32 位 MCU
ES32M0150

参 考 手 册

☐ 产品简介

☐ 数据手册

☑ 参考手册

上海东软载波微电子有限公司

2022-08-01

目 录

图目录

表目录

第1章 文档约定

1.1 寄存器读写权限的设定

缩写词	说明	描述
R/W	读/写（__IO）	软件可以读写这些位
R	只读（__I）	软件只能读取这些位
W	只写（__O）	软件只能写入该位，读取该位时将返回复位值
W1	只写（写 1）	软件只能写入该位，写 1 有效，写 0 无作用。
R/C_W1	读取/清零（写 1）	软件可以读取该位，也可以通过写入 1 将该位清零。写入“0”对该位的值无影响
R/C_W0	读取/清零（写 0）	软件可以读取该位，也可以通过写入 0 将该位清零。写入“1”对该位的值无影响
R/C_R	读取/清零（读取）	软件可以读取该位。读取该位时，将自动清零。写入“0”对该位的值无影响
C_W1	清零（写 1）	通过写入 1 将该位清零。写入“0”对该位的值无影响
S_W1	置位（写 1）	通过写入 1 将该位置位。写入“0”对该位的值无影响
C_W0	清零（写 0）	通过写入 0 将该位清零。写入“1”对该位的值无影响
T_W1	触发（写 1）	通过写入 1 将触发硬件动作。写入“0”对该位的值无影响
Reserved	保留	保留位，必须保持复位值。

第2章 系统概述

2.1 概述

本章节从系统层介绍 ES32M0150 系列 MCU 所涵盖的功能模块。

2.2 系统框图

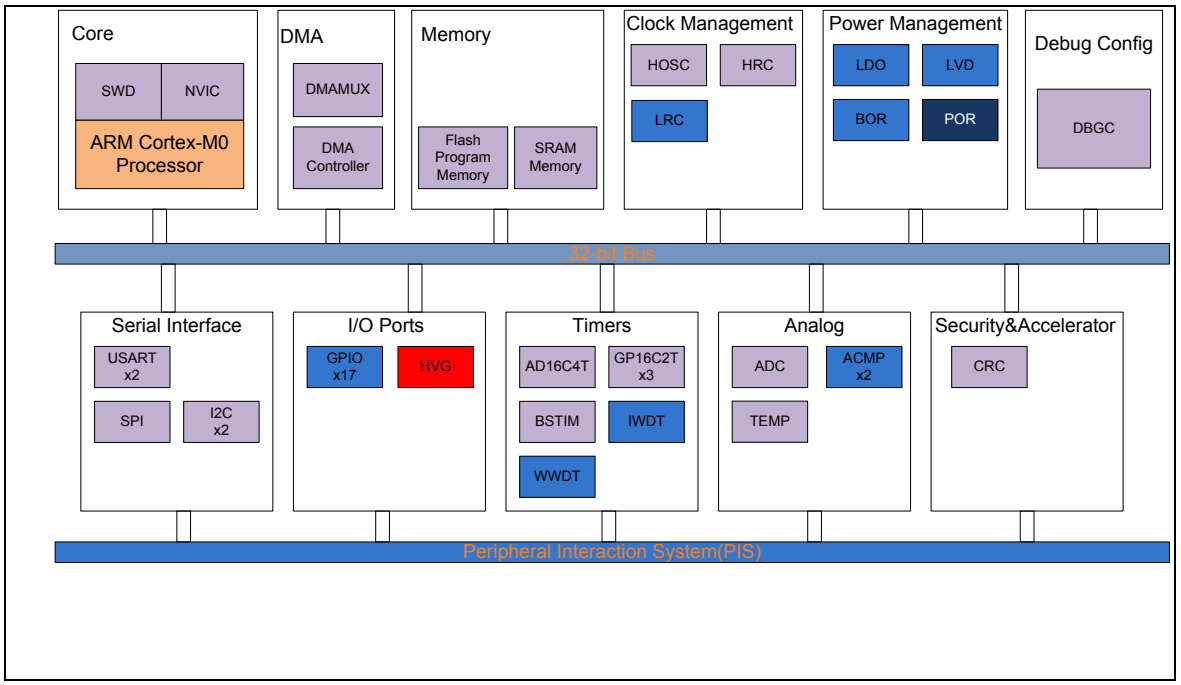


图 2-1 系统框图

2.3 模块功能类别

类别	描述
ARM 32 位 Cortex-M0 CPU	◆ ARM Cortex M 系列 32 位 MCU 内核，最高系统时钟频率可达 52 MHz ◆ 调试 ◆ 中断和事件 ◆ ES32M0150 不支持 CPU 休眠模式
存储	◆ 内部存储： ◇ FLASH 存储器 ◇ SRAM 存储器 ◆ 系统总线和存储器 ◆ 存储器系统控制
系统管理	◆ 系统配置控制器 ◆ 复位控制 ◆ DMA ◇ 支持多个 DMA 请求通道，DMAMUX 为每个 DMA 通道提供片上 DMA 请求源选择 ◆ 外设互联（PIS） ◆ 看门狗定时器 ◇ 独立看门狗定时器 ◇ 窗口看门狗定时器 ◆ 调试配置控制器（DBGC）
时钟管理	◆ 提供外部和内部多种时钟源选择 ◇ HOSC ◇ HRC ◇ LRC ◆ 外部时钟停振检测 ◆ 可灵活选择内核，系统及外设时钟 ◆ 可灵活设置外设时钟门控及时钟分频，以满足低功耗应用需求
外部接口	◆ 通用 IO(GPIO)
安全管理及运算加速	◆ 循环冗余校验模块（CRC）
定时器	◆ 通用定时器（GP16C2T0~2） ◆ 基本定时器（BS16T） ◆ 通用定时器（AD16C4T）
通信	◆ I2C 总线接口（I2C0~1） ◆ 串行外设接口（SPI） ◆ 通用同步异步收发器（USART0~1）
模拟	◆ 模数转换（ADC） ◆ 模拟比较器（ACMP） ◆ 温感（TS）

表 2-1 系统功能模块

2. 3. 1 ARM 32 位Cortex-M0 内核模块

ES32M0150 微控制器内核模块包含以下功能：

模块	描述
ARM 32 位 Cortex-M0 内核	◆ 支持 Thumb 指令集； ◆ 32 位硬件乘法器； ◆ 高效可靠的中断响应； ◆ **不支持使用 WFI,WFE 进入低功耗模式**
NVIC	◆ 中断使能控制； ◆ 中断优先级设置； ◆ 支持末尾连锁和迟来； ◆ 支持 32 个外部中断向量
调试接口	◆ SWD 协议调试接口

表 2-2 ARM 32 位 Cortex-M0 内核模块

2.3.2 存储器及存储器接口

ES32M0150 微控制器包含以下存储器及存储器接口模块：

模块	描述
系统总线和存储器	◆ 系统总线架构，存储器地址空间映射
存储器系统控制	◆ FLASH 存储器的访问控制
FLASH	◆ FLASH 存储器
SRAM	◆ SRAM 存储器

表 2-3 存储器及存储接口

2.3.3 系统模块

ES32M0150 微控制器包含以下系统模块：

模块	描述
系统配置控制器（SYSCFG）	◆ 系统的相关配置
复位控制（RMU）	◆ 系统所有复位的管理
DMA 多路复用（DMAMUX）	◆ DMA 请求源的多路复用选择器
DMA 控制器（DMAC）	◆ DMA 控制器可减少 CPU 负荷，提高系统效率；
外设互联（PIS）	◆ 外设互联系统为外设提供互联接口，可减少软件负担，提高了系统响应的及时性，同时为扩展应用场景提供了便利和灵活性。
看门狗定时器（IWDT,WWDT）	◆ 包含了独立看门狗和窗口看门狗。
调试配置控制器（DBGC）	◆ 系统调试相关的配置控制 ◆ 产生断点时，定时器/WDT 是否继续计数可以配置；

表 2-4 系统模块

2.3.4 时钟管理

ES32M0150 微控制器包含以下时钟管理模块：

模块	描述
时钟管理（CMU）	◆ 时钟源配置 ◆ 系统和外设时钟的选择及切换 ◆ 外部时钟停振检测（时钟安全机制） ◆ 外设时钟门控 ◆ 系统和外设时钟分频

表 2-5 时钟管理

2.3.5 外部接口

ES32M0150 微控制器包含以下外部接口模块：

模块	描述
通用 IO 及端口控制	◆ 通用 IO 的输入输出功能。 ◆ 对 IO 的控制还包括：上、下拉选择，开漏选择，驱动能力选择，端口 CMOS/TTL 输入功能选择，端口模拟滤波器使能控制等。 ◆ 通用 IO 端口支持 16 个中断源和 DMA 请求。

表 2-6 外部接口

2.3.6 安全管理及运算加速

ES32M0150 微控制器包含以下安全管理模块：

模块	描述
循环冗余校验模块(CRC)	◆ 支持四个常用的多项式： ◇ CRC-CCITT ◇ CRC-8 ◇ CRC-16 ◇ CRC-32

表 2-7 系统模块

2.3.7 定时器

ES32M0150 微控制器包含以下定时器模块：

模块	描述
通用定时器（AD16C4T）	◆ 16 位递增、递减、递增/递减自动重载计数器。 ◆ 16 位可编程预分频器，用于对计数器时钟频率进行分频，分频系数介于 1 到 65536 之间。 ◆ 多达 4 个独立通道，可用于： ◇ 输入捕获 ◇ 输出比较 ◇ PWM 生成（边沿和中心对齐模式） ◇ 单脉冲模式输出 ◇ 通道 1、2、3 可各自输出 1 组互补 PWM ◆ 3 组带可编程死区的互补输出（CH1/CH1n、CH2/CH2n、CH3/CH3n）。 ◆ 使用外部信号控制定时器且可实现多个定时器互连的同步电路。 ◆ 重复计数器，用于仅在给定数目的计数器周期后更新定时器寄存器。 ◆ 刹车功能，可将定时器的输出信号置于复位状态或已知状态。 ◆ 以下事件将生成中断请求： ◇ 更新：计数器上溢/下溢、计数器初始化（通过软件或内部/外部触发） ◇ 触发事件（计数器启动、停止、初始化或通过内部/外部触发计数） ◇ 输入捕获 ◇ 输出比较 ◇ 刹车输入 ◆ 支持定位用增量（正交）编码器和霍尔传感器电路。 ◆ 外部时钟触发输入或逐周期电流管理。 ◆ 该定时器可与其他定时器连接，一起配合使用，以达到同步或事件串联的目的。 ◆ 在调试模式下，定时器可被冻结。
通用定时器（GP16C2T）	◆ 16 位递增自动重载计数器。 ◆ 16 位可编程预分频器，用于对计数器时钟频率进行分频（可运行时修改），分频系数 介于 1 到 65536 之间。 ◆ 多达 2 个独立通道，可用于： ◇ 输入捕获 ◇ 输出比较

模块	描述
	◇ PWM 生成（边沿和中心对齐模式） ◇ 单脉冲模式输出 ◆ 带可编程死区的互补输出 ◆ 使用外部信号控制定时器且可实现多个定时器互连的同步电路 ◆ 发生如下事件时生成中断/DMA 请求： ◇ 更新：计数器上溢、计数器初始化（通过软件或内部/外部触发） ◇ 触发事件（计数器启动、停止、初始化或通过内部/外部触发计数） ◇ 输入捕获 ◇ 输出比较 ◇ 断路输入 ◆ 支持定位用增量（正交）编码器和霍尔传感器电路。 ◆ 外部时钟触发输入
基本定时器（BS16T）	◆ 16 位递增自动重载计数器 ◆ 16 位可编程预分频器，用于对计数器时钟频率进行分频（可运行时修改），分频系数 介于 1 到 65536 之间 ◆ 发生如下事件时生成中断/DMA 请求： ◇ 更新：计数器上溢

表 2-8 定时器

2. 3. 8 通信模块

ES32M0150 微控制器包含以下通信模块：

模块	描述
I2C 总线接口（I2C）	◆ 支持多主模式和总线仲裁 ◆ 可编程 I2C 地址检测 ◆ 最高通信速率为 400 KHz ◆ 可配置时钟延长 ◆ 兼容 SMBus2.0 协议 ◆ 兼容 PMBus 协议
串行外设接口（SPI）	◆ 支持半双工/全双工的同步串行通信 ◆ 主模式或从模式操作 ◆ 8 位或 16 位传输帧格式选择
通用同步异步收发器（USART）	◆ 支持与外部设备进行全双工数据通信和半双工单线通信 ◆ 提供小数波特率发生器可灵活配置多种波特率 ◆ 支持同步单向通信 ◆ 智能卡协议（ISO 7816） ◆ 支持多点通信 ◆ 硬件自动流控制 (CTS/RTS)

表 2-9 通信模块

2. 3. 9 模拟模块

ES32M0150 微控制器包含以下模拟模块：

模块	描述
模数转换（ADC）	12 位逐次逼近型模数转换器 ◆ 可配置的转换分辨率（6/8/10/12 位） ◆ 在规则转换、注入转换结束后以及发生模拟看门狗或溢出事件时产生中断 ◆ 支持单次或连续转换模式 ◆ 用于自动将通道 0 转换为通道“n”的扫描模式 ◆ 可配置的数据对齐方式 ◆ 可独立设置各通道采样时间 ◆ 可配置外部触发器选项，可为规则转换和注入转换配置极性 ◆ 支持不连续采样模式 ◆ 可配置的参考源选择 ◆ 可配置的转换时钟分频 ◆ 规则通道转换期间可产生 DMA 请求
模拟比较器（ACMP）	◆ 可配置的迟滞选择，0 ~ ±60mv 之间可选 8 个等级 ◆ 可配置多种低功耗工作模式
温感电压（TS）	可生成温感电压输出到 ADC 通道 17

表 2-10 模拟模块

第3章 芯片配置指引

3.1 概述

本章节主要说明以下内容：

- 芯片顶层相关模块的连接及信号路径
- 阅读各模块时可参考的相关信息链接
- 芯片顶层连接资源的相关配置
- 模块之间特殊的交互

3.2 ARM Cortex-M0 内核配置

ARM Cortex-M0 提供了高性能，低功耗，低成本的平台来满足 MCU 的实现要求。具备出色的计算性能，并能够快速响应中断。

3.2.1 ARM Cortex-M0 内核

关于 ARM Cortex-M0 内核技术细节可参考 ARM 官网 http://www.arm.com。

3.2.2 总线

支持 32 位 AMBA3 AHB 协议总线。Cortex-M0 内核提供一条 System 总线。

3.2.3 系统节拍定时器

系统节拍定时器为递减计数器，计数时钟为内核时钟。具体配置可参考系统节拍控制和状态寄存器（SysTick Control and Status Register）相关说明。

3.2.4 调试器件

支持标准 SWD 协议的调试接口。

3.3 嵌套向量中断控制器

ES32M0150 系列 MCU 的嵌套向量中断控制器可支持 4 个优先级设定。并具备以下特性：

- ◇ NVIC 与内核紧密配合支持快速中断响应时间
- ◇ 中断向量表直接传递至内核
- ◇ 支持中断嵌套，咬尾和迟来

3.3.1 中断优先级

中断优先级寄存器（Interrupt Priority Register）每个 byte 的高 2 位为有效位，支持 4 个中断优先级设置。

3.3.2 中断向量分配

中断向量分配如下表所示：

向量号	优先级	名称	说明	地址
0	—	—	保留	0x0000_0000
1	-3	Reset	复位	0x0000_0004
2	-2	NMI	时钟安全事件	0x0000_0008
3	-1	HardFault	所有类型的错误	0x0000_000C
4~10	—	—	保留	0x0000_0010 ~ 0x0000_002B
11	—	SVCall	通过 SWI 指令调用的系统服务	0x0000_002C
12-13	—	—	保留	0x0000_0030 0x0000_0034
14	—	PendSV	可挂起的系统服务	0x0000_0038
15	可设置	Systick	系统定时器	0x0000_003C
16	可配置	WWDT	WWDT 全局	0x0000_0040
17	可配置	IWDT	IWDT 全局	0x0000_0044
18	可配置	LVD	LVD 全局	0x0000_0048
19	可配置	—	—	0x0000_004C
20	可配置	CMU	CMU 全局	0x0000_0050
21	可配置	EXTI0_3	外部端口中断 0~3	0x0000_0054
22	可配置	EXTI4_7	外部端口中断 4~7	0x0000_0058
23	可配置	EXTI8_11	外部端口中断 8~11	0x0000_005C
24	可配置	EXTI12_15	外部端口中断 12~15	0x0000_0060
25	可配置	DMA	DMA 全局	0x0000_0064
26	可配置	ACMP0	ACMP0 全局	0x0000_0068
27	可配置	ACMP1	ACMP1 全局	0x0000_006C
28	可配置	ADC	ADC 全局	0x0000_0070
29	可配置	AD16C4T_UP _TRIG_COM	AD16C4T0 更新、触发和 COM 中断	0x0000_0074
30	可配置	AD16C4T_CC	AD16C4T0 捕捉和比较中断	0x0000_0078

31	可配置	BSTIM0	BS16T0 全局	0x0000_007C
32	可配置	—	—	0x0000_0080
33	可配置	GPTIMC0	GP16C2T0 全局	0x0000_0084
34	可配置	GPTIMC1	GP16C2T1 全局	0x0000_0088
35	可配置	GPTIMC2	GP16C2T2 全局	0x0000_008C
36	可配置	—	—	0x0000_0090
37	可配置	AD16C4T	AD16C4T 全局	0x0000_0094
38	可配置	—	—	0x0000_0098
39	可配置	I2C0	I2C0 全局	0x0000_009C
40	可配置	I2C1	I2C1 全局	0x0000_00A0
41	可配置	SPI0	SPI0 全局	0x0000_00A4
42	可配置	—	—	0x0000_00A8
43	可配置	—	—	0x0000_00AC
44	可配置	—	—	0x0000_00B0
45	可配置	USART0	USART0 全局	0x0000_00B4
46	可配置	USART1	USART1 全局	0x0000_00B8
47	可配置	—	—	0x0000_00BC

表 3-1 中断向量分配

3.4 存储器及存储器接口

3.4.1 系统总线和存储器

ES32M0150 系列产品外设存储映射如下表所示：

总线	边界地址	外设
APB	0x4000 0000 - 0x4000 03FF	AD16C4T
	0x4000 0400 - 0x4000 07FF	BS16T
	0x4000 0800 - 0x4000 0BFF	GP16C2T0
	0x4000 0C00 - 0x4000 0FFF	GP16C2T1
	0x4000 1000 - 0x4000 13FF	GP16C2T2
	0x4000 1400 - 0x4000 17FF	—
	0x4000 1800 - 0x4000 1BFF	—
	0x4000 1C00 - 0x4000 1FFF	—
	0x4000 2000 - 0x4000 3FFF	—
	0x4000 4000 - 0x4000 43FF	—
	0x4000 4400 - 0x4000 47FF	—
	0x4000 4800 - 0x4000 4BFF	—
	0x4000 4C00 - 0x4000 4FFF	—
	0x4000 5000 - 0x4000 53FF	USART0
	0x4000 5400 - 0x4000 57FF	USART1
	0x4000 5800 - 0x4000 5FFF	—
	0x4000 6000 - 0x4000 63FF	SPI0
	0x4000 6400 - 0x4000 67FF	—
	0x4000 6800 - 0x4000 6BFF	—
	0x4000 6C00 - 0x4000 7FFF	—
	0x4000 8000 - 0x4000 83FF	I2C0
	0x4000 8400 - 0x4000 87FF	I2C1
	0x4000 8800 - 0x4000 8BFF	WWDT
	0x4000 8C00 - 0x4000 8FFF	IWDT
	0x4000 9000 - 0x4000 93FF	DBGC
	0x4000 9400 - 0x4000 97FF	ADC
	0x4000 9800 - 0x4000 9BFF	ACMP0
	0x4000 9C00 - 0x4000 9FFF	ACMP1
	0x4000 A000 - 0x4000 B3FF	—
	0x4000 B400 - 0x4000 BFFF	—
	0x4000 C000 - 0x4000 D3FF	DMA
	0x4000 D400 - 0x4007 FFFF	—
AHB	0x4008 0000 - 0x4008 03FF	SYSCFG
	0x4008 0400 - 0x4008 07FF	CMU
	0x4008 0800 - 0x4008 0BFF	RMU
	0x4008 0C00 - 0x4008 0FFF	PMU

总线	边界地址	外设
	0x4008 1000 - 0x4008 13FF	MSC
	0x4008 1400 - 0x4008 3BFF	—
	0x4008 3C00 - 0x4008 3FFF	—
	0x4008 4000 - 0x4008 4FFF	GPIO
	0x4008 5000 - 0x4008 53FF	CRC
	0x4008 5400 - 0x4008 57FF	—
	0x4008 5800 - 0x4008 5BFF	—
	0x4008 5C00 - 0x4008 5FFF	—
	0x4008 6000 - 0x4008 63FF	PIS
	0x4008 6400 - 0x4008 67FF	—
	0x4008 6800 - 0x4008 FFFF	—

表 3-2　外设存储映射

3.5 系统模块配置

3.5.1 DMA控制器配置

DMA 控制器包含 4 个通道，每个 DMA 通道对应一个 DMA 多路复用器，每个多路复用器包含了微控制器所有的 DMA 申请源，由 DMA_CHx_SELCON（x=0,1,2,3）配置选择。多路复用器和 DMA 之间连接图如下：

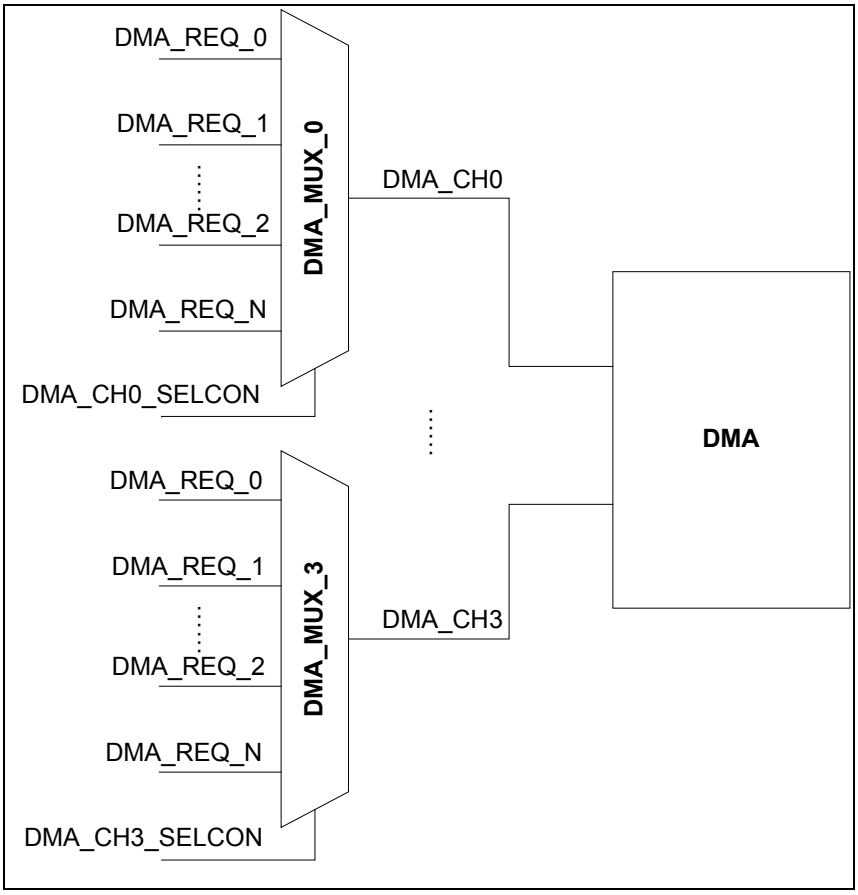


图 3-1 DMA 多路复用器与 DMA 连接图

每个 DMA 多路复用器可选择的 DMA 请求如下表所示：

模块名	DMA 请求源
GPIO	EXTI0~EXTI15
ADC	ADC 转换结束
AD16C4T0	AD16C4T0_CH1
	AD16C4T0_CH2
	AD16C4T0_CH3
	AD16C4T0_CH4
	AD16C4T0_TRIG
	AD16C4T0_COM
	AD16C4T0_UP
BS16T0	BS16T0_UP
GP16C2T0	GP16C2T0_CH1
	GP16C2T0_CH2
	GP16C2T0_TRIG
	GP16C2T0_COM
	GP16C2T0_UP
GP16C2T1	GP16C2T1_CH1
	GP16C2T1_CH2
	GP16C2T1_TRIG
	GP16C2T1_COM
	GP16C2T1_UP
GP16C2T2	GP16C2T2_CH1
	GP16C2T2_CH2
	GP16C2T2_TRIG
	GP16C2T2_COM
	GP16C2T2_UP
USART0	USART0_TX
	USART0_RX
USART1	USART1_TX
	USART1_RX
SPI0	SPI0_TX
	SPI0_RX
I2C0	I2C0_TX
	I2C0_RX
I2C1	I2C1_TX
	I2C1_RX
CRC	CRC DMA 写请求
PIS	PIS 通道 0~5

表 3-3 DMA 请求列表

3.6 外部接口配置

3.6.1 通用IO及端口控制配置

3.6.1.1 端口特殊配置说明

JLINK 通讯接口 SWDIO、SWCLK 与 PB0、PB1 复用，若程序有配置 PB0、PB1 的复用模式为非 0，可能导致后续 JLINK 连接异常。

3.7 模拟配置

3.7.1 ADC控制配置

3.7.1.1 ADC模块例化

ES32M0150 包含 1 路 ADC。

3.7.1.2 ADC转换通道配置

ADC 支持 20 个通道选择（其中包含 16 个外部通道），具体分配如下表所示，每个 ADC 通道在管脚上的对应关系，请参考数据手册的管脚功能定义表格。

ADC 通道	信号分配
ADC 通道 0	AIN0
ADC 通道 1	AIN1
ADC 通道 2	AIN2
ADC 通道 3	AIN3
ADC 通道 4	AIN4
ADC 通道 5	AIN5
ADC 通道 6	AIN6
ADC 通道 7	AIN7
ADC 通道 8	AIN8
ADC 通道 9	AIN9
ADC 通道 10	AIN10
ADC 通道 11	AIN11
ADC 通道 12	AIN12
ADC 通道 13	AIN13
ADC 通道 14	AIN14
ADC 通道 15	AIN15
ADC 通道 16	保留
ADC 通道 17	VTEMP（温感）
ADC 通道 18	VDD/4 电压
ADC 通道 19	VREF1V
—	—

表 3-4 ADC 转换通道配置

注 1：ES32M0150 系列芯片暂不支持 ADC 通道 AIN15。

注 2：当使用 ADC 通道 19 时，模拟通道输入信号是内部 VREF1V 电压，ADC 时钟速率需小于 16KHz，.采样周期需大于 1ms，否则会影响 ADC 转换精度，推荐将 ADC 时钟设置为 128 分频（寄存器 ADC_CCR 的 CKDIV=111），采样时间设置为 15 个周期（寄存器 ADC_SMPTx 的 CHTy=0011）。则当 PCLK 频率为 2MHz 时，可满足上述 ADC 时钟速率和采样周期的要求；当 PCLK 频率大于 2MHz 时，可在 ADC 转换过程中，通过反复开关 ADC 模块外设时钟的方法来降低 ADC 时钟速率，并满足采样周期的要求：例如先设置 CMU_APBENR.ADCEN=0，延时约 12us，再设置 CMU_APBENR.ADCEN=1，然后继续设置 CMU_APBENR.ADCEN=0，如此循环，直到 ADC 转换完成，具体可参考芯片应用笔记和例程。

3.7.1.3 ADC电源及参考电压

ADC 电源由 VDD 提供。

ADC 的参考电压可选 VDD 或 VREFP（VREFP 由外部端口 VREFP 输入）。

3.7.1.4 ADC的时钟

ADC 的总线时钟和模块时钟源为 PCLK。

3.7.2 ACMP控制配置

3.7.2.1 ACMP模块例化

ES32M0150 包含 2 路 ACMP 模块（ACMP0，1）。

3.7.2.2 ACMP比较通道配置

ACMP0，1 正端支持 4 个外部通道选择，负端支持 2 个外部通道和 2 个内部通道选择。具体分配如下表。ACMP 通道与管脚的对应关系请参考数据手册。

寄存器 ACMP_INPUTSEL 的 PSEL	ACMP 通道	信号分配
000	ACMP 外部通道 0	CMPxP0
001	ACMP 外部通道 1	CMPxP1
010	ACMP 外部通道 2	CMPxP2
011	ACMP 外部通道 3	CMPxP3

表 3-5 ACMP 正端通道选择

寄存器 ACMP_INPUTSEL 的 NSEL	ACMP 通道	信号分配
0000	ACMP 外部通道 0	参考电压 VREF（1V）
0001	ACMP 外部通道 1	VDD 分压
0010	ACMP 内部通道 2	CMPxN0
0011	ACMP 内部通道 3	CMPxN1

表 3-6 ACMP 负端通道选择

3.7.2.3 ACMP电源及参考电压

ACMP 电源为 VDD。

3.7.2.4 ACMP的时钟

ACMP 的总线时钟和模块时钟源为 PCLK。

第4章　系统总线和存储器

4.1　概述

主系统由 32 位多层 AHB 总线矩阵构成，可实现以下部分的互连。

2 条主控总线：

- Cortex-M0 内核总线
- DMA 总线

4 条被控总线：

- 内部 Flash 总线
- 内部 SRAM 总线
- AHB 外设
- APB 外设

借助总线矩阵，可以实现主控总线到被控总线的访问，这样即使在多个高速外设同时运行期间，系统也可以实现并发访问和高效运行。

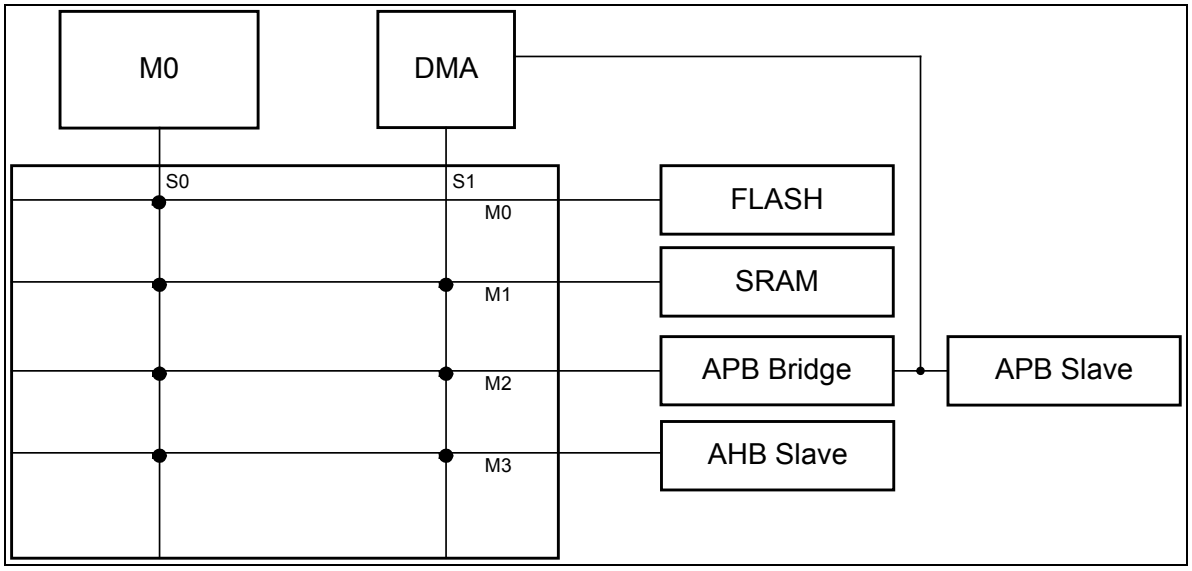


图 4-1　系统总线矩阵

4.2 系统总线

4.2.1 S0：Cortex-M0 内核总线

此总线用于将 Cortex-M0 内核的系统总线连接到总线矩阵。此总线用于访问位于存储器中的程序和数据，以及系统控制寄存器和其他外设寄存器空间。

4.2.2 S1：DMA总线

此总线用于将 DMA 存储器总线主接口连接到总线矩阵。DMA 通过此总线访问外设或执行存储器间的数据传输。此总线访问的对象是 AHB 和 APB 外设以及内部 SRAM。

4.2.3 总线矩阵

总线矩阵用于主控总线之间的访问仲裁管理。仲裁采用循环调度算法。

4.2.4 AHB/APB总线桥

借助两个 AHB/APB 总线桥可实现 AHB 总线与 APB 总线之间的桥接，从而可灵活配置外设频率。

4.3 存储器的组织结构

程序存储器、数据存储器、寄存器和 I/O 端口排列在同一个顺序的 4 GB 地址空间内。

各字节按小端格式在存储器中编码。字中编号最低的字节被视为该字的最低有效字节，而编号最高的字节被视为最高有效字节。

有关外设寄存器映射的详细信息，请参见相关章节。

可寻址的存储空间分为 8 个主要块，每个块为 512MB。

未分配给片上存储器和外设的所有存储区域均视为“保留区”。请参见产品数据手册中的存储器映射图。

4.3.1 系统存储器映射

系统存储器映射图如下表所示：

地址范围	存储	备注
0x0000_0000~0x0000_FFFF	FLASH 主存储区	64KB
0x0001_0000~0x0001_0400	FLASH 信息区	1KB
0x0004_0000~0x1FFF_FFFF	Reserved	
0x2000_0000~0x2000_0FFF	SRAM	4KB
0x2000_8000~0x3FFF_FFFF	Reserved	
0x4000_0000~0x4007_FFFF	APB 外设	
0x4008_0000~0x4008_FFFF	AHB 外设	
0x4009_0000~0x5FFF_FFFF	Reserved	
0x6000_0000~0x9FFF_FFFF	Reserved	
0xA000_0000~0xDFFF_FFFF	Reserved	
0xE000_0000~0xE00F_FFFF	私有外设	

地址范围	存储	备注
0xE010_0000~0xFFFF_FFFF	Reserved	

表 4-1 系统存储器映射

4.3.2 FLASH存储器映射

Flash 接口管理 CPU 通过系统总线对 Flash 进行访问。该接口可针对 Flash 执行擦除和编程操作，并实施读写保护机制。

Flash 结构如下：

- ◇ ES32M0150 系列主存储区（基地址 0x0000_0000）共 64K Bytes，分为 128 页，每页 512 Bytes。
- ◇ 芯片配置字存放在信息区（基地址 0x0001_0000）中，用于配置读写保护、BOR 电压、软件/硬件看门狗等。

4.3.3 SRAM存储器映射

SRAM 容量为 4K Bytes，地址为 0x2000_0000~0x2000_0FFF，单周期访问时间。

4.3.4 外设存储映射

ES32M0150 产品外设存储映射请参考章节“芯片配置指引”的外设存储映射表。

4.4 启动引导

芯片发生上电或复位后（除内核软复位和 CPU 复位），CPU 从用户配置字指定的地址开始运行程序，在程序运行中可软件配置中断向量偏移寄存器（SYSCFG_VTOR），再执行内核软复位或 CPU 复位，程序将从 SYSCFG_VTOR 指定的地址运行。

第5章 存储器系统控制（MSC）

5.1 概述

存储器系统控制（MSC）主要作用是控制系统程序编程 Flash 的各种操作，包括全擦除，页擦除，写操作，读操作以及对应的访问权限管理等，并实时反馈各种控制操作中的状态，以便于系统对 Flash 编程进行控制。

存储器系统控制（MSC）支持 Flash 主程序区的全局读保护。

存储器系统控制（MSC）中可以划分独立数据 Flash 区域用于存放用户数据，用户可根据具体应用灵活选择。

5.2 特性

- 支持对 Flash 的编程和擦除控制
  - 程序区全擦除
  - 非私有保护区全擦除
  - 页擦除
  - 字编程
- 支持对存储器等待时间控制
  - 支持 Flash 读取等待时间可配置为 0~3 个系统时钟周期
  - 支持 Sram 读取等待时间可配置为 0~3 个系统时钟周期

5.3 结构框图

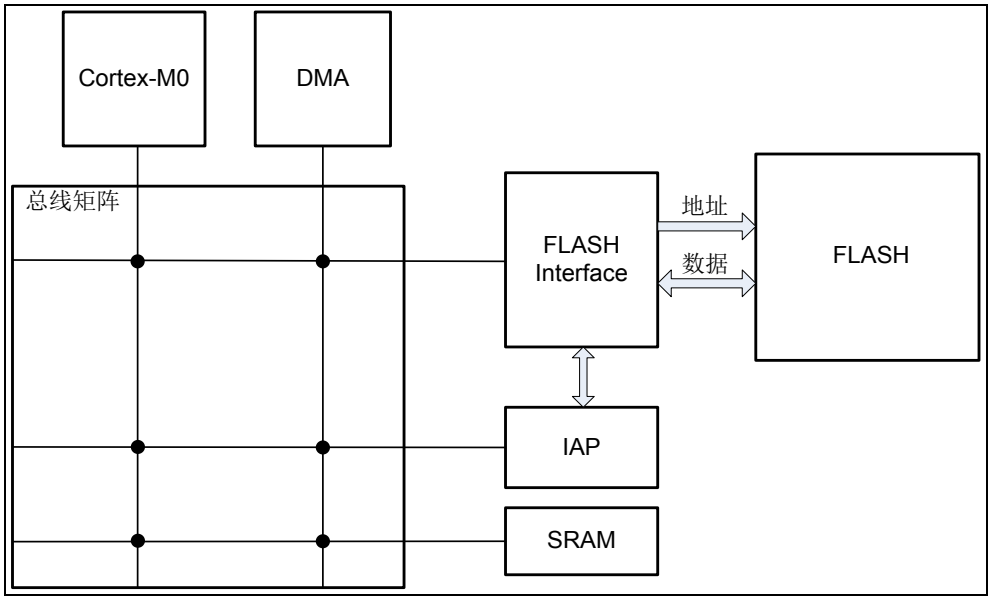


图 5-1 存储器控制结构图

在上电时，系统从 Flash 中加载配置字，配置字加载成功后，系统从用户配置字指定启动地址开始运行。

5.4 功能描述

5.4.1 Flash保护

5.4.1.1 IAP操作保护KEY

软件通过写 MSC_FLASHKEY 寄存器，可解除对程序区的保护，处于保护状态时，无法进行擦除和编程的操作。通过检查 MSC_FLASHKEY.STATUS 是否为 0，判断 Flash 是否处于保护状态。

5.4.1.2 Flash写保护区

Flash 存储器可以通过配置字 CFG_WRP0 和 CFG_WRP1 的 START、END 位配置两段写保护区域，通过 ENB 位配置两段写保护区域使能。

Flash 页擦除和 Flash 字编程，无法对写保护区擦除和写入，Flash 全擦时，可以将写保护区数据清除。

写保护区域	使能	起始页号	结束页号
区域 1	CFG_WRP0.ENB	CFG_WRP0.START	CFG_WRP0.END
区域 2	CFG_WRP1.ENB	CFG_WRP1.START	CFG_WRP1.END

表 5-1 写保护区配置字对应表

5.4.1.3 Flash私有读保护区

Flash 存储器可以通过配置字 CFG_PCROP0 和 CFG_PCROP1 的 START、END 位配置两段私有读保护区，通过 ENB 位配置使能。

Flash 对私有读保护区进行任何非法的读取或擦写，均会置位对应的错误标识；但是可以进行 Flash 全擦操作，且 Flash 全擦时，可以将读保护区数据清除。

私有读保护区	使能	起始页号	结束页号
区域 1	CFG_PCROP0.ENB	CFG_PCROP0.START	CFG_PCROP0.END
区域 2	CFG_PCROP1.ENB	CFG_PCROP1.START	CFG_PCROP1.END

表 5-2 私有读保护区配置字对应表

5.4.1.4 数据Flash区

Flash 区域可以通过配置字 CFG_DAFLS 划分数据 Flash 区，通过 CFG_DAFLS 的 ENB 位配置数据 Flash 的使能。

Data Flash	使能	起始页号	结束页号
区域 1	CFG_DAFLS.ENB	CFG_DAFLS.START	CFG_DAFLS.END

表 5-3 数据 Flash 配置字对应表

5. 4. 1. 5 Flash全局读保护

Flash 存储器可以进行全局读保护，保护等级分为 Level0，Level1，Level2。

当全局保护字为 32 位全 1 时，全局保护级别即为 Level0。

当全局保护字高 16 位为全 1 且低 16 位为非全 1 时，全局保护级别即为 Level1。

当全局保护字高 16 位为非全 1 且低 16 位也为非全 1 时，全局保护级别即为 Level2。

不同全局加密保护级别下的访问限制如下表：

存储区		全局保护级别	调试模式 运行程序			用户模式					
						在 FLASH 中运行			在 SRAM 中运行		
			擦	写	读	擦	写	读	擦	写	读
Flash Code 区	非私有读保护区	Level0	全擦/页擦	是	是	NA	NA	是	页擦	是	是
		Level1	全擦/页擦	否	否	NA	NA	是	页擦	是	否
		Level2	否	否	否	NA	NA	是	页擦	是	否
	私有区保护区	Level0	全擦	否	是	NA	NA	否	否	否	否
		Level1	全擦	否	否	NA	NA	否	否	否	否
		Level2	否	否	否	NA	NA	否	否	否	否
	写保护区	Level0	全擦	是	是	NA	NA	是	否	否	是
		Level1	全擦	是	是	NA	NA	是	否	否	是
		Level2	否	否	否	NA	NA	是	否	否	是

表 5-4 不同全局保护级别下的访问限制表

注 1：调试模式下、在 SRAM 中运行程序时，若全局读保护等级为 Level1 或 Level2 时，不能读取 Flash Code 区。
注 2：用户模式下，在 Flash 中运行程序时，只有私有读保护读保护区不能被读出，其他均可以被读出。
注 3：在 Flash 中运行程序时，禁止对 Flash 本身进行擦写操作，见上表标识 NA。
注 4：用户模式下，在 SRAM 或 IAPROM 中运行程序时，可以擦写 Flash Code 区中的非私有读保护区。
注 5：用户模式下，不支持对 Flash 的全擦和非私有读保护区全擦命令。
注 6：info 区在所有全局读保护等级下都为只读。

5. 4. 2 Flash程序区全擦除

程序区全擦除可擦除全部程序区空间，一次全擦除耗时约 8ms。具体步骤如下：

1. 查 MSC_FLASHSR.BUSY 标志是否处于空闲状态；
2. 通过 MSC_FLASHKEY 解除 Flash 程序区保护状态；
3. 设置 Flash 操作请求使能；
4. 写入 MSC_FLASHCMD.CMD 命令触发全擦除；
5. 等待 MSC_FLASHSR.BUSY 标志再次变为空闲状态；
6. 判断 MSC_FLASHSR.MASE 标志位是否置起；
7. 设置 Flash 操作请求禁止。

5. 4. 3 Flash非私有读保护区全擦除

擦除私有读保护区以外的程序区空间。具体步骤如下：

1. 查 MSC_FLASHSR.BUSY 标志是否处于空闲状态；
2. 通过 MSC_FLASHKEY 解除 Flash 程序区保护状态；
3. 设置 Flash 操作请求使能；
4. 写入程序区的首地址；
5. 写入 MSC_FLASHCMD.CMD 命令触发全擦除；
6. 等待 MSC_FLASHSR.BUSY 标志再次变为空闲状态；
7. 判定 MSC_FLASHSR.UPCEBUSY 是否为空闲状态；
8. 判断 MSC_FLASHSR.UPCEDONE 标志位是否置起；
9. 设置 Flash 操作请求禁止。

5. 4. 4 Flash页擦除

页擦除可擦除固定一页空间，其中程序区一页大小可为 512Bytes，一次页擦除耗时约 2ms。具体步骤如下：

1. 检查 MSC_FLASHSR.BUSY 标志是否处于空闲状态；
2. 通过 MSC_FLASHKEY 解除 Flash 程序区保护状态；
3. 设置 Flash 操作请求使能；
4. 写入需擦除页的首地址；
5. 选择页大小；
6. 写入 MSC_FLASHCMD.CMD 命令触发页擦除；
7. 等待 MSC_FLASHSR.BUSY 标志再次变为空闲状态；
8. 判断 MSC_FLASHSR.SERA 标志位是否置起；
9. 设置 Flash 操作请求禁止。

注：数据 Flash 页擦除流程与普通 Flash 页擦除流程一致，仅触发命令不同。

5. 4. 5 Flash字编程

程序区字编程可一次编程 4 Bytes 空间，一次字编程耗时约 25us。具体步骤如下：

1. 检查 MSC_FLASHSR.BUSY 标志是否处于空闲状态；
2. 通过 MSC_FLASHKEY 解除 Flash 程序区保护状态；
3. 设置 Flash 操作请求使能；
4. 写入需编程地址；
5. 写入需编程数据 MSC_FLASHDR.DATA；
6. 写入 MSC_FLASHCMD.CMD 命令触发字编程；

7. 等待 MSC_FLASHSR.BUSY 标志再次变为空闲状态；
8. 判断 MSC_FLASHSR.PROG 标志位是否置起；
9. 设置 Flash 操作请求禁止。

注：数据 Flash 字编程流程与普通 Flash 字编程流程一致，仅触发命令不同。

5.4.6 Flash编程数据FIFO

FLASH 编程数据 FIFO 可通过 FIFOEN 使能，该 FIFO 为写入 FIFO，读取无效。当数据写入 FIFO 后，可在 MSC_FLASHDR 寄存器中体现。在 FIFO 中写入一次数据时，可触发一次编程。

5.4.7 IAP自编程硬件固化模块

芯片内置 IAP 自编程固化模块，由硬件电路实现，在 IAP 自编程操作程序中可以调用这些自编程固化模块，以减少 SRAM 中的 IAP 操作代码量。

IAP 自编程硬件固化模块支持页擦，单字编程，双字编程和多字编程，每次调用 IAP 操作函数之前，需要进行解锁操作。分别由如下 IAP 操作函数来实现：

5.4.7.1 CODE区单页擦函数

- 函数功能：擦除 CODE 区指定的页
- 入口地址：0x10000004
- 输入参数：R0-擦除页的首地址，R1-擦除页首地址的反码，R2-设为 0
- 返回值： R0-函数执行状态（R0=1 为成功，R0=0 为失败）

5.4.7.2 CODE区单字编程函数

- 函数功能：向 FLASH CODE 区指定地址写入一个字(32-bits)
- 入口地址：0x10000008
- 输入参数：R0-待编程的 FLASH 地址，R1-待编程的 FLASH 地址的反码，R2-待编程数据
- 返回值： R0-函数执行状态（R0=1 为成功，R0=0 为失败）

5.4.7.3 CODE区多字编程

- 函数功能：向 FLASH CODE 区指定地址写入多个字
- 入口地址：0x10000000
- 输入参数：R0-待编程的 FLASH 首地址，R1-待编程的 FLASH 首地址的反码，R2-放在 SRAM 空间的编程数据首地址，R3-编程数据长度，R4-当编程到页首时是否先进行页擦除（R4≠0 为擦除，R4=0 为不擦除）
- 返回值： R0-函数执行状态（R0=1 为成功，R0=0 为失败）

5.4.7.4 DATA区单页擦函数

- 函数功能：擦除 DATA 区指定的页
- 入口地址：0x10000014

- 输入参数：R0-擦除页的首地址，R1-擦除页首地址的反码，R2-设为 0
- 返回值： R0-函数执行状态（R0=1 为成功，R0=0 为失败）

5. 4. 7. 5 DATA区单字编程函数

- 函数功能：向 FLASH DATA 区指定地址写入一个字(32-bits)
- 入口地址：0x10000018
- 输入参数：R0-待编程的 FLASH 地址，R1-待编程的 FLASH 地址的反码，R2-待编程数据
- 返回值： R0-函数执行状态（R0=1 为成功，R0=0 为失败）

5. 4. 7. 6 DATA区多字编程

- 函数功能：向 FLASH DATA 区指定地址写入多个字
- 入口地址：0x10000010
- 输入参数：R0-待编程的 FLASH 首地址，R1-待编程的 FLASH 首地址的反码，R2-放在 SRAM 空间的编程数据首地址，R3-编程数据长度，R4-当编程到页首时是否先进行页擦除（R4≠0 为擦除，R4=0 为不擦除）
- 返回值： R0-函数执行状态（R0=1 为成功，R0=0 为失败）

注：在进行页擦时传递的输入参数 R0 必须为 512 的整数倍，否则会导致操作失败。

5. 5 特殊功能寄存器

5. 5. 1 寄存器列表

MSC 寄存器列表		
名称	偏移地址	描述
MSC_FLASHKEY	000_H	FLASH 程序区操作关键码寄存器
—	004_H	保留
MSC_FLASHADDR	008_H	FLASH 擦除编程地址寄存器
MSC_FLASHFIFO	$00C_H$	FLASH 编程数据写缓存寄存器
MSC_FLASHDR	010_H	FLASH 编程数据寄存器
—	014_H	保留
MSC_FLASHCMD	018_H	FLASH 操作命令寄存器
MSC_FLASHCR	$01C_H$	FLASH 控制寄存器
MSC_FLASHSR	020_H	FLASH 状态寄存器
—	024_H	保留
MSC_MEMWAIT	028_H	存储器读取等待时间寄存器
MSC_FLASHADDINV	$02C_H$	FLASH 擦除编程地址反码寄存器

5. 5. 2 寄存器描述

5. 5. 2. 1 FLASH程序区关键码寄存器（MSC_FLASHKEY）

FLASH 程序区关键码寄存器（MSC_FLASHKEY）																															
偏移地址：00_H																															
复位值：$00000000\_00000000\_00000000\_00000011_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																														STATUS	

Reserved	Bit 31-2	—	保留
STATUS	Bit 1-0	R/W	**FLASH 程序区状态位** 00：可擦除或编程 其他：被保护，不可擦除或编程 IAP复位可将该寄存器复位

注：对上述该寄存器连续写入 0x8ACE0246 和 0x9BDF1357 可去除保护，写入其他值或中间插入其他操作将失效。

5. 5. 2. 2 FLASH擦除编程地址寄存器（MSC_FLASHADDR）

FLASH 擦除编程地址寄存器（MSC_FLASHADDR）

偏移地址：08_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-16	—	保留
ADDR	Bit 15-0	R/W	**FLASH 地址**

关于上述寄存器中的 ADDR 位：

注 1：低 2 位写入无效，读出始终为 0。

注 2：页擦除完成后，若 MSC_FLASHCR.PGSZ 配置为 0，则地址自动加 0x200；若 MSC_FLASHCR.PGSZ 配置为 1，则地址自动加 0x800。

注 3：字编程完成后，地址自动加 4。

5. 5. 2. 3 FLASH编程FIFO寄存器（MSC_FLASHFIFO）

FLASH 编程 FIFO 寄存器（MSC_FLASHFIFO）

偏移地址：$0C_H$

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

FIFO	Bit 31-0	W	**FLASH编程FIFO**

注 1：需通过 FIFOEN 使能 FIFO，写入编程数据，可适用于 DMA 传输数据，先写入低位数据，再写入高位数据

注 2：当写入相应个数数据后，将自动触发字编程

5. 5. 2. 4 FLASH编程数据寄存器（MSC_FLASHDR）

FLASH 编程数据寄存器（MSC_FLASHDR）

偏移地址：10_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

DATA

DATA	Bit 31-0	R/W	**FLASH编程数据**

5. 5. 2. 5 FLASH命令寄存器（MSC_FLASHCMD）

FLASH 命令寄存器（MSC_FLASHCMD）

偏移地址：18_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

CMD	Bit 31-0	W	**FLASH编程命令** 0x000051AE：Flash Code区全擦除 0x00005EA1：普通Flash页擦除 0x00005DA2：普通Flash字编程 0x00005BA4：数据Flash页擦除 0x00005AA5：数据Flash字编程 0x000050AF：非私有读保护区全擦除 其他：保留

注：私有保护区通过私有读出保护区配置字（CFG_PCROPx）设置，如果未设置私有保护区，建议使用全擦除命令。

5. 5. 2. 6 FLASH控制寄存器（MSC_FLASHCR）

FLASH 控制寄存器（MSC_FLASHCR）

偏移地址：$1C_H$

复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																								FIFODF	Reserved	FIFOEN	FLASHREQ	Reserved	OTFEN	IAPRST	IAPEN

Reserved	Bit 31-8	—	**保留**
FIFODF	Bit 7	R/W	**FIFO 编程数据 Flash 使能** 0：禁止 1：使能

Reserved	Bit 6	—	保留
FIFOEN	Bit 5	R/W	**FIFO 使能** 0：禁止 1：使能
FLASHREQ	Bit 4	R/W	**FLASH 操作请求使能** 0：禁止 1：使能
Reserved	Bit 3	—	保留
OTFEN	Bit2	R/W	**自编程操作 FLASH 运行使能** 0：禁止 1：使能（仅用作测试，实际应用需设置 0）
IAPRST	Bit 1	W1	**自编程复位** 0：无操作 1：自编程复位
IAPEN	Bit 0	R/W	**自编程使能** 0：禁止 1：使能

5. 5. 2. 7 FLASH状态寄存器（MSC_FLASHSR）

FLASH 状态寄存器（MSC_FLASHSR）

偏移地址：20_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						UPCEUL	MASEUL	ADDERR	Reserved					UPCEDONE	UPCEBUSY	Reserved								TIMEOUT	PROG	SERA	MASE	WAE	WPE	BUSY	FLASHACK

Reserved	Bit 31-26	—	保留
UPCEUL	Bit 25	R	**非私有读保护区全擦保护解除位** 0：未解除 1：已解除 解除保护后，允许对私有读保护配置区进行擦除
MASEUL	Bit 24	R	**程序区全擦保护解除位** 0：未解除 1：已解除 解除保护后，允许对对用户配置区和私有读保护配置区进行擦除
ADDERR	Bit 23	R	**地址反码错误标识位** 0：反码正确 1：反码错误
Reserved	Bit 22-18	—	保留
UPCEDONE	Bit 17	R	**非私有读保护区全擦完成标志**

			0：未进行或正在进行中 1：已完成 重新启动新的擦除或编程操作时自动清除
UPCEBUSY	Bit 16	R	**非私有读保护区全擦状态位** 0：空闲 1：正在进行
Reserved	Bit 15-8	—	**保留**
TIMEOUT	Bit 7	R	**超时错误标志** 0：无错误 1：发生错误 未在规定时间内完成相应擦除或编程动作时产生错误标志，可能硬件发生了故障，需软件触发一次IAP复位
PROG	Bit 6	R	**字编程完成标志** 0：未进行或正在进行中 1：已完成 重新启动新的擦除或编程操作时自动清除
SERA	Bit 5	R	**页擦除完成标志** 0：未进行或正在进行中 1：已完成 重新启动新的擦除或编程操作时自动清除
MASE	Bit 4	R	**程序区全擦除完成标志** 0：未进行或正在进行中 1：已完成 重新启动新的擦除或编程操作时自动清除
WAE	Bit 3	R	**擦写地址错误标志** 0：无错误 1：发生错误 可能是在IAP操作在非法的FLASH地址，或是在擦除和编程时使用了错误的命令，需软件触发一次IAP复位
WPE	Bit 2	R	**擦写保护错误标志** 0：无错误 1：发生错误 触发了保护区域的擦除或编程，操作失败，需软件触发一次IAP复位
BUSY	Bit 1	R	**自编程状态复位** 0：空闲 1：正在进行
FLASHACK	Bit 0	R	**FLASH 操作许可状态** 0：禁止操作 1：允许操作

5. 5. 2. 8 存储器读取等待时间寄存器（MSC_MEMWAIT）

存储器读取等待时间寄存器（MSC_MEMWAIT）

偏移地址：28_H

复位值：$00000000\_00000000\_00000000\_00000010_B$

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

Reserved	Bit 31-10	—	保留
SRAM_W	Bit 9-8	R/W	**SRAM 读取等待时间** 00：无等待 01：1 个 SYSCLK 10：2 个 SYSCLK 11：3 个 SYSCLK
Reserved	Bit 7-2	—	保留
FLASH_W	Bit 1-0	R/W	**FLASH 读取等待时间选择** 00：无等待 01：1 个 SYSCLK 10：2 个 SYSCLK 11：3 个 SYSCLK 仅 SYSCLK 不超过 24MHz 时才可设为 00，其他情况不限。

5. 5. 2. 9 FLASH擦除编程地址反码寄存器（MSC_FLASHADDINV）

FLASH 擦除编程地址反码寄存器（MSC_FLASHADDINV）

偏移地址：08_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-16	—	保留
ADDRINV	Bit 15-0	R/W	**FLASH 地址反码**

第6章 系统配置控制器（SYSCFG）

6.1 概述

系统配置模块（SYSCFG）用于芯片的系统级功能配置。

6.2 特性

- 支持寄存器写保护功能
- 支持存储器重映射功能
- 支持中断向量重映射功能

6.3 功能描述

6.3.1 系统寄存器写保护

为避免程序的异常运行对系统级模块的误操作，系统写保护寄存器 SYSCFG_PROT 用于阻止程序对系统级模块其它寄存器的误写入。该寄存器保护范围为除 SYSCFG_PROT 寄存器外的 SYSCFG、CMU、RMU 模块所有寄存器。

SYSCFG_PROT 寄存器为虚拟寄存器。要对系统级模块其它寄存器进行写操作时，需先对 SYSCFG_PROT 寄存器写 0x55AA6996，之后可对系统级模块其它寄存器进行写操作。对 SYSCFG_PROT 寄存器写入其他值重新进入写保护状态，写保护状态下对系统寄存器进行的写操作将被忽略。

可以通过读取 SYSCFG_PROT 寄存器确认系统级模块是否处于写保护状态，读出值为 0x00000000，表示当前可对系统级模块寄存器进行写操作；读出值为 0x00000001 表示系统级模块处于写保护状态。SYSCFG_PROT 寄存器无其它读出值。

6.3.2 存储器重映射

存储器重映射应用于系统的 2 种启动模式：

- ◇ 从用户程序启动（地址为 0x00000000）
- ◇ 从用户 Boot 启动（地址为 0x0000F000）

系统在发生上电复位、低电压复位或外部端口复位时，会固定从 Flash 中运行。通过配置芯片配置字中的 Flash 启动地址选择位来决定跳转至 BootFlash 还是用户 Flash。用户根据实际需要编写 BootFlash 的程序内容，启动完成并清除 SYSCFG_MEMRMP.BFRMPEN，再跳转至用户 Flash 区域。

6.4 特殊功能寄存器

6.4.1 寄存器列表

SYSCFG 寄存器列表		
名称	偏移地址	描述
SYSCFG_PROT	000_H	系统写保护寄存器
SYSCFG_MEMRMP	004_H	存储器重映射寄存器
SYSCFG_VTOR	008_H	中断向量偏移寄存器
SYSCFG_TBKCFG	$00C_H$	TIM 刹车源配置寄存器

6.4.2 寄存器描述

6.4.2.1 系统写保护寄存器（SYSCFG_PROT）

系统写保护寄存器（SYSCFG_PROT）

偏移地址：000_H

复位值：$00000000\_00000000\_00000000\_00000001_B$

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

KEY	Bit 31-1	W	**保护关键码** 0x55AA6996：去除写保护 其他：开启写保护
PROT	Bit 0	R/W	**保护状态位** 0：无写保护 1：写保护

6.4.2.2 存储器重映射寄存器（SYSCFG_MEMRMP）

存储器重映射寄存器（SYSCFG_MEMRMP）

偏移地址：004_H

复位值：$00000000\_00000000\_00000001\_00000000_B$

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

Reserved	Bit 31-17	—	**保留**
VTOEN	Bit 16	R/W	**中断向量偏移使能位** 0：禁止 1：使能 注：偏移量为 VTOR 所设置的值
Reserved	Bits 15-9	—	**保留**
BFRMPEN	Bit 8	R/W	**Boot Flash 映射使能位** 0：禁止 1：使能
Reserved	Bit 7-0	—	**保留**

6. 4. 2. 3 中断向量偏移寄存器（SYSCFG_VTOR）

中断向量偏移寄存器（SYSCFG_VTOR）

偏移地址：008_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-30	—	保留
VTO	Bit 29-0	R/W	**中断向量偏移量** 注：最低 7 位固定为 0

6. 4. 2. 4 TIM刹车源配置寄存器（SYSCFG_TBKCFG）

TIM 刹车源配置寄存器(SYSCFG_TBKCFG)

偏移地址：$00C_H$

复位值：$00000000\_00000000\_00000001\_00000000_B$

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

Reserved	Bit 31-3	—	保留
CLUBKE	Bit 2	R/W	**CPU 锁死作为 TIM 刹车源使能位** 0：禁止 1：使能
LVDBKE	Bit 1	R/W	**LVD 事件作为 TIM 刹车源使能位** 0：禁止 1：使能
CCSBKE	Bit 0	R/W	**时钟安全事件作为 TIM 刹车源使能位** 0：禁止 1：使能

第7章 复位管理（RMU）

7.1 概述

系统复位可以由多种事件触发。这些复位事件标志可以通过读取 RMU_RSTSR 寄存器来判断复位源。

7.2 特性

- 支持 POR/BOR
- 支持外部端口复位 MRSTN
- 支持看门狗溢出复位
- 内核锁死（LOCKUP）复位
- 读取配置字错误复位（CFG_RST）
- 支持 3 种软件复位
  - 复位整个数字域
  - 复位除启动配置外的整个数字域
  - 复位内核
- 支持各外设模块的单独复位

7.3 结构框图

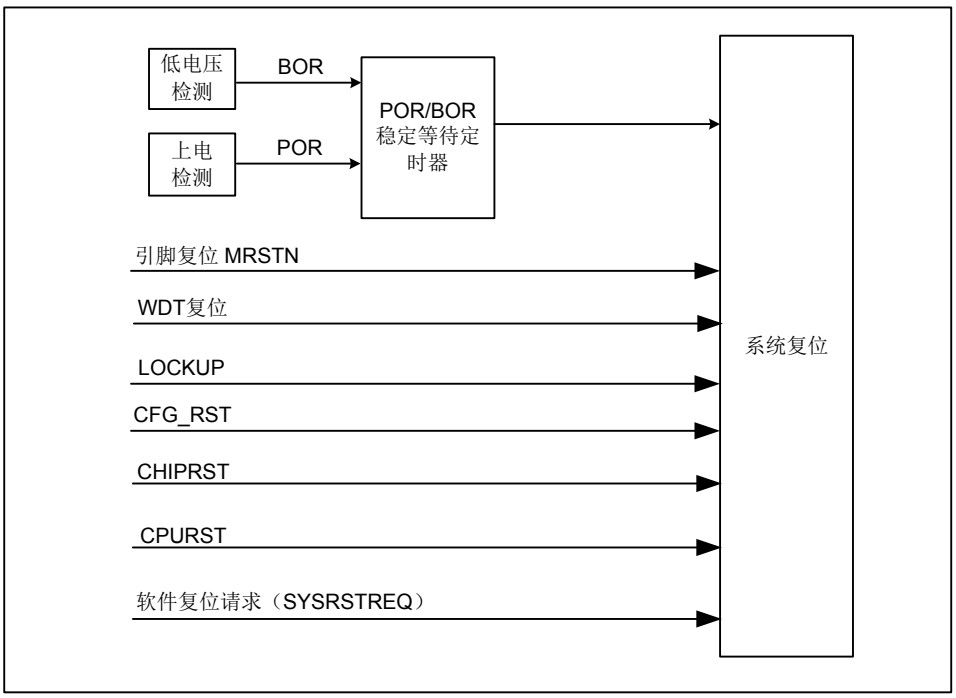


图 8-1 复位结构图

7.4 功能描述

ES32M0150 微控制器支持 8 种复位源。CPURST 只复位 CPU 内核（不包含调试部分）；其他复位源则复位 CPU 内核和所有外设。各个复位源及寄存器关系如下表所示：

	POR	BOR	MRSTN	WDT	LOCKUP	CHIPRST	SYSRST REQ	CPU RST
RMU 复位状态寄存器 (RMU_RSTSR)	POR=1	BOR=1	NMRST=1	WWDT=1 或 IWDT=1	LOCKUP=1	CHIP=1	MCU=1	CPU=1
LVDEN (LVDCR[0])	0x0	0x0	0x0	0x0	0x0	0x0	0x0	–
SYS_STU (CMU_CSR)	0x1	0x1	0x1	0x1	–	0x1	–	–
CFT_STU (CMU_CSR)	0x0	0x0	0x0	0x0	–	0x0	–	–
HRCFSW (CMU_CFGR)	0x2	0x2	0x2	0x2	–	0x2	–	–
系统和外设时钟分频	1 分频	1 分频	1 分频	1 分频	–	1 分频	–	–
HOSC_EN	0x0	0x0	0x0	0x0	0x0	0x0	0x0	–
HRC_EN	0x1	0x1	0x1	0x1	0x1	0x1	0x1	–
CMU_AHBENR	0xFFFF	0xFFFF	0xFFFF	0xFFFF	0xFFFF	0xFFFF	0xFFFF	–
CMU_APBENR	0x1000	0x1000	0x1000	0x1000	0x1000	0x1000	0x1000	–
CPU 内核调试模块	复位值	复位值	–	–	–	复位值	–	–
其他外设	复位值	复位值	复位值	复位值	复位值	复位值	复位值	–

表 8-1 系统复位与寄存器关系

7.4.1 硬件复位

硬件复位包括上电复位，欠压复位，外部端口复位，LOCKUP 复位，WDT 复位和读取配置字错误复位。

7.4.2 上电复位

芯片内部集成 POR 产生电路，当 VDD 低于指定阈值 VPOR/VPDR 时，器件无需外部复位电路便会保持复位状态。

7.4.3 欠压复位

上电期间，欠压复位（BOR）将使器件保持复位状态，直到电源电压达到由配置字设定的 VBOR 阈值。芯片支持 3 个 VBOR 阈值选择。

当电源电压（VDD）降至所选 VBOR 阈值以下时，将使器件复位。

7.4.4 端口复位

可复位除内核调试模块以外的芯片整体，复位解除后，芯片从 FLASH 启动。

7.4.5 看门狗复位

详细描述请参照独立看门狗和窗口看门狗的说明。

可复位除内核调试模块以外的芯片整体，复位解除后，芯片正常从 FLASH 启动。

7.4.6 LOCKUP复位

由不可恢复的异常导致的内核锁死，此时将产生复位信号来重新启动内核及系统。详细说明可参考 Cortex-M0 技术手册。

7.4.7 读取配置字错误复位

在配置字加载或者程序运行过程中，由于异常干扰导致配置字的读取和控制出现错误，可能导致严重系统错误，此时将产生复位并重新加载配置字。

7.4.8 软件复位

7.4.8.1 芯片复位（CHIPRST）

芯片复位由寄存器 RMU_AHB2RSTR.CHIPRST 位控制，可复位整体芯片。复位后芯片从 FLASH 空间启动。

7.4.8.2 CPU复位（CPURST）

CPU 复位由寄存器 RMU_AHB2RSTR.CPURST 位控制，可复位内核（不包含调试部分）。

7.4.8.3 内核复位请求（SYSRSTREQ）

MCU 复位从内核产生。

7.4.8.4 外设软件复位

对应每个外设分别分配了一个软件复位。

AHB1 外设复位寄存器（RMU_AHB1RSTR）为 GPIO，CRC，PIS 等模块提供软件复位。

AHB2 外设复位寄存器（RMU_AHB2RSTR）作为 CPU 复位和芯片复位的控制寄存器。

APB 外设复位寄存器（RMU_APBRSTR）为看门狗定时器，模数转换器（ADC），定时器，通信外设等 APB 模块提供软件复位。

7.5 特殊功能寄存器

7.5.1 寄存器列表

RMU 寄存器列表		
名称	偏移地址	描述
RMU_CR	000_H	RMU 控制寄存器
RMU_RSTSR	010_H	RMU 复位状态寄存器
RMU_CRSTSR	014_H	RMU 清复位状态寄存器
RMU_AHB1RSTR	020_H	AHB1 外设复位寄存器
RMU_AHB2RSTR	024_H	AHB2 外设复位寄存器
RMU_APBRSTR	030_H	APB 外设复位寄存器

7. 5. 2 寄存器描述

7. 5. 2. 1 RMU控制寄存器（RMU_CR）

RMU 控制寄存器（RMU_CR）

偏移地址：00_H

复位值：$00000000\_00000000\_00000000\_00000001_B$

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

Reserved	Bit 31-4	—	保留
BORFLT	Bits 3-1	R/W	**BOR 滤波时钟选择** 00x：1 个 LRC 周期 010：2个LRC周期 011：3个LRC周期 100：4个LRC周期 101：5个LRC周期 110：6个LRC周期 111：7个LRC周期
Reserved	Bit 0	—	保留

7. 5. 2. 2 RMU复位状态寄存器（RMU_RSTSR）

RMU 复位状态寄存器（RMU_RSTSR）

偏移地址：10_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															CFGERR	Reserved					CFG	CPU	MCU	CHIP	LOCKUP	WWDT	IWDT	NMRST	BOR	Reserved	POR

Reserved	Bit 31-17	—	保留
CFGERR	Bit 16	R	**配置字错误状态标志** 0：无配置字错误 1：产生配置字错误 注：当程序运行过程中出现配置字错误时，软件需要触发看门狗复位或芯片全局复位来复位芯片
Reserved	Bit 15-11	—	保留
CFG	Bit 10	R	**配置字复位状态标志** 0：无复位发生或标志位已被清除 1：有复位发生

CPU	Bit 9	R	**软件 CPU 复位状态标志** 0：无复位发生或标志位已被清除 1：有复位发生
MCU	Bit 8	R	**软件 MCU 复位状态标志** 0：无复位发生或标志位已被清除 1：有复位发生 注：该复位从内核产生。由应用中断和复位控制寄存器（Application Interrupt and Reset Control Register）的SYSRESETREQ位控制，将该位置1可对系统复位。详细可参考ARM相关技术手册。
CHIP	Bit 7	R	**软件 CHIP 复位状态标志** 0：无复位发生或标志位已被清除 1：有复位发生
LOCKUP	Bit 6	R	**Lockup 复位状态标志** 0：无复位发生或标志位已被清除 1：有复位发生
WWDT	Bit 5	R	**WWDT 复位状态标志** 0：无复位发生或标志位已被清除 1：有复位发生
IWDT	Bit 4	R	**IWDT 复位状态标志** 0：无复位发生或标志位已被清除 1：有复位发生
NMRST	Bit 3	R	**NMRST 复位状态标志** 0：无复位发生或标志位已被清除 1：有复位发生
BOR	Bit 2	R	**BOR 复位状态标志** 0：无复位发生或标志位已被清除 1：有复位发生
Reserved	Bit 1	—	**保留**
POR	Bit 0	R	**POR 复位状态标志** 0：无复位发生或标志位已被清除 1：有复位发生

7. 5. 2. 3 RMU清复位状态寄存器（RMU_CRSTSR）

RMU 清复位状态寄存器（RMU_CRSTSR）																													
偏移地址：14_H																													
复位值：$00000000\_00000000\_00000000\_00000000_B$																													
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																					CFG	CPU	MCU	CHIP	LOCKUP	WWDT	IWDT	NMRST	BOR	Reserved	POR

Reserved	Bit 31-11	—	保留
CFG	Bit 10	W	**配置字复位标志清除** 0：无操作 1：清除标志
CPU	Bit 9	W	**软件 CPU 复位标志清除** 0：无操作 1：清除标志
MCU	Bit 8	W	**软件 MCU 复位标志清除** 0：无操作 1：清除标志
CHIP	Bit 7	W	**软件 CHIP 复位标志清除** 0：无操作 1：清除标志
LOCKUP	Bit 6	W	**LOCKUP 复位标志清除** 0：无操作 1：清除标志
WWDT	Bit 5	W	**WWDT 复位标志清除** 0：无操作 1：清除标志
IWDT	Bit 4	W	**IWDT 复位标志清除** 0：无操作 1：清除标志
NMRST	Bit 3	W	**NMRST 复位标志清除** 0：无操作 1：清除标志
BOR	Bit 2	W	**BOR 复位标志清除** 0：无操作 1：清除标志
Reserved	Bit 1	—	保留
POR	Bit 0	W	**POR 复位标志清除** 0：无操作 1：清除标志

7. 5. 2. 4 AHB1 外设复位寄存器（RMU_AHB1RSTR）

AHB1 外设复位寄存器（RMU_AHB1RSTR）

偏移地址：10_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-6	—	**保留**
PISRST	Bit 5	R/W	**PIS 复位** 0：无操作 1：复位
Reserved	Bit4-2	—	**保留**
CRCRST	Bit 1	R/W	**CRC 复位** 0：无操作 1：复位
GPIORST	Bit 0	R/W	**GPIO 复位** 0：无操作 1：复位

7. 5. 2. 5 AHB2 外设复位寄存器（RMU_AHB2RSTR）

AHB2 外设复位寄存器（RMU_AHB2RSTR）

偏移地址：14_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-2	—	**保留**
CPURST	Bit 1	R/W	**处理器内核复位** 0：无操作 1：复位 注：该复位只复位处理器内核（不包括 DEBUG 逻辑）
CHIPRST	Bit 0	R/W	**芯片全局复位** 0：无操作 1：复位

7. 5. 2. 6 APB外设复位寄存器（RMU_APBRSTR）

APB 外设复位寄存器（RMU_APBRSTR）

偏移地址：10_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				ACMP1RST	ACMP0RST	ADCRST	DBGCONRST	IWDTRST	WWDTRST	I2C1RST	I2C0RST	Reserved			SPI0RST	Reserved		USART1RST	USART0RST	Reserved							GP16C2T2RST	GP16C2T1RST	GP16C2T0RST	BS16T0RST	AD16C4T0RST

Reserved	Bit 31-28	—	保留
ACMP1RST	Bit27	R/W	**ACMP1 复位** 0：无操作 1：复位
ACMP0RST	Bit26	R/W	**ACMP0 复位** 0：无操作 1：复位
ADCRST	Bit25	R/W	**ADC 复位** 0：无操作 1：复位
DBGCONRST	Bit24	R/W	**DBGCON 复位** 0：无操作 1：复位
IWDTRST	Bit23	R/W	**IWDT 复位** 0：无操作 1：复位
WWDTRST	Bit22	R/W	**WWDT 复位** 0：无操作 1：复位
I2C1RST	Bit 21	R/W	**I2C1 复位** 0：无操作 1：复位
I2C0RST	Bit 20	R/W	**I2C0 复位** 0：无操作 1：复位
Reserved	Bit 19-17	—	保留
SPI0RST	Bit 16	R/W	**SPI0 复位** 0：无操作 1：复位
Reserved	Bit 15-14	—	保留
USART1RST	Bit 13	R/W	**USART1 复位** 0：无操作

			1：复位
USART0RST	Bit 12	R/W	**USART0 复位** 0：无操作 1：复位
Reserved	Bit11-5	—	**保留**
GP16C2T2RST	Bit 4	R/W	**GP16C2T2 复位** 0：无操作 1：复位
GP16C2T1RST	Bit 3	R/W	**GP16C2T1 复位** 0：无操作 1：复位
GP16C2T0RST	Bit 2	R/W	**GP16C2T0 复位** 0：无操作 1：复位
BS16T0RST	Bit 1	R/W	**BS16T0 复位** 0：无操作 1：复位
AD16C4T0RST	Bit 0	R/W	**AD16C4T0 复位** 0：无操作 1：复位

第8章 时钟管理（CMU）

8.1 概述

时钟管理单元（CMU）的作用是控制时钟和振荡器。MCU 各外设时钟可独立配置。外设时钟的灵活配置可以有效降低系统功耗。

8.2 特性

- 支持多种时钟源
  - 1~24MHz 外部高速晶体振荡（HOSC）
  - 52/32/16/2MHz 可配置内部高速 RC 振荡器（HRC）
  - 32KHz 内部低速振 RC 荡器（LRC）
- 支持低功耗配置
- 系统时钟、APB 外设时钟可分别预分频
- 内核和外设均支持独立的时钟门控
- 支持系统时钟输出可配

8.3 结构框图

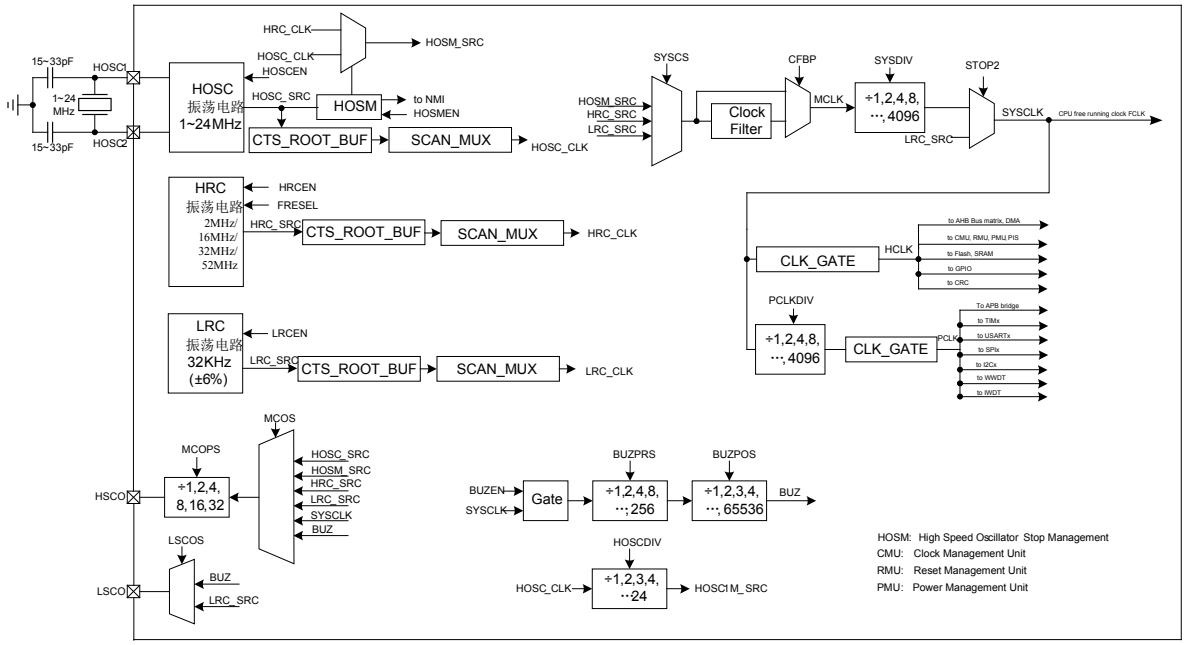


图 9-1 时钟管理结构图

8.4 功能描述

8.4.1 外部高速振荡器时钟（HOSC）

HOSC 高速振荡电路可驱动 1~24MHz 晶体振荡器和陶瓷振荡器。驱动晶体振荡器时需要匹配 15~33pF 电容，驱动陶瓷振荡器时不需要匹配电容。HOSC 内部自带反馈电阻，驱动外置振荡器不需要外接电阻。

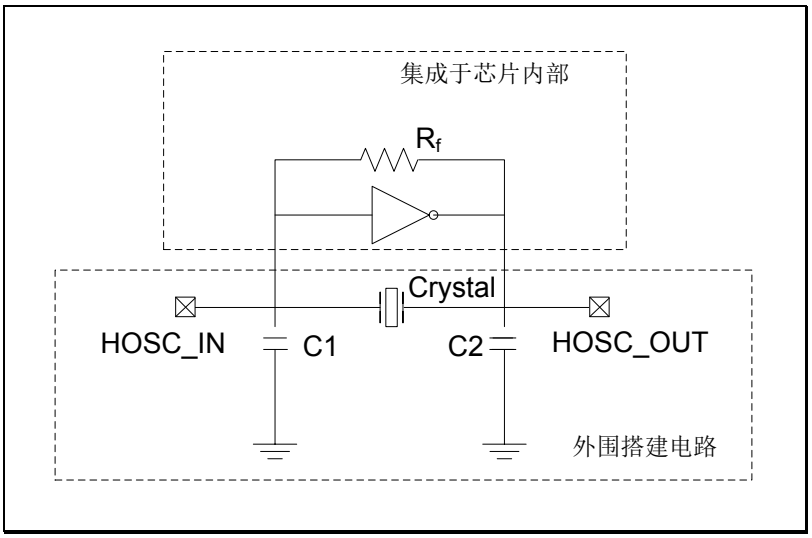


图 9-2 HOSC 电路图

8.4.2 内部高速RC振荡器时钟（HRC）

内部高速 RC 振荡器 HRC，可输出 2MHz/16MHz/32MHz/52MHz 四种频率时钟。系统默认是 32MHz。

8.4.3 内部低速RC振荡器时钟（LRC）

内部低速 RC 振荡器 LRC 可输出 32KHz 时钟。LRC 常开，不能关闭。

8.4.4 系统时钟选择

通过 CMU 控制状态寄存器（CMU_CSR）的 SYS_CMD 位进行配置选择和切换时钟。

系统时钟源有：

- ◇ HRC
- ◇ LRC
- ◇ 带安全管理的 HOSC，可配置 HOSC 停振时自动切换至 HRC
- ◇ 系统时钟可通过 CMU_CFGR 寄存器进行分频

8.4.5 时钟安全管理

高速振荡器安全管理 HOSM

HOSM 时钟安全系统可以通过软件使能（CMU_HOSMCR 寄存器中的 EN 为 1）。HOSM 时钟安全监测机制在 HOSC 振荡器启动并稳定后被激活，当软件将 HOSC 振荡器关闭时，安全监测机制也将关闭。若 HOSC 时钟发生故障，系统时钟将切换至 HRC（内部高速振

荡器时钟）。时钟故障事件会被送到高级定时器的刹车输入端，且时钟失效中断标志位（CMU_HOSMCR 寄存器中的 STPIF 位）被置位。如果中断使能（CMU_HOSMCR 的 STPIE 为 1），则产生时钟安全中断，允许软件完成营救操作。若 NMI 中断使能（CMU_HOSMCR 的 NMIE 为 1），则时钟失效事件将产生 NMI 中断(不可屏蔽中断)。

8.5 特殊功能寄存器

8.5.1 寄存器列表

CMU 寄存器列表		
名称	偏移地址	描述
CMU_CSR	000_H	CMU 控制状态寄存器
CMU_CFGR	004_H	CMU 配置寄存器
Reserved	008_H~$00C_H$	保留
CMU_CLKENR	010_H	CMU 时钟使能寄存器
CMU_CLKSR	014_H	CMU 时钟状态寄存器
Reserved	018_H	保留
CMU_HOSCCFG	$01C_H$	HOSC 配置寄存器
CMU_HOSMCR	020_H	HOSC 安全管理控制寄存器
Reserved	024_H~$02C_H$	保留
CMU_CLKOCR	030_H	CMU 时钟输出控制寄存器
CMU_BUZZCR	034_H	BUZZ 控制寄存器
Reserved	038_H~$03C_H$	保留
CMU_AHBENR	040_H	AHB 外设时钟使能寄存器
Reserved	044_H~$04C_H$	保留
CMU_APBENR	050_H	APB 外设时钟使能寄存器
Reserved	054_H~$05C_H$	保留
CMU_LPENR	060_H	外设时钟低功耗模式使能寄存器
Reserved	064_H~$09C_H$	保留

8. 5. 2 寄存器描述

8. 5. 2. 1 CMU控制状态寄存器（CMU_CSR）

CMU 控制状态寄存器（CMU_CSR）

偏移地址：000_H

复位值：$00000000\_00000000\_00000001\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						CFT_RDYN	CFT_STU	CFT_CMD								Reserved			SYS_RDYN	Reserved		SYS_STU		Reserved						SYS_CMD	

Reserved	Bit 31-26	—	**保留**
CFT_RDYN	Bit 25	R	**系统时钟滤波器切换状态位** 0：系统时钟滤波器切换完成或未发生切换动作 1：系统时钟滤波器正在切换 注：系统时钟滤波器在切换时仅需若干个系统时钟，软件读取时不保证能读到系统时钟滤波器正在切换的状态
CFT_STU	Bit 24	R	**系统时钟滤波器激活状态位** 0：系统时钟滤波器被选择 1：系统时钟滤波器被旁路
CFT_CMD	Bit 23-16	W	**系统时钟滤波切换命令位** 0x55：选择系统时钟滤波器 0xAA：旁路系统时钟滤波器 其他：无操作 注1：系统默认为选择系统时钟滤波器。 注2：当系统时钟滤波器正在切换时，该位写入无效。该位读出始终为0。 注3：当系统时钟大于32MHz时，需旁路系统时钟滤波器，否则可能会造成系统时钟失效。
Reserved	Bit 15-13	—	**保留**
SYS_RDYN	Bit 12	R	**系统时钟切换状态位** 0：系统时钟切换完成或未发生切换动作 1：系统时钟正在切换 注：系统时钟在切换时仅需若干个系统时钟，软件读取时不保证能读到系统时钟正在切换的状态
Reserved	Bit 11-10	—	**保留**
SYS_STU	Bit 9-8	R	**当前系统时钟状态位** 00：保留 01：选择HRC 10：选择LRC 11：选择HOSC
Reserved	Bit 7-2	—	**保留**

SYS_CMD	Bit 1-0	W	**系统时钟切换命令位** 00：无操作 01：选择HRC 10：选择LRC 11：选择HOSC 注1：系统默认为选择HRC。 注2：当系统时钟正在切换时，该位写入无效。该位读出始终为0。

注：当系统时钟大于 32MHz（如 48MHz）时，需旁路系统时钟滤波器，否则可能会造成系统时钟失效，其他情况下则不建议旁路系统时钟滤波器，可进一步提升系统工作稳定性。

8. 5. 2. 2 CMU配置寄存器（CMU_CFGR）

CMU 配置寄存器（CMU_CFGR）

偏移地址：004_H

复位值：$00000010\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-26	—	**保留**
HRCFSW	Bit 25-24	R/W	**HRC 频率软件选择位** 00：2MHz 01：16MHz 10：32MHz（默认） 11：52MHz
Reserved	Bit 23-20	—	保留
PCLKDIV	Bit 19-16	R/W	**PCLK 分频选择位** 0000：1分频 0001：2分频 0010：4分频 0011：8分频 …… 1100：4096分频 其他：保留 注：非1分频功能会增大HRC高频模式(32MHz以上)的时钟频率偏差约0.3%，建议对时钟精度要求高的应用保持1分频。
SYSDIV	Bit 15-12	R/W	**SYSCLK 分频选择位** 0000：1分频 0001：2分频

			0010：4分频 0011：8分频 …… 1100：4096分频 其他：保留 注：非1分频功能会增大HRC高频模式(32MHz以上)的时钟频率偏差约0.3%，建议对时钟精度要求高的应用保持1分频。
Reserved	Bit11-0	—	**保留**

8. 5. 2. 3 CMU时钟使能寄存器（CMU_CLKENR）

CMU 时钟使能寄存器（CMU_CLKENR）

偏移地址：010_H

复位值：$00000000\_00000000\_00000000\_00011100_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																					HOSCFLYBPS	HOSCBPS	HOSC1M	Reserved					HRCEN	Reserved	HOSCEN

Reserved	Bit 31-11	—	**保留**
HOSCFLYBPS	Bit 10	R/W	**HOSC Bypass Filter 使能位** 0：禁止 1：使能
HOSCBPS	Bit 9	R/W	**HOSC Bypass 使能位** 0：禁止 1：使能
HOSC1M	Bit 8	R/W	**HOSC1M 软件使能位** 0：禁止 1：使能
Reserved	Bit 7-3	—	**保留**
HRCEN	Bit 2	R/W	**HRC 软件使能位** 0：禁止 1：使能
Reserved	Bit 1	—	保留
HOSCEN	Bit 0	R/W	**HOSC 软件使能位** 0：禁止 1：使能

8. 5. 2. 4 CMU时钟状态寄存器（CMU_CLKSR）

CMU 时钟状态寄存器（CMU_CLKSR）

偏移地址：014_H

复位值：$00000000\_0000xx00\_00000000\_00000000_B$

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

Reserved	Bit 31-20	—	**保留**
LRCRDY	Bit 19	R	**LRC 稳定标志位** 0：未稳定或未激活 1：稳定
HRCRDY	Bit 18	R	**HRC 稳定标志位** 0：未稳定或未激活 1：稳定
Reserved	Bit 17	—	保留
HOSCRDY	Bit 16	R	**HOSC 稳定标志位** 0：未稳定或未激活 1：稳定
Reserved	Bit 15-3	—	**保留**
HRCACT	Bit 2	R	**HRC 激活状态位** 0：未激活 1：激活
Reserved	Bit 1	—	保留
HOSCACT	Bit 0	R	**HOSC 激活状态位** 0：未激活 1：激活

8. 5. 2. 5 HOSC配置寄存器（CMU_HOSCCFG）

HOSC 配置寄存器（CMU_HOSCCFG）

偏移地址：$1C_H$

复位值：$00000000\_00000000\_00000000\_00010111_B$

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

Reserved	Bit 31-5	—	**保留**
FREQ	Bit 4-0	R/W	**HOSC 频率配置位**

			0x00：1MHz 0x01：2MHz …… 0x16：23MHz 0x17：24MHz 其他：保留

8. 5. 2. 6 HOSC安全管理控制寄存器（CMU_HOSMCR）

HOSC 安全管理控制寄存器（CMU_HOSMCR）

偏移地址：020_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved											NMIE	STPIF	STRIF	STPIE	STRIE	Reserved					FRQS			Reserved						FLAG	EN

Reserved	Bit 31-21	—	**保留**
NMIE	Bit 20	R/W	**HOSC安全事件不可屏蔽中断使能位** 0：禁止 1：使能
STPIF	Bit 19	R/W	**HOSC停振标志位** 0：未发生 HOSC 停振或标志位已被清除 1：发生HOSC停振 注：该位写1清除，写0无效
STRIF	Bit 18	R/W	**HOSC起振标志位** 0：未发生 HOSC 起振或标志位已被清除 1：发生HOSC起振 注：该位写1清除，写0无效
STPIE	Bit 17	R/W	**HOSC停振中断使能位** 0：禁止 1：使能
STRIE	Bit 16	R/W	**HOSC起振中断使能位** 0：禁止 1：使能
Reserved	Bit 15-11	—	**保留**
FRQS	Bit 10-8	R/W	**HOSC 频率配置位** 000：1~2MHz 001：2~4MHz 010：4~8MHz 011：8~16MHz 1xx：16~24MHz 注：当频率处在两个相邻档位的临界值时，这两

			个档位可任意选择
Reserved	Bit 7-2	—	**保留**
FLAG	Bit 1	R	**HOSC 停振状态位** 0：HOSC 正常 1：HOSC停振
EN	Bit 0	R/W	**HOSC 安全管理使能位** 0：禁止 1：使能

注：在时钟管脚无外接晶振或存在外部干扰的情况下，HOSC 停振标志 STPIF，起振标志 STRIF，停振状态 FLAG 均可能会失效。

8. 5. 2. 7 CMU时钟输出控制寄存器（CMU_CLKOCR）

CMU 时钟输出控制寄存器（CMU_CLKOCR）

偏移地址：030_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-27	—	**保留**
LSCOS	Bit 26-24	R/W	**低速时钟输出源选择位** 000：BUZZ 001：LRC 其余：保留
Reserved	Bit 23-17	—	**保留**
LSCOEN	Bit 16	R/W	**低速时钟输出使能位** 0：禁止 1：使能
Reserved	Bit 15	—	**保留**
HSCODIV	Bit 14-12	R/W	**高速时钟输出分频选择位** 000：1 分频 001：2 分频 010：4 分频 011：8 分频 100：16 分频 101：32 分频 110：64 分频 111：128 分频
Reserved	Bit 11	—	**保留**
HSCOS	Bit 10-8	R/W	**高速时钟输出源选择位** 000：HOSC

			001：HOSM 010：HRC 011：LRC 100：SYSCLK 101：BUZZ 其余：保留
Reserved	Bit 7-1	—	**保留**
HSCOEN	Bit 0	R/W	**高速时钟输出使能位** 0：禁止 1：使能

8. 5. 2. 8 BUZZ控制寄存器（CMU_BUZZCR）

BUZZ 控制寄存器（CMU_BUZZCR）

偏移地址：034_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

DAT	Bit 31-16	R/W	**BUZZ频率数据值** $Freq_{BUZZ} = \frac{Freq_{SYSCLK}}{2^{DIV+1} \times (DAT+1)}$
Reserved	Bit 15-11	—	**保留**
DIV	Bit 10-8	R/W	**BUZZ 时钟分频选择位** 000：2 分频 001：4 分频 010：8 分频 011：16 分频 100：32 分频 101：64 分频 110：128 分频 111：256 分频
Reserved	Bit 7-1	—	**保留**
EN	Bit 0	R/W	**BUZZ 使能位** 0：禁止 1：使能

8. 5. 2. 9 AHB外设时钟使能寄存器（CMU_AHBENR）

AHB 外设时钟使能寄存器（CMU_AHBENR）

偏移地址：040_H

复位值：$00000000\_00000000\_00010000\_00000000_B$

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

Reserved	Bit 31-6	—	**保留**
PISEN	Bit 5	R/W	**PIS 时钟使能位** 0：禁止 1：使能
Reserved	Bit 4-2	—	**保留**
CRCEN	Bit 1	R/W	**CRC 时钟使能位** 0：禁止 1：使能
GPIOEN	Bit 0	R/W	**GPIO 时钟使能位** 0：禁止 1：使能

8. 5. 2. 10 APB外设时钟使能寄存器（CMU_APBENR）

APB 外设时钟使能寄存器（CMU_APBENR）

偏移地址：050_H

复位值：$00000000\_00000000\_00010000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				ACMP1EN	ACMP0EN	ADCEN	DBGCEN	IWDTEN	WWDTEN	I2C1EN	I2C0EN	Reserved			SPI0EN	Reserved		USART1EN	USART0EN	Reserved							GP16C2T2EN	GP16C2T1EN	GP16C2T0EN	BS16T0EN	AD16C4T0EN

Reserved	Bit 31-28	—	**保留**
ACMP1EN	Bit27	R/W	**ACMP1 时钟使能位** 0：禁止 1：使能
ACMP0EN	Bit26	R/W	**ACMP0 时钟使能位** 0：禁止 1：使能
ADCEN	Bit25	R/W	**ADC 时钟使能位** 0：禁止 1：使能

DBGCEN	Bit24	R/W	**DBGC 时钟使能位** 0：禁止 1：使能
IWDTEN	Bit23	R/W	**IWDT 时钟使能位** 0：禁止 1：使能
WWDTEN	Bit22	R/W	**WWDT 时钟使能位** 0：禁止 1：使能
I2C1EN	Bit 21	R/W	**I2C1 时钟使能位** 0：禁止 1：使能
I2C0EN	Bit 20	R/W	**I2C0 时钟使能位** 0：禁止 1：使能
Reserved	Bit 19-17	—	**保留**
SPI0EN	Bit 16	R/W	**SPI0 时钟使能位** 0：禁止 1：使能
Reserved	Bit 15-14	—	**保留**
USART1EN	Bit 13	R/W	**USART1 时钟使能位** 0：禁止 1：使能
USART0EN	Bit 12	R/W	**USART0 时钟使能位** 0：禁止 1：使能（默认）
Reserved	Bit11-5	—	**保留**
GP16C2T2EN	Bit 4	R/W	**GP16C2T2 时钟使能位** 0：禁止 1：使能
GP16C2T1EN	Bit 3	R/W	**GP16C2T1 时钟使能位** 0：禁止 1：使能
GP16C2T0EN	Bit 2	R/W	**GP16C2T0 时钟使能位** 0：禁止 1：使能
BS16T0EN	Bit 1	R/W	**BS16T0 时钟使能位** 0：禁止 1：使能
AD16C4T0EN	Bit 0	R/W	**AD16C4T0 时钟使能位** 0：禁止 1：使能

第9章 DMA控制器（DMA）

9.1 概述

DMA 控制器可独立于 CPU 对内部存储进行操作，利用 DMA 可减轻 CPU 的负荷、提升系统效率。每个 DMA 通道可选择芯片上的任意外设资源，实现数据的搬运、传输及控制。

9.2 特性

该模块的控制逻辑提供以下特性:

- 支持 4 个独立的 DMA 通道
- 仲裁到达的请求
- 指示有效通道
- 指示通道完成
- 指示 AHB-Lite 接口上发生错误
- 使能低速外设，用来拖延 DMA 周期
- 完成一个 DMA 周期前，等待清零请求
- 进行多个或单个 DMA 传输
- 进行以下不同类型的 DMA 传输（安全模式不支持 FLASH 数据 DMA 传输）
  - ◇ 存储器到存储器
  - ◇ 存储器到外设
  - ◇ 外设到存储器

注： 因每个通道只提供单个 DMA 请求接口，因此不支持外设到外设传输。

9.3 结构框图

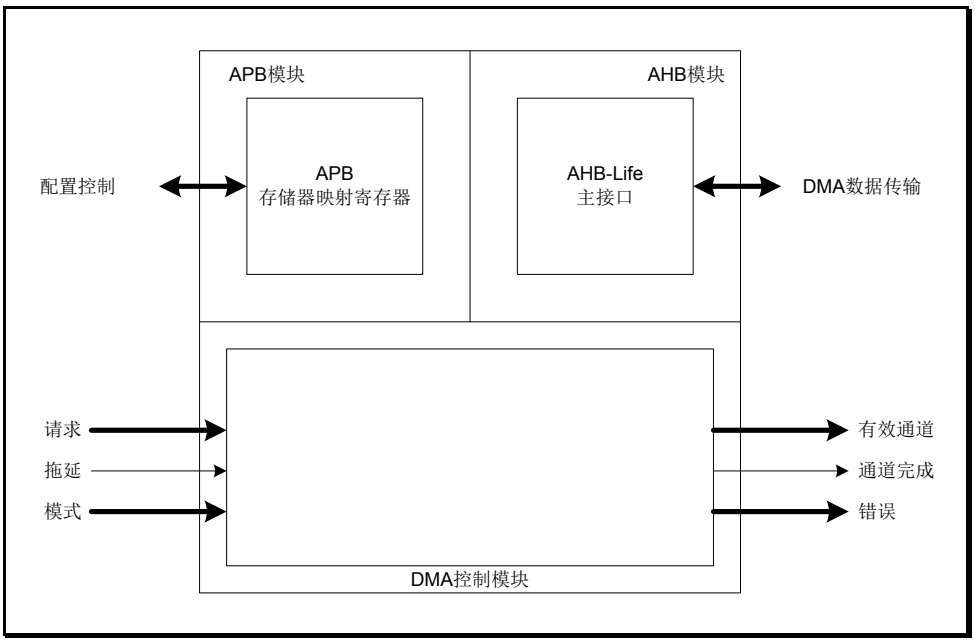


图 10-1 DMA 结构框图

9.4 功能描述

以下章节介绍了控制器的操作功能。

- ◇ APB 从机接口
- ◇ AHB 主机接口
- ◇ DMA 控制
- ◇ 通道控制数据结构

9.4.1 APB从机接口

APB 从机接口可连接控制器和 APB，并提供一个可访问寄存器的主控控制器。

APB 从机接口支持以下特性

- ◇ 读写字访问
- ◇ 32 位数据总线接口

因 APB 从机接口不支持等待状态和错误响应，因此该控制器实施 AMBA 2.0 APB 协议。更多关于 APB 接口的详细描述，请参考 AMBA 规格（Rev 2.0）。

9.4.2 AHB主机接口

AHB 主机接口具有以下特性:

- ◇ 不同的传输类型
- ◇ 传输数据宽度
- ◇ 保护控制
- ◇ 地址增量

9.4.3 DMA控制

该章节介绍了以下内容:

- ◇ 握手规则
- ◇ DMA 信号
- ◇ DMA 仲裁率
- ◇ 优先级
- ◇ DMA 周期类型
- ◇ 错误信号

9. 4. 3. 1 握手规则

DMA 握手规则适用于以下条件:

◇ 使能通道，即 CHNL_ENABLE_SET[C]和 MASTER_ENABLE 置 1

◇ 不屏蔽 dma_req[C]和 dma_sreq[C]请求输入，即 CHNL_REQ_MASK_SET[C]清 0。

规则	功能描述
1	当 dma_active[C]为低电平，dma_req[C] or dma_sreq[C]维持 1 个或多个 hclk 周期高电平(连续或非连续)可在通道 C 上启动数据传输
2	控制器仅允许单个 dma_active[]为高电平
3	当通道 C 启动传输时，dma_active[C]置为高电平
4	对于不同 DMA 周期类型，除了外设分散-聚集(scatter-gather)模式外，在控制器完成 2^R 或 $n\_minus\_1^a$ (两者中，选择较小值)的传输前，dma_active[C]保持高电平。在外设分散-聚集模式下，控制器先使用主要数据结构(primary data structure)进行 2^R 次传输，不进行仲裁，接着使用交替数据结构(alternate data structure)进行 2^R 或 $n\_minus\_1^a$ 次传输(两者中，选择较小值)。当完成交替数据结构的传输后，dma_active[C]将变成低电平。
5	在设置 dma_active[C]或 dma_active[ ]为高电平前，dma_active[C]必须设置为低电平至少 1 个 hclk 周期。
6	对于使能的通道，仅单个 dma_done[ ]可置为高电平。
7	当 dma_active[C]或 dma_stall 为高电平时，若 dma_req[C]为高电平，则 dma_req[C]在上一个周期为低电平的情况下，控制器才能检测到请求信号。
8	若某一通道的 cycle_ctrl 设置为 3’b100，3’b101，3’b110 或 3’b111，则 dma_done[C]将不会被置为高电平。
9	当某一通道上的数据传输已全部完成，并且 cycle_ctrl 将 dma_done[ ]置高，则在 dma_active[ ]的下降沿会产生以下动作： - 若 dma_stall 为低电平，dma_done[ ]将置高 1 个 hclk 周期。 - 若 dma_stall 为高电平，将延迟控制器。当 dma_stall 变为低电平，则控制器将置高 dma_done[ ]1 个 hclk 周期。
10	dma_waitonreq[C]的状态仅能在 C 通道禁止的情况下改变。
11	当 dma_waitonreq[C]为高电平，则在下列条件发生前，dma_active[C]会一直保持高电平 - 控制器完成了 2^R 或 $n\_minus\_1^a$ 次传输 - dma_req[C]为低电平 - dma_sreq[C]为低电平
12	若在 hclk 周期上，设置 dma_active[C]为低电平前，设置 dma_stall 为高电平，则: - 在下一个 hclk 周期，控制器设置 dma_active 为低电平 - 在 dma_stall 设置为低电平前，通道 C 才会完成传输。
13	当 dma_waitonreq[C]为低电平，控制器将无视 dma_sreq[C]。
14	当 CHNL_USEBURST_SET[C]为 1^b, 控制器将无视 dma_sreq[C]。
15	除了外设分散-聚集模式，对于其他 DMA 周期类型，当完成 2^R 次传输时，若剩余的传输次数小于 2^R，CHNL_USEBURST_SET[C]的读取值将被置为 0。 在外设分散-聚集模式下，若剩余的使用交替数据结构的传输次数小于 2^R，则 CHNL_USEBURST_SET[C]的读取值将被置为 0
16	除了外设分散-聚集模式，对于其他 DMA 周期类型,当 dma_sreq[C]和 dma_waitonreq[C]为高电平，且 dma_req[C]为低电平，在 dma_active[C]在 hclk 周期上变为高电平之前，控制

规则	功能描述
	器将会进行一次 DMA 传输，在外设分散-聚集模式下，当 dma_sreq[C]和 dma_waitonreq[C]为高电平，且 dma_req[C]为低电平，在 dma_active[C]在 hclk 周期上变为高电平之前，控制器会先完成 2^R 次主要数据结构的传输，且不进行仲裁，接着再完成一次交替数据结构的 DMA 传输。
17	除了外设分散-聚集模式，在 DMA 其他周期模式下，当 dma_req[C]和 dma_sreq[C]为高电平，在 dma_active[C]在 hclk 周期上变为高电平前，先发出 dma_req[C]请求，接着控制器进行 2^R 或 $n\_minus\_1^a$ 次传输。 在外设分散-聚集模式下，当 dma_sreq[C]和 dma_sreq [C]为高电平，在 dma_active[C]在 hclk 周期上变为高电平之前，先发出 dma_req[C]请求，控制器进行 2^R 次主要数据结构的传输，且不进行仲裁，接着再完成 2^R 或 $n\_minus\_1^a$ 次(两者中，选择较小值)交替数据结构的 DMA 传输。
18	当 CHNL_REQ_MASK_SET[C]为 1，控制器将无视 dma_req[C]和 dma_sreq[C]的请求。
19	当 dma_req[C]为高电平，则 dma_done[C]被置为高电平。在这种情况下，即便通道在禁止的状态，控制器也会向主机的处理器发送请求。
20	当 dma_sreq[C]为高电平，若 dma_waitonreq[C]为高电平且 CHNL_USEBURST_SET[C]为 0，则 dma_done[C]置为高电平。在这种情况下，即便通道在禁止的状态，控制器也会向主机的处理器发送请求。
21	dma_active[C]始终为低电平。

表 10-1 DMA 握手规则

注 1： a: n_minus_1 在 channel_cfg 的地址中。

注 2： b: 需谨慎设置该比特位。当 n_minus_1 小于 2^R，控制器不会将 CHNL_USEBURST_SET 清零，因此 dma_req[C]请求将被屏蔽。若外设不将 dma_req[C]置高，控制器将不会进行剩余的传输。请参考 Channel useburst set。

9. 4. 3. 2 DMA信号

该章节通过握手规则对以下请求进行了详细描述。

◇ 脉冲请求

◇ 电平请求

◇ 完成信号

◇ 等待请求信号

脉冲请求

下图为当外设使用脉冲信号时，DMA 的请求时序图。

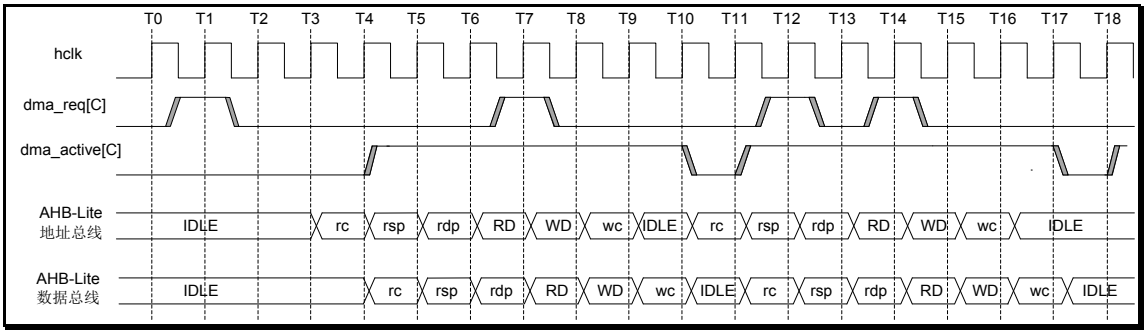


图 10-2 DMA 信号（当外设使用脉冲请求）

	描述
T1	若 CHNL_REQ_MASK_SET[C]为 0（参考规则 18），控制器会在通道 C 上检测到一个请求（参考规则 1）。
T4	控制器将 dma_active[C]置为有效（参考规则 2 和规则 3）并在通道 C 上启动 DMA 传输。
T4-T7	读取数据结构： **rc**: 读取通道配置，channel_cfg **rsp**: 读取源数据的结束指针，src_data_end_ptr **rdp**: 读取目标数据结束指针，dst_data_end_ptr
T7	若 dma_active[C]为高电平且 CHNL_REQ_MASK_SET[C]为 0（参考规则 18），通道 C 上会检测到一个请求信号（参考规则 7）。该请求将存在于下一个仲裁过程中。
T7-T9	在通道 C 上进行 DMA 传输 **RD**: 读取数据 **WD**: 写入数据
T9-T10	写通道配置寄存器 channel_cfg wc: 写通道配置，channel_cfg
T10	置 dma_active[C]为无效，表明 DMA 传输已完成（参考规则 4）
T10-T11	dma_active[C]需维持低电平至少一个 hclk 周期（参考规则 5）
T11	若通道 C 上的请求为最高优先级，则由于 T7 请求，dma_active[C]将置为有效（参考规则 2 和规则 3）。
T12	若 dma_active[C]为高电平且 CHNL_REQ_MASK_SET[C]为 0（参考规则 18），通道 C 上将检测到一个请求信号（参考规则 7）。该请求将存在于下一个仲裁过程中。

	描述
T14	因 T12 请求挂起，控制器将无视通道 C 上的请求。
T17	置 dma_active[C]为无效，表明 DMA 传输已完成（参考规则 4）
T17-T18	dma_active[C]需维持低电平至少一个 hclk 周期（参考规则 5）。
T18	若通道 C 上的请求为最高优先级，则由于 T12 请求，dma_active[C]将置为有效（参考规则 2 和规则 3）。

电平请求

下图为当外设使用电平信号时，DMA 的请求时序图。

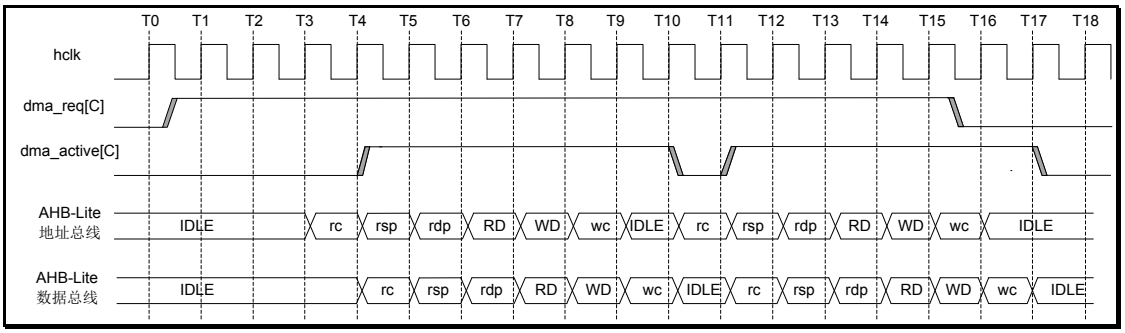


图 10-3 DMA 信号（当外设使用电平请求）

	描述
T1	若 CHNL_REQ_MASK_SET[C]为 0（参考规则 18），控制器会在通道 C 上检测到一个请求（参考规则 1）。
T4	控制器将 dma_active[C]置为有效（参考规则 2 和规则 3）并在通道 C 上启动 DMA 传输。
T4-T7	读取数据结构: **rc**: 读取通道配置，channel_cfg **rsp**: 读取源数据的结束指针，src_data_end_ptr **rdp**: 读取目标数据结束指针，dst_data_end_ptr
T7	若 dma_active[C]为高电平且 CHNL_REQ_MASK_SET[C]为 0（参考规则 18），通道 C 上会检测到一个请求信号（参考规则 7）。该请求将存在于下一个仲裁过程中。
T7-T9	在通道 C 上进行 DMA 传输 **RD**: 读取数据 **WD**: 写入数据
T9-T10	写通道配置寄存器 channel_cfg wc: 写通道配置，channel_cfg
T10	置 dma_active[C]为无效，表明 DMA 传输已完成（参考规则 4）
T10-T11	dma_active[C]需维持低电平至少一个 hclk 周期（参考规则 5）
T11	若通道 C 上的请求为最高优先级，则由于 T7 请求，dma_active[C]将置为有效（参考规则 2 和规则 3）。
T11-T14	读取数据结构
T14-T16	在通道 C 上进行 DMA 传输。
T15-T16	外设确认 DMA 传输已开始并且置 dma_req[C]为无效。
T16-T17	写通道配置 channel_cfg
T17	置 dma_active[C]为无效，表明 DMA 传输已完成（参考规则 4）。

使用电平请求信号时，如果外设不要求额外的 DMA 传输，但因时间太短不能取消请求，则 dma_stall 必须置为有效，可防止控制器结束当前传输（参考规则 7）。

完成信号

下图为当以下条件满足的情况下，dma_done[ ]信号图。

◇ dma_stall 和 dma_waitonreq[ ]为低电平

◇ dma_stall 为高电平

◇ dma_waitonreq[ ]为高电平

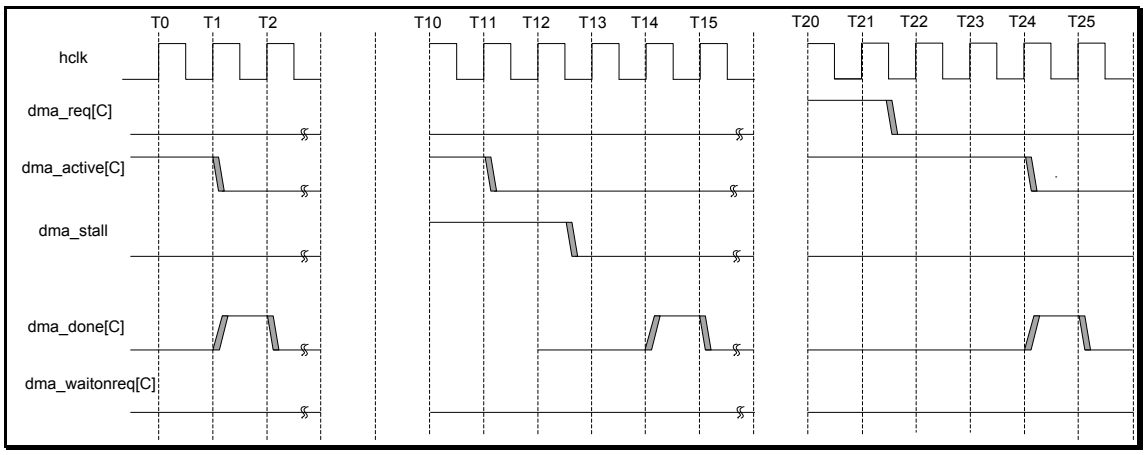


图 10-4 DMA 完成信号

	描述
T1	置 dma_active[C]为无效，表明 DMA 传输已完成（参考规则 4）。
T1-T2	完成 DMA 周期，且若 cycle_ctrl[2]为 0，dma_done[C]将保持有效一个 hclk 周期（参考规则 8 和规则 9）。对于其他使能的通道，dma_done[ ]为低电平（参考规则 6）。
T11	置 dma_active[C]为无效，表明 DMA 传输已完成（参考规则 4）。 注: 由于 dma_stall 在上一个 hclk 周期上为高电平，因此 dma_done[C]不能置为有效（参考规则 9 和规则 12）。
T12-T13	外设将 dma_stall 置为无效。
T14-T15	完成 DMA 周期，且若 cycle_ctrl[2]为 0，dma_done[C]将保持有效一个 hclk 周期（参考规则 8 和规则 9）。对于其他使能的通道，dma_done[ ]为低电平（参考规则 6）。
T20	当控制器已完成DMA传输，但由于dma_waitonreq[C]为高电平，因此只有等dma_req[C]变为低电平后，才能将 dma_active[C] (参考规则 11)和 dma_done[C]分别置为无效和有效（参考规则 9）。
T21-T22	外设将 dma_req 置为无效。
T24	置 dma_active[C]为无效，表明 DMA 传输已完成（参考规则 4）。
T24-T25	完成 DMA 周期，且若 cycle_ctrl[2]为 0，dma_done[C]将保持有效一个 hclk 周期（参考规则 8 和规则 9）。对于其他使能的通道，dma_done[ ]为低电平（参考规则 6）。

等待请求信号

下图分别举例说明如何利用等待请求信号进行 2^R 次和单次传输。

◇ 仅使用 dma_waitonreq 时的 DMA 信号

◇ 使用 dma_waitonreq 和 dma_sreq 时的 DMA 信号

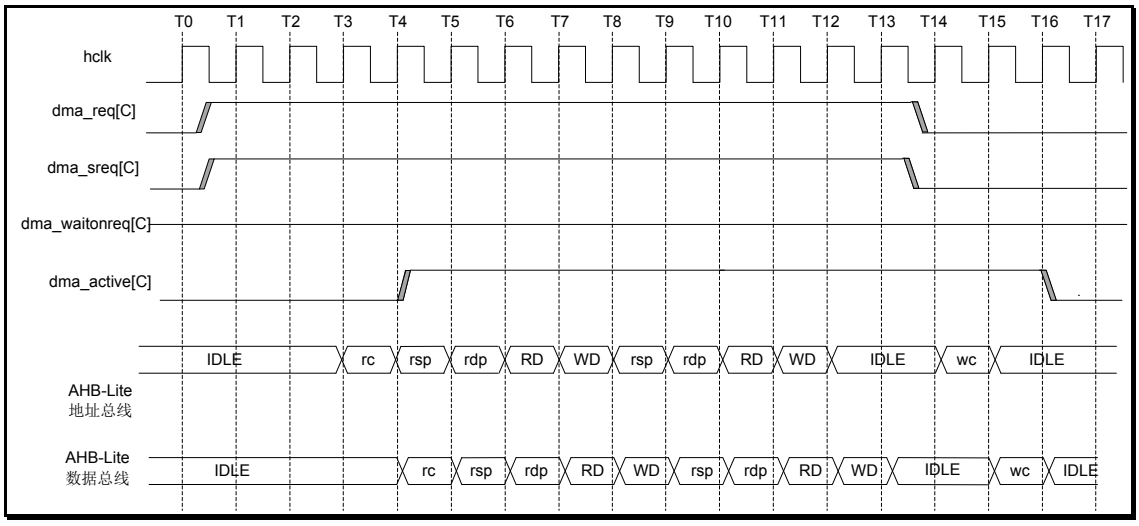


图 10-5 DMA 使用 dma_waitonreq 时序图

	描述
T0-T16	外设必须保持 dma_waitonreq[C]的状态不变（参考规则 10）。
T0-T1	若 CHNL_REQ_MASK_SET[C]为 0（参考规则 18），通道 C 上会检测到请求信号（参考规则 1）。
T3-T4	dma_req[ ]和 dma_sreq[ ]保持为高电平。控制器会回应 dma_req[ ]请求，但忽略 dma_sreq[ ]请求（参考规则 16 和规则 17）。
T4	dma_active[C]置为有效（参考规则 2 和规则 3）且启动通道 C 上的 DMA 传输。
T4-T7	读取数据结构: **rc**: 读取通道配置，channel_cfg **rsp**: 读取源数据的结束指针，src_data_end_ptr **rdp**: 读取目标数据结束指针，dst_data_end_ptr
T7-T9	在通道 C 上进行 DMA 传输 **RD**: 读取数据 **WD**: 写入数据
T9-T11	读取两个结束指针地址，rsp 和 rdp。
T11-T13	通道 C 上进行 DMA 传输。在该例子中，R = 1，因此进行 2^1 = 2 次 DMA 传输。
T13-T14	dma_req[C]和 dma_sreq[C]置为无效。

	描述
T15-T16	控制器写通道配置 **wc**：写通道配置， channel_cfg
T16	置 dma_active[C]为无效，表明 DMA 传输已完成（参考规则 11）。 如果剩余的传输次数小于 2^R（参考规则 15），将 CHNL_USEBURST_SET[C]的读取值写 0。

下图为当 dma_waitonreq[ ]为高电平，进行单次 DMA 传输时的 DMA 信号图。

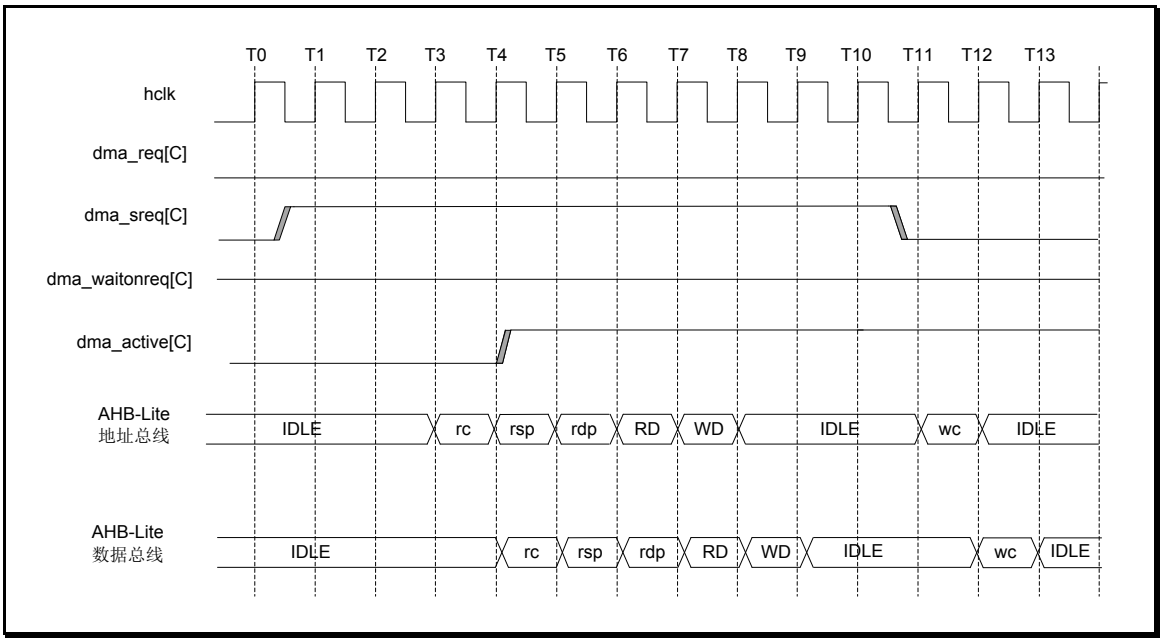


图 10-6 DMA 信号（当外设使用 dma_waitonreq 和 dma_sreq 请求）

	描述
T0-T13	外设必须保持 dma_waitonreq[C]的状态不变（参考规则 10）。
T0-T1	若 chnl_useburst_ set[C]为 0（参考规则 13 和规则 14），通道 C 上会检测到请求信号（参考规则 1）。
T3-T4	控制器对 dma_sreq[ ]请求作出回应（参考规则 16）
T4	dma_active[C]置为有效（参考规则 2 和规则 3）且启动通道 C 上的 DMA 传输。
T4-T7	读取数据结构: **rc**：读取通道配置，channel_cfg **rsp**：读取源数据的结束指针，src_data_end_ptr **rdp**：读取目标数据结束指针，dst_data_end_ptr
T7-T9	在通道 C 上进行 DMA 传输

	描述
	RD：读取数据 **WD**：写入数据 R = 0，只进行 2^0 = 1 次 DMA 传输。
T10-T11	外设置 dma_sreq[C]为无效。
T12-T13	控制器写通道配置 **wc**：写通道配置，channel_cfg。
T13	控制器置 dma_active[C]为无效，表明 DMA 传输已完成。

DMA 仲裁率

在 DMA 传输过程中，用户可配置控制器何时进行仲裁。该动作可缩短响应高优先通道的延时。

控制器提供 4 个比特位，R_power，用来配置在重新进行仲裁前，AHB 总线传输的次数。其中 R 为 2 的次方，2^R，决定了仲裁率。当 R = 4，则仲裁率为 2^4 = 16，即每进行 16 次 DMA 传输就进行一次仲裁。

R_power	x 次 DMA 传输后进行仲裁
b0000	x = 1
b0001	x = 2
b0010	x = 4
b0011	x = 8
b0100	x = 16
b0101	x = 32
b0110	x = 64
b0111	x = 128
b1000	x = 256
b1001	x = 512
b1010-b1111	x = 1024

表 10-2 仲裁设置

注：不要给优先级低的通道赋一个较大的 R_power 值，这样会导致在重新进行仲裁前，控制器不会响应优先级高的请求。

当 N > 2^R 且 N 不是 2^R 的整数倍，控制器会依次先完成 2^R 次传输直到剩余的次数 N 小于

2^R。剩余的次数 N 会在 DMA 周期的最后完成。

用户可将 R_power 的值保存在通道控制数据结构中。关于 R_power 在数据结构中的具体位置，请参考控制数据配置。

优先级

当控制器进行仲裁的时候，下一个进行仲裁的通道可由以下信息决定:

- 通道数
- 通道优先级

每个通道都可以通过 CHNL_PRIORITY_SET 寄存器设置为默认优先级或者高优先级。

通道 0 的优先级最高。随着通道数增高，优先级随之降低。

通道数	优先级设置	优先级(由高到低)
0	高	最高优先级
1	高	-
2	高	-
-	高	-
-	高	-
-	高	-
30	高	-
31	高	-
0	默认	-
1	默认	-
2	默认	-
-	默认	-
-	默认	-
-	默认	-
30	默认	-
31	默认	最低优先级

表 10-3 DMA 通道优先级

当完成一个 DMA 传输，控制器将轮询所有的 DMA 通道。下方为轮询流程图。

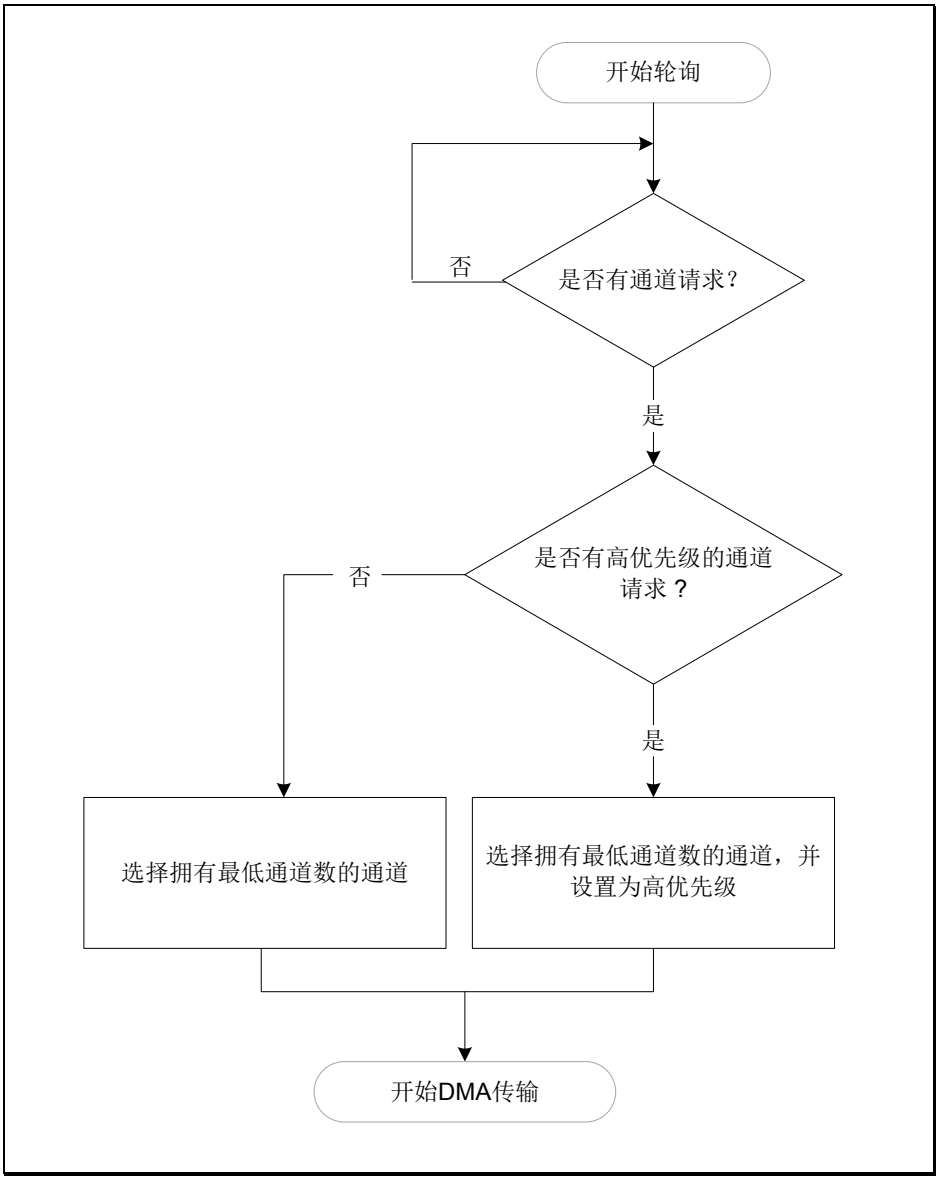


图 10-7 轮询流程图

DMA 周期类型

控制器可通过 cycle_ctrl 来选择 DMA 周期的类型，请参考下表。

cycle_ctrl	功能描述
b000	通道控制数据结构无效
b001	基础 DMA 传输
b010	自动请求
b011	乒乓
b100	存储器分散-聚集(主要数据结构)
b101	存储器分散聚集(交替数据结构)

cycle_ctrl	功能描述
b110	外设分散-聚集(主要数据结构)
b111	外设分散聚集(交替数据结构)

表 10-4 DMA 周期类型

注: cycle_ctrl 位于 channel_cfg 存储地址。

在所有的周期模式下，控制器在完成 2^R 次传输后进行仲裁。若一个低优先级通道被赋予了一个较大的 2^R 值，则到该通道完成 DMA 传输之前，所有其它的通道都不会进行 DMA 传输。因此，用户需谨慎设置 R_power 的值，避免增加高优先级通道的延时。

无效模式

当完成一个 DMA 传输后，控制器会将周期类型设置为无效，避免控制器发送同一个 DMA 周期。

基础模式

在此模式下，用户可以设置控制器使用主要数据结构或者交替数据结构。当通道使能后且控制器接收到请求信号，接下来的流程如下

1. 控制器展开 2^R 次传输。如果剩余的传输次数为 0，则继续步骤 3。
2. 控制器仲裁如下:
   - 若高优先级通道有请求信号，则控制器先响应该通道
   - 若外设或者软件发出请求信号，则继续进行步骤 1
3. 控制器将 dma_done[C]置高一个 hclk 周期，向主控处理器表明该 DMA 周期已完成。

自动请求

在自动请求模式下，仅需要收到一次请求信号就可以完成整个 DMA 周期。无须大幅度增加回应高优先级请求的延时，也无须多次向处理器或者外设发出请求，就可以完成大数据的传输。

用户可以配置控制器使用主要数据结构或者交替数据结构。当通道使能后且控制器接收到该通道上的请求，则流程如下:

1. 控制器展开 2^R 次传输。如果剩余的传输次数为 0，则继续步骤 3。
2. 控制器进行仲裁。当通道 C 的优先级最高，则继续进行步骤 1。
3. 控制器将 dma_done[C]置高一个 hclk 周期，向主控处理器表明该 DMA 周期已完成。

乒乓

在乒乓模式下，控制器先使用其中一种数据结构完成一个 DMA 周期，接着再使用另外一种数据结构完成一次 DMA 周期。控制器将会继续转换这两种数据结构直到读到无效的数据结构或者通道被主控处理器禁止。

下图显示为在乒乓模式下的 DMA 传输。

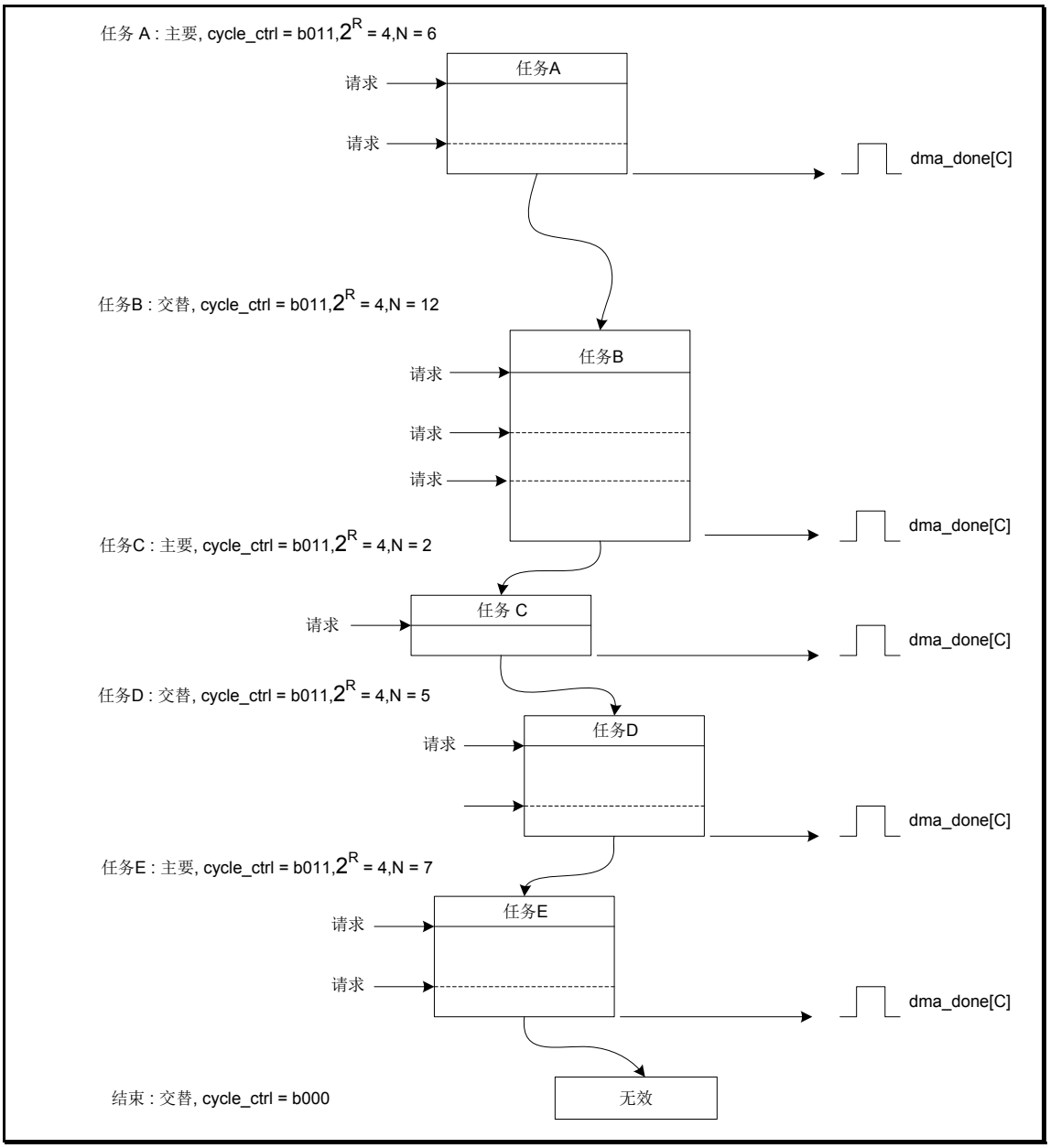


图 10-8 乒乓示例

任务 A

1. 主控处理器配置任务 A 为主要数据结构。
2. 主控处理器配置任务 B 为交替数据结构。当任务 A 完成时，控制器会立刻转换去任务 B，前提是没有高优先级通道提出请求。
3. 控制器接收到一个请求并完成 4 次 DMA 传输。
4. 控制器进行仲裁。当控制器接收到该通道上的请求，如果该通道具有最高优先级，则继续流程。
5. 控制器进行剩余的 2 次 DMA 传输。
6. 控制器将 dma_done[C]置高一个 hclk 周期，并进入仲裁流程。

当任务 A 完成后，主控处理器可配置任务 C 为主要数据结构。在任务 B 完成之后，控制器可立即转换去任务 C，前提是没有高优先级通道提出请求。

当控制器接收到该通道上新的请求且拥有最高优先级，则任务 B 开始执行:

任务 B

7. 控制器进行 4 次 DMA 传输。
8. 控制器进行仲裁。当控制器接收到该通道上的请求，如果该通道具有最高优先级，则继续流程。
9. 控制器进行 4 次 DMA 传输。
10. 控制器进行仲裁。当控制器接收到该通道上的请求，如果该通道具有最高优先级，则继续流程。
11. 控制器进行剩余的 4 次 DMA 传输。
12. 控制器将 dma_done[C]置高一个 hclk 周期，并进入仲裁流程。

当任务 B 完成后，主控处理器可将任务 D 配置为交替数据结构。

当控制器接收到该通道上新的请求且拥有最高优先级，则任务 C 开始执行:

任务 C

13. 控制器进行 2 次 DMA 传输。
14. 控制器将 dma_done[C]置高一个 hclk 周期，并进入仲裁流程。

当任务 C 完成后，主控处理器可将任务 E 配置为主要数据结构。

当控制器接收到该通道上新的请求且拥有最高优先级，则任务 D 开始执行:

任务 D

15. 控制器进行 4 次 DMA 传输。
16. 控制器进行仲裁。当控制器接收到该通道上的请求，如果该通道具有最高优先级，则继续流程。
17. 控制器展开剩余的 DMA 传输。

18. 控制器将 dma_done[C]置高一个 hclk 周期，并进入仲裁流程。

当控制器接收到该通道上新的请求且拥有最高优先级，则任务 E 开始执行:

