32 位 MCU
ES32F0943

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

2025-04-24

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)(__O)	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) (__IO)	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) (__IO)	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) (__IO)	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) (__O)	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) (__O)	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) (__O)	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) (__O)	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

ES32F0943 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 48MHz，以及最大 128K bytes Flash 与 8K bytes 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，4 个 UART，1 个通用 32 位定时器，4 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，1 个 RTC 万年历。

ES32F0943 微控制器，具备高精度 Sigma Delta ADC，轨对轨运算放大器 OPAMP，比较器与 4X38/6X36 LCD 液晶驱动器。

ES32F0943 微控制器，工作电压为 1.8 ~ 5.5V，可操作在-40 ~ +85° C，拥有完善的低功耗模式，可应用于低功耗产品。主要用于血压计、额温枪、电字秤与高精度量测仪器等产品。

本章介绍了 ES32F0943 的特点，其系统和内存结构:

- ES32F0943 系统体系结构
- ES32F0943 系统特点
- ES32F0943 内存映射

外设		ES32F0943LV	ES32F0943LT4	ES32F0943NK
Flash (K bytes)		128		
SRAM (K bytes)		8		
GPIO		Max. 62	Max. 56	Max. 24
CALC 运算加速器		32bits 除法/开根号		
CMP		1		
LCD		4x38、6x36	4x32、6x30	—
Sigma-Delta ADC		1(8channels)+ IA 仪表放大器		1(4channels)+ IA 仪表放大器
OPAMP 轨对轨运算放大器		1		
定时器	GP32C4T	1		
	GP16C2T	4		
	BS16T	1		
	WWDT	1		
	IWDT	1		
	RTC	1		
通信接口	I2C	1		
	SPI	1		
	UART	4		
CPU 操作频率		Max. 48Mhz		
数字工作电压 (V_{DDH})		1.8V - 5.5V		
模拟工作电压 (V_{DDA})		2.2/2.4/2.7/3.0/3.3V		
LCD 工作电压(V_{LCD})		2.5~4.0V		
封装类型		LQFP80（12x12mm）	LQFP64（7x7mm）	QFN32（4x4mm）

表 1-1　系统功能

1. 2 结构图

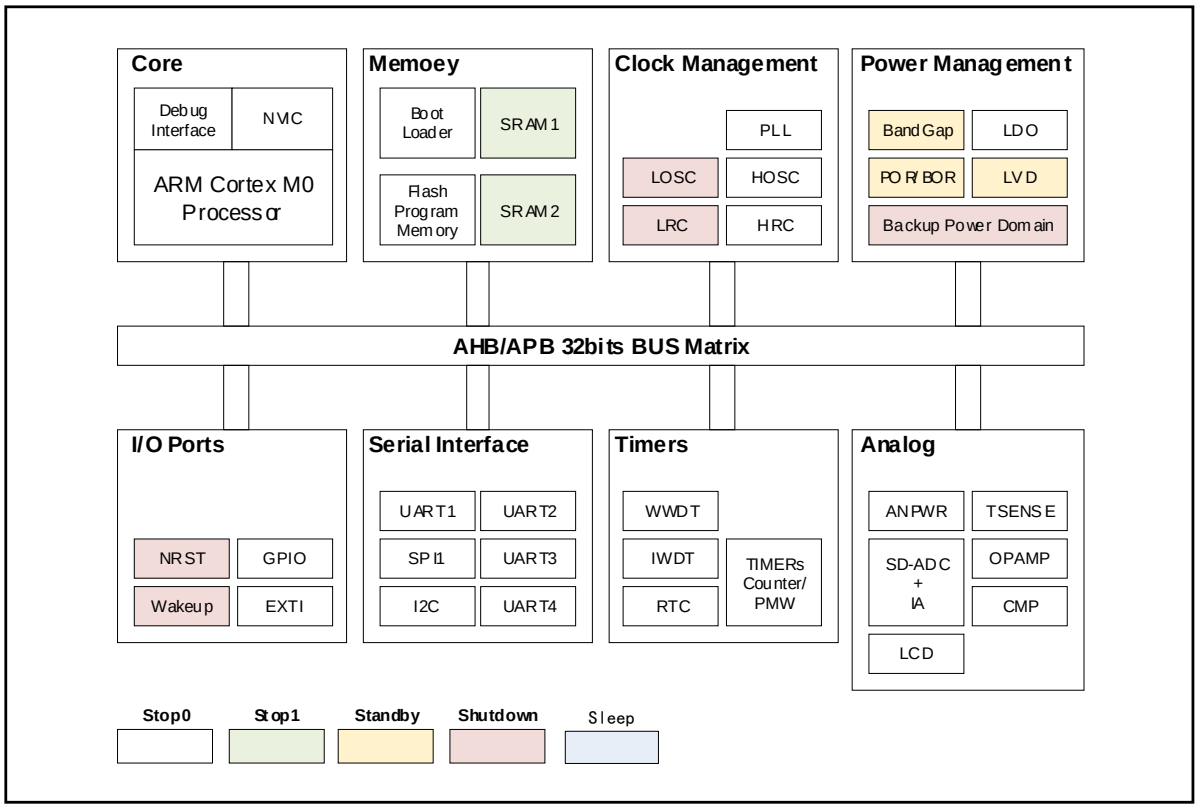


图 1-1 ES32F0943 系统框图

1. 3 功能描述

本节概述 ES32F0943 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	48MHz操作；35.2 DMIPS性能
闪存	128 KB程序或数据区
系统静态随机存取存储器	8KB的SRAM
系统	
功率控制	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	五种低功耗模式: 睡眠模式(SLEEP) 停止模式0(STOP0) 停止模式1(STOP1) 待机模式(STANDBY) 停止运转模式(SHUTDOWN)
时钟控制	32 kHz内部低速RC振荡器(LRC) 16 MHz内部高速RC振荡器(HRC) 4-32 MHz外部高速晶体振荡器(HOSC) PLL锁相回路电路，最高倍频至48MHz 用于RTC和低功耗块(LOSC)的32.768 kHz外部低速晶体振荡器
计时器	
通用定时器A型:GP32C4T **通用定时器C型:GP16C2T1，GP16C2T2，GP16C2T3，GP16C2T4** **基本定时器:BS16T**	**通用定时器A型:** 通过同步或事件链接的计时器链接功能一起工作 可以在调试模式下冻结计数器。 类型A基于32位自动重载上/下数计数器和16位预分频器。 PWM或单脉冲模式输出 4捕获或比较通道 **通用定时器C型:** 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 **基本定时器:** 16位自动重载上数计数器和16位预分频器
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

特征	描述
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
实时时钟(RTC)	睡眠模式下的支持计数 时间:小时，分钟，秒 日历:年、月、日、周、闰年 闹铃:年、月、日、周、小时、分钟、秒。(单次/周期)
输入输出端口	
通用输入输出（GPIO)	四组GPIO 外部中断能力(边沿或电平触发) 去抖动能力
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62个GPIO可以连接到16个外部中断线
模拟	
模数转换器（ADC)	支持仪表放大器增益可配置0.5/1/2/4/8/16/32/64倍 支持ADC输入增益可配置为1/2/4/8倍 支持两路差分20位AD采样有效精度 温度传感器
模拟比较器（CMP)	可编程去抖计数器 内建16节点的4位数字电阻器 支持消隐源比较器输出
轨对轨运算放大器（OPAMP)	正输入端有8个独立的选择开关，负输入端有8个独立的选择开关 内置10pF电容 可作为比较器使用，作为比较器时具有斩波器功能 内置尖峰脉冲数字低通滤波器
液晶驱动控制器（LCD)	内置倍压电路(Regulated charge pump) 16 阶可调整VLCD电压范围(2.5-4.0V) A、B驱动波型最大支持4x38或6x36的COM/SEG组合 四阶可调式驱动偏压与三种闪烁功能 LCD 控制信号数字输出
串行接口	
通用异步接收机/发射机（UART)	可编程波特率发生器 自动波特率检测 自动硬件流量控制 可编程(CTSn，RTSn)触发电平

特征	描述
内部集成电路总线 (I2C)	支持主从模式 支持不同的通信速度(100k/400k/1 MHz) 7位/ 10位寻址的产生与检测 可编程I2C地址检测 SMBus能力 PMBUS规范Rev 1.1兼容性
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 提供独立4级深度的发送和接收FIFO数据缓存
加速器	
运算加速器 (CALC)	**平方根的运算加速:** 2~17个HCLK时钟的单周期计算 **32位除法:** 有符号2的补码整数计算 32位被除数与32位除数计算能力 32位商和32位余数输出 2~17个HCLK时钟的单周期计算
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　ES32F0943 微控制器特性

1. 4 内存映射

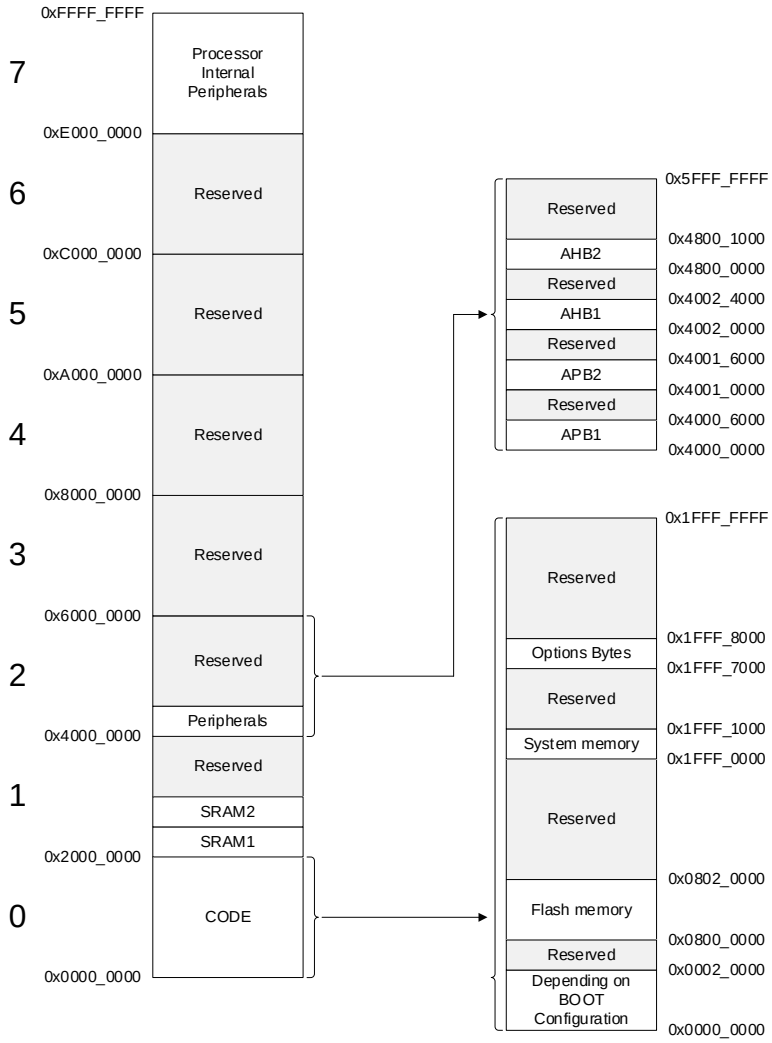


图 1-2 内存映射

边界地址		大小 (Byte)	外设	外设寄存器映象	总线
起点	终点				
0xE010_0000	0xFFFF_FFFF	-	Reserved	-	-
0xE000_0000	0xE00F_FFFF	1M	MCU 内部外设	-	MCU
0x6000_0000	0xDFFF_FFFF	-	Reserved	-	-
0x4800_1000	0x5FFF_FFFF	-	Reserved	-	AHB2
0x4800_0C00	0x4800_0FFF	1K	GPIO D	-	
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		Reserved	-	AHB1
0x4002_3C00	0x4002_3FFF	1K	CALC	-	
0x4002_2400	0x4002_3BFF	6K	Reserved	-	
0x4002_2000	0x4002_23FF	1K	FLASH	-	
0x4002_1C00	0x4002_1FFF	1K	Reserved	-	
0x4002_1800	0x4002_1BFF	1K	RTC	-	
0x4002_1400	0x4002_17FF	1K	SYSCFG+OPT	-	
0x4002_1000	0x4002_13FF	1K	RCU	-	
0x4002_0000	0x4002_0FFF	4K	Reserved	-	
0x4001_6000	0x4001_FFFF	40K	Reserved	-	APB2
0x4001_5C00	0x4001_5FFF	1K	CMP	-	
0x4001_5800	0x4001_5BFF	1K	DBGMCU	-	
0x4001_5000	0x4001_57FF	2K	Reserved	-	
0x4001_4C00	0x4001_4FFF	1K	GP16C2T4	-	
0x4001_4800	0x4001_4BFF	1K	GP16C2T3	-	
0x4001_4400	0x4001_47FF	1K	GP16C2T2	-	
0x4001_4000	0x4001_43FF	1K	GP16C2T1	-	
0x4001_3C00	0x4001_3FFF	1K	Reserved	-	
0x4001_3800	0x4001_3BFF	1K	UART1	-	
0x4001_3400	0x4001_37FF	1K	Reserved	-	
0x4001_3000	0x4001_33FF	1K	SPI1	-	
0x4001_1800	0x4001_2FFF	6K	Reserved	-	
0x4001_1400	0x4001_17FF	1K	OPAMP	-	
0x4001_1000	0x4001_13FF	1K	ANPWR	-	
0x4001_0C00	0x4001_0FFF	1K	ADC	-	
0x4001_0800	0x4001_0BFF	1K	LCD	-	
0x4001_0400	0x4001_07FF	1K	EXTI	-	
0x4001_0000	0x4001_03FF	1K	Reserved	-	
0x4000_5800	0x4000_FFFF	42K	Reserved	-	APB1
0x4000_5400	0x4000_57FF	1K	I2C1	-	
0x4000_5000	0x4000_53FF	1K	Reserved	-	

边界地址		大小 (Byte)	外设	外设寄存器映象	总线
起点	终点				
0x4000_4C00	0x4000_4FFF	1K	UART4	-	
0x4000_4800	0x4000_4BFF	1K	UART3	-	
0x4000_4400	0x4000_47FF	1K	UART2	-	
0x4000_3400	0x4000_43FF	4K	Reserved	-	
0x4000_3000	0x4000_33FF	1K	IWDT	-	
0x4000_2C00	0x4000_2FFF	1K	WWDT	-	
0x4000_1400	0x4000_2BFF	6K	Reserved	-	
0x4000_1000	0x4000_13FF	1K	BS16T	-	
0x4000_0400	0x4000_0FFF	3K	Reserved	-	
0x4000_0000	0x4000_03FF	1K	GP32C4T	-	
0x2000_4000	0x3FFF_FFFF	~512M	Reserved	-	-
0x2000_2000	0x2000_3FFF	8K	Reserved	-	-
0x2000_1000	0x2000_1FFF	4K	SRAM2	-	-
0x2000_0000	0x2000_0FFF	4K	SRAM1	-	-
0x1FFF_F000	0x1FFF_FFFF	4K	Reserved	-	-
0x1FFF_EC00	0x1FFF_EFFF	1K	Flash Option	-	-
0x1FFF_E800	0x1FFF_EBFF	1K	System Option	-	-
0x1FFF_E600	0x1FFF_E7FF	0.5K	Reserved	-	-
0x1FFF_E400	0x1FFF_E5FF	0.5K	USER Option/WP	-	-
0x1FFF_E200	0x1FFF_E3FF	0.5K	RP	-	-
0x1FFF_E000	0x1FFF_E1FF	0.5K	UCRP	-	-
0x1FFF_1000	0x1FFF_DFFF	52K	Reserved	-	-
0x1FFF_0000	0x1FFF_0FFF	4K	System memory(ROM)	-	-
0x1000_0000	0x1FFE_FFFF	256M	Reserved	-	-
0x0802_0000	0x0FFF_FFFF		Reserved	-	-
0x0800_0000	0x0801_FFFF	128K	Main Flash	-	-
0x0002_0000	0x07FF_FFFF		Reserved	-	-
0x0000_0000	0x0001_FFFF	128K	取决于开机存储配置	-	-

表 1-3　外设寄存器地址映射

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 ES32F0943 中，该处理器配置以下特征:

- 内置嵌套向量中断控制器(NVIC)：32 个外部中断
- 小端模式
- 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

处理器外围设备：

- 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、递减计数到 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**) 中的值递减计数到 0，并在下一个时钟周期中重新加载 SysTick 重载值寄存器 (**SYST_RVR**) 中的值，然后在随后的时钟开始递减。一旦计数器计数为 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。

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 的中断线)
- 可编程优先级
- 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

位置	优先次序	优先级类型	名称	描述	地址
-	-	-	-	保留	0x0000 0000
-	-3	固定的	Reset	复位	0x0000 0004
-	-2	固定的	NMI_Handler	不可屏蔽中断。RCU 的 CSS 与 NMI 向量连接	0x0000 0008
-	-1	固定的	HardFault_Handler	所有类型的故障	0x0000 000C
-	-	-	-	保留	
-	3	可设置的	SVC_Handler	通过 SWI 指令调用的系统服务	0x0000 002C
-	-	-	-	保留	
-	5	可设置的	PendSV_Handler	可挂起的系统服务	0x0000 0038
-	6	可设置的	SysTick_Handler	系统定时器中断	0x0000 003C
0	7	可设置的	WWDT	WWDT 全局中断	0x0000 0040
1	8	可设置的	LVD	LVD 通过 EXTI 线 20 检测中断	0x0000 0044
2	9	可设置的	RTC	实时时钟(RTC)全局中断	0x0000 0048
3	10	可设置的	Low Power Wakeup	所有低功耗唤醒通过 EXTI 线 21 检测中断	0x0000 004C
4	11	可设置的	RCU	RCU 全局中断	0x0000 0050
5	12	可设置的	EXTI[1:0]	EXTI 线 0 至 1 中断	0x0000 0054
6	13	可设置的	EXTI[3:2]	EXTI 线 2 至 3 中断	0x0000 0058
7	14	可设置的	EXTI[15:4]	EXTI 线 4 至 15 中断	0x0000 005C
8	15	-	-	保留	0x0000 0060
9	16	-	-	保留	0x0000 0064
10	17	可设置的	LCD	LCD 全局中断	0x0000 0068
11	18	可设置的	ADC	ADC 全局中断	0x0000 006C
12	19	可设置的	CMP	CMP 全局中断	0x0000 0070
13	20	可设置的	OPAMP	OPAMP 全局中断	0x0000 0074
14	21	可设置的	BS16T	BS16T 全局中断	0x0000 0078
15	22	可设置的	GP32C4T	GP32C4T 全局中断	0x0000 007C
16	23	-	-	保留	0x0000 0080
17	24	-	-	保留	0x0000 0084
18	25	-	-	保留	0x0000 0088
19	26	可设置的	GP16C2T1	GP16C2T1 全局中断	0x0000 008C
20	27	可设置的	GP16C2T2	GP16C2T2 全局中断	0x0000 0090
21	28	可设置的	GP16C2T3	GP16C2T3 全局中断	0x0000 0094
22	29	可设置的	GP16C2T4	GP16C2T4 全局中断	0x0000 0098
23	30	可设置的	I2C1	I2C1 全局中断	0x0000 009C
24	31	-	-	保留	0x0000 00A0
25	32	可设置的	SPI1	SPI1 全局中断	0x0000 00A4
26	33	-	-	保留	0x0000 00A8
27	34	可设置的	UART1	UART1 全局中断	0x0000 00AC
28	35	可设置的	UART2	UART2 全局中断	0x0000 00B0
29	36	可设置的	UART3	UART3 全局中断	0x0000 00B4
30	37	可设置的	UART4	UART4 全局中断	0x0000 00B8
31	38	-	-	保留	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 定时器没有计数到 0。 1:自上次读取该位以来，SysTick 定时器已计数为 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:处理器时钟用于 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:计数器被禁用 1:计数器被使能

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。寄存器是写清除的。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:0x308																													
复位值:0x0000 0000																													
31	30	29	28	27	26	25	24	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: 在调试停止状态退出时，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，使用者必须向该位写入 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眠。 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，使用者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。

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 4 组 32 位的备份寄存器。
- 支持 5 种低功耗模式。
- 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)下让定时器(Timer)暂停计数。
- 支持调试模式(Debug Mode)下让 I2C(SMBus)暂停计数。
- 支持系统电源掉电、HOSC 发生故障或是 CPU 发生 Hard Fault 时，让定时器(Timer)暂停计数。
- 支持红外线模块(IR)控制信号组合配置。

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 1.8 伏至 5.5 伏的电压操作范围。系统内部依据工作电压的不同共可分为 2 个电压域，分为使用外部供电的系统电源域(VDDH Domain)与备份域(Backup Domain)，以及使用稳压器的核心电源域(VDD Domain)。

