32 位 MCU
ES32M0502

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

2025-03-19

目录

图目录

表目录

文件约定

下表解释了文档中经常使用的缩写。

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(无)	保留位，必须保持复位值

第1章　系统和内存概述

1. 1　概述

ES32M0502 微控制器是一系列低功耗微控制器，集成高性能 ARM Cortex-M0 32 位 RISC 内核。芯片最高工作频率为 72MHz，以及最大 128Kbytes Flash 与 8Kbytes SRAM。提供广泛且有效的功能模块，以及符合标准的通讯接口，包含 1 个 I2C，1 个 SPI，2 个 UART，1 个 CAN2.0，2 个高级 16 位定时器，2 个通用 32 位定时器，2 个通用 16 位定时器，1 个基本 16 位定时器，1 个独立看门狗，1 个窗口看门狗。

ES32M0502 微控制器，具备 2 个 ADC ，2 个比较器与 4 个 OPAMP 运算放大器，可配置成差分/单端 PGA，支持 x1 ~ x16 倍率，另外提供运算加速单元，包含 CALC 除法/开根号与 SVPWM 加速单元 Clarke/Park。

根据以上特性，ES32M0502 微控制器适用于广泛的应用，如马达控制、白色家电与智能家电。

本章介绍了 ES32M0502 的特点，其系统和内存结构：

- ES32M0502 系统体系结构
- ES32M0502 系统特点
- ES32M0502 内存映射

外设		ES32M0502LQ	ES32M0502LK
Flash (Kbytes)		128	
SRAM(KBytes)		8	
GPIO		44	28
DMA		6 channels	
CMP		2	
ADC		2（12bit，19 channels）	
OPA		4（可配置差分 PGA）	
CALC 运算加速器		32 位除法/平方根	
SVA 运算加速器		SVPWM/Clarke/Park	
定时器	AD16C6T	2	
	GP32C4T	2	
	GP16C2T	2	
	BS16T	1	
	WWDT	1	
	IWDT	1	
通信接口	I2C	1	
	SPI	1	
	UART	2	
	CAN2.0	1	
CPU 操作频率		72 MHz	
芯片工作电压范围（V_{DDH}/V_{DDA}）		2.4V - 5.5V	
封装类型		LQFP48	LQFP32

表 1-1　设备功能和外围设备数量

1. 2 结构图

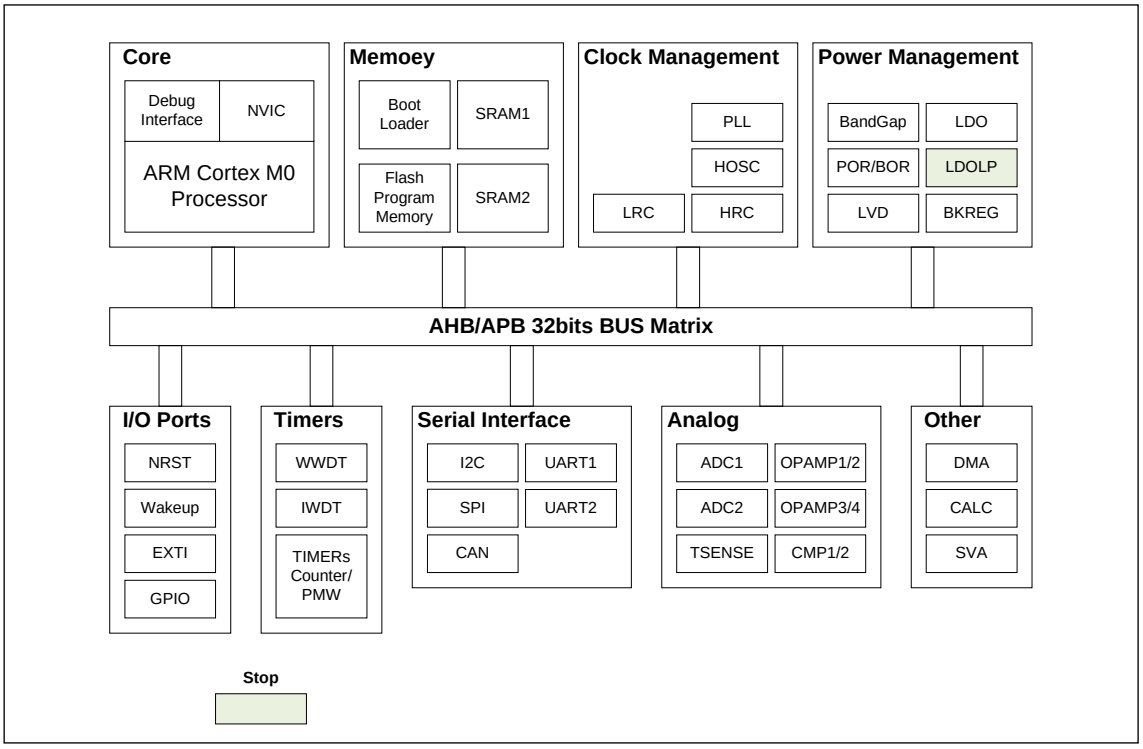


图 1-1 ES32M0502 系统框图

1. 3 功能描述

本节概述 ES32M0502 微控制器特点。关于这些特征的进一步信息在它们各自的章节中描述。

特征	描述
处理器	ARM Cortex-M0处理器
性能	56 DMIPS @72MHz (Dhrystone 2.1)
存储单元	
闪存	128K bytes Flash
系统静态随机存取存储器	8K bytes SRAM
备用缓存器	16 bytes
系统	
电源管理	上电/掉电/欠压复位(POR/PDR/BOR) 可编程低电压侦测(LVD) 稳压器
低功耗模式	两种低功耗模式: 睡眠模式(SLEEP) 停止模式(STOP)
时钟控制	4-32MHz 晶振(HOSC) 内部16MHz RC振荡器(HRC) 内部32KHz RC振荡器(LRC) PLL锁相环电路，最高倍频至72MHz，支持分数频设定
计时器	
高级控制定时器: **AD16C6T1，AD16C6T2** **通用定时器A型:GP32C4T1，GP32C4T2** **通用定时器C型:GP16C2T1，GP16C2T2** **基本定时器:BS16T**	**高级控制定时器：** 通过同步或事件链接的定时器链接功能一起工作 可在调试模式下冻结计数器 16位自动重载上/下数计数器和16位预分频器 可编程插入死区的互补PWM输出 4捕获或比较通道和4互补输出 支持刹车功能 支持增量（正交）编码及霍尔电路进行定位 **通用定时器A型：** 通过同步或事件链接的定时器链接功能一起工作 可在调试模式下冻结计数器 32位自动重载上/下数计数器和16位预分频器 4捕获或比较通道 支持增量（正交）编码及霍尔电路进行定位 **通用定时器C型：** 通过同步或事件链接的定时器链接功能一起工作

特征	描述
	可在调试模式下冻结计数器 16位自动重载上数计数器和16位预分频器 可编程插入死区的互补PWM输出 2捕获或比较通道和1互补输出 支持刹车功能 **基本定时器：** 16位自动重载上数计数器和16位预分频器 可在调试模式下冻结计数器
窗口看门狗(WWDT)	一个WWDT 7位自由运行计数器，可编程超时间隔
独立看门狗(IWDT)	一个IWDT 12位自由运行下数计数器和8位预分频器
输入输出端口	
通用输入输出（GPIO)	3个物理GPIO块 支持外部中断(边沿或电平触发) 去抖动功能
扩展中断和事件控制器（EXTI)	生成多达20个事件/中断请求 每一个可以独立配置以选择触发事件(上升沿、下降沿，两者皆有)，并且可以独立屏蔽 最多44个GPIOS可以连接到16个外部中断
模拟控制单元	
内部分压电阻(VRES)	2组内部分压电阻，基于模拟电源(VDDA)进行电阻分压 分别连接至比较器(CMP)、运算放大器(OPAMP)及模拟数字转换器(ADC)
比较器(CMP)	2个单独的模拟比较器 CMP1与CMP2可以配置为一个监控窗口比较器 支持消隐源比较器输出 支持触发Timer或通过EXTI产生中断
模拟数字转换器(ADC)	2个独立的模拟数字转换器，具有 12 个外部输入讯号和 3 个内部讯号 分辨率支持12位/10位/8位/6位 支持单次、连续或不连续转换模式
运算放大器(OPAMP)	4个运算放大器
通信接口	
通用异步收发器(UART)	可程序设置波特率发生器 自动波特率检测 自动硬件流量控制

特征	描述
	可程序设置(CTSn，RTSn)触发电平
内部集成电路总线(I2C)	支持多主机模式 支持标准模式(Sm)、快速模式(Fm)与极快速模式(Fm+) 支持SMBus(系统管理总线)与PMBus(电源管理总线)
串行外设接口(SPI)	三线全双工同步传输 双线(双向数据线)的半双工同步传输 双线(单向数据线)的单工同步传输 8位至16位传输帧格式选择 支持SPI TI模式 主从操作 提供TX和RX FIFO，深度为4
基本扩展控制器局域网络(bxCAN)	支持2.0A和2.0B Active版本的CAN协议规范 2.0A版本的协议规范支持11位标准标识符 2.0B Active版本的协议规范支持11位标准标识符和29位扩展标识符 支持3个优先级可配置的发送邮箱，2个可以存储三级邮箱深度的接收FIFO
加速器	
运算加速器(CALC)	支持最大32位无符号数平方根运算 支持最大32位有符号数或无符号数除法运算 支持除零警示标志 支持根据运算数值大小，运算时间自动调整为2~17个 HCLK 时钟周期 支持固定运算时间(17 HCLK)
空间向量运算加速器(SVA)	支持Clarke 正逆转换 支持Park 正逆转换 支持SVPWM 支持输入输出使用有符号16位定点数表示
无限脉冲响应滤波器(IIR)	供一个滤波器的运算加速器 使用双二阶滤波器架构 有两级滤波器，可配置采用一级或两级 支持系数由用户配置
调试配置	
串行线调试 (SWD)	提供了一个ARM SWD接口，以允许串行线调试工具连接到MCU。

表 1-2　ES32M0502 微控制器特性

1. 4 内存映射

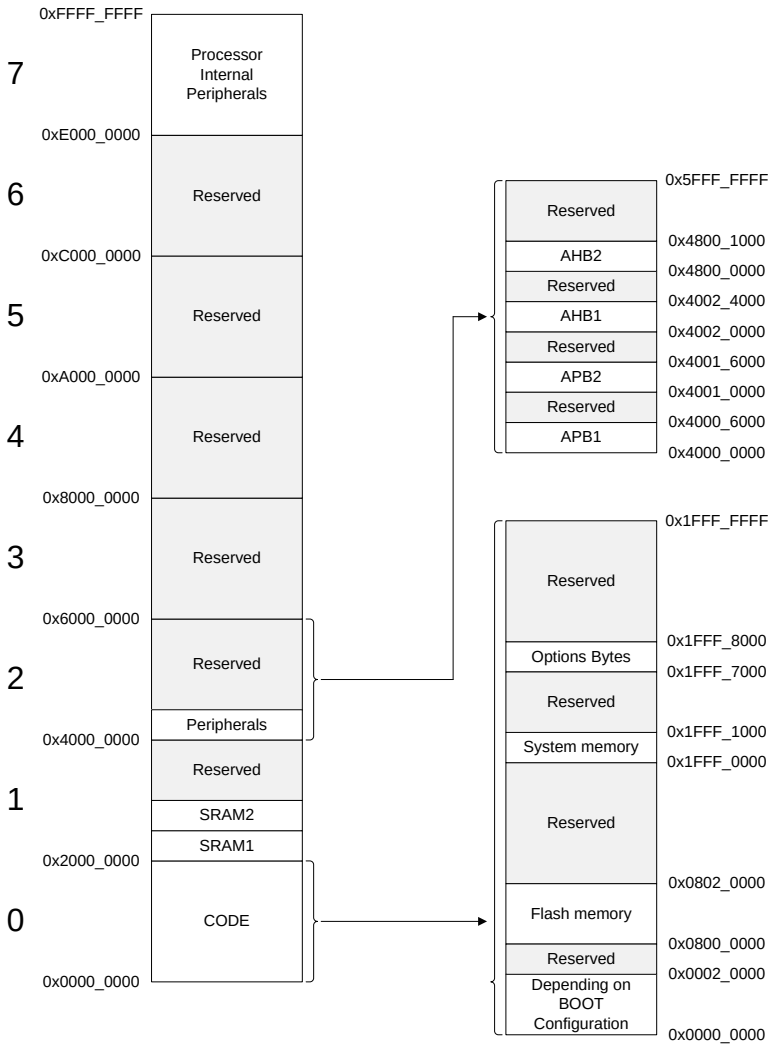


图 1-2 内存映射

边界地址		大小 (Byte)	外设	总线
起点	终点			
0xE000_0000	0xE00F_FFFF	1M	Cortex®-M0 内部外设	
0x4800_2400	0x5FFF_FFFF	~384M	Reserved	AHB2
0x4800_2000	0x4800_23FF	1K	ADC	
0x4800_1C00	0x4800_1FFF	1K	Reserved	
0x4800_1800	0x4800_1BFF	1K	Reserved	
0x4800_1400	0x4800_17FF	1K	Reserved	
0x4800_1000	0x4800_13FF	1K	Reserved	
0x4800_1C00	0x4800_0FFF	1K	Reserved	
0x4800_0800	0x4800_0BFF	1K	GPIO C	
0x4800_0400	0x4800_07FF	1K	GPIO B	
0x4800_0000	0x4800_03FF	1K	GPIO A	
0x4002_4000	0x47FF_FFFF	~128M	Reserved	AHB1
0x4002_3C00	0x4002_3FFF	1K	CALC	
0x4002_3800	0x4002_3BFF	1K	Reserved	
0x4002_3400	0x4002_37FF	1K	SVA	
0x4002 3000	0x4002_33FF	1K	IIR	
0x4002_2C00	0x4002_2FFF	1K	Reserved	
0x4002_2800	0x4002_2BFF	1K	Reserved	
0x4002_2400	0x4002_27FF	1K	Reserved	
0x4002_2000	0x4002_23FF	1K	FLASH	
0x4002_1C00	0x4002_1FFF	1K	Reserved	
0x4002_1800	0x4002_1BFF	1K	Reserved	
0x4002_1400	0x4002_17FF	1K	SYSCFG+OPT	
0x4002_1000	0x4002_13FF	1K	RCU	
0x4002_0C00	0x4002_0FFF	1K	Reserved	
0x4002_0800	0x4002_0BFF	1K	DMA Mux	
0x4002_0400	0x4002_07FF	1K	Reserved	
0x4002_0000	0x4002_03FF	1K	DMA	
0x4001_8000	0x4001_FFFF	32K	Reserved	APB2
0x4001_6000	0x4001_7FFF	8K	VRES	
0x4001_5C00	0x4001_5FFF	1K	CMP	
0x4001_5800	0x4001_5BFF	1K	DBGMCU	
0x4001_5400	0x4001_57FF	1K	Reserved	
0x4001_5000	0x4001_53FF	1K	Reserved	
0x4001_4C00	0x4001_4FFF	1K	Reserved	
0x4001_4800	0x4001_4BFF	1K	Reserved	
0x4001_4400	0x4001_47FF	1K	GP16C2T2	
0x4001_4000	0x4001_43FF	1K	GP16C2T1	
0x4001_3C00	0x4001_3FFF	1K	Reserved	

边界地址		大小 (Byte)	外设	总线
起点	终点			
0x4001_3800	0x4001_3BFF	1K	UART1	
0x4001_3400	0x4001_37FF	1K	AD16C6T2	
0x4001_3000	0x4001_33FF	1K	SPI1	
0x4001_2C00	0x4001_2FFF	1K	AD16C6T1	
0x4001_2800	0x4001_2BFF	1K	Reserved	
0x4001_2400	0x4001_27FF	1K	Reserved	
0x4001_2000	0x4001_23FF	1K	Reserved	
0x4001_1C00	0x4001_1FFF	1K	Reserved	
0x4001_1800	0x4001_1BFF	1K	Reserved	
0x4001_1400	0x4001_17FF	1K	OPAMP	
0x4001_1000	0x4001_13FF	1K	Reserved	
0x4001_0C00	0x4001_0FFF	1K	Reserved	
0x4001_0800	0x4001_0BFF	1K	Reserved	
0x4001_0400	0x4001_07FF	1K	EXTI	
0x4001_0000	0x4001_03FF	1K	Reserved	
0x4000_8000	0x4000_FFFF	32K	Reserved	APB1
0x4000_6800	0x4000_7FFF	6K	Reserved	
0x4000_6400	0x4000_67FF	1K	CAN1	
0x4000_6000	0x4000_63FF	1K	Reserved	
0x4000_5C00	0x4000_5FFF	1K	Reserved	
0x4000_5800	0x4000_5BFF	1K	Reserved	
0x4000_5400	0x4000_57FF	1K	I2C1	
0x4000_5000	0x4000_53FF	1K	Reserved	
0x4000_4C00	0x4000_4FFF	1K	Reserved	
0x4000_4800	0x4000_4BFF	1K	Reserved	
0x4000_4400	0x4000_47FF	1K	UART2	
0x4000_4000	0x4000_43FF	1K	Reserved	
0x4000_3C00	0x4000_3FFF	1K	Reserved	
0x4000_3800	0x4000_3BFF	1K	Reserved	
0x4000_3400	0x4000_37FF	1K	Reserved	
0x4000_3000	0x4000_33FF	1K	IWDT	
0x4000_2C00	0x4000_2FFF	1K	WWDT	
0x4000_2800	0x4000_2BFF	1K	Reserved	
0x4000_2400	0x4000_27FF	1K	Reserved	
0x4000_2000	0x4000_23FF	1K	Reserved	
0x4000_1C00	0x4000_1FFF	1K	Reserved	
0x4000_1800	0x4000_1BFF	1K	Reserved	
0x4000_1400	0x4000_17FF	1K	Reserved	
0x4000_1000	0x4000_13FF	1K	BS16T	

边界地址		大小 (Byte)	外设	总线
起点	终点			
0x4000_0C00	0x4000_0FFF	1K	Reserved	
0x4000_0800	0x4000_0BFF	1K	Reserved	
0x4000_0400	0x4000_07FF	1K	GP32C4T2	
0x4000_0000	0x4000_03FF	1K	GP32C4T1	
0x2000_2000	0x3FFF_FFFF	~512M	Reserved	
0x2000_1000	0x2000_1FFF	4K	SRAM2	
0x2000_0000	0x2000_0FFF	4K	SRAM1	
0x1FFF_F000	0x1FFF_FFFF	4K	Reserved	
0x1FFF_EC00	0x1FFF_EFFF	1K	Option3 bytes	
0x1FFF_E800	0x1FFF_EBFF	1K	Option2 bytes	
0x1FFF_E000	0x1FFF_E7FF	2K	Option1 bytes(Protect Setting)	
0x1000_1000	0x1FFF_DFFF	~255M	Reserved	
0x1000_0000	0x1000_0FFF	4K	SROM1 (BootROM)	
0x0802_0000	0x0FFF_FFFF	~128M	Reserved	
0x0800_0000	0x0801_FFFF	128K	Main Flash	
0x0002_0000	0x07FF_FFFF	~128M	Reserved	
0x0000_0000	0x0001_FFFF	128K	取决于开机存储配置	

表 1-3　外设寄存器边界地址

第2章 ARM ® Cortex™-M0 Core

2. 1 概述

ARMCortex™-M0 处理器是最小型和最节能的 ARM 处理器。它满足了越来越低成本应用的需求，同时增加了连通性。M0 处理器是一个可配置的多级 32 位 RISC 处理器。

2. 2 特性

在 ES32M0502 中，该处理器配置以下特征：

- 内置嵌套向量中断控制器（NVIC）- 32 外部中断
- 小端序
- 集成系统定时器 - SysTick
- 支持暂停调试
- 快速乘法器
- 支持串行线调试（SWD）连接

本章提供以下处理器外围设备的基本信息：

- CPU 系统定时器控制（SysTick）
- CPU 嵌套向量中断控制器（NVIC）
- CPU 系统控制

详情请参阅：

- ARM Cortex™-M0 技术参考手册
- ARM v6-M 结构参考手册

2. 3 功能描述

2. 3. 1 CPU 系统定时器控制寄存器(SYST)

Cortex™-M0 包括一个集成的系统定时器 - **SysTick**，它提供了一个简单的、24 位的写入清零、递减、倒数到 0 重载计数器，并具有灵活的控制机制。计数器可作为实时操作系统(RTOS)时钟定时器或简单计数器使用。

当系统定时器被启用时，它开始从 SysTick 当前值寄存器 (**SYST_CVR**) 中的值倒数到 0，并在下一个时钟周期中重新加载(包装) SysTick 重载值寄存器 (**SYST_RVR**) 中的值，然后在随后的时钟上递减。一旦计数器转换为 0，设置 COUNTFLAG 状态位。COUNTFLAG 位在读取时清除。

复位时，系统的 **SYST_CVR** 值未知。在启用该功能之前，软件应该写入寄存器以将其清除为零。这确保了定时器在启用时将从 **SYST_RVR** 值计数而不是任意值。

如果 **SYST_RVR** 为 0，则定时器将用该值重新加载后保持当前值为 0。该机制可用于独立于定时器允许位禁用该特征。

2. 3. 2 嵌套向量中断控制器(NVIC)

2. 3. 2. 1 NVIC主要特征

- 32 个可屏蔽中断通道 (不包括十六个 Cortex®-M0 的中断)
- 可编程优先级(使用了 2 位中断优先级)
- 低延迟异常和中断处理
- 电源管理控制
- 系统控制寄存器的实现

NVIC 与处理器内核接口紧密配合，可以实现低延迟的中断处理和和高效地处理晚到的中断。包括内核异常在内的所有中断都由 NVIC 管理。

2. 3. 2. 2 SysTick校准值寄存器

SysTick 校准值被设置为 5000，提供了 10 ms 的基准时间，SysTick 时钟被设置为 500 kHz(默认 f_{HCLK}/8 = 4 MHz/8)。

2. 3. 2. 3 中断和异常向量

编号	优先级	名称	描述	地址
0	-		保留	0x0000 0000
1	-3	Reset	复位	0x0000 0004
2	-2	NMI_Handler	不可屏蔽中断，时钟安全事件	0x0000 0008
3	-1	HardFault_Handler	所有类型的故障	0x0000 000C
4~10	-		保留	
11	可设置	SVC_Handler	通过 SWI 指令调用的系统服务	0x0000 002C
12~13	-		保留	
14	可设置	PendSV_Handler	可挂起的系统服务	0x0000 0038
15	可设置	SysTick_Handler	系统定时器中断	0x0000 003C
16	可设置	WWDT	WWDT 全局中断	0x0000 0040
17	可设置	LVD	LVD 通过 EXTI 线 20 检测中断	0x0000 0044
18	可设置	CM0IK	保留，仅用于内部测试	0x0000 0048
19	可设置	Low Power Wakeup	所有低功耗唤醒通过 EXTI 线 21 检测中断	0x0000 004C
20	可设置	RCU	RCU 全局中断	0x0000 0050
21	可设置	EXTI<1:0>	EXTI 线 0 至 1 中断	0x0000 0054
22	可设置	EXTI<3:2>	EXTI 线 2 至 3 中断	0x0000 0058
23	可设置	EXTI<15:4>	EXTI 线 4 至 15 中断	0x0000 005C
24	可设置	DMA1_CH0	DMA 通道 0 中断	0x0000 0060
25	可设置	DMA1_CH1	DMA 通道 1 中断	0x0000 0064
26	可设置	DMA1_CH2	DMA 通道 2 中断	0x0000 0068
27	可设置	DMA1_CH3	DMA 通道 3 中断	0x0000 006C
28	可设置	DMA1_CH4	DMA 通道 4 中断	0x0000 0070
29	可设置	DMA1_CH5	DMA 通道 5 中断	0x0000 0074
30	可设置	ADC	ADC 全局中断	0x0000 0078
31	可设置	SVA	SVA 全局中断	0x0000 007C
32	可设置	CAN_TX	CAN 发送中断	0x0000 0080
33	可设置	CAN_RXFIFO0	CAN 接收 FIFO0 中断	0x0000 0084
34	可设置	CAN_RXFIFO1	CAN 接收 FIFO1 中断	0x0000 0088
35	可设置	CAN	CAN 状态改变和错误中断	0x0000 008C
36	可设置	AD16C6T1	AD16C6T1 全局中断	0x0000 0090
37	可设置	GP16C2T1	GP16C2T1 全局中断	0x0000 0094
38	可设置	GP16C2T2	GP16C2T2 全局中断	0x0000 0098
39	可设置	GP32C4T1	GP32C4T1 全局中断	0x0000 009C
40	可设置	GP32C4T2	GP32C4T2 全局中断	0x0000 00A0
41	可设置	I2C	I2C 全局中断	0x0000 00A4
42	可设置	SPI	SPI 全局中断	0x0000 00A8
43	可设置	UART1	UART1 全局中断	0x0000 00AC
44	可设置	UART2	UART2 全局中断	0x0000 00B0
45	可设置	AD16C6T2	AD16C6T2 全局中断	0x0000 00B4
46	可设置	BS16T	BS16T 全局中断	0x0000 00B8
47	可设置	CMP	CMP 全局中断	0x0000 00BC

2. 3. 3 CPU 系统控制

Cortex™-M0 的状态和操作模式控制由 CPU 系统控制寄存器管理，包括 CPUID 在内，可以通过这些系统控制寄存器来控制 Cortex™-M0 中断优先级和 Coretex™-M0 电源管理。

2. 4 特殊功能寄存器

2. 4. 1 寄存器列表

SYST 寄存器列表			
名称	偏移地址	类型	描述
SYST_CSR	0010_H	R/W	SysTick 控制和状态寄存器
SYST_RVR	0014_H	R/W	SysTick 重载值寄存器
SYST_CVR	0018_H	R/W	SysTick 当前值寄存器

NVIC 寄存器列表			
名称	偏移地址	类型	描述
NVIC_ISER	000H	R/W	NVIC IRQ 设置使能控制寄存器
NVIC_ICER	080H	R/W	NVIC IRQ 清除使能控制寄存器
NVIC_ISPR	100H	R/W	NVIC IRQ 设置挂起的控制寄存器
NVIC_ICPR	180H	R/W	NVIC IRQ 清除挂起的控制寄存器
NVIC_IPR0	300H	R/W	NVIC IRQ0 - IRQ3 优先级控制寄存器
NVIC_IPR1	304H	R/W	NVIC IRQ4 - IRQ7 优先级控制寄存器
NVIC_IPR2	308H	R/W	NVIC IRQ8 - IRQ11 优先级控制寄存器
NVIC_IPR3	30CH	R/W	NVIC IRQ12 - IRQ15 优先级控制寄存器
NVIC_IPR4	310H	R/W	NVIC IRQ16 - IRQ19 优先级控制寄存器
NVIC_IPR5	314H	R/W	NVIC IRQ20 - IRQ23 优先级控制寄存器
NVIC_IPR6	318H	R/W	NVIC IRQ24 - IRQ27 优先级控制寄存器
NVIC_IPR7	31CH	R/W	NVIC IRQ28 - IRQ31 优先级控制寄存器

SYS 寄存器列表			
名称	偏移地址	类型	描述
SYS_CPUID	000H	R/W	CPU ID 寄存器
SYS_ICSR	004H	R/W	中断控制与状态寄存器
SYS_AIRCR	00CH	R/W	应用中断和复位控制寄存器
SYS_SCR	010H	R/W	系统控制寄存器
SYS_SHPR2	01CH	R/W	系统处理程序优先级寄存器 2
SYS_SHPR3	020H	R/W	系统处理程序优先级寄存器 3

2. 4. 2 寄存器描述

2. 4. 2. 1 SysTick控制和状态寄存器(SYST_CSR)

SysTick 控制和状态寄存器 (SYST_CSR)																												
偏移地址:0x10																												
复位值:0x0000 0000																												
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
															COUNT														CLKSRC	TICKIE	ENABLE

—	Bits 31-17	—	—
COUNT	Bit 16	R	**计数标示** 0:自上次读取该位以来，SysTick 定时器没有计算到 0。 1:自上次读取该位以来，SysTick 定时器已计算为 0。 COUNT 在读取或写入当前值寄存器时被清除
—	Bits 15-3	—	—
CLKSRC	Bit 2	R/W	**System Tick 时钟源选择** 0:时钟源是(任意的)外部参考时钟。 1:核心时钟用于 SysTick.
TICKIE	Bit 1	R/W	**System Tick 中断使能** 0:倒数定时到 0 时产生 SysTick 异常请求. 软件可以使用 COUNTFLAG 来确定是否发生了 0 的计数。 1:倒数定时到 0 时不会产生 SysTick 异常请求。通过写入软件清除 SysTick 当前值寄存器将不会产生 SysTick 异常请求
ENABLE	Bit 0	R/W	**System 计数器使能** 0:计数器已禁用 1:计数器以多种方式执行

2. 4. 2. 2 SysTick 重载值寄存器 (SYST_RVR)

SysTick 重载值寄存器 (SYST_RVR)																															
偏移地址:0x14																															
复位值:0x0000 0000																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								RELOAD<23:0>																							

—	Bits 31-24	—	—
RELOAD	Bits 23-0	R/W	**重载值** 当计数器达到 0 时，加载到 SysTick 当前值寄存器(SYST_CVR)寄存器中。

2. 4. 2. 3 SysTick当前值寄存器 (SYST_CVR)

SysTick 当前值寄存器 (SYST_CVR)																															
偏移地址:0x18																															
复位值:0x0000 0000																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								CURRENT <23:0>																							

—	Bits 31-24	—	—
CURRENT	Bits 23-0	R/W	**System Tick 当前计数值** 当前计数器值，是计数器在采样时的值。计数器不提供读修改写保护。寄存器是写清楚的。任何值的软件写入都会将寄存器清除为 0。

2. 4. 2. 4 NVIC IRQ 设置使能控制寄存器 (NVIC_ISER)

NVIC IRQ 设置使能控制寄存器 (NVIC_ISER)																															
偏移地址:0x00																															
复位值:0x0000 0000																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SETENA<31:0>																															

SETENA	Bits 31-0	R/W	**中断使能** 启用一个或多个中断。每个位代表从 IRQ0 到 IRQ31 中断号码 0:在读取时，指示中断被禁用；在写入时，没有影响 1:在读取时，指示中断已启用；在写入时，启用中断

2. 4. 2. 5 NVIC IRQ 清除使能控制寄存器 (NVIC_ICER)

NVIC IRQ 清除使能控制寄存器 (NVIC_ICER)																															
偏移地址:0x80																															
复位值:0x0000 0000																															
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
CLRENA<31:0>																															

CLRENA	Bits 31-0	R/W	**中断禁用** 禁用一个或多个中断。每个位代表从 IRQ0 到 IRQ31 中断号码 0:在读取时，指示中断被禁用；在写入时，没有影响 1:在读取时，指示中断已启用；在写入时，禁用中断

2. 4. 2. 6 NVIC IRQ 设置挂起控制寄存器 (NVIC_ISPR)

NVIC IRQ 设置挂起控制寄存器 (NVIC_ISPR)																															
偏移地址:0x100																															
复位值:0x0000 0000																															
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
SETPEND<31:0>																															

SETPEND	Bits 31-0	R/W	**设置中断挂起** 禁用一个或多个中断。每个位代表从 IRQ0 到 IRQ31 中断号码 0:在读取时，指示中断不被停止；在写入时，没有影响。 1:在读取时，指示中断被停止；在写入时，相应的中断被设置为停止，即使它被禁用。

2. 4. 2. 7 NVIC IRQ 清除挂起控制寄存器 (NVIC_ICPR)

NVIC IRQ 清除挂起控制寄存器 (NVIC_ICPR)																															
偏移地址:0x180																															
复位值:0x0000 0000																															
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
CLRPEND<31:0>																															

CLRPEND	Bits 31-0	R/W	**设置中断挂起** 禁用一个或多个中断。每个位代表从 IRQ0 到 IRQ31 中断号码 0:在读取时，指示中断不被停止；在写入时，没有影响。 1:在读取时，指示中断被停止；在写入时，写入 1 以清除停止状态，以便相应的中断不再停止。

2. 4. 2. 8 NVIC IRQ0 - IRQ3 优先级控制寄存器 (NVIC_IPR0)

NVIC IRQ0 - IRQ3 优先级控制寄存器 (NVIC_IPR0)																													
偏移地址:0x300																													
复位值:0x0000 0000																													
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
PRI3<1:0>								PRI2<1:0>								PRI1<1:0>								PRI0<1:0>							

PRI3	Bits 31-30	R/W	**IRQ3** 优先级
—	Bits 29-24	—	—
PRI2	Bits 23-22	R/W	**IRQ2** 优先级
—	Bits 21-16	—	—
PRI1	Bits 15-14	R/W	**IRQ1** 优先级
—	Bits 13-8	—	—
PRI0	Bits 7-6	R/W	**IRQ0** 优先级
—	Bits 5-0	—	—

2. 4. 2. 9 NVIC IRQ4 – IRQ7 优先级控制寄存器 (NVIC_IPR1)

NVIC IRQ4 – IRQ7 优先级控制寄存器 (NVIC_IPR1)																													
偏移地址:0x304																													
复位值:0x0000 0000																													
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
PRI7<1:0>								PRI6<1:0>								PRI5<1:0>								PRI4<1:0>							

PRI7	Bits 31-30	R/W	**IRQ7** 优先级
—	Bits 29-24	—	—
PRI6	Bits 23-22	R/W	**IRQ6** 优先级
—	Bits 21-16	—	—
PRI5	Bits 15-14	R/W	**IRQ5** 优先级
—	Bits 13-8	—	—
PRI4	Bits 7-6	R/W	**IRQ4** 优先级
—	Bits 5-0	—	—

2. 4. 2. 10 NVIC IRQ8 – IRQ11 优先级控制寄存器 (NVIC_IPR2)

NVIC IRQ8 – IRQ11 优先级控制寄存器 (NVIC_IPR2)																													
偏移地址:0x308H																													
复位值:0x0000 0000																													
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
PRI11<1:0>		—	—	—	—	—	—	PRI10<1:0>		—	—	—	—	—	—	PRI9<1:0>		—	—	—	—	—	—	PRI8<1:0>		—	—	—	—	—	—

PRI11	Bits 31-30	R/W	**IRQ11** 优先级
—	Bits 29-24	—	—
PRI10	Bits 23-22	R/W	**IRQ10** 优先级
—	Bits 21-16	—	—
PRI9	Bits 15-14	R/W	**IRQ9** 优先级
—	Bits 13-8	—	—
PRI8	Bits 7-6	R/W	**IRQ8** 优先级
—	Bits 5-0	—	—

2. 4. 2. 11 NVIC IRQ12 – IRQ15 优先级控制寄存器 (NVIC_IPR3)

NVIC IRQ12 – IRQ15 优先级控制寄存器 (NVIC_IPR3)																													
偏移地址:0x30C																													
复位值:0x0000 0000																													
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
PRI15<1:0>		—	—	—	—	—	—	PRI14<1:0>		—	—	—	—	—	—	PRI13<1:0>		—	—	—	—	—	—	PRI12<1:0>		—	—	—	—	—	—

PRI15	Bits 31-30	R/W	**IRQ15** 优先级
—	Bits 29-24	—	—
PRI14	Bits 23-22	R/W	**IRQ14** 优先级
—	Bits 21-16	—	—
PRI13	Bits 15-14	R/W	**IRQ13** 优先级
—	Bits 13-8	—	—
PRI12	Bits 7-6	R/W	**IRQ12** 优先级
—	Bits 5-0	—	—

2. 4. 2. 12 NVIC IRQ16 – IRQ19 优先级控制寄存器 (NVIC_IPR4)

NVIC IRQ16 – IRQ19 优先级控制寄存器 (NVIC_IPR4)																													
偏移地址:0x310																													
复位值:0x0000 0000																													
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
PRI19<1:0>								PRI18<1:0>								PRI7<1:0>								PRI6<1:0>							

PRI19	Bits 31-30	R/W	**IRQ19** 优先级
—	Bits 29-24	—	—
PRI18	Bits 23-22	R/W	**IRQ18** 优先级
—	Bits 21-16	—	—
PRI17	Bits 15-14	R/W	**IRQ17** 优先级
—	Bits 13-8	—	—
PRI16	Bits 7-6	R/W	**IRQ16** 优先级
—	Bits 5-0	—	—

2. 4. 2. 13 NVIC IRQ20 – IRQ23 优先级控制寄存器 (NVIC_IPR5)

NVIC IRQ20 – IRQ23 优先级控制寄存器 (NVIC_IPR3)																													
偏移地址:0x314																													
复位值:0x0000 0000																													
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
PRI23<1:0>								PRI22<1:0>								PRI21<1:0>								PRI20<1:0>							

PRI23	Bits 31-30	R/W	**IRQ23** 优先级
—	Bits 29-24	—	—
PRI22	Bits 23-22	R/W	**IRQ22** 优先级
—	Bits 21-16	—	—
PRI21	Bits 15-14	R/W	**IRQ21** 优先级
—	Bits 13-8	—	—
PRI20	Bits 7-6	R/W	**IRQ20** 优先级
—	Bits 5-0	—	—

2. 4. 2. 14 NVIC IRQ24 – IRQ27 优先级控制寄存器 (NVIC_IPR6)

NVIC IRQ24 – IRQ27 优先级控制寄存器 (NVIC_IPR6)																													
偏移地址:0x318																													
复位值:0x0000 0000																													
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
PRI27<1:0>								PRI26<1:0>								PRI25<1:0>								PRI24<1:0>							

PRI27	Bits 31-30	R/W	**IRQ27** 优先级
—	Bits 29-24	—	—
PRI26	Bits 23-22	R/W	**IRQ26** 优先级
—	Bits 21-16	—	—
PRI25	Bits 15-14	R/W	**IRQ25** 优先级
—	Bits 13-8	—	—
PRI24	Bits 7-6	R/W	**IRQ24** 优先级
—	Bits 5-0	—	—

2. 4. 2. 15 NVIC IRQ28 – IRQ31 优先级控制寄存器 (NVIC_IPR7)

NVIC IRQ20 – IRQ23 优先级控制寄存器(NVIC_IPR3)																													
偏移地址:0x31C																													
复位值:0x0000 0000																													
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
PRI31<1:0>								PRI30<1:0>								PRI29<1:0>								PRI28<1:0>							

PRI31	Bits 31-30	R/W	**IRQ31** 优先级
—	Bits 29-24	—	—
PRI30	Bits 23-22	R/W	**IRQ30** 优先级
—	Bits 21-16	—	—
PRI29	Bits 15-14	R/W	**IRQ29** 优先级
—	Bits 13-8	—	—
PRI28	Bits 7-6	R/W	**IRQ28** 优先级
—	Bits 5-0	—	—

2. 4. 2. 16 CPU ID 寄存器 (SYS_CPUID)

CPU ID 寄存器 (SYS_CPUID)																													
偏移地址:0x00																													
复位值:0x410C C200																													
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
IMPC<7:0>								—	—	—	—	PART<3:0>				PARTNO<11:0>												REV<3:0>			

IMPC	Bits 31-24	R	**ARM 分配的实现代码** ARM = 0x41
—	Bits 23-20	—	—
PART	Bits 19-16	R/W	**处理器的结构**
PARTNO	Bits 15-4	R	**处理器的零件号**
REV	Bits 3-0	R/W	**修订号**

2. 4. 2. 17 中断控制与状态寄存器 (SYS_ICSR)

中断控制与状态寄存器 (SYS_ICSR)																													
偏移地址:0x04																													
复位值:0x0000 0000																													
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NMISP	—	—	PENDSV	PENDSVC	PENDST	PENDSTC	—	ISRPRE	SRPEND	—	—	—	—	VTPEND<5:0>						—	—	—	—	—	—	VTACT<5:0>					

NMISP	Bit 31	R/W	**NMI (非可屏蔽中断) 挂起位** 0:在读取时，指示 NMI 异常未挂起。在写入时，没有影响。 1:在读取时，指示 NMI 异常挂起。在写入时，将 NMI 异常状态更改为挂起状态。 因为 NMI 是最高优先级的异常，所以处理器通常在检测到 1 对这个位的写时立即进入 NMI 异常处理程序。进入处理程序然后将此位清除为 0。这意味着，只有当处理器正在执行 NMI 异常处理程序时，NMI 信号被重新置入，NMI 异常处理程序才读取该位，返回 1。

—	Bits 30-29	—	—
PENDSV	Bit 28	R/W	**PendSV 挂起位** 0:在读取时，指示 PendSV 异常未挂起。在写入时，没有影响。 1:在读取时，指示 PendSV 异常挂起。在写入时，将 PendSV 异常状态更改为挂起状态。 只有通过写一个“1”到这个位，才能将 PendSV 异常状态设置为挂起状态。
PENDSVC	Bit 27	W	**PendSV 清除挂起位** 0:没有影响。 1:从 PendSV 异常中移除挂起状态。
PENDST	Bit 26	R/W	**SysTick 异常挂起位** 0:在读取时，指示一个 SysTick 异常未挂起。在写入时，没有效果。 1:在读取时，指示一个 SysTick 异常挂起。在写入时，将 PendSV 异常状态更改为挂起状态。
PENDSTC	Bit 25	W	**SysTick 异常清除挂起位** 0:没有影响。 1:从 SysTick 异常移除挂起状态。
—	Bit 24	—	—
ISRPRE	Bit 23	R	**调试中断处理** 0:停止退出不中断 1:在调试停止状态退出时，挂起异常将会执行。
SRPEND	Bit 22	R	**中断挂起标志位，不包括 NMI 和故障** 0:中断未挂起 1:中断挂起
—	Bits 21-18	—	—
VTPEND	Bits 17-12	R/W	**中断挂起向量数** 0:没有挂起的异常。 其它:最高优先级挂起启用异常的异常数目。
—	Bits 11-6	—	—
VTACT	Bits 5-0	R/W	**活动异常数** 此字段包含活动异常数。 0:线程模式。 其它:当前活动异常的异常数目。

2. 4. 2. 18 应用中断和复位控制寄存器 (SYS_AIRCR)

应用中断和复位控制寄存器(SYS_AIRCR)																															
偏移地址:0x0C																															
复位值:0xFA05 0000																															
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
VTKEY<15:0>																													SYSRERQ	VTACTC	

VTKEY	Bits 31-16	R/W	**寄存器访问密钥** 当写入此寄存器时，需要将 VTKEY 字段设置为 0x05FA，否则将忽略写入操作。 VTKEY 用于防止对该寄存器的意外写入重置系统或清除异常状态。
—	Bits 15-3	—	—
SYSRERQ	Bit 2	R/W	**系统重置请求** 0:不请求支持 1:请求支持
VTACTC	Bit 1	R/W	**清除活动 NMI /故障** 保留用于调试使用。当写入寄存器时，用户必须向该位写入 0，否则反应是不可预测的。
—	Bit 0	—	—

2. 4. 2. 19 系统控制寄存器 (SYS_SCR)

系统控制寄存器 (SYS_SCR)																														
偏移地址:0x10																														
复位值:0x0000 0000																														
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																											EVONPEND		SLPDEEP	SLPONE	

—	Bits 31-5	—	—
EVONPEND	Bit 4	R/W	**在挂起位上发送事件** 0:只有使能的中断或事件才能唤醒处理器。(不包括禁用的中断) 1:所有使能的中断、事件和禁用中断都可以唤醒处理器。 当事件或中断进入挂起状态时，事件信号从 WFE 唤醒处理器。如果处理器不等待事件，则事件被记录并影响下一个 WFE。 处理器还唤醒 SEV 指令或外部事件的执行。
—	Bit 3	—	—
SLPDEEP	Bit 2	R/W	**深度睡眠与睡眠模式选择** 控制处理器是否使用睡眠或深度睡眠作为其低功耗模式: 0:睡眠模式。 1:深度睡眠模式。
SLPONE	Bit 1	R/W	**退出启用睡眠** 0:返回线程模式时不要睡觉。 1:当从 ISR 返回到线程模式时，进入睡眠或深度睡眠。 将此位设置为 1 使得中断驱动的应用程序避免返回空的主应用程序。
—	Bit 0	—	—

2. 4. 2. 20 系统处理程序优先级寄存器 2 (SYST_SHPR2)

系统处理程序优先级寄存器 2 (SYST_SHPR2)																															
偏移地址:0x1C																															
复位值:0x0000 0000																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PRISH11<1:0>																															

PRISH11	Bits 31-30	R/W	**系统处理程序 11 优先级 - SVCall** “0”表示最高优先级，“3”表示最低优先级。
—	Bits 29-0	—	—

2. 4. 2. 21 系统处理程序优先级寄存器 3 (SYST_SHPR3)

系统处理程序优先级寄存器 3 (SYST_SHPR3)																															
偏移地址:0x20																															
复位值:0x0000 0000																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PRISH15<1:0>								PRISH14<1:0>																							

PRISH15	Bits 31-30	R/W	**系统处理程序 15 优先级 - SVCall** “0”表示最高优先级，“3”表示最低优先级。
—	Bits 29-24	—	—
PRISH14	Bits 23-22	R/W	**系统处理程序 14 优先级- SVCall** “0”表示最高优先级，“3”表示最低优先级。
—	Bits 21-0	—	—

第3章　系统配置控制器 (SYSCFG)

3. 1　概述

系统配置控制器主要针对系统级的应用与电源管理进行说明，用户可通过阅读此章节了解如何配置电源电压侦测，同时也可了解如何开启低功耗模式来降低芯片的功率消耗，让使用外部电源的用户能够延长装置使用的时间。

3. 2　特性

- 支持配置欠压复位(BOR)。
- 支持配置低电压检测(LVD)用于检测系统电源。
- 支持 2 种低功耗模式。
- 支持调试模式(Debug Mode)下让控制器局域网络(CAN bus)暂停计数。
- 支持调试模式(Debug Mode)下让独立看门狗(IWDT)与窗口看门狗(WWDT)暂停计数。
- 支持调试模式(Debug Mode)下让计数器(Timer)暂停计数。
- 支持调试模式(Debug Mode)下让 I2C(SMBus)暂停计数。
- 支持 HOSC 发生故障或是 CPU 发生 Hard Fault 时，让定时器(Timer)暂停计数。

3. 3 功能描述

3. 3. 1 电源

此芯片的电源为外部电压源提供，支持 2.4~5.5V 的电压操作范围。系统内部依据工作电压的不同共可分为 3 个电压域，分为使用外部供电的系统电源域(VDDH Domain)、提供 ADC、CMP 与 OPA 电路电源的模拟电源域(VDDA Domain)，以及使用电压调节器的核心电源域(VDD Domain)。

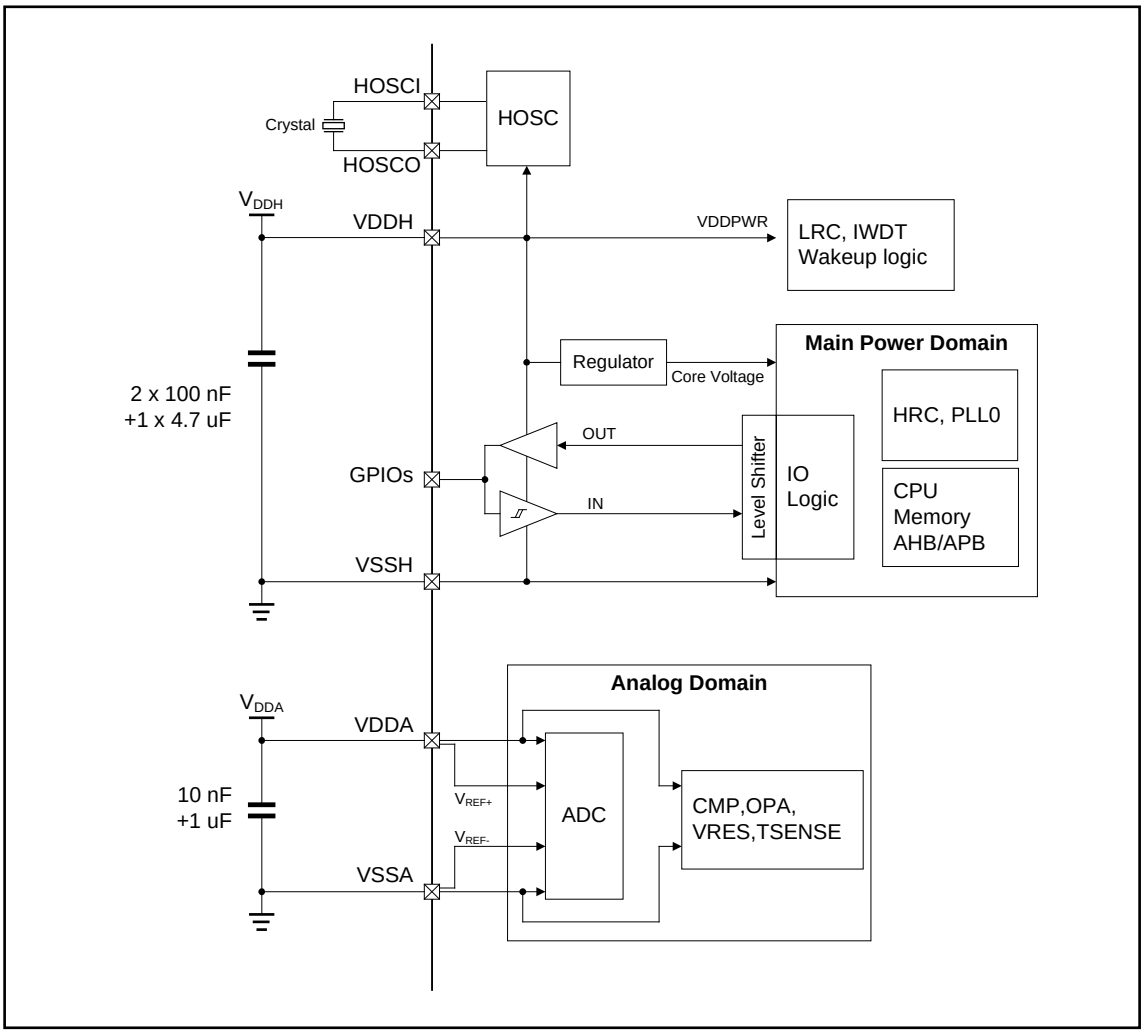


图 3-1 电源架构

模拟电源域(VDDA Domain)的电源预设为未开启，需要使用 ADC、CMP、OPA 电路、温度传感器或电阻分压电路前，用户需将 **SYSCFG_APWR** 内的 VDDA_PWREN 设定为 1，并等待约 10us 的稳定时间以确保稳定性。

3. 3. 1. 1 稳压器 (Voltage Regulator)

芯片内建 1 个稳压器，功能如下:

- 核心稳压器:稳压器可提供稳定的电压，确保 VDD 电源域内的 CPU、SRAM、闪存、AHB 外设以及 APB 外设能够稳定运作。由于稳压器使用的是 VDDH 电源，因此芯片刚上电时为确保电源稳定会让稳压器为关闭状态，系统会在开机流程内等待电源稳定后才开启稳压器，避免影响到 VDD 电源域内的逻辑。

3. 3. 1. 2 电源侦测

- 上电/掉电复位(Power On/Down Reset)

  当系统电源从 0 伏上升至超过 V_{POR} 时 POR 逻辑会再等待约 3.5 毫秒(3.5ms)后会拉高 POR 标志位，此时系统便会离开复位模式并开始执行开机流程。当系统电源从 VDDH 降至低于 V_{PDR} 时会拉低 POR 标志位，此时系统会进入复位状态。 POR 检测上电/掉电复位发生时的电压为固定值，并不支持用户自行配置，此电压值仅受温度与芯片制程影响。系统在上电的过程中产生的 POR 复位标志位会记录于 RCU 逻辑内，用户可于程序内加入检查复位标志位的流程。

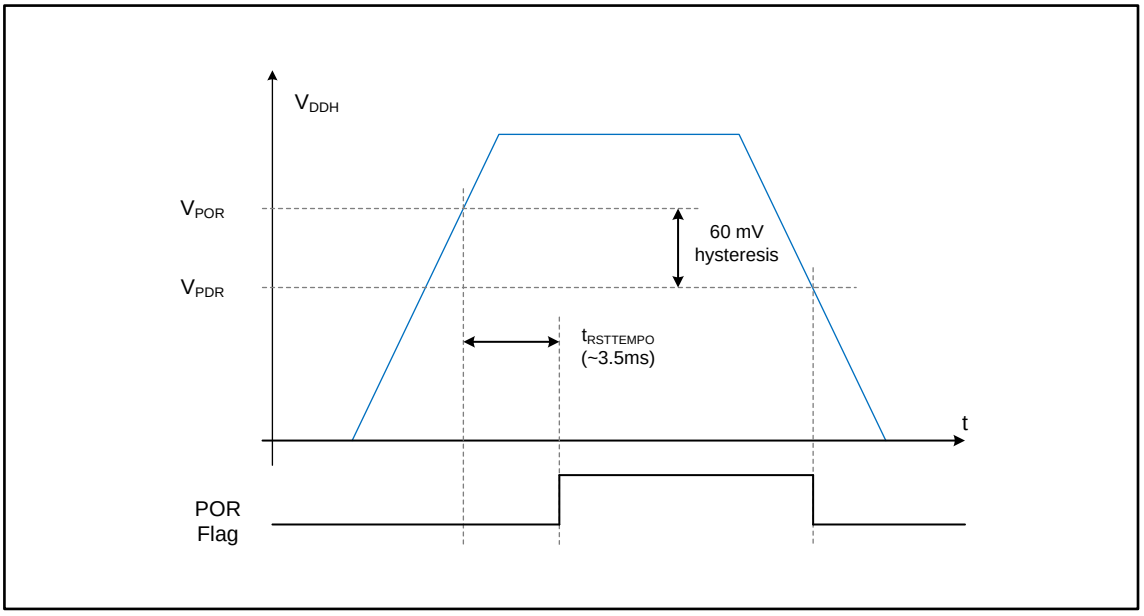


图 3-2 POR/PDR 复位

◆ 欠压复位(Brownout Reset)

欠压复位仅能够过修改用户配置字来开启，无法藉由修改 **SYSCFG_PWRCON** 内的 BORLS(位 8 至位 10)与 BOREN(位 11)开启或关闭欠压复位的功能。当系统电压从 VDDH 降至低于用户配置的电压 V_R 时便会触发系统复位，直到 VDDH 电压上升超过 V_F 时才会离开复位状态。系统复位发生时会清除 VDD 电源域的数据与 VDDH 电源域内的备份寄存器，并产生欠压复位标志位供用户检测。

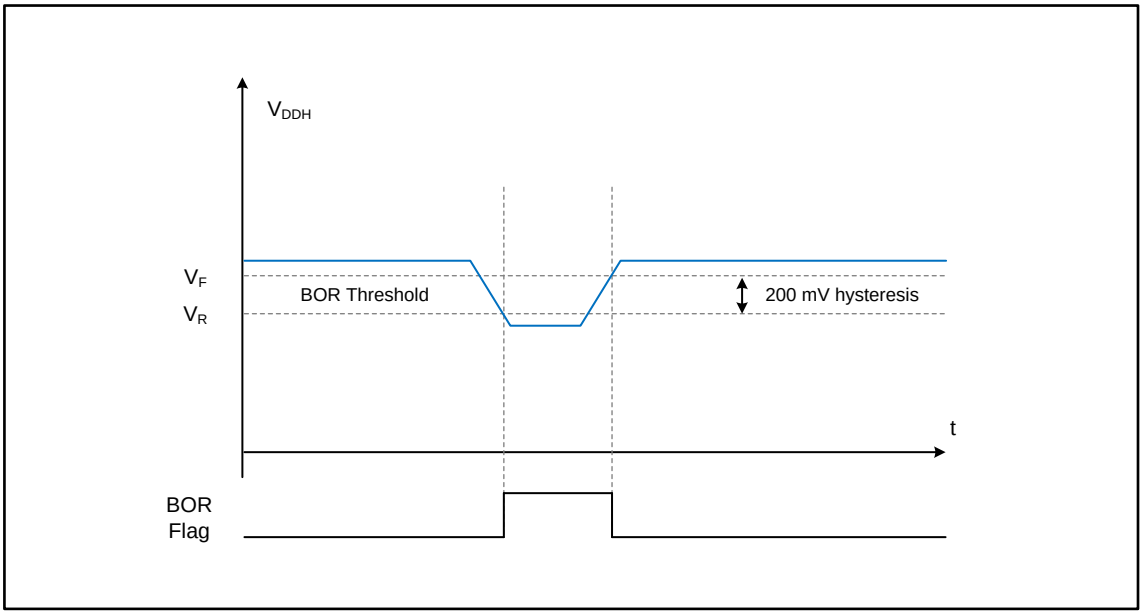


图 3-3 BOR 复位

◆ 低电压检测(Low Voltage Detector)

用户可配置 **SYSCFG_PWRCON** 内的 LVDLS(位 0 至位 3)决定要检测的电压值 V_R 与 V_F，再配置 LVDEN(位 4)为 1 开启电压检测功能。当系统电源从 VDDH 降至低于用户配置的电压 V_R 时，便会发出低电压标志位通知位于 APB 内的 EXTI 逻辑，若用户有开启 EXTI 的中断功能则会在低电压标志位产生时触发中断，让用户能够提前在掉电复位发生前将重要信息进行保留。由于 EXTI 逻辑支持用户检测低电压标志位的上升沿发生点与下降沿发生点，因此当电源上升至超过 V_F 时，低电压标志位会因为被拉低而再一次产生中断告知用户。此外低电压检测标志位也可当作外部触发信号唤醒处在 STOP 模式的系统。

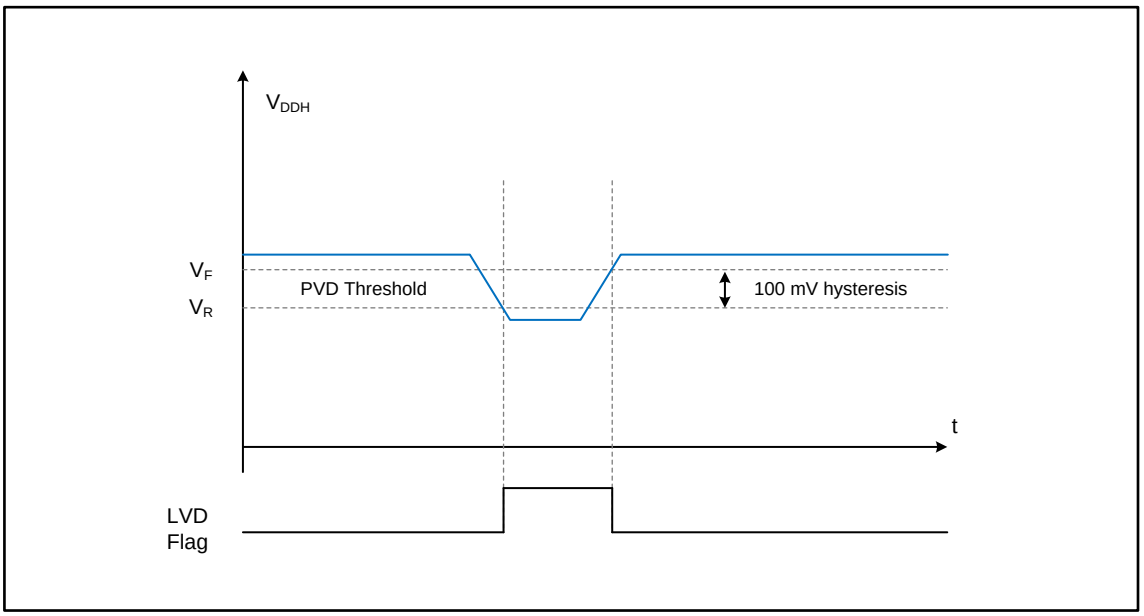


图 3-4 LVD 复位

3. 3. 2 低功耗模式 (Low Power Mode)

当用户需要让程序长时间进行待机或是等待外部信号触发时，可让系统进入低功耗模式进行等待，藉此降低系统功率消耗。若用户希望在系统持续运行的情况下降低功率消耗，可藉由降低系统时钟频率以及关闭不使用的外设时钟来达成。

模式	进入条件	唤醒条件	VDD 域时钟源	VDDH 电源域时钟源	LDO/BandGap	唤醒时间
SLEEP 模式	WFI/WFE	任意中断	关闭 CPU 时钟与未被选择的外设时钟	IWDT 功能或用户决定配置 LRC 开/关	LDO ON & BandGap ON	<4us
STOP 模式	LPLS bits & SLPDEEP bit & WFI/WFE	$WKUP_0$ ~ $WKUP_8$, IWDT, NRST, LVD 触发 EXTI 唤醒中断	VDD 域时钟源全部关闭		LDO low-power & BandGap ON	<400us

表 3-1 低功耗模式

3. 3. 2. 1 低速执行(Low Speed)

芯片支持用户配置系统运行在低速，当程序需要长时间等待触发信号时，可通过对系统时钟预分频来降低运行速度，支持时钟设定 2、4、8、16、64、128、256 与 512 的分频比，同时也可暂时关闭不会使用的时钟源与外设时钟，藉此降低系统的功率消耗。当系统收到唤醒信号以后，再重新进行系统时钟配置，恢复系统正常运行。

3. 3. 2. 2 SLEEP模式

进入 SLEEP 模式后会暂时关闭 CPU 时钟与外设时钟但并不会关闭 VDD 电源域的电源，因此外设的配置与 SRAM 内的数据依然会保存。系统进入 SLEEP 模式时可额外配置 RCU 逻辑内位的 **RCU_AHBSL**、**RCU_APB1SL** 与 **RCU_APB2SL**，让被开启的外设在系统进入 SLEEP 模式后依然能够继续运行。

- 让 CPU 进入睡眠模式

  配置 CPU 系统控制寄存器(System Control Register)内的 SLEEPDEEP(位 2)为 0 后即可搭配 WFI 让 CPU 进入睡眠模式。在此模式下默认关闭 CPU 时钟与外设时钟，用户若有需要让外设持续运行的需求时，可配置 RCU 逻辑内的 **RCU_AHBSL**、**RCU_APB1SL** 与 **RCU_APB2SL** 寄存器来开启外设时钟。

- 从 SLEEP 模式唤醒

  当系统进入 SLEEP 模式时，可选择使用外设中断或是拉低外部 NRST 引脚来唤醒 CPU。

  ◇ 外设中断唤醒:

  使用外设中断唤醒时需在系统进入 SLEEP 模式前配置 RCU 逻辑内 **RCU_AHBSL**、**RCU_APB1SL** 与 **RCU_APB2SL** 的 SLEEP 模式外设时钟使能寄存器开启外设时钟，让外设在 SLEEP 模式也能继续运行。使用中断唤醒 CPU 时并不会产生低功耗复位标志位，同时 CPU 会在唤醒后继续往下执行用户代码。

  ◇ 低电压检测标志位唤醒(LVD):

  当用户开启低电压检测功能时，会在 VDDH 电压降至用户选定的电压时产生 LVD 标

志位，当 LVD 标志位发生时可触发 EXTI 中断唤醒 CPU。

◇ NRST 引脚唤醒:

当用户拉低外部 NRST 引脚时会重置系统并产生 NRST 复位标志位，用户可于系统重启后检查 RCU 逻辑内的 NRST 复位标志位。

当 CPU 被唤醒后，会继续往下执行用户程序，CPU 可于被唤醒后检查中断来判断又由哪一个外设中断事件唤醒。

3. 3. 2. 3 低功耗STOP模式

进入低功耗 STOP 模式后，除了关闭所有时钟源外，还会将核心电压调节器进入低功耗模式。STOP 模式不支持使用外设中断进行唤醒，唯一能够唤醒 CPU 的方式是通过唤醒标志位触发的 EXTI 唤醒中断，该标志位可以由唤醒引脚(WKUPx)、IWDT 事件、低电压检测(LVD)或拉低外部 NRST 引脚来触发。建议在进入低功耗 STOP 模式前，将系统时钟切换至高速 RC 时钟(HRC)，并在唤醒后重新配置系统时钟。

当 CPU 被唤醒后，可以通过读取 **SYSCFG_WKSR** 内的 FG 位(位 0 至位 15)来判断是哪个唤醒事件唤醒了 CPU。然后，将 **SYSCFG_WKSR** 内的 WKCLR 位(位 31)设为 1，以清除唤醒标志位。否则，系统无法再次进入低功耗模式。建议在设定 WKCLR 为 1 后，加入读取判断，确认 WKCLR 是否被自动清除为 0。只有在 WKCLR 被清除为 0 后，才能再次进入低功耗模式。

此外，在进入 STOP 模式之前，用户可以将 **FC_CTL** 内的 FCSLEEP 设定为 1，以启用闪存停止模式。当系统进入 STOP 模式时，闪存也会进入闪存停止模式，进一步降低电流消耗。如果需要使用此功能，必须根据当前系统频率配置 **RCU_CFG2** 内的 SYSFREQ，以确保闪存有足够的时间可以离开闪存停止模式。

◆ 让 CPU 进入深度睡眠模式

当用户配置 **SYSCFG_WKSR** 内的 LPLS(位 28 至位 29)为 1，并配置 CPU 内系统控制寄存器(System Control Register)内的 SLEEPDEEP(位 2)为 1 后即可搭配 WFI 让 CPU 进入深度睡眠模式。

◆ 从 STOP 模式唤醒

◇ WKUPx 唤醒:

系统最大支持用户使用 9 个唤醒引脚(WKUPx)唤醒。用户需先配置 **SYSCFG_WKUP** 内 WKEN 的位 0 至位 7 与位 11 为 1，以开启外部唤醒引脚。接着，需配置 WKEG(位 16 至位 23 与位 27)，以决定要在唤醒信号的上升沿或是下降沿发生时产生唤醒标志位。

◇ IWDT 唤醒:

用户可配置 IWDT，并于 IWDT 发生复位时唤醒系统。当用户开启 IWDT 后，**SYSCFG_WKUP** 内 WKEN 的位 10 会自动设定为 1 并开启 IWDT 唤醒功能。由于 IWDT 仅支持侦测下降沿发生时唤醒，因此用户无需再配置 WKEG。

◇ 低电压检测标志位唤醒(LVD):

当用户开启低电压检测功能时，会在 VDDH 电压降至用户选定的电压时产生 LVD

标志位，并唤醒系统。当用户开启检测功能时，**SYSCFG_WKUP** 内 WKEN 的位 9 会自动设定为 1 并开启 LVD 标志位唤醒功能。由于 LVD 标志位仅支持侦测上升沿发生时唤醒，因此用户无需再配置 WKEG。

◇ NRST 引脚唤醒:

当用户拉低外部 NRST 引脚时会产生 NRST 唤醒事件，并且在清除唤醒标志位前不会产生 NRST 复位。由于 NRST 仅支持侦测上升沿发生时唤醒，因此用户毋须再配置 WKEG。

当 CPU 被唤醒后，系统会继续执行用户程序。CPU 可通过读取 **SYSCFG_WKSR** 内的 FG 位来判断是由哪一个唤醒事件唤醒。为了确保系统可以再次进入低功耗模式，需要配置 **SYSCFG_WKSR** 内的 WKCLR 为 1 来清除唤醒标志位，否则系统无法进入低功耗模式。建议在设置 WKCLR 为 1 后，应加入读取和检查 WKCLR 是否被自动清除为 0 的程序，以确保系统能够再次进入低功耗模式。

3. 3. 3 系统重映射(Remap)

此芯片的存储区共分为 2 个区块，分别为闪存(Flash)以及系统内存(System Memory)。闪存可供用户存放代码，而系统内存主要存放芯片下载程序(Bootrom)，其主要功能为协助用户将代码下载至闪存内。当系统开机完毕后默认会从系统内存开始执行芯片下载程序，若用户未与芯片下载程序进行同步沟通，则下载程序在等待约 20ms 后会自动映射至闪存开始执行用户代码。

- 硬件重映射(Hardware Remap) 硬件重映射的主要功能为当用户不再需要使用芯片下载程序(Bootrom)，可于系统开机完毕后直接从闪存开始执行用户的代码，达到快速开机的效果。硬件映射可由用户决定是否开启，若决定开启此功能时用户需自行配置位于闪存信息区页 2 内的用户配置字。有关用户配置字的说明请参阅闪存控制器章节内的描述。

- 软件重映射(Software Remap) 软件重映射支持用户重新选择要执行闪存的程序或是执行位于系统内存内的芯片下载程序，如当系统运行在闪存时可藉由软件重映射的功能重新映射回系统内存执行芯片下载程序。当用户在使用软件重映射的功能时，需配置 **SYSCFG_REMAP** 内的 MEMMOD(位 2 至位 3)来决定后续程序要映射至哪一个存储区继续执行，并配置 REMAP(位 0)为 1 开启重映射流程。开启重映射流程以后系统并不会立即进行映射，用户需使用 NVIC_SystemReset()复位函数重置 CPU 后，系统才会真正完成重映射程序。当系统完成映射流程以后，可读取 **SYSCFG_REMAP** 内的 REALMOD(位 11 至位 10)，检查当前主存储器映射状态是否与配置的结果相符。

- 闪存重映射(Flash Remap) 当用户使用软件重映射将系统映射至闪存执行时，可进一步设定闪存的内部映射。内部映射是以 8 个页(4K Byte)为基准进行映射，用户可在配置 **SYSCFG_REMAP** 内的 MEMMOD(位 2 至位 3)时再额外配置 EFBASE(位 4 至位 8)来决定要映射至闪存的哪一个 4K Byte 区块开始执行。闪存的内部映射只有在读取闪存的数据时才会进行映射，当用户对闪存进行编程与擦除时并不会受到内部映射影响。此外若用户使用闪存的实体位置来读取数据时，同样不会受到内部映射的影响。

3. 3. 4 停止外设计数

芯片支持控制器局域网络(CAN bus)、计数器(Timer)、看门狗与 I2C 在调试模式下能够暂时停止计数。用户可藉由配置 **SYSCFG_CFG** 的 DBGHEN(位 16 至位 31)决定是否开启相对映外设的功能。当开启看门狗在调试模式停止计数的功能后，可避免用户在调试模式时因为没有重置看门狗而触发系统复位。而计数器则支持在调试模式下停止输出无法控制的 PWM 信号。除了调试模式以外，用户也可配置 **SYSCFG_CFG** 内的 LVD_LCK(位 14)、CSS_LCK(位 13)与 CPU_LCK(位 12)让系统发生低电压检测标志位(LVD)、系统时钟安全标志位(CSS)或是 CPU 发生锁定时让计数器暂时停止计数。

3. 4 特殊功能寄存器

3. 4. 1 寄存器列表

SYSCFG 寄存器列表			
名称	偏移地址	类型	描述
SYSCFG_REMAP	0000_H	R/W	系统启动内存重映射寄存器
SYSCFG_APWR	0038_H	R/W	仿真电路电源开关寄存器
SYSCFG_CFG	$003C_H$	R/W	系统配置寄存器
SYSCFG_PWRCON	0040_H	R/W	系统电源检测控制寄存器
SYSCFG_WKUP	0048_H	R/W	低功耗模式(Low Power Mode)与唤醒状态寄存器
SYSCFG_WKSR	$004C_H$	R/W	唤醒状态寄存器

3. 4. 2 寄存器描述

3. 4. 2. 1 系统启动内存重映射寄存器 (SYSCFG_REMAP)

系统启动内存重映射寄存器(SYSCFG_REMAP)																												
偏移地址:0x00																												
复位值:0x0000 0804																												
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
—	—	—	—	—	—	—	—	—	—	—	—	REALBASE<6:0>							REALMOD<1:0>		EFBASE<6:0>							MEMMOD<1:0>		—	REMAP

—	Bit 31-20	—	—
REALBASE	Bit 19-13	R	**当前闪存起始地址状态** 这些位表示当前闪存映射至主存储器时，是以哪一个 4KB 为单位配置闪存起始位置。 举例来说，当 REALBASE 数值为 1 时，代表当前闪存第二个 4KB 的位置会映射至主存储器 0x0000 0000 的位置。
REALMOD	Bit 12-11	R	**当前主存储器映射状态** 这些位表示当前主存储器的映射状态，可藉由配置 MEMMOD 与 REMAP 开启软件重映射流程，来改变主存储器映射状态。 0x0:闪存映射至主存储器。 0x1:System Memory 映射至主存储器。 0x2: SRAM 映射至主存储器。 0x3:保留。
EFBASE	Bit 10-4	R/W	**重映射闪存起始地址选择** 闪存映射至主存储器时，可以 4KB 为单位配置闪存起始位置。 举例来说，当 EFBASE 数值为 1 时，闪存第二个 4KB 的位置会映射至主存储器 0x0000 0000 的位置。
MEMMOD	Bit 3-2	R/W	**主存储器映射选择** 主存储器映射流程依据此寄存器的配置内容，将相对应的内存位置映射至主存储器。 0x0:闪存映射至主存储器。 0x1: System Memory 映射至主存储器. 0x2: SRAM 映射至主存储器。

			0x3:保留。
—	Bit 1	—	—
REMAP	Bit 0	W1	**主存储器重映射启动** 配置此位为高时启动主存储器重映射流程，待映射流程结束后自动拉低此位。 0:主存储器重映射流程已结束。 1:启动主存储器重映射流程。

3. 4. 2. 2 仿真电路电源开关寄存器 (SYSCFG_APWR)

仿真电路电源开关寄存器 (SYSCFG_APWR)																													
偏移地址:0x38																													
复位值:0x0000 0000																													
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																													VREFINT_EN	VTSENSE_EN	VDDA_PWREN

—	Bit 31-3	—	—
VREFINT_EN	Bit 2	R/W	**ADC、CMP 参考电压电源开关** 当 ADC、CMP 需要量测参考电压时，需配置此开关开启电源。开启参考电压电源后，需等待 10us 的稳定时间。 0: 关闭 ADC、CMP 参考电压电源。 1: 开启 ADC、CMP 参考电压电源。
VTSENSE_EN	Bit 1	R/W	**温度传感器开关** 0: 关闭温度传感器 1: 开启温度传感器
VDDA_PWREN	Bit 0	R/W	**VDDA 电源域电源开关** 当使用 ADC、CMP、OPA、TSENSE 或 VRES 时，需配置此开关开启电源。开启 VDDA 电源域电源后，需等待 10us 的稳定时间。 0: 关闭 VDDA 电源域电源。 1: 开启 VDDA 电源域电源。

3. 4. 2. 3 系统配置寄存器 (SYSCFG_CFG)

系统配置寄存器 (SYSCFG_CFG)																													
偏移地址:0x3C																													
复位值:0x0000 0101																													
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
DBGHEN<15:0>																—	LVDLCK	CSSLCK	CPULCK	—	—	—	—	—	—	—	—	—	—	—	—

DBGHEN	Bit 31-16	R/W	**调试模式外设停止开关** 配置 DBGHEN 让外设在 CPU 进入调试模式后，能够停止计数。DBGHEN[X]配置为 1 代表开启调试模式下停止计数的功能；配置为 0 时，外设在 CPU 进入调试模式后仍会继续计数。 DBGHEN[15] : 保留。 DBGHEN[14] : CAN1。 DBGHEN[13] : IWDT。 DBGHEN[12] : WWDT。 DBGHEN[11] : 保留。 DBGHEN[10] : I2C1。 DBGHEN[9] : 保留。 DBGHEN[8] : 保留。 DBGHEN[7] : 保留。 DBGHEN[6] : BS16T1。 DBGHEN[5] : GP32C4T2。 DBGHEN[4] : GP32C4T1。 DBGHEN[3] : GP16C2T2。 DBGHEN[2] : GP16C2T1。 DBGHEN[1] : AD16C6T2。 DBGHEN[0] : AD16C6T1。
—	Bit 15	—	—
LVDLCK	Bit 14	R/W	**低电压检测(LVD)事件输入开关** LVD 事件可从 AD16C6T1、AD16C6T2 与 GP16C2T1~GP16C2T2 的断路输入(Break Input)脚位输入，有关断路输入的说明请参阅 Timer 章节内的描述。

			0:关闭 LVD 事件输入至 Timer。 1:开启 LVD 事件输入至 Timer。
CSSLCK	Bit 13	R/W	**时钟安全事件(CSS)输入开关** CSS 事件可从 AD16C6T1、AD16C6T2 与 GP16C2T1~GP16C2T2 的断路输入(Break Input)脚位输入，有关断路输入的说明请参阅 Timer 章节内的描述。 0:关闭 CSS 事件输入至 Timer。 1:开启 CSS 事件输入至 Timer。
CPULCK	Bit 12	R/W	**CPU Lockup 事件输入开关** CPU Lockup(Hard Fault)事件可从 AD16C6T1、AD16C6T2 与 GP16C2T1~GP16C2T2 的断路输入(Break Input)脚位输入，有关断路输入的说明请参阅 Timer 章节内的描述。 0:关闭 CPU Lockup 事件输入至 Timer。 1:开启 CPU Lockup 事件输入至 Timer。
—	Bit 11-0	—	—

3. 4. 2. 4 系统电源检测控制寄存器(SYSCFG_PWRCON)

此寄存器只允许 32 位存取

系统电源检测控制寄存器 (SYSCFG_PWRCON)																												
偏移地址:0x40																												
复位值:0x0000 0000																												
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																IWDTEN				BOREN	BORLS<2:0>						LVDEN	LVDLS<3:0>			

—	Bit 31-16	—	—
IWDTEN	Bit 15	R	**IWDT 开启状态** 0: IWDT 未被开启。 1: IWDT 已被开启。
—	Bit 14-12	—	—
BOREN	Bit 11	R	**低电压复位开关** 通过用户配置字配置 BOREN 为 1 可开启低电压复位功能，当系统电压低于 BORLS 所选定的电压区间时，便会触发 BOR 复位，藉此确保芯片永远工作在高于选定的电压范围以上。 0:关闭低电压复位。 1:开启低电压复位。
BORLS	Bit 10-8	R	**低电压复位电压区间选择** BORLS 选择触发 BOR 复位的电压区间。详细电压区间请参阅 Datasheet 内的数据。此电压仅能通过用户配置字进行修改。 000: — 001:BOR level 1 010:BOR level 2 011:BOR level 3 100:BOR level 4 101:BOR level 5 110:BOR level 6 111:BOR level 7
—	Bit 7-5	—	—
LVDEN	Bit 4	R/W	**低电压检测开关** 配置 LVDEN 为 1 可开启低电压检测器，此检

			测器用于检查系统电压值，当系统电压低于选 LVDLS 所选择的电压区间时，检测器会拉高 LVD 标志位，直到系统电压高于选择的电压区间时才会将标志位拉低。此标志位可当作低功耗模式的唤醒事件、触发 CPU 中断事件或是让 Timer 停止计数。 0:关闭低电压检测器。 1:开启低电压检测器。
LVDLS	Bit 3-0	R/W	**低电压检测电压区间选择** 配置 LVDLS 选择低电压检测器所需要检测的系统电压区间.当系统电压降至选则的区间时，低电压检测器会拉高 LVD 标志位。详细电压区间请参阅 Datasheet 内的数据。 0000: LVD level 0。 0001: LVD level 1。 0010: LVD level 2。 0011: LVD level 3。 0100: LVD level 4。 0101: LVD level 5。 0110: LVD level 6。 0111: LVD level 7。 1000: LVD level 8。 1001: LVD level 9。 1010: LVD level 10。 1011: LVD level 11。 1100: LVD level 12。 1101: LVD level 13。 1110: LVD level 14。 1111: LVD level 15。

3. 4. 2. 5 低功耗模式(Low Power Mode)与唤醒控制寄存器 (SYSCFG_WKUP)

此寄存器只允许 32 位存取

低功耗模式(Low Power Mode)与唤醒控制寄存器(SYSCFG_WKUP)																													
偏移地址:0x48																													
复位值:0x0300 0100																													
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
WKEG <15:0>																WKEN<15:0>														

WKEG	Bit 31-16	R/W	**唤醒事件上升沿或下降沿模式** 此寄存器决定唤醒引脚或事件，由上升沿或下降沿触发唤醒。 0:下降沿发生时，触发唤醒系统。 1:上升沿发生时，触发唤醒系统。 除了IWDT唤醒寄数器与LVD标志位预设检测事件上升沿发生时唤醒系统外，其余唤醒引脚与唤醒事件预设检测事件下降沿发生时唤醒系统。 WKEG[0] : WKUP0引脚上升沿/下降沿事件。 WKEG[1] : WKUP1引脚上升沿/下降沿事件。 WKEG[2] : WKUP2引脚上升沿/下降沿事件。 WKEG[3] : WKUP3引脚上升沿/下降沿事件。 WKEG[4] : WKUP4引脚上升沿/下降沿事件。 WKEG[5] :WKUP5引脚上升沿/下降沿事件。 WKEG[6] :WKUP6引脚上升沿/下降沿事件。 WKEG[7] :WKUP7引脚上升沿/下降沿事件。 WKEG[8] : NRST引脚上升沿唤醒，固定为1。(Read Only) WKEG[9] : LVD标志位上升沿唤醒，固定为1。(Read Only) WKEG[10] : IWDT计数重置下降沿唤醒，固定为0。(Read Only) WKEG[11] :WKUP8引腳上升沿/下降沿事件。 WKEG[12] :保留。 WKEG[13] :保留。 WKEG[14] :保留。

			WKEG[15] :保留。
WKEN	Bit 15-0	R/W	**唤醒引脚或唤醒事件开关** 当系统进入低功耗模式后，通过以下配置触发唤醒。 0:关闭唤醒功能。 1:开启唤醒功能。 WKEN[0] :开启/关闭 WKUP0 引脚唤醒功能。 WKEN[1] :开启/关闭 WKUP1 引脚唤醒功能。 WKEN[2] :开启/关闭 WKUP2 引脚唤醒功能。 WKEN[3] :开启/关闭 WKUP3 引脚唤醒功能。 WKEN[4] :开启/关闭 WKUP4 引脚唤醒功能。 WKEN[5] :开启/关闭 WKUP5 引脚唤醒功能。 WKEN[6] :开启/关闭 WKUP6 引脚唤醒功能。 WKEN[7] :开启/关闭 WKUP7 引脚唤醒功能。 WKEN[8] ： NRST 引脚唤醒功能，固定开启。(Read Only) WKEN[9] : LVD 标志位唤醒功能，依据 LVDEN 配置状态而定。(Read Only) WKEN[10] ： IWDT 计数重置唤醒功能，依据 IWDT 开启状态而定。(Read Only) WKEN[11] :开启/关闭 WKUP8 引脚唤醒功能。 WKEG[12] :保留。 WKEG[13] :保留。 WKEG[14] :保留。 WKEG[15] :保留。

注:

1. BOR 标志位会触发 PDR 复位，无法当作唤醒标志位将系统唤醒。

3. 4. 2. 6 唤醒状态寄存器 (SYSCFG_WKSR)

此寄存器只允许 32 位存取

唤醒状态寄存器(SYSCFG_WKSR)																													
偏移地址:0x4C																													
复位值:0x0040 0000																													
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
WKCLR	—	LPLS<1:0>		LPVSEL<1:0>		—	—	—	—	—	—	—	—	—	FLAG	FG<15:0>															

WKCLR	Bit 31	R/C_W1	**清除唤醒标志位** 设定 WKCLR 为 1 可以清除唤醒标志位并结束低功耗流程，而 WKCLR 会在低功耗流程结束后自动清除为 0。如果使用唤醒引脚或唤醒事件将系统从低功耗模式唤醒，务必设定 WKCLR 来清除唤醒标志位，否则系统无法再次进入低功耗模式。建议在设置 WKCLR 为 1 后，加入读取和检查 WKCLR 是否被自动清除为 0 的程序，以确保系统能够再次进入低功耗模式。
—	Bit 30	—	—
LPLS	Bit 29-28	R/W	**低功耗模式选择** 配置 LPLS 决定系统要进入哪一种低功耗模式。有关各种低功耗模式的说明与唤醒方式，请参阅低功耗模式(Low Power Mode)内的描述。 0: 保留。 1: STOP 模式。 2: 保留。 3: 保留。
LPVSEL	Bit 27-26	R/W	**LDO Low Power Mode 电压选择** 配置 LPVSEL 决定系统要进入 STOP 模式时 LDO 的输出电压值。 0: 保留。 1: 1.4V。(默认值) 2: 1.5V。 3: 保留
—	Bit 25-17	—	—
FLAG	Bit 16	R	**唤醒标志位**

			使用唤醒引脚或是唤醒事件将系统从低功耗模式下唤醒时，FLAG 会被拉高，告知系统被唤醒引脚或是唤醒事件所唤醒。可藉由设定 WKCLR 为 1 来清除标志位。 0:无唤醒标志位。 1:系统由唤醒引脚或是唤醒事件唤醒。
FG	Bit 15-0	R	**低功耗模式唤醒标志位** 使用唤醒引脚或是唤醒事件将系统从低功耗模式中唤醒时，此寄存器记录系统唤醒是由那些唤醒引脚或事件。可藉由设定 WKCLR 位源来清除标志位。 FG[0] :由 WKUP0 引脚上升沿/下降沿唤醒。 FG[1] :由 WKUP1 引脚上升沿/下降沿唤醒。 FG[2] :由 WKUP2 引脚上升沿/下降沿唤醒。 FG[3] :由 WKUP3 引脚上升沿/下降沿唤醒。 FG[4] :由 WKUP4 引脚上升沿/下降沿唤醒。 FG[5] :由 WKUP5 引脚上升沿/下降沿唤醒。 FG[6] :由 WKUP6 引脚上升沿/下降沿唤醒。 FG[7] :由 WKUP7 引脚上升沿/下降沿唤醒。 FG[8] :由 NRST 引脚上升沿唤醒。 FG[9] :由 LVD 标志位上升沿唤醒。 FG[10] :由 IWDT 计数重置下降沿唤醒。 FG[11] :由 WKUP8 引脚上升沿/下降沿唤醒。 FG[12] :保留。 FG[13] :保留。 FG[14] :保留。 FG[15] :保留。

第4章　复位和时钟控制 (RCU)

4.1 概述

此章节内容包含系统时钟架构、时钟相关配置与外设复位设定，用户通过阅读此章节可以了解如何配置系统时钟以及如何使用外设复位。

4.2 特性

- 支持 4 种时钟源可当作系统时钟。
- 支持 AHB 时钟(HLCK)分频，可以根据系统时钟(SYSCLK)设定 2、4、8、16、64、128、256 与 512 的除数。
- 支持 APB 时钟(PCLK)分频，可以根据 AHB 时钟设定 2、4、8 与 16 的除数。
- 支持微控制器时钟输出(MCO)。
- 支持微控制器时钟输出分频，可以设定 2、4、8、16、32、64 与 128 的除数。
- 支持 SLEEP 模式下开启外设时钟。
- 支持时钟安全系统(CSS)。
- 提供复位标志位确认系统状态。

4.3 功能描述

4.3.1 复位

此芯片的复位包含上电/掉电复位、低功耗复位与系统复位，每一种复位都有相对应的标志位于 **RCU_RSTF** 供用户检视。

4.3.1.1 电源复位

当下列事件发生时，会产生电源复位，相关信息请参阅系统配置控制器章节内的电源章节。

- 上电/掉电复位(POR/PDR)

  上电复位是指当系统电源 VDDH 从 0 伏特上升并且超过 VPOR 时，POR 模块会等待约 3.5 毫秒(3.5 ms)才拉高 POR 标志位，此时系统便会离开复位状态并开始执行开机流程。掉电复位是指当系统电源 VDDH 下降并且低于 V_{PDR} 时，POR 模块会拉低 POR 标志位并进入复位状态，以确保所有模块不会因为电源不稳定，而产生非预期的行为。当掉电复位发生时所有寄存器包含备份寄存器都会被清除。

- 欠压复位(BOR)

  当用户希望工作电压下降至一定程度后，提早发生系统复位，可藉由欠压复位提早让芯片进入复位状态，欠压复位的电压可由用户自行配置，发生复位后系统会产生复位标志位，供用户于下一次开机时进行判断。当欠压复位发生时所有寄存器包含备份寄存器都会被清除。

4.3.1.2 系统复位

当系统复位发生时，除了备份寄存器，所有寄存器都将被清除。触发系统复位的原因如下，每一种复位发生后皆会产生相对应的复位标志位。

- 软件复位

  用户可以通过软件使用复位函式 NVIC_SystemReset()来重新启动系统。

- 配置字重载复位

  当用户开启闪存保护设定时，需藉由配置字重载功能来重载保护设定，此时会触发系统复位并让系统重新执行程序。相关信息请参阅闪存控制器章节内的配置字重载章节。

- 看门狗复位

  看门狗分为独立看门狗与窗口看门狗，这两种看门狗都可以定时触发系统复位，避免系统因软件错误或故障导致死机。相关信息请参阅独立看门狗章节与窗口看门狗章节。

- 外部复位

  当用户将 NRST 引脚的信号拉低一段时间(超过 100ns)后，即触发系统复位。

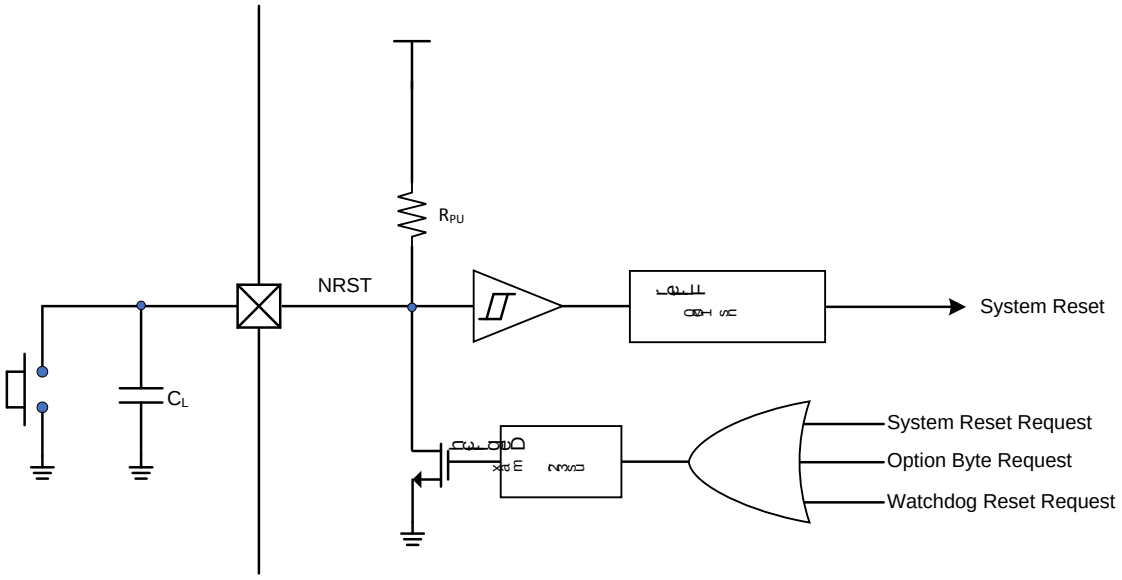


图 4-1 系统复位

4. 3. 1. 3 复位标志位

在系统发生复位时，会记录导致复位的原因。用户可以读取复位标志位寄存器 **RCU_RSTF** 来查看复位原因。由于系统启动时一定会发生上电复位，因此 POR 或 PDR 复位标志位 PORRSTF 的初始值为 1。用户可以检查复位标志位后，设置 CLRFLG 来清除复位标志位。清除完成后，CLRFLG 会自动归零。需要注意的是，当发生上电或掉电复位时，复位标志位寄存器会被复位，只有 POR 或 PDR 复位标志位会被保留。

4. 3. 1. 4 AHB和APB外设复位

AHB 以及 APB 的外设初始皆处于复位状态。当用户配置了 AHB 外设时钟控制寄存器 **RCU_AHBEN**、APB1 外设时钟控制寄存器 **RCU_APB1EN** 或 APB2 外设时钟控制寄存器 **RCU_APB2EN** 之后，需要等待 3 个外设时钟才能使外设离开复位状态。如果用户想复位外设，可以通过配置 AHB 外设复位控制寄存器 **RCU_AHBRST**、APB1 外设复位控制寄存器 **RCU_APB1RST** 或 APB2 外设复位控制寄存器 **RCU_APB2RST** 来实现。当这三组寄存器中的某个被设置为 1 时，外设就处于复位状态。例如，配置 **RCU_AHBRST**.GPAEN 为 1，就可以让 GPIOA 外设进入复位状态。用户需要手动将这些寄存器清除为 0，使外设离开复位状态。

注: 在释放外设复位后，需要等待 3 个外设时钟才能使外设离开复位状态。在此期间内，无法对外设的寄存器进行读写操作。由于 AHB 时钟与 APB 时钟的频率最多可能相差 16 倍，因此在使用此功能时，用户需要特别注意程序上的配置，不建议在将复位控制寄存器清除后立即访问外设的寄存器。

4. 3. 2 时钟

系统时钟(SYSCLK)可选择的时钟源如下:

- HRC - 内部高速 16 MHz RC 振荡器
- HOSC - 外部高速振荡器，支持 4MHz 至 32MHz
- PLL - 锁相环
- LRC - 内部低速 32 KHz RC 振荡器

每种时钟源不使用时，皆可独立设置开启或关闭，进而优化系统功率消耗。并提供分频电路，让用户依据应用场景与功耗需求，配置 AHB 与 APB 操作频率。

所有周边外设时钟状态，都依据所属的总线(Bus)时钟而定，如 AHB 总线时钟为 HCLK，APB 总线时钟为 PCLK。除了下列几个特殊区块各别说明:

- I2C 核心时钟 (I2CCLK) 可以选择以下三种时钟源其中之一:
  - APB
  - SYS
  - HRC
- ADC 核心时钟 (ADCKCLK) 可以选择以下三种时钟源其中之一:
  - PLL
  - SYS
  - AHB
- IWDT 时钟(IWDTCLK)只允许使用 LRC 时钟
- 系统节拍器(System Tick)时钟(STCLK)，可选择由 AHB 总线 HCLK 时钟或分频 8 倍后提供，通过 SysTick 控制寄存器可配置选择

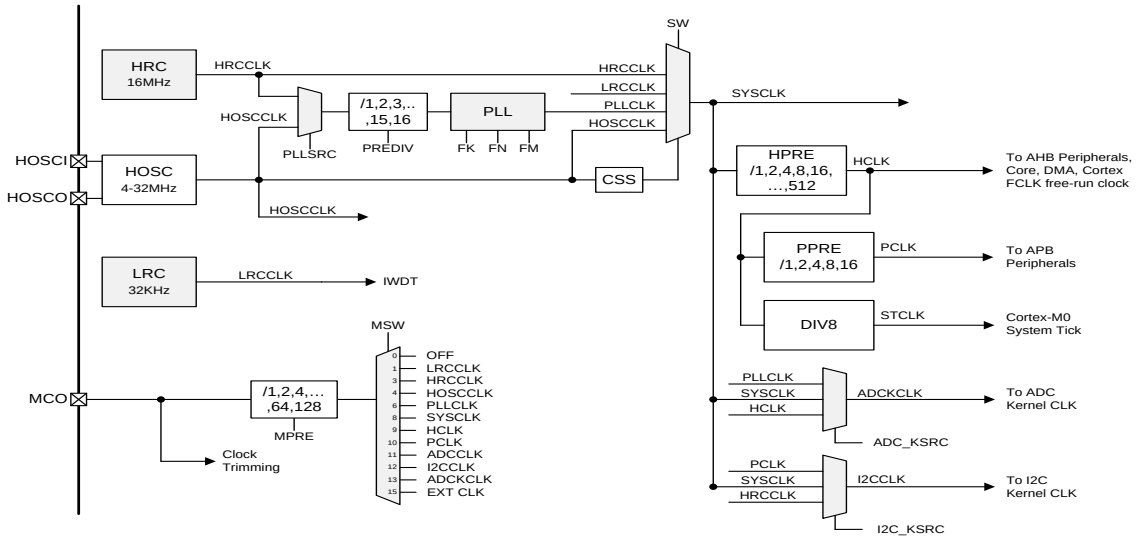


图 4-2 时钟架构图

4. 3. 2. 1 外部高速振荡器时钟 (HOSCCLK)

高速外部时钟信号 HOSC，可通过以下两种来源提供:

- 外部石英晶体振荡器(Crystal Osillator)
- 外部时钟源(External Clock Source)

当使用石英晶体振荡器时，为了减少输出失真与缩短启动稳定时间，必须尽可能将振荡器组件与负载电容 C_{L1}/C_{L2} 靠近 HOSC 引脚。另外在负载电容值选择上，需匹配所选择振荡器。

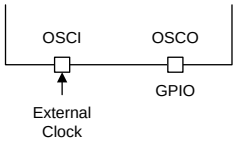
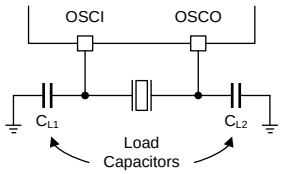
	外部电路配置状态
外部时钟源	
石英振荡器	

图 4-3 HOSC 时钟源

外部时钟源(HOSC Bypass)

此模式必须由外部提供有效的时钟源，并且通过设定寄存器 **RCU_CON**.HOSCBYP 与 **RCU_CON**.HOSCON 启用 HOSC。外部时钟信号源 (方波、弦波或三角波)可由 HOSCI 引脚输入，输入时钟工作周期(Duty Cycle)须介于 40~60%，相关讯息请参阅电器特性章节。此外 HOSCO 引脚可设定为 GPIO 功能引脚，如图 4-3 所示。

外部石英晶体振荡器(HOSC Crystal)

使用石英晶体振荡器的优点是可以提供准确的时钟源。硬件电路配置如图 4-3 所示，其余细节请参阅电器特性章节。配置 **RCU_CON**.HOSCON 位控制 HOSC 开启与关闭。HOSC 时钟状态可藉由 **RCU_CON**.HOSCRDY 标志位确认，当 HOSC 时钟稳定时标志位将被硬件自动配置为高电位，也可以设定 **RCU_IER**.HOSCRDY 位开启 HOSC 时钟稳定中断。

注:开启 HOSC 后，硬件将计数 4096 个 HOSC 时钟周期，即使没有连接石英晶体振荡器，也可能因为 HOSCI 引脚上噪声导致 HOSC 错误启动。关闭 HOSC 后，需经过 5 个 HOSC 时钟周期，才能完成关闭的流程。不论任何原因导致 HOSCI 引脚的时钟消失，都将造成 HOSC 无法被关闭，而产生不必要的功率消耗。因此强烈建议开启时钟安全系统(CSS)功能，即使在这种情况也能关闭 HOSC。

4. 3. 2. 2 内部高速RC振荡器时钟 (HRCCLK)

HRC 时钟信号是由内部高速 16 MHz RC 振荡器产生，其优点为提供低成本的时钟源(无需外部组件)。相较于 HOSC，时钟信号启动时间较快，但精准度仍不如外部晶体振荡器，即使经过频率校准。使用 **RCU_CON**.HRCON 位可控制 HRC 开启与关闭，而 **RCU_CON**.HRCRDY 标志位可用于确认 HRC 时钟状态，当时钟稳定时，该标志位将被硬件自动配置为高电位。此外，可以使用 **RCU_IER**.HRCRDY 位来启用 HRC 时钟稳定中断。若因外在因素造成 HOSC 无法产生时钟信号，HRC 时钟也可作为 HOSC 的辅助时钟，需启用时钟安全系统功能。更多相关讯息，请参阅外部高速振荡器时钟安全系统(HOSC CSS)章节。

4. 3. 2. 3 内部倍频时钟 (PLLCLK)

芯片内建一组锁相环(Phase-Locked Loop，PLL)，可提供非整数倍频的输出时钟频率。输入时钟频率应介于 3 至 16 MHz 之间，用户可以通过设定 **RCU_CFG**.PLLSRC 选择 HRC 或 HOSC 作为输入时钟源，并利用 **RCU_CFG**.PREDIV 预分频将频率除到适当范围。建议输入时钟频率维持在 4 MHz 左右。

输出时钟频率范围介于 4 至 72 MHz 之间，并由压控振荡器(Voltage Controlled Oscillator，VCO)提供。输出时钟频率是 VCO 频率除上一个可变倍率，该倍率可以是整数或非整数。倍率由 **RCU_CFG1**.FN 与 **RCU_CFG1**.FK 控制，其中 FN 表示整数倍率，FK 表示小数倍率。根据输出时钟频率区间调整可以设定 **RCU_CFG1**.FM，FM 可以选择 2、4、6、8、12、16、24、32 或 48。倍率的计算公式如下：

$$f_{VCO}=f_{PLLIN}\times\left(FN+\frac{FK}{2^{19}}\right),\ 192MHz\leq f_{vco}\leq 384MHz$$

$$f_{PLL}=\frac{f_{VCO}}{FM},\ FM=2,\ 4,\ 6,\ 8,\ 12,\ 16,\ 24,\ 32,\ 48$$

注:

压控振荡器 fVCO 频率必须介于 192 至 384 MHz 之间。

FN 必须大于 16，如果设定值小于 16，硬件会自动将 FN 数值设定为 16。

使用 **RCU_CON**.PLLON 位来控制 PLL 开启或关闭；可使用 PLLRDY 标志位检查 PLL 时钟的状态，当 PLL 时钟稳定时，该标志位将自动由硬件配置为高电位，也可以使用 **RCU_IER**.PLLRDY 位设定 PLL 时钟稳定中断。关于 PLL 的详细特性，请参阅电气特性章节。

为了进行 PLL 的配置，必须先选择输入时钟源、进行预分频，以及设定倍率等。这些步骤必须在开启 PLL 之前完成，一旦 PLL 被开启后，就无法再进行这些设定。如果需要重新配置 PLL，必须先将 PLL 关闭，然后再进行相应的设定。修改 PLL 配置的流程如下:

- 关闭 PLL，通过 **RCU_CON**.PLLON 位控制 PLL 关闭。
- 确认 **RCU_CON**.PLLRDY = 0 后，PLL 才算是完成关闭的流程。
- 修改 PLL 配置，包括输入时钟源的选择、预分频和倍率设定。
- 开启 PLL，通过 **RCU_CON**.PLLON 位控制 PLL 开启。

- 确认 PLL 稳定并且可用，可以通过 **RCU_CON**.PLLRDY = 1 标志位或是 PLL 时钟稳定中断确认。

PLL 的配置方法很广泛，下表范例提供了一些参考。例如，范例一假设输入时钟源为 HRC 且预分频 4 倍(PLLSRC = 00 且 PREDIV = 0x3)，范例二假设输入时钟源为 HOSC 30 MHz 且预分频 7 倍(PLLSRC = 01 且 PREDIV = 0x6)，其中 PLL 输出频率的目标值分别为 22.579、24.576、48 以及 72 MHz。

f_{PLLIN} (MHz)	f_{VCO} (MHz)	FM	FN	FK	f_{PLL} (MHz)
4 MHz 输入时钟源: HRC 16 MHz 预分频: 4 倍	270.948	4	67	386400	22.579
	196.608	3	49	79691	24.576
	288	2	72	0	48
	288	1	72	0	72
4.285714 MHz 输入时钟源: HOSC 30 MHz 预分频: 7 倍	270.948	4	63	115972	22.579
	294.912	4	68	426141	24.576
	288	2	67	104857	48
	288	1	67	104857	72

表 4-1　PLL 配置范例

注: RCU 没有针对 PLL 的输入时钟源做硬件检查，因此用户在开启 PLL 之前，需确保选择的输入时钟源已经开启且稳定。

4. 3. 2. 4　　内部低速RC振荡器时钟 (LRCCLK)

内部低速 RC 振荡器 LRC 是一种低功耗时钟源，可让外部设备在低功耗模式下持续运行独立看门狗(IWDT)；其时钟频率约为 32kHz。详细信息请参阅电器特性章节。可以通过设置 **RCU_LCON**.LRCON 位来控制 LRC 的启用和禁用。LRC 时钟状态可通过 LRCRDY 标志位确认，当 LRC 时钟稳定时，此标志位将被硬件自动配置为高电位。也可以设置 **RCU_IER**.LRCRDY 位来启用 LRC 时钟稳定中断。

4. 3. 2. 5　　系统时钟 (SYSCLK)

系统时钟(SYSCLK)可选择的时钟源，如下:

- HRC
- HOSC
- PLL
- LRC

系统复位后，默认的系统时钟源是外部高速振荡器(HRC)。若要将系统时钟源切换至其他时钟源，可以修改 **RCU_CFG**.SW 寄存器，只有当目标时钟源已开启且稳定之后才能切换。注意，某些时钟源无法通过寄存器 **RCU_CON** 进行关闭。可以使用 **RCU_CFG**.SWS 寄存器来确认当前系统时钟源。

4. 3. 2. 6 外部高速振荡器时钟安全系统(HOSC CSS)

外部高速振荡器时钟安全系统(Clock Security System，CSS)是为了避免当系统时钟使用到 HOSC 时，HOSC 因任何外在因素发生故障，进而导致系统死机。可以通过 **RCU_CON**.HOSCCSSON 开启或关闭时钟安全系统，只有当 HOSC 时钟开启且信号稳定才能开启时钟安全系统，并且在侦测到 HOSC 停止时关闭并且将系统时钟切换至 HRC。当开启时钟安全系统时，HRC 与时钟侦测器将自动开启。

系统时钟有使用到 HOSC 的情况

- 系统时钟为 HOSC
- 系统时钟为 PLL，PLL 参考时钟为 HOSC

当侦测到 HOSC 故障时

- 自动关闭 HOSC
- 如果 PLL 为系统时钟(HOSC 为参考频率)，将关闭 PLL。
- 系统时钟自动切换成 HRC
- 时钟故障事件将被送到通用定时器 16 位 6 通道(GP16C6T 1/2)、通用定时器 16 位 2 通道(GP16C2T 1/2)
- 产生时钟安全系统中断(CSSHOSC)

注:当开启时钟安全系统且 HOSC 发生故障时，会触发时钟安全系统中断(CSSHOSC)，此中断与不可屏蔽中断(NMI)异常状态相连接。因此，除非通过使用 RCU_ICR.CSSHOSC 清除时钟安全系统中断状态，否则 NMI 将不断触发。
论 HOSC 是否直接作为系统时钟，或是作为 PLL 的输入时钟并由 PLL 作为系统时钟，当侦测到 HOSC 故障时，系统时钟将自动切换成 HRC，并关闭 HOSC。若 PLL 输入时钟源为 HOSC，也会关闭 PLL。

4. 3. 2. 7 ADC核心时钟、ADC时钟(ADCKCLK、ADCCLK)

当使用 ADC 进行数字转换时，ADC 核心时钟需通过下列设定，调整 ADC 时钟的频率，以符合应用需求。

- ADC 核心时钟可以通过设置 **RCU_CFG2**.ADC_KSRC，选择 ADC 核心时钟的来源为 PLL、系统时钟或 AHB 时钟。需要注意的是，如果要修改时钟来源的配置，必须先关闭时钟来源设置，将 **RCU_CFG2**.ADC_KSRC 设置为 0，然后再进行修改。
- ADC 核心时钟经由 **ADC_CCR**.PRESCALE 来进行分频，以生成 ADC 时钟。

4. 3. 2. 8 I2C核心时钟 (I2CCLK)

I2C 核心时钟来源可通过 **RCU_CFG2**.I2C_KSRC 选择 APB 时钟、系统时钟或 HRC。如果要修改时钟来源的配置，必须先关闭时钟来源设置，将 **RCU_CFG2**.I2C_KSRC 设置为 0 之后才能进行修改。

4. 3. 2. 9 IWDT时钟 (IWDTCLK)

IWDT 时钟只能由 LRC 提供，无论是通过配置字还是软件方式启动独立看门狗(IWDT)，都会强制启动 LRC，且无法关闭。

4. 3. 2. 10 微控制器输出时钟 (MCOCLK)

微控制器时钟输出功能允许将时钟信号输出到 MCO 引脚，不仅可以用于检查时钟频率，还可以当作其他装置或模块的输入时钟源。通过配置寄存器 **RCU_CFG**.MSW 选择要输出的时钟，并且可以根据需求通过 MPRE 将微控制器时钟输出进行分频，可以分频 2、4、8、16、32、64 或 128 倍。可选择以下时钟输出：

- LRC 时钟
- HRC 时钟
- HOSC 时钟
- PLL 时钟
- 系统时钟
- AHB 时钟
- APB 时钟
- ADC 时钟
- I2C 核心时钟
- ADC 核心时钟

4.4 特殊功能寄存器

4.4.1 寄存器列表

RCU 寄存器列表			
名称	偏移地址	类型	描述
RCU_CON	0000_H	R/W	时钟控制寄存器
RCU_CFG	0004_H	R/W	时钟配置寄存器
RCU_CFG1	0008_H	R/W	时钟配置寄存器 1
RCU_CFG2	$000C_H$	R/W	时钟配置寄存器 2
RCU_IER	0010_H	W1	RCU 中断开启寄存器
RCU_IDR	0014_H	W1	RCU 中断关闭寄存器
RCU_IVS	0018_H	R	RCU 中断功能有效状态寄存器
RCU_RIF	$001C_H$	R	RCU 原始中断状态寄存器
RCU_IFM	0020_H	R	RCU 中断标志位状态寄存器
RCU_ICR	0024_H	C_W1	RCU 中断清除寄存器
RCU_AHBRST	0030_H	R/W	AHB 外设复位控制寄存器
RCU_APB1RST	0034_H	R/W	APB1 外设复位控制寄存器
RCU_APB2RST	0038_H	R/W	APB2 外设复位控制寄存器
RCU_AHBEN	$003C_H$	R/W	AHB 外设时钟控制寄存器
RCU_APB1EN	0040_H	R/W	APB1 外设时钟控制寄存器
RCU_APB2EN	0044_H	R/W	APB2 外设时钟控制寄存器
RCU_AHBSL	0048_H	R/W	SLEEP 模式 AHB 外设时钟控制寄存器
RCU_APB1SL	$004C_H$	R/W	SLEEP 模式 APB1 外设时钟控制寄存器
RCU_APB2SL	0050_H	R/W	SLEEP 模式 APB2 外设时钟控制寄存器
RCU_LCON	0060_H	R/W	低速时钟控制寄存器
RCU_RSTF	0064_H	R/W	复位标志位寄存器

4. 4. 2 寄存器描述

4. 4. 2. 1 时钟控制寄存器 (RCU_CON)

时钟控制寄存器(RCU_CON)																												
偏移地址:0x00																												
复位值:0x0000 0003																												
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
							CSSON			PLLRDY	PLLON														HOSCBYP	HOSCRDY	HOSCON			HRCRDY	HRCON

—	Bit 31-25	—	—
CSSON	Bit 24	R/W	**时钟安全系统使能(CSS)** HOSC 时钟信号稳定就绪时(HOSCRDY=1)，可软件控制开启安全保护功能；当HOSC时钟信号尚未稳定(HOSCRDY = 0)，此位无法被开启，或硬件侦测到 HOSC 时钟失效时，硬件自动关闭此功能。 0:关闭时钟安全系统(关闭时钟侦测) 1:开启时钟安全系统(开启时钟侦测，必须HOSCRDY=1)
—	Bit 23-22	—	—
PLLRDY	Bit 21	R	**PLLCLK 时钟源，稳定状态标志位** 此位由硬件设置，其表示时钟源稳定状态 0: PLL 时钟信号，尚未稳定 1: PLL 时钟信号，已准备就绪
PLLON	Bit 20	R/W	**锁相环 PLL，PLLCLK 时钟源使能** 0:关闭 PLL 1:开启 PLL
—	Bit 19-7	—	—
HOSCBYP	Bit 6	R/W	**外部高速时钟振荡器，旁路模式使能** 此位只能在 HOSC 关闭时(HOSCON=0)，允许写入 0:关闭 HOSC 旁路模式(振荡器模式) 1:开启 HOSC 旁路模式(使用外部输入时钟源，通过 HOSCIN 引脚)
HOSCRDY	Bit 5	R	**HOSCCLK 时钟源，稳定状态标志位** 此位由硬件设置，其表示时钟源稳定状态

			0: HOSC 时钟信号，尚未稳定 1: HOSC 时钟信号，已准备就绪
HOSCON	Bit 4	R/W	**外部高速时钟振荡器，HOSCCLK 时钟源使能** 0:关闭 HOSC 1:开启 HOSC
—	Bit 3-2	—	—
HRCRDY	Bit 1	R	**HRCCLK 时钟源，稳定状态标志位** 此位由硬件设置，其表示时钟源稳定状态 0: HRC 时钟信号，尚未稳定 1: HRC 时钟信号，已准备就绪
HRCON	Bit 0	R/W	**内部高速 RC 振荡器，HRCCLK 时钟源使能** 0:关闭 HRC 1:开启 HRC

4. 4. 2. 2 时钟配置寄存器 (RCU_CFG)

时钟配置寄存器 (RCU_CFG)

偏移地址:0x04

复位值:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
—	MPRE<2:0>			MSW<3:0>				PLLSRC<1:0>		PREDIV<3:0>				—	—	—	PPRE<2:0>			HPRE<3:0>				—	—	SWS<2:0>			SW<2:0>		

—	Bit 31	—	—
MPRE	Bit 30-28	R/W	**MCO 微控制器时钟输出分频** 000: MCO 不分频 001: MCO 2 倍分频 010: MCO 4 倍分频 011: MCO 8 倍分频 100: MCO 16 倍分频 101: MCO 32 倍分频 110: MCO 64 倍分频 111: MCO 128 倍分频
MSW	Bit 27-24	R/W	**MCO 微控制器时钟输出选择** 0000:关闭 MCO 时钟输出，MCO 引脚没有时钟输出

			0001:选择输出 LRC 时钟(LRCCLK) 0010:保留 0011:选择输出 HRC 时钟(HRCCLK) 0100:选择输出 HOSC 时钟(HOSCCLK) 0101:保留 0110:选择输出 PLL 时钟(PLLCLK) 0111:保留 1000:选择输出系统时钟(SYSCLK) 1001:选择输出 AHB 时钟(HCLK) 1010:选择输出 APB 时钟(PCLK) 1011:选择输出 ADC 时钟(ADCCLK) 1100:选择输出 I2C 核心时钟 1101:选择输出 ADC 核心时钟 1110:选择输出定时器核心时钟 1111:保留
PLLSRC	Bit23-22	R/W	**选择 PLL 输入时钟来源** 当 PLL 关闭(PLLON=0)时，允许改变设置。 00:选择 HRC/PREDIV 为 PLL 输入时钟源 01:选择 HOSC/PREDIV 为 PLL 输入时钟源 10:保留 11:保留
PREDIV	Bit 21-18	R/W	**PLL 输入时钟预分频** 当 PLL 关闭(PLLON=0)时，允许改变分频设置。 PLL 输入时钟来源由 PLLSRC 选择，需将 PLL 输入频率配置为 4MHz 0000: PLL 输入时钟不分频 0001: PLL 输入时钟 2 倍分频 0010: PLL 输入时钟 3 倍分频 0011: PLL 输入时钟 4 倍分频 0100: PLL 输入时钟 5 倍分频 0101: PLL 输入时钟 6 倍分频 0110: PLL 输入时钟 7 倍分频 0111: PLL 输入时钟 8 倍分频 1000: PLL 输入时钟 9 倍分频 1001: PLL 输入时钟 10 倍分频 1010: PLL 输入时钟 11 倍分频 1011: PLL 输入时钟 12 倍分频

			1100: PLL 输入时钟 13 倍分频 1101: PLL 输入时钟 14 倍分频 1110: PLL 输入时钟 15 倍分频 1111: PLL 输入时钟 16 倍分频
—	Bit 17-15	—	—
PPRE	Bit 14-12	R/W	**APB 时钟(PCLK)预分频** 0xx: HCLK 不分频 100: HCLK 2 倍分频 101: HCLK 4 倍分频 110: HCLK 8 倍分频 111: HCLK 16 倍分频
HPRE	Bit 11-8	R/W	**AHB 时钟(HCLK)预分频** 0xxx: SYSCLK 不分频 1000: SYSCLK 2 倍分频 1001: SYSCLK 4 倍分频 1010: SYSCLK 8 倍分频 1011: SYSCLK 16 倍分频 1100: SYSCLK 64 倍分频 1101: SYSCLK 128 倍分频 1110: SYSCLK 256 倍分频 1111: SYSCLK 512 倍分频
—	Bit 7-6	—	—
SWS	Bit 5-3	R	**系统时钟选择状态** 此位状态由硬件控制，显示当前系统使用那种时钟来源 000:系统时钟为 HRCCLK 001:系统时钟为 HOSCCLK 010:系统时钟为 PLLCLK 011:系统时钟为 LRCCLK 1xx:保留
SW	Bit 2-0	R/W	**选择系统时钟(SYSCLK)来源** 000: 选择 HRCCLK 作为系统时钟 001: 选择 HOSCCLK 作为系统时钟 010: 选择 PLLCLK 作为系统时钟 011: 选择 LRCCLK 作为系统时钟 1xx:保留(等同于选择 HRCCLK 作为系统时钟)

4. 4. 2. 3 时钟配置寄存器 1 (RCU_CFG1)

时钟配置寄存器 1 (RCU_CFG1)																												
偏移地址:0x08																												
复位值:0x0000 0100																												
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
—	FK<18:0>																			FN<7:0>								FM<3:0>			

—	Bit 31	—	—
FK	Bit 30-12	R/W	**PLL VCO 频率小数倍率值 (Fractional)** 当 PLL 关闭(PLLON = 0)时，允许改变设置。 此数值设定 PLL f_{VCO} 频率输出为 PLL 输入频率乘上(FN + FK/ 2^{19})倍的小数值。 计算方式请参考 PLL 章节说明。
FN	Bit 11-4	R/W	**PLL VCO 频率整数倍率值 (Integer)** 当 PLL 关闭(PLLON = 0)时，允许改变设置。 此数值设定 PLL f_{VCO} 频率输出为 PLL 输入频率乘上 FN 倍的整数值。FN 最小为 16 倍，最大为 255 倍。 NOTE：FN 数值不可以设定低于 16，当设定值低于 16 倍时，硬件固定为 16。 计算方式请参考 PLL 章节说明。
FM	Bit 3-0	R/W	**PLL 时钟输出分频倍率** 当 PLL 关闭(PLLON = 0)时，允许改变设置。 此数值 PLL 时钟输出频率为，f_{VCO} 频率除以 FM 倍。 0000: 2 倍 0001: 4 倍 0010: 6 倍 0011: 8 倍 0100: 12 倍 0101: 16 倍 0110: 24 倍 0111: 32 倍 1000: 48 倍 其他: 保留 计算方式请参考 PLL 章节说明。

4. 4. 2. 4 时钟配置寄存器 2 (RCU_CFG2)

时钟配置寄存器 2 (RCU_CFG2)																												
偏移地址:0x0C																												
复位值:0x1000 FF80																												
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
SYSFREQ<7:0>								—	—	—	—	—	—	—	—	PLLRDYCNT<8:0>									—	—	HOSC32EN	I2C_KSRC<1:0>		ADC_KSRC<1:0>	

SYSFREQ	Bit 31-24	R/W	**当前系统时钟频率** 用于纪录当前系统时钟的频率数值，初始数值为 4，且不得填入小于 4 的数值，若填入的入值小于 4，则 SYSFREQ 的数值自动带入初始数值 4。当系频率改为 8 MHz，需修改 SYSFREQ 的数值为 8，其余频率依此类推。
—	Bit 23-16	—	—
PLLRDYCNT	Bit 15-7	R/W	**PLL 稳定时间设置** 开启 PLL 之后，使用 PLL 参考时钟计数 PLL 的稳定时间。
—	Bit 6-5	—	—
HOSC32EN	Bit 4	R/W	**HOSC 32 倍分频开关** 0: 关闭 HOSC 32 倍分频 1: 开启 HOSC 32 倍分频
I2C_KSRC	Bit 3-2	R/W	**选择 I2C 核心时钟来源** 00: 关闭 I2C 核心时钟 01: 选择 APBCLK 作为 I2C 核心时钟 10: 选择 SYSCLK 作为 I2C 核心时钟 11: 选择 HRCCLK 作为 I2C 核心时钟 NOTE: 必须先将 I2C_KSRC 设定为 0，才能重新配置 I2C 核心时钟来源。
ADC_KSRC	Bit 1-0	R/W	**选择 ADC 核心时钟来源** 00: 关闭 ADC 核心时钟 01: 选择 PLLCLK 作为 ADC 核心时钟 10: 选择 SYSCLK 作为 ADC 核心时钟 11: 选择 AHBCLK 作为 ADC 核心时钟 NOTE: 必须先将 ADC_KSRC 设定为 0，才能重新配置 ADC 核心时钟来源。

4. 4. 2. 5 RCU中断开启寄存器 (RCU_IER)

RCU 中断开启寄存器 (RCU_IER)																												
偏移地址:0x10																												
复位值:0x0000 0000																												
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	CSSHOSC	—	—	PLLRDY	—	HOSCRDY	HRCRDY	—	LRCRDY

—	Bit 31-9	—	—
CSSHOSC	Bit 8	W1	**开启 HOSC 时钟安全系统中断功能** 此位设置时，开启中断功能，当 HOSC 时钟检测功能开启，而 HOSC 失效时发生中断
—	Bit 7-6	—	—
PLLRDY	Bit 5	W1	**开启 PLL 时钟源稳定中断功能** 此位设置时，开启中断功能，当 PLL 时钟稳定后发生中断
—	Bit 4	—	—
HOSCRDY	Bit 3	W1	**开启 HOSC 时钟源稳定中断功能** 此位设置时，开启中断功能，当 HOSC 时钟稳定后发生中断
HRCRDY	Bit 2	W1	**开启 HRC 时钟源稳定中断功能** 此位设置时，开启中断功能，当 HRC 时钟稳定后发生中断
—	Bit 1	—	—
LRCRDY	Bit 0	W1	**开启 LRC 时钟源稳定中断功能** 此位设置时，开启中断功能，当 LRC 时钟稳定后发生中断

4. 4. 2. 6 RCU中断关闭寄存器 (RCU_IDR)

RCU 中断关闭寄存器 (RCU_IDR)																												
偏移地址:0x14																												
复位值:0x0000 0000																												
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																							CSSHOSC			PLLRDY		HOSCRDY	HRCRDY		LRCRDY

—	Bit 31-9	—	—
CSSHOSC	Bit 8	W1	**关闭 HOSC 时钟安全系统中断功能** 此位设置时，关闭 HOSC 时钟安全系统中断功能
—	Bit 7-6	—	—
PLLRDY	Bit 5	W1	**关闭 PLL 时钟源稳定中断功能** 此位设置时，关闭 PLL 时钟稳定中断功能
—	Bit 4	—	—
HOSCRDY	Bit 3	W1	**关闭 HOSC 时钟源稳定中断功能** 此位设置时，关闭 HOSC 时钟稳定中断功能
HRCRDY	Bit 2	W1	**关闭 HRC 时钟源稳定中断功能** 此位设置时，关闭 HRC 时钟稳定中断功能
—	Bit 1	—	—
LRCRDY	Bit 0	W1	**关闭 LRC 时钟源稳定中断功能** 此位设置时，关闭 LRC 时钟稳定中断功能

4. 4. 2. 7 RCU中断功能有效状态寄存器 (RCU_IVS)

RCU 中断功能有效状态寄存器 (RCU_IVS)																												
偏移地址:0x18																												
复位值:0x0000 0000																												
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																							CSSHOSC			PLLRDY		HOSCRDY	HRCRDY		LRCRDY

—	Bit 31-9	—	—
CSSHOSC	Bit 8	R	**HOSC 时钟安全系统中断功能开启/关闭状态** 0:中断功能处于关闭状态 1:中断功能处于开启状态
—	Bit 7-6	—	—
PLLRDY	Bit 5	R	**PLL 时钟源稳定中断功能开启/关闭状态** 0:中断功能处于关闭状态 1:中断功能处于开启状态
—	Bit 4	—	—
HOSCRDY	Bit 3	R	**HOSC 时钟源稳定中断功能开启/关闭状态** 0:中断功能处于关闭状态 1:中断功能处于开启状态
HRCRDY	Bit 2	R	**HRC 时钟源稳定中断功能开启/关闭状态** 0:中断功能处于关闭状态 1:中断功能处于开启状态
—	Bit 1	—	—
LRCRDY	Bit 0	R	**LRC 时钟源稳定中断功能开启/关闭状态** 0:中断功能处于关闭状态 1:中断功能处于开启状态

RCU_IVS 寄存器，是实时反映系统配置 RCU_IER 与 RCU_IDR 的中断开启状态。此寄存器状态是将 RCU_IER 与 RCU_IDR 进行硬件运算，公式如下:RCU_IVS = RCU_IER & !RCU_IDR

4. 4. 2. 8 RCU原始中断状态寄存器 (RCU_RIF)

RCU 原始中断状态寄存器 (RCU_RIF)																												
偏移地址:0x1C																												
复位值:0x0000 0000																												
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	CSSHOSC	—	—	PLLRDY	—	HOSCRDY	HRCRDY	—	LRCRDY

—	Bit 31-9	—	—
CSSHOSC	Bit 8	W1	**HOSC时钟安全系统，原始中断状态** 0:无发生中断 1:已发生中断
—	Bit 7-6	—	—
PLLRDY	Bit 5	W1	**PLL 时钟源稳定时，原始中断状态** 0:无发生中断 1:已发生中断
—	Bit 4	—	—
HOSCRDY	Bit 3	R	**HOSC 时钟源稳定时，原始中断状态** 0:无发生中断 1:已发生中断
HRCRDY	Bit 2	R	**HRC 时钟源稳定时，原始中断状态** 0:无发生中断 1:已发生中断
—	Bit 1	—	—
LRCRDY	Bit 0	R	**LRC 时钟源稳定时，原始中断状态** 0:无发生中断 1:已发生中断

4. 4. 2. 9 RCU中断标志位状态寄存器 (RCU_IFM)

RCU 中断标志位状态寄存器 (RCU_IFM)																												
偏移地址:0x20																												
复位值:0x0000 0000																												
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	CSSHOSC	—	—	PLLRDY	—	HOSCRDY	HRCRDY	—	LRCRDY

—	Bit 31-9	—	—
CSSHOSC	Bit 8	R	**HOSC 时钟安全系统，标志位中断状态** 0:无发生中断 1:已发生中断
—	Bit 7-6	—	—
PLLRDY	Bit 5	R	**PLL 时钟源稳定时，标志位中断状态** 0:无发生中断 1:已发生中断
—	Bit 4	—	—
HOSCRDY	Bit 3	R	**HOSC 时钟源稳定时，标志位中断状态** 0:无发生中断 1:已发生中断
HRCRDY	Bit 2	R	**HRC 时钟源稳定时，标志位中断状态** 0:无发生中断 1:已发生中断
—	Bit 1	—	—
LRCRDY	Bit 0	R	**LRC 时钟源稳定时，标志位中断状态** 0:无发生中断 1:已发生中断

RCU_IFM 寄存器，是滤除已关闭中断功能的中断事件，只关注开启中断功能的事件。此寄存器状态是将 RCU_RIF 与 RCU_IVS 进行硬件运算，公式如下:RCU_IFM = RCU_RIF & RCU_IVS

4. 4. 2. 10 RCU中断清除寄存器 (RCU_ICR)

RCU 中断清除寄存器 (RCU_ICR)																												
偏移地址:0x24																												
复位值:0x0000 0000																												
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																							CSSHOSC			PLLRDY		HOSCRDY	HRCRDY		LRCRDY

—	Bit 31-9	—	—
CSSHOSC	Bit 8	C_W1	**清除 HOSC 时钟安全系统中断状态** 此位设置时，清除中断状态(RCU_RIF 与 RCU_IFM)
—	Bit 7-6	—	—
PLLRDY	Bit 5	C_W1	**清除 PLL 时钟源稳定后中断状态** 此位设置时，清除中断状态(RCU_RIF 与 RCU_IFM)
—	Bit 4	—	—
HOSCRDY	Bit 3	C_W1	**清除 HOSC 时钟源稳定后中断状态** 此位设置时，清除中断状态(RCU_RIF 与 RCU_IFM)
HRCRDY	Bit 2	C_W1	**清除 HRC 时钟源稳定后中断状态** 此位设置时，清除中断状态(RCU_RIF 与 RCU_IFM)
—	Bit 1	—	—
LRCRDY	Bit 0	C_W1	**清除 LRC 时钟源稳定后中断状态** 此位设置时，清除中断状态(RCU_RIF 与 RCU_IFM)

RCU_ICR 寄存器设置时，将清除 RCU_RIF 与 RCU_IFM 中断标志位状态；此设置不影响中断 RCU_IER、RCU_IDR 与 RCU_IVS 寄存器，只清除标志位状态 RCU_RIF 与 RCU_IFM。此寄存器通过硬件清除中断，公式如下:

RCU_RIF = RCU_RIF & !RCU_ICR

4. 4. 2. 11 AHB外设复位控制寄存器 (RCU_AHBRST)

AHB 外设复位控制寄存器(RCU_AHBRST)																														
偏移地址:0x30																														
复位值:0x0000 0000																														
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
							ADCEN						GPCEN	GPBEN	GPAEN	CALCEN		SVAEN	IIREN										DMAMUXEN		DMA1EN

—	Bit 31-25	—	—
ADCEN	Bit 24	R/W	**ADC 复位需求** 0: 取消复位需求 1: 提交复位需求
—	Bit 23-19	—	—
GPCEN	Bit 18	R/W	**GPIOC 复位需求** 0: 取消复位需求 1: 提交复位需求
GPBEN	Bit 17	R/W	**GPIOB 复位需求** 0: 取消复位需求 1: 提交复位需求
GPAEN	Bit 16	R/W	**GPIOA 复位需求** 0: 取消复位需求 1: 提交复位需求
CALCEN	Bit 15	R/W	**CALC 复位需求** 0: 取消复位需求 1: 提交复位需求
—	Bit 14	—	—
SVAEN	Bit 13	R/W	**SVA 复位需求** 0: 取消复位需求 1: 提交复位需求
IIREN	Bit 12	R/W	**IIR 复位需求** 0: 取消复位需求 1: 提交复位需求
—	Bit 11-3	—	—
DMAMUXEN	Bit 2	R/W	**DMA MUX 复位需求** 0: 取消复位需求 1: 提交复位需求

—	Bit 1	—	—
DMA1EN	Bit 0	R/W	**DMA1 复位需求** 0: 取消复位需求 1: 提交复位需求

4. 4. 2. 12 APB1 外设复位控制寄存器 (RCU_APB1RST)

APB1 外设复位控制寄存器(RCU_APB1RST)

偏移地址:0x34

复位值:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
—	—	—	—	—	—	CAN1EN	—	—	—	I2C1EN	—	—	—	UART2EN	—	—	—	—	—	WWDTEN	—	—	—	—	—	—	BS16T1EN	—	—	GP32C4T2EN	GP32C4T1EN

—	Bit 31-26	—	—
CAN1EN	Bit 25	R/W	**CAN1 复位需求** 0: 取消复位需求 1: 提交复位需求
—	Bit 24-22	—	—
I2C1EN	Bit 21	R/W	**I2C1 复位需求** 0: 取消复位需求 1: 提交复位需求
—	Bit 20-18	—	—
UART2EN	Bit 17	R/W	**UART2 复位需求** 0: 取消复位需求 1: 提交复位需求
—	Bit 16-12	—	—
WWDTEN	Bit 11	R/W	**WWDT 复位需求** 0: 取消复位需求 1: 提交复位需求
—	Bit 10-5	—	—
BS16T1EN	Bit 4	R/W	**BS16T1 复位需求** 0: 取消复位需求 1: 提交复位需求
—	Bit 3-2	—	—
GP32C4T2EN	Bit 1	R/W	**GP32C4T2EN 复位需求** 0: 取消复位需求

			1: 提交复位需求
GP32C4T1EN	Bit 0	R/W	**GP32C4T1 复位需求** 0: 取消复位需求 1: 提交复位需求

4. 4. 2. 13 APB2 外设复位控制寄存器 (RCU_APB2RST)

APB2 外设复位控制寄存器(RCU_APB2RST)

偏移地址:0x38

复位值:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
—	—	—	—	—	—	—	VRESEN	CMPEN	—	—	—	—	—	GP16C2T2EN	GP16C2T1EN	—	UART1EN	AD16C6T2EN	SPI1EN	AD16C6T1EN	—	—	—	—	—	OPAMPEN	—	—	—	—	—

—	Bit 31-24	—	—
VRESEN	Bit 24	R/W	**VRES 复位需求** 0: 取消复位需求 1: 提交复位需求
CMPEN	Bit 23	R/W	**CMP 复位需求** 0: 取消复位需求 1: 提交复位需求
—	Bit 22-18	—	—
GP16C2T2EN	Bit 17	R/W	**GP16C2T2 复位需求** 0: 取消复位需求 1: 提交复位需求
GP16C2T1EN	Bit 16	R/W	**GP16C2T1 复位需求** 0: 取消复位需求 1: 提交复位需求
—	Bit 15	—	—
UART1EN	Bit 14	R/W	**UART1 复位需求** 0: 取消复位需求 1: 提交复位需求
AD16C6T2EN	Bit 13	R/W	**AD16C6T2 复位需求** 0: 取消复位需求 1: 提交复位需求
SPI1EN	Bit 12	R/W	**SPI1 复位需求** 0: 取消复位需求

			1: 提交复位需求
AD16C6T1EN	Bit 11	R/W	**AD16C6T1 复位需求** 0: 取消复位需求 1: 提交复位需求
—	Bit 10-6	—	—
OPAMPEN	Bit 5	R/W	**OPAMP 复位需求** 0: 取消复位需求 1: 提交复位需求
—	Bit 4-0	—	—

4. 4. 2. 14 AHB外设时钟控制寄存器(RCU_AHBEN)

AHB 外设时钟控制寄存器(RCU_AHBEN)

偏移地址:0x3C

复位值:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
							ADCEN						GPCEN	GPBEN	GPAEN	CALCEN		SVAEN	IIREN											DMAMUXEN		DMA1EN

—	Bit 31-25	—	—
ADCEN	Bit 24	R/W	**ADC 时钟开关** 0: 关闭时钟 1: 开启时钟
—	Bit 23-19	—	—
GPCEN	Bit 18	R/W	**GPIOC 时钟开关** 0: 关闭时钟 1: 开启时钟
GPBEN	Bit 17	R/W	**GPIOB 时钟开关** 0: 关闭时钟 1:开启时钟
GPAEN	Bit 16	R/W	**GPIOA 时钟开关** 0: 关闭时钟 1: 开启时钟
CALCEN	Bit 15	R/W	**CALC 时钟开关** 0: 关闭时钟 1: 开启时钟
—	Bit 14	—	—

SVAEN	Bit 13	R/W	**SVA 时钟开关** 0: 关闭时钟 1: 开启时钟
IIREN	Bit 12	R/W	**IIR 时钟开关** 0: 关闭时钟 1: 开启时钟
—	Bit 11-3	—	—
DMAMUXEN	Bit 2	R/W	**DMA MUX 时钟开关** 0: 关闭时钟 1: 开启时钟
—	Bit 1	—	—
DMA1EN	Bit 0	R/W	**DMA1 时钟开关** 0: 关闭时钟 1: 开启时钟

4. 4. 2. 15 APB1 外设时钟控制寄存器 (RCU_APB1EN)

APB1 外设时钟控制寄存器(RCU_APB1EN)

偏移地址:0x40

复位值:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
—	—	—	—	—	—	CAN1EN	—	—	—	I2C1EN	—	—	—	UART2EN	—	—	—	—	—	WWDTEN	—	—	—	—	—	—	BS16T1EN	—	—	GP32C4T2EN	GP32C4T1EN

—	Bit 31-26	—	—
CAN1EN	Bit 25	R/W	**CAN1 时钟开关** 0: 关闭时钟 1: 开启时钟
—	Bit 24-22	—	—
I2C1EN	Bit 21	R/W	**I2C1 时钟开关** 0: 关闭时钟 1: 开启时钟
—	Bit 20-18	—	—
UART2EN	Bit 17	R/W	**UART2 时钟开关** 0: 关闭时钟 1: 开启时钟
—	Bit 16-12	—	—

WWDTEN	Bit 11	R/W	**WWDT 时钟开关** 0: 关闭时钟 1: 开启时钟
—	Bit 10-5	—	—
BS16T1EN	Bit 4	R/W	**BS16T1 时钟开关** 0: 关闭时钟 1: 开启时钟
—	Bit 3-2	—	—
GP32C4T2EN	Bit 1	R/W	**GP32C4T2 时钟开关** 0: 关闭时钟 1: 开启时钟
GP32C4T1EN	Bit 0	R/W	**GP32C4T1 时钟开关** 0: 关闭时钟 1: 开启时钟

4. 4. 2. 16 APB2 外设时钟控制寄存器 (RCU_APB2EN)

APB2 外设时钟控制寄存器(RCU_APB2EN)

偏移地址:0x44

复位值:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
—	—	—	—	—	—	—	VRESEN	CMPEN	—	—	—	—	—	GP16C2T2EN	GP16C2T1EN	—	UART1EN	AD16C6T2EN	SPI1EN	AD16C6T1EN	—	—	—	—	—	OPAMPEN	—	—	—	—	—

—	Bit 31-25	—	—
VRESEN	Bit 24	R/W	**VRES 时钟开关** 0: 关闭时钟 1: 开启时钟
CMPEN	Bit 23	R/W	**CMP 时钟开关** 0: 关闭时钟 1: 开启时钟
—	Bit 22-18	—	—
GP16C2T2EN	Bit 17	R/W	**GP16C2T2 时钟开关** 0: 关闭时钟 1: 开启时钟
GP16C2T1EN	Bit 16	R/W	**GP16C2T1 时钟开关** 0: 关闭时钟

			1: 开启时钟
—	Bit 15	—	—
UART1EN	Bit 14	R/W	**UART1 时钟开关** 0: 关闭时钟 1: 开启时钟
AD16C6T2EN	Bit 13	R/W	**AD16C6T2 时钟开关** 0: 关闭时钟 1: 开启时钟
SPI1EN	Bit 12	R/W	**SPI1 时钟开关** 0: 关闭时钟 1: 开启时钟
AD16C6T1EN	Bit 11	R/W	**AD16C6T1 时钟开关** 0: 关闭时钟 1: 开启时钟
—	Bit 10-6	—	—
OPAMPEN	Bit 5	R/W	**OPAMP 时钟开关** 0: 关闭时钟 1: 开启时钟
—	Bit 4-0	—	—

4. 4. 2. 17 SLEEP模式AHB外设时钟控制寄存器 (RCU_AHBSL)

SLEEP 模式 AHB 外设时钟控制寄存器 (RCU_AHBSL)																														
偏移地址:0x48																														
复位值:0x0000 0000																														
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
							ADCEN						GPCEN	GPBEN	GPAEN	CALCEN		SVAEN	IIREN										DMAMUXEN		DMA1EN

—	Bit 31-25	—	—
ADCEN	Bit 24	R/W	**SLEEP 模式时，ADC 时钟开关** 0: 关闭时钟 1: 开启时钟
—	Bit 23-19	—	—
GPCEN	Bit 18	R/W	**SLEEP 模式时，GPIOC 时钟开关** 0: 关闭时钟 1: 开启时钟
GPBEN	Bit 17	R/W	**SLEEP 模式时，GPIOB 时钟开关** 0: 关闭时钟 1: 开启时钟
GPAEN	Bit 16	R/W	**SLEEP 模式时，GPIOA 时钟开关** 0: 关闭时钟 1: 开启时钟
CALCEN	Bit 15	R/W	**SLEEP 模式时，CALC 时钟开关** 0: 关闭时钟 1: 开启时钟
—	Bit 14	—	—
SVAEN	Bit 13	R/W	**SLEEP 模式时，SVA 时钟开关** 0: 关闭时钟 1: 开启时钟
IIREN	Bit 12	R/W	**SLEEP 模式时，IIR 时钟开关** 0: 关闭时钟 1: 开启时钟
—	Bit 11-3	—	—
DMAMUXEN	Bit 2	R/W	**SLEEP 模式时，DMA MUX 时钟开关** 0: 关闭时钟 1: 开启时钟

—	Bit 1	—	—
DMA1EN	Bit 0	R/W	**SLEEP 模式时，DMA1 时钟开关** 0: 关闭时钟 1: 开启时钟

4. 4. 2. 18 SLEEP模式APB1 外设时钟控制寄存器 (RCU_APB1SL)

SLEEP 模式 APB1 外设时钟控制寄存器 (RCU_APB1SL)

偏移地址:0x4C

复位值:0x0000 0000

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
						CAN1EN				I2C1EN				UART2EN						WWDTEN							BS16T1EN			GP32C4T2EN	GP32C4T1EN

—	Bit 31-26	—	—
CAN1EN	Bit 25	R/W	**SLEEP 模式时，CAN1 时钟开关** 0: 关闭时钟 1: 开启时钟
—	Bit 24-22	—	—
I2C1EN	Bit 21	R/W	**SLEEP 模式时，I2C1 时钟开关** 0: 关闭时钟 1: 开启时钟
—	Bit 20-18	—	—
UART2EN	Bit 17	R/W	**SLEEP 模式时，UART2 时钟开关** 0: 关闭时钟 1: 开启时钟
—	Bit 16-12	—	—
WWDTEN	Bit 11	R/W	**SLEEP 模式时，WWDT 时钟开关** 0: 关闭时钟 1: 开启时钟
—	Bit 10-5	—	—
BS16T1EN	Bit 4	R/W	**SLEEP 模式时，BS16T1 时钟开关** 0: 关闭时钟 1: 开启时钟
—	Bit 3-2	—	—
GP32C4T2EN	Bit 1	R/W	**SLEEP 模式时，GP32C4T2 时钟开关** 0: 关闭时钟

			1: 开启时钟
GP32C4T1EN	Bit 0	R/W	**SLEEP 模式时，GP32C4T1 时钟开关** 0: 关闭时钟 1: 开启时钟

4. 4. 2. 19 SLEEP模式APB2 外设时钟控制寄存器 (RCU_APB2SL)

SLEEP 模式 APB2 外设时钟控制寄存器 (RCU_APB2SL)

偏移地址:0x50

复位值:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
—	—	—	—	—	—	—	VRESEN	CMPEN	—	—	—	—	—	GP16C2T2EN	GP16C2T1EN	—	UART1EN	AD16C6T2EN	SPI1EN	AD16C6T1EN	—	—	—	—	—	OPAMPEN	—	—	—	—	—

—	Bit 31-25	—	—
VRESEN	Bit 24	R/W	**SLEEP 模式时，VRES 时钟开关** 0: 关闭时钟 1: 开启时钟
CMPEN	Bit 23	R/W	**SLEEP 模式时，CMP 时钟开关** 0: 关闭时钟 1: 开启时钟
—	Bit 22-18	—	—
GP16C2T2EN	Bit 17	R/W	**SLEEP 模式时，GP16C2T2 时钟开关** 0: 关闭时钟 1: 开启时钟
GP16C2T1EN	Bit 16	R/W	**SLEEP 模式时，GP16C2T1 时钟开关** 0: 关闭时钟 1: 开启时钟
—	Bit 15	—	—
UART1EN	Bit 14	R/W	**SLEEP 模式时，UART1 时钟开关** 0: 关闭时钟 1: 开启时钟
AD16C6T2EN	Bit 13	R/W	**SLEEP 模式时，AD16C6T2 时钟开关** 0: 关闭时钟 1: 开启时钟
SPI1EN	Bit 12	R/W	**SLEEP 模式时，SPI1 时钟开关** 0: 关闭时钟

			1: 开启时钟
AD16C6T1EN	Bit 11	R/W	**SLEEP 模式时，AD16C6T1 时钟开关** 0: 关闭时钟 1: 开启时钟
—	Bit 10-6	—	—
OPAMPEN	Bit 5	R/W	**SLEEP 模式时，OPAMP 时钟开关** 0: 关闭时钟 1: 开启时钟
—	Bit 4-0	—	—

4. 4. 2. 20 低速时钟控制寄存器 (RCU_LCON)

此寄存器只允许 32 位存取

低速时钟控制寄存器 (RCU_LCON)																												
偏移地址:0x60																												
复位值:0x0000 0000 (Power On Reset)																												
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
																														LRCRDY	LRCON

—	Bit 31-2	—	—
LRCRDY	Bit 1	R	**LRCCLK 时钟源，稳定状态标志位** 此位由硬件设置，其表示时钟源稳定状态 0: LRC 时钟信号，尚未稳定 1: LRC 时钟信号，已准备就绪
LRCON	Bit 0	R/W	**内部低速 RC 振荡器，LRCCLK 时钟源使能** 0:关闭 LRC 1:开启 LRC

4. 4. 2. 21 复位标志位寄存器 (RCU_RSTF)

此寄存器只允许 32 位存取

复位标志位寄存器 (RCU_RSTF)																												
偏移地址:0x64																												
复位值:0x0001 0000																												
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
—	—	—	—	—	—	—	—	LPRSTF	WWDTRSTF	IWDTRSTF	SWRSTF	OBLRSTF	NRSTF	BORRSTF	PORRSTF	CLRFLG	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

—	Bit 31-24	—	—
LPRSTF	Bit 23	R	**低功耗复位标志位** 此标志位表示，系统复位事件，由低功耗模式触发。从 STOP 模式唤醒会触发此标志位，但系统不会复位。
WWDTRSTF	Bit 22	R	**WWDT 复位标志位** 此标志位表示，系统复位事件，由 WWDT 计数触发
IWDTRSTF	Bit 21	R	**IWDT 复位标志位** 此标志位表示，系统复位事件，由 IWDT 计数触发
SWRSTF	Bit 20	R	**软件复位标志位** 此标志位表示，系统复位事件，由软件控制触发
OBLRSTF	Bit 19	R	**配置字重载复位标志位** 此标志位表示，系统复位事件，由配置字重载时触发
NRSTF	Bit 18	R	**NRST 外部引脚复位标志位** 此标志位表示，系统复位事件，由外部 NRST 引脚触发
BORRSTF	Bit 17	R	**BOR 复位标志位** 此标志位表示，系统复位事件，由电源下电达到 BOR 所设定的等级，触发复位
PORRSTF	Bit 16	R	**POR/PDR 复位标志位** 此标志位表示，系统复位事件，由电源系统上电或下电触发。此标志位初始状态必定为 1，如系统进入低功耗模式前清除此标志位，当由低

			功耗唤醒后，此标志位应该为 0；因此可由此判别系统是否重新上电。
CLRFLG	Bit 15	C_W1	**清除复位标志位** 此位设置时，清除所有复位标志位
—	Bit 14-0	—	—

第5章 闪存控制器 (FLASH)

5. 1 概述

嵌入式闪存的容量视芯片型号而定，最高可达 128KB，供用户存放应用程序(Application Code)或储存数据。使用闪存控制器可通过在线系统编程器(ISP)、SWD、Bootrom 或内部闪存程序等方式，对已焊接在 PCB 版上的芯片进行数据数据的修改。

5. 2 特性

- 程序区大小依芯片型号而定，最大支持 128KB。
- 闪存操作最小单位
  - 以 32 Bit 为单位进行编程(Program)，每次编程耗时约 25us。
  - 以 512 Byte 为单为进行页擦除，每次擦除耗时约 2ms。
- 支持擦除(Erase)程序区内未受保护的区域。
- 支持配置 3 种程序区保护。
  - 以页(Page)为单位配置用户代码读出保护(UCRP)，最多支持 2 组区间保护。
  - 以整个程序区为单位配置读出保护(RP)。
  - 以页(Page)为单位配置写保护(WP)，最多支持 2 组区间保护。
- 支持最多配置 3 个读取等待周期。
  - 系统频率不超过 24MHz 时，配置 0 个等待周期。
  - 系统频率超过 24MHz，但不超过 48MHz 时，配置 1 个等待周期。
  - 系统频率超过 48MHz，但不超过 72MHz 时，配置 2 个等待周期。
- 支持以 4K Byte 为单位配置闪存内部重映射(Remap)。
- 支持用户开启闪存读取缓存，减少执行循环时所需要读取闪存的次数。

5.3 闪存结构

闪存分为三个区块：程序区(Main Block)、系统内存区(System Memory)和信息区(Information Block)。其详细描述如下：

- 程序区:大小依芯片型号而定，最大支持 128KB。其内存位置为 0x0800_0000 至 0x0801_FFFF。程序区内的最小单位为页(Page)，每一页大小为 512 Byte，最多支援 256 个页。
- 系统内存区:大小为 4K Byte，其内存位置为 0x1FFF_0000 至 0x1FFF_0FFF。共分为 8 个页(Page)，每个页的大小以 512 Byte 为单位。此区主要用来存放芯片下载程序(Bootrom)。此区不开放用户进行读取与修改，芯片下载程序会在工厂测试完毕以后刻录至此区。
- 信息区:大小为 4KB，其内存位置为 0x1FFF_E000 至 0x1FFF_EFFF。信息区内的最小单位为页(Page)，最多支持 8 个页，全部数据只读，程序不可擦写，主要功能如下：
  - 用户配置字:位于信息区页 0 至页 3，此区主要存闪存保护设定与用户校准信息。此区的内容仅程序烧录时可以修改，程序运行时不能擦写。
  - 系统配置字:位于信息区页 4 至页 7，此区主要存放芯片校准信息，所有校准信息皆会在芯片出厂时针对芯片的特性填入相对应的校准值。为确保芯片的稳定性与安全性，此区的内容仅开放读取。

名称	页码	实体位置
用户配置字	页 0	0x1FFF_E000 ~ 0x1FFF_E1FF
	页 1	0x1FFF_E200 ~ 0x1FFF_E3FF
	页 2	0x1FFF_E400 ~ 0x1FFF_E5FF
	页 3	0x1FFF_E600 ~ 0x1FFF_E7FF
系统配置字	页 4	0x1FFF_E800 ~ 0x1FFF_E9FF
	页 5	0x1FFF_EA00 ~ 0x1FFF_EBFF
	页 6	0x1FFF_EC00 ~ 0x1FFF_EDFF
	页 7	0x1FFF_EE00 ~ 0x1FFF_EFFF

5.4 功能描述

下列章节将针对闪存控制器的基本功能进行描述。

5.4.1 用户配置字

用户配置字包含信息区页 0~页 3，每一页存放的内容如下列章节所示。

5.4.1.1 信息区页 0

此区主要存放闪存程序区的用户代码读出保护(UCRP)。启用保护后，保护区内仅允许读取指令，不允许读取数据，同时保护区内也支持写保护。当对此页进行擦除时，会被视为清除 UCRP 保护，硬件会自动清除包含 UCRP0 与 UCRP1 所设定的保护区内的数据。信息区页 0 无法通过闪存的编程指令修改保护设定，只能通过 UCRP 保护设定流程进行配置。如欲了解有关用户代码读出保护的配置方式与功能，请参阅程序区保护的描述。

配置字 UCRP0 保护																															
偏移地址:0x000																															
复位值:0xFFFF FFFF																															
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
UCRP0<31:0>																															

UCRP0	Bits 31-0	**闪存配置字用户代码读出保护设定 0** 程序区用户代码读出保护设定，以页为单位进行保护。 – UCRP0[31]为保护开关，数值为 0 代表开启保护。 – UCRP0[23:16]为保护终止页。 – UCRP0[7:0]为保护起始页。

配置字 UCRP0 保护取反																															
偏移地址:0x004																															
复位值:0xFFFF FFFF																															
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
UCRP0REV <31:0>																															

UCRP0REV	Bits 31-0	**闪存配置字用户代码读出保护设定 0 取反值** 存放 FC_UCRP0 的取反数值。当取反不满足，且 FC_UCRP0 与 FC_UCRP0REV 的数值不为 0xFFFFFFFF 时自动开启程序区全区的 UCRP 保护。

配置字 UCRP1 保护																															
偏移地址:0x008																															
复位值:0xFFFF FFFF																															
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
UCRP1<31:0>																															

UCRP1	Bits 31-0	**闪存配置字用户代码读出保护设定 1** 程序区用户代码读出保护设定，以页为单位进行保护。 – UCRP1[31]为保护开关，数值为 0 代表开启保护。 – UCRP1[23:16]为保护终止页。 – UCRP1[7:0]为保护起始页。

配置字 UCRP1 保护取反																															
偏移地址:0x00C																															
复位值:0xFFFF FFFF																															
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
UCRP1REV<31:0>																															

UCRP1REV	Bits 31-0	**闪存配置字用户代码读出保护设定 1 取反值** 存放 FC_UCRP1 的取反数值。当取反不满足，且 FC_UCRP1 与 FC_UCRP1REV 的数值不为 0xFFFFFFFF 时自动开启程序区全区的 UCRP 保护。

5. 4. 1. 2 信息区页 1

此页主要存放读保护设定，其主要功能为防止 ISP、Debug Port 和 Bootrom 读出与修改程序区，共分为三个等级。在经过 CP 测试后，芯片会将读保护设定填入 0xAAAA_AAAA，因此默认的读保护等级为 Lv0。如果对信息区页 1 进行擦除或者设定非 0xCCCC_CCCC 或 0xAAAA_AAAA 的数值，则读保护等级会提升至 Lv1 保护。此区无法藉由闪存的编程指令修改保护设定，需藉由读保护设定流程进行配置。有关读保护的配置方式和功能，请参阅程序区保护章节的描述。

配置字读保护等级																															
偏移地址:0x200																															
复位值:0xAAAA AAAA																															
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
RP<31:0>																															

RP	Bits 31-0	**读保护** 0xAAAA_AAAA:读保护等级 0(默认) 0xCCCC_CCCC:读保护等级 2 其他:读保护等级 1

5. 4. 1. 3 信息区页 2

此页存放多项保护设定，包括写保护设定、硬件映射设定、系统欠压复位设定以及 IWDT 设定。

写保护设定的主要目的是防止程序区内的保护区被擦除或覆盖。当对此页进行擦除时，仅会擦除保护设定，程序区内的数据仍会被保留。进行写保护设定时，无法使用闪存的编程指令进行修改，需通过写保护设定流程进行配置。

硬件映射设定主要用于配置闪存区域与 CPU 地址空间的映射关系。系统欠压复位设定用于确保系统在欠压情况下也能正常运行。IWDT 设定用于配置内部看门狗定时器的相关参数，以确保系统在运行过程中不会因为意外情况而死机。

此区域的保护设定均需通过相应的设定流程进行配置，无法使用闪存的编程指令进行修改。更多有关写保护、硬件映射、系统欠压复位和 IWDT 设定的信息，请参阅程序区保护的描述。

配置字 WP0 保护																															
偏移地址:0x400																															
复位值:0xFFFF FFFF																															
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
WP0<31:0>																															

WP0	Bits 31-0	**闪存配置字写保护设定 0** 程序区写保护设定，以页为单位进行保护。 – WP0[31]为保护开关，数值为 0 代表开启保护。 – WP0[23:16]为保护终止页。 – WP0[7:0]为保护起始页。

配置字 WP0 保护取反																															
偏移地址:0x404																															
复位值:0xFFFF FFFF																															
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
WP0REV<31:0>																															

WP0REV	Bits 31-0	**闪存配置字写保护设定 0 取反值** 存放 FC_WP0 的取反数值。当取反不满足，且 FC_WP0 与 FC_WP0REV 的数值不为 0xFFFFFFFF 时自动开启程序区

		全区的 WP 保护。

配置字 WP1 保护																													
偏移地址:0x408																													
复位值:0xFFFF FFFF																													
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
WP1<31:0>																													

WP1	Bits 31-0	**闪存配置字写保护设定 1** 程序区写保护设定，以页为单位进行保护。 – WP1[31]为保护开关，数值为 0 代表开启保护。 – WP1[23:16]为保护终止页。 – WP1[7:0]为保护起始页。

配置字 WP1 保护取反																													
偏移地址:0x40C																													
复位值:0xFFFF FFFF																													
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
WP1REV<31:0>																													

WP1REV	Bits 31-0	**闪存配置字写保护设定 1 取反值** 存放 FC_WP1 的取反数值。当取反不满足，且 FC_WP1 与 FC_WP1REV 的数值不为 0xFFFFFFFF 时自动开启程序区全区的 WP 保护。

当用户不再需要使用 Bootrom 时，可以将信息区页 2 位于 0x412 的 BOOTBYP 编程为 0xA5。当此位置被填入 0xA5 时，芯片会在开机时跳过 Bootrom 程序，直接映射至闪存程序区开始执行。如果信息区 0x412 的位置已经被编程为 0xA5，用户只能藉由程序区的程序，或使用 SWD 接口，对信息区页 2 进行擦除后，才有办法重新使用 Bootrom 更新程序区的程序。但在擦除的同时也会一并清除写保护设定。

配置字中的 SELECT 与 SEFBASE 主要提供映射信息给 Bootrom 参考。当 Bootrom 流程结束时，会依据 SELECT 与 SEFBASE 的设定决定要映射至程序区的哪一个位置继续执行。SELECT 提供程序重新映射至程序区、系统内存或是 SRAM 内继续执行的信息，而 SEFBASE 则是提供程序区内任意 4KB 位置的重映射信息。用户可将位于信息区 0x411 的 SELECT 配置为 0x0，同时依据需求将数值 0x0 到 0x1F(此芯片闪存最大支持 128KB)配置至信息区 0x410 的 SEFBASE。若设置为 0x1，则意味着映射至程序区第二个 4K Byte (0x0000_1000)的位置开始执行。若设置为 0x2，则意味着映射至程序区第三个 4K Byte (0x0000_2000)的位置开始执行，依此类推。若用户开启 Bootrom Bypass 功能时，则 SELECT 与 SEFBASE 的设定会直接反映在硬件映射(Hardware Remap)上，亦即系统在开机完毕后会依照 SELECT 与 SEFBASE 的设定进行映射。

配置字硬件映射选项

偏移地址:0x410

复位值:0xFFFF FFFF

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
—	—	—	—	—	—	—	—	BOOTBYP <7:0>>								SELECT <7:0>								SEFBASE <7:0>							

—	Bit 31-24	—
BOOTBYP	Bit 23-16	**硬件重映射选项** 0xA5:开机时跳过 Bootrom，并映射到主闪存开始执行。 其他:开机时从 Bootrom 开始执行。
SELECT	Bit 15-8	**软件重映射选项** 0x0:主闪存映射到 0x0000_0000。 0x1:System Memory 映射到 0x0000_0000。 0x2:SRAM 映射到 0x0000_0000。 0x3:保留。
SEFBASE	Bit 7-0	**主闪存重映射地址选择** 如果设置 0x1，则意味着从主闪存的第二个 4 KByte 开始执行，如果设置为 0x2，则意味着从主闪存的第三个 4 KByte，依此类推。SEFBASE 仅有在 BOOTBYP 被设定为 0 时才有效。

信息区页 2 也支持用户在系统开机时，同步开启 IWDT 与欠压复位(BOR)功能。

配置字系统选项																													
偏移地址:0x414																													
复位值:0xFFFF FFFF																													
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
								IWDTEN <7:0>>								BOREN <7:0>								BORLS <7:0>							

—	Bit 31-24	—
IWDTEN	Bit 23-16	**IWDT 开启选项** 0xA5:开机时自动开启 IWDT，计数周期 1 秒。 其他:开机时关闭 IWDT。
BOREN	Bit 15-8	**低电压复位开启选项** 0xA5:开机时自动开启 BOR，BOR 检测电压依据 BORLS 数值而定。（推荐配置为开启） 其他:开机时关闭 BOR。
BORLS	Bit 7-0	**低电压复位电压区间选择** 配置 BORLS 选择触发 BOR 复位的电压区间，详细电压区间请参阅 Datasheet 文档内电器特性章节描述。 000:— 001:BOR level 1 010:BOR level 2 011:BOR level 3 100:BOR level 4 101:BOR level 5 110:BOR level 6 111:BOR level 7

5. 4. 1. 4　　信息区页 3

此区大小固定为 512Byte，保留不能使用。

5. 4. 2　　系统配置字

系统配置字主要储存在信息区页 4 中，其中包含产品标识符以及唯一标识符等信息，这些信息会在制造时针对每一颗芯片进行调整，因此用户可以读取此区的信息，但无法对其进行修改。

5. 4. 2. 1 芯片产品识别码CHIPID

芯片产品标识符共 32 位，可用来区分芯片产品型号。

配置字芯片产品识别																													
偏移地址:0x8C0																													
复位值:																													
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
CHIPID<31:0>																													

CHIPID	Bits 31-0	芯片产品识别

5. 4. 2. 2 芯片唯一识别码UID

芯片唯一识别码共 128 位，每一颗芯片都是唯一的编码，用户可藉由此识别码实现下列功能:

- 终端产品序列号
- 通过特定的加密算法生成安全密钥

配置字芯片唯一标识码 0																															
偏移地址:0x8C4																															
复位值:																															
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
UID0<31:0>																															

UID0	Bits 31-0	**芯片唯一识别码 0**

配置字芯片唯一标识码 1																															
偏移地址:0x8C8																															
复位值:																															
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
UID1<31:0>																															

UID1	Bits 31-0	**芯片唯一识别码 1**

配置字芯片唯一标识码 2																															
偏移地址:0x8CC																															
复位值:																															
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
UID2<31:0>																															

UID2	Bits 31-0	**芯片唯一识别码 2**

配置字芯片唯一标识码 3																													
偏移地址:0x8D0																													
复位值:																													
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
UID3<31:0>																													

UID3	Bits 31-0	**芯片唯一识别码 3**

5. 4. 3 闪存操作解锁

闪存控制器初始会处于锁定状态，此时用户无法通过闪存控制器对闪存进行编程与擦除，以避免闪存内的数据被意外擦除或覆盖。若需要使用闪存的编程与擦除功能，用户需要先对闪存控制器进行解锁。解锁流程需连续输入两组解锁密钥，若输入错误的密钥或连续输入超过两组密钥时，闪存控制器会继续保持在上锁状态或重新锁定。当完成解锁流程后，用户可藉由读取闪存控制寄存器 **FC_STA** 的 CMDULK 位来判定是否解锁成功，当此位被设定为 1 时代表闪存控制器已经成功解锁。解锁流程如下所示：

- 检查 **FC_STA** 的 CMDULK 为 0，确认闪存控制器为锁定状态。
- 对 **FC_UL** 填入第一组密钥 0x0011_2233。
- 对 **FC_UL** 填入第二组密钥 0x5566_7788。
- 检查 **FC_STA** 的 CMDULK 为 1，确认闪存控制器解锁成功。

建议用户在完成闪存的编程与擦除流程后，重新对 **FC_UL** 寄存器填入 0x0000_0000，将闪存控制器重新上锁，以避免闪存数据被意外擦除或覆盖。

5. 4. 4 闪存保护

闪存的保护逻辑可防止程序区内的数据被读出或误修改。根据功能可区分为用户代码读出保护(UCRP)、读保护(RP)与写保护(WP)。信息区则依据不同区块有不同程度的保护，主要目的是防止系统校准信息被修改或程序区的保护设定被修改，进而影响系统运行。

5. 4. 4. 1 信息区保护

信息区可区分为 8 个页，每一页大小为 512Byte，每一页分别受到不同程度的保护。

- 页 0:存放用户代码读出保护设定。此页仅能读取，无法编程。若对此页进行页擦除，则会清除用户代码读出保护设定，导致用户代码读出保护区内的数据全部清除。开启用户代码读出保护需藉由特定流程进行，无法藉由编程信息区页 0 开启保护功能。
- 页 1:存放读保护设定。读保护需藉由特定流程开启，无法藉由编程信息区页 1 开启保护功能。有关读保护的说明，请参阅后续章节描述。
- 页 2:存放写保护设定。若对此页进行擦除，则会清除写保护设定，但保护区内的数据仍会保留。开启写保护需藉由特定流程进行，无法藉由编程信息区页 2 开启保护功能。此外，

此页还存放硬件映射设定、系统欠压复位(BOR)设定与 IWDT 设定。有关寄存器的说明，请参阅用户配置字

- 页 3:此页不受到任何保护，可存放用户数据。
- 页 4~页 7:存放系统校准信息，此区不开放给用户修改，但可读取此区的数据。此区的校准信息会在出厂前针对每一颗芯片的特性进行配置，这些校准信息会在系统开机时自动加载。

信息区页码	描述	位置	数据内容
页 0	UCRP 保护 0	0x1FFF_E000 ~ 0x1FFF_E003	用户定义
	UCRP 保护 0 取反	0x1FFF_E004 ~ 0x1FFF_E007	UCRP 保护 0 数值取反
	UCRP 保护 1	0x1FFF_E008 ~ 0x1FFF_E00B	用户定义
	UCRP 保护 1 取反	0x1FFF_E00C ~ 0x1FFF_E00F	UCRP 保护 1 数值取反
	保留	0x1FFF_E010 ~ 0x1FFF_E1FF	保留
页 1	RP 保护	0x1FFF_E200 ~ 0x1FFF_E203	用户定义
	保留	0x1FFF_E204 ~ 0x1FFF_E3FF	保留
页 2	WP 保护 0	0x1FFF_E400 ~ 0x1FFF_E403	用户定义
	WP 保护 0 取反	0x1FFF_E404 ~ 0x1FFF_E407	WP 保护 0 数值取反
	WP 保护 1	0x1FFF_E408 ~ 0x1FFF_E40B	用户定义
	WP 保护 1 取反	0x1FFF_E40C ~ 0x1FFF_E40F	WP 保护 1 数值取反
	主闪存重映射地址选择	0x1FFF_E410	用户定义
	软件重映射选项	0x1FFF_E411	用户定义
	硬件重映射选项	0x1FFF_E412	用户定义
	保留	0x1FFF_E413	保留
	欠压复位电压区间选择	0x1FFF_E414	用户定义
	欠压复位开启选项	0x1FFF_E415	用户定义
	IWDT 开启选项	0x1FFF_E416	用户定义
	保留	0x1FFF_E417 ~ 0x1FFF_E5FF	保留
页 3	保留	0x1FFF_E600 ~ 0x1FFF_E7FF	用户定义
页 4 ~ 页 7	系统配置字	0x1FFF_E800 ~ 0x1FFF_EFFF	芯片出厂前配置

5. 4. 4. 2 程序区保护

程序区的保护功能可分为用户代码读出保护(UCRP)、读保护(RP)和写保护(WP)。除了读保护外的保护都支持以页为单位进行配置，最多支持 2 组区间保护。当这 2 组不连续区间有重迭区域时，以能保护的最大范围为基准。例如，如果第一组保护设定范围为从页 0 至页 31，而第二组保护为页 30 至页 40，则全部的保护范围为页 0 至页 40。

所有的保护设定在配置完毕后都不会立即反映。用户需要藉由配置字重载流程、重新上电才会反映新的保护设定。

程序区的 3 种保护功能分别如下所述:

用户代码读出保护(UCRP)

UCRP 的主要功能为防止保护区内的数据被读出与修改，因此保护区内禁止任何人以读取”数据”的方式进行读取，但允许读取指令执行。同时，为了防止保护区内的信息被覆盖，保护区内也禁止编程与擦除。

UCRP 的设定主要储存在信息区页 0。若用户需要清除保护设定，则须藉由擦除信息区页 0 来清除保护设定。需要注意的是，在清除 UCRP 设定时，保护区内的数据会一并被擦除，以避免原先受到保护的数据被读出。但原先未受到 UCRP 保护的页面内的数据仍会被保留。同时，在信息区页 0 擦除完毕后，UCRP 保护设定会立即更新为未保护的状态。

开启 UCRP 的流程如下所示。在配置流程结束后，保护设定并不会立即反映。用户需藉由配置字重载、重新上电才会反映新的保护设定。

- 解锁闪存控制器。(请参阅闪存操作解锁)
- 对 **FC_UP0** 填入保护区间的起始页与终止页，并配置保护开关为 0。在配置起始页与终止页时，需注意终止页不可小于起始页，否则保护设定为无效。
  - **FC_UP0**[31]为保护开关，数值为 0 代表开启保护。
  - **FC_UP0**[23:16]为保护终止页。
  - **FC_UP0**[7:0]为保护起始页。
- 对 **FC_CMD** 填入编程指令 0xF5 可开启配置 UCRP0 的流程，而填入 0xF6 则可开启配置 UCRP1 的流程。在进行保护配置的过程中，硬件会自动在相对应的位置填入取反的数据。
- 配置流程结束后，需要重新将闪存控制器上锁。(请参阅闪存操作解锁)
- 需藉由配置字重载、重新上电才会反映新的保护设定。

读保护(RP)

读保护的主要功能是防止程序区被 ISP、Debug Port 和 Bootrom 读出和修改，但保护区内的程序仍可读取和修改保护区内的数据。读保护共分为 3 个等级，分别如下:

- Lv0:不保护，保护设定数值为 0xAAAA_AAAA。可藉由擦除信息区页 1 将保护等级提升至 Lv1 保护。用户可藉由读保护设定流程将读保护等级提升至 Lv1 或是 Lv2，在提升保护等级的过程中，程序区内所储存的信息仍会保留。
- Lv1:当保护设定不为 0xAAAA_AAAA 或 0xCCCC_CCCC 时开启此保护模式。在此模式下仅有程序区的程序才可以对程序区进行读取和修改，但存放保护设定的信息区则不受限

制。可藉由保护设定流程将保护等级降回 Lv0 保护，在设定过程中会触发程序区全清除。由于程序区已被清除，因此会同时清除用户代码读出保护设定、写保护设定、用户配置字与存放于信息区页 3 内的用户数据。在设定完毕后，仅能藉由重新上电来触发保护更新。当用户确认不会再藉由 Debug Port 和 Bootrom 修改闪存程序区的内容时，可藉由读保护设定流程将读保护等级提升至 Lv2，在提升保护等级的过程中，程序区内所储存的信息仍会保留。在 Lv1 读保护下，若对信息区页 1 执行清除时仅会清除信息区页 1，程序区内的信息仍会保留，而读保护等级将继续维持在 Lv1。

- Lv2:读保护模式中的最高等级，保护设定数值为 0xCCCC_CCCC 时才会启用。开启此保护模式后，无法再将保护等级降回 Lv1 或 Lv0，因此在使用上需特别小心。一旦开启保护后，系统会自动断开 Debug Port，同时强制映射在程序区，使得用户无法再藉由 Debug Port 与 Bootrom 来修改程序区的内容。然而，用户仍然可以通过程序区内事先写好的更新流程来修改程序区的内容。

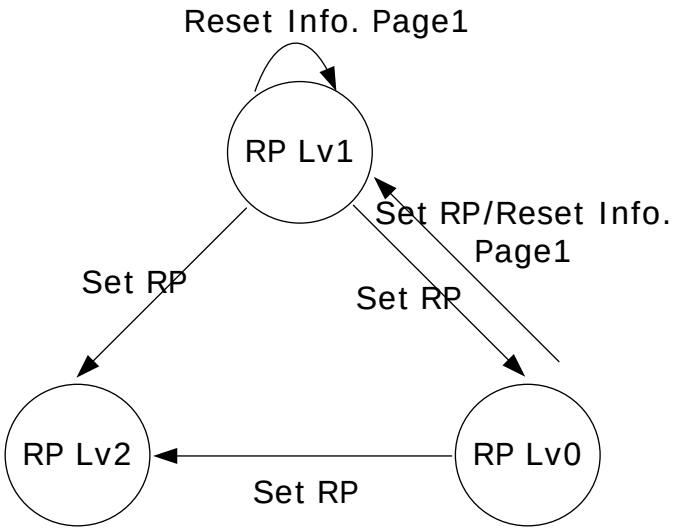


图 5-1　读保护等级转换示意图

以下是读保护的配置流程，需要注意的是配置完成后，保护设定并不会立即生效，用户需藉由配置字重载、重新上电才会反映新的保护设定:

- 解锁闪存控制器。(请参阅闪存操作解锁)
- 对 **FC_UP0** 填入保护设定。填入数值为 0xCCCC_CCCC 代表设定读保护等级为 Lv2；填入数值为 0xAAAA_AAAA 代表设定读保护等级为 Lv0，若此时读保护不为 Lv0 则会触发程序区全擦除的流程。
- 对 **FC_CMD** 填入编程指令 0xF7 开启配置 RP 流程。
- 于流程结束后重新将闪存控制器上锁。(请参阅闪存操作解锁)
- 藉由配置字重载、重新上电才会反映新的保护设定。

写保护(WP)

写保护的主要目的是为了保护程序区内的数据不被误擦除或覆盖。保护区域会禁止进行编程和擦除，但仍可读取数据。写保护的设定存放于信息区页 2。若用户擦除信息区页 2，将会清除写保护。在清除写保护的过程中，原先保护区内的数据仍会被保留。同时，WP 保护设定会在信息区页 2 擦除完毕后，立即更新为未保护的状态。需注意的是，在清除写保护的过程中，会一并擦除存放于信息区页 2 的用户配置字，如硬件映射设定、系统欠压复位设定以及 IWDT 设定。

下面是开启写保护的流程，请注意，配置流程结束后，保护设定不会立即生效，用户需藉由配置字重载、重新上电才会反映新的保护设定：

- 解锁闪存控制器。(请参阅闪存操作解锁)
- 对 **FC_UP0** 填入保护区间的起始页与终止页，并配置保护开关为 0。在配置起始页与终止页时，需注意终止页不可小于起始页，否则保护设定为无效。
  - FC_UP0[31]为保护开关，数值为 0 代表开启保护。
  - FC_UP0[23:16]为保护终止页。
  - FC_UP0[7: 0]为保护起始页。
- 对 **FC_CMD** 填入指令 0xF8 可开启配置 WP0 的流程，而填入指令 0xF9 则可开启配置 WP1 的流程。在进行保护配置的过程中，硬件会自动将相对应位置的数据填入取反的值。
- 于流程结束后重新将闪存控制器上锁。(请参阅闪存操作解锁)
- 藉由配置字重载、重新上电才会反映新的保护设定。

综合上述保护的描述，保护的权限整理如下表所示:

	映射在程序区		映射在 System Memory/SRAM、SWD 界面		
读保护等级	Lv0/Lv1	Lv2	Lv0	Lv1	Lv2
程序区(闪存控制器解锁)	R/W	R/W	R/W	X	X
程序区(闪存控制器未解锁)	R	R	R	X	X
程序区(WP Sector)	R	R	R	X	X
程序区(UCRP Sector)	Fetch	Fetch	Fetch	X	X
信息区 - UCRP 设定	R/W	R/W	R/W	R/W	X
信息区 - RP 设定	R/W	R	R/W	R/W	X
信息区 - WP 设定	R/W	R/W	R/W	R/W	X

5. 4. 5 闪存重映射

当用户想要从闪存的程序区开启系统程序时，可以进一步设定闪存的内部映射。这种映射方式的实现方式是通过偏移中断向量表的方式实现，以 8 个页(4 KByte)为基准进行映射。这样，可以将程序区分为 32 个区块。用户可以通过设定 SYSCFG 寄存器内 SYSCFG_REMAP(0x0)的 EFBASE 来决定要映射至哪一个区块开始执行。下面是映射至程序区的第 3 个区块的设定流程：

- 设定 SYSCFG_REMAP. EFBASE 的数值为 0x2(代表映射至第 3 个 4K Byte)。
- 设定 SYSCFG_REMAP. MEMMOD 的数值为 0x0。

- 设定 SYSCFG_REMAP. REMAP 的数值为 0x1。
- 执行 NVIC_SystemReset()复位函数复位 CPU。
- 当CPU 被重置以后，随即会从程序区位在 0x0800_2000 的位置开始读取中断向量表(CPU 仍然是从 0x0000_0000 的位置开始读取中断向量表)。

闪存程序区的内部映射仅在读取闪存数据时才进行，对闪存进行编程和擦除时不会受到内部映射的影响。此外，如果使用闪存的实体位置来读取数据，也不会受到内部映射的影响。

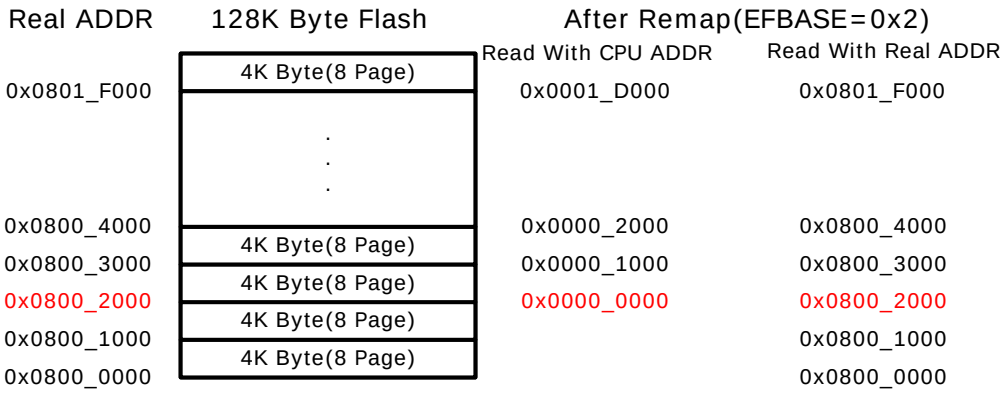


图 5-2　闪存映射后读取位置对照图

5. 4. 6　配置字重载

在用户配置保护后，若要使新的保护设定生效，可以通过闪存控制器重启配置字重载流程。需要注意的是，此流程会将系统重置，使程序重新运行。在程序区的程序重新运行时，可以通过检查 Reset and Clock Control Unit (RCU)内的 **RCU_RSTF** 状态位是否被设定为 1 来确认系统是否已触发过配置字重载流程。配置字重载的流程如下:

- 解锁闪存控制器。(请参阅闪存操作解锁)
- 对 **FC_CTL** 的 OPRLD 填入 0xE 开启加载流程。

5. 4. 7　闪存编程

闪存的编程仅能通过闪存控制器进行，编程是以 32 位(Bit)为单位进行，编程一组 32 位的数据共需 25us。闪存编程的流程如下:

- 解锁闪存控制器。(请参阅闪存操作解锁)
- 对 **FC_PA** 填入欲编程的位置以及编程的次数。总编程的次数为设定的编程次数+1，每次编程完毕后编程位置会自动累加，最多可连续编程 128 次而不必重新填入新的编程位置。
  - 对 FC_PA[31:25]填入连续编程的次数。
  - 对 FC_PA[19:0]填入编程的位置。
- 若需要对信息区进行编程时，需设定 **FC_PA** 位于第 24 位的 IFREN 为 1，否则需设定此位为 0。
- 对 **FC_PLD** 填入欲编程的 32 位数据。

- 对 **FC_CMD** 填入编程指令 0xF0 进行闪存编程。在闪存进行编程的期间，会暂时停住 CPU，避免系统误动作。
- 当编程完毕以后会自动清除 **FC_CMD** 内的指令，但仍会保留 **FC_PLD** 内的编程数据。当编程的次数已满足 **FC_PA**.PCNT 所设定的次数时，会自动清除 **FC_PA**，否则会自动将目前的位置加上 4。
- 重复对 **FC_PLD** 与 **FC_CMD** 填入编程数据与编程指令直到满足编程次数。
- 于流程结束后重新将闪存控制器上锁。(请参阅闪存操作解锁)

5. 4. 8 闪存擦除

闪存的擦除仅能通过闪存控制器进行，闪存控制器支持下列 3 种擦除模式:

页擦除

以页(512 Byte)为单位进行擦除，每一页擦除耗时约 2ms。擦除留程如下:

- 解锁闪存控制器。(请参阅闪存操作解锁)
- 对 **FC_PA** 填入欲擦除的页面位置。
- 若需要对信息区进行擦除时，需设定 **FC_PA** 位于第 24 位的 IFREN 为 1，否则需设定此位为 0。
- 对 **FC_CMD** 填入编程指令 0xF1 进行闪存编程。在闪存编程期间，会暂时停住 CPU，避免系统误动作。
- 当编程完毕以后会自动清除 **FC_CMD** 内的指令。
- 于流程结束后重新将闪存控制器上锁。(请参阅闪存操作解锁)

全擦除

闪存全擦除可依据闪存保护是否开启，区分为下列 3 种模式:

- 未开启 UCRP、WP 保护，且 RP 等级为 Lv0 时，会将闪存程序区全部擦除，擦除约耗时 8ms。
- 开启 UCRP 或 WP 保护，但 RP 等级为 Lv0 时，会将闪存程序区内未受到 UCRP 与 WP 保护的 Sector 擦除，每擦除一个 Sector 需耗时 2ms。
- 读保护等级不为 Lv0 时，不再支持闪存全擦除的功能，避免因闪存内的程序误操作而将闪存程序区的数据完全擦除。

闪存全擦除的留程如下:

- 解锁闪存控制器。(请参阅闪存操作解锁)
- 对 **FC_PA** 填入 0x0000_0000。
- 对 **FC_CMD** 填入编程指令 0xF3 进行闪存全擦除。在闪存编程期间，会暂时停住 CPU，避免系统误动作。
- 当编程完毕以后会自动清除 **FC_CMD** 内的指令。
- 于流程结束后重新将闪存控制器上锁。(请参阅闪存操作解锁)

5. 5 特殊功能寄存器

5. 5. 1 寄存器列表

FC 寄存器列表			
名称	偏移地址	类型	描述
FC_CMD	0000_H	R/W	闪存命令寄存器
FC_PA	0004_H	R/W	闪存编程地址寄存器
FC_PLD	0008_H	R/W	闪存编程数据低位寄存器
FC_CTL	0010_H	R/W	闪存控制寄存器
FC_STA	0014_H	R	闪存状态寄存器
FC_UL	0018_H	R/W	闪存控制解锁寄存器
FC_UP0	0020_H	R/W	闪存保护更新寄存器 0
FC_UCRP0	0050_H	R	闪存配置字用户代码读出保护设定寄存器 0
FC_UCRP0REV	0054_H	R	闪存配置字用户代码读出保护设定取反寄存器 0
FC_UCRP1	0058_H	R	闪存配置字用户代码读出保护设定寄存器 1
FC_UCRP1REV	$005C_H$	R	闪存配置字用户代码读出保护设定取反寄存器 1
FC_RP	0070_H	R	闪存配置字读保护设定寄存器
FC_WP0	0074_H	R	闪存配置字写保护设定寄存器 0
FC_WP0REV	0078_H	R	闪存配置字写保护设定取反寄存器 0
FC_WP1	$007C_H$	R	闪存配置字写保护设定寄存器 1
FC_WP1REV	0080_H	R	闪存配置字写保护设定取反寄存器 1
FC_REMAP	0094_H	R	闪存配置字硬件映射寄存器

5. 5. 2 寄存器描述

5. 5. 2. 1 闪存命令寄存器(FC_CMD)

闪存命令寄存器 (FC_CMD)																													
偏移地址:0x00																													
复位值:0x0000 0000																													
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																								CMD<7:0>							

—	Bits 31-8	—	—
CMD	Bits 7-0	R/W	**闪存命令** 藉由配置此寄存器对闪存执行编程与擦除。 当程序完成时，闪存命令将自动被清除。 0xF0:闪存编程。 0xF1:闪存页擦除(1 Page Erase)。 0xF3:闪存全擦除。 0xF5:配置用户代码读出保护(UCRP0)。 0xF6:配置用户代码读出保护(UCRP1)。 0xF7:配置读保护(RP)。 0xF8:配置写保护(WP0)。 0xF9:配置写保护(WP1)。

5. 5. 2. 2 闪存编程地址寄存器(FC_PA)

闪存编程地址寄存器(FC_PA)																															
偏移地址:0x04																															
复位值:0x0000 0000																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PCNT<6:0>							IFREN	—	—	—	—	PA <19:0>																			

PCNT	Bits 31-25	R/W	**编程计数器** 提供(PCNT + 1)次连续编程，最多可提供128次连续编程，在编程期间，用户仅需填写FC_PLD与FC_CMD即可，编程完毕后自动将PA位置加4。
IFREN	Bit 24	R/W	**信息区块始能** 0:禁用信息区访问。 1:启用信息区访问。
—	Bits 23-20	—	—
PA	Bits19-0	R/W	**编程/擦除地址** 闪存编程: PA[15:2]为Word地址。 闪存页擦除: PA[15:9]为页地址，PA[8:0]可忽略。 闪存全擦除:PA[15:0]不需配置，可忽略。

5. 5. 2. 3 闪存编程数据低位寄存器(FC_PLD)

闪存编程数据低位寄存器 (FC_PLD)																															
偏移地址:0x08																															
复位值:0xFFFF FFFF																															
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
PLD <31:0>																															

PLD	Bits 31-0	R/W	**闪存编程数据低 32 位** 闪存 32 位编程数据。

5. 5. 2. 4 闪存控制寄存器 (FC_CTL)

闪存控制寄存器(FC_CTL)																													
偏移地址:0x10																													
复位值:0x0000 0000																													
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																					FCSLEEP			OPRLD<3:0>					WAIT<2:0>	

—	Bits 31-11	—	—
FCSLEEP	Bits 10	R/W	**停止模式开关** 当系统进入STOP模式时，可配置FCSLEEP为1，让闪存进入停止模式。 0x0:关闭闪存停止模式开关。 0x1:开启闪存停止模式开关。
—	Bits 9-8	—	—
OPRLD	Bits 7-4	R/W	**配置字载入密钥** 填入固定值0xE触发配置字重载，重载配置字后会触发系统复位。
—	Bits 3	—	—
WAIT	Bits2-0	R/W	**闪存读取等待周期** 闪存读取时间固定40ns，藉由配置此寄存器，确保有足够的时间读取闪存。当系统时钟周期大于40ns时，可配置为0；若系统时钟周期小于40ns时，则配置此寄存器确保读取时间大于40ns。 0x0: 系统频率≤24Mhz 0x1: 系统频率>24Mhz且系统频率≤48Mhz 0x2: 系统频率>48Mhz且系统频率≤72Mhz 其他: 保留

5. 5. 2. 5 闪存状态寄存器 (FC_STA)

闪存状态寄存器(FC_STA)																												
偏移地址:0x14																												
复位值:0x0000 0009																												
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
																				OPRLDLOOP <3:0>				PRTAREARD	PRTAREAWR	CMDULK	FCBUSY	WPDIS	RPLV<1:0>		UCRPDIS

—	Bits 31-12	—	—
OPRLDLOOP	Bit 11-8	R	**Option Byte重载次数** 纪录开机后系统加载Option Byte时，总共重载了几次以后才读取成功，最大计数为15次。
PRTAREARD	Bit 7	R/C_W1	**保护区读取状态** 纪录保护区是否遭到非法读取操作。 0x0:保护区未被进行读取操作。 0x1:发生程序区读取保护区数据的不合法动作。
PRTAREAWR	Bit 6	R/C_W1	**保护区操作状态** 纪录保护区是否遭到非法Program/Erase操作。 0x0:保护区未被进行Program/Erase操作。 0x1:发生程序区操作保护区的不合法动作。
CMDULK	Bit 5	R	**FC_CMD 寄存器保护状态** 0x0:FC_CMD 寄存器锁定。 0x1FC_CMD 寄存器已解锁。
FCBUSY	Bit 4	R	**闪存控制器忙碌状态** 0x0:闪存控制器闲置。 0x1:闪存控制器忙碌中。
WPDIS	Bit 3	R	**程序区写保护(WP)保护状态** 0x0:保护功能已开启。 0x1:保护功能未开启。
RPLV	Bit 2-1	R	**程序区读保护(RP)保护状态** 0x0:读保护等级为 Lv0。 0x1:读保护等级为 Lv1。 0x2:读保护等级为 Lv2。
UCRPDIS	Bit 0	R	**程序区用户代码读出保护(UCRP)保护状态** 0x0:保护功能已开启。 0x1:保护功能未开启。

5. 5. 2. 6 闪存控制解锁寄存器 (FC_UL)

闪存控制解锁寄存器(FC_UL)																															
偏移地址:0x18																															
复位值:0x0000 0000																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
UL<31:0>																															

UL	Bits 31-0	R/W	**闪存控制解锁密钥** 连续输入 2 组密钥解锁 FC_CMD 寄存器(第一组密钥为 0x00112233，第二组密钥为 0x55667788)。输入错误的密钥或是输入第 3 组密钥时，FC_CMD 寄存器会重新锁上。

5. 5. 2. 7 闪存保护更新寄存器 0 (FC_UP0)

闪存保护更新寄存器 0(FC_UP0)																															
偏移地址:0x20																															
复位值:0xFFFF FFFF																															
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
UP0<31:0>																															

UP0	Bits 31-0	R/W	**更新保护设定 0** 更新程序区的保护设定 • 配置 UCRP 与 WP 时，以页为单位进行配置: – UP0[31]为保护开关，设定 0 代表开启保护。 – UP0[23:16]为保护终止页。 – UP0[7:0]为保护起始页。 • 配置 RP 时: – Lv0:UP0[31:0]填入 0xAAAAAAAA – Lv2:UP0[31:0]填入 0xCCCCCCCC – Lv1:UP0[31:0]填入 0xAAAAAAAA 与 0xCCCCCCC 以外之任意数值。

5. 5. 2. 8 闪存配置字用户代码读出保护设定寄存器 0 (FC_UCRP0)

闪存配置字用户代码读出保护设定寄存器 0 (FC_UCRP0)																															
偏移地址:0x50																															
复位值:0xFFFF FFFF																															
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
UCRP0<31:0>																															

UCRP0	Bits 31-0	R	**闪存配置字用户代码读出保护设定 0** 程序区用户代码读出保护设定，以页为单位进行保护。 • UCRP0[31]为保护开关，数值为 0 代表开启保护。 • UCRP0[23:16]为保护终止页。 • UCRP0[7:0]为保护起始页。

5. 5. 2. 9 闪存配置字用户代码读出保护设定取反寄存器 0 (FC_UCRP0REV)

闪存配置字用户代码读出保护设定取反寄存器 0 (FC_UCRP0REV)																															
偏移地址:0x54																															
复位值:0xFFFF FFFF																															
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
UCRP0REV <31:0>																															

UCRP0REV	Bits 31-0	R	**闪存配置字用户代码读出保护设定 0 取反值** 存放 FC_UCRP0 的取反数值。当取反不满足，且 FC_UCRP0 与 FC_UCRP0REV 的数值不为 0xFFFFFFFF 时自动开启程序区全区的 UCRP 保护。

5. 5. 2. 10 闪存配置字用户代码读出保护设定寄存器 1 (FC_UCRP1)

闪存配置字用户代码读出保护设定寄存器 1 (FC_UCRP1)																															
偏移地址:0x58																															
复位值:0xFFFF FFFF																															
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
UCRP1<31:0>																															

UCRP1	Bits 31-0	R	**闪存配置字用户代码读出保护设定 1** 程序区用户代码读出保护设定，以页为单位进行保护。 • UCRP1[31]为保护开关，数值为 0 代表开启保护。 • UCRP1[23:16]为保护终止页。 • UCRP1[7:0]为保护起始页。

5. 5. 2. 11 闪存配置字用户代码读出保护设定取反寄存器 1 (FC_UCRP1REV)

闪存配置字用户代码读出保护设定取反寄存器 1 (FC_UCRP1REV)																															
偏移地址:0x5C																															
复位值:0xFFFF FFFF																															
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
UCRP1REV <31:0>																															

UCRP1REV	Bits 31-0	R	**闪存配置字用户代码读出保护设定 1 取反值** 存放 FC_UCRP1 的取反数值。当取反不满足，且 FC_UCRP1 与 FC_UCRP1REV 的数值不为 0xFFFFFFFF 时自动开启程序区全区的 UCRP 保护。

5. 5. 2. 12 闪存配置字读保护设定寄存器 (FC_RP)

闪存配置字读保护设定寄存器 (FC_RP)																															
偏移地址:0x70																															
复位值:0xAAAA AAAA																															
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
RP<31:0>																															

RP	Bits 31-0	R	**闪存配置字读保护设定** 0xAAAA AAAA:读保护等级为 Lv0。 0xCCCC CCCC:读保护等级为 Lv2。 else:读保护等级为 Lv1。

5. 5. 2. 13 闪存配置字写保护设定寄存器 0 (FC_WP0)

闪存配置字写保护设定寄存器 0 (FC_WP0)																															
偏移地址:0x74																															
复位值:0xFFFF FFFF																															
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
WP0<31:0>																															

WP0	Bits 31-0	R	**闪存配置字写保护设定0** 程序区写保护设定，以页为单位进行保护。 • WP0[31]为保护开关，数值为 0 代表开启保护。 • WP0[23:16]为保护终止页。 • WP0[7:0]为保护起始页。

5. 5. 2. 14 闪存配置字写保护设定取反寄存器 0 (FC_WP0REV)

闪存配置字写保护设定取反寄存器 0 (FC_WP0REV)																															
偏移地址:0x78																															
复位值:0xFFFF FFFF																															
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
WP0REV<31:0>																															

WP0REV	Bits 31-0	R	**闪存配置字写保护设定0取反值** 存放 FC_WP0 的取反数值。当取反不满足，且 FC_WP0 与 FC_WP0REV 的数值不为 0xFFFFFFFF 时自动开启程序区全区的 WP 保护。

5. 5. 2. 15 闪存配置字写保护设定寄存器 1 (FC_WP1)

闪存配置字写保护设定寄存器 1 (FC_WP1)																															
偏移地址:0x7C																															
复位值:0xFFFF FFFF																															
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
WP1<31:0>																															

WP1	Bits 31-0	R	**闪存配置字写保护设定1** 程序区写保护设定，以页为单位进行保护。 • WP1[31]为保护开关，数值为 0 代表开启保护。 • WP1[23:16]为保护终止页。 • WP1[7:0]为保护起始页。

5. 5. 2. 16 闪存配置字写保护设定取反寄存器 1 (FC_WP1REV)

闪存配置字写保护设定取反寄存器 1 (FC_WP1REV)																															
偏移地址:0x80																															
复位值:0xFFFF FFFF																															
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
WP1REV<31:0>																															

WP1REV	Bits 31-0	R	**闪存配置字写保护设定1取反值** 存放 FC_WP1 的取反数值。当取反不满足，且 FC_WP1 与 FC_WP1REV 的数值不为 0xFFFFFFFF 时自动开启程序区全区的 WP 保护。

5. 5. 2. 17 闪存配置字硬件映射寄存器 (FC_REMAP)

闪存配置字硬件映射寄存器(FC_REMAP)																															
偏移地址:0x94																															
复位值:0xFFFF FFFF																															
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
								BOOTBYP <7:0>								SELECT <7:0>								SEFBASE<7:0>							

—	Bits 31-24	—	—
BOOTBYP	Bit 23-16	R	**硬件重映射选项** BOOTBYP 数值决定系统开机后是否执行 Bootrom 流程。 0xA5:系统开机后跳过 Bootrom，映射至闪存程序区执行。 其他:系统开机后优先执行 Bootrom。
SELECT	Bit 15-8	R	**硬件重映射选项** SREMAP 数值告知 Bootrom 流程结束后的映射位置。 0x00:闪存映射至主存储器。 0x01:System Memory 映射至主存储器。 0x02: SRAM 映射至主存储器。

SEFBASE	Bit 7-0	R	**闪存映射起始地址选择** 闪存映射至主存储器时，可以 4KB 为单位配置闪存起始位置。SEFBASE 仅有在 BOOTBYP 被设定为 0 时才有效。当 BOOTBYP 数值为 0xA5 时，系统依据 SEFBASE 数值映射至闪存相对应 4KB 的位置执行；当 BOOTBYP 数值不为 0xA5 时，SEFBASE 数值影响 Bootrom 流程结束后映射至闪存的位置。 举例来说，当 SEFBASE 数值为 1 时，代表主存储器 0x0000 0000 的位置对应到闪存第二个 4KB 的位置。

第6章 通用 I/Os (GPIO)

6. 1 概述

每个通用 I/O 端口具有三个 32 位配置寄存器(**GPIOx_MOD**、**GPIOx_OT**、**GPIOx_PUD**)、两个 32 位数据寄存器(**GPIOx_ID** 和 **GPIOx_OD**)和一个 32 位置位/复位寄存器(**GPIOx_BSBR**)。此外，所有 GPIO 都具有一个 32 位锁定寄存器(**GPIOx_LCK**)和两个 32 位复用功能寄存器(**GPIOx_AFH** 和 **GPIOx_AFL**)。

6. 2 特性

- 输出状态:带有上拉或下拉的推挽输出或开漏输出
- 从输出数据寄存器(**GPIOx_OD**)或外围设备(复用功能输出)输出数据
- 可选的每个 IO 端口的驱动电流
- 输入状态:悬空、上拉/下拉、模拟输入
- 输入数据到输入数据寄存器(**GPIOx_ID**)或外围设备(复用功能输入)
- 置位和复位寄存器(**GPIOx_ BSBR**)，对 **GPIOx_OD** 具有按位操作权限
- 锁定机制(**GPIOx_LCK**)，可冻结 I/O 配置
- 模拟功能
- 快速翻转，每次翻转最快只需要两个时钟周期
- 允许 GPIO 端口和外设引脚的高灵活性复用

6.3 结构图

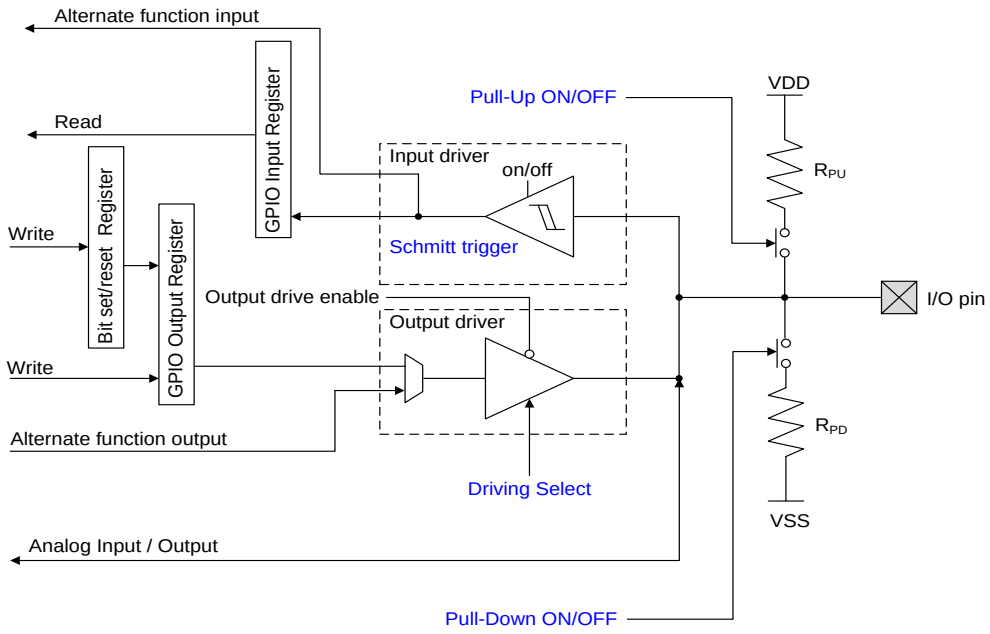


图 6-1 I/O 端口位的基本结构图

6.4 功能描述

根据数据表中列出的每个 I/O 端口的具体硬件特性，通用 I/O(GPIO)端口的每个端口位可以通过软件以几种模式单独配置:

- 悬空输入
- 上拉输入
- 下拉输入
- 模拟输入
- 具有上拉或下拉能力的开漏输出
- 具有上拉或下拉能力的推挽输出
- 复用功能且具有上拉或下拉能力的开漏输出
- 复用功能且具有上拉或下拉能力的推挽输出

每个 I/O 端口位可以自由编程，然而 I/O 端口寄存器可按 32 位字、半字或字节访问。**GPIOx_BSBR** 寄存器的用途是对 **GPIOx_OD** 寄存器的每一个位进行个别配置。这种情况下，在读和写访问之间产生 IRQ 时也不会有风险。

MOD[1:0]	OT	PUPD[1:0]	I/O 配置
00	X	00	GP 输入+悬空
00	X	01	GP 输入+上拉
00	X	10	GP 输入+下拉
00	X	11	Reserved
01	0	00	GP 输出+推挽
01	0	01	GP 输出+推挽+上拉
01	0	10	GP 输出+推挽+下拉
01	0	11	Reserved
01	1	00	GP 输出+开漏
01	1	01	GP 输出+开漏+上拉
01	1	10	GP 输出+开漏+下拉
01	1	11	Reserved
10	0	00	AF 复用+推挽
10	0	01	AF 复用+推挽+上拉
10	0	10	AF 复用+推挽+下拉
10	0	11	Reserved
10	1	00	AF 复用+开漏
10	1	01	AF 复用+开漏+上拉

MOD[1:0]	OT	PUPD[1:0]	I/O 配置
10	1	10	AF 复用+开漏+下拉
10	1	11	Reserved
11	X	00	模拟输入/输出
11	X	01	Reserved
11	X	10	Reserved
11	X	11	Reserved

表 6-1　GPIO 配置表

6. 4. 1　通用 I/O (GPIO)

复位期间和刚复位后，复位功能未开启且所有的 I/O 端口被配置为模拟功能。

当作为输出配置时，写到输出数据寄存器(**GPIOx_OD**)的值输出到相应的引脚上。可以以推挽模式或开漏模式(仅低电平被驱动，高电平表现为高阻)使用输出驱动器。

输入数据寄存器(**GPIOx_ID**)在每个 AHB 时钟周期捕捉 I/O 引脚上的数据。所有 GPIO 引脚都有一个内部弱上拉和弱下拉电阻，它们被激活或断开有赖 **GPIOx_PUD** 寄存器的值。

6. 4. 2　I/O端口复用功能多任务与映射

每个 I/O 端口都同时通过多任务器连结至内部各个外设模块，并在同一时间只允许 1 个外设模块通过复用功能(AF)连结至 I/O 端口上。通过这种方式，外设模块将不会在同一个 I/O 端口上产生冲突。

每个 I/O 端口拥有高达 9 个复用功能(AF0 ~ AF9)，可通过配置 **GPIOx_AFL** 与 **GPIOx_AFH** 寄存器来选用。

除了这种灵活的 I/O 多路复用架构之外，每个外设还具有复用功能映射到不同的 I/O 引脚，以优化可用的外设数量以及较小的芯片封装。

要在给定的配置中使用 I/O，用户必须执行以下操作:

- 调试功能:在系统复位后，这些引脚会被配置为复用功能，并能够立即的让调试主机使用。
- 通用 I/O:在 **GPIOx_MOD** 寄存器中配置所需的 I/O 为输出、输入或模拟功能。
- 外设复用功能:
  - 将想要的 I/O 通过 **GPIOx_AFL** 或 **GPIOx_AFH** 寄存器配置 AFx
  - 选择一个型态，通过 **GPIOx_PUD** 与 **GPIOx_OT** 寄存器来选择上拉/下拉、推挽/开漏。
  - 通过 **GPIOx_MOD** 寄存器来配置想要的 I/O 为复用功能。
- 附加功能:
  - 对于 ADC、LCD，通过 **GPIOx_MOD** 寄存器来配置想要的 I/O 为复用模式，并在 ADC、LCD 寄存器配置想要的功能。如上所述，对于附加功能(例如 ADC、LCD)，

输出输入是由相应外设控制，在启用附加功能输出前必须谨慎选择 I/O 端口复用功能。

- 对于 CMP，通过 **GPIOx_MOD** 寄存器来配置想要的 I/O 为模拟模式，并在 CMP 寄存器配置想要的功能。
- 对于 WKUPx，在 SYSCFG 内配置相关的寄存器。这些功能配置优先于标准通用 I/O 寄存器中的配置。

6. 4. 3 I/O端口控制寄存器

每个 GPIO 端口都有 3 个 32 位的控制寄存器(**GPIOx_MOD**、**GPIOx_OT**、**GPIOx_PUD**)用来配置多达 16 个 I/O 端口。**GPIOx_MOD** 寄存器用来选择 I/O 模式 (如输入、输出、复用或模拟)。**GPIOx_OT** 寄存器用来选择输出类型 (如推挽或开漏)。**GPIOx_PUD** 寄存器用来选择上拉/下拉方式。

6. 4. 4 I/O端口数据寄存器

每个 GPIO 端口有两个 16 位数据寄存器:输入和输出数据寄存器(**GPIOx_ID** 和 **GPIOx_OD**)。**GPIOx_OD** 寄存器用于存储输出数据，其可进行读/写访问。从 I/O 口的输入数据存放在 **GPIOx_ID** 寄存器中，该寄存器为只读寄存器。

6. 4. 5 I/O数据位操作

端口置位复位寄存器(**GPIOx_BSBR**)是一个 32 位的寄存器，其允许应用对输出数据寄存器(**GPIOx_OD**)的每个位进行置位和复位操作。端口置位和复位寄存器的有效数据宽度是 **GPIOx_OD** 有效数据宽度的两倍。

对于 **GPIOx_OD** 中的每个位，在 **GPIOx_BSBR** 中有两个位与之对应:BS(i)和 BR(i)。当对位 BS(i)写 1 时则设置相应的 OD(i)位。当对 BR(i)写 1 时，则复位相应的 OD(i)位。对 **GPIOx_BSBR** 中的任意位写 0 都不会影响 **GPIOx_OD** 寄存器的值。若对 **GPIOx_BSBR** 的 BS(i)和 BR(i)同时置 1，那么其置位操作具有优先权 (即对相应位做置位操作)。

6. 4. 6 GPIO锁定机制

通过 **GPIOx_LCK** 寄存器，并经由特定锁定流程，可锁定冻结 GPIO 控制寄存器，包括 **GPIOx_MOD**、**GPIOx_OT**、**GPIOx_PUD**、**GPIOx_AFL** 和 **GPIOx_AFH** 寄存器。

锁定流程是通过写 **GPIOx_LCK** 寄存器，需写两次相同数值，每次以 32 位数值写入，**GPIOx_LCK**[31:16]必须为 **GPIOx_LCK**[15:0]取反值。当锁定流程正确时，**GPIOx_LCK**.LCKK 位锁定为 1，表示锁定功能开启，功能开启后无法取消，只经由复位清除。

6. 4. 7 I/O复用功能输入/输出

对于每个 I/O 提供两个寄存器来选择复用功能输入/输出间的其中一个。使用这些寄存器，用户可以根据应用来选择将复用功能连至其他某个引脚上。

这意味着通过 **GPIOx_AFL** 和 **GPIOx_AFH** 寄存器可让每个 GPIO 上覆用了许多可能的外设功能。软件应用可为每个 I/O 选择任何一种可能的功能。当一个复用功能被选择时，会将选定的 I/O 配置为该复用功能的输入或输出。

6. 4. 8 外部中断/唤醒通道

所有端口均具有外部中断功能。要使用外部中断通道，端口可以配置在输入、输出或复用功能模式(端口不得在模拟模式)。

6. 4. 9 输入配置

当 I/O 端口配置为输入模式:

- 输出缓冲器关闭。
- 施密特触发器输入开启。
- 根据 GPIOx_PUD 寄存器来决定是上拉或下拉功能。
- I/O 引脚的状态将会被 AHB 时钟取样后存到输入寄存器内。
- 对输入寄存器进行读操作，将会提供此时 I/O 的状态。

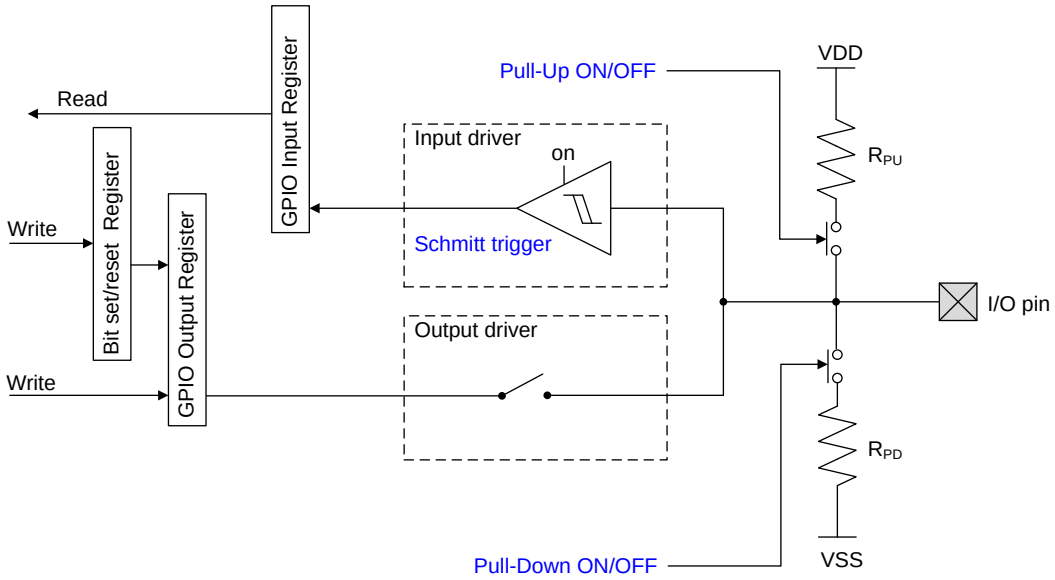


图 6-2 I/O 端口位的输入配置

6. 4. 10 输出配置

当 I/O 端口配置为输出模式:

- 输出缓冲器开启。
  - 开漏模式:当 **GPIOx_OD** 寄存器值为 0 时，输出值为"0"；当 **GPIOx_OD** 寄存器值为 1 时，输出值为"高阻态"。
  - 推挽模式:当 **GPIOx_OD** 寄存器值为 0 时，输出值为"0"；当 **GPIOx_OD** 寄存器值为 1 时，输出值为"1"。
- 施密特触发器输入开启。
- 根据 **GPIOx_PUD** 寄存器来决定是上拉或下拉功能。
- I/O 引脚的状态将会被 AHB 时钟取样后存到输入寄存器内。
- 对输入寄存器进行读操作，将会得到此时 I/O 的状态。
- 对输出寄存器进行读操作，将会得到最后一次写入的数值。

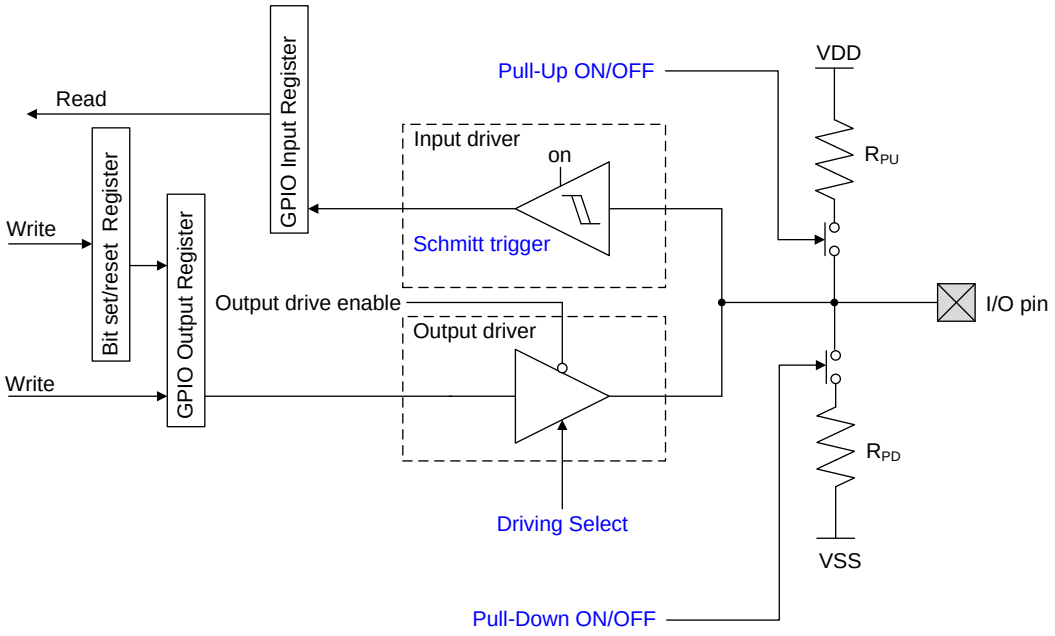


图 6-3 I/O 端口位的输出配置

6. 4. 11 复用功能配置

当 I/O 端口配置为复用模式:

- 输出缓冲器可配置为开漏或推挽模式。
- 输出缓冲器的输出信号将来自外部设备。
- 施密特触发器输入开启。
- 根据 GPIOx_PUD 寄存器来决定是上拉或下拉功能。
- I/O 引脚的状态将会被 AHB 时钟取样后存到输入寄存器内。
- 对输入寄存器进行读操作，将会得到此时 I/O 的状态。

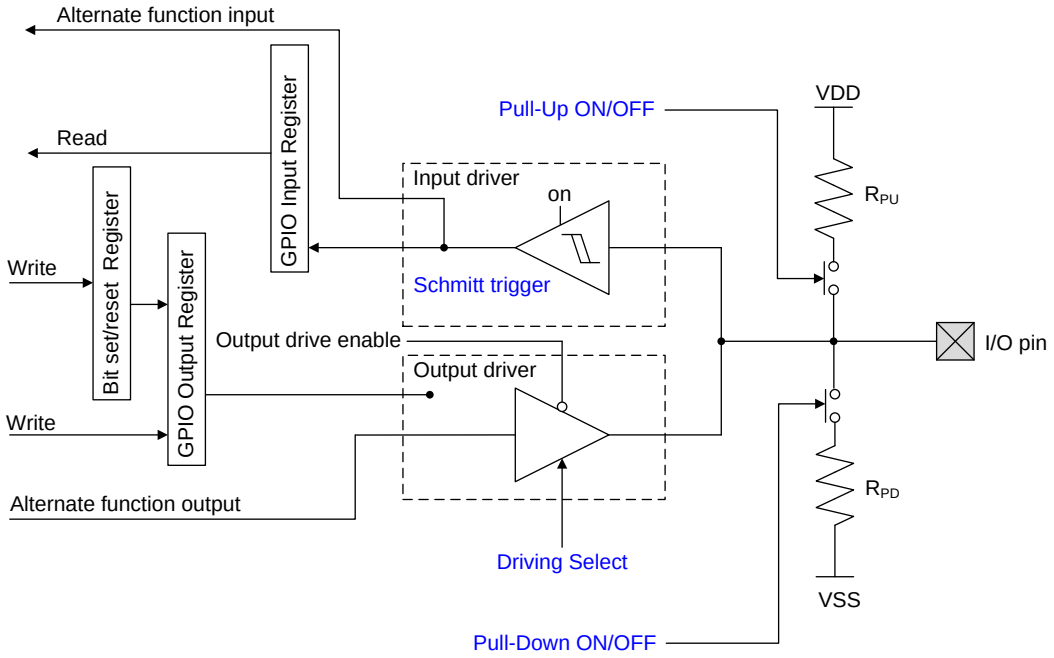


图 6-4 I/O 端口位的复用配置

6. 4. 12 模拟配置

当 I/O 端口配置为模拟模式:

- 输出缓冲器关闭。
- 施密特触发器输入关闭。并对被配置为模拟模式的 I/O 引脚提供"0"至输入寄存器内。
- 硬件会关闭上拉或下拉功能。
- 对输入寄存器进行读操作，将会得到"0"。

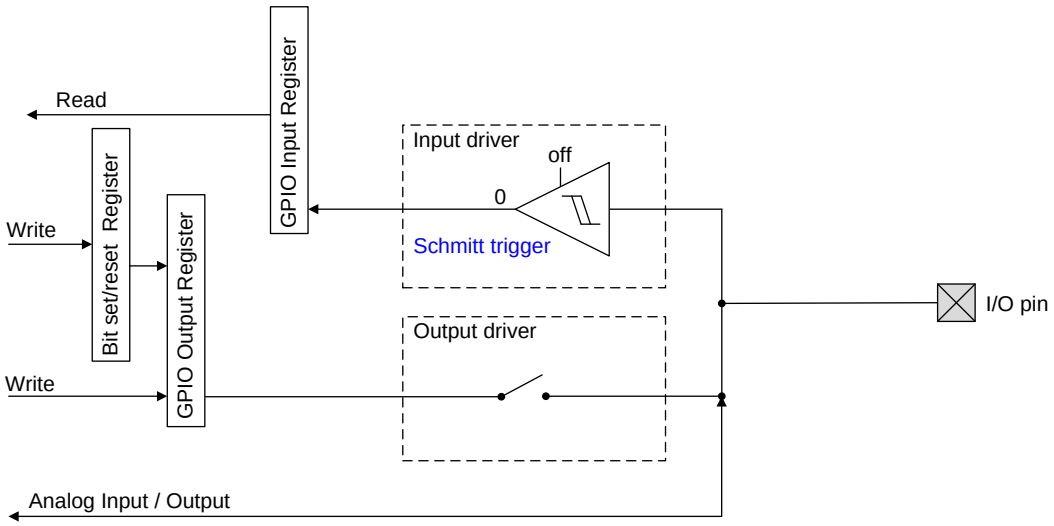


图 6-5 I/O 端口位的模拟配置

6. 4. 13 将HOSC与LOSC晶振引脚配置为通用I/Os

当系统复位时，PB04、PB05 与 PB06、PB07 将会被配置模拟模式 (相应名称为 LOSCI、LOSCO 与 HOSCI、HOSCO)。

若用户不需要使用 HOSC 或 LOSC 时，可将对应的引脚切换为通用 I/O 端口，并且确保不需要使用的 HOSC 或 LOSC 的开关为关闭状态。可在 RCU_CON 寄存器内确认 HOSC 的开关，RCU_LCON 寄存器内确认 LOSC 的开关。

6.5 特殊功能寄存器

6.5.1 寄存器列表

GPIO 寄存器列表			
名称	偏移地址	类型	描述
GPIOx_ID	0000_H	R	GPIOx 端口输入数据寄存器
GPIOx_OD	0004_H	R/W	GPIOx 端口输出数据寄存器
GPIOx_BSBR	0008_H	W1	GPIOx 端口置位和复位寄存器
GPIOx_LCK	$000C_H$	R/W	GPIOx 端口锁定寄存器
GPIOx_MOD	0010_H	R/W	GPIOx 端口模式寄存器
GPIOx_PUD	0014_H	R/W	GPIOx 端口上拉和下拉寄存器
GPIOx_OT	0018_H	R/W	GPIOx 端口输出类型寄存器
GPIOx_DS	$001C_H$	R/W	GPIOx 端口输出驱动寄存器
GPIOx_FIR	0020_H	R/W	GPIOx 端口滤波寄存器
GPIOx_IST	0024_H	R/W	GPIOx 端口输入类型寄存器
GPIOx_AFL	0028_H	R/W	GPIOx 复用功能低位寄存器
GPIOx_AFH	$002C_H$	R/W	GPIOx 复用功能高位寄存器

6. 5. 2 寄存器描述

6. 5. 2. 1 GPIOx 端口输入数据寄存器 (GPIOx_ID)

GPIOx 端口输入数据寄存器 (GPIOx_ID)																													
偏移地址:0x00																													
复位值:0x0000 0000																													
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	ID15	ID14	ID13	ID12	ID11	ID10	ID9	ID8	ID7	ID6	ID5	ID4	ID3	ID2	ID1	ID0

—	Bits 31-16	—	—
ID15	Bit 15	R	**IDy:端口输入数据 (y = 0...15)** 这些位只读。它们包含相应I/O口的输入值。
ID14	Bit 14	R	
ID13	Bit 13	R	
ID12	Bit 12	R	
ID11	Bit 11	R	
ID10	Bit 10	R	
ID9	Bit 9	R	
ID8	Bit 8	R	
ID7	Bit 7	R	
ID6	Bit 6	R	
ID5	Bit 5	R	
ID4	Bit 4	R	
ID3	Bit 3	R	
ID2	Bit 2	R	
ID1	Bit 1	R	
ID0	Bit 0	R	

6. 5. 2. 2 GPIOx 端口输出数据寄存器(GPIOx_OD)

GPIOx 端口输出数据寄存器 (GPIOx_OD)																													
偏移地址:0x04																													
复位值:0x0000 0000																													
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																OD15	OD14	OD13	OD12	OD11	OD10	OD9	OD8	OD7	OD6	OD5	OD4	OD3	OD2	OD1	OD0

—	Bits 31-16	—	—
OD15	Bit 15	R/W	**ODy:端口输出数据 (y = 0..15)** 这些位可由软件读写。 注:对于单独位的设置/清除，可单独对GPIOx_BSBR寄存器操作来实现。
OD14	Bit 14	R/W	
OD13	Bit 13	R/W	
OD12	Bit 12	R/W	
OD11	Bit 11	R/W	
OD10	Bit 10	R/W	
OD9	Bit 9	R/W	
OD8	Bit 8	R/W	
OD7	Bit 7	R/W	
OD6	Bit 6	R/W	
OD5	Bit 5	R/W	
OD4	Bit 4	R/W	
OD3	Bit 3	R/W	
OD2	Bit 2	R/W	
OD1	Bit 1	R/W	
OD0	Bit 0	R/W	

6. 5. 2. 3 GPIOx 端口置位和复位寄存器 (GPIOx_BSBR)

GPIOx 端口置位和复位寄存器(GPIOx_BSBR)																														
偏移地址:0x08																														
复位值:0x0000 0000																														
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0	BS15	BS14	BS13	BS12	BS11	BS10	BS9	BS8	BS7	BS6	BS5	BS4	BS3	BS2	BS1	BS0

BR15	Bit 31	W1	**BRy:端口 x 复位位。(y= 0..15)** 这些位只写。读这些位时返回0x0000数值。 0:对相应的ODx位无影响。 1:复位相应的ODx位。 注: 若BSy以及BRy同时被设定，BSy有较高的优先权。
BR14	Bit 30	W1	
BR13	Bit 29	W1	
BR12	Bit 28	W1	
BR11	Bit 27	W1	
BR10	Bit 26	W1	
BR9	Bit 25	W1	
BR8	Bit 24	W1	
BR7	Bit 23	W1	
BR6	Bit 22	W1	
BR5	Bit 21	W1	
BR4	Bit 20	W1	
BR3	Bit 19	W1	
BR2	Bit 18	W1	
BR1	Bit 17	W1	
BR0	Bit 16	W1	
BS15	Bit 15	W1	**BSy:端口 x 设置位。(y= 0..15)** 这些位只写。读这些位时返回0x0000数值。 0:对相应的ODx位无影响。 1:置位相应的ODx位。
BS14	Bit 14	W1	
BS13	Bit 13	W1	
BS12	Bit 12	W1	
BS11	Bit 11	W1	
BS10	Bit 10	W1	
BS9	Bit 9	W1	
BS8	Bit 8	W1	
BS7	Bit 7	W1	
BS6	Bit 6	W1	
BS5	Bit 5	W1	
BS4	Bit 4	W1	
BS3	Bit 3	W1	
BS2	Bit 2	W1	

BS1	Bit 1	W1	
BS0	Bit 0	W1	

6. 5. 2. 4 GPIOx 端口锁定寄存器 (GPIOx_LCK)

GPIOx 端口锁定寄存器(GPIOx_LCK)																															
偏移地址:0x0C																															
复位值:0x0000 0000																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LCKK<15:0>																LCK15	LCK14	LCK13	LCK12	LCK11	LCK10	LCK9	LCK8	LCK7	LCK6	LCK5	LCK4	LCK3	LCK2	LCK1	LCK0

LCKK	Bit 31-16	R/W	**LCKK:锁定键** 当执行正确的锁定流程，此寄存器用于锁定端口的配置。 LCKK[0]可随时读取(LCKK[15:1]只能写，读为0x0000)。他仅能由锁定流程来改写。 0:端口配置锁定未启动。 1:端口配置锁定已启动。GPIOx_LCK寄存器锁定直到下一个MCU复位产生。 锁定流程: 锁定流程如下，需写两次相同数值，每次以32位数值写入，LCK[31:16]必须为LCK[15:0]取反值 WR GPIOx_LCK = (~LCK[15:0]<<16) + LCK[15:0] WR GPIOx_LCK = (~LCK[15:0]<<16) + LCK[15:0] RD GPIOx_LCK[16] = '1' (此流程为确认锁定功能是否开启)
LCK15	Bit 15	R/W	**LCKy:端口 x 锁定位。(y= 0..15)** 这些位可读/写，但仅LCKK为 '0' 时写。冻结的寄存器包括GPIOx_MOD、GPIOx_PUD、GPIOx_OT 、 GPIOx_DS 、 GPIOx_FIR 、GPIOx_IST、GPIOx_AFL和GPIOx_AFH。 0:端口配置未锁定。 1:端口配置锁定。
LCK14	Bit 14	R/W	
LCK13	Bit 13	R/W	
LCK12	Bit 12	R/W	
LCK11	Bit 11	R/W	
LCK10	Bit 10	R/W	
LCK9	Bit 9	R/W	
LCK8	Bit 8	R/W	

LCK7	Bit 7	R/W	
LCK6	Bit 6	R/W	
LCK5	Bit 5	R/W	
LCK4	Bit 4	R/W	
LCK3	Bit 3	R/W	
LCK2	Bit 2	R/W	
LCK1	Bit 1	R/W	
LCK0	Bit 0	R/W	

6. 5. 2. 5 GPIOx 端口模式寄存器 (GPIOx_MOD)

GPIOx 端口模式寄存器(GPIOx_MOD)

偏移地址:0x10

复位值:

0xFFFF FFFA (GPIOB)

0xFFFF FFFF (其它端口)

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
MOD15<1:0>		MOD14<1:0>		MOD13<1:0>		MOD12<1:0>		MOD11<1:0>		MOD10<1:0>		MOD9<1:0>		MOD8<1:0>		MOD7<1:0>		MOD6<1:0>		MOD5<1:0>		MOD4<1:0>		MOD3<1:0>		MOD2<1:0>		MOD1<1:0>		MOD0<1:0>	

MOD15	Bits 31-30	R/W	**MODy:端口 x 配置位。(y = 0...15)** 这些位可由软件写入来配置I/O端口模式。 00:通用输入模式。 01:通用输出模式。 10:复用功能模式。 11:模拟模式。(复位状态)
MOD14	Bits 29-28	R/W	
MOD13	Bits 27-26	R/W	
MOD12	Bits 25-24	R/W	
MOD11	Bits 23-22	R/W	
MOD10	Bits 21-20	R/W	
MOD9	Bits 19-18	R/W	
MOD8	Bits 17-16	R/W	
MOD7	Bits 15-14	R/W	
MOD6	Bits 13-12	R/W	
MOD5	Bits 11-10	R/W	
MOD4	Bits 9-8	R/W	
MOD3	Bits 7-6	R/W	
MOD2	Bits 5-4	R/W	
MOD1	Bits 3-2	R/W	
MOD0	Bits 1-0	R/W	

6. 5. 2. 6 GPIOx 端口上拉和下拉寄存器 (GPIOx_PUD)

GPIOx 端口上拉和下拉寄存器(GPIOx_PUD)																													
偏移地址:0x14																													
复位值: 0x0000 0009(GPIOB) 0x0000 0000 (其它端口)																													
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
PUD15<1:0>		PUD14<1:0>		PUD13<1:0>		PUD12<1:0>		PUD11<1:0>		PUD10<1:0>		PUD9<1:0>		PUD8<1:0>		PUD7<1:0>		PUD6<1:0>		PUD5<1:0>		PUD4<1:0>		PUD3<1:0>		PUD2<1:0>		PUD1<1:0>		PUD0<1:0>	

PUD15	Bits 31-30	R/W	**PUDy:端口 x 配置位。(y = 0..15)** 这些位可由软件写入来配置I/O端口为上拉或下拉。 00:无上拉和下拉 (复位状态)。 01:上拉。 10:下拉。 11:保留。
PUD14	Bits 29-28	R/W	
PUD13	Bits 27-26	R/W	
PUD12	Bits 25-24	R/W	
PUD11	Bits 23-22	R/W	
PUD10	Bits 21-20	R/W	
PUD9	Bits 19-18	R/W	
PUD8	Bits 17-16	R/W	
PUD7	Bits 15-14	R/W	
PUD6	Bits 13-12	R/W	
PUD5	Bits 11-10	R/W	
PUD4	Bits 9-8	R/W	
PUD3	Bits 7-6	R/W	
PUD2	Bits 5-4	R/W	
PUD1	Bits 3-2	R/W	
PUD0	Bits 1-0	R/W	

6. 5. 2. 7 GPIOx 端口输出类型寄存器 (GPIOx_OT)

GPIOx 端口输出类型寄存器(GPIOx_OT)																														
偏移地址:0x18																														
复位值:0x0000 0000																														
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																OT15	OT14	OT13	OT12	OT11	OT10	OT9	OT8	OT7	OT6	OT5	OT4	OT3	OT2	OT1	OT0

—	Bits 31-16	—	—
OT15	Bit 15	R/W	**OTy:端口 x 配置位。(y = 0..15)** 这些位可由软件写入来配置I/O端口的输出类型。 0:推挽输出。(复位状态) 1:开漏输出。
OT14	Bit 14	R/W	
OT13	Bit 13	R/W	
OT12	Bit 12	R/W	
OT11	Bit 11	R/W	
OT10	Bit 10	R/W	
OT9	Bit 9	R/W	
OT8	Bit 8	R/W	
OT7	Bit 7	R/W	
OT6	Bit 6	R/W	
OT5	Bit 5	R/W	
OT4	Bit 4	R/W	
OT3	Bit 3	R/W	
OT2	Bit 2	R/W	
OT1	Bit 1	R/W	
OT0	Bit 0	R/W	

6. 5. 2. 8 GPIOx 端口输出驱动寄存器 (GPIOx_DS)

GPIOx 端口输出驱动寄存器(GPIOx_DS)																														
偏移地址:0x1C																														
复位值:0x0000 0000																														
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																DS15	DS14	DS13	DS12	DS11	DS10	DS9	DS8	DS7	DS6	DS5	DS4	DS3	DS2	DS1	DS0

—	Bits 31-16	—	—
DS15	Bit 15	R/W	**DSy:端口 x 配置位。(y = 0..15)** 这些位可由软件写入来配置I/O端口的输出驱动电流。 0:level 0。(复位状态) 1:level 1。 注: 驱动电流可以参照数据手册。
DS14	Bit 14	R/W	
DS13	Bit 13	R/W	
DS12	Bit 12	R/W	
DS11	Bit 11	R/W	
DS10	Bit 10	R/W	
DS9	Bit 9	R/W	
DS8	Bit 8	R/W	
DS7	Bit 7	R/W	
DS6	Bit 6	R/W	
DS5	Bit 5	R/W	
DS4	Bit 4	R/W	
DS3	Bit 3	R/W	
DS2	Bit 2	R/W	
DS1	Bit 1	R/W	
DS0	Bit 0	R/W	

6. 5. 2. 9 GPIOx 端口滤波寄存器 (GPIOx_FIR)

GPIOx 端口滤波寄存器 (GPIOx_FIR)																														
偏移地址:0x20																														
复位值:0x0000 0000																														
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																FIR15	FIR14	FIR13	FIR12	FIR11	FIR10	FIR9	FIR8	FIR7	FIR6	FIR5	FIR4	FIR3	FIR2	FIR1	FIR0

—	Bits 31-16	—	—
FIR15	Bit 15	R/W	**FIRy:端口 x 配置位。(y = 0..15)** 这些位可由软件写来配置输入是否要通过滤波。 0:旁路。(复位状态) 1:消除一定脉冲宽度的毛刺。 注: 滤波的参数可以参照数据参考手册。
FIR14	Bit 14	R/W	
FIR13	Bit 13	R/W	
FIR12	Bit 12	R/W	
FIR11	Bit 11	R/W	
FIR10	Bit 10	R/W	
FIR9	Bit 9	R/W	
FIR8	Bit 8	R/W	
FIR7	Bit 7	R/W	
FIR6	Bit 6	R/W	
FIR5	Bit 5	R/W	
FIR4	Bit 4	R/W	
FIR3	Bit 3	R/W	
FIR2	Bit 2	R/W	
FIR1	Bit 1	R/W	
FIR0	Bit 0	R/W	

6. 5. 2. 10 GPIOx 端口输入类型寄存器 (GPIOx_IST)

GPIOx 端口输入类型寄存器(GPIOx_IST)																													
偏移地址:0x24																													
复位值:0x0000 0000																													
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	IST15	IST14	IST13	IST12	IST11	IST10	IST9	IST8	IST7	IST6	IST5	IST4	IST3	IST2	IST1	IST0

—	Bits 31-16	—	—
IST15	Bit 15	R/W	**ISTy:端口 x 配置位。(y = 0..15)** 这些位可由软件写来配置输入施密特触发器。 0: TTL IO电平。 1: CMOS IO电平。
IST14	Bit 14	R/W	
IST13	Bit 13	R/W	
IST12	Bit 12	R/W	
IST11	Bit 11	R/W	
IST10	Bit 10	R/W	
IST9	Bit 9	R/W	
IST8	Bit 8	R/W	
IST7	Bit 7	R/W	
IST6	Bit 6	R/W	
IST5	Bit 5	R/W	
IST4	Bit 4	R/W	
IST3	Bit 3	R/W	
IST2	Bit 2	R/W	
IST1	Bit 1	R/W	
IST0	Bit 0	R/W	

6. 5. 2. 11 GPIOx 复用功能低位寄存器 (GPIOx_AFL)

GPIOx 复用功能低位寄存器(GPIOx_AFL)																													
偏移地址:0x28																													
复位值:0x0000 0000																													
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AFSEL7<3:0>				AFSEL6<3:0>				AFSEL5<3:0>				AFSEL4<3:0>				AFSEL3<3:0>				AFSEL2<3:0>				AFSEL1<3:0>				AFSEL0<3:0>			

AFSEL7	Bits 31-28	R/W	**AFSELy:端口 x 位 y 的复用功能选择。(y=0…7)** 这些位可由软件写入来配置复用功能 I/O。 AFSELy 选择如下: 0000: AFSEL0 0001: AFSEL1 0010: AFSEL2 0011: AFSEL3 0100: AFSEL4 0101: AFSEL5 0110: AFSEL6 0111: AFSEL7 1000: AFSEL8 1XXX:保留
AFSEL6	Bits 27-24	R/W	
AFSEL5	Bits 23-20	R/W	
AFSEL4	Bits 19-16	R/W	
AFSEL3	Bits 15-12	R/W	
AFSEL2	Bits 11-8	R/W	
AFSEL1	Bits 7-4	R/W	
AFSEL0	Bits 3-0	R/W	

6. 5. 2. 12 GPIOx 复用功能高位寄存器 (GPIOx_AFH)

GPIOx 复用功能高位寄存器(GPIOx_AFH)																													
偏移地址:0x2C																													
复位值:0x0000 0000																													
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AFSEL15<3:0>				AFSEL14<3:0>				AFSEL13<3:0>				AFSEL12<3:0>				AFSEL11<3:0>				AFSEL10<3:0>				AFSEL9<3:0>				AFSEL8<3:0>			

AFSEL15	Bits 31-28	R/W	**AFSELy:端口 x 位 y 的复用功能选择。(y=8…15)** 这些位可由软件写入来配置复用功能 I/O。 AFSELy 选择如下: 0000: AFSEL0 0001: AFSEL1 0010: AFSEL2 0011: AFSEL3 0100: AFSEL4 0101: AFSEL5 0110: AFSEL6 0111: AFSEL7 1000: AFSEL8 1XXX:保留
AFSEL14	Bits 27-24	R/W	
AFSEL13	Bits 23-20	R/W	
AFSEL12	Bits 19-16	R/W	
AFSEL11	Bits 15-12	R/W	
AFSEL10	Bits 11-8	R/W	
AFSEL9	Bits 7-4	R/W	
AFSEL8	Bits 3-0	R/W	

第7章 外设互联(PIS)

7.1 概述

PIS(外围设备互连系统)是微控制器中用作外围设备互连的桥梁接口，可实现外围设备之间的相互触发、控制和自动化工作，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，为各种应用提供便捷。

多个外围设备之间直接连接。这可以在外围设备之间实现自动通信或同步，节省了 CPU 资源，进而降低功耗。此外，这些硬件连接消除了软件延迟。

7.2 连接总览

源	目标												
	AD16C6T1	AD16C6T2	GP32C4T1	GP32C4T2	GP16C2T1	GP16C2T2	BS16T1	CMP1	CMP2	ADC1	ADC2	OPAMP	IR
AD16C6T1	—	1	1	1	1	1	—	2	2	3	3	4	—
AD16C6T2	1	—	1	1	1	1	—	2	2	3	3	4	—
GP32C4T1	1	1	—	1	1	1	—	2	2	3	3	—	—
GP32C4T2	1	1	1	—	1	1	—	2	2	3	3	—	—
GP16C2T1	1	1	1	1	—	1	—	2	2	3	3	—	5
GP16C2T2	1	1	1	1	1	—	—	2	2	3	3	—	5
BS16T1	—	—	—	—	—	—	—	—	—	3	3	—	—
CMP1	6	6	6	6	6	6	—	—	—	—	—	—	—
CMP2	6	6	6	6	6	6	—	—	—	—	—	—	—
ADC1	7	—	7	—	—	—	—	—	—	—	—	—	—
ADC2	—	7	—	7	—	—	—	—	—	—	—	—	—
OPAMP	—	—	—	—	—	—	—	—	—	11	11	—	—
V_{TSENSE}	—	—	—	—	—	—	—	—	—	11	—	—	—
V_{REFINT}	—	—	—	—	—	—	—	10	10	11	—	—	—
V_{RES1}	—	—	—	—	—	—	—	10	10	11	—	12	—
V_{RES2}	—	—	—	—	—	—	—	10	10	—	11	12	—
MCO	—	—	—	—	—	8	—	—	—	—	—	—	—
HOSC/32	—	—	—	—	—	8	—	—	—	—	—	—	—
LRC	—	—	—	—	—	8	—	—	—	—	—	—	—
SYSERR	9	9	9	9	9	9	—	—	—	—	—	—	—

表 7-1 互连矩阵

注:

1. 白色单元格中的数字表示说明位于互连描述中对应的章节。
2. 灰色单元格中的“—”符号表示没有互连。

7.3 互连描述

7.3.1 定时器互连

定时器输入触发信号	定时器输入触发源分配					
	AD16C6T1	AD16C6T2	GP32C4T1	GP32C4T2	GP16C2T1	GP16C2T2
IT0	—	AD16C6T1	AD16C6T1	AD16C6T1	AD16C6T1	AD16C6T1
IT1	GP32C4T1	GP32C4T1	—	GP32C4T1	GP32C4T1	GP32C4T1
IT2	GP32C4T2	GP32C4T2	GP32C4T2	—	GP32C4T2	GP32C4T2
IT3	—	—	—	—	—	—
IT4	—	—	—	—	—	—
IT5	AD16C6T2	—	AD16C6T2	AD16C6T2	AD16C6T2	AD16C6T2
IT6	GP16C2T1	GP16C2T1	GP16C2T1	GP16C2T1	—	GP16C2T1
IT7	GP16C2T2	GP16C2T2	GP16C2T2	GP16C2T2	GP16C2T2	—

表 7-2 定时器互连

某些定时器从内部连接在一起，以实现定时器同步或链接。

当某个定时器配置为主模式时，可以对另一个配置为从模式的定时器的计数器执行复位、启动、停止操作或为其提供时钟。

详细说明可参考定时器(AD16C6T/GP32C4T/GP16C2T)"定时器同步"章节。

触发信号

在主模式定时器预配置事件发生后，触发信号由主模式定时器 TRGOUT 输出，并输入到从模式定时器的 ITx(x=0～7)。

7.3.2 从定时器到比较器

定时器可用作 CMP 的消隐窗口输入。

BLANKSEL[5:0]	CMP1	CMP2
xxxxx1	AD16C6T1_CH6	
xxxx1x	AD16C6T2_CH6	
xxx1xx	GP32C4T1_CH4	
xx1xxx	GP32C4T2_CH4	
x1xxxx	GP16C2T1_CH2	
1xxxxx	GP16C2T2_CH2	

表 7-3 时器到比较器

详细说明可参考比较器"CMP 消隐功能"章节。

触发信号

定时器输出信号是 CMP 的消隐源输入。

7. 3. 3 从定时器、EXTI到ADC

定时器和 EXTI 可用于生成 ADC 触发事件。

	ADC Regular	ADC Inject
ADC EXT TRG0	AD16C6T1_CH1	AD16C6T1_TRGOUT
ADC EXT TRG1	AD16C6T1_CH2	AD16C6T1_CH4
ADC EXT TRG2	AD16C6T1_CH3	GP32C4T1_TRGOUT
ADC EXT TRG3	GP32C4T1_CH2	GP32C4T1_CH1
ADC EXT TRG4	GP32C4T2_TRGOUT	GP32C4T2_CH4
ADC EXT TRG5	AD16C6T1_CH4	GP32C4T2_CH2
ADC EXT TRG6	EXTI_0	EXTI_1
ADC EXT TRG7	AD16C6T2_TRGOUT	AD16C6T2_CH4
ADC EXT TRG8	AD16C6T2_TRGOUT2	AD16C6T1_TRGOUT2
ADC EXT TRG9	AD16C6T1_TRGOUT	AD16C6T2_TRGOUT
ADC EXT TRG10	AD16C6T1_TRGOUT2	AD16C6T2_TRGOUT2
ADC EXT TRG11	GP32C4T1_TRGOUT	GP32C4T2_CH3
ADC EXT TRG12	AD16C6T2_CH4	GP32C4T2_TRGOUT
ADC EXT TRG13	BS16T_TRGOUT	GP32C4T2_CH1
ADC EXT TRG14	GP16C2T1_TRGOUT	BS16T_TRGOUT
ADC EXT TRG15	GP32C4T2_CH4	GP16C2T1_TRGOUT
ADC EXT TRG16	GP16C2T1_CH1	GP16C2T1_CH1
ADC EXT TRG17	GP16C2T1_CH2	GP16C2T1_CH2
ADC EXT TRG18	AD16C6T2_CH1	AD16C6T2_CH1
ADC EXT TRG19	AD16C6T2_CH2	AD16C6T2_CH2
ADC EXT TRG20	AD16C6T2_CH3	AD16C6T2_CH3
ADC EXT TRG21	GP16C2T2_CH1	Reserved
ADC EXT TRG22	GP16C2T2_CH2	GP16C2T2_CH2
ADC EXT TRG23	GP16C2T2_TRGOUT	GP16C2T2_TRGOUT
ADC EXT TRG24	GP32C4T1_CH1	GP32C4T1_CH2
ADC EXT TRG25	GP32C4T1_CH3	GP32C4T1_CH3
ADC EXT TRG26	GP32C4T1_CH4	GP32C4T1_CH4
ADC EXT TRG27	Reserved	GP16C2T2_CH1
ADC EXT TRG28	GP32C4T2_CH1	AD16C6T1_CH1
ADC EXT TRG29	GP32C4T2_CH2	AD16C6T1_CH2
ADC EXT TRG30	GP32C4T2_CH3	AD16C6T1_CH3
ADC EXT TRG31	Reserved	Reserved

表 7-4 从定时器、EXTI 到 ADC

详细说明可参考 ADC"外部触发转换和触发极性"章节。

触发信号

触发信号由定时器的 TRGOUT、CHx(x=1~4)或 EXTI 输出，并输入到 ADC 的 adc_rext_trgx 或 adc_jext_trgx(x=0~31)信号上。

7. 3. 4 从定时器到OPAMP

定时器 AD16C6T1_CH6 和 AD16C6T2_CH6 的输出，可用来控制 OPAMP 反相和非反相的输入选择。

详细说明可参考 OPAMP"定时器控制多任务器模式"章节。

7. 3. 5 从定时器到IR

定时器 GP16C2T1_CH1 和 GP16C2T2_CH1 的输出，可用来生成红外线 IR 输出波形。

详细说明可参考 GP16C2T"红外线(IR)控制信号"章节。

7. 3. 6 从比较器到定时器

比较器的输出可作为定时器(AD16C6T1、AD16C6T2、GP16C2T1、GP16C2T2)的刹车信号源。

详细说明可参考定时器"刹车功能"章节。

7. 3. 7 从ADC到定时器

ADC 可通过看门狗信号向定时器(AD16C6T1、AD16C6T2、GP32C4T1、GP32C4T2)提供触发事件。

ADC_AWD	AD16C6T1	AD16C6T2	GP32C4T1	GP32C4T2
ADC1_AWD1	AD16C6T1_ETR	—	GP32C4T1_ETR	—
ADC1_AWD2	AD16C6T1_ETR	—	GP32C4T1_ETR	—
ADC1_AWD3	AD16C6T1_ETR	—	GP32C4T1_ETR	—
ADC2_AWD1	—	AD16C6T2_ETR	—	GP32C4T2_ETR
ADC2_AWD2	—	AD16C6T2_ETR	—	GP32C4T2_ETR
ADC2_AWD3	—	AD16C6T2_ETR	—	GP32C4T2_ETR

表 7-5 ADC 到定时器

详细说明可参考 ADC"模拟看门狗"章节。

7. 3. 8 从时钟源到定时器

定时器 GP16C2T2 的 CH1 输入可以选择微控制器输出时钟(MCOCLK)、内部低速 32kHz RC 振荡器(LRC)，或外部高速振荡器 32 分频(HOSC/32)作为捕获信号源。

定时器允许进行校准或精确测量内部时钟(如 HRC 或 LRC)，以及使用精确的时钟(如 HOSC)作为参考时序。

7. 3. 9 从系统錯誤到定时器

MCU 内部故障事件可作为定时器(AD16C6T1、AD16C6T2、GP16C2T1、GP16C2T2)的刹车信号源。

MCU 内部故障事件：

- 外部高速振荡器时钟安全系统(HOSC CSS)生成的时钟故障事件
- LVD 输出
- Cortex-M0 LOCKUP(硬故障)输出

详细说明可参考定时器"刹车功能"章节。

7. 3. 10 从内部模拟源到比较器

内部参考电压 V_{REFINT}、内部电阻分压 V_{RES1}、V_{RES2} 可作为比较器负端的输入信号源。

INNSEL[2:0]	CMP1_INN	CMP2_INN
100	V_{RES1}	
101	V_{RES2}	
110	V_{REFINT}	

表 7-6 内部模拟源到比较器

详细说明可参考"比较器"以及"电阻分压电路控制器"章节。

7. 3. 11 从内部模拟源、OPAMP到ADC

温度传感器 V_{TSENSE}、内部参考电压 V_{REFINT}、内部电阻分压 V_{RES1}、V_{RES2} 和运算放大器 OPAMPx(x=1～4)可作为 ADC 的输入信号源。

ADC 通道	ADC1	ADC2
IN0	ADC12_IN0	ADC12_IN0
IN1	ADC12_IN1	ADC12_IN1
IN2	ADC12_IN2	ADC12_IN2
IN3	ADC12_IN3	ADC12_IN3
IN4	ADC1_IN4	ADC2_IN4
IN5	ADC1_IN5	ADC2_IN5
IN6	ADC1_IN6	ADC2_IN6
IN7	ADC1_IN7	ADC2_IN7
IN8	ADC1_IN8	ADC2_IN8
IN9	ADC1_IN9	ADC2_IN9
IN10	ADC1_IN10	ADC2_IN10
IN11	ADC1_IN11	ADC2_IN11
IN12	V_{OPA1_OUT}	V_{OPA1_OUT}
IN13	V_{OPA2_OUT}	V_{OPA2_OUT}
IN14	V_{OPA3_OUT}	V_{OPA3_OUT}

ADC 通道	ADC1	ADC2
IN15	V_{OPA4_OUT}	V_{OPA4_OUT}
IN16	V_{TSENSE}	ADC2_IN16
IN17	V_{REFINT}	ADC2_IN17
IN18	V_{RES1}	V_{RES2}

表 7-7　从内部模拟源、OPAMP 到 ADC

详细说明可参考"ADC"、"比较器"以及"电阻分压电路控制器"章节。

7. 3. 12　从内部模拟源到OPAMP

内部电阻分压 V_{RES1}、V_{RES2} 可作为 OPAMP 正端的输入信号源。

正端信号	OPAMP1	OPAMP2	OPAMP3	OPAMP4
VINP2	V_{RES1}			
VINP3	V_{RES2}			

表 7-8　内部模拟源到 OPAMP

详细说明可参考"运算放大器 OPAMP"章节。

第8章 运算加速器 (CALC)

8.1 概述

运算加速器(CALC)可以执行平方根和除法的运算加速。

8.2 特性

- 支持最大 32 位无符号数平方根运算
- 支持最大 32 位有符号数或无符号数除法运算
- 除零警示标志位
- 支持根据运算数值大小，运算时间自动调整为 2~17 个 HCLK 时钟周期
- 支持固定运算时间(17 HCLK)
- 支持使用 DMA 写数据执行运算操作

8.3 功能描述

8.3.1 平方根运算

8.3.1.1 算法概述

无符号数平方根算法可对最大 32 位的无符号数进行平方根运算，运算结果最大为 16 位无符号数。若理论平方根值中含有小数部分，则硬件在计算时会舍去小数，即向 0 的方向取最大整数。硬件运算电路中带有预判决功能，可根据被开方数寄存器 **CALC_RDCND** 的数量级，自动选取最小的计算时间。

8.3.1.2 使用说明

当被开方数寄存器 **CALC_RDCND** 发生写入动作时，平方根运算即开始，开始时平方根运算标志位 **CALC_STAT**.BUSY 被置位。

当软件检测到 **CALC_STAT**.BUSY 被硬件清零时，表示平方根运算已经完成。通过读取平方根运算结果寄存器 **CALC_SQRTRES** 可获得被开方数的平方根近似值。

当运算标志位 **CALC_STAT**.BUSY 位为 1 时，软件读取平方根运算结果寄存器 **CALC_SQRTRES** 将会停止软件的指令，直到 **CALC_STAT**.BUSY 被硬件清零时，运算加速器返回开方数的平方根近似值，并解除停止软件指令。

若当运算还未完成，被开方数寄存器 **CALC_RDCND** 发生写入动作时，硬件会重新开始新的运算，原先的运算结果将被丢弃。

为使得运算的结果更精确，在操作上可采取移位元的方式来减小运算的误差。

例如，需要运算 X 的平方根，由于 X 较小，若直接写入被开方数寄存器 **CALC_RDCND** 中，

产生的结果误差较大。可以先将 X 进行左移 n 位(n 需为偶数)，将 X 左移后可得新的被开方数即$X' = X * 2^n$，将 X' 写入被开方数寄存器 **CALC_RDCND**，得到运算结果 Y'，可知$Y' = \sqrt{X'} = \sqrt{(X * 2^n)} = 2^{(n/2)} * \sqrt{X}$，可将 **CALC_SQRTRES** 中的结果右移 n/2 位后，即得到 X 的平方根$Y = \sqrt{X} = Y'/2^{(n/2)}$。

原先 X 允许的位数 m 最大可至 32 位，当 X 的实际位数没有 32 位时，可适当的调整 n 的位数以最大程度的利用平方根运算器的性能。n 值越大，运算结果越精确。

以计算 2 的平方根为例。$\sqrt{2}$= 1.4142135623731。

Radicand (Hex)	Radicand 格式	Result(Hex)	Result(Dec)	误差(%)
0x0000 0002	m=32, n=0	0x0000 0001	1.0	-29.289
0x0000 0020	m=28, n=4	0x0000 0005	1.25	-11.612
0x0000 0200	m=24, n=8	0x0000 0016	1.3750	-2.773
0x0000 2000	m=20, n=12	0x0000 005A	1.406250	-0.563
0x0002 0000	m=16, n=16	0x0000 016A	1.41406250	-0.011
0x0020 0000	m=12, n=20	0x0000 05A8	1.41406250	-0.011
0x0200 0000	m=8, n=24	0x0000 16A0	1.41406250	-0.011
0x2000 0000	m=4, n=28	0x0000 5A82	1.4141845703	-0.002
0x8000 0000	m=2, n=30	0x0000 B504	1.4141845703	-0.002

表 8-1　平方根运算误差示例

8. 3. 1. 3　　完成时间

- CALC_DIVCON 寄存器的 FIXED_SPEED 位为 1 时
  平方根算法所需的运算时间固定为 17 个时钟个数。
- CALC_DIVCON 寄存器的 FIXED_SPEED 位为 0 时
  平方根算法根据被开方数 CALC_RDCND 寄存器中输入的数值大小所运算的时间不同，数值大小所运算的时间可根据下表所示。

CALC_RDCND [31:0]		运算时间 (BUSY=1 的时钟个数)
二进制	十进制	
1xxx_xxxx_xxxx_xxxx_xxxx_xxxx_xxxx_xxxx 01xx_xxxx_xxxx_xxxx_xxxx_xxxx_xxxx_xxxx	4,294,967,295 ~1,073,741,824	17
001x_xxxx_xxxx_xxxx_xxxx_xxxx_xxxx_xxxx 0001_xxxx_xxxx_xxxx_xxxx_xxxx_xxxx_xxxx	1,073,741,823 ~ 268,435,456	16
0000_1xxx_xxxx_xxxx_xxxx_xxxx_xxxx_xxxx 0000_01xx_xxxx_xxxx_xxxx_xxxx_xxxx_xxxx	268,435,455 ~ 67,108,864	15
0000_001x_xxxx_xxxx_xxxx_xxxx_xxxx_xxxx 0000_0001_xxxx_xxxx_xxxx_xxxx_xxxx_xxxx	67,108,863 ~ 16,777,216	14
0000_0000_1xxx_xxxx_xxxx_xxxx_xxxx_xxxx 0000_0000_01xx_xxxx_xxxx_xxxx_xxxx_xxxx	16,777,215 ~ 4,194,304	13
0000_0000_001x_xxxx_xxxx_xxxx_xxxx_xxxx 0000_0000_0001_xxxx_xxxx_xxxx_xxxx_xxxx	4,194,303~ 1,048,576	12