任务 E

19. 控制器进行 4 次 DMA 传输。
20. 控制器进行仲裁。当控制器接收到该通道上的请求，如果该通道具有最高优先级，则继续流程。
21. 控制器进行剩余的 3 次 DMA 传输。
22. 控制器将 dma_done[C]置高一个 hclk 周期，并进入仲裁流程。

若控制器接收到该通道上的新请求且拥有最高优先级，则控制器会企图开始下一个任务。但由于主控处理器尚未配置交替数据结构，且当任务 D 完成的时候 cycle_ctrl 被设置为 b000，则会终止乒乓 DMA 传输。

注: 用户可通过设置 cycle_ctrl 为 b001 将任务 E 配置为基础 DMA 周期，以用来终止乒乓 DMA 周期。

存储器分散-聚集

在存储器分散-聚集模式下，控制器先接收到一个初始请求，接着采用主要数据结构进行 4 次 DMA 传输。当传输完成时，会以交替数据结构开始一个新的 DMA 周期。当该周期完成时，控制器将会采用主要数据结构开始又一轮 4 次 DMA 传输。若发生以下任意一种情况，则控制器将停止主要数据结构和交替数据结构的转换:

- 主控处理器将交替数据结构配置为一个基础周期
- 读取到一个无效数据结构

注: 当完成 N 次主要数据结构的传输后，可设置 cycle_ctrl 为 b000 使其变为无效的数据结构。

当完成一个基础周期的分散-聚集模式传输，控制器仅将 dma_done[C]置为有效。

在分散-聚集模式下，控制器利用主要数据结构来编程交替数据结构。下表主要数据结构的 channel_cfg 配置，分为固定值配置和用户配置。

位	域	值	功能描述
固定值			
[31:30]	dst_inc	b10	配置控制器使用字作为地址增量
[29:28]	dst_size	b10	配置控制器使用字传输
[27:26]	src_int	b10	配置控制器使用字作为地址增量
[25:24]	src_size	b10	配置控制器使用字传输
[17:14]	R_power	b1101	配置控制器进行 4 次 DMA 传输
[3]	next_usebusrt	0	当配置为存储器分散-聚集模式时，该位必须设置为 0
[2:0]	cycle_ctrl	b100	配置控制器进行分散-聚集 DMA 周期
用户定义			
[23:21]	dst_prot_ctrl	-	当写入目标数据后，配置 HPROT 的状态
[20:18]	src_prot_ctrl	-	当写入源数据后，配置 HPROT 的状态
[13:4]	n_minus_1	N[a]	配置控制器进行 N 次 DMA 传输，其中 N 为 4 的倍数

表 10-5 存储器分散-聚集数据结构

注 a: 由于 R_power 设计为 4，因此 N 必须设置为 4 的倍数。N/4 的值为配置交替数据结构的次数。

下图为存储器分散-聚集模式的示例:

初始化:

1. 配置主要数据结构来使能复制 A，B，C 的操作：cycle_ctrl = b100,2^R = 4,N = 16
2. 利用下表中的结构，将主要源数据写入存储器。

	src_data_end_ptr	dst_data_end_ptr	channel_cfg	Unused
任务 A 数据	0x0A000000	0x0AE00000	Cycle_ctrl = b101,2^R = 4,N = 3	0xXXXXXXXX
任务 B 数据	0x0B000000	0x0BE00000	Cycle_ctrl = b101,2^R = 2,N = 8	0xXXXXXXXX
任务 C 数据	0x0C000000	0x0CE00000	Cycle_ctrl = b101,2^R = 8,N = 5	0xXXXXXXXX
任务 D 数据	0x0D000000	0x0DE00000	Cycle_ctrl = b001,2^R = 4,N = 4	0xXXXXXXXX

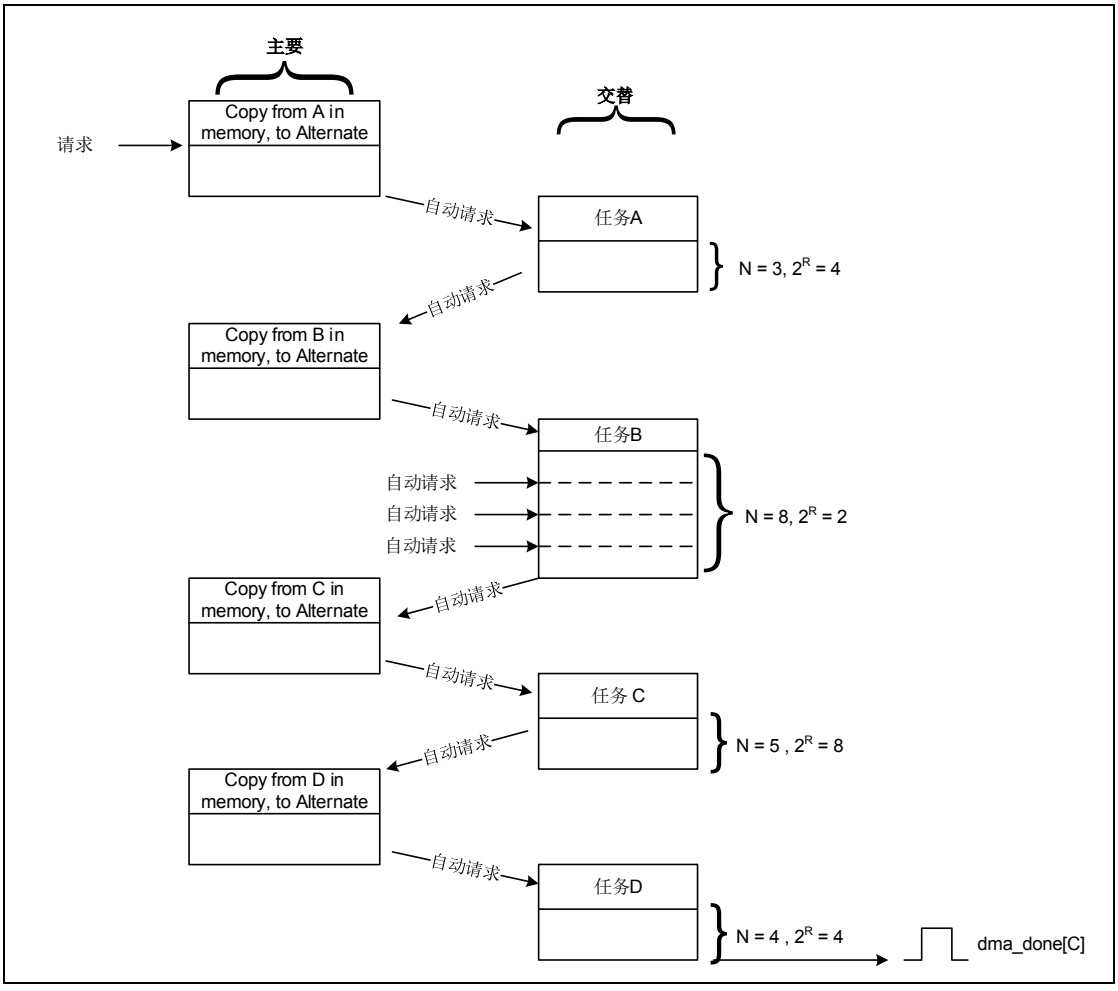


图 10-9 存储器分散-聚集示例

初始化

1. 主控处理器通过设置 cycle_ctrl 为 b100，使主要数据结构运行于存储器分散-聚集模式。 由于单个通道的数据结构包含 4 个字，所以 2^R 必须设置为 4。在该示例中，有 4 个任务，因此 N 设为 16。

2. 主控处理器将任务 A, B, C 和 D 的数据结构写入由 src_data_end_ptr 指定的存储器地址中。

3. 主控处理器使能该通道。

当控制器接收到 dma_req[ ]请求或者来自主控处理器的手动请求，则存储器分散-聚集传输开始执行，流程如下:

主要，复制 A

1. 在接收到请求后，控制器进行 4 次 DMA 传输，并且任务 A 写为交替数据结构。

2. 控制器生成一个自动请求接着进行仲裁。

任务 A

3. 控制器进行任务 A。当任务 A 完成，控制器生成一个自动请求，接着进行仲裁。

主要，复制 B

4. 控制器进行 4 次 DMA 传输，且任务 B 写为交替数据结构。

5. 控制器生成一个自动请求接着进行仲裁。

任务 B

6. 控制器进行任务 B。当任务 B 完成，控制器生成一个自动请求，接着进行仲裁。

主要，复制 C

7. 控制器进行 4 次 DMA 传输，且任务 C 写为交替数据结构。

8. 控制器生成一个自动请求接着进行仲裁。

任务 C

9. 控制器进行任务 C。当任务 C 完成，控制器生成一个自动请求，接着进行仲裁。

主要，复制 D

10. 控制器进行 4 次 DMA 传输，且任务 D 写为交替数据结构。

11. 控制器设置主要数据结构的 cycle_ctrl 为 b000，表明该数据结构为无效。

12. 控制器生成一个自动请求接着进行仲裁。

任务 D

13. 控制器使用基础 DMA 周期执行任务 D。

14. 控制器将 dma_done[C]置高一个 hclk 周期，并进入仲裁流程。

外设分散-聚集

在外设分散-聚集模式下，控制器接收到一个来自外设的初始请求，接着使用主要数据结构执行 4 次 DMA 传输，然后立即使用交替数据结构启动新的 DMA 周期，无须重新仲裁也无须将 dma_active[C]置为低电平。

注: 仅在该状况下，当完成主要数据结构的传输后，控制器无须进入仲裁流程。

在该 DMA 周期完成后，控制器重新仲裁。如果控制器接收到外设请求且拥有最高优先级，则执行 4 次主要数据结构的 DMA 传输。接着立即启动交替数据结构的 DMA 周期，无须重新仲裁也无须将 dma_active[C]置为低电平。若发生以下任意一种情况，则控制器将停止主要数据结构和交替数据结构的转换:

- 主控处理器将交替数据结构配置为一个基础周期
- 读取到一个无效数据结构

注: 当完成 N 次主要数据结构的传输后，可设置 cycle_ctrl 为 b000 使其变为无效的数据结构。

当完成一个基础周期的分散-聚集模式传输，控制器将 dma_done[C]置为有效。

在分散-聚集模式下，控制器利用主要数据结构来编程交替数据结构。下表主要数据结构的 channel_cfg 配置，分为固定值配置和用户配置。

位	域	值	功能描述
固定值			
[31:30]	dst_inc	b10	配置控制器使用字作为地址增量
[29:28]	dst_size	b10	配置控制器使用字传输
[27:26]	src_int	b10	配置控制器使用字作为地址增量
[25:24]	src_size	b10	配置控制器使用字传输
[17:14]	R_power	b1101	配置控制器进行 4 次 DMA 传输
[2:0]	cycle_ctrl	b100	配置控制器进行外设分散-聚集 DMA 周期
用户定义			
[23:21]	dst_prot_ctrl	-	当写入目标数据后，配置 HPROT 的状态
[20:18]	src_prot_ctrl	-	当写入源数据后，配置 HPROT 的状态
[13:4]	n_minus_1	N[a]	配置控制器进行 N 次 DMA 传输，其中 N 为 4 的倍数
[3]	next_usebusrt	-	当设置为 1，在交替传输完成后，控制器会将 CHNL_USEBURST_SET[C]置 1。

表 10-6 外设分散-聚集数据结构

注 a: 由于 R_power 设计为 4，因此 N 必须设置为 4 的倍数。N/4 的值为配置交替数据结构的次数。

下图为外设分散-聚集模式的示例:

初始化:

1. 配置主要数据结构来使能复制 A，B，C 的操作：cycle_ctrl = b110,2^R = 4,N = 16
2. 利用下表中的结构，将主要源数据写入存储器。

	src_data_end_ptr	dst_data_end_ptr	channel_cfg	Unused
任务 A 数据	0x0A000000	0x0AE00000	Cycle_ctrl = b101,2^R = 4,N = 3	0xXXXXXXXX
任务 B 数据	0x0B000000	0x0BE00000	Cycle_ctrl = b101,2^R = 2,N = 8	0xXXXXXXXX
任务 C 数据	0x0C000000	0x0CE00000	Cycle_ctrl = b101,2^R = 8,N = 5	0xXXXXXXXX
任务 D 数据	0x0D000000	0x0DE00000	Cycle_ctrl = b001,2^R = 4,N = 4	0xXXXXXXXX

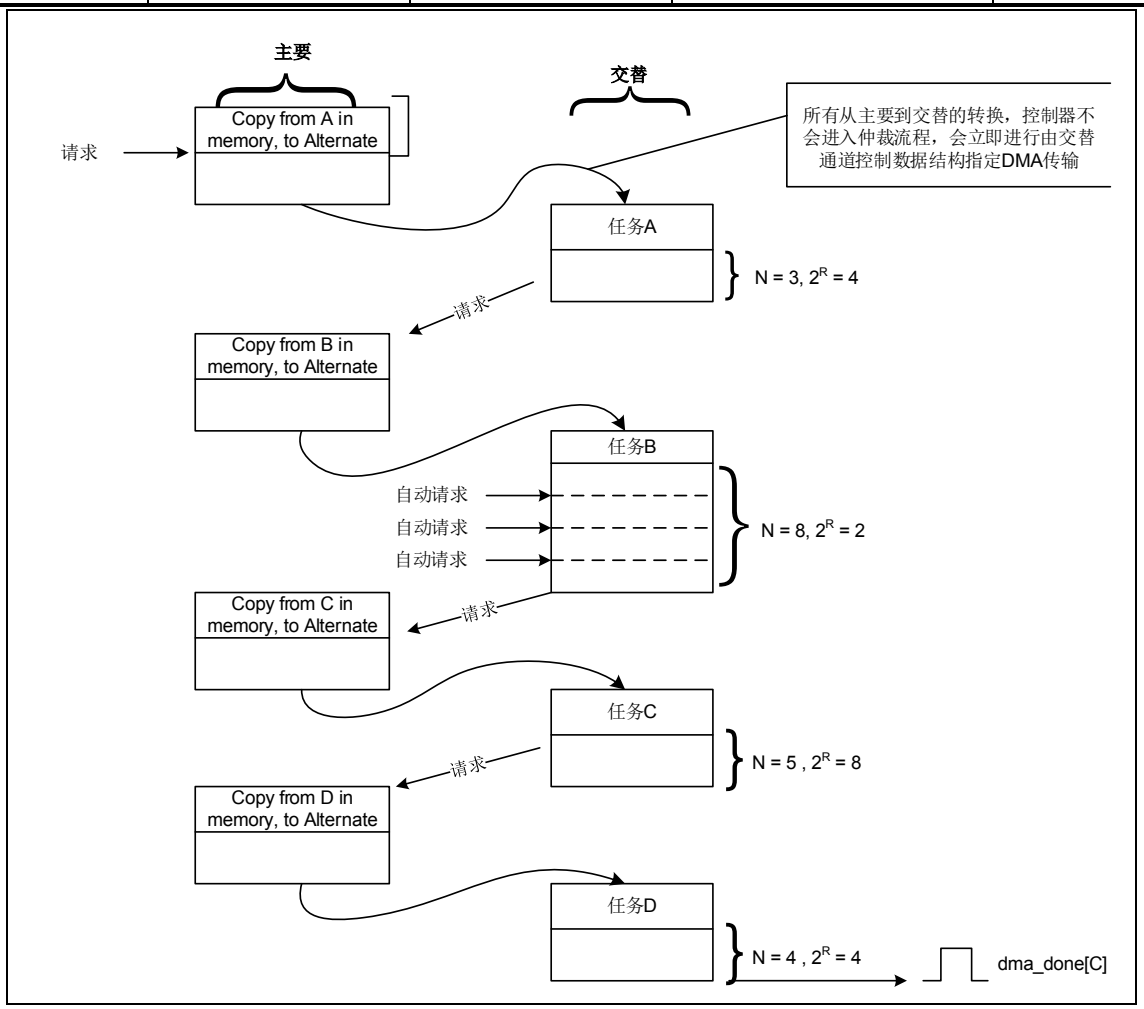


图 10-10 外设交替-聚集示例

初始化

1. 主控处理器通过设置 cycle_ctrl 为 b110，使主要数据结构运行于外设分散-聚集模式。由于单个通道的数据结构包含 4 个字，所以 2^R 必须设置为 4。在该示例中，有 4 个任务，因此 N 设为 16。

2. 主控处理器将任务 A, B, C 和 D 的数据结构写入由 src_data_end_ptr 指定的存储器地址中。

3. 主控处理器使能该通道。

当控制器接收到 dma_req[ ]请求，则外设分散-聚集传输开始执行，流程如下:

主要，复制 A

1. 在接收到请求后，控制器进行 4 次 DMA 传输，并且任务 A 写为交替数据结构。

任务 A

2. 控制器进行任务 A。

3. 当完成任务 A 后，控制器进入仲裁流程。

在外设发出一个新的请求且拥有最高优先级，则继续以下流程:

主要，复制 B

4. 控制器进行 4 次 DMA 传输，且任务 B 写为交替数据结构。

任务 B

5. 控制器执行 4 次 DMA 传输。为使控制器完成该任务，外设还必须再发出 3 次请求。

6. 当任务 B 完成，进度仲裁流程。

在外设发出一个新的请求且拥有最高优先级，则继续以下流程:

主要，复制 C

7. 控制器进行 4 次 DMA 传输，且任务 C 写为交替数据结构。

任务 C

8. 控制器执行任务 C。

9. 当任务 C 完成，进入仲裁流程。

在外设发出一个新的请求且拥有最高优先级，则继续以下流程:

主要，复制 D

10. 控制器进行 4 次 DMA 传输，且任务 D 写为交替数据结构。

11. 控制器设置主要数据结构的 cycle_ctrl 为 b000，表明该数据结构为无效。

任务 D

12. 控制器使用基础 DMA 周期执行任务 D。

13. 控制器将 dma_done[C]置高一个 hclk 周期，并进入仲裁流程。

错误信号

如果控制器在 AHB-Lite 主机接口上检测到一个错误回应，则:

◇ 禁止该错误信号对应的通道

◇ 置 dma_err 为高电平

1. 读取 CHNL_ENABLE_SET 寄存器，获取一张禁止通道的列表。

当某通道将 dma_done[ ]置为有效，则控制器将禁止该通道。主控处理器上的程序必须记录哪些通道曾置高 dma_done[ ]输出。

2. 将步骤 1 中得到的列表和主程序上的记录作比较。若某通道没有 dma_done[C]被置高的记录，则就是该通道发生了错误信号。

9. 4. 4 通道控制数据结构

用户必须提供系统存储器空间用来包含通道控制数据结构。该系统存储器必须:

◇ 提供一个连续的存储空间，以便控制器和主控处理器可以访问

◇ 其基地址为通道控制数据结构总容量的整数倍

下图为当使用 32 个通道和交替数据结构时，控制器所需要的通道控制数据的存储器映射。

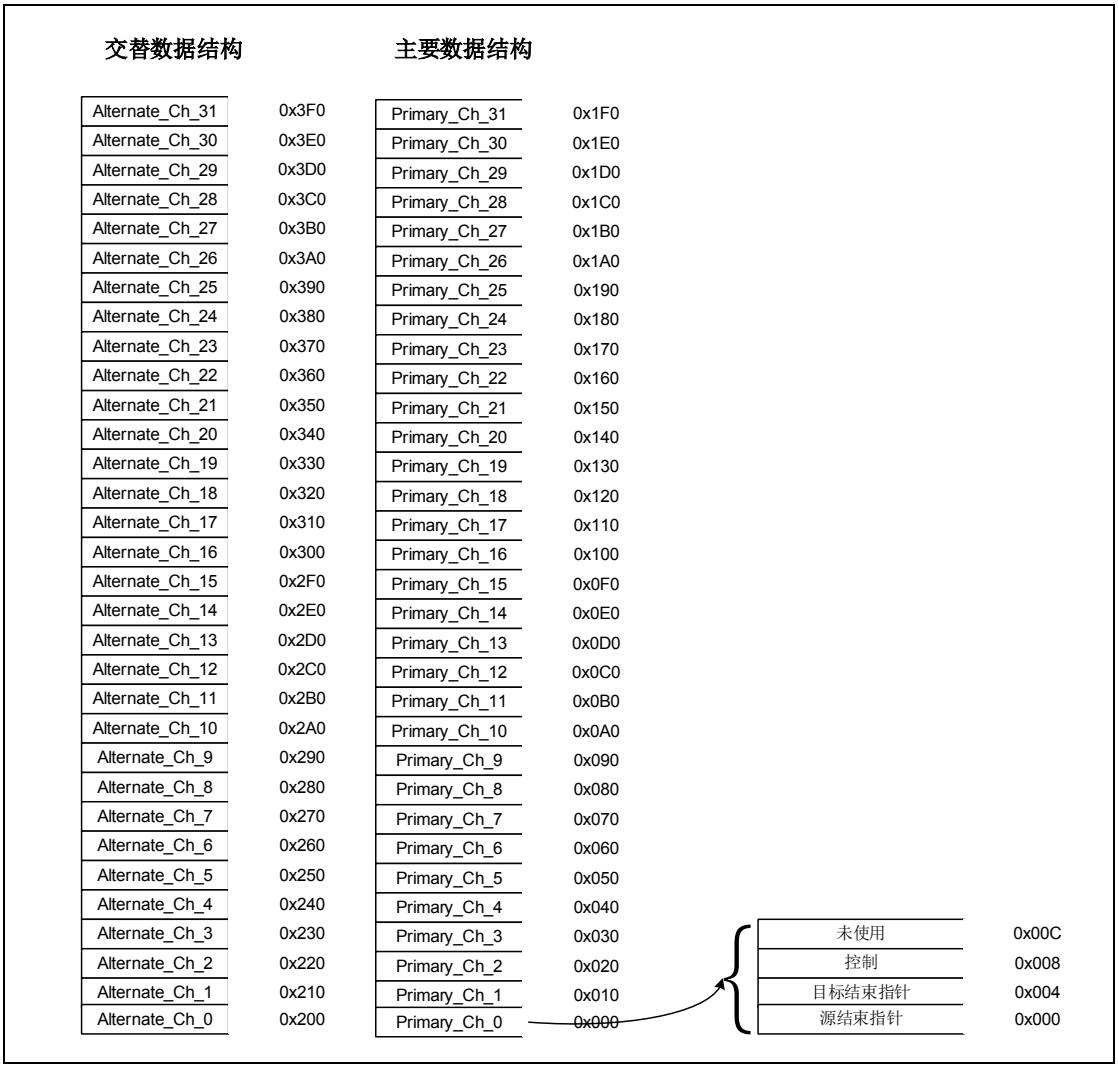


图 10-11 32 通道存储器映射（包括交替数据结构）

图中的通道数据结构使用了 1KB 的系统存储器。该示例中，控制器用地址低 10 位来访问数据结构中的所有地址，因此基地址必须为 0xXXXXX000, 0xXXXXX400, 0xXXXXX800 或者 0xXXXXXC00.

通过准确设置 CTRL_BASE_PTR 寄存器，用户可配置主要数据结构的基地址。

所需的系统存储器大小取决于以下几个条件:

◇ 控制器所需要用的 DMA 通道个数

◇ 若 DMA 通道需要配置使用交替数据结构，请参考通道主要-交替设置。

DMA 通道数	[9]	[8]	[7]	[6]	[5]	[4]	[3:0]
1						A	0x0, 0x4 或者 0x8
2					A	C[0]	
3-4				A	C[1]	C[0]	
5-8			A	C[2]	C[1]	C[0]	
9-16		A	C[3]	C[2]	C[1]	C[0]	
17-32	A	C[4]	C[3]	C[2]	C[1]	C[0]	

A 选择通道控制数据结构

A = 0 选择主要数据结构

A = 1 选择交替数据结构

C[x:0] 选择 DMA 通道

地址[3:0] 选择以下任一

0x0 选择源数据结束指针

0x4 选择目标数据结束指针

0x8 选择控制数据配置

0xC 控制器不访问该地址。若有需求，可使主控处理器将该地址用作系统存储器

注: 用户无需计算交替数据结构的基地址，该信息可由 alt_ctrl_ptr 寄存器提供。

下图显示的存储器映射图为控制器使用 3 个 DMA 通道和交替数据结构。

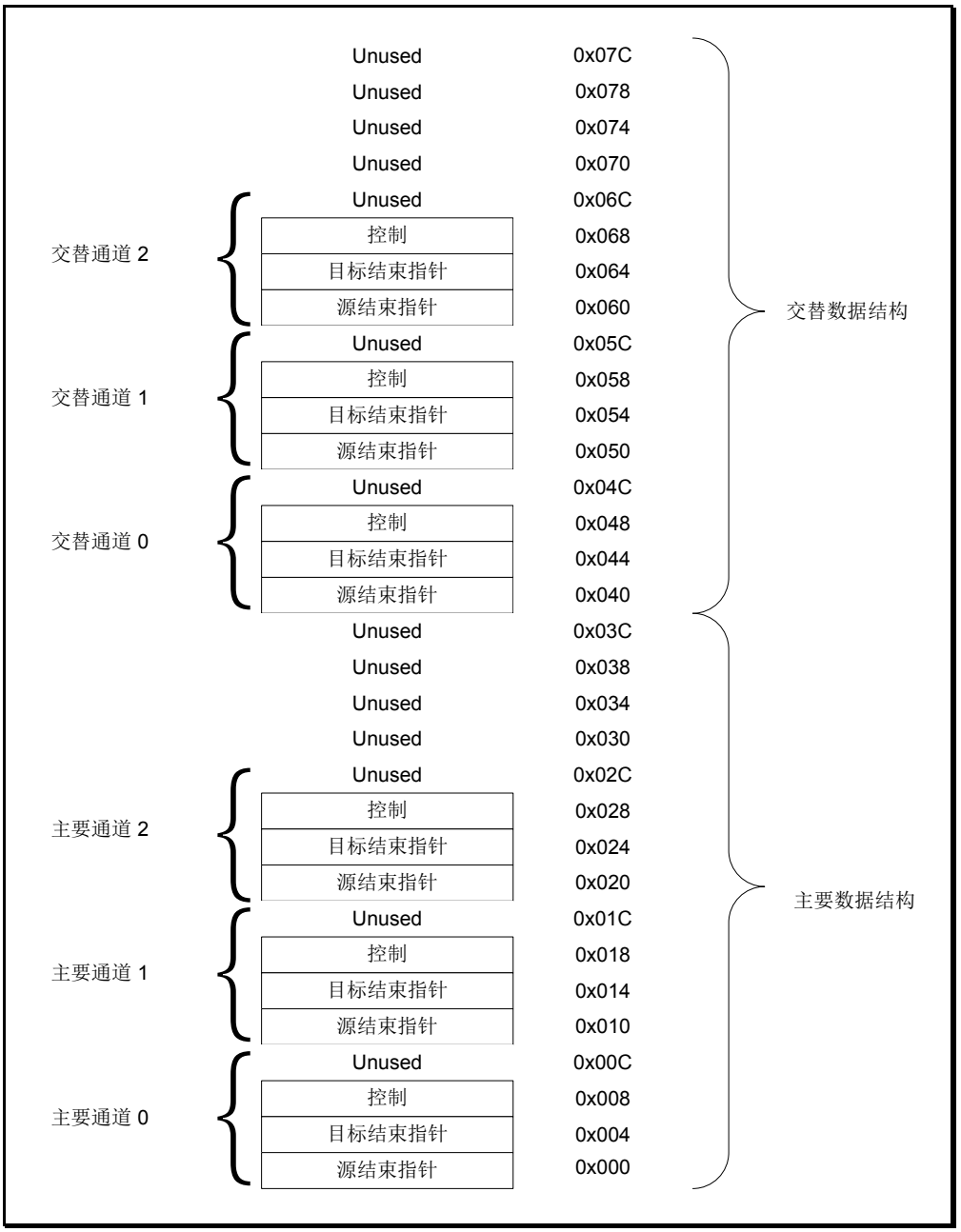


图 10-12　3 DMA 通道存储器映射（包括交替数据结构）

上图显示的系统存储器有 128 个字节。在该示例中，控制器使用地址低 6 位来访问数据结构中的所有地址，因此基地址必须为 0xXXXXXX00 或者 0xXXXXXX80.

下表列出了针对主要数据结构可用的基地址。

DMA 通道数	可用基地址 (主要数据结构)
1	0xXXXXXX00, 0xXXXXXX20, 0xXXXXXX40, 0xXXXXXX60, 0xXXXXXX80, 0xXXXXXXA0, 0xXXXXXXC0, 0xXXXXXXE0
2	0xXXXXXX00, 0xXXXXXX40, 0xXXXXXX80, 0xXXXXXXC0
3-4	0xXXXXXX00, 0xXXXXXX80
5-8	0xXXXXX000, 0xXXXXX100, 0xXXXXX200, 0xXXXXX300, 0xXXXXX400, 0xXXXXX500, 0xXXXXX600, 0xXXXXX700, 0xXXXXX800, 0xXXXXX900, 0xXXXXXA00, 0xXXXXXB00, 0xXXXXXC00, 0xXXXXXD00, 0xXXXXXE00, 0xXXXXXF00
9-16	0xXXXXX200, 0xXXXXX400, 0xXXXXX600, 0xXXXXX800, 0xXXXXXA00, 0xXXXXXC00, 0xXXXXXE00
17-32	0xXXXXX000, 0xXXXXX400, 0xXXXXX800, 0xXXXXXC00

控制器利用系统存储器来访问两个指针及每个通道控制信息。以下内容详细描述了 32 位存储地址及 DMA 传输地址的计算方法。

- 源数据结束指针
- 目标数据结束指针
- 控制数据配置
- 地址计算

源数据结束指针

src_data_end_ptr 存储地址包含一个指针，指向源数据的最后一个地址。

在执行 DMA 传输前，该存储地址写入源数据的结束地址。当启动 2^R 次 DMA 传输时，控制器读取 src_data_end_ptr。注意控制器不能写该存储器地址。

控制数据配置

channel_cfg 会向控制器提供每一个次 DMA 传输的控制信息。

通道配置 (channel_cfg)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
dst_inc		dst_size		src_inc		src_size		dst_prot_ctrl			src_prot_ctrl			R_power				n_minus_1										next_useburst		cycle_ctrl	

dst_inc	Bit 31-30	R/W	**目标地址增量** 地址增量取决于源数据的宽度:

			源数据宽度 = 字节 b00 = 字节 b01 = 半字 b10 = 字 b11 = 无增量。地址仍然是 dst_data_end_ptr 中包含的地址。 源数据宽度 = 半字 b00 = 保留 b01 = 半字 b10 = 字 b11 = 无增量。地址仍然是 dst_data_end_ptr 中包含的地址。 源数据宽度 = 字 b00 = 保留 b01 = 保留 b10 = 字 b11 = 无增量。地址仍然是 dst_data_end_ptr 中包含的地址。
dst_size	Bit 29-28	R/W	**目标数据大小** 需要注意的是 dst_size 和 src_size 的值必须一致。
src_inc	Bit 27-26	R/W	**目标地址增量** 地址增量取决于源数据的宽度： b00 = 字节 b01 = 半字 b10 = 字 b11 = 无增量
src_size	Bit 25:24	R/W	**设置该位段用来匹配源数据** b00 = 字节 b01 = 半字 b10 = 字 b11 = 保留
dst_prot_ctrl	Bit 23-21	R/W	**当控制器写目标数据的时候，HRPOT[3:1]状态控制**

			位 bit[23] HRPOT[3]状态控制位 0 = HRPOT[3]为低电平，访问不带高速缓存。 1 = HRPOT[3]为高电平，访问带高速缓存。 bit[22] HRPOT[2]状态控制位 0 = HRPOT[2]为低电平，访问不带缓冲。 1 = HRPOT[2]为高电平，访问带缓冲。 bit[21] HRPOT[1]状态控制位 0 = HRPOT[1]为低电平，访问为用户模式。 1 = HRPOT[1]为高电平，访问为特权模式。
src_prot_ctrl	Bit 20:18	R/W	**当控制器读取源数据的时候，HRPOT[3:1]状态控制位** bit[20] HRPOT[3]状态控制位 0 = HRPOT[3]为低电平，访问不带高速缓存。 1 = HRPOT[3]为高电平，访问带高速缓存。 bit[19] HRPOT[2]状态控制位 0 = HRPOT[2]为低电平，访问不带缓冲。 1 = HRPOT[2]为高电平，访问带缓冲。 bit[18] HRPOT[1]状态控制位 0 = HRPOT[1]为低电平，访问为用户模式。 1 = HRPOT[1]为高电平，访问为特权模式。
R_power	Bit 17:14	R/W	**在控制器重新仲裁前，该位段决定了 DMA 传输可以发生的次数** b0000: 发生 1 次 DMA 传输后仲裁 b0001: 发生 2 次 DMA 传输后仲裁 b0010: 发生 4 次 DMA 传输后仲裁 b0011: 发生 8 次 DMA 传输后仲裁 b0100: 发生 16 次 DMA 传输后仲裁

			b0101: 发生 32 次 DMA 传输后仲裁 b0110: 发生 64 次 DMA 传输后仲裁 b0111: 发生 128 次 DMA 传输后仲裁 b1000: 发生 256 次 DMA 传输后仲裁 b1001: 发生 512 次 DMA 传输后仲裁 b1010-b1111: 发生 1024 次 DMA 传输后仲裁。由于最大的传输次数为 1024，由此表明 DMA 传输中无仲裁发生。
n_minus_1	Bit 13:4	R/W	在 DMA 周期开始前，该位段代表的是 DMA 周期中包含的 DMA 传输的总次数。用户须根据所需要的 DMA 周期的大小来设置该位段。该 10 位值为 DMA 传输次数减 1。 B0000000000: 1 次 DMA 传输 B0000000001: 2 次 DMA 传输 B0000000010: 3 次 DMA 传输 B0000000011: 4 次 DMA 传输 B0000000100: 5 次 DMA 传输 …… B1111111111: 1024 次 DMA 传输 在进入仲裁流程前，控制器会立即更新该位段，可使控制器保存该 DMA 周期中还需完成的剩下的 DMA 传输次数。
next_useburst	Bit 3	R/W	**当控制器在外设分散-聚集模式下，且使用交替数据结构，该位段控制 CHNL_USEBURST_SET[C]是否置 1。** 需要注意的是，在完成由交替数据结构指定的 DMA 周期前，如果剩余的传输次数小于 2^R，控制器会将 CHNL_USEBURST_SET[C]设置为 0。 Next_useburst 的设定控制了是否需要再次修改 CHNL_USEBURST_SET[C]。 在外设分散-聚集 DMA 周期模式下，当使用交替数据结构的 DMA 周期完成后，会发生以下任一情况: 0: 控制器不改变 CHNL_USEBURST_SET[C]的值。

			如果 CHNL_USEBURST_SET[C]为 0，则对于所有在外设分散-聚集模式下的剩余的 DMA 周期，控制器将回应 dma_req[ ]和 dma_sreq[ ]的请求。 1: 控制器将 CHNL_USEBURST_SET[C]置 1。因此，对于所有在外设分散-聚集模式下的剩余的 DMA 周期，控制器仅回应 dma_req[ ]的请求。
cycle_ctrl	Bit 2-0	R/W	**DMA 周期的工作模式:** b000: 停止。表明该数据结构无效。 b001: 基础。在进入仲裁流程前，控制器必须接收到一个新的请求才能完成 DMA 周期。 b010: 自动请求。在仲裁过程中，控制器在合适的通道上自动插入一个请求。这就意味着最初的请求已足够完成 DMA 周期。 b011: 乒乓。控制器使用任一数据结构完成一个 DMA 周期。在该周期完成后，且如果主控处理器已更新了原始数据结构，则控制器使用原始数据结构执行一次 DMA 周期。直到读取到无效数据结构或者主控处理器将 cycle_ctrl 改为 b001 或 b010，控制器将会继续进行 DMA 周期。 b100: 存储器分散-聚集。当控制器运行在存储器分散-聚集模式下，用户仅可在主要数据结构下使用该值。 b101: 存储器分散-聚集。当控制器运行在存储器分散-聚集模式下，用户仅可在交替数据结构下使用该值。 b110: 外设分散-聚集。当控制器运行在外设分散-聚集模式下，用户仅可在主要数据结构下使用该值。 b111: 外设分散-聚集。当控制器运行在外设分散-聚集模式下，用户仅可在主要交替结构下使用该值。

在 DMA 周期或者 2^R DMA 传输开始的时候，控制器会从系统存储器中获取 channel_cfg 的值。当完成 2^R 或 N 次传输后，新的 channel_cfg 值会被存储到系统存储器中。

控制器不支持 dst_size 和 src_size 拥有两个不同的值。如果检测到两个值不匹配，src_size 的值会被作为源数据和目标数据的大小。当下一次 n_minus_1 更新时， dst_size 将被设置为 src_size 的值。

当完成 N 次传输后，控制器会将 cycle_ctrl 设置为 b000，以此表明 channel_cfg 数据为无效，用来防止控制器重复相同的 DMA 传输。

地址计算

为了计算 DMA 传输的源地址，控制器必须将 n_minus_1 的值向左移，移动值由 src_inc

定义。接着从源数据结束指针中减去移动后的 n_minus_1 的值。同样地，为了计算 DMA 传输的目标地址，控制器也必须将 n_minus_1 的值向左移，移动量由 dst_inc 定义，接着从目标数据结束指针中减去移动后的 n_minus_1 的值。

根据 src_inc 和 dst_inc 的值，源地址和目标地址的计算可使用同一个等式。

Src_inc = b00, dst_inc = b00

源地址 = src_data_end_ptr – n_minus_1

目标地址 = dst_data_end_ptr - n_minus_1

Src_inc = b01, dst_inc = b01

源地址 = src_data_end_ptr – (n_minus_1<<1)

目标地址 = dst_data_end_ptr – (n_minus_1<<1)

Src_inc = b10, dst_inc = b10

源地址 = src_data_end_ptr – (n_minus_1<<2)

目标地址 = dst_data_end_ptr – (n_minus_1<<2)

Src_inc = b11, dst_inc = b11

源地址 = src_data_end_ptr

目标地址 = dst_data_end_ptr

下表为 6 个字的 DMA 周期，地址增量为 1 个字。

channel_cfg 初始值 (DMA 周期前)				
src_size=b10, dst_inc=b10, n_minus_1=b101, cycle_ctrl=1				
DMA 传输	**结束指针**	**次数**	**差值**	**地址**
	0x2AC	5	0x14	0x298
	0x2AC	4	0x10	0x29C
	0x2AC	3	0xC	0x2A0
	0x2AC	2	0x8	0x2A4
	0x2AC	1	0x4	0x2A8
	0x2AC	0	0x0	0x2AC
channel_cfg 最终值 (DMA 周期后)				
src_size=b10, dst_inc=b10, n_minus_1=0, cycle_ctrl=0				

注：上表中的差值为次数向左移动 dst_inc 后的结果。

channel_cfg 初始值 (DMA 周期前)				
src_size=b00, dst_inc=b01, n_minus_1=b1011, cycle_ctrl=1, R_power=b11				
	结束指针	**次数**	**差值**	**地址**
DMA 传输	0x5E7	11	0x16	0x5D1
	0x5E7	10	0x14	0x5D3
	0x5E7	9	0x12	0x5D5
	0x5E7	8	0x10	0x5D7
	0x5E7	7	0xE	0x5D9
	0x5E7	6	0xC	0x5DB
	0x5E7	5	0xA	0x5DD
	0x5E7	4	0x8	0x5DF
2^R 次 DMA 传输完成后，channel_cfg 的值				
src_size=b00, dst_inc=b01, n_minus_1=b011, cycle_ctrl=1, R_power=b11				
	结束指针	**次数**	**差值**	**地址**
DMA 传输	0x5E7	3	0x6	0x5E1
	0x5E7	2	0x4	0x5E3
	0x5E7	1	0x2	0x5E5
	0x5E7	0	0x0	0x5E7
channel_cfg 最终值 (DMA 周期后)				
src_size=b00, dst_inc=b01, n_minus_1=0, cycle_ctrl=0, R_power=b11				

注 1: 上表中的差值为次数向左移动 dst_inc 后的结果。

注 2: 当控制器完成 DMA 周期后，通过将 cycle_ctrl 清零使 channel_cfg 无效。

9.5 特殊功能寄存器

该章节描述了 DMAC 寄存器及提供了编程控制器的信息。

- ◇ 编程器模型介绍
- ◇ 寄存器描述

9.5.1 编程器模型介绍

下列条件适用于控制器提供的寄存器:

- ◇ 控制器的基地址并非固定不变的，但任何特定寄存器的偏移地址都是固定的。
- ◇ 用户禁止访问任何保留的地址，否则可导致控制器产生不可预期的行为。
- ◇ 除非有其他相关说明，用户必须将保留的和未使用的寄存器位写 0，无视读取值。
- ◇ 除非有其他相关说明，系统复位或者上电复位会将所有寄存器位写 0。
- ◇ 除非有其他相关说明，所有寄存器都支持读写访问。写操作可更新寄存器内容，读操作则返回寄存器内容。

9. 5. 2 寄存器列表

DMA 寄存器列表		
名称	偏移地址	描述
DMA_STATUS	0000_H	DMA 状态寄存器
DMA_CFG	0004_H	DMA 配置寄存器
DMA_CTRLBASE	0008_H	通道控制数据基指针寄存器
DMA_ALTCTRLBASE	$000C_H$	通道交替控制数据基指针寄存器
DMA_CHWAITSTATUS	0010_H	通道等待请求状态寄存器
DMA_CHSWREQ	0014_H	通道软件请求寄存器
DMA_CHUSEBURSTSET	0018_H	通道使用冲突设置寄存器
DMA_CHUSEBURSTCLR	$001C_H$	通道使用冲突清除寄存器
DMA_CHREQMASKSET	0020_H	通道请求屏蔽设置寄存器
DMA_CHREQMASKCLR	0024_H	通道请求屏蔽清除寄存器
DMA_CHENSET	0028_H	通道使能设置寄存器
DMA_CHENCLR	$002C_H$	通道使能清除寄存器
DMA_CHPRIALTSET	0030_H	通道主要-交替设置寄存器
DMA_CHPRIALTCLR	0034_H	通道主要-交替清除寄存器
DMA_CHPRSET	0038_H	通道优先级设置寄存器
DMA_CHPRCLR	$003C_H$	通道优先级清除寄存器
Reserved	0040_H~0048_H	保留
DMA_ERRCLR	$004C_H$	总线错误清除寄存器寄存器
Reserved	0050_H~$0FFC_H$	保留
DMA_IFLAG	1000_H	DMA 中断状态寄存器
Reserved	1004_H	保留
DMA_ICFR	1008_H	DMA 中断状态清零寄存器
DMA_IER	$100C_H$	DMA 中断使能控制寄存器
Reserved	1010_H~$10FC_H$	保留
DMA_CH0_SELCON	1100_H	DMA 通道 0 复用选择寄存器
DMA_CH1_SELCON	1104_H	DMA 通道 1 复用选择寄存器
DMA_CH2_SELCON	1108_H	DMA 通道 2 复用选择寄存器
DMA_CH3_SELCON	$110C_H$	DMA 通道 3 复用选择寄存器

9. 5. 3 寄存器描述

9. 5. 3. 1 DMA状态寄存器（DMA_STATUS）

该寄存器可返回控制器的状态，为只读寄存器。当控制器处于复位状态时，用户不能读取该寄存器。

DMA 状态寄存器 (DMA_STATUS)

偏移地址: 000_H

复位值: $00010000\_00001011\_00000000\_00000000_B$

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

Reserved	Bit 31-8	-	**保留**
STATUS	Bit 7-4	R	**控制状态机的当前状态** b0000: 空闲 b0001: 读取通道控制器数据 b0010: 读取源数据结束指针 b0011: 读取目标数据结束指针 b0100: 读取源数据 b0101: 写目标数据 b0110: 等待 DMA 请求清 0 b0111: 写通道控制器数据 b1000: 延迟 b1001: 完成 b1010: 外设分散-聚集转换 b1011-b1111: 未定义
Reserved	Bit 3-1	-	**保留**
MASTER_ENABLE	Bit 0	R	**控制器使能状态位** 0: 使能 1: 禁止

9. 5. 3. 2 DMA配置寄存器（DMA_CFG）

该寄存器为只写寄存器，可对控制器进行配置。

DMA 配置寄存器 (DMA_CFG)

偏移地址: 004_H

复位值: $00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-8	-	**未定义，写0。**
CHNL_PROT_CTRL	Bit 7-5	W	**通过控制 HPROT[3:1]的信号电平对 AHB-Lite 做保护设置** Bit[7]: 控制 HPROT[3]来表明是否正在发生可缓存访问 Bit[6]: 控制 HPROT[2]来表明是否正在发生可缓冲访问 Bit[5]: 控制 HPROT[1]来表明是否正在发生特殊权限访问 注意: 当 bit[n] =1，相应 HPROT 为高电平。 当 bit[n] =0，相应 HPROT 为低电平。
Reserved	Bit 4-1	-	**保留**
MASTER_ENABLE	Bit 0	W	**控制器使能位** 0: 使能 1: 禁止

9. 5. 3. 3 通道控制数据基指针寄存器（DMA_CTRLBASE）

该寄存器为读写寄存器。用户须配置该寄存器用来指向系统存储器中的地址。
需要指出的是，控制器不提供内部存储器用来存储通道控制数据结构。
控制器中用户所需指定的系统存储器的大小取决于以下两个条件：DMA 通道的个数和是否使用交替数据结构。因此，该基指针的设置取决于不同系统实现，可参考下图。
当控制器处于复位状态时，用户不可读取该寄存器。

通道控制数据基指针寄存器 (DMA_CTRLBASE)

偏移地址: 008_H

复位值: $00000000\_00000000\_00000000\_00000100_B$

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

CTRL_BASE_PTR	Bits 31- 9	R/W	主要数据结构基地址指针
Reserved	Bits 8-0	-	未定义，写 **0**。

9. 5. 3. 4 通道交替控制数据基指针寄存器（DMA_ALTCTRLBASE）

该寄存器为只读寄存器，可返回交替数据结构的基地址，因此用户无需在应用软件中计算交替数据结构的基地址。当控制器处于复位状态时，用户不可读取该寄存器。

通道交替控制数据基指针寄存器 (DMA_ALTCTRLBASE)

偏移地址: $00C_H$

复位值: $00000000\_00000000\_00000000\_00000100_B$

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

ALT_CTRL_BASE_PTR	Bit 31-0	R	**交替数据结构的基地址**

9. 5. 3. 5 通道等待请求状态寄存器（DMA_CHWAITSTATUS）

该寄存器为只读寄存器，可返回 dma_waitonreq[ ]的状态。当控制器处于复位状态时，用户不可读取该寄存器。Bit0 返回的为 dma_waitonreq[0]的状态，Bit1 返回的为 dma_waitonreq[1]的状态，以此类推，Bit31 返回的为 dma_waitonreq[31]的状态。

通道请求状态寄存器 (DMA_CHWAITSTATUS)

偏移地址: 010_H

复位值: $00000000\_00000000\_00000000\_00111111_B$

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

DMA_WAITONREQ_STATUS	Bit 31-0	R	**通道等待状态** 0: dma_waitonreq[n]为低电平 1: dma_waitonreq[n]为高电平

9. 5. 3. 6 通道软件请求寄存器（DMA_CHSWREQ）

该寄存器为只写寄存器，每个对应的 bit 都可在相应的通道上生产软件 DMA 请求。

通道软件请求寄存器 (DMA_CHSWREQ)																															
偏移地址: 014_H																															
复位值: $00000000\_00000000\_00000000\_00000000_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CHSWREQ																															

CHSWREQ	Bit 31-0	W	**通道软件请求** 0: 通道n上不生成DMA请求。 1: 在通道n上生成一个DMA请求。

9. 5. 3. 7 通道使用冲突设置寄存器（DMA_CHUSEBURSTSET）

该寄存器为可读写寄存器，可禁止单次请求 dma_sreq[ ]产生请求信号，因此仅有 dma_req[ ]产生请求。读取该寄存器可返回冲突的使用状态。每个 bit 都有相对应的通道，bit0 对应通道 0，bit1 对应通道 1，以此类推。

当完成倒数第二次 2^R 传输，如果剩余的次数 N 小于 2^R，控制器会将 CHNL_USEBURST_SET 复位成 0。剩余的传输次数可通过 dma_req[ ]或 dma_sreq[ ]完成。

注: 当 channel_cfg 的设置值 N 小于 2^R，如果外设没有将 dma_req[ ]置为有效， 则不应该将 CHNL_USEBURST_SET 置 1。

在外设分散-聚集模式下，当使用交替数据结构的 DMA 周期完成的时候，如果 channel_cfg 中的 next_useburst 已被设置为 1，控制器会将 CHNL_USEBURST_SET[C]置 1。

通道使用冲突设置寄存器 (DMA_CHUSEBURSTSET)

偏移地址: 018_H

复位值: $00000000\_00000000\_00000000\_00000000_B$

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

CHNL_USEBURST_SET	Bit 31-0	R/W	**返回冲突的使用状态，或禁止dma_sreq[C]产生DMA请求** 读取: 0: 通道C对dma_req[C]或dma_sreq[C]请求做出回应。控制器执行2^R或单次总线传输。 1: 通道C不回应dma_sreq[C]请求。控制器只回应dma_req[C]请求，执行2^R次传输。 写入: 0: 无效。使用CHNL_USEBURST_CLR寄存器将bit[C]写0。 1: 禁止dma_sreq[C]产生DMA请求。控制器执行2^R次传输。

9. 5. 3. 8 通道使用冲突清除寄存器（DMA_CHUSEBURSTCLR）

通道使用冲突清除寄存器 (DMA_CHUSEBURSTCLR)

偏移地址: $01C_H$

复位值: $00000000\_00000000\_00000000\_00000000_B$

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

CHNL_USEBURST_CLR	Bit 31-0	W	**使能dma_sreq[C]产生请求** 写入: 0: 无效。使用CHNL_USEBURST_SET寄存器禁止dma_sreq[ ]产生请求。 1: 使能dma_sreq[C]产生DMA请求。

9. 5. 3. 9 通道请求屏蔽设置寄存器（DMA_CHREQMASKSET）

该寄存器为读写寄存器，可禁止 dma_req[ ]和 dma_sreq[ ]产生请求。读取时可返回 dma_req[ ]和 dma_sreq[ ]的屏蔽状态。

通道请求屏蔽设置寄存器 (DMA_CHREQMASKSET)

偏移地址: 020_H

复位值: $00000000\_00000000\_00000000\_00000000_B$

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

CHNL_REQ_MASK_SET	Bit 31-0	R/W	**返回dma_req[ ]和dma_sreq[ ]的屏蔽状态，或禁止相应通道产生DMA请求** 读取: 0: 通道C上的外部请求已使能。 1: 通道C上的外部请求已禁止。 写入: 0: 无效。使用CHNL_REQ_MASK_CLR寄存器来使能DMA请求。 1: 禁止dma_req[C]和dma_sreq[C]产生DMA

			请求。

9. 5. 3. 10 通道请求屏蔽清除寄存器（DMA_CHREQMASKCLR）

通道请求屏蔽清除寄存器 (DMA_CHREQMASKCLR)

偏移地址: 024_H

复位值: $00000000\_00000000\_00000000\_00000000_B$

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

CHNL_REQ_MASK_CLR	Bit 31-0	W	**使能通道上的dma_req[ ]和dma_sreq[ ]请求** 写入: 0: 无效。使用chnl_req_mask _set寄存器禁止dma_req[ ]和dma_sreq[ ]产生请求。 1: 使能dma_sreq[C]或dma_sreq[ ]产生DMA请求。

9. 5. 3. 11 通道使能设置寄存器（DMA_CHENSET）

该寄存器为读写寄存器，设置该寄存器可使能通道。读取时，可返回该通道的状态。

通道使能设置寄存器 (DMA_CHENSET)																															
偏移地址: 028_H																															
复位值: $00000000\_00000000\_00000000\_00000000_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CHNL_ENABLE_SET																															

CHNL_ENABLE_SET	Bit 31-0	R/W	**返回通道的状态，或使能相应的通道** 读取: 0: 通道C禁止。 1: 通道C使能。 写入: 0: 无效。设置CHNL_ENABLE_CLR用来禁止某通道。 1: 使能相应通道。

9. 5. 3. 12 通道使能清除寄存器（DMA_CHENCLR）

该寄存器为只写寄存器，可用来禁止相应的通道。

通道使能清除寄存器 (DMA_CHENCLR)																															
偏移地址: $02C_H$																															
复位值: $00000000\_00000000\_00000000\_00000000_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CHNL_ENABLE_CLR																															

CHNL_ENABLE_CLR	Bit 31-0	W	**禁止相应DMA通道** 写入: 0: 无效。设置CHNL_ENABLE_SET寄存器来使能DMA通道。 1: 禁止相应通道。

注: 当发生以下任意一种情况时，控制器可通过设置相应 CHNL_ENABLE_CLR 来禁止通道：

1. 控制器完成 DMA 周期
2. 控制器读取 channel_cfg，其 cycle_ctrl 位段为 b000
3. AHB-Lite 总线上发生错误

9. 5. 3. 13 通道主要-交替设置寄存器（DMA_CHPRIALTSET）

该寄存器为读写寄存器。通过该寄存器，用户可将 DMA 通道设置为使用交替数据结构。读取时，可返回相应通道的数据结构使用状态。

通道主要-交替设置寄存器 (DMA_CHPRIALTSET)																															
偏移地址: 030_H																															
复位值: $00000000\_00000000\_00000000\_00000000_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CHNL_PRI_ALT_SET																															

CHNL_PRI_ALT_SET	Bit 31-0	R/W	**返回通道的控制数据结构的状态，或为相应的DMA通道选择交替数据结构** 读取: 0: 相应DMA通道使用的是主要数据结构。 1: 相应DMA通道使用的是交替数据结构。 写入: 0: 无效。设置CHNL_PRI_ALT_CLR中相应的比特位为0。 1: 为相应DMA通道选择交替数据结构。

注: 当发生以下情况时，控制器将翻转 CHNL_PRI_ALT_SET 中相应的比特位的值:

1. 在存储器/外设分散-聚集 DMA 周期模式下，完成由主要数据结构指定的 4 次传输。
2. 在乒乓 DMA 周期模式下，完成由主要数据结构指定的所有传输。
3. 在乒乓/存储器分散-聚集/外设分散-聚集 DMA 周期模式下，完成由交替数据结构指定的所有传输。

9. 5. 3. 14 通道主要-交替清除寄存器（DMA_CHPRIALTCLR）

该寄存器为只写寄存器。通过该寄存器，用户可将 DMA 通道设置为使用主要数据结构。

通道主要-交替清除寄存器 (DMA_CHPRIALTCLR)

偏移地址: 034_H

复位值: $00000000\_00000000\_00000000\_00000000_B$

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

CHNL_PRI_ALT_CLR	Bit 31-0	W	**设置相应的比特位，为相应的DMA通道选择主要数据结构** 写入: 0: 无效。设置CHNL_PRI_ALT_SET寄存器用来选择交替数据结构 1: 为相应DMA通道选择主要数据结构。

注: 当发生以下情况时，控制器将翻转 CHNL_PRI_ALT_CLR 中相应的比特位的值:

1. 在存储器/外设分散-聚集 DMA 周期模式下，完成由主要数据结构指定的 4 次传输。
2. 在乒乓 DMA 周期模式下，完成由主要数据结构指定的所有传输。
3. 在乒乓/存储器分散-聚集/外设分散-聚集 DMA 周期模式下，完成由交替数据结构指定的所有传输。

9. 5. 3. 15 通道优先级设置寄存器（DMA_CHPRSET）

该寄存器为读写寄存器。通过该寄存器，用户可配置 DMA 通道为高优先级。读取时，可返回通道优先级的屏蔽状态。

通道优先级设置寄存器 (DMA_CHPRSET)

偏移地址: 038_H

复位值: $00000000\_00000000\_00000000\_00000000_B$

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

CHNL_PRIORITY_SET	Bit 31-0	R/W	**返回通道的优先级屏蔽状态，或设置通道优先级为高** 读取: 0: 相应DMA通道为默认优先级。 1: 相应DMA通道为高优先级。 写入: 0: 无效。设置CHNL_PRIORITY_CLR寄存器将相应通道设置为默认优先级。 1: 设置相应DMA通道为高优先级。

9. 5. 3. 16 通道优先级清除寄存器（DMA_CHPRCLR）

该寄存器为只写寄存器。通过该寄存器，用户可将 DMA 通道设置为默认优先级。

通道优先级清除寄存器 (DMA_CHPRCLR)																															
偏移地址: $03C_H$																															
复位值: $00000000\_00000000\_00000000\_00000000_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CHNL_PRIORITY_SET																															

CHNL_PRIORITY_CLR	Bit 31-0	W	**设置相应的比特位，为相应的DMA通道选择默认优先级** 写入: 0: 无效。设置CHNL_PRIORITY_SET寄存器，将相应DMA通道设置为高优先级。 1: 将相应DMA通道设置为默认优先级。

9. 5. 3. 17 总线错误清除寄存器（DMA_ERRCLR）

该读写寄存器可返回 dma_err 的状态，可将 dma_err 设置为低电平。

总线错误清除寄存器 (DMA_ERRCLR)																															
偏移地址: $04C_H$																															
复位值: $00000000\_00000000\_00000000\_00000000_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																															ERR_CLR

Reserved	Bit 31-1	-	**未定义，写0。**
ERR_CLR	Bit 0	R/W	**返回dma_err的状态，或设置该信号为低电平** 读取: 0: dma_err为低电平 1: dma_err为高电平 写入: 0: 无效。dma_err状态不变。 1: 设置dma_err为低电平。 若以测试为目的，设置dma_err为高电平。

注: 当 AHB-Lite 总线上发生错误的同时，若 dma_err 被置为无效，则错误条件先发生，dma_err 保持有效。

9. 5. 3. 18 DMA中断状态寄存器（DMA_IFLAG）

DMA 中断状态寄存器（DMA_IFLAG）

偏移地址：1000_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DMAERRIF	Reserved																											CH3DONEIF	CH2DONEIF	CH1DONEIF	CH0DONEIF

DMAERRIF	Bit 31	R	**DMA 错误中断标志** 0：DMA 未发生错误 1：DMA 发生错误
Reserved	Bit 30-4	–	**保留**
CH3DONEIF	Bit 3	R	**DMA 通道 3 结束中断标志** 0：DMA 未结束 1：DMA 结束
CH2DONEIF	Bit 2	R	**DMA 通道 2 结束中断标志** 0：DMA 未结束 1：DMA 结束
CH1DONEIF	Bit 1	R	**DMA 通道 1 结束中断标志** 0：DMA 未结束 1：DMA 结束
CH0DONEIF	Bit 0	R	**DMA 通道 0 结束中断标志** 0：DMA 未结束 1：DMA 结束

9. 5. 3. 19 DMA中断状态清零寄存器（DMA_ICFR）

DMA 中断状态清零寄存器（DMA_ICFR）

偏移地址：1008_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DMAERRC	Reserved																											CH3DONEC	CH2DONEC	CH1DONEC	CH0DONEC

DMAERRC	Bit 31	W1	**DMA 错误中断标志清零** 对该位写 1 有效。
Reserved	Bit 30-4	–	**保留**
CH3DONEC	Bit 3	W1	**DMA 通道 3 结束中断标志清零** 对该位写 1 有效。
CH2DONEC	Bit 2	W1	**DMA 通道 2 结束中断标志清零** 对该位写 1 有效。
CH1DONEC	Bit 1	W1	**DMA 通道 1 结束中断标志清零** 对该位写 1 有效。
CH0DONEC	Bit 0	W1	**DMA 通道 0 结束中断标志清零** 对该位写 1 有效。

9. 5. 3. 20 DMA中断使能控制寄存器（DMA_IER）

DMA 中断使能控制寄存器（DMA_IER）

偏移地址：$100C_H$

复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DMAERRIE	Reserved																											CH3DONEIE	CH2DONEIE	CH1DONEIE	CH0DONEIE

DMAERRIE	Bit 31	R/W	**DMA 错误中断使能** 0：中断禁止 1：中断使能
Reserved	Bit 30-4	–	**保留**
CH3DONEIE	Bit 3	R/W	**DMA 通道 3 结束中断使能** 0：中断禁止 1：中断使能
CH2DONEIE	Bit 2	R/W	**DMA 通道 2 结束中断使能** 0：中断禁止 1：中断使能
CH1DONEIE	Bit 1	R/W	**DMA 通道 1 结束中断使能** 0：中断禁止 1：中断使能
CH0DONEIE	Bit 0	R/W	**DMA 通道 0 结束中断使能** 0：中断禁止 1：中断使能

9. 5. 3. 21 DMA通道 0 复用选择寄存器（DMA_CH0_SELCON）

DMA 通道 0 复用选择寄存器（DMA_CH0_SELCON）

偏移地址：1100_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-12	–	**保留**
MSEL	Bit 11-8	R/W	**输入源选择** 0000：无输入 0001：GPIO 0010：ADC 0011：CRC 0100：USART0 0101：USART1 0110：I2C1 0111：SPI 1000：I2C0 1001：AD16C4T 1010：GP16C2T0 1011：GP16C2T1 1100：GP16C2T2 1101：PIS 1110：BSTIM0 1111：- 其他：保留
Reserved	Bit 7-4	–	**保留**
MSIGSEL	Bit 3-0	R/W	**MSEL=0000** MSIGSEL 无效 **MSEL=0001** 0000：EXTI0 0001：EXTI1 0010：EXTI2 0011：EXTI3 0100：EXTI4 0101：EXTI5 0110：EXTI6 0111：EXTI7 1000：EXTI8 1001：EXTI9

			1010：EXTI10 1011：EXTI11 1100：EXTI12 1101：EXTI13 1110：EXTI14 1111：EXTI15 **MSEL=0010(SRC=ADC)** ADC DMA 申请 **MSEL=0011(SRC=CRC)** CRC DMA 申请 **MSEL=0100, 0101 (SRC=USART0,USART1)** 0000: 接收缓冲器非空申请 0001: 发送缓冲器空申请 **MSEL=0111(SRC=SPI)** 0000: 接收缓冲器非空申请 0001: 发送缓冲器空申请 **MSEL=1000, 0110 (I2C0、I2C1)** 0000: 接收缓冲器非空申请 0001: 发送缓冲器空申请 **MSEL=1001, 1010, 1011, 1100 (SRC=AD16C4T，GP16C2T0，GP16C2T1，GP16C2T2)** 0000: 捕捉比较通道 1 申请 0001: 捕捉比较通道 2 申请 0010: 捕捉比较通道 3 申请 0011: 捕捉比较通道 4 申请 0100: TIMER 触发申请 0101: TIMER 比较匹配申请 0110: TIMER 更新事件申请 **MSEL=1101（SRC=PIS）** 0000：PIS 通道 0 0001：PIS 通道 1 0010：PIS 通道 2 0011：PIS 通道 3 0100：PIS 通道 4 0101：PIS 通道 5 **MSEL=1110（SRC=BSTIM0）** BSTIM0 DMA 申请

9. 5. 3. 22 DMA通道 1 复用选择寄存器（DMA_CH1_SELCON）

DMA 通道 1 复用选择寄存器（DMA_CH1_SELCON）

偏移地址：1104_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-12	–	保留
MSEL	Bit 11-8	R/W	**输入源选择** 0000：无输入 0001：GPIO 0010：ADC 0011：CRC 0100：USART0 0101：USART1 0110：I2C1 0111：SPI 1000：I2C0 1001：AD16C4T 1010：GP16C2T0 1011：GP16C2T1 1100：GP16C2T2 1101：PIS 1110：BSTIM0 1111：- 其他：保留
Reserved	Bit 7-4	–	保留
MSIGSEL	Bit 3-0	R/W	**MSEL=0000** MSIGSEL 无效 **MSEL=0001** 0000：EXTI0 0001：EXTI1 0010：EXTI2 0011：EXTI3 0100：EXTI4 0101：EXTI5 0110：EXTI6 0111：EXTI7 1000：EXTI8 1001：EXTI9

			1010：EXTI10 1011：EXTI11 1100：EXTI12 1101：EXTI13 1110：EXTI14 1111：EXTI15 **MSEL=0010(SRC=ADC)** ADC DMA 申请 **MSEL=0011(SRC=CRC)** CRC DMA 申请 **MSEL=0100, 0101 (SRC=USART0,USART1)** 0000: 接收缓冲器非空申请 0001: 发送缓冲器空申请 **MSEL=0111(SRC=SPI)** 0000: 接收缓冲器非空申请 0001: 发送缓冲器空申请 **MSEL=1000, 0110 (I2C0、I2C1)** 0000: 接收缓冲器非空申请 0001: 发送缓冲器空申请 **MSEL=1001, 1010, 1011, 1100 (SRC=AD16C4T，GP16C2T0，GP16C2T1，GP16C2T2)** 0000: 捕捉比较通道 1 申请 0001: 捕捉比较通道 2 申请 0010: 捕捉比较通道 3 申请 0011: 捕捉比较通道 4 申请 0100: TIMER 触发申请 0101: TIMER 比较匹配申请 0110: TIMER 更新事件申请 **MSEL=1101（SRC=PIS）** 0000：PIS 通道 0 0001：PIS 通道 1 0010：PIS 通道 2 0011：PIS 通道 3 0100：PIS 通道 4 0101：PIS 通道 5 **MSEL=1110（SRC=BSTIM0）** BSTIM0 DMA 申请

9. 5. 3. 23 DMA通道 2 复用选择寄存器（DMA_CH2_SELCON）

DMA 通道 2 复用选择寄存器（DMA_CH2_SELCON）

偏移地址：1108_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-12	–	保留	
MSEL	Bit 11-8	R/W	**输入源选择** 0000：无输入 0001：GPIO 0010：ADC 0011：CRC 0100：USART0 0101：USART1 0110：I2C1 0111：SPI 1000：I2C0 1001：AD16C4T 1010：GP16C2T0 1011：GP16C2T1 1100：GP16C2T2 1101：PIS 1110：BSTIM0 1111：- 其他：保留	
Reserved	Bit 7-4	–	保留	
MSIGSEL	Bit 3-0	R/W	**MSEL=0000** MSIGSEL 无效 **MSEL=0001** 0000：EXTI0 0001：EXTI1 0010：EXTI2 0011：EXTI3 0100：EXTI4 0101：EXTI5 0110：EXTI6 0111：EXTI7 1000：EXTI8 1001：EXTI9	

			1010：EXTI10 1011：EXTI11 1100：EXTI12 1101：EXTI13 1110：EXTI14 1111：EXTI15 **MSEL=0010(SRC=ADC)** ADC DMA 申请 **MSEL=0011(SRC=CRC)** CRC DMA 申请 **MSEL=0100, 0101 (SRC=USART0,USART1)** 0000: 接收缓冲器非空申请 0001: 发送缓冲器空申请 **MSEL=0111(SRC=SPI)** 0000: 接收缓冲器非空申请 0001: 发送缓冲器空申请 **MSEL=1000, 0110 (I2C0、I2C1)** 0000: 接收缓冲器非空申请 0001: 发送缓冲器空申请 **MSEL=1001, 1010, 1011, 1100 (SRC=AD16C4T，GP16C2T0，GP16C2T1，GP16C2T2)** 0000: 捕捉比较通道 1 申请 0001: 捕捉比较通道 2 申请 0010: 捕捉比较通道 3 申请 0011: 捕捉比较通道 4 申请 0100: TIMER 触发申请 0101: TIMER 比较匹配申请 0110: TIMER 更新事件申请 **MSEL=1101（SRC=PIS）** 0000：PIS 通道 0 0001：PIS 通道 1 0010：PIS 通道 2 0011：PIS 通道 3 0100：PIS 通道 4 0101：PIS 通道 5 **MSEL=1110（SRC=BSTIM0）** BSTIM0 DMA 申请	

9. 5. 3. 24 DMA通道 3 复用选择寄存器（DMA_CH3_SELCON）

DMA 通道 3 复用选择寄存器（DMA_CH3_SELCON）

偏移地址：$110C_H$

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-12	–	保留	
MSEL	Bit 11-8	R/W	**输入源选择** 0000：无输入 0001：GPIO 0010：ADC 0011：CRC 0100：USART0 0101：USART1 0110：I2C1 0111：SPI 1000：I2C0 1001：AD16C4T 1010：GP16C2T0 1011：GP16C2T1 1100：GP16C2T2 1101：PIS 1110：BSTIM0 1111：- 其他：保留	
Reserved	Bit 7-4	–	保留	
MSIGSEL	Bit 3-0	R/W	**MSEL=0000** MSIGSEL 无效 **MSEL=0001** 0000：EXTI0 0001：EXTI1 0010：EXTI2 0011：EXTI3 0100：EXTI4 0101：EXTI5 0110：EXTI6 0111：EXTI7 1000：EXTI8 1001：EXTI9	

			1010：EXTI10 1011：EXTI11 1100：EXTI12 1101：EXTI13 1110：EXTI14 1111：EXTI15 **MSEL=0010(SRC=ADC)** ADC DMA 申请 **MSEL=0011(SRC=CRC)** CRC DMA 申请 **MSEL=0100, 0101 (SRC=USART0,USART1)** 0000: 接收缓冲器非空申请 0001: 发送缓冲器空申请 **MSEL=0111(SRC=SPI)** 0000: 接收缓冲器非空申请 0001: 发送缓冲器空申请 **MSEL=1000, 0110 (I2C0、I2C1)** 0000: 接收缓冲器非空申请 0001: 发送缓冲器空申请 **MSEL=1001, 1010, 1011, 1100 (SRC=AD16C4T，GP16C2T0，GP16C2T1，GP16C2T2)** 0000: 捕捉比较通道 1 申请 0001: 捕捉比较通道 2 申请 0010: 捕捉比较通道 3 申请 0011: 捕捉比较通道 4 申请 0100: TIMER 触发申请 0101: TIMER 比较匹配申请 0110: TIMER 更新事件申请 **MSEL=1101（SRC=PIS）** 0000：PIS 通道 0 0001：PIS 通道 1 0010：PIS 通道 2 0011：PIS 通道 3 0100：PIS 通道 4 0101：PIS 通道 5 **MSEL=1110（SRC=BSTIM0）** BSTIM0 DMA 申请	

第10章 外设互联（PIS）

10. 1 概述

PIS（Peripheral Interaction System）在微控制器中作为外设互联的桥接口使用，利用 PIS 可实现外设之间的相互触发，控制及自动化工作，提高系统的实时性和快速响应能力，可避免占用过多的 CPU 资源并简化软件工作，提高系统效率。

10. 2 特性

- 最多支持 6 个 PIS 通道选择
- 支持通道输出到管脚
- USART0 输出调制可配置

10. 3 结构框图

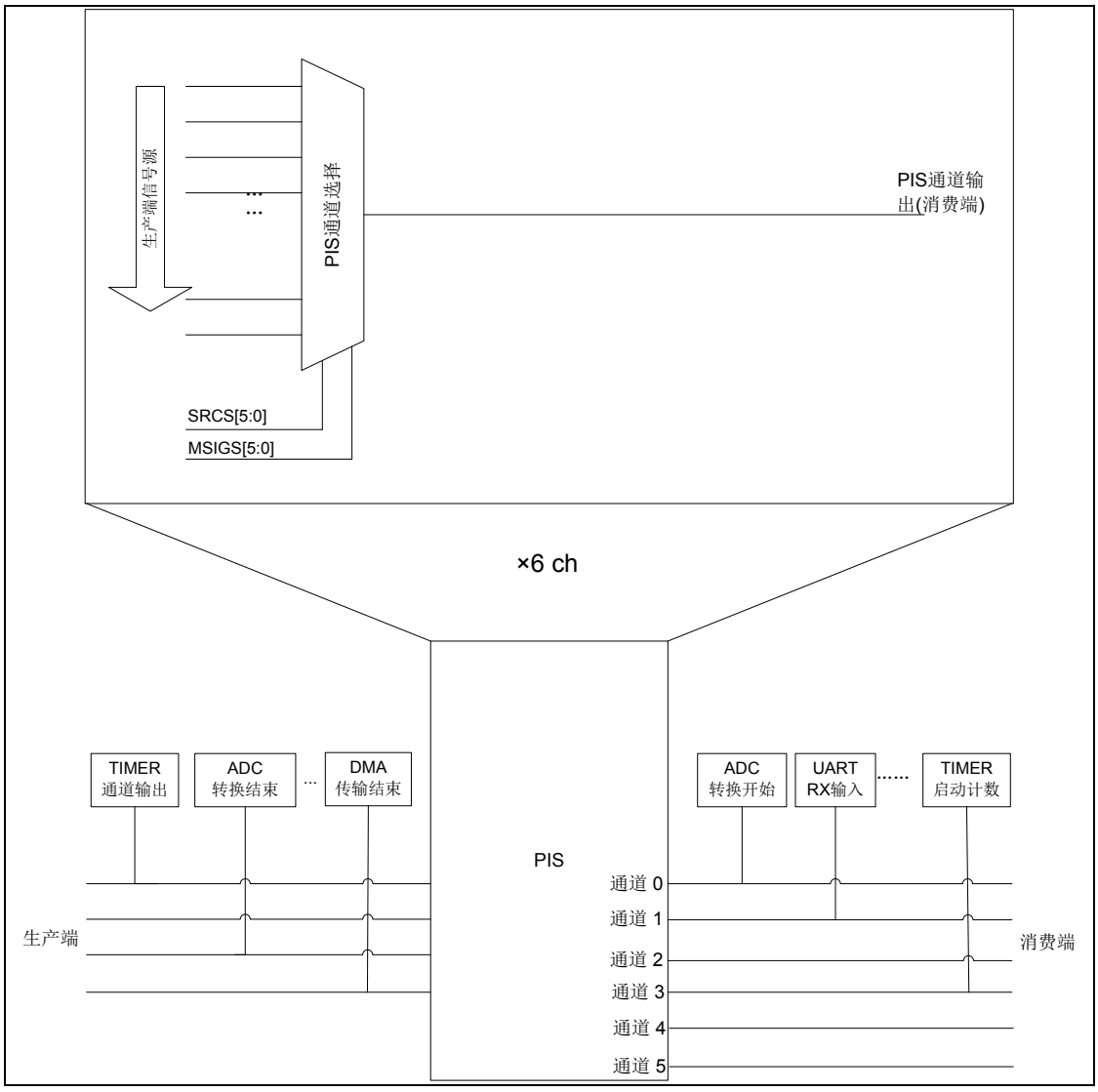


图 11-1 PIS 结构框图

10.4 功能描述

ES32M0150 产品的外设互联可支持 6 个通道资源，每个通道均可对生产端信号进行多路复用。针对不同应用可灵活配置。

10.4.1 生产端信号

外设互联的生产端信号如下表所示：

	生产端	输出形式	异步支持	位置（APB 或 AHB）
GPIO	PA0~PA15、PB0~PB6 输入	电平	是	AHB 外设
USART	接收缓冲器非空	脉冲	—	APB 外设
	发送缓冲器空	脉冲	—	
SPI	接收缓冲器非空	脉冲	—	APB 外设
	发送缓冲器空	脉冲	—	
I2C	接收缓冲器非空	电平	—	APB 外设
	发送缓冲器空	电平	—	
TIMER	更新事件	脉冲	—	APB 外设
	触发事件	脉冲	—	
	输入捕获	脉冲	—	
	输出比较	脉冲	—	
	TRIGOUT	电平	—	
ADC	模拟看门狗事件	脉冲	—	APB 外设
	规则组转换结束	脉冲	—	
	注入组转换结束	脉冲	—	
ACMP	比较器结果输出	电平	是	APB 外设
LVD	LVD 检测输出	电平	是	APB 外设
DMA	DMA 通道完成	脉冲	是	AHB 外设

表 11-1 生产端信号

10.4.2 消费端信号

外设互联的消费端信号如下表所示：

	消费端	输入形式	异步支持	位置（APB，或 AHB）
USART	RX 输入	电平	—	APB 外设
SPI	RX 输入	电平	—	APB 外设
	CLK 输入	电平	—	
TIMER	启动	脉冲	—	APB 外设
	停止	脉冲	—	
	清零	脉冲	—	
	比较捕捉通道输入	电平或脉冲	—	

	断路输入	电平	—	
ADC	启动规则组转换	脉冲	—	APB 外设
	启动注入组转换	脉冲	—	
DMA	DMA 通道请求	脉冲	是	AHB 外设

表 11-2 消费端信号

消费端信号的 PIS 通道分配如下表所示：

	消费端	源通道	备注
USART0	RX 输入	PIS 通道 1	由 USART0_RXD_SEL 设定
USART1	RX 输入	PIS 通道 4	由 USART1_RXD_SEL 设定
SPI0	MISO 输入	PIS 通道 0	由 SPI0_RX_SEL 设定
	MOSI 输入	PIS 通道 0	由 SPI0_RX_SEL 设定
	CLK 输入	PIS 通道 1	由 SPI0_CLK_SEL 设定
AD16C4T0	ITR0	PIS 通道 3	—
	ITR1	PIS 通道 3	—
	ITR2	PIS 通道 3	—
	ITR3	PIS 通道 3	—
	捕捉通道 1	PIS 通道 0	由 AD16C4T0_CH1IN_SEL 设定
	捕捉通道 2	PIS 通道 1	由 AD16C4T0_CH2IN_SEL 设定
	捕捉通道 3	PIS 通道 2	由 AD16C4T0_CH3IN_SEL 设定
	捕捉通道 4	PIS 通道 3	由 AD16C4T0_CH4IN_SEL 设定
	断路输入	PIS 通道 0	由 AD16C4T0_BRKIN_SEL 设定
GP16C2T0	ITR0	PIS 通道 2	—
	ITR1	PIS 通道 2	—
	ITR2	PIS 通道 2	—
	ITR3	PIS 通道 2	—
	捕捉通道 1	PIS 通道 0	由 GP16C2T0_CH1IN_SEL 设定
	捕捉通道 2	PIS 通道 1	由 GP16C2T0_CH2IN_SEL 设定

	消费端	源通道	备注
GP16C2T1	ITR0	PIS 通道 0	—
	ITR1	PIS 通道 0	—
	ITR2	PIS 通道 0	—
	ITR3	PIS 通道 0	—
	捕捉通道 1	PIS 通道 2	由 GP16C2T1_CH1IN_SEL 设定
	捕捉通道 2	PIS 通道 3	由 GP16C2T1_CH2IN_SEL 设定
GP16C2T2	ITR0	PIS 通道 1	—
	ITR1	PIS 通道 1	—
	ITR2	PIS 通道 1	—
	ITR3	PIS 通道 1	—
	捕捉通道 1	PIS 通道 0	由 GP16C2T2_CH1IN_SEL 设定
	捕捉通道 2	PIS 通道 1	由 GP16C2T2_CH2IN_SEL 设定
ADC	启动规则组转换	PIS 通道 4	—
	启动注入组转换	PIS 通道 5	—
DMA	DMA 请求	PIS 通道 0	—

表 11-3 消费端的 PIS 通道分配

10. 4. 3 PIS通道选择

PIS 的源端定义为生产端信号，PIS 的输出信号用于消费端。消费端可根据应用需要，来选择合适的生产端信号，并通过选择同步路径以保证正确采样到生产端信号。PIS 的生产端信号选择由 PIS 通道控制寄存器（PIS_CHx_CON,x=0~5）配置。PIS 通道控制寄存器的 PIS_CHx_CON.SRCS 位（x=0~5）用来选择生产端模块，配置 PIS_CHx_CON.MSIGS 位（x=0~5）则可从生产端模块的多路信号中选择一路作为生产端信号。

以 GP16C2T0（生产端）和 ADC（消费端）为例说明 PIS 的配置，参考如下：

1. 通过上表可知 ADC 规则组转换的启动信号为 PIS 通道 4。
2. 设定寄存器 PIS_CH4_CON.SRCS 为 1001 选择 GP16C2T0 为生产端模块。
3. 设定寄存器 PIS_CH4_CON.MSIGS 为 0000，选择 GP16C2T0 的同步更新事件作

为生产端信号。

4. 配置 ADC 选择外部触发方式进行规则组转换。
5. 配置 GP16C2T0 进行计数，产生更新事件后将触发 ADC 开始 AD 转换。

10. 4. 4 USART输出调制

USART0 的输出调制功能利用定时器 PWM 波或 BUZ 信号对 USART0 的 TX 调制后发送到端口。

调制方式如下图所示：

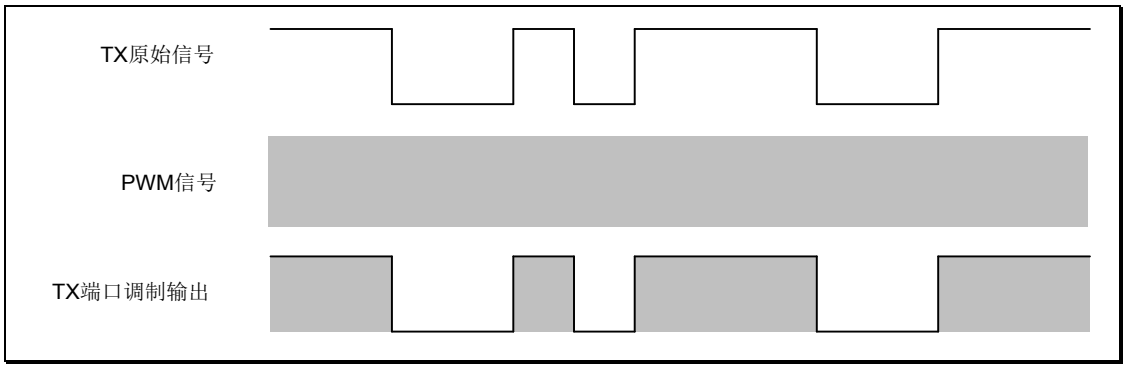


图 11-2 高电平调制输出波形图

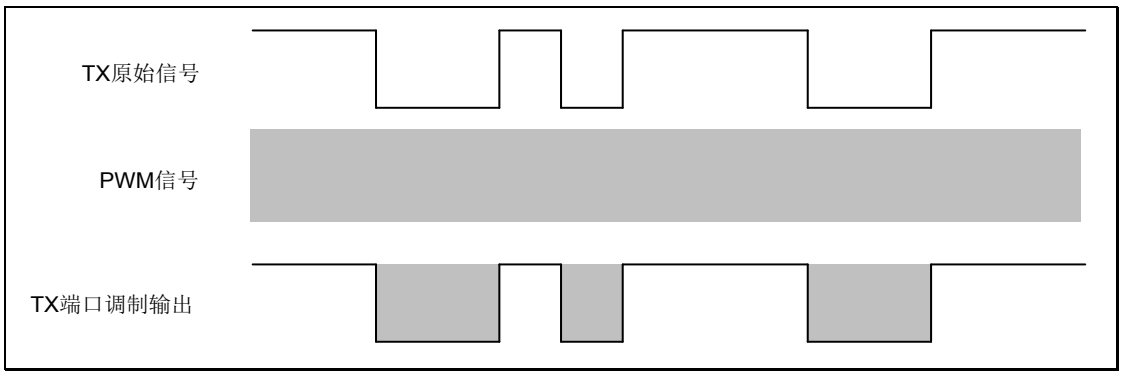


图 11-3 低电平调制输出波形图

以下为 USART 输出调制的参考配置流程（以 AD16C4T0 调制 USART0 为例）：

1. 设置寄存器 USART0_TXMCR.TXMSS 为 0001，选择 AD16C4T0 作为调制源。
2. 设置寄存器 USART0_TXMCR.TXSIGS 为 0000，选择 AD16C4T0 通道 1 输出作为调制信号。
3. 设置 USART0_TXMCR.TXMLVLS 选择调制电平。
4. 配置 AD16C4T0 进行计数。
5. 配置 USART0 发送数据。

10. 5 特殊功能寄存器

10. 5. 1 寄存器列表

PIS 寄存器列表		
名称	偏移地址	描述
PIS_CH0_CON	0000_H	PIS 通道 0 控制寄存器
PIS_CH1_CON	0004_H	PIS 通道 1 控制寄存器
PIS_CH2_CON	0008_H	PIS 通道 2 控制寄存器
PIS_CH3_CON	$000C_H$	PIS 通道 3 控制寄存器
PIS_CH4_CON	0010_H	PIS 通道 4 控制寄存器
PIS_CH5_CON	0014_H	PIS 通道 5 控制寄存器
Reserved	0010_H~$003C_H$	保留
PIS_CH_OER	0040_H	PIS 通道端口输出使能寄存器
PIS_TAR_CON0	0044_H	PIS 消费端通道控制寄存器 0
PIS_TAR_CON1	0048_H	PIS 消费端通道控制寄存器 1
Reserved	$004C_H$~$005C_H$	保留
USART0_TXMCR	0060_H	USART0 输出调制控制寄存器

10. 5. 2 寄存器描述

10. 5. 2. 1 PIS通道 0 控制寄存器（PIS_CH0_CON）

PIS 通道 0 控制寄存器（PIS_CH0_CON）																															
偏移地址：00_H																															
复位值：$00000000\_00000000\_00000000\_00000000_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																				SRCS				Reserved				MSIGS			

Reserved	Bit 31-12	—	保留
SRCS	Bit 11-8	R/W	**输入源选择** 0000：无输入 0001：GPIOA 0010：GPIOB 0011：LVD/CMP 0100：USART0 0101：USART1 0110：I2C1 0111：SPI 1000：I2C0 1001：GP16C2T0 1010：GP16C2T1 1011：GP16C2T2 1100：AD16C4T 1101：DMA 1110：ADC 1111：BSTIM0 其他：保留
Reserved	Bit 7-4	—	保留
MSIGS	Bit 3-0	R/W	**SRCS=0000** MSIGS 无效 **SRCS=0001（SRC=GPIOA）** 0000：PA0 …… 1110：PA14 1111：PA15 **SRCS=0010（SRC=GPIOB）** 0000：PB0 …… 0110：PB6 **SRCS=0011（SRC=LVD/CMP）**

			0000：LVD 比较器输出 0001：CMP0 比较器输出 0010：CMP1 比较器输出 **SRCS=0100, 0101（SRC=USART0,USART1）** 0000：接收缓冲器非空脉冲 0001：发送缓冲器空脉冲 0010：TX 输出 **SRCS=0111（SRC=SPI）** 0000：接收缓冲器非空脉冲 0001：发送缓冲器空脉冲 0010：“片选”输出 **SRCS=1000，0110（SRC=I2C0、I2C1）** 0000：接收缓冲器非空电平 0001：发送缓冲器空电平 **SRCS=1001, 1010, 1011（SRC= GP16C2T0,GP16C2T1,GP16C2T2）** 0000：更新事件脉冲 0001：触发事件脉冲 0010：通道 1 输入捕获输出比较脉冲 0011：通道 2 输入捕获输出比较脉冲 0110：TRGOUT 外部触发信号 **SRCS= 1100（SRC= AD16C4T0）** 0000：更新事件脉冲 0001：触发事件脉冲 0010：通道 1 输入捕获输出比较脉冲 0011：通道 2 输入捕获输出比较脉冲 0100：通道 3 输入捕获输出比较脉冲 0101：通道 4 输入捕获输出比较脉冲 0110：TRGOUT 外部触发信号 **SRCS=1101（SRC=DMA）** DMA 通道完成脉冲 **SRCS=1110（SRC=ADC）** 0000：注入组转换结束 0001：规则组转换结束 0010：窗口比较（模拟看门狗）触发 **SRCS=1111（SRC=BSTIM0）** BSTIM0 更新事件脉冲

10. 5. 2. 2 PIS通道 1 控制寄存器（PIS_CH1_CON）

PIS 通道 1 控制寄存器（PIS_CH1_CON）

偏移地址：04_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-12	—	**保留**
SRCS	Bit 11-8	R/W	**输入源选择** 0000：无输入 0001：GPIOA 0010：GPIOB 0011：LVD/CMP 0100：USART0 0101：USART1 0110：I2C1 0111：SPI 1000：I2C0 1001：GP16C2T0 1010：GP16C2T1 1011：GP16C2T2 1100：AD16C4T 1101：DMA 1110：ADC 1111：BSTIM0 其他：保留
Reserved	Bit 7-4	—	**保留**
MSIGS	Bit 3-0	R/W	**SRCS=0000** MSIGS 无效 **SRCS=0001（SRC=GPIOA）** 0000：PA0 …… 1110：PA14 1111：PA15 **SRCS=0010（SRC=GPIOB）** 0000：PB0 …… 0110：PB6 **SRCS=0011（SRC=LVD/CMP）** 0000：LVD 比较器输出

			0001：CMP0 比较器输出 0010：CMP1 比较器输出 **SRCS=0100, 0101（SRC=USART0,USART1）** 0000：接收缓冲器非空脉冲 0001：发送缓冲器空脉冲 0010：TX 输出 **SRCS=0111（SRC=SPI）** 0000：接收缓冲器非空脉冲 0001：发送缓冲器空脉冲 0010：“片选”输出 **SRCS=1000，0110（SRC=I2C0、I2C1）** 0000：接收缓冲器非空电平 0001：发送缓冲器空电平 **SRCS=1001, 1010, 1011（SRC= GP16C2T0,GP16C2T1,GP16C2T2）** 0000：更新事件脉冲 0001：触发事件脉冲 0010：通道 1 输入捕获输出比较脉冲 0011：通道 2 输入捕获输出比较脉冲 0110：TRGOUT 外部触发信号 **SRCS= 1100（SRC= AD16C4T0）** 0000：更新事件脉冲 0001：触发事件脉冲 0010：通道 1 输入捕获输出比较脉冲 0011：通道 2 输入捕获输出比较脉冲 0100：通道 3 输入捕获输出比较脉冲 0101：通道 4 输入捕获输出比较脉冲 0110：TRGOUT 外部触发信号 **SRCS=1101（SRC=DMA）** DMA 通道完成脉冲 **SRCS=1110（SRC=ADC）** 0000：注入组转换结束 0001：规则组转换结束 0010：窗口比较（模拟看门狗）触发 **SRCS=1111（SRC=BSTIM0）** BSTIM0 更新事件脉冲

10. 5. 2. 3 PIS通道 2 控制寄存器（PIS_CH2_CON）

PIS 通道 2 控制寄存器（PIS_CH2_CON）

偏移地址：08_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-12	—	**保留**
SRCS	Bit 11-8	R/W	**输入源选择** 0000：无输入 0001：GPIOA 0010：GPIOB 0011：LVD/CMP 0100：USART0 0101：USART1 0110：I2C1 0111：SPI 1000：I2C0 1001：GP16C2T0 1010：GP16C2T1 1011：GP16C2T2 1100：AD16C4T 1101：DMA 1110：ADC 1111：BSTIM0 其他：保留
Reserved	Bit 7-4	—	**保留**
MSIGS	Bit 3-0	R/W	**SRCS=0000** MSIGS 无效 **SRCS=0001（SRC=GPIOA）** 0000：PA0 …… 1110：PA14 1111：PA15 **SRCS=0010（SRC=GPIOB）** 0000：PB0 …… 0110：PB6 **SRCS=0011（SRC=LVD/CMP）** 0000：LVD 比较器输出

			0001：CMP0 比较器输出 0010：CMP1 比较器输出 **SRCS=0100, 0101（SRC=USART0,USART1）** 0000：接收缓冲器非空脉冲 0001：发送缓冲器空脉冲 0010：TX 输出 **SRCS=0111（SRC=SPI）** 0000：接收缓冲器非空脉冲 0001：发送缓冲器空脉冲 0010：“片选”输出 **SRCS=1000，0110（SRC=I2C0、I2C1）** 0000：接收缓冲器非空电平 0001：发送缓冲器空电平 **SRCS=1001, 1010, 1011（SRC= GP16C2T0,GP16C2T1,GP16C2T2）** 0000：更新事件脉冲 0001：触发事件脉冲 0010：通道 1 输入捕获输出比较脉冲 0011：通道 2 输入捕获输出比较脉冲 0110：TRGOUT 外部触发信号 **SRCS= 1100（SRC= AD16C4T0）** 0000：更新事件脉冲 0001：触发事件脉冲 0010：通道 1 输入捕获输出比较脉冲 0011：通道 2 输入捕获输出比较脉冲 0100：通道 3 输入捕获输出比较脉冲 0101：通道 4 输入捕获输出比较脉冲 0110：TRGOUT 外部触发信号 **SRCS=1101（SRC=DMA）** DMA 通道完成脉冲 **SRCS=1110（SRC=ADC）** 0000：注入组转换结束 0001：规则组转换结束 0010：窗口比较（模拟看门狗）触发 **SRCS=1111（SRC=BSTIM0）** BSTIM0 更新事件脉冲

10. 5. 2. 4 PIS通道 3 控制寄存器（PIS_CH3_CON）

PIS 通道 3 控制寄存器（PIS_CH3_CON）

偏移地址：$0C_H$

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-12	—	**保留**
SRCS	Bit 11-8	R/W	**输入源选择** 0000：无输入 0001：GPIOA 0010：GPIOB 0011：LVD/CMP 0100：USART0 0101：USART1 0110：I2C1 0111：SPI 1000：I2C0 1001：GP16C2T0 1010：GP16C2T1 1011：GP16C2T2 1100：AD16C4T 1101：DMA 1110：ADC 1111：BSTIM0 其他：保留
Reserved	Bit 7-4	—	**保留**
MSIGS	Bit 3-0	R/W	**SRCS=0000** MSIGS 无效 **SRCS=0001（SRC=GPIOA）** 0000：PA0 …… 1110：PA14 1111：PA15 **SRCS=0010（SRC=GPIOB）** 0000：PB0 …… 0110：PB6 **SRCS=0011（SRC=LVD/CMP）** 0000：LVD 比较器输出

			0001：CMP0 比较器输出 0010：CMP1 比较器输出 **SRCS=0100, 0101（SRC=USART0,USART1）** 0000：接收缓冲器非空脉冲 0001：发送缓冲器空脉冲 0010：TX 输出 **SRCS=0111（SRC=SPI）** 0000：接收缓冲器非空脉冲 0001：发送缓冲器空脉冲 0010：“片选”输出 **SRCS=1000，0110（SRC=I2C0、I2C1）** 0000：接收缓冲器非空电平 0001：发送缓冲器空电平 **SRCS=1001, 1010, 1011（SRC= GP16C2T0,GP16C2T1,GP16C2T2）** 0000：更新事件脉冲 0001：触发事件脉冲 0010：通道 1 输入捕获输出比较脉冲 0011：通道 2 输入捕获输出比较脉冲 0110：TRGOUT 外部触发信号 **SRCS= 1100（SRC= AD16C4T0）** 0000：更新事件脉冲 0001：触发事件脉冲 0010：通道 1 输入捕获输出比较脉冲 0011：通道 2 输入捕获输出比较脉冲 0100：通道 3 输入捕获输出比较脉冲 0101：通道 4 输入捕获输出比较脉冲 0110：TRGOUT 外部触发信号 **SRCS=1101（SRC=DMA）** DMA 通道完成脉冲 **SRCS=1110（SRC=ADC）** 0000：注入组转换结束 0001：规则组转换结束 0010：窗口比较（模拟看门狗）触发 **SRCS=1111（SRC=BSTIM0）** BSTIM0 更新事件脉冲

10. 5. 2. 5 PIS通道 4 控制寄存器（PIS_CH4_CON）

PIS 通道 4 控制寄存器（PIS_CH4_CON）

偏移地址：10_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-12	—	**保留**
SRCS	Bit 11-8	R/W	**输入源选择** 0000：无输入 0001：GPIOA 0010：GPIOB 0011：LVD/CMP 0100：USART0 0101：USART1 0110：I2C1 0111：SPI 1000：I2C0 1001：GP16C2T0 1010：GP16C2T1 1011：GP16C2T2 1100：AD16C4T 1101：DMA 1110：ADC 1111：BSTIM0 其他：保留
Reserved	Bit 7-4	—	**保留**
MSIGS	Bit 3-0	R/W	**SRCS=0000** MSIGS 无效 **SRCS=0001（SRC=GPIOA）** 0000：PA0 …… 1110：PA14 1111：PA15 **SRCS=0010（SRC=GPIOB）** 0000：PB0 …… 0110：PB6 **SRCS=0011（SRC=LVD/CMP）** 0000：LVD 比较器输出

			0001：CMP0 比较器输出 0010：CMP1 比较器输出 **SRCS=0100, 0101（SRC=USART0,USART1）** 0000：接收缓冲器非空脉冲 0001：发送缓冲器空脉冲 0010：TX 输出 **SRCS=0111（SRC=SPI）** 0000：接收缓冲器非空脉冲 0001：发送缓冲器空脉冲 0010：“片选”输出 **SRCS=1000，0110（SRC=I2C0、I2C1）** 0000：接收缓冲器非空电平 0001：发送缓冲器空电平 **SRCS=1001, 1010, 1011（SRC= GP16C2T0,GP16C2T1,GP16C2T2）** 0000：更新事件脉冲 0001：触发事件脉冲 0010：通道 1 输入捕获输出比较脉冲 0011：通道 2 输入捕获输出比较脉冲 0110：TRGOUT 外部触发信号 **SRCS= 1100（SRC= AD16C4T0）** 0000：更新事件脉冲 0001：触发事件脉冲 0010：通道 1 输入捕获输出比较脉冲 0011：通道 2 输入捕获输出比较脉冲 0100：通道 3 输入捕获输出比较脉冲 0101：通道 4 输入捕获输出比较脉冲 0110：TRGOUT 外部触发信号 **SRCS=1101（SRC=DMA）** DMA 通道完成脉冲 **SRCS=1110（SRC=ADC）** 0000：注入组转换结束 0001：规则组转换结束 0010：窗口比较（模拟看门狗）触发 **SRCS=1111（SRC=BSTIM0）** BSTIM0 更新事件脉冲

10. 5. 2. 6 PIS通道 5 控制寄存器（PIS_CH5_CON）

PIS 通道 5 控制寄存器（PIS_CH5_CON）

偏移地址：14_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-12	—	**保留**
SRCS	Bit 11-8	R/W	**输入源选择** 0000：无输入 0001：GPIOA 0010：GPIOB 0011：LVD/CMP 0100：USART0 0101：USART1 0110：I2C1 0111：SPI 1000：I2C0 1001：GP16C2T0 1010：GP16C2T1 1011：GP16C2T2 1100：AD16C4T 1101：DMA 1110：ADC 1111：BSTIM0 其他：保留
Reserved	Bit 7-4	—	**保留**
MSIGS	Bit 3-0	R/W	**SRCS=0000** MSIGS 无效 **SRCS=0001（SRC=GPIOA）** 0000：PA0 …… 1110：PA14 1111：PA15 **SRCS=0010（SRC=GPIOB）** 0000：PB0 …… 0110：PB6 **SRCS=0011（SRC=LVD/CMP）** 0000：LVD 比较器输出

			0001：CMP0 比较器输出 0010：CMP1 比较器输出 **SRCS=0100, 0101（SRC=USART0,USART1）** 0000：接收缓冲器非空脉冲 0001：发送缓冲器空脉冲 0010：TX 输出 **SRCS=0111（SRC=SPI）** 0000：接收缓冲器非空脉冲 0001：发送缓冲器空脉冲 0010：“片选”输出 **SRCS=1000，0110（SRC=I2C0、I2C1）** 0000：接收缓冲器非空电平 0001：发送缓冲器空电平 **SRCS=1001, 1010, 1011（SRC= GP16C2T0,GP16C2T1,GP16C2T2）** 0000：更新事件脉冲 0001：触发事件脉冲 0010：通道 1 输入捕获输出比较脉冲 0011：通道 2 输入捕获输出比较脉冲 0110：TRGOUT 外部触发信号 **SRCS= 1100（SRC= AD16C4T0）** 0000：更新事件脉冲 0001：触发事件脉冲 0010：通道 1 输入捕获输出比较脉冲 0011：通道 2 输入捕获输出比较脉冲 0100：通道 3 输入捕获输出比较脉冲 0101：通道 4 输入捕获输出比较脉冲 0110：TRGOUT 外部触发信号 **SRCS=1101（SRC=DMA）** DMA 通道完成脉冲 **SRCS=1110（SRC=ADC）** 0000：注入组转换结束 0001：规则组转换结束 0010：窗口比较（模拟看门狗）触发 **SRCS=1111（SRC=BSTIM0）** BSTIM0 更新事件脉冲

10. 5. 2. 7 PIS通道端口输出使能寄存器（PIS_CH_OER）

PIS 通道端口输出使能寄存器（PIS_CH_OER）																													
偏移地址：40_H																													
复位值：$00000000\_00000000\_00000000\_00000000_B$																													
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Reserved	CH5OE	CH4OE	CH3OE	CH2OE	CH1OE	CH0OE

Reserved	Bit 31-4	—	保留
CH5OE	Bit 5	R/W	**PIS 通道 5 输出至端口使能** 0：输出到端口禁止 1：输出到端口使能
CH4OE	Bit 4	R/W	**PIS 通道 4 输出至端口使能** 0：输出到端口禁止 1：输出到端口使能
CH3OE	Bit 3	R/W	**PIS 通道 3 输出至端口使能** 0：输出到端口禁止 1：输出到端口使能
CH2OE	Bit 2	R/W	**PIS 通道 2 输出至端口使能** 0：输出到端口禁止 1：输出到端口使能
CH1OE	Bit 1	R/W	**PIS 通道 1 输出至端口使能** 0：输出到端口禁止 1：输出到端口使能
CH0OE	Bit 0	R/W	**PIS 通道 0 输出至端口使能** 0：输出到端口禁止 1：输出到端口使能

10. 5. 2. 8 PIS消费端通道控制寄存器 0（PIS_TAR_CON0）

PIS 消费端通道控制寄存器 0（PIS_TAR_CON0）

偏移地址：44_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						GP16C2T2_CH2IN_SEL	GP16C2T2_CH1IN_SEL	Reserved						GP16C2T1_CH2IN_SEL	GP16C2T1_CH1IN_SEL	Reserved						GP16C2T0_CH2IN_SEL	GP16C2T0_CH1IN_SEL	Reserved			AD16C4T0_BRKIN_SEL	AD16C4T0_CH4IN_SEL	AD16C4T0_CH3IN_SEL	AD16C4T0_CH2IN_SEL	AD16C4T0_CH1IN_SEL

Reserved	Bit 31-26	—	保留
GP16C2T2_CH2IN_SEL	Bit 25	R/W	**GP16C2T2 输入捕捉通道 2 输入选择** 0：从端口输入 1：将 PIS 通道 1 输出作为输入
GP16C2T2_CH1IN_SEL	Bit 24	R/W	**GP16C2T2 输入捕捉通道 1 输入选择** 0：从端口输入 1：将 PIS 通道 0 输出作为输入

Reserved	Bit 23-18	—	保留
GP16C2T1_CH2IN_SEL	Bit 17	R/W	**GP16C2T1 输入捕捉通道 2 输入选择** 0：从端口输入 1：将 PIS 通道 3 输出作为输入
GP16C2T1_CH1IN_SEL	Bit 16	R/W	**GP16C2T1 输入捕捉通道 1 输入选择** 0：从端口输入 1：将 PIS 通道 2 输出作为输入
Reserved	Bit 15-10	—	保留
GP16C2T0_CH2IN_SEL	Bit 9	R/W	**GP16C2T0 输入捕捉通道 2 输入选择** 0：从端口输入 1：将 PIS 通道 1 输出作为输入
GP16C2T0_CH1IN_SEL	Bit 8	R/W	**GP16C2T0 输入捕捉通道 1 输入选择** 0：从端口输入 1：将 PIS 通道 0 输出作为输入
Reserved	Bit 7-5	—	保留
AD16C4T0_BRKIN_SEL	Bit 4	R/W	**AD16C4T0 刹车输入选择** 0：从端口输入 1：将 PIS 通道 0 输出作为输入
AD16C4T0_CH4IN_SEL	Bit 3	R/W	**AD16C4T0 输入捕捉通道 4 输入选择** 0：从端口输入 1：将 PIS 通道 3 输出作为输入
AD16C4T0_CH3IN_SEL	Bit 2	R/W	**AD16C4T0 输入捕捉通道 3 输入选择** 0：从端口输入 1：将 PIS 通道 2 输出作为输入
AD16C4T0_CH2IN_SEL	Bit 1	R/W	**AD16C4T0 输入捕捉通道 2 输入选择** 0：从端口输入 1：将 PIS 通道 1 输出作为输入
AD16C4T0_CH1IN_SEL	Bit 0	R/W	**AD16C4T0 输入捕捉通道 1 输入选择** 0：从端口输入 1：将 PIS 通道 0 输出作为输入

10. 5. 2. 9 PIS消费端通道控制寄存器 1（PIS_TAR_CON1）

PIS 消费端通道控制寄存器 1（PIS_TAR_CON1）

偏移地址：48_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																						SPI_CLK_SEL	SPI_RX_SEL	Reserved						USART1_RXD_SEL	USART0_RXD_SEL

Reserved	Bit 31-10	—	保留
SPI_CLK_SEL	Bit9	R/W	**SPI CLK 输入选择** 0：从端子输入 1：将 PIS 通道 1 作为输入
SPI_RX_SEL	Bit 8	R/W	**SPI RX 输入选择** 0：从端子输入 1：将 PIS 通道 0 作为输入
Reserved	Bit 7-2	—	保留
USART1_RXD_SEL	Bit 1	R/W	**USART1 RXD 输入选择** 0：从端子 RXD 输入 1：将 PIS 通道 4 作为输入
USART0_RXD_SEL	Bit 0	R/W	**USART0 RXD 输入选择** 0：从端子 RXD 输入 1：将 PIS 通道 1 作为输入

10. 5. 2. 10 USART0 输出调制控制寄存器（USART0_TXMCR）

USART0 输出调制控制寄存器（USART0_TXMCR）

偏移地址：60_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-9	—	保留
TXMLVLS	Bit 8	R/W	**TX 调制电平选择** 0：低电平调制（TX 与所选取的调制信号进行硬件或操作） 1：高电平调制（TX 与所选取的调制信号进行硬件与操作）
TXMSS	Bit 7-4	R/W	**TX 调制源选择** 0000：调制禁止 0001：AD16C4T0 0010：BUZ 0011：GP16C2T0 0100：GP16C2T1 0101：GP16C2T2 其余：无调制
TXSIGS	Bit 3-0	R/W	**TX 调制信号选择** **TXMSS=0000** TXSIGS 无效 **TXMSS=0001, 0011,0100,0101 （调制源**

			=AD16C4T0, GP16C2T0, GP16C2T1,GP16C2T2） 0000：TIMER 通道 1 0001：TIMER 通道 2 0010：TIMER 通道 3 0011：TIMER 通道 4 **TXMSS=0010（调制源=BUZ）** 调制信号为 BUZ 输出

第11章 独立看门狗（IWDT）

11. 1 概述

独立看门狗 IWDT 可用来检测软件和硬件异常引起的故障，如主时钟停振、用户程序异常无法喂狗等；当计数器达到给定的超时值时，将触发系统复位。

独立看门狗 IWDT，当硬件使能时，时钟强制为独立的 LRC 时钟，且用户无法通过软件关闭 IWDT。

独立看门狗 IWDT 最适合于独立于主程序之外，并且对时间精度要求较低的场合。

11. 2 特性

- 支持硬件使能和关闭看门狗
  - 芯片配置位 CFG_WORD0.IWDTEN 配置为 1，则硬件使能 IWDT
  - 芯片配置位 CFG_WORD0.IWDTEN 位配置为 0，则硬件关闭 IWDT，但可以软件使能 IWDT
  - 硬件使能后不可通过软件关停
  - 硬件使能后 IWDT 时钟强制为 32KHz LRC 时钟
- IWDT 溢出时间可设定
  - 写入 IWDT_LOAD 寄存器将重新加载看门狗
  - 溢出时产生 IWDT 复位
- IWDT 中断可唤醒 STOP 模式

11. 3 功能描述

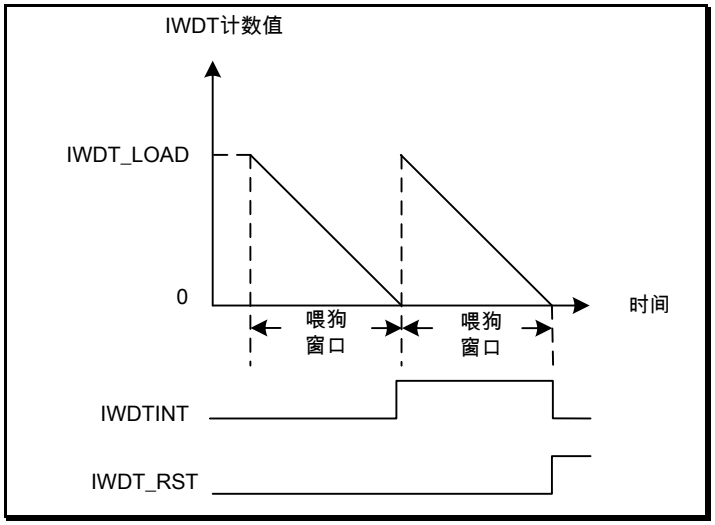


图 12-1 独立看门狗时序图

11.3.1 硬件看门狗

IWDT 看门狗可用于检测软件和硬件异常，用户可通过 CFG_WORD0.IWDTEN 配置位启用硬件看门狗功能，以提高系统健壮性；硬件看门狗使能后，时钟强制为 32KHz LRC 时钟，即使系统时钟失效，IWDT 仍可正常工作。

当硬件看门狗使能时，系统上电后看门狗立即运行（时钟固定为 32KHz LRC 时钟）；IWDT 载入 IWDT_LOAD 默认值 0x0000_8000（约 1s，对应复位时间约 2s），并从默认值递减计数。

计数器计数到 0 时，IWDT 产生中断标志，并在下一计数时钟到来时，计数器再次载入 IWDT_LOAD，并继续递减计数；当计数器再次计数到 0 时，如果 IWDT 中断标志未被清零，IWDT 将产生复位信号。

IWDT_INTCLR 写入任意值，则清除中断标志位，并重新载入计数初值，进行递减计数。

喂狗操作

修改 IWDT_LOAD.LOAD 值，以确定喂狗间隔；

开启 IWDG_IRQn 中断服务，并使能 IWDT 中断（IWDT_CON.IE=1）；

在中断服务中检查中断标志位（IWDT_RIS.WDTIF）是否被置起；

如果 IWDT_RIS.WDTIF 标志位置起，则执行喂狗操作；

11.3.2 软件看门狗

当硬件看门狗禁止时，系统上电看门狗不运行，但可通过软件配置 IWDT_CON.EN=1 使能看门狗，俗称软件看门狗。

当软件看门狗使能时，计数器载入 IWDT_LOAD 值，并开始递减计数；当计数到 0 时，IWDT 产生中断标志，并在下一个计数时钟到来时，计数器再次载入 IWDT_LOAD，并继续递减计数；当计数器再次计数到 0 时，如果 IWDT 中断标志未被清零，IWDT 将产生复位信号。

IWDT_INTCLR 写入任意值，则清除中断标志位，并重新载入计数初值，进行递减计数。

操作流程

1. 开启 IWDT 中断服务，并使能 IWDT 中断（IWDT_CON.IE=1）；
2. 修改 IWDT_LOAD.LOAD 值，以确定喂狗间隔；
3. 选择 IWDT 时钟源 IWDT_CON.CLKS；
4. 使能软件看门狗 IWDT_CON.IE=1；
5. 在中断服务中检查中断标志位（IWDT_RIS.WDTIF）是否被置起；
6. 如果 IWDT_RIS.WDTIF 标志位置起，则执行喂狗操作；

注：软件看门狗可以通过 IWDT_CON.CLKS 选择时钟源，系统提供了 LRC 和 PCLK 两种时钟源供选择；

11.3.3 调试模式

当系统进入调试模式时，通过 DBG_APB2FZ.IWDT_STOP 配置，可以在 IWDT 内核停止时，暂停计数，以便查询系统内部状态，查询完成后，恢复程序执行。

11.3.4 寄存器访问保护

IWDT_LOAD、IWDT_CON、IWDT_INTCLR 寄存器具有写保护功能，对其修改时，必须先向 IWDT_LOCK 写入 0x1ACCE551 移除写保护；当 IWDT_LOCK 写入其他任意值时，寄存器重新被保护。

11. 4 特殊功能寄存器

11. 4. 1 寄存器列表

IWDT 寄存器列表		
名称	偏移地址	描述
IWDT_LOAD	0000_H	IWDT 计数器装载值寄存器
IWDT_VALUE	0004_H	IWDT 计数器当前值寄存器
IWDT_CON	0008_H	IWDT 控制寄存器
IWDT_INTCLR	$000C_H$	IWDT 中断标志清除寄存器
IWDT_RIS	0010_H	IWDT 中断标志寄存器
IWDT_LOCK	0100_H	IWDT 锁定寄存器

11.4.2 寄存器描述

11.4.2.1 IWDT计数器装载值寄存器（IWDT_LOAD）

IWDT 计数器装载值寄存器（IWDT_LOAD）																															
偏移地址：00_H																															
复位值：$00000000\_00000000\_10000000\_00000000_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LOAD																															

LOAD<31:0>	Bit 31-0	W	**IWDT 计数器重载值** 计数范围 0x0000_0001~0xFFFF_FFFF。如果为 0，IWDT 不计数。

11.4.2.2 IWDT计数器当前值寄存器（IWDT_VALUE）

IWDT 计数器当前值寄存器（IWDT_VALUE）																															
偏移地址：04_H																															
复位值：$11111111\_ 11111111\_ 11111111 \_ 11111111_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VALUE																															

VALUE	Bit 31-0	R	**IWDT 计数器当前值** 读取时返回 IWDT 计数器的当前计数值

11.4.2.3 IWDT控制寄存器（IWDT_CON）

IWDT 控制寄存器（IWDT_CON）																															
偏移地址：08_H																															
复位值：$00000000\_00000000\_00000000\_00000000_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																												CLKS	RSTEN	IE	EN

Reserved	Bit 31-4	—	**保留**
CLKS	Bit 3	R/W	**IWDT 计数时钟选择位（仅在配置位 CFG_WORD0.IWDTEN=0 时有效）** 0：PCLK

			1：LRC 时钟（32KHz）
RSTEN	Bit 2	R/W	**IWDT 复位使能位（仅在配置位 CFG_WORD0.IWDTEN=0 时有效）** 0：禁止 1：使能，IWDT 计数到 0 时，产生复位信号，将芯片复位
IE	Bit 1	R/W	**IWDT 中断使能位（仅在配置位 CFG_WORD0.IWDTEN=0 时有效）** 0：禁止 1：使能，IWDT 计数到 0 时，产生中断标志
EN	Bit 0	R/W	**IWDT 模块使能位（仅在配置位 CFG_WORD0.IWDTEN=0 时有效）** 0：禁止 1：使能

注 1：IWDT_CON 寄存器中的各个控制位，仅在配置字 CFG_WORD 的配置位 IWDTEN=0 时才有效。

注 2：如果 IWDT 使用 LRC 时钟计数，则程序不能在 3 个 LRC 时钟周期内连续两次或多次喂狗，否则可能会导致后续无法正常喂狗，误产生 IWDT 计数溢出复位，所以用户程序中的喂狗时间间隔应大于 3 个 LRC 时钟周期，或者确保两次喂狗的时间间隔小于 IWDT 计数溢出时间，也不会误产生 IWDT 计数溢出复位；如果 IWDT 使用 PCLK 时钟计数，则无需考虑两次喂狗的最小时间间隔。

11. 4. 2. 4 IWDT中断标志清除寄存器（IWDT_INTCLR）

IWDT 中断标志清除寄存器（IWDT_INTCLR）

偏移地址：$0C_H$

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

INTCLR<31:0>	Bit 31-0	W	**IWDT 中断标志清 0 位** 对 IWDT_INTCLR 寄存器进行任意写操作，IWDT 中断标志位均被清零，计数器重载 IWDT_LOAD 寄存器值，继续递减计数

11. 4. 2. 5 IWDT中断标志寄存器（IWDT_RIS）

IWDT 中断标志寄存器（IWDT_RIS）

偏移地址：10_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-1	—	保留
WDTIF	Bit 0	R	**IWDT 中断标志位** 0：未产生中断 1：IWDT 计数器计数到 0，产生中断 写寄存器 IWDT_INTCLR，可清除 IWDT 中断标志位

11. 4. 2. 6 IWDT锁定寄存器（IWDT_LOCK）

IWDT 锁定寄存器（IWDT_LOCK）

偏移地址：14_H

复位值：$00000000\_00000000\_00000000\_00000001_B$

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

Reserved	Bit 31-1	—	保留
LOCK	Bit 0	R/W	**IWDT 寄存器保护状态位** 0：IWDT 寄存器处于未保护状态 1：IWDT 寄存器处于保护状态 对 IWDT_LOCK 寄存器写入 0x1ACCE551，被保护的寄存器处于未保护状态；写入其它值，处于保护状态

第12章 窗口看门狗（WWDT）

12. 1 概述

窗口看门狗 WWDT 通常用来监视，由外部干扰或不可预见的逻辑条件造成的应用程序背离运行序列而产生的软件故障。

窗口看门狗 WWDT 对于过早或过晚喂狗都将产生 WWDT 复位，可用于检测软件没有喂狗或在禁止喂狗区内的喂狗行为，防止程序运行至不可控状态。

12. 2 特性

窗口看门狗 WWDT 功能概述：

- 可编程的递减计数器
- 支持设定喂狗禁止区
  - 通过 WWDT_CON.WWDTWIN 设置喂狗禁止区
    - WWDTWIN 寄存器设定为 11 时，任何区域喂狗均不产生复位
  - 窗口外喂狗产生 WWDT 复位
  - 窗口内产生 WWDT 中断
    - WWDT 中断可用作喂狗请求
- 安全可靠
  - 当配置位 CFG_WORD0.WWDTEN 为 1 时，一旦软件使能，则只能通过复位关断
- WWDT 溢出长度可设定
  - 可通过 WWDT_LOAD.LOAD 寄存器设定喂狗窗口溢出长度
  - 溢出时产生 WWDT 复位
- WWDT 中断可用作喂狗请求，不能唤醒 STOP 模式

12. 3 功能描述

12. 3. 1 窗口看门狗

对于窗口看门狗 WWDT，过早或过晚喂狗都将产生 WWDT 复位，可用于检测软件的过晚或过早行为，防止程序跑至不可控状态，可通过 WWDT 复位消除。如中断异常，程序不断进入一个带喂狗指令的子程序的情况。

用户可根据程序正常执行的时间设定喂狗窗口，可检测到程序未按正常次序执行，跳过某些程序的异常情况。

窗口看门狗时钟源

窗口看门狗 WWDT 对时间窗口有一定的要求，系统内提供了 LRC 和 PCLK 作为 WWDT 时钟，可根据自己喂狗精度的需要，选择合适的时钟源。

用户可通过寄存器 WWDT_CON.CLKS 进行时钟源选择。

工作流程

系统上电后，窗口看门狗不启动，配置 WWDT_LOAD.LOAD，设置计数初值，配置 WWDT_CON.CLKS 选择时钟源，通过配置 WWDT_CON.EN 使能窗口看门狗；

使能后，WWDT 计数器载入 WWDT_LOAD.LOAD 的 1/4，并开始递减计数，当计数到 0 时，窗口计数器加 1；

在下一个计数时钟到来时，计数器再次载入 WWDT_LOAD.LOAD 寄存器值的 1/4，并继续递减计数；

WWDTWIN 设置为 25%，则窗口计数器为 1 时，WWDT 产生中断标志；

WWDTWIN 设置为 50%，则窗口计数器为 2 时，WWDT 产生中断标志；

WWDTWIN 设置为 75%，则窗口计数器为 3 时，WWDT 产生中断标志；

WWDT 产生中断后，直至窗口计数器计数到 4 之前（即累计计数等于 WWDT_LOAD），如果没有在喂狗窗口内进行喂狗，则 WWDT 模块将产生复位信号，如下图所示；

喂狗窗口内寄存器 WWDT_INTCLR.INTCLR 写入任意值，将清除中断标志位，并新载入计数初值进行递减计数；

注：若配置字中的 CFG_WORD0.WWDTEN 位配置为 1，则软件使能窗口看门狗之后，不可再通过软件关闭窗口看门狗。

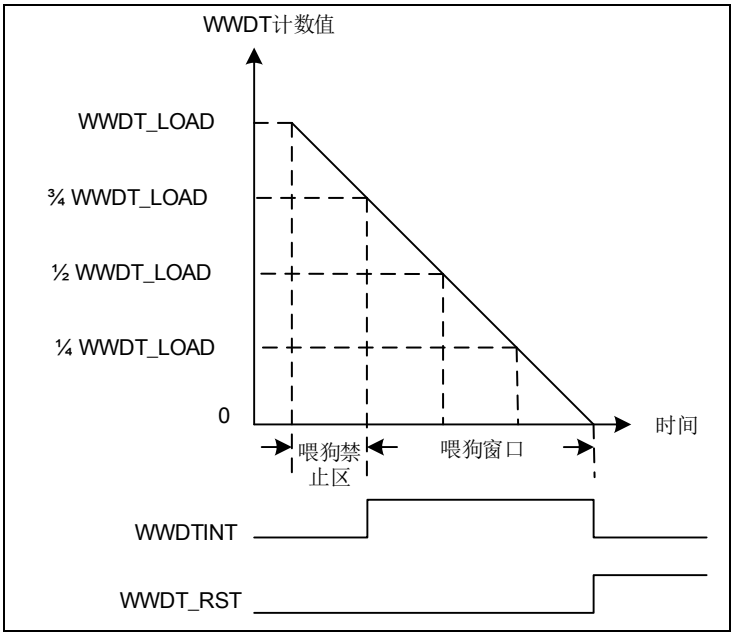


图 13-1 窗口看门狗中断和溢出复位产生时序图（WWDTWIN 设定为 25%）

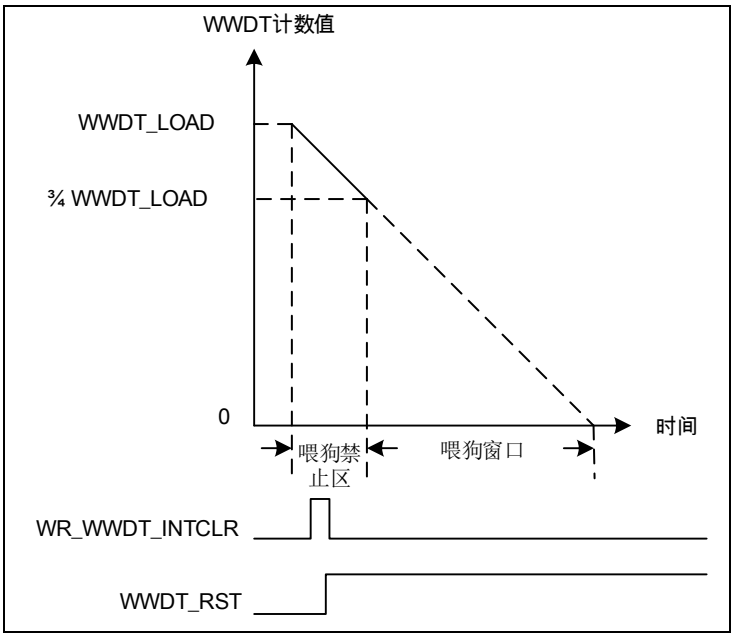


图 13-2 错误的喂狗时序图（WWDTWIN 设定为 25%）

12. 3. 2 调试模式

当系统进入调试模式时，通过 DBG_APB2FZ.WWDT_STOP 调试暂停选择位配置，可以在 WWDT 内核停止时，暂停计数，以便查询系统内部状态，查询完成后，恢复程序执行。

12. 3. 3 寄存器访问保护

WWDT_LOAD、WWDT_CON、WWDT_INTCLR 寄存器具有写保护功能，对其修改时，必须先向 WWDT_LOCK 写入 0x1ACCE551 移除写保护；当 WWDT_LOCK 写入其他任意值时，寄存器重新被保护。

12. 4 特殊功能寄存器

12. 4. 1 寄存器列表

WWDT 寄存器列表		
名称	偏移地址	描述
WWDT_LOAD	0000_H	WWDT 计数器装载值寄存器
WWDT_VALUE	0004_H	WWDT 计数器当前值寄存器
WWDT_CON	0008_H	WWDT 控制寄存器
WWDT_INTCLR	$000C_H$	WWDT 中断标志清除寄存器
WWDT_RIS	0010_H	WWDT 中断标志寄存器
WWDT_LOCK	0100_H	WWDT 锁定寄存器

12. 4. 2 寄存器描述

12. 4. 2. 1 WWDT计数器装载值寄存器（WWDT_LOAD）

WWDT 计数器装载值寄存器（WWDT_LOAD）																															
偏移地址：00_H																															
复位值：$00000000\_00000010\_00000000\_00000000_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LOAD[31:0]																															

LOAD	Bit 31-0	W	**WWDT 计数器重载值** 计数范围 0x0000_0001~0xFFFF_FFFF。如果为 0，WWDT 不计数。

12. 4. 2. 2 WWDT计数器当前值寄存器（WWDT_VALUE）

WWDT 计数器当前值寄存器（WWDT_VALUE）																															
偏移地址：04_H																															
复位值：$00111111\_\ 11111111\_\ 11111111\ \_\ 11111111_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VALUE																															

VALUE	Bit 31-0	R	**WWDT 计数器当前值** 读取时返回 WWDT 计数器的当前计数值，其中高两位为窗口计数器当前值

12. 4. 2. 3 WWDT控制寄存器（WWDT_CON）

WWDT 控制寄存器（WWDT_CON）																															
偏移地址：08_H																															
复位值：$00000000\_00000000\_00000000\_00000000_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																										WWDTWIN		CLKS	RSTEN	IE	EN

Reserved	Bit 31-6	—	保留
WWDTWIN	Bit 5-4	R/W	**WWDT 禁止喂狗窗口选择位**

			00：25%窗口内禁止喂狗，窗口内喂狗产生复位 01：50%窗口内禁止喂狗，窗口内喂狗产生复位 10：75%窗口内禁止喂狗，窗口内喂狗产生复位 11：不禁止喂狗，喂狗将使看门狗计数器重载
CLKS	Bit 3	R/W	**WWDT 计数时钟选择位** 0：PCLK 1：LRC 时钟（32KHz）
RSTEN	Bit 2	R/W	**WWDT 复位使能位** 0：禁止 1：使能，WWDT 计数到 0 时，产生复位信号，将芯片复位
IE	Bit 1	R/W	**WWDT 中断使能位** 0：禁止 1：使能，WWDT 计数到 0 时，产生中断标志
EN	Bit 0	R/W	**WWDT 模块使能位** 0：禁止 1：使能

注：如果 WWDT 使用 LRC 时钟计数，则程序不能在 3 个 LRC 时钟周期内连续两次或多次喂狗，否则可能会导致后续无法正常喂狗，推荐在 WWDT 中断服务程序中喂狗。

12. 4. 2. 4 WWDT中断标志清除寄存器（WWDT_INTCLR）

WWDT 中断标志清除寄存器（WWDT_INTCLR）

偏移地址：$0C_H$

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

INTCLR	Bit 31-0	W	**WWDT 中断标志清 0 位** 对 WWDT_INTCLR 寄存器进行任意写操作，WWDT 中断标志位均被清零，计数器重载 WWDT_LOAD 寄存器值，继续递减计数

12. 4. 2. 5 WWDT中断标志寄存器（WWDT_RIS）

WWDT 中断标志寄存器（WWDT_RIS）

偏移地址：10_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-1	—	保留
WWDTIF	Bit 0	R	**WWDT 中断标志位** 0：未产生中断 1：WWDT 计数器计数到 0，产生中断 写寄存器 WWDT_INTCLR，可清除 WWDT 中断标志位

12. 4. 2. 6 WWDT锁定寄存器（WWDT_LOCK）

WWDT 锁定寄存器（WWDT_LOCK）

偏移地址：100_H

复位值：$00000000\_00000000\_00000000\_00000001_B$

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

Reserved	Bit 31-1	—	保留
LOCK	Bit 0	R/W	**WWDT 寄存器保护状态位** 0：WWDT 寄存器处于未保护状态 1：WWDT 寄存器处于保护状态 对 WWDT_LOCK 寄存器写入 0x1ACCE551，被保护的寄存器处于未保护状态；写入其它值，处于保护状态

第13章 通用IO及端口控制（GPIO）

13. 1 概述

每组通用 GPIO 端口包含最多 16 个独立的引脚。这些引脚可单独配置为输入或输出，每个引脚输出输入功能中可配置为开漏输出、推挽输出以及浮空输入、带滤波输入模式，配置为输出模式时还可选择每个引脚的驱动强度。

GPIO 引脚可复用为外设功能端口，例如 PWM 输出口、USART 通信口或模拟输入，每个外设均支持复用到多个引脚上。GPIO 端口支持最多 16 个异步外部中断，可被配置到任何一个 IO 引脚上。

13. 2 特性

- 可配置为输入或输出
- 输出模式可配置
  - 推挽/开漏
  - 上拉/下拉
- 输入模式
  - 端口浮空
  - 上拉/下拉
  - 模拟端口
- 支持端口输出数据的复位、置位或取反，可按位操作
- 支持复用为外设功能端口
- 输出驱动能力可配置：驱动能力选择
- 支持 16 个外部输入中断
- 支持端口配置写保护功能

13. 3 结构框图

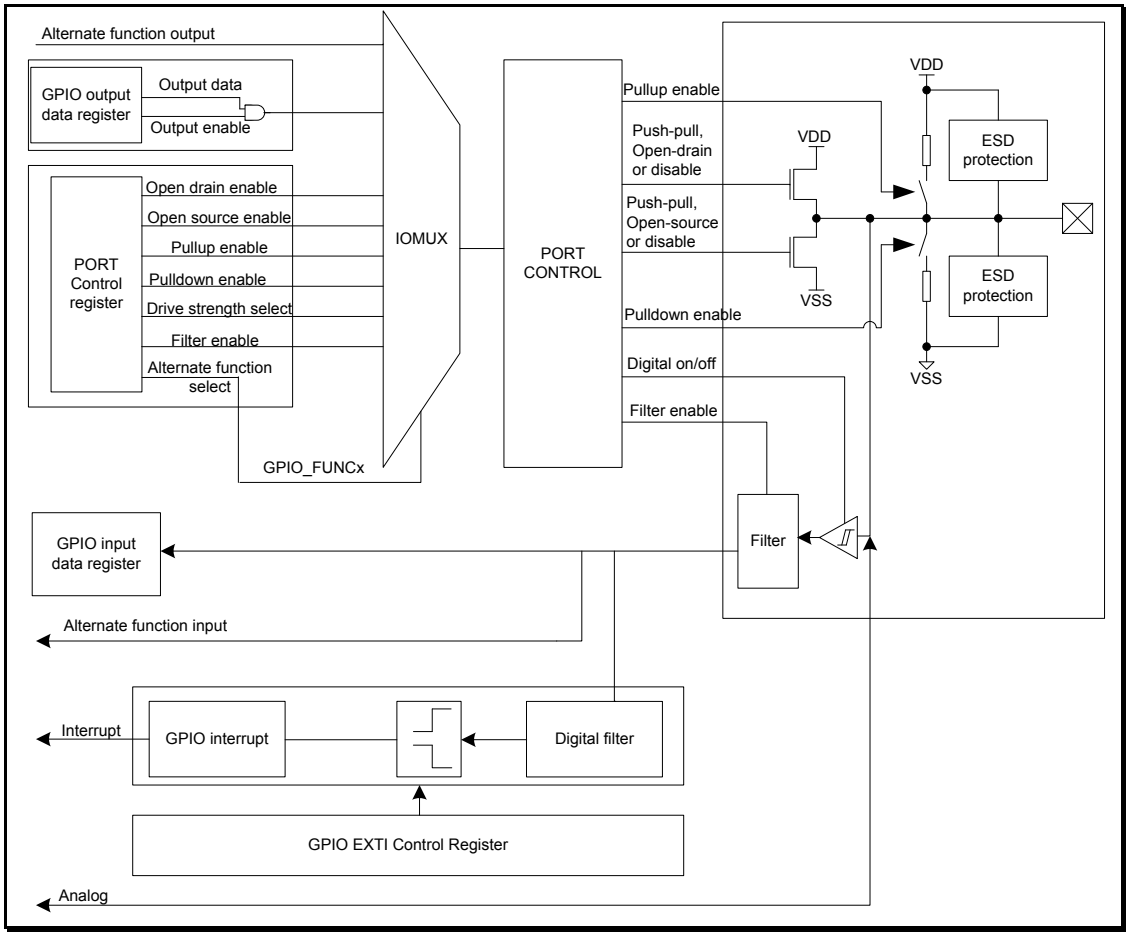


图 14-1 GPIO 结构框图

13. 4 功能描述

13. 4. 1 端口控制寄存器

每组 GPIO 最多有 16 个相对独立的引脚，每个引脚都可以通过寄存器自由配置。

通过配置 GPIO_MODE，可选择相应端口的模式，可配置为输入模式，输出模式和端口关闭模式。选择输出模式时，端口输入同样有效。选择端口关闭模式时，相应端口可被复用为模拟功能。

通过配置 GPIO_PUPD，可使能相应端口的上拉或/和下拉电阻。

通过配置 GPIO_OD，可将相应端口输出方式配置为推挽、漏极开路两种模式。在配置为漏极开路模式时，可使能相应端口的上拉，以保证正常输出，也可在端口外部连接上拉电阻。

通过配置 GPIO_ODRV，可选择相应端口的输出驱动能力，以满足不同的负载要求。

通过配置 GPIO_FLT，可使能相应端口的输入滤波功能，可滤除外部引线上高频信号干扰或毛刺。若输入需要较高的实时性，建议关闭输入滤波功能。

通过配置 GPIO_TYPE，可选择相应端口的输入类型，可选择 TTL 或 CMOS 两种模式。

通过配置 GPIO_FUNC，可选择相应 GPIO 的复用功能，可选择 FUNC_ALT0 ~ FUNC_ALT7。复用功能 GPIO 为输入时必须配置为输入模式，且需外部驱动；复用功能 GPIO 为输出时必须配置为输出模式，推挽或开漏；复用功能 GPIO 为双向模式时，端口必须配置为输出模式，推挽或开漏。

通过配置 GPIO_LOCK，可锁定相应端口的控制寄存器数值。直到下一次 CPU 复位锁定才可被解除。端口数据寄存器不受锁定的控制。

13.4.2 端口数据寄存器

软件可通过读取 GPIO_DIN 才获知端口的电平状态，若相应端口输入滤波被使能，则读到的是端口滤波之后的状态。

通过配置 GPIO_DOUT，可选择端口输出电平值，若端口模式已配置为输出，则该值所对应的电平会在管脚上立即生效。

通过配置 GPIO_BSRR，可按位改写端口输出电平值。对置位寄存器某些位进行写入 1 可置位相应端口，写入 0 的位不会影响相应端口的输出电平。对复位寄存器某些位进行写入 1 可复位相应端口，写入 0 的位不会影响相应端口的输出电平。若同时将某位置位和复位，则置位的优先级更高。

通过配置 GPIO_BIR，可按位翻转端口输出电平值。对翻转寄存器某些位进行写入 1 可将相应端口电平值翻转，写入 0 的位不会影响相应端口的输出电平。

13.4.3 外部端口中断

通过配置 GPIO_EXTIRER，可设置外部中断上升沿触发使能或禁止，其中低 16bit 为上升沿中断触发使能位，高 16bit 为保留位。

通过配置 GPIO_EXTIFER，可设置外部中断下降沿触发使能或禁止，其中低 16bit 为下降沿中断触发使能位，高 16bit 为保留位。

通过配置 GPIO_EXTIEN，可设置外部中断使能或禁止，其中低 16bit 为中断使能设置位，外部中断对应 bit 位配置为 1 表示中断使能，配置为 0 表示中断禁止，高 16bit 为保留位。

通过 GPIO_EXTIFLAG 寄存器，可检测当前有效中断。外部中断对应 bit 位为 1 表示检测到有效中断，为 0 表示未检测到有效中断。该寄存器为只读。

通过配置 GPIO_EXTISFR，可将外部端口中断标志位置位。其中低 16bit 为中断标志位置位位，为 1 表示置位中断标志位，为 0 无操作，高 16bit 为保留位。

通过配置 GPIO_EXTICFR，可将外部端口中断标志位清除。其中低 16bit 为中断标志位清除位，为 1 表示清零中断标志位，为 0 无操作，高 16bit 为保留位

通过配置 GPIO_EXTIPSR0 和 GPIO_EXTIPSR1 可选择外部中断的 GPIO 端口，每个外部中断可选择 2 个不同的 GPIO 口（PAn~PBn）。

通过配置 GPIO_EXTIFLTCR，可设置外部中断滤波参数。GPIO_EXTIFLTCR.FLTEN 位可使能相应外部端口中断滤波功能，配置 GPIO_EXTIFLTCR.FLTSEL 位可设置滤波时间。