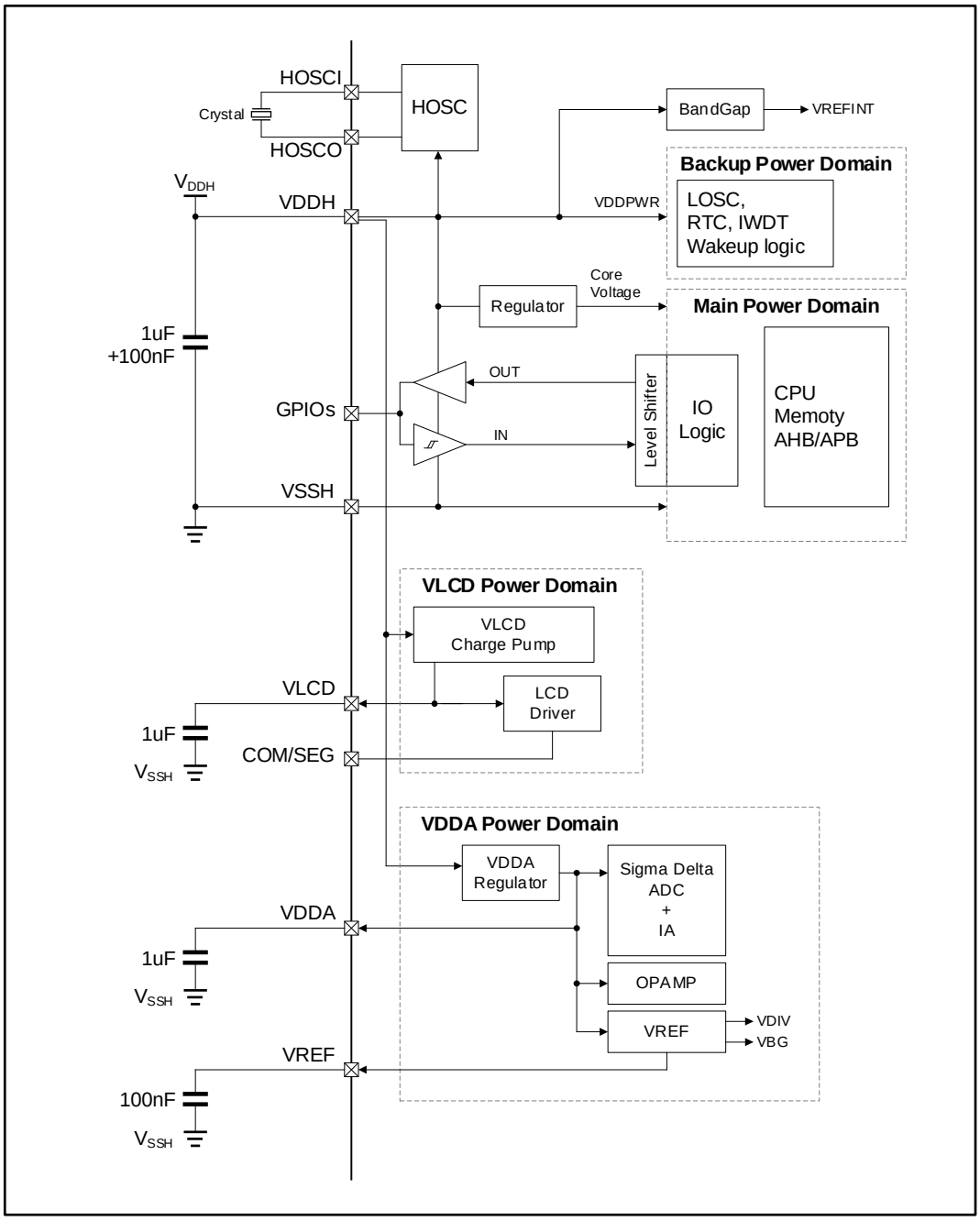


图 3-1　电源架构

3. 3. 1. 1 备份域 (Backup Domain)

备份域主要由操作在 VDDH 电源域的备份寄存器、低功耗控制单元、IWDT 组成，此区的电源仅受外部电源电压影响，因此当系统进入低功耗模式时仍会保留此区的电源。

- 备份寄存器的大小为 4 组 32 位的寄存器，分别为 **SYSCFG_BKREG0** 、**SYSCFG_BKREG1**、 **SYSCFG_BKREG2** 与 **SYSCFG_BKREG3**，当系统进入低功耗模式前可将需要备份的信息存放进备份寄存器内，当系统从低功耗模式唤醒后再从备份寄存器中读出暂存信息。
- 低功耗控制单元共有 3 组寄存器，分别为 **SYSCFG_PWRCON**、**SYSCFG_WKUP** 与 **SYSCFG_WKSR**，这 3 组寄存器可控制系统电源侦测与低功耗模式。

此电源域的寄存器仅支持以 32 位为单位进行读写操作，由于牵涉到跨电源域的读写操作，因此读写备份寄存器时会有一段延迟时间，在完成读写此电源域的寄存器前 CPU 会进行等待，用户程序内不用再额外多增加等待时间。

3. 3. 1. 2 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，系统会在芯片上电后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。

  由于稳压器主要提供电源给 VDD 电源域，因此当用户需要让系统长时间待机时，可让系统进入低功耗模式下的 STANDBY 模式与 SHUTDOWN 模式。在进入这 2 种低功耗模式时皆会关闭核心稳压器，此时 VDD 电源域内的外设皆会被关闭。

3. 3. 1. 3 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。

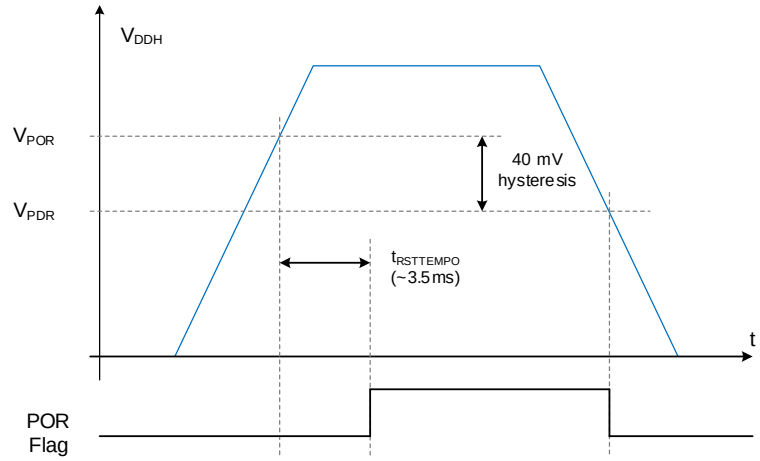


图 3-2　POR/PDR 复位

◆ 欠压复位(Brownout Reset)

欠压复位仅能够过修改用户配置字来开启，无法通过修改 **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。

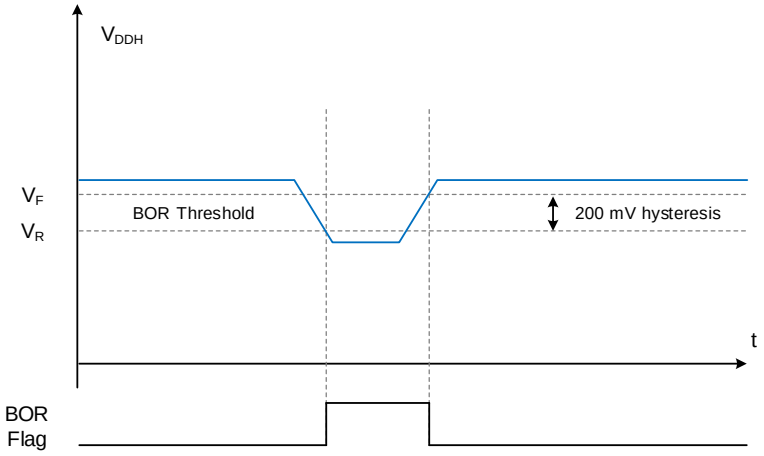


图 3-3　BOR 复位

◆ 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，若使用者有开启 EXTI 的中断功能则会在低电压标志位产生时触发中断，让使用者能够提前在掉电复位发生前将重要信息进行保留。由于 EXTI 逻辑支持使用者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使用者。此外低电压检测标志位也可当作外部触发信号唤醒处在 STOP0 模式、STOP1 模式或 STANDBY 模式的系统。

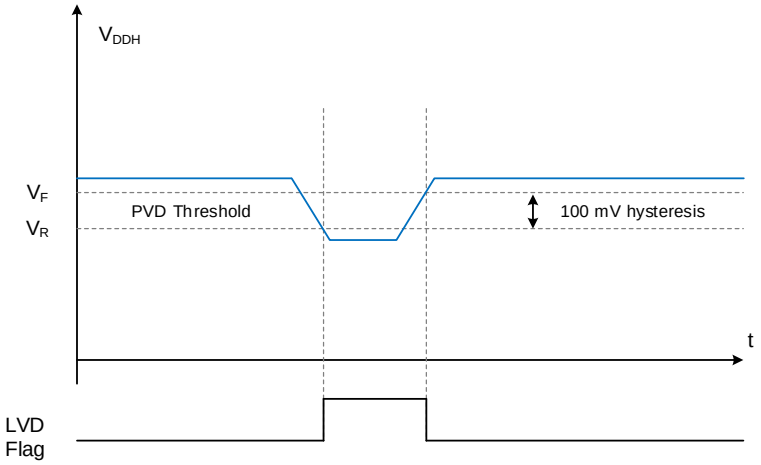


图 3-4　LVD 复位

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。若使用者希望在系统持续运行的情况下降低功率消耗，可通过降低系统时钟频率以及关闭不使用的外设时钟来达成。

芯片进入低功耗模式后，所有 I/O 端口将保持进入休眠前的状态。为了降低功耗，所有 I/O 端口都应保持为高电平或低电平，同时避免输入端口悬空而产生漏电流，可通过弱上拉或下拉将悬空的输入端口固定为高电平或低电平。如果产品封装引脚数小于最大引脚数，则未引出的和未使用的 I/O 管脚都需设置为输出低电平并禁止内部弱上下拉。

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、RTC 功能或用户决定配置 LRC/LOSC 开/关	LDO ON & BandGap ON	<4us
STOP0 模式	LPLS bits & SLPDEEP bit & WFI/WFE	$WKUP_0$ ~ $WKUP_4$, IWDT, NRST, LVD, RTC 触发 EXTI 唤醒中断	VDD 域时钟源全部关闭			<300us
STOP1 模式					LDO low-power & BandGap ON	<800us

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
STANDBY 模式		$WKUP_0$ ~ $WKUP_4$, IWDT, NRST, LVD		IWDT 功能或用户 决定配置 LRC/LOSC 开/关	LDO OFF & BandGap ON	~2ms
SHUTDOWN 模式		$WKUP_0$ ~ $WKUP_4$, IWDT, NRST			LDO OFF & BandGap OFF	~4ms

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，使用者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使用者代码。

  ◇ 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:

  当使用者拉低外部 NRST 引脚时会重置系统并产生 NRST 复位标志位，使用者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由哪一个外

设中断事件唤醒。

3. 3. 2. 3 低功耗STOP0 模式

进入低功耗 STOP0 模式后，所有时钟源都会被关闭，因此无法使用外设中断进行唤醒。唯一能够唤醒 CPU 的方式是通过唤醒事件触发的 EXTI 唤醒中断，该事件可以由唤醒引脚(WKUPx)、RTC 唤醒事件、IWDT 事件、低电压检测(LVD)或拉低外部 NRST 引脚来触发。

在唤醒后，可以通过读取 **SYSCFG_WKSR** 中的 FG 位(位 0 至位 15)来判断唤醒事件。在设定 WKCLR 为 1 后，还需要读取并判断 WKCLR 是否被自动清除为 0。只有在 WKCLR 被清除为 0 之后，才能再次进入低功耗模式。

此外，在进入 STOP0 模式之前，用户可以将 **FC_CTL** 内的 FCSLEEP 设定为 1，开启闪存停止模式。当系统进入 STOP0 模式时，闪存也会进入闪存停止模式，进一步降低电流消耗。如果需要开启闪存停止功能，在进入低功耗 STOP0 模式之前，建议将系统时钟切换至内部高速 RC 时钟(HRC)，并在唤醒后重新切回原系统时钟；且必须根据当前系统频率配置 **RCU_CFG1** 内的 SYSFREQ，以确保闪存有足够的时间可以退出闪存停止模式。

- 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 0，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:

    最大支持用户使用 5 个唤醒引脚(WKUPx)唤醒，用户需配置 **SYSCFG_WKUP** 内 WKEN 的位 0 至位 4 为 1 开启外部唤醒引脚，并配置 WKEG(位 16 至位 20)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。
  - RTC 唤醒:

    用户可配置 RTC 的唤醒定时器，并于计数完毕以后唤醒系统。用户可配置 **SYSCFG_WKUP** 内 WKEN 的位 11 为 1 开启 RTC 唤醒功能。由于 RTC 仅支持侦测上升沿发生时唤醒，因此用户毋须再配置 WKEG。
  - 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，因此用户毋须再配置 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，因此用户毋须再配置 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，会继续往下执行用户程序，CPU 可于被唤醒后检查 **SYSCFG_WKSR** 内的 FG 判断是由哪一个唤醒事件唤醒，并配置 **SYSCFG_WKSR** 内的 WKCLR 为 1 来清除唤醒标志位，若未清除标志位则系统无法再次进入低功耗模式。

3. 3. 2. 4 低功耗STOP1 模式

进入低功耗 STOP1 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。与 STOP0 模式相同，STOP1 模式也不支持使用外设中断进行唤醒，唯一能够唤醒 CPU 的方式是通过唤醒事件触发的 EXTI 唤醒中断，该事件可以由唤醒引脚(WKUPx)、RTC 唤醒事件、IWDT 事件、低电压检测(LVD)或拉低外部 NRST 引脚来触发。

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，以清除唤醒事件标志。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 STOP1 模式之前，用户可以将 **FC_CTL** 内的 FCSLEEP 设定为 1，开启闪存停止模式。当系统进入 STOP1 模式时，闪存也会进入闪存停止模式，进一步降低电流消耗。如果需要开启闪存停止功能，在进入低功耗 STOP1 模式前，将系统时钟切换至高速 RC 时钟(HRC)，并在唤醒后重新切回原系统时钟；且必须根据当前系统频率配置 **RCU_CFG1** 内的 SYSFREQ，以确保闪存有足够的时间可以退出闪存停止模式。

- 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 STOP1 模式唤醒
  - ◇ WKUPx 唤醒:

    最大支持用户使用 5 个唤醒引脚(WKUPx)唤醒，用户需配置 **SYSCFG_WKUP** 内 WKEN 的位 0 至位 4 为 1 开启外部唤醒引脚，并配置 WKEG(位 16 至位 20)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。
  - ◇ RTC 唤醒:

    用户可配置 RTC 的唤醒定时器，并于计数完毕以后唤醒系统。用户可配置 **SYSCFG_WKUP** 内 WKEN 的位 11 为 1 开启 RTC 唤醒功能。由于 RTC 仅支持侦测上升沿发生时唤醒，因此用户毋须再配置 WKEG。
  - ◇ 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，因此用户毋须再配置 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，因此用户毋须再配置 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，会继续往下执行用户程序，CPU 可于被唤醒后检查 **SYSCFG_WKSR** 内的 FG 判断是由哪一个唤醒事件唤醒，并配置 **SYSCFG_WKSR** 内的 WKCLR 为 1 来清除唤醒标志位，若未清除标志位则系统无法再次进入低功耗模式。

3. 3. 2. 5 低功耗STANDBY模式

当系统进入 STANDBY 模式时，会关闭 VDD 电源域内所有的电源。当系统从此低功耗模式唤醒后，系统将重启并重新开始执行闪存内的用户程序，用户需要重新配置系统与外设。为了确认唤醒事件，建议在程序中检查 **RCU_RSTF** 内的复位标志以及 **SYSCFG_WKSR** 内的 FG。此外，也需要配置 **SYSCFG_WKSR** 内的 WKCLR 为 1 来清除唤醒标志，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 WKCLR 是否被自动清除为 0，待 WKCLR 被清除为 0 才可再次进入低功耗模式。

需注意的是，在此低功耗模式下，将不再支持 RTC 计数功能，若有让 RTC 在低功耗模式期间持续计数功能需求的应用，建议选择配置为低功耗模式 STOP0 或 STOP1。

在系统进入 STANDBY 模式时，需要注意到唤醒引脚(WKUPx)的配置会被自动锁定，因此用户在进入 STANDBY 模式前必须配置好唤醒引脚(WKUPx)。当系统从 STANDBY 模式唤醒后，必须设置 **SYSCFG_WKSR** 内的 WKCLR 为 1 以清除唤醒标志，系统会在唤醒流程执行完毕后自动解除唤醒引脚(WKUPx)的锁定。如果用户将唤醒引脚(WKUPx)配置为 UART、I2C 或 SPI 等复用功能，则必须确保在唤醒流程执行完毕后再开始进行通讯功能。

◆ 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 2，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 STANDBY 模式唤醒

◇ WKUPx 唤醒:

最大支持用户使用 5 个唤醒引脚(WKUPx)唤醒，用户需配置 **SYSCFG_WKUP** 内 WKEN 的位 0 至位 4 为 1 开启外部唤醒引脚，并配置 WKEG(位 16 至位 20)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，因此用户毋须再配置 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，因此用户毋须再配置 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。

当系统从此低功耗模式唤醒时，会重启系统并重新开始执行闪存内的用户程序，建议用户于程序内加入检查 **RCU_RSTF** 内的复位标志位与 **SYSCFG_WKSR** 内的 FG 来检查唤醒事件，并配置 **SYSCFG_WKSR** 内的 WKCLR 为 1 来清除唤醒标志位，若未清除标志位则系统无法再次进入低功耗模式。

3. 3. 2. 6 低功耗SHUTDOWN模式

进入此模式后会关闭 VDD 电源域内所有的电源与 BandGap，能提供最佳的省电效果。由于 BandGap 电路被关闭，因此无法再使用 LVD 标志来唤醒系统，同时从此低功耗模式唤醒所需的时间也较其他低功耗模式长。当系统被唤醒后，系统将重启并重新开始执行闪存内的用户程序。需注意的是，在此低功耗模式下，将不再支持 RTC 计数功能，若有让 RTC 在低功耗模式期间持续计数功能需求的应用，建议选择配置为低功耗模式 STOP0 或 STOP1。

在系统进入 SHUTDOWN 模式时，需要注意到唤醒引脚(WKUPx)的配置会被自动锁定，因此用户在进入 SHUTDOWN 模式前必须配置好唤醒引脚(WKUPx)。当系统从 SHUTDOWN 模式唤醒后，必须设置 **SYSCFG_WKSR** 内的 WKCLR 为 1 以清除唤醒标志，系统会在唤醒流程执行完毕后自动解除唤醒引脚(WKUPx)的锁定。如果用户将唤醒引脚(WKUPx)配置为 UART、I2C 或 SPI 等复用功能，则必须确保在唤醒流程执行完毕后再开始进行通讯功能。

◆ 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 3，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 SHUTDOWN 模式唤醒

◇ WKUPx 唤醒:
最大支持用户使用 5 个唤醒引脚(WKUPx)唤醒，用户需配置 **SYSCFG_WKUP** 内 WKEN 的位 0 至位 4 为 1 开启外部唤醒引脚，并配置 WKEG(位 16 至位 20)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，因此用户毋须再配置 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。

当系统从此低功耗模式唤醒时，会重启系统并重新开始执行闪存内的用户程序，建议用户于程序内加入检查 **RCU_RSTF** 内的复位标志位与 **SYSCFG_WKSR** 内的 FG 来检查唤醒事件，并配置 **SYSCFG_WKSR** 内的 WKCLR 为 1 来清除唤醒标志位，若未清除标志位则系统无法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)。闪存可供使用者存放代码，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使用者代码。

◆ 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)时，可于系统开机完毕后直接从闪存开始执行使用者的代码，达到快速开机的效果。硬件映射可由使用者决定是否开启，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) 软件重映射支持使用者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通过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，使用者需使用 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) 当使用者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，当使用者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

芯片支持定时器(Timer)、看门狗与 I2C 在调试模式下能够暂时停止计数。用户可通过配置 **SYSCFG_CFG** 的 DBGHEN(位 16 至位 31)决定是否开启对应外设的功能。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。而定时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. 3. 5 红外线(IR)控制信号

芯片支持使用者使用定时器(Timer)搭配串口(UART)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 **SYSCFG_IRSEL** 内的 SEL1(位 0 至位 3)与 SEL2(位 4 至位 7)决定要如何搭配出 PWM 控制信号，此芯片目前仅支持使用者从 GP16C2T1 与 GP16C2T2 中挑选其中一个通道输出 PWM 信号去与 GP32C4T1、UART1 Tx 与 URAT2 Tx 的其中一个信号进行逻辑 AND 运算，并配置 PLR(位 8)决定 IR 输出信号的极性。当 PLR 数值为 1 时代表 IR Out 正向输出运算结果信号，反之若 PLR 数值为 0 则代表 IR Out 反向输出运算结果信号。有关计数器与串口的配置方式请参阅相关章节的说明。

