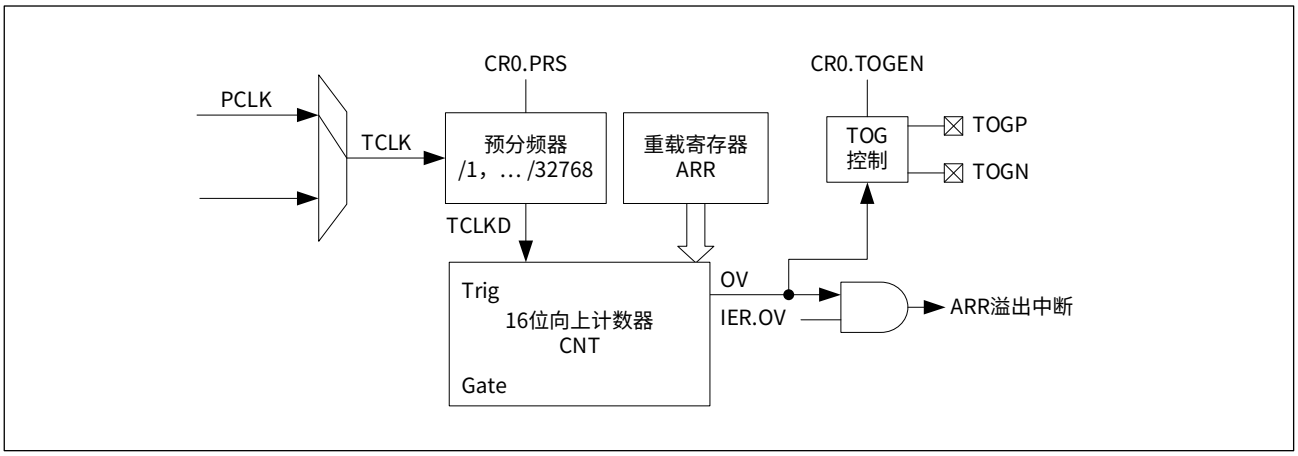
表 14-3 GTIM 基本工作模式

通用定时器	MODE 位域值	基本工作模式	描述
GTIMx_CR0 (x=1,2,3,4)	00	定时器模式	时钟源为 PCLK
	01	计数器模式	计数源为 TRS 信号
	10	触发启动模式	时钟源为 PCLK，TRS 信号触发计数器启动
	11	门控模式	时钟源为 PCLK，ETR 输入信号作为门控信号

14.3.2.1 定时器模式

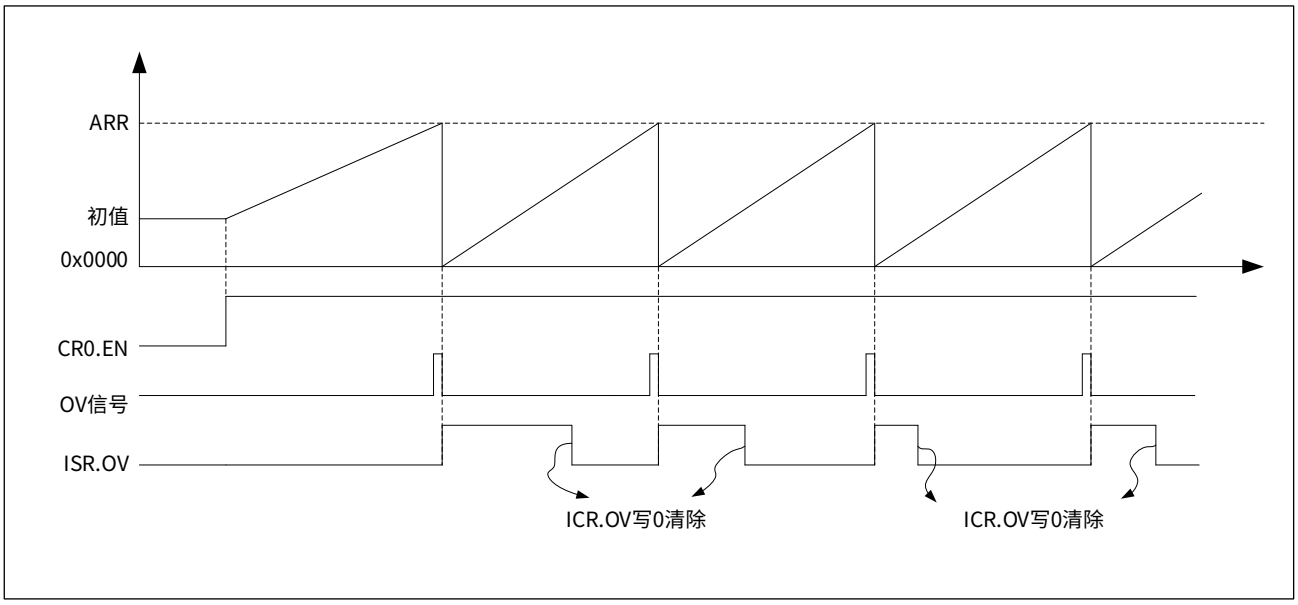
设置控制寄存器 GTIMx_CR0 的 MODE 位域为 0x00，使 GTIMx 工作于定时器模式。在该模式下，计数器时钟源为内部系统时钟 PCLK，经预分频器分频后得到计数时钟 TCLKD，来驱动计数器 CNT 计数。定时器模式功能框图如下图所示：

图 14-6 定时器模式框图



在实际应用中，系统时钟 PCLK 的频率是已知的，通过合理设置预分频器系数 PRS 可以对固定时长的时钟进行计数，配合对重载值 ARR 的设置及计数器 ARR 溢出中断标志位的使用，可以精确获得某一特定时长，从而达到定时的目的。下图是定时器在连续计数模式下的定时波形：

图 14-7　连续计数，定时器模式波形



定时时间 T 计算公式：

$$T = (2^{PRS} / PCLK) \times (ARR + 1)$$

其中，PCLK 为计数器时钟源，PRS 为预分频系数，ARR 为重载值。

例：

当计数器时钟源 PCLK 的频率为 24MHz 时，要求定时 10ms。

如果设置预分频系数 PRS 为 0x08，

计算

$$T = 10ms = (2^{8} / 24MHz) \times (ARR + 1)$$

则

$$ARR = 936.5$$

即可以设置重载值 ARR 为 937(0x3A9)。

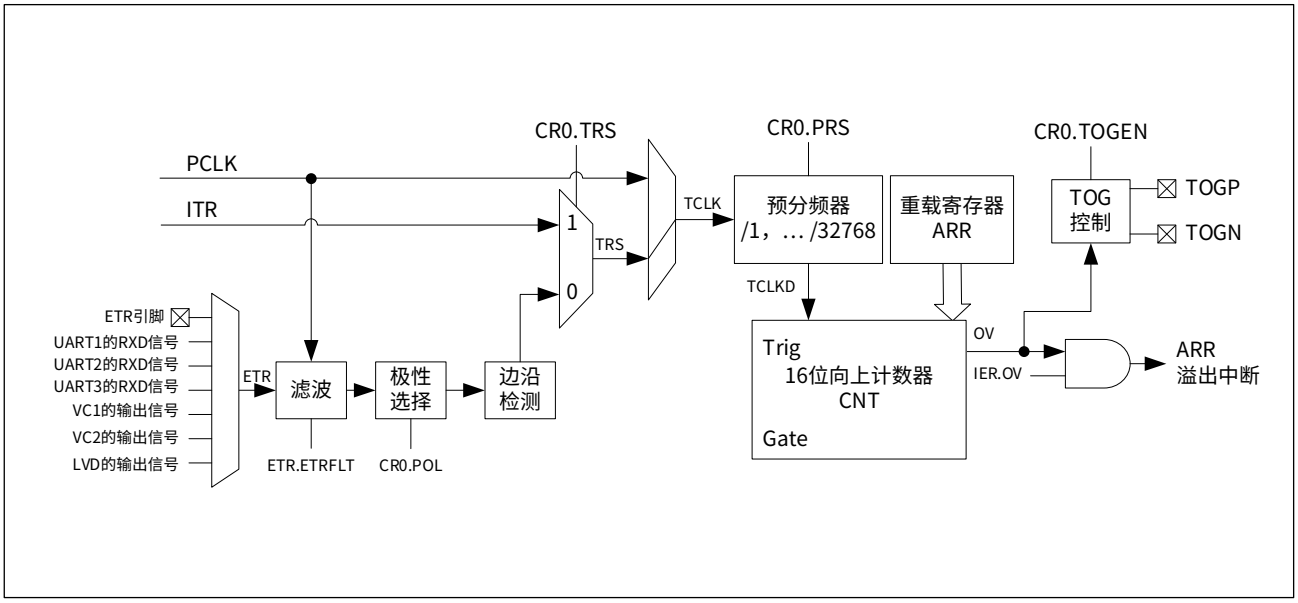
14.3.2.2 计数器模式

设置控制寄存器 GTIMx_CR0 的 MODE 位域为 0x01，使 GTIMx 工作于计数器模式。在该模式下，计数器时钟源为 TRS 信号，通过控制寄存器 GTIMx_CR0 的 TRS 位域可以选择 TRS 的来源为内部 ITR 信号或者 ETR 输入信号。

当设置控制寄存器 GTIMx_CR0 的 TRS 位域为 0 时，TRS 信号的来源为 ETR 输入信号，通过设置控制寄存器 GTIMx_CR0 的 POL 位域，可以选择在 ETR 输入信号的上升沿或下降沿进行计数；当设置控制寄存器 GTIMx_CR0 的 TRS 位域为 1 时，TRS 信号的来源为内部 ITR 信号，ITR 信号为其他定时器的输出信号（具体 ITR 来源参见表 14-8 ITR 来源配置）。

计数器模式功能框图如下图所示：

图 14-8 计数器模式功能框图



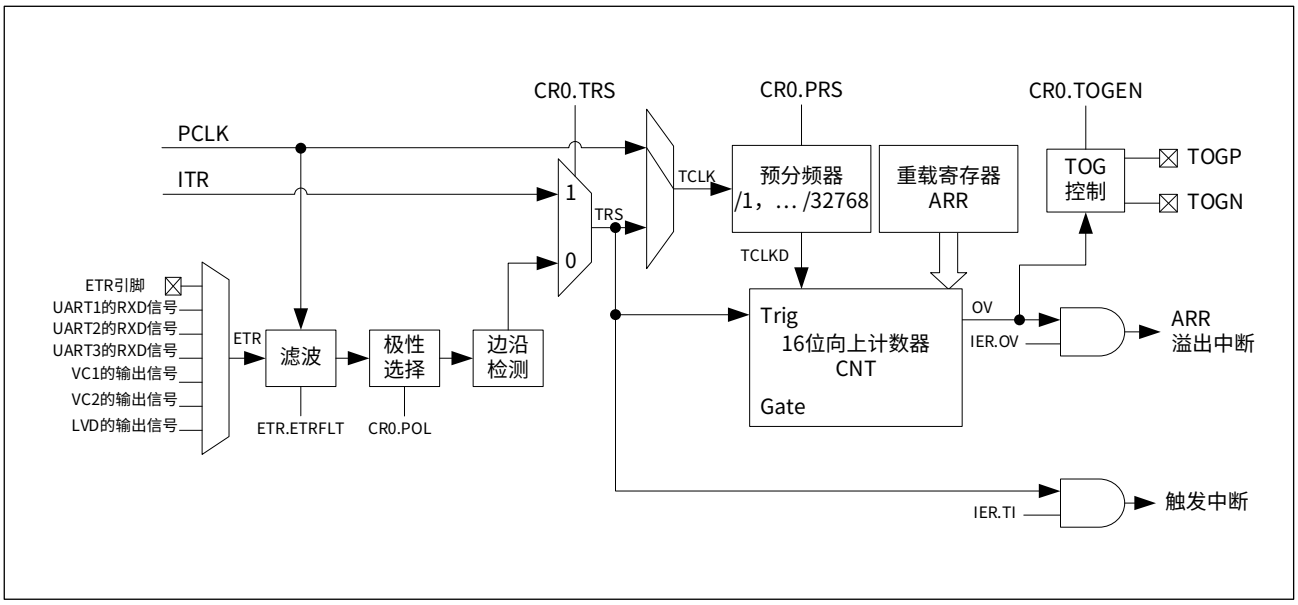
当设置控制寄存器 GTIMx_CR0 的 EN 位域为 1 后，计数器对分频后的 TCLKD 信号累加计数，计数到 ARR 值后产生溢出，并重新从 0 开始计数。计数器模式的具体寄存器配置流程请参见 14.6 编程示例。

14.3.2.3 触发启动模式

设置控制寄存器 GTIMx_CR0 的 MODE 位域为 0x02，使 GTIMx 工作于触发启动模式。在该模式下，计数器的计数时钟是系统时钟 PCLK 经预分频器分频后的信号 TCLKD，设置 GTIMx_CR0.EN 为 1 或触发信号有效时，将启动计数器计数。

触发信号 TRS 的来源可以是内部 ITR 信号或 ETR 输入信号，具体通过控制寄存器 GTIMx_CR0 的 TRS 位域来选择。当设置控制寄存器 GTIMx_CR0 的 TRS 位域为 0 时，TRS 信号的来源为 ETR 输入信号，通过设置控制寄存器 GTIMx_CR0 的 POL 位域，可以选择 ETR 输入信号是上升沿或下降沿有效；当设置控制寄存器 GTIMx_CR0 的 TRS 位域为 1 时，TRS 信号的来源为内部 ITR 信号，ITR 信号为其他定时器的输出信号（具体 ITR 来源参见表 14-8 ITR 来源配置）。触发启动模式功能框图如下图所示：

图 14-9 触发启动模式框图

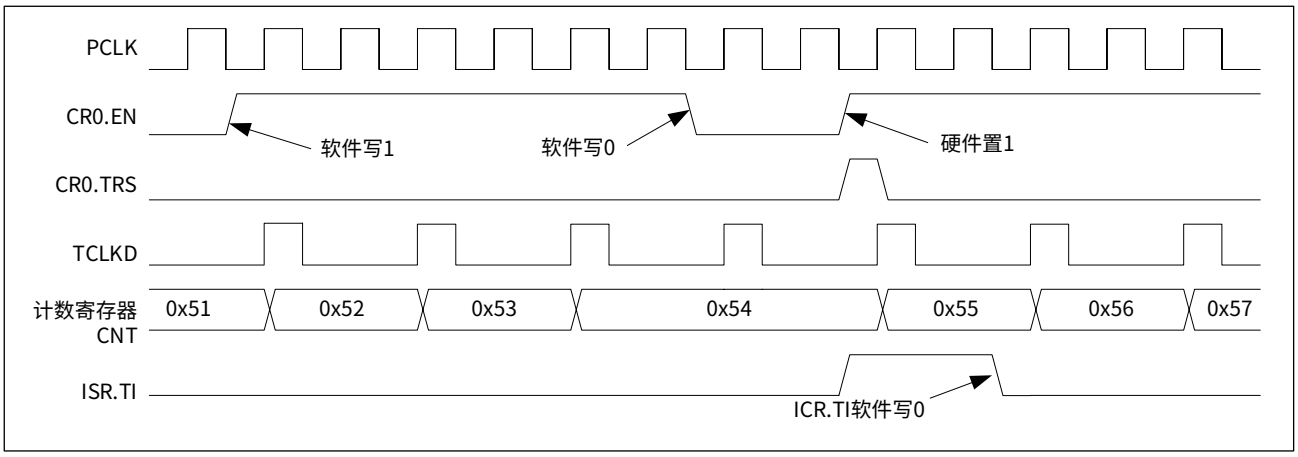


当检测到有效的触发信号时，将产生以下影响：

1. GTIMx_CR0.EN 被硬件置位；
2. 触发标志位 GTIMx_ISR.TI 被硬件置位；
3. 计数器启动，开始计数。

计数器启动后，计数器从初始值开始向上计数，当软件设置 GTIMx_CR0.EN 为 0 时，计数器暂停计数，再次设置 GTIMx_CR0.EN 为 1 或触发信号 TRS 为一个有效跳变时，计数器恢复计数并产生一个触发标志。下图所示为触发启动模式时序图示例：

图 14-10 触发启动模式时序图（分频比为 2，PRS=0x01）



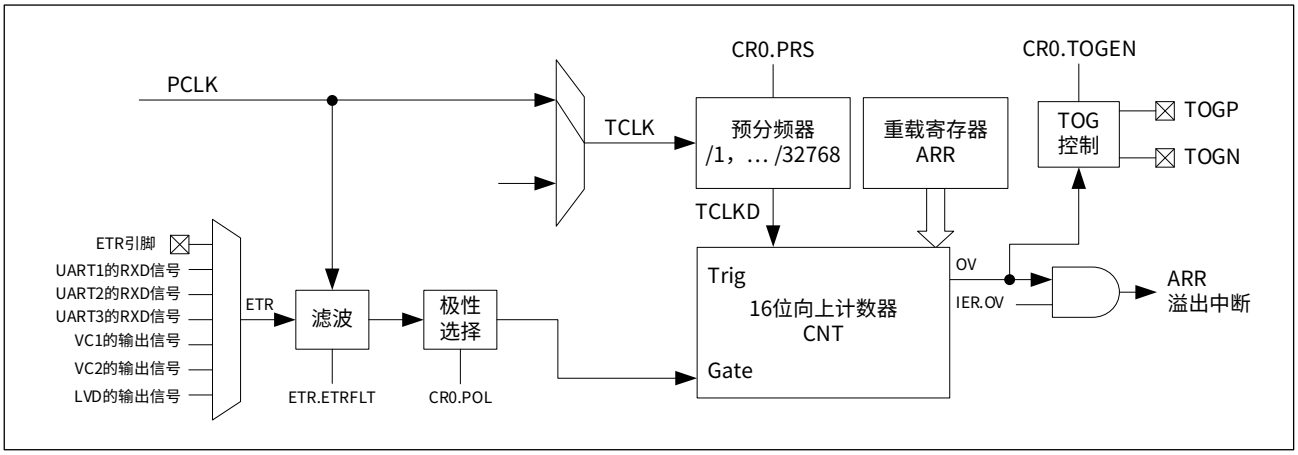
14.3.2.4 门控模式

设置控制寄存器 GTIMx_CR0 的 MODE 位域为 0x03，使 GTIMx 工作于门控模式。在该模式下，计数器的计数时钟是系统时钟 PCLK 经预分频器分频后的信号 TCLKD，设置 GTIMx_CR0.EN 为 1 且门控信号有效时，将启动计数器计数。

门控信号的来源为 ETR 输入信号，ETR 信号来源可以是外部 GTIMx_ETR 引脚，也可以是片内其它外设，通过通用定时器 ETR 来源配置寄存器 SYSCTRL_GTIMETR 进行选择，请参见 14.3.7 片内外设互联 ETR。通过设置控制寄存器 GTIMx_CR0 的 POL 位域，可以选择 ETR 信号的极性是否反相，从而选择门控信号的有效极性为高电平还是低电平。

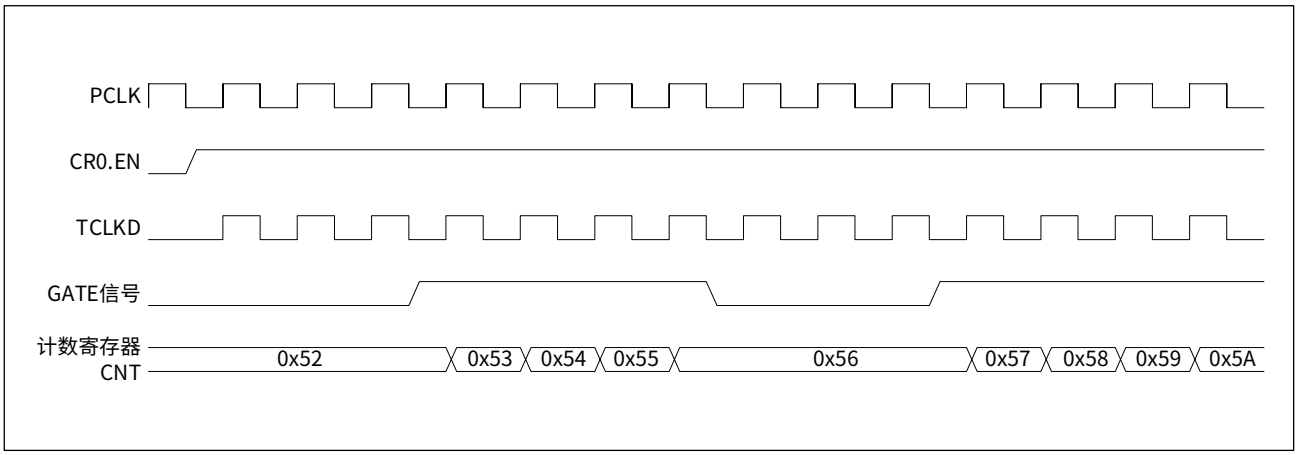
门控模式功能框图如下图所示：

图 14-11 门控模式框图



下图用于说明在门控模式下，当门控电平为无效电平时，即使 GTIMx_CR0.EN 设置为 1，计数器也不会进行计数，只有当门控电平为有效电平时，计数器才进行计数，一旦门控电平失效，则立即暂停计数。

图 14-12 门控模式时序图（分频比为 1，PRS=0x00，门控高电平有效，POL=0x00）

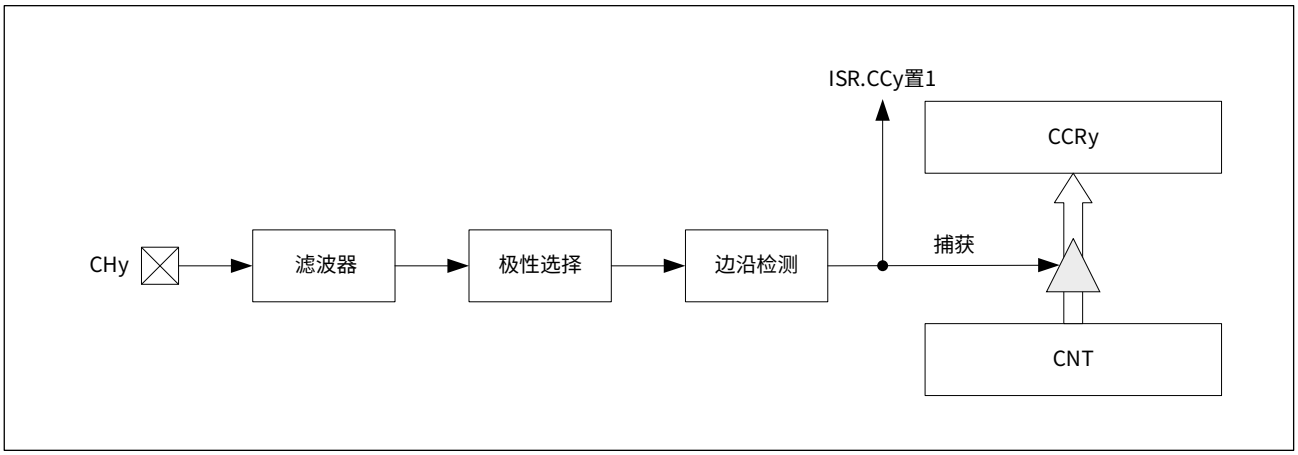


14.3.3 输入捕获功能

14.3.3.1 输入捕获

当捕获比较通道 CHy 上信号发生跳变（上升沿或下降沿）时，硬件自动将当前计数寄存器 GTIMx_CNT 的值存放到对应通道的比较捕获寄存器 GTIMx_CCRy 中，完成一次捕获。输入捕获功能可以用来测量脉冲宽度或者测量频率，其功能框图如下图所示：

图 14-13 输入捕获模式框图



各个通道上触发捕获的条件由比较捕获控制寄存器 GTIMx_CMMR 决定，参见下表：

表 14-4 输入捕获模式配置

通用定时器	通道 CHy	CCyM 位域值	捕获模式配置
GTIMx_CMMR (x=1,2,3,4)	CCyM (y=1,2,3,4)	0001	CHy 上升沿捕获
		0010	CHy 下降沿捕获
		0011	CHy 上下沿同时捕获

当发生一次捕获时，通道 CHy 比较捕获中断标志 GTIMx_ISR.CCy 被硬件置位，如果允许中断 (设置中断使能寄存器 GTIMx_IER.CCy 为 1)，CPU 将响应中断服务程序。退出中断服务程序之前，应设置中断标志清除寄存器 GTIMx_ICR.CCy 为 0 以清除该标志。

14.3.3.2 输入捕获来源

GTIM 的输入捕获来源可以是外部 GTIMx_CHy 引脚，也可以是片内其它外设，通过通用定时器输入捕捉来源配置寄存器 SYSCTRL_GTIMxCAP 进行配置。

当 SYSCTRL_GTIMxCAP.CHy 为 0x00 时，输入捕获信号的外部输入端口由 GPIO 复用功能寄存器 (CPIOx_AFRH 和 CPIOx_AFRL) 进行配置。

当 SYSCTRL_GTIMxCAP.CHy 为 0x01 ~ 0x06 时，输入捕获信号来自片内其它外设，如下表所示：

表 14-5 通用定时器输入捕获来源

SYSCTRL_GTIMxCAP 位域	值	GTIMx 的输入捕获来源
CHy	000	由 GPIOx_AFRH 和 GPIOx_AFRL 配置
	001	UART1 的 RXD 信号
	010	UART2 的 RXD 信号
	011	UART3 的 RXD 信号
	100	VC1 的比较输出信号
	101	VC2 的比较输出信号
	110	LVD 的输出信号

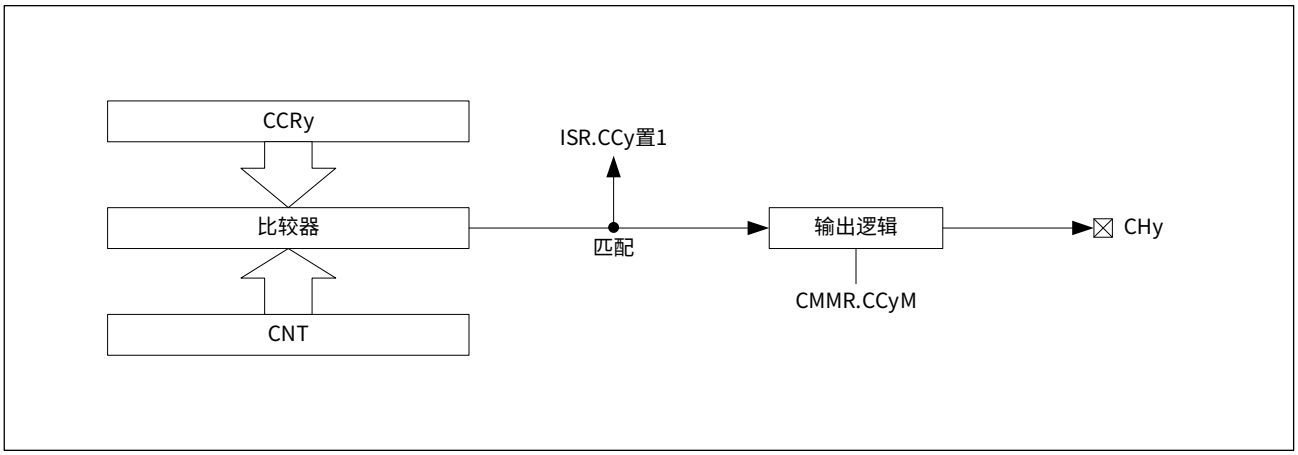
这种配置下，可以在芯片内部实现外部输入的互联，例如将 UART 的 RXD 信号作为输入捕获来源，可以实现对 UART 波特率的自动检测。

14.3.4 输出比较功能

14.3.4.1 输出比较

在输出比较模式下，当前计数寄存器 GTIMx_CNT 的值与对应通道 CHy 的比较捕获寄存器 GTIMx_CCRy 的值相比较，当两者匹配时，比较捕获通道 CHy 可以输出高电平或低电平，同时产生比较中断。输出比较功能可以用来控制一个输出波形或指示一段给定的时间已到达，其功能框图如下图所示：

图 14-14 输出比较模式框图



CHy 引脚的输出动作取决于比较捕获控制寄存器 GTIMx_CMMR 的 CCyM 位域，如下表所示：

表 14-6 输出比较模式配置

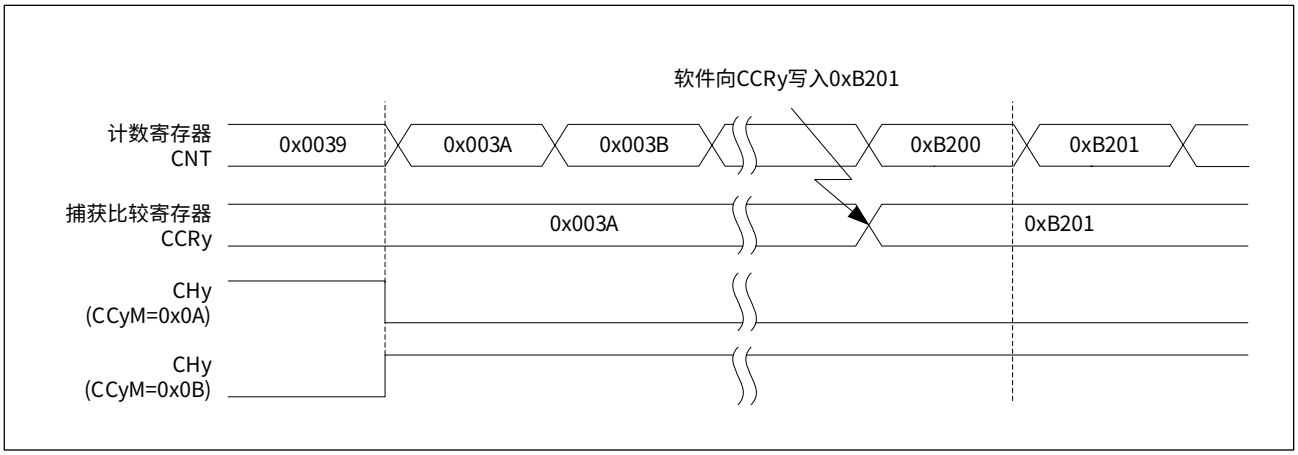
通用定时器	通道 CHy	CCyM 位域值	比较模式配置
GTIMx_CMMR (x=1,2,3,4)	CCyM (y=1,2,3,4)	1000	强制 CHy 输出低电平
		1001	强制 CHy 输出高电平
		1010	在比较匹配时 CHy 置 0
		1011	在比较匹配时 CHy 置 1
		1110	PWM 正向输出（CNT >= CCR 输出高电平）
		1111	PWM 反向输出（CNT < CCR 输出高电平）

当 GTIMx_CMMR 寄存器的 CCyM 位的值为 0x0A 时，比较匹配时，CHy 引脚输出低电平，并且 GTIMx_ISR 寄存器的 CCy 位置 1。

当 GTIMx_CMMR 寄存器的 CCyM 位的值为 0x0B 时，比较匹配时，CHy 引脚输出高电平，并且 GTIMx_ISR 寄存器的 CCy 位置 1。

下图显示了两种比较输出模式：置低电平、置高电平，其中 CCRy 预置为 0x003A。

图 14-15　两种比较输出模式



14.3.4.2　强制输出功能

在强制输出模式下，输出比较信号能够直接由软件强置为高或低状态，而不依赖于比较捕获寄存器 GTIMx_CCRy 和计数寄存器 GTIMx_CNT 的比较结果。

将 GTIMx_CMMR 寄存器中相应的 CCyM 位域设置为 0x8，即可强置 CHy 通道输出低电平。将 GTIMx_CMMR 寄存器中相应的 CCyM 位域设置为 0x9，即可强置 CHy 通道输出高电平。

该模式下，在 GTIMx_CCRy 寄存器和计数器 GTIMx_CNT 之间的比较仍然在进行，相应的标志也会被修改，也会产生相应的中断和 DMA 请求。

14.3.4.3 PWM 输出功能

脉冲宽度调制 (PWM) 模式可以产生一个由重载寄存器 GTIMx_ARR 确定频率、由比较捕获寄存器 GTIMx_CCRy 确定占空比的信号。

向 GTIMx_CCMR 寄存器中的 CCyM 位写入 0xE 或 0xF，能够独立地设置每个 CHy 输出通道产生一路 PWM。

设置 GTIMx_CMMR.CCyM 为 0xE，当 GTIMx_CNT >= GTIMx_CCRy 时，CHy 通道输出高电平，否则输出低电平。

注：

如果 GTIMx_CCRy 中的比较值大于重载寄存器 GTIMx_ARR 的值，则 CHy 通道输出保持为低电平；

如果 GTIMx_CCRy 中的比较值为 0，则 CHy 通道输出保持为高电平。

设置 GTIMx_CMMR.CCyM 为 0xF，当 GTIMx_CNT < GTIMx_CCRy 时，CHy 通道输出高电平，否则输出低电平。

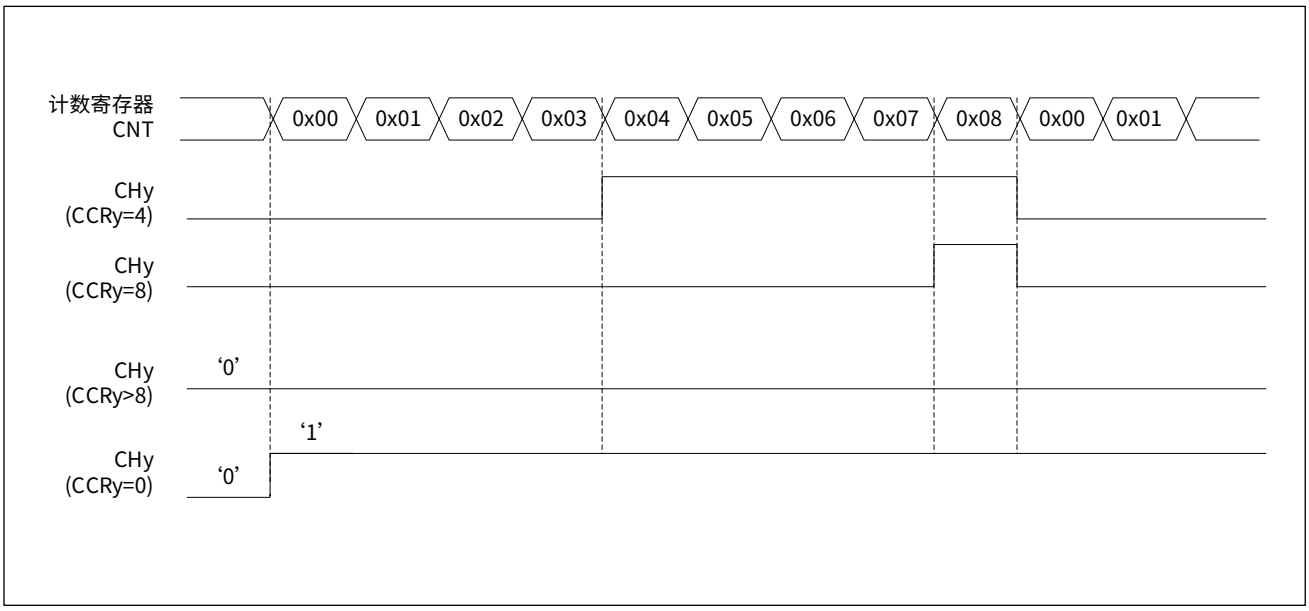
注：

如果 GTIMx_CCRy 中的比较值大于重载寄存器 GTIMx_ARR 的值，则 CHy 通道输出保持为高电平；

如果 GTIMx_CCRy 中的比较值为 0，则 CHy 通道输出保持为低电平。

下图是 GTIMx_CMMR.CCyM 为 0xE、GTIMx_ARR 为 0x08 时，PWM 波形实例：

图 14-16 PWM 波形实例



14.3.5 编码计数模式

GTIM 的捕获 / 比较通道 CH1 和 CH2 引脚可作为正交编码器的接口与编码器连接。设置控制寄存器 GTIMx_CR0 的 ENCMODE 位域为非 0x00 值时，GTIMx 工作在正交编码模式下，计数器的计数时钟由 CH1 和 CH2 引脚输入。

GTIM 支持 3 种编码计数模式：设置 GTIMx_CR0.ENCMODE 为 0x01，计数器只在 CH1 的边沿计数；设置 GTIMx_CR0.ENCMODE 为 0x02，计数器只在 CH2 的边沿计数；设置 GTIMx_CR0.ENCMODE 为 0x03，计数器同时在 CH1 和 CH2 的边沿计数。

通过控制寄存器 GTIMx_CR1 的 CH1POL 和 CH2POL 位域，可以选择通道 CH1 和 CH2 输入信号送入内部电路的极性。通过控制寄存器 GTIMx_CR1 的 CH1FLT 和 CH2FLT 位域，可以配置通道 CH1 和 CH2 的输入滤波器。

设置 GTIMx_CR0.EN 为 1 启动计数器，计数器将由通道 CH1 和 CH2 引脚输入信号经滤波器和极性控制后的信号的每次有效跳变驱动。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。编码器当前计数方向标志 GTIMx_ISR.DIR 由硬件自动设置。当计数方向发生改变时，编码器计数方向改变中断标志位 GTIMx_ISR.DIRCHANGE 将被硬件置位，如果允许中断 (设置 GTIMx_IER.DIRCHANGE 为 1)，CPU 将响应中断服务程序。退出中断服务程序之前，应设置 GTIMx_ICR. DIRCHANGE 为 0 以清除该标志。

计数方向和编码器信号的关系如下表所示：

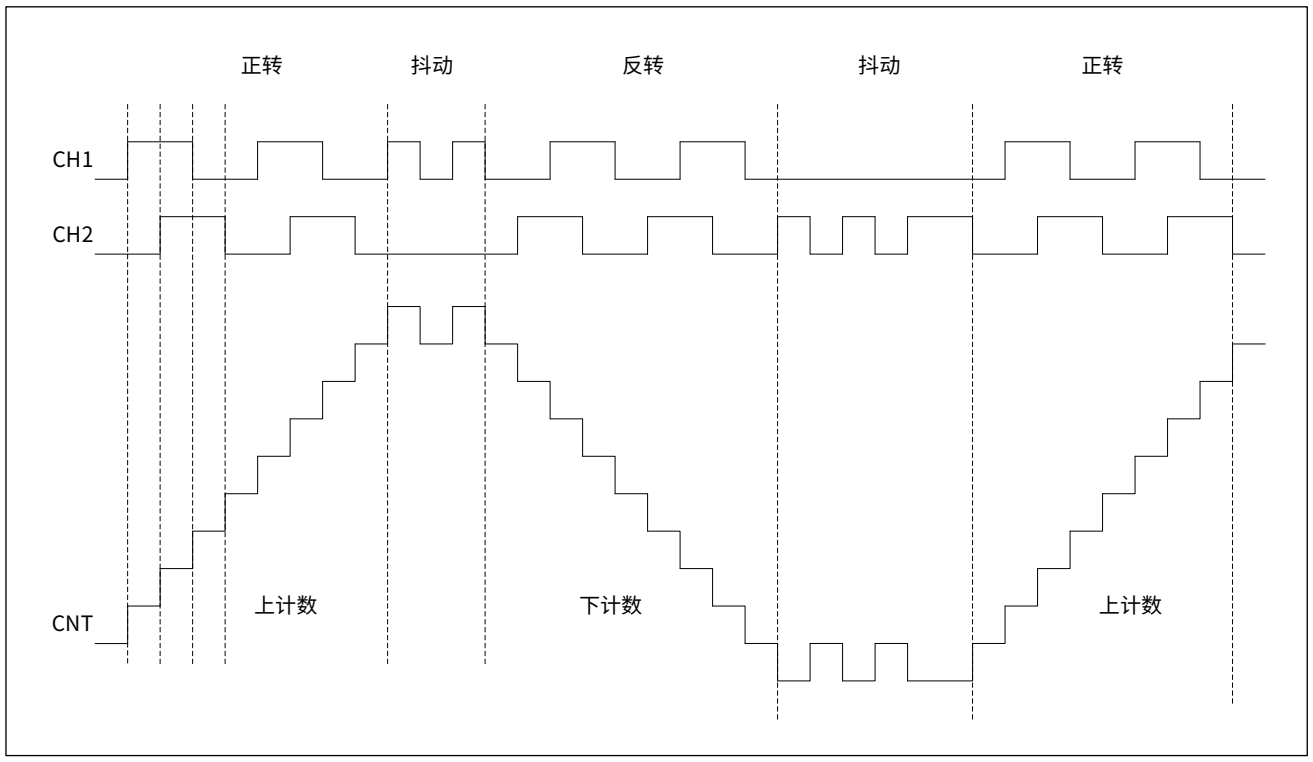
表 14-7 计数方向和编码器信号的关系

	信号的电平		CH1		CH2	
	CH2	CH1	上升	下降	上升	下降
模式 1	高	-	向下计数	向上计数	不计数	不计数
	低	-	向上计数	向下计数	不计数	不计数
模式 2	-	高	不计数	不计数	向上计数	向下计数
	-	低	不计数	不计数	向下计数	向上计数
模式 3	高	高	向下计数	向上计数	向上计数	向下计数
	低	低	向上计数	向下计数	向下计数	向上计数

编码器输出的第三个信号表示机械零点，可以把它连接到 CH3 通道上。通过控制寄存器 GTIMx_CR0 的 ENCRELOAD 位域，选择在上升沿还是下降沿重载计数器 CNT 的值，也可以通过控制寄存器 GTIMx_CR0 的 ENCRESET 位域，选择在上升沿还是下降沿复位计数器 CNT 的值（复位操作使 CNT 寄存器的值为 0x0000）。

下图是一个编码计数模式操作实例，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。

图 14-17　编码器模式下的计数器操作实例



14.3.6　内部级联 ITR

通过 ITR 信号可以实现定时器级联，从而扩大定时器的计数范围，也可以实现对计数时钟源 PCLK 进行 1 ~ 65535 之间任意数分频。

GTIM 的级联输入 ITR 的来源为 BTIM、GTIM 的溢出信号，可通过定时器 ITR 来源配置寄存器 SYSCTRL_TIMITR 来选择，具体配置如下表所示：

表 14-8　ITR 来源配置

SYSCTRL_TIMITR 位域	值	ITR 来源
BTIM3ITR BTIM2ITR BTIM1ITR GTIM4ITR GTIM3ITR GTIM2ITR GTIM1ITR	000	BTIM1 的溢出信号
	001	BTIM2 的溢出信号
	010	BTIM3 的溢出信号
	011	GTIM1 的溢出信号
	100	GTIM2 的溢出信号
	101	GTIM3 的溢出信号
	110	GTIM4 的溢出信号

注：

定时器的 ITR 来源不能选择自身的溢出信号 OV。

14.3.7 片内外设互联 ETR

GTIM 的 ETR 信号来源可以是外部 GTIMx_ETR 引脚，也可以是片内其它外设，通过通用定时器 ETR 来源配置寄存器 SYSCTRL_GTIMETR 进行选择。

当 SYSCTRL_GTIMETR.GTIMxETR 为 0x00 时，ETR 信号的外部输入端口由 GPIO 复用功能寄存器 (CPIOx_AFRH 和 CPIOx_AFRL) 进行配置，如下表所示：

表 14-9 外部 GTIMx_ETR 引脚复用

ETR	引脚	AFR
GTIM1_ETR	PB02/PB03	0x06
GTIM2_ETR	PA00	0x07
GTIM2_ETR	PA04/PA15	0x06
GTIM2_ETR	PA05	0x01
GTIM3_ETR	PA08/PC13	0x06
GTIM4_ETR	PB04/PB11	0x02
GTIM4_ETR	PC15	0x03

当 SYSCTRL_GTIMETR.GTIMxETR 为 0x01 ~ 0x06 时，ETR 信号来自片内其它外设，实现片内外设互联，如下表所示:

表 14-10 GTIMx 的 ETR 来源

SYSCTRL_GTIMETR.GTIMxETR	GTIMx 的 ETR 来源
000	由 GPIOx_AFRH 和 GPIOx_AFRL 配置
001	UART1 的 RXD 信号
010	UART2 的 RXD 信号
011	UART3 的 RXD 信号
100	VC1 的比较输出信号
101	VC2 的比较输出信号
110	LVD 的输出信号

14.4 直接内存访问（DMA）

通过 DMA 触发寄存器 GTIMx_DMA 可选择 GTIMx 在发生 ARR 溢出、触发启动、捕获比较匹配时触发 DMA 功能。使用 DMA 功能，可以实现数据从其他位置写入定时器，或从定时器读出写入其他位置。可应用于数据捕获后数据的自动搬运和更改 PWM 输出的周期或脉宽。

14.5 调试支持

GTIM 支持在调试模式下停止或继续计数，通过调试状态定时器控制寄存器 SYSCTRL_DEBUG 的 GTIMx 位域来设置。

设置 SYSCTRL_DEBUG.GTIMx 为 1，则在调试状态时暂停 GTIMx 的计数器计数；

设置 SYSCTRL_DEBUG.GTIMx 为 0，则在调试状态时 GTIMx 的计数器继续计数。

14.6 编程示例

14.6.1 定时器模式编程示例

以下是定时器模式的编程示例，在这个编程示例中，设置系统时钟 PCLK 的频率为 24MHz，启动定时器 10ms(9.99ms) 后，将产生一个 ARR 溢出中断请求并停止计数。

配置流程如下：

步骤 1：设置 SYSCTRL_APBEN1.GTIMx 位或 SYSCTRL_APBEN2.GTIMx 位为 1，打开 GTIMx 的配置时钟及工作时钟；

注：

GTIM1 和 GTIM2 由 SYSCTRL_APBEN1 控制，GTIM3 和 GTIM4 由 SYSCTRL_APBEN2 控制。

步骤 2：设置 GTIMx_CR0.MODE 为 0x00，使 GTIMx 工作在定时器模式下；

步骤 3：设置 GTIMx_CR0.ONESHOT 为 0x01，使计数器工作在单次计数模式下；

步骤 4：设置 GTIMx_CR0.PRS 为 0x08，分频比为 256，则计数时钟 TCLKD 一个周期约为 10.67μs；

步骤 5：设置 GTIMx_ARR 为 0x03A9（937），则 ARR 溢出周期约为 10.008ms(ARR+1)；

步骤 6：设置 GTIMx_CNT 为 0x0000，以便计数；

步骤 7：设置 GTIMx_IER.OV 为 1，允许 ARR 溢出中断；

步骤 8：设置 GTIMx_CR0.EN 为 1，启动 GTIMx；

步骤 9：ARR 溢出中断产生，进入中断服务程序，设置 GTIMx_ICR.OV 为 0 清除中断标志。

14.6.2 计数器模式编程示例

14.6.2.1 对外部 ETR 信号计数

测量从外部 GTIMx_ETR 引脚输入的信号的上升沿个数的操作步骤如下：

步骤 1：设置 SYSCTRL_AHBEN.GPIOx 为 1，SYSCTRL_APBENx.GTIMx 为 1，打开 GTIMx_ETR 引脚对应的 GPIO 时钟和 GTIMx 配置时钟及工作时钟；

步骤 2：将 GTIMx_ETR 引脚对应的 GPIO 配置成复用输入模式，具体寄存器配置请参见 9 通用输入输出端口(GPIO) 章节；

步骤 3：设置 GTIMx_CR0.MODE 为 0x01，使 GTIMx 工作在计数器模式下；

步骤 4：设置 GTIMx_CR0.TRS 为 0x0，选择计数源为 ETR 输入信号；

步骤 5：配置 GTIMx_ETR.ETRFLT，选择是否滤波或滤波的带宽；

步骤 6：设置 GTIMx_CR0.POL 为 0x0，选择上升沿为计数有效；

步骤 7：设置 GTIMx_CR0.PRS 为 0x00，本例中不对输入的 ETR 信号进行分频；

步骤 8：设置 GTIMx_ARR 为 0xFFFF，将 ARR 溢出周期设为最大；

步骤 9：设置 GTIMx_CNT 为 0x0000，以便计数；

步骤 10：设置 GTIMx_CR0.EN 为 0x1，启动 GTIMx。

14.6.2.2 对内部 ITR 信号计数

以下示例中使用 BTIM1 作为 GTIM1 的分频器，对 PCLK 时钟进行 13 分频，操作步骤如下：

步骤 1：设置 SYSCTRL_APBEN1.GTIM1 位为 1，打开 GTIM1 的配置时钟及工作时钟；

步骤 2：设置 GTIM1_CR0.MODE 为 0x01，使 GTIM1 工作在计数器模式下；

步骤 3：设置 GTIM1_CR0.TRS 为 0x1，选择计数源为内部 ITR 信号；

步骤 4：设置 SYSCTRL_TIMITR.GTIM1ITR 为 0x00，选择 GTIM1 的 ITR 来源为 BTIM1 的溢出信号；

步骤 5：设置 GTIM1_CR0.PRS 为 0x00，不对 ITR 信号进行分频，分频通过 BTIM1 完成；

步骤 6：设置 GTIM1_ARR 为 0xFFFF，将 ARR 溢出周期设为最大；

步骤 7：设置 GTIM1_CNT 的初值为 0x0000，以便计数；

步骤 8：设置 GTIM1_CR0.EN 为 0x1，开启定时器；

步骤 9：设置 SYSCTRL_APBEN2.BTIM 位为 1，打开 BTIM1 的配置时钟及工作时钟；

步骤 10：设置 BTIM1_BCR.MODE 为 0x00，选择 BTIM1 工作在定时器模式，使用 PCLK 时钟计数；

步骤 11：设置 BTIM1_BCR.PRS 为 0x00，不对 PCLK 时钟信号分频；

步骤 12：设置 BTIM1_BCR.ONESHOT 为 0x00，BTIM1 为连续计数模式；

步骤 13：设置 BTIM1_ARR 寄存器为 0xC，这样 BTIM1 在计数 (ARR+1) 个 PCLK 周期，即 13 个 PCLK 周期后，将产生一个溢出信号 OV，此 OV 信号作为 GTIM1 的输入信号 ITR，引发 GTIM1 的 1 次计数；

步骤 14：设置 BTIM1_CNT 的初值为 0x0000，以便计数；

步骤 15：设置 BTIM1_BCR.EN 为 0x1，启动 BTIM1。

14.6.3 触发启动模式编程示例

通过外部 GTIMx_ETR 引脚输入的信号触发启动计数器计数，操作步骤如下：

步骤 1：设置 SYSCTRL_AHBEN.GPIOx 为 1，SYSCTRL_APBENx.GTIMx 为 1，打开 GTIMx_ETR 引脚对应的 GPIO 时钟和 GTIMx 配置时钟及工作时钟；

步骤 2：将 GTIMx_ETR 引脚对应的 GPIO 配置成复用输入模式，具体寄存器配置请参见 9 通用输入输出端口(GPIO) 章节；

步骤 3：设置 GTIMx_CR0.TRS 为 0x0，选择触发信号为 ETR 输入信号；

步骤 4：配置 GTIMx_ETR.ETRFLT，选择是否滤波或滤波的带宽；

步骤 5：配置 GTIMx_CR0.POL，设置触发信号为上升沿还是下降沿有效；

步骤 6：设置 GTIMx_CR0.PRS 为 0x00，本例中不对 PCLK 时钟进行分频；

步骤 7：设置 GTIMx_ARR 为 0xFFFF，将 ARR 溢出周期设为最大；

步骤 8：设置 GTIMx_CNT 的初值为 0x0000，以便计数；

步骤 9：设置 GTIMx_ICR.TI 为 0x0，清除中断标志；

步骤 10：如果需要使能触发中断请求，设置寄存器 GTIMx_IER.TI 位为 1；

步骤 11：设置 GTIMx_CR0.MODE 为 0x02，使 GTIMx 工作在触发模式下；当有效触发信号输入时，计数器将开始工作。

注：

应先设置触发信号的极性，再设置定时器的模式，否则会引起误触发。

14.6.4 门控模式编程示例

以外部 GTIMx_ETR 引脚输入的信号作为门控信号，控制计数器计数，操作步骤如下：

步骤 1：设置 SYSCTRL_AHBEN.GPIOx 为 1，SYSCTRL_APBENx.GTIMx 为 1，打开 GTIMx_ETR 引脚对应的 GPIO 时钟和 GTIMx 配置时钟及工作时钟；

步骤 2：将 GTIMx_ETR 引脚对应的 GPIO 配置成复用输入模式，具体寄存器配置请参见 9 通用输入输出端口(GPIO) 章节；

步骤 3：设置 GTIMx_CR0.MODE 为 0x03，使 GTIMx 工作在门控模式下；

步骤 4：设置 GTIMx_CR0.TRS 为 0x0，选择门控信号为 ETR 输入信号；

步骤 5：配置 GTIMx_ETR.ETRFLT，选择是否滤波或滤波的带宽；

步骤 6：配置 GTIMx_CR0.POL，设置门控信号为高电平还是低电平有效；

步骤 7：设置 GTIMx_CR0.PRS 为 0x00，本例中不对 PCLK 时钟进行分频；

步骤 8：设置 GTIMx_ARR 为 0xFFFF，将 ARR 溢出周期设为最大；

步骤 9：设置 GTIMx_CNT 的初值为 0x0000，以便计数；

步骤 10：设置 GTIMx_CR0.EN 为 0x01，启动定时器；当门控信号有效时，计数器将开始工作。

14.6.5 输入捕获编程示例

在定时器模式下，测量一个脉冲的宽度的操作步骤如下：

步骤 1：设置 SYSCTRL_APBEN1.GTIMx 位或 SYSCTRL_APBEN2.GTIMx 位为 1，打开 GTIMx 的配置时钟及工作时钟；

注：

GTIM1 和 GTIM2 由 SYSCTRL_APBEN1 控制，GTIM3 和 GTIM4 由 SYSCTRL_APBEN2 控制。

步骤 2：设置 GTIMx_CR0.MODE 为 0x00，使 GTIMx 工作在定时器模式下；

步骤 3：设置 GTIMx_CR0.PRS，选择适当的分频系数；

步骤 4：根据信号的特性，设置 GTIMx_ARR 为合适值，如果 GTIMx_ARR 设置的较小，计数器时钟频率较快，输入脉宽的时间较长，则有可能 GTIMx_ARR 发生溢出；如发生了溢出，需要及时清除溢出标记，并根据溢出标记产生的次数，计算脉宽时间；

步骤 5：配置 GTIMx_CR1.CHyPOL，选择通道 CHy 送入内部电路的信号极性；

步骤 6：配置 GTIMx_CR1.CHyFLT，配置通道 CHy 输入滤波器；

步骤 7：设置 GTIMx_CMMR.CCyM 为 0x01，选择上升沿捕获；

步骤 8：设置 GTIMx_IER.CCy 为 0x01，允许捕获中断；

步骤 9：设置 GTIMx_CR0.EN 为 0x1，开启定时器；

步骤 10：等待捕获中断的产生，读取当前 GTIMx_CCRy 中的值，然后设置 GTIMx_ICR.CCy 的相应位为 0x0，清除中断标志；

步骤 11：设置 GTIMx_CMMR.CCyM 为 0x02，选择下降沿捕获；

步骤 12：等待下一次捕获中断的产生，读取当前 GTIMx_CCRy 中的值，这样两次读取的 GTIMx_CCRy 值得差值即为输入脉宽的长度。

14.6.6 输出比较编程示例

在定时器模式下，通过通道 CH1 对外输出低电平表明定时时间到 , 操作步骤如下：

步骤 1：设置 SYSCTRL_APBEN1.GTIMx 位或 SYSCTRL_APBEN2.GTIMx 位为 1，打开 GTIMx 的配置时钟及工作时钟；

注：

GTIM1 和 GTIM2 由 SYSCTRL_APBEN1 控制，GTIM3 和 GTIM4 由 SYSCTRL_APBEN2 控制。

步骤 2：设置 GTIMx_CR0.MODE 为 0x00，使 GTIMx 工作在定时器模式下；

步骤 3：设置 GTIMx_CR0.PRS，选择适当的分频系数；

步骤 4：设置 GTIMx_CMMR.CC1M 为 0x0A，使 GTIMx 的 CH1 通道在比较匹配时置 0；

步骤 5：设置 GTIMx_CNT 为 0x0000，清零计数器；

步骤 6：设置 GTIMx_CCR1，设定定时时间；

步骤 7：设置 GTIMx_CR0.EN 为 0x1，开启定时器；当 GTIMx_CNT 中的计数值等于 GTIMx_CCR1 的值时，CH1 上的输出电平由高变为低。

14.6.7 PWM 输出编程示例

在定时器模式下，输出一个 PWM 波形的操作步骤如下：

步骤 1：设置 SYSCTRL_APBEN1.GTIMx 位或 SYSCTRL_APBEN2.GTIMx 位为 1，打开 GTIMx 的配置时钟及工作时钟；

注：

GTIM1 和 GTIM2 由 SYSCTRL_APBEN1 控制，GTIM3 和 GTIM4 由 SYSCTRL_APBEN2 控制。

步骤 2：设置 GTIMx_CR0.MODE 为 0x00，使 GTIMx 工作在定时器模式下；

步骤 3：设置 GTIMx_CR0.PRS，选择适当的分频系数；

步骤 4：根据输出 PWM 波形的周期要求，设置 GTIMx_ARR 为合适值，即 GTIMx_ARR 可以改变 PWM 的周期；

步骤 5：根据输出 PWM 波形的脉宽要求，设置 GTIMx_CCRy 为合适值，即 GTIMx_CCRy 可以改变 PWM 的占空比；

步骤 6：设置 GTIMx_CMMR.CCyM 为 0x0E，选择 PWM 正向输出，即 GTIMx_CNT>=GTIMx_CCRy 输出高电平；

步骤 7：设置 GTIMx_IER.CCy 为 0x01，允许比较中断；

步骤 8：设置 GTIMx_CR0.EN 为 0x1，开启定时器；

步骤 9：等待比较中断的产生，可根据需要修改 GTIMx_CCRy 的值以改变 PWM 波形的占空比，然后设置 GTIMx_ICR.CCy 的相应位为 0x0，清除中断标志。

14.6.8 编码器模式编程示例

配置 GTIMx 工作在编码器模式 3 的操作步骤如下：

步骤 1：设置 SYSCTRL_APBEN1.GTIMx 位或 SYSCTRL_APBEN2.GTIMx 位为 1，打开 GTIMx 的配置时钟及工作时钟；

注：

GTIM1 和 GTIM2 由 SYSCTRL_APBEN1 控制，GTIM3 和 GTIM4 由 SYSCTRL_APBEN2 控制。

步骤 2：设置寄存器 GTIMx_CR0.ENCMODE 为 0x03，使 GTIMx 工作在编码模式 3 下，计数器同时在 CH1 和 CH2 的边沿计数；

步骤 3：设置寄存器 GTIMx_ARR，选择合适的值；

步骤 4：设置 GTIMx_CR1 寄存器中的 CH1POL 和 CH2POL 位，选择 CH1 和 CH2 引脚输入信号送入内部电路的极性；

步骤 5：设置 GTIMx_CR1 寄存器中的 CH1FLT 和 CH2FLT 位，选择合适的滤波器配置；

步骤 6：如果需要设置寄存器 GTIMx_IER 中的 DIRCHANGE 位使能，开启方向改变的中断；

步骤 7：设置 GTIMx_CR0.EN 为 0x1，开启定时器。

14.7 寄存器列表

GTIM1 基地址：GTIM1_BASE = 0x4000 0400

GTIM2 基地址：GTIM2_BASE = 0x4000 1000

GTIM3 基地址：GTIM3_BASE = 0x4001 4000

GTIM4 基地址：GTIM4_BASE = 0x4001 4400

表 14-11 GTIM 寄存器列表

寄存器名称	寄存器地址	寄存器描述
GTIMx_ARR	GTIMx_BASE + 0x300	重载寄存器
GTIMx_CNT	GTIMx_BASE + 0x304	计数寄存器
GTIMx_CMMR	GTIMx_BASE + 0x308	比较捕获控制寄存器
GTIMx_ETR	GTIMx_BASE + 0x30C	外部触发控制寄存器
GTIMx_CR0	GTIMx_BASE + 0x310	控制寄存器 0
GTIMx_IER	GTIMx_BASE + 0x314	中断使能寄存器
GTIMx_ISR	GTIMx_BASE + 0x318	中断标志寄存器
GTIMx_ICR	GTIMx_BASE + 0x31C	中断标志清除寄存器
GTIMx_CCR1	GTIMx_BASE + 0x320	比较捕获寄存器 1
GTIMx_CCR2	GTIMx_BASE + 0x324	比较捕获寄存器 2
GTIMx_CCR3	GTIMx_BASE + 0x328	比较捕获寄存器 3
GTIMx_CCR4	GTIMx_BASE + 0x32C	比较捕获寄存器 4
GTIMx_CR1	GTIMx_BASE + 0x330	控制寄存器 1
GTIMx_DMA	GTIMx_BASE + 0x340	DMA 触发寄存器

14.8 寄存器描述

有关寄存器描述里所使用的缩写，请参见 1 文档约定章节。

14.8.1 GTIMx_CR0 控制寄存器 0

Address offset: 0x310 Reset value: 0x0000 0000

位域	名称	权限	功能描述
31:21	RFU	-	保留位，请保持默认值
20:19	ENCRELOAD	RW	编码模式，计数寄存器外部重载条件配置 00：无功能 01：CH3 上升沿重载计数寄存器 CNT 10：CH3 下降沿重载计数寄存器 CNT 注：重载操作使 CNT 寄存器的值等于 ARR 寄存器的值
18:17	ENCRESET	RW	编码模式，计数寄存器外部复位条件配置 00：无功能 01：CH3 上升沿复位计数寄存器 CNT 10：CH3 下降沿复位计数寄存器 CNT 注 1：复位操作使 CNT 寄存器的值等于 0x0000 注 2：当 ENCRESET 与 ENCRELOAD 配置为同一边沿时，ENCRESET 有效。
16:15	ENCMODE	RW	编码计数模式使能配置 00：定时器功能由 MODE 位配置 01：编码计数模式 1，CH1 信号变化沿计数 10：编码计数模式 2，CH2 信号变化沿计数 11：编码计数模式 3，CH1/CH2 信号变化沿计数
14:11	PRSSTATUS	RO	分频电路当前正在使用的预分频系数 定时器溢出或 EN 从 0 变会 1 时会从 PRS 更新该值
10:7	PRS	RW	预分频器分频系数配置 0：DIV1 4：DIV16 8：DIV256 12：DIV4096 1：DIV2 5：DIV32 9：DIV512 13：DIV8192 2：DIV4 6：DIV64 10：DIV1024 14：DIV16384 3：DIV8 7：DIV128 11：DIV2048 15：DIV32768
6	TOGEN	RW	TOG 引脚输出使能控制 0：TOGP、TOGN 输出电平均为 0 1：TOGP、TOGN 输出电平相反的信号
5	ONESHOT	RW	单次 / 连续计数模式控制 0：连续计数模式 1：单次计数模式
4	POL	RW	ETR 输入信号极性选择 0：ETR 正向 (触发模式上升沿有效，门控模式高电平有效) 1：ETR 反向 (触发模式下降沿有效，门控模式低电平有效)

位域	名称	权限	功能描述
3	TRS	RW	触发源选择 0：ETR 输入信号 1：ITR，详见 14.3.6 内部级联 ITR
2:1	MODE	RW	基本定时模式配置 00：定时器模式，计数时钟源为 PCLK 01：计数器模式，计数时钟源为 TRS 信号 10：触发启动模式，计数时钟源为 PCLK，TRS 信号触发计数器启动 11：门控模式，计数时钟源为 PCLK，ETR 输入信号作为门控
0	EN	RW	定时器运行控制 0：定时器停止 1：定时器运行

14.8.2 GTIMx_CR1 控制寄存器 1

Address offset: 0x330 Reset value: 0x0000 0000

位域	名称	权限	功能描述
31:16	RFU	-	保留位，请保持默认值
15	CH4POL	RW	CH4 通道输入信号极性配置 0：CH4 引脚反相信号送入内部电路 1：CH4 引脚同相信号送入内部电路
14:12	CH4FLT	RW	CH4 输入信号滤波配置 功能描述详见 CH1FLT
11	CH3POL	RW	CH3 通道输入信号极性配置 0：CH3 引脚反相信号送入内部电路 1：CH3 引脚同相信号送入内部电路
10:8	CH3FLT	RW	CH3 输入信号滤波配置 功能描述详见 CH1FLT
7	CH2POL	RW	CH2 通道输入信号极性配置 0：CH2 引脚反相信号送入内部电路 1：CH2 引脚同相信号送入内部电路
6:4	CH2FLT	RW	CH2 输入信号滤波配置 功能描述详见 CH1FLT
3	CH1POL	RW	CH1 通道输入信号极性配置 0：CH1 引脚反相信号送入内部电路 1：CH1 引脚同相信号送入内部电路
2:0	CH1FLT	RW	CH1 输入信号滤波配置；采样时钟为 PCLK 或 PCLK 分频，采样点个数为 N 000：无滤波 001：F_{sample} = PCLK， N=2 010：F_{sample} = PCLK， N=4 011：F_{sample} = PCLK， N=6 100：F_{sample} = PCLK/4， N=4 101：F_{sample} = PCLK/4， N=6 110：F_{sample} = PCLK/8， N=4 111：F_{sample} = PCLK/8， N=6

14.8.3 GTIMx_ETR 外部触发控制寄存器

Address offset: 0x30C Reset value: 0x0000 0000

位域	名称	权限	功能描述
31:7	RFU	-	保留位，请保持默认值
6:4	ETRFLT	RW	ETR 输入信号滤波配置；采样时钟为 PCLK 或 PCLK 分频，采样点个数为 N 000：无滤波 001：F_{sample} = PCLK， N=2 010：F_{sample} = PCLK， N=4 011：F_{sample} = PCLK， N=6 100：F_{sample} = PCLK/4， N=4 101：F_{sample} = PCLK/4， N=6 110：F_{sample} = PCLK/8， N=4 111：F_{sample} = PCLK/8， N=6
3:0	RFU	-	保留位，请保持默认值