13.4.4 通用GPIO配置

每个 GPIO 都可以由软件设置为输出模式或输入模式，也可以复用为外设功能接口。所有 GPIO 端口都有内部上拉或下拉，应用中可以激活也可以断开。

端口配置表				
配置模式		MODEn[1:0]	PUPDn[1:0]	ODOSn[1:0]
输出	推挽输出	1x	xx	0x
	开漏输出	1x	xx	1x
输入	浮空输入	01	00	xx
	上拉输入	01	01	xx
	下拉输入	01	10	xx
	模拟输入	01	禁止	xx

表 14-1 端口配置表

在输出模式中，还可以软件方式设置每个 GPIO 的驱动能力。输入模式中，可以软件方式设置每个 GPIO 的滤波特性。还可以软件选择 GPIO 管脚类型，有 CMOS 和 TTL 两种选择。

ODRVn[1:0]	意义
00	普通电流驱动
01	强电流驱动
10	预留
11	预留

表 14-2 端口驱动表

13.4.5 外部中断与唤醒

外部中断可以配达每个 GPIO，配置为外部中断的 GPIO 必须设置为输入模式。

每个外部中断通道可以独立的配置输入类型和对应的触发事件（上升沿触发、下降沿触发或双边触发），都可以独立的设置触发或禁止。

EXTI 主要特性如下：

- ◇ 每个外部中断都有独立的触发或禁止设置
- ◇ 每个中断通道有独立的状态标识位
- ◇ 支持多达 16 个外部中断请求
- ◇ 独立设置外部中断滤波特性

外部中断映像如下：

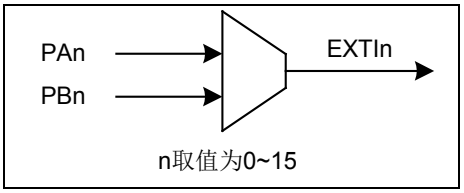


图 14-2 外部中断 GPIO 映像

应用说明：

要产生中断，必须配置好中断端口并使能，然后根据需要通过触发事件寄存器来设置边沿检测，另外根据应用需求可以设置输入滤波。当外部端口发生了预期的边沿时将产生一个中断请求，外部中断标志寄存器中对应的位随之被置 1，此时在外部中断标志清零寄存器对应标志位置 1，可以清除中断请求及对应标志位。

外部中断配置步骤如下：

1. 将对应 GPIO 端口配置为输入模式。
2. 配置外部中断端口选择寄存器 0 和外部中断端口选择寄存器 1 选择相应的 GPIO 端口。
3. 配置外部中断上升沿触发使能寄存器设置上升沿触发事件，配置外部中断下降沿触发使能寄存器设置下降沿触发事件。
4. 配置外部中断滤波控制寄存器设置需求对应的滤波特性。
5. 配置外部中断使能寄存器对应的中断使能位，使得外部中断可以有效响应。

13.4.6 外设功能端口复用

使用复用功能时必须先对 GPIO 端口复用功能寄存器 0 或 GPIO 端口复用功能寄存器 1 进行配置，对应的配置参数请查看数据手册管脚功能复用表。

- ◇ 当复用功能为输入时，端口必须配置为输入模式（浮空、上拉或下拉），且输入引脚由外部驱动。
- ◇ 当复用功能为输出时，端口必须配置为输出模式（推挽或开漏）。在配置为开漏模式时，GPIO 输入功能有效，此时可以用作双向模式。

如果端口复用功能为输出，则引脚会和片上外设的输出信号连接，如果对应外设没有被激活，GPIO 的输出信号将不确定。

13.4.7 GPIO锁定

芯片 GPIO 锁定机制的处理是将端口配置冻结。当一个端口执行了锁定程序后，对应 GPIO 控制寄存器数值将被锁定，在下一次复位之前，不能再被更改。

13.4.8 GPIO输入配置

当 GPIO 端口配置为输入模式时

- ◇ 输出缓冲器被禁止。
- ◇ 根据输入模式配置参数的不同，连接内部上拉、下拉。
- ◇ 输入数据采样到 GPIO 端口输入数据寄存器
- ◇ 访问 GPIO 端口输入数据寄存器即可得到 GPIOn 状态数据

13.4.9 GPIO输出配置

当 GPIO 端口配置为输出模式时：

- ◇ 输出缓冲器被激活
- ◇ 根据配置参数的不同，连接内部上拉、下拉

◇ 输入数据采样到 GPIO 端口输入数据寄存器

◇ 在开漏模式下，访问输入数据寄存器得到当前 GPIOn 的状态数据。

◇ 在推挽模式下，访问输出数据寄存器得到最后一次写的值。

13. 4. 10 模拟输入配置

当 GPIO 端口配置为模拟输入模式时：

◇ 输出缓冲器被禁止

◇ 施密特输入触发禁止

◇ 内部上拉或下拉禁止

◇ 访问输入数据寄存器得到的数据为全 0

13. 5 特殊功能寄存器

13. 5. 1 寄存器列表

GPIO 寄存器列表		
名称	偏移地址	描述
基地址： GPIOA_BASE（000_H） GPIOB_BASE（040_H）		
GPIO_DIN	000_H	GPIO 端口输入数据寄存器
GPIO_DOUT	004_H	GPIO 端口输出数据寄存器
GPIO_BSRR	008_H	GPIO 端口置位和复位寄存器
GPIO_BIR	$00C_H$	GPIO 端口翻转寄存器
GPIO_MODE	010_H	GPIO 端口模式寄存器
GPIO_OD	014_H	GPIO 端口开漏寄存器
GPIO_PUPD	018_H	GPIO 端口上拉和下拉寄存器
GPIO_ODRV	$01C_H$	GPIO 端口输出驱动寄存器
GPIO_FLT	020_H	GPIO 端口滤波寄存器
GPIO_TYPE	024_H	GPIO 端口类型寄存器
GPIO_FUNC0	028_H	GPIO 端口复用功能寄存器 0
GPIO_FUNC1	$02C_H$	GPIO 端口复用功能寄存器 1
GPIO_LOCK	030_H	GPIO 端口锁定寄存器
基地址：EXTI_BASE（300_H）		
GPIO_EXTIRER	000_H	外部中断上升沿触发使能寄存器
GPIO_EXTIFER	008_H	外部中断下降沿触发使能寄存器
GPIO_EXTIEN	010_H	外部中断使能寄存器
GPIO_EXTIFLAG	018_H	外部中断标志寄存器
GPIO_EXTISFR	020_H	外部中断标志置位寄存器
GPIO_EXTICFR	028_H	外部中断标志清零寄存器
GPIO_EXTIPSR0	030_H	外部中断端口选择寄存器 0
GPIO_EXTIPSR1	034_H	外部中断端口选择寄存器 1
GPIO_EXTIFLTCR	040_H	外部中断滤波控制寄存器

13. 5. 2 寄存器描述

13. 5. 2. 1 GPIO端口输入数据寄存器（GPIO_DIN）

GPIO 端口输入数据寄存器（GPIO_DIN）

偏移地址：00_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																DIN15	DIN14	DIN13	DIN12	DIN11	DIN10	DIN9	DIN8	DIN7	DIN6	DIN5	DIN4	DIN3	DIN2	DIN1	DIN0

Reserved	Bit 31-16	—	保留
DIN<y>	Bit 15-0	R	**GPIO 输入数据**

13. 5. 2. 2 GPIO端口输出数据寄存器（GPIO_DOUT）

GPIO 端口输出数据寄存器（GPIO_DOUT）

偏移地址：04_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																DOUT15	DOUT14	DOUT13	DOUT12	DOUT11	DOUT10	DOUT9	DOUT8	DOUT7	DOUT6	DOUT5	DOUT4	DOUT3	DOUT2	DOUT1	DOUT0

Reserved	Bit 31-16	—	保留
DOUT<y>	Bit 15-0	R/W	**GPIO 输出数据**

13. 5. 2. 3 GPIO端口置位和复位寄存器（GPIO_BSRR）

GPIO 端口置位和复位寄存器（GPIO_BSRR）																															
偏移地址：08_H																															
复位值：$00000000\_00000000\_00000000\_00000000_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BRR15	BRR14	BRR13	BRR12	BRR11	BRR10	BRR9	BRR8	BRR7	BRR6	BRR5	BRR4	BRR3	BRR2	BRR1	BRR0	BSR15	BSR14	BSR13	BSR12	BSR11	BSR10	BSR9	BSR8	BSR7	BSR6	BSR5	BSR4	BSR3	BSR2	BSR1	BSR0

BRR<y>	Bit 31-16	W	**GPIO 复位控制位** 0：无操作 1：相应位复位 注：如果同时对 GPIO 置位和复位，置位的优先级更高
BSR<y>	Bit 15-0	W	**GPIO 置位控制位** 0：无操作 1：相应位置位

13. 5. 2. 4 GPIO端口翻转寄存器（GPIO_BIR）

GPIO 端口翻转寄存器（GPIO_BIR）																															
偏移地址：$0C_H$																															
复位值：$00000000\_00000000\_00000000\_00000000_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																BIR15	BIR14	BIR13	BIR12	BIR11	BIR10	BIR9	BIR8	BIR7	BIR6	BIR5	BIR4	BIR3	BIR2	BIR1	BIR0

Reserved	Bit 31-16	—	**保留**
BIR<y>	Bit 15-0	W	**GPIO 翻转控制位** 0：无操作 1：相应位翻转

13. 5. 2. 5 GPIO端口模式寄存器（GPIO_MODE）

GPIO 端口模式寄存器（GPIO_MODE）																															
偏移地址：10_H																															
复位值：$00000000\_00000000\_00000000\_00000000_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MODE15		MODE14		MODE13		MODE12		MODE11		MODE10		MODE9		MODE8		MODE7		MODE6		MODE5		MODE4		MODE3		MODE2		MODE1		MODE0	

MODE<y>	Bit 31-0	R/W	GPIO 模式控制位 00：输入输出禁止 01，10：端口输入 1x：端口输出

13. 5. 2. 6 GPIO端口开漏寄存器（GPIO_OD）

GPIO 端口开漏寄存器（GPIO_OD）																															
偏移地址：14_H																															
复位值：$00000000\_00000000\_00000000\_00000000_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ODOS15		ODOS14		ODOS13		ODOS12		ODOS11		ODOS10		ODOS9		ODOS8		ODOS7		ODOS6		ODOS5		ODOS4		ODOS3		ODOS2		ODOS1		ODOS0	

ODOS<y>	Bit 31-0	R/W	**GPIO 开漏控制位** 0x：推挽输出 1x：开漏输出

13. 5. 2. 7 GPIO端口上拉和下拉寄存器（GPIO_PUPD）

GPIO 端口上拉和下拉寄存器（GPIO_PUPD）																															
偏移地址：18_H																															
复位值：$00000000\_00000000\_00000000\_00000000_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PUPD15		PUPD14		PUPD13		PUPD12		PUPD11		PUPD10		PUPD9		PUPD8		PUPD7		PUPD6		PUPD5		PUPD4		PUPD3		PUPD2		PUPD1		PUPD0	

PUPD<y>	Bit 31-0	R/W	**GPIO 上拉和下拉控制位** 00：无上拉和下拉 01：上拉 10：下拉 11：同时上拉和下拉

13. 5. 2. 8 GPIO端口输出驱动寄存器（GPIO_ODRV）

GPIO 端口输出驱动寄存器（GPIO_ODRV）																															
偏移地址：$1C_H$																															
复位值：$10101010\_10101010\_10101010\_10101010_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ODRV15		ODRV14		ODRV13		ODRV12		ODRV11		ODRV10		ODRV9		ODRV8		ODRV7		ODRV6		ODRV5		ODRV4		ODRV3		ODRV2		ODRV1		ODRV0	

ODRV<y>	Bits 31-0	R/W	**GPIO 输出驱动控制位** x0：普通电流驱动 x1：强电流驱动

13. 5. 2. 9 GPIO端口滤波寄存器（GPIO_FLT）

GPIO 端口滤波寄存器（GPIO_FLT）

偏移地址：20_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																FLT15	FLT14	FLT13	FLT12	FLT11	FLT10	FLT9	FLT8	FLT7	FLT6	FLT5	FLT4	FLT3	FLT2	FLT1	FLT0

Reserved	Bit 31-16	—	保留
FLT<y>	Bit 15-0	R/W	**GPIO 滤波控制位** 0：输入滤波禁止 1：输入滤波使能

13. 5. 2. 10 GPIO端口类型寄存器（GPIO_TYPE）

GPIO 端口类型寄存器（GPIO_TYPE）

偏移地址：24_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																TYPE15	TYPE14	TYPE13	TYPE12	TYPE11	TYPE10	TYPE9	TYPE8	TYPE7	TYPE6	TYPE5	TYPE4	TYPE3	TYPE2	TYPE1	TYPE0

Reserved	Bit 31-16	—	保留
TYPE<y>	Bit 15-0	R/W	**GPIO 类型选择位** 0：CMOS 1：TTL

13. 5. 2. 11 GPIO端口复用功能寄存器 0（GPIO_FUNC0）

GPIO 端口复用功能寄存器 0（GPIO_FUNC0）																															
偏移地址：28_H																															
复位值：$00000000\_00000000\_00000000\_00000000_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FSEL_IO7				FSEL_IO6				FSEL_IO5				FSEL_IO4				FSEL_IO3				FSEL_IO2				FSEL_IO1				FSEL_IO0			

FSEL_IO7	Bit 31-28	R/W	**GPIO<7>功能复用选择位** 请查看管脚功能复用表
FSEL_IO6	Bit 27-24	R/W	**GPIO<6>功能复用选择位** 请查看管脚功能复用表
FSEL_IO5	Bit 23-20	R/W	**GPIO<5>功能复用选择位** 请查看管脚功能复用表
FSEL_IO4	Bit 19-16	R/W	**GPIO<4>功能复用选择位** 请查看管脚功能复用表
FSEL_IO3	Bit 15-12	R/W	**GPIO<3>功能复用选择位** 请查看管脚功能复用表
FSEL_IO2	Bit 11-8	R/W	**GPIO<2>功能复用选择位** 请查看管脚功能复用表
FSEL_IO1	Bit 7-4	R/W	**GPIO<1>功能复用选择位** 请查看管脚功能复用表
FSEL_IO0	Bit 3-0	R/W	**GPIO<0>功能复用选择位** 请查看管脚功能复用表

13. 5. 2. 12 GPIO端口复用功能寄存器 1（GPIO_FUNC1）

GPIO 端口复用功能寄存器 1（GPIO_FUNC1）																															
偏移地址：$2C_H$																															
复位值：$00000000\_00000000\_00000000\_00000000_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FSEL_IO15				FSEL_IO14				FSEL_IO13				FSEL_IO12				FSEL_IO11				FSEL_IO10				FSEL_IO9				FSEL_IO8			

FSEL_IO15	Bit 31-28	R/W	**GPIO<15>功能复用选择位** 请查看管脚功能复用表
FSEL_IO14	Bit 27-24	R/W	**GPIO<14>功能复用选择位** 请查看管脚功能复用表
FSEL_IO13	Bit 23-20	R/W	**GPIO<13>功能复用选择位** 请查看管脚功能复用表
FSEL_IO12	Bit 19-16	R/W	**GPIO<12>功能复用选择位** 请查看管脚功能复用表
FSEL_IO11	Bit 15-12	R/W	**GPIO<11>功能复用选择位** 请查看管脚功能复用表
FSEL_IO10	Bit 11-8	R/W	**GPIO<10>功能复用选择位** 请查看管脚功能复用表
FSEL_IO9	Bit 7-4	R/W	**GPIO<9>功能复用选择位** 请查看管脚功能复用表
FSEL_IO8	Bit 3-0	R/W	**GPIO<8>功能复用选择位** 请查看管脚功能复用表

13. 5. 2. 13 GPIO端口锁定寄存器（GPIO_LOCK）

GPIO 端口锁定寄存器（GPIO_LOCK）

偏移地址：30_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY																LOCK15	LOCK14	LOCK13	LOCK12	LOCK11	LOCK10	LOCK9	LOCK8	LOCK7	LOCK6	LOCK5	LOCK4	LOCK3	LOCK2	LOCK1	LOCK0

KEY	Bit 31-16	W	**GPIO 锁定寄存器关键码** 注：检测到写入关键码 0x55AA，才可对 LOCK<y> 进行写操作，读该位始终为 0
LOCK<y>	Bit 15-0	R/W	**GPIO<y>锁定控制位** 0：相应端口未锁定 1：相应端口锁定 注：被锁定相应端口只允许改变输出数据，LOCK<y>一旦被置位，必须等到下一次 CPU 复位才被清除。

13. 5. 2. 14 外部中断上升沿触发使能寄存器（GPIO_EXTIRER）

外部中断上升沿触发使能寄存器（GPIO_EXTIRER）																															
偏移地址：00_H																															
复位值：$00000000\_00000000\_00000000\_00000000_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																EXTIRER15	EXTIRER14	EXTIRER13	EXTIRER12	EXTIRER11	EXTIRER10	EXTIRER9	EXTIRER8	EXTIRER7	EXTIRER6	EXTIRER5	EXTIRER4	EXTIRER3	EXTIRER2	EXTIRER1	EXTIRER0

Reserved	Bit 31-16	—	保留
EXTIRER<y>	Bit 15-0	R/W	**EXTI<y>上升沿触发使能位** 0：禁止 1：使能

13. 5. 2. 15 外部中断下降沿触发使能寄存器（GPIO_EXTIFER）

外部中断下降沿触发使能寄存器（GPIO_EXTIFER）																															
偏移地址：08_H																															
复位值：$00000000\_00000000\_00000000\_00000000_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																EXTIFER15	EXTIFER14	EXTIFER13	EXTIFER12	EXTIFER11	EXTIFER10	EXTIFER9	EXTIFER8	EXTIFER7	EXTIFER6	EXTIFER5	EXTIFER4	EXTIFER3	EXTIFER2	EXTIFER1	EXTIFER0

Reserved	Bit 31-16	—	保留
EXTIFER<y>	Bit 15-0	R/W	**EXTI<y>下降沿触发使能位** 0：禁止 1：使能

13. 5. 2. 16 外部中断使能寄存器（GPIO_EXTIEN）

外部中断使能寄存器（GPIO_EXTIEN）

偏移地址：10_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																EXTIEN15	EXTIEN14	EXTIEN13	EXTIEN12	EXTIEN11	EXTIEN10	EXTIEN9	EXTIEN8	EXTIEN7	EXTIEN6	EXTIEN5	EXTIEN4	EXTIEN3	EXTIEN2	EXTIEN1	EXTIEN0

Reserved	Bit 31-16	—	保留
EXTIEN<y>	Bit 15-0	R/W	**EXTI<y>中断使能位** 0：禁止 1：使能

13. 5. 2. 17 外部中断标志寄存器（GPIO_EXTIFLAG）

外部中断标志寄存器（GPIO_EXTIFLAG）

偏移地址：18_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																EXTIFLAG15	EXTIFLAG14	EXTIFLAG13	EXTIFLAG12	EXTIFLAG11	EXTIFLAG10	EXTIFLAG9	EXTIFLAG8	EXTIFLAG7	EXTIFLAG6	EXTIFLAG5	EXTIFLAG4	EXTIFLAG3	EXTIFLAG2	EXTIFLAG1	EXTIFLAG0

Reserved	Bit 31-16	—	保留
EXTIFLAG<y>	Bit 15-0	R	**EXTI<y>中断状态位** 0：未检测到有效中断 1：检测到有效中断

13. 5. 2. 18 外部中断标志置位寄存器（GPIO_EXTISFR）

外部中断标志置位寄存器（GPIO_EXTISFR）

偏移地址：20_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																EXTISFR15	EXTISFR14	EXTISFR13	EXTISFR12	EXTISFR11	EXTISFR10	EXTISFR9	EXTISFR8	EXTISFR7	EXTISFR6	EXTISFR5	EXTISFR4	EXTISFR3	EXTISFR2	EXTISFR1	EXTISFR0

Reserved	Bit 31-16	—	保留
EXTISFR<y>	Bit 15-0	W1	**EXTI<y>中断标志位置位** 0：无操作 1：置位中断标志位 注：对该位写 1 将中断标志置位，写 0 无效

13. 5. 2. 19 外部中断标志清零寄存器（GPIO_EXTICFR）

外部中断标志清零寄存器（GPIO_EXTICFR）

偏移地址：28_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																EXTICFR15	EXTICFR14	EXTICFR13	EXTICFR12	EXTICFR11	EXTICFR10	EXTICFR9	EXTICFR8	EXTICFR7	EXTICFR6	EXTICFR5	EXTICFR4	EXTICFR3	EXTICFR2	EXTICFR1	EXTICFR0

Reserved	Bit 31-16	—	保留
EXTICFR<y>	Bit 15-0	W1	**EXTI<y>中断标志位清零** 0：无操作 1：清零中断标志位 注：对该位写 1 将中断标志复位，写 0 无效

13. 5. 2. 20 外部中断端口选择寄存器 0（GPIO_EXTIPSR0）

外部中断端口选择寄存器 0（GPIO_EXTIPSR0）																															
偏移地址：30_H																															
复位值：$00000000\_00000000\_00000000\_00000000_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	EXTIS7			Reserved	EXTIS6			Reserved	EXTIS5			Reserved	EXTIS4			Reserved	EXTIS3			Reserved	EXTIS2			Reserved	EXTIS1			Reserved	EXTIS0		

EXTIS7	Bit 31-28	R/W	**EXTI<7>端口选择位** 0000：PA7 0001：PB7 其他：保留
EXTIS6	Bit 27-24	R/W	**EXTI<6>端口选择位** 0000：PA6 0001：PB6 其他：保留
EXTIS5	Bit 23-20	R/W	**EXTI<5>端口选择位** 0000：PA5 0001：PB5 1111：PIS5 其他：保留
EXTIS4	Bit 19-16	R/W	**EXTI<4>端口选择位** 0000：PA4 0001：PB4 1111：PIS4 其他：保留
EXTIS3	Bit 15-12	R/W	**EXTI<3>端口选择位** 0000：PA3 0001：PB3 1111：PIS3 其他：保留
EXTIS2	Bit 11-8	R/W	**EXTI<2>端口选择位** 0000：PA2 0001：PB2 1111：PIS2 其他：保留
EXTIS1	Bit 7-4	R/W	**EXTI<1>端口选择位** 0000：PA1 0001：PB1 1111：PIS1 其他：保留

EXTIS0	Bit 3-0	R/W	**EXTI<0>端口选择位** 0000：PA0 0001：PB0 1111：PIS0 其他：保留

13. 5. 2. 21 外部中断端口选择寄存器 1（GPIO_EXTIPSR1）

外部中断端口选择寄存器 1（GPIO_EXTIPSR1）																															
偏移地址：34_H																															
复位值：$00000000\_00000000\_00000000\_00000000_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	EXTIS15			Reserved	EXTIS14			Reserved	EXTIS13			Reserved	EXTIS12			Reserved	EXTIS11			Reserved	EXTIS10			Reserved	EXTIS9			Reserved	EXTIS8		

EXTIS15	Bit 31-28	R/W	**EXTI<15>端口选择位** 0000：PA15 0001：PB15 其他：保留
EXTIS14	Bit 27-24	R/W	**EXTI<14>端口选择位** 0000：PA14 0001：PB14 1111：PIS5 其他：保留
EXTIS13	Bit 23-20	R/W	**EXTI<13>端口选择位** 0000：PA13 0001：PB13 1111：PIS4 其他：保留
EXTIS12	Bit 19-16	R/W	**EXTI<12>端口选择位** 0000：PA12 0001：PB12 其他：保留
EXTIS11	Bit 15-12	R/W	**EXTI<11>端口选择位** 0000：PA11 0001：PB11 1111：PIS3 其他：保留
EXTIS10	Bit 11-8	R/W	**EXTI<10>端口选择位** 0000：PA10 0001：PB10 1111：PIS2 其他：保留
EXTIS9	Bit 7-4	R/W	**EXTI<9>端口选择位** 0000：PA9 0001：PB9 1111：PIS1 其他：保留

EXTIS8	Bit 3-0	R/W	**EXTI<8>端口选择位** 0000：PA8 0001：PB8 1111：PIS0 其他：保留

13. 5. 2. 22 外部中断滤波控制寄存器（GPIO_EXTIFLTCR）

外部中断滤波控制寄存器（GPIO_EXTIFLTCR）																															
偏移地址：40_H																															
复位值：$00000000\_00000000\_00000000\_00000000_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								FLTSEL								FLTEN15	FLTEN14	FLTEN13	FLTEN12	FLTEN11	FLTEN10	FLTEN9	FLTEN8	FLTEN7	FLTEN6	FLTEN5	FLTEN4	FLTEN3	FLTEN2	FLTEN1	FLTEN0

Reserved	Bits 31-24	—	**保留**
FLTSEL	Bit 23-16	R/W	**EXTI 端口中断去抖滤波选择** 滤波时间 = (FLTSEL<7:0> + 1)×2 个 LRC 时钟周期
FLTEN<y>	Bit 15-0	R/W	**EXTI<y>端口中断去抖滤波使能** 0：禁止 1：使能 注：该位需在中断使能之前配置，使能之后禁止更改

第14章 循环冗余校验（CRC）

14. 1 概述

循环冗余校验（CRC）发生器可以执行带可编程多项式设定的 CRC 计算，用于对数据传输的完整性和正确性进行校验 。

14. 2 特性

- 支持四个常用的多项式：CRC-CCITT，CRC-8，CRC-16 和 CRC-32
  - CRC-CCITT：$X^{16} + X^{12} + X^5 + 1$
  - CRC-8：$X^8 + X^2 + X + 1$
  - CRC-16：$X^{16} + X^{15} + X^2 + 1$
  - CRC-32：$X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + X^8 + X^7 + X^5 + X^4 + X^2 + X + 1$
- 支持可编程的种子值
- 支持对输入数据和 CRC 校验值的可编程的反序设定
- 支持对输入数据和 CRC 校验值的可编程的反码设定
- 支持 8/16/32 位数据宽度
  - 8-bit 写模式：1 个 AHB 时钟周期操作
  - 16-bit 写模式：2 个 AHB 时钟周期操作
  - 32-bit 写模式：4 个 AHB 时钟周期操作
- 支持使用 DMA 写数据执行 CRC 操作

14. 3 结构框图

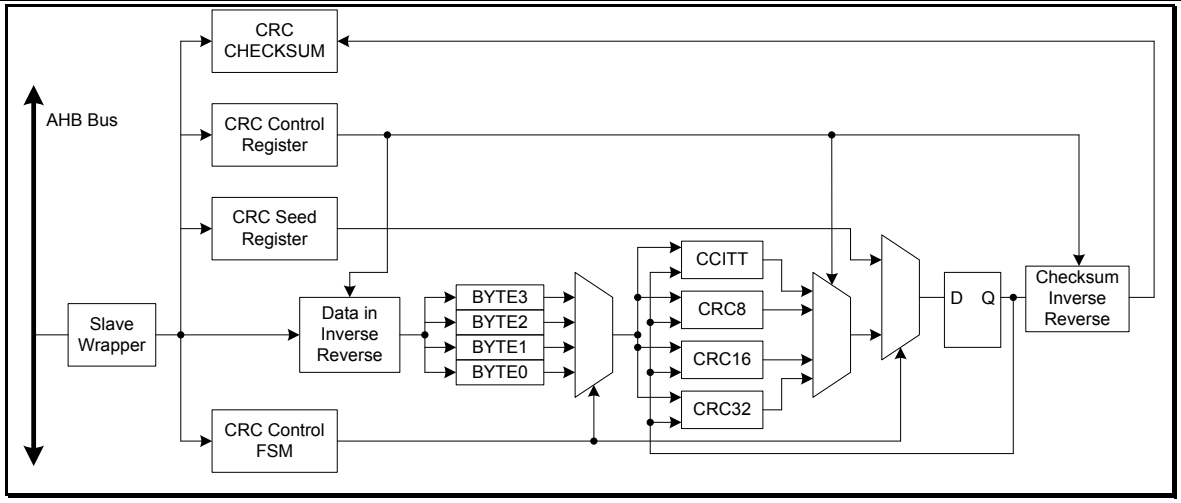


图 15-1 CRC 结构框图

14. 4 功能描述

14. 4. 1 常规操作

CRC 发生器可以执行带可编程多项式设定的 CRC 运算。多项式操作包括 CRC-CCITT，CRC-8，CRC-16 和 CRC-32。用户可以通过设置 MODE 选择 CRC 多项式操作模式。

操作示例：

1. 通过设置 CRC_CR.EN 使能 CRC 发生器
2. CRC 运算初始化设置
   a. 通过设置 CRC_CR.CHSINV 配置 CRC 校验值反码
   b. 通过设置 CRC_CR.CHSREV 配置 CRC 校验值位反序
   c. 通过设置 CRC_CR.DATINV 配置 CRC 写入数据反码
   d. 通过设置 CRC_CR.DATREV 配置 CRC 写入数据位反序
   e. 通过设置 CRC_CR.MODE 配置 CRC 校验模式
   f. 通过设置 CRC_CR.DATLEN 配置 CRC 写入数据长度
3. 通过设置 CRC_CR.RST 执行 CRC 复位，CRC 复位将装载初始种子值到 CRC 运算电路
4. 写数据到 CRC_DATA 寄存器来计算 CRC 校验值
5. 通过读 CRC_CHECKSUM 寄存器来获得 CRC 校验结果

14. 4. 2 DMA请求

DMA 请求（在使能后）仅用于数据传输，当需要减少 MCU 负荷时可以使用 DMA 功能，在计算完 CRC 后，DMA 模块将 CRC_CHECKSUM 里面的计算值传输到用户提供的存储区中。

操作示例：

1. 通过设置 CRC_CR.EN 使能 CRC 发生器
2. CRC 运算初始化设置
   a. 通过设置 CRC_CR.CHSINV 配置 CRC 校验值反码
   b. 通过设置 CRC_CR.CHSREV 配置 CRC 校验值位反序
   c. 通过设置 CRC_CR.DATINV 配置 CRC 写入数据反码
   d. 通过设置 CRC_CR.DATREV 配置 CRC 写入数据位反序
   e. 通过设置 CRC_CR.MODE 配置 CRC 校验模式
   f. 通过设置 CRC_CR.DATLEN 配置 CRC 写入数据长度
3. 通过设置 CRC_CR.RST 执行 CRC 复位，CRC 复位将装载初始种子值到 CRC 运算

电路

4. 用户先定义一段特殊类型的存储区，假设此存储区命名为 DESCR，把 DMA_CTRLBASE+x 的地址赋给此存储区首地址，此存储区数据类型参考 DMA 章节
5. 将 CRC_DATA 寄存器的地址填入 DESCR.DST，表示每发生一次 CRC 的 DMA 申请事件，数据搬运的目标地址，数据都会从需要计算 CRC 的数据存储区中传到 CRC_DATA 寄存器地址下
6. 设置 DESCR.DSR_INC，表示目标地址每一次发生 CRC 的 DMA 申请事件后的地址增加量
7. 将用户定义的数据存储区的地址填入 DESCR.SRC，表示每发生一次 CRC 的 DMA 申请事件后数据搬运的源地址，数据都会从该存储区中传入 CRC_DR 地址下
8. 设置 DESCR.SRC_INC,表示源地址每一次发生 CRC 的 DMA 申请事件后的地址增加量
9. 需要传输的总字节数减 1 填入 DESCR.N_MINUS_1，则每发生一次 CRC 的 DMA 申请事件后，该值都会递减，注意最大字节数为 1024 个字节
10. 设置 DMA_CHx_SELCON.MSEL 选择触发 DMA 的输入源为 CRC
11. 通道的优先级通过设置 DMA_CHPRSET 或 DMA_CHPRCLR 寄存器来配置
12. 以上步骤都执行完毕后，初始化阶段完成了，需要激活该 DMA 通道，设置 DMA_CHENSET 使能用户选择的通道
13. 使能 CRC_CR.DMAEN 位。

14. 5 特殊功能寄存器

14. 5. 1 寄存器列表

CRC 寄存器列表		
名称	偏移地址	描述
CRC_CR	0000_H	CRC 控制寄存器
CRC_DATA	0004_H	CRC 写数据寄存器
CRC_SEED	0008_H	CRC 种子寄存器
CRC_CHECKSUM	$000C_H$	CRC 校验值寄存器

14. 5. 2 寄存器描述

14. 5. 2. 1 CRC控制寄存器（CRC_CR）

CRC 控制寄存器（CRC_CR）

偏移地址：00_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved							BYTORD	DATLEN		MODE		CHSINV	DATINV	CHSREV	DATREV	Reserved											DMAEN	CWERR	WERR	RST	EN

Reserved	Bit 31-25	—	**保留**
BYTORD	Bit 24	R/W	**校验值字节顺序选择位** 0：优先校验低字节 1：优先校验高字节 注：优先校验低字节即优先校验Bit 7-0；优先校验高字节，16-bit模式即优先校验Bit 15-8，32-bit模式优先校验Bit 31-24；8-bit模式校验顺序无区别
DATLEN	Bit 23-22	R/W	**数据长度选择位** 00：通过写 CRC_DATA 寄存器方式自动判断 01：数据为 8-bit（CRC_DATA[7:0]有效） 10：数据为 16-bit（CRC_DATA[15:0]有效） 11：数据为 32-bit（CRC_DATA[31:0]有效）
MODE	Bit 21-20	R/W	**模式选择位** 00：CRC-CCITT 多项式模式 01：CRC-8 多项式模式 10：CRC-16 多项式模式 11：CRC-32 多项式模式
CHSINV	Bit 19	R/W	**校验值反码使能位** 0：禁止 1：使能
DATINV	Bit 18	R/W	**写数据反码使能位** 0：禁止 1：使能
CHSREV	Bit 17	R/W	**校验值反序使能位** 0：禁止 1：使能
DATREV	Bit 16	R/W	**写数据反序使能位** 0：禁止 1：使能
Reserved	Bit 15-5	—	**保留**
DMAEN	Bit 4	R/W	**DMA 使能位** 0：禁止

			1：使能
CWERR	Bit 3	W1	**CRC 写数据错误标志清除位** 0：无操作 1：清除标志位
WERR	Bit 2	R	**CRC 写数据错误标志位** 0：无错误 1：发生写数据错误 注：写入格式与 DATLEN 所选择不相符时会将标志位置位，未在最低字节或低半字写数据时也会将标志位置位。
RST	Bit 1	W1	**CRC 复位位** 0：无操作 1：复位 注：该位复位CRC内部状态机、DMAEN和缓存以及初始化种子值，但不会复位寄存器值
EN	Bit 0	R/W	**CRC 使能位** 0：禁止 1：使能

14. 5. 2. 2 CRC写数据寄存器（CRC_DATA）

CRC 写数据寄存器（CRC_DATA）

偏移地址：04_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

DATA	Bit 31-0	R/W	**CRC 写入数据位** 用户看可以通过 CPU 或 DMA 直接写数据到该位来执行 CRC 操作。 注 1：当写数据长度是 8-bit 模式，该位有效数据为 DATA[7:0]，如果数据长度是 16-bit 模式，该位有效数据为 DATA[15:0]。如果自动检测数据长度，则可通过最低字节、低半字或字写入方式任意写入数值。 注 2：当写入到错误的字节或半字时，硬件会置位写数据错误标志位 WERR。

14. 5. 2. 3 CRC种子寄存器（CRC_SEED）

CRC 种子寄存器（CRC_SEED）																															
偏移地址：08_H																															
复位值：$11111111\_11111111\_11111111\_11111111_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SEED																															

SEED	Bit 31-0	R/W	**CRC 种子值** 该位表示 CRC 校验种子值

14. 5. 2. 4 CRC校验值寄存器（CRC_CHECKSUM）

CRC 校验值寄存器（CRC_CHECKSUM）																															
偏移地址：$0C_H$																															
复位值：$00000000\_00000000\_00000000\_00000000_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CHECKSUM																															

CHECKSUM	Bit 31-0	R	**CRC 校验值** 该位表示 CRC 校验结果

第15章 高级定时器（AD16C4T）

15. 1 概述

高级控制定时器（AD16C4T）是一个功能强大、配置灵活的定时器模块，它包含一个 16-bit，定时器,具有定时、计数、脉冲输入信号测量（输入捕获）、产生特定 PWM 波（输出比较）等功能。

15. 2 特性

- 16 位递增，递减，递增/递减自动加载计数器
- 16 位可编程预分频器，可在定时器运行中对计数器工作时钟进行 1 到 65536 间的任意分频
- 带有四个独立通道，每个通道支持以下功能
  - 输入捕获
  - 输出比较
  - PWM 输出
  - 单脉冲输出
- 通道 1~3 支持互补输出，死区时间可配
- 在给定数目的计数周期之后更新重复计数寄存器
- 支持刹车功能，刹车后定时器输出状态可控
- 支持中断/DMA：
  - 更新事件：计数器上溢/下溢，计数器初始化
  - 触发事件
  - 换相事件
  - 输入捕获
  - 输出比较
  - 刹车输入
- 支持增量（正交）编码及霍尔电路
- 通过外设互联（PIS）可支持与片上其他定时器的互联工作

15. 3 结构框图

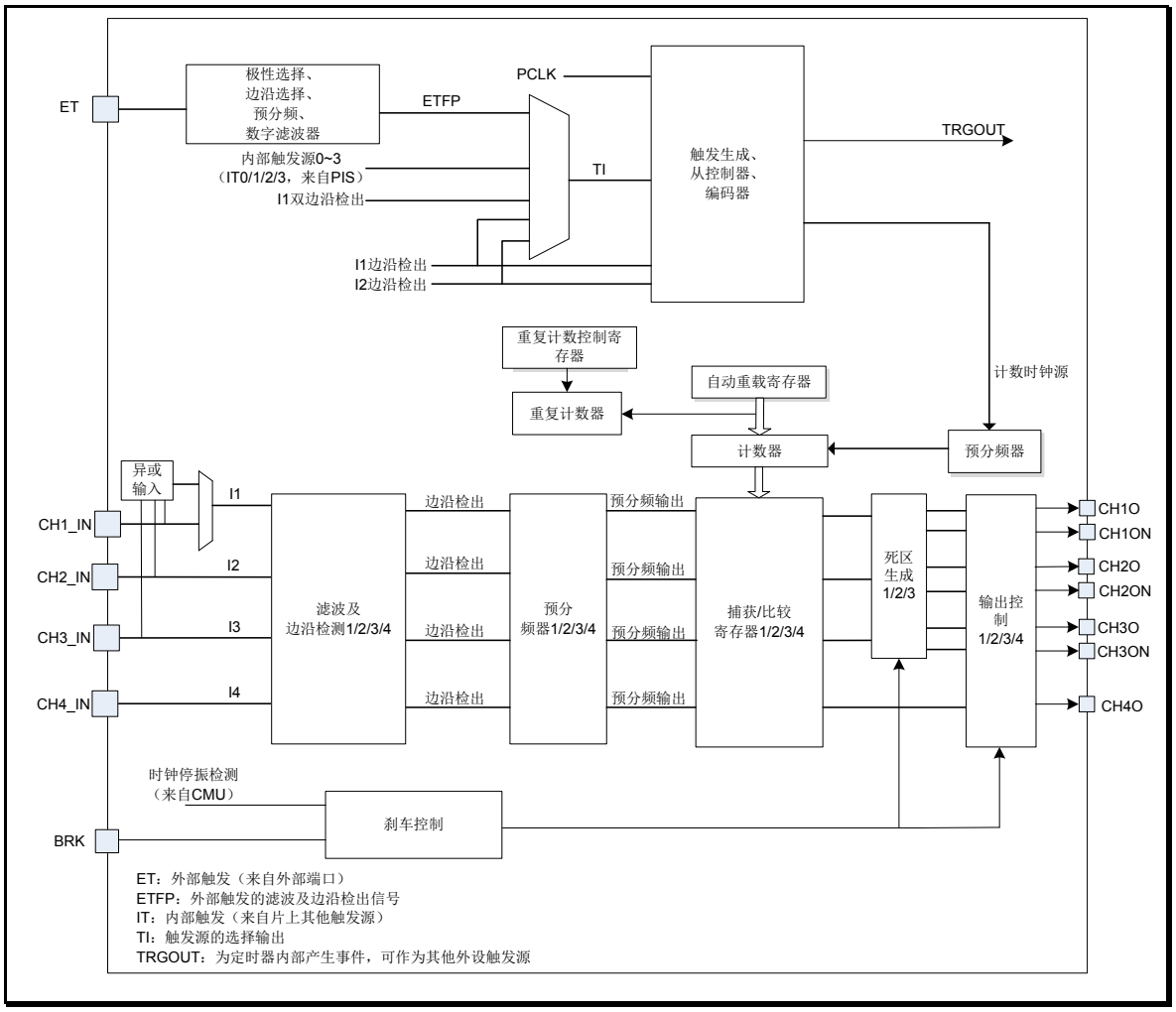


图 16-1　高级定时器电路结构框图

15. 4 功能描述

15. 4. 1　预分频器

定时器包含一个 16-bit 的计数器（AD16C4Tn_COUNT），计数时钟由预分频寄存器（AD16C4Tn_PRES）进行分频。计数周期由自动重载计数器（AD16C4Tn_AR）设定。重复计数寄存器则可指定计数周期数目（AD16C4Tn_REPAR）。

自动重载寄存器（AD16C4Tn_AR）是一个可缓存的寄存器。当 AD16C4Tn_CON1 寄存器的 ARPEN 位复位时，AD16C4Tn_AR 寄存器重载功能失效，AD16C4Tn_AR 就是有效寄存器；ARPEN 置位时，AD16C4Tn_AR 寄存器具有重载功能，产生更新事件（UEV）时，加载值（AD16C4Tn_AR 寄存器值）更新到影子寄存器。

当 AD16C4Tn_CON1 寄存器中 DISUE 位为 0 时，计数器计数上溢（或递减下溢）时会产生更新事件（UEV）。同样，软件方式也可产生更新事件。AD16C4Tn_CON1 寄存器的 CNTEN 置位时，计数器开始计数。

注：计数器在 CNTEN 位置位 1 个时钟周期后开始计数。

预分频器可对定时器工作时钟进行 AD16C4Tn_PRES 寄存器值+1 次分频。由于 AD16C4Tn_PRES 是一个可重载寄存器，因此，定时器工作时可以对该寄存器进行修改，修改值在下次更新事件（UEV）后有效。

下图给出了定时器运行过程中改变预分频值时计数器的计数情况。

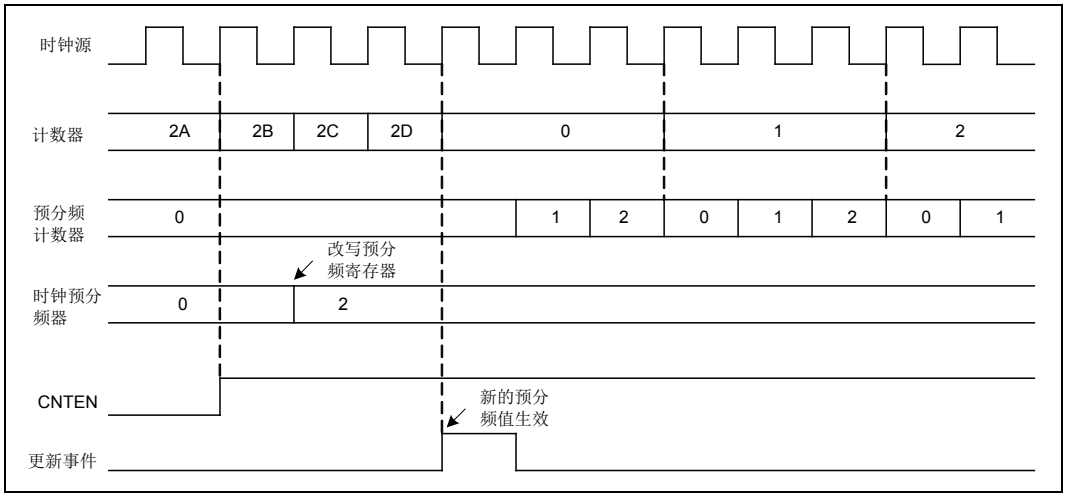


图 16-1　预分频值计数时序图

15. 4. 2 重复计数器

重复计数器用于控制发生多少次上溢或下溢次后产生更新事件。

重复计数器递减：

- ◇ 递增模式的每次上溢
- ◇ 递减模式的每次下溢
- ◇ 中心对齐模式时，计数器上溢与下溢。中心对齐模式限制了最大重复次数为 128 个 PWM 周期，每个 PWM 周期内可更新两次占空比。

AD16C4Tn_REPAR 寄存器是一个可缓存寄存器。软件（置位 AD16C4Tn_SGE 寄存器中的 SGU 位）或硬件从机模式控制方式产生更新事件时，无论重复计数器为何值，AD16C4Tn_REPAR 寄存器中值会立即更新到重复计数器的影子寄存器中。

中心对齐模式下，AD16C4Tn_REPAR 中值为奇数时，更新事件是在上溢或下溢时产生，取决于何时写 AD16C4Tn_REPAR 寄存器及何时开始计数。若在启动计数器前写 AD16C4Tn_REPAR，则上溢时产生 UEV，反之则在下溢时产生 UEV。

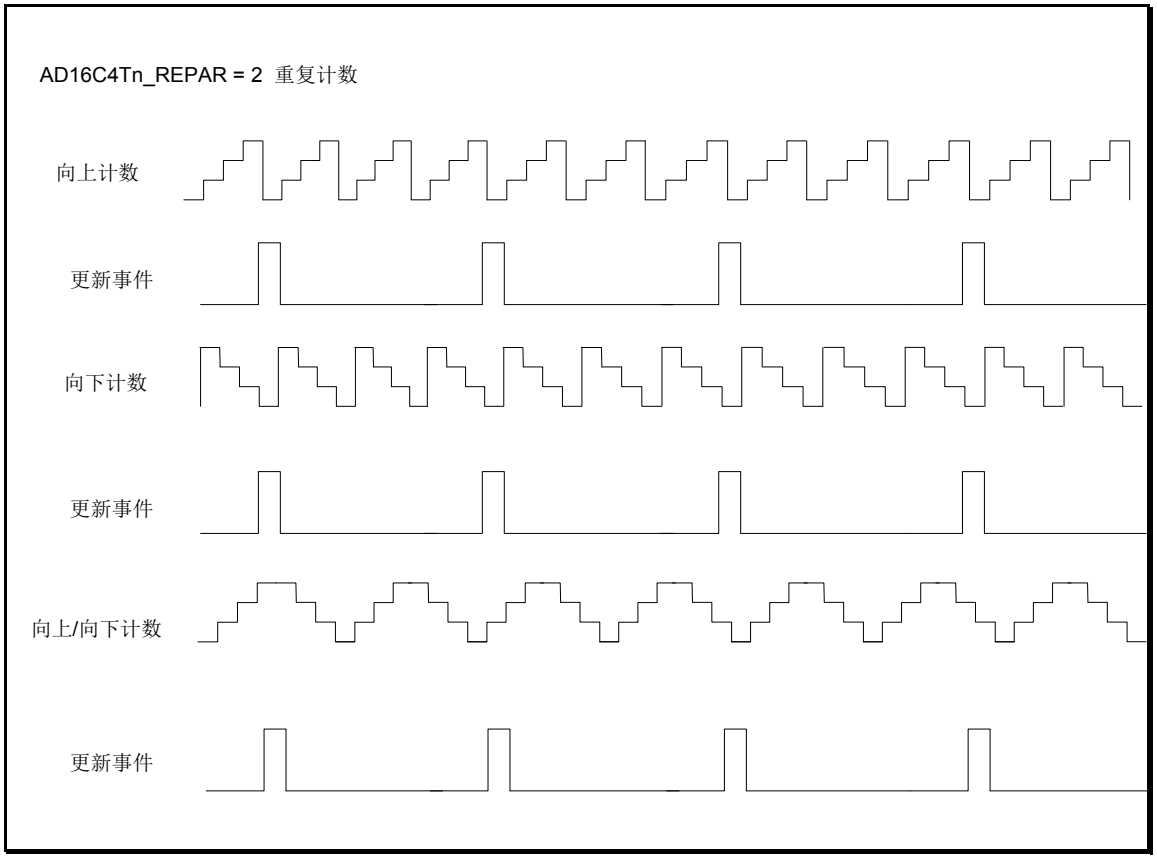


图 16-2 重复计数器工作模式

注意：置位 AD16C4Tn_SGE 寄存器中的 SGU 位也可以产生更新事件。

15. 4. 3 时钟源

计数器工作时钟可以可以选择内部时钟(INT_CLK)、外部时钟源 1（I1、I2、I3、I4）、外部时钟源 2（ET），内部触发输入（IT1、IT2、IT3、IT4）

15. 4. 3. 1 内部时钟源（INT_CLK）

若从模式控制器被关闭（AD16C4Tn_SMCON 寄存器的 SMODS= "000"），则 CNTEN，AD16C4Tn_CON1.DIRSEL 与 AD16C4Tn_SGE.SGU 位为实际控制位，这些位只能软件修改（SGU 位除外，仍硬件自动清除）。一旦 CNTEN 位被写为'1'，预分频器就由内部 INT_CLK 提供时钟。

下图给出了通常模式下控制电路和递增计数的情况，没有分频。

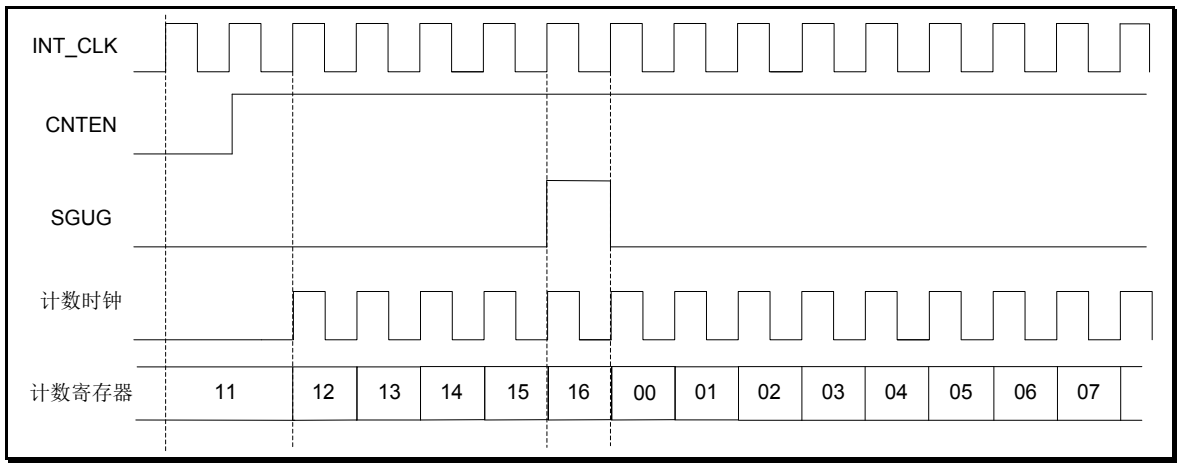


图 16-3 采用内部时钟计数

15. 4. 3. 2 外部时钟源 1

AD16C4Tn_SMCON 寄存器的 SMODS= "111"时，可选择外部时钟源 1。计数器可根据选定的上升沿或下降沿计数。

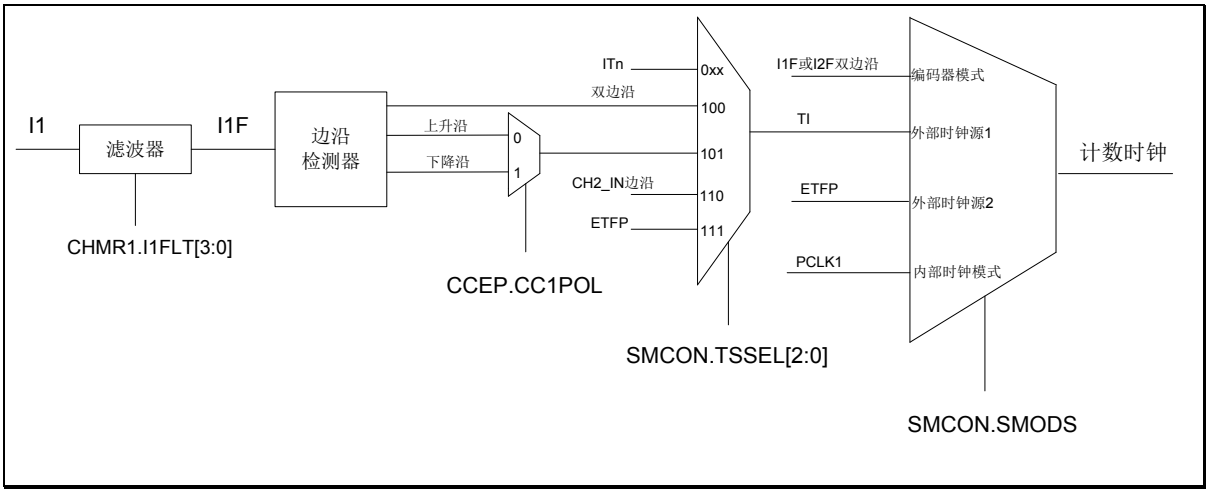


图 16-4 I1 外部时钟连接

配置计数器为外部时钟源 1，步骤如下：

1. AD16C4Tn_SMCON 寄存器中 SMODS = "111"，配置定时器外部时钟模式 1。
2. 设置 AD16C4Tn_SMCON 寄存器中的 TSSEL 选择外部时钟源。
3. 如外部时钟源为 I1，可配置 AD16C4Tn_CHMR1 寄存器 CC1SSEL = "01"，配置通道 1 检测 I1 输入的上升沿；设置 AD16C4Tn_CCEP 寄存器中 CC1POL = '0'，选择极性位上升沿。
4. 写 AD16C4Tn_CHMR1 寄存器的 I1FLT[3：0]位，配置输入滤波器时间（若没有滤波器需求，维持 I1FLT = "0000"）。
5. AD16C4Tn_CON1 寄存器中 CNTEN = '1'，使能计数器。

当 I1 上出现一次上升沿时，计数器计数一次且 TRGIF 标志位置位。

15. 4. 3. 3 外部时钟源 2

置位 AD16C4Tn_SMCON 寄存器的 ECM2EN 位选定外部时钟源 2。

计数器可对外部触发输入 ET 进行上升沿或下降沿计数。

下图给出了外部输入输入模块的概况。

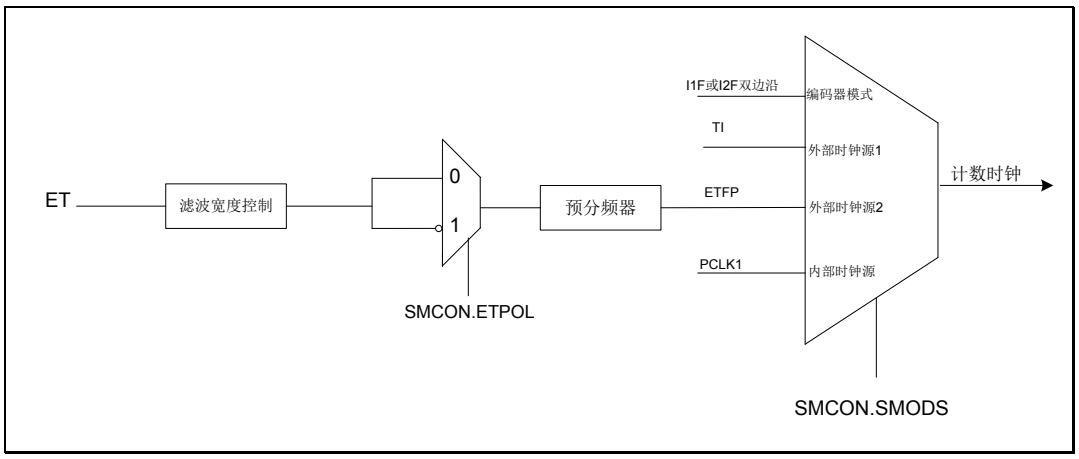


图 16-5 外部触发输入模块

配置计数器为外部时钟源 2，配置过程如下：

1. 设置 AD16C4Tn_SMCON 寄存器的 ETFLT[3：0]，配置输入滤波时间。
2. 设置 AD16C4Tn_SMCON 寄存器中 ETPSEL[1：0]，设置预分频器。
3. 设置 AD16C4Tn_SMCON 寄存器中 ETPOL，检测 ET 引脚上升沿或下降沿。
4. 设置 AD16C4Tn_SMCON 寄存器中 ECM2EN = '1'，使能外部时钟模式 2。
5. 设置 AD16C4Tn_CON1 寄存器的 CNTEN = '1'，使能计数器。

计数器每两个上升沿计一次数。

15. 4. 3. 4 内部触发输入（ITn）

当 AD16C4Tn_SMCON 寄存器的 SMODS= "111"，选定外部时钟模式 1。计数器根据选定的内部输入端的上升或下降沿计数。

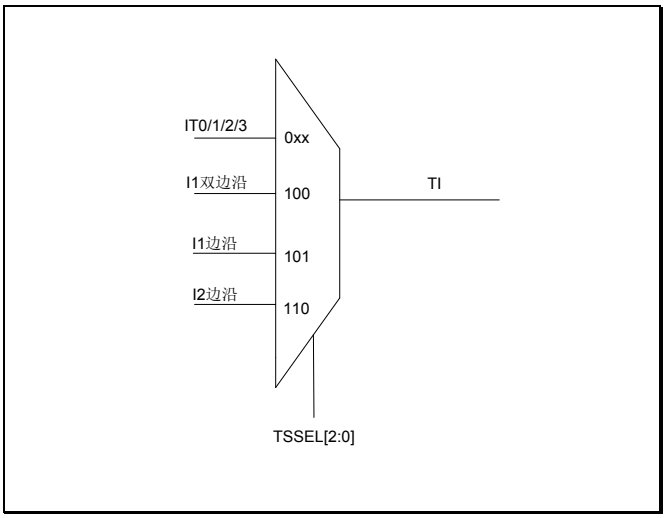


图 16-6 ITn 外部时钟连接

配置计数器在 ITn 输入端的上升沿递增计数，步骤如下：

1. AD16C4Tn_SMCON 寄存器中 SMODS = "111"，配置外部时钟模式 1。
2. AD16C4Tn_SMCON 寄存器的 TSSEL = "0xx"，选定 ITn 作为触发输入源。
3. AD16C4Tn_CON1 寄存器的 CNTEN = '1'，使能计数器。

ITn 产生上升沿时，计数器计数一次。ITn 上升沿与实际时钟间的延时，取决于 ITn 输入的再同步电路。

15. 4. 4 计数模式

15. 4. 4. 1 递增计数模式

当 AD16C4Tn_REPAR 寄存器值为 0 时，定时器配置为递增模式，计数器从 0 开始递增，直至 AD16C4Tn_AR 寄存器值；然后从 0 重新开始计数并产生一个更新事件（UEV）。当 AD16C4Tn_REPAR 寄存器不为 0 时，则在 AD16C4Tn_REPAR+1 次计数后产生更新事件。

当有更新事件（UEV）产生时，预装载寄存器会更新到影子寄存器，更新标志位（AD16C4Tn_RIF 寄存器中的 UEVTIF 位）置位（取决于 UERSEL 位）：

- ◇ 更新 AD16C4Tn_REPAR 寄存器的值到影子寄存器
- ◇ 更新 AD16C4Tn_AR 寄存器的值到影子寄存器
- ◇ 更新 AD16C4Tn_PRES 寄存器的值到影子寄存器

下图为 AD16C4Tn_AR = 0x16，预分频设为 2 分频时的计数器时序。

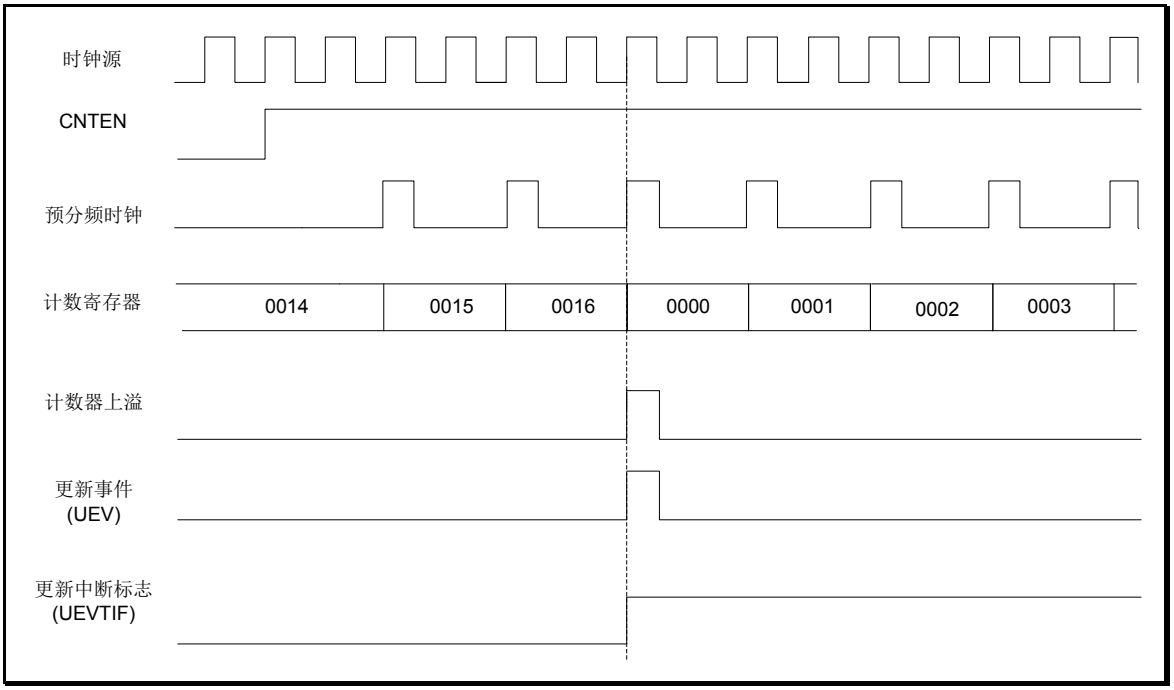


图 16-7　计数器递增计数时序图

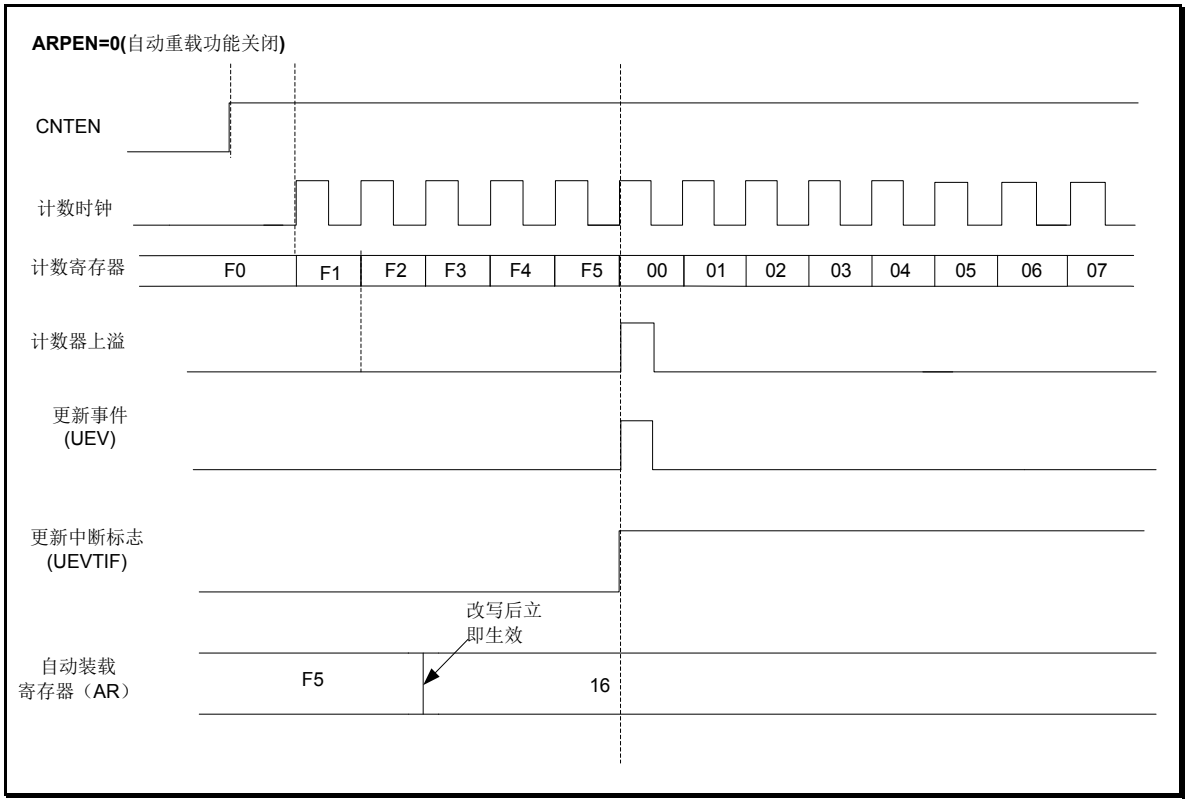


图 16-8　当 ARPEN=0 时计数器时序图

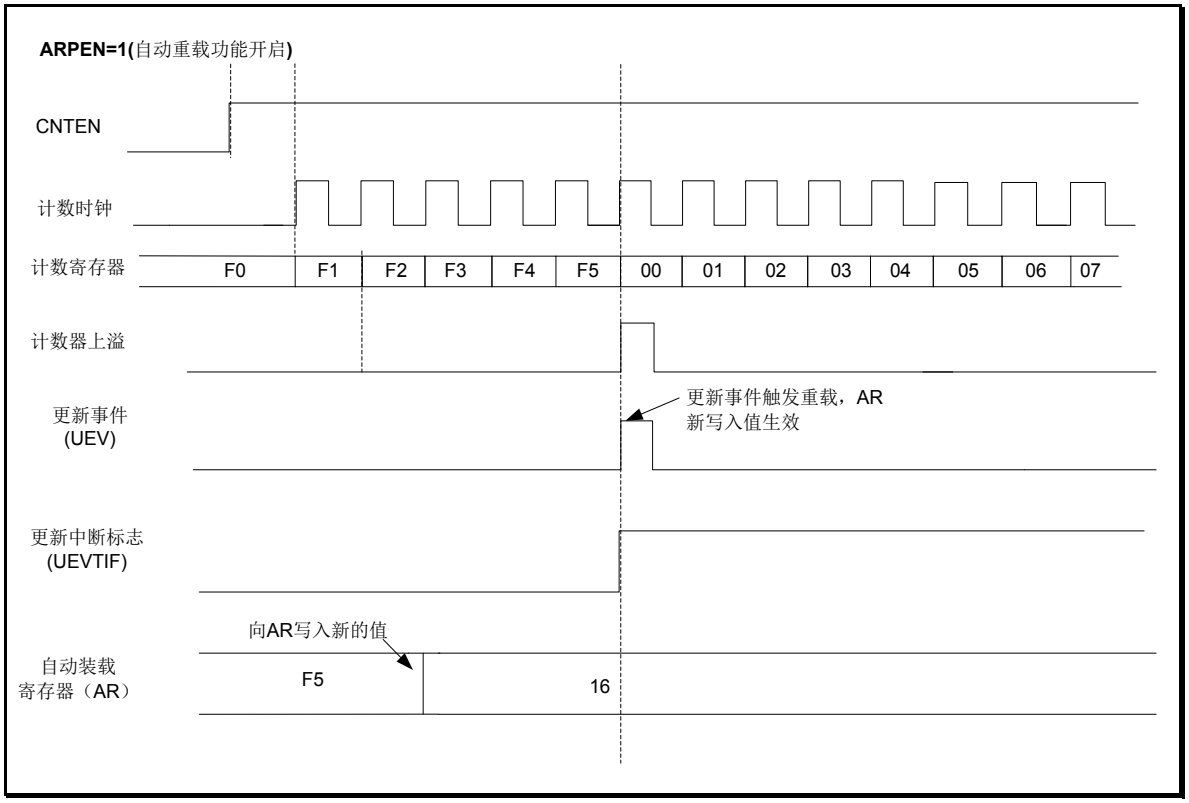


图 16-9　当 ARPEN=1 时计数器时序图

15. 4. 4. 2　　递减计数模式

当 AD16C4Tn_REPAR 寄存器值为 0 时，定时器配置为递减模式，计数器从 AD16C4Tn_AR 寄存器值开始递减至 0；然后重复递减并产生更新事件（UEV）。当 AD16C4Tn_REPAR 寄存器不为 0 时，则在 AD16C4Tn_REPAR+1 次后产生更新事件。

置位 AD16C4Tn_SGE 寄存器中的 SGU 位（通过软件或使用从机模式控制器）同样会产生更新事件。

当有更新事件（UEV）产生时，预载寄存器值会更新到影子寄存器，更新标志位（AD16C4Tn_RIF 寄存器中的 UEVTIF 位）置位（取决于 UERSEL 位）。

下图为 AD16C4Tn_AR = 0x27，预分频设为 1 分频时的计数器时序。

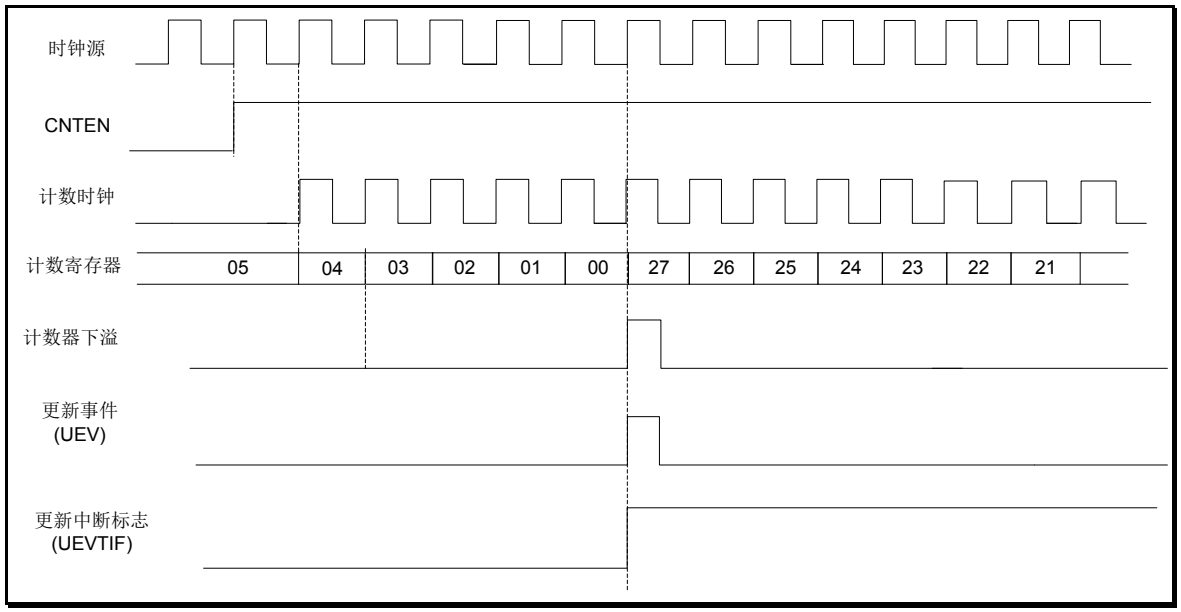


图 16-10　定时器递减计数时序图

15. 4. 4. 3　　中心对齐模式

当 AD16C4Tn_CON1 寄存器的 CMSEL 位的值不等于"00"时，定时器工作在中心对齐模式。定时器配置为中心对齐模式时，计数器先从 0 开始递增至 AD16C4Tn_AR 寄存器值减 1，并产生更新事件（UEV）；接着计数器从 AD16C4Tn_AR 寄存器值递减至 1，并产生更新事件。如此循环计数。计数器递减计数（中心对称模式 1，CMSEL="01"）、计数器递增计数（中心对称模式 2，CMSEL="10"）、计数器递增和递减计数（中心对称模式 3，CMSEL="11"），每个通道的输出比较中断标志位都会置位。

在中心对齐模式下，AD16C4Tn_CON1 寄存器的 DIRSEL 位无法进行写操作，该位由硬件自动更新指示当前计数方向。

计数上溢、下溢或者置位 AD16C4Tn_EGR 寄存器的 SGU 位（通过软件或使用从模式控制器）都会产生更新事件。因此，计数器和预分频器都会从 0 开始计数。

软件置位 AD16C4Tn_CON1 寄存器中的 DISUE 位可关闭更新事件（UEV）的产生。更新事件（UEV）关闭时，可避免向预载寄存器写新值时更新影子寄存器。DISUE 复位之前都不会产生更新事件。而在正常产生更新事件时，计数器仍然从 0 开始，同样预分频计数也是从 0 开始(但预分频值没有改变)。此外，若置位 AD16C4Tn_CON1 寄存器中的 UERSEL 位（更新请求选择），置位 SGU 位时会产生一次更新事件（UEV），但 UEVTIF 标志位不会置位（因此，不会触发中断或 DMA 请求）。这就避免了在捕获事件时，清除计数器值时产生更新和捕获中断。

当有更新事件（UEV）产生时，预载寄存器值会更新到影子寄存器，更新标志位（AD16C4Tn_RIF 寄存器中的 UEVTIF 位）置位（取决于 UERSEL 位）。

注：若更新源为计数器上溢，自动重载会在计数器重载前更新。因此，下一周期即为预期值（计数器载入新值）。

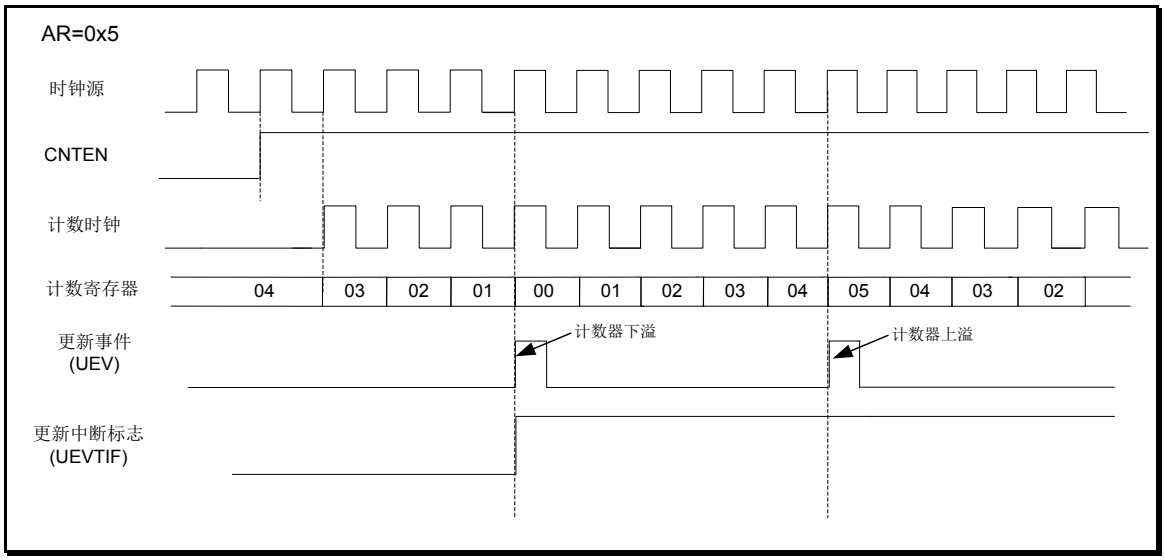


图 16-11 增减计数器时序图

15. 4. 5 捕获/比较通道

下方 3 张图为捕获/比较通道的概述。

输入电路对 In 输入端的信号进行采样，产生一个经过滤波的信号 InF。之后，一个可极性选择的边沿检测器产生 In 边沿检出信号，该信号可作为从模式控制器的触发输入或作为捕获控制命令，且信号经过分频后进入捕获寄存器。

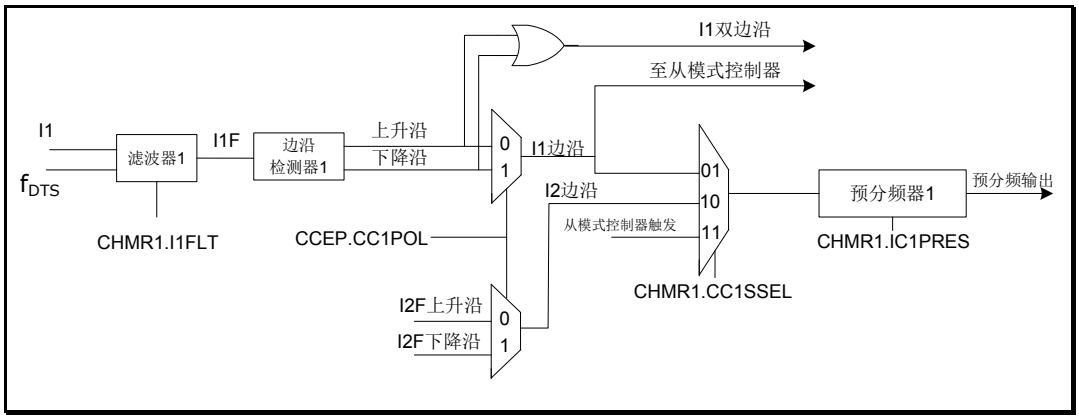


图 16-12 捕获/比较通道

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

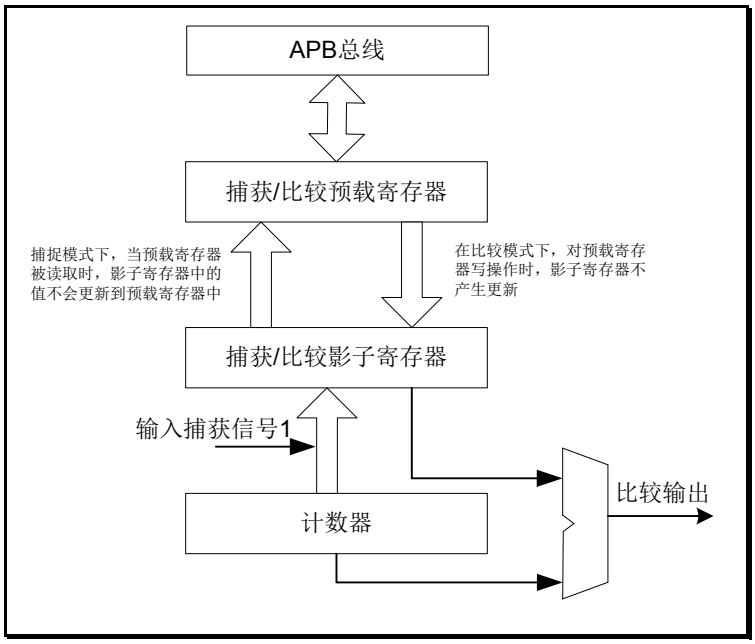


图 16-13 捕获/比较通道 1 结构图

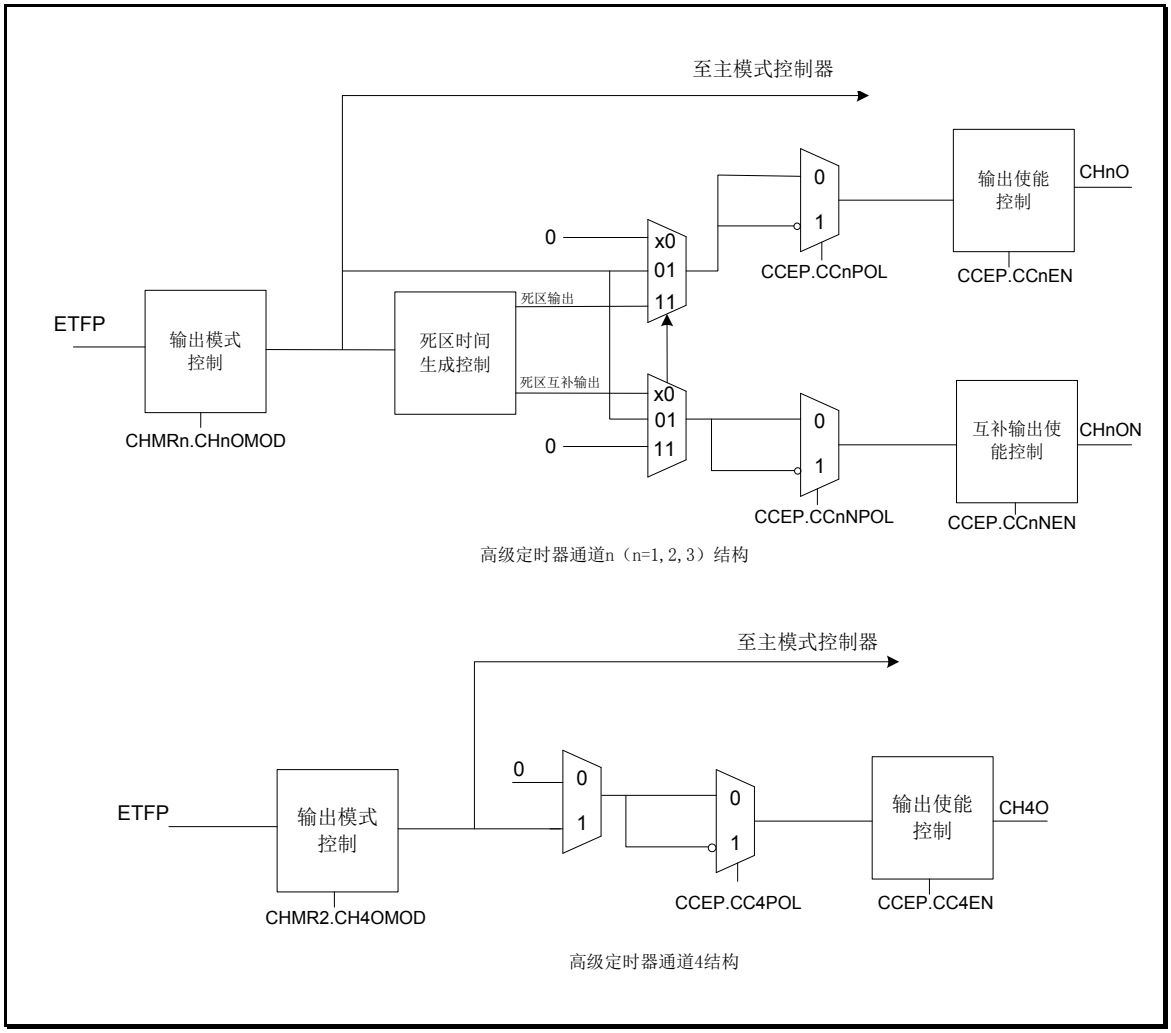


图 16-14 捕获/比较信道的输出部分

15. 4. 6 输入捕获模式

在输入捕获模式下，当检测到 In 上相应信号变化时，计数器的值就会被锁存到捕获/比较寄存器（AD16C4Tn_CCVALn）寄存器中。当捕获发生时，相应的 CHnIF 标志位（AD16C4Tn_RIF）会置位，同时会触发中断或 DMA（如果使能）请求。若发生捕获时，CHnIF 标志位已经置位，则过捕获 CHnOVIF 标志位（AD16C4Tn_RIF）置位。软件写'0'或读取 AD16C4Tn_CCVALn 寄存器中的捕获值都可以复位 CHnIF 标志位。对 CHnOVIF 位写'0'可清空该标志位。

以下为以 I1 输入上升沿作为捕获输入时的流程：

1. 选择有效输入端：AD16C4Tn_CCVAL1 必须连接到 I1 输入端，因此需将 AD16C4Tn_CHMR1 寄存器中的 CC1SSEL 位写"01"。只要 CC1SSEL 不为"00"，通道被配置为输入且 AD16C4Tn_CCVAL1 寄存器为只读。
2. 根据定时器连接的输入信号，配置输入滤波器的持续时间。当输入信号翻转时，前 5 个内部时钟信号内信号是不稳定的，因此必须配置滤波器的时间大于 5 个时钟周期。当 I1 检测到新的电平，连续 8 次采样可确认电平变化有效。
3. 选择 I1 信道的有效边沿变换。AD16C4Tn_CCEP 寄存器中的 CC1POL 写'0'（上升沿）。
4. 配置输入预分频器。
5. 置位 AD16C4Tn_CCEP 寄存器中的 CC1EN 位，使能捕获计数器的值到捕获寄存器。
6. 如有需要，置位 AD16C4Tn_IER 寄存器中的 CC1IT 位，使能中断请求。置位 AD16C4Tn_DMAEN 寄存器中的 CC1DS 位，使能 DMA 请求。

当发生输入捕获时：

1. 有效边沿产生，AD16C4Tn_CCVAL1 寄存器获取计数器的值。
2. CH1IF 标志位置位（中断标志）。若至少 2 个连续的捕获发生，但标志位没有及时清除，则 CH1OVIF 也会置位。
3. 中断的产生取决于 CC1IT 位。
4. DMA 请求的产生取决于 CC1DS 位。

为了处理捕获溢出，建议在读取捕获标志位前先读取捕获数据。这可以避免错过读取捕获标志位之后，读数据之前产生的捕获。

注：捕获中断请求可由软件设置 AD16C4Tn_SGE 寄存器中 SGCCnE 位产生。

15. 4. 6. 1 PWM输入模式

测量 I1 上 PWM 信号的周期和占空比的过程如下：

1. 为 AD16C4Tn_CCVAL1 选择有效的输入：AD16C4Tn_CHMR1 寄存器中的 CC1SSEL 位写"01"（I1 被选择）。

2. 为 I1 边沿检出选择有效的极性（用于捕获数据到 AD16C4Tn_CCVAL1 寄存器和计数器清零）：CC1POL 位写'0'（上升沿有效）。

3. 为 AD16C4Tn_CCVAL2 选择有效输入：AD16C4Tn_CHMR1 寄存器的 CC2SEL 位写"10"（I1 被选择）。

4. 为 I1 边沿检出选择有效极性（用于捕获数据到 AD16C4Tn_CCVAL2）：CC2POL 位写'1'。

5. 选择有效的触发输入：AD16C4Tn_SMCON 寄存器的 TSSEL 位写"101"（I1 边沿检出被选择）。

6. 配置从机模式控制器为复位模式：AD16C4Tn_SMCON 寄存器的 SMODS 位写"100"。

7. 使能捕获：AD16C4Tn_CCEP 寄存器的 CC1EN 位和 CC2EN 位写'1'。

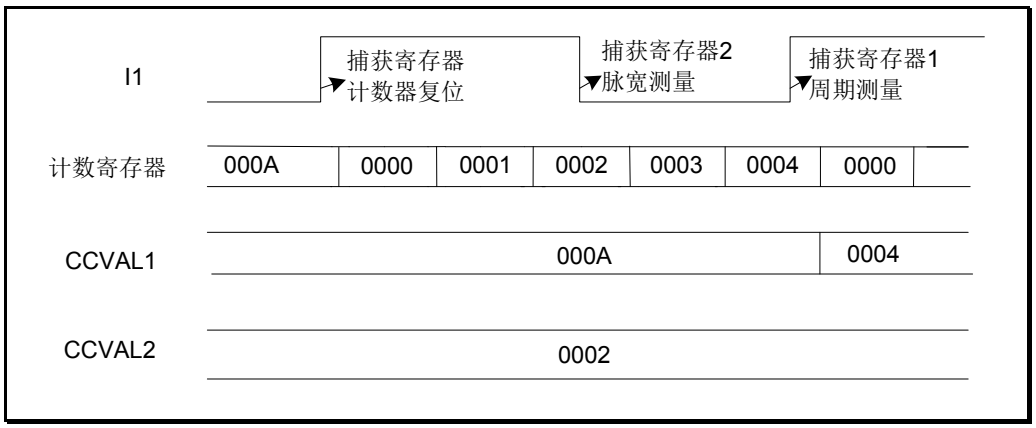


图 16-15 PWM 输入模式时序

15. 4. 7 PWM模式

脉宽调制模式可以产生一个 AD16C4Tn_AR 寄存器值确定频率，AD16C4Tn_CCVALn 寄存器值确定占空比的信号。

每个通道的 PWM 模式是相互独立的（每个 CHnO 输出一个 PWM），AD16C4Tn_CHMRn 寄存器的 CHnOMOD 位写"110"（PWM 模式 1）或写"111"（PWM 模式 2）。必须通过置位 AD16C4Tn_CHMRn 寄存器的 CHnOPEN 位来使能相应的预载寄存器，最后还需置位 AD16C4Tn_CON1 寄存器的 ARPEN 位来使能自动重装预载功能。

只有当更新事件发生时预载寄存器中的值才会传到影子寄存器，因此，在使能计数前，必须通过置位 AD16C4Tn_SGE 寄存器的 SGU 位来初始化所有的寄存器。

CHnO 的极性可通过 AD16C4Tn_CCEP 寄存器的 CCnPOL 位配置，有效极性可配置为高或低。CHnO 的输出使能由 CCnEN、CCnNE、GOEN、OFFSSI 和 OFFSSR 位

（AD16C4Tn_CCEP 和 AD16C4Tn_BDCFG 寄存器）组合控制。

在 PWM 模式（1 或 2）中，AD16C4Tn_COUNT 和 AD16C4Tn_CCVALn 寄存器的值会持续的比较，确定 AD16C4Tn_CCVALn<= AD16C4Tn_COUNT 或 AD16C4Tn_CCVALn>= AD16C4Tn_COUNT（取决于计数器的计数方向）。

定时器产生 PWM 波形是边沿对齐或中心对齐，取决于 AD16C4Tn_CON1 寄存器的 CMSEL 位。

15. 4. 7. 1 PWM边沿对齐模式

1. 递增计数配置

当 AD16C4Tn_CON1 寄存器的 DIRSEL 位为低时，计数器递增计数。

下图给出了 AD16C4Tn_AR = 8 时的边沿对齐 PWM 波形。

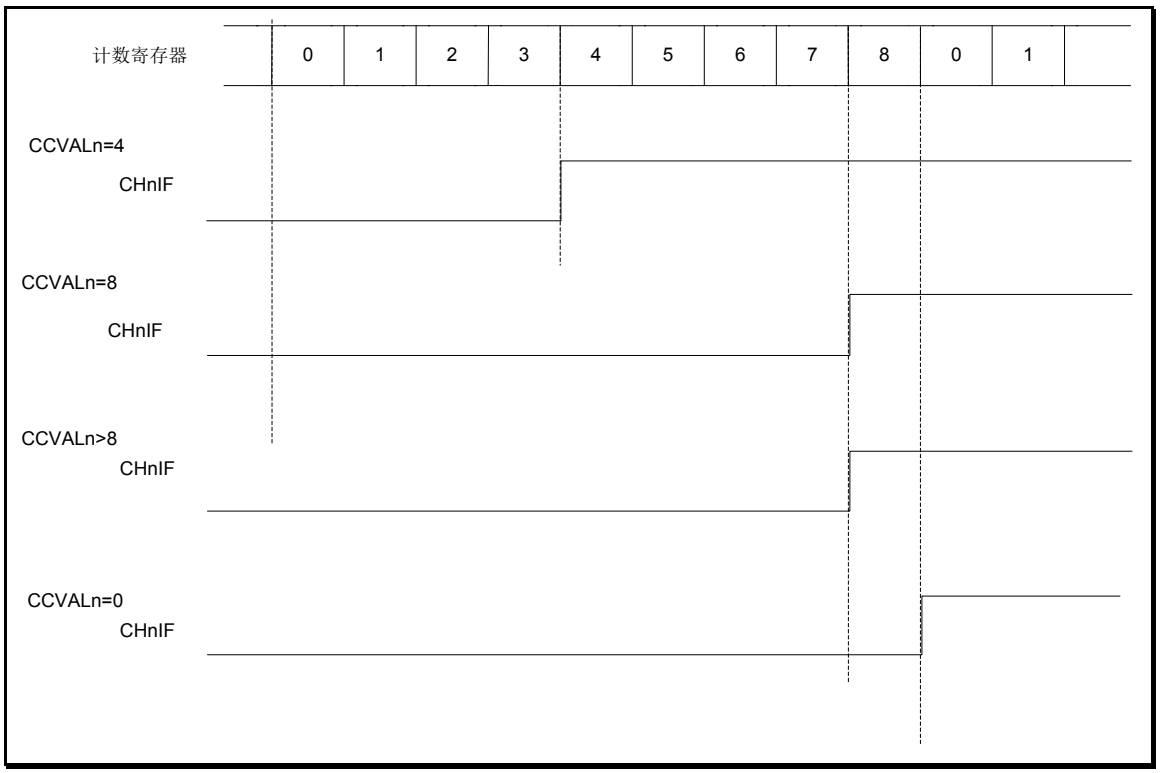


图 16-16 边沿对齐 PWM 波形（AR=8）

2. 递减计数配置

当 AD16C4Tn_CON1 寄存器的 DIRSEL 位为高时，计数器递减计数。

15. 4. 7. 2 PWM中心对齐模式

当 AD16C4Tn_CON1 寄存器中的 CMSEL 位不为"00"时，中心对齐模式有效。计数器是递增、递减计数分别置比较标志位或递增递减都置比较标志位，取决于 CMSEL 位的配置。AD16C4Tn_CON1 寄存器的方向位（DIRSEL）是由硬件更新的，软件无法修改。

下图为中心对齐方式产生的 PWM 波形的例子：

◇ AD16C4Tn_AR=8

◇ PWM 模式 1

- AD16C4Tn_CON1 寄存器的 CMSEL= "01"，在中心对齐模式 1 下，计数器向下计数时会置位比较标志位。

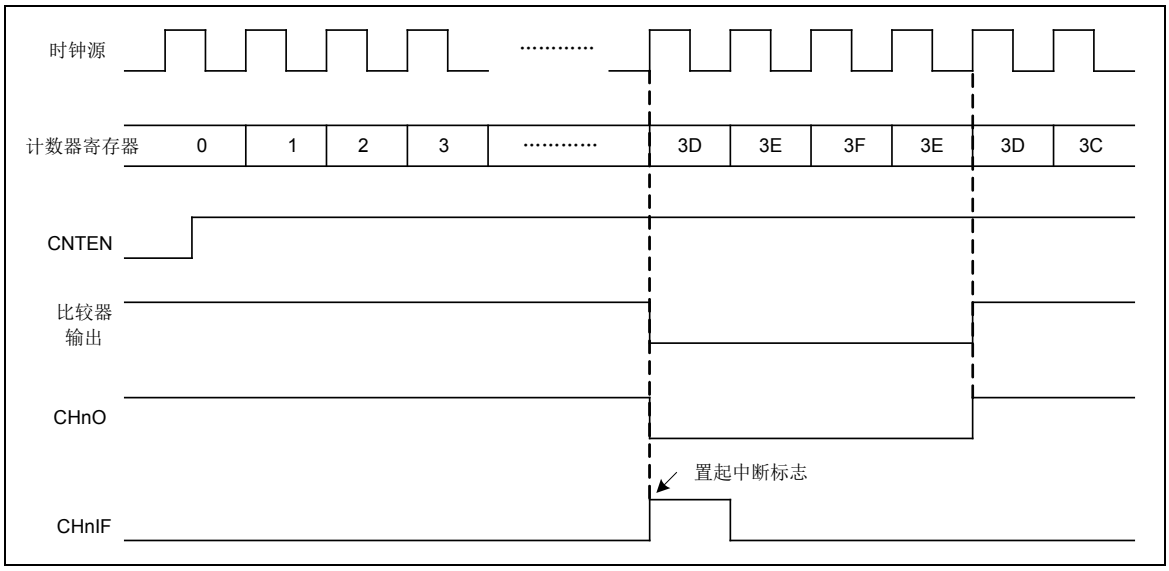


图 16-17 中心对齐 PWM 波形（AR=8）

中心对齐模式的使用技巧：

◇ 当进入中心对齐模式后，当前递增或递减配置生效。计数器递增或递减计数取决于 AD16C4Tn_CON1 寄存器的 DIRSEL 位的值。此外，软件无法对 DIRSEL 和 CMSEL 位同时进行修改。

◇ **计数器在中心对齐模式下运行时，对计数器写操作可能导致不可预知的结果**。特别是：

- 若向计数器写入的值大于自动重载值（AD16C4Tn_COUNT>AD16C4Tn_AR），计数方向不更新。例如，如果计数器递增计数，写入值后仍旧递增计数。
- 若向计数器写 0 或 AD16C4Tn_AR 中的重载值，则计数方向更新，但并没有产生 UEV。

◇ 使用中心对齐模式最安全的方式是计数器开始计数前通过软件产生更新事件（置位 AD16C4Tn_SGE 寄存器中的 SGU 位）且在计数器运行过程中不对计数器写值。

15. 4. 8 输出比较模式

该功能用于控制输出波形或指示周期时间的结束。

当捕获/比较寄存器和计数器值匹配时，输出比较功能：

◇ 输出比较模式（AD16C4Tn_CHMRn 寄存器中的 CHnOMOD 位）和输出极性（AD16C4Tn_CCEP 寄存器中的 CCnPOL 位）的配置值输出到对应的引脚上。

◇ 中断状态寄存器中的标志位置位（AD16C4Tn_RIF 寄存器的 CHnIF 位）。

◇ 若相应的中断掩码置位，则产生中断（AD16C4Tn_IER 寄存器的 CCnIT 位）。

◇ 若相应的使能位置位（AD16C4Tn_DMAEN 寄存器的 CCnDS 位，AD16C4Tn_CON2 寄存器的 CCDMASEL 位用于 DMA 请求的选择），则发送

DMA 请求。

AD16C4Tn_CHMRn 寄存器中 CHnOPEN 位的值可决定 AD16C4Tn_CCVALn 寄存器是否带有预装载寄存器。

在输出比较模式中，更新事件 UEV 对 CHnO 的输出没有影响。计时分辨率为计数器的一次计数。输出比较模式同样可以用来输出单个脉冲（单脉冲模式）。

输出比较的配置过程：

1. 选定计数器时钟（内部，外部，预分频）。
2. AD16C4Tn_AR 与 AD16C4Tn_CCVALn 寄存器中写入预期值。
3. 若需要产生中断请求，置位 AD16C4Tn_IER 寄存器中的 CCnIT 位。
4. 选择输出模式，例如：
   - CHnOMOD = "011"，当 CNTV 与 CCRVn 匹配时，CHnO 输出翻转。
   - CHnOPEN = '0'，关闭预载寄存器。
   - CCnPOL = '0'，选择有效极性为高。
   - CCnEN = '1'，使能输出。
5. AD16C4Tn_CON1 寄存器中的 CNTEN 位置位，使能计数器。

假设预载寄存器没有使能（CHnOPEN = '0'，否则 AD16C4Tn_CCVALn 影子寄存器只有在下次更新事件发生时才更新）。通过软件方式，AD16C4Tn_CCVALn 寄存器的值可随时更新控制输出波形。

15. 4. 8. 1 外部事件清除比较输出

ETFP 输入端（AD16C4Tn_CHMRn 寄存器的 CHnOCLREN 位写'1'）上的高电平，可将给定通道的比较输出信号拉低。在下次更新事件（UEV）发生前，比较输出会一直保持为低。该功能只能应用在输出比较和 PWM 模式中，强制输出模式中不起作用。

ET 信号可以接到电流控制比较器的输出端。该例中，ET 须按如下流程配置：

1. 外部触发预分频器应该关闭：AD16C4Tn_SMCON 寄存器的 ETPSEL[1：0]位应该写"00"

2. 外部时钟源 2 关闭：AD16C4Tn_SMCON 寄存器的 ECM2EN 位写'0'
3. 外部触发极性（ETPOL）和外部触发滤波器（ETFLT）可根据用户需要配置

15. 4. 8. 2 强制输出模式

在输出模式中（AD16C4Tn_CHMRn 寄存器中 CCnSSEL = "00"），软件可强制将每个输出比较信号（CHnO/CHnON）改为有效或无效状态，这种修改独立于输出比较寄存器和计数器的比较结果。

为了将某输出比较信号（CHnO）强制为有效状态，需将相应的 AD16C4Tn_CHMRn 寄存器中 CHnOMOD 位写"101"。因此，比较输出被强制为高（高时为有效状态）且 CHnO 的值为 CCnPOL 极性位的相反值。

例如：CCnPOL= '0'（CHnO 高电平有效），则 CHnO 被强制为高电平。

对 AD16C4Tn_CHMRn 寄存器的 CHnOMOD 位写"100"，比较输出可被置低。

无论怎样，AD16C4Tn_CCVALn 影子寄存器和计数器之间的比较仍然进行，相应的标志位仍可置位。

15. 4. 9 单脉冲模式

单脉冲模式（SPMEN）下，响应某个触发后，定时器的输出通道在可配置的延迟时间后产生一个脉冲，脉冲长度可配。从模式控制器可控制计数器的启动。脉冲波形可在输出比较模式和 PWM 模式下产生。置位 AD16C4Tn_CON1 寄存器的 SPMEN 位可选择单脉冲模式。计数器会在下次更新事件 UEV 产生时自动停止。

只有比较值不同于计数器初始值时，单脉冲才可以正确的产生。计数器开始计数前（定时器等待触发），必须如下配置：

- ◇ 递增计数：CNT < CCVALn≤ AR（特别地，0 < CCVALn）
- ◇ 递减计数：CNT > CCVALn

基于 PWM 模式设置单脉冲输出波形的步骤如下：

- ◇ 设置 AD16C4Tn_CHMRn 寄存器的 CHnOMOD 位，选择 PWM 模式 1 或 2；
- ◇ 设置 AD16C4Tn_CCEPn 寄存器的 CCnPOL 位，选择通道端口 CHnO 的输出极性；
- ◇ 设置 AD16C4Tn_CON1 寄存器的 DIRSEL，CMSEL，SPMEN 位，配置为递增或递减计数，PWM 普通波形模式，单脉冲模式使能；
- ◇ 设置 AD16C4Tn_CHMR1 寄存器的 CH1OPREN =1，AD16C4Tn_CON1 寄存器的 ARPEN =1，使能比较寄存器和计数重载寄存器的缓冲功能（也可以根据实际情况不使能缓冲）；
- ◇ 设置 AD16C4Tn_CCVALn 寄存器和 AD16C4Tn_AR 寄存器，配置单脉冲输出延时和脉宽时间；
- ◇ 设置 AD16C4Tn_SGE 寄存器的 SGU 位来产生一个更新事件；
- ◇ 设置 AD16C4Tn_CON1 寄存器的 CNTEN=1 来启动计数器，也可以在触发模式下，通过外部触发输入信号来触发硬件自动设置 CNTEN=1。

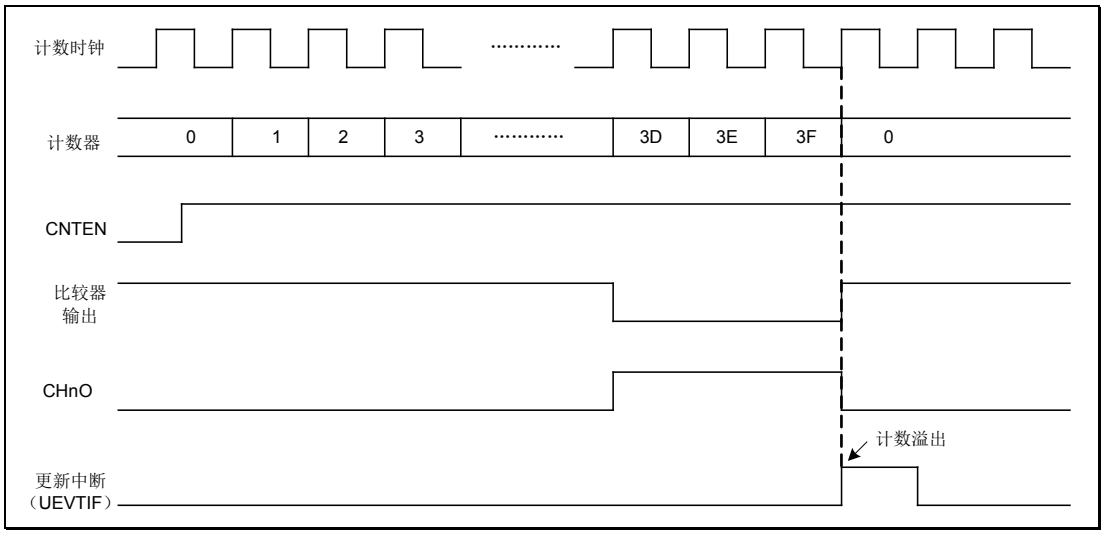


图 16-18 单脉冲模式

15. 4. 10 互补输出与死区时间

两个互补的通道输出信号，可以用来控制输出的瞬时开关。死区时间可配置。

每个输出可独立选择输出极性（主输出 CHnO 或互补输出 CHnON），该操作可通过写 AD16C4Tn_CCEP 寄存器的 CCnPOL 和 CCnNPOL 位完成。

互补信号 CHnO 和 CHnON 由几个控制位共同控制，分别是 AD16C4Tn_CCEP 寄存器中的 CCnEN 和 CCnNE 位，AD16C4Tn_BDCFG 和 AD16C4Tn_CON2 寄存器中的 GOEN、OISSn、OISSnN、OFFSSI 及 OFFSSR 位。特别是死区时间使能后的空闲状态的切换（GOEN 变为 0）。

置位 CCnEN 和 CCnNE 位，使能死区时间插入，若有刹车电路，同样需要置位 GOEN 位。AD16C4Tn_BDCFG 寄存器的 DT[7：0]可以控制所有通道的死区时间的产生。根据比较输出波形，产生 CHnO 和 CHnON 两路输出。若 CHnO 和 CHnON 有效电平为高：

- ◇ CHnO 的输出信号与参考信号一致。上升沿除外，相对参考信号的上升沿，CHnO 输出会有延迟。
- ◇ CHnON 的输出信号与参考信号相反。上升沿除外，相对参考信号的下降沿，CHnON 输出会有延迟。

若延迟时间大于有效输出的宽度（CHnO 或 CHnON），则相应的脉冲不会产生。

下图给出了死区时间输出信号和比较输出波形之间的关系。假设 CCnPOL = 0，CCnNPOL = 0，GOEN = 1，CCnEN = 1，和 CCnNE = 1

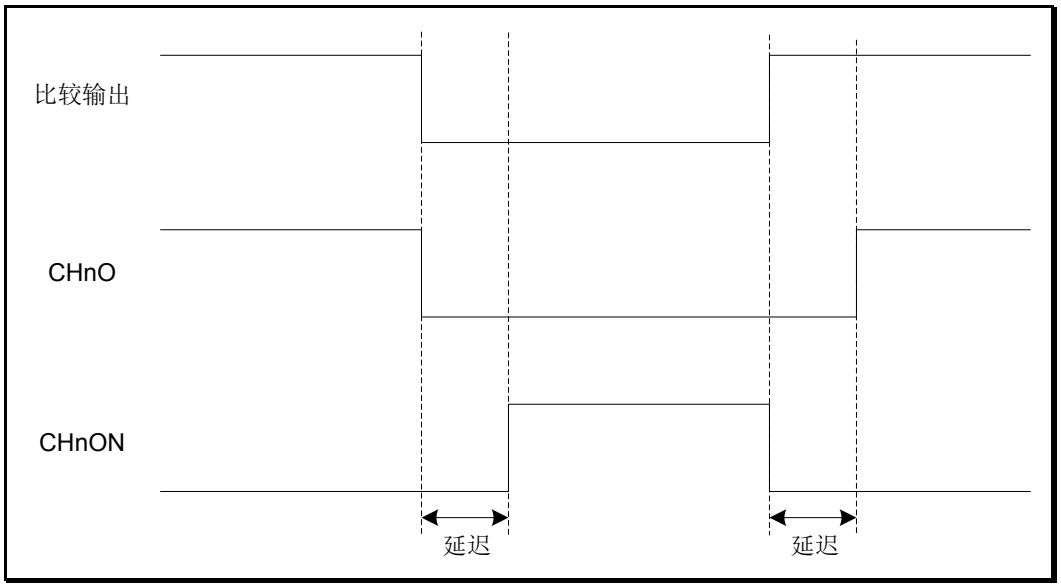


图 16-19 互补输出含死区时间插入

当 PWM 通道配置为互补输出时，如下寄存器控制位都会有缓冲：CHnOMOD、CCnE 和 CCnNE。发生互补通道更新事件时，这些寄存器位才会真正生效，这样就可以预先设置好下一步的配置，并同时对所有互补通道的配置进行更新。互补通道更新事件可以通过设置 AD16C4Tn_SGE 寄存器的 SGCOM=1 产生，或由触发信号产生（由 AD16C4Tn_SMCON 寄存器的 TSSEL 位选择触发信号）。

15. 4. 11 刹车功能

刹车功能模式由以下几个控制位进行设置：AD16C4Tn_BDCFG 寄存器中的 GOEN、OFFSSI 和 OFFSSR 位，AD16C4Tn_CON2 寄存器中的 OISSn 和 OISSnN 位，输出使能信号和无效电平都会被修改。

刹车源可以是刹车输入引脚、时钟失败事件以及软件控制 AD16C4Tn_SGE 寄存器的的 SGBRK 位。时钟失败事件由时钟控制器（CMU）中，时钟安全系统（CSS）产生。时钟安全系统（CSS）详细信息可参考时钟安全系统章节。

系统复位后，刹车电路被禁止且 GOEN 位被复位。置位 AD16C4Tn_BDCFG 寄存器的 BRKEN 位可使能刹车功能，同样寄存器中，BRKEN 位可选择刹车输入信号的极性。BRKEN 和 BRKP 位可同时修改。对 BRKEN 和 BRKP 位写操作后，1 个 APB 时钟周期延时后写入值才会生效。因此，写操作后，需等待 1 个 APB 时钟周期后才能正确读回写入值。

由于 GOEN 的下降沿可以是异步的，在实际信号（作用在输出端）和同步控制位（AD16C4Tn_BDCFG 寄存器中）之间插入了一个同步电路。这也导致了异步和同步信号之间会产生一些延迟。特别是 GOEN 之前为低时对 GOEN 写 1 操作后，要读取正确值，必须先插入一个延时（空指令）。这是因为写入的是异步信号，而读取的是同步信号。

当发生刹车请求时（刹车输入端有刹车电平）：

- ◇ GOEN 位被异步清除，输出端进入无效状态，空闲状态或复位状态（OFFSSI 位选择）。即使 MCU 的振荡器关闭，该功能仍然有效。
- ◇ 一旦 GOEN=0，每个通道输出预先配置的电平。AD16C4Tn_CON2 寄存器中的 OISSn 位配置该电平。如果 OFFSSI=0，则定时器释放使能输出，否则使能输出一直为高。
- ◇ 当使用互补输出时：
  - 输出端首先被置于复位状态，无效状态（取决于极性）。这个过程异步的，即使定时器没有时钟也有效。
  - 如果定时器时钟仍然存在，则死区时间生成器会重新生效，这样在死区时间后，OISSn 和 OISSnN 位的配置电平可驱动输出。这种情况下，CHnO 和 CHnON 无法驱动输出端都为有效电平。
  - 输出首先被置于复位状态即无效的状态（取决于极性）。这是异步操作，即使定时器没有时钟时，此功能也有效。注，由于对 GOEN 的重新同步，死区时间的周期会比通常情况下长一些（大约 2 个 TIMER 模块时钟周期）。
  - 如果 OFFSSI = 0，则定时器释放使能输出，否则使能输出保持或变高（一旦 CCnEN 和 CCnNE 有一个变高时）。
- ◇ 当刹车状态标志位（AD16C4Tn_RIF 寄存器中的 BRKIF 位）置位时，若 AD16C4Tn_IER 寄存器中的 BRKIT 位置位，可触发中断。
- ◇ 当 AD16C4Tn_BDCFG 寄存器中的 AOEN 位置位时，在下次更新事件（UEV）发生时，GOEN 位会自动置位。例如，该功能可用来整形。否则，GOEN 位会保持为低，直到对其写'1'操作，该特性可用于安全方面的应用，可以将刹车输入端接到一个电源驱动的报警端、热敏传感器或其他安全器件上。

注：刹车输入为电平有效。所以，当刹车输入有效时，不能置位（自动地或者通过软件）GOEN。同时，状态标志 BRKIF 不能被清除。

除刹车输入和输出管理，为保证应用程序的安全，内部刹车电路具有写保护功能。用户可冻结几个配置参数（死区时间，CHnO/CHnON 极性和失能时状态，CHnOMOD 配置，刹车使能和极性）。通过 AD16C4Tn_BDCFG 寄存器中的 LOCKLVL 位，可从三个保护等级中选择一种保护等级。MCU 复位后，LOCKLVL 位只能写一次。

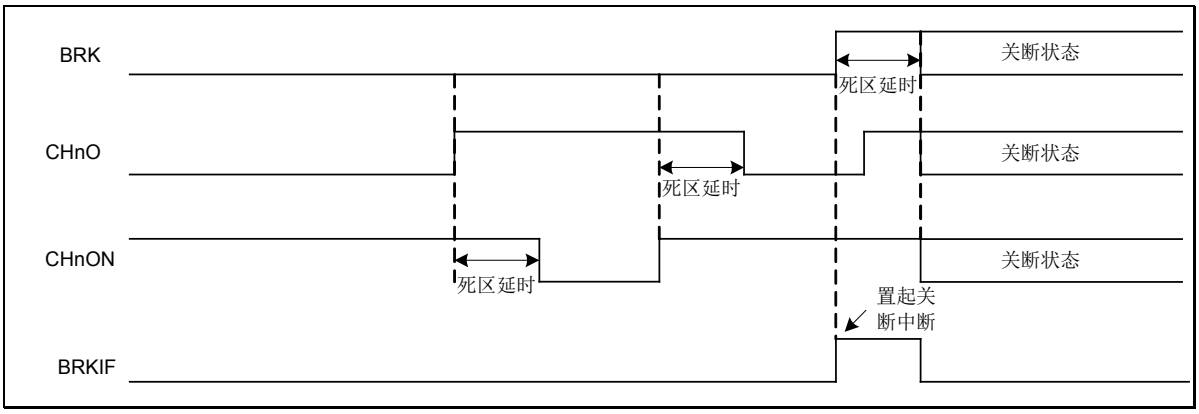


图 16-20　刹车输出行为

15. 4. 12 编码器接口模式

编码器接口模式的三种配置：若计数器只根据 I2 上的边沿计数，则 AD16C4Tn_SMCON 寄存器中的 SMODS = "001"；若计数器只根据 I1 上的边沿计数，则 AD16C4Tn_SMCON 寄存器中的 SMODS = "010"；若计数器同时根据 I1 和 I2 上的边沿计数，则 AD16C4Tn_SMCON 寄存器中的 SMODS = "011"。

配置 AD16C4Tn_CCEP 寄存器中的 CC1POL 和 CC2POL 位的值可选择 I1 和 I2 的极性。如果需要，也可以配置输入滤波器。

CH1_IN 和 CH2_IN 端口作为增量编码器的接口。当计数器使能时，计数器根据 I1 或 I2 上滤波后的有效电平变化时钟计数。I1 和 I2 滤波后的有效信号顺序会产生计数脉冲及方向信号。计数器是递增或递减计数由信号的跳变顺序决定，AD16C4Tn_CON1 寄存器中的 DIRSEL 计数方向位由自动硬件更新。

编码器接口模式的工作方式类似于一个带有方向选择的外部时钟。计数器在 0 到 AD16C4Tn_AR 寄存器中的自动重载值之间连续计数。因此，必须在开始计数前配置 AD16C4Tn_AR 寄存器。同样的，捕获器、预分频器、重复计数器、触发输出的特性正常工作。设定编码模式和选择外部时钟源 2 不兼容，不可以同时选择。

该模式下，计数器会根据增量式编码器的速度和方向自动修改，计数器的值反应的是编码器的位置。计数方向对应着连接传感器的旋转方向。

下表列出了所有的可能组合，假设 I1 和 I2 不同时变换。

有效边沿	有效边沿相对信号的电平（I1 滤波信号对应 I2,I2 滤波信号对应 I1）	I1 滤波信号		I2 滤波信号	
		上升	下降	上升	下降
仅在 I1 计数	高	下降	上升	不计数	不计数
	低	上升	下降	不计数	不计数
仅在 I2 计数	高	不计数	不计数	上升	下降
	低	不计数	不计数	下降	上升
在 I1 和 I2 上计数	高	下降	上升	上升	下降
	低	上升	下降	下降	上升

表 16-1 计数方向与编码器信号的关系

外部增量编码器可直接与 MCU 连接，无需外部逻辑接口逻辑。而比较器通常用于将编码器的差分输出转换为数字信号，这极大地增加了抗噪声能力。编码器的第三个输出端用于指示机械零点，可以连接到外部中断输入引脚以触发一次计数复位。

下图给出了计数器操作的例子，给出了计数信号了产生和方向控制。同样给出了选择双边沿时，输入抖动如何被补偿。输入抖动可能发生在传感器靠近切换点处。

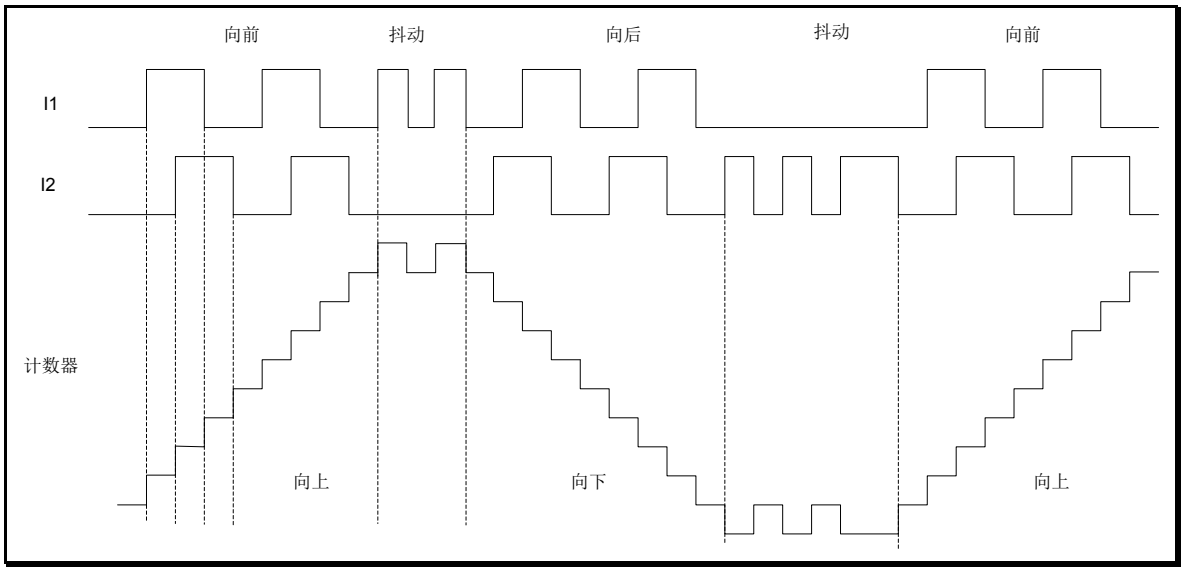


图 16-21 编码器接口模式下的计数操作

下图给出了计数器在 I1 滤波信号极性反相时的计数过程（除了 CC1POL = '1'，其他配置与上面一致）

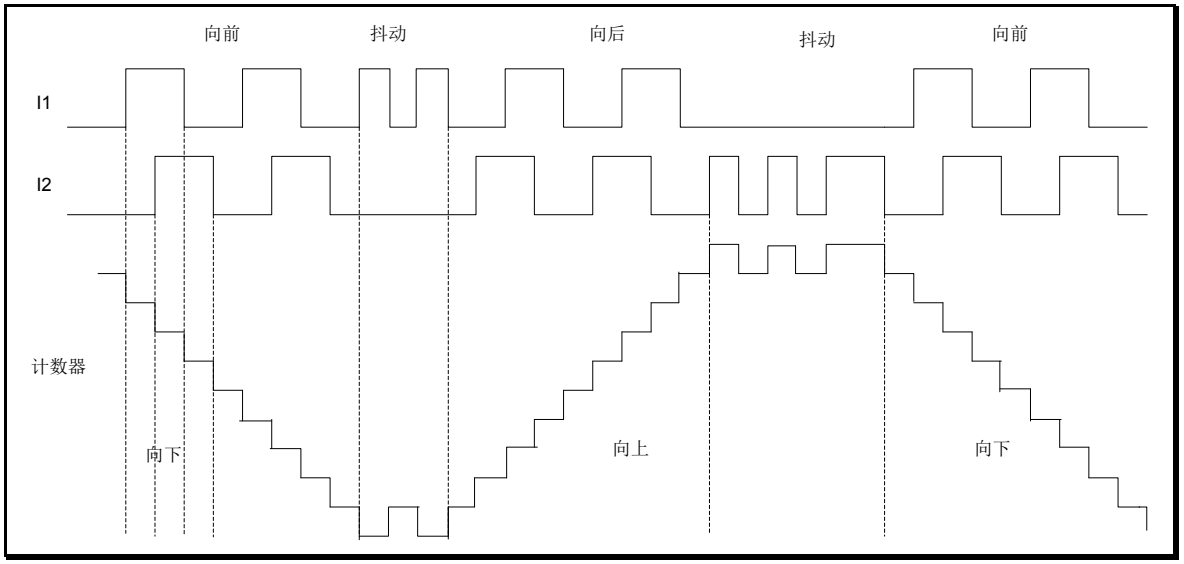


图 16-22 I1 滤波后极性反相时编码器接口例子

当配置为编码器接口模式时，定时器可提供传感器的当前位置信息。配置一个额外定时器为捕获模式，用于测量两个编码器事件的间隔，根据间隔时常获取动态信息（速度、加速度、减速度）。编码器用于指示机械零点的输出就是此用处。根据编码器两个事件间隔，可以周期性的读取计数器的值。如果允许，可以将计数器值锁存到第三个输入捕获寄存器（捕获信号必须是周期性的且可由其它定时器产生）。条件允许时，可通过实时时钟产生 DMA 请求的方式读取计数器值。

15. 4. 13 输入异或功能

通过 AD16C4Tn_CON2 寄存器中 I1FSEL 位，可将通道 1 的输入滤波器连接到 XOR 门的输出端，XOR 门联合了 CH1_IN、CH2_IN 和 CH3_IN 三个输入引脚。

XOR 输出用于定时器的所有输入功能，如触发或输入捕获。该功能参见下节的霍尔传感器接口。

15. 4. 14 霍尔传感器接口

高级控制定时器产生 PWM 信号驱动电机，另外一个定时器作为“接口定时器”。“接口定时器”捕获 3 个引脚输入（CH1_IN，CH2_IN，CH3_IN），引脚信号经过一个异或门后连接到 I1 的输入通道（置位 AD16C4Tn_CON2 寄存器中的 I1FSEL 位选择）。

从模式控制器配置为复位模式，3 个输入后的异或信号作为从输入。因此，只要 3 个输入信号有一个变化，计数器重新从 0 开始计数。霍尔输入端的任何变化都可触发创建一个时钟基准。

15. 4. 15 外部触发的同步

AD16C4Tn 定时器可在多种模式下与外部触发同步：复位模式、门控模式及触发模式。

15. 4. 15. 1 复位模式

计数器及其预分频器可以在响应触发输入事件时重新初始化。此外，若 AD16C4Tn_CON1 寄存器的 UERSEL 位为低时会产生一次更新事件 UEV。所有预载寄存器（AD16C4Tn_AR，AD16C4Tn_CCVALn）都会因更新事件 UEV 而被更新。

在下面例子中，I1 输入端的上升沿让递增计数被清空：

◇ 配置通道 1 上检测 I1 上的上升沿。配置输入滤波周期（本例无需滤波器，故 I1FLT = "0000"）。触发捕获分频器没有使用，无需配置。CC1SSEL 位只选择输入捕获源，AD16C4Tn_CHMR1 寄存器中 CC1SSEL = "01"。AD16C4Tn_CCEP 寄存器中 CC1POL = 0 以确定极性（只检测上升沿）。

◇ 定时器配置位复位模式：AD16C4Tn_SMCON 寄存器中 SMODS = "100"。选择 I1 作为输入源：AD16C4Tn_SMCON 寄存器中 TSSEL = "101"。

◇ 启动计数器：AD16C4Tn_CON1 寄存器中 CNTEN = '1'。

计数器依据内部时钟开始计数，正常计数直到 I1 上出现上升沿。当 I1 上出现上升沿时，计数器会被清零且从 0 重新开始计数。同时，标志位置位（AD16C4Tn_RIF 寄存器中 TRGIF 位），如果中断及 DMA 使能（取决于 AD16C4Tn_IER 寄存器中的 TRGIT 和 AD16C4Tn_DMAEN 寄存器的 TDS 位），会发送中断及 DMA 请求。

下图给出了当自动重载寄存器 AD16C4Tn_AR = 0x36 时的信号变化。由于 I1 输入的再同步电路，I1 上的上升沿和计数器实际复位之间会存在延时。

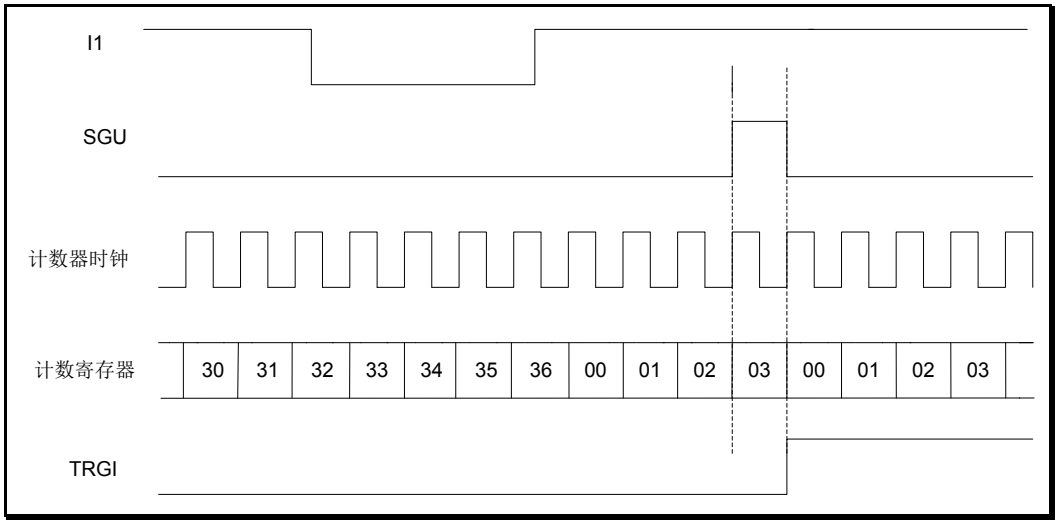


图 16-23 复位模式控制电路

15. 4. 15. 2 门控模式

计数器根据选中的输入电平被使能。

下面的例子中，计数器只在 I1 输入为低电平时才递增计数：

◇ 配置通道 1 在 I1 上检测低电平。配置输入滤波周期（本例不需要滤波器，I1FLT = "0000"）。触发捕获分频器没有使用，无需配置。AD16C4Tn_CHMR1 寄存器中的 CC1SSEL = "01"，选择输入捕获源。AD16C4Tn_CCEP 寄存器中 CC1POL = '1'，确认极性（只检测低电平）。

◇ 配置定时器为门控模式：AD16C4Tn_SMCON 寄存器中 SMODS = "101"。选择 I1 作为输入源：AD16C4Tn_SMCON 寄存器中 TSSEL = "101"。

◇ 使能计数器：AD16C4Tn_CON1 寄存器中 CNTEN = '1'（门控模式中，如果 CNTEN = '0'，无论触发输入为何电平，计数器都不会启动）。

只要 I1 为低电平，计数器依据内部时钟开始计数，一旦 I1 为高则停止计数。由于 I1 输入端再同步电路的原因，I1 上出现上升沿和计数器实际停止之间会有一定的延时。

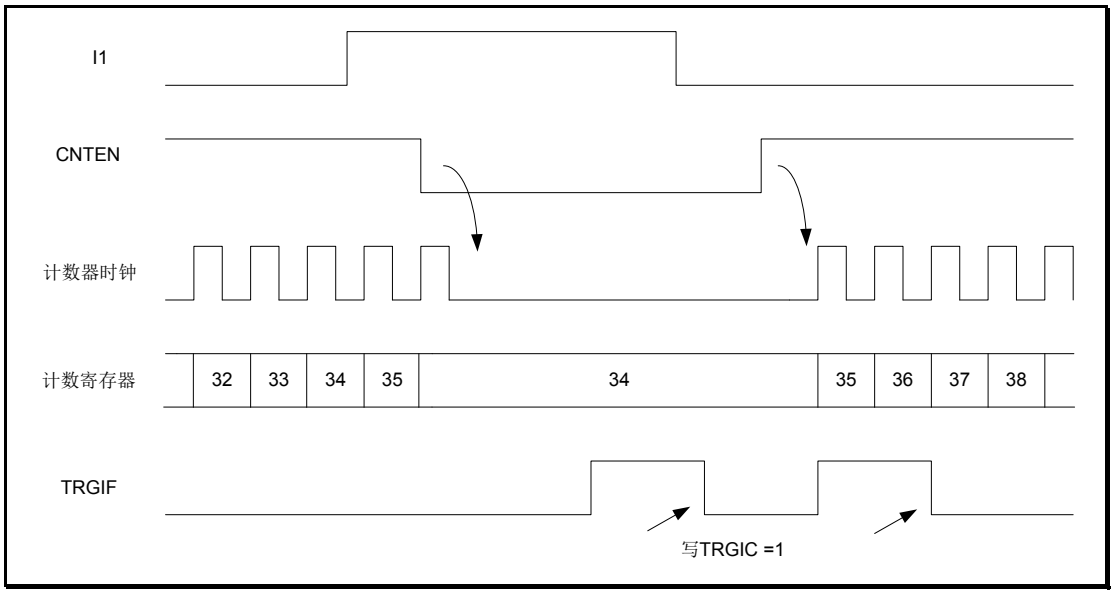


图 16-24 门控模式控制电路

15. 4. 15. 3 触发模式

输入端选中的事件可以使能计数器。

下面的例子中，I2 输入端上的上升沿可以启动递增计数：

◇ 配置通道 2 可以检测 I2 上的上升沿。配置滤波时间（本例不需要滤波，I2FLT = "0000"）。触发捕获分频器没有使用，无需配置。AD16C4Tn_CHMR1 寄存器中 CC2SEL = "01",用于选择捕获源。AD16C4Tn_CCEP 寄存器中 CC2POL = '1'，确认极性（只检测低电平）。

◇ 配置定时器为触发模式：AD16C4Tn_SMCON 寄存器中 SMODS = "110"。AD16C4Tn_SMCON 寄存器中 TSSEL = "110"，用于选择输入源。

I2 上出现上升沿时，计数器开始依据内部时钟计数并置位 TRGIF 标志位。

由于 I2 输入的再同步原因，I2 上出现上升沿和计数器实际停止之间会有一定的延时。

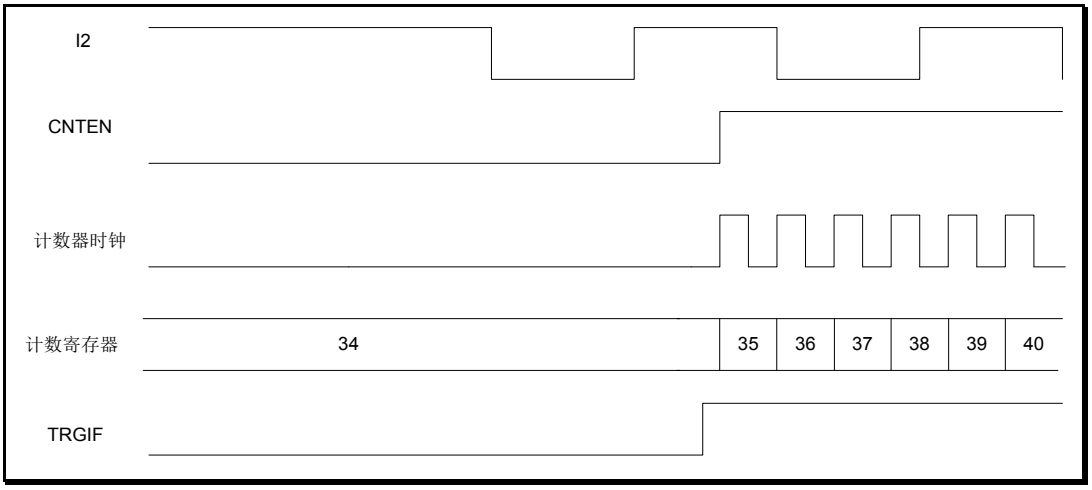


图 16-25 触发模式控制电路

15. 4. 15. 4　选择外部时钟源 2 的触发模式

外部时钟源 2 可和其他模式一起使用（除编码模式）。ET 信号可作为外部时钟输入，另一个输入可选择为触发输入（复位模式、门控模式或触发模式）。不推荐用 AD16C4Tn_SMCON 寄存器的 TSSEL 位选择 ET 作为 TI。

下面的例子中，一旦 I1 上出现上升沿时，计数器会依据 ET 信号的每个上升沿递增计数。

◇　通过 AD16C4Tn_SMCON 寄存器，配置外部触发输入电路，过程如下：

ETFLT = "000"：无滤波

ETPSEL = "00"：禁止分频

ETPOL = '0'：检测 ET 的上升沿，ECM2EN = '1'使能外部时钟模式 2

◇　配置通道 1 检测 I 的上升沿，过程如下：

I1FLT = "0000"：无滤波。

触发捕获分频器没有使用，无需配置。

AD16C4Tn_CHMR1 寄存器中 CC1SSEL = "01"选择输入捕获源，
AD16C4Tn_CCEP 寄存器的 CC1POL = '0'确认极性（只检测上升沿）。

◇　配置定时器为触发模式：AD16C4Tn_SMCON 寄存器中 SMODS = "110"。
AD16C4Tn_SMCON 寄存器中 TSSEL = "101"选择 I1 作为输入源。

I1 上出现上升沿时，计数器使能且 TRGIF 标志位置位，然后计数器根据 ET 上的上升沿开始计数。

由于 ETFP 输入再同步电路的原因，ET 信号的上升沿和实际计数器的复位会有延时。

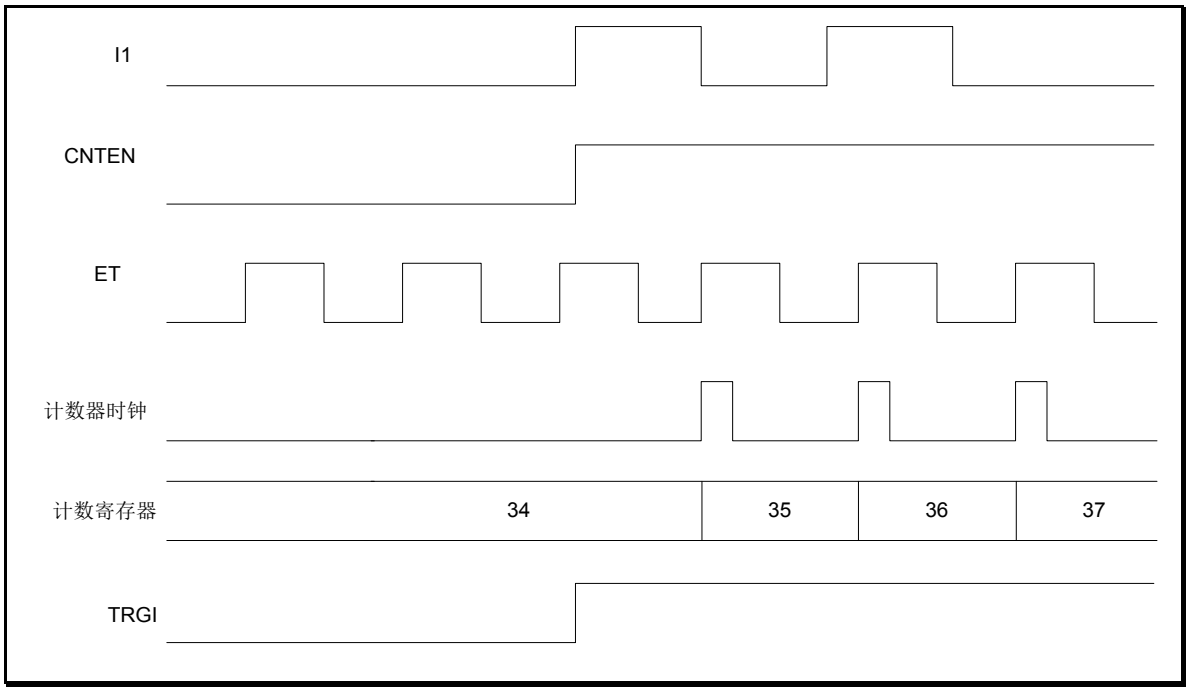


图 16-26　外部时钟源 2＋触发模式下的控制电路

15. 4. 16 6 步PWM生成

当在一个通道上需要互补输出时，预装载位有 CHxOMOD、CCxEN 和 CCxNEN。在发生 COM 换相事件时，这些预装载位被传送到影子寄存器位。这样可以预先设置好下一步骤配置，并在同一个时刻同时修改所有通道的配置。COM 可以通过设置 AD16C4T_SGE 寄存器的 SGCOM 位由软件产生，或在 TRGI 上升沿由硬件产生。