0000_0000_0000_1xxx_xxxx_xxxx_xxxx_xxxx 0000_0000_0000_01xx_xxxx_xxxx_xxxx_xxxx	1,048,575~ 262,144	11
0000_0000_0000_001x_xxxx_xxxx_xxxx_xxxx 0000_0000_0000_0001_xxxx_xxxx_xxxx_xxxx	262,143~ 65,536	10
0000_0000_0000_0000_1xxx_xxxx_xxxx_xxxx 0000_0000_0000_0000_01xx_xxxx_xxxx_xxxx	65,535~ 16,384	9
0000_0000_0000_0000_001x_xxxx_xxxx_xxxx 0000_0000_0000_0000_0001_xxxx_xxxx_xxxx	16,383~ 4,096	8
0000_0000_0000_0000_0000_1xxx_xxxx_xxxx 0000_0000_0000_0000_0000_01xx_xxxx_xxxx	4,095~ 1,024	7
0000_0000_0000_0000_0000_001x_xxxx_xxxx 0000_0000_0000_0000_0000_0001_xxxx_xxxx	1,023~ 256	6
0000_0000_0000_0000_0000_0000_1xxx_xxxx 0000_0000_0000_0000_0000_0000_01xx_xxxx	255~ 64	5
0000_0000_0000_0000_0000_0000_001x_xxxx 0000_0000_0000_0000_0000_0000_0001_xxxx	63~ 16	4
0000_0000_0000_0000_0000_0000_0000_1xxx 0000_0000_0000_0000_0000_0000_0000_01xx	15~ 4	3
0000_0000_0000_0000_0000_0000_0000_00xx	3~ 0	2

表 8-2　平方根运算时间表(CALC_DIVCON.FIXED_SPEED=0)

8. 3. 2　除法运算

8. 3. 2. 1　　算法概述

除法算法可对最大 32 位的有符号数或无符号数进行除法运算，运算结果最大为 32 位有符号数或无符号数。有符号运算中 Bit31 位为符号位，负数使用二进制补码的方式表示。

商的符号由被除数和除数共同决定。当被除数和除数符号相同时，商为正数；当被除数和除数符号不同时，商为负数。余数的符号由被除数的符号决定。

硬件运算电路中带有预判决功能，可根据被除数的数量级，自动选取最小的计算时间。

8. 3. 2. 2　　特例说明

溢出

在 32 位有符号除法运算中，当被除数为 0x8000_0000，除数为 0xFFFF_FFFF 时，即-2^{31}/-1，得到的结果应为 2^{31}，但 32 位有符号数最大可表示的正整数为 2^{31}-1，该次运算结果将溢出。此时硬件计算所得的商为 0x8000_0000，余数为 0x0000_0000。硬件并无标识位指示运算结果是否为溢出。

除数零

在 32 位有符号数或无符号数除法中，若除数设置为 0，则硬件将标志位 **CALC_STAT**.DZ 置 1。硬件计算所得的商与余数皆无意义。

8. 3. 2. 3 使用说明

通过设置 **CALC_DIVCON**.SIGN 位来选择运算的是有符号数还是无符号数。通过设置 **CALC_DIVCON**.TRM 位来选择触发源。根据操作习惯，一般选择最后一个操作数的写入操作作为除法运算的触发源。

- 写入被除数后触发

  在 **CALC_DIVCON**.TRM 位置 0 时，软件先对除数寄存器 **CALC_DIVSR** 写入值，接着对被除数寄存器 **CALC_DIVDR** 写入值。当被除数寄存器 **CALC_DIVDR** 写入值时，除法运算即触发开始，此时除法运算标志位 **CALC_STAT**.BUSY 被置位。

  当软件检测到 **CALC_STAT**.BUSY 被硬件清零时，表示除法运算已经完成。通过读取商寄存器 **CALC_DIVQR** 和余数寄存器 **CALC_DIVRR** 可获得此次除法运算的结果。

  当运算标志位 **CALC_STAT**.BUSY 位为 1 时，软件读取商寄存器 **CALC_DIVQR** 或余数寄存器 **CALC_DIVRR** 时，将会停止软件的指令，直到 **CALC_STAT**.BUSY 被硬件清零时，运算加速器将返回读取商寄存器 **CALC_DIVQR** 或余数寄存器 **CALC_DIVRR** 的数值，并解除停止软件指令。

  若当运算还未完成，除数寄存器 **CALC_DIVDR** 发生写入动作时，硬件会重新开始新的运算，原先的运算结果将被丢弃。

- 写入除数后触发

  在 **CALC_DIVCON**.TRM 位置 1 时，软件先对被除数寄存器 **CALC_DIVDR** 写入值，接着对除数寄存器 **CALC_DIVSR** 写入值。当除数寄存器 **CALC_DIVSR** 写入值时，除法运算即触发开始，此时除法运算标志位 **CALC_STAT**.BUSY 被置位。

  当软件检测到 **CALC_STAT**.BUSY 被硬件清零时，表示除法运算已经完成。通过读取商寄存器 **CALC_DIVQR** 和余数寄存器 **CALC_DIVRR** 可获得此次除法运算的结果。

  当运算标志位 **CALC_STAT**.BUSY 位为 1 时，软件读取商寄存器 **CALC_DIVQR** 或余数寄存器 **CALC_DIVRR** 时，将会停止软件的指令，直到 **CALC_STAT**.BUSY 被硬件清零时，运算加速器将返回读取商寄存器 **CALC_DIVQR** 或余数寄存器 **CALC_DIVRR** 的数值，并解除停止软件指令。

  若当运算还未完成，除数寄存器 **CALC_DIVSR** 发生写入动作时，硬件会重新开始新的运算，原先的运算结果将被丢弃。

8. 3. 2. 4 完成时间

- CALC_DIVCON 寄存器的 FIXED_SPEED 位为 1 时
  除法算法所需的运算时间固定为 17 个时钟个数。
- CALC_DIVCON 寄存器的 FIXED_SPEED 位为 0 时
  除法算法可根据除数 CALC_DIVSR 寄存器中输入数值大小所运算的时间不同，数值大小所运算的时间可根据下表所示。

除数的绝对值(abs(CALC_DIVSR))		运算时间(BUSY=1 的时钟个数)
二进制	十进制	
1xxx_xxxx_xxxx_xxxx_xxxx_xxxx_xxxx_xxxx 01xx_xxxx_xxxx_xxxx_xxxx_xxxx_xxxx_xxxx	4,294,967,295 ~1,073,741,824	2
001x_xxxx_xxxx_xxxx_xxxx_xxxx_xxxx_xxxx 0001_xxxx_xxxx_xxxx_xxxx_xxxx_xxxx_xxxx	1,073,741,823 ~ 268,435,456	3
0000_1xxx_xxxx_xxxx_xxxx_xxxx_xxxx_xxxx 0000_01xx_xxxx_xxxx_xxxx_xxxx_xxxx_xxxx	268,435,455 ~ 67,108,864	4
0000_001x_xxxx_xxxx_xxxx_xxxx_xxxx_xxxx 0000_0001_xxxx_xxxx_xxxx_xxxx_xxxx_xxxx	67,108,863 ~ 16,777,216	5
0000_0000_1xxx_xxxx_xxxx_xxxx_xxxx_xxxx 0000_0000_01xx_xxxx_xxxx_xxxx_xxxx_xxxx	16,777,215 ~ 4,194,304	6
0000_0000_001x_xxxx_xxxx_xxxx_xxxx_xxxx 0000_0000_0001_xxxx_xxxx_xxxx_xxxx_xxxx	4,194,303 ~ 1,048,576	7
0000_0000_0000_1xxx_xxxx_xxxx_xxxx_xxxx 0000_0000_0000_01xx_xxxx_xxxx_xxxx_xxxx	1,048,575~ 262,144	8
0000_0000_0000_001x_xxxx_xxxx_xxxx_xxxx 0000_0000_0000_0001_xxxx_xxxx_xxxx_xxxx	262,143~ 65,536	9
0000_0000_0000_0000_1xxx_xxxx_xxxx_xxxx 0000_0000_0000_0000_01xx_xxxx_xxxx_xxxx	65,535~ 16,384	10
0000_0000_0000_0000_001x_xxxx_xxxx_xxxx 0000_0000_0000_0000_0001_xxxx_xxxx_xxxx	16,383~ 4,096	11
0000_0000_0000_0000_0000_1xxx_xxxx_xxxx 0000_0000_0000_0000_0000_01xx_xxxx_xxxx	4,095~ 1,024	12
0000_0000_0000_0000_0000_001x_xxxx_xxxx 0000_0000_0000_0000_0000_0001_xxxx_xxxx	1,023~ 256	13
0000_0000_0000_0000_0000_0000_1xxx_xxxx 0000_0000_0000_0000_0000_0000_01xx_xxxx	255~ 64	14
0000_0000_0000_0000_0000_0000_001x_xxxx 0000_0000_0000_0000_0000_0000_0001_xxxx	63~ 16	15
0000_0000_0000_0000_0000_0000_0000_1xxx 0000_0000_0000_0000_0000_0000_0000_01xx	15~ 4	16
0000_0000_0000_0000_0000_0000_0000_00xx	3~ 0	17

表 8-3 除法运算时间表

8.4 特殊功能寄存器

8.4.1 寄存器列表

CALC 寄存器列表			
名称	偏移地址	类型	描述
CALC_DIVDR	0000_H	R/W	CALC 被除数寄存器
CALC_DIVSR	0004_H	R/W	CALC 除数寄存器
CALC_DIVQR	0008_H	R	CALC 商寄存器
CALC_DIVRR	$000C_H$	R	CALC 余数寄存器
CALC_DIVCON	0010_H	R/W	CALC 除法运算控制寄存器
CALC_RDCND	0014_H	R/W	CALC 被开方数寄存器
CALC_SQRTRES	0018_H	R	CALC 平方根运算结果寄存器
CALC_STAT	0020_H	R	CALC 运算状态寄存器

8. 4. 2 寄存器描述

8. 4. 2. 1 CALC被除数寄存器 (CALC_DIVDR)

CALC 被除数寄存器 (CALC_DIVDR)																															
偏移地址:0x000																															
复位值:0x0000 0000																															
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
DIVD <31:0>																															

DIVD	Bit 31-0	R/W	**被除数** 32位被除数

8. 4. 2. 2 CALC除数寄存器(CALC_DIVSR)

CALC 除数寄存器(CALC_DIVSR)																															
偏移地址:0x004																															
复位值:0x0000 0000																															
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
DIVS <31:0>																															

DIVS	Bit 31-0	R/W	**除数** 32位除数

8. 4. 2. 3 CALC商寄存器(CALC_DIVQR)

CALC 商寄存器(CALC_DIVQR)																															
偏移地址:0x008																															
复位值:0x0000 0000																															
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
DIVQ <31:0>																															

DIVQ	Bit 31-0	R	**商** 32位商

8. 4. 2. 4 CALC余数寄存器(CALC_DIVRR)

CALC 余数寄存器(CALC_DIVRR)																															
偏移地址:0x00C																															
复位值:0x0000 0000																															
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
DIVR <31:0>																															

DIVR	Bit 31-0	R	**余数** 32位余数

8. 4. 2. 5 CALC除法运算控制寄存器(CALC_DIVCON)

CALC 除法运算控制寄存器(CALC_DIVCON)																														
偏移地址:0x010																														
复位值:0x0000 0002																														
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
—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	FIXED_SPEED	TRM	SIGN

—	Bit 31-3	—	—
FIXED_SPEED	Bit 2	R/W	**固定运算时钟模式** 0:关闭固定运算时钟模式 1:开启固定运算时钟模式
TRM	Bit 1	R/W	**除法运算触发模式选择** 0:写入被除数后触发 1:写入除数后触发
SIGN	Bit 0	R/W	**除法运算符号选择** 0:无符号数 1:有符号数

8. 4. 2. 6 CALC被开方数寄存器(CALC_RDCND)

CALC 被开方数寄存器(CALC_RDCND)																														
偏移地址:0x014																														
复位值:0x0000 0000																														
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
RADICAND <31:0>																														

RADICAND	Bit 31-0	R/W	**被开方数** 32位无符号被开方数

8. 4. 2. 7 CALC平方根运算结果寄存器(CALC_SQRTRES)

CALC 平方根运算结果寄存器(CALC_SQRTRES)																												
偏移地址:0x018																												
复位值:0x0000 0000																												
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
																RESULT <15:0>															

—	Bit 31-16	—	—
RESULT	Bit 15-0	R	**平方根运算结果值** 16 位平方根运算结果

8. 4. 2. 8 CALC运算状态寄存器(CALC_STAT)

CALC 运算状态寄存器(CALC_STAT)																												
偏移地址:0x020																												
复位值:0x0000 0002																												
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
																														DZ	BUSY

—	Bit 31-2	—	—
DZ	Bit 1	R	**除数零警告** 0:除数不为 0 1:除数为 0 注意:每当写入除数寄存器时更新。
BUSY	Bit 0	R	**运算状态位** 0:完成 1:进行中

第9章 通用异步收发器 (UART)

9. 1 概述

通用异步收发器(UART)提供了一个灵活的方式，使 MCU 可以与外部设备通过工业标准 NRZ 的形式实现全双工异步串行数据通信。UART 可以使用小数波特率产生器，提供了超宽的波特率设置范围。

UART 支持异步通信模式和半双工单线通信，也支持 LIN(本地互联网络)、智能卡协议、IrDA(红外数据协会)SIR ENDEC 规范和 modem 流控操作(CTS/RTS)，同时还支持多机通信方式。

支持使用 DMA 实现多缓冲区设置，从而能够支持高速数据通信。

9. 2 特性

- 全双工异步通信
- 8-byte 接收和发送 FIFOs
- 兼容 16C550 标准
  - 可软件控制接收 FIFO 触发点
  - 可设置的通信波特率
- 支持自动波特率检测
- 十八个中断源
- 支持 DMA 使用
  - 利用 DMA 功能将收/发字节缓冲到保留的 SRAM 空间
- 内置小数波特率发生器，覆盖范围广
  - 在时钟频率为 48 MHz 下，可设置收发波特率高达 3 MBps，最低可达 732.4 Bps
  - 在时钟频率为 4 MHz 下，可设置收发波特率高达 250 KBps，最低可达 61 Bps
- 支持硬件自动流量控制功能(CTS、RTS)，可设置 RTS 控制触发点
  - Modem 硬件自动控制
  - RS485 发送使能控制
- 支持 CTS 唤醒功能
- 支持 IrDA SIR 模式
  - 支持 3/16 位宽功能
- 支持 RS-485
  - 支持 9-位模式
  - 多处理器通信
- 可设置的串行接口特性

- 可设置数据位个数，即 5、6、7、8、9 位，9 位用于 RS485 模式
- 校验位，包含奇、偶、无校验
- 停止位长度可设置: 1，2 位，在智能卡模式中支持 0.5，1.5 位
- 支持设置高位在前或低位在前

◆ 单线半双工通讯

◆ 可配置交换 TX/RX 引脚

◆ LIN 主机的断路信号发送功能和 LIN 从机的断路信号检测功能

- UART 设置为 LIN 模式时，有 13 位的断路信号产生器与断路信号检测功能

◆ 智能卡模式

- 支持 ISO/IEC7816-3 标准定义的 T=0 和 T=1 智能卡异步协议
- 智能卡使用的 1.5 停止位长度

◆ 支持 ModBus 通信

- CR/LF 字节检测
- 超时检测功能

◆ 噪声侦测

9. 3 结构图

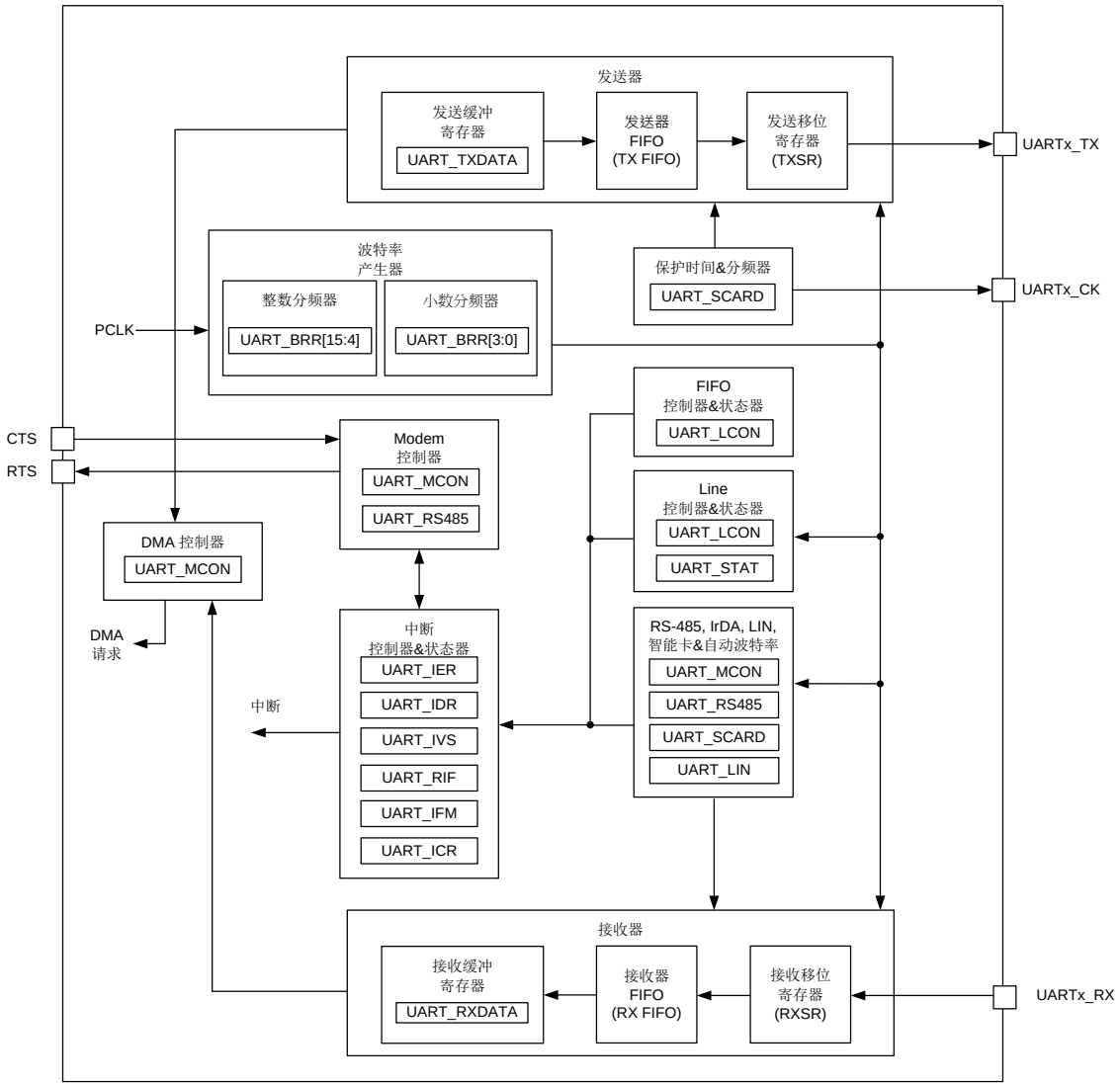


图 9-1 UART 框图

9.4 功能描述

任何 UART 双向通讯要求最少有两个引脚:接收数据输入(RX)和发送数据输出(TX)

- RX:接收数据输入，是串行数据的输入串口。使用过采样方式完成数据恢复，以区别输入数据和噪声。
- TX:数据发送输出。当发送器被关闭，引脚回到其 I/O 串口配置状态。当发送器开启，但不发送数据时，TX 脚为高电平输出。在单线半双工通信和智能卡模式中，这个串口既用于发送数据也用于接收数据。

通过这些引脚，串行数据用数据帧的形式发送和接收:

- 在发送和接收之前为空闲状态
- 起始位
- 资料可通过设置 UART_LCON 寄存器的 MSB 位
- 1, 2 个停止位表明帧的结束(0.5, 1.5 个停止位用于智能卡模式)
- 采用小数波特率产生器，整数 12 位与小数 4 位
- 一个状态寄存器(**UART_STAT**)
- 分开的接收和发送数据寄存器(**UART_RXDATA**, **UART_TXDATA**)
- 一个波特率寄存器(**UART_BRR**)-12 位整数和 4 位小数。
- 一个智能卡寄存器(**UART_SCARD**)用于智能卡模式。
- 一个接收时间寄存器(**UART_RTOR**)侦测输入信号时间并产生中断

下面的引脚用于智能卡模式:

CK: 时钟输出。智能卡模式中，CK 引脚会向智能卡提供时钟。

下列引脚用于支持硬件流量控制模式:

CTS:低发送，当高电平时作为发送阻止信号。

RTS:请求发送，表明 UART 已经准备好接收数据(低的时候)。

下列引脚用于 RS485 驱动开启控制:

DE:驱动开启将开启外部收发器的发送模式。

注:DE 和 RTS 共享同一个外部引脚。

9. 4. 1　具体功能配置

UART 模式与特征	UART1	UART2
Modem 的硬件控制	v	v
使用 DMA 实现连续通信	v	v
多机通信模式	v	v
智能卡模式	v	
单线半双工模式	v	
IrDA SIR 模块	v	
LIN 模式	v	
超时检测功能	v	v
Modbus 通信	v	v
自动波特率检测模式	v	v
RS485 的驱动开启信号	v	v
UART 数据宽度	5、6、7、8、9 Bits	

表 9-1　UART1~2 具体功能配置

注: v: 支持，支持 RS485 9bit 模式

9. 4. 2　功能描述

配置 **UART_LCON** 寄存器中的 DLS 位可选择 5、6、7、8 位字长。

默认设置中，发送和接收的起始位都是低电平。而停止位都是高电平。

这个逻辑可以在 **UART_LCON** 寄存器的 TXINV 与 RXINV 位设置为反向。

- 8 位字节宽度: DLS[1:0]=00
- 7 位字节宽度: DLS[1:0]=01
- 6 位字节宽度: DLS[1:0]=10
- 5 位字节宽度: DLS[1:0]=11

注:第 9 位使用于 RS485 多机通信模式

空闲符号被视为完全由'1’组成的完整的数据帧，后面跟着包含了数据的下一帧的开始位('1’的位数也包括了停止位的位数)。

断路符号被视为在一个帧周期内全部收到'0'(包括停止位期间，也是'0')。在断路帧结束时，发送器会再插入 2 个停止位。

发送和接收由一个共享的波特率产生器驱动，当发送器和接收器的开启位分别设置为 1 时，分别为其产生时钟。

下面是每个模块的详细说明。

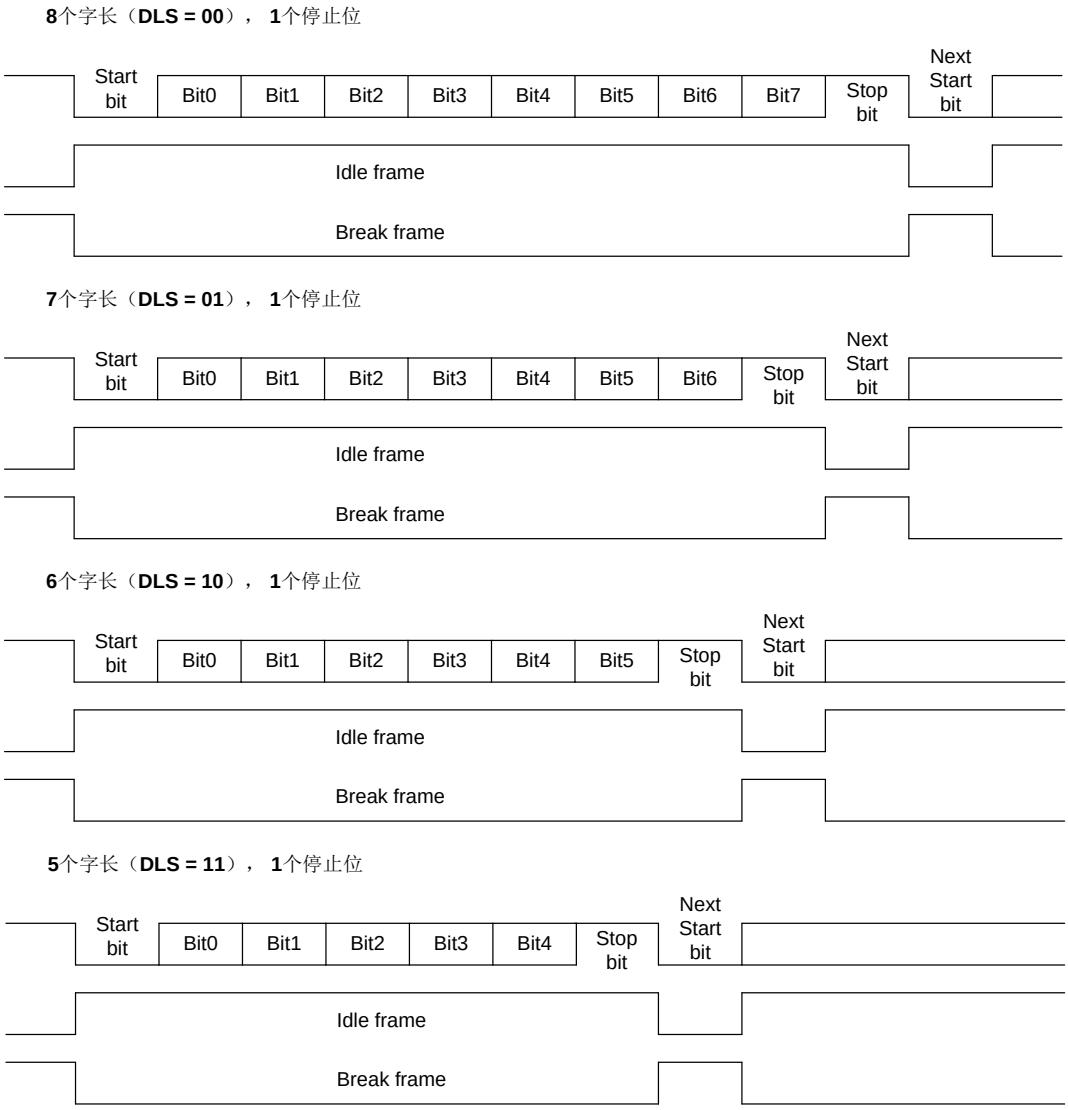


图 9-2　数据宽度设置

9. 4. 3　发送器

发送器根据 **UART_LCON** 寄存器的 DLS 位选择发送 5、6、7、8 位的数据，当写入 **UART_TXDATA** 寄存器，数据将存入移位寄存器中并将数据在 TX 引脚上输出。

在 UART 发送期间，在 TX 引脚上首先发送数据为最低有效位。在此模式中，**UART_TXDATA** 寄存器充当了一个内部总线和发送移位寄存器之间的缓冲器(TXSR)。

在发送每个字节之前有一个低电平的起始位，在字节发送结束后发送停止位，设置 **UART_LCON** 寄存器的 STOP 位选择停止位数。

UART 支持多种停止位的选择:0.5,1,1.5 和 2 个停止位(智能卡模式支持 0.5 与 1.5 停止位)。

- 0.5 个停止位: 在智能卡模式中发送和接收数据时使用。
- 1 个停止位:停止位的位数的默认值。

- 1.5 个停止位: 在智能卡模式中发送和接收数据时使用。
- 2 个停止位:可用于普通 UART 模式、以及调制解调器模式。

注:

1. 在写入 UART_TXDATA 寄存器数据前必须先等待 UART_STAT 寄存器中的 TFEMPTY 位为 1
2. 开启 TX 开关后，数据才能在 TX 脚上输出与停止位，随着每个字节发送，停止位的位数可以通过 UART_LCON 寄存器中的 STOP 位进行设置。

空闲帧包括了停止位。

断路帧可通过 UART_MCON 寄存器的 BKREQ 位产生 10 位低电平(当 DLS=00 时)，9 位低电平(当 DLS=01 时)，8 位低电平(当 DLS=10 时)或者 7 位低电平(当 DLS=11 时)，后跟 2 个停止位。

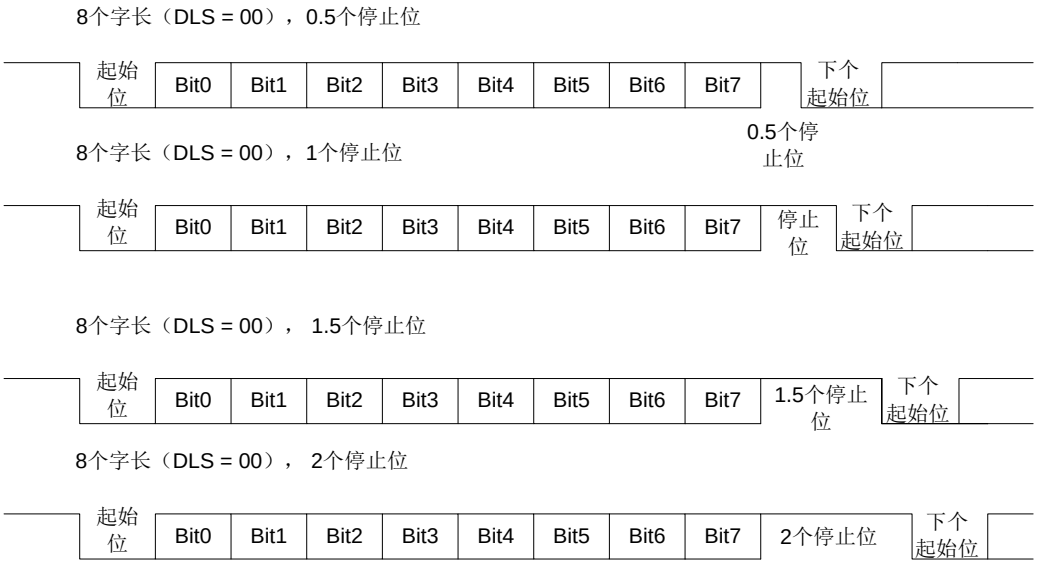


图 9-3 配置停止位

TX FIFO 门坎设置:

设置 **UART_FCON** 寄存器的 TXTH 位，可设置 FIFO 的门坎为 0、2、FIFO 深度/4、FIFO 深度/2，当 FIFO 内的数据 1 笔数小于门坎会使得 **UART_STAT** 寄存器的 TFTH 位为 1，告知用户需要再填入数据，以避免数据传送中断。

开启 **UART_IER** 寄存器的 TFTH 位为 1，UART 会判断 FIFO 内的笔数是否小于门坎，硬件设置 **UART_RIF** 寄存器的 TFTH 位为 1，产生中断。在产生中断的期间设置 **UART_ICR** 寄存器的 TFTH 位为 1,清除 **UART_RIF** 寄存器的 TFTH 位清除,若用户未写入新的数据至 FIFO 中，FIFO 内的数据 1 笔数仍然小于门坎则不会再次产生中断，需要重新开启 **UART_IER** 寄存器的 TFTH 位。

注:一开始 UART_RIF 寄存器的 TFTH 与 TFEMPTY 位为 0，当产生 FIFO 内的数据 1 笔数小于门坎事件与 FIFO 空事件时，使得 TFTH 与 TFEMPTY 位为 1。UART_STAT 寄存器的 TFTH 与 TFEMPTY 位初始为 1，反应 TX FIFO 状态。

配置步骤:

1. 设置 **UART_LCON** 寄存器中的 DLS 位配置字长。
2. 设置 **UART_LCON** 寄存器中的 STOP 位配置停止位的位数。
3. 设置 **UART_LCON** 寄存器中的 PE 与 PS 位配置校验控制开关与极性。
4. 设置 **UART_BRR** 寄存器选择希望的波特率。
5. 如果采用多缓冲器通信，设置 **UART_MCON** 寄存器中的 TXDMAEN 位为 1。按多缓冲器通信中的描述设置 DMA 寄存器。
6. 设置 **UART_LCON** 寄存器中的 TXEN 位，开启发送器。
7. 把要发送的数据写进 **UART_TXDATA** 寄存器(此动作将清除 **UART_STAT** 寄存器中的 TFEMPTY 位)。
8. 在 **UART_TXDATA** 寄存器中写入数据字时，要等待 **UART_STAT** 寄存器中的 TFEMPTY 位为 1。当需要关闭 UART，需要确认传输结束。**UART_STAT** 寄存器中的 TSBUSY 位为 0，避免破坏最后一次传输。

注:当 UART_LCON 寄存器的 TXEN 与 RXEN 位为 1 时，无法写入 UART_LCON 寄存器与 UART_BRR 寄存器

9. 4. 4 接收器

9. 4. 4. 1 消抖电路

在 **UART_RX** 引脚上配置了一个防抖动电路，设置 **UART_LCON** 寄存器中的 DBCEN 位开启功能，输入信号须维持至少 8 个高位或低位，才能使得信号反应至 UART 中，反之则被忽略，如下叙述。

在下图中，SYNC0 是指输入信号由系统时钟一次采样;SYNC1 是指 SYNC0 被系统时钟一次采样；SYNC2 表示 SYNC1 由系统时钟一次采样。SYNC0、SYNC1 和 SYNC2 可以表示成 x[n]×T_S，x[n+1]×T_S 和 x[n+2]×T_S。T_S 是系统时钟周期，n 是采样时间。如果关闭防抖动模块，时间就可以表示为 x[n+1]×T_S。

如果开启防抖动模块，SYNC2 信号将进入去抖电路，那么它将被采样与计数(用户设置样本频率和计数时间值)。时间可表示为[(SPT+1)×(FILTCNT+1)]×T_S。SPT 是采样频率值，FILTCNT 是计数次数。当 SYNC1 和 SYNC2 不相等时，计数值将被清零。如果计数值溢满 FILTCNT 寄存器，防抖动模块将输出信号。

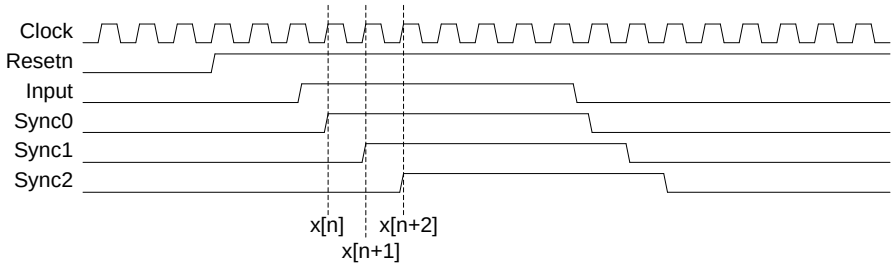


图 9-4　防抖动波形

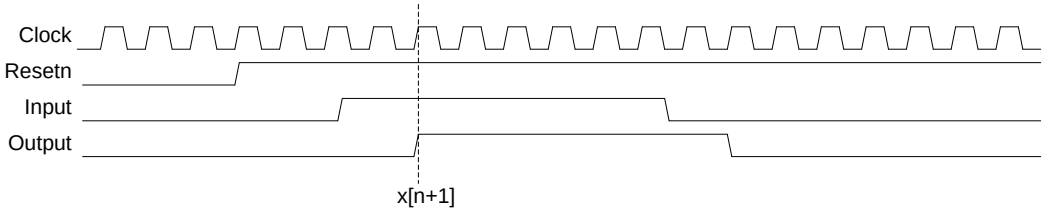


图 9-5　防抖动输出

9. 4. 4. 2　　起始位侦测

接收器根据 **UART_LCON** 寄存器中 DLS 位的状态接收 5、6、7、8 位的数据字。

在 UART 中，如果辨认出一个特殊的采样序列，那么就认为侦测到一个起始位。

该序列为:1 1 1 0 X 0 X 0 X 0 0 0 0 X X X X X X X

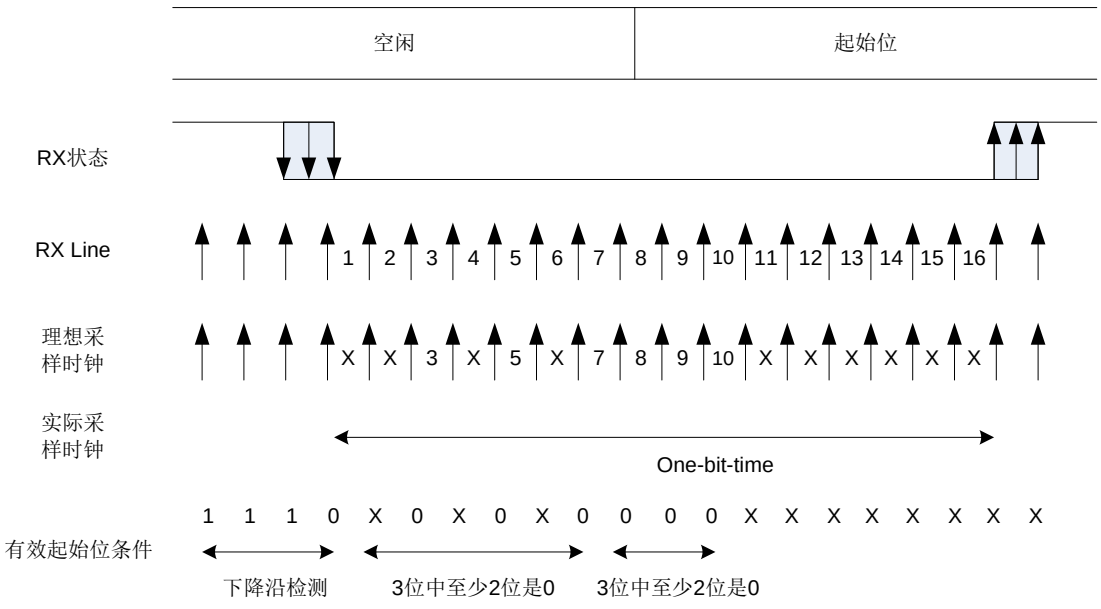


图 9-6　起始位侦测

注:

1. 如果该序列不完整，那么接收端将退出起始位侦测并回到空闲状态(不设置标志位)开始等待下降沿。
2. 如果 3 个采样点都为'0'(在第 3、5、7 位的第一次采样，和在第 8、9、10 的第二次采样都为'0')，则确认收到起始位。
3. 如果两次 3 个采样点上有 2 个是'0'(第 3、5、7 位的采样点和第 8、9、10 位的采样点)，那么起始位仍然是有效的。如果不能满足这个条件，则中止起始位的侦测过程，接收器会回到空闲状态。
4. 如果两次 3 个采样点上有 2 个是'1'(第 3、5、7 位的采样点和第 8、9、10 位的采样点)，那么起始位是无效的，将退出起始位侦测并回到空闲状态，并会设置噪声标志位。

RX FIFO 门坎设置:

设置 **UART_FCON** 寄存器的 RXTH 位，可设置 FIFO 的门坎为 1、FIFO 深度/4、FIFO 深度/2、FIFO 深度-2，当 FIFO 内的数据 1 笔数大于门坎会使得 **UART_STAT** 寄存器的 RFTH 位为 1，告知用户需要读取数据，以避免数据遗失。

开启 **UART_IER** 寄存器的 RFTH 位为 1，UART 会判断 FIFO 内的笔数是否大于门坎，硬件设置 **UART_RIF** 寄存器的 RFTH 位为 1，产生中断。在产生中断的期间设置 **UART_ICR** 寄存器的 RFTH 位为 1，清除 UART_RIF 寄存器的 RFTH 位，若用户未读取数据，FIFO 内的数据 1 笔数仍然大于门坎则不会再次产生中断，需要重新开启 **UART_IER** 寄存器的 RFTH 位。

注:一开始 UART_RIF 寄存器的 RFTH 位为 0，当产生 FIFO 内的数据 1 笔数大于门坎事件时，使得 RFTH 位为 1。UART_FCON 寄存器的 RFTH 位则是反应 RX FIFO 状态。

配置步骤:

1. 设置 **UART_LCON** 寄存器中的 DLS 位配置字长。
2. 设置 **UART_LCON** 寄存器中的 STOP 位配置停止位的位数。
3. 设置 **UART_LCON** 寄存器中的 PE 与 PS 位配置校验控制开关与极性。
4. 设置 **UART_BRR** 寄存器选择配置的波特率。
5. 如果采用多缓冲器通信，设置 **UART_MCON** 寄存器中的 RXDMAEN 位为 1。根据多缓冲器通信中的描述设置 DMA 寄存器。
6. 设置 **UART_LCON** 寄存器中的 RXEN 位，这将开启接收器，使它开始寻找起始位。

当一个字节被接收到时，**UART_STAT** 寄存器的 RFNEMPTY 位被设置为 1。它表明移位寄存器的内容被转移到 RX FIFO 中。换句话说，资料已经被接收并且可以被读出(包括与之有关的错误标志)。

这时如果 **UART_IER** 寄存器的 RFTH 位是 1，且 RX FIFO 门坎为 1，将会引起中断请求。

在接收期间如果检测到帧错误，噪音或溢出错误，错误标志将被设置为 1。同时 RXBERR 位也会和 RFNEMPTY 位一起被设置为 1。

在多缓冲器通信时，RFNEMPTY 在接收 1 个字节后被设置为 1，并由 DMA 对数据寄存器的读取而清除。

RFFULL 位必须在下一字节接收结束前被清零，以避免溢出错误。

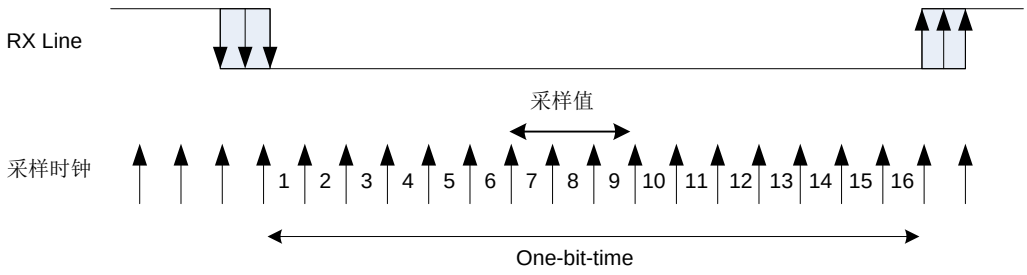


图 9-7　数值采样

帧错误

当以下情况发生时检测到帧错误:

由于没有同步上或大量噪音的原因，停止位没有在预期的时间上接收和检测出来。

当帧错误被检测到时:

1. FERR 位被硬件设置为 1
2. 此时读取的 **UART_RXDATA** 寄存器数据可能有错。
3. 寄存器 **UART_STAT** 中的 FERR 位显示当前从 FIFO 读取的 **UART_RXDATA** 寄存器是否为帧错误。
4. 寄存器 **UART_RIF** 中的 RXBERR 位则会在接收的过程中被设置为 1，若 **UART_IER** 中的 RXBERR 位为 1 则会产生中断。

奇偶位错误

当以下情况发生时检测到奇偶性错误:

由于没有同步上或大量噪音的原因，奇偶位没有在预期的时间上接收和检测出来。

当奇偶位错误被检测到时:

1. PERR 位被硬件设置为 1
2. 此时读取的 **UART_RXDATA** 寄存器数据可能有错。
3. 寄存器 **UART_STAT** 中的 PERR 位显示当前从 FIFO 读取的 **UART_RXDATA** 寄存器是否为奇偶位错误。
4. 寄存器 **UART_RIF** 中的 RXBERR 位则会在接收的过程中被设置为 1，若 **UART_IER** 中的 RXBERR 位为 1 则会产生中断。

断路错误

当以下情况发生时检测到断路错误:

由于没有同步上或大量噪音的原因，数据与停止位为 0 且没有在预期的时间上接收和检测出来。

当断路错误被检测到时:

1. BKERR 位被硬件设置为 1
2. 此时读取的 **UART_RXDATA** 寄存器数据可能有错。
3. 寄存器 **UART_STAT** 中的 BKERR 位显示当前从 FIFO 读取的 **UART_RXDATA** 寄存器是否为断路错误。
4. 这个错误并不会产生中断。

溢出错误

当以下情况发生时检测到溢出错误:

由于 RX 还没有被读取，而又接收到一个字节，则发生溢出错误。

当溢出错误被检测到时:

1. RFOERR 位被硬件设置为 1
2. **UART_RXDATA** 内容将不会丢失，读取 **UART_RXDATA** 寄存器仍能得到先前的数据。
3. 移位寄存器中以前的内容将被覆盖，随后接收的数据将丢失。
4. 寄存器 **UART_RIF** 中的 RFOERR 位则会在接收的过程中被设置为 1，若 **UART_IER** 中的 RFOERR 位为 1 则会产生中断。

采样值	采样位数值
000	0
001	0
010	0
011	1
100	0
101	1
110	1
111	1

表 9-2　采样资料的噪音检测数值

9. 4. 5　状态寄存器

在 UART 中配置了 15 种 UART 状态提供用户使用，叙述如下

- PERR (Parity error):
  当所接收的字节没有正确的校验位，产生奇偶位错误。该错误与 FIFO 顶部的字节有关，即读取 **UART_RXDATA** 寄存器的字节。
- FERR (Frame error):
  当接收到的字节停止位为 0 时，产生帧错误。该错误与 FIFO 顶部的字节有关，即读取 **UART_RXDATA** 寄存器的字节。
- BKERR (Break error):
  当接收到的字节与字节停止位为 0 时，产生断路错误。该错误与 FIFO 顶部的字节有关，

即读取 **UART_RXDATA** 寄存器的字节。

- CTSSTA (CTS status):
  清除发送，此位为显示 CTS 引脚上的输入状态。当 CTSSTA 位为 0，表示调制解调器和数据设备已准备好与 UART 进行数据交换。
- RSBUSY (RX shifter register busy):
  当此位为 1 表示接收器正在接收字节，为 0 则为完成接收。
- RFTH(RX FIFO threshold):
  当此位为 1 表示 RX FIFO 内资料大于门坎(设置 **UART_FCON** 寄存器的 RXTH 位，可设置门坎为 1、FIFO 深度/4、FIFO 深度/2、FIFO 深度-2)，告知用户需要读取 **UART_RXDATA** 寄存器。
- RFNEMPTY (RXnot empty):
  当此位为 1 表示 RX FIFO 内已接收 1 个以上的字节。
- RFFULL (RX FIFO full):
  当此位为 1 表示 RX FIFO 内已满，需要被读取。
- RFOERR (RX overrun):
  当此位为 1 表示 RX FIFO 内已满，且又再接收 1 个字节，此时 FIFO 内字节不会丢失，接收的字节则会被丢失。
- RFUERR (RX underrun):
  当此位为 1 表示 RX 内无任何字节，且又被读取。
- TSBUSY (TX shifter register busy):
  当此位为 1 表示发送器正在传送字节，为 0 则为传送完成，当写入第一个数据至 **UART_TXDATA** 寄存器就会使得 TSBUSY 位为 1。
- TFTH(TX FIFO threshold):
  当此位为 1 表示 TX FIFO 内资料小于门坎(设置 **UART_FCON** 寄存器的 TXTH 位，可设置门坎为 0、2、FIFO 深度/4、FIFO 深度/2)，告知用户需要写入 **UART_TXDATA** 寄存器。
- TFEMPTY (TX empty):
  当此位为 1 表示 TXFIFO 内无任何字节，为 0 则为准备发送 1 个以上的字节。
- TFFULL (TX FIFO full):
  当此位为 1 表示 TX FIFO 内已满，准备发送。
- TFOERR (TX overrun):
  当此位为 1 表示 TX FIFO 内已满，且又再写入 1 个字节，此时 TX 字节不会丢失，写入的字节则会被丢失。

9. 4. 6 波特率产生器

设置 **UART_BRR** 寄存器配置接收器和发送器的波特率。

- UARTDIV=UART.BRR

◆ TX/RX baud = f_{PCLK} / UARTDIV

注:

1. 当 UART_LCON 寄存中的 RXEN 位与 TXEN 位为 1 时，UART_BRR 寄存器无法被写入。
2. 当 BRR[15:4]=0 时，无法运行

从 **UART_BRR** 寄存器值中推断出 UART 波特率

◆ 在 4 MHz 下，为了得到 115200 波特
- UARTDIV = 4000000/115200 = 34.7
- BRR[15:0] = UARTDIV = 35d = 23h (四舍五入)

◆ 在 8 MHz 下，为了得到 9600 波特
- UARTDIV = 8000000/9600 = 833.33
- BRR[15:0] = UARTDIV = 833d = 341h (四舍五入)

◆ 在 16 MHz 下，为了得到 1200 波特
- UARTDIV = 16000000/1200 = 13333.33
- BRR[15:0] = UARTDIV = 13333.33d = 3415h (四舍五入)

◆ 在 24 MHz 下，为了得到 460800 波特
- UARTDIV = 24000000/460800 = 52.08
- BRR[15:0] = UARTDIV = 52d = 34h (四舍五入)

◆ 在 48 MHz 下，为了得到 115200 波特
- UARTDIV = 48000000/115200 = 416.66
- BRR[15:0] = UARTDIV = 417d = 1A1h (四舍五入)

◆ 在 48 MHz 下，为了得到 921600 波特
- UARTDIV = 48000000/921600 = 52.08
- BRR[15:0] = UARTDIV = 52d = 34h (四舍五入)

预期波特率	实际波特率	BRR[15:0]	%误差
734Bps	733KBps	0xFFCC	0
1.2KBps	1.2KBps	0x9C40	0
2.4KBps	2.4KBps	0x4E20	0
4.8KBps	4.8KBps	0x2710	0
9.6KBps	9.6KBps	0x1388	0
19.2KBps	19.2KBps	0x9C4	0
38.4KBps	38.4KBps	0x4E2	0
57.6KBps	57.62KBps	0x341	0.03
115.2KBps	115.11KBp	0x1A1	0.08
230.4KBps	230.77KBp	0xD0	0.16
460.8KBps	461.54KBp	0x68	0.16
921.6KBps	923.07KBp	0x34	0.16

预期波特率	实际波特率	BRR[15:0]	%误差
1.5MBps	1.5MBps	0x20	0
2MBps	2MBps	0x18	0
3MBps	3MBps	0x10	0

表 9-3　时钟为 48MHz 下，设置波特率时的误差计算

9. 4. 7　自动波特率侦测

UART 可以根据接收到的一个字节来检测和自动设置 **UART_BRR** 寄存器的值。自动波特率检测用于以下两种情况下:

- 通信速度不可知的情况下
- 使用低精度时钟源时，需要在不测量时钟偏差的条件下调整波特率的时候

时钟源的频率必须和预期的波特率保持相对的比例(过采样率为 16，并且波特率处于 f_{PCLK}/65535 和 f_{PCLK}/16 之间)。

在开启自动波特率检测时，必须先确认字节的内容。根据不同的字节内容选择模式，能够通过 **UART_MCON** 寄存器中的 ABRMOD 位进行选择。具体是:

- 模式 0(0x00):波特率在 UART 的 RX 引脚的两个连续的下降沿时间测量(起始位的下降沿和最低数据位的下降沿(LSB))
- 模式 1(0x01):波特率在 UART 的 RX 引脚下降沿和后续上升沿时间(起始位的长度)测量。
- 模式 2(0x10):波特率在 UART 的 RX 引脚下降沿和后续上升沿时间(起始位的长度加上 bit[0]的长度)测量(e.g. 0xFE)。

设置 **UART_MCON** 寄存器中的 ABREN 位为 1，开启自动波特率检测功能。UART 在 RX 上等待第一个字节过来。当自动波特率操作结束后，**UART_RIF** 寄存器中的 ABEND 位会被硬件自动设设置为 1，若 **UART_IER** 寄存器中的 ABEND 位为 1 则会产生中断。

如果线路噪声严重，不能保证得到的波特率是准确的。这时 BRR 值可能是错的或者 ABEND 位会被设置为 1。在通信速度超出自动波特率检测范围(位长度不在 0x10 到 0xFFFF 个时钟周期之间)时也会发生这种情况。

在此模式中配置了两个中断，分别为侦测超时与侦测结束。

在软件并未清除 **UART_MCON** 寄存器中的 ABREN 的情况下，UART 计数器会持续计数直到 0xFFFF 后，硬件会自动将 ABREN 清除，并设置 **UART_RIF** 寄存器中的 ABTO 位设置为 1，如果开启中断，则会产生 ABTO 中断。

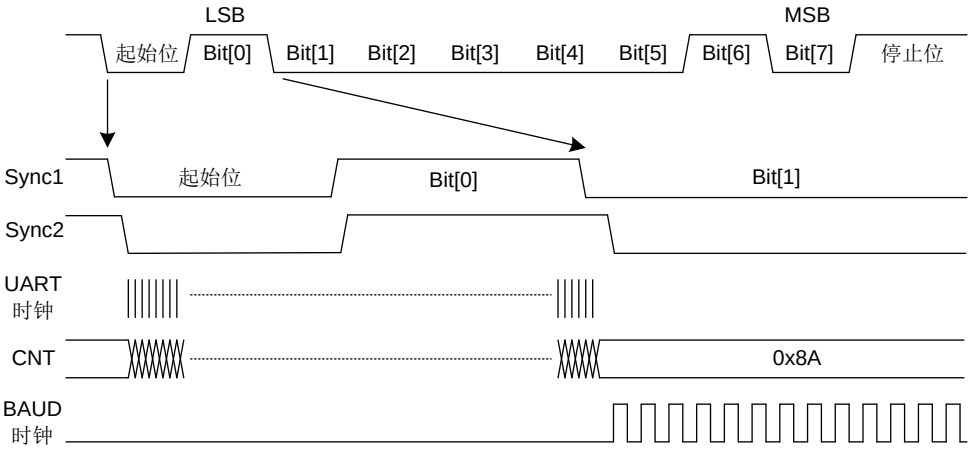


图 9-8 自动波特率侦测模式 0

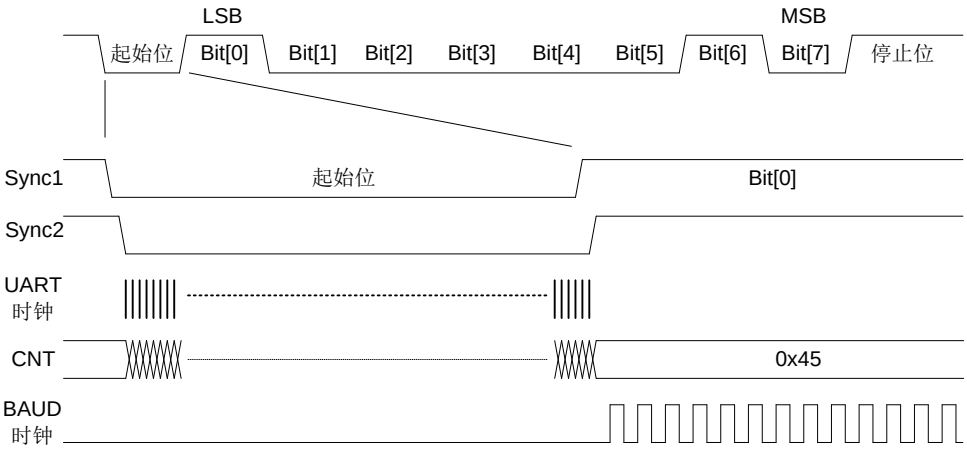


图 9-9 自动波特率侦测模式 1

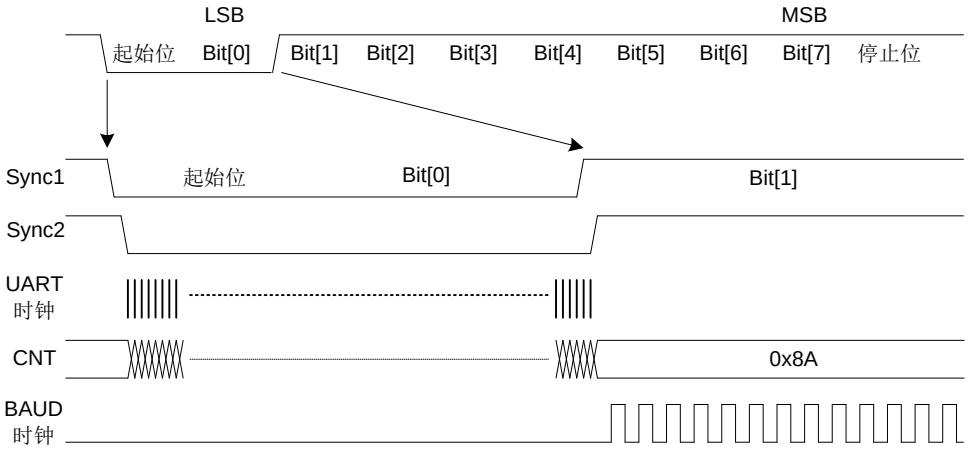


图 9-10　自动波特率侦测模式 2

9. 4. 8 自动流量控制

如果开启自动流量控制，接收器和发送器会通过 **UARTx_RTS** 和 **UARTx_CTS** 引脚去控制 UART 的接收(RX)和发送(TX)。

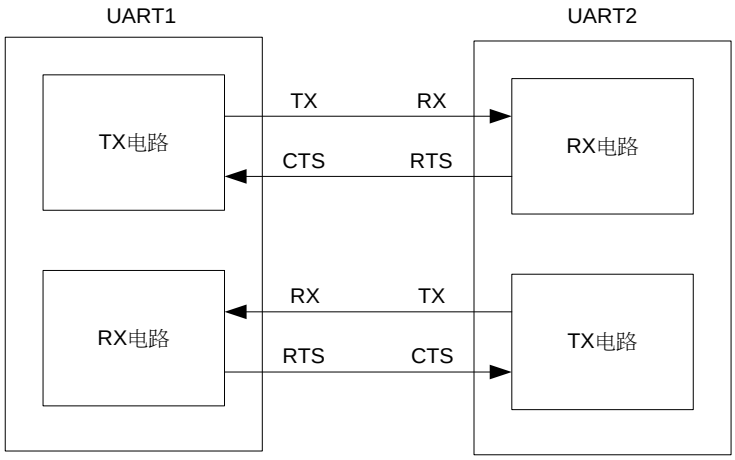


图 9-11 自动流量控制框图

9. 4. 8. 1 RTS流量控制

当开启自动 RTS 控制时(AFCEN=1)，当 UART 接收器准备好接收新数据，便会将 RTS 转为有效状态(输出低电平)。当接收器已满时，会将 RTS 转为无效状态(输出高电平)，表示发送过程会在当前帧结束后停止。

可选择接收 FIFO 门坎的值是: 1、FIFO 深度/4、FIFO 深度/2、FIFO 深度-2 个字节。一个额外的字节有可能会在 **UART_RTS** 成为无效后传送到 UART(由于 UART 进入发送器的数据尚未发送完成)，门坎设置为 FIFO 深度-2 个字节能在最大限度安全区下使用 FIFO。

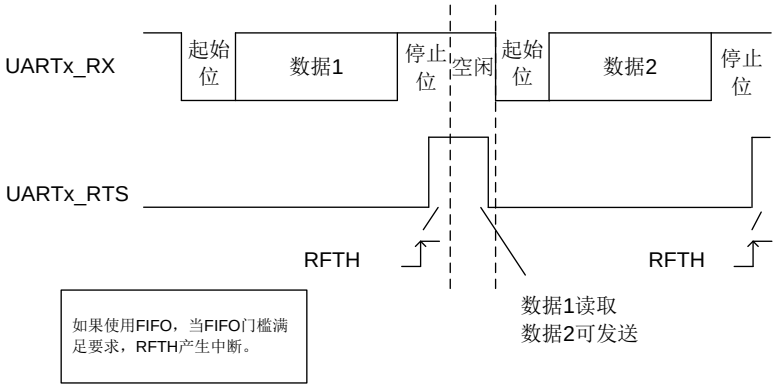


图 9-12 自动 RTSn 控制

9. 4. 8. 2 CTS流量控制

当开启自动 CTS 控制时(AFCEN=1)，发送器会在发送下一帧前检查 CTS。如果 CTS 为有效状态(输入低电平)，则会发送下一数据(假设数据已准备好发送，即 TFEMPTY=0);否则不会进行发送。如果在发送过程中 CTS 转为无效状态(输入高电平)，则当前发送完成之后停止发送器。只要 CTS 发生变化，**UART_STAT** 寄存器的 CTSSTA 位便会由硬件自动设置为 1 或 0。这表示接收器是否已准备好进行通信。如果 **UART_IER** 寄存器中的 DCTS 位为 1 则会产生中断。

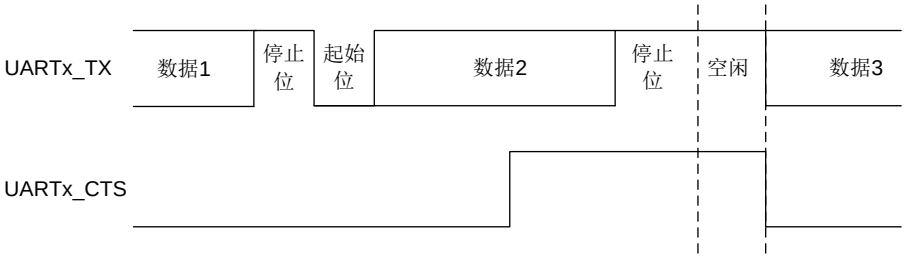


图 9-13 自动 CTSn 控制

9. 4. 8. 3 RS485 驱动使能(DE)

当开启 RS485 驱动开启功能(**UART_RS485** 寄存器的 AADEN=1 或 AADNEN=1)。它允许用户通过 DE(驱动开启)信号来开启外部收发器的控制端。延迟时间是在发送最后一个字节的停止位和释放 DE 信号之间插入延迟，这个时间通过 **UART_RS485** 寄存器中的 DLY 位设置。而 DE 信号的极性则可以通过 **UART_RS485** 寄存器中的 AADINV 位中设置并进行选择极性。

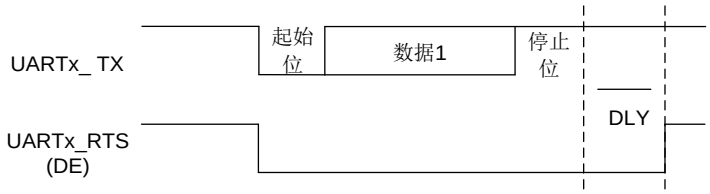


图 9-14 驱动开启当 AADINV=0

9. 4. 9 Modbus通信

UART 提供对 Modbus/RTU 和 Modbus/ASCII 协议实现的基本支持。Modbus/RTU 是一个半双工的块式传输协议。协议的控制部分(地址检测，块完整性控制和命令解析)必须由软件来完成。UART 提供对块尾的基本支持检测，无需软件的经常性介入。

- Modbus/RTU

  这个模式中，一个块结束为一个超过 2 个字节长度的空闲状态。通过设置一个超时长度来实现功能。

  超时功能和相应的中断必须通过**UART_RTOR** 寄存器中的 RTOEN 位和**UART_IER** 寄存器中的 RXTO 位来开启。**UART_RTOR** 寄存器中的 RTO 位要填入一个与超时长度相当

的数字(例如 2 个字节长度为 22 个位长)。当接收线路保持空闲阶段达到这个长度时，在最后一个停止位被收到之后，会产生一个中断，表示当前的块接收已经完毕。

- Modbus/ASCII

  在这个模式，一个块结束通过特定(CR/LF)字节侦测。通过字节匹配功能实现这个机制。将 LF 的 ASCII 码写到 ADD[7:0]区域，设置 **UART_IER** 寄存器中的 ADDRM 位为 1 开启功能，那么软件就会在收到 LF 字节后或者能够在 DMA 缓冲区中找到 CR/LF 字节时得到通知。

9. 4. 10 校验控制

设置 **UART_LCON** 寄存器中的 PE 位为 1 开启校验控制(发送时生成一个校验位，接收时进行校验位检查)。根据 DLS 位配置的帧长度，下表中列出可能的 UART 帧格式。

DLS[1:0]	PE	UART 帧
00	0	起始位+8 位数据+停止位
00	1	起始位+8 位数据+校验位+停止位
01	0	起始位+7 位数据+停止位
01	1	起始位+7 位数据+校验位+停止位
10	0	起始位+6 位数据+停止位
10	1	起始位+6 位数据+校验位+停止位
11	0	起始位+5 位数据+停止位
11	1	起始位+5 位数据+校验位+停止位

表 9-4 帧格式

- 奇校验

  校验位为一帧中的 8、7、6 或 5 个 LSB 位以及校验位'1’的个数为奇数。

  例如:数据=00110101，有 4 个'1'，如果选择奇校验(在 **UART_LCON** 寄存器中的 PS 位为 0)，校验位将是'1'。

- 偶校验

  校验位为一帧中的 8、7、6 或 5 个 LSB 位以及校验位'1’的个数为偶数。

  例如:数据=00110101，有 4 个'1'，如果选择偶校验(在 **UART_LCON** 寄存器中的 PS 位为 1)，校验位将是'0'。

- 接收时的校验检查

  如果校验检查失败，**UART_STAT** 寄存器中的 PERR 位会被设置为 1，如果 **UART_IER** 寄存器中的 RXBERR 位为 1 则会产生中断。

- 发送时的校验生成

  设置 **UART_LCON** 寄存器的 PE 位为 1 时，写进数据寄存器的数据的 MSB 位由校验位替换后发送出去(选择奇校验奇数个'1'(PS=0)或选择偶校验偶数个'1'(PS=1)

9. 4. 11 多处理器通信

设置 DLS 位为 8 位字长(第 9bit 为判断地址或数据)
设置 **UART_RS485** 寄存器的 AADEN 位为 1 以进入模式。
设置 **UART_RS485** 寄存器的 ADDR 位配置匹配地址

可以将多个 UART 连接成一个网络来实现多机通信。例如某个 UART 设备可以是主 UART，它的 TX 输出和其他 UART 从设备的 RX 输入相连接；UART 从设备各自的 TX 输出在逻辑上通过与运算连在一起，并且和主设备的 RX 输入相连接。

在多处理器配置中，通过特定地址的将接收器开启，来接收随后的数据，这样就可以减少由未被寻址接收器的参与带来的多余的 UART 服务开销。

未被寻址的设备可开启静默功能进入静默模式。设置 **UART_RS485** 寄存器的 AADEN 位为 1 开启静默模式功能。

在这个模式中，如果 MSB 是 1，该字节被认为是地址，否则被认为是数据。在一个地址字节中，目标接收器的地址被放在 **UART_RS485** 寄存器的 ADDR 位。

如果接收到的字节与它的设置地址不匹配时，UART 进入静默模式。当 UART 进到静默模式后，接收字节时既不会动 **UART_RIF** 寄存器的 RFNEMPTY 位也不会产生中断或发出 DMA 请求。

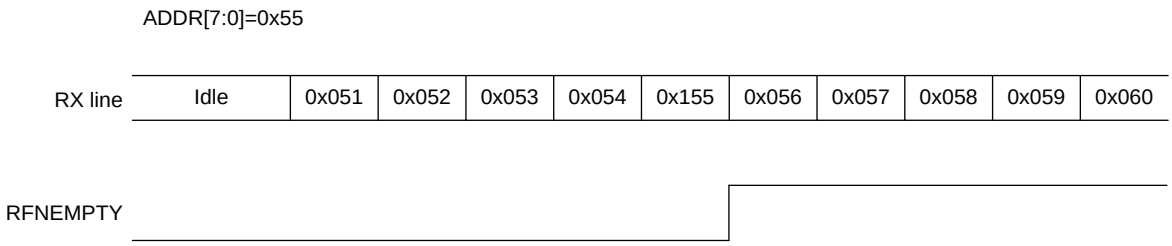


图 9-15 使用地址标示侦测模式

9. 4. 12 LIN模式

- LIN 发送

  和普通的 UART 发送相同，但包含下列区别:
  设置 DLS 位为 8 位字长
  设置 **UART_LIN** 寄存器的 LINEN 位为 1 以进入 LIN 模式。设置 **UART_LIN** 寄存器的 LINBKRQ 位为 1 将发送 13 位'0’作为断路符号。然后发两位'1'，以检测下一个开始位。

- LIN 接收

  当 LIN 模式开启时(**UART_LIN** 寄存器的 LINEN 位为 1)，检测断路符号电路自动被开启。该检测独立于 UART 接收器。不管是在空闲状态还是在发送数据期间，断路符号只要一出现就能被检测到。
  一旦接收器被开启(LCON 寄存器的 RXEN 位为 1)，电路就开始检测 RX 上的起始信号。检测起始位的方法和检测断路符号或数据是一样的。当起始位被检测到后，电路检测接下来的每个位，在每个位的第 8，9，10 个过采样时钟点上进行采样，就像采样数据一样。

如果 10 个(当 **UART_LIN** 寄存器中的 LINBDL=0)或 11 个(当 **UART_MCON** 寄存器中的 LINBDL=1)连续位都是'0'，并且又跟着一个分隔符，**UART_RIF** 寄存器的 LINBK 位就会被设置为 1。如果 **UART_IER** 寄存器的 LINBK 位为 1 则会产生中断。在确认断路符号前，要检查分隔符，因为它表示 RX 已经回到高电平。如果在第 10 或 11 个采样点之前采样到了'1'，检测电路取消当前检测并重新寻找起始位。如果 LIN 模式被关闭，接收器则继续正常 UART 工作，不再考虑检测断路符号。如果 LIN 模式被开启(LINEN=1)，只要一发生帧错误(例如: 停止位检测到'0'，这种情况出现在断路符号被接收到的时候)，接收器就会立即停止，直到电路检测断路符号后接收到一个'1'(这种情况发生于断路符号没有完整的发出来)，或一个分隔符(这种情况发生于已经检测到一个完整的断路符号)。

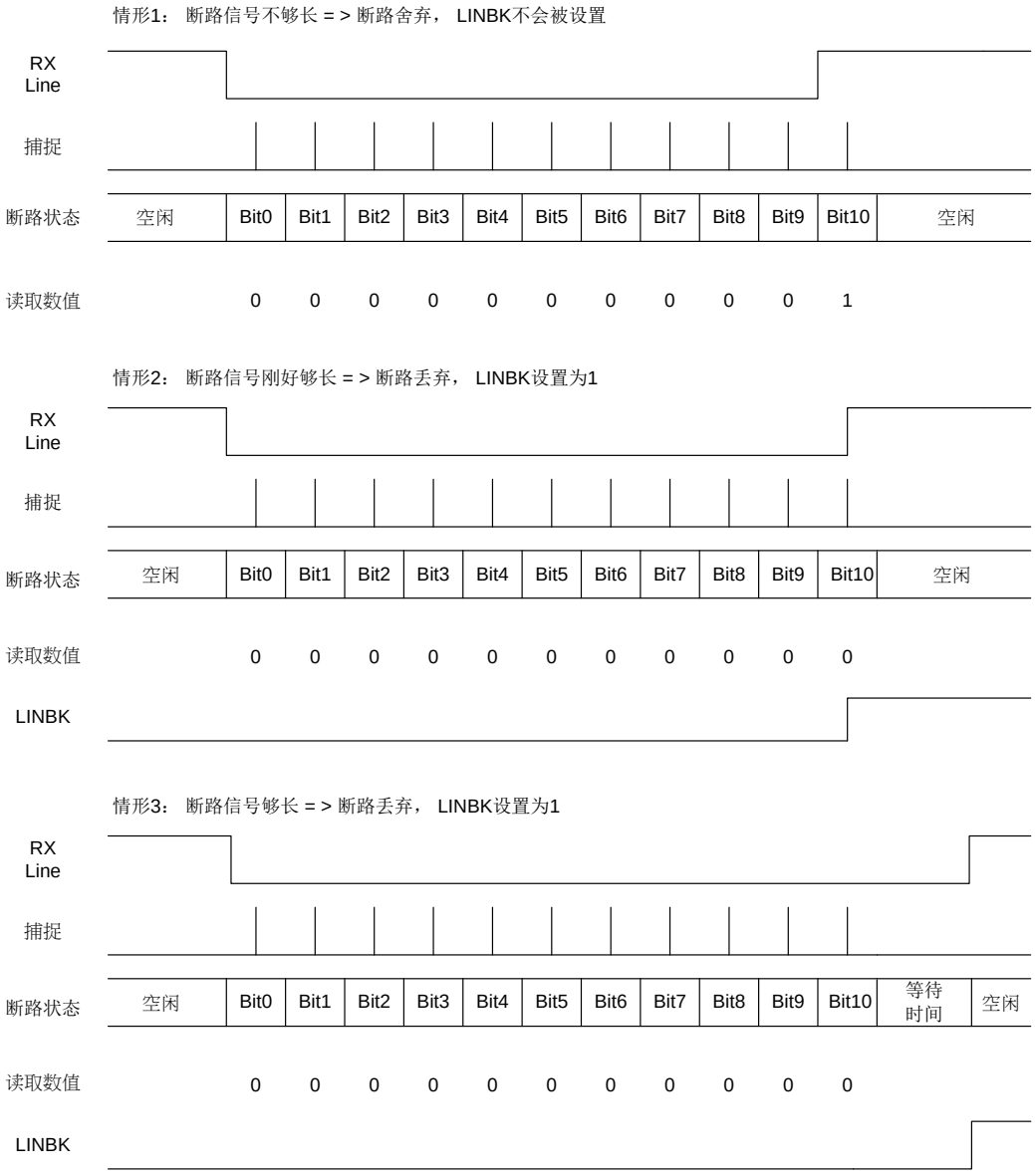


图 9-16 LIN 模式侦测断路信号(11 位断路长度 - LBDL 位为 1)

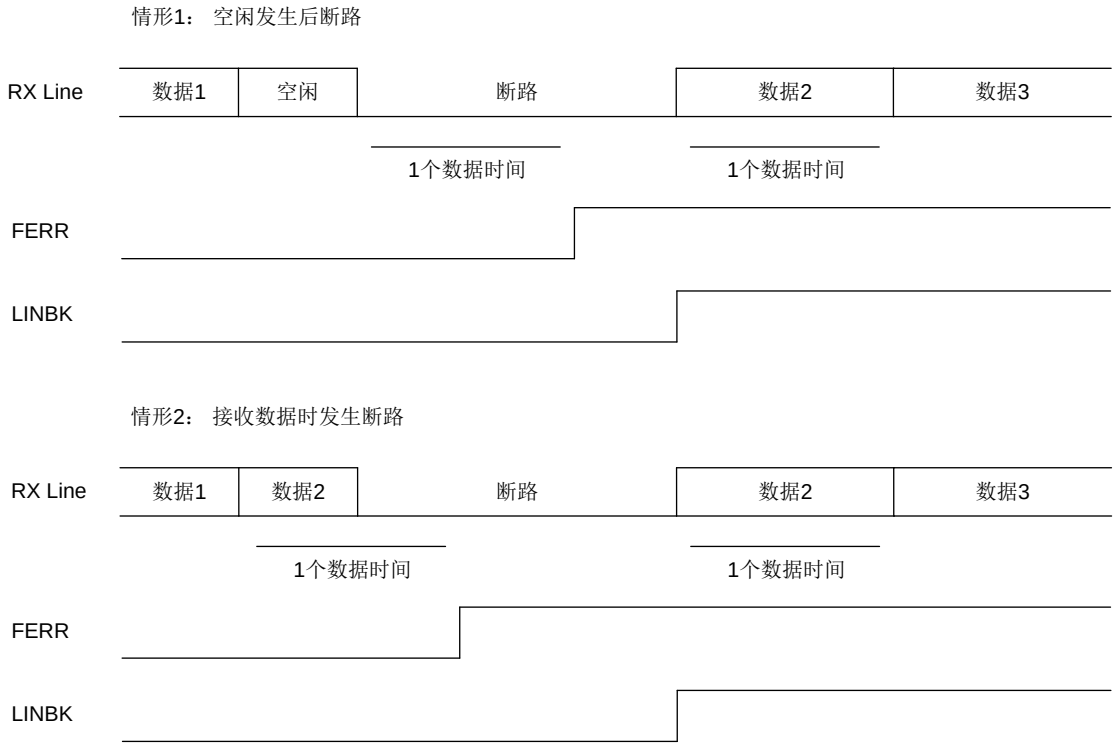


图 9-17　LIN 模式侦测断路信号与帧错误信号

9. 4. 13　单线半双工通讯

UART 可以配置成遵循单线半双工协议。在单线半双工模式中，TX 和 RX 引脚在芯片内部是连在一起的。使用控制位(**UART_MCON** 寄存器中的 HDEN 位)选择半双工或全双工通信。

- 当 HDEN 为 1 时:
  - TX 和 RX 引脚在芯片内部是连在一起的
  - RX 不再被使用
  - 当没有数据传输时，TX 引脚为释放状态。因此，在空闲状态或接收状态时为一个标准 I/O 串口。这就表示该 I/O 串口在不被 UART 驱动时，必须配置成悬空输入(或开漏的高输出)。

除此以外，通信与正常 UART 模式类似。由软件来管理在线的冲突(例如通过使用一个中央仲裁器)。特别的是，发送从不会被硬件所阻碍。当 LCON 寄存器的 TXEN 位为 1，只要数据一写到数据寄存器中，发送就会开始。

9. 4. 14 智能卡模式

设置 8 位数据位加校验位: 即 LCON 寄存器中 DLS=00，PE=1
设置 0.5、1.5 个停止位: 即 LCON 寄存器的 STOP=0 或 1
设置 UART_SCARD 寄存器的 SCEN 位为 1 以进入智能卡模式

在 T=0(字节)模式中，当校验错误时在字节发送完毕后将接收器拉为低电平后产生保护时间。所示为在数据在线有校验错误和没有校验错误时的情形。

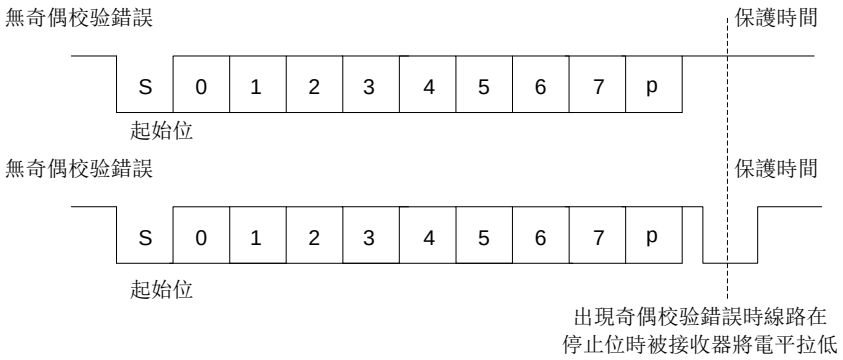


图 9-18 ISO 7816-3 异步协议

当连接到智能卡时，UART 的 TX 引脚和智慧卡通过同一根双向数据线进行通信。所以 TX 引脚必须配置成开漏状态。

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。在智能卡模式中，此发送过程还会产生一个 1/2 波特时钟周期的延迟。
- 如果在接收一个使用 0.5 或 1.5 个停止位设置的帧期间检测到奇偶位错误，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 UART 的资料尚未正确接收。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。软件根据协议重新发送资料。如果设置 NACK 位为 1，则接收器会发送’NACK’信号；否则不会发送 NACK 信号(T=1 模式中使用)。
- 通过对保护时间寄存器进行设置，可以延迟 **UART_STAT** 寄存器的 TBC 的设置时间。正常工作时，当发送移位寄存器为空时，会对 TBC 位被设置为 1。在智能卡模式中，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，TBC 位被强制设置为 0。当保护时间计数器达到设置值时，TBC 位被设置为 1。
- 对 TBC 位的设置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，如果检测到奇偶位错误并发送了 NACK 信号，则接收端不会将 NACK 作为起

始位进行检测。

注: 在智能卡模式中带有帧错误的 0x00 数据将被视为数据，而非中断。

下图详细介绍了 UART 如何对 NACK 信号采样。在本例中，UART 正在发送数据并配置了 1.5 个停止位。UART 的接收部分已开启，以检查数据的完整性和 NACK 信号。

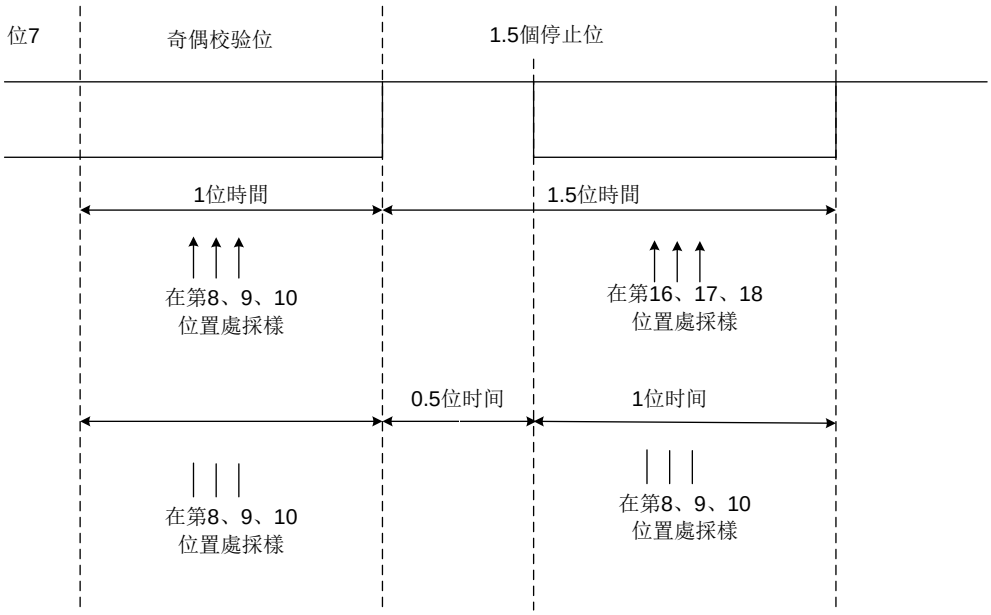


图 9-19　1.5 位停止位时检测校验错误

UART 可以通过 CK 引脚向智能卡提供时钟。智能卡模式中，CK 时钟和通信没有关系，只是通过一个 5 位的预分频器从内部外设时钟源得到时钟信号。这个分频系数在 **UART_SCARD** 寄存器的 PSC 位配置。CK 频率可以设置在 $f_{CK}/2$ 到 $f_{CK}/64$ 之间, f_{CK} 指外设输入时钟。

9. 4. 15　IrDA SIR模块

设置 **UART_MCON** 寄存器的 IREN 为 1 以进入 IrDA 模式。
IrDA SIR 物理层规定使用反相归零(RZI)调制方案，它以红外光脉冲表示逻辑 0。

SIR 发送编码器用于调制 UART 发出的非归零(NRZ)编码。输出脉冲会发送到外部输出驱动器和红外线 LED。UART 支持的 SIR 编码比特率最高为 115.2Kbps。在正常模式中，所发送的脉冲宽度规定为一个位周期的 3/16。

SIR 接收译码器用于解调由红外探测器发出的归零编码，并将接收到的 NRZ 串行编码输出到 UART。在空闲状态下，译码器输入为高电平(标记状态)。发送编码器输出的极性与译码器输入相反。当译码器输入为低电平时，会检测到起始位。

IrDA 是一个半双工通信协议。如果发送器发送时，例如 UART 正在向 IrDA 编码器发送数据，则 IrDA 译码器会忽略 IrDA 接收在线的所有数据；如果接收器接收时，例如 UART 正在接收来

自 RX 引脚上的数据，则 IrDA 不会对 UART 发送到 IrDA 的 TX 数据进行编码。在接收数据时，应避免同时进行发送，因为这样做可能会破坏要发送的数据。

- SIR 发送逻辑把 0 作为高脉冲发送，把 1 作为低电平发送。脉冲的宽度规定为所选位周期的 3/16
- SIR 译码器用于将兼容 IrDA 的接收信号转换为 UART 的编码。
- SIR 接收逻辑把高电平状态作为 1，把低脉冲作为 0。
- 发送编码器输出的极性与译码器输入相反。SIR 输出在空闲时处于低电平状态。
- 在 IrDA 模式中，LCON 寄存器中的 STOP 位必须配置成 1 个停止位。

接收器的建立时间应由软件进行管理。IrDA 物理层规范规定发送和接收之间至少要经过 10ms 的时间间隔(IrDA 是一个半双工协议)。

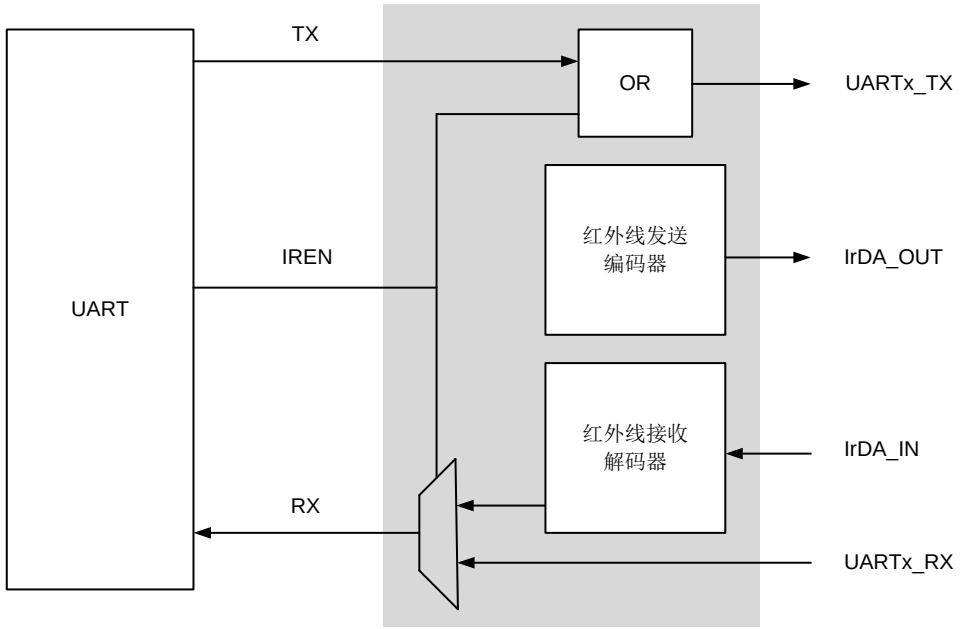


图 9-20　红外收发框图

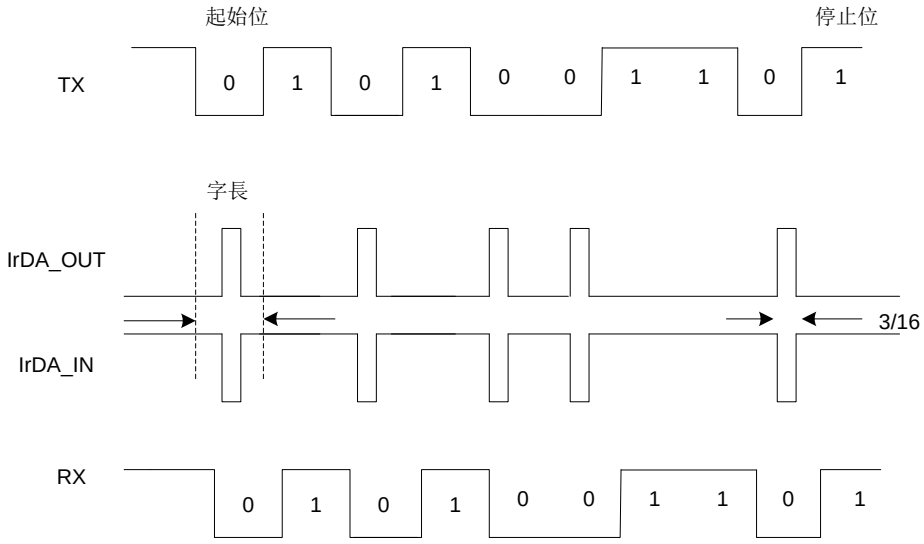


图 9-21　IrDA 数据调制(3/16) – 正常模式

9. 4. 16　使用DMA连续通信

设置 **UART_MCON** 的 RXDMAEN 位为 1 开启 RX DMA 或 TXDMAEN 位为 1 开启 TX DMA。

UART 可以利用 DMA 连续通信。RX 缓冲器和 TX 缓冲器分别产生 DMA 请求。

利用 DMA 发送

使用 DMA 进行发送，可以通过设置 **UART_MCON** 寄存器中的 TXDMAEN 位开启。可使用 DMA 外设(请参见相应的 DMA 控制器部分)将数据从配置的 SRAM 地址加载到 **UART_TXDATA** 寄存器中。要映射一个 DMA 通道以进行 UART 发送，请按以下步骤操作

- 在 DMA 控制寄存器上设置 **UART_TXDATA** 寄存器地址配置成 DMA 传输的目的地址。在每个 TFTH 事件后，数据将被传送到这个地址。
- 在 DMA 控制寄存器上将 SRAM 地址配置成 DMA 传输的来源地址。在每个 TFTH 事件后，将从此内存区读出数据并传送到 **UART_TXDATA** 寄存器中。
- 在 DMA 控制寄存器中配置要传输的字节数。
- 在 DMA 寄存器上配置通道优先级。
- 根据应用程序的要求，配置在传输完成一半还是全部完成时产生 DMA 中断。
- 设置 **UART_ICR** 寄存器的 TFTH 位为 1 以清除 **UART_RIF** 寄存器的 TFTH 位。
- 在 DMA 寄存器上开启该通道。

注: 若使用 FIFO，可一次写入多个传输字节数，并设置 UART_FCON 寄存器中的 TXTH 位，可设置 FIFO 深度，新的数据将被传送到 UART_TXDATA 寄存器中。

在发送模式中，当 DMA 传输完所有要发送的数据时，DMA 控制器设置 **UART_IFM** 寄存器的 TFEMPTY 位；检查 **UART_RIF** 寄存器的 TFEMPTY 位可以确认 UART 通信是否结束。这样可以在关闭 UART 或进入停机模式之前避免破坏最后一次传输的数据。软件必须等待 TSBUSY

被设置为 0。TSBUSY 位在全部数据发送期间会是 1，并且在最后一帧数据发送完成之后会由硬件设置为 0。

利用 DMA 接收

使用 DMA 进行接收，可以通过设置 **UART_MCON** 寄存器中的 RXDMAEN 位为 1 开启。当接收数据字节时，可使用 DMA 外设(请参见相应的 DMA 控制器部分)将数据从 **UART_RXDATA** 寄存器读取出来加载到配置的 SRAM 地址。要映射一个 DMA 通道以进行 UART 接收，请按以下步骤操作

- 在 DMA 控制寄存器上设置 **UART_RXDATA** 寄存器地址配置成 DMA 传输的来源地址。在每个 RFTH 事件后，读取数据将从这个地址加载。
- 在 DMA 控制寄存器上将 SRAM 地址配置成 DMA 传输的目标地址。在每个 RFTH 事件后，读取数据将从 **UART_RXDATA** 寄存器加载这个目标地址。
- 在 DMA 控制寄存器中配置要传输的字节数。
- 在 DMA 寄存器上配置通道优先级。
- 根据应用程序的要求，配置在传输完成一半还是全部完成时产生 DMA 中断。
- 设置 **UART_ICR** 寄存器的 RFTH 位为 1 以清除 **UART_RIF** 寄存器的 RFTH 位。
- 在 DMA 寄存器上开启该通道。

注: 若使用 FIFO，可一次接收多笔传输字节数，并设置 UART_FCON 寄存器中的 RXTH，可设置 FIFO 深度，将新的数据将被传送到 UART_RXDATA 寄存器中。当传输完成时，若开启了 DMA 中断，DMA 控制器会产生中断。

多缓冲器通信中的错误标志和中断产生

在多缓冲器通信的情况下，通信期间如果发生任何错误，会在当前字节传输后将错误标志设置为 1。如果中断开启位被设置为 1 则会产生中断。在单个字节接收的情况下，和 **UART_IFM** 寄存器中的 RXBERR 位和 RFOERR 位一起被设置为 1 表示帧错误和溢出错误，有分别的错误标志中断开启位；如果被设置为 1 了，会在当前字节传输结束后，产生中断。

9. 4. 17 中断配置

- **UART_IER 中断开启寄存器**

  此位设置 1 时，表示开启中断功能，并且同时反映在 **UART_IVS** 寄存器。此寄存器只能写入，并且只允许写入 1，无法写 0 取消开启中断设置。
- **UART_IDR 中断关闭寄存器**

  此位设置 1 时，表示关闭中断功能，并且同时反映在 **UART_IVS** 寄存器。此寄存器只能写入，并且只允许写入 1，无法写 0 取消关闭中断设置。
- **UART_IVS 中断功能有效状态寄存器**

  反映 **UART_IER** 与 **UART_IDR** 寄存器所设置的结果。0:中断关闭 1:中断开启
- **UART_RIF 原始中断状态寄存器**

  反映所有发生中断事件的状态，无论 **UART_IVS** 是否有开启中断，皆会反映在此寄存器中，主要提供用户监控无屏蔽的中断位，是否有错误事件发生。
- **UART_IFM 中断标志位状态寄存器**

记录中断开启位所发生中断事件。0:无中断事件 1:发生中断事件

- **UART_ICR 中断清除寄存器**

  此位设置 1 时，清除中断标志位 **UART_RIF** 与 **UART_IFM**，此寄存器通过写入 1 将该位清零，并且只允许写入 1 清除，无法写 0 清除。

在 UART 中，配置了 18 种中断，分别为下表。

中断事件	中断标志位
接收器字节格式错误	RXBERR
自动波特率侦测结束	ABEND
自动波特率侦测超时	ABTO
CTS 引脚电平改变	DCTS
接收超时	RXTO
地址匹配	ADDRM
LIN 断路侦测	LINBK
块结束	EOB
噪声位侦测	NOISE
接收器 FIFO 触发门坎	RFTH
接收器非空	RFNEMPTY
接收器已满	RFFULL
接收器溢出	RFOERR
接收器下溢	RFUERR
发送器字节完成	TBC
发送器 FIFO 触发门坎	TFTH
发送器空	TFEMPTY
发送器溢出	TFOVER

表 9-5　中断配置表

9. 5 特殊功能寄存器

9. 5. 1 寄存器列表

UART 寄存器列表			
名称	偏移地址	类型	描述
UART_RXDATA	0000_H	R	UART 接收缓冲寄存器
UART_TXDATA	0004_H	R/W	UART 发送缓冲寄存器
UART_BRR	0008_H	R/W	UART 波特率寄存器
UART_LCON	$000C_H$	R/W	UART 格式控制寄存器
UART_MCON	0010_H	R/W	UART 模式控制寄存器
UART_RS485	0014_H	R/W	UART RS485 控制寄存器
UART_SCARD	0018_H	R/W	UART 智能卡控制寄存器
UART_LIN	$001C_H$	R/W	UART LIN 控制寄存器
UART_RTOR	0020_H	R/W	UART 接收超时寄存器
UART_FCON	0024_H	R/W	UART FIFO 控制寄存器
UART_STAT	0028_H	R	UART 状态寄存器
UART_IER	$002C_H$	W1	UART 中断开启寄存器
UART_IDR	0030_H	W1	UART 中断关闭寄存器
UART_IVS	0034_H	R	UART 中断功能有效状态寄存器
UART_RIF	0038_H	R	UART 原始中断状态寄存器
UART_IFM	$003C_H$	R	UART 中断标志位状态寄存器
UART_ICR	0040_H	C_W1	UART 中断清除寄存器

9. 5. 2 寄存器描述

9. 5. 2. 1 UART接收缓冲寄存器(UART_RXDATA)

UART 接收缓冲寄存器(UART_RXDATA)																														
偏移地址:0x00																														
复位值:0x0000 0000																														
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																							RXDATA<8:0>								

—	Bits 31-9	—	—
RXDATA	Bits 8-0	R	**接收缓冲寄存器** 包含接收到的字节。 RXDATA寄存器提供接收移位寄存器和内部总线间的并行接口。 注:位8用于RS485地址模式

9. 5. 2. 2 UART发送缓冲寄存器(UART_TXDATA)

UART 发送缓冲寄存器 (UART_TXDATA)																														
偏移地址:0x04																														
复位值:0x0000 0000																														
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																							TXDATA<8:0>								

—	Bits 31-9	—	—
TXDATA	Bits 8-0	R/W	**发送缓冲寄存器** 用于写入要发送的数据字节。 TXDATA寄存器提供发送移位寄存器与内部总线间的并行接口。 注:位8用于RS485地址模式

9. 5. 2. 3 UART波特率寄存器(UART_BRR)

UART 波特率寄存器(UART_BRR)																													
偏移地址:0x08																													
复位值:0x0000 0000																													
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																BRR<15:0>															

—	Bits 31-16	—	—
BRR	Bits 15-0	R/W	**波特率寄存器** 整数部分 BRR[15:4] = DIVISOR[11:0] 小数部分 BRR[3:0] = FRACTION[3:0] 此位在LCON寄存器中的RXEN位与TXEN位为0时才可以写入。 注:使用自动波特率功能时则可自动写入

9. 5. 2. 4 UART格式控制寄存器(UART_LCON)

UART 格式控制寄存器(UART_LCON)																													
偏移地址:0x0C																													
复位值:0x0000 0000																													
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																TXEN	RXEN	DBCEN			BREAK	SWAP	TXINV	RXINV	DATAINV	MSB	PS	PE	STOP	DLS<1:0>	

—	Bits 31-16	—	—
TXEN	Bit 15	R/W	**发送器开启** 开启发送器，此位由软件设置1和清除。 0:发送器关闭 1:发送器开启
RXEN	Bit 14	R/W	**接收器开启** 开启接收器，此位由软件设置1和清除。 0:接收器关闭 1:接收器开启
DBCEN	Bit 13	R/W	**防抖动开启** 开启防抖动功能，此位由软件设置为1和清除。

			0:防抖动关闭 1:防抖动开启，在RX在线须维持8个时钟的电平
—	Bits 12-11	—	—
BREAK	Bit 10	R/W	**断路开启** 开启断路功能，此位由软件设置为1和清除。此位在LCON寄存器中的RXEN位与TXEN位为0时才可以写入。 0:断路关闭 1:断路开启，会使得TX输出为0
SWAP	Bit 9	R/W	**交换TX/RX引脚** 开启交换功能，此位由软件设置为1和清除。此位在LCON寄存器中的RXEN位与TXEN位为0时才可以写入。 0:交换关闭，TX和TX引脚按照标准引脚配置使用 1:交换开启，TX和RX引脚功能交换使用，此功能用与和其他UART接口进行交叉互联时
TXINV	Bit 8	R/W	**TX引脚电平反向** TX引脚反向功能，此位由软件设置为1和清除。此位在LCON寄存器中的RXEN位与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上带有外部反向器时
RXINV	Bit 7	R/W	**RX引脚电平反向** RX引脚反向功能，此位由软件设置为1和清除。此位在LCON寄存器中的RXEN位与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在线带有外部反向器时
DATAINV	Bit 6	R/W	**二进制反向开启** 二进制反向功能，此位由软件设置为1和清除。

			此位在LCON寄存器中的RXEN位与TXEN位为0时才可以写入。 0:缓冲寄存器中的逻辑数据在接收的时候，采用正/直接逻辑。 (1=H，0=L) 1:缓冲寄存器中的逻辑数据在接收的时候，采用负/反向逻辑。 (1=L，0=H)
MSB	Bit 5	R/W	**高位在前开启** 高位在前功能，此位由软件设置为1和清除。 此位在LCON寄存器中的RXEN位与TXEN位为0时才可以写入。 0:数据在发送和接收的时候，采用起使位后面跟着第0位的顺序 1:数据在发送和接收的时候，采用起使位后面跟着的最高位的顺序
PS	Bit 4	R/W	**校验位奇偶选择** 当开启校验功能时，选择校验位为奇校验或偶校验，此位由软件设置为1和清除。 此位在LCON寄存器中的RXEN位与TXEN位为0时才可以写入。 0:奇校验 1:偶校验
PE	Bit 3	R/W	**校验开启** 开启校验功能，计算好的校验位被插入到最高位，并检测接收数据的校验位(接收与发送功能)，此位由软件设置为1和清除。 此位在LCON寄存器中的RXEN位与TXEN位为0时才可以写入。 0:校验位关闭 1:校验位开启
STOP	Bit 2	R/W	**停止位选择** 此位由软件设置为1和清除。 此位在LCON寄存器中的RXEN位与TXEN位为0时才可以写入。 **普通模式下** 0: 1个停止位 1: 2个停止位(在5字长模式为1.5个停止位) **智能卡模式** 0: 0.5个停止位

			1: 1.5个停止位
DLS	Bits 1-0	R/W	**数据字长选择** 此位由软件设置为1和清除。 此位在LCON寄存器中的RXEN位与TXEN位为0时才可以写入。 00: 8字长 01: 7字长 10: 6字长 11: 5字长

9. 5. 2. 5 UART模式控制寄存器(UART_MCON)

UART 模式控制寄存器 (UART_MCON)

偏移地址:0x10

复位值:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
															TXFLOAT	TXDMAEN	RXDMAEN			ABRREPT	ABRMOD		ABREN			BKREQ	HDEN	IREN	AFCEN	RTSSET	LPBKEN

—	Bits 31-17	—	—
TXFLOAT	Bit 16	R/W	**发送器等待发送状态选择** 此位由软件设置为1和清除。 0:发送器未发送时，TX引脚输出高准位 1:发送器未发送时，TX引脚开漏
TXDMAEN	Bit 15	R/W	**发送器DMA开启** 此位由软件设置为1和清除。 0: 发送器DMA通信关闭 1: 发送器DMA通信开启
RXDMAEN	Bit 14	R/W	**接收器DMA开启** 此位由软件设置为1和清除。 0: 接收器DMA通信关闭 1: 接收器DMA通信开启
—	Bits 13-12	—	—
ABRREPT	Bit 11	R/W	**重复侦测自动波特率** 在开启侦测自动波特率时，侦测波特率超时时并不会清除自动波特率开关，并在下一个下降沿时重复侦测自动波特率，此位由软件

			设置为1和清除。 0:重复侦测自动波特率关闭 1:重复侦测自动波特率开启
ABRMOD	Bits 10-9	R/W	**自动波特率模式选择** 此位由软件设置为1和清除。 00:模式0，侦测第一个下降沿到第二个下降沿时间(侦测2位) 01:模式1，侦测第一个下降沿到第一个上升沿时间(侦测1位) 10:模式2，侦测第一个下降沿到第一个上升沿时间(侦测2位) 11:保留
ABREN	Bit 8	R/W	**自动波特率开启** 此位在开启并完成侦测自动波特率后会自动清除，也可由软件设置为1和清除。 0:自动波特率关闭 1:自动波特率开启
—	Bits 7-6	—	—
BKREQ	Bit 5	W	**断路请求** 此位在写入后的下一个时钟会自动清除。 0: 断路请求关闭 1: 断路请求开启，根据设定的N位长(8、7、6或5)产生N个低脉冲信号
HDEN	Bit 4	R/W	**单线半双工开启** 此位由软件设置为1和清除。 此位在LCON寄存器中的RXEN位与TXEN位为0时才可以写入。 0: 单线半双工关闭 1: 单线半双工开启
IREN	Bit 3	R/W	**IrDA红外线模式开启** 此位由软件设置为1和清除。 0: IrDA红外线模式关闭 1: IrDA红外线模式开启
AFCEN	Bit 2	R/W	**自动流量控制开启** 此位由软件设置为1和清除。 0:自动流量控制关闭 1:自动流量控制开启
RTSSET	Bit 1	R/W	**RTS设置控制**

			此位由软件设置为1和清除。 0:自动流量控制关闭时，RTS引脚输出高准位 1:自动流量控制关闭时，RTS引脚输出低准位
LPBKEN	Bit 0	R/W	**回送模式使能** 此模式是用于UART测试检测模式，在UART普通模式下运行，TX引脚输出为高电平，串行的数据在内部回送至RX。在此模式下，所有中断都是正常运行的，此位由软件设置为1和清除。 0:回送模式关闭 1:回送模式开启

9. 5. 2. 6 UART RS485 控制寄存器 (UART_RS485)

UART RS485 控制寄存器(UART_RS485)

偏移地址:0x14

复位值:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								DLY<7:0>								ADDR<7:0>												AADINV	AADACEN	AADNEN	AADEN

—	Bits 31-24	—	—
DLY	Bit 23-16	R/W	**延迟数值** 此位由软件设置和清除。 用于设置延迟RTS的输出时间，是由一个8位计数器计数，时钟源为1/16 Baud 时钟。
ADDR	Bit 15-8	R/W	**地址匹配述值** 此位由软件设置和清除。 用于多机通信时地址标记的检测。 接收器在RS485自动检测模式时，当接收数据的最高位为1且匹配ADDR，数据才允许接收，否则舍弃此数据。
—	Bits 7-4	—	—
AADINV	Bit 3	R/W	**驱动开启反向** 在自动流量控制模式时，设置驱动开启引脚(RTS/DE)的输出电平，此位由软件设置和清除。

			0:当发送器开始发送数据时，驱动开启引脚输出0，发送完成且TX内无数据时，驱动开启引脚输出1 1:当发送器开始发送数据时，驱动开启引脚输出1，发送完成且TX内无数据时，驱动开启引脚输出0
AADACEN	Bit 2	R/W	**自动流量控制模式开启** 此位由软件设置为1和清除。 0:自动流量控制模式关闭 1:自动流量控制模式开启
AADNEN	Bit 1	R/W	**普通模式开启** 此位由软件设置为1和清除。 0:普通模式关闭 1:普通模式开启，接收地址位为第8位(UART_RXDATA)
AADEN	Bit 0	R/W	**自动地址侦测模式开启** 在普通模式时，设置此位无效，此位由软件设置为1和清除。 0:自动地址侦测模式关闭 1:自动地址侦测模式开启，当接收数据的地址为1且匹配ADDR时，才会接收数据

9. 5. 2. 7 UART智能卡控制寄存器 (UART_SCARD)

UART 智能卡控制寄存器 (UART_SCARD)																															
偏移地址:0x18																															
复位值:0x0000 0000																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BLEN<7:0>								GT<7:0>								PSC<7:0>										SCCNT<2:0>			SCLKEN	SCNACK	SCEN

BLEN	Bit 31-24	R/W	**块长度** 设置了智能卡模式T=1的接收时的块长度，此位由软件设置为1和清除。 例如: BLEN = 0 → 0个信号字节 BLEN = 1 → 1个信号字节 BLEN = 255 → 255个信号字节 这个功能也可以在其他模式中使用，当LCON寄存器的RXEN位清除时，块长度计数器会重新计数
GT	Bit 23-16	R/W	**保护时间** 设置保护时间长度，是使用波特时钟为单位。在智能卡模式中使用，完成标志(RIF寄存器TBC位)在保护时间过后设置为1，此位由软件设置为1和清除
PSC	Bit 15-8	R/W	**分频器数值** 此位由软件设置为1和清除。 **在红外低功耗和正常模式下：** PSC[7:0]: IrDA正常和低功耗模式波特率 对UART时钟源进行分频以获得低功耗模式下的频率： 00000000: 保留 00000001: 1分频 00000010: 2分频 **智能卡模式:** PSC[4:0]: 输出时钟分频数值 用于设定UART时钟的分频数，得到智能卡输出时钟，由五个有效位组成，乘以2得到的数值作

			为分频 00000: 保留 00001: 2分频 00010: 4分频 00011: 6分频
—	Bits 7-6	—	—
SCCNT	Bit 5-3	R/W	**智能卡重试计数器** 设置智能卡模式中接收和发送的重试次数。此位由软件设置为1和清除。 在发送模式下，在产生帧错误前重试发送的次数 在接收模式下，在接收到NACK后重试接收的次数 0x0: 重试功能关闭，在发送与接收模式下不进行自动重试 0x1~0x7: 在产生错误前自动重试的次数
SCLKEN	Bit 2	R/W	**智能卡时钟开启** 此位由软件设置为1和清除。 0: CK引脚关闭 1: CK引脚开启
SCNACK	Bit 1	R/W	**智能卡 NACK 发送开启** 此位由软件设置为 1 和清除。 0: 出现校验错误时关闭发送 NACK 信号 1: 出现校验错误时开启发送 NACK 信号
SCEN	Bit 0	R/W	**智能卡模式开启** 此位由软件设置为 1 和清除。 0: 智能卡模式关闭 1: 智能卡模式开启

9. 5. 2. 8 UARTLIN控制寄存器 (UART_LIN)

UARTLIN 控制寄存器 (UART_LIN)																													
偏移地址:0x1C																													
复位值:0x0000 0000																													
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																													LINBKREQ	LINBDL	LINEN

—	Bits 31-3	—	—
LINBKREQ	Bit 2	W	**LIN模式断路请求** 在LIN模式下，发送器将发送13位'0’作为断路符号后，发送2位1用于对下一个开始位的检测。此位由软件设置为1并在下一个时钟后自动清除。 0: LIN模式断路请求关闭 1: LIN模式断路请求开启
LINBDL	Bit 1	R/W	**LIN模式断路字长** 此位由软件设置为1和清除。 0: 10位断路字节侦测 1: 11位断路字节侦测
LINEN	Bit 0	R/W	**LIN 模式开启** 此位由软件设置为 1 和清除。 0: LIN 模式关闭 1: LIN 模式开启

9. 5. 2. 9 UART接收器超时寄存器(UART_RTOR)

UART 接收器超时寄存器 (UART_RTOR)																															
偏移地址:0x20																															
复位值: UART1: 0x00FF FFFF UART2: 0x0000 00FF																															
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
							RTOEN	RTO<23: 0>																							

—	Bits 31-25	—	—
RTOEN	Bit 24	R/W	**接收器超时开启** 此位由软件设置为1和清除。 0:接收器超时关闭 1:接收器超时开启
RTO	Bits 23-0	R/W	**接收器超时数值** 设置接收超时时间，使用波特率时钟的字长为单位。 在标准模式下，接收最后一个字节后，在超时时间内未检测到新的起始位，将 RIF 寄存器的 RXTO 位设置为 1，此位由软件设置和清除。 在智能卡模式下，这个数值是用来实现 CWT 和 BWT。 注: UART2 只支持低 8 位 RTO<7:0>

9. 5. 2. 10 UARTFIFO控制寄存器 (UART_FCON)

UARTFIFO 控制寄存器 (UART_FCON)																													
偏移地址:0x24																													
复位值:0x0000 000																													
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
																TXFL<4:0>					TXTH<1:0>		TFRST	RXFL<4:0>					RXTH<1:0>		RFRST

—	Bits 31-16	—	—
TXFL	Bit 15-11	R	**发送器FIFO中资料笔数** 显示发送器FIFO内准备发送的笔数，此位由硬件设置且只能读取
TXTH	Bit 10-9	R/W	**发送器触发门坎** 设置发送器触发门坎，当TXFL笔数小于TXTH设定的笔数时，将设置UART_RIF寄存器的TFTH位与UART_STAT寄存器的TFTH位 00:发送器FIFO内无数据 01:发送器FIFO中资料少于或等于2笔 10:发送器FIFO中数据少于或等于其深度的1/4 11:发送器FIFO中数据少于或等于其深度的1/2 注：如果TXDMAEN使能，该位也提供发送FIFO缓存的DMA请求条件。
TFRST	Bit 8	W1	**发送器FIFO重置** 设置清除TX FIFO内字节，此位由软件设置1且在下一个时钟自动清除。
RXFL	Bit 7-3	R	**接收器FIFO中数据笔数** 显示接收器FIFO内准备接收的笔数，此位由硬件设置且只能读取
RXTH	Bit 2-1	R/W	**接收器触发门坎** 设置接收器触发门坎，当RXFL笔数到达RXTH设定的笔数时，将设置UART_RIF寄存器的RFTH位与UART_STAT寄存器的RFTH位 00:接收器FIFO中有1笔数据 01:接收器FIFO中数据达到其深度的1/4 10:接收器FIFO中数据达到其深度的1/2 11:接收器FIFO中数据再2笔达到其深度

			注：如果RXDMAEN使能，该位也提供接收FIFO缓存的DMA请求条件。
RFRST	Bit 0	W1	**接收器 FIFO 重置** 设置清除 RX FIFO 内字节，此位由软件设置 1 且在下一个时钟自动清除。

9. 5. 2. 11 UART状态寄存器(UART_STAT)

UART 状态寄存器(UART_STAT)

偏移地址:0x28

复位值:0x0001 0008

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
—	—	—	—	—	—	—	—	—	—	—	—	—	TFOERR	TFFULL	TFEMPTY	TFTH	TSBUSY	RFUERR	RFOERR	RFFULL	RFNEMPTY	RFTH	RSBUSY	—	—	—	—	CTSSTA	BKERR	FERR	PERR

—	Bit 31-19	—	—
TFOERR	Bit 18	R/C_R	**发送器溢出错误** 当发送器内已有数据时，有新数据再次写入TX中时，此位由硬件设置为1并舍弃新数据。此位由硬件设置为1，在发送数据或读取UART_STAT寄存器后清除 0:发送器溢出错误未产生 1:发送器溢出错误产生
TFFULL	Bit 17	R	**发送器FIFO满** 当发送器FIFO内满足数据时，此位由硬件设置1，在FIFO内未满足资料时清除。 0:发送器FIFO未满足资料 1:发送器FIFO满足资料
TFEMPTY	Bit 16	R	**发送器空** 当发送器内无任何数据时，此位由硬件设置为1，在TX写入资料时清除。 0:发送器有资料 1:发送器无资料
TFTH	Bit 15	R	**发送器FIFO触发门坎** 当UART_FCON寄存器的TXFL位小于TXTL设定的门坎，此位由硬件设置1且在未达到门坎清除。

			0:发送器FIFO未小于门坎 1:发送器FIFO小于门坎
TSBUSY	Bit 14	R	**发送器移位寄存器忙碌** 当写入数据由硬件设置为1在发送最后一个数据完成后清除。 0:发送器内无数据等待传送 1:发送器内有数据等待传送且未发送完最后一个数据
RFUERR	Bit 13	R/C_R	**接收器下溢错误** 当接收器无数据时，又再次读取接收器时，由硬件设置为1。此位由硬件设置为1，在接收数据或读取UART_STAT寄存器后清除 0:接收器下溢错误未产生 1:接收器下溢错误产生
RFOERR	Bit 12	R/C_R	**接收器溢位错误** 当接收器内已有数据时，有新数据再次接收时，此位由硬件设置为1并舍弃新数据。此位由硬件设置为1，在读取数据或读取UART_STAT寄存器后清除 0:接收器溢出错误未产生 1:接收器溢出错误产生
RFFULL	Bit 11	R	**接收器FIFO满** 当接收器FIFO内满足数据时，此位由硬件设置1，在FIFO内未满足资料时清除。 0:接收器FIFO未满足数据 1:接收器FIFO满足数据
RFNEMPTY	Bit 10	R	**接收器非空** 当接收器内有1笔数据时，此位由硬件设置为1，接收数据时清除。 0:接收器无数据 1:接收器有数据
RFTH	Bit 9	R	**接收器FIFO触发门坎** 当UART_FCON寄存器的RXFL位到达RXTL设定的门坎，此位由硬件设置1且在未达到门坎清除。 0:接收器FIFO未到达门坎 1:接收器FIFO到达门坎
RSBUSY	Bit 8	R	**接收移位寄存器忙碌**

			当接收数据时，由硬件设置为1在完成接收数据后清除 0:接收器未接收数据 1:接收器正在接收数据
—	Bit 7-4	—	—
CTSSTA	Bit 3	R	**CTS状态** 此位显示CTS输入引脚状态，由硬件设置为1和清除。 0: CTS输入引脚为0 1: CTS输入引脚为1
BKERR	Bit 2	R	**断路错误** 当接收数据与停止位皆为0时，由硬件设置为1。此位为显示当前读取接收器数值状态。 0:断路错误未产生 1:断路错误产生
FERR	Bit 1	R	**帧错误** 当接收数据的停止位为0时，由硬件设置为1。此位为显示当前读取接收器数值。 0:帧错误未产生 1:帧错误产生
PERR	Bit 0	R	**校验错误** 当接收数据的校验位接收错误时，由硬件设置为1。此位为显示当前读取接收器数值。 0:校验错误未产生 1:校验错误产生

9. 5. 2. 12 UART中断开启寄存器(UART_IER)

UART 中断开启寄存器(UART_IER)																														
偏移地址:0x2C																														
复位值:0x0000 0000																														
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
—	—	—	—	—	—	—	—	—	—	—	—	—	TFOERR	—	TFEMPTY	TFTH	TBC	RFUERR	RFOERR	RFFULL	RFNEMPTY	RFTH	NOISE	EOB	LINBK	ADDRM	RXTO	DCTS	ABTO	ABEND	RXBERR

—	Bit 31-19	—	—
TFOERR	Bit 18	W1	**开启发送器溢出中断功能** 此位设置时，开启中断功能，硬件侦测发送器溢出事件时发生中断
—	Bit 17	—	—
TFEMPTY	Bit 16	W1	**开启发送器空中断功能** 此位设置时，开启中断功能，硬件侦测发送器空事件时发生中断
TFTH	Bit 15	W1	**开启发送器FIFO触发门坎中断功能** 此位设置时，开启中断功能，硬件侦测发送器FIFO触发门坎事件时发生中断
TBC	Bit 14	W1	**开启发送器字节完成中断功能** 此位设置时，开启中断功能，硬件侦测发送器字节完成事件时发生中断
RFUERR	Bit 13	W1	**开启接收器下溢中断功能** 此位设置时，开启中断功能，硬件侦测接收器下溢事件时发生中断
RFOERR	Bit 12	W1	**开启接收器溢出中断功能** 此位设置时，开启中断功能，硬件侦测接收器溢出事件时发生中断
RFFULL	Bit 11	W1	**开启接收器FIFO满中断功能** 此位设置时，开启中断功能，硬件侦测接收器FIFO满事件时发生中断
RFNEMPTY	Bit 10	W1	**开启接收器非空中断功能** 此位设置时，开启中断功能，硬件侦测接收器非空事件时发生中断
RFTH	Bit 9	W1	**开启接收器FIFO触发门坎中断功能** 此位设置时，开启中断功能，硬件侦接收器FIFO

			触发门坎事件时发生中断
NOISE	Bit 8	W1	**开启侦测噪声位中断功能** 此位设置时，开启中断功能，硬件侦测噪声事件时发生中断
EOB	Bit 7	W1	**开启块结束中断功能** 此位设置时，开启中断功能，硬件侦测块结束事件时发生中断
LINBK	Bit 6	W1	**开启侦测LIN断路中断功能** 此位设置时，开启中断功能，硬件侦测LIN断路事件时发生中断
ADDRM	Bit 5	W1	**开启地址匹配中断功能** 此位设置时，开启中断功能，硬件侦测地址匹配事件时发生中断
RXTO	Bit 4	W1	**开启接收超时中断功能** 此位设置时，开启中断功能，硬件侦测接收超时事件时发生中断
DCTS	Bit 3	W1	**开启CTS引脚电平中断功能** 此位设置时，开启中断功能，硬件侦测CTS引脚电平事件时发生中断
ABTO	Bit 2	W1	**开启侦测自动波特率超时中断功能** 此位设置时，开启中断功能，硬件侦测侦测自动波特率超时事件时发生中断
ABEND	Bit 1	W1	**开启侦测自动波特率结束中断功能** 此位设置时，开启中断功能，硬件侦测侦测自动波特率事件时发生中断
RXBERR	Bit 0	W1	**开启接收器字节格式错误中断功能** 此位设置时，开启中断功能，硬件侦测接收器字节格式错误事件时发生中断

9. 5. 2. 13 UART中断关闭寄存器(UART_IDR)

UART 中断关闭寄存器(UART_IDR)																												
偏移地址:0x30																												
复位值:0x0000 0000																												
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
													TFOERR		TFEMPTY	TFTH	TBC	RFUERR	RFOERR	RFFULL	RFNEMPTY	RFTH	NOISE	EOB	LINBK	ADDRM	RXTO	DCTS	ABTO	ABEND	RXBERR

—	Bit 31-19	—	—
TFOERR	Bit 18	W1	**关闭发送器溢出中断功能** 此位设置时，关闭发送器溢出中断功能
—	Bit 17	—	—
TFEMPTY	Bit 16	W1	**关闭发送器空中断功能** 此位设置时，关闭发送器空中断功能
TFTH	Bit 15	W1	**关闭发送器FIFO触发门坎中断功能** 此位设置时，关闭发送器FIFO触发门坎中断功能
TBC	Bit 14	W1	**关闭发送器字节完成中断功能** 此位设置时，关闭发送器字节完成中断功能
RFUERR	Bit 13	W1	**关闭接收器下溢中断功能** 此位设置时，关闭接收器下溢中断功能
RFOERR	Bit 12	W1	**关闭接收器溢出中断功能** 此位设置时，关闭接收器溢出中断功能
RFFULL	Bit 11	W1	**关闭接收器FIFO满中断功能** 此位设置时，关闭接收器FIFO满中断功能
RFNEMPTY	Bit 10	W1	**关闭接收器非空中断功能** 此位设置时，关闭接收器非空中断功能
RFTH	Bit 9	W1	**关闭接收器FIFO触发门坎中断功能** 此位设置时，关闭接收器FIFO触发门坎中断功能
NOISE	Bit 8	W1	**关闭侦测噪声位中断功能** 此位设置时，关闭侦测噪声中断功能
EOB	Bit 7	W1	**关闭块结束中断功能** 此位设置时，关闭块结束中断功能
LINBK	Bit 6	W1	**关闭侦测LIN断路中断功能** 此位设置时，关闭侦测LIN断路中断功能

ADDRM	Bit 5	W1	**关闭地址匹配中断功能** 此位设置时，关闭地址匹配中断功能
RXTO	Bit 4	W1	**关闭接收超时中断功能** 此位设置时，关闭接收超时中断功能
DCTS	Bit 3	W1	**关闭CTS引脚电平中断功能** 此位设置时，关闭CTS引脚电平中断功能
ABTO	Bit 2	W1	**关闭侦测自动波特率超时中断功能** 此位设置时，关闭侦测自动波特率超时中断功能
ABEND	Bit 1	W1	**关闭侦测自动波特率结束中断功能** 此位设置时，关闭侦测自动波特率结束中断功能
RXBERR	Bit 0	W1	**关闭接收器字节格式错误中断功能** 此位设置时，关闭接收器字节格式错误中断功能

9. 5. 2. 14 UART中断功能有效状态寄存器(UART_IVS)

UART 中断功能有效状态寄存器 (UART_IVS)

偏移地址:0x34

复位值:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
—	—	—	—	—	—	—	—	—	—	—	—	—	TFOERR	—	TFEMPTY	TFTH	TBC	RFUERR	RFOERR	RFFULL	RFNEMPTY	RFTH	NOISE	EOB	LINBK	ADDRM	RXTO	DCTS	ABTO	ABEND	RXBERR

—	Bit 31-19	—	—
TFOERR	Bit 18	R	**发送器溢出中断功能状态** 0:中断功能处于关闭状态 1:中断功能处于开启状态
—	Bit 17	—	—
TFEMPTY	Bit 16	R	**发送器空中断功能状态** 0:中断功能处于关闭状态 1:中断功能处于开启状态
TFTH	Bit 15	R	**发送器FIFO触发门坎中断功能状态** 0:中断功能处于关闭状态 1:中断功能处于开启状态
TBC	Bit 14	R	**发送器字节完成中断功能状态**

			0:中断功能处于关闭状态 1:中断功能处于开启状态
RFUERR	Bit 13	R	**接收器下溢中断功能状态** 0:中断功能处于关闭状态 1:中断功能处于开启状态
RFOERR	Bit 12	R	**接收器溢出中断功能状态** 0:中断功能处于关闭状态 1:中断功能处于开启状态
RFFULL	Bit 11	R	**接收器非空中断功能功能状态** 0:中断功能处于关闭状态 1:中断功能处于开启状态
RFNEMPTY	Bit 10	R	**接收器非空中断功能状态** 0:中断功能处于关闭状态 1:中断功能处于开启状态
RFTH	Bit 9	R	**接收器FIFO触发门坎中断功能状态** 0:中断功能处于关闭状态 1:中断功能处于开启状态
NOISE	Bit 8	R	**侦测噪声位中断功能状态** 0:中断功能处于关闭状态 1:中断功能处于开启状态
EOB	Bit 7	R	**块结束中断功能状态** 0: 中断功能处于关闭状态 1: 中断功能处于开启状态
LINBK	Bit 6	R	**侦测LIN断路中断功能状态** 0: 中断功能处于关闭状态 1: 中断功能处于开启状态
ADDRM	Bit 5	R	**地址匹配中断功能状态** 0:中断功能处于关闭状态 1:中断功能处于开启状态
RXTO	Bit 4	R	**接收超时中断功能状态** 0:中断功能处于关闭状态 1:中断功能处于开启状态
DCTS	Bit 3	R	**CTS引脚电平中断功能状态** 0:中断功能处于关闭状态 1:中断功能处于开启状态
ABTO	Bit 2	R	**侦测自动波特率超时中断功能状态** 0:中断功能处于关闭状态 1:中断功能处于开启状态

ABEND	Bit 1	R	**侦测自动波特率结束中断功能状态** 0:中断功能处于关闭状态 1:中断功能处于开启状态
RXBERR	Bit 0	R	**接收器字节格式错误中断功能状态** 0:中断功能处于关闭状态 1:中断功能处于开启状态

UART_IVS 寄存器，是实时反映系统配置 UART_IER 与 UART_IDR 的中断开启状态。此寄存器状态是将 UART_IER 与 UART_IDR 进行硬件运算，公式如下:UART_IVS = UART_IER & ~UART_IDR

9. 5. 2. 15 UART原始中断状态寄存器(UART_RIF)

UART 原始中断状态寄存器(UART_RIF)

偏移地址:0x38

复位值:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
													TFOERR		TFEMPTY	TFTH	TBC	RFUERR	RFOERR	RFFULL	RFNEMPTY	RFTH	NOISE	EOB	LINBK	ADDRM	RXTO	DCTS	ABTO	ABEND	RXBERR

—	Bit 31-19	—	—
TFOERR	Bit 18	R	**发送器溢出，原始中断状态** 0:无发生中断 1:已发生中断
—	Bit 17	—	—
TFEMPTY	Bit 16	R	**发送器空，原始中断状态** 0:无发生中断 1:已发生中断
TFTH	Bit 15	R	**发送器FIFO触发门坎，原始中断状态** 0:无发生中断 1:已发生中断
TBC	Bit 14	R	**发送器字节完成，原始中断状态** 0:无发生中断 1:已发生中断
RFUERR	Bit 13	R	**接收器下溢，原始中断状态** 0:无发生中断 1:已发生中断
RFOERR	Bit 12	R	**接收器溢出，原始中断状态**

			0:无发生中断 1:已发生中断
RFFULL	Bit 11	R	**接收器FIFO满，原始中断状态** 0:无发生中断 1:已发生中断
RFNEMPTY	Bit 10	R	**接收器非空，原始中断状态** 0:无发生中断 1:已发生中断
RFTH	Bit 9	R	**接收器FIFO触发门坎，原始中断状态** 0:无发生中断 1:已发生中断
NOISE	Bit 8	R	**侦测噪声位，原始中断状态** 0:无发生中断 1:已发生中断
EOB	Bit 7	R	**块结束，原始中断状态** 0: 无发生中断 1: 已发生中断
LINBK	Bit 6	R	**侦测LIN断路，原始中断状态** 0: 无发生中断 1: 已发生中断
ADDRM	Bit 5	R	**地址匹配，原始中断状态** 0:无发生中断 1:已发生中断
RXTO	Bit 4	R	**接收超时，原始中断状态** 0:无发生中断 1:已发生中断
DCTS	Bit 3	R	**CTS引脚电平，原始中断状态** 0:无发生中断 1:已发生中断
ABTO	Bit 2	R	**侦测自动波特率超时，原始中断状态** 0:无发生中断 1:已发生中断
ABEND	Bit 1	R	**侦测自动波特率，原始中断状态** 0:无发生中断 1:已发生中断
RXBERR	Bit 0	R	**接收器字节格式错误，原始中断状态** 0:无发生中断 1:已发生中断

9. 5. 2. 16 UART中断标志位状态寄存器(UART_IFM)

UART 中断标志位状态寄存器 (UART_IFM)																														
偏移地址:0x3C																														
复位值:0x0000 0000																														
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
—	—	—	—	—	—	—	—	—	—	—	—	—	TFOERR	—	TFEMPTY	TFTH	TBC	RFUERR	RFOERR	RFFULL	RFNEMPTY	RFTH	NOISE	EOB	LINBK	ADDRM	RXTO	DCTS	ABTO	ABEND	RXBERR

—	Bit 31-19	—	—
TFOERR	Bit 18	R	**发送器溢出，标志位中断状态** 0:无发生中断 1:已发生中断
—	Bit 17	—	—
TFEMPTY	Bit 16	R	**发送器空，标志位中断状态** 0:无发生中断 1:已发生中断
TFTH	Bit 15	R	**发送器FIFO触发门坎，标志位中断状态** 0:无发生中断 1:已发生中断
TBC	Bit 14	R	**发送器字节完成，标志位中断状态** 0:无发生中断 1:已发生中断
RFUERR	Bit 13	R	**接收器下溢，标志位中断状态** 0:无发生中断 1:已发生中断
RFOERR	Bit 12	R	**接收器溢出，标志位中断状态** 0:无发生中断 1:已发生中断
RFFULL	Bit 11	R	**接收器FIFO满，标志位中断状态** 0:无发生中断 1:已发生中断
RFNEMPTY	Bit 10	R	**接收器非空，标志位中断状态** 0:无发生中断 1:已发生中断
RFTH	Bit 9	R	**接收器FIFO触发门坎，标志位中断状态** 0:无发生中断

			1:已发生中断
NOISE	Bit 8	R	**侦测噪声位，标志位中断状态** 0:无发生中断 1:已发生中断
EOB	Bit 7	R	**块结束，标志位中断状态** 0: 无发生中断 1: 已发生中断
LINBK	Bit 6	R	**侦测LIN断路，标志位中断状态** 0: 无发生中断 1: 已发生中断
ADDRM	Bit 5	R	**地址匹配，标志位中断状态** 0:无发生中断 1:已发生中断
RXTO	Bit 4	R	**接收超时，标志位中断状态** 0:无发生中断 1:已发生中断
DCTS	Bit 3	R	**CTS引脚电平，标志位中断状态** 0:无发生中断 1:已发生中断
ABTO	Bit 2	R	**侦测自动波特率超时，标志位中断状态** 0:无发生中断 1:已发生中断
ABEND	Bit 1	R	**侦测自动波特率，标志位中断状态** 0:无发生中断 1:已发生中断
RXBERR	Bit 0	R	**接收器字节格式错误，标志位中断状态** 0:无发生中断 1:已发生中断

UART_IFM 寄存器，是滤除已关闭中断功能的中断事件，只关注开启中断功能的事件。

此寄存器状态是将 UART_RIF 与 UART_IVS 进行硬件运算，公式如下:UART_IFM = UART_RIF & UART_IVS

9. 5. 2. 17 UART中断清除寄存器 (UART_ICR)

UART 中断清除寄存器(UART_ICR)																														
偏移地址:0x40																														
复位值:0x0000 0000																														
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
													TFOERR		TFEMPTY	TFTH	TBC	RFUERR	RFOERR	RFFULL	RFNEMPTY	RFTH	NOISE	EOB	LINBK	ADDRM	RXTO	DCTS	ABTO	ABEND	RXBERR

—	Bit 31-19	—	—
TFOERR	Bit 18	C_W1	**清除发送器溢出中断状态** 此位设置时，清除中断状态(UART_RIF与UART_IFM)
—	Bit 17	—	—
TFEMPTY	Bit 16	C_W1	**清除发送器空中断状态** 此位设置时，清除中断状态(UART_RIF与UART_IFM)
TFTH	Bit 15	C_W1	**清除发送器FIFO触发门坎中断状态** 此位设置时，清除中断状态(UART_RIF与UART_IFM)
TBC	Bit 14	C_W1	**清除发送器字节完成中断状态** 此位设置时，清除中断状态(UART_RIF与UART_IFM)
RFUERR	Bit 13	C_W1	**清除接收器下溢中断状态** 此位设置时，清除中断状态(UART_RIF与UART_IFM)
RFOERR	Bit 12	C_W1	**清除接收器溢出中断状态** 此位设置时，清除中断状态(UART_RIF与UART_IFM)
RFFULL	Bit 11	C_W1	**清除接收器FIFO满中断状态** 此位设置时，清除中断状态(UART_RIF与UART_IFM)
RFNEMPTY	Bit 10	C_W1	**清除接收器非空中断状态** 此位设置时，清除中断状态(UART_RIF与UART_IFM)
RFTH	Bit 9	C_W1	**清除接收器FIFO触发门坎中断状态** 此位设置时，清除中断状态(UART_RIF与

			UART_IFM)
NOISE	Bit 8	C_W1	**清除侦测噪声位中断状态** 此位设置时，清除中断状态(UART_RIF与UART_IFM)
EOB	Bit 7	C_W1	**清除块结束中断状态** 此位设置时，清除中断状态(UART_RIF与UART_IFM)
LINBK	Bit 6	C_W1	**清除侦测LIN断路中断状态** 此位设置时，清除中断状态(UART_RIF与UART_IFM)
ADDRM	Bit 5	C_W1	**清除地址匹配中断状态** 此位设置时，清除中断状态(UART_RIF与UART_IFM)
RXTO	Bit 4	C_W1	**清除接收超时中断状态** 此位设置时，清除中断状态(UART_RIF与UART_IFM)
DCTS	Bit 3	C_W1	**清除CTS引脚电平中断状态** 此位设置时，清除中断状态(UART_RIF与UART_IFM)
ABTO	Bit 2	C_W1	**清除侦测自动波特率超时中断状态** 此位设置时，清除中断状态(UART_RIF与UART_IFM)
ABEND	Bit 1	C_W1	**清除侦测自动波特率结束中断状态** 此位设置时，清除中断状态(UART_RIF与UART_IFM)
RXBERR	Bit 0	C_W1	**清除接收器字节格式错误中断状态** 此位设置时，清除中断状态(UART_RIF与UART_IFM)

UART_ICR 寄存器设置时，将清除 UART_RIF 与 UART_IFM 中断标志位状态；此设置不影响中断 UART_IER、UART_IDR 与 UART_IVS 寄存器，只清除标志位状态 UART_RIF 与 UART_IFM。此寄存器通过硬件清除中断，公式如下:UART_RIF = UART_RIF & ~UART_ICR

第10章 直接存储器访问控制器 (DMA)

10. 1 概述

直接存储器访问控制器(DMA)可独立于 CPU 对内部存储进行操作，利用 DMA 可减轻 CPU 的负担并且节省功耗。DMA 控制器可以通过它的 6 个通道来实现在存储器和外设之间的数据传输(外设到存储器，存储器到外设或者存储器到存储器)。

10. 2 特性

- 单 AHB 主控制总线
- 支持 6 个 DMA 通道。每个通道支持单一方向的传输
- 支持来源地址和目的地地址的增加模式或固定模式
- 来源和目的地的传输大小(字节、半字、字)彼此独立，来源和目的地的地址必须根据传输数据的大小进行对齐
- 硬件通道优先级。DMA 通道 0 具有最高优先级和通道 5 具有最低优先级
- 软件通道优先级。设置软件通道优先级(每个通道有 4 个级别 ：最高、高、中、低)，当两个 DMA 通道设置为相同的优先级时，它将根据硬件通道优先级被执行
- 支持循环模式
- 可编程的传输数量: 0 到 2^{16}-1
- 进行多个或单个 DMA 传输
- 进行以下不同类型的 DMA 传输
  - ◇ 存储器到存储器
  - ◇ 存储器到外设
  - ◇ 外设到存储器
  - ◇ 外设到外设

10. 3 請求映射

DMA 控制器通过 DMAPMUX 外设连接至 AHB 或 APB 外设的 DMA 请求。

10. 4 结构图

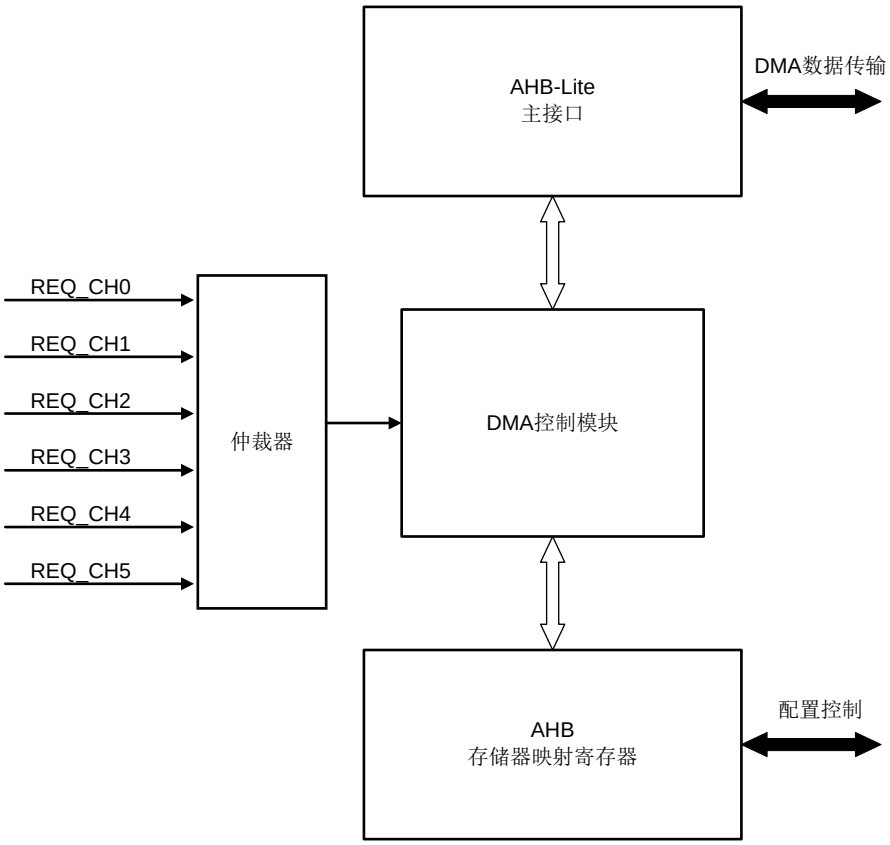


图 10-1　DMA 结构框图

DMA 控制器通过与其他系统主设备共享 AHB 系统总线来执行直接存储器传输。总线矩阵执行循环调度。当 CPU 与 DMA 访问相同的存储器或外设地址时，DMA 请求会将 CPU 对系统总线的访问停止数个总线周期。

根据通过软件的配置，DMA 控制器会在 DMA 通道及其接收的相关请求之间进行仲裁。 DMA 控制器还会通过单一 AHB 主控制总线调度 DMA 数据传输。

10. 5 功能描述

10. 5. 1 传输事务

配置 DMA 控制器中的通道级别，以便执行数据传输，此类传输由一系列 AHB 总线传输组成。可以通过外设请求或在存储器到存储器(M2M = 1)模式时设置 DMA_CONn 寄存器的 CHEN 位为 1 触发数据传输。

当触发事件发生后，会按照以下步骤进行传输 ：

1. 外设向 DMA 控制器发送 DMA 请求信号。
2. DMA 控制器按照与此外设请求相关的通道的优先级来处理该请求。
3. 只要 DMA 控制器授权给外设，当传输到最后一笔时 DMA 控制器就会向外设发送确认信号。
4. 外设获得 DMA 控制器的确认信号后，便会立即释放其请求信号。
5. 一但外设释放请求信号后 DMA 控制器就会释放确认信号。

外设可继续发送请求，再次启动 DMA 传输。

不论外设是传输来源或是传输目的地，都使用请求与应答协议。例如外设到存储器传输，外设向 DMA 控制器发送请求信号来启动传输。DMA 控制器随后读取外设单笔数据，并将该数据写入存储器。

对于给定通道 n，DMA 数据传输由以下的重复序列组成:

- 单次 DMA 传输(DMA_CONn.MAX_BURST 为 0)，需要两次总线访问(读取数据与写入数据)来执行传输(通过 DMA AHB 主控制总线实现) :
  - DMA 控制器获得请求信号。
  - 通过 DMA_SARn 寄存器从外设数据寄存器或存储器单元中读取单个数据(字节、半字或字)。
  - 通过 DMA_DARn 寄存器向外设数据寄存器或存储器单元中写入单个数据(字节、半字或字)。
  - DMA_NDTn 寄存器中 NRDT 位域在传输后递减，该寄存器包含剩余数据传输数量("先读后写"的传输次数)。
  - DMA 控制器发送确认信号。
  - 重复该序列，直到 DMA_NDTn 寄存器中 NRDT 位域等于 0。
- 多次 DMA 传输(DMA_CONn.MAX_BURST 不为 0)，需要多组两次总线访问(读取数据与写入数据)来执行传输(通过 DMA AHB 主控制总线实现) :
  - DMA 控制器获得请求信号。
  - 通过 DMA_SARn 寄存器从外设数据寄存器或存储器单元中读取单个数据(字节、半字或字)。

- 通过 DMA_DARn 寄存器向外设数据寄存器或存储器单元中写入单个数据(字节、半字或字)。
- DMA_NDTn 寄存器中 NRDT 位域在传输后递减，该寄存器包含剩余数据传输数量("先读后写"的传输次数)。
- 当传输次数等于 DMA_CONn.MAX_BURST 设定值。
- DMA 控制器发送确认信号。
- 重复该序列，直到 DMA_NDTn 寄存器中 NRDT 位域等于 0。

10. 5. 2 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，用来配置在重新进行仲裁前 AHB 总线传输的次数。这些位称为 MAX_BURST 位，输入的数值会提高到 2 的次方(2^{MAX_BURST})，进而决定了仲裁率。当 MAX_BURST = 4，则仲裁率为 2^4 = 16，即每进行 16 次 DMA 先读后写传输就进行一次仲裁。

MAX_BURST	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 10-1 仲裁率设置

注：优先级低的通道中 MAX_BURST 值不要配置太大，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。

DMA 通道的传输数量 **DMA_NDTn**.TNDT 是由用户指定的。当 **DMA_NDTn**.NRDT>2^{MAX_BURST} 且 **DMA_NDTn**.NRDT 不是 2^{MAX_BURST} 的整数倍，控制器会依次先完成 2^{MAX_BURST} 次传输直到剩余的次数小于 2^{MAX_BURST}。

当剩余的次数小于 2^{MAX_BURST} 时数据传输方式有两种 ：

1. 在存储器到存储器模式或存储器到外设模式时，剩余的次数会在下一次请求时完成
2. 在外设到存储器模式时，每次外设请求只会执行一次传输，直到传输完成

10. 5. 3 优先级

DMA 仲裁器管理不同通道间的优先级。

当 DMA 仲裁器授予某个有效通道 n 优先权后，根据 DMA_CONn.MAX_BURST 设定值会发起单次或多次 DMA 先读后写传输。随后仲裁器会再次对有效通道进行仲裁，并选择优先级最高的通道。

优先级分两个阶段进行管理:

- 软件: 可以在 DMA_CONn 寄存器中的 CHPRI 位域配置该通道优先级，分为四个级别 :
  - 最高优先级
  - 高优先级
  - 中优先级
  - 低优先级
- 硬件:如果两个通道请求具有相同的软件优先级，则通道编号低优先于通道编号高。例如通道 2 的优先级高于通道 4。

表 10-2 DMA 通道优先级

10. 5. 4 传输模式

10. 5. 4. 1 存储器到存储器模式(Memory-to-memory)

DMA 通道在没有外设请求触发的情况下同样可以工作。该模式被称为存储器到存储器模式，由 **DMA_CONn** 寄存器的 M2M 位设置 1，且 CHEN 位也设置 1 时启动传输。在 **DMA_NDTn** 寄存器的 NRDT 位域等于 0 后传输停止。

存储器到存储器模式的示例:

配置 DMA 通道 n 为存储器到存储器模式，将数据从 SRAM 地址 0x2000_0000 搬移到 0x2000_000。

配置流程如下:

1. 设置来源地址: **DMA_SARn** = 0x2000_0000
2. 设置目的地地址: **DMA_DARn** = 0x2000_1000
3. 设置总数据传输数量: **DMA_NDTn**.TNDT = 0x400
4. 使能存储器到存储器模式: **DMA_CONn**.M2M = 1
5. 使能通道: **DMA_CONn**.CHEN = 1

注:在循环模式下，不得使用存储器到存储器模式。在存储器到存储器模式(DMA_CONn.M2M = 1)下使能通道(DMA_CONn.CHEN = 1)前，软件必须将 DMA_CONn 寄存器的 CIRC 位清零。

10. 5. 4. 2 存储器到外设模式(Memory-to-peripheral)

DMA 通道在 **DMA_CONn** 寄存器的 DIR 位设置 1 且 M2M 为 0 时，DMA 控制器工作在存储器到外设模式下。

存储器到外设模式的示例:

配置 DMA 通道 n 为存储器到外设模式，将数据从 SRAM 地址 0x2000_0000 搬移到 UART 外设的 **UART_TXBUF**，此外设必须开启发送器 DMA(**UART_MCON**.TXDMAEN = 1)。

配置流程如下:

1. 设置来源地址: **DMA_SARn** = 0x2000_0000
2. 设置目的地地址: **DMA_DARn** = **UART_TXBUF**
3. 设置总数据传输数量: **DMA_NDTn**.TNDT = 0x400
4. 禁止存储器到存储器模式: **DMA_CONn**.M2M = 0
5. 使能存储器到外设模式: **DMA_CONn**.DIR = 1
6. 使能通道: **DMA_CONn**.CHEN = 11

10. 5. 4. 3 外設到存儲器模式(Peripheral-to-memory)

DMA 通道在 **DMA_CONn** 寄存器的 DIR 位设置 0 且 M2M 为 0 时，DMA 控制器工作在外设到存储器模式下。

外设到存储器模式的示例:

配置 DMA 通道 n 为外设到存储器模式，将数据从 UART 外设的 **UART_RXBUF** 搬移到 SRAM 地址 0x2000_0000，此外设必须开启接收器 DMA (**UART_MCON**.RXDMAEN = 1)。

配置流程如下:

1. 设置来源地址: **DMA_DARn** = **UART_TXBUF**
2. 设置目的地地址: **DMA_SARn** = 0x2000_0000
3. 设置总数据传输数量: **DMA_NDTn**.TNDT = 0x400
4. 禁止存储器到存储器模式: **DMA_CONn**.M2M = 0
5. 使能外设到存储器模式: **DMA_CONn**.DIR = 1
6. 使能通道: **DMA_CONn**.CHEN = 11

10. 5. 4. 4 外設到外設模式(Peripheral-to-peripheral)

DMA 控制器可通过软件配置外设的方式与限制，实现外设到外设模式。

外设到外设模式的示例:

配置 DMA 通道 n 为外设到存储器模式，将数据从 UART 外设 **UART_RXBUF** 搬移到 UART 外设的 **UART_TXBUF**，此外设仅可开启接收器 DMA(**UART_MCON**.RXDMAEN = 1)，并关闭发送器 DMA (**UART_MCON**.TXDMAEN = 0)。

配置流程如下:

1. 设置来源地址:**DMA_SARn = UART_RXBUF**

2. 设置目的地地址: **DMA_DARn = UART_TXBUF**
3. 设置总数据传输数量: **DMA_NDTn**.TNDT = 0x400
4. 禁止存储器到存储器模式: **DMA_CONn**.M2M = 0
5. 使能外设到存储器模式: **DMA_CONn**.DIR = 1
6. 使能通道: **DMA_CONn**.CHEN = 11

注:

1. 在外设到外设模式下，DMA_CONn 寄存器的 MAX_BURST 位域必须设置为 0。
2. 因为 DMA 每个通道仅有一个请求信号，所以在此模式时仅能启动一端外设的 DMA 功能。

10. 5. 5 地址递增

根据 **DMA_CSRn** 寄存器中 SINC 和 DINC 的设置，再每次传输后，来源和目的地地址可以选择自动递增或保持不变。

如果使能了递增模式(SINC 或 DINC 置 1)，则下次传输的地址是前一次传输的地址加上 1(对应于字节传输)、2(对应于半字传输)或 4(对应于字传输)，具体取决于在 **DMA_CONn** 寄存器中的 SDWSEL[1:0]或 DDWSEL[1:0]中定义的数据宽度。首次传输的地址可在 **DMA_SARx** 或 **DMA_DARx** 寄存器中进行编程。在传输过程中，这些寄存器将保持初始编程的值。软件无法获得当前传输的地址。

如果将通道配置为非循环模式，则在最后一次数据传输完成后，将不处理任何 DMA 请求，并将 **DMA_CONn** 寄存器的 CHEN 清零。

注:如果禁止通道 n，DMA 寄存器不会被复位。DMA_CONn、DMA_SARn 和 DMA_DARn 寄存器仍保留在通道配置阶段设置的初始值。

在循环模式下，最后一次数据传输完成后，当前的内部地址寄存器重新加载 **DMA_SARn** 和 **DMA_DARn** 寄存器中的基址值，**DMA_NDTn** 寄存器的 NRDT 位域重新加载 TNDT 数据数量。

10. 5. 6 循环模式(在存储器到外设和外设到存储器传输模式)

循环模式可用于处理循环缓冲区和连续数据流(例如 ADC 扫描模式)。可以使用 **DMA_CONn** 寄存器中的 CIRC 使能该功能。

在存储器到存储器模式下，不能使用循环模式。在循环模式(CIRC = 1)下使能通道前，软件必须将 **DMA_CONn** 寄存器的 M2M 位清零。

当使能循环模式时，在最后一次数据传输完成后，**DMA_NDTn** 寄存器的 NRDT 位域将重新加载 TNDT 数据数量，并继续响应 DMA 请求。

退出循环模式可通过两种方式:

- 当发生块传输完成中断事件后，通过软件将 **DMA_CONn** 寄存器的 CIRC 位清零，此时 DMA 控制器会持续响应外设请求，直到再次发生块传输完成中断事件后，硬件会将

DMA_CONn 寄存器中的 CHEN 位清零且停止响应外设请求。

- 软件需要先将外设停止生成 DMA 请求，接着通过软件将 **DMA_CONn** 寄存器中的 CHEN 位与 CIRC 位清零，停止响应外设的请求并退出循环模式。

10. 5. 7 数据宽度、对齐和字节序

当 **DMA_CONn** 寄存器的 SDWSEL[1:0]和 DDWSEL[1:0]不相等时，则 DMA 控制器将按照下表所述方式进行数据对齐。

来源端口宽度 (SDWSEL)	目的地端口宽度 (DDWSEL)	传输数量 (TNDT)	来源地址/数据	目的地地址/数据
8	8	4	0x0/0xB0	0x0/0xB0
			0x1/0xB1	0x1/0xB1
			0x2/0xB2	0x2/0xB2
			0x3/0xB3	0x3/0xB3
8	16	4	0x0/0xB0	0x0/0x00B0
			0x1/0xB1	0x2/0x00B1
			0x2/0xB2	0x4/0x00B2
			0x3/0xB3	0x6/0x00B3
8	32	4	0x0/0xB0	0x0/0x000000B0
			0x1/0xB1	0x4/0x000000B1
			0x2/0xB2	0x8/0x000000B2
			0x3/0xB3	0xC/0x000000B3
16	8	4	0x0/0xB1B0	0x0/0xB0
			0x2/0xB3B2	0x1/0xB2
			0x4/0xB5B4	0x2/0xB4
			0x6/0xB7B6	0x3/0xB6
16	16	4	0x0/0xB1B0	0x0/0xB1B0
			0x2/0xB3B2	0x2/0xB3B2
			0x4/0xB5B4	0x4/0xB5B4
			0x6/0xB7B6	0x6/0xB7B6
16	32	4	0x0/0xB1B0	0x0/0x0000B1B0
			0x2/0xB3B2	0x4/0x0000B3B2
			0x4/0xB5B4	0x8/0x0000B5B4
			0x6/0xB7B6	0xC/0x0000B7B6
32	8	4	0x0/0xB3B2B1B0	0x0/0xB0

来源端口宽度 (SDWSEL)	目的地端口宽度 (DDWSEL)	传输数量 (TNDT)	来源地址/数据	目的地地址/数据
			0x4/0xB7B6B5B4	0x1/0xB4
			0x8/0xBBBAB9B8	0x2/0xB8
			0xC/0xBFBEBDBC	0x3/0xBC
32	16	4	0x0/0xB3B2B1B0	0x0/0xB1B0
			0x4/0xB7B6B5B4	0x2/0xB5B4
			0x8/0xBBBAB9B8	0x4/0xB9B8
			0xC/0xBFBEBDBC	0x6/0xBDBC
32	32	4	0x0/0xB3B2B1B0	0x0/0xB3B2B1B0
			0x4/0xB7B6B5B4	0x4/0xB7B6B5B4
			0x8/0xBBBAB9B8	0x8/0xBBBAB9B8
			0xC/0xBFBEBDBC	0xC/0xBFBEBDBC

表 10-3　可编程的数据宽度和字节序(SINC = DINC = 1 时)

10. 5. 7. 1　解决AHB外设不支持字节或半字写传输的问题

如果 DMA 控制器启动 AHB 字节或半字写传输，则 32 位 AHB 主控数据总线(HWDATA[31:0])的未使用数据在线将会重复所传输的数据。

如果 AHB 从外设不支持字节或半字写传输且不会产生任何错误，则 DMA 控制器会对 HWDATA 的 32 个位执行写操作，如下列两个范例所示：

- 要写入半字数据为 0xABCD，DMA 控制器会将 HWDATA 总线设为 0xABCDABCD，并采用半字数据大小(在 AHB 主控总线中，将 HSIZE 设为 HalfWord)。
- 要写入字节数据为 0xAB，DMA 控制器会将 HWDATA 总线设为 0xABABABAB，并采用字节数据大小(在 AHB 主控总线中，将 HSIZE 设为 Byte)。

当目的地地址为 APB 外设，且该外设不考虑 HSIZE 数据(寄存器为字切齐)，则任何 AHB 字节或半字传输都将转换为 32 位 APB 传输，如下所述：

- 将 AHB 字节写传输转换为 APB 字写传输，如向 0x0、0x1、0x2 或 0x3 地址之一写入数据 0xB0，将转换为向 0x0 地址写 0xB0B0B0B0。
- 将 AHB 半字写传输转换为 APB 字写传输，如向 0x0 或 0x2 地址写入数据 0xB1B0，将转换为向 0x0 地址写入 0xB1B0B1B0。

10. 5. 8　通道配置流程

配置 DMA 通道 n 时需按照以下步骤操作：

1. 在 **DMA_IER** 寄存器中配置下列参数：
   - 传输完成一半或全部完成时的中断使能
2. 将来源地址设置到 **DMA_SARn** 寄存器。

使能通道后，在存储器到存储器模式或外设请求发生后，DMA 控制器会通过该地址读取数据。

3. 将目的地地址设置 **DMA_DARn** 寄存器。
   使能通道后，在存储器到存储器模式或外设请求发生后，DMA 控制器会通过该地址写入数据。
4. 在 **DMA_NDTn** 寄存器的 TNDT 位域配置传输的总数据数。
   当通道使能(软件对 **DMA_CONn**.CHEN 设置为 1)时，NRDT 会自动加载 TNDT 数值，并每次数据传输后，NRDT 位域都会递减。
5. 在 **DMA_CONn** 寄存器中配置下列参数：
   - 通道优先级
   - 数据传输方向
   - 循环模式
   - 外设和存储器递增模式
   - 外设和存储器数据大小
   - 仲裁率
6. 将 **DMA_CONn** 寄存器中的 CHEN 位置 1 以使能通道。

通道在使能后可处理来自此通道所连接外设的任意 DMA 请求，或者启动存储器到存储器数据传输。

注：通道配置流程的最后两步可合并为对 DMA_CONn 寄存器进行单次访问来配置和使能通道。

10. 5. 9 传输完成

- 通道被配置为非循环模式

  在 DMA_NDTn 寄存器的 NRDT 位域递减到零则传输结束，通道被禁止(DMA_CONn 寄存器中的 CHEN 位被硬件清零)并将 DMA_RIF.CHnTC 设置为 1，此时 DMA 控制器不会响应外设请求，除非软件重新编程并重新启用它(通过设置 DMA_CONn 寄存器中的 CHEN 位为 1)。

- 通道被配置为循环模式

  在 DMA_NDTn 寄存器的 NRDT 位域递减到零时，NRDT 位域将重新加载 TNDT 数据数量并将 DMA_RIF.CHnTC 设置为 1，此时 DMA 控制器会继续响应外设的请求。

10. 5. 10 传输暂停

可以随时暂停 DMA 传输以供稍后重新开始。

分为两种情况：

- 禁止传输，暂停以后不从停止点重新开始。这种情况下只需将 **DMA_CONn** 寄存器中的 CHEN 位清零，除此之外不需要任何其他操作。禁止传输可能要花费一些时间(需要首先完成正在进行的传输)。等待 **DMA_CONn** 寄存器中的 CHEN 位的值为 0，藉此确认已经终止传输。**DMA_NDTn** 寄存器的 NRDT 位域包含数据停止时剩余数据项的数目，这样软

件便可以确定数据传输中断前已传输了多少数据项。

- 暂停并恢复传输，在 **DMA_NDTn** 寄存器中的 TNDT 位域递减到 0 之前暂停传输。目的是以后通过重新使能通道并重新开始传输。为了在传输停止点重新开始传输，软件必须在通过写入 **DMA_CONn** 寄存器中的 CHEN 位(然后检查确认该位为 0)禁止通道之后，首先读取 **DMA_NDTn** 寄存器中的 NRDT 位域来了解已经传输的数据项的数目。然后：
  - 必须更新来源或目的地地址以调整地址指针
  - 必须使用要传输的剩余数据项的数目，在暂停时读取 **DMA_CONn** 寄存器的 NRDT 位域数值，并更新到 **DMA_CONn** 寄存器的 TNDT 位域
  - 重新使能通道(设置 **DMA_CONn** 寄存器的 CHEN 为 1)，以从停止点重新开始传输

10. 5. 11 中断事件

对于每个 DMA 通道在发生传输一半或传输完成时都会触发中断事件。

10. 6 特殊功能寄存器

10. 6. 1 寄存器列表

DMA 寄存器列表			
名称	偏移地址	类型	描述
DMA_IER	0000_H	W1	DMA 中断始能寄存器
DMA_IDR	0004_H	W1	DMA 中断禁用寄存器
DMA_IVS	0008_H	R	DMA 中断有效状态寄存器
DMA_RIF	$000C_H$	R	DMA 原始中断事件标志寄存器
DMA_IFM	0010_H	R	DMA 中断标志位状态寄存器
DMA_ICR	0014_H	C_W1	DMA 中断清除寄存器
DMA_CON0	0020_H	R/W	DMA 通道 0 控制寄存器
DMA_SAR0	0024_H	R/W	DMA 通道 0 来源地址寄存器
DMA_DAR0	0028_H	R/W	DMA 通道 0 目的地地址寄存器
DMA_NDT0	$002C_H$	R/W	DMA 通道 0 数据传输数量寄存器
DMA_CON1	0030_H	R/W	DMA 通道 1 控制寄存器
DMA_SAR1	0034_H	R/W	DMA 通道 1 来源地址寄存器
DMA_DAR1	0038_H	R/W	DMA 通道 1 目的地地址寄存器
DMA_NDT1	$003C_H$	R/W	DMA 通道 1 数据传输数量寄存器
DMA_CON2	0040_H	R/W	DMA 通道 2 控制寄存器
DMA_SAR2	0044_H	R/W	DMA 通道 2 来源地址寄存器
DMA_DAR2	0048_H	R/W	DMA 通道 2 目的地地址寄存器
DMA_NDT2	$004C_H$	R/W	DMA 通道 2 数据传输数量寄存器
DMA_CON3	0050_H	R/W	DMA 通道 3 控制寄存器
DMA_SAR3	0054_H	R/W	DMA 通道 3 来源地址寄存器
DMA_DAR3	0058_H	R/W	DMA 通道 3 目的地地址寄存器
DMA_NDT3	$005C_H$	R/W	DMA 通道 3 数据传输数量寄存器
DMA_CON4	0060_H	R/W	DMA 通道 4 控制寄存器
DMA_SAR4	0064_H	R/W	DMA 通道 4 来源地址寄存器
DMA_DAR4	0068_H	R/W	DMA 通道 4 目的地地址寄存器
DMA_NDT4	$006C_H$	R/W	DMA 通道 4 数据传输数量寄存器
DMA_CON5	0070_H	R/W	DMA 通道 5 控制寄存器
DMA_SAR5	0074_H	R/W	DMA 通道 5 来源地址寄存器
DMA_DAR5	0078_H	R/W	DMA 通道 5 目的地地址寄存器
DMA_NDT5	$007C_H$	R/W	DMA 通道 5 数据传输数量寄存器

10. 6. 2 寄存器描述

10. 6. 2. 1 DMA中断开启寄存器(DMA_IER)

DMA 中断开启寄存器 (DMA_IER)																												
偏移地址：0x000																												
复位值：0x0000 0000																												
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
																				CH5HT	CH5TC	CH4HT	CH4TC	CH3HT	CH3TC	CH2HT	CH2TC	CH1HT	CH1TC	CH0HT	CH0TC

—	Bit 31-12	—	—
CH5HT	Bit 11	W1	**DMA通道5传输一半中断开启** 0：写入0无效 1：开启DMA通道5传输一半中断
CH5TC	Bit 10	W1	**DMA通道5传输完成中断开启** 0：写入0无效 1：开启DMA通道5传输完成中断
CH4HT	Bit 9	W1	**DMA通道4传输一半中断开启** 0：写入0无效 1：开启DMA通道4传输一半中断
CH4TC	Bit 8	W1	**DMA通道4传输完成中断开启** 0：写入0无效 1：开启DMA通道4传输完成中断
CH3HT	Bit 7	W1	**DMA通道3传输一半中断开启** 0：写入0无效 1：开启DMA通道3传输一半中断
CH3TC	Bit 6	W1	**DMA通道3传输完成中断开启** 0：写入0无效 1：开启DMA通道3传输完成中断
CH2HT	Bit 5	W1	**DMA通道2传输一半中断开启** 0：写入0无效 1：开启DMA通道2传输一半中断
CH2TC	Bit 4	W1	**DMA通道2传输完成中断开启** 0：写入0无效 1：开启DMA通道2传输完成中断
CH1HT	Bit 3	W1	**DMA通道1传输一半中断开启** 0：写入0无效

			1：开启DMA通道1传输一半中断
CH1TC	Bit 2	W1	**DMA通道1传输完成中断开启** 0：写入0无效 1：开启DMA通道1传输完成中断
CH0HT	Bit 1	W1	**DMA通道0传输一半中断开启** 0：写入0无效 1：开启DMA通道0传输一半中断
CH0TC	Bit 0	W1	**DMA通道0传输完成中断开启** 0：写入0无效 1：开启DMA通道0传输完成中断

10. 6. 2. 2 DMA中断关闭寄存器 (DMA_IDR)

DMA 中断关闭寄存器(DMA_IDR)																												
偏移地址：0x004																												
复位值：0x0000 0000																												
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
																				CH5HT	CH5TC	CH4HT	CH4TC	CH3HT	CH3TC	CH2HT	CH2TC	CH1HT	CH1TC	CH0HT	CH0TC

—	Bit 31-12	—	—
CH5HT	Bit 11	W1	**DMA通道5传输一半中断关闭** 0：写入0无效 1：关闭DMA通道5传输一半中断
CH5TC	Bit 10	W1	**DMA通道5传输完成中断关闭** 0：写入0无效 1：关闭DMA通道5传输完成中断
CH4HT	Bit 9	W1	**DMA通道4传输一半中断关闭** 0：写入0无效 1：关闭DMA通道4传输一半中断
CH4TC	Bit 8	W1	**DMA通道4传输完成中断关闭** 0：写入0无效 1：关闭DMA通道4传输完成中断
CH3HT	Bit 7	W1	**DMA通道3传输一半中断关闭** 0：写入0无效 1：关闭DMA通道3传输一半中断
CH3TC	Bit 6	W1	**DMA通道3传输完成中断关闭**

			0：写入0无效 1：关闭DMA通道3传输完成中断
CH2HT	Bit 5	W1	**DMA通道2传输一半中断关闭** 0：写入0无效 1：关闭DMA通道2传输一半中断
CH2TC	Bit 4	W1	**DMA通道2传输完成中断关闭** 0：写入0无效 1：关闭DMA通道2传输完成中断
CH1HT	Bit 3	W1	**DMA通道1传输一半中断关闭** 0：写入0无效 1：关闭DMA通道1传输一半中断
CH1TC	Bit 2	W1	**DMA通道1传输完成中断关闭** 0：写入0无效 1：关闭DMA通道1传输完成中断
CH0HT	Bit 1	W1	**DMA通道0传输一半中断关闭** 0：写入0无效 1：关闭DMA通道0传输一半中断
CH0TC	Bit 0	W1	**DMA通道0传输完成中断关闭** 0：写入0无效 1：关闭DMA通道0传输完成中断

10. 6. 2. 3 DMA 中断功能有效状态寄存器 (DMA_IVS)

DMA 中断功能有效状态寄存器(DMA_IVS)

偏移地址：0x008

复位值：0x0000 0000

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
																				CH5HT	CH5TC	CH4HT	CH4TC	CH3HT	CH3TC	CH2HT	CH2TC	CH1HT	CH1TC	CH0HT	CH0TC

—	Bit 31-12	—	—
CH5HT	Bit 11	R	**DMA通道5传输一半中断功能状态** 0：中断功能处于关闭状态 1：中断功能处于开启状态
CH5TC	Bit 10	R	**DMA通道5传输完成中断功能状态** 0：中断功能处于关闭状态 1：中断功能处于开启状态

CH4HT	Bit 9	R	**DMA通道4传输一半中断功能状态** 0：中断功能处于关闭状态 1：中断功能处于开启状态
CH4TC	Bit 8	R	**DMA通道4传输完成中断功能状态** 0：中断功能处于关闭状态 1：中断功能处于开启状态
CH3HT	Bit 7	R	**DMA通道3传输一半中断功能状态** 0：中断功能处于关闭状态 1：中断功能处于开启状态
CH3TC	Bit 6	R	**DMA通道3传输完成中断功能状态** 0：中断功能处于关闭状态 1：中断功能处于开启状态
CH2HT	Bit 5	R	**DMA通道2传输一半中断功能状态** 0：中断功能处于关闭状态 1：中断功能处于开启状态
CH2TC	Bit 4	R	**DMA通道2传输完成中断功能状态** 0：中断功能处于关闭状态 1：中断功能处于开启状态
CH1HT	Bit 3	R	**DMA通道1传输一半中断功能状态** 0：中断功能处于关闭状态 1：中断功能处于开启状态
CH1TC	Bit 2	R	**DMA通道1传输完成中断功能状态** 0：中断功能处于关闭状态 1：中断功能处于开启状态
CH0HT	Bit 1	R	**DMA通道0传输一半中断功能状态** 0：中断功能处于关闭状态 1：中断功能处于开启状态
CH0TC	Bit 0	R	**DMA通道0传输完成中断功能状态** 0：中断功能处于关闭状态 1：中断功能处于开启状态

DMA_IVS 寄存器，是实时反映系统配置 DMA_IER 与 DMA_IDR 的中断功能状态。此寄存器状态是将 DMA_IER 与 DMA_IDR 进行硬件运算，公式如下：

DMA_IVS = DMA_IER & ~DMA_IDR

10. 6. 2. 4 DMA 原始中断状态寄存器 (DMA_RIF)

DMA 原始中断状态寄存器(DMA_RIF)																												
偏移地址：0x00C																												
复位值：0x0000 0000																												
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
																				CH5HT	CH5TC	CH4HT	CH4TC	CH3HT	CH3TC	CH2HT	CH2TC	CH1HT	CH1TC	CH0HT	CH0TC

—	Bit 31-12	—	—
CH5HT	Bit 11	R	**DMA通道5传输一半，原始中断状态** 0：未发生中断事件 1：发生中断事件
CH5TC	Bit 10	R	**DMA通道5传输完成，原始中断状态** 0：未发生中断事件 1：发生中断事件
CH4HT	Bit 9	R	**DMA通道4传输一半，原始中断状态** 0：未发生中断事件 1：发生中断事件
CH4TC	Bit 8	R	**DMA通道4传输完成，原始中断状态** 0：未发生中断事件 1：发生中断事件
CH3HT	Bit 7	R	**DMA通道3传输一半，原始中断状态** 0：未发生中断事件 1：发生中断事件
CH3TC	Bit 6	R	**DMA通道3传输完成，原始中断状态** 0：未发生中断事件 1：发生中断事件
CH2HT	Bit 5	R	**DMA通道2传输一半，原始中断状态** 0：未发生中断事件 1：发生中断事件
CH2TC	Bit 4	R	**DMA通道2传输完成，原始中断状态** 0：未发生中断事件 1：发生中断事件
CH1HT	Bit 3	R	**DMA通道1传输一半，原始中断状态** 0：未发生中断事件 1：发生中断事件

CH1TC	Bit 2	R	**DMA通道1传输完成，原始中断状态** 0：未发生中断事件 1：发生中断事件
CH0HT	Bit 1	R	**DMA通道0传输一半，原始中断状态** 0：未发生中断事件 1：发生中断事件
CH0TC	Bit 0	R	**DMA通道0传输完成，原始中断状态** 0：未发生中断事件 1：发生中断事件

10. 6. 2. 5 DMA 中断标志位状态寄存器 (DMA_IFM)

DMA 中断标志位状态寄存器(DMA_IFM)																												
偏移地址：0x060																												
复位值：0x0000 0000																												
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
																				CH5HT	CH5TC	CH4HT	CH4TC	CH3HT	CH3TC	CH2HT	CH2TC	CH1HT	CH1TC	CH0HT	CH0TC

—	Bit 31-12	—	—
CH5HT	Bit 11	R	**DMA通道5传输一半，中断标志位状态** 0：未发生中断事件或中断未使能 1：产生中断
CH5TC	Bit 10	R	**DMA通道5传输完成，中断标志位状态** 0：未发生中断事件或中断未使能 1：产生中断
CH4HT	Bit 9	R	**DMA通道4传输一半，中断标志位状态** 0：未发生中断事件或中断未使能 1：产生中断
CH4TC	Bit 8	R	**DMA通道4传输完成，中断标志位状态** 0：未发生中断事件或中断未使能 1：产生中断
CH3HT	Bit 7	R	**DMA通道3传输一半，中断标志位状态** 0：未发生中断事件或中断未使能 1：产生中断
CH3TC	Bit 6	R	**DMA通道3传输完成，中断标志位状态** 0：未发生中断事件或中断未使能

			1：产生中断
CH2HT	Bit 5	R	**DMA通道2传输一半，中断标志位状态** 0：未发生中断事件或中断未使能 1：产生中断
CH2TC	Bit 4	R	**DMA通道2传输完成，中断标志位状态** 0：未发生中断事件或中断未使能 1：产生中断
CH1HT	Bit 3	R	**DMA通道1传输一半，中断标志位状态** 0：未发生中断事件或中断未使能 1：产生中断
CH1TC	Bit 2	R	**DMA通道1传输完成，中断标志位状态** 0：未发生中断事件或中断未使能 1：产生中断
CH0HT	Bit 1	R	**DMA通道0传输一半，中断标志位状态** 0：未发生中断事件或中断未使能 1：产生中断
CH0TC	Bit 0	R	**DMA通道0传输完成，中断标志位状态** 0：未发生中断事件或中断未使能 1：产生中断

DMA_IFM 寄存器，是滤除已关闭中断功能的中断事件，只关注开启中断功能的事件，此寄存器状态是将 DMA_RIF 与 DMA_IVS 进行硬件运算，公式如下：

DMA_IFM = DMA_RIF & DMA_IVS

10. 6. 2. 6 DMA 中断清除寄存器 (DMA_ICR)

DMA 中断清除寄存器(DMA_ICR)

偏移地址：0x014

复位值：0x0000 0000

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
																				CH5HT	CH5TC	CH4HT	CH4TC	CH3HT	CH3TC	CH2HT	CH2TC	CH1HT	CH1TC	CH0HT	CH0TC

—	Bit 31-12	—	—
CH5HT	Bit 11	C_W1	**DMA通道5传输一半中断清除** 0：写入0无效 1：清除中断事件与中断

CH5TC	Bit 10	C_W1	**DMA通道5传输完成中断清除** 0：写入0无效 1：清除中断事件与中断
CH4HT	Bit 9	C_W1	**DMA通道4传输一半中断清除** 0：写入0无效 1：清除中断事件与中断
CH4TC	Bit 8	C_W1	**DMA通道4传输完成中断清除** 0：写入0无效 1：清除中断事件与中断
CH3HT	Bit 7	C_W1	**DMA通道3传输一半中断清除** 0：写入0无效 1：清除中断事件与中断
CH3TC	Bit 6	C_W1	**DMA通道3传输完成中断清除** 0：写入0无效 1：清除中断事件与中断
CH2HT	Bit 5	C_W1	**DMA通道2传输一半中断清除** 0：写入0无效 1：清除中断事件与中断
CH2TC	Bit 4	C_W1	**DMA通道2传输完成中断清除** 0：写入0无效 1：清除中断事件与中断
CH1HT	Bit 3	C_W1	**DMA通道1传输一半中断清除** 0：写入0无效 1：清除中断事件与中断
CH1TC	Bit 2	C_W1	**DMA通道1传输完成中断清除** 0：写入0无效 1：清除中断事件与中断
CH0HT	Bit 1	C_W1	**DMA通道0传输一半中断清除** 0：写入0无效 1：清除中断事件与中断
CH0TC	Bit 0	C_W1	**DMA通道0传输完成中断清除** 0：写入0无效 1：清除中断事件与中断

DMA_IC 寄存器设置时，将清除 DMA_RIF 与 DMA_IFM 中断标志状态；此设置不影响中断 DMA_IER、DMA_IDR 与 DMA_IVS 寄存器，只清除标志状态 DMA_RIF 与 DMA_IFM。此寄存器通过硬件清除中断，公式如下：

DMA_RIF = DMA_RIF & ~DMA_ICR

10. 6. 2. 7 DMA通道 0 控制寄存器 (DMA_CON0)

DMA 通道 0 控制寄存器 (DMA_CON0)																												
偏移地址：0x020																												
复位值：0x0000 0000																												
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
												MAX_BURST<3:0>					DDWSEL<1:0>		DINC		SDWSEL<1:0>		SINC			CHPRI<1:0>		M2M	DIR	CIRC	CHEN

—	Bit 31-20	—	—
MAX_BURST	Bit 19-16	R/W	**在控制器重新仲裁前，该位域决定DMA传输次数** 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。 注：只能在CHEN = 0时对此位域执行写操作。
—	Bit 15	—	—
DDWSEL	Bit 14-13	R/W	**目的地传输数据宽度选择** b00：字节 b01：半字 b10：字 b11：保留 注：只能在CHEN = 0时对此位域执行写操作。
DINC	Bit 12	R/W	**目的地转移增量模式** 0：禁止目的地地址增量模式 1：使能目的地地址增量模式 注：只能在CHEN = 0时对此位执行写操作。
—	Bit 11	—	—
SDWSEL	Bit 10-9	R/W	**来源传输数据宽度选择**

			b00：字节 b01：半字 b10：字 b11：保留 注：只能在CHEN = 0时对此位域执行写操作。
SINC	Bit 8	R/W	**来源转移增量模式** 0：禁止来源地址增量模式 1：使能来源地址增量模式 注：只能在CHEN = 0时对此位执行写操作。
—	Bit 7-6	—	—
CHPRI	Bit 5-4	R/W	**通道优先级** b00：低 b01：中 b10：高 b11：最高 注：只能在CHEN = 0时对此位域执行写操作。
M2M	Bit 3	R/W	**存储器到存储器模式** 0：禁止 1：使能 注：只能在CHEN = 0时对此位执行写操作。
DIR	Bit 2	R/W	**数据传输方向** 0：外设到存储器 1：存储器到外设 注：只能在CHEN = 0时对此位执行写操作。
CIRC	Bit 1	R/W	**循环模式** 0：禁止 1：使能 注：在CHEN = 1时只能对此位设置为0。
CHEN	Bit 0	R/W	**通道使能** 0：禁止 1：使能 注：TNDT 为 0 时，不能将此位设置为 1。

10. 6. 2. 8 DMA通道 0 来源地址寄存器 (DMA_SAR0)

DMA 通道 0 来源地址寄存器 (DMA_SAR0)																															
偏移地址：0x024																															
复位值：0x0000 0000																															
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
SAR<31:0>																															

SAR	Bit 31-0	R/W	**来源数据起始地址寄存器** 此字段指示 DMA 的 32 位来源起始地址。 注：只能在 CHEN =0 时对此位域执行写操作。

10. 6. 2. 9 DMA通道 0 目的地地址寄存器 (DMA_DAR0)

DMA 通道 0 目的地地址寄存器 (DMA_DAR0)																															
偏移地址：0x028																															
复位值：0x0000 0000																															
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
DAR<31:0>																															

DAR	Bit 31-0	R/W	**目的地数据起始地址** 此字段指示 DMA 的 32 位目的地起始地址。 注：只能在 CHEN =0 时对此位域执行写操作。

10. 6. 2. 10 DMA通道 0 数据传输数量寄存器 (DMA_NDT0)

DMA 通道 0 数据传输数量寄存器 (DMA_NDT0)																													
偏移地址：0x02C																													
复位值：0x0000 0000																													
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
NRDT<15:0>																TNDT<15:0>														

NRDT	Bit 31-16	R	**剩余数据传输数量** 此字段指示DMA的16位剩余数据传输数量。
TNDT	Bit 15-0	R/W	**总数据传输数量** 此字段指示 DMA 的 16 位总数据传输数量。 注：只能在 CHEN =0 时对此位域执行写操作。

10. 6. 2. 11 DMA通道 1 控制寄存器 (DMA_CON1)

DMA 通道 1 控制寄存器 (DMA_CON1)																													
偏移地址：0x030																													
复位值：0x0000 0000																													
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
—	—	—	—	—	—	—	—	—	—	—	—	MAX_BURST<3:0>				—	DDWSEL<1:0>		DINC	—	SDWSEL<1:0>		SINC	—	—	CHPRI<1:0>		M2M	DIR	CIRC	CHEN

—	Bit 31-20	—	—
MAX_BURST	Bit 19-16	R/W	**在控制器重新仲裁前，该位域决定DMA传输次数** 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。 注：只能在CHEN = 0时对此位域执行写操作。
—	Bit 15	—	—
DDWSEL	Bit 14-13	R/W	**目的地传输数据宽度选择** b00：字节 b01：半字 b10：字 b11：保留 注：只能在CHEN = 0时对此位域执行写操作。
DINC	Bit 12	R/W	**目的地转移增量模式** 0：禁止目的地地址增量模式 1：使能目的地地址增量模式 注：只能在CHEN = 0时对此位执行写操作。
—	Bit 11	—	—
SDWSEL	Bit 10-9	R/W	**来源传输数据宽度选择** b00：字节 b01：半字 b10：字 b11：保留 注：只能在CHEN = 0时对此位域执行写操作。
SINC	Bit 8	R/W	**来源转移增量模式** 0：禁止来源地址增量模式 1：使能来源地址增量模式 注：只能在CHEN = 0时对此位执行写操作。
—	Bit 7-6	—	—
CHPRI	Bit 5-4	R/W	**通道优先级** b00：低 b01：中 b10：高 b11：最高 注：只能在CHEN = 0时对此位域执行写操作。
M2M	Bit 3	R/W	**存储器到存储器模式** 0：禁止 1：使能 注：只能在CHEN = 0时对此位执行写操作。
DIR	Bit 2	R/W	**数据传输方向** 0：外设到存储器 1：存储器到外设

			注：只能在CHEN = 0时对此位执行写操作。
CIRC	Bit 1	R/W	**循环模式** 0：禁止 1：使能 注：在CHEN = 1时只能对此位设置为0。
CHEN	Bit 0	R/W	**通道使能** 0：禁止 1：使能 注：TNDT 为 0 时，不能将此位设置为 1。

10. 6. 2. 12 DMA通道 1 来源地址寄存器 (DMA_SAR1)

DMA 通道 1 来源地址寄存器 (DMA_SAR1)																															
偏移地址：0x034																															
复位值：0x0000 0000																															
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
SAR<31:0>																															

SAR	Bit 31-0	R/W	**来源数据起始地址寄存器** 此字段指示 DMA 的 32 位来源起始地址。 注：只能在 CHEN =0 时对此位域执行写操作。

10. 6. 2. 13 DMA通道 1 目的地地址寄存器 (DMA_DAR1)

DMA 通道 1 目的地地址寄存器 (DMA_DAR1)																															
偏移地址：0x038																															
复位值：0x0000 0000																															
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
DAR<31:0>																															

DAR	Bit 31-0	R/W	**目的地数据起始地址** 此字段指示 DMA 的 32 位目的地起始地址。 注：只能在 CHEN =0 时对此位域执行写操作。

10. 6. 2. 14 DMA通道 1 数据传输数量寄存器 (DMA_NDT1)

DMA 通道 1 数据传输数量寄存器 (DMA_NDT1)																													
偏移地址：0x03C																													
复位值：0x0000 0000																													
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
NRDT<15:0>																TNDT<15:0>														

NRDT	Bit 31-16	R	**剩余数据传输数量** 此字段指示DMA的16位剩余数据传输数量。
TNDT	Bit 15-0	R/W	**总数据传输数量** 此字段指示 DMA 的 16 位总数据传输数量。 注：只能在 CHEN =0 时对此位域执行写操作。

10. 6. 2. 15 DMA通道 2 控制寄存器 (DMA_CON2)

DMA 通道 2 控制寄存器 (DMA_CON2)																													
偏移地址：0x040																													
复位值：0x0000 0000																													
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
—	—	—	—	—	—	—	—	—	—	—	—	MAX_BURST<3:0>				—	DDWSEL<1:0>		DINC	—	SDWSEL<1:0>		SINC	—	—	CHPRI<1:0>		M2M	DIR	CIRC	CHEN

—	Bit 31-20	—	—
MAX_BURST	Bit 19-16	R/W	**在控制器重新仲裁前，该位域决定DMA传输次数** 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。 注：只能在CHEN = 0时对此位域执行写操作。
—	Bit 15	—	—
DDWSEL	Bit 14-13	R/W	**目的地传输数据宽度选择** b00：字节 b01：半字 b10：字 b11：保留 注：只能在CHEN = 0时对此位域执行写操作。
DINC	Bit 12	R/W	**目的地转移增量模式** 0：禁止目的地地址增量模式 1：使能目的地地址增量模式 注：只能在CHEN = 0时对此位执行写操作。
—	Bit 11	—	—
SDWSEL	Bit 10-9	R/W	**来源传输数据宽度选择** b00：字节 b01：半字 b10：字 b11：保留 注：只能在CHEN = 0时对此位域执行写操作。
SINC	Bit 8	R/W	**来源转移增量模式** 0：禁止来源地址增量模式 1：使能来源地址增量模式 注：只能在CHEN = 0时对此位执行写操作。
—	Bit 7-6	—	—
CHPRI	Bit 5-4	R/W	**通道优先级** b00：低 b01：中 b10：高 b11：最高 注：只能在CHEN = 0时对此位域执行写操作。
M2M	Bit 3	R/W	**存储器到存储器模式** 0：禁止 1：使能 注：只能在CHEN = 0时对此位执行写操作。
DIR	Bit 2	R/W	**数据传输方向** 0：外设到存储器 1：存储器到外设

			注：只能在CHEN = 0时对此位执行写操作。
CIRC	Bit 1	R/W	**循环模式** 0：禁止 1：使能 注：在CHEN = 1时只能对此位设置为0。
CHEN	Bit 0	R/W	**通道使能** 0：禁止 1：使能 注：TNDT 为 0 时，不能将此位设置为 1。

10. 6. 2. 16 DMA通道 2 来源地址寄存器 (DMA_SAR2)

DMA 通道 2 来源地址寄存器 (DMA_SAR2)																														
偏移地址：0x044																														
复位值：0x0000 0000																														
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
SAR<31:0>																														

SAR	Bit 31-0	R/W	**来源数据起始地址寄存器** 此字段指示 DMA 的 32 位来源起始地址。 注：只能在 CHEN =0 时对此位域执行写操作。

10. 6. 2. 17 DMA通道 2 目的地地址寄存器 (DMA_DAR2)

DMA 通道 2 目的地地址寄存器 (DMA_DAR2)																														
偏移地址：0x048																														
复位值：0x0000 0000																														
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
DAR<31:0>																														

DAR	Bit 31-0	R/W	**目的地数据起始地址** 此字段指示 DMA 的 32 位目的地起始地址。 注：只能在 CHEN =0 时对此位域执行写操作。

10. 6. 2. 18 DMA通道 2 数据传输数量寄存器 (DMA_NDT2)

DMA 通道 2 数据传输数量寄存器 (DMA_NDT2)																													
偏移地址：0x070																													
复位值：0x0000 0000																													
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
NRDT<15:0>																TNDT<15:0>														

NRDT	Bit 31-16	R	**剩余数据传输数量** 此字段指示DMA的16位剩余数据传输数量。
TNDT	Bit 15-0	R/W	**总数据传输数量** 此字段指示 DMA 的 16 位总数据传输数量。 注：只能在 CHEN =0 时对此位域执行写操作。

10. 6. 2. 19 DMA通道 3 控制寄存器 (DMA_CON3)

DMA 通道 3 控制寄存器 (DMA_CON3)																													
偏移地址：0x050																													
复位值：0x0000 0000																													
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
—	—	—	—	—	—	—	—	—	—	—	—	MAX_BURST<3:0>				—	DDWSEL<1:0>		DINC	—	SDWSEL<1:0>		SINC	—	—	CHPRI<1:0>		M2M	DIR	CIRC	CHEN

—	Bit 31-20	—	—
MAX_BURST	Bit 19-16	R/W	**在控制器重新仲裁前，该位域决定DMA传输次数** 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。 注：只能在CHEN = 0时对此位域执行写操作。
—	Bit 15	—	—
DDWSEL	Bit 14-13	R/W	**目的地传输数据宽度选择** b00：字节 b01：半字 b10：字 b11：保留 注：只能在CHEN = 0时对此位域执行写操作。
DINC	Bit 12	R/W	**目的地转移增量模式** 0：禁止目的地地址增量模式 1：使能目的地地址增量模式 注：只能在CHEN = 0时对此位执行写操作。
—	Bit 11	—	—
SDWSEL	Bit 10-9	R/W	**来源传输数据宽度选择** b00：字节 b01：半字 b10：字 b11：保留 注：只能在CHEN = 0时对此位域执行写操作。
SINC	Bit 8	R/W	**来源转移增量模式** 0：禁止来源地址增量模式 1：使能来源地址增量模式 注：只能在CHEN = 0时对此位执行写操作。
—	Bit 7-6	—	—
CHPRI	Bit 5-4	R/W	**通道优先级** b00：低 b01：中 b10：高 b11：最高 注：只能在CHEN = 0时对此位域执行写操作。
M2M	Bit 3	R/W	**存储器到存储器模式** 0：禁止 1：使能 注：只能在CHEN = 0时对此位执行写操作。
DIR	Bit 2	R/W	**数据传输方向** 0：外设到存储器 1：存储器到外设

			注：只能在CHEN = 0时对此位执行写操作。
CIRC	Bit 1	R/W	**循环模式** 0：禁止 1：使能 注：在CHEN = 1时只能对此位设置为0。
CHEN	Bit 0	R/W	**通道使能** 0：禁止 1：使能 注：TNDT 为 0 时，不能将此位设置为 1。

10. 6. 2. 20 DMA通道 3 来源地址寄存器 (DMA_SAR3)

DMA 通道 3 来源地址寄存器 (DMA_SAR3)																													
偏移地址：0x054																													
复位值：0x0000 0000																													
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
SAR<31:0>																													

SAR	Bit 31-0	R/W	**来源数据起始地址寄存器** 此字段指示 DMA 的 32 位来源起始地址。 注：只能在 CHEN =0 时对此位域执行写操作。

10. 6. 2. 21 DMA通道 3 目的地地址寄存器 (DMA_DAR3)

DMA 通道 3 目的地地址寄存器 (DMA_DAR3)																													
偏移地址：0x058																													
复位值：0x0000 0000																													
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
DAR<31:0>																													

DAR	Bit 31-0	R/W	**目的地数据起始地址** 此字段指示 DMA 的 32 位目的地起始地址。 注：只能在 CHEN =0 时对此位域执行写操作。

10. 6. 2. 22 DMA通道 3 数据传输数量寄存器 (DMA_NDT3)

DMA 通道 3 数据传输数量寄存器 (DMA_NDT3)																													
偏移地址：0x05C																													
复位值：0x0000 0000																													
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
NRDT<15:0>																TNDT<15:0>														

NRDT	Bit 31-16	R	**剩余数据传输数量** 此字段指示DMA的16位剩余数据传输数量。
TNDT	Bit 15-0	R/W	**总数据传输数量** 此字段指示 DMA 的 16 位总数据传输数量。 注：只能在 CHEN =0 时对此位域执行写操作。

10. 6. 2. 23 DMA通道 4 控制寄存器 (DMA_CON4)

DMA 通道 4 控制寄存器 (DMA_CON4)																													
偏移地址：0x060																													
复位值：0x0000 0000																													
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
—	—	—	—	—	—	—	—	—	—	—	—	MAX_BURST<3:0>				—	DDWSEL<1:0>		DINC	—	SDWSEL<1:0>		SINC	—	—	CHPRI<1:0>		M2M	DIR	CIRC	CHEN

—	Bit 31-20	—	—
MAX_BURST	Bit 19-16	R/W	**在控制器重新仲裁前，该位域决定DMA传输次数** 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。 注：只能在CHEN = 0时对此位域执行写操作。
—	Bit 15	—	—
DDWSEL	Bit 14-13	R/W	**目的地传输数据宽度选择** b00：字节 b01：半字 b10：字 b11：保留 注：只能在CHEN = 0时对此位域执行写操作。
DINC	Bit 12	R/W	**目的地转移增量模式** 0：禁止目的地地址增量模式 1：使能目的地地址增量模式 注：只能在CHEN = 0时对此位执行写操作。
—	Bit 11	—	—
SDWSEL	Bit 10-9	R/W	**来源传输数据宽度选择** b00：字节 b01：半字 b10：字 b11：保留 注：只能在CHEN = 0时对此位域执行写操作。
SINC	Bit 8	R/W	**来源转移增量模式** 0：禁止来源地址增量模式 1：使能来源地址增量模式 注：只能在CHEN = 0时对此位执行写操作。
—	Bit 7-6	—	—
CHPRI	Bit 5-4	R/W	**通道优先级** b00：低 b01：中 b10：高 b11：最高 注：只能在CHEN = 0时对此位域执行写操作。
M2M	Bit 3	R/W	**存储器到存储器模式** 0：禁止 1：使能 注：只能在CHEN = 0时对此位执行写操作。
DIR	Bit 2	R/W	**数据传输方向** 0：外设到存储器 1：存储器到外设

			注：只能在CHEN = 0时对此位执行写操作。
CIRC	Bit 1	R/W	**循环模式** 0：禁止 1：使能 注：在CHEN = 1时只能对此位设置为0。
CHEN	Bit 0	R/W	**通道使能** 0：禁止 1：使能 注：TNDT 为 0 时，不能将此位设置为 1。

10. 6. 2. 24 DMA通道 4 来源地址寄存器 (DMA_SAR4)

DMA 通道 4 来源地址寄存器 (DMA_SAR4)																														
偏移地址：0x064																														
复位值：0x0000 0000																														
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
SAR<31:0>																														

SAR	Bit 31-0	R/W	**来源数据起始地址寄存器** 此字段指示 DMA 的 32 位来源起始地址。 注：只能在 CHEN =0 时对此位域执行写操作。

10. 6. 2. 25 DMA通道 4 目的地地址寄存器 (DMA_DAR4)

DMA 通道 4 目的地地址寄存器 (DMA_DAR4)																														
偏移地址：0x068																														
复位值：0x0000 0000																														
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
DAR<31:0>																														

DAR	Bit 31-0	R/W	**目的地数据起始地址** 此字段指示 DMA 的 32 位目的地起始地址。 注：只能在 CHEN =0 时对此位域执行写操作。

10. 6. 2. 26 DMA通道 4 数据传输数量寄存器 (DMA_NDT4)

DMA 通道 4 数据传输数量寄存器 (DMA_NDT4)																													
偏移地址：0x06C																													
复位值：0x0000 0000																													
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
NRDT<15:0>																TNDT<15:0>														

NRDT	Bit 31-16	R	**剩余数据传输数量** 此字段指示DMA的16位剩余数据传输数量。
TNDT	Bit 15-0	R/W	**总数据传输数量** 此字段指示 DMA 的 16 位总数据传输数量。 注：只能在 CHEN =0 时对此位域执行写操作。

10. 6. 2. 27 DMA通道 5 控制寄存器 (DMA_CON5)

DMA 通道 5 控制寄存器 (DMA_CON5)																													
偏移地址：0x070																													
复位值：0x0000 0000																													
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
—	—	—	—	—	—	—	—	—	—	—	—	MAX_BURST<3:0>				—	DDWSEL<1:0>		DINC	—	SDWSEL<1:0>		SINC	—	—	CHPRI<1:0>		M2M	DIR	CIRC	CHEN

—	Bit 31-20	—	—
MAX_BURST	Bit 19-16	R/W	**在控制器重新仲裁前，该位域决定DMA传输次数** 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。 注：只能在CHEN = 0时对此位域执行写操作。
—	Bit 15	—	—
DDWSEL	Bit 14-13	R/W	**目的地传输数据宽度选择** b00：字节 b01：半字 b10：字 b11：保留 注：只能在CHEN = 0时对此位域执行写操作。
DINC	Bit 12	R/W	**目的地转移增量模式** 0：禁止目的地地址增量模式 1：使能目的地地址增量模式 注：只能在CHEN = 0时对此位执行写操作。
—	Bit 11	—	—
SDWSEL	Bit 10-9	R/W	**来源传输数据宽度选择** b00：字节 b01：半字 b10：字 b11：保留 注：只能在CHEN = 0时对此位域执行写操作。
SINC	Bit 8	R/W	**来源转移增量模式** 0：禁止来源地址增量模式 1：使能来源地址增量模式 注：只能在CHEN = 0时对此位执行写操作。
—	Bit 7-6	—	—
CHPRI	Bit 5-4	R/W	**通道优先级** b00：低 b01：中 b10：高 b11：最高 注：只能在CHEN = 0时对此位域执行写操作。
M2M	Bit 3	R/W	**存储器到存储器模式** 0：禁止 1：使能 注：只能在CHEN = 0时对此位执行写操作。
DIR	Bit 2	R/W	**数据传输方向** 0：外设到存储器 1：存储器到外设

			注：只能在CHEN = 0时对此位执行写操作。
CIRC	Bit 1	R/W	**循环模式** 0：禁止 1：使能 注：在CHEN = 1时只能对此位设置为0。
CHEN	Bit 0	R/W	**通道使能** 0：禁止 1：使能 注：TNDT 为 0 时，不能将此位设置为 1。

10. 6. 2. 28 DMA通道 5 来源地址寄存器 (DMA_SAR5)

DMA 通道 5 来源地址寄存器 (DMA_SAR5)																														
偏移地址：0x074																														
复位值：0x0000 0000																														
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
SAR<31:0>																														

SAR	Bit 31-0	R/W	**来源数据起始地址寄存器** 此字段指示 DMA 的 32 位来源起始地址。 注：只能在 CHEN =0 时对此位域执行写操作。

10. 6. 2. 29 DMA通道 5 目的地地址寄存器 (DMA_DAR5)

DMA 通道 5 目的地地址寄存器 (DMA_DAR5)																														
偏移地址：0x078																														
复位值：0x0000 0000																														
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
DAR<31:0>																														

DAR	Bit 31-0	R/W	**目的地数据起始地址** 此字段指示 DMA 的 32 位目的地起始地址。 注：只能在 CHEN =0 时对此位域执行写操作。

10. 6. 2. 30 DMA通道 5 数据传输数量寄存器 (DMA_NDT5)

DMA 通道 5 数据传输数量寄存器 (DMA_NDT5)																															
偏移地址：0x07C																															
复位值：0x0000 0000																															
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
NRDT<15:0>																TNDT<15:0>															

NRDT	Bit 31-16	R	**剩余数据传输数量** 此字段指示DMA的16位剩余数据传输数量。
TNDT	Bit 15-0	R/W	**总数据传输数量** 此字段指示 DMA 的 16 位总数据传输数量。 注：只能在 CHEN =0 时对此位域执行写操作。

第11章 DMA 请求复用器(DMAMUX)

11. 1 概述

外设通过设置其 DMA 请求信号来触发 DMA 传输请求。 DMA 控制器会处理 DMA 请求并生成 DMA 确认信号，而且相应的 DMA 请求信号也将变为无效，但在此之前 DMA 请求一直处于挂起状态。

DMAMUX 请求复用器可在产品的外设和 DMA 控制器之间重新配置 DMA 请求线。该功能通过可编程的通道复用选择寄存器(DMA_CH0_SELCON-DMA_CH5_SELCON)来确保实现。每条通道可不受限制地选择一个 DMA 请求线。

11. 2 特性

可编程的 6 通道 DMA 请求线复用输出

11. 3 功能描述

11. 3. 1 通道选择配置

通道选择模块可以选择要连接到每个 DMA 通道的外设请求线路。该配置是由软件通过控制寄存器 DMA_CH0_SELCON-DMA_CH5_SELCON 完成的。这些控制寄存器都是由 MSIGSEL 选择外设信号。

注：当所选外设的时钟关闭时，DMA 通道不应该被使能。

请求编号	外设	请求编号	外设	请求编号	外设	请求编号	外设
0	UART1_TX	**32**	BS16T1_UP	**64**	GP16C2T1_CH1	**96**	
1		**33**	AD16C6T1_CH1	**65**	GP16C2T1_CH2	**97**	
2	UART2_TX	**34**	AD16C6T1_CH2	**66**	GP16C2T1_UP	**98**	
3		**35**	AD16C6T1_CH3	**67**	GP16C2T1_TRIG	**99**	
4		**36**	AD16C6T1_CH4	**68**	GP16C2T1_COM	**100**	
5	SPI1_TX	**37**	AD16C6T1_UP	**69**		**101**	
6		**38**	AD16C6T1_TRIG	**70**	GP16C2T2_CH1	**102**	
7	I2C1_TX	**39**	AD16C6T1_COM	**71**	GP16C2T2_CH2	**103**	
8		**40**	AD16C6T2_CH1	**72**	GP16C2T2_UP	**104**	
9		**41**	AD16C6T2_CH2	**73**	GP16C2T2_TRIG	**105**	
10		**42**	AD16C6T2_CH3	**74**	GP16C2T2_COM	**106**	
11		**43**	AD16C6T2_CH4	**75**		**107**	
12		**44**	AD16C6T2_UP	**76**		**108**	
13		**45**	AD16C6T2_TRIG	**77**		**109**	
14		**46**	AD16C6T2_COM	**78**		**110**	
15	UART1_RX	**47**	GP32C4T1_CH1	**79**		**111**	

16	UART2_RX	**48**	GP32C4T1_CH2	**80**		**112**	
17		**49**	GP32C4T1_CH3	**81**		**113**	
18		**50**	GP32C4T1_CH4	**82**		**114**	
19		**51**	GP32C4T1_UP	**83**		**115**	
20	SPI1_RX	**52**	GP32C4T1_TRIG	**84**		**116**	
21		**53**	GP32C4T2_CH1	**85**		**117**	
22	I2C1_RX	**54**	GP32C4T2_CH2	**86**		**118**	
23		**55**	GP32C4T2_CH3	**87**		**119**	
24		**56**	GP32C4T2_CH4	**88**		**120**	
25	ADC1	**57**	GP32C4T2_UP	**89**		**121**	
26	ADC2	**58**	GP32C4T2_TRIG	**90**		**122**	
27		**59**		**91**		**123**	
28	SVA	**60**		**92**		**124**	
29		**61**		**93**		**125**	
30		**62**		**94**		**126**	
31		**63**		**95**		**127**	

表 11-1　外设请求对应表

11. 4 特殊功能寄存器

11. 4. 1 寄存器列表

DMAMUX 寄存器列表			
名称	偏移地址	类型	描述
DMA_CH0_SELCON	0000_H	R/W	DMA 通道 0 復用選擇寄存器
DMA_CH1_SELCON	0004_H	R/W	DMA 通道 1 復用選擇寄存器
DMA_CH2_SELCON	0008_H	R/W	DMA 通道 2 復用選擇寄存器
DMA_CH3_SELCON	$000C_H$	R/W	DMA 通道 3 復用選擇寄存器
DMA_CH4_SELCON	0010_H	R/W	DMA 通道 4 復用選擇寄存器
DMA_CH5_SELCON	0014_H	R/W	DMA 通道 5 復用選擇寄存器

11. 4. 2 寄存器描述

11. 4. 2. 1 通道 0 复用选择寄存器 (DMA_CH0_SELCON)

通道 0 复用选择寄存器 (DMA_CH0_SELCON)																												
偏移地址:0x00																												
复位值:0x0000 0000																												
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																									MSIGSEL<6:0>						

—	Bits 31-7	—	—
MSIGSEL	Bits 6-0	R/W	复用选择

11. 4. 2. 2 通道 1 复用选择寄存器 (DMA_CH1_SELCON)

通道 1 复用选择寄存器 (DMA_CH1_SELCON)																												
偏移地址:0x04																												
复位值:0x0000 0000																												
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																									MSIGSEL<6:0>						

—	Bits 31-7	—	—
MSIGSEL	Bits6-0	R/W	复用选择

11. 4. 2. 3 通道 2 复用选择寄存器 (DMA_CH2_SELCON)

通道 2 复用选择寄存器 (DMA_CH2_SELCON)																												
偏移地址:0x08																												
复位值:0x0000 0000																												
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																									MSIGSEL<6:0>						

—	Bits 31-7	—	—
MSIGSEL	Bits6-0	R/W	复用选择

11. 4. 2. 4 通道 3 复用选择寄存器 (DMA_CH3_SELCON)

通道 3 复用选择寄存器 (DMA_CH3_SELCON)																												
偏移地址:0x0C																												
复位值:0x0000 0000																												
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																									MSIGSEL<6:0>						

—	Bits 31-7	—	—
MSIGSEL	Bits6-0	R/W	复用选择

11. 4. 2. 5 通道 4 复用选择寄存器 (DMA_CH4_SELCON)

通道 4 复用选择寄存器 (DMA_CH4_SELCON)																												
偏移地址:0x010																												
复位值:0x0000 0000																												
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																									MSIGSEL<6:0>						

—	Bits 31-7	—	—
MSIGSEL	Bits6-0	R/W	复用选择

11. 4. 2. 6 通道 5 复用选择寄存器 (DMA_CH5_SELCON)

通道 5 复用选择寄存器 (DMA_CH5_SELCON)																												
偏移地址:0x014																												
复位值:0x0000 0000																												
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
																									MSIGSEL<6:0>						

—	Bits 31-7	—	—
MSIGSEL	Bits6-0	R/W	复用选择

第12章 外部中断 (EXTI)

12. 1 概述

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

EXTI 允许管理多达 20 个外部/内部事件通道，可以在停止模式唤醒设备。

有些通道是可配置输入的：在这种情况下，可以独立地选择触发边沿，并且通过状态标志位来标示中断的来源。这些可配置的通道是由 I/Os 外部中断和少数外围设备使用。有些通道是直接输入的：它们被一些外围设备通过停止事件或者中断来产生唤醒信号。在这种情况下，状态标志位由外围设备提供。

作为外部或内部事件请求的每一个通道都可独立配置。EXTI 控制器还允许通过软件配置专用寄存器，来模拟相应的硬件事件或中断。

12. 2 特性

- 支持产生多达 20 个事件/中断请求。
- 每个事件/中断通道都有独立的屏蔽。
- 可选择上升沿触发或下降沿触发。
- 每个外部中断通道都有专用的状态位。
- 可软件模拟所有外部事件请求。

12. 3 结构图

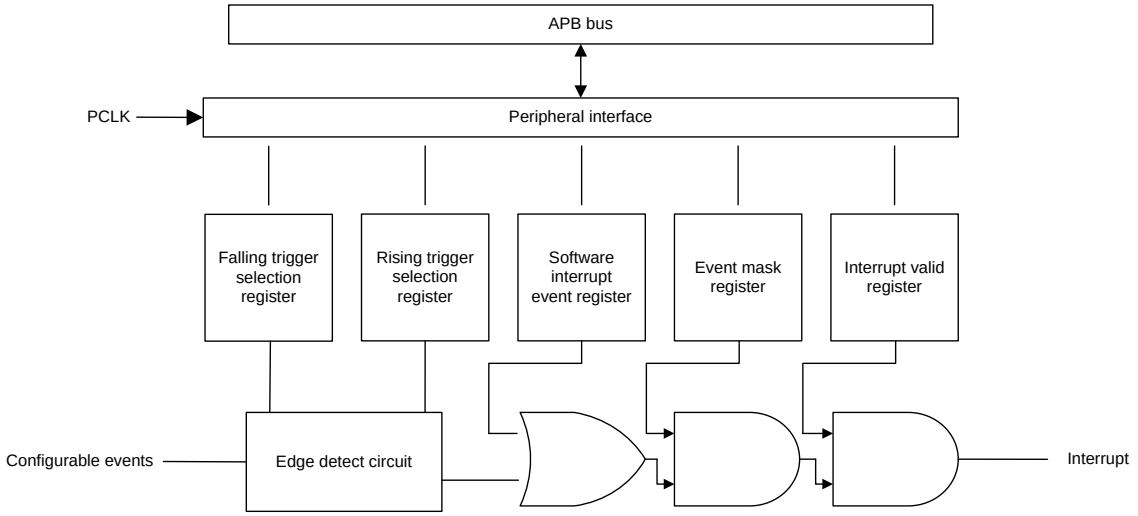


图 12-1　外部中断/事件框图

12. 4 功能描述

12. 4. 1 硬件中断选择

硬件发生中断所需要的步骤为下:

1. 设定中断的 **EXTI_IER** 以及 **EXTI_IDR**，来开启或是关闭中断的来源。
2. 选择中断的触发方式，配置 **EXTI_RTS** 或 **EXTI_FTS** 寄存器。
3. 用户可以由 **EXTI_IVS** 来判断中断开启的状态，中断发生后，可以由 **EXTI_IFM** 来观察中断是否发生。
4. 当中断产生后，配置 **EXTI_ICR**，用户便可清除中断。

12. 4. 2 软件中断选择

用户可使用软件在某些时间点发生中断，步骤如下:

1. 设定中断的 **EXTI_IER** 以及 **EXTI_IDR**，来开启或是关闭中断。
2. 用户可以根据需求设定 **EXTI_SWI**，让 EXTI 在某些时间点发出中断。

12. 4. 3 外部和内部中断/事件通道映射

因为 GPIO 有 A/B/C 组，每一组最多会有 16 bit 的讯号，3 组 GPIO 共享 EXTI 的中断，使用者必须事先设定要使用哪个 GPIO 来产生中断。根据 **EXTI_ICFG1** 以及 **EXTI_ICFG2** 两个寄存器，可以选择该中断是由哪组的 GPIO 产生。

EXTI 通道	通道来源	通道型态
0 - 15	GPIO	可配置
16	CMP1 输出	可配置
17	CMP2 输出	可配置
18 - 19	保留	保留
20	LVD	可配置
21	WAKEUP	可配置
22 - 31	保留	保留

表 12-1 EXTI 通道连线

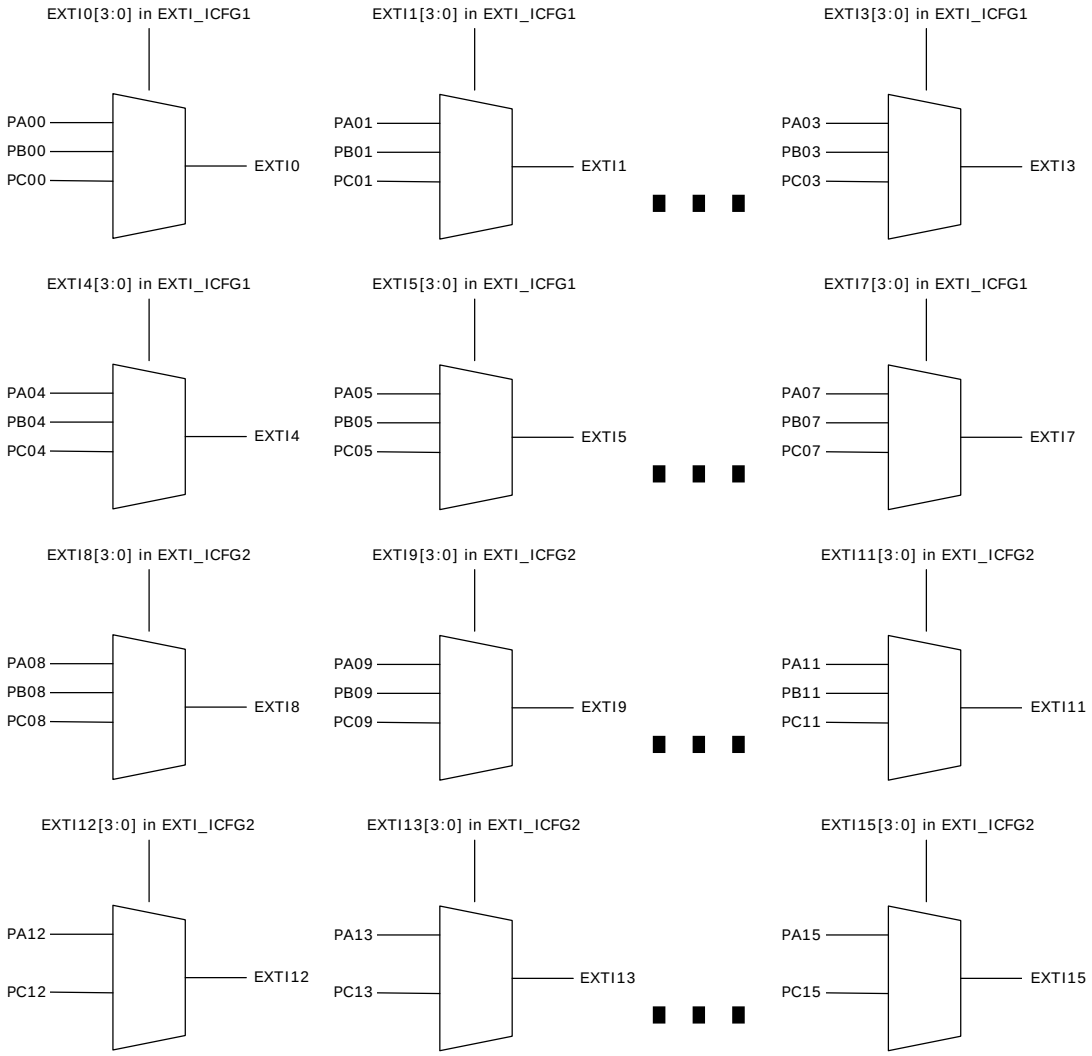


图 12-2　外部中断/事件 GPIO 映射

12. 5 特殊功能寄存器

12. 5. 1 寄存器列表

EXTI 寄存器列表			
名称	偏移地址	类型	描述
EXTI_IER	0000_H	W1	EXTI 中断开启寄存器
EXTI_IDR	0004_H	W1	EXTI 中断关闭寄存器
EXTI_IVS	0008_H	R	EXTI 中断功能有效状态寄存器
EXTI_RIF	$000C_H$	R	EXTI 原始中断状态寄存器
EXTI_IFM	0010_H	R	EXTI 中断标志位状态寄存器
EXTI_ICR	0014_H	C_W1	EXTI 中断清除寄存器
EXTI_RTS	0018_H	R/W	EXTI 上升沿触发选择寄存器
EXTI_FTS	$001C_H$	R/W	EXTI 下降沿触发选择寄存器
EXTI_SWI	0020_H	R/W	EXTI 软件中断事件寄存器
EXTI_ADTE1	0024_H	R/W	EXTI ADC 触发启用寄存器 1
EXTI_ADTE2	0028_H	R/W	EXTI ADC 触发启用寄存器 2
EXTI_DB	$002C_H$	R/W	EXTI 弹跳消除寄存器
EXTI_DBC	0030_H	R/W	EXTI 弹跳消除取样率控制寄存器
EXTI_ICFG1	0034_H	R/W	EXTI 中断配置寄存器 1
EXTI_ICFG2	0038_H	R/W	EXTI 中断配置寄存器 2

12. 5. 2 寄存器描述

寄存器控制的说明中，提到 EXTI 通道的对应位，可以参考表 12-1 EXTI 通道连线。

12. 5. 2. 1 EXTI中断开启寄存器 (EXTI_IER)

EXTI 中断开启寄存器(EXTI_IER)																													
偏移地址:0x00																													
复位值:0x0000 0000																													
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
—	—	—	—	—	—	—	—	—	—	IE21	IE20	—	—	IE17	IE16	IE15	IE14	IE13	IE12	IE11	IE10	IE9	IE8	IE7	IE6	IE5	IE4	IE3	IE2	IE1	IE0

—	Bits 31-22	—	—
IE20	Bit 21	W1	**启用EXTI通道y中断。(y=20至21)。** 0: 无影响。 1: 开启EXTI通道y的中断。
IE21	Bit 20	W1	
—	Bits 19-18	—	—
IE17	Bit 17	W1	**启用EXTI通道y中断。(y=0至17)。** 0: 无影响。 1: 开启 EXTI 通道 y 的中断。
IE16	Bit 16	W1	
IE15	Bit 15	W1	
IE14	Bit 14	W1	
IE13	Bit 13	W1	
IE12	Bit 12	W1	
IE11	Bit 11	W1	
IE10	Bit 10	W1	
IE9	Bit 9	W1	
IE8	Bit 8	W1	
IE7	Bit 7	W1	
IE6	Bit 6	W1	
IE5	Bit 5	W1	
IE4	Bit 4	W1	
IE3	Bit 3	W1	
IE2	Bit 2	W1	
IE1	Bit 1	W1	
IE0	Bit 0	W1	

12. 5. 2. 2 EXTI中断关闭寄存器 (EXTI_IDR)

EXTI 中断关闭寄存器(EXTI_IDR)																														
偏移地址:0x04																														
复位值:0x0000 0000																														
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
										ID20	ID21			ID17	ID16	ID15	ID14	ID13	ID12	ID11	ID10	ID9	ID8	ID7	ID6	ID5	ID4	ID3	ID2	ID1	ID0

—	Bits 31-22	—	—
ID20	Bit 21	W1	**禁止EXTI通道y中断。(y=20至21)。** 0: 无影响。 1: 禁止EXTI通道y的中断。
ID21	Bit 20	W1	
—	Bits 19-18	—	—
ID17	Bit 17	W1	**禁止EXTI通道y中断。(y=20至21)。** 0: 无影响。 1: 禁止 EXTI 通道 y 的中断。
ID16	Bit 16	W1	
ID15	Bit 15	W1	
ID14	Bit 14	W1	
ID13	Bit 13	W1	
ID12	Bit 12	W1	
ID11	Bit 11	W1	
ID10	Bit 10	W1	
ID9	Bit 9	W1	
ID8	Bit 8	W1	
ID7	Bit 7	W1	
ID6	Bit 6	W1	
ID5	Bit 5	W1	
ID4	Bit 4	W1	
ID3	Bit 3	W1	
ID2	Bit 2	W1	
ID1	Bit 1	W1	
ID0	Bit 0	W1	

12. 5. 2. 3 EXTI中断功能有效状态寄存器 (EXTI_IVS)

EXTI 中断功能有效状态寄存器(EXTI_IVS)																														
偏移地址:0x08																														
复位值:0x0000 0000																														
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
										IV20	IV21			IV17	IV16	IV15	IV14	IV13	IV12	IV11	IV10	IV9	IV8	IV7	IV6	IV5	IV4	IV3	IV2	IV1	IV0

—	Bits 31-22	—	—
IV20	Bit 21	R	**EXTI通道y中断始能状态。(y=20至21)。** 0: EXTI通道y中断禁止状态。 1: EXTI通道y中断始能状态。
IV21	Bit 20	R	
—	Bits 19-18	—	—
IV17	Bit 17	R	**EXTI通道y中断始能状态。(y=0至17)。** 0: EXTI通道y中断禁止状态。 1: EXTI 通道 y 中断始能状态。
IV16	Bit 16	R	
IV15	Bit 15	R	
IV14	Bit 14	R	
IV13	Bit 13	R	
IV12	Bit 12	R	
IV11	Bit 11	R	
IV10	Bit 10	R	
IV9	Bit 9	R	
IV8	Bit 8	R	
IV7	Bit 7	R	
IV6	Bit 6	R	
IV5	Bit 5	R	
IV4	Bit 4	R	
IV3	Bit 3	R	
IV2	Bit 2	R	
IV1	Bit 1	R	
IV0	Bit 0	R	

12. 5. 2. 4 EXTI原始中断状态寄存器 (EXTI_RIF)

EXTI 原始中断状态寄存器(EXTI_RIF)																														
偏移地址:0x0C																														
复位值:0x0000 0000																														
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
										RF20	RF 21			RF17	RF16	RF15	RF14	RF13	RF12	RF11	RF10	RF9	RF8	RF7	RF6	RF5	RF4	RF3	RF2	IRF1	RF0

—	Bits 31-22	—	—
RF20	Bit 21	R	**EXTI通道y原始中断状态。(y=20至21)。** 0: 未产生中断。 1: EXTI通道y中断产生。
RF21	Bit 20	R	
—	Bits 19-18	—	—
RF17	Bit 17	R	**EXTI通道y原始中断状态。(y=0至17)。** 0: 未产生中断。 1: EXTI 通道 y 中断产生。
RF16	Bit 16	R	
RF15	Bit 15	R	
RF14	Bit 14	R	
RF13	Bit 13	R	
RF12	Bit 12	R	
RF11	Bit 11	R	
RF10	Bit 10	R	
RF9	Bit 9	R	
RF8	Bit 8	R	
RF7	Bit 7	R	
RF6	Bit 6	R	
RF5	Bit 5	R	
RF4	Bit 4	R	
RF3	Bit 3	R	
RF2	Bit 2	R	
RF1	Bit 1	R	
RF0	Bit 0	R	

12. 5. 2. 5 EXTI中断标志位状态寄存器 (EXTI_IFM)

EXTI 中断标志位状态寄存器(EXTI_IFM)																														
偏移地址:0x10																														
复位值:0x0000 0000																														
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
										IM20	IM21			IM17	IM16	IM15	IM14	IM13	IM12	IM11	IM10	IM9	IM8	IM7	IM6	IM5	IM4	IM3	IM2	IM1	IM0

—	Bits 31-22	—	—
IM20	Bit 21	R	**EXTI通道y中断标志位状态。(y=20至21)。** 0: 未产生中断或中断未始能。 1: EXTI通道y中断产生。
IM21	Bit 20	R	
—	Bits 19-18	—	—
IM17	Bit 17	R	**EXTI通道y中断标志位状态。(y=0至17)。** 0: 未产生中断或中断未始能。 1: EXTI 通道 y 中断产生。
IM16	Bit 16	R	
IM15	Bit 15	R	
IM14	Bit 14	R	
IM13	Bit 13	R	
IM12	Bit 12	R	
IM11	Bit 11	R	
IM10	Bit 10	R	
IM9	Bit 9	R	
IM8	Bit 8	R	
IM7	Bit 7	R	
IM6	Bit 6	R	
IM5	Bit 5	R	
IM4	Bit 4	R	
IM3	Bit 3	R	
IM2	Bit 2	R	
IM1	Bit 1	R	
IM0	Bit 0	R	

12. 5. 2. 6 EXTI中断清除寄存器 (EXTI_ICR)

EXTI 中断清除寄存器(EXTI_ICR)																														
偏移地址:0x14																														
复位值:0x0000 0000																														
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
—	—	—	—	—	—	—	—	—	—	IC20	IC21	—	—	IC17	IC16	IC15	IC14	IC13	IC12	IC11	IC10	IC9	IC8	IC7	IC6	IC5	IC4	IC3	IC2	IC1	IC0

—	Bits 31-22	—	—
IC20	Bit 21	W1	**清除EXTI通道y中断。(y=20至21)。** 0: 无影响。 1: 清除EXTI通道y的中断。
IC21	Bit 20	W1	
—	Bits 19-18	—	—
IC17	Bit 17	W1	**清除EXTI通道y中断。(y=0至17)。** 0: 无影响。 1: 清除 EXTI 通道 y 的中断。
IC16	Bit 16	W1	
IC15	Bit 15	W1	
IC14	Bit 14	W1	
IC13	Bit 13	W1	
IC12	Bit 12	W1	
IC11	Bit 11	W1	
IC10	Bit 10	W1	
IC9	Bit 9	W1	
IC8	Bit 8	W1	
IC7	Bit 7	W1	
IC6	Bit 6	W1	
IC5	Bit 5	W1	
IC4	Bit 4	W1	
IC3	Bit 3	W1	
IC2	Bit 2	W1	
IC1	Bit 1	W1	
IC0	Bit 0	W1	

12. 5. 2. 7 EXTI上升沿触发选择寄存器 (EXTI_RTS)

EXTI 上升沿触发选择寄存器(EXTI_RTS)																														
偏移地址:0x18																														
复位值:0x0000 0000																														
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
										RT21	RT20			RT17	RT16	RT15	RT14	RT13	RT12	RT11	RT10	RT9	RT8	RT7	RT6	RT5	RT4	RT3	RT2	RT1	RT0

—	Bits 31-22	—	—
RT21	Bit 21	R/W	**EXTI通道y上升沿触发配置。(y=20,21)** 0:禁止EXTI通道y上升沿触发中断。 1:始能EXTI通道y上升沿触发中断。
RT20	Bit 20	R/W	
—	Bits 19-18	—	—
RT17	Bit 17	R/W	**EXTI通道y上升沿触发配置。(y=0,17)** 0:禁止EXTI通道y上升沿触发中断。 1:始能EXTI通道y上升沿触发中断。
RT16	Bit 16	R/W	
RT15	Bit 15	R/W	
RT14	Bit 14	R/W	
RT13	Bit 13	R/W	
RT12	Bit 12	R/W	
RT11	Bit 11	R/W	
RT10	Bit 10	R/W	
RT9	Bit 9	R/W	
RT8	Bit 8	R/W	
RT7	Bit 7	R/W	
RT6	Bit 6	R/W	
RT5	Bit 5	R/W	
RT4	Bit 4	R/W	
RT3	Bit 3	R/W	
RT2	Bit 2	R/W	
RT1	Bit 1	R/W	
RT0	Bit 0	R/W	

12. 5. 2. 8 EXTI下降沿触发选择寄存器 (EXTI_FTS)

EXTI 下降沿触发选择寄存器(EXTI_FTS)																														
偏移地址:0x1C																														
复位值:0x0000 0000																														
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
										FT21	FT20			FT17	FT16	FT15	FT14	FT13	FT12	FT11	FT10	FT9	FT8	FT7	FT6	FT5	FT4	FT3	FT2	FT1	FT0

—	Bits 31-22	—	—
FT21	Bit 21	R/W	**EXTI通道y下降沿触发配置。(y=20,21)** 0:禁止EXTI通道y下降沿触发中断。 1:始能EXTI通道y下降沿触发中断。
FT20	Bit 20	R/W	
—	Bits 19-18	—	—
FT17	Bit 17	R/W	**EXTI通道y下降沿触发配置。(y=0,17)** 0:禁止EXTI通道y下降沿触发中断。 1:始能EXTI通道y下降沿触发中断。
FT16	Bit 16	R/W	
FT15	Bit 15	R/W	
FT14	Bit 14	R/W	
FT13	Bit 13	R/W	
FT12	Bit 12	R/W	
FT11	Bit 11	R/W	
FT10	Bit 10	R/W	
FT9	Bit 9	R/W	
FT8	Bit 8	R/W	
FT7	Bit 7	R/W	
FT6	Bit 6	R/W	
FT5	Bit 5	R/W	
FT4	Bit 4	R/W	
FT3	Bit 3	R/W	
FT2	Bit 2	R/W	
FT1	Bit 1	R/W	
FT0	Bit 0	R/W	

12. 5. 2. 9 EXTI软件中断事件寄存器 (EXTI_SWI)

EXTI 软件中断事件寄存器(EXTI_SWI)																														
偏移地址:0x20																														
复位值:0x0000 0000																														
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
										SW21	SW20			SW17	SW16	SW15	SW14	SW13	SW12	SW11	SW10	SW9	SW8	SW7	SW6	SW5	SW4	SW3	SW2	SW1	SW0

—	Bits 31-22	—	—
SW21	Bit 21	R/W	**EXTI通道y的软件中断。(y=20,21)** 0:无影响。 1:EXTI通道y产生软件中断。
SW20	Bit 20	R/W	
—	Bits 19-18	—	—
SW17	Bit 17	R/W	**EXTI通道y的软件中断。(y=0,17)** 0:无影响。 1:EXTI通道y产生软件中断。
SW16	Bit 16	R/W	
SW15	Bit 15	R/W	
SW14	Bit 14	R/W	
SW13	Bit 13	R/W	
SW12	Bit 12	R/W	
SW11	Bit 11	R/W	
SW10	Bit 10	R/W	
SW9	Bit 9	R/W	
SW8	Bit 8	R/W	
SW7	Bit 7	R/W	
SW6	Bit 6	R/W	
SW5	Bit 5	R/W	
SW4	Bit 4	R/W	
SW3	Bit 3	R/W	
SW2	Bit 2	R/W	
SW1	Bit 1	R/W	
SW0	Bit 0	R/W	

12. 5. 2. 10 EXTI ADC触发启用寄存器 1 (EXTI_ADTE1)

EXTIADC 触发启用寄存器 1 (EXTI_ADTE1)																														
偏移地址:0x24																														
复位值:0x0000 0000																														
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
														ADRT17	ADRT16	ADRT15	ADRT14	ADRT13	ADRT12	ADRT11	ADRT10	ADRT9	ADRT8	ADRT7	ADRT6	ADRT5	ADRT4	ADRT3	ADRT2	ADRT1	ADRT0

—	Bits 31-18	—	—
ADRT17	Bit 17	R/W	**配置EXTI通道y中断为ADC标准触发来源。(y=0…17)。** 0:禁止EXTI通道y中断为ADC标准触发来源。 1:始能EXTI通道y中断为ADC标准触发来源。
ADRT16	Bit 16	R/W	
ADRT15	Bit 15	R/W	
ADRT14	Bit 14	R/W	
ADRT13	Bit 13	R/W	
ADRT12	Bit 12	R/W	
ADRT11	Bit 11	R/W	
ADRT10	Bit 10	R/W	
ADRT9	Bit 9	R/W	
ADRT8	Bit 8	R/W	
ADRT7	Bit 7	R/W	
ADRT6	Bit 6	R/W	
ADRT5	Bit 5	R/W	
ADRT4	Bit 4	R/W	
ADRT3	Bit 3	R/W	
ADRT2	Bit 2	R/W	
ADRT1	Bit 1	R/W	
ADRT0	Bit 0	R/W	

12. 5. 2. 11 EXTI ADC触发启用寄存器 2 (EXTI_ADTE2)

EXTIADC 触发启用寄存器 2 (EXTI_ADTE2)																														
偏移地址:0x28																														
复位值:0x0000 0000																														
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
														ADJT17	ADJT16	ADJT15	ADJT14	ADJT13	ADJT12	ADJT11	ADJT10	ADJT9	ADJT8	ADJT7	ADJT6	ADJT5	ADJT4	ADJT3	ADJT2	ADJT1	ADJT0

—	Bits 31-18	—	—
ADJT17	Bit 17	R/W	**配置EXTI通道y中断为ADC插入触发来源。(y=0…17)。** 0:禁止EXTI通道y中断为ADC插入触发来源。 1:始能EXTI通道y中断为ADC插入触发来源。
ADJT16	Bit 16	R/W	
ADJT15	Bit 15	R/W	
ADJT14	Bit 14	R/W	
ADJT13	Bit 13	R/W	
ADJT12	Bit 12	R/W	
ADJT11	Bit 11	R/W	
ADJT10	Bit 10	R/W	
ADJT9	Bit 9	R/W	
ADJT8	Bit 8	R/W	
ADJT7	Bit 7	R/W	
ADJT6	Bit 6	R/W	
ADJT5	Bit 5	R/W	
ADJT4	Bit 4	R/W	
ADJT3	Bit 3	R/W	
ADJT2	Bit 2	R/W	
ADJT1	Bit 1	R/W	
ADJT0	Bit 0	R/W	

12. 5. 2. 12 EXTI弹跳消除寄存器 (EXTI_DB)

EXTI 弹跳消除寄存器(EXTI_DB)																													
偏移地址:0x2C																													
复位值:0x0000 0000																													
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
										DBEN21	DBEN20			DBEN17	DBEN16	DBEN15	DBEN14	DBEN13	DBEN12	DBEN11	DBEN10	DBEN9	DBEN8	DBEN7	DBEN6	DBEN5	DBEN4	DBEN3	DBEN2	DBEN1	DBEN0

—	Bits 31-22	—	—
DBEN21	Bit 21	R/W	**消除EXTI通道y弹跳的功能开关。(y=20,21)。** 0:关闭EXTI通道y弹跳消除功能。 1:启用EXTI通道y弹跳消除功能。
DBEN20	Bit 20	R/W	
—	Bits 19-18	—	—
DBEN15	Bit 17	R/W	**消除EXTI通道y弹跳的功能开关。(y=0…17)。** 0:关闭EXTI通道y弹跳消除功能。 1:启用EXTI通道y弹跳消除功能。
DBEN15	Bit 16	R/W	
DBEN15	Bit 15	R/W	
DBEN14	Bit 14	R/W	
DBEN13	Bit 13	R/W	
DBEN12	Bit 12	R/W	
DBEN11	Bit 11	R/W	
DBEN10	Bit 10	R/W	
DBEN9	Bit 9	R/W	
DBEN8	Bit 8	R/W	
DBEN7	Bit 7	R/W	
DBEN6	Bit 6	R/W	
DBEN5	Bit 5	R/W	
DBEN4	Bit 4	R/W	
DBEN3	Bit 3	R/W	
DBEN2	Bit 2	R/W	
DBEN1	Bit 1	R/W	
DBEN0	Bit 0	R/W	

12. 5. 2. 13 EXTI弹跳消除取样率控制寄存器 (EXTI_DBC)

EXTI 弹跳消除取样率控制寄存器(EXTI_DBC)																													
偏移地址:0x30																													
复位值:0x0000 0000																													
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																DBPRE<7:0>													DBCNT<2:0>		

—	Bits 31-16	—	—
DBPRE	Bits 15-8	R/W	**弹跳消除预分频器。** 配置弹跳消除的取样时间: 0:无间隔，根据 APB 时钟频率取样。 1:取样频率为 APB 时钟频率/2。 … 255:取样频率为 APB 时钟频率/256。
—	Bits 7-3	—	—
DBCNT	Bits 2-0	R/W	**DBCNT:弹跳消除计数器。** 配置弹跳消除的计数次数: 000:输出值立即反映输入状态。 001:取到 2 次相同的输入状态后才会更换输出值。 010:取到 3 次相同的输入状态后才会更换输出值。 011:取到 4 次相同的输入状态后才会更换输出值。 100:取到 5 次相同的输入状态后才会更换输出值。 101:取到 6 次相同的输入状态后才会更换输出值。 110:取到 7 次相同的输入状态后才会更换输出值。 111:取到 8 次相同的输入状态后才会更换输出值。

12. 5. 2. 14 EXTI中断配置寄存器 1 (EXTI_ICFG1)

EXTI 中断配置寄存器 1 (EXTI_ICFG1)																															
偏移地址:0x34																															
复位值:0x0000 0000																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXTI7<3:0>				EXTI6<3:0>				EXTI5<3:0>				EXTI4<3:0>				EXTI3<3:0>				EXTI2<3:0>				EXTI1<3:0>				EXTI0<3:0>			

EXTI7	Bits 31-28	R/W	**EXTIy:中断配置。(y=0…7)。** 选择GPIOA/B/C为EXTI的外部中断源。 0000: PA[y] 引脚。 0001: PB[y] 引脚。 0010: PC[y] 引脚。 其他配置保留。
EXTI6	Bits 27-24	R/W	
EXTI5	Bits 23-20	R/W	
EXTI4	Bits 19-16	R/W	
EXTI3	Bits 15-12	R/W	
EXTI2	Bits 11-8	R/W	
EXTI1	Bits 7-4	R/W	
EXTI0	Bits 3-0	R/W	

12. 5. 2. 15 EXTI中断配置寄存器 2 (EXTI_ICFG2)

EXTI 中断配置寄存器 2 (EXTI_ICFG2)																															
偏移地址:0x38																															
复位值:0x0000 0000																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXTI15<3:0>				EXTI14<3:0>				EXTI13<3:0>				EXTI12<3:0>				EXTI11<3:0>				EXTI10<3:0>				EXTI9<3:0>				EXTI8<3:0>			

EXTI15	Bits 31-28	R/W	**EXTIy:中断配置。(y=8…15)。** 选择GPIOA/B为EXTI的外部中断源。 0000: PA[y] 引脚。 0001: PB[y] 引脚。 0010: PC[y] 引脚。 其他配置保留。
EXTI14	Bits 27-24	R/W	
EXTI13	Bits 23-20	R/W	
EXTI12	Bits 19-16	R/W	
EXTI11	Bits 15-12	R/W	
EXTI10	Bits 11-8	R/W	
EXTI9	Bits 7-4	R/W	
EXTI8	Bits 3-0	R/W	

第13章 模拟比较器 (CMP)

13. 1 概述

模拟比较器是一个外围设备，它可以比较两个仿真电压的值，并以逻辑输出的形式显示比较结果。模拟比较器支持两个单独的比较器(CMP1 和 CMP2)，每个比较器可以向设备管脚提供输出并替换电路板上的仿真比较器。独立的外部参考电压。

13. 2 特性

- 两个单独的模拟比较器
- 共享内部参考电压
- CMP1 与 CMP2 可以配置为一个监控窗口比较器
- 比较器的输出可搭配消隐功能
- 比较器的正端以及负端输入皆可配置来源，输入的讯号可由外部脚位以及内部产生电压
- 提供触发给其他 IP 使用(Timer)
- 每个比较器都可以产生中断(通过 EXTI 控制器)

13. 3 结构图

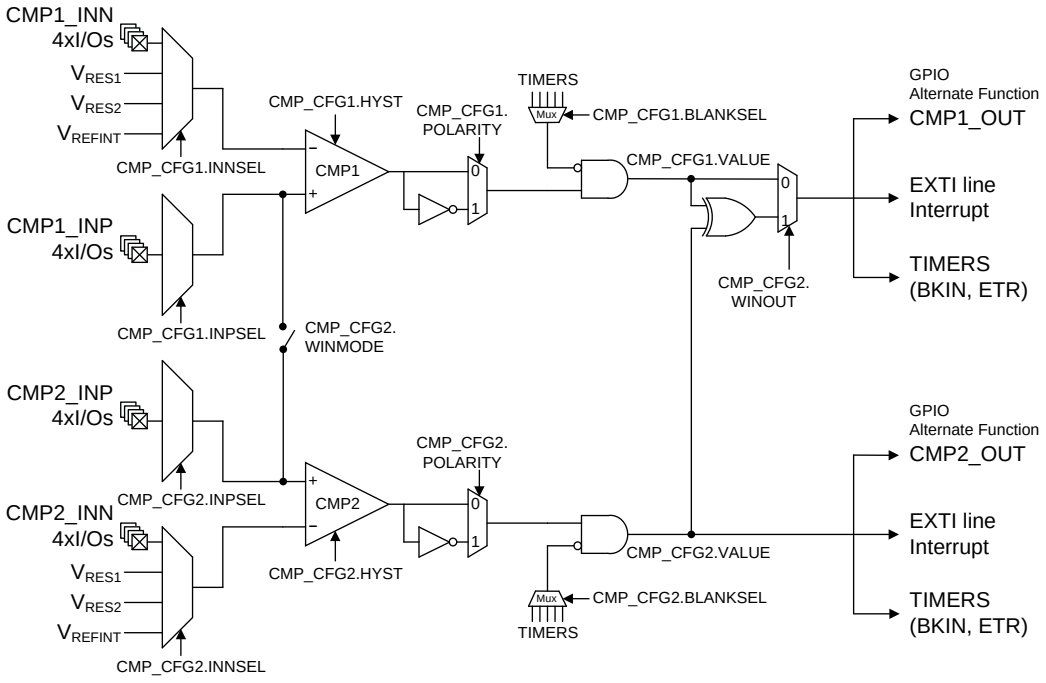


图 13-1 CMP 架构图

13. 4 功能描述

13. 4. 1 CMP引脚与外部信号

CMP 输入的 I/O 必须在 GPIO 中以模似模式配置寄存器。CMP 输出可以使用多路复用引脚功能选择连接到 I/O 上，请参考数据表的“引脚复用功能”章节。

CMP 输出还可以触发以下各种计时器:

- 使用 BKIN 来紧急关闭 PWM 信号。
- 输入捕捉次数计数。

CMP 可以将输出同时输出至内部与外部做使用。

INPSEL[1:0]	CMP1_INP	CMP2_INP
00	PA04	PA01
01	PA05	PA02
10	PA06	PA03
11	PA00	PB11

表 13-1 正端输入讯号选择

INNSEL[2:0]	CMP1_INN	CMP2_INN
000	PB07	PB06
001	PA10	PA09
010	PA11	PA08
011	PA07	PB10
100	V_{RES1}	
101	V_{RES2}	
110	V_{RESINT}	
111	Reserve	

表 13-2 负端输入讯号选择

注: 若要使用 V_{RES}，需至 SYSCFG 寄存器中配置 RESEN 与 RESSRC，来选择分压电路的电压来源与启动分压电路。详细内容请参考 System Config (SYSCFG)章节。

13. 4. 2 CMP复位与时钟

CMP 输出是使用 PCLK(APB 时钟)进行同步后的结果。

13. 4. 3 CMP锁定机制

CMP 可用于安全目的，例如过载保护或过热保护。对于拥有特定安全功能要求的应用，有必要确保在任何情况下，CMP 的寄存器配置皆不能被更改。为此，可以对 CMP 控制和状态寄存器进行写入保护(意即只能读取)。

CMP 配置完成后，可以将 CMP_CFGx.LOCK 位设置为 1。这将导致整个 CMP_CFGx 寄存器变为只能进行读取操作，包括 LOCK 位。

开启写入保护后只能通过 RCU 来将寄存器复位。

13. 4. 4 CMP窗口模式

比较器的窗口模式用于监控模拟电压是否处于阀值上下限锁定一的特定电压范围内。可使用 2 个比较器来建立监控窗口。受监控的模拟电压连接到 2 个比较器的正端输入，阀值上下限电压连接到 2 个比较器的负端输入。可通过开启 WINMODE 位将 2 个正端输入在内部互相连接，进而保留一个外部 IO 来用于其他用途。

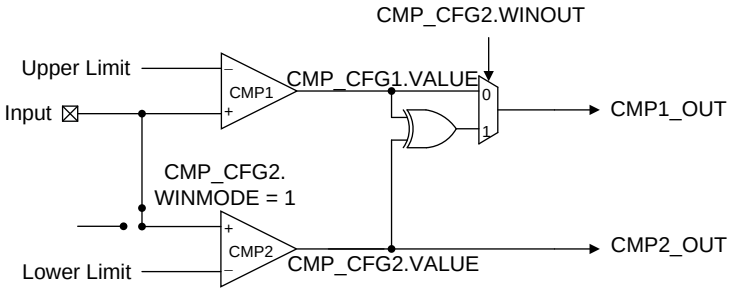


图 13-2 CMP 窗口模式

CMP_CFG2.WINOUT	条件	CMP1_OUT	CMP2_OUT
0	Input > Upper Limit	1	1
	Lower Limit < Input < Upper Limit	0	1
	Input < Lower Limit	0	0
1	Input > Upper Limit	0	1
	Lower Limit < Input < Upper Limit	1	1
	Input < Lower Limit	0	0

表 13-3 窗口模式输出对照

13. 4. 5 CMP迟滞功能

CMP 具有可配置的迟滞功能，当比较器输入信号有噪声时，可能会导致有不稳定的输出，开启迟滞功能可避免此问题。如果不需要迟滞功能，可以将其禁用。

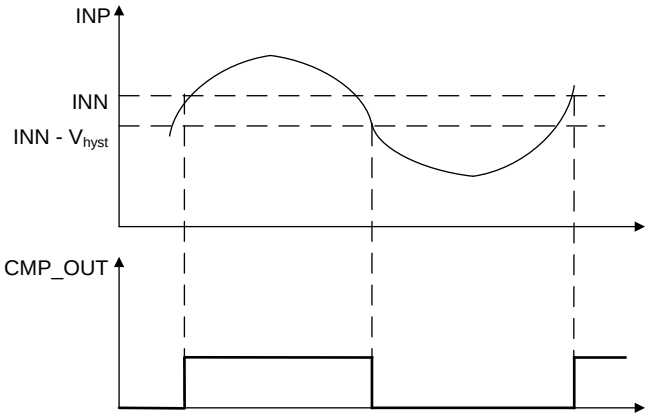


图 13-3 迟滞功能示意图

13. 4. 6 CMP消隐功能

消隐功能的目的是防止电流在 PWM 周期开始处出现短暂的电流尖峰(通常为功率开关，反向并联二极管中的恢复电流)时发生跳闸。这功能是通过配置定时器输出比较信号来定义消隐窗口的大小，并通过软件配置 CMP_CFGx.BLANKSEL[5:0]进行选择(请参见比较器寄存器说明)。然后，将消隐信号进行取反后与比较器输出执行逻辑"AND"运算，以提供所需的比较器输出。请参考下图所示的例子：

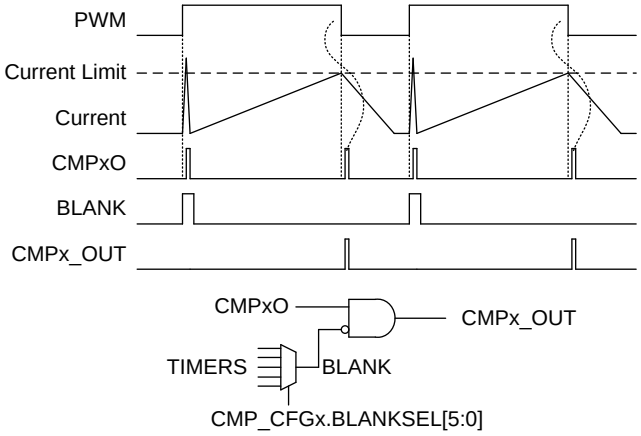


图 13-4 消隐功能示意图

BLANKSEL[5:0]	CMP1	CMP2
xxxxx1	AD16C6T1_OC6	
xxxx1x	AD16C6T2_OC6	
xxx1xx	GP32C4T1_OC4	
xx1xxx	GP32C4T2_OC4	
x1xxxx	GP16C2T1_OC2	
1xxxxx	GP16C2T2_OC2	

表 13-4　消隐讯号选择

13. 4. 7　CMP中断功能

CMP 输出在内部会连接到扩展中断和事件控制器。每个 CMP 都有自己的 EXTI，并且可以产生中断或事件。

13. 5 特殊功能寄存器

13. 5. 1 寄存器列表

CMP 寄存器列表			
名称	偏移地址	类型	描述
CMP_CFG1	0000_H	R/W	CMP 配置寄存器 1
CMP_CFG2	0004_H	R/W	CMP 配置寄存器 2

13. 5. 2 寄存器描述

13. 5. 2. 1 CMP配置寄存器 1 (CMP_CFG1)

CMP 配置寄存器 1(CMP_CFG1)																												
偏移地址:0x00																												
复位值:0x0000 0000																												
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LOCK	VALUE	—	—	—	—	—	BLANKSEL<5:0>						—	—	HYST	POLARITY	—	—	—	—	—	INPSEL<1:0>		—	INNSEL<2:0>			—	—	—	EN

LOCK	Bit 31	R/W	**Comparator 1 锁定** 0: CMP_CFG1[31:0]位可正常读写操作。 1: CMP_CFG1[31:0]位仅供读操作。
VALUE	Bit 30	R	**Comparator 1 输出** 0: CMP输出为低准位。 1: CMP输出为高准位。
—	Bit 29-25	—	—
BLANKSEL	Bit 24-19	R/W	**Comparator 1 消隐功能** 000000: 关闭消隐功能。 xxxxx1: 开启AD16C6T1_OC6消隐功能。 xxxx1x: 开启AD16C6T2_OC6消隐功能。 xxx1xx: 开启GP32C4T1_OC4消隐功能。 xx1xxx: 开启GP32C4T2_OC4消隐功能。 x1xxxx: 开启GP16C2T1_OC2消隐功能。 1xxxxx: 开启GP16C2T2_OC2消隐功能。 可同时开启多个输入消隐。
—	Bit 18-17	—	—
HYST	Bit 16	R/W	**Comparator 1 迟滞功能** 0: 关闭迟滞功能。 1: 开启迟滞功能。
POLARITY	Bit 15	R/W	**Comparator 1 输出极性** 0: 输出无反相。 1: 输出反相。
—	Bit 14-10	—	—
INPSEL	Bit 9-8	R/W	**Comparator 1 正端输入选择** 00: PA04 01: PA05

			10: PA06 11: PA00
—	Bit 7	—	—
INNSEL	Bit 6-4	R/W	**Comparator 1 负端输入选择** 000: PB07 001: PA10 010: PA11 011: PA07 100: V_{RES1} 101: V_{RES2} 110: V_{RESINT} 111: Reserved
—	Bit 3-1	—	—
EN	Bit 0	R/W	**Comparator 1 开关** 0: Comparator 1 关闭。 1: Comparator 1 启动。

13. 5. 2. 2 CMP配置寄存器 2 (CMP_CFG2)

CMP 配置寄存器 2 (CMP_CFG2)																												
偏移地址:0x04																												
复位值:0x0000 0000																												
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LOCK	VALUE	—	—	—	—	—	BLANKSEL<5:0>						—	—	HYST	POLARITY	—	—	—	—	—	INPSEL<1:0>		—	INNSEL<2:0>			WINOUT	WINMODE	—	EN

LOCK	Bit 31	R/W	**Comparator 2 锁定** 0: CMP_CFG2[31:0]位可正常读写操作。 1: CMP_CFG2[31:0]位仅供读操作。
VALUE	Bit 30	R	**Comparator 2 输出** 0: CMP输出为低准位。 1: CMP输出为高准位。
—	Bit 29-25	—	—
BLANKSEL	Bit 24:19	R/W	**Comparator 1 消隐功能** 000000: 关闭消隐功能。 xxxxx1: 开启AD16C6T1_OC6消隐功能。 xxxx1x: 开启AD16C6T2_OC6消隐功能。 xxx1xx: 开启GP32C4T1_OC4消隐功能。 xx1xxx: 开启GP32C4T2_OC4消隐功能。 x1xxxx: 开启GP16C2T1_OC2消隐功能。 1xxxxx: 开启GP16C2T2_OC2消隐功能。 可同时开启多个输入消隐。
—	Bit 18-17	—	—
HYST	Bit 16	R/W	**Comparator 2 迟滞功能** 0: 关闭迟滞功能。 1: 开启迟滞功能。
POLARITY	Bit 15	R/W	**Comparator 2 输出极性** 0: 输出无反相。 1: 输出反相。
—	Bit 14-10	—	—
INPSEL	Bit 9:8	R/W	**Comparator 2 正端输入选择** 00: PA01 01: PA02 10: PA03

			11: PB11
—	Bit 7	—	—
INNSEL	Bit 6:4	R/W	**Comparator 2 负端输入选择** 000: PB06 001: PA09 010: PA08 011: PB10 100: V_{RES1} 101: V_{RES2} 110: V_{RESINT} 111: *Reserved*
WINOUT	Bit 3	R/W	**Comparator 2 窗口输出模式** 0: 关闭CMP1窗口输出模式。 1: 开启CMP1窗口输出模式。
WINMODE	Bit 2	R/W	**Comparator 2 窗口监控模式** 0: 关闭窗口监控模式。 1: 开启窗口监控模式。
—	Bit 1	—	—
EN	Bit 0	R/W	**Comparator 2 开关** 0: Comparator 2 关闭。 1: Comparator 2 启动。

第14章 模数转换器（ADC)

14. 1 概述

12 位 ADC 是一种逐次逼近型模拟数字转换器(SAR A/D)，具有 16 个外部输入信号和 3 个内部信号。各种不同通道的 A/D 转换可在单次、连续、扫描或不连续模式下进行。ADC 的结果可在一个 16 位寄存器中读取，并且数据可以选择左对齐或右对齐形式。此外，ADC 具有模拟看门狗特性，允许应用检测输入电压是否高过了用户定义的阈值上限或低于阈值下限。

14. 2 特性

- 可配置的分辨率：12 位、10 位、8 位或 6 位
- 中断：在转换结束、插入转换结束、模拟看门狗或溢出事件时产生中断
- 可配置数据对齐方式
- 可选择单次、连续或不连续转换模式
- 可独立配置各个通道的取样时间
- 可配置标准转换和插入转换通道的外部触发极性
- 可配置参考源和转换时钟
- DMA 请求：在标准通道转换期间可产生 DMA 请求
- 4 个专用的插入数据寄存器，供插入通道使用
- 低功耗模式：自动延迟、自动关闭转换模式

14. 3 结构图

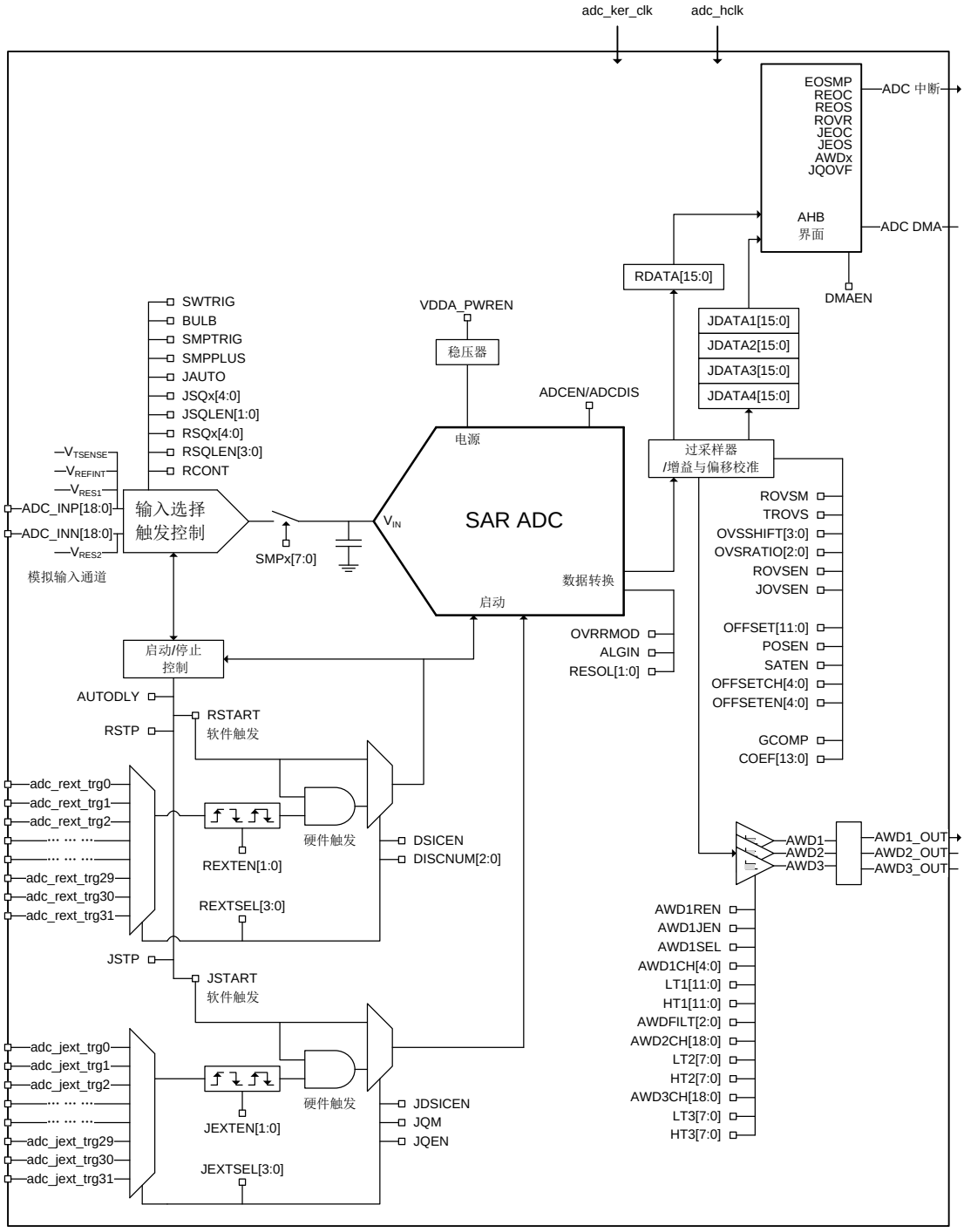


图 14-1　ADC 结构图

14. 4 功能描述

14. 4. 1 ADCClock

ADC 时钟与 AHB ADC 接口的时钟，可以是互相独立的，产生的方式可分成：

- ADC 的时钟来源是 ADC 核心时钟(ADC kernel clock)，可通过 **RCU_CFG2**.ADC_KSRC 选择系统频率、AHB 时钟或 PLL 时钟。ADC 核心时钟的产生方法详细介绍可参考 RCU 章节。
- 可以通过 **ADC_CCR**.PRESCALE 寄存器将 ADC 核心时钟分频以产生 ADC 时钟。