14.8.4 GTIMx_CMMR 比较捕获控制寄存器

Address offset: 0x308 Reset value: 0x0000 0000

位域	名称	权限	功能描述
31:16	RFU	-	保留位，请保持默认值
15:12	CC4M	RW	CH4 捕获比较模式配置 功能描述详见 CC1M
11:8	CC3M	RW	CH3 捕获比较模式配置 功能描述详见 CC1M
7:4	CC2M	RW	CH2 捕获比较模式配置 功能描述详见 CC1M
3:0	CC1M	RW	CH1 捕获比较模式配置 0000：无功能 0001：上升沿捕获 0010：下降沿捕获 0011：上下沿同时捕获 1000：强制输出低电平 1001：强制输出高电平 1010：在比较匹配时置 0 1011：在比较匹配时置 1 1110：PWM 正向输出（CNT >= CCR 输出高电平） 1111：PWM 反向输出（CNT < CCR 输出高电平）

14.8.5 GTIMx_ARR 重载寄存器

Address offset: 0x300 Reset value: 0x0000 FFFF

位域	名称	权限	功能描述
31:16	RFU	-	保留位，请保持默认值
15:0	ARR	RW	定时器重载值 注：编码计数模式下，ARR 应设置为 0xFFFF。

14.8.6 GTIMx_CNT 计数寄存器

Address offset: 0x304 Reset value: 0x0000 0000

位域	名称	权限	功能描述
31:16	RFU	-	保留位，请保持默认值
15:0	CNT	RW	定时器计数值

14.8.7 GTIMx_CCR1 比较捕获寄存器 1

Address offset: 0x320 Reset value: 0x0000 FFFF

位域	名称	权限	功能描述
31:16	RFU	-	保留位，请保持默认值
15:0	CCR	RW	CH1 通道比较 / 捕获值

14.8.8 GTIMx_CCR2 比较捕获寄存器 2

Address offset: 0x324 Reset value: 0x0000 0000

位域	名称	权限	功能描述
31:16	RFU	-	保留位，请保持默认值
15:0	CCR	RW	CH2 通道比较 / 捕获值

14.8.9 GTIMx_CCR3 比较捕获寄存器 3

Address offset: 0x328 Reset value: 0x0000 FFFF

位域	名称	权限	功能描述
31:16	RFU	-	保留位，请保持默认值
15:0	CCR	RW	CH3 通道比较 / 捕获值

14.8.10 GTIMx_CCR4 比较捕获寄存器 4

Address offset: 0x32C Reset value: 0x0000 0000

位域	名称	权限	功能描述
31:16	RFU	-	保留位，请保持默认值
15:0	CCR	RW	CH4 通道比较 / 捕获值

14.8.11 GTIMx_IER 中断使能寄存器

Address offset: 0x314 Reset value: 0x0000 0000

位域	名称	权限	功能描述
31:10	RFU	-	保留位，请保持默认值
9	DIRCHANGE	RW	编码器计数方向改变中断控制 0：禁止 1：使能
8:7	RFU	-	保留位，请保持默认值
6	CC4	RW	CH4 捕获比较中断使能控制 0：禁止 1：使能
5	CC3	RW	CH3 捕获比较中断使能控制 0：禁止 1：使能
4	CC2	RW	CH2 捕获比较中断使能控制 0：禁止 1：使能
3	CC1	RW	CH1 捕获比较中断使能控制 0：禁止 1：使能
2	UD	RW	下溢出中断使能控制 0：禁止下溢出中断 1：使能下溢出中断 注：仅适用于编码计数模式
1	TI	RW	触发中断使能控制 0：禁止触发中断 1：使能触发中断
0	OV	RW	计数器 ARR 溢出中断使能控制 0：禁止 ARR 溢出中断 1：使能 ARR 溢出中断

14.8.12 GTIMx_ISR 中断标志寄存器

Address offset: 0x318 Reset value: 0x0000 0000

位域	名称	权限	功能描述
31:11	RFU	-	保留位，请保持默认值
10	DIR	RO	编码器当前计数方向标志 0：正向计数 1：反向计数
9	DIRCHANGE	RO	编码器计数方向改变中断标志 0：计数方向未改变 1：计数方向已改变
8:7	RFU	-	保留位，请保持默认值
6	CC4	RO	CH4 比较捕获中断标志 0：未发生比较捕获事件 1：已发生比较捕获事件
5	CC3	RO	CH3 比较捕获中断标志 0：未发生比较捕获事件 1：已发生比较捕获事件
4	CC2	RO	CH2 比较捕获中断标志 0：未发生比较捕获事件 1：已发生比较捕获事件
3	CC1	RO	CH1 比较捕获中断标志 0：未发生比较捕获事件 1：已发生比较捕获事件
2	UD	RO	计数器下溢出标志 0：计数器未下溢出 1：计数器已下溢出 注：仅适用于编码计数模式
1	TI	RO	触发标志 0：未发生触发事件 1：已发生触发事件
0	OV	RO	计数器 ARR 溢出标志 0：计数器未 ARR 溢出 1：计数器已 ARR 溢出 注：当计数器值从 ARR 变为 0 时产生该标志

14.8.13 GTIMx_ICR 中断标志清除寄存器

Address offset: 0x31C Reset value: 0x0000 03FF

位域	名称	权限	功能描述
31:10	RFU	-	保留位，请保持默认值
9	DIRCHANGE	R1W0	编码器计数方向改变中断控制 W0：清除编码器计数方向改变中断标志 W1：无功能
8:7	RFU	-	保留位，请保持默认值
6	CC4	R1W0	CH4 比较捕获中断标志清除 W0：清除比较捕获中断标志 W1：无功能
5	CC3	R1W0	CH3 比较捕获中断标志清除 W0：清除比较捕获中断标志 W1：无功能
4	CC2	R1W0	CH2 比较捕获中断标志清除 W0：清除比较捕获中断标志 W1：无功能
3	CC1	R1W0	CH1 比较捕获中断标志清除 W0：清除比较捕获中断标志 W1：无功能
2	UD	R1W0	计数器下溢出标志清除 W0：清除计数器下溢出标志 W1：无功能 注：仅适用于编码计数模式
1	TI	R1W0	触发标志清除 W0：清除触发标志 W1：无功能
0	OV	R1W0	计数器 ARR 溢出标志清除 W0：清除计数器 ARR 溢出标志 W1：无功能

14.8.14 GTIMx_DMA DMA 触发寄存器

Address offset: 0x340 Reset value: 0x0000 0000

位域	名称	权限	功能描述
31:6	RFU	-	保留位，请保持默认值
5	CC4	RW	CH4 捕获比较中断触发 DMA 使能控制 0：禁止 1：使能
4	CC3	RW	CH3 捕获比较中断触发 DMA 使能控制 0：禁止 1：使能
3	CC2	RW	CH2 捕获比较中断触发 DMA 使能控制 0：禁止 1：使能
2	CC1	RW	CH1 捕获比较中断触发 DMA 使能控制 0：禁止 1：使能
1	TRS	RW	计数器被触发启动时同步触发 DMA 使能控制 0：无功能 1：计数器被触发启动时同步触发 DMA
0	OV	RW	计数器 ARR 溢出触发 DMA 使能控制 0：无功能 1：计数器 ARR 溢出触发 DMA

15 独立看门狗定时器（IWDT）

15.1 概述

CW32F020 内部集成独立看门狗定时器 (IWDT)，使用专门的内部 RC 时钟源 RC10K，可避免运行时受到外部因素影响。一旦启动 IWDT，用户需要在规定时间间隔内对 IWDT 的计数器进行重载，否则计数器溢出会触发复位或产生中断信号。IWDT 启动后，可停止计数。可选择在深度休眠模式下 IWDT 保持运行或暂停计数。

专门设置的键值寄存器，可以锁定 IWDT 的关键寄存器，防止寄存器被意外修改。

15.2 主要特性

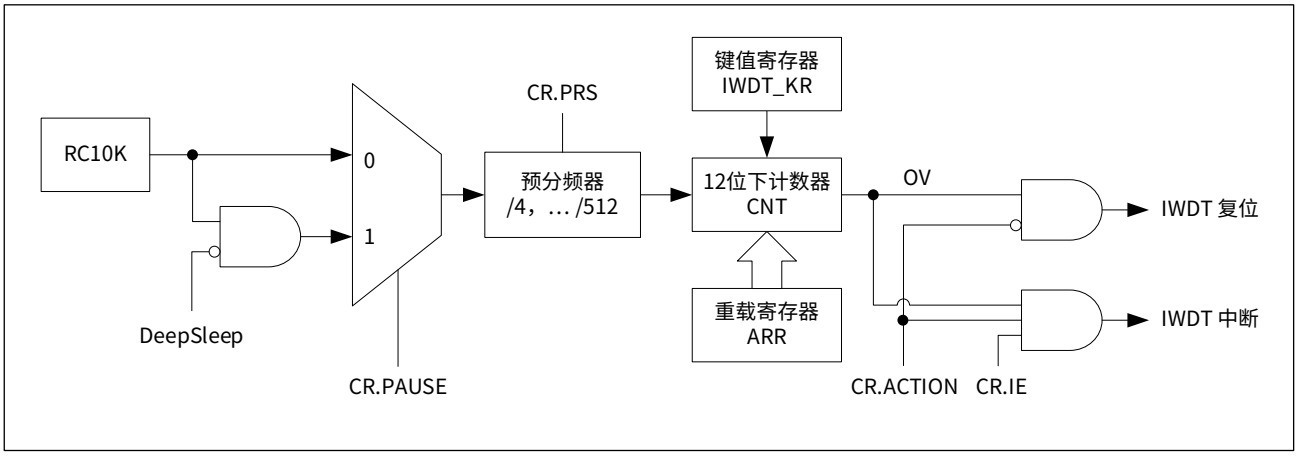
- 12bit 的向下计数器
- 独立内部低速 RC 振荡器
- 可编程时钟预分频周期
- 溢出可触发中断或复位
- 寄存器保护锁功能
- 深度休眠模式下可暂停计数

15.3 功能描述

15.3.1 功能框图

IWDT 功能框图如下图所示：

图 15-1 IWDT 功能框图



IWDT 由一个 12 位可重载的向下计数器实现，其计数时钟源为内部专用低速 RC 振荡器 RC10K，通过控制寄存器 IWDT_CR 的 PRS 位域可对其时钟源 RC10K 信号进行 4 ~ 512 的预分频。IWDT 计数器发生溢出时可选择产生中断和复位信号。

15.3.2 工作方式

启动 IWDT 的计数器，需要向键值寄存器 IWDT_KR 写入 0xCCCC，计数器开始从 0xFFF 向下计数。

在计数器减到 0 之前，向 IWDT_KR 寄存器写入 0xAAAA，会触发计数器重载，将 ARR 寄存器值加载到计数器。当 IWDT 计数器值递减到 0，会产生溢出事件，溢出事件可触发 MCU 复位或产生 IWDT 中断信号，同时触发计数器重载。控制寄存器 IWDT_CR 的 ACTION 和 IE 位域，用于控制看门狗溢出时是否产生中断和复位，如下表所示：

表 15-1 工作方式

IWDT_CR.ACTION	IWDT_CR.IE	IWDT 溢出后动作
0	X	复位
1	1	不复位，只产生中断
1	0	不复位，不产生中断

IWDT 在 MCU 进入深度休眠模式时，可选择暂停 IWDT 计数，从而达到更低的系统整体功耗：控制寄存器 IWDT_CR 的 PAUSE 位域为 0，深度休眠模式时保持 IWDT 定时器运行；为 1 时暂停计数，当 MCU 退出深度休眠模式时 IWDT 自动恢复计数。

15.3.3 窗口选项

向窗口寄存器 IWDT_WINR 写入一个小于重载寄存器 IWDT_ARR 的值，可使 IWDT 工作于窗口看门狗模式。IWDT_WINR 寄存器的默认值是 0x0FFF，即窗口选项默认是关闭的。

修改 IWDT_WINR 寄存器的值，会触发重载操作，将 ARR 寄存器值加载到计数器。

使用 IWDT 窗口功能，用户应在计数器值小于等于 IWDT_WINR 窗口值，并且递减到 0 之前进行重载操作，以避免产生复位或看门狗定时器溢出。在看门狗计数器的值大于窗口值时进行重载操作，将触发系统复位。

15.3.4 寄存器锁定功能

通过 IWDT 的键值寄存器 IWDT_KR，可配置 IWDT 的重要寄存器 IWDT_CR、IWDT_ARR、IWDT_WINR 为锁定状态或解除锁定，锁定状态下不可对寄存器进行修改操作。

向 IWDT_KR 寄存器写入 0x5555，解除对 IWDT_CR、IWDT_ARR、IWDT_WINR 寄存器的锁定；向 IWDT_KR 寄存器写入其他任何值，启动锁定保护。

CW32F020 在上电复位后，IWDT_CR、IWDT_ARR、IWDT_WINR 寄存器默认处于锁定状态，用户需先解除锁定，才可对其进行修改操作。

15.3.5 启动刷新与停止

配置 IWDT_KR 寄存器，可实现 IWDT 的启动、刷新和停止操作：

- 写入 0xCCCC，启动 IWDT
- 写入 0xAAAA，重载计数器，即刷新 IWDT
- 顺序写入 0x5A5A、0xA5A5，停止 IWDT

注：

上述操作会同时启动 IWDT 寄存器锁定保护。

15.3.6 状态寄存器

状态寄存器 IWDT_SR，指示 IWDT 当前运行状态或寄存器更新状态，如下表所示：

表 15-2 IWDT 状态指示

IWDT_SR 位	名称	描述	1	0
4	RUN	运行标志	运行	未运行
3	OV	溢出标志	产生溢出	未溢出
5	RELOAD	计数器重载标志	正在重载	重载完成
2	WINRF	WINR 寄存器更新标志	正在更新	更新完成
1	ARRF	ARR 寄存器更新标志	正在更新	更新完成
0	CRF	CR 寄存器更新标志	正在更新	更新完成

为确保对 IWDT 寄存器的操作正确性，用户在更新 IWDT_WINR、IWDT_ARR、IWDT_CR 之后，需要检查 WINRF、ARRF、CRF 标志位是否为 0，以确认操作是否完成。

为确保对 IWDT 的重载操作，用户在进行重载操作后，应当检查 RELOAD 标志位是否为 0。

15.3.7 定时时长设定

IWDT 的计数时钟源为专用低速时钟 RC10K（时钟频率约为 10kHz，具体请参阅数据手册），通过控制寄存器 IWDT_CR 的 PSR 位域，可对其时钟源 RC10K 信号进行分频，如下表所示：

表 15-3 IWDT 分频系数表

IWDT_CR.PSR	预分频值
000	4
001	8
010	16
011	32
100	64
101	128
110	256
111	512

看门狗定时时长计算公式：

$$T = (4 \times 2^{PRS} / f) \times (ARR + 1)$$

其中，f 为时钟源 RC10K 的频率，PRS 为预分频系数，ARR 为重载值。

故，当时钟源 RC10K 的频率为 10000Hz 时，IWDT 的最长和最短定时范围：

$$\text{IWDT 最短定时} = (4 \times 2^{0} / 10000) \times (0x000 + 1) \approx 400\ \mu s$$

$$\text{IWDT 最长定时} = (4 \times 2^{7} / 10000) \times (0xFFF + 1) \approx 209.7\ s$$

例：当时钟源 RC10K 的频率为 10000Hz 时，设置预分频值为 64，重载值为 512，则：

$$\text{IWDT 定时时长} = (4 \times 2^{4} / 10000) \times (512 + 1) = 3.28\ s$$

15.4 编程示例

15.4.1 配置 IWDT 为独立看门狗

步骤 1：设置 SYSCTRL_APBEN1.IWDT 为 1，使能 IWDT 的配置时钟；

步骤 2：向 IWDT_KR 寄存器写入 0xCCCC，启动 IWDT；

注：

需要启动 IWDT 后，才可对相关寄存器进行修改。

步骤 3：向 IWDT_KR 寄存器写入 0x5555，解除 IWDT 寄存器锁定功能；

步骤 4：配置 IWDT_CR，配置看门狗计数时钟与 RC10K 振荡器的预分频值、溢出后动作、深度休眠模式下是否自动暂停；

步骤 5：配置 IWDT_ARR，配置看门狗的溢出周期；

步骤 6：等待 IWDT_SR.ARRF 和 IWDT_SR.CRF 变为 0，等待重载值和 CR 寄存器更新完成；

步骤 7：向 IWDT_KR 寄存器写入 0xAAAA，加载 ARR 到 IWDT 计数器。

15.4.2 配置 IWDT 为窗口看门狗

步骤 1：设置 SYSCTRL_APBEN1.IWDT 为 1，使能 IWDT 的配置时钟；

步骤 2：向 IWDT_KR 寄存器写入 0xCCCC，启动 IWDT；

注：

需要启动 IWDT 后，才可对相关寄存器进行修改。

步骤 3：向 IWDT_KR 寄存器写入 0x5555，解除 IWDT 寄存器锁定功能；

步骤 4：配置 IWDT_CR，配置看门狗计数时钟与 RC10K 振荡器的预分频值、溢出后动作、深度休眠模式下是否自动暂停；

步骤 5：配置 IWDT_ARR，配置看门狗的重载值；

步骤 6：配置 IWDT_WINR，配置窗口大小，注意 IWDT_WINR 必须小于 IWDT_ARR 重载值；

步骤 7：等待 IWDT_SR.ARRF、IWDT_SR.WINRF 和 IWDT_SR.CRF 变为 0，等待重载值、窗口寄存器和 CR 寄存器更新完成；

步骤 8：向 IWDT_KR 寄存器写入 0xAAAA，加载 ARR 到 IWDT 计数器。

15.4.3 刷新 IWDT（喂狗操作）

步骤 1：向 IWDT_KR 寄存器写入 0xAAAA，加载 ARR 到计数器；

步骤 2：等待 IWDT_SR.RELOAD 变为 0，等待重载操作完成。

15.5 寄存器列表

IWDT 基地址：IWDT_BASE = 0x4000 3000

表 15-4 IWDT 寄存器列表

寄存器名称	寄存器地址	寄存器描述
IWDT_KR	IWDT_BASE + 0x00	键值寄存器
IWDT_CR	IWDT_BASE + 0x04	控制寄存器
IWDT_ARR	IWDT_BASE + 0x08	重载寄存器
IWDT_SR	IWDT_BASE + 0x0C	状态寄存器
IWDT_WINR	IWDT_BASE + 0x10	窗口寄存器
IWDT_CNT	IWDT_BASE + 0x24	计数值寄存器

15.6 寄存器描述

有关寄存器描述里所使用的缩写，请参见 1 文档约定章节。

15.6.1 IWDT_KR 键值寄存器

Address offset: 0x00 Reset value: 0x0000 0000

位域	名称	权限	功能描述
31:16	RFU	-	保留位，请保持默认值
15:0	KR	WO	写入 0xCCCC：启动 IWDT 计数器 写入 0xAAAA：重载 IWDT 计数值 依次写入 0x5A5A、0xA5A5：停止看门狗 写入 0x5555：解除 IWDT_ARR、IWDT_WINR、IWDT_CR 的写保护 写入非 0x5555：使能 IWDT_ARR、IWDT_WINR、IWDT_CR 的写保护

15.6.2 IWDT_CR 控制寄存器

Address offset: 0x04 Reset value: 0x0000 0000

位域	名称	权限	功能描述
31:6	RFU	-	保留位，请保持默认值
5	PAUSE	RW	Deepsleep 模式下 IWDT 暂停控制 0：DeepSleep 模式下 IWDT 继续运行 1：DeepSleep 模式下 IWDT 自动暂停
4	IE	RW	IWDT 中断使能控制 0：禁止看门狗中断 1：使能看门狗中断
3	ACTION	RW	IWDT 溢出后动作配置 0：看门狗溢出后产生复位 1：看门狗溢出后产生中断
2:0	PRS	RW	配置看门狗计数时钟与 RC10K 振荡器的分频比 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 111：512 分频

注：

向 IWDT_KR 寄存器写入 0x5555 后，才可修改本寄存器；IWDT_SR.CRF 为 0 时，才可修改 IWDT_CR。

15.6.3 IWDT_ARR 重载寄存器

Address offset: 0x08 Reset value: 0x0000 0FFF

位域	名称	权限	功能描述
31:12	RFU	-	保留位，请保持默认值
11:0	ARR	RW	IWDT 重载值

注：

向 IWDT_KR 寄存器写入 0x5555 后，才可修改本寄存器；IWDT_SR.ARRF 为 0，才可修改本寄存器。

15.6.4 IWDT_CNT 计数值寄存器

Address offset: 0x24 Reset value: 0x0000 0FFF

位域	名称	权限	功能描述
31:12	RFU	-	保留位，请保持默认值
11:0	CNT	RO	计数值寄存器 注：连续两次读到的数值相同，才可认为读出该计数值

15.6.5 IWDT_WINR 窗口寄存器

Address offset: 0x10 Reset value: 0x0000 0FFF

位域	名称	权限	功能描述
31:12	RFU	-	保留位，请保持默认值
11:0	WINR	RW	窗口比较器比较值

15.6.6 IWDT_SR 状态寄存器

Address offset: 0x0C Reset value: 0x0000 0000

位域	名称	权限	功能描述
31:6	RFU	-	保留位，请保持默认值
5	RELOAD	RO	WDT 计数器重载标志 0：WDT 计数器已完成重载操作 1：WDT 计数器正在进行重载操作 注：窗口模式下，进行重载操作后需要等待该标志变为 0
4	RUN	RO	WDT 运行标志 0：WDT 没有运行 1：WDT 正在运行
3	OV	RW0	WDT 溢出标志 R0：WDT 没有溢出 R1：WDT 溢出，写 0 清零
2	WINRF	RO	WINR 寄存器更新标志 0：WINR 寄存器更新完成 1：WINR 寄存器正在更新 注：写 WINR 寄存器后需要等待该标志变为 0
1	ARRF	RO	ARR 寄存器更新标志 0：ARR 寄存器更新完成 1：ARR 寄存器正在更新 注：写 ARR 寄存器后需要等待该标志变为 0
0	CRF	RO	CR 寄存器更新标志 0：CR 寄存器更新完成 1：CR 寄存器正在更新 注：写 CR 寄存器后需要等待该标志变为 0

16 窗口看门狗定时器（WWDT）

16.1 概述

CW32F020 内部集成窗口看门狗定时器 (WWDT)，用户需要在设定的时间窗口内进行刷新，否则将触发系统复位。WWDT 通常被用来监测有严格时间要求的程序执行流程，防止由外部干扰或未知条件造成应用程序的执行异常，导致发生系统故障。

16.2 主要特性

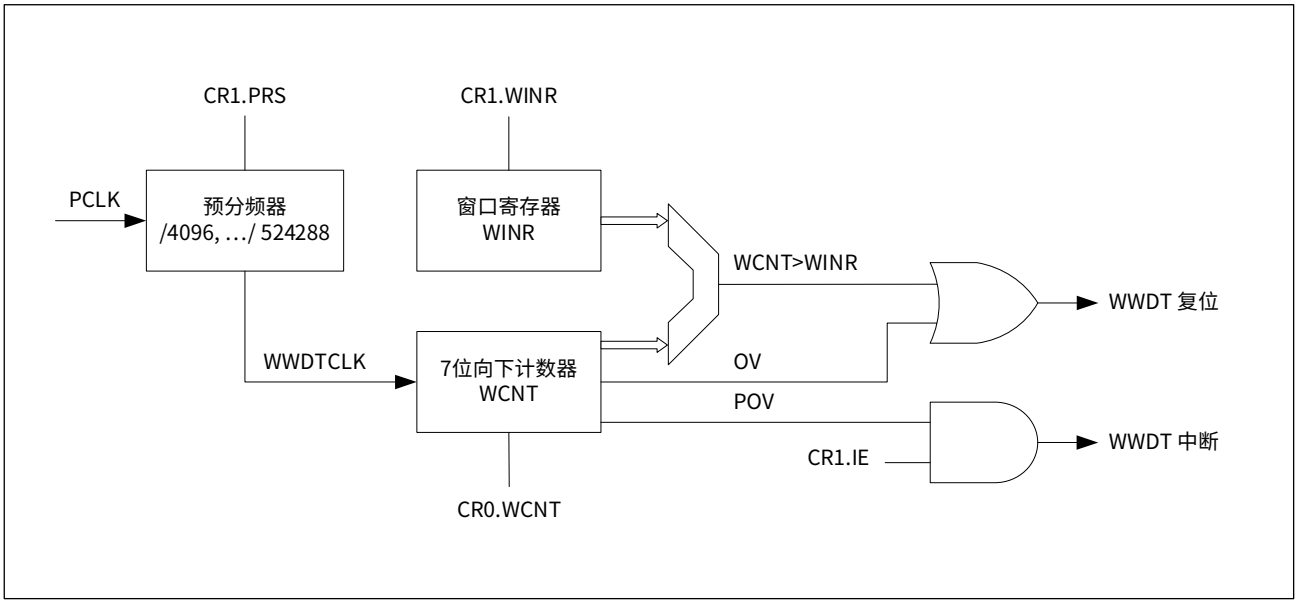
- 7 位向下计数器
- PCLK 时钟驱动，8 级预分频，最大分频 524288
- 支持预溢出中断和计数溢出、加载计数值出错复位
- 开启后不可关闭，除非系统复位

16.3 功能描述

16.3.1 功能框图

WWDT 功能框图如下图所示：

图 16-1 WWDT 功能框图



WWDT 内含一个 7 位递减计数器，计数时钟源为内部系统时钟 PCLK，通过控制寄存器 WWDT_CR1 的 PRS 位域可对其时钟源 PCLK 进行分频，分频后得到计数时钟 WWDTCLK 用来驱动计数器计数。

WWDT 在深度休眠模式下将停止计数，CPU 被唤醒后恢复正常工作。

16.3.2 工作方式

系统复位后，窗口看门狗 WWDT 处于关闭状态，设置控制寄存器 WWDT_CR0 的 EN 位域为 1，可启动 WWDT。WWDT 开启后，除非发生复位，否则不能被关闭。

启动 WWDT 之前，用户必须预设看门狗溢出时间和窗口时间。设置 WWDT_CR0.WCNT 的值，可更新计数器的初始值，设置 WWDT_CR1.WINR 的值，可配置看门狗的窗口值，窗口值必须小于看门狗计数器的初始值。启动 WWDT 后，计数器从初始值开始递减计数。

当计数器递减到 0x40 时，将产生预溢出信号 POV，设置 WWDT_CR1.IE 为 1 将产生预溢出中断。

当计数器递减到 0x3F 时，将产生溢出信号 OV，该溢出信号可触发系统复位。

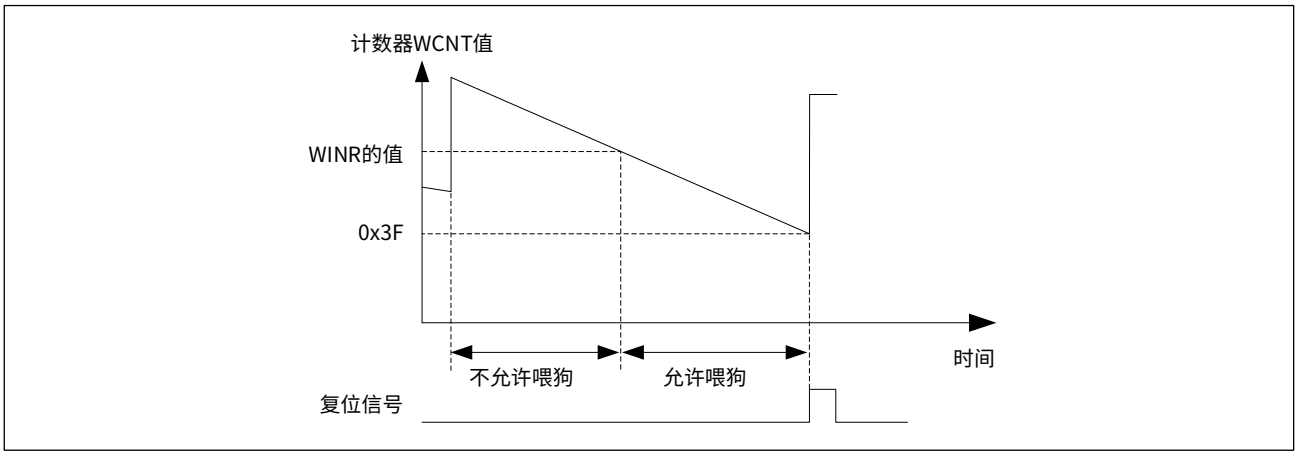
在当前计数值 WCNT 大于窗口值 WINR 时，更新看门狗计数器，也会触发系统复位。

16.3.3 刷新计数器

WWDT 运行时，只有在计数值小于等于窗口值且计数值递减到 0x3F 之前，才能刷新计数器，即喂狗操作，否则将产生系统复位。

设置 WWDT_CR1.WINR 的值，可配置复位前延时时间窗口的上限值，延时时间窗口的下限固定为 0x3F，窗口看门狗的喂狗操作时机，如下图所示：

图 16-2 窗口看门狗的刷新（喂狗）时机



16.3.4 喂狗时间

窗口看门狗的喂狗时间计算公式：

喂狗时间下限：

$$T_{WWDT\_MIN} = T_{PCLK} \times 4096 \times 2^{PRS} \times (WCNT - WINR)$$

喂狗时间上限：

$$T_{WWDT\_MAX} = T_{PCLK} \times 4096 \times 2^{PRS} \times (WCNT - 0x3F)$$

其中：

T_{PCLK}：PCLK 的时钟周期

PRS：预分频系数

WCNT：计数器当前计数值

WINR：看门狗窗口值

例：

当 PCLK 频率为 24MHz 时，预分频系数 PRS 设置为 0x01，WCNT 设置为 0x6F，窗口值 WINR 设置为 0x4F，则喂狗时间如下：

$$T_{WWDT\_MIN} = 1 / 24MHz \times 4096 \times 2^{1} \times (0x6F - 0x4F) \approx 10.922\ ms$$

$$T_{WWDT\_MAX} = 1 / 24MHz \times 4096 \times 2^{1} \times (0x6F - 0x3F) = 16.384\ ms$$

即，喂狗时间最早不能早于 10.922ms，否则将发生加载计数值出错而引起系统复位；喂狗最晚不能迟于 16.384ms，否则将发生下溢出引起系统复位。正确喂狗的时间窗口大小约为 5ms。

16.3.5 复位与中断

当计数器 WCNT 的值递减到 0x40 时，预溢出中断标志位 WWDT_SR.POV 由硬件置位，若设置了控制寄存器 WWDT_CR1 的 IE 位域为 1（**注：该位置 1 后不可清 0**），将产生预溢出中断请求。用户可在中断服务程序中更新计数器 WCNT，以避免 WWDT 产生复位。

在以下条件之一成立时，均可触发系统复位：

1. 计数器 WCNT 的值递减到 0x3F；
2. 更新计数器 WCNT 时当前计数值大于窗口值；
3. 向 WWDT_CR0.WCNT 写入小于或等于 0x3F 的值。

16.4 编程示例

WWDT 基本配置流程

步骤 1：设置 SYSCTRL_APBEN1.WWDT 为 1，使能 WWDT 的配置时钟及工作时钟；

步骤 2：通过 WWDT_CR1.PRS 配置窗口看门狗计数器时钟的预分频；

步骤 3：通过 WWDT_CR1.WINR 配置窗口看门狗计数的比较值；

步骤 4：通过 WWDT_CR0.WCNT 配置计数器的初始值；

步骤 5：根据是否需要使能预溢出中断，配置 WWDT_CR1.IE；

步骤 6：设置 WWDT_CR0.EN 为 1 启动窗口看门狗。

WWDT 喂狗

当计数器 WCNT 的值递减到小于等于 WWDT_CR1.WINR，且大于 0x3F 之前，向 WWDT_CR0.WCNT 重新写入计数器的初始值。

16.5 寄存器列表

WWDT 基地址：WWDT_BASE = 0x4000 2C00

表 16-1 WWDT 寄存器列表

寄存器名称	寄存器地址	寄存器描述
WWDT_CR0	WWDT_BASE + 0x00	控制寄存器 0
WWDT_CR1	WWDT_BASE + 0x04	控制寄存器 1
WWDT_SR	WWDT_BASE + 0x08	状态寄存器

16.6 寄存器描述

有关寄存器描述里所使用的缩写，请参见 1 文档约定章节。

16.6.1 WWDT_CR0 控制寄存器 0

Address offset: 0x00 Reset value: 0x0000 007F

位域	名称	权限	功能描述
31:8	RFU	-	保留位，请保持默认值
7	EN	RW1	WWDT 使能控制，使能后不能禁止 0：禁止 WWDT 1：使能 WWDT
6:0	WCNT	RW	读：WWDT 计数器当前计数值 写：更新 WWDT 计数器当前计数值

16.6.2 WWDT_CR1 控制寄存器 1

Address offset: 0x04 Reset value: 0x0000 007F

位域	名称	权限	功能描述
31:11	RFU	-	保留位，请保持默认值
10	IE	RW	WWDT 预溢出中断使能控制 0：禁止预溢出中断 1：使能预溢出中断 注：置 1 后不可清零
9:7	PRS	RW	WWDT 计数时钟预分频值 000：PCLK / 4096 001：PCLK / 8192 010：PCLK / 16384 011：PCLK / 32768 100：PCLK / 65536 101：PCLK / 131072 110：PCLK / 262144 111：PCLK / 524288
6:0	WINR	RW	看门狗窗口值

16.6.3 WWDT_SR 状态寄存器

Address offset: 0x08 Reset value: 0x0000 0000

位域	名称	权限	功能描述
31:1	RFU	-	保留位，请保持默认值
0	POV	RW	WWDT 预溢出标志 0：WWDT 未发生预溢出 1：WWDT 已发生预溢出，写 0 以清除该标志

17 通用异步收发器（UART）

17.1 概述

CW32F020 内部集成 3 个通用异步收发器 (UART)，支持异步全双工、同步半双工和单线半双工模式，支持硬件数据流控和多机通信；可编程数据帧结构，可以通过小数波特率发生器提供宽范围的波特率选择。

UART 控制器工作在双时钟域下，允许在深度休眠模式下进行数据的接收，接收完成中断可以唤醒 MCU 回到运行模式。

17.2 主要特性

- 支持双时钟域驱动
  - 配置时钟 PCLK
  - 传输时钟 UCLK
- 可编程数据帧结构
  - 数据字长：8、9 位，LSB 在前
  - 校验位：无校验、奇校验、偶校验
  - 停止位长度：1、1.5、2 位
- 16 位整数、4 位小数波特率发生器
- 支持异步全双工、同步半双工、单线半双工
- 支持硬件流控 RTS、CTS
- 支持直接内存访问 (DMA)
- 支持多机通信，自动地址识别
- 6 个带中断标志的中断源
- 错误检测：奇偶校验错误、帧结构错误
- 低功耗模式下收发数据，中断唤醒 MCU

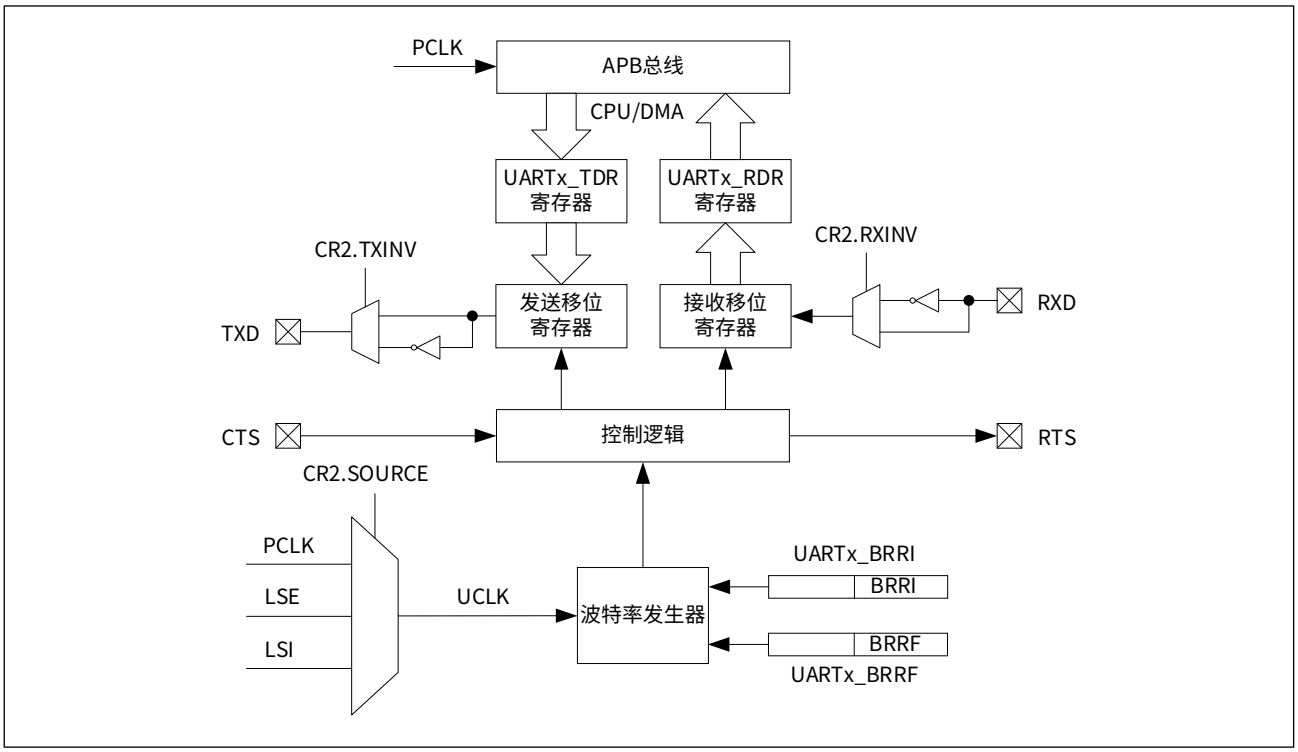
17.3 功能描述

17.3.1 功能框图

UART 控制器挂载到 APB 总线上，配置时钟域 PCLK，固定为 APB 总线时钟 PCLK，用于寄存器配置逻辑工作；传输时钟域 UCLK，用于数据收发逻辑工作，其来源可选择 PCLK 时钟、外部低速时钟（LSE）以及内部低速时钟（LSI）。双时钟域的设计更便于波特率的设置，支持从深度休眠模式下唤醒控制器。

UART 控制器的功能框图如下图所示：

图 17-1 UART 功能框图



UART 控制器支持多种工作模式，在不同的工作模式下，各引脚具有不同的功能，引脚配置也不同，如下表所示（其中 x = 1 , 2 , 3）：

表 17-1 UART 端口配置

工作模式	UART 引脚	作用	GPIO 配置
异步全双工	UARTx_TXD	数据串行输出	数字，推挽输出 / 开漏输出，复用
	UARTx_RXD	数据串行输入	数字，上拉输入，复用
	UARTx_RTS [1]	请求发送	数字，推挽输出 / 开漏输出，复用
	UARTx_CTS [1]	允许发送	数字，上拉输入，复用
同步半双工	UARTx_TXD	时钟信号输出	数字，推挽输出 / 开漏输出，复用
	UARTx_RXD	数据的发送和接收	数字，推挽输出 / 开漏输出，复用
单线半双工	UARTx_TXD	数据的发送和接收	数字，开漏输出，复用（需外接上拉）
	UARTx_RXD	不使用	可作通用 IO 使用

注 1：

仅硬件数据流控模式下需要配置。

17.3.2 同步模式

UART 支持同步半双工工作模式。在该模式下，UARTx_TXD 引脚输出同步移位时钟信号，UARTx_RXD 引脚进行数据的发送和接收。UARTx_TXD 和 UARTx_RXD 引脚的具体配置参见表 17-1 UART 端口配置。

UART 的同步模式可以与 SPI 的单线半双工模式进行通信，此时 SPI 必须配置为固定的电平模式：时钟极性 CPOL 为 1、时钟相位 CPHA 为 1。

设置控制寄存器 UARTx_CR1 的 SYNC 位域为 1，使 UART 工作于同步半双工模式。此时 UART 只能作为主机，数据字长只能是 8 位，且 UARTx_CR2. SIGNAL 和 UARTx_CR2.ADDREN 必须清零。

17.3.2.1 波特率设置

同步半双工模式下，波特率计算公式：

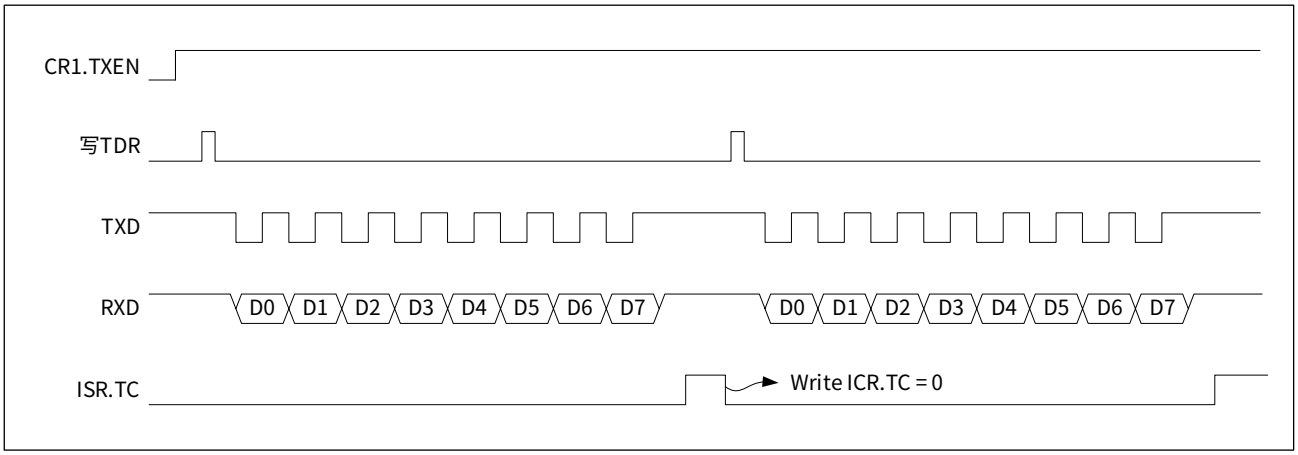
$$BaudRate = UCLK / 12$$

其中，UCLK 是 UART 的传输时钟，其来源可以是 PCLK、LSE 或 LSI，通过控制寄存器 UARTx_CR2 的 SOURCE 位域来选择。

17.3.2.2 数据收发

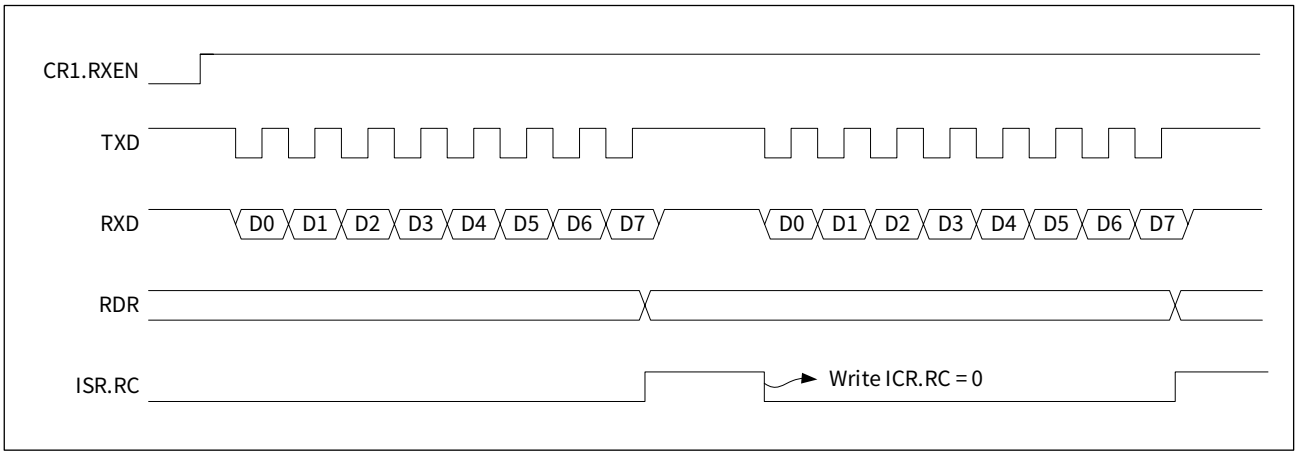
将 UART 发送使能位 UARTx_CR1.TXEN 置 1 使 UART 工作于发送状态，将数据写入 UARTx_TDR 寄存器后，数据将从 UARTx_RXD 引脚串行输出，同时 UARTx_TXD 引脚输出同步移位时钟信号。当数据发送完成后，发送完成中断标志位 UARTx_ISR.TC 会被硬件置位。

图 17-2 同步半双工模式发送时序（发送两个字节数据）



将 UART 接收使能位 UARTx_CR1.RXEN 置 1 使 UART 工作于接收状态，UARTx_TXD 引脚将输出同步移位时钟信号，数据从 UARTx_RXD 引脚串行输入。数据接收完成后，接收完成中断标志位 UARTx_ISR.RC 会被硬件置位，此时可以读取 UARTx_RDR 寄存器。接收下一个字节之前，必须将 UARTx_ISR.RC 标志位清零，RC 标志清零后，立即开始接收下一个字节数据。

图 17-3 同步半双工模式接收时序（接收两个字节数据）



17.3.3 异步模式

17.3.3.1 数据帧格式

在 UART 异步通信中，数据是以数据帧的形式发送和接收的，一帧数据包括一个起始位、一个数据域、一个可选的校验位和宽度可编程的停止位。

起始位

起始位长度固定为 1 位，起始位的判定方式可以选择下降沿或低电平，通过控制寄存器 UARTx_CR1 的 START 位域来选择。一般选择下降沿作为起始位的判定方式，低电平方式判定起始位主要应用于低功耗模式，请参见 17.4 低功耗模式。

数据域

数据字长可以设置为 8 位或 9 位，通过有无校验位来自动配置。当设置为无奇偶校验时，数据字长为 8 位，当设置为有校验时（包括自定义校验、偶校验、奇校验），数据字长自动设置为 9 位。请参见表 17-2 数据帧结构。

校验位

校验方式支持自定义校验、偶校验和奇校验，具体通过控制寄存器 UARTx_CR1 的 PARITY 位域来选择。

自定义校验：校验位由软件写入 UARTx_TDR[8] 和读取 UARTx_RDR[8] 决定，主要应用于多机通信，请参见 17.3.3.6 多机通信。

奇校验：校验位使一帧数据中数据位和校验位中“1”的总数为奇数。

偶校验：校验位使一帧数据中数据位和校验位中“1”的总数为偶数。

UARTx_CR1.PARITY，用于设置校验方式，与数据帧对应关系如下表所示：

表 17-2 数据帧结构

UARTx_CR1.PARITY	校验方式	数据帧
00	无校验	起始位 +8 位数据 + 停止位
01	自定义	起始位 +8 位数据 + 自定义校验位 + 停止位
10	偶校验	起始位 +8 位数据 + 偶校验位 + 停止位
11	奇校验	起始位 +8 位数据 + 奇校验位 + 停止位

停止位

停止位长度可以配置为 1、1.5 或 2 位，具体通过控制寄存器 UARTx_CR1 的 STOP 位域来配置。

17.3.3.2 小数波特率发生器

波特率的产生

UART 的接收和发送波特率是相同的，由同一个波特率发生器产生。

波特率发生器支持 16 倍采样、8 倍采样、4 倍采样和专用采样这 4 种采样模式，具体的采样模式通过控制寄存器 UARTx_CR1 的 OVER 位域来选择。

1. OVER = 00，设置 16 倍采样，波特率计算公式：

$$BaudRate = UCLK / (16 \times BRRI + BRRF)$$

UCLK 是 UART 的传输时钟，其来源可以是 PCLK、LSE 或 LSI，具体来源通过 UARTx_CR2.SOURCE 来选择。

BRRI（UARTx_BRRI［15:0］），是波特率计数器的整数部分，可设置范围为 1 ~ 65535。

BRRF（UARTx_BRRF［3:0］），是波特率计数器的小数部分，可设置范围为 0 ~ 15。

例 1：

当传输时钟 UCLK 的频率为 24MHz 时，设置 BRRI = 156（即 UARTx_BRRI = 0x9C），BRRF = 4（即 UARTx_BRRF = 0x04），则：

$$BaudRate = 24000000 / (16 \times 156 + 4) = 9600 \text{ bps}$$

例 2：

当传输时钟 UCLK 的频率为 24MHz 时，要求配置 BaudRate = 115200 bps，计算

$$16 \times BRRI + BRRF = 24000000 / 115200 = 208.33$$

则：

BRRI = 208.33 / 16 = 13.02，最接近的整数是：13（0x0D）

BRRF = 0.02×16 = 0.32，最接近的整数是：0（0x00）

即需要设置 UARTx_BRRI 为 0x0D，UARTx_BRRF 为 0x00

此时，实际波特率 BaudRate = 115384.62 bps，误差率为 0.16%

2. OVER = 01，设置 8 倍采样，波特率计算公式：

$$BaudRate = UCLK / (8 \times BRRI)$$

3. OVER = 10，设置 4 倍采样，波特率计算公式：

$$BaudRate = UCLK / (4 \times BRRI)$$

4. OVER = 11，设置专用采样，波特率计算公式：

$$BaudRate = (256 \times UCLK) / BRRI$$

注：

专用采样仅适合传输时钟源为 LSE 或者 LSI 时，进行 2400bps、4800bps 或 9600bps 波特率下的 UART 通信。

例 1：

传输时钟 UCLK 选择 LSE，频率为 32.768kHz，要求配置 9600 bps 波特率，则：

BRRI = 256×32768 / 9600 = 873.81，最接近的整数是：874（0x36A）

即需要设置 UARTx_BRRI 为 0x36A

此时，实际波特率 BaudRate = 9597.95 bps，误差为 0.02%。

波特率设置示例

表 17-3 ~ 表 17-9 列举了常用 MCU 频率时，波特率计数寄存器的设定值及波特率误差。

表 17-3　UCLK 为 4MHz 波特率设置示例（OVER = 00）

波特率（bps）	BRRI	BRRF	实际波特率	误差率
4800	52	1	4801.92	0.04%
9600	26	1	9592.33	-0.08%
14400	17	6	14388.49	-0.08%
19200	13	0	19230.77	0.16%
38400	6	8	38461.54	0.16%
56000	4	7	56338.03	0.60%
57600	4	5	57971.01	0.64%
115200	2	3	114285.71	-0.79%
256000	1	0	250000	-2.34%

表 17-4　UCLK 为 8MHz 波特率设置示例（OVER = 00）

波特率（bps）	BRRI	BRRF	实际波特率	误差率
4800	104	3	4799.04	-0.02%
9600	52	1	9603.84	0.04%
14400	34	12	14388.49	-0.08%
19200	26	1	19184.65	-0.08%
38400	13	0	38461.54	0.16%
56000	8	15	55944.06	-0.10%
57600	8	11	57553.96	-0.08%
115200	4	5	115942.03	0.64%
500000	1	0	500000	0.00%

表 17-5　UCLK 为 16MHz 波特率设置示例（OVER = 00）

波特率（bps）	BRRI	BRRF	实际波特率	误差率
4800	208	5	4800.48	0.01%
9600	104	3	9598.08	-0.02%
14400	69	7	14401.44	0.01%
19200	52	1	19207.68	0.04%
38400	26	1	38369.30	-0.08%
56000	17	14	55944.06	-0.10%
57600	17	6	57553.96	-0.08%
115200	8	11	115107.91	-0.08%
1000000	1	0	1000000	0.00%

表 17-6　UCLK 为 24MHz 波特率设置示例（OVER = 00）

波特率（bps）	BRRI	BRRF	实际波特率	误差率
4800	312	8	4800.00	0.00%
9600	156	4	9600.00	0.00%
14400	104	3	14397.12	-0.02%
19200	78	2	19200.00	0.00%
38400	39	1	38400.00	0.00%
56000	26	13	55944.06	-0.10%
57600	26	1	57553.96	-0.08%
115200	13	0	115384.62	0.16%
1500000	1	0	1500000	0.00%

表 17-7　UCLK 为 32MHz 波特率设置示例（OVER = 00）

波特率（bps）	BRRI	BRRF	实际波特率	误差率
4800	416	11	4799.76	0.00%
9600	208	5	9600.96	0.01%
14400	138	14	14401.44	0.01%
19200	104	3	19196.16	-0.02%
38400	52	1	38415.37	0.04%
56000	35	11	56042.03	0.08%
57600	34	12	57553.96	-0.08%
115200	17	6	115107.91	-0.08%
2000000	1	0	2000000	0.00%

表 17-8　UCLK 为 48MHz 波特率设置示例（OVER = 00）

波特率（bps）	BRRI	BRRF	实际波特率	误差率
4800	625	0	4800.00	0.00%
9600	312	8	9600.00	0.00%
14400	208	5	14401.44	0.01%
19200	156	4	19200.00	0.00%
38400	78	2	38400.00	0.00%
56000	53	9	56009.33	0.02%
57600	52	1	57623.05	0.04%
115200	26	1	115107.91	-0.08%
3000000	1	0	3000000	0.00%

表 17-9　UCLK 为 32.768kHz 波特率设置示例（OVER = 11）

波特率（bps）	BRRI	实际波特率	误差率
2400	3495	2400.17	0.01%
4800	1748	4798.97	-0.02%
9600	874	9597.95	-0.02%

波特率自动检测

CW32F020 使用 UART 作为从机进行通信时，可以通过自动波特率检测的方法，自动适应 UART 主机的波特率。可将通用定时器（GTIM）的输入捕获来源配置为 UART 的 RXD 信号，或者将 GTIM 的门控信号配置为 UART 的 RXD 信号，配合使用相关软件算法测量 UART 的波特率，以实现波特率自适应。详细请参考相关应用笔记。

17.3.3.3 发送控制

数据发送

设置 UARTx_CR1.TXEN 为 1 使能发送电路。将数据写入 UARTx_TDR 寄存器后，数据会被硬件转移到发送移位寄存器，发送移位寄存器将数据从最低位开始串行移出。具体发送流程的寄存器配置请参见 17.7 编程示例。

发送中断

在发送控制环节，存在 UARTx_ISR.TXE、UARTx_ISR.TC 和 UARTx_ISR.TXBUSY 三个标志位，其中，UARTx_ISR.TXE 和 UARTx_ISR.TC 可以产生中断请求。

当数据被硬件转移到发送移位寄存器后，发送缓冲器空中断标志位 UARTx_ISR.TXE 会被硬件置位，表示 UARTx_TDR 寄存器已空，此时允许对 UARTx_TDR 寄存器写入新的待发送数据。在对 UARTx_TDR 寄存器写入数据的同时，UARTx_ISR.TXE 标志位会被自动清零。如果此时发送移位寄存器中尚有未发送完成的数据，那么新写入的数据会在 UARTx_TDR 寄存器中暂存，在当前数据发送完成后，UARTx_TDR 寄存器中的数据会被硬件转移到发送移位寄存器中继续发送。

当发送移位寄存器中的数据帧发送完成后，发送完成中断标志位 UARTx_ISR.TC 会被硬件置位，表示已经发送完成一帧数据。

当发送移位寄存器中的数据帧发送完成后，如果 UARTx_TDR 寄存器中没有待发送的数据（即 UARTx_ISR.TXE = 1），UART 发送忙标志位 UARTx_ISR.TXBUSY 会被硬件清零，表示 UARTx_TDR 寄存器和发送移位寄存器中的数据已经发送完毕。

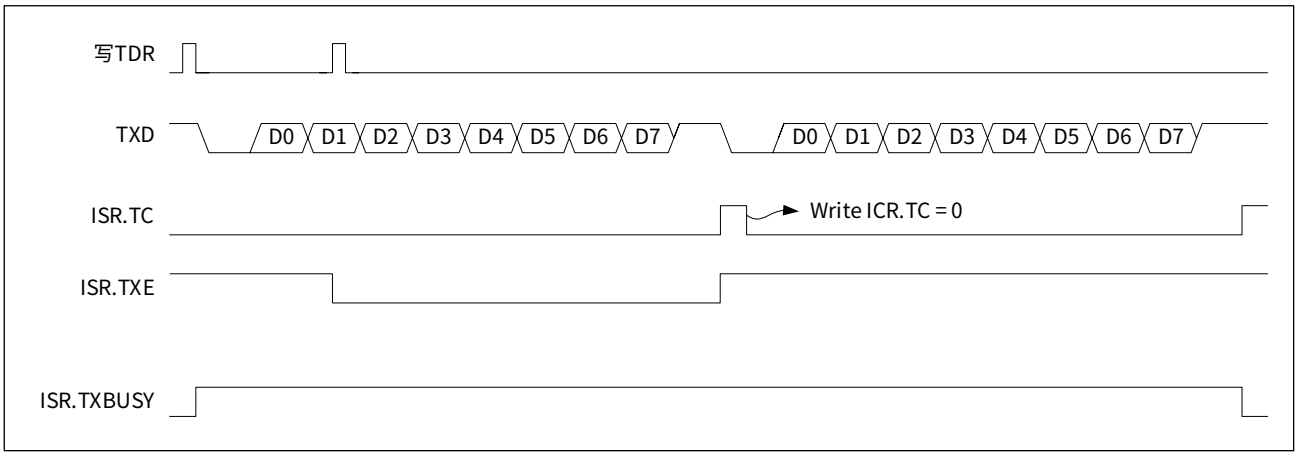
在用户应用中，在关闭 UART 之前，建议查询 UARTx_ISR.TXBUSY 标志位是否已经清零，确保数据已全部发送完成，避免数据丢失。

UARTx_ISR.TXE 标志位和 UARTx_ISR.TC 标志位置位是由硬件产生，均可以产生中断请求，分别由中断允许位 UARTx_IER.TXE 和 UARTx_IER.TC 控制是否产生。

在 UART 中断服务程序中，通过读取 UARTx_ISR.TXE 和 UARTx_ISR.TC 标志位判定中断源，退出中断服务程序之前应该清除对应的标志位，以避免重复进入中断服务程序。

UART 发送控制环节的时序图如下图所示：

图 17-4 异步工作模式发送控制时序（发送两个字节数据）



17.3.3.4 接收控制

数据接收

设置 UARTx_CR1.RXEN 为 1 使能接收电路。当接收器侦测到起始位后，开始接收数据，接收期间，数据从最低位开始串行移入接收移位寄存器，并行转移到 UARTx_RDR 寄存器，此时数据可以被读出。具体接收流程的寄存器配置请参见 17.7 编程示例。

接收中断

在接收控制环节，存在 UARTx_ISR.RC、UARTx_ISR.PE 和 UARTx_ISR.FE 三个标志位，且均可以产生中断请求。

当接收数据从移位寄存器转移到 UARTx_RDR 寄存器后，接收完成中断标志位 UARTx_ISR.RC 会被硬件置位，表示已经完成一帧数据的接收。用户程序中，一旦检测到 UARTx_ISR.RC 标志位为 1，应尽快读取 UARTx_RDR 寄存器，并清除 UARTx_ISR.RC 标志位。如果未及时读取 UARTx_RDR 寄存器，新接收的数据会覆盖 UARTx_RDR 寄存器中的数据，造成数据丢失。

在接收一个数据帧时，会自动进行奇偶校验，若检测到奇偶校验错误，UARTx_ISR.PE 标志位会被硬件置位，表示奇偶校验出错。

在接收一个数据帧时，如果在预期时间内未识别到正确的停止位，则判定发生帧结构错误，UARTx_ISR.FE 标志位会被硬件置位。

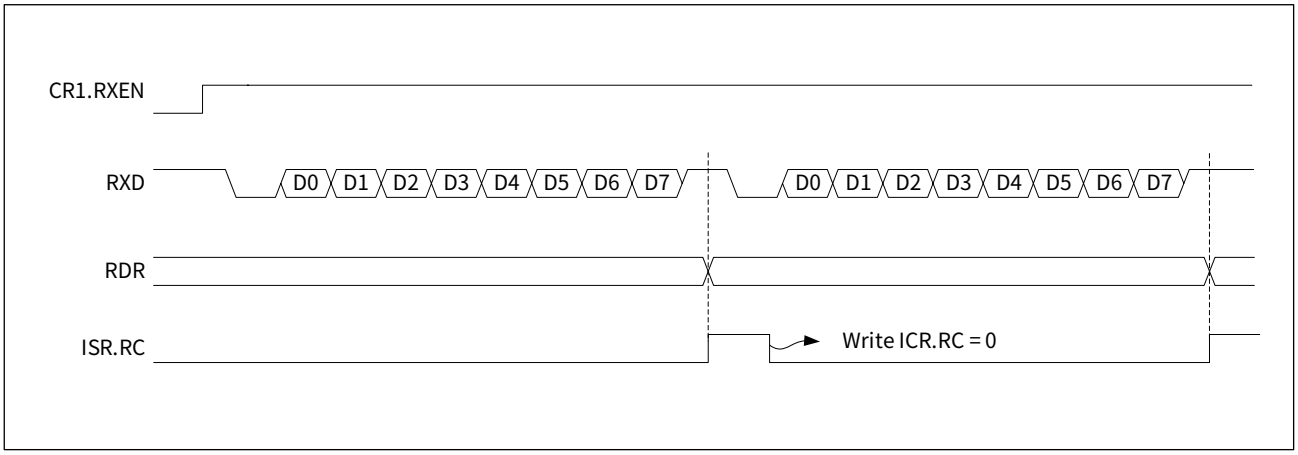
UARTx_ISR.RC、UARTx_ISR.PE 和 UARTx_ISR.FE 标志位置位是由硬件产生，均可以产生中断请求，分别由中断允许位 UARTx_IER.RC、UARTx_IER.PE 和 UARTx_IER.FE 控制是否产生。

在 UART 中断服务程序中，通过读取 UARTx_ISR.RC、UARTx_ISR.PE 和 UARTx_ISR.FE 标志位以判定中断源，退出中断服务程序之前应清除对应的标志位，以避免重复进入中断服务程序。

在用户应用中，在关闭 UART 之前，建议检查 UARTx_ISR.RC 标志位是否已经置 1，确保数据已全部接收完成，避免数据丢失。

UART 接收控制环节的时序图如下图所示：

图 17-5 异步工作模式接收控制时序（接收两个字节数据）



17.3.3.5 单线半双工模式

设置 UARTx_CR2.SIGNAL 为 1 使 UART 工作于单线半双工工作模式。在该模式下，使用 UARTx_TXD 引脚进行数据的发送和接收，不占用 UARTx_RXD 引脚（UARTx_RXD 可作通用 IO 使用），引脚具体配置参见表 17-1 UART 端口配置。

写数据到 UARTx_TDR 寄存器后，UARTx_TXD 引脚立即进入发送状态，输出 UARTx_TDR 寄存器中的数据。数据发送完成后，UARTx_TXD 引脚恢复到常态的接收状态。

没有发送数据时，UARTx_TXD 引脚处于接收状态，数据接收完成后，接收完成标志位 UARTx_ISR.RC 会被硬件置位，此时应尽快读取 UARTx_RDR 寄存器，并清除 UARTx_ISR.RC 标志位。

注：

用户应采取适当的应用层保护机制，以确保不会出现多主机同时向总线发送数据。

17.3.3.6 多机通信

UART 支持多机通信方式。在该模式下，UART 总线上有一个主机和多台从机，每个从机有唯一的从机地址，通信时主机会先发送地址帧对从机寻址，只有地址匹配的从机才被激活，接收随后主机发送的数据帧。

主机发送

多机通信模式下，主机需设置 UARTx_CR1.PARITY 为 1，配置为自定义校验，帧数据长度自动设置为 9 位。

UARTx_TDR 寄存器的最高位决定主机发送地址帧还是数据帧，UARTx_TDR[8] 为 1 表示主机发送的是地址帧，UARTx_TDR[8] 为 0 表示主机发送的是数据帧。

从机接收

多机通信模式下，从机需设置 UARTx_CR1.PARITY 为 1，配置为自定义校验，并设置 UARTx_CR2.ADDREN 为 1，使能从机地址识别，从机硬件自动检测主机发送的地址与本机地址是否匹配。

如果地址匹配，从机会将接收到的地址帧保存到 UARTx_RDR 寄存器中，UARTx_ISR.RC 标志位被硬件置位，同时 UARTx_ISR.MATCH 标志位被硬件置位，从机接收随后主机发送的数据帧。通信过程中，从机需

1. 应用程序在接收完成中断 RC 里查询 UARTx_RDR[8]，以判断接收到的是地址帧还是数据帧。
2. 从机在发送数据帧时，需要将 UARTx_TDR[8] 设置为 0，以避免被其它从机当作地址帧。

如果地址不匹配，从机不会接收主机发送的数据帧，也不产生接收完成中断，已置位的 UARTx_ISR.MATCH 标志位将被清零。

从机地址与地址掩码

从机地址由 UARTx_ADDR 寄存器配置，从机地址应配置为唯一。UARTx_MASK 寄存器是地址掩码，UARTx_MASK［7:0］中为‘1’的位对应的从机地址位参与从机地址匹配运算，为‘0’的位对应的从机地址位则不参与从机地址匹配运算。

当 UARTx_MASK［7:0］中全部位设置为‘1’时，即向 UARTx_MASK 寄存器写入 0xFF，则从机地址的 8 位地址位全部参与从机地址匹配，主机能唯一识别到从机。

当 UARTx_MASK［7:0］中部分位设置为‘0’时，对应的从机地址位不参与从机地址匹配，用以实现多个从机响应主机发出的同一地址帧，即主机对多个从机同时寻址。