当发生 COM 事件时会设置一个标志位(AD16C4T_RIF 寄存器中的 COMI 位)，这时如果已设置了 AD16C4T_IER 寄存器的 COMI 位，则产生一个中断；如果已设置了 AD16C4T_DMAEN 寄存器的 COMDS 位，则产生一个 DMA 请求。

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

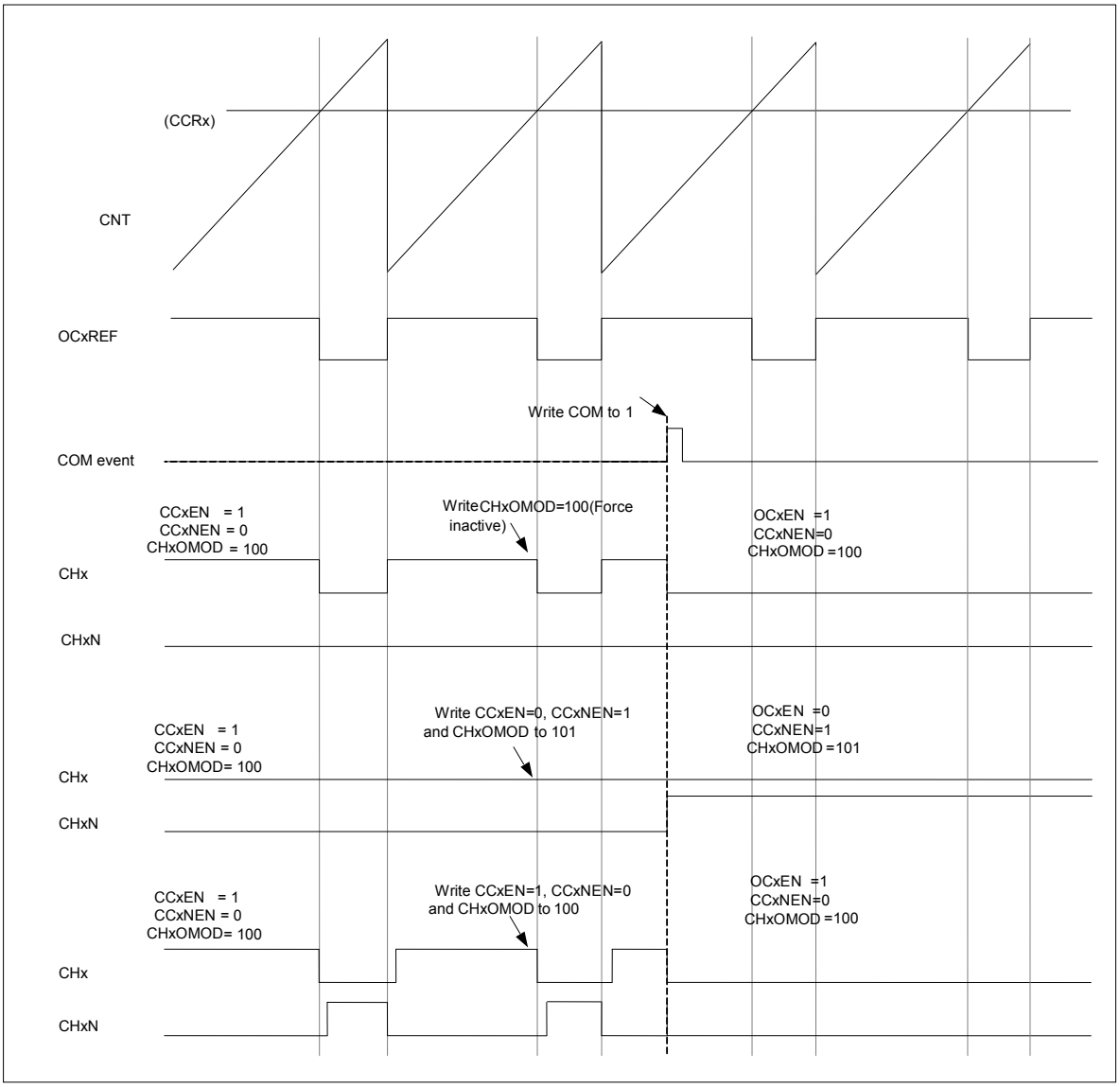


图 16-27　6 步 PWM 波形示例

15. 4. 17 调试模式

当微控制器进入调试模式（CPU 内核终止），AD16C4Tn 计数器可停止计数。

15. 5 特殊功能寄存器

15. 5. 1 寄存器列表

AD16C4Tn 寄存器列表		
寄存器名称	偏移地址	寄存器描述
AD16C4Tn_CON1	000_H	控制寄存器 1
AD16C4Tn_CON2	004_H	控制寄存器 2
AD16C4Tn_SMCON	008_H	从模式控制寄存器
AD16C4Tn_IER	$00C_H$	中断使能寄存器
AD16C4Tn_IDR	010_H	中断禁止寄存器
AD16C4Tn_IVS	014_H	中断有效状态寄存器
AD16C4Tn_RIF	018_H	原始中断标志寄存器
AD16C4Tn_IFM	$01C_H$	中断标志屏蔽寄存器
AD16C4Tn_ICR	020_H	中断清零寄存器
AD16C4Tn_SGE	024_H	事件生成寄存器
AD16C4Tn_CHMR1	028_H	通道捕获/比较模式寄存器 1
AD16C4Tn_CHMR2	$02C_H$	通道捕获/比较模式寄存器 2
AD16C4Tn_CCEP	030_H	捕获/比较使能极性寄存器
AD16C4Tn_COUNT	034_H	计数器
AD16C4Tn_PRES	038_H	时钟预分频器
AD16C4Tn_AR	$03C_H$	计数器自动重载寄存器
AD16C4Tn_REPAR	040_H	重复计数寄存器
AD16C4Tn_CCVAL1	044_H	通道捕获/比较寄存器 1
AD16C4Tn_CCVAL2	048_H	通道捕获/比较寄存器 2
AD16C4Tn_CCVAL3	$04C_H$	通道捕获/比较寄存器 3
AD16C4Tn_CCVAL4	050_H	通道捕获/比较寄存器 4
AD16C4Tn_BDCFG	054_H	刹车和死区配置寄存器
AD16C4Tn_DMAEN	058_H	DMA 使能寄存器

15. 5. 2 寄存器描述

15. 5. 2. 1 控制寄存器 1（AD16C4Tn_CON1）

控制寄存器 1（AD16C4Tn_CON1）
偏移地址：000_H
复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																DBGSEL	Reserved	OCCISS			OCCISP	DFCKSEL		APREN	CMSEL		DIRSEL	SPMEN	UERSEL	DISUE	CNTEN

Reserved	Bit 31-16	-	**保留，必须保持为复位值**
DBGSEL	Bit 15	R/W	**调试时通道输出状态选择：** 0：通道输出为高阻态 1：通道输出保持
Reserved	Bit 14	-	**保留，必须保持为复位值**
OCCISS	Bit 13-11	R/W	**通道输出清除内部触发源选择：** 000：无效（实际会产生高电平触发源） 011：清除触发源通道 2（ADC 模拟看门狗） 其他：不建议使用
OCCISP	Bit 10	R/W	**通道输出清除内部触发源极性：** 0：低电平有效（OCCISS 设置无效后该位设 0） 1：高电平有效
DFCKSEL	Bit 9-8	R/W	**死区发生器和数字滤波器工作时钟频率 Fdfck 选择位** 该位为计数器时钟（INT_CLK）频率，显示了死区时间和由死区发生器与数字滤波器（ETR，In）所用的采样时钟（tDTS）之间的分频关系。 00：tDTS=tINT_CLK 01：tDTS=2*tINT_CLK 10：tDTS=4*tINT_CLK 11：预留，不允许编程该值。
ARPEN	Bit 7	R/W	**计数器自动重载寄存器预载** 0：AD16C4Tn_AR 寄存器未缓冲 1：AD16C4Tn_AR 寄存器被装入缓冲器
CMSEL	Bit 6-5	R/W	**中心对齐模式选择** 00：**边沿对齐模式**。计数器根据方向为（DIRSEL）来向上或向下计数。 01：**中央对齐模式 1**。计数器以交替方式向上或向下计数。仅当计数器向下计数时，配置为输出的通道（AD16C4Tn_CHMRn 寄存器中 CCnSSEL=00）的输出比较中断标志位才会被设

			置。 10：**中央对齐模式 2**。计数器以交替方式向上或向下计数。仅当计数器向上计数时，配置为输出的通道（AD16C4Tn_CHMRn 寄存器中 CCnSSEL=00）的输出比较中断标志位才会被设置。 11：**中央对齐模式 3**。计数器以交替方式向上或向下计数。当计数器向上或向下计数时，配置为输出的通道（AD16C4Tn_CHMRn 寄存器中 CCnSSEL=00）的输出比较中断标志位均会被设置。 注意：当计数器使能时（CNTEN=1），不允许从边沿对齐模式转换到中央对齐模式。
DIRSEL	Bit 4	R/W	**计数器方向选择** 0：计数器向上计数 1：计数器向下计数 注意：当计数器配置为中央对齐模式或者编码器模式时，该位只读。
SPMEN	Bit 3	R/W	**使能单脉冲模式** 0：当发生更新事件时，计数器不停止。 1：当发生下一次更新事件（CNTEN 位清零）时，计数器停止。
UERSEL	Bit 2	R/W	**选择更新事件请求** 该位由软件置 1 或清零，来选择 UEV 事件源。 0：如果更新中断或 DMA 请求使能，则下述任一时间都可产生更新中断或 DMA 请求： – 计数器上溢/下溢 – 设置 SGU 位 – 从模式控制器产生的更新 1：如果更新中断或 DMA 请求使能，仅计数器上溢/下溢才能产生更新中断或 DMA 请求中断
DISUE	Bit 1	R/W	**禁止更新事件** 该位由软件置 1 或清零来使能/禁止 UEV 事件的产生。 0：UEV 使能。更新事件（UEV）由下列任一事件产生： – 计数器上溢/下溢 –设置 SGU 位 –从模式控制器产生的更新 缓冲寄存器载入他们的预载值。 1：UEV 禁止。不产生更新事件，影子寄存器保持他们的值（ARRV, ICnPRES, CCRVn）。如果从从模式控制器接收到硬件复位，计数器和预分频器将被重新初始化。

CNTEN	Bit 0	R/W	**使能计数器** 0：计数器禁止 1：计数器使能 注意：如果软件设置了 CNTEN 位，外部时钟，门控模式和编码器模式才能工作。触发模式可由硬件自动设置 CNTEN 位。

15. 5. 2. 2 控制寄存器 2（AD16C4Tn_CON2）

控制寄存器 2（AD16C4Tn_CON2）
偏移地址：004_H
复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																	OISS4	OISS3N	OISS3	OISS2N	OISS2	OISS1N	OISS1	I1FSEL	TRGOSEL			CCDMASEL	CCUSEL	Reserved	CCPCEN

Reserved	Bit 31-15	-	**保留，必须保持为复位值**
OISS4	Bit 14	R/W	**通道 4 输出的空闲状态选择位** 参考 OISS1 描述
OISS3N	Bit 13	R/W	**通道 3 互补输出的空闲状态选择位** 参考 OISS1N 描述
OISS3	Bit 12	R/W	**通道 3 输出的空闲状态 3 选择位** 参考 OISS1 描述
OISS2N	Bit 11	R/W	**通道 2 互补输出的空闲状态选择位** 参考 OISS1N 描述
OISS2	Bit 10	R/W	**通道 2 输出的空闲状态选择位** 参考 OISS1 描述
OISS1N	Bit 9	R/W	**通道 1 互补输出的空闲状态选择位** 0：当 GOEN=0，在一段死区时间后，CH1ON=0 1：当 GOEN=0，在一段死区时间后，CH1ON=1 注意：当 AD16C4Tn_BDCFG 寄存器中的 LOCKLVL 位被设置为锁定级别 1,2,或 3 后，OISS1N 不可更改。
OISS1	Bit 8	R/W	通道 1 输出的空闲状态选择位 0：当 GOEN=0，如果 CH1ON 已实现，在一段死区时间后，CH1O=0 1：当 GOEN=0，如果 CH1ON 已实现，在一段死区时间后，CH1O=1 注意：当 AD16C4Tn_BDCFG 寄存器中的 LOCKLVL 位被设置为锁定级别 1,2,或 3 后，OISS1 不可更改。
I1FSEL	Bit 7	R/W	**选择 I1 引脚功能** 0：AD16C4Tn_CH1 引脚与 I1 输入连接 1：AD16C4Tn_CH1，CH2 和 CH3 引脚与 I1 输入（XOR）连接。
TRGOSEL	Bit 6-4	R/W	**选择主模式 TRGOUT 输出** 为同步（TRGOUT），该位可选择在主模式下发送至从计数器的信息。 000：**复位**–AD16C4Tn_SGE 寄存器中的 SGU 位

			被采用为触发输出（TRGOUT）。如果复位由触发输入生成（从模式控制器配置复位模式），则相较与实际复位，TRGOUT 上的信号将会延迟。 001：**使能**–计数器使能新高 CNTEN 被用作触发输出（TRGOUT）。在从计数器使能的情况下，该设置用于在同一时间启动数次或者用来控制窗口。计数器使能信号是由 CNTEN 控制位与配置为门控模式的触发输入进行 OR 操作产生的。当计数器使能信号由触发输入控制，TRGOUT 上会产生延迟，除非被选为主/从模式（参考 AD16C4Tn_SMCON 寄存器中的 MSCFG 位的描述）。 010：**更新**–更新事件被选为触发输出（TRGOUT）。举例，主计数器可被用作从计数器的预分频器。 011：**比较脉冲**–一旦捕获或者比较匹配发生，当 CH1IF 标志位被置起（即便已为高电平），触发输出会发送一个正脉冲。 100：**比较**- 通道 1 比较输出信号用作触发输出 TRGOUT 101：**比较**- 通道 2 比较输出信号用作触发输出 TRGOUT 110：**比较**- 通道 3 比较输出信号用作触发输出 TRGOUT 111：**比较**- 通道 4 比较输出信号用作触发输出 TRGOUT
CCDMASEL	Bit 3	R/W	**使能捕获/比较 DMA** 0：当 CCn 事件发生，会发出 CCn DMA 请求。 1：当发生更新时间，会发出 CCn DMA 请求。
CCUSEL	Bit 2	R/W	**选择捕获比较更新** 0：当捕获/比较控制位为预载值（CCPCEN=1），则当设置 SGCOM 位时才会被更新。 1：当捕获/比较控制位为预载值（CCPCEN=1），则当设置 SGCOM 位或者当 TI 边沿上升时，均会被更新。 注意：该位只有用作于通道时才有互补输出。
Reserved	Bit 1	-	**保留，必须保持为复位值**
CCPCEN	Bit 0	R/W	**使能捕获比较预载控制** 0：CCnEN, CCnNE 和 CHnOMOD 不为预载值。 1：CCnEN, CCnNE 和 CHnOMOD 为预载值。当写入预载值后，仅当发生通信事件（COM）时（SGCOM 位设置或者 TI 上检测到上升沿由 CCUSEL 位决定），才会被更新）。 注意：该位只有用作于通道时才有互补输出。

15. 5. 2. 3 从模式控制寄存器（AD16C4Tn_SMCON）

从模式控制寄存器（AD16C4Tn_SMCON）
偏移地址：008_H
复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																ETPOL	ECM2EN	ETPSEL		ETFLT				MSCFG	TSSEL			OCCS	SMODS		

Reserved	Bit 31-16	-	保留，必须保持为复位值
ETPOL	Bit 15	R/W	**选择外部触发信号边沿** 0：正向 ETR,高电平有效或上升沿有效。 1：反正 ETR，低电平有效或下降沿有效。
ECM2EN	Bit 14	R/W	**使能外部时钟模式 2** 该位使能外部时钟模式 2 0：禁止外部时钟模式 2 1：使能外部时钟模式 2。计数器由 ETRF 信号上的有效边沿计数。 注意： 1. 设置 ECM2EN 位与选择外部时钟模式 1 且 TI 与 ETRF 相连接（SMODS=111 和 TSSEL=111）具有相同的效果。 2. 可同时使用外部时钟模式 2 与下列从模式：复位模式，门控模式和除法模式。在这种情况下，TI 不能与 ETRF 相连接(TSSEL 不能设置为 111。 3. 如果外部时钟模式 1 和外部时钟模式 2 同时使能，外部时钟输入为 ETRF。
ETPSEL	Bit 13-12	R/W	**选择外部触发分频** 外部触发信号 ETRP 频率最多为 AD16C4TnCLK 频率的 1/4。可使能预分频器来减小 ETRP 频率。该位有效用于输入高速外部时钟的情况。 00：预分频器关闭 01：ETRP 频率 2 分频 10：ETRP 频率 4 分频 11：ETRP 频率 8 分频
ETFLT	Bit 11-8	R/W	**选择外部触发滤波采样时钟** 该位定义了 ETRP 信号的采样频率和数字滤波器的滤波长度。 数字滤波器由一个事件计数器组成，每 N 个连续事件才视为一个有效边沿。 0000：无滤波器，采样频率为 fDTS 0001：fSAMPLING = fINT_CLK, N = 2

			0010：fSAMPLING = fINT_CLK, N = 4 0011：fSAMPLING = fINT_CLK, N = 8 0100：fSAMPLING = fDTS / 2, N = 6 0101：fSAMPLING = fDTS / 2, N = 8 0110：fSAMPLING = fDTS / 4, N = 6 0111：fSAMPLING = fDTS / 4, N = 8 1000：fSAMPLING = fDTS / 8, N = 6 1001：fSAMPLING = fDTS / 8, N = 8 1010：fSAMPLING = fDTS / 16, N = 5 1011：fSAMPLING = fDTS / 16, N = 6 1100：fSAMPLING = fDTS / 16, N = 8 1101：fSAMPLING = fDTS / 32, N = 5 1110：fSAMPLING = fDTS / 32, N = 6 1111：fSAMPLING = fDTS / 32, N = 8 注意：当 ETFLT[3：0] = 1, 2 or 3 时，公式中的 fDTS 由 INT_CLK 取代。
MSCFG	Bit 7	R/W	**主/从模式配置** 0：无动作 1：延迟触发输入（In）上的事件来允许当前计时器和其从器件之间的同步。该设置有效用于使用单个外部事件来同步多个计时器。
TSSEL	Bit 6-4	R/W	**选择从模式触发源** 该位用来选择不同的触发输入来同步计数器。 000：内部触发 0（IT0） 001：内部触发 1（IT1） 010：内部触发 2（IT2） 011：内部触发 3（IT3） 100：I1 双边沿边沿检出（I1F_ED） 101：I1 滤波输入 1 110：I2 滤波输入 2 111：外部触发输入 注意：为了避免错误边沿检测，该位在不使用时（SMODS=000）才能改变。
OCCS	Bit 3	R/W	**输出通道清除源选择** 0：OCCISS 选择的内部通道 1：ETFP
SMODS	Bit 2-0	R/W	**选择从模式功能** 当选择外部信号，触发信号 TI 的有效边沿与外部输入的极性有关系（详见输入控制寄存器和控制寄存器描述） 000：**禁止从模式**–如果 CNTEN = ‘1’，则预分频器直接由内部时钟计数。 001 **编码器模式 1**–计数器向上/向下计数 I2 边沿，取决于 I1 电平 。

			010：**编码器模式 2** -计数器向上/向下计数 I1 边沿检出边沿，取决于 I2 边沿检出电平 011：**编码器模式 3** -计数器向上/向下计数 I1 边沿检出和 I2 边沿检出边沿，取决于另一个输入的电平。 100：**复位模式–**选中的触发输入的上升沿重新初始化计数器，生成寄存器的更新 101：**门控模式–**当触发输入 TI 为高电平，计数器时钟使能。一旦触发变为低电平，计数器停止计数（并非复位）。计数器的启动和停止均受控制。 110：**触发模式–**计数器在触发信号 TI 的上升沿处启动（不复位）。仅寄存器的启动受控制。 111：**外部时钟源 1–**计数器在 TI 的上升沿计数 注意：如果 I1 双边沿检出被选为触发输入（TSSEL='100'），不能使用门控模式。I1 每一次转换，I1 双边沿检出就会输出 1 个脉冲，而门控模式则是检查触发信号的电平。 注意：在发生来自主计时器的接收事件之前，从计时器的时钟必须先使能，且在接收来自主计时器的触发过程中，从计数器时钟不能即时更改。

15. 5. 2. 4 中断使能寄存器（AD16C4Tn_IER）

中断配置寄存器（AD16C4Tn_IER）
偏移地址：$00C_H$
复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																								BRKIT	TRGIT	COMIT	CC4IT	CC3IT	CC2IT	CC1IT	UIT

Reserved	Bit31-8	-	保留，必须保持复位值。
BRKIT	Bit7	W	**使能刹车中断** 0：无效 1：使能
TRGIT	Bit6	W	**使能触发中断** 0：无效 1：使能
COMIT	Bit5	W	**使能 COM 中断** 0：无效 1：使能
CC4IT	Bit4	W	**使能捕获/比较 4 中断** 0：无效 1：使能
CC3IT	Bit3	W	**使能捕获/比较 3 中断** 0：无效 1：使能
CC2IT	Bit2	W	**使能捕获/比较 2 中断** 0：无效 1：使能
CC1IT	Bit1	W	**使能捕获/比较 1 中断** 0：无效 1：使能
UIT	Bit0	W	**使能更新事件中断** 0：无效 1：使能

15. 5. 2. 5 中断禁止寄存器（AD16C4Tn_IDR）

中断禁止寄存器（AD16C4Tn_IDR）
偏移地址：0010_H
复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																								BRKI	TRGI	COMI	CC4I	CC3I	CC2I	CC1I	UI

Reserved	Bit 31-8	-	保留，必须保持复位值。
BRKI	Bit7	W	**禁止刹车中断** 0：无效 1：禁止
TRGI	Bit 6	W	**禁止触发中断** 0：无效 1：禁止
COMI	Bit 5	W	**禁止 COM 中断** 0：无效 1：禁止
CC4I	Bit 4	W	**禁止捕获/比较 4 中断** 0：无效 1：禁止
CC3I	Bit 3	W	**禁止捕获/比较 3 中断** 0：无效 1：禁止
CC2I	Bit 2	W	**禁止捕获/比较 2 中断** 0：无效 1：禁止
CC1I	Bit 1	W	**禁止捕获/比较 1 中断** 0：无效 1：禁止
UI	Bit 0	W	**禁止更新中断** 0：无效 1：禁止

15. 5. 2. 6 中断有效状态寄存器（AD16C4Tn_IVS）

中断有效状态寄存器（AD16C4Tn_IVS）
偏移地址：0014_H
复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																								BKI	TRGI	COMI	CC4I	CC3I	CC2I	CC1I	UEI

Reserved	Bit 31-8	N/A	保留
BKI	Bit 7	R	**刹车中断状态** 0：禁止 1：使能
TRGI	Bit 6	R	**触发中断状态** 0：禁止 1：使能
COMI	Bit 5	R	**CM中断状态** 0：禁止 1：使能
CC4I	Bit 4	R	**通道 4 捕获/比较中断状态** 0：禁止 1：使能
CC3I	Bit 3	R	**通道3捕获/比较中断状态** 0：禁止 1：使能
CC2I	Bit 2	R	**通道 2 捕获/比较中断状态** 0：禁止 1：使能
CC1I	Bit 1	R	**通道2捕获/比较中断状态** 0：禁止 1：使能
UEI	Bit 0	R	**更新事件中断状态** 0：禁止 1：使能

15. 5. 2. 7 原始中断标志寄存器（AD16C4Tn_RIF）

中断标志寄存器（AD16C4Tn_RIF）
偏移地址：0018_H
复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																			CH4OVIF	CH3OVIF	CH2OVIF	CH1OVIF	Reserved	BRKIF	TRGIF	COMIF	CH4IF	CH3IF	CH2IF	CH1IF	UEVTIF

Reserved	Bit 31-13	N/A	保留
CH4OVIF	Bit 12	R	**通道 4 捕获溢出中断标志** 仅当相应的通道配置为捕获输入状态时，该标志位才由硬件设置。对 AD16C4Tn_ICR 写 1 来清除原始中断。 0：未检测到捕获溢出 1：当 CH4IF 标志为置起时，捕获计数器值至 AD16C4Tn_CCVAL1 寄存器
CH3OVIF	Bit 11	R	**通道 3 捕获溢出中断标志** 仅当相应的通道配置为捕获输入状态时，该标志位才由硬件设置。对 AD16C4Tn_ICR 写 1 来清除原始中断。 0：未检测到捕获溢出 1：当 CC3IF 标志为置起时，捕获计数器值至 AD16C4Tn_CCVAL1 寄存器
CH2OVIF	Bit 10	R	**通道 2 捕获溢出中断标志** 仅当相应的通道配置为捕获输入状态时，该标志位才由硬件设置。对 AD16C4Tn_ICR 写 1 来清除原始中断。 0：未检测到捕获溢出 1：当 CH2IF 标志为置起时，捕获计数器值至 AD16C4Tn_CCVAL1 寄存器
CH1OVIF	Bit 9	R	**通道 1 捕获溢出中断标志** 仅当相应的通道配置为捕获输入状态时，该标志位才由硬件设置。对 AD16C4Tn_ICR 写 1 来清除原始中断。 0：未检测到捕获溢出 1：当 CH1IF 标志为置起时，捕获计数器值至 AD16C4Tn_CCVAL1 寄存器
Reserved	Bit 8	R	**保留，必须保持复位值。**
BRKIF	Bit 7	R	**刹车中断标志** 如果刹车中断使能，一旦刹车输入有效，该标志

			位由硬件置起。对 AD16C4Tn_ICR 写 1 来清除原始中断。 0：未发生刹车事件。 1：在刹车输入上检测到有效电平
TRGIF	Bit 6	R	**触发中断标志** 如果触发中断使能，当从模式控制器在门控模式以外的所有模式下使能，发生触发事件时（TI 上检测到有效边沿），该标志位被硬件置起。对 AD16C4Tn_ICR 写 1 来清除原始中断。 0：未发生触发事件 1：触发中断被挂起
COMIF	Bit 5	R	**COM 中断标志** 如果 COM 中断使能，当发生 COM 事件（当捕获/比较控制位 CCnE, CCnNE, CHnOMOD 更新后），该标志位被硬件置起。对 AD16C4Tn_ICR 写 1 来清除原始中断。 0：未发生 COM 事件 1：COM 中断被挂起
CH4IF	Bit 4	R	**通道 4 捕获/比较中断标志** 参考 CH1IF 描述
CH3IF	Bit 3	R	**通道 4 捕获/比较中断标志** 参考 CH1IF 描述
CH2IF	Bit 2	R	**通道 2 捕获/比较中断标志** 参考 CH1IF 描述
CH1IF	Bit 1	R	**通道 1 捕获/比较 1 中断标志** **如果 CC1 通道配置为输出：** 如果中断使能，除去中央对齐模式的情况（参考 AD16C4Tn_CON1 寄存器中 CMSEL 的描述），当计数值与比较值匹配，该标志位由硬件置起。对 AD16C4Tn_ICR 写 1 来清除原始中断。 0：不匹配。 1：AD16C4Tn_COUNT 计数值与 AD16C4Tn_CCVAL1 值匹配。当 AD16C4Tn_CCVAL1 寄存器值大于 AD16C4Tn_AR 值，发生计数器上溢时（递增模式和递增/递减模式）或下溢时（递减模式），CH1IF 为被置起。 **如果 CC1 通道配置为输入：** 发生捕获时，该位由硬件置起。该位可通过软件或者读取 AD16C4Tn_CCVAL1 寄存器来清零。 0：未发生输入捕获 1：计数值捕获至 AD16C4Tn_CCVAL1 寄存器（I1 上检测到与选中极性匹配的边沿）
UEVTIF	Bit 0	R	**更新事件中断标志**

			如果更新中断使能，当发生更新事件，该标志位由硬件置起。对 AD16C4Tn_ICR 写 1 来清除原始中断。 0：未发生更新。 1：更新中断被挂起。当寄存器更新时，该位被硬件置起： –当重复计数器值发生上溢或者下溢（若重复计数器=0，则更新）和当 AD16C4Tn_CON1 寄存器中 DISUE=0 –当使用 AD16C4Tn_SGE 寄存器中的 SGU 位来由软件重新初始化 CNT 时，如果 AD16C4Tn_CON1 寄存中的 UERSEL=0 和 DISUE=0 –当 CNT 由触发事件来重新初始化，如果 AD16C4Tn_CON1 寄存中的 UERSEL=0 和 DISUE=0

15. 5. 2. 8 中断标志屏蔽寄存器（AD16C4Tn_IFM）

中断标志屏蔽寄存器（AD16C4Tn_IFM）
偏移地址：$001C_H$
复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																								BRKIM	TRGI	COMI	CH4CCI	CH3CCI	CH2CCI	CH1CCI	UEI

Reserved	Bit 31-8	-	保留
BRKIM	Bit 7	R	**刹车中断标志屏蔽** 如果刹车中断使能，一旦刹车输入有效，该标志位由硬件置起。对 AD16C4Tn_ICR 写 1 来清除原始中断。 0：未发生刹车事件。 1：在刹车输入上检测到有效电平
TRGI	Bit 6	R	**屏蔽触发中断标志** 如果触发中断使能，当从模式控制器在门控模式以外的所有模式下使能，发生触发事件时（TI 上检测到有效边沿），该标志位被硬件置起。对 AD16C4Tn_ICR 写 1 来清除原始中断。 0：未发生触发事件 1：触发中断被挂起
COMI	Bit 5	R	**屏蔽 COM 中断标志** 如果 COM 中断使能，当发生 COM 事件（当捕获/比较控制位 CCnEN, CCnNE, CHnOMOD 更新后），该标志位被硬件置起。对 AD16C4Tn_ICR 写 1 来清除原始中断。 0：未发生 COM 事件 1：COM 中断被挂起
CH4CCI	Bit 4	R	**屏蔽通道 4 捕获/比较中断标志** 参考 CH1CCI 描述
CH3CCI	Bit 3	R	**屏蔽通道 3 捕获/比较中断标志**参考 CH1CCI 描述
CH2CCI	Bit 2	R	**屏蔽通道 2 捕获/比较中断标志**参考 CH1CCI 描述
CH1CCI	Bit 1	R	**屏蔽通道 1 捕获/比较中断标志如果通道 1 配置为输出：** 如果中断使能，除去中央对齐模式的情况（参考 AD16C4Tn_CON1 寄存器中 CMSEL 的描述），当计数值与比较值匹配，该标志位由硬件置起。对 AD16C4Tn_ICR 写 1 来清除原始中断。 0：不匹配。 1：AD16C4Tn_COUNT 计数值与

			AD16C4Tn_CCVAL1 值匹配。当 AD16C4Tn_CCVAL1 寄存器值大于 AD16C4Tn_AR 值，发生计数器上溢时（递增模式和递增/递减模式）或下溢时（递减模式），CH1IF 为被置起。 **如果通道配置为输入：** 发生捕获时，该位由硬件置起。该位可通过软件或者读取 AD16C4Tn_CCVAL1 寄存器来清零。 0：未发生输入捕获 1：计数值捕获至 AD16C4Tn_CCVAL1 寄存器（I1 上检测到与选中极性匹配的边沿）
UEI	Bit 0	R	**屏蔽更新事件中断标志**如果更新中断使能，当发生更新事件，该标志位由硬件置起。对 AD16C4Tn_ICR 写 1 来清除原始中断。 0：未发生更新。 1：更新中断被挂起。当寄存器更新时，该位被硬件置起： –当重复计数器值发生上溢或者下溢（若重复计数器=0，则更新）和当 AD16C4Tn_CON1 寄存器中 DISUE=0 –当使用 AD16C4Tn_SGE 寄存器中的 SGU 位来由软件重新初始化 CNT 时，如果 AD16C4Tn_CON1 寄存中的 UERSEL=0 和 DISUE=0 –当 CNT 由触发事件来重新初始化，如果 AD16C4Tn_CON1 寄存中的 UERSEL=0 和 DISUE=0

15. 5. 2. 9 中断清零寄存器（AD16C4Tn_ICR）

中断清零寄存器（AD16C4Tn_ICR）
偏移地址：0020_H
复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																								BRKIC	TRGIC	COMIC	CH4CCIC	CH3CCIC	CH2CCIC	CH1CCIC	UEIC

Reserved	Bit 31-8	N/A	保留
BRKIC	Bit 7	W1C	**清刹车中断** 0：无效 1：清零（AD16C4Tn_RIF）
TRGIC	Bit 6	W1C	**清触发中断** 0：无效 1：清零（AD16C4Tn_RIF）
COMIC	Bit 5	W1C	**清 COM 中断** 0：无效 1：清零（AD16C4Tn_RIF）
CH4CCIC	Bit 4	W1C	**清通道 4 捕获/比较中断** 参考 CC1IC 描述
CH3CCIC	Bit 3	W1C	**清通道 3 捕获/比较中断** 参考 CC1IC 描述
CH2CCIC	Bit 2	W1C	**清通道捕获/比较中断** 参考 CC1IC 描述
CH1CCIC	Bit 1	W1C	**清通道捕获/比较中断** 0：无效 1：清零（AD16C4Tn_RIF）
UEIC	Bit 0	W1C	**清更新事件中断** 0：无效 1：清零（AD16C4Tn_RIF）

15. 5. 2. 10 事件生成寄存器（AD16C4Tn_SGE）

事件生成寄存器（AD16C4Tn_SGE）
偏移地址：0024_H
复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																								SGBRK	SGTRG	SGCOM	SGCC4E	SGCC3E	SGCC2E	SGCC1E	SGU

Reserved	Bit 31-8	N/A	保留
SGBRK	Bit 7	W	**软件生成刹车事件** 该位由软件设置来生成刹车事件，可由硬件自动清零。 0：无动作 1：产生刹车事件。GOEN 清零，BRKIF 标志位置起，产生相关中断或 DMA 传输。
SGTRG	Bit 6	W	**软件生成触发事件** 该位由软件设置来生成触发事件，可由硬件自动清零 。 0：无动作 1：AD16C4Tn_RIF 寄存器中的 TRGIF 被置起，产生相关中断或 DMA 传输
SGCOM	Bit 5	W	**软件生成换相事件捕获** 该位由软件设置，由硬件自动清零。 0：无动作 1：当 CCPCEN 被置 1，则可更新 CCnEN, CCnNE 和 CHnOMOD 注意：该位只有用作于通道时才有互补输出
SGCC4E	Bit 4	W	**软件触发通道 4 捕获/比较事件** 参考 SGCC1E 描述
SGCC3E	Bit 3	W	**软件触发通道 3 捕获/比较事件** 参考 SGCC1E 描述
SGCC2E	Bit 2	W	**软件触发通道 2 捕获/比较事件** 参考 SGCC1E 描述
SGCC1E	Bit 1	W	**软件触发通道 1 捕获/比较事件** 该位由软件设置来生成事件，可由硬件自动清零。 0：无动作 1：通道 1 上产生捕获/比较事件： **如果通道 1 配置为输出：** CH1IF 标志位被置起，产生相应中断或 DMA 请求发送 。 **如果通道 1 配置为输入：**

			当前计数值捕获至 AD16C4Tn_CCVAL1 寄存器。CH1IF 标志位被置起，产生相应中断或 DMA 请求发送。CH1OVIF 标志位置起（如果 CH1IF 标志位已为高电平）。
SGU	Bit 0	W	**软件触发更新事件** 该位由软件设置，可由硬件自动清零。 0：无动作 1：重新初始化计数器，更新寄存器。注意，预分频器也会被清零（但预分频比不会受到影响）。如果使用中央对齐模式或者 DIRSEL=0（递增），则计数器将清零；否则如果 DIRSEL=1（递减），则将使用自动重载入值。

15. 5. 2. 11 通道捕获/比较模式寄存器 1（AD16C4Tn_CHMR1）

◆ 输出比较模式

通道捕获模式寄存器 1（AD16C4Tn_CHMR1）
偏移地址：0028_H
复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																CH2OCLREN	CH2OMOD			CH2OPEN	CH2OFEN	CC2SEL		CH1OCLREN	CH1OMOD			CH1OPREN	CH1OHSEN	CC1SSEL	

Reserved	Bit 31-16	-	**保留**
CH2OCLREN	Bit 15	R/W	**使能通道 2 输出比较清零** 参考 CH1OCLREN 描述
CH2OMOD	Bit 14-12	R/W	**通道 2 输出比较模式** 参考 CH1OMOD 描述
CH2OPEN	Bit 11	R/W	**使能通道 2 输出比较预载入** 参考 CH1OPREN 描述
CH2OFEN	Bit 10	R/W	**使能通道 2 输出比较高速模式** 参考 CH1OHSEN 描述
CC2SEL	Bit 9-8	R/W	**选择通道 2 输出比较** 该位定义了通道以及使用的输入的方向（输入/输出） 00：通道配置为**输出** 01：通道配置为输入，捕获源为 **I2** 10：通道配置为输入，捕获源为 **I1** 11：通道配置为输入，捕获源为 **ITn 或 I1** 的双边沿检出。 仅当内部触发输入通过 TSSEL 位（AD16C4Tn_SMCON 寄存器）选择时，该模式才能工作。 注意：当通道为关闭状态时（AD16C4Tn_CCEP 中 CC2EN = '0'），CC2SEL 为只写。
CH1OCLREN	Bit 7	R/W	**使能通道 1 输出比较清零** 0：通道 1 比较输出不会受到 ETRF 输入影响 1：当 ETRF 输入上检测到高电平时，通道 1 比较输出将被清零
CH1OMOD	Bit 6-4	R/W	**输出比较 1 模式** 该位定义了输出参考信号通道 1 比较输出的行为。通道 1 比较输出为高有效，CH1O 和 CH1ON 的有效电平由 CC1POL 和 CC1NPOL 位决定。 000 :**冻结**–输出比较寄存器 AD16C4Tn_CCVAL1

			寄存器和 AD16C4Tn_COUNT 计数器之间的比较对输出无效。 001：**发生匹配时设置通道 1 为有效电平**–当计数器 AD16C4Tn_COUNT 与捕获/比较寄存器 1AD16C4Tn_CCVAL1 发生匹配后，通道 1 比较输出信号强制为高电平。 010：**发生匹配时设置通道 1 为无效电平**。当计数器 AD16C4Tn_COUNT 与捕获/比较寄存器 1AD16C4Tn_CCVAL1 发生匹配后，通道 1 比较输出信号强制为低电平。 011：**翻转** -当 AD16C4Tn_COUNT=AD16C4Tn_CCVAL1，通道 1 比较输出发生翻转。 100：**强制为无效电平** - 通道 1 比较输出强制为低电平。 101：**强制为有效电平**- 通道 1 比较输出强制为高电平。 110：**PWM 模式 1** -在递增模式下，当 AD16C4Tn_COUNT<AD16C4Tn_CCVAL1，通道 1 为有效电平，否则，通道 1 为无效电平。在递减模式下，当 AD16C4Tn_COUNT>AD16C4Tn_CCVAL1，通道 1 为无效电平（通道 1 比较输出='0'），否则通道 1 为有效电平（通道 1 比较输出='1'）。 111：**PWM 模式 2** -在递增模式下，当 AD16C4Tn_COUNT<AD16C4Tn_CCVAL1，通道 1 为无效电平，否则，通道 1 为有效电平。在递减模式下，当 AD16C4Tn_COUNT>AD16C4Tn_CCVAL1，通道 1 为有效电平，否则通道 1 为无效电平。 注意： 1：当 AD16C4Tn_BDCFG 寄存器中的 LOCKLVL 位被设置为锁定级别 3，且 CC1SSEL=00（通道为输出模式），该位将不能更改。 2：在 PWM 模式 1 和 2 中，仅当比较结果更改或当输出比较模式从冻结模式转换成 PWM 模式，比较输出电平才会更改。 3：对于有互补输出的通道，该位设置为预载值。如果 AD16C4Tn_CON2 寄存器中的 CCPCEN 位设置为 1，则只有当 COM 事件发生时，CH1OMOD 有效位才会设置为预载值中新的值。
CH1OPREN	Bit 3	R/W	**输出比较 1 预载使能** 0：AD16C4Tn_CCVAL1 的预载寄存器禁止。AD16C4Tn_CCVAL1 在任何时候都可写，新写入

			的值将立刻生效。 1：AD16C4Tn_CCVAL1 的预载寄存器禁止。读/写操作可访问预载寄存器。每当发生一次更新事件，AD16C4Tn_CCVAL1 预载入值将会被填入有效寄存器。 注意： 1：当 AD16C4Tn_BDCFG 寄存器中的 LOCKLVL 位被设置为锁定级别 3，且 CC1SSEL=00（通道为输出模式），该位将不能更改。 2：仅在单脉冲模式下（AD16C4Tn_CON1 寄存器中的 SPMEN 设置为 1），PWM 模式可在不经过验证预载寄存器的情况下使用。其他情况下的行为不做保证。
CH1OHSEN	Bit 2	R/W	**使能通道 1 输出比较高速** 该位用来加速在通道 1 输出上的输入触发事件的效应。 0：当触发开启，通道 1 运作正常取决于计数器和 CCRV1 的值。当触发输入上发现边沿时，至少需要 5 个时钟周期来激活通道 1 输出。 1：触发输入上的有效沿类似于通道 1 输出上的比较匹配。设置 OC 为 1 用来比较电平，采样触发输入和激活通道 1 输出的延时将会减少至 3 个时钟周期。只有当通道配置为 PWM1 或 PWM2 模式，CH1OHSEN 才会起作用。
CC1SSEL	Bit 1-0	R/W	**捕获/比较 1 选择** 该位定义了通道和使用的输入的方向。 00：通道配置为**输出** 01：通道配置为输入，捕获源为 **I1** 10：通道配置为输入，捕获源为 **I2** 11：通道配置为输入，捕获源为 ITn 或 I1 的双边沿检出。 只有当内部触发输入是通过 TSSEL 位（AD16C4Tn_SMCON 寄存器）选择时，该模式才运行。 注意：当通道关闭（AD16C4Tn_CCEP 寄存器中的 CC1EN = ‘0’），CC1SSEL 为只写。

◆ 输入捕获模式

通道捕获模式寄存器 1（AD16C4Tn_CHMR1）
偏移地址：0028_H
复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																I2FLT				IC2PRES		CC2SEL		I1FLT				IC1PRES		CC1SSEL	

Reserved	Bit 31-16	-	**保留**
I2FLT	Bit 15-12	R/W	**通道 2 输入捕获滤波器** 参考 I1FLT 描述
IC2PRES	Bit 11-10	R/W	**通道 2 输入捕获预分频器** 参考 IC1PRES 描述
CC2SEL	Bit 9-8	R/W	**选择通道 2 输入捕获源** 该位定义了通道和使用的输入的方向。 00：通道配置为**输出** 01：通道配置为输入，捕获源为 **I2** 10：通道配置为输入，捕获源为 **I1** 11：通道配置为输入，捕获源为 **ITn 或 I1 的双边沿检出** 只有当内部触发输入是通过 TSSEL 位（AD16C4Tn_SMCON 寄存器）选择时，该模式才运行 注意：当通道关闭（AD16C4Tn_CCEP 寄存器中的 CC2EN = '0'），CC2SEL 为只写。
I1FLT	Bit 7-4	R/W	**I1 滤波器** 该位定义了 I1 输入的采样频率和数字滤波器的长度。 数字滤波器由一个事件计数器组成，每 N 个连续事件才视为一个有效边沿： 0000：无滤波器，采样频率为 fDTS 0001：fSAMPLING = fINT_CLK, N = 2 0010：fSAMPLING = fINT_CLK, N = 4 0011：fSAMPLING = fINT_CLK, N = 8 0100：fSAMPLING = fDTS / 2, N = 6 0101：fSAMPLING = fDTS / 2, N = 8 0110：fSAMPLING = fDTS / 4, N = 6 0111：fSAMPLING = fDTS / 4, N = 8 1000：fSAMPLING = fDTS / 8, N = 6 1001：fSAMPLING = fDTS / 8, N = 8 1010：fSAMPLING = fDTS / 16, N = 5

			1011：fSAMPLING = fDTS / 16, N = 6 1100：fSAMPLING = fDTS / 16, N = 8 1101：fSAMPLING = fDTS / 32, N = 5 1110：fSAMPLING = fDTS / 32, N = 6 1111：fSAMPLING = fDTS / 32, N = 8 注意：当 I1FLT [3：0] = 1, 2 或 3 时，公式中的 fDTS 由 INT_CLK 取代
IC1PRES	Bit3-2	R/W	**捕获比较器 1 输入预分频器** 该位定义了作用在 CC1 输入（I1）上的预分频比。当 CC1EN='0' （AD16C4Tn_CCEP 寄存器），预分频器将复位 。 00：无预分频器。每当捕获输入上检测到边沿时，发生捕获动作。 01：每发生 2 次事件，执行一次捕获 10：每发生 4 次事件，执行一次捕获 11：每发生 8 次事件，执行一次捕获
CC1SSEL	Bit 1-0	R/W	**捕获比较器 1 输入源选择** 该位定义了通道和使用的输入的方向。 00：CC1 通道配置为**输出** 01：CC1 通道配置为输入，捕获源为 **I1** 10：CC1 通道配置为输入，捕获源为 **I2** 11：CC1 通道配置为输入，捕获源为 **ITn 或 I1 的双边沿检出**。 只有当内部触发输入是通过 TSSEL 位（AD16C4Tn_SMCON 寄存器）选择时，该模式才运行 注意：当通道关闭（AD16C4Tn_CCEP 寄存器中的 CC1EN = '0'），CC1SSEL 为只写。

15. 5. 2. 12 通道捕获/比较模式寄存器 2（AD16C4Tn_CHMR2）

◆ 输出比较模式

通道捕获模式寄存器 2（AD16C4Tn_CHMR2）
偏移地址：$002C_H$
复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																CH4OCLREN	CH4OMOD			CH4OPEN	CH4OHSEN	CC4SSEL		CH3OCLREN	CH3OMOD			CH3OPEN	CH3OFEN	CC3SSEL	

Reserved	Bit 31-16	-	**保留**
CH4OCLREN	Bit 15	R/W	**通道 4 输出比较清零使能** 参考 CH1OCLREN 描述
CH4OMOD	Bit 14-12	R/W	**通道 4 输出比较模式** 参考 CH1OMOD 描述
CH4OPEN	Bit 11	R/W	**通道 4 输出比较预载入使能** 参考 CH1OPREN 描述
CH4OHSEN	Bit 10	R/W	**通道 4 输出比较高速使能** 参考 CH1OHSEN 描述
CC4SSEL	Bit 9-8	R/W	**通道 4 输出比较选择** 该位定义了通道以及使用的输入的方向（输入/输出）。 00：通道配置为输出 01：通道配置为输入，捕获源为 I4 10：通道配置为输入，捕获源为 I3 11：通道配置为输入，捕获源为 ITn 或 I1 的双边沿检出 仅当内部触发输入通过 TSSEL 位（AD16C4Tn_SMCON 寄存器）选择时，该模式才能工作。 注意：当通道为关闭状态时（AD16C4Tn_CCEP 中 CC4EN = '0'），CC4SSEL 为只写。
CH3OCLREN	Bit 7	R/W	**通道 3 输出比较清零使能** 参考 CH1OCLREN 描述
CH3OMOD	Bit 6-4	R/W	**通道 3 输出比较模式** 参考 CH1OMOD 描述
CH3OPEN	Bit 3	R/W	**通道 3 输出比较预载使能** 参考 CH1OPREN 描述
CH3OFEN	Bit 2	R/W	**通道 3 输出比较高速使能** 参考 CH1OHSEN 描述
CC3SSEL	Bit 1-0	R/W	**通道 3 捕获/比较选择**

			该位定义了通道和使用的输入的方向。 00：通道配置为**输出** 01：通道配置为输入，捕获源为 **I3** 10：通道配置为输入，捕获源为 **I4** 11：通道配置为输入，捕获源为 ITn 或 I1 的双边沿检出。 只有当内部触发输入是通过 TSSEL 位（AD16C4Tn_SMCON 寄存器）选择时，该模式才运行。 注意：当通道关闭（AD16C4Tn_CCEP 寄存器中的 CC3EN = '0'），CC3SSEL 为只写

◆ 输入捕获模式

通道捕获模式寄存器 2（AD16C4Tn_CHMR2）

偏移地址：$002C_H$

复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																I4FLT				IC4PRES		CC4SSEL		I3FLT				IC3PRES		CC3SSEL	

Reserved	Bit 31-16	-	**保留**
I4FLT	Bit 15-12	R/W	**通道 4 输入捕获滤波器** 参考 I1FLT 描述
IC4PRES	Bit 11-10	R/W	**通道 4 输入捕获预分频器** 参考 IC1PRES 描述
CC4SSEL	Bit 9-8	R/W	**选择通道 4 输入捕获源** 该位定义了通道和使用的输入的方向。 00：通道配置为**输出** 01：通道配置为输入，捕获源为 **I4** 10：通道配置为输入，捕获源为 **I3** 11：通道配置为输入，捕获源为 **ITn 或 I1 的双边沿检出** 只有当内部触发输入是通过 TSSEL 位（AD16C4Tn_SMCON 寄存器）选择时，该模式才运行 注意：当通道关闭（AD16C4Tn_CCEP 寄存器中的 CC4EN = '0'），CC4SSEL 为只写。
I3FLT	Bit 7-4	R/W	**通道 3 输入捕获滤波器** 参考 I1FLT 描述
IC3PRES	Bit3-2	R/W	**通道 3 输入捕获预分频器**

			参考 IC1PRES 描述
CC3SSEL	Bit 1-0	R/W	**选择通道 3 输入捕获源** 该位定义了通道和使用的输入的方向。 00：通道配置为**输出** 01：通道配置为输入，捕获源为 **I3** 10：通道配置为输入，捕获源为 **I4** 11：通道配置为输入，捕获源为 ITn 或 I1 的双边沿检出 只有当内部触发输入是通过 TSSEL 位（AD16C4Tn_SMCON 寄存器）选择时，该模式才运行 注意：当通道关闭（AD16C4Tn_CCEP 寄存器中的 CC3EN = '0'），CC3SSEL 为只写。

15. 5. 2. 13 捕获/比较使能极性寄存器（AD16C4Tn_CCEP）

通道捕获/比较使能寄存器（AD16C4Tn_CCEP）
偏移地址：0030_H
复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																		CC4POL	CC4EN	CC3NPOL	CC3NE	CC3POL	CC3EN	CC2NPOL	CC2NE	CC2POL	CC2EN	CC1NPOL	CC1NE	CC1POL	CC1EN

Reserved	Bit 31-14	-	**保留**
CC4POL	Bit 13	R/W	**通道 4 捕获/比较输出极性** 参考 CC1POL 描述
CC4EN	Bit 12	R/W	**使能通道 4 捕获/比较输出** 参考 CC1EN 描述
CC3NPOL	Bit 11	R/W	**通道 3 捕获/比较互补输出极性** 参考 CC1NPOL 描述
CC3NE	Bit 10	R/W	**使能通道 3 捕获/比较互补输出** 参考 CC1NE 描述
CC3POL	Bit9	R/W	**通道 3 捕获/比较输出极性** 参考 CC1POL 描述
CC3EN	Bit 8	R/W	**使能通道 3 捕获/比较输出** 参考 CC1EN 描述
CC2NPOL	Bit 7	R/W	**通道 2 捕获/比较 2 互补输出极性** 参考 CC1NPOL 描述
CC2NE	Bit 6	R/W	**使能通道 2 捕获/比较互补输出** 参考 CC1NE 描述
CC2POL	Bit 5	R/W	**通道 2 捕获/比较输出极性** 参考 CC1POL 描述
CC2EN	Bit 4	R/W	**使能通道 2 捕获/比较输出** 参考 CC1EN 描述
CC1NPOL	Bit 3	R/W	**通道 1 捕获/比较互补输出极性** **通道配置为输出：** 0：CH1ON 高有效。 1：CH1ON 低有效。 **通道配置为输入：** 该位需和 CC1POL 一起使用来定义输入边沿的极性。参考 CC1POL 描述。 注意：对于有互补输出的通道，该位设置为预载值。如果 AD16C4Tn_CON2 寄存器中的 CCPCEN 位设置为 1，则只有当 COM 事件发生时，CC1NPOL 有效位才会设置为预载值中新的值。 注意：当 AD16C4Tn_BDCFG 寄存器中的

			LOCKLVL 位被设置为锁定级别 2 或 3，且 CC1SSEL=00（通道为输出模式），该位将不可写。
CC1NE	Bit 2	R/W	**使能通道 1 捕获/比较互补输出** 0：关闭 - CH1ON 无效。CH1ON 电平取决于 GOEN, OFFSSI, OFFSSR, OISS1, OISS1N 和 CC1EN 的功能 1：开启 - CH1ON 为对应输出引脚上的输出信号，由 GOEN, OFFSSI, OFFSSR, OISS1, OISS1N 和 CC1EN 决定。 注意：对于有互补输出的通道，该位设置为预载值。如果 AD16C4Tn_CON2 寄存器中的 CCPCEN 位设置为 1，则只有当 COM 事件发生时，CC1NE 有效位才会设置为预载值中新的值
CC1POL	Bit 1	R/W	**通道 1 捕获/比较输出极性** **通道配置为输出：** 0：CH1O 高有效 1：CH1O 低有效 **通道配置为输入：** CC1NPOL/CC1POL 为触发和捕获操作选择 I1 边沿检出和 I2 边沿检出的有效极性。 00：正向/上升沿 电路对 In 边沿检出的上升沿敏感（在复位，外部时钟或触发模式下，进行捕获或触发），In 边沿检出不反向（门控模式或编码器模式下，进行触发）。 01：反向/下降沿 电路对 In 边沿检出的下降沿敏感（在复位，外部时钟或触发模式下，进行捕获或触发），In 边沿检出反向（门控模式或编码器模式下，进行触发）。 10：预留，不要使用该配置。。 11：正向/上升沿和下降沿 电路对 In 边沿检出的上升沿和下降沿均敏感（在复位，外部时钟或触发模式下，进行捕获或触发），In 边沿检出不反向（门控模式或编码器模式下，进行触发）。 注意：对于有互补输出的通道，该位设置为预载值。如果 AD16C4Tn_CON2 寄存器中的 CCPCEN 位设置为 1，则只有当 COM 事件发生时，CC1POL 有效位才会设置为预载值中新的值 。 注意：当 AD16C4Tn_BDCFG 寄存器中的 LOCKLVL 位被设置为锁定级别 2 或 3，且 CC1SSEL=00（通道为输出模式），该位将不可写。
CC1EN	Bit 0	R/W	**使能通道 1 捕获/比较输出** **通道配置为输出：** 0：关闭 - CH1O 无效。CH1ON 电平取决于

			GOEN, OFFSSI, OFFSSR, OISS1, OISS1N 和 CC1EN 的功能 1：开启 - CH1O 为对应输出引脚上的输出信号，由 GOEN, OFFSSI, OFFSSR, OISS1, OISS1N 和 CC1EN 决定 **通道配置为输入：** 该位决定了计数值是否能捕获到输入捕获/比较寄存器 1（AD16C4Tn_CCVAL1）。 0：禁止捕获。 1：使能捕获。 注意：对于有互补输出的通道，该位设置为预载值。如果 AD16C4Tn_CON2 寄存器中的 CCPCEN 位设置为 1，则只有当 COM 事件发生时，CC1EN 有效位才会设置为预载值中新的值。

15. 5. 2. 14 计数器（AD16C4Tn_COUNT）

计数器寄存器（AD16C4Tn_COUNT）
偏移地址：0034_H
复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-16	-	保留
CNTV	Bit 15-0	R/W	计数值

15. 5. 2. 15 时钟预分频器（AD16C4Tn_PRES）

时钟预分频器寄存器（AD16C4Tn_PRES）
偏移地址：0038_H
复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-16	-	保留
PSCV	Bit 15-0	R/W	**时钟预分频器值** 计数器时钟频率（CK_CNT）=fCK_PSC /（PSCV[15：0] + 1）。 每发生一次更新事件（包括当计数器由 AD16C4Tn_SGE 寄存器中的 SGU 位清零或当配置为复位模式时，通过触发控制器清零），PSCV 包含的值需填入到有效的预分频寄存器内。

15. 5. 2. 16 计数器自动装载寄存器（AD16C4Tn_AR）

计数器自动装载寄存器（AD16C4Tn_AR）
偏移地址：$003C_H$
复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-16	-	保留
ARRV	Bit 15-0	R/W	**计数器自动装载值** ARRV 中的值将被载入实际的自动重载寄存器中。当自动重载值为空，计数器被屏蔽。

15. 5. 2. 17 重复计数寄存器（AD16C4Tn_REPAR）

重复计数寄存器（AD16C4Tn_REPAR）
偏移地址：0040_H
复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-8	-	保留
REPV	Bit 7-0	R/W	**重复计数值** 当预载寄存器使能，该位允许用户设置比较寄存器的更新率（例如：预载到有效寄存器的周期性传输），同样也可以设置更新中断生成率。每次当 REPV_CNT 的相关递减计数器递减至 0，会产生更新事件，会从 REPV 值重新计数。因为只有当发生重复更新事件 U_RC 时，REPV_CNT 才会重新载入 REPV 值，所以只有在发生下一次重复更新事件时，写入 AD16C4Tn_REPAR 寄存器的值才会生效。 即，在 PWM 模式下，（REPV+1）相当于： –在边沿对齐模式下，（REPV+1）对应的是 PWM 的周期数 –在中央对齐模式下，（REPV+1）对应的是 1/2 PWM 的周期数

15. 5. 2. 18 通道捕获/比较寄存器 1（AD16C4Tn_CCVAL1）

通道捕获/比较寄存器 1 （AD16C4Tn_CCVAL1）
偏移地址：0044_H
复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																CCRV1															

Reserved	Bit 31-16	-	保留
CCRV1	Bit 15-0	R/W	**捕获/比较值 1** **如果通道 CCn 配置为输出：** CCRVn 中的值将被载入实际的捕获/比较寄存器中（预载值）。 如果在 AD16C4Tn_CCMRn 寄存器中的预载功能没有选中，CCRVn 中的值将被永久载入；否则，每当发生更新事件，预载值将会复制到有效的捕获/比较寄存器中。有效捕获/比较寄存器中包含的值将会与 AD16C4Tn_COUNT 中的值进行比较，并在 OCn 上输出。 **如果通道 CCn 配置为输入：** CCRVn 为由上一个输入捕获事件（ICn）传输的计数值。

15. 5. 2. 19 通道捕获/比较寄存器 2（AD16C4Tn_CCVAL2）

通道捕获/比较寄存器 2（AD16C4Tn_CCVAL2）
偏移地址：0044_H
复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																CCRV2															

Reserved	Bit 31-16	-	保留
CCRV2	Bit15-0	R/W	**通道捕获/比较值 2** 参考 CCRV1 描述

15. 5. 2. 20 通道捕获/比较寄存器 3（AD16C4Tn_CCVAL3）

捕获/比较寄存器 3（AD16C4Tn_CCVAL3）
偏移地址：$004C_H$
复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																CCRV3															

Reserved	Bit 31-16	-	**保留**
CCRV3	Bit 15-0	R/W	**捕获/比较值 3** 参考 CCRV1 描述

15. 5. 2. 21 通道捕获/比较寄存器 4（AD16C4Tn_CCVAL4）

通道捕获/比较寄存器 4（AD16C4Tn_CCVAL4）
偏移地址：0050_H
复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																CCRV4															

Reserved	Bit 31-16	-	**保留**
CCRV4	Bit 15-0	R/W	**通道捕获/比较值 4** 参考 CCRV1 描述

15. 5. 2. 22 刹车和死区配置寄存器（AD16C4Tn_BDCFG）

刹车和死区配置寄存器（AD16C4Tn_BDCFG）
偏移地址：0054_H
复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																GOEN	AOEN	BRKP	BRKEN	OFFSSR	OFFSSI	LOCKLVL		DT							

Reserved	Bit 31-16	-	保留
GOEN	Bit 15	R/W	**通道主要输出使能** 一旦刹车输入有效，该位会由硬件异步清零。该位可由软件置 1 或自动置 1，取决于 AOEN 位。该位仅作用于配置为输出的通道。 0：OC 和 OCN 输出禁止或强制为空闲状态。 1：如果 OC 和 OCN 各自的使能位都置 1（AD16C4Tn_CCEP 寄存器中的 CCnEN, CCnNE），则 OC 和 OCN 输出使能。
AOEN	Bit 14	R/W	**通道自动输出使能** 0：GOEN 仅可由软件置位 1：在下一个更新事件发生时（如果刹车输入无效），GOEN 可由软件或自动置位。 注意：当 AD16C4Tn_BDCFG 寄存器中的 LOCKLVL 位已被设置为锁定级别 1，则该位不可更改。
BRKP	Bit 13	R/W	**选择通道刹车极性** 0：刹车输入 BRKP 为低有效 1：刹车输入 BRKP 为高有效 注意：当 AD16C4Tn_BDCFG 寄存器中的 LOCKLVL 位已被设置为锁定级别 1，则该位不可更改 注意：任何对该位的写操作都要延时 1 个 APB 时钟周期后才变为有效。
BRKEN	Bit 12	R/W	**使能刹车** 0：刹车输入（BRKP 和 CCS 时钟失效事件）禁止 1：刹车输入（BRKP 和 CCS 时钟失效事件）使能 注意：当 AD16C4Tn_BDCFG 寄存器中的 LOCKLVL 位已被设置为锁定级别 1，则该位不可更改 注意：任何对该位的写操作都要延时 1 个 APB 时

			钟周期后才变为有效。
OFFSSR	Bit 11	R/W	运行模式下的无效状态选择位 该位使用于，当 GOEN=1 时，被配置为输出并使用互补输出的通道。如果计时器中没有使用互补输出，则 OFFSSR 不使用。 0：无效状态时, OC/OCN 输出禁止（OC/OCN 使能输出信号=0）。 1：无效状态时，当 CCnEN=1 或 CCnNE=1 时，便使能 OC/OCN 输出并将其设为无效电平。（OC/OCN 使能输出信号=1） 注意：当 AD16C4Tn_BDCFG 寄存器中的 LOCKLVL 位已被设置为锁定级别 2，则该位不可更改。
OFFSSI	Bit 10	R/W	空闲模式下的空闲状态选择位该位使用于，当 GOEN=0 时，被配置为输出通道 0：无效状态时, OC/OCN 输出禁止（OC/OCN 使能输出信号=0）。 1：无效状态时，当 CCnEN=1 或 CCnNE=1，便将 OC/OCN 输出首先强制为其空闲电平。（OC/OCN 使能输出信号=1） 注意：当 AD16C4Tn_BDCFG 寄存器中的 LOCKLVL 位已被设置为锁定级别 2，则该位不可更改。
LOCKLVL	Bit 9-8	R/W	**锁定级别配置** 针对软件错误，该位提供写保护。 00：**锁定关闭**–不提供写保护 01：**锁定级别 1** = AD16C4Tn_BDCFG 寄存器中的 DT，AD16C4Tn_CON2 寄存器中的 OISSx 和 OISSxN，和 AD16C4Tn_BDCFG 寄存器中的 BRKEN/BRKP/AOEN 不再可写。 10：**锁定级别 2** = 锁定级别 1 + CC 极性位（AD16C4Tn_CCEP 寄存器中的 CCnPOL/CCnNPOL，只要相关通道由 CCnSSEL 配置为输出）以及 OFFSSR 和 OFFSSI 都不再可写。 11：**锁定级别 3** = 锁定级别 2 + CC 控制位（AD16C4Tn_CHMRn 寄存器中的 CHnOMOD 和 CHnOPEN，只要相关通道由 CCnSSEL 配置为输出）都不再可写。 注意：锁定配置为仅在复位后可写。一旦 AD16C4Tn_BDCFG 已写，其设置内容在下一个复位前都处于冻结状态。
DT	Bit 7-0	R/W	死区延时设置值该位定义了互补输出之间插入的死区时间。DT 对应的就是该时间段。

			DT[7：5]=0xx => DT=DT[7:0]x t_{dtg}，式中 $t_{dtg}=t_{DTS}$ **DT[7：5]=10x** => DT=（64+DT[5:0]）xt_{dtg}，式中 $t_{dtg}=2xt_{DTS}$ **DT[7：5]=110**=> DT=（32+DT[4:0]）xt_{dtg}，式中 $t_{dtg}=8xt_{DTS}$ **DT[7：5]=111** => DT=（32+DT[4：0]）xt_{dtg}，式中 $t_{dtg}=16xt_{DTS}$ 注意：当 AD16C4Tn_BDCFG 寄存器中的 LOCKLVL 位已被设置为锁定级别 1，2 或 3，则该位不可更改

15. 5. 2. 23 DMA使能寄存器（AD16C4Tn_DMAEN）

DMA 使能寄存器（AD16C4Tn_DMAEN）

偏移地址：058_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																									TDS	COMDS	CC4DS	CC3DS	CC2DS	CC1DS	UDS

Reserved	Bit 31-7	-	保留，必须保持为复位值
TDS	Bit 6	R/W	**触发DMA请求使能** 0：DMA请求禁止 1：DMA请求使能
COMDS	Bit 5	R/W	**COM DMA访问使能** 0：DMA请求禁止 1：DMA请求使能
CC4DS	Bit 4	R/W	**捕获/比较值 4 DMA 访问使能** 0：DMA 请求禁止 1：DMA 请求使能
CC3DS	Bit 3	R/W	**捕获/比较值 3 DMA 访问使能** 0：DMA 请求禁止 1：DMA 请求使能
CC2DS	Bit 2	R/W	**捕获/比较值 2 DMA 访问使能** 0：DMA 请求禁止 1：DMA 请求使能
CC1DS	Bit 1	R/W	**捕获/比较值 1 DMA 访问使能** 0：DMA 请求禁止 1：DMA 请求使能
UDS	Bit 0	R/W	**更新 DMA 请求使能** 0：DMA 请求禁止 1：DMA 请求使能

第16章 通用定时器（GP16C2T0~2）

16. 1 概述

芯片内部共有 3 个 16 位 2 通道通用定时器（GP16C2T0~2），GP16C2Tn 包含一个 16 位自动重载计数器，该计数器由可配置的预分频器驱动。

通用定时器 16 位 2 通道（GP16C2Tn）的用途广泛，可测量信号脉冲长度（输入捕获）或输出脉冲波形（比较输出、PWM 及带死区时间插入的互补 PWM）。

16. 2 特性

- 16 位递增自动加载计数器
- 16 位可编程预分频器，可在定时器运行中对计数器工作时钟进行 1 到 65536 间的任意分频
- 带有两个独立信道，每个信道支持以下功能：
  - 输入捕获
  - 输出比较
  - PWM 输出
  - 单脉冲输出
- 通道 1 支持互补输出，死区时间可配
- 同步电路可用于外部信号控制定时器及内部互联多个定时器
- 在给定数目的计数周期之后更新重复计数寄存器
- 支持刹车功能，刹车后定时器输出状态可控
- 支持中断/DMA：
  - 更新事件：计数器上溢，计数器初始化（通过软件或内/外部触发）
  - 触发事件（计数器起始、停止、初始化或内/外触发计数）
  - 通信事件
  - 输入捕获
  - 输出比较
  - 刹车输入

16. 3 结构图

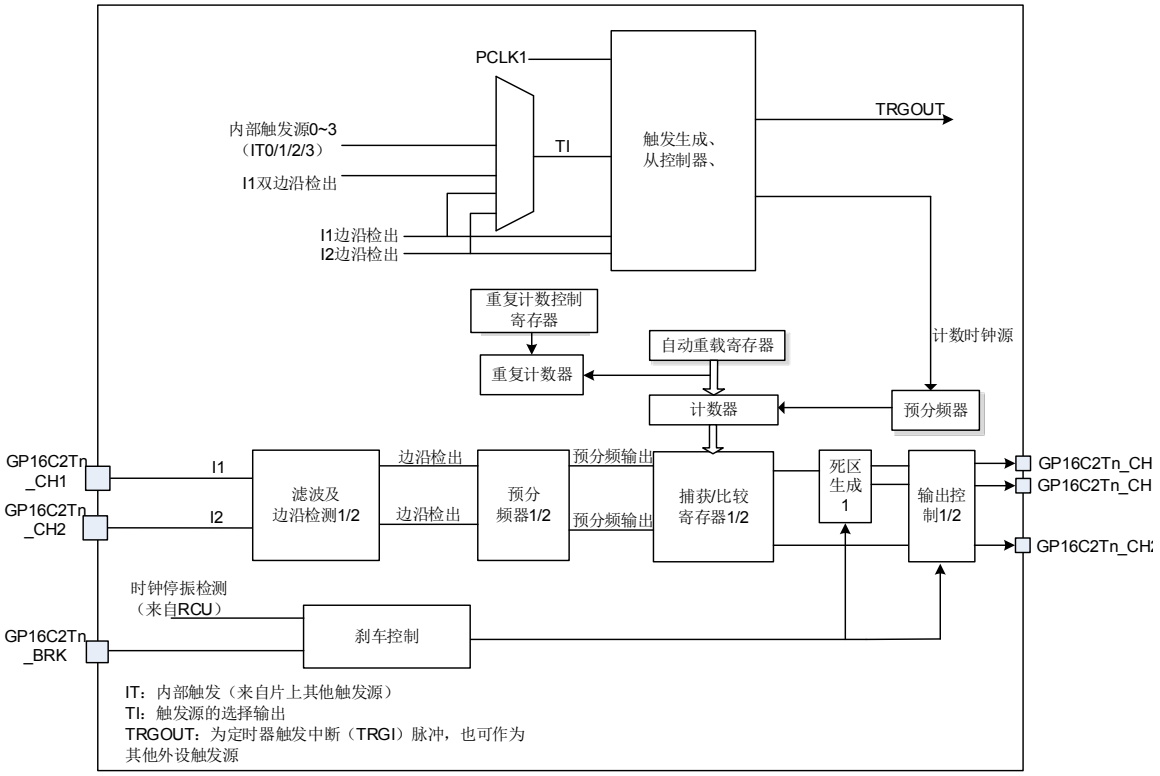


图 17-1 GP16C2Tn 结构框图

16. 4 功能描述

16. 4. 1 预分频器

定时器包含一个 16-bit 的计数器（GP16C2Tn_COUNT），计数时钟由预分频寄存器（GP16C2Tn_PRES）进行分频。计数周期由自动重载计数器（GP16C2Tn_AR）设定。重复计数寄存器则可指定计数周期数目（GP16C2Tn_REPAR）。

自动重载寄存器（GP16C2Tn_AR）是一个可缓存的寄存器。当 GP16C2Tn_CON1 寄存器的 ARPEN 位为 0 时，GP16C2Tn_AR 寄存器重载功能失效，GP16C2Tn_AR 就是有效寄存器；ARPEN 为 1 时，GP16C2Tn_AR 寄存器具有重载功能，产生更新事件（UEV）时，加载值（GP16C2Tn_AR 寄存器值）更新到影子寄存器。

当 GP16C2Tn_CON1 寄存器中 DISUE 位为 0 时，计数器计数上溢时会产生更新事件（UEV）。同样，软件方式也可产生更新事件。GP16C2Tn_CON1 寄存器的 CNTEN 置位时，计数器开始计数。

注：计数器在 CNTEN 位置位 1 个时钟周期后开始计数。

预分频器可对定时器工作时钟进行 GP16C2Tn_PRES 寄存器值+1 次分频。由于 GP16C2Tn_PRES 是一个可重载寄存器，因此，定时器工作时可以对该寄存器进行修改，修改值在下次更新事件（UEV）后有效。

下图给出了定时器运行过程中改变预分频值时计数器的计数情况。

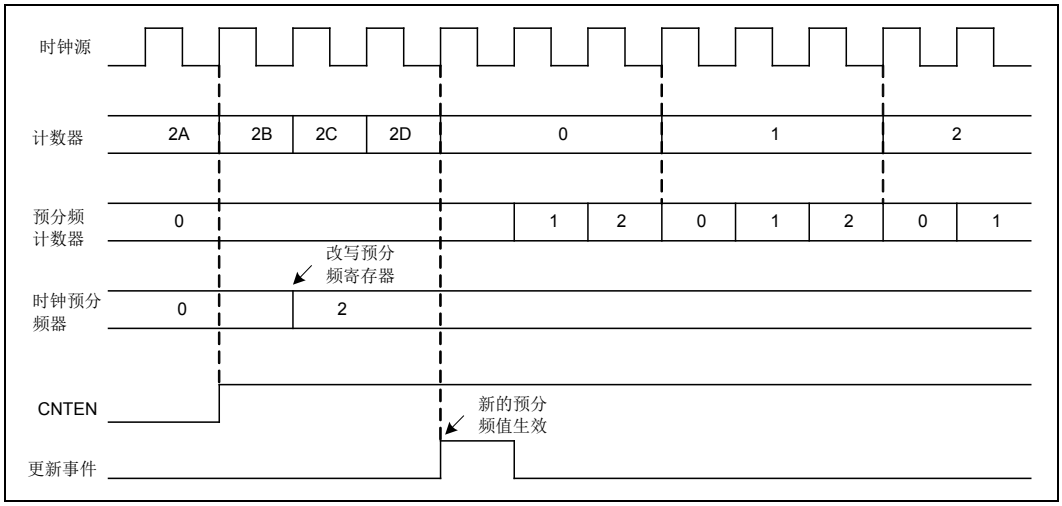


图 17-2　预分频值计数时序图

16. 4. 2　重复计数器

重复计数器用于控制发生多少次上溢出后产生更新事件。

重复计数器递减：

◇　递增模式的每次上溢

GP16C2Tn_REPAR 寄存器是一个可缓存寄存器。软件（置位 GP16C2Tn_SGE 寄存器中的 SGU 位）或硬件从机模式控制方式产生更新事件时，无论重复计数器为何值，GP16C2Tn_REPAR 寄存器中的值都会立即更新到重复计数器的影子寄存器中。

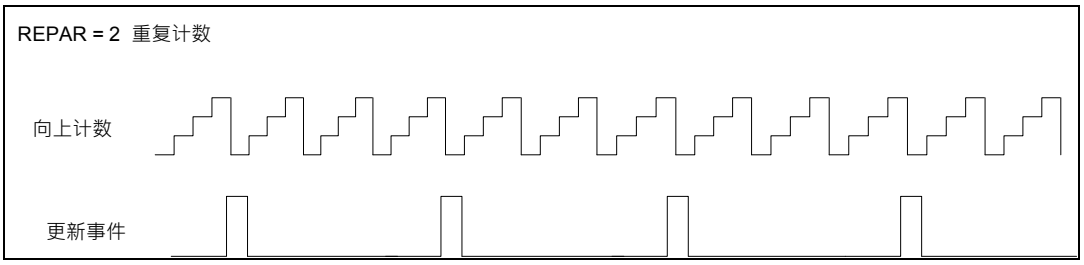


图 17-3　重复计数器工作模式

注意：置位 GP16C2Tn_SGE 寄存器中的 SGU 位也可以产生更新事件。

16. 4. 3　时钟源

计数器工作时钟可以选择内部时钟(INT_CLK)、外部时钟源 1（I1、I2、I3、I4），内部触发输入（IT0、IT1、IT2、IT3）

16. 4. 3. 1　内部时钟源（INT_CLK）

若从模式控制器被关闭（GP16C2Tn_SMCON 寄存器内，SMODS= "000"），则 CNTEN，GP16C2Tn_SGE.SGU 位为实际控制位，这些位只能软件修改（SGU 位除外，仍硬件

自动清除）。一旦 CNTEN 位被写为'1'，预分频器就由内部 INT_CLK 提供时钟。

下图给出了通常模式下控制电路和递增计数的情况，没有分频。

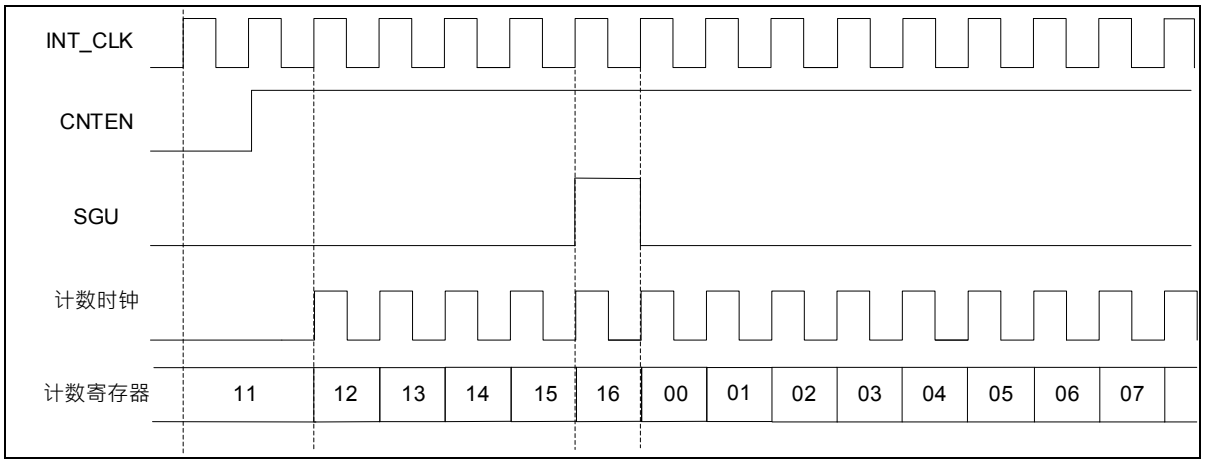


图 17-4　采用内部时钟计数

16. 4. 3. 2　外部时钟源 1

GP16C2Tn_SMCON 寄存器的 SMODS= "111"时，可选择外部时钟源 1。计数器可根据选定的上升沿或下降沿计数。

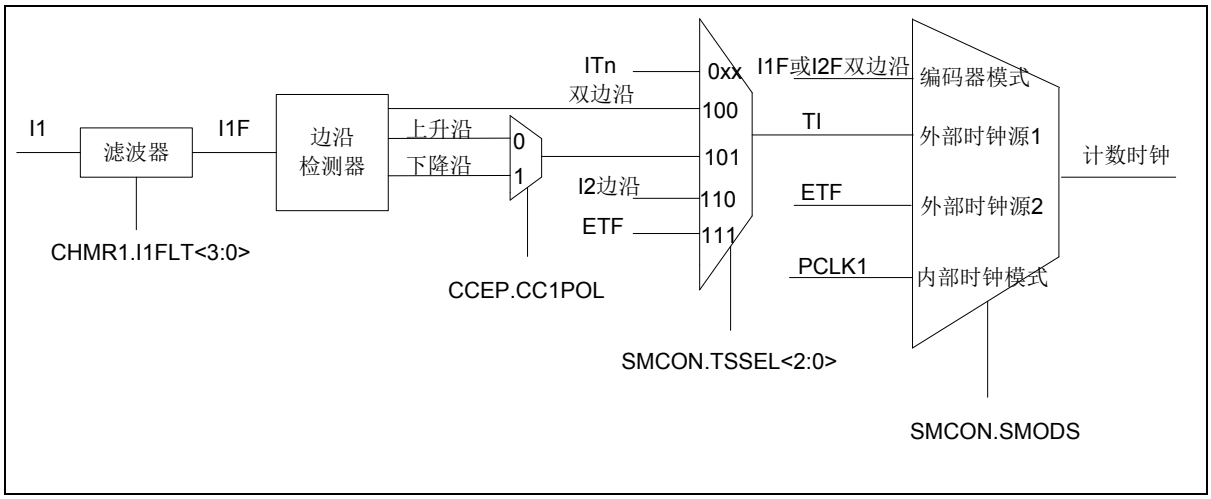


图 17-5　外部时钟连接

配置计数器为外部时钟源 1，步骤如下：

1. GP16C2Tn_SMCON 寄存器中 SMODS = "111"，配置定时器外部时钟模式 1。
2. 设置 GP16C2Tn_SMCON 寄存器中的 TSSEL 选择外部时钟源。
3. 如外部时钟源为 I1，可配置 GP16C2Tn_CHMR1 寄存器 CC1SSEL = "01"，配置信道 1 检测 I1 输入的上升沿；设置 GP16C2Tn_CCEP 寄存器中 CC1POL = '0'，选择极性为上升沿。
4. 写 GP16C2Tn_CHMR1 寄存器的 I1FLT<3:0>位，配置输入滤波器时间（若没有滤波器需求，维持 I1FLT = "0000"）。

5. GP16C2Tn_CON1 寄存器中 CNTEN = '1'，使能计数器。

当 I1 上出现一次上升沿时，计数器计数一次且 TRGI 标志位置位。

16. 4. 3. 3 内部触发输入（ITn）

当 GP16C2Tn_SMCON 寄存器的 SMODS= "111"，选定内部触发模式。计数器根据选定的内部输入端的上升或下降沿计数。

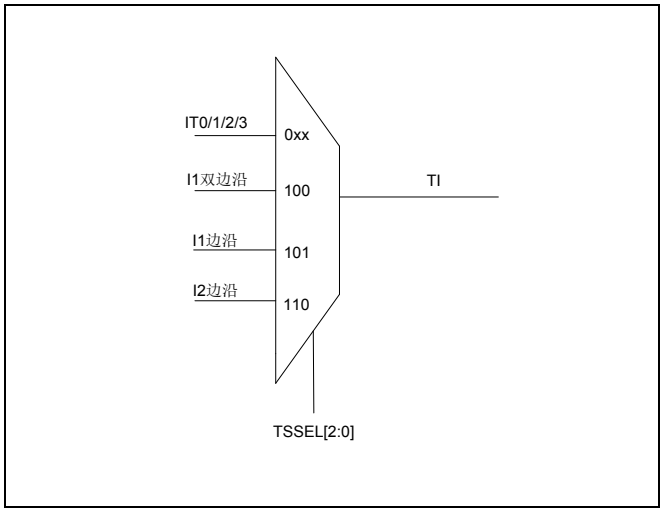


图 17-6 ITn 外部时钟连接

配置计数器在 ITn 输入端的上升沿递增计数，步骤如下：

1. GP16C2Tn_SMCON 寄存器中 SMODS = "111"，配置外部时钟模式 1。
2. GP16C2Tn_SMCON 寄存器的 TSSEL = "0xx"，选定 ITn 作为触发输入源。
3. GP16C2Tn_CON1 寄存器的 CNTEN = '1'，使能计数器。

ITn 产生上升沿时，计数器计数一次。ITn 上升沿与实际时钟间的延时，取决于 ITn 输入的再同步电路。

16. 4. 4 计数模式

16. 4. 4. 1 递增计数模式

定时器配置为递增模式，计数器从 0 开始递增，直至 GP16C2Tn_AR 寄存器值；然后从 0 重新开始计数并产生一个更新事件（UEV）。当 GP16C2Tn_REPAR 寄存器不为 0 时，则在 GP16C2Tn_REPAR+1 次计数后产生更新事件。

当有更新事件（UEV）产生时，预装载寄存器会更新到影子寄存器，更新标志位（GP16C2Tn_RIF 寄存器中的 UI 位）置位（取决于 UERSEL 位）：

◇ 更新 GP16C2Tn_REPAR 寄存器的值到影子寄存器

◇ 更新 GP16C2Tn_AR 寄存器的值到影子寄存器

◇ 更新 GP16C2Tn_PRES 寄存器的值到影子寄存器

下图为 GP16C2Tn_AR = 0x16，预分频设为 2 分频时的计数器时序。

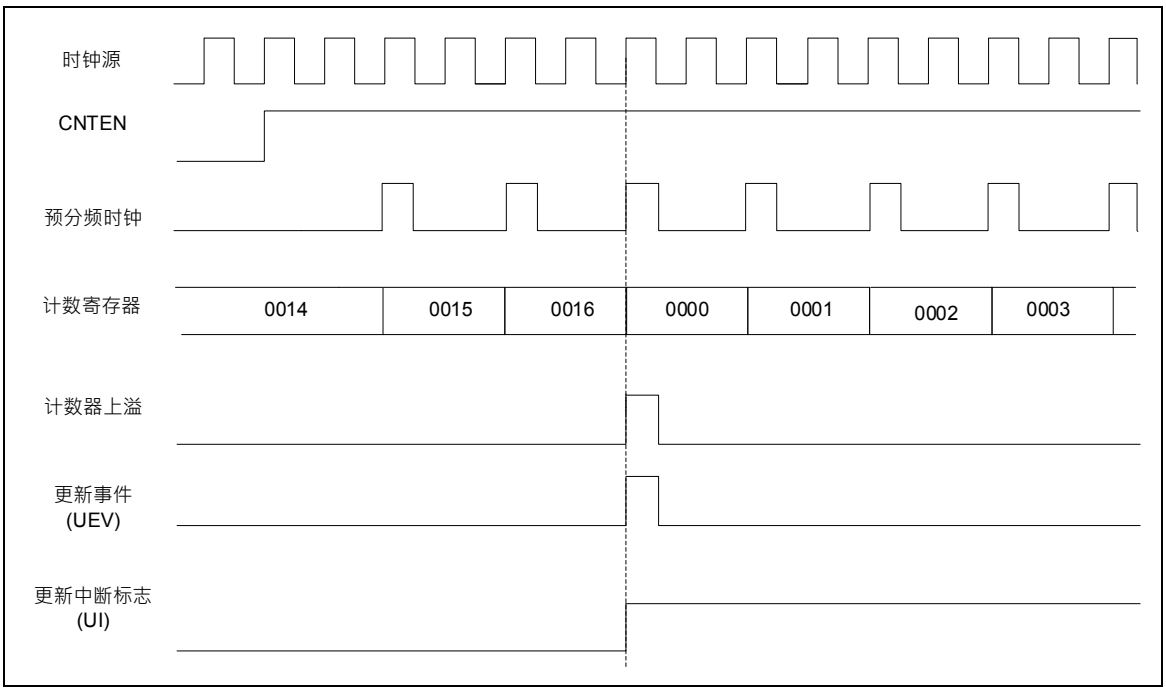


图 17-7　计数器递增计数时序图

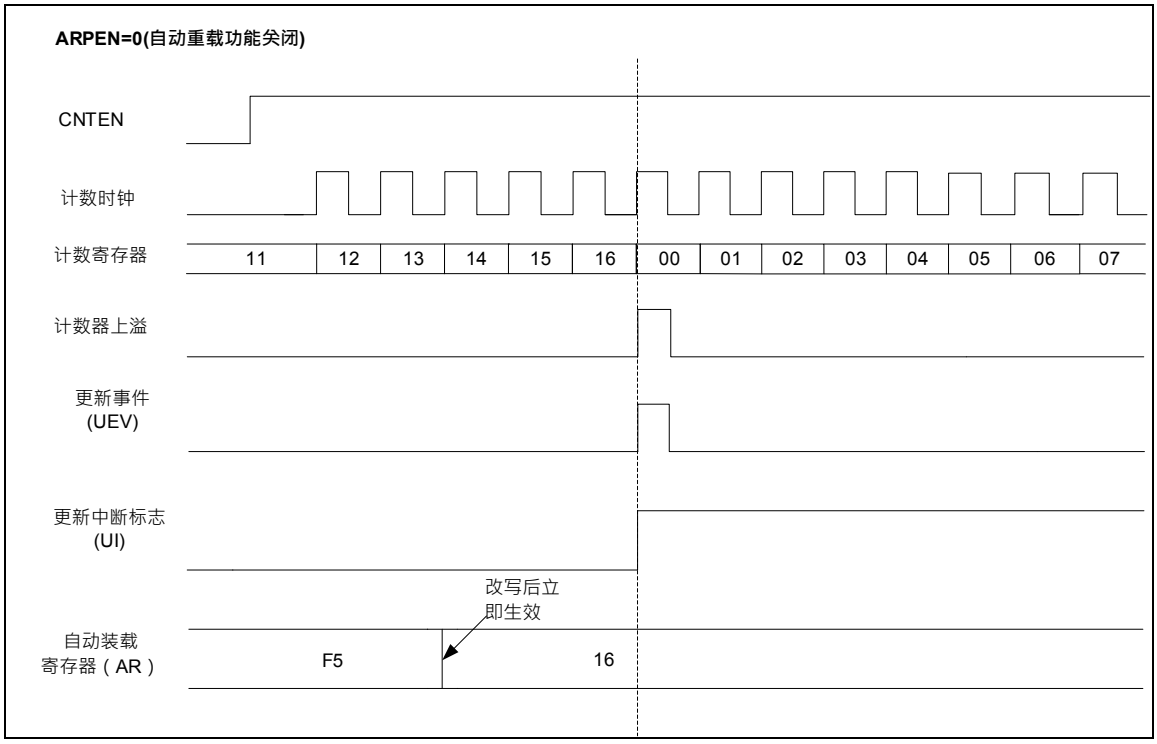


图 17-8　当 ARPEN=0 时计数器时序图

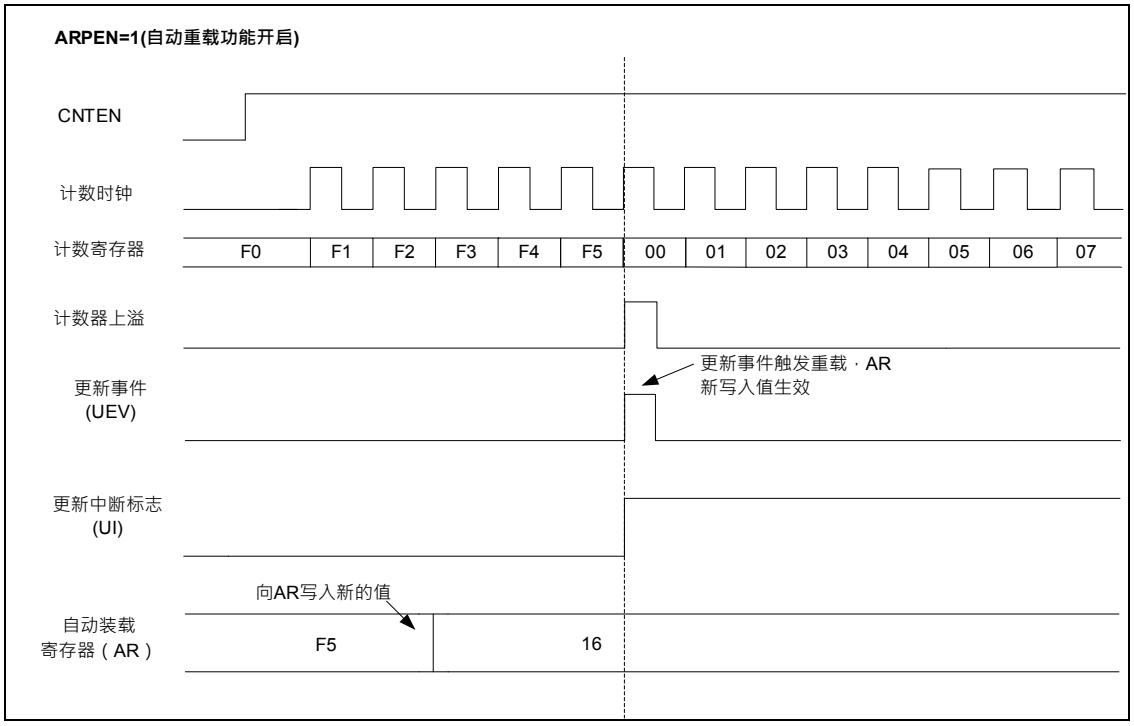


图 17-9　当 ARPEN=1 时计数器时序图

16. 4. 5　捕获/比较通道

输入电路对 In 输入端的信号进行采样，产生一个经过滤波的信号 InF。之后，一个可极性选择的边沿检测器产生 In 边沿检测信号，该信号可作为从模式控制器的触发输入或作为捕获控制命令，且信号经过分频后进入捕获寄存器。

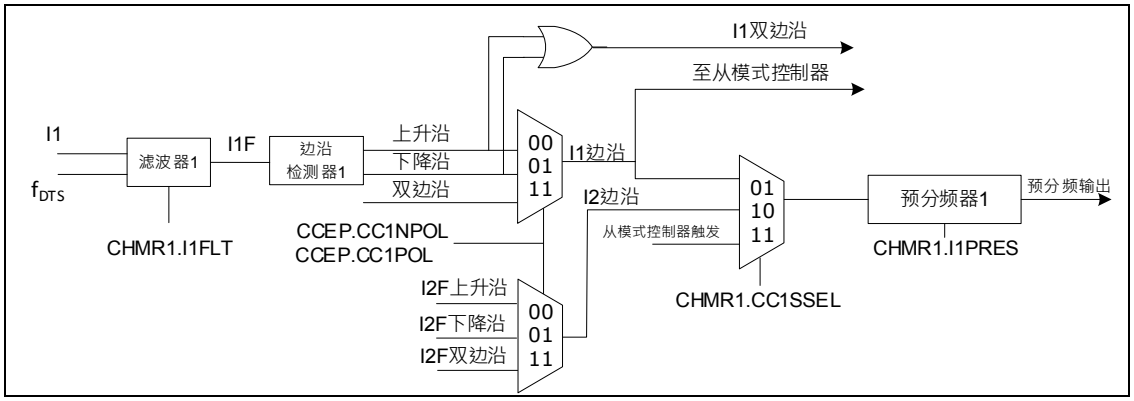


图 17-10　捕获/比较通道

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

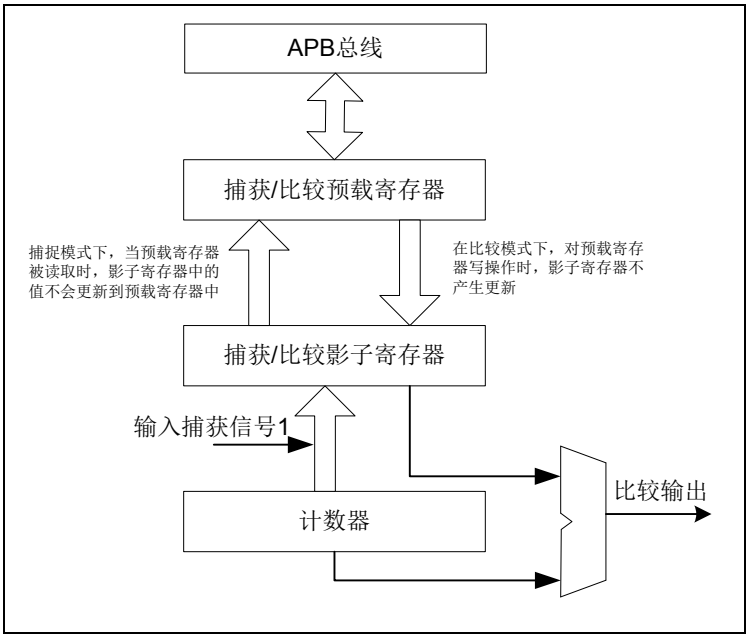


图 17-11　捕获/比较通道 1 结构图

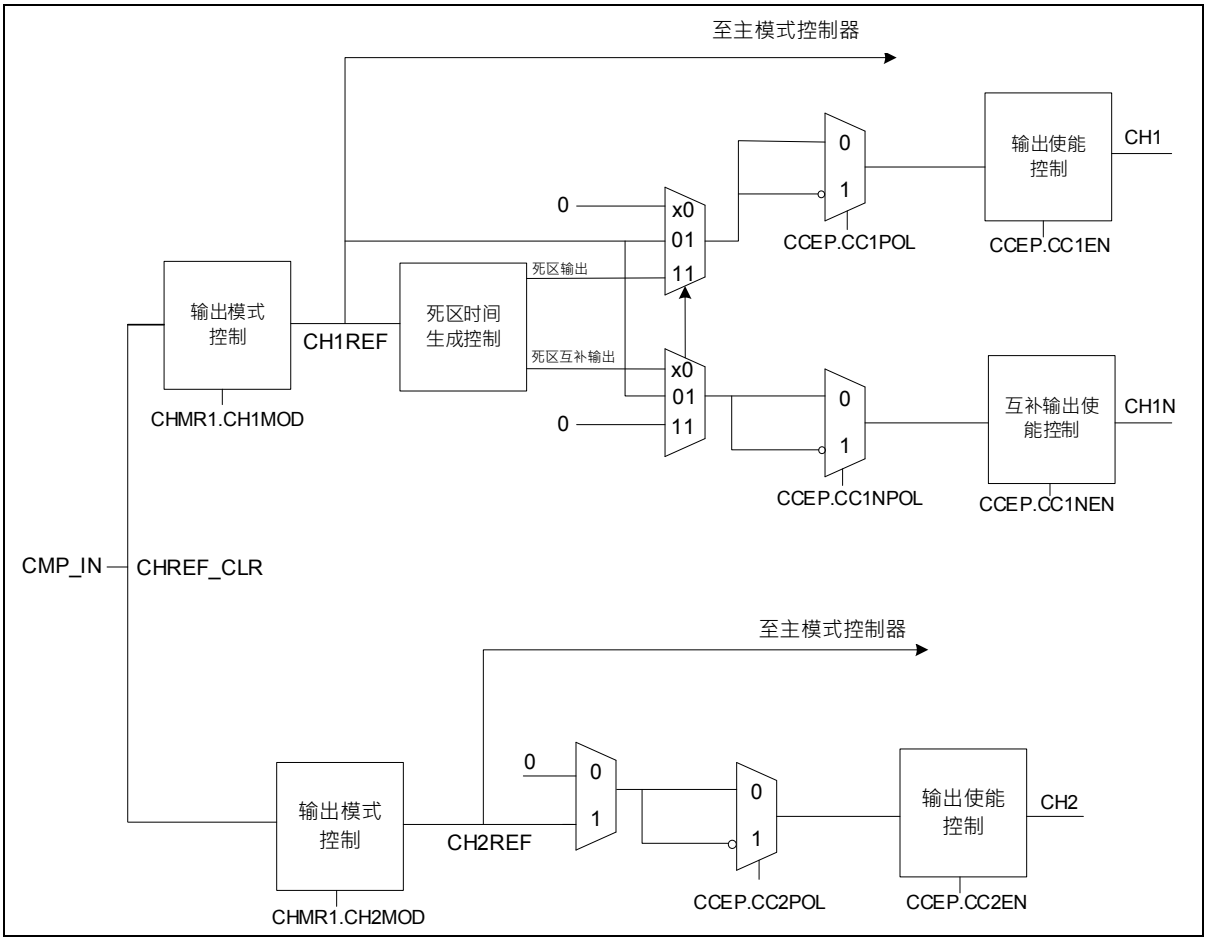


图 17-12　捕获/比较信道的输出部分

16. 4. 6　输入捕获模式

在输入捕获模式下，当检测到 In 上相应信号变化时，计数器的值就会被锁存到捕获/比较

寄存器（GP16C2Tn_CCVALn）寄存器中。当捕获发生时，相应的 CHnI 标志位（GP16C2Tn_RIF）会置位，同时会触发中断或 DMA（如果使能）请求。若发生捕获时，CHnI 标志位已经置位，则过捕获 CHnOVI 标志位（GP16C2Tn_RIF）置位。软件于 GP16C2Tn_ICR 对 CHnI 与 CHnOVI 位写'1'可以清零 CHnI 标志位与 CHnOVI 标志位。

以下为以 I1 输入上升沿作为捕获输入时的流程：

1. 选择有效输入端：GP16C2Tn_CCVAL1 必须连接到 I1 输入端，因此需将 GP16C2Tn_CHMR1 寄存器中的 CC1SSEL 位写"01"。只要 CC1SSEL 不为"00"，信道被配置为输入且 GP16C2Tn_CCVAL1 寄存器为只读。
2. 根据定时器连接的输入信号，配置输入滤波器的持续时间。当输入信号翻转时，前 5 个内部时钟信号内信号是不稳定的，因此必须配置滤波器的时间大于 5 个时钟周期。当 I1 检测到新的电平，连续 8 次采样可确认电平变化有效。
3. 选择 I1 信道的有效边沿变换。GP16C2Tn_CCEP 寄存器中的 CC1POL 写'0'（上升沿）。
4. 配置输入预分频器。
5. 置位 GP16C2Tn_CCEP 寄存器中的 CC1EN 位，使能捕获计数器的值到捕获寄存器。
6. 如有需要，置位 GP16C2Tn_IER 寄存器中的 CH1I 位，使能中断请求。置位 GP16C2Tn_DMAEN 寄存器中的 CH1DE 位，使能 DMA 请求。

当发生输入捕获时：

1. 有效边沿产生，GP16C2Tn_CCVAL1 寄存器获取计数器的值。
2. CH1I 标志位置位（中断标志）。若至少 2 个连续的捕获发生，但标志位没有及时清除，则 CH1OVI 也会置位。
3. 中断的产生取决于 GP16C2Tn_IER 寄存器中的 CH1I 位。
4. DMA 请求的产生取决于 GP16C2Tn_DMAEN 寄存器中的 CH1DE 位。

为了处理捕获溢出，建议在读取过捕获标志位前先读取捕获数据。这可以避免错过读过捕获标志位之后，读之前产生的捕获数据。

注：捕获中断请求可由软件设置 GP16C2Tn_SGE 寄存器中 SGCHn 位产生。

16. 4. 6. 1 PWM输入模式

测量 I1 上 PWM 信号的周期和占空比的过程如下：

1. 为 GP16C2Tn_CCVAL1 选择有效的输入：GP16C2Tn_CHMR1 寄存器中的 CC1SSEL 位写"01"（I1 被选择）。
2. 为 I1 边沿检测选择有效的极性（用于捕获数据到 GP16C2Tn_CCVAL1 寄存器和计数器清零）：CC1POL 位写'0'（上升沿有效）。
3. 为 GP16C2Tn_CCVAL2 选择有效输入：GP16C2Tn_CHMR1 寄存器的 CC2SEL 位写"10"（I1 被选择）。
4. 为 I1 边沿检测选择有效极性（用于捕获数据到 GP16C2Tn_CCVAL2）：CC2POL 位写'1'。
5. 选择有效的触发输入：GP16C2Tn_SMCON 寄存器的 TSSEL 位写"101"（I1 边沿检测被选择）。
6. 配置从机模式控制器为复位模式：GP16C2Tn_SMCON 寄存器的 SMODS 位写"100"。
7. 使能捕获：GP16C2Tn_CCEP 寄存器的 CC1EN 位和 CC2EN 位写'1'。

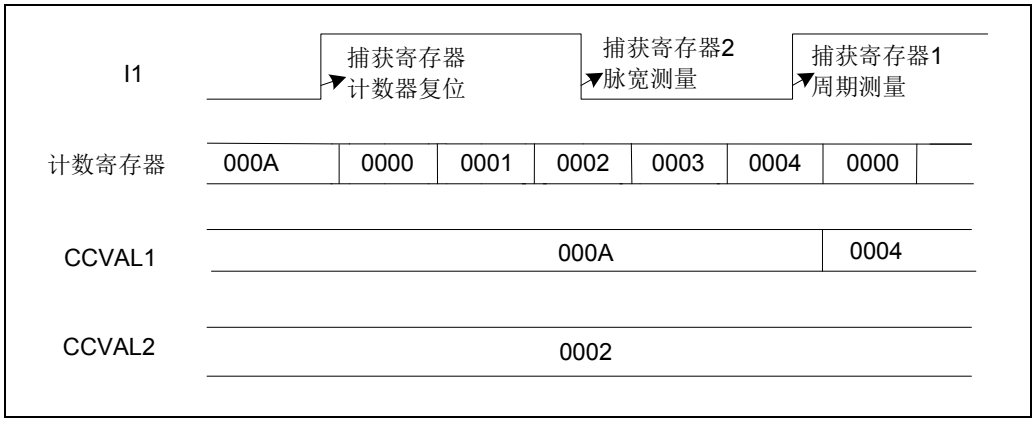


图 17-13 PWM 输入模式时序

16. 4. 7 PWM模式

脉宽调制模式可以产生一个 GP16C2Tn_AR 寄存器值确定频率，GP16C2Tn_CCVALn 寄存器值确定占空比的信号。

每个信道的 PWM 模式是相互独立的（每个 CHn 输出一个 PWM），GP16C2Tn_CHMR1 寄存器的 CHnMOD 位写"110"（PWM 模式 1）或写"111"（PWM 模式 2）。必须通过置位 GP16C2Tn_CHMR1 寄存器的 CHnPEN 位来使能相应的预载寄存器，最后还需置位 GP16C2Tn_CON1 寄存器的 ARPEN 位来使能自动重装预载功能。

只有当更新事件发生时预载寄存器中的值才会传到影子寄存器，因此，在使能计数前，必须通过置位 GP16C2Tn_SGE 寄存器的 SGU 位来初始化所有的寄存器。

CHn 的极性可通过 GP16C2Tn_CCEP 寄存器的 CCnPOL 位配置，有效极性可配置为高或低。CHn 的输出使能由 CCnEN、CCnNE、GOEN、OFFSSI 和 OFFSSR 位（GP16C2Tn_CCEP 和 GP16C2Tn_BDCFG 寄存器）组合控制。

在 PWM 模式（1 或 2）中，GP16C2Tn_COUNT 和 GP16C2Tn_CCVALn 寄存器的值会持续比较，确定 GP16C2Tn_CCVALn<=GP16C2Tn_COUNT 或 GP16C2Tn_CCVALn>= GP16C2Tn_COUNT。

16. 4. 7. 1 PWM边沿对齐模式

- ◇ GP16C2Tn_AR=8
- ◇ PWM 模式 1
- ◇ 递增计数配置

下图给出了 GP16C2Tn_AR = 8 时的边沿对齐 PWM 波形。

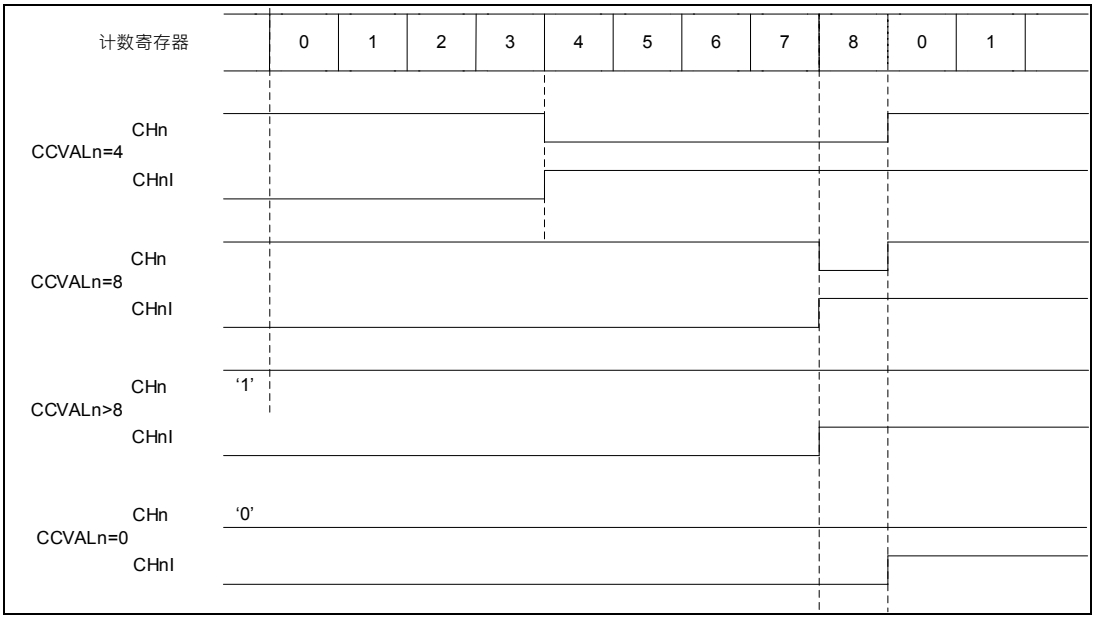


图 17-14 边沿对齐递增计数 PWM 波形（AR=8）

16. 4. 8 输出比较模式

该功能用于控制输出波形或指示周期时间的结束。

当捕获/比较寄存器和计数器值匹配时，输出比较功能：

- ◇ 输出比较模式（GP16C2Tn_CHMR1 寄存器中的 CHnMOD 位）和输出极性（GP16C2Tn_CCEP 寄存器中的 CCnPOL 位）的配置值输出到对应的引脚上。
- ◇ 中断状态寄存器中的标志位置位（GP16C2Tn_RIF 寄存器的 CHnI 位）。
- ◇ 若相应的中断使能置位，则产生中断（GP16C2Tn_IER 寄存器的 CHnI 位）。
- ◇ 若相应的使能位置位（GP16C2Tn_DMAEN 寄存器的 CHnDE 位，GP16C2Tn_CON2 寄存器的 CCDMASEL 位用于 DMA 请求的选择），则发送 DMA 请求。

GP16C2Tn_CHMR1 寄存器中 CHnPEN 位的值可决定 GP16C2Tn_CCVALn 寄存器是否带有预装载寄存器。

在输出比较模式中，更新事件 UEV 对 CHn 的输出没有影响。计时分辨率为计数器的一次计数。输出比较模式同样可以用来输出单个脉冲（单脉冲模式）。

输出比较的配置过程：

1. 选定计数器时钟（内部，外部，预分频）。
2. GP16C2Tn_AR 与 GP16C2Tn_CCVALn 寄存器中写入预期值。
3. 若需要产生中断请求，置位 GP16C2Tn_IER 寄存器中的 CHnI 位。
4. 选择输出模式，例如：
   - CHnMOD = "011"，当 CNTV 与 CCRVALn 匹配时，CHn 输出翻转。
   - CHnPEN = '0'，关闭预载寄存器。
   - CCnPOL = '0'，选择有效极性为高。
   - CCnEN = '1'，使能输出。
5. GP16C2Tn_CON1 寄存器中的 CNTEN 位置位，使能计数器。

假设预载寄存器没有使能（CHnPEN = '0'，否则 GP16C2Tn_CCVALn 影子寄存器只有在下次更新事件发生时才更新）。通过软件方式，GP16C2Tn_CCVALn 寄存器的值可随时更新控制输出波形。

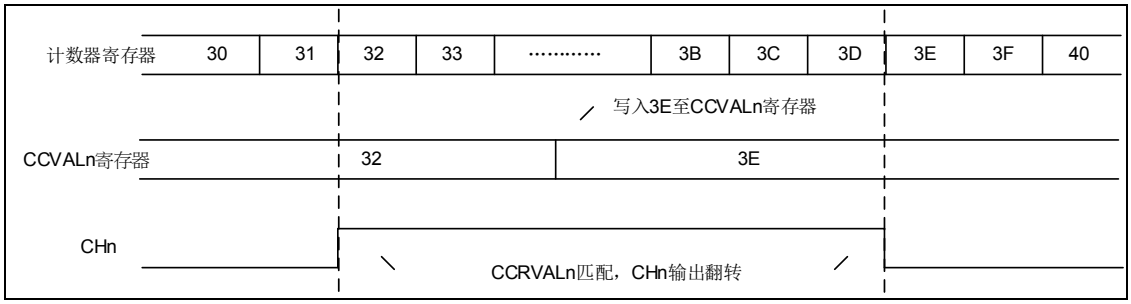


图 17-15　输出比较模式，触发 CHn

16. 4. 8. 1　强制输出模式

在输出模式中（GP16C2Tn_CHMR1 寄存器中 CCnSSEL = "00"），软件可强制将每个输出比较信号（CHnN/CHnN）改为有效或无效状态，这种修改独立于输出比较寄存器和计数器的比较结果。

为了将某输出比较信号（CHn）强制为有效状态，需将相应的 GP16C2Tn_CHMR1 寄存器中 CHnMOD 位写"101"。因此，比较输出被强制为高（高时为有效状态）且 CHn 的值为 CCnPOL 极性位的相反值。

例如：CCnPOL= '0'（CHn 高电平有效），则 CHn 被强制为高电平。

对 GP16C2Tn_CHMR1 寄存器的 CHnMOD 位写"100"，比较输出可被置低。

无论怎样，GP16C2Tn_CCVALn 影子寄存器和计数器之间的比较仍然进行，相应的标志位仍可置位。

16. 4. 9　单脉冲模式

单脉冲模式（SPMEN）下，响应某个触发后，定时器的输出信道在可配置的延迟时间后产生一个脉冲，脉冲长度可配。从模式控制器可控制计数器的启动。脉冲波形可在输出比较模式和 PWM 模式下产生。置位 GP16C2Tn_CON1 寄存器的 SPMEN 位可选择单脉冲

模式。计数器会在下次更新事件 UEV 产生时自动停止。

只有比较值不同于计数器初始值时，单脉冲才可以正确的产生。计数器开始计数前（定时器等待触发），必须如下配置：

◇ 递增计数：CNTV < CCVALn≤ AR（特别地，0 < CCVALn）

基于 PWM 模式设置单脉冲输出波形的步骤如下：

◇ 设置 GP16C2Tn_CHMR1 寄存器的 CHnMOD 位，选择 PWM 模式 1 或 2；

◇ 设置 GP16C2Tn_CCEPn 寄存器的 CCnPOL 位，选择通道端口 CHn 的输出极性；

◇ 设置 GP16C2Tn_CON1 寄存器的 SPMEN 位，配置为递增计数，PWM 普通波形模式，单脉冲模式使能；

◇ 设置 GP16C2Tn_CHMR1 寄存器的 CH1PEN =1，GP16C2Tn_CON1 寄存器的 ARPEN =1，使能比较寄存器和计数重载寄存器的缓冲功能（也可以根据实际情况不使能缓冲）；

◇ 设置 GP16C2Tn_CCVALn 寄存器和 GP16C2Tn_AR 寄存器，配置单脉冲输出延时和脉宽时间；

◇ 设置 GP16C2Tn_SGE 寄存器的 SGU 位来产生一个更新事件；

◇ 设置 GP16C2Tn_CON1 寄存器的 CNTEN=1 来启动计数器，也可以在触发模式下，通过外部触发输入信号来触发硬件自动设置 CNTEN=1。

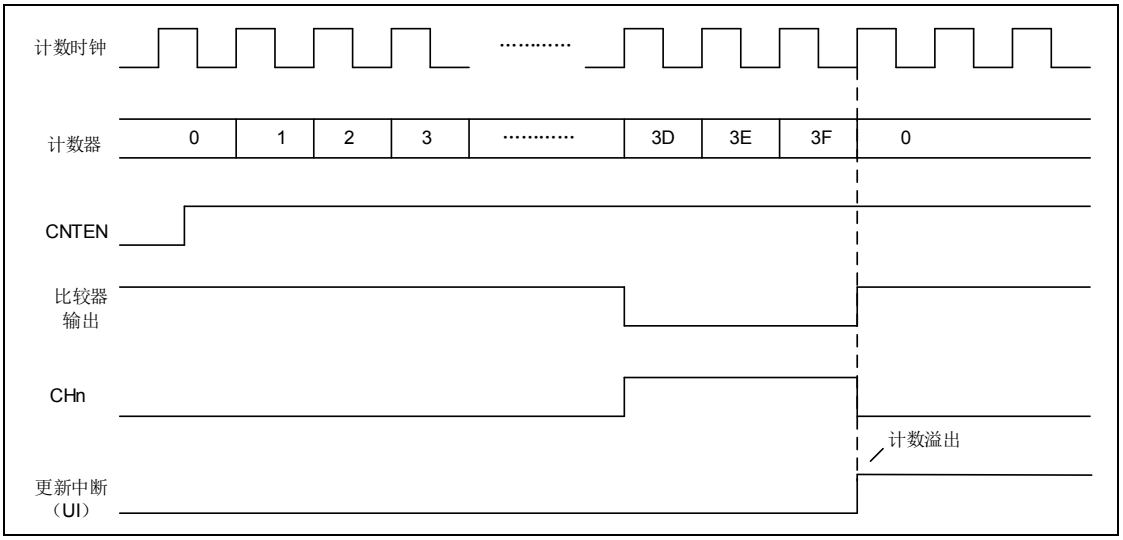


图 17-16　单脉冲模式

16. 4. 10 互补输出与死区时间

两个互补的通道输出信号，可以用来控制输出的瞬时开关，死区时间可配置。

每个输出可独立选择输出极性（主输出 CHn 或互补输出 CHnN），该操作可通过写 GP16C2Tn_CCEP 寄存器的 CCnPOL 和 CCnNPOL 位完成。

互补信号 CHn 和 CHnN 由几个控制位共同控制，分别是 GP16C2Tn_CCEP 寄存器中的 CCnEN 和 CCnNE 位，GP16C2Tn_BDCFG 和 GP16C2Tn_CON2 寄存器中的 GOEN、OISSn、OISSnN、OFFSSI 及 OFFSSR 位，特别是死区时间使能后的空闲状态的切换（GOEN 变为 0）。

置位 CCnEN 和 CCnNE 位，使能死区时间插入，若有刹车电路，同样需要置位 GOEN 位。GP16C2Tn_BDCFG 寄存器的 DT<7:0>可以控制所有通道的死区时间的产生。根据比较输出波形，产生 CHn 和 CHnN 两路输出。若 CHn 和 CHnN 有效电平为高：

- ◇ CHn 的输出信号与参考信号一致。上升沿除外，相对参考信号的上升沿，CHn 输出会有延迟。
- ◇ CHnN 的输出信号与参考信号相反。上升沿除外，相对参考信号的下降沿，CHnN 输出会有延迟。

若延迟时间大于有效输出的宽度（CHn 或 CHnN），则相应的脉冲不会产生。

下图给出了死区时间输出信号和比较输出波形之间的关系。假设 CCnPOL = 0，CCnNPOL = 0，GOEN = 1，CCnEN = 1，和 CCnNE = 1

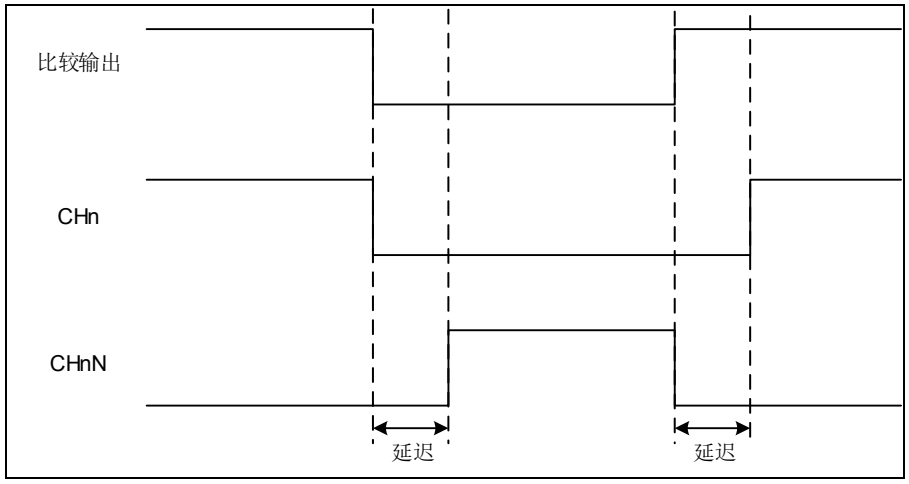


图 17-17　互补输出含死区时间插入

当 PWM 信道配置为互补输出时，如下寄存器控制位都会有缓冲：CHnMOD、CCnEN 和 CCnNE。发生互补通道更新事件时，这些寄存器位才会真正生效，这样就可以预先设置好下一步的配置，并同时对所有互补信道的配置进行更新。互补通道更新事件可以通过设置 GP16C2Tn_SGE 寄存器的 SGCOM=1 产生，或由触发信号产生（由 GP16C2Tn_SMCON 寄存器的 TSSEL 位选择触发信号）。

16. 4. 11 刹车功能

刹车功能模式由以下几个控制位进行设置：GP16C2Tn_BDCFG 寄存器中的 GOEN、OFFSSI 和 OFFSSR 位，GP16C2Tn_CON2 寄存器中的 OISSn 和 OISSnN 位，输出使能信号和无效电平都会被修改。

刹车源可以是刹车输入引脚、时钟失败事件以及软件控制 GP16C2Tn_SGE 寄存器的 SGBRK 位。时钟失败事件由时钟控制器（RCU）中的时钟安全系统（CSS）产生。时钟安全系统（CSS）详细信息可参考时钟安全系统章节。

系统复位后，刹车电路被禁止且 GOEN 位被复位。置位 GP16C2Tn_BDCFG 寄存器的 BRKEN 位可使能刹车功能，同样寄存器中，BRKEN 位可选择刹车输入信号的极性。BRKEN 和 BRKP 位可同时修改。对 BRKEN 和 BRKP 位写操作后，1 个 APB 时钟周期延时后写入值才会生效。因此，写操作后，需等待 1 个 APB 时钟周期后才能正确读回写入值。

由于 GOEN 的下降沿可以是异步的，在实际信号（作用在输出端）和同步控制位（GP16C2Tn_BDCFG 寄存器中）之间插入了一个同步电路。这也导致了异步和同步信号之间会产生一些延迟。特别是 GOEN 之前为低时对 GOEN 写 1 操作后，要读取正确值，必须先插入一个延时（空指令）。这是因为写入的是异步信号，而读取的是同步信号。

当发生刹车请求时（刹车输入端有刹车电平）：

◇ GOEN 位被异步清除，输出端进入无效状态，空闲状态或复位状态（OFFSSI 位选择）。即使 MCU 的振荡器关闭，该功能仍然有效。

◇ 一旦 GOEN=0，每个信道输出预先配置的电平。GP16C2Tn_CON2 寄存器中的 OISSn 位配置该电平。如果 OFFSSI=0，则定时器释放使能输出，否则使能输出一直为高。

◇ 当使用互补输出时：

- 输出端首先被置于复位状态，无效状态（取决于极性）。这个过程异步的，即使定时器没有时钟也有效。
- 如果定时器时钟仍然存在，则死区时间生成器会重新生效，这样在死区时间后，OISSn 和 OISSnN 位的配置电平可驱动输出。这种情况下，CHn 和 CHnN 无法驱动输出端都为有效电平。
- 由于对 GOEN 的重新同步，死区时间的周期会比通常情况下长一些（大约 2 个 TIMER 模块时钟周期）。
- 如果 OFFSSI = 0，则定时器释放使能输出，否则使能输出保持或变高（一旦 CCnEN 和 CCnNE 有一个变高时）。

◇ 当刹车状态标志位（GP16C2Tn_RIF 寄存器中的 BRKI 位）置位时，若 GP16C2Tn_IER 寄存器中的 BRKI 位置位，可触发中断。

◇ 当 GP16C2Tn_BDCFG 寄存器中的 AOEN 位置位时，在下次更新事件（UEV）发生时，GOEN 位会自动置位。例如，该功能可用来整形。否则，GOEN 位会保持为低，直到对其写'1'操作，该特性可用于安全方面的应用，可以将刹车输入端接到一个电源驱动的报警端、热敏传感器或其他安全器件上。

注：刹车输入为电平有效。所以，当刹车输入有效时，不能置位（自动地或者通过软件）GOEN。同时，状态标志 BRKI 不能被清除。

除刹车输入和输出管理，为保证应用程序的安全，内部刹车电路具有写保护功能。用户可冻结几个配置参数（死区时间，CHn/CHnN 极性和失能时状态，CHnMOD 配置，刹车使能和极性）。通过 GP16C2Tn_BDCFG 寄存器中的 LOCKLVL 位，可从三个保护等级中选择一种保护等级。MCU 复位后，LOCKLVL 位只能写一次。

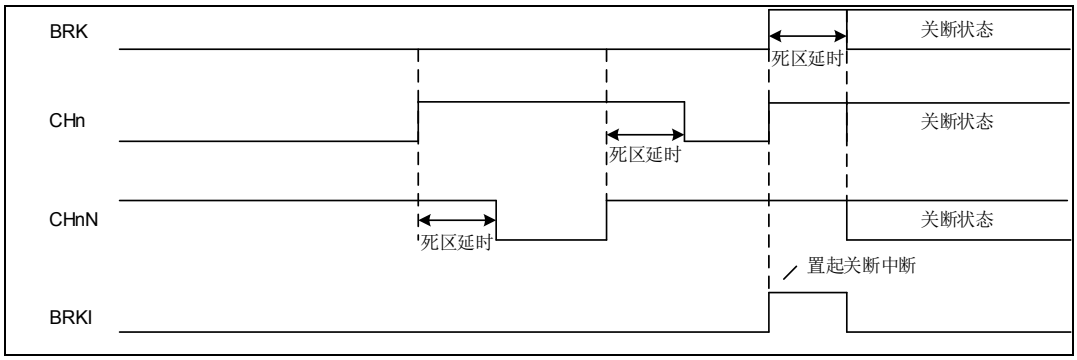


图 17-18　刹车输出行为

16. 4. 12 外部触发的同步

GP16C2Tn 定时器可在多种模式下与外部触发同步：复位模式、门控模式及触发模式。

16. 4. 12. 1　复位模式

计数器及其预分频器可以在响应触发输入事件时重新初始化。此外，若 GP16C2Tn_CON1 寄存器的 UERSEL 位为低时会产生一次更新事件 UEV。所有预载寄存器（GP16C2Tn_AR，GP16C2Tn_CCVALn）都会因更新事件 UEV 而被更新。

在下面例子中，I1 输入端的上升沿让递增计数被清空：

◇　配置信道 1 上检测 I1 上的上升沿。配置输入滤波周期（本例无需滤波器，故 I1FLT = "0000"）。触发捕获分频器没有使用，无需配置。CC1SSEL 位只选择输入捕获源，GP16C2Tn_CHMR1 寄存器中 CC1SSEL = "01"。GP16C2Tn_CCEP 寄存器中 CC1POL = 0 以确定极性（只检测上升沿）。

◇　定时器配置为复位模式：GP16C2Tn_SMCON 寄存器中 SMODS = "100"。选择 I1 作为输入源：GP16C2Tn_SMCON 寄存器中 TSSEL = "101"。

◇　启动计数器：GP16C2Tn_CON1 寄存器中 CNTEN = '1'。

计数器依据内部时钟开始计数，正常计数直到 I1 上出现上升沿。当 I1 上出现上升沿时，计数器会被清零且从 0 重新开始计数。同时，标志位置位（GP16C2Tn_RIF 寄存器中 TRGI 位），如果中断及 DMA 使能（取决于 GP16C2Tn_IER 寄存器中的 TRGI 和 GP16C2Tn_DMAEN 寄存器中的 TRGIDE 位），会发送中断及 DMA 请求。

下图给出了当自动重载寄存器 GP16C2Tn_AR = 0x36 时的信号变化。由于 I1 输入的再同步电路，I1 上的上升沿和计数器实际复位之间会存在延时。

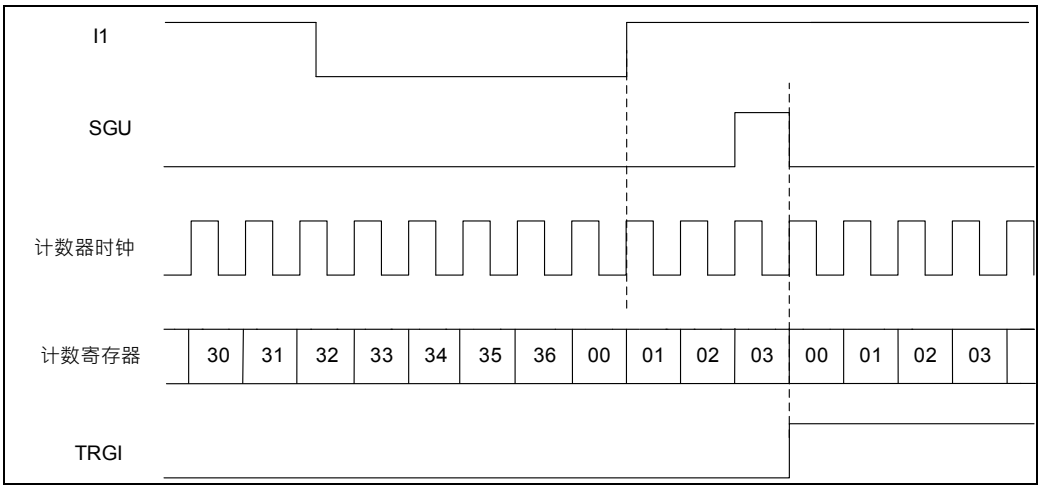


图 17-19　复位模式控制电路

16. 4. 12. 2　门控模式

计数器根据选中的输入电平被使能。

下面的例子中，计数器只在 I1 输入为低电平时才递增计数：

◇　配置信道 1 在 I1 上检测低电平。配置输入滤波周期（本例不需要滤波器，I1FLT = "0000"）。触发捕获分频器没有使用，无需配置。GP16C2Tn_CHMR1 寄存器中的 CC1SSEL = "01"，选择输入捕获源。GP16C2Tn_CCEP 寄存器中 CC1POL = '1'，确认极性（只检测低电平）。

◇　配置定时器为门控模式：GP16C2Tn_SMCON 寄存器中 SMODS = "101"。选择 I1 作为输入源：GP16C2Tn_SMCON 寄存器中 TSSEL = "101"。

◇　使能计数器：GP16C2Tn_CON1 寄存器中 CNTEN = '1'（门控模式中，如果 CNTEN = '0'，无论触发输入为何电平，计数器都不会启动）。

只要 I1 为低电平，计数器依据内部时钟开始计数，一旦 I1 为高则停止计数。由于 I1 输入端的再同步电路的原因，I1 上出现上升沿和计数器实际停止之间会有一定的延时。

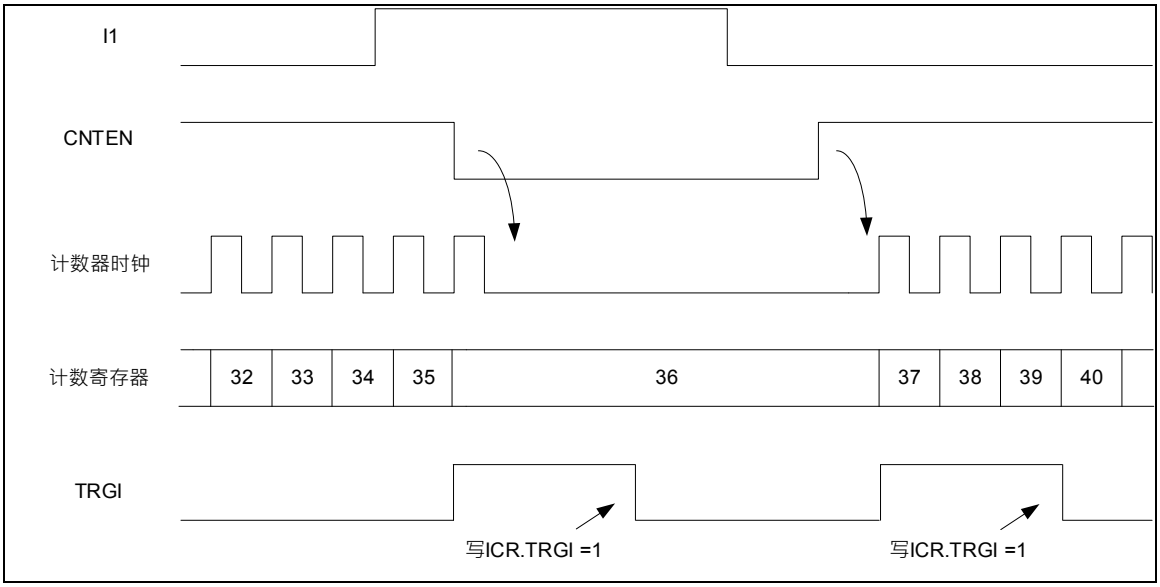


图 17-20　门控模式控制电路

16. 4. 12. 3 触发模式

输入端选中的事件可以使能计数器。

下面的例子中，I2 输入端上的上升沿可以启动递增计数：

◇ 配置信道 2 可以检测 I2 上的上升沿。配置滤波时间（本例不需要滤波，I2FLT = "0000"）。触发捕获分频器没有使用，无需配置。GP16C2Tn_CHMR1 寄存器中 CC2SEL = "01",用于选择捕获源。GP16C2Tn_CCEP 寄存器中 CC2POL = '0'，确认极性（只检测高电平）。

◇ 配置定时器为触发模式：GP16C2Tn_SMCON 寄存器中 SMODS = "110"。GP16C2Tn_SMCON 寄存器中 TSSEL = "110"，用于选择输入源。

I2 上出现上升沿时，计数器开始依据内部时钟计数并置位 TRGI 标志位。

由于 I2 输入的再同步原因，I2 上出现上升沿和计数器实际停止之间会有一定的延时。

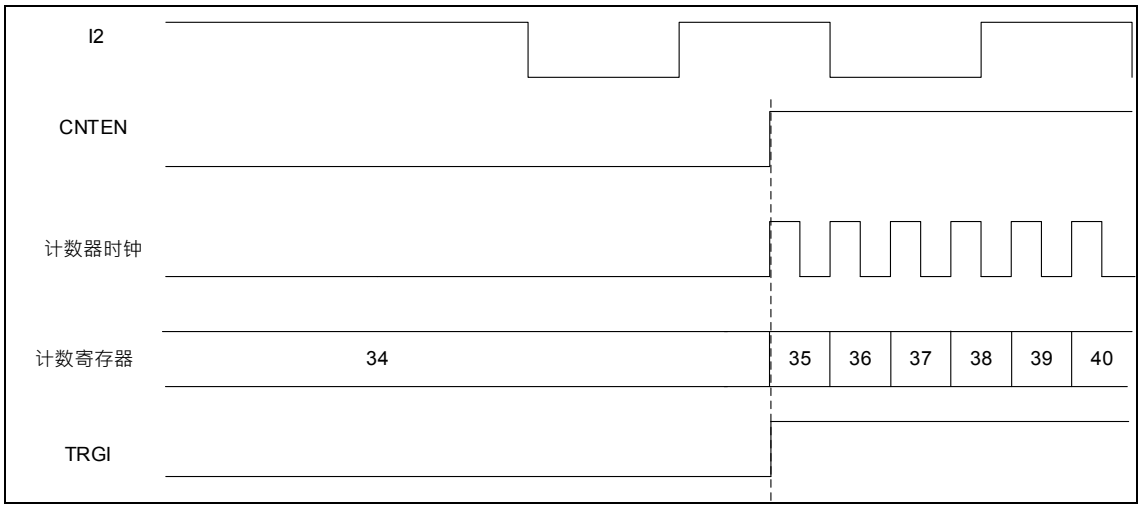


图 17-21 触发模式控制电路

16. 4. 13 定时器同步

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

下图显示了触发选择和主模式选择模块的概况

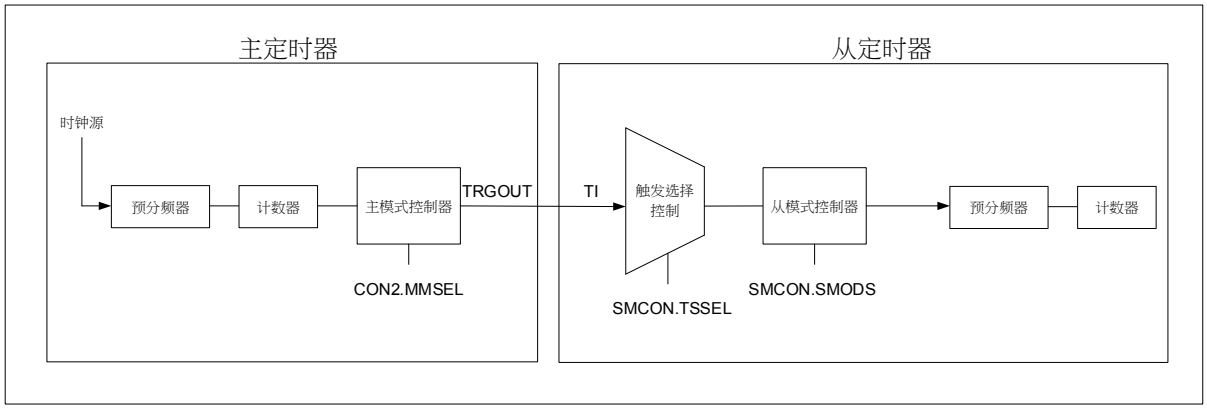


图 17-22 主/从定时器范例

16. 4. 13. 1 使用一个定时器去使能其他定时器

在这个例子中，定时器 2 的使能由定时器 1 的输出比较控制。参考下图的连接。只当定时器 1 的 CH1REF 为高时，定时器 2 才对分频后的内部时钟计数。

配置定时器 1 为主模式，送出它的输出比较参考信号(CH1REF)为触发输出(GP16C2T1_CON2 寄存器的 MMSEL=100)