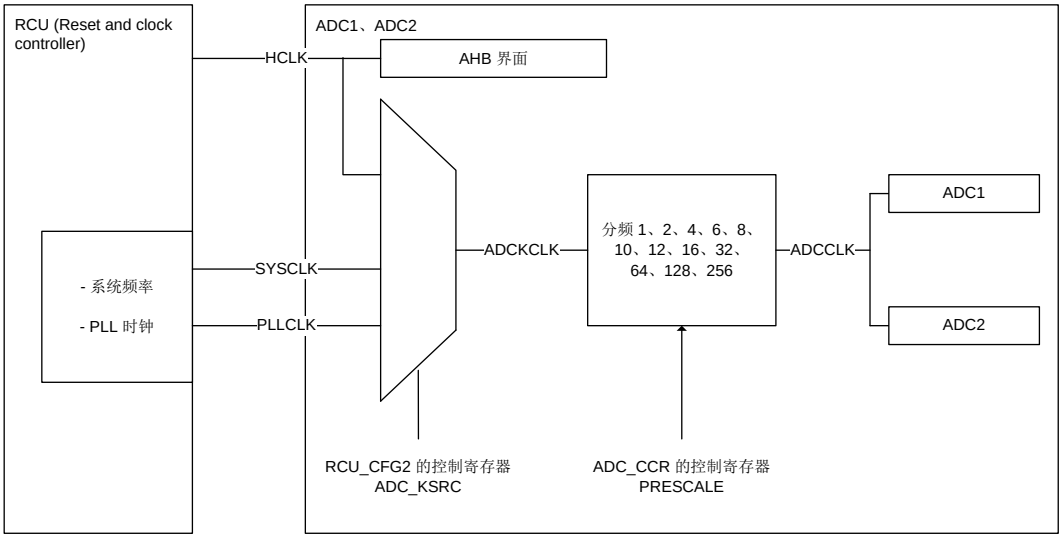


图 14-2 ADC 时钟模式选择

14. 4. 2 ADC开关控制(ADCEN、ADCDIS、ADCRDY)

将寄存器 **ADC_CON** 的 ADCEN 设置为 1，即可启用 ADC。

要关闭 ADC，需要将 **ADC_CON** 的 ADCDIS 位设置为 1，这会将 ADC 关闭并进入断电状态。当 ADC 关闭完成后，硬件会自动清除 **ADC_CON** 的 ADCEN 和 ADCDIS 位状态。

可以通过将 **ADC_CON** 的 RSTART 位设置为 1 或是设置由外部事件触发的标准转换来启动 ADC。

同样地，也可以通过将 **ADC_CON** 的 JSTART 位设置为 1 或是设置由外部事件触发的插入转换来启动 ADC。

- 软件开启 ADC 的流程：
  1. 设置 **SYSCFG_APWR**.VDDA_PWREN 开启电源开关。
  2. 等待 100 us 的电源稳定时间。
  3. 设置 **ADC_CON.**ADCEN。
- 软件关闭 ADC 的流程：

1. 检查 **ADC_CON**.RSTART = 0 以及 **ADC_CON**.JSTART = 0，确保没有转换正在进行。如果停止转换，需要设置 **ADC_CON**.RSTP = 1 或是 **ADC_CON**.JSTP = 1。之后，等待 **ADC_CON**.RSTP = 0 或是 **ADC_CON**.JSTP = 0。
2. 设置 **ADC_CON**.ADCDIS = 1。
3. 等待 **ADC_CON**.ADCEN = 0。(ADCDIS 会自动将 ADCEN 清除为 0)

14. 4. 3 寄存器的写入限制

在 ADC 关闭的状态下(ADCEN=0)，才可以设定 RCU 的配置以开启 ADC 时钟，以及在寄存器 **ADC_CON** 的 LP_EN。

在 ADC 开启并且没有 ADC 关闭请求的状态下(ADCEN=1、ADCDIS=0)，才可以设定 **ADC_CON** 的 RSTART、JSTART 以及 ADCDIS。

其他的控制寄存器 **ADC_CFG1**、**ADC_CFG2**、**ADC_SMPTx**、**ADC_TRx**、**ADC_SQRx**、**ADC_JSQR** 以及 **ADC_OFSTx**：

- 在 ADC 开启以及没有正在进行的标准转换的状态下(ADCEN = 1、RSTART = 0)，才可以设定与标准转换相关的寄存器。
- 在 ADC 开启以及没有正在进行的插入转换的状态下(ADCEN = 1、JSTART = 0)，才可以设定与插入转换相关的寄存器。

在 ADC 开启以及可能有正在进行的标准转换或是插入转换，并且没有 ADC 关闭请求的状态下(ADCEN = 1、RSTART = 1 或是 JSTART = 1、ADCDIS = 0)，才可以设定 **ADC_CON** 的 RSTP 或是 JSTP。

如果设定 **ADC_CFG1**.JQEN=1 开启了插入队列的功能(queue of context for injected converison)的功能，在 ADC 开启的状态下，**ADC_JSQR** 可以随时修改设定。

注: 硬件并没有强制禁止写入 ADC 寄存器，所以在部分情况下设定错误的寄存器可能会导致 ADC 进入未知的状态。为了恢复这情况发生，ADC 必须关闭。

14. 4. 4 通道的选择

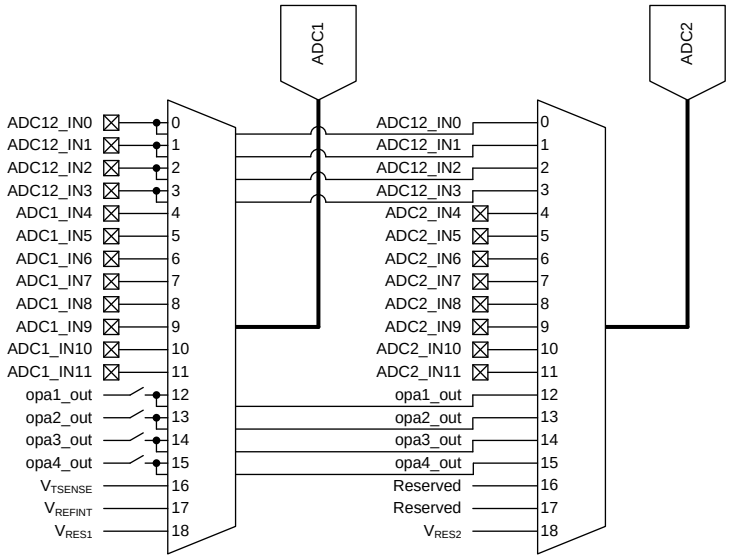


图 14-3 ADC 输入通道

每个 ADC 有 19 条复用通道。

- 藉由 GPIO 的外部模拟输入(ADCx_IN[0..11])
- 有一些内部模拟组件的连接通道如下:
  - 4 个运算放大器 V_{OAPx_OUT} 连接到 ADCx_IN[12..15]
  - 温度传感器 V_{TSENSE} 连接到 ADC1_IN[16]
  - 内部参考电压 V_{REFINT} 连接到 ADC1_IN[17]
  - 内部电阻分压 V_{RES1} 连接到 ADC1_IN[18]，V_{RES2} 连接到 ADC2_IN[18]

注:

1. 转换的通道选择温度传感器，需要先设置 SYSCFG.VTSENSE_EN。
2. 转换通道选择内部参考电压，需要先设置 SYSCFG.VREFINT_EN。
3. 转换内部电阻分压，需要先设置 VRES_CTRL.VRESEN，详细控制方法可以参考 VRES 章节。

ADC 通道	ADC1	ADC2
IN0	ADC12_IN0	ADC12_IN0
IN1	ADC12_IN1	ADC12_IN1
IN2	ADC12_IN2	ADC12_IN2
IN3	ADC12_IN3	ADC12_IN3
IN4	ADC1_IN4	ADC2_IN4
IN5	ADC1_IN5	ADC2_IN5
IN6	ADC1_IN6	ADC2_IN6
IN7	ADC1_IN7	ADC2_IN7
IN8	ADC1_IN8	ADC2_IN8

ADC 通道	ADC1	ADC2
IN9	ADC1_IN9	ADC2_IN9
IN10	ADC1_IN10	ADC2_IN10
IN11	ADC1_IN11	ADC2_IN11
IN12	V_{OPA1_OUT}	V_{OPA1_OUT}
IN13	V_{OPA2_OUT}	V_{OPA2_OUT}
IN14	V_{OPA3_OUT}	V_{OPA3_OUT}
IN15	V_{OPA4_OUT}	V_{OPA4_OUT}
IN16	V_{TSENSE}	Reserved
IN17	V_{REFINT}	Reserved
IN18	V_{RES1}	V_{RES2}

表 14-1　ADC 通道来源选择

ADC 可以分成标准转换和插入转换两组。每组转换包含一个转换序列，可以将任何通道按任意顺序安排在序列中。

- 标准转换序列最多可以包含 16 个转换。必须在 **ADC_SQRx** 设定标准序列的长度和通道顺序。
- 插入转换序列最多可以包含 4 个转换。必须在 **ADC_JSQR** 设定插入序列的长度和通道顺序。

可以通过修改 **ADC_SQRx** 来设定标准转换序列的长度和通道的顺序。但是，在进行修改之前，必须确保没有正在进行标准转换，否则需要先设定 **ADC_CON**.RSTP = 1，使标准转换停止，然后才能进行修改。

只有在 **ADC_CFG1**.JQEN = 1 开启了插入队列的功能，**ADC_JSQR** 才可以随时修改设定。

14. 4. 5　通道的采样时间配置

在 ADC 启动转换之前，需要将 ADC 的输入和内嵌采样电容连接起来，因此需要进行一段足够长的采样时间，以让输入电压源将内嵌电容充电至与输入电压相同的水平。

各个通道皆能独立设置采样时间，可以通过 **ADC_SMPT1**、**ADC_SMPT2**、**ADC_SMPT3**、**ADC_SMPT4** 以及 **ADC_SMPT5** 进行设定。

下列介绍各个通道采样时间寄存器控制 SMPx[7:0] (x 为对应的通道)。

SMP[7:0]，将区分成 SMP[7:6]表示倍数以及起始位置的设定，SMP[5:0]为倍数的基数。

- SMP[7:6]: 为 1、2、4 以及 8 的倍数，并且起始位置不同。
- SMP[5:0]: 倍数相乘的基数 0 至 63。

寄存器的采样时间 (TSMP) 计算如下：

- SMP[7:6]=00: 从 0 开始，采样时间 T_{SMP} = 0 + SMP[5:0] * 1，范围 0 至 63，每一阶差距为 1。

- SMP[7:6]=01: 从 64 开始，采样时间 T_{SMP}= 64 + SMP[5:0] * 2，范围 64 至 190，每一阶差距为 2。
- SMP[7:6]=10：从 192 开始，采样时间 T_{SMP}= 192 + SMP[5:0] * 4，范围 192 至 444，每一阶差距为 4。
- SMP[7:6]=11：从 448 开始，采样时间 T_{SMP} = 448 + SMP[5:0] * 8，范围 448 至 952，每一阶差距为 8。

采样时间的单位为 ADC 时钟，初始的采样时间为 3.5 个 ADC 时钟，所以按照 SMPx[7:0]的设定，总采样时间为：

$$\text{Sampling time}=3.5+T_{SMP}$$

经过采样时间，12 bit 数据总转换时间如下：

$$T_{CONV} = \text{Sampling time} + 13.5\text{ADC 时钟周期}$$

举例，ADC 时钟为 34MHz、采样时间为 3.5 ADC 时钟周期：

$$T_{CONV} = 3.5 + 13.5\text{ADC 时钟周期}=17\text{ADC 时钟周期}= 500\text{ns}$$

各个通道都会有最小的采样时间限制，该时间限制可以参照 datasheets 的 ADC 特性章节。

◆ 快门采样模式 (Bulb sampling mode)

设定 ADC 寄存器 **ADC_CFG2**.BULB = 1 可以开启快门采样模式。在此模式下，当上一个转换完成后，ADC 会立即进入下一个转换的采样周期。当硬件或软件触发转换，ADC 会再经过该通道在寄存器 **ADC_SMPTx** 所设定的采样时间。在快门采样模式下，只有第一个 ADC 转换的采样时间是按照 SMPx[7:0]的设定。快门采样模式会在第二个转换开始时启用。

快门模式无法适用于连续转换模式以及插入转换。

当 **ADC_CFG2**.BULB = 1，就无法设定 **ADC_CFG2**.SMPTRIG。

注: 只有在该次转换需要有外部事件触发，快门采样模式才会有效果。

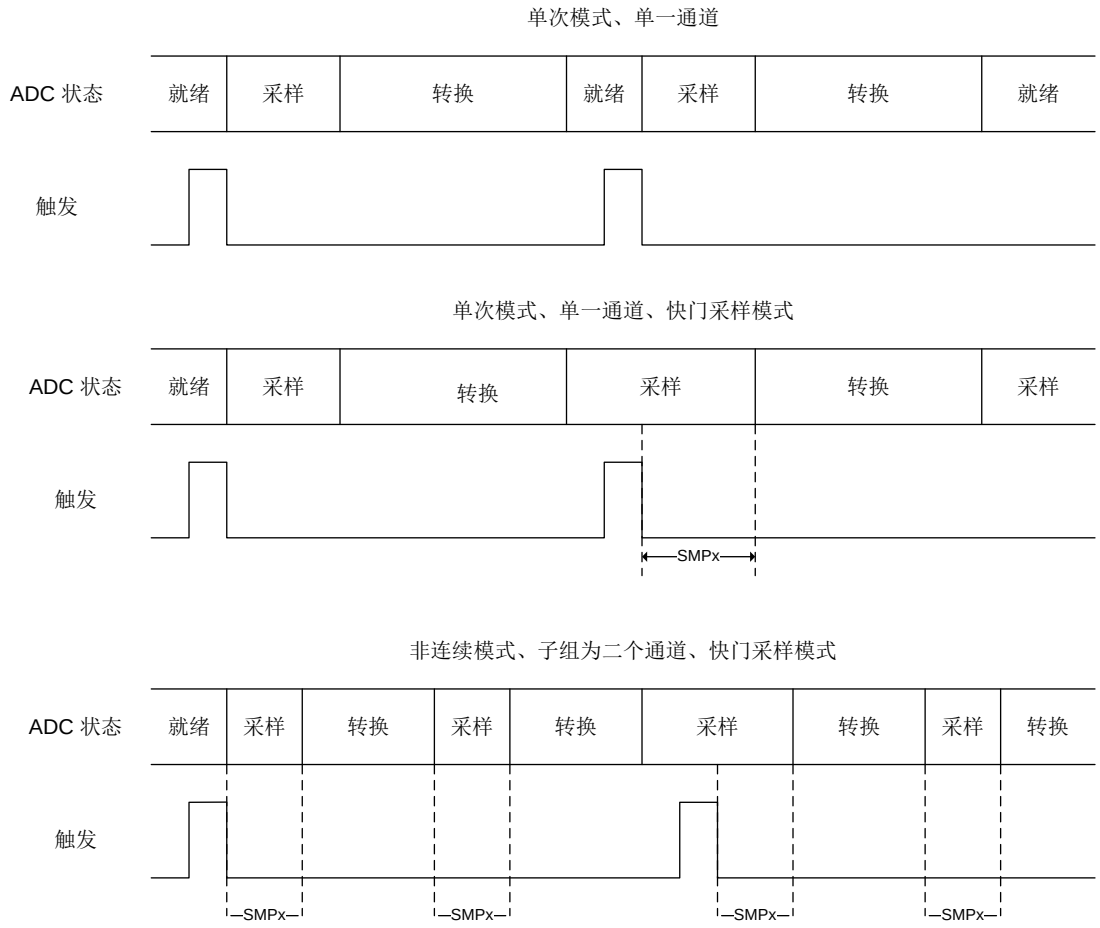


图 14-4　快门模式的时间轴

◆ 采样时间控制触发模式 (Sampling time control trigger mode)

设定 ADC 寄存器 **ADC_CFG2**.SMPTRIG = 1 开启该模式，此模式下，SMPx[7:0]的设定无效，采样时间将由触发信号沿决定。

选择硬件触发时，触发信号上升沿会启动采样周期，下降沿则会结束采样周期并开始转换。硬件触发必须将寄存器 **ADC_CFG1**.REXTEN[1:0]设为 01。

如果选择使用软件触发方式进行 ADC 转换，则不能通过设置 **ADC_CON**.RSTART 来触发转换。必须需要通过设置 **ADC_CFG2**.SWTRIG 来启动采样周期，并且在完成采样后清除该位以开始转换。此外，必须将 **ADC_CFG1**.REXTEN[1:0]设置为 00。

采样时间控制触发模式无法适用于连续转换模式以及插入转换。

只有在需要通过外部事件触发进行转换的情况下，才需要进行采样时间和触发模式的设定。

注: 在单次触发模式下进行多次转换时，如果设置 SMPTRIG，硬件将强制要求每次转换都需要有新的触发事件。

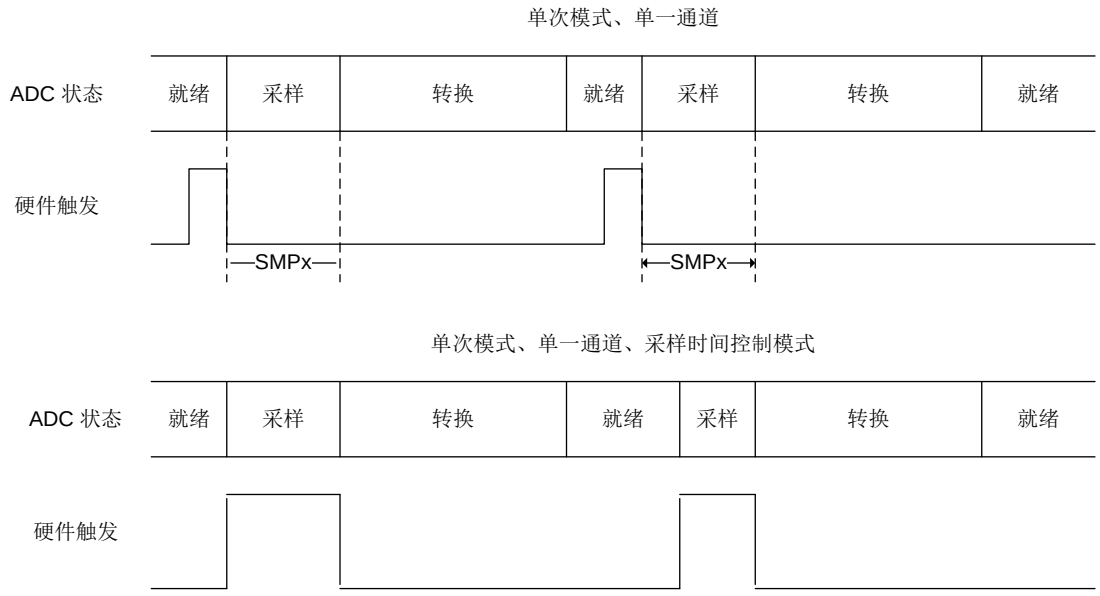


图 14-5　采样时间控制模式的时间轴

◆ 采样时间增加 (Sampling plus control bit)

初始的采样时间为 3.5 个 ADC 时钟周期，12 位的数据，总转换时间为 17 个 ADC 时钟周期。

如果因为 SMPx[7:0]的设定，导致转换的时间不是偶数个 ADC 时钟周期，则在双重 ADC 的交替模式(Interleaved mode)下，就无法将转换时间均分为相等的时间间隔。因此，ADC 提供了一个 **ADC_SMPT5**.SMPPLUS 的功能，可以将基准的采样时间从 3.5 个 ADC 时钟周期增加至 4.5 个 ADC 时钟周期，以避免此类问题。

14. 4. 6 单次转换模式

在单次转换模式下，如果 **ADC_CFG1**.RCONT 设为 0，则 ADC 将对每个设定的通道进行一次转换。

- 于标准通道，设置 **ADC_CON**.RSTART = 1 以启动标准软件触发。
- 对于插入通道，设置 **ADC_CON**.JSTART = 1 以启动插入软件触发。
- 对于标准通道或插入通道，使用外部硬件触发。在使用外部触发之前，需要先将 RSTART 或 JSTART 设置为 1。

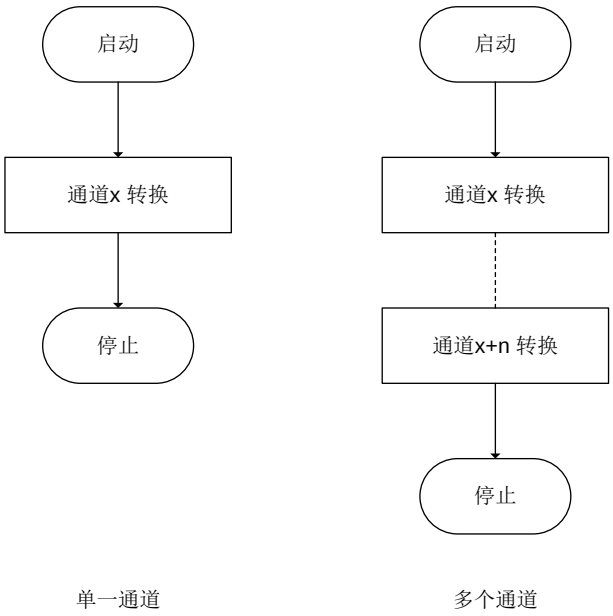


图 14-6 单次转换模式流程图

在标准序列中，每次转换完成后:

- 转换的数据存放在 **ADC_DR** 寄存器中。
- REOC(标准转换结束)的标志位会被设置为 1。
- 如果 **ADC_IER**.REOC 被设置为 1，ADC 将会产生中断。

在插入序列中，每次转换完成后 ：

- 转换数据存放在四个 **ADC_JDRx** 寄存器之一。
- JEOC (插入转换结束)的标志位为 1。
- 如果 **ADC_IER**.JEOC 设置为 1，将会产生中断。

标准序列完成后 ：

- REOS (标准序列结束)的标志位为 1。
- 如果 **ADC_IER**.REOS 设置为 1，将会产生中断。

插入序列完成后 ：

- JEOS (插入序列结束)的标志位为 1。
- 如果 **ADC_IER**.JEOS 设置为 1，将会产生中断。

ADC 在完成标准或插入序列之后，会进入停止模式，直到下一次触发信号到达，触发新的转换序列。在停止模式下，ADC 的所有设置都保持不变，等待新的转换序列触发。

14. 4. 6. 1　　单一通道，单次转换模式

设定序列长度为 1，使用单次转换模式。每次触发，转换单一设定的通道后，ADC 就会停止工作。

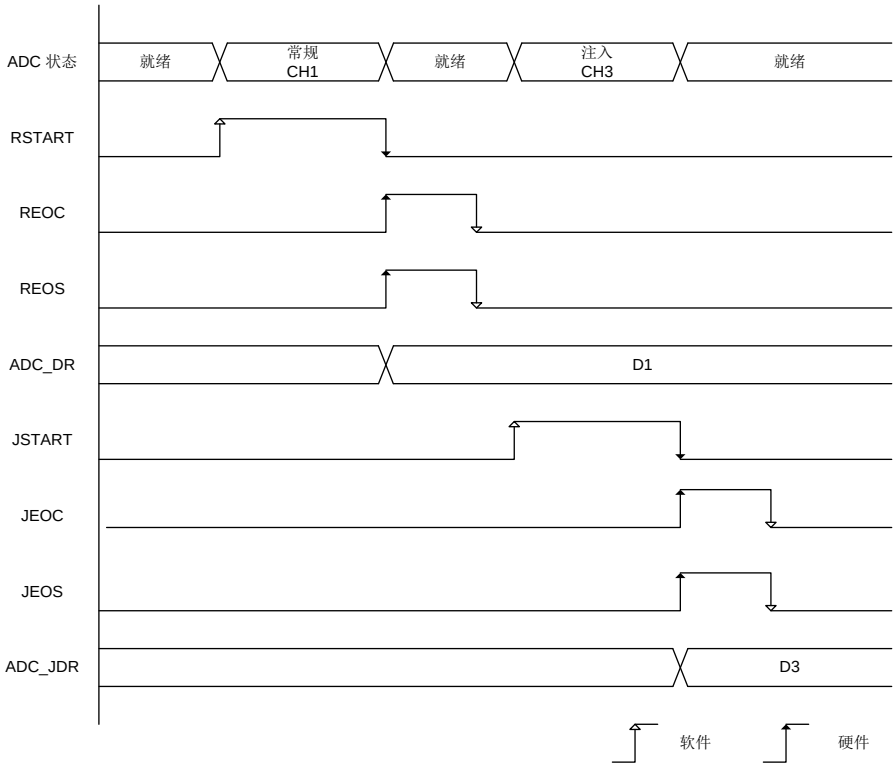


图 14-7　单一通道，单次转换模式，软件触发

注:

1. REXTEN[1:0] = 00、RCONT = 0。
2. JEXTEN[1:0] = 00。
3. RSQLEN[3:0] = 0，序列长度为 1，通道选择 1。
4. JSQLEN[1:0] = 0，序列长度为 1，通道选择 3。

14. 4. 6. 2　多个通道，单次转换模式

此模式可称为扫描模式(SCAN mode)，设定序列长度大于 1，使用单次转换模式。ADC 会将所有设定的通道进行一次转换，然后停止工作，直到下一次触发。

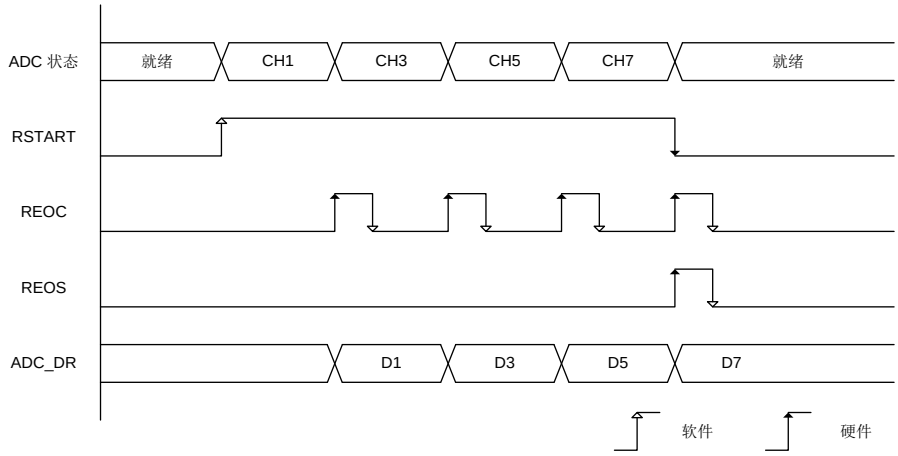


图 14-8　多个通道，单次转换模式，软件触发

注:

1. REXTEN[1:0] = 00、RCONT = 0
2. RSQLEN[3:0] = 3，序列长度为 4，通道选择 1、3、5、7。

14. 4. 7　连续转换模式

当 **ADC_CFG1**.RCONT 设置为 1 开启连续模式，此模式只适用于标准通道。

ADC 会在完成标准通道的转换后，自动重新开始转换第一个通道，实现了不间断的连续转换。该模式可用于需要连续监测的应用，例如温度监测、电池电压监测等。

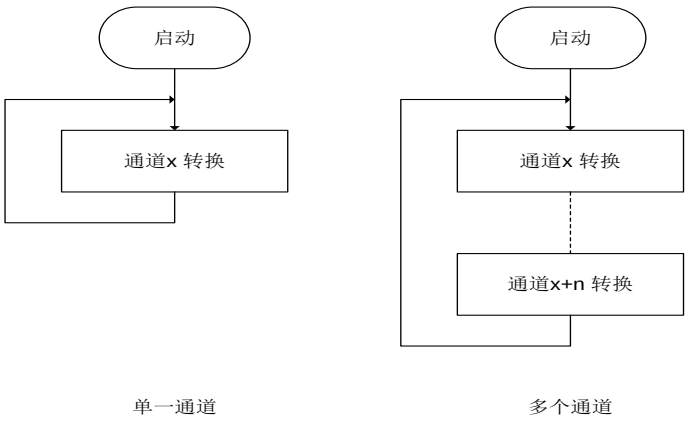


图 14-9　单一通道，连续转换模式流程图

在标准序列中，每次转换完成后 ：

- 转换的数据存放在 **ADC_DR** 寄存器中。
- 如果使用的通道序列中有多个通道，ADC 会自动切换到下一个通道进行下一次转换。
- REOC (标准转换结束) 的标志位会被设置为 1。
- 如果 **ADC_IER**.REOC 被设置为 1，ADC 将会产生中断。

标准序列完成后 ：

- REOS (标准序列结束) 的标志位为 1。
- 如果 **ADC_IER**.REOS 被设置为 1，ADC 将会产生中断。

序列完成后，会立即重复进行序列转换。

注:不能同时设定非连续模式 (DISCEN = 1) 以及连续模式 (RCONT = 1)。插入通道是无法执行连续转换。但是可以藉由自动插入模式的设定，在连续模式的标准序列结束后自动开始进行插入转换。达到插入通道连续转换的期望。

14. 4. 7. 1 单一通道，连续转换模式

设定序列长度为 1，使用连续转换模式，每次触发时都会重复转换单一设定的通道。

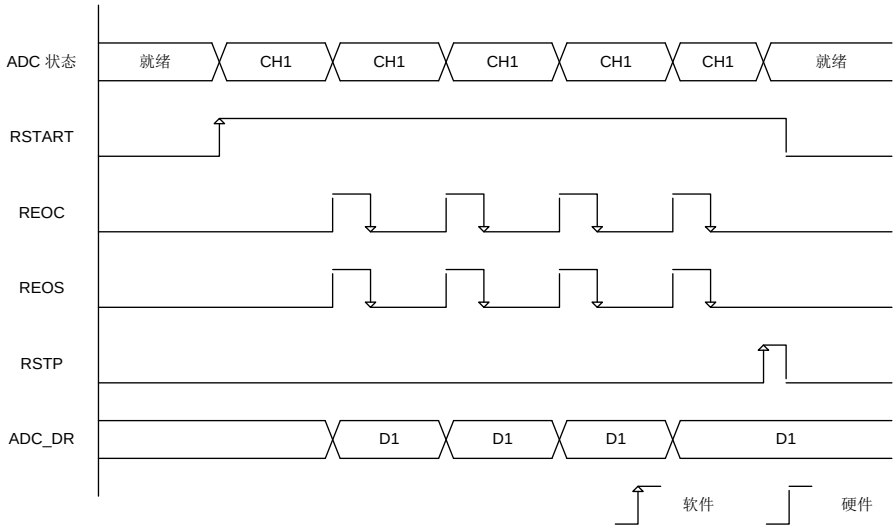


图 14-10 单一通道，连续转换模式，软件转换

注:

1. REXTEN[1:0] = 00、RCONT = 1。
2. RSQLEN[3:0] = 0，序列长度为 1，通道选择 1。

14. 4. 7. 2　多个通道，连续转换模式

设定序列长度大于 1，使用连续转换模式。每次触发，重复转换所有设定的通道。

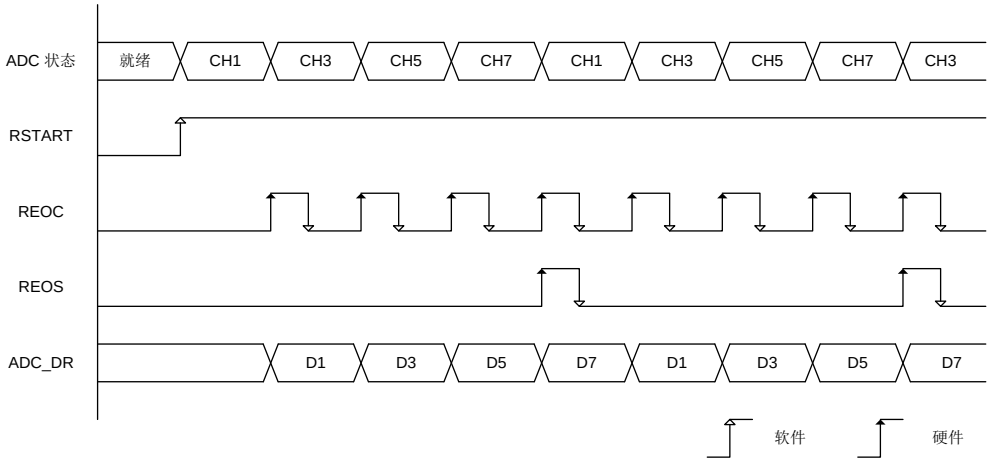


图 14-11　多个通道，连续转换模式，软件转换

注:
1. REXTEN[1:0] = 00、RCONT = 1。
2. RSQLEN[3:0] = 3，序列长度为 4，通道选择 1、3、5、7。

14. 4. 8　启动转换 (RSTART、JSTART)

ADC 可藉由软件设定 **ADC_CON**.RSTART 来启动标准转换，启动标准转换的条件如下：

- 若 REXTEN[1:0]设为 00，则设为软件触发模式，启动后即立即开始转换。
- 若 REXTEN[1:0]不为 00，则设为硬件触发模式，需要侦测到有效的触发边沿才会启动转换。

软件设定 **ADC_CON**.JSTART 以启动 ADC 插入转换。

ADC 可藉由软件设定 **ADC_CON**.JSTART 来启动插入转换，启动插入转换的条件如下：

- 若 JEXTEN[1:0]设为 00，则设为软件触发模式，启动后即立即开始转换。
- 若 JEXTEN[1:0]不为 00，则设为硬件触发模式，需要侦测到有效的触发边沿才会启动转换。

注: 在自动插入模式(JAUTO=1)，RSTART 启动标准转换。当标准序列结束后，会自动启动插入转换(JSTART 需要被清除为 0)。

当 RSTART=0 以及 JSTART=0 时，代表 ADC 停止进行转换，此时可以重新配置 ADC 的相关设定，包括时钟频率、通道选择、触发模式、转换精度等等。在重新配置完毕之后，若要启动 ADC 的转换，则需要再次将 RSTART 或 JSTART 设定为 1。

硬件清除 RSTART 的时间点 ：

- 在单次转换模式下，使用软件触发(REXTEN[1:0] = 00、RCONT = 0)，当转换序列结束后

(REOS = 1)就会清除为 0。

- 在非连续转换模式下，使用软件触发(REXTEN[1:0] = 00、RCONT = 0、DSICEN = 1)，当子序列转换结束后就会清除为 0。
- 在所有模式下，设置 RSTP = 1，就会清除为 0。

注: 在连续模式下，转换序列是自动重新启动的，因此在 REOS 发生之后，硬件不会清除 RSTART。如果设定软件触发，则应在 REOC 被清除之后再设置 RSTART。

当使用硬件触发时，单次转换模式下 REOS 发生后，硬件不会清除 RSTART，这样的设计是为了避免错过后续的硬件触发事件而设计的。这样可以确保不会在转换序列结束后立即重新启动转换，而是等待下一次硬件触发事件的发生。这样可以确保转换的正确性和稳定性。

硬件清除 JSTART 的时间点 ：

- 在单次转换模式下，使用软件触发(JEXTEN[1:0] = 00)；当转换序列结束后(JEOS = 1)，就会清除为 0。
- 在非连续转换模式下，使用软件触发(JEXTEN[1:0] = 00、JDSICEN = 1)；当每次转换结束后(JEOC = 1)，就会清除为 0。
- 在所有模式下，设置 JSTP = 1，就会清除为 0。

与标准相同，使用硬件触发时，单次转换模式下 JEOS 发生后，硬件不会清除 JSTART。

14. 4. 9 ADC 时间

ADC 转换时间，可以区分成两个阶段，采样(sample period)以及 SAR 转换(SAR converted period)。

$$T_{CONV} = T_{SMPL} + T_{SAR} = \left[6.5 + 13.5_{|12bit}\right] \times T_{ADCCLK}$$

$$T_{CONV}=T_{SMPL}+T_{SAR}=\frac{20bit}{48\ MHz}\cong 417ns$$

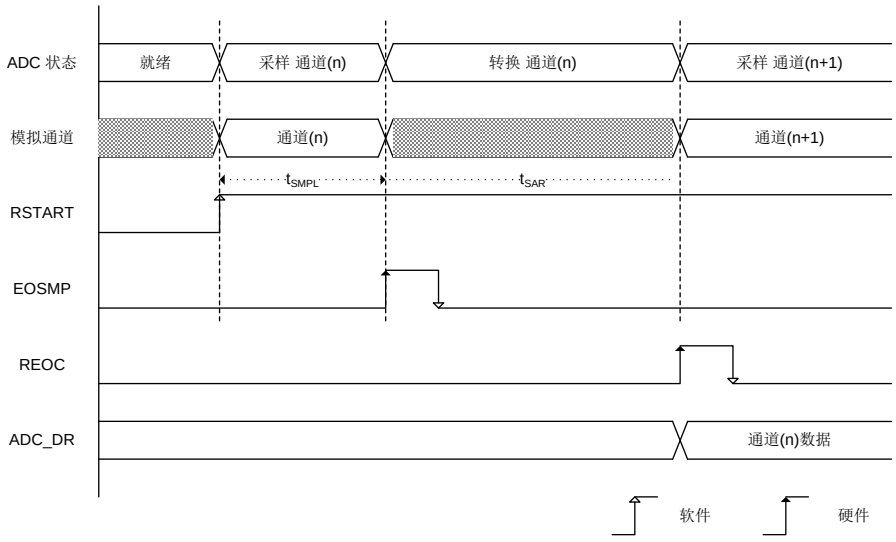


图 14-12 ADC 转换时间

注:

1. T_{SMPL} 取决在寄存器 SMPx[7:0]
2. T_{SAR} 取决在寄存器 RESOL[1:0]

14. 4. 10 停止转换 (RSTP、JSTP)

软件可以通过设置 **ADC_CON**.RSTP = 1 以及 **ADC_CON**.JSTP = 1 分别停止标准转换以及插入转换。

停止转换，将正在运行的 ADC 进行复位。随后可以重新配置 ADC 转换所需要的寄存器。

- 设置 **ADC_CON**.RSTP = 1，会停止正在进行的标准转换，并且 **ADC_DR** 不会是当前转换的结果。
- 设置 **ADC_CON**.JSTP = 1，会停止正在进行的插入转换，并且 **ADC_JDR** 不会是当前转换的结果。

在标准转换时，设置 RSTP 后，RSTART 以及 RSTP 皆由硬件清除状态。设置 JSTP 后，JSTART 以及 JSTP 皆由硬件清除状态。

注: 在自动插入模式下，只有设置 RSTP 才可以有效的停止正在进行的标准转换或是插入转换。(不得使用 JSTP)

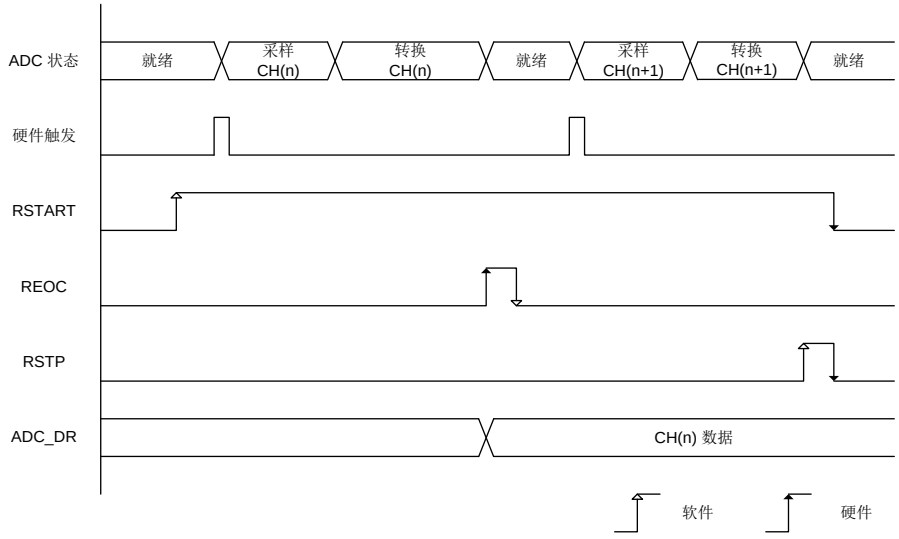


图 14-13 停止正在进行的标准转换

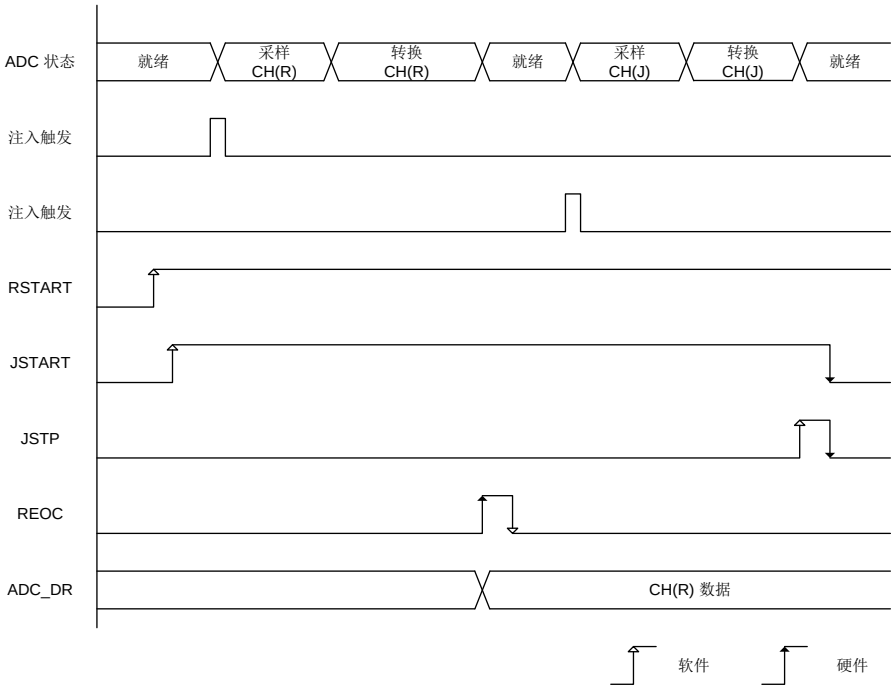


图 14-14　停止正在进行的插入转换

14. 4. 11 外部触发转换和触发极性

序列转换的启动可以通过寄存器设定为软件触发或是外部事件触发。若 REXTEN[1:0]或 JEXTEN[1:0]设定为非零值且符合极性选择的外部事件发生，则可以启动转换。

如果使用插入队列的模式，无法实践软件的插入触发。

只有在 RSTART 或 JSTART 被设置后，触发的选择才会生效。

表 14-2 以及表 14-3 提供了 REXTEN[1:0] 以及 JEXTEN[1:0]设定叙述

REXTEN[1:0]	触发来源
00	启用软件触发侦测，禁用硬件触发侦测
01	侦测上升沿为硬件触发
10	侦测下降沿为硬件触发
11	侦测上升以及下降沿为硬件触发

表 14-2　外部标准触发的极性配置

注: 无法随时改变标准触发的配置。

JEXTEN[1:0]	触发来源
00	当 JQEN 设为 0 时，启用软件触发侦测，禁用硬件触发侦测 当 JQEN 设为 1 时，禁用硬件和软件触发侦测
01	侦测上升沿为硬件触发
10	侦测下降沿为硬件触发
11	侦测上升以及下降沿为硬件触发

表 14-3　外部插入触发的极性配置

注: 当使用插入队列模式，是可以随时更改插入触发的配置。

通过 REXTSEL 和 JEXTSEL，可以分别选择 32 个外部事件作为标准和插入触发事件。外部事件的选择方式可以参考表 14-4 和表 14-5 的描述。

插入触发可以中断进行中的标准转换。

注: 在标准触发的外部事件选择配置中，无法随时更改选择的配置。但若使用插入队列模式，则可以随时更改触发事件的选择。

主机 ADC 与从机 ADC，共享相同的外部触发输入。

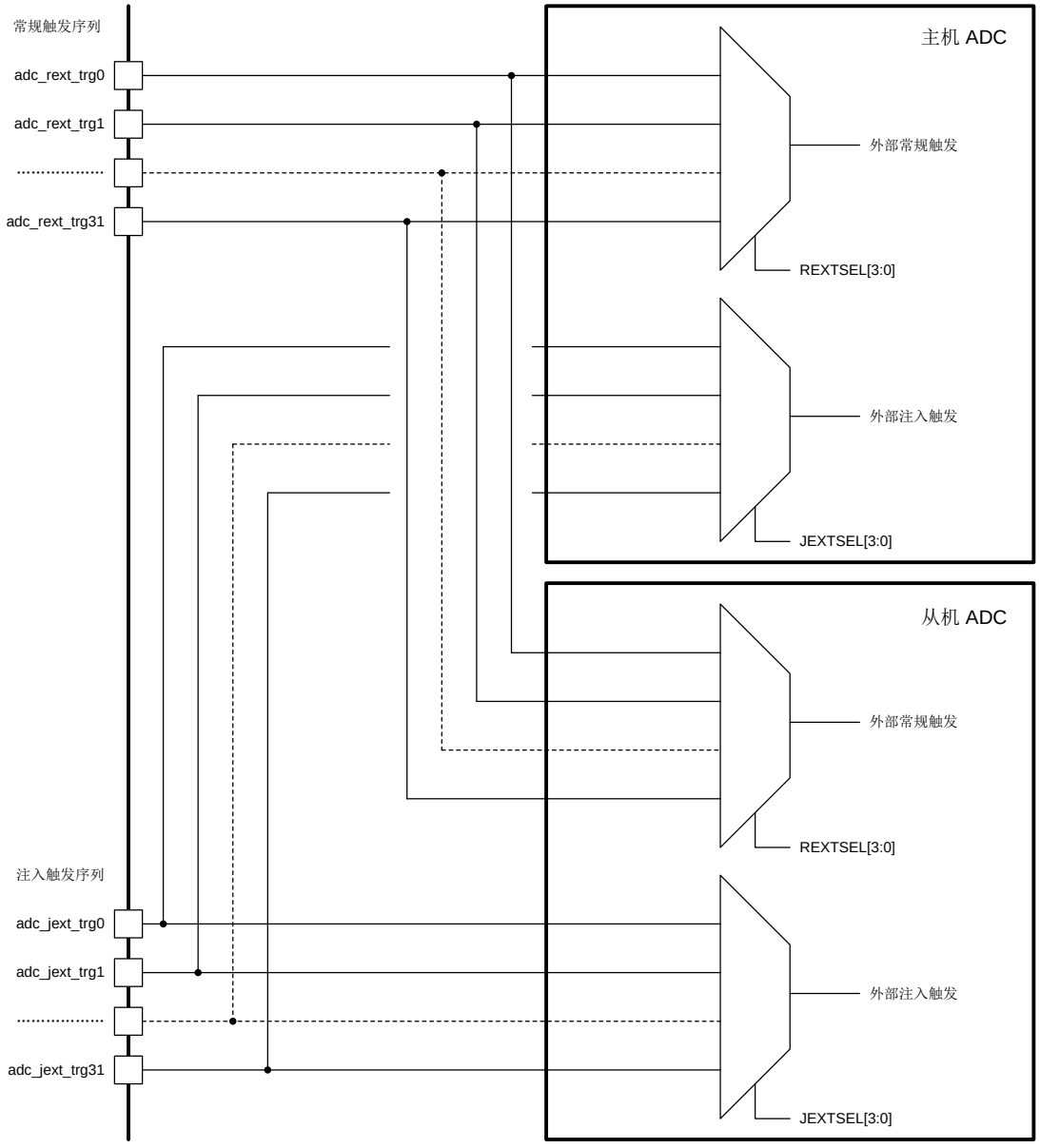


图 14-15　主机与从机 ADC 的触发输入

名称	信号源	信号种类	REXTSEL[4:0]
adc_rext_trg0	AD16C6T1_CH1	内部定时器	00000
adc_rext_trg1	AD16C6T1_CH2	内部定时器	00001
adc_rext_trg2	AD16C6T1_CH3	内部定时器	00010
adc_rext_trg3	GP32C4T1_CH2	内部定时器	00011
adc_rext_trg4	GP32C4T2_TRGOUT	内部定时器	00100
adc_rext_trg5	AD16C6T1_CH4	內部計時器	00101
adc_rext_trg6	EXTI_ADTE1	外部脚位	00110
adc_rext_trg7	AD16C6T2_TRGOUT	内部定时器	00111
adc_rext_trg8	AD16C6T2_TRGOUT2	内部定时器	01000
adc_rext_trg9	AD16C6T1_TRGOUT	内部定时器	01001
adc_rext_trg10	AD16C6T1_TRGOUT2	内部定时器	01010
adc_rext_trg11	GP32C4T1_TRGOUT	内部定时器	01011
adc_rext_trg12	AD16C6T2_CH4	内部定时器	01100
adc_rext_trg13	BS16T_TRGOUT	内部定时器	01101
adc_rext_trg14	GP16C2T1_TRGOUT	内部定时器	01110
adc_rext_trg15	GP32C4T2_CH4	内部定时器	01111
adc_rext_trg16	GP16C2T1_CH1	内部定时器	10000
adc_rext_trg17	GP16C2T1_CH2	内部定时器	10001
adc_rext_trg18	AD16C6T2_CH1	内部定时器	10010
adc_rext_trg19	AD16C6T2_CH2	内部定时器	10011
adc_rext_trg20	AD16C6T2_CH3	内部定时器	10100
adc_rext_trg21	GP16C2T2_CH1	内部定时器	10101
adc_rext_trg22	GP16C2T2_CH2	内部定时器	10110
adc_rext_trg23	GP16C2T2_TRGOUT	内部定时器	10111
adc_rext_trg24	GP32C4T1_CH1	内部定时器	11000
adc_rext_trg25	GP32C4T1_CH3	内部定时器	11001
adc_rext_trg26	GP32C4T1_CH4	内部定时器	11010
adc_rext_trg27	保留	—	11011
adc_rext_trg28	GP32C4T2_CH1	内部定时器	11100
adc_rext_trg29	GP32C4T2_CH2	内部定时器	11101
adc_rext_trg30	GP32C4T2_CH3	内部定时器	11110
adc_rext_trg31	保留	—	11111

表 14-4　标准通道的外部触发选择

名称	信号源	信号种类	REXTSEL[4:0]
adc_jext_trg0	AD16C6T1_TRGOUT	内部定时器	00000
adc_jext_trg1	AD16C6T1_CH4	内部定时器	00001
adc_jext_trg2	GP32C4T1_TRGOUT	内部定时器	00010
adc_jext_trg3	GP32C4T1_CH1	内部定时器	00011
adc_jext_trg4	GP32C4T2_CH4	内部定时器	00100
adc_jext_trg5	GP32C4T2_CH2	内部定时器	00101
adc_jext_trg6	EXTI_ADTE2	外部脚位	00110
adc_jext_trg7	AD16C6T2_CH4	内部定时器	00111
adc_jext_trg8	AD16C6T1_TRGOUT2	内部定时器	01000
adc_jext_trg9	AD16C6T2_TRGOUT	内部定时器	01001
adc_jext_trg10	AD16C6T2_TRGOUT2	内部定时器	01010
adc_jext_trg11	GP32C4T2_CH3	内部定时器	01011
adc_jext_trg12	GP32C4T2_TRGOUT	内部定时器	01100
adc_jext_trg13	GP32C4T2_CH1	内部定时器	01101
adc_jext_trg14	BS16T_TRGOUT	内部定时器	01110
adc_jext_trg15	GP16C2T1_TRGOUT	内部定时器	01111
adc_jext_trg16	GP16C2T1_CH1	内部定时器	10000
adc_jext_trg17	GP16C2T1_CH2	内部定时器	10001
adc_jext_trg18	AD16C6T2_CH1	内部定时器	10010
adc_jext_trg19	AD16C6T2_CH2	内部定时器	10011
adc_jext_trg20	AD16C6T2_CH3	内部定时器	10100
adc_jext_trg21	保留	—	10101
adc_jext_trg22	GP16C2T2_CH2	内部定时器	10110
adc_jext_trg23	GP16C2T2_TRGOUT	内部定时器	10111
adc_jext_trg24	GP32C4T1_CH2	内部定时器	11000
adc_jext_trg25	GP32C4T1_CH3	内部定时器	11001
adc_jext_trg26	GP32C4T1_CH4	内部定时器	11010
adc_jext_trg27	GP16C2T2_CH1	内部定时器	11011
adc_jext_trg28	AD16C6T1_CH1	内部定时器	11100
adc_jext_trg29	AD16C6T1_CH2	内部定时器	11101
adc_jext_trg30	AD16C6T1_CH3	内部定时器	11110
adc_jext_trg31	保留	—	11111

表 14-5　插入通道的外部触发选择

14. 4. 12 插入通道管理

14. 4. 12. 1 触发插入模式 (Triggered injection mode)

在触发插入模式下，**ADC_CFG1**.JAUTO 需要为清除的状态。插入模式下的转换流程如下：

1. 通过外部触发或设置 RSTART 的软件触发，启动标准通道的转换。
2. 在标准转换期间，若发生外部插入触发或设置 JSTART 的软件触发，正在进行的标准转换将会停止，并开始进行插入序列的转换。
3. 当插入序列转换完成后，原先的标准序列转换会从中断的地方恢复转换。
4. 在插入转换期间，若发生标准触发，不会中断正在进行的插入转换。如果发生标准触发时标准序列已经完成，该触发将会被保留，等到插入序列转换完成后，启动标准转换。

注: 若使用触发插入模式，需要确保触发事件的发生时间，使得事件之间的时间差大于插入序列的转换时间，才能顺利进行插入。

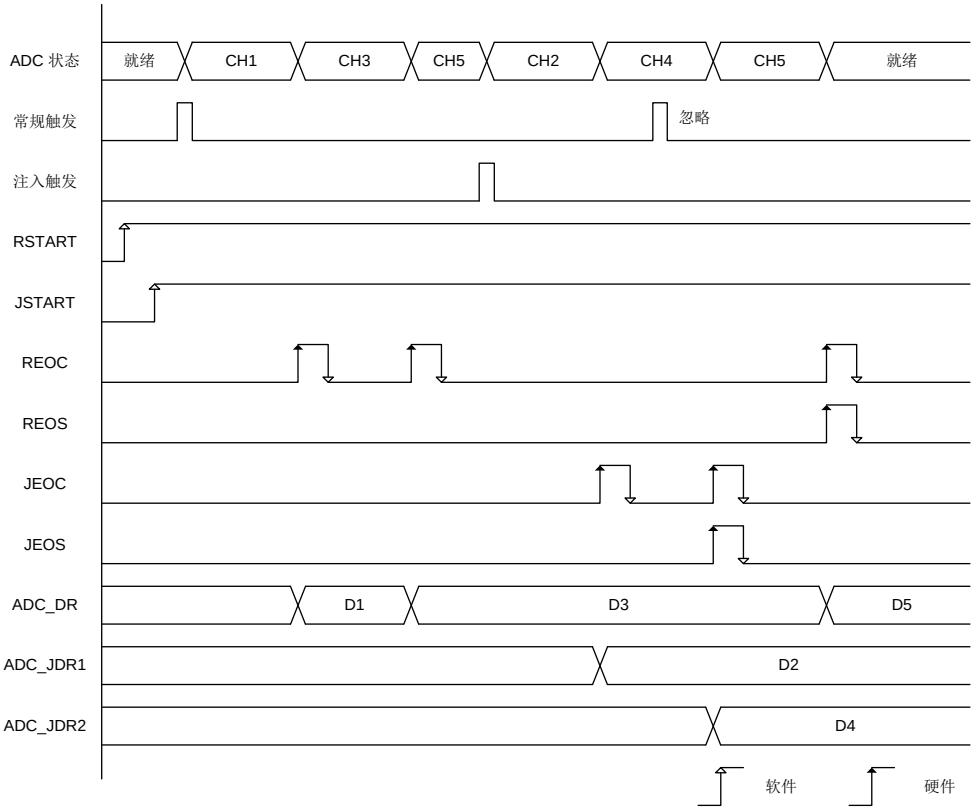


图 14-16 触发插入模式，插入触发在 REOS 发生之前

注:

1. 标准转换为硬件触发，标准序列长度 3，标准通道为 1、3、5。
2. 插入转换为硬件触发，插入序列长度 2，插入通道为 2、4。
3. JAUTO = 0，RCONT = 0。

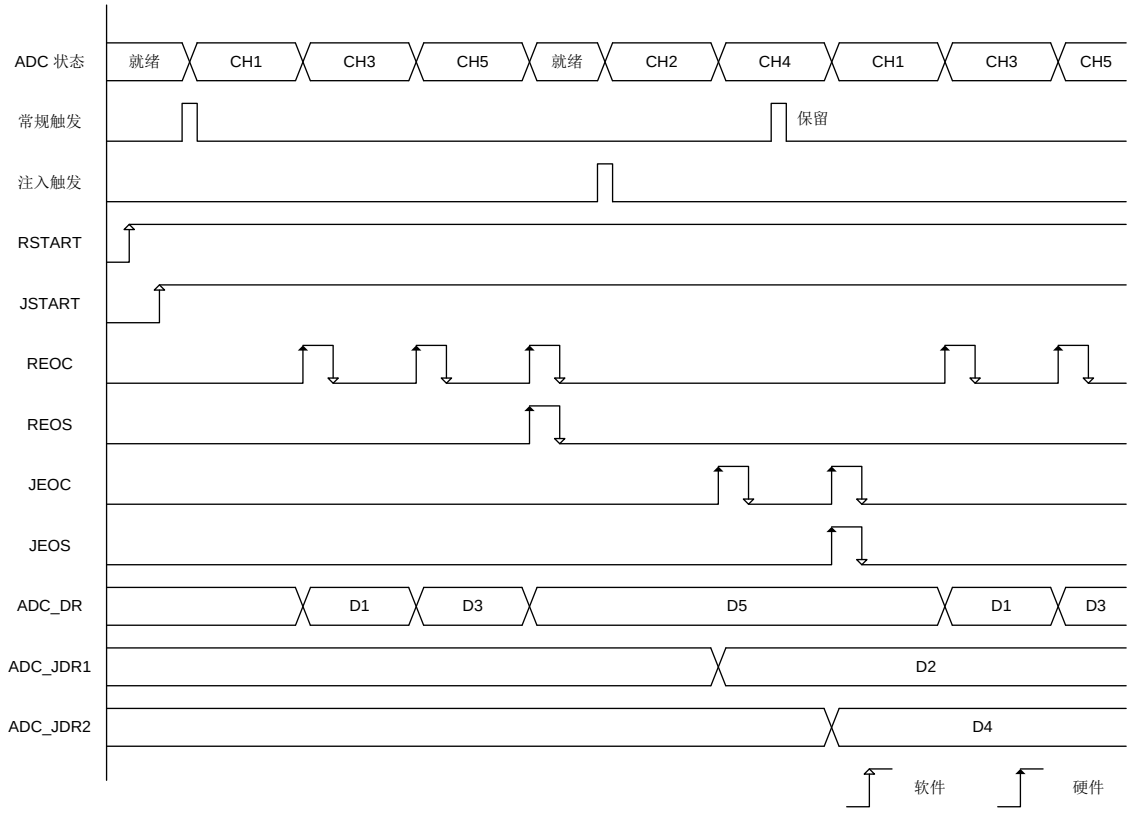


图 14-17　触发插入模式，插入触发在 REOS 发生之后

注:

1. 标准转换为硬件触发，标准序列长度 3，标准通道为 1、3、5。
2. 插入转换为硬件触发，插入序列长度 2，插入通道为 2、4。
3. JAUTO = 0，RCONT = 0。

14. 4. 12. 2 自动插入模式 (Auto-injection mode)

当寄存器 **ADC_CFG1**.JAUTO 设置为 1 时，会启动自动插入模式，插入转换将等到标准序列结束后自动执行，此模式最多可有效执行 20 次转换。

在此模式下，若设置 RSTART 则可启动标准转换，插入转换将接续在标准序列完成后自动执行(此时 JSTART 必须为清除的状态)，而设置 RSTP 则会停止标准及插入转换 (不得使用 JSTP)。

在自动插入模式下，插入通道的外部触发会被禁用。

若设置连续模式(RCONT = 1)，则标准序列完成后会接续进行插入转换，并在插入序列完成后重新开始进行标准转换。

注: 不能够同时存在自动插入以及非连续模式。

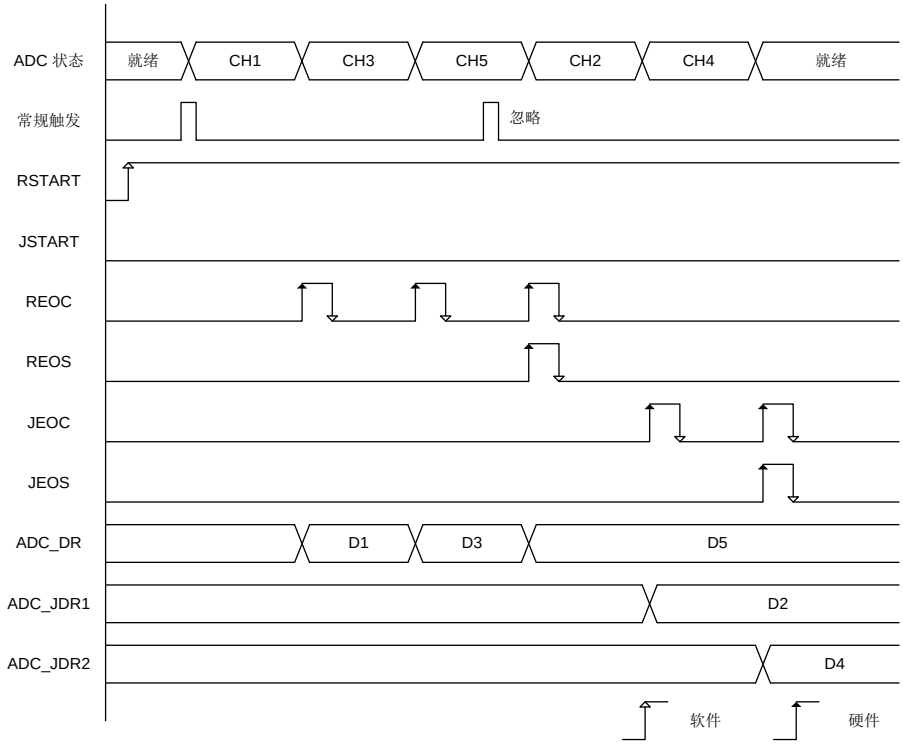


图 14-18 自动插入模式，单次转换模式

注: 1. 标准转换为硬件触发，标准序列长度 3，标准通道为 1、3、5。 2. 插入序列长度 2，插入通道为 2、4。 3. JAUTO = 1，RCONT = 0。

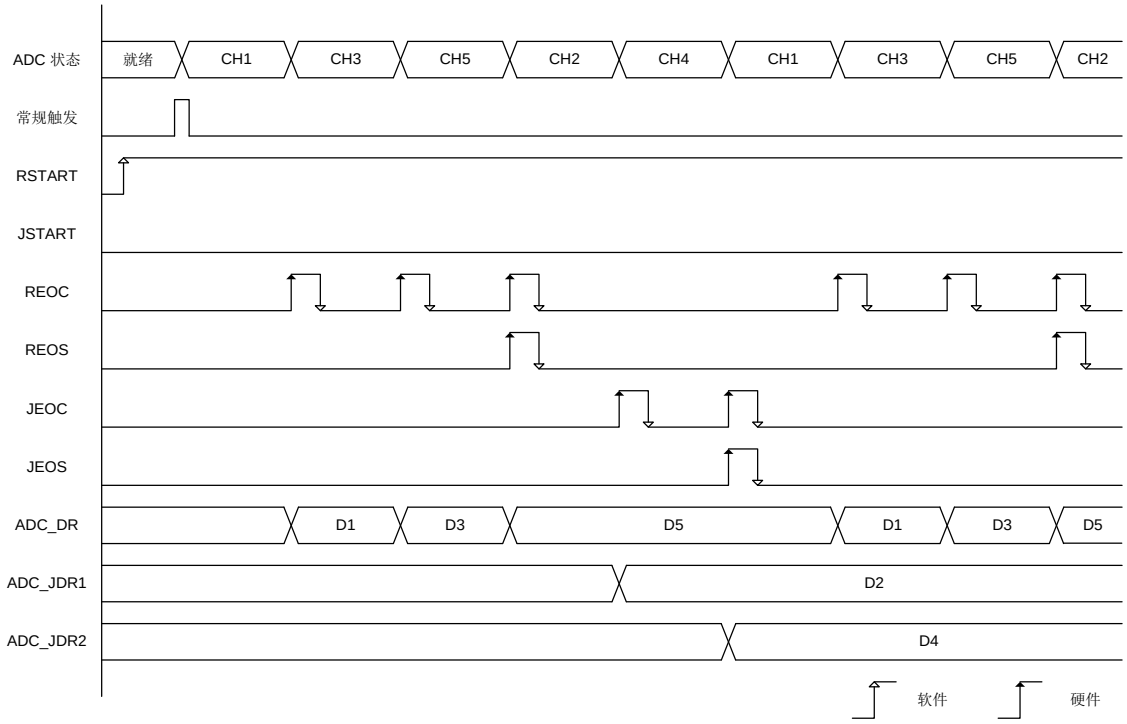


图 14-19　自动插入模式，连续转换模式

注:

1. 标准转换为硬件触发，标准序列长度 3，标准通道为 1、3、5。
2. 插入序列长度 2，插入通道为 2、4。
3. JAUTO = 1，RCONT = 1。

14. 4. 13 非连续模式 (DISCEN、DISCNUM、JDISCEN)

14. 4. 13. 1　标准通道

若要使用标准通道的非连续模式，需要设定寄存器 **ADC_CFG1**.DISCEN 为 1。

在非连续模式下，寄存器 **ADC_SQR1**.RSQLEN 所定义的标准序列长度会被分成 n 个子序列(n ≤8)，数值 n 可通过寄存器 **ADC_CFG1**.DISCNUM[2:0]设定。

当所有子序列完成后，REOS 标志位会被产生，下一次新的触发将从第一组的子序列重新开始转换。

注: 若标准触发设定为软件触发，使 ADC_CON.RSTAR 启动转换，在每个子序列完成后，硬件会自动将 RSTART 清除为 0。

范例 ：

- 寄存器 **ADC_CFG1**.DISCEN 设为 1，并通过寄存器 **ADC_CFG1**.DISCNUM[2:0]设定子序列的数量(n ∈ 8)。例如，若 DISCEN 设为 1，DISCNUM[2:0]设为 2 (n = 3)，且通道序列长度为 9，则选择的通道依序为 1、2、3、6、7、8、9、10、11(通道的顺序可以通过寄存器 **ADC_SQRx** 设定)。

- ◇ 在第一次触发时，会依序转换通道 1、2、3，每个通道转换完成后会发生 REOC 事件。
- ◇ 在第二次触发时，会依序转换通道 6、7、8，每个通道转换完成后会发生 REOC 事件。
- ◇ 在第三次触发时，会依序转换通道 9、10、11，每个通道转换完成后会发生 REOC 事件。当通道 11 的转换完成后会发生 REOS 事件。
- ◇ 在第四次触发时，序列会重新开始，依序转换通道 1、2、3，每个通道转换完成后会发生 REOC 事件。

◆ **ADC_CFG1**.DISCEN 设为 0，每次触发都会完成整个序列的通道转换。通道序列长度为 9，则选择的通道依序为 1、2、3、6、7、8、9、10、11(通道的顺序可以通过寄存器 **ADC_SQRx** 设定)。

- ◇ 每次触发都会依序转换通道 1、2、3、6、7、8、9、10、11，并在通道 11 的转换完成时发生 REOS 事件。
- ◇ 每次的触发将完成整个序列的通道转换。

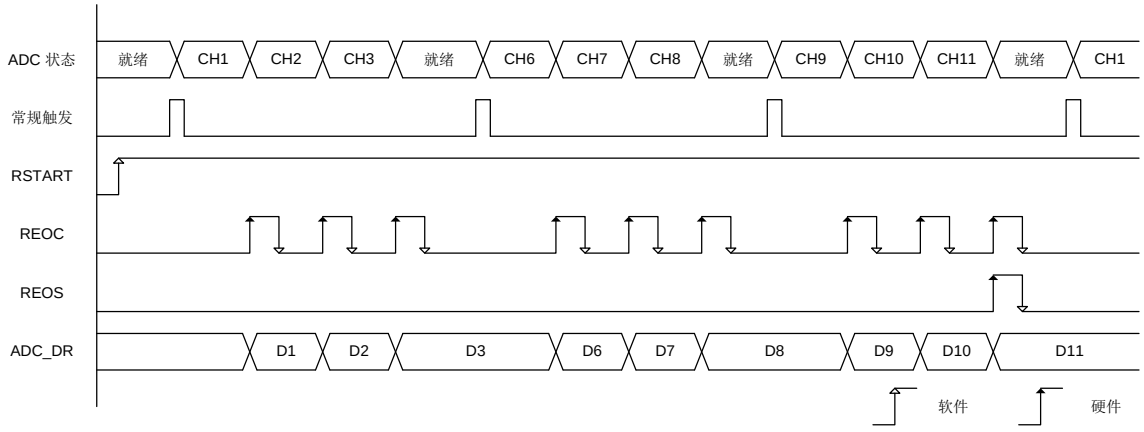


图 14-20　标准转换非连续模式，使用硬件触发

注:

1. 标准转换为硬件触发，标准序列长度 9，标准通道为 1、2、3、6、7、8、9、10、11。
2. 子序列长度 3。
3. DISCEN = 1，DISCNUM[2:0] = 2。

在非连续模式下，如果最后一个子序列的数量少于 DISCNUM[2:0]所设定的值，那么不会发生翻转的现象。

范例:

- 寄存器 **ADC_CFG1**.DISCEN 设为 1，寄存器 **ADC_CFG1**.DISCNUM[2:0]设为 2 (n = 3)，通道序列长度为 8，选择的通道依序为 1、2、3、6、7、8、9、10 (通道的顺序可以通过寄存器 **ADC_SQRx** 设定)。
  - ◇ 第一次触发时，会转换通道 1、2、3，每当一个通道转换完成时都会发生 REOC 事件。
  - ◇ 第二次触发时，会转换通道 6、7、8，每当一个通道转换完成时都会发生 REOC 事件。
  - ◇ 第三次触发时，会转换通道 9、10，每当一个通道转换完成时都会发生 REOC 事件。当通道 10 的转换完成后，会发生 REOS 事件。(注意：最后一个子序列只会将剩下的通道转换完成，而不会翻转回序列开头。)
  - ◇ 第四次触发时，序列将重新开始，并会转换通道 1、2、3。每当一个通道转换完成时都会发生 REOC 事件。

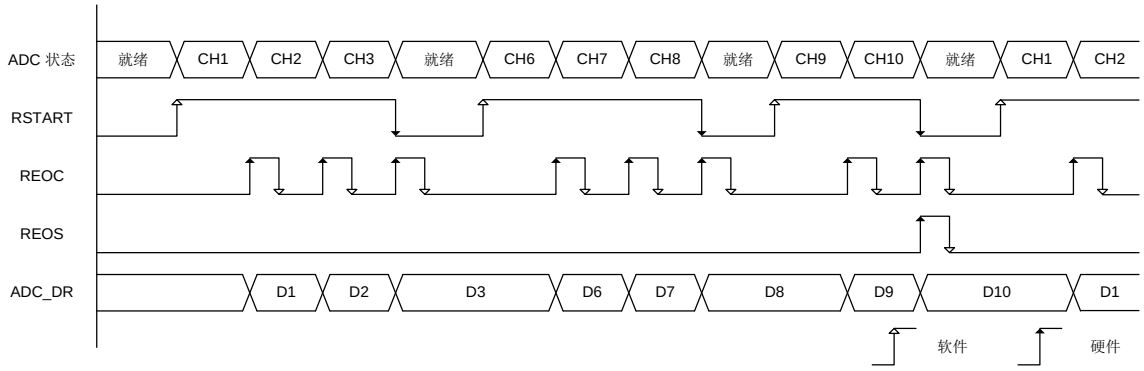


图 14-21　标准转换非连续模式，使用软件触发

注:

1. 标准转换为软件触发，标准序列长度 8，标准通道为 1、2、3、6、7、8、9、10。
2. 子序列长度 3。
3. DISCEN = 1，DISCNUM[2:0] = 2。
4. 不能够同时存在非连续以及连续模式，所以禁止同时设定 DISCEN = 1 以及 RCONT = 1。

14. 4. 13. 2 插入通道

若要启用插入转换的非连续模式，请通过设定寄存器 **ADC_CFG1**.JDISCEN = 1。

插入序列的通道顺序与长度可通过 **ADC_JSQR** 寄存器设定。每次外部触发都会启动一次通道转换，并在一个通道完成后自动切换至下一个通道。若将子序列长度设定为 1，与插入转换的非连续模式的运行动作相同。

范例:

- 插入序列长度为 3，需要转换的通道为 1,2,3:
  - 第一次触发，转换通道 1。转换完成后会发生 JEOC。
  - 第二次触发，转换通道 2。转换完成后会发生 JEOC。
  - 第三次触发，转换通道 3。转换完成后会发生 JEOC 以及 JEOS。(插入转换的最后一个子序列完成后，会发生 JEOS 事件)
  - 第四次触发，序列重新开始，转换通道 1。

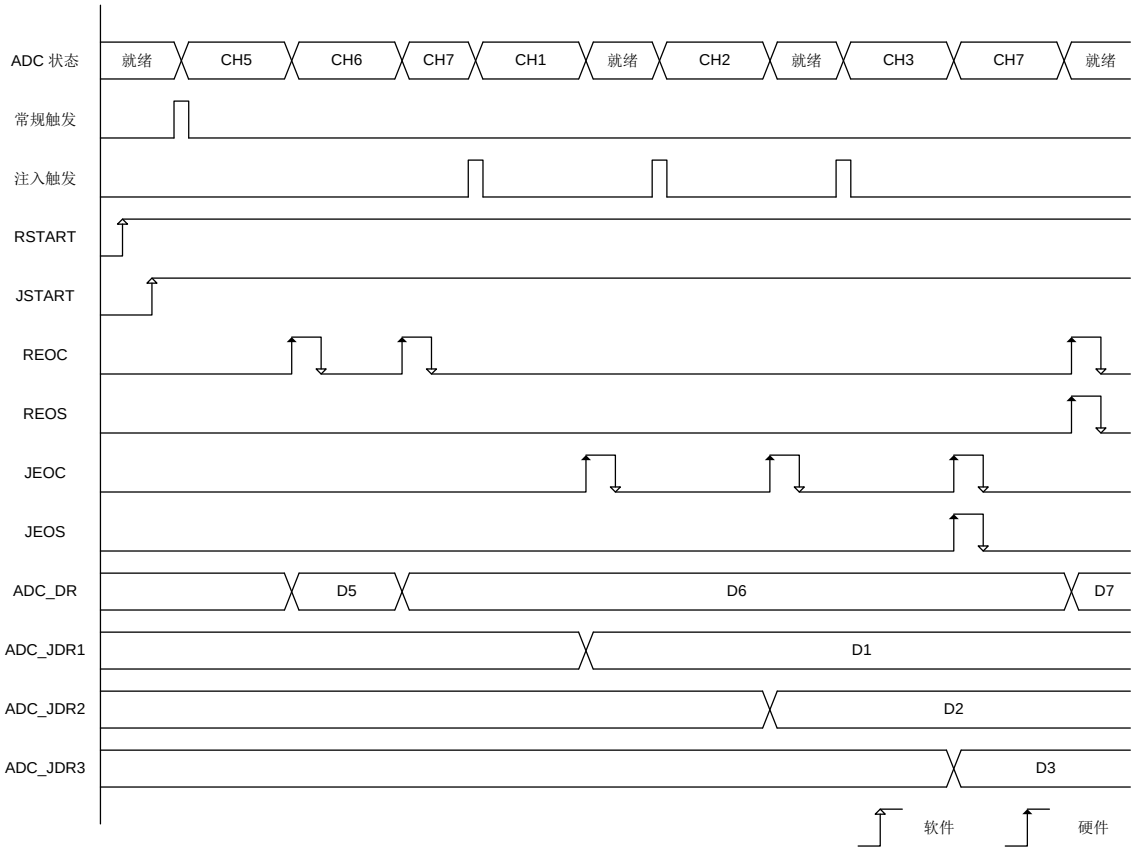


图 14-22 插入转换非连续模式，使用硬件触发

注:

1. 在标准转换的硬件触发模式下，设定标准序列长度为 3，需要转换的通道为 5、6、7。
2. 在插入转换的硬件触发模式下，设定插入序列长度为 3，需要转换的通道为 1、2、3。
3. 启动非连续模式，设定 JDISCEN = 1。
4. 不能够同时存在自动插入以及非连续模式。

14. 4. 14 插入转换的队列

设定 **ADC_CFG1**.JQEN = 1，启用插入转换队列，可支持两个插入序列的设定。此模式的插入转换仅能使用硬件触发。

队列的内容包括以下设定:

- 插入触发的配置:使用寄存器 **ADC_JSQR** 的 JEXTEN[1:0]和 JEXTSEL[4:0]设定插入触发源。
- 插入序列的配置:使用寄存器 **ADC_JSQR** 的 JSQx[4:0]和 JSQLEN[1:0]设定插入序列的通道和序列长度。

使用插入队列模式下，可以将两组插入序列的设定值写入寄存器 **ADC_JSQR** 的两个缓冲区中，以实现两组插入序列的参数设定。

- 可以随时写入 **ADC_JSQR**，无论是否有正在进行中的插入转换。
- 每次写入 **ADC_JSQR** 的设定值将会被储存在队列中。
- 如果队列一开始是空的，第一次写入 **ADC_JSQR** 的设定值将会更新队列，然后等待插入触发。
- 当插入序列完成后，会将缓冲区中储存的设定值更新到队列中。下一个插入转换将由新的队列提供设定值。
- 如果插入队列已满，再次写入 **ADC_JSQR** 将导致队列益出(JQOVF)。当队列益出时，写入 **ADC_JSQR** 的设定值将无法更新到队列中，队列内容将不会改变。如果已设置 **ADC_IER**.JQOVF，则会产生中断。
- 如果在队列为空的情况下设定了寄存器 **ADC_CFG1**.JQM，会有两种行为:
  - ◇ 当 JQM = 0 时，只有在启动 ADC 时才会出现队列为空的情况。在后续操作中，不允许队列为空的情况，将保留上一个有效的队列内容，并设定接下来的插入序列。
  - ◇ 当 JQM = 1 时，队列会处于空状态。如果队列为空，硬件触发将被禁用。只有写入新的 **ADC_JSQR** 设定值，才能接受后续的硬件触发。
- 读取寄存器 **ADC_JSQR** 的值会显示当前队列的内容。如果队列为空，则读取的值为 0x0000。
- 当设置了 JSTP 和 ADCDIS 时，队列将被清空。
  - ◇ 如果 JQM = 0，则队列将保留上一个有效的队列内容。
  - ◇ 如果 JQM = 1，则队列将被清空并忽略触发。

注: 当配置为不连续模式下，只有插入序列的最后一次触发才会更改队列的内容。

- 在非连续模式下，如果设定了两个队列，例如队列 1 包含通道 1、2、3 的插入序列，队列 2 包含通道 4、5、6 的插入序列，那么触发时会按照下列方式进行:
  - ◇ 第一次触发，使用队列 1 的设定值，转换通道 1。
  - ◇ 第二次触发，使用队列 1 的设定值，转换通道 2。
  - ◇ 第三次触发，使用队列 1 的设定值，转换通道 3。

◇ 第四次触发，使用队列 2 的设定值，转换通道 4。
◇ 第五次触发，使用队列 2 的设定值，转换通道 5。
◇ 第六次触发，使用队列 2 的设定值，转换通道 6。

14. 4. 14. 1 更改插入序列设定的行为

以下的图将示意队列在插入转换期间的操作行为。

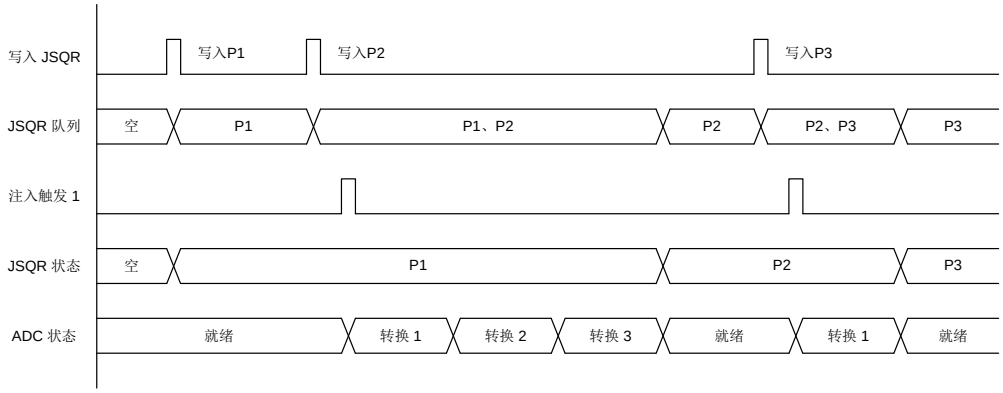


图 14-23 JSQR 队列示意图(序列更改)

注:
队列 P1，插入序列长度为 3，使用硬件触发来源 1。
队列 P2，插入序列长度为 1，使用硬件触发来源 1。
队列 P3，插入序列长度为 4，使用硬件触发来源 1。

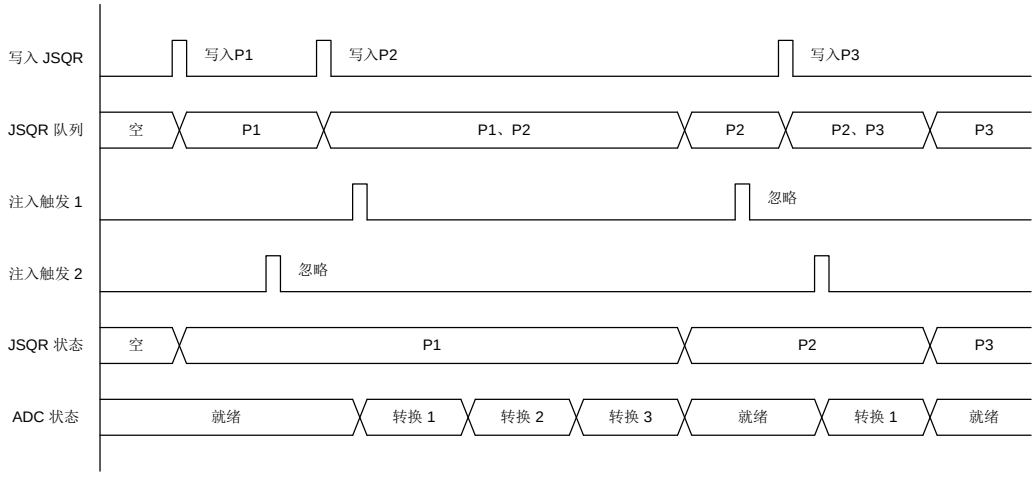


图 14-24 JSQR 队列示意图(触发更改)

注:
队列 P1，插入序列长度为 3，使用硬件触发来源 1。
队列 P2，插入序列长度为 1，使用硬件触发来源 2。
队列 P3，插入序列长度为 4，使用硬件触发来源 1。

14. 4. 14. 2　插入队列溢出的行为

以下的图将示意队列发生溢出的操作行为。

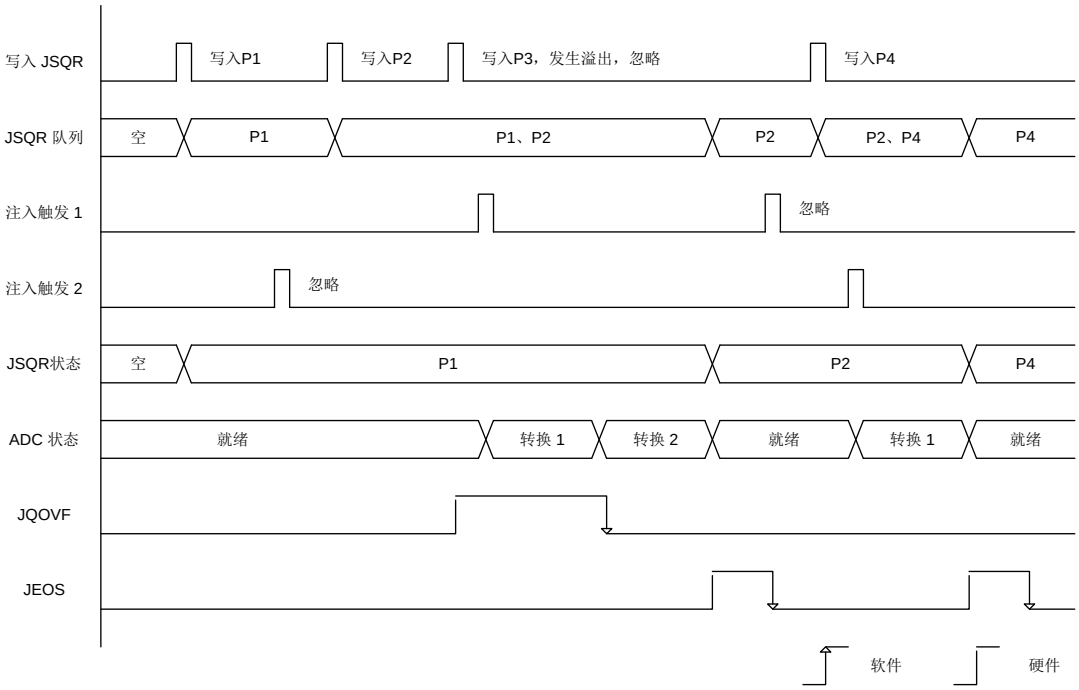


图 14-25　在插入转换前，发生 JSQR 队列溢出

注:
队列 P1，插入序列长度为 2，使用硬件触发来源 1。
队列 P2，插入序列长度为 1，使用硬件触发来源 2。
队列 P3，插入序列长度为 4，使用硬件触发来源 1。
队列 P4，插入序列长度为 3，使用硬件触发来源 1。

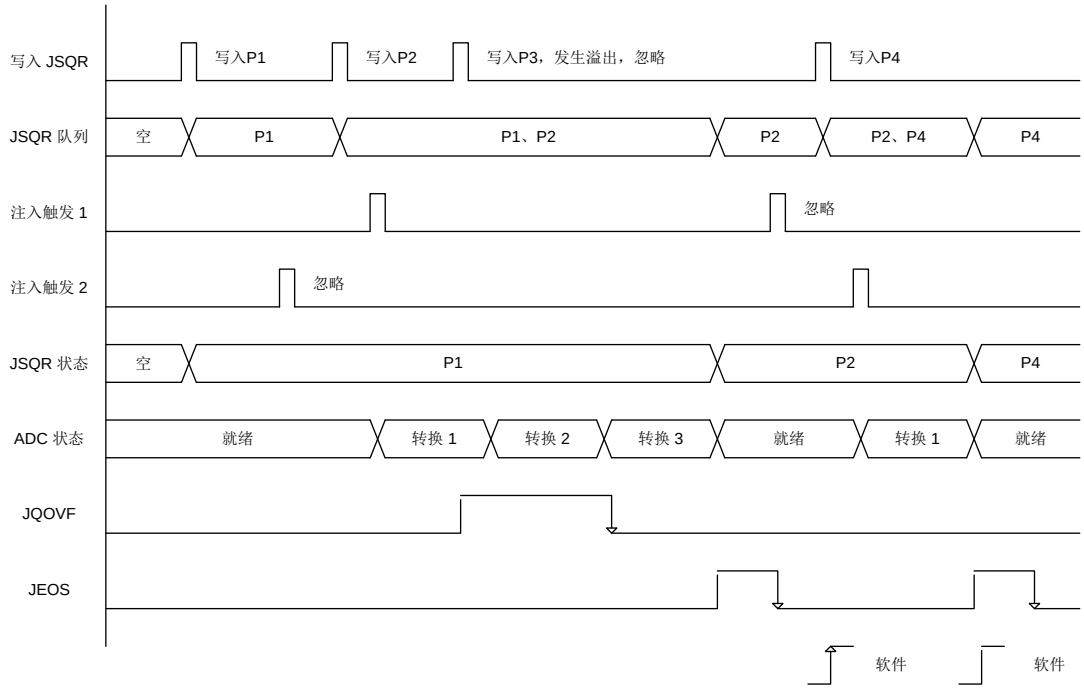


图 14-26　在插入转换后，发生 JSQR 队列溢出

注:
队列 P1，插入序列长度为 3，使用硬件触发来源 1。
队列 P2，插入序列长度为 1，使用硬件触发来源 2。
队列 P3，插入序列长度为 4，使用硬件触发来源 1。
队列 P4，插入序列长度为 3，使用硬件触发来源 1。

队列溢出是指在插入序列的设定长度内，新的插入序列仍然被写入，此时原本队列中的部分数据将会被覆盖或者丢失。为了避免发生队列溢出，需要进行管理。

- 监控 JQOVF 标志位: 当写入新的插入序列时，如果发生 JQOVF 标志位，表示写入的设定值被忽略，因为插入队列已经满了。此时需要及时处理，避免数据丢失。
- 确保 JEOS 的发生: 在插入序列完成后，ADC 模块会发出 JEOS 标志位。可以利用 JEOS 标志位，确保插入序列已经完成，再写入新的插入序列，避免队列溢出。

14. 4. 14. 3　插入队列为空的操作行为

以下的图示意，队列为空在 **ADC_CFG1**.JQM 的不同设定下的行为模式。

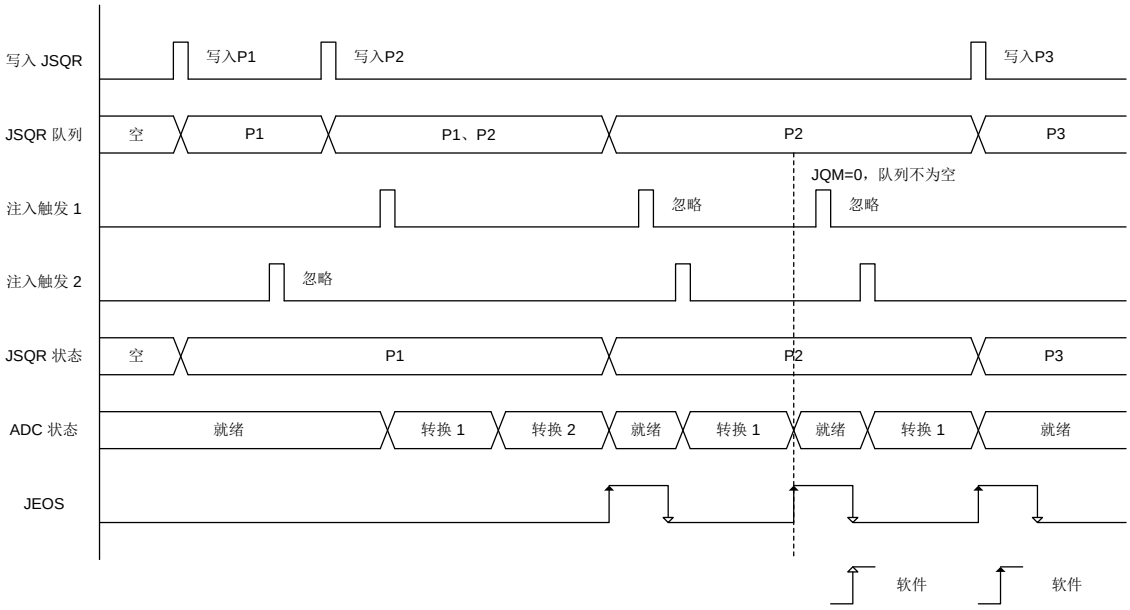


图 14-27　JSQR 队列为空，JQM=0 的示意图

注 1:

队列 P1，插入序列长度为 3，使用硬件触发来源 1。

队列 P2，插入序列长度为 1，使用硬件触发来源 2。

队列 P3，插入序列长度为 4，使用硬件触发来源 1。

队列 P4，插入序列长度为 3，使用硬件触发来源 1。

注 2:

将 P3 写入 ADC_JSQR，队列会立即改变。但由于内部硬件同步的因素，如果在写入 P3 的前后发生插入触发，可能会依照前面队列 2 的设定启动插入序列转换。为了避免这个情况，用户必须确保写入新的设定值时，不会有 ADC 触发的发生。

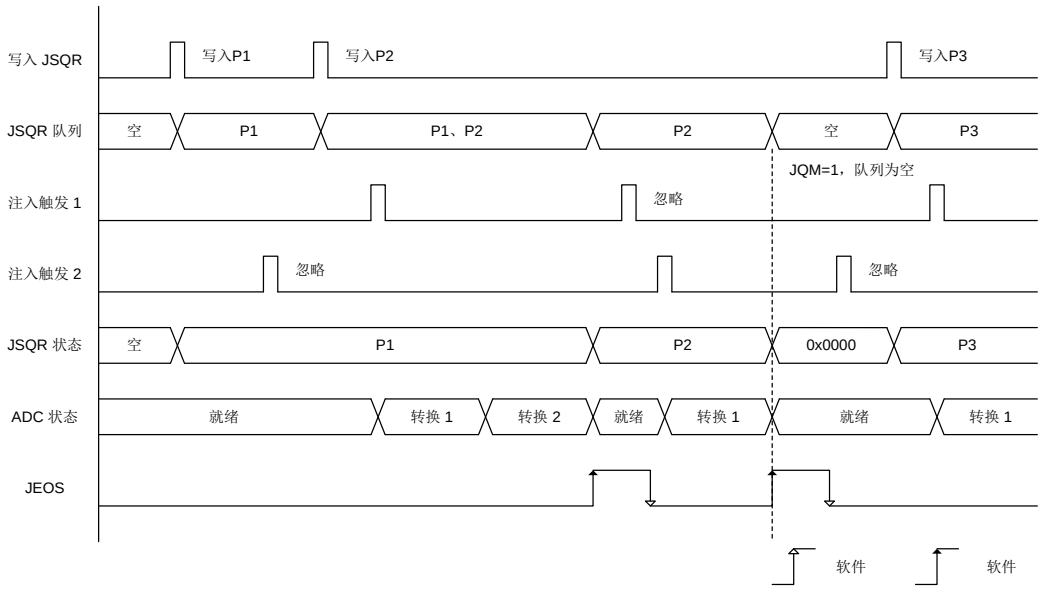


图 14-28　JSQR 队列为空，JQM=1 的示意图

注:
队列 P1，插入序列长度为 2，使用硬件触发来源 1。
队列 P2，插入序列长度为 1，使用硬件触发来源 2。
队列 P3，插入序列长度为 3，使用硬件触发来源 1。

14. 4. 14. 4　插入队列清空的操作行为

以下图示意队列被清空的各种情况。

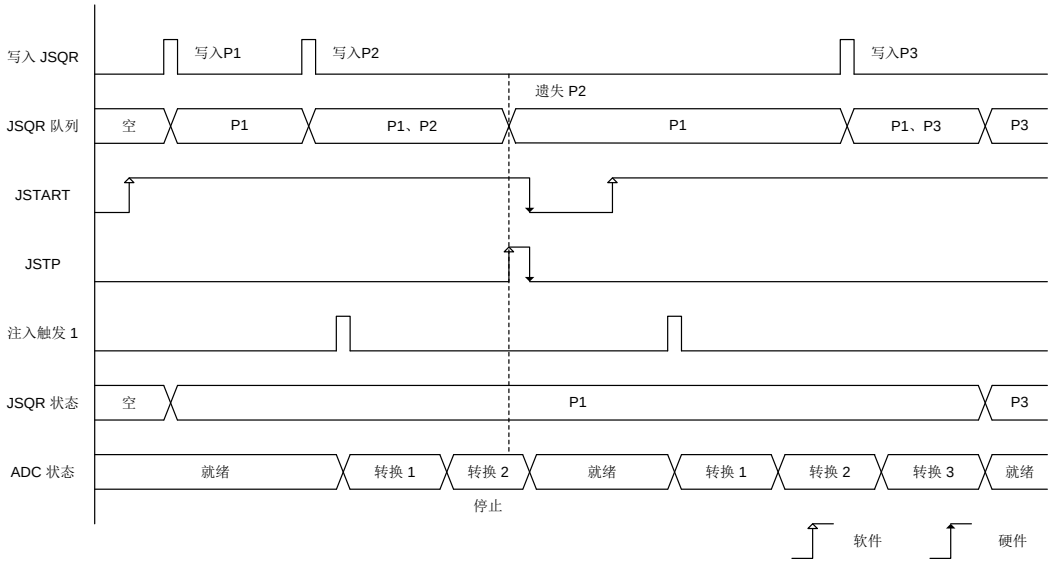


图 14-29　JQM=0，使用 JSTP 暂停正在进行的转换，在写入新的设定值前发生触发

注:
队列 P1，插入序列长度为 3，使用硬件触发来源 1。
队列 P2，插入序列长度为 1，使用硬件触发来源 1。
队列 P3，插入序列长度为 3，使用硬件触发来源 1。

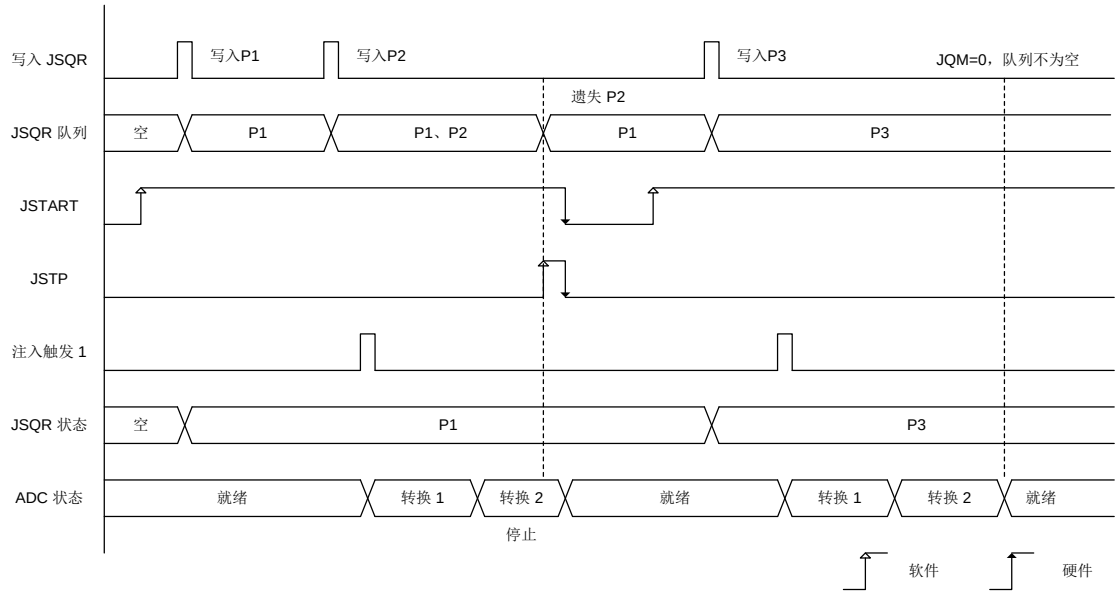


图 14-30　JQM=0，使用 JSTP 暂停正在进行的转换，发生触发前写入新的设定值

注:
队列 P1，插入序列长度为 3，使用硬件触发来源 1。
队列 P2，插入序列长度为 1，使用硬件触发来源 1。
队列 P3，插入序列长度为 2，使用硬件触发来源 1。

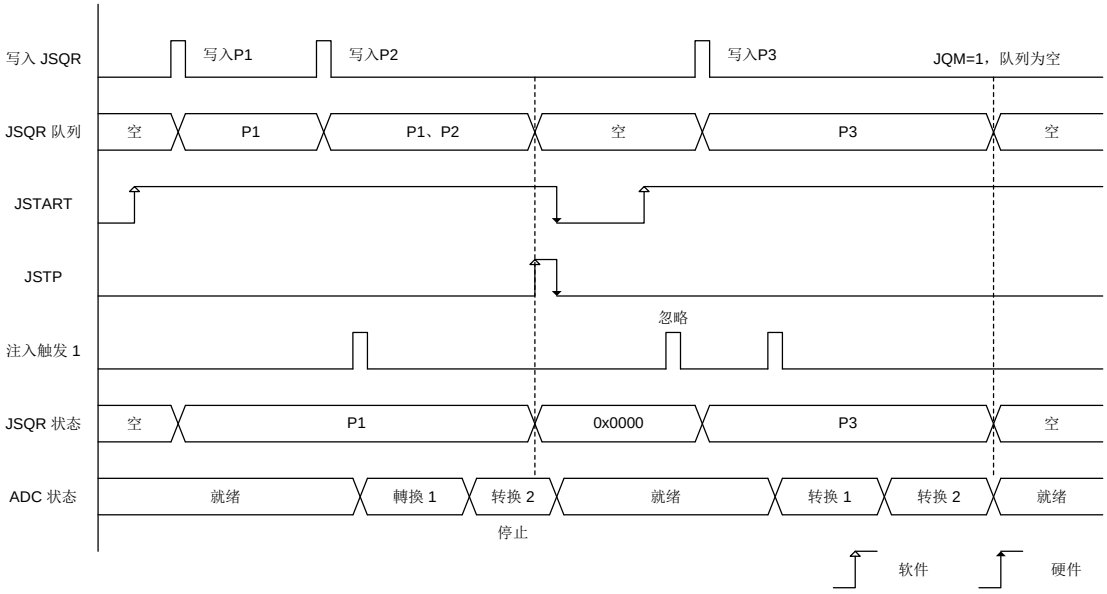


图 14-31　JQM=1，使用 JSTP 暂停正在进行的转换

注:
队列 P1，插入序列长度为 3，使用硬件触发来源 1。
队列 P2，插入序列长度为 1，使用硬件触发来源 1。
队列 P3，插入序列长度为 2，使用硬件触发来源 1。

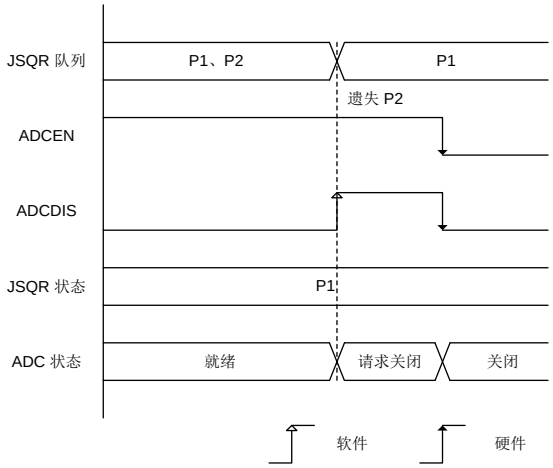


图 14-32　JQM=0，使用 ADCDIS 停止 ADC

注:
队列 P1，插入序列长度为 1，使用硬件触发来源 1。
队列 P2，插入序列长度为 1，使用硬件触发来源 1。

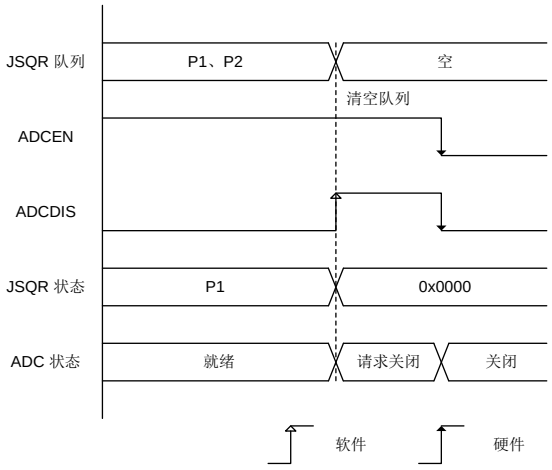


图 14-33　JQM=1，使用 ADCDIS 停止 ADC

注:
队列 P1，插入序列长度为 1，使用硬件触发来源 1。
队列 P2，插入序列长度为 1，使用硬件触发来源 1。

在队列为空的情况下启动 ADC，需要按照下列步骤执行，以避免 ADC 无法获得第一组队列。这些步骤仅适用于 JQM=0 的情况:

- 写入 JSQR，JEXTEN[1:0]不能等于 0，以免使用软件触发。
- 设置 JSTART = 1。
- 设置 JSTP = 1。
- 等待 JSTART = 0 与 JSTP = 0。
- 设置 JSTART = 1。

可以通过清除 **ADC_CFG1**.JQEN=0，关闭插入队列的功能。

14. 4. 15 可配置的数据分辨率

为了加速转换速度，可以通过调整 ADC 数据的分辨率。使用 ADC 寄存器 **ADC_CFG1**.RESOL 可以选择 12 位、10 位、8 位或 6 位分辨率。

当应用对于 ADC 数据的精度要求不高时，可以考虑使用较低的数据分辨率以减少转换时间，这可参考下表公式。

RESOL	T_{SAR} (时钟周期)	T_{CONV} (时钟周期) (采样时间 6.5 时钟周期)
12	13.5 时钟周期	20 时钟周期
10	11.5 时钟周期	18 时钟周期
8	9.5 时钟周期	16 时钟周期
6	7.5 时钟周期	14 时钟周期

表 14-6 T_{SAR} 与数据分辨率的关系

14. 4. 16 转换结束、采样周期结束 (REOC、JEOC、EOSMP)

当每个标准转换和插入转换完成时，ADC 会提供 REOC 和 JEOC 的信息。

标准转换的数据将在 REOC 发生后存储在 **ADC_DR** 中。可以通过设置 **ADC_IER**.REOC 来开启中断。REOC 可以通过设置 **ADC_ICR**.REOC 或读取 **ADC_DR** 来清除状态。

插入转换的数据将在 JEOC 发生后存储在相应的 **ADC_JDRx** 中。可以通过设置 **ADC_IER**.JEOC 来开启中断。JEOC 可以通过设置 **ADC_ICR**.JEOC 或读取相应的 **ADC_JDRx** 来清除状态。

采样周期结束后，会产生 EOSMP 状态，可以通过设置 **ADC_IER**.EOSMP 来开启中断。EOSMP 可以通过设置 **ADC_ICR**.EOSMP 来清除状态。

14. 4. 17 序列转换结束 (REOS、JEOS)

当每个标准转换序列以及插入转换序列完成后，ADC 会发出 REOS 和 JEOS 的信息。

设定 **ADC_IER**.REOS 以开启中断。REOS 可以通过设置 **ADC_ICR**.REOS 来清除状态。

设定 **ADC_IER**.JEOS 以开启中断。JEOS 可以通过设置 **ADC_ICR**.JEOS 来清除状态。

14. 4. 18 数据管理

14. 4. 18. 1 数据以及对齐

每当完成一个标准转换序列并产生 REOS 事件后，最后一个转换的结果会存储在 **ADC_DR** 中。

每当完成一个插入转换序列并产生 JEOS 事件后，最后一个转换的结果会存储在对应的 **ADC_JDRx** 中。

寄存器 **ADC_CFG1** 的 ALIGN 可以用来设定存储数据的对齐方式，可以选择右对齐或左对齐，

具体选择右对齐的方式可以参考 Figure 34: , “右对齐 (偏移关闭、无符号数)”、Figure 35: , “右对齐 (偏移开启、有符号数)”，而左对齐的方式可以参考 Figure 36: , “左对齐 (偏移关闭、无符号数)”、Figure 37: , “左对齐 (偏移开启、有符号数)”。

注:
当选择了 6 位的数据分辨率以及左对齐的存储方式，数据将会按字节对齐，具体可以参考 Figure 36: , “左对齐 (偏移关闭、无符号数)”、Figure 37: , “左对齐 (偏移开启、有符号数)”。在使用过采样模式时，禁止使用左对齐的存储方式。当 ADC_CFG2 中的 ROVSEN 或 JOVSEN 被设置后，硬件会忽略 ADC_CFG1.ALIGN 的设置，并且只能提供右对齐的存储方式。

14. 4. 18. 2 偏移

设定 **ADC_OFSTx**.OFFSETEN=1 后，可以开启对 4 个通道(y=1,2,3,4)的偏移设定。偏移的运算值是基于寄存器 **ADC_OFSTx**.OFFSET 的设定。当进行偏移运算时，运算的结果有可能为负数，所以储存的数据会延展有符号数 S。

注: 在过采样模式下，是无法使用偏移运算。当 ADC_CFG2 的 ROVSEN 或是 JOVSEN 设定后，硬件会忽略 ADC_OFSTx.OFFSETEN 的设定。

在偏移补偿中，对于每个通道，都会有一个独立的偏移值，用来校正 ADC 转换过程中产生的固定偏移值。偏移值的计算可以通过以下步骤进行:

1. 设定 ADC 的数据分辨率和对齐方式。
2. 清除 OFFSET 值 (即将 OFFSET 设置为 0)。
3. 进行一次 ADC 转换。
4. 读取 **ADC_DR** 和 **ADC_OFRy** (y 为通道号)的值。
5. 计算 OFFSET 值，偏移值= $2^{(n-1)}$ - **ADC_DR** - **ADC_OFRy**，其中 n 是数据分辨率。

计算偏移值的过程中，需要注意数据分辨率和对齐方式。例如 :

- 如果数据分辨率为 12 位，并且使用右对齐的方式存放数据，则偏移值的计算公式为偏移值 = 2048 - **ADC_DR** - **ADC_OFRy**。
- 如果数据分辨率为 6 位，并且使用左对齐的方式存放数据，则偏移值的计算公式为偏移值 = 32 - **ADC_DR** - **ADC_OFRy**。

下表描述偏移补偿与数据分辨率的关系比较。

数据分辨率	转换后的数据减去偏移量		结果	批注
	转换后的数据左对齐	偏移量		
12 位	DATA[11:0]	OFFSET[11:0]	12 位有符号数	-
10 位	DATA[9:0] 00	OFFSET[11:0]	10 位有符号数	OFFSET[1:0]需要设定为 00
8 位	DATA[7:0] 0000	OFFSET[11:0]	8 位有符号数	OFFSET[3:0]需要设定为 0000
6 位	DATA[5:0] 000000	OFFSET[11:0]	6 位有符号数	OFFSET[5:0] 需要设定为 000000

表 14-7 偏移计算与数据分辨率

读取 **ADC_DR** 或是 **ADC_JDR**，对应的通道是否有开启偏移的计算，存放的数据会有不同的摆放方式。

- 对应的通道有开启偏移计算，读取的数据将会为有符号数。
- 对应的通道关闭偏移计算，读取的数据将会为无符号数。

图 14-34、图 14-35、图 14-36、图 14-37 显示数据的对其方式以及有符号数或无符号数。

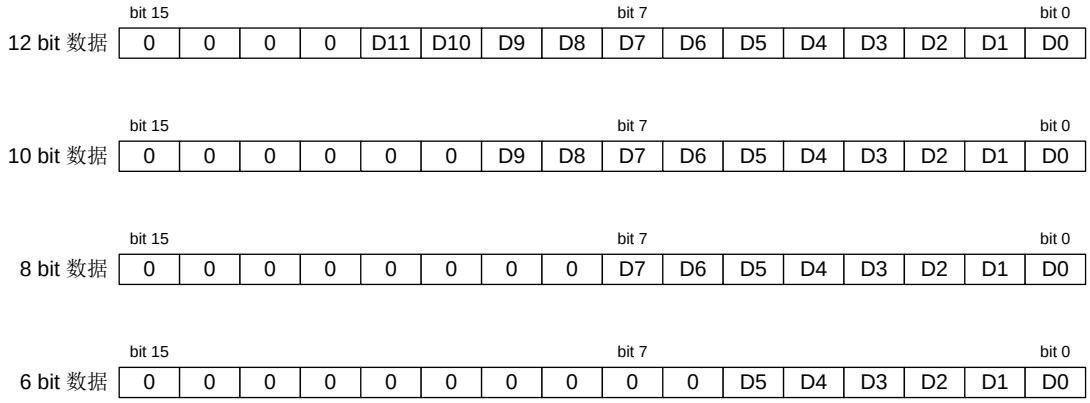


图 14-34　右对齐(偏移关闭、无符号数)

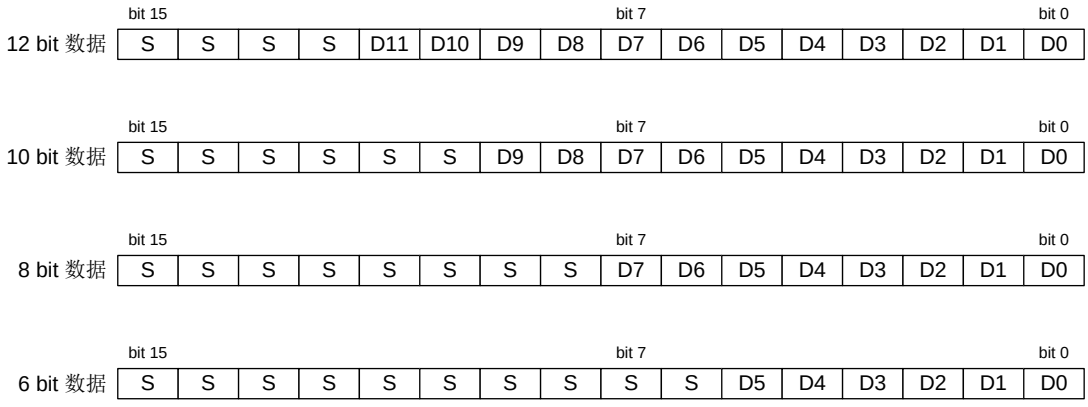


图 14-35　右对齐(偏移开启、有符号数)

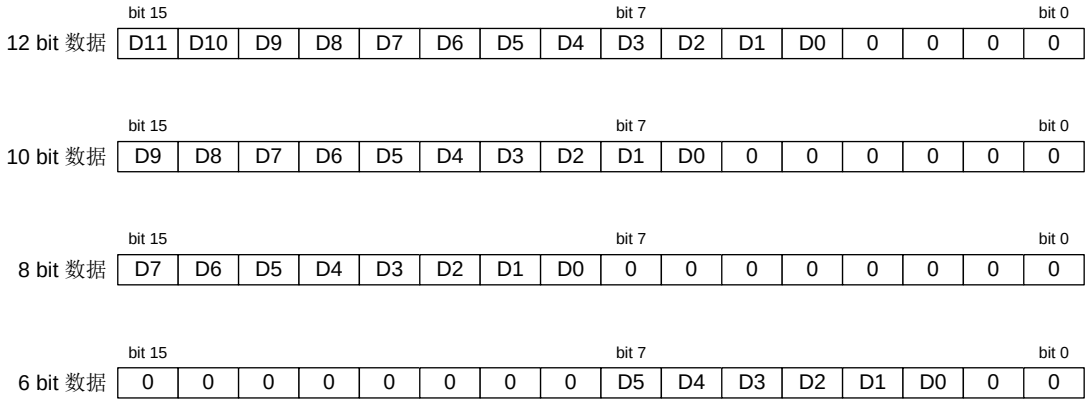


图 14-36　左对齐(偏移关闭、无符号数)

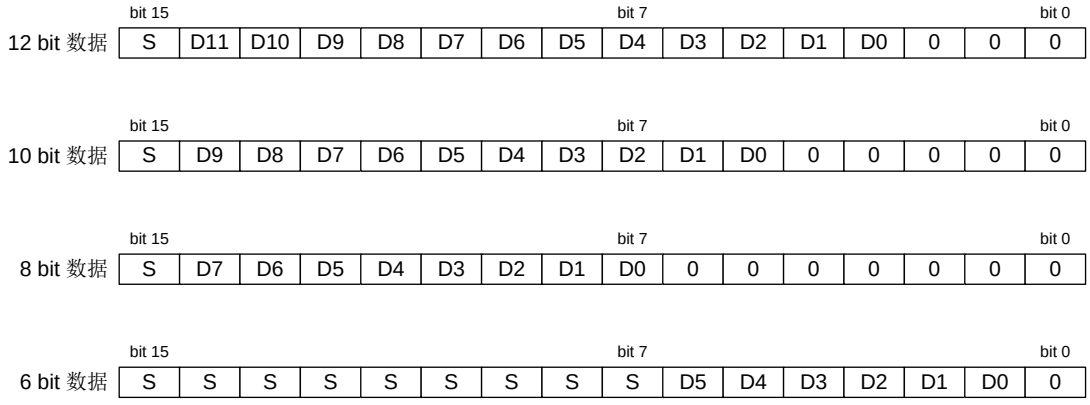


图 14-37　左对齐(偏移开启、有符号数)

14. 4. 18. 3　增益补偿

当设定了 **ADC_CFG2**.GCOMP，转换后的数据会进行增益补偿。该补偿的公式为：

$$DATA = DATA(adc\ result) \times (GCOMP\ COEFF) / 4096$$

其中，**ADC_GCOMP**.COEF 可以被设定为 0 到 16383 之间，对应的增益补偿因子范围为 0 到 3.999756。

在储存计算完的数据到寄存器之前，会先对 LSB - 1 的数据进行四舍五入。

在过采样模式下，也可以开启增益补偿功能，不过此时会在数据进行累加后右移再进行增益计算，每次过采样转换完后，只会经过一次增益计算。

14. 4. 18. 4　偏移补偿

如果启用了通道偏移补偿功能，且设定了 **ADC_OFSTx**.SATEN，那么数据将会被视为无符号数。在此情况下，所有经过计算的数据将会饱和至 0x000。如果通道偏移为正值，并设定了 **ADC_OFSTx**.POSEN，那么计算完后的数据将会饱和至 0xFFF(根据数据分辨率)。

模拟看门狗使用经过偏移和增益补偿的数据进行比较，而其比较基准为无符号数。为了确保看门狗的计算正确，需要设定 **ADC_OFSTx**.SATEN，将经过补偿后的数据视为无符号数进行比较。

14. 4. 18. 5 ADC 数据溢出

当标准转换产生新的数据后，如果之前的数据还未被读出，将会触发 ROVR 标志位。如果在 **ADC_IER**.ROVR 被设定的情况下，将会开启中断。ROVR 和 REOC 标志位将会同时产生。

当 ROVR 标志位触发后，ADC 可以持续进行转换。藉由写入 1 到标志位地址，可以清除 ROVR 标志位。

可以通过设定 **ADC_CFG1**.OVRRMOD 来决定 ROVR 触发后，新数据是否会覆盖早先的数据：

- 当 OVRRMOD=0 时，ADC 将会保存寄存器的数据，以免被覆盖。如果 ROVR 标志位持续为 1，后续转换产生的数据都会被丢弃。
- 当 OVRRMOD=1 时，ADC 会覆盖寄存器的数据，早先的数据将会遗失。如果 ROVR 标志位持续为 1，新转换产生的数据可以通过读取寄存器来获取。

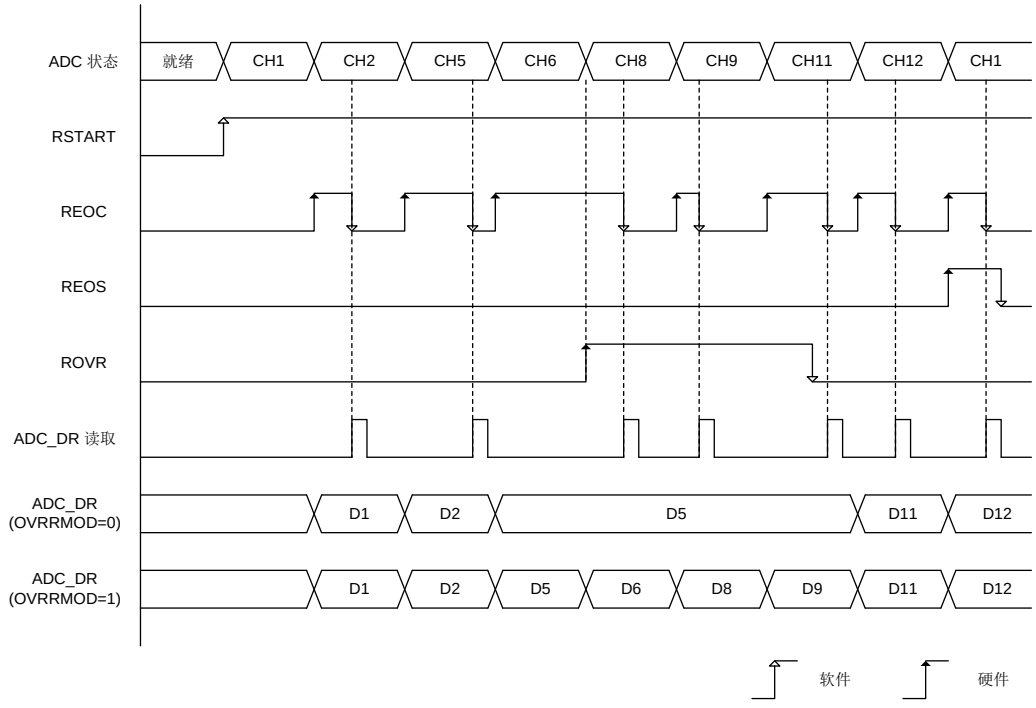


图 14-38 ADC 数据溢出示意图

注 1:

1. 标准转换为硬件触发，标准序列长度 8，标准通道为 1、2、5、6、8、9、11、12。
2. JAUTO = 0，RCONT = 1。

注 2: 由于插入通道拥有四个对应通道的数据寄存器，因此不需要对插入通道进行溢出侦测。

14. 4. 18. 6 不使用DMA的情况下管理转换数据

当转换速度过慢时，可以使用软件来处理转换序列。软件需检查标准通道的 REOC 标志位或状

态，以确定是否能够处理转换数据。一旦 REOC 被设置，表示转换已经完成，标准通道转换数据已经存放到 **ADC_DR** 中。而插入通道只能使用软件处理转换序列的数据，需检查插入通道的 JEOC 标志位或状态，一旦 JEOC 被设置，表示转换已完成，而插入通道的数据会存放至对应的 **ADC_JDRx**。

14. 4. 18. 7 使用DMA管理转换数据

使用 DMA 搬移数据可以防止在下一次写入标准通道转换数据之前，丢失存储在 **ADC_DR** 中的前一笔转换数据。

当设置 **ADC_CFG1**.DMAEN 启用 DMA 模式后，每当标准通道转换完成时就会产生 DMA 传输请求。转换后的数据将由 DMA 从 **ADC_DR** 搬移至软件设定的目的地位置。

由于 DMA 无法及时处理数据搬移请求，会导致数据溢出状态(ROVR=1) 的产生。此时，ADC 将停止产生 DMA 传输请求的信号，并且后续的标准通道转换数据也将不会通过 DMA 进行搬移。这样可以确保内存的数据都是有效的。

无论 **ADC_CFG1**.OVRRMOD 的设置如何，只要 ROVR 发生，就不会产生 DMA 请求。只有在软件清除 ROVR 状态后，DMA 传输请求才能恢复正常。

14. 4. 19 动态低功耗管理

14. 4. 19. 1 自动延迟转换模式 (ATUDLY)

若 ADC 开启自动延迟转换模式，需要设定 **ADC_CFG1**.AUTODLY。这种模式能够简化软件的操作，并且对于低速的时钟应用特别有用。

若 AUTODLY 设为 1，则在启动新的转换前，需要先进行以下处理:

- 对于标准转换，需读取 **ADC_DR** 里的数据或清除 REOC。(可参照图 14-39)
- 对于插入转换，需清除 JEOS。

这种模式能够通过系统读取 ADC 数据，自动调整 ADC 速度。

在自动延迟转换模式下，自动延迟时间会发生在以下情况:

- 对于标准转换，每次通道转换之后。
- 对于插入转换，每个插入序列完成后。(需要注意的是，插入序列内的插入转换并不会有自动延迟时间。)

在自动延迟期间，硬件触发事件会被忽略。在使用软件触发前，也必须先读取数据或清除标志位，才能够触发新的转换。

在标准转换被插入转换中断或接续时，不会有自动延迟区间。

- 如果标准转换的自动延迟区间内发生插入触发，将立即启动插入转换。(可参照图 14-40)
- 当插入序列完成后，系统将恢复到标准转换被中断前的状态。如果当时处于自动延迟区间，则仍需等待数据读取或清除 REOC。(可参照 Figure 43)

在自动插入模式下，必须要等到上一次的插入序列完成并且 JEOS 已被清除后，才能启动下一次的标准转换。这是确保软件能够读走所有转换的数据。(可参照 Figure 44)

在自动延迟转换模式下，如果在标准转换期间或自动延迟区间内发生硬件触发事件，该事件会被忽略。但是如果在插入转换期间发生了标准触发事件，而恢复后的标准转换已经超过自动延迟时间并处于等待触发的阶段，那么该触发事件将被保留，直到插入序列结束后再开始标准转换。(可参照 Figure 41)

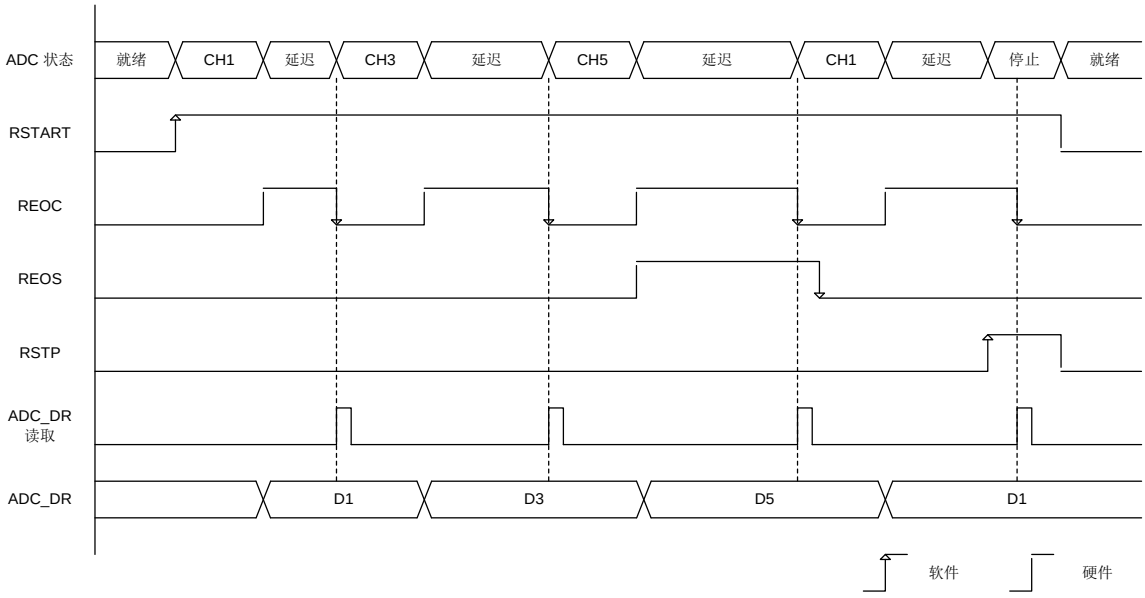


图 14-39　AUTODLY = 1，标准连续转换示意图

注:

1. 标准转换为硬件触发，标准序列长度 3，标准通道为 1、2、3。
2. 插入转换为硬件触发，插入序列长度 2，插入通道为 5、6。
3. AUTODLY = 1，RCONT = 0。

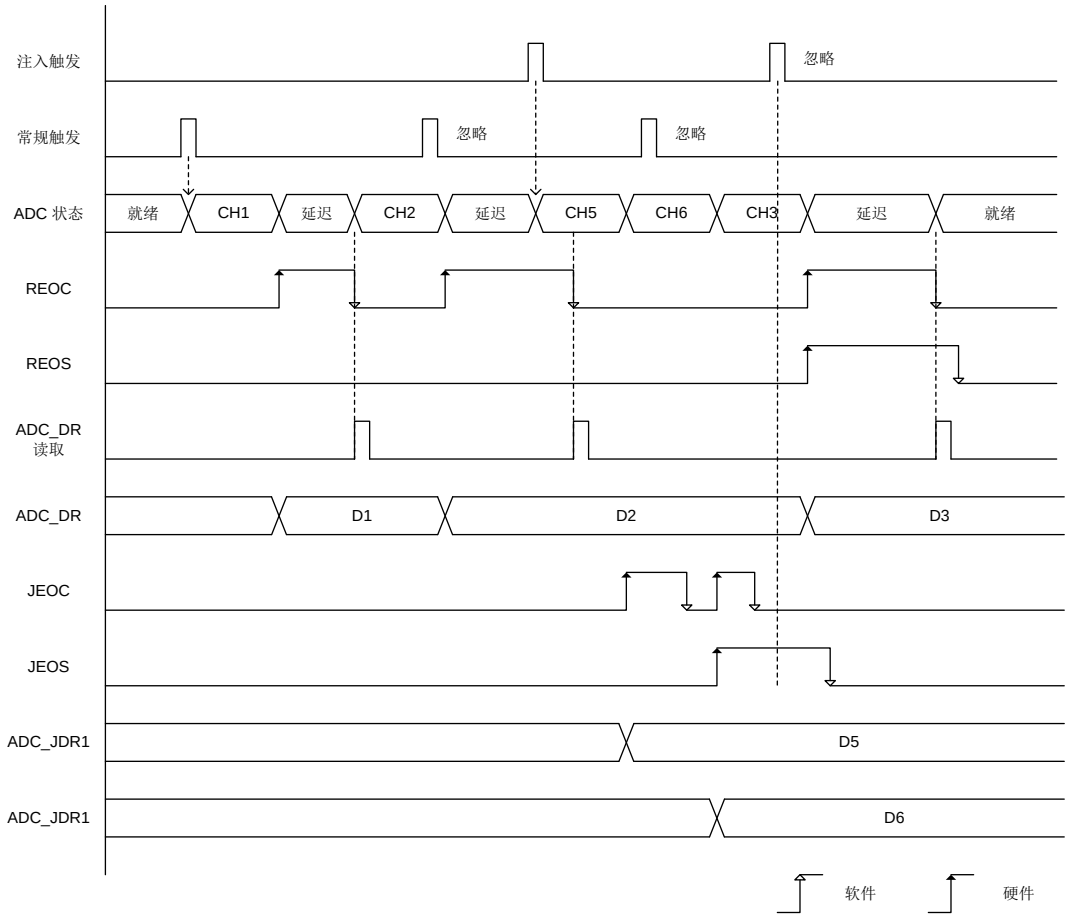


图 14-40　AUTODLY = 1，插入转换中断标准单次转换模式示意图 2

注:

1. 标准转换为硬件触发，标准序列长度 3，标准通道为 1、2、3。
2. 插入转换为硬件触发，插入序列长度 2，插入通道为 5、6。
3. AUTODLY = 1，RCONT = 0。

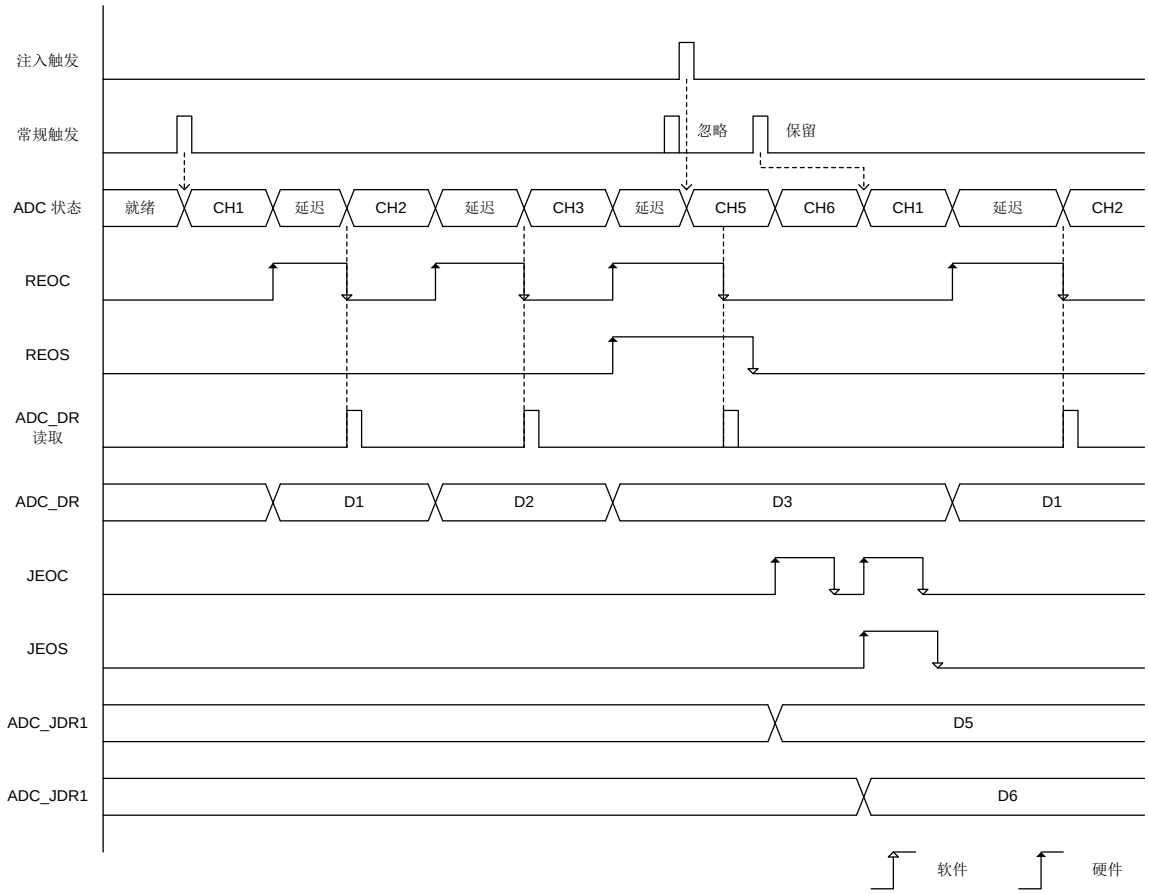


图 14-41　AUTODLY = 1，插入转换中断标准单次转换模式示意图 2

注:

1. 标准转换为硬件触发，标准序列长度 3，标准通道为 1、2、3。
2. 插入转换为硬件触发，插入序列长度 2，插入通道为 5、6。
3. AUTODLY = 1，RCONT = 0。

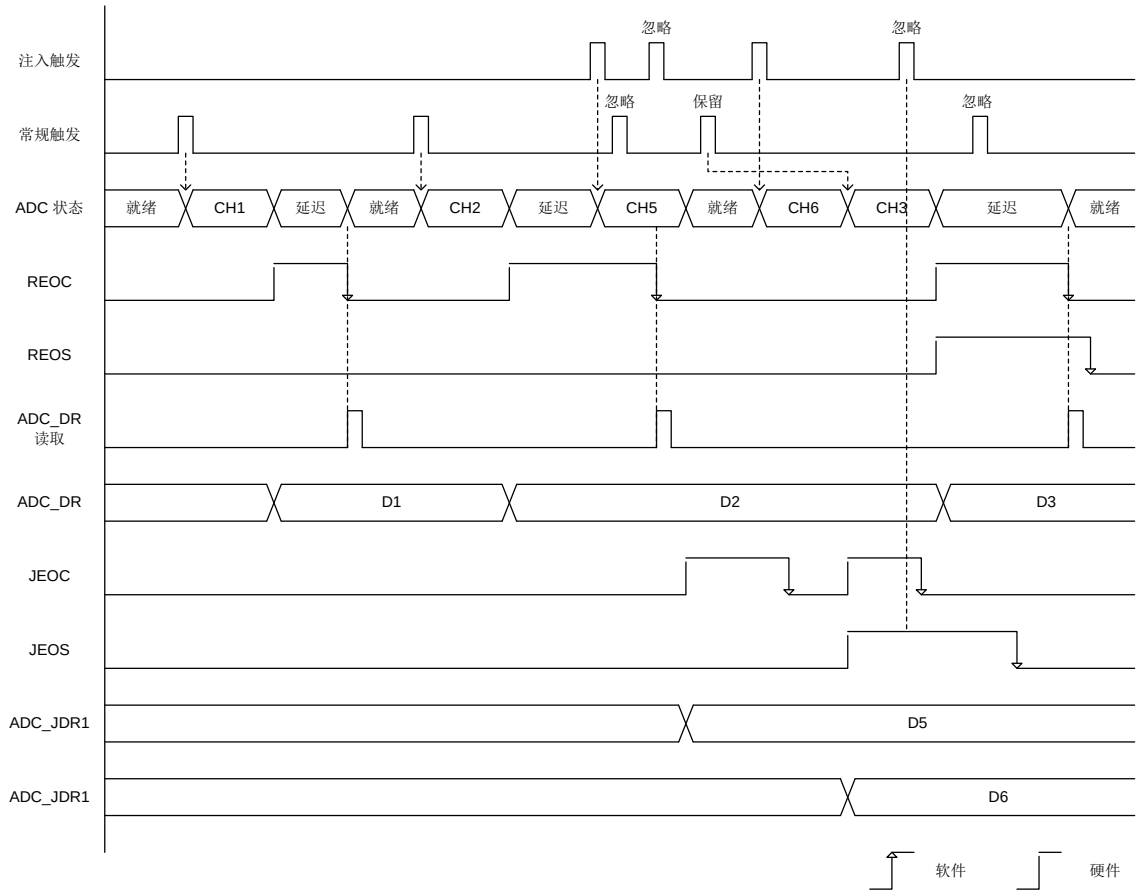


图 14-42　AUTODLY = 1，插入非连续转换中断标准非连续转换模式示意图

注:

1. 标准转换为硬件触发，标准序列长度 3，标准通道为 1、2、3，子序列长度 1。
2. 插入转换为硬件触发，插入序列长度 2，插入通道为 5、6。
3. AUTODLY = 1，DISCEN = 1，JDISCEN = 1。

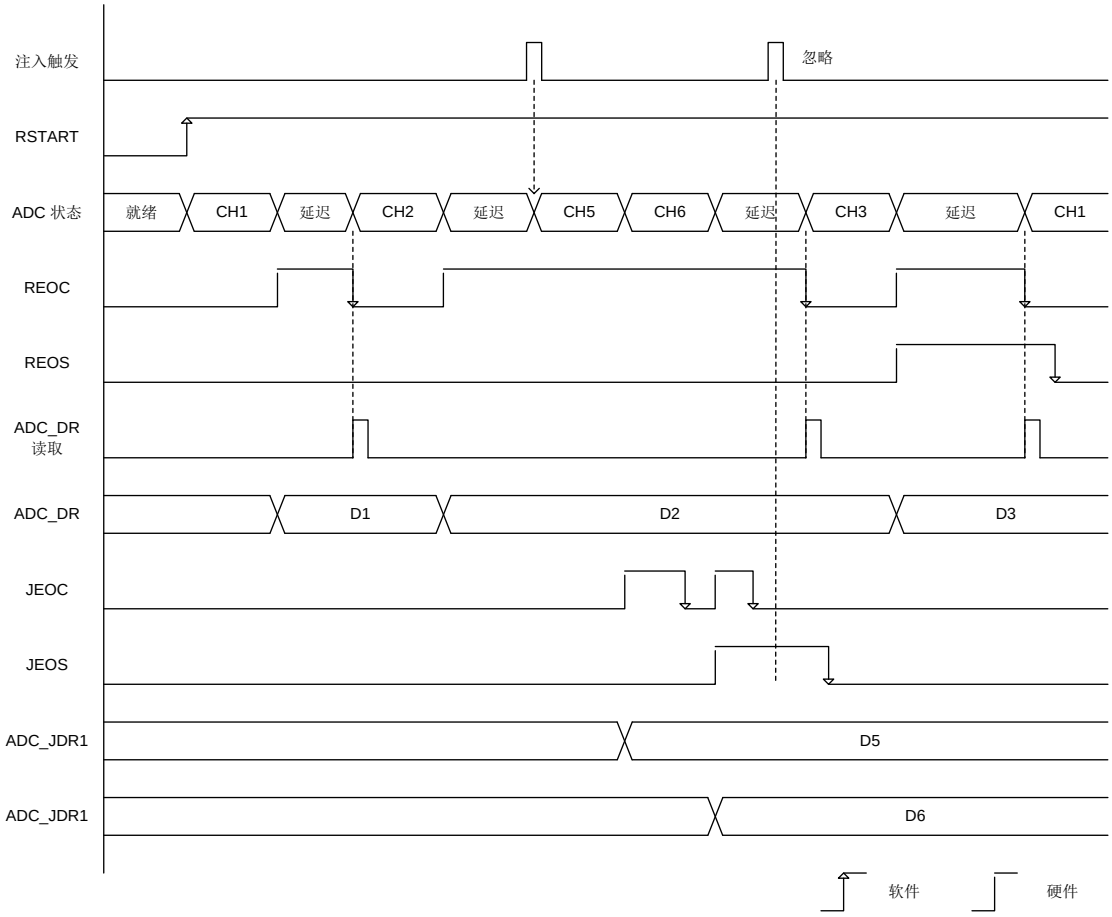


图 14-43　AUTODLY = 1，插入非连续转换中断标准非连续转换模式示意图

注:

1. 标准转换为硬件触发，标准序列长度 3，标准通道为 1、2、3，子序列长度 1。
2. 插入转换为硬件触发，插入序列长度 2，插入通道为 5、6。
3. AUTODLY = 1，DISCEN = 1，JDISCEN = 1。

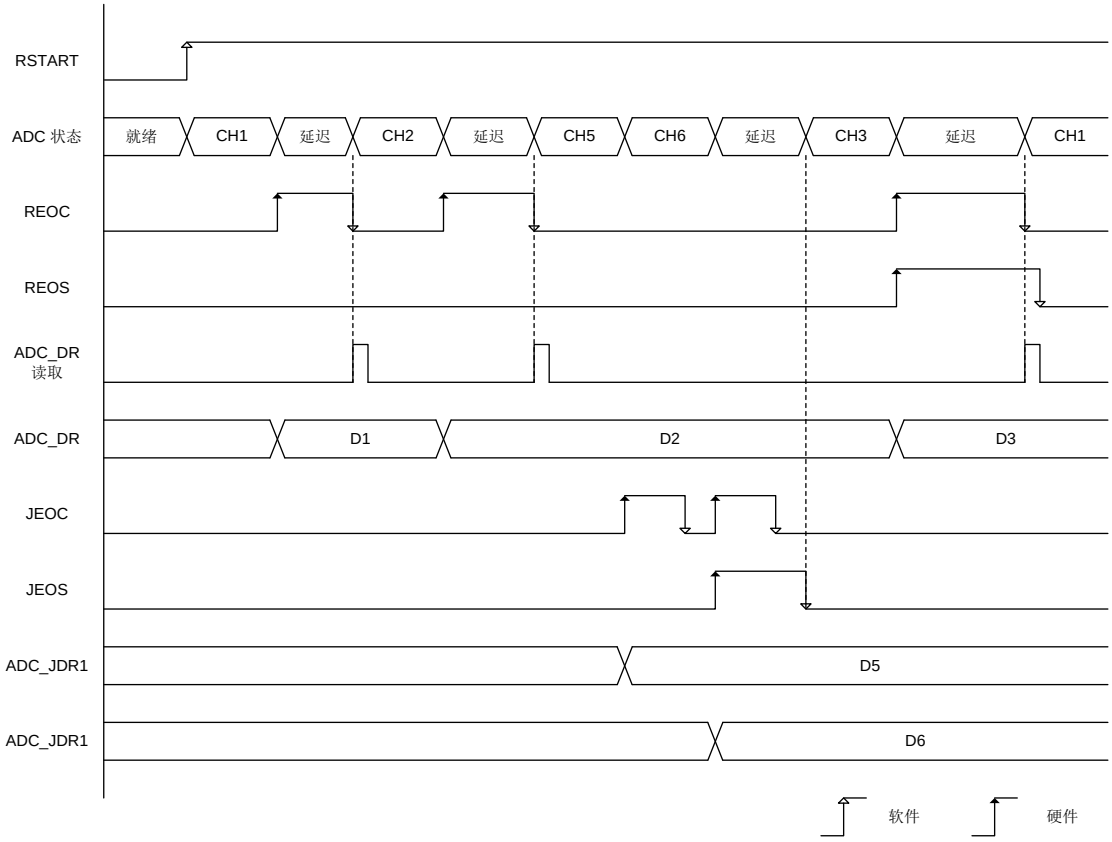


图 14-44　AUTODLY = 1，自动插入模式示意图

注:

1. 标准转换为软件触发，标准序列长度 3，标准通道为 1、2、3。
2. 自动插入模式，插入序列长度 2，插入通道为 5、6。
3. AUTODLY = 1，RCONT = 1，JAUTO = 1。

14. 4. 20 模拟看门狗

ADC 提供三种模拟看门狗，可监测设定通道的电压范围。

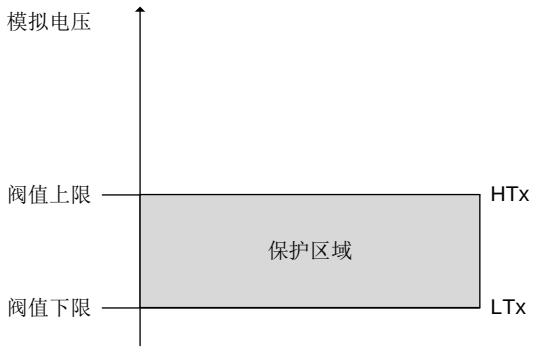


图 14-45　模拟看门狗保护范围

14. 4. 20. 1 AWD1、AWD2、AWD3 标志位以及中断

可通过寄存器 **ADC_IER** 的 AWD1、AWD2 和 AWD3 设定对应的中断开关，并且需要使用软件清除标志位来启用。此外，看门狗比较的数值是针对 ADC 转换数据进行对齐前的比较。

14. 4. 20. 2 模拟看门狗 1

设定 **ADC_CFG1** 的 AWD1REN 和 AWD1JEN 这两个控制位可以开启或关闭 ADC 的模拟看门狗 1。AWD1REN 是开启单一通道的模拟看门狗模式，AWD1JEN 则是开启所有通道的模拟看门狗模式。在启用看门狗模式后，ADC 会在每次转换后自动将数据与配置的阈值进行比较，如果数据超出阈值范围，就会产生相应的中断或触发相关的保护机制。

下表显示 **ADC_CFG1** 的配置与监控通道的选择。

监控通道	AWD1SEL	AWD1REN	AWD1JEN
无	x	0	0
所有插入通道	0	0	1
所有标准通道	0	1	0
所有通道	0	1	1
单一插入通道	1	0	1
单一标准通道	1	1	0
单一通道	1	1	1

表 14-8 模拟看门狗 1 通道选择

注: 若要选择单一通道，需要设定 ADC_CFG1.AWD1CH。被选定监控的通道必须要进行标准或插入转换。

若监控通道的电压值超过范围，即高于阀值上限或低于阀值下限，便会触发 AWD1 标志位状态。

阀值的上下限由寄存器 **ADC_TR1** 的 LT1[11:0]和 HT1[11:0]设定。如果数据分辨率低于 12 位，则需将低位的阀值保持为清除的状态。这是因为硬件的数值比较皆基于 12 位的转换数据。下表进一步介绍模拟看门狗 1 与数据分辨率的比较关系。

数据分辨率	看门狗两者的比较		批注
	转换后的数据左对齐	阀值	
12 位	DATA[11:0]	LT1[11:0]以及 HT1[11:0]	—
10 位	DATA[11:2] 00	LT1[11:0]以及 HT1[11:0]	LT1[1:0]以及 HT1[1:0]需要设定为 00。
8 位	DATA[11:4] 0000	LT1[11:0]以及 HT1[11:0]	LT1[3:0]以及 HT1[3:0]需要设定为 0000。
6 位	DATA[11:6] 000000	LT1[11:0]以及 HT1[11:0]	LT1[5:0]以及 HT1[5:0]需要设定为 000000。

表 14-9 模拟看门狗 1 与数据分辨率的比较

当 ADC 配置仅使用一个输入通道时(扫描模式下不能选择多个通道)，可以通过 **ADC_TR1** 寄存器设定有效的 ADC 转换数据间隔:

- 当转换的数据属于 **ADC_TR1** 定义的区间时，会产生 DMA 请求。
- 否则，不会发出 DMA 请求，但 **ADC_DR** 寄存器会在每次转换时更新。如果数据超出范围的次数高于 **ADC_TR1**.AWDFILT 所设定的值，则会设置对应的 AWD1 旗标并发出相应的中断。

14. 4. 20. 3 模拟看门狗 2/3

ADC 的模拟看门狗 2 和 3 可以通过设定寄存器 **ADC_AWD2CR**.AWD2CH 和 **ADC_AWD3CR**.AWD3CH 来弹性选择保护的通道。AWD2CH[18:0]和 AWD3CH[18:0]的位设定对应到监控的通道开关。

阀值设定寄存器 **ADC_TRx** 的 LTx[7:0]和 HTx[7:0] (x=2,3)，用 8 位进行比较转换数据。下表介绍了数据分辨率与比较的关系。

数据分辨率	看门狗两者的比较		批注
	转换后的数据左对齐	阀值	
12 位	DATA[11:4]	LTx[7:0] 以及 HTx[7:0]	DATA[3:0] 与比较无关
10 位	DATA[11:4]	LTx[7:0] 以及 HTx[7:0]	DATA[3:2] 与比较无关
8 位	DATA[11:4]	LTx[7:0] 以及 HTx[7:0]	—
6 位	DATA[11:6] 00	LTx[7:0] 以及 HTx[7:0]	LTx[1:0]以及 HTx[1:0]需要设定为 00。

表 14-10 模拟看门狗 2/3 与数据分辨率的比较

14. 4. 20. 4 ADC_AWD_OUT信号产生

ADC 的每个模拟看门狗都会产生一个内部硬件信号 ADCy_AWDx_OUT(y 为 ADC 编号、x 为看门狗编号)，该信号连接到定时器的 ETR 输入。ETR 信号的选择可以参考定时器的章节介绍。

当模拟看门狗启用时，ADCy_AWDx_OUT 的动作如下：

- 如果监控的通道中的数据超出了阀值区间，ADCy_AWDx_OUT 将被设置为 1。
- 当下一个转换结束后，如果转换的数据在阀值区间内，ADCy_AWDx_OUT 将被清除为 0；反之，如果在区间外，ADCy_AWDx_OUT 将保持设置的状态。
- 禁用 ADC，设置 ADCDIS 为 1，将会使 ADCy_AWDx_OUT 保持清除的状态。但是，如果使用 RSTP 或 JSTP 停止转换，对 ADCy_AWDx_OUT 的状态并不会有影响。

注: AWDx 旗标是由硬件设置，表示转换的数据是否在阀值区间内，当符合条件时会自动设置相对应的 AWDx 旗标。而软件可以读取这些旗标，并且根据需要清除它们。然而，AWDx 旗目标处理不会影响 ADCy_AWDx_OUT 的状态改变，这是由硬件自动控制的。

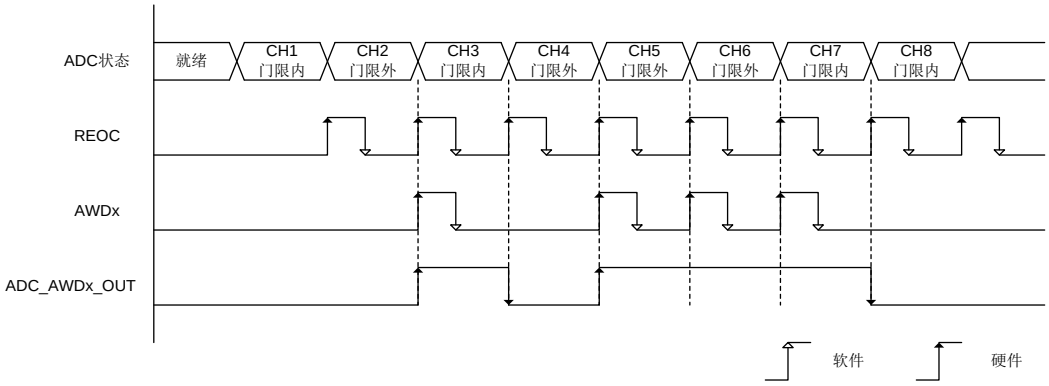


图 14-46　ADCy_AWDx_OUT 信号产生(所有标准通道)

注:
1. 标准通道为 1、2、3、4、5、6、7、8。
2. 标准通道 1、2、3、4、5、6、7、8 都受到保护。

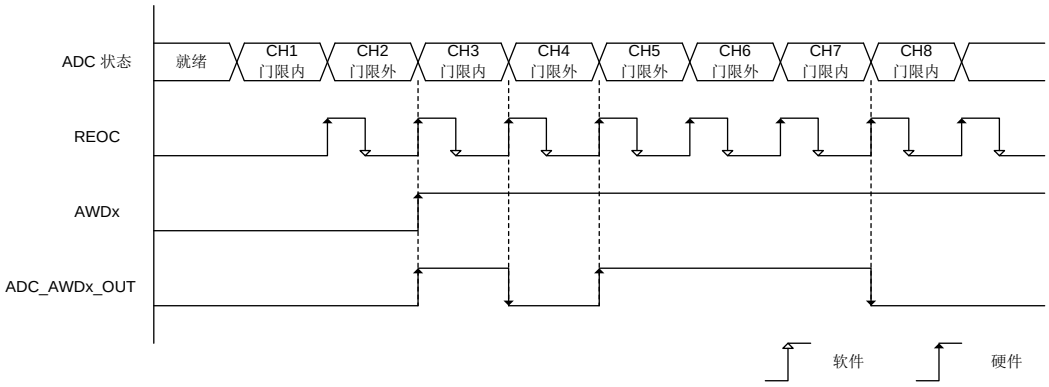


图 14-47　ADCy_AWDx_OUT 信号产生(AWD 旗标未清除)

注:
1. 标准通道为 1、2、3、4、5、6、7、8。
2. 标准通道 1、2、3、4、5、6、7、8 都受到保护。

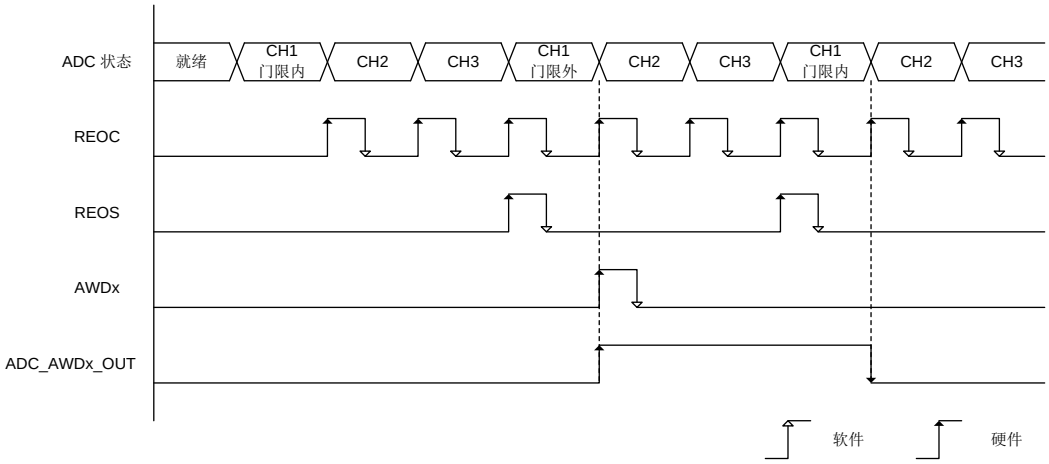


图 14-48　ADCy_AWDx_OUT 信号产生(单一通道保护)

注:
1. 标准通道为 1、2、3。
2. 标准通道 1 都受到保护。

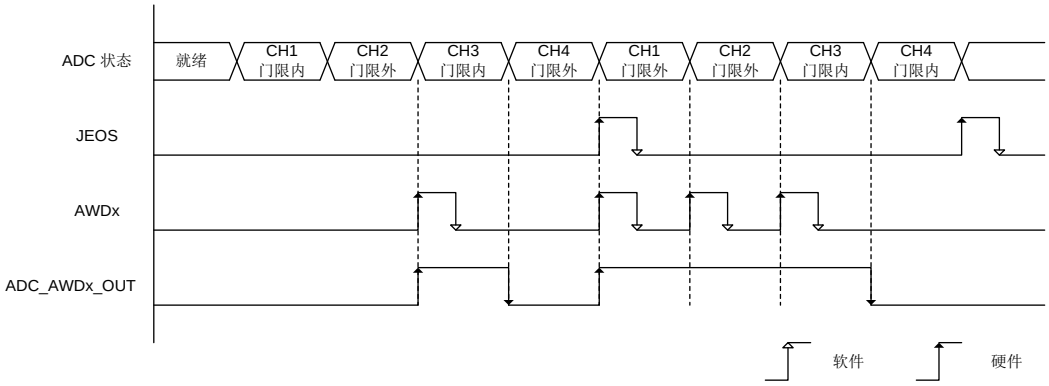


图 14-49　ADCy_AWDx_OUT 信号产生(所有插入通道)

注:
1. 插入通道为 1、2、3、4。
2. 插入通道 1、2、3、4 都受到保护。

14. 4. 20. 5　模拟看门狗与偏移以及增益补偿

当开启偏移和增益补偿时，模拟看门狗会使用补偿后的数据进行比较。

注: 若偏移补偿开启，数据可能会溢出或下溢，从而导致比较结果不正确。因此需要启用饱和(SATEN)功能，以确保看门狗的比较动作正确。此外，启用饱和功能还可以避免使用有符号数进行比较。

14. 4. 21 过采样器

过采样技术可以处理多个模拟转换，并得到它们的累加平均值，最后产生放大的数据，可达到 16 位的精度。

以下提供过采样运算公式，N 以及 M 为可调整的参数 ：

$$Result=\frac{1}{M}\times\sum_{n=0}^{n=N-1}Conversion(t_n)$$

过采样器通过硬件计算平均值、减少数据速率、改善信噪比以及基本滤波。过采样率 N 可以在 **ADC_CFG2**.OVSRATIO 寄存器中设定，范围为 2x 至 256x。除数 M 由 **ADC_CFG2**.OVSSHIFT 寄存器决定右移的位数，最多能右移 8 位。累加值最大可达 20 位。需要将其右移至 16 位，使用四舍五入将被舍弃的数据加到留下的最低位。最终的结果限制在 **ADC_DR** 寄存器中。

注: 如果数据超出 16 位，其结果将会被截断而不是使用饱和。也就是说，超出 16 位的数据位将会被直接舍弃，不会进行任何四舍五入或饱和处理。因此，需要确保使用适当的数据格式来处理输出结果，以避免数据丢失或错误。

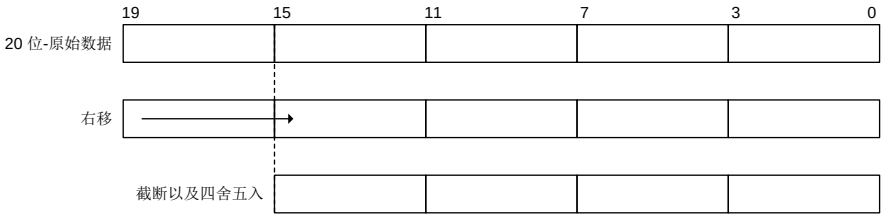


图 14-50　20 位截断至 16 位

下图显示从 20 位的原始数据如何得到 16 位结果的过程。

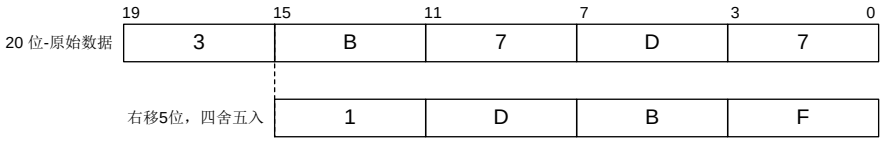


图 14-51　右移 5 位并且四舍五入的范例

下表整理出 N 与 M 的各种组合的数据输出，其原始转换数据为 0xFFF。

取样率	最大原始数据	右移 0 位	右移 1 位	右移 2 位	右移 3 位	右移 4 位	右移 5 位	右移 6 位	右移 7 位	右移 8 位
2x	0x1FFE	0x1FFE	0x0FFF	0x0800	0x0400	0x0200	0x0100	0x0080	0x0040	0x020
4x	0x3FFC	0x3FFC	0x1FFE	0x0FFF	0x0800	0x0400	0x0200	0x0100	0x0080	0x0040
8x	0x7FF8	0x7FF8	0x3FFC	0x1FFE	0x0FFF	0x0800	0x0400	0x0200	0x0100	0x0080
16x	0xFFF0	0xFFF0	0x7FF8	0x3FFC	0x1FFE	0x0FFF	0x0800	0x0400	0x0200	0x0100
32x	0x1FFE0	0xFFE0	0xFFF0	0x7FF8	0x3FFC	0x1FFE	0x0FFF	0x0800	0x0400	0x0200
64x	0x3FFC0	0xFFC0	0xFFE0	0xFFF0	0x7FF8	0x3FFC	0x1FFE	0x0FFF	0x0800	0x0400
128x	0x7FF80	0xFF80	0xFFC0	0xFFE0	0xFFF0	0x7FF8	0x3FFC	0x1FFE	0x0FFF	0x0800
256x	0xFFF00	0xFF00	0xFF80	0xFFC0	0xFFE0	0xFFF0	0x7FF8	0x3FFC	0x1FFE	0x0FFF

表 14-11　N 与 M 的最大输出配置 (蓝字表示为截断的数据)

在单一通道的过采样中，采样时间和转换时间都不会改变。每当完成 N 次转换时，就会提供新的数据。时间长度以数学式表示$N\times T_{CONV}=N\times(T_{SMPL}+T_{SAR})$。以下为个旗标设置的时机:

- 每次转换的采样周期结束后会发生 EOSMP 事件。
- 每个通道转换完成后的第 N 次转换会发生 REOC 事件，代表已经计算出该通道的过采样结果。
- 当序列中的所有通道都完成 N 次转换后，会发生 REOS 事件，代表整个序列的过采样结果都已计算完成。

14. 4. 21. 1　单个ADC运行过采样模式

使用过采样模式，ADC 支持大多数常见的工作模式，包括:

- 单次或连续转换模式
- 可使用软件或触发启动 ADC 转换
- 可在 ADC 转换过程中停止 (中止)
- 可使用 CPU 或 DMA 读取数据，并支持数据溢位侦测
- 低功耗模式(AUTODLY)
- 可编译数据分辨率

注: 过采样模式下，无法使用对齐模式。因此，ADC_CFG1.ALIGN 设定将被忽略，数据只支持右对齐。过采样模式也无法使用偏移补偿。若 ADC_CFG2 的 ROVSEN 或 JOVSEN 设定为开启，偏移开关的设定也会被忽略。

14. 4. 21. 2　模拟看门狗

模拟看门狗的功能与一般 ADC 转换有所不同:

- 比较数值采用完整的 12 位，不会受到 **ADC_CFG1**.RESOL 设定影响。
- 进行比较的数据是使用过采样模式获取的 16 位数据中的最高 12 位，即 **ADC_DR**[15:4]。

注: 使用高位移值时，必须小心，因为这会缩小比较范围。例如，如果过采样结果向右移动 4 位，因此得到一个 12 位的右对齐数据，有效的模拟看门狗比较仅能在 8 位上进行。比较是在 ADC_DR[11:4]和 HT[7:0] / LT[7:0]之间进行的，且必须保持 HT[11:8] / LT[11:8]的重置状态。

14. 4. 21. 3　触发模式

累加器可以作为基本的滤波器。虽然不是特别强大，但是可用于抑制某个特定频率(通常来自于电源本身或是电源切换)的带拒滤波器(Notch filter)。所以触发模式可以藉由 **ADC_CFG2**.TROVS 选择特定的不连续模式，可以自由定义转换时间。

下图显示不连续模式下的触发模式。

如果 TORVS=1，则会忽略 DISCEN 的设定位，将视为开启状态。

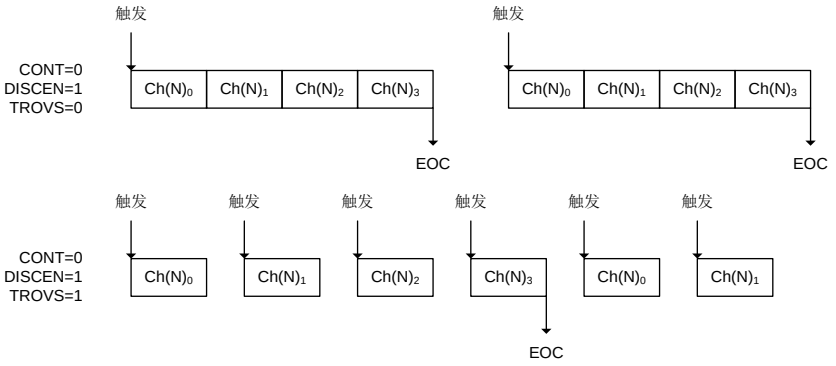


图 14-52　标准过采样不连续触发模式 (TROVS=1)

14. 4. 21. 4　标准与插入序列的过采样管理

标准与插入序列针对过采可以个别独立开关。如果两个序列必须同时使用，则需要遵守一些限制条件。

14. 4. 21. 5　仅有标准通道使用过采样

设定 **ADC_CFG2** 的 ROVSEN 为 1 可以开启标准过采样转换，然后通过 **ADC_CFG2**.ROVSM 来选择当被插入转换中断后，要如何恢复到标准序列的模式:

- 继续模式: 会从被中断之前的有效数据重新开始累加，这样可以确保在任何频率下都能完成过采样。
- 恢复模式: 会忽略先前转换的结果，从 0 开始重新累加。该模式确保过采样的数据是在同个时序内进行的转换。需要注意的是，插入触发的间隔需要超出过采样的时间，否则可能导致无法得到标准过采样的数据。

下图提供 4x 过采样率的范例说明。

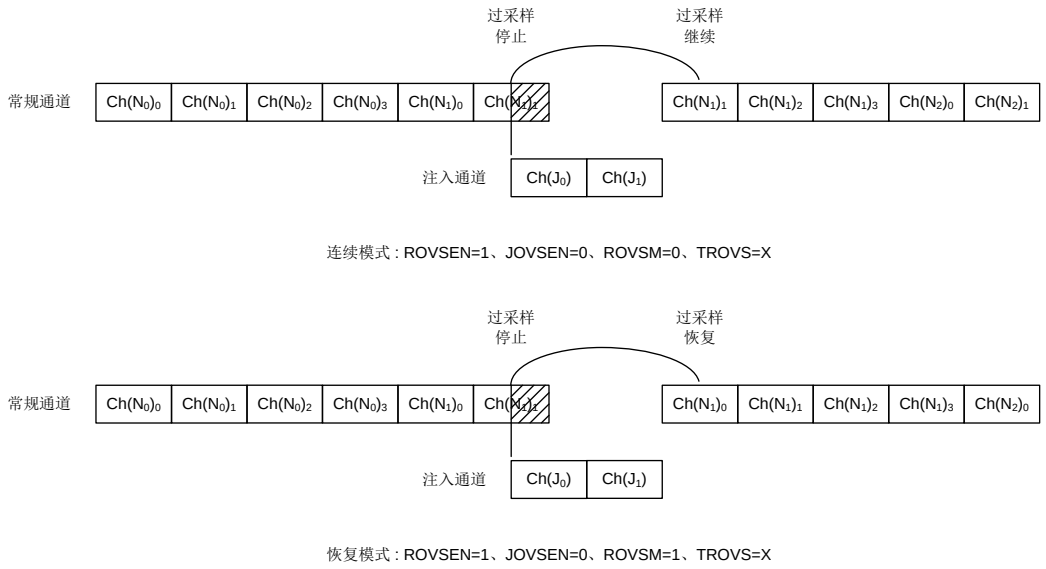


图 14-53　标准过采样模式选择示意图(4x 过采样率)

14. 4. 21. 6　仅有插入通道使用过采样

设定 **ADC_CFG2.**JOVSEN = 1 开启插入过采样转换。

14. 4. 21. 7　标准与插入通道皆使用过采样

设定 **ADC_CFG2** 的 ROVSEN 以及 JOVSEN 开启两个序列的过采样模式。在这情况下，标准过采样模式会强制进入恢复模式 (忽略 ROVSM 的设定)，如下图所示。

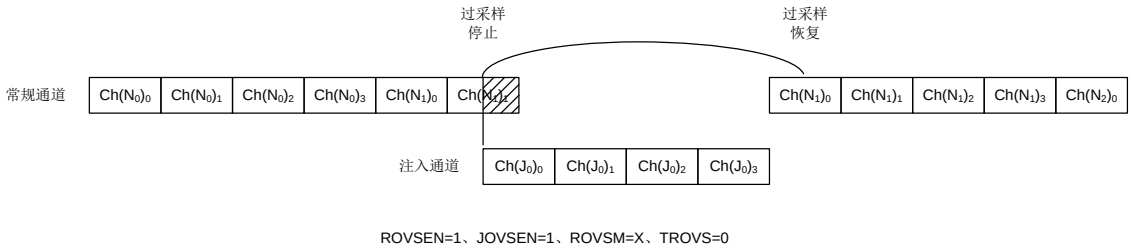


图 14-54　标准与插入同时使用过采样模式

14. 4. 21. 8　触发模式的标准过采样支持插入转换

该模式下，必须禁止插入过采样模式，并强制进入恢复模式(忽略 ROVSM 的设定)，此时 JOVSEN 必须保持清除状态。详细设定如下图所示。

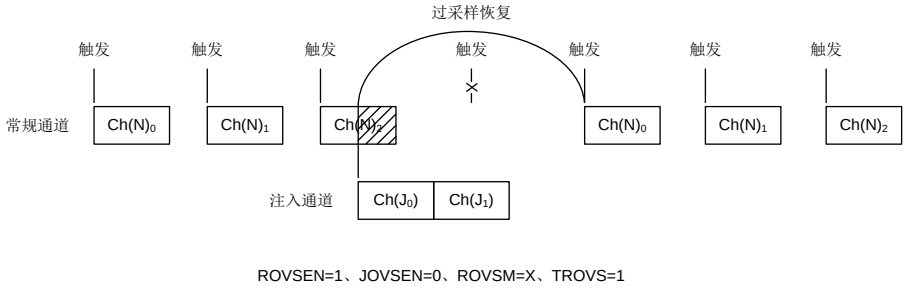


图 14-55　标准触发过采样下支持插入转换

14. 4. 21. 9　自动插入触发模式

在过采样模式下，可以使用自动插入触发来进行转换，此方式可以将所有转换的结果存储在寄存器中，以节省 DMA 资源。使用自动插入触发模式的设定为:JAUTO = 1、ROVSEN = 1、JOVSEN = 1，其他组合则不支持。在此模式下，ROVSM 的设定会被忽略。转换的方式下图所示。

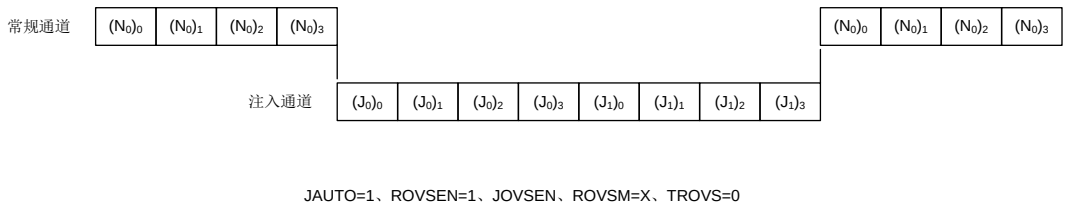


图 14-56　自动插入转换的过采样模式

该模式可以设定 TROVS 开启触发模式。在这情况下，ADC 必须配置成：JAUTO = 1、DSICEN = 0、JDISCEN = 0、ROVSEN = 1、JOVSEN = 1 以及 TROVS = 1。

14. 4. 21. 10 双重 ADC 使用过采样

双重 ADC 可以在插入同步模式和标准同步模式下配置使用过采样模式。在这种情况下，需要将两个 ADC 的配置设置完全相同。

在其他双重 ADC 的操作模式，是不支持过采样模式。

14. 4. 21. 11 组合模式汇整

下表整理了各种组合，也包含了不支持的模式。

ROVSEN	JOVSEN	ROVSM	TROVS	批注
1	0	0	0	标准继续过采样模式
1	0	0	1	不支持
1	0	1	0	标准恢复过采样模式
1	0	1	0	触发标准恢复过采样模式
1	1	0	X	不支持
1	1	1	0	标准恢复过采样模式搭配插入过采样
1	1	1	1	不支持
0	1	X	X	插入过采样

表 14-12　过采样工作模式汇整

14. 4. 22 双重ADC

双重 ADC 模式可以在拥有两个或以上 ADC 的设备中使用(参见图 14-57)。

在双重 ADC 模式下，转换由 ADC1(Master ADC)或是 ADC2 (Slave ADC)交替或同时触发，取决于 **ADC_CCR**.DUAL[4:0]所选择的模式。

四种主要模式可选择 ：

- 插入同步模式
- 标准同步模式
- 交替模式
- 交替触发模式

也可以使用以下方式结合这些模式：

- 插入同步模式 + 标准同步模式
- 标准同步模式 + 交替触发模式
- 插入同步模式 + 交替模式

当 **ADC_CCR**.DUAL[4:0]设定不等于零时，处于双重 ADC 模式。在此模式下，**ADC_CFG1** 的 RCONT、AUTDLY、DISCEN、DISCNUM[2:0]、JDISCEN、JQM 以及 JAUTO 的设定值，ADC1 以及 ADC2 之间共享，ADC2 中的设定值始终等于 ADC1 的对应位。

在双重模式下开始转换，用户必须对 ADC1 以及 ADC2 配置 REXTEN[1:0]、REXTSEL[4:0]、JEXTEN[1:0]、JEXTSEL[4:0]位，以配置软件或硬件触发和标准或插入触发。

在标准同步或交替模式下: 一旦用戶设置 ADC1 的 RSTART 或 RSTP，ADC2 的对应位也会自动设置。但是，ADC2 的 RSTART 或 RSTP 可能不会与 ADC1 同时清除。

在同时插入或替代触发模式下: 一旦用戶设置 ADC1 的 JSTART 或 JSTP，ADC2 的对应位也会自动设置。但是，ADC2 的 JSTART 或 JSTP 可能不会与 ADC1 同时清除。

在双重 ADC 模式下，可以通过读取 **ADC_CDR** 同时取得 ADC1 和 ADC2 的转换数据。可以通过读取 **ADC_CSR** 同时取得状态。

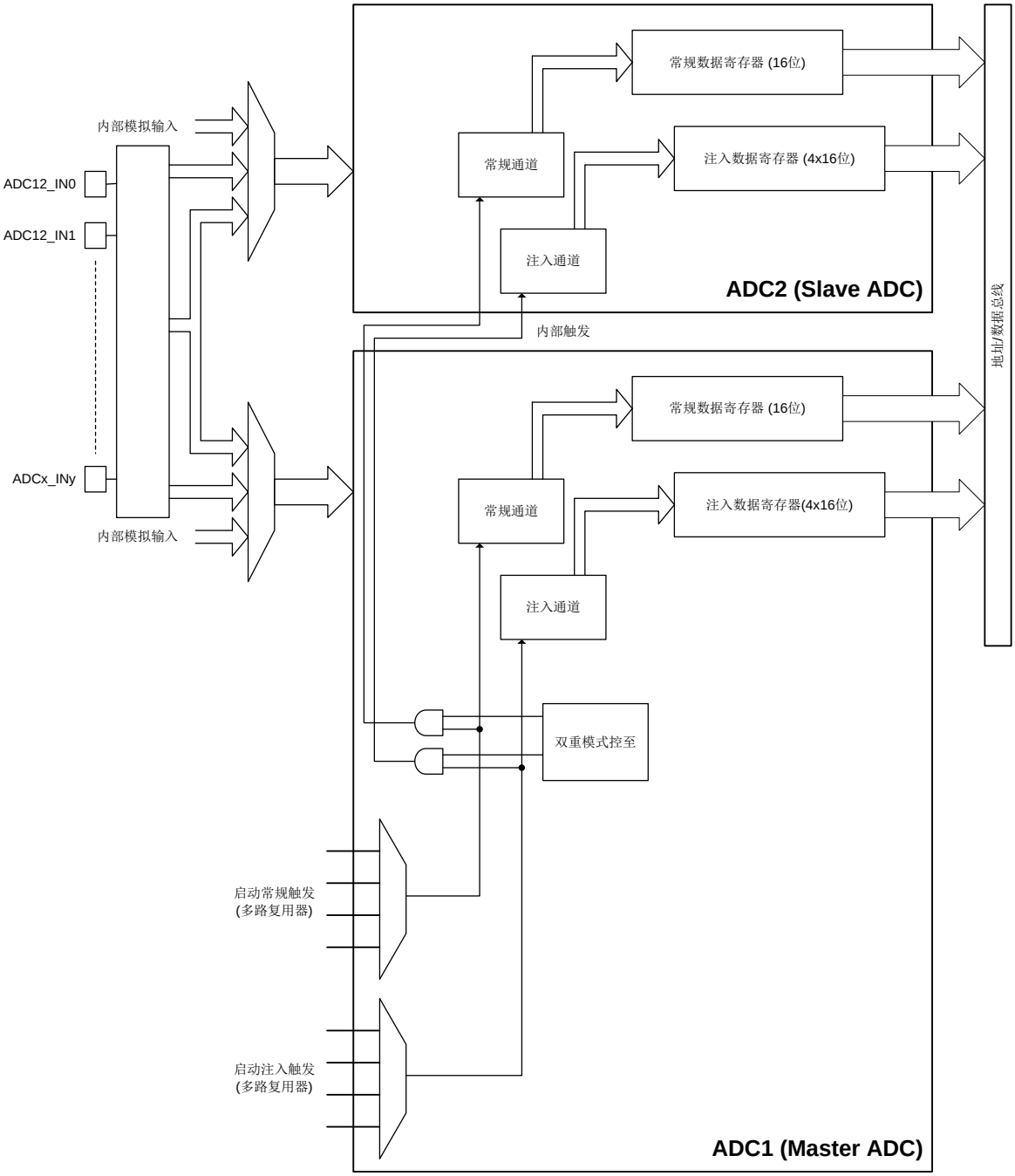


图 14-57　多重 ADC 方块图

14. 4. 22. 1　插入同步模式

通过设定 **ADC_CCR**.DUAL[4:0] = 00101 选择此模式。

在这种模式下，欲转换的插入序列的外部触发源的设定，皆来自 ADC1 的 **ADC_JSQR**.JEXTSEL 的选择。

注: 请勿在两个 ADC 上转换同一通道(当转换同一通道时，两个 ADC 的取样时间不应重迭)。
在同步模式下，必须转换具有相同长度的序列，或确保触发之间的间隔时间大于两个序列中最长的那个。否则，序列较短的 ADC 可能会重新启动，而序列较长的 ADC 还正在进行先前的转换。

可以对其中一个或所有 ADC 进行标准期转换。在这种情况下，彼此独立，在插入事件发生同时被中断，并在插入转换组结束后同时恢复。

- ADC1 的插入转换序列结束时，转换的数据被存储到 ADC1 的寄存器 **ADC_JDRy**。并产生一个 JEOS 中断(如果已启用)
  - ◇ ADC2 的插入转换序列结束时，转换的数据被存储到 ADC2 的寄存器 **ADC_JDRy**。并产生一个 JEOS 中断(如果已启用)
  - ◇ 如果 ADC1 插入序列的区间时间等于 ADC2 插入序列的区间时间(例如图 14-58)，则软件可以使用其中一个 JEOS 中断，读取两个转换的数据。

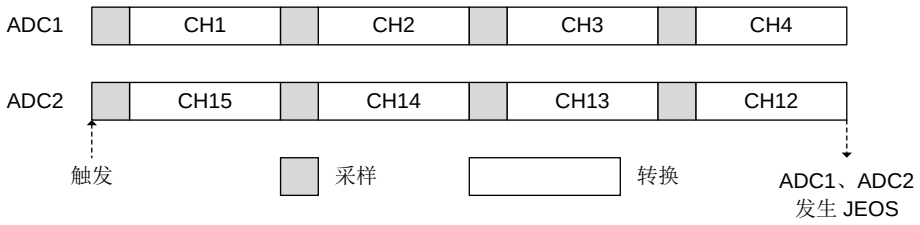


图 14-58　4 个通道的同步插入模式: 双重 ADC 模式

如果 JDISCEN=1，每次插入序列的同步转换都需要发生插入触发事件。

这种模式可以与 AUTDLY 模式结合使用：

- 当同步插入序列的转换结束后，仅在 ADC1 和 ADC2 的 JEOS 字节已清除时才会接受新的插入触发事件。在进行中的插入序列和相关的延迟阶段中发生的任何新的插入触发事件都将被忽略。
- 当 ADC1 的标准转换序列结束后，仅在读取 **ADC1_DR** 寄存器后才会接受 ADC1 的新标准触发事件。在进行中的标准序列和相关的延迟阶段中发生的任何 ADC1 的新标准触发事件都将被忽略。对于发生在 ADC2 上的标准序列也是相同的行为。

14. 4. 22. 2　标准同步模式

通过设定 **ADC_CCR**.DUAL[4:0] = 00110 选择此模式。

在这种模式下，欲转换的标准通道的外部触发源的设定，皆来自 ADC1 的 **ADC_CFG1**.REXTSEL 的选择。

在此模式中，支持各 ADC 独立的插入转换。插入事件会中止当前的标准同步转换，并在插入序列转换完成后重新启动。

注: 请勿在两个 ADC 上转换同一通道(当转换同一通道时，两个 ADC 的取样时间不应重迭)。
在标准同步模式下，必须将具有相同长度的序列进行转换，或确保触发之间的间隔时间大于 2 个序列中的最长转换时间。否则，序列较短的 ADC 可能会重新启动，而序列较长的 ADC 还正在进行先前的转换。

软件可以通过中断的通知，读取转换后的数据:

- ADC1 的标准通道转换结束时，转换的数据被存储到 ADC1 的寄存器 **ADC_DR**。并产生一个 REOC 中断(如果已启用)
- ADC2 的标准通道转换结束时，转换的数据被存储到 ADC2 的寄存器 **ADC_DR**。并产生一个 REOC 中断(如果已启用)
  - ◇ 如果 ADC1 标准序列的区间时间等于 ADC2 标准序列的区间时间(例如图 14-59)，则软件可以使用其中一个 REOC 中断，读取两个转换的数据。

接下来介绍了使用 DMA 读取 ADC 转换数据的两种方法。

- 第一种方法需要使用两个 DMA 通道(一个用于 ADC1，一个用于 ADC2)。需要保持 **ADC_CCR**.MDMA[1:0]为清零状态。具体操作如下：
  - ◇ 配置 ADC1 DMA 通道，从 **ADC_DR** 读取数据。DMA 请求会在每次 REOC 事件时生成。
  - ◇ 配置 ADC2 DMA 通道，从 **ADC_DR** 读取数据。DMA 请求会在每次 REOC 事件时生成。
- 第二种方法使用 MDMA 模式，可以保留一个 DMA 通道用于其他用途：
  - ◇ 配置 MDMA[1:0] = 10 或 11(取决于数据分辨率)。
  - ◇ 只使用 ADC1 DMA 通道，从 **ADC_CDR** 读取数据。
  - ◇ 每当 ADC1 以及 ADC2 的 REOC 事件均发生时，将生成 DMA 请求。此时，ADC2 转换的数据存放在 **ADC_CDR** 寄存器的高 16 位，ADC1 转换的数据存放在 **ADC_CDR** 寄存器的低 16 位。
  - ◇ 当 DMA 读取 **ADC_CDR** 寄存器时，两个 REOC 期交皆被清除。

注: 在 MDMA 模式中，用户必须确保 ADC1 与 ADC2 的转换序列中有着相同数量的转换。如果 ADC1 和 ADC2 的转换数量不同，那么剩余的转换将无法触发 DMA 请求。

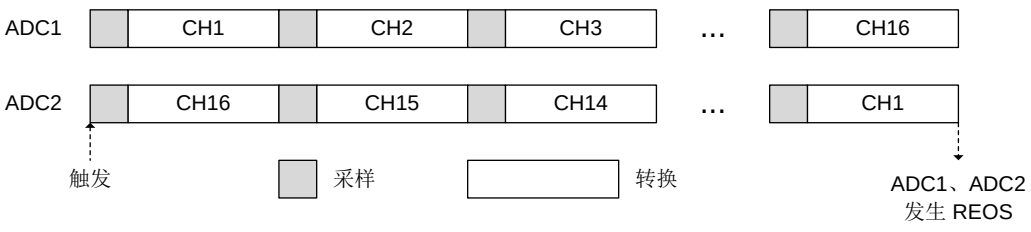


图 14-59　16 个通道的同步标准模式: 双重 ADC 模式

如果 DISCEN = 1，则每 n 个同时转换的标准子序列都需要一个标准触发事件(其中 n 由 **ADC_CFG1**.DISCNUM[2:0]定义)。

这种模式可以与 AUTDLY 模式结合使用：

- 一旦通道标准同步转换结束，只有在已读取通用数据寄存器 **ADC_CDR** 之后，才会开始序列中的下一次标准转换。
- 一旦同步的标准转换序列结束，只有在已读取通用数据寄存器 **ADC_CDR** 之后，才会接受新的标准触发事件。正在进行的标准序列和相关的延迟阶段中发生的任何新标准触发事件都会被忽略。

在 MDMA 模式 **ADC_CCR**.MDMA[1:0]必须设置为 10 或 11。当标准同步模式与 AUTDLY 结合使用时，用户必须确保以下内容：

- ADC1 以及 ADC2 的标准序列中转换的次数要相等。
- ADC2 的标准转换的长度应小于 ADC1 转换的长度。请注意，序列的长度取决于要转换的通道数、每个通道的采样时间以及数据分辨率。

注:标准同步模式和 AUTDLY 模式的组合中，仅限于使用标准通道的情况下使用。在此结合模式下，禁止使用插入通道。

14. 4. 22. 3 交替模式

通过设定 **ADC_CCR**.DUAL[4:0] = 00111 选择此模式。
此模式仅能在标准序列(通常设定为一个通道)上启动。外部触发源来自 ADC1 的设定。
外部触发发生后：

- ADC1 立即启动。
- 经过几个 ADC 时钟周期的延迟后，当 ADC1 的采样阶段完成后，ADC2 开始运行。

在交替模式下分隔两个转换的最小延迟需要由寄存器 **ADC_CCR**.DELAYSEL 进行配置。此延迟在 ADC1 转换的采样阶段结束的半个周期后开始计数。(在特定时间内只有一个 ADC 可以对输入信号进行取样)。

- 最小可能的 DELAYSEL 为 1，以确保在 ADC1 采样阶段结束和 ADC2 采样阶段开始之间至少有一个周期时间。
- 最大 DELAYSEL 周期数将对应所选的数据分辨率。用戶必须适当地计算延迟时间，以确保其中一个正在进行采样的时候，另一个开始转换。

如果在 ADC1 以及 ADC2 都设置了 RCONT，则两个 ADC 将会进行连续的标准序列转换。
当 ADC2 的每个标准通道转换结束时(REOC)，软件会通过中断通知可以读取 ADC1 以及 ADC2 的 **ADC_DR**。

注: 可以仅启用 ADC2 的 REOC 中断，并读取共同的数据寄存器 ADC_CDR。在这种情况下，用戶必须确保 ADC 转换的间隔时间是兼容的建议使用 MDMA 模式。

使用 DMA 传输标准数据时，不能使用各 ADC 单独的 DMA 请求，必须使用 MDMA 模式。

- 设置 MDMA[1:0] = 10 或 11(取决于数据分辨率)。

- DMA 配置 ADC1 的通道并且读取 ADC 共同的数据寄存器 **ADC_CDR**。
- 当 ADC1 以及 ADC2 的 REOC 发生时，将生成单个 DMA 请求。此时，ADC2 转换的数据存放在 **ADC_CDR** 寄存器的高 16 位，ADC1 转换的数据存放在 **ADC_CDR** 寄存器的低 16 位。
- 当 DMA 读取 **ADC_CDR** 寄存器时，AD1 以及 ADC2 的 REOC 旗标皆被清除。

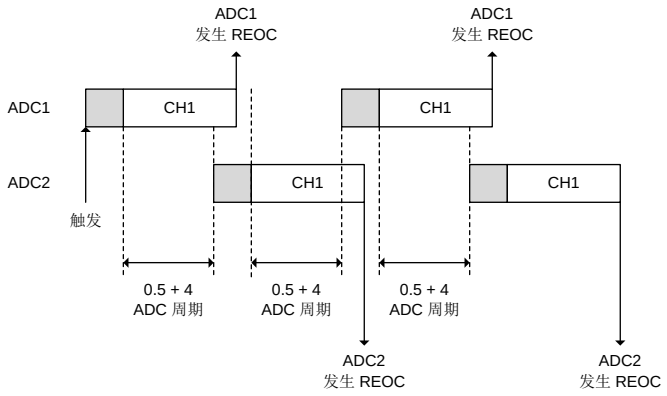


图 14-60　单一通道，DELAYSEL=3 的交替模式: 双重 ADC 模式

如果 DISCEN = 1，则每 n 个同步转换的标准子序列都需要一个标准触发事件(其中 n 由 **ADC_CFG1**.DISCNUM[2:0]定义)。

在这种模式下，支持插入转换。当插入转换发生后，ADC1 以及 ADC2 的标准转换都将被中止。而插入序列完成后，皆将从 ADC1 重新启动(参见下图)。

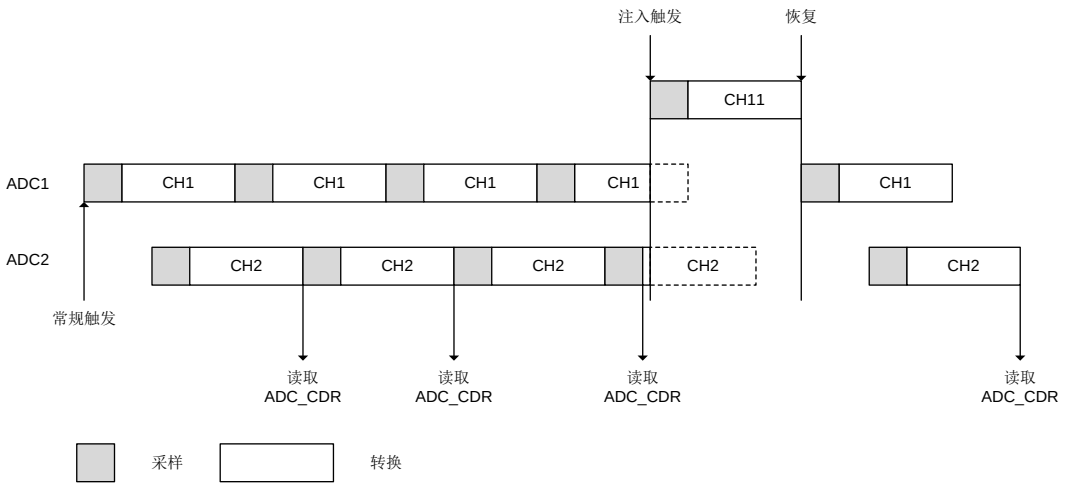


图 14-61　受到插入序列中断的交替模式

14. 4. 22. 4　交替触发模式

通过设定 **ADC_CCR**.DUAL[4:0] = 01001 选择此模式。

交替触发模式只适用于插入序列。外部触发来源为 ADC1 的插入选择，并且只能选择硬件触发(JEXTEN[1:0]不能为 00)。

- 插入非连续模式关闭 (所有 ADC 的 JDISCEN =0)
  1. 当第一个触发事件发生时，所有 ADC1 插入序列的通道都会被转换。
  2. 当第一个触发事件发生时，所有 ADC2 插入序列的通道都会被转换。
  3. 以此类推
- 如果启用了 ADC1 以及 ADC2 的 **ADC_IER**.JEOC 中断，则 ADC1 以及 ADC2 插入通道完成后会生成中断。
- 如果启用了 ADC1 以及 ADC2 的 **ADC_IER**.JEOS 中断，则 ADC1 以及 ADC2 插入序列完成后会生成中断。
- 如果 ADC1 以及 ADC2 插入序列皆转换完成后再次发生外部触发事件，则从 ADC1 重新启动序列转换。

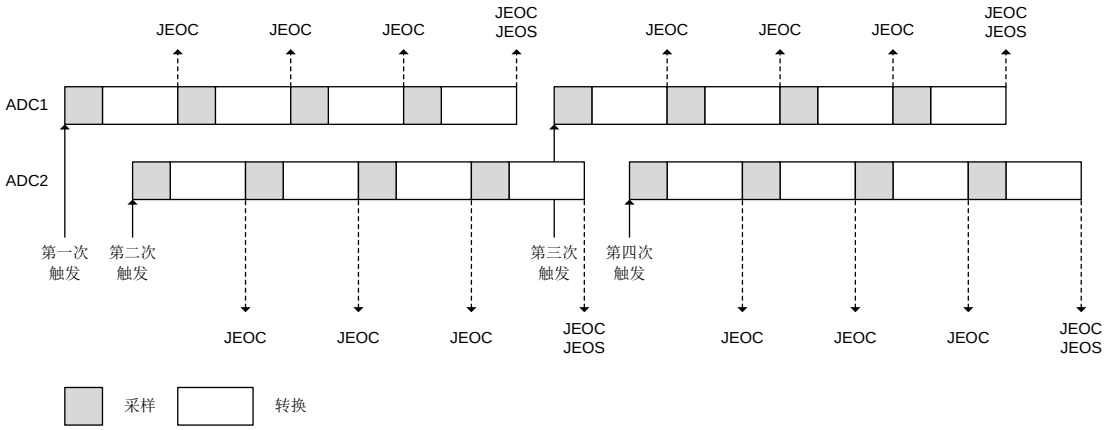


图 14-62　交替触发模式: ADC1 以及 ADC2 的插入序列

注: ADC1 以及 ADC2 可以进行独立的标准转换。当 ADC1 以及 ADC2 必须执行插入转换的时候，将会中断标准转换，等到插入序列转换结束后，才会恢复。

- 插入非连续模式开启 (所有 ADC 的 JDISCEN =1)
  1. 当第一个触发事件发生时，开始 ADC1 插入序列第一个通道转换。
  2. 当第一个触发事件发生时，开始 ADC2 插入序列第一个通道转换。
  3. 以此类推
- 如果启用了 ADC1 以及 ADC2 的 **ADC_IER**.JEOC 中断，则 ADC1 以及 ADC2 插入通道完成后会生成中断。
- 如果启用了 ADC1 以及 ADC2 的 **ADC_IER**.JEOS 中断，则 ADC1 以及 ADC2 插入序列完成后会生成中断。
- 如果 ADC1 以及 ADC2 插入序列皆转换完成后再次发生外部触发事件，则从 ADC1 重新启动序列转换。

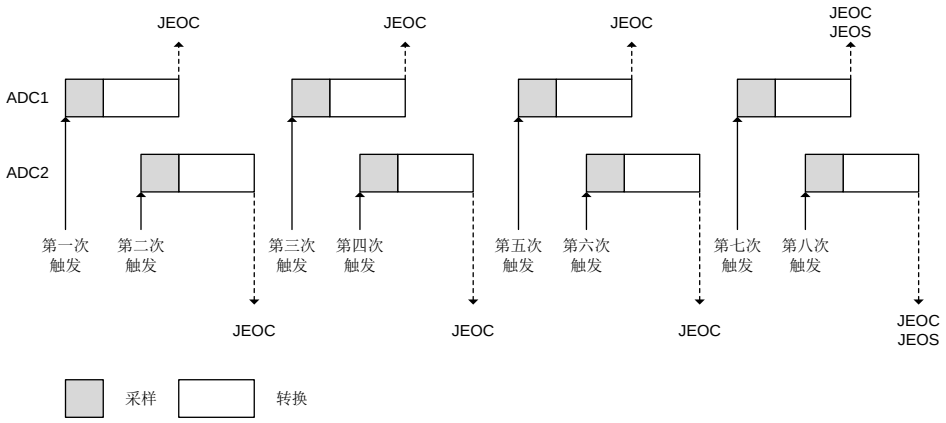


图 14-63　交替触发模式: 非连续模式

14. 4. 22. 5　标准以及插入同步组合模式

通过设定 **ADC_CCR**.DUAL[4:0] = 00001 选择此模式。

可以使用插入同步转换中断标准同步转换。

注: 在此模式中，必须将具有相同长度的序列进行转换，或确保触发之间的间隔时间大于 2 个序列中的最长转换时间。否则，序列较短的 ADC 可能会重新启动，而序列最长的 ADC 正在完成先前的转换。

14. 4. 22. 6　标准同步以及交替触发组合模式

通过设定 **ADC_CCR**.DUAL[4:0] = 00010 选择此模式。

可以使用交替触发的插入序列转换来中断标准同步转换。图 14-64 显示了交替触发中断标准同步转换的行为。

插入的交替转换会在插入事件后立即开始。如果已经开始了一个标准转换，为了确保插入转换后的同步，ADC1 以及 ADC2 的标准转换都会停止，并在插入转换结束时同步恢复。

注: 在此模式中，必须将具有相同长度的序列进行转换，或确保触发之间的间隔时间大于 2 个序列中的最长转换时间。否则，序列较短的 ADC 可能会重新启动，而序列最长的 ADC 正在完成先前的转换。

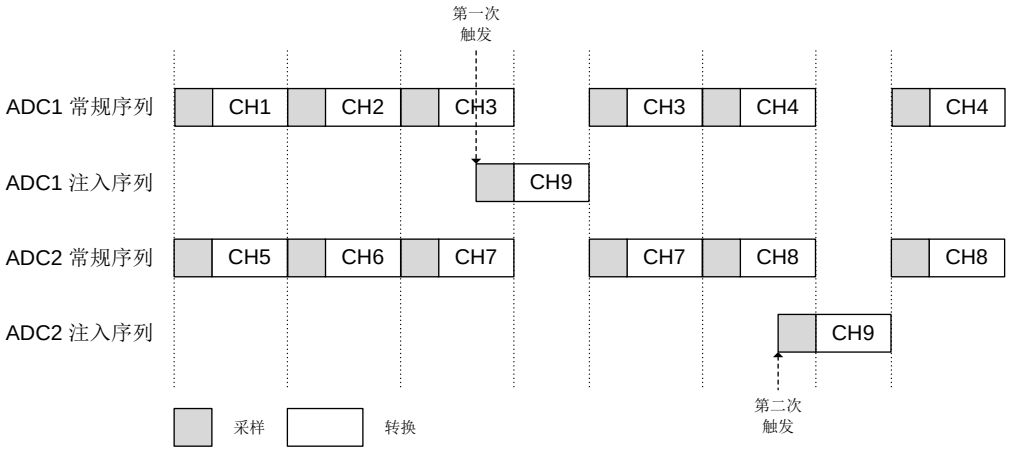


图 14-64　标准同步 + 交替触发

如果在一次插入转换期间触发了另一个插入触发，则该交替触发会被处理而启动另一个 ADC 的插入触发。但是在 ADC1 以及 ADC2 都正在进行各自的插入转换，发生了的插入触发将会被忽略。

14. 4. 22. 7　插入同步以及交替组合模式

通过设定 **ADC_CCR**.DUAL[4:0] = 00011 选择此模式。

可以使用同步插入事件中断交替转换。

在这种情况下，交替转换会立即中断并开始插入同步转换。在插入序列结束时，会恢复先前的交替转换。当交替转换恢复时，执行的第一个标准转换始终是 ADC1。图 14-65、图 14-66 以及图 14-67 展示了行为说明。

注: 在此模式下，必须使用共同数据寄存器 ADC_CDR 进行读取访问来读取 ADC1 以及 ADC2 的标准数据。

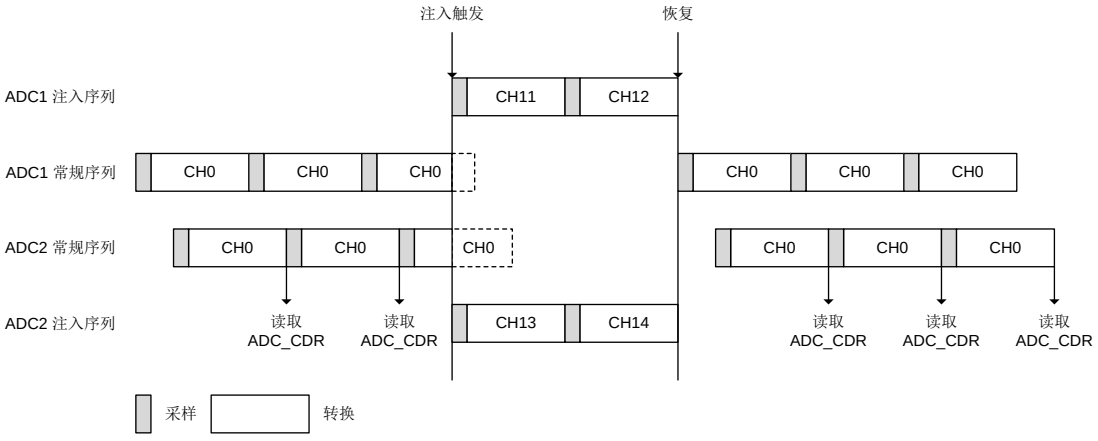


图 14-65　插入同步转换中断单一通道的交替转换

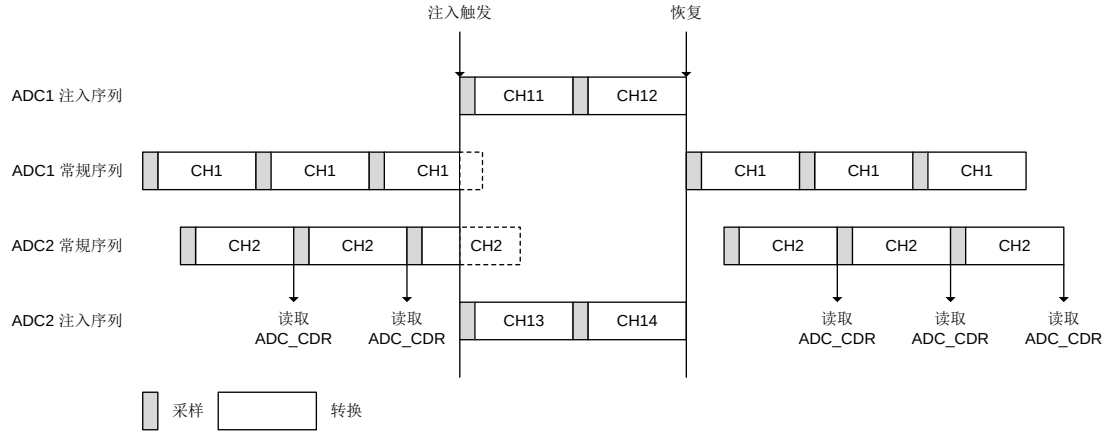


图 14-66　插入同步转换中断两个通道的交替转换: 中断 ADC1

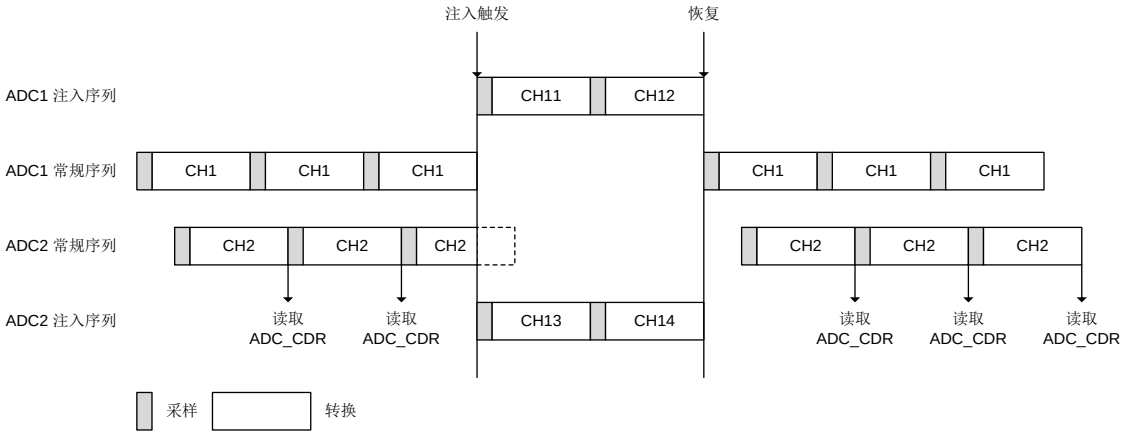


图 14-67　插入同步转换中断两个通道的交替转换: 中断 ADC2

14. 4. 22. 8　双重ADC模式的DMA请求

使用窗重 ADC 模式下，是可以使用 ADC1 以及 ADC2 各自的 DMA 进行数据传输。(参照下图)

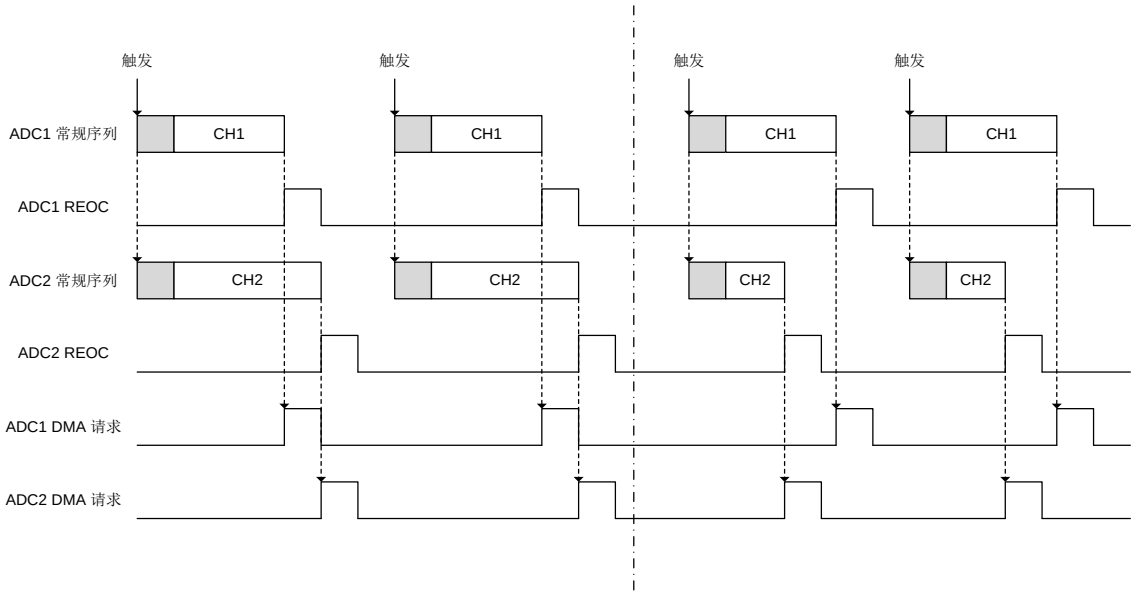


图 14-68　标准同步模式的 DMA 请求: MDMA=00

当配置了 **ADC_CCR**.MDMA[1:0]，在标准同步转换以及交替模式下，可以使用单一 DMA 通道进行 ADC1 以及 ADC2 的标准数据传输。

- MDMA = 10: 当 ADC1 以及 ADC2 的 REOC 事件都发生时，会生成单一 DMA 请求。此时，ADC2 转换的数据存放在 **ADC_CDR** 寄存器的高 16 位，ADC1 转换的数据存放在 **ADC_CDR** 寄存器的低 16 位。此配置适用于数据分辨率为 10 或是 12 位的标准转换。
  例如: 交替模式，每当 ADC1 以及 ADC2 的数据可用时，会产生一个 DMA 请求。
  DMA 请求，**ADC_CDR**[31:0] = ADC2 **ADC_DR** | AD1 **ADC_DR**
  DMA 请求，**ADC_CDR**[31:0] = ADC2 **ADC_DR** | AD1 **ADC_DR**

注: 当使用 MDMA 模式时，用户必须注意适当配置 ADC1 以及 ADC2 转换的间隔时间，以便在新的转换数据可用之前，生成 DMA 请求并且搬移数据 (ADC1 + ADC2)。

- MDMA = 11: 与 MDMA = 10 生成 DMA 请求的方式相同，唯一不同之处，两个 ADC 转换数据会以字节存放在 **ADC_CDR** 的半字节。ADC2 转换的数据存放在 **ADC_CDR** 寄存器半字节的高 8 位，ADC1 转换的数据存放在 **ADC_CDR** 寄存器半字节的低 8 位。此配置适用于数据分辨率为 6 或是 8 位的标准转换。
  例如 ：交替模式，每当 ADC1 以及 ADC2 的数据可用时，会产生一个 DMA 请求。
  DMA 请求，**ADC_CDR**[15:0] = ADC2 **ADC_DR**[7:0] | AD1 **ADC_DR**[7:0]
  DMA 请求，**ADC_CDR**[15:0] = ADC2 **ADC_DR**[7:0] | AD1 **ADC_DR**[7:0]

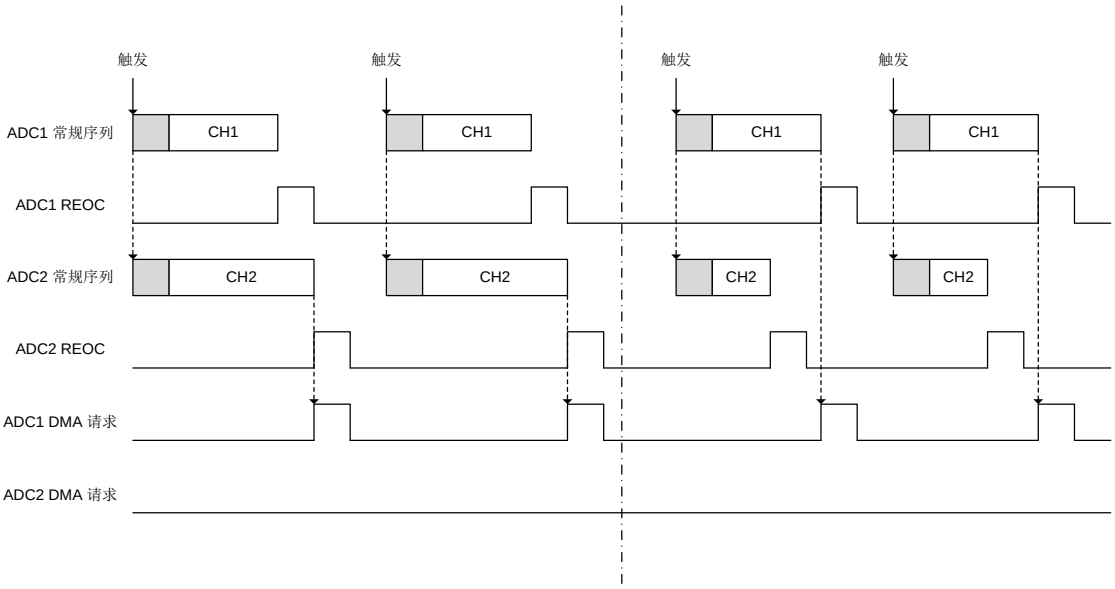


图 14-69　标准同步模式的 DMA 请求: MDMA=10

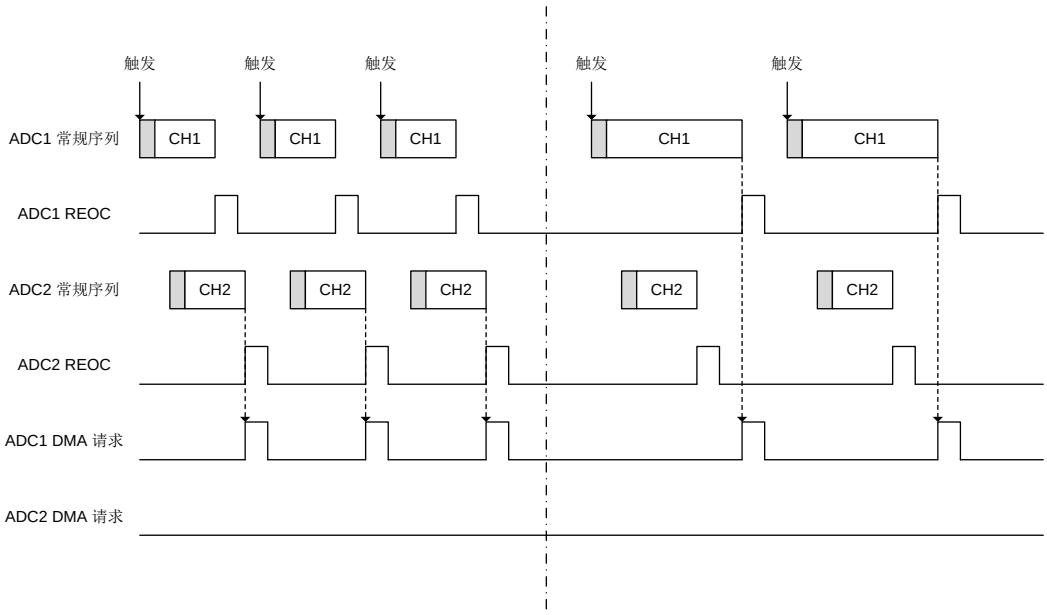


图 14-70　交替模式的 DMA 请求: MDMA=10

14. 4. 22. 9　数据溢出检测

在双重 ADC 模式中(**ADC_CCR**.DUAL[4:0]不等于 00000)，如果在其中一个 ADC 上有检测到数据溢出，将不再发出 DMA 请求，以确保传输到 RAM 的所有数据都是有效的(无论 MDMA 配置)。可能会发生其中一个 ADC 对应的 REOC 旗标仍然被设置的情况，因为该 ADC 的数据寄存器的数据为有效的。

14. 4. 22. 10　双重ADC模式下停止转换

在双重模式下，设置 ADC1 的 **ADC_CON**.RSTP 以及 JSTP 控制位，将停止两个 ADC 的转换。

而设置 ADC2 的 **ADC_CON**.RSTP 以及 JSTP 是无效的。一旦两个 ADC 都有效停止，ADC1 以及 ADC2 的 **ADC_CON**.RSTART 以及 JSTART 位皆会被硬件清除。

14. 5 特殊功能寄存器

14. 5. 1 寄存器列表

ADC 寄存器列表			
名称	偏移地址	类型	描述
ADC_IER	0000_H	W1	ADC 中断开启寄存器
ADC_IDR	0004_H	W1	ADC 中断关闭寄存器
ADC_IVS	0008_H	R	ADC 中断功能有效位状态寄存器
ADC_RIF	$000C_H$	R	ADC 原始中断状态寄存器
ADC_IFM	0010_H	R	ADC 中断旗标状态寄存器
ADC_ICR	0014_H	C_W1	ADC 中断清除寄存器
ADC_CON	0018_H	R/W	ADC 控制寄存器
ADC_CFG1	$001C_H$	R/W	ADC 配置寄存器 1
ADC_CFG2	0020_H	R/W	ADC 配置寄存器 2
ADC_SMPT1	0024_H	R/W	ADC 采样时间寄存器 1
ADC_SMPT2	0028_H	R/W	ADC 采样时间寄存器 2
ADC_SMPT3	$002C_H$	R/W	ADC 采样时间寄存器 3
ADC_SMPT4	0030_H	R/W	ADC 采样时间寄存器 4
ADC_SMPT5	0034_H	R/W	ADC 采样时间寄存器 5
ADC_TR1	0038_H	R/W	ADC 模拟看门狗 1 阀值寄存器
ADC_TR2	$003C_H$	R/W	ADC 模拟看门狗 2 阀值寄存器
ADC_TR3	0040_H	R/W	ADC 模拟看门狗 3 阀值寄存器
ADC_SQR1	0044_H	R/W	ADC 标准通道序列寄存器 1
ADC_SQR2	0048_H	R/W	ADC 标准通道序列寄存器 2
ADC_SQR3	$004C_H$	R/W	ADC 标准通道序列寄存器 3
ADC_SQR4	0050_H	R/W	ADC 标准通道序列寄存器 4
ADC_DR	0054_H	R	ADC 标准数据寄存器
ADC_JSQR	$005C_H$	R/W	ADC 插入通道序列寄存器
ADC_OFST1	0060_H	R/W	ADC 偏移寄存器 1
ADC_OFST2	0064_H	R/W	ADC 偏移寄存器 2
ADC_OFST3	0068_H	R/W	ADC 偏移寄存器 3
ADC_OFST4	$006C_H$	R/W	ADC 偏移寄存器 4
ADC_JDR1	0070_H	R	ADC 插入数据寄存器 1
ADC_JDR2	0074_H	R	ADC 插入数据寄存器 2
ADC_JDR3	0078_H	R	ADC 插入数据寄存器 3
ADC_JDR4	$007C_H$	R	ADC 插入数据寄存器 4
ADC_AWD2CR	0080_H	R/W	ADC 模拟看门狗 2 配置寄存器

ADC 寄存器列表			
名称	偏移地址	类型	描述
ADC_AWD3CR	0084_H	R/W	ADC 模拟看门狗 3 配置寄存器
ADC_GCOMP	0088_H	R/W	ADC 增益系数寄存器

ADC 通用寄存器列表			
名称	偏移地址	类型	描述
ADC_CSR	0000_H	R	ADC 通用状态寄存器
ADC_CCR	0004_H	R/W	ADC 通用控制寄存器
ADC_CDR	0008_H	R	ADC 通道数据寄存器

14. 5. 2 寄存器描述

14. 5. 2. 1 ADC中断开启寄存器(ADC_IER)

ADC 中断开启寄存器(ADC_IER)																												
偏移地址:0x00																												
复位值:0x0000 0000																												
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																TO					JQOVF	AWD3	AWD2	AWD1	JEOS	JEOC	ROVR	REOS	REOC	EOSMP	

—	Bits 31-16	—	—
TO	Bit 15	W1	**通道转换超时中断开启** 0: 写入 0 无效 1: 通道转换超时中断中断开启
—	Bit 14-11	—	—
JQOVF	Bit 10	W1	**插入寄存器队列溢出中断开启** 0: 写 0 无效 1: 插入寄存器队列溢出中断开启
AWD3	Bit 9	W1	**模拟看门狗 3 中断开启** 0: 写 0 无效 1: 模拟看门狗 3 中断开启
AWD2	Bit 8	W1	**模拟看门狗 2 中断开启** 0: 写 0 无效 1: 模拟看门狗 2 中断开启
AWD1	Bit 7	W1	**模拟看门狗 1 中断开启** 0: 写 0 无效 1: 模拟看门狗 1 中断开启
JEOS	Bit 6	W1	**插入通道序列结束中断开启** 0: 写 0 无效 1: 插入通道序列结束中断开启
JEOC	Bit 5	W1	**插入通道转换结束中断开启** 0: 写 0 无效 1: 插入通道转换结束中断开启
ROVR	Bit 4	W1	**标准数据溢出中断开启** 0: 写 0 无效 1: 标准数据溢出中断开启
REOS	Bit 3	W1	**标准通道序列结束中断开启**

			0: 写 0 无效 1: 标准通道序列结束中断开启
REOC	Bit 2	W1	**标准通道转换结束中断开启** 0: 写 0 无效 1: 标准通道转换结束中断开启
EOSMP	Bit 1	W1	**采样结束中断开启** 0: 写 0 无效 1: 采样结束中断中断开启
—	Bits 0	—	—

14. 5. 2. 2 ADC中断关闭寄存器(ADC_IDR)

ADC 中断关闭寄存器(ADC_IDR)																												
偏移地址:0x04																												
复位值:0x0000 0000																												
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																TO					JQOVF	AWD3	AWD2	AWD1	JEOS	JEOC	ROVR	REOS	REOC	EOSMP	

—	Bits 31-16	—	—
TO	Bit 15	W1	**通道转换超时中断关闭** 0: 写入 0 无效 1: 通道转换超时中断中断关闭
—	Bits 14-11	—	—
JQOVF	Bit 10	W1	**插入寄存器队列溢出中断关闭** 0: 写 0 无效 1: 插入寄存器队列溢出中断关闭
AWD3	Bit 9	W1	**模拟看门狗 3 中断关闭** 0: 写 0 无效 1: 模拟看门狗 3 中断关闭
AWD2	Bit 8	W1	**模拟看门狗 2 中断关闭** 0: 写 0 无效 1: 模拟看门狗 2 中断关闭
AWD1	Bit 7	W1	**模拟看门狗 1 中断关闭** 0: 写 0 无效 1: 模拟看门狗 1 中断关闭
JEOS	Bit 6	W1	**插入通道序列结束中断关闭**

			0: 写 0 无效 1: 插入通道序列结束中断关闭
JEOC	Bit 5	W1	**插入通道转换结束中断关闭** 0: 写 0 无效 1: 插入通道转换结束中断关闭
ROVR	Bit 4	W1	**标准数据溢出中断关闭** 0: 写 0 无效 1: 标准数据溢出中断关闭
REOS	Bit 3	W1	**标准通道序列结束中断关闭** 0: 写 0 无效 1: 标准通道序列结束中断关闭
REOC	Bit 2	W1	**标准通道转换结束中断关闭** 0: 写 0 无效 1: 标准通道转换结束中断关闭
EOSMP	Bit 1	W1	**采样结束中断关闭** 0: 写 0 无效 1: 采样结束中断中断关闭
—	Bits 0	—	—

14. 5. 2. 3 ADC中断功能有效状态寄存器(ADC_IVS)

ADC 中断功能有效状态寄存器(ADC_IVS)

偏移地址:0x08

复位值:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	TO	—	—	—	—	JQOVF	AWD3	AWD2	AWD1	JEOS	JEOC	ROVR	REOS	REOC	EOSMP	—

—	Bits 31-16	—	—
TO	Bit 15	R	**通道转换超时中断功能有效状态** 0: 通道转换超时中断功能处于关闭状态 1: 通道转换超时中断功能处于开启状态
—	Bits 14-11	—	—
JQOVF	Bit 10	R	**插入寄存器队列溢出中断功能有效状态** 0: 插入寄存器队列溢出中断功能处于关闭状态 1: 插入寄存器队列溢出中断功能处于开启状态
AWD3	Bit 9	R	**仿真看门狗 3 中断功能有效状态**

			0: 仿真看门狗 3 中断功能处于关闭状态 1: 仿真看门狗 3 中断功能处于开启状态
AWD2	Bit 8	R	**仿真看门狗 2 中断功能有效状态** 0: 仿真看门狗 2 中断功能处于关闭状态 1: 仿真看门狗 2 中断功能处于开启状态
AWD1	Bit 7	R	**仿真看门狗 1 中断功能有效状态** 0: 仿真看门狗 1 中断功能处于关闭状态 1: 仿真看门狗 1 中断功能处于开启状态
JEOS	Bit 6	R	**插入通道序列结束中断功能有效状态** 0: 插入通道序列结束中断功能处于关闭状态 1: 插入通道序列结束中断功能处于开启状态
JEOC	Bit 5	R	**插入通道转换结束中断功能有效状态** 0: 插入通道转换结束中断功能处于关闭状态 1: 插入通道转换结束中断功能处于开启状态
ROVR	Bit 4	R	**标准数据溢出中断功能有效状态** 0: 标准数据溢出中断功能处于关闭状态 1: 标准数据溢出中断功能处于开启状态
REOS	Bit 3	R	**标准通道序列结束中断功能有效状态** 0: 标准通道序列结束中断功能处于关闭状态 1: 标准通道序列结束中断功能处于开启状态
REOC	Bit 2	R	**标准通道转换结束中断功能有效状态** 0: 标准通道转换结束中断功能处于关闭状态 1: 标准通道转换结束中断功能处于开启状态
EOSMP	Bit 1	R	**采样结束中断功能有效状态** 0: 采样结束中断功能处于关闭状态 1: 采样结束中断功能处于开启状态
—	Bit 0	—	—

14. 5. 2. 4 ADC原始中断状态寄存器 (ADC_RIF)

ADC 原始中断状态寄存器(ADC_RIF)																												
偏移地址:0x0C																												
复位值:0x0000 0000																												
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	TO	—	—	—	—	JQOVF	AWD3	AWD2	AWD1	JEOS	JEOC	ROVR	REOS	REOC	EOSMP	—

—	Bits 31-16	—	—
TO	Bit 15	R	**通道转换超时原始中断状态** 0: 无发生中断 1: 已发生中断
—	Bits 14-11	—	—
JQOVF	Bit 10	R	**插入寄存器队列溢出原始中断状态** 0: 无发生中断 1: 已发生中断
AWD3	Bit 9	R	**模拟看门狗 3 原始中断状态** 0: 无发生中断 1: 已发生中断
AWD2	Bit 8	R	**模拟看门狗 2 原始中断状态** 0: 无发生中断 1: 已发生中断
AWD1	Bit 7	R	**模拟看门狗 1 原始中断状态** 0: 无发生中断 1: 已发生中断
JEOS	Bit 6	R	**插入通道序列结束原始中断状态** 0: 无发生中断 1: 已发生中断
JEOC	Bit 5	R	**插入通道转换结束原始中断状态** 0: 无发生中断 1: 已发生中断
ROVR	Bit 4	R	**标准数据溢出原始中断状态** 0: 无发生中断 1: 已发生中断
REOS	Bit 3	R	**标准通道序列结束原始中断状态** 0: 无发生中断

			1: 已发生中断
REOC	Bit 2	R	**标准通道转换结束原始中断状态** 0: 无发生中断 1: 已发生中断
EOSMP	Bit 1	R	**采样结束原始中断状态** 0: 无发生中断 1: 已发生中断
—	Bit 0	—	—

14. 5. 2. 5 ADC中断标志位状态寄存器(ADC_IFM)

ADC 中断标志位状态寄存器(ADC_IFM)																												
偏移地址:0x10																												
复位值:0x0000 0000																												
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	TO	—	—	—	—	JQOVF	AWD3	AWD2	AWD1	JEOS	JEOC	ROVR	REOS	REOC	EOSMP	—

—	Bits 31-16	—	—
TO	Bit 15	R	**通道转换超时中断旗标状态** 0: 无发生中断或没有开启中断 1: 已发生中断
—	Bits 14-11	—	—
JQOVF	Bit 10	R	**插入寄存器队列溢出中断旗标状态** 0: 无发生中断或没有开启中断 1: 已发生中断
AWD3	Bit 9	R	**模拟看门狗 3 中断旗标状态** 0: 无发生中断或没有开启中断 1: 已发生中断
AWD2	Bit 8	R	**模拟看门狗 2 中断旗标状态** 0: 无发生中断或没有开启中断 1: 已发生中断
AWD1	Bit 7	R	**模拟看门狗 1 中断旗标状态** 0: 无发生中断或没有开启中断 1: 已发生中断
JEOS	Bit 6	R	**插入通道序列结束中断旗标状态** 0: 无发生中断或没有开启中断

			1: 已发生中断
JEOC	Bit 5	R	**插入通道转换结束中断旗标状态** 0: 无发生中断或没有开启中断 1: 已发生中断
ROVR	Bit 4	R	**标准数据溢出中断旗标状态** 0: 无发生中断或没有开启中断 1: 已发生中断
REOS	Bit 3	R	**标准通道序列结束中断旗标状态** 0: 无发生中断或没有开启中断 1: 已发生中断
REOC	Bit 2	R	**标准通道转换结束中断旗标状态** 0: 无发生中断或没有开启中断 1: 已发生中断
EOSMP	Bit 1	R	**采样结束中断旗标状态** 0: 无发生中断或没有开启中断 1: 已发生中断
—	Bit 0	—	—

14. 5. 2. 6 ADC中断清除寄存器(ADC_ICR)

ADC 中断清除寄存器(ADC_ICR)

偏移地址:0x14

复位值:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	TO	—	—	—	—	JQOVF	AWD3	AWD2	AWD1	JEOS	JEOC	ROVR	REOS	REOC	EOSMP	—

—	Bits 31-16	—	—
TO	Bit 15	C_W1	**通道转换超时中断清除** 0: 写 0 无效 1: 清除中断事件
—	Bits 14-11	—	—
JQOVF	Bit 10	C_W1	**插入寄存器队列溢出中断清除** 0: 写 0 无效 1: 清除中断事件
AWD3	Bit 9	C_W1	**模拟看门狗 3 中断清除** 0: 写 0 无效

			1: 清除中断事件
AWD2	Bit 8	C_W1	**模拟看门狗 2 中断清除** 0: 写 0 无效 1: 清除中断事件
AWD1	Bit 7	C_W1	**模拟看门狗 1 中断清除** 0: 写 0 无效 1: 清除中断事件
JEOS	Bit 6	C_W1	**插入通道序列结束中断清除** 0: 写 0 无效 1: 清除中断事件
JEOC	Bit 5	C_W1	**插入通道转换结束中断清除** 0: 写 0 无效 1: 清除中断事件
ROVR	Bit 4	C_W1	**标准数据溢出中断清除** 0: 写 0 无效 1: 清除中断事件
REOS	Bit 3	C_W1	**标准通道序列结束中断清除** 0: 写 0 无效 1: 清除中断事件
REOC	Bit 2	C_W1	**标准通道转换结束中断清除** 0: 写 0 无效 1: 清除中断事件
EOSMP	Bit 1	C_W1	**采样结束中断清除** 0: 写 0 无效 1: 清除中断事件
—	Bit 0	—	—

14. 5. 2. 7 ADC控制寄存器(ADC_CON)

ADC 控制寄存器 (ADC_CON)																													
偏移地址:0x18																													
复位值:0x0000 0000																													
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
															LP_EN											JSTP	RSTP	JSTART	RSTART	ADCDIS	ADCEN

—	Bits 31-17	—	—
LP_EN	Bit 16	R/W	**ADC 低功率模式开关** 在 ADC 时钟低于 1 MHz，可选择开启 ADC 低功率模式。 0: ADC 低功率模式关闭 1: ADC 低功率模式开启 ADC 电压稳压器开启以及关闭的细节，可以参照 datasheet 的说明。 注: JSTART = 0、RSTART = 0、RSTP = 0、JSTP = 0、ADCDIS = 0 以及 ADCEN = 0 的时候，才允许写入配置。
—	Bits 15-6	—	—
JSTP	Bit 5	R/W1	**ADC 停止插入转换指令** 停止或是丢弃正在进行的插入转换。 当 ADC 的插入转换已经被终止，寄存器状态会藉由硬件清除。ADC 插入的序列以及触发条件都可以重新设定。ADC 准备接受新的 JSTART 的指令。 0: 没有停止插入转换的指令正在进行 1: 写 1 停止插入转换。读取状态为 1，表示 ADC 正在进行停止插入转换的指令 注: JSTART = 1 以及 ADCDIS = 0 (表示 ADC 正在运行等待插入通道的转换，并且没有 ADC 关闭的请求)，才允许设置 JSTP。 注: 使用自动插入模式(JAUTO = 1)，设置 RSTP，将使得标准转换以及插入转换一同关闭(不能使用 JSTP)。
RSTP	Bit 4	R/W1	**ADC 停止标准转换指令**

			停止或是丢弃正在进行的标准转换。 当 ADC 的标准转换已经被终止，寄存器状态会藉由硬件清除。ADC 标准的序列以及触发条件都可以重新设定。ADC 准备接受新的 RSTART 的指令。 0: 没有停止标准转换的指令正在进行 1: 写 1 停止标准转换。读取状态为 1，表示 ADC 正在进行停止标准转换的指令 注: RSTART = 1 以及 ADCDIS = 0 (表示 ADC 正在运行等待标准通道的转换，并且没有 ADC 关闭的请求)，才允许设置 RSTP。 注: 使用自动插入模式(JAUTO = 1)，设置 RSTP，将使得标准转换以及插入转换一同关闭(不能使用 JSTP)。 注: 双重 ADC 的标准同步以及交错模式，必须使用 ADC1 的 RSTP 停止标准转换，ADC2 的 RSTP 并不能起作用。
JSTART	Bit 3	R/W1	**ADC 启动插入转换** 启动插入通道的 ADC 转换。视寄存器 JEXTEN[1:0]的设定，转换会立即开始(软件触发) 或是等待触发事件(硬件触发)。 寄存器状态藉由硬件清除: – 在单次转换模式，选择软件触发 (JEXTEN[1:0] = 0x0)，发生插入通道序列结束 (JEOS)，该位会通过硬件清除。 – 所有模式，执行 JSTP 的指令，同时间 JSTP 与该位会通过硬件清除。 0: 没有正在进行的插入转换 1: 写 1 启动插入转换。读取状态为 1，表示 ADC 正在运行等待插入通道的转换 注: ADCEN = 1 以及 ADCDIS = 0 (ADC 已经开启，并且没有关闭 ADC 的请求)，才允许设置 JSTART。 注: 使用自动插入模式(JAUTO = 1)，是使用 RSTART 启动，该位要保持清除状态。
RSTART	Bit 2	R/W1	**ADC 启动标准转换** 启动标准通道的 ADC 转换。视寄存器 REXTEN[1:0]的设定，转换会立即开始(软件触

			发)或是等待触发事件(硬件触发)。 寄存器状态藉由硬件清除： – 在单次转换模式(RCONT = 0、DISCEN = 0)，选择软件触发(REXTEN[1:0] = 0x0)，发生标转通道序列结束(REOS)，该位会通过硬件清除。 – 在不连续的模式(RCONT = 0、DISCEN = 1)，选择软件触发(REXTEN[1:0] = 0x0)，发生标准通道转换结束(REOC)，该位会通过硬件清除。 – 所有模式，执行 RSTP 的指令，同时间 RSTP 与该位会通过硬件清除。 0: 没有正在进行的标准转换 1: 写 1 启动标准转换。读取状态为 1，表示 ADC 正在运行等待标准通道的转换 注: ADCEN = 1 以及 ADCDIS = 0 (ADC 已经开启，并且没有关闭 ADC 的请求)，才允许设置 RSTART。 注: 使用自动插入模式(JAUTO = 1)，藉由设置 RSTART 开始标准转换和自动插入转换(JSTART 必须保持清除状态)。
ADCDIS	Bit 1	R/W1	**ADC 关闭控制** 关闭 ADC，使得 ADC 进入掉电的状态(OFF state)。 当 ADC 已经确定处于关闭状态，硬件会清除 ADCDIS 的状态(同 ADCEN 会被硬件清除)。 0: 没有正在执行 ADCDIS 的指令 1: 写 1 关闭 ADC。读取状态为 1，表示正在执行 ADCDIS 的指令 注: ADCEN = 1、JSTART = 0 以及 RSTART= 0(为了确保没有转换正在进行)，才允许设置 ADCDIS。
ADCEN	Bit 0	R/W1	**ADC 开启控制** 开启 ADC，当 ADCRDY 状态发生后，代表 ADC 已经稳定。 藉由 ADCDIS 的寄存器设定，硬件会清除 ADCEN 的状态。 0: 写 0 无效，ADC 关闭的状态

			1: 写 1 开启 ADC 注: JSTART = 0、RSTART = 0、RSTP = 0、JSTP = 0、ADCDIS = 0 以及 ADCEN = 0 的时候，才允许设置 ADCEN。(软件必须等待电压稳压器的启动时间)

14. 5. 2. 8 ADC配置寄存器 1(ADC_CFG1)

ADC 配置寄存器 1 (ADC_CFG1)

偏移地址:0x1C

复位值:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
JQEN	AWD1CH<4:0>					JAUTO	AWD1JEN	AWD1REN	AWD1SEL	JQM	JDISCEN	DISCNUM<2:0>			DISCEN	ALIGN	AUTODLY	RCONT	OVRRMOD	REXTEN<1:0>		REXTSEL<4:0>					RESOL<1:0>				DMAEN

JQEN	Bit 31	R/W	**插入队列开关** 0: 插入队列关闭。 1: 插入队列开启。 注: RSTART = 0 以及 JSTART = 0 (确保没有正在进行中的转换)，才允许写入配置。 注: JQEN 清除或是复位，会导致插入队列被清空，JSQR 也会被清空。
AWD1CH	Bit 30-26	R/W	**模拟看门狗 1 的监控通道选择** 当 AWD1SEL = 1，模拟看门狗监控单一通道的选择。 00000: 监控 ADC 模拟输入通道 0 00001: 监控 ADC 模拟输入通道 1 00010: 监控 ADC 模拟输入通道 2 … 10010: 监控 ADC 模拟输入通道 18 注: RSTART = 0 以及 JSTART = 0 (确保没有正在进行中的转换)，才允许写入配置。 注: AWD1CH 通道的选择，必须在寄存器 SQR 以及 JSQR 也有设定的通道。
JAUTO	Bit 25	R/W	**自动插入转换模式开关** 标准通道序列结束后，自动进行插入转换的模式开关。

			0: 自动插入转换模式关闭。 1: 自动插入转换模式开启。 注: JSTART = 0 (确保没有正在进行中的插入转换)，才允许写入配置。 注: 双重 ADC 模式，ADC2 的 JAUTO 是无法写入，其数值将与 ADC1 的 JAUTO 相同。
AWD1JEN	Bit 24	R/W	**插入通道的模拟看门狗 1 开关** 0: 插入通道的模拟看门狗 1 功能关闭。 1: 插入通道的模拟看门狗 1 功能开启。 注: JSTART = 0(确保没有正在进行中的插入转换)，才允许写入配置。
AWD1REN	Bit 23	R/W	**标准通道的模拟看门狗 1 开关** 0: 标准通道的模拟看门狗 1 功能关闭。 1: 标准通道的模拟看门狗 1 功能开启。 注: RSTART = 0(确保没有正在进行中的标准转换)，才允许写入配置。
AWD1SEL	Bit 22	R/W	**模拟看门狗 1 的监控通道选择** 设置模拟看门狗 1 监控藉由 AWD1CH[4:0]选择单一的通道或是全部的通道。 0: 模拟看门狗 1 监控所有通道。 1: 模拟看门狗 1 监控单一通道。 注: RSTART = 0 以及 JSTART = 0 (确保没有正在进行中的转换)，才允许写入配置。
JQM	Bit 21	R/W	**JSQR 队列模式** 设定队列为空的数据管理。 0: JSQR 队列不会为空，维持最后一次写入 JSQR 的配置。 1: JSQR 队列会为空，当最后一次的插入序列完成后，将会关闭软件以及硬件的外部触发。 注: JSTART = 0(确保没有正在进行中的插入转换)，才允许写入配置。 注: 双重 ADC 模式，ADC2 的 JQM 是无法写入，其数值将与 ADC1 的 JQM 相同。
JDISCEN	Bit 20	R/W	**插入通道的非连续模式开关** 开启或关闭插入通道的非连续模式。 0: 插入通道的非连续模式关闭 1: 插入通道的非连续模式开启 注: JSTART = 0 (确保没有正在进行中的插入

			转换)，才允许写入配置。 注: 不能够同时存在自动插入以及非连续模式，所以 JAUTO = 1 的情况下，DSICEN 以及 JDISCEN 必须维持清除的状态。 注: 双重 ADC 模式，ADC2 的 JDISCEN 是无法写入，其数值将与 ADC1 的 JDISCEN 相同。
DISCNUM	Bit 19-17	R/W	**非连续模式的标准转换通道数量** 非连续模式下，外部触发发生后，启动标准转换的通道数量设定值。 000: 1 个通道 001: 2 个通道 010: 3 个通道 … 111: 8 个通道 注: RSTART = 0 (确保没有正在进行中的标准转换)，才允许写入配置。 注: 双重 ADC 模式，ADC2 的 DISCNUM[2:0] 是无法写入，其数值将与 ADC1 的 DISCNUM[2:0]相同。
DISCEN	Bit 16	R/W	**标准通道的非连续模式开关** 开启或关闭标准通道的非连续模式。 0: 标准通道的非连续模式关闭 1: 标准通道的非连续模式开启 注: RSTART = 0 (确保没有正在进行中的标准转换)，才允许写入配置。 注: 不能够同时存在非连续以及连续模式，所以禁止同时设定 DISCEN = 1 以及 RCONT = 1。 注: 不能够同时存在自动插入以及非连续模式，所以 AUTO = 1 的情况下，DSICEN 以及 JDISCEN 必须维持清除的状态。 注: 双重 ADC 模式，ADC2 的 DISCEN 是无法写入，其数值将与 ADC1 的 DISCEN 相同。
ALIGN	Bit 15	R/W	**数据对齐模式** 选择转换后的数据为左对齐或是又对齐，转换后的数据将存放在 ADC_DR。 0: 右对齐 1: 左对齐

			注: RSTART = 0 以及 JSTART = 0 (确保没有正在进行中的转换)，才允许写入配置。
AUTODLY	Bit 14	R/W	**延迟转换模式** 开启或关闭自动延迟转换模式。 0: 自动延迟转换模式关闭 1: 自动延迟转换模式开启 注: RSTART = 0 以及 JSTART = 0 (确保没有正在进行中的转换)，才允许写入配置。 注 ：双重 ADC 模式，ADC2 的 AUTODLY 是无法写入，其数值将与 ADC1 的 AUTODLY 相同。
RCONT	Bit 13	R/W	**标准连续转换模式选择** 设置此寄存器，会连续进行标准转换，直到清除该位。 0: 单次标准转换 1: 连续标准转换 注: RSTART = 0(确保没有正在进行中的标准转换)，才允许写入配置。 注: 不能够同时存在非连续以及连续模式，所以禁止同时设定 DISCEN = 1 以及 RCONT = 1。 注: 双重 ADC 模式，ADC2 的 RCONT 是无法写入，其数值将与 ADC1 的 RCONT 相同。
OVRRMOD	Bit 12	R/W	**标准数据溢出模式** 标准通道转换后的数据溢出管理方式。 0 ：侦测到数据溢出，ADC_DR 将保留旧有的数据，最新的转换数据将会被丢弃 1 ：侦测到数据溢出，ADC_DR 将覆盖成最新的转换的数据 注: RSTART = 0(确保没有正在进行中的标准转换)，才允许写入配置。
REXTEN	Bit 11-10	R/W	**标准外部触发开启以及极性选择** 标准的外部触发选择，开启硬件以及极性选择。 00: 硬件侦测触发关闭(允许软件触发启动转换) 01: 侦测事件上升沿，产生硬件触发 10: 侦测事件下降沿，产生硬件触发 11: 侦测事件上升以及下降沿，产生硬件触发 注: RSTART = 0(确保没有正在进行中的标准

			转换)，才允许写入配置。
REXTSEL	Bit 9-5	R/W	**标准外部触发事件选择** 选择启动标准转换的外部触发事件。可参照表 14-4 00000: 事件 0 00001: 事件 1 00010: 事件 2 00011: 事件 3 00100: 事件 4 00101: 事件 5 … 11111: 事件 31 注: RSTART = 0(确保没有正在进行中的标准转换)，才允许写入配置。
RESOL	Bit 4-3	R/W	**ADC 数据分辨率** 选择转换后的数据分辨率。 00: 12 bit 01: 10 bit 10: 8 bit 11: 6 bit 注: RSTART = 0 以及 JSTART = 0(确保没有正在进行中的转换)，才允许写入配置。
—	Bits 2-1	—	—
DMAEN	Bit 0	R/W	**DMA 开关** 0: 关闭 DMA 功能 1: 开启 DMA 功能 注: RSTART = 0 以及 JSTART = 0(确保没有正在进行中的转换)，才允许写入配置。 注：双重 ADC 的模式，由通用寄存器 ADC_CCR 的 MDMA[1:0]进行 DMA 的开关。该位将无意义。

14. 5. 2. 9 ADC配置寄存器 2(ADC_CFG2)

ADC 配置寄存器 2 (ADC_CFG2)																												
偏移地址:0x20																												
复位值:0x0000 0000																												
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
				SMPTRIG	BULB	SWTRIG									GCOMP						ROVSM	TROVS	OVSSHIFT<3:0>				OVSRATIO<2:0>			JOVSEN	ROVSEN

—	Bits 31-28	—	—
SMPTRIG	Bit 27	R/W	**采样时间控制触发模式** 0: 采样时间控制触发模式关闭 1: 采样时间控制触发模式开启 EXTEN[1:0]设置为 01，采样启动于触发的上升沿，转换启动于触发的下降沿。 EXTEN[1:0] 设置为 00，采样启动于设置 SWTRIG = 1，转换启动于清除 SWTRIG = 0。 注: RSTART = 0(确保没有正在进行中的标准转换)，才允许写入配置。 注: 当 BULB = 1，SMPTRIG 是不能被设定的。
BULB	Bit 26	R/W	**快门(Bulb)采样模式** 0: 快门采样模式关闭 1: 快门采样模式开启 第一个 ADC 转换的采样时间是基于 SMPx 的设置。 注: RSTART = 0 (确保没有正在进行中的标准转换)，才允许写入配置。 注: 当 BULB = 1，SMPTRIG 是不能被设定的。
SWTRIG	Bit 25	R/W	**采样时间控制触发模式的软件控制** 0: 采样时间控制触发模式的软件触发开始转换 1: 采样时间控制触发模式的软件触发开始采样 注: RSTART = 0 (确保没有正在进行中的标准转换)，才允许写入配置。
—	Bits 24-17	—	—
GCOMP	Bit 16	R/W	**增益补偿模式** 0: 一般运算模式 1: 所有运行通道的增益补偿开启

			注: RSTART = 0 以及 JSTART = 0(确保没有正在进行中的转换)，才允许写入配置。
—	Bits 15-11	—	—
ROVSM	Bit 10	R/W	**标准过采样模式** 0: 连续模式: 当插入转换被触发，当前的过采样暂时停止，等插入序列完成后，过采样从中断的地方开始进行 1: 恢复模式: 当插入转换被触发，当前的过采样直接中止，等插入序列完成后，过采样从头开始进行 注: RSTART = 0 以及 JSTART = 0(确保没有正在进行中的转换)，才允许写入配置。
TROVS	Bit 9	R/W	**过采样触发模式** 0: 一个通道的所有过采样转换在一个触发后接连完成 1: 一个通道的每个过采样转换都需要新的触发 注: RSTART = 0 以及 JSTART = 0 (确保没有正在进行中的转换)，才允许写入配置。
OVSSHIFT	Bit 8-5	R/W	**过采样数据位移** 设定采样过后的数据，向右移的 bit 数。 0000: No shifts 0001: Shift 1-bits 0010: Shift 2-bits 0011: Shift 3-bits 0100: Shift 4-bits 0101: Shift 5-bits 0110: Shift 6-bits 0111: Shift 7-bits 1000: Shift 8-bits Other: 保留 注: RSTART = 0 以及 JSTART = 0 (确保没有正在进行中的转换)，才允许写入配置。
OVSRATIO	Bit 4-2	R/W	**过采样率** 000: 2x 001: 4x 010: 8x 011: 16x 100: 32x

			101: 64x 110: 128x 111: 256x 注: RSTART = 0 以及 JSTART = 0 (确保没有正在进行中的转换)，才允许写入配置。
JOVSEN	Bit 1	R/W	**插入过采样开关** 0: 关闭插入过采样功能 1: 开启插入过采样功能 注: RSTART = 0 以及 JSTART = 0(确保没有正在进行中的转换)，才允许写入配置。
ROVSEN	Bit 0	R/W	**标准过采样开关** 0: 关闭标准过采样功能 1: 开启标准过采样功能 注: RSTART = 0 以及 JSTART = 0 (确保没有正在进行中的转换)，才允许写入配置。

14. 5. 2. 10 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:0x24

复位值:0x0000 0000

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
SMP3<7:0>								SMP2<7:0>								SMP1<7:0>								SMP0<7:0>							

SMP3	Bit 31-24	R/W	**通道 3 的采样时间设定** 设定通道的采样时间设定，时间的单位为 ADC 时钟。 T_{SMP} 的公式可以参照章节 14. 4. 5 采样时间 = T_{SMP} + 3.5 ADC 时钟周期。 注: RSTART = 0 以及 JSTART = 0 (确保没有正在进行中的转换)，才允许写入配置。
SMP2	Bit 23-16	R/W	**通道 2 的采样时间设定** 设定通道的采样时间设定，时间的单位为 ADC 时钟。 T_{SMP} 的公式可以参照章节 14. 4. 5 采样时间 = T_{SMP} + 3.5 ADC 时钟周期。

			注: RSTART = 0 以及 JSTART = 0 (确保没有正在进行中的转换)，才允许写入配置。
SMP1	Bit 15-8	R/W	**通道 1 的采样时间设定** 设定通道的采样时间设定，时间的单位为 ADC 时钟。 T_{SMP} 的公式可以参照章节 14. 4. 5 采样时间 = T_{SMP} + 3.5 ADC 时钟周期。 注: RSTART = 0 以及 JSTART = 0(确保没有正在进行中的转换)，才允许写入配置。
SMP0	Bit 7-0	R/W	**通道 0 的采样时间设定** 设定通道的采样时间设定，时间的单位为 ADC 时钟。 T_{SMP} 的公式可以参照章节 14. 4. 5 采样时间 = T_{SMP} + 3.5 ADC 时钟周期。 注: RSTART = 0 以及 JSTART = 0 (确保没有正在进行中的转换)，才允许写入配置。

14. 5. 2. 11 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:0x28

复位值:0x0000 0000

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
SMP7<7:0>								SMP6<7:0>								SMP5<7:0>								SMP4<7:0>							

SMP7	Bit 31-24	R/W	**通道 7 的采样时间设定** 设定通道的采样时间设定，时间的单位为 ADC 时钟。 T_{SMP} 的公式可以参照章节 14. 4. 5 采样时间 = T_{SMP} + 3.5 ADC 时钟周期。 注: RSTART = 0 以及 JSTART = 0 (确保没有正在进行中的转换)，才允许写入配置。
SMP6	Bit 23-16	R/W	**通道 6 的采样时间设定** T_{SMP} 的公式可以参照章节 14. 4. 5 采样时间 = T_{SMP} + 3.5 ADC 时钟周期。 注: RSTART = 0 以及 JSTART = 0 (确保没有正

			在进行中的转换)，才允许写入配置。
SMP5	Bit 15-8	R/W	**通道 5 的采样时间设定** T_{SMP} 的公式可以参照章节 14. 4. 5 采样时间 = T_{SMP} + 3.5 ADC 时钟周期。 注: RSTART = 0 以及 JSTART = 0 (确保没有正在进行中的转换)，才允许写入配置。
SMP4	Bit 7-0	R/W	**通道 4 的采样时间设定** T_{SMP} 的公式可以参照章节 14. 4. 5 采样时间 = T_{SMP} + 3.5 ADC 时钟周期。 注: RSTART = 0 以及 JSTART = 0 (确保没有正在进行中的转换)，才允许写入配置。

14. 5. 2. 12 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:0x2C

复位值:0x0000 0000

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
SMP11<7:0>								SMP10<7:0>								SMP9<7:0>								SMP8<7:0>							

SMP11	Bit 31-24	R/W	**通道 11 的采样时间设定** 设定通道的采样时间设定，时间的单位为 ADC 时钟。 T_{SMP} 的公式可以参照章节 14. 4. 5 采样时间 = T_{SMP} + 3.5 ADC 时钟周期。 注: RSTART = 0 以及 JSTART = 0 (确保没有正在进行中的转换)，才允许写入配置。
SMP10	Bit 23-16	R/W	**通道 10 的采样时间设定** T_{SMP} 的公式可以参照章节 14. 4. 5 采样时间 = T_{SMP} + 3.5 ADC 时钟周期。 注: RSTART = 0 以及 JSTART = 0 (确保没有正在进行中的转换)，才允许写入配置。
SMP9	Bit 15-8	R/W	**通道 9 的采样时间设定** T_{SMP} 的公式可以参照章节 14. 4. 5 采样时间 = T_{SMP} + 3.5 ADC 时钟周期。 注: RSTART = 0 以及 JSTART = 0 (确保没有正

			在进行中的转换)，才允许写入配置。
SMP8	Bit 7-0	R/W	**通道 8 的采样时间设定** T_{SMP} 的公式可以参照章节 14. 4. 5 采样时间 = T_{SMP} + 3.5 ADC 时钟周期。 注: RSTART = 0 以及 JSTART = 0 (确保没有正在进行中的转换)，才允许写入配置。

14. 5. 2. 13 ADC采样时间寄存器 4(ADC_SMPT4)

ADC 采样时间寄存器 4 (ADC_SMPT4)

偏移地址:0x30

复位值:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SMP15<7:0>								SMP14<7:0>								SMP13<7:0>								SMP12<7:0>							

SMP15	Bit 31-24	R/W	**通道 15 的采样时间设定** 设定通道的采样时间设定，时间的单位为 ADC 时钟。 T_{SMP} 的公式可以参照章节 14. 4. 5 采样时间 = T_{SMP} + 3.5 ADC 时钟周期。 注: RSTART = 0 以及 JSTART = 0 (确保没有正在进行中的转换)，才允许写入配置。
SMP14	Bit 23-16	R/W	**通道 14 的采样时间设定** T_{SMP} 的公式可以参照章节 14. 4. 5 采样时间 = T_{SMP} + 3.5 ADC 时钟周期。 注: RSTART = 0 以及 JSTART = 0 (确保没有正在进行中的转换)，才允许写入配置。
SMP13	Bit 15-8	R/W	**通道 13 的采样时间设定** T_{SMP} 的公式可以参照章节 14. 4. 5 采样时间 = T_{SMP} + 3.5 ADC 时钟周期。 注: RSTART = 0 以及 JSTART = 0 (确保没有正在进行中的转换)，才允许写入配置。
SMP12	Bit 7-0	R/W	**通道 12 的采样时间设定** T_{SMP} 的公式可以参照章节 14. 4. 5 采样时间 = T_{SMP} + 3.5 ADC 时钟周期。 注: RSTART = 0 以及 JSTART = 0 (确保没有正

			在进行中的转换)，才允许写入配置。

14. 5. 2. 14 ADC采样时间寄存器 5(ADC_SMPT5)

ADC 采样时间寄存器 5 (ADC_SMPT5)

偏移地址:0x34

复位值:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
							SMPPLUS	SMP18<7:0>								SMP17<7:0>								SMP16<7:0>							

—	Bits 31-25	—	—
SMPPLUS	Bit 24	R/W	**额外增加采样时间开关** 原先既有的 SMPx 采样时间设定外，额外增加 1 个 ADC 时钟的开关设定。 0: 不额外增加采样时间，基准维持为 3.5 ADC 时钟 1: 增加采样时间，基准增加为 4.5 ADC 时钟 注: RSTART = 0 以及 START = 0 (确保没有正在进行中的转换)，才允许写入配置。
SMP18	Bit 23-16	R/W	**通道 18 的采样时间设定** T_{SMP} 的公式可以参照章节 14. 4. 5 采样时间 = T_{SMP} + 3.5 ADC 时钟周期。 注: RSTART = 0 以及 JSTART = 0 (确保没有正在进行中的转换)，才允许写入配置。
SMP17	Bit 15-8	R/W	**通道 17 的采样时间设定** T_{SMP} 的公式可以参照章节 14. 4. 5 采样时间 = T_{SMP} + 3.5 ADC 时钟周期。 注: RSTART = 0 以及 JSTART = 0 (确保没有正在进行中的转换)，才允许写入配置。
SMP16	Bit 7-0	R/W	**通道 16 的采样时间设定** T_{SMP} 的公式可以参照章节 14. 4. 5 采样时间 = T_{SMP} + 3.5 ADC 时钟周期。 注: RSTART = 0 以及 JSTART = 0 (确保没有正在进行中的转换)，才允许写入配置。