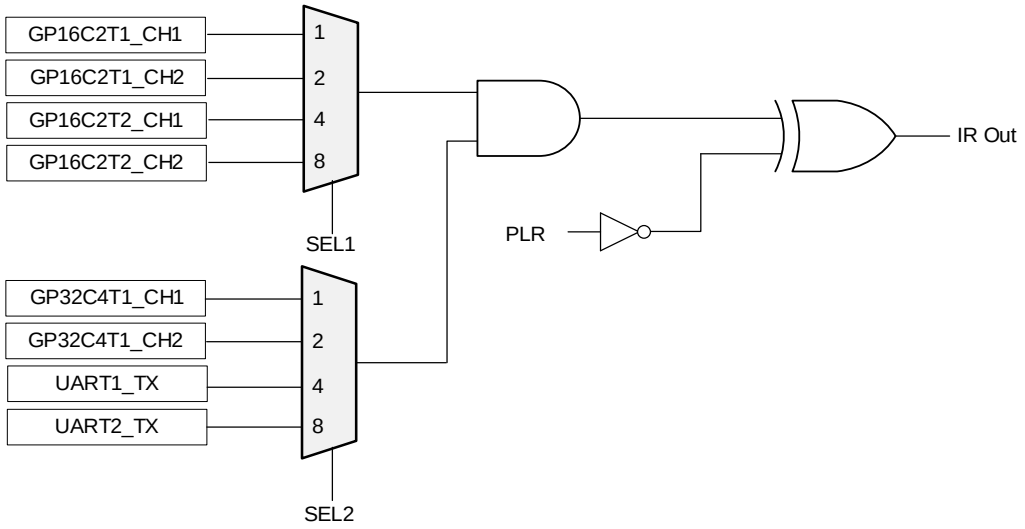


图 3-5 红外线控制信号组合

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_IRSEL	$000C_H$	R/W	红外线控制输出寄存器
SYSCFG_SYSTRIM	0010_H	R	系统校准寄存器
SYSCFG_CLKTRIM	0014_H	R	系统时钟校准寄存器
SYSCFG_OSCTRIM	0018_H	R	外部晶振校准寄存器
SYSCFG_IPTRIM	$001C_H$	R	周边模块校准寄存器
SYSCFG_AHBIPEN	0020_H	R	AHB 外设有效列表寄存器
SYSCFG_APB1IPEN	0024_H	R	APB1 外设有效列表寄存器
SYSCFG_APB2IPEN	0028_H	R	APB2 外设有效列表寄存器
SYSCFG_MEMMOD	$002C_H$	R	内存模式寄存器
SYSCFG_SYSSET	0038_H	R	系统周边设定寄存器
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_WKTRIM	0044_H	R	备份域校准状态寄存器
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	唤醒状态寄存器
SYSCFG_BKREG0	0050_H	R/W	备份寄存器 0
SYSCFG_BKREG1	0054_H	R/W	备份寄存器 1
SYSCFG_BKREG2	0058_H	R/W	备份寄存器 2
SYSCFG_BKREG3	$005C_H$	R/W	备份寄存器 3

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，可通过配置 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 红外线控制输出寄存器 (SYSCFG_IRSEL)

红外线控制输出寄存器(SYSCFG_IRSEL)

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
																							PLR	SEL2<3:0>				SEL1<3:0>			

—	Bits 31-9	—	—
PLR	Bit 8	R/W	**红外线控制输出极性** 配置此位可将红外线输出的控制信号进行反相。 0:开启输出控制信号反相，控制信号可表示为(SEL1 & SEL2)。 1:关闭输出控制信号反相，控制信号可表示为(SEL1 & SEL2)。
SEL2	Bits 7-4	R/W	**红外线控制输出选择 2** 配置此位决定红外线输出控制信号 SEL2 的来源。 0000:关闭控制输出。 0001:控制输出来源选择 GP32C4T1_CH1。 0010:控制输出来源选择 GP32C4T1_CH2。 0100:控制输出来源选择 UART1_Tx。 1000:控制输出来源选择 UART2_Tx。
SEL1	Bits 3-0	R/W	**红外线控制输出选择 1** 配置此位决定红外线输出控制信号 SEL1 的来源。 0000:关闭控制输出。

			0001:控制输出来源选择 GP16C2T1_CH1。 0010:控制输出来源选择 GP16C2T1_CH2。 0100:控制输出来源选择 GP16C2T2_CH1。 1000:控制输出来源选择 GP16C2T2_CH2。

3. 4. 2. 3 系统校准寄存器 (SYSCFG_SYSTRIM)

系统校准寄存器 (SYSCFG_SYSTRIM)																												
偏移地址:0x10																												
复位值:0x0088 0288(数值依据芯片出厂校准而定)																												
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								LDOLPTRIM<3:0>				LDONPTRIM<3:0>										LVDTRIM <1:0>		BG1VTRIM<3:0>				BG1V2TRIM<3:0>			

—	Bit 31-24	—	—
LDOLPTRIM	Bit 23-20	R	**Low Power LDO 校准值** LDOLPTRIM存放LDOLP电压的校准数值，此数值在芯片出厂前已针对芯片特性进行调整，仅开放读取。
LDONPTRIM	Bit 19-16	R	**Normal LDO 校准值** LDONPTRIM 存放 LDONP 电压的校准数值，此数值在芯片出厂前已针对芯片特性进行调整，仅开放读取。
—	Bit 15-10	—	—
LVDTRIM	Bit 9-8	R	**低电压检测校准值** LVDTRIM 存放低电压检器的校准数值，此数值在芯片出厂前已针对芯片特性进行调整，仅开放读取。
BG1VTRIM	Bit 7-4	R	**BandGap 1V 校准值** BG1VTRIM 存放 BandGap 1V 的校准数值，此数值在芯片出厂前已针对芯片特性进行调整，仅开放读取。
BG1V2TRIM	Bit 3-0	R	**BandGap 1.2V 校准值** BG1V2TRIM 存放 BandGap 1.2V 的校准数值，此数值在芯片出厂前已针对芯片特性进行调整，仅开放读取。

3. 4. 2. 4 系统时钟校准寄存器 (SYSCFG_CLKTRIM)

系统时钟校准寄存器 (SYSCFG_CLKTRIM)																													
偏移地址:0x14																													
复位值:0x0050 7E80(数值依据芯片出厂校准而定)																													
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
									LRCTRIM<6:0>															HRCTRIML<7:0>						

—	Bit 31-23	—	—
LRCTRIM	Bit 22-16	R	**LRC 时钟校准值** LRCTRIM 存放 LRC 时钟的校准数值，此数值在芯片出厂前已针对芯片特性进行调整，仅开放读取。
—	Bit 15-8	—	—
HRCTRIML	Bit 7-0	R	**HRC 时钟校准值 L** HRCTRIML 存放 HRC 时钟 16MHz 的校准数值，此数值在芯片出厂前已针对芯片特性进行调整，仅开放读取。

3. 4. 2. 5 外部晶振校准寄存器 (SYSCFG_OSCTRIM)

外部晶振校准寄存器 (SYSCFG_OSCTRIM)																													
偏移地址:0x18																													
复位值:0x0000 3742(数值依据芯片出厂校准而定)																													
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																		LOSCRCNT <1:0>		LOSCCURTRIM <3:0>					HOSCRCNT <2:0>					HOSCCURSEL <1:0>	

—	Bit 31-14	—	—
LOSCRCNT	Bit 13-12	R	**LOSC 时钟稳定计数值** LOSCRDYCNT 数值代表开启 LOSC 时钟后到 LOSC Ready 标志位拉高所需计数的 LOSC 时钟周期数。此数值在芯片出厂前已针对芯片特性进行调整，仅开放读取。 0x0: LOSC 计数 2048 个周期后拉高 Ready 标志位。

			0x1: LOSC 计数 4096 个周期后拉高 Ready 标志位。 0x2: LOSC 计数 8192 个周期后拉高 Ready 标志位。 0x3: LOSC 计数 16384 个周期后拉高 Ready 标志位。(默认值)
LOSCCURTRIM	Bit 11-8	R	**LOSC 电流选择** 此数值在芯片出厂前已针对芯片特性进行调整，并于系统开机后自动从闪存内读出。
—	Bit 7	—	—
HOSCRCNT	Bit 6-4	R	**HOSC 时钟稳定计数值** HOSCRDYCNT 数值代表开启 HOSC 时钟后到 HOSC Ready 标志位拉高所需计数的 HOSC 时钟周期数。此数值在芯片出厂前已针对芯片特性进行调整，仅开放读取。 0x0: HOSC 计数 256 个周期后拉高 Ready 标志位。 0x1: HOSC 计数 512 个周期后拉高 Ready 标志位。 0x2: HOSC 计数 1024 个周期后拉高 Ready 标志位。 0x3: HOSC 计数 2048 个周期后拉高 Ready 标志位。 0x4: HOSC 计数 4096 个周期后拉高 Ready 标志位。(默认值) 0x5: HOSC 计数 8192 个周期后拉高 Ready 标志位。 0x6: HOSC 计数 16384 个周期后拉高 Ready 标志位。 0x7: HOSC 计数 32768 个周期后拉高 Ready 标志位。
—	Bit 3-2	—	—
HOSCCURSEL	Bit 1-0	R	**HOSC 电流选择** 此数值在芯片出厂前已针对芯片特性进行调整，并于系统开机后自动从闪存内读出。

3. 4. 2. 6 周边模块校准寄存器 (SYSCFG_IPTRIM)

周边模块校准寄存器 (SYSCFG_IPTRIM)																														
偏移地址:0x1C																														
复位值:0x0000 7840(数值依据芯片出厂校准而定)																														
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																CMPTRIM<7:0>									AFEVBGTRIM <6:0>						

—	Bit 31-16	—	—
CMPTRIM	Bit 15-8	R	**CMP 校准数值** 此数值在芯片出厂前已针对芯片特性进行调整，仅开放读取。
—	Bit 7	—	—
AFEVBGTRIM	Bit 6-0	R	**AFE Bandgap 校准数值** 此数值在芯片出厂前已针对芯片特性进行调整，仅开放读取。

3. 4. 2. 7 AHB外设有效列表寄存器 (SYSCFG_AHBIPEN)

AHB 外设有效列表寄存器 (SYSCFG_AHBIPEN)																														
偏移地址:0x20																														
复位值:0xFFFF FFFF(数值依据芯片出厂校准而定)																														
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
												GPDEN	GPCEN	GPBEN	GPAEN	CALCEN									RTCEN						

—	Bit 31-20	—	—
GPDEN	Bit 19	R	**GPIO D 外设有效状态** 芯片是否含有此外设功能主要依据芯片型号而定，使用者可从此寄存器的数值判断，确认芯片是否含有此 AHB 外设功能。 0:芯片内不含此外设功能。 1:芯片内含有此外设功能。
GPCEN	Bit 18	R	**GPIO C 外设有效状态** 芯片是否含有此外设功能主要依据芯片型号而定，使用者可从此寄存器的数值判断，确认芯

			片是否含有此 AHB 外设功能。 0:芯片内不含此外设功能。 1:芯片内含有此外设功能。
GPBEN	Bit 17	R	**GPIO B 外设有效状态** 芯片是否含有此外设功能主要依据芯片型号而定，使用者可从此寄存器的数值判断，确认芯片是否含有此 AHB 外设功能。 0:芯片内不含此外设功能。 1:芯片内含有此外设功能。
GPAEN	Bit 16	R	**GPIO A 外设有效状态** 芯片是否含有此外设功能主要依据芯片型号而定，使用者可从此寄存器的数值判断，确认芯片是否含有此 AHB 外设功能。 0:芯片内不含此外设功能。 1:芯片内含有此外设功能。
CALCEN	Bit 15	R	**CALC 外设有效状态** 芯片是否含有此外设功能主要依据芯片型号而定，使用者可从此寄存器的数值判断，确认芯片是否含有此 AHB 外设功能。 0:芯片内不含此外设功能。 1:芯片内含有此外设功能。
—	Bit 14-7	—	—
RTCEN	Bit 6	R	**RTC 外设有效状态** 芯片是否含有此外设功能主要依据芯片型号而定，使用者可从此寄存器的数值判断，确认芯片是否含有此 AHB 外设功能。 0:芯片内不含此外设功能。 1:芯片内含有此外设功能。
—	Bit 5-0	—	—

3. 4. 2. 8 APB1 外设有效列表寄存器 (SYSCFG_APB1IPEN)

APB1 外设有效列表寄存器 (SYSCFG_APB1IPEN)																												
偏移地址:0x24																												
复位值:0xFFFF FFFF(数值依据芯片出厂校准而定)																												
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
—	—	—	—	—	—	—	—	—	—	I2C1EN	—	UART4EN	UART3EN	UART2EN	—	—	—	—	—	WWDTEN	—	—	—	—	—	—	BS16T1EN	—	—	—	GP32C4T1EN

—	Bit 31-22	—	—
I2C1EN	Bit 21	R	**I2C1 外设有效状态** 芯片是否含有此外设功能主要依据芯片型号而定，使用者可从此寄存器的数值判断，确认芯片是否含有此 APB1 外设功能。 0:芯片内不含此外设功能。 1:芯片内含有此外设功能。
—	Bit 20	—	—
UART4EN	Bit 19	R	**UART4 外设有效状态** 芯片是否含有此外设功能主要依据芯片型号而定，使用者可从此寄存器的数值判断，确认芯片是否含有此 APB1 外设功能。 0:芯片内不含此外设功能。 1:芯片内含有此外设功能。
UART3EN	Bit 18	R	**UART3 外设有效状态** 芯片是否含有此外设功能主要依据芯片型号而定，使用者可从此寄存器的数值判断，确认芯片是否含有此 APB1 外设功能。 0:芯片内不含此外设功能。 1:芯片内含有此外设功能。
UART2EN	Bit 17	R	**UART2 外设有效状态** 芯片是否含有此外设功能主要依据芯片型号而定，使用者可从此寄存器的数值判断，确认芯片是否含有此 APB1 外设功能。 0:芯片内不含此外设功能。 1:芯片内含有此外设功能。
—	Bit 16-12	—	—
WWDTEN	Bit 11	R	**WWDT 外设有效状态**

			芯片是否含有此外设功能主要依据芯片型号而定，使用者可从此寄存器的数值判断，确认芯片是否含有此 APB1 外设功能。 0:芯片内不含此外设功能。 1:芯片内含有此外设功能。
—	Bit 10-5	—	—
BS16T1EN	Bit 4	R	**BS16T1 外设有效状态** 芯片是否含有此外设功能主要依据芯片型号而定，使用者可从此寄存器的数值判断，确认芯片是否含有此 APB1 外设功能。 0:芯片内不含此外设功能。 1:芯片内含有此外设功能。
—	Bit 3-1	—	—
GP32C4T1EN	Bit 0	R	**GP32C4T1 外设有效状态** 芯片是否含有此外设功能主要依据芯片型号而定，使用者可从此寄存器的数值判断，确认芯片是否含有此 APB1 外设功能。 0:芯片内不含此外设功能。 1:芯片内含有此外设功能。

3. 4. 2. 9 APB2 外设有效列表寄存器 (SYSCFG_APB2IPEN)

APB2 外设有效列表寄存器 (SYSCFG_APB2IPEN)

偏移地址:0x28

复位值:0xFFFF FFFF(数值依据芯片出厂校准而定)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
—	—	—	—	—	—	—	—	CMPEN	—	—	—	GP16C2T4EN	GP16C2T3EN	GP16C2T2EN	GP16C2T1EN	—	UART1EN	—	SPI1EN	—	—	—	—	—	—	OPAMPEN	ANPWREN	ADCEN	LCDEN	—	—

—	Bit 31-20	—	—
CMPEN	Bit 23	R	**CMP外设有效状态** 芯片是否含有此外设功能主要依据芯片型号而定，用户可从此寄存器的数值判，确认芯片是否含有此APB2外设功能。 0: 芯片内不含此外设功能。 1: 芯片内含有此外设功能。
—	Bit 22-20	—	—

GP16C2T4EN	Bit 19	R	**GP16C2T4 外设有效状态** 芯片是否含有此外设功能主要依据芯片型号而定，使用者可从此寄存器的数值判断，确认芯片是否含有此 APB2 外设功能。 0:芯片内不含此外设功能。 1:芯片内含有此外设功能。
GP16C2T3EN	Bit 18	R	**GP16C2T3 外设有效状态** 芯片是否含有此外设功能主要依据芯片型号而定，使用者可从此寄存器的数值判断，确认芯片是否含有此 APB2 外设功能。 0:芯片内不含此外设功能。 1:芯片内含有此外设功能。
GP16C2T2EN	Bit 17	R	**GP16C2T2 外设有效状态** 芯片是否含有此外设功能主要依据芯片型号而定，使用者可从此寄存器的数值判断，确认芯片是否含有此 APB2 外设功能。 0:芯片内不含此外设功能。 1:芯片内含有此外设功能。
GP16C2T1EN	Bit 16	R	**GP16C2T1 外设有效状态** 芯片是否含有此外设功能主要依据芯片型号而定，使用者可从此寄存器的数值判断，确认芯片是否含有此 APB2 外设功能。 0:芯片内不含此外设功能。 1:芯片内含有此外设功能。
—	Bit 15	—	—
UART1EN	Bit 14	R	**UART1 外设有效状态** 芯片是否含有此外设功能主要依据芯片型号而定，使用者可从此寄存器的数值判断，确认芯片是否含有此 APB2 外设功能。 0:芯片内不含此外设功能。 1:芯片内含有此外设功能。
—	Bit 13	—	—
SPI1EN	Bit 12	R	**SPI1 外设有效状态** 芯片是否含有此外设功能主要依据芯片型号而定，使用者可从此寄存器的数值判断，确认芯片是否含有此 APB2 外设功能。 0:芯片内不含此外设功能。 1:芯片内含有此外设功能。

—	Bit 11-6	—	—
OPAMPEN	Bit 5	R	**OPAMP 外设有效状态** 芯片是否含有此外设功能主要依据芯片型号而定，用户可从此寄存器的数值判断，确认芯片是否含有此 APB2 外设功能。 0:芯片内不含此外设功能。 1:芯片内含有此外设功能。
ANPWREN	Bit 4	R	**ANPWR 外设有效状态** 芯片是否含有此外设功能主要依据芯片型号而定，用户可从此寄存器的数值判断，确认芯片是否含有此 APB2 外设功能。 0:芯片内不含此外设功能。 1:芯片内含有此外设功能。
ADCEN	Bit 3	R	**ADC 外设有效状态** 芯片是否含有此外设功能主要依据芯片型号而定，用户可从此寄存器的数值判断，确认芯片是否含有此 APB2 外设功能。 0:芯片内不含此外设功能。 1:芯片内含有此外设功能。
LCDEN	Bit 2	R	**LCD 外设有效状态** 芯片是否含有此外设功能主要依据芯片型号而定，用户可从此寄存器的数值判断，确认芯片是否含有此 APB2 外设功能。 0:芯片内不含此外设功能。 1:芯片内含有此外设功能。
—	Bit 1-0	—	—

3. 4. 2. 10 内存模式寄存器 (SYSCFG_MEMMOD)

内存模式寄存器 (SYSCFG_MEMMOD)																												
偏移地址:0x2C																												
复位值:0x0000 0F7F(数值依据芯片出厂校准而定)																												
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								SMOD <7:0>															FMOD <8:0>							

—	Bit 31-24	—	—
SMOD	Bit 23-16	R	**内存大小** 内存的大小依据芯片型号而有所不同，使用者可读取 SMOD 的数值判断此芯片型号的内存大小。此数值在芯片出厂前已针对芯片特性进行调整，仅开放读取。 0x7F:内存大小为 128KB。 0x3F:内存大小为 64KB。 0x1F:内存大小为 32KB。 0x0F:内存大小为 16KB。 0x07:内存大小为 8KB。(默认) 0x03:内存大小为 4KB。 0x01:内存大小为 2KB。
—	Bit 15-9	—	—
FMOD	Bit 8-0	R	**闪存大小** 闪存的大小依据芯片型号而有所不同，使用者可读取 FMOD 的数值判断此芯片型号的闪存大小。此数值在芯片出厂前已针对芯片特性进行调整，仅开放读取。 0x1FF:闪存大小为 512KB。 0xFF:闪存大小为 256KB。 0x7F:闪存大小为 128KB。(默认) 0x3F:闪存大小为 64KB。 0x1F:闪存大小为 32KB。 0x0F:闪存大小为 16KB。

3. 4. 2. 11 系统周边设定寄存器 (SYSCFG_SYSSET)

系统周边设定寄存器 (SYSCFG_SYSSET)																													
偏移地址:0x38																													
复位值:0xFFFF FFFF(数值依据芯片出厂校准而定)																													
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
—	—	—	—	—	—	—	—	SYS_IWDTEN <7:0>								SYS_BOREN <7:0>								—	—	—	—	—	SYS_BORLS <2:0>		

—	Bit 31-24	—	—
SYS_IWDTEN	Bit 23-16	R	**用户配置字 IWDT 开关** 反映用户配置字内的 IWDT 开关设定数值。当 SYS_IWDTEN 数值为 0xA5 时代表系统默认开启 IWDT。
SYS_BOREN	Bit 15-8	R	**用户配置字 BOR 开关** 反映用户配置字内的 BOR 开关设定数值。当 SYS_BOREN 数值为 0xA5 时代表系统默认开启 BOR。
—	Bit 7-3	—	—
SYS_BORLS	Bit 2-0	R	**用户配置字 BOR 电压阀值选择** 反映用户配置字内所设定的 BOR 侦测电压阀值。详细电压区间请参阅 Datasheet 内电气特性章节描述。 000:BOR level 0 001: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