- ◇ 配置定时器 1 的 CH1REF 波形(GP16C2T1_CHMR1 寄存器)
- ◇ 配置定时器 2 从定时器 1 获得输入触发(GP16C2T2_SMCON 寄存器的 TSSEL=000)
- ◇ 配置定时器 2 为门控模式(GP16C2T2_SMCON 寄存器的 SMODS=101)
- ◇ 配置 GP16C2T2_CON1 寄存器的 CNTEN=1 以使能定时器 2
- ◇ 配置 GP16C2T1_CON1 寄存器的 CNTEN=1 以启动定时器 1

注：定时器 2 的时钟不与定时器 1 的时钟同步，这个模式只影响定时器 2 计数器的使能信号。

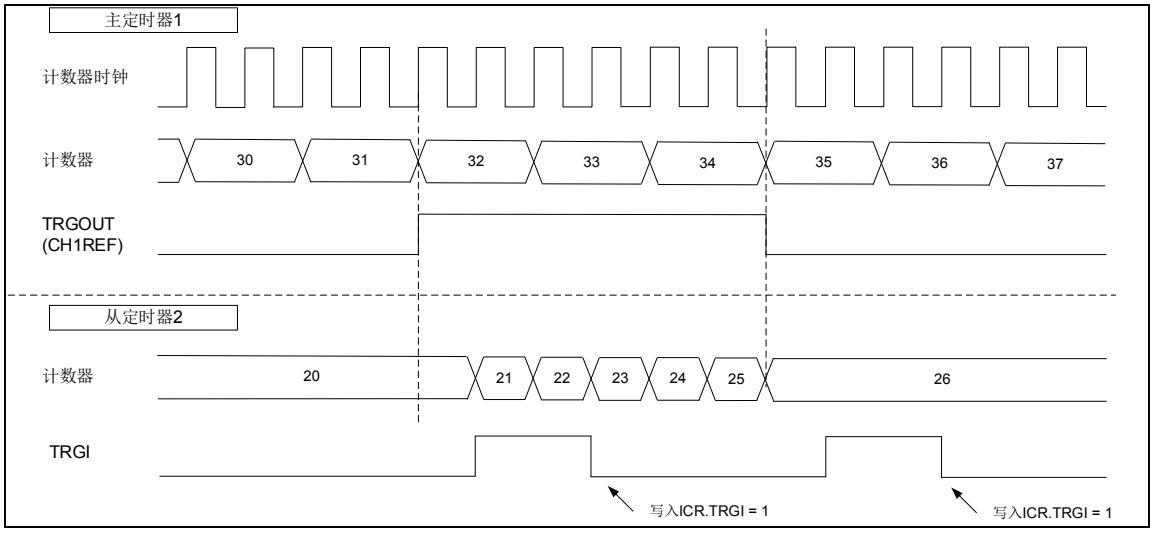


图 17-23 门控从定时器使用主定时器 CH1REF

在上图的例子中，在定时器 2 启动之前，它们的计数器和预分频器未被初始化，因此它们从当前的数值开始计数。可以在启动定时器 1 之前复位 2 个定时器，使它们从给定的数值开始，即在定时器计数器中写入需要的任意数值。写 GP16C2T1_SGE 寄存器的 SGU 位即可复位定时器。

在下一个例子中，需要同步定时器 1 和定时器 2。定时器 1 是主模式并从 0 开始，定时器 2 是从模式并从 0xE7 开始；2 个定时器的预分频器系数相同。写'0'到 GP16C2T1_CON1 的 CNTEN 位将禁止定时器 1，定时器 2 随即停止。

- ◇ 配置定时器 1 为主模式，送出 CNTEN 位做为触发输出(GP16C2T1_CON2 寄存器的 MMSEL=001)。
- ◇ 配置定时器 2 从定时器 1 获得输入触发(GP16C2T2_SMCON 寄存器的 TSSEL=000)
- ◇ 配置定时器 2 为门控模式(GP16C2T2_SMCON 寄存器的 SMODS=101)

◇ 配置 GP16C2T1_SGE 寄存器的 SGU=1，复位定时器 1。

◇ 配置 GP16C2T2_SGE 寄存器的 SGU=1，复位定时器 2。

◇ 写'0xE7'至定时器 2 的计数器(GP16C2T2_COUNT)，初始化它为 0xE7。

◇ 配置 GP16C2T2_CON1 寄存器的 CNTEN=1 以使能定时器 2。

◇ 配置 GP16C2T1_CON1 寄存器的 CNTEN=1 以启动定时器 1。

◇ 配置 GP16C2T1_CON1 寄存器的 CNTEN=0 以停止定时器 1。

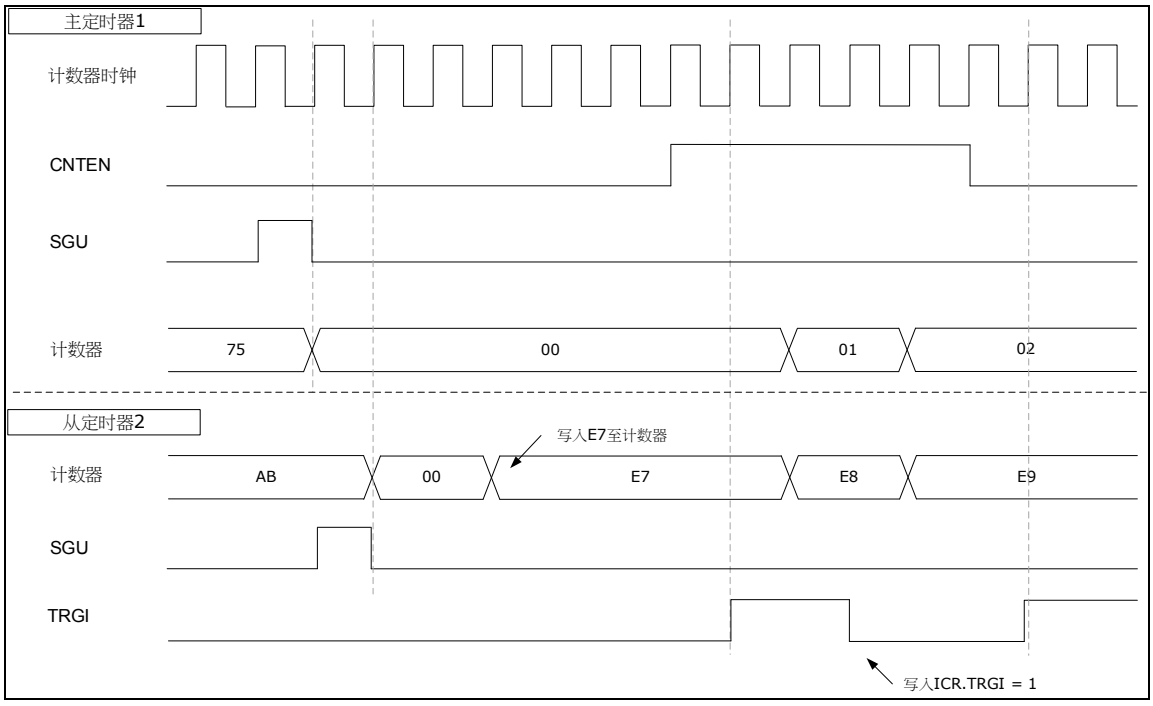


图 17-24　通过使能定时器 1 可以控制定时器 2

16. 4. 13. 2　使用一个定时器去开启其他定时器

在这个例子中，使用定时器 1 的更新事件使能定时器 2。一旦定时器 1 产生更新事件，定时器 2 即从它当前的数值(可以是非 0)按照分频的内部时钟开始计数。在收到触发信号时，定时器 2 的 CNTEN 位被自动地置'1'，同时计数器开始计数直到写'0'到 GP16C2T1_CON1 寄存器的 CNTEN 位。

◇ 配置定时器 1 的周期(GP16C2T1_AR 寄存器)。

◇ 配置定时器 2 从定时器 1 获得输入触发(GP16C2T2_SMCON 寄存器的 TSSEL=000)

◇ 配置定时器 2 为触发模式(GP16C2T2_SMCON 寄存器的 SMODS=110)

◇ 配置 GP16C2T1_CON1 寄存器的 CNTEN=1 以启动定时器 1。

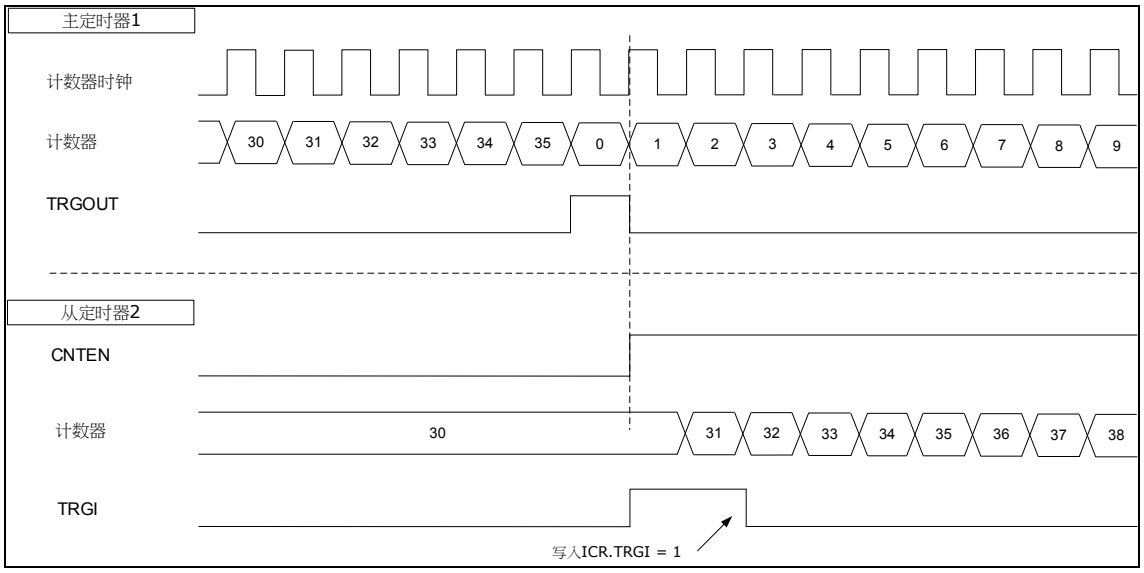


图 17-25　触发中从定时器使用主定时器更新事件

在上图例子中，可以在启动计数之前初始化两个计数器。上图显示在上上图相同配置情况下，使用触发模式而不是门控模式(GP16C2T2_SMCON 寄存器的 SMODS=110)的动作。

16. 4. 13. 3　使用外部触发同步开始两个定时器

这个例子中当定时器 1 的 I1 输入上升时使能定时器 1，使能定时器 1 的同时使能定时器 2，参见下图。为保证计数器的对齐，定时器 1 必须配置为主/从模式(对应 I1 为从，对应定时器 2 为主)：

- ◇　配置定时器 1 为主模式，送出它的使能做为触发输出(GP16C2T1_CON2 寄存器的 MMSEL=001)。
- ◇　配置定时器 1 为从模式，从 TI1 获得输入触发(GP16C2T1_SMCON 寄存器的 TSSEL=’101’)。
- ◇　配置定时器 1 为触发模式(GP16C2T1_SMCON 寄存器的 SMODS=’110’)。
- ◇　配置定时器 1 为主/从模式，GP16C2T1_SMCON 寄存器的 MSCFG=’1’。
- ◇　配置定时器 2 从定时器 1 获得输入触发(GP16C2T2_SMCON 寄存器的 TSSEL=000)
- ◇　配置定时器 2 为触发模式(GP16C2T2_SMCON 寄存器的 SMODS=’110’)。

当定时器 1 的 I1 上出现一个上升沿时，两个定时器同步地按照内部时钟开始计数，两个 TRGI 标志也同时被设置。

注：在这个例子中，在启动之前两个定时器都被初始化(设置相应的 SGU 位)，两个计数器都从 0 开始，但可以通过写入任意一个计数器寄存器(GP16C2T1_COUNT)在定时器间插入一个偏移。下图中能看到主/从模式下在定时器 1 的 CNTEN 和计数器时钟之间有个延迟。

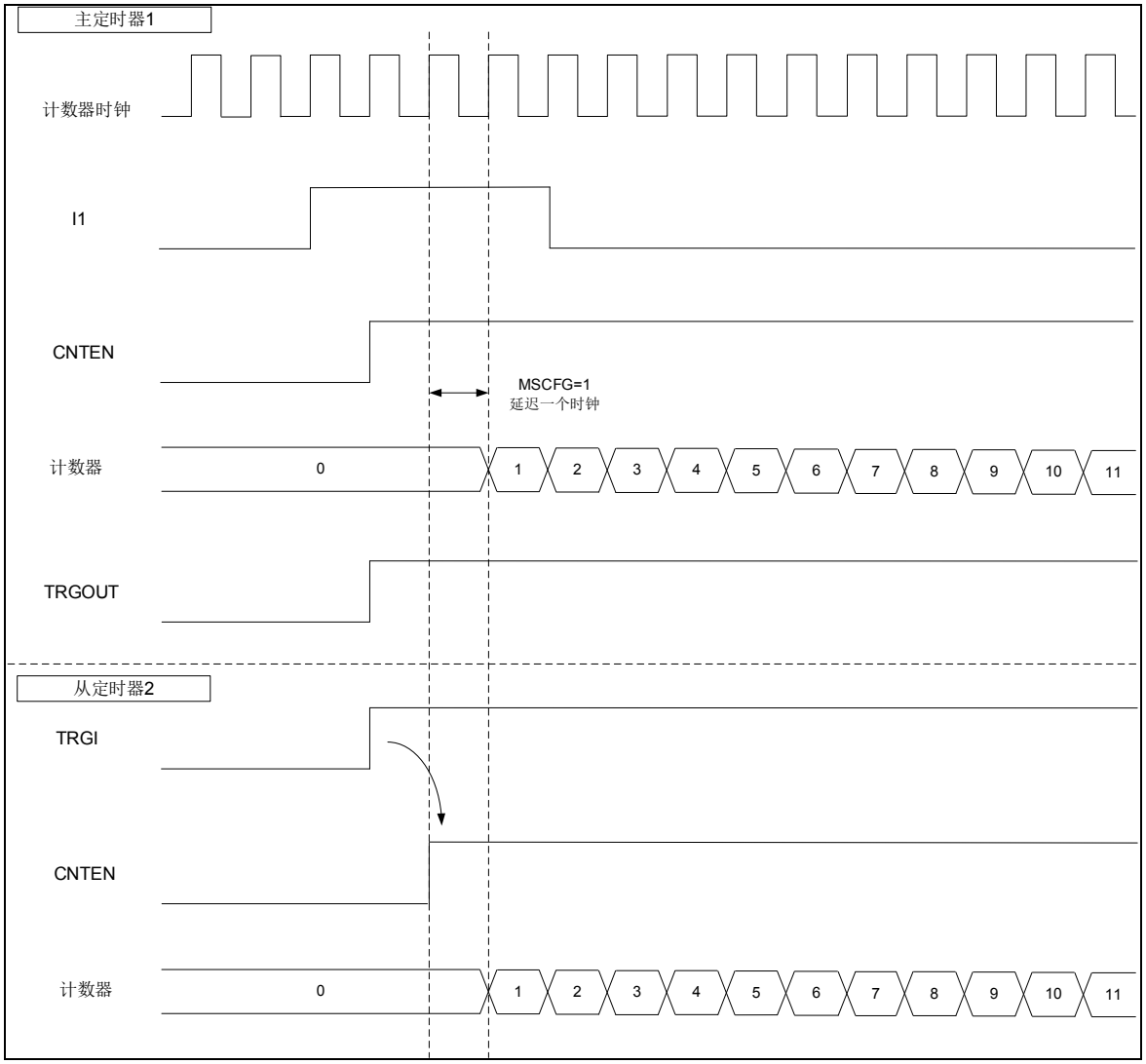


图 17-26　使用定时器 1 的 I1 输入触发定时器 1 和定时器 2

16. 4. 14 调试模式

当微控制器进入调试模式(Cortex™-M0 核停止运行)，计数器停止计数。

16. 5 特殊功能寄存器

16. 5. 1 寄存器列表

GP16C2T0 基地址：4000_0800 H
GP16C2T1 基地址：4000_0C00 H
GP16C2T2 基地址：4000_1000 H

GP16C2Tn 寄存器列表		
名称	偏移地址	描述
GP16C2Tn_CON1	00_H	控制寄存器 1
GP16C2Tn_CON2	04_H	控制寄存器 2
GP16C2Tn_SMCON	08_H	从模式控制寄存器
GP16C2Tn_IER	$0C_H$	中断使能寄存器
GP16C2Tn_IDR	10_H	中断禁止寄存器
GP16C2Tn_IVS	14_H	中断有效状态寄存器
GP16C2Tn_RIF	18_H	原始中断标志寄存器
GP16C2Tn_IFM	$1C_H$	中断屏蔽标志寄存器
GP16C2Tn_ICR	20_H	中断标志清除寄存器
GP16C2Tn_SGE	24_H	软件生成事件寄存器
GP16C2Tn_CHMR1	28_H	捕获/比较模式寄存器 1
GP16C2Tn_CCEP	30_H	捕获/比较使能极性寄存器
GP16C2Tn_COUNT	34_H	计数器
GP16C2Tn_PRES	38_H	时钟预分频寄存器
GP16C2Tn_AR	$3C_H$	自动重装载寄存器
GP16C2Tn_REPAR	40_H	重复计数寄存器
GP16C2Tn_CCVAL1	44_H	通道捕获/比较寄存器 1
GP16C2Tn_CCVAL2	48_H	通道捕获/比较寄存器 2
GP16C2Tn_BDCFG	54_H	刹车和死区配置寄存器
GP16C2Tn_DMAEN	58_H	DMA 事件使能寄存器

16. 5. 2 寄存器描述

16. 5. 2. 1 控制寄存器 1（GP16C2Tn_CON1）

控制寄存器 1（GP16C2Tn_CON1）
偏移地址：00_H
复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																DBGSEL	Reserved					DFCKSEL		ARPEN	Reserved			SPMEN	UERSEL	DISUE	CNTEN

Reserved	Bits 31-16	—	**保留**
DBGSEL	Bit 15	R/W	**调试模式OEN选择** 在输出模式中，选择信道为输入或持续输出 0：通道输入 1：通道持续输出
Reserved	Bits 14-10	—	**保留**
DFCKSEL<1:0>	Bits 9-8	R/W	**时钟分频器** 设置定时器的频率(INT_CLK)，死区产生器与数字滤波器(In)所采样时钟之间的分频比例。 00：tDTS=tINT_CLK 01：tDTS=2*tINT_CLK 10：tDTS=4*tINT_CLK 11：保留
ARPEN	Bit 7	R/W	**自动重装载使能** 发生更新事件时，将设定的值载入至缓冲寄存器中 0：GP16C2Tn_AR 寄存器未缓冲 1：GP16C2Tn_AR 寄存器被装入缓冲器
Reserved	Bits 6-4	—	**保留**
SPMEN	Bit 3	R/W	**单脉冲模式** 0：单脉冲模式禁止，计数器不停止 1：单脉冲模式使能，计数器在发生下一次更新事件时，清除 CNTEN 位，计数器停止
UERSEL	Bit 2	R/W	**更新事件请求来源选择** 设置更新事件(UEV)的来源 0：若使能 UI 中断或 DMA 请求，下列事件中产生更新事件请求 - 计数器溢出 - 设置 SGE 寄存器的 SGU 位 - 从模式中产生的复位请求 1：若使能 UI 中断或 DMA 请求，只在计数器溢出时产生更新事件请求

DISUE	Bit 1	R/W	**更新事件禁止** 设置更新事件(UEV)的产生 0：更新事件(UEV) 使能，下列事件中产生更新事件请求 - 计数器溢出 - 设置 SGE 寄存器的 SGU 位 - 从模式中产生的复位请求 缓冲寄存器载入预装载值 1：更新事件(UEV)禁止，不产生更新事件请求，AR、PRES、CCVALn 寄存器保持数值 禁止更新事件时，设置 SGE 寄存器的 SGU 位或从模式中产生的复位请求，计数器和预分频器仍会被重新初始化
CNTEN	Bit 0	R/W	**计数器使能** 使能计数器后，在外部时钟模式、门控模式和编码模式才能运作。触发模式则可以由硬件设置 CNTEN 位 0：计数器禁止 1：计数器使能

16. 5. 2. 2 控制寄存器 2（GP16C2Tn_CON2）

控制寄存器 2（GP16C2Tn_CON2）

偏移地址：04_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																					OISS2	OISS1N	OISS1	Reserved	MMSEL			CCDMASEL	CCUSEL	Reserved	CCPCEN

Reserved	Bits 31-11	—	**保留**
OISS2	Bit 10	R/W	**通道 2 输出的空闲状态选择位** 参考 OISS1 描述
OISS1N	Bit 9	R/W	**通道 1 互补输出的空闲状态选择位** 0：当 GOEN=0，在一段死区时间后，CH1N=0 1：当 GOEN=0，在一段死区时间后，CH1N=1 注意：当 BDCFG 寄存器中的 LOCKLVL 位被设置为锁定级别 1,2,或 3 后，OISS1N 不可更改。
OISS1	Bit 8	R/W	**通道 1 输出的空闲状态选择位** 0：当 GOEN=0，如果 CH1ON 已实现，在一段死区时间后，CH1=0 1：当 GOEN=0，如果 CH1ON 已实现，在一段死

			区时间后，CH1=1 注意：当 BDCFG 寄存器中的 LOCKLVL 位被设置为锁定级别 1,2,或 3 后，OISS1N 不可更改。
Reserved	Bit 7	—	**保留**
MMSEL<2:0>	Bits 6-4	R/W	**主模式选择** 设置在主模式下发送到从定时器的同步信号(TRGOUT)与 ADC 输入： 000：复位 - 设置 SGE 寄存器信号用于同步触发输出(TRGOUT)。从模式的复位触发产生的复位信号(TRGOUT)则与实际信号相差一个时钟； 001：使能 - 计数器的使能信号 CNTEN 用于同步触发输出(TRGOUT)，可用于同步使能数个定时器。门控模式下，是使用 CON1 寄存器的 CNTEN 位与触发输入信号逻辑产生。当计数器使能信号受控于触发输入时，TRGOUT 上会有一个时钟延迟，可设置 SMCON 寄存器的 MSCFG 位延迟一个时钟同步定时器计数器； 010：更新事件 - 更新事件被用于同步触发输出(TRGOUT)。一个主定时器的更新事件可当作从定时器的预分频器时钟； 011：比较脉冲 - 在发生一次捕获或一次比较成功时，当要设置 CH1I 标志时，触发输出送出一个正脉冲(TRGOUT)； 100：比较信号 - CH1REF 信号用于触发输出(TRGOUT)； 101：比较信号 - CH2REF 信号用于触发输出(TRGOUT)； 110：保留； 111：保留。
CCDMASEL	Bit 3	R/W	**捕获/比较事件的 DMA 选择** 0：当发生 CHn 事件时，设置 CHnDMA 请求 1：当发生更新事件时，设置 CHnDMA 请求
CCUSEL	Bit 2	R/W	**捕获/比较更新控制选择** 此功能只在有互补输出通道作用 0：在捕获/比较预装载时(CCPCEN =1)，只能透过 SGE 寄存器的 SGCOM 位更新 1：在捕获/比较预装载时(CCPCEN =1)，可透过 SGE 寄存器的 SGCOM 与 TI 的上升沿时被更新
Reserved	Bit 1	R/W	**保留**
CCPCEN	Bit 0	R/W	**捕获/比较预装载控制** 设置后只在通信事件(COM)，即 SGE 寄存器的 SGCOM 与 TI 的上升沿时更新 0：CCnEN, CCnNE 和 CHnMOD 位预装载禁止 1：CCnEN, CCnNE 和 CHnMOD 位预装载使能

16. 5. 2. 3 从模式控制寄存器（GP16C2Tn_SMCON）

从模式控制寄存器（GP16C2Tn_SMCON）
偏移地址：08_H
复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bits 31-8	—	**保留**
MSCFG	Bit 7	R/W	**主/从模式** 0：写入0无效 1：延迟触发输入（TI）上的事件来允许当前计时器和其从器件之间的同步。该设置有效用于使用单个外部事件来同步多个计时器。
TSSEL<2:0>	Bits 6-4	R/W	**触发选择** 设置触发选择，用于同步寄数器 000：内部触发 0 (IT0) 001：内部触发 1 (IT1) 010：内部触发 2 (IT2) 011：内部触发 3 (IT3) 100：I1 边沿检测(I1F_ED) 101：I1 滤波后信号 110：I2 滤波后信号 111：外部触发输入
Reserved	Bit 3	—	**保留**
SMODS<2:0>	Bits 2-0	R/W	**从模式选择** 000：从模式关闭 – 设置 CON1 寄存器 CNTEN 位，计数器由内部时钟计数 001：编码器模式 1 – 计数器向上/向下计数 I2 边沿，取决于 I1 电平 010：编码器模式 2 – 计数器向上/向下计数 I1 边沿检测边沿，取决于 I2 边沿检测电平 011：编码器模式 3 – 计数器向上/向下计数 I1 边沿检测和 I2 边沿检测边沿，取决于另一个输入的电平 100：复位模式 – 选中的触发输入(TI)的上升沿重新初始化计数器，并且产生一次更新事件 101：门控模式 – 当触发输入(TI)为高电平时，计数器的时钟开启。一旦触发输入变为低，则计数器

			停止(但不复位)。计数器的启动和停止都是受控 110：触发模式 – 计数器在触发输入 TI 的上升沿启动(但不复位)，只有计数器的启动是受控 111：外部时钟模式 1 – 选中的触发输入(TI) 的上升沿驱动计数器 注：如果 I1 双边沿检测被选为触发输入（TSSEL='100'），不能使用门控模式。I1 每一次转换，I1 双边沿检测就会输出 1 个脉冲，而门控模式则是检查触发信号的电平

16. 5. 2. 4 中断使能寄存器（GP16C2Tn_IER）

中断使能寄存器（GP16C2Tn_IER）

偏移地址：$0C_H$

复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																					CH2OVI	CH1OVI	Reserved	BRKI	TRGI	COMI	Reserved			CH2I	CH1I	UI

Reserved	Bits 31-11	—	**保留**
CH2OVI	Bit 10	W1	**捕获溢出 2 中断使能** 0：写入 0 无效 1：捕获溢出 CH2 中断使能
CH1OVI	Bit 9	W1	**捕获溢出 1 中断使能** 0：写入 0 无效 1：捕获溢出 CH1 中断使能
Reserved	Bit 8	—	**保留**
BRKI	Bit 7	W1	**刹车中断使能** 0：写入 0 无效 1：刹车中断使能
TRGI	Bit 6	W1	**触发中断使能** 0：写入 0 无效 1：触发中断使能
COMI	Bit 5	W1	**通信中断使能** 0：写入 0 无效 1：通信中断使能
Reserved	Bits 4-3	—	**保留**
CH2I	Bit 2	W1	**捕获/比较 2 捕获中断使能** 0：写入 0 无效 1：捕获 CH2 中断使能
CH1I	Bit 1	W1	**捕获/比较 1 捕获中断使能**

			0：写入 0 无效 1：捕获 CH1 中断使能
UI	Bit 0	W1	**更新中断使能** 0：写入 0 无效 1：更新中断使能

16. 5. 2. 5　　中断禁止寄存器（GP16C2Tn_IDR）

中断禁止寄存器（GP16C2Tn_IDR）
偏移地址：10_H
复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																					CH2OVI	CH1OVI	Reserved	BRKI	TRGI	COMI	Reserved		CH2I	CH1I	UI

Reserved	Bits 31-11	—	**保留**
CH2OVI	Bit 10	W1	**捕获溢出 2 中断禁止** 0：写入 0 无效 1：捕获溢出 CH2 中断禁止
CH1OVI	Bit 9	W1	**捕获溢出 1 中断禁止** 0：写入 0 无效 1：捕获溢出 CH1 中断禁止
Reserved	Bit 8	—	**保留**
BRKI	Bit 7	W1	**刹车中断禁止** 0：写入 0 无效 1：刹车中断禁止
TRGI	Bit 6	W1	**触发中断禁止** 0：写入 0 无效 1：触发中断禁止
COMI	Bit 5	W1	**通信中断禁止** 0：写入 0 无效 1：通信中断禁止
Reserved	Bits 4-3	—	**保留**
CH2I	Bit 2	W1	**捕获/比较 2 捕获中断禁止** 0：写入 0 无效 1：捕获 CH2 中断禁止
CH1I	Bit 1	W1	**捕获/比较 1 捕获中断禁止** 0：写入 0 无效 1：捕获 CH1 中断禁止
UI	Bit 0	W1	**更新中断禁止** 0：写入 0 无效

			1：更新中断禁止

16. 5. 2. 6　　中断有效状态寄存器（GP16C2Tn_IVS）

中断有效状态寄存器（GP16C2Tn_IVS）
偏移地址：14_H
复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																					CH2OVI	CH1OVI	Reserved	BRKI	TRGI	COMI	Reserved		CH2I	CH1I	UI

Reserved	Bits 31-11	—	**保留**
CH2OVI	Bit 10	R	**捕获溢出 2 中断有效位** 设置 IER 和 IDR 寄存器使能或禁止 0：捕获溢出 CH2 中断禁止 1：捕获溢出 CH2 中断使能
CH1OVI	Bit 9	R	**捕获溢出 1 中断有效位** 设置 IER 和 IDR 寄存器使能或禁止 0：捕获溢出 CH1 中断禁止 1：捕获溢出 CH1 中断使能
Reserved	Bit 8	—	**保留**
BRKI	Bit 7	R	**刹车中断有效位** 设置 IER 和 IDR 寄存器使能或禁止 0：刹车中断禁止 1：刹车中断使能
TRGI	Bit 6	R	**触发中断有效位** 设置 IER 和 IDR 寄存器使能或禁止 0：触发中断禁止 1：触发中断使能
COMI	Bit 5	R	**通信中断有效位** 设置 IER 和 IDR 寄存器使能或禁止 0：通信中断禁止 1：通信中断使能
Reserved	Bits 4-3	—	**保留**
CH2I	Bit 2	R	**捕获/比较 2 捕获中断有效位** 设置 IER 和 IDR 寄存器使能或禁止 0：捕获 CH2 中断禁止 1：捕获 CH2 中断使能
CH1I	Bit 1	R	**捕获/比较 1 捕获中断有效位** 设置 IER 和 IDR 寄存器使能或禁止 0：捕获 CH1 中断禁止

			1：捕获 CH1 中断使能
UI	Bit 0	R	**更新中断有效位** 设置 IER 和 IDR 寄存器使能或禁止 0：更新中断禁止 1：更新中断使能

16. 5. 2. 7 原始中断标志寄存器（GP16C2Tn_RIF）

原始中断标志寄存器（GP16C2Tn_RIF）
偏移地址：18_H
复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																					CH2OVI	CH1OVI	Reserved	BRKI	TRGI	COMI	Reserved			CH2I	CH1I	UI

Reserved	Bits 31-11	—	**保留**
CH2OVI	Bit 10	R	**捕获溢出2原始中断标志** 参照CH1OVI描述 0：无中断产生 1：捕获溢出 CH2 原始中断产生
CH1OVI	Bit 9	R	**捕获溢出1原始中断标志** 当CH1I已经被设置又再次收到捕获信号将计数器数值写入CCVAL1寄存器中。此位由硬件设置1，设置ICR寄存器清除 0：无中断产生 1：捕获溢出 CH1 原始中断产生
Reserved	Bit 8	—	**保留**
BRKI	Bit 7	R	**刹车原始中断标志** 产生刹车事件时产生。此位由硬件设置 1，设置 ICR 寄存器清除 0：无中断产生 1：刹车原始中断产生
TRGI	Bit 6	R	**触发原始中断标志** 有触发事件时产生(在从模式-门控时，在任一沿产生，其他模式则在有效沿时产生)。此位由硬件设置 1，设置 ICR 寄存器清除 0：无中断产生 1：触发原始中断产生
COMI	Bit 5	R	**通信原始中断标志** 有通信事件时产生(在捕获/比较预装载控制使能时，更新 CCnEN, CCnNE, CHnMOD 位)。此位由

			硬件设置 1，设置 ICR 寄存器清除 0：无中断产生 1：通信原始中断产生
Reserved	Bits 4-3	—	**保留**
CH2I	Bit 2	R	**捕获/比较 2 原始中断标志** 参照 CH1I 描述 0：无中断产生 1：捕获/比较 2 原始中断产生
CH1I	Bit 1	R	**捕获/比较 1 原始中断标志** **通道 CH1 设置为输出:** 计数器匹配 CCVAL1 寄存器时设置，当 CCVAL1 寄存器大于 AR 寄存器时，则在递增边界时置起。此位由硬件设置 1，设置 ICR 寄存器清除 0：无中断产生 1：捕获/比较 1 原始中断产生 **通道 CH1 设置为输入:** 发生捕获事件，将计数器捕获至 CCVAL1 寄存器中，于 I1 的有效沿产生。此位由硬件设置 1，设置 ICR 寄存器清除 0：无中断产生 1：捕获/比较 1 原始中断产生
UI	Bit 0	R	**更新原始中断标志** 当 CON1 寄存器的 UERSEL、DISUE 为 0，以下三件事件触发中断 - 当重复计数器数值上溢时 (重复计数器为 0 时产生更新事件)。 - 设置 SGE 寄存器的 SGU 位时产生更新事件，通过软件对计数器重新初始化时。 - 当计数器被触发事件重新初始化时 0：无中断产生 1：更新原始中断产生

16. 5. 2. 8 中断屏蔽标志寄存器（GP16C2Tn_IFM）

中断屏蔽标志寄存器（GP16C2Tn_IFM）
偏移地址：$1C_H$
复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																					CH2OVI	CH1OVI	Reserved	BRKI	TRGI	COMI	Reserved			CH2I	CH1I	UI

Reserved	Bits 31-11	—	**保留**
CH2OVI	Bit 10	R	**捕获溢出2中断标志** 参照CH1OVI描述 0：无中断产生 1：捕获溢出 CH2 中断屏蔽标志产生
CH1OVI	Bit 9	R	**捕获溢出1中断屏蔽标志** 当IVS寄存器为1时，CH1I已经被设置又再次收到捕获信号将计数器数值写入CCVAL1寄存器中。此位由硬件设置1，设置ICR寄存器清除 0：无中断产生 1：捕获溢出 CH1 中断屏蔽标志产生
Reserved	Bit 8	—	**保留**
BRKI	Bit 7	R	**刹车中断屏蔽标志** 当 IVS 寄存器为 1 时，有刹车事件时产生。此位由硬件设置 1，设置 ICR 寄存器清除 0：无中断产生 1：刹车中断屏蔽标志产生
TRGI	Bit 6	R	**触发中断屏蔽标志** 当 IVS 寄存器为 1 时，有触发事件时产生(在从模式-门控时，在任一沿产生，其他模式则在有效沿时产生)。此位由硬件设置 1，设置 ICR 寄存器清除 0：无中断产生 1：触发中断产生
COMI	Bit 5	R	**通信中断屏蔽标志** 当 IVS 寄存器为 1 时，产生通信事件时产生(在捕获/比较预装载控制使能时，更新 CCnEN, CCnNE 和 CH1MOD 位)。此位由硬件设置 1，设置 ICR 寄存器清除 0：无中断产生 1：通信中断屏蔽标志产生
Reserved	Bits 4-3	—	**保留**
CH2I	Bit 2	R	**捕获/比较 2 中断屏蔽标志** 参照 CH1I 描述 0：无中断产生 1：捕获/比较 2 中断屏蔽标志产生
CH1I	Bit 1	R	**捕获/比较 1 中断屏蔽标志** 当 IVS 寄存器为 1 时 **通道 CH1 设置为输出:** 计数器匹配 CCVAL1 寄存器时设置，当 CCVAL1 寄存器大于 AR 寄存器时，则在递增边界时置起。此位由硬件设置 1，设置 ICR 寄存器清除 0：无中断产生 1：捕获/比较 1 中断屏蔽标志产生

			通道 CH1 设置为输入: 发生捕获事件，将计数器捕获至 CCVAL1 寄存器中，于 I1 的有效沿产生。此位由硬件设置 1，设置 ICR 寄存器清除 0：无中断产生 1：捕获/比较 1 中断屏蔽标志产生
UI	Bit 0	R	**更新中断屏蔽标志** 当 CON1 寄存器的 UERSEL、DISUE 为 0，以下三件事件触发中断 - 当重复计数器数值上溢时 (重复计数器为 0 时产生更新事件)。 - 设置 SGE 寄存器的 SGU 位时产生更新事件，通过软件对计数器重新初始化时。 - 当计数器被触发事件重新初始化时 0：无中断产生 1：更新中断屏蔽标志产生

16. 5. 2. 9 中断标志清除寄存器（GP16C2Tn_ICR）

中断标志清除寄存器（GP16C2Tn_ICR）
偏移地址：20_H
复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																					CH2OVI	CH1OVI	Reserved	BRKI	TRGI	COMI	Reserved			CH2I	CH1I	UI

Reserved	Bits 31-11	—	**保留**
CH2OVI	Bit 10	C_W1	**捕获溢出 2 中断标志清除** 0：写入0无效 1：捕获溢出 CH2 中断标志清除
CH1OVI	Bit 9	C_W1	**捕获溢出 1 中断标志清除** 0：写入0无效 1：捕获溢出 CH1 中断标志清除
Reserved	Bit 8	—	**保留**
BRKI	Bit 7	C_W1	**刹车中断标志清除** 0：写入 0 无效 1：刹车中断标志清除
TRGI	Bit 6	C_W1	**触发中断标志清除** 0：写入 0 无效 1：触发中断标志清除
COMI	Bit 5	C_W1	**通信中断标志清除**

			0：写入 0 无效 1：通信中断标志清除
Reserved	Bits 4-3	—	**保留**
CH2I	Bit 2	C_W1	**捕获/比较 2 中断标志清除** 0：写入 0 无效 1：捕获 CH2 中断标志清除
CH1I	Bit 1	C_W1	**捕获/比较 1 中断标志清除** 0：写入 0 无效 1：捕获 CH1 中断标志清除
UI	Bit 0	C_W1	**更新中断标志清除** 0：写入 0 无效 1：更新中断标志清除

16. 5. 2. 10　软件生成事件生寄存器（GP16C2Tn_SGE）

软件生成事件寄存器（GP16C2Tn_SGE）

偏移地址：24_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																								SGBRK	SGTRG	SGCOM	Reserved		SGCH2	SGCH1	SGU

Reserved	Bits 31-8	—	**保留**
SGBRK	Bit 7	W1	**软件生成刹车事件** 该位由软件设置来生成刹车事件，可由硬件自动清零。 0：无动作 1：产生刹车事件。GOEN清零，BRKI标志位置起，产生相关中断或DMA传输。
SGTRG	Bit 6	W1	**软件生成触发事件** 该位由软件设置来生成触发事件，可由硬件自动清零。 0：无动作 1： RIF 寄存器中的 TRGI 被置起，产生相关中断或 DMA 传输
SGCOM	Bit 5	W1	**软件生成通信事件捕获** 该位由软件设置来生成通信事件，可由硬件自动清零。 0：无动作 1：当 CCPCEN 被置 1，则可更新 CCnEN, CCnNE 和 CHnMOD

			注意：该位只有用作于通道时才有互补输出
Reserved	Bits 4-3	—	**保留**
SGCH2	Bit 2	W1	**软件生成通道 2 捕获/比较事件** 参考 SGCH1 描述
SGCH1	Bit 1	W1	**软件生成通道 1 捕获/比较事件** **通道 CH1 设置为输出:** 产生捕获/比较但不影响输出，若使能中断或 DMA，则产生中断与请求。由软件设置，于下一个时钟自动清除 0：写入 0 无效 1：捕获/比较 1 中断产生 **通道 CH1 设置为输入:** 发生捕获事件，将计数器捕获至 CCVAL1 寄存器中，于 I1 的有效沿产生，若使能中断或 DMA，则产生中断与请求。由软件设置，于下一个时钟自动清除 0：写入 0 无效 1：捕获/比较 1 中断产生
SGU	Bit 0	W1	**软件触发更新事件** 该位由软件设置，可由硬件自动清零。 0：无动作 1：重新初始化计数器，更新寄存器。注意，预分频器也会被清零（但预分频比不会受到影响）。

16. 5. 2. 11 捕获/比较模式寄存器 1（GP16C2Tn_CHMR1）

捕获/比较模式寄存器 1（GP16C2Tn_CHMR1）

偏移地址：28_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																	CH2MOD			CH2PEN	CH2FEN	CC2SSEL		Reserved	CH1MOD			CH1PEN	CH1FEN	CC1SSEL	
Reserved																	I2FLT			I2PRES				I1FLT				I1PRES			

输出比较模式

Reserved	Bits 31-15	—	**保留**
CH2MOD<2:0>	Bits 14-12	R/W	**输出比较信道 2 模式** 参考 CH1MOD 描述

CH2PEN	Bit 11	R/W	**输出比较通道 2 预装载使能** 参考 CH1PEN 描述
CH2FEN	Bit 10	R/W	**输出比较通道 2 快速使能** 参考 CH1FEN 描述
CC2SSEL<1:0>	Bits 9-8	R/W	**捕获/比较通道 2 选择** 设置通道的输出方向与信号的选择，当 CCEP 寄存器的 CC2EN 为 0 才可写入。 00：通道设置为输出 01：通道设置为输入，捕获源为 **I2** 10：通道设置为输入，捕获源为 **I1** 11：通道设置为输入，捕获源为 **ITn 或 I1** 的双边沿检测
Reserved	Bits 7	—	**保留**
CH1MOD<2:0>	Bits 6-4	R/W	**输出比较信道 1 模式** 设置 CH1REF 输出模式，CH1 与 CH1N 由 CH1REF 产生，而有效位由 CCEP 寄存器的 CC1POL 与 CC1NPOL 位设置 000：禁止 - 无作用 001：匹配时设置高电平 - 当计数器匹配 CCVAL1 寄存器时，CH1REF 设置为 1 010：匹配时设置低电平 - 当计数器匹配 CCVAL1 寄存器时，CH1REF 设置为 0 011：匹配时设置翻转 - 当计数器匹配 CCVAL1 寄存器时，CH1REF 设置翻转(当前高/低电平翻转成低/高电平) 100：强制低电平 - CH1REF 强制设置低电平 101：强制高电平 - CH1REF 强制设置高电平 110：PWM 模式 1 - 递增模式时，当计数器小于 CCVAL1 寄存器时，输出高电平，其他则输出低电平。递减模式，当计数器大于 CCVAL1 寄存器时输出低电平，其他则输出高电平 111：PWM 模式 2 - 递增模式时，当计数器小于 CCVAL1 寄存器时，输出低电平，其他则输出高电平。递减模式时，当计数器大于 CCVAL1 寄存器时输出高电平，其他则输出低电平
CH1PEN	Bit 3	R/W	**输出比较通道 1 预装载使能** 设置后在更新事件时，将设置的寄存器 CCVAL1 数值载入预装载 CCVAL1 寄存器中 0：CCVAL1 寄存器预装载禁止 1：CCVAL1 寄存器预装载使能
CH1FEN	Bit 2	R/W	**输出比较通道 1 快速使能** 用于加速事件的产生 0：CH1 的正常操作依赖于计数器与 CCVAL1 的值，即使工作于触发器状态。当触发器的输入有一

			个有效沿时，激活 CH1 输出的最小延时为 5 个时钟周期。 1：输入到触发器的有效沿的作用就像发生了一次比较匹配。因此，CH1 被设置为比较电平而与比较结果无关。采样触发器的有效沿和 CH1 输出间的延时被缩短为 3 个时钟周期。CH1FEN 只在信道被配置成 PWM1 或 PWM2 模式时起作用
CC1SSEL<1:0>	Bits 1-0	R/W	**捕获/比较通道 1 选择** 设置通道的输出方向与信号的选择，当 CCEP 寄存器的 CC1EN 为 0 才可写入。 00：通道设置为输出 01：通道设置为输入，捕获源为 **I1** 10：通道设置为输入，捕获源为 **I2** 11：通道设置为输入，捕获源为 **ITn 或 I1** 的双边沿检测

输入捕获模式

Reserved	Bits 31-16	—	**保留**
I2FLT<3:0>	Bits 15-12	R/W	**输入捕获通道2滤波器** 参照I1FLT描述
I2PRES<1:0>	Bits 11-10	R/W	**输入捕获通道 2 预分频器** 参照 IC1PRES 描述
CC2SSEL<1:0>	Bits 9-8	R/W	**捕获/比较通道 2 选择** 设置通道的输出方向与信号的选择，当 CCEP 寄存器的 CC2EN 为 0 才可写入。 00：通道设置为输出 01：通道设置为输入，捕获源为 **I2** 10：通道设置为输入，捕获源为 **I1** 11：通道设置为输入，捕获源为 **ITn 或 I1** 的双边沿检测
I1FLT<3:0>	Bits 7-4	R/W	**输入捕获通道 1 滤波器** 设置 I1 信号采样的频率和数字滤波的带宽。数字滤波器是一个事件计数器，它记录到 N 个事件后会产生一个输出的跳变 0000：采样频率 fDTS，滤波器禁止 0001：采样频率 fINT_CLK, N = 2 0010：采样频率 fINT_CLK, N = 4 0011：采样频率 fINT_CLK, N = 8 0100：采样频率 fDTS / 2, N = 6 0101：采样频率 fDTS / 2, N = 8 0110：采样频率 fDTS / 4, N = 6 0111：采样频率 fDTS / 4, N = 8 1000：采样频率 fDTS / 8, N = 6 1001：采样频率 fDTS / 8, N = 8 1010：采样频率 fDTS / 16, N = 5 1011：采样频率 fDTS / 16, N = 6 1100：采样频率 fDTS / 16, N = 8 1101：采样频率 fDTS / 32, N = 5 1110：采样频率 fDTS / 32, N = 6 1111：采样频率 fDTS / 32, N = 8
I1PRES<1:0>	Bits 3-2	R/W	**输入捕获通道 1 预分频器** 设置 I1 的预分频计数器数值，当清除 CCEP 寄存器的 CC1EN 位，预分频计数器同时被清除 00：预分频禁止，于每次事件时捕获 01：每 2 次事件捕获 10：每 4 次事件捕获 11：每 8 次事件捕获
CC1SSEL<1:0>	Bits 1-0	R/W	**捕获/比较通道 1 选择** 设置通道的输出方向与信号的选择，当 CCEP 寄

			存器的 CC1EN 为 0 才可写入。 00：通道设置为输出 01：通道设置为输入，捕获源为 **I1** 10：通道设置为输入，捕获源为 **I2** 11：通道设置为输入，捕获源为 **ITn 或 I1** 的双边沿检测

16. 5. 2. 12　捕获/比较使能极性寄存器（GP16C2Tn_CCEP）

捕获/比较使能寄存器（GP16C2Tn_CCEP）
偏移地址：30_H
复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																								CC2NPOL	Reserved	CC2POL	CC2EN	CC1NPOL	CC1NE	CC1POL	CC1EN

Reserved	Bits 31-8	—	**保留**
CC2NPOL	Bit 7	R/W	**捕获/比较通道 2 互补输出有效位极性** 参照 CC1NPOL 描述
Reserved	Bit 6	—	**保留**
CC2POL	Bit 5	R/W	**捕获/比较通道 2 输出有效位极性** 参照 CC1POL 描述
CC2EN	Bit 4	R/W	**捕获/比较通道 2 输出使能** 参照 CC1EN 描述
CC1NPOL	Bit 3	R/W	**捕获/比较通道 2 互补输出有效位极性** **通道 CH1 设置为输出:** 0：CH1N 高电平有效 1：CH1N 低电平有效 **通道 CH1 设置为输入:** 该位需和 CC1POL 一起使用来定义输入边沿的极性。参考 CC1POL 描述。 注：对于有互补输出的通道，该位设置为预载值。如果 GP16C2Tn_CON2 寄存器中的 CCPCEN 位设置为 1，则只有当 COM 事件发生时，CC1NP 有效位才会设置为预载值中新的值。 注意：当 GP16C2Tn_BDCFG 寄存器中的 LOCKLVL 位被设置为锁定级别 2 或 3，且 CC1SSEL=00（信道为输出模式），该位将不可写。
CC1NE	Bit 2	R/W	**捕获/比较通道 1 互补输出使能** 0：关闭 - CH1N 无效。CH1N 电平取决于 GOEN, OFFSSI, OFFSSR, OISS1, OISS1N 和 CC1EN 的

			功能 1：开启 - CH1N 为对应输出引脚上的输出信号，由 GOEN, OFFSSI, OFFSSR, OISS1, OISS1N 和 CC1EN 决定。 注：对于有互补输出的通道，该位设置为预载值。如果 GP16C2Tn_CON2 寄存器中的 CCPCEN 位设置为 1，则只有当 COM 事件发生时，CC1NE 有效位才会设置为预载值中新的值
CC1POL	Bit 1	R/W	**捕获/比较通道 1 输出有效位极性** **通道 CH1 设置为输出:** 0：CH1 高电平有效 1：CH1 低电平有效 **通道 CC1 设置为输入:** CC1NPOL/CC1POL 位选择触发边沿或捕获模式下 I1 和 I2 的极性 00：非反相/上升沿 01：反相/下降沿 10：保留 11：非反相/上升沿+下降沿 注：对于有互补输出的通道，该位设置为预载值。如果 GP16C2Tn_CON2 寄存器中的 CCPCEN 位设置为 1，则只有当 COM 事件发生时，CC1POL 有效位才会设置为预载值中新的值。 注：当 GP16C2Tn_BDCFG 寄存器中的 LOCKLVL 位被设置为锁定级别 2 或 3，且 CC1SSEL=00（信道为输出模式），该位将不可写。
CC1EN	Bit 0	R/W	**捕获/比较通道 1 输出使能** **通道 CH1 设置为输出:** 0：关闭 - CH1 无效。CH1 电平取决于 GOEN, OFFSSI, OFFSSR, OISS1, OISS1N 和 CC1NE 的功能 1：开启 - CH1 为对应输出引脚上的输出信号，由 GOEN, OFFSSI, OFFSSR, OISS1, OISS1N 和 CC1NE 决定 **通道 CH1 设置为输入:** 0：捕获禁止 1：捕获使能 注：对于有互补输出的通道，该位设置为预载值。如果 GP16C2Tn_CON2 寄存器中的 CCPCEN 位设置为 1，则只有当 COM 事件发生时，CC1EN 有效位才会设置为预载值中新的值。

16. 5. 2. 13 计数寄存器（GP16C2Tn_COUNT）

计数寄存器（GP16C2Tn_COUNT）
偏移地址：34_H
复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bits 31-16	—	保留
CNTV<15:0>	Bits 15-0	R/W	计数器数值

16. 5. 2. 14 时钟预分频寄存器（GP16C2Tn_PRES）

时钟预分频寄存器（GP16C2Tn_PRES）
偏移地址：38_H
复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bits 31-16	—	保留
PSCV<15:0>	Bits 15-0	R/W	预分频数值 当计数器时钟频率等于fINT_CLK/(PSCV<15:0> + 1)时计数器递增。在更新事件产生时，将PSCV数值被载入预装载寄存器中

16. 5. 2. 15 自动重装载寄存器（GP16C2Tn_AR）

自动重装载寄存器（GP16C2Tn_AR）
偏移地址：$3C_H$
复位值：$00000000\_00000000\_11111111\_11111111_B$

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

Reserved	Bits 31-16	—	保留
ARV<15:0>	Bits 15-0	R/W	**自动装载数值** 设置计数器的递增边界，设置数值为 0 时计数器停止计数

16. 5. 2. 16 重复计数寄存器（GP16C2Tn_REPAR）

重复计数寄存器（GP16C2Tn_REPAR）
偏移地址：40_H
复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bits 31-8	—	保留
REPV<7:0>	Bits 7-0	R/W	**重复计数数值** 当预载寄存器使能，该位允许用户设置比较寄存器的更新率（例如：预载到有效寄存器的周期性传输），同样也可以设置更新中断生成率。每次当 REPV_CNT 的相关递减计数器递减至 0，会产生更新事件，会从 REPV 值重新计数。因为只有当发生重复更新事件时，REPV_CNT 才会重新载入 REPV 值，所以只有在发生下一次重复更新事件时，写入 GP16C2Tn_REPAR 寄存器的值才会生效。 即，在 PWM 模式下，（REPV+1）相当于： – 在边沿对齐模式下，（REPV+1）对应的是 PWM 的周期数 – 在中央对齐模式下，（REPV+1）对应的是1/2 PWM的周期数

16. 5. 2. 17 通道捕获/比较寄存器 1（GP16C2Tn_CCVAL1）

通道捕获/比较寄存器 1（GP16C2Tn_CCVAL1）
偏移地址：44_H
复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																CCRV1															

Reserved	Bits 31-16	—	保留
CCRV1<15:0>	Bits 15-0	R/W	**捕获/比较数值 1** **信道 CHn 配置为输出：** CCRVn 中的值将被载入实际的捕获/比较寄存器中（预载值）。 如果在GP16C2Tn_CHMR1寄存器中的预载功能没有选中，CCRVn中的值将被永久载入；否则，每当发生更新事件，预载值将会复制到有效的捕获/比较寄存器中。有效捕获/比较寄存器中包含的值将会与GP16C2Tn_COUNT中的值进行比较，并在CHn上输出。 **信道CHn配置为输入：** CCRVn为以上一个输入捕获事件（In）传输的计数值。

16. 5. 2. 18 通道捕获/比较寄存器 2（GP16C2Tn_CCVAL2）

通道捕获/比较寄存器 2（GP16C2Tn_CCVAL2）
偏移地址：48_H
复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																CCRV2															

Reserved	Bits 31-16	—	保留
CCRV2<15:0>	Bits 15-0	R/W	**捕获/比较数值2** CCRVn 中的值将被载入实际的捕获/比较寄存器中（预载值）。 如果在GP16C2Tn_CHMR1寄存器中的预载功能没有选中，CCRVn中的值将被永久载入；否则，

			每当发生更新事件，预载值将会复制到有效的捕获/比较寄存器中。有效捕获/比较寄存器中包含的值将会与GP16C2Tn_COUNT中的值进行比较，并在CHn上输出。 **信道CHn配置为输入：** CCRVn为以上一个输入捕获事件（In）传输的计数值。

16. 5. 2. 19 刹车和死区配置寄存器（GP16C2Tn_BDCFG）

刹车和死区配置寄存器（GP16C2Tn_BDCFG）

偏移地址：54_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																GOEN	AOEN	BRKP	BRKEN	OFFSSR	OFFSSI	LOCKLVL	DT								

Reserved	Bits 31-16	—	保留
GOEN	Bit 15	R/W	**通道主要输出使能** 一旦刹车输入有效，该位会由硬件异步清零。该位可由软件置1或自动置1，取决于AOEN位。该位仅作用于配置为输出的通道。 0：CHn和CHnN输出禁止或强制为空闲状态。 1：如果CHn和CHnN各自的使能位都置1（GP16C2Tn_CCEP寄存器中的CCnEN, CCnNE），则CHn和CHnN输出使能。
AOEN	Bit14	R/W	**通道自动输出使能** 在发生更新事件时，将 GOEN 位置起 0：GOEN 仅可由软件置位 1：在下一个更新事件发生时（如果刹车输入无效），GOEN 可由软件或自动置位。 注意：当 GP16C2Tn_BDCFG 寄存器中的 LOCKLVL 位已被设置为锁定级别 1，则该位不可更改。
BRKP	Bit 13	R/W	**选择通道刹车极性** 0：刹车输入 BRKP 为低有效 1：刹车输入 BRKP 为高有效 注意：当 GP16C2Tn_BDCFG 寄存器中的 LOCKLVL 位已被设置为锁定级别 1，则该位不可更改 注意：任何对该位的写操作都要延时 1 个 APB 时

			钟周期后才变为有效。
BRKEN	Bit 12	R/W	**使能刹车** 0：刹车输入（BRKP 和 CCS 时钟失效事件）禁止 1：刹车输入（BRKP 和 CCS 时钟失效事件）使能 注意：当 GP16C2Tn_BDCFG 寄存器中的 LOCKLVL 位已被设置为锁定级别 1，则该位不可更改 注意：任何对该位的写操作都要延时 1 个 APB 时钟周期后才变为有效。
OFFSSR	Bit 11	R/W	**运行模式下关闭状态选择** 当 GOEN 为 1 且通道设置为输出时。当定时器禁止时，CCnEN 为 1 或 CCnNE 为 1 时，则 CHn/CHnN 输出 CCxPOL/CCxNPOL 0：当定时器禁能时，输出禁止（CHn/CHnN 使能输出信号=0）。 1：当定时器禁止时，输出 CCxPOL/CCxNOL 注：当 GP16C2Tn_BDCFG 寄存器中的 LOCKLVL 位已被设置为锁定级别 2，则该位不可更改。
OFFSSI	Bit 10	R/W	**空闲模式下关闭选择** 当 GOEN 为 0 且通道设置为输出时。当定时器禁止时，CCnEN 为 1 或 CCnNE 为 1 时，CHn/CHnN 输出空闲状态 0：当定时器禁止时，输出禁止（CHn/CHnN 使能输出信号=0）。 1：当定时器禁止时，输出空闲状态(OISSx /OISSxN) 注：当 GP16C2Tn_BDCFG 寄存器中的 LOCKLVL 位已被设置为锁定级别 2，则该位不可更改。
LOCKLVL<1:0>	Bits 9-8	R/W	**锁定级别配置** 针对软件错误，该位提供写保护。 00：**锁定关闭**–不提供写保护 01：**锁定级别 1 =** BDCFG 寄存器中的 DT，CON2 寄存器中的 OISSx 和 OISSxN，和 BDCFG 寄存器中的 BRKEN/BRKP/AOEN 不再可写。 10：**锁定级别 2 =** 锁定级别 1 + CC 极性位（CCEP 寄存器中的 CCnPOL/CCnNPOL，只要相关通道由 CCnSSEL 配置为输出）以及 OFFSSR 和 OFFSSI 都不再可写。 11：**锁定级别 3 =** 锁定级别 2 + CC 控制位（CHMR1 寄存器中的 CHnMOD 和 CHnPEN，只要相关通道由 CCnSSEL 配置为输出）都不再可写。 注：锁定配置为仅在复位后可写。一旦 BDCFG 已写，其设置内容在下一个复位前都处于冻结状态。

DT<7:0>	Bits 7-0	R/W	**死区延时** 设置值该位定义了互补输出之间插入的死区时间。DT 对应的就是该时间段。 **DT<7:5>=0xx** => DT=DT<7:0>x t_{dtg}，式中 $t_{dtg}=t_{DTS}$。 **DT<7:5>=10x** => DT=（64+DT<5:0>）xt_{dtg}，式中 $t_{dtg}=2xt_{DTS}$。 **DT<7:5>=110**=> DT=（32+DT<4:0>）xt_{dtg}，式中 $t_{dtg}=8xt_{DTS}$。 **DT<7:5>=111** => DT=（32+DT<4:0>）xt_{dtg}，式中 $t_{dtg}=16xt_{DTS}$。 注意：当 GP16C2Tn_BDCFG 寄存器中的 LOCKLVL 位已被设置为锁定级别 1，2 或 3，则该位不可更改

16. 5. 2. 20 DMA事件使能寄存器（GP16C2Tn_DMAEN）

DMA 事件使能寄存器（GP16C2Tn_DMAEN）

偏移地址：58_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																									TRGIDE	COMDE	Reserved		CH2DE	CH1DE	UDE

Reserved	Bits 31-7	—	**保留**
TRGIDE	Bit 6	R/W	**触发DMA请求使能** 0：触发DMA请求禁止 1：触发DMA请求使能
COMDE	Bit 5	R/W	**通信 DMA 请求使能** 0：通信 DMA 请求禁止 1：通信 DMA 请求使能
Reserved	Bits 4-3	—	**保留**
CH2DE	Bit 2	R/W	**通道捕获/比较 2 DMA 请求使能** 0：捕获/比较 2 DMA 请求禁止 1：捕获/比较 2 DMA 请求使能
CH1DE	Bit 1	R/W	**通道捕获/比较 1 DMA 请求使能** 0：捕获/比较 1 DMA 请求禁止 1：捕获/比较 1 DMA 请求使能
UDE	Bit 0	R/W	**更新 DMA 请求使能** 0：更新 DMA 请求禁止 1：更新 DMA 请求使能

第17章 基本定时器(BS16T)

17.1 概述

基本定时器（BS16T）包含一个 16 位自动重载计数器，该计数器由可配置的预分频器驱动。

通过使用定时器的分频器和 APB 时钟控制器的预分频功能，可对脉冲长度和波形周期进行数微妙到几毫秒的调整。

17.2 特性

- 16 位自动加载递增计数器
- 16 位可编程预分频器，可对计数器工作时钟进行 1 到 65536 的任意分频(运行中也可以)
- 计数上溢更新事件产生中断/DMA 请求

17.3 结构框图

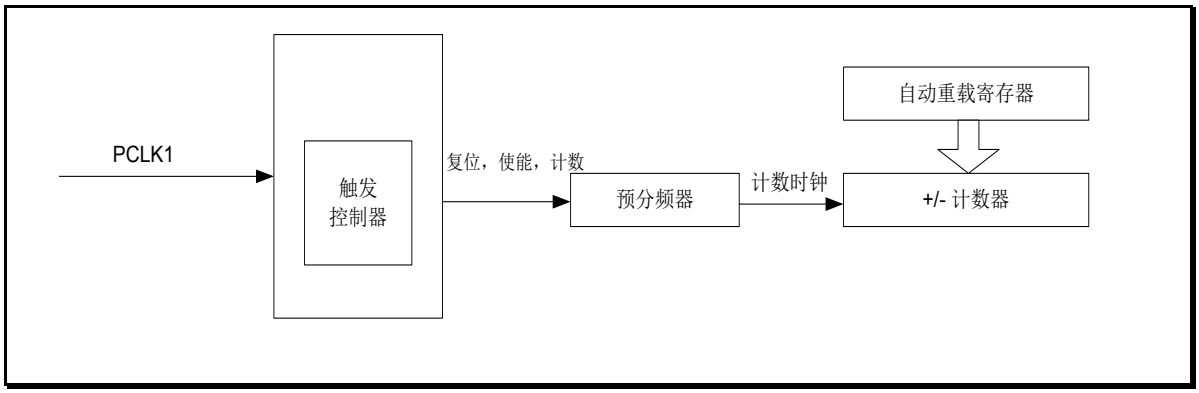


图 18-1 基本定时器电路结构框图

17.4 功能描述

17.4.1 预分频器

定时器包含一个 16-bit 的计数器（BS16Tn_COUNT），计数时钟由预分频寄存器（BS16Tn_PRES）进行分频。计数周期由自动重载计数器（BS16Tn_AR）设定。

自动重载寄存器（BS16Tn_AR）是一个可缓存的寄存器。当 BS16Tn_CON1 寄存器的 ARPEN 位为 0 时，BS16Tn_AR 寄存器重载功能失效，BS16Tn_AR 就是有效寄存器；ARPEN 为 1 时，BS16Tn_AR 寄存器具有重载功能，产生更新事件（UEV）时，加载值（BS16Tn_AR 寄存器值）更新到影子寄存器。

当 BS16Tn_CON1 寄存器中 DISUE 位为 0 时，计数器计数上溢（或递减下溢）时会产生

更新事件（UEV）。同样，软件方式也可产生更新事件。BS16Tn_CON1 寄存器的 CNTEN 置位时，计数器开始计数。

注：计数器在 CNTEN 位置位 1 个时钟周期后开始计数。

预分频器可对定时器工作时钟进行 BS16Tn_PRES 寄存器值+1 次分频。由于 BS16Tn_PRES 是一个可重载寄存器，因此，定时器工作时可以对该寄存器进行修改，修改值在下次更新事件（UEV）后有效。

下图给出了定时器运行过程中改变预分频值时计数器的计数情况。

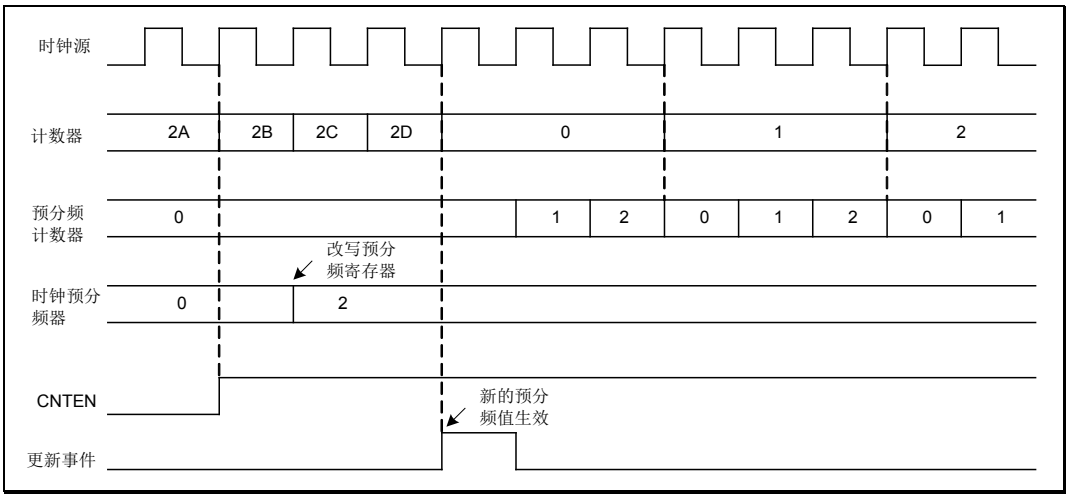


图 24-1　预分频值计数时序图

17.4.2　时钟源

计数时钟由内部时钟源（PCLK1）提供。

CNTEN 位（BS16Tn_CON1 寄存器）与 SGU 位（BS16Tn_SGE 寄存器）为实际控制位，这两个位只能软件修改（SGU 位除外，仍硬件自动清除）。一旦 CNTEN 位被写为'1'，预分频器就由内部 PCLK1 提供时钟。

17.4.3　递增计数模式

在递增计数模式中，计数器由 0 开始计数至自动重载值（BS16Tn_AR 寄存器中的值），然后从 0 开始重新计数并产生一个计数溢出事件。

软件置位 BS16Tn_CON1 寄存器中的 DISUE 位可关闭更新事件（UEV）的产生。更新事件（UEV）关闭，可避免向预载寄存器写新值的过程中更新影子寄存器。这种情况下，DISUE 位在写'0'之前都不会产生更新事件。正常产生更新事件后，计数器和预载计数器都是从 0 重新开始（但预分频值没有改变）。此外，若置位 BS16Tn_CON1 寄存器中的 UERSEL 位（更新请求选择），置位 SGU 位时会产生一次更新事件（UEV），但 UEVTIF 标志位不会置位（因此，不会触发中断或 DMA 请求）。

当更新事件发生时，所有寄存器都会被更新且更新标志位（BS16Tn_RIF 寄存器中的 UEVTIF 位）置位（取决于 UERSEL 位）：

◇　自动重载影子寄存器加载 BS16Tn_AR 寄存器中的值

◇ 预分频器之缓冲器加载 BS16Tn_PRES 寄存器中的值

17. 4. 4 调试模式

当微控制器进入调试模式（Cortex™-M 内核终止）,计数器可被设定停止计数。

17. 5 特殊功能寄存器

17. 5. 1 寄存器列表

BS16T 寄存器列表		
寄存器名称	偏移地址	寄存器描述
BS16Tn_CON1	000_H	控制寄存器 1
Reserved	004_H	保留
BS16Tn_IER	$00C_H$	中断使能寄存器
BS16Tn_IDR	010_H	中断禁止寄存器
BS16Tn_IVS	014_H	中断有效状态寄存器
BS16Tn_RIF	018_H	原始中断标志寄存器
BS16Tn_IFM	$01C_H$	中断标志屏蔽寄存器
BS16Tn_ICR	020_H	中断清零寄存器
BS16Tn_SGE	024_H	事件生成寄存器
BS16Tn_COUNT	034_H	计数寄存器
BS16Tn_PRES	038_H	预分频寄存器
BS16Tn_AR	$03C_H$	自动重载寄存器
BS16Tn_DMAEN	058_H	DMA 使能寄存器

17. 5. 2 寄存器描述

17. 5. 2. 1 控制寄存器 1（BS16Tn_CON1）

控制寄存器 1（BS16Tn_CON1）
偏移地址：000_H
复位值：$00000000\_00000000\_00000000\_00000000_B$

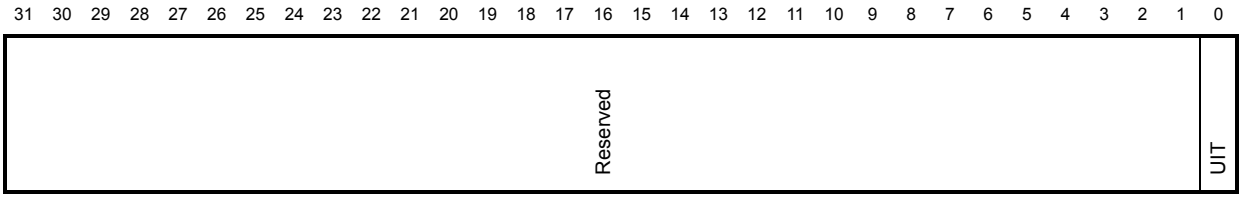
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																								APREN	Reserved			SPMEN	UERSEL	DISUE	CNTEN

Reserved	Bit 31-8	-	**保留，必须保持为复位值**
APREN	Bit7	R/W	**自动重载预载使能** 0：BS16Tn_AR 寄存器未缓冲 1：BS16Tn_AR 寄存器被装入缓冲器
Reserved	Bit 6-4	-	**保留，必须保持为复位值**
SPMEN	Bit 3	R/W	**单脉冲模式** 0：当发生更新事件时，计数器不停止。 1：当发生下一次更新事件（CEN 位清零）时，计数器停止。
UERSEL	Bit 2	R/W	**更新请求源** 该位由软件置 1 或清零，来选择 UEV 事件源。 0：如果更新中断或 DMA 请求使能，则下述任一事件都可产生更新中断或 DMA 请求： –计数器上溢/下溢 –设置 SGU 位 –从模式控制器产生的更新 1：如果更新中断或 DMA 请求使能，仅计数器上溢/下溢才能产生更新中断或 DMA 请求中断
DISUE	Bit1	R/W	**更新禁止** 该位由软件置 1 或清零来使能/禁止 UEV 事件的产生。 0：UEV 使能. 更新事件（UEV）由下列任一事件产生： – 计数器上溢/下溢 –设置 SGU 位 –从模式控制器产生的更新 缓冲寄存器载入他们的预载值。 1：UEV 禁止.不产生更新事件，影子寄存器保持他们的值（AR，PSCV）。如果从从模式控制器接收到硬件复位，计数器和预分频器将被重新初始化。
CNTEN	Bit0	R/W	**CNTEN：计数器使能**

			0：计数器禁止 1：计数器使能 注意：如果软件设置了 CNTEN 位，外部时钟，门控模式和编码器模式才能工作。触发模式可由硬件自动设置 CNTEN 位。

17. 5. 2. 2 中断使能寄存器（BS16Tn_IER）

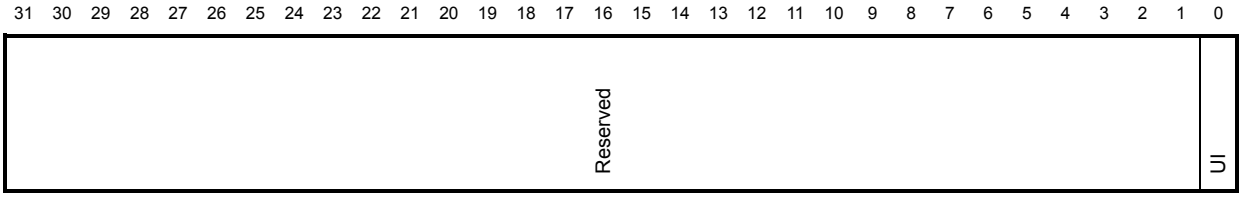
中断使能寄存器（BS16Tn_IER）
偏移地址：$00C_H$
复位值：$00000000\_00000000\_00000000\_00000000_B$



Reserved	Bit 31-1	-	**保留，必须保持为复位值**
UIT	Bit 0	W1	**更新中断使能** 0：无效 1：使能

17. 5. 2. 3 中断禁止寄存器（BS16Tn_IDR）

中断禁止寄存器（BS16Tn_IDR）
偏移地址：010_H
复位值：$00000000\_00000000\_00000000\_00000000_B$



Reserved	Bit 31-1	-	**保留，必须保持为复位值**
UI	Bit 0	W1	**更新中断禁止** 0：无效 1：禁止

17. 5. 2. 4 中断有效状态寄存器（BS16Tn_IVS）

中断有效状态寄存器（BS16Tn_IVS）
偏移地址：014_H
复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-1	-	**保留，必须保持为复位值**
UEI	Bit 0	R	**更新中断有效状态** 0：禁止更新中断 1：使能更新中断 IER/IDR 写 1 来使能或禁止该位。

17. 5. 2. 5 原始中断标志寄存器（BS16Tn_RIF）

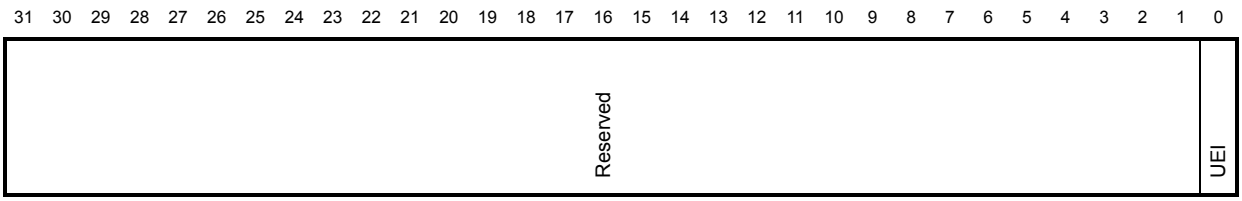
原始中断标志寄存器（BS16Tn_RIF）
偏移地址：018_H
复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-1	-	保留，必须保持为复位值
UEVTIF	Bit 0	R	**更新中断标志** 如果更新中断使能，当发生更新事件，该标志位由硬件置起。对 BS16Tn_ICR 写 1 来清除原始中断. 0：未发生更新. 1：更新中断被挂起. 当寄存器更新时，该位被硬件置起： –当重复计数器值发生上溢或者下溢（若重复计数器=0，则更新）和当 BS16Tn_CON1 寄存器中 DISUE=0 –当使用 BS16Tn_SGE 寄存器中的 SGU 位来由软件重新初始化 CNT 时，如果 BS16Tn_CON1 寄存中的 UERSEL=0 和 DISUE=0 –当 CNT 由触发事件来重新初始化，如果 BS16Tn_CON1 寄存中的 UERSEL=0 和 DISUE=0。

17. 5. 2. 6 中断标志屏蔽寄存器（BS16Tn_IFM）

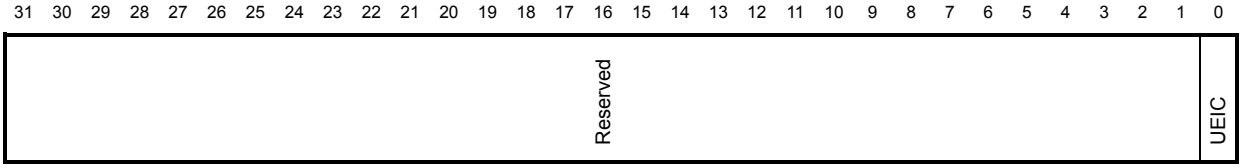
中断标志屏蔽寄存器（BS16Tn_IFM）
偏移地址：$01C_H$
复位值：$00000000\_00000000\_00000000\_00000000_B$



Reserved	Bit 31-1	-	保留，必须保持为复位值
UEI	Bit 0	R	**更新中断标志屏蔽** 如果更新中断使能，当发生更新事件，该标志位由硬件置起。对 BS16Tn_ICR 写 1 来清除原始中断. 0：未发生更新. 1：更新中断被挂起. 当寄存器更新时，该位被硬件置起: –当重复计数器值发生上溢或者下溢（若重复计数器=0，则更新）和当 BS16Tn_CON1 寄存器中 DISUE=0 –当使用 BS16Tn_SGE 寄存器中的 SGU 位来由软件重新初始化 CNT 时，如果 BS16Tn_CON1 寄存中的 UERSEL=0 和 DISUE=0 –当 CNT 由触发事件来重新初始化，如果 BS16Tn_CON1 寄存中的 UERSEL=0 和 DISUE=0。

17. 5. 2. 7 中断清零寄存器（BS16Tn_ICR）

中断清零寄存器（BS16Tn_ICR）
偏移地址：020_H
复位值：$00000000\_00000000\_00000000\_00000000_B$



Reserved	Bit 31-1	-	**保留，必须保持为复位值**
UEIC	Bit 0	C_W1	**更新中断清零** 0：无效 1：更新中断清零（BS16Tn_RIF）

17. 5. 2. 8 事件生成寄存器（BS16Tn_SGE）

事件生成寄存器（BS16Tn_SGE）
偏移地址：024_H
复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-1	-	**保留，必须保持为复位值**
SGU	Bit 0	W1	**更新生成** 该位由软件设置，可由硬件自动清零. 0：无动作 1：重新初始化计数器，更新寄存器。注意，预分频器也会被清零（但预分频比不会受到影响）。

17. 5. 2. 9 计数寄存器（BS16Tn_COUNT）

计数寄存器（BS16Tn_COUNT）
偏移地址：034_H
复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-16	-	**保留，必须保持为复位值**
CNTV	Bit 15-0	R/W	**计数值**

17. 5. 2. 10 预分频寄存器（BS16Tn_PRES）

预分频寄存器（BS16Tn_PRES）
偏移地址：038_H
复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-16	-	**保留，必须保持为复位值**
PSCV	Bit 15-0	R/W	**预分频器值** 计数器时钟频率（CK_CNT）= fCK_PSC /（PSCV<15:0> + 1） 每发生一次更新事件（包括当计数器由 BS16Tn_SGE 寄存器中的 SGU 位清零或当配置为复位模式时，通过触发控制器清零），PSCV 包含的值将被填入到有效的预分频寄存器内。

17. 5. 2. 11 自动重载寄存器（BS16Tn_AR）

自动重载寄存器（BS16Tn_AR）
偏移地址：$03C_H$
复位值：$00000000\_00000000\_11111111\_11111111_B$

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

Reserved	Bit 31-16	-	**保留，必须保持为复位值**
ARRV	Bit 15-0	R/W	**自动重载值** AR 中的值将被载入实际的自动重载寄存器中。 当自动重载值为空，计数器被屏蔽。

17. 5. 2. 12 DMA使能寄存器（BS16Tn_DMAEN）

DMA 使能寄存器（BS16Tn_DMAEN）																															
偏移地址：058_H																															
复位值：$00000000\_00000000\_00000000\_00000000_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																															UDEN

Reserved	Bit 31-1	-	保留，必须保持为复位值
UDEN	Bit 0	R/W	**DMA 访问使能** 0：DMA 访问禁止 1：DMA 访问使能

第18章 串行总线（I2C0~1）

18. 1 概述

I2C 总线用于微控制器与外部 I2C 总线节点通信连接，可支持多主模式功能，适合用于多主机应用的场景。用户可选择 100kHz 或 400kHz 通信速率。接口同时兼容 SMBus 2.0。硬件支持 CRC 校验来提高通信可靠性。同时 I2C 通信可结合 DMA 使用，减少 CPU 的软件负荷。

18. 2 特性

- 多主模式功能：同一接口主机或从机模式可配
- 作为主机时硬件可为总线产生时钟，通信时可硬件产生开始和结束信号
- 作为从机时通过编程的 I2C 地址检测来识别通信对象，为更加灵活，自身可设置两个地址，对任一地址都进行应答，支持停止位的检测
- 通讯地址可编程，支持常用的 7 位，10 位以及广播方式的呼叫和地址检测
- 通信速度可编程，100kHz 以内选择标准模式，高达 400kHz 时选择快速模式
- 状态标识可检测，通过以下检测的状态标识来判断当前完成的动作
  - 可查询发送/接收模式标识判断当前接口模式
  - 可查询字节传输结束标识判断当前字节是否传输完成
  - 可查询忙碌标识判断 I2C 总线是否正在通信
- 错误标识可检测，通过以下检测的错误标识来判断当前通信错误的原因
  - 可查询主模式下的仲裁是否丢失
  - 可查询地址或数据传输完成后的应答是否失败
  - 可查询是否发生误放的起始位或停止位
  - 可查询禁止时钟延长后出现的上溢或下溢事件发生
- 支持 2 个中断源：
  - 由成功的地址/数据字节传输事件触发中断
  - 由错误状态触发中断
- 根据应用可配置是否启用时钟延长功能
- 带 DMA 功能的 1 字节缓冲
- 可使能数据包错误校验（PEC）功能，在发送模式下生成 PEC 值并在发送完数据后发送 PEC 值，在接收模式下将最后一个接收的字节作为 PEC 去进行错误校验
- 可匹配 SMBus 2.0 的使用，特性如下：
  - SCL 时钟引脚低电平保持时间最多达 25ms
  - 主器件累计的 SCL 时钟引脚低电平延长时间最多达 10ms
  - 从器件累计的 SCL 时钟引脚低电平延长时间最多达 25ms
  - 使能 PEC 功能后支持硬件 PEC 的生成和校验
  - 支持地址解析协议

◇ 同时支持 SMBus

18. 3 结构框图

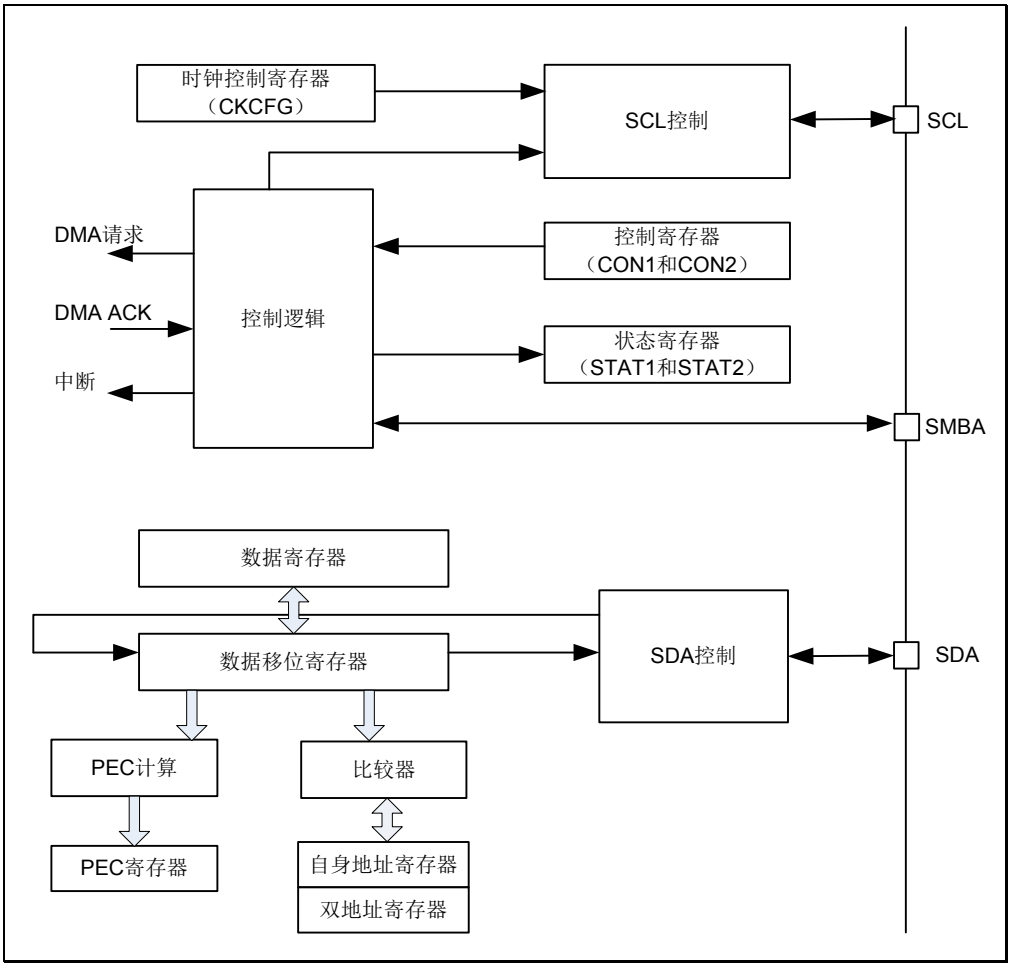


图 19-1 I2C 电路结构框图

18. 4 功能描述

I2C 总线有两根线，时钟线（SCL）和数据线（SDA），接口通过这两个引脚接入总线中，支持的通信速度可支持标准模式（100kHz 以内）或高速模式（400kHz 以内）。

18. 4. 1 通信协议简介

I2C 接口可配置以下四种模式：从发送器、从接收器、主发送器、主接收器

接口的默认配置为从机。当接口生成起始位后，可自动由从机转换为主机，因为只有主机会产生起始信号；当发生仲裁丢失或产生停止信号时，接口从主机转换为从机。不论从机发送还是主机发送，时钟信号总是由主机发送给从机。而数据的传输总是以起始位发送后开始，出现停止位后结束。作为主机时用户可控制起始位和停止位的产生来控制数据传输。作为从机时，可编程自身的地址（7 位或 10 位可配置）以及广播呼叫地址，当通信时检测到与自身地址或广播呼叫地址匹配后产生应答。广播呼叫地址检测功能可以通过软件使能或关闭。I2C 接口以字节（8 位）进行传输，MSB 在前。标准通信格式为主机在发送完起始位后，发送的是地址，根据地址长度的设置（7 位或 10 位），来决定地址有几个字节（1 个或 2 个）。数据和地址均以 8 位字节传输，MSB 在前。起始位后紧随地址字节（7 位地址占据一个字节；10 位地址占据两个字节）。地址始终在主模式下传送。在字节传输 8 个时钟周期后是第 9 个时钟脉冲，在此期间接收器若通信成功必须向发送器发送一个应答位（ACK）。

请参见下图：

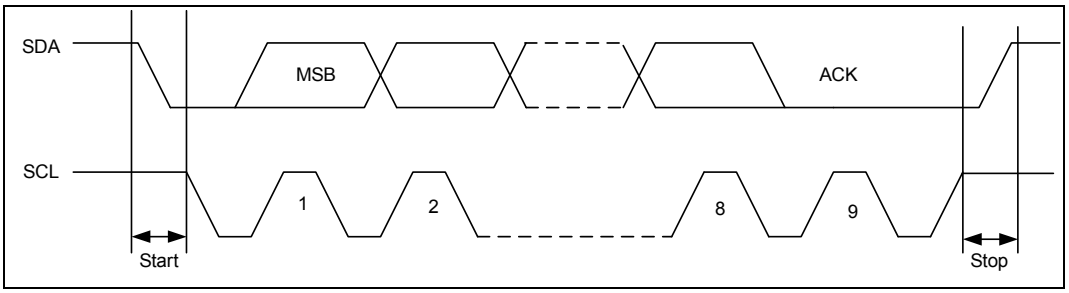


图 19-2 I2C 总线协议

应答位可由软件使能或关闭。

18. 4. 2 I2C主模式通信

当 I2C 接口工作在主模式时，输出时钟并控制着数据的传输。置位 I2C_CON1.START 位会在总线上发送起始位，进入此模式。串行总线上的数据总是在起始信号出现后开始传输，并在停止信号出现后停止。

I2C 总线的时钟信号是由主机提供的，所以在主模式下需要先对时钟进行设置，具体执行步骤如下

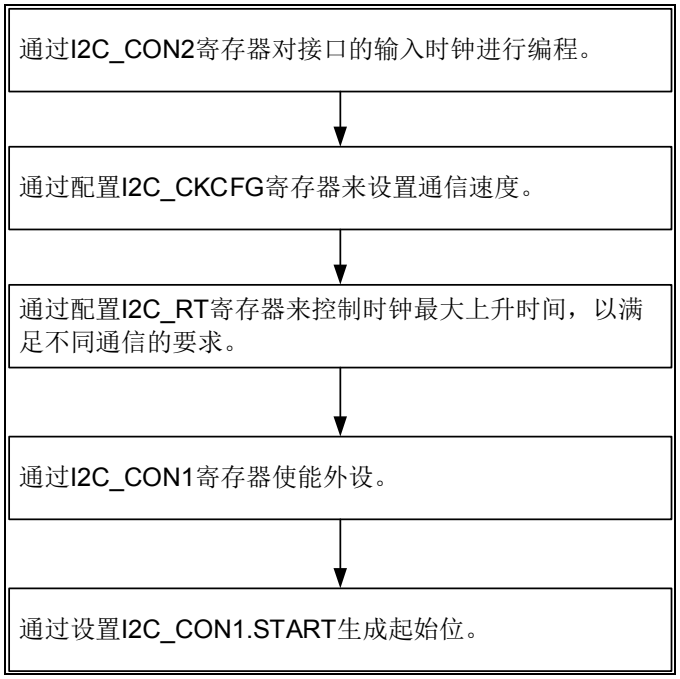


图 19-3 I2C 主模式时钟设定

注：外设输入时钟频率是有下限的，在标准模式下不得低于 2MHz，在快速模式下不得低于 4MHz。

起始位

当此时接口处于空闲状态，即 I2C_STAT2.BSYF 为清 0 状态时，可以将 I2C_CON1.START 位置 1 以生成一个起始信号，随后 I2C 会自动切换成主模式状态，可通过 I2C_STAT2.MASTER 位去判断是否切换成主模式。

注意：若 I2C 接口一开始便处于主机状态，将 I2C_CON1.START 位置 1 后则会在当前字节传输完成后，生成一个重复起始信号。

起始信号发出之后，I2C_STAT1.SENDSTR 位会被硬件置 1，当 I2C_CON2.EVTIE 位置 1 时会产生一个中断，接着主机等待软件对 I2C_STAT1 进行读操作，然后把从机地址写入 I2C_DATA 寄存器中，来发出地址。

从地址传输

接下来从地址会通过内部移位寄存器发送到 SDA 线。

◇ 在 10 位寻址模式中，发送头序列会产生以下事件：

- I2C_STAT1.SENDADD10 位会由硬件置 1 并在 I2C_CON2.EVTIE 位置 1 时生成一个中断。接下来主设备会等待软件读取 I2C_STAT1，然后把第二个地址字节写入 I2C_DATA 寄存器。
- I2C_STAT1.ADDR 位会由硬件置 1 并在 I2C_CON2.EVTIE 位置 1 时生成一个中断。接下来主设备会等待对 I2C_STAT1 寄存器执行读操作，然后对 I2C_STAT2 寄存器执行读操作。

◇ 在 7 位寻址模式下，会发送一个地址字节。地址字节被发出后，

- I2C_STAT1.ADDR 位会由硬件置 1 并在 I2C_CON2.EVTIE 位置 1 时生成一个中断。接下来主设备会等待对 I2C_STAT1 寄存器执行读操作，然后对 I2C_STAT2 寄存

器执行读操作。主设备会根据发送的从地址字节 LSB 来决定是进入发送模式还是接收模式。

◇ 在 7 位寻址模式下，地址字节的最后一位表示主机要进入发送模式或是接收模式，当主机发送从机地址，并将地址字节最低位复位，则是进入发送模式；若将最低位置 1，则主机进入接收模式。

◇ 在 10 位寻址模式下，主机都要先发头序列，然后发送从地址低 8 位，若是要进入发送模式，则此时地址发送完成后就已经进入发送模式；若要进入接收模式，则在发送完从地址低 8 位后，需要再发送一个重复起始信号，然后再发送头序列，主机便进入接收模式。

I2C_STAT2.TRF 位指示主设备是处于接收模式还是处于发送模式。

主发送器

在主机发送完地址，从机响应后，主机 I2C_STAT1.ADDR 位会置 1，用户需要将 I2C_STAT1.ADDR 位清 0，才会通过其内部移位寄存器将写入 I2C_DATA 寄存器中的字节从 SDA 线上发出。在将 I2C_STAT1.ADDR 清 0 后，主机会一直等待用户将数据写入 I2C_DATA。

当接收到从机发来的应答脉冲后，I2C_STAT1.TXBE 位会由硬件置 1，用户若想使用中断发送功能，则可配置 I2C_CON2.EVTIE 和 I2C_CON2.BUFIE 位都置 1，在 I2C_STAT1.TXBE 位为 1 时会生成一个发送空中断。

I2C_STAT1.TXBE 被硬件置位，而在下一个数据传输结束之前没有新的数据写入 I2C_DATA 寄存器中，则 I2C_STAT1.BTC 位会自动置位，而如果不将 I2C_STAT1.BTC 位清 0，则 SCL 线会被硬件一直拉低。I2C_STAT1.BTC 位清 0 的操作为先对 I2C_STAT1 寄存器进行一次读操作，再对 I2C_DATA 寄存器执行一次写操作，即可将 I2C_STAT1.BTC 位清 0。

当最后一个字节写入 I2C_DATA 寄存器后，用户将 I2C_CON1.STOP 位置 1 以生成一个停止信号。接口会自动切换回从模式。

注意：当 I2C_STAT1.TXBE 或 I2C_STAT1.BTC 中的任何一个置 1 时，应在 EV8_2 事件期间对停止位进行编程。

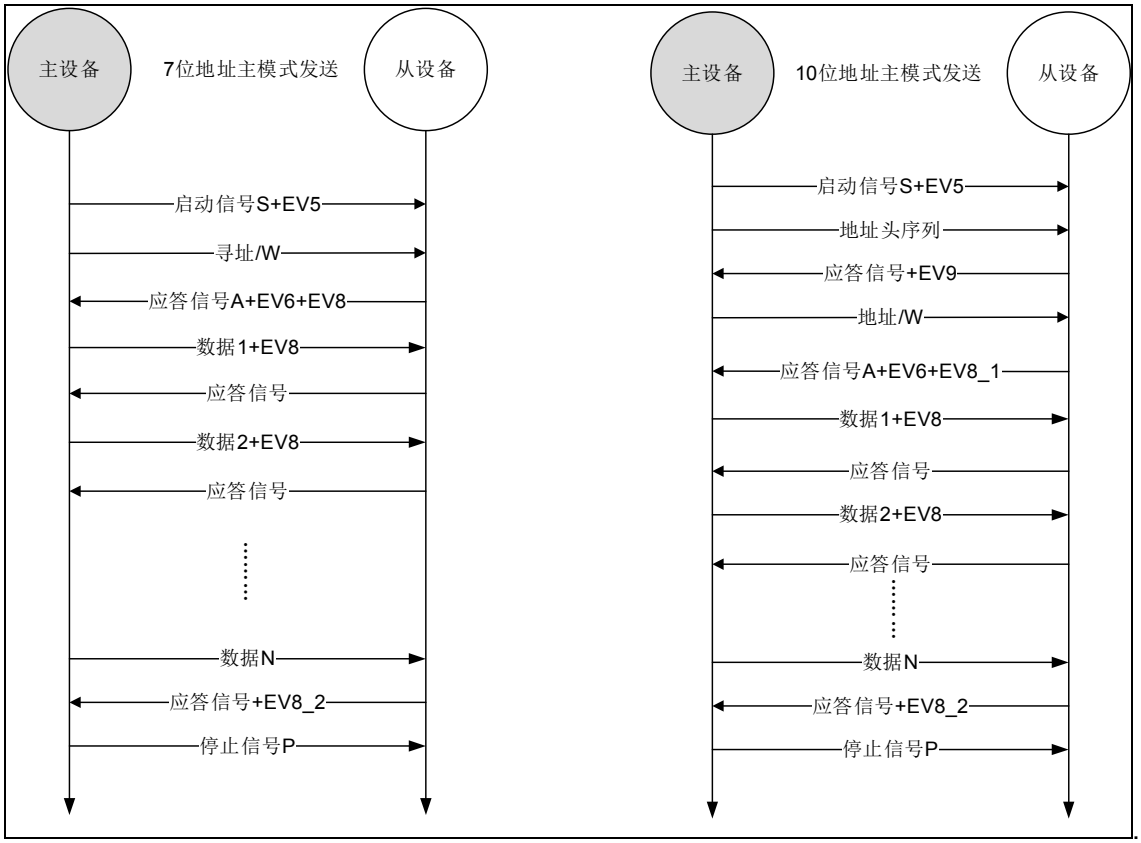


图 19-4 主发送器的传输序列图

注 1：S=起始位，SR=重复起始位，P=停止位，A=应答

注 2：EVx=事件（如果 EVTIE=1 则发生中断）

注 3：EV5：当 I2C_STAT1.SENDSTR=1 时，需要先读 I2C_STAT1 寄存器，再将从机地址写入 I2C_DATA 寄存器来清零。

注 4：EV6：当 I2C_STAT1.ADDR=1 时，需要先读 I2C_STAT1 寄存器，再读 I2C_STAT2 寄存器来清零。

注 5：EV8_1：当 I2C_STAT1.TXBE=1 时，移位寄存器为空，数据寄存器为空，在 I2C_DATA 中写入数据 1。

注 6：EV8_2：当 I2C_STAT1.TXBE=1，I2C_STAT1.BTC=1，表示程序请求停止。I2C_STAT1.TXBE 和 I2C_STAT1.BTC 位由硬件通过停止条件清零。

注 7：EV9：当 I2C_STAT1.SENDADD10=1 时，需要先读 I2C_STAT1 寄存器,再写 I2C_DATA 寄存器来清零。

注 8：EV5、EV6、EV9、EV8_1 和 EV8_2 事件可延长 SCL 低电平时间，直到相应的软件序列结束为止。

注 9：如果软件序列在下一个字节传输结束之前未能完成，EV8 事件会延长 SCL 低电平时间。

主接收器

当传输完地址，从机响应后，主机 I2C_STAT1.ADDR 位置 1，用户将 I2C_STAT1.ADDR 位清 0 后，I2C 接口会进入主接收模式，主设备会通过内部移位寄存器接收从 SDA 线传来的字节数据并将其保存到 I2C_DATA 寄存器。每次接收到一个字节后，若使能了 ACK 功能，则主接收器会发出一个应答信号，同时 I2C_STAT1.RXBNE 位会由硬件置 1。与发送类似，若用户希望使用中断接收方式，则可同时置位 I2C_CON2.EVTIE 和 I2C_CON2.BUFIE 位，则每当 I2C_STAT1.RXBNE 置 1 时会产生一次中断。

如果在上一次数据接收结束之前 I2C_STAT1.RXBNE 位已置 1 但 I2C_DATA 寄存器中的数据尚未读取，则 I2C_STAT1.BTC 位会由硬件置 1，而接口会一直延长 SCL 低电平，等待 I2C_DATA 寄存器被写入，以将 I2C_STAT1.BTC 清零。

在主设备接收到从设备传来的最后一个字节数据后，需要发送一个 NACK（注意不是 ACK），这样从设备接收完成后将释放对 SCL 和 SDA 线的控制。在软件具体操作时，有一些流程需要注意：

1. 为了在最后一个接收数据字节后生成非应答脉冲，须在读取倒数第二个数据字节后即将 I2C_CON1.ACKEN 位清零。
2. 要生成停止位/重复起始位，须在读取倒数第二个数据字节后将 I2C_CON1.STOP 或 I2C_CON1.START 位置 1。
3. 在只接收单个字节的情况下，会在 EV6 期间（在 I2C_STAT1.ADDR 标志清零之前）禁止应答并在 EV6 之后生成停止位。

生成停止位后，接口会自动返回从模式。

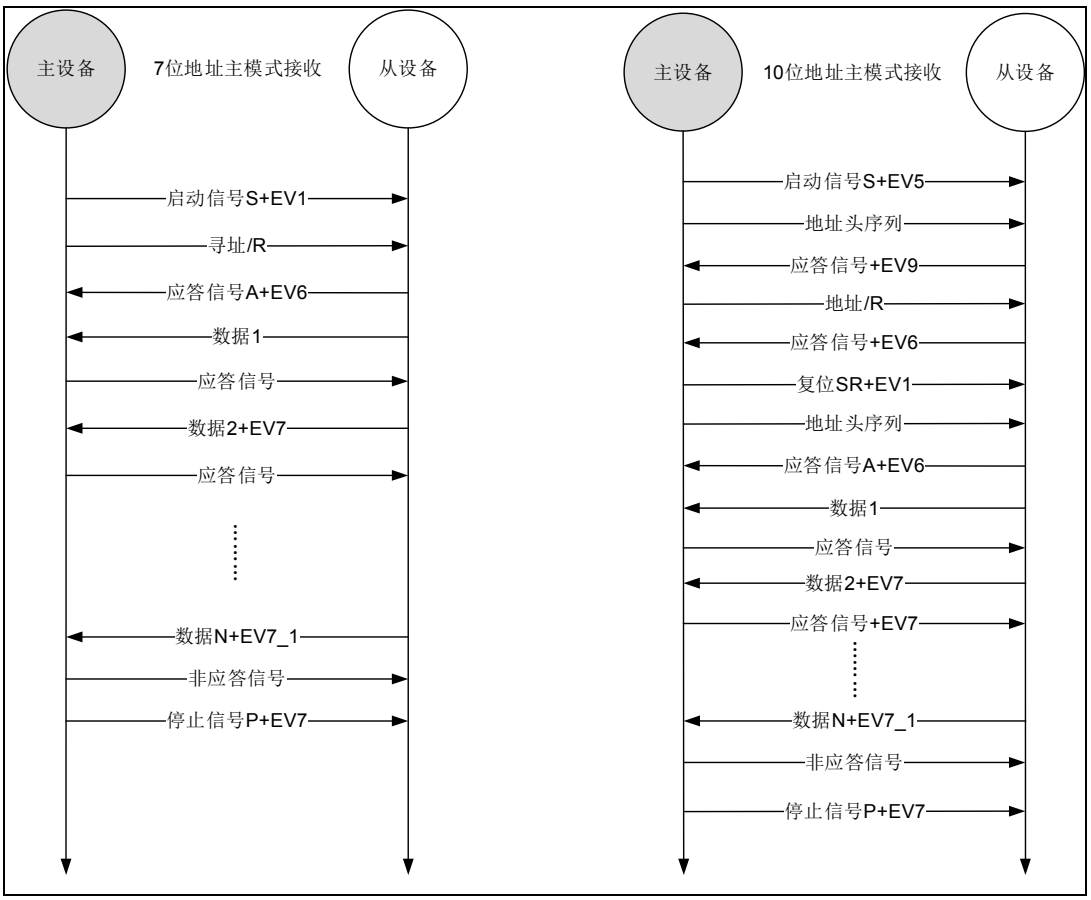


图 19-5 主接收器的传输序列图

注 1：S=起始位，SR=重复起始位，P=停止位，A=应答，NA=非应答

注 2：EVx=事件（如果 EVTIE=1 则发生中断）

注 3：EV5：当 I2C_STAT1.SENDSTR=1 时，需要先读 I2C_STAT1 寄存器，再将从机地址写入 I2C_DATA 寄存器来清零。

注 4：EV6：当 I2C_STAT1.ADDR=1 时，需要先读 I2C_STAT1 寄存器,再读 I2C_STAT2 寄存器来清零。在 10 位主接收器模式下，执行此序列后，应该在 I2C_CON1.START=1 的情况下写 I2C_CON2。如果接收 1 个字节，则必须在 EV6 事件期间（将 I2C_STAT1.ADDR 标志清零之前）禁止应答。

注 5：EV7：当 I2C_STAT1.RXBNE=1 时，需要读 I2C_DATA 寄存器来清零。

注 6：EV7_1：当 I2C_STAT1.RXBNE=1 时，需要读 I2C_DATA 寄存器、设定 I2C_CON1.ACKEN=0 和 I2C_CON1.STOP 请求来清零。

注 7：EV9：当 I2C_STAT1.SENDADD10=1 时，需要先读 I2C_STAT1 寄存器,再写 I2C_DATA 寄存器来清零。

注 8：如果接收单个字节，则为 NA。

注 9：EV5、EV6 和 EV9 事件可延长 SCL 低电平时间，直到相应的软件序列结束为止。

注 10：如果软件序列在下一个字节接收结束之前未能完成，EV7 事件会延长 SCL 低电平时间。

注 11：EV7_1 软件序列必须在当前字节传输的 ACK 脉冲之前结束。

如果 EV7_1 软件序列未能在当前字节传输的 ACK 脉冲之前结束，建议采用下列步骤。

必须遵循以下步骤以确保：

◇ 在结束最后一个数据接收之前及时将 I2C_CON1.ACKEN 位复位

◇ 在最后一个数据接收后将 I2C_CON1.STOP 位置位且不接收补充数据。

2 字节的接收流程：

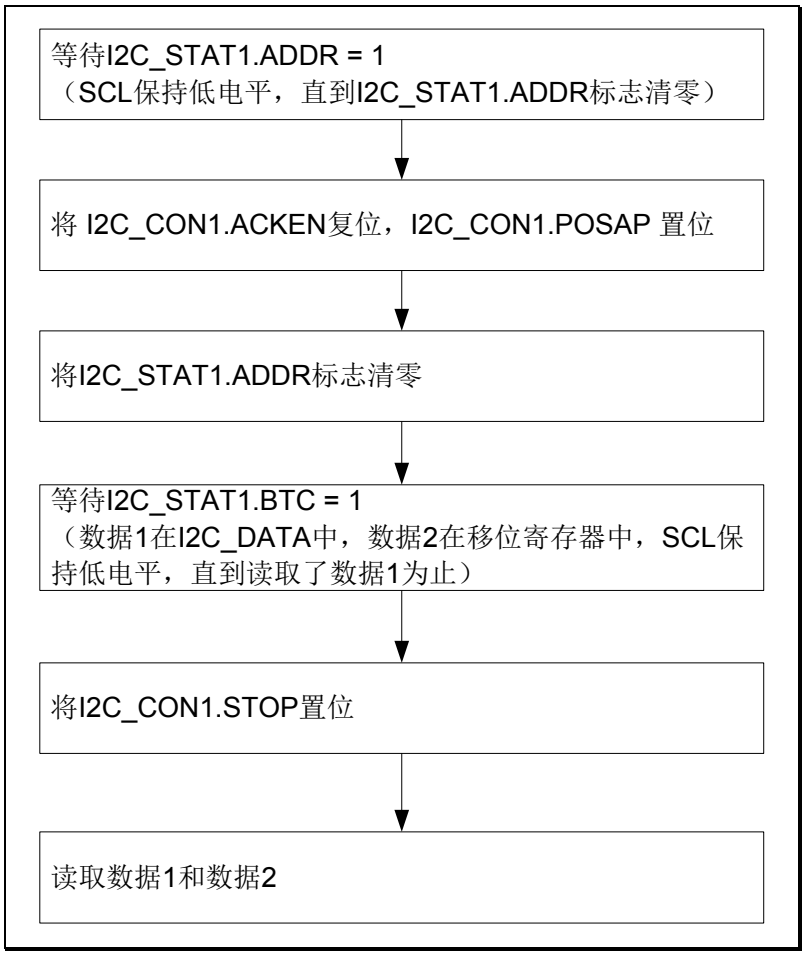


图 19-6 I2C 主模式 2 字节接收

N > 2 的字节接收流程，从 N-2 数据接收开始

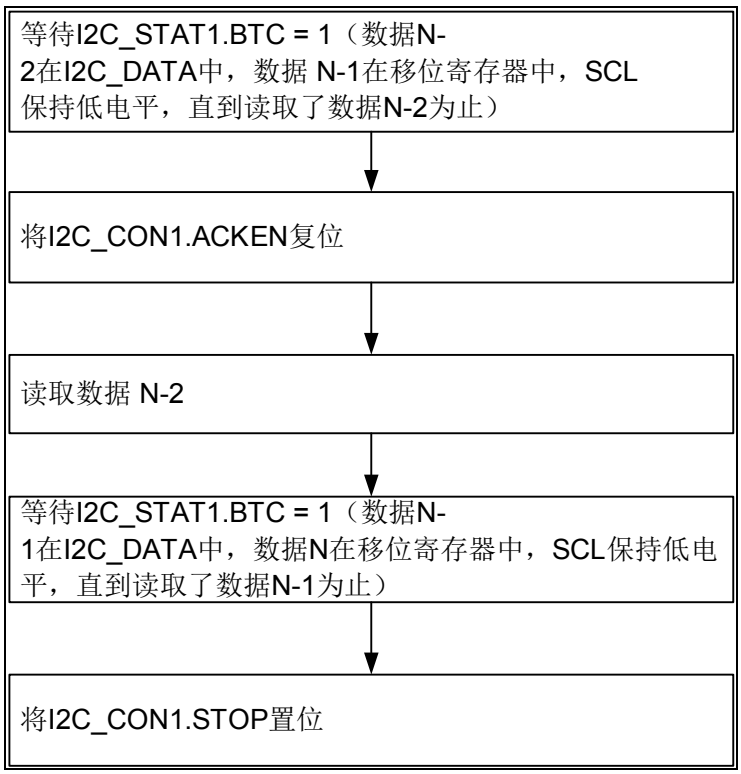


图 19-7 I2C 主模式接收 2

18. 4. 3 I2C从模式通信

I2C 接口默认工作在从模式状态，可以通过主动发送起始位的方式切换到主模式。时钟是通信的基础，所以在初始化时必须先进行时钟编程，通过对 I2C_CON2.CLKF 位编程，对外设时钟输入进行配置。其中有一些注意事项：标准模式下时钟频率不能低于 2MHz，快速模式下时钟频率不能低于 4MHz。

当从机检测到起始位后，会将随后接收到的 1 字节或 2 字节数据（地址长度设为 10）作为地址，将其与接口地址 I2C_ADDR1 和 I2C_ADDR2（当 I2C_ADDR2.DUALEN=1）或广播呼叫地址（当 I2C_CON1.GCEN=1）进行比较。

注：对于地址长度为 10 位时，2 字节数据分别是头段序列（11110xx0）和低 8 位地址，其中 xx 是地址的高两位有效位。

具体在地址匹配过程中可能会发生以下几种情况：

当 1 或 2 字节地址中任一地址不匹配，接口会忽略此次通信连接并等待下一个起始位。

在 10 位模式下，头地址匹配后 I2C_CON1.ACKEN 位置 1，接口会生成一个应答脉冲并等待低 8 位地址。

当地址匹配后，接口会做出如下动作

1. I2C_CON1.ACKEN 会置 1，并产生一个应答脉冲
2. I2C_STAT1.ADDR 位置 1，当需要在匹配地址后产生一个中断，用户可配置 I2C_CON2.EVTIE 位置 1

3. 如果 I2C_ADDR2.DUALEN=1，则软件必须读取 I2C_STAT2.DMF 位状态来核对哪些从地址进行了应答

从机默认为接收模式，若需要进入发送模式，则在匹配好地址后，主机会给从机发送重复起始信号，当从机接收到该重复起始位后，进入发送模式。用户可在从模式下检测 I2C_STAT2.TRF 位来判断当前处于接收模式还是发送模式。

从发送器

在接收到匹配的地址时，I2C_STAT1.ADDR 位会置位，需要软件将 I2C_STAT1.ADDR 清零，从设备会通过内部移位寄存器将 I2C_DATA 寄存器中的字节发送到 SDA 线。

从设备会延长 SCL 低电平时间，直到 I2C_STAT1.ADDR 位清零且 I2C_DATA 寄存器中填满待发送数据为止。

当发送成功，主机会回应答信号，从机接收到应答信号时，I2C_STAT1.TXBE 位被硬件置位，用户若想使用中断发送功能，则可配置 I2C_CON2.EVTIE 和 I2C_CON2.BUFIE 位都置 1，在 I2C_STAT1.TXBE 位为 1 时会生成一个发送空中断。

I2C_STAT1.TXBE 被硬件置位，而在下一个数据传输结束之前没有新的数据写入 I2C_DATA 寄存器中，则 I2C_STAT1.BTC 位会自动置位，而如果不将 I2C_STAT1.BTC 位清 0，则 SCL 线会被硬件一直拉低。I2C_STAT1.BTC 位清 0 的操作为先对 I2C_STAT1 寄存器进行一次读操作，再对 I2C_DATA 寄存器执行一次写操作，即可将 I2C_STAT1.BTC 位清 0。

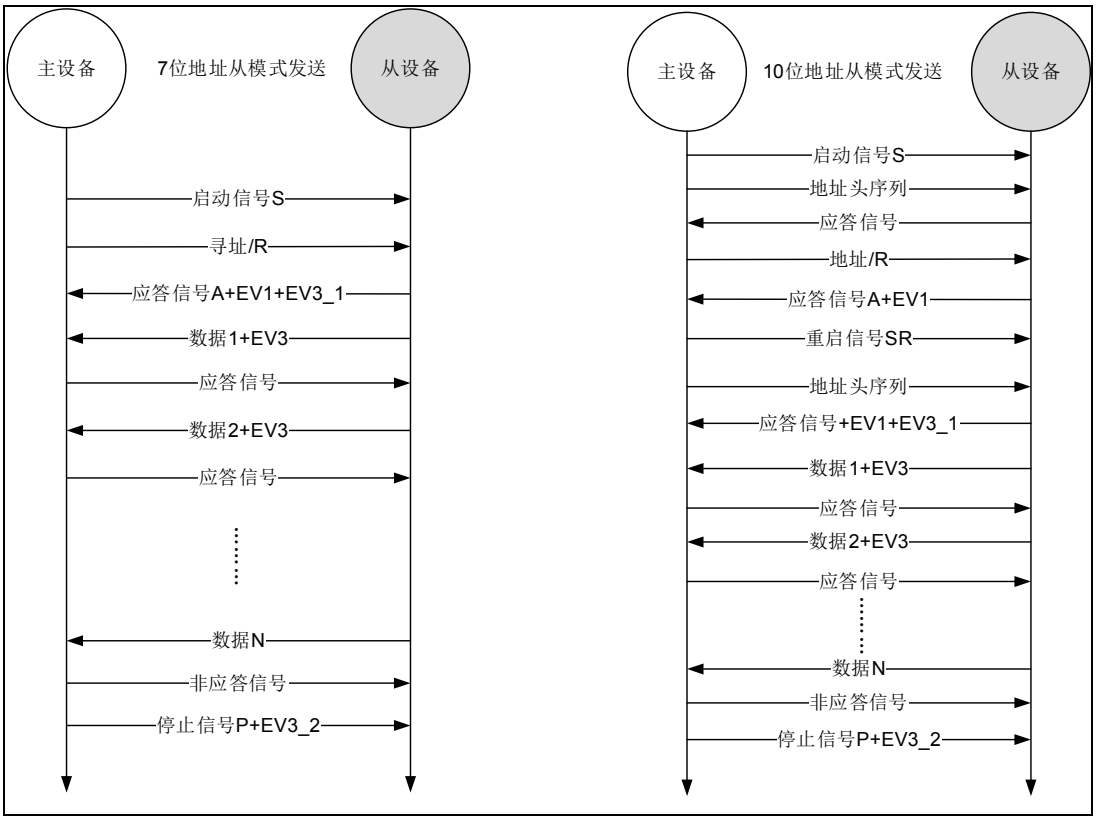


图 19-8 I2C 从发送器的传输序列图

注 1：S=起始位，SR=重复起始位，P=停止位，A=应答，NA=非应答

注 2：EVx=事件（如果 EVTIE=1 则发生中断）

注 3：EV1：当 I2C_STAT1.ADDR=1 时，需要先读 I2C_STAT1 再读 I2C_STAT2 寄存器来清零 I2C_STAT1.ADDR 位。

注 4：EV3_1：当 I2C_STAT1.TXBE=1 时，移位寄存器和数据寄存器都为空，此时可在 I2C_DATA 中写入数据 1。

注 5：EV3：当 I2C_STAT1.TXBE=1 时，移位寄存器非空，数据寄存器为空，需要对 I2C_DATA 进行写操作来清零。

注 6：EV3_2：当 ACKERR=1 时：需要对 I2C_STAT1.ACKERR 位写 0 来将其标志位清零。

注 7：EV1 和 EV3_1 事件可延长 SCL 低电平时间，直到相应的软件序列结束为止。

注 8：如果软件序列在下一个字节传输结束之前未能完成，EV3 事件会延长 SCL 低电平时间。

从接收器

在接收到匹配的地址时，I2C_STAT1.ADDR 位会置位，需要软件将 I2C_STAT1.ADDR 清零，从设备会通过内部移位寄存器接收 SDA 线中的字节并将其保存到 I2C_DATA 寄存器。每次接收到一个字节后，若使能了 ACK 功能，则从接收器会硬件发出一个应答信号，同时 I2C_STAT1.RXBNE 位会由硬件置 1。与发送类似，若用户希望使用中断接收方式，则可同时置位 I2C_CON2.EVTIE 和 I2C_CON2.BUFIE 位，每当 I2C_STAT1.RXBNE 置 1 时会产生一次中断。

发出应答脉冲（如果 I2C_CON1.ACKEN 位置 1）

I2C_STAT1.RXBNE 位会由硬件置 1 并在 I2C_CON2.EVTIE 和 I2C_CON2.BUFIE 位均置 1 时生成一个中断。

如果在下一次数据接收结束之前 I2C_STAT1.RXBNE 位已置 1 但 I2C_DATA 寄存器中的数据尚未读取，则 I2C_STAT1.BTC 位会置 1，而接口会一直延长 SCL 低电平，直到软件通过读取 I2C_DATA 寄存器来把 I2C_STAT1.BTC 清零。

如果 I2C_STAT1.RXBNE 被硬件置位，而在下一个数据接收结束之前，I2C_DATA 寄存器中的数据仍未被读取，则 I2C_STAT1.BTC 位会自动置位，而如果不将 I2C_STAT1.BTC 位清 0，则 SCL 线会被硬件一直拉低。I2C_STAT1.BTC 位清 0 的操作为先对 I2C_STAT1 寄存器进行一次读操作，再对 I2C_DATA 寄存器执行一次写操作，即可将 I2C_STAT1.BTC 位清 0。

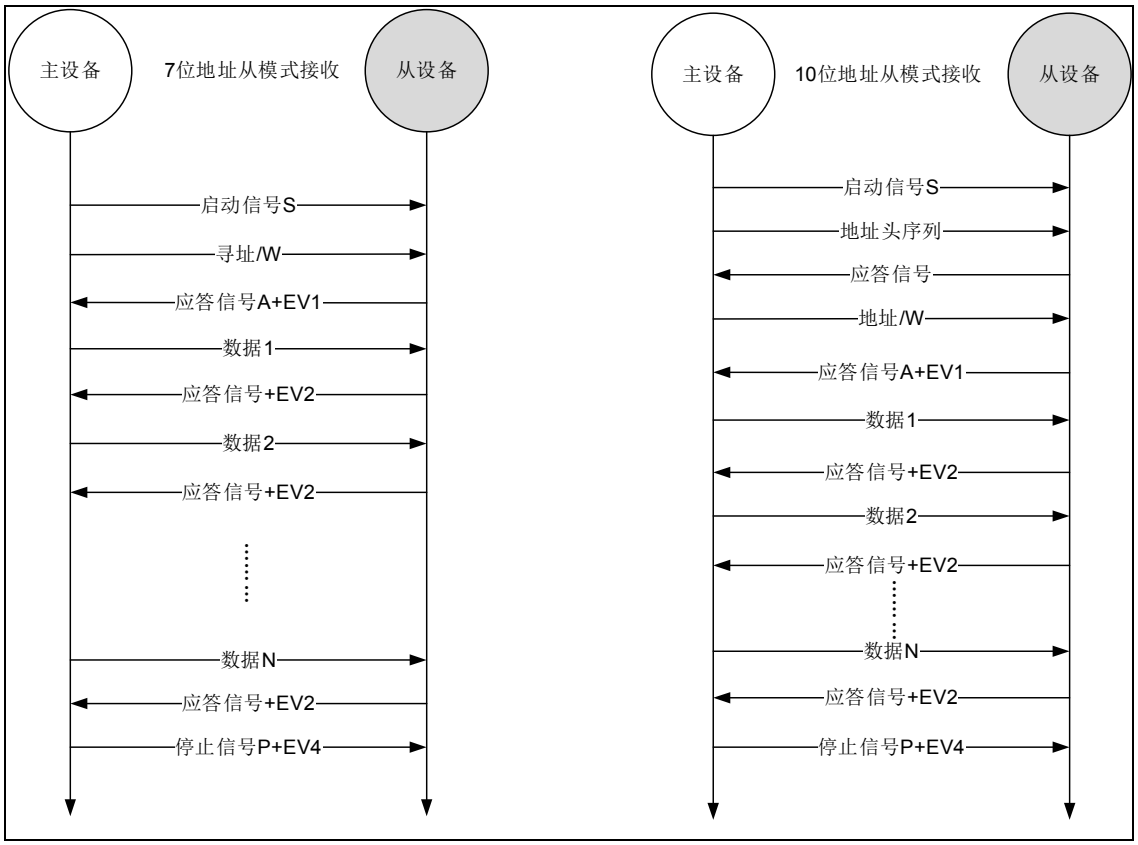


图 19-9 从接收器的传输序列图

注 1：S=起始位，SR=重复起始位，P=停止位，A=应答，NA=非应答

注 2：EVx=事件（如果 EVTIE=1 则发生中断）

注 3：EV1：当 I2C_STAT1.ADDR=1 时，需要先读 I2C_STAT1 再读 I2C_STAT2 寄存器来清零 I2C_STAT1.ADDR 位。

注 4：EV2：当 I2C_STAT1.RXBNE=1 时，需要读取 I2C_DATA 来将 I2C_STAT1.RXBNE 位清零。

注 5：EV4：当 DETSTP=1 时，需要先读 I2C_STAT1 寄存器，再写 I2C_CON1 寄存器来清零。

注 6：EV1 事件可延长 SCL 低电平时间，直到相应的软件序列结束为止。

注 7：如果软件序列在下一个字节接收结束之前未能完成，EV2 事件会延长 SCL 低电平时间。

注 8：检查了 I2C_STAT1 寄存器内容之后，用户应针对各个被置位的标志执行完整的清零序列。 对于 I2C_STAT1.ADDR 和 I2C_STAT1.DETSTP 标志，需要在 I2C 中断程序中执行以下序列： 对 I2C_STAT1 执行读操作 if （I2C_STAT1.ADDR == 1） {对 I2C_STAT1 执行读操作；对 I2C_STAT2 执行读操作} if （I2C_STAT1.DETSTP == 1） {对 I2C_STAT1 执行读操作；对 I2C_CON1 执行写操作}□这样做的目的是为了确保 I2C_STAT1.ADDR 和 I2C_STAT1.DETSTP 这两个标志都清零（如果二者均为被置位）。

传输完最后一个数据字节之后，主设备会生成一个停止位。接口会检测此条件并将 I2C_STAT1.DETSTP 位置 1 并在 I2C_CON2.EVTIE 位置 1 时生成一个中断。

通过先读取 I2C_STAT1 寄存器然后写入 I2C_CON1 寄存器的方式将 I2C_STAT1.DETSTP 位清零（请参见上图：从接收器的传输序列图 EV4）。

18. 4. 4 通信错误类型

当发生通信失败时，用户通过查询以下状态标志来判断发生了哪种错误类型。

总线错误 （BUSERR）

当 I2C 接口在传输地址或数据期间检测到外部停止位或起始位时，会出现此错误。该错误会产生以下动作：首先 I2C_STAT1.BUSERR 位置 1，若之前用户将 I2C_CON2.ERRIE 位置 1，则会生成一个中断。而若在从模式下，当由于检测到起始位发生的错误，从机会认为它是重复起始位，所以会等待主机传来地址或停止位；当由于检测到停止位发生的错误，从机会按停止位操作并释放总线。而若在主模式下，发生总线错误不会释放总线，并且也不会影响当前传输状态。

应答失败（ACKERR）

当接口未在应答位处检测到应答信号会出现此错误，该错误会产生以下动作：首先 I2C_STAT1.ACKERR 位置 1，若之前用户将 I2C_CON2.ERRIE 位置 1，则会生成一个中断。若发送器接收到 1 个 NACK 时，必须重新启动通信。主机复位通信需要软件设置生成一个停止位或重复起始位；从机复位通信需要硬件释放总线。

仲裁丢失 （LARB）

当 I2C 接口检测到仲裁丢失时会出现此错误。该错误会产生以下动作：首先 I2C_STAT1.LARB 位置 1，若之前用户将 I2C_CON2.ERRIE 位置 1，则会生成一个中断。然后硬件释放总线。接着 I2C 接口会自动切换成从模式。在本次仲裁丢失后，接口无法再响应它的从地址，但是能够在赢得仲裁的主机发出重复起始信号后，对自身地址做出响应。

上溢/下溢错误 （ROUERR）

当时钟延长已禁止且 I2C 接口正在接收数据时，从模式中可能出现上溢错误。接口已经收到一个字节（I2C_STAT1.RXBNE=1），但是收到下一个字节之前 I2C_DATA 中的数据未被读走。在这种情况下，最后一个字节被丢弃，若用户希望得到最后一个字节，需要软件将 I2C_STAT1.RXBNE 位清 0，发送器重新发送最后一次发送的数据。

下溢错误仅出现在发送端，当从机禁止时钟延长功能时，下一个字节的时钟信号到来时，从机还未将下一个需要发送的数据写入 I2C_DATA 寄存器中，则会发生下溢错误。由于未将需要发送的数据写入 I2C_DATA 寄存器中，所以从发送器会将之前的数据重复发送。若用户发现发生下溢错误时，接收端应将最后接收的重复数据给丢弃，而从发送器需要按照 I2C 的协议标准，在规定时间内发送正确的数据。

对于要传送的首个字节，必须在 I2C_STAT1.ADDR 清零之后且出现首个 SCL 上升沿之前将其写入 I2C_DATA 寄存器。否则，接收器必须丢弃首个数据。

18. 4. 5 SDA/SCL线控制

当使能时钟延长功能时，在一些情况下接口会强制将时钟线拉低。如在发送模式下，若出现 I2C_STAT1.TXBE=1 且 I2C_STAT1.BTC=1 时，接口会在发送数据之前一直将时钟线拉低，以等待用户将数据写入到 I2C_DATA 寄存器中。如在接收模式下，当 I2C_STAT1.RXBNE=1 且 I2C_STAT1.BTC=1，接口会在接收数据后一直将时钟线拉低，以等待用户从 I2C_DATA 寄存器中将数据读走。

当在从模式下禁止时钟延长功能时，可能由于发送数据未及时将数据写入 I2C_DATA 寄存器，或要接收数据未及时从 I2C_DATA 寄存器中读走数据，发生下溢或上溢错误。

注意：不会对写冲突进行管理。

18. 4. 6 SMBus

系统管理总线（SMBus）是一个双线制接口，各器件可通过它在彼此之间或者与系统的其余部分进行通信。它以 I2C 的工作原理为基础。SMBus 可针对系统和电源管理相关的任务提供控制总线。系统可使用 SMBus 与设备进行消息传递，而无需切换各个控制线。系统管理总线规范涉及三类器件。从器件用于接收或响应命令。主器件用于发出命令、生成时钟和中止传输。主机，专用的主器件，可提供连接系统 CPU 的主接口。主机必须具有主从设备功能，并且必须支持 SMBus 主机通知协议。系统中只允许存在一个主机。

通过访问系统管理总线，器件可以向用户提供制造商信息、告诉系统器件自身的型号或部件号、保存暂停事件的状态、报告不同的错误类型、接受控制参数并返回其状态。SMBus 可针对系统和电源管理相关的任务提供控制总线。

SMBus 特性

双线制总线协议（1 个时钟总线，1 个数据总线）和可选 SMBus 报警线

- ◇ 主从通信，主器件提供时钟
- ◇ 多主器件功能
- ◇ SMBus 数据格式与 I2C 的 7 位地址格式相似
- ◇ 通信速度范围 10KHz~100KHz
- ◇ 支持超时功能
- ◇ 通信端口电平为固定电平

器件标识

系统管理总线中作为从器件的任何器件均具有一个唯一地址，被称为从地址。为了提供一种机制来针对地址分配隔离各个器件，各器件必须具有唯一的器件标识符（UDID）。

地址解析协议（ARP）

每一个从器件，用户都可以动态地为其分配一个新的唯一的地址，这是为了解决 SMBus 从地址访问冲突的问题。地址协议（ARP）具有以下属性：

- ◇ 地址分配采用标准的 SMBus 物理层仲裁机制
- ◇ 器件通电时，分配的地址保持不变；器件断电时，也允许保留地址
- ◇ 完成地址分配后不会带来额外的 SMBus 数据包开销。（即，对已分配的从地址进行后续访问与访问固定地址的器件所产生的开销相同。）
- ◇ 任何 SMBus 主器件都可以遍历总线

SMBus 报警模式

SMBus 报警是带有中断线的可选信号，通过该模式可以扩展它们的控制能力，但是代价是需要消耗一个引脚。SMBA 信号是一种线与信号，与 SMBus 广播地址配合使用。使用 SMBus 调用的消息长度为 2 字节。

当一个从功能设备，希望与主机通信时，需要设置从设备的 I2C_CON1 寄存器上的 ALARM 位，此时会将 SMBA 引脚拉低，表示希望与主机通信。主机接收到此信号后会触发中断，通过处理该中断并通过报警响应地址（简称 ARA，其值为 0001 100X）同步访问所有 SMBA 器件。而此时，只有那些将 SMBA 引脚拉到低电平的器件会确认报警响应地址。通过 I2C_STAT1 寄存器中的 SMBALARM 状态标志可以确定这一状态。主机会执行修改后的接收字节操作。由从发送器件提供的 7 位器件地址被放置在字节的 7 个最高有效位。第 8 位可以是 0 或 1。

如果有不止一个器件将 SMBA 拉为低电平，则在从地址传输期间，具有最高优先级（最低位地址）的器件会通过标准仲裁获得通信权限。在确认从地址之后，器件必须释放 SMBA。如果消息传输结束后主机检测到 SMBA 仍为低电平，会再次读取 ARA。未实现 SMBA 信号的主机会定期访问 ARA。

超时错误

SMBus 和 I2C 之间存在一些定时规范方面的差异。SMBus 规定当时钟保持低电平超过 35ms，则视为 Timeout。另外，SMBus 还指定 Tlow：SEXT 作为从器件的累积时钟低电平延长时间。SMBus 指定 Tlow：MEXT 作为主器件的累积时钟低电平延长时间。

I2C_STAT1 寄存器中的 Timeout 或 Tlow 错误状态标志表明了此特性的状态。

SMBus 模式操作流程

要从 I2C 模式切换到 SMBus 模式，应执行以下步骤。

- ◇ 将 I2C_CON1 寄存器中的 PMOD 位置 1，设置为 SMBus 模式
- ◇ 根据应用的要求配置 I2C_CON1 寄存器中的 SMBMOD 和 ARPEN 位

如果要将某个器件配置为主器件，请按照主模式的起始位生成步骤进行操作。否则按照 I2C 从模式中的顺序操作。

该应用需要通过软件控制各种 SMBus 协议。

- ◇ ARPEN=1 且 SMBMOD=0 时使用 SMB 器件默认地址
- ◇ ARPEN=1 且 SMBMOD=1 时使用 SMB 主机头字段

SMBALARM=1 时使用 SMB 报警响应地址。

18. 4. 7 DMA请求

DMA 请求（在使能后）仅用于数据传输。当发送数据寄存器变空以及接收数据寄存器变满时会生成 DMA 请求。进行 I2C 数据传输之前，必须先初始化并使能 DMA。I2C_CON2.DMAEN 位必须在 ADDR 事件之前置 1。在主模式或从模式下，如果已使能时钟延长，I2C_CON2.DMAEN 位也可以在 ADDR 事件期间于 I2C_STAT1.ADDR 标志清零之前置 1。结束当前字节传输之前，必须发出 DMA 请求。当传输的数据量达到相应 DMA 通道编程设定的值时，DMA 控制器会发送一个结束传输信号给 I2C 接口，并生成一个传输完成中断（如果已使能）：

当用户希望 I2C 接口通信速度快，同时又想减轻 MCU 的负荷时，可以使用 DMA 方式进行数据的传输，可以设置在 I2C_STAT1.TXBE=1 或 I2C_STAT1.RXBNE=1 时触发 DMA 请求，DMA 从用户提供的地址中取值填入 I2C_DATA 寄存器中或从 I2C_DATA 寄存器中读取数据传到用户提供的地址中。在不同模式下，中断服务程序的处理方式有所不同：

◇ 在主发送模式下：在完成中断服务程序中，需要禁止 DMA 请求，然后等到 I2C_STAT1.BTC 事件发生后，发出停止信号（设置停止位）。

◇ 在主接收模式下：

- 当要接收的字节数等于或大于 2 时，DMA 控制器会在收到倒数第二个数据字节（第 N - 1 个数据时）发送一个硬件信号指示倒数第二个字节传输结束。I2C_CON2.LDMA 位置 1，I2C 会在 DMA 发出硬件指示信号后的下一个字节之后自动发送一个 NACK。用户可在 DMA 传输完成中断（如果已使能）程序中生成停止位。
- 当必须接收单个字节时：必须在 ADDR 事件期间于 I2C_STAT1.ADDR 标志清零之前对 I2C_CON1.ACKEN 进行编程，即当 I2C_STAT1.ADDR=1 时编程设定 I2C_CON1.ACKEN=0。接下来，用户可在 I2C_STAT1.ADDR 标志清零之后或者在执行 DMA 传输完成中断程序时编程设定停止位。

使用 DMA 进行发送

首先用户需要初始化 DMA，设置成 I2C_STAT1.TXBE=1 时触发 DMA 模式，配置完 DMA 后将 I2C 接口的 I2C_CON2.DMAEN 位置 1 以使能 DMA。当 I2C_STAT1.TXBE=1 时，会触发 DMA 控制器将数据从用户设定的存储区传入 I2C_DATA 寄存器中。具体要映射一个 DMA 通道来进行 I2C 发送，请按以下步骤操作：

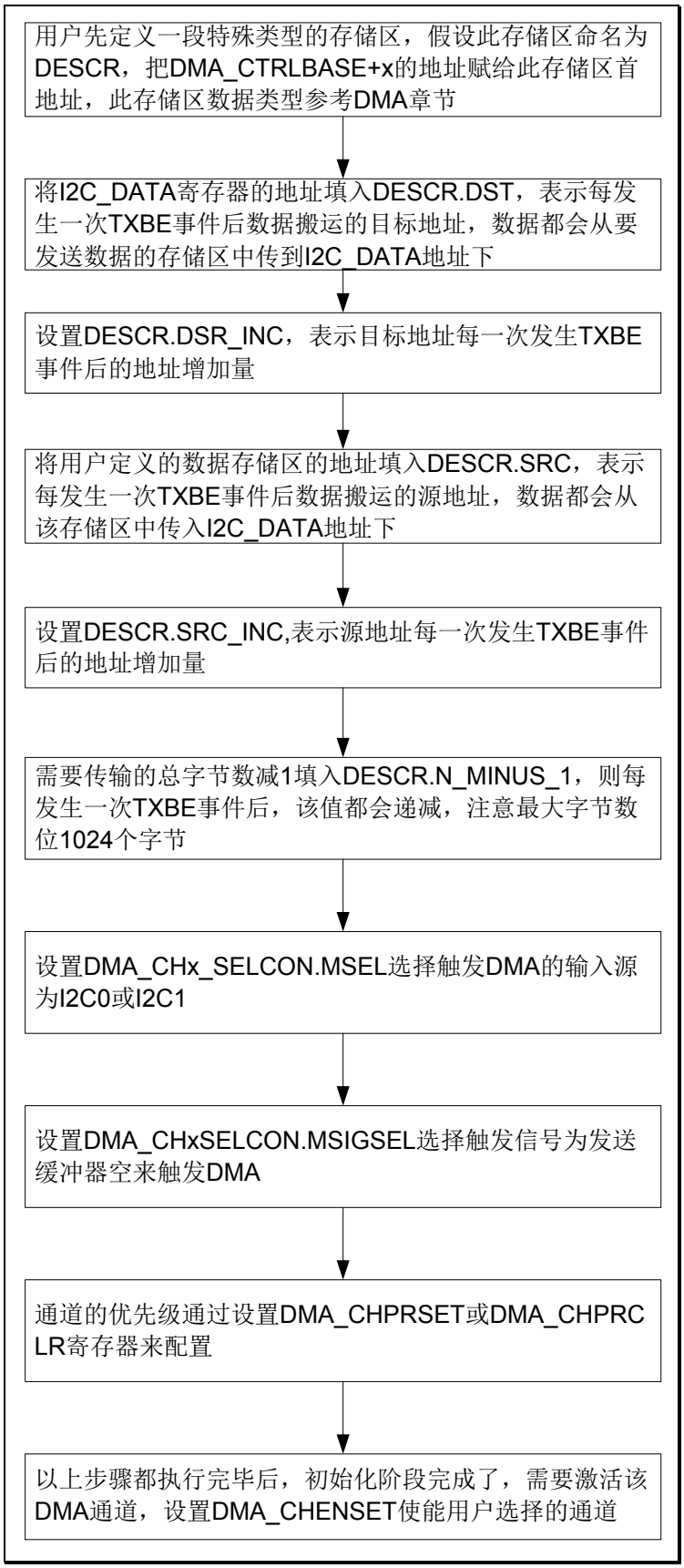


图 19-10 I2C 的 DMA 发送

当传输完成后，DMA 控制器发送一个传输完成信号给 MCU，而 DMA 会在 DMA 通道中断向量上生成一个中断（如果中断使能）。

注 1：x 表示 DMA 的通道号。

注 2：如果使用 DMA 进行传送，请勿使能 I2C_CON2 寄存器中的 BUFIE 位。

使用 DMA 进行接收

首先用户需要初始化 DMA，设置成 I2C_STAT1.RXBNE=1 时触发 DMA 模式，配置完 DMA 后将 I2C 接口的 I2C_CON2.DMAEN 位置 1 以使能 DMA。当 I2C_STAT1.RXBNE=1 时，会触发 DMA 控制器将数据从 I2C_DATA 寄存器中读取到用户设定的存储区里。具体要映射一个 DMA 通道来进行 I2C 接收，请按以下步骤操作：