例 1：

对从机 A 设置：UARTx_ADDR = 0xA0，UARTx_MASK = 0xFE

对从机 B 设置：UARTx_ADDR = 0xA1，UARTx_MASK = 0xFE

对从机 C 设置：UARTx_ADDR = 0xA2，UARTx_MASK = 0xFC

对从机 D 设置：UARTx_ADDR = 0xA3，UARTx_MASK = 0xFC

对从机 E 设置：UARTx_ADDR = 0xA5，UARTx_MASK = 0xFF

主机发送 0xA0 或 0xA1 地址帧时，可以寻址从机 A、从机 B、从机 C、从机 D

主机发送 0xA2 或 0xA3 地址帧时，可以寻址从机 C、从机 D

主机发送 0xA4 地址帧时，没有从机被寻址

主机发送 0xA5 地址帧时，可以寻址从机 E

广播地址

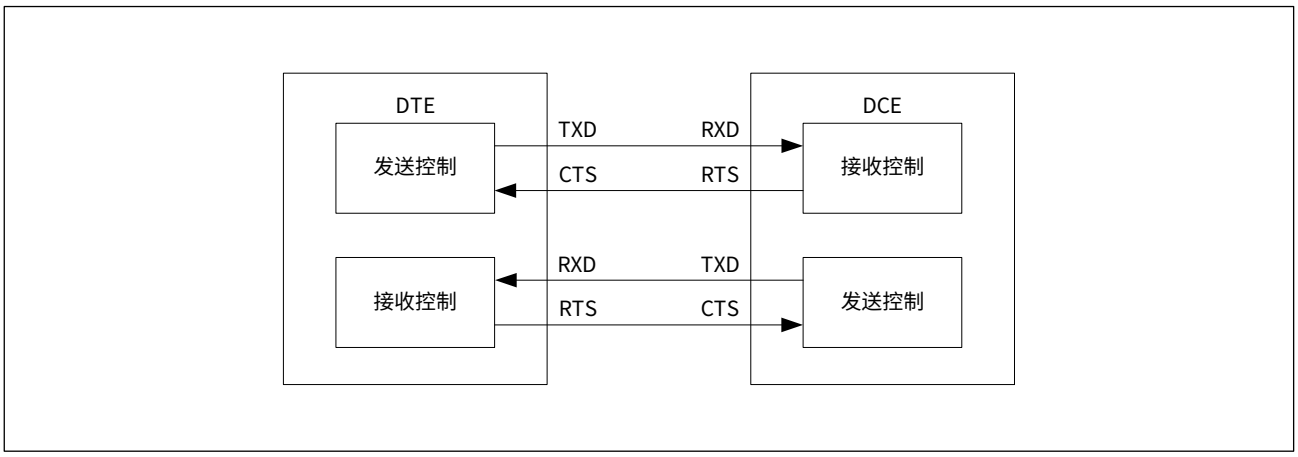
多机通信中，地址 0xFF 被定义为广播地址，主机发送广播地址帧时，所有地址的从机都被寻址。

17.3.3.7 硬件流控

CW32F020 支持 UART 硬件流控模式，即支持请求发送 RTS 和允许发送 CTS，用于自动控制数据的发送和接收。

数据通信系统中，数据终端设备 DTE 与数据电路终接设备 DCE 之间的通信硬件流控示意图如下图所示：

图 17-6 硬件流控示意图



请求发送 RTS

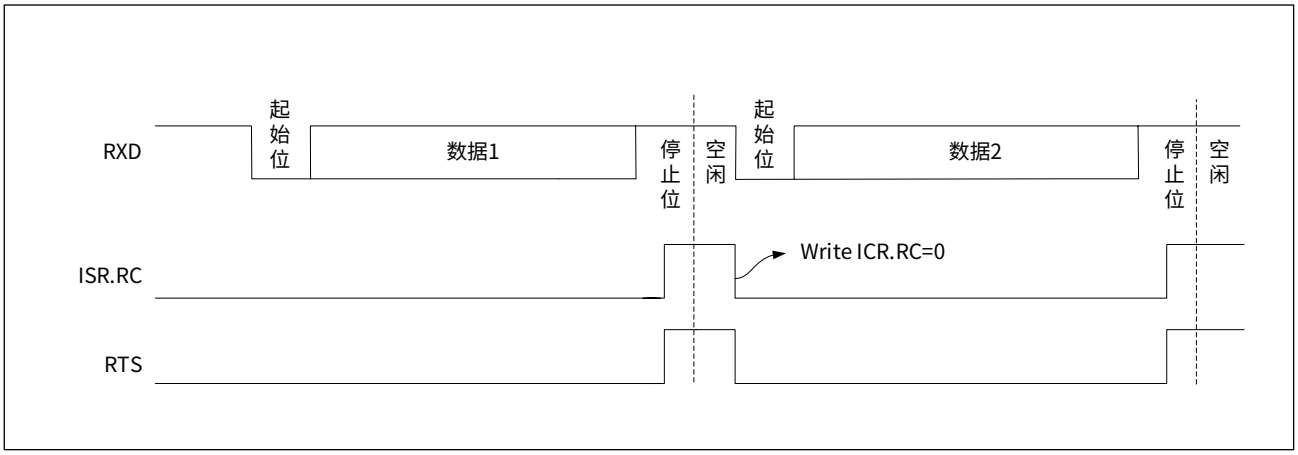
将 UARTx_CR2.RTSEN 置 1 使能 RTS 流控，RTS 表示请求发送，低电平有效。

当接收端准备好接收新的数据时（即 UARTx_ISR.RC = 0），RTS 引脚就会输出低电平，请求发送端发送一帧数据。

当接收端接收完成一帧数据时（即 UARTx_ISR.RC = 1），RTS 引脚就会输出高电平，通知发送端暂停发送。

用户应用中，读取 UARTx_RDR 寄存器中的数据后，如需继续接收下一帧数据，须软件将 UARTx_ISR.RC 标志位清零，此时 RTS 引脚将输出低电平，请求发送端发送下一帧数据。

图 17-7　RTS 流控时序



允许发送 CTS

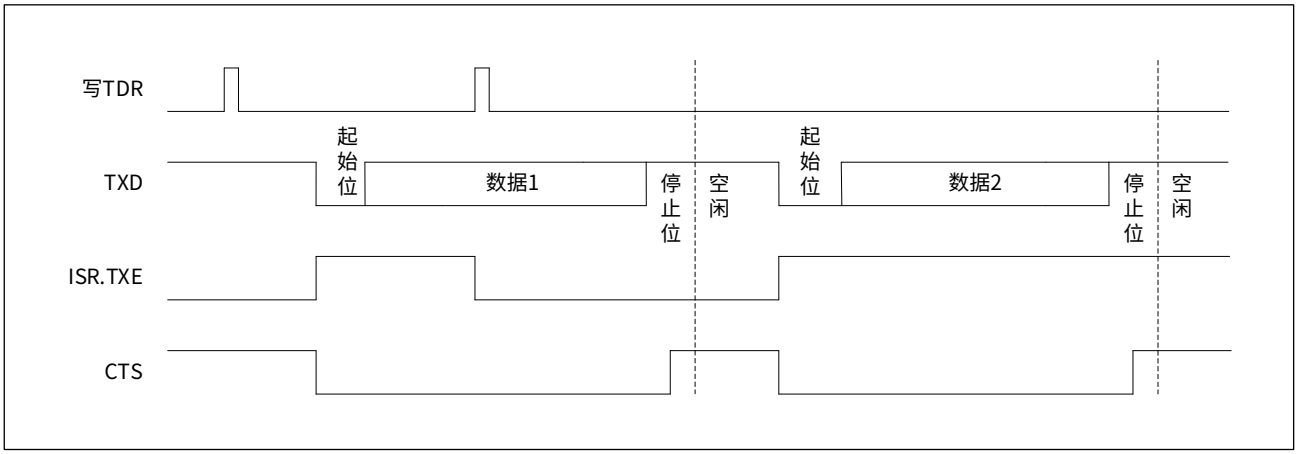
将 UARTx_CR.CTSEN 置 1 使能 CTS 流控，CTS 表示允许发送，低电平有效。

当发送端检测到 CTS 信号为低电平时，如果有待发送的数据（即 UARTx_ISR.TXE = 0），且当前没有正在发送的数据，则会发送这一帧数据。

当发送端检测到 CTS 信号为高电平时，如果有正在进行的数据发送，当前发送会继续进行，当前的发送完成后将停止发送。

在 CTS 流控过程中，CTS 信号电平标志位 UARTx_ISR.CTSLV 指示了当前 CTS 引脚的电平状态；CTS 信号变化标志位 UARTx_ISR.CTS 指示了 CTS 信号是否发生了变化，CTS 信号发生变化时，UARTx_ISR.CTS 标志位会被硬件置位，当 CTS 中断允许时（即设置 UARTx_IER.CTS 为 1），则会产生中断，写 UARTx_ICR.CTS 为 0 清除该中断标志。

图 17-8　CTS 流控时序



17.4 低功耗模式

UART 控制器工作在双时钟域下，支持在深度休眠模式下进行正常的数据收发，并通过接收完成中断唤醒 MCU 回到运行模式。

如果设置了传输时钟 UCLK 来源为低速时钟，当系统进入深度休眠模式后，高速时钟将停止，低速时钟保持运行，UART 仍可以进行正常的数据收发（波特率仅支持 2400 bps、4800 bps 和 9600 bps）。

要实现深度休眠模式下使用 UART 唤醒功能，需在进入深度休眠模式之前使能 UART 接收完成中断（即设置 UARTx_IER.RC 为 1），数据接收完成时，接收完成中断将唤醒 MCU 恢复到运行模式。

如果设置了传输时钟 UCLK 来源为高速时钟，当系统进入深度休眠模式后，高速时钟会停止运行，UART 不会接收数据。此时，仍可通过 GPIO 中断唤醒 MCU，实现在深度休眠模式下接收数据，参考配置步骤如下：

步骤 1：使能 UARTx_RXD 对应引脚的 GPIO 下降沿中断；

步骤 2：设置 UARTx_CR1.START 为 1，选择 RXD 信号起始位判定方式为低电平；

步骤 3：使能 UART 接收（即设置 UARTx_CR1.RXEN 为 1）；

步骤 4：进入深度休眠模式；

步骤 5：等待主机发送数据，产生 GPIO 下降沿中断，唤醒 MCU；

步骤 6：关闭 RXD 对应引脚的 GPIO 中断功能，等待 RXD 接收完成。

更多关于低功耗模式进入与唤醒的优化设计，请参见 3 电源控制（PWR）与功耗章节。

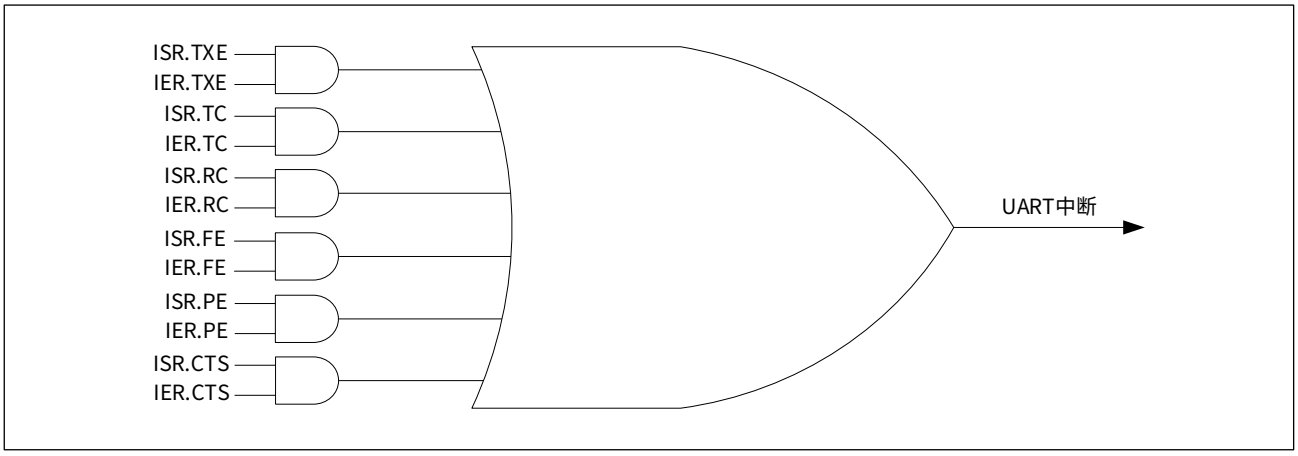
17.5 UART 中断

UART 控制器支持 6 个中断源，当 UART 中断触发事件发生时，中断标志位会被硬件置位，如果设置了对应的中断使能控制位，将产生中断请求。

CW32F020 的一个 UART 模块使用一个相同的系统 UART 中断，UART 中断是否产生中断跳转由嵌套向量中断控制器 (NVIC) 的中断使能设置寄存器 NVIC_ISER 的相应位控制。

系统 UART 中断示意图如下图所示：

图 17-9 UART 中断结构示意图



在用户 UART 中断服务程序中，应查询相关 UART 中断标志位，以进行相应的处理，在退出中断服务程序之前，要清除该中断标志位，避免重复进入中断程序。

各 UART 中断源的标志位、中断使能位、中断标志清除位或清除方法，如下表所示：

表 17-10 UART 中断控制

中断事件	中断标志位	中断使能位	标志清除方法
发送缓冲器空	ISR.TXE	IER.TXE	写 UARTx_TDR 寄存器
发送完成	ISR.TC	IER.TC	写 0 到 ICR.TC
接收完成	ISR.RC	IER.RC	写 0 到 ICR.RC
帧结构错误	ISR.FE	IER.FE	写 0 到 ICR.FE
奇偶校验错误	ISR.PE	IER.PE	写 0 到 ICR.PE
CTS 信号变化	ISR.CTS	IER.CTS	写 0 到 ICR.CTS

17.6 直接内存访问（DMA）

UART 控制器支持 DMA 传输功能，允许在 UART 收发数据寄存器和存储器之间进行高速数据传输，无须 CPU 干预。DMA 请求可以由软件或硬件触发，发送 TXD 和接收 RXD 的 DMA 请求是分别产生的。

DMA 发送

设置 UARTx_CR2.DMATX 为 1 使能 DMA 发送。当 UARTx_TDR 寄存器为空时，将触发 DMA 将一帧数据从指定的存储区传输到 UARTx_TDR 寄存器中，直到 DMA 控制器中配置的数据量全部传输完毕。

DMA 发送的具体寄存器配置流程请参见 8 直接内存访问（DMA）章节和 17.7 编程示例。

当 DMA 传输完成后，对应的 DMA 通道 y 传输完成标志位 DMA_ISR.TCy 会被硬件置位，如果允许中断（即设置 DMA_CSRy.TCIE 为 1），将产生中断请求。

在用户应用中，在关闭 UART 之前，建议查询 UARTx_ISR.TXBUSY 标志位是否已经清零，确保数据已全部发送完毕，避免破坏最后一次传输过程。

DMA 接收

设置 UARTx_CR2.DMARX 为 1 使能 DMA 接收。当收到一帧数据时（即 UARTx_RDR 寄存器非空），将触发 DMA 将数据从 UARTx_RDR 寄存器传输到指定的存储区，直到 DMA 控制器中配置的数据量全部传输完毕。

DMA 接收的具体寄存器配置流程请参见 8 直接内存访问（DMA）章节和 17.7 编程示例。

当 DMA 传输完成后，对应的 DMA 通道 y 传输完成标志位 DMA_ISR.TCy 会被硬件置位，如果允许中断（即设置 DMA_CSRy.TCIE 为 1），将产生中断请求。

17.7 编程示例

17.7.1 异步全双工编程示例

17.7.1.1 查询方式发送数据

步骤 1：设置 SYSCTRL_AHBEN.GPIOx 为 1，SYSCTRL_APBENx.UARTx 为 1，使能 UART 引脚对应的 GPIO 时钟和 UART 配置时钟；

步骤 2：将 UARTx_TXD 引脚配置成推挽复用输出模式，具体寄存器配置步骤请参见 9 通用输入输出端口（GPIO）章节；

步骤 3：设置 UARTx_CR1.SYNC 为 0，配置 UARTx 为异步全双工通信模式；

步骤 4：配置数据帧；

1. 起始位判定方式：配置 UARTx_CR1.START
2. 校验位：配置 UARTx_CR1.PARITY
3. 停止位：配置 UARTx_CR1.STOP

步骤 5：配置 UARTx_CR2.SOURCE，选择传输时钟源；

步骤 6：配置 UARTx_CR1.OVER，选择采样模式；

步骤 7：配置 UARTx_BRRI 和 UARTx_BRRF 寄存器，配置波特率，具体配置请参见 17.3.3.2 小数波特率发生器；

步骤 8：设置 UARTx_CR1.TXEN 为 1 使能发送；

步骤 9：设置 UARTx_ICR.TC 为 0，清除发送完成标志位；

步骤 10：将要发送的一帧数据写入 UARTx_TDR 寄存器；

步骤 11：查询等待 UARTx_ISR.TC 标志位置 1，确认一帧数据发送完成；

步骤 12：重复步骤 9 至步骤 12，发送下一帧数据。

17.7.1.2 查询方式接收数据

步骤 1：设置 SYSCTRL_AHBEN.GPIOx 为 1，SYSCTRL_APBENx.UARTx 为 1，使能 UART 引脚对应的 GPIO 时钟和 UART 配置时钟；

步骤 2：将 RXD 引脚配置成上拉输入复用模式，具体寄存器配置步骤请参见 9 通用输入输出端口（GPIO）章节；

步骤 3：设置 UARTx_CR1.SYNC 为 0，配置 UARTx 为异步全双工通信模式；

步骤 4：配置数据帧；

1. 起始位判定方式：配置 UARTx_CR1.START
2. 校验位：配置 UARTx_CR1.PARITY
3. 停止位：配置 UARTx_CR1.STOP

步骤 5：配置 UARTx_CR2.SOURCE，选择传输时钟源；

步骤 6：配置 UARTx_CR1.OVER，选择采样模式；

步骤 7：配置 UARTx_BRRI 和 UARTx_BRRF 寄存器，配置波特率，具体配置请参见 17.3.3.2 小数波特率发生器；

步骤 8：向 UARTx_ICR 寄存器写入 0x00，清除所有标志位；

步骤 9：设置 UARTx_CR1.RXEN 为 1 使能接收；

步骤 10：查询等待 UARTx_ISR.RC 标志位置 1，确认接收完一帧数据；

步骤 11：查询错误标志 UARTx_ISR.PE 和 UARTx_ISR.FE，确认数据是否有效，如果数据无效，则进行出错处理，如果数据有效，则读取 UARTx_RDR 寄存器并保存数据；

步骤 12：设置 UARTx_ICR.RC 为 0，清除接收完成标志位；

步骤 13：重复步骤 10 至步骤 13，接收下一帧数据。

17.7.1.3 中断方式发送数据

步骤 1：设置 SYSCTRL_AHBEN.GPIOx 为 1，SYSCTRL_APBENx.UARTx 为 1，使能 UART 引脚对应的 GPIO 时钟和 UART 配置时钟；

步骤 2：将 TXD 引脚配置成推挽复用输出模式，具体寄存器配置步骤请参见 9 通用输入输出端口（GPIO）章节；

步骤 3：设置 UARTx_CR1.SYNC 为 0，配置 UARTx 为异步全双工通信模式；

步骤 4：配置数据帧；

1. 起始位判定方式：配置 UARTx_CR1.START
2. 校验位：配置 UARTx_CR1.PARITY
3. 停止位：配置 UARTx_CR1.STOP

步骤 5：配置 UARTx_CR2.SOURCE，选择传输时钟源；

步骤 6：配置 UARTx_CR1.OVER，选择采样模式；

步骤 7：配置 UARTx_BRRI 和 UARTx_BRRF 寄存器，配置波特率，具体配置请参见 17.3.3.2 小数波特率发生器；

步骤 8：配置 NVIC 控制器，请参见 5 中断章节；

步骤 9：设置 UARTx_IER.TXE 为 1，使能发送缓冲器空中断；

步骤 10：设置 UARTx_CR1.TXEN 为 1 使能发送；

步骤 11：将要发送的一帧数据写入 UARTx_TDR 寄存器；

步骤 12：数据开始发送，发送缓冲器空，进入中断服务函数：查询判断 UARTx_ISR.TXE 标志位，如果标志位为 1，写一帧新的数据到 UARTx_TDR 寄存器；

步骤 13：查询等待 UARTx_ISR.TXBUSY 标志位清零，关闭 UART。

17.7.1.4 中断方式接收数据

步骤 1：设置 SYSCTRL_AHBEN.GPIOx 为 1，SYSCTRL_APBENx.UARTx 为 1，使能 UART 引脚对应的 GPIO 时钟和 UART 配置时钟；

步骤 2：将 RXD 引脚配置成上拉输入复用模式，具体寄存器配置步骤请参见 9 通用输入输出端口（GPIO）章节；

步骤 3：设置 UARTx_CR1.SYNC 为 0，配置 UARTx 为异步全双工通信模式；

步骤 4：配置数据帧；

1. 起始位判定方式：配置 UARTx_CR1.START
2. 校验位：配置 UARTx_CR1.PARITY
3. 停止位：配置 UARTx_CR1.STOP

步骤 5：配置 UARTx_CR2.SOURCE，选择传输时钟源；

步骤 6：配置 UARTx_CR1.OVER，选择采样模式；

步骤 7：配置 UARTx_BRRI 和 UARTx_BRRF 寄存器，配置波特率，具体配置请参见 17.3.3.2 小数波特率发生器；

步骤 8：配置 NVIC 控制器，请参见 5 中断章节；

步骤 9：设置 UARTx_IER.RC 为 1 使能接收完成中断；

步骤 10：设置 UARTx_CR1.RXEN 为 1 使能接收；

步骤 11：等待一帧数据接收完成，进入中断服务函数：查询 UARTx_ISR.PE 和 UARTx_ISR.FE 标志位，确认数据是否有效，如果无效，则进行出错处理；如果有效，查询判断 UARTx_ISR.RC 标志位，如果标志位为 1，则读取 UARTx_RDR 寄存器并保存数据，并清除该标志位。

17.7.1.5 DMA 发送数据

以下是 UART1（UART1_TXD 引脚为 PA08）的 DMA 通道 1 发送数据示例。在这个应用示例中，设置传输时钟 UCLK 为 24MHz，波特率为 115200bps，DMA 数据传输量为 100，通过硬件触发启动 DMA 传输，将在 SRAM 区定义的一个 8 位数组中的数据传输到 UART1_TDR 寄存器。

具体配置流程如下：

步骤 1：设 置 SYSCTRL_AHBEN.GPIOA 为 1，SYSCTRL_APBEN2.UART1 为 1，SYSCTRL_AHBEN.DMA 为 1， 分别使能 GPIOA、UART1 和 DMA 的配置时钟及工作时钟；

步骤 2：将 UART1_TXD 引脚（即 PA08）配置成推挽复用输出模式，具体寄存器配置步骤请参见 9 通用输入输出端口（GPIO）章节；

步骤 3：设置 UART1_CR1.SYNC 为 0，配置 UART1 为异步全双工通信模式；

步骤 4：配置数据帧；

1. 起始位判定方式：设置 UART1_CR1.START 为 0，RXD 信号下降沿
2. 校验位：设置 UART1_CR1.PARITY 为 0，无奇偶校验
3. 停止位：配置 UART1_CR1.STOP 为 0，1 位长度的停止位

步骤 5：设置 UART1_CR2.SOURCE 为 0，选择传输时钟源为 PCLK = 24MHz；

步骤 6：设置 UART1_CR1.OVER 为 0，选择 16 倍采样模式；

步骤 7：设置 UART1_BRRI.BRRI 为 0x0D，UART1_BRRF.BRRF 为 0x00，配置波特率为 115200bps；

步骤 8：设置 UART1_CR1.TXEN 为 1 使能发送；

步骤 9：定义一个变量 uint8_t SendBuff[100]，并赋值；

步骤 10：向 UART1_ICR 写入 0x00，清除 UART1 所有标志位；

步骤 11：设置 DMA_ICR.TC1 和 DMA_ICR.TE1 为 0，清除 DMA 通道 1 所有中断标志位；

步骤 12：设置 DMA_SRCADDR1 = (uint32_t) SendBuff，源地址为 SendBuff；

步骤 13：设置 DMA_DSTADDR1 = (uint32_t) &UART1_TDR，目的地址为 UART1_TDR 寄存器；

步骤 14：设置 DMA_CSR1.SRCINC 为 1，源地址自增；

步骤 15：设置 DMA_CSR1.DSTINC 为 0，目的地址固定；

步骤 16：向 DMA_CNT1 写入 0x10064，数据传输量为 100；

步骤 17：设置 DMA_CSR1.SIZE 为 0，传输数据位宽为 8 位；

步骤 18：设置 DMA_CSR1.TRANS 为 1，块传输（BLOCK）；

步骤 19：设置 DMA_TRIG1.Type 为 1，硬件触发模式；

步骤 20：设置 DMA_TRIG1. HardSrc 为 0x01，UART1_TDR 寄存器为空时触发 DMA 传输；

步骤 21：设置 DMA_CSR1.EN 为 1，使能 DMA 通道 1 传输；

步骤 22：设置 UART1_CR2.DMATX 为 1，打开 UART1 TXD 的 DMA 硬件握手逻辑；

步骤 23：查询等待 DMA_ISR.TC1 变为 1，确认 DMA 传输完成；

步骤 24：设置 UART1_CR2.DMATX 为 0，关闭 UART1 TXD 的 DMA 硬件握手逻辑；

步骤 25：查询等待 UART1_ISR.TXBUSY 变为 0，确认数据已通过 UART1 发送完成；

步骤 26：如果要传输新的数据，重复步骤 9 至步骤 26。

17.7.1.6 DMA 接收数据

以下是 UART1（UART1_RXD 引脚为 PA09）的 DMA 通道 2 接收数据示例。在这个应用示例中，设置传输时钟 UCLK 为 24MHz，波特率为 115200bps，DMA 数据传输量为 100，通过硬件触发启动 DMA 传输，将 UART1_RDR 寄存器中的数据传输到 SRAM 区中的一个 8 位数组中。

具体配置流程如下：

步骤 1：设置 SYSCTRL_AHBEN.GPIOA 为 1，SYSCTRL_APBEN2.UART1 为 1，SYSCTRL_AHBEN.DMA 为 1，分别使能 GPIOA、UART1 和 DMA 的配置时钟及工作时钟；

步骤 2：将 UART1_RXD 引脚（即 PA09）配置成上拉复用输入模式，具体寄存器配置步骤请参见 9 通用输入输出端口（GPIO）章节；

步骤 3：设置 UART1_CR1.SYNC 为 0，配置 UART1 为异步全双工通信模式；

步骤 4：配置数据帧；

1. 起始位判定方式：设置 UART1_CR1.START 为 0，RXD 信号下降沿
2. 校验位：设置 UART1_CR1.PARITY 为 0，无奇偶校验
3. 停止位：配置 UART1_CR1.STOP 为 0，1 位长度的停止位

步骤 5：设置 UART1_CR2.SOURCE 为 0，选择传输时钟源为 PCLK = 24MHz；

步骤 6：设置 UART1_CR1.OVER 为 0，选择 16 倍采样模式；

步骤 7：设置 UART1_BRRI.BRRI 为 0x0D，UART1_BRRF.BRRF 为 0x00，配置波特率为 115200bps；

步骤 8：设置 UART1_CR1.RXEN 为 1 使能接收；

步骤 9：定义一个变量 uint8_t RecvBuff[100]；

步骤 10：向 UART1_ICR 写入 0x00，清除 UART1 所有标志位；

步骤 11：设置 DMA_ICR.TC2 和 DMA_ICR.TE2 为 0，清除 DMA 通道 2 所有中断标志位；

步骤 12：设置 DMA_SRCADDR2 = (uint32_t) &UART1_RDR，源地址为 UART1_RDR 寄存器；

步骤 13：设置 DMA_DSTADDR2 = (uint32_t) RecvBuff，目的地址为 RecvBuff；

步骤 14：设置 DMA_CSR2.SRCINC 为 0，源地址固定；

步骤 15：设置 DMA_CSR2.DSTINC 为 1，目的地址自增；

步骤 16：向 DMA_CNT2 写入 0x10064，数据传输量为 100；

步骤 17：设置 DMA_CSR2.SIZE 为 0，传输数据位宽为 8 位；

步骤 18：设置 DMA_CSR2.TRANS 为 1，块传输（BLOCK）；

步骤 19：设置 DMA_TRIG2.Type 为 1，硬件触发模式；

步骤 20：设置 DMA_TRIG2. HardSrc 为 0x00，UART1_RDR 寄存器非空时触发 DMA 传输；

步骤 21：设置 DMA_CSR2.EN 为 1，使能 DMA 通道 2 传输；

步骤 22：设置 UART1_CR2.DMARX 为 1，打开 UART1 RXD 的 DMA 硬件握手逻辑；

步骤 23：查询等待 DMA_ISR.TC2 变为 1，确认 DMA 传输完成；

步骤 24：设置 UART1_CR2.DMARX 为 0，关闭 UART1 RXD 的 DMA 硬件握手逻辑；

步骤 25：如果要传输新的数据，重复步骤 10 至步骤 25。

17.7.2 同步半双工编程示例

17.7.2.1 查询方式发送数据

步骤 1：设置 SYSCTRL_AHBEN.GPIOx 为 1，SYSCTRL_APBENx.UARTx 为 1，使能 GPIO 时钟和 UART 配置时钟；

步骤 2：将 UARTx_TXD、UARTx_RXD 引脚配置成推挽复用输出模式，从机片选引脚 NCS 配置成推挽输出模式，具体寄存器配置步骤请参见 9 通用输入输出端口（GPIO）章节；

步骤 3：设置 UARTx_CR1.SYNC 为 1，配置 UARTx 为同步半双工通信模式；

步骤 4：配置 UARTx_CR2.SOURCE，选择传输时钟源，配置波特率，具体配置请参见 17.3.2 同步模式；

步骤 5：拉低 NCS 信号，选择某一个从机；

步骤 6：设置 UARTx_CR1.TXEN 为 1 使能发送；

步骤 7：设置 UARTx_ICR.TC 为 0，清除发送完成标志位；

步骤 8：将要发送的 8 位数据写入 UARTx_TDR 寄存器；

步骤 9：查询等待 UARTx_ISR.TC 标志位置 1，确认数据发送完成；

步骤 10：重复步骤 7 至步骤 10，发送下一帧数据；

步骤 11：拉高 NCS 信号，结束通信。

17.7.2.2 查询方式接收数据

步骤 1：设置 SYSCTRL_AHBEN.GPIOx 为 1，SYSCTRL_APBENx.UARTx 为 1，使能 GPIO 时钟和 UART 配置时钟；

步骤 2：将 UARTx_TXD、UARTx_RXD 引脚配置成推挽复用输出模式，从机片选引脚 NCS 配置成推挽输出模式，具体寄存器配置步骤请参见 9 通用输入输出端口（GPIO）章节；

步骤 3：设置 UARTx_CR1.SYNC 为 1，配置 UARTx 为同步半双工通信模式；

步骤 4：配置 UARTx_CR2.SOURCE，选择传输时钟源，配置波特率，具体配置请参见 17.3.2 同步模式；

步骤 5：拉低 NCS 信号，选择某一个从机；

步骤 6：向 UARTx_ICR 写入 0x00，清除所有标志位；

步骤 7：设置 UARTx_CR1.RXEN 为 1 使能接收；

步骤 8：查询等待 UARTx_ISR.RC 标志位置 1，确认数据接收完成；

步骤 9：读 UARTx_RDR 寄存器并保存数据；

步骤 10：设置 UARTx_ICR.RC 为 0，清除接收完成标志位；

步骤 11：重复步骤 8 至步骤 11，接收下一个 8 位数据；

步骤 12：拉高 NCS 信号，结束通信。

17.7.3 单线半双工编程示例

17.7.3.1 查询方式发送数据

步骤 1：设置 SYSCTRL_AHBEN.GPIOx 为 1，SYSCTRL_APBENx.UARTx 为 1，使能 UART 引脚对应的 GPIO 时钟和 UART 配置时钟；

步骤 2：将 UARTx_TXD 引脚配置成开漏复用输出模式，并外接上拉电阻，具体寄存器配置步骤请参见 9 通用输入输出端口（GPIO）章节；

步骤 3：设置 UARTx_CR2.SIGNAL 为 1，配置 UARTx 为单线半双工通信模式；

步骤 4：配置数据帧；

1. 起始位判定方式：配置 UARTx_CR1.START
2. 校验位：配置 UARTx_CR1.PARITY
3. 停止位：配置 UARTx_CR1.STOP

步骤 5：配置 UARTx_CR2.SOURCE，选择传输时钟源；

步骤 6：配置 UARTx_CR1.OVER，选择采样模式；

步骤 7：配置 UARTx_BRRI 和 UARTx_BRRF 寄存器，配置波特率，具体配置请参见 17.3.3.2 小数波特率发生器；

步骤 8：设置 UARTx_CR1.TXEN 为 1，UARTx_CR1.RXEN 为 1 使能发送和接收；

步骤 9：设置 UARTx_ICR.TC 为 0，清除发送完成标志位；

步骤 10：将要发送的一帧数据写入 UARTx_TDR 寄存器；

步骤 11：查询等待 UARTx_ISR.TC 标志位置 1，确认数据发送完成；

步骤 12：重复步骤 9 至步骤 12，发送下一帧数据。

17.7.3.2 查询方式接收数据

步骤 1：设置 SYSCTRL_AHBEN.GPIOx 为 1，SYSCTRL_APBENx.UARTx 为 1，使能 UART 引脚对应的 GPIO 时钟和 UART 配置时钟；

步骤 2：将 UARTx_TXD 引脚配置成开漏复用输出模式，并外接上拉电阻，具体寄存器配置步骤请参见 9 通用输入输出端口（GPIO）章节；

步骤 3：设置 UARTx_CR2.SIGNAL 为 1，配置 UARTx 为单线半双工通信模式；

步骤 4：配置数据帧；

1. 起始位判定方式：配置 UARTx_CR1.START
2. 校验位：配置 UARTx_CR1.PARITY
3. 停止位：配置 UARTx_CR1.STOP

步骤 5：配置 UARTx_CR2.SOURCE，选择传输时钟源；

步骤 6：配置 UARTx_CR1.OVER，选择采样模式；

步骤 7：配置 UARTx_BRRI 和 UARTx_BRRF 寄存器，配置波特率，具体配置请参见 17.3.3.2 小数波特率发生器；

步骤 8：向 UARTx_ICR 写入 0x00，清除所有标志位；

步骤 9：设置 UARTx_CR1.TXEN 为 1，UARTx_CR1.RXEN 为 1 使能发送和接收；

步骤 10：查询等待 UARTx_ISR.RC 标志位置 1，确认接收完一帧数据；

步骤 11：查询错误标志 UARTx_ISR.PE 和 UARTx_ISR.FE，确认数据是否有效，如果数据无效，则进行出错处理，如果数据有效，则读取 UARTx_RDR 寄存器并保存数据；

步骤 12：设置 UARTx_ICR.RC 为 0，清除接收完成标志位；

步骤 13：重复步骤 10 至步骤 13，接收下一帧数据。

17.8 寄存器列表

UART1 基地址：UART1_BASE = 0x4001 3800

UART2 基地址：UART2_BASE = 0x4000 4400

UART3 基地址：UART3_BASE = 0x4000 4800

表 17-11 UART 寄存器列表

寄存器名称	寄存器地址	寄存器描述
UARTx_CR1	UARTx_BASE + 0x00	控制寄存器 1
UARTx_CR2	UARTx _BASE + 0x04	控制寄存器 2
UARTx_IER	UARTx_BASE + 0x08	中断使能寄存器
UARTx_BRRI	UARTx_BASE + 0x0C	波特率计数器整数部分寄存器
UARTx_BRRF	UARTx_BASE + 0x10	波特率计数器小数部分寄存器
UARTx_ISR	UARTx_BASE + 0x1C	中断标志寄存器
UARTx_ICR	UARTx_BASE + 0x20	中断标志清除寄存器
UARTx_RDR	UARTx_BASE + 0x24	接收数据寄存器
UARTx_TDR	UARTx_BASE + 0x28	发送数据寄存器
UARTx_ADDR	UARTx_BASE + 0x30	从机地址寄存器
UARTx_MASK	UARTx_BASE + 0x34	从机地址掩码寄存器

17.9 寄存器描述

有关寄存器描述里所使用的缩写，请参见 1 文档约定章节。

17.9.1 UARTx_CR1 控制寄存器 1

Address offset: 0x00 Reset value: 0x0000 0000

位域	名称	权限	功能描述
31:11	RFU	-	保留位，请保持默认值
10:9	OVER	RW	采样方式设置 00：16 倍采样， Baud = UCLK / (BRRI×16 + BRRF) 01：8 倍采样， Baud = UCLK / (BRRI×8) 10：4 倍采样， Baud = UCLK / (BRRI×4) 11：专用采样， Baud = UCLK×256 / BRRI 注 1：专用采样仅适合传输时钟为 LSI/LSE 时进行 2400、4800、9600 通信 注 2：当 BRRF 为非零值，自动采用 16 倍采样
8	START	RW	RXD 信号 START 位判定方式设置 0：RXD 信号下降沿 1：RXD 信号低电平
7	RFU	-	保留位，请保持默认值
6	SYNC	RW	传输同步异步模式设置 0：工作于异步模式 1：工作于同步模式
5:4	STOP	RW	停止位长度设置 00：1 位 01：1.5 位 10：2 位 11：保留 注：同步模式下 STOP 位必须清零
3:2	PARITY	RW	校验位设置 00：无奇偶校验 01：自定义校验 10：偶校验 11：奇校验 注 1：无奇偶校验时，数据字长为 8 位，有校验位时，数据字长为 9 位； 注 2：多机通信模式下，主机和从机都配置为自定义校验。
1	RXEN	RW	接收电路使能控制 0：禁止 1：使能
0	TXEN	RW	发送电路使能控制 0：禁止 1：使能

17.9.2 UARTx_CR2 控制寄存器 2

Address offset: 0x04 Reset value: 0x0000 0000

位域	名称	权限	功能描述
31:10	RFU	-	保留位，请保持默认值
9:8	SOURCE	RW	UART 传输时钟来源配置 00：UCLK 来自 PCLK 01：UCLK 来自 PCLK 10：UCLK 来自 LSE 11：UCLK 来自 LSI
7	DMATX	RW	DMA 发送数据使能控制 0：禁止 1：使能
6	DMARX	RW	DMA 接收数据使能控制 0：禁止 1：使能
5	TXINV	RW	TXD 引脚输出信号反相控制 0：TXD 引脚输出正相信号 1：TXD 引脚输出反相信号
4	RXINV	RW	RXD 引脚输入信号反相控制 0：RXD 引脚信号直接送入接收电路 1：RXD 引脚信号反相送入接收电路
3	RTSEN	RW	RTS 硬件信号使能控制 0：禁止 1：使能
2	CTSEN	RW	CTS 硬件信号使能控制 0：禁止 1：使能
1	SIGNAL	RW	单总线模式使能控制 0：禁止，通过 TXD 端口发送数据，通过 RXD 端口接收数据 1：使能，通过 TXD 端口进行数据收发
0	ADDREN	RW	从机地址识别使能控制 0：禁止 1：使能

17.9.3 UARTx_BRRI 波特率计数器整数部分寄存器

Address offset: 0x0C Reset value: 0x0000 0000

位域	名称	权限	功能描述
31:16	RFU	-	保留位，请保持默认值
15:0	BRRI	RW	波特率计数器整数部分

17.9.4 UARTx_BRRF 波特率计数器小数部分寄存器

Address offset: 0x10 Reset value: 0x0000 0000

位域	名称	权限	功能描述
31:4	RFU	-	保留位，请保持默认值
3:0	BRRF	RW	波特率计数器小数部分

17.9.5 UARTx_TDR 发送数据寄存器

Address offset: 0x28 Reset value: 0x0000 0000

位域	名称	权限	功能描述
31:9	RFU	-	保留位，请保持默认值
8:0	TDR	WO	待发送的数据 注：当 UARTx_CR1.PARITY 设置为自定义校验时，TXD 引脚会发送 UARTx_TDR[8]

17.9.6 UARTx_RDR 接收数据寄存器

Address offset: 0x24 Reset value: 0x0000 0000

位域	名称	权限	功能描述
31:9	RFU	-	保留位，请保持默认值
8:0	RDR	RO	已接收的数据 注：当 UARTx_CR1.PARITY 设置为自定义校验时， 接收电路会接收 UARTx_RDR[8]

17.9.7 UARTx_IER 中断使能寄存器

Address offset: 0x08 Reset value: 0x0000 0000

位域	名称	权限	功能描述
31:7	RFU	-	保留位，请保持默认值
6	CTS	RW	CTS 信号变化中断使能控制 0：禁止 1：使能
5	RFU	-	保留位，请保持默认值
4	PE	RW	奇偶校验错误中断使能控制 0：禁止 1：使能
3	FE	RW	帧结构错误中断使能控制 0：禁止 1：使能
2	RC	RW	接收完成中断使能控制 0：禁止 1：使能
1	TC	RW	发送完成中断使能控制 0：禁止 1：使能
0	TXE	RW	发送缓冲器空中断使能控制 0：禁止 1：使能

17.9.8 UARTx_ISR 中断标志寄存器

Address offset: 0x1C Reset value: 0x0000 0001

位域	名称	权限	功能描述
31:9	RFU	-	保留位，请保持默认值
8	TXBUSY	RO	TXD 发送状态 0:TDR 寄存器及发送移位寄存器空 1:TDR 寄存器或发送移位寄存器非空
7	CTSLV	RO	CTS 信号电平状态 0：CTS 信号为低电平 1：CTS 信号为高电平
6	CTS	RO	CTS 信号变化中断标志 0：未检测到 CTS 电平变化 1：已检测到 CTS 电平变化
5	MATCH	RO	从机地址匹配标志 0：未检测到匹配的地址 1：已检测到匹配的地址
4	PE	RO	奇偶校验错误中断标志 0：未检测到奇偶校验错误 1：已检测到奇偶校验错误
3	FE	RO	帧结构错误中断标志 0：未检测到帧结构错误 1：已检测到帧结构错误
2	RC	RO	接收完成中断标志 0：尚未完成一帧数据的接收 1：已经完成一帧数据的接收
1	TC	RO	发送完成中断标志 发送移位寄存器发送完成一帧数据就被硬件置位。 0：尚未完成一帧数据的发送 1：已经完成一帧数据的发送
0	TXE	RO	发送缓冲器空标志 0：发送缓冲器内有数据 1：发送缓冲器内无数据 注：写 UARTx_TDR 寄存器将清除该标志位

17.9.9 UARTx_ICR 中断标志清除寄存器

Address offset: 0x20 Reset value: 0x0000 00FF

位域	名称	权限	功能描述
31:7	RFU	-	保留位，请保持默认值
6	CTS	R1W0	CTS 信号变化中断标志清除控制 W0：清除 CTS 信号变化中断标志 W1：无功能
5	RFU	-	保留位，请保持默认值
4	PE	R1W0	奇偶校验错误中断标志清除控制 W0：清除奇偶校验错误中断标志 W1：无功能
3	FE	R1W0	帧结构错误中断标志清除控制 W0：清除帧结构错误中断标志 W1：无功能
2	RC	R1W0	接收完成中断标志清除控制 W0：清除接收完成中断标志 W1：无功能
1	TC	R1W0	发送完成中断标志清除控制 W0：清除发送完成中断标志 W1：无功能
0	RFU	-	保留位，请保持默认值

17.9.10 UARTx_ADDR 从机地址寄存器

Address offset: 0x30 Reset value: 0x0000 0000

位域	名称	权限	功能描述
31:8	RFU	-	保留位，请保持默认值
7:0	ADDR	RW	从机地址寄存器

17.9.11 UARTx_MASK 从机地址掩码寄存器

Address offset: 0x34 Reset value: 0x0000 0000

位域	名称	权限	功能描述
31:8	RFU	-	保留位，请保持默认值
7:0	MASK	RW	从机地址掩码寄存器

18 串行外设接口（SPI）

18.1 概述

串行外设接口（SPI）是一种同步串行数据通信接口，常用于 MCU 与外部设备之间进行同步串行通信。CW32F020 内部集成 2 个串行外设 SPI 接口，支持双向全双工、单线半双工和单工通信模式，可配置 MCU 作为主机或从机，支持多主机通信模式，支持直接内存访问（DMA）。

18.2 主要特性

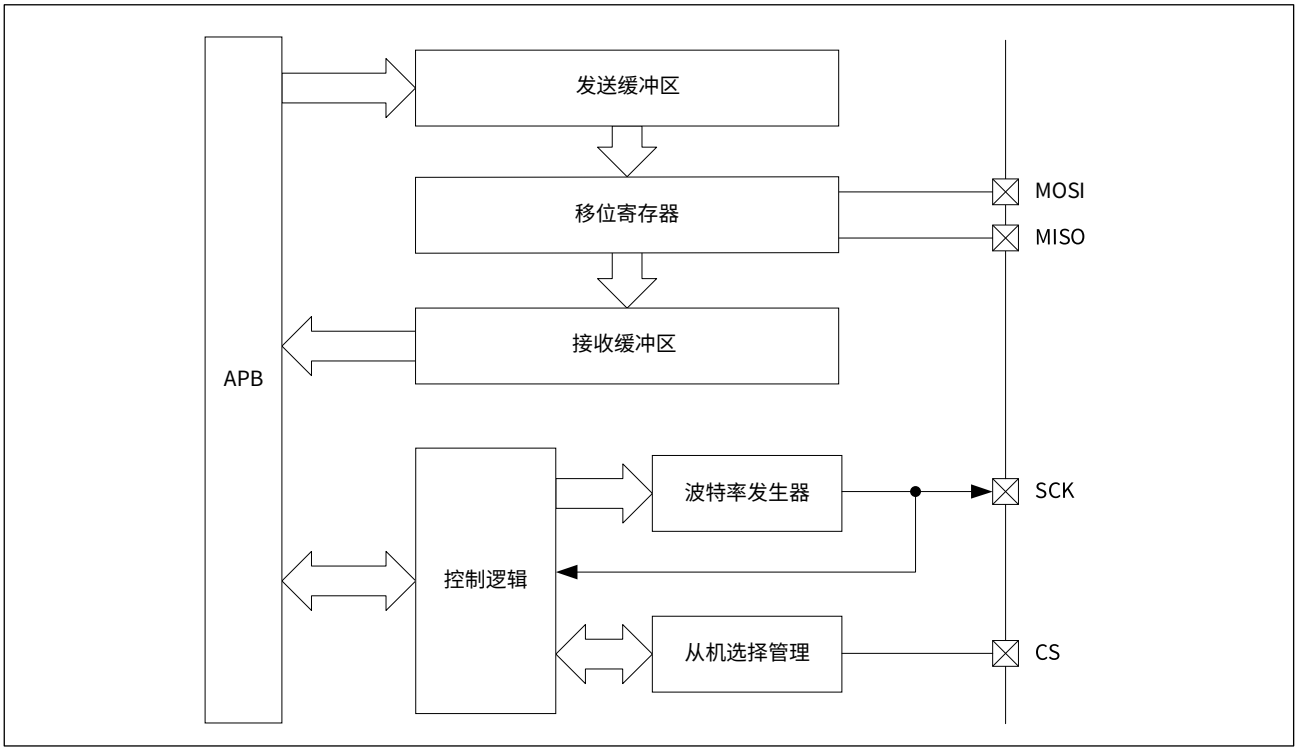
- 支持主机模式、从机模式
- 支持全双工、单线半双工、单工
- 可选的 4 位到 16 位数据帧宽度
- 支持收发数据 LSB 或 MSB 在前
- 可编程时钟极性和时钟相位
- 主机模式下通信速率高达 PCLK/2
- 从机模式下通信速率高达 PCLK/4
- 支持多机通信模式
- 8 个带标志位的中断源
- 支持直接内存访问 (DMA)

18.3 功能描述

18.3.1 功能框图

SPI 控制器的功能框图如下图所示：

图 18-1 SPI 功能框图



SPI 一般通过 4 个引脚与外部设备相连：

- MOSI
  主机输出 / 从机输入，用于主机模式下的数据发送和从机模式下的数据接收；
- MISO
  主机输入 / 从机输出，用于主机模式下的数据接收和从机模式下的数据发送；
- SCK
  同步串行时钟，主机时钟输出和从机时钟输入，发送或接收主机同步时钟；
- CS
  从机选择，也用于多机通信时总线冲突检测，参见 18.3.3 从机选择引脚 CS。

CW32F020 支持用户灵活选择 GPIO 作为 SPI 通信引脚，通过 AFR 功能复用实现，如下表所示：

表 18-1 SPI 引脚选择

SPI1	GPIO	AFR	SPI2	GPIO	AFR
SPI1_CS	PA04	0x05	SPI2_CS	PA03	0x05
	PA09	0x05		PB09	0x05
	PA15	0x05		PF01	0x05
	PB12	0x05		PB12	0x04
SPI1_MISO	PA06	0x05	SPI2_MISO	PA00	0x05
	PA11	0x05		PB07	0x05
	PB04	0x05		PC14	0x05
	PB14	0x05		PB14	0x04
SPI1_MOSI	PA07	0x05	SPI2_MOSI	PA01	0x05
	PA12	0x05		PB06	0x05
	PB05	0x05		PC15	0x05
	PB15	0x05		PB15	0x04
SPI1_SCK	PA05	0x05	SPI2_SCK	PA02	0x05
	PA10	0x05		PB08	0x05
	PB03	0x05		PB10	0x05
	PB13	0x05		PF00	0x05
				PB13	0x04

SPI 支持多种工作模式，在不同的工作模式下，各引脚具有不同的功能，引脚配置如下表所示（其中 x=1，2）：

表 18-2 SPI 端口配置

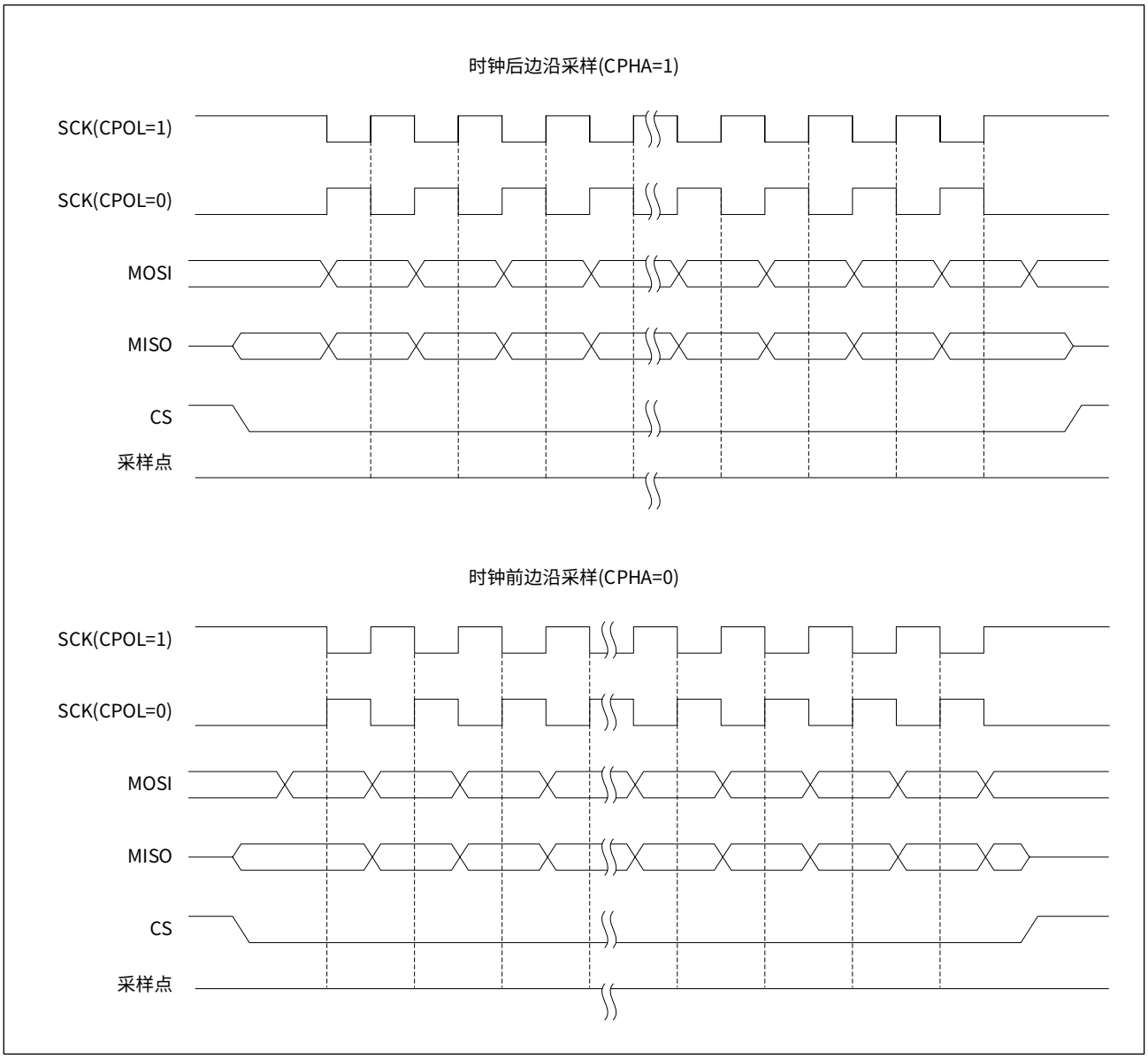
SPI 引脚	模式配置	作用	GPIO 配置
SPIx_SCK	主机	串行时钟输出	数字，推挽输出，复用
	从机	串行时钟输入	数字，浮空输入，复用
SPIx_MOSI	全双工 / 主机	主机数据输出	数字，推挽输出，复用
	全双工 / 从机	从机数据输入	数字，浮空输入，复用
	单线半双工 / 主机	主机数据收发	数字，推挽输出，复用
	单线半双工 / 从机	不使用	可作为通用 I/O
	单工单发 / 主机	主机数据输出	数字，推挽输出，复用
	单工单发 / 从机	不使用	可作为通用 I/O
	单工单收 / 主机	不使用	可作为通用 I/O
	单工单收 / 从机	从机数据输入	数字，浮空输入，复用
SPIx_MISO	全双工 / 主机	主机数据输入	数字，浮空输入，复用
	全双工 / 从机	从机数据输出	数字，推挽输出，复用
	单线半双工 / 主机	不使用	可作为通用 I/O
	单线半双工 / 从机	从机数据收发	数字，推挽输出，复用
	单工单发 / 主机	不使用	可作为通用 I/O
	单工单发 / 从机	从机数据输出	数字，推挽输出，复用
	单工单收 / 主机	主机数据输出	数字，推挽输出，复用
	单工单收 / 从机	不使用	可作为通用 I/O
SPIx_CS	主机（SSM=0）	从机选择输入	数字，浮空输入，复用
	主机（SSM=1）	从机选择输出	数字，推挽输出，复用
	从机（SSM=0）	从机选择输入	数字，浮空输入，复用
	从机（SSM=1）	不使用	可作为通用 I/O

18.3.2 通信时序和数据格式

通信时序

下图是 SPI 的通信时序，CS、SCK、MOSI 信号都由主机控制产生，MISO 信号是从机响应信号。从机选择 CS 信号由高电平变低，是 SPI 通信的起始信号，当从机检测到起始信号后，开始与主机通信。在每个 SCK 的时钟周期中，MOSI 和 MISO 信号线传输一位数据。CS 由低电平变高，是通信的停止信号，表示本次通信结束。

图 18-2 SPI 通信时序



数据帧格式

数据帧宽度由控制寄存器 SPIx_CR1 的 WIDTH 位域配置，可设置 4 ~ 16bit 任意数据位宽。

数据的大小端由控制寄存器 SPIx_CR1 的 LSBF 位域配置，可选择最高有效位在前（MSB）或最低有效位在前（LSB）。

时钟频率

同步串行时钟 SCK 信号由 SPI 主机控制产生，其时钟来源是 PCLK，通过配置控制寄存器 SPIx_CR1 的 BR 位域来设置分频因子，可选择 2 ~ 128 分频。对于 SPI 从机，配置 SPIx_CR1.BR 无影响。

时钟极性、时钟相位

时钟极性 CPOL，指设备处于没有数据传输的空闲状态时，SCK 串行时钟线的电平状态。通过控制寄存器 SPIx_CR1 的 CPOL 位域进行配置：设置 SPIx_CR1.CPOL 为 0，SCK 时钟线在空闲时为低电平；设置 SPIx_CR1.CPOL 为 1，SCK 时钟线在空闲时为高电平。

时钟相位 CPHA，指数据的采样和移位时刻。通过控制寄存器 SPIx_CR1 的 CPHA 进行配置：设置 SPIx_CR1.CPHA 为 0，在 SCK 的前边沿（SCK 由空闲状态变为非空闲状态的时钟边沿）采样、后边沿（SCK 由非空闲状态变为空闲状态的时钟边沿）移位；设置 SPIx_CR1.CPHA 为 1，在 SCK 的前边沿移位、后边沿采样。

根据时钟极性 CPOL 和时钟相位 CPHA 的不同配置，SPI 可设置 4 种电平模式，主机和从机需要配置成相同的电平模式才能保证正常通信。

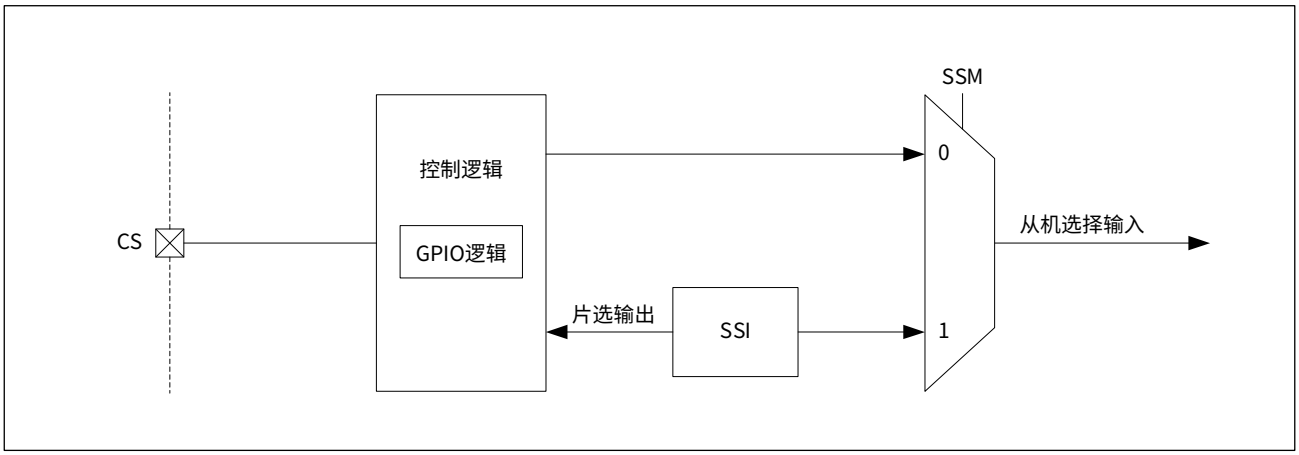
表 18-3　SPI 电平模式

CPOL	CPHA	空闲时 SCK 时钟	采样时刻
0	0	低电平	前边沿
0	1	低电平	后边沿
1	0	高电平	前边沿
1	1	高电平	后边沿

18.3.3 从机选择引脚 CS

下图所示为从机选择引脚 CS 的配置管理示意图。在主机模式下，CS 可选择作为输入或输出，作为输入时用于检测多主机模式下的总线冲突，作为输出时用于产生从机片选信号。在从机模式下，CS 作为一个器件片选输入。

图 18-3 从机选择引脚管理



在主机和从机模式下，通过控制寄存器 SPIx_CR1 的 SSM 位域来管理从机选择引脚 CS。

主机模式下 CS 引脚

设置 SPIx_CR1.SSM 为 0，CS 被配置为输入，用于检测多机通信系统中的总线是否发生冲突（注意，此时主机需通过其它 GPIO 来选择需通信的从机），请参见 18.3.7 多机通信。

设置 SPIx_CR1.SSM 为 1，从机选择引脚 CS 被配置为输出，用于产生从机选择信号。CS 引脚输出电平由从机选择寄存器 SPIx_SSI 的 SSI 位域决定：设置 SPIx_SSI.SSI 为 1，CS 输出高电平；设置 SPIx_SSI.SSI 为 0，CS 输出低电平。

从机模式下 CS 引脚

设置 SPIx_CR1.SSM 为 0，使用 CS 信号作为器件选择。CS 电平决定本机是否被选中：CS 为低电平时本机被选中；CS 为高电平时本机被取消选中。

设置 SPIx_CR1.SSM 为 1，不使用 CS 作为器件选择，从机选择由 SPIx_SSI 寄存器的 SSI 位域值决定。设置 SPIx_SSI.SSI 为 0，本机被选中；设置 SPIx_SSI.SSI 为 1，本机被取消选中。这种配置下用户程序以软件方式设置本机是否被选中，CS 引脚可作普通 IO 使用。

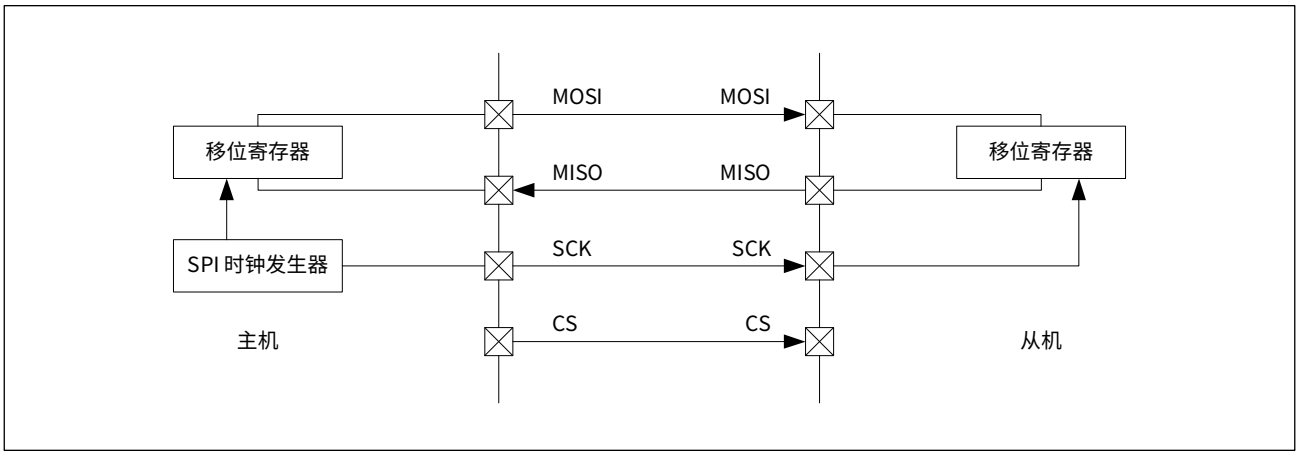
18.3.4 全双工模式

SPI 支持全双工通信模式，在该模式下，主机和从机的移位寄存器通过 MOSI、MISO 两条单向数据线进行连接，在主机提供的 SCK 时钟信号的控制下同时进行两个线路数据传输。

设置控制寄存器 SPIx_CR1 的 MODE 位域为 0x0，使 SPI 工作于全双工通信模式。

全双工模式的典型应用框图下图所示：

图 18-4 全双工通信模式



主机收发

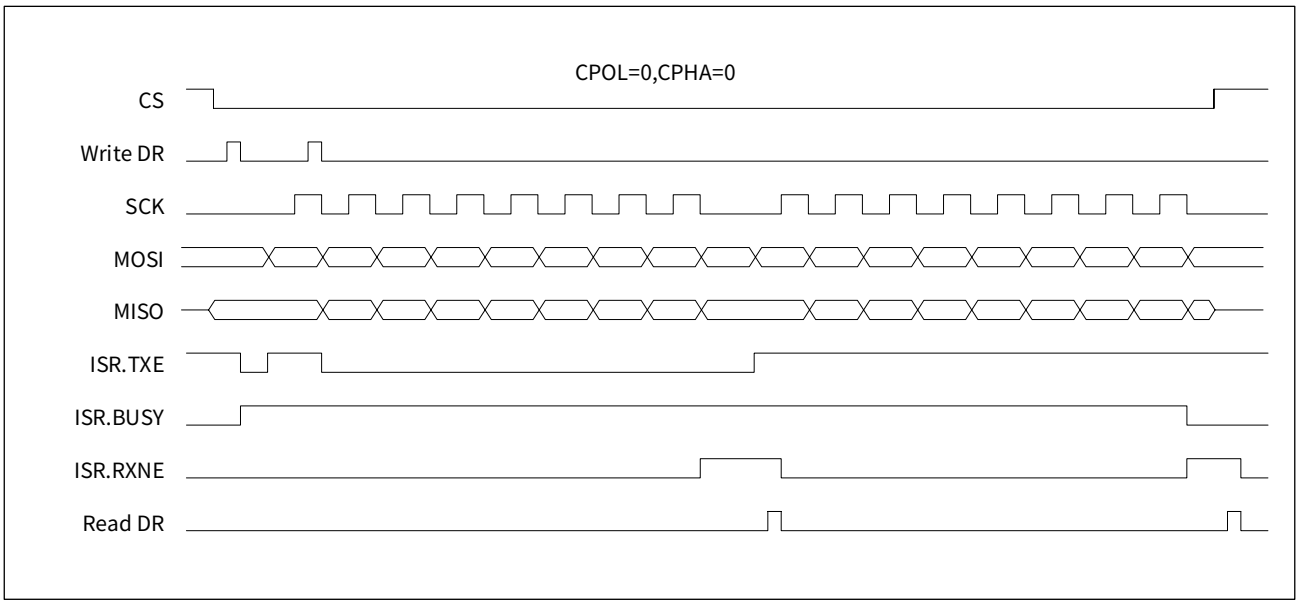
设置 SPIx_CR1.MSTR 为 1，SPI 工作于主机模式。

主机设置 SPIx_SSI.SSI 为 0，在从机选择 CS 引脚输出低电平，作为通信起始信号。

当发送缓冲器为空时，即 SPIx_ISR.TXE 标志位为 1，将待发送的一帧数据写入 SPIx_DR 寄存器，数据在同步移位时钟信号的控制下，从 MOSI 引脚输出，同时将 MISO 引脚的数据接收到移位寄存器。发送一个数据帧结束时，接收缓冲器非空标志位 SPIx_ISR.RXNE 由硬件置 1，表示已经接收完成一帧数据，此时可以读取 SPIx_DR 寄存器。