3. 4. 2. 12 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												BKREADY

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] :保留。 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] :保留。 DBGHEN[4] :GP32C4T1。 DBGHEN[3] : GP16C2T4。 DBGHEN[2] : GP16C2T3。 DBGHEN[1] : GP16C2T2。 DBGHEN[0] :GP16C2T1。
—	Bit 15	—	—
LVDLCK	Bit 14	R/W	**低电压检测(LVD)事件输入开关** LVD 事件可从 GP16C2T1~GP16C2T4 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 GP16C2T1~GP16C2T4 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 GP16C2T1~GP16C2T4 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 11-1	—	—
BKREADY	Bit 0	R	**备份寄存器总线空闲标志位** 在对备份寄存器进行读取/写入操作时，需先确认 BKREADY 为 1 后才可进行操作。 0:备份寄存器总线处于忙碌状态。 1:备份寄存器总线处于空闲状态。

3. 4. 2. 13 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**

			通过者用户配置字配置 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:BOR level 0 001: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阀值。当系统电压降至选择的电压阀值时，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. 14 备份域校准状态寄存器 (SYSCFG_WKTRIM)

此寄存器只允许 32 位存取

备份域校准状态寄存器 (SYSCFG_WKTRIM)																															
偏移地址:0x44																															
复位值:0xB750 8888																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LVDTRIM<1:0>		LOSCRCNT<1:0>		LOSCCURTRIM<2:0			^	|	LRCTRIM<6:0>							LDOLPTRIM<3:0>				LDONPTRIM<3:0>				BG1V2TRIM<3:0>				BG1VTRIM<3:0>			

LVDTRIM	Bit 31-30	R	**低电压检测(LVD)校准值** LVDTRIM存放低电压检测器的校准数值，此数值在芯片出厂前已针对芯片特性进行调整，仅开放读取。
LOSCRCNT	Bit 29-28	R	**外部低速时钟振荡器，时钟稳定时间计数器** 00: LOSC 计数 2048 个周期后拉高 Ready 标志位。 01: LOSC 计数 4096 个周期后拉高 Ready 标志位。 10: LOSC 计数 8192 个周期后拉高 Ready 标志位。 11: LOSC 计数 16384 个周期后拉高 Ready 标志位。(默认值)
LOSCCURSEL	Bit 27-24	R	**外部低速时钟振荡器，起振电流配置** 000:起振电流选择 120nA。 001:起振电流选择 200nA。 010:起振电流选择 280nA。 011:起振电流选择 360nA。 100:起振电流选择 440nA。 101:起振电流选择 520nA。 110:起振电流选择 600nA。 111:起振电流选择 680nA。
—	Bit 23	—	—
LRCTRIM	Bit 22-16	R	**LRC 时钟校准值** LRCTRIM 存放 LRC 时钟的校准数值，此数值在芯片出厂前已针对芯片特性进行调整，仅开放读取。

LDOLPTRIM	Bit 15-12	R	**Low Power LDO 校准值** LDOLPTRIM 存放 Low Power LDO 的校准数值，此数值在芯片出厂前已针对芯片特性进行调整，仅开放读取。
LDONPTRIM	Bit 11-8	R	**Normal LDO 校准值** LDONPTRIM 存放 Normal Power LDO 的校准数值，此数值在芯片出厂前已针对芯片特性进行调整，仅开放读取。
BG1V2TRIM	Bit 7-4	R	**BandGap 1.2V 校准值** BG1V2TRIM 存放 BandGap 的校准数值，此数值在芯片出厂前已针对芯片特性进行调整，仅开放读取。
BG1VTRIM	Bit 3-0	R	**BandGap 1V 校准值** BG1VTRIM 存放 BandGap 的校准数值，此数值在芯片出厂前已针对芯片特性进行调整，仅开放读取。

3. 4. 2. 15 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:0x0B00 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唤醒计数器与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] :保留。 WKEG[6] :保留。 WKEG[7] :保留。 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] : RTC计数标志位上升沿唤醒，固定为1。(Read Only) 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] :保留。 WKEN[6] :保留。 WKEN[7] :保留。 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] : RTC 计数唤醒功能，依据 RTCEN 配置状态而定。(Read Only) WKEG[12] :保留 WKEG[13] :保留 WKEG[14] :保留。 WKEG[15] :保留。

BOR 标志位会触发 PDR 复位，无法当作唤醒标志位将系统唤醒。

当系统进入 SHUTDOWN 模式后，会关闭 BandGap，此时仅能通过唤醒引脚(WKUPx)唤醒。

3. 4. 2. 16 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:0x0000 0000																													
31	30	29	28	27	26	25	24	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 清除唤醒标志位，并结束低功耗流程。WKCLR 会在低功后流程结束后自动清除为 0。若使用唤醒引脚或唤醒事件将系统从低功耗模式唤醒时，务必设定 WKCLR 清除唤醒标志位，否则系统无法再次进入低功耗流程。
—	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: STOP0 模式。 1:STOP1 模式。 2: STANDBY 模式。 3: SHUTDOWN 模式。 注:STANDBY 模式与 SHUTDOWN 模式不支持使用 RTC。

LPVSEL	Bit 27-26	R/W	**Low Power LDO 电压选择** 配置 LPVSEL 决定系统要进入 STOP1 模式时，Low Power LDO 输出电压值。 0: 1.3V。 1: 1.4V。(默认值) elsa: 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。可通过设定 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哪些唤醒引脚或事件。可通过设定 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] :保留。 FG[6] :保留。 FG[7] :保留。 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 RTC 闹钟标志位上升沿事件唤醒。 FG[12] :保留。 FG[13] :保留。 FG[14] :保留。 FG[15] :保留。

3. 4. 2. 17 备份寄存器 0 (SYSCFG_BKREG0)

此寄存器只允许 32 位存取

备份寄存器 0 (SYSCFG_BKREG0)																															
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
BKREG0<31:0>																															

BKREG0	Bits 31-0	R/W	**备份寄存器0** 此寄存器的内容不受NRST复位影响，仅有在系统电源掉电并触发PDR/BOR时才会被清除。

3. 4. 2. 18 备份寄存器 1 (SYSCFG_BKREG1)

此寄存器只允许 32 位存取

备份寄存器 1 (SYSCFG_BKREG1)																															
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
BKREG1<31:0>																															

BKREG1	Bits 31-0	R/W	**备份寄存器1** 此寄存器的内容不受NRST复位影响，仅有在系统电源掉电并触发PDR/BOR时才会被清除。

3. 4. 2. 19 备份寄存器 2 (SYSCFG_BKREG2)

此寄存器只允许 32 位存取

备份寄存器 2 (SYSCFG_BKREG2)																															
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
BKREG2<31:0>																															

BKREG2	Bits 31-0	R/W	**备份寄存器2** 此寄存器的内容不受NRST复位影响，仅有在系统电源掉电并触发PDR/BOR时才会被清除。

3. 4. 2. 20 备份寄存器 3 (SYSCFG_BKREG3)

此寄存器只允许 32 位存取

备份寄存器 3(SYSCFG_BKREG3)																															
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
BKREG3<31:0>																															

BKREG3	Bits 31-0	R/W	**备份寄存器3** 此寄存器的内容不受NRST复位影响，仅有在系统电源掉电并触发PDR/BOR时才会被清除。

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，使用者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 5 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 的分频数。
- 支持 APB 时钟(PCLK)分频，可以根据 AHB 时钟设定 2、4、8 与 16 的分频数。
- 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 的分频数。
- 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** 供使用者检视。

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 伏上升并且超过 V_{POR} 时，POR 模块会等待约 3.5 毫秒(3.5ms)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，可通过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，供使用者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 低功耗复位

当系统从 STANDBY 模式以及 SHUTDOWN 模式时唤醒时，会触发低功耗复位，让 CPU 重新执行程序。相关信息请参阅系统配置控制器章节内的低功耗模式章节。当发生低功耗复位时，除了备份寄存器，所有寄存器都将被清除。

4. 3. 1. 3 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数 NVIC_SystemReset()来重新启动系统。
- 配置字重载复位
  当使用者开启闪存保护设定时，需通过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。

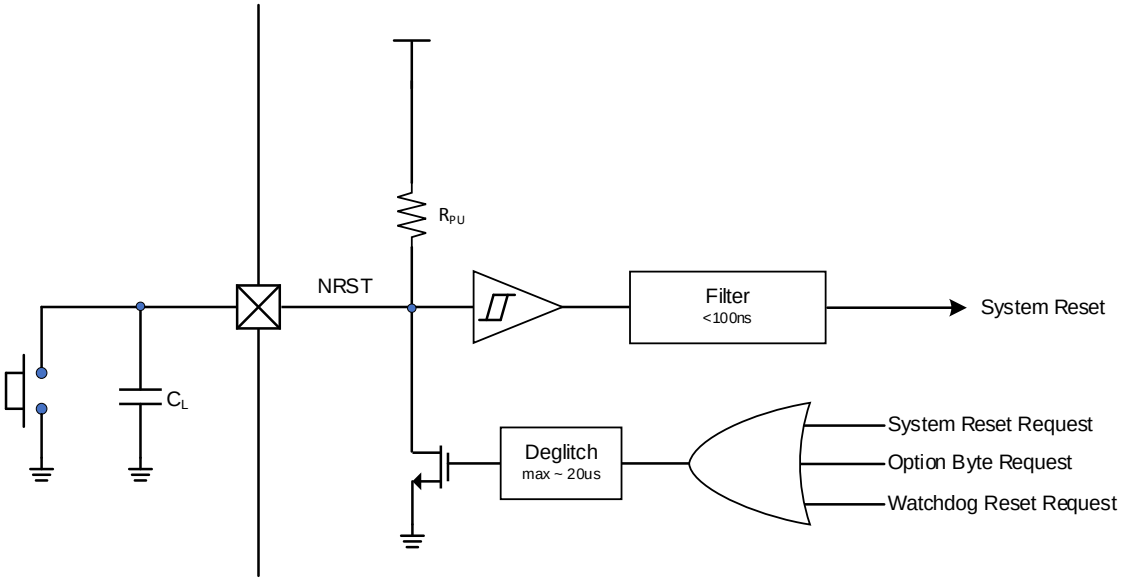


图 4-1 系统复位

4. 3. 1. 4 复位标志位

当复位发生时系统会记录触发复位的原因，使用者可以读取复位标志位寄存器 **RCU_RSTF** 来了解何种因素导致系统被复位。由于系统在启动过程中，一定会触发上电复位，因此 POR/PDR 复位标志位 PORRSTF 初始值为 1。当使用者检查完复位标志位后，可以设置 CLRFLG 清除

复位标志位，此位会在清除完复位标志位后自动清零。须注意的是当发生上电/掉电复位时，复位标志位寄存器会被复位，仅保留 POR/PDR 复位标志位。

4. 3. 1. 5 AHB/APB 外设复位

AHB/APB 外设初始状态处于复位状态，当用户配置 AHB 外设时钟控制寄存器 **RCU_AHBEN**、APB1 外设时钟控制寄存器 **RCU_APB1EN** 与 APB2 外设时钟控制寄存器 **RCU_APB2EN**，需要等待 3 个外设时钟后，外设才会离开复位状态。用户可通过配置 AHB 外设复位控制寄存器 **RCU_AHBRST**、APB1 外设复位控制寄存器 **RCU_APB1RST** 与 APB2 外设复位控制寄存器 **RCU_APB2RST** 来复位外设，当这三组寄存器被设定为 1 时，外设会处于复位状态，例如配置 **RCU_AHBEN**.GPAEN 为 1，即可让 GPIOA 外设进入复位状态；用户需要自行再将寄存器清除为 0，使外设离开复位状态。须注意释放外设复位后，需要等待 3 个外设时钟才会离开复位状态，在这段期间内，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差 16 倍，因此用户在使用此功能时，须特别注意程序上的配置，不建议用户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 4 MHz 至 32 MHz
- PLL - 锁相回路
- LRC - 内部低速 32 kHz RC 振荡器
- LOSC - 外部低速 32.768 kHz 时钟振荡器

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分别说明:

- RTC 时钟(RTCCLK)可以选择以下两种时钟源其中之一:
  - LRC
  - LOSC
- Chopper 时钟(CHOPCLK)可以选择以下三种时钟源其中之一:
  - HOSC
  - HRC
  - PLL
- ADC 时钟(ADCCLK)可以选择以下三种时钟源其中之一:
  - HOSC
  - HRC
  - PLL
- VLCD 时钟(VLCDCLK)可以选择以下三种时钟源其中之一:

- HOSC
- HRC
- PLL

◆ LCD 时钟(LCDCLK)可以选择以下两种时钟源其中之一:

- LRC
- LOSC

◆ 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

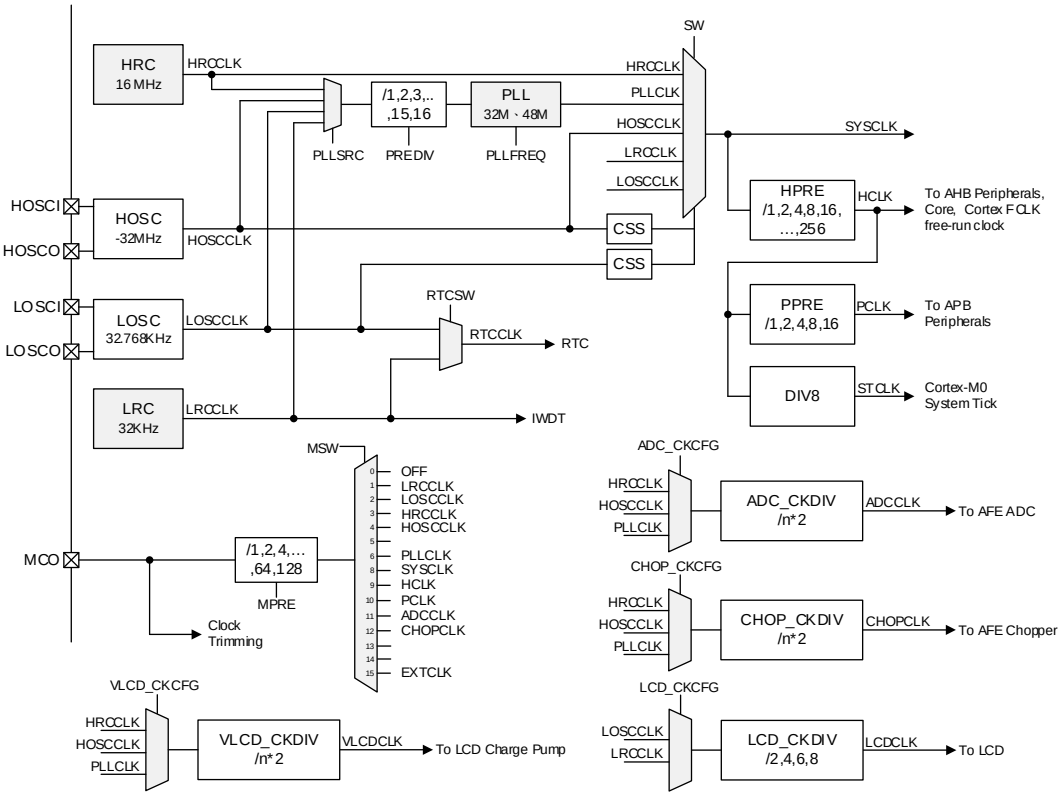


图 4-2　时钟架构图

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。

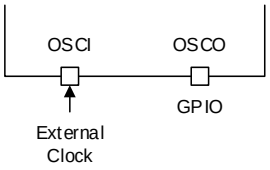
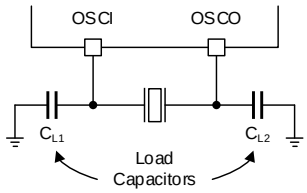
	外部电路配置状态
外部时钟源	
石英振荡器	

图 4-3　HOSC/LOSC 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 HOSCON 两个位启用 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%，相关信息请参阅电气特性章节。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，其余细节请参阅电气特性章节。配置 **RCU_CON**.HOSCON 位控制 HOSC 开启与关闭。HOSC 时钟状态可通过 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 1，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 错误地启动。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，HRC RC 振荡器的优点是提供低成本的时钟源(无外部组件)。时钟信号启动时间相较于 HOSC 来得快速，但即使频率经过校准，精准度仍不如外部晶体振荡器。通过 **RCU_CON**.HRCON 位控制 HRC 开启与关闭；HRC 时钟状态可由 HRCRDY 标志位确认，当 HRC 时钟稳定时，标志位将被硬件自动配置为 1，也可以设定 **RCU_IER**.HRCRDY 位开启 HRC 时钟稳定中断。假如 HOSC 受到任何外在因素造成无法产生时钟信号时，HRC 时钟亦可作为 HOSC 的辅助时钟，需开启时钟安全系统功能，相关信

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)

芯片提供一组锁相回路(Phase-Locked Loop，PLL)，其输入参考时钟频率，通过 **RCU_CFG**.PLLSRC 选择

- HRC
- HOSC
- LRC
- LOSC

如果选择 HRC 或 HOSC 作为输入时钟源，可以通过设定 **RCU_CFG**.PREDIV 确保输入频率为 4MHz。

输出时钟频率通过 **RCU_CFG1**.PLLFREQ 选择参考时钟倍频，倍频的大小会依照参考频率的来源而有不同

- 参考时钟为 LOSC、LRC
  - x1024
  - x1536
- 参考时钟为 HOSC、HRC，须分频成 4MHz。
  - x8
  - x12

通过 **RCU_CON**.PLLON 位控制 PLL 开启与关闭；PLL 时钟状态可由 PLLRDY 标志位确认，当 PLL 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**.PLLRDY 位开启 PLL 时钟稳定中断。RCU 针对 PLL 的参考时钟进行检查，如果输入时钟源还未稳定前，PLLRDY 也都不会动作。

PLL 在不同的参考时钟下，会有不同的稳定时间，依照 **RCU_CFG**.PLLSRC 以及 **RCU_CFG**.PREDIV 所设定的参考时钟频率，提供寄存器 **RCU_CFG1**.PLLRDYCNT 设定稳定时间。有关 PLL 特性请参阅电气特性章节。

PLL 的配置(包括参考时钟源的选择、预分频和倍率设定)必须在 PLL 开启之前完成，开启 PLL 后便无法修改配置，必须先关闭 PLL 才可以重新配置。修改 PLL 配置的流程如下:

- 关闭 PLL，设定 RCU_CON.PLLON=0
- 确认 RCU_CON.PLLRDY=0 后，PLL 才完成关闭的流程
- 根据需求重新配置
- 再次开启 PLL，设定 RCU_CON.PLLON=1
- 等待 RCU_CON.PLLRDY=1，等待 PLL 输出时钟稳定

PLLSRC[1:0]	PLLFREQ	PLL 输出频率
2'b00 (HRC)	0	PLL 输入频率 x8
2'b00 (HRC)	1	PLL 输入频率 x12
2'b01 (HOSC)	0	PLL 输入频率 x8
2'b01 (HOSC)	1	PLL 输入频率 x12
2'b10 (LRC)	0	PLL 输入频率 x1024
2'b10 (LRC)	1	PLL 输入频率 x1536
2'b11(LOSC)	0	PLL 输入频率 x1024
2'b11(LOSC)	1	PLL 输入频率 x1536

4. 3. 2. 4 外部低速振荡器时钟 (LOSCCLK)

LOSC 时钟是由外部 32.768kHz 石英晶体振荡器或一个稳定低速时钟源提供，它的优点在于提供一个低功耗且精准的时钟给外设实时时钟(RTC)计数时间与日历使用。配置 **RCU_LCON**.LOSCON 位控制 LOSC 开启与关闭。LOSC 时钟状态可通过 LOSCRDY 标志位确认，当 LOSC 时钟稳定时标志位将被硬件自动配置为 1，也可以设定 **RCU_IER**.LOSCRDY 位开启 LOSC 时钟稳定中断。

外部时钟来源 (LOSC 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_LCON**.LOSCBYP 与 LOSCON 两个位启用 LOSC。外部时钟信号源(方波、弦波或三角波)可由 LOSCI 引脚输入，输入时钟工作周期(Duty Cycle)须介于 40~60%，相关信息请参阅电气特性章节。此外，LOSCO 引脚可设定为 GPIO 功能引脚，如图 4-3 所示。

4. 3. 2. 5 内部低速RC振荡器时钟 (LRCCLK)

内部低速 RC 振荡器 LRC 是一个低功耗时钟源，并可以在低功耗模式下让外设独立看门狗(IWDT)和实时时钟(RTC)持续运行；时钟频率大约为 32kHz，相关信息请参阅电气特性章节。配置 **RCU_LCON**.LRCON 位控制 LRC 开启与关闭。LRC 时钟状态可通过 LRCRDY 标志位确认，当 LRC 时钟稳定时标志位将被硬件自动配置为 1，也可以设定 **RCU_IER**.LRCRDY 位开启 LRC 时钟稳定中断。假如 LOSC 受到任何外在因素造成无法产生时钟信号时，LRC 时钟亦可作为 LOSC 的辅助时钟，需开启时钟安全系统功能，相关信息请参阅外部低速振荡器时钟安全系统(LOSC CSS)章节。

4. 3. 2. 6 系统时钟 (SYSCLK)