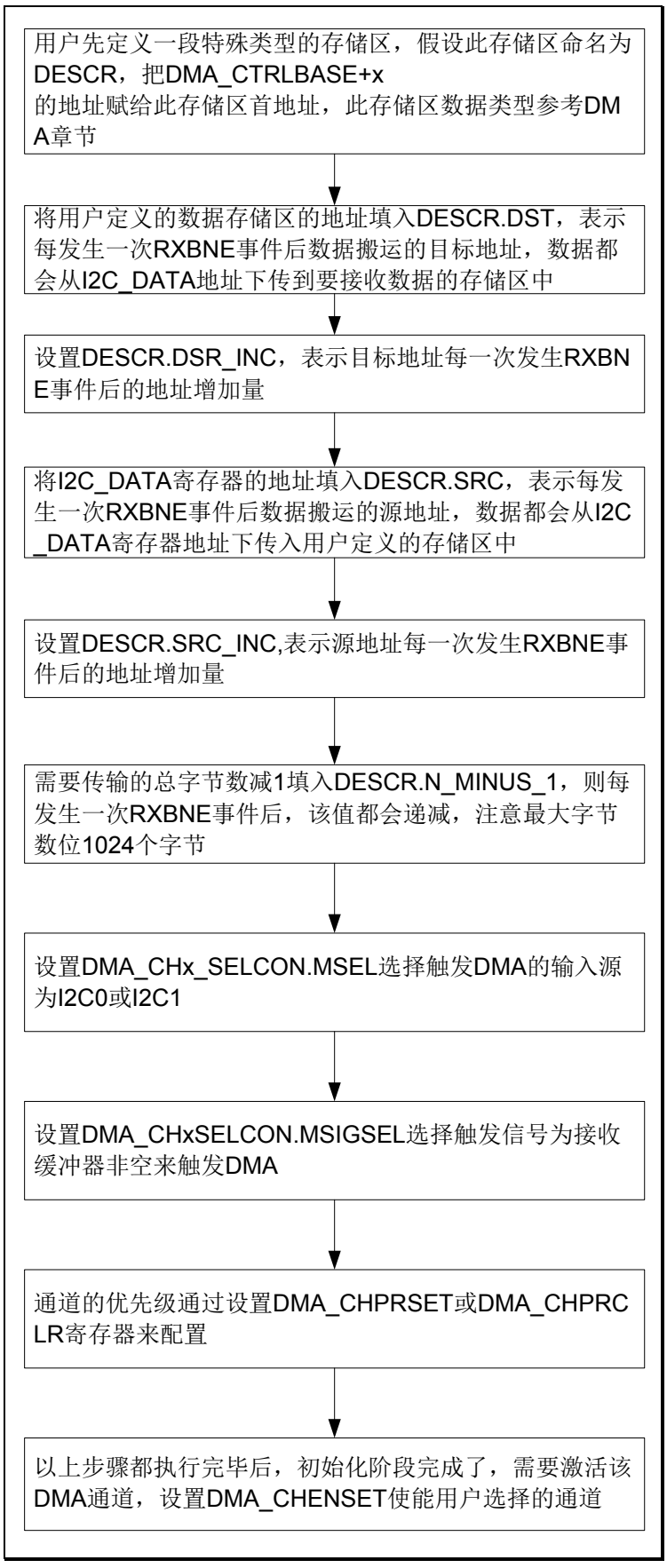


图 19-11 I2C 的 DMA 接收

当传输完成后，DMA 控制器发送一个传输完成信号给 MCU，而 DMA 会在 DMA 通道中断向量上生成一个中断（如果中断使能）。

注 1：x 表示 DMA 的通道号。
注 2：如果使用 DMA 进行传送，请勿使能 I2C_CON2 寄存器中的 BUFIE 位。

18. 4. 8 数据包错误校验

为了提高通信的可靠性，我们提供数据包错误校验功能（PEC），该计算原理是对各个位使用 CRC-8 多项式 C（x） = $x^8 + x^2 + x + 1$ 公式进行串行计算。具体使用方法如下：

◇ 将 I2C_CON1.PECEN 位置 1 即可使能 PEC 计算。PEC 是针对所有消息字节(包括地址和 R/W 位）的 CRC-8 计算。

- 在发送过程中：在与最后一个字节对应的接收事件发生后，将 I2C_CON1.PECEN 位置 1。PEC 会在最后一个传输的字节之后进行传送。
- 在接收过程中：在与最后一个字节对应的接收事件发生之后，将 I2C_CON1.PECEN 位置 1，以便接收器在接收到的下一个字节不等于内部计算的 PEC 时发送一个 NACK。在主接收器中，无论校验结果如何，PEC 后都将发送 NACK。在从模式下，必须在 CRC 接收的 ACK 之前设置 I2C_CON1.PECEN 位。

在主模式下，必须在复位 I2C_CON1.ACKEN 位时设置 I2C_CON1.PECEN 位。

◇ 若发生 PEC 计算错误，可检测 I2C_STAT1.PECERR 位，若使能了错误中断，发生此错误时会产生中断。

◇ 当使用 DMA 控制 I2C 接口的情况下，使能 PEC 计算，则会发生以下动作：

- 在发送过程中：当 I2C 接口接收来自 DMA 控制器的发送完成信号时，会在最后一个字节之后自动发送 PEC。
- 在接收过程中：当 I2C 接口接收来自 DMA 控制器的接收完成信号时，会自动将下一个字节视为 PEC 并对其进行校验。在 PEC 接收之后会生成一个 DMA 请求。

◇ 为了允许进行中间 PEC 传输，可以检测 I2C_CON2.LDMA 位来确定它是否真的是最后一个 DMA 传输。如果确实是主接收器的最后一个 DMA 请求，则会在接收最后一个字节后自动发送一个 NACK。

PEC 计算会因仲裁丢失而失效。

18. 4. 9 I2C中断

下表列出了 I2C 中断请求列表。

中断事件	事件标志	使能控制位
发送起始位（主模式）	SENDSTR	EVTIE
地址已发送（主模式）或地址匹配（从模式）	ADDR	
10 位地址的头段已发送（主模式）	SENDADD10	
已收到停止位（从模式）	DETSTP	
完成数据字节传输	BTC	
接收缓冲区非空	RXBNE	EVTIE 和 BUFIE
发送缓冲区为空	TXBE	
总线错误	BUSERR	ERRIE
仲裁丢失（主模式）	LARB	
应答失败	ACKERR	
上溢/下溢	ROUERR	
PEC 错误	PECERR	
超时/Tlow 错误	SMBTO	
SMBus 报警	SMBALARM	

表 19-1 I2C 中断请求

注 1：I2C_STAT1.SENDSTR、I2C_STAT1.ADDR、I2C_STAT1.SENDADD10、I2C_STAT1.DETSTP、I2C_STAT1.BTC、I2C_STAT1.RXBNE 和 I2C_STAT1.TXBE 通过逻辑或映射到同一个中断通道上。

注 2：I2C_STAT1.BUSERR、I2C_STAT1.LARB、I2C_STAT1.ACKERR、I2C_STAT1.ROUERR、I2C_STAT1.PECERR、I2C_STAT1.SMBTO 和 I2C_STAT1.SMBALARM 通过逻辑或映射到同一个中断通道上。

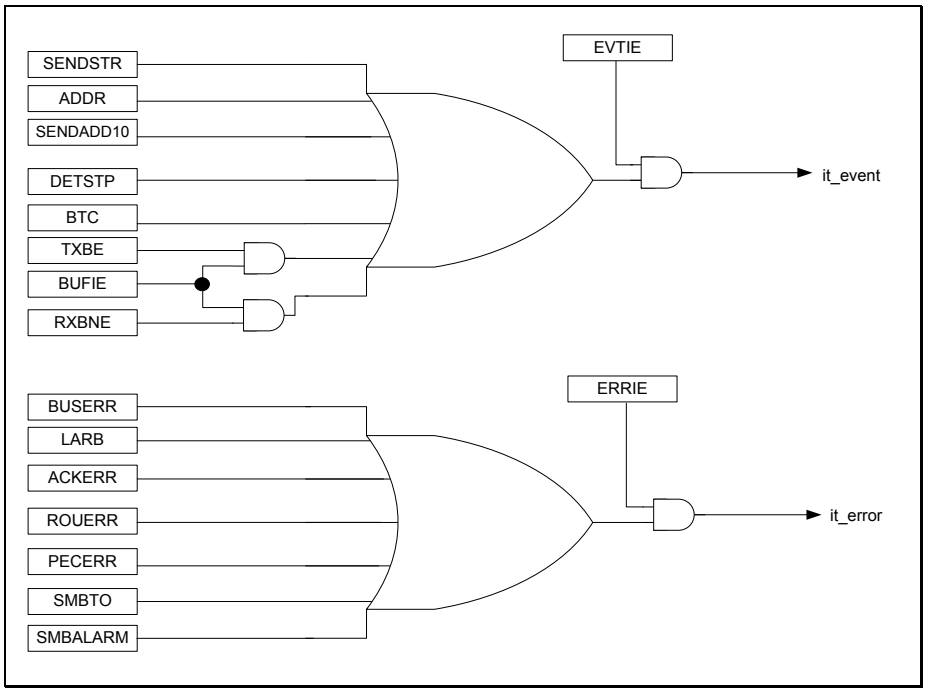


图 19-12 I2C 中断映射图

18. 4. 10 I2C调试模式

当微控制器进入调试模式时，SMBUS 超时会根据 DBGC 模块中的配置位选择继续正常工作或者停止工作。

18. 5 特殊功能寄存器

18. 5. 1 寄存器列表

寄存器可支持半字（16 位）或字（32 位）访问。

I2C 寄存器列表		
寄存器名称	偏移地址	寄存器描述
I2C_CON1	0000_H	I2C 控制寄存器 1
I2C_CON2	0004_H	I2C 控制寄存器 2
I2C_ADDR1	0008_H	I2C 自有地址寄存器 1
I2C_ADDR2	$000C_H$	I2C 自有地址寄存器 2
I2C_DATA	0010_H	I2C 数据寄存器
I2C_STAT1	0014_H	I2C 状态寄存器 1
I2C_STAT2	0018_H	I2C 状态寄存器 2
I2C_CKCFG	$001C_H$	I2C 时钟控制寄存器
I2C_RT	0020_H	I2C 上升时间寄存器

18. 5. 2 寄存器描述

18. 5. 2. 1 I2C控制寄存器 1（I2C_CON1）

I2C 控制寄存器 1 （I2C_CON1）

偏移地址：00_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																SRST	Reserved	ALARM	TRPEC	POSAP	ACKEN	STOP	START	DISCS	GCEN	PECEN	ARPEN	SMBMOD	Reserved	PMOD	PEN

Reserved	Bit 31-16	—	**保留**
SRST	Bit 15	R/W	**I2C 软件复位** 当置 1 时，I2C 处于复位状态。在复位此位之前，确保 I2C 线已释放且总线空闲。 0：I2C 外设未处于复位状态 1：I2C 外设处于复位状态 注意：当出现错误或锁定状态后，可使用此位重新初始化外设。例如，如果 BSYF 位已置 1 但因母线干扰而不能复位，则可使用 SRST 位退出此状态。
Reserved	Bit 14	—	**保留**
ALARM	Bit 13	R/W	**SMBus 报警** 此位由软件置 1 和清零，并可在 PEN=0 时由硬件清零。 0：释放 SMBA 引脚使其变成高电平。报警响应地址头后跟 NACK。 1：驱动 SMBA 引脚使其变成低电平。报警响应地址头后跟 ACK。
TRPEC	Bit 12	R/W	**传输数据包错误校验**此位由软件置 1 和清零，并可在 PEC 传输完成时由硬件清零，或者在 PEN=0 时或在检测到起 始位或停止位时由硬件清零。 0：不传输 PEC 1：PEC 传输（在 Tx 或 Rx 模式下） 注意：PEC 计算会因仲裁丢失而失效。
POSAP	Bit 11	R/W	**ACK/PEC 位置**（针对接收数据） 此位由软件置 1 和清零，并可在 PEN=0 时由硬件清零。 0：ACKEN 位控制移位寄存器中当前正在接收的字节的（N）ACK。TRPEC 位指示移位寄存器中的当前字节是一个 PEC。

			1：ACKEN 位控制移位寄存器中要接收的下一个字节的（N）ACK。TRPEC 位指示移位寄存器的下 一个字节是一个 PEC。 注意： POSAP 位只能用于主设备接收 2 个字节时。它必须在数据开始接收之前进行配置，如主接收器一节中建议的 2 字节接收步骤所述。
ACKEN	Bit 10	R/W	**接收应答使能** 此位由软件置 1 和清零，并可在 PEN=0 时由硬件清零。 0：不返回应答 1：在接收一个字节（匹配地址或数据）之后返回应答
STOP	Bit 9	R/W	**发送停止位** 该位由软件置 1 和清零，也可在检测到停止位时由硬件清零，在检测到超时错误时由硬件 置 1。 在主模式下： 0：不生成停止位。 1：在传输当前字节或发送当前起始位后生成停止位。 在从模式下： 0：不生成停止位。 1：完成当前字节传输后释放 SCL 和 SDA 线。
START	Bit 8	R/W	**发送起始位** 此位由软件置 1 和清零，并可在起始位发送完成后或 PEN=0 时由硬件清零。 在主模式下： 0：不生成起始位 1：生成重复起始位 在从模式下： 0：不生成起始位 1：在总线空闲时生成起始位
DISCS	Bit 7	R/W	**从模式下禁止时钟下拉等待** 在从模式下，当 ADDR 或 BTC 标志置 1 时，此位用于禁止时钟延长，直到软件将其复位为止。 0：使能时钟延长 1：禁止时钟延长
GCEN	Bit 6	R/W	**广播呼叫使能** 0：禁止广播呼叫。不对地址 00h 应答。 1：使能广播呼叫。对地址 00h 应答。
PECEN	Bit 5	R/W	**PEC 使能** 0：禁止 PEC 计算 1：使能 PEC 计算
ARPEN	Bit 4	R/W	**ARP 使能** 0：禁止 ARP □

			1：使能 ARP SMBMOD=0 时识别 SMBus 器件默认地址 SMBMOD=1 时识别 SMBus 主机地址
SMBMOD	Bit 3	R/W	**SMBus 模式** 0：SMBus 器件 1：SMBus 主机
Reserved	Bit 2	—	**保留**
PMOD	Bit 1	R/W	**外设模式** 0：I2C 模式 1：SMBus 模式
PEN	Bit 0	R/W	**模块使能** 0：禁止外设 1：使能外设 注意：如果此位在通信进行过程中清 0，在结束本次通信后会进入 IDLE 状态，外设被禁止。由于通信结束时 PEN=0，所有位均会复位。在主模式下，此位不能在通信结束之前清 0。

18. 5. 2. 2 I2C控制寄存器 2（I2C_CON2）

I2C 控制寄存器 2 （I2C_CON2）

偏移地址：04

复位值：00000000_00000000_00000000_00000000B

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																			LDMA	DMAEN	BUFIE	EVTIE	ERRIE	Reserved		CLKF						

Reserved	Bit 31-13	—	保留
LDMA	Bit 12	R/W	**末次 DMA 传输标志** 0：下一个 DMA 传输结束指示不是最后一次传输 1：下一个 DMA 传输结束指示是最后一次传输 注意：此位用于主接收模式，可对最后接收的数据生成 *NACK*。
DMAEN	Bit 11	R/W	**DMA 模式使能** 0：禁止 DMA 请求 1：当 TXBE=1 或 RXBNE =1 时使能 DMA 请求
BUFIE	Bit 10	R/W	**数据缓冲中断使能** 0：TXBE = 1 或 RXBNE = 1 时不生成任何中断。 1：TXBE = 1 或 RXBNE = 1 时生成事件中断（与 DMAEN 状态无关）
EVTIE	Bit 9	R/W	**事件中断使能** 0：事件中断禁止 1：事件中断使能 满足以下条件时将生成此中断： 1） SENDSTR = 1（主模式） 2） ADDR = 1（主/从模式） 3） SENDADD10= 1（主模式） 4） DETSTP = 1（从模式） 5） BTC = 1，无 TXBE 或 RXBNE 事件 6） BUFIE = 1 且 TXBE 事件置 1 7） BUFIE = 1 且 RXBNE 事件置 1
ERRIE	Bit 8	R/W	**错误中断使能** 0：错误中断禁止 1：错误中断使能 满足以下条件时将生成此中断： 1） BUSERR = 1 2） LARB = 1 3） ACKERR = 1 4） ROUERR = 1 5） PECERR = 1

			6） SMBTO = 1 **7**） SMBALARM = 1
Reserved	Bit 7-6	—	**保留**
CLKF	Bit 5-0	R/W	**外设时钟频率** 外设时钟频率必须使用 APB 时钟频率进行配置（I2C 外设连接到 APB）。允许的最小频率为 2 MHz，最大频率则受限于 APB 最大频率 （52 MHz）。 000000：不允许 000001：不允许 000010：2MHz 000011：3MHz... 110000：48MHz 110001：49MHz 110010：50MHz 110011：51MHz 110100：52MHz 大于 110100：不允许

18. 5. 2. 3 I2C自有地址寄存器 1（I2C_ADDR1）

I2C 自有地址寄存器 1 （I2C_ADDR1）																															
偏移地址：08																															
复位值：00000000_00000000_00000000_00000000$_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																ADDTYPE	Reserved					ADDH[9:8]		ADD[7:1]							ADDLSB

Reserved	Bit 31-16	—	保留
ADDTYPE	Bit 15	R/W	**从模式寻址类型** 0：7 位从地址（无法应答 10 位地址） 1：10 位从地址（无法应答 7 位地址）
Reserved	Bit 14-10	R/W	保留
ADDH[9:8]	Bit 9-8	R/W	**10 位从模式地址 8-9 位** 7 位寻址模式：无意义 10 位寻址模式：地址的第 9:8 位
ADD[7:1]	Bit 7-1	R/W	**10 位从模式地址第 1-7 位或 7 位从模式地址** 地址的第 7:1 位
ADDLSB	Bit 0	R/W	**10 位从模式地址第 0 位** 7 位寻址模式：无意义 10 位寻址模式：地址的第 0 位

18. 5. 2. 4 I2C自有地址寄存器 2（I2C_ADDR2）

I2C 自有地址寄存器 2 （I2C_ADDR2）																															
偏移地址：0C																															
复位值：00000000_00000000_00000000_00000000$_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																								ADD[7:1]							DUALEN

Reserved	Bit31-8	—	保留
ADD[7:1]	Bit7-1	R/W	**接口地址 7 位从模式地址** 双寻址模式下的地址第 7:1 位
DUALEN	Bit0	R/W	**10 位和 7 位寻址模式使能** 0：7 位寻址模式下仅对 ADDR1 地址响应 1：7 位寻址模式下能对 ADDR1 和 ADDR2 两个地址响应

18. 5. 2. 5 I2C数据寄存器（I2C_DATA）

I2C 数据寄存器 （I2C_DATA）																															
偏移地址：10_H																															
复位值：$00000000\_00000000\_00000000\_00000000_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																								TRBUF							

Reserved	Bit 31-8	—	保留
TRBUF	Bit 7-0	R/W	**数据接收和发送缓冲区** 接收的字节或者要发送到总线的字节。 发送模式：在 DATA 寄存器中写入第一个字节时自动开始发送字节。如果在启动传送 （TXBE=1）后立即将下一个要传送的数据置于 DATA 中，则可以保持连续的传送流 接收模式：将接收到的字节复制到 DATA 中（RXBNE=1）。如果在接收下一个数据字节（RXBNE=1）之前读取 DATA，则可保持连续的传送流。 注意：在从模式下，地址并不会复制到 DATA 中。 注意：硬件不对写冲突进行管理（TXBE=0 时也可对 DATA 执行写操作）。 注意：如果发出 ACK 脉冲时出现 LARB 事件，则不会将接收到的字节复制到 DATA 寄存器，因而也无法读取字节。

18. 5. 2. 6 I2C状态寄存器 1（I2C_STAT1）

I2C 状态寄存器 1 （I2C_STAT1）

偏移地址：14_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																SMBALARM	SMBTO	Reserved	PECERR	ROUERR	ACKERR	LARB	BUSERR	TXBE	RXBNE	Reserved	DETSTP	SENDADD10	BTC	ADDR	SENDSTR

Reserved	Bit 31-16	—	**保留**
SMBALARM	Bit 15	R/C_W0	**SMBus 报警标志** 在 SMBus 主机模式下： 0：无 SMBALARM 1：引脚上发生 SMBALARM 事件 在 SMBus 从模式下： 0：无 SMBALARM 响应地址头 1：接收到指示 SMBALARM 低电平的 SMBALARM 响应地址头 该位由软件写入 0 来清零，或在 PEN=0 时由硬件清零。
SMBTO	Bit 14	R/C_W0	**SMBus 时钟下拉超时标志** 0：无超时错误 1：SCL 低电平时长持续 25 ms（超时）或主器件累计时钟低电平延长时间超过 10 ms （Tlow:mext） 或从器件累计时钟低电平延长时间超过 25 ms （Tlow:sext） 在从模式下置 1 时，从器件复位通信且硬件释放数据线；在主模式下置 1 时，由硬件发送停止位；该位由软件写入 0 来清零，或在 PEN=0 时由硬件清零。 注意： 此功能仅在 *SMBus* 模式下可用。
Reserved	Bit 13	—	保留
PECERR	Bit 12	R/C_W0	**接收模式下 PEC 错误** 0：无 PEC 错误：接收器在接收 PEC 后返回 ACK（如果 ACKEN=1） 1：PEC 错误：接收器在接收 PEC 后返回 NACK（无论 ACK 什么值） 该位由软件写入 0 来清零，或在 PEN=0 时由硬件清零。 注意：接收到错误的 CRC 时，如果在结束 CRC 接收之前 PECEN 控制位没有置 1，则 PECERR 位在从模式下不会置 1。不过可以通过读取 PECV 值来判定接收到的 CRC 是否正确。
ROUERR	Bit 11	R/C_W0	**接收溢出错误**

			0：未发生上溢/下溢 1：上溢或下溢 在 DISCS=1 时，从模式下由硬件置 1，且： 1）接收过程中接收到一个新字节（包括 ACK 脉冲）但尚未读取 DATA 寄存器。新接收的字节将丢失。 2）发送过程中将发送一个新字节但尚未向 DATA 寄存器写入数据。同一字节发送两次。 该位由软件写入 0 来清零，或在 PEN=0 时由硬件清零。 注意： 如果 *DATA* 写操作时间与出现 SCL 上升沿的时间非常接近，则发出的数据不确定，并且出现数据保持时间错误。
ACKERR	Bit 10	R/C_W0	**发送应答错误** 0：未发生应答失败 1：应答失败 无应答返回时由硬件置 1。该位由软件写入 0 来清零，或在 PEN=0 时由硬件清零。
LARB	Bit 9	R/C_W0	**总线仲裁丢失** 0：未检测到仲裁丢失 1：检测到仲裁丢失 当接口在竞争总线时输给另一个主设备，由硬件将该位置 1 该位由软件写入 0 来清零，或在 PEN=0 时由硬件清零。 发生 LARB 事件后，接口会自动切换回从模式（MASTER=0）。 注意：在 SMBUS 中，从模式下的数据仲裁仅发生在数据阶段或发送确认期间（不适用于地址确认）。
BUSERR	Bit 8	R/C_W0	**总线错误** 0：无误放的起始或停止位 1：存在误放的起始或停止位 SCL 为高电平时，若接口在字节传输期间检测到某个无效位置出现 SDA 上升沿或下降沿，则会由硬件将该位置 1。 该位由软件写入 0 来清零，或在 PEN=0 时由硬件清零。
TXBE	Bit 7	R	**数据发送缓冲器空** 0：数据寄存器非空 1：数据寄存器为空 发送过程中 DATA 为空时该位置 1。TXBE 不会在地址阶段置 1。该位由软件写入 DATA 寄存器来清零，或在出现起始、停止位或者 PEN=0 时由硬件清零。 如果接收到 NACK 或要发送的下一个字节为 PEC（PECEN=1），TXBE 将不会置 1 注意： 写入第一个要发送的数据或在 BTC 置 1 时写入数据都无法将 TXBE 清零，因为这两种情况下数据寄存器仍为空。
RXBNE	Bit 6	R	**数据接收缓冲器满**

			0：数据寄存器为空 1：数据寄存器非空 接收模式下数据寄存器非空时置 1。RXBNE 不会在地址阶段置 1。该位由软件读取或写入 DATA 寄存器来清零，或在 PEN=0 时由硬件清零。发生 LARB 事件时 RXBNE 不会置 1。 注意： BTC 置 1 时无法通过读取数据将 RXBNE 清零，因为此时数据寄存器仍为满。
Reserved	Bit 5	—	保留
DETSTP	Bit 4	R	**总线检测到停止位信号** 0：未检测到停止位 1：检测到停止位 从设备在应答脉冲后（如果 ACKEN=1）检测到停止位，由硬件置 1。该位由软件分别对 STAT1 寄存器和 CON1 寄存器执行读操作和写操作来清零，或在 PEN=0 时由硬件清零。 注意：收到 NACK 后 DETSTP 位不会置 1。建议在 DETSTP 置 1 后执行完整的清零序列（首先读取 STAT1，然后写入 CON1）。
SENDADD10	Bit3	R	**发送 10 位地址头序（主模式）** 0：未发生 SENDADD10 事件。 1：主器件已发送第一个地址字节（头）。 主器件在 10 位地址模式下已发送第一个字节时由硬件置 1。该位由软件在读取 STAT1 寄存器后在 DATA 寄存器中写入第二个地址字节来清零，或在 PEN=0 时由硬件清零。 注意：收到 NACK 后 SENDADD10 位不会置 1
BTC	Bit 2	R	**字节传输完成** 0：数据字节传输未完成 1：数据字节传输成功完成 DISCS=0 时，该位由硬件置 1，且： 1）接收过程中接收到一个新字节（包括 ACK 脉冲）但尚未读取 DATA 寄存器（RXBNE=1）。 2）发送过程中将发送一个新字节但尚未向 DATA 寄存器写入数据（TXBE=1）。 该位由软件读或写 DATA 寄存器来清零，或在发送过程中出现起始或停止位后由硬件清零，也可以在 PEN=0 时由硬件清零。 注意：收到 NACK 后 BTC 位不会置 1 如果下一个要发送的字节为 PECI2C_STAT2 寄存器中的 TRF=1，I2C_CON1 寄存器中的 PECEN=1），则 BTC 位不会置 1
ADDR	Bit 1	R	**地址已发送（主模式）/地址匹配（从模式）** 由软件在读取 STAT1 寄存器后读取 STAT2 寄存器

			来清零，或在 PEN=0 时由硬件清零。 地址匹配（从模式）： 0：地址不匹配或未接收到地址。 1：接收到的地址匹配。 当接收到的从地址与 ADDR 寄存器内容、广播呼叫地址或 SMBus 器件默认地址匹配时，或者识别到 SMBus 主机或 SMBus 报警时，该位由硬件置 1。 注意：在从模式下，建议在 ADDR 置 1 后执行完整的清零序列（首先读取 STAT1，然后写入 STAT2）。 地址已发送（主模式） 0：地址发送未结束 1：地址发送结束 在 10 位寻址模式下，接收到第二个地址字节的 ACK 后该位置 1。在 7 位寻址模式下，接收到地址字节的 ACK 后该位置 1。 注意： 收到 NACK 后 ADDR 位不会置 1
SENDSTR	Bit 0	R	**发送起始位信号（主模式）** 0：无起始位 1：起始位已经发送。 生成启动条件时该位置 1。该位由软件在读取 STAT1 寄存器后写入 DATA 寄存器来清零，或在 PEN=0 时由硬件清零

18. 5. 2. 7 I2C状态寄存器 2（I2C_STAT2）

I2C 状态寄存器 2 （I2C_STAT2）

偏移地址：18_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																PECV								DMF	SMBHH	SMBDEF	RXGCF	Reserved	TRF	BSYF	MASTER

Reserved	Bit 31-16	—	**保留**
PECV	Bit 15-8	R	**数据包错误校验值** PECEN=1 时，此寄存器包含内部 PEC。
DMF	Bit 7	R	**双寻址模式地址匹配标志（从模式）** 0：接收到的地址与 ADDR1 匹配 1：接收到的地址与 ADDR2 匹配 出现停止位、重复起始位或 PEN=0 时由硬件清零。
SMBHH	Bit 6	R	**SMBus 主机地址头序列（从模式）** 0：无 SMBus 主机地址 1：SMBMOD=1 且 ARPEN=1 时接收到 SMBus 主机地址。 出现停止位、重复起始位或 PEN=0 时由硬件清零。
SMBDEF	Bit 5	R	**SMBus 器件默认地址（从模式）** 0：无 SMBus 器件默认地址 1：ARPEN =1 时接收到 SMBus 器件默认地址 出现停止位、重复起始位或 PEN=0 时由硬件清零。
RXGCF	Bit 4	R	**从模式广播地址接收标志** 0：无广播呼叫 1：GCEN=1 时接收到广播呼叫地址 出现止位、重复起始位或 PEN=0 时由硬件清零。
Reserved	Bit 3	-	**保留**
TRF	Bit 2	R	**发送或接收模式** 0：接收器 1：发送器 此位在整个地址阶段的结尾处根据地址字节的 R/W 位状态进行置 1。同样，检测到停止位（DETSTP=1）、重复起始位、总线仲裁丢失（LARB=1）或当 PEN=0 时该位也由硬件清零。
BSYF	Bit 1	R	**总线忙碌** 0：总线上无通信 1：总线正在进行通信 检测到 SDA 或 SCL 低电平时由硬件置 1，检测到停止位时由硬件清零。

			该位指示总线上是否正在进行通信。即使接口禁止（PEN=0）后此信息也会更新。
MASTER	Bit 0	R	**主模式选择** 0：从模式 1：主模式 接口进入主模式时（SENDSTR=1）由硬件置 1。检测到总线上的停止位、仲裁丢失（LARB=1）或当 PEN=0 时由硬件清零。

注意：读取 I2C_STAT1 后再读取 I2C_STAT2 可将 ADDR 标志清零，即使 ADDR 标志在读取 I2C_STAT1 之后置 1 也如此。因此，必须仅在 I2C_STAT1 中的 ADDR 位已置 1 或者 DETSTP 位已清零后读取 I2C_STAT2。

18. 5. 2. 8 I2C时钟控制寄存器（I2C_CKCFG）

I2C 时钟控制寄存器 （I2C_CKCFG）

偏移地址：$1C_H$

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-16	—	**保留**
CLKMOD	Bit 15	R/W	**I2C 主模式时钟速度选择** 0：标准模式 1：快速模式
DUTY	Bit 14	R/W	**快速模式占空比** 0：快速模式 tlow/thigh = 2 1：快速模式 tlow/thigh = 16/9（参见 CKCFG）
Reserved	Bit 13-12	—	**保留**
CLKSET	Bit 11-0	R/W	**快速/标准模式下的时钟控制寄存器（主模式）** 控制主模式下的 SCL 时钟。 标准模式或 SMBus 模式： Thigh = CKCFG * TPCLK1 Tlow = CKCFG * TPCLK1 快速模式： 如果 DUTY = 0： Thigh = CKCFG * TPCLK1 Tlow = 2 * CKCFG * TPCLK1 如果 DUTY = 1：（达到 400 kHz） Thigh = 9 * CKCFG * TPCLK1 Tlow = 16 * CKCFG * TPCLK1 例如：要在标准模式下生成 100 kHz 的 SCL 频率： 如果 CLKF = 0A，TPCLK1 = 100 ns，则必须将 CKCFG 编程为 0x32（ 50×100 ns = 5000 ns）。 注意： 1.CKCFG 寄存器须 I2C（PEN = 0）的情况下配置。 2.允许的最小值为 0x04，但快速占空比模式例外，其最小值为 0x01。这些时间均未经过滤波。

18. 5. 2. 9 I2C上升时间寄存器（I2C_RT）

I2C 上升时间寄存器（I2C_RT）

偏移地址：20_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-6	—	保留
RISET	Bit 5-0	R/W	**主模式下允许的最大上升时间** 这些位必须编程为 I2C 总线规范中给定的最大 SCL 上升时间加 1。 例如：标准模式下允许的最大 SCL 上升时间为 1000 ns。如果 I2C_CON2 寄存器中 CLKF<5:0> 位的值等于 0x0A 且 T_{PCLK1} = 100ns，则 RISET<5:0> 位必须编程为 0Bh。（1000ns/ 100 ns = 10 + 1）滤波器值也可以叠加到 RISET<5:0>。 注意：RISET<5:0>须在 I2C（PEN = 0）禁止的情况下配置，且必须编程为整数。

第19章 串行外设接口（SPI）

19. 1 概述

串行外设接口（SPI）可与外部器件进行半双工/全双工的同步串行通信。该接口可配置为主模式，在这种情况下，它可为外部从器件提供通信时钟（SCK）。该接口还能够在多主模式配置下工作。

此接口能运行多种模式，如基于双线的单工同步传输，其中一条线用于双向数据传输，同时可用 CRC 校验提高通信可靠性。

19. 2 特性

- 基于三条线的全双工同步传输
- 基于双线的单工同步传输，其中一条可作为双向数据线
- 8 位或 16 位帧格式选择
- 主模式或从模式操作
- 多主模式功能
- 8 个主模式波特率预分频器（最大值为 fPCLK/2）
- 从模式频率（最大值为 fPCLK/2）
- 对于主模式和从模式都可实现更快的通信
- 对于主模式和从模式都可通过硬件或软件进行 NSS 控制：动态切换主/从操作
- 时钟极性和相位可编程
- 数据顺序可编程，如最先移位 MSB 或 LSB
- 具有专用的发送或接收标志可触发中断
- 具有显示 SPI 总线忙状态标志
- 用于确保可靠通信的硬件 CRC 功能：
  - 在发送模式下可将 CRC 值作为最后一个字节数据发送
  - 根据收到的最后一个字节自动进行 CRC 错误校验
- 模式错误、上溢和 CRC 错误标志均可触发中断事件
- 具有 DMA 功能的 1 字节发送和接收缓冲器：发送和接收请求

19. 3 结构框图

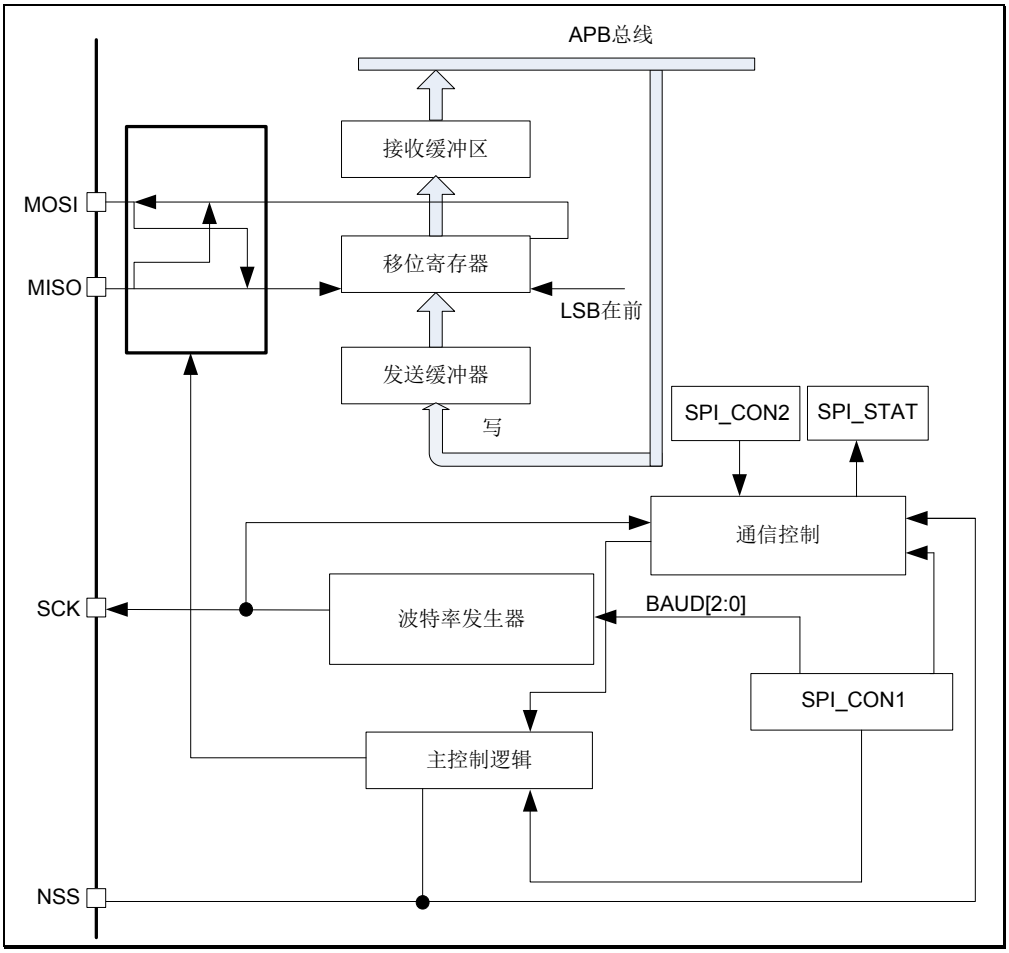


图 20-1 SPI 电路结构框图

19. 4 功能描述

19. 4. 1 功能概述

图 20-2 给出了单个主器件和单个从器件之间的互连的基本示例。

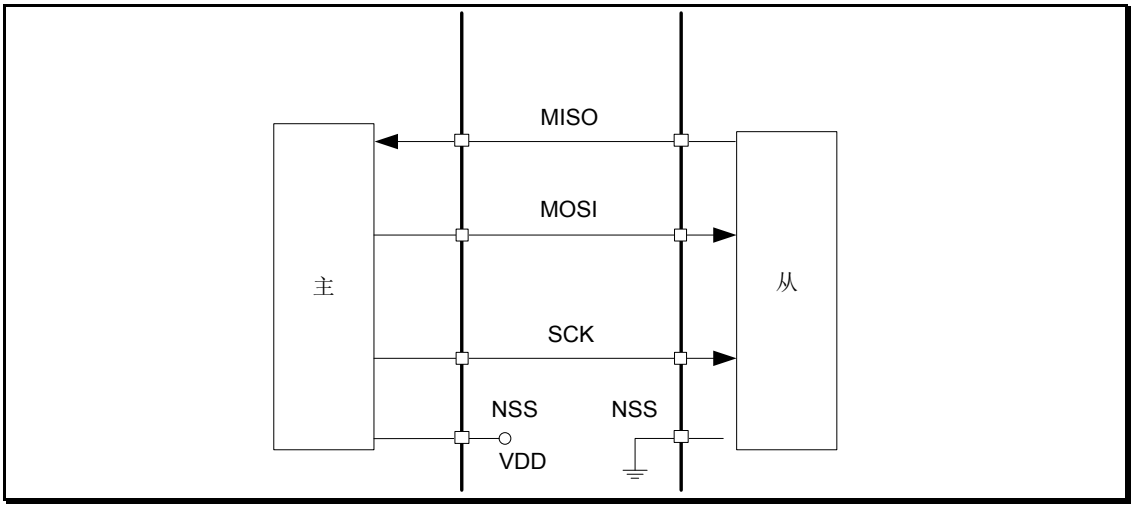


图 20-2 单个主器件/单个从器件应用

注 1：上图中，NSS 引脚配置为输入。
注 2：如果 NSS 由软件控制则不使用。

通常，SPI 通过 4 个引脚与外部器件连接：

- ◇ MISO：主输入/从输出数据引脚。
- ◇ MOSI：主输出/从输入数据引脚。
- ◇ SCK：用于 SPI 主器件的串行时钟输出以及 SPI 从器件的串行时钟输入。
- ◇ NSS：从器件选择。这是用于选择从器件的可选引脚。此引脚用作“片选”，可让 SPI 主器件与从器件进行单独通信，从而并避免数据线上的竞争。从器件的 NSS 输入可由主器件上的标准 IO 端口驱动。NSS 引脚在使能（SPI_CON2.NSSOE 位）时还可用作输出，并可在 SPI 处于主模式配置时驱动为低电平。通过这种方式，只要器件配置成 NSS 硬件控制模式，所有连接到该主器件 NSS 引脚的其它器件 NSS 引脚都将呈现低电平，并因此而作为从器件。当配置为主模式，且 NSS 配置为输入（SPI_CON1.MSTREN=1 且 SPI_CON2.NSSOE=0）时，如果 NSS 拉至低电平，SPI 将进入模式错误状态：SPI_CON1.MSTREN 位自动清零，并且器件配置为从模式（参见章节：错误标志）

MOSI 引脚连接在一起，MISO 引脚连接在一起。通过这种方式，主器件和从器件之间以串行方式传输数据（最高有效位在前）。

通信始终由主器件发起。当主器件通过 MOSI 引脚向从器件发送数据时，从器件同时通过 MISO 引脚发出准备好的数据。这是一个数据输出和数据输入都由同一时钟进行同步的全双工通信过程，时钟信号由主器件的 SCK 引脚发出提供给从器件。

从器件选择（NSS）引脚管理

可以使用 SPI_CON1.SSEN 位设置硬件或软件控制从器件选择。

◇ 软件控制 NSS（SPI_CON1.SSEN = 1）从器件选择信息在内部由 SPI_CON1.SSOUT 位的值驱动。

◇ 硬件管理 NSS（SPI_CON1.SSEN = 0）根据 NSS 输出配置（SPI_CON2.NSSOE 位），硬件管理 NSS 有两种模式。

- NSS 输出使能（SPI_CON1.SSEN = 0，SPI_CON2.NSSOE = 1）仅当器件在主模式下工作时才使用此配置。当主器件开始通信时，NSS 信号驱动为低电平，并保持到 SPI 被关闭为止。
- NSS 输出禁止（SPI_CON1.SSEN = 0，SPI_CON2.NSSOE = 0）对于在主模式下工作的器件，此配置允许多主模式功能。对于设置为从模式的器件，NSS 引脚用作传统 NSS 输入：在 NSS 为低电平时片选该从器件，在 NSS 为高电平时取消对它的片选。

时钟相位和时钟极性

通过配置 SPI_CON1.CPOL 和 SPI_CON1.CPHA 位，可以用软件选择四种可能的时序关系。SPI_CON1.CPOL（时钟极性）位控制空闲时时钟线上的电平状态，此位对主器件和从器件都有作用。如果复位 SPI_CON1.CPOL 位，SCK 引脚在空闲状态时处于低电平。如果将 SPI_CON1.CPOL 位置 1，SCK 引脚在空闲状态时处于高电平。

如果将 SPI_CON1.CPHA 位置 1，则 SCK 引脚上的第二个边沿（如果 SPI_CON1.CPOL 位配置为 0，则为下降沿；如果 SPI_CON1.CPOL 位配置为 1，则为上升沿）对 MSB 采样。即，在第二个时钟边沿锁存数据。如果复位 CPHA 位，则 SCK 引脚上的第一个边沿（如果 SPI_CON1.CPOL 位配置为 0，则为下降沿；如果 SPI_CON1.CPOL 位配置为 1，则为上升沿）对 MSB 采样。即，在第一个时钟边沿锁存数据。

用户通过组合 SPI_CON1.CPOL 和 SPI_CON1.CPHA 位来选择数据捕获的时钟边沿。

下图显示了在 SPI_CON1.CPHA 和 SPI_CON1.CPOL 位的四种组合下的 SPI 传输。可以将该图解释为主器件或从器件时序图，其中 SCK 引脚、MISO 引脚、MOSI 引脚直接连接在主器件和从器件之间。

注：在切换 SPI_CON1.CPOL/SPI_CON1.CPHA 位之前，必须通过复位 SPI_CON1.SPIEN 位来关闭 SPI。

必须以同一时序模式对主器件和从器件进行编程。

主器件和从器件的时序需要配置成相同，通信才能正常。

SCK 的空闲状态必须与 SPI_CON1 寄存器中选择的极性相对应（如果 SPI_CON1.CPOL=1，则上拉 SCK；如果 SPI_CON1.CPOL=0，则下拉 SCK）。

通过 SPI_CON1.FLEN 位选择数据帧长度（8 或 16 位），该格式决定了发送/接收过程中的数据长度。

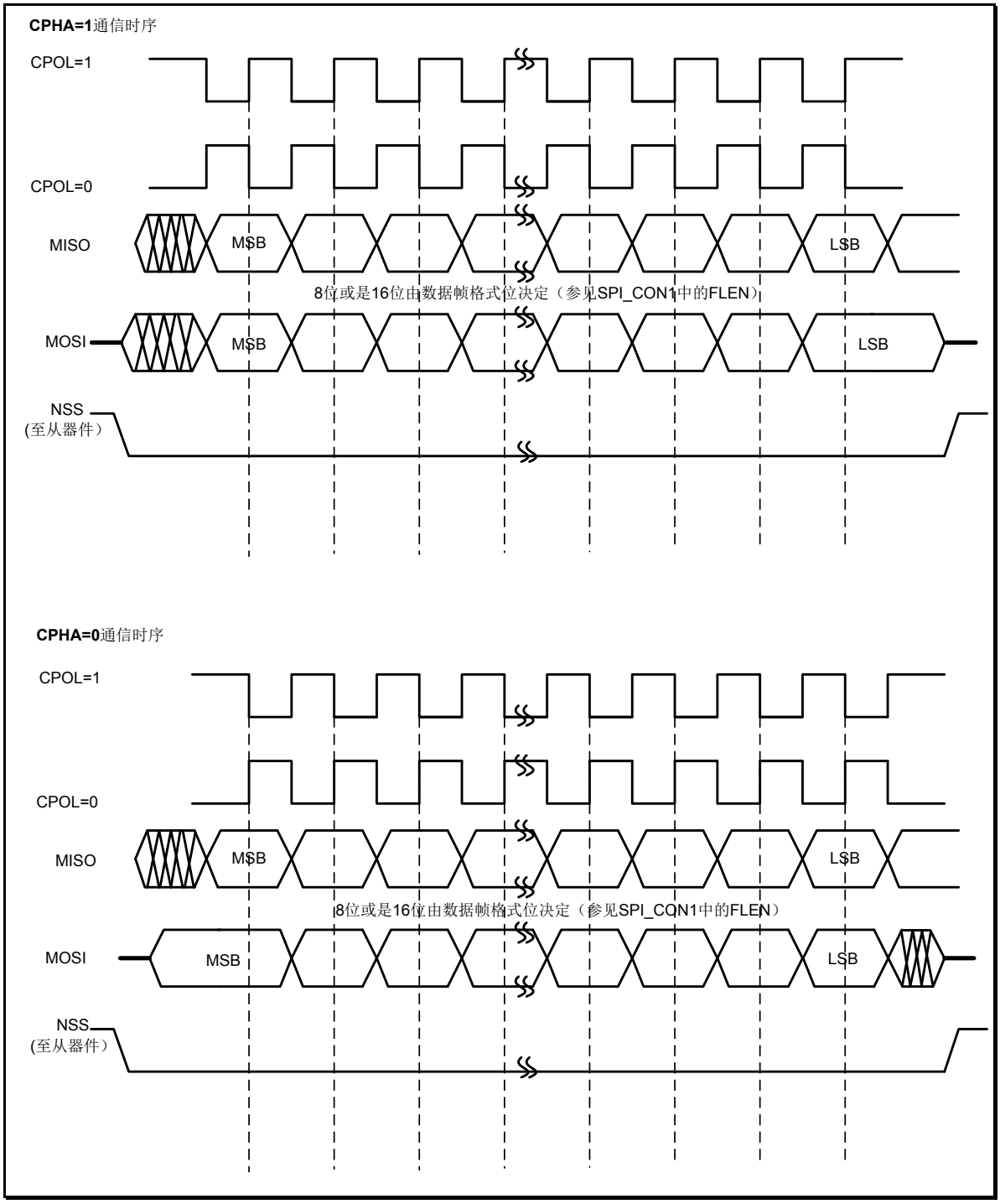


图 20-3 数据时钟时序图

注：上图中显示的是当 SPI_CON1.LSBFST 处于复位状态时的时序图。

数据帧长度

移出数据时 MSB 在前还是 LSB 在前取决于 SPI_CON1.LSBFST 位的值。

每个数据帧的长度均为 8 位或 16 位，具体取决于 SPI_CON1.FLEN 位的配置。所选的数据帧长度适用于发送和/或接收。

19. 4. 2 SPI从器件模式

SPI 作为从器件时，时钟信号由主器件提供，所以建议先使能从器件，然后主器件再发送时钟，否则数据传输可能不正常。建议按以下步骤配置 SPI 为从器件：

步骤

1. 先配置时钟，将 SPI_CON1.CPOL 和 SPI_CON1.CPHA 位配置好，以定义数据传输和时钟之间的关系，注意这两位从器件和主器件的配置需保持一致。
2. 通过 SPI_CON1.FLEN 位设置数据帧的长度，可选择 8 位或 16 位。
3. 帧格式（MSB 在前或 LSB 在前取决于 SPI_CON1.LSBFST 位的值）必须与主器件的帧格式相同。
4. 在硬件模式下（参见从器件选择（NSS）引脚管理），NSS 引脚在整个字节发送序列期间都会保持低电平。在 NSS 软件模式下，将 SPI_CON1.SSEN 位置 1，将 SPI_CON1.SSOUT 位清零。
5. 将 SPI_CON1.MSTREN 位清零，并将 SPI_CON1.SPIEN 位置 1。
6. 在此配置中，MOSI 引脚作为数据输入，MISO 引脚作为数据输出。

发送序列

数据字节在写周期内被并行加载到发送缓冲区中。

当从器件在其 MOSI 引脚上收到时钟信号和数据的最高有效位时，发送序列开始。其余位（8 位数据帧长度中的 7 个位，16 位数据帧长度中的 15 个位）将加载到移位寄存器中。SPI_STAT.TXBE 标志在数据从发送缓冲区传输到移位寄存器时置 1，并且在 SPI_CON2.TXBEIE 位置 1 时将生成中断。

接收序列

对于接收器，在数据传输完成时会产生以下动作：

- ◇ 移位寄存器中的数据将传输到接收缓冲区，并且 SPI_STAT.RXBNE 位置 1。
- ◇ 如果 SPI_CON2.RXBNEIE 位置 1，则生成中断。

在出现最后一个采样时钟边沿后，SPI_STAT.RXBNE 位置 1，移位寄存器中接收的数据字节被拷贝到接收缓冲区中。当读取 SPI_DATA 寄存器时，SPI 外设将返回此缓冲值。

通过读取 SPI_DATA 寄存器将 SPI_STAT.RXBNE 位清零。

19. 4. 3 SPI主器件模式

时钟信号由主器件从 SCK 引脚发出，传输给从器件。

步骤

1. 先配置时钟，设置 SPI_CON1.BAUD[2:0]位，定义时钟波特率。
2. 配置 SPI_CON1.CPOL 和 SPI_CON1.CPHA 位，以定义数据传输和串行时钟之间的关系（四种关系中的一种）（参见上图）。
3. 通过 SPI_CON1.FLEN 位设置数据帧的长度，可选择 8 位或 16 位。
4. 通过 SPI_CON1.LSBFST 位设置帧格式，可选择 MSB 在前或 LSB 在前。
5. 当 NSS 引脚配置成输入，在 NSS 硬件模式下，NSS 引脚在整个字节发送序列期间都要连接到高电平信号；在 NSS 软件模式下，需要将 SPI_CON1.SSEN 和 SPI_CON1.SSOUT 位都置 1。如果 NSS 引脚配置成输出，只需要将 SPI_CON2.NSSOE 位置 1 即可。
6. SPI_CON1.MSTREN 位置位使 SPI 工作在主模式下，然后 SPI_CON1.SPIEN 位置位使能 SPI 模块（仅当 NSS 引脚与高电平信号连接时，这两个位才保持置 1）。
7. 在此配置中，MOSI 引脚作为数据输出，MISO 引脚作为数据输入。

发送序列

在发送缓冲区中写入字节时，发送序列开始。

在第一个位传输期间，数据字节（从内部总线）并行加载到移位寄存器中，然后以串行方式移出到 MOSI 引脚，至于是 MSB 在前还是 LSB 在前则取决于 SPI_CON1.LSBFST 位。TXBE 标志在数据从发送缓冲区传输到移位寄存器时置 1，并且在 SPI_CON2.TXBEIE 位置 1 时将生成中断。

接收序列

对于接收器，在数据传输完成时产生如下动作：

- ◇ 移位寄存器中的数据将传输到接收缓冲区，并且 SPI_STAT.RXBNE 位置 1
- ◇ 如果 SPI_CON2.RXBNEIE 位置 1，则生成中断

在出现最后一个采样时钟边沿时，SPI_STAT.RXBNE 位置 1，移位寄存器中接收的数据字节被拷贝到接收缓冲区中。当读取 SPI_DATA 寄存器时，SPI 外设将返回此缓冲值。

通过读取 SPI_DATA 寄存器将 SPI_STAT.RXBNE 位清零。

如果在发送开始后将要发送的下一个数据置于发送缓冲区，则可保持连续的发送流。请注意，仅当 SPI_STAT.TXBE 位为 1 时，才可以对发送缓冲区执行写操作。

注：如果与之通信的从器件需要在每个字节传输之间拉低片选信号，必须将该主器件的 NSS 配置成 GPIO，或使用另外 GPIO，通过软件控制从器件的片选。

19.4.4 半双工通信模式

SPI 能够在以下两种配置中以半双工模式工作。

1 个时钟和 1 条双向数据线（SPI_CON1.BIDEN=1）

可将 SPI_CON1.BIDEN 位置 1 来使能此模式。在此模式下，SCK 作为时钟信号输出引脚，MOSI（主模式下）或 MISO（从模式下）作为数据通信引脚。通过 SPI_CON1.BIDOEN 位来选择传输方向（输入/输出）。当该位置 1 时，数据线为输出，否则为输入。

1 个时钟和 1 条单向数据线（SPI_CON1.BIDEN=0）

在此模式下，应用程序可使用 SPI 的只发送或只接收功能。

◇ 只发送模式类似于全双工模式：在发送引脚（主模式下的 MOSI 或从模式下的 MISO）上发送数据，不再接收数据。

◇ 只接收模式下，应用程序可将 SPI_CON1.RXO 位置 1 来关闭 SPI 输出功能。

当 SPI 进入只读模式后：

◇ 一旦在主模式下使能 SPI 后，主器件会立即从 SCK 引脚发送时钟，即意味着通信开启，当 SPI_CON1.SPIEN 位清 0 后，SPI 模块关闭，通信也立即停止。此模式下无需读取 BUSY 标志，因为开始通信后此标志一直为 1。

◇ 在从模式下，只要 NSS 引脚被拉低（或在 NSS 软件模式下将 SPI_CON1.SSOUT 位清零），意味着从器件被选中，同时一直有来自主器件的 SCK 输入，SPI 就会继续接收。

19.4.5 数据发送和接收

接收和发送缓冲区

数据的发送和接收都会被存在各自内部的缓冲区中，如接收到数据后，数据会先存在内部接收缓冲区，而在发送前，数据会被先准备好存在内部发送缓冲区中。

在主模式下启动通信序列

◇ 在全双工模式下（SPI_CON1.BIDEN=0 且 SPI_CON1.RXO=0）

- 将数据写入到 SPI_DATA 寄存器（发送缓冲区）后，通信序列启动。
- 随后在第一个位的发送期间，将数据从发送缓冲区并行加载到 8 位移位寄存器中，然后以串行方式将其移出到 MOSI 引脚。
- 同时，将 MISO 引脚上接收的数据以串行方式移入 8 位移位寄存器，然后并行加载到 SPI_DATA 寄存器（接收缓冲区）中。

◇ 在单向只接收模式下（SPI_CON1.BIDEN=0 且 SPI_CON1.RXO=1）

- 只要 SPI_CON1.SPIEN = 1，通信序列就立即开始。
- 需要先将接收器激活，然后 MISO 引脚上接收的数据会先以串行方式移入 8 位移位寄存器，接着再从移位寄存器并行加载到 SPI_DATA 寄存器（接收缓冲区）中。

◇ 在双向单线模式下，进行发送时（SPI_CON1.BIDEN=1 且 SPI_CON1.BIDOEN=1）

- 将数据写入到 SPI_DATA 寄存器（发送缓冲区）时，通信序列启动。
- 随后在第一个位的发送期间，将数据从发送缓冲区并行加载到 8 位移位寄存器中，然后以串行方式将其移出到 MOSI 引脚。
- 不接收任何数据。

◇ 在双向单线模式下，进行接收时（SPI_CON1.BIDEN=1 且 SPI_CON1.BIDOEN=0）

- 只要 SPI_CON1.SPIEN=1 且 SPI_CON1.BIDOEN=0，通信序列就立即开始。
- 在 MOSI 引脚上接收的数据以串行方式移入 8 位移位寄存器，然后并行加载到 SPI_DATA 寄存器（接收缓冲区）中。
- 发送器没有激活，因此不会有数据以串行方式移出 MOSI 引脚。

在从模式下启动通信序列

◇ 在全双工模式下（SPI_CON1.BIDEN=0 且 SPI_CON1.RXO=0）

- 当从器件在其 MOSI 引脚上收到时钟信号和数据的第一个位时，通信序列开始。其余 7 个位将加载到移位寄存器中。
- 同时，在第一个位的发送期间，将数据从发送缓冲区并行加载到 8 位移位寄存器中，然后以串行方式将其移出到 MISO 引脚。在 SPI 主器件启动传输前，软件必须已把要从器件发送的数据写入发送缓冲区。

◇ 在单向只接收模式下（SPI_CON1.BIDEN=0 且 SPI_CON1.RXO=1）

- 当从器件在其 MOSI 引脚上收到时钟信号和数据的第一个位时，通信序列开始。其余 7 个位将加载到移位寄存器中。
- 由于发送器没有激活，因此不会有数据以串行方式移出 MISO 引脚。

◇ 在双向单线模式下，进行发送时（SPI_CON1.BIDEN=1 且 SPI_CON1.BIDOEN=1）

- 当从器件收到时钟信号，并且 MISO 引脚上发出发送缓冲区中的第一位数据时，通信序列开始。
- 随后在第一个位的发送期间，将数据从发送缓冲区并行加载到 8 位移位寄存器中，然后以串行方式将其移出到 MISO 引脚。在 SPI 主器件启动传输前，软件必须已把要从器件发送的数据写入发送缓冲区。
- 不接收任何数据。

◇ 在双向单线模式下，进行接收时（SPI_CON1.BIDEN=1 且 SPI_CON1.BIDOEN=0）

- 当从器件在其 MISO 引脚上收到时钟信号和数据的第一个位时，通信序列开始。
- 在 MISO 引脚上接收的数据以串行方式移入 8 位移位寄存器，然后并行加载到 SPI_DATA 寄存器（接收缓冲区）中。
- 由于发送器没有被激活，因此不会有数据以串行方式移出 MISO 引脚。

处理数据发送与接收

将数据从发送缓冲区传输到移位寄存器时，SPI_STAT.TXBE 位置 1。该标志表示内部发送缓冲区已准备好加载接下来的数据。如果 SPI_CON2.TXBEIE 位置 1，则此时可产生中断。通过对 SPI_DATA 寄存器执行写操作将 SPI_STAT.TXBE 位清零。

注意：软件必须确保在尝试写入发送缓冲区之前 TXBE 标志已置 1。否则，将覆盖之前写入发送缓冲区但未发送或发送完全的数据。

当数据从移位寄存器转移到接收缓冲区时，SPI_STAT.RXBNE 位会置 1，表示用户此时可从 SPI_DATA 寄存器读取接收的数据，若用户将 SPI_CON2.RXBNEIE 位置 1，则会在 SPI_STAT.RXBNE 位置 1 时产生中断。通过对 SPI_DATA 寄存器进行读操作将 SPI_STAT.RXBNE 位清 0。

对于某些配置，可以在最后一次数据传输期间使用 BUSY 标志来等待传输完成。

主模式或从模式下的全双工发送和接收过程（SPI_CON1.BIDEN=0 且 SPI_CON1.RXO=0）

发送和接收数据的处理过程，用户请参见以下步骤（参见下文 2 张图）：

1. 通过将 SPI_CON1.SPIEN 位置 1 来使能 SPI。
2. 等待 SPI_STAT.TXBE=1，将第一个要发送的数据项写入 SPI_DATA 寄存器（此操作会将 SPI_STAT.TXBE 位清零），然后等待 SPI_STAT.RXBNE=1，读取 SPI_DATA 以接收第一个接收到的数据（此操作会将 SPI_STAT.RXBNE 位清零）。重复以上操作，直到发送完 N-1 个数据，同时收到 N-1 个数据后。
3. 等待 SPI_STAT.TXBE 标志置位，然后写入要发送的第二个数据项。然后等待至 SPI_STAT.RXBNE=1，读取 SPI_DATA 以获取收到的第一个数据项（此操作会将 SPI_STAT.RXBNE 位清零）。对每个要发送/接收的数据项重复此操作，直到发送并接收完最后的数据。
4. 等待至 SPI_STAT.TXBE=1，然后等待至 SPI_STAT.BUSY=0，再关闭 SPI。
5. 此外，还可以使用在 RXBNE 或 TXBE 标志所产生的中断对应的各个中断子程序来实现该过程。

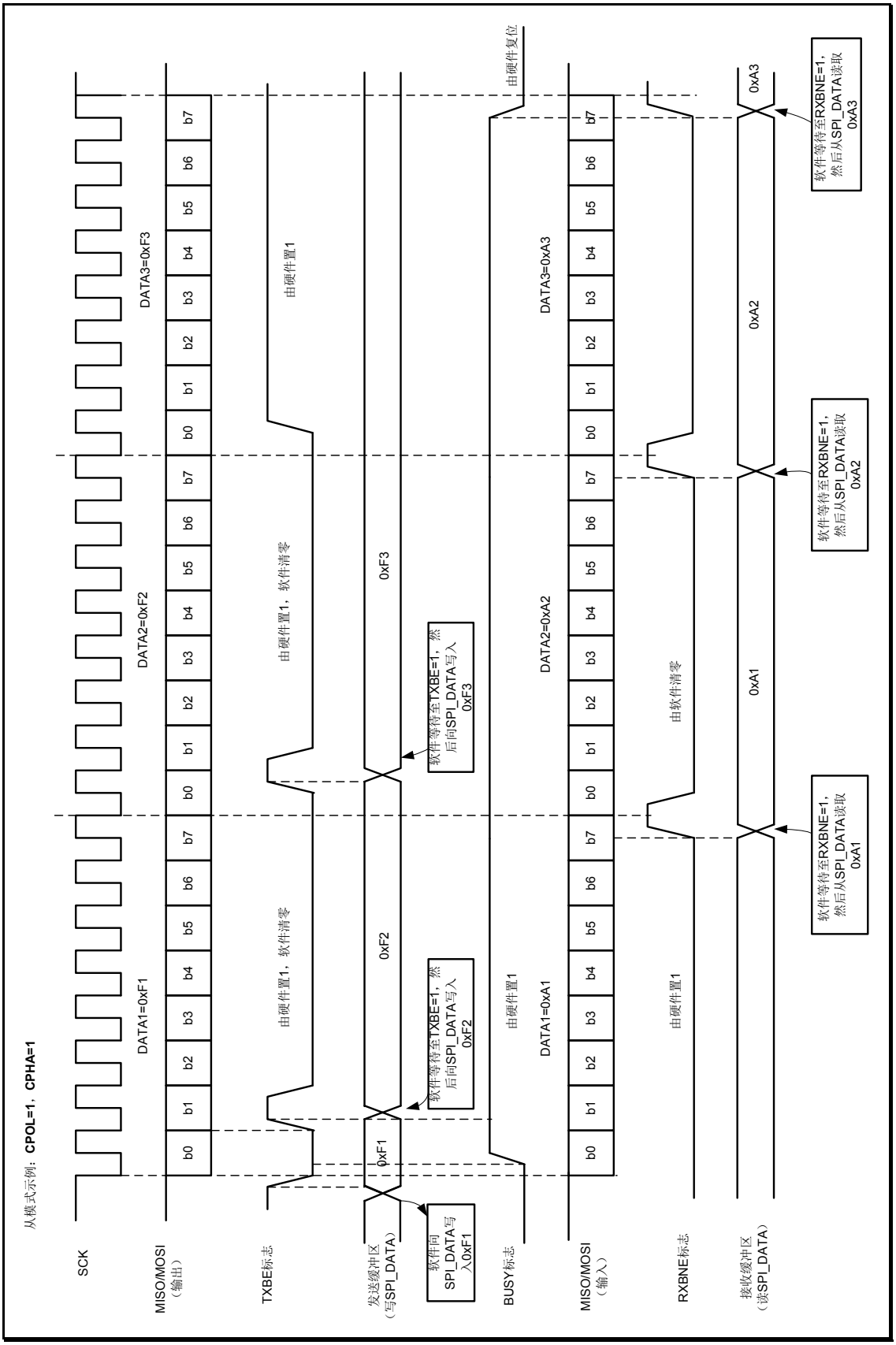


图 20-4　主/全双工模式（BIDEN=0 且 RXO=0）的 TXBE/RXBNE/BUSY 行为（连续传输）

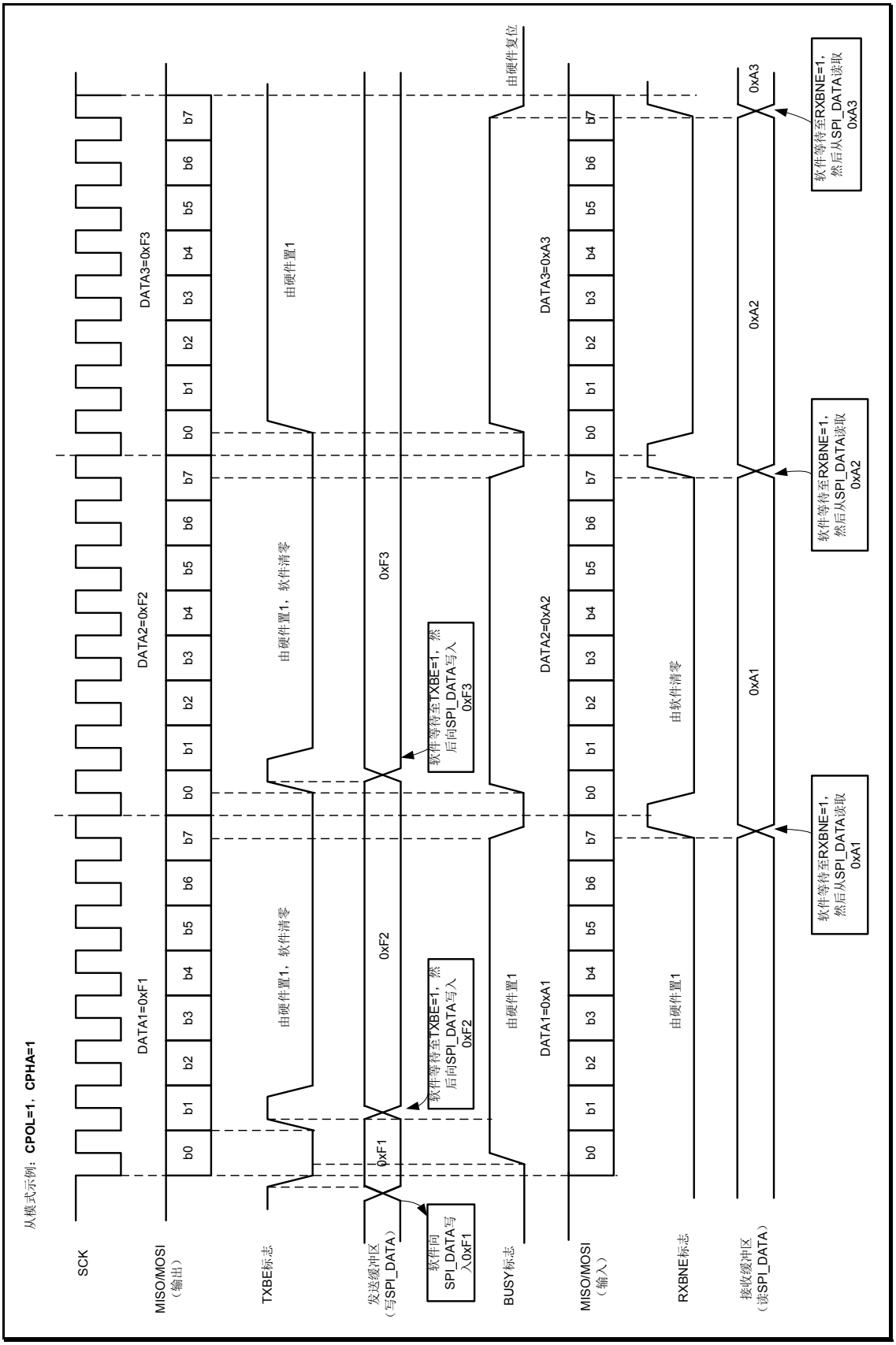


图 20-5　从/全双工模式（BIDEN=0 且 RXO=0）的 TXBE/RXBNE/BUSY 行为（连续传输）

只发送模式下的数据发送过程（SPI_CON1.BIDEN=0、SPI_CON1.RXO=0）

步骤如下:

1. 通过将 SPI_CON1.SPIEN 位置 1 来使能 SPI。
2. 等待 SPI_STAT.TXBE=1，然后写入要发送的数据项。对每个要发送的数据项重复此步骤。
3. 将最后一个数据项写入 SPI_DATA 寄存器后，等待至 SPI_STAT.TXBE=1，然后等待至 SPI_STAT.BUSY=0，这表示最后的数据发送完成。

此外，还可以使用在 TXBE 标志所产生的中断对应的中断子程序来实现该过程。

注 1：在不连续通信期间，在对 SPI_DATA 执行写操作与 SPI_STAT.BUSY 位置 1 之间有 2 个 APB 时钟周期的延迟。因此，在只发送模式下，写入最后的数据后，必须先等待 SPI_STAT.TXBE 位置 1，然后等待 SPI_STAT.BUSY 位清零。

注 2：在只发送模式下，发送两个数据项后，SPI_STAT.OVERR 标志将置 1，因为始终不会读取接收的数据。

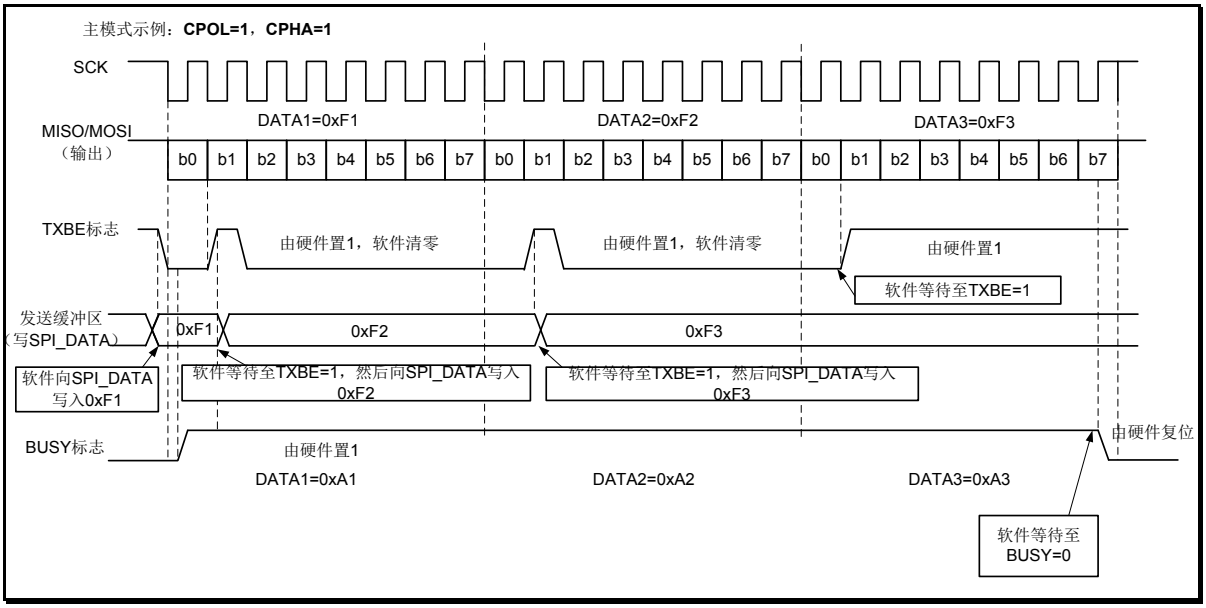


图 20-6 主设备只发送模式（BIDEN=0 且 RXO=0）下的 TXBE/BUSY 行为（连续传输）

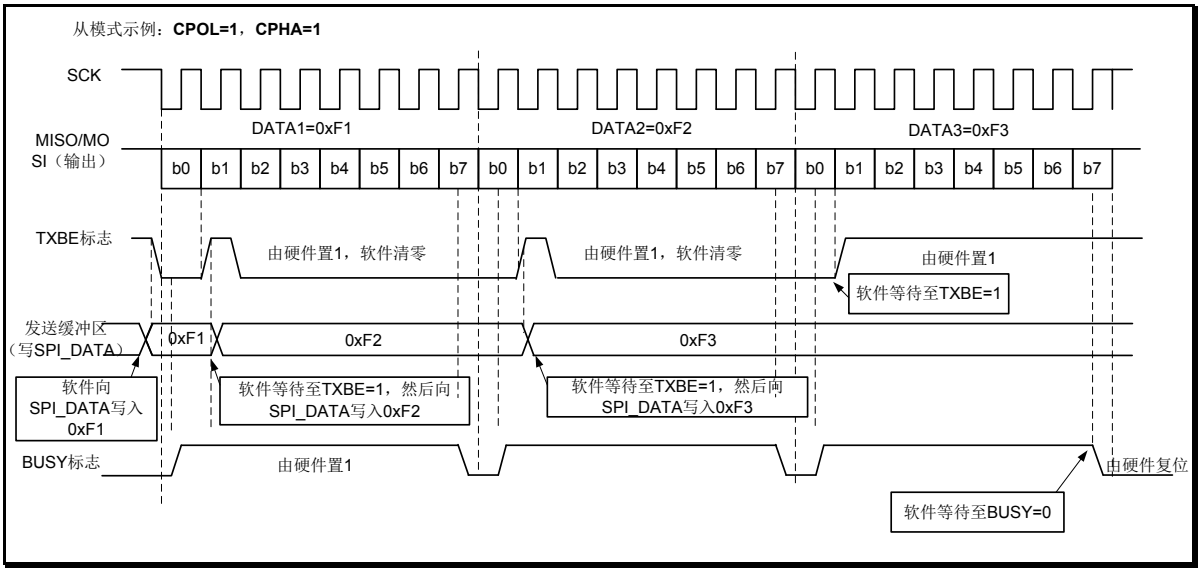


图 20-7　从设备只发送（BIDEN=0 且 RXO=0）下的 TXBE/BUSY 行为（连续传输）

单线双向模式下的发送过程（SPI_CON1.BIDEN=1 且 SPI_CON1.BIDOEN=1）

在此模式下，过程与双线只发送模式下的过程相似，但是在 SPI 模块使能前，必须将 SPI_CON1.BIDEN 位和 SPI_CON1.BIDOEN 位置 1。

单向只接收过程（SPI_CON1.BIDEN=0 且 SPI_CON1.RXO=1）

在此模式下，可以按如下所述简化过程：

1. 将 SPI_CON1.RXO 位置 1。
2. 通过将 SPI_CON1.SPIEN 位置 1 使能 SPI。
3. 等待 SPI_STAT.RXBNE=1，然后读取 SPI_DATA 寄存器以获取接收的数据（此操作会将 SPI_STAT.RXBNE 位清零）。对每个要接收的数据项重复此操作。

a) 在主模式下，一旦 SPI 使能，SCK 会立即发送时钟，从器件接收到时钟后会发送数据，直到主器件关闭 SPI 功能，通信结束。

b) 在从模式下，当 SPI 主器件将该从器件的 NSS 驱动为低电平并输出 SCK 时钟时，接收数据。

此外，还可以使用在 SPI_STAT.RXBNE 标志所产生的中断对应的中断子程序来实现该过程。

注：如果需要在最后一次传输后关闭 SPI，请采纳章节：关闭 SPI 中所述的建议。

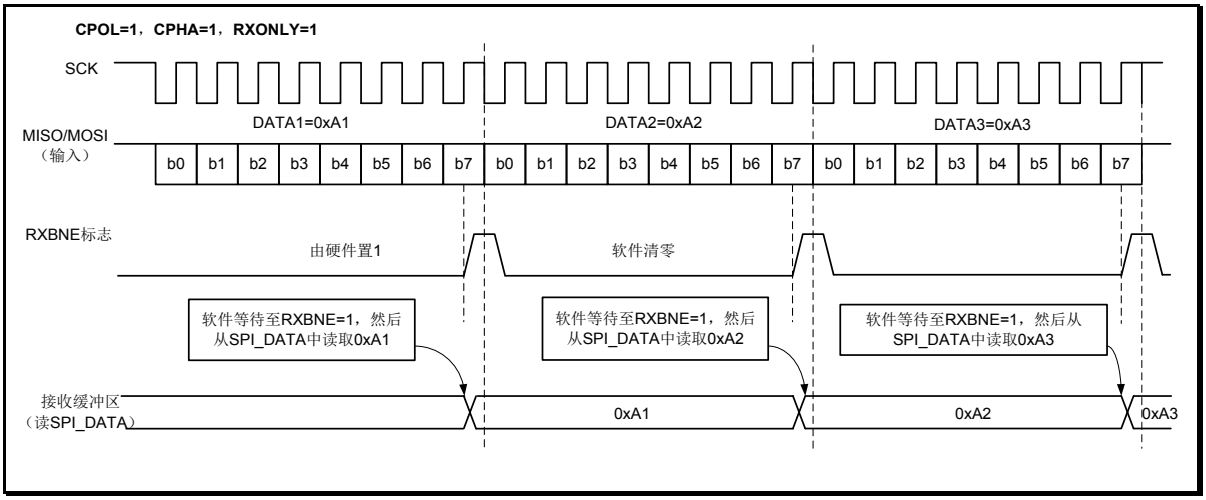


图 20-8　只接收模式（BIDEN=0 且 RXO=1）下的 RXBNE 行为（连续传输）

单线双向模式下的接收过程（SPI_CON1.BIDEN=1 和 SPI_CON1.BIDOEN=0）

在此模式下，过程与双线只接收模式的过程相似，但是在 SPI 模块使能之前，需要将 SPI_CON1.BIDEN 位置 1,并将 SPI_CON1.BIDOEN 位清 0。

连续传输和间断传输

在主模式下发送数据时，如果软件处理速度足够快，可以在检测到 SPI_STAT.TXBE=1（或发生 TXBE 中断），并且当前数据传输未结束，立即将下一次的数据写入 SPI_DATA 寄存器，则能实现连续的通信。观察到的现象是 SPI_STAT.BUSY 位一直为 1 不被清除，并且每个数据的 SPI 时钟保持连续。

相反，如果软件速度不够快，则可能导致通信中断。在这种情况下，各数据传输之间会清零 SPI_STAT.BUSY 位。

在主设备仅接收模式（SPI_CON1.RXO=1）下，通信始终是连续的，且 SPI_STAT.BUSY 位始终为 1。

在从模式下，通信的连续性由 SPI 主器件决定。任何情况下（即使通信是连续的），在各个数据传输之间 BUSY 标志都会短暂变为低电平，持续时间为一个 SPI 时钟周期。

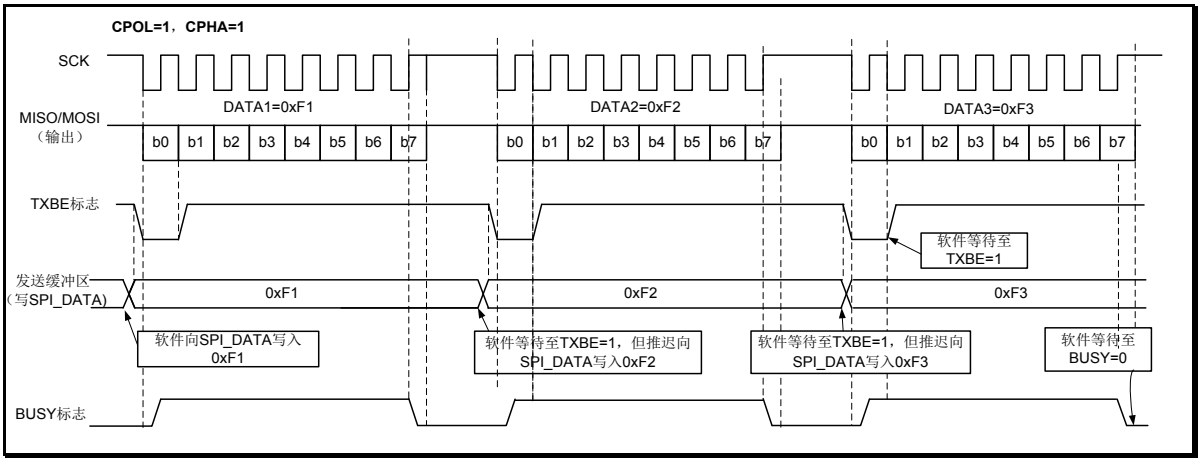


图 20-9　发送时（BIDEN =0 且 RXO=0）的 TXBE/BUSY 行为（间断传输）

19. 4. 6 CRC计算

为确保通信的可靠性，SPI 模块实现了硬件 CRC 功能。

针对发送或接收的数据帧宽度有 8 位和 16 位的选择，硬件 CRC 计算也提供了两种计算标准，分别为 8 位数据的 CRC8 和 16 位数据的 CRC16。

将 SPI_CON1.CRCEN 位置 1 来使能 CRC 计算功能，此操作会复位 CRC 寄存器（SPI_RXCRC 和 SPI_TXCRC）。在全双工或只发送模式下，如果传输由软件（CPU 模式）管理，则在将最后传输的数据写入 SPI_DATA 后，必须立即对 SPI_CON1.NXTCRC 位执行写操作。最后一次数据传输结束时，将发送 SPI_TXCRC 寄存器内的值。

在只接收模式下，如果传输由软件（CPU 模式）管理，则在接收到倒数第二个数据后，必须对 SPI_CON1.NXTCRC 位执行写操作。在收到最后一个数据后会收到 CRC，然后执行 CRC 校验。

如果传输过程中出现数据损坏，则在数据和 CRC 传输结束时，SPI_STAT.CRCERR 位将置 1。

如果发送缓冲区中存在数据，则只有在发送数据字节后才会发送 CRC 值。在 CRC 发送期间，CRC 计算器处于关闭状态且寄存器值保持不变。

可通过以下步骤使用 CRC 进行 SPI 通信：

1. 对 SPI_CON1.BAUD、SPI_CON1.CPOL、SPI_CON1.CPHA、SPI_CON1.LSBFST、SPI_CON1.SSEN、SPI_CON1.SSOUT 和 SPI_CON1.MSTREN 值进行设置。
2. 向 SPI_CRCPOLY 寄存器中写入计算 CRC 的多项式。
3. 通过将 SPI_CON1.CRCEN 位置 1 来使能 CRC 计算。此操作还会将 SPI_RXCRC 和 SPI_TXCRC 寄存器清零。
4. 通过将 SPI_CON1.SPIEN 位置 1 使能 SPI。
5. 启动并保持通信，直到只剩下一个字节或半字未发送或接收。
   - 在全双工或只发送模式下，如果传输由软件管理，则在向发送缓冲区写入最后一个字节或半字后，将 SPI_CON1.NXTCRC 位置 1，以表示在发送完最后一个字节后将发送 CRC。
   - 在只接收模式下，在接收倒数第二个数据后，立即将 SPI_CON1.NXTCRC 位置 1，以便使 SPI 准备好在接收完最后一个数据后进入 CRC 阶段。在 CRC 传输期间，CRC 计算将冻结。
6. 传输完最后一个字节或半字后，SPI 进入 CRC 传输和校验阶段。在全双工模式或只接收模式下，将接收的 CRC 与 SPI_RXCRC 值进行比较。如果两个值不匹配，则 SPI_STAT.CRCERR 位将置 1，并且在 SPI_CON2.ERRIE 位置 1 时会产生中断。

当 SPI 处于从模式时，注意只能在时钟稳定（即，时钟处于空闲电平）时使能 CRC 计算。否则，可能导致 CRC 计算错误。因为，只要 SPI_CON1.CRCEN 位置 1，无论 SPIEN 位的值如何，只要有时钟输入，CRC 计算器就开始工作。

在 SPI 通信时钟频率较高的情况下，发送 CRC 时务必小心。由于在 CRC 传输阶段 CPU 应尽可能空闲，因此禁止在 CRC 发送阶段调用函数，以便避免最后的数据和 CRC 接收出

错。实际上，在发送/接收最后的数据之前必须对 SPI_CON1.NXTCRC 位执行写操作。

SPI 通信时钟频率较高时，建议使用 DMA 模式来避免由于 CPU 访问影响 SPI 带宽而导致 SPI 速度性能下降。

如果将器件配置为从器件，并且使用 NSS 硬件模式，则需要在数据阶段和 CRC 阶段之间将 NSS 引脚保持为低电平。

当 SPI 配置为从模式并且 CRC 功能已使能时，即使 NSS 引脚上为高电平，也会进行 CRC 计算。例如，在多从模式环境下可能出现这种情况，此时通信主器件会交替寻址从器件。

在对从器件片选的切换期间内，应在主器件和从器件两端同时将 CRC 值清零，以重新同步主从双方的 CRC 计算。

要将 CRC 清零，请按以下步骤操作：

1. 关闭 SPI（SPI_CON1.SPIEN = 0）
2. 将 SPI_CON1.CRCEN 位清零
3. 将 SPI_CON1.CRCEN 位置 1
4. 使能 SPI（SPI_CON1.SPIEN = 1）

19. 4. 7 状态标志

用户监视三种状态标志来判断 SPI 总线的状态。

发送缓冲区为空（TXBE）

此标志置 1 时，表示发送缓冲区为空，此时可以将待发送的数据加载到缓冲区中。对 SPI_DATA 寄存器执行写操作时，将清零 TXBE 标志。

接收缓冲区非空（RXBNE）

此标志置 1 时，表示接收缓冲区中存在有效的已接收数据。此时用户可读取 SPI_DATA 寄存器，读取后此标志位会被自动清除。

通信忙（BUSY）

BUSY 标志用于指示 SPI 通信的状态。此标志由硬件置 1 和清零。

SPI_STAT.BUSY 位置 1 时，表示 SPI 正在通信中。但是在主模式下的双向通信接收模式（SPI_CON1.MSTREN=1 且 SPI_CON1.BIDEN=1 且 SPI_CON1.BIDOEN=0）时，BUSY 标志在接收过程中保持低电平。

在通信结束前，用户可检测 SPI_STAT.BUSY 位是否为 0，表示通信已结束，此时关闭 SPI 模块，停止通信。

BUSY 标志还可用于避免在多主模式系统中发生写冲突。

传输开始时，SPI_STAT.BUSY 位将置 1，但在主模式下的双向通信接收模式（SPI_CON1.MSTREN=1 且 SPI_CON1.BIDEN=1 且 SPI_CON1.BIDOEN=0）下例外。

在以下情况硬件将清零该标志：

◇ 传输完成时（主模式下的连续通信除外）

◇ 关闭 SPI 时

◇ 发生模式错误时（SPI_STAT.MODERR=1）

当通信不连续时，BUSY 标志在各通信之间处于低电平。

当通信连续时：

◇ 在主模式下，BUSY 标志在所有传输期间均保持高电平

◇ 在从模式下，BUSY 标志在各传输之间的一个 SPI 时钟周期内变为低电平

注：请勿使用 BUSY 标志处理每次数据发送或接收，最好改用 TXBE 标志和 RXBNE 标志。

19. 4. 8 SPI关闭流程

传输终止时，通过清除 SPI_CON1.SPIEN 位来关闭 SPI 模块。

建议在关闭 SPI 时按以下步骤操作：

在主模式或全双工从模式（SPI_CON1.BIDEN=0、SPI_CON1.RXO=0）下

1. 等待 SPI_STAT.RXBNE=1 以接收最后的数据
2. 等待 SPI_STAT.TXBE=1，并且 SPI_STAT.BUSY=0
3. 设置 SPI_CON1.SPIEN=0 以关闭 SPI，最后进入停止模式（或关闭外设时钟）

在主模式或单向只发送从模式（SPI_CON1.BIDEN=0、SPI_CON1.RXO=0）或双向通信发送模式 （SPI_CON1.BIDEN=1、SPI_CON1.BIDOEN=1）下

在最后的数据写入 SPI_DATA 寄存器后：

1. 等待 SPI_STAT.TXBE=1
2. 然后等待 SPI_STAT.BUSY=0
3. 设置 SPI_CON1.SPIEN=0 以关闭 SPI，最后进入停止模式（或关闭外设时钟）

在单向只接收主模式（SPI_CON1.MSTREN=1、SPI_CON1.BIDEN=0、SPI_CON1.RXO=1）或双向通信接收模式（SPI_CON1.MSTREN=1、SPI_CON1.BIDEN=1、SPI_CON1.BIDOEN=0）下

必须以特殊方式管理这种情况，以避免多余的 SPI 数据传输：

1. 等待倒数第二个数据（第 n-1 个）对应的 RXBNE 标志置位
2. 然后等待一个 SPI 时钟周期（使用软件循环），才能关闭 SPI（SPIEN=0）
3. 再等待最后的 SPI_STAT.RXBNE=1，然后进入停止模式（或关闭外设时钟）

注：在双向通信接收主模式（SPI_CON1.*MSTREN=1*、SPI_CON1.*BIDEN=1*、SPI_CON1.*BIDOEN=0*）下，*BUSY* 标志在传输期间保持低电平。

在只接收从模式（SPI_CON1.MSTREN=0、SPI_CON1.BIDEN=0、SPI_CON1.RXO=1）或双向通信接收模式（SPI_CON1.MSTREN=0、SPI_CON1.BIDEN=1、SPI_CON1.BIDOEN=0）下

1. 可以随时关闭 SPI（写入 SPI_CON1.SPIEN=0）：当前传输完成后，SPI 才被真正关闭

2. 之后，如果要进入停止模式，则必须首先等待至 SPI_STAT.BUSY = 0，然后才能进入停止模式（或关闭外设时钟）

19. 4. 9 DMA请求

为了更方便的实现高速通信，SPI 提供了 DMA 功能。

当使能 SPI_CON2 寄存器中相应的使能位时，将请求 DMA 访问。发送缓冲区和接收缓冲区会发出各自的 DMA 请求（参见下面 2 张图）：

◇ 在发送过程中，当 SPI_STAT.TXBE 位置 1 时会发出 DMA 请求。DMA 随后对 SPI_DATA 寄存器执行写操作（此操作会将 SPI_STAT.TXBE 位清零）。

◇ 在接收过程中，当 SPI_STAT.RXBNE 位置 1 时会发出 DMA 请求。DMA 随后对 SPI_DATA 寄存器执行读操作（此操作会将 SPI_STAT.RXBNE 位清零）。

当 SPI 仅用于发送数据时，可以只使能 SPI Tx DMA 通道。在这种情况下，SPI_STAT.OVERR 位会置 1，因为未读取接收的数据。

当 SPI 仅用于接收数据时，可以只使能 SPI Rx DMA 通道。

在发送模式下，DMA 完成了所有要发送数据的传输后，DMA_STATUS.STATUS 位会被置 9，可以对 BUSY 标志进行监视，以确保 SPI 通信已完成。在关闭 SPI 或进入停止模式前必须执行此步骤，以避免损坏最后一次数据的发送。软件必须先等待 SPI_STAT.TXBE=1，再等待 SPI_STAT.BUSY=0。

注：在不连续通信期间，在对 SPI_DATA 执行写操作与 SPI_STAT.BUSY 位置 1 之间有 2 个 APB 时钟周期的延迟。因此，必须在写入最后的数据后，先等待 SPI_STAT.TXBE=1，再等待 SPI_STAT.BUSY=0。

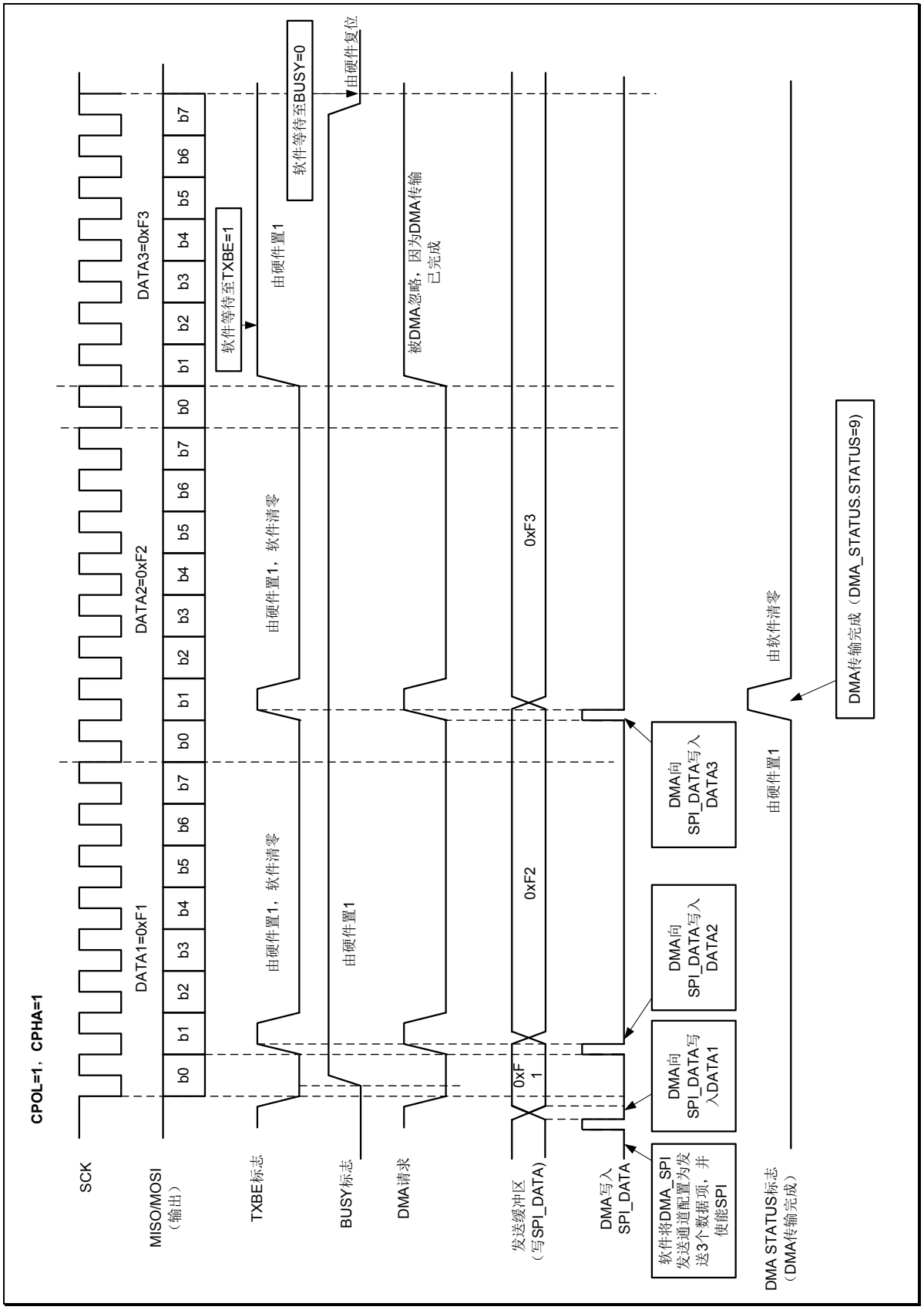


图 20-10 使用 DMA 进行发送

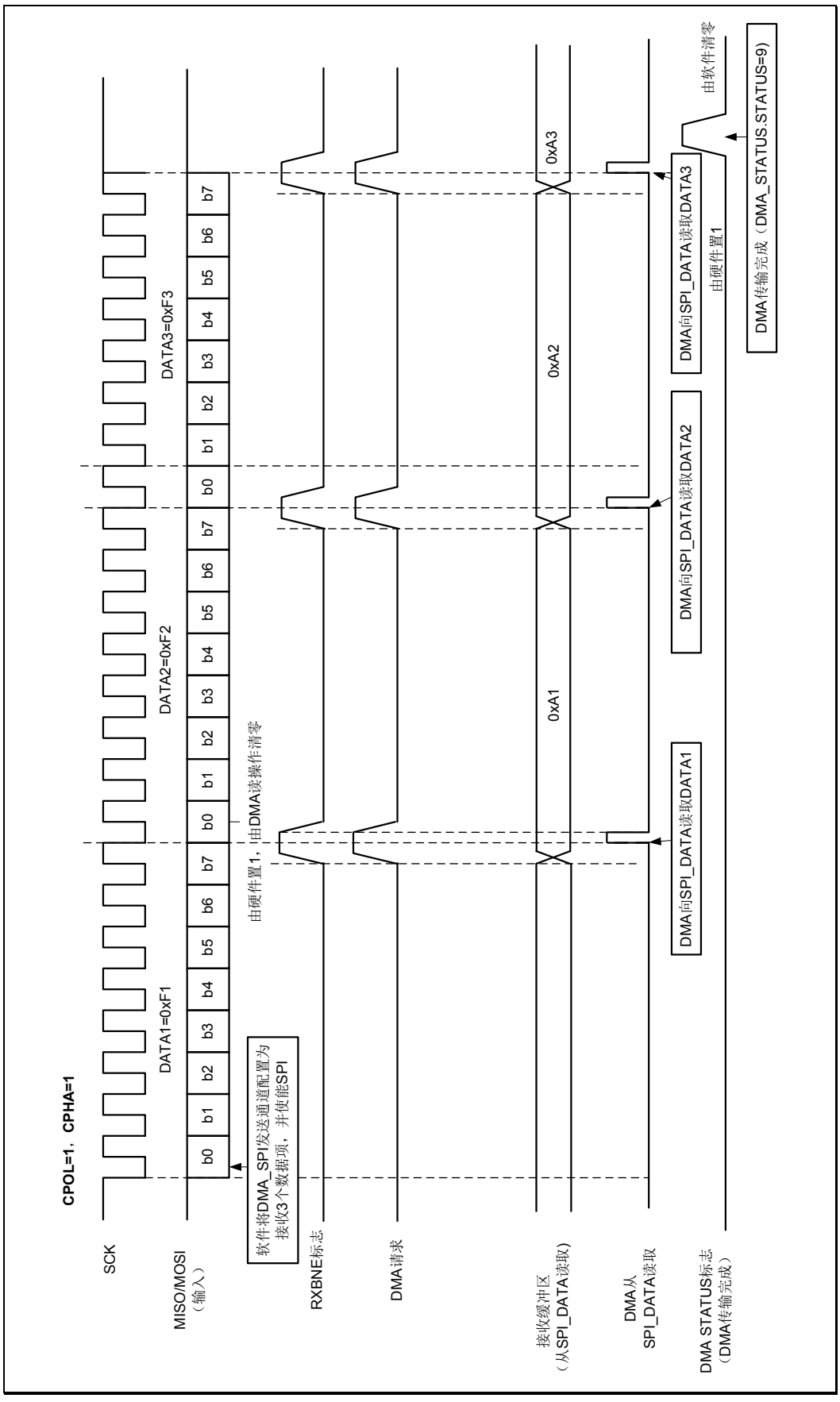


图 20-11　使用 DMA 进行接收

19. 4. 10 错误标志

模式错误（MODERR）

当主器件的 NSS 引脚拉低（NSS 硬件模式下）或 SPI_CON1.SSOUT 位为 0（NSS 软件模式下）时，会发生模式错误，这会自动将 SPI_STAT.MODERR 位置 1。模式错误会在以下几方面影响 SPI 外设：

◇ SPI_CON1.SPIEN 位清零。这将关闭器件的所有输出，并关闭 SPI 接口。

◇ SPI_CON1.MSTREN 位清零，从而强制器件进入从模式。

软件执行以下步骤可将 SPI_STAT.MODERR 位清零：

1. 在 SPI_STAT.MODERR 位置 1 时，对 SPI_STAT 寄存器执行读或写访问。

2. 然后，对 SPI_CON1 寄存器执行写操作。

为避免包含多个 MCU 的系统中发生多从模式冲突，必须在 SPI_STAT.MODERR 位清零序列期间将 NSS 引脚拉高。在该清零序列后，可以将 SPI_CON1.SPIEN 和 SPI_CON1.MSTREN 位恢复到原始状态。

硬件不允许在 SPI_STAT.MODERR 位置 1 时将 SPI_CON1.SPIEN 和 SPI_CON1.MSTREN 位置 1。

在从器件中，不能将 SPI_STAT.MODERR 位置 1。但是，在多主模式配置中，器件可在 SPI_STAT.MODERR 位置 1 时处于从模式。在这种情况下，SPI_STAT.MODERR 位指示系统控制可能存在多主模式冲突。可使用中断程序从此状态完全恢复，方法是执行复位或返回到默认状态。

溢出错误

当主器件发送完数据字节，而从器件尚未将上一个收到的数据所产生的 SPI_STAT.RXBNE 位清零时（即未从接收缓冲区中读走数据），将出现溢出情况。出现溢出错误时：

◇ SPI_STAT.OVERR 位置 1 并在 SPI_CON2.ERRIE 位置 1 时生成一个中断。

在这种情况下，接收器缓冲区内容不会被来自主器件的新数据更新。读取 SPI_DATA 寄存器将返回此字节。主器件后续发送的所有其它字节均将丢失。

依次读取 SPI_DATA 寄存器和 SPI_STAT 寄存器可将 SPI_STAT.OVERR 清除。

CRC 错误

当 SPI_CON1 寄存器中的 CRCEN 位置 1 时，此标志用于验证接收数据的有效性。如果移位寄存器中接收的值与 SPI_RXCRC 的值不匹配，SPI_STAT 寄存器中的 CRCERR 标志将置 1。

19. 4. 11 SPI中断

中断事件	事件标志	使能控制位
发送缓冲区空	TXBE	TXBEIE
接收缓冲区非空	RXBNE	RXBNEIE
模式错误	MODERR	ERRIE
溢出错误	OVERR	
CRC 错误	CRCERR	

表 20-1 SPI 中断

19. 5 特殊功能寄存器

19. 5. 1 寄存器列表

外设寄存器可支持半字（16 位）或字（32 位）访问。

SPI 寄存器列表		
寄存器名称	偏移地址	寄存器描述
SPI_CON1	0000_H	SPI 控制寄存器 1
SPI_CON2	0004_H	SPI 控制寄存器 2
SPI_STAT	0008_H	SPI 状态寄存器
SPI_DATA	$000C_H$	SPI 数据寄存器
SPI_CRCPOLY	0010_H	SPI CRC 多项式寄存器
SPI_RXCRC	0014_H	SPI RX CRC 寄存器
SPI_TXCRC	0018_H	SPI TX CRC 寄存器

19. 5. 2 寄存器描述

19. 5. 2. 1 SPI控制寄存器 1（SPI_CON1）

SPI 控制寄存器 1 （SPI_CON1）

偏移地址：00_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																BIDEN	BIDOEN	CRCEN	NXTCRC	FLEN	RXO	SSEN	SSOUT	LSBFST	SPIEN	BAUD			MSTREN	CPOL	CPHA

Reserved	Bit 31-16	—	保留
BIDEN	Bit 15	R/W	**双向通信使能** 0：选择双线单向通信数据模式 1：选择单线双向通信数据模式
BIDOEN	Bit 14	R/W	**双向通信输出使能** 此位结合 BIDEN 位，用于选择双向通信模式下的传输方向 0：禁止输出（只接收模式） 1：使能输出（只发送模式） 注意：在主模式下，使用 MOSI 引脚；在从模式下，使用 MISO 引脚。
CRCEN	Bit 13	R/W	**CRC 硬件计算使能** 0：禁止 CRC 计算 1：使能 CRC 计算 注意：为确保正确操作，只应在禁止 SPI（SPIEN =“0”）时对此位执行写操作
NXTCRC	Bit 12	R/W	**下一次传输 CRC** 0：数据阶段（无 CRC 阶段） 1：下一次传输为 CRC（CRC 阶段） 注意：当 SPI 配置为全双工或只发送模式时，只要最后一个数据写入 SPI_DATA 寄存器，就必须对 NXTCRC 执行写操作。当 SPI 配置为只接收模式时，必须在接收到倒数第二个数据之后将 NXTCRC 置 1。当传输由 DMA 管理时，此位应保持清零状态。
FLEN	Bit 11	R/W	**数据帧长度** 0：为发送/接收选择 8 位数据帧长度 1：为发送/接收选择 16 位数据帧长度 注意： 为确保正确操作，只应在禁止 SPI（SPIEN =“0”）时对此位执行写操作
RXO	Bit 10	R/W	**只接收使能** 此位结合 BIDEN 位，用于选择双线单向模式下的

			传输方向。此位也适用于多从模式系统，在此类系统中，不会访问特定从器件，也不会损坏访问的从器件的输出。 0：全双工（发送和接收） 1：关闭输出（只接收模式）
SSEN	Bit 9	R/W	**软件控制从器件使能** 当SSEN位置1时，NSS 引脚输入替换为SSOUT位的值。 0：禁止软件控制从器件 1：使能软件控制从器件
SSOUT	Bit 8	R/W	**软件控制片选输出** 仅当 SSEN 位置 1 时，此位才有效。此位的值将作用到 NSS 引脚上，并忽略 NSS 引脚的 IO 值。
LSBFST	Bit 7	R/W	**先发最低有效位** 0：先发送 MSB 1：先发送 LSB 注意：正在通信时不应更改此位。
SPIEN	Bit 6	R/W	**SPI 模块使能** 0：关闭外设 1：使能外设 注意：关闭 SPI 时，请按照章节：关闭 SPI 中所述的步骤操作。
BAUD	Bit 5-3	R/W	**波特率选择** 000：fPCLK/2 001：fPCLK/4 010：fPCLK/8 011：fPCLK/16 100：fPCLK/32 101：fPCLK/64 110：fPCLK/128 111：fPCLK/256 注意：正在通信时不应更改这些位。
MSTREN	Bit 2	R/W	**主模式使能** 0：从配置 1：主配置 注意：正在通信时不应更改此位。
CPOL	Bit 1	R/W	**时钟的极性控制** 0：在空闲的状态下，SCK 引脚保持低电平输出 1：在空闲的状态下，SCK 引脚保持高电平输出 注意：正在通信时不应更改此位。
CPHA	Bit 0	R/W	**时钟的相位控制** 0：从第一个时钟边沿开始采样数据 1：从第二个时钟边沿开始采样数据 注意：正在通信时不应更改此位。

19. 5. 2. 2 SPI控制寄存器 2（SPI_CON2）

SPI 控制寄存器 2 （SPI_CON2）

偏移地址：04_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																								TXBEIE	RXBNEIE	ERRIE	Reserved		NSSOE	TXDMA	RXDMA

Reserved	Bit 31-8	—	**保留**
TXBEIE	Bit 7	R/W	**发送缓冲区空中断使能** 0：屏蔽 TXBE 中断 1：使能 TXBE 中断。TXBE 标志置 1 时产生中断请求。
RXBNEIE	Bit 6	R/W	**接收缓冲区非空中断使能** 0：屏蔽 RXBNE 中断 1：使能 RXBNE 中断。RXBNE 标志置 1 时产生中断请求。
ERRIE	Bit 5	R/W	**错误中断使能** 此位用于控制在错误状况发生时是否产生中断（SPI 模式中的 CRCERR、OVERR、MODERR）。 0：屏蔽错误中断 1：使能错误中断
Reserved	Bit 4-3	—	**保留**
NSSOE	Bit 2	R/W	**NSS 引脚输出使能** 0：在主模式下禁止 NSS 输出，可在多主模式配置下工作 1：在主模式下使能 NSS 输出，不能在多主模式环境下工作
TXDMA	Bit 1	R/W	**发送缓冲区 DMA 使能** 当此位置 1 时，每当 TXBE 标志置 1 时，即产生 DMA 请求。 0：关闭发送缓冲区 DMA 1：使能发送缓冲区 DMA
RXDMA	Bit 0	R/W	**接收缓冲区 DMA 使能** 当此位置 1 时，每当 RXBNE 标志置 1 时，即产生 DMA 请求。 0：关闭接收缓冲区 DMA 1：使能接收缓冲区 DMA

19. 5. 2. 3 SPI状态寄存器（SPI_STAT）

SPI 状态寄存器 （SPI_STAT）

偏移地址：08_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																								BUSY	OVERR	MODERR	CRCERR	Reserved		TXBE	RXBNE

Reserved	Bit 31-8	—	**保留**
BUSY	Bit 7	R	**忙标志** 0：SPI 不繁忙 1：SPI 忙于通信或者发送缓冲区不为空 此标志由硬件置 1 和清零。 注意：BUSY 标志必须谨慎使用
OVERR	Bit 6	R	**上溢错误标志** 0：未发生上溢 1：发生上溢此标志由硬件置 1，可由软件序列复位。
MODERR	Bit 5	R	**模式错误** 0：未发生模式故障 1：发生模式故障 此标志由硬件置 1，可由软件序列复位。
CRCERR	Bit 4	R/C_W0	**硬件 CRC 错误标志** 0：接收到的 CRC 值与 SPI_RXCRC 值匹配 1：接收到的 CRC 值与 SPI_RXCRC 值不匹配 此标志由硬件置 1，通过软件写入 0 来清零。
Reserved	Bit 3-2	—	**保留**
TXBE	Bit 1	R	**发送缓冲区为空** 0：发送缓冲区非空 1：发送缓冲区为空
RXBNE	Bit 0	R	**接收缓冲区非空** 0：接收缓冲区为空 1：接收缓冲区非空

注：如果 TXBE 或 RXBNE 位分别被禁止，则波特率计数器会停止计数。

19. 5. 2. 4 SPI数据寄存器（SPI_DATA）

SPI 数据寄存器 （SPI_DATA）																															
偏移地址：$0C_H$																															
复位值：$00000000\_00000000\_00000000\_00000000_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																VALUE															

Reserved	Bit 31-16	—	保留
VALUE	Bit 15-0	R/W	**传输数据** 已接收或者要发送的数据。 数据寄存器分为 2 个缓冲区，一个用于写入（发送缓冲区），一个用于读取（接收缓冲区）。对数据寄存器执行写操作时，数据将写入发送缓冲区，从数据寄存器执行读取时，将返回接收缓冲区中的值。

19. 5. 2. 5 SPI CRC多项式寄存器（SPI_CRCPOLY）

SPI CRC 多项式寄存器 （SPI_CRCPOLY）																															
偏移地址：10_H																															
复位值：$00000000\_00000000\_00000000\_00000111_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																VALUE															

Reserved	Bit 31-16	—	保留
VALUE	Bit 15-0	R/W	**CRC 多项式寄存器** 此寄存器包含用于 CRC 计算的多项式。CRC 多项式（0007h）是此寄存器的复位值。可根据需要配置另一个多项式。

19. 5. 2. 6 SPI RX CRC寄存器（SPI_RXCRC）

SPI RX CRC 寄存器 （SPI_RXCRC）																															
偏移地址：14_H																															
复位值：$00000000\_00000000\_00000000\_00000000_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																CRCVAL															

Reserved	Bit 31-16	—	保留
CRCVAL	Bit 15-0	R	**接收 CRC 值** 使能 CRC 计算后，RXCRC[15:0]位将包含后续接收字节在计算后所得到的 CRC 值。当 SPI_CON1 寄存器中的 CRCEN 位写入 1 时，此寄存器复位。CRC 通过 SPI_CRCPOLY 寄存器中编程的多项式连续计算。数据帧长度设置为 8 位数据（SPI_CON1 的 FLEN 位清零）时， 仅考虑 8 个 LSB 位。 CRC 计算依据任意 CRC8 标准进行。 选择 16 位数据帧长度（SPI_CON1 寄存器的 FLEN 位置 1）时，考虑此寄存器的全部 16 个位。CRC 计算依据任意 CRC16 标准进行。 注意：当 BUSY 标志置 1 时，读取此寄存器可能返回一个不正确的值。

19. 5. 2. 7 SPI TX CRC 寄存器（SPI_TXCRC）

SPI TX CRC 寄存器 （SPI_TXCRC）																															
偏移地址：18_H																															
复位值：$00000000\_00000000\_00000000\_00000000_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																CRCVAL															

Reserved	Bit 31-16	-	保留
CRCVAL	Bit 15-0	R	**发送 CRC 值** 使能 CRC 计算后，TXCRC[7:0] 位将包含后续发送字节在计算后所得到的 CRC 值。当 SPI_CON1 寄存器中的 CRCEN 位写入 1 时，此寄存器复位。CRC 通过 SPI_CRCPOLY 寄存器中编程的多项式连续计算。 数据帧长度设置为 8 位数据（SPI_CON1 的 FLEN 位清零）时，仅考虑 8 个 LSB 位。CRC 计算依据任意 CRC8 标准进行。 选择 16 位数据帧长度（SPI_CON1 寄存器的 FLEN 位置 1）时，考虑此寄存器的全部 16 个位。CRC 计算依据任意 CRC1 标准进行。注意： 当 BUSY 标志置 1 时，读取此寄存器可能返回一个不正确的值。

第20章 通用同步异步收发器（USART0~1）

20. 1 概述

通用同步异步收发器（USART）支持与外部设备进行全双工数据通信和半双工单线通信，同时支持 NRZ 标准格式。USART 提供了小数波特率发生器，可灵活配置多种波特率。USART 能够进行同步单向通信，还支持多点通信，智能卡协议，以及硬件自动流控制（CTS/RTS）。USART 还支持配合 DMA 进行数据收发。

20. 2 特性

- 全双工异步通信、单线半双工通信
- NRZ 标准格式
- USART 发送器和接收器可分别使能
- 小数波特率发生器
- 数据字长度可编程：8 位或 9 位
- 停止位可配置：支持 1 或 2 个停止位
- 校验功能：奇校验，偶校验
- 同步通信模式的时钟输出
- 智能卡模式
  - 智能卡接口支持符合 ISO 7816-3 标准中定义的异步协议智能卡
  - 智能卡工作模式下，支持 0.5 或 1.5 个停止位
- 多点通信模式
- 支持从静默模式唤醒（通过线路空闲检测或地址标记检测）
- 接收器唤醒模式：地址位（MSB，第 9 位），线路空闲
- 支持使用 DMA 进行连续通信

20. 3 结构框图

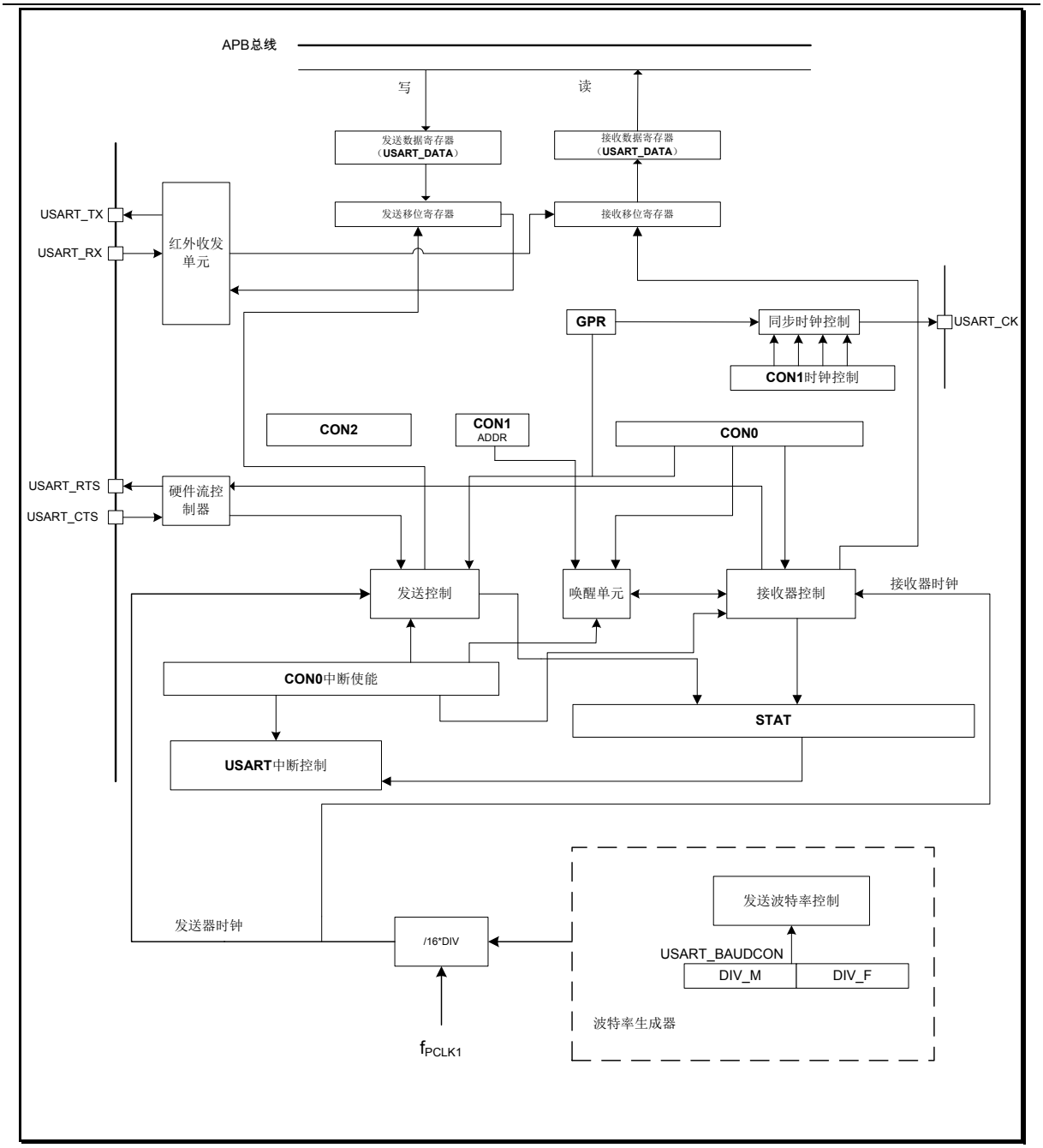


图 21-1　USART 电路结构框图

注：DIV = DIV_Mantissa + (DIV_Fraction/16))

20. 4 功能描述

20. 4. 1 引脚说明

下表为 USART 通信所用的引脚。具体的管脚配置请参考管脚分配图。

引脚	输入输出类型	说明
USART_RX	输入	串行数据输入
USART_TX	在智能卡模式和半双工模式下为双向数据口，其余情况下输出口	发送数据输出。在智能卡模式和半双工模式下，同时作为数据输出输入。
USART_SCK	时钟输出	同步模式或智能卡模式下向外部设备提供时钟
USART_RTS	输出	自动硬件流控，用于指示 USART 已准备好接收数据（低电平时）。
USART_CTS	输入	自动硬件流控，用于在当前传输结束时阻止数据发送（高电平时）。

表 21-1 USART 引脚说明

20. 4. 2 数据帧

可通过设置 USART_CON0.DLEN 位选择 8 位或 9 位字长。奇偶校验的使能和选择分别由 USART_CON0.PEN 和 USART_CON0.PSEL 控制。停止位的长度可通过设置 USART_CON1.STPLEN 进行选择。

下面给出了各个块的详细信息。

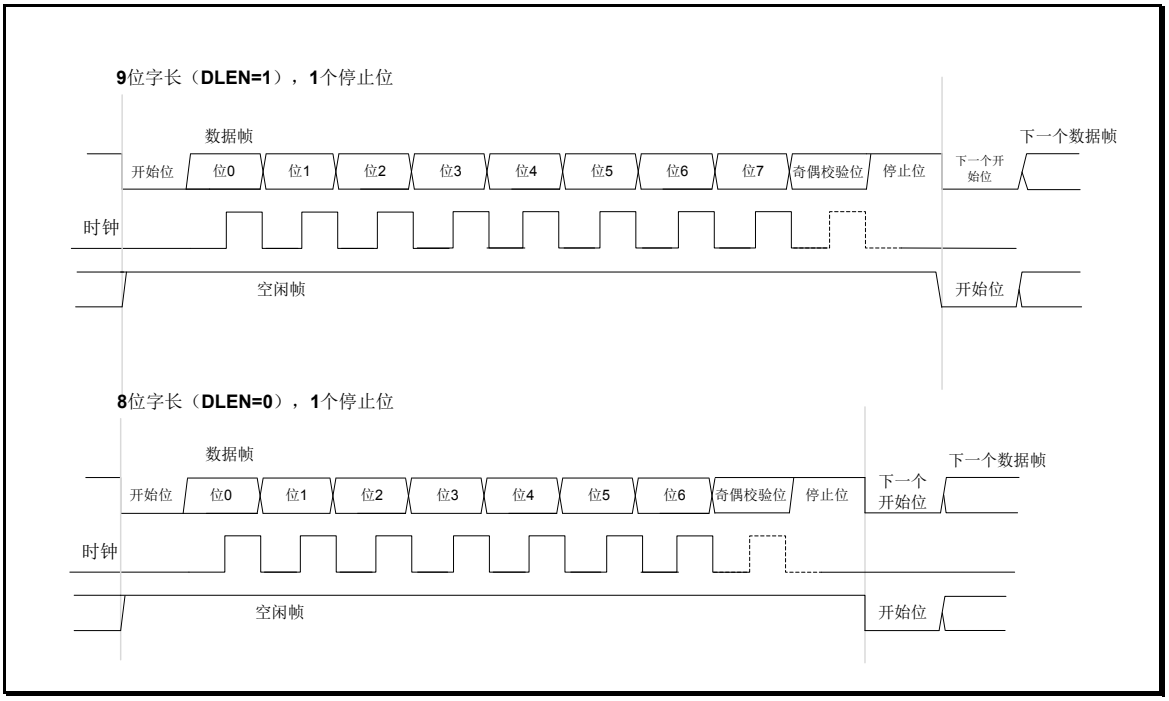


图 21-2 USART 帧格式

20.4.3 发送器

将发送使能位置位（USART_CON0.TXEN=1）后，发送移位寄存器中的数据在 TX 引脚上输出。若是同步模式或智能卡模式，则相应的时钟脉冲在 SCK 引脚输出。

USART 发送期间，LSB 首先被发送，MSB 最后被发送。在数据发送期间内，一旦复位 TXEN，则当前传输的数据将会丢失。

USART 支持以下停止位：0.5、1、1.5 和 2 个停止位。

- ◇ 1 个停止位：默认值。
- ◇ 2 个停止位：正常 USART 模式和单线模式支持该值。
- ◇ 0.5 个停止位：在智能卡模式下接收数据时使用。
- ◇ 1.5 个停止位：在智能卡模式下发送和接收数据时使用。

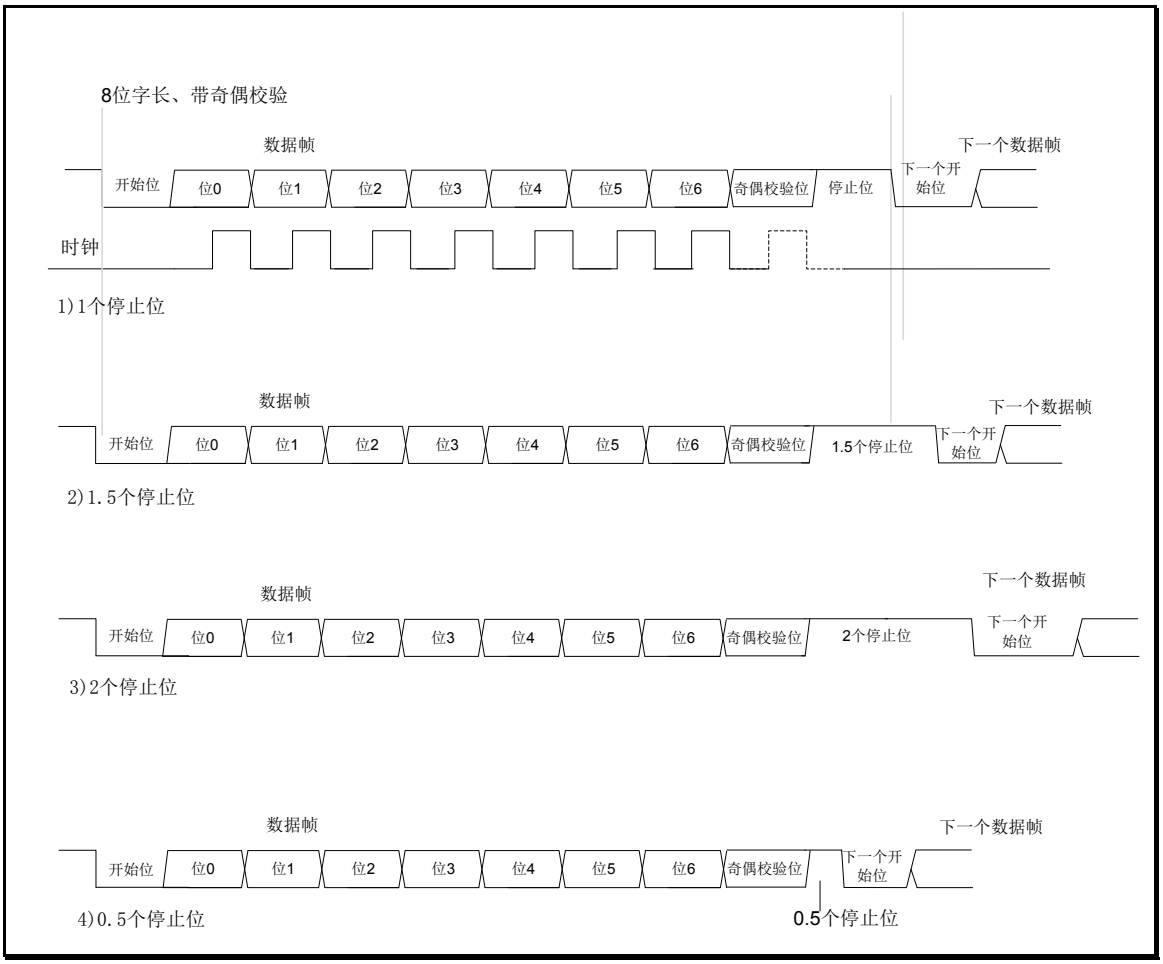


图 21-3 可配置的停止位

数据发送配置步骤：

1. 通过设置 USART_CON0. EN=1 使能 USART；
2. 通过设置 USART_CON0.DLEN 位配置通信字长；
3. 通过设置 USART_CON1. STPLEN 位配置停止位长度；
4. 若使用 DMA 搬运数据，请置位 USART_CON2.TXDMAEN。根据 DMA 模块说明配置相应 DMA 寄存器；
5. 通过设置 USART_BAUDCON 寄存器设置所需波特率；
6. 置位 USART_CON0.TXEN 位，使能发送器；
7. 在 USART_DATA.VAL 中写入要发送的数据，需要注意该操作将会清零 USART_STAT.TXEMPIF 位。重复此步，直至待发数据已全部发送完成；
8. 向 USART_DATA.VAL 写入最后一个数据后，等待至 USART_STAT.TXCIF=1。这表明最后一个帧的传送已完成。此步骤不可省略，以避免损坏最后一次发送的数据。

USART_STAT.TXEMPIF 位可通过向 USART_DATA.VAL 写入数据来清零。

USART_STAT.TXEMPIF 位由硬件置 1，此时：

- ◇ 数据已从 USART_DATA 寄存器移到移位寄存器中；
- ◇ USART_DATA 寄存器为空，此时可写入下一个数据。

USART_CON0.TXEMPIE 位置 1 时该标志位会生成中断。

如果 USART 正在发送数据，对 USART_DATA 寄存器写入的数据将存放在 TDATA 寄存器中，该数据在当前发送结束时复制到移位寄存器中。

如果 USART 处于空闲，对 USART_DATA 写操作直接将数据置于移位寄存器中，数据发送开始时，USART_STAT.TXEMPIF 位即置 1。

如果当前传输已完成且 USART_STAT.TXEMPIF=1，USART_STAT.TXCIF 位将变为高电平，若此时 USART_CON0.TXCIE 位置 1，将生成中断。

USART_STAT.TXCIF 位可通过以下软件序列清零：

1. 从 USART_STAT 寄存器读取数据。
2. 向 USART_DATA 寄存器写入数据。

注：还可通过向 TXCIF 位写入“0”将其清零。建议仅在多缓冲区通信时使用此清零序列。

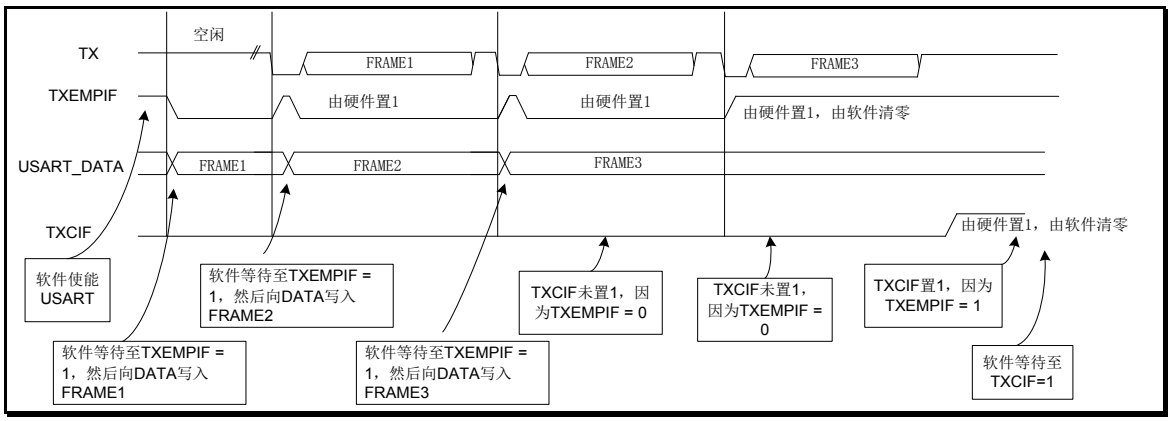


图 21-4　发送时的 TXCIF/TXEMPIF 行为

20.4.4 接收器

USART 可接收 8 位或 9 位的数据字，具体取决于 USART_CON0.DLEN 位。

USART 接收期间，首先通过 RX 引脚移入数据 LSB。该模式下，USART_DATA 寄存器的缓冲区（RDATA）位于内部总线和接收移位寄存器之间。

步骤：

1. 通过设置 USART_CON0.EN =1，使能 USART；
2. 通过设置 USART_CON0.DLEN 位，选择通信字长；
3. 通过设置 USART_CON1. STPLEN 位配置停止位长度；
4. 若使用 DMA 搬运数据，请设置 USART_CON2.RXDMAEN=1。根据 DMA 模块说明配置 DMA 寄存器；
5. 通过设置 USART_BAUDCON 寄存器设置所需波特率；
6. 通过设置 USART_CON0.RXEN=1，使能接收器，此后接收器开始搜索起始位。

接收到字符时：

1. 若 USART_STAT.RXNEIF 位被置 1。这表明移位寄存器的内容已传送到 RDATA。此时可读取已接收到的数据（以及其相应的错误标志）；
2. 若 USART_CON0.RXNEIE 位被置 1，则会生成中断；
3. 如果接收期间已检测到帧错误、噪声错误或上溢错误，相应的错误标志位将被置 1。
4. 在 DMA 模式下，每接收到一个字节后 USART_STAT.RXNEIF 均置 1，此时通过 DMA 对数据寄存器执行读操作清零。
5. 在单缓冲区模式下，通过软件对 USART_DATA 寄存器执行读操作将 RXNEIF 位清零。USART_STAT.RXNEIF 标志也可以通过向该位写入零来清零。USART_STAT.RXNEIF 位必须在结束接收下一个字符前清零，以避免发生上溢错误。

注：接收数据时，不应将 USART_CON0.RXEN 位复位。如果接收期间禁止此位，则会中止接收当前字节。

接收到断开符时，USART 将会按照帧错误对其进行处理。

检测到空闲帧时，处理步骤与接收到数据的情况相同；如果 USART_CON0.IDLEIE =1，则会产生中断。

上溢错误

如果在 USART_STAT.RXNEIF 未清零时接收到字符，则会发生上溢错误。在将此位清零前，数据无法从移位寄存器传送到 RDATA 寄存器。

每接收到一个字节后，USART_STAT.RXNEIF 标志位都将置 1。当此位是 1 时，如果在接收到下一个数据或尚未处理上一个 DMA 请求时，则会发生上溢错误。

发生上溢错误时：

- ◇ USART_STAT.OVRIF 位置 1。
- ◇ RDATA 中的内容不会丢失。对 USART_DATA 执行读操作时可使用先前的数据。

◇ 移位寄存器将被覆盖。之后，上溢期间接收到的任何数据都将丢失。

◇ 若 USART_CON2.ERRIE 位为 1，则会生成中断。

◇ 通过先后对 USART_STAT 寄存器和 USART_DATA 寄存器执行读操作将 USART_STAT.OVRIF 位清除。

注：USART_STAT.OVRIF 位置 1 时表示至少 1 个数据丢失。存在两种可能：

1) 若 USART_STAT.RXNEIF=1，则最后一个有效数据存储于接收寄存器 RDATA 中并且可进行读取；

2) 若 USART_STAT.RXNEIF=0，则表示最后一个有效数据已被读取，因此 RDATA 中没有要读取的数据。接收到新（丢失）数据的同时已读取 RDR 中的最后一个有效数据时，会发生该情况。读取序列期间（在 USART_STAT 寄存器读访问与 USART_DATA 读访问之间）接收到新数据时也会发生该情况。

噪声错误

接收器采用过采样技术（除了同步模式下），可以从噪声中提取有效数据。

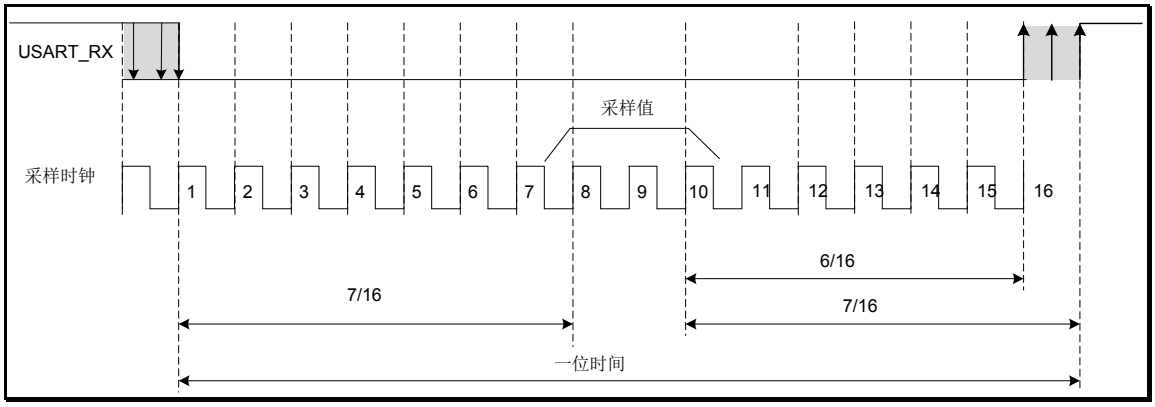


图 21-5 噪声检测时的数据采样

采样值	NDETIF 状态	接收的位值
000	0	0
001	0	0
010	0	0
011	0	1
100	0	0
101	0	1
110	0	1
111	0	1