设置 SPIx_SSI.SSI 为 1，从机选择引脚 CS 输出高电平，结束本次通信。

图 18-5 全双工模式下主机收发时序 (两个字节)



从机收发

设置 SPIx_CR1.MSTR 为 0，SPI 工作于从机模式。

在从机选择 CS 信号被拉低之前，从机需要设置 SPIx_ICR.FLUSH 为 0，以清空发送缓冲区和移位寄存器，并将待发送的第一帧数据写入 SPIx_DR 寄存器。

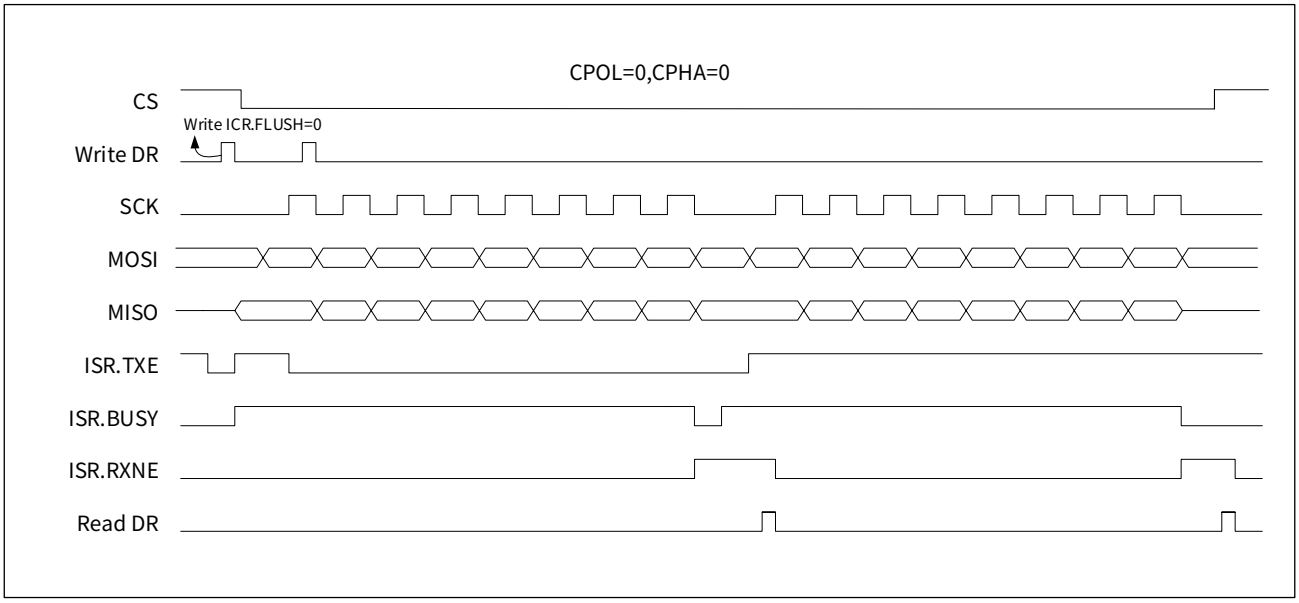
当 CS 信号被拉低后，被写入的数据将在主机的同步移位时钟信号的控制下，从 MISO 引脚输出，同时接收 MOSI 引脚的数据到移位寄存器。

如果是多数据帧连续通信，用户应当不断查询 SPIx_ISR.TXE 标志位，一旦标志为 1，立即将待发送的数据写入 SPIx_DR 寄存器，以免出现数据漏发。

当接收缓冲器非空时，即 SPIx_ISR.RXNE 标志位为 1，表示已经接收完成一帧数据，此时可以读取 SPIx_DR 寄存器。

当检测到 CS 引脚变为高电平时，本次通信结束。

图 18-6　全双工模式下从机收发时序（两个字节）



18.3.5 单线半双工模式

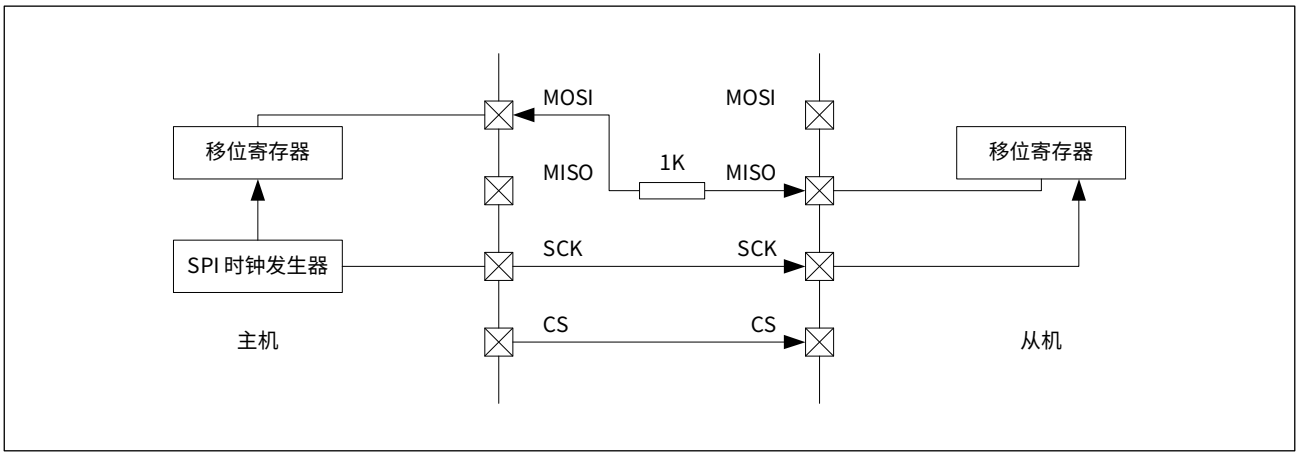
SPI 支持单线半双工通信模式，在该模式下，主机和从机通过一条双向数据线进行数据通信，主机使用 MOSI，从机使用 MISO，未使用的其他 SPI 信号线可供其它功能使用。

设置控制寄存器 SPIx_CR1 的 MODE 位域为 0x3，使 SPI 工作于单线半双工模式。

控制寄存器 SPIx_CR2 的 HDOE 位域，选择收发方向，HDOE 为 0 仅接收，HDOE 为 1 仅发送。

单线半双工模式的典型应用框图，如下图所示：

图 18-7 单线半双工模式



单线半双工通信时，由于主机和从机的通信方向无法保证同步切换 ，有可能会出现驱动冲突而导致器件损害，建议在主机和从机之间的数据线上串联一个电阻，以限制电流。

当 SPI 作为从机时，使用单线半双工模式，可以实现与一个标准的 UART 收发器进行同步模式通信，此时 SPI 必须配置为固定的电平模式：时钟极性 CPOL 为 1，时钟相位 CPHA 为 1。

主机发送

设置 SPIx_CR1.MSTR 为 1，SPI 工作于主机模式；设置 SPIx_CR2.HDOE 为 1，主机仅发送数据。

主机设置 SPIx_SSI.SSI 为 0，在从机选择引脚 CS 输出低电平，作为通信起始信号。

当发送缓冲器为空时，即 SPIx_ISR.TXE 标志位为 1，将待发送的一帧数据写入 SPIx_DR 寄存器，数据在同步移位时钟信号的控制下从 MOSI 引脚输出。

当写入最后一帧数据后，必须等待发送缓冲器为空，即 SPIx_ISR.TXE 标志位变为 1，同时 SPIx_ISR.BUSY 标志位变为 0，以确保数据发送完毕。然后设置 SPIx_SSI.SSI 为 1，在从机选择 CS 引脚输出高电平，结束本次通信。

主机接收

设置 SPIx_CR1.MSTR 为 1，SPI 工作于主机模式；设置 SPIx_CR2.HDOE 为 0，主机仅接收数据。

主机设置 SPIx_SSI.SSI 为 0，在从机选择 CS 引脚输出低电平，作为通信起始信号。

当发送缓冲器为空时，即 SPIx_ISR.TXE 标志位为 1，向 SPIx_DR 寄存器写入一帧虚拟数据以启动传输。

当接收缓冲器非空时，即 SPIx_ISR.RXNE 标志位变为 1，表示已经接收完成一帧数据，此时可以读取 SPIx_DR 寄存器。

如果要接收多帧数据，重复以上步骤写入 SPIx_DR 并从 SPIx_DR 读取接收到的数据。

当接收完所有数据帧后，设置 SPIx_SSI.SSI 为 1，在从机选择 CS 引脚输出高电平，结束本次通信。

从机发送

设置 SPIx_CR1.MSTR 为 0，SPI 工作于从机模式；设置 SPIx_CR2.HDOE 为 1，从机仅发送数据。

在从机选择 CS 信号被拉低之前，从机需要设置 SPIx_ICR.FLUSH 为 0，以清空发送缓冲区和移位寄存器，并将待发送的第一帧数据写入 SPIx_DR 寄存器。

当 CS 信号被拉低后，被写入的数据将在主机的同步移位时钟信号的控制下，从 MISO 引脚输出。

如果是多数据帧连续通信，用户应当不断查询 SPIx_ISR.TXE 标志位，一旦标志为 1，立即将待发送的数据写入 SPIx_DR 寄存器，以免出现数据漏发。

当检测到 CS 引脚变为高电平时，本次通信结束。

从机接收

设置 SPIx_CR1.MSTR 为 0，SPI 工作于从机模式；设置 SPIx_CR2.HDOE 为 0，从机仅接收数据。

当检测到 CS 引脚变为低电平时，从机开始与主机通信。

当接收缓冲器非空时，即 SPIx_ISR.RXNE 标志位变为 1，表示已经接收完成一帧数据，此时可以读取 SPIx_DR 寄存器。

当检测到 CS 引脚变为高电平时，本次通信结束。

18.3.6 单工模式

SPI 支持单工通信模式，主机和从机通过一根单向数据线进行单发或单收通信。

主机使用 MOSI 信号线进行单发通信，使用 MISO 信号线进行单收通信；从机使用 MOSI 信号线进行单收通信，使用 MISO 信号线进行单发通信，未使用的信号线可供其它功能使用。

设置控制寄存器 SPIx_CR1 的 MODE 位域为 0x1，SPI 工作于单工单发通信模式；设置 MODE 位域为 0x2，SPI 工作于单工单收通信模式。

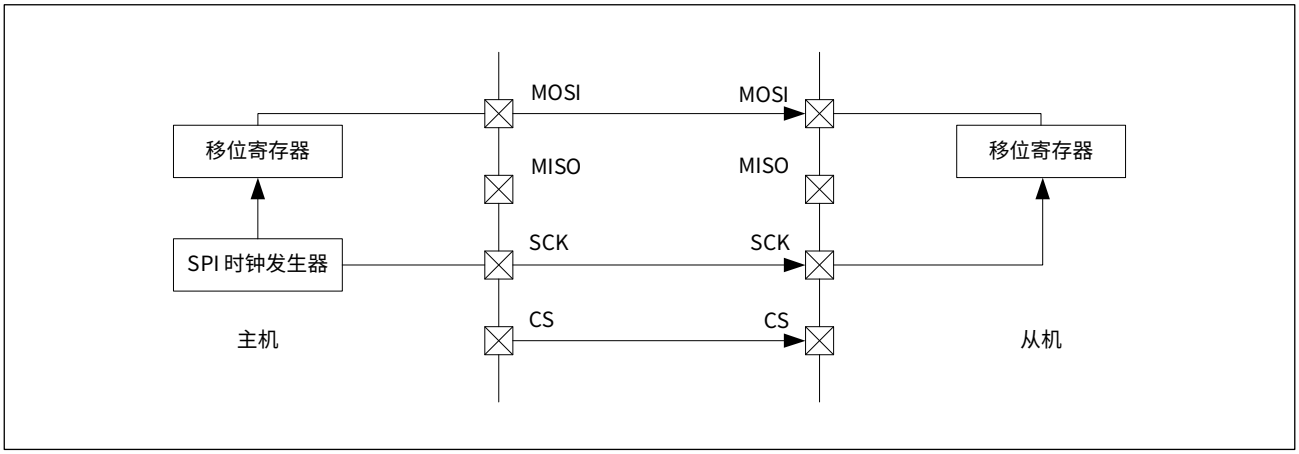
注：

在单工单发模式下，必须忽略所有与发送端接收流相关的事件，例如接收缓冲上溢错误标志位 SPIx_ISR.OV。

18.3.6.1 主机单发 / 从机单收

在主机单发、从机单收的应用场景下，主机和从机使用一根 MOSI 数据线进行通信，其应用框图如下图所示：

图 18-8 主机单发、从机单收模式



主机单发

设置 SPIx_CR1.MODE 为 0x1，SPI 工作于单工单发通信模式；设置 SPIx_CR1.MSTR 为 1，SPI 工作于主机模式。

设置 SPIx_SSI.SSI 为 0，在从机选择 CS 引脚输出低电平，作为通信起始信号。

当发送缓冲器为空时，即 SPIx_ISR.TXE 标志位为 1，将待发送的一帧数据写入 SPIx_DR 寄存器，数据在同步移位时钟信号的控制下从 MOSI 引脚输出。

当写入最后一帧数据后，必须等待发送缓冲空标志位 SPIx_ISR.TXE 变为 1，同时 SPI 总线忙标志位 SPIx_ISR.BUSY 变为 0，以确保数据发送完毕。然后设置 SPIx_SSI.SSI 为 1，使从机选择 CS 引脚输出高电平，结束本次通信。

从机单收

设置 SPIx_CR1.MODE 为 0x2，SPI 工作于单工单收通信模式；设置 SPIx_CR1.MSTR 为 0，SPI 工作于从机模式。

当检测到 CS 引脚变为低电平时，从机开始与主机通信。

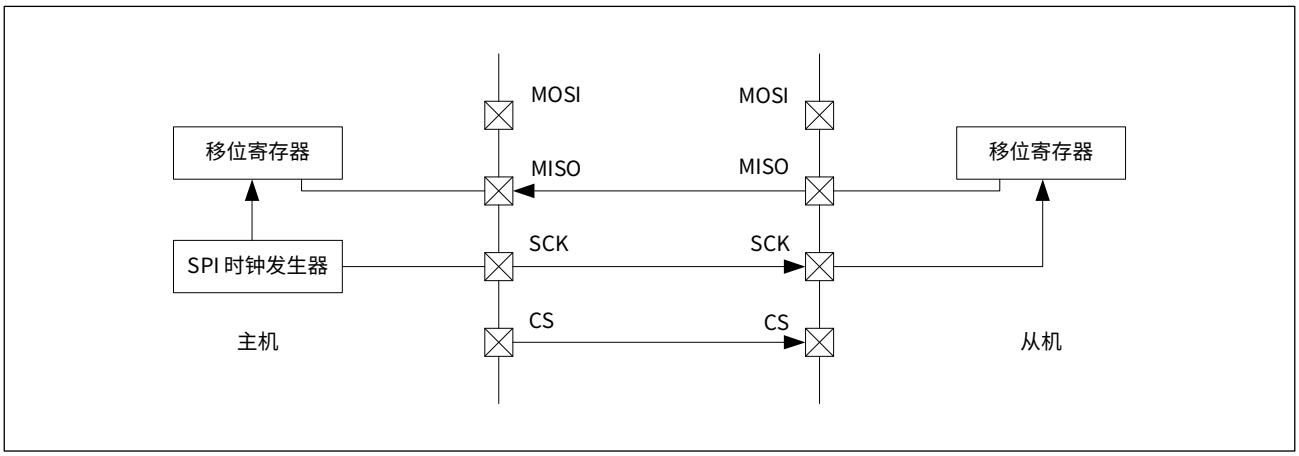
当接收缓冲器非空时，即 SPIx_ISR.RXNE 标志位为 1，表示已经接收完成一帧数据，此时可以读取 SPIx_DR 寄存器。

当检测到 CS 引脚变为高电平时，本次通信结束。

18.3.6.2 主机单收 / 从机单发

在主机单收、从机单发的应用场景下，主机和从机使用一根 MISO 数据线进行通信，其应用框图如下图所示：

图 18-9 主机单收、从机单发模式



主机单收

设置 SPIx_CR1.MODE 为 0x2，SPI 工作于单工单收通信模式；设置 SPIx_CR1.MSTR 为 1，SPI 工作于主机模式。

设置 SPIx_SSI.SSI 为 0，在从机选择 CS 引脚输出低电平，作为通信起始信号。

当发送缓冲器为空时，即 SPIx_ISR.TXE 标志位为 1，向 SPIx_DR 寄存器写入一帧虚拟数据以启动传输。

当接收缓冲器非空时，即 SPIx_ISR.RXNE 标志位为 1，表示已经接收完成一帧数据，此时可以读取 SPIx_DR 寄存器。

如果要接收多帧数据，重复以上步骤写入 SPIx_DR 并从 SPIx_DR 读取接收到的数据。

当接收完所有数据帧后，设置 SPIx_SSI.SSI 为 1，从机选择 CS 引脚输出高电平，结束本次通信。

从机单发

设置 SPIx_CR1.MODE 为 0x1，SPI 工作于单工单发通信模式；设置 SPIx_CR1.MSTR 为 0，SPI 工作于从机模式。

在从机选择信号 CS 被拉低之前，从机需要设置 SPIx_ICR.FLUSH 为 0，以清空发送缓冲区和移位寄存器，并将待发送的第一帧数据写入 SPIx_DR 寄存器。

当 CS 信号被拉低后，被写入的数据在主机的同步移位时钟信号的控制下，从 MISO 引脚输出。

如果是多数据帧连续通信，用户应当不断查询 SPIx_ISR.TXE 标志位，一旦标志为 1，立即将待发送的数据写入 SPIx_DR 寄存器，以免出现数据漏发。

当检测到 CS 引脚变为高电平时，本次通信结束。

18.3.7 多机通信

SPI 支持多机通信模式。在该模式下，主机的从机选择 CS 引脚应配置为输入，与其他主机的总线申请信号相连，用于检测 SPI 总线是否发生冲突。

如果某一主机的从机选择 CS 引脚被拉为低电平，说明有其它主机在占用总线，该主机会产生模式错误（请参见 18.3.9 错误标志），此主机 SPI 将处于禁用状态，以避免 SPI 总线冲突。

如果某一主机的从机选择 CS 引脚检测到高电平，说明 SPI 总线空闲，此时，若该主机想要与从机进行通信，必须先在总线申请信号输出低电平到其它主机的 CS 信号，获得 SPI 总线控制权，然后选择某一从机开始通信。

下图所示是一个典型的多机通信系统，以主机 A 和从机 1 进行通信为例，说明其操作流程如下：

步骤 1：主机 A 检测 CS 引脚，直到出现高电平；

步骤 2：主机 A 的 GPIO0 输出低电平，获取总线控制权，主机 B 被禁用；

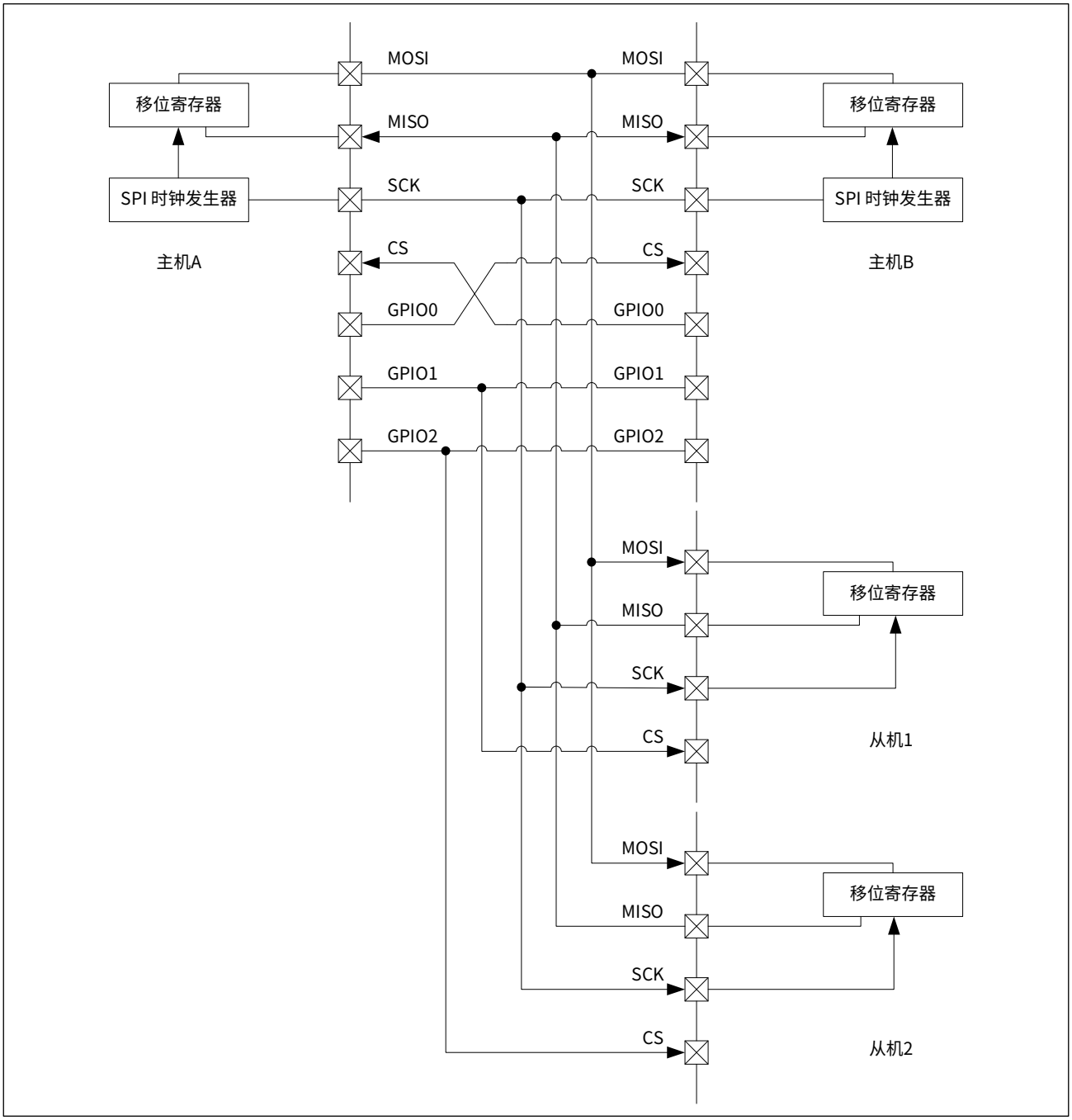
步骤 3：主机 A 的 GPIO1 输出低电平，选中从机 1；

步骤 4：主机 A 与从机 1 进行通信；

步骤 5：主机 A 的 GPIO1 输出高电平，释放从机 1；

步骤 6：主机 A 的 GPIO0 输出高电平，释放 SPI 总线。

图 18-10　多机通信系统



为保证多机通信模式下通信可靠性，从机的控制寄存器 SPIx_CR1 的 MISOHD 位域应设置为 1，当从机被选中时，MISO 引脚为 CMOS 输出；未被选中时，MISO 引脚为高阻输出。

18.3.8 状态标志

SPI 控制器存在 6 个状态标志，用来指示 SPI 的一般工作状态。

发送缓冲空标志位（SPIx_ISR.TXE）

当发送数据从 SPIx_DR 寄存器转移到移位寄存器后，SPIx_ISR.TXE 标志位会被硬件置位，表示发送缓冲器已空，此时允许对 SPIx_DR 寄存器写入新的待发送数据。在对 SPIx_DR 寄存器写入数据的同时，SPIx_ISR.TXE 标志位会被自动清零。

接收缓冲非空标志位（SPIx_ISR.RXNE）

当接收数据从移位寄存器转移到 SPIx_DR 寄存器后，SPIx_ISR.RXNE 标志位会被硬件置位，表示已经完成一帧数据的接收，此时可以读取 SPIx_DR 寄存器，读 SPIx_DR 寄存器的同时将清除 SPIx_ISR.RXNE 标志位。

用户也可以通过软件手动清除 SPIx_ISR.RXNE 标志位。

总线忙标志位（SPIx_ISR.BUSY）

SPIx_ISR.BUSY 标志位，由硬件设置和清除，用于表明当前的 SPI 通信状态。

当正在准备进行或正在进行数据传输时（发送缓冲区有数据或移位寄存器有数据），BUSY 标志位会被硬件置位，表示 SPI 接口处于忙碌状态；

当没有数据传输时（发送缓冲区无数据且移位寄存器无数据），BUSY 标志位被硬件自动清零，表示 SPI 接口处于空闲状态。

从机选择信号状态（SPIx_ISR.SSLVL）

SPIx_ISR.SSLVL 状态位，由硬件设置和清除，用于指示从机选择输入信号的电平状态。

SPIx_ISR.SSLVL 为 0，从机选择 CS 输入是低电平，从机被选中；SPIx_ISR.SSLVL 为 1，从机选择 CS 输入是高电平，从机未被选中。

从机选择输入上升沿标志（SPIx_ISR.SSR）

SPIx_ISR.SSR 标志位，用于指示从机选择 CS 输入信号是否出现了上升沿。

SPIx_ISR.SSR 为 0，表示从机选择 CS 输入没有出现上升沿；SPIx_ISR.SSR 为 1，表示从机选择输入出现了上升沿。

从机选择输入下降沿标志（SPIx_ISR.SSF）

SPIx_ISR.SSF 标志位，用于指示从机选择 CS 输入信号是否出现了下降沿。

SPIx_ISR.SSF 为 0，表示从机选择 CS 输入没有出现下降沿；SPIx_ISR.SSF 为 1，表示从机选择 CS 输入出现了下降沿。

18.3.9 错误标志

SPI 控制器存在 4 个错误标志，用来指示 SPI 是否发生了错误。

模式错误标志（SPIx_ISR.MODF）

SPIx_ISR.MODF 标志位，用于检测 SPI 多机通信模式下的总线冲突。

当 SPI 工作于主机模式，且从机选择 CS 引脚被配置为输入时，如果 CS 引脚输入为低电平，则会发生模式错误，SPIx_ISR.MODF 标志位被硬件置位，表示此时有其他 SPI 主机在占用总线。

发生模式错误后，SPIx_CR1.EN 被清零，SPI 被强制关闭。

设置 SPIx_ICR.MODF 为 0，可清除 SPIx_ISR.MODF 标志位，完成清零后，用户需要重新配置 SPI 控制器。

发生模式错误后，如果重新使能了 SPI 控制器，SPI 仍可以进行正常的数据传输，但 SPIx_ISR.MODF 标志位会保持置位状态，知道用户软件清除。

从机模式下的从机选择错误标志位（SPIx_ISR.SSERR）

当 SPI 工作于从机模式，且正在进行有效的数据传输时，如果从机选择 CS 输入被拉高，则会发生从机选择错误，SPIx_ISR.SSERR 标志位被硬件置位。

发生从机选择错误会导致当前的数据传输出错，从机进入未选中状态。

发生从机选择错误之后，如果从机选择 CS 输入再次被拉低，SPI 仍可以进行正常的数据传输，但 SPIx_ISR.SSERR 标志位会保持置位状态，直到用户软件清除。

接收缓冲上溢错误标志位（SPIx_ISR.OV）

当主机或从机在收到一帧数据后，没有清除 SPIx_ISR.RXNE 标志位，即，既没有读 SPIx_DR 寄存器又没有软件清除 SPIx_ICR.RXNE 为 0，然后收到新的一帧数据时，就会发生接收缓冲上溢错误，SPIx_ISR.OV 标志位被硬件置位。

发生接收缓冲上溢错误后，新接收到的数据将覆盖缓冲器中原有的数据。

发生接收缓冲上溢错误后，SPI 仍可以进行正常的数据传输，但 SPIx_ISR.OV 标志位会保持置位状态，直到用户软件清除。

从机模式下的发送缓冲下溢错误标志位（SPIx_ISR.UD）

当 SPI 工作于从机模式时，如果发送缓冲器空，且新一帧的传输已经开始，就会发生发送缓冲下溢错误，SPIx_ISR.UD 标志位会被硬件置位，表示从机没有及时向 SPIx_DR 寄存器写入数据，错过了给主机发送数据。

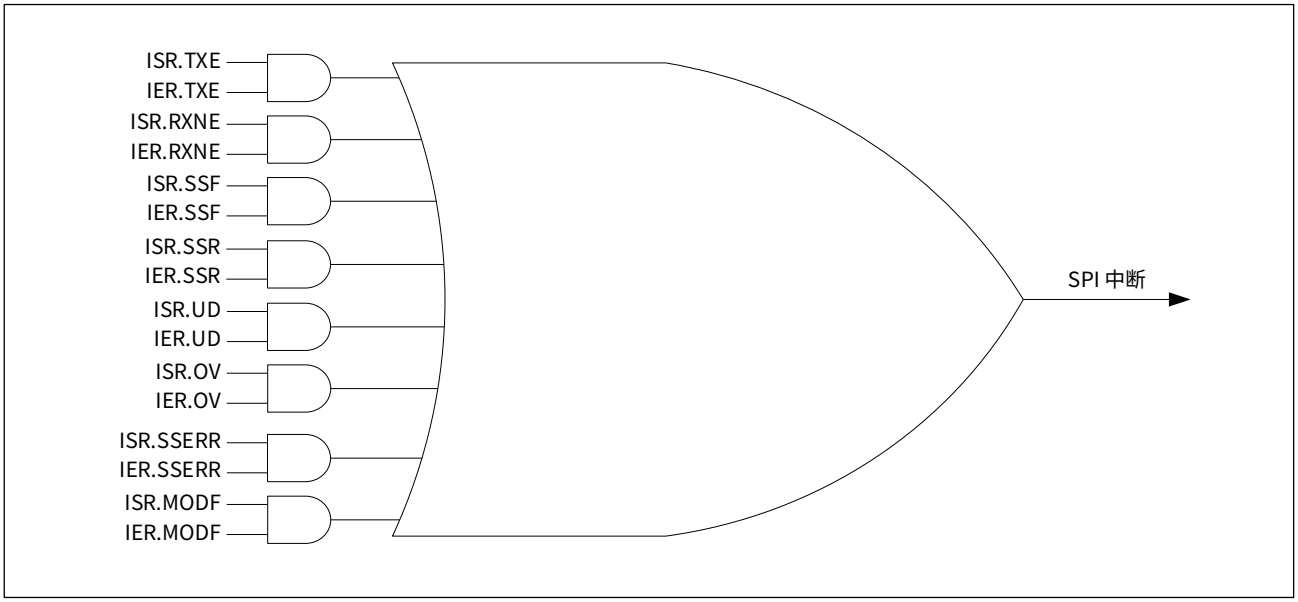
发生发送缓冲下溢错误之后，SPI 仍可以进行正常的数据传输，但 SPIx_ISR.UD 标志位会保持置位状态，直到用户软件清除。

18.4 SPI 中断

SPI 控制器支持 8 个中断源，当 SPI 中断触发事件发生时，中断标志位会被硬件置位，如果设置了对应的中断使能控制位，将产生中断请求。

CW32F020 的一个 SPI 模块使用一个相同的系统 SPI 中断，SPI 中断是否产生中断跳转由中断使能设置寄存器 NVIC_ISER 的相应位控制。系统 SPI 中断示意图如下图所示：

图 18-11 SPI 中断



在用户 SPI 中断服务程序中，应查询相关 SPI 中断标志位，以进行相应的处理，在退出中断服务程序之前，要清除该中断标志位，避免重复进入中断程序。

SPI 的各中断源的标志位、中断使能位、中断标志清除位及清除方法，如下表所示：

表 18-4 SPI 中断控制

中断事件	中断标志位	中断使能位	标志清除方法
发送缓冲器空	ISR.TXE	IER.TXE	写 SPIx_DR 寄存器
接收缓冲器非空	ISR.RXNE	IER.RXNE	读 SPIx_DR 寄存器，或写 0 到 ICR.RXNE
从机选择输入出现下降沿	ISR.SSF	IER.SSF	写 0 到 ICR.SSF
从机选择输入出现上升沿	ISR.SSR	IER.SSR	写 0 到 ICR.SSR
从机模式下发送缓冲下溢	ISR.UD	IER.UD	写 0 到 ICR.UD
接收缓冲上溢	ISR.OV	IER.OV	写 0 到 ICR.OV
从机模式下从机选择错误	ISR.SSERR	IER.SSERR	写 0 到 ICR.SSERR
模式错误	ISR.MODF	IER.MODF	写 0 到 ICR.MODF

18.5 直接内存访问（DMA）

SPI 控制器支持 DMA 传输功能，实现在 SPI 数据寄存器和存储器之间进行高速数据传输，无须 CPU 干预。DMA 请求可以由软件或硬件触发，且发送和接收的 DMA 请求是分别产生的。

DMA 发送

设置 SPIx_CR1.DMATX 为 1，使能 SPI 发送 DMA 传输请求。

当发送缓冲区为空时，产生 DMA 请求，触发 DMA 将一帧数据从指定的存储区传输到 SPIx_DR 寄存器中，直到 DMA 控制器中配置的数据量全部传输完毕。

DMA 发送的具体寄存器配置流程请参见 8 直接内存访问（DMA）章节和 18.6 编程示例。

当 DMA 传输完成后，对应的 DMA 通道 y 传输完成标志位 DMA_ISR.TCy 会被硬件置位，如果允许中断（即设置 DMA_CSRy.TCIE 为 1），将产生中断请求。

在用户应用中，在关闭 SPI 之前，建议查询 SPIx_ISR.BUSY 标志位是否已经清零，确保数据已全部通过 SPI 发送完毕，避免破坏最后一次传输的数据。

DMA 接收

设置 SPIx_CR1.DMARX 为 1，使能 SPI 接收 DMA 传输请求。

当接收缓冲区非空时，产生 DMA 请求，触发 DMA 将数据从 SPIx_DR 寄存器传输到指定的存储区，直到 DMA 控制器中配置的数据量全部传输完毕。

DMA 接收的具体寄存器配置流程请参见 8 直接内存访问（DMA）章节和 18.6 编程示例。

当 DMA 传输完成后，对应的 DMA 通道 y 传输完成标志位 DMA_ISR.TCy 会被硬件置位，如果允许中断（即设置 DMA_CSRy.TCIE 为 1），将产生中断请求。

18.6 编程示例

18.6.1 全双工 / 主模式

18.6.1.1 查询方式收发

步骤 1：设置 SYSCTRL_AHBEN.GPIOx 为 1，SYSCTRL_APBENx.SPIx 为 1，使能 SPI 引脚对应的 GPIO 时钟和 SPI 工作时钟；

步骤 2：将 SPIx_SCK、SPIx_MOSI 和 SPIx_CS 引脚配置为推挽复用输出模式，SPIx_MISO 引脚配置为浮空输入、复用模式，具体寄存器配置步骤请参见 9 通用输入输出端口（GPIO）章节；

步骤 3：设置 SPIx_CR1.MODE 为 0x0，配置 SPIx 为双向全双工通信模式；

步骤 4：设置 SPIx_CR1.MSTR 为 1，配置 SPIx 为主机模式；

步骤 5：配置 SPIx_CR1.WIDTH，设置每帧数据的宽度；

步骤 6：配置 SPIx_CR1.LSBF，设置数据位收发顺序；

步骤 7：配置 SPIx_CR1.CPOL，设置时钟极性；

步骤 8：配置 SPIx_CR1.CPHA，设置时钟相位；

步骤 9：设置 SPIx_CR1.SSM 为 1，通过 SSI 寄存器决定从机选择输出值；

步骤 10：配置 SPIx_CR1.BR，设置 SCK 波特率；

步骤 11：设置 SPIx_CR1.EN 为 1，使能 SPIx 控制器；

步骤 12：设置 SPIx_SSI.SSI 为 0，通信开始；

步骤 13：查询等待 SPIx_ISR.TXE 标志位置 1，确认发送缓冲器为空；

步骤 14：将待发送的一帧数据写入 SPIx_DR 寄存器；

步骤 15：查询等待 SPIx_ISR.RXNE 标志位置 1，确认接收完成一帧数据；

步骤 16：读取 SPIx_DR 寄存器并保存数据；

步骤 17：重复步骤 13 至步骤 17，进行下一帧数据的收发，直到全部数据收发完毕；

步骤 18：设置 SPIx_SSI.SSI 为 1，通信结束。

18.6.1.2 中断方式收发

步骤 1：设置 SYSCTRL_AHBEN.GPIOx 为 1，SYSCTRL_APBENx.SPIx 为 1，使能 SPI 引脚对应的 GPIO 时钟和 SPI 工作时钟；

步骤 2：将 SPIx_SCK、SPIx_MOSI 和 SPIx_CS 引脚配置为推挽复用输出模式，SPIx_MISO 引脚配置为浮空输入、复用模式，具体寄存器配置步骤请参见 9 通用输入输出端口（GPIO）章节；

步骤 3：设置 SPIx_CR1.MODE 为 0x0，配置 SPIx 为双向全双工通信模式；

步骤 4：设置 SPIx_CR1.MSTR 为 1，配置 SPIx 为主机模式；

步骤 5：配置 SPIx_CR1.WIDTH，设置每帧数据的宽度；

步骤 6：配置 SPIx_CR1.LSBF，设置数据位收发顺序；

步骤 7：配置 SPIx_CR1.CPOL，设置时钟极性；

步骤 8：配置 SPIx_CR1.CPHA，设置时钟相位；

步骤 9：设置 SPIx_CR1.SSM 为 1，通过 SSI 寄存器决定从机选择输出值；

步骤 10：配置 SPIx_CR1.BR，设置 SCK 波特率；

步骤 11：设置 SPIx_CR1.EN 为 1，使能 SPIx 控制器；

步骤 12：向 SPIx_ICR 写入 0x00，清除所有标志位；

步骤 13：配置 NVIC 控制器，请参见 5 中断章节；

步骤 14：设置 SPIx_SSI.SSI 为 0，通信开始；

步骤 15：设置 SPIx_IER.RXNE 为 1 使能接收缓冲非空中断；

步骤 16：设置 SPIx_IER.TXE 为 1 使能发送缓冲空中断；

步骤 17：进入中断服务函数：查询判断 SPIx_ISR.TXE 标志位，如果标志位为 1，则将待发送的一帧数据写入 SPIx_DR 寄存器；查询判断 SPIx_ISR.RXNE 标志位，如果标志位为 1，则读取 SPIx_DR 寄存器并保存数据；

步骤 18：设置 SPIx_SSI.SSI 为 1，通信结束。

18.6.1.3 DMA 方式收发

以下是 SPI1（SPI1_CS 引脚为 PA04、SPI1_SCK 引脚为 PA05、SPI1_MISO 引脚为 PA06、SPI1_MOSI 引脚为 PA07）工作于主机模式时，通过 DMA 通道 1 进行全双工模式数据发送、DMA 通道 2 进行全双工模式数据接收的编程示例。

在这个应用示例中，设置 PCLK 为 24MHz，波特率为 6Mbps，DMA 数据传输量为 100，通过硬件触发启动 DMA 传输，将在 SRAM 区定义的一个 8 位数组 SendBuff 中的数据传输到发送缓冲区，将接收缓冲区中的数据传输到一个 8 位数组 RecvBuff 中保存。

步骤 1：设置 SYSCTRL_AHBEN.GPIOA 为 1，SYSCTRL_APBEN2.SPI1 为 1，SYSCTRL_AHBEN.DMA 为 1，分别使能 GPIOA、SPI1 和 DMA 的配置时钟及工作时钟；

步骤 2：将 PA04、PA05 和 PA07 引脚配置为推挽复用输出模式，PA06 引脚配置为浮空输入、复用模式，具体寄存器配置步骤请参见 9 通用输入输出端口（GPIO）章节；

步骤 3：设置 SPI1_CR1.MODE 为 0x0，配置 SPI1 为双向全双工通信模式；

步骤 4：设置 SPI1_CR1.MSTR 为 1，配置 SPI1 为主机模式；

步骤 5：设置 SPI1_CR1.WIDTH 为 0x7，数据宽度为 8bit；

步骤 6：设置 SPI1_CR1.LSBF 为 0，最高有效位收发在前；

步骤 7：设置 SPI1_CR1.CPOL 为 1，时钟空闲时为高电平；

步骤 8：设置 SPI1_CR1.CPHA 为 0，前边沿采样，后边沿移位；

步骤 9：设置 SPI1_CR1.SSM 为 1，通过 SSI 寄存器决定从机选择输出值；

步骤 10：设置 SPI1_CR1.BR 为 0x1，波特率为 6Mbps；

步骤 11：设置 SPI1_CR1.EN 为 1，使能 SPI1 控制器；

步骤 12：定义一个变量 uint8_t SendBuff [100]，并赋值，存放待发送的数据；

步骤 13：定义一个变量 uint8_t RecvBuff [100]，存放待接收的数据；

步骤 14：向 SPI1_ICR 写入 0x00，清除 SPI1 所有标志位；

步骤 15：设置 DMA_ICR.TC1、DMA_ICR.TE1、DMA_ICR.TC2、DMA_ICR.TE2 为 0，清除 DMA 通道 1 和 DMA 通道 2 所有中断标志位；

步骤 16：设置 DMA_SRCADDR1= (uint32_t) SendBuff，源地址为 SendBuff；

步骤 17：设置 DMA_DSTADDR1= (uint32_t) &SPI1_DR，目的地址为 SPI1_DR 寄存器；

步骤 18：设置 DMA_CSR1.SRCINC 为 1，源地址自增；

步骤 19：设置 DMA_CSR1.DSTINC 为 0，目的地址固定；

步骤 20：向 DMA_CNT1 写入 0x10064，数据传输量为 100；

步骤 21：设置 DMA_CSR1.SIZE 为 0，传输数据位宽为 8bit；

步骤 22：设置 DMA_CSR1.TRANS 为 1，块传输（BLOCK）；

步骤 23：设置 DMA_TRIG1.Type 为 1，硬件触发模式；

步骤 24：设置 DMA_TRIG1. HardSrc 为 0x7，SPI1 发送缓冲区为空时触发 DMA 传输；

步骤 25：设置 DMA_SRCADDR2= (uint32_t) &SPI1_DR，源地址为 SPI1_DR 寄存器；

步骤 26：设置 DMA_DSTADDR2=(uint32_t) RecvBuff，目的地址为 RecvBuff；

步骤 27：设置 DMA_CSR2.SRCINC 为 0，源地址固定；

步骤 28：设置 DMA_CS2.DSTINC 为 1，目的地址自增；

步骤 29：向 DMA_CNT2 写入 0x10064，数据传输量为 100；

步骤 30：设置 DMA_CSR2.SIZE 为 0，传输数据位宽为 8bit；

步骤 31：设置 DMA_CSR2.TRANS 为 1，块传输（BLOCK）；

步骤 32：设置 DMA_TRIG2.Type 为 1，硬件触发模式；

步骤 33：设置 DMA_TRIG2. HardSrc 为 0x6，SPI1 接收缓冲区非空时触发 DMA 传输；
步骤 34：设置 SPI1_SSI.SSI 为 0，通信开始；
步骤 35：设置 DMA_CSR1.EN 为 1，使能 DMA 通道 1 传输；
步骤 36：设置 DMA_CSR2.EN 为 1，使能 DMA 通道 2 传输；
步骤 37：设置 SPI1_CR1.DMATX 为 1，打开 SPI1 的 DMA 发送硬件握手逻辑；
步骤 38：设置 SPI1_CR1.DMARX 为 1，打开 SPI1 的 DMA 接收硬件握手逻辑；
步骤 39：查询等待 DMA_ISR.TC1 变为 1，确认 DMA 通道 1 传输完成；
步骤 40：查询等待 DMA_ISR.TC2 变为 1，确认 DMA 通道 2 传输完成；
步骤 41：设置 SPI1_CR1.DMATX 为 0，关闭 SPI1 的 DMA 发送硬件握手逻辑；
步骤 42：设置 SPI1_CR1.DMARX 为 0，关闭 SPI1 的 DMA 接收硬件握手逻辑；
步骤 43：查询等待 SPI1_ISR.BUSY 变为 0，确保 SPI1 总线空闲；
步骤 44：设置 SPI1_SSI.SSI 为 1，通信结束；
步骤 45：如果要传输新的数据，重复步骤 14 至步骤 45。

18.6.2 单线半双工 / 主模式

18.6.2.1 查询方式发送

步骤 1：设置 SYSCTRL_AHBEN.GPIOx 为 1，SYSCTRL_APBENx.SPIx 为 1，使能 SPI 引脚对应的 GPIO 时钟和 SPI 工作时钟；

步骤 2：将 SPIx_SCK、SPIx_MOSI 和 SPIx_CS 引脚配置为推挽复用输出模式，具体寄存器配置步骤请参见 9 通用输入输出端口（GPIO）章节；

步骤 3：设置 SPIx_CR1.MODE 为 0x3，配置 SPIx 为单线半双工通信模式；

步骤 4：设置 SPIx_CR1.MSTR 为 1，配置 SPIx 为主机模式；

步骤 5：配置 SPIx_CR1.WIDTH，设置每帧数据的宽度；

步骤 6：配置 SPIx_CR1.LSBF，设置数据位收发顺序；

步骤 7：配置 SPIx_CR1.CPOL，设置时钟极性；

步骤 8：配置 SPIx_CR1.CPHA，设置时钟相位；

步骤 9：设置 SPIx_CR1.SSM 为 1，通过 SSI 寄存器决定从机选择输出值；

步骤 10：配置 SPIx_CR1.BR，设置 SCK 波特率；

步骤 11：设置 SPIx_CR1.EN 为 1，使能 SPIx 控制器；

步骤 12：设置 SPIx_SSI.SSI 为 0，通信开始；

步骤 13：设置 SPIx_CR2. HDOE 为 1，仅发送；

步骤 14：查询等待 SPIx_ISR.TXE 标志位置 1，确认发送缓冲器为空；

步骤 15：将待发送的一帧数据写入 SPIx_DR 寄存器；

步骤 16：如果还有数据待发送，重复步骤 14 至步骤 16，发送下一帧数据；

步骤 17：查询等待 SPIx_ISR.TXE 标志位置 1，最后一帧数据开始发送；

步骤 18：查询等待 SPIx_ISR.BUSY 标志位变为 0，SPIx 总线空闲；

步骤 19：设置 SPIx_SSI.SSI 为 1，通信结束。

18.6.2.2 查询方式接收

步骤 1：设置 SYSCTRL_AHBEN.GPIOx 为 1，SYSCTRL_APBENx.SPIx 为 1，使能 SPI 引脚对应的 GPIO 时钟和 SPI 工作时钟；

步骤 2：将 SPIx_SCK、SPIx_MOSI 和 SPIx_CS 引脚配置为推挽复用输出模式，具体寄存器配置步骤请参见 9 通用输入输出端口（GPIO）章节；

步骤 3：设置 SPIx_CR1.MODE 为 11，配置 SPIx 为单线半双工通信模式；

步骤 4：设置 SPIx_CR1.MSTR 为 1，配置 SPIx 为主机模式；

步骤 5：配置 SPIx_CR1.WIDTH，设置每帧数据的宽度；

步骤 6：配置 SPIx_CR1.LSBF，设置数据位收发顺序；

步骤 7：配置 SPIx_CR1.CPOL，设置时钟极性；

步骤 8：配置 SPIx_CR1.CPHA，设置时钟相位；

步骤 9：设置 SPIx_CR1.SSM 为 1，通过 SSI 寄存器决定从机选择输出值；

步骤 10：配置 SPIx_CR1.BR，设置 SCK 波特率；

步骤 11：设置 SPIx_CR1.EN 为 1，使能 SPIx 控制器；

步骤 12：设置 SPIx_SSI.SSI 为 0，通信开始；

步骤 13：设置 SPIx_CR2. HDOE 为 0，仅接收；

步骤 14：查询等待 SPIx_ISR.TXE 标志位置 1，确认发送缓冲器为空；

步骤 15：将一帧虚拟数据写入 SPIx_DR 寄存器；

步骤 16：查询等待 SPIx_ISR.RXNE 标志位置 1，确认接收完成一帧数据；

步骤 17：读取 SPIx_DR 寄存器并保存数据；

步骤 18：重复步骤 14 至步骤 18，接收下一帧数据，直到全部数据接收完毕；

步骤 19：设置 SPIx_SSI.SSI 为 1，通信结束。

18.6.3 单工模式 / 主模式

18.6.3.1 查询方式发送

步骤 1：设置 SYSCTRL_AHBEN.GPIOx 为 1，SYSCTRL_APBENx.SPIx 为 1，使能 SPI 引脚对应的 GPIO 时钟和 SPI 工作时钟；

步骤 2：将 SPIx_SCK、SPIx_MOSI 和 SPIx_CS 引脚配置为推挽复用输出模式，具体寄存器配置步骤请参见 9 通用输入输出端口（GPIO）章节；

步骤 3：设置 SPIx_CR1.MODE 为 01，配置 SPIx 为单工单发通信模式；

步骤 4：设置 SPIx_CR1.MSTR 为 1，配置 SPIx 为主机模式；

步骤 5：配置 SPIx_CR1.WIDTH，设置每帧数据的宽度；

步骤 6：配置 SPIx_CR1.LSBF，设置数据位收发顺序；

步骤 7：配置 SPIx_CR1.CPOL，设置时钟极性；

步骤 8：配置 SPIx_CR1.CPHA，设置时钟相位；

步骤 9：设置 SPIx_CR1.SSM 为 1，通过 SSI 寄存器决定从机选择输出值；

步骤 10：配置 SPIx_CR1.BR，设置 SCK 波特率；

步骤 11：设置 SPIx_CR1.EN 为 1，使能 SPIx 控制器；

步骤 12：设置 SPIx_SSI.SSI 为 0，通信开始；

步骤 13：查询等待 SPIx_ISR.TXE 标志位置 1，确认发送缓冲器为空；

步骤 14：将待发送的一帧数据写入 SPIx_DR 寄存器；

步骤 15：如果还有数据待发送，重复步骤 13 至步骤 15，发送下一帧数据；

步骤 16：查询等待 SPIx_ISR.TXE 标志位置 1，最后一帧数据开始发送；

步骤 17：查询等待 SPIx_ISR.BUSY 标志位变为 0，SPIx 总线空闲；

步骤 18：设置 SPIx_SSI.SSI 为 1，通信结束。

18.6.3.2 查询方式接收

步骤 1：设置 SYSCTRL_AHBEN.GPIOx 为 1，SYSCTRL_APBENx.SPIx 为 1，使能 SPI 引脚对应的 GPIO 时钟和 SPI 工作时钟；

步骤 2：将 SPIx_SCK、SPIx_CS 引脚配置为推挽复用输出模式，SPIx_MISO 引脚配置为浮空输入、复用模式，具体寄存器配置步骤请参见 9 通用输入输出端口（GPIO）章节；

步骤 3：设置 SPIx_CR1.MODE 为 10，配置 SPIx 为单工单收通信模式；

步骤 4：设置 SPIx_CR1.MSTR 为 1，配置 SPIx 为主机模式；

步骤 5：配置 SPIx_CR1.WIDTH，设置每帧数据的宽度；

步骤 6：配置 SPIx_CR1.LSBF，设置数据位收发顺序；

步骤 7：配置 SPIx_CR1.CPOL，设置时钟极性；

步骤 8：配置 SPIx_CR1.CPHA，设置时钟相位；

步骤 9：设置 SPIx_CR1.SSM 为 1，通过 SSI 寄存器决定从机选择输出值；

步骤 10：配置 SPIx_CR1.BR，设置 SCK 波特率；

步骤 11：设置 SPIx_CR1.EN 为 1，使能 SPIx 控制器；

步骤 12：设置 SPIx_SSI.SSI 为 0，通信开始；

步骤 13：查询等待 SPIx_ISR.TXE 标志位置 1，确认发送缓冲器为空；

步骤 14：将一帧虚拟数据写入 SPIx_DR 寄存器；

步骤 15：查询等待 SPIx_ISR.RXNE 标志位置 1，确认接收完成一帧数据；

步骤 16：读取 SPIx_DR 寄存器并保存数据；

步骤 17：重复步骤 13 至步骤 17，接收下一帧数据，直到全部数据接收完毕；

步骤 18：设置 SPIx_SSI.SSI 为 1，通信结束。

18.7 寄存器列表

SPI1 基地址：SPI1_BASE = 0x4001 3000

SPI2 基地址：SPI2_BASE = 0x4000 3800

表 18-5 SPI 寄存器列表

寄存器名称	寄存器地址	寄存器描述
SPIx_CR1	SPIx_BASE + 0x00	控制寄存器 1
SPIx_CR2	SPIx_BASE + 0x08	控制寄存器 2
SPIx_SSI	SPIx_BASE + 0x0C	从机选择寄存器
SPIx_IER	SPIx_BASE + 0x04	中断使能寄存器
SPIx_ISR	SPIx_BASE + 0x10	中断标志寄存器
SPIx_ICR	SPIx_BASE + 0x14	中断标志清除寄存器
SPIx_DR	SPIx_BASE + 0x18	数据寄存器

18.8 寄存器描述

有关寄存器描述里所使用的缩写，请参见 1 文档约定章节。

18.8.1 SPIx_CR1 控制寄存器 1

Address offset: 0x00 Reset value: 0x0000 1C04

位域	名称	权限	功能描述
31:19	RFU	-	保留位，请保持默认值
18	MISOHD	RW	从机 MISO 输出配置 0：MISO 始终为 COMS 输出 1：从机被选中时 MISO 为 COMS 输出，未被选中时为高阻输出
17	DMATX	RW	DMA 发送数据使能控制 0：禁止 1：使能
16	DMARX	RW	DMA 接收数据使能控制 0：禁止 1：使能
15:14	MODE	RW	通信模式配置 00：双向全双工 01：单工单发 10：单工单收 11：单线半双工
13:10	WIDTH	RW	每帧数据的宽度为 WIDTH+1，例如 0011：4bit 0100：5bit … 1110：15bit 1111：16bit
9	SSM	RW	从机选择配置（主机模式） 0：SPI_CS 引脚为输入模式，引脚低电平可选中本机 1：SPI_CS 引脚为输出模式，输出电平来自 SSI 从机选择配置（从机模式） 0：SPI_CS 引脚电平决定本机是否被选中 1：SSI 寄存器的值决定本机是否被选中
8	SMP	RW	主机模式延后采样 0：SPI 传统方式，主机接收数据按照 CPOL 和 CPHA 的设定进行采样 1：延后采样方式，采样时刻相比传统方式延后半个 SCK 周期
7	LSBF	RW	数据帧高低位顺序选择 0：最高有效位 MSB 收发在前 1：最低有效位 LSB 收发在前

位域	名称	权限	功能描述
6	EN	RW	使能控制 0：禁止 SPI 模块 1：使能 SPI 模块
5:3	BR	RW	主机模式 SCK 波特率配置 000：PCLK/2 001：PCLK/4 010：PCLK/8 011：PCLK/16 100：PCLK/32 101：PCLK/64 110：PCLK/128 111：保留
2	MSTR	RW	工作模式配置 0：从机模式 1：主机模式
1	CPOL	RW	串行时钟极性配置 0：待机时低电平 1：待机时高电平
0	CPHA	RW	串行时钟相位配置 0：前边沿采样 / 后边沿移位 1：前边沿移位 / 后边沿采样

18.8.2 SPIx_CR2 控制寄存器 2

Address offset: 0x08 Reset value: 0x0000 0000

位域	名称	权限	功能描述
31:1	RFU	-	保留位，请保持默认值
0	HDOE	RW	单线半双工时，数据发送 / 接收状态控制 1：仅发送 0：仅接收 注：仅在单线半双工通信时有效

18.8.3 SPIx_SSI 从机选择寄存器

Address offset: 0x0C Reset value: 0x0000 0001

<table>
<tr><th>位域</th><th>名称</th><th>权限</th><th>功能描述</th></tr>
<tr><td>31:1</td><td>RFU</td><td>-</td><td>保留位，请保持默认值</td></tr>
<tr><td>0</td><td>SSI</td><td>RW</td><td>从机选择，当 SPIx_CR1.SSM 为 1 时有效
<table>
<tr><th>SSM</th><th>MSTR</th><th>SSI</th><th>描述</th></tr>
<tr><td>0</td><td>-</td><td>-</td><td>无效</td></tr>
<tr><td rowspan="4">1</td><td rowspan="2">0(从机)</td><td>0</td><td>设置本机为选中态</td></tr>
<tr><td>1</td><td>设置本机为未选中态</td></tr>
<tr><td rowspan="2">1(主机)</td><td>0</td><td>SPIx_CS 引脚输出低</td></tr>
<tr><td>1</td><td>SPIx_CS 引脚输出高</td></tr>
</table></td></tr>
</table>

18.8.4 SPIx_IER 中断使能寄存器

Address offset: 0x04 Reset value: 0x0000 0000

位域	名称	权限	功能描述
31:8	RFU	-	保留位，请保持默认值
7	MODF	RW	模式错误中断使能控制 0：禁止 1：使能
6	SSERR	RW	从机模式下的从机选择错误中断使能控制 0：禁止 1：使能
5	OV	RW	接收缓冲上溢错误中断使能控制 0：禁止 1：使能
4	UD	RW	从机模式下发送缓冲下溢错误中断使能控制 0：禁止 1：使能
3	SSR	RW	从机选择输入上升沿中断使能控制 0：禁用 1：使能
2	SSF	RW	从机选择输入下降沿中断使能控制 0：禁用 1：使能
1	RXNE	RW	接收缓冲非空中断使能控制 0：禁用 1：使能
0	TXE	RW	发送缓冲空中断使能控制 0：禁用 1：使能

18.8.5 SPIx_ISR 中断标志寄存器

Address offset: 0x10 Reset value: 0x0000 0001

位域	名称	权限	功能描述
31:10	RFU	-	保留位，请保持默认值
9	SSLVL	RO	从机选择信号状态 0：从机选择信号为低电平 1：从机选择信号为高电平
8	BUSY	RO	总线忙标志 0：总线空闲 1：总线忙
7	MODF	RO	模式错误标志 0：正常 1：出错 注：主机模式下从机选择输入为低则触发 MODF
6	SSERR	RO	从机模式下的从机选择错误标志位 0：正常 1：出错
5	OV	RO	接收缓冲上溢错误标志位 0：正常 1：出错
4	UD	RO	从机模式下的发送缓冲下溢错误标志位 0：正常 1：出错
3	SSR	RO	从机选择输入上升沿标志位 0：未出现上升沿 1：出现了上升沿
2	SSF	RO	从机选择输入下降沿标志位 0：未出现下降沿 1：出现了下降沿
1	RXNE	RO	接收缓冲非空标志位 0：接收缓冲空 1：接收缓冲非空 注：对 DR 寄存器的读操作或对 ICR.RXNE 写 0 均可以清除该标志
0	TXE	RO	发送缓冲空标志位 0：发送缓冲非空 1：发送缓冲空

18.8.6 SPIx_ICR 中断标志清除寄存器

Address offset: 0x14 Reset value: 0x0000 00FF

位域	名称	权限	功能描述
31:8	RFU	-	保留位，请保持默认值
7	MODF	R1W0	模式错误标志清除 W0：清除模式错误标志 W1：无功能
6	SSERR	R1W0	从机模式下的从机选择错误标志清除 W0：清除从机模式下的从机选择错误标志 W1：无功能
5	OV	R1W0	接收缓冲上溢错误标志清除 W0：清除接收缓冲上溢错误标志 W1：无功能
4	UD	R1W0	从机模式下的发送缓冲下溢错误标志清除 W0：清除从机模式下的发送缓冲下溢错误标志 W1：无功能
3	SSR	R1W0	从机选择输入上升沿标志清除 W0：清除从机选择输入上升沿标志 W1：无功能
2	SSF	R1W0	从机选择输入下降沿标志清除 W0：清除从机选择输入下降沿标志 W1：无功能
1	RXNE	R1W0	接收缓冲非空标志清除 W0：清除接收缓冲非空标志 W1：无功能
0	FLUSH	R1W0	清空发送缓冲区和移位寄存器 W0：清空发送缓冲区和移位寄存器（清除后 ISR.TXE 置 1） W1：无功能

18.8.7 SPIx_DR 数据寄存器

Address offset: 0x18 Reset value: 0x0000 0000

位域	名称	权限	功能描述
31:16	RFU	-	保留位，请保持默认值
15:0	DR	RW	写入为待发送的数据 读出为接收到的数据

19 I2C 接口

19.1 概述

CW32F020 内部集成 2 个 I2C 控制器，能按照设定的传输速率（标准，快速，高速）将需要发送的数据按照 I2C 规范串行发送到 I2C 总线上，并对通信过程中的状态进行检测，另外还支持多主机通信中的总线冲突和仲裁处理。

19.2 主要特性

- 支持主机发送 / 接收，从机发送 / 接收四种工作模式
- 支持时钟延展 (时钟同步) 和多主机通信冲突仲裁
- 支持标准 (100Kbps)/ 快速 (400Kbps)/ 高速 (1Mbps) 三种工作速率
- 支持 7bit 寻址功能
- 支持 3 个从机地址
- 支持广播地址
- 支持输入信号噪声过滤功能
- 支持中断状态查询功能

19.3 协议描述

I2C 总线使用两根信号线（数据线 SDA 和时钟线 SCL）在设备间传输数据。SCL 为单向时钟线，固定由主机驱动。SDA 为双向数据线，在数据传输过程中由收发两端分时驱动。

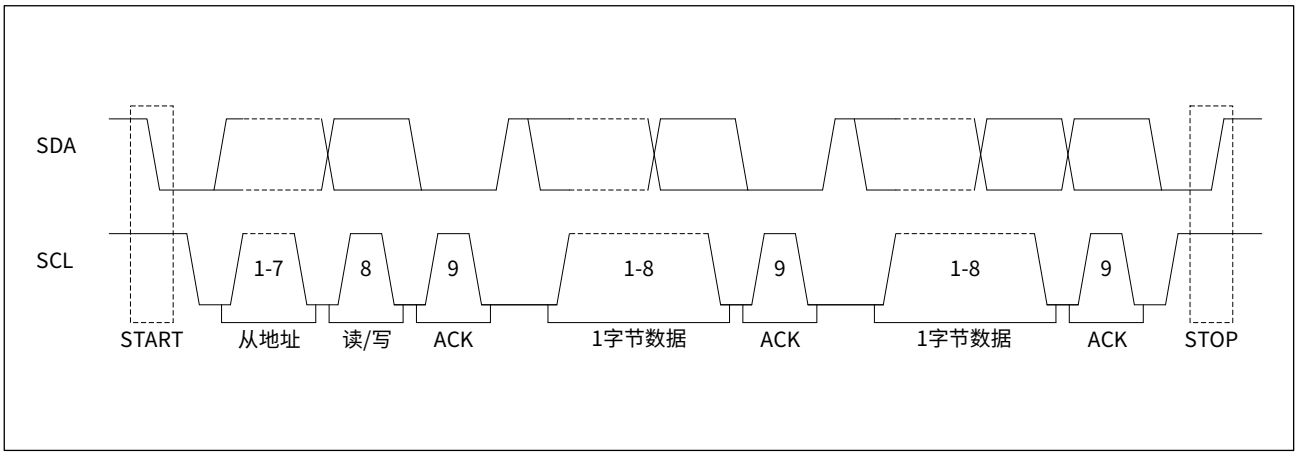
I2C 总线上可以连接多个设备，所有设备在没有进行数据传输时都处于空闲状态（未寻址从机接收模式），任一设备都可以作为主机发送 START 起始信号来开始数据传输，在 STOP 停止信号出现在总线上之前，总线一直处于被占用状态。

I2C 通信采用主从结构，并由主机发起和结束通信。主机通过发送 START 起始信号来发起通信，之后发送 SLA+W/R 共 8bit 数据（其中，SLA 为 7bit 从机地址，W/R 为读写位），并在第 9 个 SCL 时钟释放 SDA 总线，对应的从机在第 9 个 SCL 时钟占用 SDA 总线并输出 ACK 应答信号，完成从机寻址。此后根据主机发送的第 1 字节的 W/R 位来决定数据通信的发端和收端，发端每发送 1 个字节数据，收端必须回应 1 个 ACK 应答信号。数据传输完成后，主机发送 STOP 信号结束本次通信。

19.3.1 协议帧格式

标准 I2C 传输协议帧包含四个部分：起始信号 (START) 或重复起始信号 (Repeated START) 信号，从机地址及读写位，数据传输，停止信号 (STOP)。如下图所示。

图 19-1 I2C 协议帧



- 起始信号 (START)

  当总线处于空闲状态时（SCL 和 SDA 线同时为高），SDA 线上出现由高到低的下降沿信号，则被定义为起始信号。主机向总线发出起始信号后开始数据传输，并占用总线。

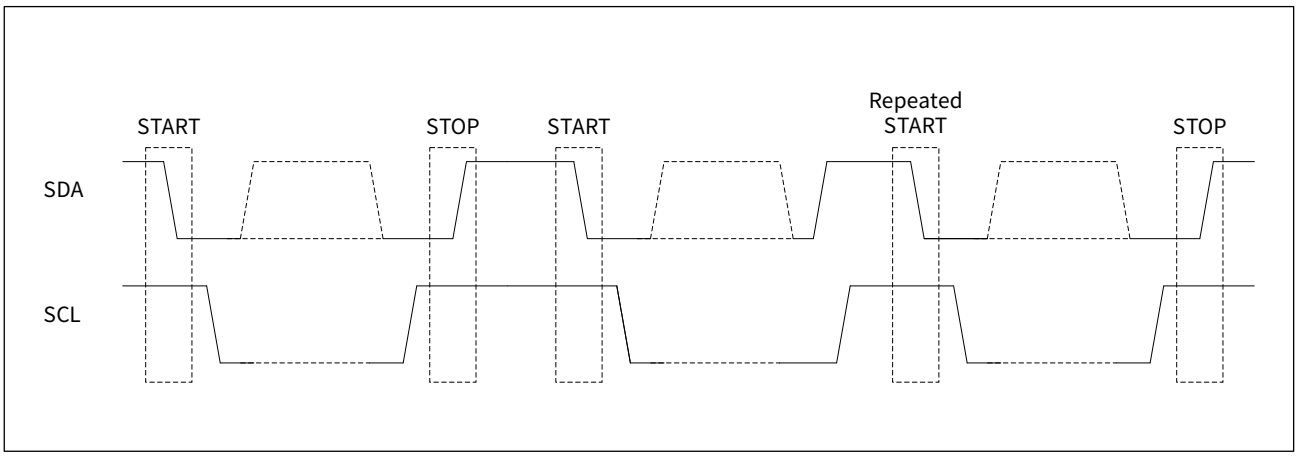
- 重复起始信号 (Repeated START)