系统时钟(SYSCLK)可选择的时钟源，如下:

- HRC
- HOSC
- PLL
- LRC
- LOSC

系统复位后，默认选择 HRC 作为系统时钟。当时钟源作为系统时钟时，该时钟源无法通过寄存器 **RCU_CON** 关闭。修改 **RCU_CFG**.SW 可以将系统时钟切换成其它时钟源，只有在目标时钟源开启且稳定时，才可以将系统时钟切换至目标时钟源，读取 **RCU_CFG**.SWS 可以确认当前系统时钟源。

4. 3. 2. 7 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。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 2 通道(GP16C2T 1/2/3/4)
- 产生时钟安全系统中断(CSSHOSC)

> 注:当启用时钟安全系统时且 HOSC 发生故障，则会产生时钟安全系统中断(CSSHOSC)，此中断与不可屏蔽中断(NMI)异常状态连接，因此，除非通过 RCU_ICR.CSSHOSC 清除时钟安全系统中断状态，否则 NMI 将无止尽地执行。
> 无论 HOSC 是直接或是间接作为系统时钟(间接是指 HOSC 作为 PLL 输入时钟，而且 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，并且关闭 HOSC，若 PLL 输入时钟源为 HOSC，也会关闭 PLL。

4. 3. 2. 8 外部低速振荡器时钟安全系统(LOSC CSS)

外部低速振荡器时钟安全系统是为了避免当系统时钟使用到 LOSC 时，LOSC 因任何外在因素发生故障，进而导致系统死机。可以通过 **RCU_LCON**.RCU_LCSCCSSON 开启或关闭时钟安全系统，只有当 LOSC 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 LOSC 停止时关闭。当开启时钟安全系统时，必须开启 LRC 时钟。

系统时钟有使用到 LOSC 的情况

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

当侦测到 LOSC 故障时

- 系统时钟自动切换成 LRC
- 如果 PLL 为系统时钟(LOSC 为参考频率)，将关闭 PLL。
- 时钟故障事件将被送到通用定时器 16 位 2 通道(GP16C2T 1/2/3/4)
- 产生时钟安全系统中断(CSSLOSC)
- 使用者需要自行关闭 LOSC

注:当启用时钟安全系统时且 LOSC 发生故障，则会产生时钟安全系统中断(CSSLOSC)，此中断与不可屏蔽中断(NMI)异常状态连接，因此，除非通过 RCU_ICR.CSSLOSC 清除时钟安全系统中断状态，否则 NMI 将无止尽地执行。

无论 LOSC 是直接或是间接作为系统时钟(间接是指 LOSC 作为 PLL 输入时钟，而且 PLL 作为系统时钟)，当侦测到 LOSC 故障时，系统时钟将自动切换成 LRC，若 PLL 输入时钟源为 LOSC，也会关闭 PLL。

4. 3. 2. 9 ADC时钟 (ADCCLK)

ADC 输入时钟必须为 4MHz，时钟来源可通过 **RCU_CFG2**.ADC_CKCFG 选择 HOSC、HRC 或是 PLL。使用 **RCU_CFG2**.ADC_CKDIV，可以配置分频。当 ADC 时钟启用，便无法更改 ADC 时钟的来源以及分频配置;如果想要重新配置 ADC 时钟，必须先关闭 ADC 时钟将 **RCU_CFG2**.ADC_CKCFG 设定为 0。

4. 3. 2. 10 Chopper时钟 (CHOPCLK)

模拟电路中有使用到斩波器(Chopper)电路有:

- VBG 斩波器
- VREF 斩波器
- IA 斩波器
- OPAMP 斩波器

各个斩波器的输入 chopper 时钟皆是同一个来源控制，输入时钟必须为 4 MHz。Chopper 时钟通过 **RCU_CFG2**.CHOP_CKCFG 可选 HOSC、HRC 或是 PLL。使用 **RCU_CFG2**.CHOP_CKDIV，可以配置除频。当 chopper 时钟启用，便无法更改 chopper 时钟的来源以及除频配置;如果想要重新配置 chopper 时钟，必须先关闭 chopper 时钟，将 **RCU_CFG2**.CHOP_CKCFG 设定为 0。

注:与其他斩波器时钟不同，SDADC 斩波器的时钟来源为 ADCCLK。

4. 3. 2. 11 VLCD时钟 (VLCDCLK)

VLCD 时钟通过 **RCU_CFG2**.VLCD_CKCFG 可选择 HOSC、HRC 或是 PLL。使用 **RCU_CFG2**.VLCD_CKDIV，可以配置分频。当 VLCD 时钟启用，便无法更改 VLCD 时钟的来源以及分频配置;如果想要重新配置 VLCD 时钟，必须先关闭 VLCD 时钟，将 **RCU_CFG2**.VLCD_CKCFG 设定为 0。

4. 3. 2. 12 LCD时钟 (LCDCLK)

LCD 时钟通过 **RCU_CFG2**.LCD_CKCFG 可选择 LRC 或是 LOSC。使用 **RCU_CFG2**.LCD_CKDIV，可以配置分频。当 LCD 时钟启用，便无法更改 LCD 时钟的来源以及分频配置;如果想要重新配置 LCD 时钟，必须先关闭 LCD 时钟，将 **RCU_CFG2**.LCD_CKCFG 设定为 0。

4. 3. 2. 13 RTC时钟 (RTCCLK)

RTC 时钟源可选择 LRC 或 LOSC，通过 **RTC_CTRL**.CKSEL 选择，当 RTC 启动，便无法更改时钟来源；如果想重新配置时钟来源，必须先关闭 RTC。

4. 3. 2. 14 IWDT时钟 (IWDTCLK)

IWDT 时钟只能由 LRC 提供，无论通过配置字或软件方式启动独立看门狗(IWDT)，都会强制打开 LRC 且无法关闭。

4. 3. 2. 15 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，除了提供检查以外也可当作其他装置或模块的输入时钟源。通过配置 **RCU_CFG**.MSW 选择要输出哪个时钟，根据需求可以配置 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。可以选择以下时钟输出:

- LRC 时钟
- LOSC 时钟
- HRC 时钟
- HOSC 时钟
- PLL 时钟
- 系统时钟
- AHB 时钟
- APB 时钟
- ADC 时钟
- Chopper 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
							HOSCCSSON			PLLRDY	PLLON														HOSCBYP	HOSCRDY	HOSCON			HRCRDY	HRCON

—	Bit 31-25	—	—
HOSCCSSON	Bit 24	R/W	**HOSC 时钟安全系统使能(CSS)** 当 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；相反的当 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:选择输出 LOSC 时钟(LOSCCLK) 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 Chopper 时钟(CHOPCLK) 1101:保留。 1110:保留。 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 为 PLL 输入时钟源 01:选择 HOSC 为 PLL 输入时钟源 10:选择 LRC 为 PLL 输入时钟源 11:选择 LOSC 为 PLL 输入时钟源
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 32kHz 或是 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，显示当前系统使用哪种时钟来源 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 100: 系统时钟为 LOSCCLK 其他: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 100: 选择 LOSCCLK 作为系统时钟

			其他: 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:0x1003 FE20																															
31	30	29	28	27	26	25	24	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>									PLLFREQ	|	|	HRCTUNE<5:0>						

SYSFREQ	Bit 31-24	R/W	**当前系统时钟频率** 用于记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，若填入的数值小于 4，则 SYSFREQ 的数值自动带入初始数值 4。当系统频率改为 8MHz，需修改 SYSFREQ 的数值为 8，其余频率依此类推。
—	Bit 23-18	—	—
PLLRDYCNT	Bit 17-9	R/W	**PLL 稳定计数器设置** 当 PLL 关闭(PLLON=0)时，允许改变设置。计数器的频率单位，RCU_CFG.PLLSRC 以及 RCU_CFG.PREDIV 所设定的参考时钟频率。
PLLFREQ	Bit 8	R/W	**PLL 时钟频率选择** 当 PLL 关闭(PLLON=0)时，允许改变设置。将视 RCU_CFG.PLLSRC 所选择的输入时钟来源而定。 输入时钟为 LRC 或 LOSC 0 : 1024 倍频 1 : 1536 倍频 输入时钟为 HRC 或 HOSC 0 : 8 倍频 1 : 12 倍频
—	Bit 7-6	—	—
HRCTUNE	Bit 5-0	R/W	**HRC 频率调整** 0x00: 14.7 MHz

			0x20: 16 MHz 0x3F: 17.2 MHz

4. 4. 2. 4 时钟配置寄存器 2 (RCU_CFG2)

时钟配置寄存器 2 (RCU_CFG)																													
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
—	—	—	—	—	—	—	—	CHOP_CKDIV<3:0>				ADC_CKDIV<3:0>				—	—	LCD_CKDIV<1:0>		VLCD_CKDIV<3:0>				CHOP_CKCFG<1:0>		ADC_CKCFG<1:0>		LCD_CKCFG<1:0>		VLCD_CKCFG<1:0>	

—	Bit 31-24	—	—
CHOP_CKDIV	Bit 23-20	R/W	**选择 Chopper 时钟分频器** Note: 只有在 Chopper 时钟配置为禁用的情况下，才能更改配置。 Chopper 时钟除频 = (CHOP_CKDIV*2)
ADC_CKDIV	Bit 19-16	R/W	**选择 ADC 时钟分频器** Note:只有在 ADC 时钟配置为禁用的情况下，才能更改配置。 ADC 时钟除频 = (ADC_CKDIV*2)
—	Bit 15-14	—	—
LCD_CKDIV	Bit 13-12	R/W	**选择 LCD 时钟分频器** Note ： 只有 LCD 时钟配置为禁用的情况下，才能更改配置。 00 : LCD 时钟除 2 01 : LCD 时钟除 4 10 : LCD 时钟除 6 11 : LCD 时钟除 8
VLCD_CKDIV	Bit 11-8	R/W	**选择 VLCD 时钟分频器** Note ： 只有在 VLCD 时钟配置为禁用的情况下，才能更改配置。 0 : Bypass n : Divider values is = n*2, Min.1 Max.63
CHOP_CKCFG	Bit 7-6	R/W	**Chopper 时钟配置寄存器** Note ： 只有在 Chopper 时钟配置为禁用的情况下，才能更改配置。 00: 禁用

			01: 选择 HOSC 时钟 10: 选择 HRC 时钟 11: 选择 PLL 时钟
ADC_CKCFG	Bit 5-4	R/W	**ADC 时钟配置寄存器** Note: 只有在 ADC 时钟配置为禁用的情况下，才能更改配置。 00: 禁用 01: 选择 HOSC 时钟 10: 选择 HRC 时钟 11: 选择 PLL 时钟
LCD_CKCFG	Bit 3-2	R/W	**LCD 时钟配置寄存器** Note: 只有在 LCD 时钟配置为禁用的情况下，才能更改配置。 00: 禁用 01: 选择 LRC 时钟 10: 选择 LOSC 时钟 11: 保留
VLCD_CKCFG	Bit 1-0	R/W	**VLCD 时钟配置寄存器** Note: 只有在 VLCD 时钟配置为禁用的情况下，才能更改配置。 00: 禁用 01: 选择 HOSC 时钟 10: 选择 HRC 时钟 11: 选择 PLL 时钟

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
—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	CSSLOSC	CSSHOSC	—	—	PLLRDY	—	HOSCRDY	HRCRDY	LOSCRDY	LRCRDY

—	Bit 31-10	—	—
CSSLOSC	Bit 9	W1	**开启LOSC时钟安全系统中断功能** 此位设置时，开启中断功能，当时钟检测功能开启，而LOSC失效时发生中断
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
LOSCRDY	Bit 1	W1	**开启 LOSC 时钟源稳定中断功能** 此位设置时，开启中断功能，当 LOSC 时钟稳定后发生中断
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
—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	CSSLOSC	CSSHOSC	—	—	PLLRDY	—	HOSCRDY	HRCRDY	LOSCRDY	LRCRDY

—	Bit 31-10	—	—
CSSLOSC	Bit 9	W1	**关闭LOSC时钟安全系统中断功能** 此位设置时，关闭时钟安全系统中断功能
CSSHOSC	Bit 8	W1	**关闭时钟安全系统中断功能** 此位设置时，关闭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
LOSCRDY	Bit 1	W1	**关闭 LOSC 时钟源稳定中断功能** 此位设置时，关闭 LOSC 时钟稳定中断功能
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
																						CSSLOSC	CSSHOSC			PLLRDY		HOSCRDY	HRCRDY	LOSCRDY	LRCRDY

—	Bit 31-10	—	—
CSSLOSC	Bit 9	R	**LOSC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
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
LOSCRDY	Bit 1	R	**LOSC 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
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 0004																												
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																						CSSLOSC	CSSHOSC			PLLRDY		HOSCRDY	HRCRDY	LOSCRDY	LRCRDY

—	Bit 31-10	—	—
CSSLOSC	Bit 9	R	**LOSC时钟安全系统，原始中断状态** 0: 无发生中断 1: 已发生中断
CSSHOSC	Bit 8	R	**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	R	**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
LOSCRDY	Bit 1	R	**LOSC 时钟源稳定时，原始中断状态** 0:无发生中断 1:已发生中断
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
—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	CSSLOSC	CSSHOSC	—	—	PLLRDY	—	HOSCRDY	HRCRDY	LOSCRDY	LRCRDY

—	Bit 31-10	—	—
CSSLOSC	Bit 9	R	**LOSC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
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
LOSCRDY	Bit 1	R	**LOSC 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
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
—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	CSSLOSC	CSSHOSC	—	—	PLL0RDY	HRC48RDY	HOSCRDY	HRCRDY	LOSCRDY	LRCRDY

—	Bit 31-10	—	—
CSSLOSC	Bit 9	C_W1	**清除LOSC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)
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)
LOSCRDY	Bit 1	C_W1	**清除 LOSC 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)
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
												GPDEN	GPCEN	GPBEN	GPAEN	CALCEN									RTCEN						

—	Bit 31-20	—	—
GPDEN	Bit 19	R/W	**GPIOD 复位需求** 0:取消复位需求 1:提交复位需求
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 14-7	—	—
RTCEN	Bit 6	R/W	**RTC 复位需求** 0:取消复位需求 1:提交复位需求
—	Bit 5-0	—	—

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
										I2C1EN		UART4EN	UART3EN	UART2EN						WWDTEN							BS16T1EN				GP32C4T1EN

—	Bit 31-22	—	—
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	—	—
UART4EN	Bit 19	R/W	**UART4 复位需求** 0:取消复位需求 1:提交复位需求
UART3EN	Bit 18	R/W	**UART3 复位需求** 0:取消复位需求 1:提交复位需求
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 3-1	—	—
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
—	—	—	—	—	—	—	—	CMPEN	—	—	—	GP16C2T4EN	GP16C2T3EN	GP16C2T2EN	GP16C2T1EN	—	UART1EN	—	SPI1EN	—	—	—	—	—	—	OPAMPEN	ANPWREN	ADCEN	LCDEN	—	—

—	Bit 31-24	—	—
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 22-20	—	—
GP16C2T4EN	Bit 19	R/W	**GP16C2T4 复位需求** 0:取消复位需求 1:提交复位需求
GP16C2T3EN	Bit 18	R/W	**GP16C2T3 复位需求** 0:取消复位需求 1:提交复位需求
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
—	Bit 13	—	—
SPI1EN	Bit 12	R/W	**SPI1 复位需求** 0:取消复位需求 1:提交复位需求
—	Bit 11-6	—	—
OPAMPEN	Bit 5	R/W	**OPAMP 复位需求** 0: 取消复位需求

			1: 提交复位需求
ANPWREN	Bit 4	R/W	**ANPWR 复位需求** 0: 取消复位需求 1: 提交复位需求
ADCEN	Bit 3	R/W	**ADC 复位需求** 0: 取消复位需求 1: 提交复位需求
LCDEN	Bit 2	R/W	**LCD 复位需求** 0: 取消复位需求 1: 提交复位需求
—	Bit 1-0	—	—

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
												GPDEN	GPCEN	GPBEN	GPAEN	CALCEN									RTCEN						

—	Bit 31-20	—	—
GPDEN	Bit 19	R/W	**GPIOD 时钟开关** 0:关闭时钟 1:开启时钟
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 :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 14-7	—	—
RTCEN	Bit 6	R/W	**RTC 时钟开关** 0:关闭时钟 1:开启时钟
—	Bit 5-0	—	—

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
										I2C1EN		UART4EN	UART3EN	UART2EN						WWDTEN							BS16T1EN				GP32C4T1EN

—	Bit 31-22	—	—
I2C1EN	Bit 21	R/W	**I2C1 时钟开关** 0:关闭时钟 1:开启时钟
—	Bit 20	—	—
UART4EN	Bit 19	R/W	**UART4 时钟开关** 0:关闭时钟 1:开启时钟
UART3EN	Bit 18	R/W	**UART3 时钟开关** 0:关闭时钟 1:开启时钟
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 3-1	—	—
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
—	—	—	—	—	—	—	—	CMPEN	—	—	—	GP16C2T4EN	GP16C2T3EN	GP16C2T2EN	GP16C2T1EN	—	UART1EN	—	SPI1EN	—	—	—	—	—	—	OPAMPEN	ANPWREN	ADCEN	LCDEN	—	—

—	Bit 31-24	—	—
CMPEN	Bit 23	R/W	**CMP 时钟开关** 0:关闭时钟 1:开启时钟
—	Bit 22-20	—	—
GP16C2T4EN	Bit 19	R/W	**GP16C2T4 时钟开关** 0:关闭时钟 1:开启时钟
GP16C2T3EN	Bit 18	R/W	**GP16C2T3 时钟开关** 0:关闭时钟 1:开启时钟
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
—	Bit 13	—	—
SPI1EN	Bit 12	R/W	**SPI1 时钟开关**

			0:关闭时钟 1:开启时钟
—	Bit 11-6	—	—
OPAMPEN	Bit 5	R/W	**OPAMP 时钟开关** 0: 关闭时钟 1: 开启时钟
ANPWREN	Bit 4	R/W	**ANPWR 时钟开关** 0: 关闭时钟 1: 开启时钟
ADCEN	Bit 3	R/W	**ADC 时钟开关** 0: 关闭时钟 1: 开启时钟
LCDEN	Bit 2	R/W	**LCD 时钟开关** 0: 关闭时钟 1: 开启时钟
—	Bit 1-0	—	—

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
												GPDEN	GPCEN	GPBEN	GPAEN	CALCEN									RTCEN						

—	Bit 31-20	—	—
GPDEN	Bit 19	R/W	**SLEEP 模式时，GPIOD 时钟开关** 0:关闭时钟 1:开启时钟
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 14-7	—	—
RTCEN	Bit 6	R/W	**SLEEP 模式时，RTC 时钟开关** 0:关闭时钟 1:开启时钟
—	Bit 5-0	—	—

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
										I2C1EN		UART4EN	UART3EN	UART2EN						WWDTEN							BS16T1EN				GP32C4T1EN

—	Bit 31-22	—	—
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	—	—
UART4EN	Bit 19	R/W	**SLEEP 模式时，UART4 时钟开关** 0:关闭时钟 1:开启时钟
UART3EN	Bit 18	R/W	**SLEEP 模式时，UART3 时钟开关** 0:关闭时钟 1:开启时钟
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 3-1	—	—
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
—	—	—	—	—	—	—	—	CMPEN	—	—	—	GP16C2T4EN	GP16C2T3EN	GP16C2T2EN	GP16C2T1EN	—	UART1EN	—	SPI1EN	—	—	—	—	—	—	OPAMPEN	ANPWREN	ADCEN	LCDEN	—	—

—	Bit 31-24	—	—
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 22-20	—	—
GP16C2T4EN	Bit 19	R/W	**SLEEP 模式时，GP16C2T4 时钟开关** 0:关闭时钟 1:开启时钟
GP16C2T3EN	Bit 18	R/W	**SLEEP 模式时，GP16C2T3 时钟开关** 0:关闭时钟 1:开启时钟
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
—	Bit 13	—	—
SPI1EN	Bit 12	R/W	**SLEEP 模式时，SPI1 时钟开关** 0:关闭时钟 1:开启时钟
—	Bit 11-6	—	—
OPAMPEN	Bit 5	R/W	**SLEEP 模式时，OPAMP 时钟开关** 0: 关闭时钟 1: 开启时钟
ANPWREN	Bit 4	R/W	**SLEEP 模式时，ANPWR 时钟开关** 0: 关闭时钟 1: 开启时钟
ADCEN	Bit 3	R/W	**SLEEP 模式时，ADC 时钟开关** 0: 关闭时钟 1: 开启时钟
LCDEN	Bit 2	R/W	**SLEEP 模式时，LCD 时钟开关** 0: 关闭时钟 1: 开启时钟
—	Bit 1-0	—	—

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 4000 (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
																	LOSCITUNE<2:0>			LOSCCSSON	LOSCBYP	LOSCRDY	LOSCON						LRCRDY	LRCON

—	Bit 31-15	—	—
LOSCITUNE	Bit 14-12	R/W	**LOSC时钟电流调整** 此位只能在LOSC关闭时(LOSCON=0)，允许修改 000: 180nA