表 21-2 通过采样数据进行噪声检测

帧中检测到噪声时：

◇ 在 USART_STAT.RXNEIF 位的上升沿时 USART_STAT.NDETIF 位置 1。

◇ 无效数据从移位寄存器传送到 USART_DATA 寄存器。

◇ 单字节通信时无中断产生。然而，在产生中断时，USART_STAT.RXNEIF 位出现上升沿。DMA 模式时，USART_CON2.ERRIE 位置 1 时将产生中断。

◇ 通过先后对 USART_STAT 寄存器和 USART_DATA 寄存器执行读操作将 USART_STAT.NDETIF 位清零。

帧错误

接收数据时未在预期时间内识别出停止位，从而出现同步失效或过度的噪声，将检测出帧错误。

检测到帧错误时：

- ◇ USART_STAT.FERRIF 位由硬件置 1
- ◇ 无效数据从移位寄存器传送到 USART_DATA 寄存器。
- ◇ 单字节通信时无中断产生。然而，在产生中断时，该位出现上升沿。DMA 模式时，USART_CON2.ERRIE 位置 1 时将产生中断。
- ◇ 通过先后对 USART_STAT 寄存器和 USART_DATA 寄存器执行读操作将 USART_STAT.FERRIF 位清零。

接收期间可配置的停止位

STPLEN 的设定值	停止位长度	说明
00	1	将在第 8、第 9 和第 10 次采样时对 1 个停止位进行采样。
01	0.5	智能卡模式下使用，不会对 0.5 个停止位进行采样。结果，选择 0.5 个停止位时，无法检测到帧错误。
10	2	采样 2 个停止位时在第 8、第 9 和第 10 次采样时对第一个停止位进行采样。如果在第一个停止位期间检测到帧错误，则帧错误标志位将会置 1。发生帧错误时不检测第 2 个停止位。RXNEIF 标志将在第一个停止位末尾时置 1。
11	1.5	在智能卡模式下发送时，设备必须检查数据是否正确发送。因此必须使能接收器块（USART_CON0 寄存器中的 RXEN =1）并检查停止位，以测试智能卡是否已检测到奇偶校验错误。发生奇偶校验错误时，智能卡会在采样时将数据信号强制为低电平，即 NACK 信号，该信号被标记为帧错误。之后，FERRIF 标志在 1.5 个停止位的末尾由 RXNEIF 置 1。在第 16、第 17 和第 18 次采样时对 1.5 个停止位进行采样（停止位采样开始后维持 1 个波特时钟周期）。1.5 个停止位可分为 2 个部分：0.5 个波特时钟周期（未发生任何动作），然后是 1 个正常的停止位周期（一半时间处进行采样）。有关详细信息，请参见智能卡部分的说明。

表 21-3 停止位长度设定

注：USART 通讯的健壮性不能过于依赖帧错误检测，未检测到帧错误并不表示通讯一定正常，建议从软件应用角度完善通讯异常检测功能。

为提高 USART 接收功能可靠性，务必使能 USART_CON0.IDLEIE，并在中断服务程序清除 USART_STAT.IDLEIF，确保在接收期间及时清除 IDLE 状态。

20.4.5 小数波特率

对 DIV 的尾数值和小数值进行编程时，接收器和发送器（Rx 和 Tx）的波特率均设置为相同值。

$$\text{Tx/Rx baud} = \frac{f_{PCLK\,1}}{(16 * \text{DIV})}$$

注：对 USART_BAUDCON 写入设定值后，波特率计数器将会更新成新的设定值。

在通信期间不能改变波特率寄存器的设定值。

从 USART_BAUDCON 的值计算 DIV：

◇ 示例 1：

如果 DIV_M = 18 且 DIV_F = 10 （USART_BAUDCON = 0x12A），那么

DIV 整数部分为 18，DIV 小数部分为 10/16 = 0.625

所以 DIV = 18.625

从 DIV 计算 USART_BAUDCON 的值：

◇ 示例 2：

若设定 DIV = 21.56

可得出:

DIV_F = 16*0.56 = 8.96

四舍五入后为 9 = 0x9（此处应四舍五入后取整，若取整后为 16，则需向整数部分进位）

DIV_M = 21 = 0x15

那么 USART_BAUDCON = 0x159 ，所以实际 USARTDIV = 15.5625

波特率		f_{PCLK1} = 24Mhz			f_{PCLK1} = 48Mhz		
索引	设定值	实际值	寄存器值	误差	实际值	寄存器值	误差
1	2400	2400	0x271	0%	2400	0x4E2	0%
2	9600	9600	0x9C4	0%	9600	0x1388	0%
3	19200	19200	0x4E2	0%	19200	0x9C4	0%
4	57600	57553	0x1A1	0.08%	57623	0x341	0.04%
5	115200	115384	0xD0	0.16%	115107	0x1A1	0.08%
6	230400	230769	0x68	0.16%	230769	0xD0	0.16%
7	460800	461538	0x34	0.16%	461538	0x68	0.16%
8	921600	923076	0x1A	0.16%	923076	0x34	0.16%

表 21-4 波特率误差

注：对于某一特定的波特率，CPU 时钟频率越高，波特率精度越高。波特率上限可使用公式确定。

20. 4. 6 接收器容差

仅当总时钟系统偏差小于 USART 接收器的容差时，USART 异步接收器才能正常工作。影响总偏差的因素包括：

- ◇ 发送器误差引起的偏差
- ◇ 接收器的波特率量化引起的误差
- ◇ 振荡器的偏差
- ◇ 传输线路引起的偏差
- ◇ 对于正常接收数据，USART 接收器的容差等于所容许的最大偏差，具体取决于以下选择：
  - 由 USART_CON0.DLEN 位定义的字长
  - 是否使用小数波特率

20.4.7 多点通信

多个 USART 连接在一个网络中。其中一个 USART 是主 USART，其 TX 输出与其它所有 USART 的 RX 输入相连接。其它 USART 均为从 USART，其 TX 输出在逻辑上通过与运算和主 USART 的 RX 连在一起。

通常只有接收方主动接收消息内容，未被寻址的 USART 忽略线路上的信息。

通过静默功能可将未被寻址的器件置于静默模式。在静默模式下：

- ◇ 任何接收状态位都不会被设置。
- ◇ 禁止任何接收中断。
- ◇ USART_CON0.RXWK 位置 1，RXWK 可由硬件自动控制，或在特定条件下由软件写入。

根据 USART_CON0.WKMOD 位的设置，USART 可使用两种方法进入或退出静默模式：

- ◇ 如果 USART_CON0.WKMOD 位为 0，则进行空闲线路检测；
- ◇ 如果 USART_CON0.WKMOD 位置 1，则进行地址标记检测。

空闲线路检测 （WKMOD=0）

当向 USART_CON0.RXWK 位写入 1 时，USART 进入静默模式。

当 USART 检测到空闲帧时，被唤醒。此时 USART_CON0.RXWK 位会由硬件清零，但 USART_STAT.IDLEIF 位不会置 1。还可通过软件向 USART_CON0.RXWK 位写入 0。

下图给出了使用空闲线路检测时静默模式行为的示例。

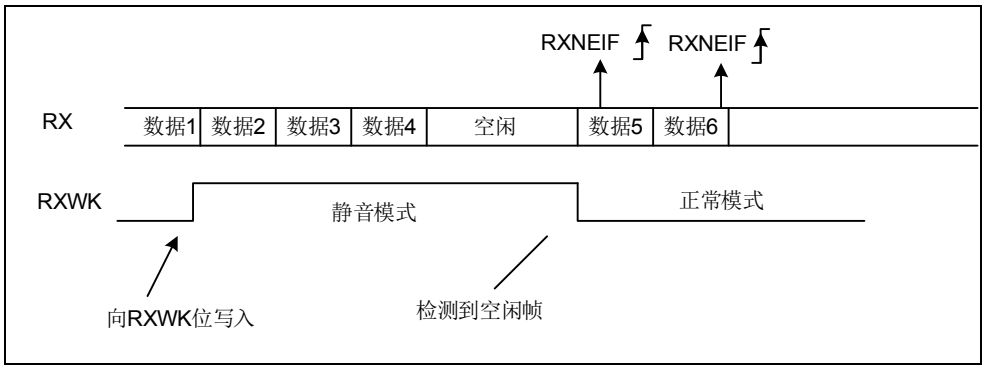


图 21-6 使用空闲线路检测时的静音模式

地址标记检测 （WKMOD=1）

如果字节的 MSB 为 1，则将这些字节识别为地址，否则将其识别为数据。

在地址字节中，目标接收器的地址位于 4 个 LSB 上。USART 将此 4 位数据与本地配置的地址进行比较，不匹配时，USART 会进入静默模式。此时，USART_CON0.RXWK 位将由硬件置 1。由于当时 USART 已经进入了静默模式，所以 USART_STAT.RXNEIF 标志不会针对此地址字节置 1，也不会触发中断或 DMA 请求。

匹配时，USART 会退出静默模式。然后 USART_CON0.RXWK 位被清零，可以开始正常接收后续字节。此时，USART_STAT.RXNEIF 位会针对地址字符置 1。

当接收器的缓冲区不包含任何数据（USART_STAT.RXNEIF=0）时，可配置

USART_CON0.RXWK 位进入或退出静默模式。

下图中给出了使用地址标记检测时静默模式行为的示例。

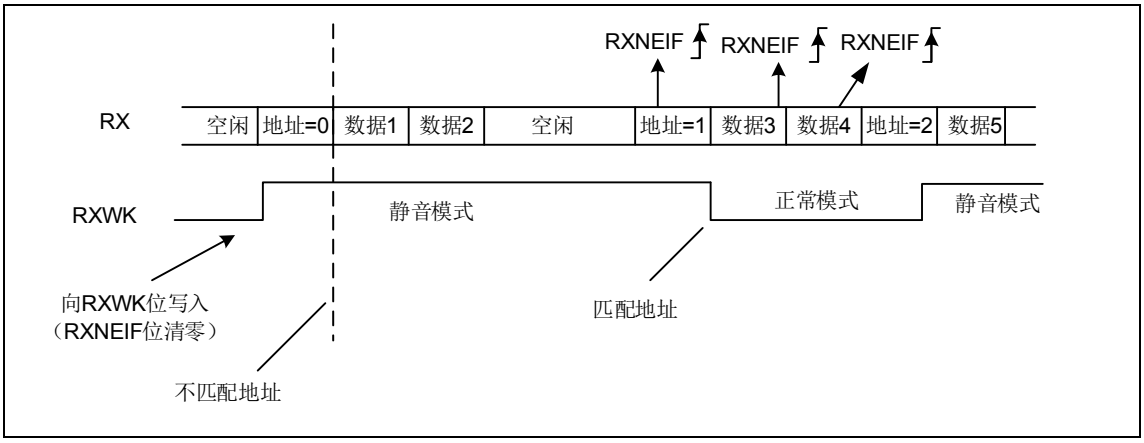


图 21-7　使用地址标记检测时的静默模式

20. 4. 8　奇偶校验控制

设置 USART_CON0.PEN 位为 1，使能奇偶校验控制。再根据 USART_CON0.DLEN 位可以确定帧格式，下表列出了 USART 帧格式。

DLEN 位	PEN 位	USART 帧格式
0	0	8 位数据+停止位
0	1	7 位数据+奇偶校验位+停止位
1	0	9 位数据+停止位
1	1	8 位数据+奇偶校验位+停止位

表 21-5　USART 帧格式

偶校验

对奇偶校验位进行计算，使数据帧和奇偶校验位中 1 的数量为偶数。

例如：数据=00100100；2 个 1=>如果选择偶校验，则校验位是 0。

奇校验

对奇偶校验位进行计算，使帧和奇偶校验位中 1 的数量为奇数。

例如：数据=00100100；2 个 1 =>如果选择奇校验，则校验位是 1。

接收时进行奇偶校验检查

如果奇偶校验检查失败，则 USART_STAT.PERRIF 标志置 1；如果 USART_CON0.PERRIE 位置 1，则会产生中断。USART_STAT.PERRIF 标志由软件序列清零（读 USART_STAT 寄存器，然后对 USART_DATA 寄存器执行读或写操作）。

注意：如果被地址标记唤醒，会使用数据的 MSB 位而非奇偶校验位来识别地址。而且，接收器不会对地址数据进行奇偶校验检查。

发送时的奇偶校验生成

如果 USART_CON0.PEN 位被置 1，则在数据寄存器中所写入数据的 MSB 位会被奇偶校

验位改写。

20. 4. 9 同步模式

通过设置 USART_CON1.SCKEN 位为 1 选择同步模式。在同步模式下，以下位需要清零：

◇ USART_CON2.SMARTEN、USART_CON2.HDPSEL 和 USART_CON2.IREN 位。

通过 USART 同步模式，用户可以在主模式下控制双向同步串行通信。SCK 引脚是 USART 发送器时钟的输出。在起始位或停止位期间，不会向 SCK 引脚发送时钟脉冲。通过配置 USART_CON1.LBCP 来确定在最后一个有效数据位期间，是否生成时钟脉冲。通过 USART_CON1.SCKPOL 位，用户可以选择时钟极性；通过配置 USART_CON1.SCKPHA 位，可以选择时钟相位。

在空闲状态期间， SCK 引脚将不发送时钟。

USART 发送器在同步模式下的工作方式与异步模式下完全相同。但是由于 SCK 与 TX 同步，因此 TX 上的数据是同步的。

在此模式下，USART 接收器的工作方式与异步模式下不同。如果 USART_CON0.RXEN=1，则数据在 SCK 上采样（上升或下降沿，取决于 USART_CON1.SCKPOL 和 USART_CON1.SCKPHA），而不进行过采样。此时必须确保建立时间和保持时间符合要求。

注意：SCK 引脚可与 TX 引脚结合使用。因此，仅当使能发送器（USART_CON0.TXEN=1）且正在发送数据时（对数据寄存器 USART_DATA 执行写入），才会提供时钟。即：没有发送数据的情况下无法接收同步数据。

当发送器和接收器都被禁止时，必须配置 USART_CON1.LBCP、USART_CON1.SCKPOL 和 USART_CON1.SCKPHA 位，以确保时钟脉冲正常工作。当使能发送器或接收器时，不能修改这些位。

建议按照相同指令将 TXEN 和 RXEN 位置 1，以尽量缩短接收器的建立时间和保持时间。

USART 同步模式只支持主模式：它不能接收或发送与输入时钟相关的数据（SCK 引脚始终为输出）。

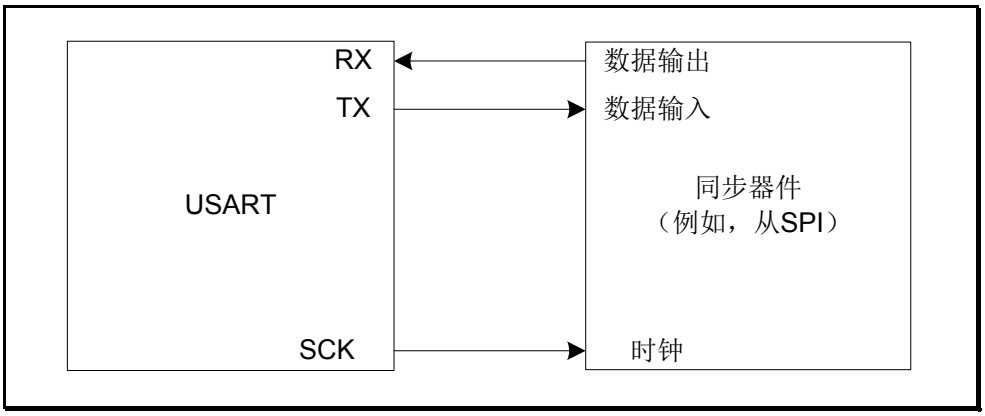


图 21-8 USART 同步发送示例

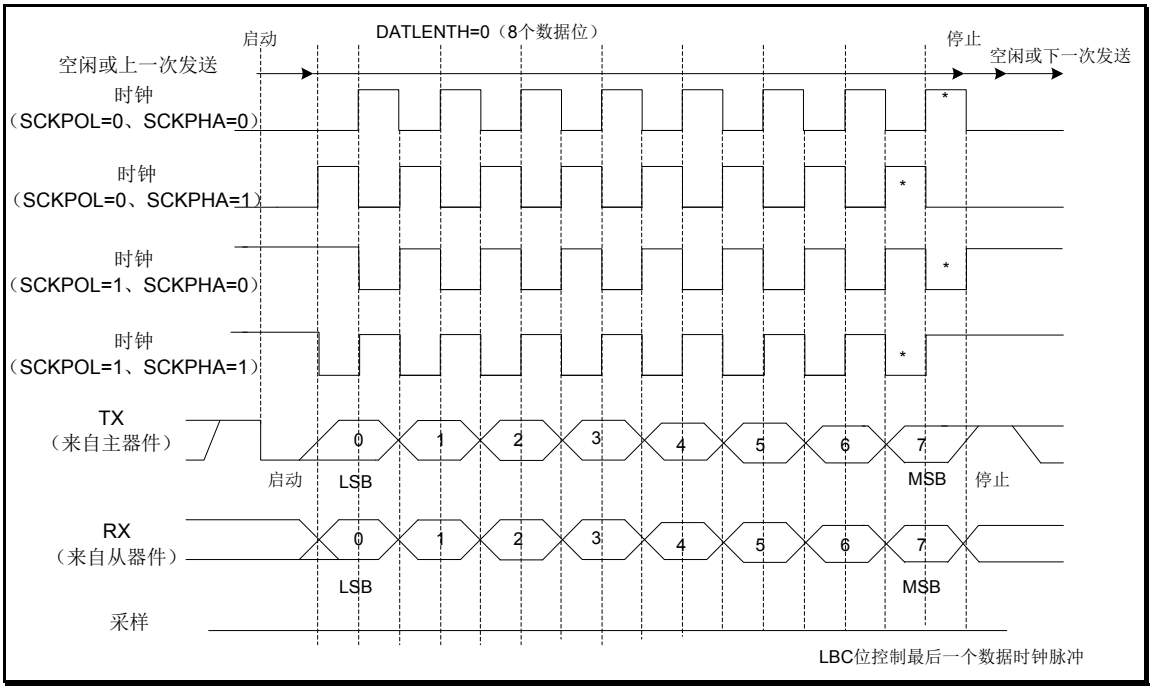


图 21-9 USART 数据时钟时序图 （DLEN=1）

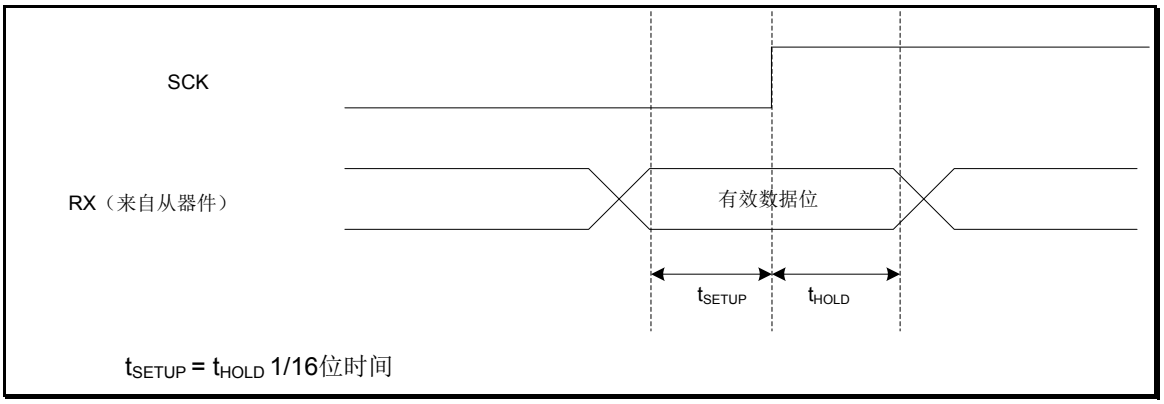


图 21-10 RX 数据建立/保持时间

注：在智能卡模式下，SCK 的功能有所不同。有关详细信息，请参见智能卡模式一章。

20. 4. 10 单线半双工通信

通过设置 USART_CON2.HDPSEL=1 选择单线半双工模式。在此模式下，以下位需要清零：

◇ USART_CON1.SCKEN 位；

◇ USART_CON2.SMARTEN 和 USART_CON2.IREN 位。

USART 单线半双工模式中 TX 和 RX 从内部相连接。使用控制位 USART_CON2.HDPSEL 选择此模式。

当 USART_CON2.HDPSEL=1 时：

◇ TX 和 RX 线路从内部相连接；

◇ RX 引脚的 RX 复用功能无效，但是该管脚其他复用功能可用；

◇ 无数据传输时，TX 引脚始终处于空闲状态。因此，它在空闲状态或接收过程中用作标准 I/O。必须将 TX 对应的 PIN 配置成开漏输出。

除此之外，通信与正常 USART 模式下的通信相似。线路上的冲突必须由软件进行管理。注意，发送过程永远不会被硬件阻塞，只要数据是在 TE 位置 1 的情况下写入数据寄存器，发送就会一直进行。

20. 4. 11 智能卡

通过设置 USART_CON2.SMARTEN=1 选择智能卡模式。在此模式下，以下位需要清零：

◇ USART_CON2.HDPSEL 和 USART_CON2.IREN 位。

此外，可能需要将 USART_CON1.SCKEN 位置 1，以便为智能卡提供时钟。

智能卡接口支持符合 ISO 7816-3 标准的异步协议智能卡。USART 应如下所示进行配置：

◇ 8 个数据位加奇偶校验：USART_CON0.DLEN=1 且 USART_CON0.PEN=1；

◇ 发送和接收时使用 1.5 个停止位：当 USART_CON1.STPLEN=0b11。

接收时也可以选择 0.5 个停止位，但为了避免在两种配置之间切换，建议发送和接收时均使用 1.5 个停止位。

下图显示了有奇偶校验错误和无奇偶校验错误时数据线上情况：

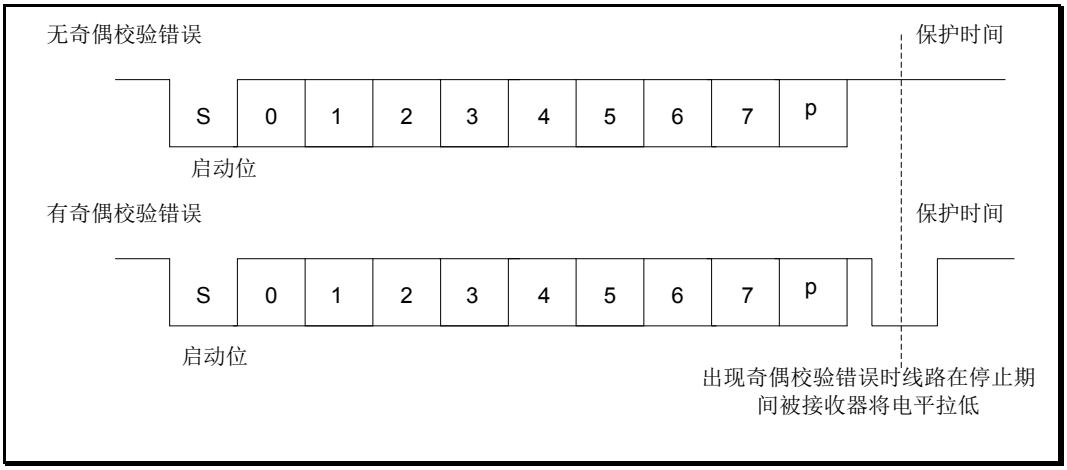


图 21-11 ISO 7816-3 异步协议

连接到智能卡时，USART 的 TX 输出会驱动一条双向线（它也由智能卡驱动）。必须将 TX 对应的引脚配置为开漏输出。

智能卡是一个单线半双工通信协议：

◇ 从发送移位寄存器发送数据会经过至少 1/2 个时钟周期的延迟。正常工作时，已满的发送移位寄存器会在下一个时钟边沿开始移位。在智能卡模式下，此发送过程还会进一步经过 1/2 波特率时钟周期的延迟。

◇ 如果在接收一个使用 0.5 或 1.5 个停止位编程的帧期间检测到奇偶校验错误，则在完成接收帧后，发送线会被拉低一个时钟周期。这是为了向智能卡指出发送到

USART 的数据尚未正确接收。此 NACK 信号（将发送线拉低 1 个时钟周期）会导致发送器端（配置为 1.5 个停止位）出现帧错误。应用程序可根据协议重新发送数据。如果 NACK 控制位置 1，则接收器会发送 “NACK”信号；否则不会发送 NACK 信号。

- ◇ 通过对保护时间寄存器进行编程，可以延迟 USART_STAT.TXCIF 标志的置位。正常工作时，当发送移位寄存器为空且没有新的发送请求出现时，会对 USART_STAT.TXCIF 标志进行置位。在智能卡模式下，空的发送移位寄存器会触发保护时间计数器，使其递增计数至保护时间寄存器中的值。在此期间，USART_STAT.TXCIF 标志被强制为低电平。当保护时间计数器达到设置值时，USART_STAT.TXCIF 置位为高电平。
- ◇ USART_STAT.TXCIF 标志的释放不受智能卡模式的影响。
- ◇ 如果在发送端检测到帧错误（由来自接收器的 NACK 信号引起），则发送端的接收器不会将 NACK 作为起始位进行检测。根据 ISO 协议，接收到的 NACK 信号的持续时间可以是 1 或 2 个时钟周期。
- ◇ 在接收端，如果检测到奇偶校验错误并发送了 NACK 信号，则接收端不会将 NACK 作为起始位进行检测。

注：在智能卡模式下带有帧错误的 0x00 数据将被视为数据，而非中断。

当翻转 USART_CON0.TXEN 位时，不会发送空闲帧。空闲帧（在其它配置中进行了定义）在 ISO 协议中未进行定义。

下图详细介绍了 USART 如何对 NACK 信号采样。在本例中，USART 正在发送数据并配置了 1.5 个停止位。USART 的接收部分已被使能，以检查数据的完整性和 NACK 信号。

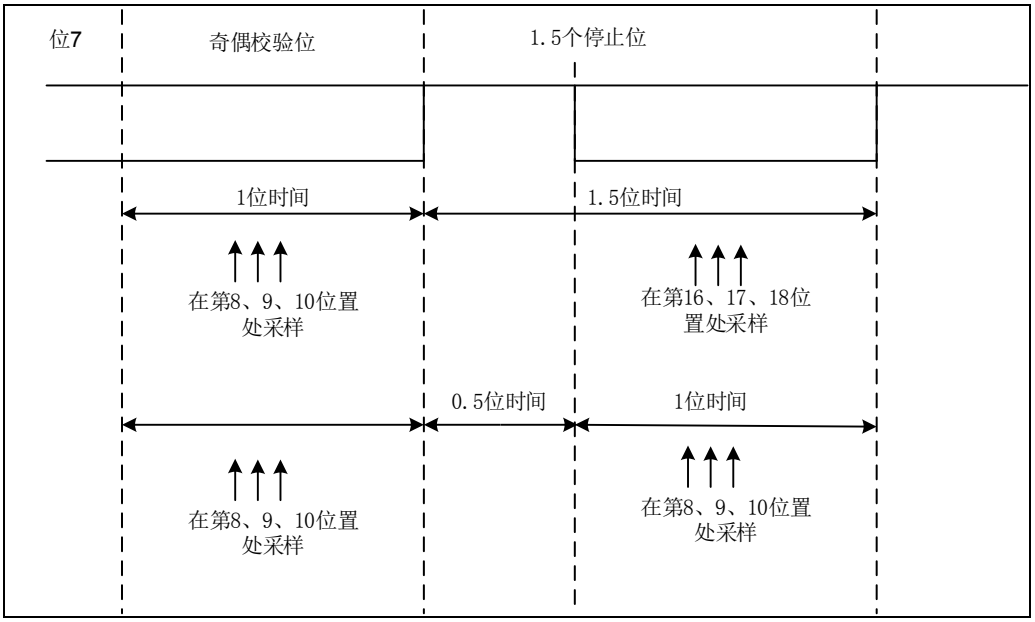


图 21-12 使用 1.5 个停止位检测奇偶校验错误

USART 可以通过 SCK 为智能卡提供时钟。在智能卡模式下，SCK 仅通过一个预分频器输出内部外设时钟。分频比在预分频器寄存器 USART_GP.PSC 中进行配置。SCK 频率可在 $f_{CK}/2$ 到 $f_{CK}/62$ 之间进行编程，其中 f_{CK} 为外设输入时钟。

20. 4. 12 连续通信（使用DMA）

USART 能够使用 DMA 进行连续通信。接收缓冲区和发送缓冲区的 DMA 请求是独立的。

使用 DMA 进行发送

设置 USART_CON2.TXDMAEN 位为 1，可以使能 DMA 模式进行发送。当 USART_STAT.TXEMPIF=1 时，DMA 可以将数据从 SRAM 区搬运到 USART_DATA 寄存器。使用 DMA 配合 USART 进行发送：

1. 在 DMA 寄存器中写入 USART_DATA 寄存器地址，将其配置为传输的目标地址，每次发生 TXEMPIF 事件后，数据都会从存储器搬运到此地址；
2. 在 DMA 寄存器中写入存储器地址，将其配置为传输的源地址。每次发生 TXEMPIF 事件后，数据都会从这个存储区域搬运到 USART_DATA 寄存器中；
3. 在 DMA 寄存器中配置要传输的总字节数；
4. 在 DMA 寄存器中选择通道，并配置通道优先级；
5. 根据应用的需求，在完成全部传输后产生 DMA 中断；
6. 向 USART_STAT.TXCIF 位写入 0，将其清零；
7. 在 DMA 寄存器中激活该通道。

当达到在 DMA 控制器中设置的数据传输量时，DMA 控制器会在 DMA 通道的中断向量上产生一个中断。

使用 DMA 进行发送时，建议同时将 TXCIF 中断使能，在 TXCIF 中断服务程序中进行关闭 USART 或进入低功耗模式等相关操作。

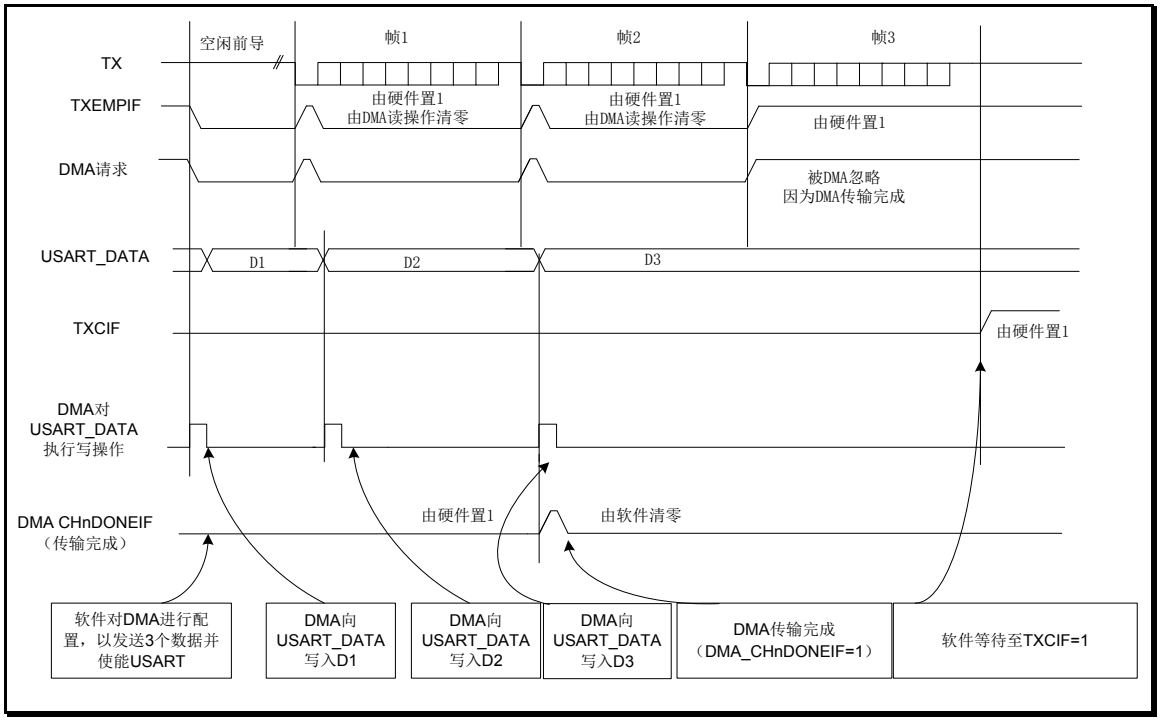


图 21-13 使用 DMA 进行发送

使用 DMA 进行接收

设置 USART_CON2.RXDMAEN 位为 1 可以使能 DMA 模式进行接收。接收数据字节时，数据会从 USART_DATA 寄存器加载到 SRAM 区域中。使用 DMA 配合 USART 进行接收：

1. 在 DMA 寄存器中写入 USART_DATA 寄存器地址，将其配置为传输的源地址。每次发生 RBNE 事件后，数据都会从此地址移动到存储器；
2. 在 DMA 寄存器中写入存储器地址，将其配置为传输的目标地址。每次发生 RBNE 事件后，数据都会从 USART_DATA 寄存器加载到此存储区；
3. 在 DMA 寄存器中配置要传输的总字节数；
4. 在 DMA 寄存器中选择通道，并配置通道优先级；
5. 根据应用的需求，在完成全部传输后产生 DMA 中断；
6. 在 DMA 寄存器中激活该通道。

当达到在 DMA 控制器中设置的数据传输量时，DMA 控制器会在 DMA 通道的中断向量上产生一个 DMA 中断。在中断服务程序中，USART_CON2.RXDMAEN 位应由软件清零。

注：如果 DMA 用于接收，则不要使能 USART_CON0.RXNEIE 位。

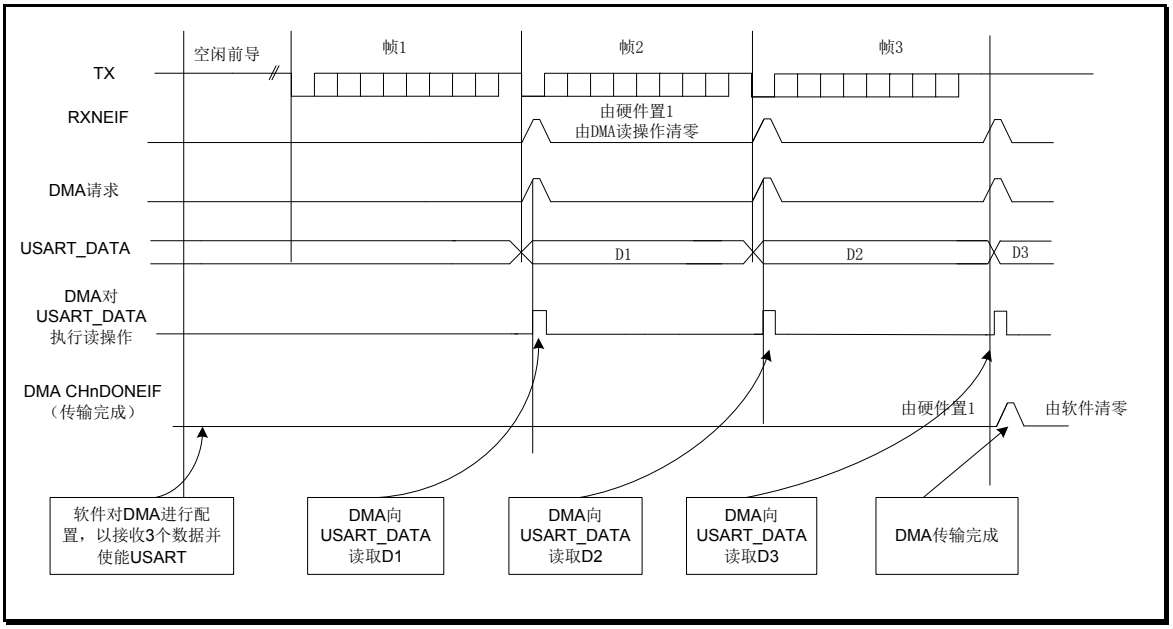


图 21-14　使用 DMA 进行接收

多缓冲区通信中的错误标志和中断生成

在多缓冲区通信中，如果事务中发生任何错误，都会在传输完当前字节后设置错误标志。如果中断使能置 1，则会产生中断。在单字节接收过程中，与 USART_STAT.RXNEIF 一同置位的帧错误、上溢错误和噪声标志具有单独的错误标志中断使能位（USART_CON2.ERRIE 位）；如果该位置 1，则会因其中任何一个错误均会在传输完当前字节后产生中断。

20. 4. 13 硬件流控制

使用 CTS/RTS 输出可以控制 2 个器件之间的数据流。下图显示了该模式下的连接方式：

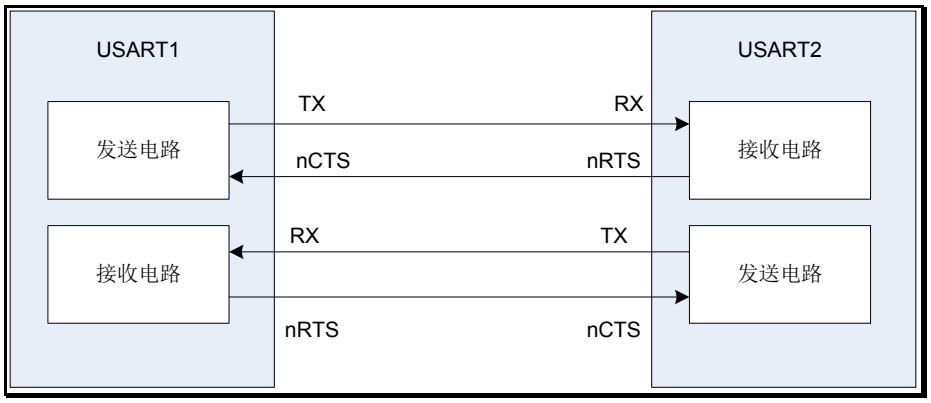


图 21-15　2 个 USART 间的硬件流控制

RTS 流控制：

使能 RTS 流控制 （USART_CON2.RTSEN=1），一旦 USART 准备好接收新数据，则将 RTS 变为有效（输出低电平）。当接收寄存器已满时，会将 RTS 变为无效（输出高电平），表明发送过程会在当前帧结束后停止。下图显示了 RTS 流控制的进行通信模式：

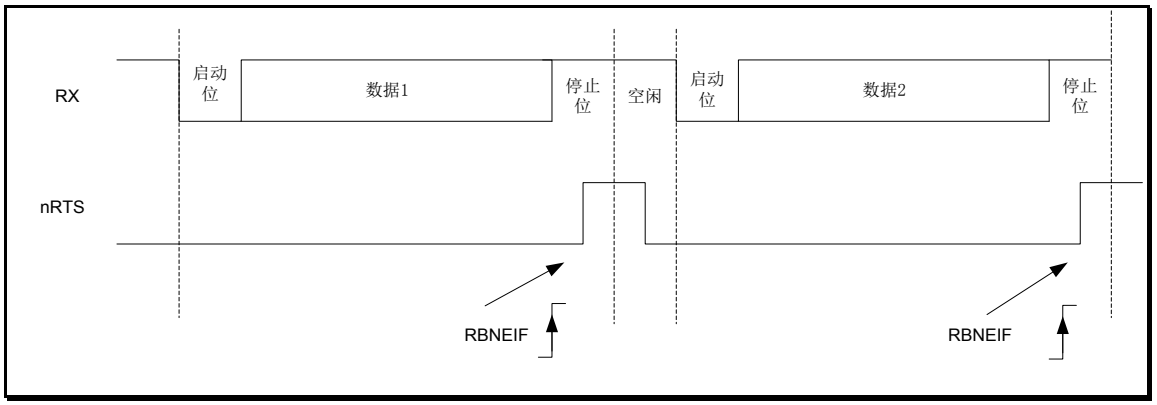


图 21-16　RTS 流控制

CTS 流控制：

使能 CTS 流控制（USART_CON2.CTSEN=1），则 USART 会在发送下一帧前检查 CTS。若 CTS 有效（接收到低电平），则会发送下一帧；否则阻止发送。若在发送过程中 CTS 变为无效（接收到高电平），则发送完当前帧之后，发送器停止。

当 USART_CON2.CTSEN=1 时，只要 CTS 发生变化，USART_STAT.CTSIF 状态位便会由硬件自动置 1。这指示接收器是否已准备好进行通信。如果 USART_CON2.CTSIE 位置 1，则会产生中断。

注：停止帧的特殊行为：使能 CTS 流控后，发送器发送停止信号时将不检查 CTS 输入状态。

20. 4. 14 中断源

状态或事件	状态标志	使能控制
接收缓存中有数据供读取	RXNEIF	RXNEIE
发送缓存空	TXEMPIF	TXEMPIE
发送完成	TXCIF	TXCIE
CTS 变化检测	CTSIF	CTSIE
接收缓存上溢错误	OVRIF	RXNEIE
空闲符检测	IDLEIF	IDLEIE
校验错误	PERRIF	PERRIE
多缓冲区通信中的噪声错误，上溢错误和帧错误	NDETIF, OVRIF, FERRIF	ERRIE

表 21-6　USART 中断请求

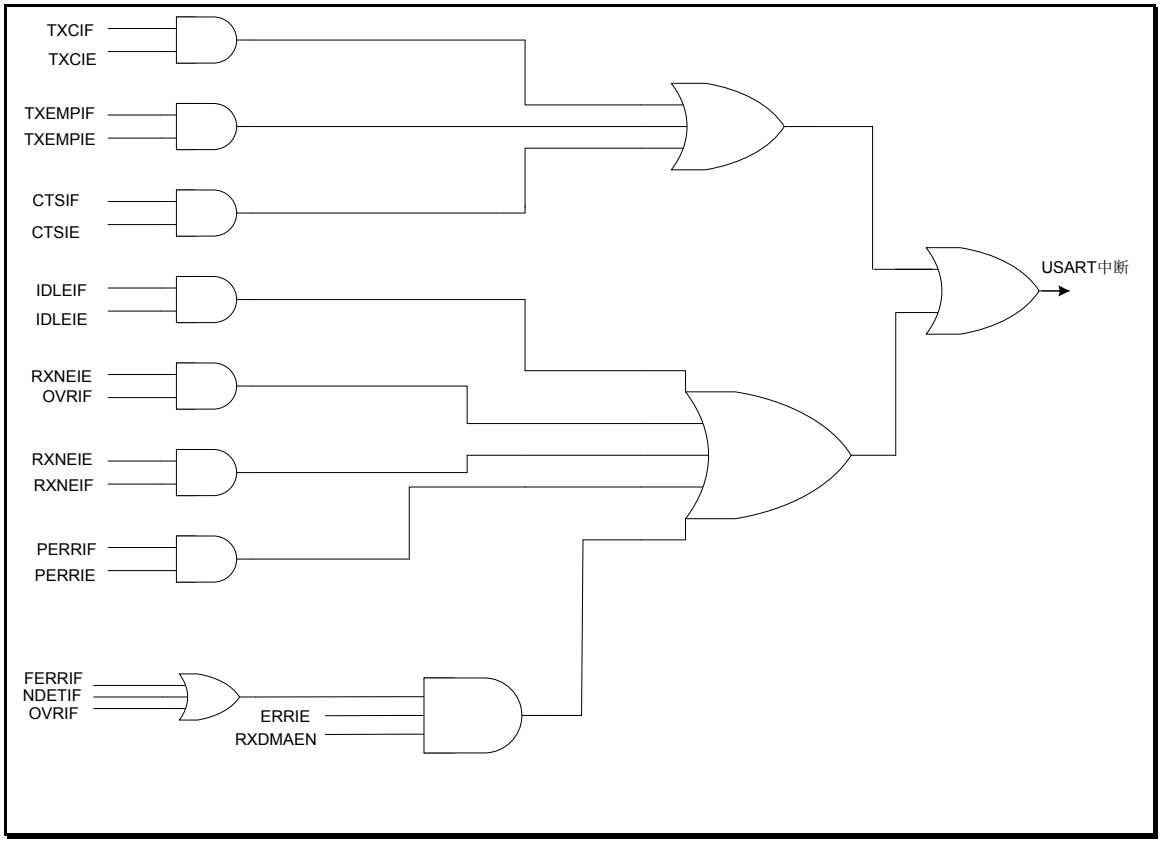


图 21-17　USART 中断映射图

20. 5 特殊功能寄存器

20. 5. 1 寄存器列表

外设寄存器可支持半字（16 位）或字（32 位）访问。

USART 寄存器列表		
名称	偏移地址	描述
USART_STAT	0000_H	USART 状态寄存器
USART_DATA	0004_H	USART 数据寄存器
USART_BAUDCON	0008_H	USART 波特率寄存器
USART_CON0	$000C_H$	USART 控制寄存器 0
USART_CON1	0010_H	USART 控制寄存器 1
USART_CON2	0014_H	USART 控制寄存器 2
USART_GP	0018_H	USART 保护时间和预分频寄存器

20. 5. 2 寄存器描述

20. 5. 2. 1 USART状态寄存器（USART_STAT）

USART 状态寄存器 （USART_STAT）

偏移地址：00_H

复位值：$00000000\_11000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																						CTSIF	Reserved	TXEMPIF	TXCIF	RXNEIF	IDLEIF	OVRIF	NDETIF	FERRIF	PERRIF

Reserved	Bit 31-10	—	**保留**
CTSIF	Bit 9	R/C_W0	**CTS 变化检测标志** 如果 CTSEN 位置 1，当 CTS 输入变换时，此位由硬件置 1。通过软件将该位清零（通过向该位中写入 0）。如果 USART_CON2 寄存器中 CTSIE =1，则会生成中断。 0：CTS 状态线上未发生变化 1：CTS 状态线上发生变化
Reserved	Bit 8	—	**保留**
TXEMPIF	Bit 7	R	**发送缓冲区空标志位** 当 TDATA 寄存器的内容已传输到移位寄存器时，该位由硬件置 1。如果 USART_CON0 寄存器中 TXEMPIE 位= 1，则会生成中断。通过对 USART_DATA 寄存器执行写入操作将该位清零。 0：数据未传输到移位寄存器 1：数据传输到移位寄存器 注意：单缓冲区发送期间使用该位。
TXCIF	Bit 6	R/C_W0	**发送完成** 如果已完成对包含数据的帧的发送并且 TXEMPIF 置 1，则该位由硬件置 1。如果 USART_CON0 寄存器中 TXCIE =1，则会生成中断。该位由软件序列清零（读取 USART_STAT 寄存器，然后写入 USART_DATA 寄存器）。也可以通过向该位写入‘0’来清零。建议仅在多缓冲区通信 时使用此清零序列。 0：传送未完成 1：传送已完成
RXNEIF	Bit 5	R/C_W0	**接收缓冲区非空标志位** 当 RDATA 移位寄存器的内容已传输到 USART_DATA 寄存器时，该位由硬件置 1。如果 USART_CON0 寄存器中 RXNEIE = 1，则会生成中断。通过对 USART_DATA 寄存器执行读入操

			作将该位清零。RXNEIF 标志也可以通过向该位写入零来清零。建议仅在多缓冲区通信时使用此清零序列。 0：未接收到数据 1：已准备好读取接收到的数据
IDLEIF	Bit 4	R	**检测到空闲线路** 检测到空闲线路时，该位由硬件置 1。如果 USART_CON0 寄存器中 IDLEIE = 1，则会生成中断。该位由软件序列清零（读入 USART_STAT 寄存器，然后读入 USART_DATA 寄存器）。 0：未检测到空闲线路 1：检测到空闲线路 注意： 直到 RXNEIF 位本身已置 1 时（即，当出现新的空闲线路时）IDLEIF 位才会被再次置 1。
OVRIF	Bit 3	R	**上溢错误** 在 RXNEIF =1 的情况下，当移位寄存器中当前正在接收的字准备好传输到 RDATA 寄存器时，该位由硬件置 1。如果 USART_CON0 寄存器中 RXNEIE = 1，则会生成中断。该位由软件序列清零（读入 USART_STAT 寄存器，然后读入 USART_DATA 寄存器）。 0：无上溢错误 1：检测到上溢错误 注意：当该位置 1 时，RDATA 寄存器的内容不会丢失，但移位寄存器会被覆盖。如果 ERRIE 位置 1， 则在进行多缓冲区通信时会对 OVRIF 标志生成一个中断。
NDETIF	Bit 2	R	**检测到噪声标志** 当在接收的帧上检测到噪声时，该位由硬件置 1。该位由软件序列清零（读入 USART_STAT 寄存器，然后读入 USART_DATA 寄存器）。 0：未检测到噪声 1：检测到噪声 注意：该位不会产生中断，因为它和 RXNEIF 一起出现，硬件会在设置 RXNEIF 标志时产生中断。在多缓冲区通信模式下，如果设置了 ERRIE 位，则设置 NDETIF 标志时会产生中断
FERRIF	Bit 1	R	**帧错误** 当检测到去同步化、过度的噪声或断开符时，该位由硬件置 1。该位由软件序列清零（读入 USART_STAT 寄存器，然后读入 USART_DATA 寄存器）。 0：未检测到帧错误 1：检测到帧错误或断开符

			注意：该位不会生成中断，因为该位出现的时间与本身生成中断的 RXNEIF 位出现的时间相同。如果当前正在传输的字同时导致帧错误和上溢错误，则会传输该字，且仅有 OVRIF 位被置 1。 如果 ERRIE 位置 1，则在进行多缓冲区通信时会对 FERRIF 标志生成一个中断。
PERRIF	Bit 0	R	**奇偶校验错误** 当在接收器模式下发生奇偶校验错误时，该位由硬件置 1。该位由软件序列清零（读取状态寄存器，然后对 USART_DATA 数据寄存器执行读或写访问）。将 PERRIF 位清零前软件必须等待 RXNEIF 标志被置 1。 如果 USART_CON0 寄存器中 PERRIE = 1，则会生成中断。 0：无奇偶校验错误 1：奇偶校验错误

20. 5. 2. 2 USART数据寄存器（USART_DATA）

数据寄存器 （USART_DATA）																															
偏移地址：04_H																															
复位值：$xxxxxxxx\_xxxxxxxx\_xxxxxxxx\_xxxxxxxx_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																							VAL								

Reserved	Bit 31-9	—	保留
VAL	Bit 8-0	R/W	**数据值** 包含接收到数据字符或已发送的数据字符，具体取决于所执行的操作是“读取”操作还是“写入”操作。 因为数据寄存器包含两个寄存器，一个用于发送（TDATA），一个用于接收 （RDATA），因此它具有双重功能（读和写）。 TDATA 寄存器在内部总线和输出移位寄存器之间提供了并行接口。 RDATA 寄存器在输入移位寄存器和内部总线之间提供了并行接口。 在使能奇偶校验位的情况下（USART_CON0 寄存器中的 PEN 位被置 1）进行发送时，由于 MSB 的写入值（位 7 或位 8，具体取决于数据长度）会被奇偶校验位所取代，因此该值不起任何作用。 在使能奇偶校验位的情况下进行接收时，从 MSB 位中读取的值为接收到的奇偶校验位。

20. 5. 2. 3 USART波特率寄存器 USART_BAUDCON）

USART 波特率寄存器 （USART_BAUDCON）																															
偏移地址：08_H																															
复位值：$00000000\_00000000\_00000000\_00000000_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																DIV_M												DIV_F			

Reserved	Bit 31-16	—	保留
DIV_M	Bit 15-4	R/W	**DIV 的整数** 这 12 个位用于定义 USART 除数（DIV）的整数 注：DIV_M 应设置不小于 2 的值
DIV_F	Bit 3-0	R/W	**DIV 的小数** 这 4 个位用于定义 USART 除数（DIV）的小数。

注：如果 TXEN 或 RXEN 位分别被禁止，则波特率计数器会停止计数。

20. 5. 2. 4 USART控制寄存器 0（USART_CON0）

USART 控制寄存器 0 （USART_CON0）

偏移地址：$0C_H$

复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																TXINV	RXINV	EN	DLEN	WKMOD	PEN	PSEL	PERRIE	TXEMPIE	TXCIE	RXNEIE	IDLEIE	TXEN	RXEN	RXWK	Reserved

Reserved	Bit 31-16	—	保留
TXINV	Bit 15	R/W	**TX引脚电平反向** TX引脚反向功能，此位由软件设置1和清除。 此位在RXEN与TXEN位为0时才可以写入。 0：TX引脚信号工作于标准逻辑电平（VDD=1/idle，Gnd=0/mark） 1：TX 引脚信号反向（VDD=0/mark，Gnd=1/idle）。 此功能可用于 TX 线上带有外部反向器时
RXINV	Bit 14	R/W	**RX引脚电平反向** RX引脚反向功能，此位由软件设置1和清除。 此位在RXEN与TXEN位为0时才可以写入。 0：RX引脚信号工作于标准逻辑电平（VDD=1/idle，Gnd=0/mark） 1：RX 引脚信号反向(VDD=0/mark，Gnd=1/idle)。 此功能可用于 RX 线上带有外部反向器时
EN	Bit 13	R/W	**USART 使能** 该位清零后，USART 预分频器和输出将停止，并会结束当前字节传输以降低功耗。此位由软件置 1 和清零。 0：禁止 USART 预分频器和输出 1：使能 USART
DLEN	Bit 12	R/W	**字长** 该位决定了字长。该位由软件置 1 或清零。 0：8 数据位 1：9 数据位
WKMOD	Bit 11	R/W	**唤醒方法** 该位决定了 USART 唤醒方法，该位由软件置 1 或清零。 0：空闲线路 1：地址标记
PEN	Bit 10	R/W	**奇偶校验使能** 该位选择硬件奇偶校验控制（生成和检测）。使能奇偶校验控制时，计算出的奇偶校验位被插入到

			MSB 位置（如果 DLEN=1，则为第 9 位；如果 DLEN=0，则为第 8 位），并对接收到的数据 检查奇偶校验位。此位由软件置 1 和清零。一旦该位置 1，PEN 在当前字节的后面处于活动 状态（在接收和发送时）。 0：禁止奇偶校验 1：使能奇偶校验
PSEL	Bit 9	R/W	**奇偶校验选择** 该位用于在使能奇偶校验生成/检测（PEN 位置 1）时选择奇校验或偶校验。该位由软件置 1 和清零。将在当前字节的后面选择奇偶校验。 0：偶校验 1：奇校验
PERRIE	Bit 8	R/W	**PERRIF 中断使能** 此位由软件置 1 和清零。 0：禁止中断 1：当 USART_STAT 寄存器中 PERRIF=1 时，生成 USART 中断
TXEMPIE	Bit 7	R/W	**TXEMPIF 中断使能** 此位由软件置 1 和清零。 0：禁止中断 1：当 USART_STAT 寄存器中 TXEMPIF=1 时，生成 USART 中断。
TXCIE	Bit 6	R/W	**传送完成中断使能** 此位由软件置 1 和清零。 0：禁止中断 1：当 USART_STAT 寄存器中 TXCIF =1 时，生成 USART 中断
RXNEIE	Bit 5	R/W	**RBNE 中断使能** 此位由软件置 1 和清零。 0：禁止中断 1：当 USART_STAT 寄存器中 OVRIF=1 或 RXNEIF=1 时，生成 USART 中断
IDLEIE	Bit 4	R/W	**IDLE 中断使能** 此位由软件置 1 和清零。 0：禁止中断 1：当 USART_STAT 寄存器中 IDLEIF=1 时，生成 USART 中断 注意：为提高通讯稳定性，务必使能该位。
TXEN	Bit 3	R/W	**发送器使能** 该位使能发送器。该位由软件置 1 和清零。 0：禁止发送器 1：使能发送器 注意：

			1：除了在智能卡模式下以外，传送期间 TXEN 位上的“0”脉冲（“0”后紧跟的是“1”） 会在当前字的后面发送一个报头（空闲线路）。 2：当 TXEN 置 1 时，在发送开始前存在 1 位的时间延迟。
RXEN	Bit 2	R/W	**接收器使能** 该位使能接收器。该位由软件置 1 和清零。 0：禁止接收器 1：使能接收器并开始搜索起始位
RXWK	Bit 1	R/W	**接收器唤醒** 该位决定 USART 是否处于静音模式。该位由软件置 1 和清零，并可在识别出唤醒序列时由硬件清零。 0：接收器处于活动模式 1：接收器处于静音模式 注意： 1：选择静音模式前（通过将 RXWK 位置 1），USART 必须首先接收一个数据字节，否则当由空闲线路检测到唤醒时，它无法于静音模式下正常工作。 2：在地址标记检测唤醒配置（WKMOD 位 = 1）中， RXNEIF 位置 1 时，RXWK 位不能由软件进行修改。
Reserved	Bit 0	—	**保留**

20. 5. 2. 5 USART控制寄存器 1（USART_CON1）

USART 控制寄存器 1 （USART_CON1）

偏移地址：10_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																		STPLEN		SCKEN	SCKPOL	SCKPHA	LBCP	Reserved				ADDR[3:0]			

Reserved	Bit 31-14	—	保留
STPLEN	Bit 13-12	R/W	**停止位** 这些位用于编程停止位。 00：1 个停止位 01：0.5 个停止位 10：2 个停止位 11：1.5 个停止位
SCKEN	Bit 11	R/W	**时钟使能** 该位允许用户使能 SCK 引脚。 0：禁止 SCK 引脚 1：使能 SCK 引脚
SCKPOL	Bit 10	R/W	**时钟极性** 该位允许用户在同步模式下选择 SCK 引脚上时钟输出的极性。它与 SCKPHA 位结合使用可获得所需 的时钟/数据关系 0：空闲时 SCK 引脚为低电平。 1：空闲时 SCK 引脚为高电平。
SCKPHA	Bit 9	R/W	**时钟相位** 该位允许用户在同步模式下选择 SCK 引脚上时钟输出的相位。它与 SCKPOL 位结合使用可获得所需的时钟/数据关系 0：在时钟第一个变化沿捕获数据 1：在时钟第二个变化沿捕获数据
LBCP	Bit 8	R/W	**最后一个位时钟脉冲** 该位允许用户在同步模式下选择与发送的最后一个数据位（MSB）关联的时钟脉冲是否必须在 SCK 引脚上输出。 0：最后一个数据位的时钟脉冲不在 SCK 引脚上输出 1：最后一个数据位的时钟脉冲在 SCK 引脚上输出 注意： 最后一位为发送的第 8 或第 9 个数据位，具体取决于 USART_CON0 寄存器中 DLEN 位所

			选择的 8 位或 9 位格式。
Reserved	Bit 7-4	—	**保留**
ADDR	Bit 3-0	R/W	**USART 节点的地址** 该位域用于指定 USART 节点的地址。 将在多点通信时于静音模式下使用该位域，以通过地址标记检测进行唤醒。 注意：使能发送器时不应对这 3 个位（SCKPOL、SCKPHA、LBCP）进行写操作。

20. 5. 2. 6 USART控制寄存器 2（USART_CON2）

USART 控制寄存器 2 （USART_CON2）

偏移地址：14_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																					CTSIE	CTSEN	RTSEN	TXDMAEN	RXDMAEN	SMARTEN	NACK	HDPSEL	Reserved	Reserved	ERRIE

Reserved	Bit 31-11	—	保留
CTSIE	Bit 10	R/W	**CTS 中断使能** 0：禁止中断 1：当 USART_STAT 寄存器中 CTSIF = 1 时，生成中断
CTSEN	Bit 9	R/W	**CTS 使能** 0：禁止 CTS 硬件流控制 1：使能 CTS 模式，仅当 CTS 输入有效（连接到 0）时才发送数据。如果在发送数据时使 CTS 输入无效，会在停止之前完成发送。如果使 CTS 有效时数据已写入数据寄存器，则 将延迟发送，直到 CTS 有效。
RTSEN	Bit 8	R/W	**RTS 使能** 0：禁止 RTS 硬件流控制 1：使能 RTS 中断，仅当接收缓冲区中有空间时才会请求数据。 发送完当前字符后应停止发送数据。 可以接收数据时使 RTS 输出有效（连接到 0）。
TXDMAEN	Bit 7	R/W	**发送 DMA 使能** 0：发送时禁止 DMA 传输 1：发送时使能 DMA 传输
RXDMAEN	Bit 6	R/W	**接收 DMA 使能** 0：接收时禁止 DMA 传输 1：接收时使能 DMA 传输
SMARTEN	Bit 5	R/W	**智能卡模式使能** 该位用于使能智能卡模式。 0：禁止智能卡模式 1：使能智能卡模式
NACK	Bit 4	R/W	**智能卡 NACK 使能** 0：出现奇偶校验错误时禁止 NACK 发送 1：出现奇偶校验错误时使能 NACK 发送
HDPSEL	Bit 3	R/W	**半双工选择** 选择单线半双工模式 0：未选择半双工模式 1：选择半双工模式

Reserved	Bit 2-1	—	**保留**
ERRIE	Bit 0	R/W	**错误中断使能** 对于多缓冲区通信（USART_CON2 寄存器中 RXDMAEN = 1），如果发生帧错误、上溢错误或出现噪声标志（USART_STAT 寄存器中 FERRIF = 1 或 OVRIF = 1 或 NDETIF = 1），则需要使用错误中断使能位来使能中断生成。 0：禁止中断 1：当 USART_CON2 寄存器中的 RXDMAEN = 1 并且 USART_STAT 寄存器中的 FERRIF = 1 或 OVRIF = 1 或 NDETIF = 1 时，将生成中断。

20. 5. 2. 7 USART保护时间和预分频寄存器（USART_GP）

USART 保护时间和预分频寄存器 （USART_GP）																															
偏移地址：18_H																															
复位值：$00000000\_00000000\_00000000\_00000000_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																GTVAL								PSC							

Reserved	Bit 31-16	—	保留
GTVAL	Bit 15-8	R/W	**保护时间值** 该位域提供保护时间值（以波特时钟数为单位）。 该位用于智能卡模式。经过此保护时间后，发送完成标志置 1。
PSC	Bit 7-0	R/W	**预分频器值** — 在正常模式下：PSC 必须设置为 00000001。 — 在智能卡模式下：PSC[4:0]：预分频器值用于编程预分频器，进行系统时钟分频以提供智能卡时钟。 将寄存器中给出的值（5 个有效位）乘以 2 得出源时钟频率的分频系数： 00000：保留 - 不编程此值 00001：源时钟 2 分频 00010：源时钟 4 分频 00011：源时钟 6 分频 ... 注意：如果使用智能卡模式，则位[7:5]不起作用。

第21章 模数转换器（ADC）

21. 1 概述

该 ADC 为逐次逼近型模数转换器，采样精度为 12 位，最多可以测量 16 个外部信号（对应通道 0~15）和 4 个内部信号，分别是通道 19 对应内部参考电压(VREF 1.0V)、通道 18 对应 1/4 VDD 电压、通道 17 对应温感通道和通道 16 对应内部测试信号（不可用）。通道的转换可选择单次、连续、扫描或不连续等采样模式，其采样结果存储在 16 位数据寄存器，数据存储格式可以选择左对齐或右对齐存储。

ADC 模块具有模拟看门狗特性，允许应用程序检测输入电压是否超过了用户设定的阈值上限或下限。

21. 2 特性

- 可配置的转换精度（6/8/10/12 位）
- 支持单次或连续工作模式
- 在标准转换、插入转换结束后以及发生模拟看门狗或溢出事件时产生中断
- 用于自动将通道 0 转换为通道“n”的扫描模式
- 可配置的数据对齐方式
- 可独立设置各通道采样时间
- 可配置外部触发选项，可为标准转换和插入转换配置极性
- 支持不连续采样模式
- 可配置的参考源选择
- 可配置的转换时钟分频
- 支持标准数据转换的 DMA 请求标准

21. 3 结构框图

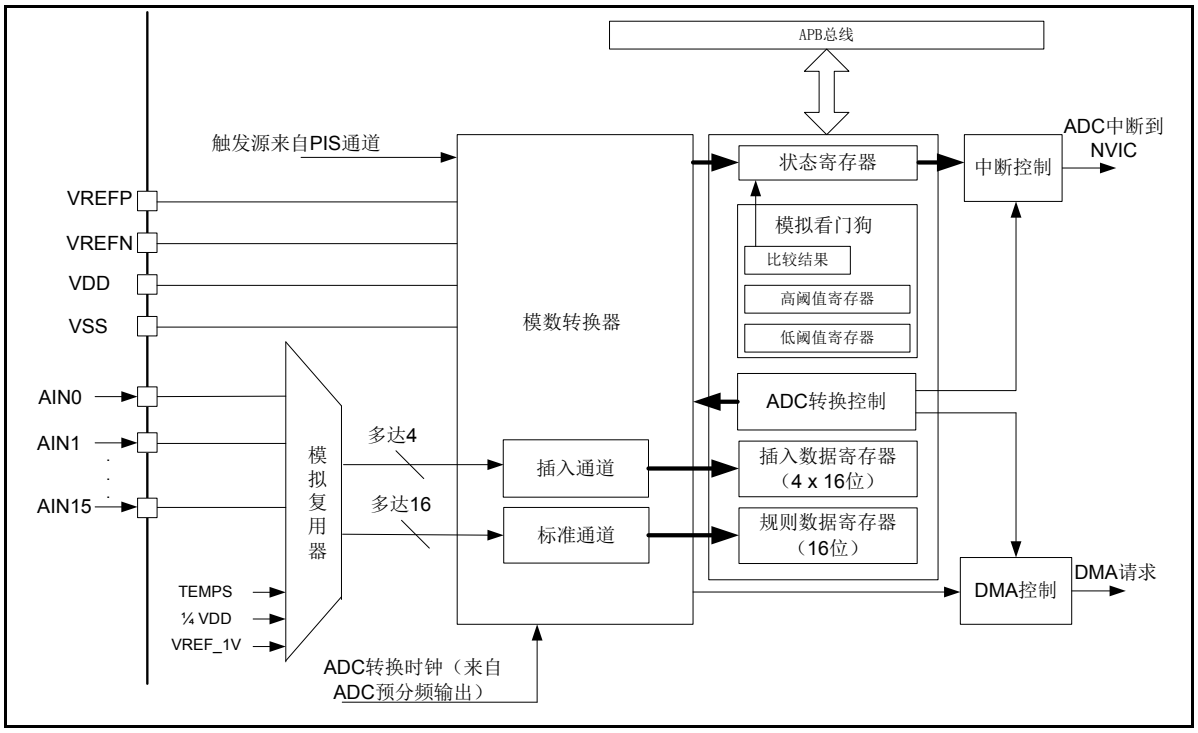


图 22-1 ADC 结构框图

21.4 功能描述

21.4.1 ADC控制

通过将 ADC_CON1 寄存器中的 ADCEN 位置 1 来使能 ADC。

通过将 ADC_CON1 寄存器中的 NCHTRG 或 ICHTRG 位置 1 来启动 AD 转换。

通过将 ADC_CON1 寄存器中的 ADCEN 位清零来关闭 ADC。

21.4.2 ADC时钟

ADC 内部工作需要两路时钟：

◇ 用于模拟电路的时钟：ADCCLK

此时钟来自于经可编程预分频器分频的 APB 时钟，该预分频器可将 APB 时钟产生 1~128 分频时钟供 ADC 模拟电路使用。

◇ 用于数字接口的时钟（用于寄存器读/写访问）

此时钟为 APB 时钟。

21.4.3 通道控制

外部 16 条复用通道和内部 4 条专用通道可分为两组：标准转换和插入转换，每个组包含一个转换序列，该序列可按任意顺序在任意通道上完成。例如，可按以下顺序对序列进行转换：AIN3、AIN8、AIN2、AIN2、AIN0、AIN2、AIN2、AIN15。

◇ 一个标准转换组最多由 16 个转换构成。必须在 ADC_NCHSx 寄存器中选择转换序列的标准通道及其顺序。标准转换组中的转换总数必须写入 ADC_CHSL 寄存器中的 NSL 位。

◇ 一个插入转换组最多由 4 个转换构成。必须在 ADC_ICHS 寄存器中选择转换序列的插入通道及其顺序。插入转换组中的转换总数必须写入 ADC_CHSL 寄存器中的 ISL 位。

如果在转换期间修改 ADC_NCHSx 或 ADC_ICHS 寄存器，将复位当前转换并向 ADC 发送一个新的启动脉冲，以转换新选择的组。

21.4.4 单次工作模式

单次工作模式是指 ADC 执行一次转换，其工作流程如下：

◇ 将 ADC_CON1.CM 位清 0，然后通过后续方式来启动此模式：

◇ 将 ADC_CON1 寄存器中的 NCHTRG 位置 1（仅适用于标准通道）

◇ 将 ADC_CON1 寄存器中的 ICHTRG 位置 1（仅适用于插入通道）

◇ 外部触发（适用于标准通道或插入通道）

完成所选通道的转换之后：

◇ 使用标准通道：

- 转换数据存储在 16 位 ADC_NCHDR 寄存器中
- 标准转换结束标志 ADC_STAT.NCHE 置 1
- 若标准转换完成中断使能位 ADC_CON0.NCHEIE 置 1，将产生中断

◇ 使用插入通道：

- 转换数据存储在 16 位 ADC_ICHDRx 寄存器中
- 插入转换结束标志 ADC_STAT.ICHE 置 1
- 若插入转换完成中断使能位 ADC_CON0.ICHEIE 置 1，将产生中断

然后，ADC 停止。

21.4.5 连续工作模式

连续工作模式是指 ADC 执行一次转换任务后，立即执行下一次转换任务，其工作流程如下：

◇ 将 ADC_CON1.CM 位置 1

◇ 通过外部触发或 ADC_CON1.NCHTRG 置 1，来启动此模式（仅适用于标准通道）

每次转换之后：

◇ 如果转换了标准通道组：

- 上次转换的数据存储在 16 位 ADC_NCHDR 寄存器中
- 标准转换结束标志 ADC_STAT.NCHE 置 1
- 若标准转换完成中断使能位 ADC_CON0.NCHEIE 置 1，将产生中断

注 1：连续转换模式不适用于插入组通道。

注 2：连续转换模式下唯一的例外情况是，在使能 ADC_CON0.IAUTO 时，插入通道组在标准通道之后自动转换。

21.4.6 时序图

在使能 ADC 后，需要一段稳定时间 t_{STAB}，然后再启动 ADC 转换，并经过若干个时钟周期后 CHE（NCHE 或 ICHE）标志置 1，如下图所示。

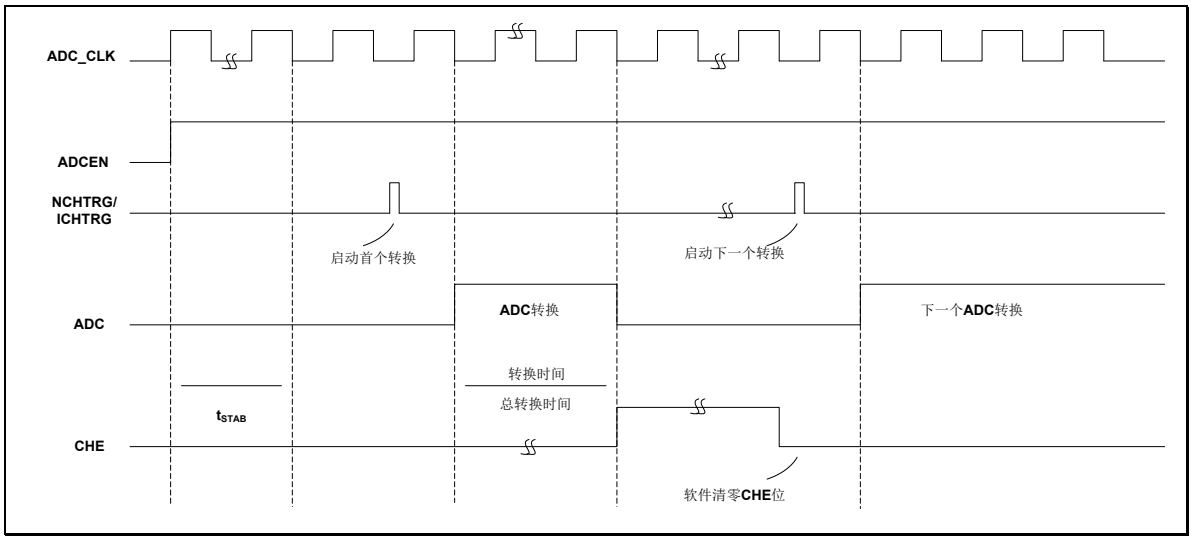


图 22-2 ADC 转换时序图

21.4.7 模拟看门狗

如果 ADC 转换的模拟电压低于阈值下限（寄存器 ADC_WDTL）或高于阈值上限（寄存器 ADC_WDTH），则 AWD 模拟看门狗状态位 ADC_STAT.AWDF 置 1。使能 ADC_CON0.AWDIE 位可以打开模拟看门狗中断。

上下限阈值寄存器是低 12 位有效的寄存器，而 ADC 结果寄存器可以左对齐和右对齐，但看门狗是在对齐之前将模拟电压与阈值上限和下限进行比较。

下表介绍了应如何配置 ADC_CON0 寄存器才能在一个或多个通道上使能模拟看门狗。

模拟看门狗保护通道	AWDSGL	NCHWDEN	ICHWDTEN
无	X	0	0
所有插入通道	0	0	1
所有标准通道	0	1	0
所有标准通道和插入通道	0	1	1
单个插入通道(AWDCH 选择)	1	0	1
单个标准通道(AWDCH 选择)	1	1	0
单个标准/插入通道(AWDCH 选择)	1	1	1

表 22-1 模拟看门狗通道选择

使用模拟看门狗流程

1. 设置 ADC_CON0.AWDSGL 位来选择看门狗作用于单个通道或一组通道，清 0 选择一组通道，置 1 选择单个通道。选择单个通道时可通过 AWDCH 选择哪个通道。

2. 置位 ADC_CON0.NCHWDEN 或 ADC_CON0.ICHWDTEN 选择使能标准组或插入组通道。

21.4.8 通道扫描

扫描模式是指连续依次转换标准组所有通道或者插入组所有通道。通过将ADC_CON0.SCANEN 位置 1 来选择扫描模式。ADC 会扫描在 ADC_NCHSx 寄存器（对于标准通道）或 ADC_ICHS 寄存器（对于插入通道）中选择的所有通道，为组中的每个通道依次执行一次转换。如果将 CM 位置 1，并且是标准组扫描，ADC 会循环转换标准组所有通道。

如果将 ADC_CON1.DMA 位置 1，则在每次通道转换结束标志置位后通过 DMA 控制器将转换自标准通道组的数据寄存器（ADC_NCHDR）传输到 SRAM。

在以下情况下，ADC_STAT 寄存器中的 NCHE 位置 1：

- ◇ 如果 ADC_CON1.NCHESEL 位清零，在每个标准组序列转换结束时
- ◇ 如果 ADC_CON1.NCHESEL 位置 1，在每个标准通道转换结束时

从插入通道转换的数据始终存储在 ADC_ICHDRx 寄存器中。

21.4.9 插入通道控制

触发插入

要使用触发插入，必须将 ADC_CON0.IAUTO 位清零。使用触发插入时，必须确保触发事件之间的间隔长于插入序列。

在插入转换期间出现标准通道转换事件，插入转换不会被中断，标准转换在插入转换结束时执行。

在标准组转换期间触发插入：

1. 通过外部触发或将 ADC_CON1.NCHTRG 位置 1 来启动标准通道组转换。

2. 如果在标准通道组转换期间出现外部插入触发或者 ADC_CON1.ICHTRG 位置 1，则当前的转换会被打断，并且插入通道序列会切换为单次扫描模式。

3. 然后，标准通道组的标准转换会从上次中断的标准转换处恢复。

在插入转换期间触发插入

自动插入

如果将 IAUTO 位置 1，则插入组中的通道会在标准组通道之后自动转换。这可用于转换最多由 20 个 AD 转换构成的序列，这些转换在 ADC_NCHSx 和 ADC_ICHS 寄存器中编程。

在此模式下，必须禁止插入通道上的外部触发。

如果 CM 位和 IAUTO 位均已置 1，则在转换标准通道之后会继续转换插入通道。

21.4.10 不连续采样控制

标准组

可将 ADC_CON0.NCHDCEN 位置 1 来使能此模式。该模式可用于转换含有 n (n≤8)个转换的短序列，该短序列是在 ADC_NCHSx 寄存器中选择的转换序列的一部分。可通过写入 ADC_CON0.ETRGN 位来指定 n 的值。

出现外部触发时，将启动在 ADC_NCHSx 寄存器中选择的 n 个转换，直到序列中的所有转换均完成为止。通过 ADC_CHSL.NSL 位定义总序列长度。

示例：

n = 4，ADC_CHSL.NSL=9，要转换的通道 = 0、2、3、5、6、7、8、10、11

第 1 次触发：转换序列 0、2、3、5

第 2 次触发：转换序列 6、7、8、10

第 3 次触发：转换序列 11 并生成 NCHE 事件

第 4 次触发：转换序列 0、2、3、5

在不连续采样模式下转换标准组时，最后一次触发剩余的通道不足 ADC_CON0.ETRGN 指定的通道数时，不会接着从头转换，转换到序列最后一个通道就会停止，生成 NCHE 事件。转换完所有序列通道后，下一个触发信号将启动序列的第一个通道。在上述示例中，第 4 次触发重新转换了序列的 0、2、3、5 通道。

插入组

可将 ADC_CON0 寄存器中的 ICHDCEN 位置 1 来使能此模式。在出现外部触发事件之后，可使用该模式逐通道（n=1）转换在 ADC_ICHS 寄存器中选择的序列。

出现外部触发时，将启动在 ADC_ICHS 寄存器中选择的下一个通道转换，直到序列中的所有转换均完成为止。通过 ADC_CHSL 寄存器中的 ISL 位定义总序列长度。

示例：

n = 1，ADC_CHSL.ISL=3，要转换的通道 = 1、3、4

第 1 次触发：转换通道 1

第 2 次触发：转换通道 3

第 3 次触发：转换通道 4 并生成 ICHE 事件

第 4 次触发：通道 1

转换完所有插入通道后，下一个触发信号将启动第一个插入通道的转换。在上述示例中，第 4 次触发重新转换了第 1 个插入通道。

不能同时使用自动插入和不连续采样模式。

不得同时为标准组和插入组设置不连续采样模式。只能选择其一进行不连续采样模式。

21. 4. 11 数据对齐

ADC_CON1.ALIGN 位用于选择转换后存储的数据的对齐方式,可选择左对齐和右对齐两种方式，如下图所示。

插入通道组的转换数据将加上 ADC_ICHOFFx 寄存器中写入的用户自定义偏移量,因此结果可以是一个负值，图中 EXS 位表示扩展的符号值。

对于标准组中的通道，不会减去任何偏移量，因此只有 12 个位有效。

标准组

0	0	0	0	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0

插入组

EXS	EXS	EXS	EXS	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0

图 22-3 右对齐数据示意图

标准组

D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	0	0	0	0

插入组

EXS	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	0	0	0

图 22-4 左对齐数据示意图

21. 4. 12 可独自设置各通道采样时间

ADC 会在多个 ADCCLK 周期内对输入电压进行采样，可使用 ADC_SMPTx 寄存器中的 CHTy 位修改周期数。每个通道均可以使用不同的采样时间进行采样。

总转换时间的计算公式如下：

Tconv = 采样时间 + 14 个周期

示例：

ADCCLK = 12MHz 且采样时间 = 4 个周期时：

Tconv = 4 + 14 = 18 个周期 = 1.5μs

21. 4. 13 外部触发转换和触发极性

可以通过配置 PIS 通道选择相应信号触发 ADC 转换，PIS 的配置请查看 PIS 相应章节描述。当 PIS 某一通道配置成 ADC 外部触发时，若相应的触发源事件发生，将自动触发标准序列或插入序列转换。

21. 4. 14 快速转换模式

可通过降低 ADC 分辨率来执行快速转换。ADC_CON0.RSEL 位用于选择数据寄存器中可用的位数。每种分辨率的最小转换时间如下：

- ◇ 12 位：采样时间+ 14 ADCCLK 周期
- ◇ 10 位：采样时间+ 12 ADCCLK 周期
- ◇ 8 位：采样时间+ 10 ADCCLK 周期
- ◇ 6 位：采样时间+ 8 ADCCLK 周期

21. 4. 15 数据管理

21. 4. 15. 1 使用DMA

标准组只有一个数据寄存器（ADC_NCHDR）用于存储 AD 转换结果值，所以，对于多个标准通道的转换，使用 DMA 可以快速存储数据，避免在上一次转换结果的值还未读出时，新的 ADC 结果值又写入 ADC_NCHDR 寄存器，造成数据丢失。

每完成标准通道组中的一个通道转换后，都会生成一个 DMA 请求。这样便可将转换的数据从 ADC_NCHDR 寄存器传输到软件指定的目标内存位置。

21. 4. 15. 2 在不使用DMA的情况下管理转换序列

如果转换过程足够慢，则可使用软件来处理转换序列。在这种情况下，必须将 ADC_CON1 寄存器中的 NCHESEL 位置 1，才能使 NCHE 状态位在每次转换结束时置 1，而不仅是在序列结束时置 1。当 NCHESEL = 1 时，会自动使能溢出检测。因此，每当转换结束时，NCHE 都会置 1，并且可以读取 ADC_NCHDR 寄存器。如果数据丢失（溢出），则会将 ADC_STAT 寄存器中的 OVR 位置 1 并生成一个中断（如果 ADC_CON0.OVRIE 位已置 1）。

要在 NCHESEL 位置 1 时将 ADC 从 OVR 状态中恢复，请按以下步骤操作：

1. 将 ADC_STAT 寄存器中的 OVR 位清零
2. 触发 ADC 开始转换

21. 4. 15. 3 在不使用DMA和溢出检测情况下进行转换

当 ADC 存在转换一个或多个通道时不需要每次读取数据的情况时，例如使用模拟看门狗，必须禁止 DMA (ADC_CON1.DMA = 0)，并且仅在序列结束 (NCHESEL = 0) 时才将 NCHE 位置 1。这样溢出检测被禁止。

21. 4. 16 ADC中断

当模拟看门狗状态位和溢出状态位分别置 1 时，标准组和插入组在转换结束时可能会产生中断。可以使用单独的中断使能位以实现灵活性。

中断事件	事件标志位	使能控制位
结束标准组的转换	NCHE	NCHEIE
结束插入组的转换	ICHE	ICHEIE
发生模拟看门狗事件	AWDF	AWDIE
溢出	OVR	OVRIE

表 22-2 ADC 中断

21. 5 特殊功能寄存器

21. 5. 1 寄存器列表

ADC 寄存器列表			
名称	偏移地址	类型	描述
ADC_STAT	000_H	R	ADC 状态寄存器
ADC_CLR	004_H	W	ADC 清零寄存器
ADC_CON0	008_H	R/W	ADC 控制寄存器 0
ADC_CON1	$00C_H$	R/W	ADC 控制寄存器 1
ADC_SMPT1	010_H	R/W	ADC 采样时间寄存器 1
ADC_SMPT2	014_H	R/W	ADC 采样时间寄存器 2
ADC_SMPT3	018_H	R/W	ADC 采样时间寄存器 3
ADC_NCHOFF	$01C_H$	R/W	ADC 标准通道数据偏移寄存器
ADC_ICHOFF1	020_H	R/W	ADC 插入通道数据偏移寄存器 1
ADC_ICHOFF2	024_H	R/W	ADC 插入通道数据偏移寄存器 2
ADC_ICHOFF3	028_H	R/W	ADC 插入通道数据偏移寄存器 3
ADC_ICHOFF4	$02C_H$	R/W	ADC 插入通道数据偏移寄存器 4
ADC_NCHS1	030_H	R/W	ADC 标准通道序列寄存器 1
ADC_NCHS2	034_H	R/W	ADC 标准通道序列寄存器 2
ADC_NCHS3	038_H	R/W	ADC 标准通道序列寄存器 3
ADC_NCHS4	$03C_H$	R/W	ADC 标准通道序列寄存器 4
ADC_ICHS	040_H	R/W	ADC 插入通道序列寄存器
ADC_CHSL	044_H	R/W	ADC 通道序列长度寄存器
ADC_WDTH	048_H	R/W	ADC 看门狗高阈值寄存器
ADC_WDTL	$04C_H$	R/W	ADC 看门狗低阈值寄存器
ADC_ICHDR1	050_H	R	ADC 插入通道数据寄存器 1
ADC_ICHDR2	054_H	R	ADC 插入通道数据寄存器 2
ADC_ICHDR3	058_H	R	ADC 插入通道数据寄存器 3
ADC_ICHDR4	$05C_H$	R	ADC 插入通道数据寄存器 4
ADC_NCHDR	060_H	R	ADC 标准通道数据寄存器
ADC_CCR	064_H	R/W	ADC 通用控制寄存器

21. 5. 2 寄存器描述

21. 5. 2. 1 ADC状态寄存器（ADC_STAT）

ADC 状态寄存器（ADC_STAT）

偏移地址：00_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																						ICHS	NCHS	Reserved				OVR	ICHE	NCHE	AWDF

Reserved	Bit 31-10	—	**保留**
ICHS	Bit 9	R	**插入通道转换开始标志位** 0：未开始插入转换或标志位已被清除 1：已开始插入转换 注：该位由硬件置1，通过操作ADC_CLR清零
NCHS	Bit 8	R	**标准通道转换开始标志位** 0：未开始标准转换或标志位已被清除 1：已开始标准转换 注：该位由硬件置1，通过操作ADC_CLR清零
Reserved	Bit 7-4	—	**保留**
OVR	Bit 3	R	**转换溢出标志位** 0：未发生溢出或标志位已被清除 1：发生溢出 注 1：溢出检测仅在 DMA=1 或 NCHESEL=1 时使能 注 2：该位由硬件置 1，通过操作 ADC_CLR 清零
ICHE	Bit 2	R	**插入通道转换结束标志位** 0：所有插入转换未完成或标志位已被清除 1：所有插入转换已完成 注：该位由硬件置 1，通过操作 ADC_CLR 清零
NCHE	Bit 1	R	**标准通道转换结束标志位** NCHESEL = 0时 0：标准转换序列未完成或标志位已被清除 1：标准转换序列已完成 NCHESEL = 1 时 0：单次标准转换未完成或标志位已被清除 1：单次标准转换已完成 注：该位由硬件置1，通过操作ADC_CLR清零
AWDF	Bit 0	R	**模拟看门狗标志位** 0：未发生看门狗事件或标志位已被清除 1：已发生看门狗事件 注：该位由硬件置1，通过操作ADC_CLR清零

21. 5. 2. 2 ADC清零寄存器（ADC_CLR）

ADC 清零寄存器（ADC_CLR）

偏移地址：04_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																						ICHS	NCHS	Reserved				OVR	ICHE	NCHE	AWDF

Reserved	Bit 31-10	—	**保留**
ICHS	Bit 9	W1	**插入通道开始标志位清零** 0：无操作 1：插入转换开始标志位清零
NCHS	Bit 8	W1	**标准通道开始标志位清零** 0：无操作 1：标准转换开始标志位清零
Reserved	Bit 7-4	—	**保留**
OVR	Bit 3	W1	**转换溢出标志位清零** 0：无操作 1：转换溢出标志位清零
ICHE	Bit 2	W1	**插入转换结束标志位清零** 0：无操作 1：插入转换结束标志位清零
NCHE	Bit 1	W1	**标准转换结束标志位清零** 0：无操作 1：标准转换结束标志位清零
AWDF	Bit 0	W1	**模拟看门狗标志位清零** 0：无操作 1：模拟看门狗标志位清零

21. 5. 2. 3 ADC控制寄存器 0（ADC_CON0）

ADC 控制寄存器 0（ADC_CON0）

偏移地址：08_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved					OVRIE	RSEL		NCHWDEN	ICHWDTEN	Reserved			CNTW			ETRGN			ICHDCEN	NCHDCEN	IAUTO	AWDSGL	SCANEN	ICHEIE	AWDIE	NCHEIE	AWDCH				

Reserved	Bit 31-27	—	**保留**
OVRIE	Bit 26	R/W	**溢出中断使能位** 0：禁止 1：使能
RSEL	Bit 25-24	R/W	**ADC 转换精度选择位** 00：6 位 01：8 位 10：10 位 11：12 位
NCHWDEN	Bit 23	R/W	**标准通道看门狗使能位** 0：禁止 1：使能
ICHWDTEN	Bit 22	R/W	**插入通道看门狗使能位** 0：禁止 1：使能
Reserved	Bit 21-19	—	**保留**
CNTW	Bit 18-16	R/W	**通道切换等待时间** 000：0个ADC时钟 001：1个ADC时钟 ..... 111：7个ADC时钟
ETRGN	Bit 15-13	R/W	**外部触发不连续转换通道数** 000：1 个通道 001：2 个通道 …… 111：8 个通道
ICHDCEN	Bit 12	R/W	**插入通道不连续转换使能位** 0：禁止 1：使能
NCHDCEN	Bit 11	R/W	**标准通道不连续转换使能位** 0：禁止 1：使能
IAUTO	Bit 10	R/W	**插入组自动转换使能位**

			0：禁止 1：使能
AWDSGL	Bit 9	R/W	**扫描模式单一通道模拟看门狗使能位** 0：禁止 1：使能
SCANEN	Bit 8	R/W	**扫描模式使能位** 0：禁止 1：使能
ICHEIE	Bit 7	R/W	**插入通道转换完成中断使能位** 0：禁止 1：使能
AWDIE	Bit 6	R/W	**模拟看门狗中断使能位** 0：禁止 1：使能
NCHEIE	Bit 5	R/W	**标准通道转换完成中断使能位** 0：禁止 1：使能
AWDCH	Bit 4-0	R/W	**模拟看门狗通道选择位** 00000：ADC 输入通道 0 00001：ADC 输入通道 1 …… 10011：ADC 输入通道 19 其他：保留

21. 5. 2. 4 ADC控制寄存器 1（ADC_CON1）

ADC 控制寄存器 1（ADC_CON1）

偏移地址：$0C_H$

复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	NCHTRG	NETS		Reserved					ICHTRG	IETS		Reserved								ALIGN	NCHESEL	Reserved	DMA	Reserved						CM	ADCEN

Reserved	Bit 31	—	**保留**
NCHTRG	Bit 30	W1	**标准通道触发位** 0：无操作 1：触发开始标准通道转换
NETS	Bits 29-28	R/W	**标准转换外部触发极性选择位** 00：外部触发禁止 01：上升沿触发 10：下降沿触发 11：上升沿和下降沿触发
Reserved	Bit 27-23	—	**保留**
ICHTRG	Bit 22	W1	**插入通道触发位** 0：无操作 1：触发开始插入通道转换
IETS	Bits 21-20	R/W	**插入转换外部触发极性选择位** 00：外部触发禁止 01：上升沿触发 10：下降沿触发 11：上升沿和下降沿触发
Reserved	Bits 19-12	—	**保留**
ALIGN	Bit 11	R/W	**数据对齐方式位** 0：右对齐 1：左对齐
NCHESEL	Bit 10	R/W	**标准转换结束标志选择位** 0：每个标准转换序列结束时将 STAT.NCHE 位置 1 1：每个标准转换结束时将 STAT.NCHE 位置 1
Reserved	Bit 9	—	**保留**
DMA	Bit 8	R/W	**DMA 访问使能位** 0：禁止 1：使能
Reserved	Bit 7-2	—	**保留**
CM	Bit 1	R/W	**转换模式** 0：单次转换

			1：连续转换
ADCEN	Bit 0	R/W	**ADC 使能位** 0：禁止 1：使能

21. 5. 2. 5 ADC采样时间寄存器 1（ADC_SMPT1）

ADC 采样时间寄存器 1（ADC_SMPT1）

偏移地址：10_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CHT7				CHT6				CHT5				CHT4				CHT3				CHT2				CHT1				CHT0			

CHT<y>	Bit 31-0	R/W	**通道 y 采样时间选择位（y=0..7）** 0000：1 个周期 0001：2 个周期 0010：4 个周期 0011：15 个周期 其他：预留

21. 5. 2. 6 ADC采样时间寄存器 2（ADC_SMPT2）

ADC 采样时间寄存器 2（ADC_SMPT2）

偏移地址：14_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CHT15				CHT14				CHT13				CHT12				CHT11				CHT10				CHT9				CHT8			

CHT<y>	Bit 31-0	R/W	**通道 y 采样时间选择位（y=8..15）** 0000：1 个周期 0001：2 个周期 0010：4 个周期 0011：15 个周期 其他：预留

21. 5. 2. 7 ADC采样时间寄存器 3（ADC_SMPT3）

ADC 采样时间寄存器 3（ADC_SMPT3）

偏移地址：18_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																CHT19				CHT18				CHT17				CHT16			

Reserved	Bits 31-16	—	**保留**
CHT<y>	Bits 15-0	R/W	**通道 y 采样时间选择位（y=16..19）** 0000：1 个周期 0001：2 个周期 0010：4 个周期 0011：15 个周期 其他：预留

21. 5. 2. 8 ADC标准通道数据偏移寄存器（ADC_NCHOFF）

ADC 标准通道数据偏移寄存器（ADC_NCHOFF）																															
偏移地址：$1C_H$																															
复位值：$00000000\_00000000\_00000000\_00000000_B$																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																				NOFF											

Reserved	Bit 31-12	—	保留
NOFF	Bit 11-0	R/W	**标准通道数据偏移量** ADC_NCHDR 中的数据为原始转换数据加上偏移量

21. 5. 2. 9 ADC插入通道数据偏移寄存器 1（ADC_ICHOFF1）

ADC 插入通道数据偏移寄存器 1（ADC_ICHOFF1）

偏移地址：20_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																				IOFF1											

Reserved	Bit 31-12	—	保留
IOFF1	Bit 11-0	R/W	**插入通道 1 数据偏移量** ADC_ICHDR1 中的数据为原始转换数据加上偏移量

21. 5. 2. 10 ADC插入通道数据偏移寄存器 2（ADC_ICHOFF2）

ADC 插入通道数据偏移寄存器 2（ADC_ICHOFF2）

偏移地址：24_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																				IOFF2											

Reserved	Bit 31-12	—	保留
IOFF2	Bit 11-0	R/W	**插入通道 2 数据偏移量** ADC_ICHDR2 中的数据为原始转换数据加上偏移量

21. 5. 2. 11 ADC插入通道数据偏移寄存器 3（ADC_ICHOFF3）

ADC 插入通道数据偏移寄存器 3（ADC_ICHOFF3）

偏移地址：28_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																				IOFF3											

Reserved	Bit 31-12	—	保留
IOFF3	Bit 11-0	R/W	**插入通道 3 数据偏移量** ADC_ICHDR3 中的数据为原始转换数据加上偏移量

21. 5. 2. 12 ADC插入通道数据偏移寄存器 4（ADC_ICHOFF4）

ADC 插入通道数据偏移寄存器 4（ADC_ICHOFF4）

偏移地址：$2C_H$

复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																				IOFF4											

Reserved	Bit 31-12	—	保留
IOFF4	Bit 11-0	R/W	**插入通道 4 数据偏移量** ADC_ICHDR4 中的数据为原始转换数据加上偏移量

21. 5. 2. 13 ADC标准通道序列寄存器 1（ADC_NCHS1）

ADC 标准通道序列寄存器 1（ADC_NCHS1）

偏移地址：30_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			NS4					Reserved			NS3					Reserved			NS2					Reserved			NS1				

Reserved	Bit 31-29	—	**保留**
NS4	Bit 28-24	R/W	**标准序列第 4 次转换通道编号** 00000~10011：通道 0~19 其他：预留
Reserved	Bit 23-21	—	**保留**
NS3	Bit 20-16	R/W	**标准序列第 3 次转换通道编号** 00000~10011：通道 0~19 其他：预留
Reserved	Bit 15-13	—	**保留**
NS2	Bit 12-8	R/W	**标准序列第 2 次转换通道编号** 00000~10011：通道 0~19 其他：预留
Reserved	Bit 7-5	—	**保留**
NS1	Bit 4-0	R/W	**标准序列第 1 次转换通道编号** 00000~10011：通道 0~19 其他：预留

21. 5. 2. 14 ADC标准通道序列寄存器 2（ADC_NCHS2）

ADC 标准通道序列寄存器 2（ADC_NCHS2）

偏移地址：34_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			NS8					Reserved			NS7					Reserved			NS6					Reserved			NS5				

Reserved	Bit 31-29	—	**保留**
NS8	Bit 28-24	R/W	**标准序列第 8 次转换通道编号** 00000~10011：通道 0~19 其他：预留
Reserved	Bit 23-21	—	**保留**
NS7	Bit 20-16	R/W	**标准序列第 7 次转换通道编号** 00000~10011：通道 0~19 其他：预留
Reserved	Bit 15-13	—	**保留**
NS6	Bit 12-8	R/W	**标准序列第 6 次转换通道编号** 00000~10011：通道 0~19 其他：预留
Reserved	Bit 7-5	—	**保留**
NS5	Bit 4-0	R/W	**标准序列第 5 次转换通道编号** 00000~10011：通道 0~19 其他：预留

21. 5. 2. 15 ADC标准通道序列寄存器 3（ADC_NCHS3）

ADC 标准通道序列寄存器 3（ADC_NCHS3）

偏移地址：38_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			NS12					Reserved			NS11					Reserved			NS10					Reserved			NS9				

Reserved	Bit 31-29	—	**保留**
NS12	Bit 28-24	R/W	**标准序列第 12 次转换通道编号** 00000~10011：通道 0~19 其他：预留
Reserved	Bit 23-21	—	**保留**
NS11	Bit 20-16	R/W	**标准序列第 11 次转换通道编号** 00000~10011：通道 0~19 其他：预留
Reserved	Bit 15-13	—	**保留**
NS10	Bit 12-8	R/W	**标准序列第 10 次转换通道编号** 00000~10011：通道 0~19 其他：预留
Reserved	Bit 7-5	—	**保留**
NS9	Bit 4-0	R/W	**标准序列第 9 次转换通道编号** 00000~10011：通道 0~19 其他：预留

21. 5. 2. 16 ADC标准通道序列寄存器 4（ADC_NCHS4）

ADC 标准通道序列寄存器 4（ADC_NCHS4）

偏移地址：$3C_H$

复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			NS16					Reserved			NS15					Reserved			NS14					Reserved			NS3				

Reserved	Bit 31-29	—	**保留**
NS16	Bit 28-24	R/W	**标准序列第 16 次转换通道编号** 00000~10011：通道 0~19 其他：预留
Reserved	Bit 23-21	—	**保留**
NS15	Bit 20-16	R/W	**标准序列第 15 次转换通道编号** 00000~10011：通道 0~19 其他：预留
Reserved	Bit 15-13	—	**保留**
NS14	Bit 12-8	R/W	**标准序列第 14 次转换通道编号** 00000~10011：通道 0~19 其他：预留
Reserved	Bit 7-5	—	**保留**
NS13	Bit 4-0	R/W	**标准序列第 13 次转换通道编号** 00000~10011：通道 0~19 其他：预留

21. 5. 2. 17 ADC插入通道序列寄存器（ADC_ICHS）

ADC 插入通道序列寄存器（ADC_ICHS）

偏移地址：40_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			IS4					Reserved			IS3					Reserved			IS2					Reserved			IS1				

Reserved	Bit 31-29	—	**保留**
IS4	Bit 28-24	R/W	**插入序列第 4 次转换通道编号** 00000~10011：通道 0~19 其他：预留
Reserved	Bit 23-21	—	**保留**
IS3	Bit 20-16	R/W	**插入序列第 3 次转换通道编号** 00000~10011：通道 0~19 其他：预留
Reserved	Bit 15-13	—	**保留**
IS2	Bit 12-8	R/W	**插入序列第 2 次转换通道编号** 00000~10011：通道 0~19 其他：预留
Reserved	Bit 7-5	—	**保留**
IS1	Bit 4-0	R/W	**插入序列第 1 次转换通道编号** 00000~10011：通道 0~19 其他：预留

21. 5. 2. 18 ADC通道序列长度寄存器（ADC_CHSL）

ADC 通道序列长度寄存器（ADC_CHSL）

偏移地址：44_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-10	—	**保留**
ISL	Bit 9-8	R/W	**插入通道序列长度** 00：1 次转换 01：2 次转换 10：3 次转换 11：4 次转换
Reserved	Bit 7-4	—	**保留**
NSL	Bit 3-0	R/W	**标准通道列长度** 0000：1 次转换 0001：2 次转换 …… 1111：16 次转换

21. 5. 2. 19 ADC看门狗高阈值寄存器（ADC_WDTH）

ADC 看门狗高阈值寄存器（ADC_WDTH）

偏移地址：48_H

复位值：$00000000\_00000000\_00001111\_11111111_B$

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

Reserved	Bit 31-12	—	保留
HT	Bit 11-0	R/W	**模拟看门狗高阈值** 当原始数据大于高阈值时产生看门狗事件

21. 5. 2. 20 ADC看门狗低阈值寄存器（ADC_WDTL）

ADC 看门狗低阈值寄存器（ADC_WDTL）

偏移地址：$4C_H$

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-12	—	保留
LT	Bit 11-0	R/W	**模拟看门狗低阈值** 当原始数据小于低阈值时产生看门狗事件

21. 5. 2. 21 ADC插入通道数据寄存器 1（ADC_ICHDR1）

ADC 插入通道数据寄存器 1（ADC_ICHDR1）

偏移地址：50_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-16	—	保留
VAL	Bit 15-0	R	**插入通道 1 转换数据** 该数据为对齐之后的数据

21. 5. 2. 22 ADC插入通道数据寄存器 2（ADC_ICHDR2）

ADC 插入通道数据寄存器 2（ADC_ICHDR2）

偏移地址：54_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-16	—	保留
VAL	Bit 15-0	R	**插入通道 2 转换数据** 该数据为对齐之后的数据

21. 5. 2. 23 ADC插入通道数据寄存器 3（ADC_ICHDR3）

ADC 插入通道数据寄存器 3（ADC_ICHDR3）

偏移地址：58_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-16	—	保留
VAL	Bit 15-0	R	**插入通道 3 转换数据** 该数据为对齐之后的数据

21. 5. 2. 24 ADC插入通道数据寄存器 4（ADC_ICHDR4）

ADC 插入通道数据寄存器 4（ADC_ICHDR4）

偏移地址：$5C_H$

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-16	—	保留
VAL	Bit 15-0	R	**插入通道 4 转换数据** 该数据为对齐之后的数据

21. 5. 2. 25 ADC标准通道数据寄存器（ADC_NCHDR）

ADC 标准通道数据寄存器（ADC_NCHDR）

偏移地址：60_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-16	—	**保留**
VAL	Bit 15-0	R	**标准通道转换数据** 该数据为对齐之后的数据

21. 5. 2. 26 ADC通用控制寄存器（ADC_CCR）

ADC 通用控制寄存器（ADC_CCR）

偏移地址：64_H

复位值：$00000000\_00000000\_11000000\_00000000_B$

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

Reserved	Bit31-21	—	**保留**
TSEN	Bit 20	R/W	**温感使能位** 0：禁止 1：使能
Reserved	Bit 19	—	**保留**
VRNSEL	Bit 18	R/W	**负向参考电压使能位** 0：无效 1：使能 VSS 为负参考 （ADC 工作时，必须设 1）
VRPSEL	Bit 17	R/W	**正向参考电压选择位** 0：VDD 1：VREFP 管脚（PA15）
Reserved	Bit 16	—	**保留**
PWRMODSEL	Bit 15	R/W	**ADC 工作模式选择** 0：高速模式 1：低速模式（ADC 时钟需低于 1MHz） 注：当ADC时钟小于1MHz时可使能低速模式，降

			低ADC模块功耗
Reserved	Bit 14-12	—	**保留**
IREFEN	Bit 11	R/W	**偏置电流使能位** 0：禁止 1：使能（ADC 工作时，必须设 1）
Reserved	Bit10-9	—	**保留**
VREFEN	Bit 8	R/W	**内部基准电压 1V 使能位** 0：禁止 1：使能
Reserved	Bit7-3	—	**保留**
CKDIV	Bit 2-0	R/W	**ADC 时钟分频选择位（基于 PCLK）** 000：1 分频 001：2 分频 …… 111：128 分频

注：ADC 正常工作时，需使能 IREFEN=1，否则会导致 ADC 工作异常；在 STOP 模式下 ADC 不工作，需设置 IREFEN=0，否则会增大芯片功耗。

第22章 模拟比较器（ACMP0~1）

22. 1 概述

ACMP 模块可以将模拟电压信号进行比较，通过一个数字输出来指示哪一个模拟信号的电压更高，输入模拟信号可以从内部产生或者外部管脚输入，其工作消耗电流的大小或者响应时间可通过驱动电流的大小进行调节。

22. 2 特性

- 4 个可选的外部模拟信号输入到正端
- 2 个可选的外部模拟信号输入到负端
- 2 个可选的内部模拟信号输入到负端
- 可支持多种工作模式选择
- 可配置的迟滞选择，0~±60mv 之间可选 8 个等级
- 可配置的响应时间
- 异步中断源可选为以下边沿触发
  - 上升沿
  - 下降沿
  - 上升下降沿
- 支持在芯片低功耗模式运行
- 还支持以下特性
  - 可调节的内部电阻
  - 可配置比较器反向输出
  - 可配置不活动时的比较器输出
  - 比较器输出直接给 PIS

22. 3 结构框图

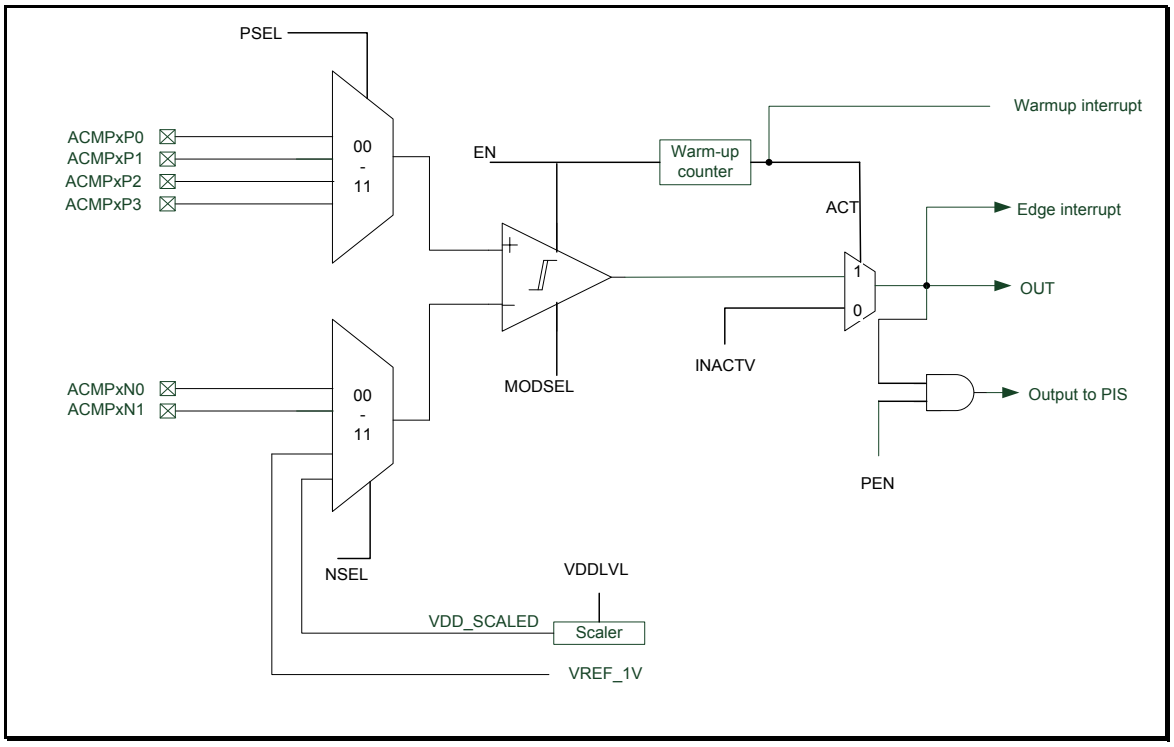


图 23-1 ACMP 结构框图

模拟比较器用来比较的是两个模拟输入量，一个正端，一个负端，通过比较器输出的数字信号来指示模拟输入量的电压高低情况，当输出为高时，表明正端输入电压高于负端输入电压，而当输出为低时，表明正端输入电压低于负端输入电压。

模拟比较器的输出数字信号保存在寄存器 OUT，该信号会随着两个模拟输入量的比较关系的变化而变化，除此之外其他对模拟比较器配置的改变要求在其不工作的情况下进行。

22. 4 功能描述

22. 4. 1 ACMP控制

22. 4. 1. 1 稳定时间

当寄存器 ACMP_CON.EN = 1 时，模拟比较器被使能，需要等待一段建立时间之后，才能正常工作，此时输出才有效，而这段建立时间称为稳定时间，该稳定时间可在寄存器 ACMP_CON.WARMUPT 位中配置在 1us ~ 50s 区间，以时钟计数值决定等待时间长短，当稳定时间结束后，寄存器 ACMP_STAT.ACT 位将置 1，表明模拟比较器进入有效状态开始正常工作。稳定时间期间的比较器处于无效状态，其输出信号配置在寄存器 ACMP_CON.INACTV 位。

若要在芯片工作模式 STOP 下使用模拟比较器，应该等到稳定时间之后（ACMP_STAT.ACT 为 1）才能配置进入对应模式，否则，将不能产生比较器中断；而要在芯片工作模式 SLEEP 下使用，稳定期间就可以配置进入。

22. 4. 1. 2 响应

在模拟比较器正常工作时，当模拟输入量的比较极性变化时，需要等待一段时间之后，数字输出量才会对应变化，该时间称为响应时间，可以通过比较器工作模式选择位 MODE 调节 BIAS 的电流大小来改变响应时间，小的 BIAS 电流会降低工作功耗，但会带来较大的响应时间。

比较器工作模式	偏置电流（uA）	响应时间（us）
超低功耗模式	1	7
低功耗模式	4	1.5
普通模式	10	0.7
高速模式	40	0.3

表 23-1 响应时间与工作模式对应关系

当选择了高速模式时，应该选择适当的迟滞档位，以消除比较器输出信号的毛刺。

22. 4. 1. 3 迟滞

通过配置寄存器 ACMP_CON.HYSTSEL 位，模拟比较器有 8 个可选的迟滞档位，包括档位 0 对应的迟滞是 0，当选为非 0 档位时，当正端和负端输入模拟信号电压的差值达到 ACMP_CON.HYSTSEL 选择的迟滞电压时，比较器输出信号才会对应变化，迟滞的作用是滤除那些在比较临界点的输出模拟信号的电压抖动变化，而只输出模拟信号压差大于迟滞范围的有效信号，所以迟滞的选取应该考虑输入量的抖动与范围。另外，迟滞的选取还会影响模拟比较器的工作功耗情况，一般是高的迟滞档位会降低工作的功耗。

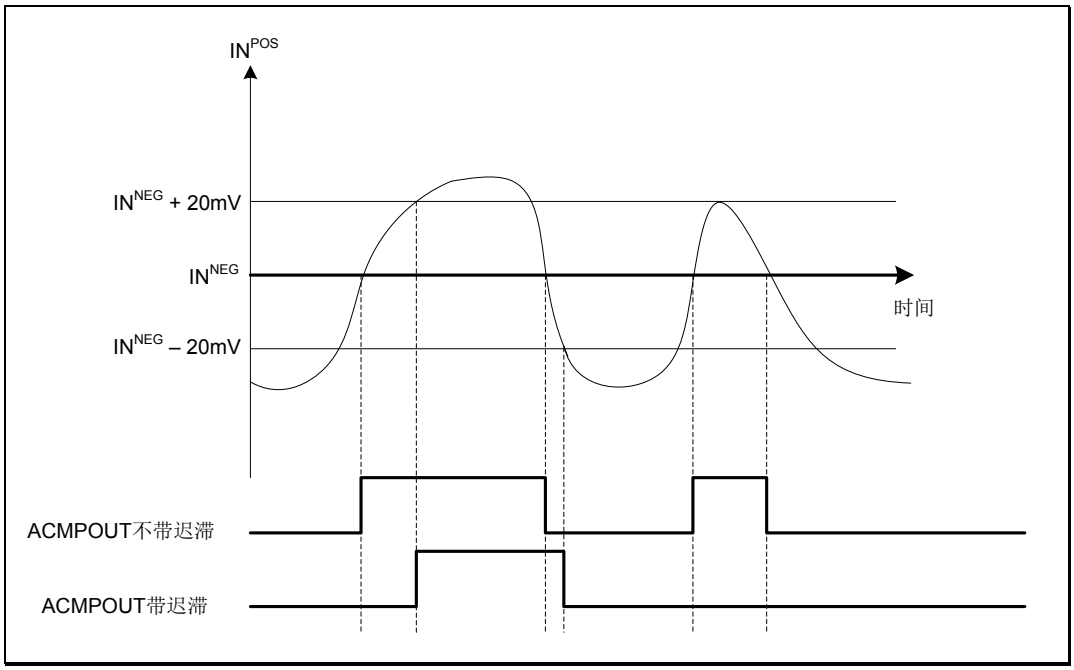


图 23-2 迟滞原理图

22. 4. 2 通道选择

模拟比较器有两个模拟输入量，正端输入可从 4 个外部输入中选择其一，负端输入可由 2 个外部输入、以及内部参考源 VREF_1V 内部输入。

22. 4. 3 数据管理

模拟比较器的输出数字信号保存在寄存器 ACMP_STAT.OUT，同时，当设置 ACMP_PORT.PEN = 1 时，可通过 PIS 输出到 IO，可以通过配置 ACMP_CON.OUTINV 来选择输出信号的极性。

22. 4. 4 中断与PIS触发

模拟比较器中断可选为边沿中断，包括上升沿或者下降沿，或双沿中断同时打开，寄存器 ACMP_RIF.EDGE 用于记录中断标志。当对应的 ACMP_IES.EDGE 中断使能被打开时，若此时产生边沿中断，则会产生中断请求，同时 ACMP_IFM.EDGE 会置 1，表明产生了中断请求，在系统响应了请求后，在对应 ACMP_IFC.EDGE 上置 1 会清掉 ACMP_IFM.EDGE 位。边沿中断可用于将系统从低功耗模式下唤醒，进入到正常工作模式。

当模拟比较器在开始的稳定工作后，也会产生一个稳定工作中断，并会被记录在 ACMP_RIF.WARMUP 中，当该中断在 ACMP_IES.WARMUP 被使能时，同样会产生中断请求。

模拟比较器的输出信号可连接到外设互联（PIS）系统中，作为其他外设的触发源。

22. 5 特殊功能寄存器

22. 5. 1 寄存器列表

ACMP 寄存器列表			
名称	偏移地址	类型	描述
ACMP_CON	000_H	R/W	ACMP 控制寄存器
ACMP_INPUTSEL	004_H	R/W	ACMP 输入选择寄存器
ACMP_STAT	008_H	R	ACMP 状态寄存器
ACMP_IES	$00C_H$	W	ACMP 中断使能设置寄存器
ACMP_IEC	010_H	W	ACMP 中断使能清除寄存器
ACMP_IEV	014_H	R	ACMP 中断使能有效寄存器
ACMP_RIF	018_H	R	ACMP 原始中断标志寄存器
ACMP_IFM	$01C_H$	R/W	ACMP 中断标志屏蔽寄存器
ACMP_IFC	020_H	W	ACMP 中断标志清除寄存器
ACMP_PORT	024_H	R/W	ACMP 端口寄存器

22. 5. 2 寄存器描述

22. 5. 2. 1 ACMP控制寄存器（ACMP _CON）

ACMP 控制寄存器（ACMP_CON）

偏移地址：00_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved									FLTSEL			Reserved		FALLEN	RISEEN	MODSEL		Reserved			WARMUPT			Reserved		HYSTSEL			OUTINV	INACTV	Reserved	EN

Reserved	Bit 31-23	—	**保留**
FLTSEL<2:0>	Bit 22-20	R/W	**数字滤波时间选择位** 000：禁止滤波 001：1 个 PCLK 时钟 010：2 个 PCLK 时钟 011：4 个 PCLK 时钟 100：8 个 PCLK 时钟 其他：保留
Reserved	Bit 19-18	—	**保留**
FALLEN	Bit 17	R/W	该位写 1，当比较输出出现下降沿时，可置起边沿中断标志。
RISEEN	Bit 16	R/W	该位写 1，当比较输出出现上升沿时，可置起边沿中断标志。
MODSEL	Bit 15-14	R/W	**模拟比较器运行模式选择** 00：超低功耗模式 01：低功耗模式 10：普通模式 11：高速模式
Reserved	Bit 13-11	—	**保留**
WARMUPT	Bit 10-8	R/W	**模拟比较器 Warm-up 时间设置** 000：4 PCLK周期 001：8 PCLK周期 010：16 PCLK周期: 011：32 PCLK周期 100：64 PCLK周期 101：128 PCLK周期 110：256 PCLK周期 111：512 PCLK 周期
Reserved	Bit 7	—	**保留**
HYSTSEL	Bit 6-4	R/W	**迟滞电平选择** 此处描述的迟滞电平为设计理论值，具体请参考

			数据手册中的模拟比较器电气特性参数 000：无迟滞 001：~15 mV迟滞 010：~22 mV迟滞 011：~29 mV迟滞 100：~36 mV迟滞 101：~43 mV迟滞 110：~50 mV迟滞 111：~57 mV 迟滞
OUTINV	Bit 3	R/W	**模拟比较器复用输出取反** 0：比较输出不取反 1：比较输出取反
INACTV	Bit 2	R/W	**比较器的无效状态值** 0：无效值为0 1：无效值为 1
Reserved	Bit 1	—	**保留**
EN	Bit 0	R/W	**模拟比较器使能** 0：禁止 1：使能

22. 5. 2. 2 ACMP输入选择寄存器（ACMP_INPUTSEL）

ACMP 输入选择寄存器（ACMP_INPUTSEL）

偏移地址：04_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-8	—	**保留**
NSEL	Bit 7-4	W	**反向输入选择** 0000：VREF_1V 0001：保留 0010：通道 0 0011：通道 1 其他：保留
Reserved	Bit 3	—	**保留**
PSEL	Bit 2-0	R	**正向输入选择** 000：通道 0 001：通道 1 010：通道 2 011：通道 3 其他：保留

22. 5. 2. 3 ACMP状态寄存器（ACMP_STAT）

ACMP 状态寄存器（ACMP_STAT）

偏移地址：08_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-2	—	保留
OUT	Bit 1	R	模拟比较器输出值
ACT	Bit 0	R	模拟比较器有效状态

22. 5. 2. 4 ACMP中断使能设置寄存器（ACMP_IES）

ACMP 中断使能设置寄存器（ACMP_IES）

偏移地址：$0C_H$

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-2	—	保留
WARMUP	Bit 1	W1	**Warm-up中断使能** 0：禁止 1：使能
EDGE	Bit 0	W1	**边沿触发中断使能** 0：禁止 1：使能

22. 5. 2. 5 ACMP中断使能清除寄存器（ACMP_IEC）

ACMP 中断使能清除寄存器（ACMP_IEC）

偏移地址：10_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-2	—	保留
WARMUP	Bit 1	W1	**Warm-up中断使能清除** 0：无效 1：写1清除
EDGE	Bit 0	W1	**边沿触发中断使能清除** 0：无效 1：写1清除

22. 5. 2. 6 ACMP中断使能有效寄存器（ACMP_IEV）

ACMP 中断使能有效寄存器（ACMP_IEV）

偏移地址：14_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-2	—	保留
WARMUP	Bit 1	R	**Warm-up中断有效状态** 0：禁止 1：使能
EDGE	Bit 0	R	**边沿触发中断有效状态** 0：禁止 1：使能

22. 5. 2. 7 ACMP原始中断标志寄存器（ACMP_RIF）

ACMP 原始中断标志寄存器（ACMP_RIF）

偏移地址：18_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-2	—	保留
WARMUP	Bit 1	R	**原始Warm-up中断标志** 0：未发生 1：已发生
EDGE	Bit 0	R	**原始边沿触发中断标志** 0：未发生 1：已发生

22. 5. 2. 8 ACMP中断标志屏蔽寄存器（ACMP_IFM）

ACMP 中断标志屏蔽寄存器（ACMP_IFM）

偏移地址：$1C_H$

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-2	—	保留
WARMUP	Bit 1	R	**屏蔽Warm-up中断标志** 0：未发生 1：已发生
EDGE	Bit 0	R	**屏蔽边沿触发中断标志** 0：未发生 1：已发生

22. 5. 2. 9 ACMP中断标志清除寄存器（ACMP_IFC）

ACMP 中断标志清除寄存器（ACMP_IFC）

偏移地址：20_H

复位值：$00000000\_00000000\_00001111\_11111111_B$

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

Reserved	Bits 31-2	—	保留
WARMUP	Bit 1	W1	**Warm-up中断标志清除** 0：无效 1：写1清除
EDGE	Bit 0	W1	**边沿触发中断标志清除** 0：无效 1：写1清除

22. 5. 2. 10 ACMP端口寄存器（ACMP_PORT）

ACMP 端口寄存器（ACMP_PORT）

偏移地址：24_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

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

Reserved	Bit 31-1	—	保留
PEN	Bit 0	R/W	**模拟比较器输出引脚使能** 0：禁止 1：使能

第23章 调试控制（DBGC）

23. 1 概述

ES32M0150 系列 MCU 使用的内核是 Cortex™-M0，该内核包含用于高级调试功能的硬件。利用这些调试功能，可以在取指（指令断点）或取访问数据（数据断点）时停止内核。内核在停止时，可以查询内核的内部状态和系统的外部状态。查询完成后，可以恢复内核和系统，并恢复程序执行。

当调试器与 MCU 相连并进行调试时，将使用内核的硬件调试模块。

ES32M0150 系列 MCU 提供 SWD（Serial Wire Debug）调试接口。

参考文档：

ARM_debug_interface_v5.pdf（ADIv5 Architecture specification）
ARM_debug_interface_v5_supplement.pdf（ADIv5.1 Spec Supplement）

DDI0419C_arm_architecture_v6m_reference_manual.pdf（ARMv6 Architecture）

Cortex-M0_TechnicalReferenceManualRevB_DDI0432_r0p0-01rel0.pdf

23. 2 特性

- 支持 SW-DP：调试端口电路，实现 DAP 电路和外部调试主机的通讯
- 支持 MEM-AP：访问端口电路，实现 DAP 电路与被调试单元的通讯
- 支持断点（Breakpoint）：4 个断点
- 支持数据观测和追踪（DWT）：2 个数据观测点

23. 3 结构框图

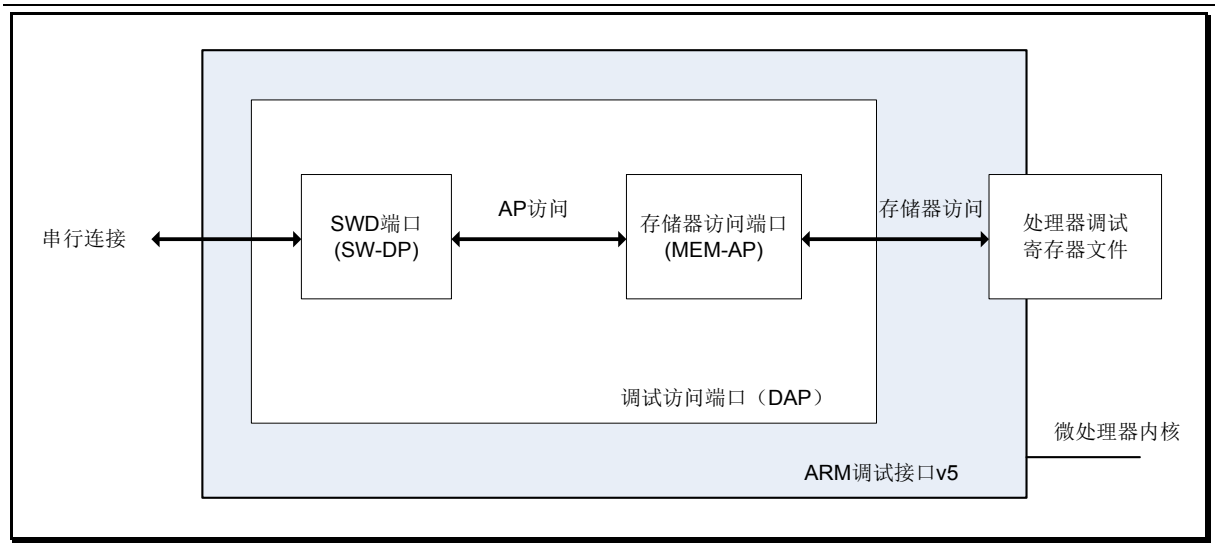


图 24-1　SWD 调试结构图

23. 4 功能描述

23. 4. 1 调试端口

下表为 SWD 调试用端口，芯片上电后默认作为调试功能使用。

端口功能	输入/输出	说明
SWCLK	输入	调试时钟。该端口在调试模式下为调试电路提供通信时钟。
SWDIO	输入/输出	调试数据输入/输出端口。用于 SW-DP 与外部调试主机的数据交互。

表 24-1 SWD 端口描述

23. 4. 2 调试冻结

程序开发过程中，会经历反复调试，通过调试工具对运行程序进行暂停，实际上在暂停内核的时候，仍有部分硬件外设在工作（如：定时器、看门狗等），影响调试效果；DBGC 模块可以实现内核和外设同时暂停，实现高级调试功能。

操作示例：

1. 按照需求正确初始化使用的外设，如定时器 GP16C2T0
2. 为保证程序在调试过程中暂停时 GP16C2T0 计数同时暂停，需设置 DBG_APB1FZ. GP16C2T0_STOP = 1
3. 如果未设置 DBG_APB1FZ. GP16C2T0_STOP = 1，程序在调试过程中暂停时，实际 GP16C2T0 仍然在计数。

23. 4. 3 调试复位

内核调试电路和调试控制寄存器只可被上电复位、欠压复位及软件复位中的芯片全局复位（RMU_AHB2RSTR.CHIPRST）所复位。

23. 4. 4 MEM-AP访问端口

MEM-AP 端口，用于访问被调试单元的存储器映射区域。

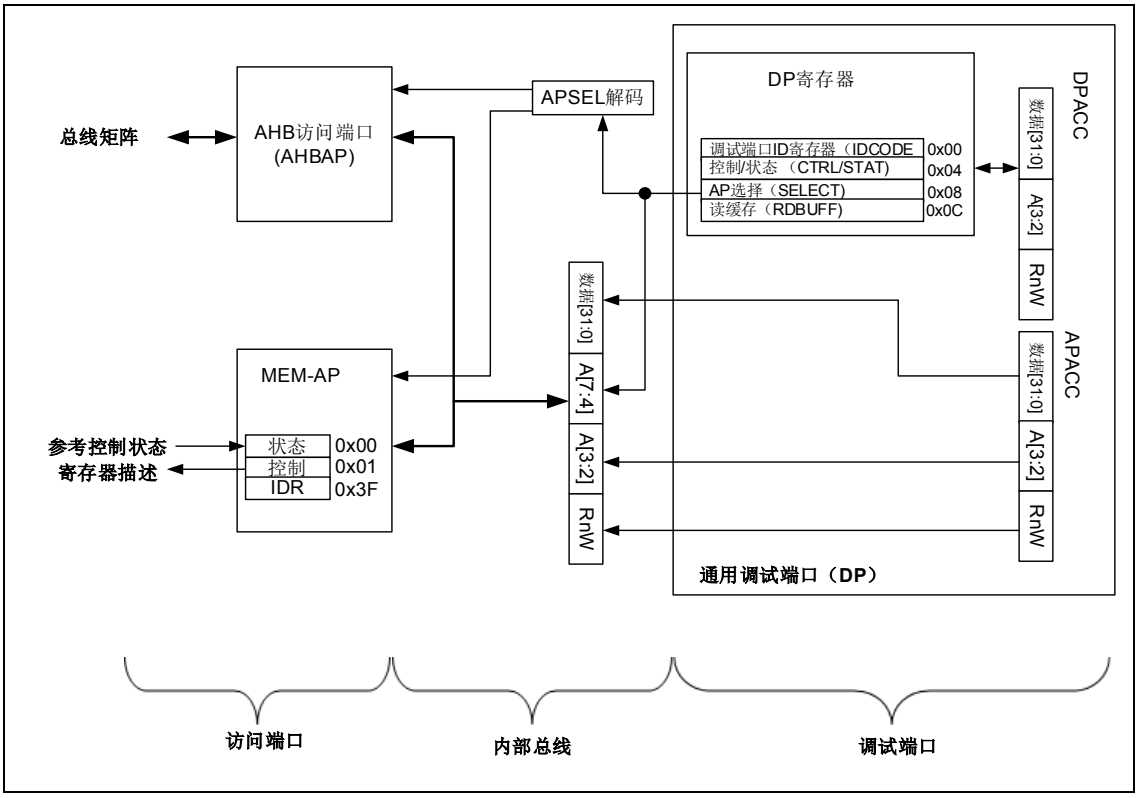


图 24-2 MEM-AP 地址映射

23. 5 特殊功能寄存器

23. 5. 1 寄存器列表

DBGC 寄存器列表		
名称	偏移地址	描述
DBG_IDCODE	000_H	DBG 器件识别码
Reserved	004_H	保留
DBG_APB1FZ	008_H	APB1 外设调试冻结寄存器
DBG_APB2FZ	$00C_H$	APB2 外设调试冻结寄存器

23. 5. 2 寄存器描述

23. 5. 2. 1 DBG器件识别码（DBG_IDCODE）

DBG 器件识别码寄存器（DBG_IDCODE）

偏移地址：000_H

上电复位值：xxxxxxxx_xxxxxxxx_xxxxxxxx_xxxxxxxx $_B$

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

REV_ID	Bit 31-16	R	**版本识别码** 0x1000：版本 A 0x1001：版本 B
CORE_ID	Bit 15-12	R	**内核识别码** 0x0：Cortex-M0
DEV_ID	Bit 11-0	R	**器件识别码** 0x032：MCU 识别码

23. 5. 2. 2 APB1 外设调试冻结寄存器（DBG_APB1FZ）

APB1 外设调试冻结寄存器（DBG_APB1FZ）

偏移地址：008_H

上电复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																											GP16C2T2_STOP	GP16C2T1_STOP	GP16C2T0_STOP	BS16T0_STOP	AD16C4T0_STOP

Reserved	Bit 31-5	—	保留
GP16C2T2_STOP	Bit 4	R/W	**GP16C2T2 调试暂停选择位** 0：内核停止时，仍正常计数 1：内核停止时，暂停计数
GP16C2T1_STOP	Bit 3	R/W	**GP16C2T1 调试暂停选择位** 0：内核停止时，仍正常计数 1：内核停止时，暂停计数
GP16C2T0_STOP	Bit 2	R/W	**GP16C2T0 调试暂停选择位** 0：内核停止时，仍正常计数 1：内核停止时，暂停计数
BS16T0_STOP	Bit 1	R/W	**BS16T0 调试暂停选择位** 0：内核停止时，仍正常计数 1：内核停止时，暂停计数
AD16C4T0_STOP	Bit 0	R/W	**AD16C4T0 调试暂停选择位** 0：内核停止时，仍正常计数 1：内核停止时，暂停计数

注：该寄存器仅支持按字写入。

23. 5. 2. 3 APB2 外设调试冻结寄存器（DBG_APB2FZ）

APB2 外设调试冻结寄存器（DBG_APB2FZ）

偏移地址：$00C_H$

上电复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																						WWDT_STOP	IDWT_STOP	Reserved						I2C1_STOP	I2C0_STOP

Reserved	Bit 31-10	—	保留

WWDT_STOP	Bit 9	R/W	**WWDT 调试暂停选择位** 0：内核停止时，仍正常计数 1：内核停止时，暂停计数
IWDT_STOP	Bit 8	R/W	**IWDT 调试暂停选择位** 0：内核停止时，仍正常计数 1：内核停止时，暂停计数
Reserved	Bit 7-2	—	**保留**
I2C1_STOP	Bit 1	R/W	**I2C1 调试暂停选择位** 0：内核停止时，仍正常计数 1：内核停止时，暂停计数
I2C0_STOP	Bit 0	R/W	**I2C0 调试暂停选择位** 0：内核停止时，仍正常计数 1：内核停止时，暂停计数

注：该寄存器仅支持按字写入。

第24章 FLASH信息区

24. 1 概述

FLASH 信息区用来存储芯片的只读信息和配置信息。

用户在程序中只能对只读信息和配置信息进行读操作。可以使用芯片烧录工具对配置信息进行修改，但无法改变只读信息。

24. 2 特性

- FLASH 信息区存储的只读信息包括：
  - 产品识别码
  - 芯片唯一码
- FLASH 信息区存储的配置信息包括：
  - 芯片配置字
  - 写保护区域配置字
  - 数据区配置字
  - 全局读保护配置字
  - 用户程序校验码

24. 3 功能描述

24. 3. 1 FLASH信息区只读信息

FLASH 信息区的基地址为 0x0001_0000，可以字读取。

24. 3. 1. 1 芯片唯一码UID

芯片唯一码 UID 为 96 位，每一颗芯片都是唯一的编码，可以用做：

◇ 终端产品序列号

◇ 通过特定的加密算法生成安全密钥

寄存器名称	芯片唯一码 0（UID0）	
地址偏移	$09E0_H$	
UID0	Bit 31-0	芯片唯一码 0

寄存器名称	芯片唯一码 1（UID1）	
地址偏移	$09E8_H$	
UID1	Bit 31-0	芯片唯一码 1

寄存器名称	芯片唯一码 2（UID2）	
地址偏移	$09F0_H$	
UID2	Bit 31-0	芯片唯一码 2

24. 3. 1. 2 芯片产品识别码CHIPID

CHIPID 用来区分芯片产品型号，为 32 位编码。

寄存器名称	芯片产品识别码（CHIPID）	
地址偏移	$0BF8_H$	
CHIPID	Bit 31-0	CHIPID

24. 3. 2 FLASH信息区配置信息

FLASH 信息区的配置信息在芯片程序运行前生效。

FLASH 信息区可以字读取。只能使用芯片烧录工具对配置信息进行修改。

芯片配置字默认值仅表示为烧录器界面的缺省设置，配置字地址单元在出厂时可能为非空，在 FLASH 编程之前必须先擦除，才能正确写入所设置的配置字。

24. 3. 2. 1 芯片配置字CFG_WORD

芯片的部分特性需要通过芯片配置字配置实现，这些配置在程序运行前生效。

寄存器名称	芯片配置字 0（CFG_WORD0）	
地址偏移	0000_H	
低 16Bits 复位值	$1001\_1001\_1000\_0000_B$（9980_H）	
—	Bits 63-32	**保留未用**
—	Bits 31-16	**Bits 15-0 取反值（不满足取反时 Bit15-0 强制为默认值）**
—	Bit 15	保留（必须设置为 1）
IWDTEN	Bit 14	**IWDT 使能位** 0：由软件使能 1：硬件强制使能 注：硬件强制使能后，软件无法关闭；中断强制使能，软件无法关闭；复位强制使能，软件无法关闭；时钟源固定为 LRC，软件无法切换。
WWDTEN	Bit 13	**WWDT 使能位** 0：软件使能后可关闭 1：软件使能后无法关闭
—	Bit 12	保留（可设置为 0）
BORVS	Bits 11-10	**BOR 电压点选择位** 00：3.1V 01：2.5V 10：2.1V（默认） 11：保留
—	Bit 9	保留（必须设置为 0）
XTAL	Bit 8	**外部振荡器模式选择位** 0：1~8MHz 1：8~24MHz
CFG_MRST	Bit 7	**MRST 管脚复用选择位** 0：GPIO 功能 1：MRST 功能(默认) 注：若 MRST 管脚配置为 GPIO 功能，芯片低功耗模式不能再使用
BOOT	Bit 6	**FLASH 启动选择位** 0：启动地址 0x0 (默认) 1：启动地址 0xF000
—	Bits 5-0	保留（可设置为 0）