  当一个起始信号后未出现停止信号之前，出现了新的起始信号，新的起始信号被定义为重复起始信号。在主机发送停止信号前，SDA 总线一直处于占用状态，其它主机无法占用总线。

- 停止信号 (STOP)

  当 SCL 线为高时，SDA 线上出现由低到高的上升沿信号，则被定义为停止信号。主机向总线发出停止信号以结束数据传输，并释放总线。

图 19-2　起始和停止信号



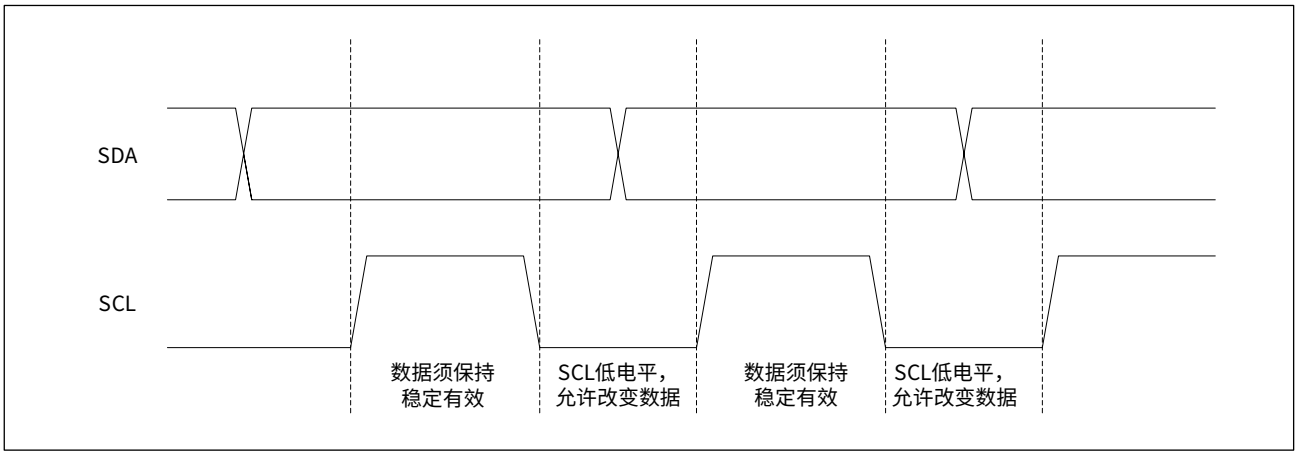
- 从机地址及读写位

  当起始信号产生后，主机开始传输第 1 字节数据：7 bit 从机地址 + 读写位。读写位（1：读；0：写）控制总线上数据传输方向。被寻址的从机在第 9 个 SCL 时钟周期内占用 SDA 总线，并将 SDA 置为低电平作为 ACK 应答。

- 数据传输

  主机在 SCL 线上输出串行时钟信号，主从机通过 SDA 线进行数据传输。数据传输过程中，1 个 SCL 时钟脉冲传输 1 个数据位（最高有效位 MSB 在前），且 SDA 线上的数据只在 SCL 为低时改变，每传输 1 个字节跟随 1 个应答位。如下图所示：

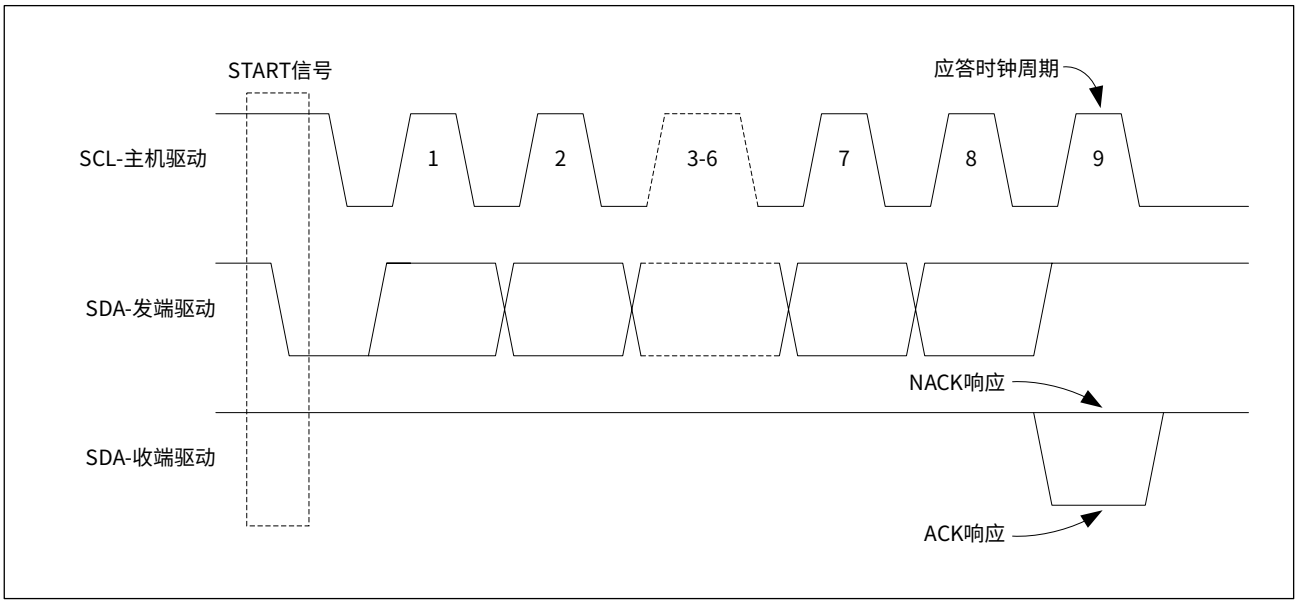
图 19-3　数据传输



19.3.2 传输应答

在总线上传输数据时，发端每传输完 1 个字节数据，在第 9 个 SCL 时钟周期发端放弃对 SDA 的控制，收端须在第 9 个 SCL 时钟周期回复 1 个应答位：接收成功，发送 ACK 应答，接收异常发送 NACK 应答。

图 19-4 传输应答



19.3.3 冲突检测与仲裁

在多主机通信系统中，总线上的每个节点都有从机地址。每个节点可以作为从节点被其它节点访问，也可以作为主节点向其它的节点发送控制字节和传送数据。如果有两个或两个以上的节点同时向总线发出起始信号并开始传输数据，就会造成总线冲突。I2C 控制器内置一个仲裁器，可对 I2C 总线冲突进行检测和仲裁，以保证数据通信的可靠性和完整性。

- 冲突检测原理

  在物理实现上， SDA 和 SCL 引脚电路结构相同，引脚的输出驱动与输入缓冲连在一起。输出结构为漏极开路的场效应管、输入结构为高输入阻抗的同相器。基于该结构：

  1. 由于 SDA、SCL 为漏极开路结构，借助于外部的上拉电阻实现了信号的“线与”逻辑；
  2. 设备向总线写数据的同时读取数据，可用来检测总线冲突，实现 “时钟同步”和“总线仲裁”。

  根据“线与”逻辑，如果 2 个主机同时发送逻辑 1 或逻辑 0，则 2 个主机都检测不到冲突，需要等到下一位数据发送再继续检测冲突；如果 2 个主机一个发送逻辑 1，一个发送逻辑 0，此时总线上为逻辑 0，发送逻辑 1 的主机检测到冲突，发送逻辑 0 的主机没有检测到冲突。

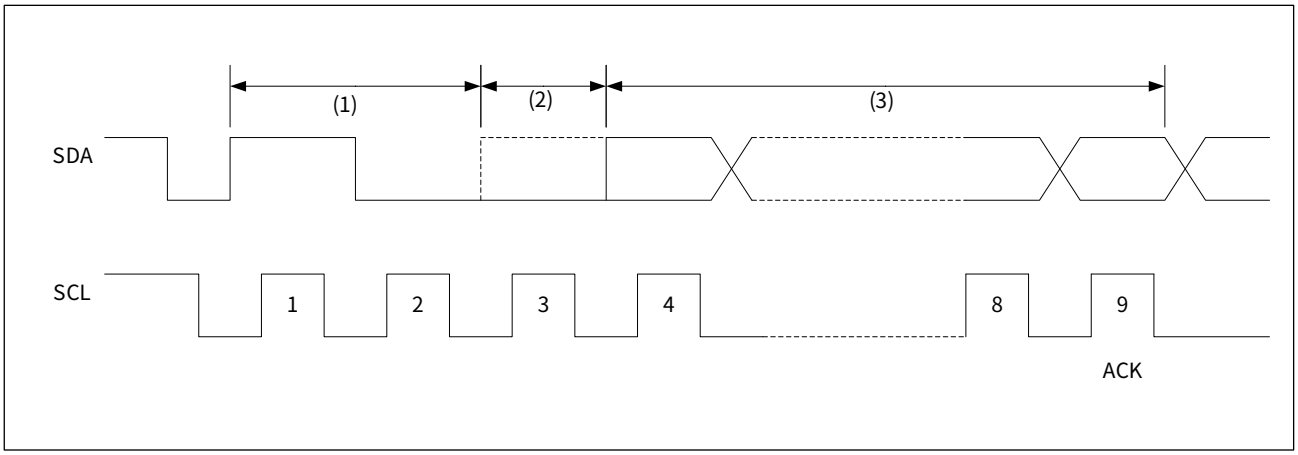
- 冲突仲裁原理

  当主机检测到总线冲突后，该主机丢失仲裁，退出主机发送模式，进入未寻址从机模式，释放 SDA 数据线，并回到地址侦测状态，之后根据接收到的 SLA+W/R 进入相应的从机模式 (SLA 地址匹配进入已寻址从机模式，SLA 地址不匹配则进入未寻址从机模式)。仲裁失败的主机，仍会发送 SCL 串行时钟，直到当前字节传输结束。

  当主机没有检测到总线冲突，该主机赢得仲裁，继续主导本次数据传输，直到通信完成。

下图为一个 I2C 总线上 A 主机和 B 主机发送冲突和仲裁示意图（假设 A，B 两个主机的发送时钟同步）：

图 19-5 冲突检测及仲裁



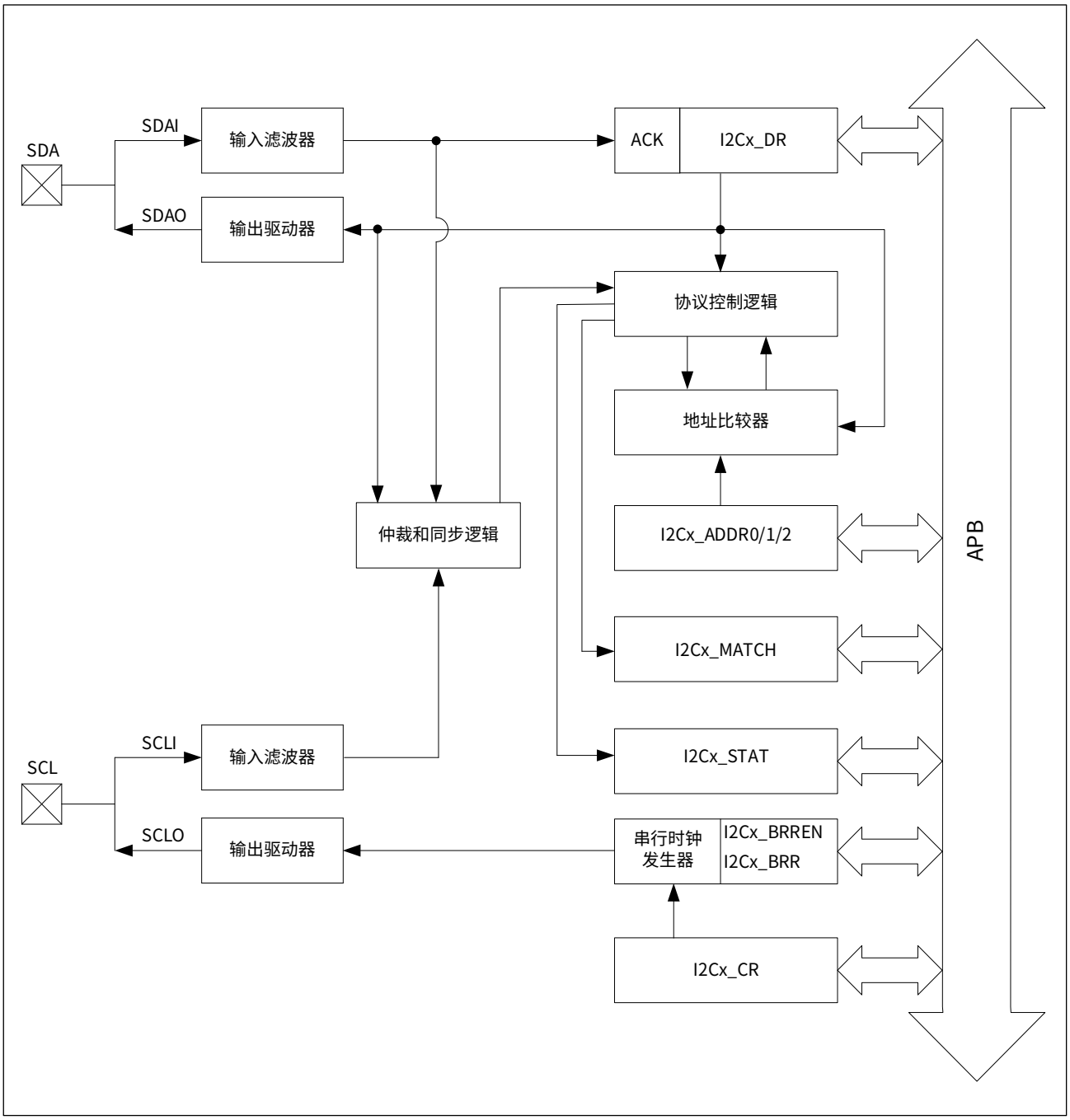
1. 在阶段（1），A 主机和 B 主机均先后发送逻辑 1 和逻辑 0，由于数据相同且同步，2 个主机都不会检测到总线冲突。
2. 在阶段（2），A 主机发送逻辑 1，B 主机发送逻辑 0，SDA 总线上数据为逻辑 0。A 主机控制器检测到数据错误，退出发送竞争，即丢失仲裁，A 主机进入未被寻址的从机接收模式。B 主机没有检测到总线冲突，赢得仲裁，继续本次数据传输。
3. 在阶段（3），A 主机处于未被寻址的从机接收模式，但仍产生 SCL 时钟脉冲，直到当前字节传输结束。此后 A 主机 I2C 控制器不再发送时钟信号，B 主机由于赢得仲裁，SCL 和 SDA 都由 B 主机来主导控制传输。

19.4 功能描述

19.4.1 功能框图

I2C 模块主要包括时钟发生器、输入滤波器、地址比较器、协议控制逻辑、仲裁和同步逻辑、以及相关寄存器等。其功能框图如下图所示：

图 19-6 I2C 功能框图



CW32F020 支持用户灵活选择 GPIO 作为 I2C 通信引脚，如下表所示：

表 19-1 I2C 引脚选择

	I2C1	I2C2
SCL	PA09/PA14/PB06/PB08/PB10/PF01/PF06	PA01/PA04/PA11/PA13/PB00/PB10/PB13/PF06
SDA	PA10/PA13/PB07/PB09/PB11/PF00/PF07	PA02/PA05/PA12/PA14/PB01/PB11/PB14/PF07

19.4.2 串行时钟发生器

串行时钟发生器用来产生 I2C 通信的波特率时钟 SCL。串行时钟发生器采用 PCLK 作为输入时钟，通过 1 个 8bit 的计数器计数，输出所需波特率的 I2C 时钟信号。

SCL 时钟频率计算公式：

$$f_{SCL} = f_{PCLK} / 8 / (BRR + 1)$$

其中，BRR 通过波特率计数器配置寄存器 I2Cx_BRR 配置，BRR 有效范围为 1 ~ 255。

串行时钟发生器的计数器计数由 I2Cx_BRREN 寄存器的 EN 位域使能，EN 为 1 使能，为 0 禁止。主机时应设置 EN 为 1，从机时该位不影响。

PCLK、BRR 组合和 SCL 时钟频率的对应关系如下表所示：

表 19-2 I2C 传输速率和配置举例

f_{SCL}(kHz) / f_{PCLK} (kHz)	I2Cx_BRR						
	1	2	3	4	5	6	7
1000	62	41	31	25	20	17	15
2000	125	83	62	50	41	35	31
4000	250	166	125	**100**	83	71	62
6000	375	250	187	150	124	107	93
8000	500	333	250	200	166	142	125
10000	625	416	312	250	208	178	156
12000	750	500	375	300	250	214	187
14000	875	583	437	350	291	250	218
16000	**1000**	666	500	**400**	333	285	250

19.4.3 输入滤波器

输入滤波器可对 SDA 和 SCL 输入信号进行滤波处理，小于 1 个 PCLK 周期的尖峰脉冲信号会被滤除。

输入滤波器可配置为两种模式：简单滤波模式和高级滤波模式。简单滤波具有更快的通信速率；高级模式则具有更高的抗干扰性能。通过 I2C 控制寄存器 I2Cx_CR 的 FLT 位域配置滤波模式，设置 I2Cx_CR.FLT 为 1 则为简单滤波模式，设置 I2Cx_CR.FLT 为 0 则为高级滤波模式。

当作为主机时，如果 BRR 的值小于或等于 9，则应设置 I2Cx_CR.FLT 为 1；如果 BRR 的值大于 9，则应设置 I2Cx_CR.FLT 为 0。

当作为从机时，如果 PCLK 与 SCL 的频率比值小于或等于 40，则应设置 I2Cx_CR.FLT 为 1；如果 PCLK 与 SCL 的频率比值大于 40，则应设置 I2Cx_CR.FLT 为 0。

19.4.4 地址比较器

I2C 总线上各设备都有从机地址，且各从机地址均不同。主机根据从机地址寻址从机，从机通过地址比较器自动检测主机发送的 7bit 寻址地址与本机地址是否匹配，以确定是否与主机通信。

I2C 控制器支持 3 个可编程的从机地址，具体地址信息通过从机地址寄存器 I2Cx_ADDR0 / I2Cx_ADDR1 / I2Cx_ADDR2 进行配置。

从机的地址比较器将接收到的 7bit 寻址地址和 3 个从机地址以及广播地址（0x00）相比较。如果符合 4 个地址中的任何一个，则认为地址匹配，同时 I2C 中断标志位 I2Cx_CR.SI 会被置 1，并产生一个中断请求。应用程序可通过查询从机地址匹配寄存器 I2Cx_MATCH 获取匹配成功的地址序号。

如果地址匹配到 I2Cx_ADD0 / 1 / 2，则从机进入相应的已寻址从机接收模式（接收到 SLA+W）或者已寻址从机发送模式（接收到 SLA+R）；如果地址匹配到广播地址 0x00（接收到 SLA+W），则从机进入广播接收模式。

19.4.5 仲裁和同步逻辑

19.4.5.1 SCL 同步

I2C 支持时钟同步（时钟延展）功能，SCL 时钟低电平的时间由 SCL 时钟低电平宽度最长的器件决定，而 SCL 时钟高电平时间由时钟高电平宽度最短的器件决定。

如果从机希望主机降低传输速度，可以在收到数据并回应 ACK 后，保持 I2Cx_CR.SI 为 1，则从机 I2C 控制器将保持 SCL 为低电平状态，以此来通知主机。当主机准备下一个字节数据传输（发送或者接收）时检测到 SCL 的电平被拉低，则进行等待，直到从机完成操作并将 I2Cx_CR.SI 清 0，从机控制器释放 SCL 的拉低控制，主机检测到 SCL 为高电平后继续下一个字节数据的传输。

19.4.5.2 SDA 仲裁

I2C 支持 SDA 冲突检测和仲裁，可以保证在多个主机企图控制 I2C 总线时，I2C 总线上的数据不被破坏。

每个主机发送数据时，都会同时比较总线上的数据与自己发送的数据是否一致，不一致则认为检测到总线冲突，会退出发送竞争，即丢失仲裁。丢失仲裁的主机会立即切换到未被寻址的从机状态，以确保自身能被仲裁成功的主机寻址到。丢失仲裁的主机会继续输出 SCL 串行时钟，直到当前字节传输完成。

SDA 仲裁一般发生在主机发送 SLA+W/R 数据阶段，如果两个主机同时向一个从机发送数据，即两个主机发送的从机地址相同，则仲裁会在第二个字节持续。

19.4.6 应答控制

I2C 数据传输的收端必须在每个字节的第 9 个 SCL 时钟周期给发端进行 ACK 或者 NACK 应答，发端通过该应答位来获取收端当前状态：回应 ACK 应答，则表明收端已正确接收该字节数据，可以继续接收下一字节数据；回应 NACK 一般表示收端已不再接收任何数据。

收端发送 ACK 还是 NACK，由收端的 I2C 控制寄存器 I2Cx_CR 的 AA 位域来控制。当设置 I2Cx_CR.AA 为 1 时，I2C 模块每收到 1 字节数据后会回应 ACK 应答，当设置 I2Cx_CR.AA 为 0 时，I2C 模块每收到 1 字节数据后回应 NACK 应答。

在主机接收数据过程中，主机作为通信发起方，控制着收发字节个数，主机（收端）在最后一个字节数据接收完成后回应 NACK 应答给从机（发端），从机收到 NACK 应答后将切换为未寻址从机接收模式，并释放 SDA 总线，以便主机发送 STOP 停止信号或 Repeated START 重复起始信号。

在从机发送数据过程中，如果自身的 I2Cx_CR.AA 应答控制位被应用程序清零，则从机在发送完最后 1 字节有效数据后，将自身切换为未寻址从机接收模式，并释放 SDA 总线，主机从总线上读数据将得到 0xFF。此时主机应能判断从机处于无响应状态，并发送 STOP 停止信号或 Repeated START 重复起始信号。

在从机接收数据过程中，如果回应 NACK 给主机（发端），则表示从机（收端）主动结束本次通信，不再接收主机发送的数据，且将自身切换为未寻址从机接收模式，并释放 SDA 总线，此时主机应发送 STOP 停止信号或 Repeated START 重复起始信号。

19.4.7 I2C 中断

I2C 控制寄存器 I2Cx_CR 的 SI 位域为中断标志位。当 I2C 状态寄存器 I2Cx_STAT 的 STAT 位域值发生改变（变成 0xF8 除外）时，I2Cx_CR.SI 标志位就会被置位，同时产生中断请求。

在用户 I2C 中断服务程序中，应查询 I2C 状态寄存器 I2Cx_STAT 的 STAT 位域值获取 I2C 总线的状态，以确定中断产生原因。设置 I2Cx_CR.SI 为 0 清除该标志位。

19.4.8 工作模式

I2C 控制器支持 4 种工作模式：主机发送模式、主机接收模式、从机发送模式、从机接收模式。另外还支持广播接收模式，其工作方式和从机接收模式类似。

19.4.8.1 主机发送模式

主机发送模式下，主机主动发送多个字节到从机。

SCL 串行时钟由主机控制产生，因此需要先根据传输波特率设置 I2Cx_BRR 寄存器并设置 I2Cx_BRREN 寄存器的 EN 位域为 1，然后启动传输。

主机设置 I2Cx_CR.STA 为 1，通知控制器发送 START 起始信号，控制器收到通知后检测总线是否空闲，当总线空闲时，主机控制器向总线发送一个 START 起始信号。如果发送成功，状态码 I2Cx_STAT 变为 0x08，中断标志位 I2Cx_CR.SI 被置 1。

主机发送完 START 起始信号后，需要软件设置 I2Cx_CR.STA 为 0，然后将从机地址和写标志位（SLA+W）写入到 I2C 数据寄存器 I2Cx_DR；清零 I2Cx_CR.SI 位，主机控制器将发送 SLA+W 到 I2C 总线上；当主机发送完 SLA+W 并收到从机 ACK 应答信号后，状态码 I2Cx_STAT 变为 0x18，中断标志位 I2Cx_CR.SI 被置 1。

此后主机根据应用需要，发送多个字节的用户自定义数据并检测 ACK 应答信号，每个字节的发送过程和发送 SLA+W 数据帧类似。

主机在发送数据过程中，如果收到 NACK 应答信号，表明从机不再接收主机发送的数据（从机的 I2Cx_CR.AA 被清零），主机应发送 STOP 信号结束本次数据传输，或者发送 Repeated START 重复起始信号开启新一轮传输。

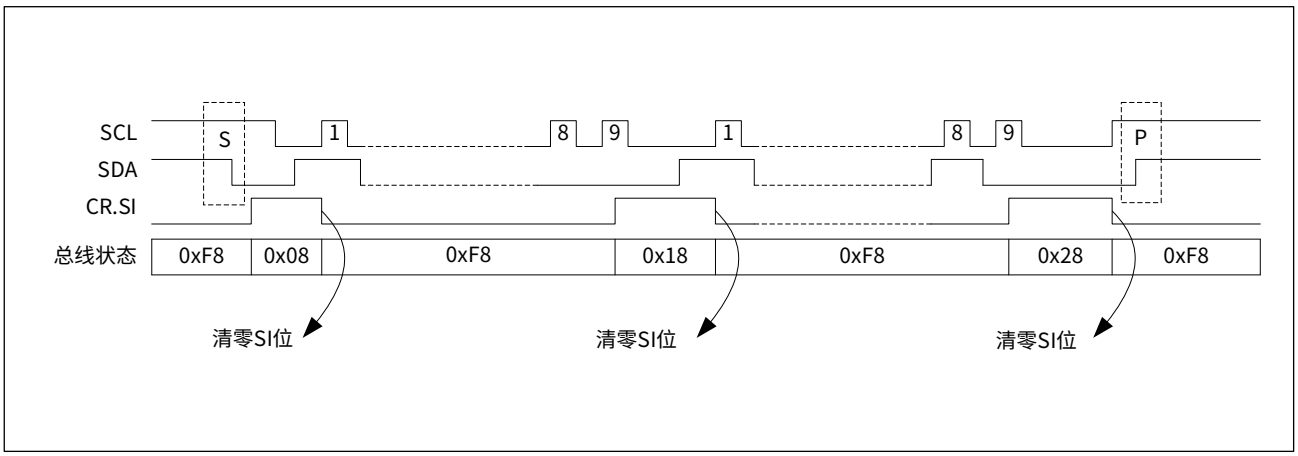
当主机发送完成所有数据后，设置 I2Cx_CR.STO 为 1，通知控制器数据已发送完成，待发送 STOP 停止信号。清零 I2Cx_CR.SI 位，主机控制器将发送 STOP 停止信号到 I2C 总线上，完成本次数据传输，释放总线。

当主机发送完成所有数据后，也可以不发送 STOP 停止信号，而是直接发送 Repeated START 重复起始信号，继续占用总线进行新一轮数据传输。

当主机由于总线冲突丢失仲裁时，会进入未寻址从机接收模式（状态码 I2Cx_STAT = 0x38）。

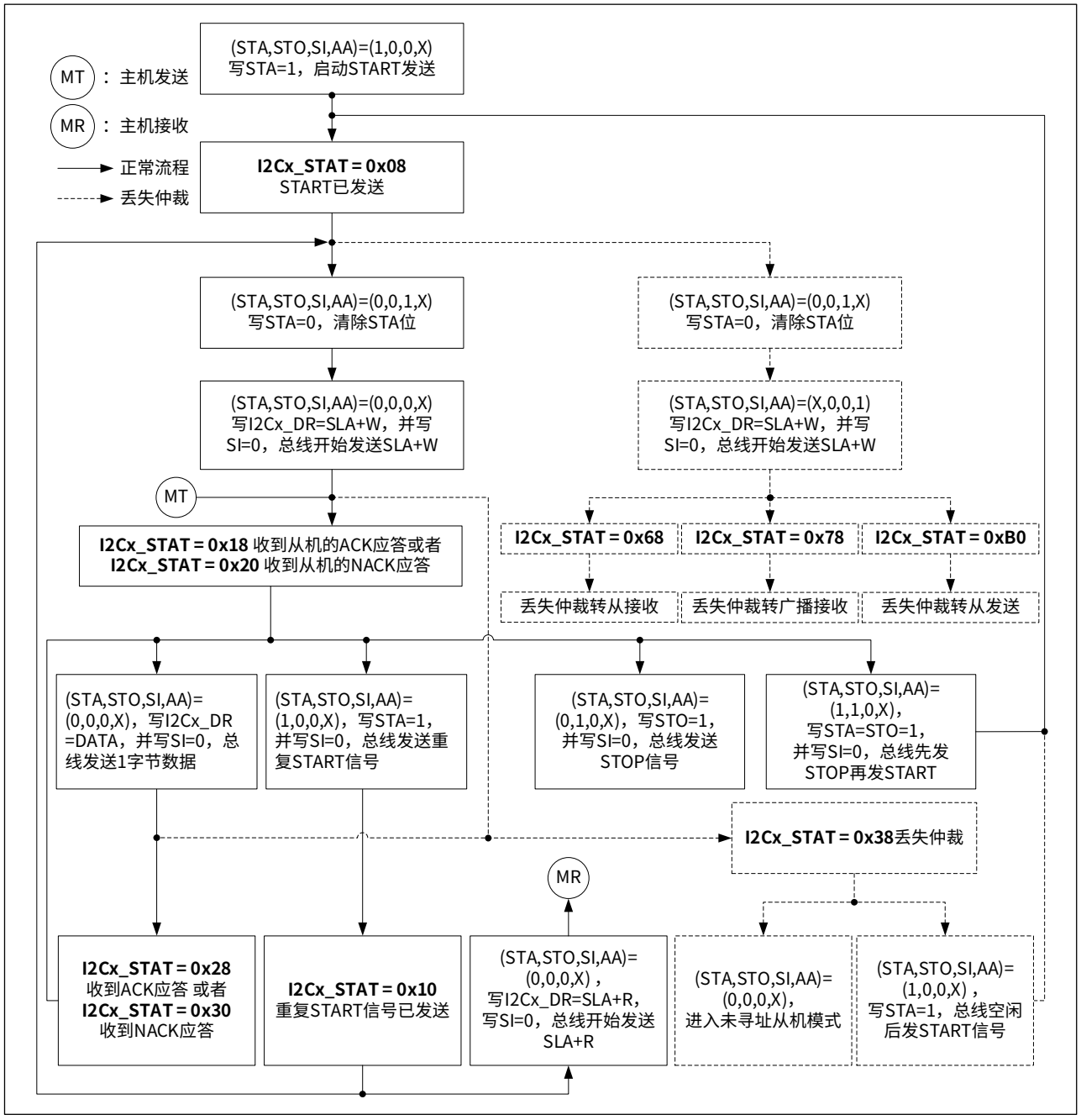
主机发送模式下状态同步图如下图所示：

图 19-7 主机发送模式状态同步图



主机发送模式下数据发送流程图及状态寄存器值如下图所示：

图 19-8　主机发送模式流程和寄存器状态



19.4.8.2 主机接收模式

主机接收模式用于接收从机发送的多个数据，主机每接收到 1 字节数据后会回应 ACK 应答信号。

SCL 串行时钟由主机控制产生，因此需要先根据传输波特率设置主机的 I2Cx_BRR 寄存器并设置 I2Cx_BRREN 寄存器的 EN 位域为 1，然后启动传输。

主机设置 I2Cx_CR.STA 为 1，通知控制器发送 START 起始信号，控制器收到通知后检测总线是否空闲，当总线空闲时，主机控制器向总线发送一个 START 起始信号。如果发送成功，状态码 I2Cx_STAT 变为 0x08，中断标志位 I2Cx_CR.SI 被置 1。

主机发送完 START 起始信号后，需要软件设置 I2Cx_CR.STA 为 0，然后将从机地址和读标志位（SLA+R）写入到 I2C 数据寄存器 I2Cx_DR；清零 I2Cx_CR.SI 标志位，主机控制器将发送 SLA+R 到 I2C 总线上；当主机发送完 SLA+R 并收到从机 ACK 应答信号后，状态码 I2Cx_STAT 变为 0x40，中断标志位 I2Cx_CR.SI 被置 1。

此后，主机设置 I2Cx_CR.AA 为 1，并清零 I2Cx_CR.SI 位，开始接收从机发送的数据，每接收完 1 字节数据后，都要回复一个 ACK 应答信号。在主机接收过程中，应注意：

1. 为保证每接收到 1 字节数据后都能正确产生 I2Cx_CR.SI 中断信号，需要在收到 1 字节数据后及时将 I2Cx_CR.SI 位清除。
2. 在接收最后一个字节前需要将 I2Cx_CR.AA 清零，即主机在接收到最后一个字节时不产生 ACK 应答信号，以此来通知从机停止数据发送。

主机在接收数据过程中，如果从机由于某种原因不再发送主机所需要的数据（从机的 I2Cx_CR.AA 被清零），主机后续将收到全 1 信号，此时主机需要在应用层对数据进行判决，判定从机为无响应状态，应发送 STOP 停止信号来结束本次传输，或发送 Repeated START 重复起始信号开启新一轮传输。

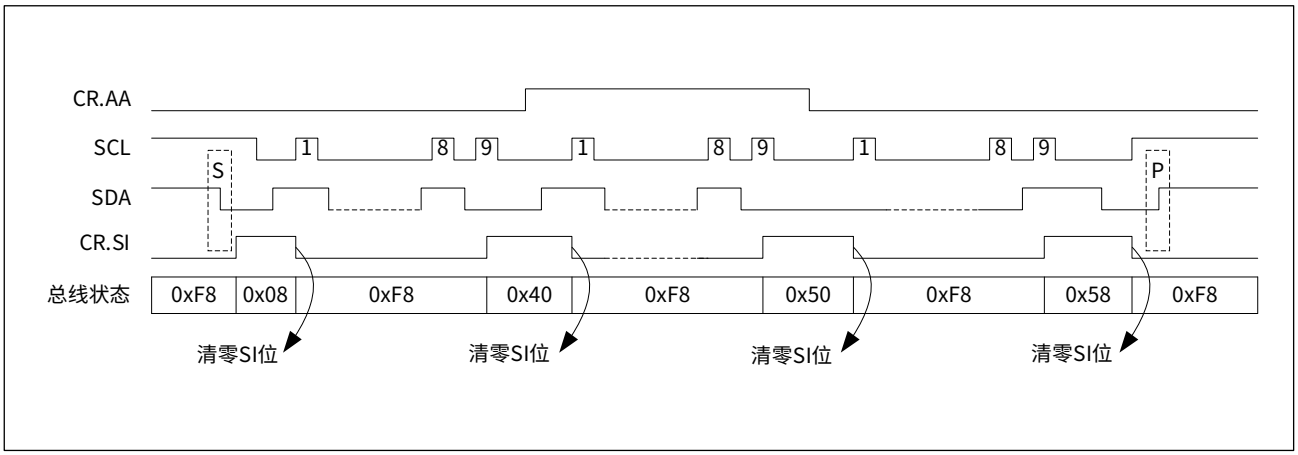
当主机接收完成所有数据后，设置 I2Cx_CR.STO 为 1，通知控制器数据已接收完成，待发送 STOP 停止信号。清零 I2Cx_CR.SI 位，主机控制器发送 STOP 停止信号到 I2C 总线上，完成本次数据传输，释放总线。

当主机接收完成所有数据后，也可以不发送 STOP 停止信号，而是直接发送 Repeated START 重复起始信号，继续占用总线进行新一轮数据传输。

当主机由于总线冲突丢失仲裁时，会进入未寻址从机接收模式（状态码 I2Cx_STAT = 0x38）。

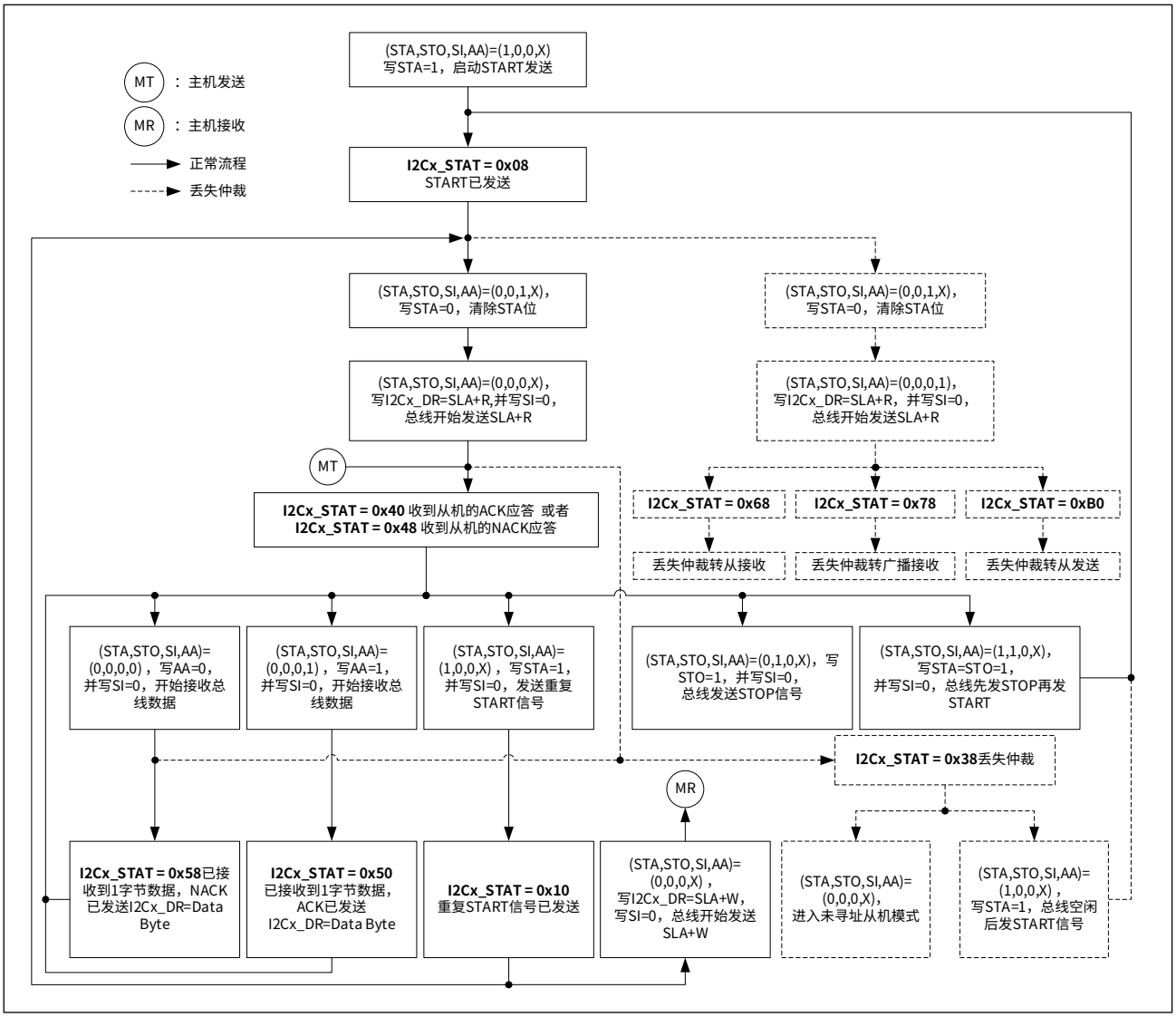
主机接收模式下状态同步图如下图所示：

图 19-9 主机接收模式状态同步图



主机接收模式下数据接收流程图及状态寄存器值如下图所示：

图 19-10　主机接收模式流程和寄存器状态



19.4.8.3 从机接收模式

从机接收模式下，从机接收主机发送的数据。

在传输之前，从机需要设定从机地址：将从机地址写入到 3 个从机地址寄存器 I2Cx_ADDR0、I2Cx_ADDR1、I2Cx_ADDR2 任意 1 个中；设置 I2Cx_CR.AA 为 1 以响应主机的寻址；I2Cx_BRR 寄存器设置值无效，不用设置。

完成上述初始化工作后，从机进入空闲状态（未寻址从机接收模式），等待被主机发送的写信号（SLA+W）寻址。当从机接收到 SLA+W 后，如果地址匹配到 I2Cx_ADDR0/1/2，则从机回应 ACK 应答，并进入已寻址从机接收模式，状态码 I2Cx_STAT 变为 0x60，中断标志位 I2Cx_CR.SI 同时被置 1。此时必须清除 I2Cx_CR.SI 位，以便从总线上接收主机发送的数据。

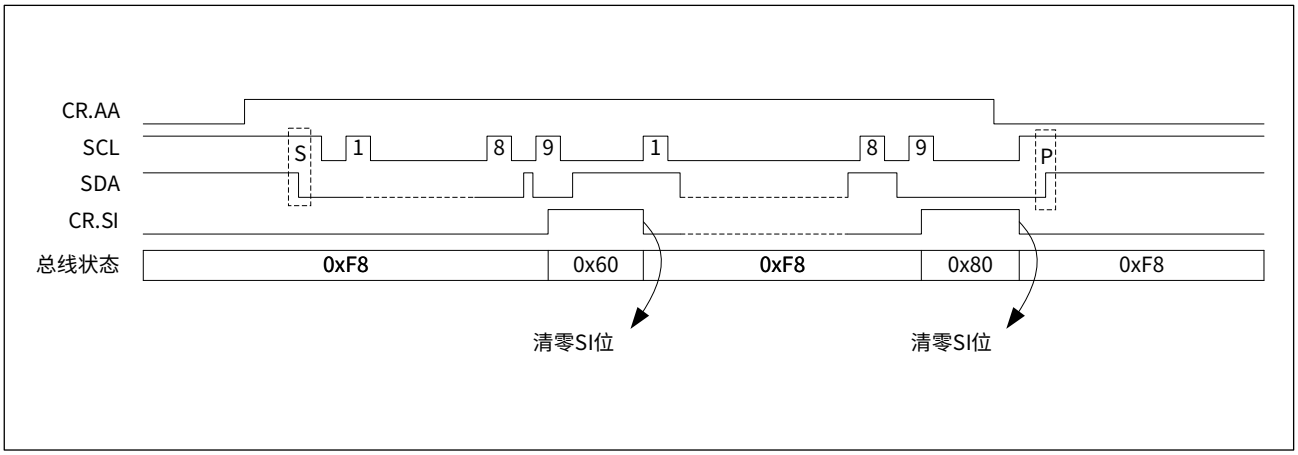
从机每接收到 1 字节数据都要回应 1 个 ACK 应答，应用程序读取完该字节数据后，必须将 I2Cx_CR.SI 位清零，为接收下一字节数据做好准备。

从机在接收数据过程中，如果 I2Cx_CR.AA 被清零，则从机将在接收到下一字节时返回 NACK 信号，从机自身状态也切换到未寻址从机接收模式，结束与主机的通信，不再接收数据，且 I2Cx_DR 寄存器保持之前接收到的数据。由该特性可知，从机应用程序可通过设置 I2Cx_CR.AA 为 0 主动将从机从已寻址从机接收模式切换到未寻址从机接收模式。

当主机在 SLA+ 读写阶段由于总线冲突丢失仲裁时会进入未寻址从机接收模式，之后如果接收到符合本机地址的 SLA+W 并回应 ACK 后（状态码 I2Cx_STAT = 0x68），则会进入已寻址从机接收模式。

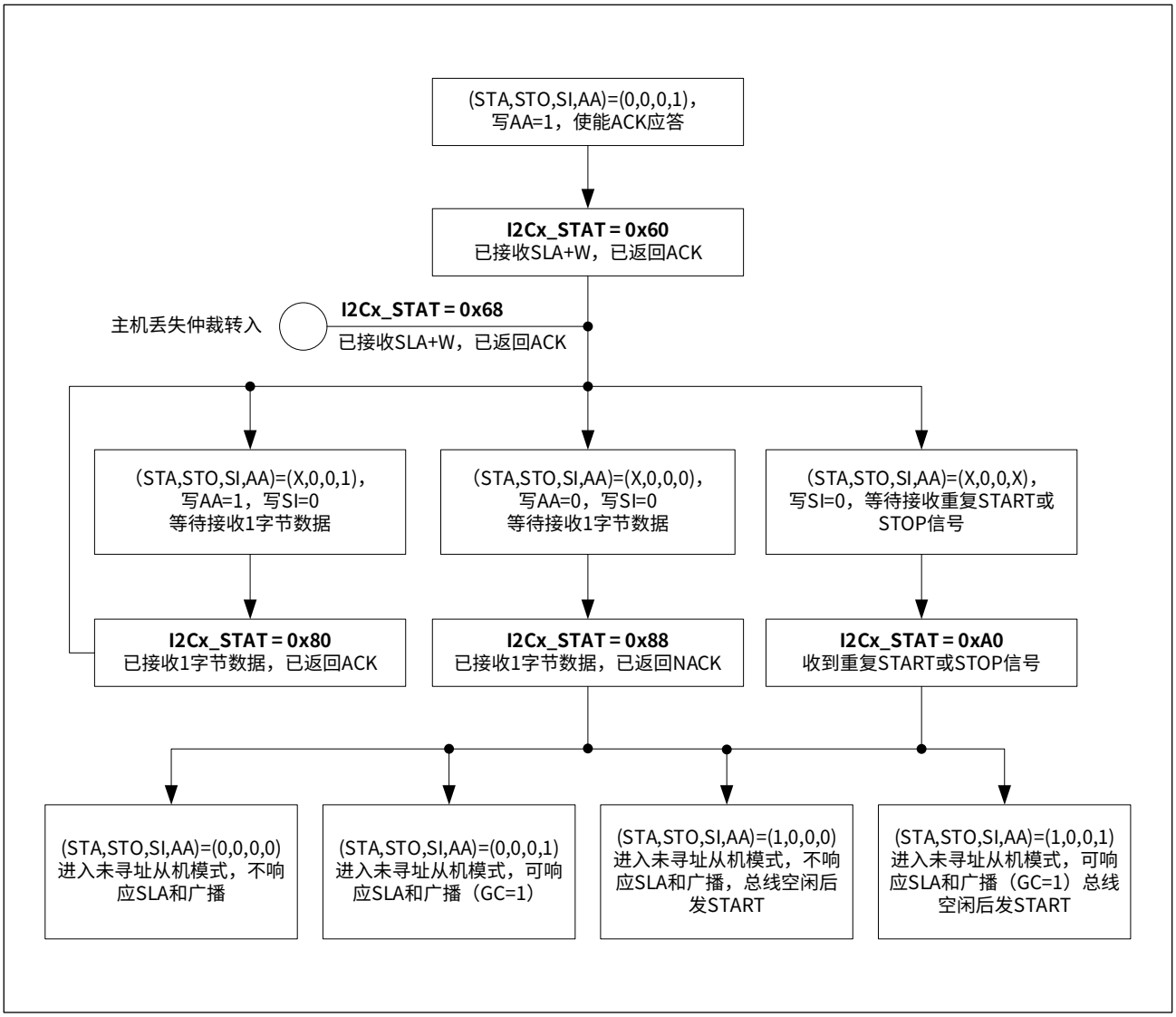
从机接收模式下状态同步图如下图所示：

图 19-11 从机接收模式状态同步图



从机接收模式下数据接收流程图及状态寄存器值如下图所示：

图 19-12　从机接收模式流程和寄存器状态



19.4.8.4 从机发送模式

从机发送模式下，数据由从机发送给主机。

在传输之前，从机需要设定从机地址：将从机地址写入到 3 个从机地址寄存器 I2Cx_ADDR0、I2Cx_ADDR1、I2Cx_ADDR2 任意 1 个中；设置 I2Cx_CR.AA 为 1 以响应主机的寻址；I2Cx_BRR 寄存器设置值无效，不用设置。

完成上述初始化工作后，从机进入空闲状态（未寻址从机接收模式），等待被主机发送的读信号（SLA+R）寻址。当从机接收到 SLA+R 后，如果地址匹配到 I2Cx_ADDR0/1/2，则从机回应 ACK 应答，并进入已寻址从机发送模式，状态码 I2Cx_STAT 变为 0xA8，中断标志位 I2Cx_CR.SI 同时被置 1。此时应及时将待发送的数据写入 I2Cx_DR 寄存器，并清除 I2Cx_CR.SI 位，等待发送 1 字节数据，并在每个字节数据发送完成后进行 ACK 应答确认，直到全部数据发送完毕。

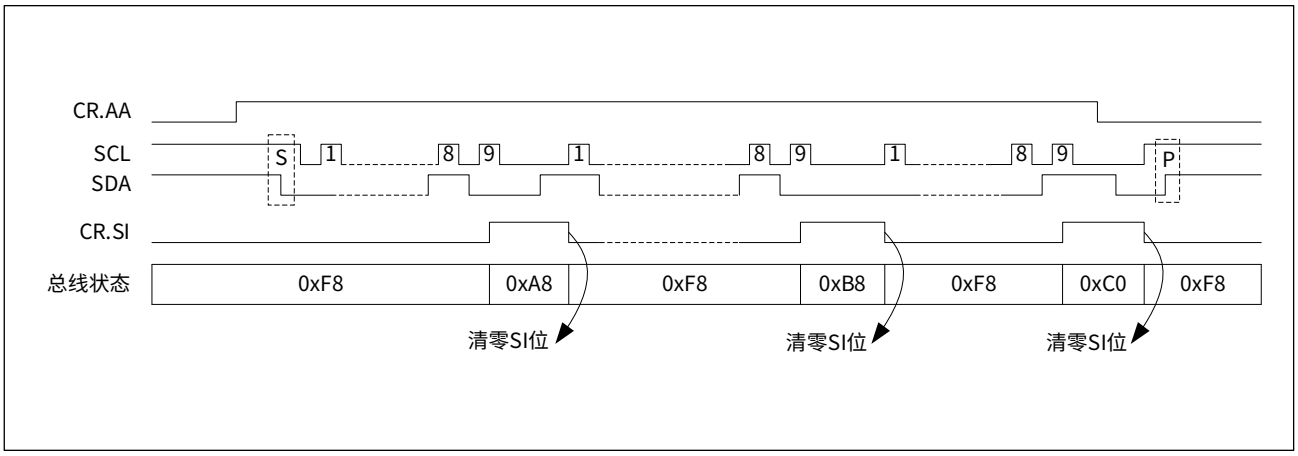
如果在传输过程中从机收到 NACK 应答，则从机不再发送数据，并进入未寻址从机接收模式。

如果在传输过程中从机主动将 I2Cx_CR.AA 设置为 0，则从机在发送完最后 1 字节有效数据后，将自身切换为未寻址从机接收模式，此后主机从总线上读数据将得到 0xFF。

当主机在 SLA+ 读写阶段由于总线冲突丢失仲裁时会进入未寻址从机接收模式，之后如果接收到符合本机地址的 SLA+R 并回应 ACK 后（状态码 I2Cx_STAT = 0xB0），则会进入已寻址从机发送模式。

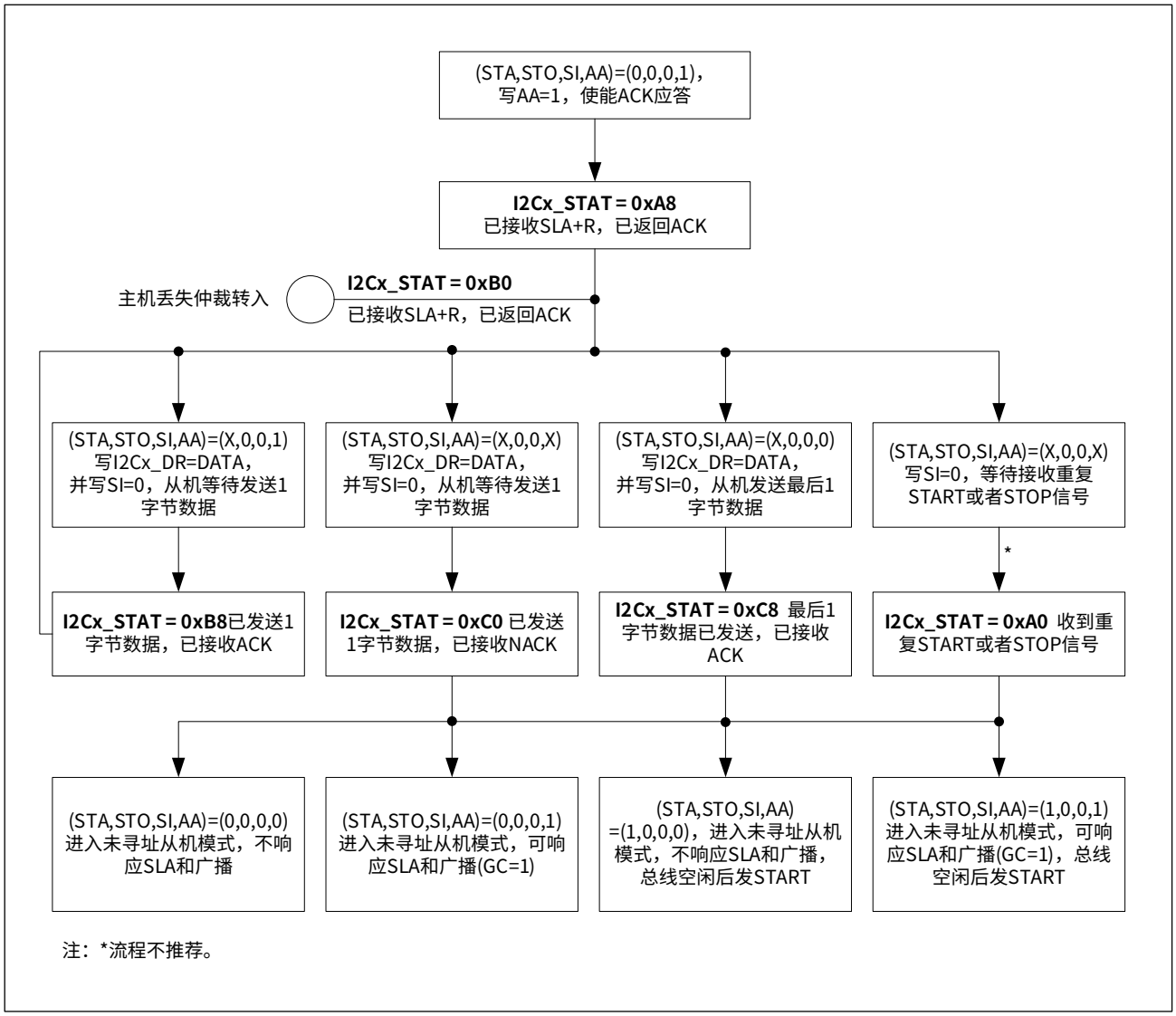
从机发送模式下状态同步图如下图所示：

图 19-13 从机发送模式状态同步图



从机发送模式下数据发送流程图及状态寄存器值如下图所示：

图 19-14　从机发送模式流程和寄存器状态



19.4.8.5 广播接收模式

广播接收模式是一种特殊的从机接收模式，当从机处于空闲状态（未寻址从机接收模式），收到主机发送的 SLA+W，且 SLA 为广播地址 0x00 时进入广播接收模式。该模式下数据接收过程和从机接收模式相同，但 I2C 总线状态码不同。

从机要接收主机发送的广播信息，需设置 I2Cx_CR.AA 为 1 以及 I2Cx_ADDR0.GC 为 1 以响应主机的广播寻址和广播数据； 3 个从机地址寄存器 I2Cx_ADDR0/1/2 不用设置； I2Cx_BRR 设置值无效，不用设置。

完成上述初始化工作后，从机进入空闲状态（未寻址从机接收模式），等待被主机发送的写信号（SLA+W）寻址。当从机接收到 SLA+W 后，如果地址匹配到广播地址 0x00，则从机回应 ACK 应答，并进入广播接收模式，状态码 I2Cx_STAT 变为 0x70，中断标志位 I2Cx_CR.SI 同时被置 1。此时必须及时清除 I2Cx_CR.SI 位，以便从总线上接收主机发送的数据。

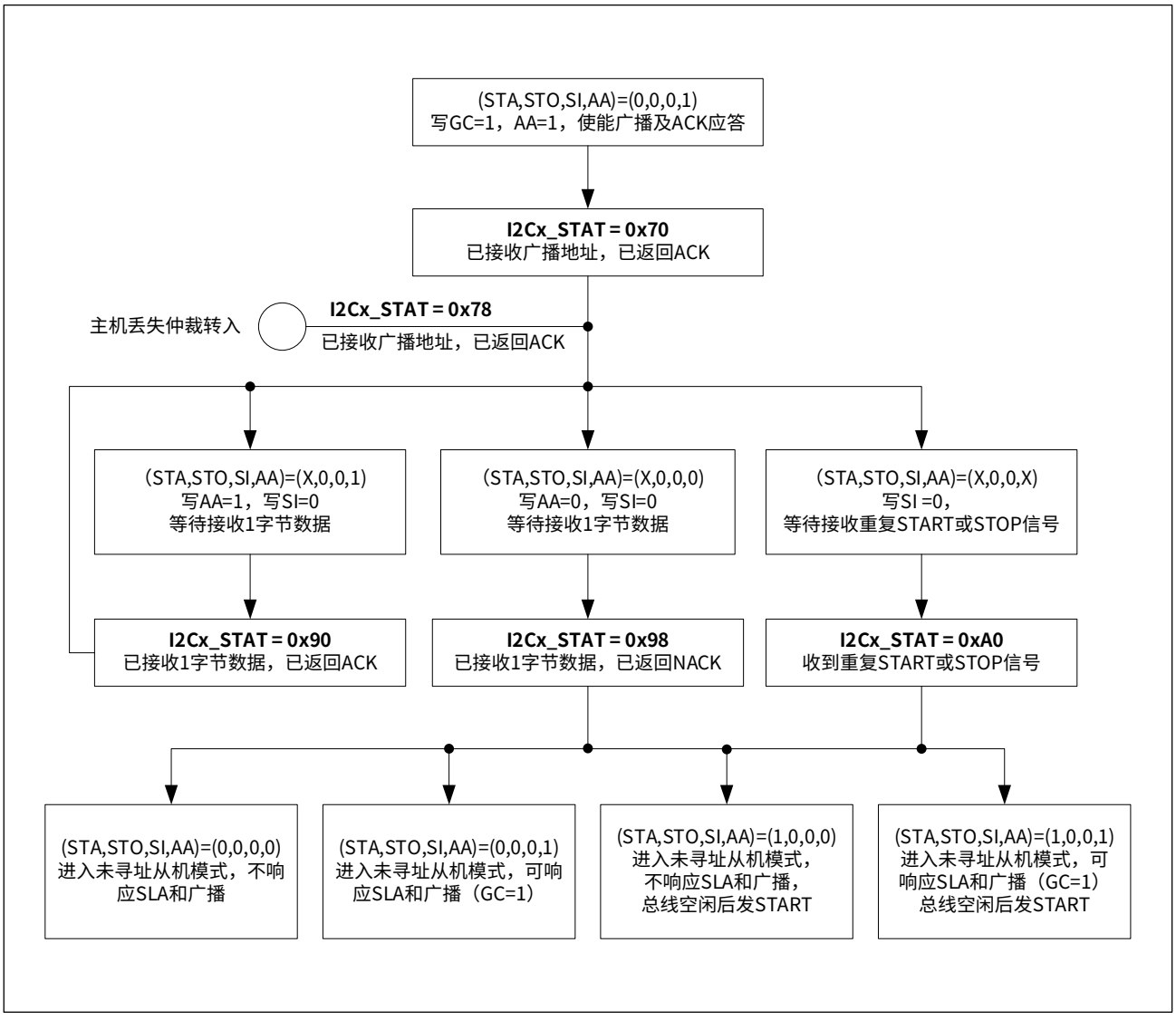
从机每接收到 1 字节数据都要回应 1 个 ACK 应答，应用程序读取完该字节数据后，必须将 I2Cx_CR.SI 位清零，为接收下 1 字节数据做好准备。

从机在接收数据过程中，如果 I2Cx_CR.AA 被清零，则从机将在接收到下一字节时返回 NACK 信号，从机自身状态也切换为未寻址从机接收模式，结束与主机的通信，不再接收数据，且 I2Cx_DR 寄存器保持之前接收到的数据。由该特性可知，从机应用程序可通过设置 I2Cx_CR.AA 为 0 主动将从机从已寻址广播接收模式切换未寻址从机接收模式。

当主机在 SLA+ 读写阶段由于总线冲突丢失仲裁时会进入未寻址从机接收模式，之后如果接收到符合广播地址的 SLA+W 并回应 ACK 后（状态码 I2Cx_STAT = 0x78），则会进入已寻址广播接收模式。

广播接收模式下数据接收流程图及状态寄存器值如下图所示：

图 19-15　广播接收模式流程和寄存器状态



19.4.9 多主机通信

在一些应用中，1 个 I2C 总线上有 2 个或多个主机同时访问从机，并有可能同时在传送数据，此时 SDA 总线上会存在数据冲突。

CW32F020 的 I2C 能进行 SDA 总线上的数据冲突检测和仲裁，实现多主机应用。如果两个主机同时发送数据，检测到冲突的主机会丢失仲裁并进入未寻址从机模式，未检测到冲突的主机会赢得仲裁并继续主导本次数据通信流程。

19.4.10 I2C 状态码

I2C 总线状态通过 I2C 状态寄存器 I2Cx_STAT 来标识，共 26 个正常接收或发送状态，和 2 个特殊状态（0xF8:I2C 总线无可用信息；0x00：总线错误）。

I2C 无论处于主机发送、主机接收、从机接收、从机发送或广播接收模式，当状态寄存器 I2Cx_STAT 的内容改变时，都会将 I2Cx_CR.SI 置位，且产生 I2C 中断。

I2C 状态码如下表所示：

表 19-3 I2C 状态码

工作模式	状态码	含义
主机发送模式	08H	已发送起始信号
	10H	已发送重复起始信号
	18H	已发送 SLA+W，已接收 ACK
	20H	已发送 SLA+W，已接收 NACK
	28H	已发送 I2Cx_DR 中的数据，已接收 ACK
	30H	已发送 I2Cx_DR 中的数据，已接收 NACK
	38H	主机在发送 SLA+W 阶段或者发送数据阶段丢失仲裁
主机接收模式	08H	已发送起始信号
	10H	已发送重复起始信号
	38H	主机在发送 SLA+R 阶段或者回应 NACK 阶段丢失仲裁
	40H	已发送 SLA+R，已接收 ACK
	48H	已发送 SLA+R，已接收 NACK
	50H	已接收数据字节，ACK 已返回
	58H	已接收数据字节，NACK 已返回
从机接收模式	60H	已接收自身的 SLA+W，已返回 ACK
	68H	当主机时在 SLA+ 读写阶段丢失仲裁，已接收自身的 SLA+W，已返回 ACK
	80H	前一次寻址使用自身从地址，已接收数据字节，已返回 ACK
	88H	前一次寻址使用自身从地址，已接收数据字节，已返回 NACK
	A0H	已寻址从机等待接收数据时，接收到停止条件或重复起始条件

工作模式	状态码	含义
从机发送模式	A8H	已接收自身的 SLA+R，已返回 ACK
	B0H	当主机时在 SLA+ 读写阶段丢失仲裁，已接收自身 SLA+R，已返回 ACK
	B8H	已发送数据字节，已接收 ACK
	C0H	已发送数据字节，已接收 NACK
	C8H	从机最后一个数据字节已被发送，并已接收 ACK
广播接收模式	70H	已接收广播地址（0x00），已返回 ACK
	78H	当主机时在 SLA+ 读写阶段丢失仲裁，已接收广播地址，已返回 ACK
	90H	前一次寻址使用广播地址，已接收数据字节，已返回 ACK
	98H	前一次寻址使用广播地址，已接收数据字节，已返回 NACK
	A0H	已寻址从机等待接收数据时，接收到停止条件或重复起始条件
其它	F8H	无可用的相关状态信息，I2Cx_CR.SI=0
	00H	传输过程出现总线错误，或外部干扰使 I2C 进入未定义的状态

特殊状态码 F8H，表示当前时刻没有任何有用信息，还不能确定当前总线的状态，因为 I2Cx_CR.SI 还没有被置位，无中断产生。这种情况在其它状态和 I2C 模块还未开始执行串行传输之前出现。

特殊状态码 00H，表示 I2C 串行传输过程中出现了总线错误，如 START 或者 STOP 信号出现在数据帧的错误位置上（包括在串行传输过程中的地址字节、数据字节或应答位）或者当外部干扰影响到内部 I2C 模块信号等。总线错误出现时，I2Cx_CR.SI 标志位会立即被置位，且设备立即被切换到未寻址从机接收模式，释放 SDA 和 SCL，并将 I2Cx_DR 寄存器清零。

当检测到 I2C 总线的 STAT 为总线错误（状态码为 00H）时，由于 I2C 总线为持续使能状态，并且对 I2C 模块来说错误并没有清除，SI 会持续保持为 1，即 SI 无法被清除。

总线错误清除方法：

1. 向总线发送 STOP 信号：置位 STO 位并清除 SI 位（由于当前处于总线错误状态，控制器并不会将 STOP 信号实际发送到总线上），STO 位会被硬件自动清 0，释放总线到正常空闲状态。
2. 如果置 STO 位仍然无法清除 SI 位，说明时序已被打乱，需要依次设置 I2Cx_CR.EN 为 0 和 1，即对 I2C 模块进行关闭并重启，然后设置 SI 为 0 清除 SI 位。

在各工作模式下，I2C 总线状态转换图如下图所示。注意，两种正常状态之间转换时，当未完成动作（如发送 SLA 过程中），即进入新的状态之前，状态码会出现短时的过渡状态，0xF8。用户可不关心，且不会产生中断。