14. 5. 2. 15 ADC模拟看门狗 1 阀值寄存器(ADC_TR1)

ADC 模拟看门狗 1 阀值寄存器 (ADC_TR1)

偏移地址:0x38

复位值:0x0FFF 0000

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
				HT1<11:0>													AWDFILT<2:0>			LT1<11:0>											

—	Bits 31-28	—	—
HT1	Bit 27-16	R/W	**模拟看门狗 1 阀值上限** 设定 ADC 看门狗 1 阀值上限，数据高于此设定，会产生 AWD1 旗标。 注: RSTART = 0 以及 JSTART = 0(确保没有正在进行中的转换)，才允许写入配置。
—	Bits 15	—	—
AWDFILT	Bit 14-12	R/W	**模拟看门狗过滤参数** 000: 不用过滤 001: 侦测连续 2 次超出范围产生 AWD 旗标或中断 010: 侦测连续 3 次超出范围产生 AWD 旗标或中断 011: 侦测连续 4 次超出范围产生 AWD 旗标或中断 100: 侦测连续 5 次超出范围产生 AWD 旗标或中断 101: 侦测连续 6 次超出范围产生 AWD 旗标或中断 110: 侦测连续 7 次超出范围产生 AWD 旗标或中断 111: 侦测连续 8 次超出范围产生 AWD 旗标或中断 注: RSTART = 0 以及 JSTART = 0 (确保没有正在进行中的转换)，才允许写入配置。
LT1	Bit 11-0	R/W	**模拟看门狗 1 阀值下限** 设定 ADC 看门狗 1 阀值下限，数据低于此设定，会产生 AWD1 旗标。

			注: RSTART = 0 以及 JSTART = 0 (确保没有正在进行中的转换)，才允许写入配置。

14. 5. 2. 16 ADC模拟看门狗 2 阀值寄存器(ADC_TR2)

ADC 模拟看门狗 2 阀值寄存器 (ADC_TR2)

偏移地址:0x3C

复位值:0x00FF 0000

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
								HT2<7:0>																LT2<7:0>							

—	Bits 31-24	—	—
HT2	Bit 23-16	R/W	**模拟看门狗 2 阀值上限** 设定 ADC 看门狗 2 阀值上限，数据高于此设定，会产生 AWD2 旗标。 注: RSTART = 0 以及 JSTART = 0 (确保没有正在进行中的转换)，才允许写入配置。
—	Bits 15-8	—	—
LT2	Bit 7-0	R/W	**模拟看门狗 2 阀值下限** 设定 ADC 看门狗 2 阀值下限，数据低于此设定，会产生 AWD2 旗标。 注: RSTART = 0 以及 JSTART = 0 (确保没有正在进行中的转换)，才允许写入配置。

14. 5. 2. 17 ADC模拟看门狗 3 阀值寄存器(ADC_TR3)

ADC 模拟看门狗 3 阀值寄存器 (ADC_TR3)																													
偏移地址:0x40																													
复位值:0x00FF 0000																													
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
								HT3<7:0>																LT3<7:0>						

—	Bits 31-24	—	—
HT3	Bit 23-16	R/W	**模拟看门狗 3 阀值上限** 设定 ADC 看门狗 3 阀值上限，数据高于此设定，会产生 AWD3 旗标。 注: RSTART = 0 以及 JSTART = 0 (确保没有正在进行中的转换)，才允许写入配置。
—	Bits 15-8	—	—
LT3	Bit 7-0	R/W	**模拟看门狗 3 阀值下限** 设定 ADC 看门狗 3 阀值下限，数据低于此设定，会产生 AWD3 旗标。 注: RSTART = 0 以及 JSTART = 0 (确保没有正在进行中的转换)，才允许写入配置。

14. 5. 2. 18 ADC标准通道序列寄存器 1(ADC_SQR1)

ADC 标准通道序列寄存器 1 (ADC_SQR1)																													
偏移地址:0x44																													
复位值:0x0000 0000																													
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
			RSQ4<4:0>						RSQ3<4:0>						RSQ2<4:0>						RSQ1<4:0>						RSQLEN<3:0>		

—	Bits 31-29	—	—
RSQ4	Bit 28-24	R/W	**标准序列中 4th 转换通道** 设定数值 0-18，定义标准序列中，4th 转换的通道。 注: RSTART = 0 (确保没有正在进行中的标准转换)，才允许写入配置。

—	Bits 23	—	—
RSQ3	Bit 22-18	R/W	**标准序列中 3rd 转换通道** 设定数值 0-18，定义标准序列中，3rd 转换的通道。 注: RSTART = 0 (确保没有正在进行中的标准转换)，才允许写入配置。
—	Bits 17	—	—
RSQ2	Bit 16-12	R/W	**标准序列中 2nd 转换通道** 设定数值 0-18，定义标准序列中，2nd 转换的通道。 注: RSTART = 0 (确保没有正在进行中的标准转换)，才允许写入配置。
—	Bits 11	—	—
RSQ1	Bit 10-6	R/W	**标准序列中 1st 转换通道** 设定数值 0-18，定义标准序列中，1st 转换的通道。 注: RSTART = 0 (确保没有正在进行中的标准转换)，才允许写入配置。
—	Bits 5-4	—	—
RSQLEN	Bit 3-0	R/W	**标准序列长度设定** 定义标准通道转换序列中的转换次数 0000: 1 次转换 0001: 2 次转换 0010: 3 次转换 …1111: 16 次转换 注: RSTART = 0 (确保没有正在进行中的标准转换)，才允许写入配置。

14. 5. 2. 19 ADC标准通道序列寄存器 2(ADC_SQR2)

ADC 标准通道序列寄存器 2 (ADC_SQR2)																												
偏移地址:0x48																												
复位值:0x0000 0000																												
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
			RSQ9<4:0>						RSQ8<4:0>						RSQ7<4:0>						RSQ6<4:0>						RSQ5<4:0>		

—	Bits 31-29	—	—
RSQ9	Bit 28-24	R/W	**标准序列中 9th 转换通道** 设定数值 0-18，定义标准序列中，9th 转换的通道。 注: RSTART = 0 (确保没有正在进行中的标准转换)，才允许写入配置。
—	Bits 23	—	—
RSQ8	Bit 22-18	R/W	**标准序列中 8th 转换通道** 设定数值 0-18，定义标准序列中，8th 转换的通道。 注: RSTART = 0 (确保没有正在进行中的标准转换)，才允许写入配置。
—	Bits 17	—	—
RSQ7	Bit 16-12	R/W	**标准序列中 7th 转换通道** 设定数值 0-18，定义标准序列中，7th 转换的通道。 注: RSTART = 0 (确保没有正在进行中的标准转换)，才允许写入配置。
—	Bits 11	—	—
RSQ6	Bit 10-6	R/W	**标准序列中 6th 转换通道** 设定数值 0-18，定义标准序列中，6th 转换的通道。 注: RSTART = 0 (确保没有正在进行中的标准转换)，才允许写入配置。
—	Bits 5	—	—
RSQ5	Bit 4-0	R/W	**标准序列中 5th 转换通道** 设定数值 0-18，定义标准序列中，5th 转换的通道。

			注: RSTART = 0 (确保没有正在进行中的标准转换)，才允许写入配置。

14. 5. 2. 20 ADC标准通道序列寄存器 3 (ADC_SQR3)

ADC 标准通道序列寄存器 3 (ADC_SQR3)

偏移地址:0x4C

复位值:0x0000 0000

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
			RSQ14<4:0>						RSQ13<4:0>						RSQ12<4:0>						RSQ11<4:0>						RSQ10<4:0>				

—	Bits 31-29	—	—
RSQ14	Bit 28-24	R/W	**标准序列中 14th 转换通道** 设定数值 0-18，定义标准序列中，14th 转换的通道。 注: RSTART = 0 (确保没有正在进行中的标准转换)，才允许写入配置。
—	Bits 23	—	—
RSQ13	Bit 22-18	R/W	**标准序列中 13th 转换通道** 设定数值 0-18，定义标准序列中，13th 转换的通道。 注: RSTART = 0 (确保没有正在进行中的标准转换)，才允许写入配置。
—	Bits 17	—	—
RSQ12	Bit 16-12	R/W	**标准序列中 12th 转换通道** 设定数值 0-18，定义标准序列中，12th 转换的通道。 注: RSTART = 0 (确保没有正在进行中的标准转换)，才允许写入配置。
—	Bits 11	—	—
RSQ11	Bit 10-6	R/W	**标准序列中 11th 转换通道** 设定数值 0-18，定义标准序列中，11th 转换的通道。 注: RSTART = 0 (确保没有正在进行中的标准转换)，才允许写入配置。
—	Bits 5	—	—

RSQ10	Bit 4-0	R/W	**标准序列中 10th 转换通道** 设定数值 0-18，定义标准序列中，10th 转换的通道。 注: RSTART = 0 (确保没有正在进行中的标准转换)，才允许写入配置。

14. 5. 2. 21 ADC标准通道序列寄存器 4(ADC_SQR4)

ADC 标准通道序列寄存器 4 (ADC_SQR4)

偏移地址:0x50

复位值:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																					RSQ16<4:0>						RSQ15<4:0>				

—	Bits 31-11	—	—
RSQ16	Bit 10-6	R/W	**标准序列中 16th 转换通道** 设定数值 0-18，定义标准序列中，16th 转换的通道。 注: RSTART = 0 (确保没有正在进行中的标准转换)，才允许写入配置。
—	Bits 5	—	—
RSQ15	Bit 4-0	R/W	**标准序列中 15th 转换通道** 设定数值 0-18，定义标准序列中，15th 转换的通道。 注: RSTART = 0 (确保没有正在进行中的标准转换)，才允许写入配置。

14. 5. 2. 22 ADC标准数据寄存器(ADC_DR)

ADC 标准数据寄存器 (ADC_DR)																												
偏移地址:0x54																												
复位值:0x0000 0000																												
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
—																RDATA<15:0>															

—	Bits 31-16	—	—
RDATA	Bit 15-0	R	**标准转换数据** 寄存器只支持读取。最新的标准通道的转换结果，数据可分成左对齐右对齐。

14. 5. 2. 23 ADC插入通道序列寄存器(ADC_JSQR)

ADC 插入通道序列寄存器 (ADC_JSQR)																												
偏移地址:0x5C																												
复位值:0x0000 0000																												
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
JSQ4<4:0>					—	JSQ3<4:0>					—	JSQ2<4:0>					—	JSQ1<4:0>					JEXTEN<1:0>		JEXTSEL<4:0>					JSQLEN<1:0>	

JSQ4	Bit 31-27	R/W	**插入序列中 4th 转换通道** 设定数值 0-18，定义插入序列中，4th 转换的通道。 注: JSTART = 0 (确保没有正在进行中的插入转换)，才允许写入配置。
—	Bits 26	—	—
JSQ3	Bit 25-21	R/W	**插入序列中 3rd 转换通道** 设定数值 0-18，定义插入序列中，3rd 转换的通道。 注: JSTART = 0 (确保没有正在进行中的插入转换)，才允许写入配置。
—	Bits 20	—	—
JSQ2	Bit 19-15	R/W	**插入序列中 2nd 转换通道**

			设定数值 0-18，定义插入序列中，2nd 转换的通道。 注: JSTART = 0 (确保没有正在进行中的插入转换)，才允许写入配置。
—	Bits 14	—	—
JSQ1	Bit 13-9	R/W	**插入序列中 1st 转换通道** 设定数值 0-18，定义插入序列中，1st 转换的通道。 注: JSTART = 0 (确保没有正在进行中的插入转换)，才允许写入配置。
JEXTEN	Bit 8-7	R/W	**插入外部触发开启以及极性选择** 插入的外部触发选择，开启硬件以及极性选择。 00: 如果 JQEN = 1，硬件以及软件侦测触发关闭 00: 如果 JQEN = 0，硬件侦测触发关闭 (允许软件触发启动转换) 01: 侦测事件上升沿，产生硬件触发 10: 侦测事件下降沿，产生硬件触发 11: 侦测事件上升以及下降沿，产生硬件触发 注: JSTART = 0 (确保没有正在进行中的插入转换)，才允许写入配置。 注: 当 JQEN = 1 以及队列已空的情况下，硬件以及软件的触发侦测皆关闭。
JEXTSEL	Bit 6-2	R/W	**插入外部触发事件选择** 选择启动插入转换的外部触发事件。表 14-5 00000: 事件 0 00001: 事件 1 00010: 事件 2 00011: 事件 3 00100: 事件 4 00101: 事件 5 … 11111: 事件 31 注: JSTART = 0 (确保没有正在进行中的插入转换)，才允许写入配置。
JSQLEN	Bit 1-0	R/W	**插入序列长度设定** 定义插入通道转换序列中的转换次数 00: 1 次转换

			01: 2 次转换 10: 3 次转换 11: 4 次转换 注: JSTART = 0 (确保没有正在进行中的插入转换)，才允许写入配置。

14. 5. 2. 24 ADC偏移寄存器 1 (ADC_OFST1)

ADC 偏移寄存器 1 (ADC_OFST1)

偏移地址:0x60

复位值:0x0000 0000

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
OFFSETEN	OFFSETCH<4:0>					SATEN	POSEN	—	—	—	—	—	—	—	—	—	—	—	—	OFFSET<11:0>											

OFFSETEN	Bit 31	R/W	**数据偏移功能开关** 0: 数据偏移关闭 1: 数据偏移开启 注: RSTART = 0 以及 JSTART = 0 (确保没有正在进行中的转换)，才允许写入配置。
OFFSETCH	Bit 30-26	R/W	**数据偏移的通道选择** 设定的通道，该通道转换后的数据，会针对 OFFSET 的设定，进行偏移量的运算 注: RSTART = 0 以及 JSTART = 0 (确保没有正在进行中的转换)，才允许写入配置。
SATEN	Bit 25	R/W	**数据饱和开关** 0: 饱和关闭，偏移过后的数据可以为有符号数 1: 饱和开启，偏移过后的数据为无符号数，数据范围会介于 0x000 与 0xFFF 之间 注: RSTART = 0 以及 JSTART = 0 (确保没有正在进行中的转换)，才允许写入配置。
POSEN	Bit 24	R/W	**正数偏移量设置** 0: 负数偏移量 1: 正数偏移量 注: RSTART = 0 以及 JSTART = 0 (确保没有正在进行中的转换)，才允许写入配置。
—	Bits 23-12	—	—

OFFSET	Bit 11-0	R/W	**对于 OFFSETCH 设定的通道数据偏移量** 设定数据偏移量，针对于 OFFSETCH 通道的原始转换后需要扣除的数值。转换过后的数据，会存放于 ADC_DR 以及 ADC_JDR 寄存器。 注: RSTART = 0 以及 JSTART = 0 (确保没有正在进行中的转换)，才允许写入配置。 注: 当 ADC_OFSTy 有多个设定到相同通道，偏移量会以 y 最小的为主。

14. 5. 2. 25 ADC偏移寄存器 2(ADC_OFST2)

ADC 偏移寄存器 2 (ADC_OFST2)

偏移地址:0x64

复位值:0x0000 0000

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
OFFSETEN	OFFSETCH<4:0>					SATEN	POSEN	-	-	-	-	-	-	-	-	-	-	-	-	OFFSET<11:0>											

OFFSETEN	Bit 31	R/W	**数据偏移功能开关** 0: 数据偏移关闭 1: 数据偏移开启 注: RSTART = 0 以及 JSTART = 0 (确保没有正在进行中的转换)，才允许写入配置。
OFFSETCH	Bit 30-26	R/W	**数据偏移的通道选择** 设定的通道，该通道转换后的数据，会针对 OFFSET 的设定，进行偏移量的运算 注: RSTART = 0 以及 JSTART = 0 (确保没有正在进行中的转换)，才允许写入配置。
SATEN	Bit 25	R/W	**数据饱和开关** 0: 饱和关闭，偏移过后的数据可以为有符号数 1: 饱和开启，偏移过后的数据为无符号数，数据范围会介于 0x000 与 0xFFF 之间 注: RSTART = 0 以及 JSTART = 0 (确保没有正在进行中的转换)，才允许写入配置。
POSEN	Bit 24	R/W	**正数偏移量设置** 0: 负数偏移量 1: 正数偏移量

			注: RSTART = 0 以及 JSTART = 0 (确保没有正在进行中的转换)，才允许写入配置。
—	Bits 23-12	—	—
OFFSET	Bit 11-0	R/W	**对于 OFFSETCH 设定的通道数据偏移量** 设定数据偏移量，针对于 OFFSETCH 通道的原始转换后需要扣除的数值。转换过后的数据，会存放于 ADC_DR 以及 ADC_JDR 寄存器。 注: RSTART = 0 以及 JSTART = 0 (确保没有正在进行中的转换)，才允许写入配置。 注: 当 ADC_OFSTy 有多个设定到相同通道，偏移量会以 y 最小的为主。

14. 5. 2. 26 ADC偏移寄存器 3(ADC_OFST3)

ADC 偏移寄存器 3 (ADC_OFST3)

偏移地址:0x68

复位值:0x0000 0000

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
OFFSETEN	OFFSETCH<4:0>					SATEN	POSEN													OFFSET<11:0>											

OFFSETEN	Bit 31	R/W	**数据偏移功能开关** 0: 数据偏移关闭 1: 数据偏移开启 注: RSTART = 0 以及 JSTART = 0 (确保没有正在进行中的转换)，才允许写入配置。
OFFSETCH	Bit 30-26	R/W	**数据偏移的通道选择** 设定的通道，该通道转换后的数据，会针对 OFFSET 的设定，进行偏移量的运算 注: RSTART = 0 以及 JSTART = 0 (确保没有正在进行中的转换)，才允许写入配置。
SATEN	Bit 25	R/W	**数据饱和开关** 0: 饱和关闭，偏移过后的数据可以为有符号数 1: 饱和开启，偏移过后的数据为无符号数，数据范围会介于 0x000 与 0xFFF 之间 注: RSTART = 0 以及 JSTART = 0 (确保没有正在进行中的转换)，才允许写入配置。

POSEN	Bit 24	R/W	**正数偏移量设置** 0: 负数偏移量 1: 正数偏移量 注: RSTART = 0 以及 JSTART = 0 (确保没有正在进行中的转换)，才允许写入配置。
—	Bits 23-12	—	—
OFFSET	Bit 11-0	R/W	**对于 OFFSETCH 设定的通道数据偏移量** 设定数据偏移量，针对于 OFFSETCH 通道的原始转换后需要扣除的数值。转换过后的数据，会存放于 ADC_DR 以及 ADC_JDR 寄存器。 注: RSTART = 0 以及 JSTART = 0 (确保没有正在进行中的转换)，才允许写入配置。 注: 当 ADC_OFSTy 有多个设定到相同通道，偏移量会以 y 最小的为主。

14. 5. 2. 27 ADC偏移寄存器 4(ADC_OFST4)

ADC 偏移寄存器 4 (ADC_OFST4)

偏移地址:0x6C

复位值:0x0000 0000

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
OFFSETEN	OFFSETCH<4:0>					SATEN	POSEN													OFFSET<11:0>											

OFFSETEN	Bit 31	R/W	**数据偏移功能开关** 0: 数据偏移关闭 1: 数据偏移开启 注: RSTART = 0 以及 JSTART = 0 (确保没有正在进行中的转换)，才允许写入配置。
OFFSETCH	Bit 30-26	R/W	**数据偏移的通道选择** 设定的通道，该通道转换后的数据，会针对 OFFSET 的设定，进行偏移量的运算 注: RSTART = 0 以及 JSTART = 0 (确保没有正在进行中的转换)，才允许写入配置。
SATEN	Bit 25	R/W	**数据饱和开关** 0: 饱和关闭，偏移过后的数据可以为有符号数 1: 饱和开启，偏移过后的数据为无符号数，数

			据范围会介于 0x000 与 0xFFF 之间 注: RSTART = 0 以及 JSTART = 0 (确保没有正在进行中的转换)，才允许写入配置。
POSEN	Bit 24	R/W	**正数偏移量设置** 0: 负数偏移量 1: 正数偏移量 注: RSTART = 0 以及 JSTART = 0 (确保没有正在进行中的转换)，才允许写入配置。
—	Bits 23-12	—	—
OFFSET	Bit 11-0	R/W	**对于 OFFSETCH 设定的通道数据偏移量** 设定数据偏移量，针对于 OFFSETCH 通道的原始转换后需要扣除的数值。转换过后的数据，会存放于 ADC_DR 以及 ADC_JDR 寄存器。 注: RSTART = 0 以及 JSTART = 0 (确保没有正在进行中的转换)，才允许写入配置。 注: 当 ADC_OFSTy 有多个设定到相同通道，偏移量会以 y 最小的为主。

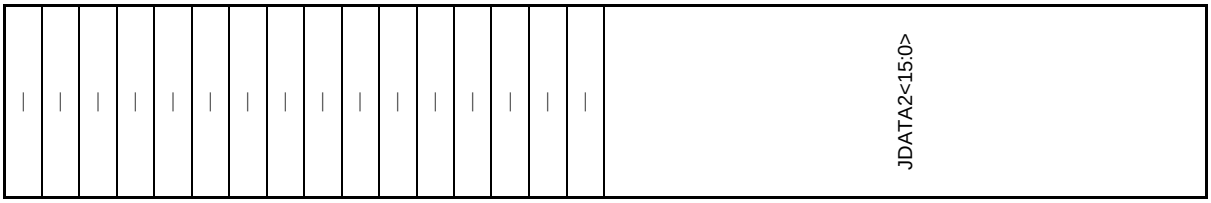
14. 5. 2. 28 ADC插入数据寄存器 1 (ADC_JDR1)

ADC 插入数据寄存器 1(ADC_ JDR1)																												
偏移地址:0x70																												
复位值:0x0000 0000																												
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
																JDATA1<15:0>															

—	Bits 31-16	—	—
JDATA1	Bit 15-0	R	**插入转换数据 1** 寄存器只支持读取。插入通道顺序 1 的转换结果，数据可分成左对齐右对齐。

14. 5. 2. 29 ADC插入数据寄存器 2 (ADC_JDR2)

ADC 插入数据寄存器 2 (ADC_JDR2)																												
偏移地址:0x74																												
复位值:0x0000 0000																												
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



—	Bits 31-16	—	—
JDATA2	Bit 15-0	R	**插入转换数据 2** 寄存器只支持读取。插入通道顺序 2 的转换结果，数据可分成左对齐右对齐。

14. 5. 2. 30 ADC插入数据寄存器 3 (ADC_JDR3)

ADC 插入数据寄存器 3 (ADC_JDR3)

偏移地址:0x78

复位值:0x0000 0000

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
																JDATA3<15:0>															

—	Bits 31-16	—	—
JDATA3	Bit 15-0	R	**插入转换数据 3** 寄存器只支持读取。插入通道顺序 3 的转换结果，数据可分成左对齐右对齐。

14. 5. 2. 31 ADC插入数据寄存器 4 (ADC_JDR4)

ADC 插入数据寄存器 4 (ADC_JDR4)																												
偏移地址:0x7C																												
复位值:0x0000 0000																												
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																JDATA4<15:0>															

—	Bits 31-16	—	—
JDATA4	Bit 15-0	R	**插入转换数据 4** 寄存器只支持读取。插入通道顺序 4 的转换结果，数据可分成左对齐右对齐。

14. 5. 2. 32 ADC模拟看门狗 2 配置寄存器(ADC_AWD2CR)

ADC 模拟看门狗 2 配置寄存器 (ADC_AWD2CR)																												
偏移地址:0x80																												
复位值:0x0000 0000																												
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
													AWD2CH<18:0>																	

—	Bits 31-19	—	—
AWD2CH	Bit 18-0	R/W	**模拟看门狗 2 通道选择开关** 设定模拟看门狗 2 的监控通道。 AWD2CH[i] = 0：模拟输入通道不受模拟看门狗 2 监控 AWD2CH[i] = 1：模拟输入通道受到模拟看门狗 2 监控 当 AWD2CH[18:0] = 000..0，模拟看门狗 2 被关闭。 注: RSTART = 0 以及 JSTART = 0 (确保没有正在进行中的转换)，才允许写入配置。 注: 通道的选择，也需要为 SQR 或 JSQR 的设置通道。

14. 5. 2. 33 ADC模拟看门狗 3 配置寄存器(ADC_AWD3CR)

ADC 模拟看门狗 3 配置寄存器 (ADC_AWD3CR)																												
偏移地址:0x84																												
复位值:0x0000 0000																												
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
													AWD3CH<18:0>																		

—	Bits 31-19	—	—
AWD3CH	Bit 18-0	R/W	**模拟看门狗 3 通道选择开关** 设定模拟看门狗 3 的监控通道。 AWD3CH[i] = 0：模拟输入通道不受模拟看门狗 3 监控 AWD3CH[i] = 1：模拟输入通道受到模拟看门狗 3 监控 当 AWD3CH[18:0] = 000..0，模拟看门狗 3 关闭。 注: RSTART = 0 以及 JSTART = 0 (确保没有正在进行中的转换)，才允许写入配置。 注: 通道的选择，也需要为 SQR 或 JSQR 的设置通道。

14. 5. 2. 34 ADC增益系数寄存器(ADC_GCOMP)

ADC 增益系数 寄存器(ADC_GCOMP)																												
偏移地址:0x88																												
复位值:0x0000 0000																												
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
																		COEF<13:0>													

—	Bits 31-16	—	—
COEF	Bit 13-0	R/W	**增益补偿系数** 00 1000 0000 0000 增益因子 0.5 01 0000 0000 0000 增益因子 1 10 0000 0000 0000 增益因子 2

			11 0000 0000 0000 增益因子 3 … 增益系数是将 0 至 3.999756 细分成 4096 阶。 注：增益补偿的应用，必须开启 ADC_CFG2.GCOMP = 1。

14. 5. 2. 35 ADC 通用控制寄存器(ADC_CCR)

AD1 以及 ADC2 共享的接口控制。

ADC 通用控制寄存器(ADC_CCR)

偏移地址:0x204

复位值:0x0000 0000

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
												PRESCALE<3:0>				DMAMODE<1:0>				DELAYSEL<3:0>							DUALMODE<4:0>				

—	Bits 31-20	—	—
PRESCALE	Bit 19-16	R/W	**ADC 时钟分频配置** 定义 ADC 时钟分频细数，该配置适用于 ADC1 以及 ADC2。 0000: 输入 ADC 时钟没有分频 0001: 输入 ADC 时钟 2 分频 0010: 输入 ADC 时钟 4 分频 0011: 输入 ADC 时钟 6 分频 0100: 输入 ADC 时钟 8 分频 0101: 输入 ADC 时钟 10 分频 0110: 输入 ADC 时钟 12 分频 0111: 输入 ADC 时钟 16 分频 1000: 输入 ADC 时钟 32 分频 1001: 输入 ADC 时钟 64 分频 1010: 输入 ADC 时钟 128 分频 1011: 输入 ADC 时钟 256 分频 Other: 保留 注: 当 ADC 禁止，JSTART = 0、RSTART = 0、RSTP = 0、JSTP = 0、ADCDIS = 0 以及 ADCEN = 0 的时候，才允许设置 PRESCALE[3:0]。

DMAMODE	Bit 15-14	R/W	**双重 ADC 的 DMA 模式** 00: 正常双重 ADC 模式，DMA 功能关闭 01: 保留 10: DMA 功能开启，用于 12 以及 10 位分辨率 11: DMA 功能开启，用于 8 以及 6 位分辨率 注: RSTART = 0 (确保没有正在进行中的插入转换)，才允许写入配置。
—	Bits 13-12	—	—
DELAYSEL	Bit 11-8	R/W	**2 个采样之间的延迟寄存器** 设置双重 ADC 交换模式，两个 ADC 采样之间的延迟。 注: 当 ADC 禁止，JSTART = 0、RSTART = 0、RSTP = 0、JSTP = 0、ADCDIS = 0 以及 ADCEN = 0 的时候，才允许设置 DEALYSEL[3:0]。
—	Bits 7-5	—	—
DUALMODE	Bit 4-0	R/W	**双重 ADC 模式选择** ADC1 以及 ADC2 独立控制 00000 : 独立模式 00001 至 01001: 双重模式，ADC1 以及 ADC2 一起运行。 00001: 标准同步 + 插入同步的组合模式 00010: 标准同步 + 交互触发的组合模式 00011: 交换模式 + 插入同步的组合磨石 00100: 保留 00101: 插入同步模式 00110: 标准同步模式 00111: 交换模式 01001: 交互触发模式 Other: 保留，且不允许写入 注: 当 ADC 禁止，JSTART = 0、RSTART = 0、RSTP = 0、JSTP = 0、ADCDIS = 0 以及 ADCEN = 0 的时候，才允许设置 DUALMODE[4:0]。

14. 5. 2. 36 ADC 通道数据寄存器(ADC_CDR)

ADC 通道数据寄存器 (ADC_CDR)																													
偏移地址:0x208																													
复位值:0x0000 0000																													
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
ADC2_RDATA <15:0>																ADC1_RDATA <15:0>															

ADC2_RDATA	Bit 31-16	R	**ADC2 标准转换数据** 寄存器只支持读取。双重 ADC 模式，ADC2 最新的标准通道的转换结果，数据可分成左对齐右对齐。
ADC1_RDATA	Bit 15-0	R	**ADC1 标准转换数据** 寄存器只支持读取。双重 ADC 最新的标准通道的转换结果，数据可分成左对齐右对齐。

第15章 运算放大器 (OPAMP)

15. 1 概述

芯片嵌入四个拥有两个输入一个输出的运算放大器。这三个 I/O 可连接到外部引脚，并能使用任何类型的外部组件互连。运算放大器可以在内部配置为跟随器，或者配置为非反相放大器其增益为 2 ~ 64 倍，又或者配置为反相放大器其增益为 -1 ~ -63 倍。

15. 2 特性

- 较低的输入偏置电流。
- 较低的输入偏移电压。
- 高频带宽增益。
- 提供高速模式以实现更好的转换率。

15. 3 结构图

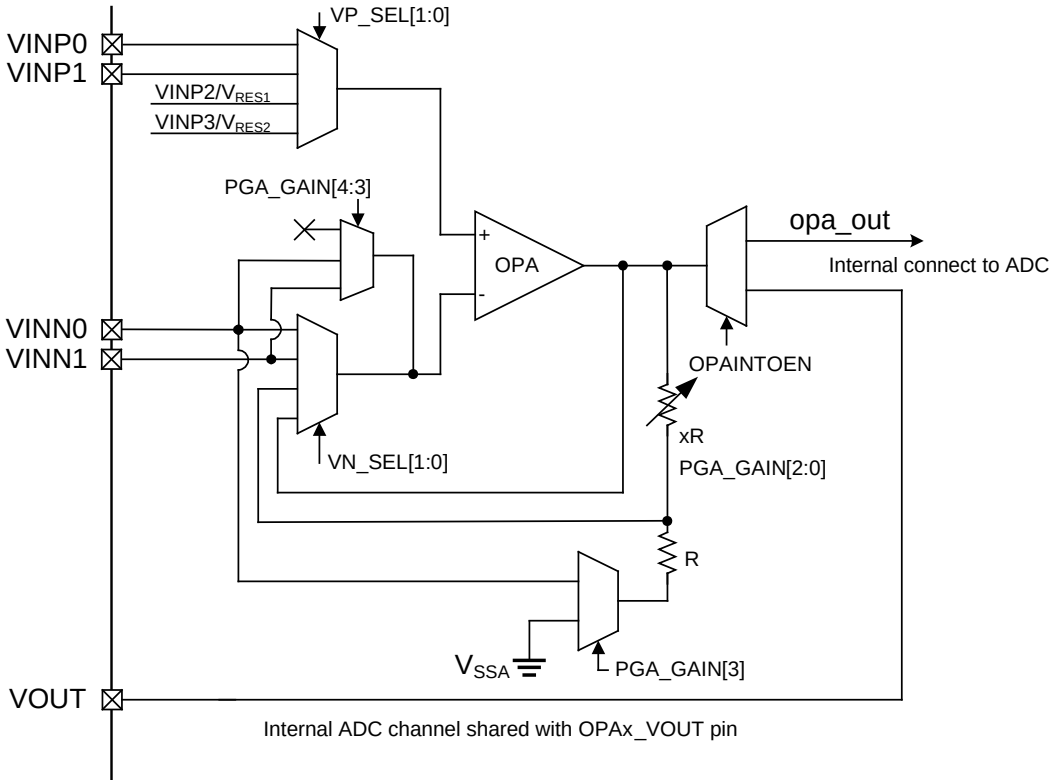


图 15-1　OPAMP 结构图

15. 4 功能描述

15. 4. 1 OPAMP功能描述

OPAMP 拥有几种工作模式。每个 OPAMP 都可以单独开启，关闭时输出为高阻抗。当开启时，他可以处于校准模式或功能模式。当在功能模式下，OPAMP 的输入与输出将按照 OPAMP 信号连接描述。

15. 4. 2 OPAMP输出至内部ADC通道

OPAMP 可以通过 **OPAx_CSR**.OPAINTOEN 位重新将输出定向至内部 ADC 通道。在这个情况下，原先在 GPIO 引脚上的 VOUT 会被释放，并可用做其他用途。ADC 可以在芯片内部量测 OPAMP 的 VOUT 输出，请参考 OPAMP 信号连接描述来确认介于 OPAMPx 的内部输出与 ADC 通道的联机。

15. 4. 3 OPAMP复位与时钟

OPAMP 输出是使用 PCLK (APB 时钟)进行同步后的结果。OPAEN 位控制开启与关闭 OPAMP 操作。OPAMP 寄存器配置需在开启 OPAEN 前配置完成，以避免对输出结果造成影响。当不需要 OPAMP 输出时，可以将 OPAMP 关闭以节省电源。在 OPAMP 关闭时，先前设置的所有配置将会保持不变(包括校准值)。

15. 4. 4 OPAMP初始化配置

在默认配置下，运算放大器处在功能模式，拥有三个输入/输出并连致外部引脚。在默认模式下，运算放大器使用预设校准值用余其偏移量。校准值是可以被调整的，请参考 OPAMP 校准方式。在默认配置下使用正常模式，其提供标准性能的转换率。可以设置 OPAHSM 位，将运算放大器切换到高速模式以便提供更好的转换率。数据表中定义了标准和高速模式特性。

当 **OPA_CSRx** 寄存器内的 OPAEN 被设置，运算放大器会进入功能模式。此时两个输入与输出引脚会连接至定义的位置，请参考 OPAMP 信号连接描述。

注: OPAMP 输入与输出引脚必须在 GPIOx_MOD 寄存器内配置为模拟模式(默认状态)。

15. 4. 5 OPAMP信号连接描述

对于运算放大器的输入与输出信号联机通过 OPA_CSRx 与 OPA_TCMRx 寄存器决定。以下表格描述了 4 个运算放大器的联机：

信号	OPAMP1	OPAMP2	OPAMP3	OPAMP4
VINP0	PA08(OPA1_INP(0))	PA10(OPA2_INP(0))	PA12(OPA3_INP(0))	PA14(OPA4_INP(0))
VINP1	PA00(OPA1_INP(1))	PA01(OPA2_INP(1))	PA02(OPA3_INP(1))	PA03(OPA4_INP(1))
VINP2	V_{RES1}	V_{RES1}	V_{RES1}	V_{RES1}
VINP3	V_{RES2}	V_{RES2}	V_{RES2}	V_{RES2}
VINN0	PA09(OPA1_INN(0))	PA11(OPA2_INN(0))	PA13(OPA3_INN(0))	PA15(OPA4_INN(0))
VINN1	PB11(OPA1_INN(1))	PB10(OPA2_INN(1))	PC00(OPA3_INN(1))	PC01(OPA4_INN(1))
VOUT	PA00(OPA1_VOUT)	PA01(OPA2_VOUT)	PA02(OPA3_VOUT)	PA03(OPA4_VOUT)
opa_out	ADC1/ADC2 CH12	ADC1/ADC2 CH13	ADC1/ADC2 CH14	ADC1/ADC2 CH15

15. 4. 6 OPAMP 模式

运算放大器的输入和输出都可以连接至外部引脚。运算放大器可以使用在多种不同的配置情况下。

- 独立模式 (外部增益设置模式)
- 跟随配置模式
- PGA 模式

注: 输入信号的阻抗必须保持在一定水平以下，以避免输入产生漏电造成信号失真(由于来源电阻下降)。有关详细信息，请参阅数据手册中的电气特性部分。

15. 4. 6. 1 独立模式 (外部增益设置模式)

下面将介绍独立模式下 OPAMP 的使用过程。

在初始状态 **OPA_CSRx** 寄存器使用默认值，并且 GPIOx_MOD 也在预设状态，当 OPAEN 设置后，OPAMP 两个输入与输出将会连接至外部引脚。此默认配置使用默认校准值，并且操作在标准模式(最高性能)。

OPAMP 行为模式可以被以下条件改变：

- OPAHSM 位 - 配置后可以让运算放大器进入高速模式以提高转换速率。
- USERTRIM 位 - 配置后可以调整输入偏移的校准值。

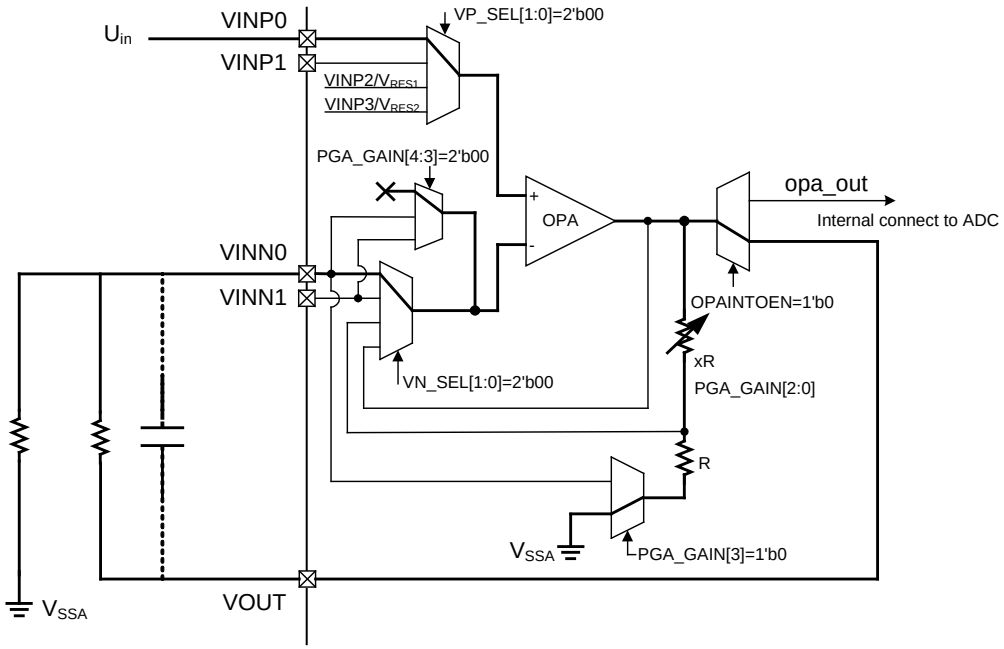


图 15-2　独立模式：外部增益设置模式(正相闭回路放大)

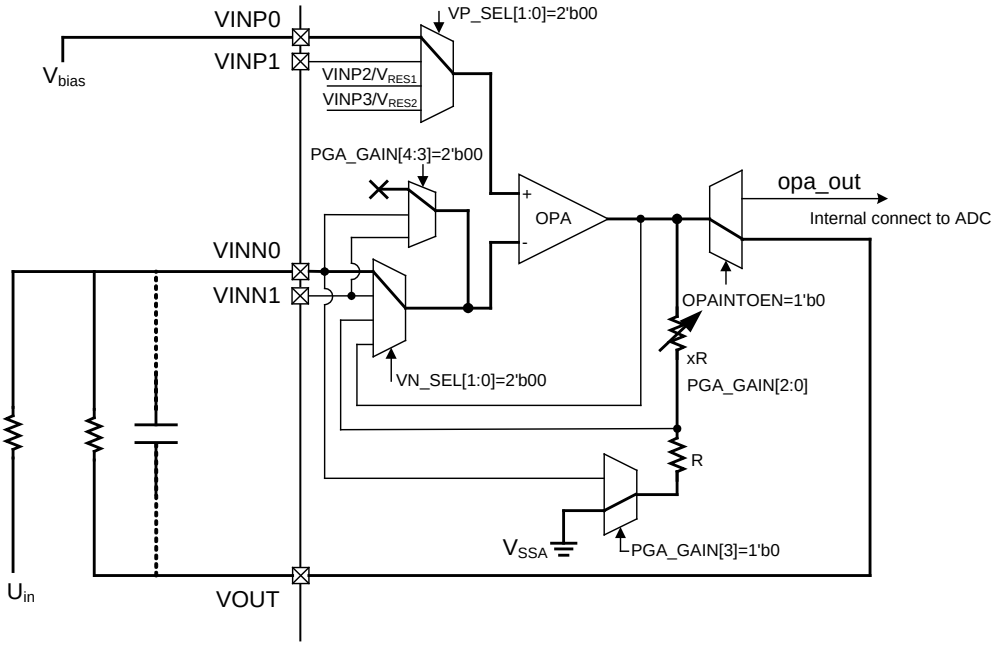


图 15-3　独立模式：外部增益设置模式(反相闭回路放大)

注: VOUT 输出可以通过配置 OPAINTOEN 位重新指向至内部 ADC 通道。在上述例子中，被配置为 VOUT 的输出 I/O 将被释放，并可做于其他用途。

注:Vbias voltage(Vbias)说明:反相闭回路放大 -反向放大器在输入 AC coupling 讯号下 OPAMP 的 VOUT 上的直流偏置由 VINP 输入的电压所决定。如果一个应用需要 VOUT 信号的电压居中于 VDDA/2 周围，那么 VINP 输入电压必须被偏置到相同的电压水平。

15. 4. 6. 2 跟随配置模式

下面将介绍跟随配置模式下 OPAMP 的使用过程

- 配置 **OPA_CSRx**.VN_SEL= 11，OPAMP 输出连接至负端输入。
- 配置 **OPA_CSRx**.VP_SEL= 00，GPIO 连接至正端输入。
- 当 OPAEN 被设置，此时正端输入上的电压会被缓存至 VOUT。

注: OPAMP 负端输入 I/O 将被释放，可做于其他用途。OPAMP 的输出可做为 ADC 的输入使用。因此，假设输入信号频率与运算放大器增益带宽规格兼容，配置为跟随配置模式的运算放大器可用在将输入信号进行阻抗匹配后再传送至 ADC 输入端。

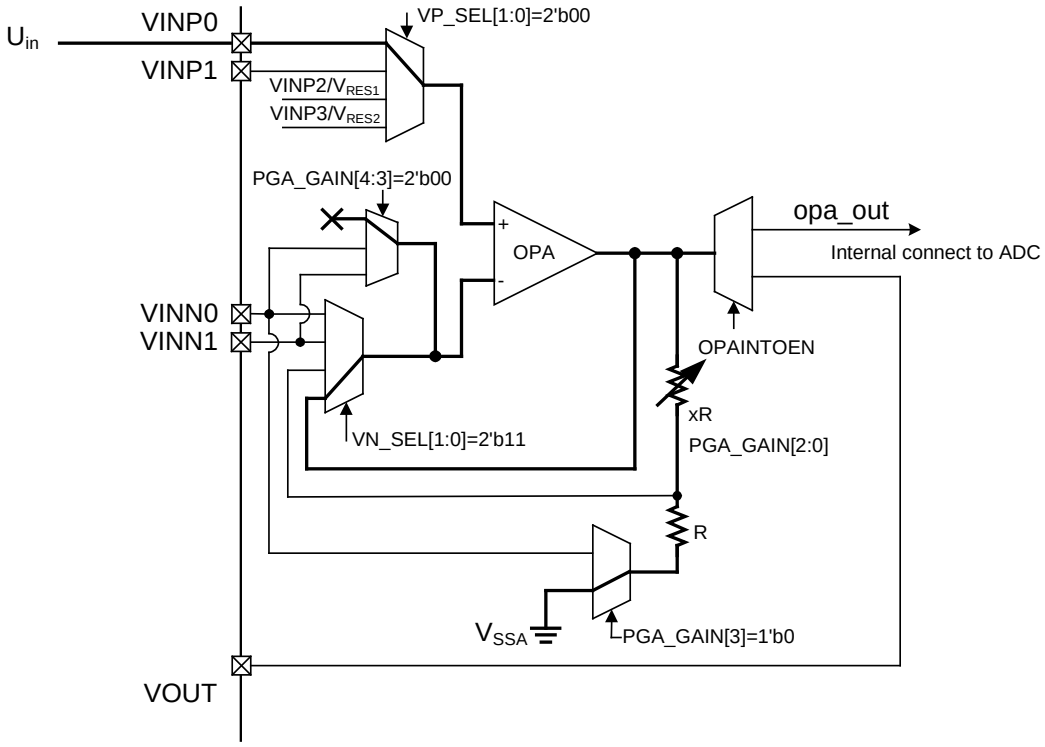


图 15-4 跟随配置模式

15. 4. 6. 3 内部增益模式:正相闭回路放大

下面将介绍此模式下 OPAMP 的配置流程。

- 配置 **OPA_CSRx**.VN_SEL= 10，OPAMP 输出通过电阻反馈电路连接至负端输入。
- 配置 **OPA_CSRx**.PGA_GAIN= 00000 ~ 00101，选择内部增益 2、4、8、16、32 或 64。
- 配置 **OPA_CSRx**.VP_SEL= 00，GPIO 连接至正端输入。
- 当 OPAEN 被设置，此时正端输入上的电压会被选定的增益放大并从 VOUT 输出。

注: 为了避免输出饱和，输入电压必须低于 VDDA 除以所选的增益。

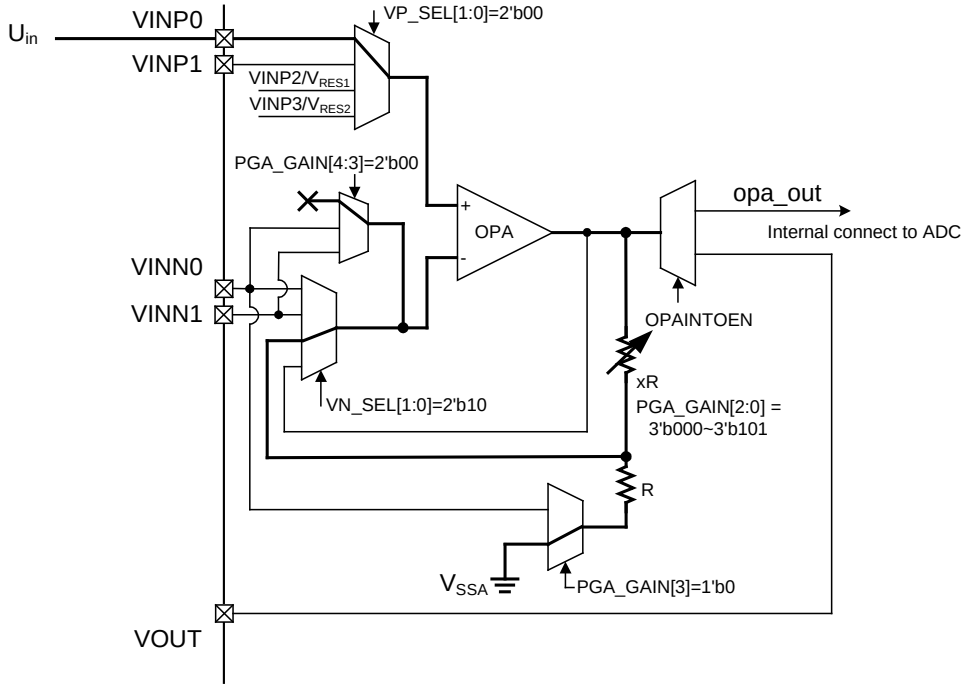


图 15-5 内部增益设置模式(正相闭回路放大)

15. 4. 6. 4 内部增益模式:正相闭回路放大搭配外部滤波电路

下面将介绍此模式下 OPAMP 的配置流程。

- 配置 **OPA_CSRx**.VN_SEL= 10，OPAMP 输出通过电阻反馈电路连接至负端输入。
- 配置 **OPA_CSRx**.PGA_GAIN= 10000 ~ 10101，选择内部增益 2、4、8、16、32 或 64，并将 VINN0 连接外部滤波电路。
- 配置 **OPA_CSRx**.VP_SEL= 00，将输入信号 U_{in} 连接至 VINP 对应的 GPIO。

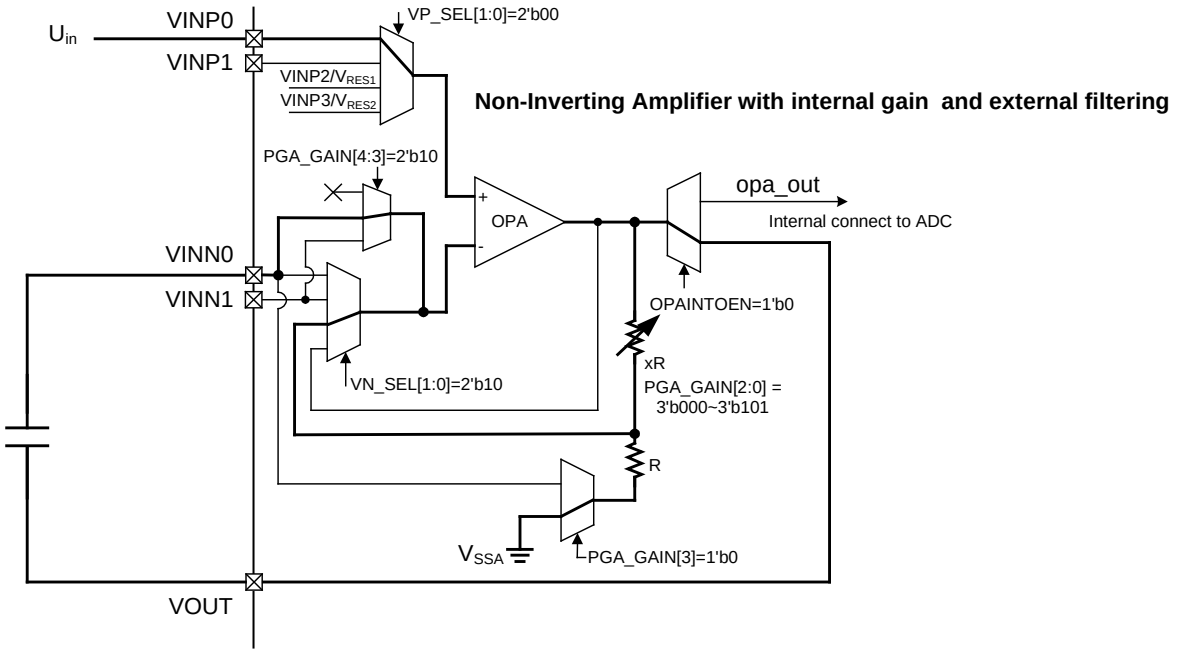


图 15-6 内部增益设置模式(正相闭回路放大搭配外部滤波电路)

15. 4. 6. 5 内部增益模式:正相闭回路放大和偏压

下面将介绍此模式下 OPAMP 的配置流程。

- 配置 **OPA_CSRx**.VN_SEL 寄存器= 10，OPAMP 输出通过电阻反馈电路连接至负端输入 VINN0。
- 配置 **OPA_CSRx**.PGA_GAIN 寄存器= 01000 ~ 01101，选择增益 2、4、8、16、32、64。
- 配置 **OPA_CSRx**.VP_SEL 寄存器 = 00，将输入信号 U_{in} 连接至 VINP 对应的 GPIO 并将偏压输入连接至 VINN 对应的 GPO。

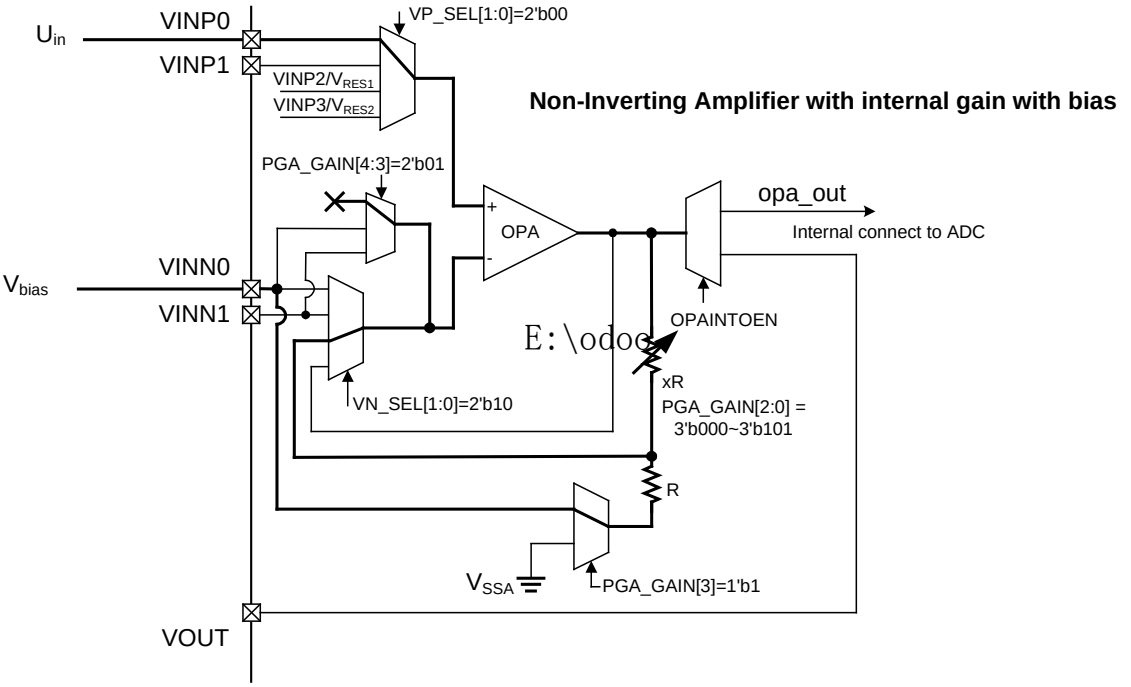


图 15-7 内部增益模式(正相闭回路放大和偏压)

15. 4. 6. 6　内部增益模式:反相閉迴路放大

下面将介绍此模式下 OPAMP 的配置流程。

- 配置 **OPA_CSRx**.VN_SEL= 10，将输入信号 U_{in} 连接至 VINN 对应的 GPIO。
- 配置 **OPA_CSRx**.PGA_GAIN= 01000 ~ 01101，选择反相增益-1、-3、-7、-15、-31、-63。
- 配置 **OPA_CSRx**.VP_SEL= 00，将 VINP 对应的 GPIO 接到 V_{SSA}。

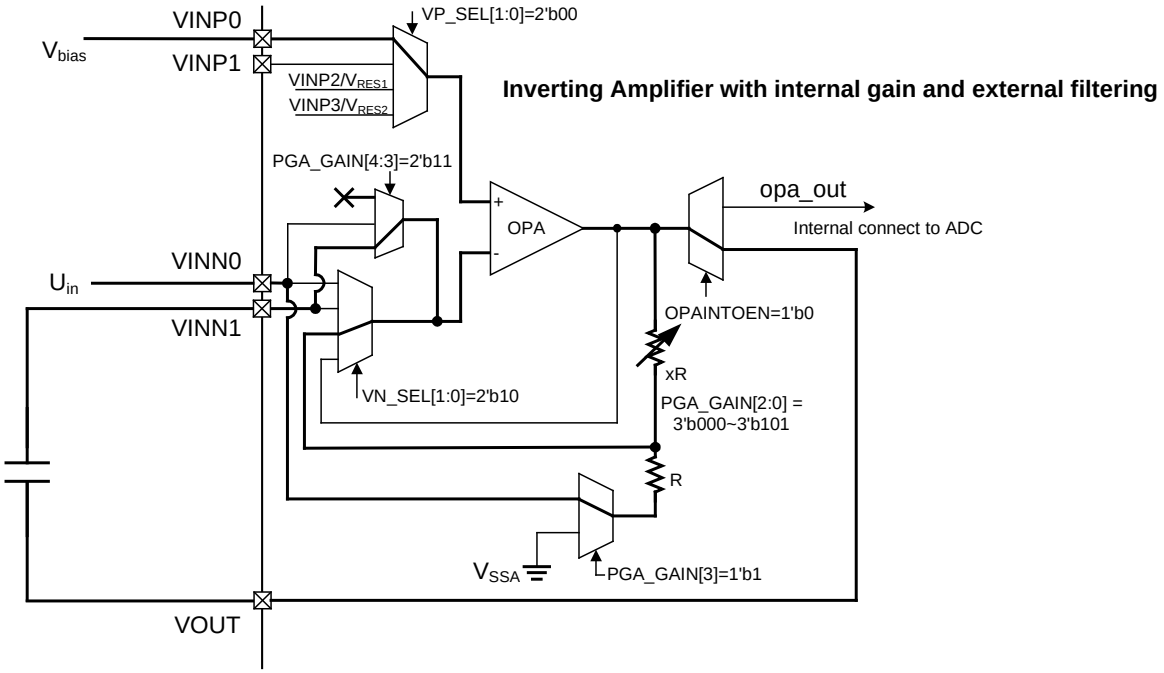


图 15-8　内部增益模式(反相闭回路放大)

15. 4. 6. 7　内部增益模式:正相闭回路放大和偏压，并搭配外部滤波电路

下面将介绍此模式下 OPAMP 的配置流程。

- 配置 **OPA_CSRx**.VN_SEL 寄存器= 10，OPAMP 输出通过电阻反馈电路连接至负端输入 VINN0。
- 配置 **OPA_CSRx**.PGA_GAIN 寄存器= 11000 ~ 11101，选择增益 2、4、8、16、32、64。
- 配置 **OPA_CSRx.**VP_SEL 寄存器= 00，将输入信号 Uin 连接至 VINP 对应的 GPIO 并将偏压输入连接至 VINN 对应的 GPIO。
- 将外部电容两端分别接到 VOUT 和 VINN1

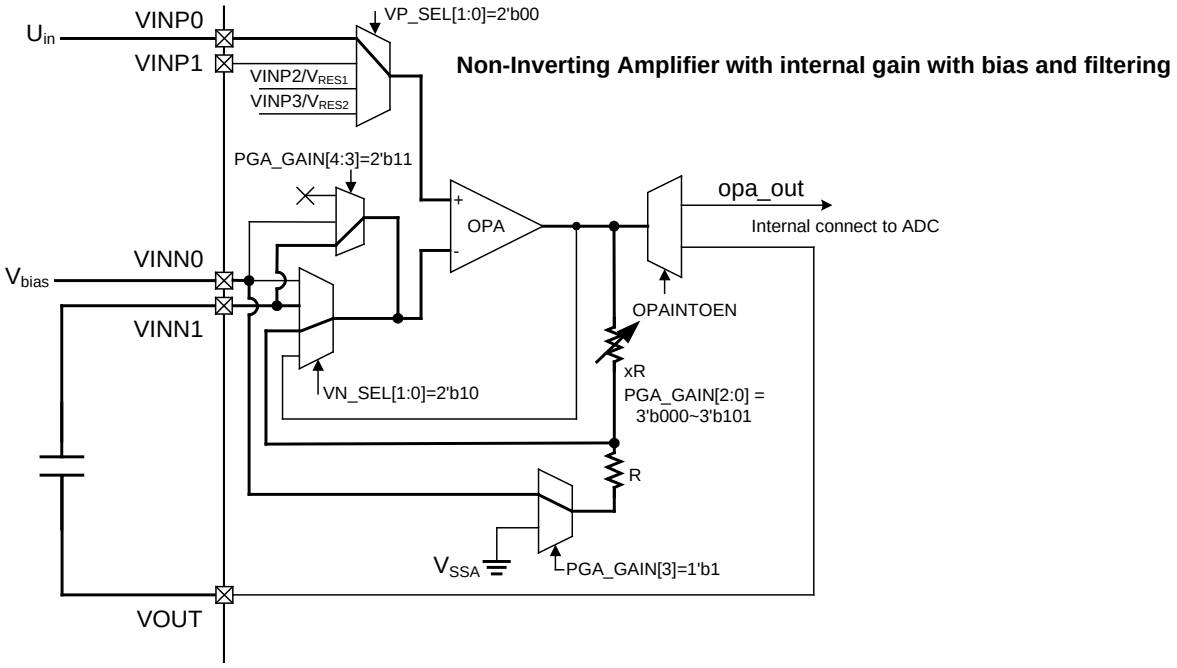


图 15-9　内部增益模式(正相闭回路放大和偏压，并搭配外部滤波电路)

注:Vbias voltage(Vbias)说明:反相闭回路放大 -反向放大器在输入 AC coupling 讯号下 OPAMP 的 VOUT 上的直流偏置由 VINP 输入的电压所决定。如果一个应用需要 VOUT 信号的电压居中于 VDDA/2 周围，那么 VINP 输入电压必须被偏置到相同的电压水平。

15. 4. 6. 8 内部增益模式:反相闭回路放大并搭配外部滤波电路

下面将介绍此模式下 OPAMP 的配置流程。

- 配置 **OPA_CSRx**.VN_SEL= 10，将输入信号 U_{in} 连接至 VINN 对应的 GPIO。
- 配置 **OPA_CSRx**.PGA_GAIN= 11000 ~ 11101，选择反相增益 -1、-3、-7、-15、-31、-63。
- 配置 **OPA_CSRx**.VP_SEL= 00，将 VINP 对应的 GPIO 接到 V_{SSA}。
- 将外部电容两端分别接到 VOUT 和 VINN1

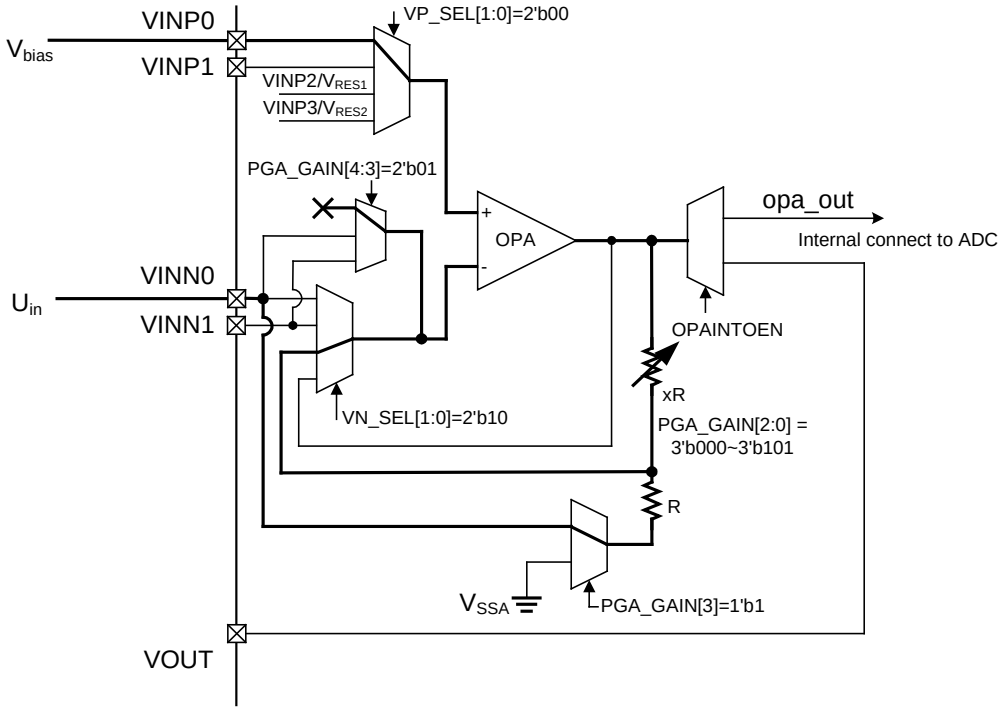


图 15-10 内部增益模式(反相闭回路放大并搭配外部滤波电路)

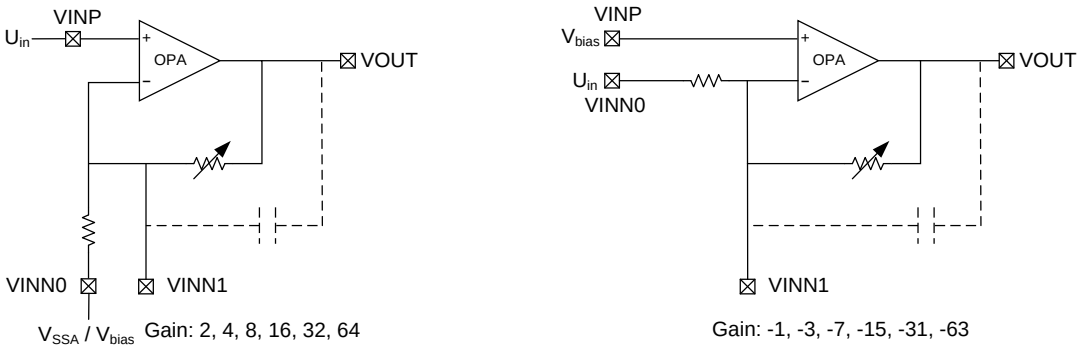


图 15-11 非反相与反相电路连接示例

15. 4. 7 OPAMP PGA增益

当 OPAMP 配置为内部增益配置模式时，可以选择正相增益 2、4、8、16、32、64 或反相增益-1、-3、-7、-15、-31、-63。

当 OPAMP 配置为正相闭回路放大模式时，增益完全取决于内部电阻分压器。当配置为反相模式时，增益因子的定义不仅只有芯片内反馈电阻，还包含了讯号源输出阻抗。如果讯号源输出阻抗与 PGA 的输入反馈电阻不匹配，则会产生增益误差。

15. 4. 8 OPAMP 校准方式

配置各个 OPAMP 的 **OPA_CSRx** 进行个别 OPAMP 校准。校准流程如下图，先配置适当输入电压以及缓存器数值后再配置 CALON 后开始进行校准。调整 TRIMOFFSETP[4:0]与 TRIMOFFSETN[4:0](校准值从大到小)，观测 **CAL_OUT** 输出结果，当输出从低变高时，此时的校准值加一即为所要的值。校准分别对输入的 P 端和 N 端进行，并将校准结果分别放入 TRIMP 和 TRIMN 变量。

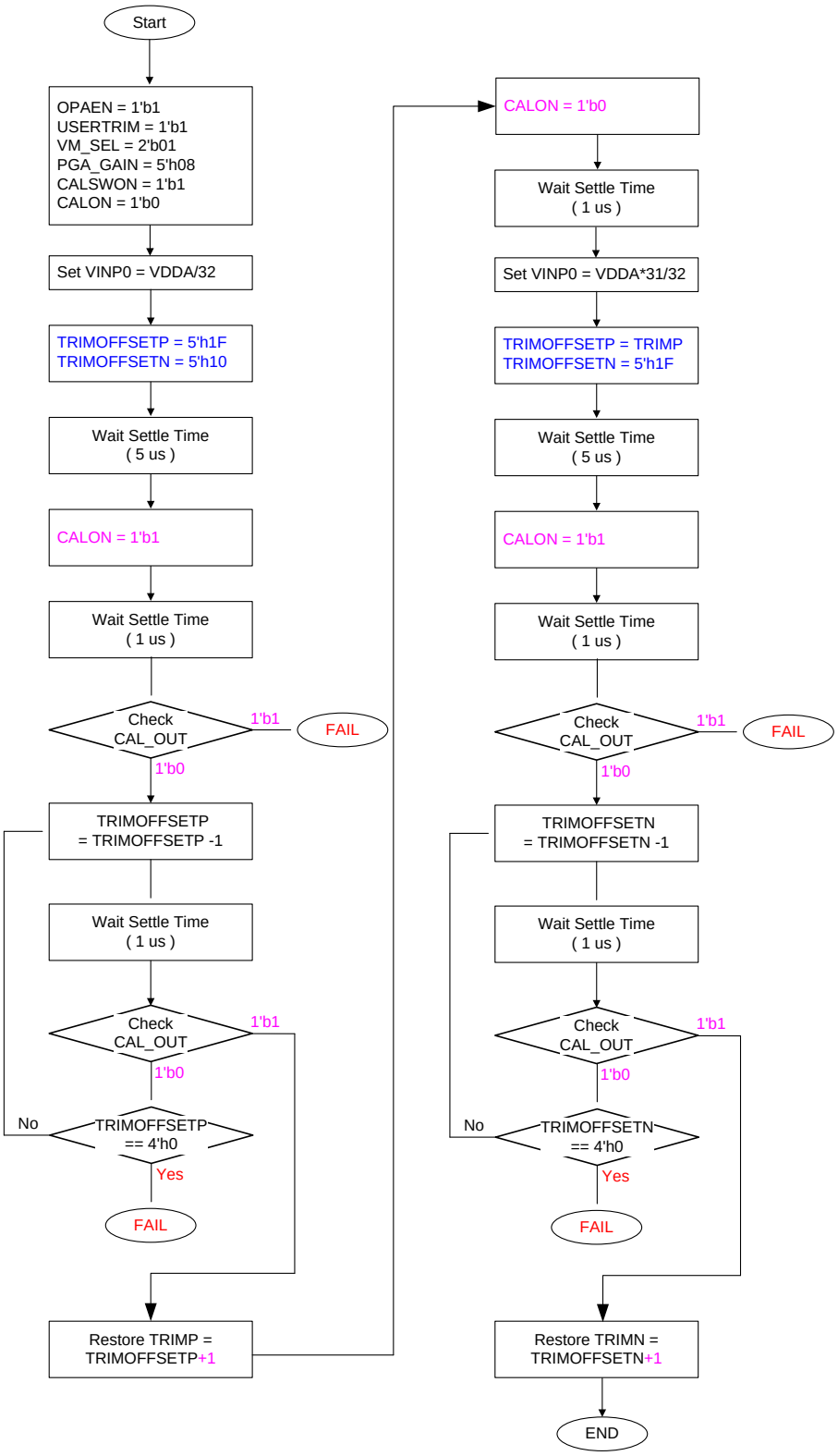


图 15-12　OPAMP 校准流程

15. 4. 9 OPAMP 定时器控制多任务器模式

OPAMP 反相和非反相输入的选择可以自动完成。在这种情况下，从一个输入切换到另一个输入是自动完成的。此自动切换是由 ADTimer1 或 ADTimer2 的输出，送至 OPAMP 输入多任务器触发。

这对于需要同时测量第一个电机和第二个电机的三相电流的双电机控制非常有用。

通过设置 **OPAx_TCMR**.ADxCM_EN 位 x = 1,2 来启用自动切换。反相和非反相输入选择是使用运算放大器的 OPAMP 定时器控制模式缓存器中的 VPS_SEL 和 VNS_SEL 位来执行的。如果 ADxCM_EN 位被清零，则使用 **OPA_CSRx** 中的 VP_SEL 和 VN_SEL 位来完成选择。

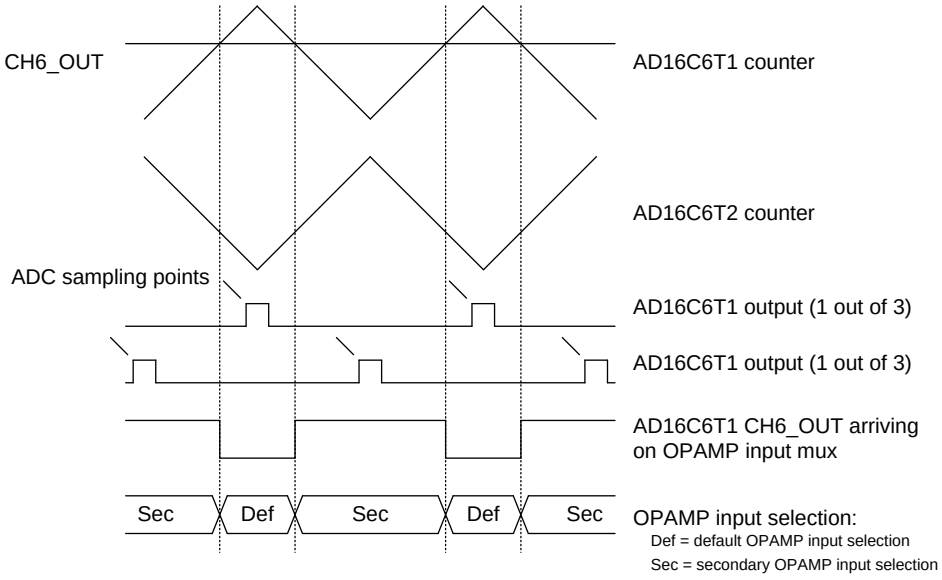


图 15-13 定时器控制多任务器模式

15. 4. 10 OPAMP 低功耗模式

模式	描述
SLEEP 模式	无影响
STOP 模式	无影响，OPAMP 寄存器数值将保持。

15. 5 特殊功能寄存器

15. 5. 1 寄存器列表

OPAMP 寄存器列表			
名称	偏移地址	类型	描述
OPA1_CSR	0000_H	R/W	OPA 1 配置寄存器
OPA2_CSR	0004_H	R/W	OPA 2 配置寄存器
OPA3_CSR	0008_H	R/W	OPA 3 配置寄存器
OPA4_CSR	$000C_H$	R/W	OPA 4 配置寄存器
OPA1_TCMR	0018_H	R/W	OPA 1 定时器控制寄存器
OPA2_TCMR	$001C_H$	R/W	OPA 2 定时器控制寄存器
OPA3_TCMR	0020_H	R/W	OPA 3 定时器控制寄存器
OPA4_TCMR	0024_H	R/W	OPA 4 定时器控制寄存器

15. 5. 2 寄存器描述

15. 5. 2. 1 OPAx配置寄存器 (OPAx_CSR) (x = 1,2,3,4)

OPAx 配置寄存器 (OPAx_CSR) (x = 1,2,3,4)																												
偏移地址:0x00、0x04、0x08、0x0C																												
复位值:0x0000 0100																												
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
LOCK	CAL_OUT	—	TRIMOFFSETN <4:0>					TRIMOFFSETP <4:0>					PGA_GAIN<4:0>					—	CALSWON	CALON	—	—	OPAINTOEN	OPAHSM	VM_SEL<1:0>		USERTRIM	VP_SEL<1:0>		—	OPAEN

LOCK	Bit 31	R/W	**Operational Amplifiers x 锁定** 锁定配置仅在复位后可写，一旦锁定需通过复位才可解除。 0: OPA_CSRx[31:0]位可正常读写操作。 1: OPA_CSRx[31:0]位仅供读操作。
CAL_OUT	Bit 30	R	**Operational Amplifiers x 输出** 当CAL_OUT从0转为1时，表示找到适当的校准数值。需要将CALON配置为0，才能开始下一次校准。
—	Bit 29	—	—
TRIMOFFSETN	Bit 28-24	R/W	**Operational Amplifiers x NMOS校准值**
TRIMOFFSETP	Bit 23-19	R/W	**Operational Amplifiers x PMOS校准值**
PGA_GAIN	Bit 18-14	R/W	**Operational Amplifiers x 放大倍率配置** 00000: 非反相增益= 2 00001: 非反相增益= 4 00010: 非反相增益= 8 00011: 非反相增益= 16 00100: 非反相增益= 32 00101: 非反相增益= 64 00110: 保留 00111: 保留 01000: 反相增益= -1/ 非反相增益=2，VINN0引脚连接输入信号或者偏压。 01001: 反相增益=-3/ 非反相增益=4，VINN0引脚连接输入信号或者偏压。 01010: 反相增益=-7 / 非反相增益=8，VINN0引脚连接输入信号或者偏压。

			01011: 反相增益=-15/ 非反相增益=16，VINN0引脚连接输入信号或者偏压。 01100: 反相增益=-31/ 非反相增益=32，VINN0引脚连接输入信号或者偏压。 01101: 反相增益= -63 / 非反相增益=64，VINN0引脚连接输入信号或者偏压。 01110: 保留 01111: 保留 10000: 非反相增益= 2，VINN0引脚上连接滤波电路。 10001: 非反相增益= 4，VINN0引脚上连接滤波电路。 10010: 非反相增益= 8，VINN0引脚上连接滤波电路。 10011: 非反相增益= 16，VINN0引脚上连接滤波电路。 10100: 非反相增益= 32，VINN0引脚上连接滤波电路。 10101: 非反相增益= 64，VINN0引脚上连接滤波电路。 10110: 保留 10111: 保留 11000: 反相增益= -1 / 非反相增益=2 ，VINN0引脚连接输入信号或者偏压，VINN1引脚上连接滤波电路。 11001: 反相增益= -3 / 非反相增益=4 ，VINN0引脚连接输入信号或者偏压，VINN1引脚上连接滤波电路。 11010: 反相增益= -7 / 非反相增益=8 ，VINN0引脚连接输入信号或者偏压，VINN1引脚上连接滤波电路。 11011: 反相增益= -15 / 非反相增益=16，VINN0引脚连接输入信号或者偏压，VINN1引脚上连接滤波电路。 11100: 反相增益= -31 / 非反相增益=32，VINN0引脚连接输入信号或者偏压，VINN1引脚上连接滤波电路。 11101: 反相增益= -63 / 非反相增益=64，

			VINN0引脚连接输入信号或者偏压，VINN1引脚上连接滤波电路。 11110: 保留 11111: 保留
—	Bit 13	—	—
CALSWON	Bit 12	R/W	**Operational Amplifiers x 校准切换开关** 0: 关闭正负端输入相连模式。 1: 开启正负端输入相连模式。
CALON	Bit 11	R/W	**Operational Amplifiers x 校准模式开关** 0: 关闭校准模式。 1: 开启校准模式。
—	Bit 9-10	-	—
OPAINTOEN	Bit 8	R/W	**Operational Amplifiers x 内部输出开关** 0: OPAMP输出联结至IO脚位。 1: OPAMP输出联结至内部ADC取样通道，并且与IO脚位断开。
OPAHSM	Bit 7	R/W	**Operational Amplifiers x 高速模式开关** 0: OPAMP为普通模式。 1: OPAMP为高速模式。
VN_SEL	Bit 6-5	R/W	**Operational Amplifiers x 负端输入选择** 00: VINN0连接至负端输入。 01: VINN1连接至负端输入。 10: 电阻反馈电路连接至负端输入。 11: OPAMP输出连接至负端输入。
USERTRIM	Bit 4	R/W	**Operational Amplifiers x 校准值选择开关** 0: 使用预设校准值。 1: 用户自定义校准值。
VP_SEL	Bit 3-2	R/W	**Operational Amplifiers x 正端输入选择** 00: VINP0 连接至正端输入。 01: VINP1 连接至正端输入。 10: VINP2 连接至正端输入。 11: VINP3 连接至正端输入。
—	Bit 1	—	—
OPAEN	Bit 0	R/W	**Operational Amplifiers x 开关** 0: OPAMP x 关闭。 1: OPAMP x 启动。

15. 5. 2. 2 OPAx 定时器控制寄存器 (OPAx_TCMR) (x = 1,2,3,4)

OPAx 定时器控制寄存器 (OPAx_TCMR) (x = 1,2,3,4)																												
偏移地址: 0x18、0x1C、0x20、0x24																												
复位值:0x0000 0000																												
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
LOCK	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	AD2CM_EN	AD1CM_EN	VPS_SEL		—	VMS_SEL

LOCK	Bit 31	R/W	**Operational Amplifiers x 锁定** 0: OPA_TCMRx[31:0]位可正常读写操作。 1: OPA_TCMRx[31:0]位仅供读操作。
—	Bit 30-6	—	—
AD2CM_EN	Bit 5	R/W	**高级定时器2多任务器控制开关 x** 0: 关闭多任务器自动切换。 1: 开启多任务器自动切换。
AD1CM_EN	Bit 4	R/W	**高级定时器1多任务器控制开关 x** 0: 关闭多任务器自动切换。 1: 开启多任务器自动切换。
VPS_SEL	Bit 3-2	R/W	**Operational Amplifiers x 正端输入第2组选择** 00: VINP0连接至正端输入。 01: VINP1连接至正端输入。 10: VINP2连接至正端输入。 11: VINP3连接至正端输入。
—	Bit 1	—	—
VNS_SEL	Bit 0	R/W	**Operational Amplifiers x 负端输入第2组选择** 若 VN_SEL = 00 或 01 : 0: VINN0连接至负端输入。 1: VINN1连接至负端输入。 若 VN_SEL = 10 或 11 : 0: 电阻反馈电路连接至负端输入(PGA模式)。 1: OPAMP输出连接至负端输入(跟随模式)。

第16章 基本扩展控制器局域网(BxCAN)

16. 1 概述

基本扩展 CAN(Basic Extended Controller Area Network)外设又称 BxCAN，可与 CAN 网络进行交互。该外设支持 2.0A 和 2.0B Active 版本的 CAN 协议规范，2.0A 版本的协议规范支持 11 位标准标识符，2.0B Active 版本的协议规范支持 11 位标准标识符和 29 位扩展标识符。

CAN 广泛应用于安全性较高的汽车电子和工业控制中，CAN 控制器支持 3 个优先级可配置的发送邮箱，2 个可以存储三级邮箱深度的接收 FIFO，同时硬件支持时间触发通信方案。

16. 2 特性

- 支持 2.0A 及 2.0B Active 版本的 CAN 协议规范
- 比特率高达 1Mbps
- 支持时间触发通信方案
- 在唯一地址空间通过软件实现高效的邮箱映射

发送

- 三个发送邮箱
- 可配置的发送优先级
- 支援发送中断

接收

- 两个具有三级邮箱深度的接收 FIFO
- 14 个可调整的筛选器组
- 标识符列表功能
- 可配置的 FIFO 上溢
- 支持接收中断

时间触发通信方案

- 禁止自动重发模式
- 专用的 16 位定时器
- 支持发送时间戳和接收时间戳

16. 3 结构图

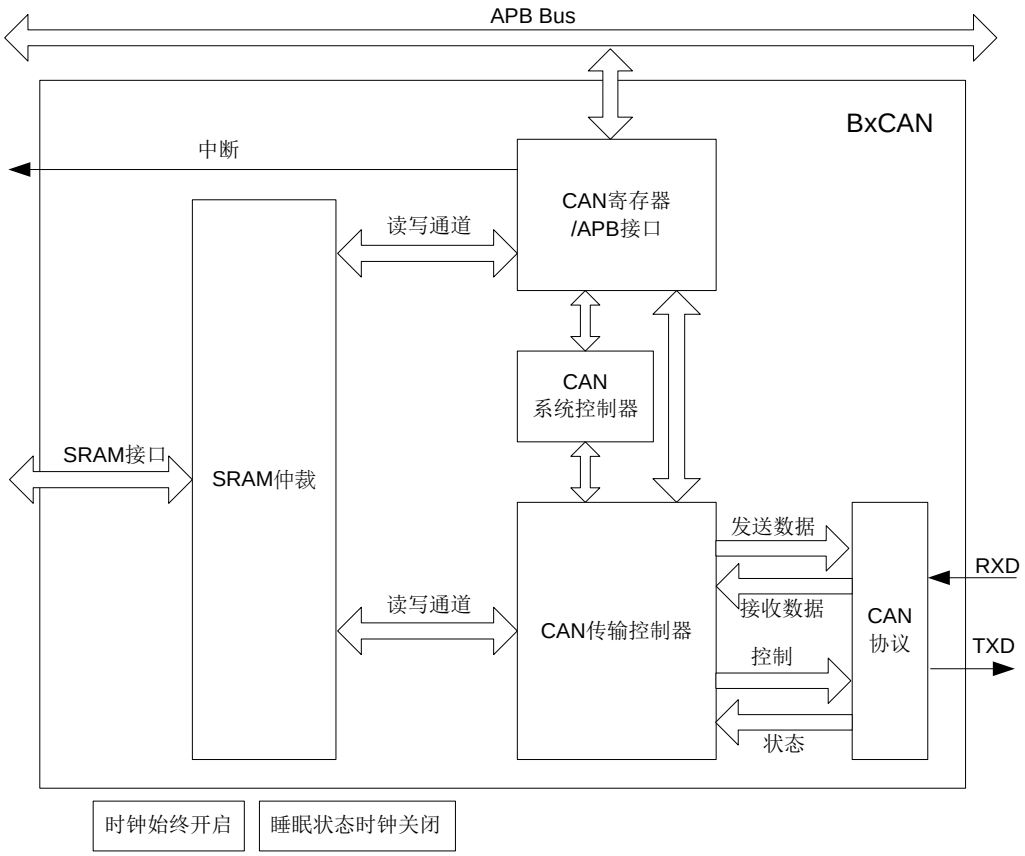


图 16-1　BxCAN 结构图

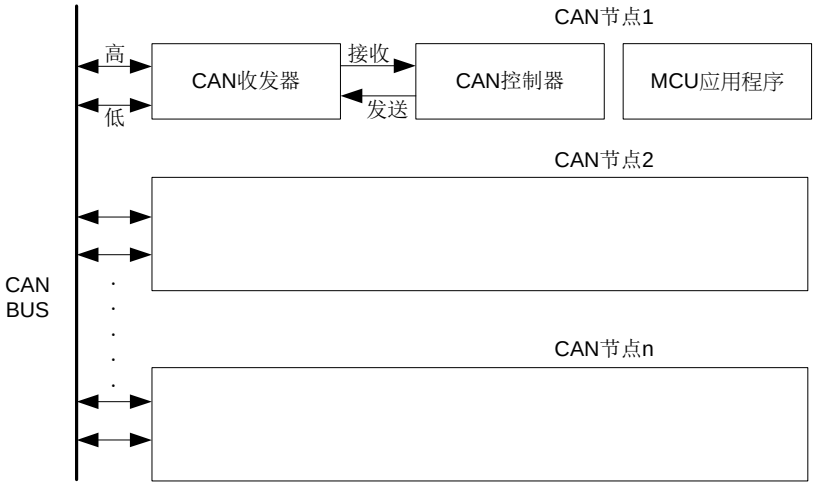
16. 4 功能描述

16. 4. 1 简介

在如今的 CAN 应用中，网络节点数量日益增多，经常需要通过网关将数个网络连接在一起，使得系统中的消息数量(以及各个节点需要处理的消息)有了显著增加。除应用程序的消息外，还引入了网络管理和诊断消息。

此外，应用程序任务需要更多的 CPU 时间，因此必须减少因消息接收而对实时处理造成的限制。

- 需要一个增强的筛选机制对各种类型的消息进行处理。
- 接收 FIFO 方案使 CPU 能够长时间专门处理应用程序任务，又不致丢失消息。基于标准 CAN 驱动程序的标准 HLP(更高层协议)需要一个高效接口来与 CAN 控制器连接。



16. 4. 2 CAN 2.0B

BxCAN2.0B 规范要求 CAN 收发器支持扩展格式的标识符(29 位)，同时能够兼容标准标识符(11 位)，且接收器可以自动识别出接收的是扩展帧还是标准帧。

16. 4. 3 CAN消息存储

CAN 消息的软件与硬件之间的接口通过邮箱实现。邮箱中包含所有与消息相关的信息：标识符、数据、控制、状态和时间戳信息。

发送邮箱

软件在空发送邮箱中设置将要发送的消息。发送状态由硬件在 CAN_TSTAT 寄存器中指示。

与发送邮箱基地址之间的偏移	寄存器名称
0	CAN_TXIDx
4	CAN_TXFCONx

8	CAN_TXDLx
12	CAN_TXDHx

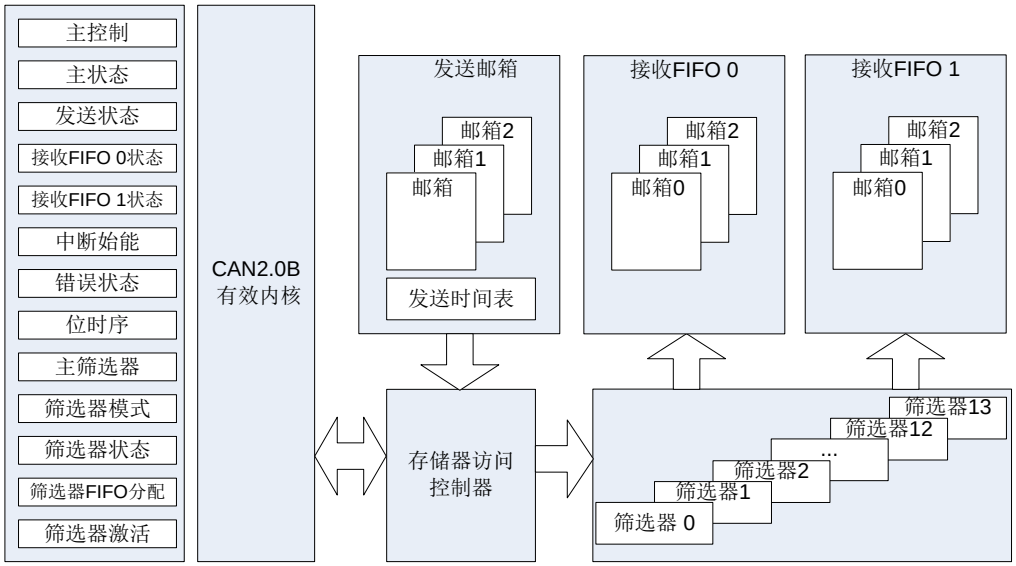
接收邮箱

消息在接收到后，将放在接收 FIFO 邮箱中供软件使用。一旦软件对消息进行了处理(例如读取)，则必须通过将 CAN_RXFx.FREE 位置 1 释放 FIFO 接收邮箱，以接收下一条传入消息。筛选器匹配索引存储在 CAN_RXFxINF.FLTIDX 中。16 位时间戳值存储在 CAN_RXFxINF.STAMP 字段中。

与接收邮箱基地址之间的偏移(字节)	寄存器名称
0	CAN_RXFxID
4	CAN_RXFxINF
8	CAN_RXFxDL
12	CAN_RXFxDH

CAN 邮箱 SRAM 存储

CAN 的发送邮箱和接收邮箱都是存储在 SRAM 中，可以存储 3 个发送邮箱。硬件使用两个接收 FIFO 来存储传入消息。每个 FIFO 中可以存储三条完整消息，FIFO 完全由硬件管理，接收邮箱寄存器访问的是最早存入 FIFO 的消息。



16. 4. 4 错误管理

如 CAN 协议所述，节点的故障状态分为错误主动、错误被动、总线关闭三种。

错误管理完全由硬件通过发送错误计数器(CAN_ERRSTAT.TXERRC 值)和接收错误计数器(CAN_ERRSTAT.RXERRC 值)来处理，这两个计数器根据错误状况进行递增或递减。

两者均可由软件读取，用以确定网络的稳定性。此外，CAN 硬件还将在 CAN_ERRSTAT 寄存器中提供当前错误状态的详细信息。通过 CAN_IE.ERRIE 位，软件可以非常灵活地配置在检测到错误时生成的中断。

总线关闭恢复

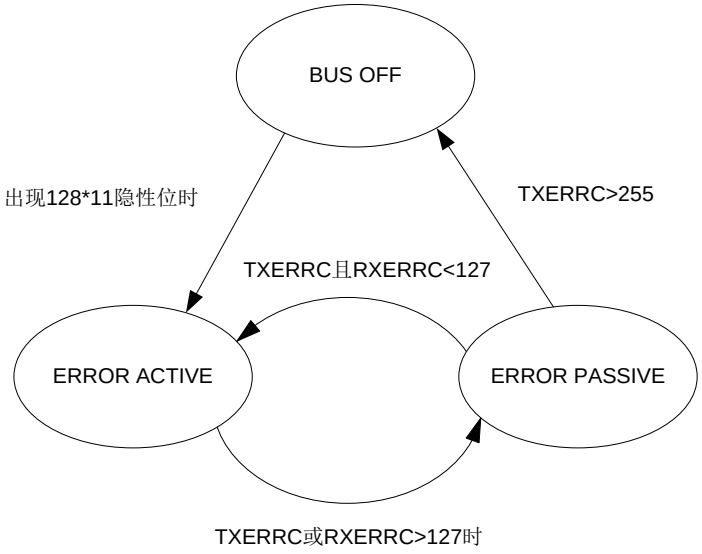
当 TXERRC 大于 255 时，节点变为“总线关闭”状态，该状态由 CAN_ERRSTAT.BOFF 位指示。在“总线关闭”状态下，BxCAN 不能再发送和接收消息。

BxCAN 可以自动或者由软件请求从“总线关闭”状态恢复为“错误主动”状态，具体取决于 CAN_CON.ABOFFEN 位。但在这两种情况下，BxCAN 都必须等待至少一个 CAN 标准恢复序列(在 CAN 总线上监测到 128 次 11 个连续隐性位)。

如果 CAN_CON.ABOFFEN=1，BxCAN 将在进入“总线关闭”状态后自动启动恢复序列。

如果 CAN_CON.ABOFFEN=0，则软件必须请求 BxCAN 先进入初始化模式再退出初始化(CAN_CON.INIREQ 置 1 再清零)，从而启动恢复序列。

注：在初始化模式下，BxCAN 不会监视 CANRX 信号，因此无法完成恢复序列。要进行恢复，BxCAN 必须处于正常模式。



16. 4. 5 位时序

位时序逻辑将监视 CAN 总线，执行采样并调整采样点，在调整采样点时，需要在起始位边沿进行同步并在后续的边沿进行再同步。

通过将标称位时间划分为以下三段，即可解释其工作过程：

- 同步段(SYNC_SEG)：位变化应该在此时间段内发生。它只有一个时间片的固定长度(1 x t CAN)。

- 位段 1(SEG1)：定义采样点的位置。它包括 CAN 标准的 PROP_SEG 和 PHASE_SEG1。其持续长度可以在 1 到 16 个时间片之间调整，但也可以自动加长，以补偿由不同网络节点的频率差异所导致的正相位漂移。
- 位段 2(SEG2)：定义发送点的位置。它代表 CAN 标准的 PHASE_SEG2。其持续长度可以在 1 到 8 个时间片之间调整，但也可以自动缩短，以补偿负相位漂移。

再同步跳转宽度(RESJW)定义位段加长或缩短的上限。它可以在 1 到 4 个时间片之间调整。

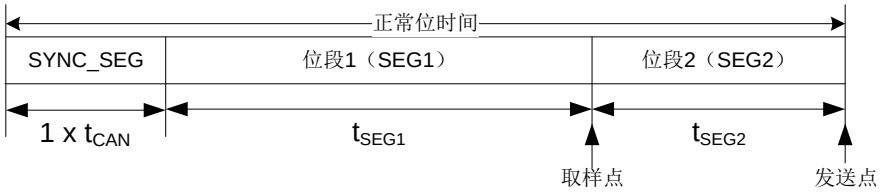
有效边沿是指一个位时间内总线电平从隐性到显性的第一次转换(前提是控制器本身不发送隐性位)。

如果在 SEG1 而不是 SYNC_SEG 中检测到有效边沿，则 SEG1 会延长最多 RESJW，以便延迟采样点。

相反地，如果在 SEG2 而不是 SYNC_SEG 中检测到有效边沿，则 SEG2 会缩短最多 RESJW，以便提前发送点。

为了避免程序设置错误，位时序寄存器(CAN_BTIME)只能在器件处于初始化模式时进行配置。

注：在初始化模式下，BxCAN 不会监视 CANRX 信号，因此无法完成恢复序列。要进行恢复，BxCAN 必须处于正常模式。



$$\text{波特率} = \frac{1}{\text{正常位时间}}$$

$$\text{正常位时间} = 1 \times t_{can} + t_{SEG1} + t_{SEG2}$$

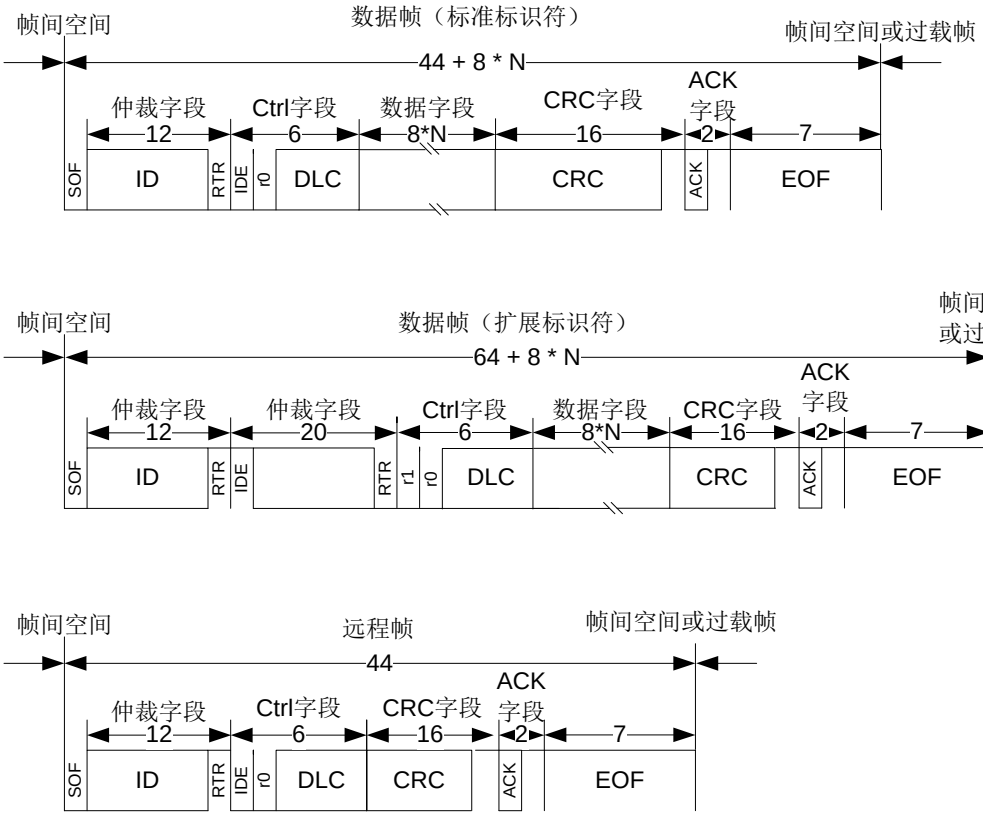
其中：

$$t_{SEG1} = t_{can} \times SEG1 + 1$$

$$t_{SEG2} = t_{can} \times SEG2 + 1$$

$$t_{can} = (BPSC[8:0] + 1) \times t_{PCLK}$$

BPSC[8:0]、SEG1 和 SEG2 在 CAN_BTIME 寄存器中的定义。



注：0≤N≤8，SOF=帧起始，ID=标识符，RTR=远程传输请求，IDE=标识符扩展位，r0=保留位，DLC=数据长度代码，CRC=循环冗余代码，EOF=帧结束，ACK=确认位，Ctrl=控制，SRR=替代远程请求位，r1=保留位

16. 4. 6 工作模式

16. 4. 6. 1 初始化模式

在初始化模式下，所有从 CAN 总线传入和传出的消息都将停止，并且 CAN 总线输出 CANTX 的状态为隐性(高)。

通过将 CAN_CON.INIREQ 置 1 以进入初始化模式，进入初始化模式不会更改任何配置寄存器。

为初始化与 CAN 筛选器组相关的寄存器(模式、宽度、FIFO 分配、启动和筛选器值)，软件必须将 CAN_FLTCON.FLTINI 置 1。当进入筛选器初始化模式时，CAN 接收停用。筛选器的初始化也可以在初始化模式之外进行。

筛选器值也可通过停用(CAN_FLTGO 寄存器)相关筛选器启动位来修改。

如果某个筛选器组未使用，建议将其保持为未启动状态(将相应 CAN_FLTGO.GO 位保持清零)。

16. 4. 6. 2 正常模式

一旦初始化完成，软件必须向硬件请求进入正常模式，这样才能在 CAN 总线上进行同步，并开始接收和发送。

进入正常模式的请求可通过 CAN_CON.INIREQ 清 0 来实现。BxCAN 进入正常模式，并与 CAN 总线上的数据传输实现同步后(RX 上检测到 11 个连续隐性位)，即可参与总线活动。硬件通过将 CAN_STAT.INISTAT 清 0，来确认切换到正常模式。

筛选器的相关配置必须在进入正常模式之前完成。

16. 4. 6. 3 睡眠模式

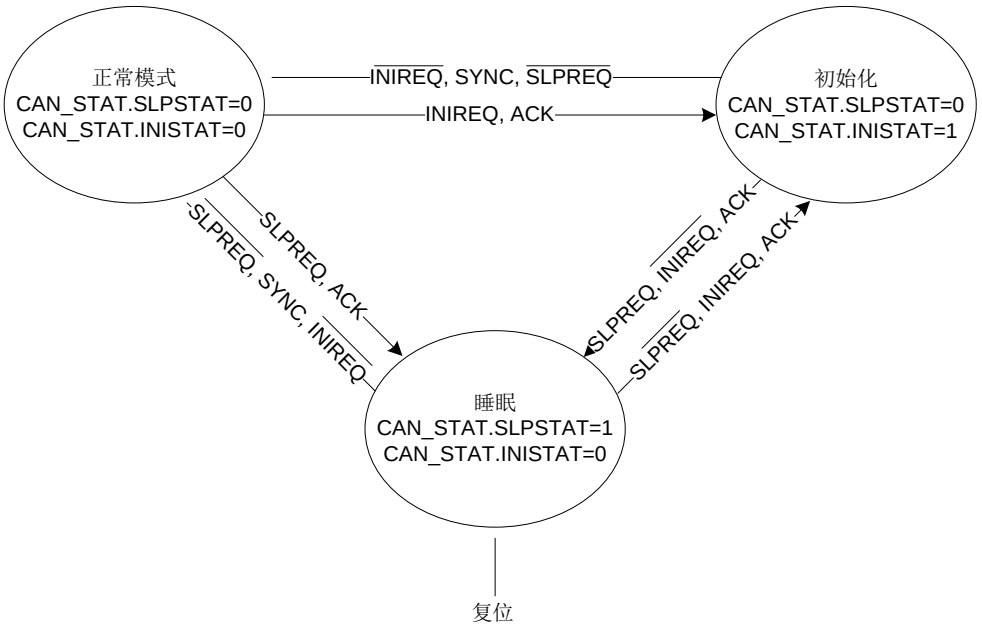
为降低能耗功耗，BxCAN 具有低功耗模式，称为睡眠模式。在正常模式下软件通过将 CAN_CON.SLPREQ 置 1 发出请求，即可进入该模式。该模式下，BxCAN 时钟停止，但软件仍可访问 BxCAN 邮箱。

在 BxCAN 处于睡眠模式时，如果软件通过将 CAN_CON.INIREQ 置 1 来请求进入初始化模式，则必须同时将 CAN_CON.SLPREQ 位清零。

软件将 CAN_CON.SLPREQ 位清零或是检测到 CAN 总线活动时，BxCAN 即被唤醒(退出睡眠模式)。

检测到 CAN 总线活动后，如果 CAN_CON.AWKEN=1，硬件将通过清零 CAN_CON.SLPREQ 位来自动执行唤醒序列。如果 CAN_CON.AWKEN=0，在发生唤醒中断时，软件必须将 CAN_CON.SLPREQ 位清零才能退出睡眠模式。

注：如果使能唤醒中断(CAN_IE.WKIE=1)，一旦检测到 CAN 总线活动，即使 BxCAN 自动执行唤醒序列，也会发生唤醒中断。



- ACK 为硬件通过将 CAN_STAT 寄存器的 INISTAT 或 SLPSTAT 位置 1 来确认请求的等待状态。
- SYNC 为 BxCAN 等待 CAN 总线变为空闲(即在 CANRX 上监测到连续 11 个隐性位)的状态。

由睡眠模式进入初始化模式

- 将寄存器 CAN_CON.SLPREQ=0
- 将寄存器 CAN_CON.INIREQ=1
- 等待寄存器 CAN_STAT.INISTAT=1

由初始化进入正常模式

- 将寄存器 CAN_CON.INIREQ=0
- 等待总线同步，CANRX 上检测到连续 11 个隐性位

由正常模式进入睡眠模式

- 将寄存器 CAN_CON.SLPREQ=1
- 等待寄存器 CAN_STAT.SLPSTAT=1

16. 4. 7 发送处理

16. 4. 7. 1 发送处理

BxCAN 提供三个发送邮箱，为了发送消息，应用程序必须在请求发送(CAN_TXIDx.TXMREQ=1)前，选择一个空发送邮箱，并设置标识符类型、数据长度(DLC)和要发送的数据。CAN_TXIDx.TXMREQ=1 时，邮箱变为非空状态，立即进入挂起态，软件不再具有对发送邮箱寄存器的写访问权限。接着该邮箱等待成为优先级最高的邮箱，请参见章节：发送优先级。一旦邮箱拥有最高优先级，即被安排发送。CAN 总线变为空闲后，被安排好的邮箱中的消息即开

始发送(进入发送状态)。邮箱一旦发送成功，即恢复空状态。硬件通过将 CAN_TXSTAT 寄存器的 MxREQC 和 MxTXC 位置 1，来表示发送成功。

如果发送失败，失败原因将由 CAN_TXSTAT 寄存器的 MxARBLST 位(仲裁丢失)或 MxTXERR 位(检测到发送错误)指示。

16. 4. 7. 2 发送优先级

当多个发送邮箱挂起时，发送顺序可以按标识符决定或按发送请求顺序决定，具体取决于寄存器 CAN 主控制寄存器 CAN_CON.TXMP 位。

- 按标识符

  当 CAN_CON.TXMP=0 时，发送顺序由邮箱中所存储消息的标识符来确定。根据 CAN 协议的仲裁，标识符值最小的消息具有最高的优先级。如果标识符值相等，则首先安排发送编号较小的邮箱。

- 按发送请求顺序

  当 CAN_CON.TXMP=1 时，发送优先级顺序按照发送请求顺序来确定，先请求的邮箱优先发送。

16. 4. 7. 3 发送停止

处于挂起态、已安排状态或发送态的邮箱可以通过软件将其终止发送。

处于挂起态或已安排状态下的邮箱，将 CAN_TXSTAT.MxSTPREQ 置 1，发送请求即被终止。

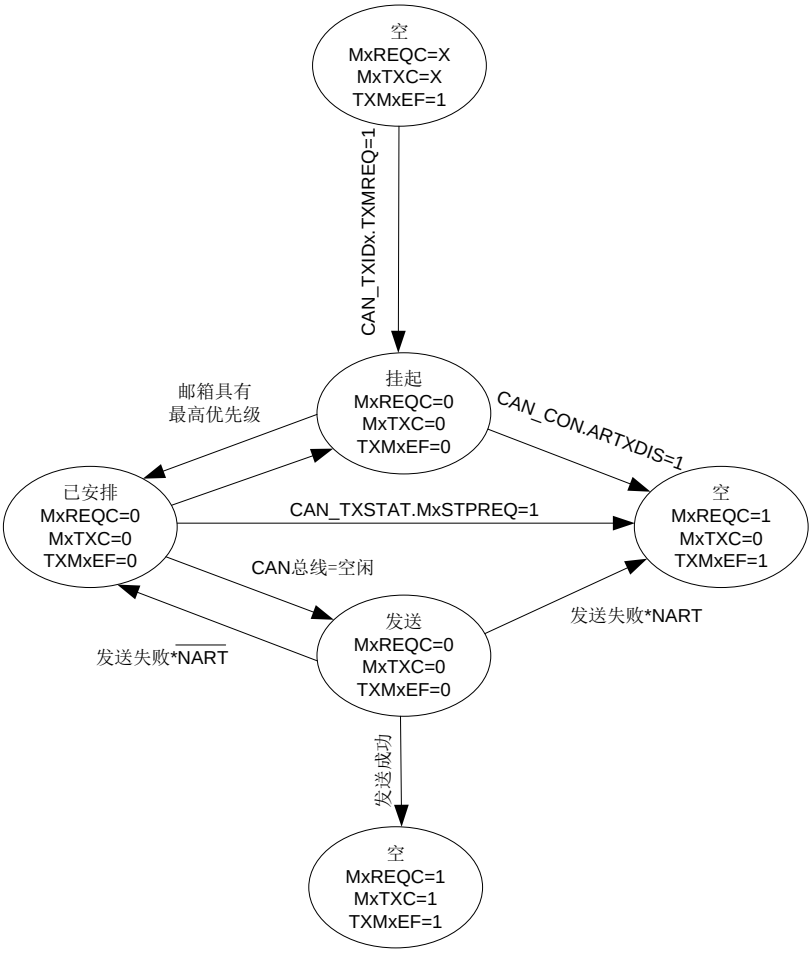
处于发送态的邮箱，将 CAN_TSTAT.MxSTPREQ=1，可能会出现两种结果：如果邮箱发送成功，将变为空状态，同时 CAN_TSTAT.MxTXC=1；如果发送失败，邮箱变为已安排状态，发送中止并变为空状态，CAN_TSTAT.MxTXC=0。

16. 4. 7. 4 禁止自动重发送模式

该模式旨在满足 CAN 标准的时间触发通信方案的要求。要将硬件配置为此模式，必须将 CAN_CON. ARTXDIS 置 1。

在此模式下，每个发送仅启动一次。如果由于仲裁丢失或错误导致第一次尝试失败，硬件将不会自动重新启动消息发送。

第一次发送尝试结束时，硬件将认为请求已完成，并将 CAN_TXSTAT.MxREQC 置 1。发送结果由 CAN_TXSTAT 寄存器的 MxTXC、MxARBLST 和 MxTXERR 位来指示。



16. 4. 7. 5　　时间触发通信模式

在此模式下，CAN 硬件的内部计数器启动，用于为接收和发送邮箱生成时间戳值，这些值分别存储在寄存器 CAN_TXFCONx.STAMP 和 CAN_RXFxINF.STAMP 寄存器中。内部计数器在每个 CAN 位时间加 1(请参见章节：位时序)。在接收和发送时，都会在帧起始位的采样点捕获内部计数器。

16. 4. 7. 6　　发送消息流程

1. 选择一个空发送邮箱，查找 CAN_TXSTAT.TXMxEF 为 1 的邮箱
2. 配置 4 个发送邮箱寄存器 CAN_TXIDx、CAN_TXFCONx、CAN_TXDLx、CAN_TXDHx
3. 将 CAN_TXIDx.TXMREQ 的位置 1
4. 检查发送状态寄存器 TXSTAT 的 MxREQC、MxTXC 和 TXMxEF 是否为 1

16. 4. 8　接收处理

BxCAN 提供了两个三级邮箱深度的接收 FIFO，为了节约 CPU 负载，简化软件并保证数据一致性，FIFO 完全由硬件进行管理。应用程序通过读取接收 FIFO 来获得最先存入 FIFO 的消息。

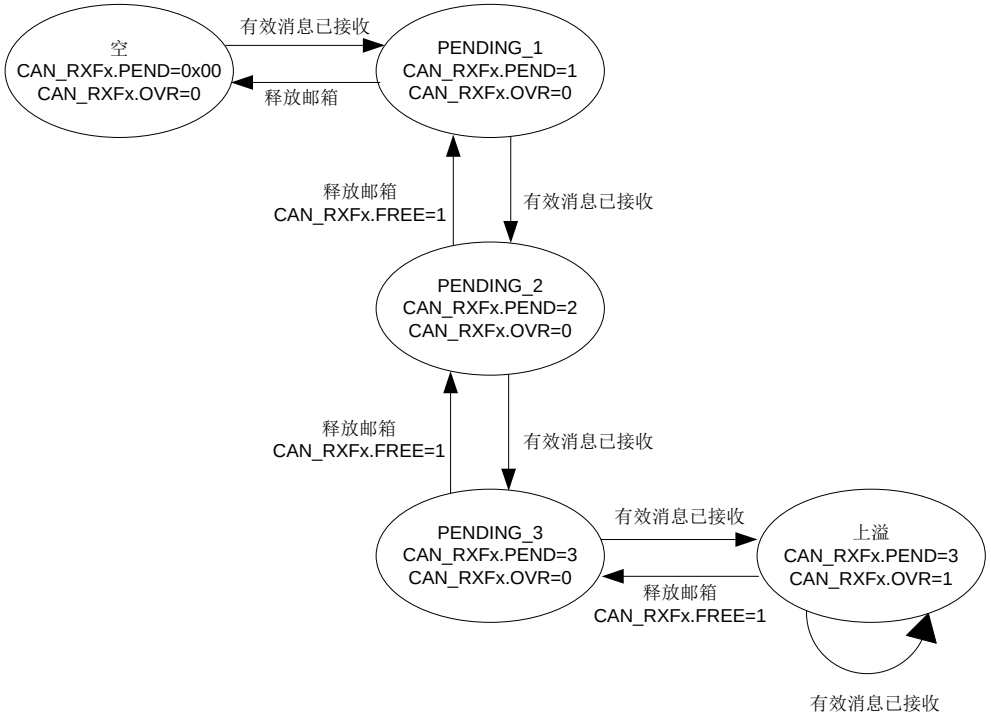
16. 4. 8. 1 有效消息

当消息依据 CAN 协议正确接收(没有发送错误)，并且成功通过了标识符筛选后，该消息被视为有效。有关标识符筛选请参见章节：标识符筛选器。

16. 4. 8. 2 FIFO管理

FIFO 开始时处于空状态，在接收的第一条有效消息存储在其中后，变为 Pending_1 状态。硬件通过将 CAN_RXFx.PEND 置 1 来指示该事件。消息将放在接收 FIFO 中供软件读取。软件读取邮箱内容后，通过将 CAN_RXFx.FREE 置 1 将邮箱释放，该接收 FIFO 便会恢复空状态。如果同时接收到新的有效消息，FIFO 将保持 Pending_1 状态，新消息将在输出邮箱中供读取。

如果应用程序未释放邮箱，下一条有效消息将继续存储在 FIFO 中，使其进入 Pending_2 状态(CAN_RXFx.PEND=2)。下一条有效消息会重复该存储过程，同时将 FIFO 变为 Pending_3 状态(CAN_RXFx.PEND=3)。此时，软件必须通过将 CAN_RXFx.FREE 置 1 来释放输出邮箱，从而留出一个空邮箱来存储下一条有效消息。否则，下一次接收到有效消息时，将导致消息丢失。



16. 4. 8. 3 上溢

一旦 FIFO 处于 Pending_3 状态(即三个邮箱均已满)，则下一次接收到有效消息时，将导致上溢并丢失一条消息。硬件通过将 CAN_RXFx.OVR 置 1 来指示上溢状态。具体丢失哪一条消息取决于 FIFO 的配置：

如果禁止 FIFO 锁定功能(CAN_CON.RXFOPM=0)，则新传入的消息将覆盖 FIFO 中存储的最后一条消息。在这种情况下，应用程序将始终能访问到最新的消息。
如果使能 FIFO 锁定功能(CAN_CON.RXFOPM=1)，则将丢弃最新的消息，软件将提供 FIFO 中最早的三条消息。

16. 4. 8. 4 接收FIFO中断

当使能消息挂起中断 CAN_IE.FxPIE 后，收到消息存储到 FIFO 中，CAN_RXFx.PEND 位非 0，即产生中断请求。

当使能 FIFO 满中断 CAN_IE.FxFULIE 后，FIFO 存满消息(即存储了第三条消息)后，CAN_RXFx.FULL 为 1，将产生满中断。

当使能 FIFIO 溢出中断 CAN_IE.FxOVRIE 后，出现上溢时，CAN_RXFx.OVR 位置 1，将产生溢出中断。

16. 4. 8. 5 接收消息流程

1. 等待接收 FIFO 寄存器 CAN_RXFx.PEND 非 0
2. 读取相关接收 FIFO 邮箱寄存器：CAN_RXFxID、CAN_RXFxINF、CAN_RXFxDL 和 CAN_RXFxDH
3. 置位 CAN_RXFx.FREE 释放接收 FIFO

16. 4. 8. 6 标识符筛选器

在 CAN 协议中，消息的标识符与节点地址无关，但与消息内容有关。因此，发送器将消息广播给所有接收器。在接收到消息时，接收器节点会根据标识符的值来确定软件是否需要该消息。如果需要，该消息将复制到 SRAM 中。如果不需要，硬件自动丢弃该消息。

为了实现这一功能，BxCAN 控制器为应用程序提供了 14 个可配置且可调整的硬件筛选器组(0-13)，以便接收器仅接收软件需要的消息。此硬件筛选功能可以节省软件筛选所需的 CPU 资源。每个筛选器组 x 均包含两个 32 位寄存器，分别是 CAN_FLTxR1 和 CAN_FLTxR2。

16. 4. 8. 7 可调整的宽度

为了根据应用程序的需求来优化和调整筛选器，每个筛选器组可分别进行伸缩调整。根据筛选器宽度不同，一个筛选器组可以：

- 为 STDID[10:0]、EXTID[17:0]、IDE 和 RTR 位提供一个 32 位筛选器。
- 为 STDID[10:0]、RTR、IDE 和 EXTID[17:15]位提供两个 16 位筛选器。

此外，筛选器还可配置为屏蔽模式或标识符列表模式。

16. 4. 8. 8 屏蔽模式

在屏蔽模式下，标识符寄存器与屏蔽寄存器关联，用以指示标识符的哪些位“必须匹配”，哪些位“无关”，CAN_FLTxR1 设定匹配 ID，CAN_FLTxR2 设定哪些位无需匹配。接收器可以接收标识符中“必须匹配位”所匹配的所有消息。

16. 4. 8. 9 标识符列表模式

在标识符列表模式下，CAN_FLTxR1 和 CAN_FLTxR2 都用于设定匹配 ID，传入标识符的所有位都必须与筛选器寄存器匹配才可以被接收器接收。

16. 4. 8. 10 筛选器组宽度和模式配置

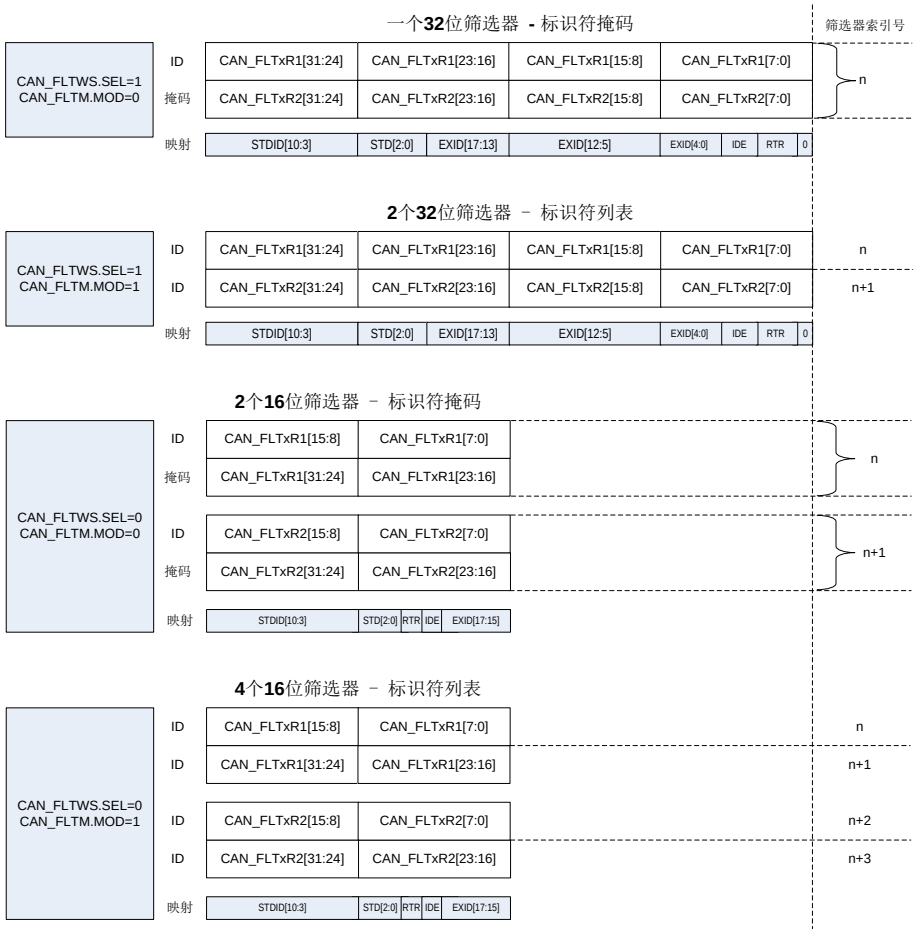
筛选器组通过相应的 CAN_FLTCON 寄存器进行配置。为了配置筛选器组，必须通过将筛选器启用寄存器 CAN_FLTGO 的对应位清零而将其停用。筛选器宽度通过筛选器宽度选择寄存器 CAN_FLTWS 的对应位进行配置。相应屏蔽/标识符寄存器的标识符列表或标识符屏蔽模式通过筛选器模式寄存器 CAN_FLTM 的对应位进行配置。

要筛选一组标识符，应将屏蔽/标识符寄存器配置为屏蔽模式。

要选择单个标识符，应将屏蔽/标识符寄存器配置为标识符列表模式。

未由应用程序使用的筛选器应保持停用。

筛选器组中的每个筛选器将按从 0 到最大值的顺序进行编号(称为筛选器索引号)，具体取决于每个筛选器组的模式和宽度。



注：

1. 上图中，x=筛选器组编号，ID=标识符。
2. 筛选标准 ID 时需要注意：若工作在标识符列表模式时，EXID 部分需要设为 0；若工作在屏蔽模式时，如果屏蔽没有屏蔽 EXID 部分时，EXID 部分需要设为 0。

16. 4. 8. 11 筛选器匹配索引

消息接收到 FIFO 中后，即可供应用程序使用。应用程序通常会复制到 SRAM 中的位置。为了将数据复制到正确的位置，应用程序必须通过标识符来识别数据。为了方便访问 SRAM 位置，CAN 控制器提供了一个筛选器匹配索引。

该索引根据筛选器优先级规则与消息一同存储在邮箱中。因此，每条收到的消息都有相关联的筛选器匹配索引。

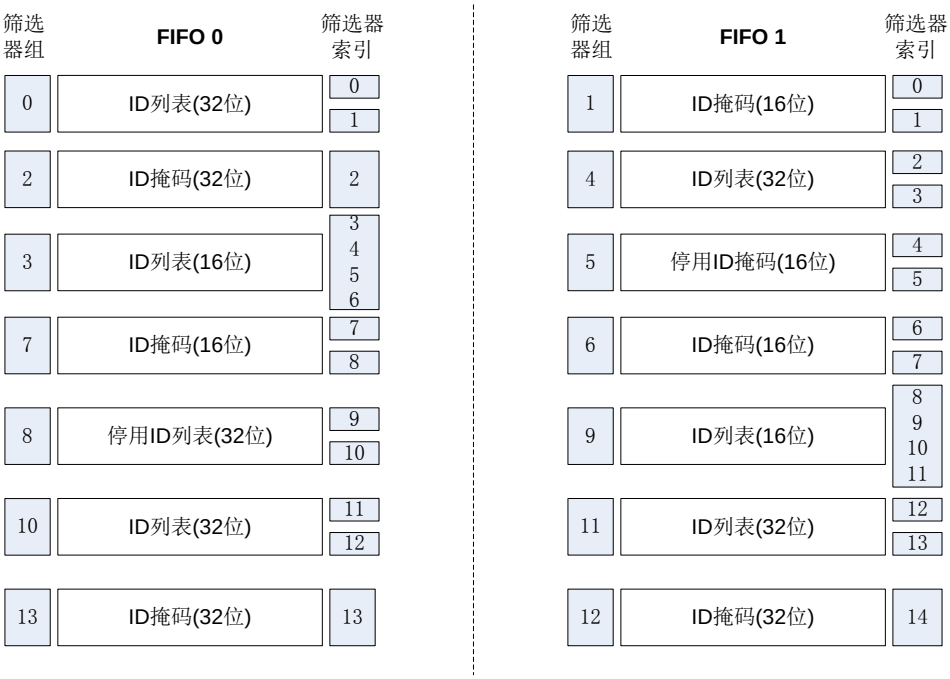
筛选器匹配索引的使用方法有两种：

- 将筛选器匹配索引与预期值列表进行比较。
- 将筛选器匹配索引用作数组索引，以访问数据目标位置。

对于标识符列表筛选器，软件不再需要比较标识符。

对于屏蔽模式筛选器，软件则只需比较屏蔽位。

筛选器编号的索引值与筛选器组的启动状态无关。此外，两个 FIFO 使用两个独立的编号方案，每个 FIFO 各一个。



16. 4. 8. 12　筛选器优先级规则

根据筛选器组合，可能会出现一个标识符成功通过数个筛选器的情况。这种情况下，将根据以下优先级规则选择接收邮箱中存储的筛选器匹配值：

- 32 位筛选器优先于 16 位筛选器。
- 对于宽度相等的筛选器，标识符列表模式优先于标识符屏蔽模式。
- 对于宽度和模式均相等的筛选器，则按筛选器编号确定优先级(编号越低，优先级越高)。

16. 4. 9　测试模式

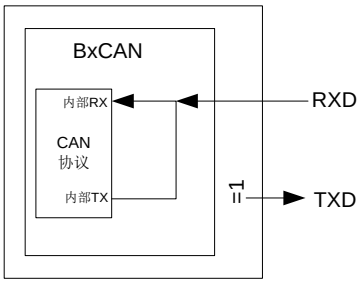
测试模式用于 CAN 设备的调试与自检，BxCAN 在初始化模式时设置 CAN_BTTIME 寄存器中的 SILENT 和 LOOP 位来进入测试模式。选择测试模式后，必须清除 CAN_CON 寄存器中的 INIREQ 位才能进入正常模式。

16. 4. 9. 1　静默模式

在初始化模式时将 CAN_BTIME 寄存器的 SILENT 位置 1，BxCAN 进入静默模式。

在静默模式下，BxCAN 可以有效接收总线上的数据帧和远程帧。但 CAN 发送通道与 CAN 总线断开，无法向 CAN 总线发出显性位，BxCAN 发送的显性位仍可以被 CAN 内核监视。静默

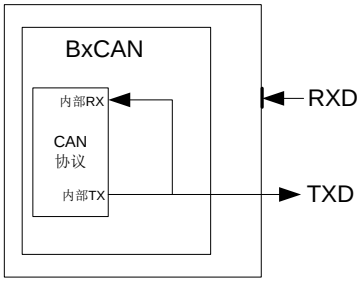
模式下，BxCAN 发送对总线保持隐性状态，通常用于分析 CAN 总线上的流量。



16. 4. 9. 2　　回环模式

在初始化模式时将 CAN_BTIME.LOOP 位置 1，BxCAN 进入回环模式。

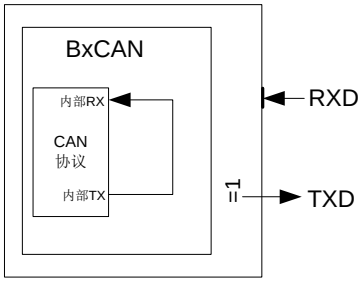
在回环模式下，BxCAN 将其自身发送的消息作为接收的消息来处理并存储(如果这些消息通过了验收筛选)在接收 FIFO 中，同时发送的消息会进入 CAN 总线网络。



16. 4. 9. 3　　回环与静默组合模式

在初始化模式时将 CAN_BTIME 寄存器的 LOOP 和 SILENT 位置 1，BxCAN 进入回环与静默组合模式。

该模式可用于“热自检”，也就是说，BxCAN 的发送通道和接收通道与总线完全断开，BxCAN 即不受 CAN 总线影响，也不影响 CAN 总线。BxCAN 发送的数据可以被 CAN 内核自己接收。



16. 4. 10 调试模式

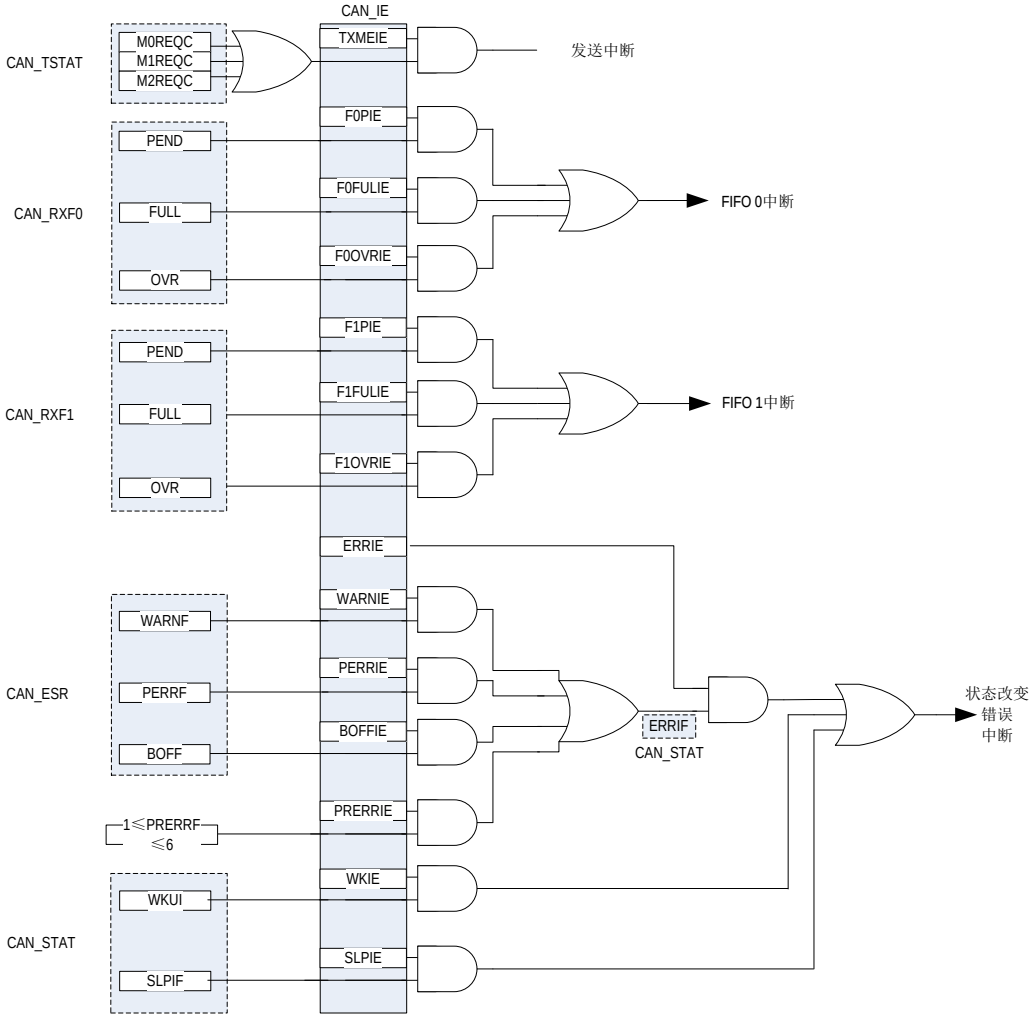
当微控制器进入调试模式时，BxCAN 可以继续正常工作，也可以停止工作，具体取决于如下寄存器的指定位的值：

- DBG 模块中 APB1 外设调试冻结寄存器的 DBG_APB1FZ 的 CAN_STOP 位。
- CAN_CON.DBGSTP 位。

16. 4. 11 中断

BxCAN 共有四个专用的中断向量：发送中断、FIFO0 中断、FIFO1 中断和状态改变错误中断。每个中断源均可通过 CAN 中断使能寄存器(CAN_IE)来单独地使能或禁止。

- 发送中断可由以下事件产生：
  - 发送邮箱 0 变为空，CAN_TXSTAT.M0REQC 位置 1
  - 发送邮箱 1 变为空，CAN_TXSTAT.M1REQC 位置 1
  - 发送邮箱 2 变为空，CAN_TXSTAT.M2REQC 位置 1
- FIFO0 中断可由以下事件产生：
  - 接收到新消息，CAN_RXF0.PEND 位非 0
  - FIFO0 满，CAN_RXF0.FULL 位置 1
  - FIFO0 上溢，CAN_RXF0.OVR 位置 1
- FIFO1 中断可由以下事件产生：
  - 接收到新消息，CAN_RXF1.PEND 位非 0
  - FIFO1 满，CAN_RXF1.FULL 位置 1
  - FIFO1 上溢，CAN_RXF1.OVR 位置 1
- 状态改变和错误中断可由以下事件产生：
  - 唤醒状况，CAN Rx 信号上监测到 SOF
  - 进入睡眠模式
  - 错误状况，有关错误状况的更多详细信息，请参见 CAN 错误状态寄存器(CAN_ERRSTAT)



16. 5 特殊功能寄存器

16. 5. 1 寄存器列表

OPAMP 寄存器列表			
名称	偏移地址	类型	描述
CAN_CON	0000_H	R/W	CAN 控制寄存器
CAN_STAT	0004_H	R	CAN 状态寄存器
CAN_IFC	0008_H	W	CAN 中断标志清零寄存器
CAN_TXSTAT	$000C_H$	R/W	CAN 发送状态寄存器
CAN_TXSTATC	0010_H	W	CAN 发送状态清零寄存器
CAN_RXF0	0014_H	R/W	CAN 接收 FIFO0 寄存器
CAN_RXF0C	0018_H	W	CAN 接收 FIFO0 状态清零寄存器
CAN_RXF1	$001C_H$	R/W	CAN 接收 FIFO1 寄存器
CAN_RXF1C	0020_H	R	CAN 接收 FIFO1 状态清零寄存器
CAN_IE	0024_H	R/W	CAN 中断使能寄存器
CAN_ERRSTAT	0028_H	R/W	CAN 错误状态寄存器
CAN_BTIME	$002C_H$	R/W	CAN 位时序寄存器
CAN_TXID0	0180_H	R/W	CAN 发送邮箱标识符寄存器 0
CAN_TXFCON0	0184_H	R/W	CAN 发送邮箱帧控制寄存器 0
CAN_TXDL0	0188_H	R/W	CAN 发送邮箱资料低位寄存器 0
CAN_TXDH0	$018C_H$	R/W	CAN 发送邮箱资料高位寄存器 0
CAN_TXID1	0190_H	R/W	CAN 发送邮箱标识符寄存器 1
CAN_TXFCON1	0194_H	R/W	CAN 发送邮箱帧控制寄存器 1
CAN_TXDL1	0198_H	R/W	CAN 发送邮箱资料低位寄存器 1
CAN_TXDH1	$019C_H$	R/W	CAN 发送邮箱资料高位寄存器 1
CAN_TXID2	$01A0_H$	R/W	CAN 发送邮箱标识符寄存器 2
CAN_TXFCON2	$01A4_H$	R/W	CAN 发送邮箱帧控制寄存器 2
CAN_TXDL2	$01A8_H$	R/W	CAN 发送邮箱资料低位寄存器 2
CAN_TXDH2	$01AC_H$	R/W	CAN 发送邮箱资料高位寄存器 2
CAN_RXF0ID	$01B0_H$	R	CAN 接收 FIFO0 邮箱标识符寄存器
CAN_RXF0INF	$01B4_H$	R	CAN 接收 FIFO0 邮箱数据信息寄存器
CAN_RXF0DL	$01B8_H$	R	CAN 接收 FIFO0 邮箱数据低位寄存器
CAN_RXF0DH	$01BC_H$	R	CAN 接收 FIFO0 邮箱数据高位寄存器
CAN_RXF1ID	$01C0_H$	R	CAN 接收 FIFO1 邮箱标识符寄存器
CAN_RXF1INF	$01C4_H$	R	CAN 接收 FIFO1 邮箱数据信息寄存器
CAN_RXF1DL	$01C8_H$	R	CAN 接收 FIFO1 邮箱数据低位寄存器
CAN_RXF1DH	$01CC_H$	R	CAN 接收 FIFO1 邮箱数据高位寄存器

CAN_FLTCON	0200_H	R/W	CAN 筛选器控制寄存器
CAN_FLTM	0204_H	R/W	CAN 筛选器模式寄存器
CAN_FLTWS	$020C_H$	R/W	CAN 筛选器宽度选择寄存器
CAN_FLTAS	0214_H	R/W	CAN 筛选器 FIFO 分配寄存器
CAN_FLTGO	$021C_H$	R/W	CAN 筛选器启动寄存器
CAN_FLT0R1	0240_H	R/W	筛选器组 0 寄存器 1
CAN_FLT0R2	0244_H	R/W	筛选器组 0 寄存器 2
CAN_FLT1R1	0248_H	R/W	筛选器组 1 寄存器 1
CAN_FLT1R2	$024C_H$	R/W	筛选器组 1 寄存器 2
CAN_FLT2R1	0250_H	R/W	筛选器组 2 寄存器 1
CAN_FLT2R2	0254_H	R/W	筛选器组 2 寄存器 2
CAN_FLT3R1	0258_H	R/W	筛选器组 3 寄存器 1
CAN_FLT3R2	$025C_H$	R/W	筛选器组 3 寄存器 2
CAN_FLT4R1	0260_H	R/W	筛选器组 4 寄存器 1
CAN_FLT4R2	0264_H	R/W	筛选器组 4 寄存器 2
CAN_FLT5R1	0268_H	R/W	筛选器组 5 寄存器 1
CAN_FLT5R2	$026C_H$	R/W	筛选器组 5 寄存器 2
CAN_FLT6R1	0270_H	R/W	筛选器组 6 寄存器 1
CAN_FLT6R2	0274_H	R/W	筛选器组 6 寄存器 2
CAN_FLT7R1	0278_H	R/W	筛选器组 7 寄存器 1
CAN_FLT7R2	$027C_H$	R/W	筛选器组 7 寄存器 2
CAN_FLT8R1	0280_H	R/W	筛选器组 8 寄存器 1
CAN_FLT8R2	0284_H	R/W	筛选器组 8 寄存器 2
CAN_FLT9R1	0288_H	R/W	筛选器组 9 寄存器 1
CAN_FLT9R2	$028C_H$	R/W	筛选器组 9 寄存器 2
CAN_FLT10R1	0290_H	R/W	筛选器组 10 寄存器 1
CAN_FLT10R2	0294_H	R/W	筛选器组 10 寄存器 2
CAN_FLT11R1	0298_H	R/W	筛选器组 11 寄存器 1
CAN_FLT11R2	$029C_H$	R/W	筛选器组 11 寄存器 2
CAN_FLT12R1	$02A0_H$	R/W	筛选器组 12 寄存器 1
CAN_FLT12R2	$02A4_H$	R/W	筛选器组 12 寄存器 2
CAN_FLT13R1	$02A8_H$	R/W	筛选器组 13 寄存器 1
CAN_FLT13R2	$02AC_H$	R/W	筛选器组 13 寄存器 2

16. 5. 2 寄存器描述

16. 5. 2. 1 CAN控制寄存器(CAN_CON)

CAN 控制寄存器 (CAN_CON)																													
偏移地址:0x00																													
复位值:0x0001 0002																													
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
															DBGSTP	RST								TTCEN	ABOFFEN	AWKEN	ARTXDIS	RXFOPM	TXMP	SLPREQ	INIREQ

—	Bit 31-20	—	—
DBGSTP	Bit 16	R/W	**调试停止** 0：调试期间CAN处于工作状态。 1：调试期间CAN处于停止状态。接收FIFO仍可正常访问/控制。
RST	Bit 15	R/W	**BxCAN软件复位** 0x0：正常工作。 0x1：BxCAN进行软件复位，复位后启动睡眠模式。此位自动复位为0。
—	Bit 14-8	—	—
TTCEN	Bit 7	R/W	**时间触发通信使能** 0x0：禁止时间触发通信模式。 0x1：使能时间触发通信模式。 注：有关时间触发通信模式的更多信息，请参见14.4.7.5。
ABOFFEN	Bit 6	R/W	**自动退出总线关闭使能** 此位控制CAN硬件在退出总线关闭状态时的行为。 0x0：一旦监测到128次连续11个隐性位，并且软件将CAN_CON.INIREQ位先置1再清零，退出总线关闭状态。 0x1：一旦监测到128次连续11个隐性位，即通过硬件自动退出总线关闭状态。
AWKEN	Bit 5	R/W	**自动唤醒使能** 此位控制CAN硬件在睡眠模式下接收到消息时的行为。 0x0：在软件通过将CAN_CON.SLPREQ位清

			零发出请求后，退出睡眠模式。 0x1：一旦监测到CAN消息，即通过硬件自动退出睡眠模式。CAN_CON.SLPREQ位和CAN_STAT.SLPSTAT位由硬件清零。
ARTXDIS	Bit 4	R/W	**自动重发禁止** 0x0：CAN硬件将自动重发送消息，直到消息发送成功。 0x1：无论发送结果如何(成功、错误或仲裁丢失)，消息均只发送一次。
RXFOPM	Bit 3	R/W	**接收FIFO溢出处理模式** 0x0：接收FIFO装满后，下一条传入消息将覆盖前一条消息。 0x1：接收FIFO装满后，下一条传入消息将被丢弃。
TXMP	Bit 2	R/W	**发送邮箱优先级** 此位用于控制在几个邮箱同时挂起时的发送顺序。 0x0：优先级由消息标识符确定。 0x1：优先级由请求顺序(时间顺序)确定。
SLPREQ	Bit 1	R/W	**睡眠请求** 此位由软件置1，用于请求CAN硬件进入睡眠模式。一旦当前CAN活动(发送或接收CAN帧)结束，即进入睡眠模式。 此位由软件清0时，将退出睡眠模式。 当CAN_CON.AWKEN位置1以及在CAN RX信号上检测到帧起始符(SOF)位时，硬件即将此位清零。 复位后，此位被置1，CAN启动睡眠模式。
INIREQ	Bit 0	R/W	**初始化请求** 软件通过将此位清零将硬件切换到正常模式。CAN 可由总线启动，一旦在总线信号上监测到连续 11 个隐性位 CAN 硬件即完成同步并准备进行发送和接收。 软件通过将此位置 1 来请求 CAN 硬件进入初始化模式。一旦当前 CAN 活动(发送或接收)结束，即进入初始化模式。 硬件通过将 CAN_STAT.INISTAT 位置 1 指示此事件。

16. 5. 2. 2 CAN状态寄存器 (CAN_STAT)

CAN 状态寄存器 (CAN_STAT)																														
偏移地址:0x04																														
复位值:0x0000 0C02																														
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
																				RX	PRESMP	RXSTAT	TXSTAT				SLPIF	WKIF	ERRIF	SLPSTAT	INISTAT

—	Bit 31-12	—	—
RX	Bit 11	R	**CAN Rx信号** 监视CAN_RX引脚的实际值。
PRESMP	Bit 10	R	**前采样点** 上一个采样点的RX值(当前接收的位值)。
RXSTAT	Bit 9	R	**接收状态** CAN硬件当前为接收器。
TXSTAT	Bit 8	R	**发送状态** CAN硬件当前为发送器。
—	Bit 7-5	—	—
SLPIF	Bit 4	R	**睡眠中断标志** 当CAN进入睡眠工作模式时，该位标志被置1。如果CAN_IE.SLPIE=1时，则当此位置1后将产生中断。此位通过CAN_IFC.SLPIFC=1清零。
WKIF	Bit 3	R	**唤醒中断标志** 此位由硬件置1，用于指示在CAN硬件处于睡眠模式期间检测到一个帧起始(SOF)位。 如果CAN_IE.WKIE=1时，则当此位置1后将产生中断。此位通过CAN_IFC.WKIFC=1清零。
ERRIF	Bit 2	R	**错误中断标志** 当错误中断使能CAN_IE.ERRIE=1时，并且CAN_ERRSTAT中某一位被置1，该位被置1，此位通过CAN_IFC.ERRIFC=1清零。
SLPSTAT	Bit 1	R	**睡眠状态** 此位由硬件置1，用于向软件指示CAN此时处于睡眠模式。此位可确认软件的睡眠模式请求(CAN_CON.SLPREQ=1)。

			当CAN退出睡眠模式 (CAN_CON.SLPREQ=0)时此位由硬件清零。
INISTAT	Bit 0	R	**初始化状态** 此位由硬件置 1，用于向软件指示 CAN 硬件此时处于初始化模式。此位可确认软件的初始化请求(CAN_CON.INIREQ=1)。 当 CAN 退出初始化模式 (CAN_CON.INIREQ=0)时，此位由硬件清零。

16. 5. 2. 3 CAN中断标志清零寄存器(CAN_IFC)

CAN 中断标志清零寄存器 (CAN_IFC)

偏移地址:0x08

复位值:0x0000 0000

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

—	Bit 31-5	—	—
SLPIFC	Bit 4	W1	**睡眠中断标志清零** 0x0：无操作。 0x1：睡眠确认中断标志清零。
WKIFC	Bit 3	W1	**唤醒中断标志清零** 0x0：无操作。 0x1：唤醒中断标志清零。
ERRIFC	Bit 2	W1	**错误中断标志清零** 0x0：无操作。 0x1：错误中断标志清零。
—	Bit 1-0	—	—

16. 5. 2. 4 CAN发送状态寄存器(CAN_TXSTAT)

CAN 发送状态寄存器 (CAN_TXSTAT)																												
偏移地址:0x0C																												
复位值:0x1C00 0000																												
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
TXM2LPF	TXM1LPF	TXM0LPF	TXM2EF	TXM1EF	TXM0EF	CODE<1:0>		M2STPREQ	—	—	—	M2TXERR	M2ARBLST	M2TXC	M2REQC	M1STPREQ	—	—	—	M1TXERR	M1ARBLST	M1TXC	M1REQC	M0STPREQ	—	—	—	M0TXERR	M0ARBLST	M0TXC	M0REQC

TXM2LPF	Bit 31	R	**发送邮箱2最低优先级标志** 当多个邮箱处于挂起态且邮箱2优先级最低时，此位由硬件置1。
TXM1LPF	Bit 30	R	**发送邮箱1最低优先级标志** 当多个邮箱处于挂起态且邮箱1优先级最低时，此位由硬件置1。
TXM0LPF	Bit 29	R	**发送邮箱0最低优先级标志** 当多个邮箱处于挂起态且邮箱0优先级最低时，此位由硬件置1。
TXM2EF	Bit 28	R	**发送邮箱2空标志** 当邮箱2没有挂起的发送请求时，此位由硬件置1。
TXM1EF	Bit 27	R	**发送邮箱1空标志** 当邮箱1没有挂起的发送请求时，此位由硬件置1。
TXM0EF	Bit 26	R	**发送邮箱0空标志** 当邮箱0没有挂起的发送请求时，此位由硬件置1。
CODE	Bit 25-24	R	**邮箱代码** 如果至少一个发送邮箱空闲，代码值等于下一个空闲发送邮箱的编号。 如果所有发送邮箱均挂起，则代码值等于优先级最低的发送邮箱的编号。
M2STPREQ	Bit 23	R/W	**邮箱2停止请求** 由软件置1，用于中止相应邮箱的发送请求。 邮箱变为空后，此位由硬件清零。 未处于挂起态的邮箱，将此位置1没有任何作用。

—	Bit 22-20	—	—
M2TXERR	Bit 19	R	**邮箱2发送错误** 如果上一次发送因错误而失败，此位将置1。 该位通过CAN_TXSTATC.M2TXERR=1清零。
M2ARBLST	Bit 18	R	**邮箱2仲裁丢失** 如果上一次发送因仲裁丢失而失败，此位将置1。 该位通过CAN_TXSTATC.M2ARBLST=1清零。
M2TXC	Bit 17	R	**邮箱2发送完成** 每次发送尝试后，硬件都将更新此位。 0x0：上一次发送失败。 0x1：上一次发送成功。 该位通过CAN_TXSTATC.M2TXC=1清零。
M2REQC	Bit 16	R	**邮箱2请求完成** 最后一个请求(发送或中止)执行完毕时，由硬件置1。 该位通过CAN_TXSTATC.M2REQC=1清零。 当产生发送请求(CAN_TXID2.TXMREQ=1)时由硬件清零。 如果将此位清零，邮箱2的所有状态位(M2TXC、M2ARBLST、M2TXERR)都将清零。
M1STPREQ	Bit 15	R/W	**邮箱1停止请求** 由软件置1，用于中止相应邮箱的发送请求。 邮箱变为空后，此位由硬件清零。 未处于挂起态的邮箱，将此位置1没有任何作用。
—	Bit 14-12	—	—
M1TXERR	Bit 11	R	**邮箱1发送错误** 如果上一次发送因错误而失败，此位将置1。 该位通过CAN_TXSTATC.M1TXERR=1清零。
M1ARBLST	Bit 10	R	**邮箱1仲裁丢失** 如果上一次发送因仲裁丢失而失败，此位将置1。 该位通过CAN_TXSTATC.M1ARBLST=1清零。
M1TXC	Bit 9	R	**邮箱1发送完成** 每次发送尝试后，硬件都将更新此位。

			0x0：上一次发送失败。 0x1：上一次发送成功。 该位通过CAN_TXSTATC.M1TXC=1清零。
M1REQC	Bit 8	R	**邮箱1请求完成** 最后一个请求(发送或中止)执行完毕时，由硬件置1。 该位通过CAN_TXSTATC.M1REQC=1清零。 当产生发送请求(CAN_TXID1.TXMREQ=1)时由硬件清零。 如果将此位清零，邮箱1的所有状态位(M1TXC、M1ARBLST、M1TXERR)都将清零。
M0STPREQ	Bit 7	R/W	**邮箱0停止请求** 由软件置1，用于中止相应邮箱的发送请求。 邮箱变为空后，此位由硬件清零。 未处于挂起态的邮箱，将此位置1没有任何作用。
—	Bit 6-4	—	—
M0TXERR	Bit 3	R	**邮箱0发送错误** 如果上一次发送因错误而失败，此位将置1。 该位通过CAN_TXSTATC.M0TXERR=1清零。
M0ARBLST	Bit 2	R	**邮箱0仲裁丢失** 如果上一次发送因仲裁丢失而失败，此位将置1。 该位通过CAN_TXSTATC.M0ARBLST=1清零。
M0TXC	Bit 1	R	**邮箱0发送完成** 每次发送尝试后，硬件都将更新此位。 0x0：上一次发送失败 0x1：上一次发送成功 该位通过CAN_TXSTATC.M0TXC=1清零。
M0REQC	Bit 0	R	**邮箱0请求完成** 最后一个请求(发送或中止)执行完毕时，由硬件置1。 该位通过CAN_TXSTATC.M0REQC=1清零。 当产生发送请求(CAN_TXID0.TXMREQ=1)时由硬件清零。 如果将此位清零，邮箱0的所有状态位(M0TXC、M0ARBLST、M0TXERR)都将清零。

16. 5. 2. 5 CAN发送状态清零寄存器(CAN_TXSTATC)

CAN 发送状态清零寄存器 (CAN_TXSTATC)																													
偏移地址:0x10																													
复位值:0x0000 0000																													
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
												M2TXERR	M2ARBLST	M2TXC	M2REQC					M1TXERR	M1ARBLST	M1TXC	M1REQC					M0TXERR	M0ARBLST	M0TXC	M0REQC

—	Bit 31-20	—	—
M2TXERR	Bit 19	W1	**邮箱2发送错误标志清零** 0x0：无操作。 0x1：邮箱2发送错误标志清零。
M2ARBLST	Bit 18	W1	**邮箱2仲裁丢失标志清零** 0x0：无操作。 0x1：邮箱2仲裁丢失标志清零。
M2TXC	Bit 17	W1	**邮箱2发送成功标志清零** 0x0：无操作。 0x1：邮箱2发送成功标志清零。
M2REQC	Bit 16	W1	**邮箱2请求完成标志清零** 0x0：无操作。 0x1：邮箱2请求完成标志清零。
—	Bit 15-12	—	—
M1TXERR	Bit 11	W1	**邮箱1发送错误标志清零** 0x0：无操作。 0x1：邮箱1发送错误标志清零。
M1ARBLST	Bit 10	W1	**邮箱1仲裁丢失标志清零** 0x0：无操作。 0x1：邮箱1仲裁丢失标志清零。
M1TXC	Bit 9	W1	**邮箱1发送成功标志清零** 0x0：无操作。 0x1：邮箱1发送成功标志清零。
M1REQC	Bit 8	W1	**邮箱1请求完成标志清零** 0x0：无操作。 0x1：邮箱1请求完成标志清零。
—	Bit 15-12	—	—
M0TXERR	Bit 3	W1	**邮箱0发送错误标志清零**

			0x0：无操作。 0x1：邮箱0发送错误标志清零。
M0ARBLST	Bit 2	W1	**邮箱0仲裁丢失标志清零** 0x0：无操作。 0x1：邮箱0仲裁丢失标志清零。
M0TXC	Bit 1	W1	**邮箱0发送成功标志清零** 0x0：无操作。 0x1：邮箱0发送成功标志清零。
M0REQC	Bit 0	W1	**邮箱0请求完成标志清零** 0x0：无操作。 0x1：邮箱0请求完成标志清零。

16. 5. 2. 6 CAN接收FIFO0 寄存器(CAN_RXF0)

CAN 接收 FIFO0 寄存器(CAN_RXF0)

偏移地址:0x14

复位值:0x0000 0000

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

—	Bit 31-6	—	—
FREE	Bit 5	R/W	**释放FIFO0接收邮箱** 由软件置1，用于释放FIFO0的接收邮箱。FIFO0至少有一条消息挂起时，才能释放输出邮箱。FIFO为空时，将此位置1没有任何作用。如果FIFO中至少有两个邮箱，软件必须释放当前接收邮箱，才能访问下一个接收邮箱。接收邮箱全部释放后，此位由硬件清零。
OVR	Bit 4	R	**FIFO0上溢** FIFO三级深度邮箱填满时，如果接收到新消息并且通过筛选器，此位将由硬件置1。 此位通过CAN_RXF0C.OVRC=1清零。
FULL	Bit 3	R	**FIFO0满** FIFO三级深度邮箱填满时，由硬件置1。 此位通过CAN_RXF0C.FULLC=1清零。
—	Bit 2	—	—

PEND	Bit 1-0	R	**FIFO0挂起接收邮箱数** 这些位用于指示接收FIFO中挂起接收邮箱数。 硬件每向FIFO存储一条新消息，CAN_RXF0.PEND即会增加。 软件每次通过将CAN_RXF0.FREE=1来释放接收邮箱，CAN_RXF0.PEND即会减小。

16. 5. 2. 7 CAN接收FIFO0 状态清零寄存器(CAN_RXF0C)

CAN 接收 FIFO0 状态清零寄存器 (CAN_RXF0C)

偏移地址:0x18

复位值:0x0000 0000

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

—	Bit 31-5	—	—
OVRC	Bit 4	W1	**FIFO0上溢标志清零** 0x0：无操作。 0x1：FIFO0上溢标志清零。
FULLC	Bit 3	W1	**FIFO0满标志清零** 0x0：无操作。 0x1：FIFO0满标志清零。
—	Bit 2-0	—	—

16. 5. 2. 8 CAN接收FIFO1 寄存器(CAN_RXF1)

CAN 接收 FIFO1 寄存器 (CAN_RXF1)																													
偏移地址:0x1C																													
复位值:0x0000 0000																													
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																										FREE	OVR	FULL		PEND<1:0>	

—	Bit 31-6	—	—
FREE	Bit 5	R/W	**释放FIFO1接收邮箱** 由软件置1，用于释放FIFO1的接收邮箱。 FIFO1至少有一条消息挂起时，才能释放输出邮箱。FIFO为空时，将此位置1没有任何作用。 如果FIFO中至少有两个邮箱，软件必须释放当前接收邮箱，才能访问下一个接收邮箱。接收邮箱全部释放后，此位由硬件清零。
OVR	Bit 4	R	**FIFO1上溢** FIFO三级深度邮箱填满时，如果接收到新消息并且通过筛选器，此位将由硬件置1。 此位通过CAN_RXF1C.OVRC=1清零。
FULL	Bit 3	R	**FIFO1满** FIFO三级深度邮箱填满时，由硬件置1。 此位通过CAN_RXF1C.FULLC=1清零。
—	Bit 2	—	—
PEND	Bit 1-0	R	**FIFO1挂起接收邮箱数** 这些位用于指示接收FIFO中挂起接收邮箱数。 硬件每向FIFO存储一条新消息，CAN_RXF1.PEND即会增加。 软件每次通过将CAN_RXF1.FREE=1来释放接收邮箱，CAN_RXF1.PEND即会减小。

16. 5. 2. 9 CAN接收FIFO1 状态清零寄存器(CAN_RXF1C)

CAN 接收 FIFO1 状态清零寄存器 (CAN_RXF1C)																														
偏移地址:0x20																														
复位值:0x0000 0000																														
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																											OVRC	FULLC			

—	Bit 31-6	—	—
OVRC	Bit 4	W1	**FIFO1上溢标志清零** 0x0：无操作。 0x1：FIFO1上溢标志清零。
FULLC	Bit 3	W1	**FIFO 0满标志清零** 0x0：无操作。 0x1：FIFO1满标志清零。
—	Bit 2-0	—	—

16. 5. 2. 10 CAN中断使能寄存器(CAN_IE)

CAN 中断使能寄存器 (CAN_IE)																														
偏移地址:0x24																														
复位值:0x0000 0000																														
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
														SLPIE	WKIE	ERRIE				PRERRIE	BOFFIE	PERRIE	WARNIE		F1OVRIE	F1FULIE	F1PIE	F0OVRIE	F0FULIE	F0PIE	TXMEIE

—	Bit 31-18	—	—
SLPIE	Bit 17	R/W	**睡眠中断使能** 0x0：当CAN_STAT.SLPIF=1时，不产生中断。 0x1：当CAN_STAT.SLPIF=1时，产生中断。
WKIE	Bit 16	R/W	**唤醒中断使能** 0x0：当CAN_STAT.WKIF=1时，不产生中断。 0x1：当CAN_STAT.WKIF=1时，产生中断。
ERRIE	Bit 15	R/W	**错误中断使能** 0x0：当CAN_STAT.ERRIF=1时，不会产生中

			断。 0x1：当CAN_STAT.ERRIF=1时，会产生中断。
—	Bit 14-12	—	—
PRERRIE	Bit 11	R/W	**上一个错误代码中断使能** 0x0：当CAN_ERRSTAT.PRERRF非0时，CAN_STAT.ERRIF不会置1。 0x1：当CAN_ERRSTAT.PRERRF非0时，CAN_STAT.ERRIF会置1。
BOFFIE	Bit 10	R/W	**总线关闭中断使能** 0x0：当CAN_ERRSTAT.BOFF=1时，CAN_STAT.ERRIF不会置1。 0x1：当CAN_ERRSTAT.BOFF=1时，CAN_STAT.ERRIF会置1。
PERRIE	Bit 9	R/W	**错误被动中断使能** 0x0：当CAN_ERRSTAT.PERRF=1时，CAN_STAT.ERRIF不会置1。 0x1：当CAN_ERRSTAT.PERRF=1时，CAN_STAT.ERRIF会置1。
WARNIE	Bit 8	R/W	**警告中断使能** 0x0：当CAN_ERRSTAT.WARNF=1时，CAN_STAT.ERRIF不会置1。 0x1：当CAN_ERRSTAT.WARNF=1时，CAN_STAT.ERRIF会置1。
—	Bit 7	—	—
F1OVRIE	Bit 6	R/W	**FIFO1上溢中断使能** 0x0：当CAN_RXF1.OVR=1时，不产生中断。 0x1：当CAN_RXF1.OVR=1时，产生中断。
F1FULIE	Bit 5	R/W	**FIFO1满中断使能** 0x0：当CAN_RXF1.FULL=1时，不产生中断。 0x1：当CAN_RXF1.FULL=1时，产生中断。
F1PIE	Bit 4	R/W	**FIFO1消息挂起中断使能** 0x0：当CAN_RXF1.PEND非0时，不产生中断。 0x1：当CAN_RXF1.PEND非0时，产生中断。
F0OVRIE	Bit 3	R/W	**FIFO0上溢中断使能** 0x0：当CAN_RXF0.OVR=1时，不产生中断。 0x1：当CAN_RXF0.OVR=1时，产生中断。
F0FULIE	Bit 2	R/W	**FIFO0满中断使能**

			0x0：当CAN_RXF0.FULL=1时，不产生中断。 0x1：当CAN_RXF0.FULL=1时，产生中断。
F0PIE	Bit 1	R/W	**FIFO0消息挂起中断使能** 0x0：当CAN_RXF0.PEND非0时，不产生中断。 0x1：当CAN_RXF0.PEND非0时，产生中断。
TXMEIE	Bit 0	R/W	**发送邮箱空中断使能** 0x0：CAN_TXSTAT.MxREQC=1时，不产生中断。 0x1：CAN_TXSTAT.MxREQC=1时，产生中断。

16. 5. 2. 11 CAN错误状态寄存器(CAN_ERRSTAT)

CAN 错误状态寄存器 (CAN_ERRSTAT)

偏移地址:0x28

复位值:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXERRC<7:0>								TXERRC<7:0>								—	—	—	—	—	—	—	—	—	PRERRF<2:0>			—	BOFF	PERRF	WARNF

RXERRC	Bit 31-24	R	**8位接收错误计数器** CAN协议故障隔离机制的实施部分。如果接收期间发生错误，该计数器按1或8递增，具体取决于CAN标准所定义的错误状况。 每次成功接收后，该计数器按1递减，如果其数值大于128，则复位为120。计数器值超过127时，CAN控制器进入错误被动状态。
TXERRC	Bit 23-16	R	**9位发送错误计数器的低八位** 同上类似。
—	Bit 15-7	—	—
PRERRF	Bit 6-4	R/W	**上一个错误代码** 该字段由硬件置1，其中的代码指示CAN总线上检测到的上一个错误的错误状况。如果消息成功传送(接收或发送)且未发生错误，该字段将清为“0”。 LEC[2:0]位可由软件置为0b111值。这些位由

			硬件更新，以指示当前通信状态。 000：无错误。 001：填充错误。 010：格式错误。 011：确认错误。 100：位隐性错误。 101：位显性错误。 110：CRC错误。 111：由软件置位。
—	Bit 3	—	—
BOFF	Bit 2	W	**总线关闭标志** 此位由硬件置1。当CAN_ERRSTAT.TXERRC上溢(超过255)时，进入总线关闭状态。
PERRF	Bit 1	R	**错误被动标志** 达到错误被动状态(接收错误计数器或发送错误计数器>127)时，此位由硬件置1。
WARNF	Bit 0	R	**警告错误标志** 达到错误警告极限时，此位由硬件置1(接收错误计数器或发送错误计数器≥96)。

16. 5. 2. 12 CAN位时序寄存器(CAN_BTIME)

CAN 位时序寄存器 (CAN_BTIME)																												
偏移地址:0x2C																												
复位值:0x0123 0000																												
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
SILENT	LOOP					RESJW<1:0>			SEG2<2:0>			SEG1<3:0>										BPSC<9:0>								

SILENT	Bit 31	R/W	**静默模式(调试)** 0x0：正常工作。 0x1：静默模式。
LOOP	Bit 30	R/W	**回环模式(调试)** 0x0：禁止环回模式。 0x1：使能环回模式。
—	Bit 29-26	—	—
RESJW	Bit 25-24	R/W	**再同步跳转宽度** 这些位定义CAN硬件在执行再同步时最多可以将位加长或缩短的时间片数目。 $t_{RESJW} = t_{CAN} \times (RESJW+1)$
—	Bit 23	—	—
SEG2	Bit 22-20	R/W	**时间段2** 这些位定义时间段2中的时间片数目。 $t_{SEG2} = t_{CAN} \times (SEG2 + 1)$
SEG1	Bit 19-16	R/W	**时间段1** 这些位定义时间段1中的时间片数目。 $t_{SEG1} = t_{CAN} \times (SEG1 + 1)$
—	Bit 15-10	—	—
BPSC	Bit 9-0	R/W	**波特率预分频器** 这些位定义一个时间片的长度。 $t_{CAN} = (BPSC[8:0] + 1) \times t_{PCLK}$

16. 5. 2. 13 CAN发送邮箱标识符寄存器 0 (CAN_TXID0)

CAN 发送邮箱标识符寄存器 0 (CAN_TXID0)																														
偏移地址:0x180																														
复位值:0xXXXX XXXX																														
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
STDID[10:0] EXID[28:18]											EXID[17:0]																	IDE	RTR	TXMREQ

STDID[10:0] EXID[28:18]	Bit 31-21	R/W	**标准标识符或扩展标识符** 标准标识符或扩展标识符的MSB(取决于IDE位的值)。
EXID[17:0]	Bit 20-3	R/W	**扩展标识符** 扩展标识符的LSB。
IDE	Bit 2	R/W	**标识符扩展** 此位用于定义邮箱中消息的标识符类型。 0x0：标准标识符。 0x1：扩展标识符。
RTR	Bit 1	R/W	**帧类型** 0x0：数据帧。 0x1：远程帧。
TXMREQ	Bit 0	R/W	**发送邮箱请求** 此位初始数值为0，由软件置1，用于请求发送相应邮箱的内容。邮箱变为空后，此位由硬件清零。

16. 5. 2. 14 CAN发送邮箱帧控制寄存器 0 (CAN_TXFCON0)

CAN 发送邮箱帧控制寄存器 0 (CAN_TXFCON0)																												
偏移地址:0x184																												
复位值:0xXXXX XXXX																												
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
STAMP<15:0>																—	—	—	—	—	—	—	TXGT	—	—	—	—	DLEN<3:0>			

STAMP	Bit 31-16	R/W	**消息时间戳** 此字段包含在进行帧起始(SOF)发送时所捕获的16位定时器值。
—	Bit 15-9	—	—
TXGT	Bit 8	R/W	**发送全局时间** 只有硬件处于时间触发通信模式(CAN_CON.TTCEN=1)时，此位才会启动。 0：不发送时间戳STAMP。 1：在8字节消息的最后两个数据字节中发送时间戳STAMP的值。数据字节6写入STAMP[7:0]，数据字节7写入STAMP[15:8]。且DLEN必须程序设置为8，才能通过CAN总线发送这两个字节。
—	Bit 7-4	—	—
DLEN	Bit 3-0	R/W	**数据长度** 该字段定义数据帧或远程帧请求中的数据字节数。 一条消息可以包含0到8个数据字节。

16. 5. 2. 15 CAN发送邮箱资料低位寄存器 0 (CAN_TXDL0)

CAN 发送邮箱资料低位寄存器 0 (CAN_TXDL0)																															
偏移地址:0x188																															
复位值:0xXXXX XXXX																															
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
BYTE3<7:0>								BYTE2<7:0>								BYTE1<7:0>								BYTE0<7:0>							

BYTE3	Bit 31-24	R/W	**数据字节3** 消息的数据字节3。
BYTE2	Bit 23-16	R/W	**数据字节2** 消息的数据字节2。
BYTE1	Bit 15-8	R/W	**数据字节1** 消息的数据字节1。
BYTE0	Bit 7-0	R/W	**数据字节0** 消息的数据字节0。

注：当邮箱未处于空状态时，该寄存器的所有位均为写保护状态。

16. 5. 2. 16 CAN发送邮箱资料高位寄存器 0 (CAN_TXDH0)

CAN 发送邮箱资料高位寄存器 0 (CAN_TXDH0)																															
偏移地址:0x18C																															
复位值:0xXXXX XXXX																															
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
BYTE7<7:0>								BYTE6<7:0>								BYTE5<7:0>								BYTE4<7:0>							

BYTE7	Bit 31-24	R/W	**数据字节7** 消息的数据字节7。
BYTE6	Bit 23-16	R/W	**数据字节6** 消息的数据字节6。
BYTE5	Bit 15-8	R/W	**数据字节5** 消息的数据字节5。
BYTE4	Bit 7-0	R/W	**数据字节4** 消息的数据字节4。

注：当邮箱未处于空状态时，该寄存器的所有位均为写保护状态。

16. 5. 2. 17 CAN发送邮箱标识符寄存器 1 (CAN_TXID1)

CAN 发送邮箱标识符寄存器 1 (CAN_TXID1)																														
偏移地址:0x190																														
复位值:0xXXXX XXXX																														
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
STDID[10:0] EXID[28:18]											EXID[17:0]																	IDE	RTR	TXMREQ

STDID[10:0] EXID[28:18]	Bit 31-21	R/W	**标准标识符或扩展标识符** 标准标识符或扩展标识符的MSB(取决于IDE位的值)。
EXID[17:0]	Bit 20-3	R/W	**扩展标识符** 扩展标识符的LSB。
IDE	Bit 2	R/W	**标识符扩展** 此位用于定义邮箱中消息的标识符类型。 0x0：标准标识符。 0x1：扩展标识符。
RTR	Bit 1	R/W	**帧类型** 0x0：数据帧。 0x1：远程帧。
TXMREQ	Bit 0	R/W	**发送邮箱请求** 此位初始数值为0，由软件置1，用于请求发送相应邮箱的内容。邮箱变为空后，此位由硬件清零。

16. 5. 2. 18 CAN发送邮箱帧控制寄存器 1 (CAN_TXFCON1)

CAN 发送邮箱帧控制寄存器 1 (CAN_TXFCON1)																												
偏移地址:0x194																												
复位值:0xXXXX XXXX																												
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
STAMP<15:0>																							TXGT					DLEN<3:0>			

STAMP	Bit 31-16	R/W	**消息时间戳** 此字段包含在进行帧起始(SOF)发送时所捕获的16位定时器值。
—	Bit 15-9	—	—
TXGT	Bit 8	R/W	**发送全局时间** 只有硬件处于时间触发通信模式(CAN_CON.TTCEN=1)时，此位才会启动。 0：不发送时间戳STAMP。 1：在8字节消息的最后两个数据字节中发送时间戳STAMP的值。数据字节6写入STAMP[7:0]，数据字节7写入STAMP[15:8]。且DLEN必须程序设置为8，才能通过CAN总线发送这两个字节。
—	Bit 7-4	—	—
DLEN	Bit 3-0	R/W	**数据长度代码** 该字段定义数据帧或远程帧请求中的数据字节数。 一条消息可以包含0到8个数据字节。

16. 5. 2. 19 CAN发送邮箱资料低位寄存器 1 (CAN_TXDL1)

CAN 发送邮箱资料低位寄存器 1 (CAN_TXDL1)																															
偏移地址:0x198																															
复位值:0xXXXX XXXX																															
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
BYTE3<7:0>								BYTE2<7:0>								BYTE1<7:0>								BYTE0<7:0>							

BYTE3	Bit 31-24	R/W	**数据字节3** 消息的数据字节3。
BYTE2	Bit 23-16	R/W	**数据字节2** 消息的数据字节2。
BYTE1	Bit 15-8	R/W	**数据字节1** 消息的数据字节1。
BYTE0	Bit 7-0	R/W	**数据字节0** 消息的数据字节0。

注：当邮箱未处于空状态时，该寄存器的所有位均为写保护状态。

16. 5. 2. 20 CAN发送邮箱资料高位寄存器 1 (CAN_TXDH1)

CAN 发送邮箱资料高位寄存器 1 (CAN_TXDH1)																															
偏移地址:0x19C																															
复位值:0xXXXX XXXX																															
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
BYTE7<7:0>								BYTE6<7:0>								BYTE5<7:0>								BYTE4<7:0>							

BYTE7	Bit 31-24	R/W	**数据字节7** 消息的数据字节7。
BYTE6	Bit 23-16	R/W	**数据字节6** 消息的数据字节6。
BYTE5	Bit 15-8	R/W	**数据字节5** 消息的数据字节5。
BYTE4	Bit 7-0	R/W	**数据字节4** 消息的数据字节4。

注：当邮箱未处于空状态时，该寄存器的所有位均为写保护状态。

16. 5. 2. 21 CAN发送邮箱标识符寄存器 2 (CAN_TXID2)

CAN 发送邮箱标识符寄存器 2 (CAN_TXID2)																														
偏移地址:0x1A0																														
复位值:0xXXXX XXXX																														
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
STDID<10:0> EXID<28:18>											EXID<17:0>																	IDE	RTR	TXMREQ

STDID<10:0> EXID<28:18>	Bit 31-21	R/W	**标准标识符或扩展标识符** 标准标识符或扩展标识符的MSB(取决于IDE位的值)。
EXID<17:0>	Bit 20-3	R/W	**扩展标识符** 扩展标识符的LSB。
IDE	Bit 2	R/W	**标识符扩展** 此位用于定义邮箱中消息的标识符类型。 0x0：标准标识符。 0x1：扩展标识符。
RTR	Bit 1	R/W	**帧类型** 0x0：数据帧。 0x1：远程帧。
TXMREQ	Bit 0	R/W	**发送邮箱请求** 此位初始数值为0，由软件置1，用于请求发送相应邮箱的内容。邮箱变为空后，此位由硬件清零。

16. 5. 2. 22 CAN发送邮箱帧控制寄存器 2 (CAN_TXFCON2)

CAN 发送邮箱帧控制寄存器 2 (CAN_TXFCON2)																												
偏移地址:0x1A4																												
复位值:0xXXXX XXXX																												
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
STAMP<15:0>																—	—	—	—	—	—	—	TXGT	—	—	—	—	DLEN<3:0>			

STAMP	Bit 31-16	R/W	**消息时间戳** 此字段包含在进行帧起始(SOF)发送时所捕获的16位定时器值。
—	Bit 15-9	—	—
TXGT	Bit 8	R/W	**发送全局时间** 只有硬件处于时间触发通信模式(CAN_CON.TTCEN=1)时，此位才会启动。 0：不发送时间戳STAMP。 1：在8字节消息的最后两个数据字节中发送时间戳STAMP的值。数据字节6写入STAMP[7:0]，数据字节7写入STAMP[15:8]。且DLEN必须程序设置为8，才能通过CAN总线发送这两个字节。
—	Bit 7-4	—	—
DLEN	Bit 3-0	R/W	**数据长度代码** 该字段定义数据帧或远程帧请求中的数据字节数。 一条消息可以包含0到8个数据字节。

16. 5. 2. 23 CAN发送邮箱资料低位寄存器 2 (CAN_TXDL2)

CAN 发送邮箱资料低位寄存器 2 (CAN_TXDL2)																															
偏移地址:0x1A8																															
复位值:0xXXXX XXXX																															
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
BYTE3<7:0>								BYTE2<7:0>								BYTE1<7:0>								BYTE0<7:0>							

BYTE3	Bit 31-24	R/W	**数据字节3** 消息的数据字节3。
BYTE2	Bit 23-16	R/W	**数据字节2** 消息的数据字节2。
BYTE1	Bit 15-8	R/W	**数据字节1** 消息的数据字节1。
BYTE0	Bit 7-0	R/W	**数据字节0** 消息的数据字节0。

注：当邮箱未处于空状态时，该寄存器的所有位均为写保护状态。

16. 5. 2. 24 CAN发送邮箱资料高位寄存器 2 (CAN_TXDH2)

CAN 发送邮箱资料高位寄存器 2 (CAN_TXDH2)																															
偏移地址:0x1AC																															
复位值:0xXXXX XXXX																															
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
BYTE7<7:0>								BYTE6<7:0>								BYTE5<7:0>								BYTE4<7:0>							

BYTE7	Bit 31-24	R/W	**数据字节7** 消息的数据字节7。
BYTE6	Bit 23-16	R/W	**数据字节6** 消息的数据字节6。
BYTE5	Bit 15-8	R/W	**数据字节5** 消息的数据字节5。
BYTE4	Bit 7-0	R/W	**数据字节4** 消息的数据字节4。

注：当邮箱未处于空状态时，该寄存器的所有位均为写保护状态。

16. 5. 2. 25 CAN接收FIFO0 邮箱标识符寄存器 (CAN_RXF0ID)

CAN 接收 FIFO0 邮箱标识符寄存器 (CAN_RXF0ID)																														
偏移地址:0x1B0																														
复位值:0xXXXX XXXX																														
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
STDID<10:0> EXID<28:18>											EXID<17:0>																	IDE	RTR	—

STDID<10:0> EXID<28:18>	Bit 31-21	R	**标准标识符或扩展标识符** 标准标识符或扩展标识符的MSB(取决于IDE位的值)。
EXID<17:0>	Bit 20-3	R	**扩展标识符** 扩展标识符的LSB。
IDE	Bit 2	R	**标识符扩展** 此位用于定义邮箱中消息的标识符类型。 0x0：标准标识符。 0x1：扩展标识符。
RTR	Bit 1	R	**帧类型** 0x0：数据帧。 0x1：远程帧。
—	Bit 0	—	—

注：所有接收寄存器均受到写保护。

16. 5. 2. 26 CAN接收FIFO0 邮箱数据信息寄存器(CAN_RXF0INF)

CAN 接收 FIFO0 邮箱数据信息寄存器 (CAN_RXF0INF)																													
偏移地址:0x1B4																													
复位值:0xXXXX XXXX																													
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
STAMP<15:0>																FLTIDX<7:0>								—	—	—	—	DLEN<3:0>			

STAMP	Bit 31-16	R	**消息时间戳** 此字段包含在接收帧起始(SOF)时所捕获的16位定时器值。
FLTIDX	Bit 15-8	R	**筛选器匹配索引** 该寄存器包含筛选器索引，邮箱中存储的消息需要经过筛选器。
—	Bit 7-4	—	—
DLEN	Bit 3-0	R	**数据长度代码** 该字段定义一个数据帧所包含的数据字节数(0到8)。如果是远程帧请求，则为0。

16. 5. 2. 27 CAN接收FIFO0 邮箱数据低位寄存器(CAN_RXF0DL)

CAN 接收 FIFO0 邮箱数据低位寄存器 (CAN_RXF0DL)																															
偏移地址:0x1B8																															
复位值:0xXXXX XXXX																															
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
BYTE3<7:0>								BYTE2<7:0>								BYTE1<7:0>								BYTE0<7:0>							

BYTE3	Bit 31-24	R	**数据字节3** 消息的数据字节3。
BYTE2	Bit 23-16	R	**数据字节2** 消息的数据字节2。
BYTE1	Bit 15-8	R	**数据字节1** 消息的数据字节1。
BYTE0	Bit 7-0	R	**数据字节0** 消息的数据字节0。

16. 5. 2. 28 CAN接收FIFO0 邮箱数据高位寄存器(CAN_RXF0DH)

CAN 接收 FIFO0 邮箱数据高位寄存器 (CAN_RXF0DH)																															
偏移地址:0x1BC																															
复位值:0xXXXX XXXX																															
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
BYTE7<7:0>								BYTE6<7:0>								BYTE5<7:0>								BYTE4<7:0>							

BYTE7	Bit 31-24	R	**数据字节7** 消息的数据字节7。
BYTE6	Bit 23-16	R	**数据字节6** 消息的数据字节6。
BYTE5	Bit 15-8	R	**数据字节5** 消息的数据字节5。
BYTE4	Bit 7-0	R	**数据字节4** 消息的数据字节4。

16. 5. 2. 29 CAN接收FIFO1 邮箱标识符寄存器(CAN_RXF1ID)

CAN 接收 FIFO1 邮箱标识符寄存器 (CAN_RXF1ID)																														
偏移地址:0x1C0																														
复位值:0xXXXX XXXX																														
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
STDID<10:0> EXID<28:18>											EXID<17:0>																	IDE	RTR	—

STDID<10:0> EXID<28:18>	Bit 31-21	R	**标准标识符或扩展标识符** 标准标识符或扩展标识符的MSB(取决于IDE位的值)。
EXID<17:0>	Bit 20-3	R	**扩展标识符** 扩展标识符的LSB。
IDE	Bit 2	R	**标识符扩展** 此位用于定义邮箱中消息的标识符类型。 0x0：标准标识符。 0x1：扩展标识符。
RTR	Bit 1	R	**帧类型** 0x0：数据帧。 0x1：远程帧。
—	Bit 0	—	—

注：所有接收寄存器均受到写保护。

16. 5. 2. 30 CAN接收FIFO1 邮箱数据信息寄存器(CAN_RXF1INF)

CAN 接收 FIFO1 邮箱数据信息寄存器 (CAN_RXF1INF)																													
偏移地址:0x1C4																													
复位值:0xXXXX XXXX																													
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
STAMP<15:0>																FLTIDX<7:0>								—	—	—	—	DLEN<3:0>			

STAMP	Bit 31-16	R	**消息时间戳** 此字段包含在接收帧起始(SOF)时所捕获的16位定时器值。
FLTIDX	Bit 15-8	R	**筛选器匹配索引** 该寄存器包含筛选器索引，邮箱中存储的消息需要经过筛选器。
—	Bit 7-4	—	—
DLEN	Bit 3-0	R	**数据长度代码** 该字段定义一个数据帧所包含的数据字节数(0到8)。如果是远程帧请求，则为0。

16. 5. 2. 31 CAN接收FIFO1 邮箱数据低位寄存器(CAN_RXF1DL)

CAN 接收 FIFO1 邮箱数据低位寄存器 (CAN_RXF1DL)																															
偏移地址:0x1C8																															
复位值:0xXXXX XXXX																															
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
BYTE3<7:0>								BYTE2<7:0>								BYTE1<7:0>								BYTE0<7:0>							

BYTE3	Bit 31-24	R	**数据字节3** 消息的数据字节3。
BYTE2	Bit 23-16	R	**数据字节2** 消息的数据字节2。
BYTE1	Bit 15-8	R	**数据字节1** 消息的数据字节1。
BYTE0	Bit 7-0	R	**数据字节0** 消息的数据字节0。

16. 5. 2. 32 CAN接收FIFO1 邮箱数据高位寄存器(CAN_RXF1DH)

CAN 接收 FIFO1 邮箱数据高位寄存器 (CAN_RXF1DH)																															
偏移地址:0x1CC																															
复位值:0xXXXX XXXX																															
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
BYTE7<7:0>								BYTE6<7:0>								BYTE5<7:0>								BYTE4<7:0>							

BYTE7	Bit 31-24	R	**数据字节7** 消息的数据字节7。
BYTE6	Bit 23-16	R	**数据字节6** 消息的数据字节6。
BYTE5	Bit 15-8	R	**数据字节5** 消息的数据字节5。
BYTE4	Bit 7-0	R	**数据字节4** 消息的数据字节4。

16. 5. 2. 33 CAN筛选器控制寄存器(CAN_FLTCON)

CAN 筛选器控制寄存器 (CAN_FLTCON)																												
偏移地址:0x200																												
复位值:0x2A1C 0E01																												
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
																															FLTINI

—	Bit 31-1	—	—
FLTINI	Bit 0	R/W	**筛选器初始化模式** 筛选器组的初始化模式。(在配置筛选器寄存器时，需要将该位置1) 0x0：筛选器工作模式。 0x1：筛选器初始化模式。

注：此寄存器的所有位均由软件置 1 和清零。

16. 5. 2. 34 CAN筛选器模式寄存器(CAN_FLTM)

CAN 筛选器模式寄存器 (CAN_FLTM)																												
偏移地址:0x204																												
复位值:0xXXXX XXXX																												
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
																		MOD<x><13:0>													

—	Bit 31-14	—	—
$MOD_{<x>}$	Bit 13-0	R/W	**筛选器模式** 筛选器x的模式。 0x0：筛选器组x的两个32位寄存器处于屏蔽模式。 0x1：筛选器组x的两个32位寄存器处于标识符列表模式。

注：仅当 CAN_FLTCON 寄存器中设置了筛选器初始化模式(FLTINI=1)时，才能对此寄存器执行写操作。

16. 5. 2. 35 CAN筛选器宽度选择寄存器(CAN_FLTWS)

CAN 筛选器宽度选择寄存器 (CAN_FLTWS)																													
偏移地址:0x20C																													
复位值:0x0000 0000																													
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
																		SEL<x><13:0>													

—	Bit 31-14	—	—
SEL<x>	Bit 13-0	R/W	**宽度选择** 这些位定义了筛选器0-13的宽度选择。 0x0：双16位宽度。 0x1：单32位宽度。

注：仅当 CAN_FLTCON 寄存器中设置了筛选器初始化模式(FLTINI=1)时，才能对此寄存器执行写操作。

16. 5. 2. 36 CAN筛选器FIFO分配寄存器(CAN_FLTAS)

CAN 筛选器 FIFO 分配寄存器 (CAN_FLTAS)																													
偏移地址:0x214																													
复位值:0x0000 0000																													
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
																		ASSIGN<x><13:0>													

—	Bit 31-14	—	—
ASSIGN<x>	Bit 13-0	R/W	**筛选器x的FIFO分配** 通过此筛选器的消息将存储在指定的FIFO中。 0x0：筛选器分配到FIFO 0。 0x1：筛选器分配到FIFO 1。

注：仅当 CAN_FLTCON 寄存器中设置了筛选器初始化模式(FLTINI=1)时，才能对此寄存器执行写操作。

16. 5. 2. 37 CAN筛选器启动寄存器(CAN_FLTGO)

CAN 筛选器启动寄存器 (CAN_FLTGO)																													
偏移地址:0x21C																													
复位值:0x0000 0000																													
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
																		GO<x><13:0>													

—	Bit 31-14	—	—
GO<x>	Bit 13-0	R/W	**筛选器启动** 软件将此位置1可启用筛选器x。要修改筛选器相关寄存器，必须将筛选器关闭或将筛选器设为初始化模式。 0x0：筛选器x关闭。 0x1：筛选器x启用。

16. 5. 2. 38 筛选器组 0 寄存器 1 (CAN_FLT0R1)

筛选器组 0 寄存器 1 (CAN_FLT0R1)																													
偏移地址:0x240																													
复位值:0xXXXX XXXX																													
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
FLTB<x><31:0>																													

FLTB<x>	Bit 31-0	R/W	**筛选器位** 标识符 寄存器的每一位极性。 0x0：需要显性位。 0x1：需要隐性位。 屏蔽 寄存器的对应位是否需要匹配。 0x0：屏蔽，该位无需比较。 0x1：匹配，该位必须与对应的标识符寄存器位极性相同。

16. 5. 2. 39 筛选器组 0 寄存器 2 (CAN_FLT0R2)

筛选器组 0 寄存器 2 (CAN_FLT0R2)																															
偏移地址:0x244																															
复位值:0xXXXX XXXX																															
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
FLTB<x><31:0>																															

FLTB<x>	Bit 31-0	R/W	**筛选器位** 标识符 寄存器的每一位极性。 0x0：需要显性位。 0x1：需要隐性位。 屏蔽 寄存器的对应位是否需要匹配。 0x0：屏蔽，该位无需比较。 0x1：匹配，该位必须与对应的标识符寄存器位极性相同。

16. 5. 2. 40 筛选器组 1 寄存器 1 (CAN_FLT1R1)

筛选器组 1 寄存器 1 (CAN_FLT1R1)																															
偏移地址:0x248																															
复位值:0xXXXX XXXX																															
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
FLTB<x><31:0>																															

FLTB<x>	Bit 31-0	R/W	**筛选器位** 标识符 寄存器的每一位极性。 0x0：需要显性位。 0x1：需要隐性位。 屏蔽 寄存器的对应位是否需要匹配。 0x0：屏蔽，该位无需比较。 0x1：匹配，该位必须与对应的标识符寄存器位极性相同。

16. 5. 2. 41 筛选器组 1 寄存器 2 (CAN_FLT1R2)

筛选器组 1 寄存器 2 (CAN_FLT1R2)																															
偏移地址:0x24C																															
复位值:0xXXXX XXXX																															
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
FLTB<x><31:0>																															

FLTB<x>	Bit 31-0	R/W	**筛选器位** 标识符 寄存器的每一位极性。 0x0：需要显性位。 0x1：需要隐性位。 屏蔽 寄存器的对应位是否需要匹配。 0x0：屏蔽，该位无需比较。 0x1：匹配，该位必须与对应的标识符寄存器位极性相同。

16. 5. 2. 42 筛选器组 2 寄存器 1 (CAN_FLT2R1)

筛选器组 2 寄存器 1 (CAN_FLT2R1)																															
偏移地址:0x250																															
复位值:0xXXXX XXXX																															
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
FLTB<x><31:0>																															

FLTB<x>	Bit 31-0	R/W	**筛选器位** 标识符 寄存器的每一位极性。 0x0：需要显性位。 0x1：需要隐性位。 屏蔽 寄存器的对应位是否需要匹配。 0x0：屏蔽，该位无需比较。 0x1：匹配，该位必须与对应的标识符寄存器位极性相同。

16. 5. 2. 43 筛选器组 2 寄存器 2 (CAN_FLT2R2)

筛选器组 2 寄存器 2 (CAN_FLT2R2)																															
偏移地址:0x254																															
复位值:0xXXXX XXXX																															
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
FLTB<x><31:0>																															

FLTB<x>	Bit 31-0	R/W	**筛选器位** 标识符 寄存器的每一位极性。 0x0：需要显性位。 0x1：需要隐性位。 屏蔽 寄存器的对应位是否需要匹配。 0x0：屏蔽，该位无需比较。 0x1：匹配，该位必须与对应的标识符寄存器位极性相同。

16. 5. 2. 44 筛选器组 3 寄存器 1 (CAN_FLT3R1)

筛选器组 3 寄存器 1 (CAN_FLT3R1)																															
偏移地址:0x258																															
复位值:0xXXXX XXXX																															
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
FLTB<x><31:0>																															

FLTB<x>	Bit 31-0	R/W	**筛选器位** 标识符 寄存器的每一位极性。 0x0：需要显性位。 0x1：需要隐性位。 屏蔽 寄存器的对应位是否需要匹配。 0x0：屏蔽，该位无需比较。 0x1：匹配，该位必须与对应的标识符寄存器位极性相同。

16. 5. 2. 45 筛选器组 3 寄存器 2 (CAN_FLT3R2)

筛选器组 3 寄存器 2 (CAN_FLT3R2)																															
偏移地址:0x25C																															
复位值:0xXXXX XXXX																															
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
FLTB<x><31:0>																															

FLTB<x>	Bit 31-0	R/W	**筛选器位** 标识符 寄存器的每一位极性。 0x0：需要显性位。 0x1：需要隐性位。 屏蔽 寄存器的对应位是否需要匹配。 0x0：屏蔽，该位无需比较。 0x1：匹配，该位必须与对应的标识符寄存器位极性相同。

16. 5. 2. 46 筛选器组 4 寄存器 1 (CAN_FLT4R1)

筛选器组 4 寄存器 1 (CAN_FLT4R1)																															
偏移地址:0x260																															
复位值:0xXXXX XXXX																															
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
FLTB<x><31:0>																															

FLTB<x>	Bit 31-0	R/W	**筛选器位** 标识符 寄存器的每一位极性。 0x0：需要显性位。 0x1：需要隐性位。 屏蔽 寄存器的对应位是否需要匹配。 0x0：屏蔽，该位无需比较。 0x1：匹配，该位必须与对应的标识符寄存器位极性相同。

16. 5. 2. 47 筛选器组 4 寄存器 2 (CAN_FLT4R2)

筛选器组 4 寄存器 2 (CAN_FLT4R2)																															
偏移地址:0x264																															
复位值:0xXXXX XXXX																															
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
FLTB<x><31:0>																															

FLTB<x>	Bit 31-0	R/W	**筛选器位** 标识符 寄存器的每一位极性。 0x0：需要显性位。 0x1：需要隐性位。 屏蔽 寄存器的对应位是否需要匹配。 0x0：屏蔽，该位无需比较。 0x1：匹配，该位必须与对应的标识符寄存器位极性相同。

16. 5. 2. 48 筛选器组 5 寄存器 1 (CAN_FLT5R1)

筛选器组 5 寄存器 1 (CAN_FLT5R1)																															
偏移地址:0x268																															
复位值:0xXXXX XXXX																															
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
FLTB<x><31:0>																															

FLTB<x>	Bit 31-0	R/W	**筛选器位** 标识符 寄存器的每一位极性。 0x0：需要显性位。 0x1：需要隐性位。 屏蔽 寄存器的对应位是否需要匹配。 0x0：屏蔽，该位无需比较。 0x1：匹配，该位必须与对应的标识符寄存器位极性相同。

16. 5. 2. 49 筛选器组 5 寄存器 2 (CAN_FLT5R2)

筛选器组 5 寄存器 2 (CAN_FLT5R2)																															
偏移地址:0x26C																															
复位值:0xXXXX XXXX																															
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
FLTB<x><31:0>																															

FLTB<x>	Bit 31-0	R/W	**筛选器位** 标识符 寄存器的每一位极性。 0x0：需要显性位。 0x1：需要隐性位。 屏蔽 寄存器的对应位是否需要匹配。 0x0：屏蔽，该位无需比较。 0x1：匹配，该位必须与对应的标识符寄存器位极性相同。

16. 5. 2. 50 筛选器组 6 寄存器 1 (CAN_FLT6R1)

筛选器组 6 寄存器 1 (CAN_FLT6R1)																															
偏移地址:0x270																															
复位值:0xXXXX XXXX																															
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
FLTB<x><31:0>																															

FLTB<x>	Bit 31-0	R/W	**筛选器位** 标识符 寄存器的每一位极性。 0x0：需要显性位。 0x1：需要隐性位。 屏蔽 寄存器的对应位是否需要匹配。 0x0：屏蔽，该位无需比较。 0x1：匹配，该位必须与对应的标识符寄存器位极性相同。

16. 5. 2. 51 筛选器组 6 寄存器 2 (CAN_FLT6R2)

筛选器组 6 寄存器 2 (CAN_FLT6R2)																															
偏移地址:0x274																															
复位值:0xXXXX XXXX																															
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
FLTB<x><31:0>																															

FLTB<x>	Bit 31-0	R/W	**筛选器位** 标识符 寄存器的每一位极性。 0x0：需要显性位。 0x1：需要隐性位。 屏蔽 寄存器的对应位是否需要匹配。 0x0：屏蔽，该位无需比较。 0x1：匹配，该位必须与对应的标识符寄存器位极性相同。

16. 5. 2. 52 筛选器组 7 寄存器 1 (CAN_FLT7R1)

筛选器组 7 寄存器 1 (CAN_FLT7R1)																															
偏移地址:0x278																															
复位值:0xXXXX XXXX																															
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
FLTB<x><31:0>																															

FLTB<x>	Bit 31-0	R/W	**筛选器位** 标识符 寄存器的每一位极性。 0x0：需要显性位。 0x1：需要隐性位。 屏蔽 寄存器的对应位是否需要匹配。 0x0：屏蔽，该位无需比较。 0x1：匹配，该位必须与对应的标识符寄存器位极性相同。

16. 5. 2. 53 筛选器组 7 寄存器 2 (CAN_FLT7R2)

筛选器组 7 寄存器 2 (CAN_FLT7R2)																															
偏移地址:0x27C																															
复位值:0xXXXX XXXX																															
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
FLTB<x><31:0>																															

FLTB<x>	Bit 31-0	R/W	**筛选器位** 标识符 寄存器的每一位极性。 0x0：需要显性位。 0x1：需要隐性位。 屏蔽 寄存器的对应位是否需要匹配。 0x0：屏蔽，该位无需比较。 0x1：匹配，该位必须与对应的标识符寄存器位极性相同。

16. 5. 2. 54 筛选器组 8 寄存器 1 (CAN_FLT8R1)

筛选器组 8 寄存器 1 (CAN_FLT8R1)																															
偏移地址:0x280																															
复位值:0xXXXX XXXX																															
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
FLTB<x><31:0>																															

FLTB<x>	Bit 31-0	R/W	**筛选器位** 标识符 寄存器的每一位极性。 0x0：需要显性位。 0x1：需要隐性位。 屏蔽 寄存器的对应位是否需要匹配。 0x0：屏蔽，该位无需比较。 0x1：匹配，该位必须与对应的标识符寄存器位极性相同。

16. 5. 2. 55 筛选器组 8 寄存器 2 (CAN_FLT8R2)

筛选器组 8 寄存器 2 (CAN_FLT8R2)																															
偏移地址:0x284																															
复位值:0xXXXX XXXX																															
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
FLTB<x><31:0>																															

FLTB<x>	Bit 31-0	R/W	**筛选器位** 标识符 寄存器的每一位极性。 0x0：需要显性位。 0x1：需要隐性位。 屏蔽 寄存器的对应位是否需要匹配。 0x0：屏蔽，该位无需比较。 0x1：匹配，该位必须与对应的标识符寄存器位极性相同。

16. 5. 2. 56 筛选器组 9 寄存器 1 (CAN_FLT9R1)

筛选器组 9 寄存器 1 (CAN_FLT9R1)																															
偏移地址:0x288																															
复位值:0xXXXX XXXX																															
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
FLTB<x><31:0>																															

FLTB<x>	Bit 31-0	R/W	**筛选器位** 标识符 寄存器的每一位极性。 0x0：需要显性位。 0x1：需要隐性位。 屏蔽 寄存器的对应位是否需要匹配。 0x0：屏蔽，该位无需比较。 0x1：匹配，该位必须与对应的标识符寄存器位极性相同。

16. 5. 2. 57 筛选器组 9 寄存器 2 (CAN_FLT9R2)

筛选器组 9 寄存器 2 (CAN_FLT9R2)																															
偏移地址:0x28C																															
复位值:0xXXXX XXXX																															
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
FLTB<x><31:0>																															

FLTB<x>	Bit 31-0	R/W	**筛选器位** 标识符 寄存器的每一位极性。 0x0：需要显性位。 0x1：需要隐性位。 屏蔽 寄存器的对应位是否需要匹配。 0x0：屏蔽，该位无需比较。 0x1：匹配，该位必须与对应的标识符寄存器位极性相同。

16. 5. 2. 58 筛选器组 10 寄存器 1 (CAN_FLT10R1)

筛选器组 10 寄存器 1 (CAN_FLT10R1)																															
偏移地址:0x290																															
复位值:0xXXXX XXXX																															
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
FLTB<x><31:0>																															

FLTB<x>	Bit 31-0	R/W	**筛选器位** 标识符 寄存器的每一位极性。 0x0：需要显性位。 0x1：需要隐性位。 屏蔽 寄存器的对应位是否需要匹配。 0x0：屏蔽，该位无需比较。 0x1：匹配，该位必须与对应的标识符寄存器位极性相同。

16. 5. 2. 59 筛选器组 10 寄存器 2 (CAN_FLT10R2)

筛选器组 10 寄存器 2 (CAN_FLT10R2)																															
偏移地址:0x294																															
复位值:0xXXXX XXXX																															
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
FLTB<x><31:0>																															

FLTB<x>	Bit 31-0	R/W	**筛选器位** 标识符 寄存器的每一位极性。 0x0：需要显性位。 0x1：需要隐性位。 屏蔽 寄存器的对应位是否需要匹配。 0x0：屏蔽，该位无需比较。 0x1：匹配，该位必须与对应的标识符寄存器位极性相同。

16. 5. 2. 60 筛选器组 11 寄存器 1 (CAN_FLT11R1)

筛选器组 11 寄存器 1 (CAN_FLT11R1)																															
偏移地址:0x298																															
复位值:0xXXXX XXXX																															
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
FLTB<x><31:0>																															

FLTB<x>	Bit 31-0	R/W	**筛选器位** 标识符 寄存器的每一位极性。 0x0：需要显性位。 0x1：需要隐性位。 屏蔽 寄存器的对应位是否需要匹配。 0x0：屏蔽，该位无需比较。 0x1：匹配，该位必须与对应的标识符寄存器位极性相同。

16. 5. 2. 61 筛选器组 11 寄存器 2 (CAN_FLT11R2)

筛选器组 11 寄存器 2 (CAN_FLT11R2)																															
偏移地址:0x29C																															
复位值:0xXXXX XXXX																															
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
FLTB<x><31:0>																															

FLTB<x>	Bit 31-0	R/W	**筛选器位** 标识符 寄存器的每一位极性。 0x0：需要显性位。 0x1：需要隐性位。 屏蔽 寄存器的对应位是否需要匹配。 0x0：屏蔽，该位无需比较。 0x1：匹配，该位必须与对应的标识符寄存器位极性相同。

16. 5. 2. 62 筛选器组 12 寄存器 1 (CAN_FLT12R1)

筛选器组 12 寄存器 1 (CAN_FLT12R1)																															
偏移地址:0x2A0																															
复位值:0xXXXX XXXX																															
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
FLTB<x><31:0>																															

FLTB<x>	Bit 31-0	R/W	**筛选器位** 标识符 寄存器的每一位极性。 0x0：需要显性位。 0x1：需要隐性位。 屏蔽 寄存器的对应位是否需要匹配。 0x0：屏蔽，该位无需比较。 0x1：匹配，该位必须与对应的标识符寄存器位极性相同。

16. 5. 2. 63 筛选器组 12 寄存器 2 (CAN_FLT12R2)

筛选器组 12 寄存器 2 (CAN_FLT12R2)																															
偏移地址:0x2A4																															
复位值:0xXXXX XXXX																															
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
FLTB<x><31:0>																															

FLTB<x>	Bit 31-0	R/W	**筛选器位** 标识符 寄存器的每一位极性。 0x0：需要显性位。 0x1：需要隐性位。 屏蔽 寄存器的对应位是否需要匹配。 0x0：屏蔽，该位无需比较。 0x1：匹配，该位必须与对应的标识符寄存器位极性相同。

16. 5. 2. 64 筛选器组 13 寄存器 1 (CAN_FLT13R1)

筛选器组 13 寄存器 1 (CAN_FLT13R1)																															
偏移地址:0x2A8																															
复位值:0xXXXX XXXX																															
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
FLTB<x><31:0>																															

FLTB<x>	Bit 31-0	R/W	**筛选器位** 标识符 寄存器的每一位极性。 0x0：需要显性位。 0x1：需要隐性位。 屏蔽 寄存器的对应位是否需要匹配。 0x0：屏蔽，该位无需比较。 0x1：匹配，该位必须与对应的标识符寄存器位极性相同。

16. 5. 2. 65 筛选器组 13 寄存器 2 (CAN_FLT13R2)

筛选器组 13 寄存器 2 (CAN_FLT13R2)																															
偏移地址:0x2AC																															
复位值:0xXXXX XXXX																															
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
FLTB<x><31:0>																															

FLTB<x>	Bit 31-0	R/W	**筛选器位** 标识符 寄存器的每一位极性。 0x0：需要显性位。 0x1：需要隐性位。 屏蔽 寄存器的对应位是否需要匹配。 0x0：屏蔽，该位无需比较。 0x1：匹配，该位必须与对应的标识符寄存器位极性相同。

第17章　空间矢量加速器(SVA)

17. 1 概述

SVA 模块是将 BLDC/PMSM 中的 FOC(Field Orientated Control)的常用运算提供运算加速功能，并计算出对应结果以提供 Timer 产生对应的 PWM 输出。

17. 2 特性

- Clarke 正逆转换
- Park 正逆转换
- SVPWM
- 输入输出使用有符号 16 位定点数表示
- 使用 AHB 时钟作为运算时钟
- 运算结束产生 DMA 要求
- 运算结束产生中断

17. 3 结构图

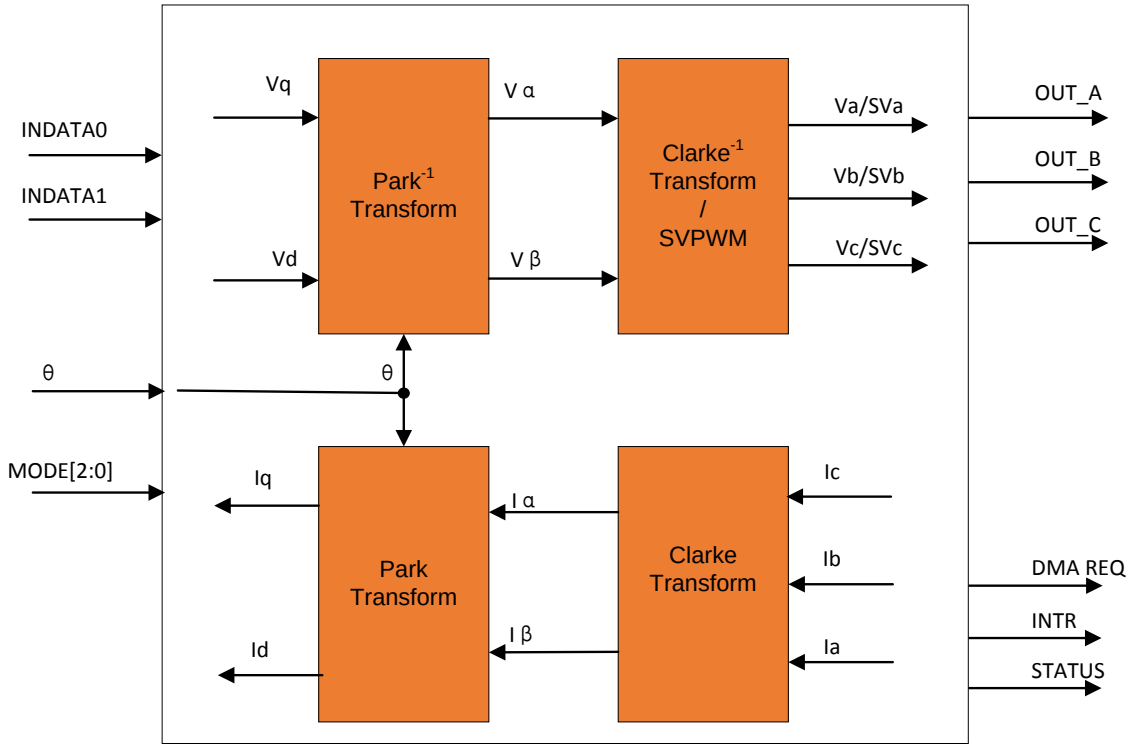


图 17-1　SVA 结构图

17. 4 功能描述

17. 4. 1 基本运算

17. 4. 1. 1 Clarke转换

将 I_a, I_b 和 I_c 转换为 I_α 和 I_β，转换公式如下:

$$I_\alpha = I_a$$

$$I_\beta = \frac{I_a + 2 \cdot I_b}{\sqrt{3}}$$

注: 由于三项电流合值为 0，因此输入公式只采用 I_a,I_b；若实际电流取样为 I_c 时，用户须先推算出 I_a 和 I_b。

17. 4. 1. 2 Clarke逆转换

将 V_α 和 V_β 转换为 V_a, V_b 和 V_c，转换公式如下:

$$V_a = V_\alpha$$

$$V_b = \frac{-V_\alpha + \sqrt{3} \cdot V_\beta}{2}$$

$$V_c = \frac{-V_\alpha - \sqrt{3} \cdot V_\beta}{2}$$

17. 4. 1. 3 Park转换

将 I_α 和 I_β 转换成 I_d 和 I_q，需额外提供角度 θ，转换公式如下:

$$I_d = I_\alpha \cdot \cos\theta + I_\beta \cdot \sin\theta$$

$$I_q = I_\beta \cdot \cos\theta - I_\alpha \cdot \sin\theta$$

17. 4. 1. 4 Park逆转换

将 V_d 和 V_q 转换成 V_α 和 V_β，需额外提供角度 θ，转换公式如下:

$$V_\alpha = V_d \cdot \cos\theta - V_q \cdot \sin\theta$$

$$V_\beta = V_q \cdot \cos\theta + V_d \cdot \sin\theta$$

17. 4. 1. 5 SVPWM

将 V_α 和 V_β 转换为 PWM 对应数值 SV_a, SV_b 和 SV_c. 转换公式是以 Timer 先上数再下数产生 PWM 波型为依据。

SVPWM 的波型会根据在不同扇区 V_α, V_β 有所不同。

- 扇区 1, 4

$$SV_a = \frac{1\cdot V_\beta+\sqrt{3}\cdot V_\alpha}{4}+\frac{1}{2}$$

$$SV_b = \frac{3\cdot V_\beta-\sqrt{3}\cdot V_\alpha}{4}+\frac{1}{2}$$

$$SV_c = \frac{-1\cdot V_\beta-\sqrt{3}\cdot V_\alpha}{4}+\frac{1}{2}$$

- 扇区 2, 5

$$SV_a=\frac{\sqrt{3}\cdot V_\alpha}{2}+\frac{1}{2}$$

$$SV_b=\frac{1\cdot V_\beta}{2}+\frac{1}{2}$$

$$SV_c=\frac{-1\cdot V_\beta}{2}+\frac{1}{2}$$

- 扇区 3, 6

$$SV_a = \frac{-1\cdot V_\beta+\sqrt{3}\cdot V_\alpha}{4}+\frac{1}{2}$$

$$SV_b = \frac{1\cdot V_\beta-\sqrt{3}\cdot V_\alpha}{4}+\frac{1}{2}$$

$$SV_c = \frac{-3\cdot V_\beta-\sqrt{3}\cdot V_\alpha}{4}+\frac{1}{2}$$

17. 4. 1. 6 角度换算

SVA 输入角度换算公式为

$$INANLGE = \frac{\theta}{\pi} X\ 32768$$

下图表示 SVA 输入角度和角度换算的关系。

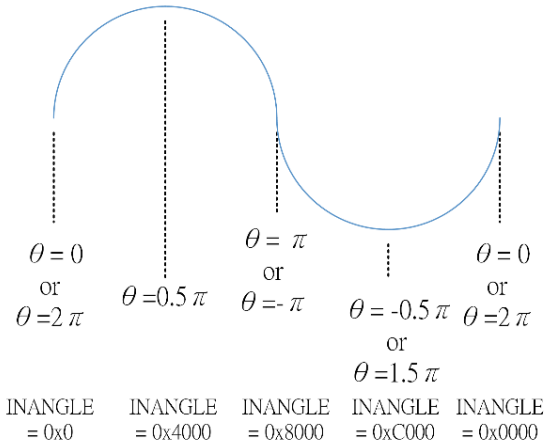


图 17-2 SVA 输入角度和角度换算的关系图

17. 4. 2 执行模式

SVA 可通过 **SVA_CTRL**.OP_MODE 寄存器配置决定执行那些运算，也包含部分基本运算的串接。对应的功能以及输入输出表示请参考下表。

OP_MODE	Function	Input	Output
3'b000	Clarke	SVA_INDATA0: Ia SVA_INDATA1: Ib SVA_INANGLE: N/A	SVA_OTDATA0: Iα SVA_OTDATA1: Iβ SVA_OTDATA2: N/A
3'b001	Park	SVA_INDATA0: Iα SVA_INDATA1: Iβ SVA_INANGLE: θ	SVA_OTDATA0: Id SVA_OTDATA1: Iq SVA_OTDATA2: N/A
3'b010	Clarke and Park	SVA_INDATA0: Ia SVA_INDATA1: Ib SVA_INANGLE: θ	SVA_OTDATA0: Id SVA_OTDATA1: Iq SVA_OTDATA2: N/A
3'b011	Inverse Park	SVA_INDATA0: Vd SVA_INDATA1: Vq SVA_INANGLE: θ	SVA_OTDATA0: Vα SVA_OTDATA1: Vβ SVA_OTDATA2: N/A
3'b100	Inverse Clarke	SVA_INDATA0: Vα SVA_INDATA1: Vβ SVA_INANGLE: N/A	SVA_OTDATA0: Va SVA_OTDATA1: Vb SVA_OTDATA2: Vc
3'b101	Inverse Park and Inverse Clarke	SVA_INDATA0: Vd SVA_INDATA1: Vq SVA_INANGLE: θ	SVA_OTDATA0: Va SVA_OTDATA1: Vb SVA_OTDATA2: Vc
3'b110	SVPWM	SVA_INDATA0: Vα SVA_INDATA1: Vβ SVA_INANGLE: N/A	SVA_OTDATA0: SVa SVA_OTDATA1: SVb SVA_OTDATA2: SVc
3'b111	Inverse Park and SVPWM	SVA_INDATA0: Vd SVA_INDATA1: Vq SVA_INANGLE: θ	SVA_OTDATA0: SVa SVA_OTDATA1: SVb SVA_OTDATA2: SVc

17. 4. 3 控制流程

下面介绍三种不同读取运算结果的方式及流程。

- Pulling Status
- 使用中断
- 使用 DMA

17. 4. 3. 1 Pulling Status

1. 配置 **SVA_CTRL**.OP_MODE。
2. 根据 OP_MODE 配置对应的 **SVA_INDATA0**, **SVA_INDATA1** ,**SVA_INANGLE**。
3. 对 **SVA_CTRL**.STR_TRIG 写 1 启动运算。
4. 监控 **SVA_STATUS**.BUSY 是否为 0，若为 0 表示运算结束。
5. 读取对应的 **SVA_OTDATA0**, **SVA_OTDATA1**, **SVA_OTDATA2**。

17. 4. 3. 2 使用中断

1. C code 规划 SVA 的 IRQ Function。
   - 读取对应的 **SVA_OTDATA0**, **SVA_OTDATA1**, **SVA_OTDATA2**。
   - 当进入 IRQ Function 对 **SVA_ICR**.DONE 写 1 清除中断。
2. 配置 **SVA_IER**.DONE 为 1。
3. 配置 **SVA_CTRL**.OP_MODE。
4. 根据 OP_MODE 配置对应的 **SVA_INDATA0**, **SVA_INDATA1**, **SVA_INANGLE**。
5. 对 **SVA_CTRL**.STR_TRIG 写 1 启动运算，并在 IRQ Function 读取运算结果。

17. 4. 3. 3 使用 DMA

1. DMA 相关设定
   - DMA IRQ Function。
   - DMA MUX 启动对应 SVA 的请求信号。
2. 配置 **SVA_CTRL**.DMA_EN 为 1。
3. 配置 **SVA_CTRL**.OP_MODE。
4. 对 **SVA_CTRL**.STR_TRIG 写 1 启动运算。在 SVA 运算后会发出 DMA 请求将运算结果搬移至 SRAM。

17. 4. 3. 4 使用 DMA 启动

可对应上面三个流程将用户激活 SVA 运算的步骤改由 DMA 写入 STR_TRIG 启动运算。

1. C code 规划对应 SRAM 空间的写入数据对应 **SVA_INDATA0**, **SVA_INDATA1**, **SVA_INANGLE** 和 **SVA_CTRL**。
   - STR_TRIG 对应 bit 必须设为 1。
2. 使用 DMA 的 M2M 模式启动，并将目的地起始地址设为 **SVA_INDATA0**。
3. DMA 在搬移最后一笔数据时写入 **SVA_CTRL** 会同时启动 SVA 运算。

17. 4. 4 其他功能

17. 4. 4. 1 PWM输出倍率

当运行有关 Inverse Clarke 或 SVPWM 有关的运算时，可将结果再乘上一个数值倍率。

1. 配置 **SVA_CTRL**.OUTMUL_EN 为 1
2. 配置增益到 **SVA_OUTMUL** 寄存器
   - 启动其他运算不会启动此功能。
3. 运行有关 Inverse Clarke 或 SVPWM 有关的运算时，会将输出再乘上配置的增益数值。

17. 4. 4. 2 扇区变更标志位

在马达应用上，通过三相信号之间 PWM 宽度的比较来提示扇区变更。

1. 配置门限值到 SVA_PWMTH 寄存器
2. 当运行有关 Inverse Clarke 或 SVPWM 有关的运算后，若任意两个输出小于此门限值则会反应到 **SVA_STATUS**.SECTOR_FLAG。
   - 每次启动运算会将 SECTOR_FLAG 清 0。
     - 启动其他运算也会清为 0，需在执行下一次运算前读走结果。
   - 启动其他运算不会在输出结果时启动此功能。
   - 门限默认为 0，当配置大于 1 时会启动该功能。

17. 5 特殊功能寄存器

17. 5. 1 寄存器列表

SVA 寄存器列表			
名称	偏移地址	类型	描述
SVA_IER	0000_H	W1	SVA 中断开启寄存器
SVA_IDR	0004_H	W1	SVA 中断关闭寄存器
SVA_IVS	0008_H	R	SVA 中断功能有效状态寄存器
SVA_RIF	$000C_H$	R	SVA 原始中断状态寄存器
SVA_IFM	0010_H	R	SVA 中断标志位状态寄存器
SVA_ICR	0014_H	C_W1	SVA 中断清除寄存器
SVA_INDATA0	0018_H	R/W	SVA 输入数据 0
SVA_INDATA1	$001C_H$	R/W	SVA 输入数据 1
SVA_INANGLE	0020_H	R/W	SVA 输入角度
SVA_CTRL	0024_H	R/W	SVA 控制寄存器
SVA_OTDATA0	0028_H	R	SVA 输出数据 0
SVA_OTDATA1	$002C_H$	R	SVA 输出数据 1
SVA_OTDATA2	0030_H	R	SVA 输出数据 2
SVA_STATUS	0034_H	R	SVA 状态寄存器
SVA_OUTMUL	0038_H	R/W	SVA 输出倍率因子
SVA_PWMTH	$003C_H$	R/W	SVA PWM 宽度侦测门限值

17. 5. 2 寄存器描述

17. 5. 2. 1 SVA中断开启寄存器(SVA_IER)

SVA 中断开启寄存器 (SVA_IER)																												
偏移地址:0x00																												
复位值:0x0000 0000																												
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																															DONE

—	Bit 31-1	—	—
DONE	Bit 0	W1	**SVA 运算完成中断开启** 0: 写 0 无效 1: SVA 运算完成中断开启

17. 5. 2. 2 SVA中断关闭寄存器(SVA_IDR)

SVA 中断关闭寄存器 (SVA_IDR)																												
偏移地址:0x04																												
复位值:0x0000 0000																												
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																															DONE

—	Bit 31-1	—	—
DONE	Bit 0	W1	**SVA运算完成中断关闭** 0: 写0无效 1: SVA运算完成中断关闭

17. 5. 2. 3 SVA中断功能有效状态寄存器(SVA_IVS)

SVA 中断功能有效状态寄存器 (SVA_IVS)																													
偏移地址:0x08																													
复位值:0x0000 0000																													
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																															DONE

—	Bit 31-1	—	—
DONE	Bit 0	R	**SVA运算完成中断功能有效状态** 0: SVA运算完成中断功能处于关闭状态 1: SVA运算完成中断功能处于开启状态

17. 5. 2. 4 SVA原始中断状态寄存器(SVA_RIF)

SVA 原始中断状态寄存器 (SVA_RIF)																													
偏移地址:0x0C																													
复位值:0x0000 0000																													
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																															DONE

—	Bit 31-1	—	—
DONE	Bit 0	R	**SVA运算完成中断状态** 0: 无发生中断 1: 已发生中断

17. 5. 2. 5 SVA中断标志位状态寄存器(SVA_IFM)

SVA 中断标志位状态寄存器 (SVA_IFM)																												
偏移地址:0x10																												
复位值:0x0000 0000																												
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																															DONE

—	Bit 31-1	—	—
DONE	Bit 0	R	**SVA运算完成中断标志位状态** 0: 无发生中断或没有开启中断 1: 已发生中断

17. 5. 2. 6 SVA中断清除寄存器(SVA_ICR)

SVA 中断清除寄存器 (SVA_ICR)																												
偏移地址:0x14																												
复位值:0x0000 0000																												
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																															DONE

—	Bit 31-1	—	—
DONE	Bit 0	W1	**SVA运算完成中断清除** 0: 写 0 无效 1: 清除中断事件

17. 5. 2. 7 SVA输入数据 0 (SVA_INDATA0)

SVA 输入数据 0 (SVA_INDATA0)																													
偏移地址:0x18																													
复位值:0x0000 0000																													
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																INDATA0<15:0>															

—	Bit 31-16	—	—
INDATA0	Bit 15-0	R/W	**输入数据 0** 16位有符号数

17. 5. 2. 8 SVA输入数据 1 (SVA_INDATA1)

SVA 输入数据 1 (SVA_INDATA1)																													
偏移地址:0x1C																													
复位值:0x0000 0000																													
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																INDATA1<15:0>															

—	Bit 31-16	—	—
INDATA1	Bit 15-0	R/W	**输入数据1** 16位有符号数

17. 5. 2. 9 SVA输入角度(SVA_INANGLE)

SVA 输入角度 (SVA_INANGLE)																													
偏移地址:0x20																													
复位值:0x0000 0000																													
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																INANGLE<15:0>															

—	Bit 31-16	—	—
INANGLE	Bit 15-0	R/W	**输入角度** 16位有符号数，用来表示 [−π,π)

			$\theta = \frac{INANGLE}{32768} \times \pi$ 0x8000 : −π 0x7FFF : π

17. 5. 2. 10 SVA控制寄存器(SVA_CTRL)

SVA 控制寄存器 (SVA_CTRL)

偏移地址:0x24

复位值:0x0000 0000

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

—	Bit 31-7	—	—
DMA_EN	Bit 6	R/W	**DMA 请求开关** 0:关闭 1:SVA运算完毕后会发送DMA请求
OUTMUL_EN	Bit 5	R/W	**SVA 输出倍率功能开关** 0:关闭 1:Clarke逆转换和SVPWM的输出最后会乘上SVA_OUTMUL.OUTMUL。
EXCHAHGE	Bit 4	R/W	**输入参数交换** 0:输入参数无互换 1:输入数据0和1互换,仅用于SVPWM或Inverse Clarke运算时
OP_MODE	Bit 3-1	R/W	**SVA 运算模式** 3'b000: Clarke 3'b001: Park 3'b010: Clarke and Park 3'b011: Inverse Park 3'b100: Inverse Clarke 3'b101: Inverse Park and Inverse Clarke 3'b110: SVPWM 3'b111: Inverse Park and SVPWM
STR_TRIG	Bit 0	W1	**SVA 启动** 0:无作用 1:写1启动SVA运算.