注 1：复位值是指芯片配置字被从信息区读出之前的电路缺省值。

24. 3. 2. 2 写保护区域配置字CFG_WRP

芯片支持 2 个保护区域，分别通过 CFG_WRP0 和 CFG_WRP1 来配置。设置为写保护的区域，用户程序将不能通过 IAP 对其进行擦写。

寄存器名称	写保护区域 x 配置字（CFG_WRPx）（x=0、1）	
地址偏移	0020_H，0028_H	
低 16 位复位值	$0000\_0000\_0000\_0001_B$（0001_H）	
—	Bits 63-32	**保留未用**
—	Bits 31-16	**Bits 15-0 取反值（不满足取反时 Bit15-0 强制为默认值）**
END	Bits 15-11	**保护结束页配置位** 0x0：Flash Page 3（默认） 0x1：Flash Page 7 0x2：Flash Page 11 …… 0x1F：Flash Page 127 注：保护结束页数必须配置为大于或等于起始页数，否则保护配置失效
—	Bits 10-8	**保留未用（可设置为 0）**
START	Bits 7-3	**保护起始页配置位** 0x0：Flash Page 0（默认） 0x1：Flash Page 4 0x2：Flash Page 8 …… 0x1F：Flash Page 124
—	Bit 2-1	**保留未用（可设置为 0）**
ENB	Bit 0	**保护使能位** 0：使能 1：禁止（默认）

24. 3. 2. 3 数据区配置字CFG_DAFLS

芯片支持 1 个数据区域，通过 CFG_DAFLS 来配置。通过其配置可以将 FLASH 空间分为程序区和数据区。程序区和数据区的 IAP 擦写命令不同。

寄存器名称	数据 Flash 配置字（CFG_DAFLS）	
地址偏移	0030_H	
低 16 位复位值	$0000\_0000\_0000\_0001_B$（0001_H）	
—	Bits 63-32	**保留未用**
—	Bits 31-16	**Bits 15-0 取反值（不满足取反时 Bit15-0 强制为默认值）**
END	Bits 15-11	**数据 Flash 结束页配置位** 0x0：Flash Page 3（默认） 0x1：Flash Page 7 0x2：Flash Page 11 …… 0x1F：Flash Page 127 注：数据 Flash 结束页数必须配置为大于或等于起始页数，否则数据 Flash 配置失效
—	Bits 10-8	**保留未用（可设置为 0）**
START	Bits 7-3	**数据 Flash 起始页配置位** 0x0：Flash Page 0（默认） 0x1：Flash Page 4 0x2：Flash Page 8 …… 0x1F：Flash Page 124
—	Bit 2-1	**保留未用（可设置为 0）**
ENB	Bit 0	**数据 Flash 使能位** 0：使能 1：禁止（默认）

24. 3. 2. 4 用户程序校验码CHKSUM

烧录工具将用户程序校验码写入此区域。

寄存器名称	用户程序校验码（CHKSUM）	
地址偏移	$01C0_H$	
—	Bits 63-32	**保留未用**
CHKSUM	Bits 31-0	**用户程序校验码**

寄存器名称	用户程序校验码反码（CHKSUMN）	
地址偏移	$01C8_H$	
—	Bits 63-32	**保留未用**
CHKSUMN	Bits 31-0	**用户程序校验码反码**

24. 3. 2. 5 全局读保护配置字CFG_GBRDP

全局读保护分为 Level0~2 三个等级，Level0 为不保护，Level1 和 Level2 的详细说明请参考“存储器系统控制（MSC）”中“FLASH 保护”章节的描述。

寄存器名称	全局读保护配置字（CFG_GBRDP）	
地址偏移	0040_H	
复位值	$0101\_0101\_1010\_1010\_0101\_0101\_1010\_1010_B$（$55AA\_55AA_H$）	
—	Bits 63-32	**保留未用**
GBRDP	Bits 31-0	**全部读保护配置位** 0xFFFF_FFFF：读保护等级 Level 0 0xFFFF_XXXX：读保护等级 Level 1（XXXX 不为 FFFF） 0xYYYY_XXXX：读保护等级为 Level 2（YYYY 不为 FFFF）（默认）

24. 3. 2. 6 私有读保护配置字CFG_PCROP

寄存器名称	私有代码读出保护区域 x 配置字（CFG_PCROPx）（x=0、1）	
地址偏移	0200_H，0208_H	
低 16 位复位值	$1111\_1000\_0000\_0000_B$（$F800_H$），Flash 容量为 64KB	
—	Bits 63-32	**保留未用**
—	Bits 31-16	**Bits 15-0 取反值（不满足取反时 Bit15-0 强制为默认值）**
END	Bits 15-11	**保护结束页配置位** 0x0：Flash Page 3 0x1：Flash Page 7 0x2：Flash Page 11 …… 0x1F：Flash Page 127（默认） 注：保护结束页数必须配置为大于或等于起始页数，否则整个 Flash 程序区域都将处于**非保护状态**
—	Bits 10-8	**保留未用（可设置为 0）**
START	Bits 7-3	**保护起始页配置位** 0x0：Flash Page 0 0x1：Flash Page 4 0x2：Flash Page 8 …… 0x1F：Flash Page 124（默认）
—	Bit 2-1	**保留未用（可设置为 0）**
ENB	Bit 0	**保护使能位** 0：使能（默认） 1：禁止

附录1 ARM Cortex-M0 参考资料

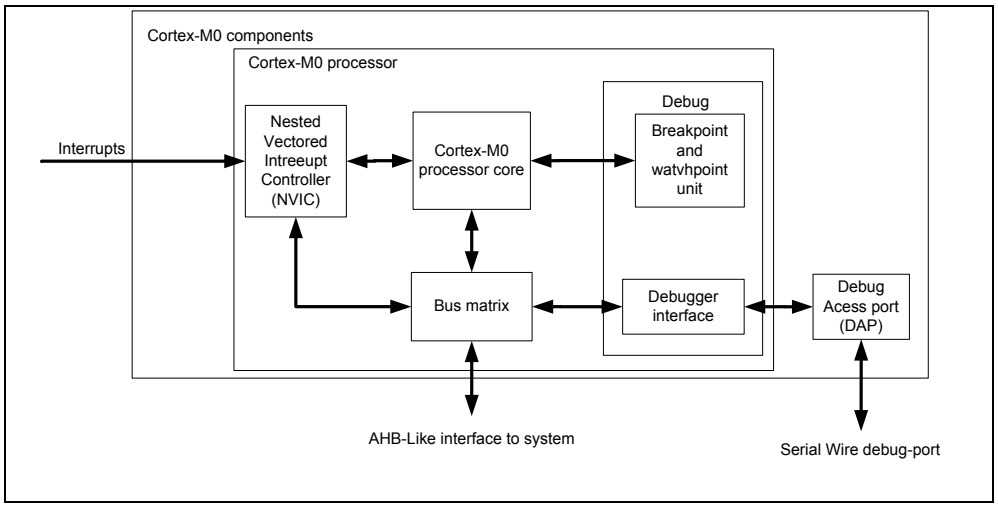
附录1.1 介绍

下面的参考资料以 ARM Cortex-M0 用户指南（ARM Cortex-M0 User Guide）为蓝本。

附录1.2 关于Cortex-M0 处理器和核心外设

Cortex-M0 处理器是一个入门级的 32 位 ARM Cortex 处理器，可用于广泛的嵌入式应用中。该处理器包含以下特性，给开发者提供了极大的便利：

- ◇ 结构简单，容易学习和编程
- ◇ 功耗极低，运算效率高
- ◇ 出色的代码密度
- ◇ 确定、高性能的中断处理
- ◇ 向上与 Cortex-M 系列处理器兼容



附录图 1-1 Cortex-M0 的具体实现

Cortex-M0 处理器基于一个高集成度、低功耗的 32 位处理器内核，采用 3 级流水线冯·诺伊曼结构。通过简单、功能强大的指令集以及全面优化的设计（提供包括一个单周期乘法器在内的高端处理硬件），Cortex-M0 处理器可实现极高的能效。

Cortex-M0 处理器采用 ARMv6-M 结构，基于 16 位 Thumb 指令集，并包含 Thumb-2 技术。因而能提供一个现代 32 位体系结构处理器所希望的出色性能，代码密度比其他 8 位和 16 位微控制器都要高。

Cortex-M0 紧密集成了一个可配置的内嵌向量中断控制器（NVIC），提供业界领先的中断性能。NVIC 具有以下功能：

- ◇ 包含一个不可屏蔽的中断（NMI）。
- ◇ 提供零抖动中断选项
- ◇ 提供四个中断优先级

处理器内核和 NVIC 的紧密结合使得中断服务程序（ISR）可以快速执行，极大地缩短了中断

延迟。这是通过硬件寄存器堆栈、放弃与重启多加载及多存储的能力来获得的。中断程序不需要任何汇编封装代码，不用消耗任何 ISR 代码。尾链优化还极大地降低了从一个 ISR 切换到另一个 ISR 时的开销。

为了优化低功耗设计，NVIC 还与睡眠模式相结合，提供一个深度睡眠功能，使整个设备迅速降低功耗。

附录1. 2. 1 系统级接口

Cortex-M0 处理器提供一个简单的系统级接口，使用 AMBA 技术来提供高速、低延迟的存储器访问。

附录1. 2. 2 集成的可配置调试

Cortex-M0 处理器实现了完整的硬件调试方案，带有大量的硬件断点和观察点选项。通过一个 2 引脚串行线调试（SWD）端口，为处理器、存储器和外设调试提供了较高的系统可见性。SWD 对微控制器和别的小封装设备是很理想的。

附录1. 2. 3 Cortex-M0 处理器特性小结

- ◇ 高代码密度，具有 32 位的性能
- ◇ 工具和二进制代码向上兼容 Cortex-M 系列处理器
- ◇ 集成了极低功耗的睡眠模式
- ◇ 高效的代码执行允许更慢的处理器时钟以及更长睡眠模式的时间
- ◇ 单周期的 32 位硬件乘法器
- ◇ 零抖动的中断处理
- ◇ 广泛的调试功能

附录1. 2. 4 Cortex-M0 核心外设

Cortex-M0 核心外设有：

NVIC — NVIC 是一个嵌入式中断控制器，支持低延迟的中断处理

系统时钟控制块 — 系统时钟控制块（SCB）是到处理器的编程模型接口。它提供系统执行和控制信息，包括配置、控制和系统异常的报告。

系统定时器 — 系统定时器（SysTick）是一个 24 位的减法定时器。可将其用作一个实时操作系统（RTOS）的节拍定时器，或者用作一个简单的计数器。

附录1. 3 处理器

附录1. 3. 1 编程模型

本节描述了 Cortex-M0 的编程模型。除了对单个内核寄存器的描述之外，本节还包含处理器模式和堆栈的相关信息。

附录1. 3. 1. 1 处理器模式

处理器模式有：

Thread 模式（线程模式） — 用来执行应用软件。处理器在退出复位时进入 Thread 模式。

Handler 模式（处理模式） — 用来处理异常。处理器在完成所有的异常处理后返回到 Thread 模式。

附录1. 3. 1. 2 堆栈

处理器使用满递减堆栈，这就意味着堆栈指针指向堆栈存储器中的最后一个堆栈项。当处理器将一个新的项压入堆栈时，堆栈指针递减，然后将该项写入新的存储器单元。处理器有两个堆栈，主堆栈和进程堆栈，两个堆栈有自己独立的堆栈指针副本，见**堆栈指针章节**。

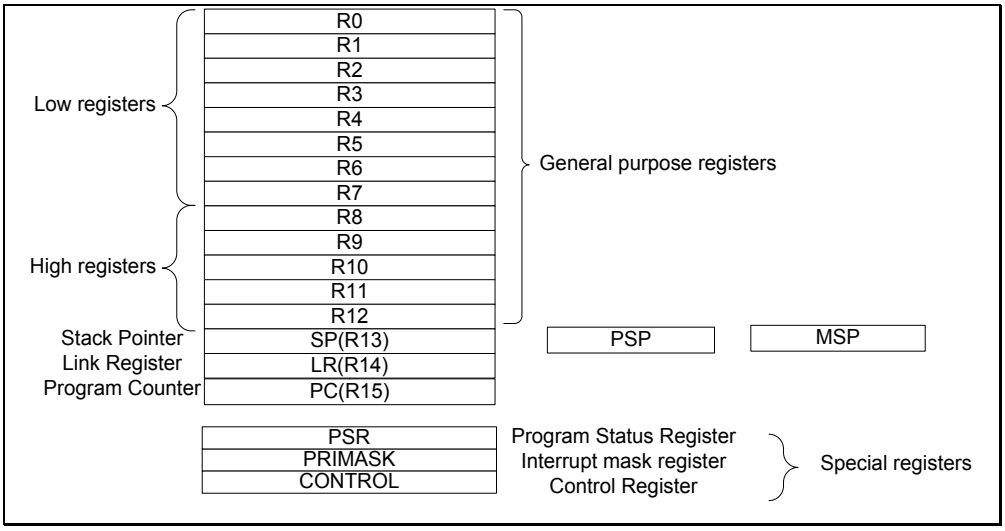
在线程模式下，CONTROL 寄存器控制着处理器使用主堆栈还是进程堆栈，见**处理器 - 控制寄存器章节**。在处理器模式下，处理器总是使用主堆栈。处理器操作的选择如下：

处理器模式	用来执行	使用的堆栈
Thread	应用程序	主堆栈或进程堆栈，见处理器 - 控制寄存器章节
Handler	异常处理程序	主堆栈

附录表 1-1 处理器模式和堆栈使用的选择

附录1. 3. 1. 3 内核寄存器

处理器内核寄存器有：



附录图 1-2 处理器核心寄存器组

名称	类型[1]	复位值	描述
R0-R12	RW	不可知	通用寄存器章节
MSP	RW	见描述	堆栈指针章节
PSP	RW	不可知	堆栈指针章节
LR	RW	不可知	链接寄存器章节
PC	RW	见描述	程序计数器章节
PSR	RW	不可知[2]	PSR 寄存器组合表格

名称	类型[1]	复位值	描述
APSR	RW	不可知	APSR 位分配表格
IPSR	R	0x00000000	IPSR 位分配表格
EPSR	R	不可知[2]	EPSR 位分配表格
PRIMASK	RW	0x00000000	PRIMASK 寄存器位分配表格
CONTROL	RW	0x00000000	CONTROL 寄存器位分配表格

附录表 1-2 内核寄存器组小结

注[1]：描述线程模式和处理模式下程序执行过程中的访问类型。调试访问可以不同。
注[2]：Bit[24]是 T 位，从复位向量的 bit[0]加载进来。

通用寄存器

R0-R12 是供数据操作使用的 32 位通用寄存器。

堆栈指针

堆栈指针（SP）是寄存器 R13。在 Thread 模式中，CONTROL 寄存器的 bit[1] 指示了堆栈

指针的使用情况：

◇ 0 = 主堆栈指针（MSP）。这是复位值。

◇ 1 = 进程堆栈指针（PSP）

复位时，处理器将地址 0x00000000 的值加载到 MSP 中。

链接寄存器

链接寄存器（LR）是寄存器 R14。它保存子程序、函数调用和异常的返回信息。复位时，LR 的值不可知。

程序计数器

程序计数器（PC）是寄存器 R15。它包含当前的程序地址。复位时，处理器将复位向量（地址：0x00000004）的值加载到 PC，该值的 bit[0] 复位时被加载到 EPSR 的 T 位，必须为 1。

程序状态寄存器

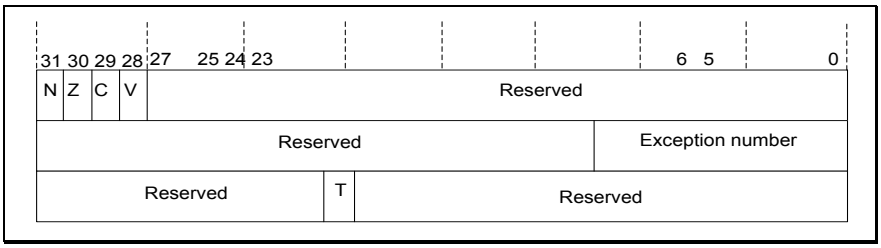
程序状态寄存器（PSR）由下列三种寄存器组合而成：

◇ 应用程序状态寄存器（APSR）

◇ 中断程序状态寄存器（IPSR）

◇ 执行程序状态寄存器（EPSR）

在 32 位的 PSR 中，这 3 个寄存器的位域分配互斥。PSR 的位域分配如下：



附录图 1-3 APSR，IPSR，EPSR 寄存器位分配

这 3 个寄存器可以单独访问，也可以 2 个一组或 3 个一组进行访问，访问时，将寄存器名称作为 MSR 或 MRS 指令的一个变量。例如：

◇ 使用寄存器名称 PSR，用 MRS 指令来读所有寄存器

◇ 使用寄存器名称 APSR，用 MSR 指令来写 APSR

PSR 的组合和属性如下所示：

寄存器	类型	组合
PSR	RW[1][2]	APSR，EPSR 和 IPSR
IEPSR	RO	EPSR 和 IPSR
IAPSR	RW[1]	APSR 和 IPSR
EAPSR	RW[2]	APSR 和 IPSR

附录表 1-3 PSR 寄存器组合

注[1]：处理器忽略对 IPSR 位的写操作

注[2]：读 EPSR 位时返回零，处理器忽略对 EPSR 位的写操作

有关访问程序状态寄存器的更多信息请参看指令描述的 MRS 指令和 MSR 指令。

应用程序状态寄存器：APSR 包含执行完前面的指令后条件标志的当前状态。有关寄存器的属性请见表格内核寄存器组小结。寄存器的位分配如下所示：

位域	名称	功能
[31]	N	负值标志
[30]	Z	零标志
[29]	C	进位或借位标志
[28]	V	溢出标志
[27:0]	-	保留

附录表 1-4 APSR 位分配

有关 APSR 的负值、零值、进位或借位以及溢出标志的更多信息请参考条件标志。

中断程序状态寄存器：IPSR 包含当前 中断服务程序（ISR）的异常编号。有关寄存器的属性请见表格内核寄存器组小结。该寄存器的位分配如下：

位域	名称	功能
[31:6]	-	保留
[5:0]	异常编号	这是当前异常的编号： 0=Thread 模式 1=保留 2=NMI 3=HardFault 4-10=保留 11-SVCall 12，13=保留 14=PendSV 15=SysTick 16=IRQ0 ..... 47=IRQ31 48-63=保留 更多信息请见异常类型

附录表 1-5 IPSR 位分配

执行程序状态寄存器：EPSR 包含 Thumb 状态位。

有关 EPSR 属性请见表格内核寄存器组小结的寄存器汇总。EPSR 的位分配如下：

位域	名称	功能
[31:25]	-	保留
[24]	T	Thumb 状态位
[23:0]	-	保留

附录表 1-6 EPSR 位分配

如果应用软件使用 MRS 指令直接读取 EPSR 将始终返回零。利用 MSR 指令来写 EPSR 的操作会被忽略。故障处理程序可以检查入栈的 PSR 的 EPSR 值来确定故障的原因。请见本章“**异常进入或返回**” 节。下面的操作可以清除 T 位的值为 0：

指令 BLX，BX 和 POP{PC}

◇ 异常返回时恢复被压入栈中的 xPSR 值

◇ 进入异常时向量值的 bit[0]

在 T 位为 0 时尝试执行指令会导致 HardFault 或锁定故障，更多信息请见锁定。

可中断— 可重启的指令：可中断- 可重启的指令有 LDM 和 STM。如果在执行这两条中的其中一条指令的过程中出现中断，处理器就放弃指令的执行。

在处理完中断后，处理器再从头开始重新执行指令。

异常屏蔽寄存器

异常屏蔽寄存器禁止处理器处理异常。当异常可能影响到时间关键性任务或要求连续执行的原子代码序列时，异常就被禁止。

可以使用 MSR 和 MRS 指令、或 CPS 指令改变 PRIMASK 的值来禁止或重新允许异常。更多信息请看指令 MRS，MSR 和 CPS。

优先级屏蔽寄存器：PRIMASK 寄存器阻止优先级可配置的所有异常被激活。有关寄存器的属性请看表格内核寄存器组小结。该寄存器的位分配如下：

位域	名称	功能
[3:1]	-	保留
[0]	PRIMASK	0=允许可配置优先级的所有异常被激活 1=阻止可配置优先级的所有异常被激活

附录表 1-7 PRIMASK 寄存器位分配

控制寄存器

CONTROL 寄存器控制着处理器处于 Thread 模式时所使用的堆栈。该寄存器的属性请看表格内核寄存器组小结 的寄存器汇总。该寄存器的位分配如下：

位域	名称	功能
[31:2]	-	保留
[1]	有效堆栈指针	定义当前的堆栈指针 0=MSP 是当前堆栈指针 1=PSP 是当前堆栈指针 在 Handler 模式中，这个位读出为 0，写操作被忽略
[0]	-	保留

附录表 1-8 CONTROL 寄存器位分配

处理模式始终使用 MSP，因此，在处理模式下，处理器忽略对 CONTROL 寄存器的有效堆栈指针位执行的明确的写操作。异常进入和返回机制会将 CONTROL 寄存器更新。

在一个 OS 环境中，推荐运行在线程模式中的线程使用进程堆栈，内核和异常处理器用主堆栈。

默认情况下，线程模式使用 MSP。要将线程模式中使用的堆栈指针切换成 PSP，只需要使用 MSR 指令将有效堆栈指针位设置为 1，请看 **指令 MRS**。

注意：当更改堆栈指针时，软件必须在 MSR 指令后立刻使用一个 ISB 指令。这样来保证 ISB 之后的指令执行时使用新的堆栈指针，请看指令 **ISB**。

附录1. 3. 1. 4 异常和中断

Cortex-M0 处理器支持中断和系统异常。处理器和内嵌向量中断控制器（NVIC）划分所有异常的优先级，并对所有异常进行处理。一个中断或异常会改变软件控制的正常流程。处理器使用处理模式来处理除复位之外的所有异常，更多信息请看**异常进入**和**异常返回**

NVIC 寄存器控制中断处理。更多信息请看**内嵌向量中断控制器**

附录1. 3. 1. 5 数据类型

处理器：

◇ 支持下列数据类型：
- 32 位字
- 16 位半字
- 8 位字节

◇ 管理所有数据存储器访问都采用小端模式。指令存储器和专用外设总线（PPB）访问。始终是小端模式。更多信息请看**存储区、类型和属性**

附录1. 3. 1. 6 Cortex 微控制器软件接口标准

ARM 为编程 Cortex-M0 微控制器提供了 Cortex 微控制器软件接口标准（CMSIS）。CMSIS 是设备驱动库的一个组成部分。

CMSIS 为 Cortex-M0 微控制器系统定义了：

◇ 一个通用的方法来：
- 访问外设寄存器
- 定义异常向量

◇ 以下名称：
- 寄存器和核心外设的名称
- 内核异常向量的名称

◇ 一个 RTOS 内核的与设备独立的接口

CMSIS 包含 Cortex-M0 处理器中核心外设的地址定义和数据结构。还包含有组成 TCP/IP 堆栈和 Flash 文件系统的中间件元件的可选接口。

通过允许模板代码的重复使用以及将不同中间件厂商提供的与 CMSIS 兼容的软件组件组合起来，CMSIS 大大简化了整个软件开发过程。软件厂商可以扩展 CMSIS，使其包含各个厂商的外设定义以及这些外设的访问函数。

本文档包含了 CMSIS 定义的寄存器名称，并对处理器内核和核心外设相关的 CMSIS 函数进行了简单描述。

注意：本文档使用 CMSIS 定义的寄存器缩略名称。在某些情况下，这些名称与其它文档中可能用到的结构缩略名称不同。

下面各节给出了有关 CMSIS 的更多信息：

◇ 电脑管理编程提示 “Power management programming hints”

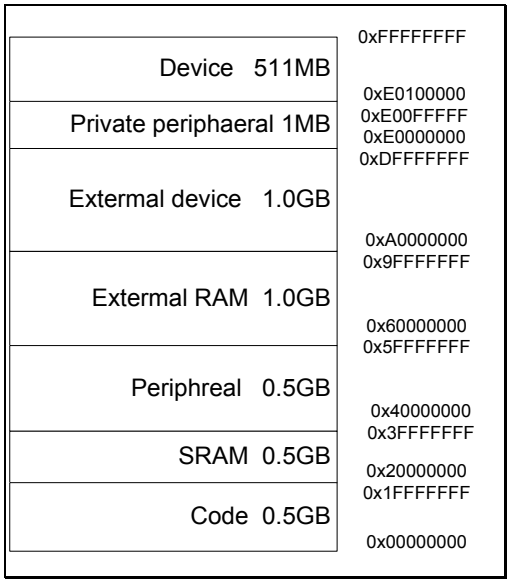
◇ 内部函数 “Intrinsic functions”

◇ 使用 CMSIS 访问 Cortex-M0 NVIC 寄存器 “Accessing the Cortex-M0 NVIC registers using CMSIS”

◇ NVIC 编程提示 “NVIC programming hints”

附录1. 3. 2 存储器模型

本节描述处理器存储器映射以及存储器访问的行为。处理器有一个固定的存储器映射，提供有高达 4GB 的可寻址存储空间。存储器映射是：



附录图 1-4 通用 ARM Cortex-M0 存储器映射

处理器为内核外设寄存器保留了专用外设总线(PPB)地址范围空间，请看**关于 Cortex-M0 处理器和核心外设**

附录1. 3. 2. 1 存储区、类型和属性

存储器映射分成多个区域。每个区域有一个定义好的存储器类型，某些区域还有附加的存储器属性。存储器类型和属性决定了各个区域的访问行为。

存储器类型是：

常规存储器 — 处理器为了提高效率，可以重新对事务进行排序，或者刻意地进行读取。

Device 存储器 — 处理器保留与 Device 存储器或强秩序存储器（Strong-ordered memory）事务相关的事务的秩序。

强秩序存储器 — 处理器保留与所有其他事务相关的事务秩序。

Device 存储器和强秩序存储器的不同秩序要求意味着，存储器系统可以缓冲一个对 Device 存储器的写操作，但不准缓冲对强秩序存储器的写操作。

附加的存储器属性包括：

永不执行（XN） — 表示处理器阻止指令访问。当执行从存储器的 XN 区提取出来的指令时，产生一个 HardFault 异常。

附录1. 3. 2. 2 存储系统的访问秩序

对于大多数由明确的存储器访问指令引发的存储器访问，存储器系统都不保证访问秩序与指令的编写顺序完全一致，只要访问秩序的重新安排不影响指令序列的行为特征就行。一般情况下，如果两个存储器访问的顺序必须与两条存储器访问指令编写的顺序完全一致程序才能正确执行，软件就必须在两条存储器访问指令之间插入一条内存屏障指令，请看**软件的存储器访问秩序**。

但是，存储器系统不保证 Device 存储器和强秩序存储器的一些访问秩序。对于两条存储器访问指令 A1 和 A2，如果 A1 的编写顺序在前，两条指令所引发的存储器访问顺序为：

A1 \ A2	Normal access	Device access		StrongIt-ordered access
		Non-shareable	Shareable	
Normal access	-	-	-	-
Device access，Non-shareable	-	<	-	<
Device access，Shareable	-	-	<	<
StrongIt-orderedaccess	-	<	<	<

附录表 1-9 存储器排序限制

在表中：

- — 表示存储器系统不保证访问秩序

< — 表示观察到访问顺序与指令编写顺序一致，即，A1 总是在 A2 之前

附录1. 3. 2. 3 存储器访问行为

存储器映射中每个区域的访问行为如下：

地址范围	存储区域	存储器类型[1]	XN[1]	描述
0x00000000-0x1FFFFFFF	Code	常规存储器	-	程序代码的可执行区域。也可以把数据保存到这里。
0x20000000-0x3FFFFFFF	SRAM	常规存储器	-	数据的可执行区域。也可以把代码保存到这里。
0x40000000-0x5FFFFFFF	外设	Device 存储器	XN	外部设备存储器
0x60000000-0x9FFFFFFF	外部 RAM	常规存储器	-	数据的可执行区域
0xA0000000-0xDFFFFFFF	外部设备	Device 存储器	XN	外部设备存储器
0xE0000000-0xE00FFFFF	专用外设总线	强秩序存储器	XN	这个区域包括 NVIC、系统定时器和系统控制块。这个区域只能使用字访问。
0xE0100000-0xFFFFFFFF	Device	Device 存储器	XN	厂商提供的特定存储器。

附录表 1-10 存储器访问行为

注[1]：更多信息请看存储区、类型和属性。

Code、SRAM 和外部 RAM 区域可以保存程序。

附录1. 3. 2. 4 软件的存储器访问秩序

程序流程的指令秩序并不能保证相应的存储器事务秩序。这是因为：

◇ 为了提高效率，处理器可以将一些处理器访问的秩序重新安排，只要不影响指令的行为特性就行。

◇ 存储器映射中的存储器或设备可能有不同的等待状态。

◇ 某些存储器访问被缓冲，或者是刻意为之的。

存储系统的访问秩序 描述了存储器系统在哪些情况下能保证存储器访问的秩序。但是，如果存储器访问的秩序十分重要，软件就必须插入一些内存屏障指令来强制保持存储器访问的秩序。处理器提供了以下内存屏障指令：

DMB — 数据存储屏障（DMB）指令保证先完成重要的存储事务，再执行后面的存储事务，见**指令 DMB**。

DSB — 数据同步屏障（DSB）指令保证先完成重要的存储器事务，再执行后面的指令，见**指令 DSB**。

ISB — 指令同步屏障（ISB）保证所有已完成的存储事务的结果，后面的指令都能辨认出来，见 **指令 ISB**。

下面是内存屏障指令使用的一些例子：

向量表 — 如果程序改变了向量表中的一个入口，然后又允许了相应的异常，那么就在操作之间插入一条 DMB 指令。这就能确保，如果异常在获得允许后立刻被调用，处理器能使用新的异常向量。

自修改代码 — 如果一个程序包含自修改代码，代码修改之后在程序中立刻使用一条 ISB 指令。这就确保后面的指令执行使用的是更新后的程序。

存储映射切换 — 如果系统包含一个存储器映射切换机制，在切换存储器映射之后使用一条 DSB 指令。这就确保了后面的指令执行使用的是更新后的存储器映射。

对强秩序存储器（例如，系统控制块）执行的存储器访问不需要使用 DMB 指令。

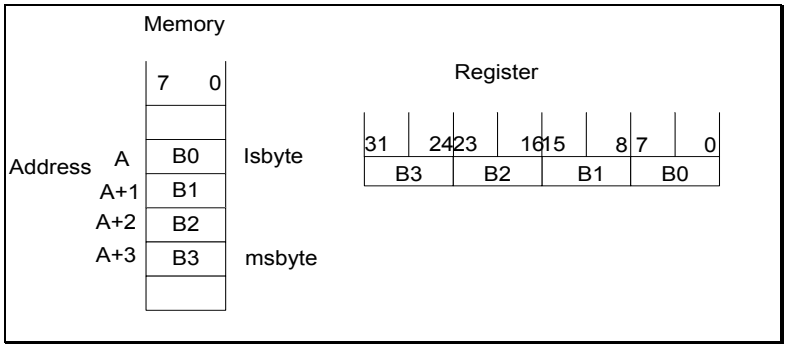
处理器保留与所有其他事务相关的事务顺序。

附录1. 3. 2. 5 存储器的字节存储顺序

处理器看到的存储器是一个从零开始、编号逐次递增的字节集合。例如，字节 0-3 存放第一个保存的字，字节 4-7 存放第二个保存的字。小端格式描述了数据的字在存储器中是如何存放的。

小端格式

在小端格式中，处理器将字的最低有效字节（lsbyte）保存在编号最小的字节中，最高有效字节（msbyte）保存在编号最大的字节中。例如：



附录图 1-5 小端格式

附录1. 3. 3 异常模型

本节描述异常模型。

附录1. 3. 3. 1 异常状态

每个异常都处于下面状态中的一种：

无效 — 异常无效，未挂起。

挂起 — 异常正在等待处理器处理。

一个外设或软件的中断请求可以改变相应的挂起中断的状态。

有效 — 一个异常正在被处理器处理，但处理尚未结束。

一个异常处理程序可以中止另一个异常处理程序的执行。在这种情况下，两个异常都处于有效状态。

有效且挂起 — 异常正在被处理器处理，而且有一个来自同一个异常源的异常正在等待处理。

附录1. 3. 3. 2 异常类型

异常类型有：

复位 — 复位在上电或热复位时启动。异常模型将复位当做一种特殊形式的异常来对待。当复位产生时，处理器的操作停止，可能停止在一条指令的任何一点上。当复位结束时，从向量表中复位入口的地址处重新启动执行。在线程模式下执行重启。

NMI — 一个不可屏蔽的中断（NMI）可以由外设产生，也可以由软件来触发。这是除复位之外优先级最高的异常。NMI 永远允许，优先级固定为 2。NMI 不能：

◇ 被屏蔽，它的执行也不能被其他任何异常中止

◇ 被除复位之外的任何异常抢占

HardFault — HardFault 是由于在正常操作过程中或在异常处理过程中出错而出现的一个异常。HardFault 的优先级固定为-1，表明它的优先级要高于任何优先级可配置的异常。

SVCall — 超级用户调用（SVC）异常是一个由 SVC 指令触发的异常。在 OS 环境下，应用程序可以使用 SVC 指令来访问 OS 内核函数和设备驱动程序。

PendSV — PendSV 是一个中断驱动的系统级服务请求。在 OS 环境下，当没有其它异常有效时，使用 PendSV 来进行上下文切换。

SysTick — SysTick 是一个系统定时器到达零时产生的异常。软件也可以产生个 SysTick 异常。在 OS 环境下，处理器可以将这个异常用作系统节拍。

中断（IRQ） — 中断（或 IRQ）是外设引起的一个异常，或者是由软件请求产生的一异常。所有中断都与指令执行不同步。在系统中，外设使用中断来与处理器通信。

异常编号[1]	IRQ 编号[1]	异常类型	优先级	向量地址[2]
1	-	复位	-3，优先级最高	0x00000004
2	-14	NMI	-2	0x00000008
3	-13	HardFault	-1	0x0000000C
4-10	-	保留	-	-
11	-5	SVCall	可配置[3]	0x0000002C
12-13	-	保留	-	-
14	-2	PendSV	可配置[3]	0x00000038
15	-1	SysTick	可配置[3]	0x0000003C
16	0 and above	中断（IRQ）	可配置[3]	0x00000040 and above[4]

附录表 1-11　各种异常类型的特性

注[1]：为了简化软件层，CMSIS 只使用 IRQ 编号，因此，对除中断外的其他异常都使用负值。IPSR 返回异常编号，请看表格 IPSR 位分配。
注[2]：更多信息请看向量表。
注[3]：请看中断优先级寄存器。
注[4]：地址值以 4 为步长，逐次递增。

对于除复位之外的异步异常，在异常被触发和处理器进入异常处理程序时间间隔内，处理器可以执行额外的指令。

被特许的软件可以将表格各种异常类型的特性 中列出的优先级可配置的异常禁止，请看**中断清除允许寄存器**。

有关 HardFault 的更多信息请看故障处理。

附录1. 3. 3. 3　异常处理程序

处理器使用以下处理程序来处理异常：

中断服务程序（ISR） — 中断 IRQ0~IRQ31 是由 ISR 来处理的异常

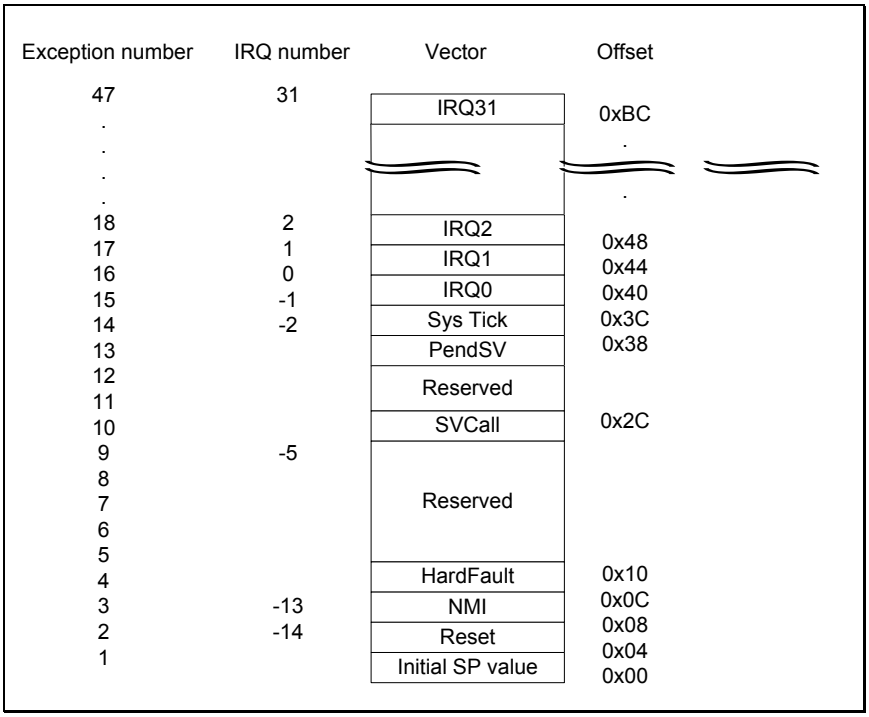
故障处理程序 — HardFault 是唯一一个由故障处理程序来处理的异常

系统处理程序 — NMI，PendSV，SVCall SysTick 和 HardFault 都是由系统处理程序来处理的异常。

附录1. 3. 3. 4　向量表

向量表包含堆栈指针的复位值以及所有向量处理程序的起始地址（也称为异常向量）。如下图显示了异常向量在向量表中的放置顺序。每个向量的最低有效位必须为 1，表明异

常处理程序都是用 Thumb 代码编写的。



附录图 1-6　向量表

向量表的地址固定为 0x00000000。

附录1. 3. 3. 5　异常优先级

如表格各种异常类型的特性所示，每个异常都有对应的优先级：

◇　越小的优先级值表示越高的优先级。

◇　除复位、HardFault 和 NMI 之外，所有异常的优先级都是可配置的。

如果软件不配置任何优先级，那么，所有优先级可配置的异常的优先级就都为 0。有关配置异常优先级的信息请见：

◇　系统处理程序优先级寄存器

◇　中断优先级寄存器

注：可配置优先级的值在 0—3 之间。复位、HardFault 和 NMI 这些有固定的负优先级值的异常的优先级高于任何其他的异常。

给 IRQ[0] 分配一个高优先级值、给 IRQ[1] 分配一个低优先级值就意味着 IRQ[1] 的优先级高于 IRQ[0]。如果 IRQ[1] 和 IRQ[0] 都有效，先处理 IRQ[1]。

如果多个挂起的异常具有相同的优先级，异常编号最小的挂起异常优先处理。例如，如果 IRQ[0] 和 IRQ[1] 正在挂起，并且两者的优先级相同，那么先处理 IRQ[0]。

当处理器正在执行一个异常处理程序时，如果出现一个更高优先级的异常，那么这个异常就被抢占。如果出现的异常的优先级和正在处理的异常的优先级相同，这个异常就不被抢占，与异常的编号大小无关。但是，新中断的状态就变为挂起。

附录1. 3. 3. 6 异常进入和返回

描述异常处理时使用了下列术语：

抢占 — 当处理器正在执行一个异常处理程序时，如果另一个异常的优先级比正在处理的异常的优先级更高，那么低优先级的异常就被抢占。

当一个异常抢占另一个异常时，这些异常就被称为嵌套异常。更多信息请见异常进入。

返回 — 当异常处理程序结束，并且满足以下条件时，异常就返回：

◇ 没有优先级足够高的挂起异常需要处理

◇ 已完成的异常处理程序没有在处理一个迟来的异常

处理器从堆栈弹出数据，使处理器状态恢复到中断出现之前的状态， 更多信息请看**异常返回**。

尾链 — 这个机制加速了异常的处理。当一个异常处理程序结束时，如果一个挂起的异常满足异常进入的要求，就跳过堆栈弹出，控制权移交给新的异常处理程序。

迟来 — 这个机制加速了抢占的处理。如果一个高优先级的异常在前一个异常正在保存状态的过程中出现，处理器就转去处理更高优先级的异常，开始提取这个异常的向量。状态保存不受迟来异常的影响，因为两个异常保存的状态相同。从迟来异常的异常处理程序返回时，要遵守正常的尾链规则。

异常进入

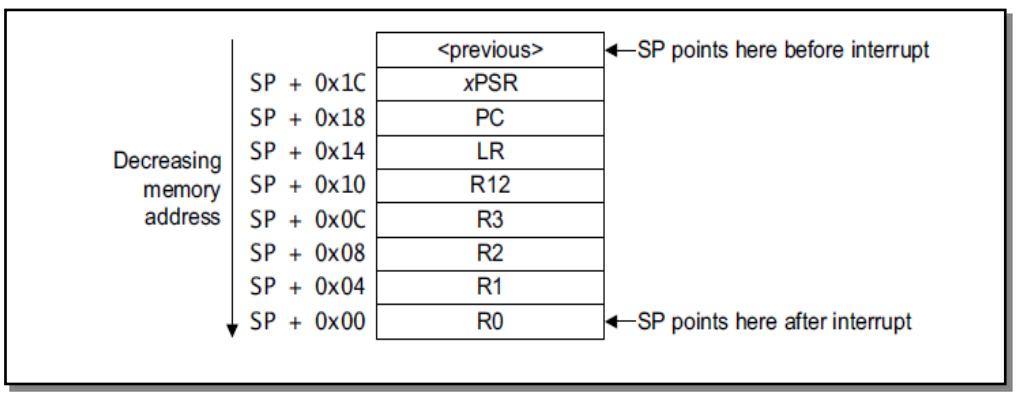
当有一个优先级足够高的挂起异常存在，并且满足下面的任何一个条件，就进入异处理：

◇ 处理器处于 Thread 模式

◇ 新异常的优先级高于正在处理的异常，这时新异常就抢占了正在处理的异常，当一个异常抢占了另一个异常时，异常就被嵌套。

优先级足够高的意思是该异常的优先级比屏蔽寄存器中所限制的任何一个异常组的优先级都要高，请看异常屏蔽寄存器。优先级比这个异常低的异常要被挂起，但不被处理器处理。

当处理器处理异常时，除非异常是一个末链异常或迟来的异常，否则，处理器都把信息压入到当前的堆栈中。这个操作被称为入栈（stacking），8 个数据字的结构被称为栈帧（stack frame）。栈帧包含以下信息：



附录图 1-7 异常入口堆栈的内容

入栈后，堆栈指针立刻指向栈帧的最低地址单元。栈帧按照双字地址对齐。

栈帧包含返回地址。这是被中止的程序中下条指令的地址。这个值在异常返回时返给 PC，使被中止的程序恢复执行。

处理器执行一次向量提取，从向量表中读出异常处理程序的起始地址。当入栈结束时，处理器开始执行异常处理程序。同时，处理器向 LR 写入一个 EXC_RETURN 值。这个值指示了栈帧对应哪个堆栈指针以及在异常出现之前处理器处于什么工作模式。

如果在异常进入的过程中没有更高优先级的异常出现，处理器就开始执行异常处理程序，并自动将相应的挂起中断的状态变为有效。

如果在异常进入的过程中有另一个优先级更高的异常出现，处理器就开始执行这个高优先级异常的异常处理程序，不改变前一个异常的挂起状态。这是一个迟来异常的情况。

异常返回

当处理器处于处理模式，并且执行下面一条指令试图将 PC 设为 EXC_RETURN 值时，出现异常返回：

◇ POP 指令，用来加载 PC

◇ BX 指令，使用任意寄存器

在异常进入时处理器将一个 EXC_RETURN 值保存到 LR 中。异常机制依靠这个值来检测处理器何时执行完一个异常处理程序。EXC_RETURN 值的 bit[31:4]为 0xFFFFFFF。当处理器将一个相应的这种形式的值加载到 PC 时，它将检测到这个操作并不是一个正常的分支操作，而是异常已经结束。因此，处理器启动异常返回。EXC_RETURN 的 bit[3:0]指出了所需的返回堆栈和处理器模式，如表格异常返回行为所示：

EXC_RETURN	描述
0xFFFFFFF1	返回到处理模式 异常返回从主堆栈获取状态信息 返回后执行使用 MSP
0xFFFFFFF9	返回到线程模式 异常返回从 MSP 获取状态信息 返回后执行使用 MSP
0xFFFFFFFD	返回到线程模式 异常返回从 PSP 获取状态信息 返回后执行使用 PSP
All other values	保留

附录表 1-12　异常返回行为

附录1. 3. 4 故障处理

故障是异常的一个子集，请看 **异常处理程序**。所有的故障都导致 HardFault 异常被处理，或者，如果故障在 NMI 或 HardFault 处理程序中出现，会导致锁定。发生以下情况会导致出现故障：

- ◇ 以等于或高于 SVCall 的优先级执行 SVC 指令
- ◇ 在没有调试器的情况下执行 BKPT 指令
- ◇ 在加载或存储时出现一个系统产生的总线错误
- ◇ 执行一个 XN 存储器地址中的指令
- ◇ 从系统产生了一个总线故障的地址单元中执行指令
- ◇ 在提取向量时出现了一个系统产生的总线错误
- ◇ 执行一个未定义的指令
- ◇ 由于 T 位之前被清零而导致不再处于 Thumb 状态的情况下执行一条指令
- ◇ 尝试对一个不对齐的地址执行加载或存储操作

注：只有复位和 NMI 可以抢占优先级固定的 HardFault 处理程序。HardFault 可以抢占除复位、NMI 或其他硬故外的任何异常。

附录1. 3. 4. 1 锁定

如果在执行 NMI 或 HardFault 处理程序时出现故障，或者，在一个使用 MSP 的异常返回时出栈的却是 PSR 的时候系统产生一个总线错误，处理器进入一个锁定状态。当处理器处于锁定状态时，它不执行任何指令。处理器保持处于锁定状态，直到下面任何一种情出现：

- ◇ 出现复位
- ◇ 调试器将锁定状态终止
- ◇ 出现一个 NMI，以及当前的锁定处于 HardFault 处理程序中

注：如果锁定状态出现在 NMI 处理程序中，后面的 NMI 就无法使处理器离开锁定状态。

附录1. 3. 5 电源管理

Cortex-M0 处理器的睡眠模式可以降低功耗，睡眠模式包含 2 种：

◇ 睡眠模式：停止处理器时钟

◇ 深度睡眠模式

SCR 的 SLEEPDEEP 位选择使用哪种睡眠模式，请看**系统控制寄存器**。

本节描述了进入睡眠模式的机制和将器件从睡眠模式唤醒的条件。

附录1. 3. 5. 1 进入睡眠模式

本节描述了软件可以用来使处理器进入睡眠模式的一种机制。

系统可以产生伪唤醒事件，例如，一个调试操作唤醒处理器。因此，软件必须能够在这样的事件之后使处理器重新回到睡眠模式。程序中可以有一个空闲循环让处理器回到睡眠模式。

等待中断

等待中断指令（WFI）使器件立刻进入睡眠模式。当执行一个 WFI 指令时，处理器停止执行指令，进入睡眠模式。更多信息请看指令 WFI。

等待事件

等待事件指令（WFE）根据一个一位的事件寄存器的值来进入睡眠模式。处理器执行一个 WFE 指令时检查事件寄存器的值：

0 — 处理器停止执行指令，进入睡眠模式

1 — 处理器将寄存器的值设为 0，并继续执行指令，不进入睡眠模式

更多信息请看**指令 WFE**。

如果事件寄存器为 1，表明处理器在执行 WFE 指令时不必进入睡眠模式。通常的原因

是出现了一个外部事件，或者系统中的另一个处理器已经执行了 SEV 指令，见**指令 SEV**。软件不能直接访问这个寄存器。

Sleep-on-exit

如果 SCR 的 SLEEPONEXIT 位被设为 1，当处理器完成一个异常处理程序的执行并返回到线程模式时，处理器立刻进入睡眠模式。如果应用只要求处理器在中断出现时运行，就可以使用这种机制。

附录1. 3. 5. 2 从睡眠模式唤醒

处理器的唤醒条件取决于使处理器进入睡眠模式所采用的机制。

从 WFI 或者 sleep-on-exit 唤醒

通常，只有当检测到一个优先级足够高的异常导致进入异常处理时，处理器才唤醒。

某些嵌入式系统在处理器唤醒之后可能必须先执行系统恢复任务，然后再执行中断处理程序。通过将 PRIMASK 位置位来实现这个操作。如果到来的中断被允许，并且优先级高于当前的异常优先级，处理器就唤醒，但不执行中断处理程序，直至处理器将 PRIMASK 设为 0，请看**异常屏蔽寄存器**。

从 WFE 唤醒

如果出现以下情况，处理器就唤醒：

◇ 处理器检测到一个优先级足够高的异常导致进入异常进入

◇ 在一个多处理器的系统中，系统中的另一个处理器执行了 SEV 指令

另外，如果 SCR 的 SEVONPEND 位被设为 1，那么任何新的挂起中断都能触发一个事件并唤醒处理器，即使这个中断被禁止，或者这个中断的优先级不够高而导致无法进入异常处理。有关 SCR 的更多信息请见**系统控制寄存器**。

附录1. 3. 5. 3 电脑管理编程提示

ISO/IEC C 不能直接产生 WFI、WFE 和 SEV 指令。CMSIS 为这些指令提供了以下内在函数：

```
void __WFE(void) // 等待事件

void __WFI(void) // 等待中断

void __SEV(void) // 发送事件
```

附录1. 4　指令集

附录1. 4. 1　指令集汇总

处理器执行一个版本的 Thumb 指令集。Cortex-M0 指令列出了所支持的指令。

注意：在 Cortex-M0 指令中

- ◇ 尖括号< >括着操作数的备用格式
- ◇ 大括号{}括着可选的操作数和助记符部分
- ◇ 操作数列所列出的操作数不完全

有关指令和操作数的信息，详见指令描述：

助记符	操作数	简述	标志	参考
ADCS	{Rd,}Rn,Rm	带进位加法	N，Z，C，V	ADC，ADD，RSB，SBC 和 SUB
ADD{S}	{Rd,}Rn,<Rm\#imm>	加法	N，Z，C，V	ADC，ADD，RSB，SBC 和 SUB
ADR	Rd,label	将基于 PC 相对偏移的地址读到寄存器	-	ADC，ADD，RSB，SBC 和 SUB
ANDS	Rd,}Rn,Rm	位与操作	N，Z	ADC，ADD，RSB，SBC 和 SUB
ASRS	{Rd,}Rm,<Rs\#imm>	算术右移	N，Z，C	ASR，LSL，LSR 和 ROR
B{cc}	label	跳转{有条件}	-	B，BL，BX 和 BLX
BICS	{Rd,}Rn,Rm	位清除	N，Z	AND，ORR，EOR 和 BIC
BKPT	#imm	断点	-	BKPT
BL	label	带链接的跳转	-	B，BL，BX 和 BLX
BLX	Rm	带链接的间接跳转	-	B，BL，BX 和 BLX
BX	Rm	间接跳转	-	B，BL，BX 和 BLX
CMN	Rn,Rm	比较负值	N，Z，C，V	CMP 和 CMN
CMP	Rn,<Rm\#imm >	比较	N，Z，C，V	CMP 和 CMN
CPSID	i	更改处理器状态，关闭中断	-	CPS
CPSIE	i	更改处理器状态，关闭中断	-	CPS
DMB	-	数据内存屏障	-	DMB
DSB	-	数据同步屏障	-	DSB
EORS	{Rd,}Rn,Rm	异或	N，Z	AND，ORR，EOR 和 BIC
ISB	-	指令同步屏障	-	ISB
LDM	Rn{!},reglist	加载多个寄存器，访问之	-	LDM 和 STM

助记符	操作数	简述	标志	参考
		后会递增地址		
LDR	Rt,label	从基于 PC 相对偏移地址上加载寄存器	-	存储器访问指令
LDR	Rt,[Rn,<Rm|#imm>]	用字加载寄存器	-	存储器访问指令
LDRB	Rt,[Rn,<Rm|#imm>]	用字节加载寄存器	-	存储器访问指令
LDRH	Rt,[Rn,<Rm|#imm>]	用半字加载寄存器	-	存储器访问指令
LDRSB	Rt,[Rn,<Rm|#imm>]	用有符号的字节加载寄存器	-	存储器访问指令
LDRSH	Rt,[Rn,<Rm|#imm>]	用有符号的半字加载寄存器	-	存储器访问指令
LSLS	{Rd,}Rn,<Rs|#imm>	逻辑左移	N，Z，C	ASR，LSL，LSR 和 ROR
U	{Rd,}Rn,<Rs|#imm>	逻辑右移	N，Z，C	ASR，LSL，LSR 和 ROR
MOV{S}	Rd,Rm	传输	N，Z	MOV 和 MVN
MRS	Rd,spec_reg	从特殊寄存器传输到通用寄存器	-	MRS
MSR	Spec_reg,Rm	从通用寄存器传输到特殊寄存器	N，Z，C，V	MSR
MULS	Rd,Rn,Rm	乘法，32 位结果值	N，Z	MULS
MVNS	Rd,Rm	位非	N，Z	MOV 和 MVN
NOP	-	无操作	-	NOP
ORRS	{Rd},Rn,Rm	逻辑或	N，Z	AND，ORR，EOR 和 BIC
POP	reglist	出栈。将堆栈的内容放入寄存器	-	PUSH 和 POP
PUSH	reglist	压栈，将寄存器的内容压入堆栈	-	PUSH 和 POP
助记符	操作数	简述	标志	参考
REV	Rd,Rm	反转字里面的字节顺序	-	REV，REV16 和 REVSH
REV16	Rd,Rm	反转每半字里面的字节顺序	-	REV，REV16 和 REVSH
REVSH	Rd,Rm	反转有符号半字里面的字节顺序	-	REV，REV16 和 REVSH
RORS	{Rd,}Rn,Rs	循环右移	N，Z，C	ASR，LSL，LSR 和 ROR
RSBS	{Rd,}Rn,#0	反向减法	N，Z，C，V	ADC，ADD，RSB，SBC 和 SUB
SBCS	{Rd,}Rn,Rm	进位减法	N，Z，C，V	ADC，ADD，RSB，SBC 和 SUB
SEV	-	发送事件	-	SEV
STM	Rn!,reglist	存储多个寄存器，在访问后地址递	-	LDM 和 STM
STR	Rt,[Rn,<Rm|#imm>]	将寄存器作为字来存储	-	存储器访问指令
STRB	Rt,[Rn,<Rm|#imm>]	将寄存器作为字节来存储	-	存储器访问指令

助记符	操作数	简述	标志	参考
STRH	Rt,[Rn,<Rm|#imm>]	将寄存器作为半字来存储	-	存储器访问指令
SUB{S}	{Rd,}Rn<Rm|#imm>	减法	N，Z，C，V	ADC，ADD，RSB，SBC 和 SUB
SVC	#imm	超级用户调用	-	SVC
SXTB	Rd，Rm	符号扩展字节	-	SXT 和 UXT
SXTH	Rd，Rm	符号扩展半字	-	SXT 和 UXT
TST	Rn，Rm	基于测试的逻辑与	N，Z	TST
UXTB	Rd，Rm	0 扩展字节	-	SXT 和 UXT
UXTH	Rd，Rm	0 扩展字节	-	SXT 和 UXT
WFE	-	等待事件	-	WFE
WFI	-	等待中断	-	WFI

附录表 1-13 Cortex-M0 指令

附录1. 4. 2 内部函数

ISO/IEC C 代码不能直接访问某些 Cortex-M0 指令。本章节对可以产生这些指令的内部函数进行了描述，内部函数可由 CMSIS 或有可能由 C 编译器提供。若 C 编译器不支持相关的内部函数，则用户可能需要使用内联汇编程序来访问相关的指令。

CMSIS 提供下列的内部函数来产生 ISO/IEC C 代码不能直接访问的指令：

指令	CMSIS 内部函数
CPSIE i	void__enable_irq(void)
CPSID i	void__disable_irq(void)
ISB	void__ISB(void)
DSB	void__DSB(void)
DMB	void__DMB(void)
NOP	void__NOP(void)
REV	uint32_t__REV(uimt32_t int value)
REV16	uint32_t__REV16(uimt32_t int value)
REVSH	uint32_t__REVSH(uimt32_t int value)
SEV	void__SEV(void)
WFE	void__WFE(void)
WFI	void__WFI(void)

附录表 1-14 产生某些 Cortex-M0 指令的 CMSIS 内部函数

CMSIS 还提供使用 MRS 和 MSR 指令来访问特别寄存器的函数：

特定寄存器	访问方式	CMSIS 函数
PRIMASK	读	uint32_t__get_PRIMASK（void）
	写	void__set_PRIMASK（uimt32_t value）
CONTROL	读	uint32_t__get_CONTROL（void）
	写	void__set_CONTROL（uimt32_t value）
MSP	读	uint32_t__get_MSP（void）
	写	void__set_MSP（uimt32_t TopOfMainStack）
PSP	读	uint32_t__get_PSP（void）
	写	void__set_PSP（uimt32_t TopOfMainStack）

附录表 1-15　访问特别寄存器的内部函数

附录1. 4. 3　关于指令的描述

下列小节对如何使用指令进行了更为详细的描述：

- ◇　操作数“Operands”
- ◇　使用 PC 或 SP 的限制“Restrictions when using PC or SP”
- ◇　移位操作“Shift Operations”
- ◇　地址对齐“Address alignment”
- ◇　PC 的相对表达式“PC- relative expressions”
- ◇　条件执行“Conditional execution”

附录1. 4. 3. 1　操作数

指令操作数可以是 ARM 寄存器，常量或其它的指令特定参数。指令在操作数上操作，并经常将结果存放在目的寄存器中。当指令中存在目的寄存器时，它通常会在其它操作数之前被指定。

附录1. 4. 3. 2　使用PC或SP的限制

对于用于操作数或目的寄存器的程序计数器（PC）或堆栈指针（SP），许多指令都不能使用它们，或者存在着用户能否使用它们的限制。更多信息详见指令的描述。

注意：当使用 BX、BLX 或 POP 指令来更新 PC 时，为正确执行程序，任何地址的位 0 都必须为 1。这是因为该位指示目标指令集，且 Cortex-M0 处理器只支持 Thumb 指令。当 BL 或 BLX 指令将位 0 的值写入 LR 时，值 1 会被自动分配。

附录1. 4. 3. 3　移位操作

寄存器移位操作通过特定的位数（移位长度）来实现寄存器位的左右移位操作。寄存器移位可以由指令 ASR、LSR、LSL 和 ROR 直接执行，且结果会被写入到目的寄存器中。允许的移位长度由移位类型和指令决定，请参考各个指令的描述。若移位长度为 0，则不发生移位操作。寄存器移位操作会更新进位标志，当移位长度被指定为 0 时除外。本节中的各小节描述了各种的移位操作以及它们是如何影响进位标志的。在这些描述中，Rm 是包含着移位值的寄存器，n 是移位长度。

ASR

算术右移 n 位的操作是将 Rm 寄存器左边的 32-n 个位向右移动 n 位，结果是寄存器右边有 32-n 个位，然后再将寄存器位[31]的原始值复制到结果寄存器左边的 n 位中，请看图片 **ASR #3**。

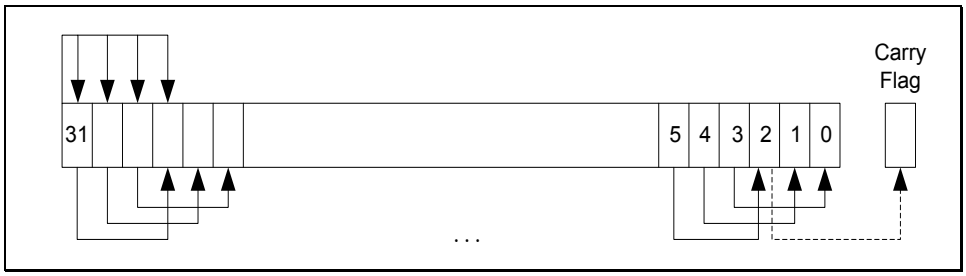
用户可以使用 ASR 对寄存器 Rm 的带符号数进行除以 2^n 的操作，得到的结果为负无穷大。

当指令为 ASRS 时，进位标志会被更新为最后移出的位值，即寄存器 Rm 的位[n-1]。

备注：

◇ 如果 n 为 32 或大于 32，那么结果中的所有位都会被置为 Rm 中位[31]的值。

◇ 如果 n 为 32 或大于 32，那么进位标志被更新为 Rm 位[31]的值。



附录图 1-8 ASR #3

LSR

逻辑右移 n 位的操作是将 Rm 寄存器 Rm 左边的 32-n 个位向右移动 n 位，结果寄存器右边有 32-n 位，然后再将结果寄存器左边的 n 个位设为 0。请看图片 LSR #3。

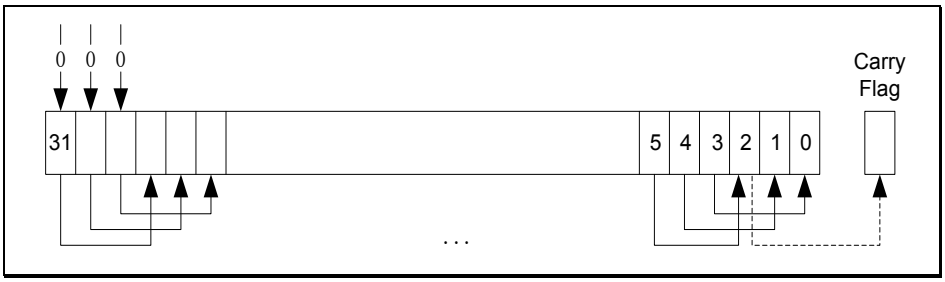
如果寄存器 Rm 值为无符号的整数，用户可以使用 LSR 操作来对其值进行除以 2^n 的操作。

当指令为 LSRS 时，进位标志会被更新为最后移出的位值，即寄存器 Rm 的位[n-1]。

备注：

◇ 如果 n 为 32 或大于 32，那么结果中的所有位都会被清除为 0。

◇ 如果 n 为 33 或大于 33，那么进位标志被更新为 0。



附录图 1-9 LSR #3

LSL

逻辑左移 n 位的操作是将 Rm 寄存器右边的 32-n 个位向左移动 n 位，结果寄存器左边有 32-n 个位，然后将结果中的寄存器右边的 n 个位设为 0。请看图片 LSL #3。

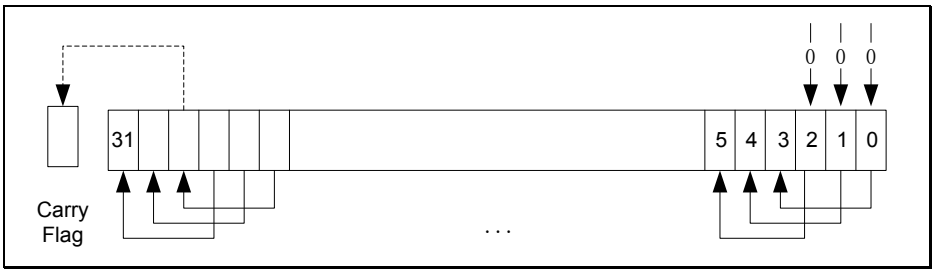
如果寄存器 Rm 值为无符号的整数或是有符号 2 的补码整数值，用户可以使用 LSL 操作来对其值与 2^n 进行乘法操作。溢出会在无警告提示下发生。

当指令为 LSLS 时，进位标志会被更新为最后移出的位值，即寄存器 Rm 的位[32-n]。当使用 LSL #0 时，这些指令不会影响进位标志。

备注：

◇ 如果 n 为 32 或大于 32，那么结果中的所有位都会被清除为 0。

◇ 如果 n 为 33 或大于 33，那么进位标志被更新为 0。



附录图 1-10 LSL #3

ROR

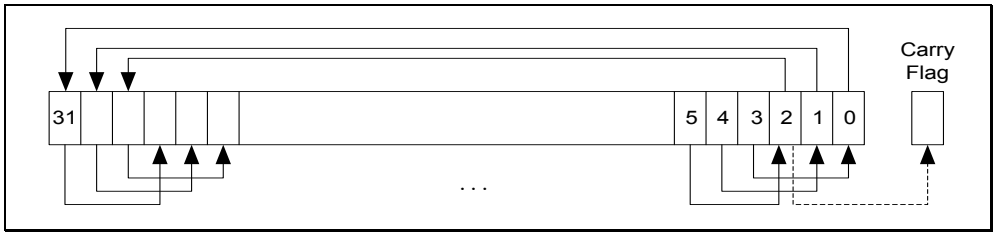
循环右移 n 位的操作是将 Rm 寄存器左边的 32-n 个位向右移动 n 位，结果是寄存器右边有 32-n 个位，然后再将寄存器右边的 n 个位移动到结果寄存器的左边的 n 个位中。请看图片 ROR #3。

当指令为 RORS 时，进位标志会被更新为最后循环出的位值，即寄存器 Rm 的[n-1]位。

备注：

◇ 如果 n 为 32，那么结果与 Rm 中的值相同，且如果进位标志被更新，则会被更新为 Rm 的位[31]的值。

◇ 移位长度 n 大于 32 的 ROR 与移位长度为 n-32 的 ROR 操作得到的结果相同。



附录图 1-11 ROR #3

附录1. 4. 3. 4 地址对齐

对齐访问是这样的一个操作：字对齐地址是用于字或多字访问，或者半字对齐地址是用于半字访问。字节访问通常是对齐访问的。

Cortex-M0 处理器不支持非对齐地址的访问。任何尝试执行一个非对齐的存储器访问操作都会导致 HardFault 异常。

附录1. 4. 3. 5 PC的相对表达式

相对 PC 表达或标签是一个代表着指令或文字数据的地址的符号。在指令中它被表示为 PC 值加上或减去一个数字偏移量。汇编器从标签和当前指令的地址中计算出所要求的偏移量。如果偏移量太大，则汇编器会产生一个错误。

备注：

◇ 对于大数多指令，PC 的值就是当前指令的地址加上 4 个字节。

◇ 汇编器可能允许用其它语法来表示 PC 相对表达式，如标签加上或减去一个数值，或者用[PC,#imm]格式表示。

附录1. 4. 3. 6 条件执行

大多数数据处理指令依据操作的结果在应用程序状态寄存器（APSR）中更新条件标志。某些指令更新所有标志，而某些指令则仅更新子集。如果标志不被更新，则原始值被保留。指令对标志的影响，请参考指令的描述。

在如下的情况下，用户可以在另一个指令中设置条件标志的基础上：

◇ 在指令更新标志后可立即执行条件性的跳转指令

◇ 在经过任意数量的没有更新标志的间隔数指令后，可以执行条件性的跳转指令

在 Cortex-M0 处理器上，通过使用条件性的跳转指令，就可以实现条件性的执行操作。

本小节描述了以下内容：

◇ **条件标志**“The condition flags”

◇ **条件代码后缀**“Condition code suffixes”

条件标志

APSR 包含了下列的条件标志：

N — 当操作的结果为负值时置为 1，否则清除为 0

Z — 当操作的结果为 0 时置为 1，否则清除为 0

C — 当操作的结果导致要进位时置为 1，否则清除为 0

V — 当操作引发溢出时置为 1，否则清除为 0

关于 APSR 的更多信息请看程序状态寄存器。

当出现下列情况时，会发生进位操作：

◇ 如果加法的结果大于或等于 2^{32}

◇ 如果减法的结果为正或等于 0

◇ 由于移位指令或循环指令而发生的进位操作

当位[31]中结果的符号值不与在无穷精度中所执行操作的结果符号值匹配时，溢出发生，

例如：

◇ 如果二个负值相加得出一个正值

◇ 如果二个正值相加得出一个负值

◇ 如果从一个负值减去一个正值得到一个正值

◇ 如果从一个正值减去一个负值得到一个负值

对于 CMP，比较操作与减法操作相同，对于 CMN，则与加法操作相同，结果值会被丢弃除外。更多信息，详情请参考指令描述。

条件代码后缀

条件性跳转在语法描述显示为B{cond}。只有 APSR 的条件代码标志符合指定的条件时，才能执行带有条件代码的跳转指令，否则要忽略跳转指令。

表格条件代码后缀显示了使用的条件代码，同时还显示了条件代码后缀和 N、Z、C 和 V 标志之关的联系。

后缀	标志	意义
EQ	Z=1	相等，最后标志设置结果为 0
NE	Z=0	不相等。最后标志设置结果为非 0
CS or HS	C=1	更高或相同，无符号
CC or LO	C=0	更低，无符号
MI	N=1	负数
PL	N=0	正数或 0
VS	V=1	溢出
VC	V=0	无溢出
HI	C=1 and Z=0	更高，无符号
LS	C=0 or Z=1	更低或相同，无符号
GE	N=V	大于或等于，有符号
LT	N!=V	少于，有符号
GT	Z=0 and N=V	大于，有符号
LE	Z=1 and N!=V	少于或等于，有符号
AL	Can have any value	总是。当没有指定后缀时，这是默认的操作

附录表 1-16　条件代码后缀

附录1. 4. 4 存储器访问指令

表格访问指令所示为存储器访问指令：

助记符	简单描述	参考
LDR{type}	使用寄存器偏移量来加载寄存器	LDR and STR，寄存器偏移量
LDR	基于 PC 相对地址来加载寄存器	LDR，PC 相对
POP	出栈，将栈中的内容存放寄存器	PUSH 和 POP
PUSH	压栈，将寄存器的内容压入堆栈	PUSH 和 POP
STM	存储多个寄存器	LDM 和 STM
STR{type}	使用立即数偏移量来存储寄存器	LDR and STR，立即数偏移量
STR{type}	使用寄存器偏移量来存储寄存器	LDR and STR，寄存器偏移量

附录表 1-17 访问指令

附录1. 4. 4. 1 ADR

产生一个 PC 相对地址。

语法

ADR Rd, label

其中：

Rd 是目标寄存器。

Label 是 PC 相对表达式。请看示例。

操作

ADR 通过将立即数值加到 PC 中来产生一个地址，并将得到的地址结果写入到目的寄存器中。

ADR 指令对产生与存储位置无关的代码非常便利，因为地址是 PC 相对地址。

如果用户使用 ADR 来产生 BX 或 BLX 指令的目标地址，为了能正确执行程序，必须要保证将产生的地址的位[0]设置为 1。

限制

在该指令中，Rd 必须指定 R0-R7。地址数据值必须是字对齐，且不能超出当前 PC 的 1020 字节。

条件标志

该指令不会改变标志。

示例

```
ADR R1, TextMessage; 将被标签为 TextMessage 单元上的地址值写入到 R1 中
ADR R3, [PC,#996]; 将 R3 的值设为 PC + 996
```

附录1. 4. 4. 2 LDR and STR，立即数偏移量

具有立即数偏移量的加载和存储。

语法

LDR Rt, [<Rn | SP> {, #imm}]
LDR<B|H> Rt, [Rn {, #imm}]
STR Rt, [<Rn | SP>, {,#imm}]
STR<B|H> Rt, [Rn {,#imm}]

其中：

Rt 是加载或存储的寄存器。

Rn 是寄存器，存储器地址基于此寄存器。

Imm 是 Rn 的偏移量。如果 imm 被省略，则假设它为 0。

操作

LDR、LDRB 和 LDRH 指令将存储器中的字、字节或半字数据值加载到 Rt 指定的寄存器中。在将数据写入 Rt 指定的寄存器之前，长度少于字的数据要用 0 扩充到 32 位的长度。

STR、STRB 和 STRH 指令将 Rt 寄存器指定的单个寄存器中所包含的字，最低位字节或低半字存放到存储器中。从加载的存储器地址或用于存放的存储器地址是 Rn 或 SP 所指定的寄存器的值与立即数 imm 的和。

限制

在这些指令中：

◇ Rt 和 Rn 必须只指定 R0-R7 的值

◇ Imm 的值必须要符合下列要求：

- 0 到 1020 之间，对于 LDR 和 STR 操作，在将 SP 用作基址寄存器时，其值必须是 4 的整数倍
- 0 到 124 之间，对于 LDR 和 STR 操作，在将 R0-R7 用作基址寄存器时，其值必须是 4 的整数倍
- 0 到 62 之间，对于 LDRH 和 STRH 操作，其值必须是 2 的整数倍
- 0 到 31 之间，对于 LDRB 和 STRB 操作

◇ 计算出的地址必须能够被事务中的字节数整除，请看**地址对齐**。

条件标志

这些指令不改变标志。

Examples

LDR R4, [R7]; 将 R7 的值作为地址，将此地址处的值载入到 R4 中

STR R2, [R0,#const-struc]; const-struc 是评估处于 0-1020 范围内的常量的表达式。

附录1. 4. 4. 3 LDR and STR，寄存器偏移量

带寄存器偏移量的加载和存储。

语法

LDR Rt, [Rn, Rm]
LDR<B|H> Rt, [Rn, Rm]
LDR<SB|SH> Rt, [Rn, Rm]
STR Rt, [Rn, Rm]
STR<B|H> Rt, [Rn, Rm]

其中：

Rt 是加载或存储的寄存器

Rn 是寄存器，存储器地址基于此寄存器

Rm 是含有用作偏移量的值的寄存器

操作

LDR、LDRB、U、LDRSB 和 LDRSH 将存储器中的字、0 扩展字节、0 扩展半字、符号扩展字节或符号扩展半字加载到 Rt 指定的寄存器中。

STR、STRB 和 STRH 指令将 Rt 寄存器指定的单个寄存器中所包含的字，最低位字节或低半字存放到存储器中。

从加载的存储器地址或用于存放的存储器地址是 Rn 和 Rm 所指定的寄存器中的值之和。

限制

在这些指令中：

◇ Rt、Rn 和 Rm 必须指定 R0-R7

◇ 计算出的地址必须能够被加载或存储的字节数整除。请看**地址对齐**。

条件标志

这些指令不改变标志。

示例

STR R0, [R5, R1]; 将 R0 的值存储到 R5 加 R1 得出的地址中。

LDRSH R1, [R2, R3]; 从（R2 + R3）所指定的存储器地址中加载半字数据，符号扩展到 32 位并将其写入到 R1 中。

附录1. 4. 4. 4 LDR，PC相对

从存储器中加载寄存器（文字数据）。

语法

LDR Rt, label

其中：

Rt 加载的寄存器

Label 是 PC 相对表达式，请看 **PC 的相对表达式**。

操作

将 label 所指定的存储器中的字加载到 Rt 所指定的寄存器中。

限制

在这些指令中，label 的大小必须位于当前 PC 的 1020 字节范围之内，且是字对齐的。

条件标志

这些指令不改变标志。

示例

LDR R0, LookUpTable; 将标签为 LookUpTable 的地址中的字数据加载到 R0 中。

LDR R3, [PC, #100]; 将（PC + 100）上的存储器字加载到 R3 中。

附录1. 4. 4. 5 LDM和STM

加载和存储多个寄存器。

语法

LDM Rn{!}, reglist
STM Rn!, reglist

其中：

Rn 是寄存器，存储器地址基于此寄存器。

!是回写后缀。

reglist 是被加载或存储的一个或多个寄存器的列表，用大括号括住。它包含着寄存器范围。若它包含着多于一个的寄存器或寄存器范围，必须要将其用逗号隔开，请看**示例**。

对于 LDM，LDMIA，它们和 LDMFD 相近。LDMIA 为每次访问后都会递增的基址寄存器。LDMFD 用法是将数据从满的递减堆栈中移出。

对于 STM，STMIA，它们和 STMEA 相近。STMIA 为每次访问后都会递增的基址寄存器。STMEA 用法是将数据压入空的递增堆栈中。

操作

LDM 指令将基于 Rn 上的存储器地址的字值加载到 reglist 的寄存器中。

STM 指令将 reglist 中的寄存器的字值存放到基于 Rn 的存储器地址中。

用于访问的存储器地址为 4 字节间隔，其范围为 Rn 所指定的寄存器的值至 Rn + 4 *（n-1）

所指定的寄存器的值，这里的 n 是 reglist 中的寄存器数量。访问的顺序是按照寄存器的编号从低到高发生，最低编号的寄存器使用最低的存储器地址，最高编号的寄存器使用最高的存储器地址。如果写回后缀被指定，则 Rn + 4 *n 所指定的寄存器的值会被写回到 Rn 所指定的寄存器中。

限制

在这些指令中：

◇ reglist 和 Rn 限制为 R0-R7

◇ 必须要使用写回后缀，除非指令是 LDM 指令，在 LDM 里，reglist 也含有 Rn，在这种情况下，要谨记不能用写回后缀。

◇ Rn 所指定的寄存器的值必须是字对齐的。更多信息请看**地址对齐**。

◇ 对于 STM，如果 reglist 中存在着 Rn，那么 Rn 必须是列表中的第一个寄存器。

条件标志

这些指令不改变标志。

示例

LDM R0,{R0,R3,R4}; LDMIA 相近于 LDM
STMIA R1!,{R2-R4,R6}

错误的示例

STM R5!,{R4,R5,R6}; 存放于 R5 的值是不可预测的
LDM R2,{}; 在列表中至少要存在着一个寄存器

附录1. 4. 4. 6 PUSH和POP

将寄存器压入满递减堆栈和将满递减堆栈中的内容移入寄存器。

语法

PUSH reglist
POP reglist

其中：

Reglist 是非空的寄存器列表，用大括号括着，它包含着寄存器范围。若它包含着多于一个的寄存器或寄存器范围，必须要将其用逗号隔开。

操作

PUSH 将寄存器存放到堆栈中，最低编号的寄存器使用低存储器地址，最高编号的寄存器使用高存储器地址。

POP 将堆栈中的内容加载到寄存器中，最低编号的寄存器使用最低存储器地址，最高编号的寄存器使用最高存储器地址。

PUSH 将 SP 寄存器的值减去 4 所得的值用作最高存储器地址，POP 将 SP 寄存器的值用作最低的存储器地址来执行满递减堆栈操作。当操作完成时，PUSH 会更新 SP 寄存器来指向最低存储值的单元，而 POP 则会更新 SP 寄存器来指向高于所加载的最高单元的单元。

如果 POP 在它的 reglist 中包含了 PC，则当 POP 指令完成时，会在该单元上执行一个跳转操作。为 PC 所读出的 Bit[0]值用来更新 EPSR T 位。该位必须为 1，以确保能正确执行程序。

限制

在这些指令中：

◇ reglist 必须只为 R0-R7

◇ 对于 PUSH 和 POP，异常情况分别是 LR 和 PC

条件标志

这些指令不改变标志。

示例

PUSH {R0,R4-R7}; 将 R0，R4，R5，R6，R7 压入堆栈

PUSH {R2,LR}; 将 R2 和链接寄存器压入堆栈
POP {R0,R6,PC}; 令 R0，R6 和 PC 出栈，然后跳转到新的 PC 值

附录1. 4. 5 通用数据处理指令

表格数据处理指令显示了数据处理指令：

助记符	简述	参考
ADCS	进位加法	ADC，ADD，RSB，SBC 和 SUB
ADD{S}	加法	ADC，ADD，RSB，SBC 和 SUB
ANDS	逻辑与	AND，ORR，EOR 和 BIC
ASRS	算术右移	ASR，LSL，LSR 和 ROR
BICS	位清零	AND，ORR，EOR 和 BIC
CMN	比较负值	CMP 和 CMN
CMP	比较	CMP 和 CMN
EORS	异或	AND，ORR，EOR 和 BIC
LSLS	逻辑左移	ASR，LSL，LSR 和 ROR
LSRS	逻辑右移	ASR，LSL，LSR 和 ROR
MOV{S}	传输	MOV 和 MVN
MULS	乘法	MULS
MVNS	取反传输	MOV 和 MVN
ORRS	逻辑或	AND，ORR，EOR 和 BIC
REV	反转字里面的字节顺序	REV，REV16 和 REVSH
REV16	反转每半字里面的字节顺序	REV，REV16 和 REVSH
REVSH	反转低半字中的字节顺序，并进行符号扩展	REV，REV16 和 REVSH
RORS	循环右移	ASR，LSL，LSR 和 ROR
RSBS	反向减法	ADC，ADD，RSB，SBC 和 SUB
SBCS	带进位减法	ADC，ADD，RSB，SBC 和 SUB
SUBS	减法	ADC，ADD，RSB，SBC 和 SUB
SXTB	符号扩展字节	SXT 和 UXT
SXTH	符号扩展字节	SXT 和 UXT
UXTB	零扩展字节	SXT 和 UXT
UXTH	零扩展字节	SXT 和 UXT
TST	测试	TST

附录表 1-18 数据处理指令

附录1. 4. 5. 1 ADC，ADD，RSB，SBC和SUB

进位加法、加法、反向减法、进位减法、减法。

语法

```
ADCS {Rd,} Rn, Rm
ADD{S} {Rd,} Rn, <Rm|#imm>
RSBS {Rd,} Rn, Rm, #0
SBCS {Rd,} Rn, Rm
SUB{S} {Rd,} Rn,
<Rm|#imm>
```

其中：

S 会令 ADD 或 SUB 指令更新标志

Rd 指定结果寄存器

Rn 指定首个源寄存器

Rm 指定第二个源寄存器

Imm 指定一个常量立即数值

当省略了可选的 Rd 寄存器限定符时，会假定其值与 Rn 相同，例如，ADDS R1,R2 与

ADDS R1,R1,R2 相同。

操作

ADCS 指令将 Rn 中的值加到 Rm 的值中，如果进位标志被置位，则将结果另行加 1，并将结果存放在 Rd 所指定寄存器里，同时更新 N、Z、C 和 V 标志。

ADD 指令将 Rn 的值加上 Rm 的值，或加上 imm 指定的立即数，并将结果存放到 Rd 所指定的寄存器中。

ADDS 指令执行的操作与 ADD 相同，并还可以更新 N、Z、C 和 V 标志。

RSBS 指令是用 0 减去 Rn 中的值，得到一个负数，然后将结果值存放在 Rd 所指定的寄存器中，并更新 N、Z、C 和 V 标志。

SBCS 指令是用 Rn 的值减去 Rm 的值，如果进位标志置位，则再减去一个 1。指令会将结果值存放到 Rd 所指定的寄存器中，并更新 N、Z、C 和 V 标志。

SUB 指令减去 Rm 的值或 imm 所指定的立即数。指令把结果值存放到 Rd 所指定的寄存器中。

SUBS 指令执行的操作与 SUB 相同，同时它还可以更新 N、Z、C 和 V 标志。

如何使用 ADC 和 SBC 来综合处理多字算术，请看**示例**。

还可以参考**指令 ADR**。

限制

ADC，ADD，RSB，SBC 和 SUB 操作数限制表格列出了寄存器指示符的合法组合和每

一个指令可以使用的立即数。

指令	Rd	Rn	Rm	imm	限制
ADCS	R0-R7	R0-R7	R0-R7	-	Rd 和 Rn 必须指定相同的寄存器
ADD	R0-R15	R0-R15	R0-PC	-	Rd 和 Rn 必须指定相同的寄存器 Rd 和 Rn 必须不能同时指定 PC
	R0-R7	SP or PC	-	0-1020	立即数必须为 4 的整数倍
	SP	SP	-	0-508	立即数必须为 4 的整数倍
ADDS	R0-R7	R0-R7	-	0-7	-
	R0-R7	R0-R7	-	0-255	Rd 和 Rn 必须指定相同的寄存器
	R0-R7	R0-R7	R0-R7	-	-
RSBS	R0-R7	R0-R7	-	-	-
SBCS	R0-R7	R0-R7	R0-R7	-	Rd 和 Rn 必须指定相同的寄存器
SUB	SP	SP	-	0-508	立即数必须为 4 的整数倍
SUBS	R0-R7	R0-R7	-	0-7	-
	R0-R7	R0-R7	-	0-255	Rd 和 Rn 必须指定相同的寄存器
	R0-R7	R0-R7	R0-R7	-	-

附录表 1-19 ADC，ADD，RSB，SBC 和 SUB 操作数限制

示例

下例所示为二个指令将 R0 和 R1 所包含的 64 位整数值加到 R2 和 R3 所包含的另一个 64 位整数值中，并将结果存放到 R0 和 R1 中。

64 位加法：

```
ADDS R0, R0, R2; 加上最低位的字
ADCS R1, R1, R3; 加上最高位的字，带进位
```

多字的值无需使用连续的寄存器。下面示例为指令会令 R4、R5 和 R6 所包含的 96 位整数值减去 R1、R2 和 R3 所包含的 96 位整数值。该例将结果值存放在 R4、R5 和 R6 中。

96 位减法：

```
SUBS R4, R4, R1; 减去最低位字
SBCS R5, R5, R2; 减去中间的字，带进位
SBCS R6, R6, R3; 减去最高位字，带进位
```

下列所示的 RSBS 指令是用来执行单个寄存器 1 的补码的操作。

算术负值运算：RSBS R7, R7, #0; 用 0 减去 R7。

附录1. 4. 5. 2 AND，ORR，EOR和BIC

逻辑 AND、OR、异或和位清除。

语法

```
ANDS {Rd,} Rn, Rm
ORRS {Rd,} Rn, Rm
EORS {Rd,} Rn, Rm
BICS {Rd,} Rn, Rm
```

其中

Rd 是目标寄存器

Rn 是保存第一个操作数的寄存器，且还是与目标寄存器相同的寄存器

Rm 是第二个寄存器

操作

AND、EOR 和 ORR 对 Rn 和 Rm 的值按位执行 AND、异或、或操作。

BIC 指令对 Rn 上的位，与 Rm 上的相应位执行逻辑非操作后，执行 AND 操作。

条件代码标志会根据操作的结果被更新，请看**条件标志**。

限制

在这些指令中，Rd、Rn 和 Rm 必须指定 R0-R7。

条件标志

这些指令会：

◇ 根据结果值来更新 N 和 Z 标志

◇ 不会影响 C 或 V 标志

示例

```
ANDS R2, R2, R1
ORRS R2, R2, R5
ANDS R5, R5, R8
EORS R7, R7, R6
BICS R0, R0, R1
```

附录1. 4. 5. 3 ASR，LSL，LSR和ROR

算术右移，逻辑左移，逻辑右移，循环右移。

语法

```
ASRS {Rd,} Rm, Rs
ASRS {Rd,} Rm, #imm
LSLS {Rd,} Rm, Rs
LSLS {Rd,} Rm, #imm
LSRS {Rd,} Rm, Rs
LSRS {Rd,} Rm, #imm
RORS {Rd,} Rm, Rs
```

其中：

Rd 是目的寄存器。如果 Rd 被省略，则假定它的值与 Rm 相同
Rm 是保存要移位的值的寄存器

Rs 是保存着移位长度（该长度要应用到 Rm 中的值）的寄存器
Imm 是移位长度

移位长度要由指令来决定：

ASR — 移位长度 1 到 32

LSL — 移位长度 0 到 31

LSR — 移位长度 1 到 32

注意：MOVS Rd, Rm 是 LSLS Rd, Rm, #0 的别名。

操作

ASR、LSL、LSR 和 ROR 对立即数 imm 所指定的长度而锁定的 Rm 寄存器的位或者 Rs 所指定的寄存器的最低位字节值执行算术左移、逻辑左移、逻辑右移或循环右移。

关于不同的指令会产生什么样的结果，请看**移位操作**。

限制

在这些指令中，Rd、Rm 和 Rs 必须只可以指定 R0-R7。对于非立即数指令，Rd 和 Rm 必须指定相同的寄存器。

条件标志

这些指令根据结果值来更新 N 和 Z 标志。

C 标志被更新为最后移出的位。当移位长度为 0 时例外，见**移位操作**。V 标志不变。

示例

```
ASRS R7, R5, #9; 算术右移 9 位
LSLS R1, R2, #3; 逻辑左移 3 位，并更新标志
LSRS R4, R5, #6; 逻辑右移 6 位
RORS R4, R4, R6; 循环右移 R6 低字节中的值
```

附录1. 4. 5. 4 CMP和CMN

比较和比较负值。

语法

CMN Rn, Rm
CMP Rn, #imm
CMP Rn, Rm

其中：

Rn 是保存第一个操作数的寄存器

Rm 是用于比较的寄存器

Imm 是用于比较的立即数值

操作

这些指令将一个寄存器中的值与另一个寄存器中的值或立即数进行比较。指令会根据结果值来更新条件标志，但不会将结果写入寄存器。

CMP 指令将 Rn 的值减去 Rm 所指定的寄存器值或立即数 imm，并更新标志。这操作与 SUBS 指令相同，不同的是结果值会被丢弃。

CMN 指令将 Rm 的值加到 Rn 的值中，并更新标志。这操作与 ADDS 指令相同，不同的是结果值会被丢弃。

限制

对于：

◇ CMN 指令

指令 Rn、Rm 必须只能指定 R0-R7。

◇ CMP 指令：

- Rn 和 Rm 可以指定 R0-R14
- 立即数的范围为 0-255

条件标志

这些指令根据结果值来更新 N、Z、C 和 V 标志。

示例

```
CMP R2, R9
CMN R0, R2
```

附录1. 4. 5. 5 MOV和MVN

传输和取反传输。

语法

MOV{S} Rd, Rm
MOVS Rd, #imm
MVNS Rd, Rm

其中：

S 是可选后缀。如果指定了 S，则会根据操作的结果值来更新条件代码标志，请看**条件执行**小节。

Rd 是目的寄存器

Rm 是寄存器

Imm 可以是 0-255 范围内的任何一个值

操作

MOV 指令将 Rm 的值复制到 Rd 中。

MOVS 指令执行的操作与 MOV 指令相同，但是它会更新 N 和 Z 标志。

MVNS 指令采用 Rm 的值，对该值执行按位的逻辑取反操作，并将结果存放到 Rd 中。

限制

在这些指令中，Rd 和 Rm 必须指定 R0-R7。

当在 MOV 指令里 Rd 是 PC 时：

◇ 结果值的位[0]被丢弃

◇ 在通过将结果值的位[0]强制为 0 来所生成的地址上执行跳转操作。T 位保持不变。

注：尽管可以将 MOV 用作跳转指令，但是为了软件的可移植性，AMR 强烈建推荐使用 BX 或 BLX 指令来执行跳转操作。

条件标志

如果 S 被指定，则这些指令会：

◇ 根据结果值更新 N 和 Z 标志

◇ 不会影响 C 或 V 标志

示例

```
MOVS R0, #0x000B; 将 0x000B 写入 R0，更新标志
MOVS R1, #0x0; 将 0 写入 R1，更新标志
MOV R10, R12; 将 R12 的值写入 R10，不更新标志
MOVS R3, #23; 将 23 写入 R3
MOV R8, SP; 将堆栈指针的值写入 R8
MVNS R2, R0; 将 R0 取反写入 R2 并更新标志
```

附录1. 4. 5. 6 MULS

使用 32 位操作数的乘法，产生 32 位的结果值。

语法

MULS Rd, Rn, Rm

其中：

Rd 是目的寄存器

Rn、Rm 是保存进行乘法操作值的寄存器

操作

MUL 指令将 Rn 和 Rm 所指定的寄存器的值进行乘法操作，并将结果值的最低 32 位存放在 Rd 中。条件代码标志会按照操作的结果值而被更新，请看**条件执行**。

该指令的结果并不是由操作数是有符号还是无符号来决定。

限制

在该指令中：

◇ Rd、Rn 和 Rm 必须只能指定 R0-R7

◇ Rd 必须要和 Rm 相同

条件标志

该指令会：

- 根据结果值来更新 N 和 Z 标志
- 不会影响 C 或 V 标志

示例

MULS R0, R2, R0; 乘法操作，标志被更新，R0 = R0 x R2

附录1. 4. 5. 7 REV，REV16 和REVSH

反转字节。

语法

REV Rd, Rn
REV16 Rd, Rn
REVSH Rd, Rn

其中：

Rd 是目的寄存器

Rn 是源寄存器

操作

使用这些指令来改变数据的端点排序：

REV — 将 32 位大端数据转换成小端的数据或将 32 位小端的数据转换成大端数据

REV16 — 将二个打包的 16 位大端数据转换成小端的数据或将二个打包的小端的数据转换成大端数据

REVSH — 将 16 位有符号的大端数据转换成 32 位有符号小端数据或将 16 位有符号小端数据转换 32 位有符号大端数据

限制

在这些指令中，Rd 和 Rn 必须只可以指定 R0-R7。

条件标志

这些指令不改变标志。

示例

```
REV R3, R7;       反转 R7 值的字节顺序，并将其写入 R3
REV16 R0, R0;     反转 R0 中的每一个 16 位半字的字节顺序
REVSH R0, R5;     反转有符号的半字
```

附录1. 4. 5. 8 SXT 和UXT

符号扩展和 0 扩展。

语法

```
SXTB Rd, Rm
SXTH Rd, Rm
UXTB Rd, Rm
UXTH Rd, Rm
```

其中：

Rd 是目的寄存器

Rm 是寄存器，其保存的值会被扩展

操作

这些指令从结果值中提取位：

- ◇ SXTB 提取位[7:0] 并将值进行符号扩展到 32 位
- ◇ UXTB 提取位[7:0] 并将值用 0 扩展到 32 位
- ◇ SXTH 提取[15:0] 并将值进行符号扩展到 32 位
- ◇ UXTH 提取[15:0] 并将值用 0 扩展到 32 位

限制

在这些指令中，Rd 和 Rm 必须只可以指定 R0-R7。

条件标志

这些指令不改变标志。

示例

```
SXTH R4, R6; 获取 R6 的低半字，然后将其进行符号扩展到 32 位，并将结果写入 R4
UXTB R3, R1; 获取 R10 最低位字节，并用 0 扩展，最后将结果写入 R3
```

附录1. 4. 5. 9 TST

测试位。

语法

TST Rn, Rm

其中：

Rn 是保存第一个操作数的寄存器

Rm 是测试的寄存器

操作

该指令将一个寄存器的值与另一个寄存器中的值进行测试。它会根据结果值来更新条件标志，但是不会将结果值写入寄存器。

TST 指令对 Rn 中的值和 Rm 中的值执行位与操作。这是与 ANDS 指令相同的操作，不同的是它会丢弃结果值。

为了测试 Rn 中的某个位是 0 还是 1，要使用 TST 指令，且寄存器的该位要设为 1，其它所有位被清除为 0。

限制

在这些指令中，Rn 和 Rm 必须只能指定 R0-R7。

条件标志

这些指令：

◇ 会根据结果来更新 N 和 Z 标志

◇ 不会影响 C 或 V 标志

示例

TST R0, R1; 对 R0 值和 R1 值执行位与操作，更新条件代码标志，但结果值会被丢弃。

附录1. 4. 6 跳转和控制指令

表格跳转和控制指令 所示位跳转和控制指令：

助记符	简述	参考
B{cc}	跳转{有条件}	B，BL，BX 和 BLX
BL	带链接的跳转	B，BL，BX 和 BLX
BLX	带链接的间接跳转	B，BL，BX 和 BLX
BX	间接跳转	B，BL，BX 和 BLX

附录表 1-20 跳转和控制指令

附录1. 4. 6. 1 B，BL，BX和BLX

跳转指令。

语法

B{cond} label
BL label
BX Rm
BLX Rm

其中：

cond 是可选的条件代码，请看 **条件执行**。

label 是 PC 相对表达式， 请看 PC **的相对表达式**。

Rm 是提供跳转地址的寄存器

操作

所有这些指令都会对 label 所指示的地址或在 Rm 所指定的寄存器中包含地址上执行跳转操作。另外：

◇ BL 和 BLX 指令将下一个指令的地址写入 LR，链接寄存器 R14

◇ 如果 Rm 的位[0] 是 0，则 BX 和 BLX 指令会导致 HardFault 异常

BL 和 BLX 指令还会将 LR 的位[0] 设置为 1。这就确保了该值适合由后续 POP{PC} 或 BX 指令使用其来执行成功的返回跳转操作。

表格跳转范围 所示为适用于各种跳转指令的跳转范围。

指令	跳转范围
B label	-2KB 到+2KB
Bcond label	-256 字节到+254 字节
BL label	-16MB 到+16MB
BX Rm	寄存器中的任何值
BLX Rm	寄存器中的任何值

附录表 1-21 跳转范围

限制

在这些指令中：

◇ 不要在 BX 或 BLX 指令里使用 SP 或 PC

◇ 对于 BX 和 BLX，为实现正确的执行操作，Rm 的位[0] 必须为 1。位[0] 用于更新 EPSR T 位，并会被从目标地址上丢弃

注意：Bcond 是在 Cortex-M0 处理器上唯一的条件指令。

条件标志

这些指令不改变标志。

示例

```
B loopA; 跳转到 loopA

BL funC; 对函数 funC 进行带链接的跳转（调用），返回存放在 LR 的地址
BX LR; 从函数调用中返回

BLX R0; 带链接的跳转，并从（调用）中更改为存放在 R0 的地址
BEQ labelD; 条件跳转到 labelD，如果 Z 标志置位则跳转，否则不执行跳转
```

附录1. 4. 7 杂项指令

表格综合指令 所示为余下的 Cortex-M0 指令：

助记符	简述	参考
BKPT	断点	BKPT
CPSID	更改处理器状态，禁止中断	CPS
CPSIE	更改处理器状态，允许中断	CPS
DMB	数据内存屏障	DMB
DSB	数据同步屏障	DSB
ISB	指令同步屏障	ISB
MRS	从特殊寄存器传输到寄存器	MRS
MSR	从寄存器传输到特殊寄存器	MSR
NOP	空操作	NOP
SEV	发送事件	SEV
SVC	超级用户调用	SVC
WFE	等待事件	WFE
WFI	等待中断	WFI

附录表 1-22 综合指令

附录1. 4. 7. 1 BKPT

断点。

语法

BKPT #imm

其中：

imm 是 0-255 范围内的整数。

操作

BKPT 指令会令处理器进入调试状态。当指令到达特定的地址时，调试工具可以使用该指令来查询系统状态。处理器会忽略 imm。如有需要，调试器可以使用它来存放断点的其它信息。

如果在执行 BKPT 指令时调试器没有连接上，那么处理器还有可能会产生 HardFault 或进入锁定状态。更多信息请看 **锁定** 。

限制

没有限制。

条件标志

该指令不改变标志。

示例

BKPT #0; 立即数值设为 0x0 的断点

附录1. 4. 7. 2 CPS

更改处理器状态 。

语法

CPSID i
CPSIE i

操作

CPS 更改 PRIMASK 特殊寄存器值。通过设置 PRIMASK，CPSID 可令中断被关闭。而通过清除 PRIMASK，CPSIE 则可允许中断。关于这些寄存器的详细描述，更多信息请看**异常屏蔽寄存器** 。

限制

没有限制。

条件标志

该指令不改变标志。

示例

CPSID i; 关闭所有的中断，NMI 除外（设置 PRIMASK）
CPSIE i; 使能中断（清除 PRIMASK）

附录1. 4. 7. 3 DMB

数据内存屏障。

语法

DMB

操作

DMB 用作数据内存屏障。它可确保先检测到程序中位于 DMB 指令前的所有显式内存访问指令，然后再检测到程序中位于 DMB 指令后的显式内存访问指令。它不影响其他指令（不访问内存的指令）在处理器上的执行顺序。

限制

没有限制。

条件标志

该指令不改变标志。

示例

DMB; 数据内存屏障

附录1. 4. 7. 4 DSB

数据同步屏障。

语法

DSB

操作

DSB 用作特殊数据同步内存屏障，只有当此指令执行完毕后，才会执行程序中位于此指令后的指令。位于此指令前的所有显式内存访问均完成时，DSB 指令才会完成。

限制
没有限制。

条件标志
该指令不改变标志。

示例
DSB ；数据同步屏障

附录1. 4. 7. 5 ISB

指令同步屏障。

语法
ISB

操作
ISB 用作指令同步屏障。它会刷新处理器的管道，因此在完成了 ISB 指令后，需要再次将 ISB 之后的所有指令从高速缓存或内存中提取出来。

限制
没有限制。

条件标志
该指令不改变标志。

示例
ISB ；指令同步屏障

附录1. 4. 7. 6 MRS

将特殊寄存器的内容移动到通用寄存器中。

语法
MRS Rd, spec_reg
其中：

Rd 是通用目的寄存器。

spec_reg 是其中一个特殊寄存器：APSR、IPSR、EPSR、IEPSR、IAPSR、EAPSR、PSR、MSP、PSP、PRIMASK 或 CONTROL

操作
MRS 将特殊寄存器的内容存放到通用寄存器中。MRS 指令可以结合 MSR 指令来产生读-修改-写序列，这适用于在 PSR 中修改特别标志。

请看指令 MSR。

限制
在该指令中，Rd 必须不能是 SP 或 PC。

条件标志
该指令不改变标志。

示例

MRS R0, PRIMASK; 读取 PRIMASK 值并将其写入 R0

附录1. 4. 7. 7 MSR

将通用寄存器的内容传移到指定的特别寄存器中

语法

MSR spec_reg, Rn

其中：

Rn 是通用源寄存器

spec_reg 是特别目的寄存器：APSR、IPSR、EPSR、IEPSR、IAPSR、EAPSR、PSR、MSP、PSP、PRIMASK 或 CONTROL

操作

MSR 使用 Rn 所指定的寄存器的值来更新其中一个特殊寄存器。

请看指令 MRS。

限制

在该指令里，Rn 必须不能为 SP 和 PC。

条件标志

该指令明确地根据 Rn 中的值来更新标志。

示例

MSR CONTROL, R1; 读取 R1 的值，并将其写入 CONTROL 寄存器

附录1. 4. 7. 8 NOP

空操作。

语法

NOP

操作

NOP 执行的是无操作，且不能保证会占用指令时间。处理器可在它到达执行阶段之前将其从管道中移除。

使用 NOP 指令来进行填充，例如，在 64 位边界上放置后续指令。

限制

没有限制。

条件标志

该指令不改变标志。

示例

NOP ；空操作

附录1. 4. 7. 9 SEV

发送事件。

语法

SEV

操作

SEV 将带有信号的事件发送到一个多处理器系统内的所有处理器中。它还可设置局部事件寄存器。请看**电源管理**。

也可以参考 **指令 WFE**。

限制

没有限制。

条件标志

该指令不改变标志。

示例

SEV ； 发送事件

附录1. 4. 7. 10 SVC

超级用户调用。

语法

SVC #imm

其中：

Imm 是 0-255 范围内的整数

操作

SVC 指令会引发 SVC 异常。

处理器会忽略 imm。如果有需要，可以通过异常处理程序获取 imm 来决定要请求什么样的服务程序。

限制

没有限制。

条件标志

该指令不改变标志。

示例

SVC #0x32; 超级用户调用（SVC 处理程度使用堆栈的 PC 来锁定立即数的位置，然后将其提取出来）。

附录1. 4. 7. 11 WFE

等待事件。

语法

WFE

操作

如果事件寄存器为 0，则 WFE 挂起执行，直至发生以下事件之一：

◇ 出现异常，除非异常屏蔽寄存器或当前优先级级别将其屏蔽

◇ 异常进入挂起状态，如果系统控制寄存器的 SEVONPEND 置位

◇ 存在调试进入请求，如果调试允许的话

◇ 外设或多处理器系统里另一个处理器通过使用 SEV 指令来发出信号事件

如果事件寄存器为 1，则 WFE 将其清除为 0 并立即完成操作。

更多信息请看 **电源管理**。

注意：WFE 的目的只是用于省电。当写软件时，假定 WFE 作为 NOP 运行。

限制

没有限制。

条件标志

该指令不改变标志。

示例

```
WFE ；等待事件
```

附录1. 4. 7. 12 WFI

等待中断。

语法

```
WFI
```

操作

WFI 挂起执行，直至发生以下事件之一：

◇ 一个异常

◇ 中断变为挂起状态，如果 PRIMASK 被清除，则该中断占用优先权

◇ 存在调试进入请求，无论调试是否被允许

注意：WFI 的目的只是用于省电。当写软件时，假定 WFI 作为 NOP 运行。

限制

没有限制

条件标志

该指令不改变标志。

示例

```
WFI ；等待中断
```

附录1. 5 外设

附录1. 5. 1 关于ARM Cortex-M0

专用外设总线（PPB）的地址映射为：

地址	核心外设	描述
0xE000E008-0xE000E00F	系统控制块	表格 SCB 寄存器小结
0xE000E010-0xE000E01F	系统定时器	表格系统定时寄存器小结
0xE000E100-0xE000E4EF	内嵌向量中断控制器	表格 NVIC 寄存器小结
0xE000ED00-0xE000ED3F	系统控制块	表格 SCB 寄存器小结
0xE000EF00-0xE000EF03	内嵌向量中断控制器	表格 NVIC 寄存器小结

附录表 1-23 核心外设寄存器区

在寄存器描述中，寄存器的类型有以下几种：

RW — 读和写

R — 只读

W — 只写

附录1. 5. 2 内嵌向量中断控制器

本节描述内嵌向量中断控制器（NVIC）以及它使用的寄存器。NVIC 支持：

- ◇ 32 个中断
- ◇ 每个中断的优先级可编程为 0~3 四种级别。级别越高对应的优先级越低。因此，级别 0 是最高的中断优先级
- ◇ 中断信号的电平和脉冲检测
- ◇ 中断尾链
- ◇ 一个外部不可屏蔽中断（NMI）。

处理器在异常进入时自动使它的状态入栈，在异常退出时自动使它的状态出栈，无需采用任何指令。这就实现了低延迟的异常处理。NVIC 的硬件寄存器有：

地址	名称	类型	复位值	描述
0xE000E100	ISER	RW	0x00000000	中断设置允许寄存器
0xE000E180	ICER	RW	0x00000000	中断清除允许寄存器
0xE000E200	ISPR	RW	0x00000000	中断设置挂起寄存器
0xE000E280	ICPR	RW	0x00000000	中断清除挂起寄存器
0xE000E400-0xE000E41C	IPR0-7	RW	0x00000000	中断优先级寄存器

附录表 1-24 NVIC 寄存器小结

附录1. 5. 2. 1 使用CMSIS 访问Cortex-M0 NVIC 寄存器

CMSIS 函数允许在不同的 Cortex-M 系列中进行软件移植。

当利用 CMSIS 来访问 NVIC 寄存器时要用到以下函数：

CMSIS 函数	描述
void NVIC_EnableIRQ(IRQn_Type IRQn)[1]	允许中断和异常
void NVIC_DisableIRQ(IRQn_Type IRQn)[1]	禁止中断和异常
void NVIC_SetPendingRQ(IRQn_Type IRQn)[1]	将中断或异常的挂起状态设为 1
void NVIC_ClearPendingIRQ(IRQn_Type IRQn)[1]	将中断或异常的挂起状态清 0
Uint32_t NVIC_GetPendingIRQ(IRQn_Type IRQn)[1]	读取中断或异常的挂起状态。如果挂起状态被设为 1，这个函数就返回非 0 值
void NVIC_SetPriority(IRQn_Type IRQn, uint32_t priority)[1]	将一个优先级可配置的中断或异常的优先级设置为级别 1
Uint32_t NVIC_ GetPriority (IRQn_Type IRQn)[1]	读取一个优先级可配置的中断或异常的优先级。这个函数返回当前的优先级级别

附录表 1-25 CMSIS 访问 NVIC 的函数

注[1]：输入参数 IRQn 是 IRQ 编号，更多信息请看**表格各种异常类型的特性**

附录1. 5. 2. 2 中断设置允许寄存器

ISER 允许中断，并显示哪些中断被允许。 有关寄存器属性请见 **表格 NVIC 寄存器小结**。

该寄存器的位分配如下：

位域	名称	功能
[31:0]	SETENA	中断设置-允许位 写： 0=无影响 1=使能中断 读： 0=中断被禁止 1=中断被允许

附录表 1-26 ISER 位分配

如果一个挂起中断被允许，NVIC 就根据它的优先级来激活该中断。如果一个中断未被允许，使该中断的中断信号有效可将中断的状态变成挂起，但是，不管这个中断的优先级如何，NVIC 都不会激活该中断。

附录1. 5. 2. 3 中断清除允许寄存器

ICER 禁止中断，并显示哪些中断被允许。有关寄存器属性请看**表格 NVIC 寄存器小结** 。

该寄存器的位分配如下：

位域	名称	功能
[31:0]	CLRENA	中断清除-允许位 写： 0=无影响 1=禁止中断 读： 0=中断被禁止 1=中断被允许

附录表 1-27 ICER 位分配

附录1. 5. 2. 4 中断设置挂起寄存器

ISPR 强制中断进入挂起状态，并显示哪些中断正在挂起。有关寄存器属性请看**表格 NVIC 寄存器小结**。

该寄存器的位分配如下：

位域	名称	功能
[31:0]	SETPEND	中断设置-挂起位 写： 0=无影响 1=中断状态变为挂起 读： 0=中断没有挂起 1=中断正在挂起

附录表 1-28 ISPR 位分配

注意：向 ISPR 位写 1 相当于下面两种情况：

◇ 正在挂起的中断不会有任何影响

◇ 被禁止的中断会将中断的状态设置成挂起

附录1. 5. 2. 5 中断清除挂起寄存器

ICPR 使中断离开挂起状态，并显示哪些中断正在挂起。有关寄存器的属性请看**表格 NVIC 寄存器小结** 。

该寄存器的位分配如下：

位域	名称	功能
[31:0]	CLRPEND	中断清除-挂起位 写： 0=无影响 1=清除中断的挂起状态 读： 0=中断没有挂起 1=中断正在挂起

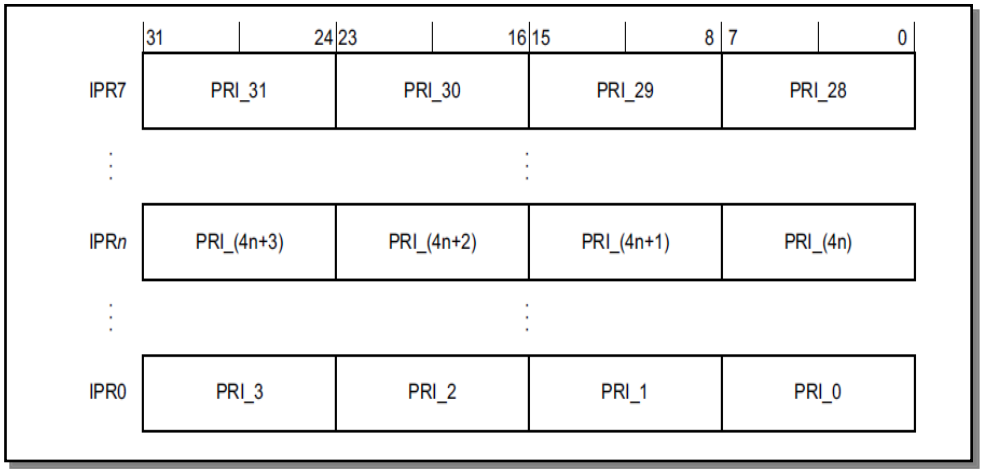
附录表 1-29 ICPR 位分配

注：向 ICPR 位写 1 不影响相应中断的有效状态。

附录1. 5. 2. 6 中断优先级寄存器

IPR0-IPR7 寄存器为每个中断提供了一个 8 位的优先级域。这些寄存器只能字访问。有关它们的属性请看 **表格 NVIC 寄存器小结** 。

每个寄存器有 4 个优先级域，如下所示：



附录图 1-12 IPR 寄存器

位域	名称	功能
[31:24]	Priority,byte offset3	每个优先级域保存一个优先级值（0~3）。值越小，对应中断的优先级越高。处理器只使用每个域的 bit[7:6]，bit[5:0]读出为 0，写操作被忽略
[23:16]	Priority,byte offset2	
[15:8]	Priority,byte offset1	
[7:0]	Priority,byte offset0	

附录表 1-30 IPR 位分配

有关中断优先级数组（提供了中断优先级的软件视角）访问的更多信息请参考**使用 CMSIS 访问 Cortex-M0 NVIC 寄存器。**

使用下面的方法为中断 M 找出 IPR 编号和字节偏移量：

◇ 相应的 IPR 编号 N，通过等式 N = N/4 得出

◇ 这个寄存器中所需优先级域的字节偏移量是 M MOD 4（M 除以 4 取余），在这里：

- 字节偏移量 0 指的是寄存器位[7:0]
- 字节偏移量 1 指的是寄存器位[15:8]
- 字节偏移量 2 指的是寄存器位[23:16]
- 字节偏移量 3 指的是寄存器位[31:24]

附录1. 5. 2. 7 电平有效的中断和脉冲中断

处理器支持电平中断和脉冲中断。脉冲中断也被描述成边沿触发的中断。

电平中断一直要保持电平有效，直至外设将中断信号撤销。通常，发生这种情况的原因是 ISR 访问外设导致外设将中断请求清除。脉冲中断是在处理器时钟的上升沿同步采样到中断信号。为了确保 NVIC 检测到中断，外设必须使中断信号至少在一个时钟周期内保持有效，在这段时间内 NVIC 检测脉冲并锁存中断。

当处理器进入 ISP 时，它自动消除中断的挂起状态，见**中断的硬件和软件控制**。对于电平中断，如果在处理器从 ISR 返回之前中断信号未被撤销，中断就再次变成挂起，处理器必须再次执行 ISR。这就表示，外设可以一直使中断信号保持有效，直到它不再需要服务为止。

中断的硬件和软件控制

Cortex-M0 锁存所有的中断。外设中断会由于以下原因之一而变为挂起：

◇ NVIC 检测到中断信号有效，而相应的中断无效

◇ NVIC 检测到中断信号的一个上升沿

◇ 软件向相应的中断设置- 挂起寄存器位写入值，请见**中断设置挂起寄存器**。

挂起的中断一直保持挂起，直到出现以下情况之一：

◇ 处理器进入中断 ISR，这就使中断的状态从挂起变为有效。而且：

- 对于电平中断，当处理器从 ISR 返回时，NVIC 采样中断信号。如果中断信号有效，中断的状态变回挂起，这可能使得处理器立刻再次进入 ISR。否则，中断的状态变为无效。
- 对于脉冲中断，NVIC 继续监测中断信号，如果中断信号一直处于脉冲状态，中断的状态就变成挂起和有效。在这种情况下，当处理器从 ISR 返回时，中断的状态变为挂起，这可能使得处理器立刻重新进入 ISR。如果当处理器在处理 ISR 时中断信号的脉冲就不存在了，那么，当处理器从 ISR 返回时中断的状态变为无效。

◇ 利用软件向相应的中断清除- 挂起寄存器位写入值。对于电平中断，如果中断信号仍然有效，中断的状态不改变。否则，中断的状态变为无效。

对于脉冲中断，中断的状态变为：

- 无效（如果中断之前的状态是挂起）

- 有效（如果中断之前的状态是有效和挂起）

附录1. 5. 2. 8 NVIC 使用提示和技巧

保证软件正确使用对齐的寄存器访问。处理器不支持不对齐的 NVIC 寄存器访问。

中断即使被禁止也可以进入挂起状态。禁止一个中断只阻止处理器处理中断。

NVIC 编程提示

软件使用 CPSIE i 和 CPSID i 指令来允许和禁止中断。CMSIS 为这些指令提供以下内在函数：

```
void __disable_irq(void) // 禁止中断
void __enable_irq(void) // 允许中断
```

另外，CMSIS 提供了许多 NVIC 控制函数，包括：

CMSIS 中断控制函数	描述
void NVIC_EnabelIRQ(IRQn_t IRQn)	允许 IRQn
void NVIC_DisabelIRQ(IRQn_t IRQn)	禁止 IRQn
uint32_t NVIC_GetPendingIRQ(IRQn_t IRQn)	如果 IRQn 正在挂起，返回 True（1）
void NVIC_SetpendingIRQ(IRQn_t IRQn)	设置 IRQn 挂起状态
void NVIC_ClearpendingIRQ(IRQn_t IRQn)	清除 IRQn 挂起状态
void NVIC_SetPriority(IRQn_t IRQn, uint32_t priority)	设置 IRQn 优先级
uint32_t NVIC_GetPriority(IRQn_t IRQn)	读取 IRQn 优先级
void NVIC_SystemReset(void)	复位系统

附录表 1-31 CMSIS 的 NVIC 控制函数

输入参数 IRQn 是 IRQ 编号。有关这些函数的更多信息，见**各种异常类型的特性**。

附录1. 5. 3 系统控制块

系统控制块（SCB）提供了系统执行和控制信息，包括配置、控制和系统异常的报告。SCB 寄存器有：

地址	名称	类型	复位值	描述
0xE00ED00	CPUID	R	0x410CC200	CPUID 寄存器
0xE00ED04	ICSR	RW[1]	0x00000000	中断控制和状态寄存器
0xE00ED0C	AIRCR	RW[1]	0xFA050000	应用中断和复位控制寄存器
0xE00ED10	SCR	RW	0x00000000	系统控制寄存器
0xE00ED14	CCR	R	0x00000204	配置和控制寄存器
0xE00ED1C	SHPR2	RW	0x00000000	系统处理程序优先级寄存器 2
0xE00ED20	SHPR3	RW	0x00000000	系统处理程序优先级寄存器 3

附录表 1-32 SCB 寄存器小结

注[1]：更多信息请看寄存器描述

附录1. 5. 3. 1 Cortex-M0 SCB 寄存器的CMSIS 映射

为了提高软件效率，CMSIS 简化了 SCB 寄存器的表现形式。在 CMSIS 中，数组 SHP[1] 对应寄存器 SHPR2-SHPR3。

附录1. 5. 3. 2 CPUID 寄存器

CPUID 寄存器包含处理器的型号、版本和实现信息。有关它的属性请见 SCB 寄存器小结。CPUID 的位分配如下：

位域	名称	功能
[31:24]	Implementer	实现代码：0x41=ARM
[23:20]	Variant	更新编号，产品版本标识符 rnpn 中 r 的值：0x0=版本 0
[19:16	Consant	定义处理器结构的常量；读取的结果是：0xC=ARMv6-M 结构
[15:4]	Partno	处理器的型号：0xC20=Cortex-M0
[3:0]	Revision	修订编号，产品版本标识符 rnpn 中的 p 的值：0x0=Patch0

附录表 1-33 CPUID 寄存器位分配

附录1. 5. 3. 3 中断控制和状态寄存器

ICSR：

◇ 提供了：

- 为不可屏蔽中断（NMI）异常提供了一个设置- 挂起位
- 为 PendSV 和 SysTick 异常提供了设置- 挂起位和清除- 挂起位

◇ 指明了：

- 正在处理的异常的异常编号
- 是否有被抢占的有效异常
- 最高优先级挂起异常的异常编号
- 是否有任何中断正在挂起

有关 ICSR 的属性请见**表格 SCB 寄存器小结**。ICSR 的位分配如下：

位域	名称	类型	功能
[31]	NMIPENDSET	RW	NMI 设置-挂起位 写： 0=无影响 1=将 NMI 异常的状态变为挂起 读： 0=NMI 异常未挂起 1=NMI 异常正在挂起 由于 NMI 是优先级最高的异常，因此，一般情况下，处理器一旦检测到向该位写 1 就立刻进入 NMI 异常处理程序。处理器进入处理程序后将该位清零。这就表示，只有当 NMI 信号在处理器正在执行 NMI 异常处理程序的过程中再次有效，通过异常处理程序读取这个位才返回 1。
[30:29]	-	-	保留
[28]	PENDSVSET	RW	PendSV 设置-挂起位 写： 0=无影响 1=将 PendSV 异常的状态变为挂起 读： 0=PendSV 异常未挂起 1=PendSV 异常正在挂起 向该位写 1 是将 PendSV 异常状态设为挂起的唯一办法
[27]	PENDSVCLR	W	PendSV 清除-挂起位 写： 0=无影响 1=撤销 PendSV 异常的挂起状态
[26]	PENDSTSET	RW	SysTick 异常设置-挂起位 写： 0=无影响 1=将 SysTick 异常的状态变为挂起 读： 0=SysTick 异常未挂起 1=SysTick 异常正在挂起
[25]	PENDSTCLR	W	SysTick 异常清除-挂起位 写： 0=无影响 1=撤销 SysTick 异常的挂起状态 该位只可写。当对这个寄存器执行读操作时，该位读出的值不可知
[24:23]	-	-	保留
[22]	ISPRENDING	R	除 NMI 和故障之外的中断的挂起标志 0=中断未挂起

位域	名称	类型	功能
			1=中断正在挂起
[21:18]	-	-	保留
[17:12]	VECTPEDING	R	指示优先级最高的、正在挂起的并且允许的异常的异常编号： 0=没有正在挂起的异常 非零=优先级最高的、正在挂起的并且允许的异常的异常编号
[11:6]	-	-	保留
[5:0]	VECTSCTIVE[1]	R	包含有效的异常编号： 0=Thread 模式 非零=当前有效异常的异常编号 注意：这个值减去 16 得到 CMSIS IRQ 编号，该编号标识出对应在中断清除-允许、设置-允许、清除-挂起、设置-挂起以及优先级寄存器中的位，请看表格 IPSR 位分配

附录表 1-34 ICSR 位分配

注[1]：这个值与 IPSR 位[5:0] 的值相同。

写 ICSR 时，如果执行下列操作，结果将不可知：

◇ 写 1 到 PENDSVSET 位和写 1 到 PENDSVCLR 位

◇ 写 1 到 PENDSTSET 位和写 1 到 PENDSTCLR 位

附录1. 5. 3. 4 应用中断和复位控制寄存器

AIRCR 提供了数据访问的字节顺序状态和系统的复位控制信息。有关寄存器的属性请见表“SCB 寄存器小结”和“AIRCR 位分配”。

如果要写这个寄存器，必须先向 VECTKEY 域写入 0x05FA，否则，处理器会将写操作忽略。

AIRCR 的位分配如下：

位域	名称	类型	功能
[31:16]	Read:Reserved Write:VECTKEY	RW	寄存器码： 读出的值不可知 执行写操作时将 0x05FA 写入 VECTKEY，否则写操作被忽略
[15]	ENDIANESS	R	采用的数据字节存储顺序： 0=小端 1=大端
[14:3]	-	-	保留
[2]	SYSRESETREQ	W	系统复位请求： 0=无影响 1=请求一个系统级复位 这个位读出为 0
[1]	VECTCLRACTIVE	W	保留供调试使用。这个位读出为 0。当写这个寄存器时，必

位域	名称	类型	功能
			须向这个位写 0，否则操作将不可预知
[0]	-	-	保留

附录表 1-35 AIRCR 位分配

附录1. 5. 3. 5 系统控制寄存器

SCR 控制着低功耗状态的进入和退出特性。有关寄存器的属性请见 **SCB 寄存器小结** 。SCR 的位分配如下：

位域	名称	功能
[31:5]	-	保留
[4]	SEVONPEMD	挂起时发送事件位： 0=只有允许的中断或事件才能唤醒处理器。不接受被禁止的中断的唤醒。 1=允许的事件和包括被禁止的中断在内的所有中断都能唤醒处理器。 当一个事件或中断进入挂起状态时，事件信号将处理器从 WFE 唤醒。如果处理器并未在等待一个事件，事件被记录，影响下一个 WFE。 处理器也可以在执行 SEV 指令或外部事件时唤醒
[3]	-	保留
[2]	SLEEPDEEP	控制处理器是将睡眠模式还是深度睡眠模式作为低功耗模式： 0=睡眠 1=深度睡眠
[1]	SLEEPONEXIT	指示当从处理器模式返回到线程模式时 sleep-on-exit(退出时进入睡眠)： 0=处理器返回到线程模式时不进入睡眠 1=处理器从 ISR 返回到线程模式时进入睡眠或深度睡眠 将该位设为 1 允许一个中断驱动的应用程序避免返回到一个空的主应用程序
[0]	-	保留

附录表 1-36 SCR 位分配

附录1. 5. 3. 6 配置和控制寄存器

CCR 是一个只读寄存器，指出了 Cortex-M0 处理器行为的一些情况。有关 CCR 属性请见 **SCB 寄存器小结**。

CCR 的位分配如下：

位域	名称	功能
[31:10]	-	保留
[9]	STKALIGN	该位读出总是为 1，指示进入异常时堆栈按 8 字节对齐。 进入异常时，处理器使用入栈的 PSR 的 bit[9]来指示栈对齐。从异常中返回时，处理器使用这个入栈的位来恢复正确的栈对齐。
[8:4]	-	保留
[3]	UNALIGN_TRP	该位读出总是为 1。指示所有未对齐的访问产生一个 HardFault
[2:0]	-	保留

附录表 1-37 CCR 位分配

附录1. 5. 3. 7 系统处理程序优先级寄存器

SHPR2-SHPR3 寄存器设置优先级可配置的异常处理程序的优先级级别（0-3）。

SHPR2-SHPR3 是字可访问的。有关它们的属性请见 **SCB 寄存器小结**。

利用 CMSIS 访问系统异常的优先级级别要用到以下 CMSIS 函数：

```
uint32_t NVIC_GetPriority(IRQn_Type IRQn)
void NVIC_SetPriority(IRQn_Type IRQn, uint32_t priority)
```

输入参数 IRQn 是 IRQ 编号，更多信息请看**各种异常类型的特性**。

系统故障处理程序、优先级域以及每个处理程序的寄存器如下所示：

处理程序	域	寄存器描述
SVCall	PRI_11	系统处理程序优先级寄存器 2
PendSV	PRI_14	系统处理程序优先级寄存器 3
SysTick	PRI_15	

附录表 1-38 系统故障处理程序优先级域

每个 PRI_N 域 8 位宽，但处理器只使用每个域的 bit[7:6] ； bit[5:0] 读出为 0，写操作被忽略。

系统处理程序优先级寄存器 2

该寄存器的位分配如下：

位域	名称	功能
[31:24]	PRI_11	系统处理程序 11（SVCall）的优先级
[23:0]	-	保留

附录表 1-39 SHPR2 寄存器位分配

系统处理程序优先级寄存器 3

该寄存器的位分配如下：

位域	名称	功能
[31:24]	PRI_15	系统处理程序 15（SysTick 异常）的优先级
[23:16]	PRI_14	系统处理程序 14（PendSV）的优先级
[15:0]	-	保留

附录表 1-40 SHPR3 寄存器的位分配

附录1. 5. 3. 8 SCB 使用提示和技巧

保证软件使用对齐的 32 位字事务来访问所有的 SCB 寄存器。

附录1. 5. 4 系统定时器，SysTick

当系统定时器被允许时，定时器从当前值（SYST_CVR）开始递减计数到零，下一个时钟周期的边沿处再重新装载系统定时重载寄存器（SYST_RVR）的值，然后在后面的时钟周期下继续开始递减计数。当计数器跳变到零时，COUNTFLAG 状态位被设为 1。读 SYST_CSR 将 COUNTFLAG 位清零。

注意：SYST_CVR 的值在复位时不可知。使能系统定时器之前软件应该使该寄存器清零。这确保定时器启用时从 SYST_RVR 的值开始计数，而不是从一个任意值开始计数。

注意：如果 SYST_RVR 的值为 0，定时器在重载后将保持为当前值 0。这个机制可用于禁止定时器的某些特性，而不必通过定时器允许位来实现禁用功能。写 SYST_CVR 会将该寄存器和 COUNTFLAG 状态位都清零。写操作导致 SYST_RVR 的值在下一个定时周期被重载到 SYST_CVR，但不触发 SysTick 异常逻辑。读操作返回的是当前被访问寄存器的值。

注意：当处理器由于调试而被终止时，计数器不递减计数。

系统定时器寄存器有：

地址	名称	类型	复位值	描述
0xE000E010	SYST_CSR	RW	0x00000000	SysTick 控制和状态寄存器
0xE000E014	SYST_RVR	RW	不可知	SysTick 重装值寄存器
0xE000E018	SYST_CVR	RW	不可知	SysTick 当前值寄存器
0xE000E01C	SYST_CALIB	R	0x00000004	SysTick 校准值寄存器

附录表 1-41 系统定时寄存器小结

附录1. 5. 4. 1 SysTick 控制和状态寄存器

SYST_CSR 允许 SysTick 特性。有关寄存器的属性请见系统定时寄存器小结，该寄存器的位分配如下：

位域	名称	功能
[31:17]	-	保留
[16]	COUNTFLAG	如果从上次读这个寄存器之后定时器计数到 0，该位就返回 1
[15:3]	-	保留
[2]	CLKSOURCE	选择 SysTick 定时器的时钟源： 0=外部基准时钟 1=处理器时钟
[1]	TICKINT	允许 SysTick 异常请求： 0=计数到零不提交 SysTick 异常请求 1=计数到零提交 SysTick 异常请求
[0]	ENABLE	允许计数器： 0=计数器被禁止 1=计数器被允许

附录表 1-42 SYST_CSR 位分配

附录1. 5. 4. 2 SysTick 重装值寄存器

SYST_RVR 设定了加载到 SYST_CVR 的起始值。有关寄存器的属性请见**系统定时寄存器小结**。该寄存器的位分配为：

位域	名称	功能
[31:24]	-	保留
[23:0]	RELOAD	当计数器被允许且计数值到达 0 时加载到 SYST_CVR 的值，请见计算 RELOAD 值

附录表 1-43 SYST_RVR 位分配

计算 RELOAD 值

RELOAD 值可以是 0x00000001-0x00FFFFFF 范围内的任何值。用户可以将 RELOAD 的值设为 0，这不会产生任何影响，因为计数值从 1 变为 0 时 SysTick 异常请求和 COUNTFLAG 都被激活了。

如果要产生一个周期为 N 个处理器时钟周期的多次触发定时器，就可以将 RELOAD 值设为 N-1。例如，如果要求每隔 100 个时钟脉冲就触发一次 SysTick 中断，RELOAD 就被设为 99。

附录1. 5. 4. 3 SysTick 当前值寄存器

SYST_CVR 包含 SysTick 计数器的当前值。有关寄存器的属性请见 **系统定时寄存器小结** 。该寄存器的位分配如下：

位域	名称	功能
[31:24]	-	保留
[23:0]	CURRENT	读取时返回 SysTick 计数器的当前值。向这个域写入任何值都会将该域清零，同时将 SYST_CSR 的 COUNTFLAG 位清零

附录表 1-44 SYST_CVR 位分配

附录1. 5. 4. 4 SysTick 校准值寄存器

SYST_CALIB 寄存器指明了 SysTick 的校准特性。有关寄存器的属性请见 **系统定时寄存器小结** 。

该寄存器的位分配如下：

位域	名称	功能
[31]	NOREF	该位读出为 1。该位指明不提供独立的基准时钟
[30]	SKEW	该位读出为 1。由于 TENMS 不可知，因此，10ms 不精确计时的校准值不能确定。这会影响 SysTick 作为软件实时时钟的适用性
[29:24]	-	保留
[23:0]	TENMS	该位读出为 0。该域指明校准值不可知

附录表 1-45 SYST_CALIB 寄存器位分配

如果校准信息不可知，就通过处理器时钟或外部时钟的频率来计算所需的校准值。

附录1. 5. 4. 5 SysTick 使用提示和技巧

利用中断控制器时钟来更新 SysTick 计数器。如果这个时钟信号由于进入低功耗模式而终止，SysTick 计数器就停止计数。

确保软件使用字访问来访问 SysTick 寄存器。

如果在复位时没有定义 SysTick 计数器的重装值和当前值，正确的 SysTick 计数器初始化序列如下：

第 1 步：设置重装值

第 2 步：清除当前值

第 3 步：设置控制和状态寄存器

附录1. 6　Cortex-M0 指令汇总

操作	描述	汇编程序	周期
Move	8 位立即数	MOVS Rd,#<imm>	1
	（R0-R7）到（R0-R7）	MOVS Rd,Rm	1
	任意寄存器到任意寄存器	MOV Rd,Rm	1
	任意寄存器到 PC	MOVS PC,Rm	3
Add	3 位立即数	ADDS Rd,Rn,#<imm>	1
	R0-R7	ADDS Rd,Rn,Rm	1
	任意寄存器到任意寄存器	ADD Rd,Rn,Rm	1
	任意寄存器到 PC	ADD PC,PC,Rm	3
	8 位立即数	ADDS Rd,Rn,#<imm>	1
	带进位的	ADCS Rd,.Rd,Rm	1
	立即数到 SP	ADD SP,SP, #<imm>	1
	从 SP 形成地址	ADD Rd,SP, #<imm>	1
	从 PC 形成地址	ADR Rd<label>	1
Subtract	（R0-R7）和（R0-R7）	SUBS Rd,Rn,Rm	1
	3 位立即数	SUBS Rd,Rn, #<imm>	1
	8 位立即数	SUBS Rd,Rd, #<imm>	1
	带借位	SBCS Rd,Rn,Rm	1
	从 SP 减去立即数	SUB SP,SP, #<imm>	1
	相反数	RSBS Rd,Rn,#0	1
Multiply	乘法	MULS Rd,Rm,Rd	1
Compare	比较	CMP Rn,Rm	1
	负值	CMN Rn,Rm	1
	立即数	CMP Rn, #<imm>	1
Logical	与	ANDS Rd,Rd,Rm	1
	异或	EORS Rd,Rd,Rm	1
	或	ORRS Rd,Rd,Rm	1
	位清零	BICS Rd,Rd,Rm	1
	取反传送	MVNS Rd,Rm	1
	与测试	TST Rn,Rm	1
Shift	立即数逻辑左移	LSLS Rd,Rm,#<shift>	1
	寄存器逻辑左移	LSLS Rd,Rd,Rs	1
	立即数逻辑右移	LSRS Rd,Rm, #<shift>	1
	寄存器逻辑右移	LSRS Rd,Rd,Rs	1
	算术右移	ASRS Rd,Rm, #<shift>	1
	寄存器算术右移	ASRS Rd,Rd,Rs	1
Rotate	寄存器循环右移	RORS Rd,Rd,Rs	1
Load	字，直接偏移量	LDR Rd,[Rn, #<imm>]	2
	半字，直接偏移量	LDRH Rd,[Rn, #<imm>]	2

操作	描述	汇编程序	周期
	字节，直接偏移量	LDRB Rd,[Rn, #<imm>]	2
	字，寄存器偏移量	LDR Rd,[Rn,Rm]	2
	半字，寄存器偏移量	LDRH Rd,[Rn,Rm]	2
	有符号的半字，寄存器偏移量	LDRSH Rd,[Rn,Rm]	2
	字节，寄存器偏移量	LDRB Rd,[Rn,Rm]	2
	有符号的字节，寄存器偏移量	LDRSB Rd,[Rn,Rm]	2
	PC 相对值	LDR Rd,<label>	2
	SP 相对值	LDR Rd,[SP,#<imm>	2
	乘法，不带基地址	LDM Rn!,{<loreglist>}	1+N[1]
	乘法，带基地址	LDM Rn,{<loreglist>}	1+N[1]
Store	字，直接偏移量	STR Rd,[Rn, #<imm>]	2
	半字，直接偏移量	STRH Rd,[Rn, #<imm>]	2
	字节，直接偏移量	STRB Rd,[Rn, #<imm>]	2
	字，寄存器偏移量	STR Rd,[Rn,Rm]	2
	半字，寄存器偏移量	STRH Rd,[Rn,Rm]	2
	字节，寄存器偏移量	STRB Rd,[Rn,Rm]	2
	SP 相对值	STR Rd,[SP,#<imm>	2
	乘法	STM Rn!, {<loreglist>}	1+N[1]
Push	进栈	PUSH{<loreglist>}	1+N[1]
	带链接寄存器的进栈	PUSH{<loreglist>,LR}	1+N[1]
Pop	出栈	POP{<loreglist>}	1+N[1]
	出栈和返回	POP{<loreglist>,PC}	4+N[2]
Branch	有条件的	B<cc><label>	1 or 3 [3]
	无条件的	B<label>	3
	带链接的	BL<label>	4
	带交换的	BX Rm	3
	带链接和交换	BLX Rm	3
Extend	有符号的半字到字	SXTH Rd,Rm	1
	有符号的字节到字	SXTB Rd,Rm	1
	无符号半字	UXTH Rd,Rm	1
	无符号字节	UXTB Rd,Rm	1
Reverse	字中的字节	REV Rd,Rm	1
	两个半字中的字节	REV 16 Rd,Rm	1
	有符号的底端半字	REVSH Rd,Rm	1
State change	超级用户调用	SVC<imm>	-[4]
	禁止中断	CPSID i	1
	允许中断	CPSIE i	1
	读特殊寄存器	MRS Rd,<specreg>	4
	写特殊寄存器	MSR <specreg>,Rn	4

操作	描述	汇编程序	周期
Hint	发送事件	SEV	1
	等待事件	WFE	2[5]
	等待中断	WFI	2[5]
	放弃	YIELD[6]	1
	空操作	NOP	1
MOVBarriers	同步指令	ISB	4
	数据存储	DMB	4
	数据同步	DSB	4

附录表 1-46 Cortex M0 指令汇总

注[1]：N 为元素的个数
注[2]：N 是堆栈出栈列表元素的个数，包括 PC 的数量并假设加载或存储不会产生 HardFault 异常
注[3]：如果采取就为 3，不采取就为 1
注[4]：周期数取决于核和调试配置
注[5]：不包括等待事件或中断花费的时间
注[6]：作为 NOP 执行

修订历史

版本	修订日期	修订内容
V1.0	2022-08-01	初版发布