图 19-16　主机发送 / 接收模式状态转换图

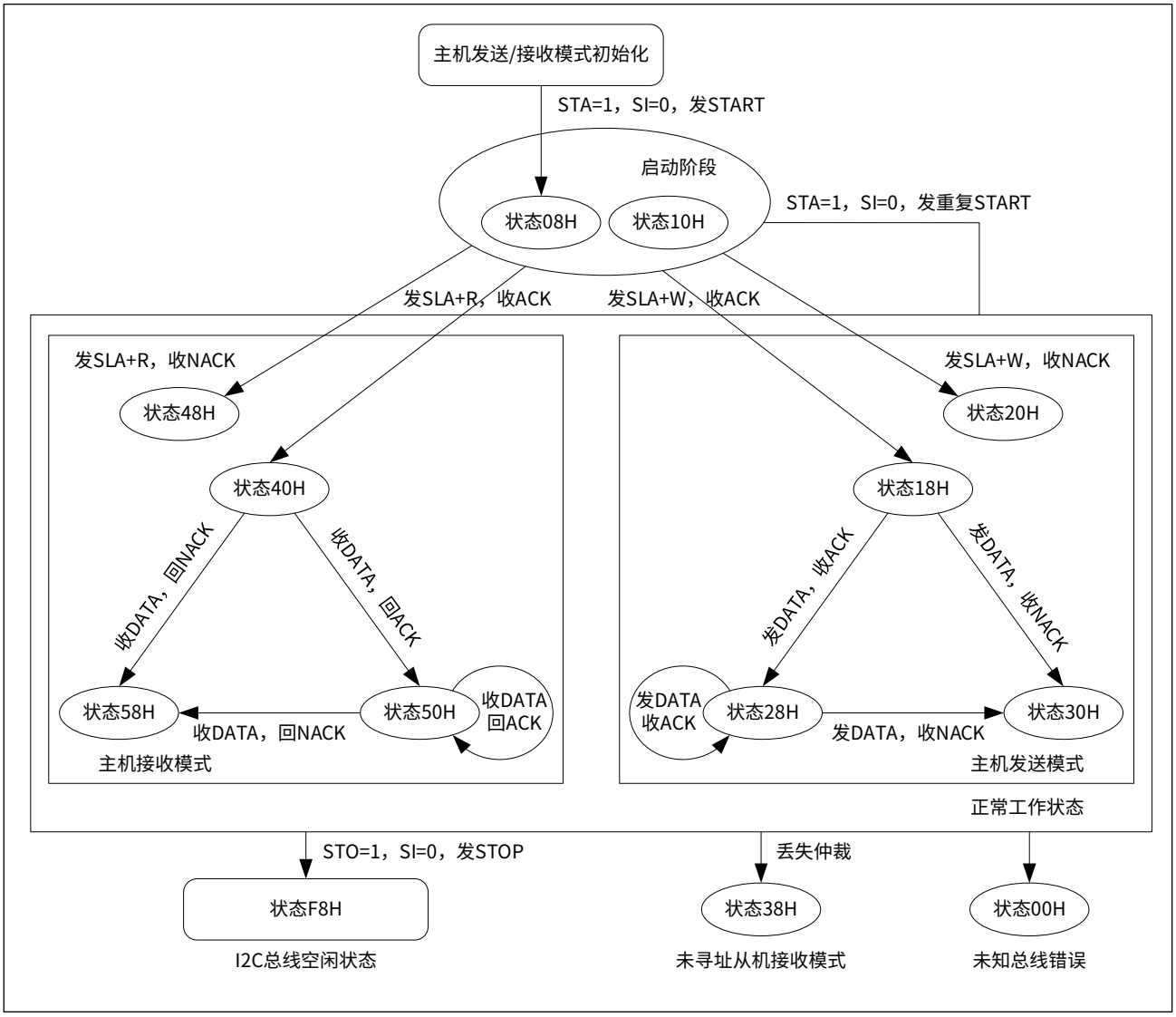
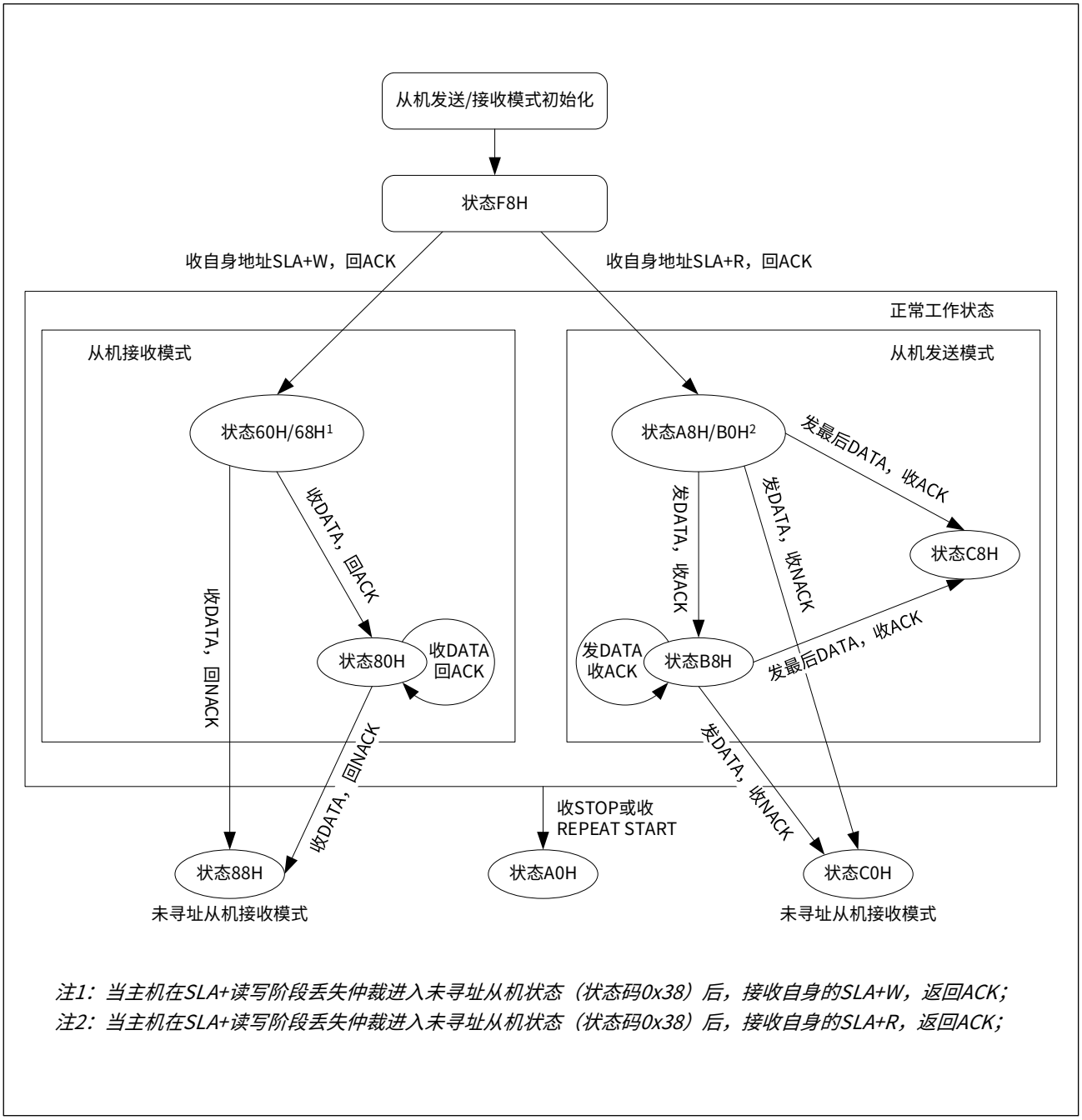
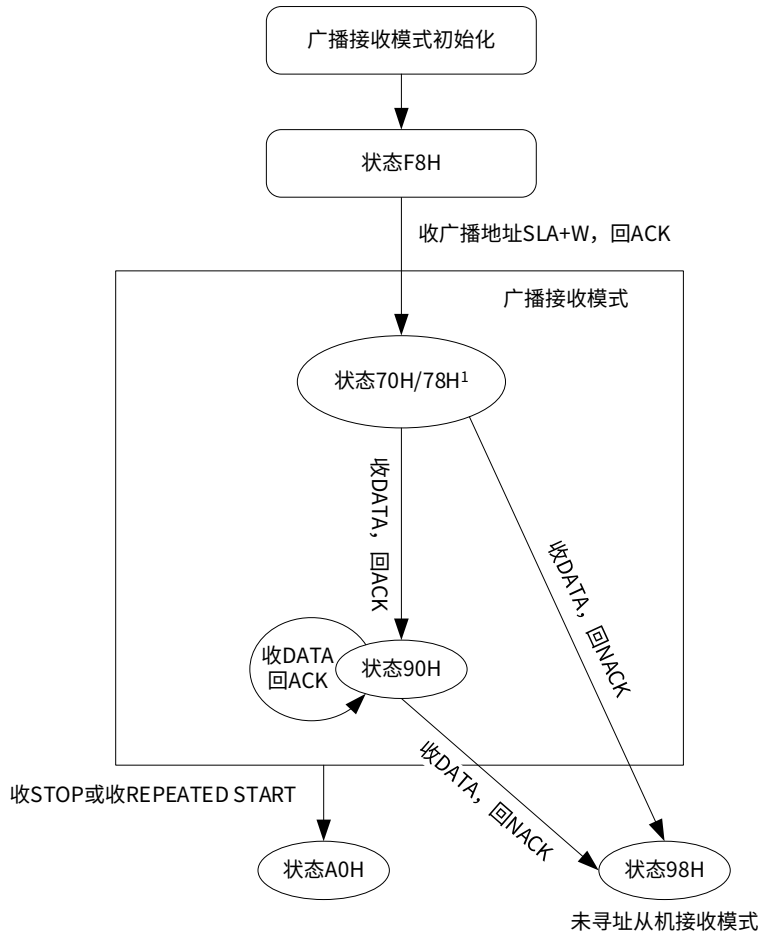


图 19-17　从机发送 / 接收模式状态转换图



注1：当主机在SLA+读写阶段丢失仲裁进入未寻址从机状态（状态码0x38）后，接收自身的SLA+W，返回ACK；
注2：当主机在SLA+读写阶段丢失仲裁进入未寻址从机状态（状态码0x38）后，接收自身的SLA+R，返回ACK；

图 19-18　广播接收模式状态转换图



注1：当主机在SLA+读写阶段丢失仲裁进入未寻址从机状态（状态码0x38）后，接收广播地址SLA+W，返回ACK；

19.5 编程示例

19.5.1 主机发送示例

步骤 1：按 GPIO 章节引脚数字复用功能的相关描述，将 SCL、SDA 映射到需要的引脚，并配置 SCL、SDA 引脚为开漏输出模式；

步骤 2：设置 SYSCTRL_APBEN1.I2Cx 为 1，使能 I2Cx 模块时钟；

步骤 3：向 SYSCTRL_APBRST1.I2Cx 依次写入 0、1，复位 I2Cx 模块；

步骤 4：配置 I2Cx_BRR，使 SCL 的时钟速率符合应用需求；

步骤 5：设置 I2Cx_BRREN 为 1，使能 SCL 时钟发生器；

步骤 6：设置 I2Cx_CR.EN 为 1，使能 I2C 模块；

步骤 7：设置 I2Cx_CR.STA 为 1，总线尝试发送 START 信号；

步骤 8：等待 I2Cx_CR.SI 变为 1，START 信号已发送到总线上；

步骤 9：查询 I2Cx_STAT，如果该寄存器值为 0x08 或 0x10，继续执行下一步骤，否则进行出错处理；

步骤 10：向 I2Cx_DR 中写入 SLA+W，设置 I2Cx_CR.STA 为 0，设置 I2Cx_CR.SI 为 0，发送 SLA+W；

步骤 11：等待 I2Cx_CR.SI 变为 1，SLA+W 已发送到总线上；

步骤 12：查询 I2Cx_STAT，如果该寄存器值为 0x18，继续执行下一步骤，否则进行出错处理；

步骤 13：向 I2Cx_DR 写入待发送的数据，设置 I2Cx_CR.SI 为 0，发送数据；

步骤 14：等待 I2Cx_CR.SI 变为 1，数据已发送到总线上；

步骤 15：查询 I2Cx_STAT，如果该寄存器值为 0x28，继续执行下一步骤，否则进行出错处理；

步骤 16：如待发送的数据未完成，则跳转到步骤 13 继续执行；

步骤 17：设置 I2Cx_CR.STO 为 1，设置 I2Cx_CR.SI 为 0，发送 STOP 停止信号，结束本次数据传输。

19.5.2 主机接收示例

步骤 1：按 GPIO 章节引脚数字复用功能的相关描述，将 SCL、SDA 映射到需要的引脚，并配置 SCL、SDA 引脚为开漏输出模式；

步骤 2：设置 SYSCTRL_APBEN1.I2Cx 为 1，使能 I2Cx 模块时钟；

步骤 3：向 SYSCTRL_APBRST1.I2Cx 依次写入 0、1，复位 I2Cx 模块；

步骤 4：配置 I2Cx_BRR，使 SCL 的时钟速率符合应用需求；

步骤 5：设置 I2Cx_BRREN 为 1，使能 SCL 时钟发生器；

步骤 6：设置 I2Cx_CR.EN 为 1，使能 I2C 模块；

步骤 7：设置 I2Cx_CR.STA 为 1，总线尝试发送 START 信号；

步骤 8：等待 I2Cx_CR.SI 变为 1，START 信号已发送到总线上；

步骤 9：查询 I2Cx_STAT，如果寄存器值为 0x08 或 0x10，继续执行下一步骤，否则进行出错处理；

步骤 10：向 I2Cx_DR 写入 SLA+R，设置 I2Cx_CR.STA 为 0，设置 I2Cx_CR.SI 为 0，发送 SLA+R；

步骤 11：等待 I2Cx_CR.SI 变为 1，SLA+R 已发送到总线上；

步骤 12：查询 I2Cx_STAT，如果寄存器值为 0x40（已收到 ACK），继续执行下一步骤，否则进行出错处理；

步骤 13：设置 I2Cx_CR.AA 为 1，使能应答标志；

步骤 14：设置 I2Cx_CR.SI 为 0，等待接收 1 字节数据（主机发送时钟，从机在时钟作用下发送数据）；

步骤 15：等待 I2Cx_CR.SI 变为 1（主机完成 1 字节数据接收并已回应 ACK 信号），从 I2Cx_DR 读取已接收到的数据；

步骤 16：查询 I2Cx_STAT，如果该寄存器值为 0x50 或 0x58，继续执行下一步骤，否则进行出错处理；

步骤 17：如果待接收的数据只差最后一个字节，设置 I2Cx_CR.AA 为 0，使能非应答标志；

步骤 18：如待接收的数据未完成，则跳转到步骤 14 继续执行；

步骤 19：设置 I2Cx_CR.STO 为 1，设置 I2Cx_CR.SI 为 0，发送 STOP 停止信号，结束本次数据传输。

19.5.3 从机接收示例

步骤 1：按 GPIO 章节引脚数字复用功能的相关描述，将 SCL、SDA 映射到需要的引脚，并配置 SCL、SDA 引脚为开漏输出模式；

步骤 2：设置 SYSCTRL_APBEN1.I2Cx 为 1，使能 I2Cx 模块时钟；

步骤 3：向 SYSCTRL_APBRST1.I2Cx 依次写入 0、1，复位 I2Cx 模块；

步骤 4：设置 I2Cx_CR.EN 为 1，使能 I2C 模块；

步骤 5：配置 I2Cx_ADDR0 为从机地址；

步骤 6：设置 I2Cx_CR.AA 为 1，使能应答标志；

步骤 7：等待 I2Cx_CR.SI 变为 1，被 SLA+W 寻址；

步骤 8：查询 I2Cx_STAT，如果该寄存器值为 0x60，继续执行下一步骤，否则进行出错处理；

步骤 9：设置 I2Cx_CR.SI 为 0，等待主机发送数据，并回应 ACK 信号；

步骤 10：等待 I2Cx_CR.SI 变为 1，从 I2Cx_DR 中读取已接收到的数据；

步骤 11：查询 I2Cx_STAT，如果该寄存器值为 0x80，继续执行下一步骤，否则进行出错处理；

步骤 12：如待接收的数据未完成，则跳转到步骤 9 继续执行；

步骤 13：设置 I2Cx_CR.AA 为 0，设置 I2Cx_CR.SI 为 0，从机切换到未寻址从机接收模式，且不响应主机寻址。

19.5.4 从机发送示例

步骤 1：按 GPIO 章节引脚数字复用功能的相关描述，将 SCL、SDA 映射到需要的引脚，并配置 SCL、SDA 引脚为开漏输出模式；

步骤 2：设置 SYSCTRL_APBEN1.I2Cx 为 1，使能 I2Cx 模块时钟；

步骤 3：向 SYSCTRL_APBRST1.I2Cx 依次写入 0、1，复位 I2Cx 模块；

步骤 4：设置 I2Cx_CR.EN 为 1，使能 I2C 模块；

步骤 5：配置 I2Cx_ADDR0 为从机地址；

步骤 6：设置 I2Cx_CR.AA 为 1，使能应答标志；

步骤 7：等待 I2Cx_CR.SI 变为 1，被 SLA+R 寻址；

步骤 8：查询 I2Cx_STAT，如果该寄存器的值为 0xA8，继续执行下一步骤，否则进行出错处理；

步骤 9：：向 I2Cx_DR 写入待发送的数据，设置 I2Cx_CR.SI 为 0，准备发送数据；

步骤 10：等待 I2Cx_CR.SI 变为 1，表示数据已发送到总线上，并收到 ACK 或者 NACK 应答；

步骤 11：查询 I2Cx_STAT，如果该寄存器的值为 0xB8 时，继续执行下一步骤，否则进行出错处理；

步骤 12：如待发送的数据未完成，则跳转到步骤 9 继续执行；

步骤 13：设置 I2Cx_CR.AA 为 0，设置 I2Cx_CR.SI 为 0，从机切换到未寻址从机接收模式，且不响应主机寻址。

19.6 寄存器列表

I2C1 基地址：I2C1_BASE = 0x4000 5400

I2C2 基地址：I2C2_BASE = 0x4000 5800

表 19-4 I2C 寄存器列表

寄存器名称	寄存器地址	寄存器描述
I2Cx_BRREN	I2Cx_BASE + 0x00	波特率计数器使能寄存器
I2Cx_BRR	I2Cx_BASE + 0x04	波特率计数器配置寄存器
I2Cx_CR	I2Cx_BASE + 0x08	控制寄存器
I2Cx_DR	I2Cx_BASE + 0x0C	数据寄存器
I2Cx_ADDR0	I2Cx_BASE + 0x10	从机地址 0 寄存器
I2Cx_STAT	I2Cx_BASE + 0x14	状态寄存器
I2Cx_ADDR1	I2Cx_BASE + 0x20	从机地址 1 寄存器
I2Cx_ADDR2	I2Cx_BASE + 0x24	从机地址 2 寄存器
I2Cx_MATCH	I2Cx_BASE + 0x28	从机地址匹配标志寄存器

19.7 寄存器描述

有关寄存器描述里所使用的缩写，请参见 1 文档约定章节。

19.7.1 I2Cx_BRREN 波特率计数器使能寄存器

Address offset: 0x00 Reset value: 0x0000 0000

位域	名称	权限	功能描述
31:1	RFU	-	保留位，请保持默认值
0	EN	RW	I2C 总线 SCL 波特率计数器使能控制 0：禁止 1：使能 注：主机时应使能 EN，从机时该位不影响

19.7.2 I2Cx_BRR 波特率计数器配置寄存器

Address offset: 0x04 Reset value: 0x0000 0000

位域	名称	权限	功能描述
31:8	RFU	-	保留位，请保持默认值
7:0	BRR	RW	I2C 总线 SCL 波特率配置 $f_{SCL} = f_{PCLK} / 8 / (BRR+1)$，其中 BRR > 0

19.7.3 I2Cx_CR 控制寄存器

Address offset: 0x08 Reset value: 0x0000 0000

位域	名称	权限	功能描述
31:7	RFU	-	保留位，请保持默认值
6	EN	RW	模块使能控制 0：禁止 1：使能
5	STA	RW	总线状态控制 W0：清零 STA W1：向总线发送 START 起始信号 注 1：设置 STA 为 1 后，如果总线空闲，则发送 START 起始信号，如果总线忙则等待 I2C 停止后，发送 START 信号。 注 2：如果设备已经在主机模式且已发送一个或多个字节，此时再设置 STA，I2C 总线将产生 Repeated START 重复起始信号。 注 3：STA 可在任何时间置 1，包括从机模式。但硬件不会在完成 START 或 repeat START 信号发送后自动清 0，需要用户手动清除 STA。
4	STO	RW	总线状态控制 W0：无功能 W1：向总线发送 STOP 停止信号 注 1：硬件会在完成 STOP 信号发送后自动对 STO 清 0。 注 2：如果在主机模式下 STA 和 STO 同时置 1，I2C 总线在发送 STOP 后马上发送 START。在从机模式下，禁止 STA 及 STO 同时置 1，以避免发出非法 I2C 帧。 注 3：当总线上产生错误状态 (STAT 状态字为 00H) STO 也会置 1，但这种情况下 I2C 总线不会发送 STOP 停止信号。
3	SI	RW	I2C 中断标志 R0：未发生 I2C 中断 R1：已发生 I2C 中断 W0：清除 I2C 中断标志并使状态机执行下一个动作 W1：无功能 注 1: I2C 所有 26 种状态中出现一种，硬件就会置 1 此位 (F8H 除外)，此时软件通过 I2Cx_STAT 寄存器值，来确认总线当前状态。 注 2：SI 需要软件清零。 注 3：在 SI 被清 0 之前，SCL 低电平周期延长，传输暂停，该状态对于从机处理接收到的数据非常有用，可以确保准确处理前一数据再接收下一个数据。 注 4：在软件清除 SI 前，软件应该准备好合适的寄存器设置。在 SI 被清除后，I2C 总线将会根据寄存器设置执行相应的操作。

位域	名称	权限	功能描述
2	AA	RW	应答控制 0：在应答阶段发送 NACK 1：在应答阶段发送 ACK 注 1：对于已被寻址的从机，在从机接收模式下未回复 ACK 应答位或在从机发送模式下未接收到 ACK 应答位，该从机将切换为未寻址从机接收模式，无法接收数据直到其 AA 被置 1，且重新被主机寻址。 注 2：特殊情况：从机发送模式时，从机发送最后一个字节给主机之前，清除 AA，发送完最后一个字节的位后，从机将切换为未被寻址的从机模式，和主机断开，状态寄存器 I2Cx_STAT 为 C8H。主机若再从总线上读数据，将得到 0xFF。
1	RFU	-	保留位，请保持默认值
0	FLT	RW	I2C 滤波参数配置 0：高级滤波，更高的抗干扰性能 1：简单滤波，更快的通信速率 注：详见 19.4.3 输入滤波器。

19.7.4 I2Cx_DR 数据寄存器

Address offset: 0x0C Reset value: 0x0000 0000

位域	名称	权限	功能描述
31:8	RFU	-	保留位，请保持默认值
7:0	DR	RW	数据寄存器 在发送模式下，写入待发送的数据 在接收模式下，读出接收到的数据

19.7.5 I2Cx_STAT 状态寄存器

Address offset: 0x14 Reset value: 0x0000 00F8

位域	名称	权限	功能描述
31:8	RFU	-	保留位，请保持默认值
7:0	STAT	RO	I2C 状态寄存器，状态值的具体定义详见 19.4.10 I2C 状态码； STAT = F8H 时，表示无可用的相关状态信息，SI 将保持为 0。 其它 26 种状态，都会让 SI 置 1，且产生中断请求。

19.7.6 I2Cx_ADDR0 从机地址 0 寄存器

Address offset: 0x10 Reset value: 0x0000 0000

位域	名称	权限	功能描述
31:8	RFU	-	保留位，请保持默认值
7:1	ADDR0	RW	从机模式地址 0 注 1：主机模式无效。 注 2：主机需要寻址该从机，需通过在 START 或 Repeated START 之后的第一个字节值地址信息与此地址相同。如果 AA 为 1，该从机响应主机，成为被寻址从机 , 否则主机广播寻址信息会被忽略。 注 3：I2Cx_ADDR0[7:1] 不能写为全 0，因为 0x00 为广播方式寻址专用。
0	GC	RW	广播地址应答使能 0：禁止 1：使能 注 1：主机模式无效。 注 2：使能 GC 后，如果 AA 置 1，则使能广播接收模式，若 AA 清 0，则忽略总线上的广播寻址信息。

19.7.7 I2Cx_ADDR1 从机地址 1 寄存器

Address offset: 0x20 Reset value: 0x0000 0000

位域	名称	权限	功能描述
31:8	RFU	-	保留位，请保持默认值
7:1	ADDR1	RW	从机模式地址 1 注：主机模式无效。
0	RFU	-	保留位，请保持默认值

19.7.8 I2Cx_ADDR2 从机地址 2 寄存器

Address offset: 0x24 Reset value: 0x0000 0000

位域	名称	权限	功能描述
31:8	RFU	-	保留位，请保持默认值
7:1	ADDR2	RW	从机模式地址 2 注：主机模式无效。
0	RFU	-	保留位，请保持默认值

19.7.9 I2Cx_MATCH 从机地址匹配寄存器

Address offset: 0x28 Reset value: 0x0000 0000

位域	名称	权限	功能描述
31:3	RFU	-	保留位，请保持默认值
2	ADDR2	RO	I2C 从机模式地址 2 匹配标志位 0：从总线收到的设备地址与 ADDR2 不相同 1：从总线收到的设备地址与 ADDR2 相同
1	ADDR1	RO	I2C 从机模式地址 1 匹配标志位 0：从总线收到的设备地址与 ADDR1 不相同 1：从总线收到的设备地址与 ADDR1 相同
0	ADDR0	RO	I2C 从机模式地址 0 匹配标志位 0：从总线收到的设备地址与 ADDR0 不相同 1：从总线收到的设备地址与 ADDR0 相同

注：

地址匹配标志位在以下情况会清零：

- 模块复位时；
- START/STOP 发送时。

20 红外调制发送器（IR）

20.1 概述

CW32F020 内部集成红外调制发送器 (IR)，通过两路通用定时器或一路通用定时器与 UART 配合使用，可方便实现各种标准的 PWM 或 PPM 编码方式，也可实现 UART 数据的红外调制发送。

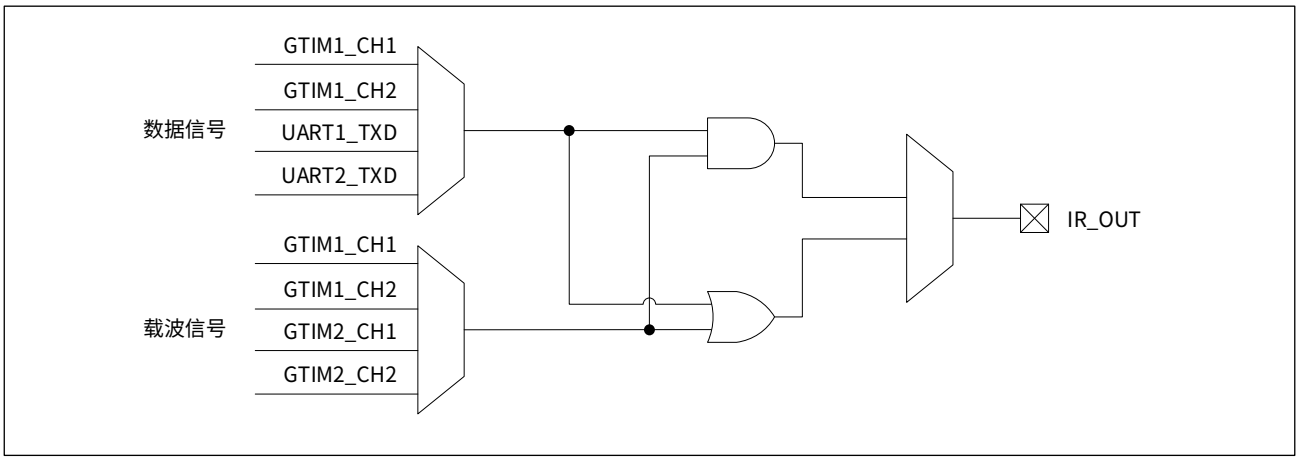
20.2 主要特性

- 支持 IrDA 标准 1.0 的 SIR
- 最高数据速率 115.2kbps
- 可适应高低电平红外发射管

20.3 功能描述

实现红外调制发送器时，使用一个通用定时器通道产生一个固定频率的方波信号，另一个通用定时器或 UART 用以产生调制数据，二者进行‘与’或‘或’运算后，从 IR_OUT 引脚输出，用户可选择 PA13 或 PB09 作为 IR_OUT 输出。IR 内部连接示意图如下图所示：

图 20-1 IR 内部连接示意图



IR 红外调制控制寄存器 SYSCTRL_IRMOD，用于选择载波信号和数据信号的来源，以及二者的‘与’ ‘或’操作。选择‘与’ ‘或’由用户的硬件红外发射管的驱动电平决定。

载波信号频率用户可自行设置，最常见的载波频率是 38kHz。

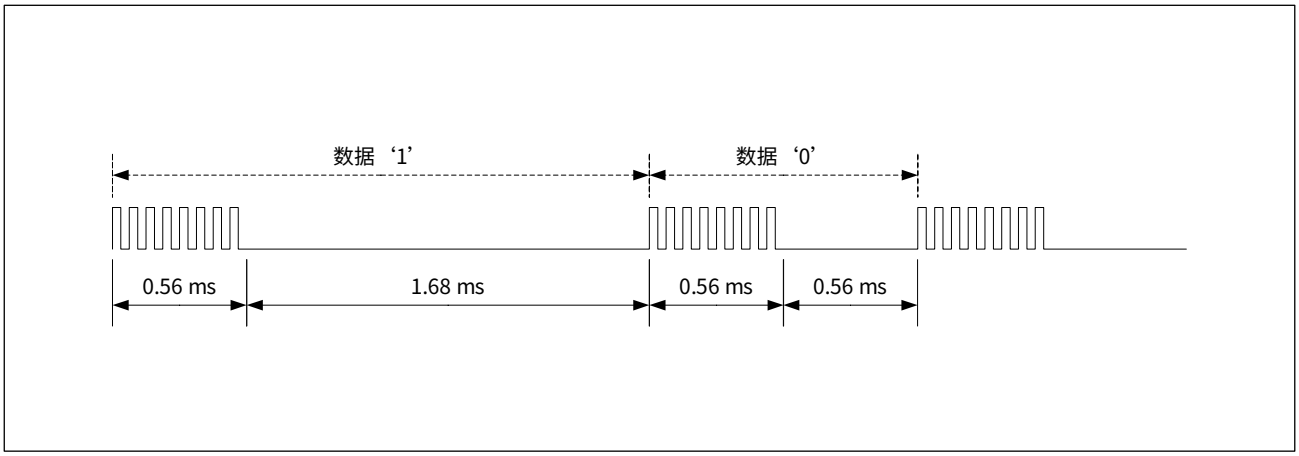
为降低发射功耗，载波的占空比一般设置为 20% 左右，产生载波的定时器可以使用 PWM 模式，请参见 14 通用定时器 (GTIM) 章节。为提高传输距离，可适当提高占空比。在 IrDA 标准 1.0 中，为实现快速的 IR 通信速率，脉冲的宽度规定为一个位周期的 3/16 或者为固定的 1.63μs，最小不能低于 1.41μs。

20.3.1 红外调制方式

常见 IR 红外编码方式有两种，脉宽调制 PWM 和脉位调制 PPM。

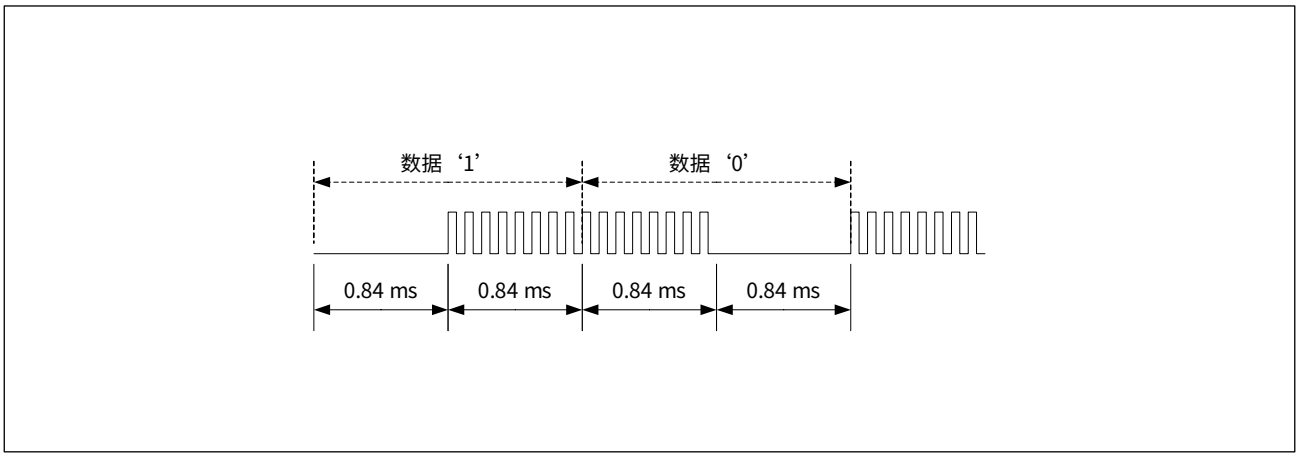
PWM 脉冲宽度调制，以发射红外载波的占空比代表‘0’和‘1’。比如 UPD6121，载波发射 0.56ms，不发射 0.56ms，表示‘0’；载波发射 0.56ms，不发射 1.68ms，表示‘1’。参考示意图如下图所示：

图 20-2 PWM 调制方式



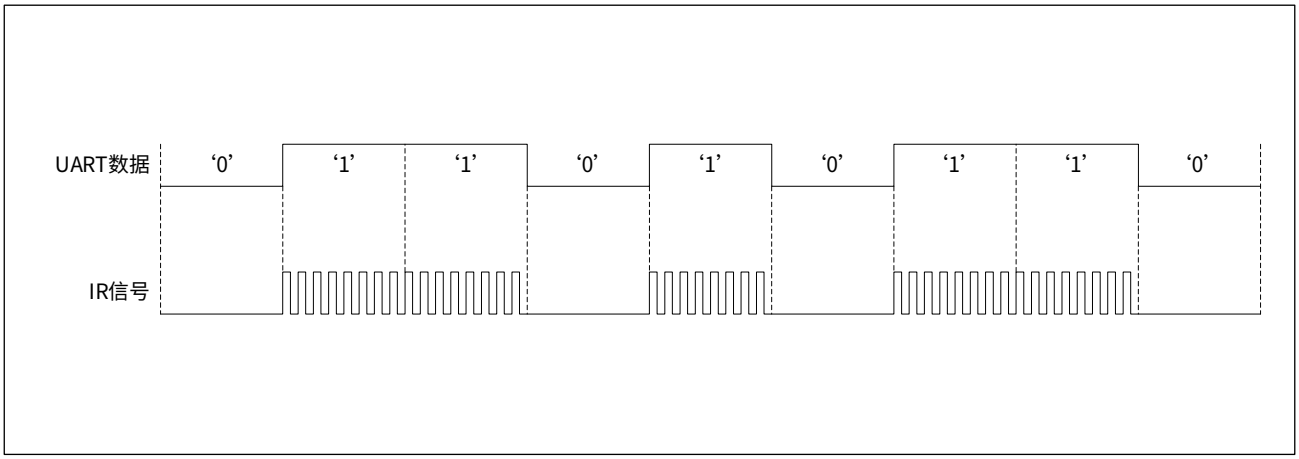
PPM 脉冲位置调制，以发射载波的位置表示‘0’和‘1’。从发射载波到不发射载波为‘0’，从不发射载波到发射载波为‘1’。比如 SAA3010，载波发射 0.84ms，不发射 0.84ms，表示‘0’；不发射 0.84ms，载波发射 0.84ms，表示‘1’。参考示意图如下图所示：

图 20-3 PPM 调制方式



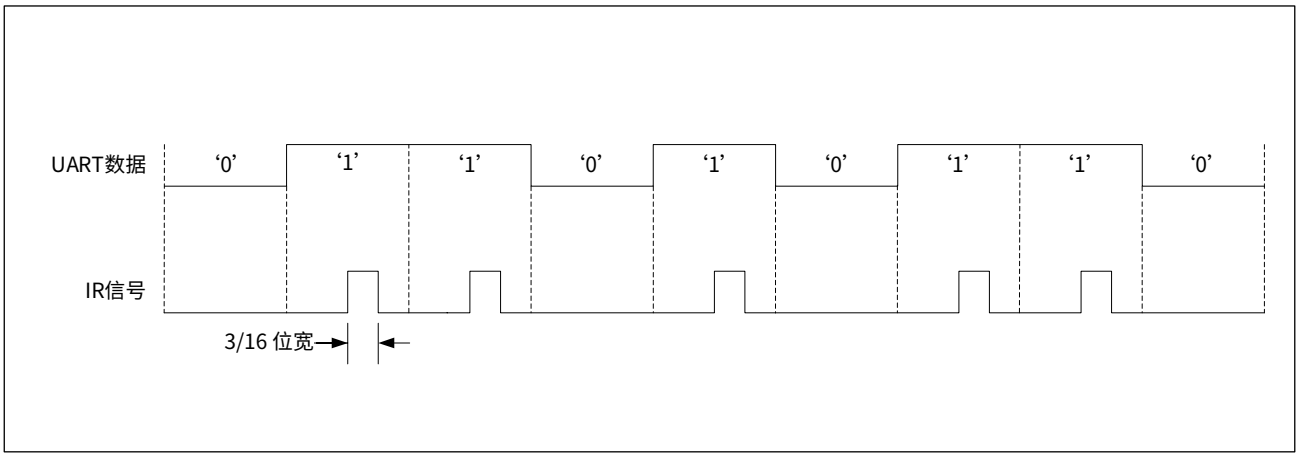
UART 调制是一种比较简单的红外调制方式，直接将串行数据与载波信号进行‘与’或‘或’运算。当 UART 串行波特率低于载波频率时，参考示意图如下图所示：

图 20-4 UART 调制方式



当 UART 串行波特率高于载波频率时，载波脉冲宽度应设置为位宽度的 3/16，载波频率需保持与串行频率一致。比如当 UART 波特率设置为 115.2Kbps 时，载波频率也设置为 115.2kHz，载波脉冲的宽度为 1.63μs（3/16×1/115200Hz = 1.63μs），参考示意图如下图所示：

图 20-5 高速 UART 调制方式



为方便调试，通常一个标准的数据帧还包括引导码、结束位、重发码等，请用户自行参阅相关数据手册。

20.3.2 红外调制初始化配置

一个完整的 IR 红外调制发送器的初始化过程要包括对定时器、UART、GPIO 的配置，参考步骤如下：

步骤 1：设置 SYSCTRL_APBEN1、SYSCTRL_APBEN2 寄存器，使能将要选择的定时器、UART 的外设时钟；

步骤 2：设置 SYSCTRL_AHBEN 中要使用的 GPIOA 或 GPIOB 时钟使能允许位；

步骤 3：配置选择作为载波的定时器，设置载波频率、占空比（常用频率 38kHz，1/3 占空比），设置定时器工作模式（推荐为 PWM 输出）；

步骤 4：配置选择作为数据的定时器或 UART，设定位宽度，UART 工作模式（不需定义输出引脚）；

步骤 5：向 GPIOA_LCKR 或 GPIOB_LCKR 寄存器写入‘0x5A5Axxxx’，解锁选择作为 IR_OUT 输出端口的 PA13 或 PB09；

步骤 6：设定作为 IR_OUT 输出端口的 GPIOx_ANALOG 为数字模式，设置 GPIOx_DIR 为输出，同时设置适合的 GPIOx_OPENDRAIN、GPIOx_DRIVER、GPIOx_SPEED、GPIOx_PUR 和 GPIOx_PDR；

步骤 7：启动作为载波和数据的定时器、UART；

步骤 8：设置 IR 红外调制控制寄存器 SYSCTRL_IRMOD 的 MOD 位域，设定红外调制方式配置；

步骤 9：设定 IR_OUT 输出端口的 GPIOx_AFRH 为 IR_OUT（PA13 为 AF6，PB09 为 AF4），启动 IR 复用功能。

20.3.3 红外接收

CW32F020 内部没有 IR 接收解调模块，在 IR 接收应用中，需要使用带有解调功能的一体化红外接收头，配合 GTIM 的捕捉功能（UART 方式可直接使用 RXD 引脚输入），可方便的实现 IR 接收功能。

20.4 SYSCTRL_IRMOD 红外调制控制寄存器

Address: 0x4001 0000 + 0x74 Reset value: 0x0000 0000

位域	名称	权限	功能描述
31:4	RFU	-	保留位，请保持默认值
3:0	MOD	RW	红外调制方式配置 0000：GTIM1_CH1 & GTIM2_CH1 0001：GTIM1_CH1 & GTIM2_CH2 0010：GTIM1_CH2 & GTIM2_CH1 0011：GTIM1_CH2 & GTIM2_CH2 0100：GTIM1_CH1 | GTIM2_CH1 0101：GTIM1_CH1 | GTIM2_CH2 0110：GTIM1_CH2 | GTIM2_CH1 0111：GTIM1_CH2 | GTIM2_CH2 1000：UART1_TXD & GTIM1_CH1 1001：UART1_TXD | GTIM1_CH1 1010：UART1_TXD & GTIM1_CH2 1011：UART1_TXD | GTIM1_CH2 1100：UART2_TXD & GTIM2_CH1 1101：UART2_TXD | GTIM2_CH1 1110：UART2_TXD & GTIM2_CH2 1111：UART2_TXD | GTIM2_CH2

21 模数转换器（ADC）

21.1 概述

CW32F020 内部集成一个 12 位精度、最高 1M SPS 转换速度的逐次逼近型模数转换器 (SAR ADC)，最多可将 16 路模拟信号转换为数字信号。现实世界中的绝大多数信号都是模拟量，如光、电、声、图像信号等，都要由 ADC 转换成数字信号，才能由 MCU 进行数字化处理。

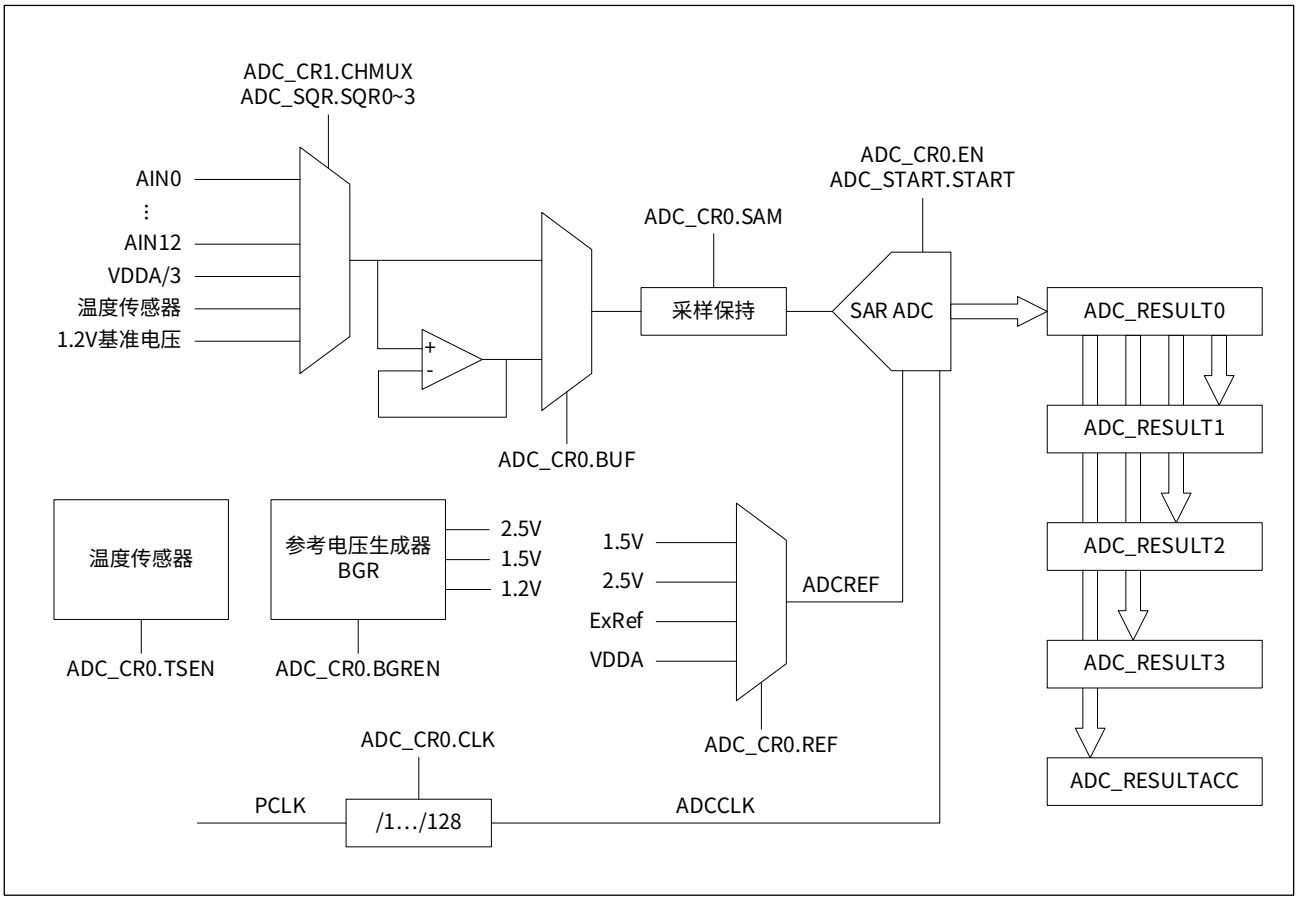
21.2 主要特性

- 12 位精度
- 可编程转换速度，最高达 1M SPS
- 16 路输入转换通道
  - 13 路外部引脚输入
  - 内置温度传感器
  - 内置 BGR 1.2V 基准
  - 1/3 VDDA 电源电压
- 4 路参考电压源（Vref）
  - VDDA 电源电压
  - ExRef（PB00）引脚电压
  - 内置 1.5V 参考电压
  - 内置 2.5V 参考电压
- 采样电压输入范围：0 ~ V_{ref}
- 多种转换模式，全部支持转换累加功能
  - 单次转换
  - 多次转换
  - 连续转换
  - 序列扫描转换
  - 序列断续转换
- 支持单通道、序列通道两种通道选择，最大同时支持 4 个序列
- 支持输入通道电压阈值监测
- 内置信号跟随器，可转换高阻抗输入信号
- 支持片内外设自动触发 ADC 转换

21.3 功能框图

ADC 功能框图如下图所示：

图 21-1 ADC 功能框图

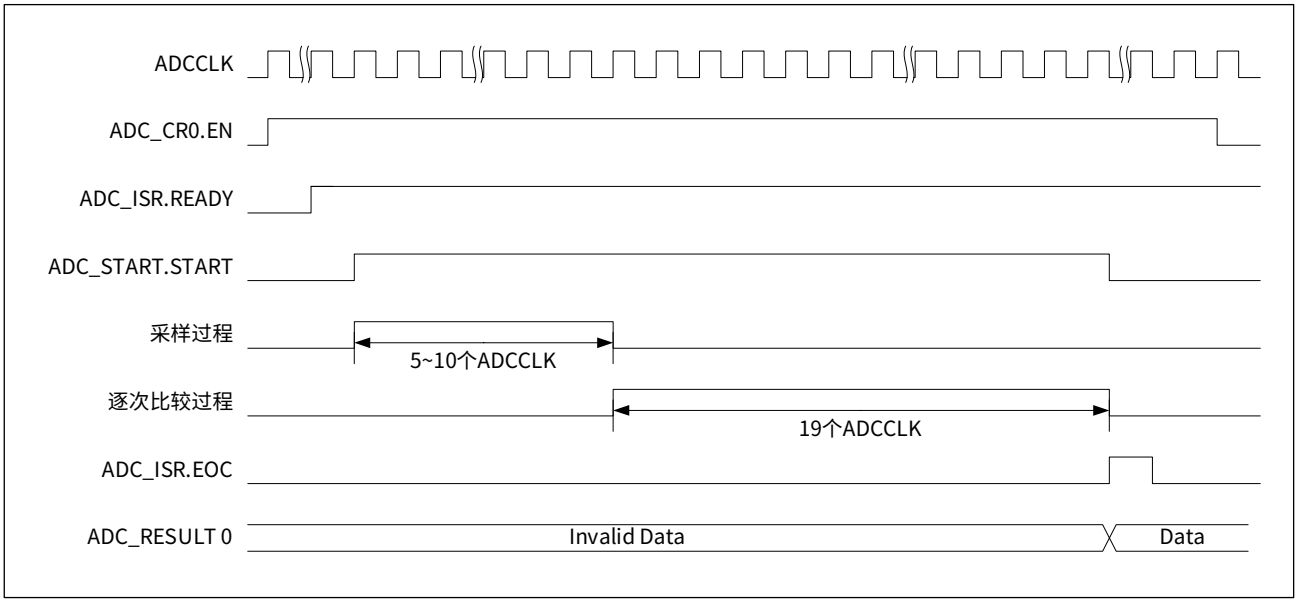


21.4 转换时序、转换速度、转换精度以及转换结果

21.4.1 转换时序

ADC 的转换时序如下图所示。

图 21-2 ADC 转换时序图



向 ADC 控制寄存器 ADC_CR0 的 EN 位域写入 1，使能 ADC 模块。

ADC_CR0.EN 由 0 变为 1 约 40μs 后 ADC_ISR.READY 标志位置 1，表示模拟电路初始化完成，可以开始进行 ADC 转换。

向 ADC 启动寄存器 ADC_START 的 START 位域写入 1，启动 ADC 转换，转换完成后硬件自动清零。

ADC 工作时钟 ADCCLK，由系统时钟 PCLK 经预分频器分频得到，通过控制寄存器 ADC_CR0 的 CLK 位域可选择 1 ~ 128 分频，如下表所示：

表 21-1 ADC 时钟配置表

ADC_CR0.CLK	ADCCLK
000	PCLK
001	PCLK/2
010	PCLK/4
011	PCLK/8
100	PCLK/16
101	PCLK/32
110	PCLK/64
111	PCLK/128

一次完整的 ADC 转换需要 24 ~ 29 个 ADCCLK 时钟周期，包括采样阶段和逐次比较两个阶段：

1. 采样阶段：需要 5 ~ 10 个 ADCCLK 时钟周期。采样周期通过控制寄存器 ADC_CR0 的 SAM 位域配置，如下表所示：

表 21-2 ADC 采样周期选择表

ADC_CR0.SAM	ADC 采样周期（ADCCLK 个数）
00	5
01	6
10	8
11	10

ADC 采样周期长度由用户对采样的速度要求和采样信号的电气特性决定，用户应选择合适的采样周期，以达到最佳的转换效果。

2. 逐次比较阶段：需要 19 个 ADCCLK 时钟周期。

ADC 转换完成之后，转换完成标志位 ADC_ISR.EOC 会被硬件置 1，ADC 转换结果存储在对应的 ADC 转换结果寄存器 ADC_RESULTy（y=0,1,2,3）中，用户可通过设置 ADC_ICR.EOC 为 0 清除该标志位。

21.4.2 转换速度

ADC 转换速度与 ADC 参考电压和 VDDA 电源电压密切相关，各种条件下的最高转换速度，如下表所示：

表 21-3 ADC 转换速度与电压对照表

ADC 参考电压	VDDA 电压	最高转换速度	最大 ADCCLK 频率
内部 1.5V	1.8V ~ 2V	100K SPS	2MHz
内部 1.5V	2V ~ 5.5V	200K SPS	4MHz
内部 2.5V	2.8V ~ 5.5V	200K SPS	4MHz
VDDA/ExRef	1.65V ~ 1.8V	25K SPS	500kHz
VDDA/ExRef	1.8V ~ 2V	100K SPS	2MHz
VDDA/ExRef	2V ~ 2.4V	200K SPS	4MHz
VDDA/ExRef	2.4V ~ 2.7V	500K SPS	12MHz
VDDA/ExRef	2.7V ~ 5.5V	1M SPS	24MHz

ADC 的转换速度的与工作时钟 ADCCLK 的对应关系如下：

$$\text{ADC 转换速率} = f_{ADCCLK} / N_T$$

其中，f_{ADCCLK} 为 ADCCLK 时钟频率，N_T 为一次 ADC 转换所需要的 ADCCLK 个数。

21.4.3 转换精度

当 ADC 外部输入信号驱动能力不足，或 ADC 输入来自芯片内部时（内置温度传感器电压、内置 1.2V 基准电压或 1/3 VDDA 电压），必须使能 ADC 模块内置的信号跟随器，并使用单通道单次转换模式。内置信号跟随器由控制寄存器 ADC_CR0 的 BUF 位域控制，设置 BUF 为 1，使能跟随器；设置 BUF 为 0，禁止跟随器。

当选择多通道 ADC 转换，其中部分通道的信号驱动能力较弱时，为了避免 ADC 驱动能力弱的输入通道受到干扰，必须使能信号跟随器，同时需使 ADC 转换速度不高于 200K SPS。

如需进一步提高 ADC 转换精度，用户可使用 ADC 转换累加功能，对同一个通道进行多次采样，将累加结果的算术平均值作为最终测量值。详细请参见 21.6 累加转换功能。

21.4.4 转换结果

ADC 转换完成后，12 位 ADC 转换结果存储在对应的转换结果寄存器 ADC_RESULTy 中。

当 ADC 工作于单通道转换模式时，转换结果存储在 ADC_RESULT0 寄存器。

当 ADC 工作于序列转换模式时，转换序列 SQRy（y=0,1,2,3）的转换结果保存在对应的 ADC_RESULTy（y=0,1,2,3）寄存器。

转换结果寄存器 ADC_RESULTy 是 16 位宽，用户可选择左对齐或右对齐，由控制寄存器 ADC_CR1 的 ALIGN 位域决定：

- ALIGN 位为 0，选择右对齐，有效值存储于 ADC_RESULTy 寄存器的低 12 位（位 11:0），高位（位 15:4）自动补 0；
- ALIGN 位为 1，选择左对齐，有效值存储于 ADC_RESULTy 寄存器的高 12 位（位 15:4），低位（位 3:0）自动补 0。

21.5 工作模式

ADC 控制寄存器 ADC_CR0 的 MODE 位域配置 ADC 工作模式，详见下表：

表 21-4 ADC 工作模式配置

ADC_CR0.MODE	ADC 工作模式
000	单通道单次转换模式
001	单通道多次转换模式，转换次数详见 CR2.CNT
010	单通道连续转换模式
011	序列连续转换模式
100	序列扫描转换模式
101	序列多次转换模式，转换次数详见 CR2.CNT
110	序列断续转换模式

启动 ADC 转换，可通过向 ADC 启动寄存器 ADC_START 的 START 位域写 1；也可通过其他外设来触发，参见 21.8 外部触发源。

21.5.1 单通道单次转换模式（MODE=0）

ADC 启动后，对指定的某一个通道，执行一次转换。

ADC 有 16 个通道可以选择，由控制寄存器 ADC_CR1 的 CHMUX 位域决定，如下表所示。其中，AIN0 ~ AIN12 为外部引脚输入，使用时需先使能对应 GPIO 的模拟功能（GPIOx_ANALOG.PINy = 1）。

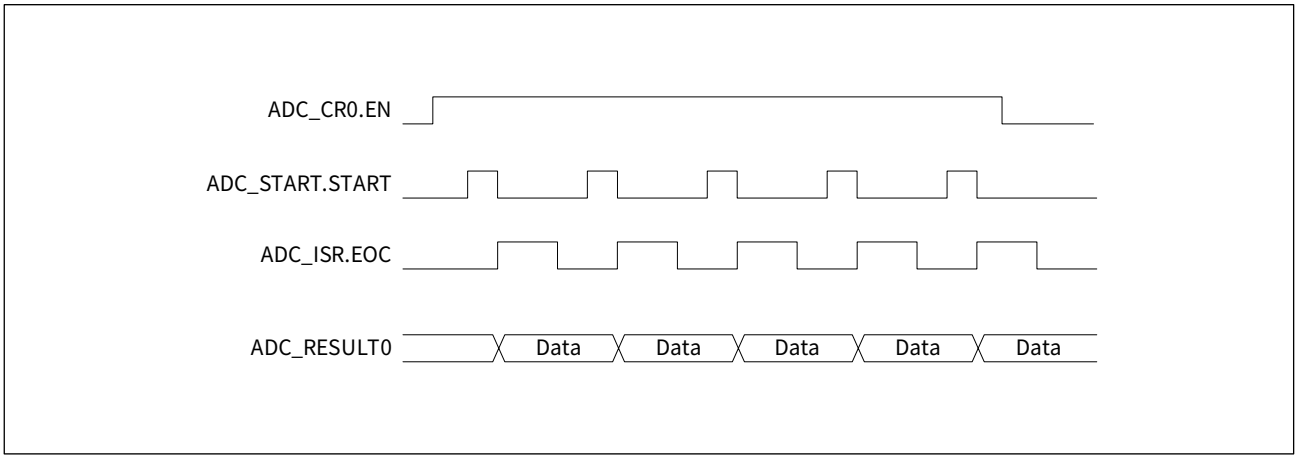
表 21-5 单通道配置

ADC_CR1.CHMUX	通道选择	GPIO
0000	AIN0	PA00
0001	AIN1	PA01
0010	AIN2	PA02
0011	AIN3	PA03
0100	AIN4	PA04
0101	AIN5	PA05
0110	AIN6	PA06
0111	AIN7	PA07
1000	AIN8	PB00
1001	AIN9	PB01
1010	AIN10	PB02
1011	AIN11	PB10
1100	AIN12	PB11
1101	VDDA/3	-
1110	TS 内置温度传感器	-
1111	1.2V 内核电压基准源	-

在单通道单次转换模式下，ADC 转换完成后，转换完成标志位 ADC_ISR.EOC 会被硬件自动置 1，转换结果保存在 ADC_RESULT0 寄存器中，同时 ADC 启动寄存器 ADC_START 的 START 位自动清 0，ADC 转换停止。

单通道单次转换模式的时序如下图所示：

图 21-3 单通道单次转换时序图



通过 START 位启动 ADC 单通道对外部模拟输入信号的单次转换，参考操作流程如下：

步骤 1：设置 SYSCTRL_AHBEN.GPIOx 为 1，SYSCTRL_APBEN2.ADC 为 1，使能 ADC 通道对应的 GPIO 时钟和 ADC 工作时钟；

步骤 2：设置 ADC 通道对应的 GPIO 引脚为模拟功能，具体寄存器配置请参见 9 通用输入输出端口（GPIO）章节；

步骤 3：设置 ADC_CR0.EN 为 1，使能 ADC 模块；

步骤 4：查询等待 ADC_ISR.READY 位变为 1，等待 ADC 模块启动完成；

步骤 5：设置 ADC_CR0.MODE 为 0，选择单通道单次转换模式；

步骤 6：配置 ADC_CR0.REF，选择 ADC 的参考电压源；

注：

如选择外部参考电压引脚，需先将此引脚配置为模拟功能。

步骤 7：配置 ADC_CR0.SAM 及 ADC_CR0.CLK，设置 ADC 的采样速度及时钟选择；

步骤 8：配置 ADC_CR1.CHMUX，选择待转换的通道；

步骤 9：设置 ADC_START.START 为 1，启动 ADC 转换；

步骤 10：等待 ADC_ISR.EOC 变为 1，读取 ADC_RESULT0 寄存器中的 ADC 转换结果；

步骤 11：如需对其它通道进行转换，重复执行步骤 8 至步骤 10；

步骤 12：设置 ADC_CR0.EN 为 0，关闭 ADC 模块。

通过外部触发启动 ADC 单通道对外部模拟输入信号的单次转换，使用 ADC 转换完成中断读取转换结果，参考操作流程如下：

步骤 1：设置 SYSCTRL_AHBEN.GPIOx 为 1，SYSCTRL_APBEN2.ADC 为 1，使能 ADC 通道对应的 GPIO 时钟和 ADC 工作时钟；

步骤 2：设置 ADC 通道对应的 GPIO 引脚为模拟功能，具体寄存器配置请参见 9 通用输入输出端口（GPIO）章节；

步骤 3：设置 ADC_CR0.EN 为 1，使能 ADC 模块；

步骤 4：查询等待 ADC_ISR.READY 位变为 1，等待 ADC 模块启动完成；

步骤 5：设置 ADC_CR0.MODE 为 0，选择单通道单次转换模式；

步骤 6：配置 ADC_CR0.REF，选择 ADC 的参考电压；

注：

如选择外部参考电压引脚，需先将此引脚配置为模拟功能。

步骤 7：配置 ADC_CR0.SAM 及 ADC_CR0.CLK，设置 ADC 的采样速度及时钟选择；

步骤 8：设置 ADC_IER.EOC 为 1，使能 ADC 转换完成中断；

步骤 9：使能 NVIC 中断向量表中的 ADC 中断；

步骤 10：设置 ADC_ICR 为 0x00，清除 ADC 中断标志；

步骤 11：配置外部触发寄存器 ADC_TRIGGER，选择外部触发源；

步骤 12：配置 ADC_CR1.CHMUX，选择待转换的通道；

步骤 13：等待 ADC 被触发启动。当 ADC 转换完成，ADC_ISR.EOC 标志位被硬件置 1，MCU 响应 ADC 中断，进入 ADC 中断服务程序，用户读取 ADC_RESULT0 寄存器中的 ADC 转换结果；退出服务程序时，应首先清除 ADC_ISR.EOC 标志位；

步骤 14：如需对其它通道进行转换，重复执行步骤 11 至步骤 13；

步骤 15：设置 ADC_CR0.EN 为 0，关闭 ADC 模块。

21.5.2 单通道多次转换模式（MODE=1）

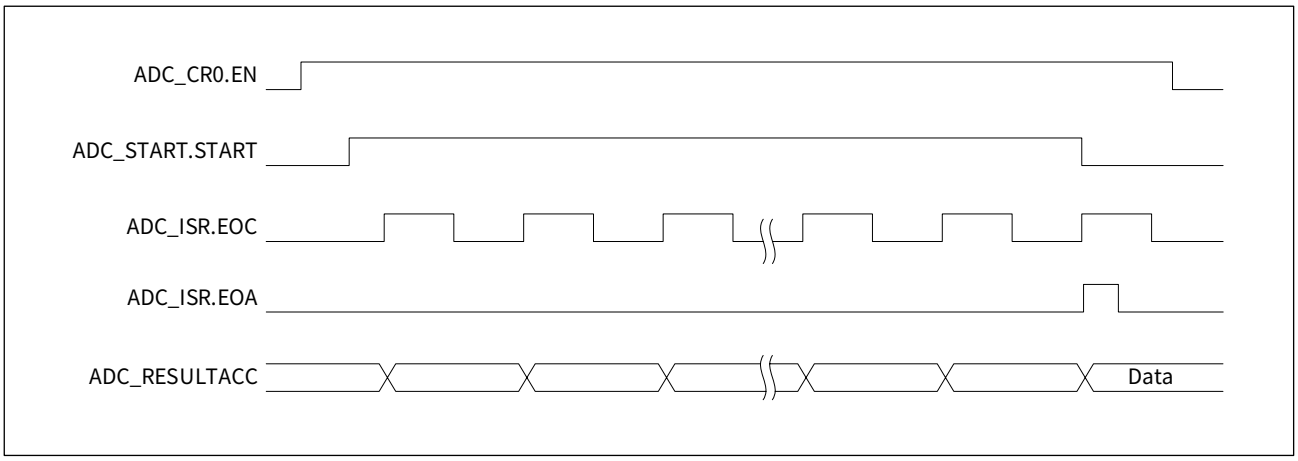
ADC 启动后对指定的某一个通道，执行多次转换，转换次数由 ADC_CR2.CNT 位域值决定，默认转换次数是 1。

多次转换模式下必须使能累加转换功能，即设置 ADC_CR2.ACCRST 为 1 清零 ADC_RESULTACC 寄存器，同时设置 ADC_CR2.ACCEN 为 1 使能 ADC 转换结果的自动累加功能。

每次 ADC 转换完成后，ADC_ISR.EOC 标志位自动置 1，转换结果保存在 ADC_RESULT0 寄存器中，同时自动对转换结果进行累加，累加值保存在 ADC_RESULTACC 寄存器。如未达到设定的 ADC 转换次数，ADC 会继续进行转换；当达到了设定的 ADC 转换次数之后，多次转换完成标志位 ADC_ISR.EOA 自动置 1，同时 ADC_START.START 位自动清 0，ADC 转换停止。

单通道多次转换模式的时序如下图所示：

图 21-4 单通道多次转换时序图



通过 START 位启动 ADC 单通道多次转换，参考操作流程如下：

步骤 1：设置 SYSCTRL_AHBEN.GPIOx 为 1，SYSCTRL_APBEN2.ADC 为 1，使能 ADC 通道对应的 GPIO 时钟、外部参考电压引脚对应的 GPIO 时钟和 ADC 工作时钟；