17. 5. 2. 11 SVA输出数据 0(SVA_OTDATA0)

SVA 输出数据 0 (SVA_OTDATA0)																													
偏移地址:0x28																													
复位值:0x0000 0000																													
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																OTDATA0<15:0>															

—	Bit 31-16	—	—
OTDATA0	Bit 15-0	R	**输出数据 0** 16位有符号数

17. 5. 2. 12 SVA输出数据 1(SVA_OTDATA1)

SVA 输出数据 1 (SVA_OTDATA1)																													
偏移地址:0x2C																													
复位值:0x0000 0000																													
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																OTDATA1<15:0>															

—	Bit 31-16	—	—
OTDATA1	Bit 15-0	R	**输出数据1** 16位有符号数

17. 5. 2. 13 SVA输出数据 2(SVA_OTDATA2)

SVA 输出数据 2 (SVA_OTDATA2)																													
偏移地址:0x30																													
复位值:0x0000 0000																													
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																OTDATA2<15:0>															

—	Bit 31-16	—	—

OTDATA2	Bit 15-0	R	**输出数据2** 16位有符号数

17. 5. 2. 14 SVA状态寄存器(SVA_STATUS)

SVA 状态寄存器 (SVA_STATUS)

偏移地址:0x34

复位值:0x0000 001E

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
																								SECTOR_FLAG<2:0>			OP_MODE<3:0>				BUSY

—	Bit 31-8	—	—
SECTOR_FLAG	Bit 7-5	R	**扇区交换标志位** 这些标志位会在两两PWM宽度接近(宽度差距小于 **SVA_PWMTH**.PWM_TH)被置1 且需要在执行SVPWM或inverse Clarke后才能读出对应的结果 bit 7置1: PWM A and C宽度小于门限 bit 6置1: PWM B and C宽度小于门限 bit5置1: PWM A and B宽度小于门限
OP_MODE	Bit 4-1	R	**SVA 运算模式** 显示当前或前一次运算使用的模式。4'b0000: Clarke 4'b0001: Park 4'b0010: Clarke and Park 4'b0011: Inverse Park 4'b0100: Inverse Clarke 4'b0101: Inverse Park and Inverse Clarke 4'b0110: SVPWM 4'b0111: Inverse Park and SVPWM 4'b1111: SVA is reset before be triggered otherwise: 非法状态
BUSY	Bit 0	R	**SVA 忙碌标志位** 启动SVA后会置1 0: 待命中 1: SVA运算中

17. 5. 2. 15 SVA输出倍率因數(SVA_OUTMUL)

SVA 输出倍率因數 (SVA_OUTMUL)																													
偏移地址:0x38																													
复位值:0x0000 8000																													
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
																OUTMUL<15:0>															

—	Bit 31-16	—	—
OUTMUL	Bit 15-0	R/W	**SVA 输出倍率因子** 当 SVA_CTRL.OUTMUL_EN=1 ， 且 执 行 Clarke逆转换和SVPWM相关运算，输出最后会乘上此数值。 Gain = OUTMUL / 2^{15} 格式为无符号数、1位整数以及15位小数，数值范围[0, 2)。

17. 5. 2. 16 SVA PWM宽度侦测门限值(SVA_PWMTH)

SVAPWM 宽度侦测门限值 (SVA_PWMTH)																													
偏移地址:0x3C																													
复位值:0x0000 0000																													
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																PWM_TH<15:0>															

—	Bit 31-16	—	—
PWM_TH	Bit 15-0	R/W	**SVA PWM宽度侦测门限值** 会侦测Clarke逆转换和SVPWM的三组输出数值(经过OUTMUL处理后)是否相差小于门限值，若有则在反应对应的标志位在SVA_STATUS。 格式为16位无符号数

第18章 无限脉冲响应滤波器(IIR)

18. 1 概述

IIR 提供一个滤波器的运算加速器，可用于滤除信号非目标频段的信号。

18. 2 特性

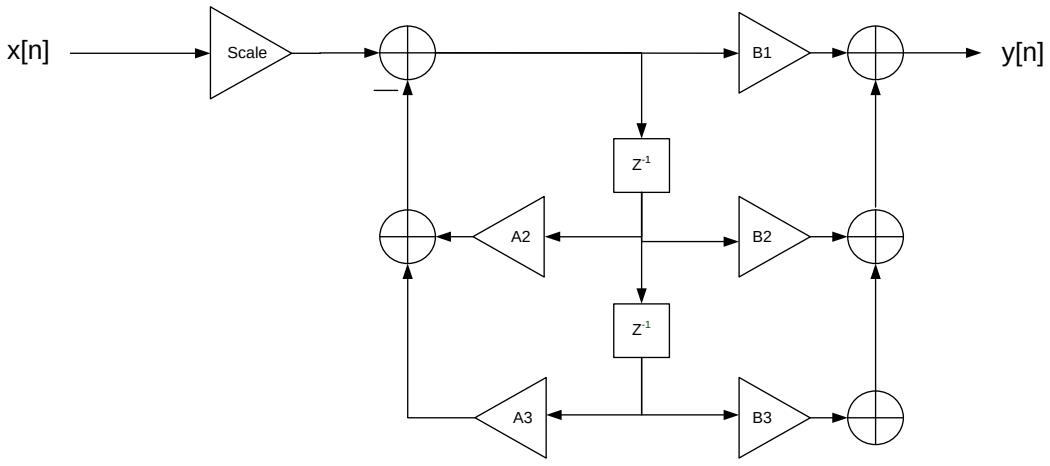
- 使用双二阶滤波器架构
- 有两级滤波器，可配置采用一级或两级。
- 支持系数由用户配置。

18. 3 功能描述

18. 3. 1 双二阶滤波器 (Biqaud Filter)

滤波器可为用户提供额外的信号处理，共有两级双二阶滤波。可通过 IIR_CTRL.mode 配置一个双二阶滤波器架构如下图所示，由 6 个系数组成。

- Biquad Filter structure



反馈寄存器可表示为W_{n-1}，W_{n-2}，计算结果可表示为

$$W_n=\left(Scale\cdot X_n-\sum_{k=2}^{3}A_k\cdot W_{n+1-k}\right)$$

$$y_n=\sum_{k=1}^{3}B_k\cdot W_{n+1-k}$$

转移函数(Transfer Function)可表示为$Scale\cdot\frac{B_1+B_2Z^{-1}+B_3Z^{-2}}{1+A_2Z^{-1}+A_3Z^{-2}}$

18. 3. 1. 1 数值范围

- 输入/输出: 16bits 有符号数。小数位元数为 14bit，数值范围[-2, 2) 。
- 滤波器系数: 24bits 有符号数。小数位元数为 19bit，数值范围[-16, 16) 。
- 反馈暂存数值: 24bits 有符号数。小数位元数为 15bit，数值范围[-256, 256) 。

18. 3. 1. 2 使用方式

1. 通过 **IIR_CTRL**.mode 配置使用阶数并配置 Scale 等相关用户决定好的系数。
2. 再通过 **IIR_CTRL**.init 初始化内部数值、并在 **IIR_INDATA** 写入滤波器输入数值。
3. 监控 **IIR_STATUS**.Busy 确认是否预算完毕；确定后再通过 **IIR_OUTDATA** 读出运算后的结果。
4. 后续重复以上步骤，之前运算的中间数值结果会保留在硬件内部直到被用户初始化。

18. 4 特殊功能寄存器

18. 4. 1 寄存器列表

IIR 寄存器列表			
名称	偏移地址	类型	描述
IIR_CTRL	0000_H	R/W	IIR 控制寄存器
IIR_STATUS	0004_H	R	IIR 状态寄存器
IIR_INDATA	0008_H	R/W	IIR 输入数据
IIR_OUTDATA	$000C_H$	R	IIR 输出数据
IIR_SCALE_0	0010_H	R/W	IIR 第一级滤波器缩放倍率
IIR_COEB1_0	0014_H	R/W	IIR 第一级系数 B1
IIR_COEB2_0	0018_H	R/W	IIR 第一级系数 B2
IIR_COEB3_0	$001C_H$	R/W	IIR 第一级系数 B3
IIR_COEA2_0	0020_H	R/W	IIR 第一级系数 A2
IIR_COEA3_0	0024_H	R/W	IIR 第一级系数 A3
IIR_INTD1_0	0028_H	R/W	IIR 第一级延迟寄存器 1
IIR_INTD2_0	$002C_H$	R/W	IIR 第一级延迟寄存器 2
IIR_SCALE_1	0030_H	R/W	IIR 第二级滤波器缩放倍率
IIR_COEB1_1	0034_H	R/W	IIR 第二级系数 B1
IIR_COEB2_1	0038_H	R/W	IIR 第二级系数 B2
IIR_COEB3_1	$003C_H$	R/W	IIR 第二级系数 B3
IIR_COEA2_1	0040_H	R/W	IIR 第二级系数 A2
IIR_COEA3_1	0044_H	R/W	IIR 第二级系数 A3
IIR_INTD1_1	0048_H	R/W	IIR 第二级延迟寄存器 1
IIR_INTD2_1	$004C_H$	R/W	IIR 第二级延迟寄存器 2

18. 4. 2 寄存器描述

18. 4. 2. 1 IIR控制寄存器(IIR_CTRL)

IIR 控制寄存器 (IIR_CTRL)																												
偏移地址:0x00																												
复位值:0x0000 0000																												
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																													MODE	INIT	STR_TRIG

—	Bit 31-3	—	—
MODE	Bit 2	R/W	**IIR Mode选择** 0:使用一级 Biquad 滤波器 1:使用二级 Biquad 滤波器
INIT	Bit 1	W1	**IIR 初始化** 0:无作用 1:写 1 Reset IIR 内部初始值
STR_TRIG	Bit 0	W1	**IIR 启动** 0:无作用 1:写 1 启动 IIR 运算

18. 4. 2. 2 IIR状态寄存器(IIR_STATUS)

IIR 状态寄存器 (IIR_STATUS)																												
偏移地址:0x04																												
复位值:0x0000 0000																												
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

—	Bit 31-1	—	—
BUSY	Bit 0	R	**IIR运算状态** 0: ready 1: busy

18. 4. 2. 3 IIR输入数据(IIR_INDATA)

IIR 输入数据 (IIR_INDATA)																															
偏移地址:0x08																															
复位值:0x0000 0000																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																INDATA<15:0>															

—	Bit 31-16	—	—
INDATA	Bit 15-0	R/W	**输入数据** IIR滤波器输入数据 16位有符号数，1位符号位以及15位小数。数值范围[-1, 1)

18. 4. 2. 4 IIR输出数据(IIR_OUTDATA)

IIR 输出数据 (IIR_OUTDATA)																															
偏移地址:0x0C																															
复位值:0x0000 0000																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																OUTDATA<15:0>															

—	Bit 31-16	—	—
OUTDATA	Bit 15-0	R	**输出数据** IIR滤波器输出结果. 16位有符号数，1位符号位以及15位小数。数值范围[-1, 1)

18. 4. 2. 5 IIR第一级滤波器缩放倍率(IIR_SCALE_0)

IIR 第一级滤波器缩放倍率 (IIR_SCALE_0)																															
偏移地址:0x10																															
复位值:0x0000 0000																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								SCALE<23:0>																							

—	Bit 31-24	—	—
SCALE	Bit 23-0	R/W	**缩放倍率** IIR滤波器输入缩放倍率 24位有符号数，1位符号位、4位整数以及19位小数。数值范围[-16, 16)

18. 4. 2. 6 IIR第一级系数B1 (IIR_COEB1_0)

IIR 第一级系数 B1 (IIR_COEB1_0)																															
偏移地址:0x14																															
复位值:0x0000 0000																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								COEB1<23:0>																							

—	Bit 31-24	—	—
COEB1	Bit 23-0	R/W	**滤波器系数B1** 24位有符号数，1位符号位、4位整数以及19位小数。数值范围[-16, 16)

18. 4. 2. 7 IIR第一级系数B2 (IIR_COEB2_0)

IIR 第一级系数 B2 (IIR_COEB2_0)																															
偏移地址:0x18																															
复位值:0x0000 0000																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								COEB2<23:0>																							

—	Bit 31-24	—	—
COEB2	Bit 23-0	R/W	**滤波器系数B2** 24位有符号数，1位符号位、4位整数以及19位小数。数值范围[-16, 16)

18. 4. 2. 8 IIR第一级系数B3 (IIR_COEB3_0)

IIR 第一级系数 B3 (IIR_COEB3_0)																															
偏移地址:0x1C																															
复位值:0x0000 0000																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								COEB3<23:0>																							

—	Bit 31-24	—	—
COEB3	Bit 23-0	R/W	**滤波器系数B3** 24位有符号数，1位符号位、4位整数以及19位小数。数值范围[-16, 16)

18. 4. 2. 9 IIR第一级系数A2 (IIR_COEA2_0)

IIR 第一级系数 A2 (IIR_COEA2_0)																															
偏移地址:0x20																															
复位值:0x0000 0000																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								COEA2<23:0>																							

—	Bit 31-24	—	—
COEA2	Bit 23-0	R/W	**IIR滤波器系数A2** 24位有符号数，1位符号位、4位整数以及19位小数。数值范围[-16, 16)

18. 4. 2. 10 IIR第一级系数A3 (IIR_COEA3_0)

IIR 第一级系数 A3 (IIR_COEA3_0)																															
偏移地址:0x24																															
复位值:0x0000 0000																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								COEA3<23:0>																							

—	Bit 31-24	—	—
COEA3	Bit 23-0	R/W	**IIR滤波器系数A3** 24位有符号数，1位符号位、4位整数以及19位小数。数值范围[-16, 16)

18. 4. 2. 11 IIR第一级延迟寄存器 1 (IIR_INTD1_0)

IIR 第一级延迟寄存器 1 (IIR_INTD1_0)																															
偏移地址:0x28																															
复位值:0x0000 0000																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								INTD1<23:0>																							

—	Bit 31-24	—	—
INTD1	Bit 23-0	R/W	**IIR延迟寄存器1** 24位有符号数，1位符号位、8位整数以及15位小数。数值范围[-256, 256)

18. 4. 2. 12 IIR第一级延迟寄存器 2 (IIR_INTD2_0)

IIR 第一级延迟寄存器 2 (IIR_INTD2_0)																															
偏移地址:0x2C																															
复位值:0x0000 0000																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								INTD2<23:0>																							

—	Bit 31-24	—	—
INTD2	Bit 23-0	R/W	**IIR延迟寄存器2** 24位有符号数，1位符号位、8位整数以及15位小数。数值范围[-256, 256)

18. 4. 2. 13 IIR第二级滤波器缩放倍率(IIR_SCALE_1)

IIR 第二级滤波器缩放倍率 (IIR_SCALE_1)																															
偏移地址:0x30																															
复位值:0x0000 0000																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								SCALE<23:0>																							

—	Bit 31-24	—	—
SCALE	Bit 23-0	R/W	**缩放倍率** IIR滤波器输入缩放倍率 24位有符号数，1位符号位、4位整数以及19位小数。数值范围[-16, 16)

18. 4. 2. 14 IIR第二级系数B1 (IIR_COEB1_1)

IIR 第二级系数 B1 (IIR_COEB1_1)																															
偏移地址:0x34																															
复位值:0x0000 0000																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								COEB1<23:0>																							

—	Bit 31-24	—	—
COEB1	Bit 23-0	R/W	**滤波器系数B1** 24位有符号数，1位符号位、4位整数以及19位小数。数值范围[-16, 16)

18. 4. 2. 15 IIR第二级系数B2 (IIR_COEB2_1)

IIR 第二级系数 B2 (IIR_COEB2_1)																															
偏移地址:0x38																															
复位值:0x0000 0000																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								COEB2<23:0>																							

—	Bit 31-24	—	—
COEB2	Bit 23-0	R/W	**滤波器系数B2** 24位有符号数，1位符号位、4位整数以及19位小数。数值范围[-16, 16)

18. 4. 2. 16 IIR第二级系数B3 (IIR_COEB3_1)

IIR 第二级系数 B3 (IIR_COEB3_1)																															
偏移地址:0x3C																															
复位值:0x0000 0000																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								COEB3<23:0>																							

—	Bit 31-24	—	—
COEB3	Bit 23-0	R/W	**滤波器系数B3** 24位有符号数，1位符号位、4位整数以及19位小数。数值范围[-16, 16)

18. 4. 2. 17 IIR第二级系数A2 (IIR_COEA2_1)

IIR 第二级系数 A2 (IIR_COEA2_1)																															
偏移地址:0x40																															
复位值:0x0000 0000																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								COEA2<23:0>																							
—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

—	Bit 31-24	—	—
COEA2	Bit 23-0	R/W	**IIR滤波器系数A2** 24位有符号数，1位符号位、4位整数以及19位小数。数值范围[-16, 16)

18. 4. 2. 18 IIR第二级系数A3 (IIR_COEA3_1)

IIR 第二级系数 A3 (IIR_COEA3_1)																															
偏移地址:0x44																															
复位值:0x0000 0000																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								COEA3<23:0>																							
—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

—	Bit 31-24	—	—
COEA3	Bit 23-0	R/W	**IIR滤波器系数A3** 24位有符号数，1位符号位、4位整数以及19位小数。数值范围[-16, 16)

18. 4. 2. 19 IIR第二级延迟寄存器 1 (IIR_INTD1_1)

IIR 第二级延迟寄存器 1 (IIR_INTD1_1)																															
偏移地址:0x48																															
复位值:0x0000 0000																															
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
								INTD1<23:0>																							

—	Bit 31-24	—	—
INTD1	Bit 23-0	R/W	**IIR延迟寄存器1** 24位有符号数，1位符号位、8位整数以及15位小数。数值范围[-256, 256)

18. 4. 2. 20 IIR第二级延迟寄存器 2 (IIR_INTD2_1)

IIR 第二级延迟寄存器 2 (IIR_INTD2_1)																															
偏移地址:0x4C																															
复位值:0x0000 0000																															
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
								INTD2<23:0>																							

—	Bit 31-24	—	—
INTD2	Bit 23-0	R/W	**IIR延迟寄存器2** 24位有符号数，1位符号位、8位整数以及15位小数。数值范围[-256, 256)

第19章　电阻分压电路控制器(VRES)

19. 1 概述

此电路主要使用电阻分压技术来依据模拟电源(VDDA)进行电压分压。分压后的信号分别连接至比较器(CMP)、运算放大器(OPAMP)及模拟数字转换器(ADC)。本章节仅介绍经过电阻分压后的信号会接至哪一个模拟外设，而有关该外设的相关配置详细内容，请参阅各外设的章节描述。

19. 2 结构图

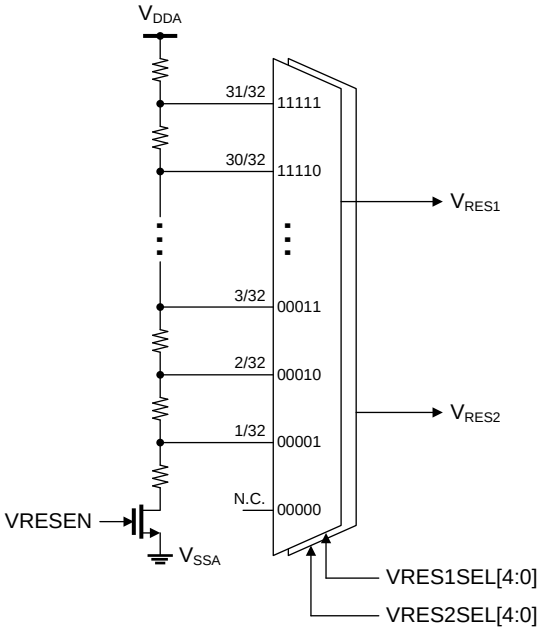


图 20-1　电路架构图

19. 3 模拟外设连接表

模拟外设	V_{RES1} 输入	V_{RES2} 输入
CMP1	O	O
CMP2	O	O
OPAMP1 VINP	O	O
OPAMP1 VINN		
OPAMP2 VINP	O	O
OPAMP2 VINN		
OPAMP3 VINP	O	O
OPAMP3 VINN		
OPAMP4 VINP	O	O
OPAMP4 VINN		
ADC1	O	
ADC2		O

19. 4 特殊功能寄存器

19. 4. 1 寄存器列表

VRES 寄存器列表			
名称	偏移地址	类型	描述
VRES_CTRL	0000_H	R/W	电阻分压电路控制寄存器
VRES_VSEL	0004_H	R/W	电阻分压电路电压选择寄存器

19. 4. 2 寄存器描述

19. 4. 2. 1 电阻分压电路控制寄存器 (VRES_CTRL)

电阻分压电路控制寄存器 (VRES_CTRL)																												
偏移地址:0x00																												
复位值:0x0000 0000																												
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																															VRESEN

—	Bits 31-1	—	—
VRESEN	Bit 0	W1	**电阻分压开关** 0: 关闭电阻分压电路。 1: 开启电阻分压电路。

19. 4. 2. 2 电阻分压电路电压选择寄存器 (VRES_VSEL)

电阻分压电路电压选择寄存器 (VRES_VSEL)																												
偏移地址:0x04																												
复位值:0x0010 0010																												
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
											VRES2SEL<4:0>																VRES1SEL<4:0>				

—	Bits 31-21	—	—
VRES2SEL	Bit 20-16	R/W	**电阻分压 V_{RES2} 输出选择** 00000 : 0V(floating) 00001 : (1*VDDA)/32 00010 : (2*VDDA)/32 … 01111 : (15*VDDA)/32 10000 : (16*VDDA)/32 10001 : (17*VDDA)/32 … 11101 : (29*VDDA)/32 11110 : (30*VDDA)/32

			11111 : (31*VDDA)/32
—	Bits 15-5	—	—
VRES1SEL	Bit 4-0	R/W	**电阻分压 V_{RES1} 输出选择** 00000 : 0V(floating) 00001 : (1*VDDA)/32 00010 : (2*VDDA)/32 … 01111 : (15*VDDA)/32 10000 : (16*VDDA)/32 10001 : (17*VDDA)/32 … 11101 : (29*VDDA)/32 11110 : (30*VDDA)/32 11111 : (31*VDDA)/32

第20章 高級控制定时器 16 位 6 通道 (AD16C6T)

20. 1 概述

高级控制定时器(AD16C6T)是一个功能强大、设置灵活的定时器模块，它包含一个 16 位计数器，具有定时、计数、脉冲输入信号测量(输入捕获)、产生特定 PWM 波形(输出比较)与可设置死区时间的互补输出等功能。

20. 2 特性

- 三种 16 位自动重载计数器模式
  - 递增
  - 递减
  - 递增/递减
- 16 位可编程预分频器，可在定时器运行时对计数器工作时钟进行 1 到 65536 之间的任意分频
- 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(边沿和中心对齐模式)
  - 单脉冲输出
- 通道 1～4 支持互补输出，可设置死区时间
- 同步电路用于外部信号控制定时器及内部互联多个定时器
- 重复计数器，用于在给定数目的计数周期后更新定时器寄存器
- 支持两个刹车输入功能，并可设置刹车后定时器输出状态
- 下列事件支持产生中断与 DMA 请求:
  - 更新事件:计数器上溢或下溢，计数器初始化(通过软件或内部与外部触发)
  - 触发事件:计数器开启、停止、初始化或通过内部与外部触发计数
  - 换向事件(COM)
  - 输入捕获
  - 输出比较
- 支持增量(正交)编码及霍尔电路进行定位
- 外部时钟输入触发计数器

20. 3结构图

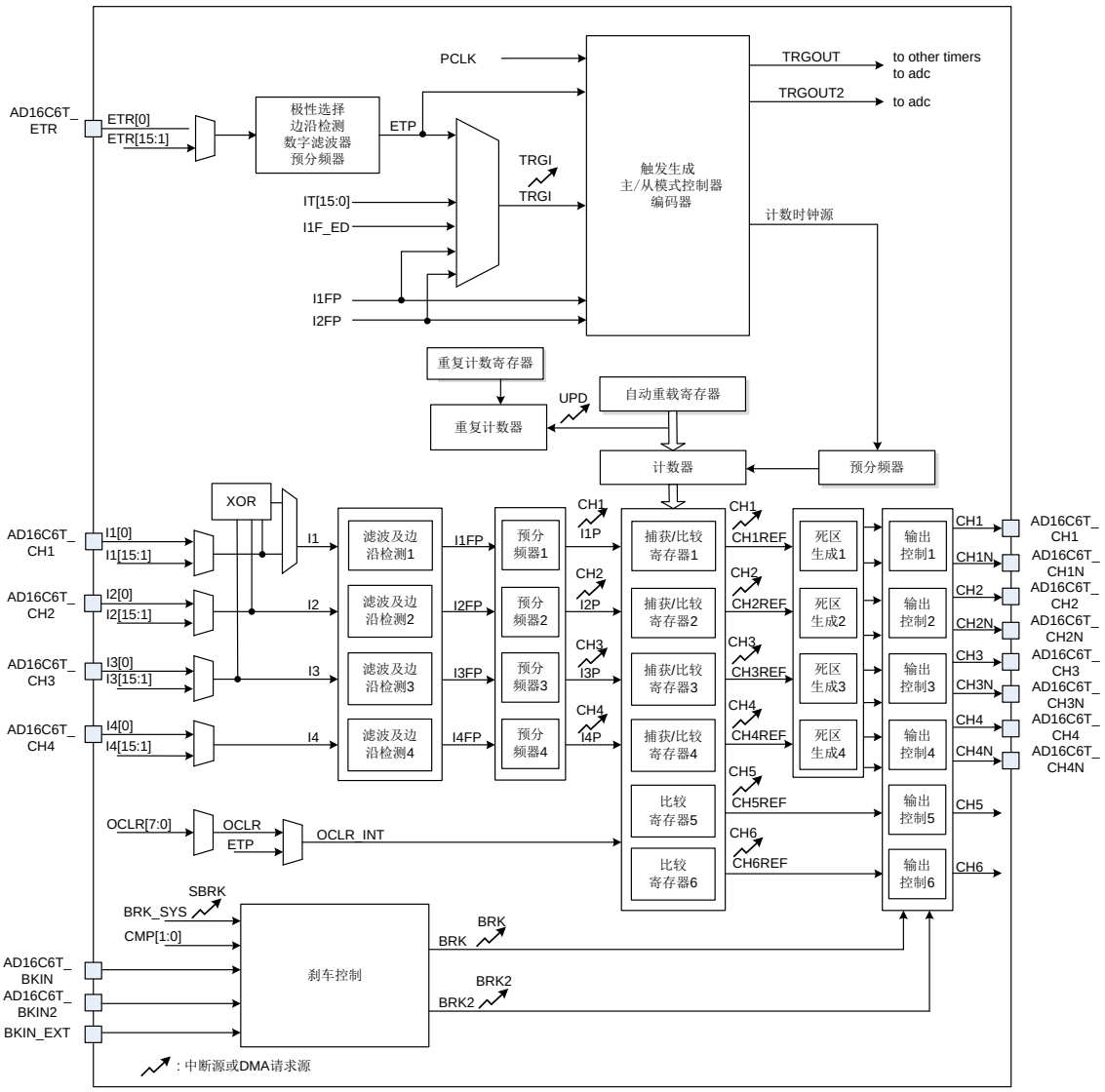


图 21-1　AD16C6T 定时器结构框图

20. 4 引脚选择与内部连接表

20. 4. 1 引脚选择

设置 **AD16C6T_CHRMP** 寄存器，可重映射通道 1 到通道 4 输出/输入引脚(含其互补通道)，如下表 ：

CHnRMP	AD16C6Tn_CH1	AD16C6Tn_CH2	AD16C6Tn_CH3	AD16C6Tn_CH4	AD16C6Tn_CH1N	AD16C6Tn_CH2N	AD16C6Tn_CH3N	AD16C6Tn_CH4N
0000	***AD16C6Tn_CH1**	AD16C6Tn_CH1	AD16C6Tn_CH1	AD16C6Tn_CH1	AD16C6Tn_CH1	AD16C6Tn_CH1	AD16C6Tn_CH1	AD16C6Tn_CH1
0001	AD16C6Tn_CH2	***AD16C6Tn_CH2**	AD16C6Tn_CH2	AD16C6Tn_CH2	AD16C6Tn_CH2	AD16C6Tn_CH2	AD16C6Tn_CH2	AD16C6Tn_CH2
0010	AD16C6Tn_CH3	AD16C6Tn_CH3	***AD16C6Tn_CH3**	AD16C6Tn_CH3	AD16C6Tn_CH3	AD16C6Tn_CH3	AD16C6Tn_CH3	AD16C6Tn_CH3
0011	AD16C6Tn_CH4	AD16C6Tn_CH4	AD16C6Tn_CH4	***AD16C6Tn_CH4**	AD16C6Tn_CH4	AD16C6Tn_CH4	AD16C6Tn_CH4	AD16C6Tn_CH4
0100	AD16C6Tn_CH1N	AD16C6Tn_CH1N	AD16C6Tn_CH1N	AD16C6Tn_CH1N	***AD16C6Tn_CH1N**	AD16C6Tn_CH1N	AD16C6Tn_CH1N	AD16C6Tn_CH1N
0101	AD16C6Tn_CH2N	AD16C6Tn_CH2N	AD16C6Tn_CH2N	AD16C6Tn_CH2N	AD16C6Tn_CH2N	***AD16C6Tn_CH2N**	AD16C6Tn_CH2N	AD16C6Tn_CH2N
0110	AD16C6Tn_CH3N	AD16C6Tn_CH3N	AD16C6Tn_CH3N	AD16C6Tn_CH3N	AD16C6Tn_CH3N	AD16C6Tn_CH3N	***AD16C6Tn_CH3N**	AD16C6Tn_CH3N
0111	AD16C6Tn_CH4N	AD16C6Tn_CH4N	AD16C6Tn_CH4N	AD16C6Tn_CH4N	AD16C6Tn_CH4N	AD16C6Tn_CH4N	AD16C6Tn_CH4N	***AD16C6Tn_CH4N**

表 21-1 AD16C6Tn 引脚输出选择表

注:带*号为寄存器初始设定

CHn_OUT_MUX 为最输出到引脚上的信号，在默认情况下，与实际 IO MUX 连接表上所定义的一致:

- AD16C6Tn_CH1 = AD16C6Tn_CH1 输出
- AD16C6Tn_CH2 = AD16C6Tn_CH2 输出
- AD16C6Tn_CH3 = AD16C6Tn_CH3 输出
- AD16C6Tn_CH4 = AD16C6Tn_CH4 输出
- AD16C6Tn_CH1N = AD16C6Tn_CH1N 输出
- AD16C6Tn_CH2N = AD16C6Tn_CH2N 输出
- AD16C6Tn_CH3N = AD16C6Tn_CH3N 输出
- AD16C6Tn_CH4N = AD16C6Tn_CH4N 输出

由于 CHn 的通道具有输入及输出性，因此真正进入定时器内部的输入信号源，也会经过重映射如下 ：

- AD16C6Tn_CH1 输入 = 设置 CHnRMP 为 0000b 的引脚
- AD16C6Tn_CH2 输入 = 设置 CHnRMP 为 0001b 的引脚

- AD16C6Tn_CH3 输入 = 设置 CHnRMP 为 0010b 的引脚
- AD16C6Tn_CH4 输入 = 设置 CHnRMP 为 0011b 的引脚
- CHnN 互补通道只具备输出，不具输入功能，暂不考虑

一般预设情况下 ：

- AD16C6Tn_CH1 输入 = AD16C6Tn_CH1
- AD16C6Tn_CH2 输入 = AD16C6Tn_CH2
- AD16C6Tn_CH3 输入 = AD16C6Tn_CH3
- AD16C6Tn_CH4 输入 = AD16C6Tn_CH4

需要要特别注意的是，应用上*不允许*同一个信号源映像在两个或以上不同引脚，假设设定成 CH1RMP = CH2RMP = 0011b:

- AD16C6Tn_CH1 = AD16C6Tn_CH4 输出
- AD16C6Tn_CH2 = AD16C6Tn_CH4 输出

在这种情况下，两个输出引脚都会变成 AD16C6Tn_CH4 的输出，而输入端会变成以下情况:

- AD16C6Tn_CH4 输入 = AD16C6Tn_CH1
- AD16C6Tn_CH4 输入 = AD16C6Tn_CH2

即两个引脚输入的信号都接到同一个 **AD16C6Tn_CH4** 输入，由于内部电路会将所有输入信号进行"OR"运算，因此无法在 CH4 上分辨是 CH1 或 CH2 的信号源

20. 4. 2 引脚输入源

设置 **AD16C6T_CHISEL** 寄存器，可选择通道 1 到通道 4 引脚输入来源，如下表 ：

I1SEL	AD16C6T1	AD16C6T2
0000	AD16C6T1_CH1	AD16C6T2_CH1
0001	CMP1	CMP1
0010	CMP2	CMP2
其余	保留	

表 21-2 通道 1 引脚输入来源

I2SEL	AD16C6T1	AD16C6T2
0000	AD16C6T1_CH2	AD16C6T2_CH2
0001	CMP1	CMP1
0010	CMP2	CMP2
其余	保留	

表 21-3 通道 2 引脚输入来源

I3SEL	AD16C6T1	AD16C6T2
0000	AD16C6T1_CH3	AD16C6T2_CH3
0001	CMP1	CMP1
0010	CMP2	CMP2
其余	保留	

表 21-4　通道 3 引脚输入来源

I4SEL	AD16C6T1	AD16C6T2
0000	AD16C6T1_CH4	AD16C6T2_CH4
0001	CMP1	CMP1
0010	CMP2	CMP2
其余	保留	

表 21-5　通道 4 引脚输入来源

设置 **AD16C6T_AFR1** 寄存器的 ETRSEL 位，可选择 ETR 引脚输入来源，如下表:

ETRSEL	AD16C6T1	AD16C6T2
0000	AD16C6T1_ETR	AD16C6T2_ETR
0001	CMP1	CMP1
0010	CMP2	CMP2
1000	ADC1_ANALOG_WD1	ADC2_ANALOG_WD1
1001	ADC1_ANALOG_WD2	ADC2_ANALOG_WD2
1010	ADC1_ANALOG_WD3	ADC2_ANALOG_WD3
其余	保留	

表 21-6　ETR 引脚输入来源

20. 4. 3　内部触发连接表

从定时器	IT0(TSSEL=00000)	IT1(TSSEL=00001)	IT2(TSSEL=00010)	IT3(TSSEL=00011)	IT4(TSSEL=01000)	IT5(TSSEL=01001)	IT6(TSSEL=01010)	IT7(TSSEL=01011)
AD16C6T1	保留	GP32C4T1	GP32C4T2	保留	保留	AD16C6T2	GP16C2T1	GP16C2T2

表 21-7　AD16C6T1 内部触发连接表

从定时器	IT0(TSSEL=00000)	IT1(TSSEL=00001)	IT2(TSSEL=00010)	IT3(TSSEL=00011)	IT4(TSSEL=01000)	IT5(TSSEL=01001)	IT6(TSSEL=01010)	IT7(TSSEL=01011)
AD16C6T2	AD16C6T1	GP32C4T1	GP32C4T2	保留	保留	保留	GP16C2T1	GP16C2T2

表 21-8　AD16C6T2 内部触发连接表

20. 5 功能描述

20. 5. 1 定时单位

定时器包含一个 16 位的计数器(**AD16C6T_COUNT**)，计数时钟由预分频寄存器(**AD16C6T_PRES**)进行分频。计数周期由自动重载计数器(**AD16C6T_AR**)设定。重复计数寄存器则可指定计数周期数目(**AD16C6T_REPAR**)。

自动重载寄存器(**AD16C6T_AR**)是一个可缓冲的寄存器。设置 **AD16C6T_CON1** 寄存器的 ARPEN 位为 0 时，关闭 **AD16C6T_AR** 寄存器缓冲功能，写入 **AD16C6T_AR** 寄存器的值会被立即反应到缓冲寄存器中；而设置 ARPEN 位为 1 时，**AD16C6T_AR** 寄存器具有缓冲功能，只有当产生更新事件(UPD)时，**AD16C6T_AR** 寄存器的重载值才会被更新到缓冲寄存器中。

设置 **AD16C6T_CON1** 寄存器的 DISUE 位为 0 时，计数器递增计数达到上溢值或递减达到下溢值时会产生更新事件(UPD)。另外可以通过 **AD16C6T_SGE** 寄存器的 SGUPD 位为 1 产生软件更新事件。设置 **AD16C6T_CON1** 寄存器的 CNTEN 为 1 时，计数器开始计数。

注:计数器在设置 CNTEN 位为 1 后，在 1 个定时器时钟周期后开始计数。

预分频器可对定时器工作时钟进行 **AD16C6T_PRES** 寄存器数值+1 次分频。由于 **AD16C6T_PRES** 是一个可缓冲寄存器，因此定时器运行时可以对该寄存器进行修改，修改值在下次更新事件(UPD)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。

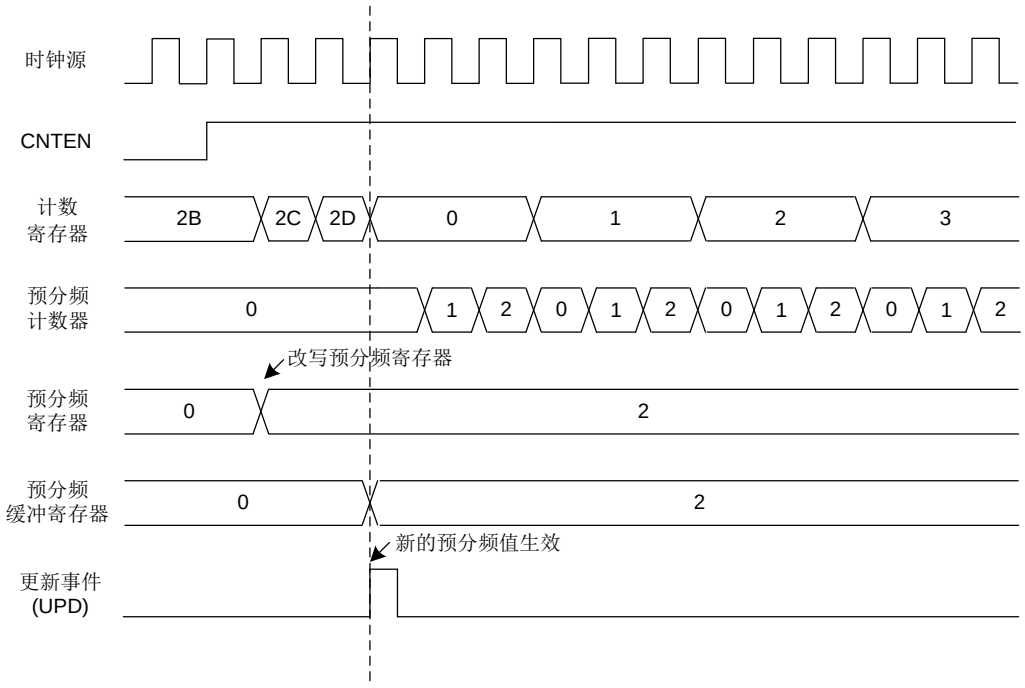


图 21-2 从 1 分频变为 3 分频时的计数时序图

20. 5. 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后产生更新事件。

重复计数器在下列情况下递减:

- 递增模式时计数器的每次上溢。
- 递减模式时计数器的每次下溢。
- 中心对齐模式时，计数器的每次上溢与下溢。中心对齐模式限制了最大重复次数为 32768 个 PWM 周期，每个 PWM 周期内可更新两次占空比。

AD16C6T_REPAR 寄存器是一个可缓冲寄存器。当由软件(设置 **AD16C6T_SGE** 寄存器的 SGUPD 位为 1)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，重复寄存器的数值会立即被更新到重复缓冲寄存器。

中心对齐模式下，当 **AD16C6T_REPAR** 为奇数时，每次更新事件只会发生在上溢或只发生在下溢，取决于何时写 **AD16C6T_REPAR** 寄存器。

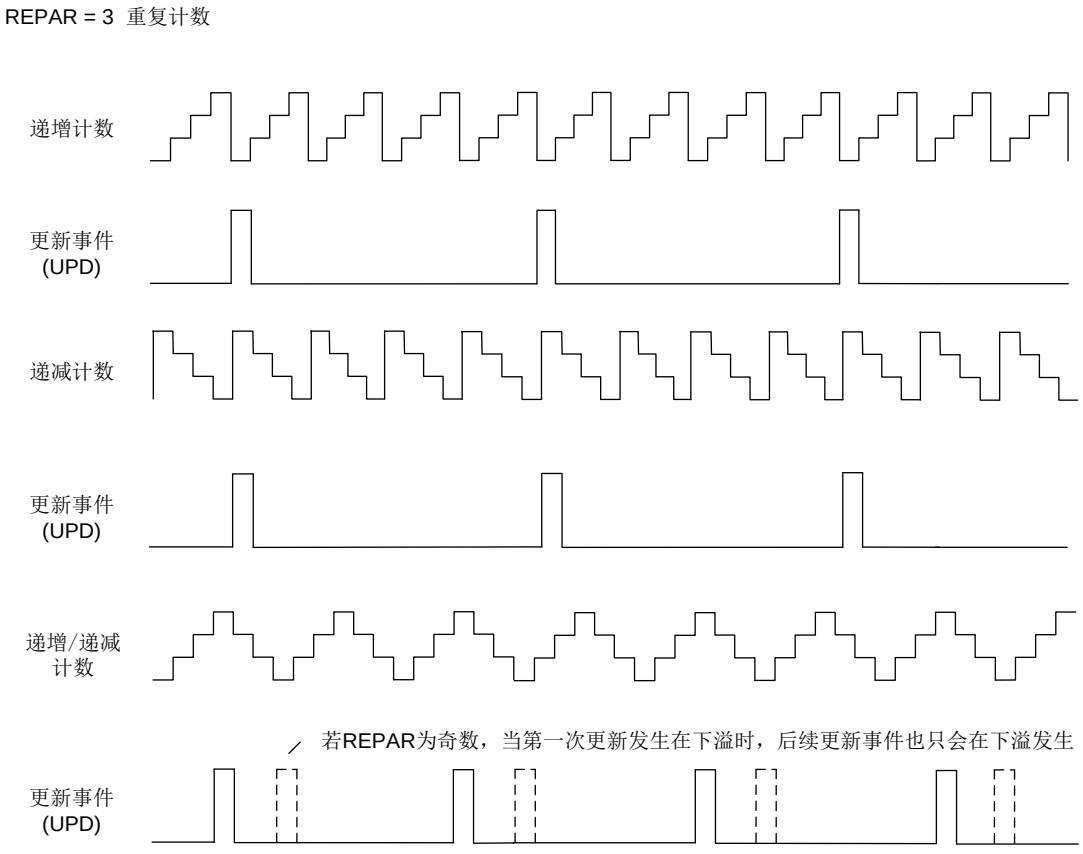


图 21-3 重复计数器工作模式

注:设置 AD16C6T_SGE 寄存器的 SGUPD 位为 1，也可以产生更新事件。

20. 5. 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(TRGI)、外部时钟源 2(ETP)与编码器输入。

20. 5. 3. 1 内部时钟源(INT_CLK)

若从模式控制器被关闭(**AD16C6T_SMCON** 寄存器的 SMODS 位为 0000b)，则 **AD16C6T_CON1** 寄存器的 CNTEN、DIRSEL 位与 **AD16C6T_SGE** 寄存器的 SGUPD 位为控制位，这些位只能软件修改(SGUPD 位除外，仍由硬件自动清除)。一旦设置 CNTEN 位为 1，预分频器就由内部 INT_CLK 提供时钟。

INT_CLK 时钟来源为 APB 时钟(PCLK)，可以参考复位与时钟控制单元(RCU)。

下图给出了正常模式下没有分频控制电路和递增计数的情况。

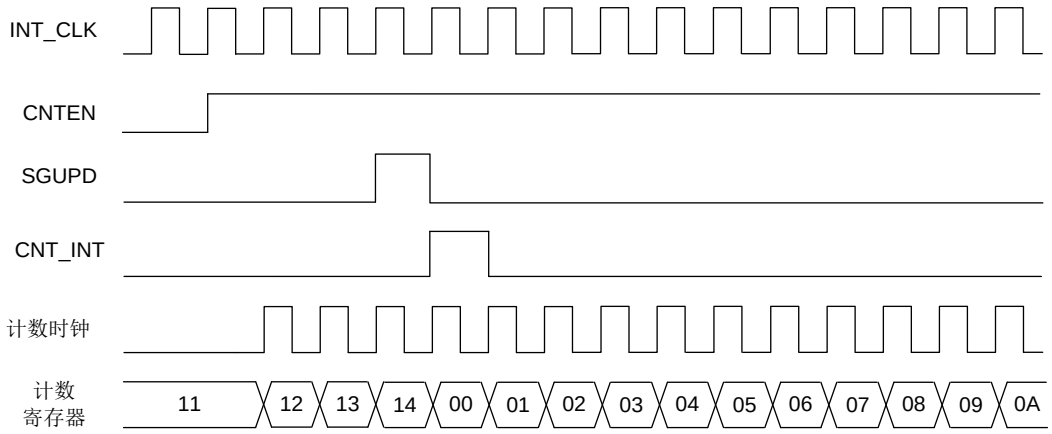


图 21-4 采用内部时钟计数

20. 5. 3. 2 外部时钟源 1

AD16C6T_SMCON 寄存器的 SMODS 位为 0111b 时，可选择外部时钟源 1。计数器可根据选定输入信号的上升沿或下降沿计数。

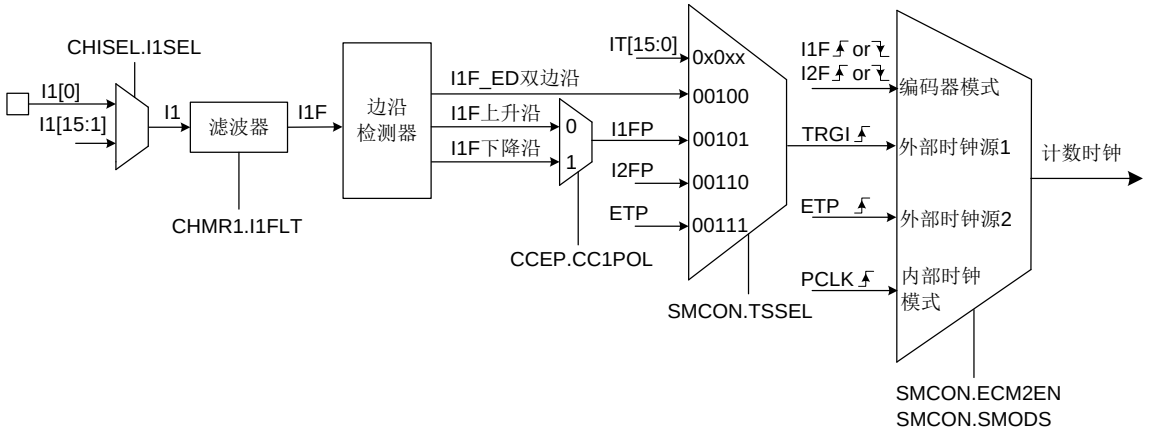


图 21-5 外部时钟连接

设置计数器外部时钟源为 I1 输入，并在 I1 上升沿时计数，步骤如下:

1. 设置 **AD16C6T_CHISEL** 寄存器的 CH1SEL 位为 0000b，选择 I1 由 IO 输入(默认值皆为 IO 输入，后续不再特别描述)。
2. 设置 **AD16C6T_CHMR1** 寄存器 CC1SSEL 位为 01b，让通道 1 为 I1 输入。
3. 设置 **AD16C6T_CHMR1** 寄存器的 I1FLT 位，输入滤波器时间(若没有滤波器需求，维持 I1FLT 位为 0000b)。
4. 设置 **AD16C6T_CCEP** 寄存器的 CC1NPOL 位为 0、CC1POL 位为 0，选择极性为上升沿。
5. 设置 **AD16C6T_SMCON** 寄存器的 TSSEL 位为 00101b，选择外部时钟源为 I1。
6. 设置 **AD16C6T_SMCON** 寄存器的 SMODS 位为 0111b，选择外部时钟模式 1。
7. 设置 **AD16C6T_CON1** 寄存器的 CNTEN 位为 1，开启计数器。

当 I1 上出现一次上升沿时，计数器计数一次且设置 TRGI 标志位为 1。I1 上升沿与实际时钟间的延迟，取决于 I1 输入的同步电路。

设置计数器外部时钟源为 IT6 输入，并在 IT6 上升沿时计数，步骤如下:

1. 设置 **AD16C6T_SMCON** 寄存器的 TSSEL 位为 01010b，选择外部时钟源为 IT6。
2. 设置 **AD16C6T_SMCON** 寄存器的 SMODS 位为 0111b，选择外部时钟模式 1。
3. 设置 **AD16C6T_CON1** 寄存器的 CNTEN 位为 1，开启计数器。

IT6 产生上升沿时，计数器计数一次。

注:以 AD16C6T1 为例，IT6 即为 GP16C2T1 的 TRGOUT，详细可参考内部触发连结表。

20. 5. 3. 3　外部时钟源 2

设置 **AD16C6T_SMCON** 寄存器的 ECM2EN 位为 1 选定外部时钟源 2。

计数器可对外部触发输入 ETR 进行上升沿或下降沿计数。

下图给出了外部触发输入模块的概况。

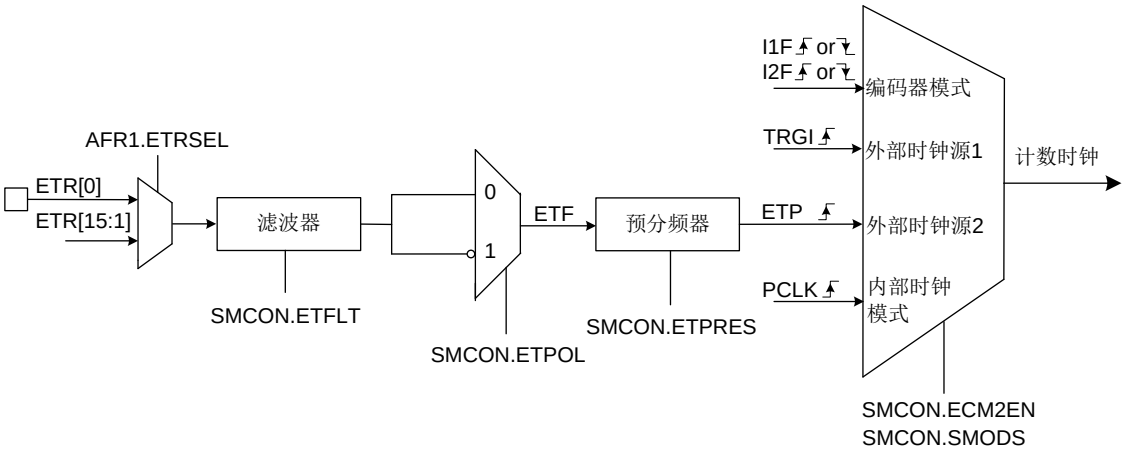


图 21-6　外部触发输入模块

设置计数器为外部时钟源 2，设置过程如下:

1. 设置 AD16C6T_SMCON 寄存器的 ETFLT 位，输入滤波器时间(若没有滤波器需求，维持 ETFLT 位为 0000b)预装载。
2. 设置 **AD16C6T_SMCON** 寄存器的 ETPRES 位为 0，关闭外部触发时钟预分频器。
3. 设置 **AD16C6T_SMCON** 寄存器的 ETPOL 位，检测 ETR 引脚上升沿或下降沿。
4. 设置 **AD16C6T_SMCON** 寄存器的 ECM2EN 位为 1，开启外部时钟模式 2。
5. 设置 **AD16C6T_CON1** 寄存器的 CNTEN 位为 1，开启计数器。

计数器在每个 ETR 上升沿计数一次。

20. 5. 4 计数模式

20. 5. 4. 1 递增计数模式

设置 **AD16C6T_CON1** 寄存器的 DIRSEL 位为 0 时，定时器设置为递增模式，计数器从 0 开始递增，直至 **AD16C6T_AR** 寄存器数值；然后从 0 重新开始计数并产生一个更新事件(UPD)。设置 **AD16C6T_REPAR** 寄存器不为0时，则在 **AD16C6T_REPAR**+1 次计数后产生更新事件。

设置 **AD16C6T_SGE** 寄存器的 SGUPD 位为 1(通过软件或使用从模式控制器)同样会产生更新事件，并让计数器和预分频器都重新从 0 开始计数。

通过软件设置 **AD16C6T_CON1** 寄存器中的 DISUE 位为 1 可关闭更新事件(UPD)产生。可以避免在写入预装载寄存器数值时产生更新事件(UPD)更新缓冲寄存器。在设置 DISUE 位为 0 之前都不会产生更新事件(UPD)。

此外，**AD16C6T_CON1** 寄存器中的 UERSEL 位为 1 时，设置 SGUPD 位为 1 会产生更新事件(UPD)，但不会更新更新标志位(**AD16C6T_RIF** 寄存器的 UPD 位)，也不会产生中断或 DMA 请求。在这个配置下，发生捕获事件时将计数器清零，将不会同时产生更新中断和捕获中断。

发生更新事件(UPD)时，所有预装载寄存器会更新到缓冲寄存器:

- 更新 **AD16C6T_REPAR** 寄存器数值到缓冲寄存器。
- 更新 **AD16C6T_AR** 寄存器数值到缓冲寄存器。
- 更新 **AD16C6T_PRES** 寄存器数值到缓冲寄存器。

下图为设置 **AD16C6T_AR** 寄存器为 16h，预分频设为 2 分频时的计数器时序。

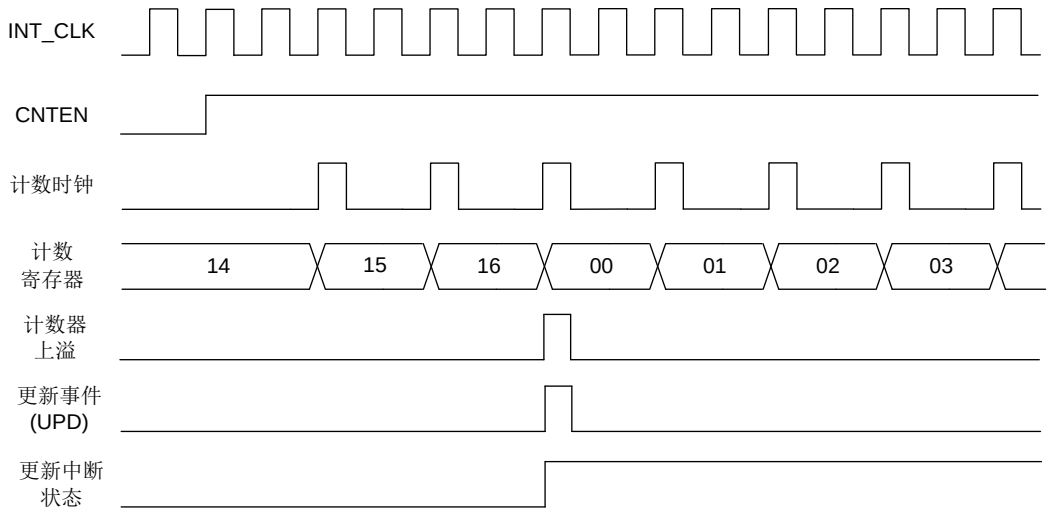


图 21-7 计数器递增计数时序图

下图为设置 **AD16C6T_AR** 寄存器从 F5h 改成 16h，分别在 ARPEN 为 0 或 1 时的计数器时序。

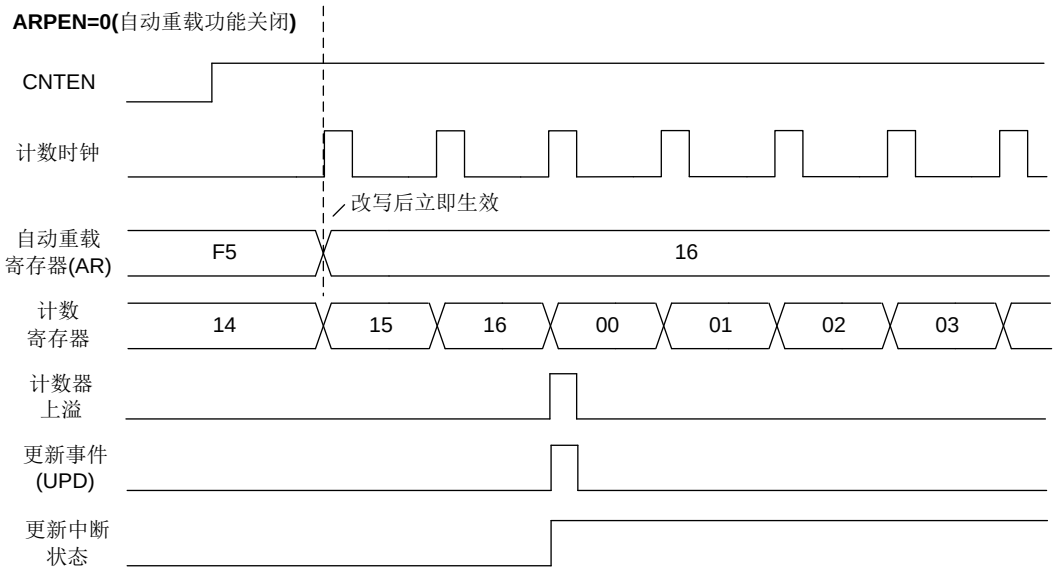


图 21-8　设置 ARPEN 位为 0 时计数器时序图

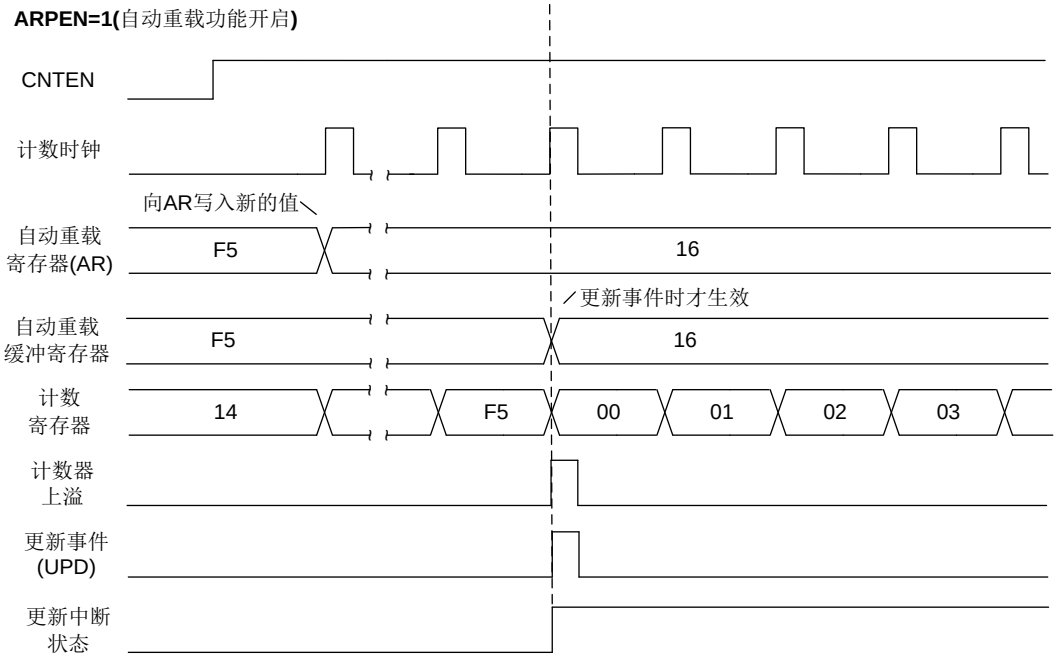


图 21-9　设置 ARPEN 位为 1 时计数器时序图

20. 5. 4. 2 递减计数模式

设置 **AD16C6T_CON1** 寄存器的 DIRSEL 位为 1 时，定时器设置为递减模式，计数器从 **AD16C6T_AR** 寄存器数值开始递减至 0;然后从 **AD16C6T_AR** 寄存器数值重新递减并产生更新事件(UPD)。设置 **AD16C6T_REPAR** 寄存器不为 0 时，则在 **AD16C6T_REPAR** +1 次后产生更新事件。

设置 **AD16C6T_SGE** 寄存器的 SGUPD 位为 1(通过软件或使用从模式控制器)同样会产生更新事件，并让计数器从自动重载值开始计数，而预分频器从 0 开始计数。

通过软件设置 **AD16C6T_CON1** 寄存器中的 DISUE 位为 1 可关闭更新事件(UPD)产生。可以避免在写入预装载寄存器数值时产生更新事件(UPD)更新缓冲寄存器。在设置 DISUE 位为 0 之前都不会产生更新事件(UPD)。

此外，**AD16C6T_CON1** 寄存器中的 UERSEL 位为 1 时，设置 SGUPD 位为 1 会产生更新事件(UPD)，但不会更新更新标志位(**AD16C6T_RIF** 寄存器的 UPD 位)，也不会产生中断或 DMA 请求。在这个配置下，发生捕获事件时将计数器清零，将不会同时产生更新中断和捕获中断。

发生更新事件(UPD)时，所有预装载寄存器会更新到缓冲寄存器。

下图为设置 **AD16C6T_AR** 寄存器为 27h，预分频设为 1 分频时的计数器时序。

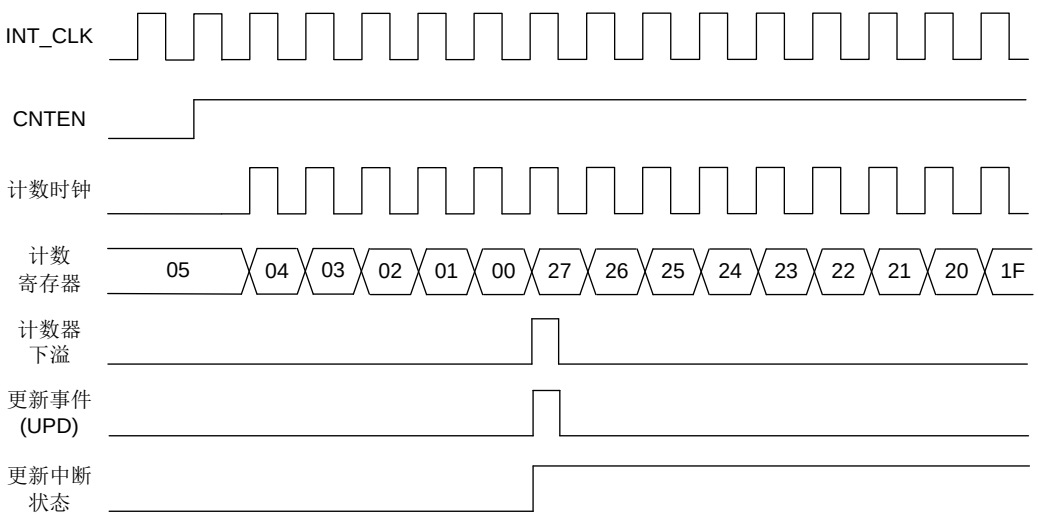


图 21-10 计数器递减计数时序图

20. 5. 4. 3 中心对齐模式

若将 **AD16C6T_CON1** 寄存器的 CMSEL 位数值设置为非 00b 值，则定时器会运作于中心对齐模式。在中心对齐模式下，计数器会从 0 开始递增至 **AD16C6T_AR** 寄存器的数值减 1，同时产生更新事件(UPD)；接着计数器会从 **AD16C6T_AR** 寄存器的数值递减至 1，并再次产生更新事件；最后计数器会从 0 重新开始计数，进行循环计数。

将通道配置为输出模式时，若要设置输出比较中断标志为 1，则需考虑定时器工作在哪种中心

对齐模式下:

- 中心对齐模式 1(设置 CMSEL 位为 01b):输出比较中断标志只产生在计数器递减计数时。
- 中心对齐模式 2(设置 CMSEL 位为 10b):输出比较中断标志只产生在计数器递增计数时。
- 中心对齐模式 3(设置 CMSEL 位为 11b):输出比较中断标志在计数器递增递减计数时都会产生。

在中心对齐模式下，**AD16C6T_CON1** 寄存器的 DIRSEL 位无法进行写入操作，该位由硬件自动更新，指示当前计数方向。

计数上溢、下溢或者设置 **AD16C6T_SGE** 寄存器的 SGUPD 位为 1(通过软件或使用从模式控制器)都会产生更新事件，而通过软件或使用从模式控制器的方式，会让计数器和预分频器都重新从 0 开始递增计数。

通过软件设置 **AD16C6T_CON1** 寄存器中的 DISUE 位为 1，可以关闭更新事件(UPD)的产生，这可以避免在写入预装载寄存器数值时，产生更新事件(UPD)并更新缓冲寄存器。在设置 DISUE 位为 0 之前，不会产生更新事件(UPD)。

此外，当设置 **AD16C6T_CON1** 寄存器中的 UERSEL 位为 1 时，设置 SGUPD 位为 1 会产生更新事件(UPD)，但不会更新更新标志位(AD16C6T_RIF 寄存器的 UPD 位)，也不会产生中断或 DMA 请求。在这种配置下，当发生捕获事件时，计数器将被清零，并且不会同时产生更新中断和捕获中断。

当发生更新事件(UPD)时，所有预装载寄存器会被更新到缓冲寄存器中。

注:若发生更新事件是因为计数器的上溢，则在计数器重载之前，自动重载缓冲寄存器就已经完成了更新。因此，下一个计数周期将使用新的预装载值（即计数器重载的新 AR 值）。

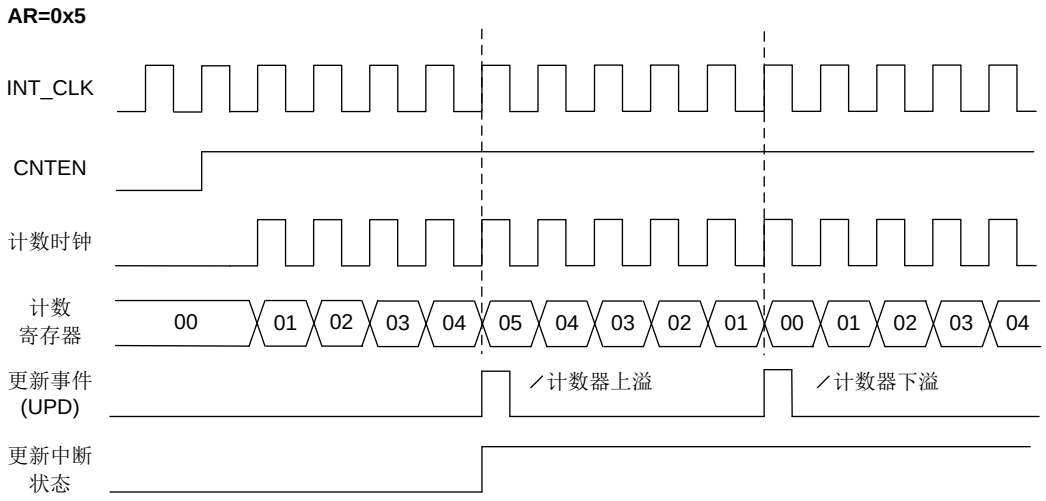


图 21-11　计数器递增减计数时序图

20. 5. 5 外部触发输入

定时器具有一个外部触发输入 ETR，主要作用于:

- 外部时钟模式。
- 从模式控制器的触发输入源。
- PWM 输出复位的触发输入源。

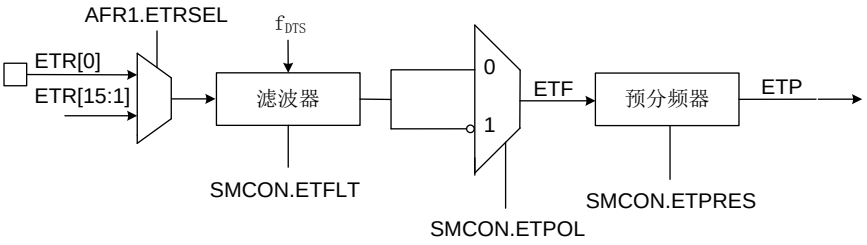


图 21-12 外部触发输入结构图

设置 **AD16C6T_AFR1** 寄存器的 ETRSEL 位，可选择 ETR 输入源为:

- ETR 输入引脚(默认选择)。
- 比较器输出(CMP1、CMP2)。
- 模拟看门狗(ADCn_AWDn_OUT)。

详细设定可参考引脚输入源章节。

20. 5. 6 捕获或比较通道

每个捕获或比较通道都包含一个捕获或比较预装载寄存器(包含缓冲寄存器)、一个输入捕获电路和一个输出比较电路。读写操作都是通过预装载寄存器进行的。在捕获模式下，实际的捕获是在缓冲寄存器中完成的，然后缓冲寄存器中的值被复制到预装载寄存器中。在比较模式下，预装载寄存器的值被复制到缓冲寄存器中，并与计数器进行比较。

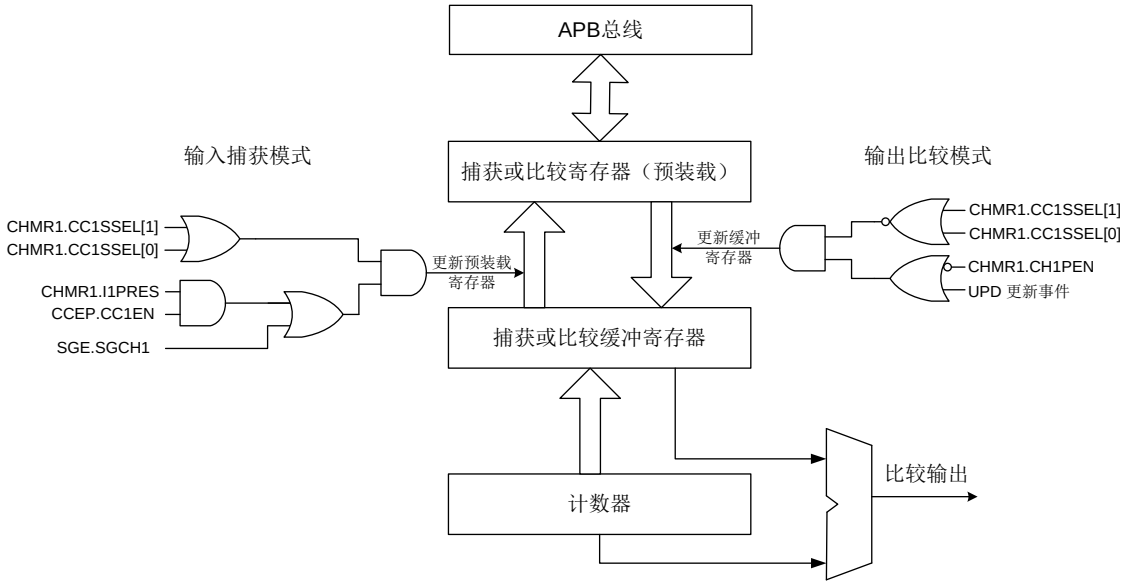


图 21-13　捕获或比较通道结构图(通道 1 为例)

20. 5. 6. 1　　输入捕获电路

输入捕获电路会对 In 输入端的信号进行采样，经过数字滤波器后产生 InF 信号，接着通过边沿检测器产生 InF 边沿信号，包括上升沿、下降沿以及双边沿。最后，根据 **AD16C6T_CCEP** 寄存器中的 CCnNPOL/CCnPOL 极性选择位，生成 InFP 信号。这个信号可以输出到从模式控制器的触发输入(TRGI)，或作为捕获输入源(InP)之一，而且还需要通过预分频器进行预处理。

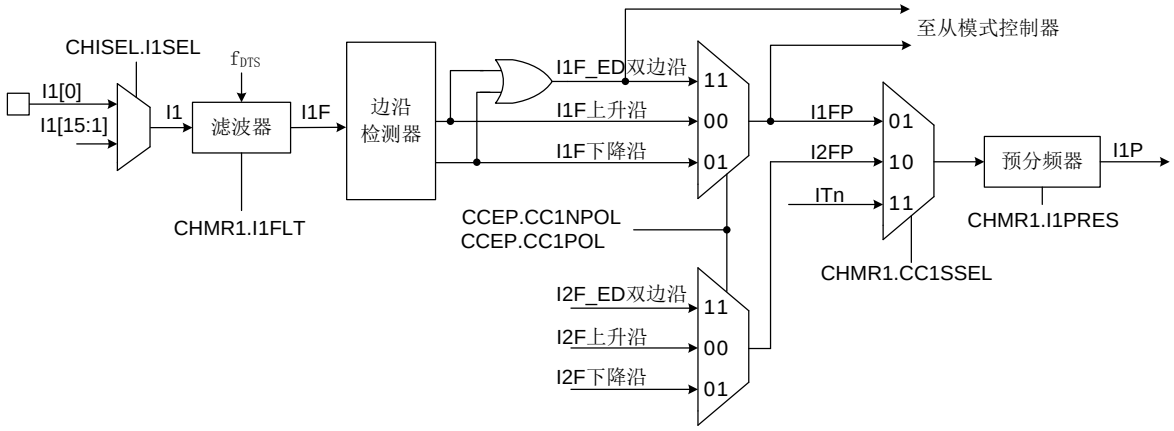


图 21-14　输入捕获电路(通道 1 为例)

20. 5. 6. 2　　输出比较电路

输出部分会根据 **AD16C6T_CHMRn** 寄存器中 CHnMOD 位的配置，产生一个输出比较参考信号 CHnREF(高电平有效)。最终输出信号的极性则由 **AD16C6T_CCEP** 寄存器中的 CCnPOL/CCnNPOL 位所决定。

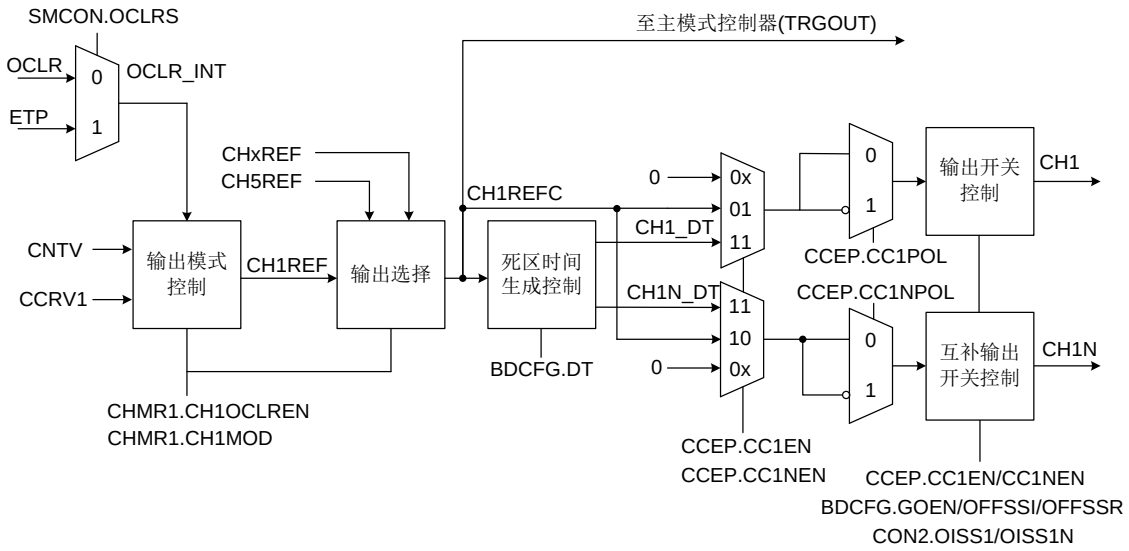


图 21-15　输出比较电路(通道 1 为例，通道 2/3/4 架构相同)

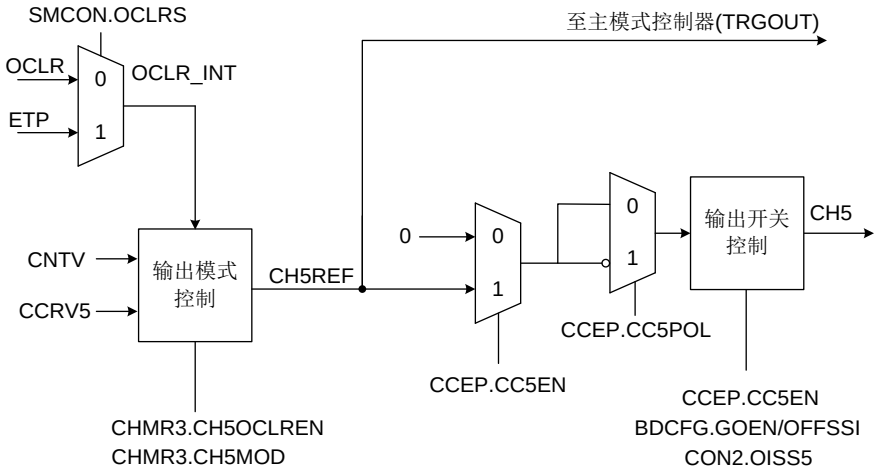


图 21-16　输出比较电路(通道 5 为例，通道 6 架构相同)

20. 5. 7　输入捕获模式

在输入捕获模式下，当 In 上检测到有效边沿变化时，计数器数值就会被锁存到捕获或比较寄存器(**AD16C6T_CCVALn**)中。此时，**AD16C6T_RIF** 寄存器中相应的 CHn 标志位会被设置为 1，同时触发中断或 DMA 请求(如果已开启)。

若相应的 CHn 标志位已经为 1，则当再次发生捕获事件时，相应的过捕获 CHnOV 标志位也会被设定为 1，表示曾经发生过捕获事件。

藉由软件将 **AD16C6T_ICR** 寄存器中的 CHn 位与 CHnOV 位设定为 1，可清除 **AD16C6T_RIF** 寄存器中相应的 CHn 与 CHnOV 标志位。

以下为以 I1 输入上升沿作为捕获输入时的流程:

1. 设置 **AD16C6T_CHMR1** 寄存器的 CC1SSEL 位为 01b，选择 I1 为有效输入端。只要 CC1SSEL 不为 00b，通道就会被设置成输入，且 **AD16C6T_CCVAL1** 寄存器为只读。
2. 设置 **AD16C6T_CHMR1** 寄存器的 I1FLT 位为 0011b，选择输入滤波器的持续时间，当 I1 检测到新的电平时，会进行连续 8 次采样，确认电平变化的有效性。
3. 设置 **AD16C6T_CCEP** 寄存器的 CC1NPOL 位为 0、CC1POL 位为 0，选择 I1 通道上升沿有效。
4. 设置 **AD16C6T_CHMR1** 寄存器的 I1PRES 位为 00b，关闭捕获预分频器，让每次有效上升沿皆执行捕获操作。
5. 如有需要，设置 **AD16C6T_IER** 寄存器的 CH1 位为 1，开启中断请求。设置 **AD16C6T_DMAEN** 寄存器的 CH1 位为 1，开启 DMA 请求。
6. 设置 **AD16C6T_CCEP** 寄存器的 CC1EN 位为 1，开启捕获计数器。

当发生输入捕获时:

1. **AD16C6T_CCVAL1** 寄存器获取计数器当前的值。
2. **AD16C6T_RIF** 寄存器的 CH1 位被设置。如果至少两个连续捕获发生，且标志位未被清除，则 CH1OV 位也被设置。

3. 中断的产生取决于 **AD16C6T_IER** 寄存器的 CH1 位。
4. DMA 请求的产生取决于 **AD16C6T_DMAEN** 寄存器的 CH1 位。

为了处理捕获溢出，建议在读取过捕获标志位之前先读取数据。这是为了避免在读取标志位和读取数据之间发生捕获溢出，从而丢失捕获讯息。

注:捕获中断请求可由软件设置 AD16C6T_SGE 寄存器的 SGCHn 位产生。

20. 5. 8 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. 设置 **AD16C6T_CHMR1** 寄存器的 CC1SSEL 位为 01b，通道 1 选择 I1 为有效输入端。
2. 设置 **AD16C6T_CCEP** 寄存器的 CC1NPOL 位为 0、CC1POL 位为 0，通道 1 选择 I1 上升沿有效，用于捕获数据到 **AD16C6T_CCVAL1** 寄存器。
3. 设置 **AD16C6T_CHMR1** 寄存器的 CC2SSEL 位为 10b，通道 2 同样选择 I1 为有效输入端。
4. 设置 **AD16C6T_CCEP** 寄存器的 CC2NPOL 位为 0、CC2POL 位为 1，通道 2 选择 I1 下降沿有效，用于捕获数据到 **AD16C6T_CCVAL2** 寄存器。
5. 设置 **AD16C6T_SMCON** 寄存器的 TSSEL1 位与 TSSEL2 位为 00101b，选择 I1FP 信号为有效的触发输入(TRGI)。
6. 设置 **AD16C6T_SMCON** 寄存器的 SMODS 位为 0100b,选择从模式控制器为复位模式，让每次有效触发输入(TRGI)初始化计数器。
7. 设置 **AD16C6T_CCEP** 寄存器的 CC1EN 位和 CC2EN 位为 1，开启捕获。
8. 设置 **AD16C6T_CON1** 寄存器的 CNTEN 位为 1，开启计数器。

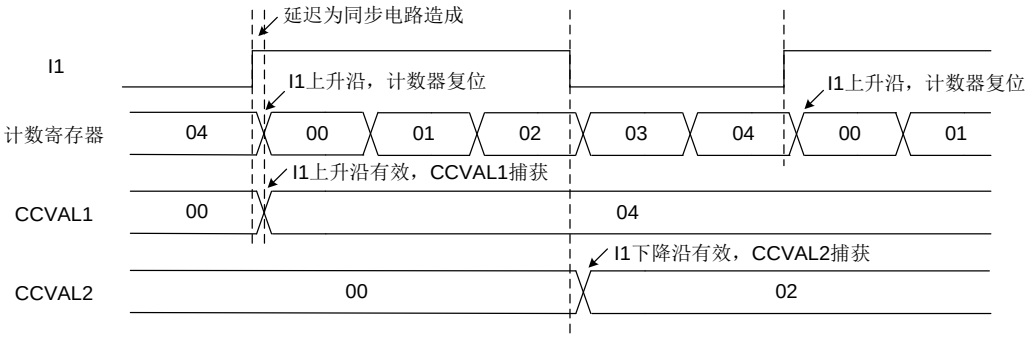


图 21-17 PWM 输入模式时序

- **AD16C6T_CCVAL1** 寄存器内的值为 PWM 周期(两次上升沿之间的时间),此范例中 PWM 周期为 5 个计数单位。
- **AD16C6T_CCVAL2** 寄存器内的值为 PWM 脉冲宽度(上升沿到下降沿之间的时间)，此范例中 PWM 脉冲宽度为 3 个计数单位，可推算其占空比为 60%。

注:计数单位取决于时钟频率以及预分频设定值

20. 5. 9 PWM输出模式

脉宽调制(PWM)模式可以产生一个由 **AD16C6T_AR** 寄存器设置输出频率，由 **AD16C6T_CCVALn** 寄存器设置占空比的信号。若要开启相应的预装载寄存器，需将 **AD16C6T_CHMRn** 寄存器的 CHnPEN 位设置为 1，并将 **AD16C6T_CON1** 寄存器的 ARPEN 位设置为 1 以开启自动重载功能。

在开启预装载和自动重载功能后，只有当更新事件发生时，才会将预装载寄存器的值写入到缓冲寄存器中。因此，在开始计数前，必须通过将 **AD16C6T_SGE** 寄存器的 SGUPD 位设置为 1，以初始化所有的寄存器。

CHn 的极性可以通过 **AD16C6T_CCEP** 寄存器的 CCnPOL 位设置，有效电平可以设置为高电平或低电平。CHn 的输出由 **AD16C6T_CCEP** 寄存器的 CCnEN 位控制。

在 PWM 模式 1 或 2 中，**AD16C6T_COUNT** 会持续与 **AD16C6T_CCVALn** 寄存器的数值比较，以确定 **AD16C6T_CCVALn** ≤ **AD16C6T_COUNT** 或 **AD16C6T_CCVALn** ≥ **AD16C6T_COUNT**(取决于计数器的计数方向)。

定时器产生 PWM 波形时可以是边沿对齐或中心对齐，取决于 **AD16C6T_CON1** 寄存器的 CMSEL 位。

20. 5. 9. 1 PWM边沿对齐模式

◆ 递增计数设置

设置 **AD16C6T_CON1** 寄存器的 DIRSEL 位为 0 时，计数器递增计数。

以 CH1 输出 PWM 模式 1 为例，相关配置流程如下:

1. 设置 **AD16C6T_CHMR1** 寄存器的 CH1MOD 位为 00110b，选择 PWM 模式 1。
2. 设置 **AD16C6T_CCEP** 寄存器的 CC1POL 位为 0，选择 CH1 通道输出为高电平有效。
3. 设置 **AD16C6T_CCVAL1** 寄存器的 CCRV1 位为 04h，当计数器数到 4 时，PWM 输出低电平。
4. 设置 **AD16C6T_AR** 寄存器的 ARV 位为 08h，当计数器递增到 8 后重载。
5. 设置 **AD16C6T_CON1** 寄存器的 CNTEN 位为 1，开启计数器。

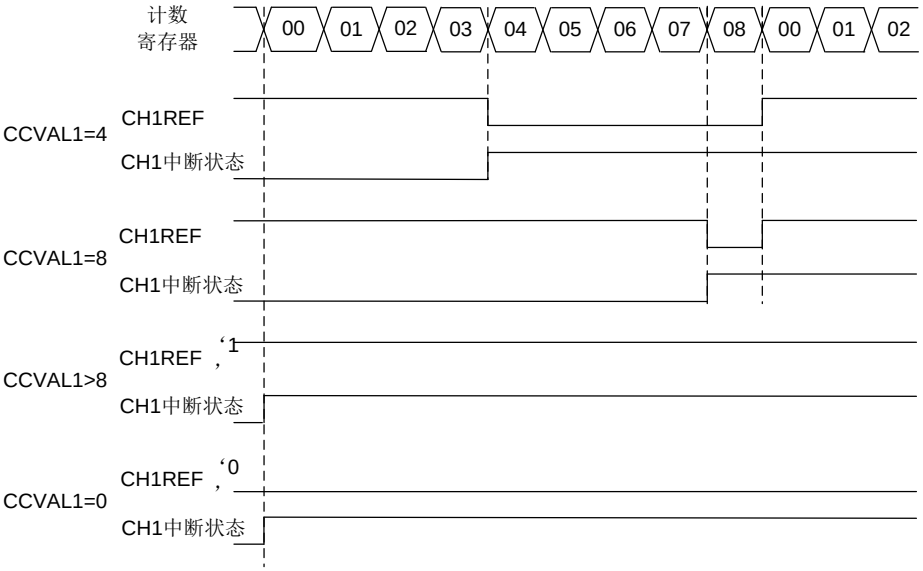


图 21-18 边沿对齐递增计数 PWM 波形(AR=8)

◆ **AD16C6T_COUNT** < **AD16C6T_CCVAL1** 时，CH1 为高电平。

◆ **AD16C6T_COUNT** ≥ **AD16C6T_CCVAL1** 时，CH1 为低电平。

通过改变 **AD16C6T_CCVAL1** 寄存器的 CCRV1 位，可以改变 PWM 信号的占空比。如果将 **AD16C6T_CCVAL1** 设置为大于 **AD16C6T_AR** 的值，则 CH1 将永远输出高电平，并在计数器上溢时产生比较匹配事件。如果将 **AD16C6T_COUNT** 设置为 0，则 CH1 将永远输出低电平。

◆ 递减计数设置

设置 **AD16C6T_CON1** 寄存器的 DIRSEL 位为 1 时，计数器递减计数

下图以 CH1 输出 PWM 模式 1 为例，相关配置流程如下:

1. 设置 **AD16C6T_CON1** 寄存器的 DIRSEL 位为 1，计数器递减计数。
2. 设置 **AD16C6T_AR** 寄存器的 ARV 位为 08h，当计数器递减到 0 后重载。
3. 设置 **AD16C6T_SGE** 寄存器的 SGUPD 位为 1，软件触发更新事件，将 ARV 重载到 **AD16C6T_COUNT** 寄存器中。

4. 设置 **AD16C6T_CHMR1** 寄存器的 CH1MOD 位为 00110b，选择 PWM 模式 1。
5. 设置 **AD16C6T_CCEP** 寄存器的 CC1POL 位为 0，选择 CH1 通道输出为高电平有效。
6. 设置 **AD16C6T_CCVAL1** 寄存器的 CCRV1 位为 04h，当计数器数到 4 时，PWM 输出高电平。
7. 设置 **AD16C6T_CON1** 寄存器的 CNTEN 位为 1，开启计数器。

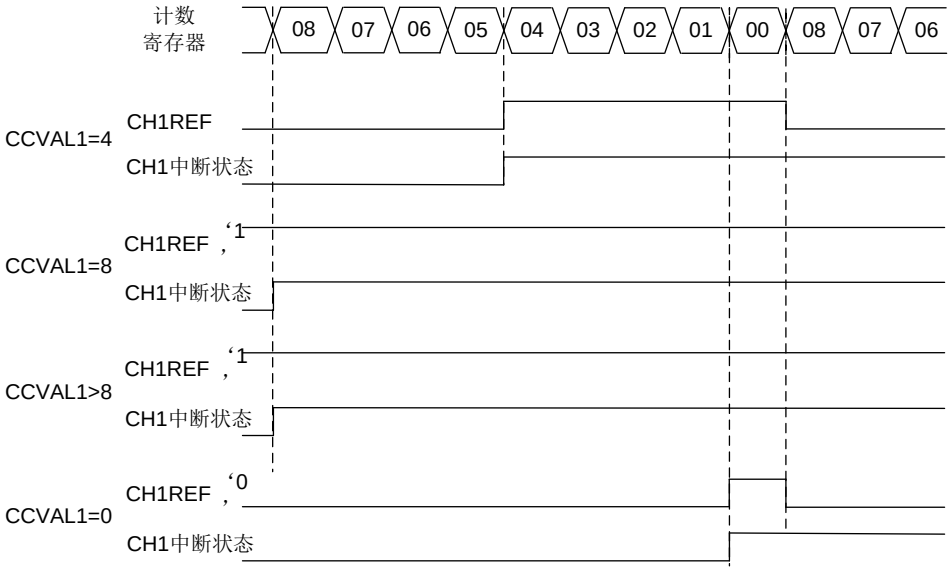


图 21-19　边沿对齐递减计数 PWM 波形(AR=8)

- **AD16C6T_COUNT** ≤ **AD16C6T_CCVAL1** 时，CH1 为高电平。
- **AD16C6T_COUNT** > **AD16C6T_CCVAL1** 時，CH1 为低电平。

其中比较特别的是，若设定 **AD16C6T_CCVAL1**≧**AD16C6T_AR** 时，CH1 会永远输出高电平(并在下溢时产生比较匹配事件)。此模式下不可能产生永远低电平的 PWM 波形。

20. 5. 9. 2 PWM中心对齐模式

若设置 **AD16C6T_CON1** 寄存器的 CMSEL 位不为 00b，就启用了中心对齐模式。根据 CMSEL 位的设置，计数器可以只在递增、只在递减或者在递增递减同时计数时，设置 AD16C6T_RIF 寄存器的比较标志位为 1。

在中心对齐模式下，计数器会在到达上溢或下溢时自动反向计数，而不需要额外的软件控制。DIRSEL 位控制计数方向的选择由硬件更新，软件无法修改。

下图为中心对齐模式下，CH1 输出 PWM 模式 1 为例，AR 位为 08h:

图 21-20 中心对齐 PWM 波形(AR=08h)

中心对齐模式的使用技巧:

- 进入中心对齐模式后，计数器会根据原本的递增或递减设置进行计数。递增或递减的方向取决于 **AD16C6T_CON1** 寄存器的 DIRSEL 位原本的设定。需要注意的是，软件不得同时修改 DIRSEL 和 CMSEL 位。
- 计数器在中心对齐模式下运行时，不建议对 **AD16C6T_COUNT** 执行写入操作。假设在递增计数的情况下，向计数器写入数值大于自动重载值(**AD16C6T_COUNT>AD16C6T_AR**)，计数方向不会更新，会持续计数下去。
- 使用中心对齐模式最安全的方式是在计数器开始计数前通过软件产生更新事件(设置 **AD16C6T_SGE** 寄存器的 SGUPD 位为 1)，并在计数器运行过程中不对

AD16C6T_COUNT 寄存器进行写值。这样可以保证计数器的计数方向正确，避免出现错误。

20. 5. 9. 3 抖动模式

通过在 **AD16C6T_CON1** 寄存器中启用 DITHEN 位，可以启用抖动模式，从而增加 PWM 模式的有效分辨率。这适用于 **AD16C6T_CCVALn**(用于占空比分辨率增加)和 **AD16C6T_AR**(用于 PWM 频率分辨率增加)两种情况。

操作原理是在 16 个连续的 PWM 周期中，通过预定义的模式，微调实际的 CCVALn(或 AR)值(添加或不添加一个定时器时钟周期)。这使得平均占空比或 PWM 周期的分辨率增加了 16 倍。下图显示了抖动模式应用于 4 个连续 PWM 周期的原理。

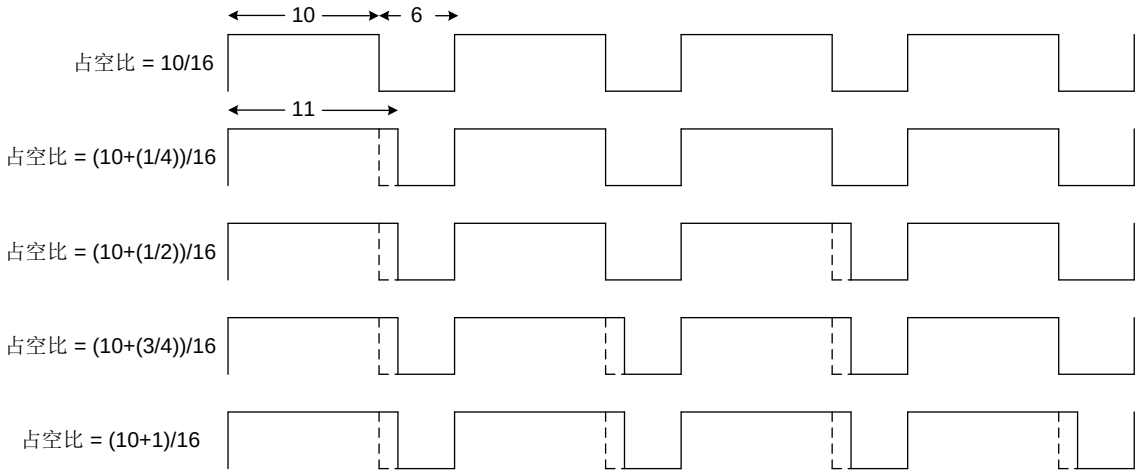


图 21-21 抖动模式原理

当开启抖动模式，**AD16C6T_CCVALn** 与 **AD16C6T_AR** 寄存器内容会被重定义如下:

- 最高 16 位[19:4]为原寄存器左移 4 位后的值，等校于 x16 倍。
- 最低 4 位[3:0]为用来提高分辨率的控制位。

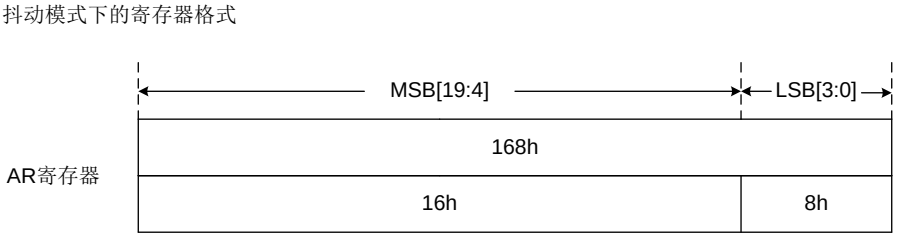


图 21-22 抖动模式下寄存器格式(以 AR 为例)

- AR[19:4] = 16h:16 个 PWM 周期的 AR 值。
- AR[3:0] = 8h:16 个 PWM 周期内，共延长了 8 个定时器时钟单位(被平均在这 16 个 PWM 周期内)。

若尚未开启抖动模式(DITHEN = 0)前的 AR 寄存器值为 16h，在开启抖动模式后(DITHEN = 1)，其中的值会被定时器自动改为 160h。

注:AD16C6T_CCVALn 与 AD16C6T_AR 寄存器在抖动模式下的最大值为 0xFFFEF。

下图为递增计数模式下，不同 LSB 设定值在连续 16 个 PWM 周期内的变化情形:

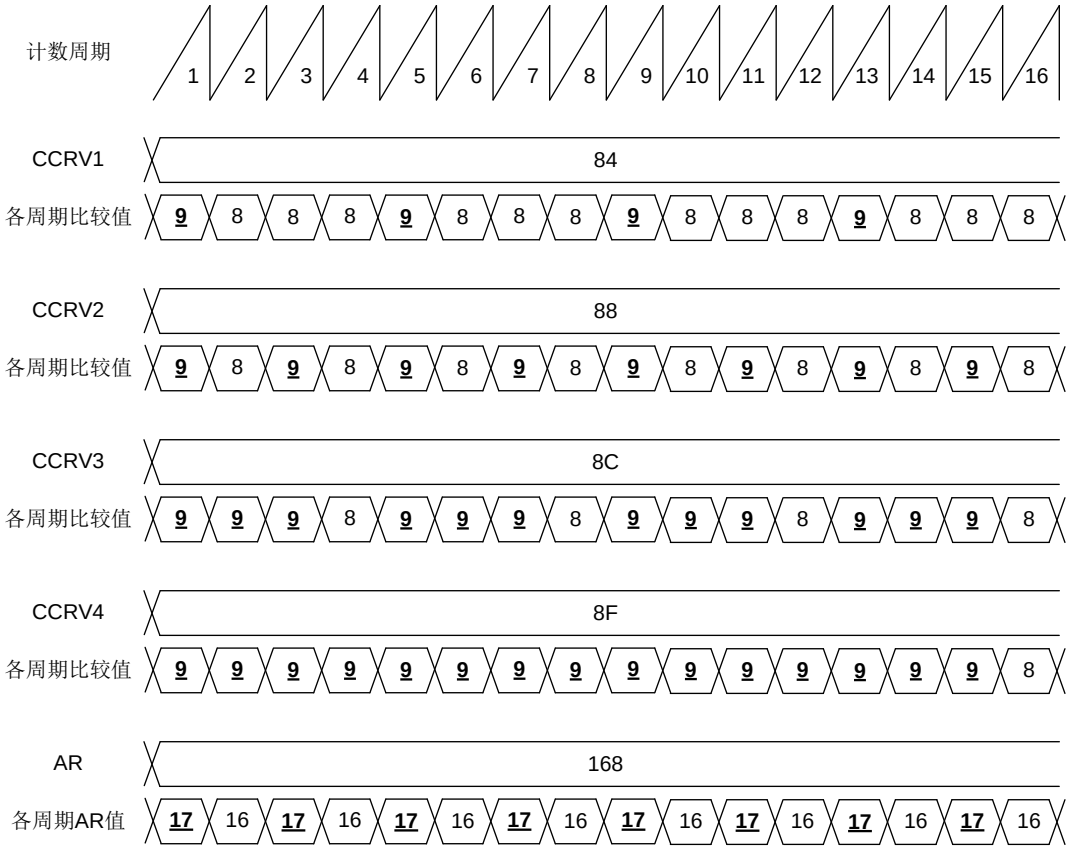


图 21-23　抖动模式下不同 LSB 的寄存器值

CCRVn 与 AR 值在 16 个周期内的所有变化表整理如下:

LSB	PWM 周期															
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
0000	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0001	+1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0010	+1	-	-	-	-	-	-	-	+1	-	-	-	-	-	-	-
0011	+1	-	-	-	+1	-	-	-	+1	-	-	-	-	-	-	-
0100	+1	-	-	-	+1	-	-	-	+1	-	-	-	+1	-	-	-
0101	+1	-	+1	-	+1	-	-	-	+1	-	-	-	+1	-	-	-
0110	+1	-	+1	-	+1	-	-	-	+1	-	+1	-	+1	-	-	-
0111	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	-	-
1000	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-

LSB	PWM 周期															
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1001	+1	+1	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-
1010	+1	+1	+1	-	+1	-	+1	-	+1	+1	+1	-	+1	-	+1	-
1011	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-	+1	-	+1	-
1100	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-
1101	+1	+1	+1	+1	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-
1110	+1	+1	+1	+1	+1	+1	+1	-	+1	+1	+1	+1	+1	+1	+1	-
1111	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	-

表 21-9　CCRVn 与 AR 值在抖动模式下的所有变化(边沿对齐)

抖动模式同样适用在中心对齐模式时，**AD16C6T_CCVALn** 与 **AD16C6T_AR** 寄存器会在连续的 8 个 PWM 周期内变化，如下表:

LSB	PWM 周期															
	1		2		3		4		5		6		7		8	
	递增	递减	递增	递减	递增	递减	递增	递减	递增	递减	递增	递减	递增	递减	递增	递减
0000	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0001	+1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0010	+1	-	-	-	-	-	-	-	+1	-	-	-	-	-	-	-
0011	+1	-	-	-	+1	-	-	-	+1	-	-	-	-	-	-	-
0100	+1	-	-	-	+1	-	-	-	+1	-	-	-	+1	-	-	-
0101	+1	-	+1	-	+1	-	-	-	+1	-	-	-	+1	-	-	-
0110	+1	-	+1	-	+1	-	-	-	+1	-	+1	-	+1	-	-	-
0111	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	-	-
1000	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-
1001	+1	+1	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-
1010	+1	+1	+1	-	+1	-	+1	-	+1	+1	+1	-	+1	-	+1	-
1011	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-	+1	-	+1	-
1100	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-
1101	+1	+1	+1	+1	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-
1110	+1	+1	+1	+1	+1	+1	+1	-	+1	+1	+1	+1	+1	+1	+1	-
1111	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	-

表 21-10　CCRVn 与 AR 值在抖动模式下的所有变化(中心对齐)

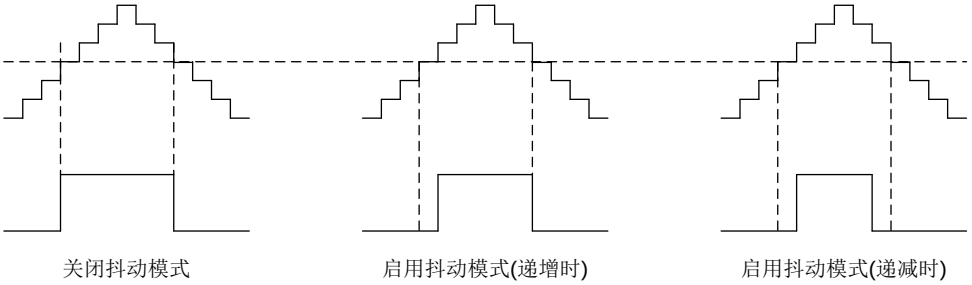


图 21-24　抖动模式下 PWM 输出影响(中心对齐)

20. 5. 10 输出比较模式

这个功能被用来控制一个输出波形或是指示一段时间是否已经过去。

通道 1 到通道 4 可输出到 IO 引脚，而通道 5 和 6 只在 MCU 内部使用(例如定时器互联或 ADC 触发)。

当捕获或比较寄存器和 **AD16C6T_COUNT** 寄存器数值匹配时:

- **AD16C6T_CHMRn** 寄存器的 CHnMOD 位选择输出模式，而输出极性由 **AD16C6T_CCEP** 寄存器的 CCnPOL 位控制:
  - ◇ 设置 CHnMOD 位为 00000b:当计数器匹配比较器时输出保持其电平。
  - ◇ 设置 CHnMOD 位为 00001b:当计数器匹配比较器时输出有效电平(假设 CCnPOL=0，有效电平为高电平)。
  - ◇ 设置 CHnMOD 位为 00010b:当计数器匹配比较器时输出无效电平(假设 CCnPOL=0，无效电平为低电平)。
  - ◇ 设置 CHnMOD 位为 00011b:当计数器匹配比较器时翻转电平。
- 设置中断状态寄存器的标志位为 1(**AD16C6T_RIF** 寄存器的 CHn 位)。
- 若设置相应的中断开启位为 1(**AD16C6T_IER** 寄存器的 CHn 位)，则产生中断。
- 若设置相应的开启位为 1(**AD16C6T_DMAEN** 寄存器的 CHn 位，**AD16C6T_CON2** 寄存器的 CCDMASEL 位用于 DMA 请求的选择)，则发送 DMA 请求。

设置 **AD16C6T_CHMRn** 寄存器的 CHnPEN 位数值可决定 **AD16C6T_CCVALn** 寄存器是否有缓冲功能。

在输出比较模式中，更新事件 UPD 对 CHn 的输出没有影响。输出比较模式同样可以用来输出单个脉冲(单脉冲模式)。

输出比较的设置过程:

1. 选定计数器时钟(内部、外部、预分频)。
2. 设置 **AD16C6T_AR** 与 **AD16C6T_CCVALn** 寄存器并写入所需数据。
3. 若需要产生中断请求，设置 **AD16C6T_IER** 寄存器的 CHn 位为 1。
4. 选择输出模式，例如:
   - 设置 CHnMOD 位为 00011b，当 CNTV 与 CCRVALn 匹配时，CHn 输出翻转。
   - 设置 CHnPEN 位为 0，关闭预装载功能。

- 设置 CCnPOL 位为 0，选择有效电平为高电平。
- 设置 CCnEN 位为 1，开启输出。

5. 设置 **AD16C6T_CON1** 寄存器的 CNTEN 位为 1，开启计数器。

假设预装载寄存器开启(CHnPEN 位为 1)，设置 **AD16C6T_CCVALn** 寄存器数值在下次更新事件发生时更新至缓冲寄存器。预装载寄存器未开启(CHnPEN 位为 0)，通过设置 **AD16C6T_CCVALn** 寄存器数值可随时更新控制输出波形。

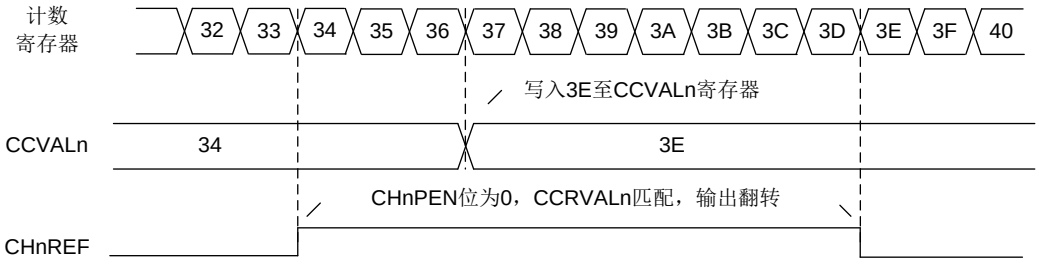


图 21-25　输出比较模式，触发 CHn

20. 5. 11 强制输出模式

在输出模式(**AD16C6T_CHMRn** 寄存器的 CCnSSEL 位为 00b)下，通过软件可以将输出比较信号强制设置为高电平或低电平，输出信号并不会参考 **AD16C6T_CCVALn** 寄存器和 **AD16C6T_COUNT** 寄存器之间的比较结果。

设置 **AD16C6T_CHMRn** 寄存器的 CHnMOD 位为 00101b，输出比较参考信号(CHnREF)为强制高电平，输出比较信号(CHn/CHnN)强制为有效电平(极性由 **AD16C6T_CCEP** 寄存器对应的 CCnPOL 位或 CCnNPOL 位决定)。反之，若设置 CHnMOD 位为 00100b 则强制设置低电平。

例如:设置 CCnPOL 位为 0(CHn 高电平有效)，则 CHn 被强制为高电平。

在此模式下，**AD16C6T_CCVALn** 寄存器和 **AD16C6T_COUNT** 寄存器之间的比较仍然进行，仍可设置相应的标志位为 1，并发送相应的中断以及 DMA 请求。

20. 5. 12 非对称PWM模式

在非对称 PWM 模式下，产生两个可编程相位移的中心对齐 PWM 信号。配置 **AD16C6T_AR** 寄存器数值决定 PWM 频率，占空比以及相位移则由成对的 **AD16C6T_CCVALn** 寄存器配置。这对寄存器分别控制递增计时和递减计时时产生的 PWM 波型，并在每半个 PWM 周期调整一次 PWM 输出。

- CH1REFC(或 CH2REFC)由配置 **AD16C6T_CCVAL1** 和 **AD16C6T_CCVAL2** 控制。
- CH3REFC(或 CH4REFC)由配置 **AD16C6T_CCVAL3** 和 **AD16C6T_CCVAL4** 控制。

2 个通道可以独立选择非对称 PWM 模式，每对 CCVALn 决定一个 CHn 输出。

设置 **AD16C6T_CHMRn** 寄存器的 CHnMOD 位写入 01110b 为非对称 PWM 模式 1，写入 01111b 为非对称 PWM 模式 2。

注:因为兼容性的关系，CHnMOD[4:0]位分为两个部分，最高两位与其他位并不连续。

在非对称 PWM 模式下，通道 1 的 CH1REFC 信号在递增计数时会使用 CCRV1 输出比较，而在递减计数时会使用 CCRV2。虽然这种模式需要成对使用 CCVALn，但相对应的通道 2 可以选择正常输出 CH2REF，或者也可以输出非对称的 CH2REFC 信号，这取决于 **AD16C6T_CHMR1** 寄存器中的 CH2MOD 设置。

下图为使用非对称 PWM 模式 2，产生两个 50%占空比、90 度相位移的 PWM 信号波型 ：

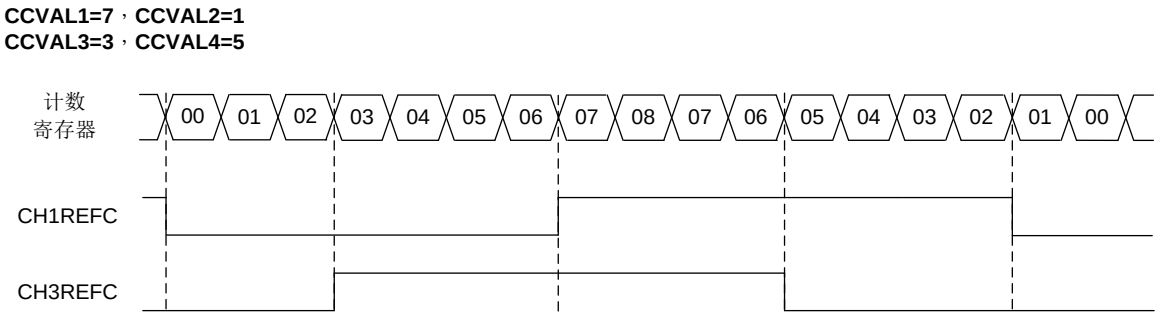


图 21-26　非对称 PWM 模式输出波型

20. 5. 13 組合PWM模式

在组合模式中，产生两个可编程延迟的边沿或中心对齐 PWM 波型。配置 **AD16C6T_AR** 寄存器数值决定 PWM 频率，PWM 占空比和延迟由两个 **AD16C6T_CCVALn** 寄存器配置。产生的 PWM 波型由两个参考的 PWM 波型做"OR"运算或"AND"运算所组成。

- CH1REFC(或 CH2REFC)由配置 **AD16C6T_CCVAL1** 和 **AD16C6T_CCVAL2** 控制。
- CH3REFC(或 CH4REFC)由配置 **AD16C6T_CCVAL3** 和 **AD16C6T_CCVAL4** 控制。

2 个通道可以独立选择非对称 PWM 模式，每对 CCVALn 决定一个 CHn 输出。

设置 **AD16C6T_CHMRn** 寄存器的 CHnMOD 位写入 01100b 为组合 PWM 模式 1，写入 01101b 为组合 PWM 模式 2。

当通道设置为组合 PWM 模式时，2 个通道必须设置成不同的 PWM 模式，例如通道 1 设置组合 PWM 模式 1，通道 2 必须设置组合 PWM 模式 2。

注:因为兼容性的关系，CHnMOD[4:0]位分为两个部分，最高两位与其他位并不连续。

下图为组合 PWM 模式中，可以产生的通道波型，通过以下配置:

- 通道 1 配置组合 PWM 模式 2。
- 通道 2 配置 PWM 模式 1。
- 通道 3 配置组合 PWM 模式 1。
- 通道 4 配置 PWM 模式 2。

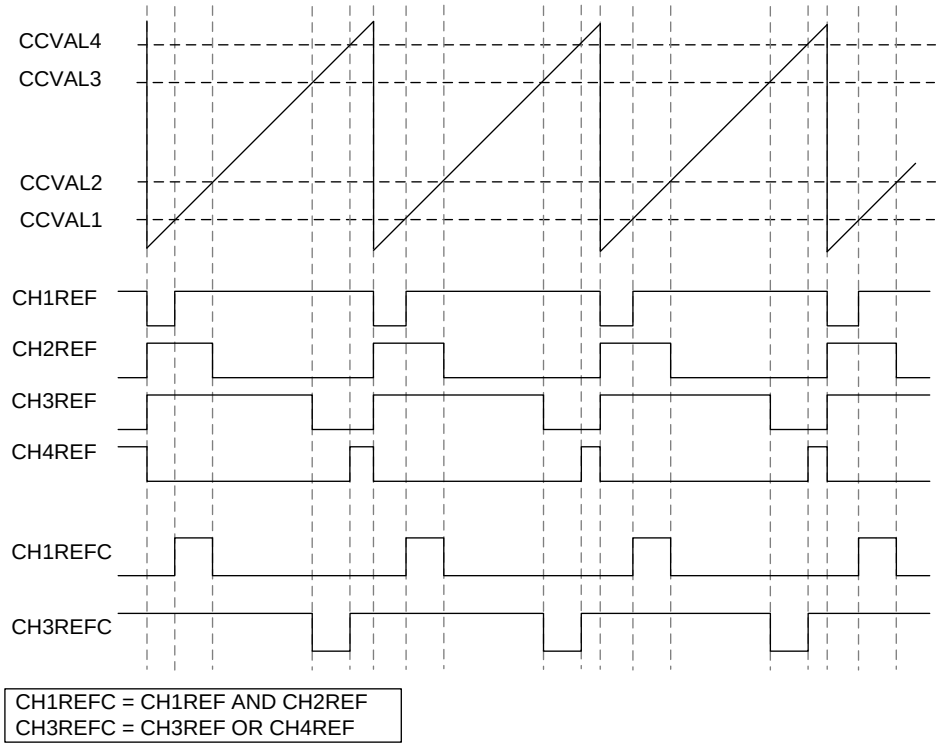


图 21-27　组合 PWM 模式输出波型

20. 5. 14 組合三相PWM模式

在组合三相 PWM 模式中，其通道 1 到通道 3 产生的 PWM 信号，可与通道 5 产生的 PWM 信号做"AND"运算所组成。

通道 5 产生的 CH5REF 信号用于定义组合信号。设置 **AD16C6T_CCVAL5** 的 GC5Cn 位选择 CH5REF 与通道 1 到通道 3 的 PWM 波型做组合"AND"运算。

- 设置 GC5C1，则 CH1REFC 由配置 **AD16C6T_CCVAL1** 和 **AD16C6T_CCVAL5** 控制。
- 设置 GC5C2，则 CH2REFC 由配置 **AD16C6T_CCVAL2** 和 **AD16C6T_CCVAL5** 控制。
- 设置 GC5C3，则 CH3REFC 由配置 **AD16C6T_CCVAL3** 和 **AD16C6T_CCVAL5** 控制。

通道 1 到通道 3 可以独立选择组合三相 PWM 模式，配置 GC5Cn 位选择。

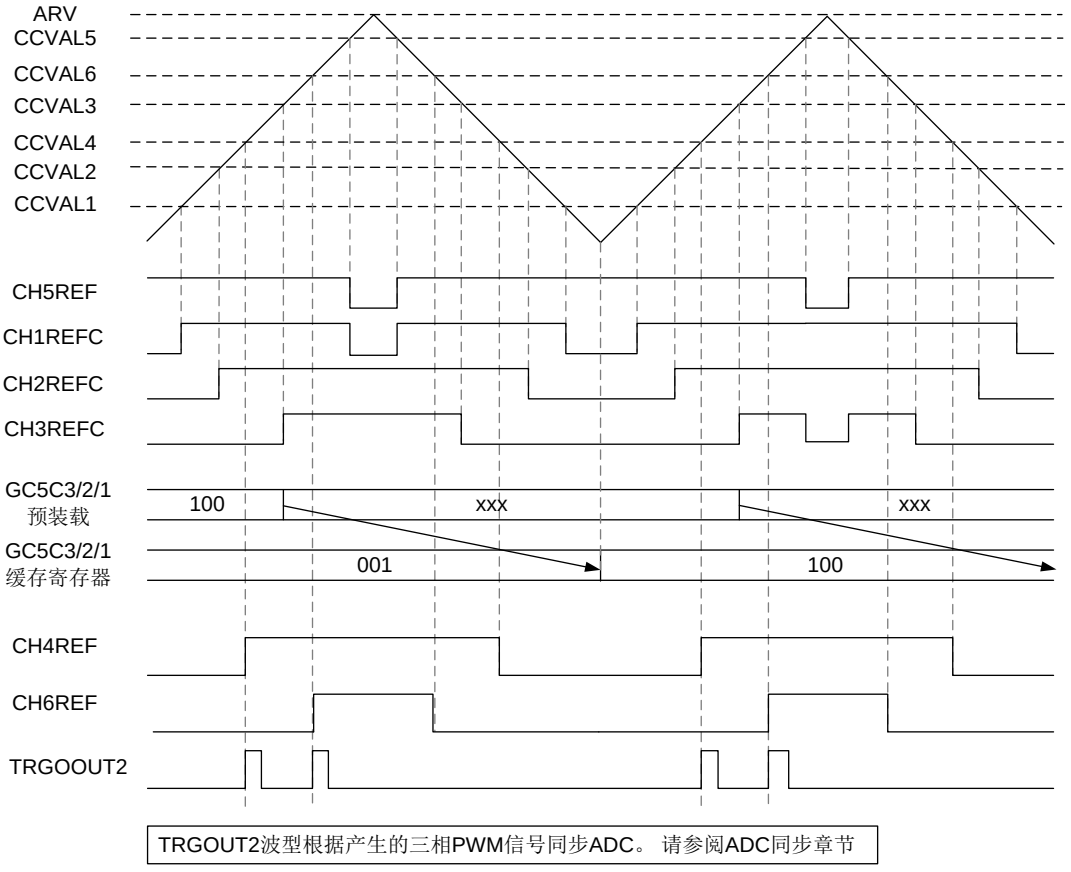


图 21-28　组合三相 PWM 模式输出波型

20. 5. 15 组合三相非对称PWM模式

在组合三相非对称 PWM 模式中，其通道 1 到通道 3 产生的 PWM 信号，可以独立选择与通道 5 组合三相非对称 PWM 模式。通过设定 **AD16C6T_CHMR1** 与 **AD16C6T_CHMR2** 寄存器的 CHnMOD 位为 10000b 与 10001b，分别产生 PWN 模式 1 与 PWM 模式 2 波型。

- CH1REFC 由配置 **AD16C6T_CCVAL1** 和 **AD16C6T_CCVAL5** 控制。
- CH2REFC 由配置 **AD16C6T_CCVAL2** 和 **AD16C6T_CCVAL5** 控制。
- CH3REFC 由配置 **AD16C6T_CCVAL3** 和 **AD16C6T_CCVAL5** 控制。

下图为组合三相非对称 PWM 模式中，可以产生的通道波型:

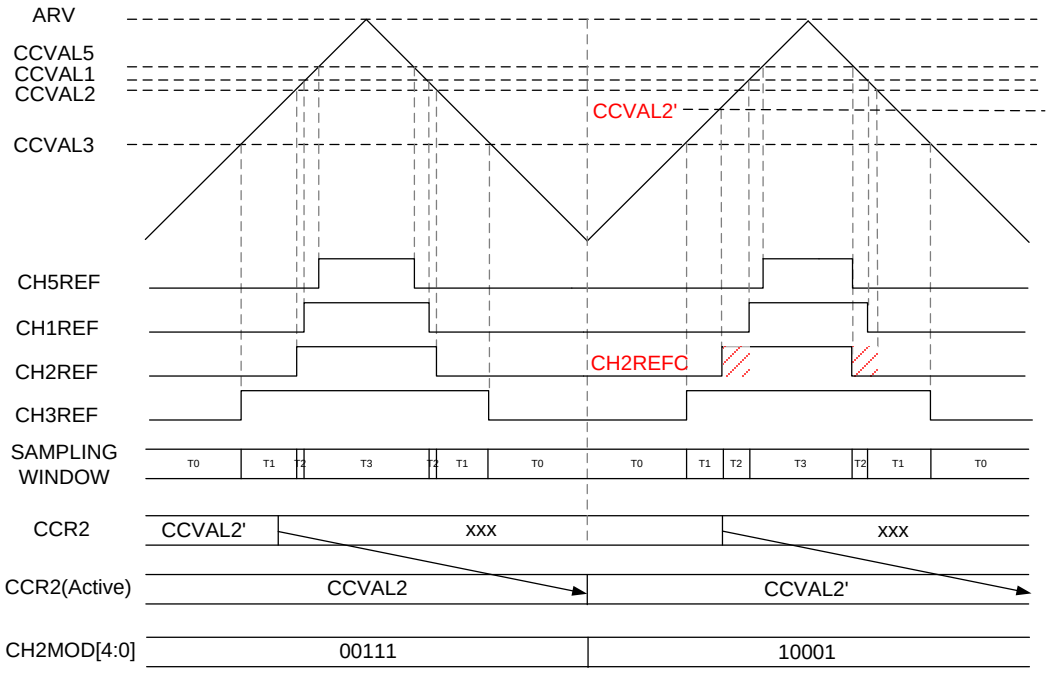


图 21-29　组合三相非对称 PWM 模式输出波型

在应用上，当两个通道比较值相近时，留给 ADC 采样的 Sampling window(T2)不足时，可以配置通道为非对称 PWM 模式(以通道 2 为例)，让 ADC 有足够采样时间。

ADC 触发信号源可以参考 ADC 触发生成章节。

20. 5. 16 外部事件清除比较输出

设置相对应的 **AD16C6T_CHMRn** 寄存器的 CHnOCLREN 位为 1，在选定的 OCLR_INT 输入端为高准位时，可将相对应的输出信号暂时清除为 0，直到下一次更新事件(UPD)产生。该功能只能在输出比较模式和 PWM 模式下使用，在强制模式下不起作用。

如下图所示，OCLR_INT 的输入来源根据 **AD16C6T_SMCON** 寄存器的 OCLRS 位设定，可以是经过数字滤波及极性选择后的 ETP 信号，也可以是 **AD16C6T_AFR2** 寄存器的 OCLRSEL 位所指定的 OCLR 输入源。

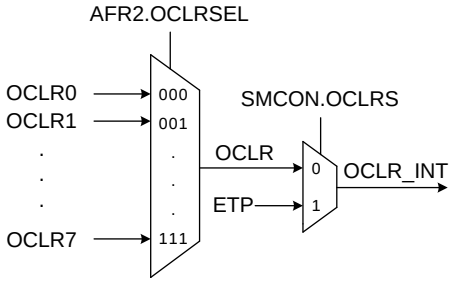


图 21-30　清除比较输出源 OCLR_INT

选择来源为 ETP 时，相关配置如下:

1. 设置 **AD16C6T_SMCON** 寄存器中的 OCLRS 位为 1，选择输出通道清除来源为 ETP。
2. 设置 **AD16C6T_SMCON** 寄存器中的 ETPRES 位为 00b，关闭外部触发预分频器。
3. 设置 **AD16C6T_SMCON** 寄存器的 ECM2EN 位为 0，关闭外部时钟源 2。
4. 外部触发极性(ETPOL)和外部触发滤波器(ETFLT)可根据用户需要设置。

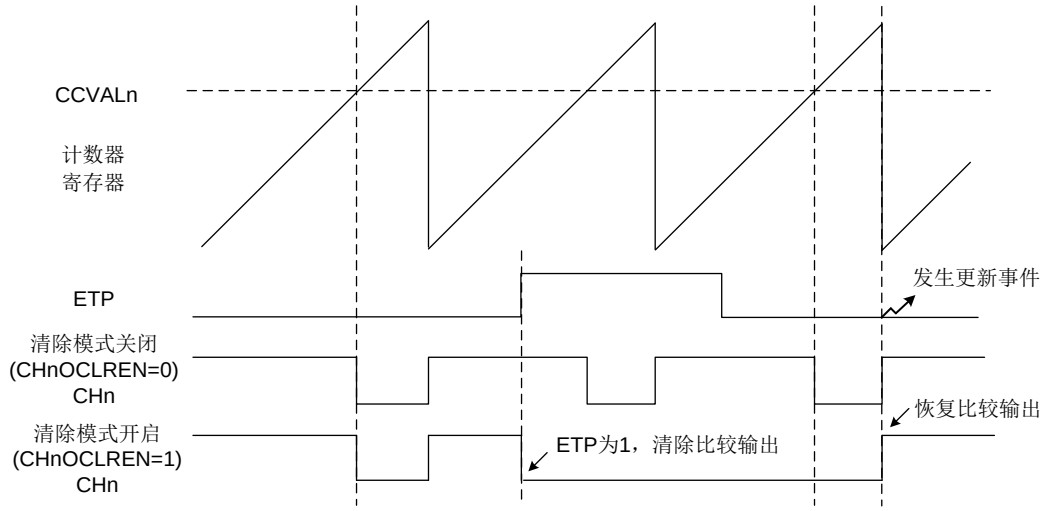


图 21-31　清除比较输出 CHn

20. 5. 17 单脉冲模式

单脉冲模式(SPMEN)是一个特殊模式。在此模式下，计数器可以通过外部触发下启动，并可以产生一个脉宽可编成的波形。

通过从模式控制器开启计数器。在输出比较模式或 PWM 模式下生成波形。设置 **AD16C6T_CON1** 寄存器的 SPMEN 位为 1 选择单脉冲模式，在下一次发生更新事件后，计数器将自动停止计数。

只有当 **AD16C6T_CCVALn** 寄存器值和初始 **AD16C6T_COUNT** 寄存器值不同时，才能正确的产生一个脉冲。计数器开始计数前(定时器等待触发)，必须如下设置:

- 递增计数:CNTV < CCVALn ≤ AR(注意:0 < CCVALn)
- 递减计数:CNTV > CCVALn

举例来说，假设今天要达到以下效果:

当检测到 I2 输入引脚出现上升沿后，定时器自动开启计数，并经过 T 延迟的时间后，在 CH1 上产生一个长度为 T 脉冲的正脉冲。

就可以设定成单脉冲模式。

选择 I2 作为触发源，相关配置如下:

1. 设置 **AD16C6T_CHMR1** 寄存器中的 CC2SSEL 位为 01b，选择通道输入源为 I2。
2. 设置 **AD16C6T_CCEP** 寄存器中的 CC2POL、CC2NPOL 位为 0，选择 I2 上升沿有效。
3. 设置 **AD16C6T_SMCON** 寄存器的 TSSEL 位为 00110b，选择触发来源(TRGI)为 I2 滤波后信号(I2FP)。
4. 设置 **AD16C6T_SMCON** 寄存器的 SMODS 位为 0110b，选择触发模式，在检测到 TRGI 上升沿时自动开启计数器。

脉冲波型由 **AD16C6T_CCVAL1** 寄存器决定，其中:

- $T_{延迟}$ = **AD16C6T_CCVAL1** 寄存器的值。
- $T_{脉冲}$ = **AD16C6T_AR** 与 **AD16C6T_CCVAL1** 之间的差决定。

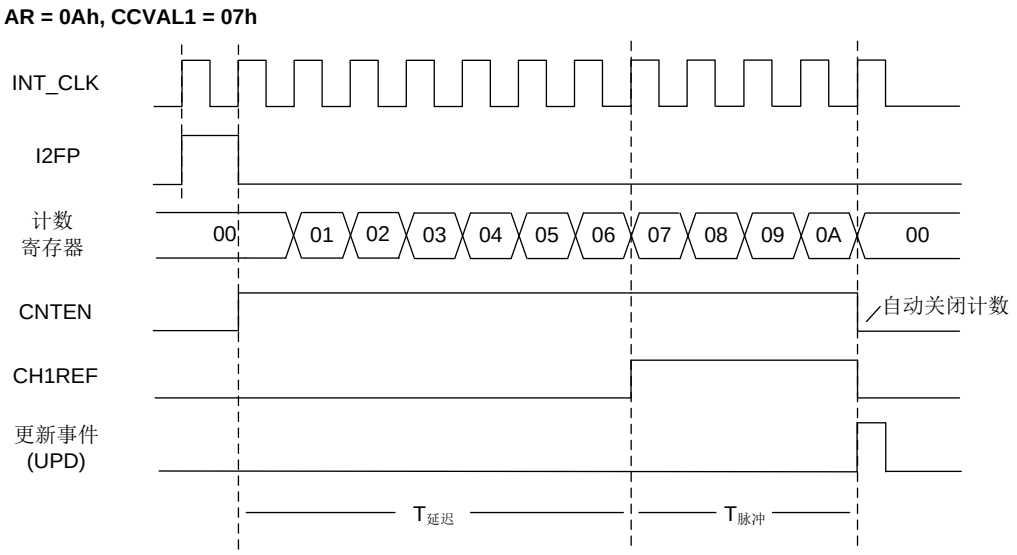


图 21-32　通道 1 触发模式下，基于 PWM 模式 2 单脉冲输出波形

CHn 快速开启模式

在单脉冲模式下，In 输入的有效边沿会自动开启计数器(硬件设置 CNTEN 位为 1)，并在 **AD16C6T_COUNT** 与 **AD16C6T_CCVALn** 比较后输出信号。然而在这个过程中需要数个时钟周期，将会延长 In 输入边沿与输出信号之间的延迟。

如果要使用最小延迟输出信号，可以设置 **AD16C6T_CHMR1** 寄存器的 CHnFEN 位为 1 开启快速开启模式。当侦测到 In 输入的有效边沿时，不再考虑比较值，强制让输出信号等效于匹配成功后的电平。此配置只在 PWM 模式 1 或 PWM 模式 2 时才能使用。

20. 5. 18 多次触发单脉冲模式

多次触发单脉冲模式下，当在一个计时周期中发生多次触发事件可以延长脉冲长度，与发单脉冲模式差别如下:

- 发生触发时，立即产生脉冲(无延迟)。
- 在前次触发脉冲结束前，又发生新的触发，则会延长脉冲长度。

以通道 1 多次触发单脉冲模式为例，相关配置如下:

1. 设置 **AD16C6T_SMCON** 寄存器中的 SMODS 位为 1000b，选择组合复位+触发模式。
2. 设置 **AD16C6T_CHMR1** 寄存器中的 CH1MOD 位为 01000b 或 01001b，选择多次触发单脉冲模式 1 或模式 2。

定时器配置递增计数模式时，对应的 CCVALn 必须配置 0，由 ARV 配置脉冲长度。

定时器配置递减计数模式时，对应的 CCVALn 必须大于或等于 ARV 数值。

注: 这个模式不能和中心对齐的 PWM 模式一起使用，CMSEL 位需设置 0。

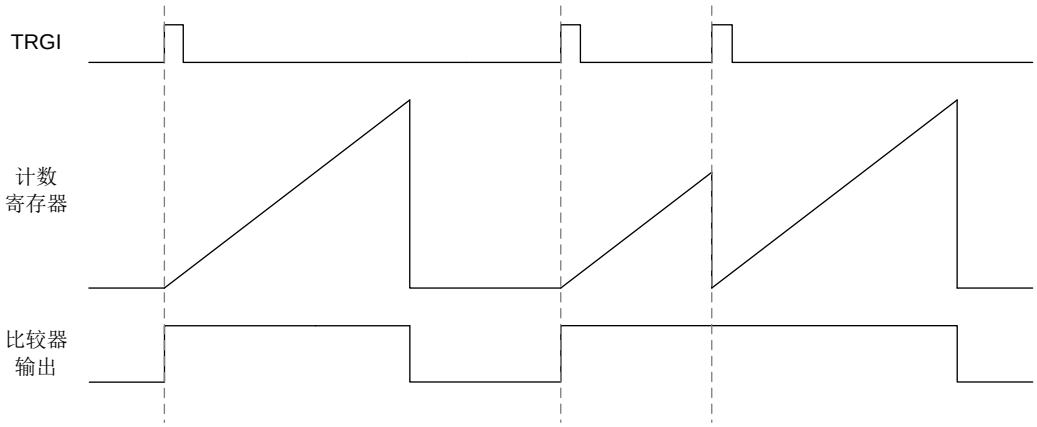


图 21-33　多次触发单脉冲模式

20. 5. 19 比较脉冲模式

此模式可以在比较器发生比较匹配时，产生一个脉冲宽度可编程的输出信号。只适用在通道 3 和通道 4，结构如下图:

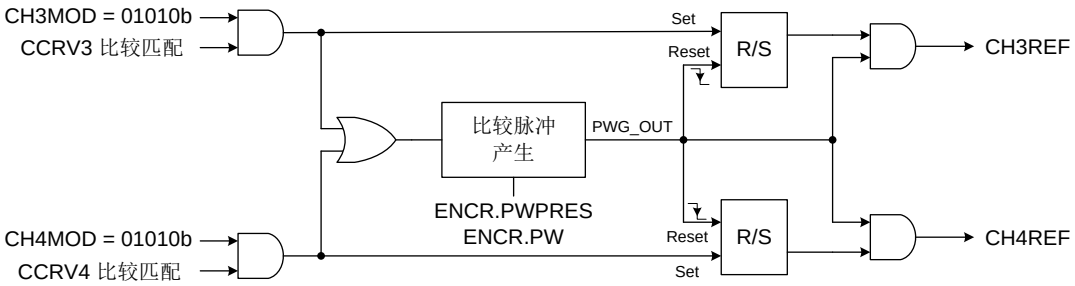


图 21-34　比较脉冲模式结构图

设置 **AD16C6T_CHMR2** 寄存器的 CH3MOD、CH4MOD 位为 01010b 可选择此模式。设置 **AD16C6T_ENCR** 寄存器的 PWPRES 和 PW 位，可编程脉冲宽度，公式如下:

- $t_{PWG} = (2^{(PWPRES[2:0])}) \times t_{INT\_CLK}$
- $t_{PW} = PW[7:0] \times t_{PWG}$

下图显示如何在边沿对齐模式下生成比较脉冲波型:

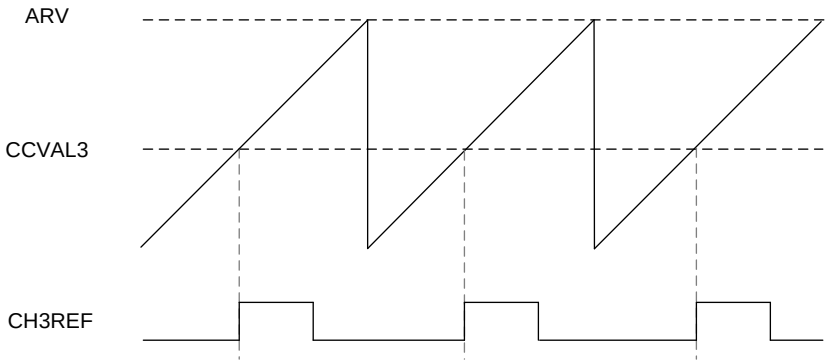


图 21-35　边沿对齐模式下的比较脉冲波型

比较脉冲是可以被重新触发的，一旦前次脉冲尚未结束前又发生新的触发，此时前次脉冲宽度持续时间将会被延长。

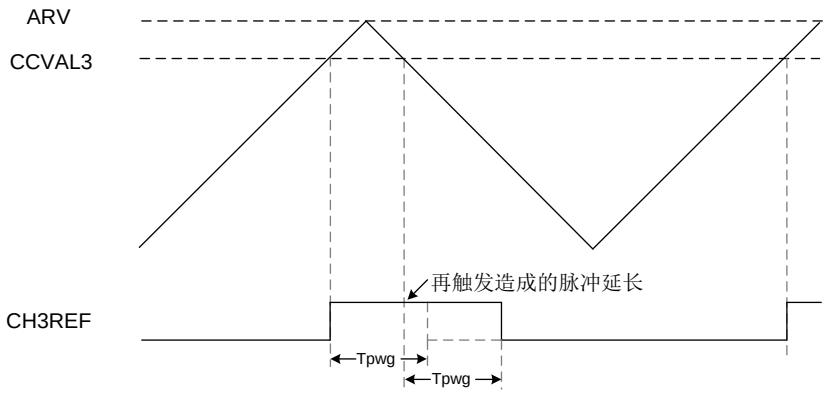


图 21-36　再触发延长脉冲宽度的比较脉冲波型

如果同时启用两个通道，其中一个通道上的比较脉冲尚未结束，但另一个通道又产生脉冲，则原通道的脉冲宽度持续时间将会被延长。

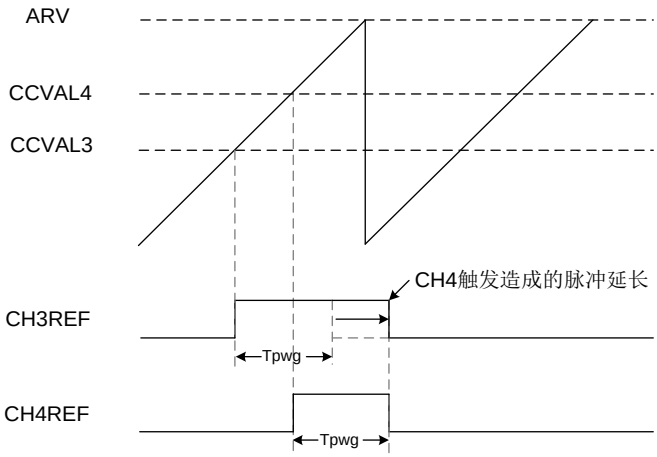


图 21-37　双通道延长脉冲宽度的比较脉冲波型

20. 5. 20 互补输出与死区时间

在控制输出时，需要根据连接到输出的装置特性（如电平转换器的固有延迟时间，功率开关的延迟时间等），延迟两个互补输出开关的时间，这个瞬时的延迟即为死区时间。

每个输出的极性（主输出 CHn 或互补输出 CHnN）可以独立选择，通过在 **AD16C6T_CCEP** 寄存器中写入 CCxP 和 CCxNP 位来实现。

互补信号 CHn 和 CHnN 是由多个控制位的组合所控制的，分别是 **AD16C6T_CCEP** 寄存器的 CCnEN 和 CCnNEN 位，**AD16C6T_BDCFG** 和 **AD16C6T_CON2** 寄存器的 GOEN、OISSn、OISSnN、OFFSSI 及 OFFSSR 位(可参考输出引脚控制章节)。

值得注意的是，当切换为空闲状态(GOEN 变为 0)后，死区时间依然有效。

死区时间长度计算公式可参考 **AD16C6T_BDCFG** 寄存器的 DT 位，用来控制所有通道的死区时间长短。输出波形参考 CHnREF 信号，产生 CHn 和 CHnN 两路输出。当 CHn 和 CHnN 皆为高电平有效时，两路输出行为如下:

- CHn 的输出信号与参考信号(CHnREF)一致。相较于参考信号的上升沿，CHn 上升沿输出会有延迟。
- CHnN 的输出信号与参考信号相反。相较于参考信号的下降沿，CHnN 上升沿输出会有延迟。

设置 **AD16C6T_DCFG2** 寄存器的 DTAEN 位为 1 时，开启非对称死区延迟功能:

- 相较于参考信号的上升沿，CHn 上升沿输出延迟为 **AD16C6T_BDCFG** 寄存器的 DT 位设定。
- 相较于参考信号的下降沿，CHnN 上升沿输出延迟为 **AD16C6T_DCFG2** 寄存器的 DT2 位设定。

设置 **AD16C6T_DCFG2** 寄存器的 DTPEN 位为 1 时，死区延迟(DT、DT2)设定具有缓冲功能，只有在更新事件发生时，才会将预装载的值写入到缓冲寄存器中。

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 位为 0，CCnNP 位为 0，GOEN 位为 1，CCnEN 位为 1，和 CCnNEN 位为 1)。

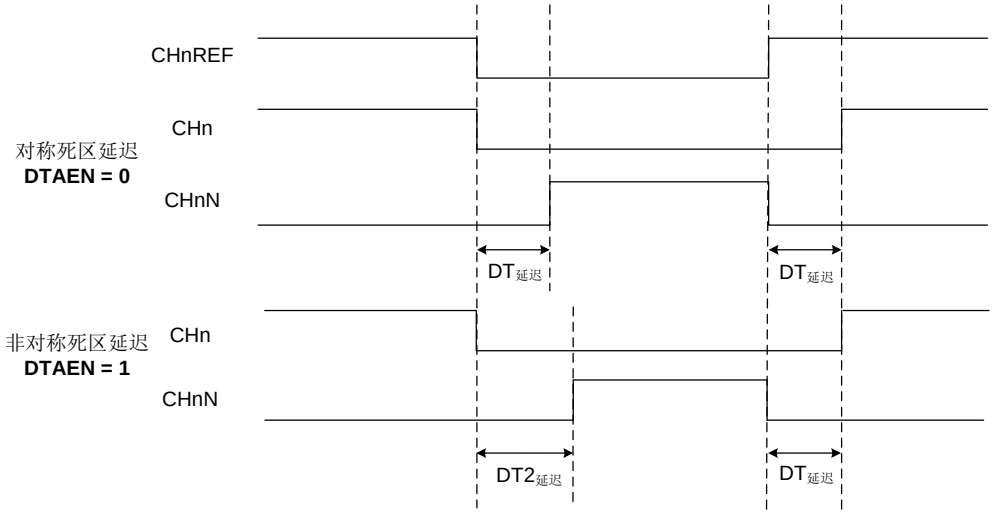


图 21-38　互补输出含死区时间插入

任何情况下，CHn 与 CHnN 的输出信号都不能同时为有效电平。

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。

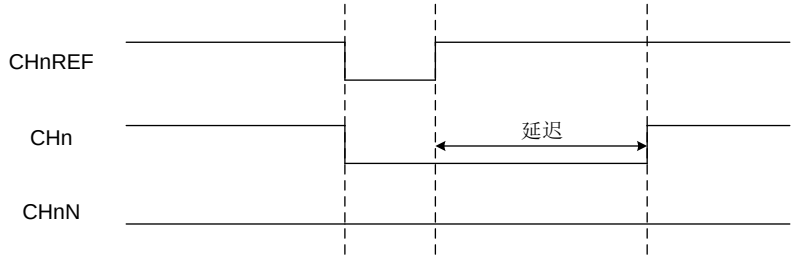


图 21-39　死区延迟时间大于 CHnN 脉冲

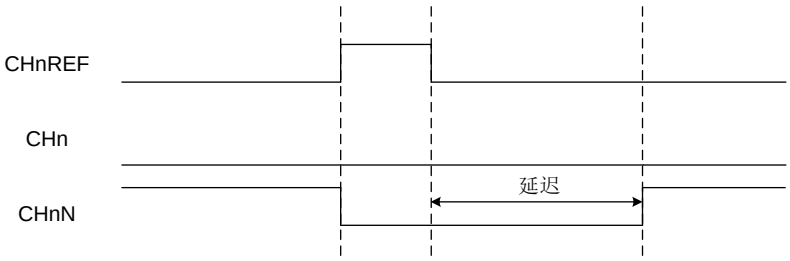


图 21-40　死区延迟时间大于 CHn 脉冲

20. 5. 21 刹车功能

刹车功能是为了保护受到 PWM 信号驱动的电路。该功能包含两个刹车输入通道，用来检测故障事件。当发生刹车事件时，硬件会自动关闭 PWM 输出，并将其强制设置为预先配置好的安全准位。此外，刹车事件也可以选择 MCU 内部事件当作刹车输入源。

刹车功能拥有两个通道。刹车输入通道 1 可以检测系统级的故障(如时钟失效、Hard Fault、低电压检测等)和应用故障(来自输入引脚和内置比较器)，并在死区延迟后强制将输出设置为预配置的电平(有效或无效)。刹车输入通道 2 仅包括应用故障，可以强制将输出设置为无效状态。

刹车期间的输出开启信号与输出电平由以下几个控制位进行决定:

- **AD16C6T_BDCFG** 寄存器的 GOEN 位，通过软件开启或关闭输出，并在刹车输入通道 1 或 2 检测到刹车事件时清 0。
- **AD16C6T_BDCFG** 寄存器的 OFFSSI 位,用来定义此时输出是由定时器控制成无效状态，或由 GPIO 控制(通常处于 High-Z 高阻态)。
- **AD16C6T_CON2** 寄存器的 OISSn 和 OISSnN 位，用来定义在输出空闲状态时的输出状态(有效或无效)。

注:OISSn 和 OISSnN 位无法配置成同时输出有效电平。

详细信息可参考输出引脚控制章节。

系统复位后，刹车电路被禁止且 GOEN 位为 0。可以通过在 **AD16C6T_BDCFG** 寄存器中设置 BRKEN 和 BRK2EN 位来启用刹车功能。刹车输入极性可以通过在同一寄存器中配置 BRKP 和 BRK2P 位来选择。

刹车(BRK)信号来源为:

- BKIN 输入引脚
- BKIN_EXT 输入引脚
- 内部来源
  - ◇ 时钟停振事件由时钟控制器(RCU)中的时钟安全系统(CSS)产生
  - ◇ 内部 LOCKUP(Hard Fault)输出
  - ◇ PVD 输出
  - ◇ 内部比较器输出

刹车 2(BRK2)信号来源为:

- BKIN2 输入引脚
- BKIN_EXT 输入引脚
- 内部比较器输出

软件可以配过配置 **AD16C6T_SGE** 寄存器的 SGBRK 位与 SGBRK2 位产生刹车事件。

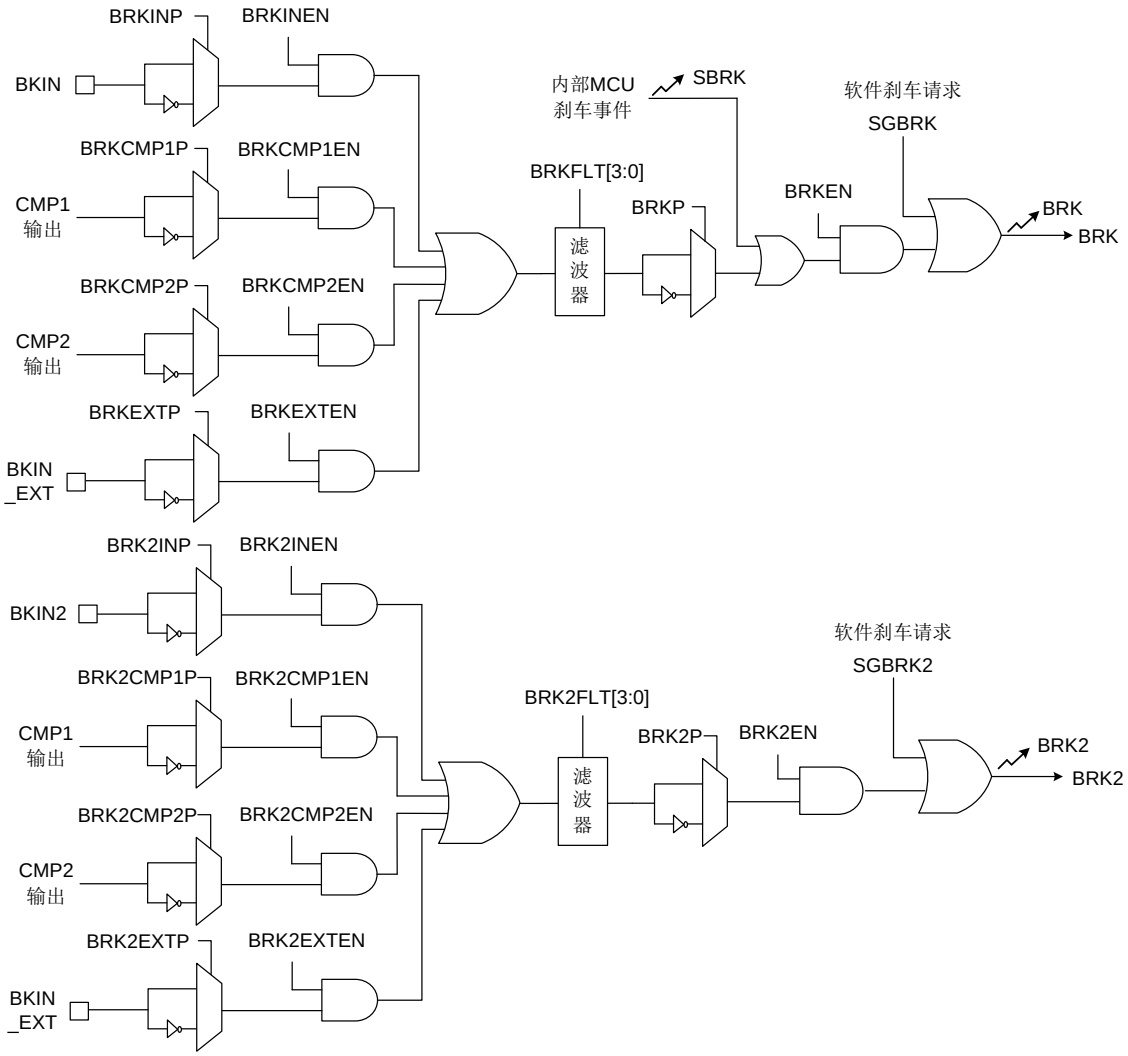


图 21-41　刹车与刹车 2 通道电路

当发生刹车事件，也就是刹车输入检测到有效电平时，会有以下几个情况:

首先，GOEN 位会被清除为 0。如果 OFFSSI 位为 0，输出通道将处于高阻态，此时不受定时器控制。

如果 OFFSSI 位为 1，则输出通道的行为取决于是否为互补输出。对于非互补输出，输出端会根据 **AD16C6T_CON2** 寄存器中的 OISSn/OISSn 位设定，输出预先配置好的空闲电平。

对于互补输出，首先输出端会输出无效状态(取决于极性)。接着在死区时间之后，才会根据 **AD16C6T_CON2** 寄存器中的 OISSn/OISSn 位设定，输出预先配置好的空闲电平。

在刹车事件发生时，还会根据不同的刹车事件，设置 **AD16C6T_RIF** 寄存器中的相应 BRK、BRK2、SBRK 位。如果 **AD16C6T_IER** 寄存器中相应位开启，则会触发中断。

最后，如果 **AD16C6T_BDCFG** 寄存器中的 AOEN 位为 1，则在下一次更新事件(UPD)发生时，硬件会自动将 GOEN 位设为 1。

注:刹车输入为电平有效。因此当刹车输入持续有效电平时，无法将 GOEN 设置为 1(自动或者通过软件)。此外，BRK、BRK2 中断状态也无法被清除。

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，通过 **AD16C6T_BDCFG** 寄存器的 LOCKLVL 位，可从三个保护等级中选择一种保护等级。MCU 复位后，只能对 LOCKLVL 位执行一次写操作。

以下两张图分别说明在互补与非互补输出，响应 BRK 有效刹车事件时的输出电位(OFFSSI=1)。

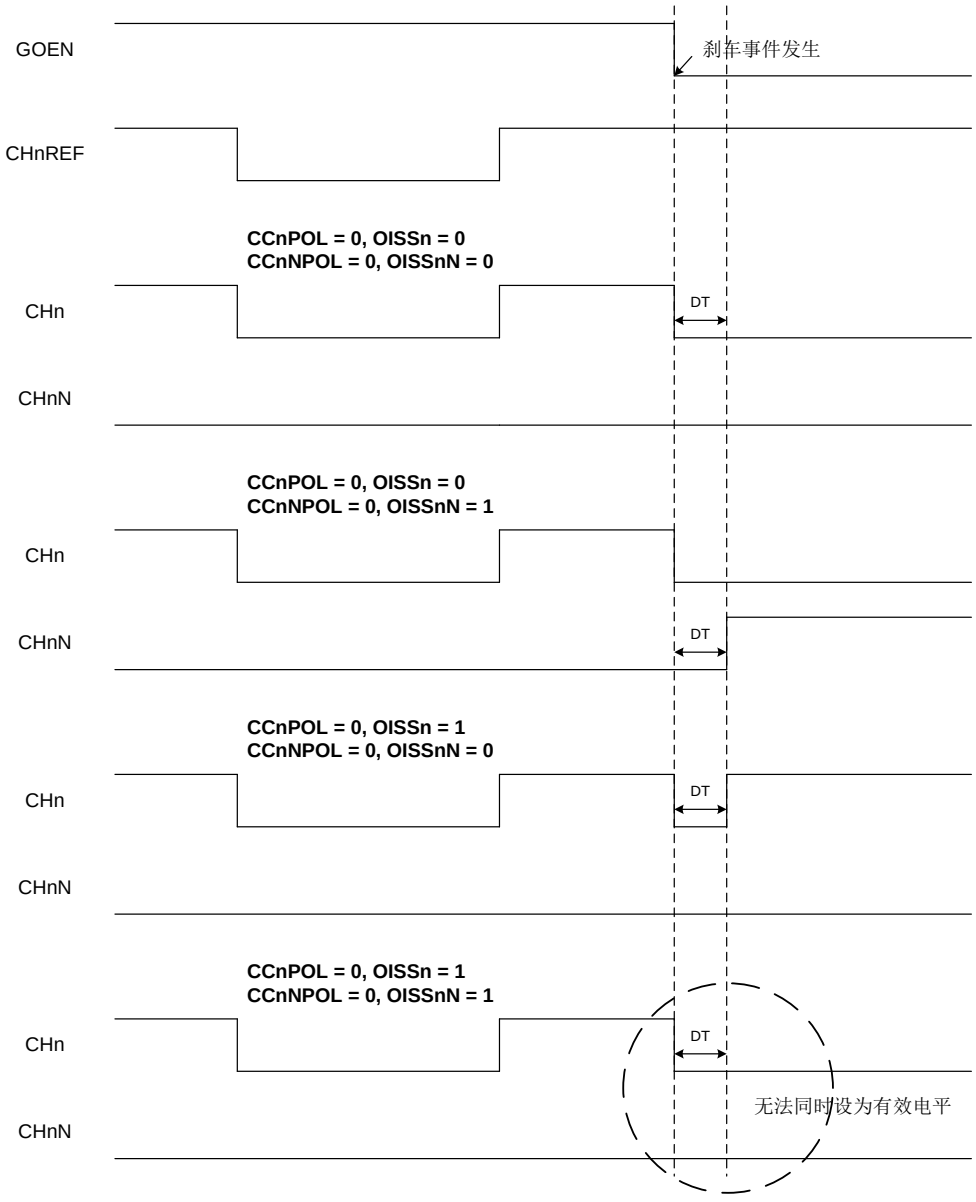


图 21-42　非互补输出下的刹车事件(OFFSSI=1, CCnEN = 1, CCnNEN = 0)

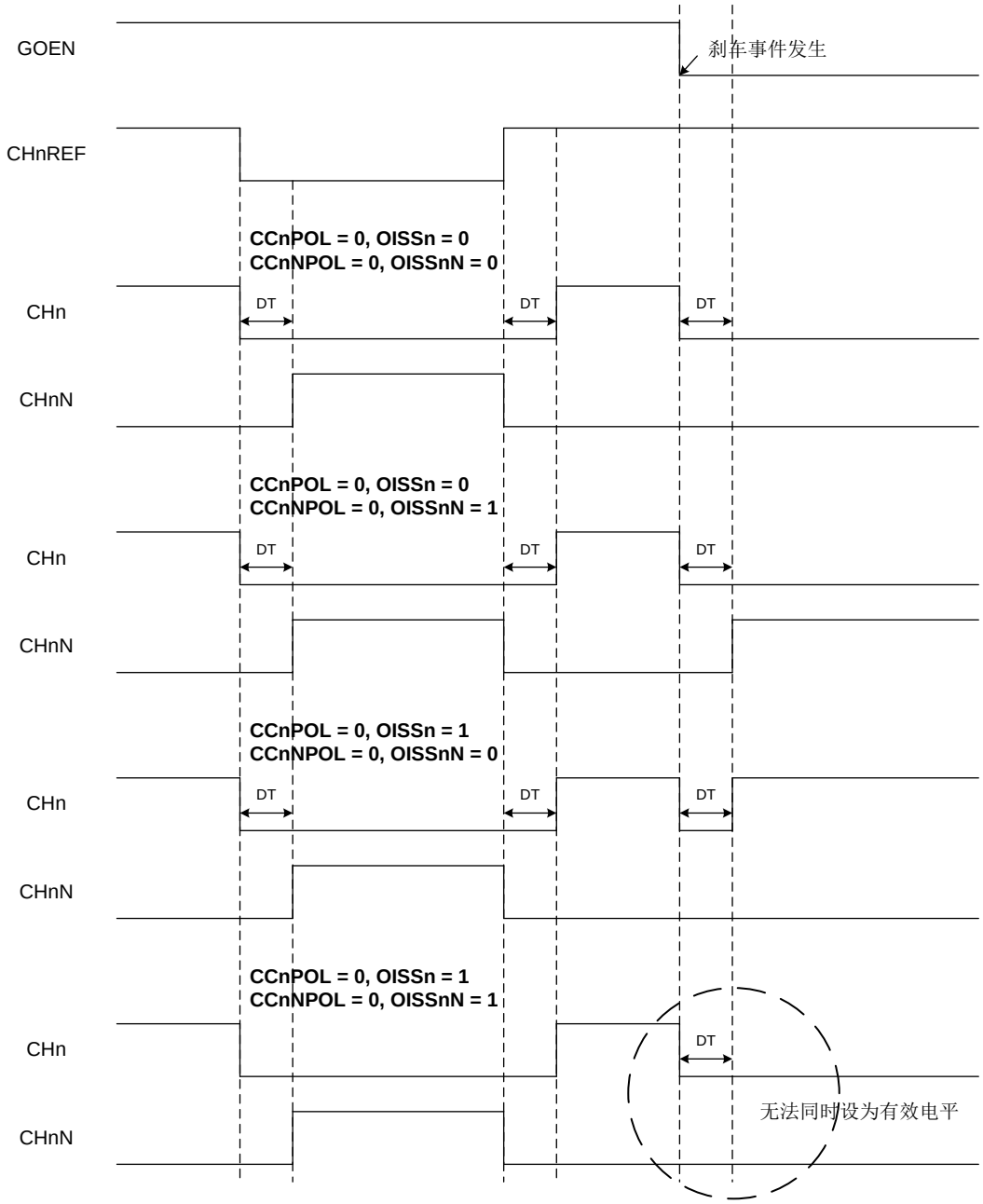


图 21-43　互补输出下的刹车事件(OFFSSI=1, CCnEN = 1, CCnNEN = 1)

有效刹车输入所响应的刹车事件，在 BRK 与 BRK2 有不同的行为:

- BRK 刹车事件可以让引脚输出无效状态，或输出预先配置好的电平。
- BRK2 刹车事件只能让引脚输出无效状态。

此外，BRK 刹车事件的优先权大于 BRK2，如下图:

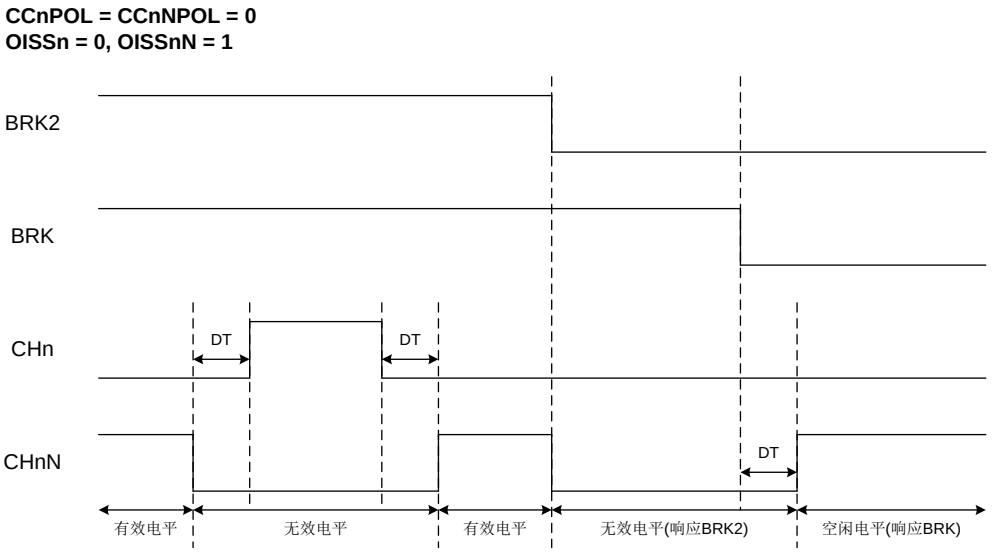


图 21-44　响应 BRK 与 BRK2 刹车事件后的 PWM 输出波型(OFFSSI=1)

注: BRK2 功能必须在 OFFSSR 和 OFFSSI 皆设为 1 时使用。

20. 5. 22 双向刹车输入

定时器支持双向刹车输入引脚(BKIN 和 BKIN2)，如下图:

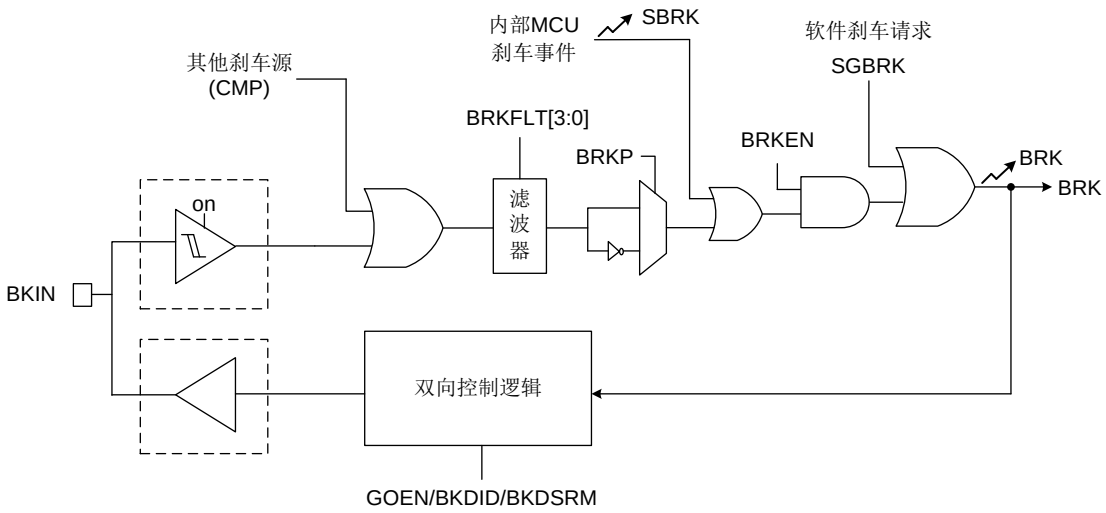


图 21-45　刹车输入通道(以 BKIN 为例)

- 通过刹车输入引脚向外部 MCU 或驱动电路发送故障信号。
- 内部刹车源与外部刹车输入引脚使用"OR"运算，触发刹车事件。

启用双向刹车输入模式需要设置 **AD16C6T_BDCFG** 寄存器的 BKDID 和 BK2BID 位为 1，将刹车输入引脚 BKIN 和 BKIN2 配置为双向模式。
当发生刹车事件时，定时器会将刹车输入引脚(BKIN 和 BKIN2)强制设为低电平输出，用来向外部驱动电路指示发生了故障事件。

因此在使用双向刹车模式时，外部驱动电路的有效刹车信号源必须设计成低电平有效，如下图:

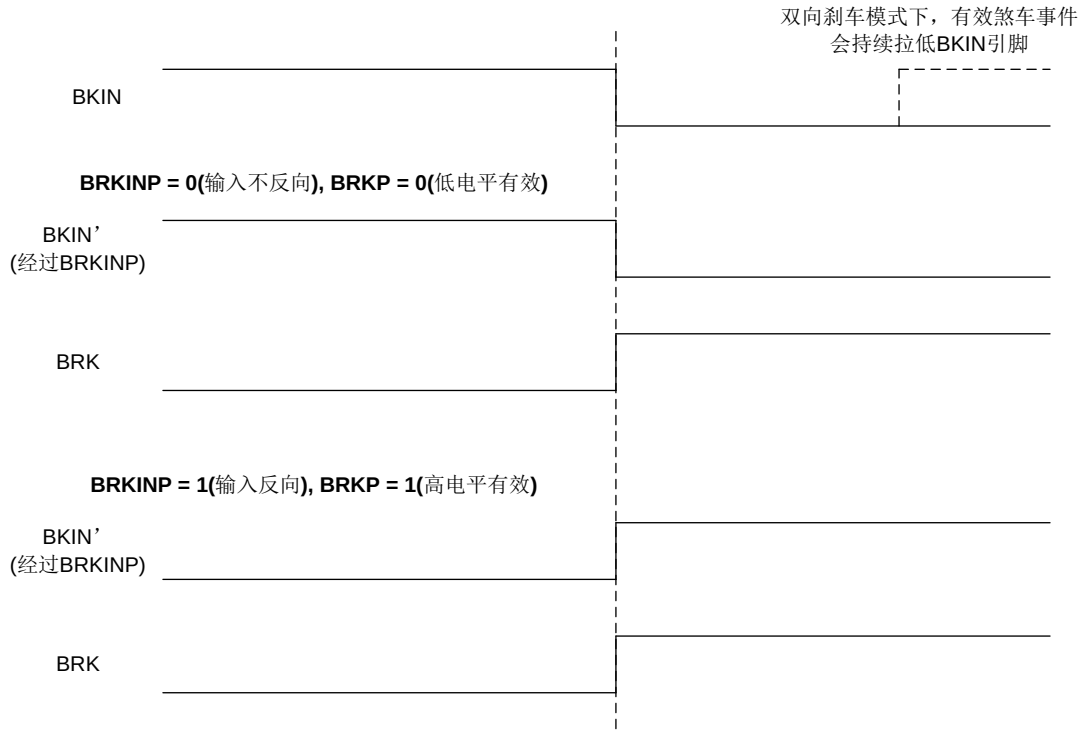


图 21-46　有效双向刹车输入模式(以 BKIN 为例)

注:当准位配置错误时(例如配置成高电平有效)，为了安全目的无法支持双向刹车输入模式。

使用软件刹车事件 SGBRK(SGBRK2)也会将 BKIN(BKIN2)引脚强制设为低电平输出。但若在刹车输入关闭(BRKEN/BRK2EN = 0)的情况下使用软件触发刹车信号，定时器只会生成有效刹车事件，并不会影响 BKIN(BKIN2)引脚状态。

当发生有效刹车事件后(GOEN 清 0)，可通过以下步骤让刹车保护机制重新警备:

1. 设置 **AD16C6T_BDCFG** 寄存器的 BKDSRM/BK2DSRM 位为 1，硬件释放刹车输入引脚(不强制设为低电平输出)。
2. 定时器会持续检测刹车输入引脚电平，直到为无效电平，硬件将自动清除 BKDSRM/BK2DSRM 位为 0。
3. 软件持续检测 BKDSRM/BK2DSRM 位，直到被硬件自动清 0，表示此时刹车保护机制重

新进入警备状态。

4. 清除相关刹车中断事件(若有开启中断)。
5. 软件设置 GOEN 位为 1，重新输出 PWM 信号，若开启 AOEN，定时器在更新事件后自动开启 PWM 输出。

20. 5. 23 输出引脚控制

在输出模式(强制输出、输出比较或 PWM 模式)下，设置参考信号 CHnREF 映像到 CHn 输出或 CHnN 输出的方式，可以通过配置 **AD16C6T_CCEP** 寄存器的 CCnEN、CCnNEN 位来实现。而输出的极性可以由 **AD16C6T_CCEP** 寄存器的 CCnPOL 与 CCnNPOL 位决定。此外，死区延迟可以通过 **AD16C6T_BDCFG** 寄存器的 DT 位和 **AD16C6T_DCFG2** 寄存器的 DT2 位(若开启非对称死区)来设置。

在运行模式下(GOEN=1)，如果关闭输出(CCnEN=0 或 CCnNEN=0)，硬件会直接禁止 CHn 或 CHnN 输出，使引脚为 Hi-Z 高阻态(不由定时器控制)。但是，如果开启了 **AD16C6T_BDCFG** 寄存器的 OFFSSR 位，并且关闭输出的通道的互补通道是开启状态，那么该通道会将输出设为无效电平(由 **AD16C6T_CCEP** 寄存器的 CCnPOL 位或 CCnNPOL 位决定)。

当 GOEN 位因刹车事件或软件写入为 0 时，硬件会直接禁止 CHn 或 CHnN 输出，使引脚为 Hi-Z 高阻态(不由定时器控制)。但是，如果开启了 **AD16C6T_BDCFG** 寄存器的 OFFSSI 位，并且 CCnEN 与 CCnNEN 位不全为 0，那么通道会先输出无效电平，并且在死区延迟后，改为输出其空闲电平(根据 **AD16C6T_CON2** 寄存器的 OISSn 及 OISSnN 位决定)。

所有输出引脚控制状态整理如下表:

<table>
<tr><th colspan="5">控制位</th><th colspan="2">输出状态</th></tr>
<tr><th>GOEN</th><th>OFFSSI</th><th>OFFSSR</th><th>CCnEN</th><th>CCnNEN</th><th>CHn</th><th>CHnN</th></tr>
<tr><td rowspan="6">1</td><td rowspan="6">x</td><td>x</td><td>0</td><td>0</td><td colspan="2">禁止引脚输出(Hi-Z，不由定时器控制)</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>禁止引脚输出(Hi-Z，不由定时器控制)</td><td>CHnN = CHnREF + 极性(CCnNPOL)</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>CHn = CHnREF + 极性(CCnPOL)</td><td>禁止引脚输出(Hi-Z，不由定时器控制)</td></tr>
<tr><td>x</td><td>1</td><td>1</td><td>CHn = CHnREF + 极性(CCnPOL) + 死区(DT)</td><td>CHnN = ~CHnREF + 极性(CCnNPOL) + 死区(DT)</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>运行模式下关闭状态(输出无效电平)
CHn = 极性(CCnPOL)</td><td>CHnN = CHnREF + 极性(CCnNPOL)</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>CHn=CHnREF + 极性(CCnPOL)</td><td>运行模式下关闭状态(输出无效电平)
CHnN=极性(CCnNPOL)</td></tr>
<tr><td rowspan="3">0</td><td>0</td><td rowspan="3">x</td><td>x</td><td>X</td><td colspan="2" rowspan="2">禁止引脚输出(Hi-Z，不由定时器控制)</td></tr>
<tr><td rowspan="2">1</td><td>0</td><td>0</td></tr>
<tr><td>0</td><td>1</td><td colspan="2">空闲模式下关闭状态(输出无效电平)</td></tr>
</table>

控制位					输出状态	
GOEN	OFFSSI	OFFSSR	CCnEN	CCnNEN	CHn	CHnN
			1	0	CHn = 极性(CCnPOL), CHnN = 极性(CCnNPOL)	
			1	1	在死区(DT)时间后，输出空闲电平 CHn = OISSn, CHnN = OISSnN 注:禁止设置 OISSn 与 OISSnN 皆为有效电平	

表 21-11 输出引脚状态表

20. 5. 24 生成 6 步PWM

配置 **AD16C6T_CON2** 寄存器的 CCPCEN 位可实现 CHnMOD、CCnEN、CCnNEN 位的预装载功能。而 CCUSEL 位可以选择预装载事件的来源。当 CCUSEL 位为 1 时，预装载位会在 COM 事件时载入到缓冲寄存器中。这使得用户可以预先编程下一个步骤的配置，并在同一时间更新所有通道的配置。COM 事件可以通过软件设置 **AD16C6T_SGE** 寄存器中的 SGCOM 位产生，也可以通过硬件（在 TRGI 上升沿）产生。

当发生 COM 事件时，硬件会自动设置 **AD16C6T_RIF** 寄存器的 COM 位为 1。开启 **AD16C6T_IER** 寄存器的 COM 位可以产生中断。开启 **AD16C6T_DMAEN** 寄存器的 COM 位可以产生 DMA 请求。

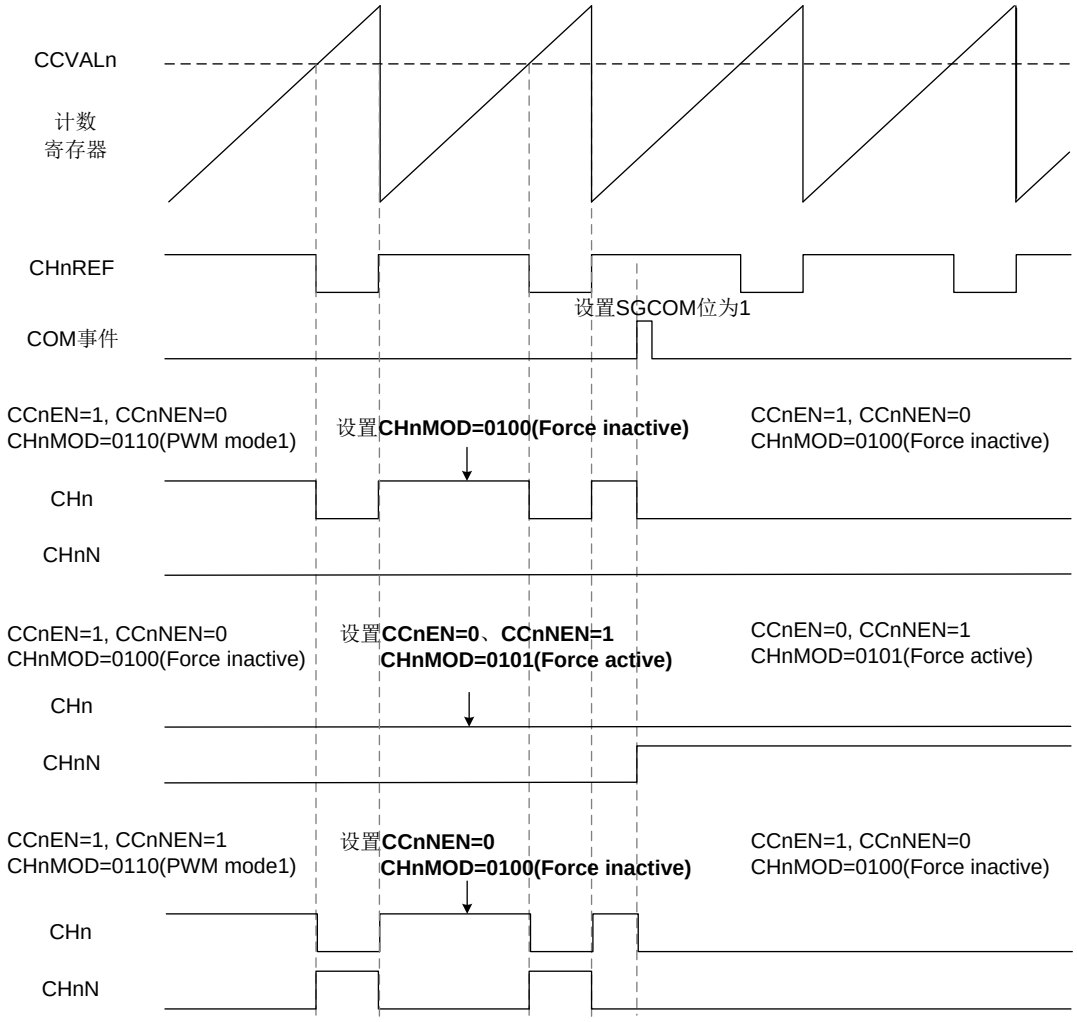


图 21-47　COM 事件生成 6 步 PWM(OFFSSR=1)

20. 5. 25 编码器接口

20. 5. 25. 1　正交编码器模式

正交编码器模式有以下五种计数方式:

- 正交编码器模式 1(x2): 设置 **AD16C6T_SMCON** 寄存器的 SMODS 位为 00001b，计数器根据 I2 电平在 I1 边沿递增或递减计数。
- 正交编码器模式 2(x2): 设置 **AD16C6T_SMCON** 寄存器的 SMODS 位为 00010b，计数器根据 I1 电平在 I2 边沿递增或递减计数。
- 正交编码器模式 3(x4): 设置 **AD16C6T_SMCON** 寄存器的 SMODS 位为 00011b，计数器根据 I2/I1 电平在 I1/I2 边沿递增或递减计数。
- 正交编码器模式 4(x1): 设置 **AD16C6T_SMCON** 寄存器的 SMODS 位为 01110b，计数器根据 I2 指定电平在 I1 边沿递增或递减计数。
- 正交编码器模式 5(x1): 设置 **AD16C6T_SMCON** 寄存器的 SMODS 位为 01111b，计数器根据 I1 指定电平在 I2 边沿递增或递减计数。

注:倍数是指在一个编码器输入信号的周期内，计数器可计算的次数。

可以通过设置 **AD16C6T_CCEP** 寄存器中的 CC1POL 和 CC2POL 位数值，选择正交编码器的输入 I1 和 I2 的极性(CC1NPOL 和 CC2NPOL 位必须置 0)。如果需要，也可以设置输入滤波器。

在正交编码器模式下，CH1 和 CH2 被用作输入接口。当计数器启动时(将 AD16C6T_CON1 寄存器的 CNTEN 位设置为 1)，计数器时钟由 I1 和 I2 经过滤波后的有效电平 I1FP 和 I2FP 转换提供。I1FP 和 I2FP 转换序列会产生计数脉冲和方向信号。计数器的递增或递减由信号的转换序列决定，而 **AD16C6T_CON1** 寄存器的 DIRSEL 方向位则由硬件自动更新。

正交编码器模式的工作方式类似于带有方向选择的外部时钟。计数器会在 0 到 **AD16C6T_AR** 寄存器中的自动重载值之间进行连续计数。因此，在开始计数之前需要设置 **AD16C6T_AR** 寄存器的值。在这种模式下，捕获器、预分频器、重复计数器和触发输出的功能都可以正常工作。需要注意的是，编码模式和选择外部时钟源 2 不兼容，因此不能同时选择。

此模式下，计数器会根据正交编码器的速度和方向自动修改，计数器的数值反映的是编码器的位置。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):

正交编码器模式	SMODS	有效边沿	有效边沿的相对信号电平(*注)	I1FP 上升沿	I1FP 下降沿	I2FP 上升沿	I2FP 下降沿
模式 1(x2)	00001b	仅在 I1 计数	高电平	递减	递增	不计数	不计数
			低电平	递增	递减	不计数	不计数
模式 2(x2)	00010b	仅在 I2 计数	高电平	不计数	不计数	递增	递减
			低电平	不计数	不计数	递减	递增
模式 3(x4)	00011b	在 I1 和 I2 上计数	高电平	递减	递增	递增	递减
			低电平	递增	递减	递减	递增
模式 4(x1)	01110b	仅在 I1 计数	高电平	递减	递增	不计数	不计数+
			低电平	不计数	不计数	不计数	不计数
模式 5(x1)	01111b	仅在 I2 计数	高电平	不计数	不计数	递增	递减
			低电平	不计数	不计数	不计数	不计数

表 21-12　计数方向与编码器输入信号的关系(CC1POL = CC2POL = 0)

注:
有效边沿的相对信号电平指的是:
在 I1FP 上升沿或下降沿计数时，计数方向由 I2FP 的电平高或低决定。
在 I2FP 上升沿或下降沿计数时，计数方向由 I1FP 的电平高或低决定。

正交编码器可直接与 MCU 连接，但一般会通过比较器将正交编码器的差分输出转换为数字信号。下图给出了计数信号产生和方向控制的例子:

配置如下:

1. 设置 **AD16C6T_CHMR1** 寄存器的 CC1SSEL 位为 01b，选择通道为 I1 输入。
2. 设置 **AD16C6T_CHMR1** 寄存器的 CC2SSEL 位为 01b，选择通道为 I2 输入。
3. 设置 **AD16C6T_CCEP** 寄存器的 CC1POL 位为 0、CC1NPOL 为 0，选择非反相输入。
4. 设置 **AD16C6T_CCEP** 寄存器的 CC2POL 位为 0、CC2NPOL 为 0，选择非反相输入。
5. 设置 **AD16C6T_SMCON** 寄存器的 SMODS 位为 00011b，选择计数器同时根据 I1 和 I2 上的边沿计数。
6. 设置 **AD16C6T_CON1** 寄存器 CNTEN 位为 1 开启计数器。

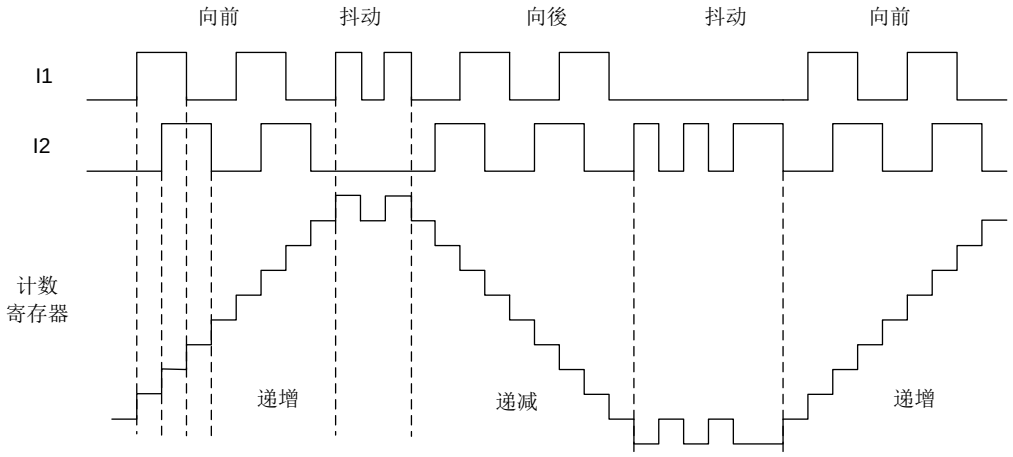


图 21-48　编码器接口模式下的计数操作

下图给出了计数器在 I1 滤波信号极性反相时的计数过程(除了设置 CC1POL 位为 1，其他设置与上面一致):

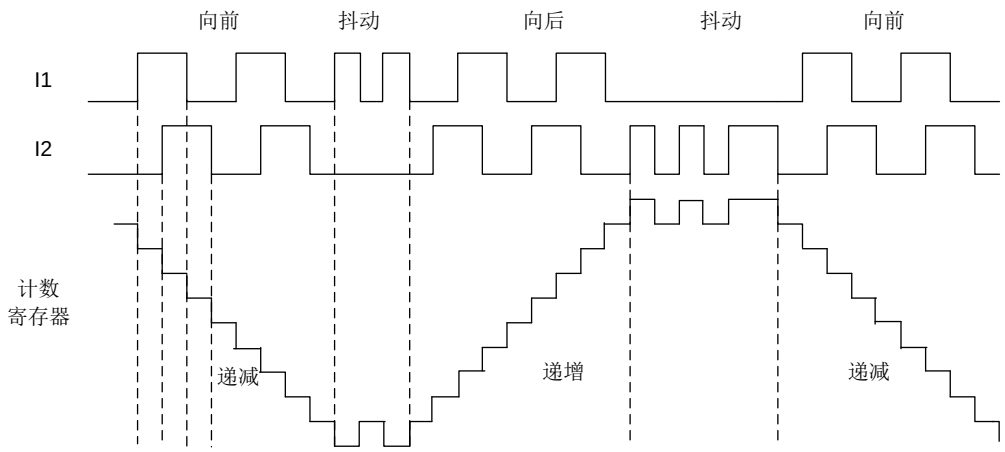


图 21-49　滤波后极性反相时编码器接口例子

下图为不同正交编码器模式下的计数方式(AR = 08h):

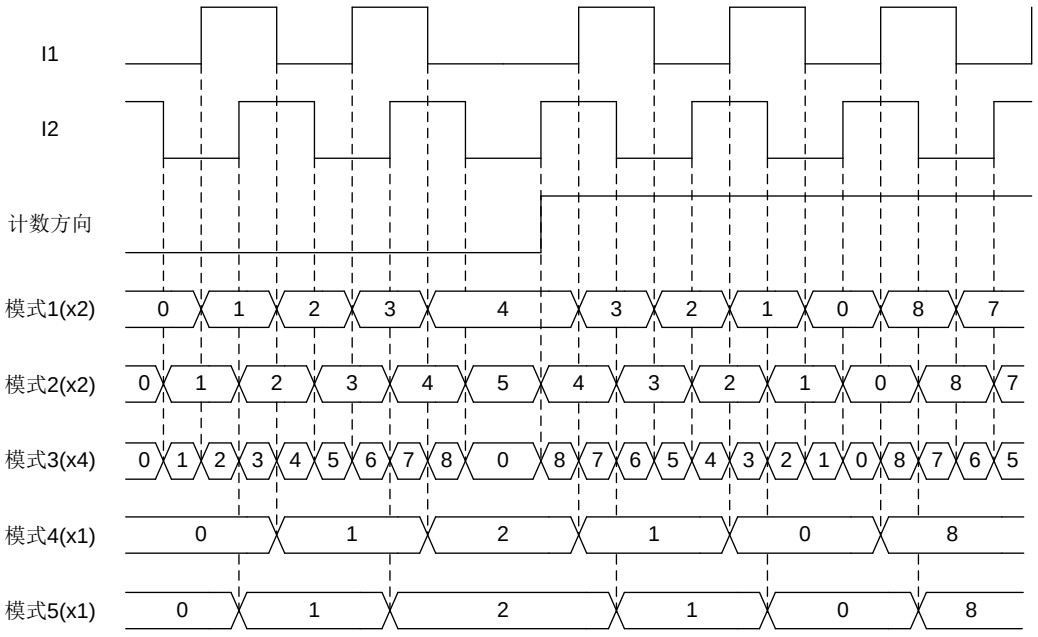


图 21-50　不同正交编码器模式下的计数方式

除了正交编码器模式外，定时器还提供了以下两种编码器模式:时钟加方向编码器模式和定向时钟编码器模式。

20. 5. 25. 2　时钟加方向编码器模式

在此模式下，定时器计数方向(递增/递减)由 I1 的电位控制，而计数时钟由 I2 的上升沿/下降沿控制，整理如下:

CC1POL	I1 电平	计数方向
0	高电平	递增
	低电平	递减
1	高电平	递减
	低电平	递增

表 21-13　计数方向与极性及 I1 电平之间的关系

时钟加方向编码器模式	SMODS	CC2POL	I2 上升沿	I2 下降沿
模式 1(x2)	01010b	x	计数	计数
模式 2(x1)	01011b	0	计数	不计数
		1	不计数	计数

表 21-14　计数时钟与极性及 I2 边沿之间的关系

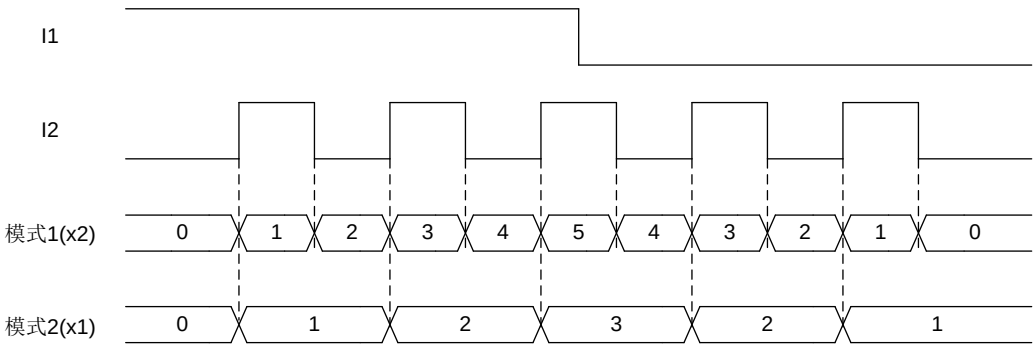


图 21-51　时钟加方向编码器模式(CC1POL = 0, CC2POL = 0)

20. 5. 25. 3　定向时钟编码器模式

与时钟加方向编码器模式不同的是，此模式下 I1/I2 皆用来控制定时器计数，I1 上升沿或下降递减计数，I2 上升沿或下降沿递增计数，整理如下:

定向时钟编码器模式	SMODS	CCnPOL	有效边沿的相对信号电平	I1FP 上升沿	I1FP 下降沿	I2FP 上升沿	I2FP 下降沿
模式 1(x2)	01100b	0	高电平	递减	递减	递增	递增
			低电平	不计数	不计数	不计数	不计数
		1	高电平	不计数	不计数	不计数	不计数
			低电平	递减	递减	递增	递增
模式 2(x1)	01101b	0	高电平	不计数	递减	不计数	递增
			低电平	不计数	不计数	不计数	不计数
		1	高电平	不计数	不计数	不计数	不计数

			低电平	递减	不计数	递增	不计数

表 21-15　计数方向与编码器输入信号的关系

CC1POL = 0, CC2POL = 0

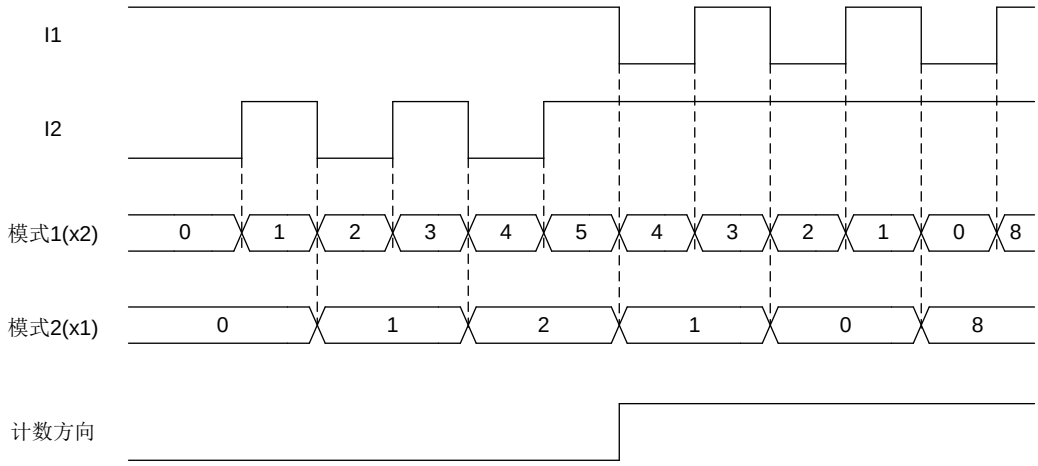


图 21-52　定向时钟编码器模式(CC1POL = 0, CC2POL = 0)

CC1POL = 1, CC2POL = 1

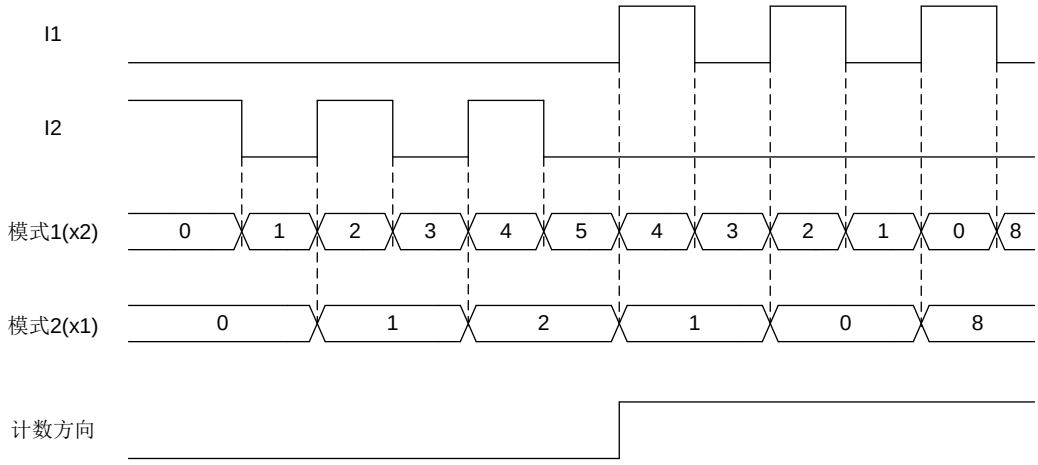


图 21-53　定向时钟编码器模式(CC1POL = 1, CC2POL = 1)

20. 5. 25. 4　正交编码器模式下的索引(Index)输入

编码器的第三个输出(Z 相/Index)用于指示机械零点，可以连接到 ETR 引脚，用来触发计数器复位。

设置 **AD16C6T_ENCR** 寄存器的 IDXEN 位为 1，开启索引侦测的功能。此功能只适用在 **AD16C6T_SMCON** 寄存器的 SMODS 设定为编码器模式(0001b、0010b、0011b、1010b、1011b、1100b、1101b、1110b、1111b)时。

大部分的编码器都具备有调整索引脉冲宽度的选项，主要有以下三种模式:

- AB 相门限模式:索引脉冲在 A、B 相的两个有效边沿对齐，宽度为 1/4 编码器输入周期。
- A 相(或 B 相)门限模式:索引脉冲在 A(或 B 相)的两个有效边沿对齐，宽度为 1/2 编码器输入周期。
- 非门限模式:索引脉冲不与 A、B 相的两个有效边沿对齐，宽度为 1 个编码器输入周期。

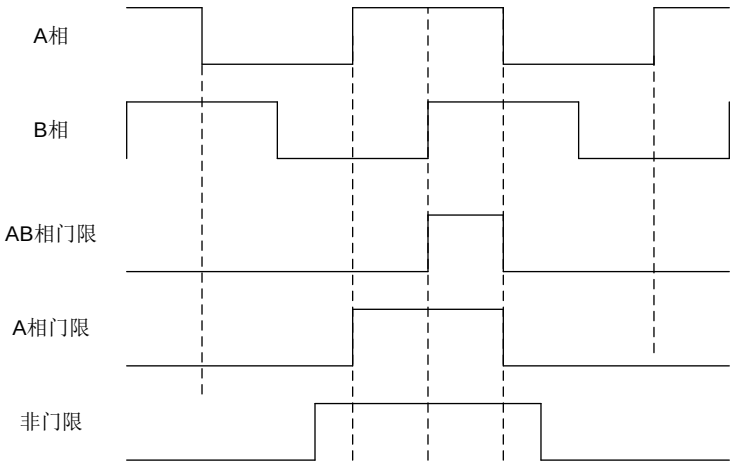


图 21-54　索引脉冲宽度

索引脉冲的抖动是可以被允许的，但抖动范围不能超过两个编码器输入周期，否则会造成计数器多次重置。

通过 **AD16C6T_ENCR** 寄存器的 IDXPOS 位，可以定义索引信号在哪个 AB 相位为有效输入，因其物理特性的关系，索引只会永远出现在其中一种相位上(假设使用的是 AB 相门限)。

下图为编码器正转时的 AB 相变化，与索引出现在不同相位时，所需要对应的 IDXPOS 位设置。

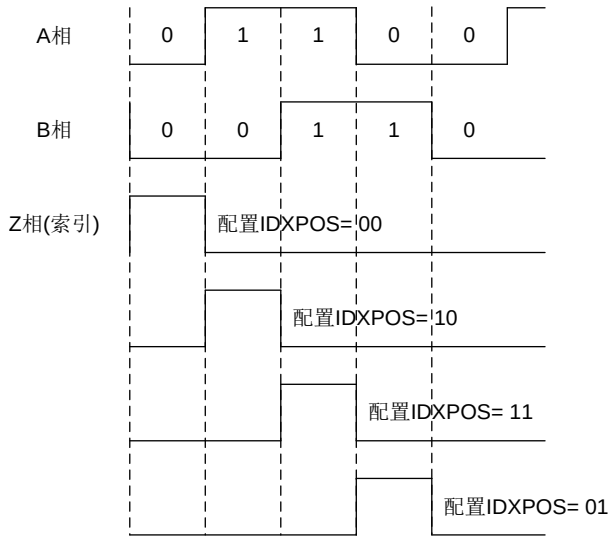


图 21-55　有效索引与 IDXPOS 位之关系图

下图为 AB 相门限索引(IDXPOS 位设定为 11b)有效时，计数器复位情形:

- 递增计数下 AB 相为 11 且侦测到有效索引时，重置计数器为 0，并产生索引中断(IDX)(若有开启 AD16C6T_IER 寄存器的 IDX 位)。
- 递减计数下 AB 相为 11 且侦测到有效索引时，在 AB 相不为 11 时，重置计数器为 AR，并产生索引中断(IDX)(若有开启 **AD16C6T_IER** 寄存器的 IDX 位)。

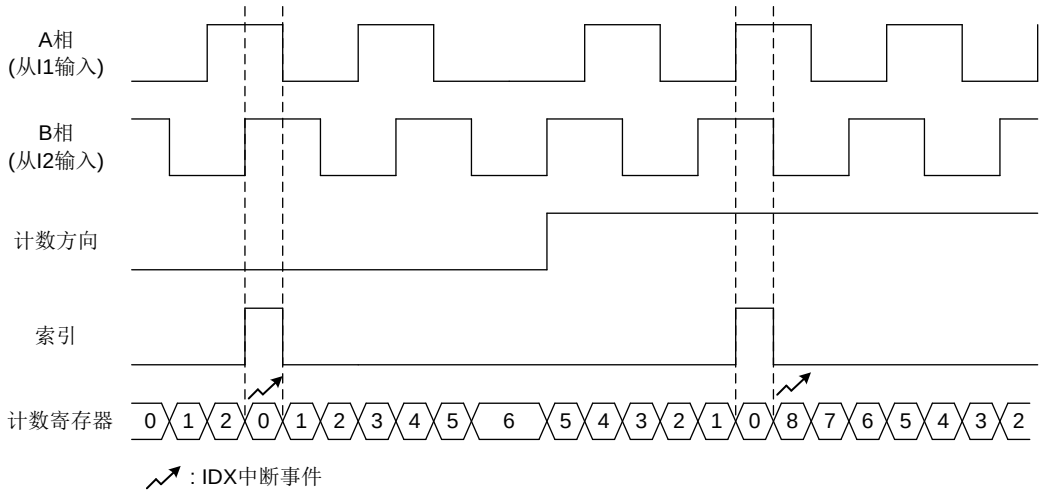


图 21-56　计数器数值与 AB 相门限索引(IDXPOS = 11b)

下图为非相门限索引(IDXPOS 位设定为 00b)有效时，计数器复位情形:

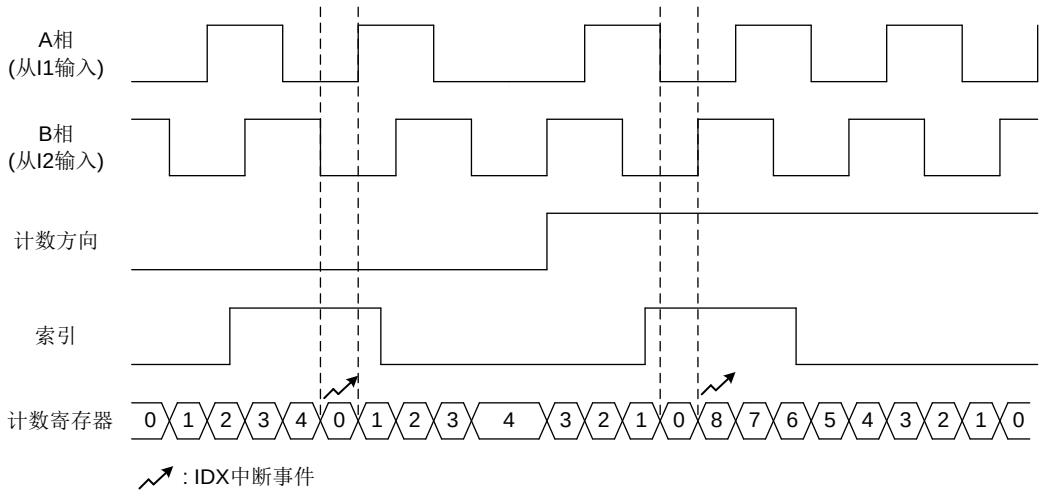


图 21-57　计数器数值与非门限索引(IDXPOS = 00b)

下图为索引因抖动造成脉冲宽度小于 AB 相门限时的波型图:

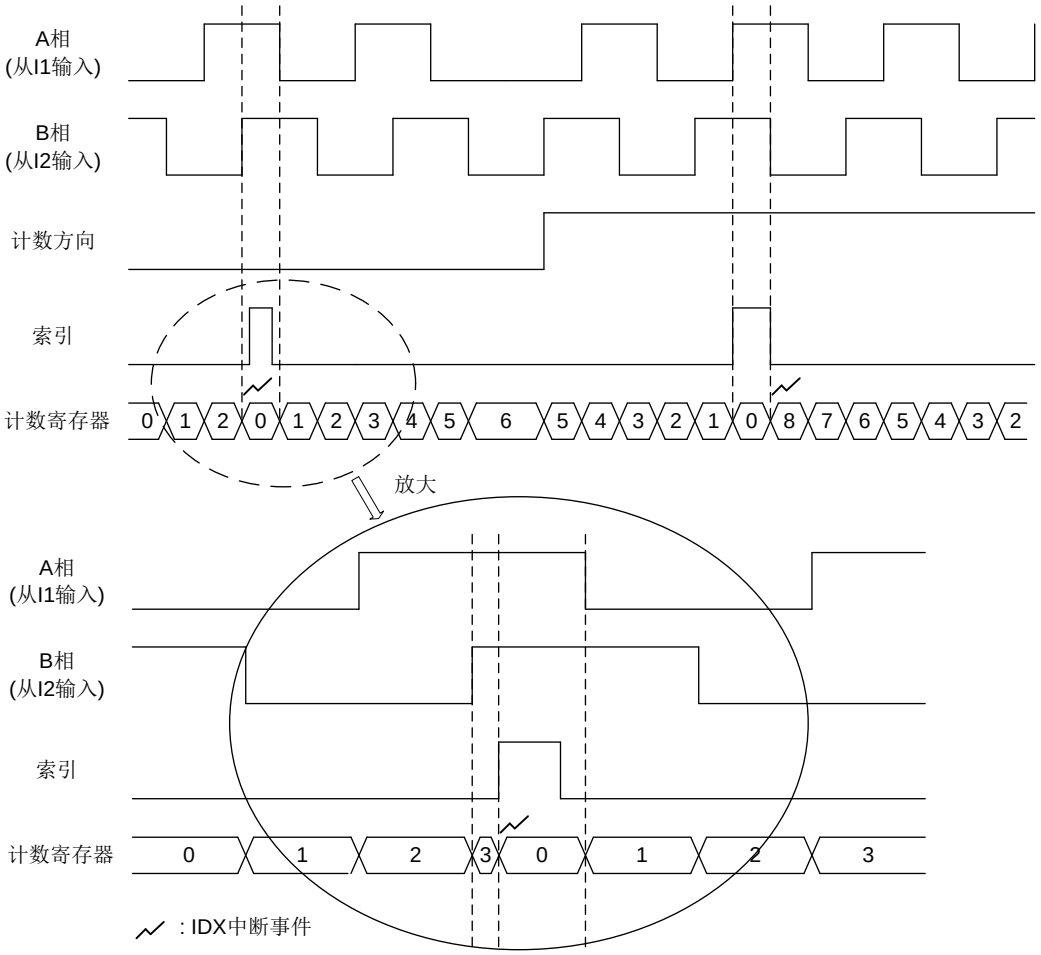


图 21-58　索引脉冲宽度小于 AB 相门限(IDXPOS = 11b)

下图为 B 相门限索引(IDXPOS 位设定为 01b)有效时，在 x2 模式与 x1 模式下计数器复位情形:

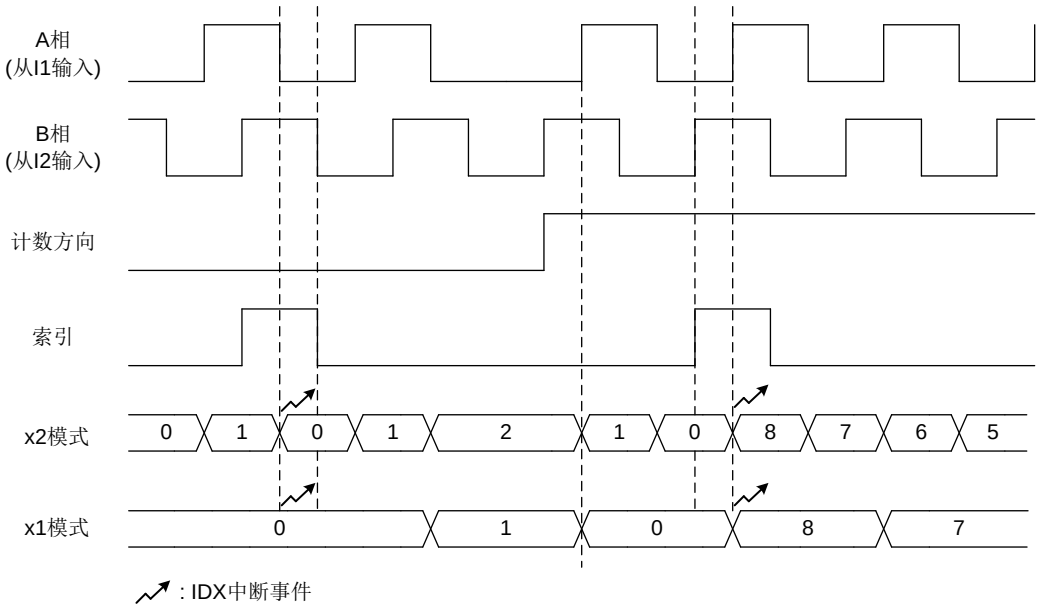


图 21-59　计数器数值与 B 相门限索引(IDXPOS = 01b)

特定计数方向下检测索引有效事件

设置 **AD16C6T_ENCR** 寄存器的 IDXDIR 位，可以在特定计数方向下检测索引有效事件。下图为 IDXDIR 位与有效索引事件在不同计数方向下的关系图:

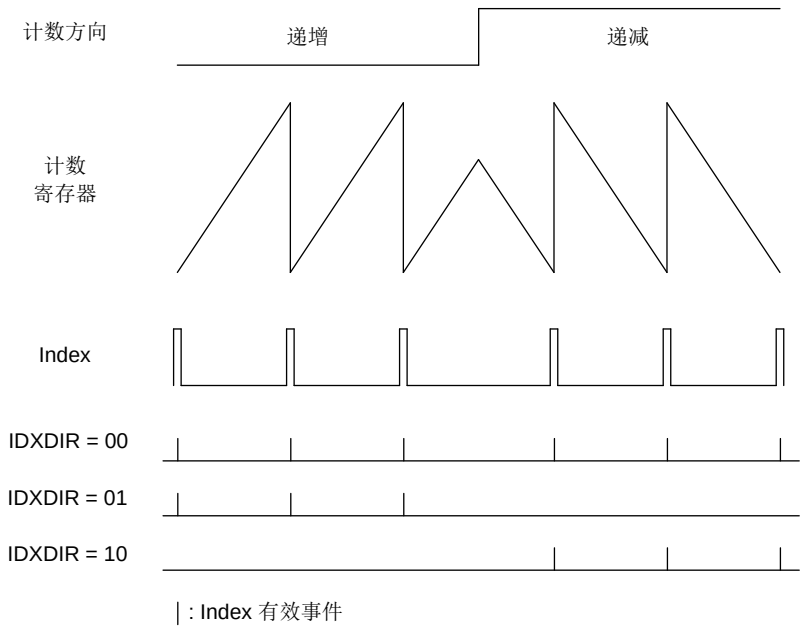


图 21-60　特定计数方向下检测索引有效事件

注:

1. 必须在索引检测功能关闭(AD16C6T_ENCR 寄存器的 IDXEN 位为 0)时，才能改变 IDXDIR 位。
2. 此功能在时钟加方向编码器模式下不支持，IDXDIR 必须设置为 0。

首次索引事件检测

设置 **AD16C6T_ENCR** 寄存器的 FIDX 位为 1，可以只检测第一次的索引为有效事件，让后续

的索引皆为无效事件。若有需要，可以将 FIDX 位写 0 再写 1，重启首次索引事件检测机制。

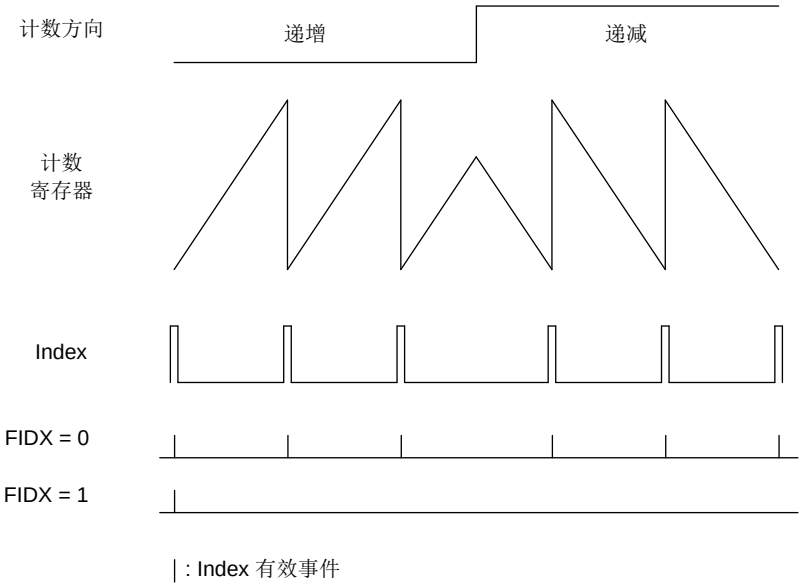


图 21-61　首次索引事件检测

20. 5. 25. 5　非正交编码器模式下的索引输入

设定成时钟加方向编码器模式(SMODS 位=01010b/01011b)或定向时钟编码器模式(SMODS 位=01100b/01101b)时，IDXPOS[1]位无效:

- IDXPOS[0] = 0:当时钟低电平时，有效的索引事件会重置计数器。
- IDXPOS[0] = 1:当时钟高电平时，有效的索引事件会重置计数器。

下图为时钟加方向编码器模式在有效索引事件发生时，计数器复位情形:

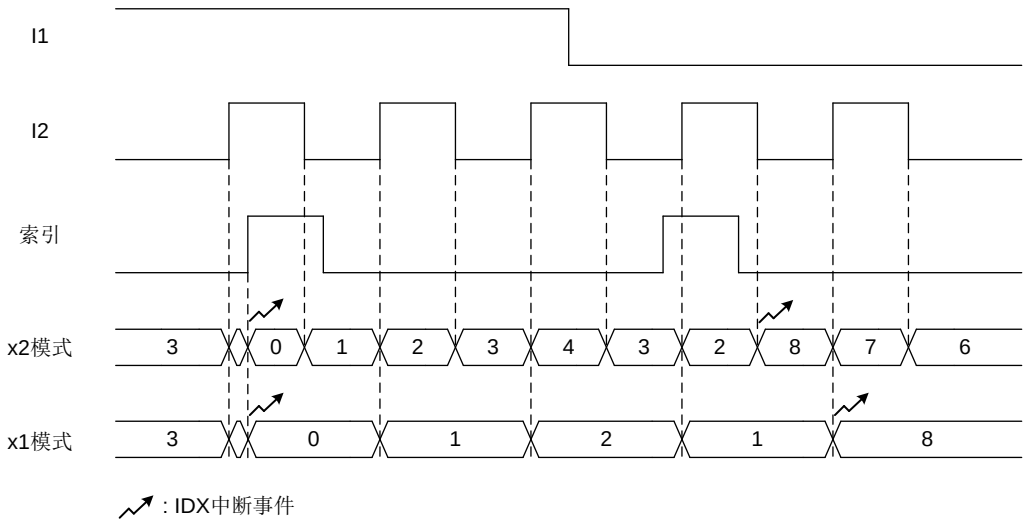


图 21-62　时钟加方向编码器模式下检测索引有效事件(IDXPOS[0] = 1)

下图为定向时钟编码器模式在有效索引事件发生时，计数器复位情形:

CC1POL = 0, CC2POL = 0

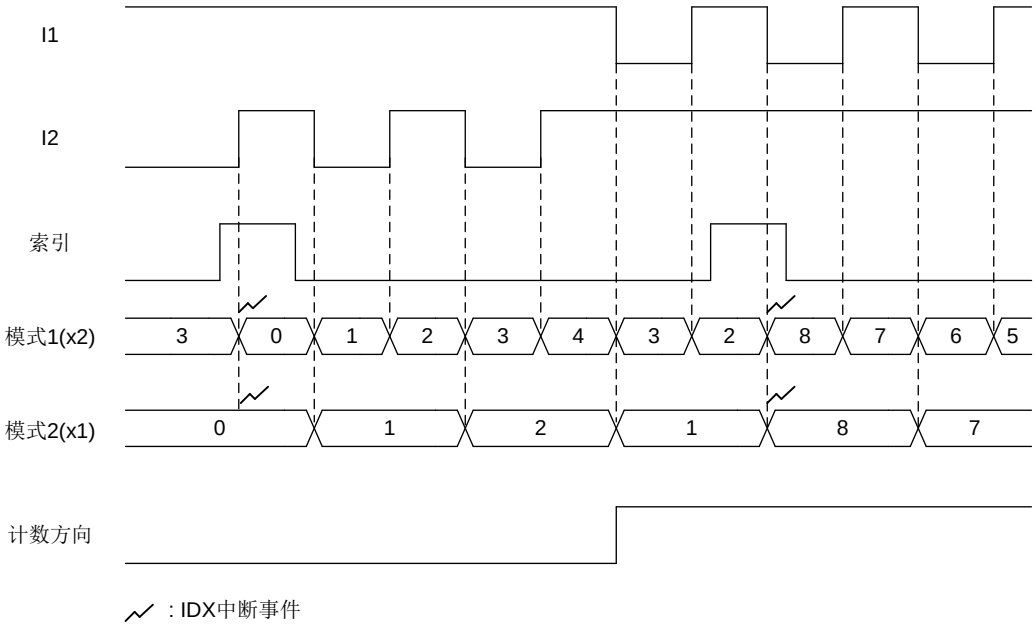


图 21-63　定向时钟编码器模式下检测索引有效事件(IDXPOS[0] = 1)

20. 5. 25. 6　运行模式下切换从模式选择

当用户在定时器运行中要更改编码器模式时，建议设置 **AD16C6T_SMCON** 寄存器的 SMODPE 位为 1，开启 SMODS 的预装载功能，并设置 SMODPS 位决定缓冲寄存器的更新时机:

- 设置 SMODPS 位为 0:发生更新事件(UPD)时，将预装载寄存器内的值更新到缓冲寄存器中。
- 设置 SMODPS 位为 1:发生索引事件(IDX)时，将预装载寄存器内的值更新到缓冲寄存器中。

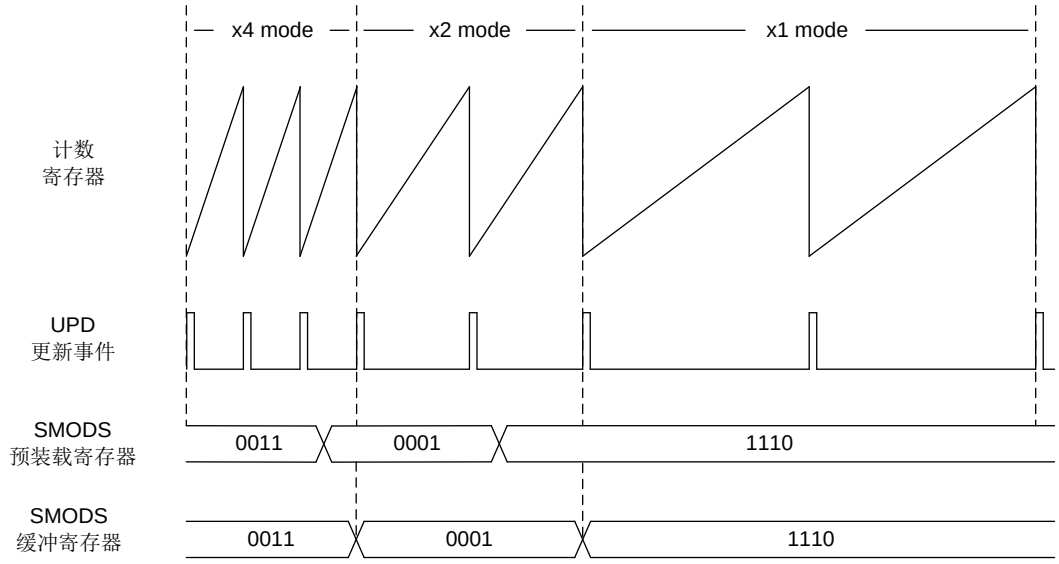


图 21-64　运行模式下切换从模式选择(SMODPS = 0)

20. 5. 25. 7　编码器时钟输出

编码器模式的工作原理不太适合用来做速度量测，因为要比较长的时间来获取足够多的时钟周期。比较好的应用方式是通过从模式控制器，将编码器的时钟输出到从定时器，再由从定时器以检测边沿到边沿的方式，执行周期量测并计算间隔以提供速度等相关信习。

设置 **AD16C6T_CON2** 寄存器的 MMSEL 位为 1000b 时，可以将编码器时钟输出到 TRGOUT。此模式只适用在 AD16C6T_SMCON 寄存器的 SMODS 设定为编码器模式(0001b、0010b、0011b、1010b、1011b、1100b、1101b、1110b、1111b)时。

20. 5. 26 方向位输出

设置 **AD16C6T_CHMR2** 寄存器的 CH3MOD/CH4MOD 位为 01011b，可将 **AD16C6T_CON1** 寄存器的 DIRSEL 位映射到 CH3/CH4 通道输出，主要用来监测中心对齐计数时的方向，或编码器模式下的方向变换。

20. 5. 27 UPD位重映射

通过设置 **AD16C6T_CON1** 寄存器的 UPDREMAP 位，可以将 UPD 原始中断状态映射至 **AD16C6T_COUNT** 寄存器的第 31 位 UPDRIF 中。

注:UPD 和 UPDRIF 标志之间没有延迟。

20. 5. 28 输入XOR功能

通过 **AD16C6T_CON2** 寄存器的 I1SEL 位，可将通道 1 的输入滤波器连接到 XOR 门的输出端，XOR 门输入端包含 CH1、CH2 和 CH3 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。

20. 5. 29 霍尔传感器界面

本文介绍使用AD16C6T定时器产生PWM信号驱动马达，以及用另一个通用定时器(GP32C4T)作为"接口定时器"来连接霍尔传感器的方法。具体操作是将 3 个定时器输入脚(CH1、CH2 和 CH3)通过 XOR 门连接到 I1 输入通道，并由接口定时器捕获这个信号。为了产生一个由霍尔输入端的任何变化而触发的时间基准，模式控制器被设置为复位模式，输入为 I1F 双边沿。每当 3 个输入之一变化时，计数器就会从 0 重新开始计数。

在接口定时器模式下，通道 1 被设置为捕获模式，捕获信号源为 I1。捕获值反映了输入端两次变化之间的时间间隔，提供与马达转速相关的讯息。接口定时器还可以在输出模式产生一个脉冲，这个脉冲通过 TRGOUT 输出到 AD16C6T 定时器，用来生成 COM 事件，改变 AD16C6T 定时器各个通道的配置，进而产生 PWM 信号驱动马达。

以下图为例:

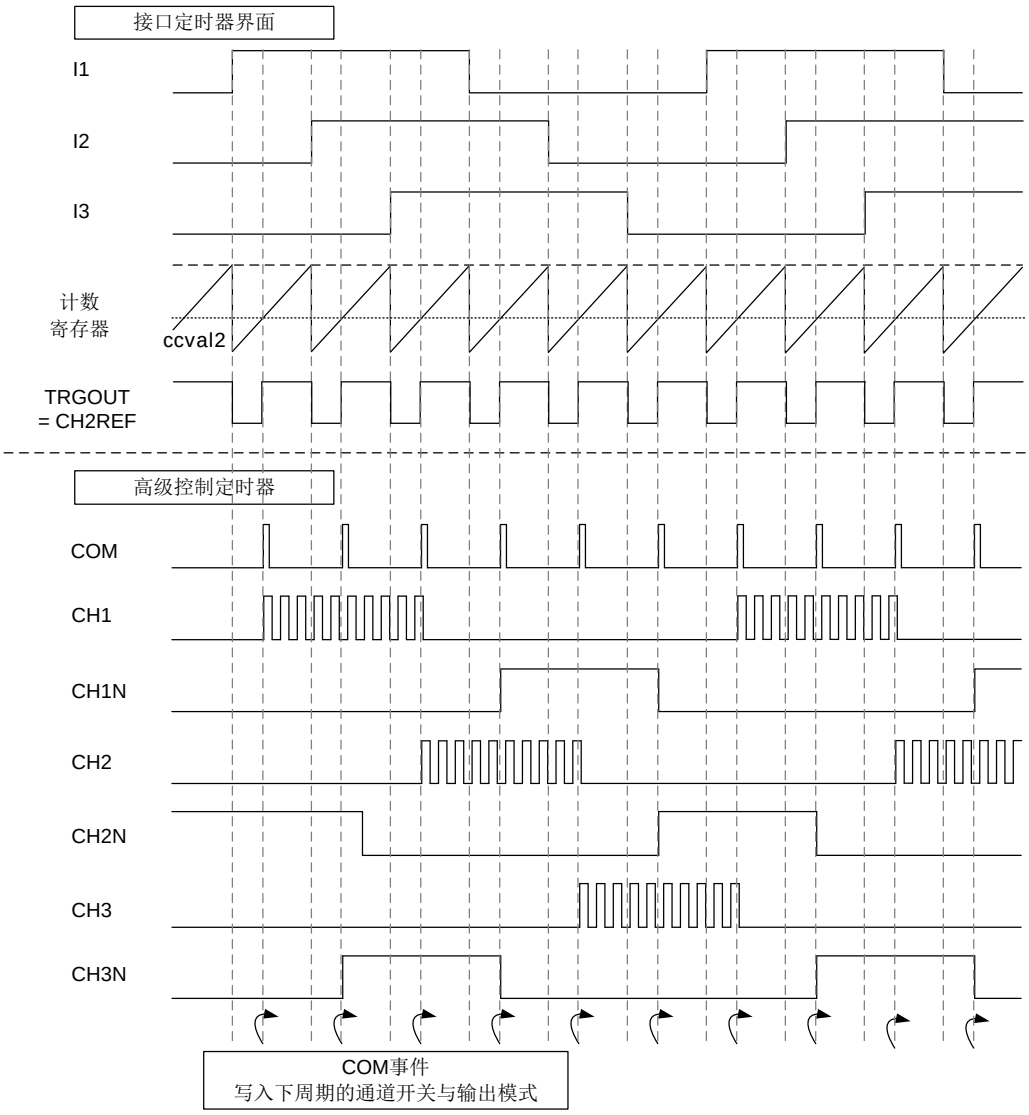


图 21-65　霍尔传感器接口范例

- 设置 GP32C4T_CON2 寄存器的 I1SEL 位为 1，通过将三个定时器输入进行 XOR 运算后

输入 I1 通道。

- 设置 GP32C4T_AR 为其最大值，以便计数器在 I1 变化时被清零。同时，设置预分频器以得到一个计数器周期，该周期长于传感器两次变化的时间间隔。
- 设置 GP32C4T_CHMR1 寄存器的 CC1SSEL 位为 01b，选择 I1 作为捕获输入通道。如果需要，还可以设置数字滤波器。
- 设置 GP32C4T_CHMR1 寄存器的 CC2SEL 位为 00b，选择通道 2 作为输出通道。
- 设置 GP32C4T_CHMR1 寄存器的 CH2MOD 位为 0111b，设置通道 2 为 PWM 模式 2。
- 设置 GP32C4T_CCVAL2 寄存器决定 PWM 输出延迟时间。
- 设置 GP32C4T_CON2 寄存器的 MMSEL 位为 101b，选择 CH2REF 作为 TRGOUT 上的触发输出。

在 AD16C6T 定时器中，TRGI 需选择 ITn 做为触发事件源，定时器被配置为产生 PWM 信号，并开启捕获或比较预装载控制(设置 **AD16C6T_CON2** 寄存器的 CCPCEN 位为 1)，COM 事件由触发输入控制(设置 **AD16C6T_CON2** 寄存器的 CCUSEL 位为 1)。发生 COM 事件后(TRGI 侦测到上升沿)，在 PWM 控制位(CCnEN、CHnMOD)中写入下一步的配置，此操作可在由 CH2REF 上升沿产生的中断子程序中完成。

20. 5. 30 外部触发的同步

AD16C6T 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。

20. 5. 30. 1 复位模式

当侦测到有效触发输入事件时，计数器及其预分频器会被重新初始化，若此时 **AD16C6T_CON1** 寄存器的 UERSEL 位为 0，则一并产生更新事件 UPD。而所有预装载寄存器(**AD16C6T_AR**，**AD16C6T_CCVALn**)都会因更新事件 UPD 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零，配置过程如下:

1. 设置 **AD16C6T_CHMR1** 寄存器的 CC1SSEL 位为 01b，选择 I1 为有效输入端。
2. 设置 **AD16C6T_CHMR1** 寄存器的 I1FLT 位为 0000b，本例无需滤波器。
3. 设置 **AD16C6T_CCEP** 寄存器的 CC1NPOL 位为 0、CC1POL 位为 0，选择 I1 通道上升沿有效。
4. 设置 **AD16C6T_CHMR1** 寄存器的 I1PRES 位为 00b，捕获预分频器不用于触发操作，无需设置。
5. 设置 **AD16C6T_SMCON** 寄存器的 TSSEL 位为 00101b，选择 I1 滤波后信号作为输入源。
6. 设置 **AD16C6T_SMCON** 寄存器的 SMODS 位为 0100b，选择复位模式。
7. 设置 **AD16C6T_CON1** 寄存器的 CNTEN 位为，开启计数器。

计数器依据内部时钟开始计数，计数器计数直到 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。同时设置标志位为 1(**AD16C6T_RIF** 寄存器的 TRGI 位)，如果中断及 DMA 开启(取决于 **AD16C6T_IER** 寄存器的 TRGI 位和 **AD16C6T_DMAEN** 寄存器的 TRGI 位)，会发送中断及 DMA 请求。

下图给出了设置自动重载寄存器 **AD16C6T_AR** 为 36h 时的信号变化。由于 I1 输入的同步电路，I1 上的上升沿和计数器实际初始化之间会存在延迟。

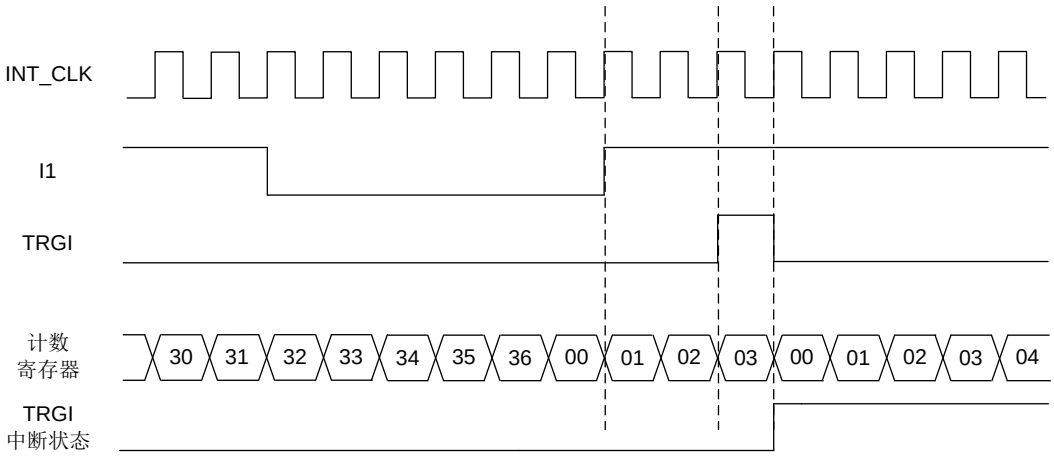


图 21-66　复位模式控制电路

20. 5. 30. 2　门控模式

计数器根据选中的输入电平被开启。

下面的例子中，计数器只在 I1 输入为低电平时才递增计数:

1. 设置 **AD16C6T_CHMR1** 寄存器的 CC1SSEL 位为 01b，选择 I1 为有效输入端。
2. 设置 **AD16C6T_CHMR1** 寄存器的 I1FLT 位为 0000b，本例无需滤波器。
3. 设置 **AD16C6T_CCEP** 寄存器的 CC1NPOL 位为 0、CC1POL 位为 1，I1 通道反相，有效极性为低电平。
4. 设置 **AD16C6T_CHMR1** 寄存器的 I1PRES 位为 00b，捕获预分频器不用于触发操作，无需设置。
5. 设置 **AD16C6T_SMCON** 寄存器的 TSSEL1 位与 TSSEL2 位为 00101b，选择 I1 滤波后信号作为输入源。
6. 设置 **AD16C6T_SMCON** 寄存器的 SMODS 位为 101b，选择门控模式。
7. 设置 **AD16C6T_CON1** 寄存器的 CNTEN 位为 1，开启计数器(在门控模式中，如果 CNTEN 位为 0，无论触发输入为何电平，计数器都不会开启)。

只要 I1 为低电平，计数器依据内部时钟开始计数，一旦 I1 为高电平则停止计数。由于 I1 输入端的同步电路的原因，I1 上出现上升沿和计数器实际停止之间会有一定的延迟。

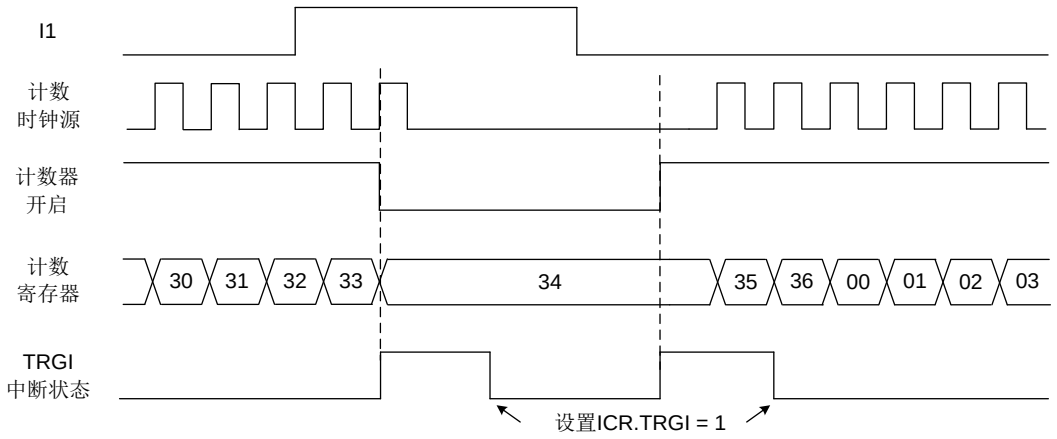


图 21-67　门控模式控制电路

20. 5. 30. 3　触发模式

输入端选中的事件可以开启计数器。

下面的例子中，I2 输入端上的上升沿可以开启递增计数:

1. 设置 **AD16C6T_CHMR1** 寄存器的 CC2SSEL 位为 01b，选择 I2 为有效输入端。
2. 设置 **AD16C6T_CHMR1** 寄存器的 I2FLT 位为 0000b，本例无需滤波器。
3. 设置 **AD16C6T_CCEP** 寄存器的 CC2NPOL 位为 0、CC2POL 位为 0，选择 I2 通道上升沿有效。
4. 设置 **AD16C6T_CHMR1** 寄存器的 I2PRES 位为 00b，捕获预分频器不用于触发操作，无需设置。
5. 设置 **AD16C6T_SMCON** 寄存器的 TSSEL1 位与 TSSEL2 位为 00110b，选择 I2 滤波后信号作为输入源。
6. 设置 **AD16C6T_SMCON** 寄存器的 SMODS 位为 110b，选择触发模式。
7. 设置 **AD16C6T_CON1** 寄存器的 CNTEN 位为，开启计数器。

I2 上出现上升沿时，计数器开始依据内部时钟计数并設置 TRGI 标志位为 1。

由于 I2 输入的同步电路原因，I2 上出现上升沿和计数器实际启动之间会有一定的延迟。

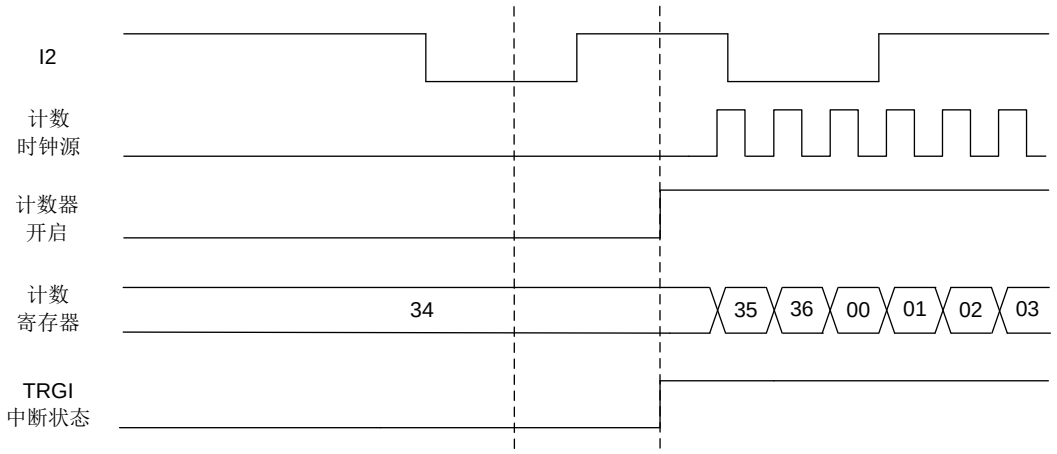


图 21-68　触发模式控制电路

20. 5. 30. 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(外部时钟模式 1 和除编码模式除外)。ETR 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)。不推荐设置 **AD16C6T_SMCON** 寄存器的 TSSEL1 位与 TSSEL2 位为 00111b 选择 ETR 作为 TRGI。

下面的例子中，当 I1 输入端出现上升沿时，开启计数器，并且在每个 ETR 信号的上升沿递增计数。

ETR 外部触发电路设定如下 ：

1. 设置 **AD16C6T_SMCON** 寄存器的 ETFLT 位为 000b，无需滤波器。
2. 设置 **AD16C6T_SMCON** 寄存器的 ETPRES 位为 00b，关闭预分频。
3. 设置 **AD16C6T_SMCON** 寄存器的 ETPOL 位为 0，ETR 的上升沿有效。
4. 设置 **AD16C6T_SMCON** 寄存器的 ECM2EN 位为 1，开启外部时钟模式 2。

通道 1 检测 I1 的上升沿，过程如下:

1. 设置 **AD16C6T_CHMR1** 寄存器的 CC1SSEL 位为 01b，选择 I1 为有效输入端。
2. 设置 **AD16C6T_CHMR1** 寄存器的 I1FLT 位为 0000b，本例无需滤波器。
3. 设置 **AD16C6T_CCEP** 寄存器的 CC1NPOL 位为 0、CC1POL 位为 0，选择 I1 通道上升沿有效。
4. 设置 **AD16C6T_CHMR1** 寄存器的 I1PRES 位为 00b，捕获预分频器不用于触发操作，无需设置。
5. 设置 **AD16C6T_SMCON** 寄存器的 TSSEL1 位与 TSSEL2 位为 00101b，选择 I1 滤波后信号作为输入源。
6. 设置 **AD16C6T_SMCON** 寄存器的 SMODS 位为 110b，选择触发模式。
7. 设置 **AD16C6T_CON1** 寄存器的 CNTEN 位为，开启计数器。

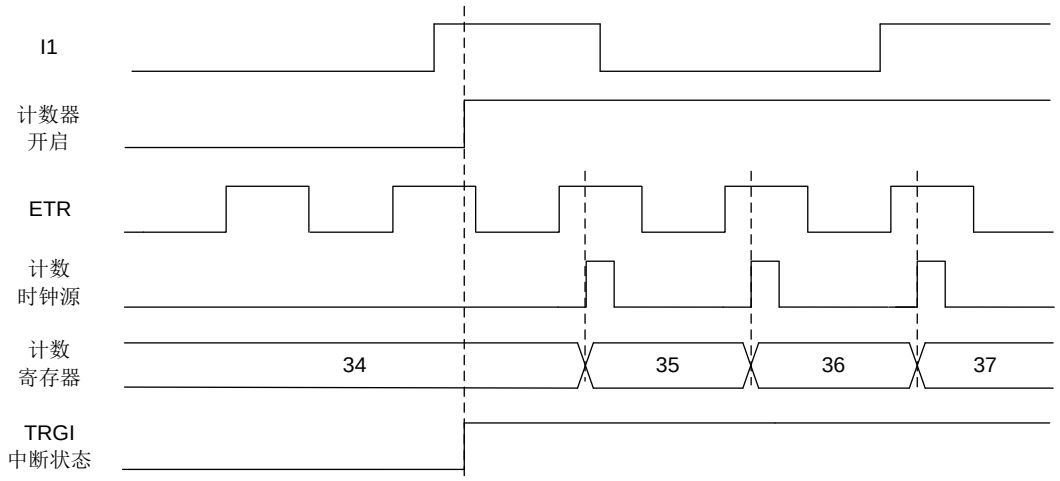


图 21-69　外部时钟源 2＋触发模式下的控制电路

I1 上出现上升沿时，计数器开启且设置 TRGI 标志位为 1，然后计数器根据 ETR 上的上升沿开始计数。

由于 ETF 输入同步电路的原因，ETR 信号的上升沿和实际计数器的计数会有延迟。

20. 5. 30. 5　组合复位模式 + 触发模式

要使用此模式，需要设置 **AD16C6T_SMCON** 寄存器的 SMODS 位为 1000b。其功能为复位模式加上触发模式，计数器在触发输入 TRGI 的上升沿启动，并重新初始化计数器以及生成更新事件。此模式主要用于单脉冲模式。

20. 5. 30. 6　组合门控模式 + 复位模式

要使用此模式，需要设置 **AD16C6T_SMCON** 寄存器的 SMODS 位为 1001b。其功能为门控模式加上复位模式，当触发输入(TRGI)为高电平时，计数器的时钟开启。一旦触发输入变为低电平，计数器停止且重新初始化。

20. 5. 31 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、开启、停止或提供时钟等操作。

下图显示了触发选择和主模选择模块的概况。

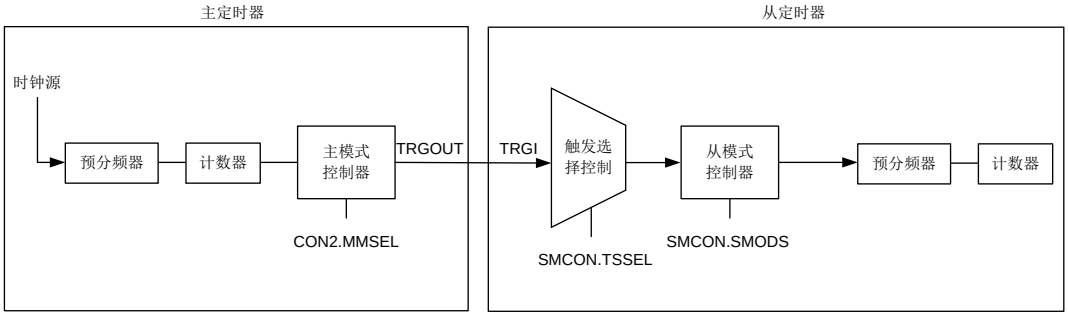


图 21-70　主/从定时器范例

20. 5. 31. 1　使用一个定时器去开启其他定时器

在这个例子中，定时器 2(AD16C6T)的开启由定时器 1(GP16C2T1)的输出比较参考信号(CH1REF)控制。只有当定时器 1 的 CH1REF 为高电平时，定时器 2 才会计数。

先设定从定时器(定时器 2)为门控模式，配置如下:

1. 设置 **AD16C6T_SMCON** 寄存器的 TSSEL1 位与 TSSEL2 位为 GP16C2T1，可参考内部触发连接表。
2. 设置 **AD16C6T_SMCON** 寄存器的 SMODS 位为 101b，选择门控模式。
3. 设置 **AD16C6T_CON1** 寄存器的 CNTEN 位为 1，开启计数器。

再设定主定时器(定时器 1)为 PWM 输出，配置如下:

1. 设置 **GP16C2T_PRES** 寄存器的 PSCV 为 01h，计数器时钟频率为 $f_{INT\_CLK/2}$。
2. 设置 **GP16C2T_CHMR1** 寄存器的 CH1MOD 位为 00111b，选择 PWM 模式 2。
3. 设置 **GP16C2T_CCEP** 寄存器的 CC1POL 位为 0，选择 CH1 通道输出为高电平有效。

4. 设置 **GP16C2T_CCVAL1** 寄存器的 CCRV1 位为 06h，当计数器数到 6 时，PWM 输出高电平。
5. 设置 **GP16C2T_AR** 寄存器的 ARV 位为 08h，当计数器递增到 8 后重载。
6. 设置 **GP16C2T_CON2** 寄存器的 MMSEL 位为 100b，选择输出比较参考信号(CH1REF)为触发输出(TRGOUT)。
7. 设置 **GP16C2T_CON1** 寄存器的 CNTEN 位为 1，开启计数器。

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 计数器的开启信号。

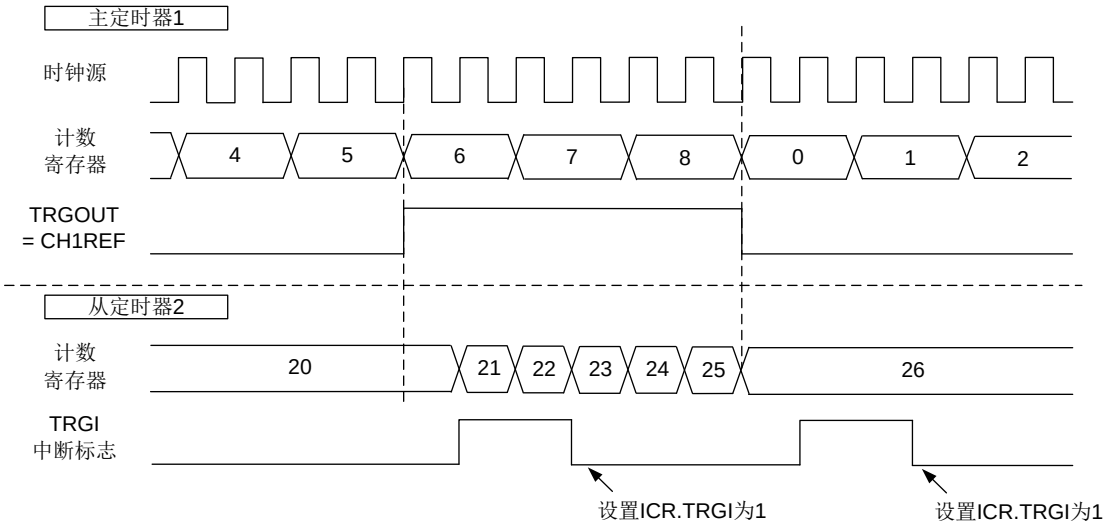


图 21-71　门控从定时器使用主定时器 CH1REF

在上图的例子中，在定时器 2 开启之前，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。可以在开启定时器 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。设置 **AD16C6T_SGE** 与定时器 2 的 SGE 寄存器的 SGUPD 位为 1 即可复位定时器。

20. 5. 31. 2　将一个定时器做为其他定时器的预分频器

在这个例子中，使用定时器 1(GP16C2T1)的更新事件作为定时器 2(AD16C6T1)的时钟来源，没有设定预分频。一旦定时器 1 产生更新事件，定时器 2 即根据其上升沿计数。

先设定从定时器(定时器 2)为外部时钟源 1 模式，配置如下:

1. 设置 **AD16C6T_AR** 寄存器的 ARV 位为 02h，当计数器递增到 2 后重载。
2. 设置 **AD16C6T_SMCON** 寄存器的 TSSEL1 位与 TSSEL2 位为 GP16C2T1，可参考内部触发连接表。
3. 设置 **AD16C6T_SMCON** 寄存器的 SMODS 位为 111b，选择外部时钟模式 1。
4. 设置 **AD16C6T_CON1** 寄存器的 CNTEN 位为 1，开启计数器。

再设定主定时器(定时器 1)配置如下:

1. 设置 **GP16C2T_AR** 寄存器的 ARV 位为 03h，当计数器递增到 3 后重载，并产生更新事件。
2. 设置 **GP16C2T_CON2** 寄存器的 MMSEL 位为 010b，选择更新事件(UPD)为触发输出(TRGOUT)。
3. 设置 **GP16C2T_CON1** 寄存器的 CNTEN 位为 1，开启计数器。

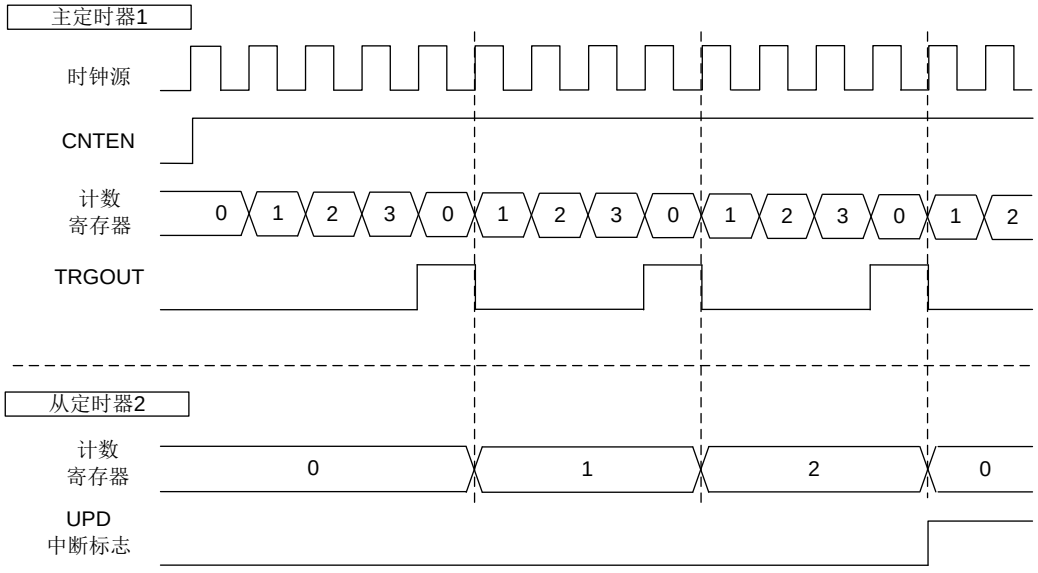


图 21-72　使用主定时器更新事件触发从定时器计数

20. 5. 31. 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(GP16C2T1)的 I1 输入上升沿时开启定时器 1，开启定时器 1 的同时开启定时器 2(AD16C6T1)。为保证计数器的对齐，定时器 1 必须设置为主/从模式(对应 I1 为从，对应定时器 2 为主)。

先设定从定时器(定时器 2)为触发模式，配置如下:

1. 设置 **AD16C6T_SMCON** 寄存器的 TSSEL1 位与 TSSEL2 位为 GP16C2T1，可参考内部触发连接表。
2. 设置 **AD16C6T_SMCON** 寄存器的 SMODS 位为 0110b，选择触发模式。

再设定主定时器(定时器 1)为触发模式，配置如下:

1. 设置定时器 1 为触发模式，相关配置可参考触发模式章节。
2. 设置 GP16C2T_CON2 寄存器的 MMSEL 位为 0001b，选择开启信号(CNTEN)为触发输出(TRGOUT)。
3. 设置 GP16C2T_SMCON 寄存器的 MSCFG 位为 1，开启主/从模式。
4. 设置 GP16C2T_CON1 寄存器的 CNTEN 位为 1，开启计数器。

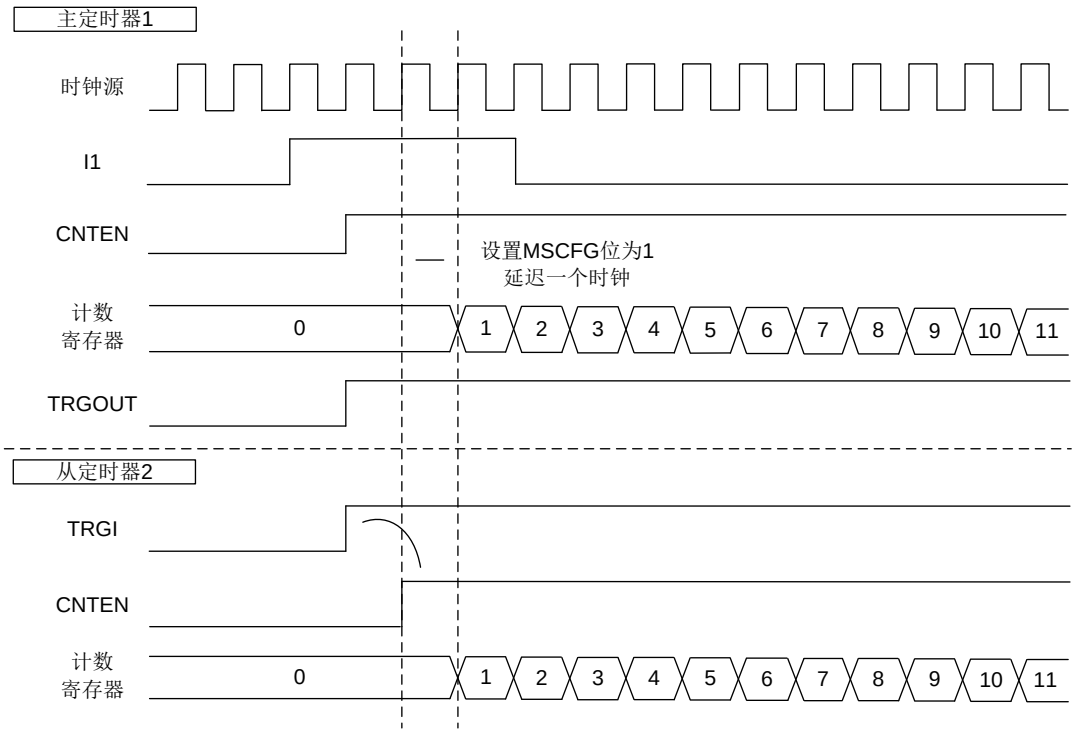


图 21-73　使用定时器 1 的 I1 输入触发定时器 1 和定时器 2

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。

20. 5. 32 ADC触发生成

定时器可生成 ADC 触发信号源如下:

- TRGOUT2:由 **AD16C6T_CON2** 寄存器的 MMSELTGO2 位决定触发事件的来源，根据配置生成脉冲或电平信号。
- TRGOUT:由 **AD16C6T_CON2** 寄存器的 MMSEL 位决定触发事件的来源，根据配置生成脉冲或电平信号。
- CHx:通道输出的电平信号。

下图以 CH1 输出 PWM 模式 2 为例，说明 TRGOUT2、TRGOUT 与 CH1 信号源，相关配置如下:

1. 设置 **AD16C6T_AR** 寄存器的 ARV 位为 07h，当计数器递增到 7 后重载。
2. 设置 **AD16C6T_CHMR1** 寄存器的 CH1MOD 位为 111b，选择 PWM 模式 2。
3. 设置 **AD16C6T_CCVAL1** 寄存器的 CCRV1 位为 04h，当计数器数到 4 时，PWM 输出高电平。
4. 设置 **AD16C6T_CON2** 寄存器的 MMSEL 位为 00100b，选择输出比较参考信号(CH1REF)为触发输出(TRGOUT)。
5. 设置 **AD16C6T_CON2** 寄存器的 MMSELTGO2 位为 00010b，选择更新事件(UDP)为触发输出(TRGOUT2)。

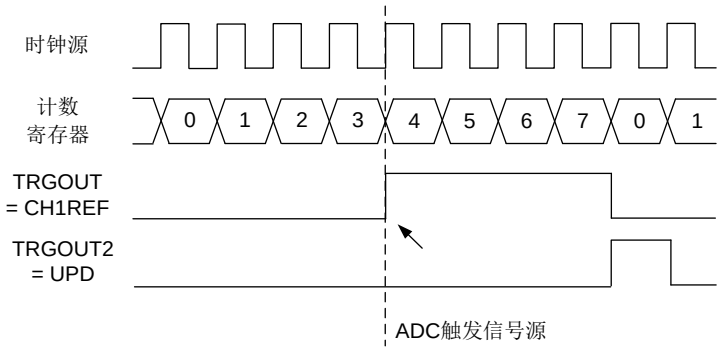


图 21-74　ADC 触发生成

当 MMSEL 设定成 1001b(MMSELTGO2 设定成 10000b)时，为多点触发输出模式，需要搭配 **AD16C6T_TGOSEL** 寄存器的设定，让相对应信号用于触发输出，如下图:

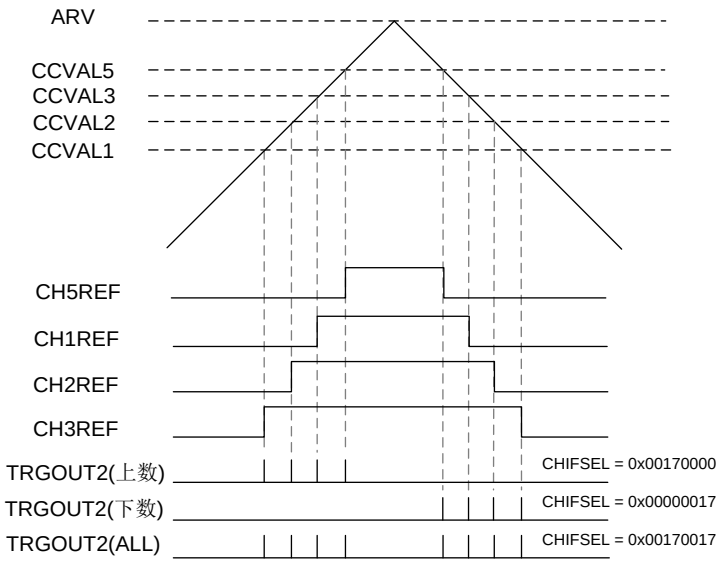


图 21-75　ADC 多点触发生成

20. 5. 33 影子寄存器

偏移地址 0x90 到 0xBC 为针对 DMA 应用所设计的影子寄存器，可通过此寄存器存取其相对应的原控制寄存器数据内容，目的是为了 DMA 搬动数据时，可以维持地址的连续性。

名称	偏移地址	类型	描述
AD16C6T_CON2	0090_H	R/W	控制寄存器 2
AD16C6T_CHMR1	0094_H	R/W	捕获或比较模式寄存器 1
AD16C6T_CHMR2	0098_H	R/W	捕获或比较模式寄存器 2
AD16C6T_CHMR3	$009C_H$	R/W	比较模式寄存器 3
AD16C6T_CCEP	$00A0_H$	R/W	捕获或比较开启极性寄存器

AD16C6T_CCVAL1	$00A4_H$	R/W	通道捕获或比较寄存器 1
AD16C6T_CCVAL2	$00A8_H$	R/W	通道捕获或比较寄存器 2
AD16C6T_CCVAL3	$00AC_H$	R/W	通道捕获或比较寄存器 3
AD16C6T_CCVAL4	$00B0_H$	R/W	通道捕获或比较寄存器 4
AD16C6T_CCVAL5	$00B4_H$	R/W	通道比较寄存器 5
AD16C6T_CCVAL6	$00B8_H$	R/W	通道比较寄存器 6
AD16C6T_TGOSEL	$00BC_H$	R/W	触发信号选择寄存器

20. 5. 34 调试模式

当微控制器进入调试模式(Cortex™-M0 核停止运行)，根据 SYSCFG 章节中 SYSCFG_CFG 寄存器配置，选择将计数器继续正常工作或停止计数。

20. 6 特殊功能寄存器

20. 6. 1 寄存器列表

AD16C6T 寄存器列表			
名称	偏移地址	类型	描述
AD16C6T_CON1	0000_H	R/W	控制寄存器 1
AD16C6T_CON2	0004_H 0090_H	R/W	控制寄存器 2
AD16C6T_SMCON	0008_H	R/W	从模式控制寄存器
AD16C6T_IER	$000C_H$	W1	中断开启寄存器
AD16C6T_IDR	0010_H	W1	中断关闭寄存器
AD16C6T_IVS	0014_H	R	中断功能有效状态寄存器
AD16C6T_RIF	0018_H	R	原始中断状态寄存器
AD16C6T_IFM	$001C_H$	R	中断标志位状态寄存器
AD16C6T_ICR	0020_H	C_W1	中断清除寄存器
AD16C6T_SGE	0024_H	T_W1	软件生成事件寄存器
AD16C6T_CHMR1	0028_H 0094_H	R/W	捕获或比较模式寄存器 1
AD16C6T_CHMR2	$002C_H$ 0098_H	R/W	捕获或比较模式寄存器 2
AD16C6T_CCEP	0030_H $00A0_H$	R/W	捕获或比较开启极性寄存器
AD16C6T_COUNT	0034_H	R/W	计数寄存器
AD16C6T_PRES	0038_H	R/W	预分频寄存器
AD16C6T_AR	$003C_H$	R/W	自动重载寄存器
AD16C6T_REPAR	0040_H	R/W	重复计数寄存器
AD16C6T_CCVAL1	0044_H $00A4_H$	R/W	通道捕获或比较寄存器 1
AD16C6T_CCVAL2	0048_H $00A8_H$	R/W	通道捕获或比较寄存器 2
AD16C6T_CCVAL3	$004C_H$ $00AC_H$	R/W	通道捕获或比较寄存器 3
AD16C6T_CCVAL4	0050_H $00B0_H$	R/W	通道捕获或比较寄存器 4
AD16C6T_BDCFG	0054_H	R/W	刹车和死区配置寄存器
AD16C6T_DMAEN	0058_H	R/W	DMA 事件开启寄存器
AD16C6T_CCVAL5	$005C_H$	R/W	通道比较寄存器 5

	$00B4_H$		
AD16C6T_CCVAL6	0060_H $00B8_H$	R/W	通道比较寄存器 6
AD16C6T_CHMR3	0064_H $009C_H$	R/W	比较模式寄存器 3
AD16C6T_DCFG2	0068_H	R/W	死区配置寄存器 2
AD16C6T_ENCR	$006C_H$	R/W	编码控制寄存器
AD16C6T_CHISEL	0070_H	R/W	通道输入来源选择寄存器
AD16C6T_AFR1	0074_H	R/W	复用功能寄存器 1
AD16C6T_AFR2	0078_H	R/W	复用功能寄存器 2
AD16C6T_TGOSEL	$007C_H$ $00BC_H$	R/W	触发信号选择寄存器
AD16C6T_CHRMP	0080_H	R/W	通道映射寄存器

20. 6. 2 寄存器描述

20. 6. 2. 1 控制寄存器 1(AD16C6T_CON1)

控制寄存器 1(AD16C6T_CON1)																												
偏移地址:0x00																												
复位值:0x0000 0000																												
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																			DITHEN	UPDREMAP		DFCKSEL<1:0>		ARPEN	CMSEL<1:0>		DIRSEL	SPMEN	UERSEL	DISUE	CNTEN

—	Bits 31-13	—	—
DITHEN	Bit 12	R/W	**抖动模式开启** 0: 抖动模式关闭 1: 抖动模式开启 注:此位仅在CNTEN关闭时修改
UPDREMAP	Bit 11	R/W	**UPD原始中断状态重新映射** UPD原始中断状态复制到AD16C6T_COUNT寄存器的31位 0: 重新映射关闭 1:重新映射开启
—	Bit 10	—	—
DFCKSEL	Bits 9-8	R/W	**时钟预分频器** 设置定时器频率(INT_CLK)，和死区产生器与数字滤波器所采样时钟之间的分频比例。 00:$t_{DTS}=t_{INT\_CLK}$ 01:$t_{DTS}=2 \times t_{INT\_CLK}$ 10:$t_{DTS}=4 \times t_{INT\_CLK}$ 11:保留
ARPEN	Bit 7	R/W	**自动重载预装载开启** 发生更新事件时，将预装载值载入到缓冲寄存器中 0:AD16C6T_AR 寄存器不具备缓冲 1:AD16C6T_AR 寄存器具备缓冲
CMSEL	Bits 6-5	R/W	**中心对齐模式选择** 00:边沿对齐模式，根据计数方向(DIRSEL)的配置，计数器为递增计数或递减计数 01:中心对齐模式 1，计数器为交替地递增计数

			和递减计数。在此模式时仅当计数器在递减计数时，才会产生配置为输出的通道(AD16C6T_CHMRn 寄存器中 CCnSSEL=00)的比较匹配中断请求 10:中心对齐模式 2，计数器为交替地递增计数和递减计数。在此模式时仅当计数器在递增计数时，才会产生配置为输出的通道(AD16C6T_CHMRn 寄存器中 CCnSSEL=00)的比较匹配中断请求 11:中心对齐模式 3，计数器为交替地递增计数和递减计数。在此模式时当计数器在递增计数或递减计数时，皆会产生配置为输出的通道(AD16C6T_CHMRn 寄存器中 CCnSSEL=00)的比较匹配中断请求 注:当 CNTEN 开启时，不允许边沿对齐模式与中心对齐模式互相切换
DIRSEL	Bit 4	R/W	**计数方向选择** 0:计数器递增计数 1:计数器递减计数 注:当计数器配置为中心对齐模式或编码器模式时，此位只能读取计数器的计数方向
SPMEN	Bit 3	R/W	**单脉冲模式** 0:单脉冲模式关闭，计数器不停止 1:单脉冲模式开启，计数器在发生下一次更新事件时，会自动清除 CNTEN 位，并停止计数器
UERSEL	Bit 2	R/W	**更新事件来源选择** 设置更新事件(UPD)的来源 0:下列事件都会产生更新中断或 DMA 的请求: – 计数器上溢或下溢 – 设置 AD16C6T_SGE 寄存器的 SGUPD 位为 1 – 通过从模式控制器所生成的更新事件 1:只有在计数器上溢或下溢时会生成更新中断或 DMA 的请求
DISUE	Bit 1	R/W	**更新事件禁止** 0:更新事件(UPD)开启时，下列事件皆会产生更新事件请求并将预装载值加载到缓冲寄存器中 – 计数器上溢或下溢

			– 设置 AD16C6T_SGE 寄存器的 SGUPD 位为 1 – 通过从模式控制器所生成的更新事件 1:更新事件(UPD)关闭时，不产生更新事件请求， AD16C6T_AR 、 AD16C6T_PRES 与 AD16C6T_CCVALn 寄存器的缓冲寄存器数值保持不变。但设置 AD16C6T_SGE 寄存器的 SGUPD 位为 1 或从模式控制器的复位请求，计数器和预分频器仍会被重新初始化
CNTEN	Bit 0	R/W	**计数器开启** 开启计数器后，外部时钟模式、门控模式和编码模式才能运作。而触发模式则可以由硬件将设置 CNTEN 位为 1 0:计数器关闭 1:计数器开启

20. 6. 2. 2 控制寄存器 2(AD16C6T_CON2)

此控制寄存器可被两个地址共同存取

控制寄存器 2(AD16C6T_CON2)																												
偏移地址: 0x04、0x90																												
复位值:0x0000 0000																												
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
—	—	—	—	—	—	MMSEL2	MMSELTGO2<4:0>					—	OISS6	—	OISS5	OISS4N	OISS4	OISS3N	OISS3	OISS2N	OISS2	OISS1N	OISS1	I1SEL	MMSEL<2:0>			CCDMASEL	CCUSEL	—	CCPCEN

—	Bits 31-26	—	—
MMSEL2	Bit 25	R/W	**主模式选择1** 参照MMSEL描述
MMSELTGO2	Bits 24-20	R/W	**主模式选择2** 选择在主模式下发送到ADC触发信号 2(TRGOUT2) 00000:复位- 设置AD16C6T_SGE寄存器中的 UPD位为触发输出。如果复位由触发输入生成 (从模式控制器配置为复位模式)，则TRGOUT2 上的信号与实际复位信号之间会有延迟 00001:开启-设置AD16C6T_CON1寄存器中的 CNTEN位为触发输出，可用于同步开启数个定

			时器。计数器开启信号可由AD16C6T_CON1寄存器中的CNTEN位或者门控模式下的触发输入产生。当计数器开启信号受控于触发输入时，TRGOUT2上的信号与实际触发信号之间会有延迟 00010:更新事件-更新事件被用于触发输出。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 00011:比较脉冲-每次发生捕获或比较匹配时，当产生CH1中断请求同时，触发输出会送出一个正脉冲 00100:比较信号-CH1REF信号用于触发输出 00101:比较信号-CH2REF信号用于触发输出 00110:比较信号-CH3REF信号用于触发输出 00111:比较信号-CH4REF信号用于触发输出 01000:比较信号-CH5REFC信号用于触发输出 01001:比较信号-CH6REFC信号用于触发输出 01010:比较脉冲-CH4REFC的上升沿或下降沿产生一个正脉冲 01011:比较脉冲-CH6REFC的上升沿或下降沿产生一个正脉冲 01100:比较脉冲-CH4REFC或CH6REFC的上升沿产生一个正脉冲 01101:比较脉冲-CH4REFC的上升沿或CH6REFC的下升沿产生一个正脉冲 01110:比较脉冲-CH5REFC或CH6REFC的上升沿产生一个正脉冲 01111:比较脉冲-CH5REFC的上升沿或CH6REFC的下升沿产生一个正脉冲 10000:多点触发-根据AD16C6T_TGOSEL寄存器的配置，让相对应脉冲用于触发输出
—	Bit 19	—	—
OISS6	Bit 18	R/W	**通道6输出的空闲状态选择位** 参照OISS1描述
—	Bit 17	—	—
OISS5	Bit 16	R/W	**通道5输出的空闲状态选择位** 参照OISS1描述
OISS4N	Bit 15	R/W	**通道4互补输出的空闲状态选择位**

			参照OISS1N描述
OISS4	Bit 14	R/W	**通道4输出的空闲状态选择位** 参照OISS1描述
OISS3N	Bit 13	R/W	**通道3互补输出的空闲状态选择位** 参照OISS1N描述
OISS3	Bit 12	R/W	**通道3输出的空闲状态选择位** 参照OISS1描述
OISS2N	Bit 11	R/W	**通道2互补输出的空闲状态选择位** 参照OISS1N描述
OISS2	Bit 10	R/W	**通道2输出的空闲状态选择位** 参照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:当AD16C6T_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时，如果CH1N已实现，则经过死区时间后，CH1=0 1:当GOEN=0时，如果CH1N已实现，则经过死区时间后，CH1=1 注:当AD16C6T_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
I1SEL	Bit 7	R/W	**I1 选择** 0:I1 输入连接到 AD16C6T_CH1 引脚 1:I1 输入连接到 AD16C6T_CH1、CH2 和 CH3 引脚异或组合(XOR)输出
MMSEL	Bits 6-4	R/W	**主模式选择 1** 此位与 MMSEL2 组合成 MMSEL = {MMSEL2,MMSEL[2:0]} 选择在主模式下发送到从定时器的同步信号与 ADC 触发信号(TRGOUT) 0000:复位 - 设置 AD16C6T_SGE 寄存器中的 UPD 位为触发输出。如果复位由触发输入生成(从模式控制器配置为复位模式)，则 TRGOUT 上的信号与实际复位信号之间会有延迟

			0001:开启 - 设置 AD16C6T_CON1 寄存器中的 CNTEN 位为触发输出，可用于同步开启数个定时器。计数器开启信号可由 AD16C6T_CON1 寄存器中的 CNTEN 位或者门控模式下的触发输入产生。当计数器开启信号受控于触发输入时，TRGOUT 上的信号与实际触发信号之间会有延迟 0010:更新事件 - 更新事件被用于触发输出。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 0011:比较脉冲 - 每次发生捕获或比较匹配时，当产生 CH1 中断请求同时，触发输出会送出一个正脉冲 0100:比较信号 - CH1REF 信号用于触发输出 0101:比较信号 - CH2REF 信号用于触发输出 0110:比较信号 - CH3REF 信号用于触发输出 0111:比较信号 - CH4REF 信号用于触发输出 1000:编码器时钟 - 编码器时钟用于触发输出 1001:多点触发 - 根据 AD16C6T_TGOSEL 寄存器的配置，让相对应脉冲用于触发输出其他:保留 注:编码器时钟输出只在 AD16C6T_SMCON 寄存器中的 SMODS 为编码器模式有效
CCDMASEL	Bit 3	R/W	**捕获或比较事件的 DMA 选择** 0:当发生 CHn 事件时，设置 CHn DMA 请求 1:当发生更新事件时，设置 CHn DMA 请求
CCUSEL	Bit 2	R/W	**捕获或比较更新事件来源选择** 0:捕获或比较预装载开启时(CCPCEN=1)，只能通过 AD16C6T_SGE 寄存器的 SGCOM 位为 1 时更新 1:捕获或比较预装载开启时(CCPCEN=1)，可通过 AD16C6T_SGE 寄存器的 SGCOM 位为 1 与 TRGI 的上升沿时被更新
—	Bit 1	—	—
CCPCEN	Bit 0	R/W	**捕获或比较预装载开启** AD16C6T_SGE 寄存器的 SGCOM 位设置为 1 或 TRGI 的上升沿时，发生换向事件(COM)并将预装载值载入到缓冲寄存器中

			0:AD16C6T_CCEP 寄存器中的 CCnEN、CCnNEN 和 AD16C6T_CHMRn 寄存器中的 CHnMOD 位不具备缓冲 1:AD16C6T_CCEP 寄存器中的 CCnEN、CCnNEN 和 AD16C6T_CHMRn 寄存器中的 CHnMOD 位具备缓冲

20. 6. 2. 3 从模式控制寄存器(AD16C6T_SMCON)

从模式控制寄存器(AD16C6T_SMCON)																													
偏移地址:0x08																													
复位值:0x0000 0000																													
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
—	—	—	—	—	—	SMODPS	SMODPE	—	—	TSSEL2 <1:0>		—	—	—	SMODS2	ETPOL	ECM2EN	ETPRES <1:0>		ETFLT<3:0>				MSCFG	TSSEL1<2:0>			OCLRS	SMODS<2:0>		

—	Bits 31-26	—	—
SMODPS	Bit 25	R/W	**SMODS更新事件来源选择** 0: 发生更新事件(UPD)时，将预装载值载入到缓冲寄存器中 1: 发生Index事件(IDX)时，将预装载值载入到缓冲寄存器中
SMODPE	Bit 24	R/W	**SMODS预装载开启** 0: SMODS位不具备缓冲 1: SMODS位具备缓冲
—	Bits 23-22	—	—
TSSEL2	Bits 21-20	R/W	**触发选择2** 参照TSSEL1描述
—	Bits 19-17	—	—
SMODS2	Bit 16	R/W	**从模式选择** 参照SMODS描述
ETPOL	Bit 15	R/W	**外部触发极性** 设置外部触发开启电平 0:ETR不反向，高电平或上升沿时开启 1:ETR反向，低电平或下降沿时开启
ECM2EN	Bit 14	R/W	**外部时钟模式2开启** 0:外部时钟模式2关闭 1:外部时钟模式2开启，计数器时钟根据ETP信号的任意有效边沿提供 注: 1:设置ECM2EN位与选择外部时钟模式1并将ETP(SMODS=111和TSSEL=00111)连到TRGI具有相同功效 2:从模式可以与外部时钟模式2同时使用:复位

			模式，门控模式和触发模式;但是这时TRGI不能连到ETP(TSSEL位不能是00111) 3:外部时钟模式1和外部时钟模式2同时被开启时，外部时钟的输入是ETP
ETPRES	Bits 13-12	R/W	**外部触发时钟预分频器** 外部触发输入信号ETR频率需要大于1/4 INT_CLK，分频器可以达到降低ETR的频率，有效应用于快速的外部时钟源 00: 预分频器禁止 01: ETR频率2分频 10: ETR频率4分频 11: ETR频率8分频
ETFLT	Bits 11-8	R/W	**外部触发滤波器** 设置ETP信号采样的频率和对ETP数字滤波器的带宽。数字滤波器是一个事件计数器，它记录到N个事件后才视为一个有效输出边沿: 0000: 采样频率f_{DTS}，滤波器禁止 0001: 采样频率f_{INT_CLK}, N = 2 0010: 采样频率f_{INT_CLK}, N = 4 0011: 采样频率f_{INT_CLK}, N = 8 0100: 采样频率f_{DTS} / 2, N = 6 0101: 采样频率f_{DTS} / 2, N = 8 0110: 采样频率f_{DTS} / 4, N = 6 0111: 采样频率f_{DTS} / 4, N = 8 1000: 采样频率f_{DTS} / 8, N = 6 1001: 采样频率f_{DTS} / 8, N = 8 1010: 采样频率f_{DTS} / 16, N = 5 1011: 采样频率f_{DTS} / 16, N = 6 1100: 采样频率f_{DTS} / 16, N = 8 1101: 采样频率f_{DTS} / 32, N = 5 1110: 采样频率f_{DTS} / 32, N = 6 1111: 采样频率f_{DTS} / 32, N = 8
MSCFG	Bit 7	R/W	**主/从模式** 延迟触发输入(TRGI)上的事件来允许当前主定时器和其从定时器之间的同步。该设置有效用于使用单个外部事件来同步多个定时器 0: 主从模式关闭 1: 主从模式开启

TSSEL1	Bits 6-4	R/W	**触发来源选择 1** 此位与 TSSEL2[1:0]组合成 TSSEL = {TSSEL2[1:0],TSSEL1[2:0]} 设置触发选择，用于同步计数器 00000: 内部触发 0 (IT0) 00001: 内部触发 1 (IT1) 00010: 内部触发 2 (IT2) 00011: 内部触发 3 (IT3) 00100: I1 边沿检测(I1F_ED) 00101: I1 滤波后信号(I1FP) 00110: I2 滤波后信号(I2FP) 00111: 外部触发输入(ETP) 01000: 内部触发 4 (IT4) 01001: 内部触发 5 (IT5) 01010: 内部触发 6 (IT6) 01011: 内部触发 7 (IT7) 其他: 保留 注:此位在需要在使用前设定(SMODS=000)，以避免产生错误的上升/下降边沿至计数器
OCLRS	Bit 3	R/W	**输出通道清除选择** 选择输出通道清除的来源 0: 选择来源为 OCLR 1: 选择来源为 ETP
SMODS	Bits 2-0	R/W	**从模式选择** 此位与 SMODS2 组合成 SMODS = {SMODS2, SMODS[2:0]} 0000: 从模式关闭 - 当 AD16C6T_CON1 寄存器 CNTEN 位为 1 时，预分频器时钟由内部时钟提供 0001: 正交编码器模式 1 - x2 模式，计数器根据 I2 电平在 I1 边沿递增或递减计数 0010: 正交编码器模式 2 - x2 模式，计数器根据 I1 电平在 I2 边沿递增或递减计数 0011: 正交编码器模式 3 - x4 模式，计数器根据 I2/I1 电平在 I1/I2 边沿递增或递减计数 0100: 复位模式 - 选中的触发输入(TRGI)的上升沿重新初始化计数器，并且产生一次更新事件

			0101: 门控模式 - 当触发输入(TRGI)为高电平时，计数器的时钟开启。一旦触发输入变为低电平，则计数器停止(但不复位)。计数器的启动和停止都是被控制 0110: 触发模式 - 计数器在触发输入 TRGI 的上升沿启动(但不复位)，只有计数器的启动是被控制 0111: 外部时钟模式 1 - 选中的触发输入(TRGI)的上升沿提供计数器时钟 1000: 组合复位模式 + 触发模式 - 选中的触发输入(TRGI)的上升沿重新初始化计数器，并且产生一次更新事件启动计数器 1001: 组合门控模式 + 复位模式 - 当触发输入(TRGI)为高电平时，计数器的时钟开启。一旦触发输入变为低电平，则计数器停止且复位。计数器的启动和停止都是被控制 1010: 时钟加方向编码器模式 1 - x2 模式，计数器根据 I1 电平控制计数方向，并在 I2 边沿计数 1011: 时钟加方向编码器模式 2 - x1 模式，计数器根据 I1 电平控制计数方向，并由 CC2POL 决定在 I2 的上升或下降边沿计数 1100: 定向时钟编码器模式 1: x2 模式，计数器根据 I2 指定电平在 I1 边沿递减计数，根据 I1 指定电平在 I2 边沿递增计数 1101: 定向时钟编码器模式 2: x1 模式，计数器根据 I2 指定电平在 I1 上升或下降边沿(根据 CC1POL)递减计数，根据 I1 指定电平在 I2 上升或下降边沿(根据 CC2POL)递增计数 1110: 正交编码器模式 4 - x1 模式，计数器根据 I2 指定电平在 I1 边沿递增或递减计数 1111: 正交编码器模式 5 - x1 模式，计数器根据 I1 指定电平在 I2 边沿递增或递减计数 注:如果 I1 双边沿检测被选为触发输入(设置 TSSEL 为 00100)时，不能使用门控模式。I1F 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

20. 6. 2. 4 中断开启寄存器(AD16C6T_IER)

中断开启寄存器(AD16C6T_IER)																													
偏移地址:0x0C																													
复位值:0x0000 0000																													
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
—	—	—	—	—	—	—	—	TERR	IERR	DIR	IDX	—	—	CH6	CH5	—	—	SBRK	CH4O	CH3OV	CH2OV	CH1OV	BRK2	BRK	TRGI	COM	CH4	CH3	CH2	CH1	UPD

—	Bits 31-24	—	—
TERR	Bit 23	W1	**开启转换错误中断功能** 此位设置时，开启中断功能，在编码器模式下，硬件侦测相位转换错误事件时发生中断
IERR	Bit 22	W1	**开启Index错误中断功能** 此位设置时，开启中断功能，在编码器模式下，硬件侦测Index错误事件时发生中断
DIR	Bit 21	W1	**开启方向转换中断功能** 此位设置时，开启中断功能，在编码器模式下，硬件侦测方向转换事件时发生中断
IDX	Bit 20	W1	**开启有效Index中断功能** 此位设置时，开启中断功能，在编码器模式下，硬件侦测有效Index事件时发生中断
—	Bits 19-18	—	—
CH6	Bit 17	W1	**开启通道6比较匹配中断功能** 此位设置时，开启中断功能，硬件侦测通道6比较匹配事件时发生中断
CH5	Bit 16	W1	**开启通道5比较匹配中断功能** 此位设置时，开启中断功能，硬件侦测通道5比较匹配事件时发生中断
—	Bits 15-14	—	—
SBRK	Bit 13	W1	**开启系统刹车中断功能** 此位设置时，开启中断功能，硬件侦测系统刹车事件时发生中断
CH4OV	Bit 12	W1	**开启通道4捕获溢出中断功能** 此位设置时，开启中断功能，硬件侦测通道4捕获溢出事件时发生中断
CH3OV	Bit 11	W1	**开启通道 3 捕获溢出中断功能** 此位设置时，开启中断功能，硬件侦测通道 3 捕获溢出事件时发生中断

CH2OV	Bit 10	W1	**开启通道 2 捕获溢出中断功能** 此位设置时，开启中断功能，硬件侦测通道 2 捕获溢出事件时发生中断
CH1OV	Bit 9	W1	**开启通道 1 捕获溢出中断功能** 此位设置时，开启中断功能，硬件侦测通道 1 捕获溢出事件时发生中断
BRK2	Bit 8	W1	**开启刹车 2 中断功能** 此位设置时，开启中断功能，硬件侦测刹车 2 信号时发生中断
BRK	Bit 7	W1	**开启刹车中断功能** 此位设置时，开启中断功能，硬件侦测刹车信号时发生中断
TRGI	Bit 6	W1	**开启触发中断功能** 此位设置时，开启中断功能，硬件侦测触发信号事件时发生中断
COM	Bit 5	W1	**开启 COM 中断功能** 此位设置时，开启中断功能，硬件侦测 COM 信号事件时发生中断
CH4	Bit 4	W1	**开启通道 4 捕获或比较匹配中断功能** 此位设置时，开启中断功能，硬件侦测通道 4 捕获或比较匹配事件时发生中断
CH3	Bit 3	W1	**开启通道 3 捕获或比较匹配中断功能** 此位设置时，开启中断功能，硬件侦测通道 3 捕获或比较匹配事件时发生中断
CH2	Bit 2	W1	**开启通道 2 捕获或比较匹配中断功能** 此位设置时，开启中断功能，硬件侦测通道 2 捕获或比较匹配事件时发生中断
CH1	Bit 1	W1	**开启通道 1 捕获或比较匹配中断功能** 此位设置时，开启中断功能，硬件侦测通道 1 捕获或比较匹配事件时发生中断
UPD	Bit 0	W1	**开启更新中断功能** 此位设置时，开启中断功能，硬件侦测更新事件时发生中断

20. 6. 2. 5 中断关闭寄存器(AD16C6T_IDR)

中断关闭寄存器(AD16C6T_IDR)																														
偏移地址:0x10																														
复位值:0x0000 0000																														
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								TERR	IERR	DIR	IDX			CH6	CH5			SBRK	CH4OV	CH3OV	CH2OV	CH1OV	BRK2	BRK	TRGI	COM	CH4	CH3	CH2	CH1	UPD

—	Bits 31-24	—	—
TERR	Bit 23	W1	**关闭转换错误中断功能** 此位设置时，关闭转换错误中断功能
IERR	Bit 22	W1	**关闭Index错误中断功能** 此位设置时，关闭Index错误中断功能
DIR	Bit 21	W1	**关闭方向转换中断功能** 此位设置时，关闭方向转换中断功能
IDX	Bit 20	W1	**关闭有效Index中断功能** 此位设置时，关闭有效Index中断功能
—	Bits 19-18	—	—
CH6	Bit 17	W1	**关闭通道6比较匹配中断功能** 此位设置时，关闭通道6比较匹配中断功能
CH5	Bit 16	W1	**关闭通道5比较匹配中断功能** 此位设置时，关闭通道5比较匹配中断功能
—	Bits 15-14	—	—
SBRK	Bit 13	W1	**关闭系统刹车中断功能** 此位设置时，关闭系统刹车中断功能
CH4OV	Bit 12	W1	**关闭通道4捕获溢出中断功能** 此位设置时，关闭通道4捕获溢出中断功能
CH3OV	Bit 11	W1	**关闭通道 3 捕获溢出中断功能** 此位设置时，关闭通道 3 捕获溢出中断功能
CH2OV	Bit 10	W1	**关闭通道 2 捕获溢出中断功能** 此位设置时，关闭通道 2 捕获溢出中断功能
CH1OV	Bit 9	W1	**关闭通道 1 捕获溢出中断功能** 此位设置时，关闭通道 1 捕获溢出中断功能
BRK2	Bit 8	W1	**关闭刹车 2 中断功能** 此位设置时，关闭刹车 2 中断功能
BRK	Bit 7	W1	**关闭刹车中断功能**

			此位设置时，关闭刹车中断功能
TRGI	Bit 6	W1	**关闭触发中断功能** 此位设置时，关闭触发中断功能
COM	Bit 5	W1	**关闭 COM 中断功能** 此位设置时，关闭 COM 中断功能
CH4	Bit 4	W1	**关闭通道 4 捕获或比较匹配中断功能** 此位设置时，关闭通道 4 捕获或比较匹配中断功能
CH3	Bit 3	W1	**关闭通道 3 捕获或比较匹配中断功能** 此位设置时，关闭通道 3 捕获或比较匹配中断功能
CH2	Bit 2	W1	**关闭通道 2 捕获或比较匹配中断功能** 此位设置时，关闭通道 2 捕获或比较匹配中断功能
CH1	Bit 1	W1	**关闭通道 1 捕获或比较匹配中断功能** 此位设置时，关闭通道 1 捕获或比较匹配中断功能
UPD	Bit 0	W1	**关闭更新中断功能** 此位设置时，关闭更新中断功能

20. 6. 2. 6 中断功能有效状态寄存器(AD16C6T_IVS)

中断功能有效状态寄存器(AD16C6T_IVS)																													
偏移地址:0x14																													
复位值:0x0000 0000																													
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								TERR	IERR	DIR	IDX			CH6	CH5			SBRK	CH4OV	CH3OV	CH2OV	CH1OV	BRK2	BRK	TRGI	COM	CH4	CH3	CH2	CH1	UPD

—	Bits 31-24	—	—
TERR	Bit 23	R	**转换错误中断功能状态** 0: 中断功能处于关闭状态 1: 中断功能处于开启状态
IERR	Bit 22	R	**Index错误中断功能状态** 0: 中断功能处于关闭状态 1: 中断功能处于开启状态
DIR	Bit 21	R	**方向转换中断功能状态** 0: 中断功能处于关闭状态 1: 中断功能处于开启状态
IDX	Bit 20	R	**有效Index中断功能状态** 0: 中断功能处于关闭状态 1: 中断功能处于开启状态
—	Bits 19-18	—	—
CH6	Bit 17	R	**通道6比较匹配中断功能状态** 0: 中断功能处于关闭状态 1: 中断功能处于开启状态
CH5	Bit 16	R	**通道5比较匹配中断功能状态** 0: 中断功能处于关闭状态 1: 中断功能处于开启状态
—	Bits 15-14	—	—
SBRK	Bit 13	R	**系统刹车中断功能状态** 0: 中断功能处于关闭状态 1: 中断功能处于开启状态
CH4OV	Bit 12	R	**通道 4 捕获溢出中断功能状态** 0: 中断功能处于关闭状态 1: 中断功能处于开启状态
CH3OV	Bit 11	R	**通道 3 捕获溢出中断功能状态** 0: 中断功能处于关闭状态

			1: 中断功能处于开启状态
CH2OV	Bit 10	R	**通道 2 捕获溢出中断功能状态** 0: 中断功能处于关闭状态 1: 中断功能处于开启状态
CH1OV	Bit 9	R	**通道 1 捕获溢出中断功能状态** 0: 中断功能处于关闭状态 1: 中断功能处于开启状态
BRK2	Bit 8	R	**刹车 2 中断功能状态** 0: 中断功能处于关闭状态 1: 中断功能处于开启状态
BRK	Bit 7	R	**刹车中断功能状态** 0: 中断功能处于关闭状态 1: 中断功能处于开启状态
TRGI	Bit 6	R	**触发中断功能状态** 0: 中断功能处于关闭状态 1: 中断功能处于开启状态
COM	Bit 5	R	**COM 中断功能状态** 0: 中断功能处于关闭状态 1: 中断功能处于开启状态
CH4	Bit 4	R	**通道 4 捕获或比较匹配中断功能状态** 0: 中断功能处于关闭状态 1: 中断功能处于开启状态
CH3	Bit 3	R	**通道 3 捕获或比较匹配中断功能状态** 0: 中断功能处于关闭状态 1: 中断功能处于开启状态
CH2	Bit 2	R	**通道 2 捕获或比较匹配中断功能状态** 0: 中断功能处于关闭状态 1: 中断功能处于开启状态
CH1	Bit 1	R	**通道 1 捕获或比较匹配中断功能状态** 0: 中断功能处于关闭状态 1: 中断功能处于开启状态
UPD	Bit 0	R	**更新中断功能状态** 0: 中断功能处于关闭状态 1: 中断功能处于开启状态