			001: 240nA 010: 300nA 011: 360nA 100: 420nA 101: 480nA 110: 540nA 111: 600nA
LOSCCSSON	Bit 11	R/W	**LOSC时钟安全系统使能(CSS)** 当LOSC时钟信号稳定就绪时(LOSCRDY=1)，可软件控制开启安全保护功能；相反的当LOSC时钟信号尚未稳定(LOSCRDY=0)，此位无法被开启。使用LOSC时钟安全系统，软件必须使能LRC。 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LOSC=1).
LOSCBYP	Bit 10	R/W	**外部低速时钟振荡器，旁路模式使能** 此位只能在 LOSC 关闭时(LOSCON=0)，允许写入 0:关闭 LOSC 旁路模式(振荡器模式) 1:开启 LOSC 旁路模式(使用外部输入时钟源，通过 LOSCI 引脚)
LOSCRDY	Bit 9	R	**LOSCCLK 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: LOSC 时钟信号，尚未稳定 1: LOSC 时钟信号，已准备就绪
LOSCON	Bit 8	R/W	**外部低速时钟振荡器，LOSCCLK 时钟源使能** 0:关闭 LOSC 1:开启 LOSC
—	Bit 7-2	—	—
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

嵌入式闪存依据芯片型号不同，最多支持 128KB 供用户存放应用程序(Application Code)或是储存数据。闪存控制器允许使用者通过在线系统编程器(ISP)、SWD、Bootrom 或是闪存内的程序，修改已焊接于 PCB 板上芯片的数据。

5. 2 特性

- 程序区大小依据芯片型号而定，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 为单位进行页擦除，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。
  - ◇ 系统频率超过 72Mhz 时，配置 3 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

闪存共分为 3 个区块，分别为程序区(Main Block)、系统内存区(System Memory)与信息区(Information Block)。

- 程序区：大小依据芯片型号而定，最大支持 128KB。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，最多支持 256 个页。
- 系统内存区：大小为 4K Byte，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，内存位置为 0x1FFF_E000 至 0x1FFF_EFFF。信息区内的最小单位为页(Page)，每一页大小为 512 Byte，最多支持 8 个页，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使用者校准信息。此区的内容允许使用者读取与修改。
  - ◇ 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，此区支持用户程序自行配置与修改，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用户代码读出保护(UCRP)。当保护被开启以后，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，仅能通过 UCRP 保护设定流程进行配置。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，读保护的主要功能为防止程序区被 ISP、Debug Port 与 Bootrom 读出与修改，共分为 3 个等级。芯片在经过 CP 测试后会将读保护设定填入 0xAAAA_AAAA，因此默认的读保护等级为 Lv0。若对信息区页 1 进行擦除或是设定非 0xCCCC_CCCC 或 0xAAAA_AAAA 的数值时，则读保护等级会提升至 Lv1 保护。此区无法通过闪存的编程指令修改保护设定，需通过读保护设定流程进行配置。有关读保护的配置方式与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

此页主要存放写保护设定、硬件映射设定、系统欠压复位设定与 IWDT 设定。

写保护主要目的为防止程序区内的保护区被擦除或是覆盖。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。此区无法通过闪存的编程指令修改保护设定，需通过写保护设定流程来进行配置。有关写保护的配置方式与功能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。

当使用者不再需要使用 Bootrom 时，可对信息区页 2 位于 0x412 的 BOOTBYP 编程 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。若信息区 0x412 的位置已被编程 0xA5 时，用户仅能通过程序区的程序，或是使用 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 主要为提供映射信息给 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、System Memory 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。用户可配置位于信息区 0x411 的 SELECT 为 0x0，同时依据需求配置数值 0x0 到 0xFF 至信息区 0x410 的 SEFBASE，如果设置 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，如果设置为 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 也支持使用者在系统开机时，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。 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 Data sheet 内电气特性章节描述。 000:BOR level 0 001: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，可由用户自行配置。在降读保护等级时，此页的数据会一并被清除。

5. 4. 2 系统配置字

系统配置字主要存放在信息区页 4，内容包含芯片校准信息、产品标识符与芯片唯一标识符，此区的信息会在芯片出厂前针对每一颗芯片进行调整，使用者可依据需求读取此区的信息但无法修改。信息区页 5 至页 7 为保留区域。详细的系统配置字说明请参阅后续章节内容。

5. 4. 2. 1 系统配置字Bandgap 1V2 校准

此配置字存放 Bandgap 1V2 的校准信息与校准数值的取反值，在不满足取反时自动带入校准数值 0x8。

配置字 Bandgap1V2 校准																													
偏移地址:0x804																													
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
REV<15:0>																-	-	-	-	-	-	-	-	-	-	-	-	BG1V2TRIM<3:0>			

REV	Bits 31-16	**数值取反** 第 0 位到第 15 位的数值取反。
—	Bits 15-4	—
BG1V2TRIM	Bits 3-0	**Bandgap 1V2 校准值** 此数值在芯片出厂前已针对芯片特性进行调整，并于系统开机后自动从闪存内读出。

5. 4. 2. 2 系统配置字Bandgap 1V校准

此配置字存放 Bandgap 1V 的校准信息与校准数值的取反值，在不满足取反时自动带入校准数值 0x8。

配置字 Bandgap 1V 校准																													
偏移地址:0x808																													
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
REV<15:0>																-	-	-	-	-	-	-	-	-	-	-	-	BG1VTRIM<3:0>			

REV	Bits 31-16	**数值取反** 第 0 位到第 15 位的数值取反。

—	Bits 15-4	—
BG1VTRIM	Bits 3-0	**Bandgap 1V 校准值** 此数值在芯片出厂前已针对芯片特性进行调整，并于系统开机后自动从闪存内读出。

5. 4. 2. 3 系统配置字LDONP校准

此配置字存放 LDONP 的校准信息与校准数值的取反值，此校准信息在不满足取反时自动带入数值 0x8。

配置字 LDONP 校准

偏移地址:0x80C

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
REV<15:0>																												LDONPTRIM<3:0>			

REV	Bits 31-16	**数值取反** 第 0 位到第 15 位的数值取反。
—	Bits 15-4	—
LDONPTRIM	Bits 3-0	**LDONP 校准值** 此数值在芯片出厂前已针对芯片特性进行调整，并于系统开机后自动从闪存内读出。

5. 4. 2. 4 系统配置字LDOLP校准

此配置字存放 LDOLP 的校准信息与校准数值的取反值，此校准信息在不满足取反时自动带入数值 0x8。

配置字 LDOLP 校准

偏移地址:0x810

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
REV<15:0>																												LDOLPTRIM<3:0>			

REV	Bits 31-16	**数值取反** 第 0 位到第 15 位的数值取反。

—	Bits 15-4	—
LDOLPTRIM	Bits 3-0	**LDOLP 校准值** 此数值在芯片出厂前已针对芯片特性进行调整，并于系统开机后自动从闪存内读出。

5. 4. 2. 5 系统配置字LVD校准

此配置字存放低电压检测器(LVD)的校准信息与校准数值的取反值，此校准信息在不满足取反时自动带入数值 0x2。

配置字 LVD 校准

偏移地址:0x814

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
REV<15:0>																														LVDTRIM<1:0>	

REV	Bits 31-16	**数值取反** 第 0 位到第 15 位的数值取反。
—	Bits 15-2	—
LVDTRIM	Bits 1-0	**LVD 校准值** 此数值在芯片出厂前已针对芯片特性进行调整，并于系统开机后自动从闪存内读出。

5. 4. 2. 6 系统配置字HRC校准L

此配置字存放 HRC 时钟校准 L 的信息与校准数值的取反值，芯片出厂前会将 HRC 校准至接近 16MHz。当此校准信息不满足取反时自动带入数值 0x80。

配置字 HRC 校准 L

偏移地址:0x818

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
REV<15:0>																								HRCTRIML<7:0>							

REV	Bits 31-16	**数值取反** 第 0 位到第 15 位的数值取反。

—	Bits 15-8	—
HRCTRIML	Bits 7-0	**HRC 校准值 L** 此数值在芯片出厂前已针对芯片特性进行调整，并于系统开机后自动从闪存内读出。

5. 4. 2. 7 系统配置字LRC校准

此配置字存放 LRC 时钟的校准信息与校准数值的取反值，芯片出厂前会将 LRC 校准至接近 32KHz。当此校准信息不满足取反时自动带入数值 0x50。

配置字 LRC 校准

偏移地址:0x820

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
REV<15:0>																									LRCTRIM<6:0>						

REV	Bits 31-16	**数值取反** 第 0 位到第 15 位的数值取反。
—	Bits 15-7	—
LRCTRIM	Bits 6-0	**LRC 校准值** 此数值在芯片出厂前已针对芯片特性进行调整，并于系统开机后自动从闪存内读出。

5. 4. 2. 8 系统配置字AFE BG校准

此配置字存放 AFE BG 的校准信息，当此校准信息不满足取反时自动带入数值 0x40。

配置字 AFE BG 校准

偏移地址:0x824

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
REV<15:0>																												AFEVBGTRIM<6:0>			

REV	Bits 31-16	**数值取反** 第 0 位到第 15 位的数值取反。
—	Bits 15-7	—

AFEVBGTRIM	Bits 6-0	**AFE BG 校准值** 此数值在芯片出厂前已针对芯片特性进行调整，并于系统开机后自动从闪存内读出。

5. 4. 2. 9 系统配置字CMP校准

此配置字存放 CMP 的校准信息，当此校准信息不满足取反时自动带入数值 0x78。

配置字 CMP 校准

偏移地址:0x828

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
REV<15:0>																—	—	—	—	—	—	—	—	CMPTRIM<7:0>							

REV	Bits 31-16	**数值取反** 第 0 位到第 15 位的数值取反。
—	Bits 15-8	—
CMPTRIM	Bits 7-0	**CMP 校准值** 此数值在芯片出厂前已针对芯片特性进行调整，并于系统开机后自动从闪存内读出。

5. 4. 2. 10 系统配置字HOSC校准

此配置字存放 HOSC 的校准信息。当取反不满足时，HOSCRDYSEL 自动带入 0x4，而 HOSCCURSEL 自动带入 0x3。

配置字 HOSC 校准

偏移地址:0x858

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
REV<15:0>																—	—	—	—	—	HOSCRDYSEL<1:0>			—	—	—	—	—	—	HOSCCURSEL<1:0>	

REV	Bit 31-16	**数值取反** 第 0 位到第 15 位的数值取反。
—	Bit 15-11	—
HOSCRDYSEL	Bit 10-8	**HOSC 稳定计数值选择**

		存放 HOSC 稳定计数值，计数周期满后系统才会拉高 HOSC 时钟的 Ready 标志位。此数值在芯片出厂前已针对芯片特性进行调整，并于系统开机后自动从闪存内读出。 0x0 : HOSC 计数 256 个周期后拉高 Ready 标志位。 0x1 : HOSC 计数 512 个周期后拉高 Ready 标志位。 0x2 : HOSC 计数 1024 个周期后拉高 Ready 标志位。 0x3 : HOSC 计数 2048 个周期后拉高 Ready 标志位。 0x4 : HOSC 计数 4096 个周期后拉高 Ready 标志位。(默认值) 0x5 : HOSC 计数 8192 个周期后拉高 Ready 标志位。 0x6 : HOSC 计数 16384 个周期后拉高 Ready 标志位。 0x7 : HOSC 计数 32768 个周期后拉高 Ready 标志位。
—	Bit 7-2	—
HOSCCURSEL	Bit 1-0	**HOSC 电流选择** 此数值在芯片出厂前已针对芯片特性进行调整，并于系统开机后自动从闪存内读出。

5. 4. 2. 11 系统配置字LOSC校准

此配置字存放 LOSC 的校准信息。当取反不满足时，LOSCRDYSEL 自动带入 0x1，而 LOSCCURSEL 自动带入 0x4。

配置字 LOSC 校准

偏移地址:0x85C

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
REV<15:0>																—	—	—	—	—	—	LOSCRDYSEL<1:0>		—	—	—	—	—	LOSCCURSEL<2:0>		

REV	Bit 31-16	**数值取反** 第 0 位到第 15 位的数值取反。
—	Bit 15-10	—
LOSCRDYSEL	Bit 9-8	**LOSC 稳定计数值选择** 存放 LOSC 稳定计数值，计数周期满后系统才会拉高 LOSC 时钟的 Ready 标志位。此数值在芯片出厂前已针对芯片特性进行调整，并于系统开机后自动从闪存内读出。 0x0: LOSC 计数 8192 个周期后拉高 Ready 标志位。 0x1:LOSC 计数 16384 个周期后拉高 Ready 标志位。(默认值) 0x2:LOSC 计数 32768 个周期后拉高 Ready 标志位。

		0x3:LOSC 计数 65536 个周期后拉高 Ready 标志位。
—	Bit 7-3	—
LOSCCURSEL	Bit 2-0	**LOSC 电流选择** 此数值在芯片出厂前已针对芯片特性进行调整，并于系统开机后自动从闪存内读出。

5. 4. 2. 12 配置字ADC常温电压量测

此配置字存放 ADC 温度传感器在 30°C下的量测数值。

配置字 ADC 常温电压量测

偏移地址:0x860

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
								ADCTEMP<23:0>																							

—	Bits 31-24	—
ADCTEMP	Bits 23-0	**ADC 温度传感器常温量测数值** 存放 ADC 量测温度传感器常温(30°C)时的量测数值。

5. 4. 2. 13 配置字ADC参考电压(VREFINT)量测

此配置字存放 30℃下的 ADC 参考电压(VREFINT)的量测数值。

配置字 ADC 参考电压(VREFINT)量测值

偏移地址:0x864

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
								ADCVREF<23:0>																							

—	Bits 31-24	—
ADCVREF	Bits 23-0	**ADC 参考电压(VREFINT)量测值** 存放 30℃下 ADC 量测参考电压(VREFINT)的量测值。

5. 4. 2. 14 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:0x8A0																														
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. 15 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，使用者可通过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:0x8A4																														
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:0x8A8																													
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:0x8AC																													
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:0x8B0																													
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在锁定状态，使用者无法通过闪存控制器对闪存进行编程与擦除，藉此避免闪存内的数据被意外擦除或是覆盖。若使用者需要使用闪存的编程与擦除功能时，需要先对闪存控制器进行解锁，解锁流程需连续输入 2 组解锁密钥，若输入错误的密钥或是连续输入超过 2 组密钥时，闪存控制器会继续保持在上锁状态或是重新锁定。当完成解锁流程后，用户可通过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。

当使用者完成闪存的编程与擦除流程时，建议用户重新对 **FC_UL** 填入 0x0000_0000 将闪存控制器重新上锁，藉此避免闪存数据被意外擦除或是覆盖。

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是被误修改，依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)。信息区依据不同区块有不同程度的保护，主要目的为防止系统校准信息被修改或是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:存放使用者代码读出保护设定，此页仅能读取无法编程，若对此页进行页擦除则视为清除使用者代码读出保护设定，此时会将使用者代码读出保护区内的数据全部清除。开启使用者代码读出保护需通过特定流程开启，无法通过编程信息区页 0 开启保护功能。
- 页 1:存放读保护设定。读保护需通过特定流程开启，无法通过编程信息区页 1 开启保护功能。有关读保护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视为清除写保护设定，清除写保护设定时仍会保留保护区内的数据。开启写保护需通过特定流程开启，无法通过编程信息区页 2 开启保护功能。此外，此页也存放硬件映射设定、系统欠压复位(BOR)设定与 IWDT 设定，有关寄存器说明，请参阅用户配置字。
- 页 3:此页不受到任何保护，可存放使用者数据。
- 页 4 ~ 页 7:存放系统校准信息，此区不开放给使用者修改，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，而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	使用者定义
	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
	主闪存重映像地址选择	0x1FFF_E410	用户定义
	软件重映像选项	0x1FFF_E411	用户定义
	硬件重映像选项	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	使用者定义
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

程序区的保护依据功能可分为用户代码读出保护(UCRP)、读保护(RP)与写保护(WP)，除了读保护以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。举例来说，若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 时，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以后都不会立反映，用户需通过配置字重载流程、重新上电或是从 STANDBY 模式或 SHUTDOWN 模式唤醒后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:

使用者代码读出保护(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，因此保护区内禁止编程与擦除。

UCRP 的设定主要存放于信息区页 0。若用户需要清除保护设定时，则须通过擦除信息区页 0 来清除保护设定，须注意的是清除 UCRP 设定时，会一并将保护区内的数据擦除，避免原先受到保护的数据被读出，但原先未受到 UCRP 保护的 Page 内的数据仍会被保留，同时 UCRP 保护设定会在信息区页 0 擦除完毕后，立即更新为未保护的状态。

开启 UCRP 的流程如下所示，配置流程结束后保护设定并不会立即反映，用户需通过配置字重载、重新上电或是从 STANDBY 模式或 SHUTDOWN 模式唤醒后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 开启配置 UCRP0 流程，若填入 0xF6 则开启配置 UCRP1 的流程。在进行保护配置的过程中，硬件会自动于相对应的位置填入取反的数据。
- 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)
- 通过配置字重载、重新上电或是从 STANDBY 模式或 SHUTDOWN 模式唤醒来反映新的保护设定。

读保护(RP)

读保护的主要功能为防止程序区被 ISP、Debug Port 与 Bootrom 读出与修改，但保护区内的程序仍可读取与修改保护区内的数据。读保护共分为 3 个等级，分别如下所示:

- Lv0:不保护，保护设定数值为 0xAAAA_AAAA。可通过擦除信息区页 1 将保护等级提升至 Lv1 保护。用户可通过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与修改，但存放保护设定的信息区则不受限制。可通过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、写保护设定与存放于信息区页 3 内的用户数据，而在设定完毕以后，仅能通过重新上电来触发保护更新。当用户确认不会再通过 Debug Port 与 Bootrom 修改闪存程序区的内容时，可通过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:保护设定数值为 0xCCCC_CCCC 时开启此保护模式。开启此模式后无法再将保护降回 Lv1 保护或是 Lv0 保护，因此在使用上需特别小心。开启保护后会断开 Debug Port，同时系统强制映射在程序区，因此用户无法在通过 Debug Port 与 Bootrom 重新修改程序区的内容，但用户仍可通过程序区内事先写好的更新流程来修改程序区的内容。

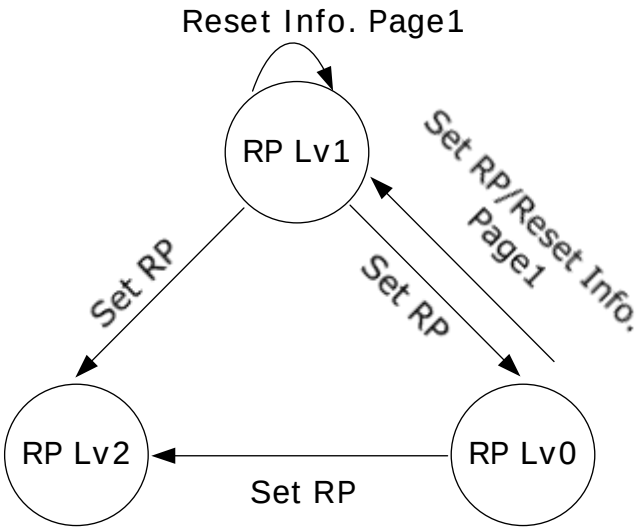


图 5-1　读保护等级转换示意图

读保护的配置流程如下所示，配置流程结束后保护设定并不会立即反映，用户需通过配置字重载、重新上电或是从 STANDBY 模式或 SHUTDOWN 模式唤醒后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)
- 通过配置字重载、重新上电或是从 STANDBY0 模式、STANDBY1 模式或 SHUTDOWN 模式唤醒来反映新的保护设定。

写保护(WP)

写保护的主要目的为防止程序区内的数据被误擦除或是被覆盖，因此受到保护的区域会禁止进行编程与擦除，但保护区内的数据仍可被读取。

写保护的设定存放于信息区页 2。当用户对信息区页 2 进行擦除时视为清除写保护，在清除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。

开启写保护流程如下所示，配置流程结束后保护设定并不会立即反映，用户需通过配置字重载、重新上电或是从 STANDBY 模式或 SHUTDOWN 模式唤醒后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 0xF8 开启配置 WP0 流程，若填入 0xF9 则开启配置 WP1 的流程。在进行保护配置的过程中，硬件会自动于相对应的位置填入取反的数据。
- 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)
- 通过配置字重载、重新上电或是从 STANDBY 模式或 SHUTDOWN 模式唤醒来反映新的保护设定。

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

当用户设定从闪存的程序区开启系统程序时，可进一步设定闪存的内部映射。内部映射的实现方式主要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，可将程序区分为 32 个区块，使用者可通过设定 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。设定流程如下，以映射至程序区的第 3 个区块为例。

- 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只有在读取闪存的数据时才会进行映射，当对闪存进行编程与擦除时不会受到内部映射影响。此外若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。