步骤 2：设置 ADC 通道对应的 GPIO 引脚为模拟功能，具体寄存器配置请参见 9 通用输入输出端口（GPIO）章节；

步骤 3：设置外部参考电压对应的 GPIO 引脚为模拟功能，具体寄存器配置请参见 9 通用输入输出端口(GPIO)章节；

注：

如果 ADC 参考电压没有选择外部参考电压引脚，则略过本步骤。

步骤 4：设置 ADC_CR0.EN 为 1，使能 ADC 模块；

步骤 5：查询等待 ADC_ISR.READY 位变为 1，等待 ADC 模块启动完成；

步骤 6：设置 ADC_CR0.MODE 为 1，选择单通道多次转换模式；

步骤 7：配置 ADC_CR0.REF，选择 ADC 的参考电压；

步骤 8：配置 ADC_CR0.SAM 及 ADC_CR0.CLK，设置 ADC 的采样速度及时钟选择；

步骤 9：配置 ADC_CR1.CHMUX，选择待转换的通道；

步骤 10：配置 ADC_CR2.CNT，设定转换次数；

步骤 11：设置 ADC_CR2.ACCRST 和 ADC_CR2.ACCEN 为 1，使能转换结果累加功能；

步骤 12：设置 ADC_IER.EOA 为 1，使能多次转换完成中断；

步骤 13：使能 NVIC 中断向量表中的 ADC 中断；

步骤 14：设置 ADC_ICR.EOA 为 0，清除 EOA 中断标志；

步骤 15：设置 ADC_START.START 为 1，启动 ADC 转换；

步骤 16：查询等待 ADC_ISR.EOA 变为 1，表示多次转换全部完成，ADC_START.START 位自动清 0，ADC 转换停止；此时用户可读取 ADC_RESULTACC 寄存器获得 ADC 转换结果累加值，除以转换次数即得到 ADC 转换结果；

步骤 17：如需对其它通道进行转换，重复执行步骤 9 至步骤 16；

步骤 18：设置 ADC_CR0.EN 为 0，关闭 ADC 模块。

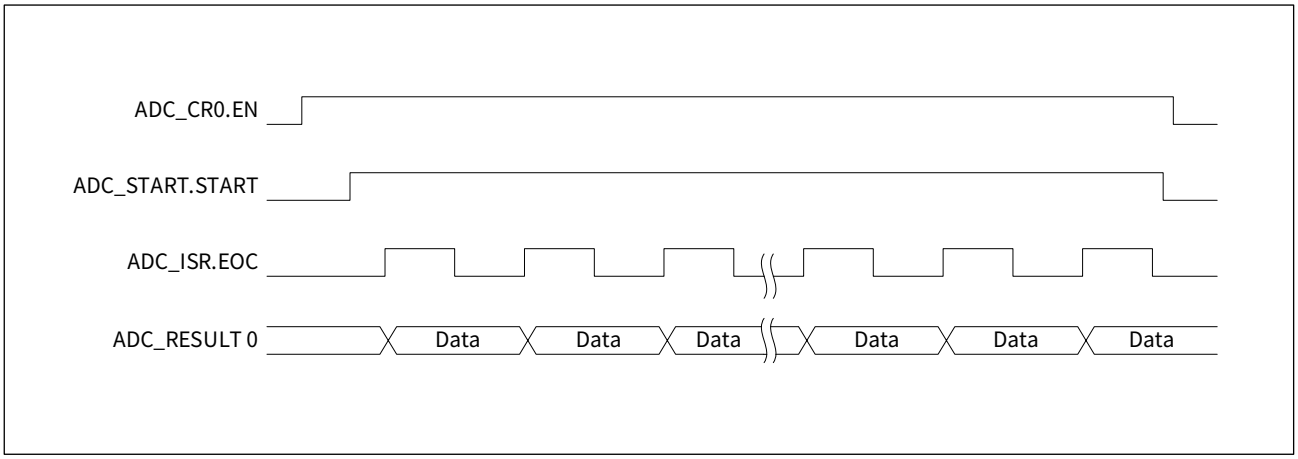
21.5.3 单通道连续转换模式（MODE=2）

在此模式下，无论是通过软件 START 位启动 ADC，还是外部触发启动，一旦启动 ADC 将对指定的某一个通道持续进行转换，直到 ADC_START.START 位清 0，才会停止转换。

每次 ADC 转换完成后，ADC_ISR.EOC 标志位自动置 1，转换结果保存在 ADC_RESULT0 寄存器中。用户应及时读取 ADC_RESULT0 中的转换结果，以避免转换结果溢出。用户向 ADC_START.START 位写 0，停止转换。

单通道连续转换模式的时序如下图所示：

图 21-5 单通道连续转换时序图



通过 START 位启动 ADC 单通道连续转换，参考操作流程如下：

步骤 1：设置 SYSCTRL_AHBEN.GPIOx 为 1，SYSCTRL_APBEN2.ADC 为 1，使能 ADC 通道对应的 GPIO 时钟、外部参考电压引脚对应的 GPIO 时钟和 ADC 工作时钟；

步骤 2：设置 ADC 通道对应的 GPIO 引脚为模拟功能，具体寄存器配置请参见 9 通用输入输出端口（GPIO）章节；

步骤 3：设置外部参考电压对应的 GPIO 引脚为模拟功能，具体寄存器配置请参见 9 通用输入输出端口(GPIO)章节；

注：

如果 ADC 参考电压没有选择外部参考电压引脚，则略过本步骤。

步骤 4：设置 ADC_CR0.EN 为 1，使能 ADC 模块；

步骤 5：查询等待 ADC_ISR.READY 位变为 1，等待 ADC 模块启动完成；

步骤 6：设置 ADC_CR0.MODE 为 2，选择单通道连续转换模式；

步骤 7：配置 ADC_CR0.REF，选择 ADC 的参考电压；

步骤 8：配置 ADC_CR0. SAM 及 ADC_CR0. CLK，设置 ADC 的采样速度及时钟选择；

步骤 9：配置 ADC_CR1.CHMUX，选择待转换的通道；

步骤 10：设置 ADC_START.START 为 1，启动 ADC 转换；

步骤 11：查询等待 ADC_ISR.EOC 变为 1，不断读取 ADC_RESULT0 寄存器，以获取 ADC 转换结果；

步骤 12：设置 ADC_START.START 为 0，停止 ADC 转换；

步骤 13：如需对其它通道进行转换，重复执行步骤 9 至步骤 13；

步骤 14：设置 ADC_CR0.EN 为 0，关闭 ADC 模块。

21.5.4 序列连续转换模式（MODE=3）

序列连续转换模式与单通道连续转换模式类似，不同之处在于，序列连续转换模式可对最多四个序列的通道进行轮流转换，而不是单个通道，每个序列 SQRy 可选择 16 个转换通道之一，具体由序列配置寄存器 ADC_SQR 的 SQRy 位域配置。待转换的序列配置由 ADC 序列配置寄存器 ADC_SQR 的 ENS 位域决定，如下表所示：

表 21-6 待转换的序列配置

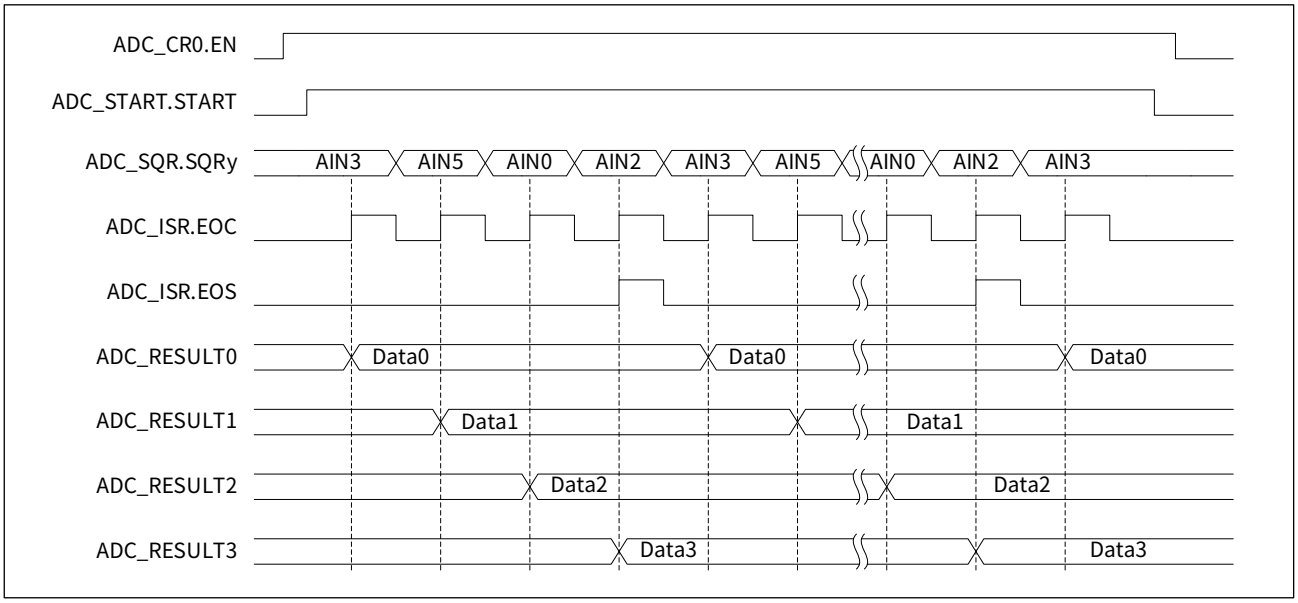
ADC_SQR.ENS	待转换的序列配置
00	仅转换 SQR0
01	转换 SQR0，SQR1
10	转换 SQR0，SQR1，SQR2
11	转换 SQR0，SQR1，SQR2，SQR3

在此模式下，无论是通过软件 START 位启动 ADC，还是外部触发启动，一旦启动 ADC 将对选择的转换序列持续进行转换，直到 ADC_START.START 位清 0，才会停止转换。

每次 ADC 转换完成后，ADC_ISR.EOC 标志位自动置 1，转换结果保存在与序列 SQR0~SQR3 相同序号的转换结果寄存器 ADC_RESULT0 ~ ADC_RESULT3 中。当所选择的转换序列全部转换完成后，序列转换完成标志位 ADC_ISR.EOS 被置 1。用户应及时读取转换结果，以避免转换结果溢出。用户向 ADC_START.START 位清 0，停止转换。

序列连续转换模式的时序如下图所示：

图 21-6 序列连续转换时序图



通过 START 位启动 ADC 序列连续转换，参考操作流程如下：

步骤 1：设置 SYSCTRL_AHBEN.GPIOx 为 1，SYSCTRL_APBEN2.ADC 为 1，使能 ADC 通道对应的 GPIO 时钟、外部参考电压引脚对应的 GPIO 时钟和 ADC 工作时钟；

步骤 2：设置 ADC 通道对应的 GPIO 引脚为模拟功能，具体寄存器配置请参见 9 通用输入输出端口（GPIO）章节；

步骤 3：设置外部参考电压对应的 GPIO 引脚为模拟功能，具体寄存器配置请参见 9 通用输入输出端口(GPIO)章节；

注：

如果 ADC 参考电压没有选择外部参考电压引脚，则略过本步骤。

步骤 4：设置 ADC_CR0.EN 为 1，使能 ADC 模块；

步骤 5：查询等待 ADC_ISR.READY 位变为 1，等待 ADC 模块启动完成；

步骤 6：设置 ADC_CR0.MODE 为 3，选择序列连续转换模式；

步骤 7：配置 ADC_CR0.REF，选择 ADC 的参考电压；

步骤 8：配置 ADC_CR0. SAM 及 ADC_CR0. CLK，设置 ADC 的采样速度及时钟选择；

步骤 9：配置 ADC_SQR.ENS，选择待转换的序列，如图 21-6 所示，设置 ADC_SQR.ENS 为 3，转换序列为 SQR0~SQR3；

步骤 10：配置 ADC_SQR.SQR0，选择待转换序列 SQR0 的待转换通道，如图 21-6 所示，设置 ADC_SQR.SQR0 为 3，序列 SQR0 的待转换通道为 AIN3；

步骤 11：配置 ADC_SQR.SQR1，选择待转换序列 SQR1 的待转换通道，如图 21-6 所示，设置 ADC_SQR.SQR1 为 5，序列 SQR1 的待转换通道为 AIN5；

步骤 12：配置 ADC_SQR.SQR2，选择待转换序列 SQR2 的待转换通道，如图 21-6 所示，设置 ADC_SQR.SQR2 为 0，序列 SQR2 的待转换通道为 AIN0；

步骤 13：配置 ADC_SQR.SQR3，选择待转换序列 SQR3 的待转换通道，如图 21-6 所示，设置 ADC_SQR.SQR3 为 2，序列 SQR3 的待转换通道为 AIN2；

步骤 14：设置 ADC_ICR 为 0，清除 ADC 中断标志；

步骤 15：设置 ADC_START.START 为 1，启动 ADC 转换；

步骤 16：查询等待 ADC_ISR.EOS 变为 1，依次不断读取 ADC_RESULT0 ~ ADC_RESULT3 寄存器，以获取各通道的 ADC 转换结果；当 ADC_ISR.EOS 为 1 时，表示一次 4 个通道的序列转换完成；

步骤 17：设置 ADC_START.START 为 0，停止 ADC 转换；

步骤 18：如需对其它通道进行转换，重复执行步骤 9 至步骤 18；

步骤 19：设置 ADC_CR0.EN 为 0，关闭 ADC 模块。

21.5.5 序列扫描转换模式（MODE=4）

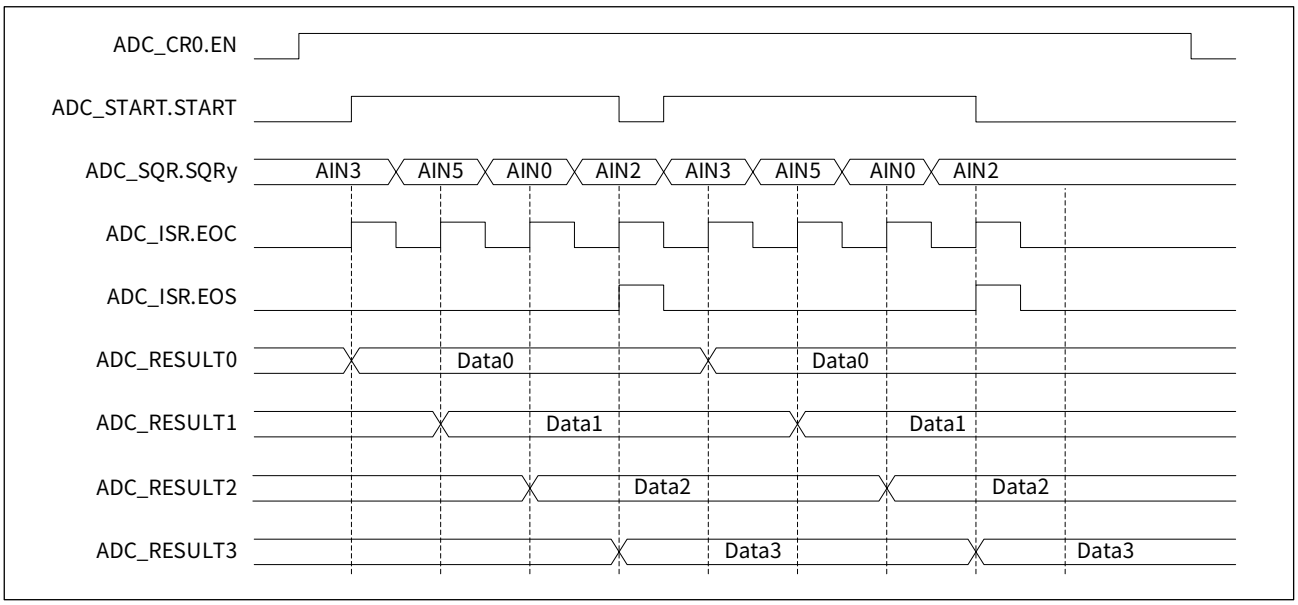
序列扫描转换模式与序列连续转换的不同之处在于，序列扫描转换模式仅完成一次对所选择的序列的转换。

在此模式下，无论是通过软件 START 位启动 ADC，还是外部触发启动，启动一次 ADC，将对选择的转换序列全部进行一次转换。

每次 ADC 转换完成后，ADC_ISR.EOC 标志位自动置 1，转换结果保存在与序列 SQR0 ~ SQR3 相同序号的转换结果寄存器 ADC_RESULT0~ADC_RESULT3 中。当所选择的转换序列全部转换完成后，ADC_ISR.EOS 标志位变为 1，ADC_START.START 位自动清 0，ADC 停止转换。

序列扫描转换模式的时序如下图所示：

图 21-7 序列扫描转换时序图



通过 START 位启动 ADC 序列扫描转换，参考操作流程如下：

步骤 1：设置 SYSCTRL_AHBEN.GPIOx 为 1，SYSCTRL_APBEN2.ADC 为 1，使能 ADC 通道对应的 GPIO 时钟、外部参考电压引脚对应的 GPIO 时钟和 ADC 工作时钟；

步骤 2：设置 ADC 通道对应的 GPIO 引脚为模拟功能，具体寄存器配置请参见 9 通用输入输出端口（GPIO）章节；

步骤 3：设置外部参考电压对应的 GPIO 引脚为模拟功能，具体寄存器配置请参见 9 通用输入输出端口(GPIO)章节；

注：

如果 ADC 参考电压没有选择外部参考电压引脚，则略过本步骤。

步骤 4：设置 ADC_CR0.EN 为 1，使能 ADC 模块；

步骤 5：查询等待 ADC_ISR.READY 位变为 1，等待 ADC 模块启动完成；

步骤 6：设置 ADC_CR0.MODE 为 4，选择序列扫描转换模式；

步骤 7：配置 ADC_CR0.REF，选择 ADC 的参考电压；

步骤 8：配置 ADC_CR0. SAM 及 ADC_CR0. CLK，设置 ADC 的采样速度及时钟选择；

步骤 9：配置 ADC_SQR.ENS，选择待转换的序列，如图 21-7 所示，设置 ADC_SQR.ENS 为 3，转换序列为 SQR0~SQR3；

步骤 10：配置 ADC_SQR.SQR0，选择待转换序列 SQR0 的待转换通道，如图 21-7 所示，设置 ADC_SQR.SQR0 为 3，序列 SQR0 的待转换通道为 AIN3；

步骤 11：配置 ADC_SQR.SQR1，选择待转换序列 SQR1 的待转换通道，如图 21-7 所示，设置 ADC_SQR.SQR1 为 5，序列 SQR1 的待转换通道为 AIN5；

步骤 12：配置 ADC_SQR.SQR2，选择待转换序列 SQR2 的待转换通道，如图 21-7 所示，设置 ADC_SQR.SQR2 为 0，序列 SQR2 的待转换通道为 AIN0；

步骤 13：配置 ADC_SQR.SQR3，选择待转换序列 SQR3 的待转换通道，如图 21-7 所示，设置 ADC_SQR.SQR3 为 2，序列 SQR3 的待转换通道为 AIN2；

步骤 14：设置 ADC_ICR 为 0，清除 ADC 中断标志；

步骤 15：设置 ADC_START.START 为 1，启动 ADC 转换；

步骤 16：等待 ADC_ISR.EOS 变为 1，依次读取 ADC_RESULT0~ADC_RESULT3 寄存器，以获取对应通道的 ADC 转换结果。当 ADC_ISR.EOS 变为 1 时，表示一次 4 个通道的序列转换完成，ADC_START.START 位自动清 0，ADC 转换停止；

步骤 17：如需对其它通道进行转换，重复执行步骤 9 至步骤 17；

步骤 18：设置 ADC_CR0.EN 为 0，关闭 ADC 模块。

21.5.6 序列多次转换模式（MODE=5）

序列多次转换模式与序列扫描转换模式的不同之处在于，序列扫描转换模式仅完成一次对所选择的序列的转换，序列多次转换则连续转换多次，序列转换次数由 ADC_CR2.CNT 位域值决定，默认转换次数是 1。

多次转换模式下必须使能累加转换功能，即设置 ADC_CR2.ACCRST 为 1 清零 ADC_RESULTACC 寄存器，同时设置 ADC_CR2.ACCEN 为 1 使能 ADC 转换结果的自动累加功能。

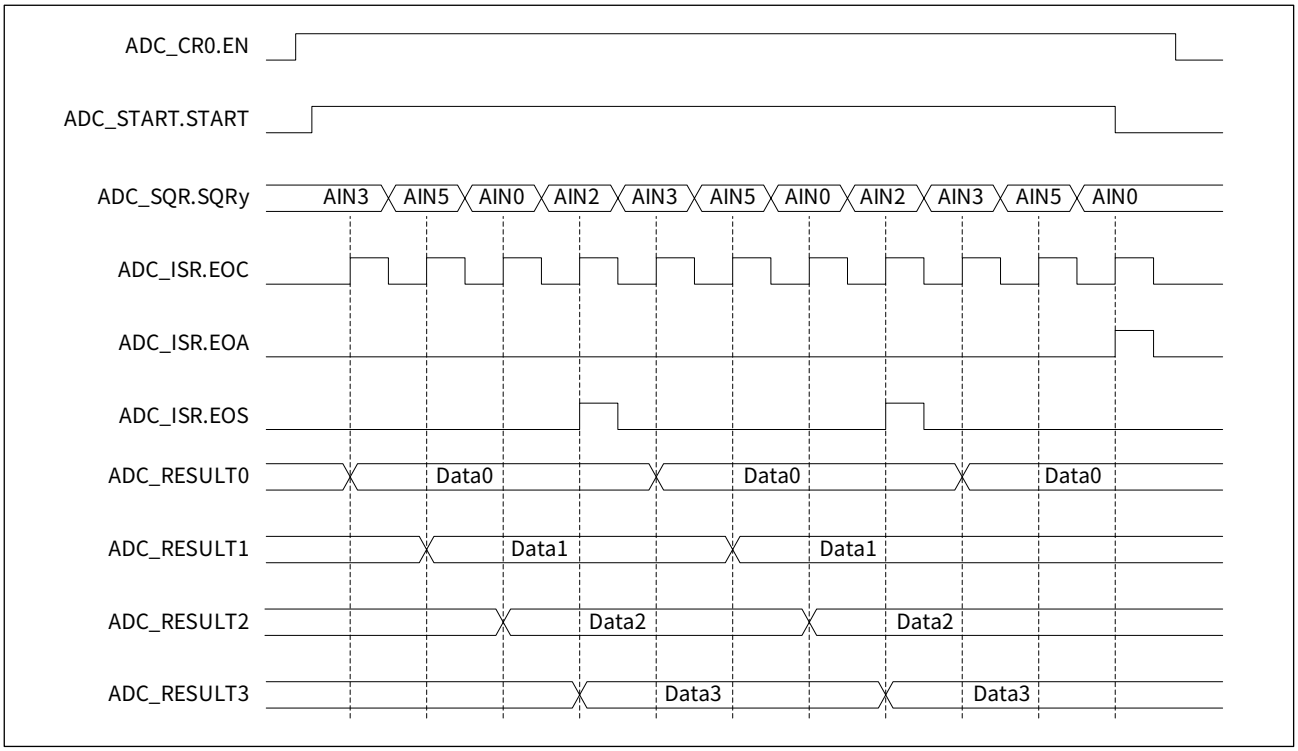
在此模式下，无论是通过软件 START 位启动 ADC，还是外部触发启动，一旦启动 ADC 将对选择的转换序列持续进行轮流转换，直到达到 ADC_CR2.CNT 位值的序列转换次数，才会停止转换。

每次 ADC 转换完成后，ADC_ISR.EOC 标志位自动置 1，转换结果保存在与序列 SQR0~SQR3 相同序号的转换结果寄存器 ADC_RESULT0 ~ ADC_RESULT3 中，同时自动对转换结果进行累加，累加值保存在 ADC_RESULTACC 寄存器。当所选择的转换序列全部转换完成后，ADC_ISR.EOS 标志位变为 1，如未达到 ADC_CR2.CNT 位域值设置的序列转换次数，将继续转换。用户应及时读取转换结果，以避免转换结果溢出。

当序列转换次数达到 ADC_CR2.CNT 位值时，ADC_ISR.EOA 标志位置 1，ADC_START.START 位自动清 0，ADC 转换停止。

序列多次转换模式的时序如下图所示，其中，ADC_CR2.CNT 为 10，转换次数为 11 次。

图 21-8 序列多次转换时序图



通过 START 位启动 ADC 序列多次转换，参考操作流程如下：

步骤 1：设置 SYSCTRL_AHBEN.GPIOx 为 1，SYSCTRL_APBEN2.ADC 为 1，使能 ADC 通道对应的 GPIO 时钟、外部参考电压引脚对应的 GPIO 时钟和 ADC 工作时钟；

步骤 2：设置 ADC 通道对应的 GPIO 引脚为模拟功能，具体寄存器配置请参见 9 通用输入输出端口（GPIO）章节；

步骤 3：设置外部参考电压对应的 GPIO 引脚为模拟功能，具体寄存器配置请参见 9 通用输入输出端口(GPIO)章节；

注：

如果 ADC 参考电压没有选择外部参考电压引脚，则略过本步骤。

步骤 4：设置 ADC_CR0.EN 为 1，使能 ADC 模块；

步骤 5：查询等待 ADC_ISR.READY 位变为 1，等待 ADC 模块启动完成；

步骤 6：设置 ADC_CR0.MODE 为 5，选择序列多次转换模式；

步骤 7：配置 ADC_CR0.REF，选择 ADC 的参考电压；

步骤 8：配置 ADC_CR0. SAM 及 ADC_CR0. CLK，设置 ADC 的采样速度及时钟选择；

步骤 9：配置 ADC_SQR.ENS，选择待转换的序列，如图 21-8 所示，设置 ADC_SQR.ENS 为 3，转换序列为 SQR0~SQR3；

步骤 10：配置 ADC_SQR.SQR0，选择待转换序列 SQR0 的待转换通道，如图 21-8 所示，设置 ADC_SQR.SQR0 为 3，SQR0 的待转换通道为 AIN3；

步骤 11：配置 ADC_SQR.SQR1，选择待转换序列 SQR1 的待转换通道，如图 21-8 所示，设置 ADC_SQR.SQR1 为 5，SQR1 的待转换通道为 AIN5；

步骤 12：配置 ADC_SQR.SQR2，选择待转换序列 SQR2 的待转换通道，如图 21-8 所示，设置 ADC_SQR.SQR2 为 0，SQR2 的待转换通道为 AIN0；

步骤 13：配置 ADC_SQR.SQR3，选择待转换序列 SQR3 的待转换通道，如图 21-8 所示，设置 ADC_SQR.SQR3 为 2，SQR3 的待转换通道为 AIN2；

步骤 14：配置 ADC_CR2.CNT，如图 21-8 所示，设置 ADC_CR2.CNT 为 10，则转换次数为 11；

步骤 15：设置 ADC_CR2.ACCRST 和 ADC_CR2.ACCEN 为 1，使能转换结果累加功能；

步骤 16：设置 ADC_ICR 为 0，清除 ADC 中断标志；

步骤 17：设置 ADC_START.START 为 1，启动 ADC 转换；

步骤 18：循环等待 ADC_ISR.EOS 变为 1，依次不断读取 ADC_RESULT0 ~ ADC_RESULT3 寄存器，以获取对应通道的 ADC 转换结果；当 ADC_ISR.EOS 为 1 时，表示一次 4 个通道的序列转换完成；

步骤 19：循环等待 ADC_ISR.EOA 变为 1，其间，依次不断读取 ADC_RESULT0~ADC_RESULT3 寄存器，以获取 ADC 转换结果。当 ADC_ISR.EOA 为 1 时，表示多次转换全部完成，ADC_START.START 位自动清 0，ADC 转换停止；

步骤 20：如需对其它通道进行转换，重复执行步骤 9 至步骤 19；

步骤 21：设置 ADC_CR0.EN 为 0，关闭 ADC 模块。

21.5.7 序列断续转换模式（MODE=6）

在序列断续转换模式下，每次启动 ADC，仅转换当前序列，而不是选择的所有序列。

在此模式下，可以通过软件 START 位启动 ADC，或者选择外部触发启动。

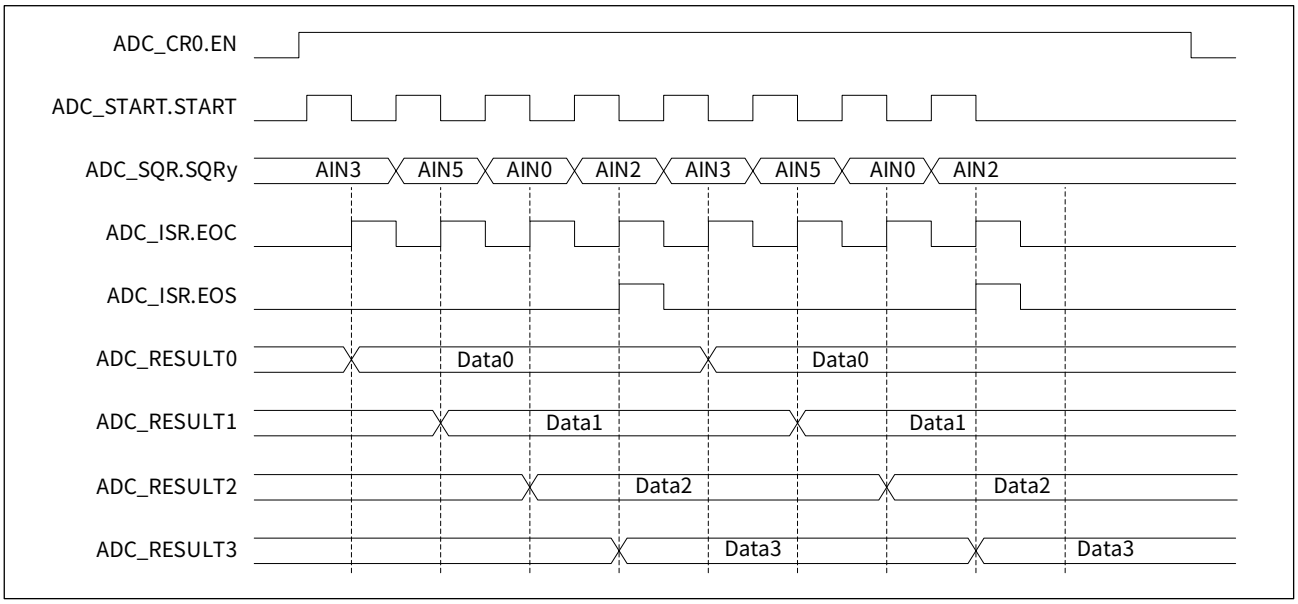
每次启动 ADC，当前 ADC 转换序列 SQRy（y = 0 ~ 3）中的指定通道，执行一次 ADC 转换。ADC 转换完成，ADC_ISR.EOC 标志位自动置 1，转换结果保存在对应的 ADC_RESULTy（y = 0 ~ 3）中，同时，ADC_START.START 位自动清 0，ADC 转换结束，等待再次启动 ADC 转换。

下一次再启动 ADC 时，则待转换通道自动切换为下一个转换序列 SQR(y+1) 的指定通道，再执行一次 ADC 转换。如果当前完成的转换序列是 SQR3，则下一个待转换序列自动重设为 SQR0。

当所有转换序列 SQRy 的全部通道转换完成后，ADC_ISR.EOS 标志位变为 1。

序列断续转换模式的时序如下图所示：

图 21-9 序列断续转换时序图



通过 START 位启动 ADC 序列断续转换，参考操作流程如下：

步骤 1：设置 SYSCTRL_AHBEN.GPIOx 为 1，SYSCTRL_APBEN2.ADC 为 1，使能 ADC 通道对应的 GPIO 时钟、外部参考电压引脚对应的 GPIO 时钟和 ADC 工作时钟；

步骤 2：设置 ADC 通道对应的 GPIO 引脚为模拟功能，具体寄存器配置请参见 9 通用输入输出端口（GPIO）章节；

步骤 3：设置外部参考电压对应的 GPIO 引脚为模拟功能，具体寄存器配置请参见 9 通用输入输出端口(GPIO)章节；

注：

如果 ADC 参考电压没有选择外部参考电压引脚，则略过本步骤。

步骤 4：设置 ADC_CR0.EN 为 1，使能 ADC 模块；

步骤 5：查询等待 ADC_ISR.READY 位变为 1，等待 ADC 模块启动完成；

步骤 6：设置 ADC_CR0.MODE 为 6，选择序列断续转换模式；

步骤 7：配置 ADC_CR0.REF，选择 ADC 的参考电压；

步骤 8：配置 ADC_CR0. SAM 及 ADC_CR0. CLK，设置 ADC 的采样速度及时钟选择；

步骤 9：配置 ADC_SQR.ENS，选择待转换的序列，如图 21-9 所示，设置 ADC_SQR.ENS 为 3，转换序列为 SQR0~SQR3；

步骤 10：配置 ADC_SQR.SQR0，选择待转换序列 SQR0 的待转换通道，如图 21-9 所示，设置 ADC_SQR.SQR0 为 3，序列 SQR0 的待转换通道为 AIN3；

步骤 11：配置 ADC_SQR.SQR1，选择待转换序列 SQR1 的待转换通道，如图 21-9 所示，设置 ADC_SQR.SQR1 为 5，序列 SQR1 的待转换通道为 AIN5；

步骤 12：配置 ADC_SQR.SQR2，选择待转换序列 SQR2 的待转换通道，如图 21-9 所示，设置 ADC_SQR.SQR2 为 0，序列 SQR2 的待转换通道为 AIN0；

步骤 13：配置 ADC_SQR.SQR3，选择待转换序列 SQR3 的待转换通道，如图 21-9 所示，设置 ADC_SQR.SQR3 为 2，序列 SQR3 的待转换通道为 AIN2；

步骤 14：设置 ADC_ICR 为 0，清除 ADC 中断标志；

步骤 15：设置 ADC_START.START 为 1，启动 ADC 转换；

步骤 16：等待 ADC_ISR.EOC 变为 1，选择读取 ADC_RESULT0 寄存器，以获取转换序列 SQR0 指定的通道的 ADC 转换结果；

步骤 17：设置 ADC_START.START 为 1，再次启动 ADC 转换；

步骤 18：等待 ADC_ISR.EOC 变为 1，选择读取 ADC_RESULT1 寄存器，以获取转换序列 SQR1 指定的通道的 ADC 转换结果；

步骤 19：设置 ADC_START.START 为 1，再次启动 ADC 转换；

步骤 20：等待 ADC_ISR.EOC 变为 1，选择读取 ADC_RESULT2 寄存器，以获取转换序列 SQR2 指定的通道的 ADC 转换结果；

步骤 21：设置 ADC_START.START 为 1，再次启动 ADC 转换；

步骤 22：等待 ADC_ISR.EOC 变为 1，选择读取 ADC_RESULT3 寄存器，以获取转换序列 SQR3 指定的通道的 ADC 转换结果；当 ADC_ISR.EOS 为 1 时，表示一次 4 个通道的序列转换完成；

步骤 23：重复执行步骤 15 至步骤 22，继续执行 ADC 转换；如需对其它通道进行转换，重复执行步骤 8 至步骤 22；

步骤 24：设置 ADC_CR0.EN 为 0，关闭 ADC 模块。

21.6 累加转换功能

累加转换功能，可以在 ADC 的任何工作模式下进行，且在多次转换模式下必须使能累加转换功能。

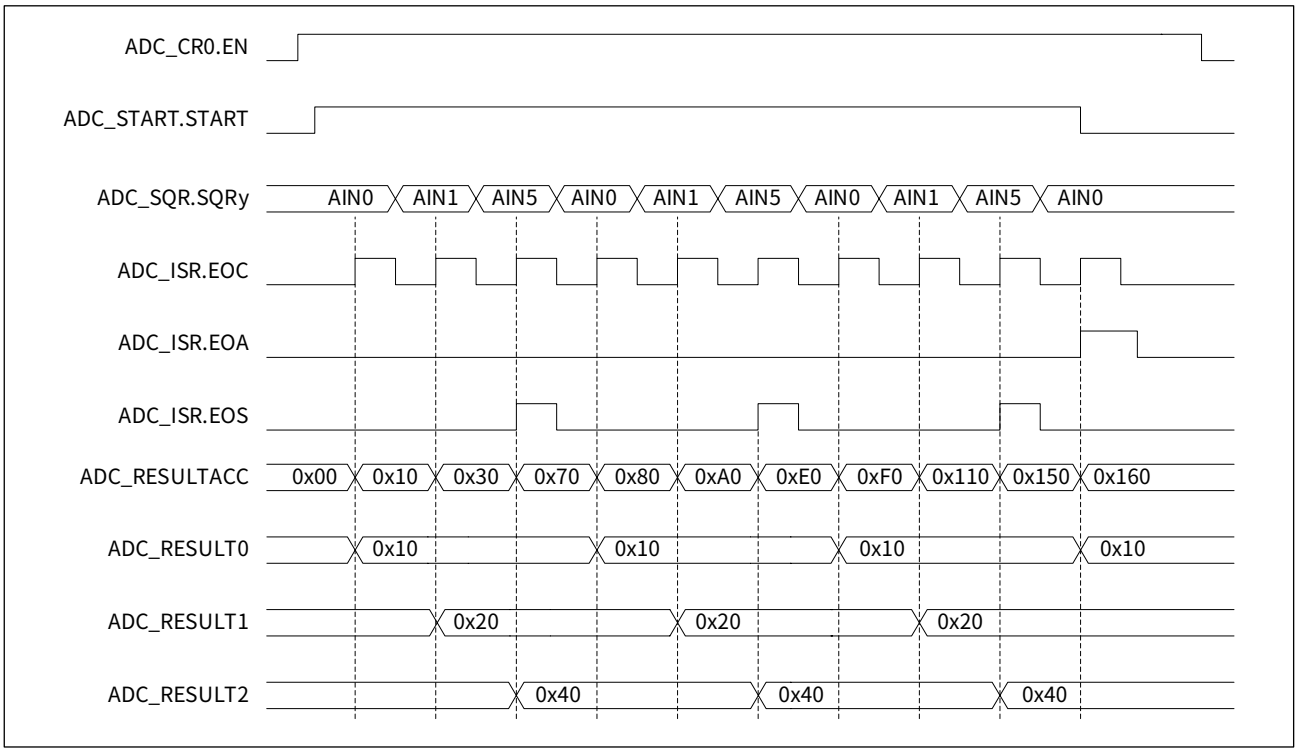
设置 ADC_CR2.ACCEN 为 1，使能 ADC 转换结果的自动累加功能。ADC 每完成一次通道转换，就自动对转换结果进行累加，累加值保存在 ADC_RESULTACC 寄存器。

使用累加功能前，必须先清零 ADC_RESULTACC 寄存器 , 设置 ADC_CR2.ACCRST 为 1 可使 ADC_RESULTACC 寄存器清 0。

图 21-10 累加转换时序图演示了对 AIN0、 AIN1、 AIN5 三个通道进行 10 次连续转换累加的过程，其中，假定 AIN0、 AIN1、 AIN5 的 ADC 转换结果依次为 0x010、0x020、0x040。

设置 ADC_START.START 为 1 启动 ADC， ADC 内部的状态机就会依次对 AIN0、 AIN1、 AIN5 进行转换，直到总转换次数达到 10 次，ADC 转换才停止。每次转换完成时， ADC_RESULTACC 寄存器自动累加转换结果。

图 21-10 累加转换时序图



通过 START 位启动 ADC 序列多次转换，并对转换结果进行累加，操作流程如下：

步骤 1：设置 SYSCTRL_AHBEN.GPIOx 为 1，SYSCTRL_APBEN2.ADC 为 1，使能 ADC 通道对应的 GPIO 时钟、外部参考电压引脚对应的 GPIO 时钟和 ADC 工作时钟；

步骤 2：设置 ADC 通道对应的 GPIO 引脚为模拟功能，具体寄存器配置请参见 9 通用输入输出端口（GPIO）章节；

步骤 3：设置外部参考电压对应的 GPIO 引脚为模拟功能，具体寄存器配置请参见 9 通用输入输出端口(GPIO)章节；

注：

如果 ADC 参考电压没有选择外部参考电压引脚，则略过本步骤。

步骤 4：设置 ADC_CR0.EN 为 1，使能 ADC 模块；

步骤 5：查询等待 ADC_ISR.READY 位变为 1，等待 ADC 模块启动完成；

步骤 6：设置 ADC_CR0.MODE 为 5，选择序列多次转换模式；

步骤 7：配置 ADC_CR0.REF，选择 ADC 的参考电压；

步骤 8：配置 ADC_CR0. SAM 及 ADC_CR0. CLK，设置 ADC 的采样速度及时钟选择；

步骤 9：配置 ADC_CR2.CNT，如图 21-10 所示，设置 ADC_CR2.CNT 为 9，则转换次数为 10；

步骤 10：设置 ADC_CR2.ACCEN 为 1，使能 ADC 转换结果自动累加控制；

步骤 11：设置 ADC_CR2.ACCRST 为 1，ADC 转换结果累加寄存器 ADC_RESULTACC 清 0；

步骤 12：配置 ADC_SQR.ENS，选择待转换的序列，如图 21-10 所示，设置 ADC_SQR.ENS 为 2，转换序列为 SQR0~SQR2；

步骤 13：配置 ADC_SQR.SQR0，选择待转换序列 SQR0 的待转换通道，如图 21-10 所示，设置 ADC_SQR.SQR0 为 0，序列 SQR0 的待转换通道为 AIN0；

步骤 14：配置 ADC_SQR.SQR1，选择待转换序列 SQR1 的待转换通道，如图 21-10 所示，设置 ADC_SQR.SQR1 为 1，序列 SQR1 的待转换通道为 AIN1；

步骤 15：配置 ADC_SQR.SQR2，选择待转换序列 SQR2 的待转换通道，如图 21-10 所示，设置 1ADC_SQR.SQR2 为 5，序列 SQR2 的待转换通道为 AIN5；

步骤 16：设置 ADC_IER.EOA 为 1，使能多次转换完成中断；

步骤 17：设置 ADC_ICR.EOA 为 0，清除 ADC_ISR.EOA 中断标志；

步骤 18：设置 ADC_ICR 为 0，清除 ADC 中断标志；

步骤 19：设置 ADC_START.START 为 1，启动 ADC 转换；

步骤 20：循环等待 ADC_ISR.EOC 变为 1，不断读取 ADC_RESULTACC 寄存器，以获取 ADC 转换结果累加值；当 ADC_ISR.EOS 为 1 时，表示一次 3 个通道的序列转换完成；

步骤 21：等待 ADC_ISR.EOA 变为 1，其间，不断读取 ADC_RESULTACC 寄存器；当 ADC_ISR.EOA 为 1 时，则表示多次转换全部完成，ADC_START.START 位自动清 0，ADC 转换停止；本例中，当 AIN0 完成了第 4 次 ADC 转换之后，ADC 转换停止；

步骤 22：设置 ADC_CR0.EN 为 0，关闭 ADC 模块。

21.7 自动关闭模式

用户可以通过设置 ADC_START.AUTOSTOP 为 1，使能 ADC 自动关闭功能。当指定的 ADC 转换完成之后，ADC_CR0.EN 位自动清 0，ADC 功能禁用。如果需要继续进行 ADC 转换，必须重新设置 ADC_CR0.EN 为 1，以使能 ADC 模块。

如果采用单通道单次转换模式，当转换完成后，自动关闭 ADC 使能。

如果采用序列扫描转换模式，当所有序列转换完成后，自动关闭 ADC 使能。

如果采用多次转换模式（单通道多次转换模式或序列多次转换模式），当达到设定的转换次数之后，ADC 转换停止，自动关闭 ADC 使能。

如果采用连续转换模式（单通道连续转换模式或序列连续转换模式），转换完成后，不会自动关闭 ADC 使能。设置 ADC_START.START 为 0，将关闭连续转换，同时自动关闭 ADC 使能。

采用序列断续转换模式时，当所有选择通道转换完成后，自动关闭 ADC 使能。

21.8 外部触发源

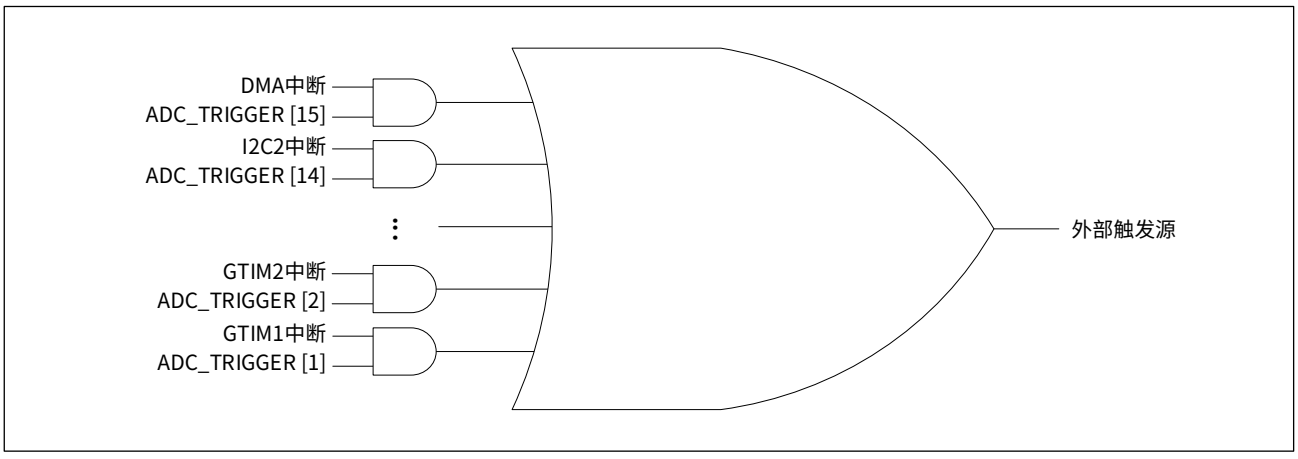
ADC 转换既可以通过软件启动（即设置 ADC_START.START 为 1），也可通过外部触发启动，触发源由外部触发寄存器 ADC_TRIGGER 选择，有 15 种触发 ADC 方式，详见下表：

表 21-7 ADC 转换外部触发源

ADC_TRIGGER 位域	位域名称	功能描述
15	DMA	DMA 中断触发 ADC 启动
14	I2C2	I2C2 中断触发 ADC 启动
13	I2C1	I2C1 中断触发 ADC 启动
12	SPI2	SPI2 中断触发 ADC 启动
11	SPI1	SPI1 中断触发 ADC 启动
10	UART3	UART3 中断触发 ADC 启动
9	UART2	UART2 中断触发 ADC 启动
8	UART1	UART1 中断触发 ADC 启动
7	BTIM3	BTIM3 中断触发 ADC 启动
6	BTIM2	BTIM2 中断触发 ADC 启动
5	BTIM1	BTIM1 中断触发 ADC 启动
4	GTIM4	GTIM4 中断触发 ADC 启动
3	GTIM3	GTIM3 中断触发 ADC 启动
2	GTIM2	GTIM2 中断触发 ADC 启动
1	GTIM1	GTIM1 中断触发 ADC 启动

ADC 转换外部触发源示意图如下图所示：

图 21-11 ADC 转换外部触发源示意图



21.9 模拟看门狗

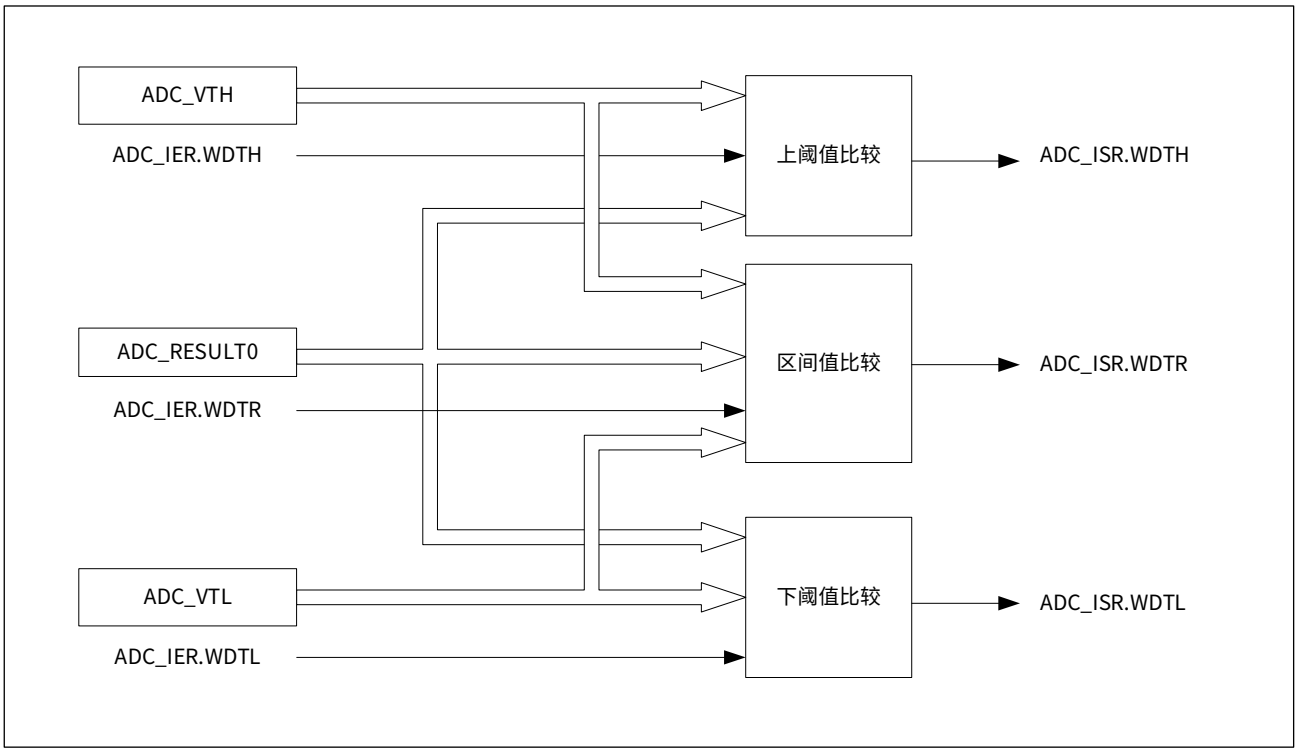
模拟看门狗功能，支持将 ADC 转换结果与用户设定的阈值进行比较，支持上阈值、下阈值、区间值比较，通过阈值寄存器 ADC_VTH 和 ADC_VTL 设置比较阈值。

模拟看门狗功能只在单通道模式时起作用。设置控制寄存器 ADC_CR1 的 WDTALL 位域为 1，使能模拟看门狗，通过 ADC_CR1 寄存器的 WDTCH 位域使能指定通道的模拟看门狗功能。

模拟看门狗功能常用于对模拟量的自动监测，如果设置了中断使能寄存器 ADC_IER 的相应位域（WDTR、WDTH、WDTL），当 ADC 转换结果符合用户预期时，将产生中断请求。

模拟看门狗的阈值比较示意图如下图所示：

图 21-12 ADC 阈值比较示意图



上阈值比较：当转换结果位于 [ADC_VTH，4095] 区间内时，ADC_ISR.WDTH 标志位置 1。

下阈值比较：当转换结果位于 [0，ADC_VTL）区间内时，ADC_ISR.WDTL 标志位置 1。

区间值比较：当转换结果位于 [ADC_VTL，ADC_VTH）区间内时，ADC_ISR.WDTR 标志位置 1。

21.10 温度传感器

CW32F020 内置温度传感器模块，传感器的输出电压随温度变化，设置 ADC 模块的采样通道为内部温度传感器，通过 ADC 测量结果可计算得到当前的环境温度。

温度传感器默认处于关闭状态，通过设置控制寄存器 ADC_CR0 的 TSEN 位域为 1，使能温度传感器。

环境温度计算公式如下：

$$环境温度 = T_0 \times 0.5 + 0.0924 \times V_{ref} \times (AdcValue - Trim)$$

其中：

V_{ref} 是当前 ADC 模块的参考电压，取值为 1.5V 或 2.5V。

T_0 是 8 位的初始校准温度值，记录在芯片的 FLASH 存储器中，其地址是 0x0001 2609，单位是 0.5 摄氏度，读取出来的值需要除以 2，才是实际的温度。

AdcValue 是 ADC 模块测量温度传感器输出电压的 ADC 转换结果，取值范围为 0 ~ 4095。

Trim 是 16 位的校准值，计算时需要从芯片的 FLASH 存储器中读出，其存放地址如下表所示：

表 21-8 ADC 校准值地址

ADC 参考电压	校准值存放地址	校准值精度
内部 1.5V	0x0001 260A - 0x0001 260B	±3°C
内部 2.5V	0x0001 260C - 0x0001 260D	±3°C

计算示例如下：

条件 1：V_{ref}=1.5、AdcValue = 0x8CB、Trim = 0x883、T0 = 0x32

温度 1：0x32×0.5 + 0.0924×1.5×(0x8CB – 0x883) = 35°C

条件 2：V_{ref} = 2.5、 AdcValue = 0x599、 Trim = 0x516、T0 = 0x32

温度 2：0x32×0.5 + 0.0924×2.5×(0x599 – 0x516) = 55.3°C

通过 ADC 测量环境温度的参考操作流程如下：

步骤 1：设置 SYSCTRL_APBEN2.ADC 为 1，使能 ADC 配置时钟及工作时钟；

步骤 2：设置 ADC_CR0.EN 为 1，使能 ADC 模块；

步骤 3：查询等待 ADC_ISR.READY 位变为 1，等待 ADC 模块启动完成；

步骤 4：设置 ADC_CR0.MODE 为 0，选择单通道单次转换模式；

步骤 5：配置 ADC_CR0.REF，选择 ADC 的参考电压为内部 1.5V 或内部 2.5V；

步骤 6：配置 ADC_CR0. SAM 及 ADC_CR0. CLK，设置 ADC 的转换速度；

步骤 7：设置 ADC_CR0.TSEN 为 1，使能温度传感器；

步骤 8：设置 ADC_CR1.CHMUX 为 0x0E，选择待转换的通道为温度传感器的电压输出；

步骤 9：设置 ADC_CR0. BUF 为 1，使能内置跟随器；

步骤 10：设置 ADC_ICR.EOC 为 0，清除 ADC_ISR.EOC 标志；

步骤 11：设置 ADC_START.START 为 1，启动 ADC 转换；

步骤 12：等待 ADC_ICR.EOC 变为 1，读取 ADC_RESULT0 寄存器，以获取 ADC 转换结果；

步骤 13：设置 ADC_CR0.EN 为 0，关闭 ADC 模块；

步骤 14：读取 T0 及 Trim，根据公式计算出当前的环境温度。

21.11 ADC 中断

ADC 中断请求，如下表所示：

表 21-9 ADC 中断源与中断标志

中断源	中断标志	中断使能	标志清除
转换结果溢出	ADC_ISR.OVW	ADC_IER.OVW 置 1	ADC_ICR.OVW 清 0
转换结果 >= ADC_VTL，且 < ADC_VTH	ADC_ISR.WDTR	ADC_IER.WDTR 置 1	ADC_ICR.WDTR 清 0
转换结果 >= ADC_VTH	ADC_ISR.WDTH	ADC_IER.WDTH 置 1	ADC_ICR.WDTH 清 0
转换结果 < ADC_VTL	ADC_ISR.WDTL	ADC_IER.WDTL 置 1	ADC_ICR.WDTL 清 0
ADC 多次转换完成	ADC_ISR.EOA	ADC_IER.EOA 置 1	ADC_ICR.EOA 清 0
ADC 序列转换完成	ADC_ISR.EOS	ADC_IER.EOS 置 1	ADC_ICR.EOS 清 0
ADC 转换完成	ADC_ISR.EOC	ADC_IER.EOC 置 1	ADC_ICR.EOC 清 0

21.12 寄存器列表

ADC 基地址：ADC_BASE = 0x4001 2400

表 21-10 ADC 寄存器列表

寄存器名称	寄存器地址	寄存器描述
ADC_CR0	ADC_BASE + 0x00	ADC 控制寄存器 0
ADC_CR1	ADC_BASE + 0x04	ADC 控制寄存器 1
ADC_START	ADC_BASE + 0x08	ADC 启动寄存器
ADC_SQR	ADC_BASE + 0x0C	ADC 序列配置寄存器
ADC_CR2	ADC_BASE + 0x10	ADC 控制寄存器 2
ADC_VTH	ADC_BASE + 0x14	ADC 高阈值寄存器
ADC_VTL	ADC_BASE + 0x18	ADC 低阈值寄存器
ADC_TRIGGER	ADC_BASE + 0x1C	ADC 外部触发寄存器
ADC_RESULT0	ADC_BASE + 0x20	ADC 转换结果 0 寄存器
ADC_RESULT1	ADC_BASE + 0x24	ADC 转换结果 1 寄存器
ADC_RESULT2	ADC_BASE + 0x28	ADC 转换结果 2 寄存器
ADC_RESULT3	ADC_BASE + 0x2C	ADC 转换结果 3 寄存器
ADC_RESULTACC	ADC_BASE + 0x30	ADC 转换结果累加值寄存器
ADC_IER	ADC_BASE + 0x34	ADC 中断使能寄存器
ADC_ICR	ADC_BASE + 0x38	ADC 中断标志清除寄存器
ADC_ISR	ADC_BASE + 0x3C	ADC 中断标志寄存器

21.13 寄存器描述

有关寄存器描述里所使用的缩写，请参见 1 文档约定章节。

21.13.1 ADC_CR0 控制寄存器 0

Address offset: 0x00 Reset value: 0x0000 0000

位域	名称	权限	功能描述
31:16	RFU	-	保留位，请保持默认值
15:14	BIAS	RW	请保持默认值不变
13	BUF	RW	内置跟随器使能控制 0：关闭跟随器，外部输入信号与 ADC 直接相连 1：打开跟随器，外部输入信号通过跟随器后与 ADC 相连 注 1：使能该功能时，最大转换速度为 200K SPS 注 2：以下条件必须使能该功能： 待转换的信号输出阻抗较高；待转换的信号为 VDDA/3、内置温度传感器输出电压、内部基准 1.2V 输出电压
12:11	SAM	RW	ADC 采样周期选择 00：5 个 ADCCLK 时钟周期 01：6 个 ADCCLK 时钟周期 10：8 个 ADCCLK 时钟周期 11：10 个 ADCCLK 时钟周期
10:8	CLK	RW	ADC 工作时钟配置 000：PCLK　　100：PCLK / 16 001：PCLK / 2　　101：PCLK / 32 010：PCLK / 4　　110：PCLK / 64 011：PCLK / 8　　111：PCLK / 128
7:6	REF	RW	ADC 参考电压选择 00：内置 1.5V 参考电压 01：内置 2.5V 参考电压 10：外部参考电压 ExRef 11：VDDA 电源电压
5	TSEN	RW	内置温度传感器使能控制 0：禁止内置温度传感器 1：使能内置温度传感器
4	BGREN	RW	内置 BGR 基准使能控制 0：禁止内置 BGR 基准 1：使能内置 BGR 基准

位域	名称	权限	功能描述
3:1	MODE	RW	ADC 工作模式配置 000：单通道单次转换模式 001：单通道多次转换模式，转换次数详见 CR2.CNT 010：单通道连续转换模式 011：序列连续转换模式 100：序列扫描转换模式 101：序列多次转换模式，转换次数详见 CR2.CNT 110：序列断续转换模式
0	EN	RW	ADC 使能控制 0：禁止 ADC 1：使能 ADC 注：使能 ADC 后，应等待 ISR.READY 变为 1 后方可执行 ADC 转换

21.13.2 ADC_CR1 控制寄存器 1

Address offset: 0x04 Reset value: 0x0000 0000

位域	名称	权限	功能描述
31:14	RFU	-	保留位，请保持默认值
13	WDTALL	RW	所有通道模拟看门狗使能控制 0：禁止模拟看门狗 1：使能模拟看门狗
12	RFU	-	保留位，请保持默认值
11:8	WDTCH	RW	单通道模拟看门狗使能配置 0000：AIN0 1000：AIN8 0001：AIN1 1001：AIN9 0010：AIN2 1010：AIN10 0011：AIN3 1011：AIN11 0100：AIN4 1100：AIN12 0101：AIN5 1101：VDDA/3 0110：AIN6 1110：TS 内置温度传感器 0111：AIN7 1111：1.2V 内核电压基准源
7	DMAEN	RW	DMA 触发信号使能控制 0：无功能 1：ADC 每次转换完成时触发 DMA
6	ALIGN	RW	ADC 转换结果对齐方式 0：右对齐，转换结果存储于 [11:0] 1：左对齐，转换结果存储于 [15:4]
5	DISCARD	RW	单通道 ADC 转换结果保存策略配置 0：覆盖未被读取的旧数据，保留新数据 1：丢弃新转换完成的数据，保留未读取的旧数据
4	RFU	-	保留位，请保持默认值
3:0	CHMUX	RW	单通道转换模式待转换通道配置 0000：AIN0 1000：AIN8 0001：AIN1 1001：AIN9 0010：AIN2 1010：AIN10 0011：AIN3 1011：AIN11 0100：AIN4 1100：AIN12 0101：AIN5 1101：VDDA/3 0110：AIN6 1110：TS 内置温度传感器 0111：AIN7 1111：1.2V 内核电压基准源

21.13.3 ADC_CR2 控制寄存器 2

Address offset: 0x10 Reset value: 0x0000 0000

位域	名称	权限	功能描述
31:10	RFU	-	保留位，请保持默认值
9	ACCRST	R0W1	累加值寄存器清 0 0：无功能 1：清零转换结果累加值寄存器 ADC_RESULTACC
8	ACCEN	RW	转换结果累加使能 0：禁止累加功能 1：使能累加功能
7:0	CNT	RW	多次转换模式转换次数配置 转换次数为 CNT+1

21.13.4 ADC_SQR 序列配置寄存器

Address offset: 0x0C Reset value: 0x0000 0000

位域	名称	权限	功能描述
31:18	RFU	-	保留位，请保持默认值
17:16	ENS	RW	待转换的序列配置 00：转换 SQR0 01：转换 SQR0-SQR1 10：转换 SQR0-SQR2 11：转换 SQR0-SQR3
15:12	SQR3	RW	序列 3 待转换通道配置 详见 SQR0
11:8	SQR2	RW	序列 2 待转换通道配置 详见 SQR0
7:4	SQR1	RW	序列 1 待转换通道配置 详见 SQR0
3:0	SQR0	RW	序列 0 待转换通道配置 0000：AIN0 1000：AIN8 0001：AIN1 1001：AIN9 0010：AIN2 1010：AIN10 0011：AIN3 1011：AIN11 0100：AIN4 1100：AIN12 0101：AIN5 1101：VDDA/3 0110：AIN6 1110：TS 内置温度传感器 0111：AIN7 1111：1.2V 内核电压基准源

21.13.5 ADC_VTH 高阈值寄存器

Address offset: 0x14 Reset value: 0x0000 0FFF

位域	名称	权限	功能描述
31:12	RFU	-	保留位，请保持默认值
11:0	VTH	RW	模拟看门狗检测高阈值

21.13.6 ADC_VTL 低阈值寄存器

Address offset: 0x18 Reset value: 0x0000 0000

位域	名称	权限	功能描述
31:12	RFU	-	保留位，请保持默认值
11:0	VTL	RW	模拟看门狗检测低阈值

21.13.7 ADC_TRIGGER 外部触发寄存器

Address offset: 0x1C Reset value: 0x0000 0000

位域	名称	权限	功能描述
31:16	RFU	-	保留位，请保持默认值
15	DMA	RW	DMA 中断触发 ADC 启动 0：禁止 1：使能
14	I2C2	RW	I2C2 中断触发 ADC 启动 0：禁止 1：使能
13	I2C1	RW	I2C1 中断触发 ADC 启动 1：使能 0：禁止
12	SPI2	RW	SPI2 中断触发 ADC 启动 0：禁止 1：使能
11	SPI1	RW	SPI1 中断触发 ADC 启动 0：禁止 1：使能
10	UART3	RW	UART3 中断触发 ADC 启动 0：禁止 1：使能

位域	名称	权限	功能描述
9	UART2	RW	UART2 中断触发 ADC 启动 0：禁止 1：使能
8	UART1	RW	UART1 中断触发 ADC 启动 0：禁止 1：使能
7	BTIM3	RW	BTIM3 中断触发 ADC 启动 0：禁止 1：使能
6	BTIM2	RW	BTIM2 中断触发 ADC 启动 0：禁止 1：使能
5	BTIM1	RW	BTIM1 中断触发 ADC 启动 1：使能 0：禁止
4	GTIM4	RW	GTIM4 中断触发 ADC 启动 0：禁止 1：使能
3	GTIM3	RW	GTIM3 中断触发 ADC 启动 0：禁止 1：使能
2	GTIM2	RW	GTIM2 中断触发 ADC 启动 0：禁止 1：使能
1	GTIM1	RW	GTIM1 中断触发 ADC 启动 0：禁止 1：使能
0	RFU	-	保留位，请保持默认值

21.13.8 ADC_START 启动寄存器

Address offset: 0x08 Reset value: 0x0000 0000

位域	名称	权限	功能描述
31:2	RFU	-	保留位，请保持默认值
1	AUTOSTOP	RW	ADC 自动禁止配置 0：转换完成后继续保持 CR0.EN 为 1 1：转换完成后自动设置 CR0.EN 为 0
0	START	RW	ADC 启动转换控制 0：停止 ADC 转换 1：启动 ADC 转换

21.13.9 ADC_IER 中断使能寄存器

Address offset: 0x34 Reset value: 0x0000 0000

位域	名称	权限	功能描述
31:7	RFU	-	保留位，请保持默认值
6	OVW	RW	转换结果溢出中断使能控制 0：禁止 1：使能
5	WDTR	RW	模拟看门狗区间中断使能控制 0：禁止 1：使能
4	WDTH	RW	模拟看门上阈值中断使能控制 0：禁止 1：使能
3	WDTL	RW	模拟看门狗下阈值中断使能控制 0：禁止 1：使能
2	EOA	RW	多次转换完成中断使能控制 0：禁止 1：使能
1	EOS	RW	序列转换完成中断使能控制 0：禁止 1：使能
0	EOC	RW	转换完成中断使能控制 0：禁止 1：使能