AD16C6T_IVS 寄存器，是实时反映系统配置 AD16C6T_IER 与 AD16C6T_IDR 的中断开启状态。此寄存器状态是将 AD16C6T_IER 与 AD16C6T_IDR 进行硬件运算，公式如下: AD16C6T_IVS = AD16C6T_IER &～AD16C6T_IDR

20. 6. 2. 7 原始中断状态寄存器(AD16C6T_RIF)

原始中断状态寄存器(AD16C6T_RIF)																													
偏移地址:0x18																													
复位值:0x0000 0000																													
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								TERR	IERR	DIR	IDX			CH6	CH5			SBRK	CH4OV	CH3OV	CH2OV	CH1OV	BRK2	BRK	TRGI	COM	CH4	CH3	CH2	CH1	UPD

—	Bits 31-24	—	—
TERR	Bit 23	R	**转换错误，原始中断状态** 0: 无发生中断 1: 已发生中断
IERR	Bit 22	R	**Index错误，原始中断状态** 0: 无发生中断 1: 已发生中断
DIR	Bit 21	R	**方向转换，原始中断状态** 0: 无发生中断 1: 已发生中断
IDX	Bit 20	R	**有效Index，原始中断状态** 0: 无发生中断 1: 已发生中断
—	Bits 19-18	—	—
CH6	Bit 17	R	**通道6比较匹配，原始中断状态** 0: 无发生中断 1: 已发生中断
CH5	Bit 16	R	**通道5比较匹配，原始中断状态** 0: 无发生中断 1: 已发生中断
—	Bits 15-14	—	—
SBRK	Bit 13	R	**系统刹车，原始中断状态** 0: 无发生中断 1: 已发生中断
CH4OV	Bit 12	R	**通道4捕获溢出，原始中断状态** 0: 无发生中断 1: 已发生中断
CH3OV	Bit 11	R	**通道 3 捕获溢出，原始中断状态**

			0: 无发生中断 1: 已发生中断
CH2OV	Bit 10	R	**通道 2 捕获溢出，原始中断状态** 0: 无发生中断 1: 已发生中断
CH1OV	Bit 9	R	**通道 1 捕获溢出，原始中断状态** 0: 无发生中断 1: 已发生中断
BRK2	Bit 8	R	**刹车 2，原始中断状态** 0: 无发生中断 1: 已发生中断
BRK	Bit 7	R	**刹车，原始中断状态** 0：无发生中断 1：已发生中断
TRGI	Bit 6	R	**触发，原始中断状态** 0: 无发生中断 1: 已发生中断
COM	Bit 5	R	**COM，原始中断状态** 0: 无发生中断 1: 已发生中断
CH4	Bit 4	R	**通道 4 捕获或比较匹配，原始中断状态** 0: 无发生中断 1: 已发生中断
CH3	Bit 3	R	**通道 3 捕获或比较匹配，原始中断状态** 0: 无发生中断 1: 已发生中断
CH2	Bit 2	R	**通道 2 捕获或比较匹配，原始中断状态** 0: 无发生中断 1: 已发生中断
CH1	Bit 1	R	**通道 1 捕获或比较匹配，原始中断状态** 0: 无发生中断 1: 已发生中断
UPD	Bit 0	R	**更新，原始中断状态** 0: 无发生中断 1: 已发生中断

20. 6. 2. 8 中断标志位状态寄存器(AD16C6T_IFM)

中断标志位状态寄存器(AD16C6T_IFM)																													
偏移地址:0x1C																													
复位值:0x0000 0000																													
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
—	—	—	—	—	—	—	—	TERR	IERR	DIR	IDX	—	—	CH6	CH5	—	—	SBRK	CH4OV	CH3OV	CH2OV	CH1OV	BRK2	BRK	TRGI	COM	CH4	CH3	CH2	CH1	UPD

—	Bits 31-24	—	—
TERR	Bit 23	R	**转换错误，中断标志位状态** 0: 无发生中断 1: 已发生中断
IERR	Bit 22	R	**Index错误，中断标志位状态** 0: 无发生中断 1: 已发生中断
DIR	Bit 21	R	**方向转换，中断标志位状态** 0: 无发生中断 1: 已发生中断
IDX	Bit 20	R	**有效Index，中断标志位状态** 0: 无发生中断 1: 已发生中断
—	Bits 19-18	—	—
CH6	Bit 17	R	**通道6比较匹配，中断标志位状态** 0: 无发生中断 1: 已发生中断
CH5	Bit 16	R	**通道5比较匹配，中断标志位状态** 0: 无发生中断 1: 已发生中断
—	Bits 15-14	—	—
SBRK	Bit 13	R	**系统刹车，中断标志位状态** 0: 无发生中断 1: 已发生中断
CH4OV	Bit 12	R	**通道4捕获溢出，中断标志位状态** 0: 无发生中断 1: 已发生中断
CH3OV	Bit 11	R	**通道 3 捕获溢出，中断标志位状态**

			0: 无发生中断 1: 已发生中断
CH2OV	Bit 10	R	**通道 2 捕获溢出，中断标志位状态** 0: 无发生中断 1: 已发生中断
CH1OV	Bit 9	R	**通道 1 捕获溢出，中断标志位状态** 0: 无发生中断 1: 已发生中断
BRK2	Bit 8	R	**刹车 2，中断标志位状态** 0: 无发生中断 1: 已发生中断
BRK	Bit 7	R	**刹车，中断标志位状态** 0: 无发生中断 1: 已发生中断
TRGI	Bit 6	R	**触发，中断标志位状态** 0: 无发生中断 1: 已发生中断
COM	Bit 5	R	**COM，中断标志位状态** 0: 无发生中断 1: 已发生中断
CH4	Bit 4	R	**通道 4 捕获或比较匹配，中断标志位状态** 0: 无发生中断 1: 已发生中断
CH3	Bit 3	R	**通道 3 捕获或比较匹配，中断标志位状态** 0: 无发生中断 1: 已发生中断
CH2	Bit 2	R	**通道 2 捕获或比较匹配，中断标志位状态** 0: 无发生中断 1: 已发生中断
CH1	Bit 1	R	**通道 1 捕获或比较匹配，中断标志位状态** 0: 无发生中断 1: 已发生中断
UPD	Bit 0	R	**更新，中断标志位状态** 0: 无发生中断 1: 已发生中断

AD16C6T_IFM 寄存器，是滤除已关闭中断功能的中断事件，只关注开启中断功能的事件。此寄存器状态是将 AD16C6T_RIF 与 AD16C6T_IVS 进行硬件运算，公式如下：AD16C6T_IFM = AD16C6T_RIF & AD16C6T_IVS

20. 6. 2. 9 中断清除寄存器(AD16C6T _ICR)

中断清除寄存器(AD16C6T _ICR)																													
偏移地址: 0x20																													
复位值:0x0000 0000																													
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								TERR	IERR	DIR	IDX			CH6	CH5			SBRK	CH4OV	CH3OV	CH2OV	CH1OV	BRK2	BRK	TRGI	COM	CH4	CH3	CH2	CH1	UPD

—	Bits 31-24	—	—
TERR	Bit 23	C_W1	**清除转换错误中断状态** 此位设置时，清除中断状态(AD16C6T_RIF与AD16C6T_IFM)
IERR	Bit 22	C_W1	**清除Index错误中断状态** 此位设置时，清除中断状态(AD16C6T_RIF与AD16C6T_IFM)
DIR	Bit 21	C_W1	**清除方向转换中断状态** 此位设置时，清除中断状态(AD16C6T_RIF与AD16C6T_IFM)
IDX	Bit 20	C_W1	**清除有效Index中断状态** 此位设置时，清除中断状态(AD16C6T_RIF与AD16C6T_IFM)
—	Bits 19-18	—	—
CH6	Bit 17	C_W1	**清除通道6比较匹配中断状态** 此位设置时，清除中断状态(AD16C6T_RIF与AD16C6T_IFM)
CH5	Bit 16	C_W1	**清除通道5比较匹配中断状态** 此位设置时，清除中断状态(AD16C6T_RIF与AD16C6T_IFM)
—	Bits 15-14	—	—
SBRK	Bit 13	C_W1	**清除系统刹车中断状态** 此位设置时，清除中断状态(AD16C6T_RIF与AD16C6T_IFM)
CH4OV	Bit 12	C_W1	**清除通道4捕获溢出中断状态** 此位设置时，清除中断状态(AD16C6T_RIF与AD16C6T_IFM)
CH3OV	Bit 11	C_W1	**清除通道 3 捕获溢出中断状态** 此位设置时，清除中断状态(AD16C6T_RIF 与AD16C6T_IFM)

CH2OV	Bit 10	C_W1	**清除通道 2 捕获溢出中断状态** 此位设置时，清除中断状态(AD16C6T_RIF 与 AD16C6T_IFM)
CH1OV	Bit 9	C_W1	**清除通道 1 捕获溢出中断状态** 此位设置时，清除中断状态(AD16C6T_RIF 与 AD16C6T_IFM)
BRK2	Bit 8	C_W1	**清除刹车 2 中断状态** 此位设置时，清除中断状态(AD16C6T_RIF 与 AD16C6T_IFM)
BRK	Bit 7	C_W1	**清除刹车中断状态** 此位设置时，清除中断状态(AD16C6T_RIF 与 AD16C6T_IFM)
TRGI	Bit 6	C_W1	**清除触发中断状态** 此位设置时，清除中断状态(AD16C6T_RIF 与 AD16C6T_IFM)
COM	Bit 5	C_W1	**清除 COM 中断状态** 此位设置时，清除中断状态(AD16C6T_RIF 与 AD16C6T_IFM)
CH4	Bit 4	C_W1	**清除通道 4 捕获或比较匹配中断状态** 此位设置时，清除中断状态(AD16C6T_RIF 与 AD16C6T_IFM)
CH3	Bit 3	C_W1	**清除通道 3 捕获或比较匹配中断状态** 此位设置时，清除中断状态(AD16C6T_RIF 与 AD16C6T_IFM)
CH2	Bit 2	C_W1	**清除通道 2 捕获或比较匹配中断状态** 此位设置时，清除中断状态(AD16C6T_RIF 与 AD16C6T_IFM)
CH1	Bit 1	C_W1	**清除通道 1 捕获或比较匹配中断状态** 此位设置时，清除中断状态(AD16C6T_RIF 与 AD16C6T_IFM)
UPD	Bit 0	C_W1	**清除更新中断状态** 此位设置时，清除中断状态(AD16C6T_RIF 与 AD16C6T_IFM)

AD16C6T_ICR 寄存器设置时，将清除 AD16C6T_RIF 与 AD16C6T_IFM 中断标志位状态；此设置不影响中断 AD16C6T_IER、AD16C6T_IDR 与 AD16C6T_IVS 寄存器，只清除标志位状态 AD16C6T_RIF 与 AD16C6T_IFM。此寄存器通过硬件清除中断，公式如下：AD16C6T_RIF = AD16C6T_RIF &～AD16C6T_ICR

20. 6. 2. 10 软件生成事件寄存器(AD16C6T_SGE)

软件生成事件寄存器(AD16C6T_SGE)																													
偏移地址:0x24																													
复位值:0x0000 0000																													
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																							SGBRK2	SGBRK	SGTRGI	SGCOM	SGCH4	SGCH3	SGCH2	SGCH1	SGUPD

—	Bits 31-9	—	—
SGBRK2	Bit 8	W1	**软件生成刹车事件2** 该位由软件设置来生成刹车事件2，可由硬件自动清零。 此位设置时，产生刹车事件2。GOEN清零，BRK2标志位置起，产生相关中断。
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。 此位设置时，产生刹车事件。GOEN清零，BRK标志位置起，产生相关中断。
SGTRGI	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。 此位设置时，产生触发事件。产生相关中断或DMA 传输
SGCOM	Bit 5	W1	**软件生成COM事件** 该位由软件设置来生成COM事件，可由硬件自动清零。 此位设置时，产生COM事件。当CCPCEN被置1，则可更新CCnEN, CCnNEN和CHnMOD。 注:该位只有用作于通道时有互补输出
SGCH4	Bit 4	W1	**软件生成通道 4 捕获/比较事件** 参照 SGCH1 描述
SGCH3	Bit 3	W1	**软件生成通道 3 捕获/比较事件** 参照 SGCH1 描述
SGCH2	Bit 2	W1	**软件生成通道 2 捕获/比较事件** 参照 SGCH1 描述

SGCH1	Bit 1	W1	**软件生成通道 1 捕获/比较事件** 该位由软件设置来生成通道 1 捕获或比较，可由硬件自动清零 **通道 CH1 设置为输出:** 此位设置时，产生比较事件，但不影响输出。若开启中断或 DMA，则产生中断与请求 **通道 CH1 设置为输入:** 此位设置时，产生捕获事件。将计数器捕获至 CCVAL1 寄存器中，于 I1 的有效沿产生，若开启中断或 DMA，则产生中断与请求
SGUPD	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 此位设置时，产生更新事件。重新初始化计数器，更新寄存器 注: 预分频器也会被清零(但预分频比不会受到影响)。如果使用中心对齐模式或者 DIRSEL=0(递增)，则计数器将清零；否则如果 DIRSEL=1(递减)，则计数器将加载自动重载值(AD16C6T_AR)

20. 6. 2. 11 捕获或比较模式寄存器 1(AD16C6T_CHMR1)

此控制寄存器可被两个地址共同存取

捕获或比较模式寄存器 1(AD16C6T_CHMR1)																													
偏移地址:0x28、0x94																													
复位值:0x0000 0000																													
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
						CH2MOD2<1:0>								CH1MOD2<1:0>		CH2OCLREN	CH2MOD<2:0>			CH2PEN	CH2FEN	CC2SSEL<1:0>		CH1OCLREN	CH1MOD<2:0>			CH1PEN	CH1FEN	CC1SSEL<1:0>	
																I2FLT<3:0>				I2PRES<1:0>				I1FLT<3:0>				I1PRES<1:0>			

输出比较模式

—	Bits 31-26	—	—
CH2MOD2	Bits 25-24	R/W	**输出比较通道2模式** 参照CH1MOD描述
—	Bits 23-18	—	—
CH1MOD2	Bits 17-16	R/W	**输出比较通道1模式** 参照CH1MOD描述
CH2OCLREN	Bit 15	R/W	**输出比较通道 2 清除开启** 参照 CH1OCLREN 描述
CH2MOD	Bits 14-12	R/W	**输出比较通道 2 模式** 参照 CH1MOD 描述
CH2PEN	Bit 11	R/W	**输出比较通道 2 预装载开启** 参照 CH1PEN 描述
CH2FEN	Bit 10	R/W	**输出比较通道 2 快速开启** 参照 CH1FEN 描述
CC2SSEL	Bits 9-8	R/W	**捕获或比较通道 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(只工作在 AD16C6T_SMCON 寄存器的 TSSEL 位选择为 ITn 时)

CH1OCLREN	Bit 7	R/W	**输出比较通道 1 清除开启** 0: CH1REF 维持输出 1:CH1REF 在 OCLR_INT(由 OCLRSEL 或 ETRSEL 选择)为高电平时清除
CH1MOD	Bits 6-4	R/W	**输出比较通道 1 模式** 此位与 CH1MOD2[1:0]组合成 CH1MOD[4:0] = {CH1MOD2[1:0], CH1MOD[2:0]} 这些位定义提供 CH1 和 CH1N 的输出参考信号 CH1REF 的行为。CH1REF 为高电平有效，而 CH1 和 CH1N 的有效电平则取决于 AD16C6T_CCEP 寄存器中的 CC1POL 位和 CC1NPOL 位 00000:关闭－无作用 00001:匹配时设置高电平－当计数器(AD16C6T_COUNT)匹配 AD16C6T_CCVAL1 寄存器时，CH1REF 设置为 1 00010:匹配时设置低电平－当计数器(AD16C6T_COUNT)匹配 AD16C6T_CCVAL1 寄存器时，CH1REF 设置为 0 00011:匹配时翻转电平－当计数器(AD16C6T_COUNT)匹配 AD16C6T_CCVAL1 寄存器时，CH1REF 翻转电平(当前为高电平则翻转成低电平，反之当前为低电平则翻转成高电平) 00100:强制低电平－CH1REF 强制设置低电平 00101:强制高电平－CH1REF 强制设置高电平 00110:PWM 模式 1－在递增计数时，当计数器(AD16C6T_COUNT)小于 AD16C6T_CCVAL1 寄存器时，输出高电平，其他则输出低电平。在递减计数时，当计数器(AD16C6T_COUNT)大于 AD16C6T_CCVAL1 寄存器时，输出低电平，其他则输出高电平 00111:PWM 模式 2－在递增计数时，当计数器(AD16C6T_COUNT)小于 AD16C6T_CCVAL1 寄存器时，输出低电平，其他则输出高电平。在递减计数时，当计数器(AD16C6T_COUNT)大于 AD16C6T_CCVAL1 寄存器时，输出高电平，其他则输出低电平

			01000: 多次触发单脉冲模式 1-在递增计数时，通道保持为高电平，直到在 TRGI 信号上侦测到触发事件后，通道正常输出 PWM 模式 1，直到下次更新事件发生后，通道再次保持高电平。在递减计数时，通道保持为低电平，直到在 TRGI 信号上侦测到触发事件后，通道正常输出 PWM 模式 1，直到下次更新事件发生后，通道再次保持低电平 01001: 多次触发单脉冲模式 2-在递增计数时，通道保持为低电平，直到在 TRGI 信号上侦测到触发事件后，通道正常输出 PWM 模式 2，直到下次更新事件发生后，通道再次保持低电平。在递减计数时，通道保持为高电平，直到在 TRGI 信号上侦测到触发事件后，通道正常输出 PWM 模式 2，直到下次更新事件发生后，通道再次保持高电平 01010: 保留 01011: 保留 01100: 组合 PWM 模式 1 - 在 PWM 模式 1 下产生 CH1REF，再将 CH1REF 和 CH2REF 做"OR"运算输出 CH1REFC 信号 01101: 组合 PWM 模式 2 - 在 PWM 模式 2 下产生 CH1REF，再将 CH1REF 和 CH2REF 做"AND"运算输出 CH1REFC 信号 01110: 非对称 C2 PWM 模式 1 - 在 PWM 模式 1 中，在递增计数时，CH1REFC 信号使用 CCRV1 输出比较。在递减计数时，CH1REFC 信号使用 CCRV2 01111: 非对称 C2 PWM 模式 2 - 在 PWM 模式 2 中，在递增计数时，CH1REFC 信号使用 CCRV1 输出比较。在递减计数时，CH1REFC 信号使用 CCRV2 10000: 非对称 C5 PWM 模式 1 - 在 PWM 模式 1 中，在递增计数时，CH1REFC 信号使用 CCRV1 输出比较。在递减计数时，CH1REFC 信号使用 CCRV5 10001: 非对称 C5 PWM 模式 2 - 在 PWM 模式 2 中，在递增计数时，CH1REFC 信号使用

			CCRV1 输出比较。在递减计数时，CH1REFC 信号使用 CCRV5 注:当设置 AD16C6T_BDCFG 寄存器中的 LOCKLVL 位为锁定级别 3 且 CC1SSEL = 00(通道设置为输出)后，此位无法更改。
CH1PEN	Bit 3	R/W	**输出比较通道 1 预装载开启** 设置后在更新事件时，将 AD16C6T_CCVAL1 寄存器预装载值载入到缓冲寄存器中。 0: CCVAL1 寄存器预装载关闭 1: CCVAL1 寄存器预装载开启 注:当设置 AD16C6T_BDCFG 寄存器中的 LOCKLVL 位为锁定级别 3 且 CC1SSEL = 00(通道设置为输出)后，此位无法更改。
CH1FEN	Bit 2	R/W	**输出比较通道 1 快速开启** 用于加速触发输入事件对于 PWM 输出的影响，CH1FEN 只在通道被配置成 PWM1 或 PWM2 模式时起作用。 0:CH1 的输出依赖于计数器与 CCVAL1 的值正常工作。 1:当触发输入(TRGI)有效时，CH1 被强制设置为比较电平(与比较结果无关)，触发输入(TRGI)的有效边沿相当于发生比较匹配。
CC1SSEL	Bit 1-0	R/W	**捕获或比较通道 1 选择** 设置通道的输出方向与信号的选择，当 AD16C6T_CCEP 寄存器的 CC1EN 位为 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(只工作在 AD16C6T_SMCON 寄存器的 TSSEL 位选择为 ITn 时)

输入捕获模式

—	Bits 31-16	—	—
I2FLT	Bits 15-12	R/W	**输入捕获通道2滤波器** 参照I1FLT描述
I2PRES	Bits 11-10	R/W	**输入捕获通道 2 预分频器**

			参照 IC1PRES 描述
CC2SSEL	Bits 9-8	R/W	**捕获或比较通道 2 选择** 设置通道的输出方向与信号的选择，当 AD16C6T_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(只工作在 AD16C6T_SMCON 寄存器的 TSSEL 位选择为 ITn 时)
I1FLT	Bits 7-4	R/W	**输入捕获通道 1 滤波器** 设置 I1 信号采样的频率和对 I1 数字滤波器的带宽。数字滤波器是一个事件计数器，它记录到 N 个事件后才视为一个有效输出边沿: 0000:采样频率 f_{DTS}，滤波器禁止 0001:采样频率 f_{INT_CLK}, N = 2 0010:采样频率 f_{INT_CLK}, N = 4 0011:采样频率 f_{INT_CLK}, N = 8 0100:采样频率 f_{DTS} / 2, N = 6 0101:采样频率 f_{DTS} / 2, N = 8 0110:采样频率 f_{DTS} / 4, N = 6 0111:采样频率 f_{DTS} / 4, N = 8 1000:采样频率 f_{DTS} / 8, N = 6 1001:采样频率 f_{DTS} / 8, N = 8 1010:采样频率 f_{DTS} / 16, N = 5 1011:采样频率 f_{DTS} / 16, N = 6 1100:采样频率 f_{DTS} / 16, N = 8 1101:采样频率 f_{DTS} / 32, N = 5 1110:采样频率 f_{DTS} / 32, N = 6 1111:采样频率 f_{DTS} / 32, N = 8
I1PRES	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 AD16C6T_ CCEP 寄存器的 CC1EN 位，预分频计数器同时被清除 00:预分频关闭，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	Bits 1-0	R/W	**捕获或比较通道 1 选择** 设置通道的输出方向与信号的选择，当 AD16C6T_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(只工作在 AD16C6T_SMCON 寄存器的 TSSEL 位选择为 ITn 时)

20. 6. 2. 12 捕获或比较模式寄存器 2(AD16C6T_CHMR2)

此控制寄存器可被两个地址共同存取

捕获或比较模式寄存器 2(AD16C6T_CHMR2)

偏移地址:0x2C、0x98

复位值:0x0000 0000

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
						CH4MOD2<1:0>								CH3MOD2<1:0>		CH4OCLREN	CH4MOD<2:0>			CH4PEN	CH4FEN	CC4SSEL<1:0>		CH3OCLREN	CH3MOD<2:0>			CH3PEN	CH3FEN	CC3SSEL<1:0>	
																I4FLT<3:0>				I4PRES<1:0>				I3FLT<3:0>				I3PRES<1:0>			

输出比较模式

—	Bits 31-26	—	—
CH4MOD2	Bits 25-24	R/W	**输出比较通道4模式** 参照CH3MOD描述
—	Bits 23-18	—	—
CH3MOD2	Bits 17-16	R/W	**输出比较通道3模式** 参照CH3MOD描述
CH4OCLREN	Bit 15	R/W	**输出比较通道 4 清除开启** 参照 CH3OCLREN 描述
CH4MOD	Bits 14-12	R/W	**输出比较通道 4 模式** 参照 CH3MOD 描述
CH4PEN	Bit 11	R/W	**输出比较通道 4 预装载开启**

			参照 CH3PEN 描述
CH4FEN	Bit 10	R/W	**输出比较通道 4 快速开启** 参照 CH3FEN 描述
CC4SSEL	Bits 9-8	R/W	**捕获或比较通道 4 选择** 设置通道的输出方向与信号的选择，当 AD16C6T_CCEP 寄存器的 CC4EN 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 I4 10:通道设置为输入，捕获源为 I3 11:通道设置为输入，捕获源为 ITn(只工作在 AD16C6T_SMCON 寄存器的 TSSEL 位选择为 ITn 时)
CH3OCLREN	Bit 7	R/W	**输出比较通道 3 清除开启** 参照 CH1OCLREN 描述
CH3MOD	Bits 6-4	R/W	**输出比较通道 3 模式** 参照 CH1MOD 描述。但多了以下两个模式： 01010: 比较脉冲模式: 输出产生一个比较脉冲，其脉冲宽度可由 AD16C6T_ENCR 的 PWPRES 位与 PW 位决定 01011: 方向输出模式. 将 AD16C6T_CON1 寄存器的 DIRSEL 位映射到通道输出。
CH3PEN	Bit 3	R/W	**输出比较通道 3 预装载开启** 参照 CH1PEN 描述
CH3FEN	Bit 2	R/W	**输出比较通道 3 快速开启** 参照 CH1FEN 描述
CC3SSEL	Bits 1-0	R/W	**捕获或比较通道 3 选择** 设置通道的输出方向与信号的选择，当 AD16C6T_CCEP 寄存器的 CC3EN 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 I3 10:通道设置为输入，捕获源为 I4 11:通道设置为输入，捕获源为 ITn(只工作在 AD16C6T_SMCON 寄存器的 TSSEL 位选择为 ITn 时)

输入捕获模式

—	Bits 31-16	—	—
I4FLT	Bits 15-12	R/W	**输入捕获通道4滤波器** 参照I3FLT描述
I4PRES	Bits 11-10	R/W	**输入捕获通道 4 预分频器** 参照 IC3PRES 描述
CC4SSEL	Bits 9-8	R/W	**捕获或比较通道 4 选择** 设置通道的输出方向与信号的选择，当 AD16C6T_CCEP 寄存器的 CC4EN 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 I4 10:通道设置为输入，捕获源为 I3 11:通道设置为输入，捕获源为 ITn(只工作在 AD16C6T_SMCON 寄存器的 TSSEL 位选择为 ITn 时)
I3FLT	Bits 7-4	R/W	**输入捕获通道 3 滤波器** 参照 I1FLT 描述
I3PRES	Bits 3-2	R/W	**输入捕获通道 3 预分频器** 参照 IC1PRES 描述
CC3SSEL	Bits 1-0	R/W	**捕获或比较通道 3 选择** 设置通道的输出方向与信号的选择，当 AD16C6T_CCEP 寄存器的 CC3EN 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 I3 10:通道设置为输入，捕获源为 I4 11:通道设置为输入，捕获源为 ITn(只工作在 AD16C6T_SMCON 寄存器的 TSSEL 位选择为 ITn 时)

20. 6. 2. 13 捕获或比较开启极性寄存器(AD16C6T_CCEP)

此控制寄存器可被两个地址共同存取

捕获或比较开启寄存器(AD16C6T_CCEP)																												
偏移地址:0x30、0xA0																												
复位值:0x0000 0000																												
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
										CC6POL	CC6EN			CC5POL	CC5EN	CC4NPOL	CC4NEN	CC4POL	CC4EN	CC3NPOL	CC3NEN	CC3POL	CC3EN	CC2NPOL	CC2NEN	CC2POL	CC2EN	CC1NPOL	CC1NEN	CC1POL	CC1EN

—	Bits 31-22	—	—
CC6POL	Bit 21	R/W	**比较通道6输出极性** 参照CC1POL描述
CC6EN	Bit 20	R/W	**比较通道6输出开启** 参照CC1EN描述
—	Bits 19-18	—	—
CC5POL	Bit 17	R/W	**比较通道5输出极性** 参照CC1POL描述
CC5EN	Bit 16	R/W	**比较通道5输出开启** 参照CC1EN描述
CC4NPOL	Bit 15	R/W	**捕获或比较通道4互补输出极性** 参照CC1NPOL描述
CC4NEN	Bit 14	R/W	**捕获或比较通道 4 互补输出开启** 参照 CC1NEN 描述
CC4POL	Bit 13	R/W	**捕获或比较通道 4 输出极性** 参照 CC1POL 描述
CC4EN	Bit 12	R/W	**捕获或比较通道 4 输出开启** 参照 CC1EN 描述
CC3NPOL	Bit 11	R/W	**捕获或比较通道 3 互补输出极性** 参照 CC1NPOL 描述
CC3NEN	Bit 10	R/W	**捕获或比较通道 3 互补输出开启** 参照 CC1NEN 描述
CC3POL	Bit 9	R/W	**捕获或比较通道 3 输出极性** 参照 CC1POL 描述
CC3EN	Bit 8	R/W	**捕获或比较通道 3 输出开启** 参照 CC1EN 描述
CC2NPOL	Bit 7	R/W	**捕获或比较通道 2 互补输出极性**

			参照 CC1NPOL 描述
CC2NEN	Bit 6	R/W	**捕获或比较通道 2 互补输出开启** 参照 CC1NEN 描述
CC2POL	Bit 5	R/W	**捕获或比较通道 2 输出极性** 参照 CC1POL 描述
CC2EN	Bit 4	R/W	**捕获或比较通道 2 输出开启** 参照 CC1EN 描述
CC1NPOL	Bit 3	R/W	**捕获或比较通道 1 互补输出极性** **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。参照 CC1POL 描述。 注 1: 对于有互补输出的通道，该位设置为预载值。如果 AD16C6T_CON2 寄存器中的 CCPCEN 位设置为 1，则只有当 COM 事件发生时，CC1NPOL 位才会设置为预载值中新的值。 注 2: 当 AD16C6T_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。
CC1NEN	Bit 2	R/W	**捕获或比较通道 1 互补输出开启** 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 AD16C6T_CON2 寄存器中的 CCPCEN 位设置为 1，则只有当 COM 事件发生时，CC1NEN 缓冲位才会设置为预载值中新的值
CC1POL	Bit 1	R/W	**捕获或比较通道 1 输出极性** **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 与 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 1:对于有互补输出的通道，该位设置为预载值。如果 AD16C6T_CON2 寄存器中的 CCPCEN 位设置为 1，则只有当 COM 事件发生时，CC1POL 缓冲位才会设置为预载值中新的值。 注 2:当 AD16C6T_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	**捕获或比较通道 1 输出开启** **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 CC1NEN 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 CC1NEN 决定 **通道 CH1 设置为输入:** 0:捕获关闭 1:捕获开启 注:对于有互补输出的通道，该位设置为预载值。如果 AD16C6T_CON2 寄存器中的 CCPCEN 位设置为 1，则只有当 COM 事件发生时，CC1EN 缓冲位才会设置为预载值中新的值。

20. 6. 2. 14 计数寄存器(AD16C6T_COUNT)

计数寄存器(AD16C6T_COUNT)																														
偏移地址:0x34																														
复位值:0x0000 0000																														
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
UPDRIF																CNTV<15:0>														

UPDRIF	Bit 31	R	**更新原始中断状态映射** 该位为映射AD16C6T_RIF寄存器的UPD位的状态，该位只能读取，表示计数器发生更新事件。 设置AD16C6T_CON1寄存器的UPDFREMAP位为1开启功能。
—	Bits 30-16	—	—
CNTV	Bits 15-0	R/W	**计数器数值**

20. 6. 2. 15 预分频寄存器(AD16C6T_PRES)

预分频寄存器(AD16C6T_PRES)																														
偏移地址:0x38																														
复位值:0x0000 0000																														
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																PSCV<15:0>														

—	Bits 31-16	—	—
PSCV	Bits 15-0	R/W	**预分频数值** 当计数器时钟频率等于f_{INT_CLK}/(PSCV<15:0> + 1)时计数器递增或递减。在更新事件产生时，将PSCV数值被载入预装载寄存器中

20. 6. 2. 16 自动重载寄存器(AD16C6T_AR)

自动重载寄存器(AD16C6T_AR)																														
偏移地址:0x3C																														
复位值:0x0000 FFFF																														
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
												ARV<19:0>																			

—	Bits 31-20	—	—
ARV	Bits 19-0	R/W	**自动重载数值** 设置计数器的递增计数的边界或递减计数的重载值 抖动模式开启时: – ARV[19:4]为整数位 – ARV[3:0]为小数位

20. 6. 2. 17 重复计数寄存器(AD16C6T_REPAR)

重复计数寄存器(AD16C6T_REPAR)																														
偏移地址:0x40																														
复位值:0x00000000																														
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
																REPV<15:0>															

—	Bits 31-16	—	—
REPV	Bits 15-0	R/W	**重复计数数值** 当预载寄存器开启，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。当 REPV_CNT 计数器递减至 0 时，会产生更新事件，同时重新从 REPV 值递减计数。REPV_CNT 具备缓冲，因此只有当发生更新事件时,才会将 REPV 重载到 REPV_CNT 中，即任何对于 REPV 的写入，只会在下一次更新事件后才会生效。

			- 在边沿对齐模式下的 PWM 输出，(REPV+1)对应的是 PWM 的周期数 - 在中心对齐模式下的 PWM 输出，(REPV+1)对应的是 1/2 PWM 的周期数

20. 6. 2. 18 通道捕获或比较寄存器 1(AD16C6T_CCVAL1)

此控制寄存器可被两个地址共同存取

通道捕获或比较寄存器 1(AD16C6T_CCVAL1)																												
偏移地址:0x44、0xA4																												
复位值:0x0000 0000																												
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
												CCRV1<19:0>																			

—	Bits 31-20	—	—
CCRV1	Bits 19-0	R/W	**捕获或比较数值 1** **通道 CH1 配置为输出:** CCRV1 是捕获或比较寄存器的预装载值。 开启 AD16C6T_CHMR1 寄存器中的 CH1PEN 预装载功能时，CCRV1 具备缓冲，当发生更新事件后，将预装载值载入到缓冲寄存器中。 CCRV1 的值会与 AD16C6T_COUNT 中的值进行比较，其结果会反应在 CH1REF。 抖动模式关闭时: – CCRV1[19:16]为无效值 – CCRV1[15:0]为有效值 抖动模式开启时: – CCRV1[19:4]为整数位 – CCRV1[3:0]为小数位 **通道 CH1 配置为输入:** CCRV1 为由先前发生输入捕获事件(I1)时的计数器数值。 抖动模式关闭时: – CCRV1[19:16]为无效位 – CCRV1[15:0]为有效位 抖动模式开启时:

			- CCRV1[19:4]为有效位 - CCRV1[3:0]为无效位

20. 6. 2. 19 通道捕获或比较寄存器 2(AD16C6T_CCVAL2)

此控制寄存器可被两个地址共同存取

通道捕获或比较寄存器 2(AD16C6T_CCVAL2)																												
偏移地址:0x48、0xA8																												
复位值:0x0000 0000																												
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
												CCRV2<19:0>																			

—	Bits 31-20	—	—
CCRV2	Bits 19-0	R/W	**捕获或比较数值2** 参照CCRV1描述

20. 6. 2. 20 通道捕获或比较寄存器 3(AD16C6T_CCVAL3)

此控制寄存器可被两个地址共同存取

通道捕获或比较寄存器 3(AD16C6T_CCVAL3)																												
偏移地址:0x4C、0xAC																												
复位值:0x0000 0000																												
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
												CCRV3<19:0>																			

—	Bits 31-20	—	—
CCRV3	Bits 19-0	R/W	**捕获或比较数值3** 参照CCRV1描述

20. 6. 2. 21 通道捕获或比较寄存器 4(AD16C6T_CCVAL4)

此控制寄存器可被两个地址共同存取

通道捕获或比较寄存器 4(AD16C6T_CCVAL4)																													
偏移地址:0x50、0xB0																													
复位值:0x0000 0000																													
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
												CCRV4<19:0>																			

—	Bits 31-20	—	—
CCRV4	Bits 19-0	R/W	**捕获或比较数值4** 参照CCRV1描述

20. 6. 2. 22 刹车和死区配置寄存器 (AD16C6T_BDCFG)

刹车和死区配置寄存器 (AD16C6T_BDCFG)																													
偏移地址:0x54																													
复位值:0x0000 0000																													
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
		BRK2BID	BRKBID	BRK2DSRM	BRKDSRM	BRK2P	BRK2EN	BRK2FLT<3:0>				BRKFLT<3:0>				GOEN	AOEN	BRKP	BRKEN	OFFSSR	OFFSSI	LOCKLVL<1:0>		DT<7:0>							

—	Bits 31-30	—	—
BRK2BID	Bit 29	R/W	**BKIN2双向开启** 参照BRKBID描述
BRKBID	Bit 28	R/W	**BKIN双向开启** 当配置成双向模式时，BKIN引脚必须设定为开漏输出模式，任何有效的刹车事件都会使BKIN引脚改为输出低电平，以向外部设备指示发生刹车事件。 0: BKIN引脚为输入模式 1: BKIN引脚为双向模式 注:当AD16C6T_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
BRK2DSRM	Bit 27	R/W	**解除BKIN2刹车事件**

			参照BRKDSRM描述
BRKDSRM	Bit 26	R/W	**解除BKIN刹车事件** 当解除刹车事件后，此位由硬件自动清除。 设置BRKDSRM位1，以解除刹车事件并释放双向控制，软件需要持续读取此位，直到由硬件清除，指示刹车事件已消失 0: BKIN引脚输入为警备状态，可立即响应刹车事件 1: BKIN引脚输入为非警备状态，无法响应刹车事件
BRK2P	Bit 25	R/W	**选择BKIN2刹车极性** 参照BRKP描述
BRK2EN	Bit 24	R/W	**开启刹车2** 参照BRKEN描述 注:刹车2功能必须在OFFSSR和OFFSSI皆为1时使用。
BRK2FLT	Bits 23-20	R/W	**刹车2滤波器** 参照BRKFLT描述
BRKFLT	Bits 19-16	R/W	**刹车滤波器** 设置BKIN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个事件后才视为一个有效输出边沿: 0000: 采样频率f_{DTS}，滤波器禁止 0001: 采样频率f_{INT_CLK}, N = 2 0010: 采样频率f_{INT_CLK}, N = 4 0011: 采样频率f_{INT_CLK}, N = 8 0100: 采样频率f_{DTS} / 2, N = 6 0101: 采样频率f_{DTS} / 2, N = 8 0110: 采样频率f_{DTS} / 4, N = 6 0111: 采样频率f_{DTS} / 4, N = 8 1000: 采样频率f_{DTS} / 8, N = 6 1001: 采样频率f_{DTS} / 8, N = 8 1010: 采样频率f_{DTS} / 16, N = 5 1011: 采样频率f_{DTS} / 16, N = 6 1100: 采样频率f_{DTS} / 16, N = 8 1101: 采样频率f_{DTS} / 32, N = 5 1110: 采样频率f_{DTS} / 32, N = 6 1111: 采样频率f_{DTS} / 32, N = 8

			注:当AD16C6T_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
GOEN	Bit 15	R/W	**通道主要输出开启** 一旦BKIN或BKIN2刹车输入有效，该位会由硬件异步清零。该位可由软件设置为1或自动设置为1(取决于AOEN位)。该位仅作用于配置为输出的通道。 0:若此位因BKIN2刹车事件清0，则禁止CHn/CHnN输出(此时引脚不由定时器控制) 若此位因BKIN刹车事件或软件设置清0，则根据OFFSSI的设定，禁止CHn/CHnN输出或输出空闲电平(根据AD16C6T_CON2寄存器的OISSn/OISSnN设定) 1:如果CHn和CHnN各自的开启位都设置为1(AD16C6T_CCEP寄存器中的CCnEN和CCnNEN位)，则开启CHn和CHnN输出。
AOEN	Bit 14	R/W	**通道自动输出开启** 在发生更新事件时，将GOEN位置起 0: GOEN仅可由软件设置为1 1: GOEN位可由软件设置为1，也可以在下一个更新事件发生且刹车输入无效时自动设置为1。 注:当AD16C6T_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BKIN刹车极性** 0:刹车输入BKIN为低电平有效 1:刹车输入BKIN为高电平有效 注:当AD16C6T_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。
BRKEN	Bit 12	R/W	**开启刹车** 0:刹车输入关闭 1:刹车输入开启 注:当AD16C6T_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

OFFSSR	Bit 11	R/W	**运行模式下关闭状态选择** 此位在GOEN位为1且设置为输出模式并具有互补输出的通道。 0:当通道关闭时(CCnEN/CCnNEN位为0)，禁止CHn/CHnN引脚输出(此时不由定时器控制) 1:当通道关闭时(CCnEN/CCnNEN位为0)，一旦其互补通道为开启状态，会将输出设为无效电平(此时输出还是由定时器控制)。 注:当AD16C6T_BDCFG寄存器中的LOCKLVL位已被设置为锁定级别2，则该位无法修改。
OFFSSI	Bit 10	R/W	**空闲模式下关闭状态选择** 此位只适用在GOEN位因刹车事件或软件写入为0，且设置为输出模式并具有互补输出的通道。 0:当通道关闭时(CCnEN/CCnNEN位为0)，禁止CHn/CHnN引脚输出(此时不由定时器控制) 1:当通道关闭时(CCnEN/CCnNEN位为0)，会先输出其无效电平，并在死区时间后，改为输出空闲电平(根据AD16C6T_CON2寄存器的OISSn/OISSnN设定)。 注:当AD16C6T_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	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 - AD16C6T_CON2寄存器中的OISSn和OISSnN、AD16C6T_BDCFG寄存器中的BRKBID/BRK2BID/BRKEN/BRK2EN/BRKP/BR2KP/AOEN/DT、AD16C6T_DCFG2寄存器、AD16C6T_AFR1寄存器和AD16C6T_AFR2寄存器不再可写 10:锁定级别2 - 锁定级别1 + AD16C6T_BDCFG寄存器中的OFFSSR和OFFSSI、AD16C6T_CCEP寄存器中的CCnPOL和CCnNPOL(通道配置成输出模式

			时)不再可写 11:锁定级别3 - 锁定级别2 + AD16C6T_CHMRn寄存器中的CHnMOD和CHnPEN(通道配置成输出模式时)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	Bits 7-0	R/W	**死区延迟** 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])x t_{dtg}，式中 $t_{dtg}=2xt_{DTS}$。 DT[7:5]=110 ⇒ DT=(32+DT[4:0])x t_{dtg}，式中 $t_{dtg}=8xt_{DTS}$。 DT[7:5]=111 ⇒ DT=(32+DT[4:0])x t_{dtg}，式中 $t_{dtg}=16xt_{DTS}$。 注:当AD16C6T_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

20. 6. 2. 23 DMA事件开启寄存器 (AD16C6T_DMAEN)

DMA 事件开启寄存器 (AD16C6T_DMAEN)

偏移地址:0x58

复位值:0x0000 0000

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
																									TRGI	COM	CH4	CH3	CH2	CH1	UPD

—	Bits 31-7	—	—
TRGI	Bit 6	R/W	**触发事件的DMA请求开启** 0: 触发事件的DMA请求关闭 1: 触发事件的DMA请求开启
COM	Bit 5	R/W	**COM事件的DMA请求开启** 0: COM事件的DMA请求关闭

			1: COM事件的DMA请求开启
CH4	Bit 4	R/W	**通道4捕获或比较匹配事件的DMA请求开启** 0: 通道4捕获或比较匹配事件的DMA请求关闭 1: 通道4捕获或比较匹配事件的DMA请求开启
CH3	Bit 3	R/W	**通道3捕获或比较匹配事件的DMA请求开启** 0: 通道3捕获或比较匹配事件的DMA请求关闭 1: 通道3捕获或比较匹配事件的DMA请求开启
CH2	Bit 2	R/W	**通道2捕获或比较匹配事件的DMA请求开启** 0: 通道2捕获或比较匹配事件的DMA请求关闭 1: 通道2捕获或比较匹配事件的DMA请求开启
CH1	Bit 1	R/W	**通道1捕获或比较匹配事件的DMA请求开启** 0: 通道1捕获或比较匹配事件的DMA请求关闭 1: 通道1捕获或比较匹配事件的DMA请求开启
UPD	Bit 0	R/W	**更新事件的DMA请求开启** 0: 更新事件的DMA请求关闭 1: 更新事件的DMA请求开启

20. 6. 2. 24 通道比較寄存器 5 (AD16C6T_CCVAL5)

此控制寄存器可被两个地址共同存取

通道比較寄存器 5 (AD16C6T_CCVAL5)																													
偏移地址:0x5C、0xB4																													
复位值:0x0000 0000																													
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
GC5C3	GC5C2	GC5C1										CCRV5<19:0>																			

GC5C3	Bit 31	R/W	**组合通道5和通道3** 设置CH3REFC输出信号 0: CH5REFC对CH3REFC无影响 1: CH3REFC是CH3REFC和CH5REFC做"AND"运算输出 注:此位也支持预装载功能，由AD16C6T_CHMR3寄存器的CH5PEN位设定
GC5C2	Bit 30	R/W	**组合通道5和通道2** 设置CH2REFC输出信号 0: CH5REFC对CH2REFC无影响 1: CH2REFC是CH2REFC和CH5REFC做"AND"运算输出 注: 此位也支持预装载功能，由AD16C6T_CHMR3寄存器的CH5PEN位设定
GC5C1	Bit 29	R/W	**组合通道5和通道1** 设置CH1REFC输出信号 0: CH5REFC对CH1REFC无影响 1: CH1REFC是CH1REFC和CH5REFC做"AND"运算输出 注: 此位也支持预装载功能，由AD16C6T_CHMR3寄存器的CH5PEN位设定
—	Bits 28-20	—	—
CCRV5	Bits 19-0	R/W	**捕获或比较数值5** 参照CCRV1描述

20. 6. 2. 25 通道比较寄存器 6 (AD16C6T_CCVAL6)

此控制寄存器可被两个地址共同存取

通道比较寄存器 6 (AD16C6T_CCVAL6)																												
偏移地址:0x60、0xB8																												
复位值:0x0000 0000																												
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
—	—	—	—	—	—	—	—	—	—	—	—	CCRV6<19:0>																			

—	Bits 31-20	—	—
CCRV6	Bits 19-0	R/W	**捕获或比较数值6** 参照CCRV1描述

20. 6. 2. 26 比较模式寄存器 3 (AD16C6T_CHMR3)

此控制寄存器可被两个地址共同存取

比较模式寄存器 3 (AD16C6T_CHMR3)																												
偏移地址:0x64、0x9C																												
复位值:0x0000 0000																												
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
—	—	—	—	—	—	—	CH6MOD2	—	—	—	—	—	—	—	CH5MOD2	CH6OCLREN	CH6MOD<2:0>			CH6PEN	CH6FEN	—	—	CH5OCLREN	CH5MOD<2:0>			CH5PEN	CH5FEN	—	—

—	Bits 31-26	—	—
CH6MOD2	Bit 24	R/W	**输出比较通道6模式** 参照CH1MOD[3:0]描述
—	Bits 23-17	—	—
CH5MOD2	Bit 16	R/W	**输出比较通道5模式** 参照CH1MOD[3:0]描述
CH6OCLREN	Bit 15	R/W	**输出比较通道6清除开启** 参照CH1OCLREN描述
CH6MOD	Bits 14-12	R/W	**输出比较通道6模式** 参照CH1MOD[3:0]描述
CH6PEN	Bit 11	R/W	**输出比较通道6预装载开启** 参照CH1PEN描述
CH6FEN	Bit 10	R/W	**输出比较通道6快速开启**

			参照CH1FEN描述
—	Bits 9-8	—	—
CH5OCLREN	Bit 7	R/W	**输出比较通道5清除开启** 参照CH1OCLREN描述
CH5MOD	Bits 6-4	R/W	**输出比较通道5模式** 参照CH1MOD[3:0]描述
CH5PEN	Bit 3	R/W	**输出比较通道5预装载开启** 参照CH1PEN描述
CH5FEN	Bit 2	R/W	**输出比较通道5快速开启** 参照CH1FEN描述
—	Bits 1-0	—	—

20. 6. 2. 27 死区配置寄存器 2 (AD16C6T_DCFG2)

死区配置寄存器 2 (AD16C6T_DCFG2)

偏移地址:0x68

复位值:0x0000 0000

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
														DTPEN	DTAEN									DT2<7:0>							

—	Bits 31-18	—	—
DTPEN	Bit 17	R/W	**死区延迟预装载开启** 0: 死区延迟不具备缓冲 1: 死区延迟具备缓冲 注:当AD16C6T_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 注:此位仅在CNTEN关闭时修改
DTAEN	Bit 16	R/W	**非对称死区延迟开启** 0: 上升沿与下降沿的死区延迟一致，参照DT配置 1: 上升沿的死区延迟参照DT配置，而下降沿的死区延迟参照DT2配置 注:当AD16C6T_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

			注:此位仅在CNTEN关闭时修改，若在CNTEN开启时配置，此位会自动清0
—	Bits 15-8	—	—
DT2	Bits 7-0	R/W	**非对称死区延迟** 参照DT描述 注:此位仅在CNTEN关闭时修改

20. 6. 2. 28 编码控制寄存器 (AD16C6T_ENCR)

编码控制寄存器 (AD16C6T_ENCR)

偏移地址:0x6C

复位值:0x0000 0000

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
					PWPRES<2:0>			PW<7:0>																	IDXPOS<1:0>		FIDX			IDXDIR<1:0>>		IDXEN

—	Bits 31-27	—	—
PWPRES	Bits 26-24	R/W	**脉冲宽度预分频值** 公式如下: $t_{PWG} = (2^{(PWPRES[2:0])}) \times t_{INT\_CLK}$
PW	Bits 23-16	R/W	**脉冲宽度** 公式如下: $t_{PW} = PW[7:0] \times t_{PWG}$
—	Bits 15-8	—	—
IDXPOS	Bits 7-6	R/W	**Index位置** 在正交编码器模式下(AD16C6T_SMCON寄存器的SMODS = 0001, 0010, 0011, 1110, 1111), 此位用来指示Index信号在哪个AB相位为有效输入，判定为有效输入时重置计数器 00: 当AB相位 = 00时，侦测到有效的Index输入会重置计数器 01: 当AB相位 = 01时，侦测到有效的Index输入会重置计数器 10: 当AB相位 = 10时，侦测到有效的Index输入会重置计数器 11: 当AB相位 = 11时，侦测到有效的Index输入会重置计数器

			在时钟加方向模式与定向时钟模式下(AD16C6T_SMCON寄存器的SMODS = 1010, 1011, 1100, 1101)，IDXPOS[1]为无效位 x0: 当时钟低电平时，侦测到有效的Index输入会重置计数器 x1: 当时钟高电平时，侦测到有效的Index输入会重置计数器
FIDX	Bit 5	R/W	**首次Index事件检测** 0: 每一个Index事件都会重置计数器 1: 只让第一次的Index事件重置计数器
—	Bits 4-3	—	—
IDXDIR	Bits 2-1	R/W	**特定计数方向下侦测Index事件** 00: Index事件在递增、递减计数下都为有效 01: Index事件只在递增计数下有效 10: Index事件只在递减计数下有效 11: 保留 注: 当Index功能关闭时才能变更IDXDIR设定
IDXEN	Bit 0	R/W	**Index检测功能开启** 侦测到Index有效事件时会重置计数器 0: Index功能关闭 1: Index功能开启

20. 6. 2. 29 通道输入来源选择寄存器 (AD16C6T_CHISEL)

通道输入来源选择寄存器 (AD16C6T_CHISEL)

偏移地址:0x70

复位值:0x0000 0000

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
				I4SEL<3:0>								I3SEL<3:0>								I2SEL<3:0>								I1SEL<3:0>			

—	Bits 31-28	—	—
I4SEL	Bits 27-24	R/W	**CH4输入来源选择** 0000: 经AD16C6T_CHRMP寄存器通道映像配置后的CH4引脚输入 其他: 参考引脚输入源章节
—	Bits 23-20	—	—

I3SEL	Bits 19-16	R/W	**CH3输入来源选择** 0000: 经AD16C6T_CHRMP寄存器通道映像配置后的CH3引脚输入 其他: 参考引脚输入源章节
—	Bits 15-12	—	—
I2SEL	Bits 11-8	R/W	**CH2输入来源选择** 0000: 经AD16C6T_CHRMP寄存器通道映像配置后的CH2引脚输入 其他: 参考引脚输入源章节
—	Bits 7-4	—	—
I1SEL	Bits 3-0	R/W	**CH1输入来源选择** 0000: 经AD16C6T_CHRMP寄存器通道映像配置后的CH1引脚输入 其他: 参考引脚输入源章节

20. 6. 2. 30 复用功能寄存器 1 (AD16C6T_AFR1)

复用功能寄存器 1 (AD16C6T_AFR1)

偏移地址:0x74

复位值:0x0000 0001

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
														ETRSEL<3:0>					BRKEXTP	BRKCMP2P	BRKCMP1P	BRKINP						BRKEXTEN	BRKCMP2EN	BRKCMP1EN	BRKINEN

—	Bits 31-18	—	—
ETRSEL	Bits 17-14	R/W	**ETR输入来源选择** 0000: AD16C6T_ETR输入 其他: 参考引脚输入源章节 注:当AD16C6T_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
—	Bit 13	—	—
BRKEXTP	Bit 12	R/W	**选择通道刹车BKIN_EXT极性** 0:BKIN_EXT引脚输入不反相 1:BKIN_EXT引脚输入反相 注1:当AD16C6T_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

BRKCMP2P	Bit 11	R/W	**选择通道刹车CMP2极性** 0:刹车输入CMP2为高电平有效 1:刹车输入CMP2为低电平有效 注:当AD16C6T_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
BRKCMP1P	Bit 10	R/W	**选择通道刹车CMP1极性** 0:刹车输入CMP1为高电平有效 1:刹车输入CMP1为低电平有效 注:当AD16C6T_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
BRKINP	Bit 9	R/W	**选择通道刹车BKIN极性** 0:BKIN引脚输入不反相 1:BKIN引脚输入反相 注1:当AD16C6T_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2:最终刹车有效电平需搭配AD16C6T_BDCFG寄存器中的BRKP极性决定
—	Bits 8-4	—	—
BRKEXTEN	Bit 3	R/W	**通道刹车BKIN_EXT开启** BKIN_EXT输出与其他刹车来源做"OR"运算 0:刹车输入BKIN_EXT禁止 1:刹车输入BKIN_EXT开启 注:当AD16C6T_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
BRKCMP2EN	Bit 2	R/W	**通道刹车CMP2开启** CMP2输出与其他刹车来源做"OR"运算 0:刹车输入CMP2禁止 1:刹车输入CMP2开启 注:当AD16C6T_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
BRKCMP1EN	Bit 1	R/W	**通道刹车CMP1开启** CMP1输出与其他刹车来源做"OR"运算 0:刹车输入CMP1禁止 1:刹车输入CMP1开启 注:当AD16C6T_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
BRKINEN	Bit 0	R/W	**通道刹车BKIN开启**

			BKIN输出与其他刹车来源做"OR"运算 0:刹车输入BKIN禁止 1:刹车输入BKIN开启 注:当AD16C6T_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

20. 6. 2. 31 复用功能寄存器 2 (AD16C6T_AFR2)

复用功能寄存器 2 (AD16C6T_AFR2)

偏移地址:0x78

复位值:0x0000 0001

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
													OCLRSEL<2:0>						BRK2EXTP	BRK2CMP2P	BRK2CMP1P	BRK2INP						BRK2EXTEN	BRK2CMP2EN	BRK2CMP1EN	BRK2INEN

—	Bits 31-19	—	—
OCLRSEL	Bits 18-16	R/W	**外部清除来源选择** 000: CMP1 001: CMP2 其他: 保留 注:当AD16C6T_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
—	Bits 15-13	—	—
BRK2EXTP	Bit 12	R/W	**选择通道刹车BKIN_EXT极性** 0:BKIN_EXT引脚输入不反相 1:BKIN_EXT引脚输入反相 注1:当AD16C6T_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
BRK2CMP2P	Bit 11	R/W	**选择通道刹车CMP2极性** 0:刹车输入CMP2为高电平有效 1:刹车输入CMP2为低电平有效 注:当AD16C6T_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
BRK2CMP1P	Bit 10	R/W	**选择通道刹车CMP1极性** 0:刹车输入CMP1为高电平有效 1:刹车输入CMP1为低电平有效

			注:当AD16C6T_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
BRK2INP	Bit 9	R/W	**选择通道刹车BKIN2极性** 0:BKIN2引脚输入不反相 1:BKIN2引脚输入反相 注1:当AD16C6T_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2:最终刹车有效电平需搭配AD16C6T_BDCFG寄存器中的BRK2P极性决定
—	Bits 8-4	—	—
BRK2EXTEN	Bit 3	R/W	**通道刹车BKIN_EXT开启** BKIN_EXT输出与其他刹车来源做"OR"运算 0:刹车输入BKIN_EXT禁止 1:刹车输入BKIN_EXT开启 注:当AD16C6T_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
BRK2CMP2EN	Bit 2	R/W	**通道刹车CMP2开启** CMP2输出与其他刹车来源做"OR"运算 0:刹车输入CMP2禁止 1:刹车输入CMP2开启 注:当AD16C6T_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
BRK2CMP1EN	Bit 1	R/W	**通道刹车CMP1开启** CMP1输出与其他刹车来源做"OR"运算 0:刹车输入CMP1禁止 1:刹车输入CMP1开启 注:当AD16C6T_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
BRK2INEN	Bit 0	R/W	**通道刹车BKIN2开启** BKIN2输出与其他刹车来源做"OR"运算 0:刹车输入BKIN2禁止 1:刹车输入BKIN2开启 注:当AD16C6T_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

20. 6. 2. 32 触发信号选择寄存器（AD16C6T_TGOSEL)

此控制寄存器可被两个地址共同存取

触发信号选择寄存器（AD16C6T_TGOSEL)																													
偏移地址:0x7C、0xBC																													
复位值:0x0000 0000																													
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
										CH6U	CH5U	CH4U	CH3U	CH2U	CH1U											CH6D	CH5D	CH4D	CH3D	CH2D	CH1D

—	Bits 31-22	—	—
CH6U	Bit 21	R/W	**递增计数下的CH6触发选择** 0: 当计数器递增时，禁止CH6输出比较事件(1T)输出到TRGOUT/TRGOUT2 1: 当计数器递增时，开启CH6输出比较事件(1T)输出到TRGOUT/TRGOUT2
CH5U	Bit 20	R/W	**递增计数下的CH5触发选择** 0: 当计数器递增时，禁止CH5输出比较事件(1T)输出到TRGOUT/TRGOUT2 1: 当计数器递增时，开启CH5输出比较事件(1T)输出到TRGOUT/TRGOUT2
CH4U	Bit 19	R/W	**递增计数下的CH4触发选择** 0: 当计数器递增时，禁止CH4输出比较事件(1T)输出到TRGOUT/TRGOUT2 1: 当计数器递增时，开启CH4输出比较事件(1T)输出到TRGOUT/TRGOUT2
CH3U	Bit 18	R/W	**递增计数下的CH3触发选择** 0: 当计数器递增时，禁止CH3输出比较事件(1T)输出到TRGOUT/TRGOUT2 1: 当计数器递增时，开启CH3输出比较事件(1T)输出到TRGOUT/TRGOUT2
CH2U	Bit 17	R/W	**递增计数下的CH2触发选择** 0: 当计数器递增时，禁止CH2输出比较事件(1T)输出到TRGOUT/TRGOUT2 1: 当计数器递增时，开启CH2输出比较事件(1T)输出到TRGOUT/TRGOUT2
CH1U	Bit 16	R/W	**递增计数下的CH1触发选择**

			0: 当计数器递增时，禁止CH1输出比较事件(1T)输出到TRGOUT/TRGOUT2 1: 当计数器递增时，开启CH1输出比较事件(1T)输出到TRGOUT/TRGOUT2
—	Bits 15-6	—	—
CH6D	Bit 5	R/W	**递减计数下的CH6触发选择** 0: 当计数器递减时，禁止CH6输出比较事件(1T)输出到TRGOUT/TRGOUT2 1: 当计数器递减时，开启CH6输出比较事件(1T)输出到TRGOUT/TRGOUT2
CH5D	Bit 4	R/W	**递减计数下的CH5触发选择** 0: 当计数器递增时，禁止CH5输出比较事件(1T)输出到TRGOUT/TRGOUT2 1: 当计数器递减时，开启CH5输出比较事件(1T)输出到TRGOUT/TRGOUT2
CH4D	Bit 3	R/W	**递减计数下的CH4触发选择** 0: 当计数器递减时，禁止CH4输出比较事件(1T)输出到TRGOUT/TRGOUT2 1: 当计数器递减时，开启CH4输出比较事件(1T)输出到TRGOUT/TRGOUT2
CH3D	Bit 2	R/W	**递减计数下的CH3触发选择** 0: 当计数器递减时，禁止CH3输出比较事件(1T)输出到TRGOUT/TRGOUT2 1: 当计数器递减时，开启CH3输出比较事件(1T)输出到TRGOUT/TRGOUT2
CH2D	Bit 1	R/W	**递减计数下的CH2触发选择** 0: 当计数器递减时，禁止CH2输出比较事件(1T)输出到TRGOUT/TRGOUT2 1: 当计数器递减时，开启CH2输出比较事件(1T)输出到TRGOUT/TRGOUT2
CH1D	Bit 0	R/W	**递减计数下的CH1触发选择** 0: 当计数器递减时，禁止CH1输出比较事件(1T)输出到TRGOUT/TRGOUT2 1: 当计数器递减时，开启CH1输出比较事件(1T)输出到TRGOUT/TRGOUT2

注: 此设定需要搭配 AD16C6T_CON2 寄存器的 MMSEL、MM2SEL 为多点触发模式使用。

20. 6. 2. 33 通道映像寄存器 (AD16C6T_CHRMP)

通道映像寄存器 (AD16C6T_CHRMP)																													
偏移地址:0x80																													
复位值:0x7654 3210																													
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
CH4NRMP<3:0>				CH3NRMP<3:0>				CH2NRMP<3:0>				CH1NRMP<3:0>				CH4RMP<3:0>				CH3RMP<3:0>				CH2RMP<3:0>				CH1RMP<3:0>			

CH4NRMP	Bits 31-28	R/W	**CH4N映射选择** 参照CH1_RMP描述
CH3NRMP	Bits 27-24	R/W	**CH3N映射选择** 参照CH1_RMP描述
CH2NRMP	Bits 23-20	R/W	**CH2N映射选择** 参照CH1_RMP描述
CH1NRMP	Bits 19-16	R/W	**CH1N映射选择** 参照CH1_RMP描述
CH4RMP	Bits 15-12	R/W	**CH4映射选择** 参照CH1_RMP描述
CH3RMP	Bits 11-8	R/W	**CH3映射选择** 参照CH1_RMP描述
CH2RMP	Bits 7-4	R/W	**CH2映射选择** 参照CH1_RMP描述
CH1RMP	Bits 3-0	R/W	**CH1映射选择** 0000: AD16C6T_CH1 0001: AD16C6T_CH2 0010: AD16C6T_CH3 0011: AD16C6T_CH4 0100: AD16C6T_CH1N 0101: AD16C6T_CH2N 0110: AD16C6T_CH3N 0111: AD16C6T_CH4N 其他: 保留 详细内容可参考引脚选择与内部连接表章节

第21章 通用定时器 32 位 4 通道(GP32C4T)

21. 1 概述

通用定时器 32 位 4 通道(GP32C4T)是一个设置灵活的定时器模块，它包含一个 32 位计数器，具有定时、计数、脉冲输入信号测量(输入捕获)、产生特定 PWM 波形(输出比较)等功能。

21. 2 特性

- 三種 32 位元自動重載計數器模式
  - 递增
  - 递减
  - 递增/递减
- 16 位可编程预分频器，可在定时器运行时对计数器工作时钟进行 1 到 65536 之间的任意分频
- 带有 4 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(边沿和中心对齐模式)
  - 单脉冲输出
- 同步电路用于外部信号控制定时器及内部互联多个定时器
- 下列事件支持产生中断与 DMA 请求:
  - 更新事件:计数器上溢或下溢，计数器初始化(通过软件或内部与外部触发)
  - 触发事件:计数器开启、停止、初始化或通过内部与外部触发计数
  - 输入捕获
  - 输出比较
- 支援增量(正交)编码及霍尔电路进行定位
- 外部时钟输入触发计数器

21. 3结构图

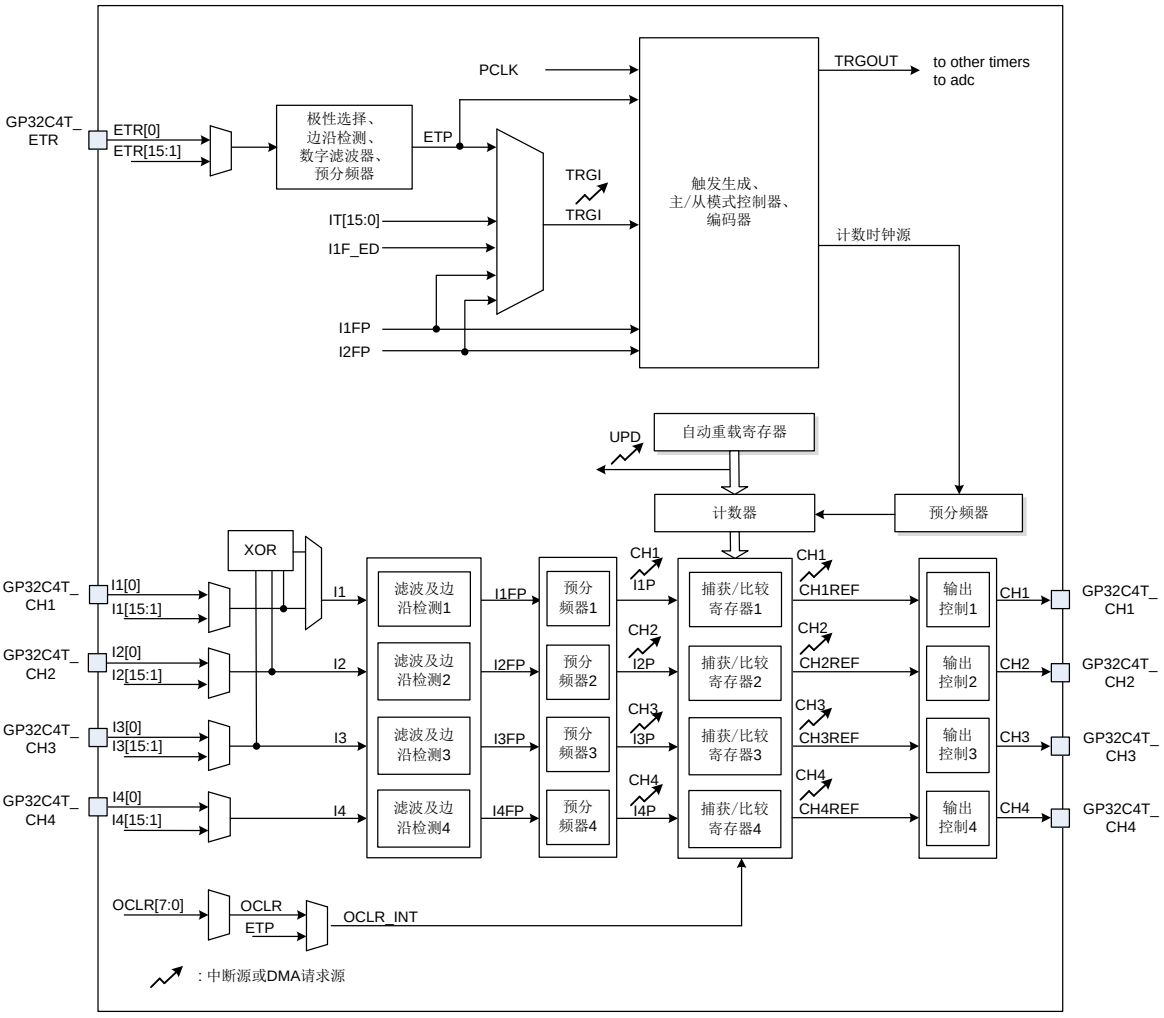


图 22-1　GP32C4T 定时器结构框图

21. 4 引脚选择与内部连接表

21. 4. 1 引脚输入源

设置 **GP32C4T_CHISEL** 寄存器，可选择通道 1 到通道 4 引脚输入来源，如下表 ：

I1SEL	GP32C4T1	GP32C4T2
0000	GP32C4T1_CH1	GP32C4T2_CH1
0001	CMP1	CMP1
0010	CMP2	CMP2
其余	保留	

表 22-1 通道 1 引脚输入来源

I2SEL	GP32C4T1	GP32C4T2
0000	GP32C4T1_CH2	GP32C4T2_CH2
0001	CMP1	CMP1
0010	CMP2	CMP2
其余	保留	

表 22-2 通道 2 引脚输入来源

I3SEL	GP32C4T1	GP32C4T2
0000	GP32C4T1_CH3	GP32C4T2_CH3
0001	CMP1	CMP1
0010	CMP2	CMP2
其余	保留	

表 22-3 通道 3 引脚输入来源

I4SEL	GP32C4T1	GP32C4T2
0000	GP32C4T1_CH4	GP32C4T2_CH4
0001	CMP1	CMP1
0010	CMP2	CMP2
其余	保留	

表 22-4 通道 4 引脚输入来源

设置 **GP32C4T_AFR1** 寄存器的 ETRSEL 位，可选择 ETR 引脚输入来源，如下表:

ETRSEL	GP32C4T1	GP32C4T2
0000	GP32C4T1_ETR	GP32C4T2_ETR
0001	CMP1	CMP1
0010	CMP2	CMP2
1000	GP32C4T2_ETR	GP32C4T1_ETR
1011	ADC1_ANALOG_WD1	ADC2_ANALOG_WD1
1100	ADC1_ANALOG_WD2	ADC2_ANALOG_WD2
其余	保留	

表 22-5　ETR 引脚输入来源

21. 4. 2　内部触发连接表

从定时器	IT0(TSSEL=00000)	IT1(TSSEL=00001)	IT2(TSSEL=00010)	IT3(TSSEL=00011)	IT4(TSSEL=01000)	IT5(TSSEL=01001)	IT6(TSSEL=01010)	IT7(TSSEL=01011)
GP32C4T1	GP32C4T1	GP32C4T1	GP32C4T1	GP32C4T1	GP32C4T1	GP32C4T1	GP32C4T1	GP32C4T1

表 22-6　GP32C4T1 内部触发连接表

从定时器	IT0(TSSEL=00000)	IT1(TSSEL=00001)	IT2(TSSEL=00010)	IT3(TSSEL=00011)	IT4(TSSEL=01000)	IT5(TSSEL=01001)	IT6(TSSEL=01010)	IT7(TSSEL=01011)
GP32C4T2	GP32C4T2	GP32C4T2	GP32C4T2	GP32C4T2	GP32C4T2	GP32C4T2	GP32C4T2	GP32C4T2

表 22-7　GP32C4T2 内部触发连接表

21. 5 功能描述

21. 5. 1 定时单位

定时器包含一个 32 位元的计数器(**GP32C4T_COUNT**)，计数时钟由预分频寄存器(**GP32C4T_PRES**)进行分频。计数周期由自动重载计数器(**GP32C4T_AR**)设定。

自动重载寄存器(**GP32C4T_AR**)是一个可缓冲的寄存器。设置 **GP32C4T_CON1** 寄存器的 ARPEN 位为 0 时，关闭 **GP32C4T_AR** 寄存器缓冲功能，写入 **GP32C4T_AR** 寄存器的值会被立即反应到缓冲寄存器中；而设置 ARPEN 位为 1 时，**GP32C4T_AR** 寄存器具有缓冲功能，只有当产生更新事件(UPD)时，**GP32C4T_AR** 寄存器的重载值才会被更新到缓冲寄存器中。

设置 **GP32C4T_CON1** 寄存器的 DISUE 位为 0 时，计数器递增计数达到上溢值或递减达到下溢值时会产生更新事件(UPD)。另外可以通过 **GP32C4T_SGE** 寄存器的 SGUPD 位为 1 产生软件更新事件。设置 **GP32C4T_CON1** 寄存器的 CNTEN 为 1 时，计数器开始计数。

注:计数器在设置 CNTEN 位为 1 后，在 1 个定时器时钟周期后开始计数。

预分频器可对定时器工作时钟进行 **GP32C4T_PRES** 寄存器数值+1 次分频。由于 **GP32C4T_PRES** 是一个可缓冲寄存器，因此定时器运行时可以对该寄存器进行修改，修改值在下次更新事件(UPD)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。

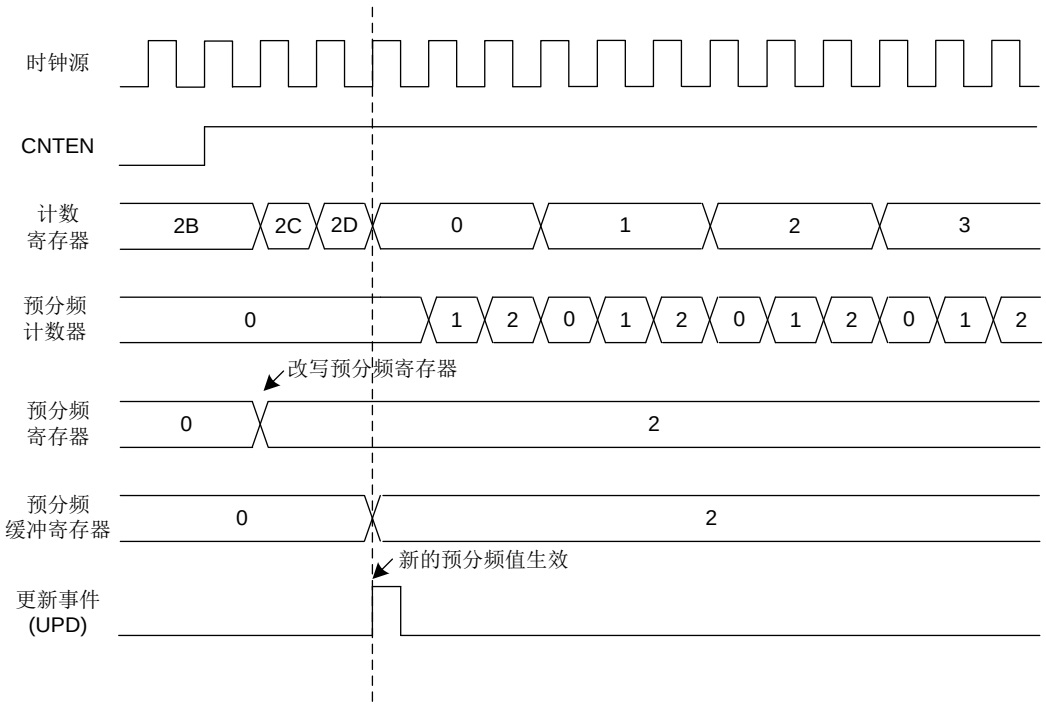


图 22-2 从 1 分频变为 3 分频时的计数时序图

21. 5. 2 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(TRGI)、外部时钟源 2(ETP)与编码器输入。

21. 5. 2. 1 内部时钟源(INT_CLK)

若从模式控制器被关闭(**GP32C4T_SMCON** 寄存器的 SMODS 位为 0000b)，则 **GP32C4T_CON1** 寄存器的 CNTEN、DIRSEL 位与 **GP32C4T_SGE** 寄存器的 SGUPD 位为控制位，这些位只能软件修改(SGUPD 位除外，仍由硬件自动清除)。一旦设置 CNTEN 位为 1，预分频器就由内部 INT_CLK 提供时钟。

INT_CLK 时钟来源为 APB 时钟(PCLK)，可以参考复位与时钟控制单元(RCU)。

下图给出了正常模式下没有分频控制电路和递增计数的情况。

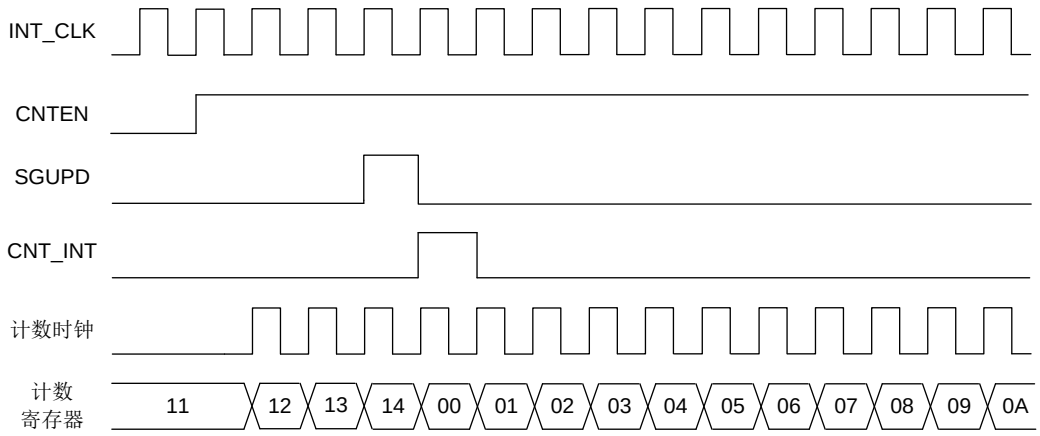


图 22-3 采用内部时钟计数

21. 5. 2. 2 外部时钟源 1

GP32C4T_SMCON 寄存器的 SMODS 位为 0111b 时，可选择外部时钟源 1。计数器可根据选定输入信号的上升沿或下降沿计数。

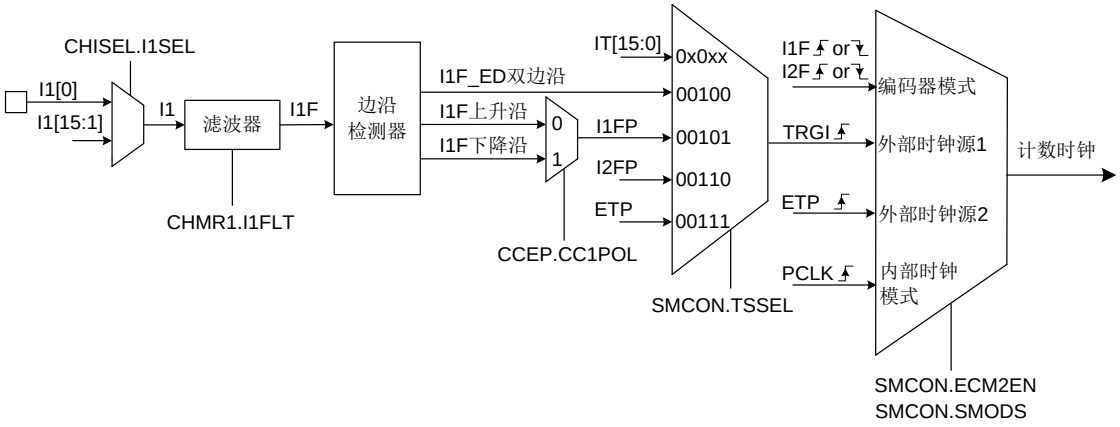


图 22-4 外部时钟连接

设置计数器外部时钟源为 I1 输入，并在 I1 上升沿时计数，步骤如下:

1. 设置 **GP32C4T_CHISEL** 寄存器的 CH1SEL 位为 0000b，选择 I1 由 IO 输入(默认值皆为 IO 输入，后续不再特别描述)。
2. 设置 **GP32C4T_CHMR1** 寄存器 CC1SSEL 位为 01b，让通道 1 为 I1 输入
3. 设置 **GP32C4T_CHMR1** 寄存器的 I1FLT 位，输入滤波器时间(若没有滤波器需求，维持 I1FLT 位为 0000b)。
4. 设置 **GP32C4T_CCEP** 寄存器的 CC1NPOL 位为 0、CC1POL 位为 0，选择极性为上升沿。
5. 设置 **GP32C4T_SMCON** 寄存器的 TSSEL 位为 00101b，选择外部时钟源为 I1。
6. 设置 **GP32C4T_SMCON** 寄存器的 SMODS 位为 0111b，选择外部时钟模式 1。
7. 设置 **GP32C4T_CON1** 寄存器的 CNTEN 位为 1，开启计数器。

当 I1 上出现一次上升沿时，计数器计数一次且设置 TRGI 标志位为 1。I1 上升沿与实际时钟间的延迟，取决于 I1 输入的同步电路。

设置计数器外部时钟源为 IT6 输入，并在 IT6 上升沿时计数，步骤如下:

1. 设置 **GP32C4T_SMCON** 寄存器的 TSSEL 位为 01010b，选择外部时钟源为 IT6。
2. 设置 **GP32C4T_SMCON** 寄存器的 SMODS 位为 0111b，选择外部时钟模式 1。
3. 设置 **GP32C4T_CON1** 寄存器的 CNTEN 位为 1，开启计数器。

IT6 产生上升沿时，计数器计数一次。

注:以 GP32C4T1 为例，IT6 即为 GP16C2T1 的 TRGOUT，详细可参考内部触发连结表。

21. 5. 2. 3　外部时钟源 2

设置 **GP32C4T_SMCON** 寄存器的 ECM2EN 位为 1 选定外部时钟源 2。

计数器可对外部触发输入 ETR 进行上升沿或下降沿计数。

下图给出了外部触发输入模块的概况。

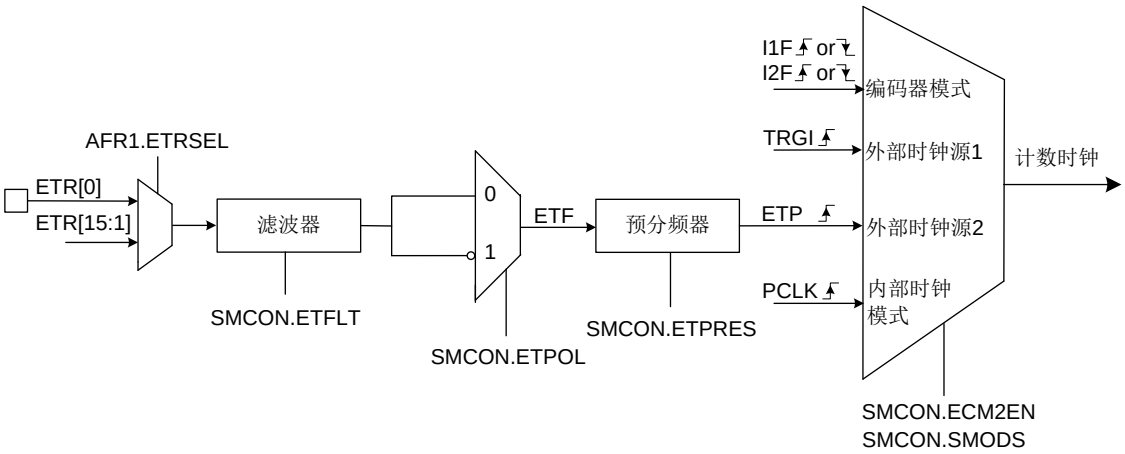


图 22-5　外部触发输入模块

设置计数器为外部时钟源 2，设置过程如下:

1. 设置 **GP32C4T_SMCON** 寄存器的 ETFLT 位，输入滤波器时间(若没有滤波器需求，维持 ETFLT 位为 0000b)预装载。
2. 设置 **GP32C4T_SMCON** 寄存器的 ETPRES 位为 0，关闭外部触发时钟预分频器。
3. 设置 **GP32C4T_SMCON** 寄存器的 ETPOL 位，检测 ETR 引脚上升沿或下降沿。
4. 设置 **GP32C4T_SMCON** 寄存器的 ECM2EN 位为 1，开启外部时钟模式 2。
5. 设置 **GP32C4T_CON1** 寄存器的 CNTEN 位为 1，开启计数器。

计数器在每个 ETR 上升沿计数一次。

21. 5. 3 计数模式

21. 5. 3. 1 递增计数模式

设置 **GP32C4T_CON1** 寄存器的 DIRSEL 位为 0 时，定时器设置为递增模式，计数器从 0 开始递增，直至 **GP32C4T_AR** 寄存器数值；然后从 0 重新开始计数并产生一个更新事件(UPD)。

设置 **GP32C4T_SGE** 寄存器的 SGUPD 位为 1(通过软件或使用从模式控制器)同样会产生更新事件，并让计数器和预分频器都重新从 0 开始计数。

通过软件设置 **GP32C4T_CON1** 寄存器中的 DISUE 位为 1 可关闭更新事件(UPD)产生。可以避免在写入预装载寄存器数值时产生更新事件(UPD)更新缓冲寄存器。在设置 DISUE 位为 0 之前都不会产生更新事件(UPD)。

此外，**GP32C4T_CON1** 寄存器中的 UERSEL 位为 1 时，设置 SGUPD 位为 1 会产生更新事件(UPD)，但不会更新更新标志位(**GP32C4T_RIF** 寄存器的 UPD 位)，也不会产生中断或 DMA 请求。在这个配置下，发生捕获事件时将计数器清零，将不会同时产生更新中断和捕获中断。

发生更新事件(UPD)时，所有预装载寄存器会更新到缓冲寄存器:

- 更新 **GP32C4T_AR** 寄存器数值到缓冲寄存器
- 更新 **GP32C4T_PRES** 寄存器数值到缓冲寄存器

下图为设置 **GP32C4T_AR** 寄存器为 16h，预分频设为 2 分频时的计数器时序。

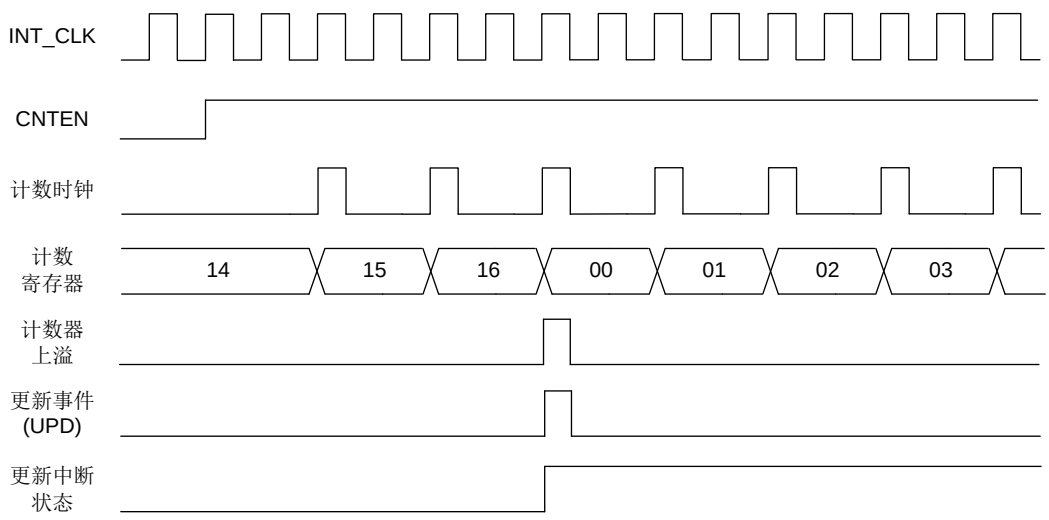


图 22-6 计数器递增计数时序图

下图为设置 **GP32C4T_AR** 寄存器从 F5h 改成 16h，分别在 ARPEN 为 0 或 1 时的计数器时序。

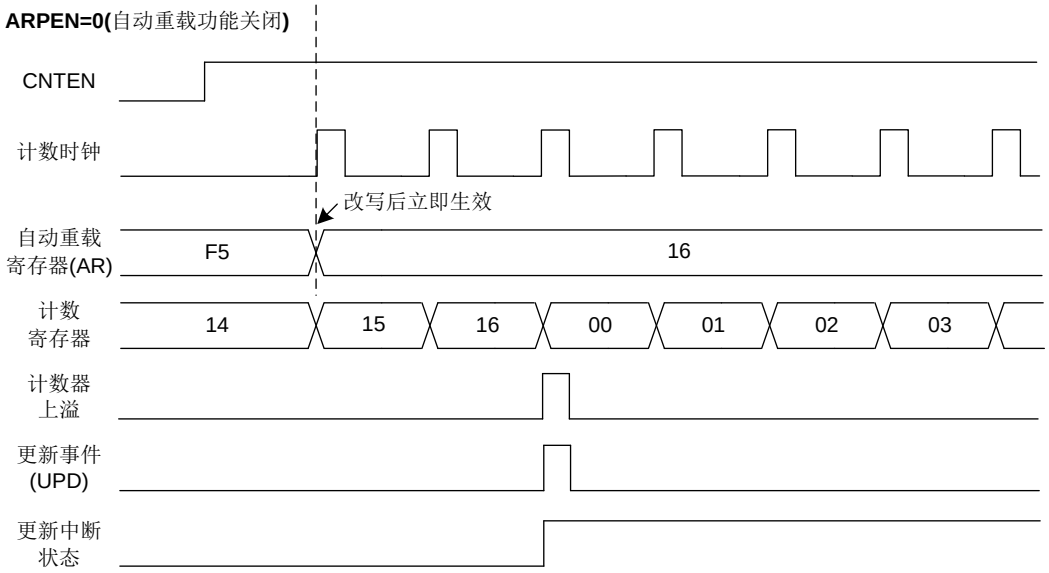


图 22-7　设置 ARPEN 位为 0 时计数器时序图

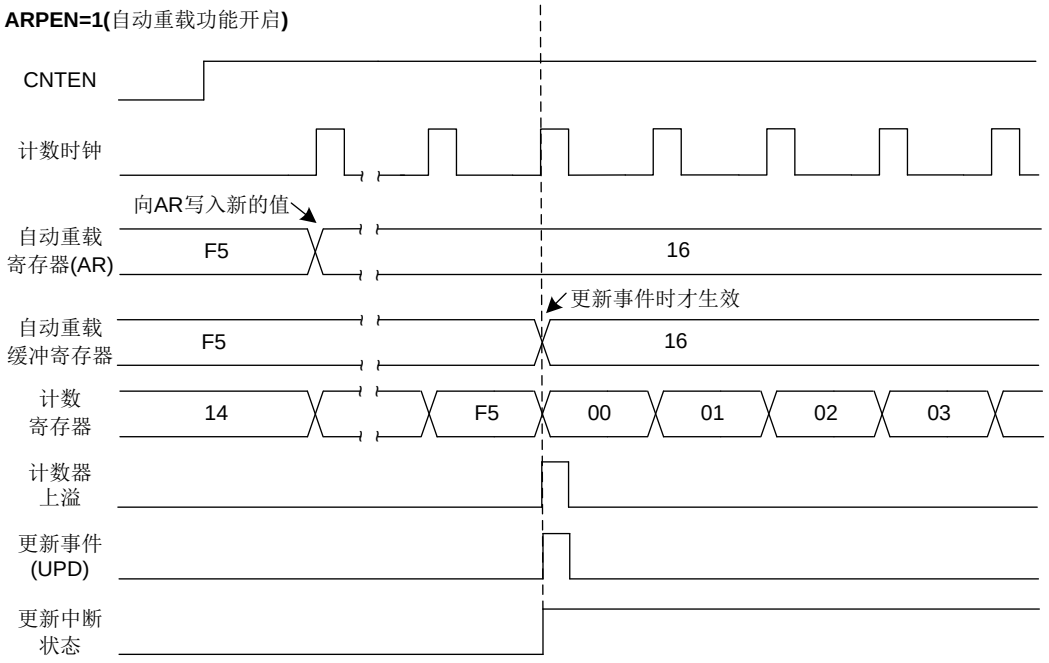


图 22-8　设置 ARPEN 位为 1 时计数器时序图

21. 5. 3. 2 递减计数模式

设置 **GP32C4T_CON1** 寄存器的 DIRSEL 位为 1 时，定时器设置为递减模式，计数器从 **GP32C4T_AR** 寄存器数值开始递减至 0；然后从 **GP32C4T_AR** 寄存器数值重新递减并产生更新事件(UPD)。

设置 **GP32C4T_SGE** 寄存器的 SGUPD 位为 1(通过软件或使用从模式控制器)同样会产生更新事件，并让计数器从自动重载值开始计数，而预分频器从 0 开始计数。

通过软件设置 **GP32C4T_CON1** 寄存器中的 DISUE 位为 1 可关闭更新事件(UPD)产生。可以避免在写入预装载寄存器数值时产生更新事件(UPD)更新缓冲寄存器。在设置 DISUE 位为 0 之前都不会产生更新事件(UPD)。

此外，**GP32C4T_CON1** 寄存器中的 UERSEL 位为 1 时，设置 SGUPD 位为 1 会产生更新事件(UPD)，但不会更新更新标志位(**GP32C4T_RIF** 寄存器的 UPD 位)，也不会产生中断或 DMA 请求。在这个配置下，发生捕获事件时将计数器清零，将不会同时产生更新中断和捕获中断。

发生更新事件(UPD)时，所有预装载寄存器会更新到缓冲寄存器。

下图为设置 **GP32C4T_AR** 寄存器为 27h，预分频设为 1 分频时的计数器时序。

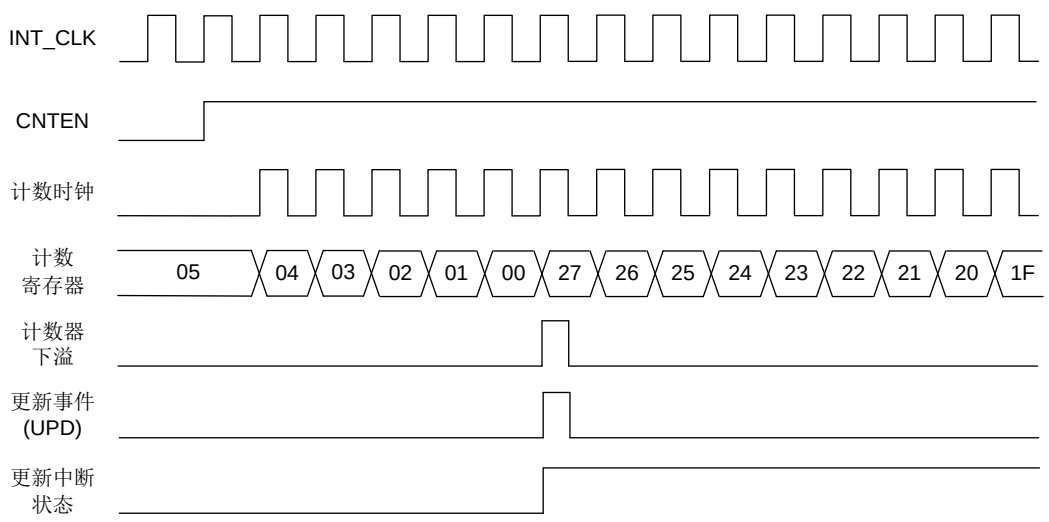


图 22-9 计数器递减计数时序图

21. 5. 3. 3 中心对齐模式

若将 **GP32C4T_CON1** 寄存器的 CMSEL 位数值设置为非 00b 值，则定时器会运作于中心对齐模式。在中心对齐模式下，计数器会从 0 开始递增至 **GP32C4T_AR** 寄存器的数值减 1，同时产生更新事件(UPD)；接着计数器会从 **GP32C4T_AR** 寄存器的数值递减至 1，并再次产生更新事件；最后计数器会从 0 重新开始计数，进行循环计数。

将通道配置为输出模式时，若要设置输出比较中断标志为 1，则需考虑定时器工作在哪种中心对齐模式下:

- 中心对齐模式 1(设置 CMSEL 位为 01b)：输出比较中断标志只产生在计数器递减计数时。
- 中心对齐模式 2(设置 CMSEL 位为 10b)：输出比较中断标志只产生在计数器递增计数时。
- 中心对齐模式 3(设置 CMSEL 位为 11b)：输出比较中断标志在计数器递增递减计数时都会产生。

在中心对齐模式下，**GP32C4T_CON1** 寄存器的 DIRSEL 位无法进行写入操作，该位由硬件自动更新指示当前计数方向。

计数上溢、下溢或者设置 **GP32C4T_SGE** 寄存器的 SGUPD 位为 1(通过软件或使用从模式控制器)都会产生更新事件，而通过软件或使用从模式控制器的方式，会让计数器和预分频器都重新从 0 开始递增计数。

通过软件设置 **GP32C4T_CON1** 寄存器中的 DISUE 位为 1，可以关闭更新事件(UPD)的产生，这可以避免在写入预装载寄存器数值时，产生更新事件(UPD)并更新缓冲寄存器。在设置 DISUE 位为 0 之前，不会产生更新事件(UPD)。

此外，当设置 **GP32C4T_CON1** 寄存器中的 UERSEL 位为 1 时，设置 SGUPD 位为 1 会产生更新事件(UPD)，但不会更新更新标志位(**GP32C4T_RIF** 寄存器的 UPD 位)，也不会产生中断或 DMA 请求。在这种配置下，当发生捕获事件时，计数器将被清零，并且不会同时产生更新中断和捕获中断。

当发生更新事件(UPD)时，所有预装载寄存器会被更新到缓冲寄存器中。

注:若发生更新事件是因为计数器的上溢，则在计数器重载之前，自动重载缓冲寄存器就已经完成了更新。因此，下一个计数周期将使用新的预装载值(即计数器重载的新 AR 值)。

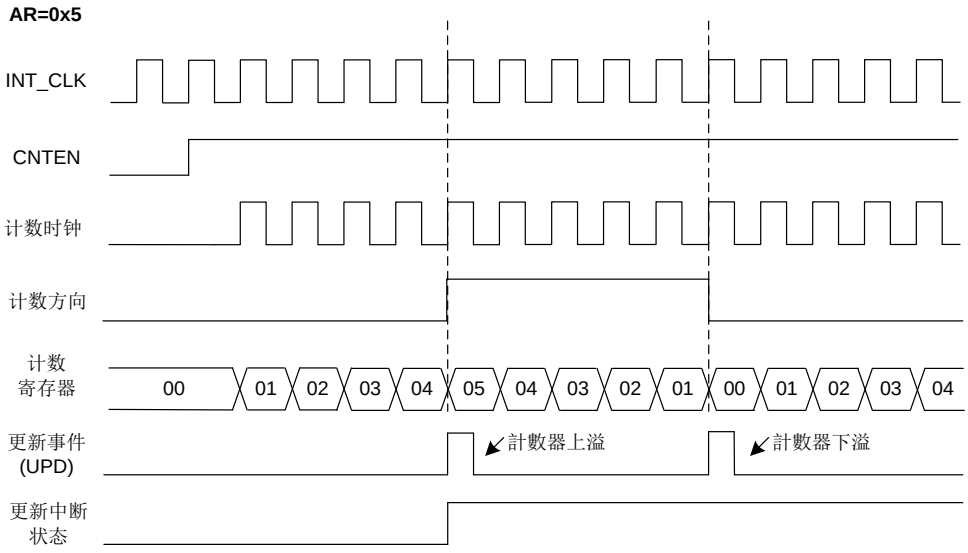


图 22-10　计数器递增减计数时序图

21. 5. 4 外部触发输入

定时器具有一个外部触发输入 ETR，主要作用于:

- 外部时钟模式。
- 从模式控制器的触发输入源。
- PWM 输出复位的触发输入源。

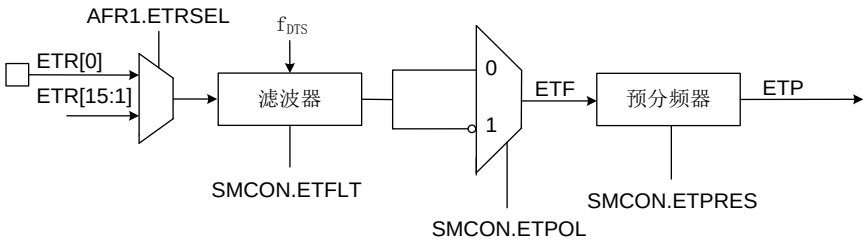


图 22-11 外部触发输入结构图

设置 **GP32C4T_AFR1** 寄存器的 ETRSEL 位，可选择 ETR 输入源为:

- ETR 输入引脚(默认选择)。
- 比较器输出(CMP1、CMP2)。
- 模拟看门狗(ADCn_AWDn_OUT)。

详细设定可参考引脚输入源章节。

21. 5. 5 捕获或比较通道

每个捕获或比较通道都包含一个捕获或比较预装载寄存器(包含缓冲寄存器)、一个输入捕获电路和一个输出比较电路。读写操作都是通过预装载寄存器进行的。在捕获模式下，实际的捕获是在缓冲寄存器中完成的，然后缓冲寄存器中的值被复制到预装载寄存器中。在比较模式下，预装载寄存器的值被复制到缓冲寄存器中，并与计数器进行比较。

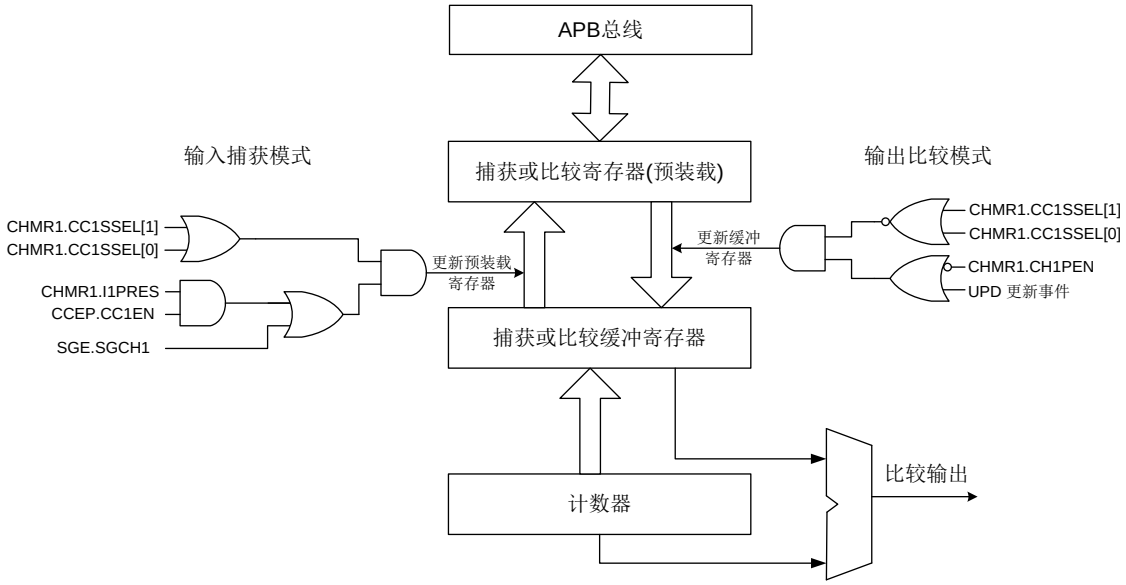


图 22-12　捕获或比较通道结构图(通道 1 为例)

21. 5. 5. 1　　输入捕获电路

输入捕获电路会对 In 输入端的信号进行采样，经过数字滤波器后产生 InF 信号，接着通过边沿检测器产生 InF 边沿信号，包括上升沿、下降沿以及双边沿。最后，根据 **GP32C4T_CCEP** 寄存器中的 CCnNPOL/CCnPOL 极性选择位，生成 InFP 信号。这个信号可以输出到从模式控制器的触发输入(TRGI)，或作为捕获输入源(InP)之一，而且还需要通过预分频器进行预处理。

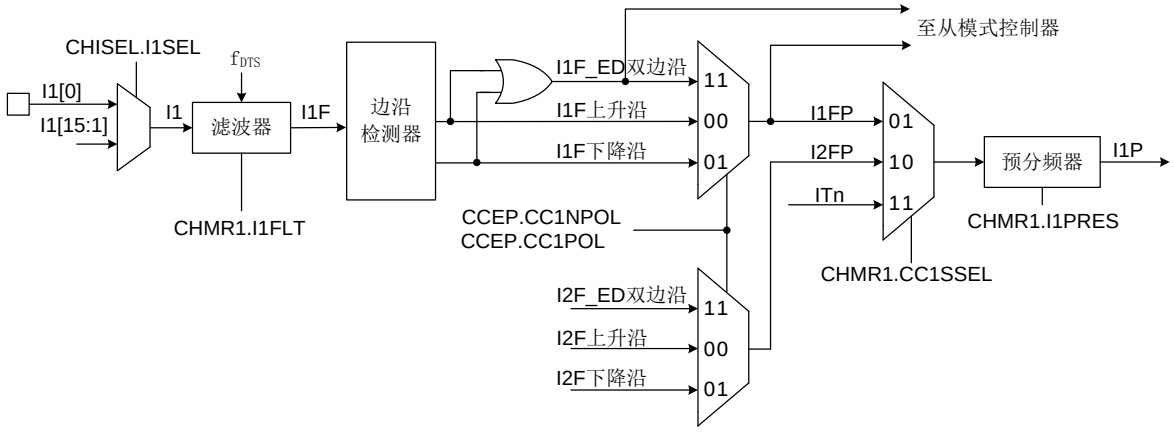


图 22-13　输入捕获电路(通道 1 为例)

21. 5. 5. 2　　输出比较电路

输出部分会根据 **GP32C4T_CHMRn** 寄存器中 CHnMOD 位的配置，产生一个输出比较参考信号 CHnREF(高电平有效)。最终输出信号的极性则由 **GP32C4T_CCEP** 寄存器中的 CCnPOL 位所决定。

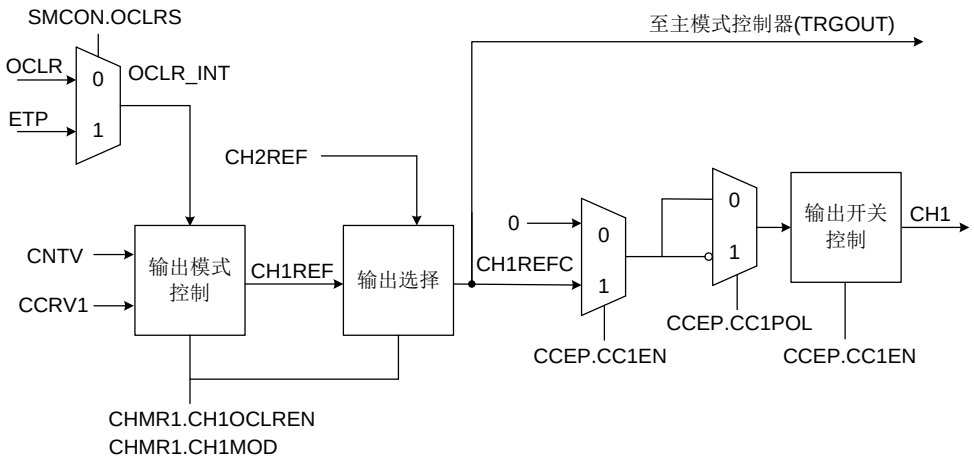


图 22-14　输出比较电路(通道 1 为例，通道 2/3/4 架构相同)

21. 5. 6　输入捕获模式

在输入捕获模式下，当 In 上检测到有效边沿变化时，计数器数值就会被锁存到捕获或比较寄存器**(GP32C4T_CCVALn)**中。此时，**GP32C4T_RIF** 寄存器中相应的 CHn 标志位会被设置为 1，同时触发中断或 DMA 请求(如果已开启)。

若相应的 CHn 标志位已经为 1，则当再次发生捕获事件时，相应的过捕获 CHnOV 标志位也会被设定为 1，表示曾经发生过捕获事件。

藉由软件将 **GP32C4T_ICR** 寄存器中的 CHn 位与 CHnOV 位设定为 1，可清除 **GP32C4T_RIF** 寄存器中相应的 CHn 与 CHnOV 标志位。

以下为以 I1 输入上升沿作为捕获输入时的流程:

1. 设置 **GP32C4T_CHMR1** 寄存器的 CC1SSEL 位为 01b，选择 I1 为有效输入端。只要 CC1SSEL 不为 00b，通道就会被设置成输入，且 **GP32C4T_CCVAL1** 寄存器为只读。
2. 设置 **GP32C4T_CHMR1** 寄存器的 I1FLT 位为 0011b，选择输入滤波器的持续时间，当 I1 检测到新的电平时，会进行连续 8 次采样，确认电平变化的有效性。
3. 设置 **GP32C4T_CCEP** 寄存器的 CC1NPOL 位为 0、CC1POL 位为 0，选择 I1 通道上升沿有效。
4. 设置 **GP32C4T_CHMR1** 寄存器的 I1PRES 位为 00b，关闭捕获预分频器，让每次有效上升沿皆执行捕获操作。
5. 如有需要，设置 **GP32C4T_IER** 寄存器的 CH1 位为 1，开启中断请求。设置 **GP32C4T_DMAEN** 寄存器的 CH1 位为 1，开启 DMA 请求。
6. 设置 **GP32C4T_CCEP** 寄存器的 CC1EN 位为 1，开启捕获计数器。

当发生输入捕获时:

1. **GP32C4T_CCVAL1** 寄存器获取计数器当前的值。
2. **GP32C4T_RIF** 寄存器的 CH1 位被设置。如果至少两个连续捕获发生，且标志位未被清除，则 CH1OV 位也被设置。
3. 中断的产生取决于 **GP32C4T_IER** 寄存器的 CH1 位。

4. DMA 请求的产生取决于 **GP32C4T_DMAEN** 寄存器的 CH1 位。

为了处理捕获溢出，建议在读取过捕获标志位之前先读取数据。这是为了避免在读取标志位和读取数据之间发生捕获溢出，从而丢失捕获讯息。

注:捕获中断请求可由软件设置 GP32C4T_SGE 寄存器的 SGCHn 位产生。

21. 5. 7 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. 设置 **GP32C4T_CHMR1** 寄存器的 CC1SSEL 位为 01b，通道 1 选择 I1 为有效输入端。
2. 设置 **GP32C4T_CCEP** 寄存器的 CC1NPOL 位为 0、CC1POL 位为 0，通道 1 选择 I1 上升沿有效，用于捕获数据到 **GP32C4T_CCVAL1** 寄存器。
3. 设置 **GP32C4T_CHMR1** 寄存器的 CC2SSEL 位为 10b，通道 2 同样选择 I1 为有效输入端。
4. 设置 **GP32C4T_CCEP** 寄存器的 CC2NPOL 位为 0、CC2POL 位为 1，通道 2 选择 I1 下降沿有效，用于捕获数据到 **GP32C4T_CCVAL2** 寄存器。
5. 设置 **GP32C4T_SMCON** 寄存器的 TSSEL1 位与 TSSEL2 位为 00101b，选择 I1FP 信号为有效的触发输入(TRGI)。
6. 设置 **GP32C4T_SMCON** 寄存器的 SMODS 位为 0100b，选择从模式控制器为复位模式，让每次有效触发输入(TRGI)初始化计数器。
7. 设置 **GP32C4T_CCEP** 寄存器的 CC1EN 位和 CC2EN 位为 1，开启捕获。
8. 设置 **GP32C4T_CON1** 寄存器的 CNTEN 位为 1，开启计数器。

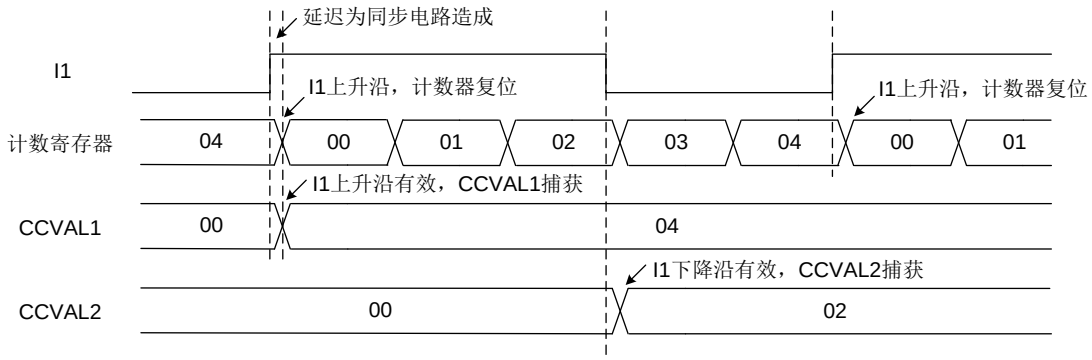


图 22-15 PWM 输入模式时序

- **GP32C4T_CCVAL1** 寄存器内的值为 PWM 周期(两次上升沿之间的时间)，此范例中 PWM 周期为 5 个计数单位。
- **GP32C4T_CCVAL2** 寄存器内的值为 PWM 脉冲宽度(上升沿到下降沿之间的时间)，此范例中 PWM 脉冲宽度为 3 个计数单位，可推算其占空比为 60%。

注:计数单位取决于时钟频率以及预分频设定值。

21. 5. 8 PWM输出模式

脉宽调制(PWM)模式可以产生一个由 **GP32C4T_AR** 寄存器设置输出频率，由 **GP32C4T_CCVALn** 寄存器设置占空比的信号。若要开启相应的预装载寄存器，需将 **GP32C4T_CHMRn** 寄存器的 CHnPEN 位设置为 1，并将 **GP32C4T_CON1** 寄存器的 ARPEN 位设置为 1 以开启自动重载功能。

在开启预装载和自动重载功能后，只有当更新事件发生时，才会将预装载寄存器的值写入到缓冲寄存器中。因此，在开始计数前，必须通过将 **GP32C4T_SGE** 寄存器的 SGUPD 位设置为 1，以初始化所有的寄存器。

CHn 的极性可以通过 **GP32C4T_CCEP** 寄存器的 CCnPOL 位设置，有效电平可以设置为高电平或低电平。CHn 的输出由 **GP32C4T_CCEP** 寄存器的 CCnEN 位控制。

在 PWM 模式 1 或 2 中，**GP32C4T_COUNT** 会持续与 **GP32C4T_CCVALn** 寄存器的数值比较，以确定 **GP32C4T_CCVALn** ≤ **GP32C4T_COUNT** 或 **GP32C4T_CCVALn** ≥ **GP32C4T_COUNT**(取决于计数器的计数方向)。

定时器产生 PWM 波形时可以是边沿对齐或中心对齐，取决于 **GP32C4T_CON1** 寄存器的 CMSEL 位。

21. 5. 8. 1 PWM边沿对齐模式

◆ 递增计数设置

设置 **GP32C4T_CON1** 寄存器的 DIRSEL 位为 0 时，计数器递增计数。

以 CH1 输出 PWM 模式 1 为例，相关配置流程如下:

1. 设置 **GP32C4T_CHMR1** 寄存器的 CH1MOD 位为 00110b，选择 PWM 模式 1。
2. 设置 **GP32C4T_CCEP** 寄存器的 CC1POL 位为 0，选择 CH1 通道输出为高电平有效。
3. 设置 **GP32C4T_CCVAL1** 寄存器的 CCRV1 位为 04h，当计数器数到 4 时，PWM 输出低电平。
4. 设置 **GP32C4T_AR** 寄存器的 ARV 位为 08h，当计数器递增到 8 后重载。
5. 设置 **GP32C4T_CON1** 寄存器的 CNTEN 位为 1，开启计数器。

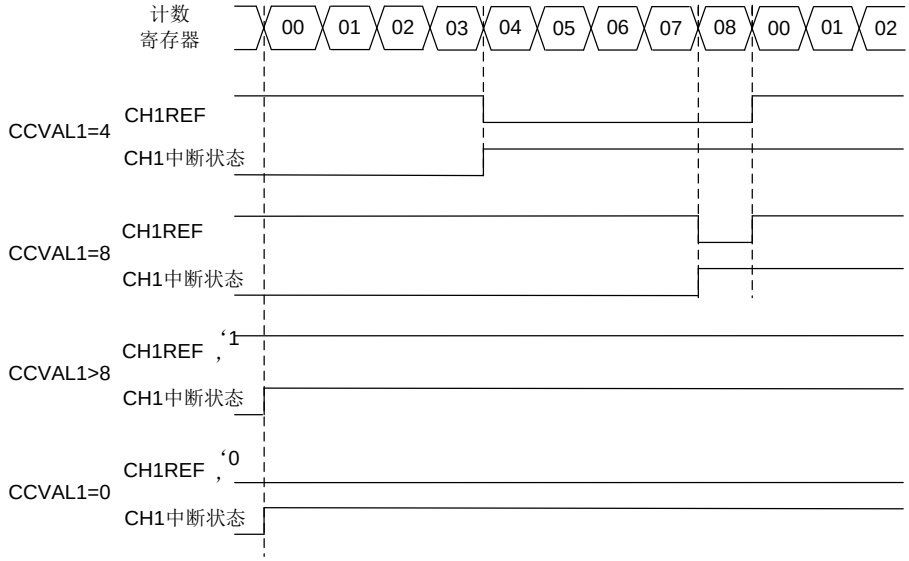


图 22-16　边沿对齐递增计数 PWM 波形(AR=8)

- **GP32C4T_COUNT< GP32C4T_CCVAL1** 时，CH1 为高电平。
- **GP32C4T_COUNT≥ GP32C4T_CCVAL1** 时，CH1 为低电平。

通过改变 **GP32C4T_CCVAL1** 寄存器的 CCRV1 位，可以改变 PWM 信号的占空比。如果将 **GP32C4T_CCVAL1** 设置为大于 **GP32C4T_AR** 的值，则 CH1 将永远输出高电平，并在计数器上溢时产生比较匹配事件。如果将 **GP32C4T_COUNT** 设置为 0，则 CH1 将永远输出低电平。

- 递减计数设置

设置 **GP32C4T_CON1** 寄存器的 DIRSEL 位为 1 时，计数器递减计数

以 CH1 输出 PWM 模式 1 为例，相关配置流程如下:

1. 设置 **GP32C4T_CON1** 寄存器的 DIRSEL 位为 1，计数器递减计数。
2. 设置 **GP32C4T_AR** 寄存器的 ARV 位为 08h，当计数器递减到 0 后重载。
3. 设置 **GP32C4T_SGE** 寄存器的 SGUPD 位为 1，软件触发更新事件，将 ARV 重载到 **GP32C4T_COUNT** 寄存器中。
4. 设置 **GP32C4T_CHMR1** 寄存器的 CH1MOD 位为 00110b，选择 PWM 模式 1。
5. 设置 **GP32C4T_CCEP** 寄存器的 CC1POL 位为 0，选择 CH1 通道输出为高电平有效。
6. 设置 **GP32C4T_CCVAL1** 寄存器的 CCRV1 位为 04h，当计数器数到 4 时，PWM 输出高电平。
7. 设置 **GP32C4T_CON1** 寄存器的 CNTEN 位为 1，开启计数器。

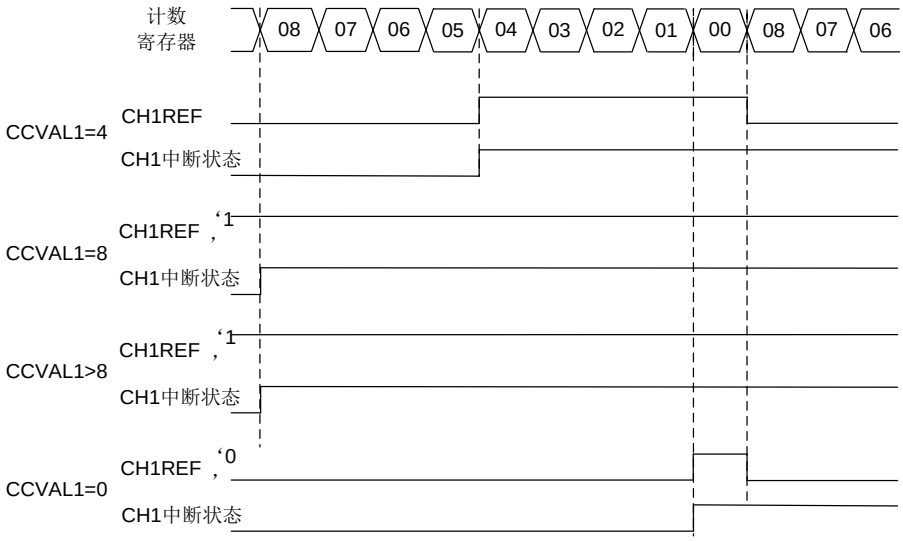


图 22-17　边沿对齐递减计数 PWM 波形(AR=8)

- **GP32C4T_COUNT≤ GP32C4T_CCVAL1** 时，CH1 为高电平。
- **GP32C4T_COUNT >GP32C4T_CCVAL1** 时，CH1 为低电平。

其中比较特别的是，若设定 **GP32C4T_CCVAL1≧ GP32C4T_AR** 时，CH1 会永远输出高电平(并在下溢时产生比较匹配事件)。此模式下不可能产生永远低电平的 PWM 波形。

21. 5. 8. 2 PWM中心对齐模式

若设置 **GP32C4T_CON1** 寄存器的 CMSEL 位不为 00b，就启用了中心对齐模式。根据 CMSEL 位的设置，计数器可以只在递增、只在递减或者在递增递减同时计数时，设置 **GP32C4T_RIF** 寄存器的比较标志位为 1。

在中心对齐模式下，计数器会在到达上溢或下溢时自动反向计数，而不需要额外的软件控制。DIRSEL 位控制计数方向的选择由硬件更新，软件无法修改。

下图为中心对齐模式下，CH1 输出 PWM 模式 1 为例，AR 位为 08h:

图 22-18 中心对齐 PWM 波形(AR=08h)

中心对齐模式的使用技巧:

- 进入中心对齐模式后，计数器会根据原本的递增或递减设置进行计数。递增或递减的方向取决于 **GP32C4T_CON1** 寄存器的 DIRSEL 位原本的设定。需要注意的是，软件不得同时修改 DIRSEL 和 CMSEL 位。
- 计数器在中心对齐模式下运行时，不建议对 **GP32C4T_COUNT** 执行写入操作。假设在递增计数的情况下，向计数器写入数值大于自动重载值(**GP32C4T_COUNT >GP32C4T_AR**)，计数方向不会更新，会持续计数下去。
- 使用中心对齐模式最安全的方式是在计数器开始计数前通过软件产生更新事件(设置 **GP32C4T_SGE** 寄存器的 SGUPD 位为 1)，并在计数器运行过程中不对 **GP32C4T_COUNT** 寄存器进行写值。这样可以保证计数器的计数方向正确，避免出现错误。

21. 5. 9 输出比较模式

这个功能被用来控制一个输出波形或是指示一段时间是否已经过去。

当捕获或比较寄存器和 **GP32C4T_COUNT** 寄存器数值匹配时:

- **GP32C4T_CHMRn** 寄存器的 CHnMOD 位选择输出模式，而输出极性由 **GP32C4T_CCEP** 寄存器的 CCnPOL 位控制 ：
  - ◇ 设置 CHnMOD 位为 00000b:当计数器匹配比较器时输出保持其电平。
  - ◇ 设置 CHnMOD 位为 00001b:当计数器匹配比较器时输出有效电平(假设 CCnPOL=0，有效电平为高电平)。
  - ◇ 设置 CHnMOD 位为 00010b:当计数器匹配比较器时输出无效电平(假设 CCnPOL=0，无效电平为低电平)。
  - ◇ 设置 CHnMOD 位为 00011b:当计数器匹配比较器时翻转电平。
- 设置中断状态寄存器的标志位为 1(**GP32C4T_RIF** 寄存器的 CHn 位)。
- 若设置相应的中断开启位为 1(**GP32C4T_IER** 寄存器的 CHn 位)，则产生中断。
- 若设置相应的开启位为 1(**GP32C4T_DMAEN** 寄存器的 CHn 位，**GP32C4T_CON2** 寄存器的 CCDMASEL 位用于 DMA 请求的选择)，则发送 DMA 请求。

设置 **GP32C4T_CHMRn** 寄存器的 CHnPEN 位数值可决定 **GP32C4T_CCVALn** 寄存器是否有缓冲功能。

在输出比较模式中，更新事件 UPD 对 CHn 的输出没有影响。输出比较模式同样可以用来输出单个脉冲(单脉冲模式)。

输出比较的设置过程:

1. 选定计数器时钟(内部、外部、预分频)。
2. 设置 **GP32C4T_AR** 与 **GP32C4T_CCVALn** 寄存器並写入所需数据。
3. 若需要产生中断请求，设置 **GP32C4T_IER** 寄存器的 CHn 位为 1。
4. 选择输出模式，例如:
   - 设置 CHnMOD 位为 00011，当 CNTV 与 CCRVALn 匹配时，CHn 输出翻转。
   - 设置 CHnPEN 位为 0，关闭预装载功能。
   - 设置 CCnPOL 位为 0，选择有效电平为高电平。
   - 设置 CCnEN 位为 1，开启输出。
5. 设置 **GP32C4T_CON1** 寄存器的 CNTEN 位为 1，开启计数器。

假设预装载寄存器开启(CHnPEN 位为 1)，设置 **GP32C4T_CCVALn** 寄存器数值在下次更新事件发生时更新至缓冲寄存器。预装载寄存器未开启(CHnPEN 位为 0)，通过设置 **GP32C4T_CCVALn** 寄存器数值可随时更新控制输出波形。

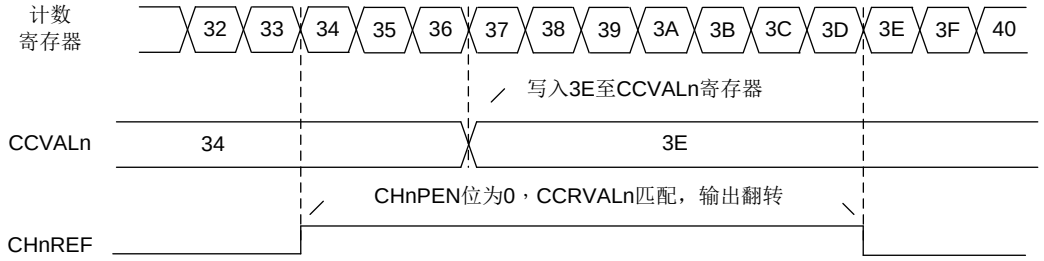


图 22-19　输出比较模式，触发 CHn

21. 5. 10 强制输出模式

在输出模式(**GP32C4T_CHMRn** 寄存器的 CCnSSEL 位为 00b)下，通过软件可以将输出比较信号强制设置为高电平或低电平，输出信号并不会参考 **GP32C4T_CCVALn** 寄存器和 **GP32C4T_COUNT** 寄存器之间的比较结果。

设置 **GP32C4T_CHMRn** 寄存器的 CHnMOD 位为 00101b，输出比较参考信号(CHnREF)为强制高电平，输出比较信号(CHn/CHnN)强制为有效电平(极性由 **GP32C4T_CCEP** 寄存器对应的 CCnPOL 位或 CCnNPOL 位决定)。反之，若设置 CHnMOD 位为 00100b 则强制设置低电平。

例如:设置 CCnPOL 位为 0(CHn 高电平有效)，则 CHn 被强制为高电平。

在此模式下，**GP32C4T_CCVALn** 寄存器和 **GP32C4T_COUNT** 寄存器之间的比较仍然进行，仍可设置相应的标志位为 1，并发送相应的中断以及 DMA 请求。

21. 5. 11 非对称PWM模式

在非对称 PWM 模式下，产生两个可编程相位移的中心对齐 PWM 信号。配置 **GP32C4T_AR** 寄存器数值决定 PWM 频率，占空比以及相位移则由成对的 **GP32C4T_CCVALn** 寄存器配置。这对寄存器分别控制递增计时和递减计时时产生的 PWM 波型，并在每半个 PWM 周期调整一次 PWM 输出。

- CH1REFC(或 CH2REFC)由配置 **GP32C4T_CCVAL1** 和 **GP32C4T_CCVAL2** 控制。
- CH3REFC(或 CH4REFC)由配置 **GP32C4T_CCVAL3** 和 **GP32C4T_CCVAL4** 控制。

2 个通道可以独立选择非对称 PWM 模式，每对 CCVALn 决定一个 CHn 输出。
设置 **GP32C4T_CHMRn** 寄存器的 CHnMOD 位写入 01110b 为非对称 PWM 模式 1，写入 01111b 为非对称 PWM 模式 2。

注:因为兼容性的关系，CHnMOD[4:0]位分为两个部分，最高两位与其他位并不连续。

在非对称 PWM 模式下，通道 1 的 CH1REFC 信号在递增计数时会使用 CCRV1 输出比较，而在递减计数时会使用 CCRV2。虽然这种模式需要成对使用 CCVALn，但相对应的通道 2 可以选择正常输出 CH2REF，或者也可以输出非对称的 CH2REFC 信号，这取决于 **GP32C4T_CHMR1** 寄存器中的 CH2MOD 设置。

下图为使用非对称 PWM 模式 2，产生两个 50%占空比、90 度相位移的 PWM 信号波型:

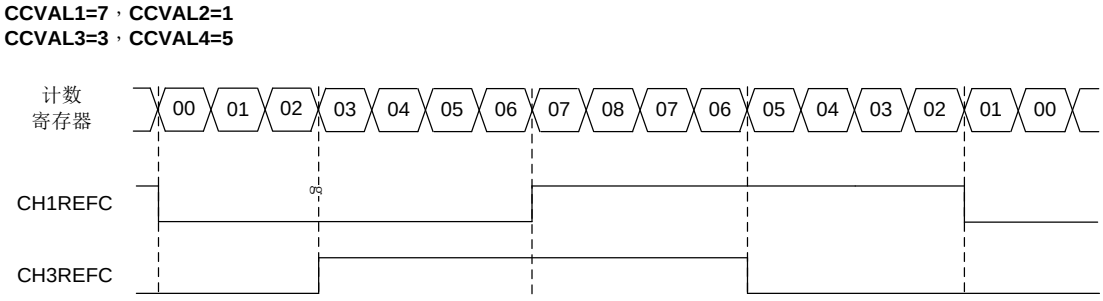


图 22-20　非对称 PWM 模式输出波型

21. 5. 12 组合PWM模式

在组合模式中，产生两个可编程延迟的边沿或中心对齐 PWM 波型。配置 **GP32C4T_AR** 寄存器数值决定 PWM 频率，PWM 占空比和延迟由两个 **GP32C4T_CCVALn** 寄存器配置。产生的 PWM 波型由两个参考的 PWM 波型做"OR"运算或"AND"运算所组成。

- CH1REFC(或 CH2REFC)由配置 **GP32C4T_CCVAL1** 和 **GP32C4T_CCVAL2** 控制。
- CH3REFC(或 CH4REFC)由配置 **GP32C4T_CCVAL3** 和 **GP32C4T_CCVAL4** 控制。

2 个通道可以独立选择非对称 PWM 模式，每对 CCVALn 决定一个 CHn 输出。

设置 **GP32C4T_CHMRn** 寄存器的 CHnMOD 位写入 01100b 为组合 PWM 模式 1，写入 01101b 为组合 PWM 模式 2。

当通道设置为组合 PWM 模式时，2 个通道必须设置成不同的 PWM 模式，例如通道 1 设置组合 PWM 模式 1，通道 2 必须设置组合 PWM 模式 2。

注:因为兼容性的关系，CHnMOD[4:0]位分为两个部分，最高两位与其他位并不连续。

下图为组合 PWM 模式中，可以产生的通道波型，通过以下配置:

- 通道 1 配置组合 PWM 模式 2。
- 通道 2 配置 PWM 模式 1。
- 通道 3 配置组合 PWM 模式 1。
- 通道 4 配置 PWM 模式 2。

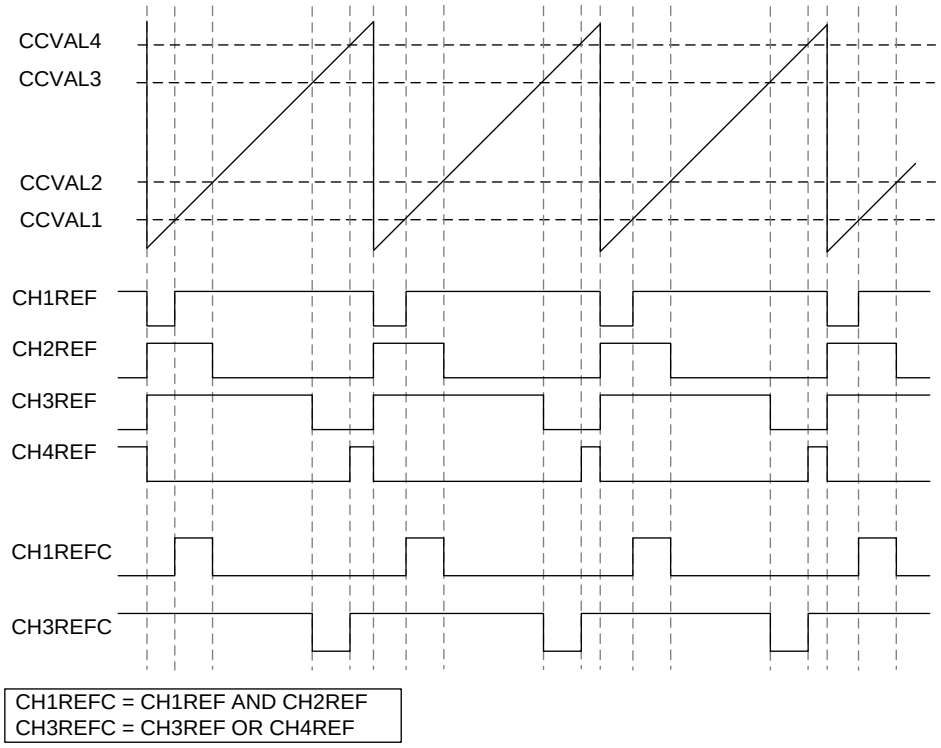


图 22-21　组合 PWM 模式输出波型

21. 5. 13 外部事件清除比较输出

设置相对应的 **GP32C4T_CHMRn** 寄存器的 CHnOCLREN 位为 1，在选定的 OCLR_INT 输入端为高准位时，可将相对应的输出信号暂时清除为 0，直到下一次更新事件(UPD)产生。该功能只能在输出比较模式和 PWM 模式下使用，在强制模式下不起作用。

如下图所示，OCLR_INT 的输入来源根据 **GP32C4T_SMCON** 寄存器的 OCLRS 位设定，可以是经过数字滤波及极性选择后的 ETP 信号，也可以是 **GP32C4T_AFR2** 寄存器的 OCLRSEL 位所指定的 OCLR 输入源。

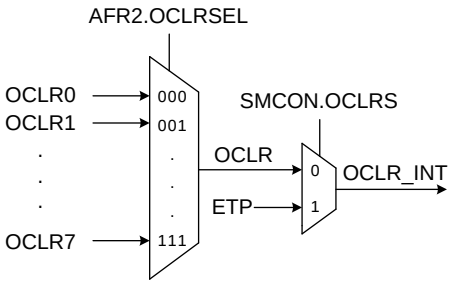


图 22-22　清除比较输出源 OCLR_INT

选择 ETR 时，ETR 配置如下:

1. 设置 **GP32C4T_SMCON** 寄存器中的 OCLRS 位为 1，选择输出通道清除来源为 ETP。

2. 设置 **GP32C4T_SMCON** 寄存器中的 ETPRES 位为 00b，关闭外部触发预分频器。
3. 设置 **GP32C4T_SMCON** 寄存器的 ECM2EN 位为 0，关闭外部时钟源 2。
4. 外部触发极性(ETPOL)和外部触发滤波器(ETFLT)可根据用户需要设置。

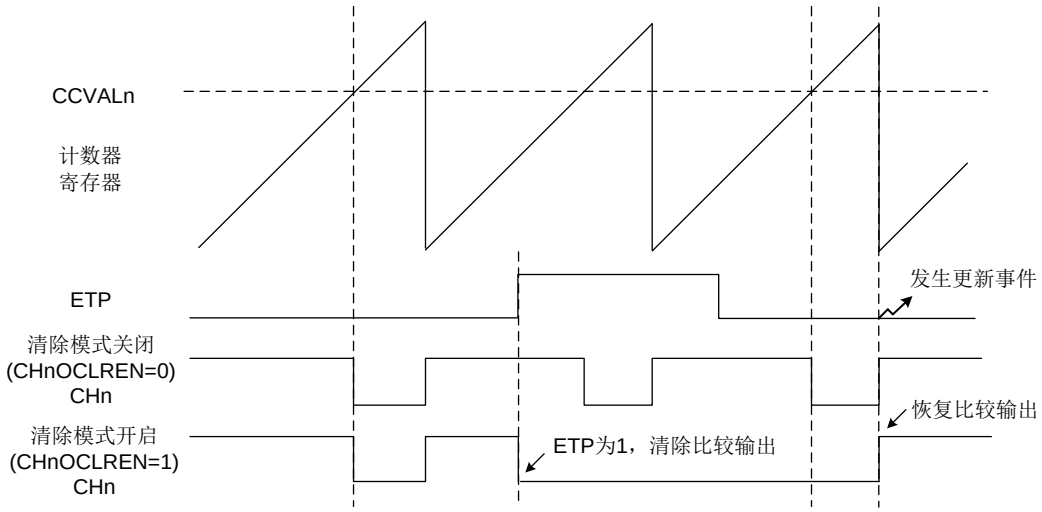


图 22-23　清除比较输出 CHn

21. 5. 14 单脉冲模式

单脉冲模式(SPMEN) 是一个特殊模式。在此模式下，计数器可以通过外部触发下启动，并可以产生一个脉宽可编程的波形。

通过从模式控制器开启计数器。在输出比较模式或 PWM 模式下生成波形。设置 **GP32C4T_CON1** 寄存器的 SPMEN 位为 1 选择单脉冲模式，在下一次发生更新事件后，计数器将自动停止计数。

只有当 **GP32C4T_CCVALn** 寄存器值和初始 **GP32C4T_COUNT** 寄存器值不同时，才能正确的产生一个脉冲。计数器开始计数前(定时器等待触发)，必须如下设置:

- 递增计数:CNTV < CCVALn ≤ AR(注意:0 < CCVALn)
- 递减计数:CNTV > CCVALn

举例来说，假设今天要达到以下效果:
当检测到 I2 输入引脚出现上升沿后，定时器自动开启计数，并经过 T 延迟的时间后，在 CH1 上产生一个长度为 T 脉冲的正脉冲。

就可以设定成单脉冲模式。

选择 I2 作为触发源，相关配置如下:

1. 设置 **GP32C4T_CHMR1** 寄存器中的 CC2SSEL 位为 01b，选择通道输入源为 I2。
2. 设置 **GP32C4T_CCEP** 寄存器中的 CC2POL、CC2NPOL 位为 0，选择 I2 上升沿有效。
3. 设置 **GP32C4T_SMCON** 寄存器的 TSSEL 位为 00110b，选择触发来源(TRGI)为 I2 滤波后信号(I2FP)。
4. 设置 **GP32C4T_SMCON** 寄存器的 SMODS 位为 0110b，选择触发模式，在检测到 TRGI

上升沿时自动开启计数器。

脉冲波型由 **GP32C4T_CCVAL1** 寄存器决定，其中:

- $T_{延迟}$ = **GP32C4T_CCVAL1** 寄存器的值。
- $T_{脉冲}$ = **GP32C4T_AR** 与 **GP32C4T_CCVAL1** 之间的差决定。

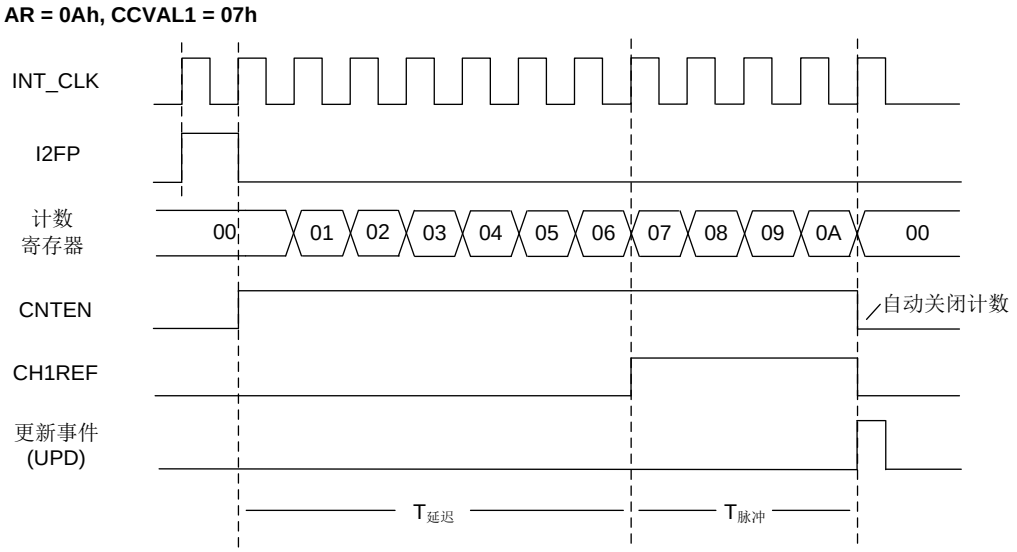


图 22-24　通道 1 触发模式下，基于 PWM 模式 2 单脉冲输出波形

CHn 快速开启模式

在单脉冲模式下，In 输入的有效边沿会自动开启计数器(硬件设置 CNTEN 位为 1)，并在 **GP32C4T_COUNT** 与 **GP32C4T_CCVALn** 比较后输出信号。然而在这个过程中需要数个时钟周期，将会延长 In 输入边沿与输出信号之间的延迟。

如果要使用最小延迟输出信号，可以设置 **GP32C4T_CHMR1** 寄存器的 CHnFEN 位为 1 开启快速开启模式。当侦测到 In 输入的有效边沿时，不再考虑比较值，强制让输出信号等效于匹配成功后的电平。此配置只在 PWM 模式 1 或 PWM 模式 2 时才能使用。

21. 5. 15 多次触发单脉冲模式

多次触发单脉冲模式下，当在一个计时周期中发生多次触发事件可以延长脉冲长度，与发单脉冲模式差别如下:

- 发生触发时，立即产生脉冲(无延迟)。
- 在前次触发脉冲结束前，又发生新的触发，则会延长脉冲长度。

以通道 1 多次触发单脉冲模式为例，相关配置如下:

1. 设置 **GP32C4T_SMCON** 寄存器中的 SMODS 位为 01000b，选择组合复位+触发模式。
2. 设置 **GP32C4T_CHMR1** 寄存器中的 CH1MOD 位为 1000b 或 1001b，选择多次触发单脉冲模式 1 或模式 2。

定时器配置递增计数模式时，对应的 CCVALn 必须配置 0，由 ARV 配置脉冲长度。

定时器配置递减计数模式时，对应的 CCVALn 必须大于或等于 ARV 数值。

注:这个模式不能和中心对齐的 PWM 模式一起使用，CMSEL 位需设置 0。

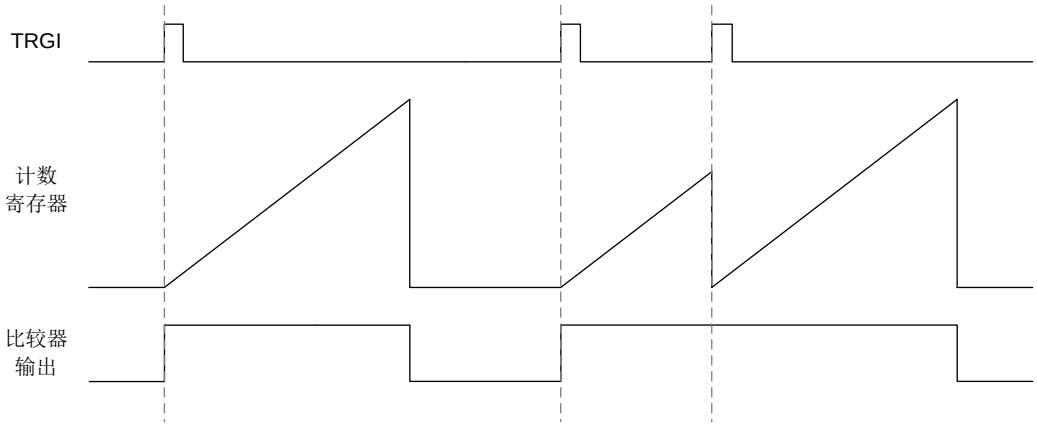


图 22-25　多次触发单脉冲模式

21. 5. 16 比较脉冲模式

此模式可以在比较器发生比较匹配时，产生一个脉冲宽度可编程的输出信号。只适用在通道 3 和通道 4，结构如下图:

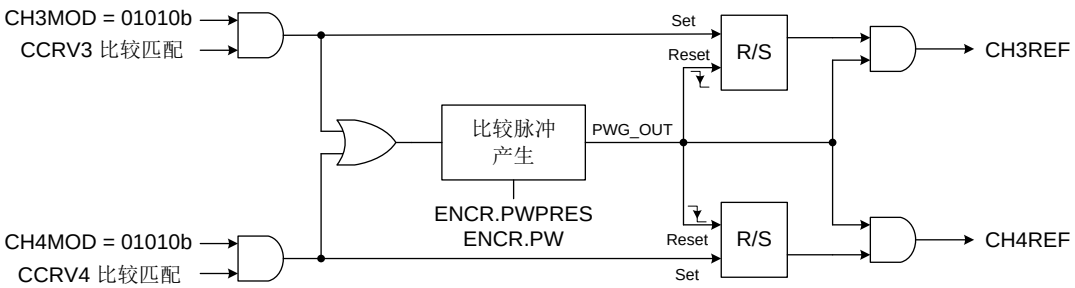


图 22-26　比较脉冲模式结构图

设置 **GP32C4T_CHMR2** 寄存器的 CH3MOD、CH4MOD 位为 01010b 可选择此模式。设置 **GP32C4T_ENCR** 寄存器的 PWPRES 和 PW 位，可编程脉冲宽度，公式如下:

- $t_{PWG} = (2^{(PWPRES[2:0])}) \times t_{INT\_CLK}$
- $t_{PW} = PW[7:0] \times t_{PWG}$

下图显示如何在边沿对齐模式下生成比较脉冲波型:

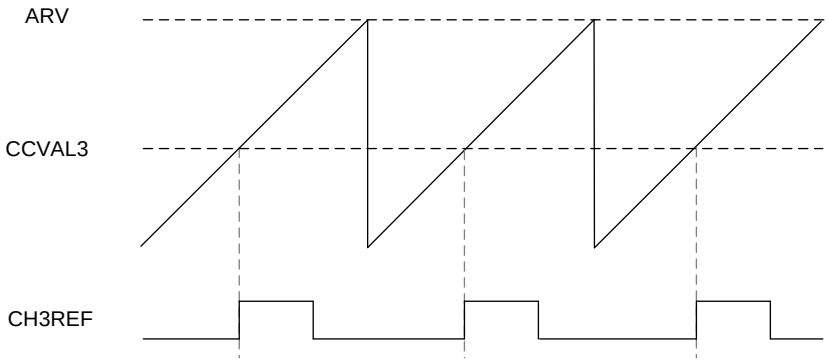


图 22-27　边沿对齐模式下的比较脉冲波型

比较脉冲是可以被重新触发的，一旦前次脉冲尚未结束前又发生新的触发，此时前次脉冲宽度持续时间将会被延长。

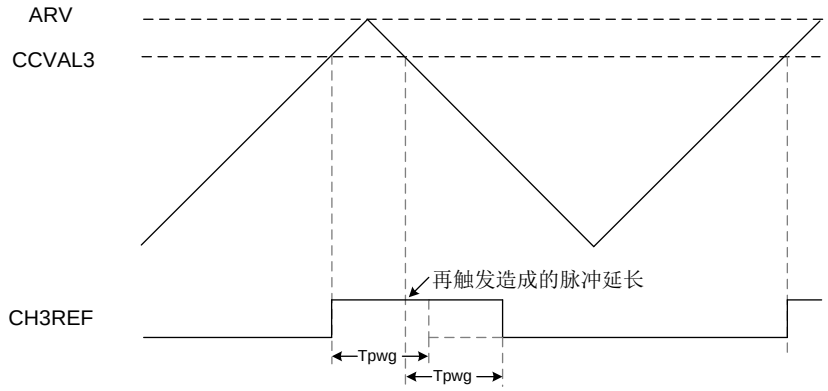


图 22-28　再触发延长脉冲宽度的比较脉冲波型

如果同时启用两个通道，其中一个通道上的比较脉冲尚未结束，但另一个通道又产生脉冲，则原通道的脉冲宽度持续时间将会被延长。

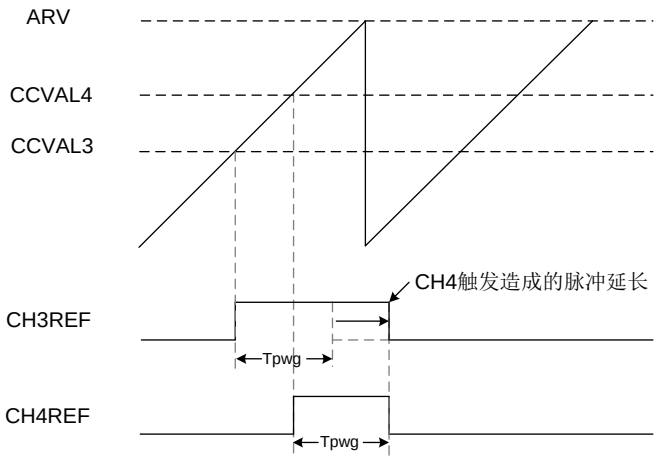


图 22-29　双通道延长脉冲宽度的比较脉冲波型

21. 5. 17 输出引脚控制

在输出模式(强制输出、输出比较或 PWM 模式)下，设置参考信号 CHnREF 映像到 CHn 输出的方式，可以通过配置 **GP32C4T_CCEP** 寄存器的 CCnEN 位来实现。而输出的极性可以由 **GP32C4T_CCEP** 寄存器的 CCnPOL 位决定。

如果关闭输出(CCnEN=0)，硬件会直接禁止 CHn 输出，使引脚为 Hi-Z 高阻态(不由定时器控制)。

输出引脚控制状态整理如下表:

控制位	输出状态
CCnEN	**CHn**
0	禁止引脚输出(Hi-Z，不由定时器控制)
1	CHn = CHnREF + 极性(CCnPOL)

表 22-8 输出引脚状态表

21. 5. 18 编码器接口

21. 5. 18. 1 正交编码器模式

正交编码器模式有以下五种计数方式:

- 正交编码器模式 1(x2)：设置 **GP32C4T_SMCON** 寄存器的 SMODS 位为 00001b，计数器根据 I2 电平在 I1 边沿递增或递减计数。
- 正交编码器模式 2(x2)：设置 **GP32C4T_SMCON** 寄存器的 SMODS 位为 00010b，计数器根据 I1 电平在 I2 边沿递增或递减计数。
- 正交编码器模式 3(x4)：设置 **GP32C4T_SMCON** 寄存器的 SMODS 位为 00011b，计数器根据 I2/I1 电平在 I1/I2 边沿递增或递减计数。
- 正交编码器模式 4(x1)：设置 **GP32C4T_SMCON** 寄存器的 SMODS 位为 01110b，计数器根据 I2 指定电平在 I1 边沿递增或递减计数。
- 正交编码器模式 5(x1)：设置 **GP32C4T_SMCON** 寄存器的 SMODS 位为 01111b，计数器根据 I1 指定电平在 I2 边沿递增或递减计数。

注:倍数是指在一个编码器输入信号的周期内，计数器可计算的次数。

可以通过设置 **GP32C4T_CCEP** 寄存器中的 CC1POL 和 CC2POL 位数值，选择正交编码器的输入 I1 和 I2 的极性(CC1NPOL 和 CC2NPOL 位必须置 0)。如果需要，也可以设置输入滤波器。

在正交编码器模式下，CH1 和 CH2 被用作输入接口。当计数器启动时(将 **GP32C4T_CON1** 寄存器的 CNTEN 位设置为 1)，计数器时钟由 I1 和 I2 经过滤波后的有效电平 I1FP 和 I2FP 转换提供。I1FP 和 I2FP 转换序列会产生计数脉冲和方向信号。计数器的递增或递减由信号的转换序列决定，而 **GP32C4T_CON1** 寄存器的 DIRSEL 方向位则由硬件自动更新。

正交编码器模式的工作方式类似于带有方向选择的外部时钟。计数器会在 0 到 **GP32C4T_AR** 寄存器中的自动重载值之间进行连续计数。因此，在开始计数之前需要设置 **GP32C4T_AR** 寄存器的值。在这种模式下，捕获器、预分频器和触发输出的功能都可以正常工作。需要注意的

是，编码模式和选择外部时钟源 2 不兼容，因此不能同时选择。

此模式下，计数器会根据正交编码器的速度和方向自动修改，计数器的数值反映的是编码器的位置。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):

正交编码器模式	SMODS	有效边沿	有效边沿的相对信号电平(*注)	I1FP 上升沿	I1FP 下降沿	I2FP 上升沿	I2FP 下降沿
模式 1(x2)	00001b	仅在 I1 计数	高电平	递减	递增	不计数	不计数
			低电平	递增	递减	不计数	不计数
模式 2(x2)	00010b	仅在 I2 计数	高电平	不计数	不计数	递增	递减
			低电平	不计数	不计数	递减	递增
模式 3(x4)	00011b	在 I1 和 I2 上计数	高电平	递减	递增	递增	递减
			低电平	递增	递减	递减	递增
模式 4(x1)	01110b	仅在 I1 计数	高电平	递减	递增	不计数	不计数
			低电平	不计数	不计数	不计数	不计数
模式 5(x1)	01111b	仅在 I2 计数	高电平	不计数	不计数	递增	递减
			低电平	不计数	不计数	不计数	不计数

表 22-9　计数方向与编码器输入信号的关系(CC1POL = CC2POL = 0)

注:

有效边沿的相对信号电平指的是:

在 I1FP 上升沿或下降沿计数时，计数方向由 I2FP 的电平高或低决定。

在 I2FP 上升沿或下降沿计数时，计数方向由 I1FP 的电平高或低决定。

正交编码器可直接与 MCU 连接，但一般会通过比较器将正交编码器的差分输出转换为数字信号。下图给出了计数信号产生和方向控制的例子:

配置如下:

1. 设置 **GP32C4T_CHMR1** 寄存器的 CC1SSEL 位为 01b，选择通道为 I1 输入。
2. 设置 **GP32C4T_CHMR1** 寄存器的 CC2SSEL 位为 01b，选择通道为 I2 输入。
3. 设置 **GP32C4T_CCEP** 寄存器的 CC1POL 位为 0、CC1NPOL 为 0，选择非反相输入。
4. 设置 **GP32C4T_CCEP** 寄存器的 CC2POL 位为 0、CC2NPOL 为 0，选择非反相输入。
5. 设置 **GP32C4T_SMCON** 寄存器的 SMODS 位为 00011b，选择计数器同时根据 I1 和 I2 上的边沿计数。
6. 设置 **GP32C4T_CON1** 寄存器 CNTEN 位为 1 开启计数器。

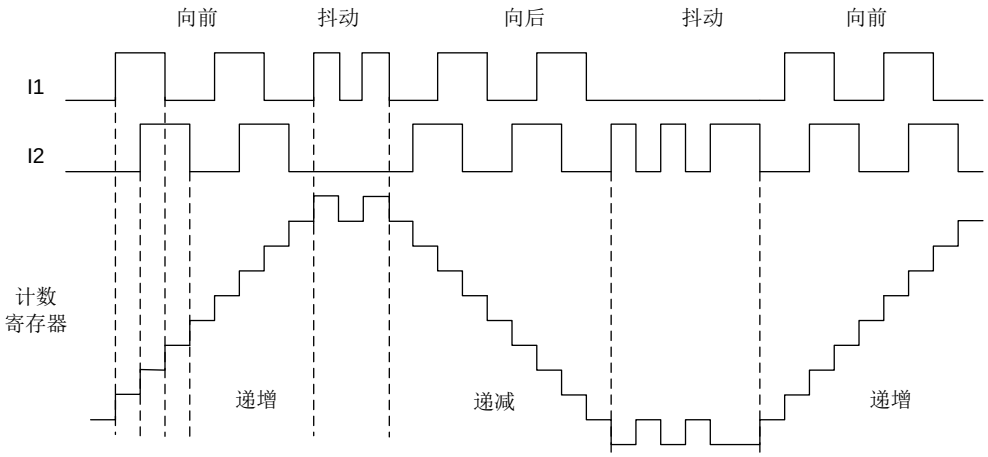


图 22-30　编码器接口模式下的计数操作

下图给出了计数器在 I1 滤波信号极性反相时的计数过程(除了设置 CC1POL 位为 1，其他设置与上面一致):

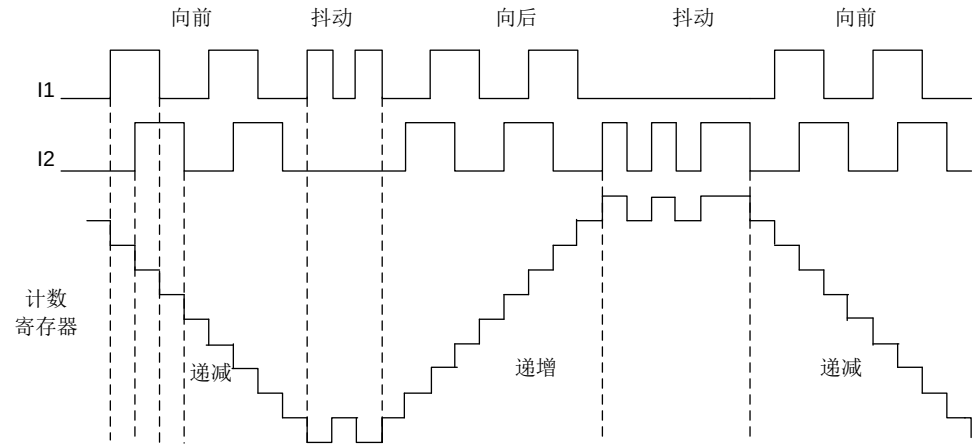


图 22-31　滤波后极性反相时编码器接口例子

下图为不同正交编码器模式下的计数方式(AR = 08h):

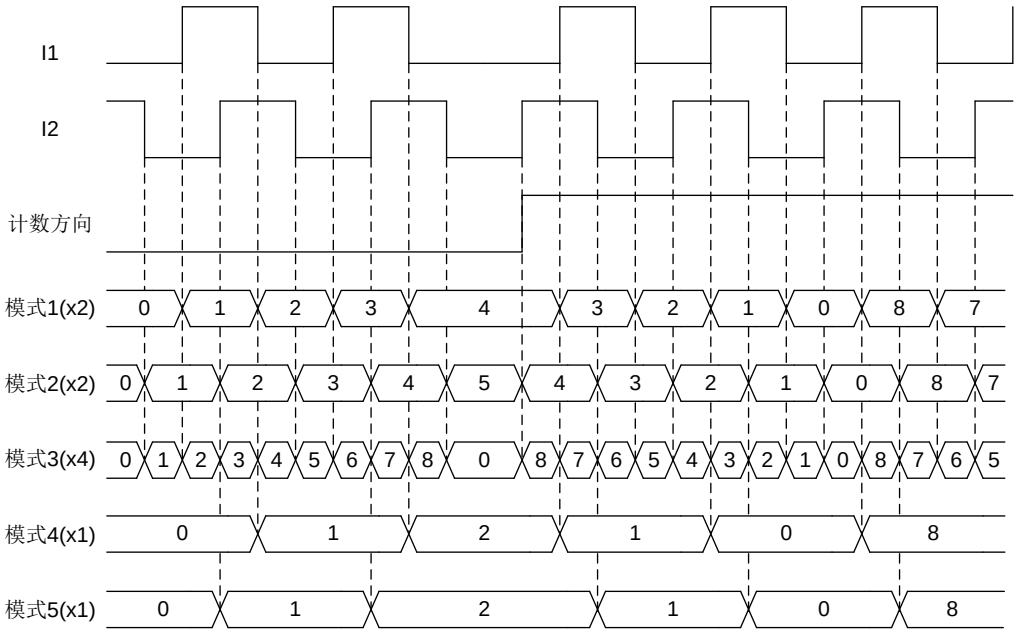


图 22-32　不同正交编码器模式下的计数方式

除了正交编码器模式外，定时器还提供了以下两种编码器模式:时钟加方向编码器模式和定向时钟编码器模式。

21. 5. 18. 2　时钟加方向编码器模式

在此模式下，定时器计数方向(递增/递减)由 I1 的电位控制，而计数时钟由 I2 的上升沿/下降沿控制，整理如下:

CC1POL	I1 电平	计数方向
0	高电平	递增
	低电平	递减
1	高电平	递减
	低电平	递增

表 22-10　计数方向与极性及 I1 电平之间的关系

时钟加方向编码器模式	SMODS	CC2POL	I2 上升沿	I2 下降沿
模式 1(x2)	01010b	x	计数	计数
模式 2(x1)	01011b	0	计数	不计数
		1	不计数	计数

表 22-11　计数时钟与极性及 I2 边沿之间的关系

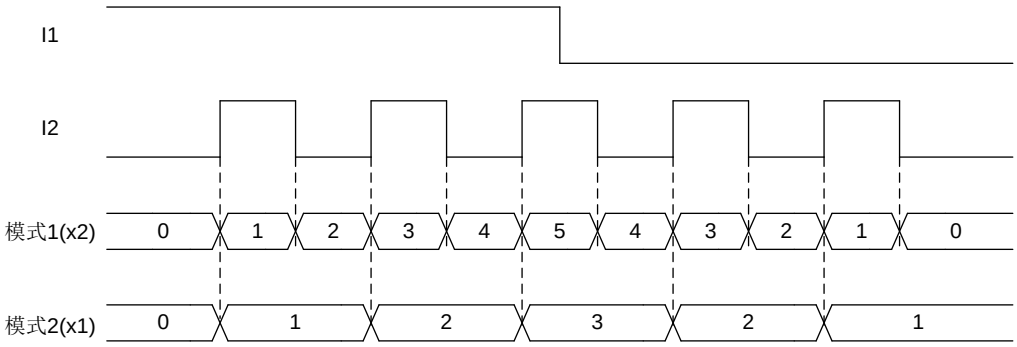


图 22-33　时钟加方向编码器模式(CC1POL = 0, CC2POL = 0)

21. 5. 18. 3　定向时钟编码器模式

与时钟加方向编码器模式不同的是，此模式下 I1/I2 皆用来控制定时器计数，I1 上升沿或下降递减计数，I2 上升沿或下降沿递增计数，整理如下:

定向時鐘編碼器模式	SMODS	CCnPOL	有效边沿的相对信号电平	I1FP 上升沿	I1FP 下降沿	I2FP 上升沿	I2FP 下降沿
模式 1(x2)	01100b	0	高电平	递减	递减	递增	递增
			低电平	不计数	不计数	不计数	不计数
		1	高电平	不计数	不计数	不计数	不计数
			低电平	递减	递减	递增	递增
模式 2(x1)	01101b	0	高电平	不计数	递减	不计数	递增
			低电平	不计数	不计数	不计数	不计数
		1	高电平	不计数	不计数	不计数	不计数
			低电平	递减	不计数	递增	不计数

表 22-12　计数方向与编码器输入信号的关系

CC1POL = 0, CC2POL = 0

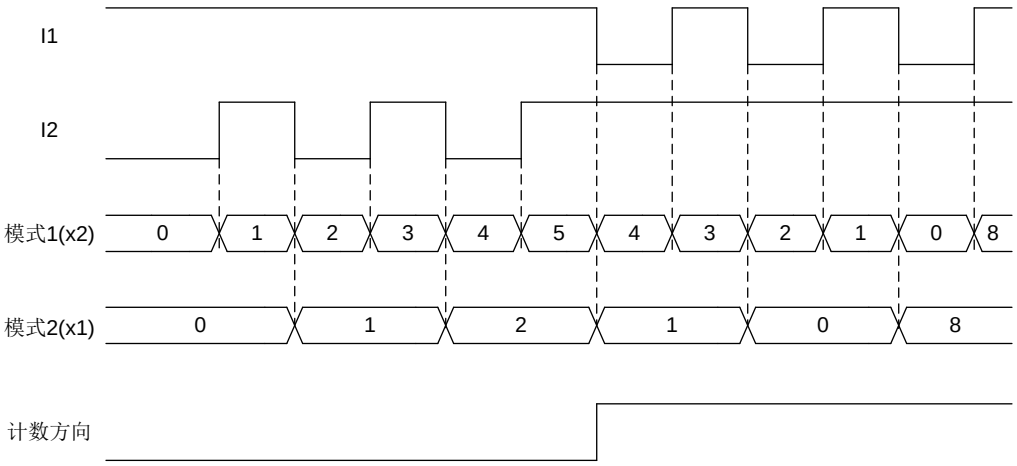


图 22-34　定向时钟编码器模式(CC1POL = 0, CC2POL = 0)

CC1POL = 1, CC2POL = 1

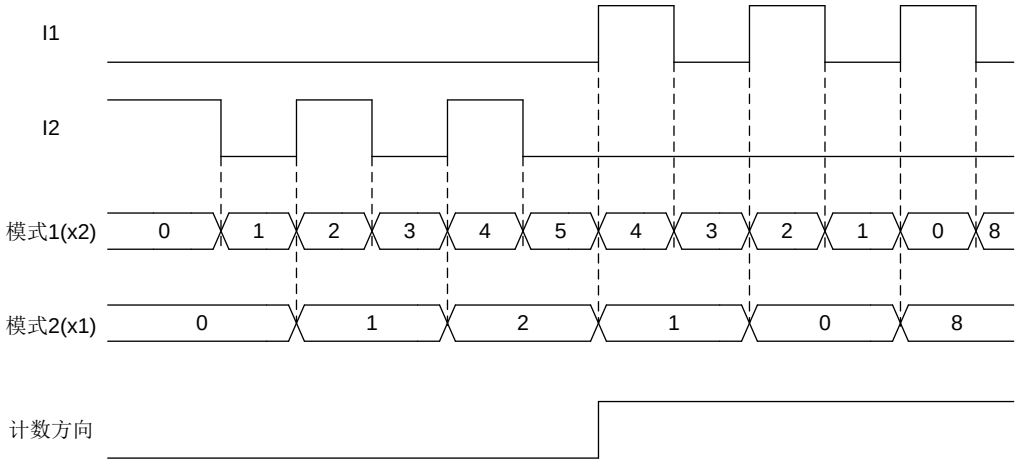


图 22-35　定向时钟编码器模式(CC1POL = 1, CC2POL = 1)

21. 5. 18. 4　正交编码器模式下的索引(Index)输入

编码器的第三个输出(Z 相/Index)用于指示机械零点，可以连接到 ETR 引脚，用来触发计数器复位。

设置 **GP32C4T_ENCR** 寄存器的 IDXEN 位为 1，开启索引侦测的功能。此功能只适用在 **GP32C4T_SMCON** 寄存器的 SMODS 设定为编码器模式(0001b、0010b、0011b、1010b、1011b、1100b、1101b、1110b、1111b)时。

大部分的编码器都具备有调整索引脉冲宽度的选项，主要有以下三种模式:

- AB 相门限模式: 索引脉冲在 A、B 相的两个有效边沿对齐，宽度为 1/4 编码器输入周期。
- A 相(或 B 相)门限模式: 索引脉冲在 A(或 B 相)的两个有效边沿对齐，宽度为 1/2 编码器输入周期。
- 非门限模式: 索引脉冲不与 A、B 相的两个有效边沿对齐，宽度为 1 个编码器输入周期。

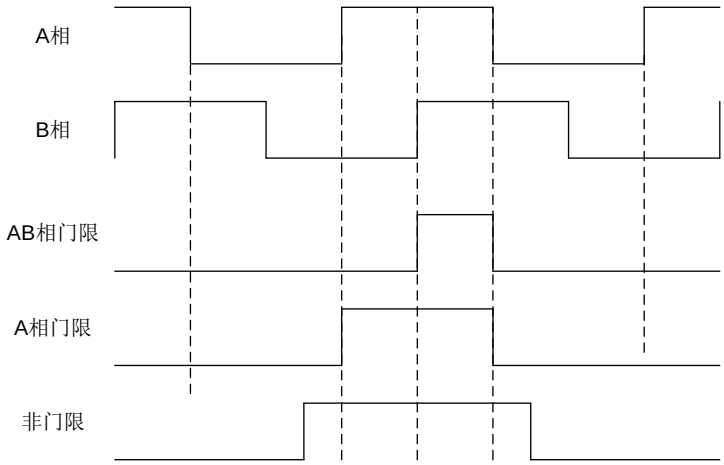


图 22-36　索引脉冲宽度

索引脉冲的抖动是可以被允许的，但抖动范围不能超过两个编码器输入周期，否则会造成计数器多次重置。

通过 **GP32C4T_ENCR** 寄存器的 IDXPOS 位，可以定义索引信号在哪个 AB 相位为有效输入，因其物理特性的关系，索引只会永远出现在其中一种相位上(假设使用的是 AB 相门限)。

下图为编码器正转时的 AB 相变化，与索引出现在不同相位时，所需要对应的 IDXPOS 位设置:

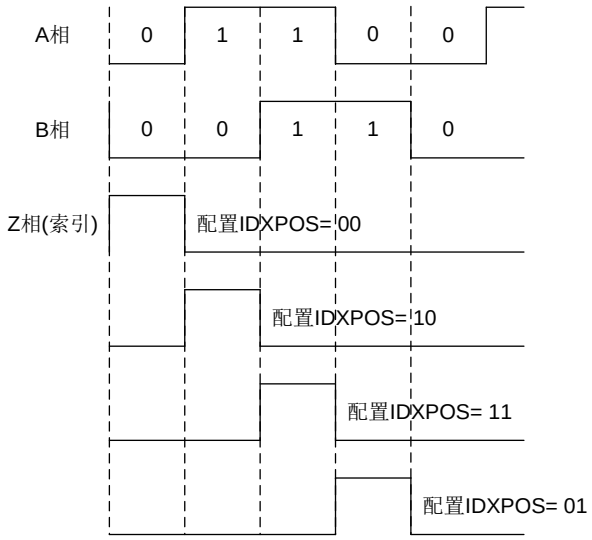


图 22-37　有效索引与 IDXPOS 位之关系图

下图为 AB 相门限索引(IDXPOS 位设定为 11b)有效时，计数器复位情形:

- 递增计数下 AB 相为 11 且侦测到有效索引时，重置计数器为 0，并产生索引中断(IDX)(若有开启 **GP32C4T_IER** 寄存器的 IDX 位)。
- 递减计数下 AB 相为 11 且侦测到有效索引时，在 AB 相不为 11 时，重置计数器为 AR，并产生索引中断(IDX)(若有开启 **GP32C4T_IER** 寄存器的 IDX 位)。

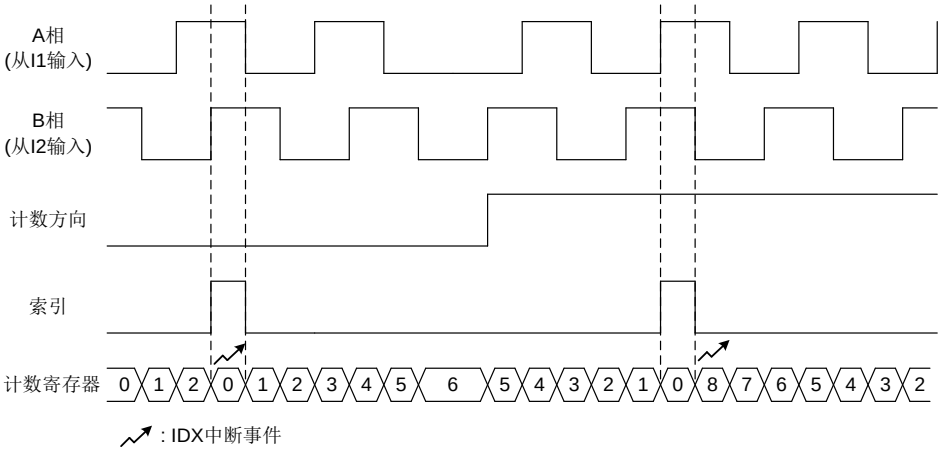


图 22-38　计数器数值与 AB 相门限索引(IDXPOS = 11b)

下图为非相门限索引(IDXPOS 位设定为 00b)有效时，计数器复位情形:

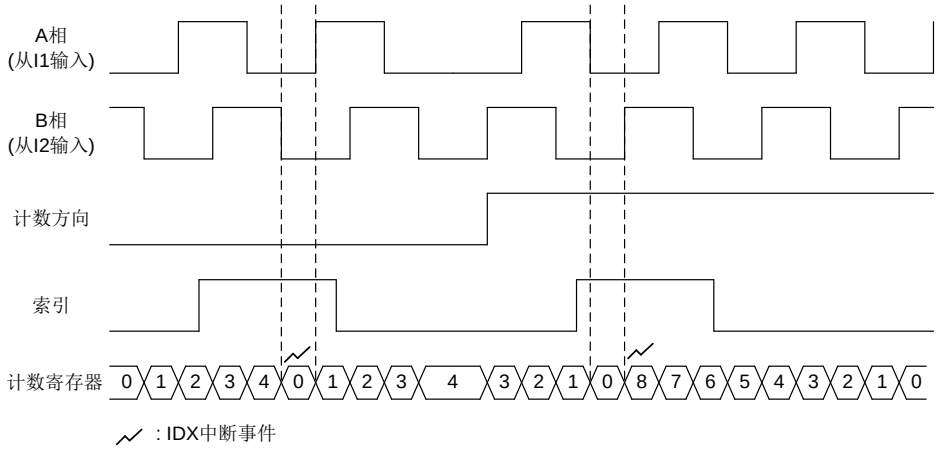


图 22-39　计数器数值与非门限索引(IDXPOS = 00b)

下图为索引因抖动造成脉冲宽度小于 AB 相门限时的波型图 ：

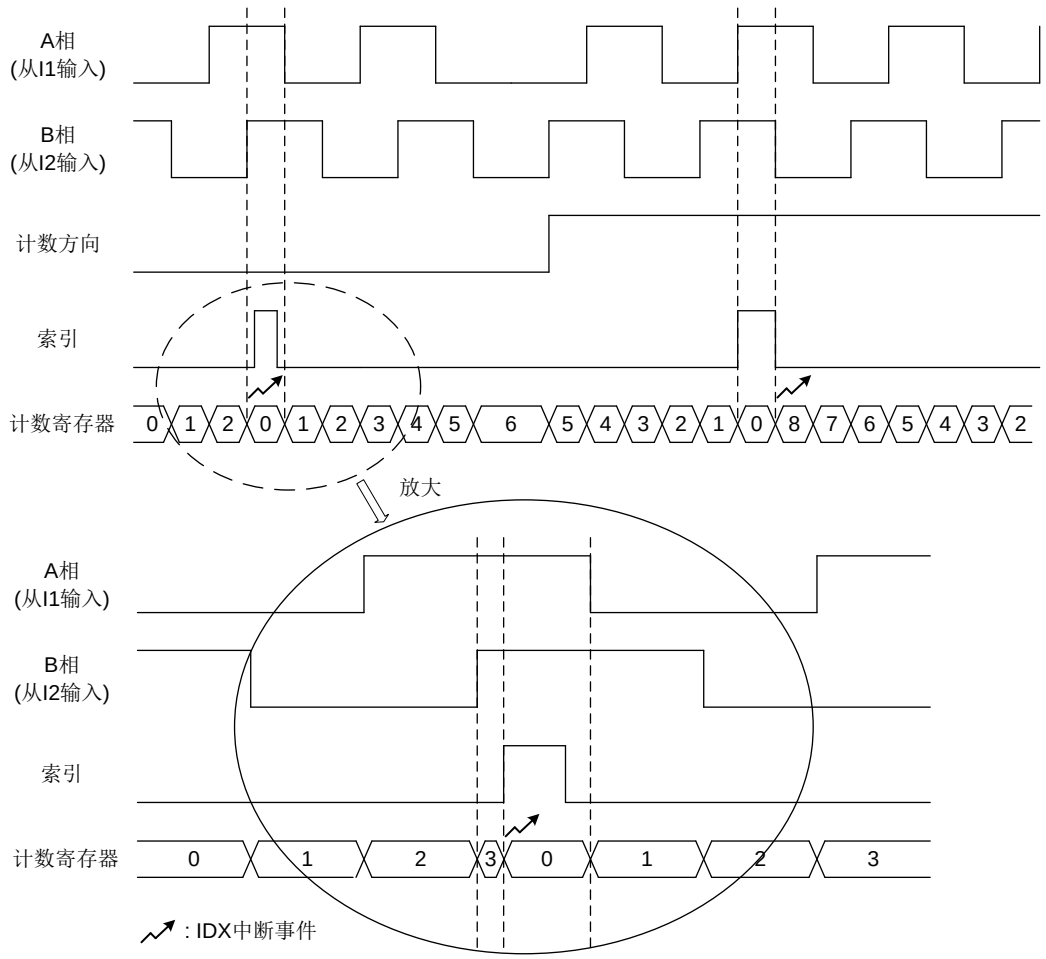


图 22-40　索引脉冲宽度小于 AB 相门限(IDXPOS = 11b)

下图为 B 相门限索引(IDXPOS 位设定为 01b)有效时，在 x2 模式与 x1 模式下计数器复位情形:

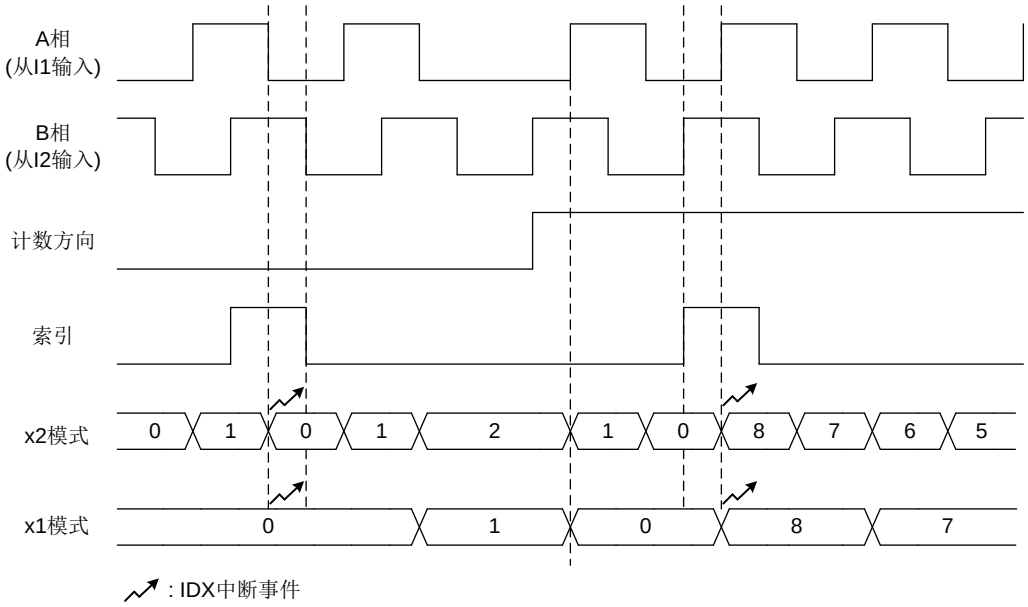


图 22-41　计数器数值与 B 相门限索引(IDXPOS = 01b)

特定计数方向下检测索引有效事件

设置 **GP32C4T_ENCR** 寄存器的 IDXDIR 位，可以在特定计数方向下检测索引有效事件。下图为 IDXDIR 位与有效索引事件在不同计数方向下的关系图:

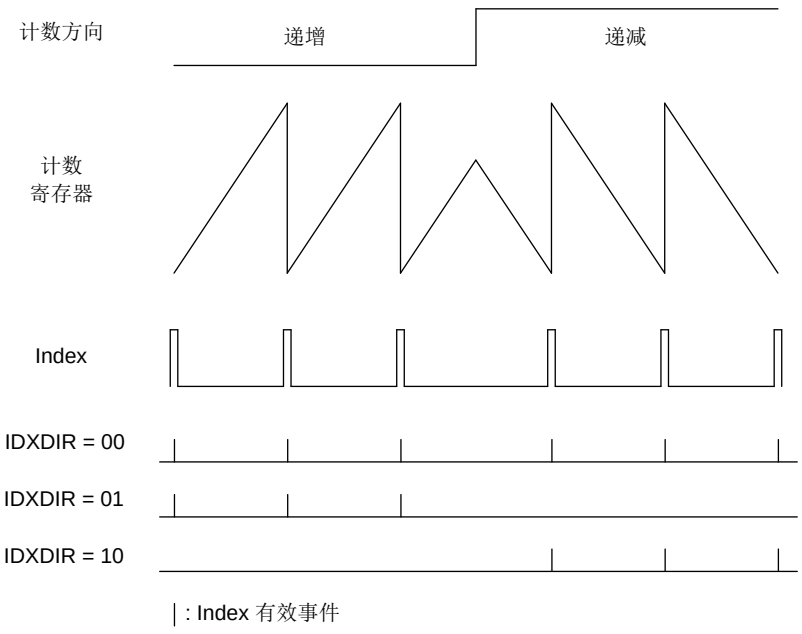


图 22-42　特定计数方向下检测索引有效事件

注:

1. 必须在索引检测功能关闭(GP32C4T_ENCR 寄存器的 IDXEN 位为 0)时，才能改变 IDXDIR 位。

2. 此功能在时钟加方向编码器模式下不支持，IDXDIR 必须设置为 0。

首次索引事件检测

设置 **GP32C4T_ENCR** 寄存器的 FIDX 位为 1，可以只检测第一次的索引为有效事件，让后续的索引皆为无效事件。若有需要，可以将 FIDX 位写 0 再写 1，重启首次索引事件检测机制。

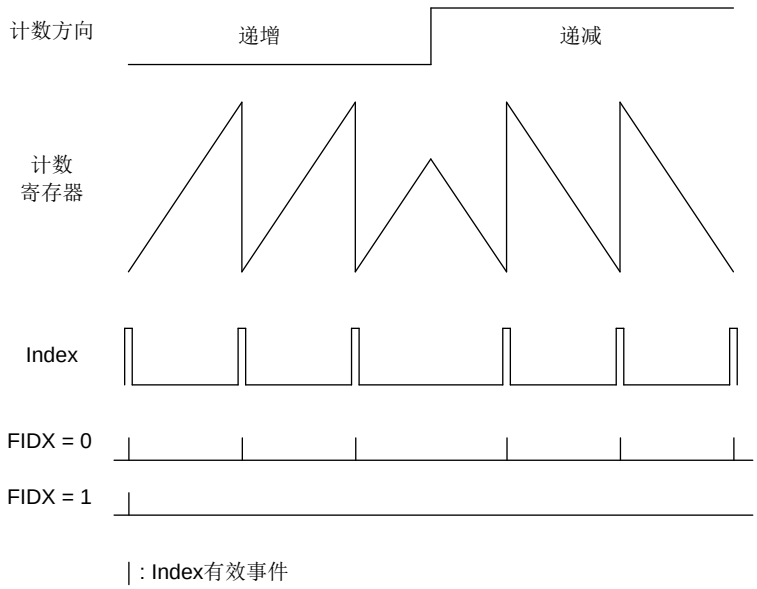


图 22-43 首次索引事件检测

21. 5. 18. 5 非正交编码器模式下的索引输入

设定成时钟加方向编码器模式(SMODS 位=01010b/01011b)或定向时钟编码器模式(SMODS 位=01100b/01101b)时，IDXPOS[1]位无效:

- IDXPOS[0] = 0：当时钟低电平时，有效的索引事件会重置计数器。
- IDXPOS[0] = 1：当时钟高电平时，有效的索引事件会重置计数器。

下图为时钟加方向编码器模式在有效索引事件发生时，计数器复位情形:

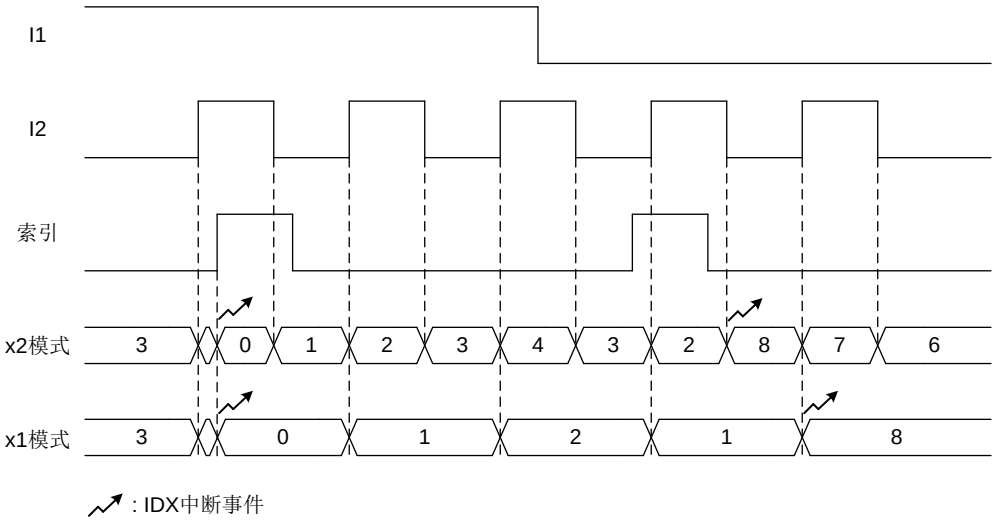


图 22-44　时钟加方向编码器模式下检测索引有效事件(IDXPOS[0] = 1)

下图为定向时钟编码器模式在有效索引事件发生时，计数器复位情形:

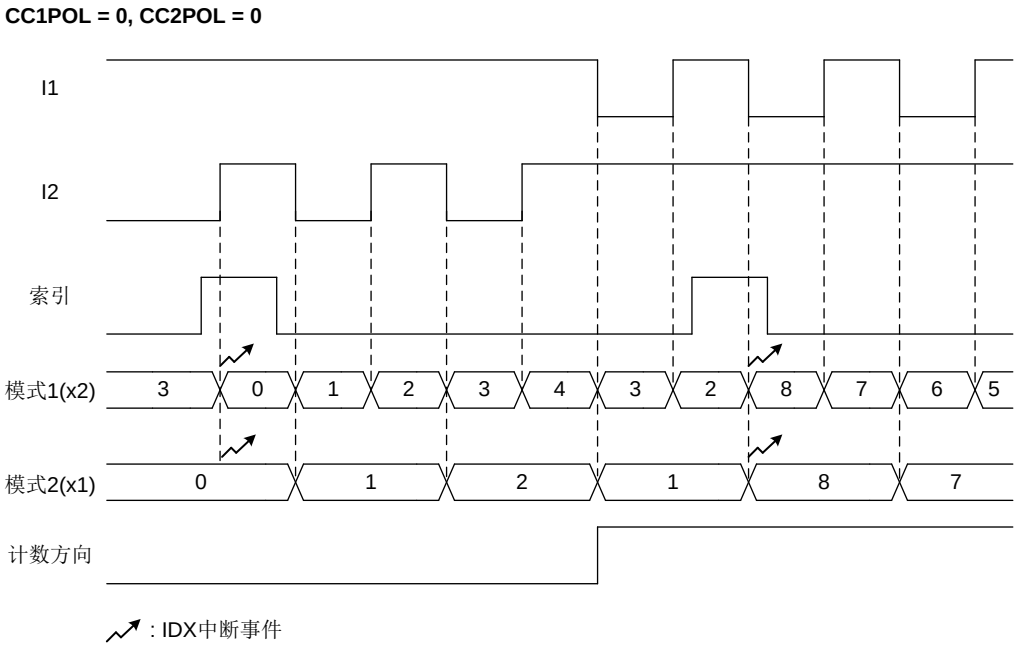


图 22-45　定向时钟编码器模式下检测索引有效事件(IDXPOS[0] = 1)

21. 5. 18. 6　运行模式下切换从模式选择

当用户在定时器运行中要更改编码器模式时，建议设置**GP32C4T_SMCON**寄存器的SMODPE位为 1，开启 SMODS 的预装载功能，并设置 SMODPS 位决定缓冲寄存器的更新时机:

- 设置 SMODPS 位为 0: 发生更新事件(UPD)时，将预装载寄存器内的值更新到缓冲寄存器中。
- 设置 SMODPS 位为 1: 发生索引事件(IDX)时，将预装载寄存器内的值更新到缓冲寄存器中。

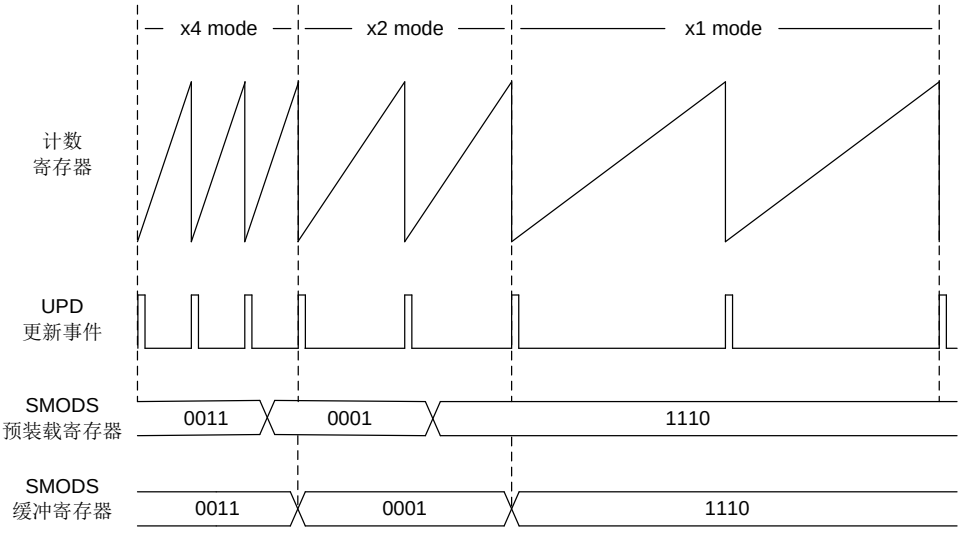


图 22-46　运行模式下切换从模式选择(SMODPS = 0)

21. 5. 18. 7 编码器时钟输出

编码器模式的工作原理不太适合用来做速度量测，因为要比较长的时间来获取足够多的时钟周期。比较好的应用方式是通过从模式控制器，将编码器的时钟输出到从定时器，再由从定时器以检测边沿到边沿的方式，执行周期量测并计算间隔以提供速度等相关信习。

设置 **GP32C4T_CON2** 寄存器的 MMSEL 位为 1000b 时，可以将编码器时钟输出到 TRGOUT。此模式只适用在 **GP32C4T_SMCON** 寄存器的 SMODS 设定为编码器模式(0001b、0010b、0011b、1010b、1011b、1100b、1101b、1110b、1111b)时。

21. 5. 19 方向位输出

设置 **GP32C4T_CHMR2** 寄存器的 CH3MOD/CH4MOD 位为 01011b，可将 **GP32C4T_CON1** 寄存器的 DIRSEL 位映射到 CH3/CH4 通道输出，主要用来监测中心对齐计数时的方向，或编码器模式下的方向变换。

21. 5. 20 UPD位重映射

通过设置 **GP32C4T_CON1** 寄存器的 UPDREMAP 位，可以将 UPD 原始中断状态映射至 **GP32C4T_COUNT** 寄存器的第 31 位 UPDRIF 中。

注:UPD 和 UPDRIF 标志之间没有延迟。

21. 5. 21 输入XOR功能

通过 **GP32C4T_CON2** 寄存器的 I1SEL 位，可将通道 1 的输入滤波器连接到 XOR 门的输出端，XOR 门输入端包含 CH1、CH2 和 CH3 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。

21. 5. 22 霍尔传感器界面

本文介绍使用 AD16C6T 定时器产生 PWM 信号驱动马达，以及用另一个通用定时器(GP32C4T)作为"接口定时器"来连接霍尔传感器的方法。具体操作是将 3 个定时器输入脚(CH1、CH2 和 CH3)通过 XOR 门连接到 I1 输入通道，并由接口定时器捕获这个信号。为了产生一个由霍尔输入端的任何变化而触发的时间基准，模式控制器被设置为复位模式，输入为 I1F 双边沿。每当 3 个输入之一变化时，计数器就会从 0 重新开始计数。

在接口定时器模式下，通道 1 被设置为捕获模式，捕获信号源为 I1。捕获值反映了输入端两次变化之间的时间间隔，提供与马达转速相关的讯息。接口定时器还可以在输出模式产生一个脉冲，这个脉冲通过 TRGOUT 输出到 AD16C6T 定时器，用来生成 COM 事件，改变 AD16C6T 定时器各个通道的配置，进而产生 PWM 信号驱动马达。

以下图为例:

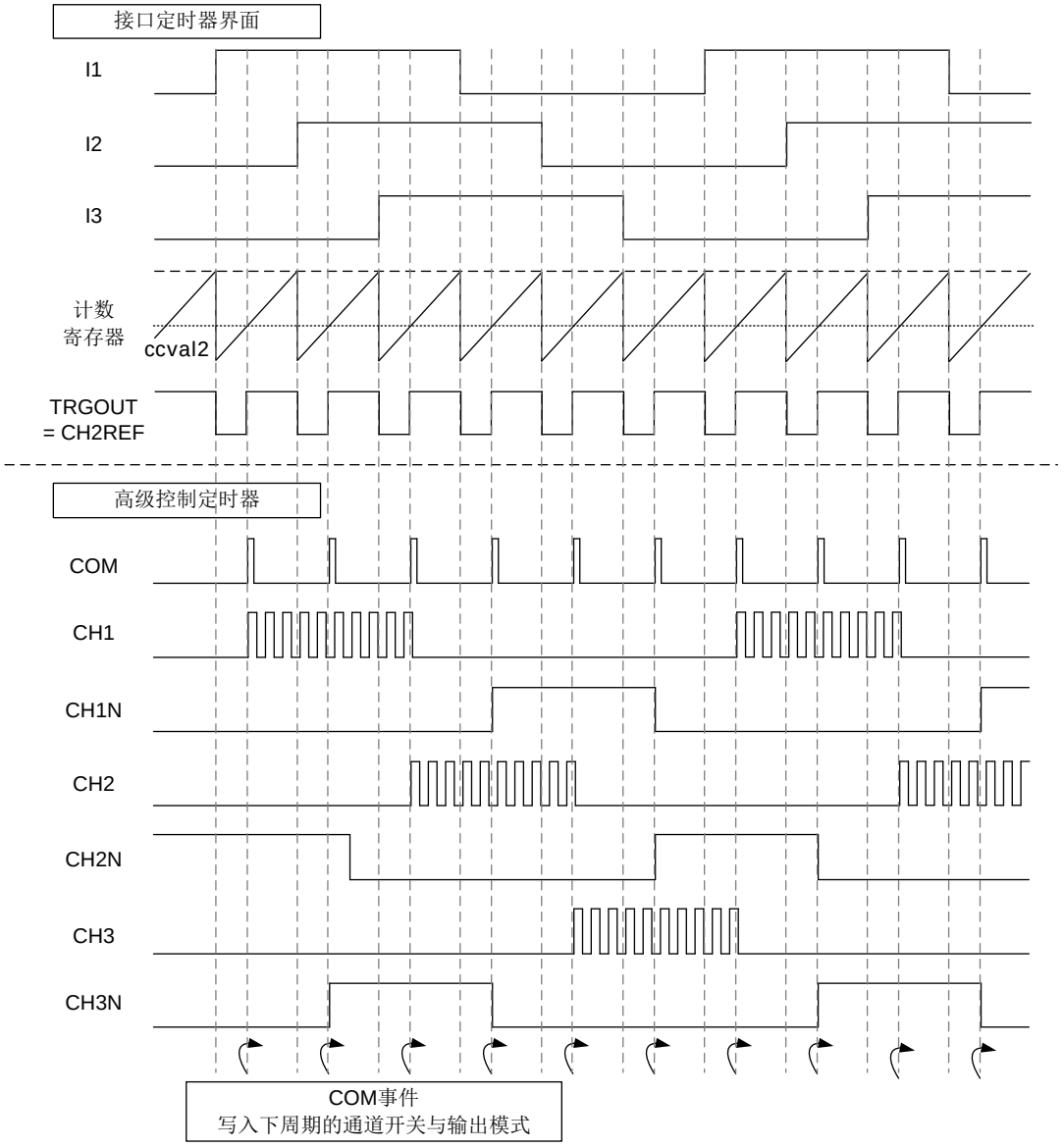


图 22-47　霍尔传感器接口范例

- 设置 **GP32C4T_CON2** 寄存器的 I1SEL 位为 1，通过将三个定时器输入进行 XOR 运算后输入 I1 通道。
- 设置 **GP32C4T_AR** 为其最大值，以便计数器在 I1 变化时被清零。同时，设置预分频器以得到一个计数器周期，该周期长于传感器两次变化的时间间隔。
- 设置 **GP32C4T_CHMR1** 寄存器的 CC1SSEL 位为 01b，选择 I1 作为捕获输入通道。如果需要，还可以设置数字滤波器。
- 设置 **GP32C4T_CHMR1** 寄存器的 CC2SEL 位为 00b，选择通道 2 作为输出通道。
- 设置 **GP32C4T_CHMR1** 寄存器的 CH2MOD 位为 0111b，设置通道 2 为 PWM 模式 2。
- 设置 **GP32C4T_CCVAL2** 寄存器决定 PWM 输出延迟时间。
- 设置 **GP32C4T_CON2** 寄存器的 MMSEL 位为 101b，选择 CH2REF 作为 TRGOUT 上的触发输出。

在 AD16C6T 定时器中，TRGI 需选择 ITn 做为触发事件源，定时器被配置为产生 PWM 信号，

并开启捕获或比较预装载控制(设置 AD16C6T_CON2 寄存器的 CCPCEN 位为 1)，COM 事件由触发输入控制(设置 AD16C6T_CON2 寄存器的 CCUSEL 位为 1)。发生 COM 事件后(TRGI 侦测到上升沿)，在 PWM 控制位(CCnEN、CHnMOD)中写入下一步的配置，此操作可在由 CH2REF 上升沿产生的中断子程序中完成。

21. 5. 23 外部触发的同步

GP32C4T 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。

21. 5. 23. 1 复位模式

当侦测到有效触发输入事件时，计数器及其预分频器会被重新初始化，若此时 **GP32C4T_CON1** 寄存器的 UERSEL 位为 0，则一并产生更新事件 UPD。而所有预装载寄存器(**GP32C4T_AR**，**GP32C4T_CCVALn**)都会因更新事件 UPD 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零，配置过程如下:

1. 设置 **GP32C4T_CHMR1** 寄存器的 CC1SSEL 位为 01b，选择 I1 为有效输入端。
2. 设置 **GP32C4T_CHMR1** 寄存器的 I1FLT 位为 0000b，本例无需滤波器。
3. 设置 **GP32C4T_CCEP** 寄存器的 CC1NPOL 位为 0、CC1POL 位为 0，选择 I1 通道上升沿有效。
4. 设置 **GP32C4T_CHMR1** 寄存器的 I1PRES 位为 00b，捕获预分频器不用于触发操作，无需设置。
5. 设置 **GP32C4T_SMCON** 寄存器的 TSSEL 位为 00101b，选择 I1 滤波后信号作为输入源。
6. 设置 **GP32C4T_SMCON** 寄存器的 SMODS 位为 0100b，选择复位模式。
7. 设置 **GP32C4T_CON1** 寄存器的 CNTEN 位为，开启计数器。

计数器依据内部时钟开始计数，计数器计数直到 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。同时设置标志位为 1(**GP32C4T_RIF** 寄存器的 TRGI 位)，如果中断及 DMA 开启(取决于 **GP32C4T_IER** 寄存器的 TRGI 位和 **GP32C4T_DMAEN** 寄存器的 TRGI 位)，会发送中断及 DMA 请求。

下图给出了设置自动重载寄存器 **GP32C4T_AR** 为 36h 时的信号变化。由于 I1 输入的同步电路，I1 上的上升沿和计数器实际初始化之间会存在延时。

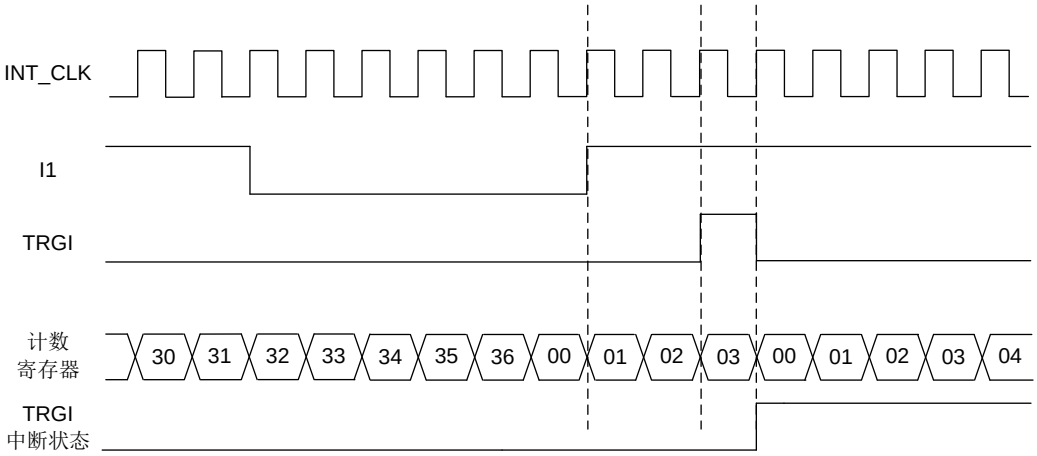


图 22-48　复位模式控制电路

21. 5. 23. 2　门控模式

计数器根据选中的输入电平被开启。

下面的例子中，计数器只在 I1 输入为低电平时才递增计数:

1. 设置 **GP32C4T_CHMR1** 寄存器的 CC1SSEL 位为 01b，选择 I1 为有效输入端。
2. 设置 **GP32C4T_CHMR1** 寄存器的 I1FLT 位为 0000b，本例无需滤波器。
3. 设置 **GP32C4T_CCEP** 寄存器的 CC1NPOL 位为 0、CC1POL 位为 1，I1 通道反相，有效极性为低电平。
4. 设置 **GP32C4T_CHMR1** 寄存器的 I1PRES 位为 00b，捕获预分频器不用于触发操作，无需设置。
5. 设置 **GP32C4T_SMCON** 寄存器的 TSSEL1 位与 TSSEL2 位为 00101b，选择 I1 滤波后信号作为输入源。
6. 设置 **GP32C4T_SMCON** 寄存器的 SMODS 位为 101b，选择门控模式。
7. 设置 **GP32C4T_CON1** 寄存器的 CNTEN 位为 1，开启计数器(在门控模式中，如果 CNTEN 位为 0，无论触发输入为何电平，计数器都不会开启)。

只要 I1 为低电平，计数器依据内部时钟开始计数，一旦 I1 为高电平则停止计数。由于 I1 输入端的同步电路的原因，I1 上出现上升沿和计数器实际停止之间会有一定的延迟。

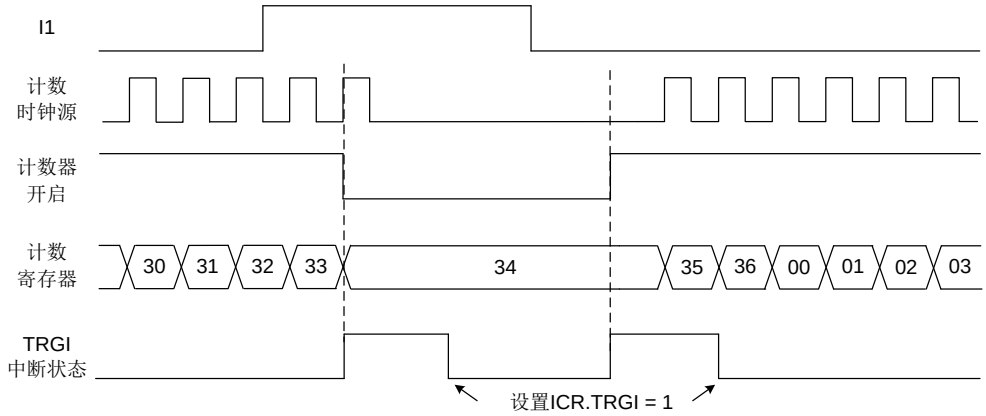


图 22-49　门控模式控制电路

21. 5. 23. 3　触发模式

输入端选中的事件可以开启计数器。

下面的例子中，I2 输入端上的上升沿可以开启递增计数:

1. 设置 **GP32C4T_CHMR1** 寄存器的 CC2SSEL 位为 01b，选择 I2 为有效输入端。
2. 设置 **GP32C4T_CHMR1** 寄存器的 I2FLT 位为 0000b，本例无需滤波器。
3. 设置 **GP32C4T_CCEP** 寄存器的 CC2NPOL 位为 0、CC2POL 位为 0，选择 I2 通道上升沿有效。
4. 设置 **GP32C4T_CHMR1** 寄存器的 I2PRES 位为 00b，捕获预分频器不用于触发操作，无需设置。
5. 设置 **GP32C4T_SMCON** 寄存器的 TSSEL1 位与 TSSEL2 位为 00110b，选择 I2 滤波后信号作为输入源。
6. 设置 **GP32C4T_SMCON** 寄存器的 SMODS 位为 110b，选择触发模式。
7. 设置 **GP32C4T_CON1** 寄存器的 CNTEN 位为，开启计数器。

I2 上出现上升沿时，计数器开始依据内部时钟计数并設置 TRGI 标志位为 1。

由于 I2 输入的同步电路原因，I2 上出现上升沿和计数器实际启动之间会有一定的延迟。

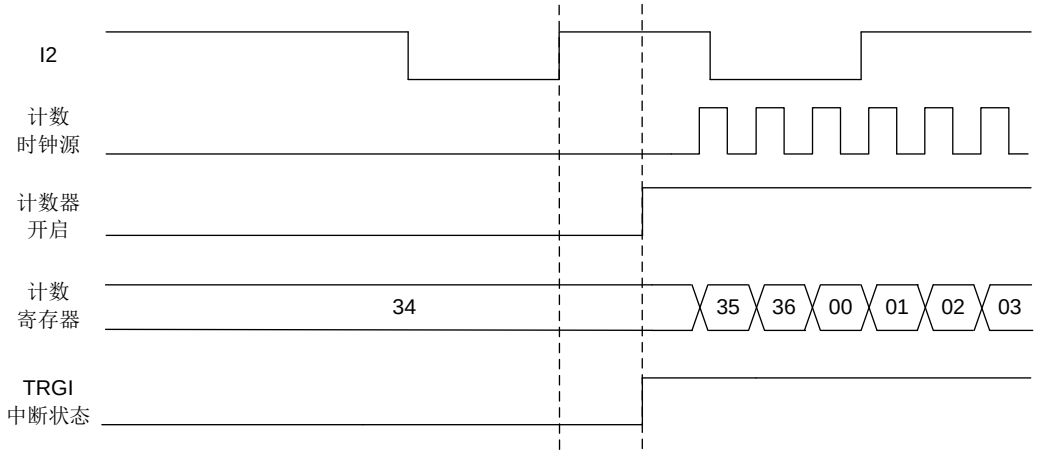


图 22-50　触发模式控制电路

21. 5. 23. 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(外部时钟模式 1 和除编码模式除外)。ETR 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)。不推荐设置 **GP32C4T_SMCON** 寄存器的 TSSEL1 位与 TSSEL2 位为 00111b 选择 ETR 作为 TRGI。

下面的例子中，当 I1 输入端出现上升沿时，开启计数器，并且在每个 ETR 信号的上升沿递增计数。

ETR 外部触发电路设定如下 ：

1. 设置 **GP32C4T_SMCON** 寄存器的 ETFLT 位为 000b，无需滤波器。
2. 设置 **GP32C4T_SMCON** 寄存器的 ETPRES 位为 00b，关闭预分频。
3. 设置 **GP32C4T_SMCON** 寄存器的 ETPOL 位为 0，ETR 的上升沿有效。
4. 设置 **GP32C4T_SMCON** 寄存器的 ECM2EN 位为 1，开启外部时钟模式 2。

通道 1 检测 I1 的上升沿，过程如下:

1. 设置 **GP32C4T_CHMR1** 寄存器的 CC1SSEL 位为 01b，选择 I1 为有效输入端。
2. 设置 **GP32C4T_CHMR1** 寄存器的 I1FLT 位为 0000b，本例无需滤波器。
3. 设置 **GP32C4T_CCEP** 寄存器的 CC1NPOL 位为 0、CC1POL 位为 0，选择 I1 通道上升沿有效。
4. 设置 **GP32C4T_CHMR1** 寄存器的 I1PRES 位为 00b，捕获预分频器不用于触发操作，无需设置。
5. 设置 **GP32C4T_SMCON** 寄存器的 TSSEL1 位与 TSSEL2 位为 00101b，选择 I1 滤波后信号作为输入源。
6. 设置 **GP32C4T_SMCON** 寄存器的 SMODS 位为 110b，选择触发模式。
7. 设置 **GP32C4T_CON1** 寄存器的 CNTEN 位为，开启计数器。

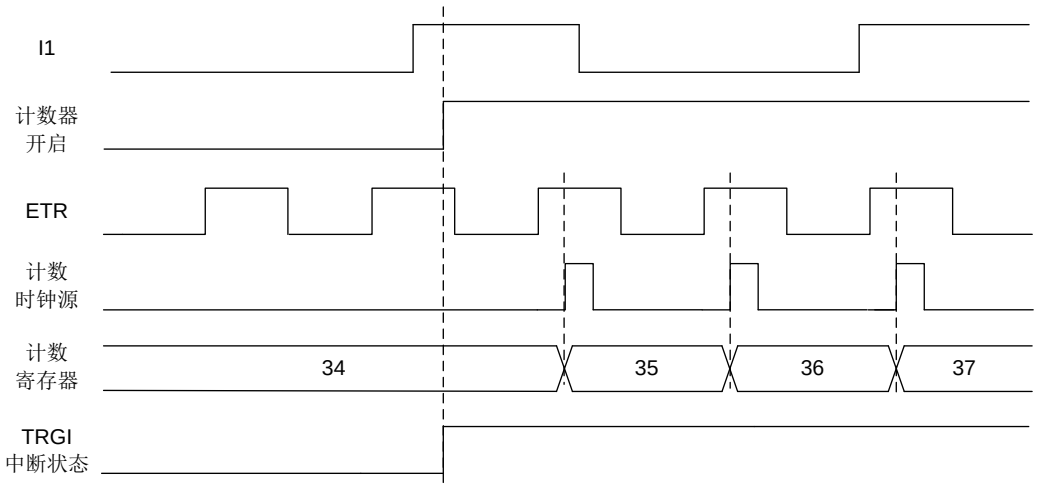


图 22-51　外部时钟源 2＋触发模式下的控制电路

I1 上出现上升沿时，计数器开启且设置 TRGI 标志位为 1，然后计数器根据 ETR 上的上升沿开始计数。

由于 ETF 输入同步电路的原因，ETR 信号的上升沿和实际计数器的计数会有延迟。

21. 5. 23. 5　组合复位模式 + 触发模式

要使用此模式，需要设置 **GP32C4T_SMCON** 寄存器的 SMODS 位为 1000b。其功能为复位模式加上触发模式，计数器在触发输入 TRGI 的上升沿启动，并重新初始化计数器以及生成更新事件。此模式主要用于单脉冲模式。

21. 5. 23. 6　组合门控模式 + 复位模式

要使用此模式，需要设置 **GP32C4T_SMCON** 寄存器的 SMODS 位为 1001b。其功能为门控模式加上复位模式，当触发输入(TRGI)为高电平时，计数器的时钟开启。一旦触发输入变为低电平，计数器停止且重新初始化。

21. 5. 24 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、开启、停止或提供时钟等操作。

下图显示了触发选择和主模选择模块的概况。

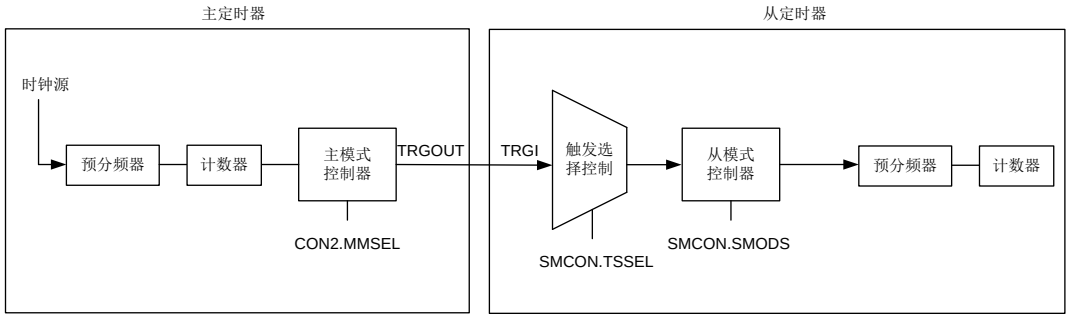


图 22-52　主/从定时器范例

21. 5. 24. 1　使用一个定时器去开启其他定时器

在这个例子中，定时器 2(GP32C4T1)的开启由定时器 1(GP16C2T1)的输出比较参考信号(CH1REF)控制。只有当定时器 1 的 CH1REF 为高电平时，定时器 2 才会计数。

先设定从定时器(定时器 2)为门控模式，配置如下:

1. 设置 **GP32C4T_SMCON** 寄存器的 TSSEL1 位与 TSSEL2 位为 GP16C2T1，可参考内部触发连接表。
2. 设置 **GP32C4T_SMCON** 寄存器的 SMODS 位为 101b，选择门控模式。
3. 设置 **GP32C4T_CON1** 寄存器的 CNTEN 位为 1，开启计数器。

再设定主定时器(定时器 1)为 PWM 输出，配置如下:

1. 设置 **GP16C2T_PRES** 寄存器的 PSCV 为 01h，计数器时钟频率为 fINT_CLK/2 。
2. 设置 **GP16C2T_CHMR1** 寄存器的 CH1MOD 位为 111b，选择 PWM 模式 2。
3. 设置 **GP16C2T_CCEP** 寄存器的 CC1POL 位为 0，选择 CH1 通道输出为高电平有效。
4. 设置 **GP16C2T_CCVAL1** 寄存器的 CCRV1 位为 06h，当计数器数到 6 时，PWM 输出高电平。
5. 设置 **GP16C2T_AR** 寄存器的 ARV 位为 08h，当计数器递增到 8 后重载。
6. 设置 **GP16C2T_CON2** 寄存器的 MMSEL 位为 100b，选择输出比较参考信号(CH1REF)为触发输出(TRGOUT)。
7. 设置 **GP16C2T_CON1** 寄存器的 CNTEN 位为 1，开启计数器。

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 计数器的开启信号。

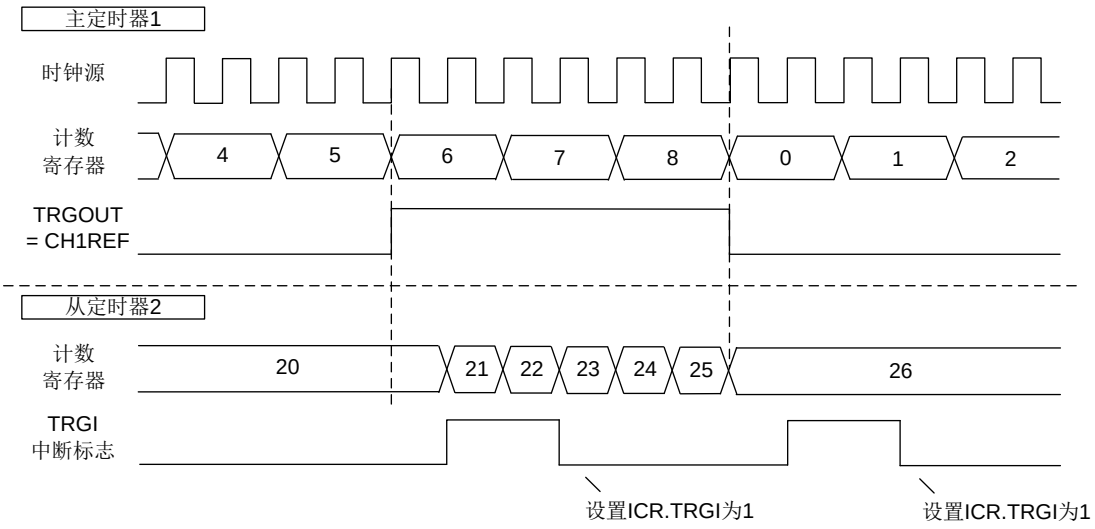


图 22-53　门控从定时器使用主定时器 CH1REF

在上图的例子中，在定时器 2 开启之前，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。可以在开启定时器 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。设置 **GP32C4T_SGE** 与定时器 2 的 SGE 寄存器的 SGUPD 位为 1 即可复位定时器。

21. 5. 24. 2　将一个定时器做为其他定时器的预分频器

在这个例子中，使用定时器 1(GP16C2T1)的更新事件作为定时器 2(GP32C4T1)的时钟来源，没有设定预分频。一旦定时器 1 产生更新事件，定时器 2 即根据其上升沿计数。

先设定从定时器(定时器 2)为外部时钟源 1 模式，配置如下:

1. 设置 **GP32C4T_AR** 寄存器的 ARV 位为 02h，当计数器递增到 2 后重载。
2. 设置 **GP32C4T_SMCON** 寄存器的 TSSEL1 位与 TSSEL2 位为 GP16C2T1，可参考内部触发连接表。
3. 设置 **GP32C4T_SMCON** 寄存器的 SMODS 位为 111b，选择外部时钟模式 1。
4. 设置 **GP32C4T_CON1** 寄存器的 CNTEN 位为 1，开启计数器。

再设定主定时器(定时器 1)配置如下:

1. 设置 **GP16C2T_AR** 寄存器的 ARV 位为 03h，当计数器递增到 3 后重载，并产生更新事件。
2. 设置 **GP16C2T_CON2** 寄存器的 MMSEL 位为 010b，选择更新事件(UPD)为触发输出(TRGOUT)。
3. 设置 **GP16C2T_CON1** 寄存器的 CNTEN 位为 1，开启计数器。

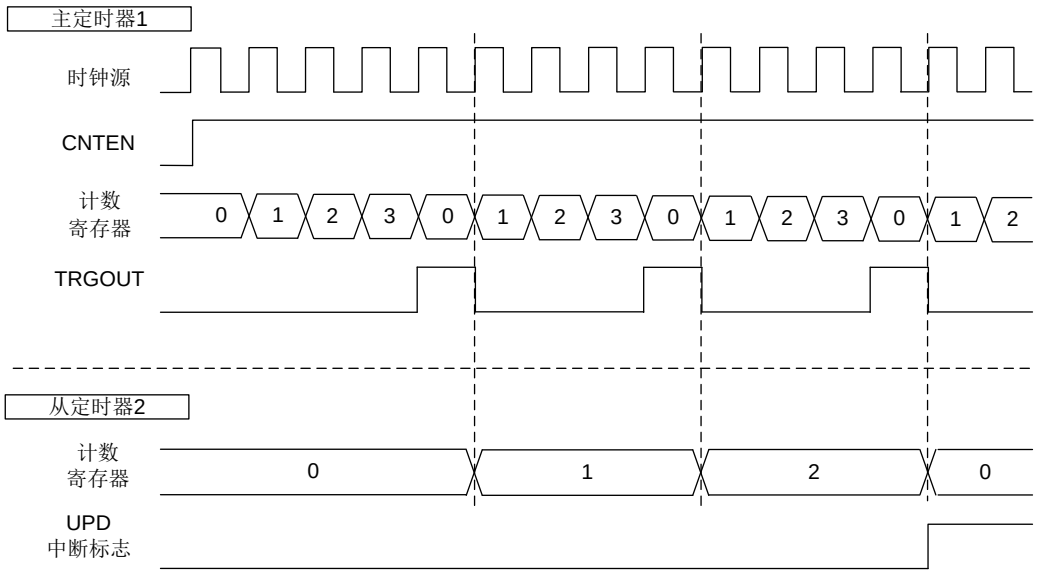


图 22-54　使用主定时器更新事件触发从定时器计数

21. 5. 24. 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(GP16C2T1)的 I1 输入上升沿时开启定时器 1，开启定时器 1 的同时开启定时器 2(GP32C4T1)。为保证计数器的对齐，定时器 1 必须设置为主/从模式(对应 I1 为从，对应定时器 2 为主):

先设定从定时器(定时器 2)为触发模式，配置如下:

1. 设置 **GP32C4T_SMCON** 寄存器的 TSSEL1 位与 TSSEL2 位为 GP16C2T1，可参考内部触发连接表。
2. 设置 **GP32C4T_SMCON** 寄存器的 SMODS 位为 110b，选择触发模式。

再设定主定时器(定时器 1)为触发模式，配置如下:

1. 设置定时器 1 为触发模式，相关配置可参考触发模式章节。
2. 设置 **GP16C2T_CON2** 寄存器的 MMSEL 位为 001b，选择开启信号(CNTEN)为触发输出(TRGOUT)。
3. 设置 **GP16C2T_SMCON** 寄存器的 MSCFG 位为 1，开启主/从模式。
4. 设置 **GP16C2T_CON1** 寄存器的 CNTEN 位为 1，开启计数器。

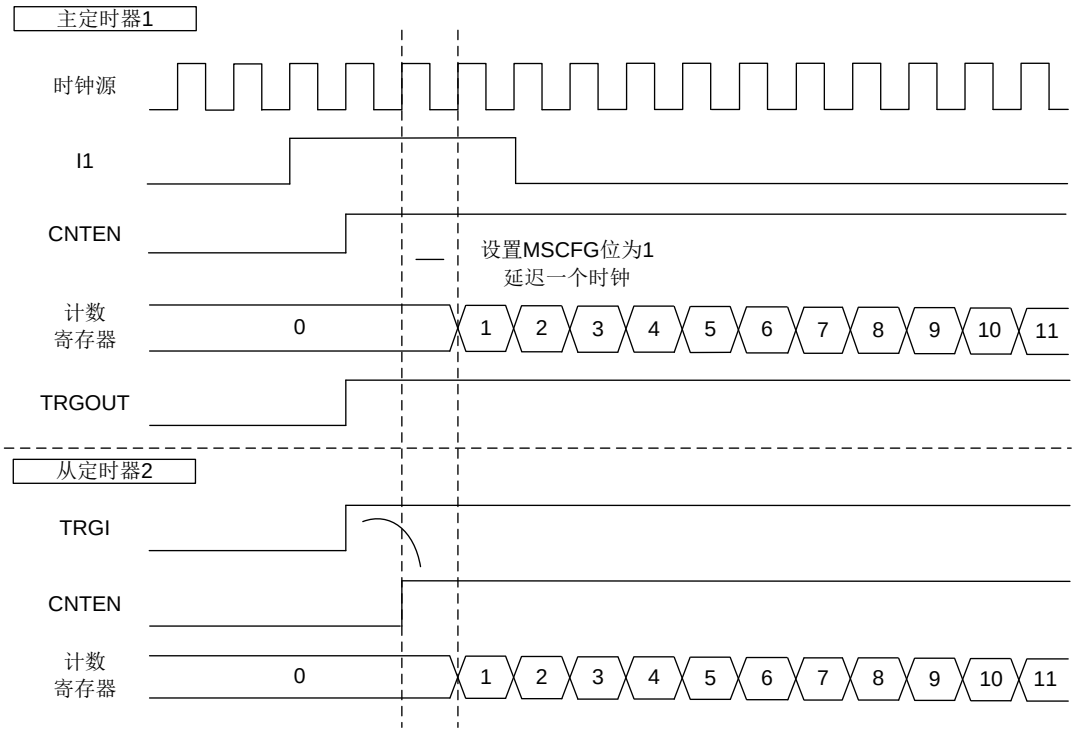


图 22-55　使用定时器 1 的 I1 输入触发定时器 1 和定时器 2

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 标志位也同时被设置。

21. 5. 25 ADC触发生成

定时器可生成 ADC 触发信号源如下 ：

- TRGOUT：由 **GP32C4T_CON2** 寄存器的 MMSEL 位决定触发事件的来源，根据配置生成脉冲或电平信号。
- CHx：通道输出的电平信号。

下图以 CH1 输出 PWM 模式 2 为例，说明 TRGOUT 与 CH1 信号源，相关配置如下:

1. 设置 **GP32C4T_AR** 寄存器的 ARV 位为 07h，当计数器递增到 7 后重载。
2. 设置 **GP32C4T_CHMR1** 寄存器的 CH1MOD 位为 111b，选择 PWM 模式 2。
3. 设置 **GP32C4T_CCVAL1** 寄存器的 CCRV1 位为 04h，当计数器数到 4 时，PWM 输出高电平。
4. 设置 **GP32C4T_CON2** 寄存器的 MMSEL 位为 00100b，选择输出比较参考信号(CH1REF)为触发输出(TRGOUT)。

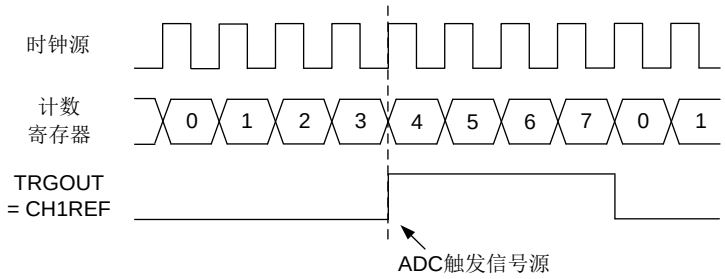


图 22-56　ADC 触发生成

21. 5. 26 调试模式

当微控制器进入调试模式(Cortex™-M0 核停止运行)，根据 SYSCFG 章节中 SYSCFG_CFG 寄存器配置，选择将计数器继续正常工作或停止计数。

21. 6 特殊功能寄存器

21. 6. 1 寄存器列表

GP32C4T 寄存器列表			
名称	偏移地址	类型	描述
GP32C4T_CON1	0000_H	R/W	控制寄存器 1
GP32C4T_CON2	0004_H	R/W	控制寄存器 2
GP32C4T_SMCON	0008_H	R/W	从模式控制寄存器
GP32C4T_IER	$000C_H$	W1	中断开启寄存器
GP32C4T_IDR	0010_H	W1	中断关闭寄存器
GP32C4T_IVS	0014_H	R	中断功能有效状态寄存器
GP32C4T_RIF	0018_H	R	原始中断状态寄存器
GP32C4T_IFM	$001C_H$	R	中断标志位状态寄存器
GP32C4T_ICR	0020_H	C_W1	中断清除寄存器
GP32C4T_SGE	0024_H	T_W1	软件生成事件寄存器
GP32C4T_CHMR1	0028_H	R/W	捕获或比较模式寄存器 1
GP32C4T_CHMR2	$002C_H$	R/W	捕获或比较模式寄存器 2
GP32C4T_CCEP	0030_H	R/W	捕获或比较开启极性寄存器
GP32C4T_COUNT	0034_H	R/W	计数寄存器
GP32C4T_PRES	0038_H	R/W	时钟预分频寄存器
GP32C4T_AR	$003C_H$	R/W	自动重载寄存器
GP32C4T_CCVAL1	0044_H	R/W	通道捕获或比较寄存器 1
GP32C4T_CCVAL2	0048_H	R/W	通道捕获或比较寄存器 2
GP32C4T_CCVAL3	$004C_H$	R/W	通道捕获或比较寄存器 3
GP32C4T_CCVAL4	0050_H	R/W	通道捕获或比较寄存器 4
GP32C4T_DMAEN	0058H	R/W	DMA 事件开启寄存器
GP32C4T_ENCR	006CH	R/W	编码控制寄存器
GP32C4T_CHISEL	0070H	R/W	通道输入来源选择寄存器
GP32C4T_AFR1	0074H	R/W	复用功能寄存器 1
GP32C4T_AFR2	0078H	R/W	复用功能寄存器 2

21. 6. 2 寄存器描述

21. 6. 2. 1 控制寄存器 1(GP32C4T_CON1)

控制寄存器 1(GP32C4T_CON1)																												
偏移地址:0x00																												
复位值:0x0000 0000																												
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																				UPDREMAP		DFCKSEL<1:0>		ARPEN	CMSEL<1:0>		DIRSEL	SPMEN	UERSEL	DISUE	CNTEN

—	Bits 31-16	—	—
UPDREMAP	Bit 11	R/W	**UPD原始中断状态重新映射** UPD原始中断状态复制到GP32C4T_COUNT寄存器的31位 0:重新映射关闭 1:重新映射开启
—	Bit 10	—	—
DFCKSEL	Bit 9-8	R/W	**时钟预分频器** 设置定时器频率(INT_CLK)，和数字滤波器所采样时钟之间的分频比例。 00: $t_{DTS}=t_{INT\_CLK}$ 01: $t_{DTS}=2 \times t_{INT\_CLK}$ 10: $t_{DTS}=4 \times t_{INT\_CLK}$ 11:保留
ARPEN	Bit 7	R/W	**自动重载预装载开启** 发生更新事件时，将预装载值载入到缓冲寄存器中 0:GP32C4T_AR 寄存器不具备缓冲 1:GP32C4T_AR 寄存器具备缓冲
CMSEL	Bit 6-5	R/W	**中心对齐模式选择** 00:边沿对齐模式，根据计数方向(DIRSEL)的配置，计数器为递增计数或递减计数。 01:中心对齐模式 1，计数器为交替地递增计数和递减计数。在此模式时仅当计数器在递减计数时，才会产生配置为输出的通道(GP32C4T_CHMRn 寄存器中 CCnSSEL=00)的比较匹配中断请求。

			10:中心对齐模式 2，计数器为交替地递增计数和递减计数。在此模式时仅当计数器在递增计数时，才会产生配置为输出的通道(GP32C4T_CHMRn 寄存器中 CCnSSEL=00)的比较匹配中断请求。 11:中心对齐模式 3，计数器为交替地递增计数和递减计数。在此模式时当计数器在递增计数或递减计数时，皆会产生配置为输出的通道(GP32C4T_CHMRn 寄存器中 CCnSSEL=00)的比较匹配中断请求。 注:当 CNTEN 开启时，不允许边沿对齐模式与中心对齐模式互相切换
DIRSEL	Bit 4	R/W	**计数方向选择** 0:计数器递增计数 1:计数器递减计数 注: 当计数器配置为中心对齐模式或编码器模式时，此位只能读取计数器的计数方向
SPMEN	Bit 3	R/W	**单脉冲模式** 0:单脉冲模式关闭，计数器不停止 1:单脉冲模式开启，计数器在发生下一次更新事件时，会自动清除 CNTEN 位，并停止计数器
UERSEL	Bit 2	R/W	**更新事件来源选择** 设置更新事件(UPD)的来源 0:下列事件都会产生更新中断或 DMA 的请求: －计数器上溢或下溢 －设置 GP32C4T_SGE 寄存器的 SGUPD 位为 1 －通过从模式控制器所生成的更新事件 1:只有在计数器上溢或下溢时会生成更新中断或 DMA 的请求
DISUE	Bit 1	R/W	**更新事件禁止** 0:更新事件(UPD)开启时，下列事件皆会产生更新事件请求并将预装载值加载到缓冲寄存器中 －计数器上溢或下溢 －设置 GP32C4T_SGE 寄存器的 SGUPD 位为 1 －通过从模式控制器所生成的更新事件 1:更新事件(UPD)关闭时，不产生更新事件请

			求，GP32C4T_AR、GP32C4T_PRES 与 GP32C4T_CCVALn 寄存器的缓冲寄存器数值保持不变。但设置 GP32C4T_SGE 寄存器的 SGUPD 位为 1 或从模式控制器的复位请求，计数器和预分频器仍会被重新初始化
CNTEN	Bit 0	R/W	**计数器开启** 开启计数器后，外部时钟模式、门控模式和编码模式才能运作。而触发模式则可以由硬件将 CNTEN 位置 1 0:计数器关闭 1:计数器开启

21. 6. 2. 2 控制寄存器 2(GP32C4T_CON2)

控制寄存器 2(GP32C4T_CON2)

偏移地址:0x04

复位值:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
						MMSEL2																		I1SEL	MMSEL<2:0>			CCDMASEL			

—	Bits 31-26	—	—
MMSEL2	Bits 25	R/W	**主模式选择1** 参照MMSEL描述
—	Bits 24-8	—	—
I1SEL	Bit 7	R/W	**I1 选择** 0:I1 输入连接到 GP32C4T_CH1 引脚 1:I1 输入连接到 GP32C4T_CH1、CH2 和 CH3 引脚异或组合(XOR)输出
MMSEL	Bit 6-4	R/W	**主模式选择 1** 此位与 MMSEL2 组合成 MMSEL = {MMSEL2,MMSEL[2:0]} 选择在主模式下发送到从定时器的同步信号与 ADC 触发信号(TRGOUT) 0000:复位 – 设置 GP32C4T_SGE 寄存器中的 UPD 位为触发输出。如果复位由触发输入生成(从模式控制器配置为复位模式)，则 TRGOUT

			上的信号与实际复位信号之间会有延迟 0001:开启－设置 GP32C4T_CON1 寄存器中的 CNTEN 位为触发输出，可用于同步开启数个定时器。计数器开启信号可由 GP32C4T_CON1 寄存器中的 CNTEN 位或者门控模式下的触发输入产生。当计数器开启信号受控于触发输入时，TRGOUT 上的信号与实际触发信号之间会有延迟 0010:更新事件－更新事件被用于触发输出。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 0011:比较脉冲－每次发生捕获或比较匹配时，当产生 CH1 中断请求同时，触发输出会送出一个正脉冲 0100:比较信号－CH1REF 信号用于触发输出 0101:比较信号－CH2REF 信号用于触发输出 0110:比较信号－CH3REF 信号用于触发输出 0111:比较信号－CH4REF 信号用于触发输出 其他:保留 注:编码器时钟输出只在 GP32C4T_SMCON 寄存器中的 SMODS 为编码器模式有效
CCDMASEL	Bit3	R/W	**捕获或比较事件的 DMA 选择** 0:当发生 CHn 事件时，设置 CHn DMA 请求 1:当发生更新事件时，设置 CHn DMA 请求
—	Bit2-0	—	—

21. 6. 2. 3 从模式控制寄存器(GP32C4T_SMCON)

从模式控制寄存器(GP32C4T_SMCON)																													
偏移地址:0x08																													
复位值:0x0000 0000																													
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
—	—	—	—	—	—	SMODPS	SMODPE	—	—	TSSEL2 <1:0>		—	—	—	SMODS2	ETPOL	ECM2EN	ETPRES <1:0>		ETFLT<3:0>				MSCFG	TSSEL1<2:0>			OCLRS	SMODS<2:0>		

—	Bits 31-26	—	—
SMODPS	Bit 25	R/W	**SMODS更新事件来源选择** 0:发生更新事件(UPD)时，将预装载值载入到缓冲寄存器中 1:发生Index事件(IDX)时，将预装载值载入到缓冲寄存器中
SMODPE	Bit 24	R/W	**SMODS预装载开启** 0:SMODS位不具备缓冲 1:SMODS位具备缓冲
—	Bits 23-22	—	—
TSSEL2	Bit 21-20	R/W	**触发来源选择2** 参照TSSEL1描述
—	Bit 19-17	—	—
SMODS2	Bit 16	R/W	**从模式选择** 参照SMODS描述
ETPOL	Bit 15	R/W	**外部触发极性** 设置外部触发开启电平 0: ETR不反向，高电平或上升沿时开启 1: ETR反向，低电平或下降沿时开启
ECM2EN	Bit 14	R/W	**外部时钟模式2开启** 0:外部时钟模式2关闭 1:外部时钟模式2开启，计数器时钟根据ETP信号的任意有效边沿提供 注: 1. 设置ECM2EN位与选择外部时钟模式1并将ETP(SMODS=111 和 TSSEL=00111) 连 到 TRGI具有相同功效。 2. 从模式可以与外部时钟模式2同时使用:复位

			模式，门控模式和触发模式;但是这时TRGI不能连到ETP(TSSEL位不能是00111)。 3. 外部时钟模式1和外部时钟模式2同时被开启时，外部时钟的输入是ETP。
ETPRES	Bit 13-12	R/W	**外部触发时钟预分频器** 外部触发输入信号ETR频率需要大于1/4 INT_CLK，分频器可以达到降低ETR的频率，有效应用于快速的外部时钟源 00:预分频器禁止 01:ETR频率2分频 10:ETR频率4分频 11:ETR频率8分频
ETFLT	Bit 11-8	R/W	**外部触发滤波器** 设置ETP信号采样的频率和对ETP数字滤波器的带宽。数字滤波器是一个事件计数器，它记录到N个事件后才视为一个有效输出边沿: 0000:采样频率f_{DTS}，滤波器禁止 0001:采样频率f_{INT_CLK}, N = 2 0010:采样频率f_{INT_CLK}, N = 4 0011:采样频率f_{INT_CLK}, N = 8 0100:采样频率f_{DTS} / 2, N = 6 0101:采样频率f_{DTS} / 2, N = 8 0110:采样频率f_{DTS} / 4, N = 6 0111:采样频率f_{DTS} / 4, N = 8 1000:采样频率f_{DTS} / 8, N = 6 1001:采样频率f_{DTS} / 8, N = 8 1010:采样频率f_{DTS} / 16, N = 5 1011:采样频率f_{DTS} / 16, N = 6 1100:采样频率f_{DTS} / 16, N = 8 1101:采样频率f_{DTS} / 32, N = 5 1110:采样频率f_{DTS} / 32, N = 6 1111:采样频率f_{DTS} / 32, N = 8
MSCFG	Bit 7	R/W	**主/从模式** 延迟触发输入(TRGI)上的事件来允许当前主定时器和其从定时器之间的同步。该设置有效用于使用单个外部事件来同步多个定时器 0: 主从模式关闭 1: 主从模式开启

TSSEL1	Bit 6-4	R/W	**触发来源选择 1** 此位与 TSSEL2[1:0]组合成 TSSEL= {TSSEL2[1:0],TSSEL1[2:0]} 设置触发选择，用于同步计数器 00000:内部触发 0 (IT0) 00001:内部触发 1 (IT1) 00010:内部触发 2 (IT2) 00011:内部触发 3 (IT3) 00100: I1 边沿检测(I1F_ED) 00101: I1 滤波后信号(I1FP) 00110: I2 滤波后信号(I2FP) 00111:外部触发输入(ETP) 01000:内部触发 4 (IT4) 01001:内部触发 5 (IT5) 01010:内部触发 6 (IT6) 01011:内部触发 7 (IT7) 其他:保留 注:此位在需要在使用前设定(SMODS=000)，以避免产生错误的上升/下降沿至计数器
OCLRS	Bit 3	R/W	**输出通道清除选择** 选择输出通道清除的来源 0: 选择来源为 OCLR 1: 选择来源为 ETP
SMODS	Bit 2-0	R/W	**从模式选择** 此位与 SMODS2 组合成 SMODS = {SMODS2, SMODS[2:0]} 0000: 从模式关闭 - 当 GP32C4T_CON1 寄存器 CNTEN 位为 1 时，预分频器时钟由内部时钟提供 0001: 正交编码器模式 1 - x2 模式，计数器根据 I2 电平在 I1 边沿递增或递减计数 0010: 正交编码器模式 2 - x2 模式，计数器根据 I1 电平在 I2 边沿递增或递减计数 0011: 正交编码器模式 3 - x4 模式，计数器根据 I2/I1 电平在 I1/I2 边沿递增或递减计数 0100: 复位模式 - 选中的触发输入(TRGI)的上升沿重新初始化计数器，并且产生一次更新事件

			0101: 门控模式 - 当触发输入(TRGI)为高电平时，计数器的时钟开启。一旦触发输入变为低电平，则计数器停止(但不复位)。计数器的启动和停止都是被控制 0110: 触发模式 - 计数器在触发输入 TRGI 的上升沿启动(但不复位)，只有计数器的启动是被控制 0111: 外部时钟模式 1 - 选中的触发输入(TRGI)的上升沿提供计数器时钟 1000: 组合复位模式 + 触发模式 - 选中的触发输入(TRGI)的上升沿重新初始化计数器，并且产生一次更新事件启动计数器 1001: 组合门控模式 + 复位模式 - 当触发输入(TRGI)为高电平时，计数器的时钟开启。一旦触发输入变为低电平，则计数器停止且复位。计数器的启动和停止都是被控制 1010: 时钟加方向编码器模式 1 - x2 模式，计数器根据 I1 电平控制计数方向，并在 I2 边沿计数 1011: 时钟加方向编码器模式 2 - x1 模式，计数器根据 I1 电平控制计数方向，并由 CC2POL 决定在 I2 的上升或下降边沿计数 1100: 定向时钟编码器模式 1: x2 模式，计数器根据 I2 指定电平在 I1 边沿递减计数，根据 I1 指定电平在 I2 边沿递增计数 1101: 定向时钟编码器模式 2: x1 模式，计数器根据 I2 指定电平在 I1 上升或下降边沿(根据 CC1POL)递减计数，根据 I1 指定电平在 I2 上升或下降边沿(根据 CC2POL)递增计数 1110: 正交编码器模式 4 - x1 模式，计数器根据 I2 指定电平在 I1 边沿递增或递减计数 1111: 正交编码器模式 5 - x1 模式，计数器根据 I1 指定电平在 I2 边沿递增或递减计数 注:如果 I1 双边沿检测被选为触发输入(设置 TSSEL 为 00100)时，不能使用门控模式。I1F 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

21. 6. 2. 4 中断开启寄存器(GP32C4T_IER)

中断开启寄存器(GP32C4T_IER)																														
偏移地址:0x0C																														
复位值:0x0000 0000																														
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
—	—	—	—	—	—	—	—	TERR	IERR	DIR	IDX	—	—	—	—	—	—	—	CH4O	CH3OV	CH2OV	CH1OV	—	—	TRGI	—	CH4	CH3	CH2	CH1	UPD

—	Bits 31-24	—	—
TERR	Bit 23	W1	**开启转换错误中断功能** 此位设置时，开启中断功能，在编码器模式下，硬件侦测相位转换错误事件时发生中断
IERR	Bit 22	W1	**开启Index错误中断功能** 此位设置时，开启中断功能，在编码器模式下，硬件侦测Index错误事件时发生中断
DIR	Bit 21	W1	**开启方向转换中断功能** 此位设置时，开启中断功能，在编码器模式下，硬件侦测方向转换事件时发生中断
IDX	Bit 20	W1	**开启有效Index中断功能** 此位设置时，开启中断功能，在编码器模式下，硬件侦测有效Index事件时发生中断
—	Bits 19-13	—	—
CH4OV	Bit 12	W1	**开启通道4捕获溢出中断功能** 此位设置时，开启中断功能，硬件侦测通道4捕获溢出事件时发生中断
CH3OV	Bit 11	W1	**开启通道 3 捕获溢出中断功能** 此位设置时，开启中断功能，硬件侦测通道 3 捕获溢出事件时发生中断
CH2OV	Bit 10	W1	**开启通道 2 捕获溢出中断功能** 此位设置时，开启中断功能，硬件侦测通道 2 捕获溢出事件时发生中断
CH1OV	Bit 9	W1	**开启通道 1 捕获溢出中断功能** 此位设置时，开启中断功能，硬件侦测通道 1 捕获溢出事件时发生中断
—	Bit 8-7	—	—
TRGI	Bit 6	W1	**开启触发中断功能**

			此位设置时，开启中断功能，硬件侦测触发信号事件时发生中断
—	Bit 5	—	—
CH4	Bit 4	W1	**开启通道 4 捕获或比较匹配中断功能** 此位设置时，开启中断功能，硬件侦测通道 4 捕获或比较匹配事件时发生中断
CH3	Bit 3	W1	**开启通道 3 捕获或比较匹配中断功能** 此位设置时，开启中断功能，硬件侦测通道 3 捕获或比较匹配事件时发生中断
CH2	Bit 2	W1	**开启通道 2 捕获或比较匹配中断功能** 此位设置时，开启中断功能，硬件侦测通道 2 捕获或比较匹配事件时发生中断
CH1	Bit 1	W1	**开启通道 1 捕获或比较匹配中断功能** 此位设置时，开启中断功能，硬件侦测通道 1 捕获或比较匹配事件时发生中断
UPD	Bit 0	W1	**开启更新中断功能** 此位设置时，开启中断功能，硬件侦测更新事件时发生中断

21. 6. 2. 5 中断关闭寄存器(GP32C4T_IDR)

中断关闭寄存器(GP32C4T_IDR)

偏移地址:0x10

复位值:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
—	—	—	—	—	—	—	—	TERR	IERR	DIR	IDX	—	—	—	—	—	—	—	CH4OV	CH3OV	CH2OV	CH1OV	—	—	TRGI	—	CH4	CH3	CH2	CH1	UPD

—	Bits 31-24	—	—
TERR	Bit 23	W1	**关闭转换错误中断功能** 此位设置时，关闭转换错误中断功能
IERR	Bit 22	W1	**关闭Index错误中断功能** 此位设置时，关闭Index错误中断功能
DIR	Bit 21	W1	**关闭方向转换中断功能** 此位设置时，关闭方向转换中断功能
IDX	Bit 20	W1	**关闭有效Index中断功能** 此位设置时，关闭有效Index中断功能

—	Bits 19-13	—	—
CH4OV	Bit 12	W1	**关闭通道4捕获溢出中断功能** 此位设置时，关闭通道4捕获溢出中断功能
CH3OV	Bit 11	W1	**关闭通道 3 捕获溢出中断功能** 此位设置时，关闭通道 3 捕获溢出中断功能
CH2OV	Bit 10	W1	**关闭通道 2 捕获溢出中断功能** 此位设置时，关闭通道 2 捕获溢出中断功能
CH1OV	Bit 9	W1	**关闭通道 1 捕获溢出中断功能** 此位设置时，关闭通道 1 捕获溢出中断功能
—	Bit 8-7	—	—
TRGI	Bit 6	W1	**关闭触发中断功能** 此位设置时，关闭触发中断功能
—	Bit 5	—	—
CH4	Bit 4	W1	**关闭通道 4 捕获或比较匹配中断功能** 此位设置时，关闭通道 4 捕获或比较匹配中断功能
CH3	Bit 3	W1	**关闭通道 3 捕获或比较匹配中断功能** 此位设置时，关闭通道 3 捕获或比较匹配中断功能
CH2	Bit 2	W1	**关闭通道 2 捕获或比较匹配中断功能** 此位设置时，关闭通道 2 捕获或比较匹配中断功能
CH1	Bit 1	W1	**关闭通道 1 捕获或比较匹配中断功能** 此位设置时，关闭通道 1 捕获或比较匹配中断功能
UPD	Bit 0	W1	**关闭更新中断功能** 此位设置时，关闭更新中断功能

21. 6. 2. 6 中断功能有效状态寄存器(GP32C4T_IVS)

中断功能有效状态寄存器(GP32C4T_IVS)																													
偏移地址:0x14																													
复位值:0x0000 0000																													
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
—	—	—	—	—	—	—	—	TERR	IERR	DIR	IDX	—	—	—	—	—	—	—	CH4OV	CH3OV	CH2OV	CH1OV	—	—	TRGI	—	CH4	CH3	CH2	CH1	UPD

—	Bits 31-24	—	—
TERR	Bit 23	R	**转换错误中断功能状态** 0: 中断功能处于关闭状态 1: 中断功能处于开启状态
IERR	Bit 22	R	**Index错误中断功能状态** 0: 中断功能处于关闭状态 1: 中断功能处于开启状态
DIR	Bit 21	R	**方向转换中断功能状态** 0: 中断功能处于关闭状态 1: 中断功能处于开启状态
IDX	Bit 20	R	**有效Index中断功能状态** 0: 中断功能处于关闭状态 1: 中断功能处于开启状态
—	Bits 19-13	—	—
CH4OV	Bit 12	R	**通道 4 捕获溢出中断功能状态** 0:中断功能处于关闭状态 1:中断功能处于开启状态
CH3OV	Bit 11	R	**通道 3 捕获溢出中断功能状态** 0:中断功能处于关闭状态 1:中断功能处于开启状态
CH2OV	Bit 10	R	**通道 2 捕获溢出中断功能状态** 0:中断功能处于关闭状态 1:中断功能处于开启状态
CH1OV	Bit 9	R	**通道 1 捕获溢出中断功能状态** 0:中断功能处于关闭状态 1:中断功能处于开启状态
—	Bit 8-7	—	—
TRGI	Bit 6	R	**触发中断功能状态**

			0:中断功能处于关闭状态 1:中断功能处于开启状态
—	Bit 5	—	—
CH4	Bit 4	R	**通道 4 捕获或比较匹配中断功能状态** 0:中断功能处于关闭状态 1:中断功能处于开启状态
CH3	Bit 3	R	**通道 3 捕获或比较匹配中断功能状态** 0:中断功能处于关闭状态 1:中断功能处于开启状态
CH2	Bit 2	R	**通道 2 捕获或比较匹配中断功能状态** 0:中断功能处于关闭状态 1:中断功能处于开启状态
CH1	Bit 1	R	**通道 1 捕获或比较匹配中断功能状态** 0:中断功能处于关闭状态 1:中断功能处于开启状态
UPD	Bit 0	R	**更新中断功能状态** 0:中断功能处于关闭状态 1:中断功能处于开启状态

GP32C4T_IVS 寄存器，是实时反映系统配置 GP32C4T_IER 与 GP32C4T_IDR 的中断开启状态。此寄存器状态是将 GP32C4T_IER 与 GP32C4T_IDR 进行硬件运算，公式如下:GP32C4T_IVS = GP32C4T_IER & ~GP32C4T_IDR

21. 6. 2. 7 原始中断状态寄存器(GP32C4T_RIF)

原始中断状态寄存器(GP32C4T_RIF)																														
偏移地址:0x18																														
复位值:0x0000 0000																														
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
—	—	—	—	—	—	—	—	TERR	IERR	DIR	IDX	—	—	—	—	—	—	—	CH4OV	CH3OV	CH2OV	CH1OV	—	—	TRGI	—	CH4	CH3	CH2	CH1	UPD

—	Bits 31-24	—	—
TERR	Bit 23	R	**转换错误，原始中断状态** 0: 无发生中断 1: 已发生中断
IERR	Bit 22	R	**Index错误，原始中断状态** 0: 无发生中断 1: 已发生中断
DIR	Bit 21	R	**方向转换，原始中断状态** 0: 无发生中断 1: 已发生中断
IDX	Bit 20	R	**有效Index，原始中断状态** 0: 无发生中断 1: 已发生中断
—	Bits 19-13	—	—
CH4OV	Bit 12	R	**通道4捕获溢出，原始中断状态** 0:无发生中断 1:已发生中断
CH3OV	Bit 11	R	**通道 3 捕获溢出，原始中断状态** 0:无发生中断 1:已发生中断
CH2OV	Bit 10	R	**通道 2 捕获溢出，原始中断状态** 0:无发生中断 1:已发生中断
CH1OV	Bit 9	R	**通道 1 捕获溢出，原始中断状态** 0:无发生中断 1:已发生中断
—	Bit 8-7	—	—
TRGI	Bit 6	R	**触发，原始中断状态** 0:无发生中断

			1:已发生中断
—	Bit 5	—	—
CH4	Bit 4	R	**通道 4 捕获或比较匹配，原始中断状态** 0:无发生中断 1:已发生中断
CH3	Bit 3	R	**通道 3 捕获或比较匹配，原始中断状态** 0:无发生中断 1:已发生中断
CH2	Bit 2	R	**通道 2 捕获或比较匹配，原始中断状态** 0:无发生中断 1:已发生中断
CH1	Bit 1	R	**通道 1 捕获或比较匹配，原始中断状态** 0:无发生中断 1:已发生中断
UPD	Bit 0	R	**更新，原始中断状态** 0:无发生中断 1:已发生中断

21. 6. 2. 8 中断标志位状态寄存器(GP32C4T_IFM)

中断标志位状态寄存器(GP32C4T_IFM)

偏移地址:0x1C

复位值:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
—	—	—	—	—	—	—	—	TERR	IERR	DIR	IDX	—	—	—	—	—	—	—	CH4OV	CH3OV	CH2OV	CH1OV	—	—	TRGI	—	CH4	CH3	CH2	CH1	UPD

—	Bits 31-13	—	—
TERR	Bit 23	R	**转换错误，中断标志位状态** 0: 无发生中断 1: 已发生中断
IERR	Bit 22	R	**Index错误，中断标志位状态** 0: 无发生中断 1: 已发生中断
DIR	Bit 21	R	**方向转换，中断标志位状态** 0: 无发生中断 1: 已发生中断
IDX	Bit 20	R	**有效Index，中断标志位状态**

			0: 无发生中断 1: 已发生中断
—	Bits 19-13	—	—
CH4OV	Bit 12	R	**通道4捕获溢出，中断标志位状态** 0:无发生中断 1:已发生中断
CH3OV	Bit 11	R	**通道 3 捕获溢出，中断标志位状态** 0:无发生中断 1:已发生中断
CH2OV	Bit 10	R	**通道 2 捕获溢出，中断标志位状态** 0:无发生中断 1:已发生中断
CH1OV	Bit 9	R	**通道 1 捕获溢出，中断标志位状态** 0:无发生中断 1:已发生中断
—	Bit 8-7	—	—
TRGI	Bit 6	R	**触发，中断标志位状态** 0:无发生中断 1:已发生中断
—	Bit 5	—	—
CH4	Bit 4	R	**通道 4 捕获或比较匹配，中断标志位状态** 0:无发生中断 1:已发生中断
CH3	Bit 3	R	**通道 3 捕获或比较匹配，中断标志位状态** 0:无发生中断 1:已发生中断
CH2	Bit 2	R	**通道 2 捕获或比较匹配，中断标志位状态** 0:无发生中断 1:已发生中断
CH1	Bit 1	R	**通道 1 捕获或比较匹配，中断标志位状态** 0:无发生中断 1:已发生中断
UPD	Bit 0	R	**更新，中断标志位状态** 0:无发生中断 1:已发生中断