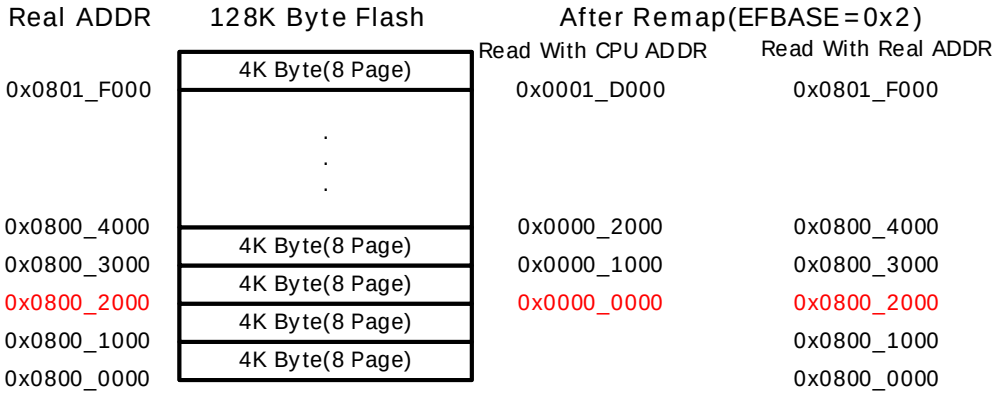


图 5-2 闪存映射后读取位置对照图

5. 4. 6 配置字重载

当用户配置保护以后，可通过闪存控制器重启配置字重载流程，藉此反映新的保护设定。须注意的是重启配置字重载流程时会一并重置系统，让程序重新运行。当程序区的程序重新运行时，可检查 Reset and Clock Control Unit (RCU)内 **RCU_RSTF** 的 OBLRSTF 是否被设定为 1 来确认系统是否有被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)

当用户需要配置 Bootrom Bypass 时，可参阅以下流程。其余用户配置字的配置方式也可参考以下流程。须注意编程完毕以后，需要通过重上电或配置字重载等流程让系统重载 Bootrom Bypass 设定。

- 解锁闪存控制器。(请参阅闪存操作解锁)
- 对 **FC_PA** 填入 0x410 。
- 设定 **FC_PA** 位于第 24 位的 IFREN 为 1，选择编程信息区。
- 对 **FC_PLD** 填入 0xFFA5_FFFF。需要注意的是在编程前，必须确定信息区 0x410 的数据为 0xFFFF_FFFF，否则该位置的编程结果可能会与预期不符。
- 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。
- 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** 通过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，使用者仅需填写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	**信息区块使能** 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	BUFEN		OPRLD<3:0>						WAIT<1:0>	

—	Bits 31-11	—	—
FCSLEEP	Bits 10	R/W	**停止模式开关** 当系统进入STOP0/STOP1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。
BUFEN	Bits 9	R/W	**闪存读取缓存开关** 0x0:关闭闪存读取缓存。 0x1:开启闪存读取缓存。
—	Bits 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 3-2	—	—
WAIT	Bits 1-0	R/W	**闪存读取等待周期** 闪存读取时间固定40ns，通过配置此寄存器，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。 0x3:系统频率>72Mhz。

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** 记录开机后系统加载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** 记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** 记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。 其他: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
- 每个 IO 端口的驱动电流可选
- 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

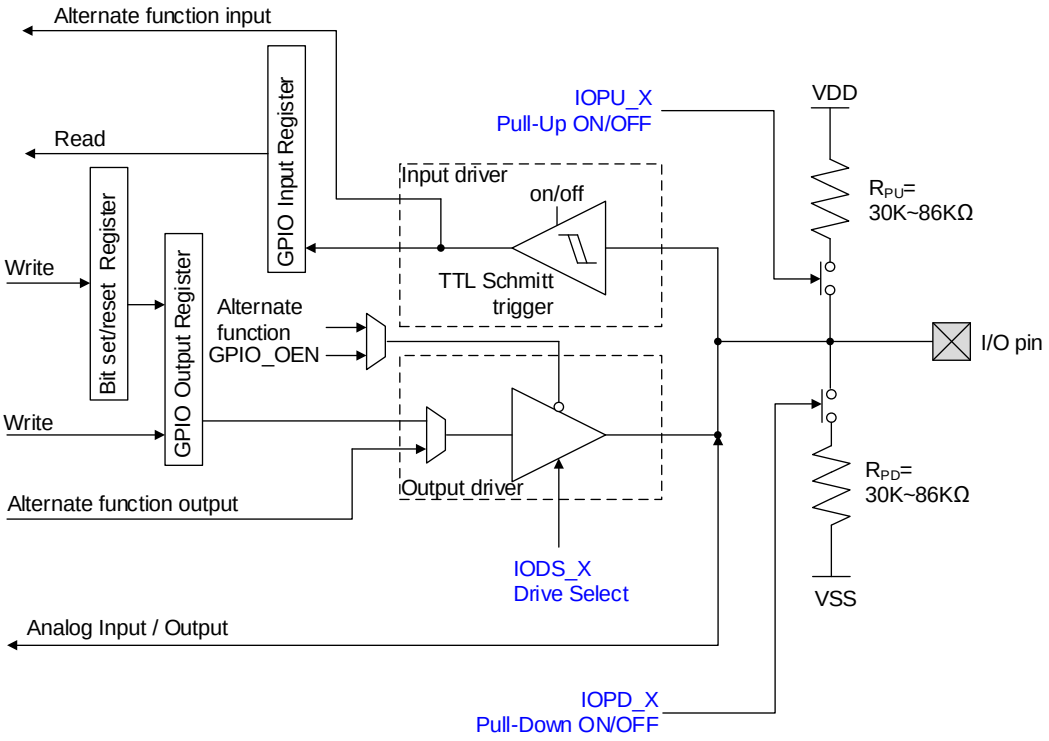


图 6-1 I/O 端口位的基本结构图

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，并且 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分别配置。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，它们被激活或断开取决于 **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接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)连接至 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:

- 除错功能: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 RTC 或 WKUPx，在 RTC 或 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。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 时则置位相应的 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，使用者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 上复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。要使用外部中断信道，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。

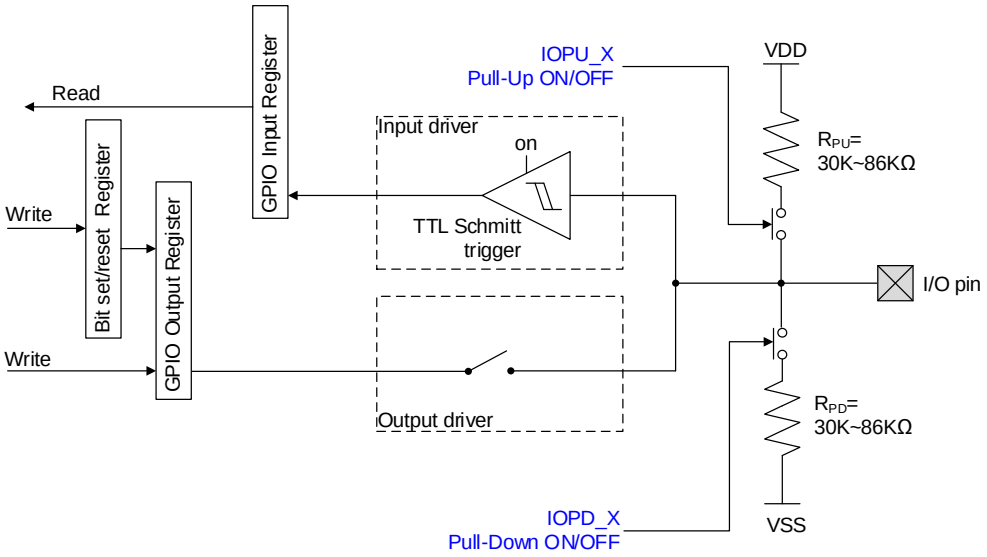


图 6-2 I/O 端口位的输入配置

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。

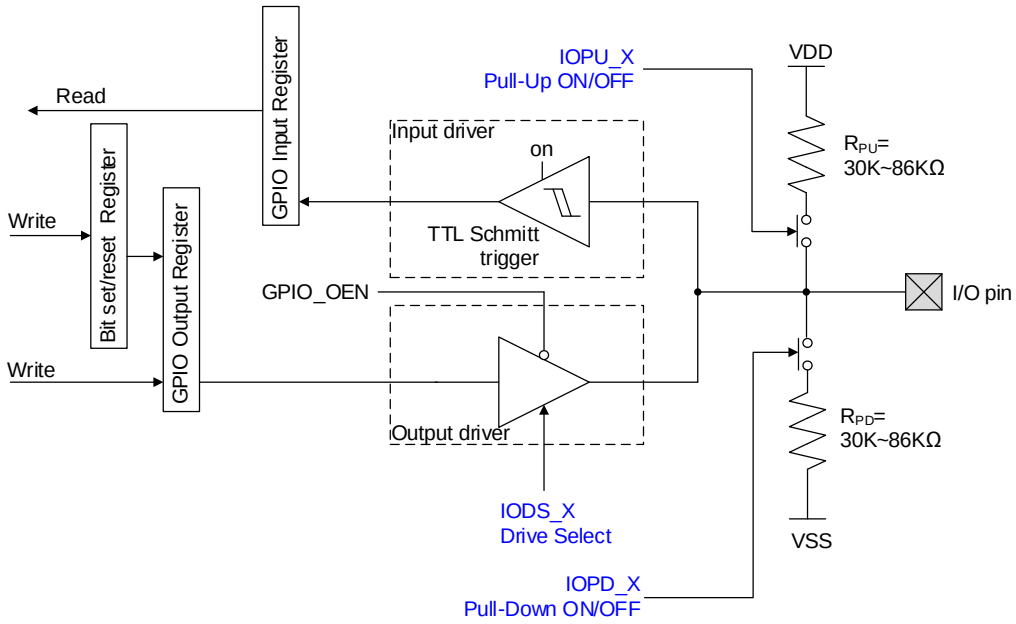


图 6-3 I/O 端口位的输出配置

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。

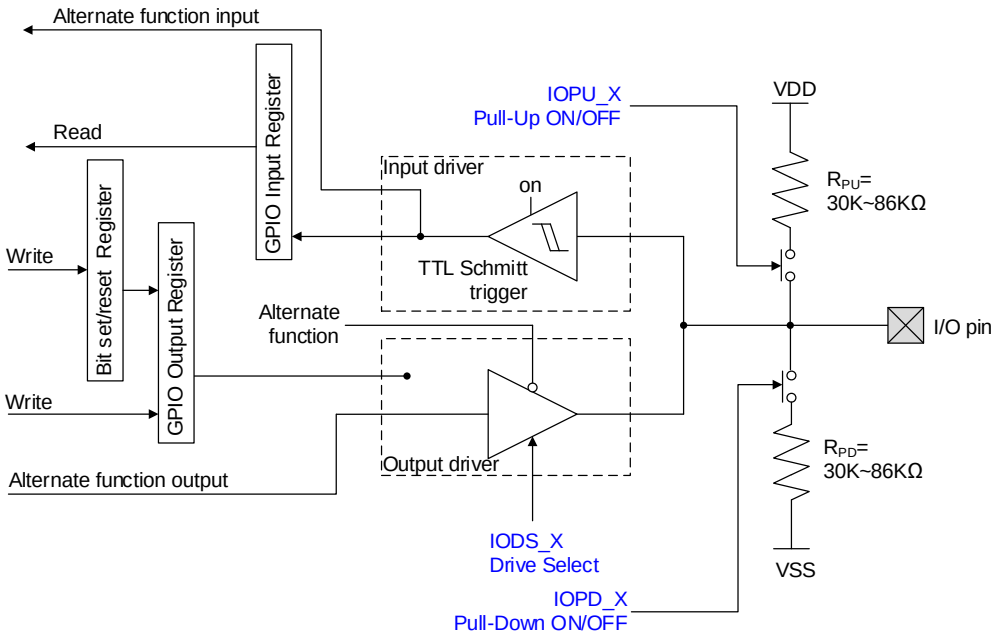


图 6-4 I/O 端口位的复用配置

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"。

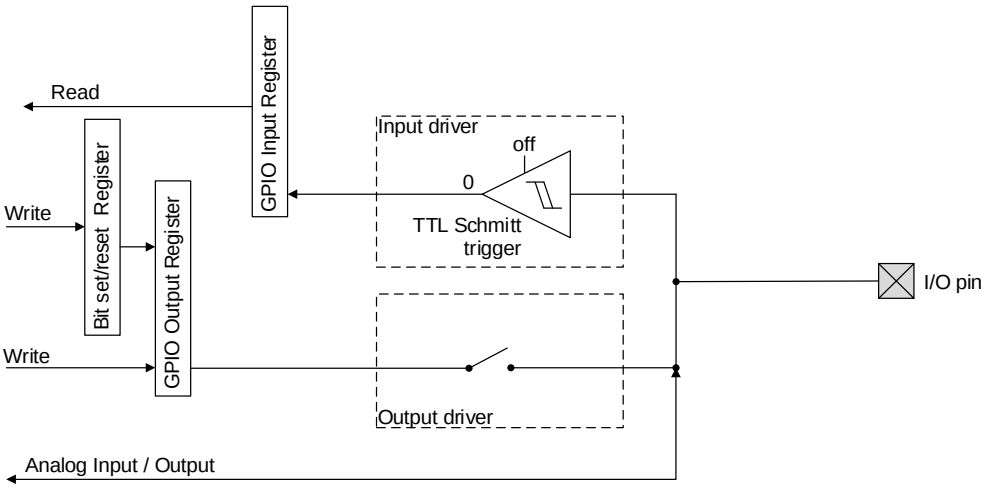


图 6-5 I/O 端口位的模拟配置

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)。

若使用者不需要使用 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。
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:

0xEBFF FFFF (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:

0x2400 0000 (GPIOA)

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:典型8 mA。(复位状态) 1:典型16 mA。
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。(20 ns)
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　连接汇总

源	目标									
	GP32C4T1	GP16C2T1	GP16C2T2	GP16C2T3	GP16C2T4	BS16T1	OPAMP	ADC	CMP	IRINF
GP32C4T1	—	1	1	1	1	—	8	—	3	7
GP16C2T1	1	—	1	1	1	—	8	—	3	7
GP16C2T2	1	1	—	1	1	—	8	—	3	7
GP16C2T3	1	1	1	—	1	—	8	—	3	—
GP16C2T4	1	1	1	1	—	—	8	—	3	—
BS16T1	—	—	—	—	—	—	—	—	—	—
UART1	—	—	—	—	—	—	—	—	—	7
UART2	—	—	—	—	—	—	—	—	—	7
UART3	—	—	—	—	—	—	—	—	—	—
UART4	—	—	—	—	—	—	—	—	—	—
ADC	—	—	—	—	—	—	—	—	—	—
T. Sensor	—	—	—	—	—	—	—	4	—	—
VBG	—	—	—	—	—	—	—	4	—	—
VREF	—	—	—	—	—	—	—	4	—	—
OPAMP	—	—	—	—	—	—	—	4	—	—
IA	—	—	—	—	—	—	—	4	—	—
MCO	2	—	—	—	—	—	—	—	—	—
RTC	2	—	—	—	—	—	—	—	—	—
CMP	5	5	5	5	5	—	—	—	—	—
SYST ERR	—	6	6	6	6	—	—	—	—	—

表 7-1　互连矩阵

注:

1. 表中的数字链接到第 7.3 节：互连描述中详细介绍的相应互连号。
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				
	GP32C4T1	GP16C2T1	GP16C2T2	GP16C2T3	GP16C2T4
IT0	—	—	—	—	—
IT1	—	—	—	—	—
IT2	—	—	—	—	—
IT3	—	—	—	—	—
IT4	—	GP32C4T1	GP32C4T1	GP32C4T1	GP32C4T1
IT5	GP16C2T1	—	GP16C2T1	GP16C2T1	GP16C2T1
IT6	GP16C2T2	GP16C2T2	—	GP16C2T2	GP16C2T2
IT7	GP16C2T3	GP16C2T3	GP16C2T3	—	GP16C2T3
IT8	GP16C2T4	GP16C2T4	GP16C2T4	GP16C2T4	—
IT9	—	—	—	—	—
IT10	—	—	—	—	—
IT11	—	—	—	—	—
IT12	—	—	—	—	—
IT13	—	—	—	—	—
IT14	—	—	—	—	—
IT15	—	—	—	—	—

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。

以下章节对同步模式进行了详细说明：

- 第 16.4.13 节：定时器同步(面向通用定时器 32 位 4 通道 GP32C4T1)
- 第 16.4.12 节：外部触发的同步(面向通用定时器 32 位 4 通道 GP32C4T1)
- 第 17.4.15 节：定时器同步(面向通用定时器 16 位 2 通道 GP16C2T1/GP16C2T2/GP16C2T3/GP16C2T4)
- 第 17.4.14 节：外部触发的同步(面向通用定时器 16 位 2 通道 GP16C2T1/GP16C2T2/GP16C2T3/GP16C2T4)

触发信号

在可配置定时器事件发生后，输出(来自主设备)出现在定时器 TRGOUT 信号上。

输入(到从设备)位于定时器 ITx 信号上。

7. 3. 2 从时钟源到GP32C4T1

可以通过微控制器输出时钟(MCO)来选择外部时钟(HOSC 和 LOSC)、内部时钟(LRC)，或选择 RTC 唤醒中断和 GPIO 作为 GP32C4T1 定时器的通道 1 输入来捕获。此外，GP32C4T1 定时器的通道 2 输入支持捕获 RTC 1Hz 信号。

定时器允许进行校准或精确测量内部时钟(如 HRC16M 或 LRC)，以及使用精确的时钟(如 LOSC 或 HOSC)作为参考时序。

7. 3. 3 从定时器到比较器

通用定时器 GP32C4T1、GP16C2T1、GP16C2T2、GP16C2T3 和 GP16C2T4 可用作 CMP 的消隐窗口输入。

第 11.4.6 节：CMP 消隐功能对消隐功能进行了说明。

以下部分给出了消隐源：

- 第 11.5.2.1 节：CMP 配置寄存器 1 位 24:20 BLANKING[4:0]

触发信号

定时器输出信号是 CMP 的消隐源输入。

7. 3. 4 从内部模拟源到ADC

内部温度传感器输出电压 V_{TS}、内部参考电压 V_{REF}、V_{BG} 与 IA 接到 ADC 输入通道。
更多信息请参见模数转换器 (ADC)

7. 3. 5 从比较器到定时器

CMP 比较器输出可连接到 GP32C4T1、GP16C2T1、GP16C2T2、GP16C2T3 或 GP16C2T4 定时器的输入。

还可将 CMP 比较器输出用作 GP16C2T1、GP16C2T2、GP16C2T3 和 GP16C2T4 定时器的 BKIN 刹车输入信号。

7. 3. 6 从系统错误到GP16C2T1、GP16C2T2、GP16C2T3 和GP16C2T4

GP16C2T1、GP16C2T2、GP16C2T3 和 GP16C2T4 定时器的 BKIN 输入收集来自以下方面的 MCU 内部故障事件：

- 由高速时钟安全系统(CSSHOSC)与低速系统时钟(CSSLOSC)生成的时钟故障事件。
- LVD 输出。
- Cortex-M0 LOCKUP(硬故障)输出

刹车功能的目的是保护由这些定时器生成的 PWM 信号所驱动的功率器件。

7. 3. 7 从GP16C2T1、GP16C2T2、GP32C4T1、UART1 和UART2 到IRINF

与 UART1 或 UART2 传输信号相关的 GP16C2T1、GP16C2T2 或 GP32C4T1 定时器的输出通道可生成红外输出波形。

第 3.3.5 节：红外线接口输出配置对此功能进行了介绍。

7. 3. 8 从GP16C2T1、GP16C2T2、GP16C2T3、GP16C2T4 和GP32C4T1 到OPAMP

比较器的输出结果还可以通过与 TIMER 输出的信号进行相乘，输出一个高频信号，可以作为 LED 驱动器。

更多信息请参见轨对轨运算放大器 (OPAMP)

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
- 2~17 HCLK 时钟取自单一周期计算

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

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

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

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
																														TRM	SIGN

—	Bit 31-2	—	—
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和 modem 流控操作(CTS/RTS)，同时还支持多机通信方式。

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
- 十六个中断源
- 内置小数波特率发生器，覆盖范围广
  - 在时钟频率为 16 MHz 下，可设置收发波特率高达 1 MBps，最低可达 245 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 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
  - 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