21.13.10 ADC_ISR 中断标志寄存器

Address offset: 0x3C Reset value: 0x0000 0000

位域	名称	权限	功能描述
31:8	RFU	-	保留位，请保持默认值
7	READY	RO	模拟电路初始化完成标志 0：初始化未完成 1：初始化已完成，可以开始进行 ADC 转换
6	OVW	RO	转换结果溢出标志 0：未发生该事件 1：结果寄存器中的数据尚未被读取前，又有新的转换完成
5	WDTR	RO	模拟看门狗区间标志 0：转换结果位于 [ADC_VTL，ADC_VTH) 区间外 1：转换结果位于 [ADC_VTL，ADC_VTH) 区间内
4	WDTH	RO	模拟看门狗上阈值标志 0：转换结果位于 [ADC_VTH，4096) 区间外 1：转换结果位于 [ADC_VTH，4096) 区间内
3	WDTL	RO	模拟看门狗下阈值标志 0：转换结果位于 [0，ADC_VTL) 区间外 1：转换结果位于 [0，ADC_VTL) 区间内
2	EOA	RO	多次转换完成标志 0：多次转换未完成 1：多次转换已完成
1	EOS	RO	序列转换完成标志 0：序列转换未完成 1：序列转换已完成
0	EOC	RO	转换完成标志 0：一次 ADC 转换未完成 1：一次 ADC 转换已完成

21.13.11 ADC_ICR 中断标志清除寄存器

Address offset: 0x38 Reset value: 0x0000 007F

位域	名称	权限	功能描述
31:7	RFU	-	保留位，请保持默认值
6	OVW	R1W0	转换结果溢出标志清 0 控制 W0：清除 ISR 寄存器中的相应标志 W1：无功能
5	WDTR	R1W0	模拟看门狗区间标志清 0 控制 W0：清除 ISR 寄存器中的相应标志 W1：无功能
4	WDTH	R1W0	模拟看门狗上阈值标志清 0 控制 W0：清除 ISR 寄存器中的相应标志 W1：无功能
3	WDTL	R1W0	模拟看门狗下阈值标志清 0 控制 W0：清除 ISR 寄存器中的相应标志 W1：无功能
2	EOA	R1W0	多次转换完成标志清 0 控制 W0：清除 ISR 寄存器中的相应标志 W1：无功能
1	EOS	R1W0	序列转换完成标志清 0 控制 W0：清除 ISR 寄存器中的相应标志 W1：无功能
0	EOC	R1W0	转换完成标志清 0 控制 W0：清除 ISR 寄存器中的相应标志 W1：无功能

21.13.12 ADC_RESULT0 转换结果 0 寄存器

Address offset: 0x20 Reset value: 0x0000 0000

位域	名称	权限	功能描述
31:16	RFU	-	保留位，请保持默认值
15:0	RESULT	RO	ADC 转换结果 0 寄存器

21.13.13 ADC_RESULT1 转换结果 1 寄存器

Address offset: 0x24 Reset value: 0x0000 0000

位域	名称	权限	功能描述
31:16	RFU	-	保留位，请保持默认值
15:0	RESULT	RO	ADC 转换结果 1 寄存器

21.13.14 ADC_RESULT2 转换结果 2 寄存器

Address offset: 0x28 Reset value: 0x0000 0000

位域	名称	权限	功能描述
31:16	RFU	-	保留位，请保持默认值
15:0	RESULT	RO	ADC 转换结果 2 寄存器

21.13.15 ADC_RESULT3 转换结果 3 寄存器

Address offset: 0x2C Reset value: 0x0000 0000

位域	名称	权限	功能描述
31:16	RFU	-	保留位，请保持默认值
15:0	RESULT	RO	ADC 转换结果 3 寄存器

21.13.16 ADC_RESULTACC 转换结果累加值寄存器

Address offset: 0x30 Reset value: 0x0000 0000

位域	名称	权限	功能描述
31:24	RFU	-	保留位，请保持默认值
23:0	RESULT	RO	ADC 转换结果累加值寄存器

22 模拟电压比较器（VC）

22.1 概述

CW32F020 内部集成 2 个模拟电压比较器（VC），用于比较两路模拟输入电压，并将比较结果从引脚输出。电压比较器的正端输入支持多达 8 路外部模拟输入，负端既支持 8 路外部模拟输入，又支持内部电压基准、内部电阻分压器、内部温度传感器等电压参考。比较结果输出具有滤波功能、迟滞窗口功能，以及极性选择。支持比较中断，可用于低功耗模式下唤醒 MCU。

22.2 主要特性

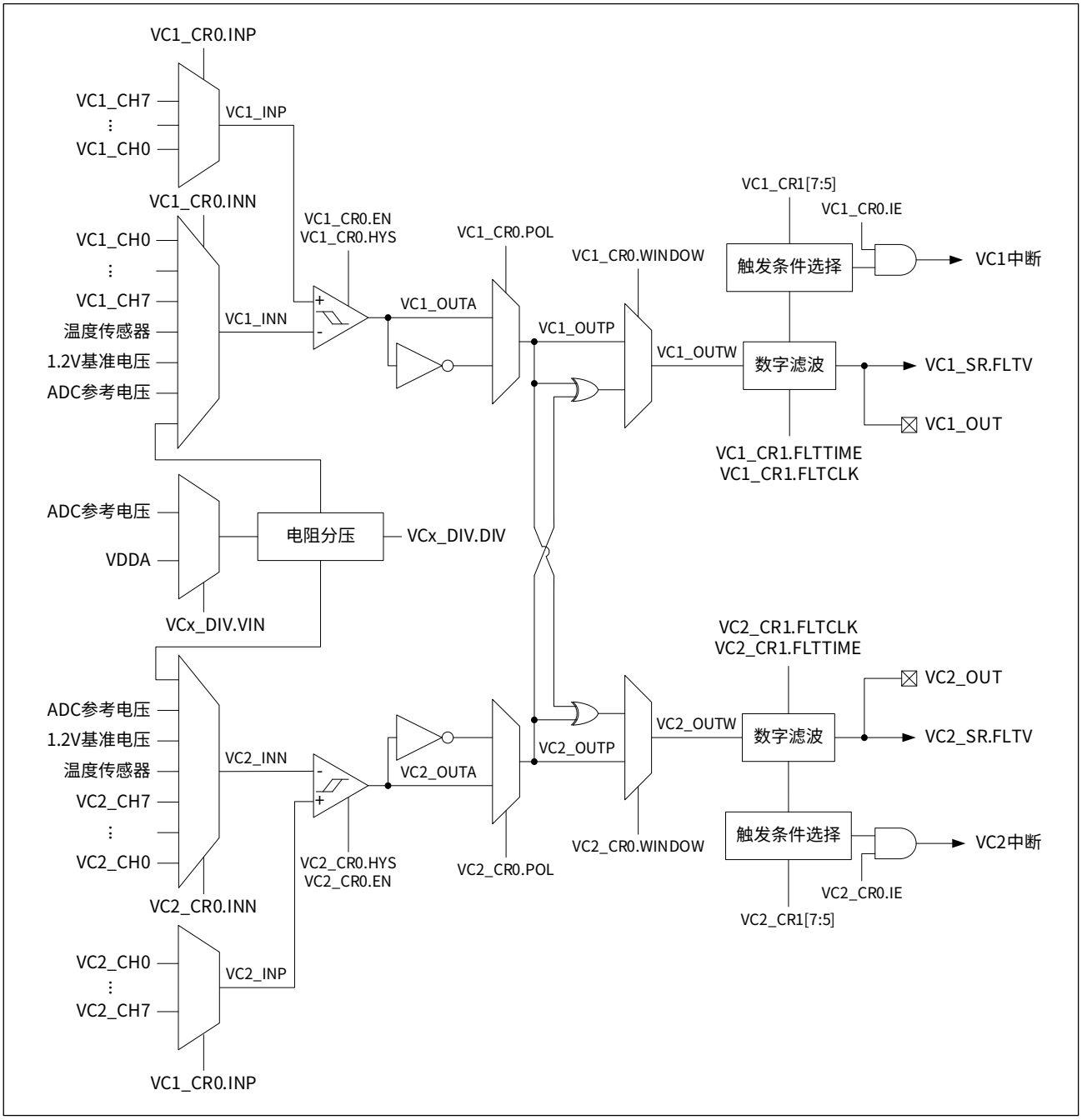
- 双路的模拟电压比较器 VC1、VC2
- 内部 64 阶电阻分压器
- 多达 8 路外部模拟信号输入
- 4 路片内模拟输入信号
  - 内置电阻分压器输出电压
  - 内置温度传感器输出电压
  - 内置 1.2V 基准电压
  - ADC 参考电压
- 可选择输出极性
- 支持迟滞窗口比较功能
- 可编程的滤波器和滤波时间
- 3 种中断触发方式，可组合使用
  - 高电平触发
  - 上升沿触发
  - 下降沿触发
- 支持低功耗模式下运行，中断唤醒 MCU

22.3 功能描述

22.3.1 功能框图

模拟电压比较器 (VC) 的功能框图如下图所示：

图 22-1 VC 功能框图



VC1、VC2 的正负端输入选择，由控制寄存器 VCx_CR0 的 INP、INN 位域选择，如下表所示：

表 22-1 VC 正负端输入信号配置

VCx_CR0.INP	VCx 正端输入信号	VCx_CR0.INN	VCx 负端输入信号
0000	VCx_CH0	0000	VCx_CH0
0001	VCx_CH1	0001	VCx_CH1
0010	VCx_CH2	0010	VCx_CH2
0011	VCx_CH3	0011	VCx_CH3
0100	VCx_CH4	0100	VCx_CH4
0101	VCx_CH5	0101	VCx_CH5
0110	VCx_CH6	0110	VCx_CH6
0111	VCx_CH7	0111	VCx_CH7
-	-	1000	内置电阻分压器输出电压
-	-	1001	ADC 模块所配置的参考电压 注：需使能 ADC_CR0.EN
-	-	1010	内置 1.2V 基准电压 注：需使能 ADC_CR0.BGREN
-	-	1011	内置温度传感器输出电压 注：需使能 ADC_CR0.TSEN 和 ADC_CR0.BGREN

内置的电阻分压配置由 VCx_DIV 寄存器 [1] 的三个位域控制：

- EN 位域，使能电阻分压器
- VIN 位域，选择分压器的输入电压，输入电压应大于 1.6V、小于等于 VCC
  - VIN 为 0，选择 VDDA
  - VIN 为 1，选择 ADC 模块配置的参考电压（注意，必须先使能 ADC）
- DIV 位域，分压比例系数，有效范围为 0 ~ 63，选择分压器的输出电压，计算公式如下：

$$\text{分压器输出电压} = \text{输入电压} \times (1 + \text{DIV}) / 64$$

注 1：

VC1_DIV 和 VC2_DIV 指向同一实体寄存器，VC1 和 VC2 共用电阻分压器电路。

在选择内置 1.2V 基准电压、或 ADC 模块参考电压作为比较器的负端输入时，需要设置 ADC 控制寄存器 ADC_CR0 的 BGREN 位域为 1，以使能芯片内部的 BGR 模块，内部 BGR 的启动时间大约为 20μs，VC 电压比较器需要等待内部 BGR 稳定后才能正常工作。

22.3.2 输入输出引脚

模拟电压比较器支持 8 路外部模拟信号输入，用户必须将对应 GPIO 端口配置为模拟功能（GPIOx_ANALOG.PINy = 1）。

模拟电压比较器支持将比较结果从引脚输出，用户必须将对应 GPIO 端口配置为数字输出，同时选择功能复用。

VC1、VC2 支持的输入输出引脚如下表所示：

表 22-2 VC 输入输出引脚配置

VC1 输入输出	GPIO	配置	VC2 输入输出	GPIO	配置
VC1_CH0	PA00	模拟	VC2_CH0	PA05	模拟
VC1_CH1	PA01	模拟	VC2_CH1	PA06	模拟
VC1_CH2	PA02	模拟	VC2_CH2	PA07	模拟
VC1_CH3	PA03	模拟	VC2_CH3	PB00	模拟
VC1_CH4	PA04	模拟	VC2_CH4	PB01	模拟
VC1_CH5	PA05	模拟	VC2_CH5	PB02	模拟
VC1_CH6	PA06	模拟	VC2_CH6	PB10	模拟
VC1_CH7	PA07	模拟	VC2_CH7	PB11	模拟
VC1_OUT	PA06	数字输出（AFR=3）	VC2_OUT	PA07	数字输出（AFR=3）
	PA00	数字输出（AFR=4）		PA02	数字输出（AFR=4）
	PA11	数字输出（AFR=4）		PA12	数字输出（AFR=4）

22.3.3 延迟 / 响应时间

设置控制寄存器 VCx_CR0 的 EN 位域为 1，使能 VC 模块。

从 VC 使能或 VC 的正负两端输入电压变化，到电压比较器输出正确比较结果的时间，被定义为比较器的延迟 / 响应时间。延迟 / 响应时间由控制寄存器 VCx_CR0 的 RESP 位域配置，响应时间值从 200ns 到 20μs 四档可调，且响应时间越短，VC 模块的功耗越大。

22.3.4 极性选择

电压比较器 VC1、VC2 的输出信号的极性，由控制寄存器 VCx_CR0 的 POL 位域设置：

- POL 为 1，VCx_OUTP 信号与 VCx_OUTA 信号极性相反，即正端大于负端时 VCx 输出低电平
- POL 为 0，VCx_OUTP 信号与 VCx_OUTA 信号极性相同，即正端大于负端时 VCx 输出高电平

22.3.5 数字滤波

电压比较器内置的数字滤波器，用于对电压比较器的输出信号进行数字滤波，由控制寄存器 VCx_CR1 的 FLTEN 位域控制，FLTEN 为 1 使能数字滤波，FLTEN 为 0 禁止数字滤波。用户可使用滤波功能过滤系统噪声，比如马达停止时的大电流噪声等，避免比较器的噪声输出引起系统的误动作。

数字滤波器的时钟由控制寄存器 VCx_CR1 的 FLTCLK 位域选择：

- FLTCLK 为 1，使用 PCLK 作为滤波时钟
- FLTCLK 为 0，使用内置 RC 振荡器时钟作为滤波时钟，其频率约 150kHz

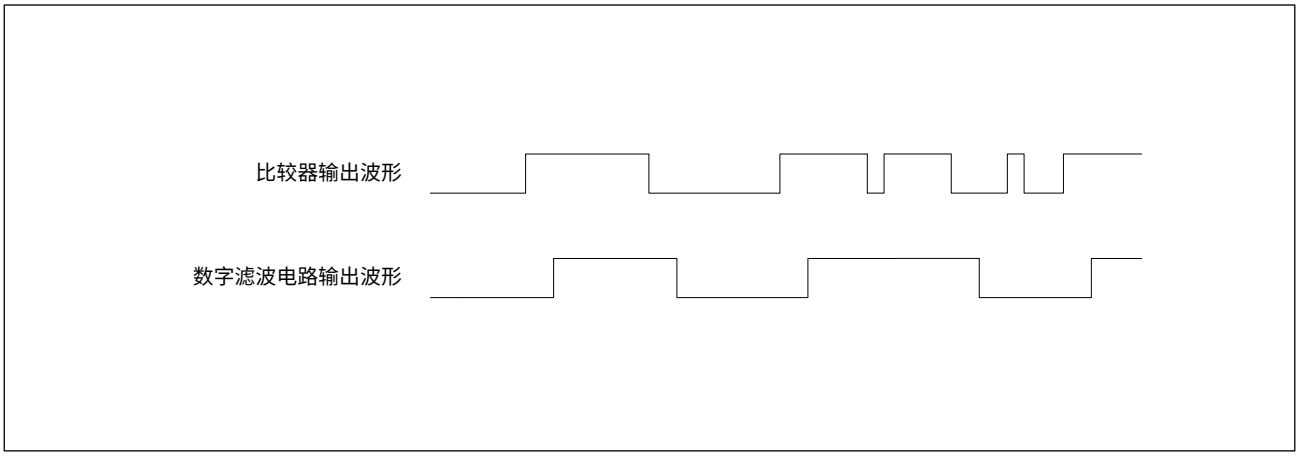
数字滤波器的滤波宽度由控制寄存器 VCx_CR1 的 FLTTIME 位域选择，滤除宽度小于一定时钟周期的信号，有 8 级滤波宽度可选择。

电压比较器经数字滤波后的输出电平，可通过状态寄存器 VCx_SR 的 FLTV 位域读出。

当用户设置了 VCx_OUT 引脚为数字输出，且选择 VC 比较输出的复用功能时，VCx_OUT 引脚将输出电压比较器经数字滤波后的输出电平。

VC 的滤波响应波形，如下图所示：

图 22-2 VC 滤波响应时间

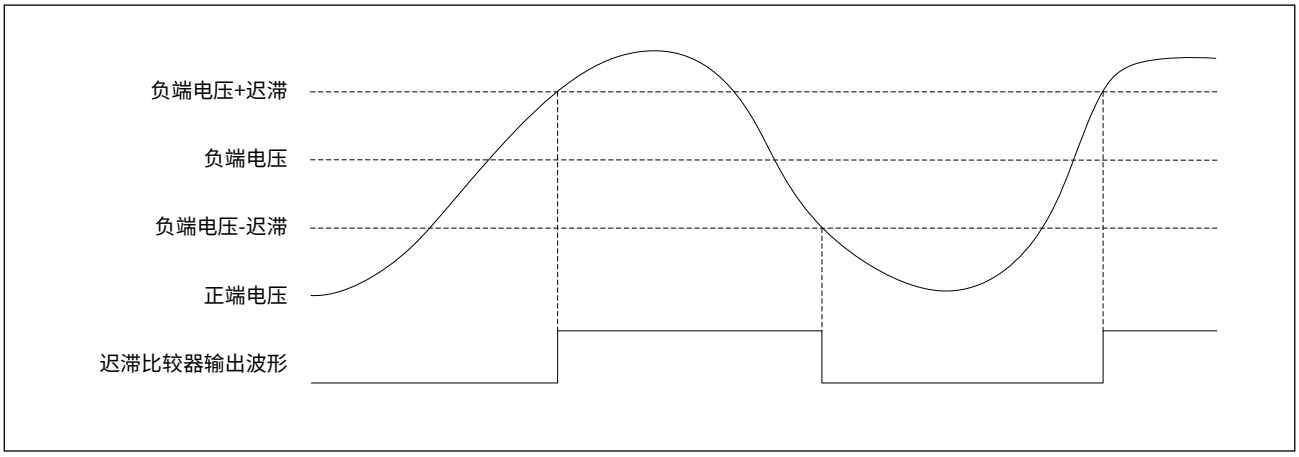


22.3.6 迟滞功能

模拟电压比较器支持迟滞功能，使用迟滞功能后，比较器的输出结果不会随输入信号的变化而立即翻转，而是在两路输入信号的偏移值高于或低于迟滞阈值电压后才发生翻转。

VC 迟滞功能波形示意图，如下图所示：

图 22-3 VC 迟滞功能



迟滞功能可以有效增强芯片的抗干扰能力，避免因输入信号的小幅度抖动，导致比较器的输出端不必要的频繁翻转。

迟滞阈值电压由控制寄存器 VCx_CR0 的 HYS 位域决定，如下表所示：

表 22-3 VC 迟滞阈值电压

VCx_CR0.HYS	迟滞窗口配置
00	没有迟滞
01	迟滞窗口大约 10mV
10	迟滞窗口大约 20mV
11	迟滞窗口大约 30mV

22.3.7 窗口比较功能

模拟电压比较器支持窗口比较功能，可将 VC1 和 VC2 的比较结果进行异或操作后输出，由控制寄存器 VCx_CR0 的 WINDOW 位域使能。

WINDOW 为 1 时，VCx_OUTW 信号为 VC1_OUTP 信号与 VC2_OUTP 信号的异或值；

WINDOW 为 0 时，VCx_OUTW 信号与 VCx_OUTP 信号电平相同。

22.4 VC 中断

CW32F020 的电压比较器支持在低功耗模式下工作，比较中断可将芯片从低功耗模式下唤醒。

设置控制寄存器 VCx_CR0 的 IE 位域为 1，使能 VCx 中断，产生中断时状态寄存器 VCx_SR 的中断标志位 INTF 会被硬件置 1，用户可以向 INTF 位写 0，清除中断标志。

设置控制寄存器 VCx_CR1 的 HIGHIE、RISEIE、FALLIE 位域，可选择不同的中断触发方式：

- HIGHIE 为 1，VCx_OUT 输出信号高电平触发中断
- RISEIE 为 1，VCx_OUT 输出信号上升沿触发中断
- FALLIE 为 1，VCx_OUT 输出信号下降沿触发中断

22.5 编程示例

在此示例中，使用数字滤波功能，并配置比较中断，配置方法如下所示：

步骤 1：配置 VCx_DIV.EN 为 1， 使能分压器；

步骤 2：配置 VCx_DIV.DIV，设置分压系数；

步骤 3：配置 VCx_CR0.INP ，选择正端待监测的电压来源；

步骤 4：配置 VCx_CR0.INN，选择负端待监测的电压来源；

步骤 5：配置 VCx_CR1.FLTTIME，选择滤波时间；

步骤 6：配置 VCx_CR1.FLTCLK，选择滤波时钟；

步骤 7：配置 VCx_CR1.FLTEN 为 1，使能 VCx 滤波；

步骤 8：设置 VCx_CR1 寄存器的 HIGHIE、RISEIE、FALLIE 为 1，选择中断触发方式；

步骤 9：设置 VCx_CR0.IE 为 1，使能 VCx 中断；

步骤 10：设置 VCx_CR0.EN 为 1，使能 VCx；

步骤 11：等待 VCx_SR.READY 标志位变为 1；

步骤 12：在 VC 模块的初始化程序和 VC 模块的中断服务程序中，对 VCx_SR 的 INTF 位写入 0，清除中断标志后，允许 VCx 中断的产生。

22.6 寄存器列表

VC1 基地址：VC1_BASE = 0x4001 2A00

VC2 基地址：VC2_BASE = 0x4001 2A10

表 22-4 VC 寄存器列表

寄存器名称	寄存器地址	寄存器描述
VCx_DIV	VCx_BASE + 0x00	电阻分压控制寄存器 [1]
VCx_CR0	VCx_BASE + 0x04	控制寄存器 0
VCx_CR1	VCx_BASE + 0x08	控制寄存器 1
VCx_SR	VCx_BASE + 0x0C	状态寄存器

注 1：

VC1_DIV 和 VC2_DIV 指向同一实体寄存器，VC1 和 VC2 共用电阻分压器电路。

22.7 寄存器描述

有关寄存器描述里所使用的缩写，请参见 1 文档约定章节。

22.7.1 VCx_DIV 电阻分压控制寄存器

Address offset: 0x00 Reset value: 0x0000 0000

位域	名称	权限	功能描述
31:8	RFU	-	保留位，请保持默认值
7	VIN	RW	电阻分压电路输入电压设置 0：VDDA 1：ADC 模块所配置的参考电压（需使能 ADC）
6	EN	RW	电阻分压电路使能控制 0：禁止 1：使能
5:0	DIV	RW	电阻分压电路输出电压配置 输出电压 = 输入电压 ×(1 + DIV) / 64 注：输入电压应大于 1.6V、小于等于 VCC。

注：

VC1_DIV 和 VC2_DIV 指向同一实体寄存器，VC1 和 VC2 共用电阻分压器电路。

22.7.2 VCx_CR0 控制寄存器 0

Address offset: 0x04 Reset value: 0x0000 0000

位域	名称	权限	功能描述
31:16	RFU	-	保留位，请保持默认值
15:12	INN	RW	VCx 负端输入信号配置 0000：VCx_CH0 0001：VCx_CH1 0010：VCx_CH2 0011：VCx_CH3 0100：VCx_CH4 0101：VCx_CH5 0110：VCx_CH6 0111：VCx_CH7 1000：内置分压器输出电压 1001：ADC 模块所配置的参考电压（注：需使能 ADC_CR0.EN） 1010：内置 1.2V 基准电压（注：需使能 ADC_CR0.BGREN） 1011：内置温度传感器输出电压（注：需使能 ADC_CR0.TSEN 和 ADC_CR0.BGREN） 注：配置 ADC_CR0 寄存器，需先使能 ADC 外设时钟
11:8	INP	RW	VCx 正端输入信号配置 0000：VCx_CH0 0001：VCx_CH1 0010：VCx_CH2 0011：VCx_CH3 0100：VCx_CH4 0101：VCx_CH5 0110：VCx_CH6 0111：VCx_CH7
7	WINDOW	RW	窗口比较功能配置 0：禁止窗口功能，VCx_OUTW 信号等于 VCx_OUTP 信号 1：使能窗口功能，VCx_OUTW 信号等于 VC1_OUTP 与 VC2_OUTP 的异或值
6	POL	RW	VCx 输出信号极性设置 0：正端大于负端时 VCx 输出高电平 1：正端大于负端时 VCx 输出低电平
5	IE	RW	中断使能配置 0：禁止 1：使能
4:3	HYS	RW	VCx 迟滞窗口配置 00：没有迟滞 01：迟滞窗口大约 10mV 10：迟滞窗口大约 20mV 11：迟滞窗口大约 30mV

位域	名称	权限	功能描述
2:1	RESP	RW	VCx 响应速度配置 00：极低速 01：低速 10：中速 11：高速 注：响应速度越快，功耗越大
0	EN	RW	VCx 使能控制 0：禁止 1：使能 注：使能 VCx 后需等待 VCx_SR.READY 标志位变为 1

22.7.3 VCx_CR1 控制寄存器 1

Address offset: 0x08 Reset value: 0x0000 0000

位域	名称	权限	功能描述
31:8	RFU	-	保留位，请保持默认值
7	HIGHIE	RW	VCx 输出信号高电平触发中断使能 0：禁止 1：使能
6	RISEIE	RW	VCx 输出信号上升沿触发中断使能 0：禁止 1：使能
5	FALLIE	RW	VCx 输出信号下降沿触发中断使能 0：禁止 1：使能
4	FLTCLK	RW	数字滤波模块滤波时钟设置 0：内置 RC 振荡器时钟，其频率约 150kHz 1：PCLK
3:1	FLTTIME	RW	数字滤波模块滤波时间配置 000：滤除宽度小于 1 个时钟周期的信号 001：滤除宽度小于 3 个时钟周期的信号 010：滤除宽度小于 7 个时钟周期的信号 011：滤除宽度小于 15 个时钟周期的信号 100：滤除宽度小于 63 个时钟周期的信号 101：滤除宽度小于 255 个时钟周期的信号 110：滤除宽度小于 1023 个时钟周期的信号 111：滤除宽度小于 4095 个时钟周期的信号
0	FLTEN	RW	数字滤波模块使能配置 0：禁止 1：使能

22.7.4 VCx_SR 状态寄存器

Address offset: 0x0C Reset value: 0x0000 0000

位域	名称	权限	功能描述
31:3	RFU	-	保留位，请保持默认值
2	READY	RO	VCx 状态标志，通过读取此位判断 VCx 是否已经稳定 0：尚未稳定，不可以被使用 1：已经稳定，可以被使用 注：若使能了 VCx，则进入深度休眠模式前必须查询等待该标志位置 1。
1	FLTV	RO	数字滤波器输出的电平值 0：数字滤波器输出低电平 1：数字滤波器输出高电平
0	INTF	RW0	中断标志 R0：未发生 VC 中断 R1：已发生 VC 中断 W0：清除 VC 中断标志 W1：无功能

23 低电压检测器（LVD）

23.1 概述

低电压检测器 (LVD) 用于监测 VDDA 电源电压或外部引脚输入电压，当被监测电压与 LVD 阈值的比较结果满足触发条件时，将产生 LVD 中断或复位信号，通常用于处理一些紧急任务。

LVD 产生的中断和复位标志，只能由软件清零；只有当中断或复位标志被清零后，在再次达到触发条件时，LVD 才能再次产生中断或者复位信号。

23.2 主要特性

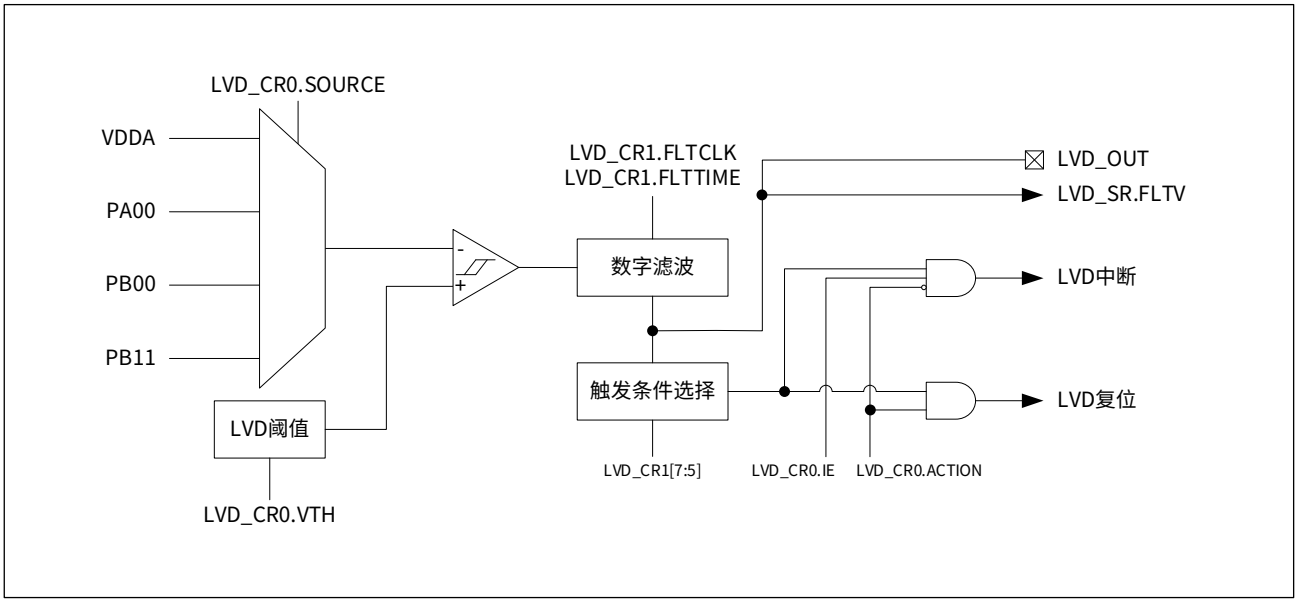
- 4 路监测电压源
  VDDA 电源电压，PA00、PB00、PB11 引脚输入。
- 16 阶阈值电压，范围 2V ~ 3.67V
- 3 种触发条件，可组合使用
  - 电平触发：电压低于阈值
  - 下降沿触发：电压跌落到阈值以下的下降沿
  - 上升沿触发：电压回升到阈值以上的上升沿
- 可触发产生中断或复位信号，二者不能同时产生
- 8 阶滤波可配置
- 支持迟滞功能
- 支持低功耗模式下运行，中断唤醒 MCU

23.3 功能描述

23.3.1 功能框图

低电压检测器 (LVD) 的功能框图如下图所示：

图 23-1 LVD 功能框图



LVD 可以监测 VDDA 电源电压，也可监测外部引脚输入电压（PA00、PB00、PB11），具体通过控制寄存器 LVD_CR0 的 SOURCE 位域来选择。当使用外部模拟信号输入时，用户必须将对应 GPIO 端口配置为模拟功能（GPIOx_ANALOG.PINy = 1）。

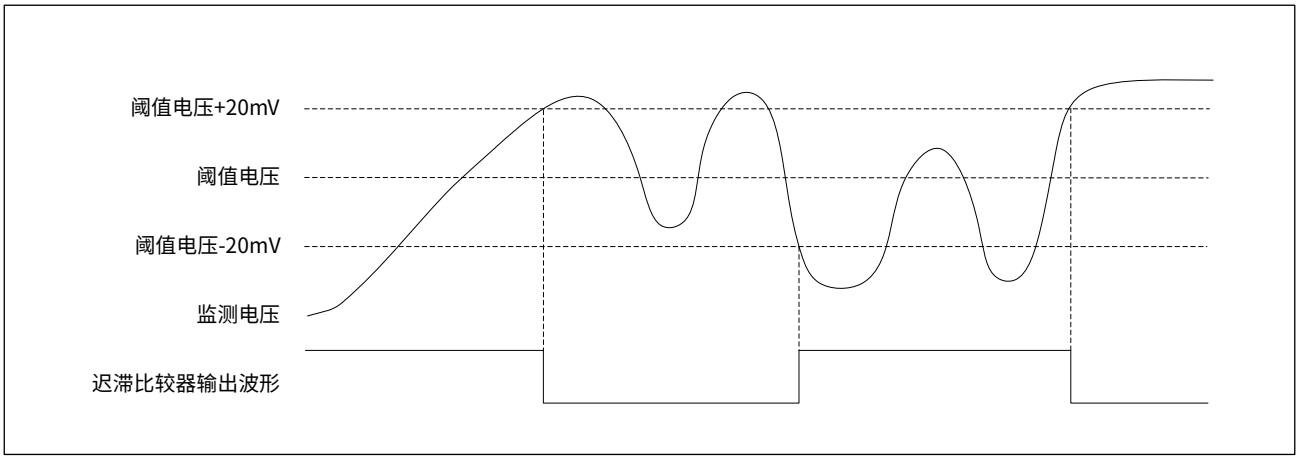
LVD 输出结果可以从 PA01 或 PA08 引脚输出，用户必须将对应 GPIO 端口配置为数字输出，同时选择功能复用。

23.3.2 迟滞功能

LVD 内置的电压比较器具有迟滞功能，可避免当 LVD 的被监测电压在阈值电压附近时，电压比较器的输出结果发生频繁翻转，增强系统抗干扰能力。

只有当被监测电压高于或低于阈值电压达到 20mV 时，比较器输出信号才会发生翻转。具体波形如下图所示：

图 23-2 LVD 迟滞响应



LVD 的阈值电压由控制寄存器 LVD_CR0 的 VTH 位值决定，有效值 0 ~ 15，如下表所示：

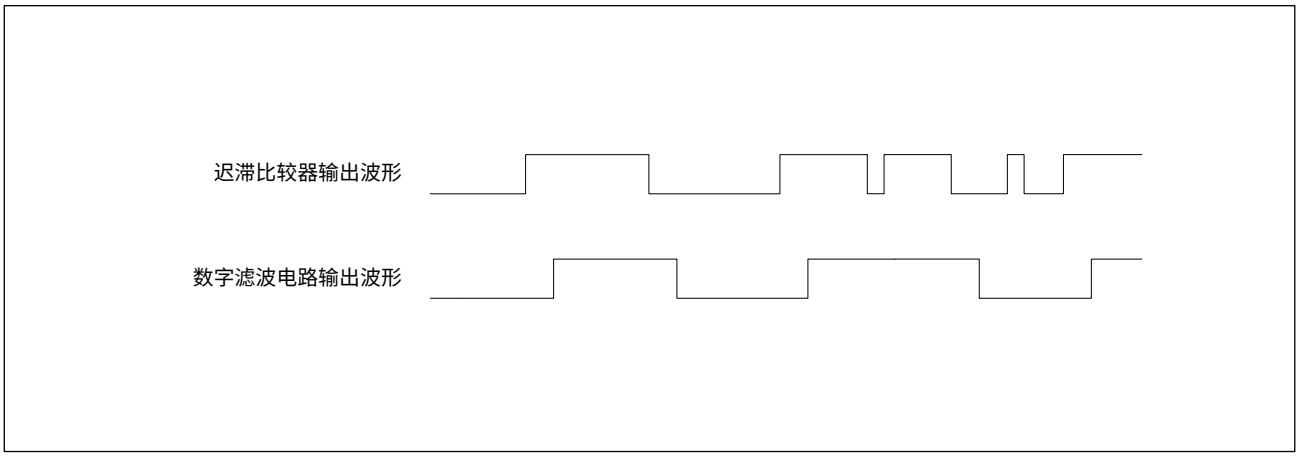
表 23-1 LVD 阈值电压

LVD_CR0.VTH	阈值电压（单位：V）	LVD_CR0.VTH	阈值电压（单位：V）
0000	2.00	1000	2.89
0001	2.11	1001	3.00
0010	2.22	1010	3.11
0011	2.33	1011	3.22
0100	2.44	1100	3.33
0101	2.56	1101	3.44
0110	2.67	1110	3.56
0111	2.78	1111	3.67

23.3.3 数字滤波

为增强系统的鲁棒性，LVD 支持数字滤波功能，可将 LVD 电压比较的输出结果信号进行数字滤波，小于滤波宽度的信号被滤除，不会触发中断或复位。具体波形如下图所示：

图 23-3 LVD 滤波输出



设置控制寄存器 LVD_CR1 的 FLTEN 位域为 1，使能数字滤波模块；

控制寄存器 LVD_CR1 的 FLTCLK 位域用于选择数字滤波的时钟：

- FLTCLK 位为 1，选择 HSIOSC 作为滤波时钟
- FLTCLK 位为 0，选择内置 RC 振荡器时钟作为滤波时钟，其频率约 150kHz

控制寄存器 LVD_CR1 的 FLTTIME 位域用于选择数字滤波的时钟个数，如下表所示：

表 23-2 数字滤波时钟个数

LVD_CR1.FLTTIME	滤波时钟个数
000	1
001	3
010	7
011	15
100	63
101	255
110	1023
111	4095

从 LVD 状态寄存器 LVD_SR 的 FLTV 位域，可以读出经 LVD 数字滤波后的信号电平；

当 GPIO 的功能复用为 LVD_OUT 时，数字滤波后的信号就可以从 GPIO 输出，以方便观察测量。

23.4 LVD 中断

LVD 支持在低功耗模式下工作，中断输出可将芯片从低功耗模式下唤醒。

当被监测电压与 LVD 阈值的比较结果满足触发条件时，可产生中断或复位信号。产生中断还是复位信号由控制寄存器 LVD_CR0 的 ACTION 位域控制：

- ACTION 为 1，LVD 触发产生复位
- ACTION 为 0，LVD 触发产生中断

设置控制寄存器 LVD_CR0 的 IE 位域为 1，使能 LVD 中断，满足触发条件时将产生 LVD 中断，中断标志位 LVD_SR.INTF 会被硬件置 1，用户可以向 INTF 位写 0，清除中断标志。

设置控制寄存器 LVD_CR1 的 LEVEL、FALL、RISE 位域，可选择不同的中断或复位触发方式，三者可组合使用：

- LEVEL 为 1，被监测电压低于阈值时触发中断或产生复位
- FALL 为 1，被监测电压跌落到阈值以下的下降沿触发中断或产生复位
- RISE 为 1，被监测电压回升到阈值以上的上升沿触发中断或产生复位

23.5 编程示例

23.5.1 欠压复位编程示例

在此示例中，被监测电压低于阈值电压时复位 MCU，使用数字滤波功能。

配置方法如下所示：

步骤 1：配置 LVD_CR0.SOURCE，选择待监测的电压来源；

步骤 2：配置 LVD_CR0.VTH，设置阈值电压；

步骤 3：配置 LVD_CR1.FLTTIME，选择 LVD 滤波时间；

步骤 4：配置 LVD_CR1.FLTCLK，选择滤波时钟；

步骤 5：配置 LVD_CR1.FLTEN，使能 LVD 滤波；

步骤 6：设置 LVD_CR1.LEVEL 为 1，选择被监测电压低于阈值时触发 LVD 动作；

步骤 7：设置 LVD_CR0.ACTION 为 1，选择 LVD 触发动作为系统复位；

步骤 8：设置 LVD_CR0.EN 为 1，使能 LVD。

23.5.2 中断编程示例

在此示例中，被监测电压高于或低于阈值电压时产生中断，使用数字滤波功能。

配置方法如下所示：

步骤 1：配置 LVD_CR0.SOURCE，选择待监测的电压来源；

步骤 2：配置 LVD_CR0.VTH，选择阈值电压；

步骤 3：配置 LVD_CR1.FLTTIME，选择 LVD 滤波时间；

步骤 4：配置 LVD_CR1.FLTCLK，选择滤波时钟；

步骤 5：配置 LVD_CR1.FLTEN，使能 LVD 滤波；

步骤 6：设置 LVD_CR1.RISE 和 FALL 均为 1，选择上升沿和下降沿触发；

步骤 7：设置 LVD_CR0.ACTION 为 0，选择 LVD 触发动作为中断；

步骤 8：设置 LVD_CR0.IE 为 1，使能 LVD 中断；

步骤 9：使能 NVIC 中断向量表中的 LVD 中断；

步骤 10：设置 LVD_CR0.EN 为 1，使能 LVD；

步骤 11：分别在 LVD 初始化程序和 LVD 中断服务程序中，对 LVD_ISR.INTF 位写入 0，以清除中断标志，允许产生新的 LVD 中断。

23.6 寄存器列表

LVD 基地址：LVD_BASE = 0x4001 2A80

表 23-3 LVD 寄存器列表

寄存器名称	寄存器地址	寄存器描述
LVD_CR0	LVD_BASE + 0x00	控制寄存器 0
LVD_CR1	LVD_BASE + 0x04	控制寄存器 1
LVD_SR	LVD_BASE + 0x08	状态寄存器

23.7 寄存器描述

有关寄存器描述里所使用的缩写，请参见 1 文档约定章节。

23.7.1 LVD_CR0 控制寄存器 0

Address offset: 0x00 Reset value: 0x0000 0000

位域	名称	权限	功能描述
31:10	RFU	-	保留位，请保持默认值
9	IE	RW	中断使能控制 0：禁止 LVD 中断 1：使能 LVD 中断
8	RFU	-	保留位，请保持默认值
7:4	VTH	RW	阈值电压选择 0000：2.00V 1000：2.89V 0001：2.11V 1001：3.00V 0010：2.22V 1010：3.11V 0011：2.33V 1011：3.22V 0100：2.44V 1100：3.33V 0101：2.56V 1101：3.44V 0110：2.67V 1110：3.56V 0111：2.78V 1111：3.67V
3:2	SOURCE	RW	监测来源配置 00：VDDA 电源电压 01：PA00 端口输入电压 10：PB00 端口输入电压 11：PB11 端口输入电压
1	ACTION	RW	触发动作配置 0：NVIC 中断 1：系统复位
0	EN	RW	使能控制 0：禁止 LVD 1：使能 LVD

23.7.2 LVD_CR1 控制寄存器 1

Address offset: 0x04 Reset value: 0x0000 0000

位域	名称	权限	功能描述
31:8	RFU	-	保留位，请保持默认值
7	LEVEL	RW	被监测电压低于阈值电平触发 （即 LVD 输出信号高电平触发） 0：禁止 1：使能
6	FALL	RW	被监测电压跌落到阈值以下的下降沿触发 （即 LVD 输出信号上升沿触发） 0：禁止 1：使能
5	RISE	RW	被监测电压回升到阈值以上的上升沿触发 （即 LVD 输出信号下降沿触发） 0：禁止 1：使能
4	FLTCLK	RW	数字滤波模块滤波时钟设置 0：内置 RC 振荡器时钟，其频率约 150kHz 1：HSIOSC
3:1	FLTTIME	RW	数字滤波模块滤波时间配置 000：滤除宽度小于 1 个时钟周期的信号 001：滤除宽度小于 3 个时钟周期的信号 010：滤除宽度小于 7 个时钟周期的信号 011：滤除宽度小于 15 个时钟周期的信号 100：滤除宽度小于 63 个时钟周期的信号 101：滤除宽度小于 255 个时钟周期的信号 110：滤除宽度小于 1023 个时钟周期的信号 111：滤除宽度小于 4095 个时钟周期的信号
0	FLTEN	RW	数字滤波模块使能配置 0：禁止 1：使能

23.7.3 LVD_SR 状态寄存器

Address offset: 0x08 Reset value: 0x0000 0000

位域	名称	权限	功能描述
31:2	RFU	-	保留位，请保持默认值
1	FLTV	RO	数字滤波器输出的电平值 0：数字滤波器输出低电平 1：数字滤波器输出高电平
0	INTF	RW0	中断标志 R0：未发生 LVD 中断 R1：已发生 LVD 中断 W0：清除 LVD 中断标志 W1：无功能

24 调试接口（DBG）

24.1 概述

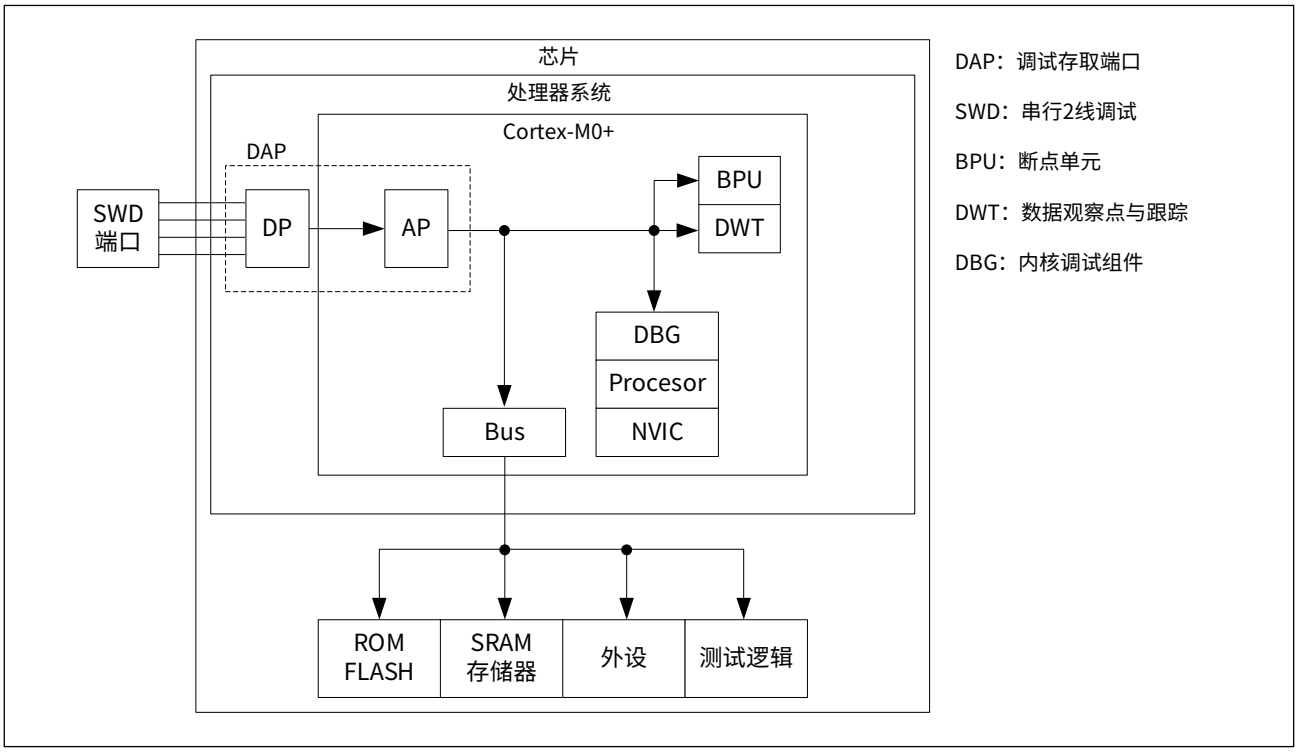
CW32F020 的内核为 ARM® Cortex®-M0+，内核内置 DAP 硬件调试模块，支持 SWD 模式调试。硬件调试模块可实现在取指 (指令断点) 或访问数据 (数据断点) 时挂起，程序停止运行，调试器可通过 DAP 对 M0 的内核状态和片内的外设状态及存储单元进行查询；且内核和外设可以被复原，程序继续执行。

当使用调试仿真工具通过 SWD 接口连接到 CW32F020，进入调试模式，通过芯片内核中的 DAP 硬件调试模块进行调试操作。

24.2 串行线调试端口 SWD

使用 CW 专用调试器或通用调试仿真工具的 SWD 接口和目标芯片内部的 DAP 调试模块连接，通过包传输协议进行数据交换，实现调试操作。SWD 方式是 2 线串行通信，包括一条时钟线 SWCLK 和一条双向数据线 SWDIO。如下图所示：

图 24-1 SWD 功能框图



CW32F020 系列的 SWD 接口引脚分配如下表所示，PA13/PA14 引脚在芯片出厂时默认为 SWD 功能。

表 24-1 SWD 引脚分配

SWD 端口名称	引脚功能	引脚分配
SWCLK	串行时钟输入	PA14
SWDIO	串行数据输入输出	PA13

PA13/PA14 引脚可配置为 SWD 功能或 GPIO/ISP 功能，由系统控制寄存器 SYSCTRL_CR2 的 SWDIO 位域进行功能配置。SWDIO 为 0，PA13/PA14 引脚被配置为 SWD 功能；SWDIO 为 1，则被配置为 GPIO/ISP 功能。

通常建议 SWD 引脚使用 100KΩ 上拉电阻，CW32F020 的 PA13/PA14 作为 SWD 功能时内置有上拉电阻，其阻值在 50kΩ ~ 200kΩ 之间，用户可以在外部增加上拉电阻，以提高抗干扰性能。

CW32F020 系列的 PA13/PA14 引脚默认为 SWD 功能，如果用户设定了加密等级，则需要根据设定的等级来判别是否支持 SWD 功能，SWD 引脚的配置与功能如下表所示：

表 24-2 SWD 引脚功能

加密等级	SYSCTRL_CR2.SWDIO	PA13/PA14 功能
Level0/1 加密	0（默认）	SWD 功能
	1	GPIO/ISP 功能
Level2 加密	0（默认）	NA，无任何功能
	1	GPIO/ISP 功能
Level3 加密	0（默认）	NA，无任何功能
	1	GPIO 功能

注意事项：

1. 加密等级可通过 ISP 通信方式进行设定，详情请参阅 ISP 通信协议文档；

2. 芯片上电后 SWDIO 和 SWCLK 均默认为内部上拉，用户可不用外接上拉电阻；

3. 当芯片加密等级设定为 2 时，SWD 功能被禁止，只能通过 ISP 方式烧录；

4. 当芯片加密等级设定为 3 时，SWD 功能及 ISP 功能均被禁止，芯片无法再次烧录新程序。

24.3 调试通信协议

调试器和目标芯片的 DAP 调试模块通过 SWD 包传输协议进行通信，包传输协议为 2 线同步串行协议，使用 SWCLK 时钟信号和 SWDIO 数据信号：

- SWCLK 为单向时钟信号，由调试器输出给目标芯片
- SWDIO 为双向数据信号，由调试器和目标芯片双向分时驱动

协议定义了长度为一个 SWCLK 周期的收发端转换时间，在收发端转换时间内，调试器和目标芯片都不驱动 SWDIO，SWDIO 由上拉电阻上拉到高电平。

SWDIO 信号线上传输数据时，遵循最低位 LSB 最先传输，最高位 MSB 最后传输原则。

通过包传输协议，调试器可以对目标芯片 DAP 调试模块内的 DP 寄存器和 AP 寄存器进行读写访问，下文中写作 SW-DP 和 SW-AP。

24.3.1 传输协议格式

DAP 调试模块包含有 DP 调试端口寄存器和 AP 存取端口寄存器，调试器和目标芯片的 DAP 调试模块进行通信实际上就是对 DP 寄存器和 AP 寄存器的读写操作。

传输通信帧通常包含 3 个字段：

- 包请求
  长度为 8bit，调试器到目标芯片
- 响应
  长度为 3bit；目标芯片到调试器
- 数据传输
  长度为 33bit，目标芯片到调试器或者调试器到目标芯片，可选字段

包请求各位的功能定义如下：

表 24-3 包请求位定义

位	名称	说明
0	启动	必须为 1
1	APnDP	0：DP 访问；1：AP 访问
2	RnW	0：写请求；1：读请求
3:4	A[3:2]	DP 或 AP 寄存器的地址字段，bit3 为 A2，bit4 为 A3
5	奇偶校验	前面 5bit 的 bit 奇偶校验位
6	停止	0
7	驻留	不受主机驱动。由于存在上拉，目标芯片读出为 1

请求包后面始终为收发端转换时间（默认 1bit），此时主机和目标都不驱动 SWDIO。

目标发送的响应位定义如下：

表 24-4 目标发送响应位定义

位	名称	说明
0:2	ACK	001：FAULT（bit0:0, bit1:0, bit2:1） 010：WAIT （bit0:0, bit1:1, bit2:0） 100：OK （bit0:1, bit1:0, bit2:0）

ACK 回应后为收发端转换时间（默认 1bit），此时主机和目标都不会驱动 SWDIO。

调试器或目标芯片发送的数据，位定义如下：

表 24-5 数据传输位定义

位	名称	说明
0:31	WDATA 或 RDATA	写入或读取数据
32	奇偶校验	32bit 数据的 bit 奇偶校验位

数据传输阶段不是必须的，如下两种情况才有数据传输：

- 数据读或者写请求后收到的回应是 OK 响应
- DP-CTRL/STAT 寄存器的 ORUNDETECT 标志位被置位 1，此时无论收到何种回应（包括 WAIT 和 FAULT）都要求必须有数据传输

24.3.2 SW-DP 状态机

SW-DP 内部有一个 ID CODE 寄存器，固定值为 0x0BB11477（为 ARM 公司 Cortex®-M0+ 识别码）。在采用 SWD 包协议进行目标寄存器读写之前，必须先读取此 ID 号以激活目标芯片的 SW-DP 逻辑，否则目标芯片的 SW-DP 的状态机不工作。

操作步骤如下：

步骤 1：在上电复位后或者 SWDIO 线路处于高电平超过 50 个周期后，SW-DP 状态机进入复位状态；

步骤 2：进入复位状态后，保持 SWDIO 线路处于低电平至少 2 个周期，SW-DP 状态机进入空闲状态；

步骤 3：进入空闲状态后，对 DP-SW 的 ID CODE 寄存器执行读访问；

注：

如果不执行此操作，则目标板在后续的包通信响应阶段，会发送 FAULT 响应。

步骤 4：按照包协议对需要访问的寄存器进行读写访问。

24.3.3 SW-DP 和 SW-AP 的读写访问

- SW-DP/SW-DP 的写访问

  调试器接收到 ACK 后，经过 1 个时钟周期的转换时间后，必须立即发送数据。

- SW-DP 的读访问
  - 当目标芯片已做好数据发送准备，则发送 OK 响应，然后立即发送数据；
  - 当目标芯片 DAP 未做好数据发送准备，则发送 WAIT 响应结束本次通信；
  - 当目标芯片 DAP 状态错误时，发送 FAULT 响应，结束本次通信。
- SW-AP 的读访问

  对 AP 的读访问为寄存式，即本次读 AP 操作并不能返回所需要的结果，而是等到下次 AP 操作才能返回本次读操作的结果。如果要执行的下次访问不是 AP 读访问操作，则必须读取 DP-RDBUFF 寄存器来获取本次读操作的结果。

- 每次进行 SW-AP 读访问或 DP-RDBUFF 读请求时都会更新 M0 内核的调试控制 / 状态寄存器 DP-CTRL/STAT 寄存器的 READOK 标志位，以指示 AP 读访问是否成功。
- SW-DP 有写缓冲区（用于 SW-DP 或 SW-AP 写入），在读写操作未完成时，可以接受下一个写入操作。如果 SW-DP 的写缓冲区已满，则芯片 SW-DP 逻辑会回复 WAIT 响应以通知主机暂缓操作。特殊操作除外，如 IDCODE 读取、DP-CTRL/STAT 读取或 ABORT 写入操作，这几项操作在写缓冲区已满时也会被接受。
- 由于 SWCLK 和 HCLK 不是同步时钟，属于异步时钟域，因此写操作后（数据传输的奇偶校验位后）还需要两个额外的 SWCLK 周期，以保证写入操作生效。在主机驱动这 2 个 SWCLK 周期时应将 SWDIO 线路驱动为低电平（空闲状态）。在对 DP-CTRL/STAT 寄存器写入上电请求操作时，这一点特别重要，否则下一个操作（在内核上电后才有效的操作）会立即执行，将导致执行失败。

24.3.4 SW-DP 寄存器

当 APnDP 为 0 时，访问 DP 寄存器，由 A[3:2] 寻址，地址相同时因读写操作不同，寄存器含义可能不相同，具体如下表所示：

表 24-6 DP 寄存器列表及定义

A[3:2]	读 / 写	寄存器名称	寄存器内容说明
00	读	IDCODE	固定为 0x0BB1 1477（用于标识 SW-DP）， ARM 的 Cortex® -M0+ 的识别码
	写	ABORT	见表 24-7 DP ABORT 寄存器位定义
01	读 / 写	DP-CTRL/STAT (SELECT.CTRLSEL = 0)	主要用于： 1、请求系统或调试上电； 2、配置 AP 访问的传输操作； 3、控制比较和验证操作； 4、读取一些状态标志（上溢和上电确认）
		WIRE CONTROL (SELECT.CTRLSEL = 1)	用于配置物理串行端口协议（如转换时间的持续时间等）
10	读	READ RESEND	允许从已损坏的调试传输中恢复读取数据，无需重复执行原始 AP 传输。
	写	SELECT	用于选择当前的访问 AP 端口和以及 AP 端口内 4 字寄存器 BANK。 见表 24-8 DP SELECT 寄存器
11	读 / 写	READ BUFFER	由于已发出 AP 访问，因此该读缓冲区非常有用（在执行下个 AP 事务时提供读取本次 AP 请求的结果）。此读取缓冲区捕获 AP 中的数据，显示为前一次读取的结果，无需启动新操作。

表 24-7 DP ABORT 寄存器位定义

位	名称	定义
31:5	保留	
4	ORUNERRCLR	写 1 清除 STICKYORUN 错误标志
3	WDATAERR	写 1 清除 WDATAERR 错误标志
2	STICKYERR	写 1 清除 STICKYERR 错误标志
1	STICKYCMP	写 1 清除 STICKYCMP 标志。
0	DAPABORT	写 1 产生 DAP ABORT 信号，放弃当前存取操作；当目标芯片回应 WAIT 响应时必须执行此操作来中止此次操作。

表 24-8 DP SELECT 寄存器

位	名称	定义
31:24	APSEL	选择当前 AP 端口，固定为 b00000000
23:8	保留	
7:4	APBANKSEL	在当前 AP 上选择活动的 4 字寄存器 BANK
3:1	保留	
0	CTRLSEL	DP 端口寄存器选择： 复位后默认为 0，选择 CTRL/STAT 寄存器； 设置为 1，选择 WCR 寄存器（Wire Control Register）

24.3.5 SW-AP 寄存器

当 APnDP 为 1 时，访问 AP 寄存器。由于 SW-AP 寄存器较多，需要将 A[3:2] 和 SW-DP 选择寄存器 SELECT 的 APBANKSEL 位域值组合才能对 SW-AP 寄存器进行唯一寻址。

本文不对 SW-AP 逐一赘述，仅介绍一个常用的 IDR 公共寄存器（地址为 0xFC），如下表所示：

表 24-9 AP IDR 公共寄存器

位	定义
31:28	AP 设计的版本 Rervision。
27:24	AP 设计者标识码，本芯片固定为 0x04。
23:17	AP 设计者标识码，本芯片固定为 0x3B。
16:13	AP 类型。如存储器存取端口类型标识码为 b1000，未定义类型存取端口类型标识码为 b0000。
12:8	保留
7:0	AP 识别码。如 MEM-AP 或者 JTAG-AP 等。

24.4 内核调试

通过操作内核调试寄存器可实现对内核的调试，调试器通过 DAP 调试访问这些寄存器。

内核调试寄存器主要包括 4 个寄存器，如下表所示：

表 24-10 内核调试寄存器

寄存器	说明
DHCSR	调试暂停控制和状态寄存器，32bit， 控制处理器的暂停、单步和重启等动作
DCRSR	调试内核寄存器选择器寄存器，17bit， 在暂停期间控制对内核寄存器的读和写
DCRDR	调试内核寄存器数据寄存器，32bit， 暂停期间读写内核内核寄存器的数据传输寄存器
DEMCR	调试异常监视控制寄存器，32bit， 用于使能数据监视点单元和向量捕捉特性，利用向量捕捉，调试器可以在处理器复位或者硬件错误产生时暂停处理器

这些寄存器只能通过上电复位来复位。更多详细信息请参阅《Cortex®-M0+ Technical Reference Manual》。

24.5 断点单元 BPU

Cortex®-M0+ BPU 断点单元提供四个断点寄存器，实现了基于 PC 指针的断点功能。

有关 BPU CoreSight 组件的标识寄存器和访问类型的更多信息，请参阅《ARMv6-M Architecture Reference Manual》和 ARM® CoreSight 组件技术参考手册。

24.6 数据观察点与跟踪 DWT

Cortex®-M0+ DWT 提供了两个观察点寄存器组。实现如下功能：

- 设置数据监视点
  数据或者外设的地址可以被标记为监视变量，对该地址的访问会产生调试事件，会暂停程序执行。
- ARMv6-M 中可选的 DWT 程序计数器采样寄存器 (DWT_PCSR) 功能。允许调试程序定期采样 PC 指针，提供粗略分析，无需停止处理器。

有关更多信息，请参阅《ARMv6-M Architecture Reference Manual》。

24.7 调试组件 DBG

调试器通过调试组件 DBG 实现断点期间的定时器、看门狗、RTC 等外设的时钟控制支持。通过调试状态定时器控制寄存器 SYSCTRL_DEBUG 设置，可控制定时器、看门狗定时器、RTC 等在调试状态下断点期间正常运行或暂停，详见下表：

表 24-11 调试状态定时器控制寄存器 SYSCTRL_DEBUG

位	名称	值	调试状态下断点期间计数器功能配置
10	WWDT	1	调试状态下， WWDT 计数器暂停计数
		0	调试状态下， WWDT 计数器正常计数
9	IWDT	1	调试状态下， IWDT 计数器暂停计数
		0	调试状态下， IWDT 计数器正常计数
8	RTC	1	调试状态下， RTC 计数器暂停计数
		0	调试状态下， RTC 计数器正常计数
6	AWT	1	调试状态下， AWT 计数器暂停计数
		0	调试状态下， AWT 计数器正常计数
5	BTIM123	1	调试状态下， BTIM1、BTIM 2、BTIM 3 计数器暂停计数
		0	调试状态下， BTIM1、BTIM 2、BTIM 3 计数器正常计数
4	GTIM4	1	调试状态下， GTIM4 计数器暂停计数
		0	调试状态下， GTIM4 计数器正常计数
3	GTIM3	1	调试状态下， GTIM3 计数器暂停计数
		0	调试状态下， GTIM3 计数器正常计数
2	GTIM2	1	调试状态下， GTIM2 计数器暂停计数
		0	调试状态下， GTIM2 计数器正常计数
1	GTIM1	1	调试状态下， GTIM1 计数器暂停计数
		0	调试状态下， GTIM1 计数器正常计数

如当定时器输出 PWM 进行电机控制时，定时器不能停止，否则会造成电机损坏。又如，看门狗定时器在断点期间要停止计数，以防止系统发生不希望的复位。

24.8 注意事项

CW32F020 在调试期间需要使用 HCLK 进行调试连接，不允许在调试会话期间关闭 HCLK，因此在调试环境下执行 WFI，会关闭内核 HCLK，导致调试功能失效。

如目标芯片当前已经进入深度休眠模式，此时 DAP 硬件调试模块不工作，调试器无法和目标芯片的 DAP 硬件调试模块进行连接，无法实现调试功能。

用户必须保证芯片处于运行模式（Active mode）或休眠模式（Sleep mode）才能使用调试器进行调试。

因此建议用户在合并深度休眠的相关代码前，先完成非深度休眠模式下设备的所有功能调试，最后再进行深度休眠相关代码的调试。

25 数字签名

25.1 概述

数字签名主要用来存放芯片唯一身份标识（UID）、产品型号、FLASH 容量、SRAM 容量、芯片封装引脚数等信息，可以通过 SWD 或者 CPU 读取。数字签名相关信息在出厂时编程，用户固件或外部设备可通过读取数字签名来对芯片的合法性进行验证。

25.2 产品唯一身份标识（UID）寄存器（80bit）

UID 寄存器存储了芯片的唯一身份标识符，其地址为 0x0001 2660 - 0x0001 2669，共 80bit。UID 在芯片生产时写入，用户无法修改。UID 寄存器支持以单字节 / 半字 / 全字等方式读取，然后使用自定义算法连接起来。

唯一身份标识符典型应用场景：

- 用作设备序列号
- 设备合法性验证 , 防止盗版

  用户在设备生产时采用私有密钥对 UID 进行加密运算，并将计算结果存放在主 FLASH 存储器或 OTP 存储器，程序在设备启动后，读取 UID 并采用同样的密钥进行加密运算，并将运算结果和之前存储的计算结果进行比较，相同则认为该设备是合法的，否则程序不启动，可有效防止用户设备被非法复制（盗版）。
- 作为安全密钥使用

  用户结合 UID 和私有算法，可在用户对 FLASH 编程前进行安全校验，提高 FLASH 内代码的安全性。
- 激活安全启动流程等

25.3 产品型号寄存器

产品型号寄存器存储了产品型号的 ASCII 码，其地址为 0x0001 2610 - 0x0001 2625，共 22 字节。产品型号不足 22 字节时，用 0x00 进行填充。

如芯片的型号为 CW32F020C6U7-QFN48，对应存储 (从低地址开始) 的数据为：0x43 0x57 0x33 0x32 0x46 0x30 0x32 0x30 0x43 0x36 0x55 0x37 0x2D 0x51 0x46 0x4E 0x34 0x38 0x00 0x00 0x00 0x00。

注：

产品型号中如果有字母，则字母需要大写。

25.4 FLASH 容量寄存器

FLASH 容量寄存器存储了芯片内置 FLASH 存储器的容量大小，其地址为 0x0001 2628 - 0x0001 262B，共 4 字节。

从 FLASH 容量寄存器读出的 FLASH 容量大小以字节为单位，如 0x0001 0000 代表 64KB，0x0000 8000 代表 32KB。

25.5 SRAM 容量寄存器

SRAM 容量寄存器存储了芯片内置 SRAM 存储器的容量大小，其地址为 0x0001 262C - 0x0001 262F，共 4 字节。

从 SRAM 容量寄存器读出的 SRAM 容量大小以字节为单位，如 0x0001 0000 代表 64KB，0x0000 4000 代表 16KB。

25.6 引脚数量寄存器

引脚数量寄存器存储了芯片引脚数，其地址为 0x0001 2626 - 0x0001 2627，共 2 字节。如 0x0020 代表 32Pin，0x0030 代表 48Pin。

26 版本信息

表 26-1 文档修订信息

日期	版本	变更信息
2022-07-05	Rev 1.0	初始发布