GP32C4T_IFM 寄存器，是滤除已关闭中断功能的中断事件，只关注开启中断功能的事件，此寄存器状态是将 GP32C4T_RIF 与 GP32C4T_IVS 进行硬件运算，公式如下:GP32C4T_IFM = GP32C4T_RIF &GP32C4T_IVS

21. 6. 2. 9 中断清除寄存器(GP32C4T_ICR)

中断清除寄存器(GP32C4T_ICR)																													
偏移地址: 0x20																													
复位值:0x0000 0000																													
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
—	—	—	—	—	—	—	—	TERR	IERR	DIR	IDX	—	—	—	—	—	—	—	CH4OV	CH3OV	CH2OV	CH1OV	—	—	TRGI	—	CH4	CH3	CH2	CH1	UPD

—	Bits 31-13	—	—
TERR	Bit 23	C_W1	**清除转换错误中断状态** 此位设置时，清除中断状态(GP32C4T_RIF与GP32C4T_IFM)
IERR	Bit 22	C_W1	**清除Index错误中断状态** 此位设置时，清除中断状态(GP32C4T_RIF与GP32C4T_IFM)
DIR	Bit 21	C_W1	**清除方向转换中断状态** 此位设置时，清除中断状态(GP32C4T_RIF与GP32C4T_IFM)
IDX	Bit 20	C_W1	**清除有效Index中断状态** 此位设置时，清除中断状态(GP32C4T_RIF与GP32C4T_IFM)
—	Bits 19-13	—	—
CH4OV	Bit 12	C_W1	**清除通道4捕获溢出中断状态** 此位设置时，清除中断状态(GP32C4T_RIF与GP32C4T_IFM)
CH3OV	Bit 11	C_W1	**清除通道 3 捕获溢出中断状态** 此位设置时，清除中断状态(GP32C4T_RIF 与GP32C4T_IFM)
CH2OV	Bit 10	C_W1	**清除通道 2 捕获溢出中断状态** 此位设置时，清除中断状态(GP32C4T_RIF 与GP32C4T_IFM)
CH1OV	Bit 9	C_W1	**清除通道 1 捕获溢出中断状态** 此位设置时，清除中断状态(GP32C4T_RIF 与GP32C4T_IFM)
—	Bit 8-7	—	—
TRGI	Bit 6	C_W1	**清除触发中断状态**

			此位设置时，清除中断状态(GP32C4T_RIF 与 GP32C4T_IFM)
—	Bit 5	—	—
CH4	Bit 4	C_W1	**清除通道 4 捕获或比较匹配中断状态** 此位设置时，清除中断状态(GP32C4T_RIF 与 GP32C4T_IFM)
CH3	Bit 3	C_W1	**清除通道 3 捕获或比较匹配中断状态** 此位设置时，清除中断状态(GP32C4T_RIF 与 GP32C4T_IFM)
CH2	Bit 2	C_W1	**清除通道 2 捕获或比较匹配中断状态** 此位设置时，清除中断状态(GP32C4T_RIF 与 GP32C4T_IFM)
CH1	Bit 1	C_W1	**清除通道 1 捕获或比较匹配中断状态** 此位设置时，清除中断状态(GP32C4T_RIF 与 GP32C4T_IFM)
UPD	Bit 0	C_W1	**清除更新中断状态** 此位设置时，清除中断状态(GP32C4T_RIF 与 GP32C4T_IFM)

GP32C4T_ICR 寄存器设置时，将清除 GP32C4T_RIF 与 GP32C4T_IFM 中断标志位状态；此设置不影响中断 GP32C4T_IER、GP32C4T_IDR 与 GP32C4T_IVS 寄存器，只清除标志位状态 GP32C4T_RIF 与 GP32C4T_IFM。此寄存器通过硬件清除中断，公式如下:GP32C4T_RIF = GP32C4T_RIF & ~GP32C4T_ICR

21. 6. 2. 10 软件生成事件寄存器(GP32C4T_SGE)

软件生成事件寄存器(GP32C4T_SGE)

偏移地址:0x24

复位值:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																									SGTRGI		SGCH4	SGCH3	SGCH2	SGCH1	SGUPD

—	Bits 31-7	—	—
SGTRGI	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。 此位设置时，产生触发事件。产生相关中断或 DMA 传输

—	Bit 5	—	—
SGCH4	Bit 4	W1	**软件生成通道 4 捕获/比较事件** 参照 SGCH1 描述
SGCH3	Bit 3	W1	**软件生成通道 3 捕获/比较事件** 参照 SGCH1 描述
SGCH2	Bit 2	W1	**软件生成通道 2 捕获/比较事件** 参照 SGCH1 描述
SGCH1	Bit 1	W1	**软件生成通道 1 捕获/比较事件** 该位由软件设置来生成通道 1 捕获或比较，可由硬件自动清零。 **通道 CH1 设置为输出:** 此位设置时，产生比较事件，但不影响输出。若开启中断或 DMA，则产生中断与请求 **通道 CH1 设置为输入:** 此位设置时，产生捕获事件。将计数器捕获至 CCVAL1 寄存器中，于 I1 的有效沿产生，若开启中断或 DMA，则产生中断与请求
SGUPD	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 此位设置时，产生更新事件。重新初始化计数器，更新寄存器 注: 预分频器也会被清零(但预分频比不会受到影响)。如果使用中心对齐模式或者 DIRSEL=0(递增)，则计数器将清零；否则如果 DIRSEL=1(递减)，则计数器将加载自动重载值(GP32C4T_AR)

21. 6. 2. 11 捕获或比较模式寄存器 1(GP32C4T_CHMR1)

捕获或比较模式寄存器 1(GP32C4T_CHMR1)																													
偏移地址:0x28																													
复位值:0x0000 0000																													
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
							CH2MOD2								CH1MOD2	CH2OCLREN	CH2MOD<2:0>			CH2PEN	CH2FEN	CC2SSEL<1:0>		CH1OCLREN	CH1MOD<2:0>			CH1PEN	CH1FEN	CC1SSEL<1:0>	
																I2FLT<3:0>				I2PRES<1:0>				I1FLT<3:0>				I1PRES<1:0>		

输出比较模式

—	Bits 31-25	—	—
CH2MOD2	Bit 24	R/W	**输出比较通道2模式** 参照CH1MOD描述
—	Bits 23-17	—	—
CH1MOD2	Bit 16	R/W	**输出比较通道1模式** 参照CH1MOD描述
CH2OCLREN	Bit 15	R/W	**输出比较通道 2 清除开启** 参照 CH1OCLREN 描述
CH2MOD	Bit 14-12	R/W	**输出比较通道 2 模式** 参照 CH1MOD 描述
CH2PEN	Bit 11	R/W	**输出比较通道 2 预装载开启** 参照 CH1PEN 描述
CH2FEN	Bit 10	R/W	**输出比较通道 2 快速开启** 参照 CH1FEN 描述
CC2SSEL	Bit 9-8	R/W	**捕获或比较通道 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(只工作在 GP32C4T_SMCON 寄存器的 TSSEL 位选择为 ITn 时)

CH1OCLREN	Bit 7	R/W	**输出比较通道 1 清除开启** 0: CH1REF 维持输出 1:CH1REF 在 OCLR_INT(由 OCLRSEL 或 ETRSEL 选择)为高电平时清除
CH1MOD	Bit 6-4	R/W	**输出比较通道 1 模式** 此位与 CH1MOD2 组合成 CH1MOD[3:0] = {CH1MOD2, CH1MOD[2:0]} 这些位定义提供 CH1 的输出参考信号 CH1REF 的行为。CH1REF 为高电平有效，而 CH1 的有效电平则取决于 GP32C4T_CCEP 寄存器中的 CC1POL 位 0000:关闭－无作用 0001:匹配时设置高电平－当计数器(GP32C4T_COUNT)匹配 GP32C4T_CCVAL1 寄存器时，CH1REF 设置为 1 0010:匹配时设置低电平－当计数器(GP32C4T_COUNT)匹配 GP32C4T_CCVAL1 寄存器时，CH1REF 设置为 0 0011:匹配时翻转电平－当计数器(GP32C4T_COUNT)匹配 GP32C4T_CCVAL1 寄存器时，CH1REF 翻转电平(当前为高电平则翻转成低电平，反之当前为低电平则翻转成高电平) 0100:强制低电平－CH1REF 强制设置低电平 0101:强制高电平－CH1REF 强制设置高电平 0110:PWM 模式 1－在递增计数时，当计数器(GP32C4T_COUNT)小于 GP32C4T_CCVAL1 寄存器时，输出高电平，其他则输出低电平。在递减计数时，当计数器(GP32C4T_COUNT)大于 GP32C4T_CCVAL1 寄存器时，输出低电平，其他则输出高电平 0111:PWM 模式 2－在递增计数时，当计数器(GP32C4T_COUNT)小于 GP32C4T_CCVAL1 寄存器时，输出低电平，其他则输出高电平。在递减计数时，当计数器(GP32C4T_COUNT)大于 GP32C4T_CCVAL1 寄存器时，输出高电平，其他则输出低电平 1000: 多次触发单脉冲模式 1－在递增计数时，

			通道保持为高电平，直到在 TRGI 信号上侦测到触发事件后，通道正常输出 PWM 模式 1，直到下次更新事件发生后，通道再次保持高电平。在递减计数时，通道保持为低电平，直到在 TRGI 信号上侦测到触发事件后，通道正常输出 PWM 模式 1，直到下次更新事件发生后，通道再次保持低电平 1001: 多次触发单脉冲模式 2 - 在递增计数时，通道保持为低电平，直到在 TRGI 信号上侦测到触发事件后，通道正常输出 PWM 模式 2，直到下次更新事件发生后，通道再次保持低电平。在递减计数时，通道保持为高电平，直到在 TRGI 信号上侦测到触发事件后，通道正常输出 PWM 模式 2，直到下次更新事件发生后，通道再次保持高电平 1010: 保留 1011: 保留 1100: 组合 PWM 模式 1 - 在 PWM 模式 1 下产生 CH1REF，再将 CH1REF 和 CH2REF 做"OR"运算输出 CH1REFC 信号 1101: 组合 PWM 模式 2 - 在 PWM 模式 2 下产生 CH1REF，再将 CH1REF 和 CH2REF 做"AND"运算输出 CH1REFC 信号 1110: 非对称 C2 PWM 模式 1 - 在 PWM 模式 1 中，在递增计数时，CH1REFC 信号使用 CCRV1 输出比较。在递减计数时，CH1REFC 信号使用 CCRV2 1111: 非对称 C2 PWM 模式 2 - 在 PWM 模式 2 中，在递增计数时，CH1REFC 信号使用 CCRV1 输出比较。在递减计数时，CH1REFC 信号使用 CCRV2
CH1PEN	Bit 3	R/W	**输出比较通道 1 预装载开启** 设置后在更新事件时，将 GP32C4T_CCVAL1 寄存器预装载值载入到缓冲寄存器中。 0: CCVAL1 寄存器预装载关闭 1: CCVAL1 寄存器预装载开启
CH1FEN	Bit 2	R/W	**输出比较通道 1 快速开启** 用于加速触发输入事件对于 PWM 输出的影响，

			CH1FEN 只在通道被配置成 PWM1 或 PWM2 模式时起作用。 0:CH1 的输出依赖于计数器与 CCVAL1 的值正常工作。 1:当触发输入(TRGI)有效时，CH1 被强制设置为比较电平(与比较结果无关)，触发输入(TRGI)的有效边沿相当于发生比较匹配。
CC1SSEL	Bit 1-0	R/W	**捕获或比较通道 1 选择** 设置通道的输出方向与信号的选择，当 GP32C4T_CCEP 寄存器的 CC1EN 位为 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(只工作在 GP32C4T_SMCON 寄存器的 TSSEL 位选择为 ITn 时)

输入捕获模式

—	Bits 31-16	—	—
I2FLT	Bit 15-12	R/W	**输入捕获通道2滤波器** 参照I1FLT描述
I2PRES	Bit 11-10	R/W	**输入捕获通道 2 预分频器** 参照 IC1PRES 描述
CC2SSEL	Bit 9-8	R/W	**捕获或比较通道 2 选择** 设置通道的输出方向与信号的选择，当 GP32C4T_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(只工作在 GP32C4T_SMCON 寄存器的 TSSEL 位选择为 ITn 时)
I1FLT	Bit 7-4	R/W	**输入捕获通道 1 滤波器** 设置 I1 信号采样的频率和对 I1 数字滤波器的带宽。数字滤波器是一个事件计数器，它记录到 N 个事件后才视为一个有效输出边沿:

			0000:采样频率 f_{DTS}，滤波器禁止 0001:采样频率 f_{INT_CLK}, N = 2 0010:采样频率 f_{INT_CLK}, N = 4 0011:采样频率 f_{INT_CLK}, N = 8 0100:采样频率 f_{DTS} / 2, N = 6 0101:采样频率 f_{DTS} / 2, N = 8 0110:采样频率 f_{DTS} / 4, N = 6 0111:采样频率 f_{DTS} / 4, N = 8 1000:采样频率 f_{DTS} / 8, N = 6 1001:采样频率 f_{DTS} / 8, N = 8 1010:采样频率 f_{DTS} / 16, N = 5 1011:采样频率 f_{DTS} / 16, N = 6 1100:采样频率 f_{DTS} / 16, N = 8 1101:采样频率 f_{DTS} / 32, N = 5 1110:采样频率 f_{DTS} / 32, N = 6 1111:采样频率 f_{DTS} / 32, N = 8
I1PRES	Bit 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 GP32C4T_CCEP 寄存器的 CC1EN 位，预分频计数器同时被清除 00:预分频关闭，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	Bit 1-0	R/W	**捕获或比较通道 1 选择** 设置通道的输出方向与信号的选择，当 GP32C4T_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(只工作在 GP32C4T_SMCON 寄存器的 TSSEL 位选择为 ITn 时)

21. 6. 2. 12 捕获或比较模式寄存器 2(GP32C4T_CHMR2)

捕获或比较模式寄存器 2(GP32C4T_CHMR2)
偏移地址:0x2C
复位值:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
							CH4MOD2								CH3MOD2	CH4OCLREN	CH4MOD<2:0>			CH4PEN	CH4FEN	CC4SSEL<1:0>		CH3OCLREN	CH3MOD<2:0>			CH3PEN	CH3FEN	CC3SSEL<1:0>	
																I4FLT<3:0>				I4PRES<1:0>				I3FLT<3:0>				I3PRES<1:0>		

输出比较模式

—	Bits 31-25	—	—
CH4MOD2	Bit 24	R/W	**输出比较通道4模式** 参照CH3MOD描述
—	Bits 23-17	—	—
CH3MOD2	Bit 16	R/W	**输出比较通道3模式** 参照CH3MOD描述
CH4OCLREN	Bit 15	R/W	**输出比较通道 4 清除开启** 参照 CH3OCLREN 描述
CH4MOD	Bit 14-12	R/W	**输出比较通道 4 模式** 参照 CH3MOD 描述
CH4PEN	Bit 11	R/W	**输出比较通道 4 预装载开启** 参照 CH3PEN 描述
CH4FEN	Bit 10	R/W	**输出比较通道 4 快速开启** 参照 CH3FEN 描述
CC4SSEL	Bit 9-8	R/W	**捕获或比较通道 4 选择** 设置通道的输出方向与信号的选择，当 GP32C4T_CCEP 寄存器的 CC4EN 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 I4 10:通道设置为输入，捕获源为 I3 11:通道设置为输入，捕获源为 ITn(只工作在 GP32C4T_SMCON 寄存器的 TSSEL 位选择为 ITn 时)
CH3OCLREN	Bit 7	R/W	**输出比较通道 3 清除开启** 参照 CH1OCLREN 描述
CH3MOD	Bit 6-4	R/W	**输出比较通道 3 模式**

			参照 CH1MOD 描述。但多了以下两个模式： 01010: 比较脉冲模式-输出产生一个比较脉冲，其脉冲宽度可由 GP32C4T_ENCR 的 PWPRES 位与 PW 位决定 01011: 方向输出模式-将 GP32C4T_CON1 寄存器的 DIRSEL 位映射到通道输出。
CH3PEN	Bit 3	R/W	**输出比较通道 3 预装载开启** 参照 CH1PEN 描述
CH3FEN	Bit 2	R/W	**输出比较通道 3 快速开启** 参照 CH1FEN 描述
CC3SSEL	Bit 1-0	R/W	**捕获或比较通道 3 选择** 设置通道的输出方向与信号的选择，当 GP32C4T_CCEP 寄存器的 CC3EN 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 I3 10:通道设置为输入，捕获源为 I4 11:通道设置为输入，捕获源为 ITn(只工作在 GP32C4T_SMCON 寄存器的 TSSEL 位选择为 ITn 时)

输入捕获模式

—	Bits 31-16	—	—
I4FLT	Bit 15-12	R/W	**输入捕获通道4滤波器** 参照I3FLT描述
I4PRES	Bit 11-10	R/W	**输入捕获通道 4 预分频器** 参照 IC3PRES 描述
CC4SSEL	Bit 9-8	R/W	**捕获或比较通道 4 选择** 设置通道的输出方向与信号的选择，当 GP32C4T_CCEP 寄存器的 CC4EN 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 I4 10:通道设置为输入，捕获源为 I3 11:通道设置为输入，捕获源为 ITn(只工作在 GP32C4T_SMCON 寄存器的 TSSEL 位选择为 ITn 时)
I3FLT	Bit 7-4	R/W	**输入捕获通道 3 滤波器**

			参照 I1FLT 描述
I3PRES	Bit 3-2	R/W	**输入捕获通道 3 预分频器** 参照 IC1PRES 描述
CC3SSEL	Bit 1-0	R/W	**捕获或比较通道 3 选择** 设置通道的输出方向与信号的选择，当 GP32C4T_CCEP 寄存器的 CC3EN 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 I3 10:通道设置为输入，捕获源为 I4 11:通道设置为输入，捕获源为 ITn(只工作在 GP32C4T_SMCON 寄存器的 TSSEL 位选择为 ITn 时)

21. 6. 2. 13 捕获或比较开启极性寄存器(GP32C4T_CCEP)

此控制寄存器可被两个地址共同存取

捕获或比较开启寄存器(GP32C4T_CCEP)																												
偏移地址:0x30、0xA0																												
复位值:0x0000 0000																												
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
—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	CC4NPOL	—	CC4POL	CC4EN	CC3NPOL	—	CC3POL	CC3EN	CC2NPOL	—	CC2POL	CC2EN	CC1NPOL	—	CC1POL	CC1EN

—	Bits 31-16	—	—
CC4NPOL	Bit 15	R/W	**捕获或比较通道4互补输出极性** 参照CC1NPOL描述
—	Bit 14	—	—
CC4POL	Bit 13	R/W	**捕获或比较通道 4 输出极性** 参照 CC1POL 描述
CC4EN	Bit 12	R/W	**捕获或比较通道 4 输出开启** 参照 CC1EN 描述
CC3NPOL	Bit 11	R/W	**捕获或比较通道 3 互补输出极性** 参照 CC1NPOL 描述
—	Bit 10	—	—
CC3POL	Bit 9	R/W	**捕获或比较通道 3 输出极性** 参照 CC1POL 描述
CC3EN	Bit 8	R/W	**捕获或比较通道 3 输出开启**

			参照 CC1EN 描述
CC2NPOL	Bit 7	R/W	**捕获或比较通道 2 互补输出极性** 参照 CC1NPOL 描述
—	Bit 6	—	—
CC2POL	Bit 5	R/W	**捕获或比较通道 2 输出极性** 参照 CC1POL 描述
CC2EN	Bit 4	R/W	**捕获或比较通道 2 输出开启** 参照 CC1EN 描述
CC1NPOL	Bit 3	R/W	**捕获或比较通道 1 互补输出极性** **通道 CH1 设置为输出:** 此位必须保持为 0 **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。参照 CC1POL 描述。
—	Bit 2	—	—
CC1POL	Bit 1	R/W	**捕获或比较通道 1 输出极性** **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 与 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
CC1EN	Bit 0	R/W	**捕获或比较通道 1 输出开启** **通道 CH1 设置为输出:** 0:关闭- CH1 无效(此时引脚不由定时器控制) 1:开启- CH1 为对应输出引脚上的输出信号。 **通道 CH1 设置为输入:** 0:捕获关闭 1:捕获开启

21. 6. 2. 14 计数寄存器(GP32C4T_COUNT)

计数寄存器(GP32C4T_COUNT)
偏移地址:0x34
复位值:0x0000 0000

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

UPDRIF	Bits 31	R/W	**更新原始中断状态映射** 设置GP32C4T_CON1寄存器的UPDFREMAP位为1时: 该位为映射GP32C4T_RIF寄存器的UPD位的状态，该位只能读取，表示计数器发生更新事件。 设置GP32C4T_CON1寄存器的UPDFREMAP位为0时: 该位为CNTV[31]
CNTV	Bits 30-0	R/W	**计数器数值**

21. 6. 2. 15 时钟预分频寄存器(GP32C4T_PRES)

时钟预分频寄存器(GP32C4T_PRES)																															
偏移地址:0x38																															
复位值:0x0000 0000																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																PSCV<15:0>															

—	Bits 31-16	—	—
PSCV	Bits 15-0	R/W	**预分频数值** 当计数器时钟频率等于f_{INT_CLK}/(PSCV<15:0> + 1)时计数器递增或递减。在更新事件产生时，将PSCV数值被载入预装载寄存器中

21. 6. 2. 16 自动重载寄存器(GP32C4T_AR)

自动重载寄存器(GP32C4T_AR)																															
偏移地址:0x3C																															
复位值:0x FFFF FFFF																															
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

ARV<31:0>

ARV	Bits 31-0	R/W	**自动装载数值** 设置计数器的递增计数的边界或递减计数的重载值

21. 6. 2. 17 通道捕获或比较寄存器 1(GP32C4T_CCVAL1)

通道捕获或比较寄存器 1(GP32C4T_CCVAL1)																													
偏移地址:0x44																													
复位值:0x0000 0000																													
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCRV1<31:0>																													

CCRV1	Bits 31-0	R/W	**捕获或比较数值 1** **通道 CH1 配置为输出:** CCRV1 是捕获或比较寄存器的预装载值。 开启 GP32C4T_CHMR1 寄存器中的 CH1PEN 预装载功能时，CCRV1 具备缓冲，当发生更新事件后，将预装载值载入到缓冲寄存器中。 CCRV1 的值会与 GP32C4T_COUNT 中的值进行比较，其结果会反应在 CH1REF。 **通道 CH1 配置为输入:** CCRV1为由先前发生输入捕获事件(I1)时的计数器数值。为只读状态，禁止写入动作

21. 6. 2. 18 通道捕获或比较寄存器 2(GP32C4T_CCVAL2)

通道捕获或比较寄存器 2(GP32C4T_CCVAL2)																															
偏移地址:0x48																															
复位值:0x0000 0000																															
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
CCRV2<31:0>																															

CCRV2	Bits 31-0	R/W	**捕获或比较数值2** 参照CCRV1描述

21. 6. 2. 19 通道捕获或比较寄存器 3(GP32C4T_CCVAL3)

通道捕获或比较寄存器 3(GP32C4T_CCVAL3)																															
偏移地址:0x4C																															
复位值:0x0000 0000																															
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
CCRV3<31:0>																															

CCRV3	Bits 31-0	R/W	**捕获或比较数值3** 参照CCRV1描述

21. 6. 2. 20 通道捕获或比较寄存器 4(GP32C4T_CCVAL4)

通道捕获或比较寄存器 4(GP32C4T_CCVAL4)																															
偏移地址:0x50																															
复位值:0x0000 0000																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCRV4<31:0>																															

CCRV4	Bits 31-0	R/W	**捕获或比较数值4** 参照CCRV1描述

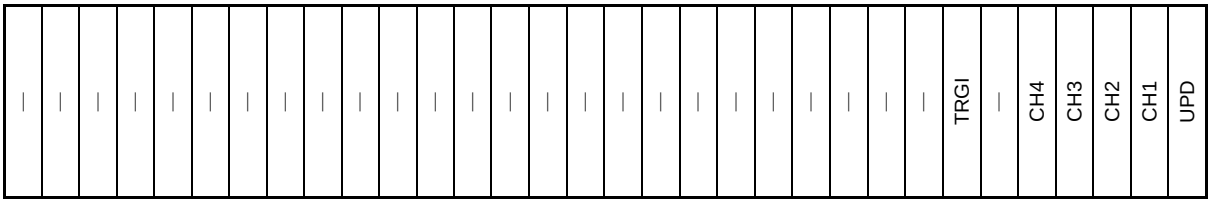
21. 6. 2. 21 DMA事件开启寄存器(GP32C4T_DMAEN)

DMA 事件开启寄存器(GP32C4T_DMAEN)																												
偏移地址:0x58																												
复位值:0x0000 0000																												
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
																									TRGI		CH4	CH3	CH2	CH1	UPD

—	Bits 31-7	—	—
TRGI	Bit 6	R/W	**触发事件的DMA请求开启** 0: 触发事件的DMA请求关闭 1: 触发事件的DMA请求开启
—	Bits 5	—	—
CH4	Bit 4	R/W	**通道4捕获或比较匹配事件的DMA请求开启** 0: 通道4捕获或比较匹配事件的DMA请求关闭 1: 通道4捕获或比较匹配事件的DMA请求开启
CH3	Bit 3	R/W	**通道3捕获或比较匹配事件的DMA请求开启** 0: 通道3捕获或比较匹配事件的DMA请求关闭 1: 通道3捕获或比较匹配事件的DMA请求开启
CH2	Bit 2	R/W	**通道2捕获或比较匹配事件的DMA请求开启** 0: 通道2捕获或比较匹配事件的DMA请求关闭 1: 通道2捕获或比较匹配事件的DMA请求开启
CH1	Bit 1	R/W	**通道1捕获或比较匹配事件的DMA请求开启** 0: 通道1捕获或比较匹配事件的DMA请求关闭 1: 通道1捕获或比较匹配事件的DMA请求开启
UPD	Bit 0	R/W	**更新事件的 DMA 请求开启** 0: 更新事件的 DMA 请求关闭 1: 更新事件的 DMA 请求开启

21. 6. 2. 22 编码控制寄存器(GP32C4T_ENCR)

编码控制寄存器(GP32C4T_ENCR)																												
偏移地址:0x6C																												
复位值:0x0000 0000																												
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



—	Bits 31-27	—	—
PWPRES	Bit 26-24	R/W	**脉冲宽度预分频值** 公式如下: $t_{PWG} = (2^{(PWPRES[2:0])}) \times t_{INT\_CLK}$
PW	Bit 23-16	R/W	**脉冲宽度** 公式如下: $t_{PW} = PW[7:0] \times t_{PWG}$
—	Bits 15-8	—	—
IDXPOS	Bit 7-6	R/W	**Index位置** 在正交编码器模式下(GP32C4T_SMCON寄存器的SMODS = 0001, 0010, 0011, 1110, 1111), 此位用来指示Index信号在哪个AB相位为有效输入，判定为有效输入时重置计数器 00: 当AB相位 = 00时，侦测到有效的Index输入会重置计数器 01: 当AB相位 = 01时，侦测到有效的Index输入会重置计数器 10: 当AB相位 = 10时，侦测到有效的Index输入会重置计数器 11: 当AB相位 = 11时，侦测到有效的Index输入会重置计数器 在时钟加方向模式与定向时钟模式下(GP32C4T_SMCON寄存器的SMODS = 1010, 1011, 1100, 1101)，IDXPOS[1]为无效位 x0: 当时钟低电平时，侦测到有效的Index输入会重置计数器 x1: 当时钟高电平时，侦测到有效的Index输入会重置计数器
FIDX	Bit 5	R/W	**首次Index事件检测** 0: 每一个Index事件都会重置计数器 1: 只让第一次的Index事件重置计数器
—	Bits 4-3	—	—
IDXDIR	Bit 2-1	R/W	**特定计数方向下侦测Index事件**

			00: Index事件在递增、递减计数下都为有效 01: Index事件只在递增计数下有效 10: Index事件只在递减计数下有效 11: 保留 注: 当Index功能关闭时才能变更IDXDIR设定
IDXEN	Bit 0	R/W	**Index 检测功能开启** 侦测到 Index 有效事件时会重置计数器 0: Index 功能关闭 1: Index 功能开启

21. 6. 2. 23 通道输入来源选择寄存器 (GP32C4T_CHISEL)

通道输入来源选择寄存器 (GP32C4T_CHISEL)

偏移地址:0x70

复位值:0x0000 0000

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
				I4SEL<3:0>								I3SEL<3:0>								I2SEL<3:0>								I1SEL<3:0>			

—	Bits 31-28	—	—
I4SEL	Bit 27-24	R/W	**CH4输入来源选择** 0000: CH4引脚输入 其他: 参考引脚输入源章节
—	Bits 23-20	—	—
I3SEL	Bit 19-16	R/W	**CH3输入来源选择** 0000: CH3引脚输入 其他: 参考引脚输入源章节
—	Bits 15-12	—	—
I2SEL	Bit 11-8	R/W	**CH2输入来源选择** 0000: CH2引脚输入 其他: 参考引脚输入源章节
—	Bits 37-4	—	—
I1SEL	Bit 3-0	R/W	**CH1 输入来源选择** 0000: CH1 引脚输入 其他: 参考引脚输入源章节

21. 6. 2. 24 复用功能寄存器 1 (GP32C4T_AFR1)

复用功能寄存器 1 (GP32C4T_AFR1)																														
偏移地址:0x74																														
复位值:0x0000 0000																														
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
														ETRSEL<3:0>																	

—	Bits 31-18	—	—
ETRSEL	Bit 17-14	R/W	**ETR输入来源选择** 0000: ETR引脚输入 其他: 参考引脚输入源章节
—	Bits 13-0	—	—

21. 6. 2. 25 复用功能寄存器 2 (GP32C4T_AFR2)

复用功能寄存器 2(GP32C4T_AFR2																														
偏移地址:0x78																														
复位值:0x0000 0000																														
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
													OCLRSEL<2:0>																		

—	Bits 31-19	—	—
OCLRSEL	Bit 18-6	R/W	**外部清除来源选择** 000: CMP1 001: CMP2 其他: 保留
—	Bits 15-0	—	—

第22章 通用定时器 16 位 2 通道 (GP16C2T)

22. 1 概述

通用定时器 16 位 2 通道(GP16C2T)是一个设置灵活的定时器模块，它包含一个 16 位计数器，具有定时、计数、脉冲输入信号测量(输入捕获)、产生特定 PWM 波形(输出比较)与可设置死区时间的互补输出等功能。

22. 2 特性

- 一种 16 位自动重载计数器模式
  - 递增
- 16 位可配置预分频器，可在定时器运行时对计数器工作时钟进行 1 到 65536 之间的任意分频
- 带有两个独立通道，每个通道支持以下功能
  - 输入捕获
  - 输出比较
  - PWM 输出(边沿对齐)
  - 单脉冲输出
- 通道 1 支持互补输出，可设置死区时间
- 同步电路用于外部信号控制定时器及内部互联多个定时器
- 重复计数器，用于在给定数目的计数周期后更新定时器寄存器
- 支持刹车输入功能，并可设置刹车后定时器输出状态
- 下列事件支持产生中断与 DMA 请求:
  - 更新事件:计数器上溢，计数器初始化(通过软件或内部与外部触发)
  - 触发事件:计数器开启、停止、初始化或通过内部与外部触发计数
  - 换向事件(COM)
  - 输入捕获
  - 输出比较
  - 刹车输入

22. 3 结构图

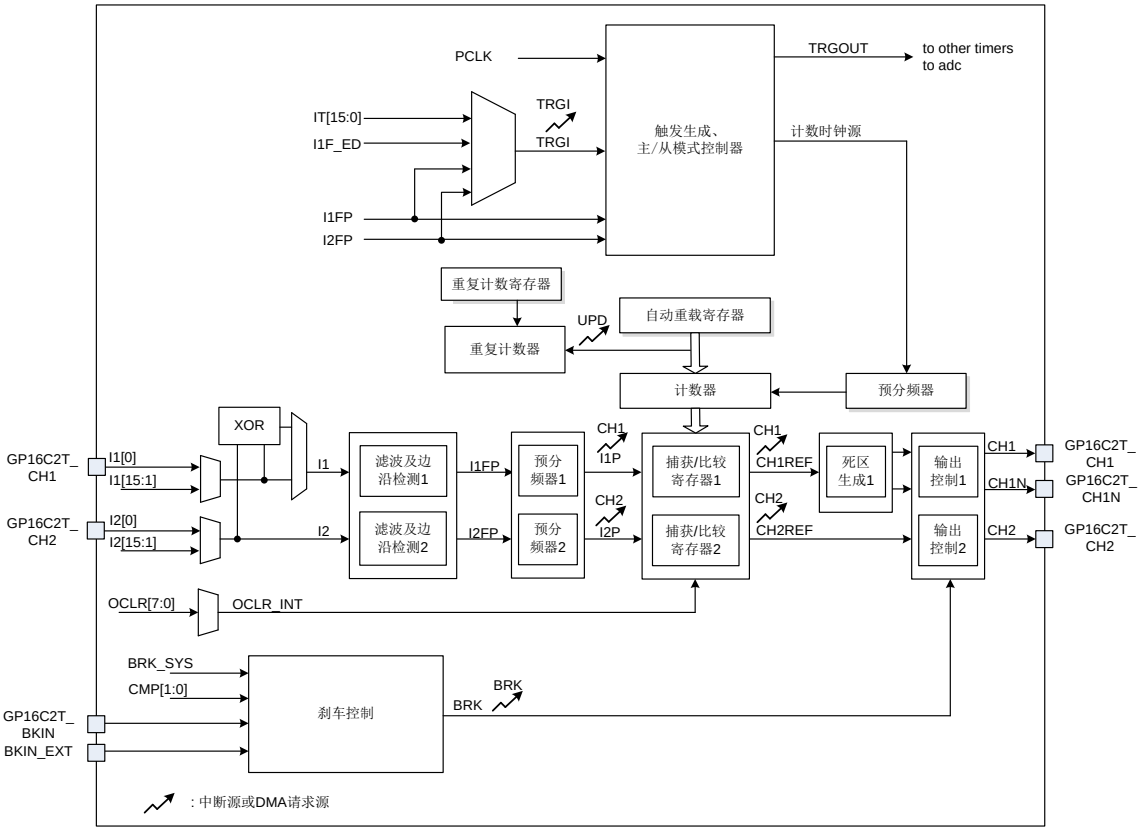


图 23-1　GP16C2T 定时器结构框图

22. 4 引脚选择与内部连接表

22. 4. 1 引脚输入源

设置 **GP16C2T_CHISEL** 寄存器，可选择通道 1 与通道 2 引脚输入来源，如下表 ：

I1SEL	GP16C2T1	GP16C2T 2
0000	GP16C2T1_CH1	GP16C2T2_CH1
0010	CMP1	MCO
0011	CMP2	HOSC/32
0110	保留	LRC
其余	保留	

表 23-1 通道 1 引脚输入来源

I2SEL	GP16C2T1	GP16C2T2
0000	GP16C2T1_CH2	GP16C2T2_CH2
0001	CMP1	CMP1
0010	CMP2	CMP2
其余	保留	

表 23-2 通道 2 引脚输入来源

22. 4. 2 内部触发连接表

从定时器	IT0(TSSEL=00000)	IT1(TSSEL=00001)	IT2(TSSEL=00010)	IT3(TSSEL=00011)	IT4(TSSEL=01000)	IT5(TSSEL=01001)	IT6(TSSEL=01010)	IT7(TSSEL=01011)
GP16C2T1	AD16C6T1	GP32C4T1	GP32C4T2	保留	保留	AD16C6T2	保留	GP16C2T2

表 23-3 GP16C2T1 内部触发连接表

从定时器	IT0(TSSEL=00000)	IT1(TSSEL=00001)	IT2(TSSEL=00010)	IT3(TSSEL=00011)	IT4(TSSEL=01000)	IT5(TSSEL=01001)	IT6(TSSEL=01010)	IT7(TSSEL=01011)
GP16C2T2	AD16C6T1	GP32C4T1	GP32C4T2	保留	保留	AD16C6T2	GP16C2T1	保留

表 23-4 GP16C2T2 内部触发连接表

22. 5 功能描述

22. 5. 1 定时单位

定时器包含一个 16 位的计数器(**GP16C2T_COUNT**)，计数时钟由预分频寄存器(**GP16C2T_PRES**)进行分频。计数周期由自动重载计数器(**GP16C2T_AR**)设定。重复计数寄存器则可指定计数周期数目(**GP16C2T_REPAR**)。

自动重载寄存器(**GP16C2T_AR**)是一个可缓冲的寄存器。设置 **GP16C2T_CON1** 寄存器的 ARPEN 位为 0 时，关闭 **GP16C2T_AR** 寄存器缓冲功能，写入 **GP16C2T_AR** 寄存器的值会被立即反应到缓冲寄存器中；而设置 ARPEN 位为 1 时，**GP16C2T_AR** 寄存器具有缓冲功能，只有当产生更新事件(UPD)时，**GP16C2T_AR** 寄存器的重载值才会被更新到缓冲寄存器中。

设置 **GP16C2T_CON1** 寄存器的 DISUE 位为 0 时，计数器递增计数达到上溢值时会产生更新事件(UPD)。另外可以通过 **GP16C2T_SGE** 寄存器的 SGUPD 位为 1 产生软件更新事件。设置 **GP16C2T_CON1** 寄存器的 CNTEN 为 1 时，计数器开始计数。

注:计数器在设置 CNTEN 位为 1 后，在 1 个定时器时钟周期后开始计数。

预分频器可对定时器工作时钟进行 **GP16C2T_PRES** 寄存器数值+1 次分频。由于 **GP16C2T_PRES** 是一个可缓冲寄存器，因此定时器运行时可以对该寄存器进行修改，修改值在下次更新事件(UPD)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。

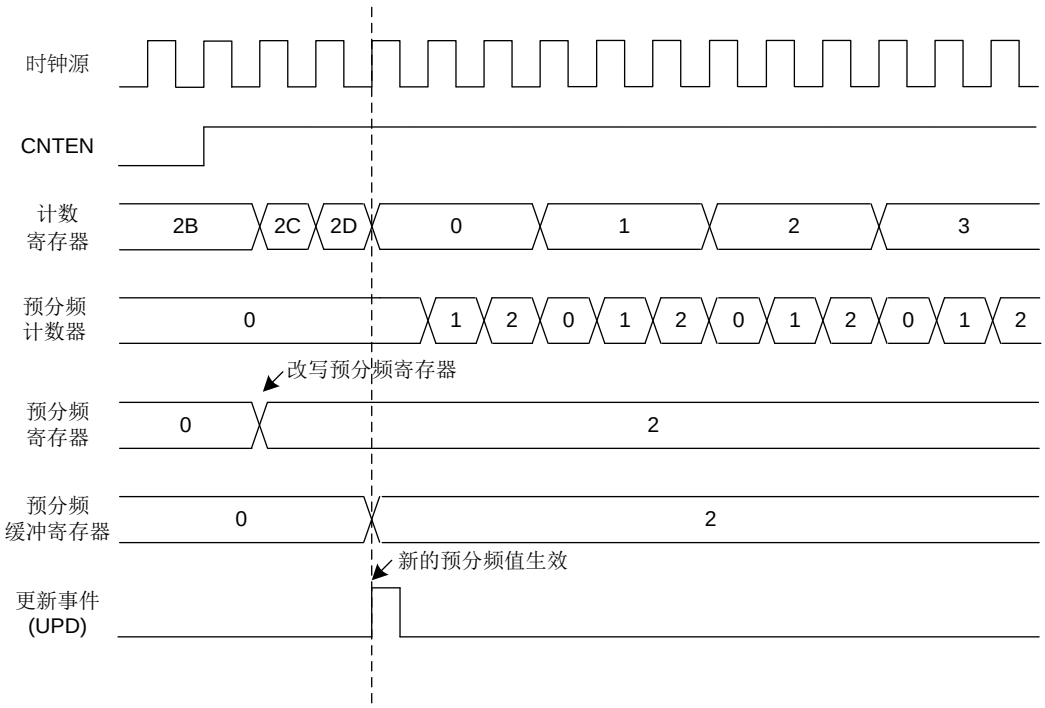


图 23-2 从 1 分频变为 3 分频时的计数时序图

22. 5. 2 重复计数器

重复计数器用于控制发生多少次上溢后产生更新事件。

重复计数器在下列情况下递减:

- 递增模式时计数器的每次上溢

GP16C2T_REPAR 寄存器是一个可缓冲寄存器。当由软件(设置 **GP16C2T_SGE** 寄存器的 SGUPD 位为 1)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，重复寄存器的数值会立即被更新到重复缓冲寄存器。

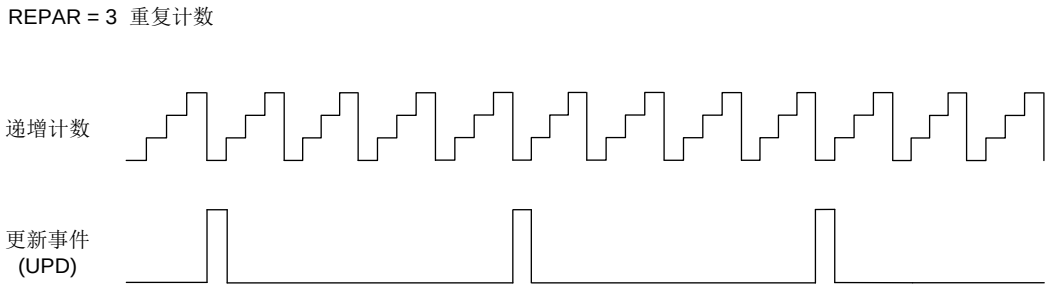


图 23-3 重复计数器工作模式

注:设置 GP16C2T_SGE 寄存器的 SGUPD 位为 1，也可以产生更新事件。

22. 5. 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(TRGI)。

22. 5. 3. 1 内部时钟源(INT_CLK)

若从模式控制器被关闭(**GP16C2T_SMCON** 寄存器的 SMODS 位为 0000b)，则 **GP16C2T_CON1** 寄存器的 CNTEN 位与 GP16C2T_SGE 寄存器的 SGUPD 位为控制位，这些位只能软件修改(SGUPD 位除外，仍由硬件自动清除)。一旦设置 CNTEN 位为 1，预分频器就由内部 INT_CLK 提供时钟。

下图给出了正常模式下没有分频控制电路和递增计数的情况。

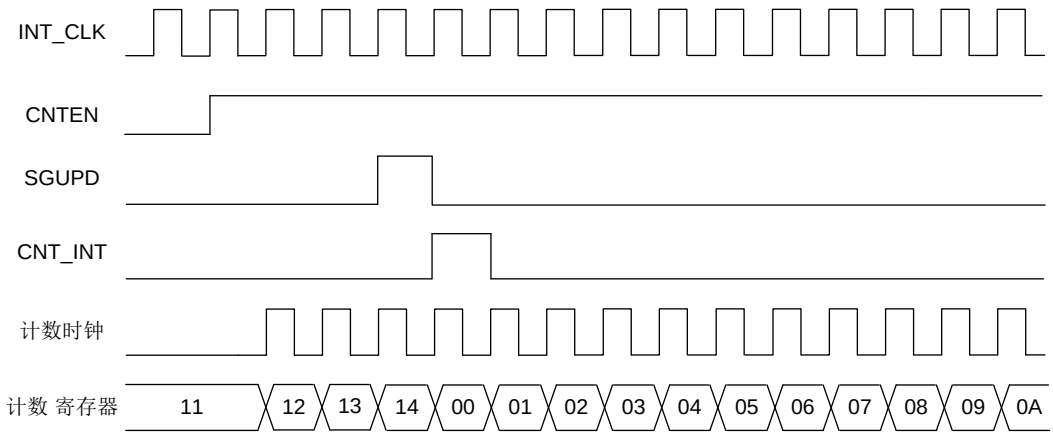


图 23-4 采用内部时钟计数

22. 5. 3. 2 外部时钟源 1

GP16C2T_SMCON 寄存器的 SMODS 位为 0111b 时，可选择外部时钟源 1。计数器可根据选定输入信号的上升沿或下降沿计数。

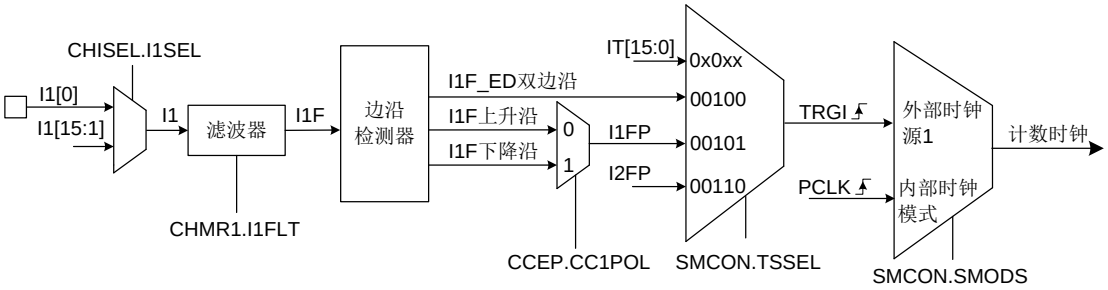


图 23-5 外部时钟连接

设置计数器外部时钟源为 I1 输入，并在 I1 上升沿时计数，步骤如下:

1. 设置 **GP16C2T_CHISEL** 寄存器的 CH1SEL 位为 0000b，选择 I1 由 IO 输入(默认值皆为 IO 输入，后续不再特别描述)。
2. 设置 **GP16C2T_CHMR1** 寄存器 CC1SSEL 位为 01b，让通道 1 为 I1 输入。
3. 设置 **GP16C2T_CHMR1** 寄存器的 I1FLT 位，输入滤波器时间(若没有滤波器需求，维持 I1FLT 位为 0000b)。
4. 设置 **GP16C2T_CCEP** 寄存器的 CC1NPOL 位为 0、CC1POL 位为 0，选择极性为上升沿。
5. 设置 **GP16C2T_SMCON** 寄存器的 TSSEL 位为 00101b，选择外部时钟源为 I1。
6. 设置 **GP16C2T_SMCON** 寄存器的 SMODS 位为 0111b，选择外部时钟模式 1。
7. 设置 **GP16C2T_CON1** 寄存器的 CNTEN 位为 1，开启计数器。

当 I1 上出现一次上升沿时，计数器计数一次且设置 TRGI 标志位为 1。I1 上升沿与实际时钟间的延迟，取决于 I1 输入的同步电路。

设置计数器外部时钟源为 IT1 输入，并在 IT1 上升沿时计数，步骤如下:

1. 设置 **GP16C2T_SMCON** 寄存器的 TSSEL 位为 00001b，选择外部时钟源为 IT1。
2. 设置 **GP16C2T_SMCON** 寄存器的 SMODS 位为 0111b，选择外部时钟模式 1。
3. 设置 **GP16C2T_CON1** 寄存器的 CNTEN 位为 1，开启计数器。

IT1 产生上升沿时，计数器计数一次。

注:以 GP16C2T1 为例，IT1 即为 GP32C4T1 的 TRGOUT，详细可参考内部触发连结表。

22. 5. 4 计数模式

22. 5. 4. 1 递增计数模式

设置 **GP16C2T_CON1** 寄存器的 DIRSEL 位为 0 时，定时器设置为递增模式，计数器从 0 开始递增，直至 **GP16C2T_AR** 寄存器数值；然后从 0 重新开始计数并产生一个更新事件(UPD)。设置 **GP16C2T_REPAR** 寄存器不为 0 时，则在 **GP16C2T_REPAR**+1 次计数后产生更新事件。

设置 **GP16C2T_SGE** 寄存器的 SGUPD 位为 1(通过软件或使用从模式控制器)同样会产生更新事件，并让计数器和预分频器都重新从 0 开始计数。

通过软件设置 **GP16C2T_CON1** 寄存器中的 DISUE 位为 1 可关闭更新事件(UPD)产生。可以避免在写入预装载寄存器数值时产生更新事件(UPD)更新缓冲寄存器。在设置 DISUE 位为 0 之前都不会产生更新事件(UPD)。

此外，**GP16C2T_CON1** 寄存器中的 UERSEL 位为 1 时，设置 SGUPD 位为 1 会产生更新事件(UPD)，但不会更新更新标志位(**GP16C2T_RIF** 寄存器的 UPD 位)，也不会产生中断或 DMA 请求。在这个配置下，发生捕获事件时将计数器清零，将不会同时产生更新中断和捕获中断。

发生更新事件(UPD)时，所有预装载寄存器会更新到缓冲寄存器:

- 更新 **GP16C2T_REPAR** 寄存器数值到缓冲寄存器
- 更新 **GP16C2T_AR** 寄存器数值到缓冲寄存器
- 更新 **GP16C2T_PRES** 寄存器数值到缓冲寄存器

下图为设置 **GP16C2T_AR** 寄存器为 16h，预分频设为 2 分频时的计数器时序。

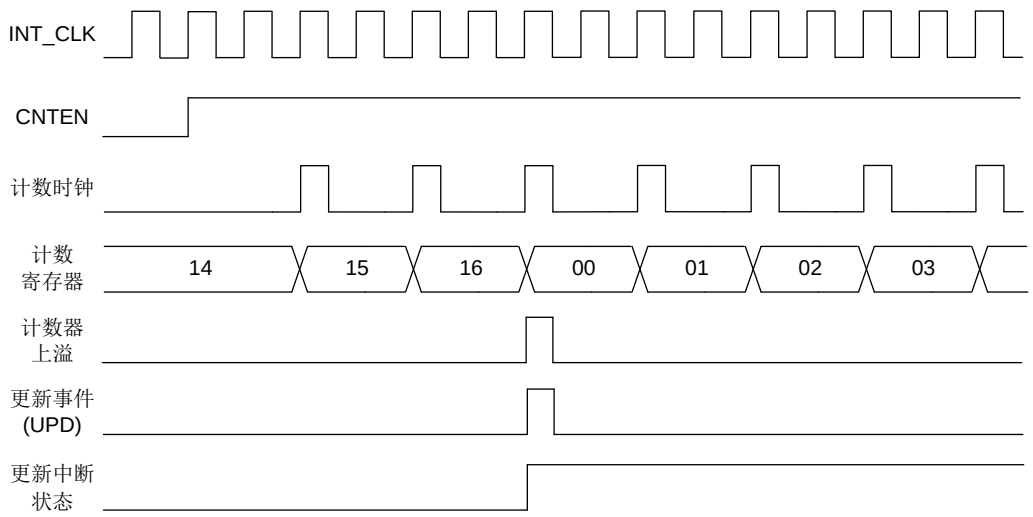


图 23-6 计数器递增计数时序图

下图为设置 **GP16C2T_AR** 寄存器从 F5h 改成 16h，分别在 ARPEN 为 0 或 1 时的计数器时序。

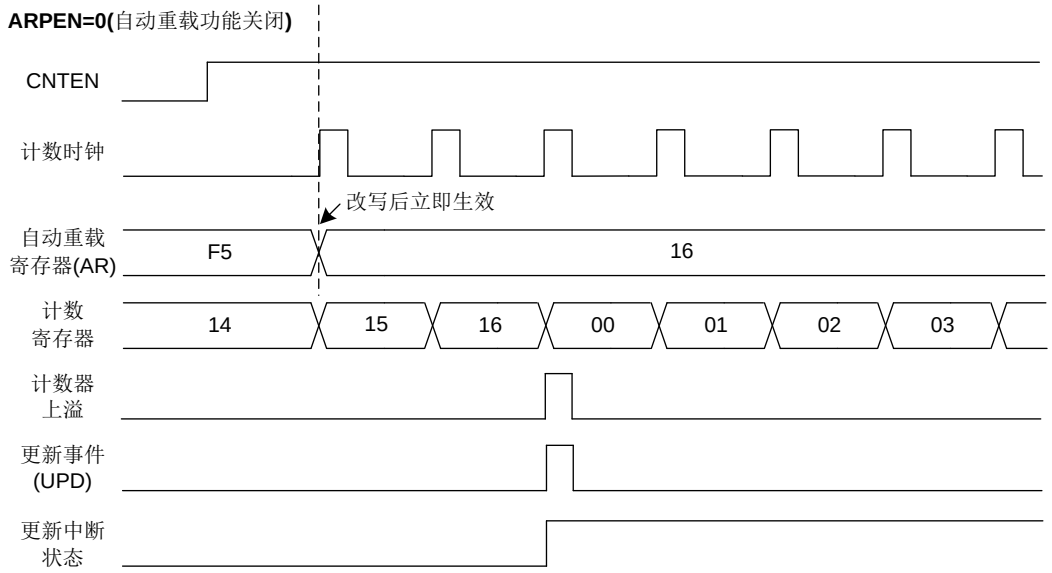


图 23-7　设置 ARPEN 位为 0 时计数器时序图

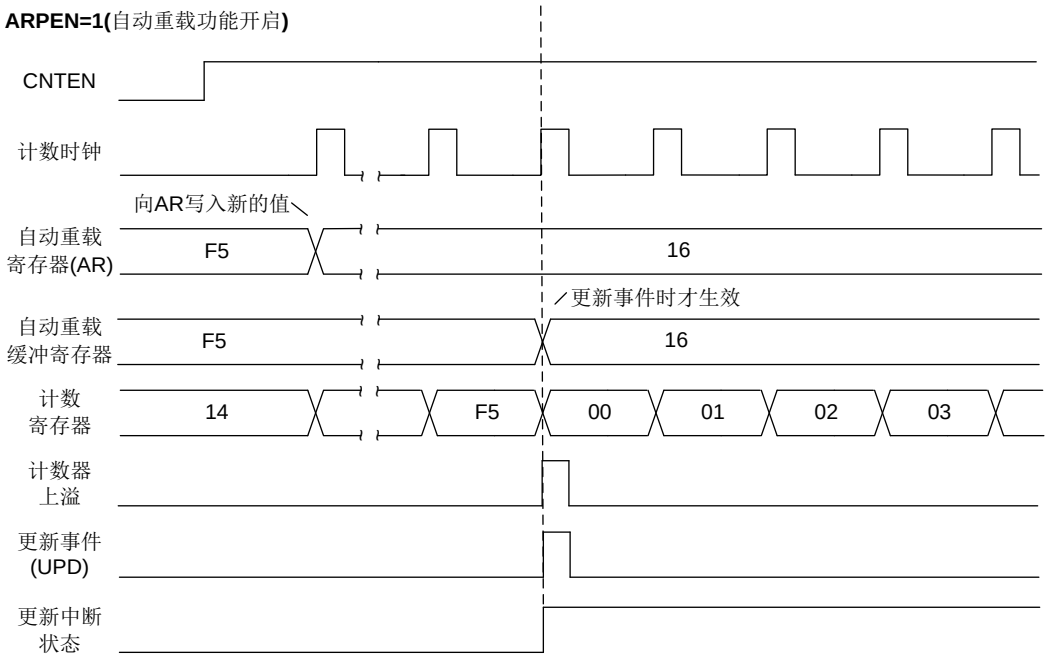


图 23-8　设置 ARPEN 位为 1 时计数器时序图

22. 5. 5 捕获或比较通道

每个捕获或比较通道都包含一个捕获或比较预装载寄存器(包含缓冲寄存器)、一个输入捕获电路和一个输出比较电路。读写操作都是通过预装载寄存器进行的。在捕获模式下，实际的捕获是在缓冲寄存器中完成的，然后缓冲寄存器中的值被复制到预装载寄存器中。在比较模式下，预装载寄存器的值被复制到缓冲寄存器中，并与计数器进行比较。

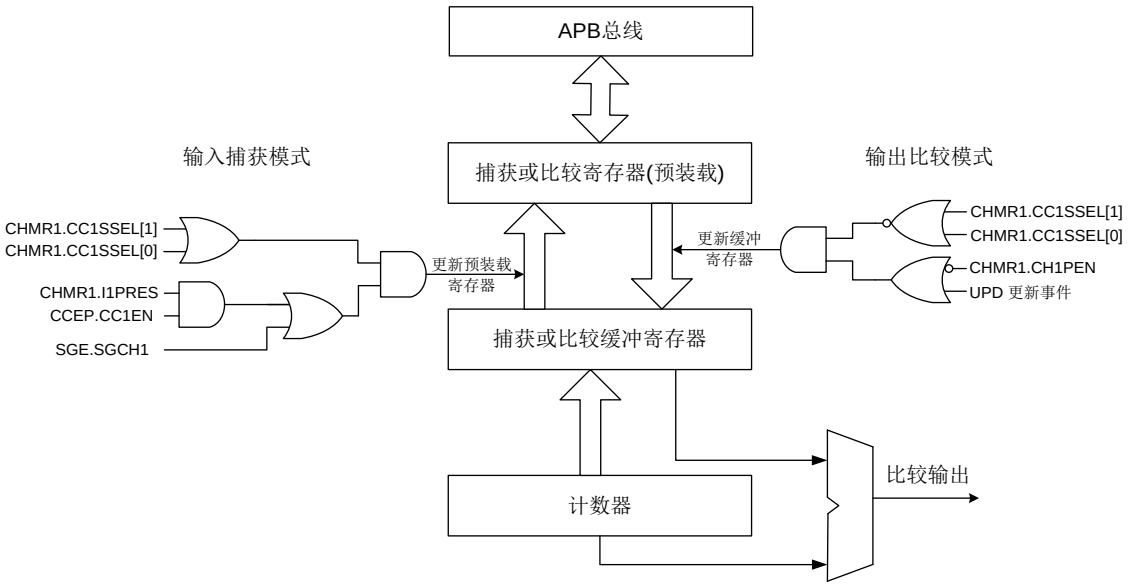


图 23-9 捕获或比较通道结构图(通道 1 为例)

22. 5. 5. 1 输入捕获电路

输入捕获电路会对 In 输入端的信号进行采样，经过数字滤波器后产生 InF 信号，接着通过边沿检测器产生 InF 边沿信号，包括上升沿、下降沿以及双边沿。最后，根据 **GP16C2T_CCEP** 寄存器中的 CCnNPOL/CCnPOL 极性选择位，生成 InFP 信号。这个信号可以输出到从模式控制器的触发输入(TRGI)，或作为捕获输入源(InP)之一，而且还需要通过预分频器进行预处理。

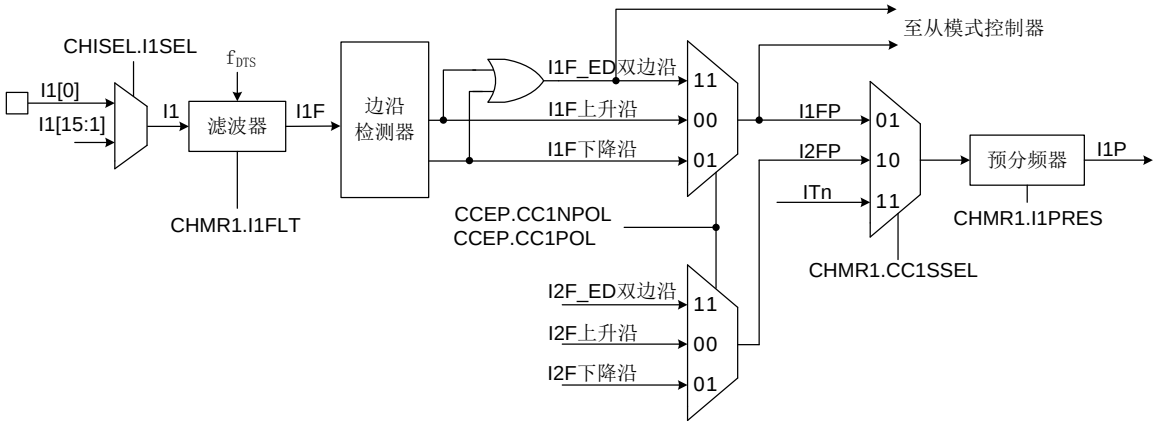


图 23-10 输入捕获电路(通道 1 为例)

22. 5. 5. 2　　输出比较电路

输出部分会根据 **GP16C2T_CHMRn** 寄存器中 CHnMOD 位的配置，产生一个输出比较参考信号 CHnREF(高电平有效)。最终输出信号的极性则由 **GP16C2T_CCEP** 寄存器中的 CCnPOL/CCnNPOL 位所决定。

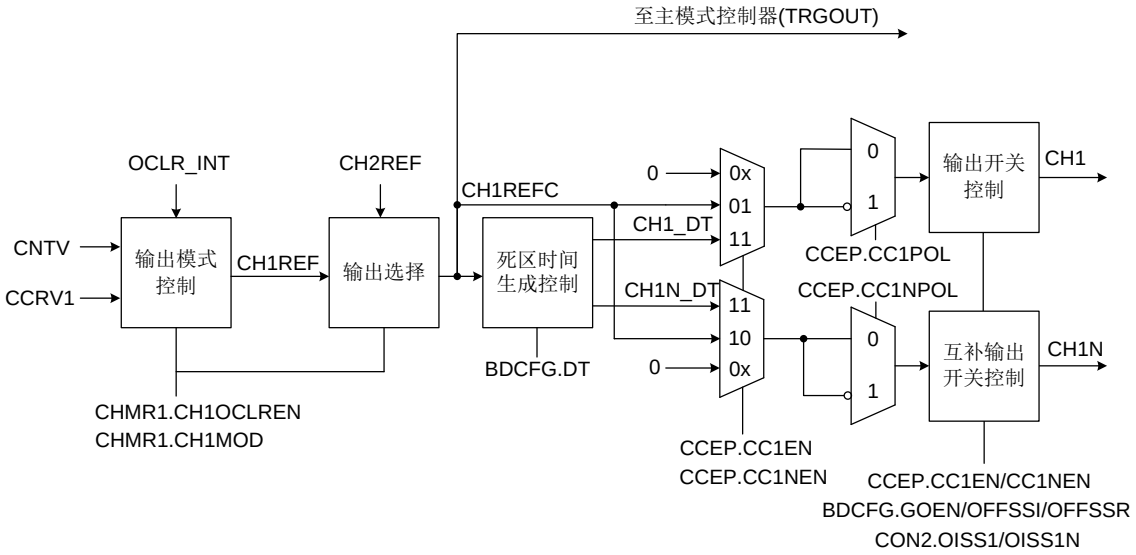


图 23-11　输出比较电路(通道 1 为例)

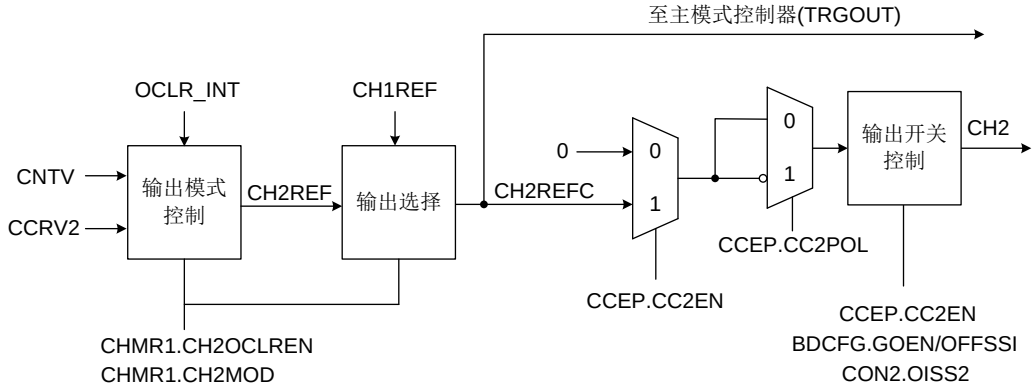


图 23-12　输出比较电路(通道 2 为例)

22. 5. 6 输入捕获模式

在输入捕获模式下，当 In 上检测到有效边沿变化时，计数器数值就会被锁存到捕获或比较寄存器(GP16C2T_CCVALn)中。此时，**GP16C2T_RIF** 寄存器中相应的 CHn 标志位会被设置为 1，同时触发中断或 DMA 请求(如果已开启)。

若相应的 CHn 标志位已经为 1，则当再次发生捕获事件时，相应的过捕获 CHnOV 标志位也会被设定为 1，表示曾经发生过捕获事件。

藉由软件将 **GP16C2T_ICR** 寄存器中的 CHn 位与 CHnOV 位设定为 1，可清除 **GP16C2T_RIF** 寄存器中相应的 CHn 与 CHnOV 标志位。

以下为以 I1 输入上升沿作为捕获输入时的流程:

1. 设置 **GP16C2T_CHMR1** 寄存器的 CC1SSEL 位为 01b，选择 I1 为有效输入端。只要 CC1SSEL 不为 00b，通道就会被设置成输入，且 **GP16C2T_CCVAL1** 寄存器为只读。
2. 设置 **GP16C2T_CHMR1** 寄存器的 I1FLT 位为 0011b，选择输入滤波器的持续时间，当 I1 检测到新的电平时，会进行连续 8 次采样，确认电平变化的有效性。
3. 设置 **GP16C2T_CCEP** 寄存器的 CC1NPOL 位为 0、CC1POL 位为 0，选择 I1 通道上升沿有效。
4. 设置 **GP16C2T_CHMR1** 寄存器的 I1PRES 位为 00b，关闭捕获预分频器，让每次有效上升沿皆执行捕获操作。
5. 如有需要，设置 **GP16C2T_IER** 寄存器的 CH1 位为 1，开启中断请求。设置 **GP16C2T_DMAEN** 寄存器的 CH1 位为 1，开启 DMA 请求。
6. 设置 **GP16C2T_CCEP** 寄存器的 CC1EN 位为 1，开启捕获计数器。

当发生输入捕获时:

1. **GP16C2T_CCVAL1** 寄存器获取计数器当前的值。
2. **GP16C2T_RIF** 寄存器的 CH1 位被设置。如果至少两个连续捕获发生，且标志位未被清除，则 CH1OV 位也被设置。
3. 中断的产生取决于 **GP16C2T_IER** 寄存器的 CH1 位。
4. DMA 请求的产生取决于 **GP16C2T_DMAEN** 寄存器的 CH1 位。

为了处理捕获溢出，建议在读取过捕获标志位之前先读取数据。这是为了避免在读取标志位和读取数据之间发生捕获溢出，从而丢失捕获讯息。

注:捕获中断请求可由软件设置 GP16C2T_SGE 寄存器的 SGCHn 位产生。

22. 5. 7 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. 设置 **GP16C2T_CHMR1** 寄存器的 CC1SSEL 位为 01b，通道 1 选择 I1 为有效输入端。
2. 设置 **GP16C2T_CCEP** 寄存器的 CC1NPOL 位为 0、CC1POL 位为 0，通道 1 选择 I1 上升沿有效，用于捕获数据到 **GP16C2T_CCVAL1** 寄存器。
3. 设置 **GP16C2T_CHMR1** 寄存器的 CC2SSEL 位为 10b，通道 2 同样选择 I1 为有效输入

端。

4. 设置 **GP16C2T_CCEP** 寄存器的 CC2NPOL 位为 0、CC2POL 位为 1，通道 2 选择 I1 下降沿有效，用于捕获数据到 **GP16C2T_CCVAL2** 寄存器。
5. 设置 **GP16C2T_SMCON** 寄存器的 TSSEL1 位与 TSSEL2 位为 00101b，选择 I1FP 信号为有效的触发输入(TRGI)。
6. 设置 **GP16C2T_SMCON** 寄存器的 SMODS 位为 100b，选择从模式控制器为复位模式，让每次有效触发输入(TRGI)初始化计数器。
7. 设置 **GP16C2T_CCEP** 寄存器的 CC1EN 位和 CC2EN 位为 1，开启捕获。
8. 设置 **GP16C2T_CON1** 寄存器的 CNTEN 位为 1，开启计数器。

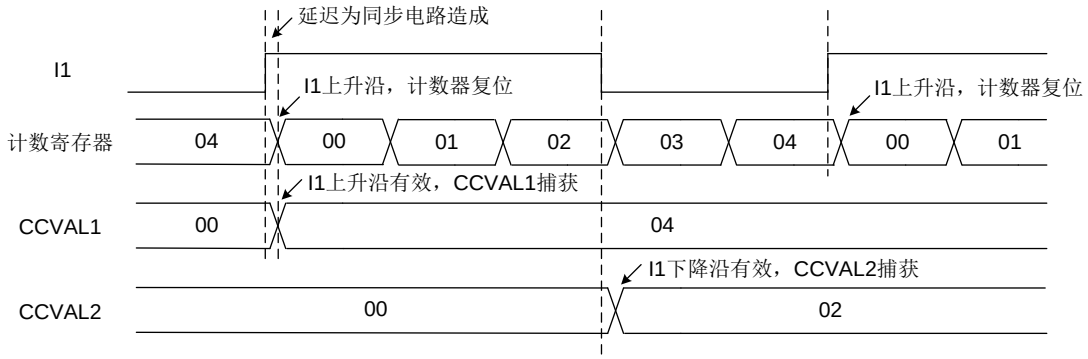


图 23-13　PWM 输入模式时序

- **GP16C2T_CCVAL1** 寄存器内的值为 PWM 周期(两次上升沿之间的时间)，此范例中 PWM 周期为 5 个计数单位。
- **GP16C2T_CCVAL2** 寄存器内的值为 PWM 脉冲宽度(上升沿到下降沿之间的时间)，此范例中 PWM 脉冲宽度为 3 个计数单位，可推算其占空比为 60%。

注:计数单位取决于时钟频率以及预分频设定值

22. 5. 8　PWM输出模式

脉宽调制(PWM)模式可以产生一个由 **GP16C2T_AR** 寄存器设置输出频率，由 **GP16C2T_CCVALn** 寄存器设置占空比的信号。若要开启相应的预装载寄存器，需将 **GP16C2T_CHMRn** 寄存器的 CHnPEN 位设置为 1，并将 **GP16C2T_CON1** 寄存器的 ARPEN 位设置为 1 以开启自动重载功能。

在开启预装载和自动重载功能后，只有当更新事件发生时，才会将预装载寄存器的值写入到缓冲寄存器中。因此，在开始计数前，必须通过将 **GP16C2T_SGE** 寄存器的 SGUPD 位设置为 1，以初始化所有的寄存器。

CHn 的极性可以通过 **GP16C2T_CCEP** 寄存器的 CCnPOL 位设置，有效电平可以设置为高电平或低电平。CHn 的输出由 **GP16C2T_CCEP** 寄存器的 CCnEN 位控制。

在 PWM 模式 1 或 2 中，**GP16C2T_COUNT** 会持续与 **GP16C2T_CCVALn** 寄存器的数值比较，

以确定 **GP16C2T_CCVALn** ≤ **GP16C2T_COUNT** 或 **GP16C2T_CCVALn** ≥ **GP16C2T_COUNT**。

定时器产生 PWM 波形时可以是边沿对齐或中心对齐，取决于 **GP16C2T_CON1** 寄存器的 CMSEL 位。

22. 5. 8. 1 PWM边沿对齐模式

◆ 递增计数配置

以 CH1 输出 PWM 模式 1 为例，相关配置流程如下:

1. 设置 **GP16C2T_CHMR1** 寄存器的 CH1MOD 位为 00110b，选择 PWM 模式 1。
2. 设置 **GP16C2T_CCEP** 寄存器的 CC1POL 位为 0，选择 CH1 通道输出为高电平有效。
3. 设置 **GP16C2T_CCVAL1** 寄存器的 CCRV1 位为 04h，当计数器数到 4 时，PWM 输出低电平。
4. 设置 **GP16C2T_AR** 寄存器的 ARV 位为 08h，当计数器递增到 8 后重载。
5. 设置 **GP16C2T_CON1** 寄存器的 CNTEN 位为 1，开启计数器。

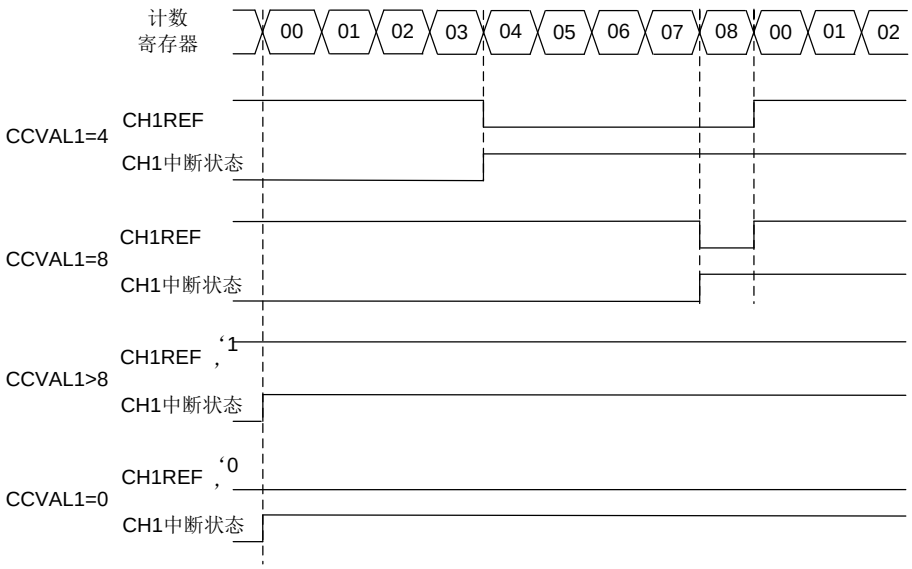


图 23-14 边沿对齐递增计数 PWM 波形(AR=8)

◆ **GP16C2T_COUNT<GP16C2T_CCVAL1** 时，CH1 为高电平。

◆ **GP16C2T_COUNT≥GP16C2T_CCVAL1** 时，CH1 为低电平。

通过改变 **GP16C2T_CCVAL1** 寄存器的 CCRV1 位，可以改变 PWM 信号的占空比。如果将 **GP16C2T_CCVAL1** 设置为大于 **GP16C2T_AR** 的值，则 CH1 将永远输出高电平，并在计数器上溢时产生比较匹配事件。如果将 **GP16C2T_COUNT** 设置为 0，则 CH1 将永远输出低电平。

22. 5. 9 输出比较模式

这个功能被用来控制一个输出波形或是指示一段时间是否已经过去。

当捕获或比较寄存器和 **GP16C2T_COUNT** 寄存器数值匹配时:

- **GP16C2T_CHMRn** 寄存器的 CHnMOD 位选择输出模式，而输出极性由 **GP16C2T_CCEP** 寄存器的 CCnPOL 位控制:
  - ◇ 设置 CHnMOD 位为 00000b:当计数器匹配比较器时输出保持其电平。
  - ◇ 设置 CHnMOD 位为 00001b:当计数器匹配比较器时输出有效电平(假设 CCnPOL=0，有效电平为高电平)。
  - ◇ 设置 CHnMOD 位为 00010b:当计数器匹配比较器时输出无效电平(假设 CCnPOL=0，无效电平为低电平)。
  - ◇ 设置 CHnMOD 位为 00011b:当计数器匹配比较器时翻转电平。
- 设置中断状态寄存器的标志位为 1 (**GP16C2T_RIF** 寄存器的 CHn 位)。
- 若设置相应的中断开启位为 1(**GP16C2T_IER** 寄存器的 CHn 位)，则产生中断。
- 若设置相应的开启位为 1(**GP16C2T_DMAEN** 寄存器的 CHn 位，**GP16C2T_CON2** 寄存器的 CCDMASEL 位用于 DMA 请求的选择)，则发送 DMA 请求。

设置 **GP16C2T_CHMRn** 寄存器的 CHnPEN 位数值可决定 **GP16C2T_CCVALn** 寄存器是否有缓冲功能。

在输出比较模式中，更新事件 UPD 对 CHn 的输出没有影响。输出比较模式同样可以用来输出单个脉冲(单脉冲模式)。

输出比较的设置过程:

1. 选定计数器时钟(内部、外部、预分频)。
2. 设置 **GP16C2T_AR** 与 **GP16C2T_CCVALn** 寄存器并写入所需数据。
3. 若需要产生中断请求，设置 **GP16C2T_IER** 寄存器的 CHn 位为 1。
4. 选择输出模式，例如:
   - 设置 CHnMOD 位为 00011b，当 CNTV 与 CCRVALn 匹配时，CHn 输出翻转。
   - 设置 CHnPEN 位为 0，关闭预装载功能。
   - 设置 CCnPOL 位为 0，选择有效电平为高电平。
   - 设置 CCnEN 位为 1，开启输出。
5. 设置 **GP16C2T_CON1** 寄存器的 CNTEN 位为 1，开启计数器。

假设预装载寄存器开启(CHnPEN 位为 1)，设置 **GP16C2T_CCVALn** 寄存器数值在下次更新事件发生时更新至缓冲寄存器。预装载寄存器未开启(CHnPEN 位为 0)，通过设置 GP16C2T_CCVALn 寄存器数值可随时更新控制输出波形。

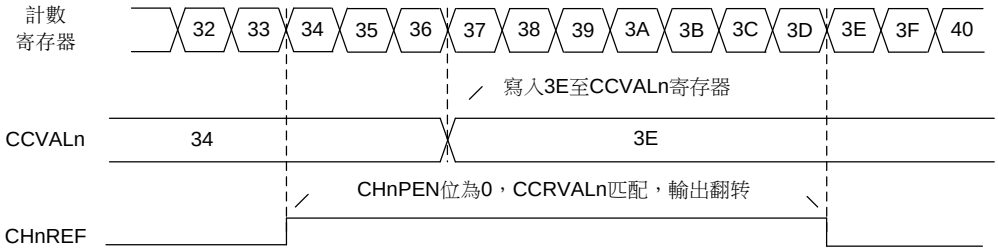


图 23-15　输出比较模式，触发 CHn

22. 5. 10 强制输出模式

在输出模式(**GP16C2T_CHMRn** 寄存器的 CCnSSEL 位为 00b)下，通过软件可以将输出比较信号强制设置为高电平或低电平，输出信号并不会参考 **GP16C2T_CCVALn** 寄存器和 **GP16C2T_COUNT** 寄存器之间的比较结果。

设置 **GP16C2T_CHMRn** 寄存器的 CHnMOD 位为 00101b，输出比较参考信号(CHnREF)为强制高电平，输出比较信号(CHn/CHnN)强制为有效电平(极性由 **GP16C2T_CCEP** 寄存器对应的 CCnPOL 位或 CCnNPOL 位决定)。反之，若设置 CHnMOD 位为 00100b 则强制设置低电平。

例如:设置 CCnPOL 位为 0(CHn 高电平有效)，则 CHn 被强制为高电平。

在此模式下，**GP16C2T_CCVALn** 寄存器和 **GP16C2T_COUNT** 寄存器之间的比较仍然进行，仍可设置相应的标志位为 1，并发送相应的中断以及 DMA 请求。

22. 5. 11 组合PWM模式

在组合模式中，产生两个可编程延迟的边沿或中心对齐 PWM 波型。配置 **GP16C2T_AR** 寄存器数值决定 PWM 频率，PWM 占空比和延迟由两个 **GP16C2T_CCVALn** 寄存器配置。产生的 PWM 波型由两个参考的 PWM 波型做"OR"运算或"AND"运算所组成。

- CH1REFC(或 CH2REFC)由配置 **GP16C2T_CCVAL1** 和 **GP16C2T_CCVAL2** 控制。

2 个通道可以独立选择非对称 PWM 模式，每对 CCVALn 决定一个 CHn 输出。

设置 **GP16C2T_CHMRn** 寄存器的 CHnMOD 位写入 01100b 为组合 PWM 模式 1，写入 01101b 为组合 PWM 模式 2。

当通道设置为组合 PWM 模式时，2 个通道必须设置成不同的 PWM 模式，例如通道 1 设置组合 PWM 模式 1，通道 2 必须设置组合 PWM 模式 2。

注:因为兼容性的关系，CHnMOD[4:0]位分为两个部分，最高两位与其他位并不连续。

下图为组合 PWM 模式中，可以产生的通道波型，通过以下配置:

- 通道 1 配置组合 PWM 模式 2。
- 通道 2 配置 PWM 模式 1。

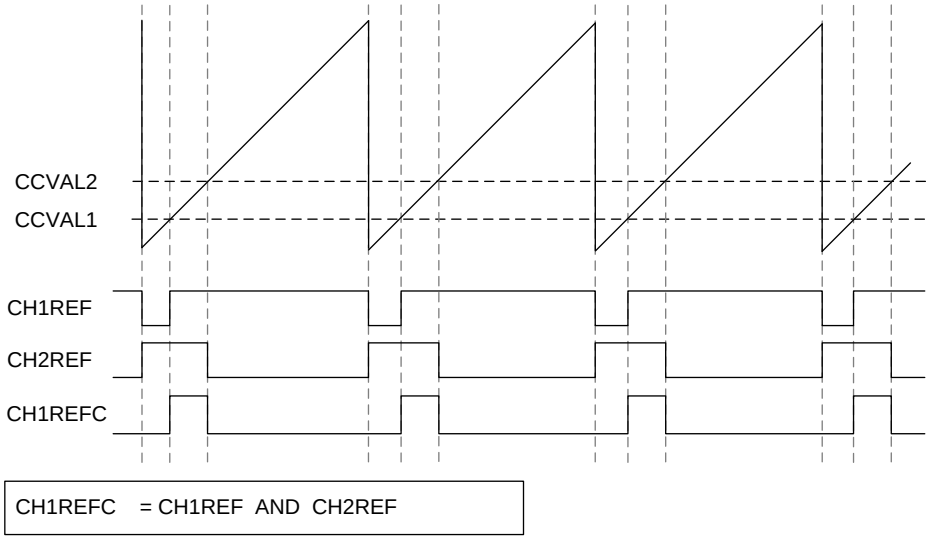


图 23-16　组合 PWM 模式 AND 输出波型

- 通道 1 配置组合 PWM 模式 1。
- 通道 2 配置 PWM 模式 2。

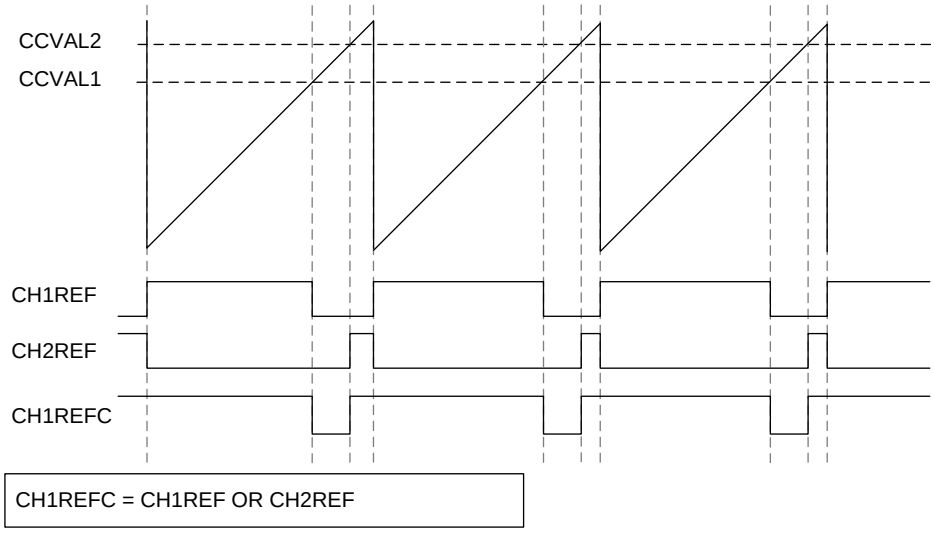


图 23-17　组合 PWM 模式 OR 输出波型

22. 5. 12 外部事件清除比较输出

设置相对应的 **GP16C2T_CHMRn** 寄存器的 CHnOCLREN 位为 1，在选定的 OCLR_INT 输入端为高准位时，可将相对应的输出信号暂时清除为 0，直到下一次更新事件(UPD)产生。该功能只能在输出比较模式和 PWM 模式下使用，在强制模式下不起作用。

如下图所示，OCLR_INT 的输入来源是 **GP16C2T_AFR2** 寄存器的 OCLRSEL 位所指定的 OCLR 输入源。

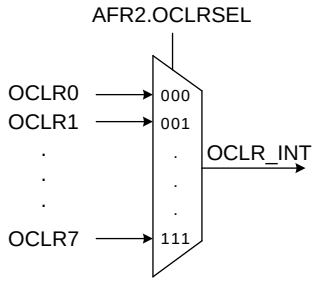


图 23-18　清除比较输出源 OCLR_INT

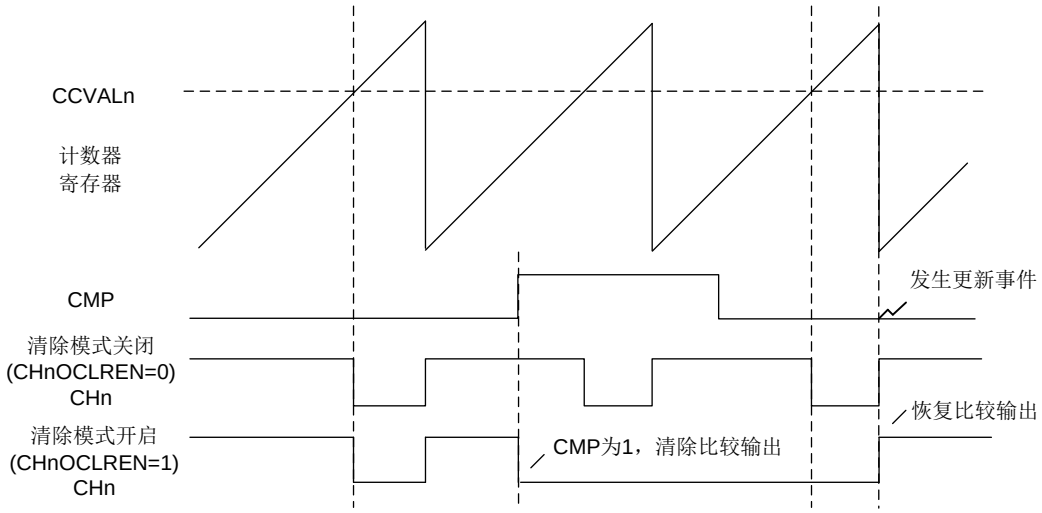


图 23-19　清除比较输出 CHn

22. 5. 13 单脉冲模式

单脉冲模式(SPMEN)是一个特殊模式。在此模式下，计数器可以通过外部触发下启动，并可以产生一个脉宽可编程的波形。

通过从模式控制器开启计数器。在输出比较模式或 PWM 模式下生成波形。设置 **GP16C2T_CON1** 寄存器的 SPMEN 位为 1 选择单脉冲模式，在下一次发生更新事件后，计数器将自动停止计数。

只有当 **GP16C2T_CCVALn** 寄存器值和初始 **GP16C2T_COUNT** 寄存器值不同时，才能正确

的产生一个脉冲。计数器开始计数前(定时器等待触发)，必须如下设置:

- 递增计数:CNTV < CCVALn ≤ AR(注意:0 < CCVALn)

举例来说，假设今天要达到以下效果:
当检测到 I2 输入引脚出现上升沿后，定时器自动开启计数，并经过 T 延迟的时间后，在 CH1 上产生一个长度为 T 脉冲的正脉冲。

就可以设定成单脉冲模式。

选择 I2 作为触发源，相关配置如下:

1. 设置 **GP16C2T_CHMR1** 寄存器中的 CC2SSEL 位为 01b，选择通道输入源为 I2。
2. 设置 **GP16C2T_CCEP** 寄存器中的 CC2POL、CC2NPOL 位为 0，选择 I2 上升沿有效。
3. 设置 **GP16C2T_SMCON** 寄存器的 TSSEL 位为 00110b，选择触发来源(TRGI)为 I2 滤波后信号(I2FP)。
4. 设置 **GP16C2T_SMCON** 寄存器的 SMODS 位为 0110b，选择触发模式，在检测到 TRGI 上升沿时自动开启计数器。

脉冲波型由 **GP16C2T_CCVAL1** 寄存器决定，其中:

- $T_{延迟}$ = **GP16C2T_CCVAL1** 寄存器的值。
- $T_{脉冲}$ = **GP16C2T_AR** 与 **GP16C2T_CCVAL1** 之间的差决定。

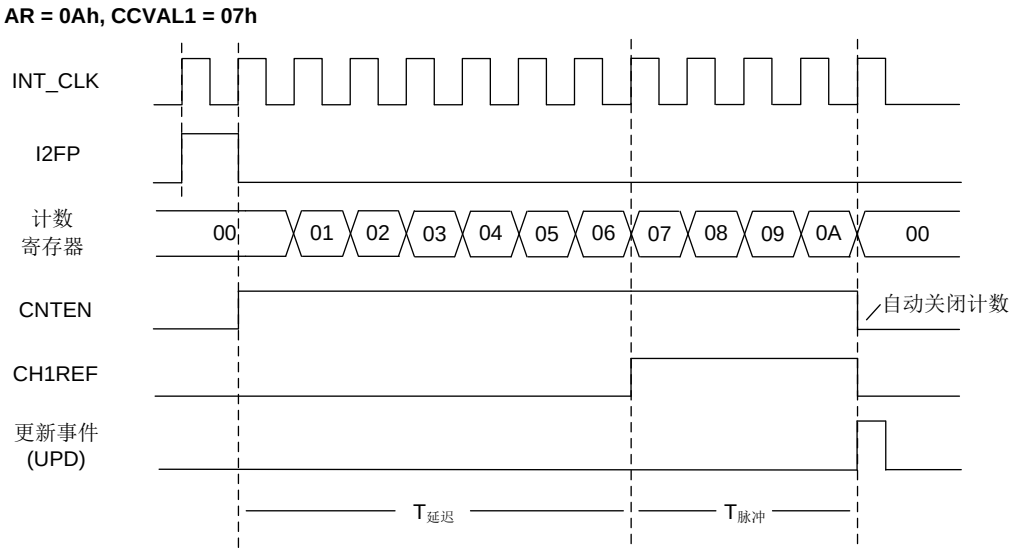


图 23-20　通道 1 触发模式下，基于 PWM 模式 2 单脉冲输出波形

CHn 快速开启模式

在单脉冲模式下，In 输入的有效边沿会自动开启计数器(硬件设置 CNTEN 位为 1)，并在 **GP16C2T_COUNT** 与 **GP16C2T_CCVALn** 比较后输出信号。然而在这个过程中需要数个时钟周期，将会延长 In 输入边沿与输出信号之间的延迟。

如果要使用最小延迟输出信号，可以设置 **GP16C2T_CHMR1** 寄存器的 CHnFEN 位为 1 开启快速开启模式。当侦测到 In 输入的有效边沿时，不再考虑比较值，强制让输出信号等效于匹配成功后的电平。此配置只在 PWM 模式 1 或 PWM 模式 2 时才能使用。

22. 5. 14 多次触发单脉冲模式

多次触发单脉冲模式下，当在一个计时周期中发生多次触发事件可以延长脉冲长度，与发单脉冲模式差别如下:

- 发生触发时，立即产生脉冲(无延迟)。
- 在前次触发脉冲结束前，又发生新的触发，则会延长脉冲长度。

以通道 1 多次触发单脉冲模式为例，相关配置如下:

1. 设置 **GP16C2T_SMCON** 寄存器中的 SMODS 位为 1000b，选择组合复位+触发模式。
2. 设置 **GP16C2T_CHMR1** 寄存器中的 CH1MOD 位为 1000b 或 1001b，选择多次触发单脉冲模式 1 或模式 2。

定时器配置递增计数模式时，对应的 CCVALn 必须配置 0，由 ARV 配置脉冲长度。

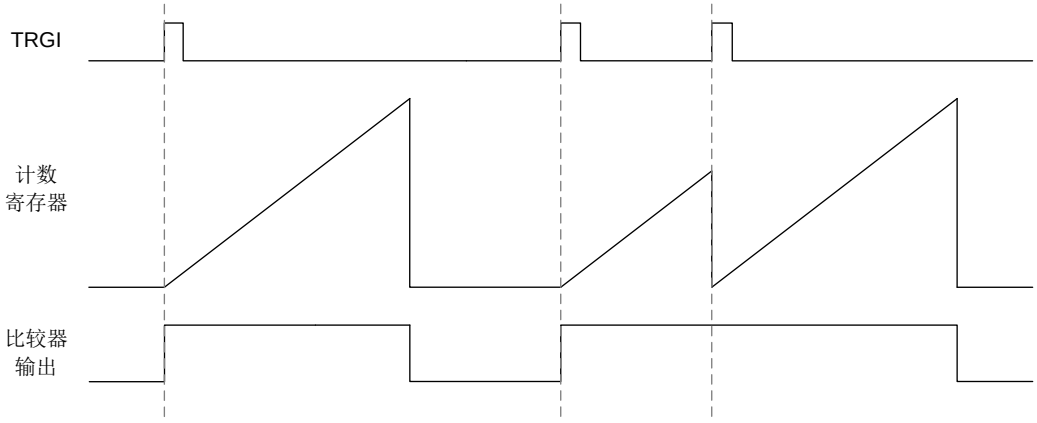


图 23-21　多次触发单脉冲模式

22. 5. 15 互补输出与死区时间

在控制输出时，需要根据连接到输出的装置特性(如电平转换器的固有延迟时间，功率开关的延迟时间等)，延迟两个互补输出开关的时间，这个瞬时的延迟即为死区时间。

每个输出的极性(主输出 CHn 或互补输出 CHnN)可以独立选择，通过在 **GP16C2T_CCEP** 寄存器中写入 CCxP 和 CCxNP 位来实现。

互补信号 CHn 和 CHnN 是由多个控制位的组合所控制的，分别是 **GP16C2T_CCEP** 寄存器的 CCnEN 和 CCnNEN 位，**GP16C2T_BDCFG** 和 **GP16C2T_CON2** 寄存器的 GOEN、OISSn、OISSnN、OFFSSI 及 OFFSSR 位(可参考输出引脚控制章节)。

值得注意的是，当切换为空闲状态(GOEN 变为 0)后，死区时间依然有效。

死区时间长度计算公式可参考 **GP16C2T_BDCFG** 寄存器的 DT 位，用来控制所有通道的死区

时间长短。输出波形参考 CHnREF 信号，产生 CHn 和 CHnN 两路输出。当 CHn 和 CHnN 皆为高电平有效时，两路输出行为如下:

- CHn 的输出信号与参考信号(CHnREF)一致。相较于参考信号的上升沿，CHn 上升沿输出会有延迟。
- CHnN 的输出信号与参考信号相反。相较于参考信号的下降沿，CHnN 上升沿输出会有延迟。

设置 **GP16C2T_DCFG2** 寄存器的 DTAEN 位为 1 时，开启非对称死区延迟功能:

- 相较于参考信号的上升沿，CHn 上升沿输出延迟为 **GP16C2T_BDCFG** 寄存器的 DT 位设定。
- 相较于参考信号的下降沿，CHnN 上升沿输出延迟为 **GP16C2T_DCFG2** 寄存器的 DT2 位设定。

设置 **GP16C2T_DCFG2** 寄存器的 DTPEN 位为 1 时，死区延迟(DT、DT2)设定具有缓冲功能，只有在更新事件发生时，才会将预装载的值写入到缓冲寄存器中。

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 位为 0，CCnNP 位为 0，GOEN 位为 1，CCnEN 位为 1，和 CCnNEN 位为 1)。

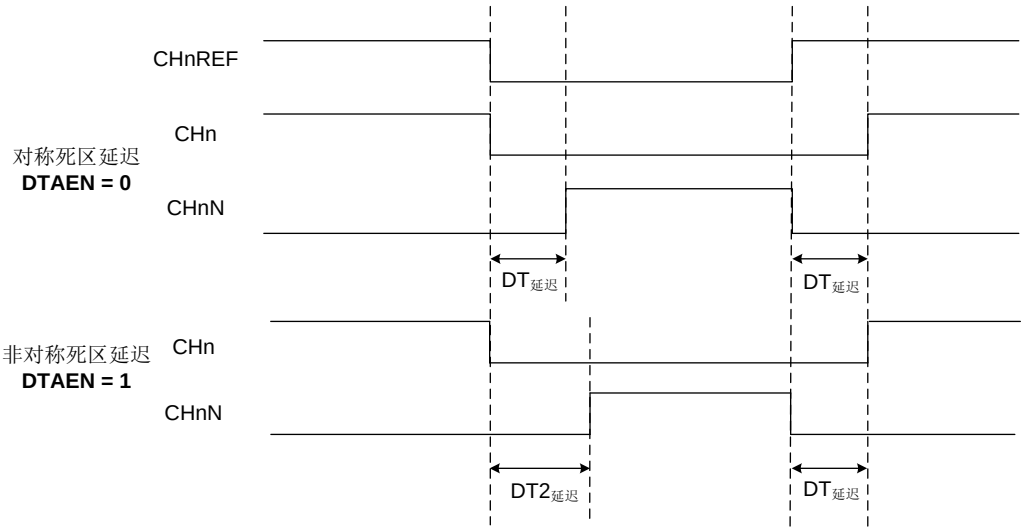


图 23-22　互补输出含死区时间插入

任何情况下，CHn 与 CHnN 的输出信号都不能同时为有效电平。

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。

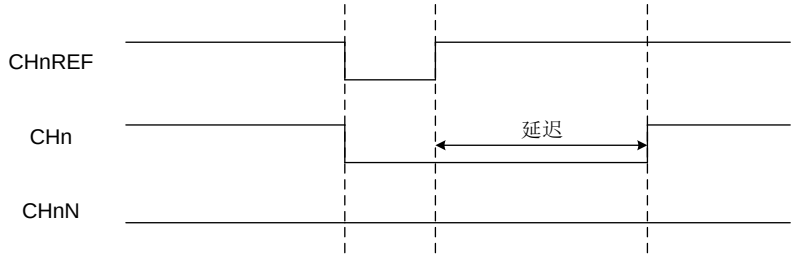


图 23-23　死区延迟时间大于 CHnN 脉冲

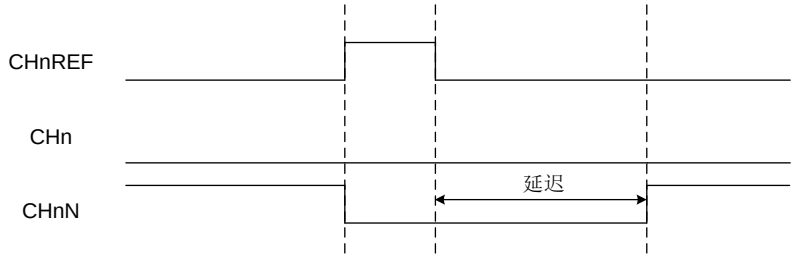


图 23-24　死区延迟时间大于 CHn 脉冲

22. 5. 16 刹车功能

刹车功能是为了保护受到 PWM 信号驱动的电路。该功能包含两个刹车输入通道，用来检测故障事件。当发生刹车事件时，硬件会自动关闭 PWM 输出，并将其强制设置为预先配置好的安全准位。此外，刹车事件也可以选择 MCU 内部事件当作刹车输入源。

刹车功能拥有两个通道。刹车输入通道 1 可以检测系统级的故障(如时钟失效、Hard Fault、低电压检测等)和应用故障(来自输入引脚和内置比较器)，并在死区延迟后强制将输出设置为预配置的电平(有效或无效)。刹车输入通道 2 仅包括应用故障，可以强制将输出设置为无效状态。

刹车期间的输出开启信号与输出电平由以下几个控制位进行决定:

- **GP16C2T_BDCFG** 寄存器的 GOEN 位，通过软件开启或关闭输出，并在刹车输入通道 1 或 2 检测到刹车事件时清 0。
- **GP16C2T_BDCFG** 寄存器的 OFFSSI 位,用来定义此时输出是由定时器控制成无效状态，或由 GPIO 控制(通常处于 High-Z 高阻态)。
- **GP16C2T_CON2** 寄存器的 OISSn 和 OISSnN 位，用来定义在输出空闲状态时的输出状态(有效或无效)。

注:OISSn 和 OISSnN 位无法配置成同时输出有效电平。

详细信息可参考输出引脚控制章节。

系统复位后，刹车电路被禁止且 GOEN 位为 0。可以通过在 **GP16C2T_BDCFG** 寄存器中设置 BRKEN 位来启用刹车功能。刹车输入极性可以通过在同一寄存器中配置 BRKP 位来选择。

刹车(BRK)信号来源为:

- BKIN 输入引脚
- BKIN_EXT 输入引脚
- 内部来源
  - 时钟停振事件由时钟控制器(RCU)中的时钟安全系统(CSS)产生
  - 内部 LOCKUP(Hard Fault)输出
  - PVD 输出
  - 内部比较器输出

软件可以配过配置 **GP16C2T_SGE** 寄存器的 SGBRK 位产生刹车事件。

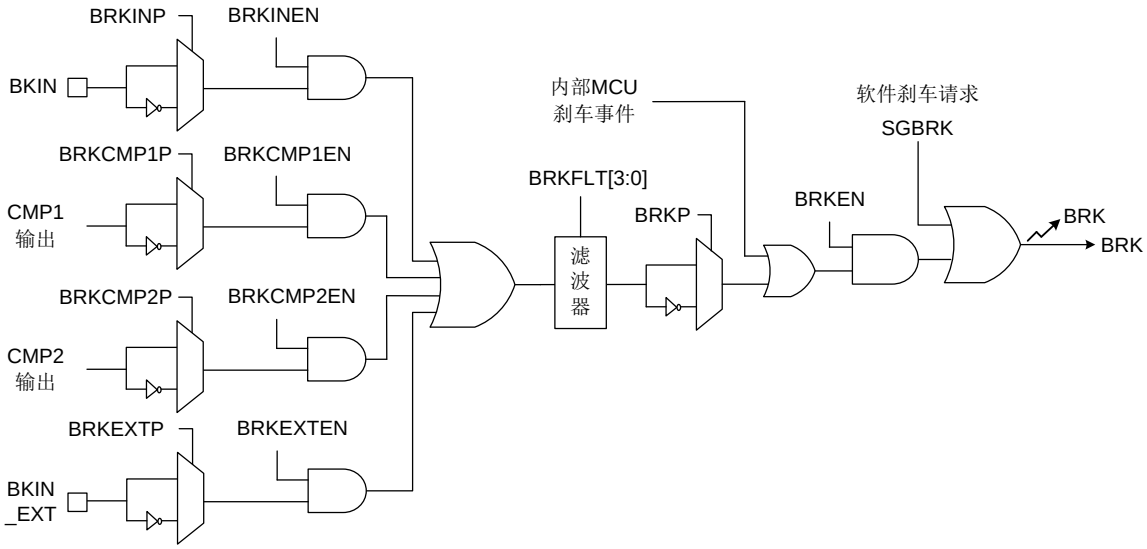


图 23-25　刹车通道电路

当发生刹车事件，也就是刹车输入检测到有效电平时，会有以下几个情况:

首先，GOEN 位会被清除为 0。如果 OFFSSI 位为 0，输出通道将处于高阻态，此时不受定时器控制。

如果 OFFSSI 位为 1，则输出通道的行为取决于是否为互补输出。对于非互补输出，输出端会根据 **GP16C2T_CON2** 寄存器中的 OISSn/OISSn 位设定，输出预先配置好的空闲电平。

对于互补输出，首先输出端会输出无效状态(取决于极性)。接着在死区时间之后，才会根据 **GP16C2T_CON2** 寄存器中的 OISSn/OISSn 位设定，输出预先配置好的空闲电平。

在刹车事件发生时，还会根据不同的刹车事件，设置 **GP16C2T_RIF** 寄存器中的相应 BRK 位。如果 **GP16C2T_IER** 寄存器中相应位开启，则会触发中断。

最后，如果 **GP16C2T_BDCFG** 寄存器中的 AOEN 位为 1，则在下一次更新事件(UPD)发生时，硬件会自动将 GOEN 位设为 1。

注:刹车输入为电平有效。因此当刹车输入持续有效电平时，无法将 GOEN 设置为 1(自动或者通过软件)。此外，BRK 中断状态也无法被清除。

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，通过 **GP16C2T_BDCFG** 寄存器的 LOCKLVL 位，可从三个保护等级中选择一种保护等级。MCU 复位后，只能对 LOCKLVL 位执行一次写操作。

以下两张图分别说明在互补与非互补输出，响应 BRK 有效刹车事件时的输出电位(OFFSSI=1)。

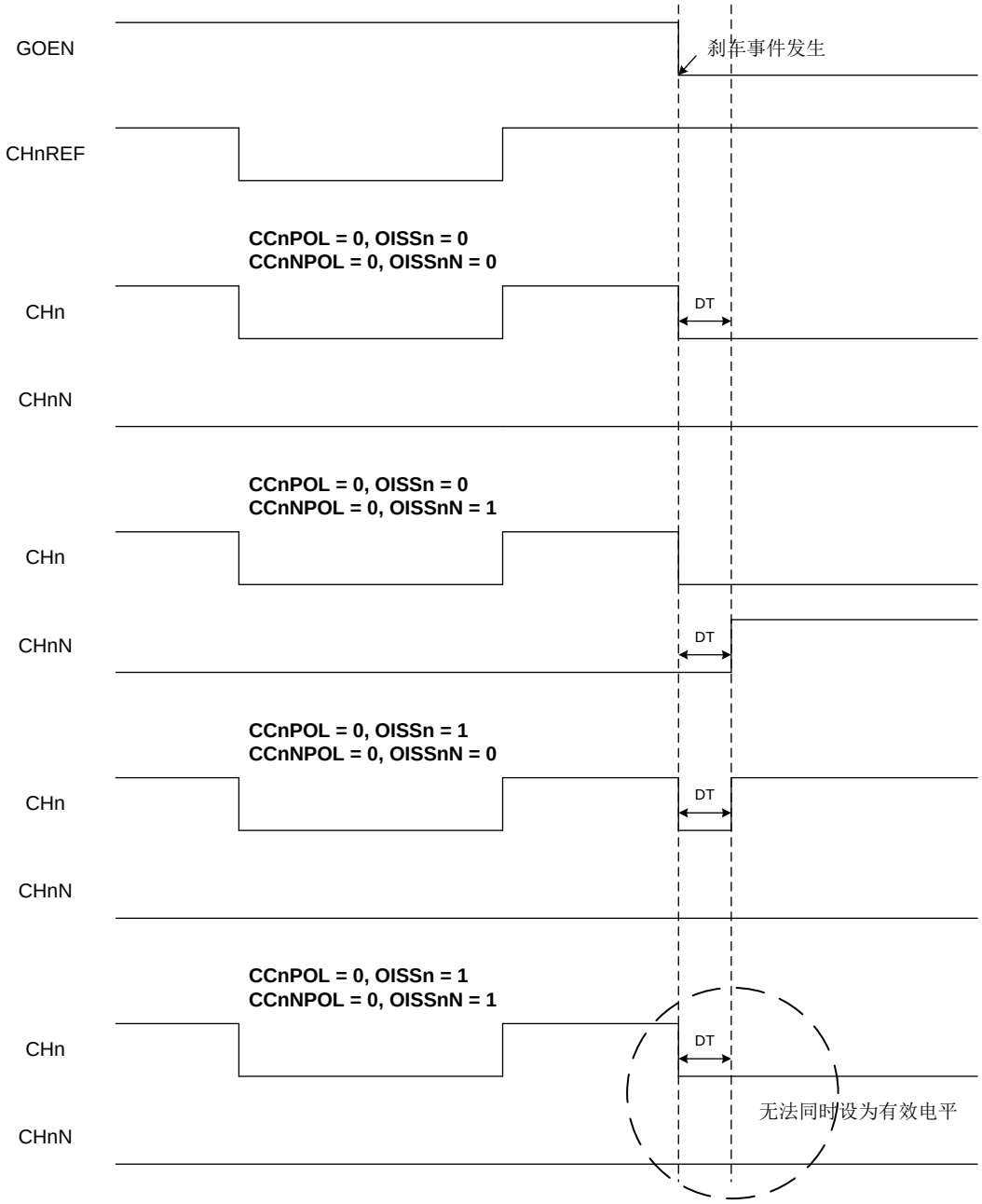


图 23-26　非互补输出下的刹车事件(OFFSSI=1, CCnEN = 1, CCnNEN = 0)

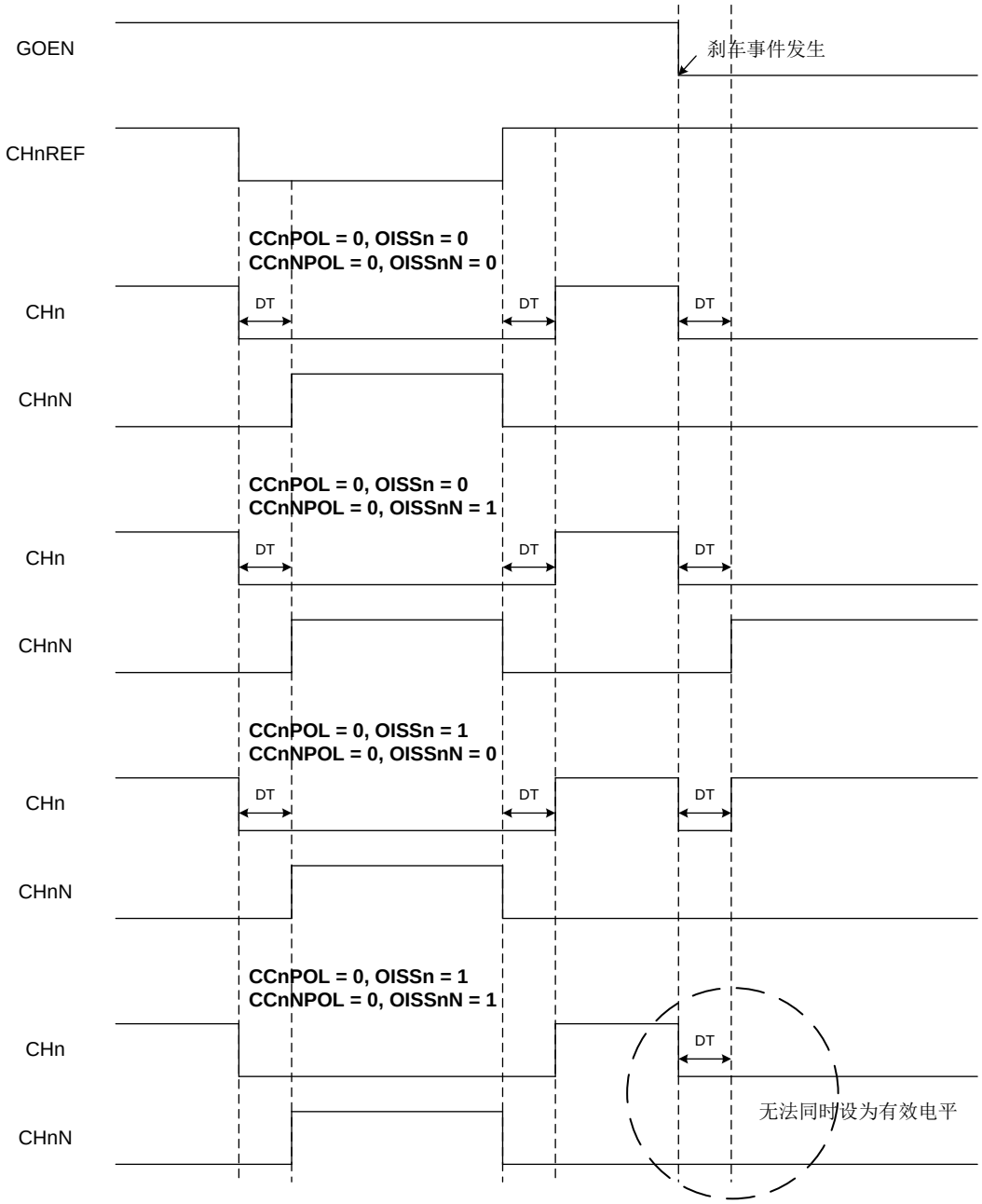


图 23-27　互补输出下的刹车事件(OFFSSI=1, CCnEN = 1, CCnNEN = 1)

下图为不支持互补输出的通道，响应 BRK 有效刹车事件时的输出电位(OFFSSI=1)

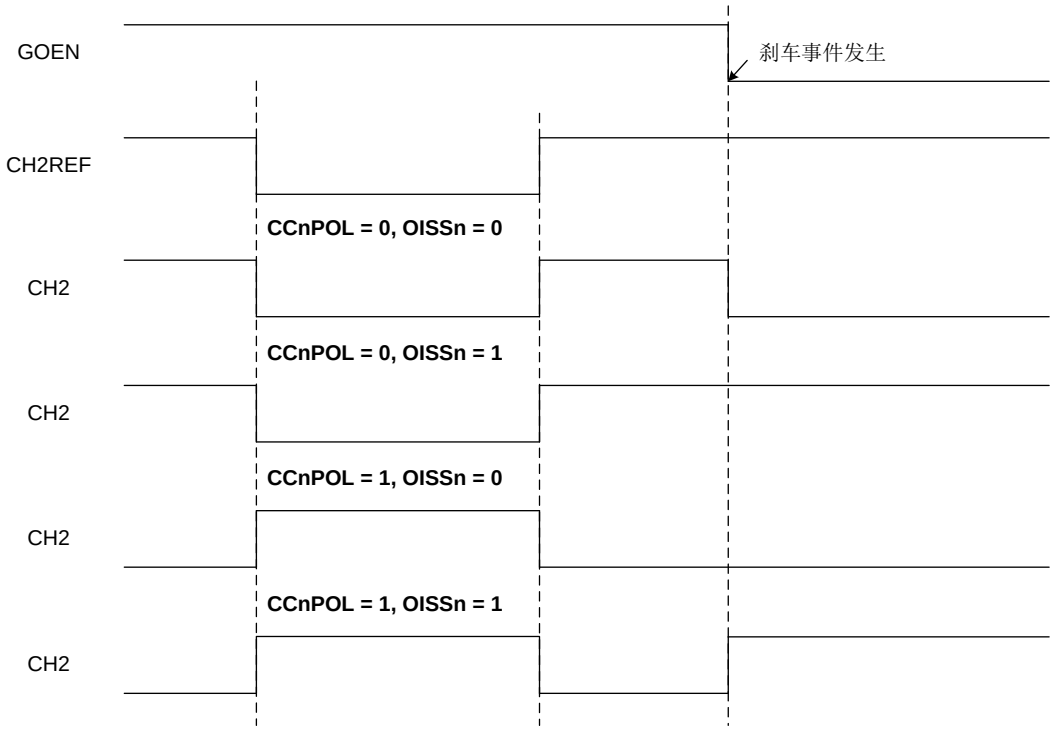


图 23-28　不支持互补输出下的刹车事件(OFFSSI=1, CCnEN = 1)

22. 5. 17 双向刹车输入

定时器支持双向刹车输入引脚(BKIN)，如下图:

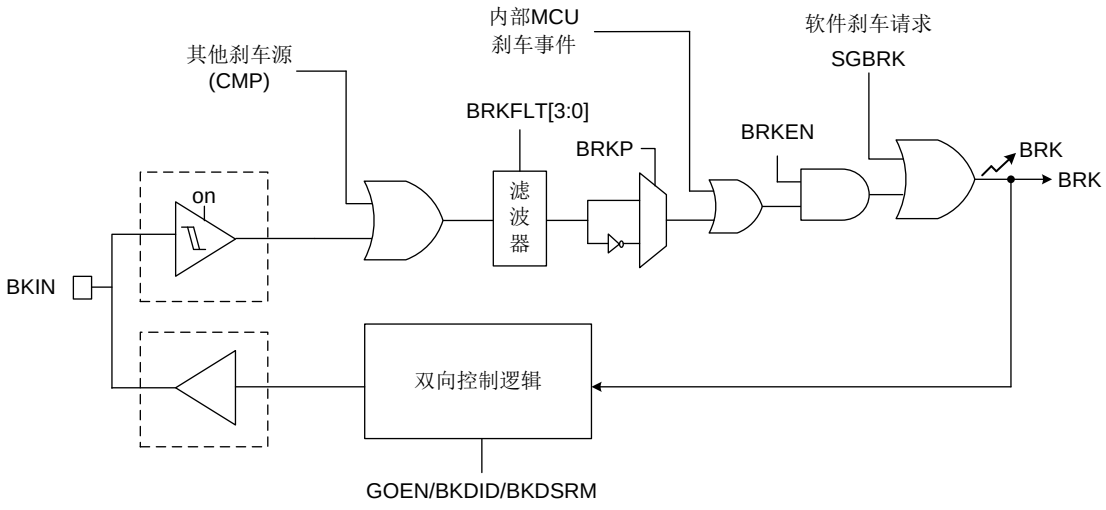


图 23-29　刹车输入通道

- 通过刹车输入引脚向外部 MCU 或驱动电路发送故障信号。
- 内部刹车源与外部刹车输入引脚使用"OR"运算，触发刹车事件。

启用双向刹车输入模式需要设置 **GP16C2T_BDCFG** 寄存器的 BKDID 位为 1，将刹车输入引脚

BKIN 配置为双向模式。

当发生刹车事件时，定时器会将刹车输入引脚(BKIN)强制设为低电平输出，用来向外部驱动电路指示发生了故障事件。

因此在使用双向刹车模式时，外部驱动电路的有效刹车信号源必须设计成低电平有效，如下图:

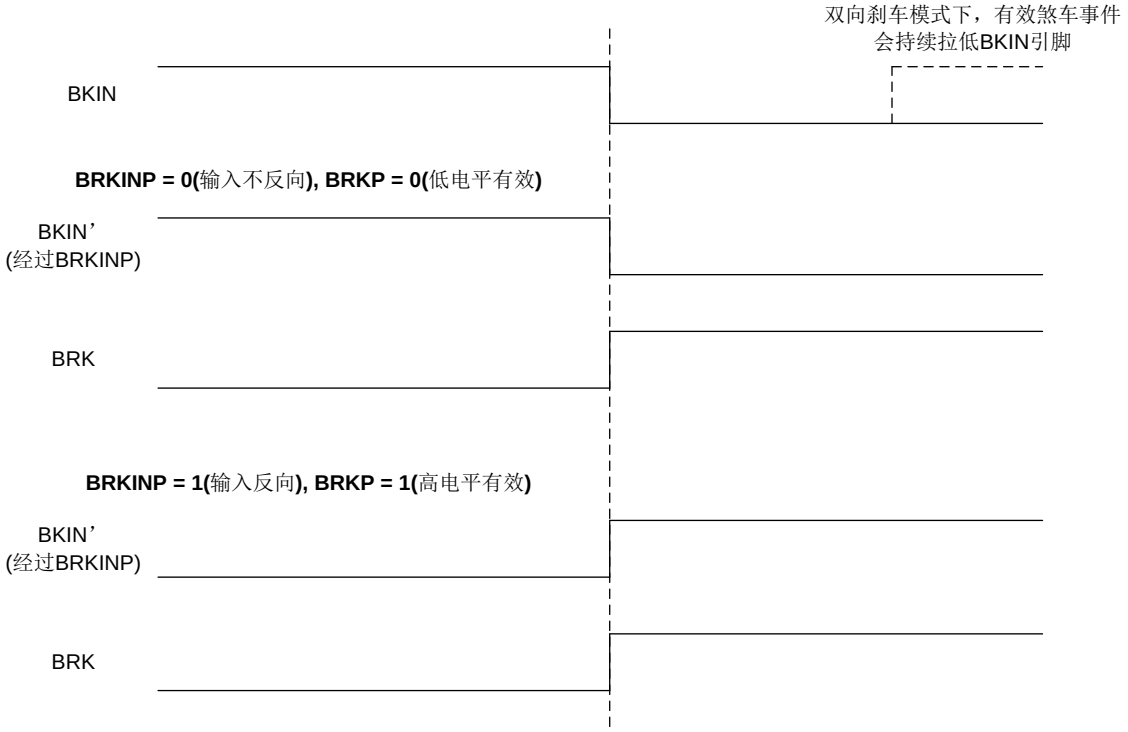


图 23-30　有效双向刹车输入模式

注: 当准位配置错误时(例如配置成高电平有效)，为了安全目的无法支持双向刹车输入模式。

使用软件刹车事件 SGBRK 也会将 BKIN 引脚强制设为低电平输出。但若在刹车输入关闭(BRKEN = 0)的情况下使用软件触发刹车信号，定时器只会生成有效刹车事件，并不会影响 BKIN 引脚状态。

当发生有效刹车事件后(GOEN 清 0)，可通过以下步骤让刹车保护机制重新警备:

1. 设置 **GP16C2T_BDCFG** 寄存器的 BKDSRM 位为 1，硬件释放刹车输入引脚(不强制设为低电平输出)。
2. 定时器会持续检测刹车输入引脚电平，直到为无效电平，硬件将自动清除 BKDSRM 位为 0。
3. 软件持续检测 BKDSRM 位，直到被硬件自动清 0，表示此时刹车保护机制重新进入警备状态。
4. 清除相关刹车中断事件(若有开启中断)。
5. 软件设置 GOEN 位为 1，重新输出 PWM 信号，若开启 AOEN，定时器在更新事件后自

动开启 PWM 输出。

22. 5. 18 输出引脚控制

在输出模式(强制输出、输出比较或 PWM 模式)下，设置参考信号 CHnREF 映像到 CHn 输出或 CHnN 输出的方式，可以通过配置 **GP16C2T_CCEP** 寄存器的 CCnEN、CCnNEN 位来实现。而输出的极性可以由 **GP16C2T_CCEP** 寄存器的 CCnPOL 与 CCnNPOL 位决定。此外，死区延迟可以通过 **GP16C2T_BDCFG** 寄存器的 DT 位和 **GP16C2T_DCFG2** 寄存器的 DT2 位(若开启非对称死区)来设置。

在运行模式下(GOEN=1)，如果关闭输出(CCnEN=0 或 CCnNEN=0)，硬件会直接禁止 CHn 或 CHnN 输出，使引脚为 Hi-Z 高阻态(不由定时器控制)。但是，如果开启了 **GP16C2T_BDCFG** 寄存器的 OFFSSR 位，并且关闭输出的通道的互补通道是开启状态，那么该通道会将输出设为无效电平(由 **GP16C2T_CCEP** 寄存器的 CCnPOL 位或 CCnNPOL 位决定)。

当 GOEN 位因刹车事件或软件写入为 0 时，硬件会直接禁止 CHn 或 CHnN 输出，使引脚为 Hi-Z 高阻态(不由定时器控制)。但是，如果开启了 **GP16C2T_BDCFG** 寄存器的 OFFSSI 位，并且 CCnEN 与 CCnNEN 位不全为 0，那么通道会先输出无效电平，并且在死区延迟后，改为输出其空闲电平(根据 **GP16C2T_CON2** 寄存器的 OISSn 及 OISSnN 位决定)。

所有输出引脚控制状态整理如下表:

控制位					输出状态	
GOEN	OFFSSI	OFFSSR	CCnEN	CCnNEN	CHn	CHnN
1	x	x	0	0	禁止引脚输出(Hi-Z，不由定时器控制)	
		0	0	1	禁止引脚输出(Hi-Z，不由定时器控制)	CHnN = CHnREF + 极性(CCnNPOL)
		0	1	0	CHn = CHnREF + 极性(CCnPOL)	禁止引脚输出(Hi-Z，不由定时器控制)
		X	1	1	CHn = CHnREF + 极性(CCnPOL) + 死区(DT)	CHnN = ~CHnREF + 极性(CCnNPOL) + 死区(DT)
		1	0	1	运行模式下关闭状态(输出无效电平) CHn = 极性(CCnPOL)	CHnN = CHnREF + 极性(CCnNPOL)
		1	1	0	CHn=CHnREF+极性(CCnPOL)	运行模式下关闭状态(输出无效电平) CHnN=极性(CCnNPOL)
0	0	x	x	x	禁止引脚输出(Hi-Z，不由定时器控制)	

	1		0	0	
			0	1	空闲模式下关闭状态(输出无效电平) CHn=极性(CCnPOL), CHnN=极性(CCnNPOL) 在死区(DT)时间后，输出空闲电平 CHn = OISSn, CHnN = OISSnN 注:禁止设置 OISSn 与 OISSnN 皆为有效电平
			1	0	
			1	1	

表 23-5　输出引脚状态表

22. 5. 19 UPD位重映射

通过设置 **GP16C2T_CON1** 寄存器的 UPDREMAP 位，可以将 UPD 原始中断状态映射至 **GP16C2T_COUNT** 寄存器的第 31 位 UPDRIF 中。

注:UPD 和 UPDRIF 标志之间没有延迟。

22. 5. 20 输入XOR功能

通过 **GP16C2T_CON2** 寄存器的 I1SEL 位，可将通道 1 的输入滤波器连接到 XOR 门的输出端，XOR 门输入端包含 CH1、CH2 两个输入引脚。

XOR 输出用于定时器的所有输入功能，可用于量测两输入边沿信号之时间间隔，如下图:

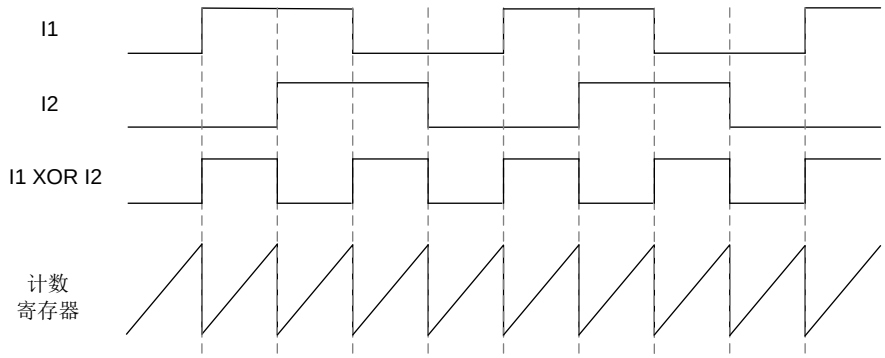


图 23-31　量测两输入边沿信号之时间间隔

22. 5. 21 外部触发的同步

GP16C2T 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。

22. 5. 21. 1 复位模式

当侦测到有效触发输入事件时，计数器及其预分频器会被重新初始化，若此时 **GP16C2T_CON1** 寄存器的 UERSEL 位为 0，则一并产生更新事件 UPD。而所有预装载寄存器(**GP16C2T_AR**，**GP16C2T_CCVALn**)都会因更新事件 UPD 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零，配置过程如下:

1. 设置 **GP16C2T_CHMR1** 寄存器的 CC1SSEL 位为 01b，选择 I1 为有效输入端。
2. 设置 **GP16C2T_CHMR1** 寄存器的 I1FLT 位为 0000b，本例无需滤波器。
3. 设置 **GP16C2T_CCEP** 寄存器的 CC1NPOL 位为 0、CC1POL 位为 0，选择 I1 通道上升沿有效。
4. 设置 **GP16C2T_CHMR1** 寄存器的 I1PRES 位为 00b，捕获预分频器不用于触发操作，无需设置。
5. 设置 **GP16C2T_SMCON** 寄存器的 TSSEL 位为 00101b，选择 I1 滤波后信号作为输入源。
6. 设置 **GP16C2T_SMCON** 寄存器的 SMODS 位为 0100b，选择复位模式。
7. 设置 **GP16C2T_CON1** 寄存器的 CNTEN 位为，开启计数器。

计数器依据内部时钟开始计数，计数器计数直到 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。同时设置标志位为 1(**GP16C2T_RIF** 寄存器的 TRGI 位)，如果中断及 DMA 开启(取决于 **GP16C2T_IER** 寄存器的 TRGI 位和 **GP16C2T_DMAEN** 寄存器的 TRGI 位)，会发送中断及 DMA 请求。

下图给出了设置自动重载寄存器 **GP16C2T_AR** 为 36h 时的信号变化。由于 I1 输入的同步电路，I1 上的上升沿和计数器实际初始化之间会存在延迟。

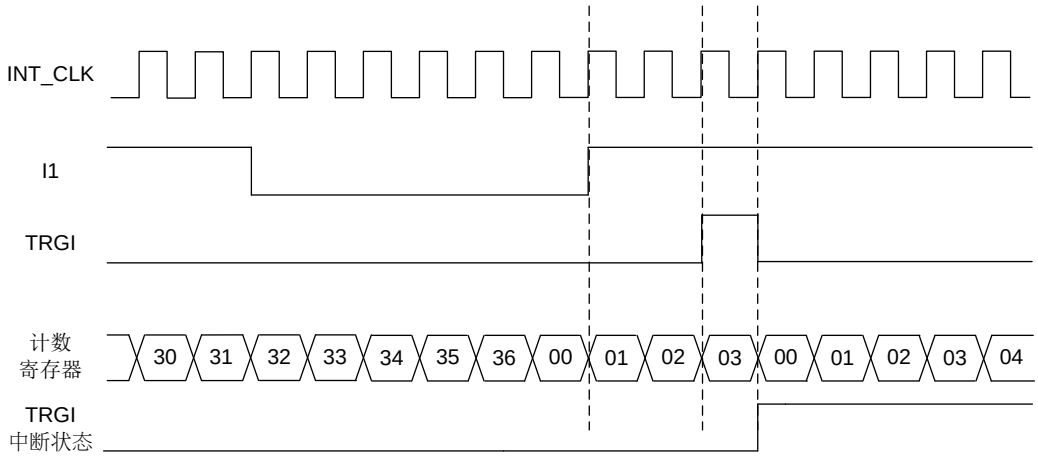


图 23-32 复位模式控制电路

22. 5. 21. 2　门控模式

计数器根据选中的输入电平被开启。

下面的例子中，计数器只在 I1 输入为低电平时才递增计数:

1. 设置 **GP16C2T_CHMR1** 寄存器的 CC1SSEL 位为 01b，选择 I1 为有效输入端。
2. 设置 **GP16C2T_CHMR1** 寄存器的 I1FLT 位为 0000b，本例无需滤波器。
3. 设置 **GP16C2T_CCEP** 寄存器的 CC1NPOL 位为 0、CC1POL 位为 1，I1 通道反相，有效极性为低电平。
4. 设置 **GP16C2T_CHMR1** 寄存器的 I1PRES 位为 00b，捕获预分频器不用于触发操作，无需设置。
5. 设置 **GP16C2T_SMCON** 寄存器的 TSSEL1 位与 TSSEL2 位为 00101b，选择 I1 滤波后信号作为输入源。
6. 设置 **GP16C2T_SMCON** 寄存器的 SMODS 位为 101b，选择门控模式。
7. 设置 **GP16C2T_CON1** 寄存器的 CNTEN 位为 1，开启计数器(在门控模式中，如果 CNTEN 位为 0，无论触发输入为何电平，计数器都不会开启)。

只要 I1 为低电平，计数器依据内部时钟开始计数，一旦 I1 为高电平则停止计数。由于 I1 输入端的同步电路的原因，I1 上出现上升沿和计数器实际停止之间会有一定的延迟。

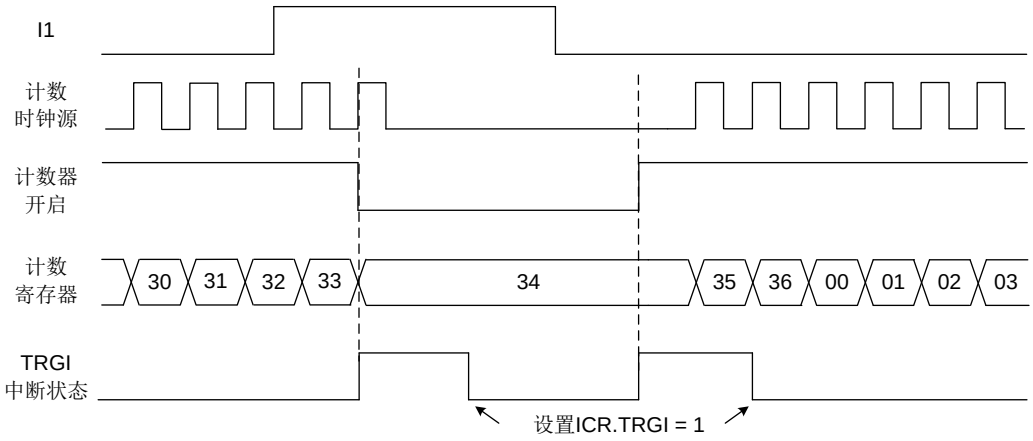


图 23-33　门控模式控制电路

22. 5. 21. 3　触发模式

输入端选中的事件可以开启计数器。

下面的例子中，I2 输入端上的上升沿可以开启递增计数:

1. 设置 **GP16C2T_CHMR1** 寄存器的 CC2SSEL 位为 01b，选择 I2 为有效输入端。
2. 设置 **GP16C2T_CHMR1** 寄存器的 I2FLT 位为 0000b，本例无需滤波器。
3. 设置 **GP16C2T_CCEP** 寄存器的 CC2NPOL 位为 0、CC2POL 位为 0，选择 I2 通道上升沿有效。
4. 设置 **GP16C2T_CHMR1** 寄存器的 I2PRES 位为 00b，捕获预分频器不用于触发操作，无需设置。
5. 设置 **GP16C2T_SMCON** 寄存器的 TSSEL1 位与 TSSEL2 位为 00110b，选择 I2 滤波后信号作为输入源。
6. 设置 **GP16C2T_SMCON** 寄存器的 SMODS 位为 110b，选择触发模式。
7. 设置 **GP16C2T_CON1** 寄存器的 CNTEN 位为，开启计数器。

I2 上出现上升沿时，计数器开始依据内部时钟计数并设置 TRGI 标志位为 1。

由于 I2 输入的同步电路原因，I2 上出现上升沿和计数器实际启动之间会有一定的延迟。

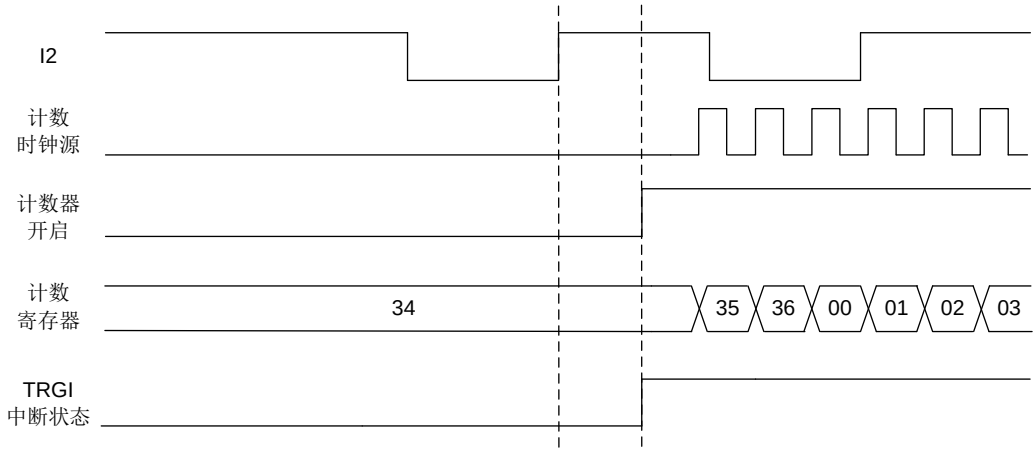


图 23-34　触发模式控制电路

22. 5. 21. 4　组合复位模式 + 触发模式

要使用此模式，需要设置 **GP16C2T_SMCON** 寄存器的 SMODS 位为 1000b。其功能为复位模式加上触发模式，计数器在触发输入 TRGI 的上升沿启动，并重新初始化计数器以及生成更新事件。此模式主要用于单脉冲模式。

22. 5. 21. 5　组合门控模式 + 复位模式

要使用此模式，需要设置 **GP16C2T_SMCON** 寄存器的 SMODS 位为 1001b。其功能为门控模式加上复位模式，当触发输入(TRGI)为高电平时，计数器的时钟开启。一旦触发输入变为低电平，计数器停止且重新初始化。

22. 5. 22 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、开启、停止或提供时钟等操作。

下图显示了触发选择和主模选择模块的概况。

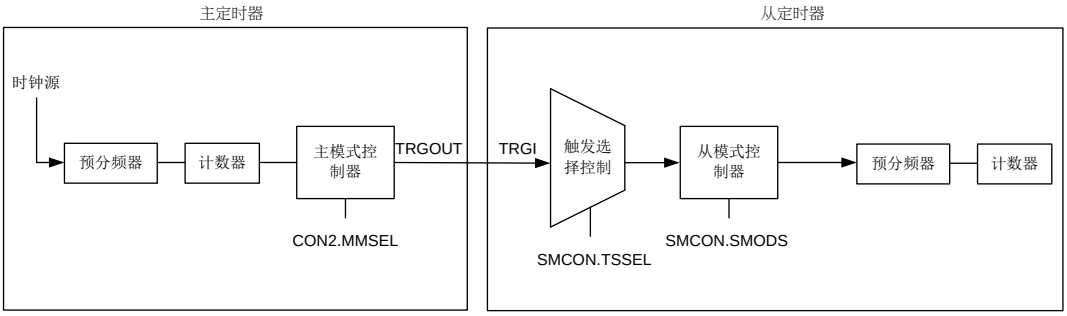


图 23-35　主/从定时器范例

22. 5. 22. 1　使用一个定时器去开启其他定时器

在这个例子中，定时器 2(GP16C2T1)的开启由定时器 1(AD16C6T1)的输出比较参考信号(CH1REF)控制。只有当定时器 1 的 CH1REF 为高电平时，定时器 2 才会计数。

先设定从定时器(定时器 2)为门控模式，配置如下:

1. 设置 **GP16C2T_SMCON** 寄存器的 TSSEL1 位与 TSSEL2 位为 AD16C6T1，可参考内部触发连接表。
2. 设置 **GP16C2T_SMCON** 寄存器的 SMODS 位为 101b，选择门控模式。
3. 设置 **GP16C2T_CON1** 寄存器的 CNTEN 位为 1，开启计数器。

再设定主定时器(定时器 1)为 PWM 输出，配置如下:

1. 设置 **AD16C6T1_PRES** 寄存器的 PSCV 为 01h，计数器时钟频率为 $f_{INT\_CLK}/2$ 。
2. 设置 **AD16C6T1_CHMR1** 寄存器的 CH1MOD 位为 0111b，选择 PWM 模式 2。
3. 设置 **AD16C6T1_CCEP** 寄存器的 CC1POL 位为 0，选择 CH1 通道输出为高电平有效。
4. 设置 **AD16C6T1_CCVAL1** 寄存器的 CCRV1 位为 06h，当计数器数到 6 时，PWM 输出高电平。
5. 设置 **AD16C6T1_AR** 寄存器的 ARV 位为 08h，当计数器递增到 8 后重载。
6. 设置 **AD16C6T1_CON2** 寄存器的 MMSEL 位为 100b，选择输出比较参考信号(CH1REF)为触发输出(TRGOUT)。
7. 设置 **AD16C6T1_CON1** 寄存器的 CNTEN 位为 1，开启计数器。

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 计数器的开启信号。

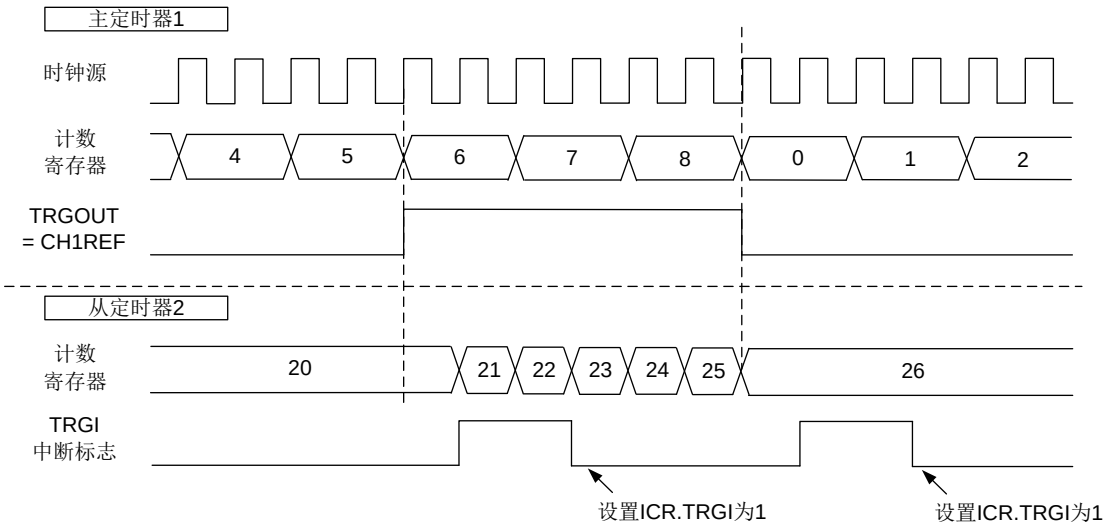


图 23-36　门控从定时器使用主定时器 CH1REF

在上图的例子中，在定时器 2 开启之前，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。可以在开启定时器 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。设置 **GP16C2T_SGE** 与定时器 2 的 SGE 寄存器的 SGUPD 位为 1 即可复位定时器。

22. 5. 22. 2　将一个定时器做为其他定时器的预分频器

在这个例子中，使用定时器 1(AD16C6T1)的更新事件作为定时器 2(GP16C2T1)的时钟来源，没有设定预分频。一旦定时器 1 产生更新事件，定时器 2 即根据其上升沿计数。

先设定从定时器(定时器 2)为外部时钟源 1 模式，配置如下:

1. 设置 **GP16C2T_AR** 寄存器的 ARV 位为 02h，当计数器递增到 2 后重载。
2. 设置 **GP16C2T_SMCON** 寄存器的 TSSEL1 位与 TSSEL2 位为 AD16C6T1，可参考内部触发连接表。
3. 设置 **GP16C2T_SMCON** 寄存器的 SMODS 位为 111b，选择外部时钟模式 1。
4. 设置 **GP16C2T_CON1** 寄存器的 CNTEN 位为 1，开启计数器。

再设定主定时器(定时器 1)配置如下:

1. 设置 **AD16C6T1_AR** 寄存器的 ARV 位为 03h，当计数器递增到 3 后重载，并产生更新事件。
2. 设置 **AD16C6T1_CON2** 寄存器的 MMSEL 位为 010b，选择更新事件(UPD)为触发输出(TRGOUT)。
3. 设置 **AD16C6T1_CON1** 寄存器的 CNTEN 位为 1，开启计数器。

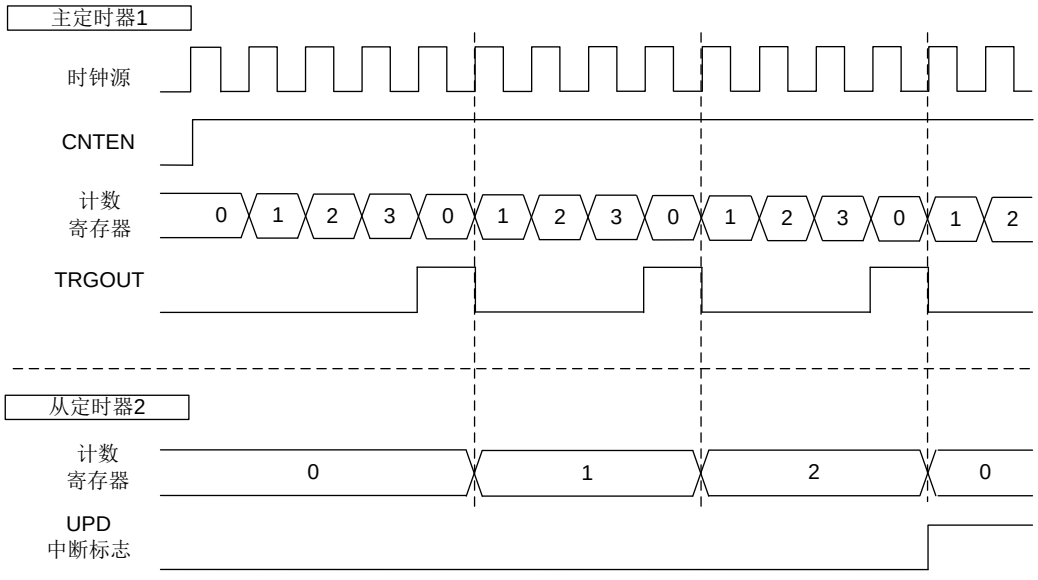


图 23-37　使用主定时器更新事件触发从定时器计数

22. 5. 22. 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(AD16C6T1)的 I1 输入上升沿时开启定时器 1，开启定时器 1 的同时开启定时器 2(GP16C2T1)。为保证计数器的对齐，定时器 1 必须设置为主/从模式(对应 I1 为从，对应定时器 2 为主):

先设定从定时器(定时器 2)为触发模式，配置如下:

1. 设置 **GP16C2T_SMCON** 寄存器的 TSSEL1 位与 TSSEL2 位为 AD16C6T1，可参考内部触发连接表。
2. 设置 **GP16C2T_SMCON** 寄存器的 SMODS 位为 110b，选择触发模式。

再设定主定时器(定时器 1)为触发模式，配置如下:

1. 设置定时器 1 为触发模式，相关配置可参考触发模式章节。
2. 设置 **AD16C6T_CON2** 寄存器的 MMSEL 位为 001b，选择开启信号(CNTEN)为触发输出(TRGOUT)。
3. 设置 **AD16C6T_SMCON** 寄存器的 MSCFG 位为 1，开启主/从模式。
4. 设置 **AD16C6T_CON1** 寄存器的 CNTEN 位为 1，开启计数器。

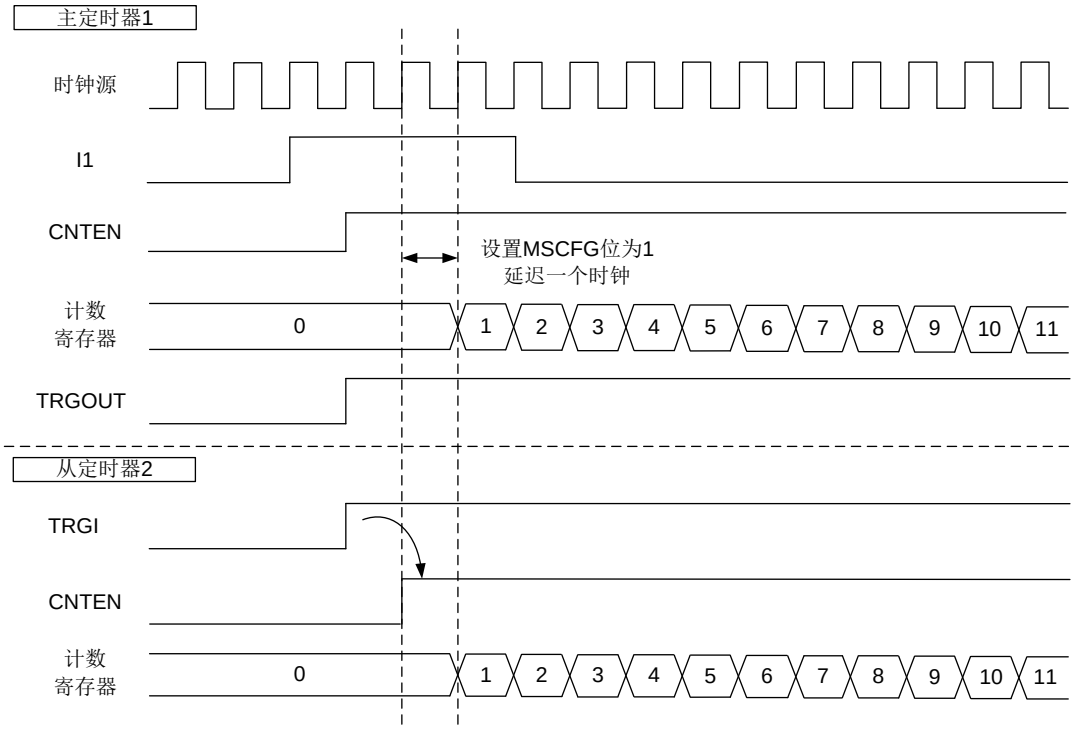


图 23-38　使用定时器 1 的 I1 输入触发定时器 1 和定时器 2

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 标志位也同时被设置。

22. 5. 23 ADC触发生成

定时器可生成 ADC 触发信号源如下:

- TRGOUT: 由 **GP16C2T_CON2** 寄存器的 MMSEL 位决定触发事件的来源，根据配置生成脉冲或电平信号。
- CHx: 通道输出的电平信号。

下图以 CH1 输出 PWM 模式 2 为例，说明 TRGOUT 与 CH1 信号源，相关配置如下:

1. 设置 **GP16C2T_AR** 寄存器的 ARV 位为 07h，当计数器递增到 7 后重载。
2. 设置 **GP16C2T_CHMR1** 寄存器的 CH1MOD 位为 111b，选择 PWM 模式 2。
3. 设置 **GP16C2T_CCVAL1** 寄存器的 CCRV1 位为 04h，当计数器数到 4 时，PWM 输出高电平。
4. 设置 **GP16C2T_CON2** 寄存器的 MMSEL 位为 100b，选择输出比较参考信号(CH1REF)为触发输出(TRGOUT)。

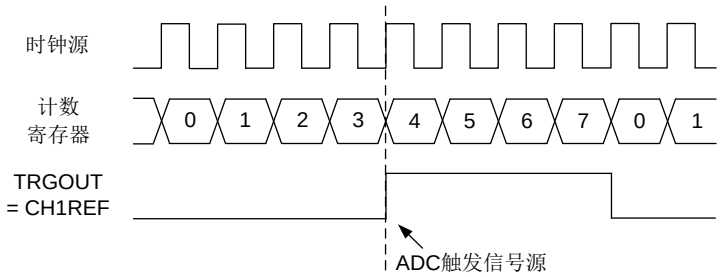


图 23-39　ADC 触发生成

22. 5. 24 红外线(IR)控制信号

芯片支持用户使用计数器搭配输出红外线(IR)的控制信号，藉此实现远程遥控的功能。在使用上需配置 GP16C2T1 与 GP16C2T2 的通道 1，决定如何搭配出 PWM 控制信号。此芯片目前仅支持用户从 GP16C2T1 的通道 1 与 GP16C2T2 的通道 1 输出进行逻辑 NAND 运算。

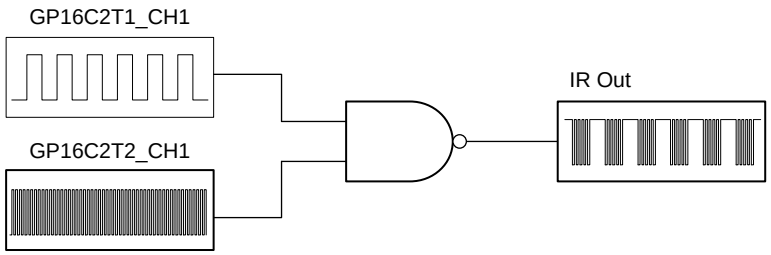


图 23-40　红外线控制信号组

22. 5. 25 调试模式

当微控制器进入调试模式(Cortex™-M0 核停止运行)，根据 SYSCFG 章节中 SYSCFG_CFG 寄存器配置，选择将计数器继续正常工作或停止计数。

22. 6 特殊功能寄存器

22. 6. 1 寄存器列表

GP16C2T 寄存器列表			
名称	偏移地址	类型	描述
GP16C2T_CON1	0000_H	R/W	控制寄存器 1
GP16C2T_CON2	0004_H	R/W	控制寄存器 2
GP16C2T_SMCON	0008_H	R/W	从模式控制寄存器
GP16C2T_IER	$000C_H$	W1	中断开启寄存器
GP16C2T_IDR	0010_H	W1	中断关闭寄存器
GP16C2T_IVS	0014_H	R	中断功能有效状态寄存器
GP16C2T_RIF	0018_H	R	原始中断状态寄存器
GP16C2T_IFM	$001C_H$	R	中断标志位状态寄存器
GP16C2T_ICR	0020_H	C_W1	中断清除寄存器
GP16C2T_SGE	0024_H	T_W1	软件生成事件寄存器
GP16C2T_CHMR1	0028_H	R/W	捕获或比较模式寄存器 1
GP16C2T_CCEP	0030_H	R/W	捕获或比较开启极性寄存器
GP16C2T_COUNT	0034_H	R/W	计数器寄存器
GP16C2T_PRES	0038_H	R/W	预分频寄存器
GP16C2T_AR	$003C_H$	R/W	自动重载寄存器
GP16C2T_REPAR	0040_H	R/W	重复计数寄存器
GP16C2T_CCVAL1	0044_H	R/W	通道捕获或比较寄存器 1
GP16C2T_CCVAL2	0048_H	R/W	通道捕获或比较寄存器 2
GP16C2T_BDCFG	0054_H	R/W	刹车和死区配置寄存器
GP16C2T_DMAEN	0058_H	R/W	DMA 事件开启寄存器
GP16C2T_DCFG2	0068_H	R/W	死区配置寄存器 2
GP16C2T_CHISEL	0070_H	R/W	通道输入来源选择寄存器
GP16C2T_AFR1	0074_H	R/W	复用功能寄存器 1
GP16C2T_AFR2	0078_H	R/W	复用功能寄存器 2

22. 6. 2 寄存器描述

22. 6. 2. 1 控制寄存器 1(GP16C2T_CON1)

控制寄存器 1(GP16C2T_CON1)																												
偏移地址:0x00																												
复位值:0x0000 0000																												
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																						DFCKSEL<1:0>		ARPEN				SPMEN	UERSEL	DISUE	CNTEN

—	Bit 31-12	—	—
UPDREMAP	Bit 11	R/W	**UPD原始中断状态重新映射** UPD原始中断状态复制到GP16C2T_COUNT寄存器的31位 0:重新映射关闭 1:重新映射开启
—	Bit 10	—	—
DFCKSEL	Bit 9-8	R/W	**时钟预分频器** 设置定时器频率(INT_CLK)，和死区产生器与数字滤波器所采样时钟之间的分频比例。 00: $t_{DTS}=t_{INT\_CLK}$ 01: $t_{DTS}=2 \times t_{INT\_CLK}$ 10: $t_{DTS}=4 \times t_{INT\_CLK}$ 11:保留
ARPEN	Bit 7	R/W	**自动重载预装载开启** 发生更新事件时，将预装载值载入到缓冲寄存器中 0:GP16C2T_AR 寄存器不具备缓冲 1:GP16C2T_AR 寄存器具备缓冲
—	Bit 6-4	—	—
SPMEN	Bit 3	R/W	**单脉冲模式** 0:单脉冲模式关闭，计数器不停止 1:单脉冲模式开启，计数器在发生下一次更新事件时，会自动清除 CNTEN 位，并停止计数器
UERSEL	Bit 2	R/W	**更新事件来源选择** 设置更新事件(UPD)的来源 0:下列事件都会产生更新中断或 DMA 的请求:

			－计数器上溢 －设置 GP16C2T_SGE 寄存器的 SGUPD 位为 1 －通过从模式控制器所生成的更新事件 1:只有在计数器上溢时会生成更新中断或 DMA 的请求
DISUE	Bit 1	R/W	**更新事件禁止** 0:更新事件(UPD)开启时，下列事件皆会产生更新事件请求并将预装载值加载到缓冲寄存器中 －计数器上溢 －设置 SGE 寄存器的 SGUPD 位为 1 －通过从模式控制器所生成的更新事件 1:更新事件(UPD)关闭时，不产生更新事件请求，GP16C2T_AR、GP16C2T_PRES 与 GP16C2T_CCVALn 寄存器的缓冲寄存器数值保持不变。但设置 GP16C2T_SGE 寄存器的 SGUPD 位为 1 或从模式控制器的复位请求，计数器和预分频器仍会被重新初始化
CNTEN	Bit 0	R/W	**计数器开启** 开启计数器后，在外部时钟模式、门控模式才能运作。而触发模式则可以由硬件将 CNTEN 位置 1 0:计数器关闭 1:计数器开启

22. 6. 2. 2 控制寄存器 2(GP16C2T_CON2)

控制寄存器 2(GP16C2T_CON2)																												
偏移地址:0x04																												
复位值:0x0000 0000																												
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																					OISS2	OISS1N	OISS1		MMSEL<2:0>			CCUSEL		CCPCEN	

—	Bits 31-11	—	—
OISS2	Bit 10	R/W	**通道 2 输出的空闲状态选择位** 参照 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:当 GP16C2T_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 时，如果 CH1N 已实现，则经过死区时间后，CH1=0 1:当 GOEN=0 时，如果 CH1N 已实现，则经过死区时间后，CH1=1 注:当 GP16C2T_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
I1SEL	Bit 7	R/W	**I1选择** 0:I1输入连接到GP16C2T_CH1引脚 1:I1输入连接到GP16C2T_CH1、CH2引脚异或组合(XOR)输出
MMSEL	Bit 6-4	R/W	**主模式选择 1** 选择在主模式下发送到从定时器的同步信号与 ADC 触发信号(TRGOUT) 000:复位 - 设置 GP16C2T_SGE 寄存器中的 UPD 位为触发输出。如果复位由触发输入生成(从模式控制器配置为复位模式)，则 TRGOUT 上的信号与实际复位信号之间会有延迟

			001:开启－设置 GP16C2T_CON1 寄存器中的 CNTEN 位为触发输出，可用于同步开启数个定时器。计数器开启信号可由 GP16C2T_CON1 寄存器中的 CNTEN 位或者门控模式下的触发输入产生。当计数器开启信号受控于触发输入时，TRGOUT 上的信号与实际触发信号之间会有延迟 010:更新事件－更新事件被用于触发输出。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－每次发生捕获或比较匹配时，当产生 CH1 中断请求同时，触发输出会送出一个正脉冲 100:比较信号－CH1REF 信号用于触发输出 101:比较信号－CH2REF 信号用于触发输出 其他:保留
CCDMASEL	Bit 3	R/W	**捕获或比较事件的 DMA 选择** 0:当发生 CHn 事件时，设置 CHn DMA 请求 1:当发生更新事件时，设置 CHn DMA 请求
CCUSEL	Bit 2	R/W	**捕获或比较更新事件来源选择** 0:捕获或比较预装载开启时(CCPCEN =1)，只能通过 GP16C2T_SGE 寄存器的 SGCOM 位为 1 时更新 1:捕获或比较预装载开启时(CCPCEN =1)，可通过 GP16C2T_SGE 寄存器的 SGCOM 位为 1 与 TRGI 的上升沿时被更新
—	Bit 1	—	—
CCPCEN	Bit 0	R/W	**捕获或比较预装载开启** GP16C2T_SGE 寄存器的 SGCOM 位设置为 1 或 TRGI 的上升沿时，发生换向事件(COM)并将预装载值载入到缓冲寄存器中 0:GP16C2T_CCEP 寄存器中的 CCnEN、CCnNEN 和 GP16C2T_CHMRn 寄存器中的 CHnMOD 位不具备缓冲 1:GP16C2T_CCEP 寄存器中的 CCnEN、CCnNEN 和 GP16C2T_CHMRn 寄存器中的 CHnMOD 位具备缓冲

22. 6. 2. 3 从模式控制寄存器(GP16C2T_SMCON)

从模式控制寄存器(GP16C2T_SMCON)																												
偏移地址:0x08																												
复位值:0x0000 0000																												
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
—	—	—	—	—	—	—	—	—	—	TSSEL2 <1:0>		—	—	—	—	—	—	—	—	—	—	—	—	MSCFG	TSSEL1<2:0>			—	SMODS<2:0>		

—	Bits 31-22	—	—
TSSEL2	Bit 21-20	W	**触发来源选择2** 参照TSSEL1描述
—	Bit 19-17	—	—
SMODS2	Bit 16	R/W	**从模式选择** 参照SMODS描述
—	Bit 15-8	—	—
MSCFG	Bit 7	R/W	**主/从模式** 延迟触发输入(TRGI)上的事件来允许当前主定时器和其从定时器之间的同步。该设置有效用于使用单个外部事件来同步多个定时器 0: 主从模式关闭 1: 主从模式开启
TSSEL1	Bit 6-4	R/W	**触发来源选择 1** 此位与 TSSEL2[1:0]组合成 TSSEL = {TSSEL2[1:0],TSSEL1[2:0]} 设置触发选择，用于同步计数器 00000:内部触发 0 (IT0) 00001:内部触发 1 (IT1) 00010:内部触发 2 (IT2) 00011:内部触发 3 (IT3) 00100: I1 边沿检测(I1F_ED) 00101: I1 滤波后信号(I1FP) 00110: I2 滤波后信号(I2FP) 00111:外部触发输入(ETP) 01000:内部触发 4 (IT4) 01001:内部触发 5 (IT5) 01010:内部触发 6 (IT6)

			01011:内部触发 7 (IT7) 其他:保留 注:此位在需要在使用前设定(SMODS=000)，以避免产生错误的上升/下降边沿至计数器
—	Bit 3	—	—
SMODS	Bit 2-0	R/W	**从模式选择** 此位与 SMODS2 组合成 SMODS = {SMODS2, SMODS[2:0]} 0000: 从模式关闭 - 当GP16C2T_CON1 寄存器 CNTEN 位为 1 时，预分频器时钟由内部时钟提供 0100: 复位模式 - 选中的触发输入(TRGI)的上升沿重新初始化计数器，并且产生一次更新事件 0101: 门控模式 - 当触发输入(TRGI)为高电平时，计数器的时钟开启。一旦触发输入变为低电平，则计数器停止(但不复位)。计数器的启动和停止都是被控制 0110: 触发模式 - 计数器在触发输入 TRGI 的上升沿启动(但不复位)，只有计数器的启动是被控制 0111: 外部时钟模式 1 - 选中的触发输入(TRGI)的上升沿提供计数器时钟 1000: 组合复位模式 + 触发模式 - 选中的触发输入(TRGI)的上升沿重新初始化计数器，并且产生一次更新事件启动计数器 1001: 组合门控模式 + 复位模式 - 当触发输入(TRGI)为高电平时，计数器的时钟开启。一旦触发输入变为低电平，则计数器停止且复位。计数器的启动和停止都是被控制 注:如果 I1 双边沿检测被选为触发输入(设置 TSSEL 为 00100)时，不能使用门控模式。I1F 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

22. 6. 2. 4 中断开启寄存器(GP16C2T_IER)

中断开启寄存器(GP16C2T_IER)																												
偏移地址:0x0C																												
复位值:0x0000 0000																												
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	CH2OV	CH1OV	—	BRK	TRGI	COM	—	—	CH2	CH1	UPD

—	Bits 31-11	—	—
CH2OV	Bit 10	W1	**开启通道 2 捕获溢出中断功能** 此位设置时，开启中断功能，硬件侦测通道 2 捕获溢出事件时发生中断
CH1OV	Bit 9	W1	**开启通道 1 捕获溢出中断功能** 此位设置时，开启中断功能，硬件侦测通道 1 捕获溢出事件时发生中断
—	Bit 8	—	—
BRK	Bit 7	W1	**开启刹车中断功能** 此位设置时，开启中断功能，硬件侦测刹车信号时发生中断
TRGI	Bit 6	W1	**开启触发中断功能** 此位设置时，开启中断功能，硬件侦测触发信号事件时发生中断
COM	Bit 5	W1	**开启 COM 中断功能** 此位设置时，开启中断功能，硬件侦测 COM 信号事件时发生中断
—	Bit 4-3	—	—
CH2	Bit 2	W1	**开启通道 2 捕获或比较匹配中断功能** 此位设置时，开启中断功能，硬件侦测通道 2 捕获或比较匹配事件时发生中断
CH1	Bit 1	W1	**开启通道 1 捕获或比较匹配中断功能** 此位设置时，开启中断功能，硬件侦测通道 1 捕获或比较匹配事件时发生中断
UPD	Bit 0	W1	**开启更新中断功能** 此位设置时，开启中断功能，硬件侦测更新事件时发生中断

22. 6. 2. 5 中断关闭寄存器(GP16C2T_IDR)

中断关闭寄存器(GP16C2T_IDR)																													
偏移地址: 0x10																													
复位值:0x0000 0000																													
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																					CH2OV	CH1OV		BRK	TRGI	COM			CH2	CH1	UPD

—	Bits 31-11	—	—
CH2OV	Bit 10	W1	**关闭通道 2 捕获溢出中断功能** 此位设置时，关闭通道 2 捕获溢出中断功能
CH1OV	Bit 9	W1	**关闭通道 1 捕获溢出中断功能** 此位设置时，关闭通道 1 捕获溢出中断功能
—	Bit 8	—	—
BRK	Bit 7	W1	**关闭刹车中断功能** 此位设置时，关闭刹车中断功能
TRGI	Bit 6	W1	**关闭触发中断功能** 此位设置时，关闭触发中断功能
COM	Bit 5	W1	**关闭 COM 中断功能** 此位设置时，关闭 COM 中断功能
—	Bit 4-3	—	—
CH2	Bit 2	W1	**关闭通道 2 捕获或比较匹配中断功能** 此位设置时，关闭通道 2 捕获或比较匹配中断功能
CH1	Bit 1	W1	**关闭通道 1 捕获或比较匹配中断功能** 此位设置时，关闭通道 1 捕获或比较匹配中断功能
UPD	Bit 0	W1	**关闭更新中断功能** 此位设置时，关闭更新中断功能

22. 6. 2. 6 中断功能有效状态寄存器(GP16C2T_IVS)

中断功能有效状态寄存器(GP16C2T_IVS)																													
偏移地址:0x14																													
复位值:0x0000 0000																													
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	CH2OV	CH1OV	—	BRK	TRGI	COM	—	—	CH2	CH1	UPD

—	Bits 31-11	—	—
CH2OV	Bit 10	R	**通道 2 捕获溢出中断功能状态** 0:中断功能处于关闭状态 1:中断功能处于开启状态
CH1OV	Bit 9	R	**通道 1 捕获溢出中断功能状态** 0:中断功能处于关闭状态 1:中断功能处于开启状态
—	Bit 8	—	—
BRK	Bit 7	R	**刹车中断功能状态** 0:中断功能处于关闭状态 1:中断功能处于开启状态
TRGI	Bit 6	R	**触发中断功能状态** 0:中断功能处于关闭状态 1:中断功能处于开启状态
COM	Bit 5	R	**COM 中断功能状态** 0:中断功能处于关闭状态 1:中断功能处于开启状态
—	Bit 4-3	—	—
CH2	Bit 2	R	**通道 2 捕获或比较匹配中断功能状态** 0:中断功能处于关闭状态 1:中断功能处于开启状态
CH1	Bit 1	R	**通道 1 捕获或比较匹配中断功能状态** 0:中断功能处于关闭状态 1:中断功能处于开启状态
UPD	Bit 0	R	**更新中断功能状态** 0:中断功能处于关闭状态 1:中断功能处于开启状态

GP16C2T_IVS 寄存器，是实时反映系统配置 GP16C2T_IER 与 GP16C2T_IDR 的中断开启状态。此寄存器状态是将 GP16C2T_IER 与 GP16C2T_IDR 进行硬件运算，公式如下:GP16C2T_IVS = GP16C2T_IER & ~GP16C2T_IDR

22. 6. 2. 7 原始中断状态寄存器(GP16C2T_RIF)

原始中断状态寄存器(GP16C2T_RIF)																													
偏移地址:0x18																													
复位值:0x0000 0000																													
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	CH2OV	CH1OV	—	BRK	TRGI	COM	—	—	CH2	CH1	UPD

—	Bits 31-11	—	—
CH2OV	Bit 10	R	**通道 2 捕获溢出，原始中断状态** 0:无发生中断 1:已发生中断
CH1OV	Bit 9	R	**通道 1 捕获溢出，原始中断状态** 0:无发生中断 1:已发生中断
—	Bit 8	—	—
BRK	Bit 7	R	**刹车，原始中断状态** 0:无发生中断 1:已发生中断
TRGI	Bit 6	R	**触发，原始中断状态** 0:无发生中断 1:已发生中断
COM	Bit 5	R	**COM，原始中断状态** 0:无发生中断 1:已发生中断
—	Bit 4-3	—	—
CH2	Bit 2	R	**通道 2 捕获或比较匹配，原始中断状态** 0:无发生中断 1:已发生中断
CH1	Bit 1	R	**通道 1 捕获或比较匹配，原始中断状态** 0:无发生中断 1:已发生中断
UPD	Bit 0	R	**更新，原始中断状态**

			0:无发生中断 1:已发生中断

22. 6. 2. 8 中断标志位状态寄存器(GP16C2T_IFM)

中断标志位状态寄存器(GP16C2T_IFM)																													
偏移地址:0x1C																													
复位值:0x0000 0000																													
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	CH2OV	CH1OV	—	BRK	TRGI	COM	—	—	CH2	CH1	UPD

—	Bits 31-11	—	—
CH2OV	Bit 10	R	**通道 2 捕获溢出，中断标志位状态** 0:无发生中断 1:已发生中断
CH1OV	Bit 9	R	**通道 1 捕获溢出，中断标志位状态** 0:无发生中断 1:已发生中断
—	Bit 8	—	—
BRK	Bit 7	R	**刹车，中断标志位状态** 0:无发生中断 1:已发生中断
TRGI	Bit 6	R	**触发，中断标志位状态** 0:无发生中断 1:已发生中断
COM	Bit 5	R	**COM，中断标志位状态** 0:无发生中断 1:已发生中断
—	Bit 4-3	—	—
CH2	Bit 2	R	**通道 2 捕获或比较匹配，中断标志位状态** 0:无发生中断 1:已发生中断
CH1	Bit 1	R	**通道 1 捕获或比较匹配，中断标志位状态** 0:无发生中断 1:已发生中断
UPD	Bit 0	R	**更新，中断标志位状态** 0:无发生中断

			1:已发生中断

GP16C2T_IFM 寄存器，是滤除已关闭中断功能的中断事件，只关注开启中断功能的事件。此寄存器状态是将 GP16C2T_RIF 与 GP16C2T_IVS 进行硬件运算，公式如下:GP16C2T_IFM = GP16C2T_RIF &GP16C2T_IVS

22. 6. 2. 9 中断清除寄存器(GP16C2T_ICR)

中断清除寄存器(GP16C2T_ICR)

偏移地址:0x20

复位值:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	CH2OV	CH1OV	—	BRK	TRGI	COM	—	—	CH2	CH1	UPD

—	Bits 31-11	—	—
CH2OV	Bit 10	C_W1	**清除通道 2 捕获溢出中断状态** 此位设置时，清除中断状态(GP16C2T_RIF 与 GP16C2T_IFM)
CH1OV	Bit 9	C_W1	**清除通道 1 捕获溢出中断状态** 此位设置时，清除中断状态(GP16C2T_RIF 与 GP16C2T_IFM)
—	Bit 8	—	—
BRK	Bit 7	C_W1	**清除刹车中断状态** 此位设置时，清除中断状态(GP16C2T_RIF 与 GP16C2T_IFM)
TRGI	Bit 6	C_W1	**清除触发中断状态** 此位设置时，清除中断状态(GP16C2T_RIF 与 GP16C2T_IFM)
COM	Bit 5	C_W1	**清除 COM 中断状态** 此位设置时，清除中断状态(GP16C2T_RIF 与 GP16C2T_IFM)
—	Bit 4-3	—	—
CH2	Bit 2	C_W1	**清除通道 2 捕获或比较匹配中断状态** 此位设置时，清除中断状态(GP16C2T_RIF 与 GP16C2T_IFM)
CH1	Bit 1	C_W1	**清除通道 1 捕获或比较匹配中断状态** 此位设置时，清除中断状态(GP16C2T_RIF 与

			GP16C2T_IFM)
UPD	Bit 0	C_W1	**清除更新中断状态** 此位设置时，清除中断状态(GP16C2T_RIF 与 GP16C2T_IFM)

GP16C2T_ICR 寄存器设置时，将清除 GP16C2T_RIF 与 GP16C2T_IFM 中断标志位状态；此设置不影响中断 GP16C2T_IER、GP16C2T_IDR 与 GP16C2T_IVS 寄存器，只清除标志位状态 GP16C2T_RIF 与 GP16C2T_IFM。此寄存器通过硬件清除中断，公式如下:GP16C2T_RIF = GP16C2T_RIF & ~GP16C2T_ICR

22. 6. 2. 10 软件生成事件寄存器(GP16C2T_SGE)

软件生成事件寄存器(GP16C2T_SGE)

偏移地址:0x24

复位值:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																								SGBRK	SGTRGI	SGCOM	—	—	SGCH2	SGCH1	SGUPD

—	Bits 31-8	—	—
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。 此位设置时，产生刹车事件。GOEN清零，BRK标志位置起，产生相关中断。
SGTRGI	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。 此位设置时，产生触发事件。产生相关中断或 DMA 传输
SGCOM	Bit 5	W1	**软件生成 COM 事件** 该位由软件设置来生成 COM 事件，可由硬件自动清零。 此位设置时，产生 COM 事件。当 CCPCEN 被置 1，则可更新 CCnEN, CCnNEN 和 CHnMOD。 注:位只有用作于通道时有互补输出
—	Bit 4-3	—	—

SGCH2	Bit 2	W1	**软件生成通道 2 捕获/比较事件** 参照 SGCH1 描述
SGCH1	Bit 1	W1	**软件生成通道 1 捕获/比较事件** 该位由软件设置来生成通道 1 捕获或比较，可由硬件自动清零。 **通道 CH1 设置为输出:** 此位设置时，产生比较事件，但不影响输出。若开启中断或 DMA，则产生中断与请求。 **通道 CH1 设置为输入:** 此位设置时，产生捕获事件。将计数器捕获至 CCVAL1 寄存器中，于 I1 的有效沿产生，若开启中断或 DMA，则产生中断与请求。
SGUPD	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。 此位设置时，产生更新事件。重新初始化计数器，更新寄存器。 注:预分频器也会被清零(但预分频比不会受到影响)。

22. 6. 2. 11 捕获或比较模式寄存器 1(GP16C2T_CHMR1)

捕获或比较模式寄存器 1(GP16C2T_CHMR1)																													
偏移地址:0x28																													
复位值:0x0000 0000																													
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
							CH2MOD2								CH1MOD2	CH2OCLREN	CH2MOD<2:0>			CH2PEN	CH2FEN	CC2SSEL<1:0>		CH1OCLREN	CH1MOD<2:0>			CH1PEN	CH1FEN	CC1SSEL<1:0>	
																I2FLT<3:0>				I2PRES<1:0>				I1FLT<3:0>				I1PRES<1:0>		

输出比较模式

—	Bits 31-25	—	—
CH2MOD2	Bit 24	R/W	**输出比较通道2模式** 参照CH1MOD描述
—	Bits 23-17	—	—
CH1MOD2	Bit 16	R/W	**输出比较通道1模式** 参照CH1MOD描述
CH2OCLREN	Bit 15	R/W	**输出比较通道2清除开启** 参照CH1OCLREN描述
CH2MOD	Bit 14-12	R/W	**输出比较通道 2 模式** 参照 CH1MOD 描述
CH2PEN	Bit 11	R/W	**输出比较通道 2 预装载开启** 参照 CH1PEN 描述
CH2FEN	Bit 10	R/W	**输出比较通道 2 快速开启** 参照 CH1FEN 描述
CC2SSEL	Bit 9-8	R/W	**捕获或比较通道 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(只工作在 GP16C2T_SMCON 寄存器的 TSSEL 位选择为 ITn 时)

CH1OCLREN	Bit 7	R/W	**输出比较通道1清除开启** 0:CH1REF维持输出 1:CH1REF在OCLR_INT(由OCLRSEL选择)为高电平时清除
CH1MOD	Bit 6-4	R/W	**输出比较通道 1 模式** 此位与 CH1MOD2 组合成 CH1MOD[3:0] = {CH1MOD2, CH1MOD[2:0]} 这些位定义提供 CH1 和 CH1N 的输出参考信号 CH1REF 的行为。CH1REF 为高电平有效，而 CH1 和 CH1N 的有效电平则取决于 GP16C2T_CCEP 寄存器中的 CC1POL 位和 CC1NPOL 位 0000:关闭－无作用 0001:匹配时设置高电平－当计数器 (GP16C2T_COUNT)匹配 GP16C2T_CCVAL1 寄存器时，CH1REF 设置为 1 0010:匹配时设置低电平－当计数器 (GP16C2T_COUNT)匹配 GP16C2T_CCVAL1 寄存器时，CH1REF 设置为 0 0011:匹配时翻转电平－当计数器 (GP16C2T_COUNT)匹配 GP16C2T_CCVAL1 寄存器时，CH1REF 翻转电平(当前为高电平则翻转成低电平，反之当前为低电平则翻转成高电平) 0100:强制低电平－CH1REF 强制设置低电平 0101:强制高电平－CH1REF 强制设置高电平 0110:PWM 模式 1－在递增计数时，当计数器 (GP16C2T_COUNT)小于 GP16C2T_CCVAL1 寄存器时，输出高电平，其他则输出低电平。 0111:PWM 模式 2－在递增计数时，当计数器 (GP16C2T_COUNT)小于 GP16C2T_CCVAL1 寄存器时，输出低电平，其他则输出高电平。 1000: 多次触发单脉冲模式 1－在递增计数时，通道保持为高电平，直到在 TRGI 信号上侦测到触发事件后，通道正常输出 PWM 模式 1，直到下次更新事件发生后，通道再次保持高电平。 1001: 多次触发单脉冲模式 2－在递增计数时，通道保持为低电平，直到在 TRGI 信号上侦测

			到触发事件后，通道正常输出 PWM 模式 2，直到下次更新事件发生后，通道再次保持低电平。 1010: 保留 1011: 保留 1100: 组合 PWM 模式 1 - 在 PWM 模式 1 下产生 CH1REF，再将 CH1REF 和 CH2REF 做"OR"运算输出 CH1REFC 信号 1101: 组合 PWM 模式 2 - 在 PWM 模式 2 下产生 CH1REF，再将 CH1REF 和 CH2REF 做"AND"运算输出 CH1REFC 信号 1110: 保留 1111: 保留 注:当设置 GP16C2T_BDCFG 寄存器中的 LOCKLVL 位为锁定级别 3 且 CC1SSEL = 00(通道设置为输出)后，此位无法更改。
CH1PEN	Bit 3	R/W	**输出比较通道 1 预装载开启** 设置后在更新事件时，将 GP16C2T_CCVAL1 寄存器预装载值载入到缓冲寄存器中。 0: CCVAL1 寄存器预装载关闭 1: CCVAL1 寄存器预装载开启 注:当设置 GP16C2T_BDCFG 寄存器中的 LOCKLVL 位为锁定级别 3 且 CC1SSEL = 00(通道设置为输出)后，此位无法更改。
CH1FEN	Bit 2	R/W	**输出比较通道 1 快速开启** 用于加速触发输入事件对于 PWM 输出的影响，CH1FEN 只在通道被配置成 PWM1 或 PWM2 模式时起作用。 0:CH1 的输出依赖于计数器与 CCVAL1 的值正常工作。 1:当触发输入(TRGI)有效时，CH1 被强制设置为比较电平(与比较结果无关)，触发输入(TRGI)的有效边沿相当于发生比较匹配。
CC1SSEL	Bit 1-0	R/W	**捕获或比较通道 1 选择** 设置通道的输出方向与信号的选择，当 GP16C2T_CCEP 寄存器的 CC1EN 位为 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(只工作在 GP16C2T_SMCON 寄存器的 TSSEL 位选择为 ITn 时)

输入捕获模式

—	Bits 31-16	—	—
I2FLT	Bit 15-12	R/W	**输入捕获通道2滤波器** 参照I1FLT描述
I2PRES	Bit 11-10	R/W	**输入捕获通道 2 预分频器** 参照 IC1PRES 描述
CC2SSEL	Bit 9-8	R/W	**捕获或比较通道 2 选择** 设置通道的输出方向与信号的选择，当 GP16C2T_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(只工作在 GP16C2T_SMCON 寄存器的 TSSEL 位选择为 ITn 时)
I1FLT	Bit 7-4	R/W	**输入捕获通道 1 滤波器** 设置 I1 信号采样的频率和对 I1 数字滤波器的带宽。数字滤波器是一个事件计数器，它记录到 N 个事件后才视为一个有效输出边沿: 0000:采样频率 f_{DTS}，滤波器禁止 0001:采样频率 f_{INT_CLK}, N = 2 0010:采样频率 f_{INT_CLK}, N = 4 0011:采样频率 f_{INT_CLK}, N = 8 0100:采样频率 f_{DTS} / 2, N = 6 0101:采样频率 f_{DTS} / 2, N = 8 0110:采样频率 f_{DTS} / 4, N = 6 0111:采样频率 f_{DTS} / 4, N = 8 1000:采样频率 f_{DTS} / 8, N = 6 1001:采样频率 f_{DTS} / 8, N = 8 1010:采样频率 f_{DTS} / 16, N = 5 1011:采样频率 f_{DTS} / 16, N = 6 1100:采样频率 f_{DTS} / 16, N = 8

			1101:采样频率 f_{DTS} / 32, N = 5 1110:采样频率 f_{DTS} / 32, N = 6 1111:采样频率 f_{DTS} / 32, N = 8
I1PRES	Bit 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 GP16C2T_CCEP 寄存器的 CC1EN 位，预分频计数器同时被清除 00:预分频关闭，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	Bit 1-0	R/W	**捕获或比较通道 1 选择** 设置通道的输出方向与信号的选择，当 GP16C2T_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(只工作在 GP16C2T_SMCON 寄存器的 TSSEL 位选择为 ITn 时)

22. 6. 2. 12 捕获或比较开启极性寄存器(GP16C2T_CCEP)

捕获或比较开启极性寄存器(GP16C2T_CCEP)																													
偏移地址:0x30																													
复位值:0x0000 0000																													
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																								CC2NPOL		CC2POL	CC2EN	CC1NPOL	CC1NEN	CC1POL	CC1EN

—	Bits 31-8	—	—
CC2NPOL	Bit 7	R/W	**捕获或比较通道 2 互补输出极性** 参照 CC1NPOL 描述
—	Bit 6	—	—
CC2POL	Bit 5	R/W	**捕获或比较通道 2 输出极性** 参照 CC1POL 描述
CC2EN	Bit 4	R/W	**捕获或比较通道 2 输出开启** 参照 CC1EN 描述
CC1NPOL	Bit 3	R/W	**捕获或比较通道 1 互补输出极性** **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。参照 CC1POL 描述。 注 1:对于有互补输出的通道，该位设置为预载值。如果 GP16C2T_CON2 寄存器中的 CCPCEN 位设置为 1，则只有当 COM 事件发生时，CC1NPOL 位才会设置为预载值中新的值。 注 2:当 GP16C2T_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。
CC1NEN	Bit 2	R/W	**捕获或比较通道 1 互补输出开启** 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 GP16C2T_CON2 寄存器中的 CCPCEN 位设置为 1，则只有当 COM 事件发生时，CC1NEN 缓冲位才会设置为预载值中新的值
CC1POL	Bit 1	R/W	**捕获或比较通道 1 输出极性** **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 与 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 1:对于有互补输出的通道，该位设置为预载值。如果 GP16C2T_CON2 寄存器中的 CCPCEN 位设置为 1，则只有当 COM 事件发生时，CC1POL 缓冲位才会设置为预载值中新的值。 注 2:当 GP16C2T_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	**捕获或比较通道 1 输出开启** **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 CC1NEN 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 CC1NEN 决定 **通道 CH1 设置为输入:** 0:捕获关闭 1:捕获开启

			注:对于有互补输出的通道，该位设置为预载值。如果 GP16C2T_CON2 寄存器中的 CCPCEN 位设置为 1，则只有当 COM 事件发生时，CC1EN 缓冲位才会设置为预载值中新的值。

22. 6. 2. 13 计数寄存器(GP16C2T_COUNT)

计数寄存器(GP16C2T_COUNT)																												
偏移地址:0x34																												
复位值:0x0000 0000																												
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
UPDRIF																CNTV <15:0>														

UPDRIF	Bits 31	R	**更新原始中断状态映射** 该位为射GP16C2T_RIF寄存器的UPD位的状态，该位只能读取，表示计数器发生更新事件。设置GP16C2T_CON1寄存器的UPDFREMAP位为1开启功能。
—	Bits 30-16	—	—
CNTV	Bits 15-0	R/W	**计数器数值**

22. 6. 2. 14 预分频寄存器(GP16C2T_PRES)

预分频寄存器(GP16C2T_PRES)																														
偏移地址:0x38																														
复位值:0x0000 0000																														
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																PSCV <15:0>															

—	Bits 31-16	—	—
PSCV	Bits 15-0	R/W	**预分频数值** 当计数器时钟频率等于f_{INT_CLK}/(PSCV<15:0> + 1)时计数器递增或递减。在更新事件产生时，将PSCV数值被加载预装载寄存器中

22. 6. 2. 15 自动重载寄存器(GP16C2T_AR)

自动重载寄存器(GP16C2T_AR)																														
偏移地址:0x3C																														
复位值:0x0000 FFFF																														
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
																ARV <15:0>															

—	Bits 31-16	—	—
ARV	Bits 15-0	R/W	**自动装载数值** 设置计数器的递增计数的边界或递减计数的重载值

22. 6. 2. 16 重复计数寄存器(GP16C2T_ REPAR)

重复计数寄存器(GP16C2T_ REPAR)																													
偏移地址:0x40																													
复位值:0x0000 0000																													
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																								REPV <7:0>							

—	Bits 31-8	—	—
REPV	Bits 7-0	R/W	**重复计数数值** 当预载寄存器开启，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。当REPV_CNT计数器递减至0时，会产生更新事件，同时重新从REPV值递减计数。REPV_CNT具备缓冲，因此只有当发生更新事件时，才会将REPV重载到REPV_CNT中，即任何对于REPV的写入，只会在下一次更新事件后才会生效。 - 在边沿对齐模式下的PWM输出，(REPV+1)对应的是PWM的周期数

22. 6. 2. 17 通道捕获或比较寄存器 1(GP16C2T_CCVAL1)

通道捕获或比较寄存器 1(GP16C2T_CCVAL1)																												
偏移地址:0x44																												
复位值:0x0000 0000																												
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																CCRV1 <15:0>															

—	Bits 31-16	—	—
CCRV1	Bits 15-0	R/W	**捕获或比较数值 1** **通道 CH1 配置为输出:** CCRV1 是捕获或比较寄存器的预装载值。 开启 GP16C2T_CHMR1 寄存器中的CH1PEN 预装载功能时，CCRV1 具备缓冲，当发生更新事件后，将预装载值载入到缓冲寄存器中。 CCRV1 的值会与 GP16C2T_COUNT 中的值进行比较，其结果会反应在 CH1REF。 **通道 CH1 配置为输入:** CCRV1为由先前发生输入捕获事件(I1)时的计数器数值。

22. 6. 2. 18 通道捕获或比较寄存器 2(GP16C2T_CCVAL2)

通道捕获或比较寄存器 2(GP16C2T_CCVAL2)																												
偏移地址:0x48																												
复位值:0x0000 0000																												
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
																CCRV2<15:0>															

—	Bits 31-16	—	—
CCRV2	Bits 15-0	R/W	**捕获或比较数值2** 参照CCRV1描述

22. 6. 2. 19 刹车和死区配置寄存器(GP16C2T_BDCFG)

刹车和死区配置寄存器(GP16C2T_BDCFG)																													
偏移地址:0x54																													
复位值:0x0000 0000																													
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
			BRKBID		BRKDSRM							BRKFLT<3:0>				GOEN	AOEN	BRKP	BRKEN	OFFSSR	OFFSSI	LOCKLVL<1:0>		DT<7:0>							

—	Bits 31-29	—	—
BRKBID	Bit 28	R/W	**BKIN双向开启** 当配置成双向模式时，BKIN引脚必须设定为开漏输出模式，任何有效的刹车事件都会使BKIN引脚改为输出低电平，以向外部设备指示发生刹车事件。 0: BKIN引脚为输入模式 1: BKIN引脚为双向模式 注:当GP16C2T_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
—	Bits 27	—	—
BRKDSRM	Bit 26	R/W	**解除BKIN刹车事件** 当解除刹车事件后，此位由硬件自动清除。 设置BRKDSRM位1，以解除刹车事件并释放双向控制，软件需要持续读取此位，直到由硬件清除，指示刹车事件已消失 0: BKIN引脚输入为警备状态，可立即响应刹车事件 1: BKIN引脚输入为非警备状态，无法响应刹车事件
—	Bits 25-20	—	—
BRKFLT	Bit 19-16	R/W	**刹车滤波器** 设置BKIN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个事件后才视为一个有效输出边沿: 0000: 采样频率f_{DTS}，滤波器禁止 0001: 采样频率f_{INT_CLK}, N = 2 0010: 采样频率f_{INT_CLK}, N = 4

			0011: 采样频率f_{INT_CLK}, N = 8 0100: 采样频率f_{DTS} / 2, N = 6 0101: 采样频率f_{DTS} / 2, N = 8 0110: 采样频率f_{DTS} / 4, N = 6 0111: 采样频率f_{DTS} / 4, N = 8 1000: 采样频率f_{DTS} / 8, N = 6 1001: 采样频率f_{DTS} / 8, N = 8 1010: 采样频率f_{DTS} / 16, N = 5 1011: 采样频率f_{DTS} / 16, N = 6 1100: 采样频率f_{DTS} / 16, N = 8 1101: 采样频率f_{DTS} / 32, N = 5 1110: 采样频率f_{DTS} / 32, N = 6 1111: 采样频率f_{DTS} / 32, N = 8 注:当GP16C2T_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
GOEN	Bit 15	R/W	**通道主要输出开启** 一旦BKIN或BKIN2刹车输入有效，该位会由硬件异步清零。该位可由软件设置为1或自动设置为1(取决于AOEN位)。该位仅作用于配置为输出的通道。 0: 若此位因BKIN2刹车事件清0，则禁止CHn/CHnN输出(此时引脚不由定时器控制) 若此位因BKIN刹车事件或软件设置清0，则根据OFFSSI的设定，禁止CHn/CHnN输出或输出空闲电平(根据GP16C2T_CON2寄存器的OISSn/OISSnN设定) 1: 如果CHn和CHnN各自的开启位都设置为1(GP16C2T_CCEP寄存器中的CCnEN和CCnNEN位)，则开启CHn和CHnN输出。
AOEN	Bit 14	R/W	**通道自动输出开启** 在发生更新事件时，将 GOEN 位置起 0: GOEN 位仅可由软件设置为 1 1: GOEN 位可由软件设置为 1，也可以在下一个更新事件发生且刹车输入无效时自动设置为 1。 注: 当 GP16C2T_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 BKIN 刹车极性** 0: 刹车输入 BKIN 为低电平有效 1: 刹车输入 BKIN 为高电平有效 注: 当 GP16C2T_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。
BRKEN	Bit 12	R/W	**开启刹车** 0: 刹车输入关闭 1: 刹车输入开启 注: 当 GP16C2T_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。
OFFSSR	Bit 11	R/W	**运行模式下关闭状态选择** 此位在 GOEN 为 1 且设置为输出模式并具有互补输出的通道。 0: 当通道关闭时(CCnEN/CCnNEN 位为 0)，禁止 CHn/CHnN 引脚输出(此时不由定时器控制) 1: 当通道关闭时(CCnEN/CCnNEN 位为 0)，一旦其互补通道为开启状态，会将输出设为无效电平(此时输出还是由定时器控制)。 注: 当 GP16C2T_BDCFG 寄存器中的 LOCKLVL 位已被设置为锁定级别 2，则该位无法修改。
OFFSSI	Bit 10	R/W	**空闲模式下关闭状态选择** 此位只适用在 GOEN 位因刹车事件或软件写入为 0，且设置为输出模式并具有互补输出的通道。 0: 当通道关闭时(CCnEN/CCnNEN 位为 0)，禁止 CHn/CHnN 引脚输出(此时不由定时器控制) 1: 当通道关闭时(CCnEN/CCnNEN 位为 0)，会先输出其无效电平，并在死区时间后，改为输出空闲电平(根据 GP16C2T_CON2 寄存器的 OISSn/OISSnN 设定)。 注: 当 GP16C2T_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 – GP16C2T_CON2 寄存器中的 OISSn 和 OISSnN、GP16C2T_BDCFG 寄存器中的 BRKBID/BRKEN/BRKP/AOEN/DT、GP16C2T_DCFG2 寄存器、GP16C2T_AFR1 寄存器和 GP16C2T_AFR2 寄存器不再可写 10: 锁定级别 2 – 锁定级别 1 + GP16C2T_BDCFG 寄存器中的 OFFSSR 和 OFFSSI、GP16C2T_CCEP 寄存器中的 CCnPOL 和 CCnNPOL(通道配置成输出模式时)不再可写 11: 锁定级别 3 – 锁定级别 2 + GP16C2T_CHMRn 寄存器中的 CHnMOD 和 CHnPEN(通道配置成输出模式时)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	Bit 7-0	R/W	**死区延迟** 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])x t_{dtg}，式中 $t_{dtg}=2xt_{DTS}$。 DT[7:5]=110 ⇒ DT=(32+DT[4:0])x t_{dtg}，式中 $t_{dtg}=8xt_{DTS}$。 DT[7:5]=111 ⇒ DT=(32+DT[4:0])x t_{dtg}，式中 $t_{dtg}=16xt_{DTS}$。 注: 当 GP16C2T_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

22. 6. 2. 20 DMA事件开启寄存器(GP16C2T_DMAEN)

DMA 事件开启寄存器 (GP16C2T_DMAEN)																														
偏移地址:0x58																														
复位值:0x0000 0000																														
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
																									TRGI	COM			CH2	CH1	UPD

—	Bits 31-7	—	—
TRGI	Bit 6	R/W	**触发事件的DMA请求开启** 0: 触发事件的DMA请求关闭 1: 触发事件的DMA请求开启
COM	Bit 5	R/W	**COM事件的DMA请求开启** 0: COM事件的DMA请求关闭 1: COM事件的DMA请求开启
—	Bits 4-3	—	—
CH2	Bit 2	R/W	**通道2捕获或比较匹配事件的DMA请求开启** 0: 通道2捕获或比较匹配事件的DMA请求关闭 1: 通道2捕获或比较匹配事件的DMA请求开启
CH1	Bit 1	R/W	**通道1捕获或比较匹配事件的DMA请求开启** 0: 通道1捕获或比较匹配事件的DMA请求关闭 1: 通道1捕获或比较匹配事件的DMA请求开启
UPD	Bit 0	R/W	**更新事件的 DMA 请求开启** 0: 更新事件的 DMA 请求关闭 1: 更新事件的 DMA 请求开启

22. 6. 2. 21 死区配置寄存器 2 (GP16C2T_DCFG2)

死区配置寄存器 2 (GP16C2T_DCFG2)																													
偏移地址:0x68																													
复位值:0x0000 0000																													
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
														DTPEN	DTAEN									DT2<7:0>							

—	Bits 31-18	—	—
DTPEN	Bit 17	R/W	**死区延迟预装载开启** 0: 死区延迟不具备缓冲 1: 死区延迟具备缓冲 注:当GP16C2T_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 注:此位仅在CNTEN关闭时修改
DTAEN	Bit 16	R/W	**非对称死区延迟开启** 0: 上升沿与下降沿的死区延迟一致，参照DT配置 1: 上升沿的死区延迟参照DT配置，而下降沿的死区延迟参照DT2配置 注:当GP16C2T_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 注:此位仅在CNTEN关闭时修改，若在CNTEN开启时配置，此位会自动清0
—	Bits 15-8	—	—
DT2	Bit 7-0	R/W	**非对称死区延迟** 参照 DT 描述 注:此位仅在 CNTEN 关闭时修改

22. 6. 2. 22 通道输入来源选择寄存器 (GP16C2T_CHISEL)

通道输入来源选择寄存器 (GP16C2T_CHISEL)																												
偏移地址:0x70																												
复位值:0x0000 0000																												
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
																				I2SEL<3:0>								I1SEL<3:0>			

—	Bits 31-12	—	—
I2SEL	Bit 11-8	R/W	**CH2输入来源选择** 0000: CH2引脚输入 其他: 参考引脚输入源章节
—	Bits 7-4	—	—
I1SEL	Bit 3-0	R/W	**CH1 输入来源选择** 0000: CH1 引脚输入 其他: 参考引脚输入源章节

22. 6. 2. 23 复用功能寄存器 1 (GP16C2T_AFR1)

复用功能寄存器 1 (GP16C2T_AFR1)																												
偏移地址:0x74																												
复位值:0x0000 0001																												
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
																			BRKEXTP	BRKCMP2P	BRKCMP1P	BRKINP						BRKEXTEN	BRKCMP2EN	BRKCMP1EN	BRKINEN

—	Bits 31-13	—	—
BRKEXTP	Bit 12	R/W	**选择通道刹车BKIN_EXT极性** 0:BKIN_EXT引脚输入不反相 1:BKIN_EXT引脚输入反相 注1:当GP16C2T_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
BRKCMP2P	Bit 11	R/W	**选择通道刹车CMP2极性** 0:刹车输入CMP2为高电平有效 1:刹车输入CMP2为低电平有效

			注:当GP16C2T_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
BRKCMP1P	Bit 10	R/W	**选择通道刹车CMP1极性** 0:刹车输入CMP1为高电平有效 1:刹车输入CMP1为低电平有效 注:当GP16C2T_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
BRKINP	Bit 9	R/W	**选择通道刹车BKIN极性** 0:BKIN引脚输入不反相 1:BKIN引脚输入反相 注1:当GP16C2T_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2:最终刹车有效电平需搭配GP16C2T_BDCFG寄存器中的BRKP极性决定
—	Bits 8-4	—	—
BRKEXTEN	Bit 3	R/W	**通道刹车BKIN_EXT开启** BKIN_EXT输出与其他刹车来源做"OR"运算 0:刹车输入BKIN_EXT禁止 1:刹车输入BKIN_EXT开启 注:当GP16C2T_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
BRKCMP2EN	Bit 2	R/W	**通道刹车CMP2开启** CMP2输出与其他刹车来源做"OR"运算 0:刹车输入CMP2禁止 1:刹车输入CMP2开启 注:当GP16C2T_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
BRKCMP1EN	Bit 1	R/W	**通道刹车CMP1开启** CMP1输出与其他刹车来源做"OR"运算 0:刹车输入CMP1禁止 1:刹车输入CMP1开启 注:当GP16C2T_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
BRKINEN	Bit 0	R/W	**通道刹车 BKIN 开启** BKIN 输出与其他刹车来源做"OR"运算 0:刹车输入 BKIN 禁止 1:刹车输入 BKIN 开启

			注:当 GP16C2T_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

22. 6. 2. 24 复用功能寄存器 2 (GP16C2T_AFR2)

复用功能寄存器 2 (GP16C2T_AFR2)

偏移地址:0x78

复位值:0x0000 0000

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
													OCLRSEL<2:0>																		

—	Bits 31-19	—	—
OCLRSEL	Bit 18-16	R/W	**外部清除来源选择** 000: CMP1 001: CMP2 其他: 保留 注:当GP16C2T_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
—	Bits 15-0	—	—

第23章　基本定时器 16 位(BS16T)

23. 1 概述

基本定时器 16 位(BS16T)包含一个 16 位计数器，该计数器由可配置的预分频器驱动。

23. 2 特性

- 一种 16 位自动重载计数器模式
  - ◇ 递增
- 16 位可配置预分频器，可在定时器运行时对计数器工作时钟进行 1 到 65536 之间的任意分频
- 下列事件支持产生中断与 DMA 请求:
  - ◇ 更新事件:计数器上溢，计数器初始化

23. 3 结构图

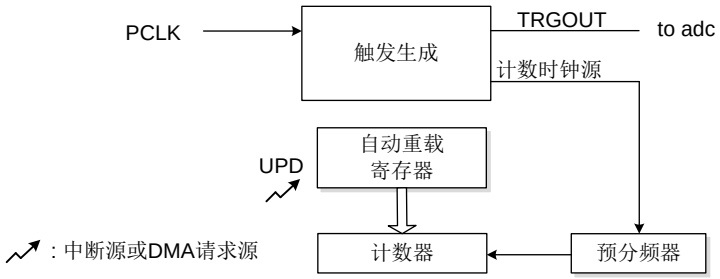


图 24-1　BS16T 定时器结构框图

23. 4 功能描述

23. 4. 1 定时单位

定时器包含一个 16 位的计数器 **BS16T_COUNT**，计数时钟由预分频寄存器 **BS16T_PRES** 进行分频。计数周期由自动重载计数器 **BS16T_AR** 设定。

自动重载寄存器 **BS16T_AR** 是一个可缓冲的寄存器。设置 **BS16T_CON1** 寄存器的 ARPEN 位为 0 时，关闭 **BS16T_AR** 寄存器缓冲功能，写入 **BS16T_AR** 寄存器的值会被立即反应到缓冲寄存器中；而设置 ARPEN 位为 1 时，**BS16T_AR** 寄存器具有缓冲功能，只有当产生更新事件(UPD)时，**BS16T_AR** 寄存器的重载值才会被更新到缓冲寄存器中。

设置 **BS16T_CON1** 寄存器的 DISUE 位为 0 时，计数器递增计数达到上溢值时会产生更新事件(UPD)。另外可以通过 **BS16T_SGE** 寄存器的 SGUPD 位为 1 产生软件更新事件。设置 **BS16T_CON1** 寄存器的 CNTEN 为 1 时，计数器开始计数。

注:计数器在设置 CNTEN 位为 1 后，在 1 个定时器时钟周期后开始计数。

预分频器可对定时器工作时钟进行 **BS16T_PRES** 寄存器数值+1 次分频。由于 **BS16T_PRES** 是一个可缓冲寄存器，因此定时器运行时可以对该寄存器进行修改，修改值在下次更新事件(UPD)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。

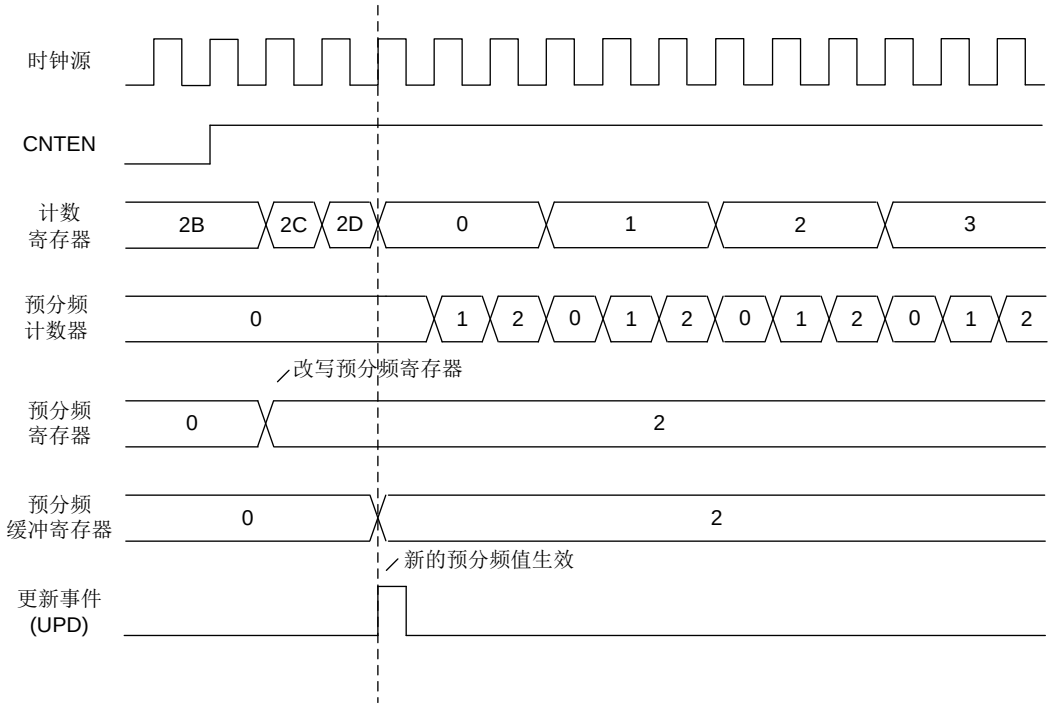


图 24-2 从 1 分频变为 3 分频时的计数时序图

23. 4. 2 时钟源

计数器工作时钟可以选择内部时钟(INT_CLK)

23. 4. 2. 1 内部时钟源(INT_CLK)

BS16T_CON1 寄存器的 CNTEN、DIRSEL 位与 **BS16T_SGE** 寄存器的 SGUPD 位为控制位，这些位只能软件修改(SGUPD 位除外，仍由硬件自动清除)。一旦设置 CNTEN 位为 1，预分频器就由内部 INT_CLK 提供时钟。

INT_CLK 时钟来源为 APB 时钟(PCLK)，可以参考复位与时钟控制单元(RCU)。

下图给出了正常模式下没有分频控制电路和递增计数的情况。

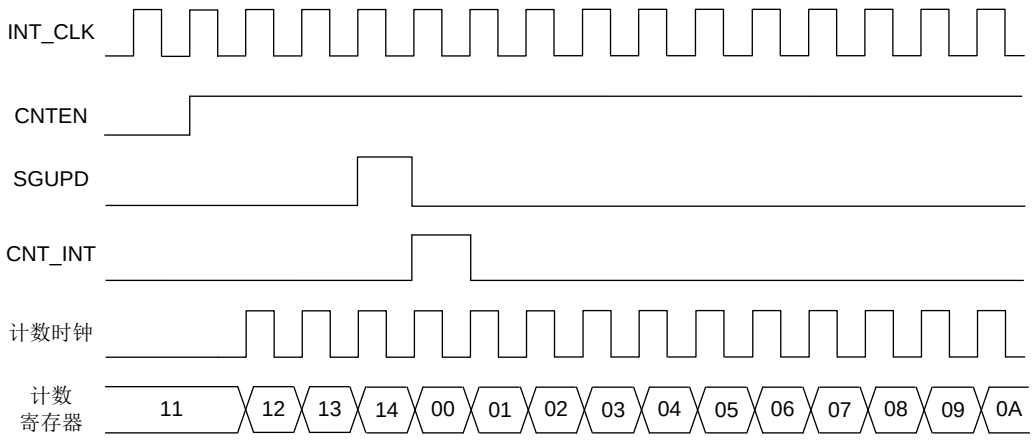


图 24-3 采用内部时钟计数

23. 4. 3 计数模式

23. 4. 3. 1 递增计数模式

设置 **BS16T_CON1** 寄存器的 DIRSEL 位为 0 时，定时器设置为递增模式，计数器从 0 开始递增，直至 **BS16T_AR** 寄存器数值；然后从 0 重新开始计数并产生一个更新事件(UPD)。

设置 **BS16T_SGE** 寄存器的 SGUPD 位为 1(通过软件)同样会产生更新事件，并让计数器和预分频器都重新从 0 开始计数。

通过软件设置 **BS16T_CON1** 寄存器中的 DISUE 位为 1 可关闭更新事件(UPD)产生。可以避免在写入预装载寄存器数值时产生更新事件(UPD)更新缓冲寄存器。在设置 DISUE 位为 0 之前都不会产生更新事件(UPD)。

此外，**BS16T_CON1** 寄存器中的 UERSEL 位为 1 时，设置 SGUPD 位为 1 会产生更新事件(UPD)，但不会更新更新标志位(**BS16T_RIF** 寄存器的 UPD 位)，也不会产生中断或 DMA 请求。

发生更新事件(UPD)时，所有预装载寄存器会更新到缓冲寄存器:

- 更新 **BS16T_AR** 寄存器数值到缓冲寄存器
- 更新 **BS16T_PRES** 寄存器数值到缓冲寄存器

下图为设置 **BS16T_AR** 寄存器为 16h，预分频设为 2 分频时的计数器时序。

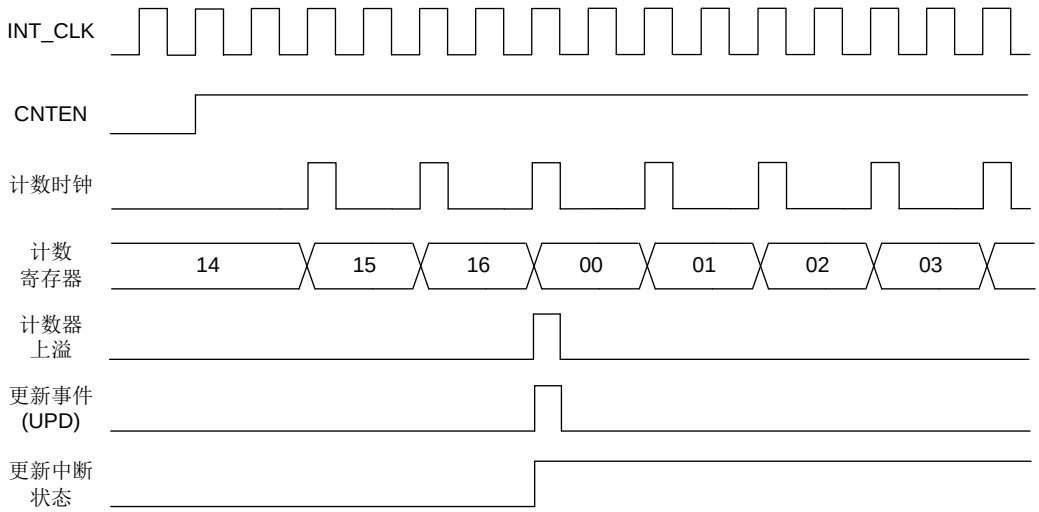


图 24-4　计数器递增计数时序图

下图为设置 **BS16T_AR** 寄存器从 F5h 改成 16h，分别在 ARPEN 为 0 或 1 时的计数器时序。

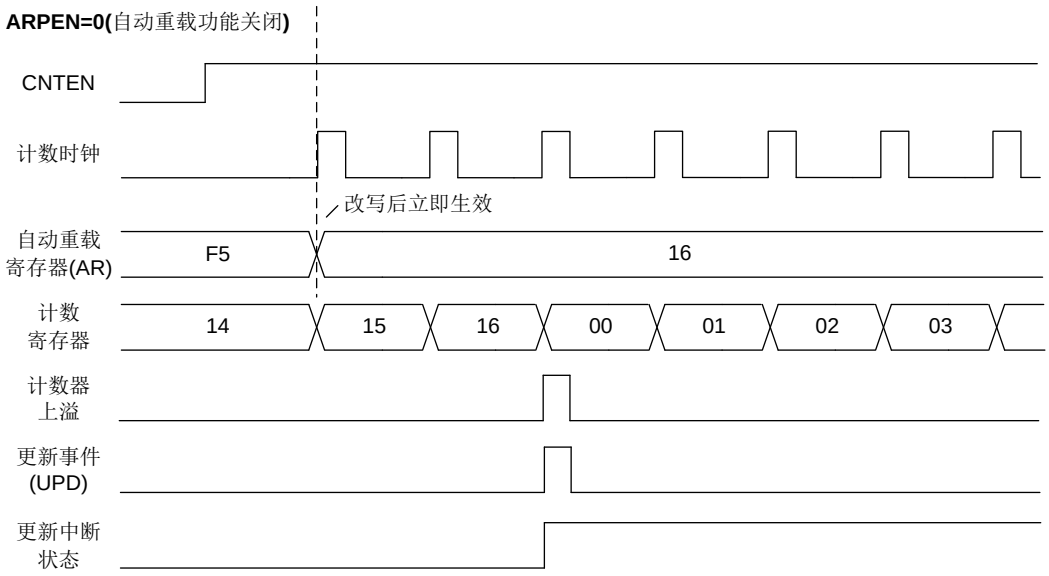


图 24-5　设置 ARPEN 位为 0 时计数器时序图

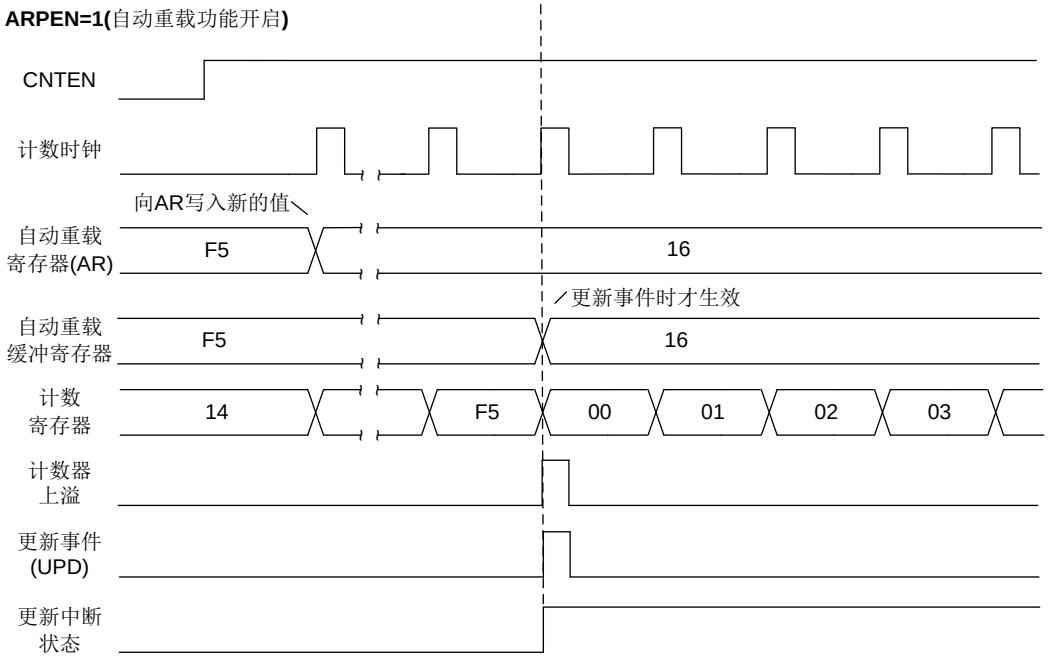


图 24-6　设置 ARPEN 位为 1 时计数器时序图

23. 4. 4　UPD位重映射

通过设置 **BS16T_CON1** 寄存器的 UPDREMAP 位，可以将 UPD 原始中断状态映射至 **BS16T_COUNT** 寄存器的第 31 位 UPDRIF 中。

注:UPD 和 UPDRIF 标志之间没有延迟。

23. 4. 5　调试模式

当微控制器进入调试模式(Cortex™-M0 核停止运行)，根据 SYSCFG 章节中 SYSCFG_CFG 寄存器配置，选择将计数器继续正常工作或停止计数。

23. 5 特殊功能寄存器

23. 5. 1 寄存器列表

BS16T 寄存器列表			
名称	偏移地址	类型	描述
BS16T_CON1	0000_H	R/W	控制寄存器 1
BS16T_CON2	0004_H	R/W	控制寄存器 2
BS16T_IER	$000C_H$	W1	中断开启寄存器
BS16T_IDR	0010_H	W1	中断关闭寄存器
BS16T_IVS	0014_H	R	中断功能有效状态寄存器
BS16T_RIF	0018_H	R	原始中断状态寄存器
BS16T_IFM	$001C_H$	R	中断标志位状态寄存器
BS16T_ICR	0020_H	C_W1	中断清除寄存器
BS16T_SGE	0024_H	T_W1	软件生成事件寄存器
BS16T_COUNT	0034_H	R/W	计数寄存器
BS16T_PRES	0038_H	R/W	预分频寄存器
BS16T_AR	$003C_H$	R/W	自动重载寄存器
BS16T_DMAEN	0058_H	R/W	DMA 事件开启寄存器

23. 5. 2 寄存器描述

23. 5. 2. 1 控制寄存器 1(BS16T_CON1)

控制寄存器 1(BS16T_CON1)																													
偏移地址:0x00																													
复位值:0x0000 0000																													
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																				UPDREMAP				ARPEN				SPMEN	UERSEL	DISUE	CNTEN

—	Bits 31-12	—	—
UPDREMAP	Bit 11	R/W	**UPD原始中断状态重新映射** UPD原始中断状态复制到BS16T_COUNT寄存器的31位 0:重新映射关闭 1:重新映射开启
—	Bits 10-8	—	—
ARPEN	Bit 7	R/W	**自动重载预装载开启** 发生更新事件时，将预装载值载入到缓冲寄存器中 0:BS16T_AR 寄存器不具备缓冲 1:BS16T_AR 寄存器具备缓冲
—	Bits 6-4	—	—
SPMEN	Bit 3	R/W	**单脉冲模式** 0:单脉冲模式关闭，计数器不停止 1:单脉冲模式开启，计数器在发生下一次更新事件时，会自动清除 CNTEN 位，并停止计数器
UERSEL	Bit 2	R/W	**更新事件来源选择** 设置更新事件(UPD)的来源 0:下列事件都会产生更新中断或 DMA 的请求: －计数器上溢 －设置 BS16T_SGE 寄存器的 SGUPD 位为 1 1:只有在计数器上溢时会生成更新中断或 DMA 的请求
DISUE	Bit 1	R/W	**更新事件禁止** 0:更新事件(UPD)开启时，下列事件皆会产生更新事件请求并将预装载值加载到缓冲寄存器中

			－计数器上溢 －设置 BS16T_SGE 寄存器的 SGUPD 位为 1 1:更新事件(UPD)关闭时，不产生更新事件请求，BS16T_AR、BS16T_PRES 寄存器的缓冲寄存器数值保持不变。但设置 BS16T_SGE 寄存器的 SGUPD 位为 1，计数器和预分频器仍会被重新初始化
CNTEN	Bit 0	R/W	**计数器开启** 0:计数器关闭 1:计数器开启

23. 5. 2. 2 控制寄存器 2 (BS16T_CON2)

控制寄存器 2 (BS16T_CON2)

偏移地址:0x04

复位值:0x0000 0000

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

—	Bits 31-7	—	—
MMSEL	Bit 6-4	R/W	**主模式选择** 选择在主模式下发送到从定时器的同步信号与 ADC 触发信号(TRGOUT) 0000:复位-设置 BS16T_SGE 寄存器中的 UPD 位为触发输出。 0001:开启-设置 BS16T_CON1 寄存器中的 CNTEN 位为触发输出。 0010:更新事件-更新事件被用于触发输出。
—	Bits 3-0	—	—

23. 5. 2. 3 中断开启寄存器(BS16T_IER)

中断开启寄存器(BS16T_IER)																												
偏移地址:0x0C																												
复位值:0x0000 0000																												
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																															UPD

—	Bits 31-1	—	—
UPD	Bit 0	W1	**开启更新中断功能** 此位设置时，开启中断功能，硬件侦测更新事件时发生中断

23. 5. 2. 4 中断关闭寄存器(BS16T_IDR)

中断关闭寄存器(BS16T_IDR)																												
偏移地址:0x10																												
复位值:0x0000 0000																												
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																															UPD

—	Bits 31-1	—	—
UPD	Bit 0	W1	**关闭更新中断功能** 此位设置时，关闭更新中断功能

23. 5. 2. 5 中断功能有效状态寄存器(BS16T_IVS)

中断功能有效状态寄存器(BS16T_IVS)																														
偏移地址:0x14																														
复位值:0x0000 0000																														
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																															UPD

—	Bits 31-1	—	—
UPD	Bit 0	R	**更新中断功能状态** 0:中断功能处于关闭状态 1:中断功能处于开启状态

BS16T_IVS 寄存器，是实时反映系统配置 BS16T_IER 与 BS16T_IDR 的中断开启状态。此寄存器状态是将 BS16T_IER 与 BS16T_IDR 进行硬件运算，公式如下:BS16T_IVS = BS16T_IER & ~BS16T_IDR

23. 5. 2. 6 原始中断状态寄存器(BS16T_RIF)

原始中断状态寄存器(BS16T_RIF)																														
偏移地址:0x18																														
复位值:0x0000 0000																														
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																															UPD

—	Bits 31-1	—	—
UPD	Bit 0	R	**更新，原始中断状态** 0:无发生中断 1:已发生中断

23. 5. 2. 7 中断标志位状态寄存器(BS16T_IFM)

中断标志位状态寄存器(BS16T_IFM)																												
偏移地址:0x1C																												
复位值:0x0000 0000																												
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																															UPD

—	Bits 31-1	—	—
UPD	Bit 0	R	**更新，中断标志位状态** 0:无发生中断 1:已发生中断

BS16T_IFM 寄存器，是滤除已关闭中断功能的中断事件，只关注开启中断功能的事件，此寄存器状态是将 BS16T_RIF 与 BS16T_IVS 进行硬件运算，公式如下:BS16T_IFM = BS16T_RIF &BS16T_IVS

23. 5. 2. 8 中断清除寄存器(BS16T_ICR)

中断清除寄存器(BS16T_ICR)																												
偏移地址:0x20																												
复位值:0x0000 0000																												
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																															UPD

—	Bits 31-1	—	—
UPD	Bit 0	C_W1	**清除更新中断状态** 此位设置时，清除中断状态(BS16T_RIF 与 BS16T_IFM)

BS16T_ICR 寄存器设置时，将清除 BS16T_RIF 与 BS16T_IFM 中断标志位状态；此设置不影响中断 BS16T_IER、BS16T_IDR 与 BS16T_IVS 寄存器，只清除标志位状态 BS16T_RIF 与 BS16T_IFM。此寄存器通过硬件清除中断，公式如下:BS16T_RIF = BS16T_RIF & ~BS16T_ICR

23. 5. 2. 9 软件生成事件寄存器(BS16T_SGE)

软件生成事件寄存器(BS16T_SGE)																													
偏移地址:0x24																													
复位值:0x0000 0000																													
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																															SGUPD

—	Bits 31-1	—	—
SGUPD	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。 此位设置时，产生更新事件。重新初始化计数器，更新寄存器。 注:预分频器也会被清零(但预分频比不会受到影响)。

23. 5. 2. 10 计数寄存器(BS16T_COUNT)

计数寄存器(BS16T_COUNT)																													
偏移地址:0x34																													
复位值:0x0000 0000																													
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
UPDRIF																CNTV<15:0>														

UPDRIF	Bit 31	R	**更新原始中断状态映射** 该位为映像BS16T_RIF寄存器的UPD位的状态，该位只能读取，表示计数器发生更新事件。 设置BS16T_CON1寄存器的UPDFREMAP位为1开启功能。
—	Bits 30-16	—	—
CNTV	Bits 15-0	R/W	**计数器数值**

23. 5. 2. 11 预分频寄存器(BS16T_PRES)

预分频寄存器(BS16T_PRES)																															
偏移地址:0x38																															
复位值:0x0000 0000																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																PSCV<15:0>															

—	Bits 31-16	—	—
PSCV	Bit 15-0	R/W	**预分频数值** 当计数器时钟频率等于f_{INT_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

23. 5. 2. 12 自动重载寄存器(BS16T_AR)

自动重载寄存器(BS16T_AR)																															
偏移地址:0x3C																															
复位值:0x0000 FFFF																															
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
																ARV<15:0>															

—	Bits 31-16	—	—
ARV	Bits 15-0	R/W	**自动装载数值** 设置计数器的递增计数的边界

23. 5. 2. 13 DMA事件开启寄存器 (BS16T_DMAEN)

DMA 事件开启寄存器 (BS16T_DMAEN)																												
偏移地址:0x58																												
复位值:0x0000 0000																												
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
																															UPD

—	Bits 31-1	—	—
UPD	Bits 0	R/W	**更新事件的 DMA 请求开启** 0: 更新事件的 DMA 请求关闭 1: 更新事件的 DMA 请求开启

第24章 独立看门狗(IWDT)

24. 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 32.768 kHz 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。

24. 2 特性

- 自由运行的递减计数器
- 由一个独立的 RC 振荡器驱动(在低功耗模式下仍可操作)，复位条件如下:
  - 当向下递减计数器的值达到 0 时产生复位请求。
  - 当向下递减计数器的值处于窗口值之外时，被重新加载就会导致复位请求。

24. 3 结构图

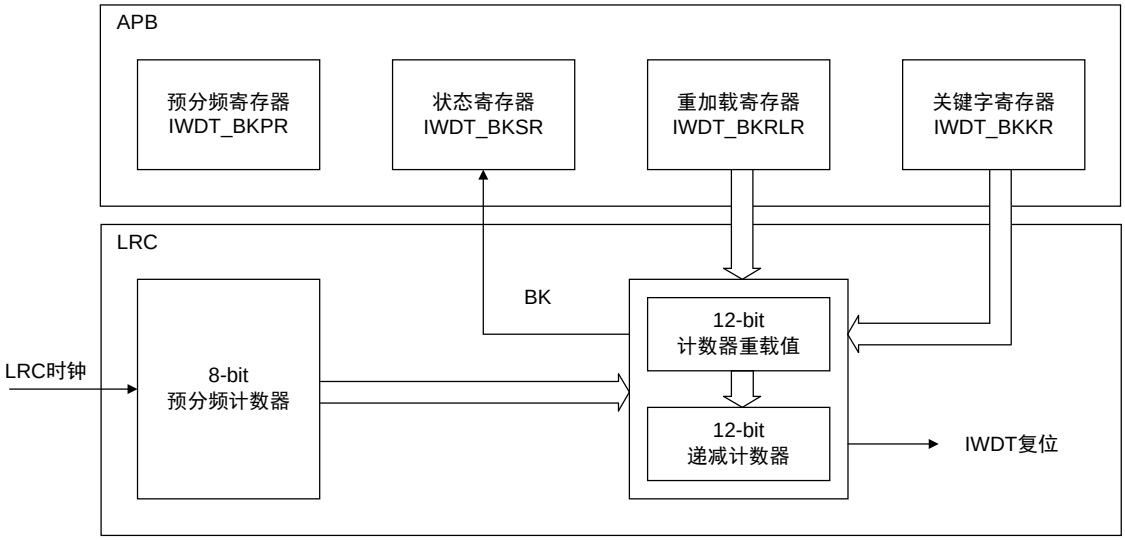


图 25-1　IWDT 架构图

当向关键词寄存器写入初始化启动指令 0x0000CCCC 的时候，将自动开启 LRC 时钟，看门狗计数器开始由复位值 0xFFF 向下计数。

当计数值达到 0x000 的时候由独立看门狗发出复位信号。

启动后将关键词 0x0000AAAA 写到 **IWDT_BKKR** 寄存器中，可以使 **IWDT_BKRLR** 寄存器中的值被重载到看门狗计数器中，从而阻止即将发生的复位动作。+ 注:欲先配置 IWDT 再启动 IWDT，需要先自行开启 LRC 时钟源。

24. 4 功能描述

24. 4. 1 窗口选项

IWDT 也能够工作在窗口看门狗模式下，只要在 **IWDT_BKWINR** 寄存器中设置适当的值即可。

如果重加载操作执行的同时，看门狗计数器的值超出了窗口寄存器(**IWDT_BKWINR**)中存储的值，也会引起复位操作。

IWDT_BKWINR 的默认值是 0x00000FFF,所以如果没有改写它，那么窗口选项默认是关闭的。

当窗口选项使能时配置 IWDT

- 将 0x0000CCCC 写到 **IWDT_BKKR** 寄存器，使能 IWDT。
- 等待状态寄存器 **IWDT_BKFR**.BUSY 的值更新为 0。
- 向 **IWDT_BKKR** 寄存器写 0x00005555 打开寄存器访问许可。
- 等候状态寄存器 **IWDT_BKFR**.BUSY 的值更新为 0。
- 向 **IWDT_BKPR** 写 0~7 的值，以配置 IWDT 的预分频器。
- 等候状态寄存器 **IWDT_BKFR**.BUSY 的值更新为 0。
- 配置重加载寄存器(**IWDT_BKRLR**)。
- 等待状态寄存器 **IWDT_BKFR**.BUSY 的值更新为 0。
- 配置窗口寄存器 **IWDT_BKWINR**。此时会将 **IWDT_BKRLR** 的值刷新到看门狗定时器。

当窗口选项被禁止时配置 IWDT

- 将 0x0000CCCC 写到 **IWDT_BKKR** 寄存器，使能 IWDT。
- 等候状态寄存器 **IWDT_BKFR**.BUSY 的值更新为 0。
- 向 **IWDT_BKKR** 寄存器写 0x00005555 打开寄存器访问许可。
- 等候状态寄存器 **IWDT_BKFR**.BUSY 的值更新为 0。
- 向 **IWDT_BKPR** 写 0~7 的值，以配置 IWDT 的预分频器。
- 等候状态寄存器 **IWDT_BKFR**.BUSY 的值更新为 0。
- 配置重加载寄存器(**IWDT_BKRLR**)。
- 等候状态寄存器 **IWDT_BKFR**.BUSY 的值更新为 0。
- 将 **IWDT_BKRLR** 的值刷新到看门狗定时器(**IWDT_BKKR**=0x0000AAAA)。

24. 4. 2 低功耗模式下的行为

使能 IWDT 后，则无法停止 IWDT。

24. 4. 2. 1 寄存器访问保护

默认条件下，对 **IWDT_BKPR**、**IWDT_BKRLR** 和 **IWDT_BKWINR** 的写访问操作都是受保护的。想要改变这一点，必须先向 **IWDT_BKKR** 写入 0x00005555 解锁码。如果向 **IWDT_BKKR** 写入别的值，寄存器的访问保护重新生效。这意味着在做重载入操作的时候(向该寄存器写入 0x0000AAAA)就属于这种情况。

预分频器的更新、看门狗计数器的重加载或窗口值的更新的状态，可以通过旗帜寄存器得知。

若想读取 **IWDT_BKPR**.PR、**IWDT_BKRLR**.RL、**IWDT_BKWINR**.WIN 和 **IWDT_BKSR**.CNT，必须先向对应的寄存器设定读取。

欲读取 IWDT_BKPR 时配置:

- 向 IWDT_BKPR.R_PR 配置 1。
- 等待状态寄存器 IWDT_BKFR.BUSY 的值更新为 0。
- 读取 IWDT_BKPR.PR。

欲读取 IWDT_BKRLR 时配置:

- 向 IWDT_BKRLR.R_RL 配置 1。
- 等待状态寄存器 IWDT_BKFR.BUSY 的值更新为 0。
- 读取 IWDT_BKRLR.RL。

欲读取 IWDT_BKWINR 时配置:

- 向 IWDT_BKWINR.R_WIN 配置 1。
- 等待状态寄存器 IWDT_BKFR.BUSY 的值更新为 0。
- 读取 IWDT_BKWINR.WIN。

欲读取 IWDT_BKSR 时配置:

- 向 IWDT_BKSR.R_CNT 配置 1。
- 等待状态寄存器 IWDT_BKFR.BUSY 的值更新为 0。
- 读取 IWDT_BKSR.CNT。

24. 4. 3 调试模式

当微控制器进入调试模式时(内核被暂停)，看门狗计数器可以继续运行，也可以被停止。

24. 5 特殊功能寄存器

24. 5. 1 寄存器列表

IWDT 寄存器列表			
名称	偏移地址	类型	描述
IWDT_BKKR	0080_H	W	IWDT 关键字寄存器
IWDT_BKPR	0084_H	R/W	IWDT 预分频寄存器
IWDT_BKRLR	0088_H	R/W	IWDT 重载入寄存器
IWDT_BKFR	$008C_H$	R	IWDT 标志位寄存器
IWDT_BKWINR	0090_H	R/W	IWDT 窗口寄存器
IWDT_BKSR	0094_H	R	IWDT 状态寄存器

24. 5. 2 寄存器描述

24. 5. 2. 1 IWDT关键字寄存器 (IWDT_BKKR)

此寄存器只允许使用 32 位存取。

IWDT 关键字寄存器 (IWDT_BKKR)																													
偏移地址:0x80																													
复位值:0x0000 0000																													
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<15:0>															

—	Bit 31-16	—	—
KEY	Bit 15-0	W	**关键值(只写，读的话会是 0x0000)** 这些位必须周期性的由软件写入 0xAAAA，否则当计数器向下计数到 0 的时候会产生硬件复位请求。 写入 0x5555 会使能对 IWDT_BKPR, IWDT_BKRLR 和 IWDT_BKWINR 三个寄存器的访问许可。写入 0xCCCC 启动看门狗 注:关键值只能在 IWDT_BKFR 寄存器中的 BUSY 位为零时更新。

24. 5. 2. 2 IWDT预分频寄存器 (IWDT_BKPR)

此寄存器只允许使用 32 位存取。

IWDT 预分频寄存器 (IWDT_BKPR)																													
偏移地址:0x84																													
复位值:0x0000 0001																													
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
																R_PR													PR<2:0>		

—	Bit 31-16	—	—
R_PR	Bit 15	S_W1	**读取预分频器** 由软件写入，用来设定读取输入时钟的预分频系数。为了能够读取到正确的预分频器的分频系数，设定后必须等待 IWDT_BKFR 寄存器中的 BUSY 位清零，再读取预分频器的分频系数。(由硬件清零)
—	Bit 14-3	—	—
PR	Bit 2-0	R/W	**预分频器** 这些位平时处于写保护状态，由软件写入，用来选择对输入时钟的预分频系数。为了能够改变预分频器的分频系数，**IWDT_BKFR** 寄存器中的 BUSY 位必须先清零。 000: 4 分频 001: 8 分频 010: 16 分频 011: 32 分频 100: 64 分频 101: 128 分频 110: 256 分频 111: 256 分频

24. 5. 2. 3 IWDT重载入寄存器 (IWDT_BKRLR)

此寄存器只允许使用 32 位存取。

IWDT 重载入寄存器 (IWDT_BKRLR)																													
偏移地址:0x88																													
复位值:0x0000 0FFF																													
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
																R_RL				RL<11:0>											

—	Bit 31-16	—	—
R_RL	Bit 15	S_W1	**读取看门狗计数器重载值** 由软件写入，用来设定读取看门狗计数器。为了能够读取到正确的看门狗计数器，设定后必须等待 IWDT_BKFR 寄存器中的 BUSY 位清零，再读取看门狗计数器。(由硬件清零)
—	Bit 14-12	—	—
RL	Bit 11-0	R/W	**看门狗计数器重载值** 这些位平时处于写保护状态，这个值是由软件来设置的，并且每次向 IWDT_BKKR 寄存器写入 0xAAAA 的时候，这个值会被更新到看门狗计数器中，如果想延时长一点，这个值就该大一些。因为看门狗计数器正是从这个值开始向下计数。定时的长度是由这个值和预分频器的设置值来共同决定的。为了能够改变看门狗计数器重载值，IWDT_BKFR 寄存器中的 BUSY 位必须先清零。

24. 5. 2. 4 IWDT标志位寄存器 (IWDT_BKFR)

此寄存器只允许使用 32 位存取。

IWDT 标志位寄存器 (IWDT_BKFR)																														
偏移地址:0x8C																														
复位值:0x0000 0000																														
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
|	|	|	|	|	|	|	|	|	|	|	|	|	|	|	BUSY	|	|	|	|	|	|	|	|	|	|	|	|	|	|	|

—	Bit 31-16	—	—
BUSY	Bit 15	R	**寄存器更新** 该位由硬件设置，以表示关键值、预分频器、重载或窗口值正在进行更新。当完成更新操作后(需要多达 5 个 LRC 周期)，会通过硬件将该位清零。 必须等到 BUSY 位被清零后，才能更新关键值、预分频器、重载值或窗口值。
—	Bit 14-0	—	—

24. 5. 2. 5 IWDT窗口寄存器 (IWDT_BKWINR)

此寄存器只允许使用 32 位存取。

IWDT 窗口寄存器 (IWDT_BKWINR)																												
偏移地址:0x90																												
复位值:0x0000 0FFF																												
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
																R_WIN				WIN<11:0>											

—	Bit 31-16	—	—
R_WIN	Bit 15	S_W1	**读取看门狗计数器窗口值** 由软件写入，用来设定读取看门狗计数器窗口值。为了能够读取到正确的看门狗计数器窗口值，设定后必须等待 IWDT_BKFR 寄存器中的 BUSY 位清零，再读取看门狗计数器窗口值。(由硬件清零)
—	Bit 14-12	—	—
WIN	Bit 11-0	R/W	**看门狗计数器窗口值** 这些位平时处于写保护状态，这些位包含的是窗口值和向下计数器的比较上限。为了阻止复位信号的产生，必须在向下计数器递减到窗口值和 0x0 之间的某个值时重加载它。要想改变这个窗口值，必须先保证 IWDT_BKFR 中的 BUSY 位为 0。

24. 5. 2. 6 IWDT状态寄存器 (IWDT_BKSR)

此寄存器只允许使用 32 位存取。

IWDT 状态寄存器 (IWDT_BKSR)																												
偏移地址:0x94																												
复位值:0x0000 0FFF																												
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
																R_CNT				CNT<11:0>											

—	Bit 31-16	—	—
R_CNT	Bit 15	S_W1	**读取看门狗计数器** 由软件写入，用来设定读取看门狗计数器。为了能够读取到正确的看门狗计数器，设定后必须等待 IWDT_BKFR 寄存器中的 BUSY 位清零，再读取看门狗计数器。(由硬件清零)
—	Bit 14-12	—	—
CNT	Bit 11-0	R	**12-bit 计数器(MSB to LSB)** 这些位包含看门狗计数器的值。

第25章 窗口看门狗 (WWDT)

25. 1 概述

窗口看门狗通常被用来监测，由外部干扰或不可预见的逻辑条件造成的应用程序背离正常的运行序列而产生的软件故障。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喂狗行为，防止程序运行至不可控状态。看门狗电路在达到预置的时间周期时，如果 7 位的递减计数器数值(在控制寄存器中) 未被刷新，会产生一个 MCU 复位。在递减计数器达到窗口寄存器数值之前，如果 7 位的递减计数器数值被刷新，那么也将产生一个 MCU 复位。这表明递减计数器需要在一个有限的时间窗口中被刷新。

WWDT 时钟从 APB1 时钟预分频，适合要求看门狗在一个精确时间窗口内做出反应的应用程序。

25. 2 特性

- 可配置的递减计数器
- 看门狗被启动后，复位产生的条件
  - 当递减计数器的值从 0x3F 翻转时，则产生复位。
  - 当递减计数器在窗口外被重新装载，则产生复位。
- 提前唤醒中断:如果启用中断，当递减计数器等于 0x40 时产生提前唤醒中断。

25. 3 结构图

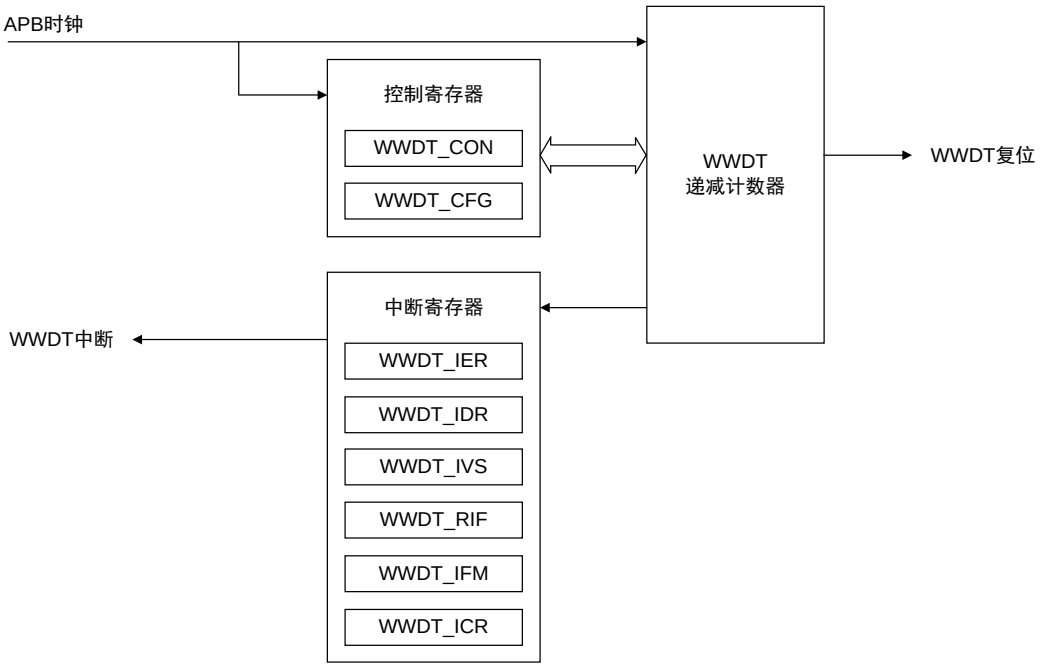


图 26-1　WWDT 架构图

25. 4 功能描述

上电复位后，窗口看门狗不启动，需通过软件配置使能窗口看门狗(**WWDT_CON** 寄存器中的 WDGA 位被置’1’)。当 7 位(T[6:0])递减计数器从 0x3F 翻转时，则产生一个复位。如果软件在计数器值大于窗口寄存器中的数值时重新装载计数器，将产生一个复位。

应用程序在正常运行过程中必须定期地写入 **WWDT_CON** 寄存器以防止 MCU 发生复位。只有当计数器值小于窗口寄存器的值时，才能进行写操作。储存在 **WWDT_CON** 寄存器中的数值必须在 0xFF 和 0xC0 之间。

25. 4. 1 启用看门狗

设置 **WWDT_CON** 寄存器的 WDGA 位能够开启看门狗，随后它不能再被关闭，除非发生复位。

25. 4. 2 控制递减计数器

递减计数器处于自由运行状态:即使看门狗被禁止，递减计数器仍继续递减计数。

T[5:0]位包含了看门狗产生复位之前的计时数目，配置寄存器(**WWDT_CFG**)中包含窗口的上限值:要避免产生复位，递减计数器必须在其值小于窗口寄存器的数值并且大于 0x3F 时被重新装载。

25. 4. 3 高级看门狗中断功能

如果在实际复位产生之前必须进行特定的安全操作或数据记录，可以用提前唤醒中断。设置 **WWDT_IER** 寄存器中的 EWI 位开启该中断。在复位之前当递减计数器到达 0x40 时，则产生此中断，同时可以用相应的中断服务程序(ISR)来触发特定的行为(例如通信或数据记录)，在某些应用中，提前唤醒中断可以用来管理软件系统检测和/或系统恢复，但不产生 WWDT 复位。在这种情况下，相应的中断服务程序(ISR)将重加载 WWDT 计数器，以避免 WWDT 复位。

注:当提前唤醒中断无法启用时(例如系统锁定在更高优先级任务)，最终将产生 WWDT 复位。

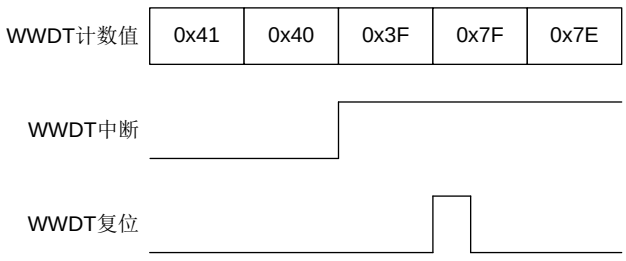


图 26-2 WWDT 中断示意图

25. 4. 4 如何配置看门狗超时

计算超时的公式如下:$t_{WWDT}=t_{PCLK}$x 4096 x 2^{WDGTB} x (t[5:0]+1) (ms)

其中:

t_{WWDT}:WWDT 超时时间

t_{PCLK}:APB1 以 ms 为单位的时钟周期

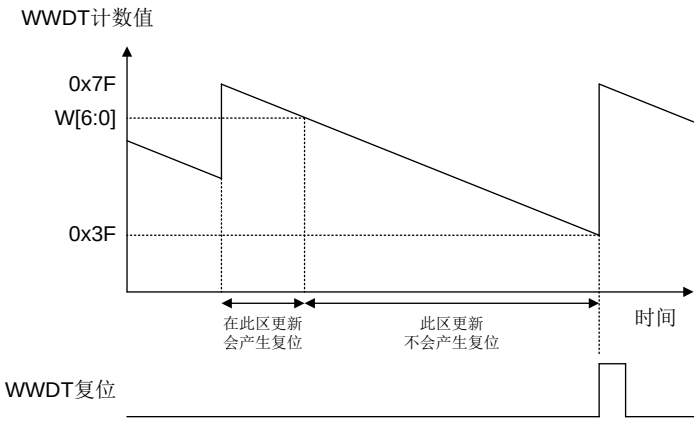


图 26-3 WWDT 时序图

25. 4. 5 调试模式

当微控制器进入调试模式时(内核被暂停)，看门狗计数器可以继续运行，也可以被停止。

25. 5 特殊功能寄存器

25. 5. 1 寄存器列表

WWDT 寄存器列表			
名称	偏移地址	类型	描述
WWDT_CON	000_H	R/W	WWDT 控制寄存器
WWDT_CFG	004_H	R/W	WWDT 配置寄存器
WWDT_IER	008_H	W1	WWDT 中断开启寄存器
WWDT_IDR	$00C_H$	W1	WWDT 中断关闭寄存器
WWDT_IVS	010_H	R	WWDT 中断功能有效状态寄存器
WWDT_RIF	014_H	R	WWDT 原始中断状态寄存器
WWDT_IFM	018_H	R	WWDT 中断标志位状态寄存器
WWDT_ICR	$01C_H$	C_W1	WWDT 中断清除寄存器

25. 5. 2 寄存器描述

25. 5. 2. 1 WWDT控制寄存器 (WWDT_CON)

WWDT 控制寄存器 (WWDT_CON)																												
偏移地址:0x00																												
复位值:0x0000 007F																												
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
																								WDGA	T<6:0>						

—	Bit 31-8	—	—
WDGA	Bit 7	R/S_W1	**启动位** 此位由软件置'1'，但仅能由硬件在复位后清'0'。当 WDGA=1 时，看门狗可以产生复位 0:写入 0 无效 1:启用看门狗
T	Bit 6-0	R/W	**7-bit 计数器 (MSB to LSB)** 这些位用来存储看门狗的计数器值。每 $(4096x2^{WDGTB[1:0]})$ 个 PCLK 周期减 1。当计数器值从 0x3F 翻转时，产生看门狗复位。

25. 5. 2. 2 WWDT配置寄存器 (WWDT_CFG)

WWDT 配置寄存器 (WWDT_CFG)																												
偏移地址:0x04																												
复位值:0x0000 007F																												
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
																							WDGTB<1:0>		W<6:0>						

—	Bit 31-9	—	—
WDGTB	Bit 8-7	R/W	**时基** 预分频器的时基可以设置如下: 00:计时器时钟 (PCLK 除以 4096) 分频器 1 01:计时器时钟 (PCLK 除以 4096) 分频器 2 10:计时器时钟 (PCLK 除以 4096) 分频器 4 11:计时器时钟 (PCLK 除以 4096) 分频器 8
W	Bit 6-0	R/W	**7-bit 窗口值** 这些位包含了用来与递减计数器进行比较用的窗口值

25. 5. 2. 3 WWDT中断开启寄存器 (WWDT_IER)

WWDT 中断开启寄存器 (WWDT_IER)																												
偏移地址:0x08																												
复位值:0x0000 0000																												
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																															EWI

—	Bit 31-1	—	—
EWI	Bit 0	W1	**开启提前唤醒中断** 0:写入 0 无效 1:开启中断

25. 5. 2. 4 WWDT中断关闭寄存器 (WWDT_IDR)

WWDT 中断关闭寄存器 (WWDT_IDR)																												
偏移地址:0x0C																												
复位值:0x0000 0000																												
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																															EWI

—	Bit 31-1	—	—
EWI	Bit 0	W1	**关闭提前唤醒中断** 0:写入 0 无效 1:关闭中断

25. 5. 2. 5 WWDT中断功能有效状态寄存器 (WWDT_IVS)

WWDT 中断功能有效状态寄存器 (WWDT_IVS)																												
偏移地址:0x10																												
复位值:0x0000 0000																												
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																															EWI

—	Bit 31-1	—	—
EWI	Bit 0	R	**提前唤醒中断功能有效状态** 0:禁止中断 1:启用中断

25. 5. 2. 6 WWDT原始中断状态寄存器 (WWDT_RIF)

WWDT 原始中断状态寄存器 (WWDT_RIF)																														
偏移地址:0x14																														
复位值:0x0000 0000																														
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																															EWI

—	Bit 31-1	—	—
EWI	Bit 0	R	**提前唤醒原始中断状态** 当计数器的值由 0x40 递减到 0x3F 并启用中断时，该位由硬件设置。

25. 5. 2. 7 WWDT中断标志位状态寄存器 (WWDT_IFM)

WWDT 中断标志位状态寄存器 (WWDT_IFM)																														
偏移地址:0x18																														
复位值:0x0000 0000																														
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																															EWI

—	Bit 31-1	—	—
EWI	Bit 0	R	**提前唤醒中断标志位状态** 当计数器的值由 0x40 递减到 0x3F 并启用中断时，该位由硬件设置。

25. 5. 2. 8 WWDT 中断清除寄存器 (WWDT_ICR)

WWDT 中断清除寄存器 (WWDT_ICR)																														
偏移地址:0x1C																														
复位值:0x0000 0000																														
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																															EWI

—	Bit 31-1	—	—
EWI	Bit 0	C_W1	**提前唤醒中断清除** 这个位是用软件来设定的。写入“1”以清除中断。

第26章 串行通讯 (I2C)

26. 1 概述

I2C 是两线双向的串行传输总线，提供了一种简单有效的方法来实现设备之间的数据交换。并且支持多主机模式功能，适合用于多主机应用的场景。

在通信速率上有标准模式(Sm)、快速模式(Fm)和极快速模式(Fm+)让用户选择。I2C 总线同时兼容 SMBus(系统管理总线)与 PMBus(电源管理总线)。

26. 2 特性

- 符合 I2C 规范第 3 版
  - 支持主机或从机模式操作
  - 多主机模式功能
  - 标准模式(高达 100 kHz)
  - 快速模式(高达 400 kHz)
  - 极快速模式(高达 1 MHz)
  - 7 位或 10 位从机地址(主机模式)
  - 提供 2 组本机地址(地址 1 支持 7 位或 10 位，地址 2 支持屏蔽功能)
  - 支持所有 7 位地址响应模式
  - 支持广播模式
  - 支持软件配置传输数据的建立时间和保持时间
  - 支持时钟延长
  - 支持软件复位
- 符合 SMBus 规范第 3.0 版
  - 硬件 PEC (数据包错误校验)产生与 ACK 控制
  - 命令与数据 ACK 控制
  - 支持地址解析通讯协议
  - 支持主机或从机设备操作
  - 支持 SMBus 警报
  - 提供超时与空闲检测功能
- 符合 PMBus 规范第 1.3 版
- 可配置数字噪声滤波器
- 支持 DMA 传输
- 独立时钟: 独立时钟源选择不为 PCLK 时，当 PCLK 时钟频率更改时，不会影响 I2C 通信速率

26. 3 结构图

关于 I2C 界面的结构如下图所示:

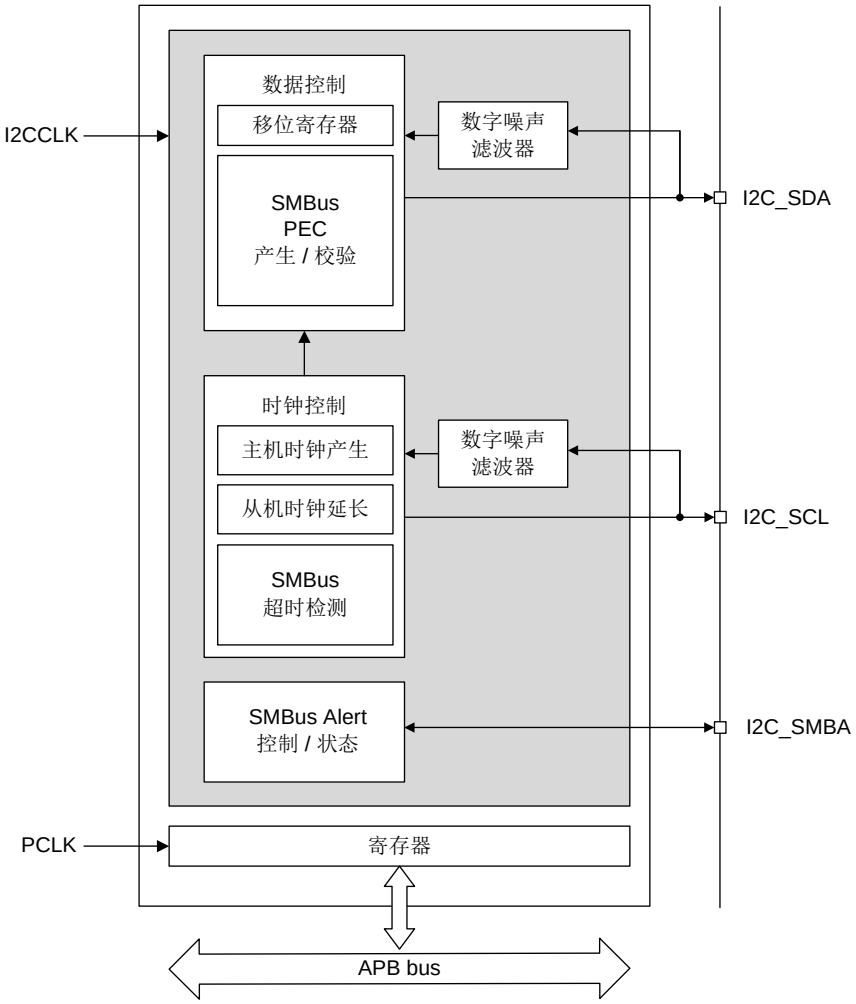


图 27-1　I2C 结构图

26. 4 I2C时钟要求

I2CCLK 提供 I2C 内核的时钟。

I2CCLK 周期 T_{I2CCLK} 必须符合以下条件:

$T_{I2CCLK} < (T_{LOW} - T_{Filter}) / 4$ 且 $T_{I2CCLK} < T_{HIGH}$

注：

1. T_{LOW} : SCL 低电平时间。
2. T_{HIGH} : SCL 高电平时间。
3. T_{Filter} : 数字滤波器开启时带来的延迟总和。数字滤波器延迟为 DNF x T_{I2CCLK}。

PCLK 时钟周期 T_{PCLK} 必须符合以下条件:

$T_{PCLK} < 4/3\ T_{SCL}$

注：

1. T_{SCL}:SCL 周期。
2. 如果 I2C 内核时钟设置为 PCLK 时，该时钟必须符合 T_{I2CCLK} 的条件。

26. 5 功能描述

该接口通过数据引脚 (SDA) 和时钟引脚 (SCL) 连接到 I2C 总线中，支持标准模式(最高 100 kHz)、快速模式(最高 400 kHz) 或极快速模式(最高 1 MHz)的通信速率。

如果支持 SMBus 功能，该接口也可通过数据引脚 (SDA) 和时钟引脚 (SCL) 连接到 SMBus。并可以使用 SMBus 警报引脚(SMBA)。

26. 5. 1 I2C 通信协议

I2C 接口可配置为以下四种模式：

- 从机发送模式
- 从机接收模式
- 主机发送模式
- 主机接收模式

该接口默认配置为从机模式。当用户设置 **I2C_CON2**.START 产生起始位(START)后，可由从机模式转换为主机模式。当发生仲裁丢失或产生停止信号时，接口由主机模式转换为从机模式，藉此实现多主机模式。

26. 5. 1. 1 通信流程

不论是发送或接收，时钟信号都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，直到出现停止位后结束。

在主机模式时，用户可设置起始位(START)或停止位(STOP)的产生来控制数据传输。

在从机模式时，可配置本机的地址(7 位或 10 位)以及广播呼叫地址，当通信时检测到与本机地址或广播呼叫地址匹配后响应 ACK。通过软件开启或关闭广播呼叫和 SMBus 保留地址的检测功能。

数据和地址都是以 8 位且使用最高有效位(MSB)在前进行传输。标准通信格式为主机发送完起始位后，紧接着发送的是地址，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)。

26. 5. 1. 2 START和STOP条件协议

I2C 规范当时钟线(SCL)为高电平时，数据线(SDA)上的电平转换被定义为起始位(START)或停止位(STOP)。

- START 条件定义为在 SCL 为高电平时，SDA 从高电平转换到低电平。表示总线从空闲状态转换为繁忙状态。
- STOP 条件定义为在 SCL 为高电平时，SDA 从低电平转换到高电平。表示总线从繁忙状态转换为空闲状态。

当总线空闲时，SCL 和 SDA 都通过总线上的外部上拉电阻拉到高电平。当在 SCL 处于高电

平期间时会对每个数据位进行采样，因此在地址或数据传输时， SDA 只能在 SCL 为低电平期间改变，并且必须在 SCL 为高电平期间保持稳定。

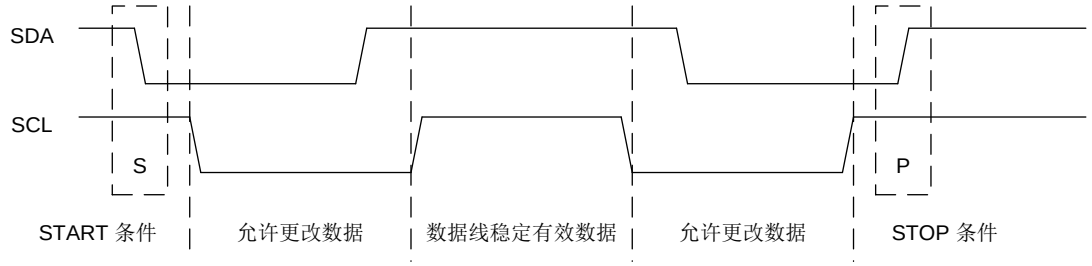


图 27-2 START 和 STOP 条件

26. 5. 1. 3 应答位

数据或地址传输时都带有应答位，并由主机产生应答位的时钟脉冲。

- 主机发送模式或从机发送模式下，SDA 必须在应答时钟脉冲低电平期间释放为高电平，等待接收应答位。
- 主机接收模式或从机接收模式下，在应答时钟脉冲低电平期间 ：
  - SDA 下拉为低电平表示响应 ACK
  - SDA 释放为高电平表示响应 NACK。

当从机正在执行某些实时功能导致无法接收或发送时，从机必须将 SDA 保持为高电平响应 NACK。主机可以产生 STOP 条件中止传输，或者产生 START 条件启动新的传输。

在主机接收模式时，必须在从机输出的最后一个字节后，由主机响应 NACK 来向从机发送数据结束信号。从机在应答位收到 NACK 后必须释放 SDA，以允许主机产生 STOP 或重复 START 条件。

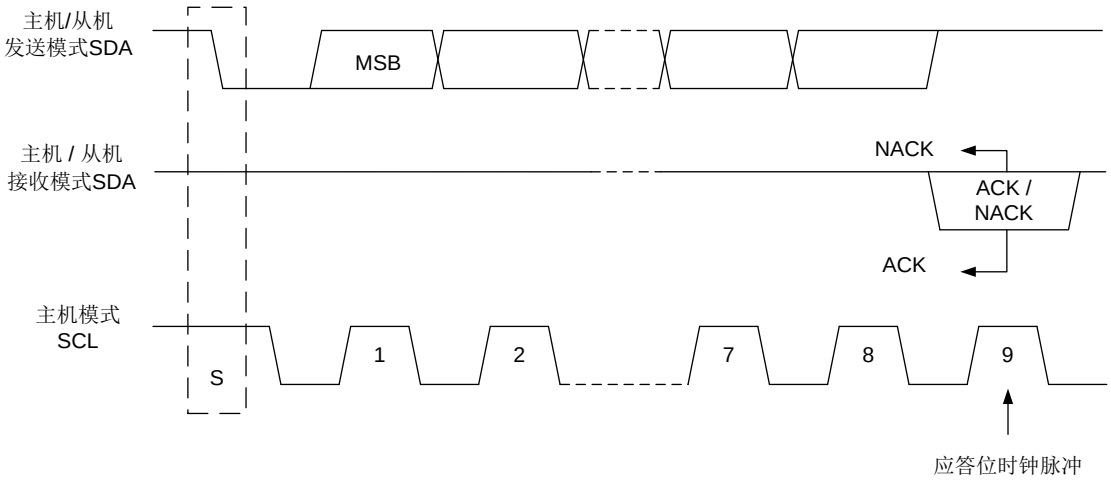
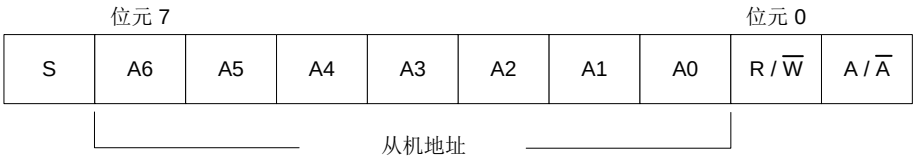


图 27-3 I2C 总线上的应答

26. 5. 1. 4 I2C从机地址协议

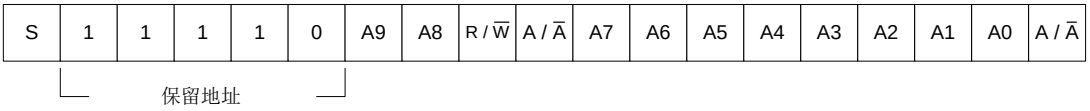
从机地址有两种地址格式：7 位地址格式和 10 位地址格式。

- 7 位从机地址格式。主机传输第一个字节的前 7 位(位 7:1) 为从机地址，最后 1 位(位 0)是 R/W 位。当位 0 设置为 0 时，主机向从机写数据。当位 0 设置为 1 时，主机从从机读数据。
- 10 位从机地址格式。主机传输第一个字节的前 5 位(位 7:3) 固定为 5’b11110，通知从机这是一个 10 位从机地址格式，接着后 2 位(位 2:1)，它是从机地址中位 9:8，最后 1 位(位 0)是 R/W 位。第二个字节传输为从机地址中位 7:0。



S = 起始位
R / W = 读 / 写
A / A = ACK / NACK (由从机发送)

图 27-4 7 位从机地址格式



S = 起始位
R / W = 读 / 写
A / A = ACK / NACK (由从机发送)

图 27-5 10 位从机地址格式

26. 5. 1. 5 I2C发送和接收协议

所有数据都以字节(8 位)格式传输，每次传输最多为 65535 个字节数。在主机发送地址和 R/W 位或主机向从机每发送一个字节数据后，接收的从机必须在应答位响应 ACK。当接收的从机在应答位响应 NACK 时，主机通过发出 STOP 条件来中止传输。从机必须将 SDA 释放并保持高电平，以便主机可以中止传输。传输格式如下:

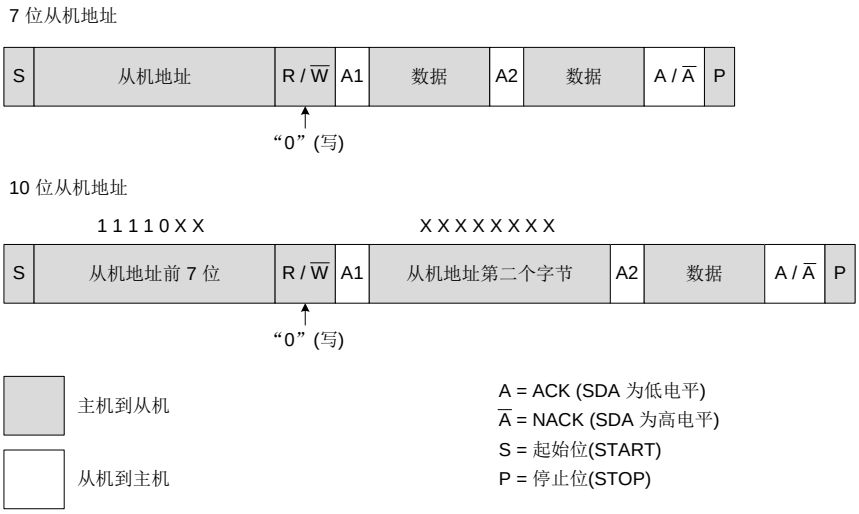


图 27-6 主机-发送协议

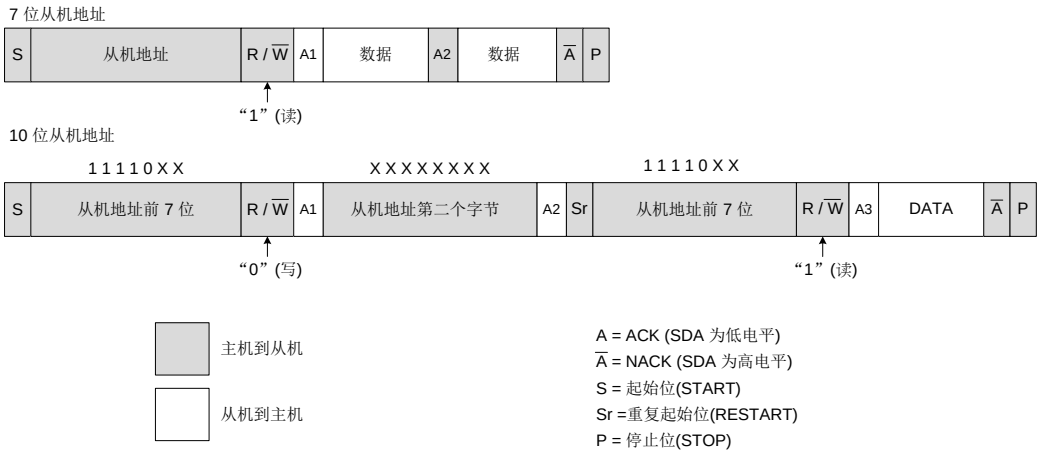


图 27-7 主机-接收协议

26. 5. 2 I2C初始化

26. 5. 2. 1 开启或关闭外设

I2C 外设时钟必须在时钟控制器中进行配置和使能。然后可通过将 **I2C_CON1** 寄存器中的 PE 位设置为 1 开启 I2C。当关闭 I2C(PE=0)时，I2C 将执行软件复位。有关更多详细信息，请参见软件复位。

26. 5. 2. 2 数字噪声滤波器

在开启 I2C(PE=1)前，用户可以通过配置 **I2C_CON1** 寄存器中的 DNF 位域来开启数字滤波器。开启数字滤波器(DNF 不为 0)时，总线上的 SCL 或 SDA 信号只有在电平稳定时间超过 DNF x I2CCLK 个周期后才会发生反应到硬件内部。因此可抑制在 1 到 15 个 I2CCLK 周期的尖峰脉宽。

注：当开启 I2C(PE=1)时，无法更改 DNF 位域。

26. 5. 2. 3 I2C 时序

用户必须配置 **I2C_TIMINGR** 寄存器中的 PRESC、SCLDEL 和 SDADEL 位域，以确保主机模式或从机模式使用正确的数据建立时间与数据保持时间。

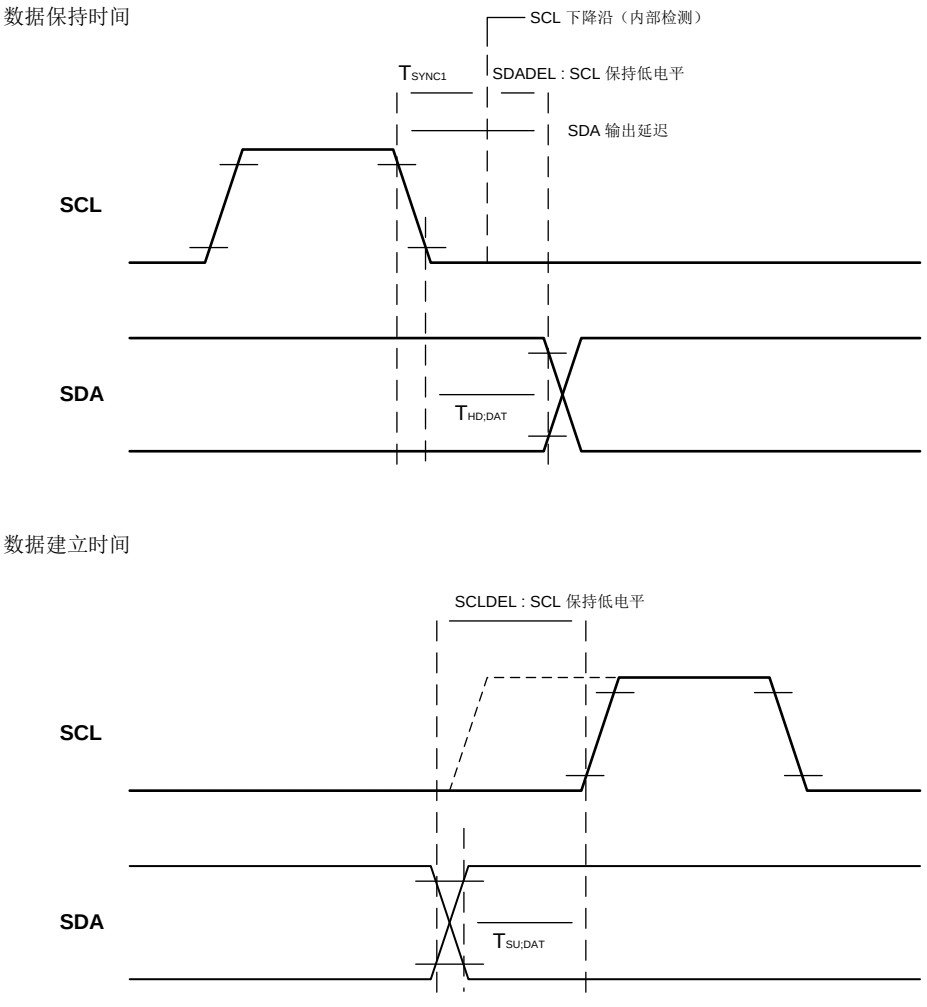


图 27-8 设置和保持时序

◆ 数据保持时间

当内部检测到 SCL 下降沿后，会先插入一段延迟时间，然后在发送 SDA 输出。该延迟时间为:

T_{SDADEL} = SDADEL x T_{PRESC} + T_{I2CCLK}

注：

1. T_{PRESC} = (PRESC+1) x T_{I2CCLK}。
2. T_{SDADEL} 会影响数据保持时间 $T_{HD;DAT}$。

SDA 总输出延迟时间为:

T_{SYNC1} + {[SDADEL x (PRESC+1) + 1] x T_{I2CCLK}}

T_{SYNC1} 的持续时间取决于以下参数:

◆ SCL 下降斜率

◆ 开启数字噪声滤波器引起的输入延迟: DNF x T_{I2CCLK}

◆ 同步 SCL 与 I2CCLK 时钟所产生的延迟: 2 至 3 个 I2CCLK 周期

在 SCL 下降沿会有一个未定义的区域，用户配置 SDADEL 时必须符合以下条件:

{$T_{f(max)}$ + $T_{HD;DAT(min)}$ - [(DNF+3) x T_{I2CCLK}]} / {(PRESC + 1) x T_{I2CCLK}} ≤ SDADEL

SDADEL ≤ {$T_{HD;DAT(max)}$ - [(DNF+4) x T_{I2CCLK}]} / {(PRESC + 1) x T_{I2CCLK}}

注：

1. 标准模式、快速模式和极快速模式下 $T_{HD;DAT}$ 的最大值分别为 3.45μs、0.9μs 和 0.45μs。
2. 只有在未延长 SCL 时钟信号的低电平周期(T_{LOW})时，才必须满足该最大值条件。
3. 如有延长 SCL 时钟，在释放 SCL 时钟之前 SDA 必须在数据建立时间内保持有效电平。
4. 主机发送或从机发送时，如果 I2C_TXDATA 内已有数据，在 SDADEL 延迟后通过 SDA 输出。
5. 有关 T_f 和 $T_{HD;DAT}$ 的标准值信息，请参见表 27-1。

◆ 数据建立时间

在 T_{SDADEL} 延时后，数据还没写入到 **I2C_TXDATA** 导致必须延长 SCL，当数据写入到 **I2C_TXDATA** 后，硬件会先输出 SDA，并将 SCL 保持低电平值到满足数据建立时间后释放。该数据建立时间为:

T_{SCLDEL} = (SCLDEL + 1) x T_{PRESC}

注：

1. T_{PRESC} = (PRESC+1) x T_{I2CCLK}。
2. T_{SCLDEL} 会影响数据建立时间 $T_{SU;DAT}$。

在 SDA 上升沿会有一个未定义的区域，用户配置 SCLDEL 时必须符合以下条件:

{[$T_{r(max)}$ + $T_{SU;DAT(min)}$] / [(PRESC+1)] x T_{I2CCLK}]} - 1 ≤ SCLDEL

注：

1. 在发送和接收模式下，在每个时钟脉冲检测到 SCL 下降沿后，I2C 主机或从机至少会在 [(SDADEL+SCLDEL+1) x (PRESC+1) + 1] x TI2CCLK 期间内延长 SCL 低电平时间。
2. 有关 T_r 和 $T_{SU;DAT}$ 的标准值信息，请参见表 27-1。
3. 从机模式下 NOSTRETCH 位为 1 时，SCL 不会延长。因此用户配置 SDADEL 时，必须确保提供充足的数据建立时间。

参数	参数名称	标准模式(Sm)		快速模式(Fm)		极快速模式(Fm+)		SMBus		单位
		最小值	最大值	最小值	最大值	最小值	最大值	最小值	最大值	
$T_{HD;DAT}$	数据保持时间	0	3.45	0	0.9	0	0.45	0.3	-	μs
$T_{SU;DAT}$	数据建立时间	250	-	100	-	50	-	250	-	ns
T_r	SDA 和 SCL 上升时间	-	1000	-	300	-	120	-	1000	
T_f	SDA 和 SCL 下降时间	-	300	-	300	-	120	-	300	

表 27-1 I2C-SMBUS 规范数据建立和保持时间

- SCL 时钟周期(主机模式)

  在主机模式下，用户必须设置 **I2C_TIMINGR** 寄存器中的 PRESC、SCLH 和 SCLL 位域来配置 SCL 时钟的高电平和低电平。

  - 當內部檢測到 SCL 下降沿時，會在釋放 SCL 輸出變為高電平之前插入一段延時。該延時時間為: T_{SCLL} =(SCLL+1) x T_{PRESC}
  - 當內部檢測到 SCL 上升沿時，會在強制 SCL 輸出變為低電平之前插入一段延時。該延時時間為: T_{SCLH} =(SCLH+1) x T_{PRESC}

注：

1. T_{PRESC} = (PRESC+1) x T_{I2CCLK}。
2. T_{SCLL} 会影响 SCL 低电平时间 T_{LOW}。
3. T_{SCLH} 会影响 SCL 高电平时间 T_{HIGH}。
4. 更多详细信息，请参见 5.6.1, “I2C 主机模式初始化”。
5. 开启 I2C 后，无法更改 I2C_TIMINGR 寄存器与 I2C_CON1.NOSTRETCH 的设置值。

26. 5. 2. 4 I2C 初始化流程

I2C 初始化流程如下:

1. 初始化设置
2. 设置 **I2C_CON1**.PE 为 0
3. 配置 **I2C_CON1**.DNF
4. 配置 **I2C_TIMINGR** 寄存器中的 PRESC、SCLH、SCLL、SCLDEL 和 SDADEL 位域
5. 配置 **I2C_CON1**.NOSTRETCH

6. 设置 **I2C_CON1**.PE 为 1

26. 5. 3 软件复位

用户可以通过将 **I2C_CON1** 寄存器中的 PE 位设置为 0 来执行软件复位。在这种情况下，I2C 总线上的 SCL 和 SDA 将被释放。硬件内部的状态机也会复位，此外也会将通信控制位和状态位恢复为成复位值。

软件复位时，下列寄存器位会恢复为成复位值：

- **I2C_CON2** 寄存器：PECBYTE、START、STOP 和 NACK
- **I2C_STAT** 寄存器：BUSY、TCR、TC、RXNE 和 TXE

26. 5. 4 数据传输

数据传输是由发送和接收数据寄存器以及移位寄存器进行管理。

接收

SDA 输入到移位寄存器。在第 8 个 SCL 脉冲后(接收到一个数据字节)，如果 RXNE = 0，表示接收数据寄存器(**I2C_RXDATA**)为空，此时将移位寄存器的数据会复制到 **I2C_RXDATA**。如果 RXNE = 1，尚未读取先前接收的数据字节，此时将延 SCL 的低电平时间，直到读取 **I2C_RXDATA** 为止。

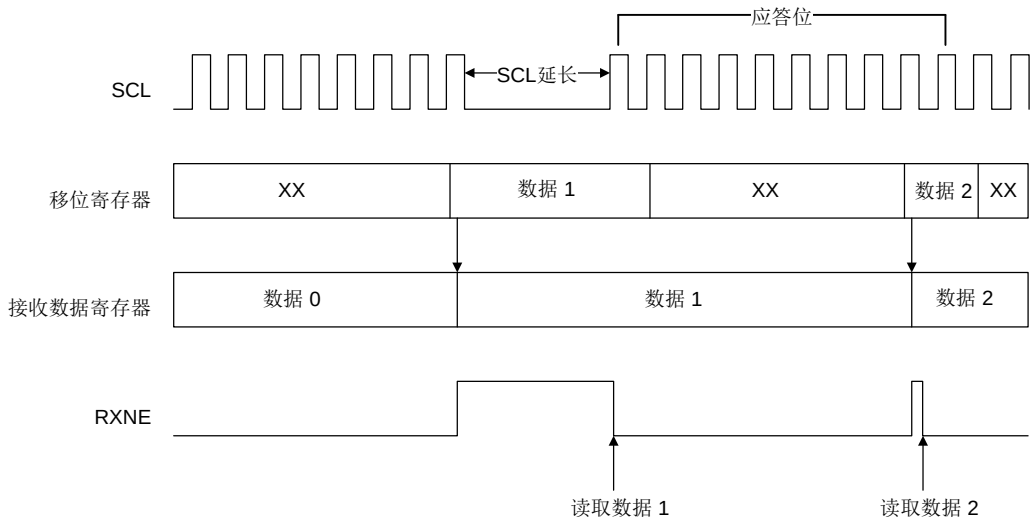


图 27-9 数据接收

发送

如果发送数据寄存器(**I2C_TXDATA**)不为空(TXE=0)，则在第 9 个 SCL 脉冲(应答位脉冲)之后将 **I2C_TXDATA** 的数据复制到移位寄存器中。然后移位寄存器的内容会在 SDA 上发送。如果 TXE=1，表示 **I2C_TXDATA** 尚未写入数据，此时将延长 SCL 的低电平时间，直到写入 **I2C_TXDATA** 为止。

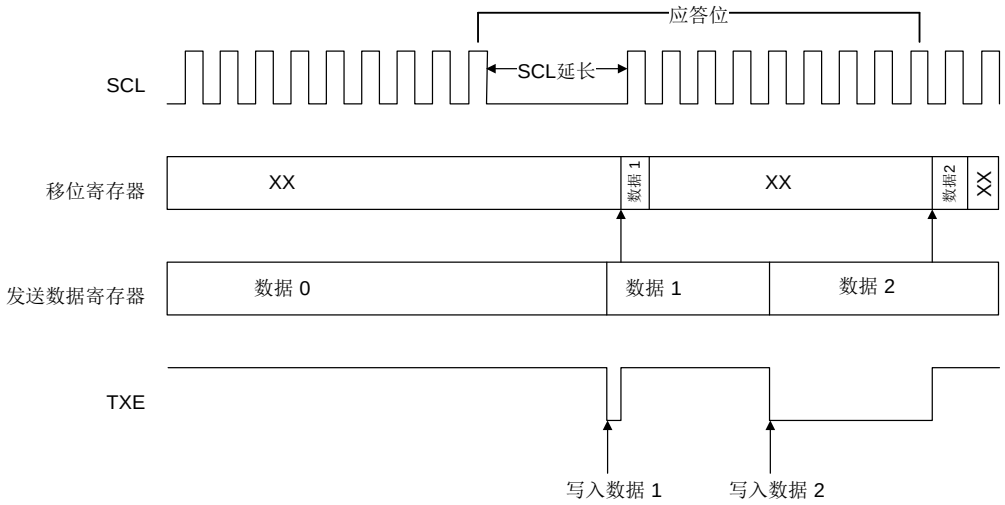


图 27-10　数据发送

硬件传输管理

I2C 在硬件中的具有字节计数器，在下列模式中管理字节传输和结束通信:

- 主机模式: 产生 NACK、STOP 和 RESTART
- 从机接收模式: 控制应答位回应 ACK 或 NACK
- SMBus 模式: PEC 产生或校验

字节计数器在主机模式下始终为开启状态，在从机模式下默认为禁止状态。在从机模式下可以通过软件设置 **I2C_CON1** 寄存器中的 SBC(从机字节控制)位为 1 开启。

传输的字节数(NBYTES)是由 CON1.NBYTESH 与 CON2.NBYTESL 组合成一个 16 位，NBYTES 填写方式为先填写 CON1.NBYTESH(高 8 位)，接着在填写 CON2.NBYTESL(低 8 位)，如果待传输的 NBYTES 大于 65535 或者在接收时想要控制接收数据字节的应答值，则必须通过设置 **I2C_CON2** 寄存器中的 RELOAD 位为 1 来选择重载模式。在此模式下，当传输的字节数达到 NBYTES 时，硬件会将 **I2C_STAT** 寄存器中的 TCR 位设置为 1，如果 **I2C_IER** 寄存器中的 TCR 位为 1 则会产生中断，通过对 **I2C_ICR** 寄存器中的 TCR 位设置为 1 清除中断。只要 **I2C_STAT**.TCR 位为 1 时，SCL 就会被延长。当软件对 NBYTES 写入非零值时，硬件会自动清除 **I2C_STAT**.TCR。

在最后一次传输设置 NBYTES 前，必须先把 **I2C_CON2** 寄存器中的 RELOAD 位设置为 0。

当主机模式下 RELOAD=0 时，字节计数器可用于下列两种模式中 ：

- **自动结束模式(I2C_CON2** 寄存器中的 AUTOEND 位为 1)。
  在此模式下，一旦传输的字节数达到 NBYTES 时，主机就会自动发送 STOP 条件。
- **软件结束模式(I2C_CON2** 寄存器中的 AUTOEND 位为 0)。
  在此模式下，一旦传输的字节数达到 NBYTES 时，硬件会将 **I2C_STAT** 寄存器中的 TC 位设置为 1，如果 **I2C_IER** 寄存器中的 TC 位为 1 则会产生中断，通过对 **I2C_ICR** 寄存器中的 TC 位设置为 1 清除中断。只要 **I2C_STAT**.TC 为 1 时，SCL 就会被延长。当软件设置 **I2C_CON** 寄存器中的 START 或 STOP 位为 1 时，会清除 **I2C_STAT**.TC。当主机要发送 RESTART 条件时，必须使用此模式。

注：当 I2C_CON2 寄存器中的 RELOAD 位为 1 时，AUTOEND 的功能无效。

26. 5. 5 I2C从机模式

26. 5. 5. 1 I2C从机模式初始化

要在从机模式下工作，用户至少要将一个本机地址开启。**I2C_ADDR1** 和 **I2C_ADDR2** 这两个寄存器可用于配置本机地址 OA1 和 OA2。

- 本机地址 1: 在默认情况下是 7 位地址模式(**I2C_ADDR1**.OA1MODE=0)，通过设置 **I2C_ADDR1** 寄存器中的 OA1MODE 位为 1 配置成 10 位地址模式，通过设置 **I2C_ADDR1** 寄存器中的 OA1EN 位为 1 开启 OA1。
- 本机地址 2: 提供第二个 7 位的本机地址 OA2，通过设置 **I2C_ADDR2** 寄存器中的 OA2EN 位为 1 开启 OA2。根据 **I2C_ADDR2** 寄存器中的 OA2MSK 位域配置值，OA2 对应到的屏蔽情况如下:
  - ◇ OA2MSK=0 ： 使用 OA2 与接收到的地址做比较，如果匹配则响应 ACK。如果用户将 OA2 配置为保留地址(0000 XXX 和 1111 XXX)时，当接收到的地址为保留地址时响应 ACK。
  - ◇ OA2MSK=1~6: 分别只有使用 OA2[7:2]、OA2[7:3]、OA2[7:4]、OA2[7:5]、OA2[7:6]或 OA2[7]与接收到的地址(保留地址除外)做比较，如果匹配则回应 ACK。
  - ◇ OA2MSK=7: 对所有接收到的地址(保留地址除外)响应 ACK。
- 广播地址: 通过设置 **I2C_CON1** 寄存器中的 GCEN 位为 1 开启广播地址。

当接收到的地址与本机地址(1 或 2)或广播地址比较结果为匹配时，**I2C_RIF** 寄存器中的 ADDR 位会被设置为 1。如果 **I2C_IER** 寄存器中的 ADDR 位为 1，则会产生中断。

在默认情况下，从机使用时钟延长功能，这表示在某些情况下会将 SCL 拉到低电平，让用户可以执行软件操作。如果主机不支持时钟延长，则必须将 **I2C_CON1** 寄存器中的 NOSTRETCH 位设置为 1。

如果开启多个地址时，在收到地址匹配中断后，用户必须读取 **I2C_STAT** 寄存器中的 ADDCODE 位域，检查匹配的地址，以及检查 DIR 位了解传输方向。

从机时钟延长(NOSTRETCH=0)

当 **I2C_CON1**.NOSTRETCH 为 0 时，I2C 从机会在以下情况下延长 SCL 时钟:

- 接收到的地址与其中一个开启地址比较结果为匹配时，**I2C_RIF** 寄存器的 ADDR 位会被设置为 1。通过对 **I2C_ICR** 寄存器中的 ADDR 位设置 1，将 **I2C_RIF** 寄存器的 ADDR 位清零，同时也会释放此时钟延长。
- 在发送模式下，先前数据传输已完成，但还没对 **I2C_TXDATA** 写入新的数据时，或者 **I2C_RIF**.ADDR 为 0 且尚未写入第一个数据(**I2C_STAT**.TXE 为 1)时。当对 **I2C_TXDATA** 写入数据时，将释放此时钟延长。
- 在接收模式下，在 **I2C_STAT**.RXNE 为 1 后，但还没对 **I2C_RXDATA** 读取数据时。当读取 **I2C_RXDATA** 时，将释放此时钟延长。
- 在从机字节控制模式(SBC=1)下，开启重载模式(RELOAD=1)时，当传输的字节数达到 NBYTES 时，会将 **I2C_STAT** 寄存器中的 TCR 位设置为 1。当对 NBYTE 写入的数值不为零时，将 **I2C_STAT** 寄存器的 TCR 位清零，同时也会释放此时钟延长。
- 在 SCL 下降沿检测后，I2C 延长 SCL 的低电平时间为:

  $[(SDADEL+SCLDEL+1) \times (PRESC+1) + 1] \times T_{I2CCLK}$

从机没有时钟延长(NOSTRETCH=1)

当 **I2C_CON1**.NOSTRETCH 为 1 时，I2C 从机不会延长 SCL 时钟:

- **I2C_RIF**.ADDR 为 1 时。不会延长 SCL 时钟。
- 在发送模式下，必须在与发送数据对应的第一个 SCL 脉冲出现之前，将数据写入 **I2C_TXDATA**。否则会将 **I2C_STAT** 寄存器中的 TXUD 位设置为 1，表示发生下溢，如果 **I2C_IER** 寄存器中的 TXUD 位为 1 则会产生中断。
- 在接收模式下，必须在下一个数据字节的第 9 个 SCL 脉冲(应答位脉冲)出现之前，从 **I2C_RXDATA** 寄存器读取数据。否则会将 **I2C_STAT** 寄存器中的 RXOV 位设置为 1，表示发生上溢，如果 **I2C_IER** 寄存器中的 RXOV 位为 1 则会产生中断。

从机字节控制模式

在从机接收模式下要实现字节的应答位控制，必须通过设置 **I2C_CON1** 寄存器中的 SBC 位为 1 来开启从机字节控制模式。藉此符合 SMBus 标准。

同时必须选择重载模式(RELOAD=1)。如果要控制每个字节的应答位时，必须在 ADDR 中断子程序中将 NBYTES 设置为 0x1，并在每接收一个字节后将 NBYTE 设置为 0x1。

当接收到字节后，**I2C_STAT**.TCR 位将设置为 1，同时在第 8 个和第 9 个 SCL 脉冲之间，将 SCL 拉低延长低电平时间。用户可以从 **I2C_RXDATA** 寄存器读取数据，然后通过设置 **I2C_CON2** 寄存器中的 NACK 位决定应答值。通过将 NBYTES 设置为非零值来释放 SCL 时钟延长，并发送 ACK 或 NACK 信号，接着可以继续接收下一个字节。

NBYTES 可以设置大于 0x1 的值，在这种情况下，接收过程在 NBYTES 个数据接收期间是连续的。

注:

1. 只有在 I2C 关闭(PE=0)时、从机尚未收到地址匹配时或 I2C_RIF.ADDR 为 1 时，才能设置 SBC 位。
2. 在 I2C_RIF.ADDR= 1 或 I2C_RIF.TCR= 1 时，可以更改 RELOAD 位的值。
3. 从机字节控制模式与 NOSTRETCH 模式不兼容。不允许在 NOSTRETCH=1 时设置 SBC 位为 1。

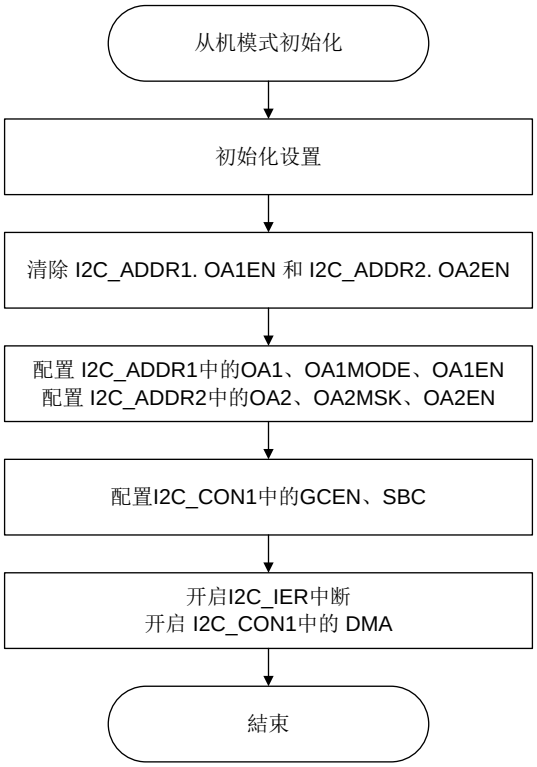


图 27-11　从机初始化流程图

26. 5. 5. 2　从机传送

在接收到匹配的地址时，**I2C_RIF** 寄存器中的 ADDR 位被设置为 1，若此时已准备好要发送的数据时，会通过内部移位寄存器发送到 SDA。若此时 **I2C_STAT**.TXE 为 1 且 **I2C_CON1**.NOSTRETCH=0 时，从机会延长 SCL 低电平时间，直到 **I2C_TXDATA** 寄存器写入发送数据为止。

当接收到 NACK 时，会将 **I2C_RIF** 寄存器中的 NACK 位设置为 1，如果 **I2C_IER** 寄存器中的 NACK 位为 1 则会产生中断。此时从机会自动释放 SCL 与 SDA 信号，让主机能够发送 STOP 或 RESTART。

当接收到 STOP 时，会将 **I2C_RIF** 寄存器中的 STOP 位设置为 1，如果 **I2C_IER** 寄存器中的 STOP 位为 1 则会产生中断。并结束通信，等待下一次收到匹配的地址。

当 **I2C_RIF**.ADDR= 1 且 **I2C_STAT**.TXE= 0 时，用户可以选择发送 **I2C_TXDATA** 寄存器的内容作为第一个数据字节，也可以通过 **I2C_STAT** 寄存器中的 TXE 位设置 1 后，对 **I2C_TXDATA** 写入新的数据字节。

在从机字节控制模式下，必须在 ADDR 中断子程序中将待发送数据字节数写入到 NBYTES 中。

注:如果 NOSTRETCH 模式开启，当 I2C_RIF.ADDR= 1 时不会延长 SCL 时钟，因此必须提前将第一个数据字节写入到 I2C_TXDATA 寄存器。

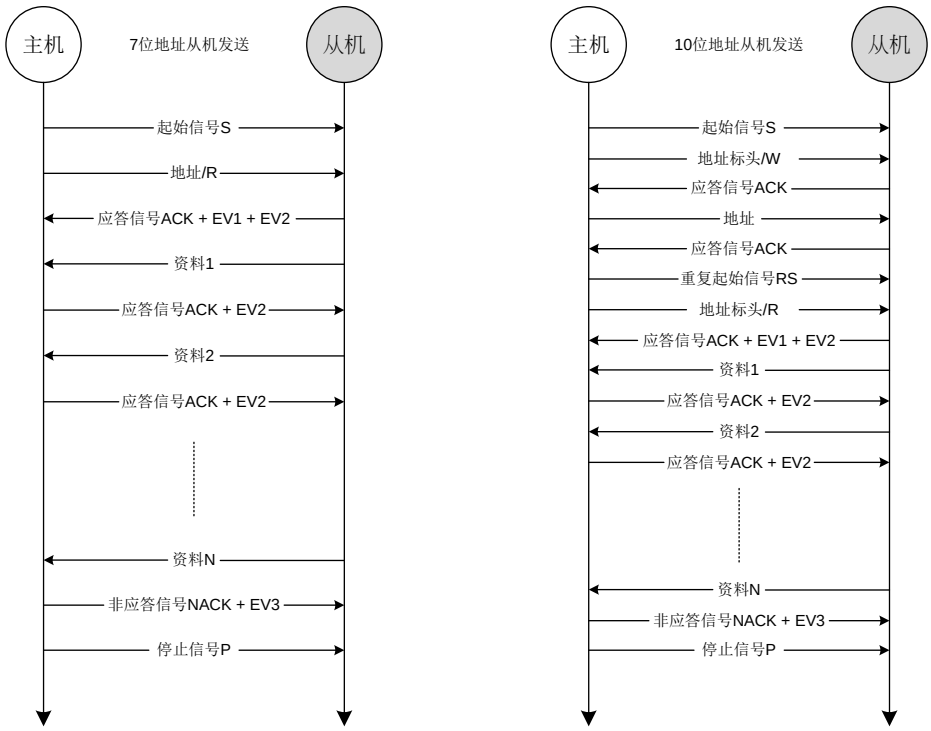


图 27-12　从机发送的传输序列图

注:

1. S=起始位，RS=重复起始位，P=停止位，ACK=应答，NACK=非应答
2. EV1：当收到匹配地址时，I2C_RIF 寄存器中的 ADDR 位被设置为 1，设置 I2C_ICR 寄存器中的 ADDR 位为 1 来清除中断。
3. EV2：判断 I2C_STAT 寄存器中的 TXE 位，若为 1 则写入数据至 I2C_TXDATA 寄存器。
4. EV3：当收到非应答信号 NACK 时，I2C_RIF 寄存器中的 NACK 位被设置为 1，设置 I2C_ICR 的 NACK 位为 1 来清除中断。此时还需判断是否还有尚未发送的字节，若有则软件需要额外处理。
5. 如果软件列在当前字节传输结束之前尚未写入下一个数据字节，EV2 事件将会延长 SCL 时钟低电平时间。

26. 5. 5. 3　从机接收

当接收到一个完整的数据字节时，会通过内部移位寄存器复制到 **I2C_RXDATA** 寄存器中，并将 **I2C_RIF** 寄存器中的 RXNE 位设置为 1，如果 **I2C_IER** 寄存器中的 RXNE 位为 1 则会产生中断。

当接收到 STOP 时，会将 **I2C_RIF** 寄存器中的 STOP 位设置为 1，如果 **I2C_IER** 寄存器中的 STOP 位为 1 则会产生中断。并结束通信，等待下一次收到匹配的地址。

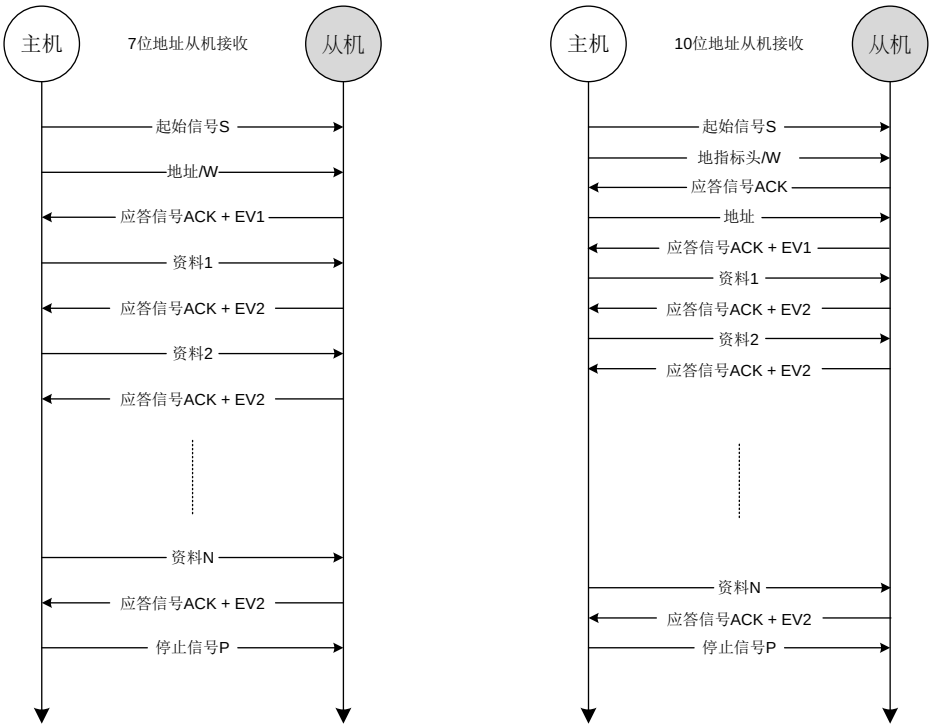


图 27-13　从机接收的传输序列图

注:

1. S=起始位，RS=重复起始位，P=停止位，ACK=应答，NACK=非应答
2. EV1：当收到匹配地址时，I2C_RIF 寄存器中的 ADDR 位被设置为 1，设置 I2C_ICR 寄存器中的 ADDR 位为 1 来清除中断。
3. EV2：判断 I2C_STAT 寄存器中的 RXNE 位，若为 1 则读取 I2C_RXDATA 寄存器数据。
4. 如果软件在 I2C_STAT.RXNE 为 1 时，未能立即读取 I2C_RXDATA 寄存器，EV2 事件将会延长 SCL 时钟低电平时间。

26. 5. 6 I2C主机模式

26. 5. 6. 1 I2C主机模式初始化

在开启 I2C 外设前，用户必须设置 **I2C_TIMINGR** 寄存器中的 PRESC、SCLH 和 SCLL 位域来配置 I2C 主机时钟。I2C 外设具有时钟同步机制，可以支持多主机环境和从机时钟延长。

为了实现时钟同步，必须执行下列操作:

- SCLL 计数器从内部检测到 SCL 低电平后，开始对时钟低电平进行计数。
- SCLH 计数器从内部检测到 SCL 高电平后，开始对时钟高电平进行计数。

I2C 在 T_{SYNC1} 延迟后检测自己的 SCL 低电平，该延迟时间取决于 SCL 下降沿、SCL 输入数字噪声滤波器和 SCL 与 I2CCLK 时钟的同步。当 SCLL 计数器达到 **I2C_TIMINGR** 寄存器中的 SCLL 位域的设置值后，I2C 会将 SCL 释放为高电平。

I2C 在 T_{SYNC2} 延迟后检测自己的 SCL 高电平，该延迟时间取决于 SCL 上升沿、SCL 输入数字噪声滤波器和 SCL 与 I2CCLK 时钟的同步。当 SCLH 计数器达到 **I2C_TIMINGR** 寄存器中的 SCLH 位域的设置值后，I2C 会将 SCL 拉低到低电平。

主机时钟周期为:

$T_{SCL}=T_{SYNC1} + T_{SYNC2} + \{ [(SCLH+1) + (SCLL+1)] \times (PRESC+1) \times T_{I2CCLK} \}$

T_{SYNC1} 的持续时间取决于以下参数:

- SCL 下降斜率
- 开启数字噪声滤波器后所增加的输入延迟:DNF x T_{I2CCLK}
- 同步 SCL 与 I2CCLK 时钟所导致的延迟: 2 至 3 个 I2CCLK 周期

T_{SYNC2} 的持续时间取决于以下参数:

- SCL 上升斜率
- 开启数字噪声滤波器后所增加的输入延迟:DNF x T_{I2CCLK}
- 同步 SCL 与 I2CCLK 时钟所导致的延迟: 2 至 3 个 I2CCLK 周期

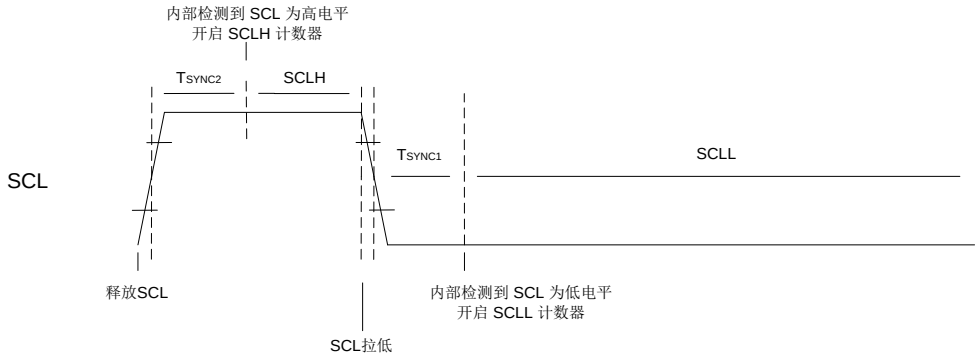


图 27-14 主机时钟产生

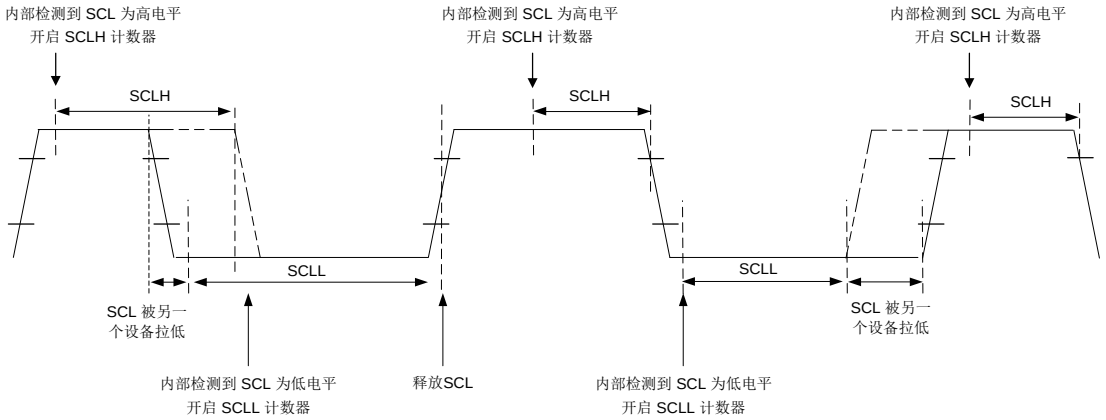


图 27-15　SCL 主机时钟同步

为了符合 I2C 或 SMBus 规范，主机时钟必须遵循下表的时序:

参数	参数名称	标准模式(Sm)		快速模式(Fm)		极快速模式(Fm+)		SMBus		单位
		最小值	最大值	最小值	最大值	最小值	最大值	最小值	最大值	
F_{SCL}	SCL 时钟频率	-	100	-	400	-	1000	-	100	kHz
$T_{HD;STA}$	(重复)起始条件的保持时间	4.0	-	0.6	-	0.26	-	4.0	-	μs
$T_{SU;STA}$	(重复)起始条件的建立时间	4.7	-	0.6	-	0.26	-	4.7	-	μs
$T_{SU;STO}$	停止条件的建立时间	4.0	-	0.6	-	0.26	-	4.0	-	μs
T_{BUF}	停止和起始条件之间的空闲时间	4.7	-	1.3	-	0.5	-	4.7	-	μs
T_{LOW}	SCL 时钟的低电平时间	4.7	-	1.3	-	0.5	-	4.7	-	μs
T_{HIGH}	SCL 时钟的高电平时间	4.0	-	0.6	-	0.26	-	4.0	50	μs
T_r	SDA 和 SCL 上升时间	-	1000	-	300	-	120	-	1000	ns
T_f	SDA 和 SCL 下降时间	-	300	-	300	-	120	-	300	ns

表 27-2　I2C-SMBUS 规范的时钟时序

注:

1. SCLL 同時也用於產生 T_{BUF} 和 $T_{SU;STA}$ 時序
2. SCLH 同時也用於產生 $T_{HD;STA}$ 和 $T_{SU;STO}$ 時序
3. I2C_TIMINGR 配置與 I2CCLK 頻率的示例，請參見 27. 5. 7

主机通信初始化(地址阶段)

在发起通信前，用户必须对 **I2C_CON2** 寄存器中有关从机地址参数进行配置:

- ADD10：地址模式(7 位或 10 位)
- SADD：从机地址
- RD_WRN：传输方向
- HEAD10R：必须设置 HEAD10R 位以表示是否必须发送完整的地址序列，或者仅发送 10 位地址的前 7 位地址标头
- NBYTES：传输的字节数。如果字节数等于或大于 65535 个字节，则 NBYTES 可以先填入 0xFFFF。

接着设置 **I2C_CON2** 寄存器中的 START 位为 1。当 START 位为 1 时，不允许更改上述所有位的设定值。当检测到总线空闲(BUSY=0)后，会在经过 TBUF 的延迟时间后自动发送起始位(START)，接着发送从机地址。

在仲裁丢失的情况下，主机自动切换回从机模式，如果在从机模式下发生地址匹配，则可以对本机地址进行应答。

注:

1. 不管接收到的应答值是 ACK 或 NACK，当主机发送在总线上的从机地址已传输完成时，START 位由硬件清除。如果发生仲裁丢失，则 START 位也由硬件清除。
2. 在 10 位地址模式下，当发送的从机地址前 7 位被从机响应 NACK 时，主机会自动重新发送从机地址前 7 位的传输，直到收到从机响应 ACK。如果在 START 位为 1 时，当发生生地址匹配中断导致 ADDR 位为 1，此时 START 位会被清零，I2C 切换到从机模式。
3. 当 BUSY = 1 时，该步骤将产生重复起始位。

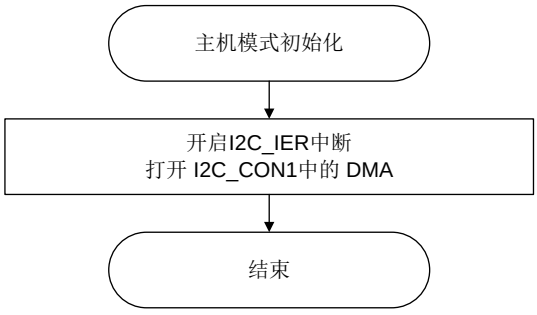


图 27-16　主机初始化流程图

主机接收模式下，采用 10 位地址模式的初始化

- 如果从机地址采用 10 位格式，用户可以将 **I2C_CON2** 寄存器中的 HEAD10R 位清零来选择发送完整的读序列。在这种情况下，主机在 START 位设置为 1 后自动发送以下完整序

列:

起始位 + 包含写入方向的 2 字节 10 位地址 + 重复起始位 + 包含读取方向的 10 位地址的前 7 位

如果主机寻址 10 位地址从机，将数据发送到该从机，然后从同一从机读取数据，则必须先完成主机传输流程，然后使用 HEAD10R=1 配置的 10 位从地址设置重复起始。在这种情况下，主机发送此序列:重复起始+从机地址 10 位标头读取。

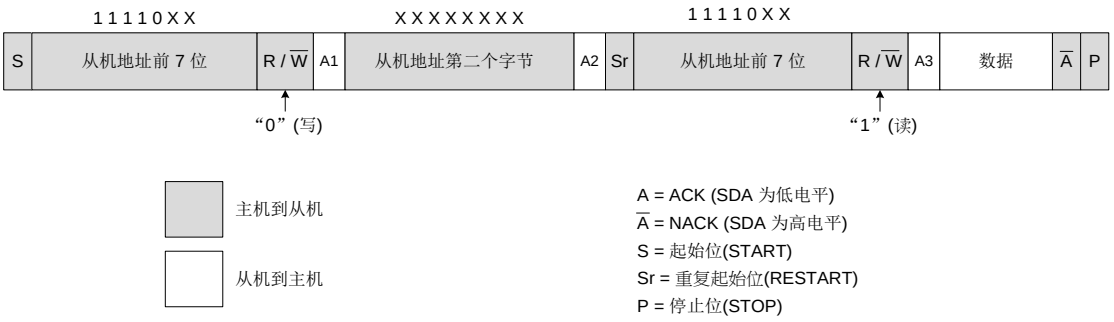


图 27-17　主机接收模式(ADD10=1、HEAD10R=0)

◆ 在从机地址采用 10 位格式时，想要先将数据发送到从机，接着再从该从机读取数据时，则必须先完成主机发送流程。然后将 **I2C_CON2** 寄存器中的 HEAD10R 与 START 位设置为 1，在这种情况下，主机发送以下序列:

重复起始位 + 从机地址 10 位地址的前 7 位，然后发送读取方向

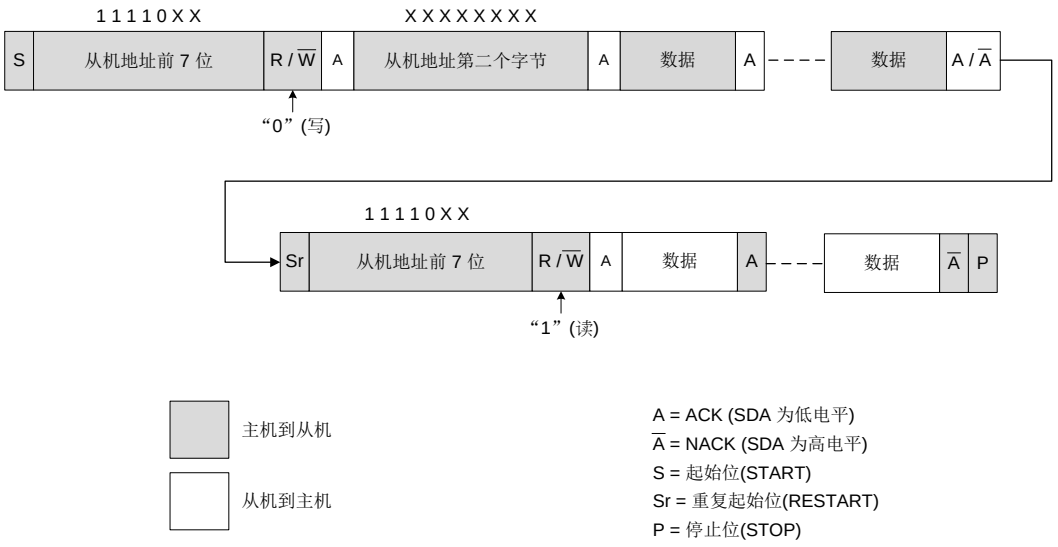


图 27-18　主机接收模式(ADD10=1、HEAD10R=0)

26. 5. 6. 2　主机发送

传输的字节数(NBYTES)是由 CON1.NBYTESH 与 CON2.NBYTESL 组合成一个 16 位，NBYTES 填写方式为先填写 CON1.NBYTESH(高 8 位)，接着在填写 CON2.NBYTESL(低 8 位)，如果待发送传输的 NBYTES 大于 65535 时，则必须通过设 **I2C_CON2** 寄存器中的 RELOAD 位为 1 来选择重载模式。在此模式下，当发送传输的字节数达到 NBYTES 时，硬件会将 **I2C_STAT** 寄存器中的 TCR 位设置为 1，如果 **I2C_IER** 寄存器中的 TCR 位为 1 则会产生中断，通过对 **I2C_ICR** 寄存器中的 TCR 位设置为 1 清除中断。只要 **I2C_STAT**.TCR 位为 1 时，SCL 就会被延长。当软件对 NBYTES 写入非零值时，硬件会自动清除 **I2C_STAT**.TCR。

当主机收到从机响应 NACK 时，**I2C_RIF** 寄存器中的 NACK 位被设置为 1 后，主机会自动发送停止位(STOP)。如果 **I2C_IER** 寄存器中的 NACK 位为 1 则会产生中断，通过对 **I2C_ICR** 寄存器中的 NACK 位设置为 1 清除中断。

当主机收到从机响应 ACK 后，如果 RELOAD=0 并且发送传输的字节数达到 NBYTES 时，会有以下情况:

- 自动结束模式(AUTOEND=1)下，将自动发送停止位(STOP)。
- 软件结束模式(AUTOEND=0)下，硬件会将 **I2C_STAT** 寄存器中的 TC 位设置为 1，此时主机会将 SCL 时钟保持低电平，等待软件控制后续操作:
  - RESTART: 设置 **I2C_CON2** 寄存器中的 START 位为 1，此时会发送重复起始位。
  - STOP: 设置 **I2C_CON2** 寄存器中的 STOP 位为 1，此时会发送停止位。

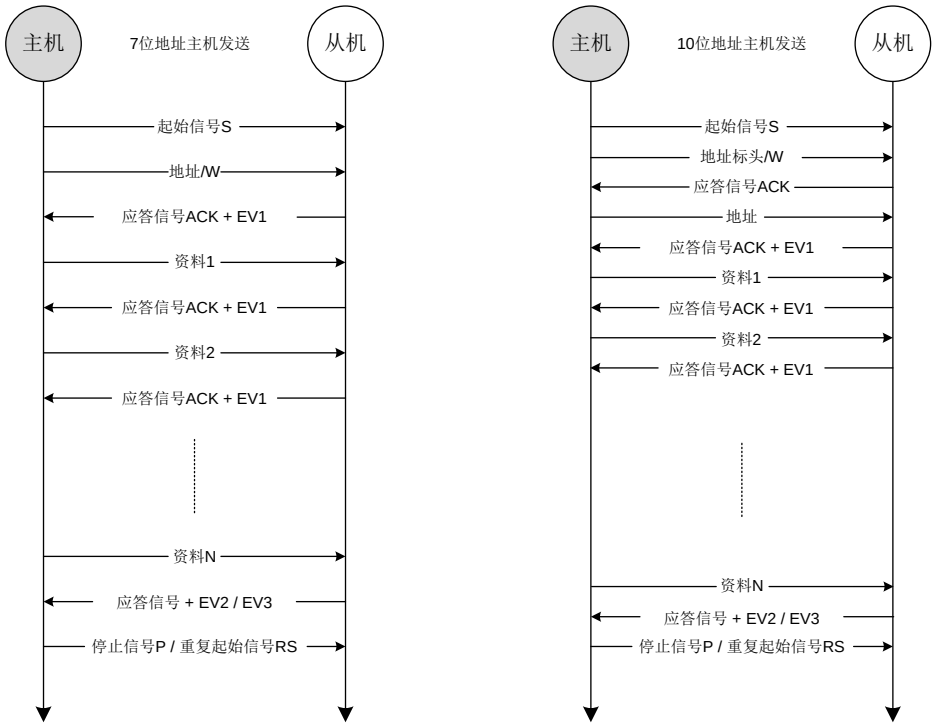


图 27-19　主机发送的传输序列图

注:
1. S=起始位，RS=重复起始位，P=停止位，ACK=应答，NACK=非应答
2. EV1=传输尚未完成事件，判断 I2C_STAT 寄存器中的 TXE 位，若为 1 写入数据至 I2C_TXDATA 寄存器。
3. EV2=当收到应答信号 ACK 时，若传输已完成，后续操作为停止位(STOP)或重复起始位(RESTART)。
4. EV3=当收到非应答信号 NACK 时，后续操作为停止位(STOP)。
5. 如果软件列在当前字节传输结束之前尚未写入下一个数据字节，EV1 事件将会延长 SCL 时钟低电平时间。

26. 5. 6. 3 主机接收

传输的字节数(NBYTES)是由 CON1.NBYTESH 与 CON2.NBYTESL 组合成一个 16 位，NBYTES 填写方式为先填写 CON1.NBYTESH(高 8 位)，接着在填写 CON2.NBYTESL(低 8 位)，如果待接收传输的 NBYTES 大于 65535 时，则必须通过设 **I2C_CON2** 寄存器中的 RELOAD 位为 1 来选择重载模式。在此模式下，当接收传输的字节数达到 NBYTES 时，硬件会将 **I2C_STAT** 寄存器中的 TCR 位设置为 1，如果 **I2C_IER** 寄存器中的 TCR 位为 1 则会产生中断，通过对 **I2C_ICR** 寄存器中的 TCR 位设置为 1 清除中断。只要 **I2C_STAT**.TCR 位为 1 时，SCL 就会被延长。当软件对 NBYTES 写入非零值时，硬件会自动清除 **I2C_STAT**.TCR。

当 RELOAD=0 并且接收传输的字节数达到 NBYTES 时，会有以下情况:

- 自动结束模式(AUTOEND=1)下，将自动发送 NACK 和停止位(STOP)。
- 软件结束模式(AUTOEND=0)下，将自动发送 NACK，并将 **I2C_STAT** 寄存器中的 TC 位设置为 1，此时主机会将 SCL 时钟保持低电平，等待软件控制后续操作:
  - RESTART: 设置 **I2C_CON2** 寄存器中的 START 位为 1，此时会发送重复起始位。
  - STOP: 设置 **I2C_CON2** 寄存器中的 STOP 位为 1，此时会发送停止位。

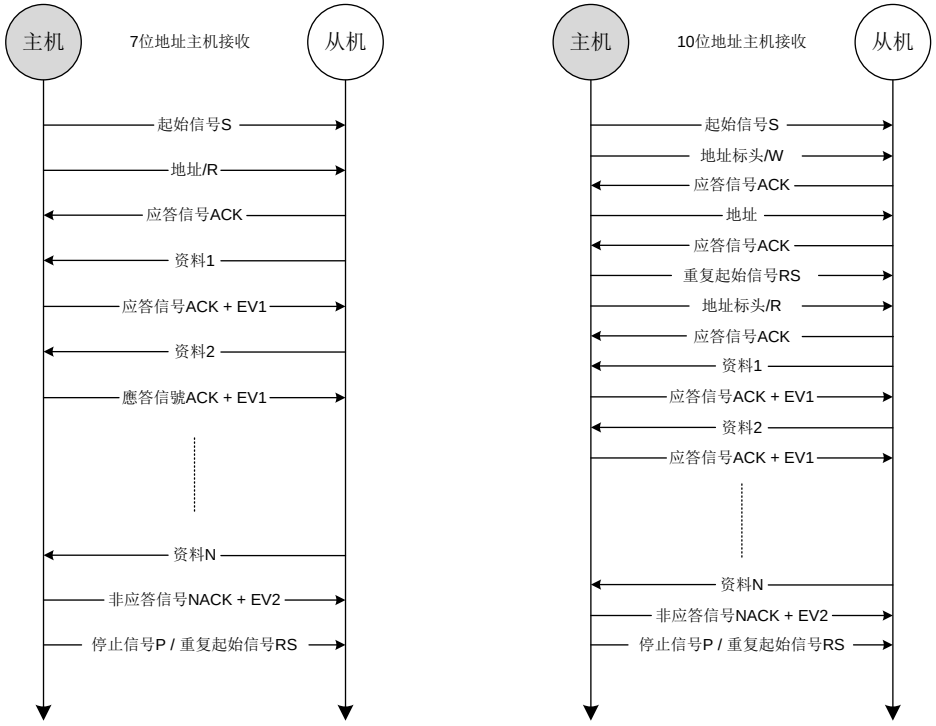


图 27-20　主机接收的传输序列图

注:

1. S=起始位，RS=重复起始位，P=停止位，ACK=应答，NACK=非应答
2. EV1=传输尚未完成事件，判断 I2C_STAT 寄存器中的 RXNE 位，若为 1 则读取 I2C_RXDATA 寄存器数据。
3. EV2=传输完成事件，此时会自动发送 NACK，后续根据软件配置来决定是发送停止位还是重复起始位。
4. 如果软件在 I2C_STAT.RXNE 为 1 时，未能立即读取 I2C_RXDATA 寄存器，EV1 事件将会延长 SCL 时钟低电平时间。

26. 5. 7　I2C_TIMINGR寄存器配置示例

下表提供了如何对 I2C_TIMINGR 进行设置以获得符合 I2C 规范的时序示例。

参数	标准模式(Sm)		快速模式(Fm)	极快速模式(Fm+)
	10KHz	100KHz	400KHz	500KHz
PRESC	1	1	0	0
SCLL	0xC7	0x13	0x9	0x6
T_{SCLL}	200x250 ns=50 µs	20x250 ns=5.0 µs	10x125 ns=1250 ns	7x125 ns=875 ns
SCLH	0xC3	0xF	0x3	0x3
T_{SCLH}	196x250 ns=49 µs	16x250 ns=4.0µs	4x125ns=500ns	4x125ns=500ns
T_{SCL}(1)	~100 µs(2)	~10 µs(2)	~2500 ns(3)	~2000 ns(4)
SDADEL	0x2	0x2	0x1	0x0

T_{SDADEL}	2x250 ns=500 ns	2x250 ns=500 ns	1x125 ns=125 ns	0 ns
SCLDEL	0x4	0x4	0x3	0x1
T_{SCLDEL}	5x250 ns=1250 ns	5x250 ns=1250 ns	4x125 ns=500 ns	2x125 ns=250 ns

表 27-3 F_{I2CCLK} = 8 MHz 的时序设置示例

1. 由于 SCL 内部检测延迟，SCL 周期 T_{SCL} 大于 T_{SCLL} + T_{SCLH}。为 T_{SCL} 提供的值仅为示例。
2. T_{SYNC1} + T_{SYNC2} 最小值为 4 x T_{I2CCLK} = 500 ns。T_{SYNC1} + T_{SYNC2} = 1000 ns 的示例。
3. T_{SYNC1} + T_{SYNC2} 最小值为 4 x T_{I2CCLK} = 500 ns。T_{SYNC1} + T_{SYNC2} = 750 ns 的示例。
4. T_{SYNC1} + T_{SYNC2} 最小值为 4 x T_{I2CCLK} = 500 ns。T_{SYNC1} + T_{SYNC2} = 655 ns 的示例。

参数	标准模式(Sm)		快速模式(Fm)	极快速模式(Fm+)
	10KHz	100KHz	400KHz	500KHz
PRESC	3	3	1	0
SCLL	0xC7	0x13	0x9	0x4
T_{SCLL}	200x250 ns=50 µs	20x250 ns=5.0 µs	10x125 ns=1250 ns	6x62.5 ns=312.5 ns
SCLH	0xC3	0xF	0x3	0x2
T_{SCLH}	196x250 ns=49 µs	16x250 ns=4.0µs	4x125ns=500ns	3x62.5ns=187.5ns
T_{SCL}(1)	~100 µs(2)	~10 µs(2)	~2500 ns(3)	~1000 ns(4)
SDADEL	0x2	0x2	0x2	0x0
T_{SDADEL}	2x250 ns=500 ns	2x250 ns=500 ns	2x125 ns=250 ns	0 ns
SCLDEL	0x4	0x4	0x3	0x2
T_{SCLDEL}	5x250 ns=1250 ns	5x250 ns=1250 ns	4x125 ns=500 ns	3x62.5 ns=187.5 ns

表 27-4 F_{I2CCLK} = 16 MHz 的时序设置示例

1. 由于 SCL 内部检测延迟，SCL 周期 T_{SCL} 大于 T_{SCLL} + T_{SCLH}。为 T_{SCL} 提供的值仅为示例。
2. T_{SYNC1} + T_{SYNC2} 最小值为 4 x T_{I2CCLK} = 250 ns。T_{SYNC1} + T_{SYNC2} = 1000 ns 的示例。
3. T_{SYNC1} + T_{SYNC2} 最小值为 4 x T_{I2CCLK} = 250 ns。T_{SYNC1} + T_{SYNC2} = 750 ns 的示例。
4. T_{SYNC1} + T_{SYNC2} 最小值为 4 x T_{I2CCLK} = 250 ns。T_{SYNC1} + T_{SYNC2} = 500 ns 的示例。

参数	标准模式(Sm)		快速模式(Fm)	极快速模式(Fm+)
	10KHz	100KHz	400KHz	500KHz
PRESC	0xB	0xB	5	5
SCLL	0xC7	0x13	0x9	0x3
T_{SCLL}	200x250 ns=50 µs	20x250 ns=5.0 µs	10x125 ns=1250 ns	4x125 ns=500 ns
SCLH	0xC3	0xF	0x3	0x1
T_{SCLH}	196x250 ns=49 µs	16x250 ns=4.0µs	4x125ns=500ns	2x125ns=250ns

T_{SCL}(1)	~100 µs(2)	~10 µs(2)	~2500 ns(3)	~875 ns(4)
SDADEL	0x2	0x2	0x3	0x0
T_{SDADEL}	2x250 ns=500 ns	2x250 ns=500 ns	3x125 ns=375 ns	0 ns
SCLDEL	0x4	0x4	0x3	0x1
T_{SCLDEL}	5x250 ns=1250 ns	5x250 ns=1250 ns	4x125 ns=500 ns	2x125ns=250ns

表 27-5 F_{I2CCLK} = 48 MHz 的时序设置示例

1. 由于 SCL 内部检测延迟，SCL 周期 T_{SCL} 大于 T_{SCLL} + T_{SCLH}。为 T_{SCL} 提供的值仅为示例。
2. T_{SYNC1} + T_{SYNC2} 最小值为 4 x T_{I2CCLK} = 83.3 ns。T_{SYNC1} + T_{SYNC2} = 1000 ns 的示例。
3. T_{SYNC1} + T_{SYNC2} 最小值为 4 x T_{I2CCLK} = 83.3 ns。T_{SYNC1} + T_{SYNC2} = 750 ns 的示例。
4. T_{SYNC1} + T_{SYNC2} 最小值为 4 x T_{I2CCLK} = 83.3 ns。T_{SYNC1} + T_{SYNC2} = 250 ns 的示例。

26. 5. 8 SMBus具体功能

介绍

系统管理总线(SMBus)是一个双线接口，各种设备可以通过该接口相互通信并或者与系统的其余部分通信。它基于 I2C 操作原理。SMBus 为系统和电源管理相关任务提供控制总线。

该外设与 SMBUS 规范相容(http://smbus.org)。

系统管理总线规范指的是三种类型的设备。

- 从机，用于接收或回应命令的设备。
- 主机，用于发出命令、产生时钟和停止传输的设备。
- 主控者，用于专用主机，为系统的 CPU 提供主接口。主控者必须可当作是主机或从机，并且必须支持 SMBus 主机通信协议。系统中只允许一个主控者。

该外设可以配置为主机或从机，也可以配置为主控者。

总线协议

对于任何给定的设备，有 11 种可用的命令协议。设备可以使用 11 个协议中的任何一个或全部来进行通信。协议包括快速命令、发送字节、接收字节、写入字节、写入字、读取字节、读取字、进程调用、块读取、块写入和块写入-块读取进程调用。这些协议应由用户软件实现。有关这些协议的更多详细信息，请参见 SMBus 规范(http://smbus.org)。

地址解析通讯协议(ARP)

可以通过为每个从设备动态分配新的唯一地址来解决 SMBus 从机地址冲突。为了提供隔离每个设备以便进行地址分配的机制，每个设备必须具有唯一的设备标识符(UDID)。UDID 为 128 位并由软件实现。

该外设支持地址解析通讯协议(ARP)。通过设置 **I2C_CON1** 寄存器中的 SMBDEN 位为 1 开启 SMBus 设备默认地址(0b1100 001)。ARP 命令由用户通过软件实现。仲裁也在从机模式下执行以支持 ARP。

有关 SMBus 地址解析通讯协议的更多详细信息，请参见 SMBus 规范(http://smbus.org)。

命令接收和数据接收的应答位控制

SMBus 在从机接收必须能够对每个接收到的命令或数据响应 NACK。为了在从机模式下允许 ACK 控制，必须通过设置 **I2C_CON1** 寄存器中的 SBC 位为 1 来开启从机字节控制模式。更多详细信息请参见从机字节控制模式。

主控者通知协议

该外设通过设置 **I2C_CON1** 寄存器中的 SMBHEN 位为 1 来支持主控者通知协议。在这种情况下，主控者将应答 SMBus 主机地址(0b0001 000)。使用此协议时，设备作为主机，主控者作为从机。

SMBus 警报

支持 SMBus ALERT 可选信号。仅从具备从机功能的设备可以通过 SMBA 引脚向主机发送信号请求通信。主机会处理中断并同时通过警报响应地址(0b0001 100)访问所有 SMBA 设备。只有拉低 SMBA 的设备才会应答警报响应地址。

当配置为从机设备(SMBHEN=0)时，通过设置 **I2C_CON1** 寄存器中的 ALERTEN 位为 1，可将 SMBA 引脚拉低。同时开启警报响应地址。

当配置为主机(SMBHEN=1)时，如果在 SMBA 引脚上检测到下降沿且 ALERTEN=1，则在 **I2C_RIF** 寄存器中的 ALERT 位被设置为 1。如果 **I2C_IER** 寄存器中的 ALERT 位为 1，则会产生中断。当 ALERTEN=0 时，即使外部 SMBA 引脚为低电平，ALERT 线也会被视为高电平。

如果不需要 SMBus ALERT 引脚，可设置 ALERTEN=0，则 SMBA 引脚可用作标准 GPIO。

数据包错误检查

SMBus 规范中引入了一种数据包错误检查机制，以提高可靠性和通信稳健性。通过在每次消息传输结束时附加数据包错误代码(PEC)来实现数据错误检查。通过在所有消息字节(包括地址和读/写位)上使用 CRC-8 多项式 $C(x)=x_8 + x^2 + x + 1$ 来计算 PEC。外设内建硬件 PEC 计算器，当接收到的字节与硬件计算的 PEC 不匹配时，会自动发送 NACK。

超时

该外设内建硬件定时器，以符合 SMBus 规范中定义的 3 个超时。

标记	参数	范围		单位
		最小	最大	
$T_{TIMEOUT}$	检测时钟低电平超时	25	35	ms
$T_{LOW:SEXT}$(1)	累积时钟低电平延长时间(从机设备)	-	25	ms
$T_{LOW:MEXT}$(2)	累积时钟低电平延长时间(主机设备)	-	10	ms

表 27-6 SMBus 超时规格

注:

1. $T_{LOW:SEXT}$ 是允许给定从机设备在一条消息中从初始 START 到 STOP 的延长时钟周期的累积时间。另一个从机设备或主机设备也可能延长时钟，导致组合时钟低电平延长时间大 $T_{LOW:SEXT}$。因此该参数是在从

机设备作为全速主机设备的唯一目标的情况下量测。

2. $T_{LOW:MEXT}$ 是允许主机设备在从 START 到 ACK，ACK 到 ACK 或 ACK 到 STOP 定义消息内每个字节的延长其时钟周期的累积时间。从机设备或另一个主机设备也可能延长时钟，导致组合时钟低电平时间大于给定字节上的 $T_{LOW:MEXT}$。因此该参数是在全速从设机备作为主机设备的唯一目标的情况下量测。

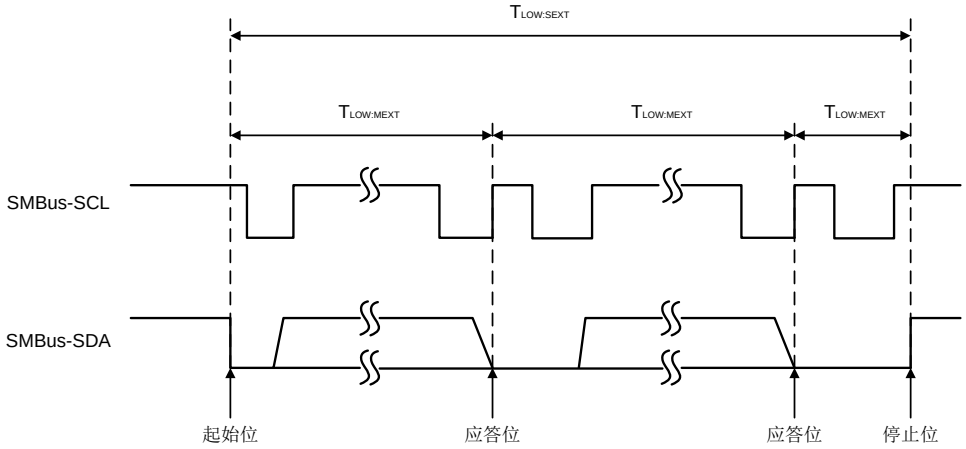


图 27-21　$T_{LOW:SEXT}$ 和 $T_{LOW:MEXT}$ 的超时间隔

总线空闲检测

该外设支持硬件总线空闲检测，如果总线检测到时钟和数据信号持续为高电平时间已经超过 T_{IDLE} 并且大于 $T_{HIGH.MAX}$，则主机设备可以认为总线处于空闲的状态。

该时序参数涵盖了主机已动态添加到总线，但可能还未检测到 SMBus-SCL 或 SMBus-SDA 线路上的状态转换的情况。在这种情况下，主机设备必须等待足够长的时间以确保当前没有进行传输。

26. 5. 9　SMBus初始化

除了 I2C 初始化之外，还必须进行一些其他特定的初始化以执行 SMBus 通信:

接收命令和数据应答位控制(从机模式)

SMBus 在从机接收必须能够对每个接收到的命令或数据响应 NACK。为了在从机模式下允许 ACK 控制，必须通过设置 **I2C_CON1** 寄存器中的 SBC 位为 1 来开启从机字节控制模式。更多详细信息请参见从机字节控制模式。

特定地址(从机模式)

外设提供三组 SMBus 的特定地址。

通过设置 **I2C_CON1** 寄存器中的 SMBDEN 位为 1 ，开启 SMBus 设备从机地址(0b1100001x)。

通过设置 **I2C_CON1** 寄存器中的 SMBHEN 位为 1，开启 SMBus 主控者从机地址(0b0001000x)。

通过设置 **I2C_CON1** 寄存器中的 ALERTEN 位为 1，开启警报响应地址(0b0001100x)。

数据封包错误检查

通过设置 **I2C_CON1** 寄存器中的 PECEN 位为 1 开启 PEC 计算。使用硬件字节计数器(NBYTES)管理 PEC 传输。必须先设置 PECEN 位，然后再开启 I2C。

因此在从机模式下连接 SMBus 时，必须设置 SBC 位。当设置 PECBYTE 位为 1 且 RELOAD 位清零时，在传输 NBYTES-1 数据字节后传输 PEC。如果设置了 RELOAD 位为 1 ，PECBYTE 则无效。

注:开启 I2C 时，不允许更改 PECEN 设置。

超时检测

通过将 **I2C_TIMEOUTR** 寄存器中的 TIMEOUTEN 位和 TEXTEN 位设置为 1 来开启超时检测。定时器必须如以下的方式配置，即在 SMBus 规范定义的最大时间之前检测到超时。

- $T_{TIMEOUT}$ 检查

  为了开启 $T_{TIMEOUT}$ 检查，必须设置 **I2C_TIMEOUTR** 寄存器中的 TIMEOUTA 位域为定时器的重载值，以检查 $T_{TIMEOUT}$ 参数。必须将 TIDLE 位设置为'0’才能检测 SCL 低电平超时。然后通过设置 **I2C_TIMEOUTR** 寄存器中的 TIMEOUTEN 位为 1 来开启定时器。如果 SCL 低电平持续时间大于(TIMEOUTA + 1)x 2048 x T_{I2CCLK}，则在 **I2C_RIF** 寄存器中的 TOUT 位将被设置为 1。

注:当设置 TIMEOUTEN 位为 1 时，不允许更改 TIMEOUTA 位域和 TIDLE 位的设置。

- $T_{LOW:SEXT}$ 和 $T_{LOW:MEXT}$ 检查

  根据外设是配置为主机还是从机，必须设置 **I2C_TIMEOUTR** 寄存器中的 TIMEOUTB 位域为定时器的重载值，以便检查从机的 $T_{LOW:SEXT}$ 和主机的 $T_{LOW:MEXT}$。由于标准仅指定最大值，因此用户可以设置这两个参数为相同的数值。通过设置 **I2C_TIMEOUTR** 寄存器中的 TEXTEN 位为 1 来开启定时器。如果 SCL 低电平累积时间大于(TIMEOUTB + 1)x 2048 x T_{I2CCLK}，则在 **I2C_RIF** 寄存器中的 TOUT 位将被设置为 1。

注:当设置 TEXTEN 位为 1 时，不允许更改设置 TIMEOUTB[11:0]位。

总线空闲检测

为了开启 T_{IDLE} 检查，必须设置 **I2C_TIMEOUTR** 寄存器中的 TIMEOUTA 位域为定时器的重载值，以获得 T_{IDLE} 参数。必须将设置 T_{IDLE} 位为 1 开启 SCL 和 SDA 高电平超时功能。

然后通过设置 **I2C_TIMEOUTR** 寄存器中的 TIMEOUTEN 位为 1 来开启定时器。如果 SCL 和 SDA 都为高电平且持续时间大于(TIMEOUTA + 1)x 4 x T_{I2CCLK}，则在 **I2C_RIF** 寄存器中的 TOUT 位将被设置为 1。

注:设置 TIMEOUTEN 时，不允许更改设置 TIMEOUTA 位和 TIDLE 位。

26. 5. 10 SMBus: I2C_TIMEOUTR寄存器配置示例

- 将 $T_{TIMEOUT}$ 的最大持续时间配置为 25 ms:

F_{I2CCLK}	TIMEOUT <11:0>	TIDLE	TIMEOUTEN	$T_{TIMEOUT}$

8 MHz	0x61	0	1	98 x 2048 x 125 ns =25 ms
16 MHz	0xC3	0	1	196 x 2048 x 62.5 ns =25 ms
48 MHz	0x249	0	1	586 x 2048 x 20.83 ns =25 ms

表 27-7　各种 I2CCLK 频率的 TIMEOUTA 设置示例(最大值 $T_{TIMEOUT}$ = 25 ms)

◆　将 $T_{LOW:SEXT}$ 和 $T_{LOW:MEXT}$ 的最大持续时间配置为 8 ms

F_{I2CCLK}	TIMEOUT<11:0>	TIMEOUTEN	$T_{LOW:EXT}$
8 MHz	0x1F	1	32 x 2048 x 125 ns =8 ms
16 MHz	0x3F	1	64 x 2048 x 62.5 ns =8 ms
48 MHz	0xBB	1	188 x 2048 x 20.83 ns =8 ms

表 27-8　各种 I2CCLK 频率的 TIMEOUTB 设置示例

◆　将 T_{IDLE} 的最大持续时间配置为 50 µs

F_{I2CCLK}	TIMEOUT<11:0>	TIDLE	TIMEOUTEN	T_{IDLE}
8 MHz	0x63	1	1	100 x 4 x 125 ns =50 µs
16 MHz	0xC7	1	1	200 x 4 x 62.5 ns =50 µs
48 MHz	0x257	1	1	500 x 4 x 20.83 ns =50 µs

表 27-9　各种 I2CCLK 频率的 TIMEOUTA 设置示例(最 T_{IDLE} =50 µs)

26. 5. 11 SMBus 从机模式

除了 I2C 从机模式传输管理(请参见 I2C 从机模式以外，还必须增加一些额外的软件流程来支持 SMBus。

26. 5. 11. 1　SMBus 从机发送

在 SMBus 从机发送模式下，必须将 **I2C_CON1** 寄存器中的 SBC 设置为 1，才能在 NBYTES 数据字节传输完成后进行 PEC 传输。当 **I2C_CON2** 寄存器中的 PECBYTE 位为 1 时，NBYTES 的字节数包含 PEC 传输。如果主机在完成 NBYTES-1 字节的数据传输后请求传输额外的字节，则自动发送 **I2C_PECR** 寄存器的数据。

注: 如果设置了 RELOAD 位为 1，PECBYTE 则无效。

26. 5. 11. 2　SMBus 从机接收

在 SMBus 从机接收模式下，必须将 **I2C_CON1** 寄存器中的 SBC 设置为 1，才能在 NBYTES 数据字节传输完成后进行 PEC 效验。要对每个字节进行 ACK 控制，必须选择重载模式(RELOAD=1)。更多详细信息请参见从机字节控制模式。

进行校验 PEC 字节前，必须将 RELOAD 位清零并将 PECBYTE 位设置为 1。在这种情况下，当接收到 NBYTES-1 数据字节后，下一个接收的字节将与内部 **I2C_PECR** 寄存器的数据进行比较。如果比较不匹配则自动应答 NACK，如果比较匹配则自动应答 ACK。接收到的 PEC 数据字节一样会复制到 **I2C_RXDATA** 寄存器中，并将 RXNE 设置为 1。当 PEC 不匹配时，**I2C_RIF** 寄存器中的 PECE 位将设置为 1，如果 **I2C_IER** 寄存器中的 PECE 位为 1，则会产生中断。如果无需软件控制 ACK，用户可以在设置 PECBYTE 为 1 时，同时将 NBYTES 配置为连续接收的字节数。当接收到 NBYTES-1 字节的数据后，会将下一个接收的字节视为 PEC 并进行校验。

注:

1. PEC 字节的应答位与 I2C_CON2 寄存器中 NACK 位的数值无关。
2. 如果设置了 RELOAD 位为 1，PECBYTE 则无效。

26. 5. 12 SMBus 主机模式

除了 I2C 主机模式传输管理(请参见 I2C 主机模式以外，还必须增加一些额外的软件流程来支持 SMBus。

26. 5. 12. 1 SMBus 主机发送

在 SMBus 主机发送模式下，要发送 PEC 数据字节，首先将 **I2C_CON2** 寄存器中的 PECBYTE 位设置为 1，同时配置 NBYTES，接着再将 START 位设置为 1。如果在 NBYTES=0x1 时，将 PECBYTE 位设置为 1，则将立即发送 **I2C_PECR** 寄存器的内容。在此模式下，开启自动结束模式(AUTOEND=1)时，当主机发送完 PEC 后将自动发送停止位。

26. 5. 12. 2 SMBus 主机接收

在 SMBus 主机接收模式下，开启自动结束模式(AUTOEND=1)时，当主机接收 PEC 后将自动发送停止位。将 **I2C_CON2** 寄存器中 START 位设置为 1 前，必须将 PECBYTE 位设置为 1 ，并且设置从机地址。在这种情况下，当接收到 NBYTES-1 数据字节后，将使用下一个接收数据字节与 **I2C_PECR** 寄存器的数据进行比较匹配，接着响应 NACK 以及停止位。

26. 5. 13 DMA请求

使用 DMA 发送

通过设置 **I2C_CON1** 寄存器中的 TXDMAEN 位为 1，可以开启 DMA 进行发送。只要 TXE 位被设置为 1，就会使用 DMA 外设配置的 SRAM 区域加载数据到 I2C_TXDATA 寄存器。

数据通过 DMA 发送。

使用 DMA 接收

通过设置 **I2C_CON1** 寄存器中的 RXDMAEN 位为 1，可以开启 DMA 进行接收。只要 RXNE 位被设置为 1，就会将读取 **I2C_RXDATA** 寄存器数据加载到使用 DMA 外设配置的 SRAM 区域。

数据通过 DMA 接收(包括 PEC)。

26. 5. 14 错误情况

当发生下列错误情况时，可能导致通信失败。

总线错误(BERR)

总线错误是指检测到起始位或停止位的时间点不是在 9 个 SCL 时钟脉冲的倍数之后。仅当 I2C 作为主机模式或从机模式进行数据传输(从机模式的地址阶段传输除外)时，才会将 **I2C_RIF** 寄存器中的 BERR 位设置为 1。如果 **I2C_IER** 寄存器中的 BERR 位为 1，则会产生中断。

在从机模式下检测到错误的起始位或重复起始位时，则 I2C 会进入地址识别状态，就像接收到正确的起始位一样

仲裁丢失(ARLO)

仲裁丢失是指当 I2C 外设发送高电平到引脚 SDA 上，但在 SCL 上升沿时却采样到引脚 SDA 为低电平。当检测到仲裁丢失时，**I2C_RIF** 寄存器中的 ARLO 位被设置为 1，如果 **I2C_IER** 寄存器中的 ARLO 位为 1，则会产生中断。

- 在主机模式下，在地址、数据和应答位传输阶段检测到仲裁丢失时，硬件会将 **I2C_CON2** 寄存器中的 START 位清除，并立即将 SDA 和 SCL 释放，同时外设由主机模式自动切换成从机模式。
- 在从机模式下，在数据和应答位传输阶段检测到仲裁丢失时，停止传输，并立即将 SDA 和 SCL 释放。

接收上溢或发送下溢错误(RXOV / TXUD)

在从机模式下，当 **I2C_CON1** 寄存器中的 NOSTRETCH 位为 1 时，在以下情况会检测到接收上溢或发送下溢的错误:

- 在从机接收模式下，当接收到一个新的数据字节时，但先前在 **I2C_RXDATA** 寄存器的数据还未被读出。新接收的字节将会丢失，并且自动应答 NACK。
- 在从机发送模式下，当在发送一个新数据字节时，但 **I2C_TXDATA** 寄存器尚未写入新的数据，并且 **I2C_STAT**.TXE=1 时，将发送 0xFF。

当检测到接收上溢或发送下溢错误时，在 **I2C_STAT** 寄存器中的 TXUD 位与 RXOV 位被设置为 1，如果 **I2C_IER** 寄存器中的 TXUD 位或 RXOV 位为 1，则会产生中断。

数据封包错误检查错误(PECE)

当接收到的 PEC 数据字节与 **I2C_PECR** 寄存器的数据比较不匹配时，会将 **I2C_RIF** 寄存器中的 PECE 位设置为 1。如果 **I2C_IER** 寄存器中的 PECE 位为 1，则会产生中断。并且自动应答 NACK。

超时错误 (TOUT)

在下列情况下均会出现超时错误：

- TIDLE=0 且 SCL 的低电平持续时间达到 TIMEOUTA 位域所定义的时间：这用于检测 SMBus 超时。
- TIDLE=1 且 SDA 和 SCL 的高电平持续时间达到 TIMEOUTA 位域所定义的时间：这用于检测总线空闲情况。
- 主机累积时钟低电平延长时间达到了 TIMEOUTB 位域所定义的时间(SMBus $T_{LOW:MEXT}$)
- 从机累积时钟低电平延长时间达到了 TIMEOUTB 位域所定义的时间(SMBus $T_{LOW:SEXT}$)

当在主机模式下检测到超时后，将自动发送停止位。

当在从机模式下检测到超时后，将自动释放 SDA 和 SCL 。

检测到超时错误时，**I2C_RIF**>寄存器中的 TOUT 位将被设置为 1，如果 **I2C_IER** 寄存器中的

TOUT 位为 1，则会产生中断。

SMBus 警报(ALERT)

当 I2C 配置为主机(SMBHEN=1)，并开启警报引脚检测(ALERTEN=1)后，在引脚 SMBA 上检测到下降沿时，**I2C_RIF**>寄存器中的 ALERT 位将被设置为 1，如果 **I2C_IER** 寄存器中的 ALERT 位为 1，则会产生中断。

26. 5. 15 I2C中断

I2C 的中断由六个寄存器控制。

- 中断控制(IER，IDR，IVS) I2C 中断开启寄存器(I2C_IER)通过设置为 1 来开启中断功能。同样，I2C 中断关闭寄存器(**I2C_IDR**)通过设置为 1 来关闭中断功能。IER 和 IDR 寄存器只能写入，上述寄存器中的结果可由中断功能有效状态寄存器(**I2C_IVS**)表示。IVS 寄存器只能读取，使用'1’或'0’来表示中断功能是否有效。
- 原始中断状态寄存器(RIF)
  I2C 原始中断状态寄存器(**I2C_RIF**) I2C 原始中断状态寄存器(**I2C_RIF**)是一个只能读取的寄存器，用于读取外设的中断状态。该寄存器中的位表示 I2C 中断的真实状态。当观察到以下条件时，I2C 可以产生中断:
  - ◇ ALERT : SMBus 警报
  - ◇ TOUT : 超时
  - ◇ PECE : PEC 错误
  - ◇ ARLO : 仲裁丢失
  - ◇ BERR : 总线错误
  - ◇ STOP : 停止位检测
  - ◇ NACK : 接收否定应答
  - ◇ ADDR : 地址匹配
  - ◇ TCR : 传输完成等待重载
  - ◇ TC : 传输完成
  - ◇ RXUD : 接收数据寄存器下溢
  - ◇ RXOV : 接收数据寄存器上溢
  - ◇ RXNE : 接收数据寄存器不为空
  - ◇ TXUD : 发送数据寄存器下溢
  - ◇ TXOV : 发送数据寄存器上溢
  - ◇ TXE : 发送数据寄存器为空
- 中断标志位状态寄存器(IFM) I2C 中断标志位状态寄存器(**I2C_IFM**)用于读取外设的中断标志位状态，表示是哪个中断。IFM 中的每个位是 IVS 和 RIF 中各个位的逻辑 AND。
- 中断清除(ICR) 向该寄存器中的位设置为 1，可以清除相应的中断。

26. 5. 16 调试模式

当微控制器进入调试模式(Cortex™-M0 核停止运行)，根据 SYSCFG 章节中 SYSCFG_CFG 寄存器配置，选择 SMBus 超时计数器继续正常计数或停止计数。

26. 6 特殊功能寄存器

26. 6. 1 寄存器列表

I2C 寄存器列表			
名称	偏移地址	类型	描述
I2C_CON1	0000_H	R/W	I2C 控制寄存器 1
I2C_CON2	0004_H	R/W	I2C 控制寄存器 2
I2C_ADDR1	0008_H	R/W	I2C 本机地址寄存器 1
I2C_ADDR2	$000C_H$	R/W	I2C 本机地址寄存器 2
I2C_TIMINGR	0010_H	R/W	I2C 时序寄存器
I2C_TIMEOUTR	0014_H	R/W	I2C 超时寄存器
I2C_STAT	0018_H	R/T_W1	I2C 状态寄存器
I2C_PECR	0020_H	R	I2C PEC 寄存器
I2C_RXDATA	0024_H	R	I2C 接收数据寄存器
I2C_TXDATA	0028_H	R/W	I2C 发送数据寄存器
I2C_IER	$002C_H$	W1	I2C 中断开启寄存器
I2C_IDR	0030_H	W1	I2C 中断关闭寄存器
I2C_IVS	0034_H	R	I2C 中断功能有效状态寄存器
I2C_RIF	0038_H	R	I2C 原始中断状态寄存器
I2C_IFM	$003C_H$	R	I2C 中断标志位状态寄存器
I2C_ICR	0040_H	C_W1	I2C 中断清除寄存器

26. 6. 2 寄存器描述

I2C 寄存器必须按字(32 位)进行访问。

26. 6. 2. 1 I2C控制寄存器 1 (I2C_CON1)

I2C 控制寄存器 1 (I2C_CON1)																															
偏移地址:0x00																															
复位值:0x0000 0000																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NBYTESH<15:8>								PECEN	ALERTEN	SMBDEN	SMBHEN	GCEN	—	NOSTRETCH	SBC	RXDMAEN	TXDMAEN	—	—	DNF<3:0>				—	—	—	—	—	—	—	PE

NBYTESH	Bit 31-24	R/W	**字节数高 8 位** 参照CON2.NBYTESL描述
PECEN	Bit 23	R/W	**PEC 开启** 0: 关闭 PEC 计算 1: 开启 PEC 计算 注: 如果不支持 SMBus 功能，则该位保留并由硬件强制为 0。
ALERTEN	Bit 22	R/W	**SMBus 警报开启** 从机模式(SMBHEN=0)： 0: 释放 SMBA 引脚为高电平并且关闭警报响应地址标头: 0b0001100x 回应 NACK。 1: 将 SMBA 引脚驱动为低电平并且开启警报响应地址标头: 0b0001100x 回应 ACK。 主机模式(SMBHEN=1)： 0: 不支持 SMBus 警报引脚(SMBA) 1: 支持 SMBus 警报引脚(SMBA) 注: 当 ALERTEN=0 时，SMBA 引脚可用作标准 GPIO。如果不支持 SMBus 功能，则该位保留并由硬件强制为 0。
SMBDEN	Bit 21	R/W	**SMBus 设备默认地址开启** 0: 关闭设备默认地址。对地址 0b1100001x 回应 NACK 1: 开启设备默认地址。对地址 0b1100001x 回应 ACK 注: 如果不支持 SMBus 功能，则该位保留并由硬件强制为 0。

SMBHEN	Bit 20	R/W	**SMBus 主机地址开启** 0: 关闭主机地址。对地址 0b0001000x 回应 NACK 1: 开启主机地址。对地址 0b0001000x 回应 ACK 注: 如果不支持 SMBus 功能，则该位保留并由硬件强制为 0。
GCEN	Bit 19	R/W	**广播呼叫开启** 0: 关闭广播呼叫。对地址 0b00000000 回应 NACK 1: 开启广播呼叫。对地址 0b00000000 回应 ACK
—	Bit 18	—	—
NOSTRETCH	Bit 17	R/W	**时钟延长关闭** 该位用于关闭从机模式下的时钟延长。在主机模式下必须保持清零状态。 0: 开启时钟延长 1: 关闭时钟延长 注: 只有在关闭 I2C(PE=0)时才能对该位进行设置。
SBC	Bit 16	R/W	**从机字节控制** 该位用于在从机模式下开启硬件字节控制。 0: 关闭从机模式下的字节控制 1: 开启从机模式下的字节控制
RXDMAEN	Bit 15	R/W	**DMA 接收请求开启** 0: 关闭 DMA 模式进行接收 1: 开启 DMA 模式进行接收
TXDMAEN	Bit 14	R/W	**DMA 发送请求开启** 0: 关闭 DMA 模式进行发送 1: 开启 DMA 模式进行发送
—	Bit 13-12	—	—
DNF	Bit 11-8	R/W	**数字噪声滤波器** 该位用于配置 SDA 和 SCL 输入端的数字噪声滤波器。数字滤波器可过滤长度高达 DNF[3:0] x T_{I2CCLK} 的宽度 0000: 关闭数字滤波器 0001: 开启数字滤波器，滤波宽度高达 1 T_{I2CCLK} …

			1111: 开启数字滤波器，滤波宽度高达 15 T_{I2CCLK} 注: 只有在关闭 I2C (PE=0)时才能对该位域进行设置。
—	Bit 7-1	—	—
PE	Bit 0	R/W	**I2C 开启** 0: I2C 关闭 1: I2C 开启 注: 当 PE=0 时，I2C 的总线信号(SCL 与 SDA)将被释放。内部状态机和状态位将恢复为复位值。软件清零时，至少保持 3 个 APB 时钟周期后才能在设置为 1。

26. 6. 2. 2 I2C控制寄存器 2 (I2C_CON2)

I2C 控制寄存器 2 (I2C_CON2)

偏移地址:0x04

复位值:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
—	—	—	—	—	PECBYTE	AUTOEND	RE LOAD	NBYTES<7:0>								NACK	STOP	START	HEAD10R	ADD10	RD_WRN	SADD<9:0>										

—	Bits 31-27	—	—
PECBYTE	Bit 26	R/W1	**数据包错误校验字节** 该位由软件设置 1，当 PEC 传输完成时，以及在接收到停止位或地址匹配事件时，或当 PE=0 时由硬件清零。 0: 不传输 PEC 1: 请求 PEC 发送或接收 注: 1. 向该位写入 0 无效。 2. RELOAD 设置为 1 时该位无效。 3. 当 SBC 设置为 0 时，该位在从机模式下无效。 4. 如果不支持 SMBus 功能，则该位保留并由硬件强制为 0。
AUTOEND	Bit 25	R/W	**自动结束模式(主机模式)** 该位由软件设置 1 和清零。

			0: 软件结束模式: 当 NBYTES 数据传输完成时，TC 标志由硬件设置 1 并将 SCL 拉低直到相应软件操作结束 1: 自动结束模式:当 NBYTES 数据传输完成时，将自动发送停止位 注: 该位在从机模式或 RELOAD 位设置为 1 时无效。
RELOAD	Bit 24	R/W	**NBYTES 重载模式** 该位由软件设置 1 和清零。 0: 传输 NBYTES 笔数据之后即完成传输 1: 传输 NBYTES 笔数据之后未完成传输(将重载 NBYTES)。当 NBYTES 数据传输完成时，TCR 标志由硬件设置 1 并将 SCL 拉低直到相应软件操作结束
NBYTESL	Bit 23-16	R/W	**字节数低 8 位** 此位域与 CON1.NBYTESH 组合成 16 位，NBYTES={CON1.NBYTESH[7:0],CON2.NBYTESL[7:0]} 在此设置要发送或接收的字节数。在 SBC=0 的从机模式下，该位域无效。 注: 当 START 位设置为 1 时，不允许更改该位域。
NACK	Bit 15	R/W1	**NACK 产生(从机模式)** 该位由软件设置 1，并在发送 NACK 时，以及在接收到停止位或地址匹配事件时，或当 PE=0 时由硬件清零。 0: 在当前接收的字节后发送 ACK 1: 在当前接收的字节后发送 NACK 注: 1. 向该位写入 0 无效。 2. 该位仅用于从机模式。在主机接收模式下，无论 NACK 位值为何，在最后一个字节之后都自动产生 NACK。 3. 当从机接收模式下且设置 NOSTRETCH=1 时，当发生接收数据寄存器上溢，无论 NACK 位值如何，都会自动产生 NACK。 4. 当开启硬件 PEC 检查(PECBYTE=1)时，

			PEC 应答值与该位无关。
STOP	Bit 14	R/W1	**停止位 (STOP) 产生(主机模式)** 该位由软件设置 1，当检测到停止位事件或 I2C 关闭(PE=0)时由硬件清零。 在主机模式下： 0: 不产生停止位(STOP) 1: 当前字节传输后产生停止位(STOP) 注: 向该位写入 0 无效。
START	Bit 13	R/W1	**起始位(START)产生** 该位由软件设置 1，并在发送起始位后跟随地址序列之后、发生仲裁丢失、检测超时错误或 PE=0 时由硬件清零。 该位也可以对 I2C_ICR 寄存器中 ADDR 位设置 1 时实现软件清零。 0: 不产生起始位(START) 1: 产生重复起始位(RESTART)/起始位(START)： 如果 I2C 处于主机模式且 AUTOEND=0，当 NBYTES 传输结束后将该位设置为 1，在 RELOAD=0 的情况下产生重复起始位。 否则将该位设置为 1 时，I2C 会在总线空闲后立即产生起始位。 注: 1. 向该位写入 0 无效。 2. 即使总线忙或 I2C 处于从机模式，也可以设置该位。 3. 在 10 位地址模式下，如果在地址的第一部分接收器收到 NACK，则 START 位不会被硬件清零，主机将重新发送地址序列，除非 START 位被软件清零。
HEAD10R	Bit 12	R/W	**读取传输时，仅发送 10 位地址的前 7 位地址标头(主机接收模式)** 0: 主机发送完整的 10 位从机地址读取序列：起始位+包含写入方向的 2 字节 10 位地址+重复起始位+包含读取方向的 10 位地址的前 7 位 1: 主机仅发送 10 位地址的前 7 位，然后发送读取方向

			注: 当 START 位设置 1 时，不允许更改该位。
ADD10	Bit 11	R/W	**10 位地址模式(主机模式)** 0: 主机使用 7 位地址模式 1: 主机使用 10 位地址模式 注: 当 START 位设置为 1 时，不允许更改该位。
RD_WRN	Bit 10	R/W	**传输方向(主机模式)** 0: 主机请求写入传输 1: 主机请求读取传输 注: 当 START 位设置 1 时，不允许更改该位。
SADD	Bit 9-0	R/W	**从机地址(主机模式)** 在 7 位地址模式(ADD10=0): 仅使用 SADD[7:1]位写入待发送的 7 位从机地址。在此模式下 SADD[9:8]与 SADD[0]位为无效。 在 10 位地址模式(ADD10=1): 该位域写入待发送的 10 位从机地址。 注: 当 START 位设置为 1 时，不允许更改该位域。

26. 6. 2. 3 I2C本机地址寄存器 1 (I2C_ADDR1)

I2C 本机地址寄存器 1 (I2C_ADDR1)																												
偏移地址:0x08																												
复位值:0x0000 0000																												
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																OA1EN					OA1MODE	OA1<9:0>									

—	Bits 31-16	—	—
OA1EN	Bit 15	R/W	**本机地址 1 开启** 0:关闭本机地址 1。对接收到的从机地址 OA1 响应 NACK 1:开启本机地址 1。对接收到的从机地址 OA1 响应 ACK
—	Bit 14-11	—	—
OA1MODE	Bit 10	R/W	**本机地址 1，10 位地址模式开启** 0:本机地址 1 使用 7 位地址模式。 1:本机地址 1 使用 10 位地址模式。 注:当 OA1EN=0 时才能写入该位。
OA1	Bit 9-0	R/W	**本机地址 1** 在 7 位地址模式(OA1MODE=0): 仅使用 OA1[7:1]位配置本机地址 1 的 7 位地址。 在此模式下 OA1[9:8] 与 OA1[0]位为无效。 在 10 位地址模式(OA1MODE=1): 该位域配置本机地址 1 的 10 位地址。 注: 当 OA1EN=0 时才能写入该位域。

26. 6. 2. 4 I2C本机地址寄存器 2 (I2C_ADDR2)

I2C 本机地址寄存器 2 (I2C_ADDR2)																												
偏移地址:0x0C																												
复位值:0x0000 0000																												
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																OA2EN					OA2MSK<2:0>			OA2<6:0>						

—	Bits 31-16	—	—
OA2EN	Bit 15	R/W	**本机地址 2 开启** 0:关闭本机地址 2。对接收到的从机地址 OA2 响应 NACK。 1:开启本机地址 2。对接收到的从机地址 OA2 响应 ACK。
—	Bit 14-11	—	—
OA2MSK	Bit 10-8	R/W	**本机地址 2 屏蔽** 000:没有屏蔽 001:OA2 [1]被屏蔽。仅比较 OA2 [7:2]。 010:OA2 [2:1]被屏蔽。仅比较 OA2 [7:3]。 011:OA2 [3:1]被屏蔽。仅比较 OA2 [7:4]。 100:OA2 [4:1]被屏蔽。仅比较 OA2 [7:5]。 101:OA2 [5:1]被屏蔽。仅比较 OA2 [7:6]。 110:OA2 [6:1]被屏蔽。仅比较 OA2 [7]。 111:OA2 [7:1]被屏蔽。不进行比较，对所有接收到的 7 位地址(保留地址除外)响应 ACK 注: 1. 当 OA2EN=0 时才能写入该位域。 2. 当 OA2MSK 不等于 0 时，I2C 即使比较匹配，也不会对保留地址(0b0000xxx 和 0b1111xxx)回应 ACK。
OA2	Bit 7-1	R/W	**本机地址 2** 7 位地址模式: 7 位地址 注:当 OA2EN=0 时才能写入该位域。
—	Bit 0	—	—

26. 6. 2. 5 I2C时序寄存器 (I2C_TIMINGR)

I2C 时序寄存器 (I2C_TIMINGR)																															
偏移地址:0x10																															
复位值:0x0000 0000																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PRESC<3:0>				—	—	—	—	SCLDEL<3:0>				SDADEL<3:0>				SCLH<7:0>								SCLL<7:0>							

PRESC	Bit 31-28	R/W	**时钟预分频器** 该位域用于对 I2CCLK 进行预分频，以产生用于数据建立和保持计数器以及 SCL 高电平和低电平计数器的时钟周期 T_{PRESC}。 T_{PRESC} =(PRESC + 1)x T_{I2CCLK}
—	Bit 27-24	—	—
SCLDEL	Bit 23-20	R/W	**数据建立时间** 该位域用于在 SDA 边沿和 SCL 上升沿之间产生延迟 T_{SCLDEL}。在主机和从机模式下，当 NOSTRETCH=0 时，SCL 低电平时间将在 T_{SCLDEL} 期间延长。 T_{SCLDEL} =(SCLDEL + 1)x T_{PRESC} 注:SCLDEL 用于产生 $T_{SU:DAT}$ 时序。
SDADEL	Bit 19-16	R/W	**数据保持时间** 该位域用于在 SCL 下降沿和 SDA 边沿之间产生延迟 T_{SDADEL}。在主机和从机模式下，当 NOSTRETCH=0 时，SCL 低电平时间将在 T_{SDADEL} 期间延长。 T_{SDADEL} =SDADEL x T_{PRESC} 注: SDADEL 用于产生 $T_{HD:DAT}$ 时序。
SCLH	Bit 15-8	R/W	**SCL 高电平周期(主机模式)** 该位域用于在主机模式下产生 SCL 高电平周期。 T_{SCLH} =(SCLH+1) x T_{PRESC} 注: SCLH 还用于产生 $T_{SU:STO}$ 和 $T_{HD:STA}$ 时序。
SCLL	Bit 7-0	R/W	**SCL 低电平周期(主机模式)** 该位域用于在主机模式下产生 SCL 低电平周期。

			T_{SCLL}=(SCLL+1) x T_{PRESC} 注: SCLL 还用于产生 T_{BUF} 和 $T_{SU:STA}$ 时序。

注: 该寄存器只有在 I2C 关闭时(PE=0)才能进行设置。

26. 6. 2. 6 I2C超时寄存器 (I2Cx_TIMEOUTR)

I2C 超时寄存器(I2Cx_TIMEOUTR)

偏移地址:0x14

复位值:0x0000 0000

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

TEXTEN	Bit 31	R/W	**时钟延长超时开启** 0: 关闭时钟延长超时检测 1: 开启时钟延长超时检测。当 I2C 接口执行 SCL 延长的累积时间超过 $T_{LOW:EXT}$ 时，将会检测到超时错误(TOUT=1)
—	Bit 30-28	—	—
TIMEOUTB	Bit 27-16	R/W	**总线超时 B** 该位域用于配置累积时钟延长超时： 在主机模式下，检测到主机累积时钟低电平延长时间($T_{LOW:MEXT}$)。 在从机模式下，检测到从机累积时钟低电平延长时间($T_{LOW:SEXT}$)。 $T_{LOW:EXT}$ = (TIMEOUTB + 1)x 2048 x T_{I2CCLK} 注: 当 TEXTEN=0 时才能写入该位域。
TIMEOUTEN	Bit 15	R/W	**时钟超时开启** 0: 关闭 SCL 超时检测 1: 开启 SCL 超时检测。当 SCL 为低电平超过 $T_{TIMEOUT}$(TIDLE=0) 或高电平超过 T_{IDLE}(TIDLE=1) 时，将检测到超时错误(TOUT=1)
—	Bit 14-13	—	—
TIDLE	Bit 12	R/W	**空闲时钟超时检测** 0: TIMEOUTA 用于检测 SCL 低电平超时

			1: TIMEOUTA 用于检测 SCL 和 SDA 高电平超时(总线空闲状态) 注: 当 TIMEOUTEN=0 时才能写入该位。
TIMEOUTA	Bit 11-0	R/W	**总线超时 A** 该位域用于配置: - 当 TIDLE=0 时,SCL 低电平超时条件 $T_{TIMEOUT}$ $T_{TIMEOUT}$ =(TIMEOUTA + 1)x 2048 x T_{I2CCLK} - 当 TIDLE=1 时，总线空闲状态(SCL 和 SDA 均为高电平) T_{IDLE} =(TIMEOUTA + 1)x 4 x T_{I2CCLK} 注: 当 TIMEOUTEN=0 时才能写入该位域。

26. 6. 2. 7 I2C状态寄存器 (I2C_STAT)

I2C 状态寄存器(I2C_STAT)

偏移地址:0x18

复位值:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
—	—	—	—	—	—	—	—	ADDCODE<6:0>							DIR	BUSY	—	—	—	TCR	TC	—	RXUD	RXOV	—	RXNE	—	TXUD	TXOV	—	TXE

—	Bits 31-24	—	—
ADDCODE	Bit 23-17	R	**地址匹配代码(从机模式)** 当发生地址匹配事件(ADDR=1)时，该位域更新为接收到的地址。 在 10 位地址的情况下，ADDCODE 提供 10 位地址的标头与地址的 2 个 MSB。
DIR	Bit 16	R	**传输方向(从机模式)** 当发生地址匹配事件(ADDR=1)时更新此标志。 0: 写入传输，从机进入接收器模式 1: 读取传输，从机进入发送器模式
BUSY	Bit 15	R	**总线繁忙** 该标志表示总线上正在进行通信。检测到起始位(START 条件)时由硬件设置为 1。当检测到停止位(STOP 条件)或 PE=0 时，该标志由硬件清零。
—	Bit 14-12	—	—

TCR	Bit 11	R	**传输完成等待重载** 当 RELOAD=1 且已将 NBYTES 笔数据传输完成时，该标志由硬件设置 1。当设置 I2C_CON2.NBYTES 且 NBYTES 笔数不为零时，该标志由硬件清零。 注: 1. 当 PE=0 时，该位由硬件清零。 2. 该标志仅用于主机模式，或者用于从机模式且设置 SBC 位为 1 时。
TC	Bit 10	R	**传输完成(主机模式)** 当 RELOAD=0、AUTOEND=0 且已将 NBYTES 笔数据传输完成时，该标志由硬件设置为 1。当设置 START 位或 STOP 位为 1 时，由硬件清零。 注: 当 PE=0 时，该位由硬件清零。
—	Bit 9	—	—
RXUD	Bit 8	R/C_R	**接收数据寄存器下溢** 0: 接收数据寄存器未发生下溢 1: 接收数据寄存器发生下溢 注:读取 STAT 寄存器清除此位。
RXOV	Bit 7	R/C_R	**接收数据寄存器上溢** 0: 接收数据寄存器未发生上溢 1: 接收数据寄存器发生上溢 注: 读取 STAT 寄存器清除此位。
—	Bit 6	—	—
RXNE	Bit 5	R	**接收数据寄存器不为空** 0: 接收数据寄存器为空 1: 接收数据寄存器不为空
—	Bit 4	—	—
TXUD	Bit 3	R/C_R	**发送数据寄存器下溢** 0: 发送数据寄存器未发生下溢 1: 发送数据寄存器发生下溢 注: 读取 STAT 寄存器清除此位。
TXOV	Bit 2	R/C_R	**发送数据寄存器上溢** 0: 发送数据寄存器未发生上溢 1: 发送数据寄存器发生上溢 注:读取 STAT 寄存器清除此位。
—	Bit 1	—	—

TXE	Bit 0	R/T_W1	**发送数据寄存器空** 0: 发送数据寄存器不为空 1: 发送数据寄存器为空 注: 1. 该位可由软件写入 1，以刷新发送数据寄存器 I2C_TXDATA。 2. 当 PEC=0 时，该位由硬件设置为 1。

26. 6. 2. 8 I2C PEC寄存器 (I2C_PECR)

I2CPEC 寄存器 (I2C_PECR)

偏移地址:0x20

复位值:0x0000 0000

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

—	Bits 31-8	—	—
PEC	Bits 7-0	R	**数据包错误校验寄存器** 当 PECEN=1 时，该位域包含内部 PEC。 当 PEC=0 时，PEC 位域由硬件清零。

26. 6. 2. 9 I2C接收器数据寄存器 (I2C_RXDATA)

I2C 接收器数据寄存器 (I2C_RXDATA)

偏移地址:0x24

复位值:0x0000 0000

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

—	Bits 31-8	—	—
RXDATA	Bits 7-0	R	**8 位接收数据** 从 I2C 总线接收的数据字节。

26. 6. 2. 10 I2C发送器数据寄存器 (I2C_TXDATA)

I2C 发送器数据寄存器(I2C_TXDATA)																													
偏移地址:0x28																													
复位值:0x0000 0000																													
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	TXDATA<7:0>							

—	Bits 31-8	—	—
TXDATA	Bits 7-0	W	**8 位发送数据** 待发送到 I2C 总线的数据字节。

26. 6. 2. 11 I2C中断开启寄存器 (I2C_IER)

I2C 中断开启寄存器(I2C_IER)																													
偏移地址:0x2C																													
复位值:0x0000 0000																													
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
—	—	—	—	—	—	—	—	—	—	—	ALERT	TOUT	PECE	ARLO	BERR	—	STOP	NACK	ADDR	TCR	TC	—	RXUD	RXOV	—	RXNE	—	TXUD	TXOV	—	TXE

—	Bits 31-21	—	—
ALERT	Bit 20	W1	**SMBus 警报中断开启** 0:写入 0 无效 1:开启 SMBus 警报中断
TOUT	Bit 19	W1	**超时中断开启** 0:写入 0 无效 1:开启超时中断
PECE	Bit 18	W1	**PEC 错误中断开启** 0:写入 0 无效 1:开启 PEC 错误中断
ARLO	Bit 17	W1	**仲裁丢失中断开启** 0:写入 0 无效 1:开启仲裁丢失中断
BERR	Bit 16	W1	**总线错误中断开启**

			0:写入 0 无效 1:开启总线错误中断
—	Bit 15	—	—
STOP	Bit 14	W1	**停止位检测中断开启** 0:写入 0 无效 1:开启停止位检测中断
NACK	Bit 13	W1	**接收否定应答中断开启** 0:写入 0 无效 1:开启接收否定应答中断
ADDR	Bit 12	W1	**地址匹配中断开启** 0:写入 0 无效 1:开启地址匹配中断
TCR	Bit 11	W1	**传输完成等待重载中断开启** 0:写入 0 无效 1:开启传输完成等待重载中断
TC	Bit 10	W1	**传输完成中断开启** 0:写入 0 无效 1:开启传输完成中断
—	Bit 9	—	—
RXUD	Bit 8	W1	**接收数据寄存器下溢中断开启** 0:写入 0 无效 1:开启接收数据寄存器下溢中断
RXOV	Bit 7	W1	**接收数据寄存器上溢中断开启** 0:写入 0 无效 1:开启接收数据寄存器上溢中断
—	Bit 6	—	—
RXNE	Bit 5	W1	**接收数据寄存器不为空中断开启** 0:写入0无效 1:开启接收数据寄存器不为空中断
—	Bit 4	—	—
TXUD	Bit 3	W1	**发送数据寄存器下溢中断开启** 0:写入 0 无效 1:开启发送数据寄存器下溢中断
TXOV	Bit 2	W1	**发送数据寄存器上溢中断开启** 0:写入 0 无效 1:开启发送数据寄存器上溢中断
—	Bit 1	—	—
TXE	Bit 0	W1	**发送数据寄存器为空中断开启**

			0:写入 0 无效 1:开启发送数据寄存器为空中断

26. 6. 2. 12 I2C中断关闭寄存器 (I2C_IDR)

I2C 中断关闭寄存器(I2C_IDR)																												
偏移地址:0x30																												
复位值:0x0000 0000																												
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
—	—	—	—	—	—	—	—	—	—	—	ALERT	TOUT	PECE	ARLO	BERR	—	STOP	NACK	ADDR	TCR	TC	—	RXUD	RXOV	—	RXNE	—	TXUD	TXOV	—	TXE

—	Bit 31-21	—	—
ALERT	Bit 20	W1	**SMBus 警报中断关闭** 0:写入 0 无效 1:关闭 SMBus 警报中断
TOUT	Bit 19	W1	**超时中断关闭** 0:写入 0 无效 1:关闭超时中断
PECE	Bit 18	W1	**PEC 错误中断关闭** 0:写入 0 无效 1:关闭 PEC 错误中断
ARLO	Bit 17	W1	**仲裁丢失中断关闭** 0:写入 0 无效 1:关闭仲裁丢失中断
BERR	Bit 16	W1	**总线错误中断关闭** 0:写入 0 无效 1:关闭总线错误中断
—	Bit 15	—	—
STOP	Bit 14	W1	**停止位检测中断关闭** 0:写入 0 无效 1:关闭停止位检测中断
NACK	Bit 13	W1	**接收否定应答中断关闭** 0:写入 0 无效 1:关闭接收否定应答中断
ADDR	Bit 12	W1	**地址匹配中断关闭** 0:写入 0 无效

			1:关闭地址匹配中断
TCR	Bit 11	W1	**传输完成等待重载中断关闭** 0:写入 0 无效 1:关闭传输完成等待重载中断
TC	Bit 10	W1	**传输完成中断关闭** 0:写入 0 无效 1:关闭传输完成中断
—	Bit 9	—	—
RXUD	Bit 8	W1	**接收数据寄存器下溢中断关闭** 0:写入 0 无效 1:关闭接收数据寄存器下溢中断
RXOV	Bit 7	W1	**接收数据寄存器上溢中断关闭** 0:写入 0 无效 1:关闭接收数据寄存器上溢中断
—	Bit 6	—	—
RXNE	Bit 5	W1	**接收数据寄存器不为空中断关闭** 0:写入0无效 1:关闭接收数据寄存器不为空中断
—	Bit 4	—	—
TXUD	Bit 3	W1	**发送数据寄存器下溢中断关闭** 0:写入 0 无效 1:关闭发送数据寄存器下溢中断
TXOV	Bit 2	W1	**发送数据寄存器上溢中断关闭** 0:写入 0 无效 1:关闭发送数据寄存器上溢中断
—	Bit 1	—	—
TXE	Bit 0	W1	**发送数据寄存器为空中断关闭** 0:写入 0 无效 1:关闭发送数据寄存器为空中断

26. 6. 2. 13 I2C中断功能有效状态寄存器 (I2C_IVS)

I2C 中断功能有效状态寄存器 (I2C_IVS)																														
偏移地址:0x34																														
复位值:0x0000 0000																														
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
—	—	—	—	—	—	—	—	—	—	—	ALERT	TOUT	PECE	ARLO	BERR	—	STOP	NACK	ADDR	TCR	TC	—	RXUD	RXOV	—	RXNE	—	TXUD	TXOV	—	TXE

—	Bit 31-21	—	—
ALERT	Bit 20	R	**SMBus 警报中断功能状态** 0:中断功能处于关闭状态 1:中断功能处于开启状态
TOUT	Bit 19	R	**超时中断功能状态** 0:中断功能处于关闭状态 1:中断功能处于开启状态
PECE	Bit 18	R	**PEC 错误中断功能状态** 0:中断功能处于关闭状态 1:中断功能处于开启状态
ARLO	Bit 17	R	**仲裁丢失中断功能状态** 0:中断功能处于关闭状态 1:中断功能处于开启状态
BERR	Bit 16	R	**总线错误中断功能状态** 0:中断功能处于关闭状态 1:中断功能处于开启状态
—	Bit 15	—	—
STOP	Bit 14	R	**停止位检测中断功能状态** 0:中断功能处于关闭状态 1:中断功能处于开启状态
NACK	Bit 13	R	**接收否定应答中断功能状态** 0:中断功能处于关闭状态 1:中断功能处于开启状态
ADDR	Bit 12	R	**地址匹配中断功能状态** 0:中断功能处于关闭状态 1:中断功能处于开启状态
TCR	Bit 11	R	**传输完成等待重载中断功能状态** 0:中断功能处于关闭状态

			1:中断功能处于开启状态
TC	Bit 10	R	**传输完成中断功能状态** 0:中断功能处于关闭状态 1:中断功能处于开启状态
—	Bit 9	—	—
RXUD	Bit 8	R	**接收数据寄存器下溢中断功能状态** 0:中断功能处于关闭状态 1:中断功能处于开启状态
RXOV	Bit 7	R	**接收数据寄存器上溢中断功能状态** 0:中断功能处于关闭状态 1:中断功能处于开启状态
—	Bit 6	—	—
RXNE	Bit 5	R	**接收数据寄存器不为空中中断功能状态** 0:中断功能处于关闭状态 1:中断功能处于开启状态
—	Bit 4	—	—
TXUD	Bit 3	R	**发送数据寄存器下溢中断功能状态** 0:中断功能处于关闭状态 1:中断功能处于开启状态
TXOV	Bit 2	R	**发送数据寄存器上溢中断功能状态** 0:中断功能处于关闭状态 1:中断功能处于开启状态
—	Bit 1	—	—
TXE	Bit 0	R	**发送数据寄存器为空中断功能状态** 0:中断功能处于关闭状态 1:中断功能处于开启状态

I2C_IVS 寄存器，是实时反映系统配置 I2C_IER 与 I2C_IDR 的中断开启状态。此寄存器状态是将 I2C_IER 与 I2C_IDR 进行硬件运算，公式如下:I2C_IVS = I2C_IER & ~I2C_IDR

26. 6. 2. 14 I2C原始中断状态寄存器 (I2C_RIF)

I2C 原始中断状态寄存器 (I2Cx_RIF)																														
偏移地址:0x38																														
复位值:0x0000 0000																														
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
—	—	—	—	—	—	—	—	—	—	—	ALERT	TOUT	PECE	ARLO	BERR	—	STOP	NACK	ADDR	TCR	TC	—	RXUD	RXOV	—	RXNE	—	TXUD	TXOV	—	TXE

—	Bits 31-21	—	—
ALERT	Bit 20	R	**SMBus 警报，原始中断状态** 0:未发生中断事件 1:发生中断事件
TOUT	Bit 19	R	**超时，原始中断状态** 0:未发生中断事件 1:发生中断事件
PECE	Bit 18	R	**PEC 错误，原始中断状态** 0:未发生中断事件 1:发生中断事件
ARLO	Bit 17	R	**仲裁丢失，原始中断状态** 0:未发生中断事件 1:发生中断事件
BERR	Bit 16	R	**总线错误，原始中断状态** 0:未发生中断事件 1:发生中断事件
—	Bit 15	—	—
STOP	Bit 14	R	**停止位检测，原始中断状态** 0:未发生中断事件 1:发生中断事件
NACK	Bit 13	R	**接收否定应答，原始中断状态** 0:未发生中断事件 1:发生中断事件
ADDR	Bit 12	R	**地址匹配，原始中断状态** 0:未发生中断事件 1:发生中断事件
TCR	Bit 11	R	**传输完成等待重载，原始中断状态** 0:未发生中断事件

			1:发生中断事件
TC	Bit 10	R	**传输完成，原始中断状态** 0:未发生中断事件 1:发生中断事件
—	Bit 9	—	—
RXUD	Bit 8	R	**接收数据寄存器下溢，原始中断状态** 0:未发生中断事件 1:发生中断事件
RXOV	Bit 7	R	**接收数据寄存器上溢，原始中断状态** 0:未发生中断事件 1:发生中断事件
—	Bit 6	—	—
RXNE	Bit 5	R	**接收数据寄存器不为空，原始中断状态** 0:未发生中断事件 1:发生中断事件
—	Bit 4	—	—
TXUD	Bit 3	R	**发送数据寄存器下溢，原始中断状态** 0:未发生中断事件 1:发生中断事件
TXOV	Bit 2	R	**发送数据寄存器上溢，原始中断状态** 0:未发生中断事件 1:发生中断事件
—	Bit 1	—	—
TXE	Bit 0	R	**发送数据寄存器为空，原始中断状态** 0:未发生中断事件 1:发生中断事件

26. 6. 2. 15 I2C中断标志位状态寄存器(I2C_IFM)

I2C 中断标志位状态寄存器 (I2C_IFM)																														
偏移地址:0x3C																														
复位值:0x0000 0000																														
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
—	—	—	—	—	—	—	—	—	—	—	ALERT	TOUT	PECE	ARLO	BERR	—	STOP	NACK	ADDR	TCR	TC	—	RXUD	RXOV	—	RXNE	—	TXUD	TXOV	—	TXE

—	Bits 31-21	—	—
ALERT	Bit 20	R	**SMBus 警报，中断标志位状态** 0:未发生中断事件或中断未开启 1:产生中断
TOUT	Bit 19	R	**超时，中断标志位状态** 0:未发生中断事件或中断未开启 1:产生中断
PECE	Bit 18	R	**PEC 错误，中断标志位状态** 0:未发生中断事件或中断未开启 1:产生中断
ARLO	Bit 17	R	**仲裁丢失，中断标志位状态** 0:未发生中断事件或中断未开启 1:产生中断
BERR	Bit 16	R	**总线错误，中断标志位状态** 0:未发生中断事件或中断未开启 1:产生中断
—	Bit 15	—	—
STOP	Bit 14	R	**停止位检测，中断标志位状态** 0:未发生中断事件或中断未开启 1:产生中断
NACK	Bit 13	R	**接收否定应答，中断标志位状态** 0:未发生中断事件或中断未开启 1:产生中断
ADDR	Bit 12	R	**地址匹配，中断标志位状态** 0:未发生中断事件或中断未开启 1:产生中断
TCR	Bit 11	R	**传输完成等待重载，中断标志位状态** 0:未发生中断事件或中断未开启

			1:产生中断
TC	Bit 10	R	**传输完成，中断标志位状态** 0:未发生中断事件或中断未开启 1:产生中断
—	Bit 9	—	—
RXUD	Bit 8	R	**接收数据寄存器下溢，中断标志位状态** 0:未发生中断事件或中断未开启 1:产生中断
RXOV	Bit 7	R	**接收数据寄存器上溢，中断标志位状态** 0:未发生中断事件或中断未开启 1:产生中断
—	Bit 6	—	—
RXNE	Bit 5	R	**接收数据寄存器不为空，中断标志位状态** 0:未发生中断事件或中断未开启 1:产生中断
—	Bit 4	—	—
TXUD	Bit 3	R	**发送数据寄存器下溢，中断标志位状态** 0:未发生中断事件或中断未开启 1:产生中断
TXOV	Bit 2	R	**发送数据寄存器上溢，中断标志位状态** 0:未发生中断事件或中断未开启 1:产生中断
—	Bit 1	—	—
TXE	Bit 0	R	**发送数据寄存器为空，中断标志位状态** 0:未发生中断事件或中断未开启 1:产生中断

I2C_IFM 寄存器，是滤除已关闭中断功能的中断事件，只关注开启中断功能的事件。此寄存器状态是将 I2C_RIF 与 I2C_IVS 进行硬件运算，公式如下:I2C_IFM = I2C_RIF & I2C_IVS

26. 6. 2. 16 I2C中断清除寄存器 (I2C_ICR)

I2C 中断清除寄存器 (I2C_ICR)																														
偏移地址:0x40																														
复位值:0x0000 0000																														
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
—	—	—	—	—	—	—	—	—	—	—	ALERT	TOUT	PECE	ARLO	BERR	—	STOP	NACK	ADDR	TCR	TC	—	RXUD	RXOV	—	RXNE	—	TXUD	TXOV	—	TXE

—	Bits 31-21	—	—
ALERT	Bit 20	C_W1	**SMBus 警报中断清除** 0: 写入 0 无效 1: 清除中断事件与中断
TOUT	Bit 19	C_W1	**超时中断清除** 0: 写入 0 无效 1: 清除中断事件与中断
PECE	Bit 18	C_W1	**PEC 错误中断清除** 0: 写入 0 无效 1: 清除中断事件与中断
ARLO	Bit 17	C_W1	**仲裁丢失中断清除** 0: 写入 0 无效 1: 清除中断事件与中断
BERR	Bit 16	C_W1	**总线错误中断清除** 0: 写入 0 无效 1: 清除中断事件与中断
—	Bit 15	—	—
STOP	Bit 14	C_W1	**停止位检测中断清除** 0: 写入 0 无效 1: 清除中断事件与中断
NACK	Bit 13	C_W1	**接收否定应答中断清除** 0: 写入 0 无效 1: 清除中断事件与中断
ADDR	Bit 12	C_W1	**地址匹配中断清除** 0: 写入 0 无效 1: 清除中断事件与中断
TCR	Bit 11	C_W1	**传输完成等待重载中断清除** 0: 写入 0 无效

			1: 清除中断事件与中断
TC	Bit 10	C_W1	**传输完成中断清除** 0: 写入 0 无效 1: 清除中断事件与中断
—	Bit 9	—	—
RXUD	Bit 8	C_W1	**接收数据寄存器下溢中断清除** 0: 写入 0 无效 1: 清除中断事件与中断
RXOV	Bit 7	C_W1	**接收数据寄存器上溢中断清除** 0: 写入 0 无效 1: 清除中断事件与中断
—	Bit 6	—	—
RXNE	Bit 5	C_W1	**接收数据寄存器不为空中断清除** 0: 写入 0 无效 1: 清除中断事件与中断
—	Bit 4	—	—
TXUD	Bit 3	C_W1	**发送数据寄存器下溢中断清除** 0: 写入 0 无效 1: 清除中断事件与中断
TXOV	Bit 2	C_W1	**发送数据寄存器上溢中断清除** 0: 写入 0 无效 1: 清除中断事件与中断
—	Bit 1	—	—
TXE	Bit 0	C_W1	**发送数据寄存器为空中断清除** 0: 写入 0 无效 1: 清除中断事件与中断

I2C_ICR 寄存器设置时，将清除 I2C_RIF 与 I2C_IFM 中断标志位状态；此设置不影响中断 I2C_IER、I2C_IDR 与 I2C_IVS 寄存器，只清除标志位状态 I2C_RIF 与 I2C_IFM。此寄存器通过硬件清除中断，
公式如下:I2C_RIF = I2C_RIF & ~I2C_ICR

第27章 串行外设接口(SPI)

27. 1 概述

串行外设接口(SPI)可与外部 SPI 设备进行半双工或全双工的同步串行通信。该接口可配置为主机模式或从机模式。在配置为主机模式下，它可为外部 SPI 从设备提供通信时钟(SCK)。该接口还能够在多主机模式配置下工作。

27. 2 特性

- 支持主机或从机模式操作
- 基于三条线的全双工同步传输
- 基于双线的半双工同步传输，其中一条可作为双向数据线
- 基于双线的单工同步传输，其中一条作为单向数据线
- 8 位或 16 位传输帧格式选择
- 支持多主机模式功能
- 8 个主机模式波特率预分频器，最高可达 $f_{PCLK}/2$
- 从机模式频率最高可达 $f_{PCLK}/2$
- 对于主机模式和从机模式都可通过硬件或软件进行 NSS 控制
- 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
- 具有发送或接收的 FIFO 缓存状态标志位
- 具有显示 SPI 总线忙状态标志位
- 支持 SPI 摩托罗拉协议
- 支持 SPI TI 协议
- 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
- 提供独立发送和接收 FIFO 数据缓存
- 提供 12 种中断事件可触发中断
- 支持 DMA 传输：发送和接收请求

27. 3 SPI实现

本手册介绍了 SPI1 中实现的全部功能。

SPI 模式/特性	SPI1
Rx 和 TxFIFO 大小(N)[x 8 位]	4

表 28-1 SPI 特性

27. 4 SPI结构图

SPI 允许 MCU 与外部设备之间进行同步串行通信。应用软件可以通过轮询状态标志位或使用专用 SPI 中断来管理通信。SPI 的主要模块及其相互关系如下面的框图所示。

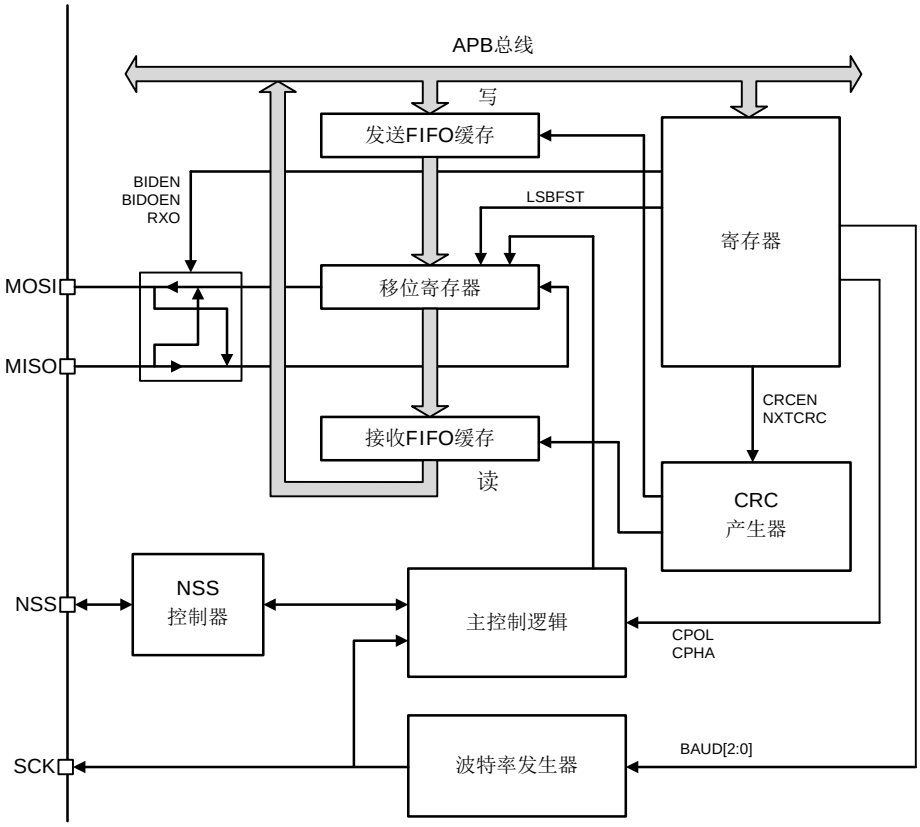


图 28-1 SPI 电路结构框图

通常，SPI 使用四个 I/O 引脚来与外部器件连接:

- MISO:主机输入/从机输出数据引脚。此引脚可用于在从机模式下发送数据和在主机模式下接收数据。
- MOSI:主机输出/从机输入数据引脚。此引脚可用于在主机模式下发送数据和在从机模式下接收数据。
- SCK:用于 SPI 主机的串行时钟输出以及 SPI 从机的串行时钟输入。
- NSS:从机选择引脚。此引脚用作“片选”，可让 SPI 主机与从机进行单独通信，从而避免

数据线上的竞争。从机的 NSS 输入可由主机上的标准 IO 端口驱动。

当配置为主机模式(**SPI_CON1**.MSTREN=1)。在 **SPI_CON2** 寄存器中 NSSOE 位置 1 时，NSS 引脚配置为输出，传输时硬件驱动 NSS 引脚为低电平。在 **SPI_CON2** 寄存器中 NSSOE 位置 0 时，NSS 引脚配置为输入，如果 NSS 被拉至低电平将产生冲突，**SPI_CON1**.MSTREN 位将自动清零，SPI 将进入模式错误状态:(更多信息请参见 28. 5. 14. 11)。

27. 5 功能描述

在 SPI 通信期间，接收和发送操作同时执行。串行时钟(SCK)同步数据线上信息的移位和采样。通信格式取决于时钟相位、时钟极性和数据帧格式。为了能够一起通信，主机和从机必须遵循相同的通信格式。

27. 5. 1 时钟相位和极性控制

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，此位对主机和从机都有作用。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，则为上升沿；如果 **SPI_CON1**.CPOL 位配置为 1，则为下降沿)对 MSB 采样。即在第一个时钟边沿锁存数据。

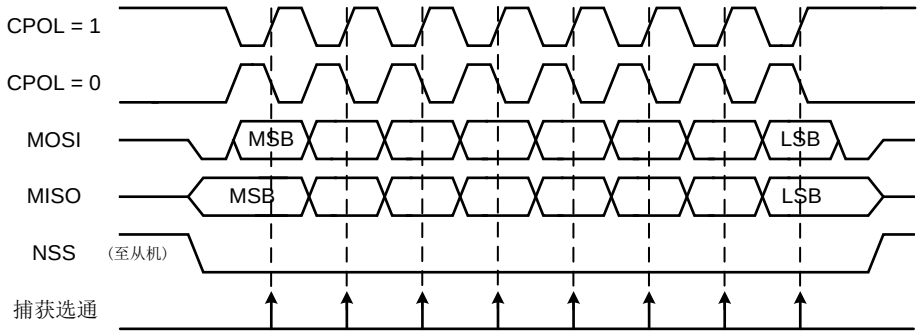
用户通过组合 **SPI_CON1**.CPOL 和 **SPI_CON1**.CPHA 位来选择数据捕获的时钟边沿。

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机或从机时序图，其中 SCK 引脚、MISO 引脚、MOSI 引脚直接连接在主机和从机之间。

注意:

1. 在切换 SPI_CON1.CPOL 或 SPI_CON1.CPHA 位之前，必须通过复位 SPI_CON1.SPIEN 位来关闭 SPI。
2. 必须以同一时序模式对主机和从机进行编程。
3. 主机和从机的时序需要配置成相同，通信才能正常。
4. 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)。
5. 通过 SPI_CON1.FLEN 位选择数据帧长度(8 或 16 位)，该格式决定了发送与接收过程中的数据长度。

CPHA=1 通信时序



CPHA=0 通信时序

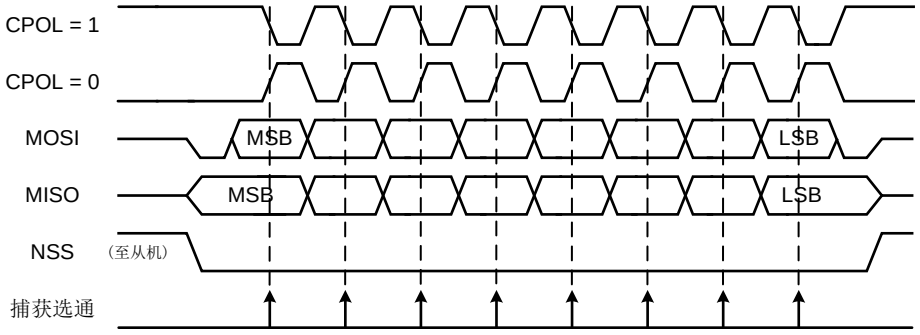


图 28-2　SPI 格式

注意:上图中显示的是当 SPI_CON1.LSBFST 处于复位状态时的时序图。

27. 5. 2　数据帧格式

SPI 移位寄存器可以设置为移出 MSB 优先或 LSB 优先，具体取决于 **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接收。

27. 5. 3　从机片选(NSS)引脚管理

可以使用 **SPI_CON1**.SSEN 位设置硬件或软件控制从机片选。

- 软件控制 NSS(**SPI_CON1**.SSEN = 1)从机片选是由内部 **SPI_CON1**.SSOUT 位的值决定。
- 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 SPI 设备在主机模式下工作时才使用此配置。当主机开始传输数据时，NSS 信号驱动为低电平，并保持到数据传输结束为止。

◇ NSS 输出禁止(**SPI_CON1**.SSEN = 0，**SPI_CON2**.NSSOE = 0)对于在主机模式下工作的设备，此配置允许多主机模式功能。对于设置为从机模式的设备，NSS 引脚用作传统 NSS 输入:在 NSS 为低电平时片选该从机，在 NSS 为高电平时取消对它的片选。

27. 5. 4 主机与从机的单对单通讯应用

SPI 允许 MCU 使用不同的配置进行通信，具体取决于所针对的设备和应用要求。当使用软件 NSS 管理时通过 2 或 3 线进行通信，使用硬件 NSS 管理时通过 3 或 4 线进行通信。通信始终由主机发起。

27. 5. 4. 1 全双工通信

默认情况下，SPI 配置为全双工通信。在此配置中，MOSI 引脚连接在一起，MISO 引脚连接在一起。通过这种方式，主机和从机之间以串行方式传输数据(最高有效位在前)。

通信始终由主机发起。当主机通过 MOSI 引脚向从机发送数据时，从机同时通过 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，时钟信号由主机的 SCK 引脚发出提供给从机。

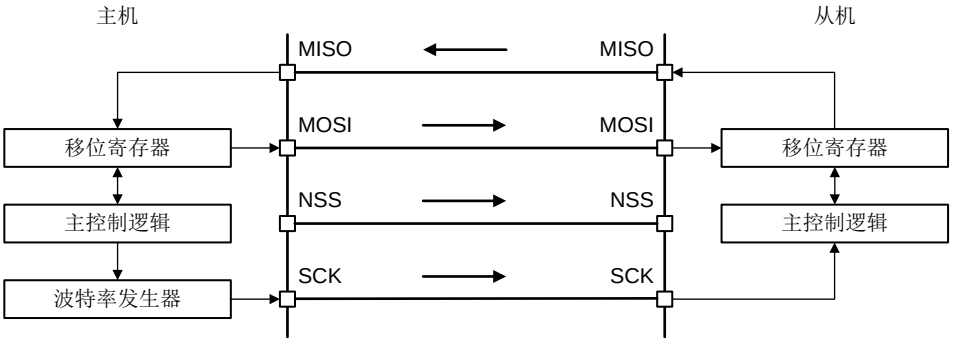


图 28-3 全双工通信

27. 5. 4. 2 半双工通信

SPI 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(主机模式下)或 MISO(从机模式下)作为数据通信引脚。通过 **SPI_CON1**.BIDOEN 位来选择传输方向(输入或输出)。当该位置 1 时数据线为输出，该位置 0 时为输入。

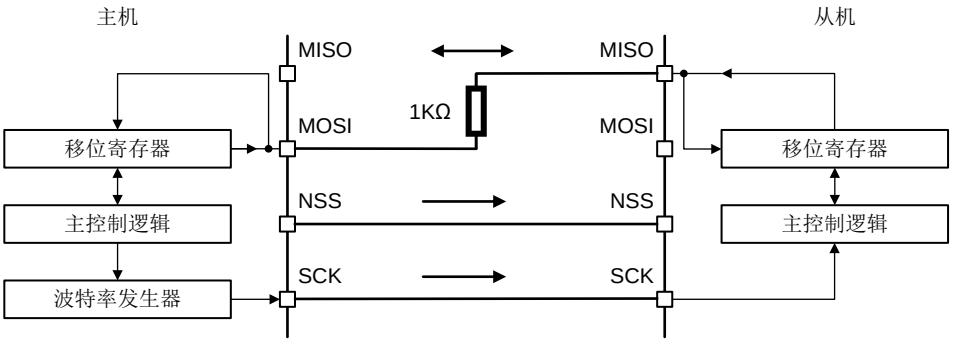


图 28-4 半双工通信

注意:

1. NSS 引脚可用于在主机和从机之间提供硬件控制流。如果不使用 NSS 引脚的话，必须在主机和从机的处理程序增加控制流。有关更多详细信息，请参见 28. 5. 3。
2. 在此配置中，主机的 MISO 引脚和从机的 MOSI 引脚可用作 GPIO。
3. 当在双向模式下工作的两个节点之间的通信方向更改不同步或同时在公共线上临时提供相反的输出电平时进行斗争时。建议在 MISO 和 MOSI 引脚之间插入一个串联电阻，以保护输出并在这种情况下限制它们之间的电流。

27. 5. 4. 3　　单工通信

SPI 可以在单工模式下通信，方法是使用 **SPI_CON1**.RXO 位将 SPI 设置为只发送或只接收。在这种配置中，只有一条线路用于主机和从机之间的数据传输。

- 只发送模式类似于全双工模式(**SPI_CON1**.BIDEN=0、**SPI_CON1**.RXO=0):在发送引脚(主机模式下的 MOSI 或从机模式下的 MISO)上发送数据。接收引脚(主机模式下的 MISO 或从机模式下的 MOSI)可用作通用 IO。在此模式下接收的数据是无意义的，应用程序只需要忽略接收 FIFO 缓存。
- 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。在这种情况下，发送 IO 引脚(主机模式下的 MOSI 或从机模式下的 MISO)可用于其它用途。

当 SPI 设置为只接收模式时:

- 一旦在主机模式下使能 SPI 后，主机会立即从 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。
- 在从机模式下，只要 NSS 引脚被拉低(或在 NSS 软件模式下将 **SPI_CON1**.SSOUT 位清零)，意味着从机被选中，同时一直有来自主机的 SCK 输入，SPI 就会继续接收。

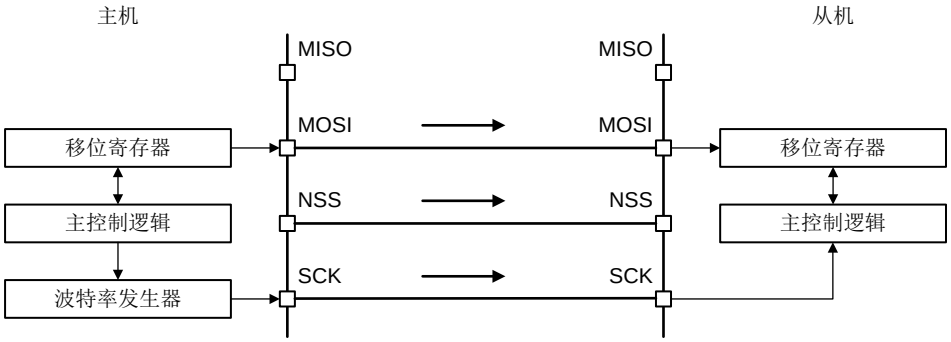


图 28-5　单工通信(主机模式下的只发送与从机模式下的只接收)

27. 5. 5 标准多从机通讯应用

在具有两个或更多独立从机的配置中，主机使用 GPIO 引脚来管理每个从机的芯片选择线，请见下图。主机必须通过拉低连接到从机 NSS 输入的 GPIO 来单独选择一个从机，实现主机和被选择从机的专用通信。

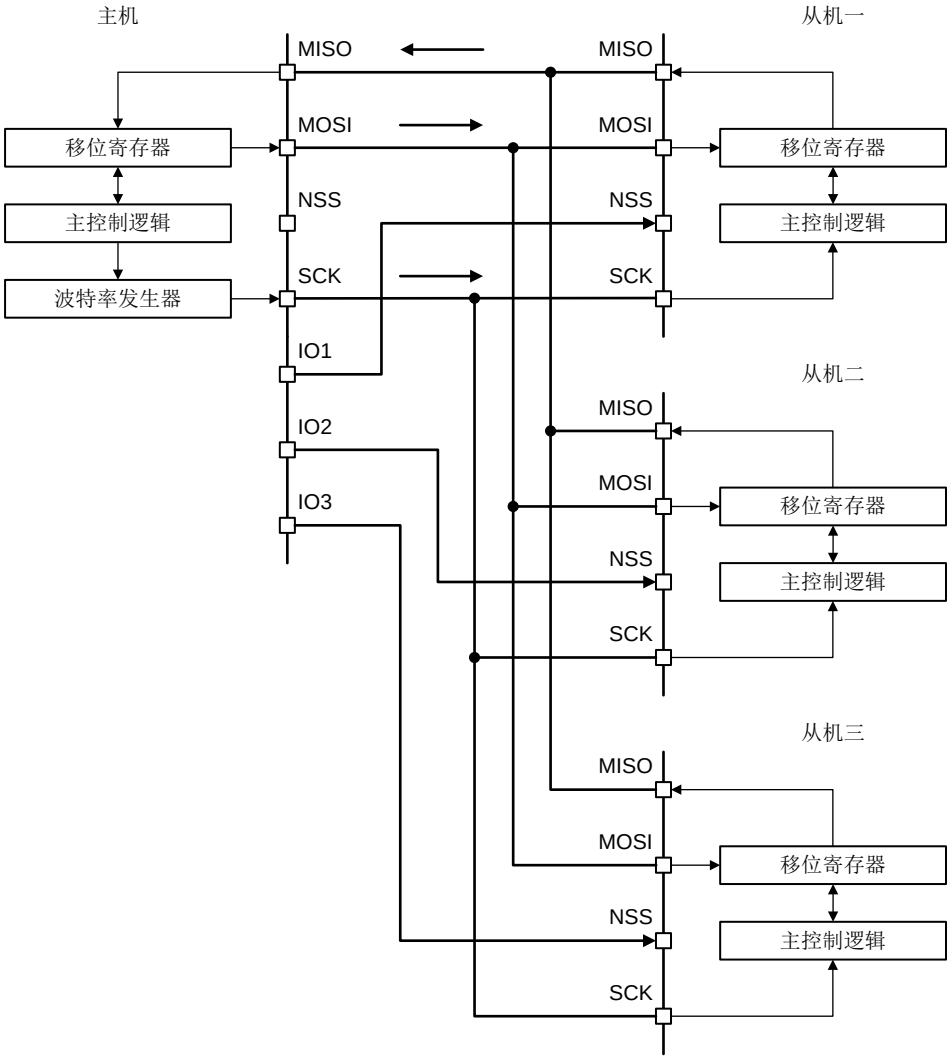


图 28-6 多从机通讯(一个主机和三个从机)

27. 5. 6 多主机通讯应用

多主机模式仅支持的两个SPI节点，因为一次只有一个节点可以将其输出应用于公共数据线上。可以使用内置功能来检测主控总线的两个节点之间的潜在冲突。对于此检测必须将 NSS 引脚配置为硬件输入模式。

当节点处于非活动状态时，默认情况下两个节点都处于从机模式。一旦一个节点想要超越总线上的控制，它就会切换到主机模式并通过专用的 GPIO 引脚对另一个节点的从机选择输入应用活动电平。会话完成后释放活动的从机选择信号，节点控制总线临时返回被动从机模式，等待下一个会话启动。

如果可能两个节点同时提出了它们的主控请求，则会出现总线冲突事件(参见 28. 5. 14. 11)。用户可以应用一些简单的仲裁过程(例如通过在两个节点上定义不同的超时机制来推迟下一次请求)。

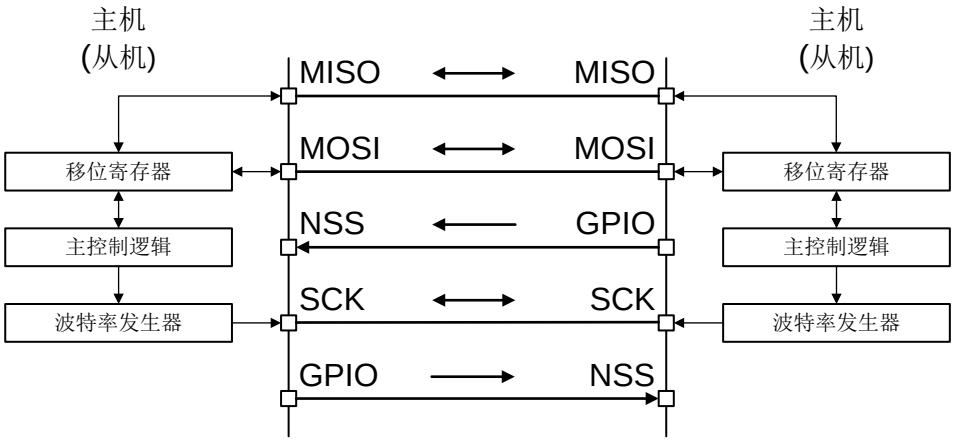


图 28-7 多主机通讯应用

27. 5. 7 SPI配置成从机模式

SPI 作为从机时，时钟信号由主机提供，所以建议先使能从机，然后主机再发送时钟，否则数据传输可能不正常。建议按以下步骤配置 SPI 为从机:

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，注意从机和主机的这两位配置需保持一致。
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)必须与主机的帧格式相同。
4. 在硬件模式下(请参见 28. 5. 3)，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 FIFO 缓存中。

当从机收到时钟信号并在 MOSI 引脚上收到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**.TXE 标志在发送 FIFO 缓存的最后一笔数据加载到移位寄存器时置 1，并且在 **SPI_IER**.TXE 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，移位寄存器中的数据将传输到接收 FIFO 缓存，并且将 **SPI_STAT**.RXNE 位置 1。

在出现最后一个采样时钟边沿后，将 **SPI_STAT**.RXNE 位置 1，移位寄存器中接收的数据字节被拷贝到接收 FIFO 缓存中，并且在 **SPI_IER**.RXNE 位置 1 时将生成中断。当读取 **SPI_DATA** 寄存器时，SPI 外设将返回此缓冲值。

27. 5. 8 SPI配置成主机模式

时钟信号由主机从 SCK 引脚发出传输给从机。

步骤

1. 先配置时钟，设置 **SPI_CON1**.BAUD 位域，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)(参见图 28-2)。
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 位设置定义帧格式，可选择 MSB 在前或 LSB 在前。
5. 当 NSS 引脚配置成输入时，在 NSS 硬件模式下，NSS 引脚在整个发送序列期间必须连接到高电平信号；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 1 使 SPI 工作在主机模式下，然后 **SPI_CON1**.SPIEN 位置 1 使能 SPI 模块(当 NSS 引脚配置成输入且在 NSS 硬件模式时，NSS 引脚必须维持高电平信号，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 **SPI_DATA** 寄存器写入字节时，发送序列开始。在第一个位传输期间，数据(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。TXE 标志在发送 FIFO 缓存的最后一笔数据加载到移位寄存器时置 1，并且在 **SPI_IER**.TXE 位置 1 时将生成中断。

接收序列

对于接收器，在数据传输最后一个采样时钟边沿时，将 **SPI_STAT**.RXNE 位置 1，移位寄存器中接收的数据字节被拷贝到接收 FIFO 缓存中，并且在 **SPI_IER**.RXNE 位置 1 时将生成中断。当读取 **SPI_DATA** 寄存器时，SPI 外设将返回此缓冲值。

如果在发送开始后将要发送的下一个数据置于发送 FIFO 缓存，则可保持连续的发送流。请注意，仅当 **SPI_STAT**.TXF 位为 0 时，才可以对发送 FIFO 缓存执行写操作。

注意:如果与之通信的从机需要在每个字节传输之间拉低片选信号，必须将该主机的 NSS 配置成 GPIO，或使用另外 GPIO，通过软件控制从机的片选。

27. 5. 9 数据发送和接收

27. 5. 9. 1 接收和发送FIFO缓存

所有 SPI 数据传输都通过嵌入式的 FIFO 缓存。使 SPI 能够连续传输工作。发送和接收都有自己的 FIFO 缓存。

对 **SPI_DATA** 寄存器的读访问将返回存储在接收 FIFO 缓存中但尚未读取的最旧的值。对 **SPI_DATA** 的写访问将已写数据存储在发送队列末尾的发送 FIFO 缓存中。**SPI_STAT** 寄存器中 RXFLV 和 TXFLV 位域指示两个 FIFO 缓存的有效数据个数。

对 **SPI_DATA** 寄存器的读访问必须由 RXTH 事件管理。当数据存储在接收 FIFO 缓存中并且达到阈值(由 **SPI_CON2** 寄存器中 RXFTH 位域定义)时，触发此事件。当 RXTH 被清除时，表示接收 FIFO 缓存中的有效数据个数小于阈值。以类似的方式，要发送的数据帧的写访问由 TXTH 事件管理。当发送 FIFO 缓存有效数据个数小于或等于阈值(由 **SPI_CON2** 寄存器中 TXFTH 位域定义)时将触发此事件。

27. 5. 9. 2 在主机模式下启动通信序列

- 在全双工通信(**SPI_CON1**.BIDEN=0 且 **SPI_CON1**.RXO=0)
  - 将数据写入到 **SPI_DATA** 寄存器(发送 FIFO 缓存)后，启动通信序列。
  - 随后在第一个位的发送期间，将数据从发送 FIFO 缓存并行加载到 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 FIFO 缓存)中。

- 在单工通信-只接收模式(**SPI_CON1**.BIDEN=0 且 **SPI_CON1**.RXO=1)
  - 只要 **SPI_CON1**.SPIEN = 1，通信序列就立即开始。
  - 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 FIFO 缓存)中。

- 在半双工通信-发送模式(**SPI_CON1**.BIDEN=1 且 **SPI_CON1**.BIDOEN=1)
  - 将数据写入到 **SPI_DATA** 寄存器(发送 FIFO 缓存)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。

- 在半双工通信-接收模式(**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 FIFO 缓存)中。
  - 不会有数据以串行方式移出 MOSI 引脚。

27. 5. 9. 3 在从机模式下启动通信序列

- 在全双工模式(**SPI_CON1**.BIDEN=0 且 **SPI_CON1**.RXO=0)
  - 当从机收到时钟信号并在 MOSI 引脚上收到数据的第一个位时，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 主机启动传输前，软件必须已把要从机发送的数据写入发送 FIFO 缓存。

- 在单工通信-只接收模式(**SPI_CON1**.BIDEN=0 且 **SPI_CON1**.RXO=1)
  - 当从机收到时钟信号并在 MOSI 引脚上收到数据的第一个位时，通信序列开始。其余 7 个位将加载到移位寄存器中。
  - 不会有数据以串行方式移出 MISO 引脚。

- 在半双工通信-发送模式(**SPI_CON1**.BIDEN=1 且 **SPI_CON1**.BIDOEN=1)
  - 当从机收到时钟信号，并且 MISO 引脚上发出发送 FIFO 缓存中的第一位数据时，通信序列开始。
  - 随后在第一个位的发送期间，将数据从发送 FIFO 缓存并行加载到 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 主机启动传输前，软件必须已把要从机发送的数据写入发送 FIFO 缓存。
  - 不接收任何数据。

- 在半双工通信-接收模式(**SPI_CON1**.BIDEN=1 且 **SPI_CON1**.BIDOEN=0)
  - 当从机收到时钟信号并在 MOSI 引脚上收到数据的第一个位时，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 FIFO 缓存)中。
  - 不会有数据以串行方式移出 MISO 引脚。

27. 5. 9. 4 处理数据发送与接收

全双工通信(SPI_CON1.BIDEN=0 且 SPI_CON1.RXO=0)，发送和接收数据的处理过程

直接存取操作模式(参见图 28-8):

1. 通过将 **SPI_CON1**.SPIEN 位置 1 来使能 SPI，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**.TXE 位清零)。
2. 等待 **SPI_STAT**.TXE=1，然后写入要发送的第二个数据项。然后等待 **SPI_STAT**.RXNE=1，读取 **SPI_DATA** 以获取第一个接收到的数据(此操作会将 **SPI_STAT**.RXNE 位清零)。对每个要发送和接收的数据项重复此操作，直到发送并接收完最后的数据。
3. 检查 **SPI_STAT**.TXE=1，然后等待至 **SPI_STAT**.BUSY=0，再关闭 SPI。
4. 此外，还可以使用 TXE 或 RXNE 中断事件对应的各个中断子程序来实现该过程。

FIFO 缓存操作模式(参见图 28-9):

1. 通过将 **SPI_CON1**.SPIEN 位置 1 来使能 SPI。

2. 配置 **SPI_CON2**.TXFTH 与 **SPI_CON2**.RXFTH。
3. 当 **SPI_STAT**.TXTH=1，将要发送的数据写入 **SPI_DATA** 寄存器(写入的数据个数必须大于 **SPI_CON2**.TXFTH 设定的阀值)，当 **SPI_STAT**.RXTH=1，读取 **SPI_DATA** 寄存器以获取接收到的数据(读取的数据个数必须为 **SPI_CON2**.RXFTH 设定的阀值)，重复此操作直到写入最后要发送的数据。
4. 等待至 **SPI_STAT**.BUSY=0，读取 **SPI_DATA** 寄存器以获取接收到的数据直到 **SPI_STAT**.RXFLV 位域为 0，再关闭 SPI。
5. 此外，还可以使用 TXTH 或 RXTH 中断事件对应的各个中断子程序来实现该过程。

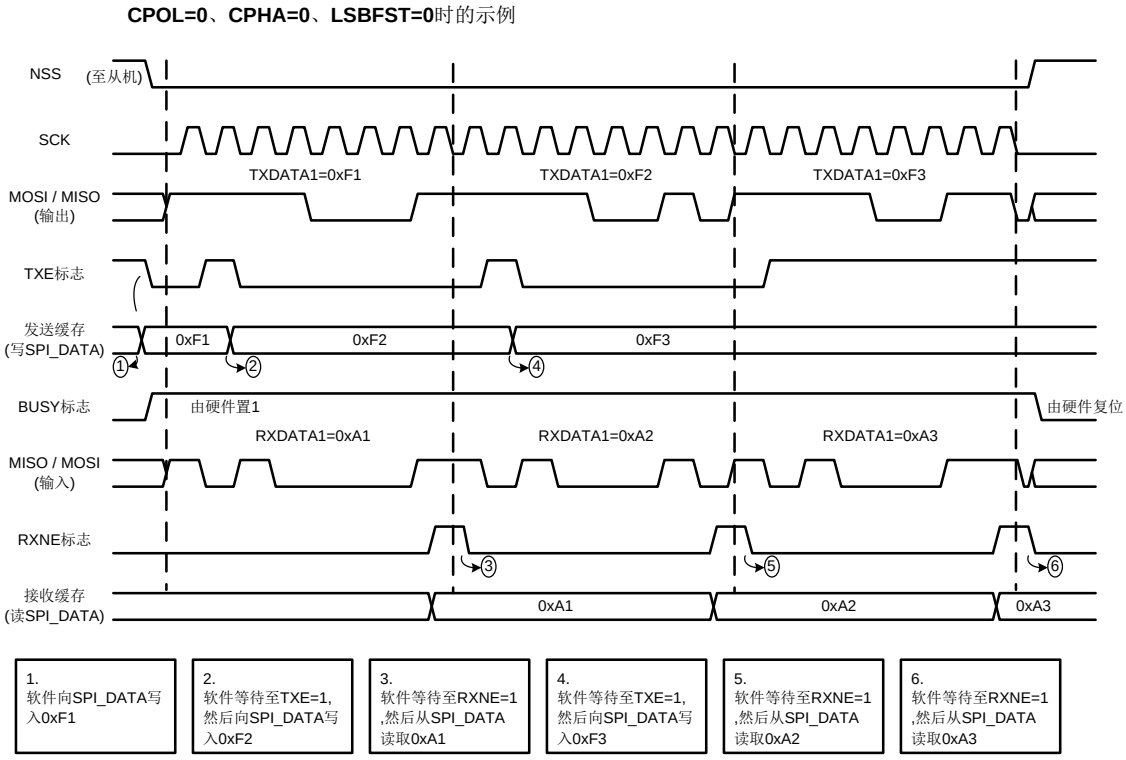


图 28-8 全双工通信(SPI_CON1.BIDEN=0 且 SPI_CON1.RXO=0)的 TXE、RXNE、BUSY 行为(直接存取操作模式在连续传输的情况下)

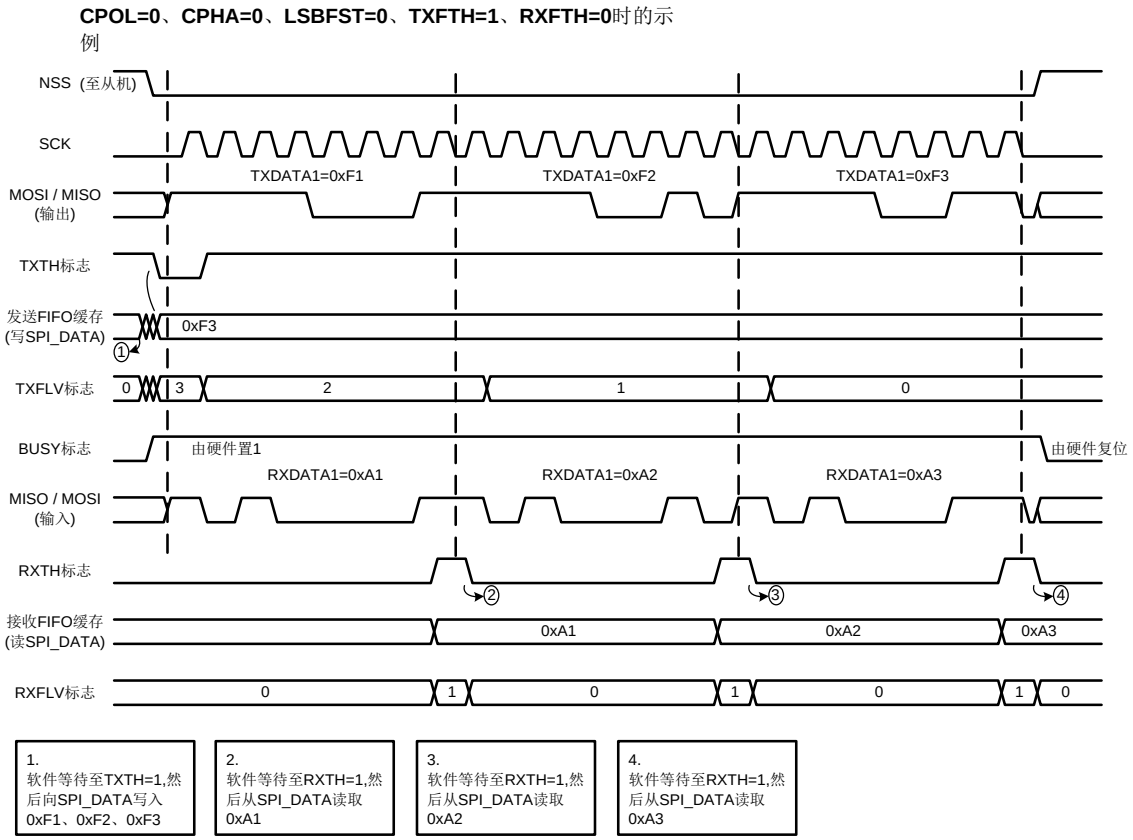


图 28-9　全双工通信(SPI_CON1.BIDEN=0 且 SPI_CON1.RXO=0)的 TXTH、RXTH、TXFLV、RXFLV、BUSY 行为(FIFO 缓存操作模式在连续传输的情况下)

单工通信-只发送模式(SPI_CON1.BIDEN=0、SPI_CON1.RXO=0)，发送数据的处理过程

直接存取操作模式(参见图 28-10):

1. 通过将 **SPI_CON1**.SPIEN 位置 1 来使能 SPI。
2. 等待 **SPI_STAT**.TXE=1 然后写入要发送的数据。对每个要发送的数据项重复此步骤。
3. 将最后一个数据写入 **SPI_DATA** 寄存器后，等待至 **SPI_STAT**.TXE=1，然后等待至 **SPI_STAT**.BUSY=0 再关闭 SPI，这表示最后的数据发送完成。
4. 此外，还可以使用在 TXE 中断事件对应的中断子程序来实现该过程。

FIFO 缓存操作模式(参见图 28-11):

1. 通过将 **SPI_CON1**.SPIEN 位置 1 来使能 SPI。
2. 配置 **SPI_CON2**.TXFTH。
3. 当 **SPI_STAT**.TXTH=1，将要发送的数据写入 **SPI_DATA** 寄存器(写入的数据个数必须大于 **SPI_CON2**.TXFTH 设定的阀值)，重复此操作直到写入最后要发送的数据。
4. 等待至 **SPI_STAT**.BUSY=0 再关闭 SPI。
5. 此外，还可以使用 TXTH 中断事件对应的中断子程序来实现该过程。

注意:

1. 在不连续通信期间，在对 SPI_DATA 寄存器执行写操作与 SPI_STAT.BUSY 位置 1 之间有延迟。因此在

只发送模式下，写入最后的数据后，必须先等待 SPI_STAT.TXE 位置 1，然后等待 SPI_STAT.BUSY 位清零。

2. 在只发送模式下，发送 17 个数据项后，SPI_STAT.RXOV 标志将置 1，因为始终不会读取接收的数据。

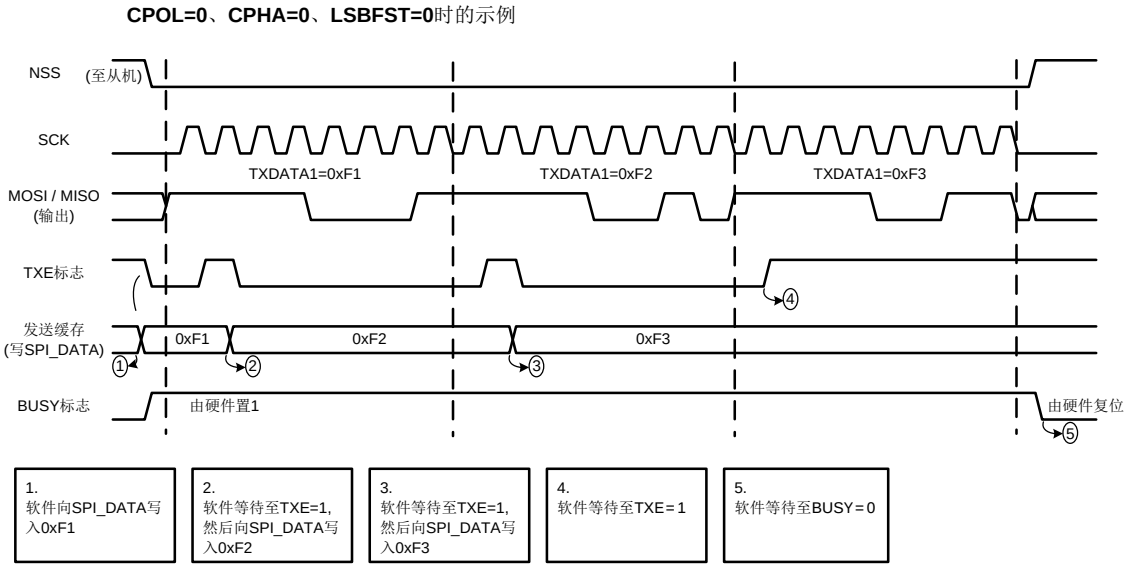


图 28-10 单工通信-只发送模式(SPI_CON1.BIDEN=0 且 SPI_CON1.RXO=0)的 TXE、BUSY 行为(直接存取操作模式在连续传输的情况下)

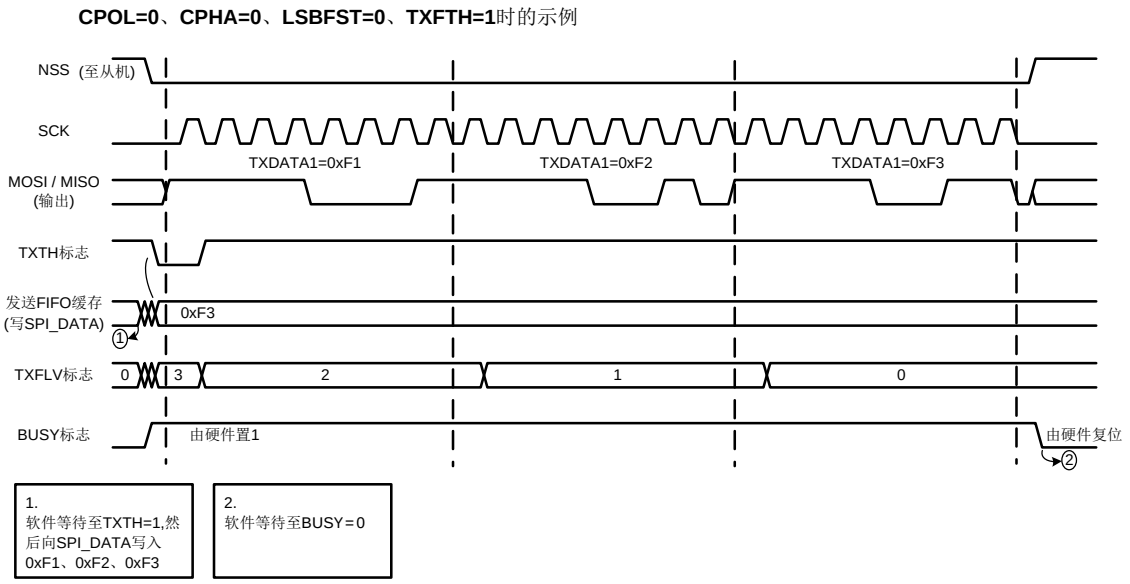


图 28-11 单工通信-只发送模式(SPI_CON1.BIDEN=0 且 SPI_CON1.RXO=0)的 TXTH、TXFLV、BUSY 行为(FIFO 缓存操作模式在连续传输的情况下)

半双工通信-发送模式(SPI_CON1.BIDEN=1 且 SPI_CON1.BIDOEN=1)，发送数据的处理过程

此模式与单工通信-只发送模式数据的处理过相似，但是在 SPI 模块使能前，必须将 **SPI_CON1**.BIDEN 位和 **SPI_CON1**.BIDOEN 位置 1。

单工通信-只接收模式(SPI_CON1.BIDEN=0 且 SPI_CON1.RXO=1)，接收数据的处理过程

直接存取操作模式(参见图 28-12):

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**.RXNE=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**.RXNE 位清零)。对每个要接收的数据项重复此操作。
4. 此外，还可以使用 RXNE 中断事件对应的中断子程序来实现该过程。

FIFO 缓存操作模式(参见图 28-13):

1. 将 **SPI_CON1**.RXO 位置 1。
2. 配置 **SPI_CON2**.RXFTH。
3. 通过将 **SPI_CON1**.SPIEN 位置 1 使能 SPI。
4. 当 **SPI_STAT**.RXTH=1，读取 **SPI_DATA** 寄存器以获取接收到的数据(读取的数据个数必须为 **SPI_CON2**.RXFTH 设定的阀值)，重复此操作直到获取最后要接收的数据。
5. 此外，还可以使用 RXTH 中断事件对应的中断子程序来实现该过程。

注意:

1. 在主机模式下，一旦 SPI 使能后 SCK 会立即发送时钟，从机接收到时钟后会发送数据，直到主机关闭 SPI 功能结束通信。
2. 在从机模式下，当 NSS 被拉低并且接收到 SCK 时钟后开始接收数据。
3. 如果需要在最后一次传输后关闭 SPI，请参见 28. 5. 10

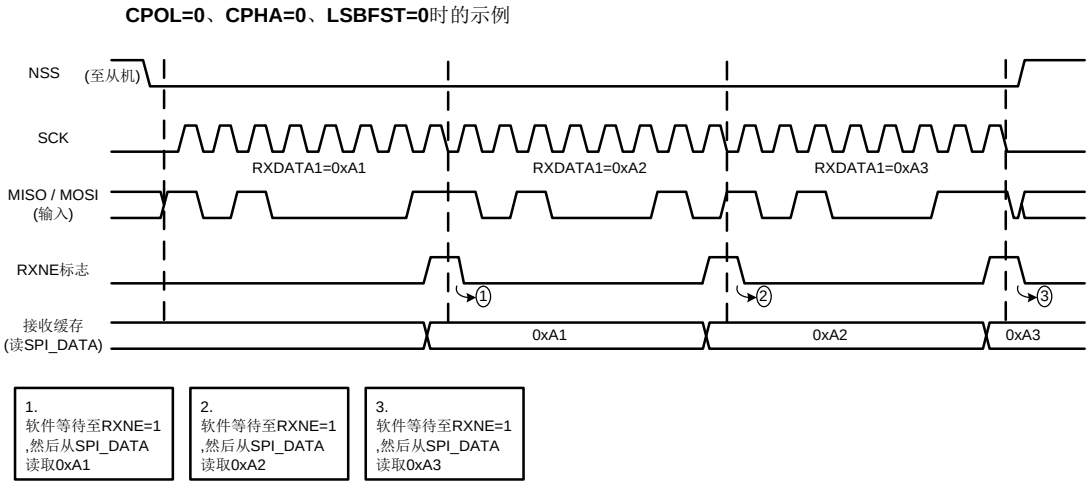


图 28-12　单工通信-只接收模式(SPI_CON1.BIDEN=0 且 SPI_CON1.RXO=1)的 RXNE 行为(直接存取操作模式在连续传输的情况下)

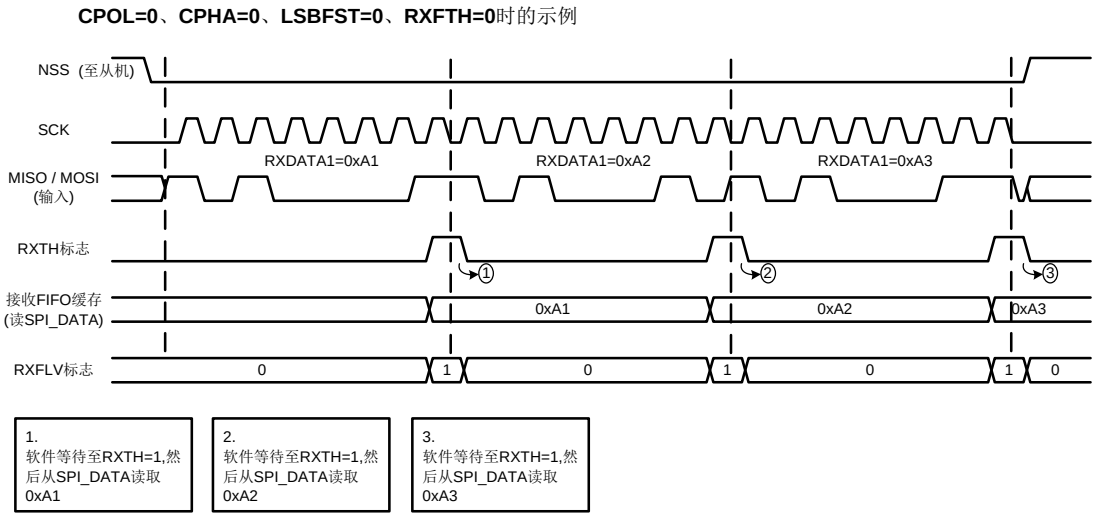


图 28-13　单工通信-只接收模式(SPI_CON1.BIDEN=0 且 SPI_CON1.RXO=1)的 RXTH、RXFLV 行为(FIFO 缓存操作模式在连续传输的情况下)

半双工通信-接收模式(SPI_CON1.BIDEN=1 和 SPI_CON1.BIDOEN=0)，接收数据的处理过程

此模式与单工通信-只接收模式数据的处理过程相似，但是在 SPI 模块使能之前，需要将 **SPI_CON1**.BIDEN 位置 1,并将 **SPI_CON1**.BIDOEN 与 **SPI_CON1**.RXO 位清 0。

连续传输和间断传输

在主机模式下发送数据时，如果软件处理速度足够快，可以在检测到 **SPI_STAT**.TXE=1(或发生 TXE 中断事件)，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。或者配置 **SPI_CON2**.TXFTH 检测当 **SPI_STAT**.TXTH=1(或发生 TXTH 中断事件)，将要发送的数据写入 **SPI_DATA** 寄存器(写入的数据个数必须大于 **SPI_CON2**.TXFTH 设定的阀值)，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。

在主机或从机模式下的单工通信-只接收模式(**SPI_CON1**.RXO=1)，通信始终是连续的，且 **SPI_STAT**.BUSY 位始终为 1。

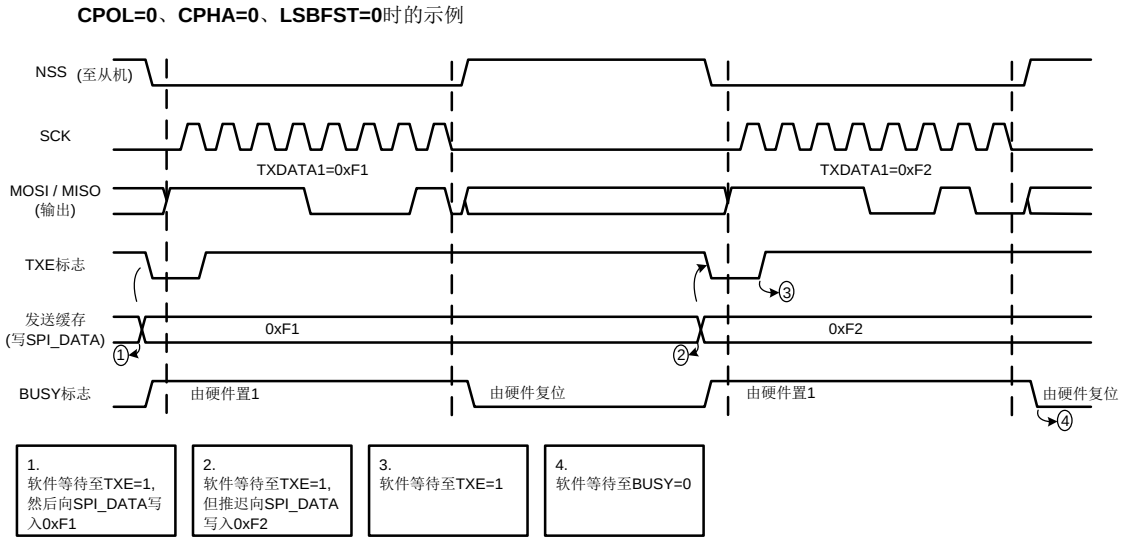


图 28-14　发送时(SPI_CON1.BIDEN =0 且 SPI_CON1.RXO=0)的 TXE、BUSY 行为(在间断传输的情况下)

27. 5. 10 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:

27. 5. 10. 1 在主机或从机的全双工通信(SPI_CON1.BIDEN=0、SPI_CON1.RXO=0)

1. 等待 **SPI_STAT**.RXNE=1 或 **SPI_STAT**.RXFLV!=0 以接收最后的数据。
2. 等待 **SPI_STAT**.TXE=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)。

27. 5. 10. 2 在主机或从机的单工通信-只发送模式(SPI_CON1.BIDEN=0、SPI_CON1.RXO=0)或半双工通信-发送模式(SPI_CON1.BIDEN=1、SPI_CON1.BIDOEN=1)

在最后的数据写入 **SPI_DATA** 寄存器后:

1. 等待 **SPI_STAT**.TXE=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)。

27. 5. 10. 3 在主机的单工通信-只接收模式(SPI_CON1.MSTREN=1、SPI_CON1.BIDEN=0、SPI_CON1.RXO=1)或半双工通信-接收模式(SPI_CON1.MSTREN=1、SPI_CON1.BIDEN=1、SPI_CON1.BIDOEN=0)

避免多余的 SPI 数据传输,必须以特殊方式管理这种情况:

1. 等待倒数第二个数据(第 n-1 个)对应的 RXNE 标志位置 1。
2. 在单工通信下将 **SPI_CON1** 寄存器中 RXO 位清零。在半双工通信下则将 **SPI_CON1** 寄存器中 BIDOEN 置 1。
3. 再等待最后的 **SPI_STAT**.RXNE=1 或 **SPI_STAT**.RXFLV!=0，才能关闭 SPI(SPIEN=0)然后进入停止模式(或关闭外设时钟)。

27. 5. 10. 4 在从机的单工通信-只接收从模式(SPI_CON1.MSTREN=0、SPI_CON1.BIDEN=0、SPI_CON1.RXO=1)或半双工通信-接收模式(SPI_CON1.MSTREN=0、SPI_CON1.BIDEN=1、SPI_CON1.BIDOEN=0)

1. 可以随时关闭 SPI(写入 **SPI_CON1**.SPIEN=0)。当前传输将舍弃并立即关闭 SPI。
2. 如果要进入停止模式，则必须首先等待至 **SPI_STAT**.BUSY = 0，才能关闭 SPI(SPIEN=0)，然后才能进入停止模式(或关闭外设时钟)。

27. 5. 11 DMA请求

为了更方便的实现高速通信，SPI 提供了 DMA 功能。DMA 请求条件是根据 **SPI_CON2** 寄存器中的 TXFTH 与 RXFTH 位域配置，当使能 **SPI_CON2** 寄存器中相应的 DMA 使能位时，将请求 DMA 访问。发送 FIFO 缓存和接收 FIFO 缓存会发出各自的 DMA 请求(参见图 28-15 和图 28-16)：

- 在发送过程中，当 **SPI_STAT**.TXTH 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**.TXTH 位清零)。
- 在接收过程中，当 **SPI_STAT**.RXTH 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**.RXTH 位清零)。

当 SPI 仅用于只发送数据时，可以只使能 SPI TX DMA 通道。在这种情况下，**SPI_STAT**.RXOV 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_RIF** 寄存器会产生相对应通道的传输完成标志，用户可以对 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等待 **SPI_STAT**.BUSY=0。

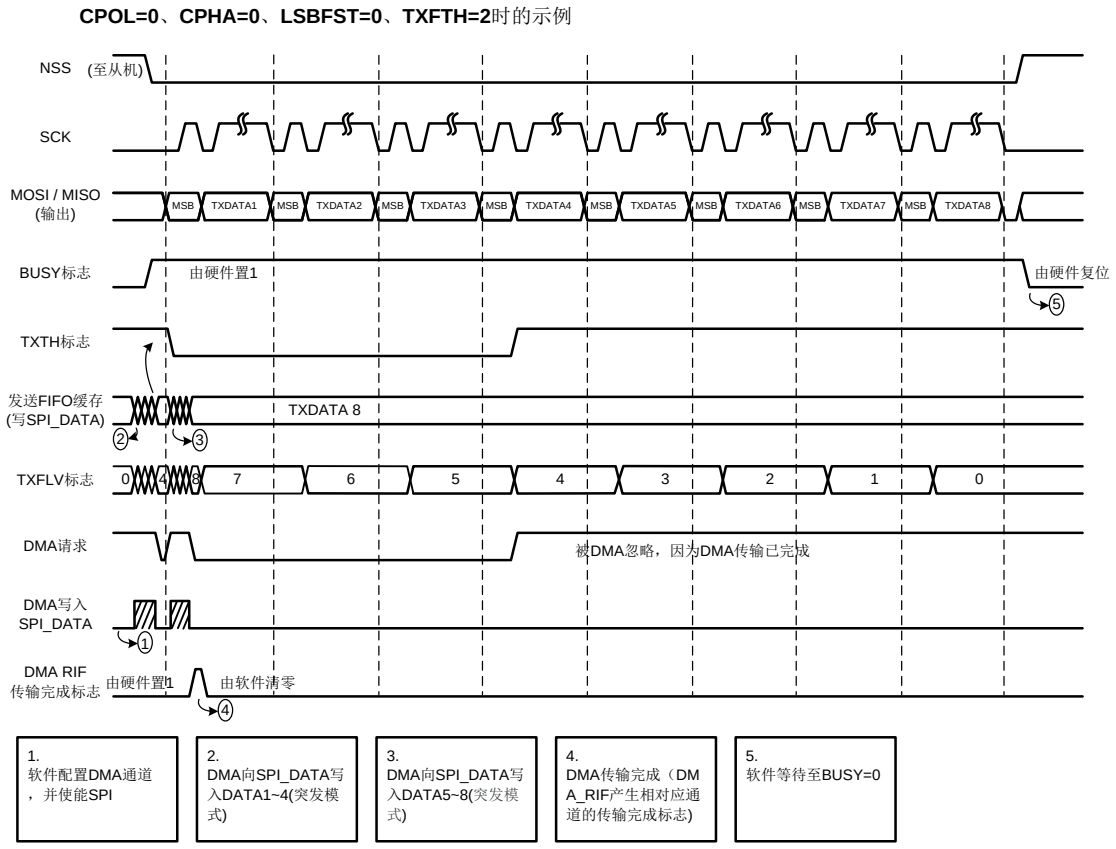


图 28-15　使用 DMA 进行发送

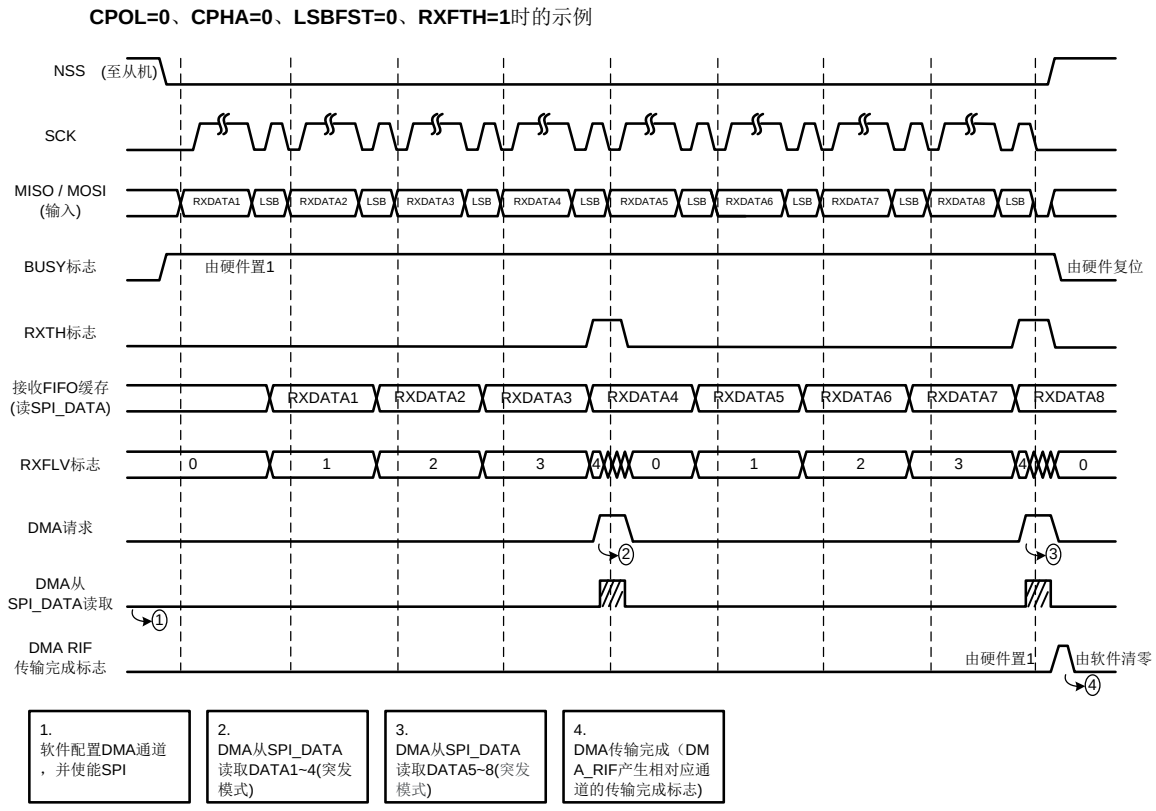


图 28-16　使用 DMA 进行接收

27. 5. 12 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。CRC 是使用 **SPI_CRCPOLY** 寄存器中编程的多项式串行计算的。

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，在连续传输的情况下,可以在最后一笔数据写入前任意时间点将 **SPI_CON1**.NXTCRC 位置 1，当最后一次数据传输结束时，将发送 **SPI_TXCRC** 寄存器内的值。在间断传输的情况下,必须在最后传输的数据写入 **SPI_DATA** 后，立即对 **SPI_CON1**.NXTCRC 位执行写操作，当最后一次数据传输结束时，将发送 **SPI_TXCRC** 寄存器内的值。如果传输由 DMA 管理，则在使能发送 FIFO 缓存 DMA 前，将 **SPI_CON1**.NXTCRC 位置 1，当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，在连续传输的情况下,可以在接收到最后一个数据前将 **SPI_CON1**.NXTCRC 位置 1，在收到最后一个数据后会收到 CRC，然后执行 CRC 校验。在间断传输的情况下,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。如果传输由 DMA 管理，则在使能接收 FIFO 缓存 DMA 前，将 **SPI_CON1**.NXTCRC 位置 1，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_RIF**.CRCERR 位将置 1。

如果发送 FIFO 缓存中存在数据，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，在连续传输的情况下,可以在最后一笔数据写入前任意时间点将 **SPI_CON1**.NXTCRC 位置 1，以表示在发送完最后一个字节后将发送 CRC。在间断传输的情况下,必须在最后传输的数据写入 **SPI_DATA** 后，立即对 **SPI_CON1**.NXTCRC 位执行写操作，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，在连续传输的情况下,可以在接收到最后一个数据前将 **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。在间断传输的情况下,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，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_RIF**.CRCERR 位将置 1，并且在 **SPI_IER**.CRCERR 位置 1 时会产生中断。

当 SPI 处于从机模式时，注意只能在时钟稳定(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。

在 SPI 通信时钟频率较高的情况下，发送 CRC 时务必小心。应于在 CRC 传输阶段 CPU 尽可能保持空闲，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。实际上在发送或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 SPI 配置为从机，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。

在对从机片选的切换期间内，应在主机和从机两端同时将 CRC 值清零，以重新同步主机和机双方的 CRC 计算。

要将 CRC 值清零，请按以下步骤操作:

1. 将 **SPI_CON1**.CRCEN 位清零。
2. 将 **SPI_CON1**.CRCEN 位置 1。

27. 5. 13 SPI状态标志

27. 5. 13. 1 发送FIFO缓存为空(TXE)

此标志置 1 时，表示发送 FIFO 缓存为空，此时可以将待发送的数据加载到发送 FIFO 缓存中。对 **SPI_DATA** 寄存器执行写操作时，会将 TXE 标志清零。

27. 5. 13. 2 发送FIFO缓存为满(TXF)

此标志置 1 时，表示发送 FIFO 缓存为满，此时无法将待发送的数据加载到发送 FIFO 缓存中。当从发送 FIFO 缓存加载一个数据到移位寄存器时，会将 TXF 标志清零。

27. 5. 13. 3 发送FIFO缓存上溢(TXOV)

当发送 FIFO 缓存已满时，用户对 **SPI_DATA** 寄存器执行写操作。在这种情况下，新写入的数据不会加载到发送 FIFO 缓存中，并将此标志置 1。对 **SPI_STAT** 寄存器执行读访问时，将 TXOV 标志清零。

27. 5. 13. 4 发送FIFO缓存下溢(TXUD)

在从机模式下当发送 FIFO 缓存为空时，但主机提出数据请求。在这种情况下，不会有数据从发送 FIFO 缓存加载到移位寄存器中,并将此标志置 1。对 **SPI_STAT** 寄存器执行读访问时，将 TXUD 标志清零。

27. 5. 13. 5 发送FIFO缓存阈值(TXTH)

此标志置 1 时，表示发送 FIFO 缓存中的有效数据个数少于或者等于 **SPI_CON2**.TXFTH 设置的值，此时可以将待发送的数据加载到发送 FIFO 缓存中。当加载到 FIFO 缓存中的有效数据个数大于 **SPI_CON2**.TXFTH 设置的值时，会将 TXTH 标志清零。

27. 5. 13. 6 接收FIFO缓存为非空(RXNE)

此标志置 1 时，表示接收 FIFO 缓存中存在有效的已接收数据。此时用户可读取 **SPI_DATA** 寄存器，当读取后接收 FIFO 缓存中没有效数据时，此标志位被清零。

27. 5. 13. 7 接收FIFO缓存为满(RXF)

此标志置 1 时，表示接收 FIFO 缓存为满，此时无法将接收的数据加载到接收 FIFO 缓存中。对 **SPI_DATA** 寄存器执行读访问时，将 RXF 标志清零。

27. 5. 13. 8 接收FIFO缓存上溢(RXOV)

当接收 FIFO 缓存已满时，用户没对 **SPI_DATA** 寄存器执行读访问。在这种情况下，主机发送的下一个数据帧不会加载到接收 FIFO 缓存中,同时将此标志置 1。对 **SPI_STAT** 寄存器执行读访问时，会将 RXOV 标志清零。

27. 5. 13. 9 接收FIFO缓存下溢(RXUD)

当接收 FIFO 缓存为空时，但用户对 **SPI_DATA** 寄存器执行读访问。在这种情况下，读访问不会从接收 FIFO 缓存中读到有效的数据,并将此标志置 1。对 **SPI_STAT** 寄存器执行读访问时，会将 RXUD 标志清零。

27. 5. 13. 10 接收FIFO缓存阈值(RXTH)

此标志置 1 时，表示接收 FIFO 缓存中的有效数据个数大于或者等于 **SPI_CON2**.RXFTH 设置的值，此时对 **SPI_DATA** 寄存器执行读访问读取接收 FIFO 缓存中的数据。当读取到 FIFO 缓存中的有效数据个数少于 **SPI_CON2**.RXFTH 设置的值时，会将 RXTH 标志清零。

27. 5. 13. 11 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。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机模式系统中发生写冲突。

在以下情况硬件将清零该标志:

- 传输完成时(主机模式下的连续通信除外)
- 关闭 SPI 时
- 发生模式错误时(**SPI_RIF**.MODF=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，BUSY 标志在所有传输期间均保持高电平。

注意:请勿使用 BUSY 标志处理每次数据发送或接收，最好改用 TXTH 标志和 RXTH 标志。

27. 5. 14 SPI中断事件

27. 5. 14. 1 发送FIFO缓存为空(TXE)

当发送 FIFO 缓存为空(**SPI_STAT**.TXE=1)时，**SPI_RIF** 寄存器的 TXE 位会被设置为 1。如果 **SPI_IER** 寄存器中的 TXE 位置 1 则产生中断。通过对 **SPI_ICR** 寄存器中的 TXE 位置 1，会将 **SPI_RIF** 寄存器的 TXE 位清零并清除中断。

27. 5. 14. 2 发送FIFO缓存上溢(TXOV)

当发送 FIFO 缓存已满(**SPI_STAT**.TXF=1)时，用户对 **SPI_DATA** 寄存器执行写操作。在这种情况下，**SPI_RIF** 寄存器的 TXOV 位会被设置为 1，如果 **SPI_IER** 寄存器中的 TXOV 位置 1 则产生中断。通过对 **SPI_ICR** 寄存器中的 TXOV 位置 1，会将 **SPI_RIF** 寄存器的 TXOV 位清

零并清除中断。

27. 5. 14. 3 发送FIFO缓存下溢(TXUD)

在从机模式下当发送 FIFO 缓存为空时，但主机提出数据请求。在这种情况下，**SPI_RIF** 寄存器的 TXUD 位会被设置为 1，如果 **SPI_IER** 寄存器中的 TXUD 位置 1 则产生中断。通过对 **SPI_ICR** 寄存器中的 TXUD 位置 1，会将 **SPI_RIF** 寄存器的 TXUD 位清零并清除中断。

27. 5. 14. 4 发送FIFO缓存阈值(TXTH)

当 **SPI_STAT**.TXTH=1 时，**SPI_RIF** 寄存器的 TXTH 位会被设置为 1。如果 **SPI_IER** 寄存器中的 TXTH 位置 1 则产生中断，通过对 **SPI_ICR** 寄存器中的 TXE 位置 1，会将 **SPI_RIF** 寄存器的 TXTH 位清零并清除中断。

27. 5. 14. 5 接收FIFO缓存为非空(RXNE)

当 **SPI_STAT**.RXNE=1 时，**SPI_RIF** 寄存器的 RXNE 位会被设置为 1。如果 **SPI_IER** 寄存器中的 RXNE 位置 1 则产生中断，通过对 **SPI_ICR** 寄存器中的 RXNE 位置 1，会将 **SPI_RIF** 寄存器的 RXNE 位清零并清除中断。

27. 5. 14. 6 接收FIFO缓存为满(RXF)

当 **SPI_STAT**.RXF=1 时，**SPI_RIF** 寄存器的 RXF 位会被设置为 1。如果 **SPI_IER** 寄存器中的 RXF 位置 1 则产生中断。通过对 **SPI_ICR** 寄存器中的 RXF 位置 1，会将 **SPI_RIF** 寄存器的 RXF 位清零并清除中断。

27. 5. 14. 7 接收FIFO缓存上溢(RXOV)

当接收 FIFO 缓存已满时，下一个接收的数据帧不会加载到接收 FIFO 缓存中。在这种情况下，**SPI_RIF** 寄存器的 RXOV 位会被设置为 1，如果 **SPI_IER** 寄存器中的 RXOV 位置 1 则产生中断。通过对 **SPI_ICR** 寄存器中的 RXOV 位置 1，会将 **SPI_RIF** 寄存器的 RXOV 位清零并清除中断。

27. 5. 14. 8 接收FIFO缓存下溢(RXUD)

当接收 FIFO 缓存为空时，但用户对 **SPI_DATA** 寄存器执行读访问。在这种情况下，**SPI_RIF** 寄存器的 RXUD 位会被设置为 1，如果 **SPI_IER** 寄存器中的 RXUD 位置 1 则产生中断。通过对 **SPI_ICR** 寄存器中的 RXUD 位置 1，会将 **SPI_RIF** 寄存器的 RXUD 位清零并清除中断。

27. 5. 14. 9 接收FIFO缓存阈值(RXTH)

当 **SPI_STAT**.RXTH=1 时，**SPI_RIF** 寄存器的 RXTH 位会被设置为 1。如果 **SPI_IER** 寄存器中的 RXTH 位置 1 则产生中断，通过对 **SPI_ICR** 寄存器中的 RXTH 位置 1，会将 **SPI_RIF** 寄存器的 RXTH 位清零并清除中断。

27. 5. 14. 10 CRC错误(CRCERR)

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_RIF** 寄存器中的 CRCERR 位将置 1。如果 **SPI_IER** 寄存器中的 CRCERR 位置 1 则产生中断。通过对 **SPI_ICR** 寄存器中的 CRCERR 位置 1，会将 **SPI_RIF** 寄存器的 CRCERR 位清零并清除中断。

27. 5. 14. 11 模式故障(MODF)

当主机的 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_RIF**.MODFRI 位置 1。模式错误会在以下几方面影响 SPI 外设:

- 如果 **SPI_IER** 寄存器中的 MODF 位置 1 则产生中断。
- **SPI_CON1**.SPIEN 位清零。这将关闭所有输出，并关闭 SPI 接口。
- **SPI_CON1**.MSTREN 位清零，从而强制 SPI 进入从机模式。

通过对 **SPI_ICR** 寄存器中的 MODF 位置 1，会将 **SPI_RIF** 寄存器的 MODF 位清零并清除中断。

为避免包含多个 MCU 的系统中发生多从机模式冲突，必须在 **SPI_RIF**.MODF 位清零前将 NSS 引脚拉高。在 **SPI_RIF**.MODF 位清零后可以将 **SPI_CON1**.SPIEN 和 **SPI_CON1**.MSTREN 位恢复到原始状态。

硬件不允许在 **SPI_RIF**.MODF 位为 1 时将 **SPI_CON1**.SPIEN 和 **SPI_CON1**.MSTREN 位置 1。

在多主机模式配置中，可在 **SPI_RIF**.MODF 位为 1 时处于从机模式。在这种情况下，**SPI_RIF**.MODF 位指示系统控制可能存在多主机模式冲突。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。

27. 5. 14. 12 TI模式帧格式错误(FRE)

如果 SPI 在从机模式下工作，并配置为符合 TI 模式协议，则在持续通信期间出现 NSS 脉冲时，将检测到 TI 模式帧格式错误。出现此错误时，**SPI_RIF** 寄存器中的 FRE 位将置 1。发生错误时不会关闭 SPI，但会忽略 NSS 脉冲，并且 SPI 会等待至下一个 NSS 脉冲，然后再开始新的传输。由于错误检测可能导致丢失两个数据字节，因此数据可能会损坏。

如果 **SPI_IER** 寄存器中的 FRE 位置 1，则检测到帧格式错误时将产生中断。在这种情况下，由于无法保证数据的连续性，应关闭 SPI 并在重新使能 SPI 后，由主机重新发起通信。

通过对 **SPI_ICR** 寄存器中的 FRE 位置 1，会将 **SPI_RIF** 寄存器的 FRE 位清零并清除中断。

27. 5. 15 SPI TI模式

SPI 接口与 TI 协议兼容。**SPI_CON2** 寄存器的 FRF 位可用于配置 SPI 以符合此协议。

无论 **SPI_CON1** 寄存器中设置的值如何，都必须强制时钟极性和相位符合 TI 协议要求。NSS 管理也特定于 TI 协议，在这种情况下无法通过 **SPI_CON1** 和 **SPI_CON2** 寄存器(SSEN、SSOUT、NSSOE)配置 NSS 管理。

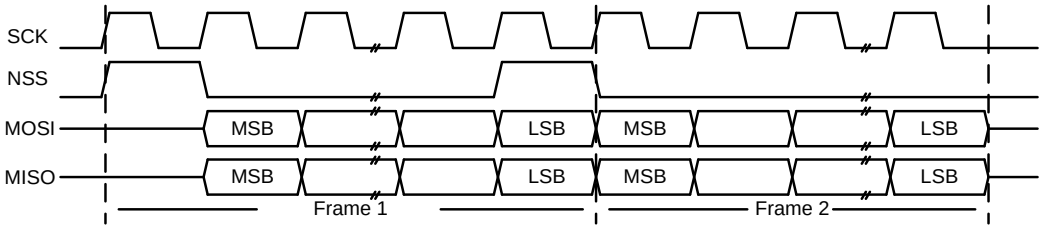


图 28-17　TI 格式

27. 6 特殊功能寄存器

27. 6. 1 寄存器列表

SPI 寄存器列表			
名称	偏移地址	类型	描述
SPI_CON1	0000_H	R/W	SPI 控制寄存器 1
SPI_CON2	0004_H	R/W	SPI 控制寄存器 2
SPI_STAT	0008_H	R	SPI 状态寄存器
SPI_DATA	$000C_H$	R/W	SPI 数据寄存器
SPI_CRCPOLY	0010_H	R/W	SPI CRC 多项式寄存器
SPI_RXCRC	0014_H	R	SPI RX CRC 寄存器
SPI_TXCRC	0018_H	R	SPI TX CRC 寄存器
SPI_IER	0024_H	W1	SPI 中断开启寄存器
SPI_IDR	0028_H	W1	SPI 中断关闭寄存器
SPI_IVS	$002C_H$	R	SPI 中断功能有效状态寄存器
SPI_RIF	0030_H	R	SPI 原始中断状态寄存器
SPI_IFM	0034_H	R	SPI 中断标志位状态寄存器
SPI_ICR	0038_H	C_W1	SPI 中断清除寄存器

27. 6. 2 寄存器描述

27. 6. 2. 1 SPI控制寄存器 1(SPI_CON1)

SPI 控制寄存器 1 (SPI_CON1)																												
偏移地址:0x000																												
复位值:0x0000 0000																												
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
																BIDEN	BIDOEN	CRCEN	NXTCRC	FLEN	RXO	SSEN	SSOUT	LSBFST	SPIEN	BAUD<2:0>			MSTREN	CPOL	CPHA

—	Bits 31-16	—	—
BIDEN	Bit 15	R/W	**双向通信使能** 该位使能使用单线双向数据线实现半双工通信。当选择半双工通信模式时，保持 RXO 位清零。 0:选择全双工通信数据模式 1:选择半双工通信数据模式
BIDOEN	Bit 14	R/W	**双向通信输出使能** 此位结合 BIDEN 位，用于选择半双工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:在主机模式下，使用 MOSI 引脚，在从机模式，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: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:数据传输结束时不传输 CRC 1:数据传输结束时传输 CRC 注:SPI 配置为全双工或只发送模式时，在写入最后一个数据前将该位置 1。当 SPI 配置为只接收模式时，必须在读取最后一个数据数据前将该位置 1。
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: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。此位也适用于多从机模式系统，在此类系统中，关闭未被访问的从机输出，以防止被访问的从机输出被其他从机干扰。 0:全双工(发送和接收) 1:关闭输出(只接收模式)
SSEN	Bit 9	R/W	**软件控制从机使能** 当 SSEN 位置 1 时，NSS 引脚输入将被 SSOUT 位的值替换。 0:禁止软件控制从机 1:使能软件控制从机 注:该位不用于 SPI TI 模式。
SSOUT	Bit 8	R/W	**软件控制片选输出** 0: NSS 引脚输入为 0 1: NSS 引脚输入为 1 仅当 SSEN 位置 1 时该位才有效。此位的值将作用到 NSS 引脚上，并忽略 NSS 引脚的 IO 值。 注:该位不用于 SPI TI 模式。
LSBFST	Bit 7	R/W	**先发最低有效位** 0:使用 MSB 发送与接收数据 1:使用 LSB 发送与接收数据 注: 1. 正在通信时不应更改此位。 2. 该位不用于 SPI TI 模式。
SPIEN	Bit 6	R/W	**SPI 模块使能** 0:关闭 SPI 外设 1:使能 SPI 外设
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:正在通信时不应更改此位。
MSTREN	Bit 2	R/W	**主机模式使能** 0:从机配置 1:主机配置 注: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: 1. 正在通信时不应更改此位。 2. 该位不用于 SPI TI 模式。
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: 1. 正在通信时不应更改此位。 2. 该位不用于 SPI TI 模式。

27. 6. 2. 2 SPI控制寄存器 2(SPI_CON2)

SPI 控制寄存器 2 (SPI_CON2)																													
偏移地址:0x004																													
复位值:0x0000 0000																													
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
																RXFTH<1:0>		TXFTH<1:0>									FRF	NSSP	NSSOE	TXDMA	RXDMA

—	Bits 31-16	—	—
RXFTH	Bit 15-14	R/W	**接收 FIFO 阈值** 用于选择接收 FIFO 缓存的阈值。 00:接收 FIFO 中有 1 笔数据 01:接收 FIFO 中数据达到其深度的 1/4 10:接收 FIFO 中数据达到其深度的 1/2 11:接收 FIFO 中数据再 2 笔达到其深度 注:如果 RXDMA 使能，该位也提供接收 FIFO 缓存的 DMA 请求条件。
TXFTH	Bit 13-12	R/W	**发送 FIFO 阈值** 用于选择发送 FIFO 缓存的阈值。 00:发送 FIFO 内无数据 01:发送 FIFO 中数据少于或等于 2 笔 10:发送 FIFO 中数据少于或等于其深度的 1/4 11:发送 FIFO 中数据少于或等于其深度的 1/2 注: 如果 TXDMA 使能，该位也提供发送 FIFO 缓存的 DMA 请求条件。
—	Bit 11-5	—	—
FRF	Bit 4	R/W	**帧格式选择** 0:SPI 摩托罗拉模式 1:SPI TI 模式 注:只有在禁止 SPI(SPIEN = 0)时才能写入该位。
NSSP	Bit 3	R/W	**NSS 脉冲管理** 该位仅用于主机模式。它允许 SPI 在进行连续传输时在两个连续数据之间产生 NSS 脉冲。在单次数据传输的情况下，它会在传输后强制 NSS 引脚为高电平。如果 CPHA ='1’ 或 FRF

			='1’则没有意义。 0:没有 NSS 脉冲 1:产生 NSS 脉冲 注: 1. 只有在禁止SPI(SPIEN = 0)时才能写入该位。 2. 该位不用于 SPI TI 模式。
NSSOE	Bit 2	R/W	**NSS 引脚输出使能** 0:在主机模式下禁止 NSS 输出，可在多主机模式配置下工作 1:在主机模式下使能 NSS 输出，不能在多主机模式环境下工作 注:该位不用于 SPI TI 模式。
TXDMA	Bit 1	R/W	**发送 FIFO 缓存 DMA 使能** 该位置 1 时，只要 TXTH 标志置 1，就会产生 DMA 请求。 0:关闭发送 FIFO 缓存的 DMA 1:使能发送 FIFO 缓存的 DMA
RXDMA	Bit 0	R/W	**接收 FIFO 缓存 DMA 使能** 该位置 1 时，只要 RXTH 标志置 1，就会产生 DMA 请求。 0:关闭接收 FIFO 缓存的 DMA 1:使能接收 FIFO 缓存的 DMA

27. 6. 2. 3 SPI状态寄存器(SPI_STAT)

SPI 状态寄存器(SPI_STAT)																														
偏移地址:0x008																														
复位值:0x0000 0011																														
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
			RXFLV<4:0>								TXFLV<4:0>					BUSY			RXTH	RXUD	RXOV	RXF	RXNE				TXTH	TXUD	TXOV	TXF	TXE

—	Bit 31-29	—	—
RXFLV	Bit 28-24	R	**接收 FIFO 缓存数据个数** 该位域表明接收 FIFO 缓存的有效数据个数。
—	Bit 23-21	—	—
TXFLV	Bit 20-16	R	**发送 FIFO 缓存数据个数** 该位域表明发送 FIFO 缓存的有效数据个数。
BUSY	Bit 15	R	**忙标志位** 0:SPI 不繁忙 1:SPI 忙于通信 此标志位由硬件置 1 和清零。
—	Bit 14-13	—	—
RXTH	Bit 12	R	**接收FIFO缓存个数超出阈值** 0:接收FIFO缓存的有效数据个数少于RXFTH设置的值 1:接收FIFO缓存的有效数据个数大于或等于RXFTH设置的值
RXUD	Bit 11	R/C_R	**接收FIFO缓存下溢** 0:接收FIFO缓存未发生下溢 1:接收FIFO缓存发生下溢 注:读取STAT寄存器清除此位。
RXOV	Bit 10	R/C_R	**接收FIFO缓存上溢** 0:接收FIFO缓存未发生上溢 1:接收FIFO缓存发生上溢 注:读取STAT寄存器清除此位。
RXF	Bit 9	R	**接收FIFO缓存满** 0:接收FIFO缓存未满 1:接收FIFO缓存已满
RXNE	Bit 8	R	**接收FIFO缓存非空**

			0:接收FIFO缓存为空 1:接收FIFO缓存为非空
—	Bit 7-5	—	—
TXTH	Bit 4	R	**发送FIFO缓存个数低于阈值** 0:发送FIFO缓存的有效数据个数大于TXFTH设置的值 1:发送FIFO缓存的有效数据个数少于或等于TXFTH设置的值
TXUD	Bit 3	R/C_R	**发送FIFO缓存下溢** 0:发送FIFO缓存未发生下溢 1:发送FIFO 缓存发生下溢 注:读取STAT寄存器清除此位。
TXOV	Bit 2	R/C_R	**发送FIFO缓存上溢** 0:发送FIFO缓存未发生上溢 1:发送FIFO缓存发生上溢 注:读取STAT寄存器清除此位。
TXF	Bit 1	R	**发送FIFO缓存满** 0:发送FIFO缓存未满 1:发送FIFO缓存已满
TXE	Bit 0	R	**发送FIFO缓存空** 0:发送FIFO缓存非空 1:发送FIFO缓存为空

27. 6. 2. 4 SPI数据寄存器(SPI_DATA)

SPI 数据寄存器(SPI_DATA)																													
偏移地址:0x00C																													
复位值:0x0000 0000																													
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<15:0>															

—	Bits 31-16	—	—
DATA	Bits 15-0	R/W	**数据寄存器** 已接收或者要发送的数据。 数据寄存器分为2个FIFO缓存，一个用于写入(发送FIFO缓存)，一个用于读取(接收FIFO缓存)。对数据寄存器执行写操作时，数据将写入发送FIFO缓存，从数据寄存器执行读取时，将返回接收FIFO缓存中的值。 注:数据始终是右对齐的。写入寄存器时忽略未使用的位，读取寄存器时未使用的位数据为0。

27. 6. 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:0x010																													
复位值:0x0000 0007																													
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
																CRCPOLY<15:0>															

—	Bits 31-16	—	—
CRCPOLY	Bits 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。 注:多项式值应仅为奇数。没有支持偶数。

27. 6. 2. 6 SPI RX CRC寄存器(SPI_RXCRC)

SPI RX CRC 寄存器(SPI_RXCRC)																												
偏移地址:0x014																												
复位值:0x0000 0000																												
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
																RXCRC<15:0>															

—	Bits 31-16	—	—
RXCRC	Bits 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 寄存器的 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:当 BUSY 标志位置 1 时，读取此寄存器可能返回一个不正确的值。

27. 6. 2. 7 SPI TX CRC寄存器(SPI_TXCRC)

SPI TX CRC 寄存器(SPI_TXCRC)																													
偏移地址:0x018																													
复位值:0x0000 0000																													
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
																TXCRC<15:0>															

—	Bits 31-16	—	—
TXCRC	Bits 15-0	R	**发送 CRC 值** 使能 CRC 计算后，TXCRC[15:0] 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 寄存器的 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:当 BUSY 标志位置 1 时，读取此寄存器可能返回一个不正确的值。

27. 6. 2. 8 SPI中断开启寄存器(SPI_IER)

SPI 中断开启寄存器 (SPI_IER)																														
偏移地址:0x024																														
复位值:0x0000 0000																														
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
													FRE	MODF	CRCERR				RXTH	RXUD	RXOV	RXF	RXNE				TXTH	TXUD	TXOV		TXE

—	Bits 31-19	—	—
FRE	Bit 18	W1	**帧格式错误中断** 0:写入 0 无效 1:开启帧格式错误中断
MODF	Bit 17	W1	**模式故障中断** 0:写入 0 无效 1:开启模式故障中断
CRCERR	Bit 16	W1	**CRC 错误中断** 0:写入 0 无效 1:开启 CRC 错误中断
—	Bit 15-13	—	—
RXTH	Bit 12	W1	**接收 FIFO 缓存超过阈值中断** 0:写入 0 无效 1:开启接收 FIFO 缓存超过阈值中断
RXUD	Bit 11	W1	**接收 FIFO 缓存下溢中断** 0:写入 0 无效 1:开启接收 FIFO 缓存下溢中断
RXOV	Bit 10	W1	**接收 FIFO 缓存上溢中断** 0:写入 0 无效 1:开启接收 FIFO 缓存上溢中断
RXF	Bit 9	W1	**接收 FIFO 缓存满中断** 0:写入 0 无效 1:开启接收 FIFO 缓存满中断
RXNE	Bit 8	W1	**接收 FIFO 缓存非空中断** 0:写入 0 无效 1:开启接收 FIFO 缓存非空中断
—	Bit 7-5	—	—
TXTH	Bit 4	W1	**发送 FIFO 缓存低于阈值中断**

			0:写入 0 无效 1:开启发送 FIFO 缓存低于阈值中断
TXUD	Bit 3	W1	**发送 FIFO 缓存下溢中断** 0:写入 0 无效 1:开启发送 FIFO 缓存下溢中断
TXOV	Bit 2	W1	**发送FIFO缓存上溢中断** 0:写入0无效 1:开启发送FIFO缓存上溢中断
—	Bit 1	—	—
TXE	Bit 0	W1	**发送 FIFO 缓存空中断** 0:写入 0 无效 1:开启发送 FIFO 缓存空中断

27. 6. 2. 9 SPI中断关闭寄存器(SPI_IDR)

SPI 中断关闭寄存器(SPI_IDR)																												
偏移地址:0x028																												
复位值:0x0000 0000																												
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
—	—	—	—	—	—	—	—	—	—	—	—	—	FRE	MODF	CRCERR	—	—	—	RXTH	RXUD	RXOV	RXF	RXNE	—	—	—	TXTH	TXUD	TXOV	—	TXE

—	Bit 31-19	—	—
FRE	Bit 18	W1	**帧格式错误中断** 0:写入0无效 1:关闭帧格式错误中断
MODF	Bit 17	W1	**模式故障中断** 0:写入 0 无效 1:关闭模式故障中断
CRCERR	Bit 16	W1	**CRC 错误中断** 0:写入 0 无效 1:关闭 CRC 错误中断
—	Bit 15-13	—	—
RXTH	Bit 12	W1	**接收 FIFO 缓存超过阈值中断** 0:写入 0 无效 1:关闭接收 FIFO 缓存超过阈值中断
RXUD	Bit 11	W1	**接收 FIFO 缓存下溢中断**

			0:写入 0 无效 1:关闭接收 FIFO 缓存下溢中断
RXOV	Bit 10	W1	**接收 FIFO 缓存上溢中断** 0:写入 0 无效 1:关闭接收 FIFO 缓存上溢中断
RXF	Bit 9	W1	**接收 FIFO 缓存满中断** 0:写入 0 无效 1:关闭接收 FIFO 缓存满中断
RXNE	Bit 8	W1	**接收FIFO缓存非空中断** 0:写入0无效 1:关闭接收FIFO缓存非空中断
—	Bit 7-5	—	—
TXTH	Bit 4	W1	**发送 FIFO 缓存低于阈值中断** 0:写入 0 无效 1:关闭发送 FIFO 缓存低于阈值中断
TXUD	Bit 3	W1	**发送FIFO缓存下溢中断** 0:写入0无效 1:关闭发送FIFO缓存下溢中断
TXOV	Bit 2	W1	**发送 FIFO 缓存上溢中断** 0:写入 0 无效 1:关闭发送 FIFO 缓存上溢中断
—	Bit 1	—	—
TXE	Bit 0	W1	**发送 FIFO 缓存空中断** 0:写入 0 无效 1:关闭发送 FIFO 缓存空中断

27. 6. 2. 10 SPI中断功能有效状态寄存器(SPI_IVS)

SPI 中断功能有效状态寄存器 (SPI_IVS)																														
偏移地址:0x02C																														
复位值:0x0000 0000																														
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
—	—	—	—	—	—	—	—	—	—	—	—	—	FRE	MODF	CRCERR	—	—	—	RXTH	RXUD	RXOV	RXF	RXNE	—	—	—	TXTH	TXUD	TXOV	—	TXE

—	Bits 31-19	—	—
FRE	Bit 18	R	**帧格式错误中断功能状态** 0:禁止帧格式错误中断 1:使能帧格式错误中断
MODF	Bit 17	R	**模式故障中断功能状态** 0:禁止模式故障中断 1:使能模式故障中断
CRCERR	Bit 16	R	**CRC错误中断功能状态** 0:禁止CRC错误中断 1:使能CRC错误中断
—	Bit 15-13	—	—
RXTH	Bit 12	R	**接收 FIFO 缓存超过阈值中断功能状态** 0:禁止接收 FIFO 缓存超过阈值中断 1:使能接收 FIFO 缓存超过阈值中断
RXUD	Bit 11	R	**接收 FIFO 缓存下溢中断功能状态** 0:禁止接收 FIFO 缓存下溢中断 1:使能接收 FIFO 缓存下溢中断
RXOV	Bit 10	R	**接收 FIFO 缓存上溢中断功能状态** 0:禁止接收 FIFO 缓存上溢中断 1:使能接收 FIFO 缓存上溢中断
RXF	Bit 9	R	**接收 FIFO 缓存满中断功能状态** 0:禁止接收 FIFO 缓存满中断 1:使能接收 FIFO 缓存满中断
RXNE	Bit 8	R	**接收 FIFO 缓存非空中断功能状态** 0:禁止接收 FIFO 缓存非空中断 1:使能接收 FIFO 缓存非空中断
—	Bit 7-5	—	—
TXTH	Bit 4	R	**发送 FIFO 缓存低于阈值中断功能状态**

			0:禁止发送 FIFO 缓存低于阈值中断 1:使能发送 FIFO 缓存低于阈值中断
TXUD	Bit 3	R	**发送 FIFO 缓存下溢中断功能状态** 0:禁止发送 FIFO 缓存下溢中断 1:使能发送 FIFO 缓存下溢中断
TXOV	Bit 2	R	**发送FIFO缓存上溢中断功能状态** 0:禁止发送FIFO缓存上溢中断 1:使能发送FIFO缓存上溢中断
—	Bit 1	—	—
TXE	Bit 0	R	**发送 FIFO 缓存空中断功能状态** 0:禁止发送 FIFO 缓存空中断 1:使能发送 FIFO 缓存空中断

27. 6. 2. 11 SPI原始中断状态寄存器 (SPI_RIF)

SPI 原始中断状态寄存器 (SPI_RIF)

偏移地址:0x030

复位值:0x0000 0000

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
—	—	—	—	—	—	—	—	—	—	—	—	—	FRE	MODF	CRCERR	—	—	—	RXTH	RXUD	RXOV	RXF	RXNE	—	—	—	TXTH	TXUD	TXOV	—	TXE

—	Bits 31-19	—	—
FRE	Bit 18	R	**帧格式错误，原始中断状态** 0:未发生中断事件 1:发生中断事件
MODF	Bit 17	R	**模式故障，原始中断状态** 0:未发生中断事件 1:发生中断事件
CRCERR	Bit 16	R	**CRC错误，原始中断状态** 0:未发生中断事件 1:发生中断事件
—	Bit 15-13	—	—
RXTH	Bit 12	R	**接收 FIFO 缓存超过阈值，原始中断状态** 0:未发生中断事件 1:发生中断事件
RXUD	Bit 11	R	**接收 FIFO 缓存下溢，原始中断状态**

			0:未发生中断事件 1:发生中断事件
RXOV	Bit 10	R	**接收 FIFO 缓存上溢，原始中断状态** 0:未发生中断事件 1:发生中断事件
RXF	Bit 9	R	**接收 FIFO 缓存满，原始中断状态** 0:未发生中断事件 1:发生中断事件
RXNE	Bit 8	R	**接收 FIFO 缓存非空，原始中断状态** 0:未发生中断事件 1:发生中断事件
—	Bit 7-5	—	—
TXTH	Bit 4	R	**发送 FIFO 缓存低于阈值，原始中断状态** 0:未发生中断事件 1:发生中断事件
TXUD	Bit 3	R	**发送 FIFO 缓存下溢，原始中断状态** 0:未发生中断事件 1:发生中断事件
TXOV	Bit 2	R	**发送FIFO缓存上溢，原始中断状态** 0:未发生中断事件 1:发生中断事件
—	Bit 1	—	—
TXE	Bit 0	R	**发送 FIFO 缓存空，原始中断状态** 0:未发生中断事件 1:发生中断事件

27. 6. 2. 12 SPI中断标志位状态寄存器(SPI_IFM)

SPI 中断标志位状态寄存器 (SPI_IFM)																														
偏移地址:0x034																														
复位值:0x0000 0000																														
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
													FRE	MODF	CRCERR				RXTH	RXUD	RXOV	RXF	RXNE				TXTH	TXUD	TXOV		TXE

—	Bits 31-19	—	—
FRE	Bit 18	R	**帧格式错误中断标志位状态** 0:未发生中断事件或中断未使能 1:产生中断
MODF	Bit 17	R	**模式故障中断标志位状态** 0:未发生中断事件或中断未使能 1:产生中断
CRCERR	Bit 16	R	**CRC错误中断标志位状态** 0:未发生中断事件或中断未使能 1:产生中断
—	Bit 15-13	—	—
RXTH	Bit 12	R	**接收 FIFO 缓存超过阈值中断标志位状态** 0:未发生中断事件或中断未使能 1:产生中断
RXUD	Bit 11	R	**接收 FIFO 缓存下溢中断标志位状态** 0:未发生中断事件或中断未使能 1:产生中断
RXOV	Bit 10	R	**接收 FIFO 缓存上溢中断标志位状态** 0:未发生中断事件或中断未使能 1:产生中断
RXF	Bit 9	R	**接收 FIFO 缓存满中断标志位状态** 0:未发生中断事件或中断未使能 1:产生中断
RXNE	Bit 8	R	**接收 FIFO 缓存非空中断标志位状态** 0:未发生中断事件或中断未使能 1:产生中断
—	Bit 7-5	—	—
TXTH	Bit 4	R	**发送 FIFO 缓存低于阈值中断标志位状态**

			0:未发生中断事件或中断未使能 1:产生中断
TXUD	Bit 3	R	**发送 FIFO 缓存下溢中断标志位状态** 0:未发生中断事件或中断未使能 1:产生中断
TXOV	Bit 2	R	**发送FIFO缓存上溢中断标志位状态** 0:未发生中断事件或中断未使能 1:产生中断
—	Bit 1	—	—
TXE	Bit 0	R	**发送 FIFO 缓存空中断标志位状态** 0:未发生中断事件或中断未使能 1:产生中断

27. 6. 2. 13 SPI中断清除寄存器 (SPI_ICR)

SPI 中断清除寄存器 (SPI_ICR)

偏移地址:0x038

复位值:0x0000 0000

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
—	—	—	—	—	—	—	—	—	—	—	—	—	FRE	MODF	CRCERR	—	—	—	RXTH	RXUD	RXOV	RXF	RXNE	—	—	—	TXTH	TXUD	TXOV	—	TXE

—	Bits 31-19	—	—
FRE	Bit 18	C_W1	**帧格式错误中断清除** 0:写入0无效 1:清除中断事件与中断
MODF	Bit 17	C_W1	**模式故障中断清除** 0:写入 0 无效 1:清除中断事件与中断
CRCERR	Bit 16	C_W1	**CRC错误中断清除** 0:写入0无效 1:清除中断事件与中断
—	Bit 15-13	—	—
RXTH	Bit 12	C_W1	**接收 FIFO 缓存超过阈值中断清除** 0:写入 0 无效 1:清除中断事件与中断
RXUD	Bit 11	C_W1	**接收 FIFO 缓存下溢中断清除**

			0:写入 0 无效 1:清除中断事件与中断
RXOV	Bit 10	C_W1	**接收 FIFO 缓存上溢中断清除** 0:写入 0 无效 1:清除中断事件与中断
RXF	Bit 9	C_W1	**接收 FIFO 缓存满中断清除** 0:写入 0 无效 1:清除中断事件与中断
RXNE	Bit 8	C_W1	**接收 FIFO 缓存非空中断清除** 0:写入 0 无效 1:清除中断事件与中断
—	Bit 7-5	—	—
TXTH	Bit 4	C_W1	**发送 FIFO 缓存低于阈值中断清除** 0:写入 0 无效 1:清除中断事件与中断
TXUD	Bit 3	C_W1	**发送 FIFO 缓存下溢中断清除** 0:写入 0 无效 1:清除中断事件与中断
TXOV	Bit 2	C_W1	**发送FIFO缓存上溢中断清除** 0:写入0无效 1:清除中断事件与中断
—	Bit 1	—	—
TXE	Bit 0	C_W1	**发送 FIFO 缓存空中断清除** 0:写入 0 无效 1:清除中断事件与中断

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:

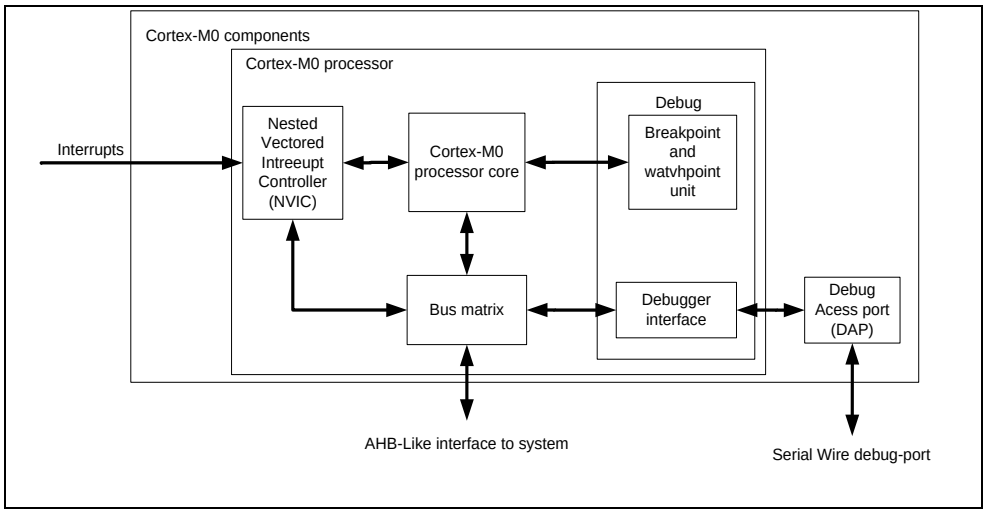
◇ 结构简单，容易学习和编程

◇ 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**。

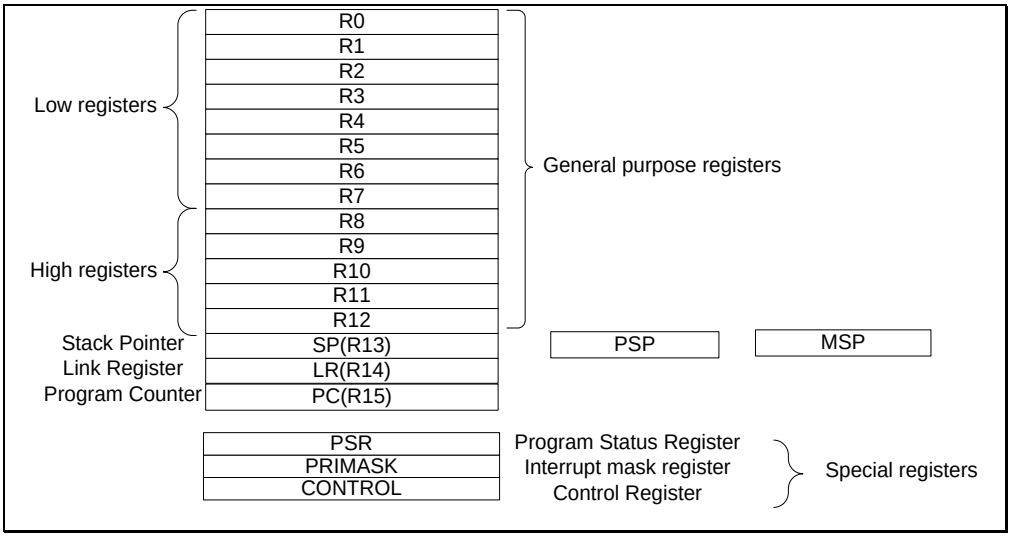
在线程模式下，CONTROL 寄存器控制着处理器使用主堆栈还是进程堆栈，见**处理器－控制寄存器章节**。在处理器模式下，处理器总是使用主堆栈。处理器操作的选择如下:

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
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

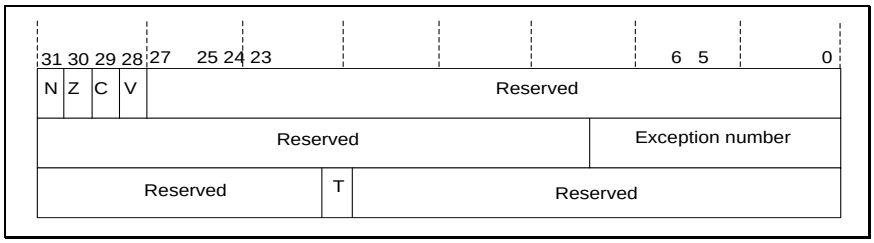
程序状态寄存器(PSR)由下列三种寄存器组合而成:

◇ 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位的当前状态。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	负值标志位
[30]	Z	零标志位
[29]	C	进位或借位标志位
[28]	V	溢出标志位
[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位的更多信息请参考条件标志位。

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”

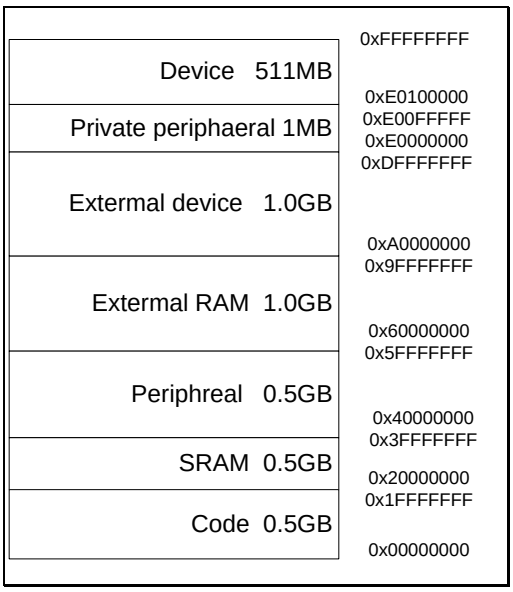
◇ 内部函数“Intrinsic functions”

◇ 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-orderedmemory)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	正常访问	设备访问		非常有序访问
		非共享	可共享	
正常访问	-	-	-	-
设备访问，非共享	-	<	-	<
设备访问，可共享	-	-	<	<
非常有序访问	-	<	<	<

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。

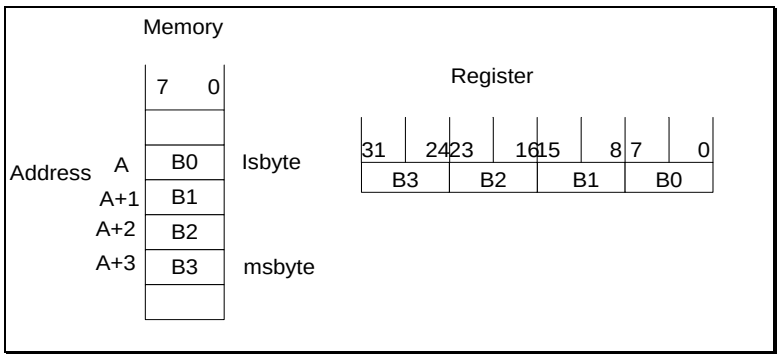
处理器保留与所有其他事务相关的事务顺序。

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	-	-

异常编号[1]	IRQ 编号[1]	异常类型	优先级	向量地址[2]
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:

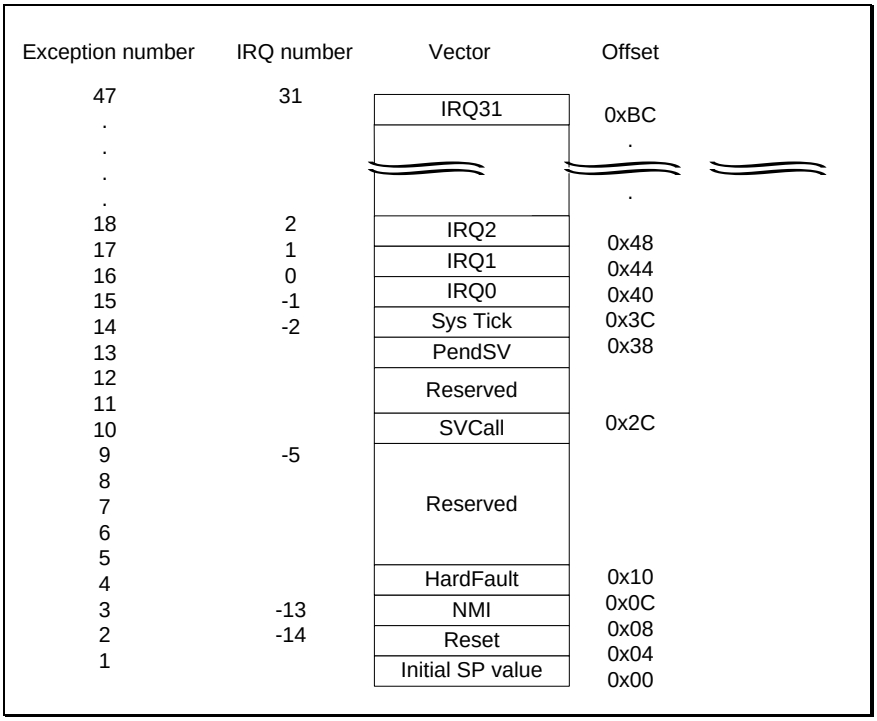
中断服务程序(ISR)—中断 IRQ0~IRQ31 是由 ISR 来处理的异常

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

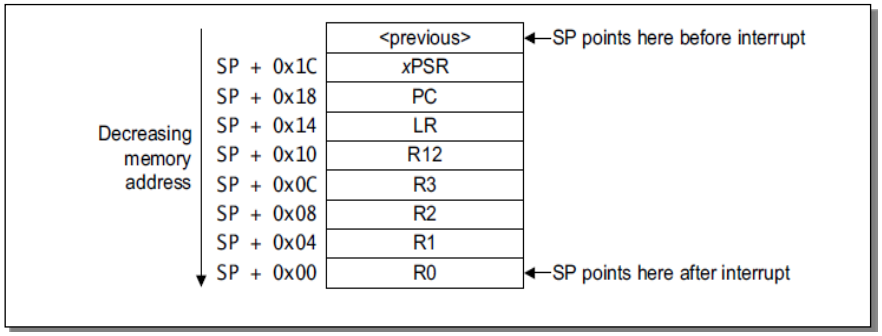
当有一个优先级足够高的挂起异常存在，并且满足下面的任何一个条件，就进入异处理:

◇　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	标志位	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	标志位	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	标志位	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

助记符	操作数	简述	标志位	参考
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
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位，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位的。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

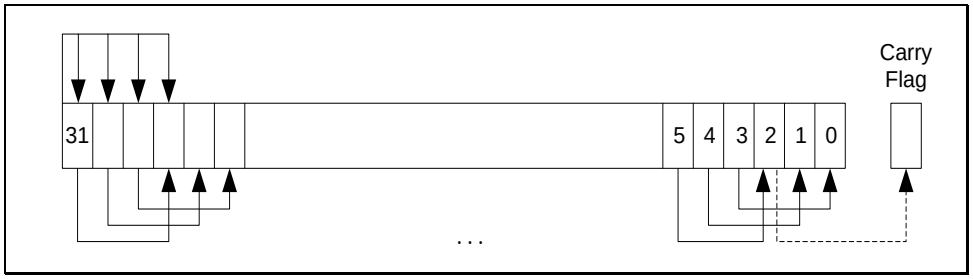
算术右移 n 位的操作是将 Rm 寄存器左边的 32-n 个位向右移动 n 位，结果是寄存器右边有 32-n 个位，然后再将寄存器位[31]的原始值复制到结果寄存器左边的 n 位中，请看图片 **ASR #3**。

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，进位标志位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，那么进位标志位被更新为 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。

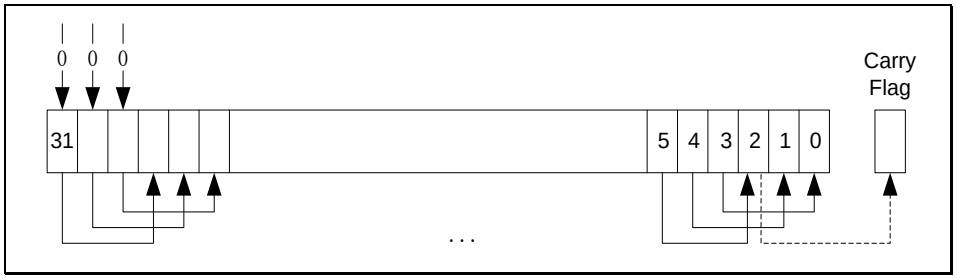
如果寄存器 Rm 值为无符号的整数，用户可以使用 LSR 操作来对其值进行除以 2^n 的操作。

当指令为 LSRS 时，进位标志位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，那么进位标志位被更新为 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。

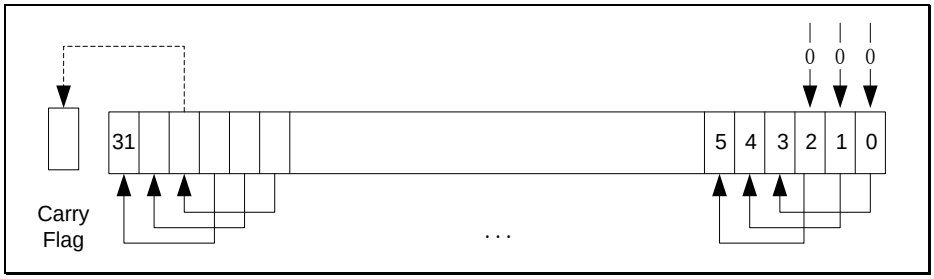
如果寄存器 Rm 值为无符号的整数或是有符号 2 的补码整数值，用户可以使用 LSL 操作来对其值与 2^n 进行乘法操作。溢出会在无警告提示下发生。

当指令为 LSLS 时，进位标志位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位。

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，那么进位标志位被更新为 0。



附录图 1-10　LSL #3

ROR

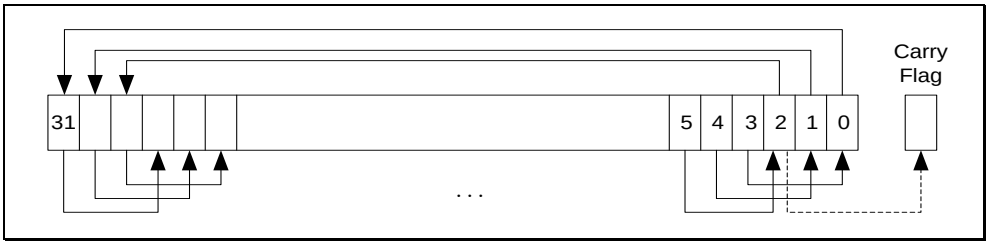
循环右移 n 位的操作是将 Rm 寄存器左边的 32-n 个位向右移动 n 位，结果是寄存器右边有 32-n 个位，然后再将寄存器右边的 n 个位移动到结果寄存器的左边的 n 个位中。请看图片 ROR #3。

当指令为 RORS 时，进位标志位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，且如果进位标志位被更新，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)中更新条件标志位。某些指

令更新所有标志位，而某些指令则仅更新子集。如果标志位不被更新，则原始值被保留。指令对标志位的影响，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位的基础上:

◇ 在指令更新标志位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位的间隔数指令后，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:

◇ **条件标志位“The condition flags”**

◇ **条件代码后缀“Condition code suffixes”**

条件标志位

APSR 包含了下列的条件标志位:

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 的条件代码标志位符合指定的条件时，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 标志位之关的联系。

后缀	标志位	意义
EQ	Z=1	相等，最后标志位设置结果为 0
NE	Z=0	不相等。最后标志位设置结果为非 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。

条件标志位

该指令不会改变标志位。

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**。

条件标志位

这些指令不改变标志位。

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**。

条件标志位

这些指令不改变标志位。

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。

条件标志位

这些指令不改变标志位。

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。

条件标志位

这些指令不改变标志位。

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

条件标志位

这些指令不改变标志位。

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 指令更新标志位

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，如果进位标志位被置位，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 标志位。

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 标志位。

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 标志位。

SBCS 指令是用 Rn 的值减去 Rm 的值，如果进位标志位置位，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 标志位。

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 标志位。

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。

条件代码标志位会根据操作的结果被更新，请看**条件标志位**。

限制

在这些指令中，Rd、Rn 和 Rm 必须指定 R0-R7。

条件标志位

这些指令会:

◇ 根据结果值来更新 N 和 Z 标志位

◇ 不会影响 C 或 V 标志位

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。

条件标志位

这些指令根据结果值来更新 N 和 Z 标志位。

C 标志位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 标志位不变。

示例

ASRS R7, R5, #9; 算术右移 9 位
LSLS R1, R2, #3; 逻辑左移 3 位，并更新标志位
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位，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，并更新标志位。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，并更新标志位。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

条件标志位

这些指令根据结果值来更新 N、Z、C 和 V 标志位。

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位，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 标志位。

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。

条件标志位

如果 S 被指定，则这些指令会:

◇ 根据结果值更新 N 和 Z 标志位

◇ 不会影响 C 或 V 标志位

示例

```
MOVS R0, #0x000B; 将 0x000B 写入 R0，更新标志位
MOVS R1, #0x0; 将 0 写入 R1，更新标志位
MOV R10, R12; 将 R12 的值写入 R10，不更新标志位
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 并更新标志位
```

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。条件代码标志位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

条件标志位

该指令会:

- 根据结果值来更新 N 和 Z 标志位
- 不会影响 C 或 V 标志位

示例

MULS R0, R2, R0; 乘法操作，标志位被更新，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。

条件标志位

这些指令不改变标志位。

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。

条件标志位

这些指令不改变标志位。

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位，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。

条件标志位

这些指令:

◇ 会根据结果来更新 N 和 Z 标志位

◇ 不会影响 C 或 V 标志位

示例

TST R0, R1; 对 R0 值和 R1 值执行位与操作，更新条件代码标志位，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。

条件标志位

这些指令不改变标志位。

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 标志位置位则跳转，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。

条件标志位

该指令不改变标志位。

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。

条件标志位

该指令不改变标志位。

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。

条件标志位

该指令不改变标志位。

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。

条件标志位

该指令不改变标志位。

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。

条件标志位

该指令不改变标志位。

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 中修改特别标志位。

请看指令 MSR。

限制

在该指令中，Rd 必须不能是 SP 或 PC。

条件标志位

该指令不改变标志位。

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。

条件标志位

该指令明确地根据 Rn 中的值来更新标志位。

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。

条件标志位

该指令不改变标志位。

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。

条件标志位

该指令不改变标志位。

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。

条件标志位

该指令不改变标志位。

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。

条件标志位

该指令不改变标志位。

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

条件标志位

该指令不改变标志位。

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

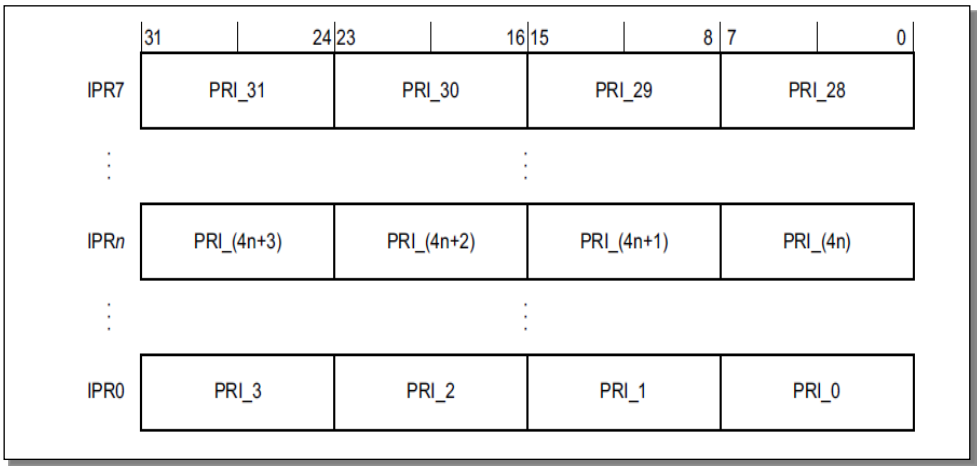
附录表 1-29　ICPR 位分配

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位 0=中断未挂起 1=中断正在挂起
[21:18]	-	-	保留

位域	名称	类型	功能
[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
	字节，直接偏移量	LDRB Rd,[Rn, #<imm>]	2

操作	描述	汇编程序	周期
	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
Hint	发送事件	SEV	1
	等待事件	WFE	2[5]

操作	描述	汇编程序	周期
	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

版本历史

版本	修改日期	更改概要
V1.0	2025-03-19	初版