- 支持 ModBus 通讯
  - CR/LF 字节检测
  - 超时检测功能
- 噪声侦测

9. 3 结构图

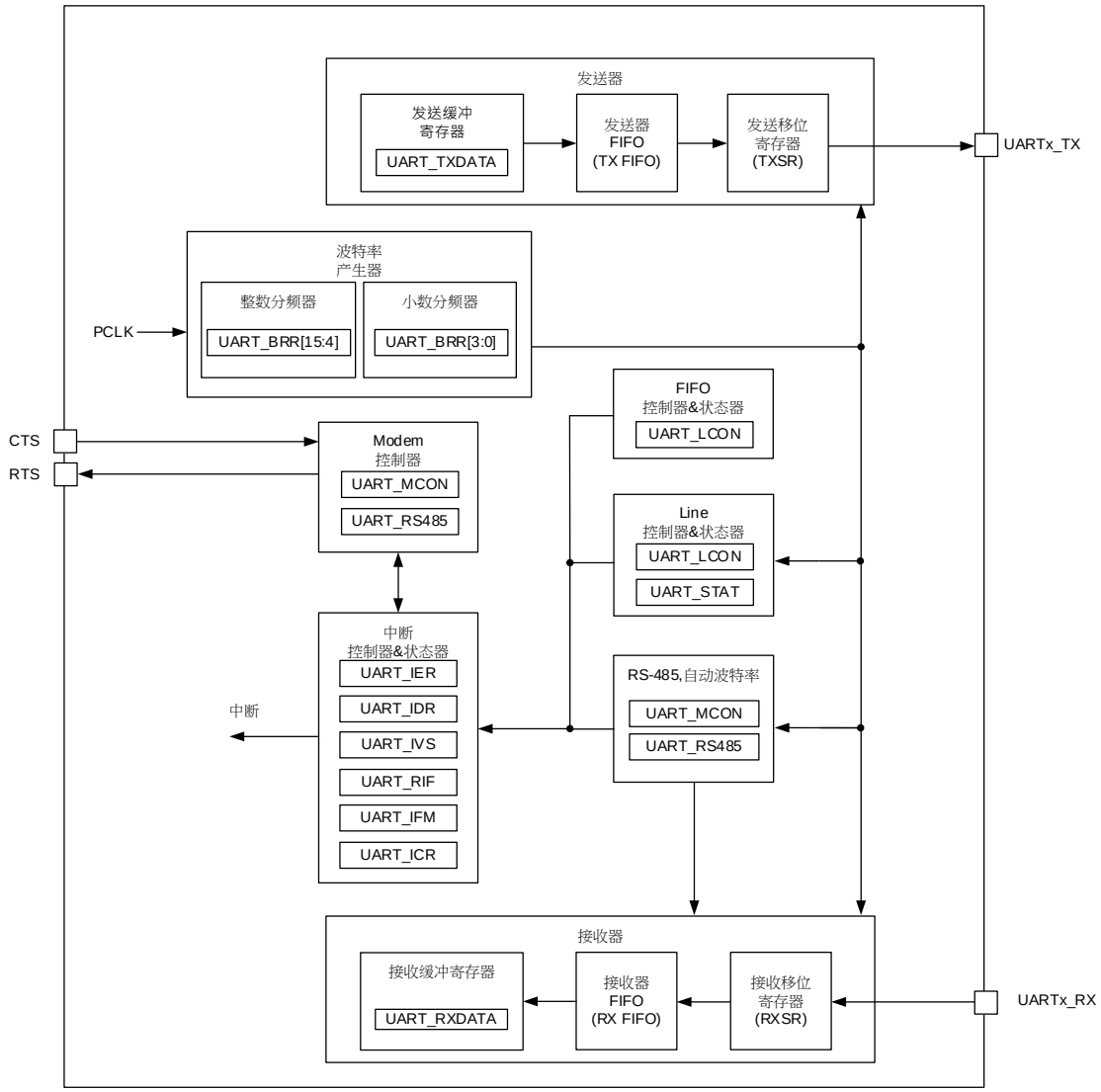


图 9-1 UART 框图

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
- 可通过设置 UART_LCON 寄存器的 MSB 位进行数据格式设置
- 1, 2 个停止位表明帧的结束
- 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。
- 一个接收时间寄存器(**UART_RTOR**)侦测输入信号时间并产生中断

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 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。

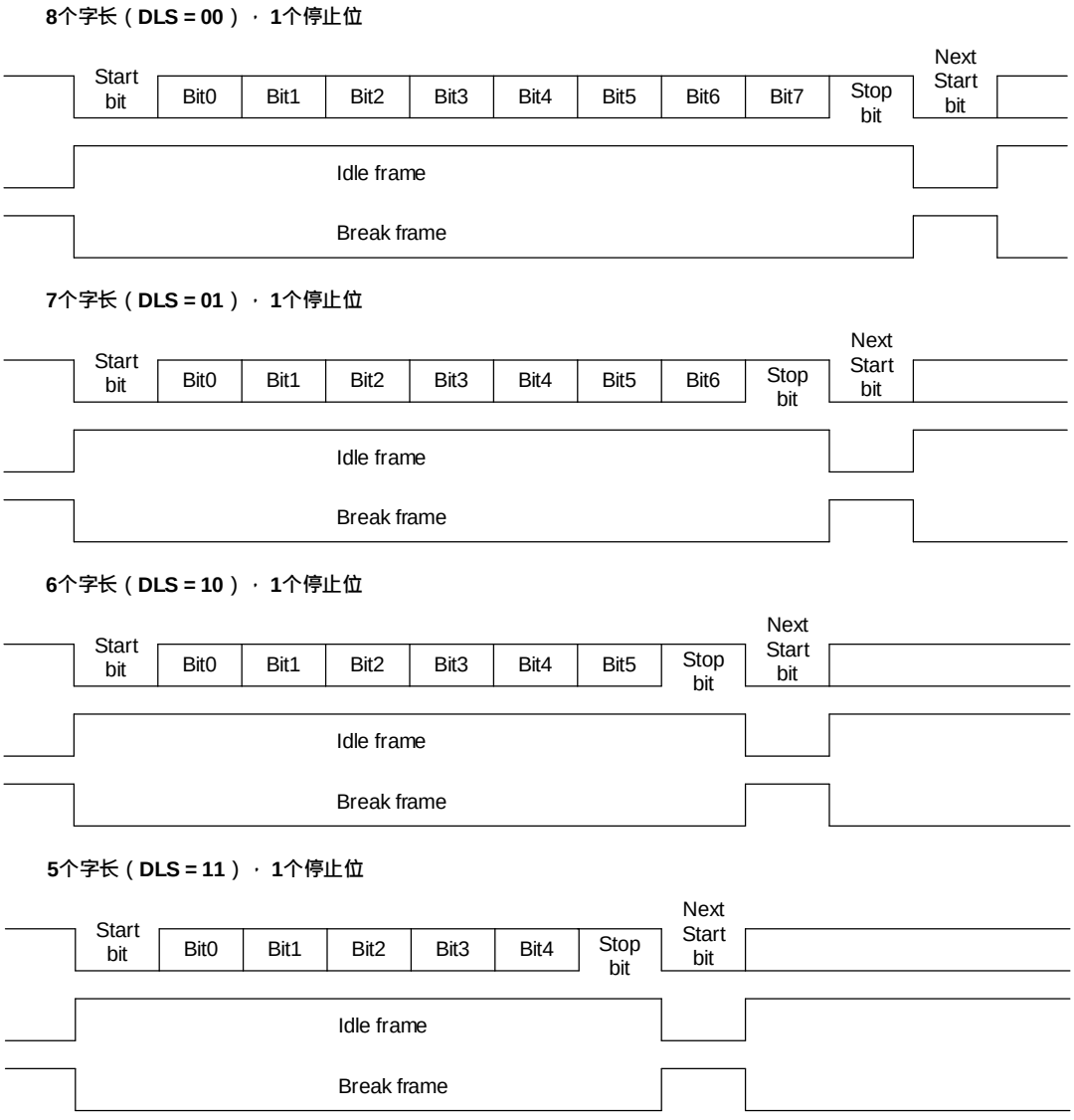


图 9-2　数据宽度设置

9. 4. 2 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: 1 和 2 个停止位。

- 1 个停止位:停止位的位数的默认值。
- 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。

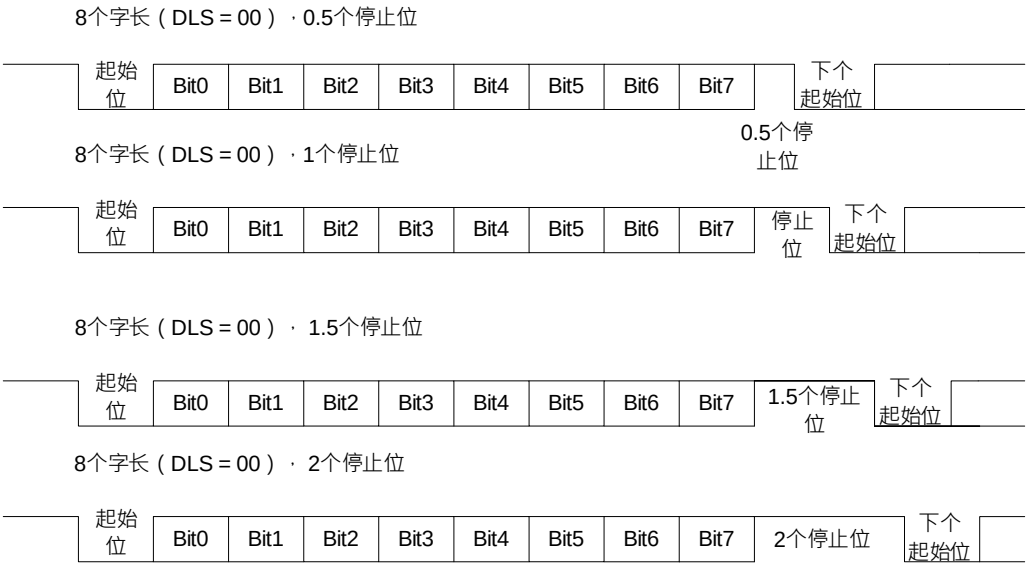


图 9-3 配置停止位

TX FIFO 门坎设置:

设置 UART_FCON 寄存器的 TXTH 位，可设置 FIFO 的门坎为 0,2,2,4，当 FIFO 内的数据个数小于门坎会使得 **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 内的个数是否小于门坎，使得 **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，使得 **UART_RIF** 寄存器的 TFTH 位清除，若使用者未写入新的数据至 FIFO 中，FIFO 内的数据个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 内的数据个数小于门坎事件与 FIFO empty 事件时，使得 TFTH 与 TFEMPTY 位为 1。UART_STAT 寄存器的 TFTH 与 TFEMPTY 位则是反应 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. 设置 **UART_LCON** 寄存器中的 TXEN 位，开启发送器。
6. 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)。
7. 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. 3 接收器

9. 4. 3. 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]×Ts，x[n+1]×Ts 和 x[n+2]×Ts。Ts 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]×Ts。

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)]×Ts。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。

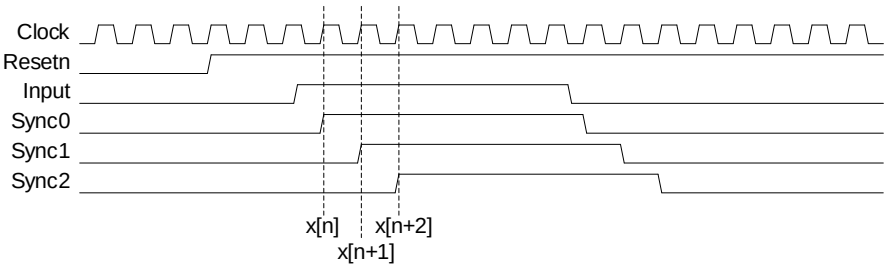


图 9-4　防抖动波形

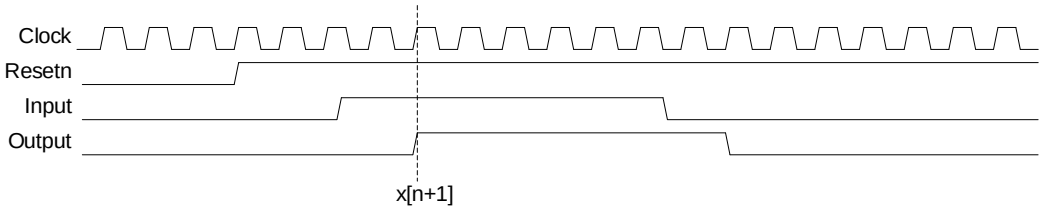


图 9-5　防抖动输出

9. 4. 3. 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。

起始位侦测在 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

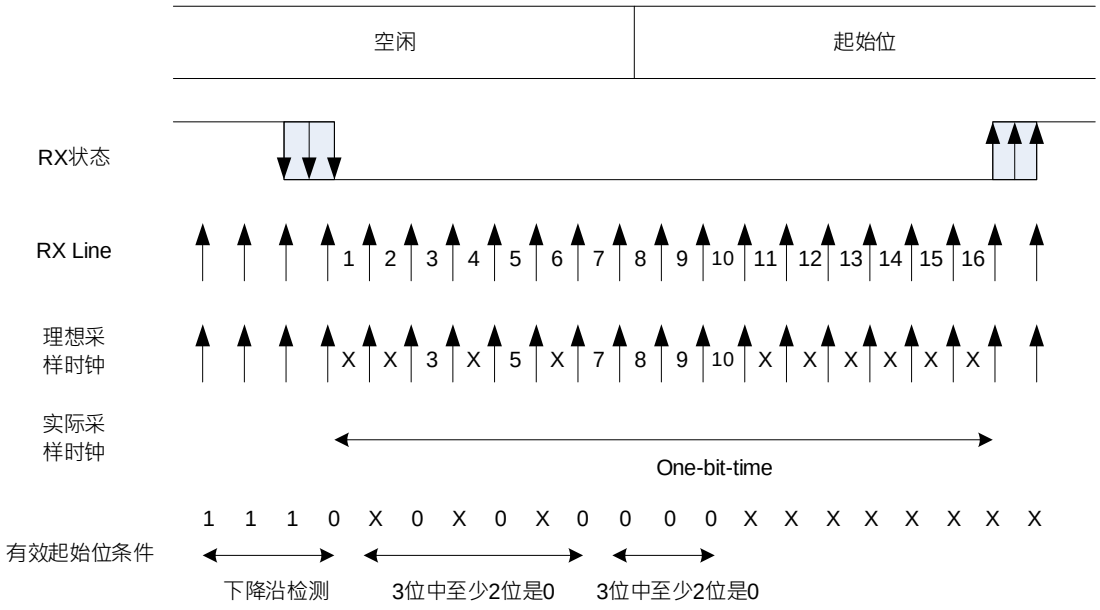


图 9-6　起始位侦测

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 1,2,4,6，当 FIFO 内的数据个数大于门坎会使得 **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 内的个数是否大于门坎，使得 **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，使得 **UART_RIF** 寄存器的 RFTH 位清除，若用户未读取数据，FIFO 内的数据个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 内的数据个数大于门坎事件时，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. 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 中。换句话说，数据已经被接收并且可以被读出(包括与之有关的错误标志位)。

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，错误标志位将被设置为 1。同时 RXBERR 位也会和 RFNEMPTY 位一起被设置为 1。

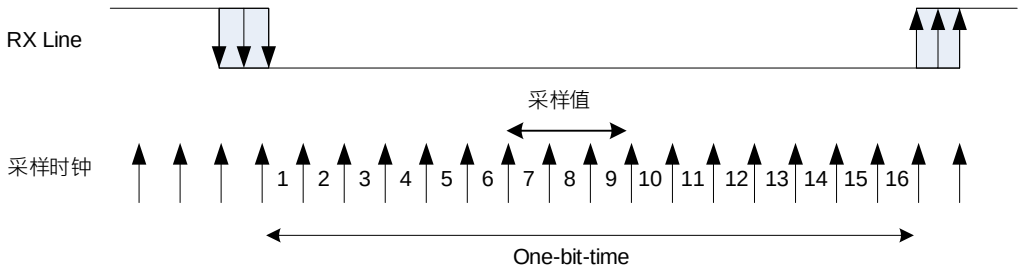


图 9-7　数值采样

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 9-1　采样数据的噪音检测数值

9. 4. 4　状态寄存器

在 UART 中配置了 15 种 UART 状态提供使用者使用，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 门坎):
  当此位为 1 表示 RX FIFO 内数据大于门坎(设置 FCR 寄存器的 RXTH，可设置门坎为 1,2,4,6)，告知使用者需要读取 **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 内已有 8 个字节，需要被读取。
- RFOERR (RX overrun):
  当此位为 1 表示 RX FIFO 内已有 8 个字节，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 门坎):
  当此位为 1 表示 TX FIFO 内数据小于门坎(设置 FCR 寄存器的 TXTH，可设置门坎为 0,2,2,4)，告知使用者需要写入 **UART_TXDATA** 寄存器。
- TFEMPTY (TX empty):
  当此位为 1 表示 TX 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 内已有 8 个字节，准备发送。
- TFOERR (TX overrun):
  当此位为 1 表示 TX 内已经满字节，且又在写入 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. 5 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 = fPCLK / 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)

预期波特率	实际波特率	BRR[15:0]	%误差
734Bps	734.01Bps	0x5526	0
1200Bps	1200.03Bps	0x3415	0
2400Bps	2399.88Bps	0x1A0B	0
4800Bps	4800.48Bps	0xD05	0.01
9600Bps	9598.08Bps	0x683	0.02
19200Bps	19207.68Bps	0x341	0.04
38400Bps	38369.3Bps	0x1A1	0.08
57600Bps	57553.96Bps	0x116	0.08
115.2KBps	115.11KBps	0x8B	0.08
230.4KBps	231.88KBps	0x45	0.64
460.8KBps	457.14KBps	0x23	0.79
921.6KBps	941.12KBps	0x11	2.12

表 9-2　时钟为 16MHz 下，设置波特率时的误差计算

9. 4. 6　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 fPCLK/65535 和 fPCLK/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。

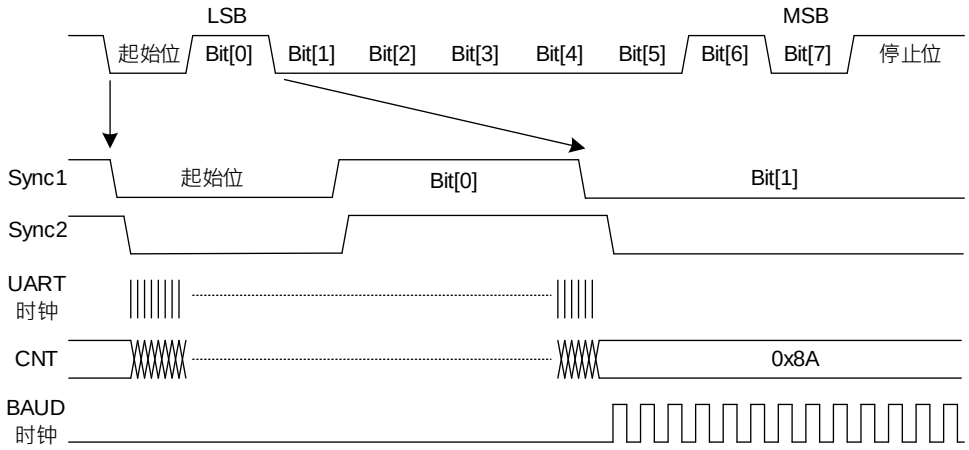


图 9-8　自动波特率侦测模式 0

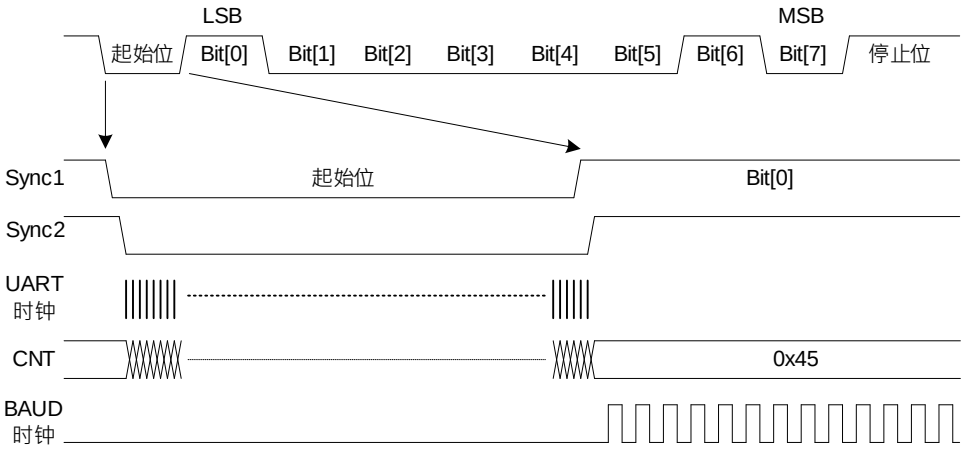


图 9-9　自动波特率侦测模式 1

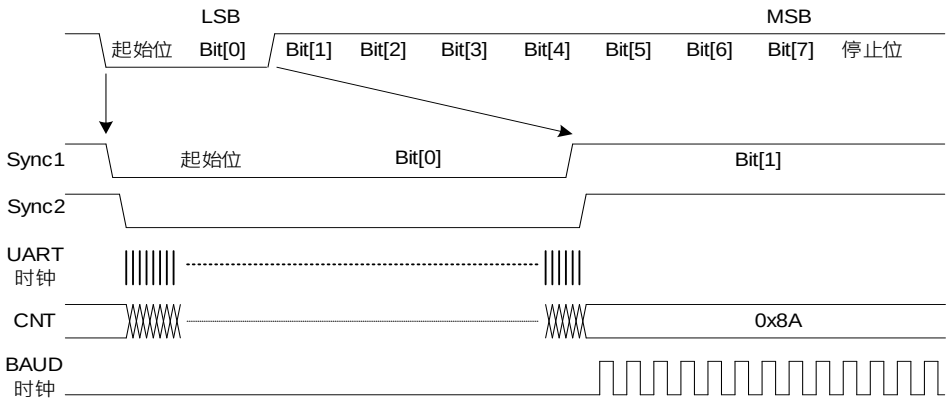


图 9-10　自动波特率侦测模式 2

9. 4. 7　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)。

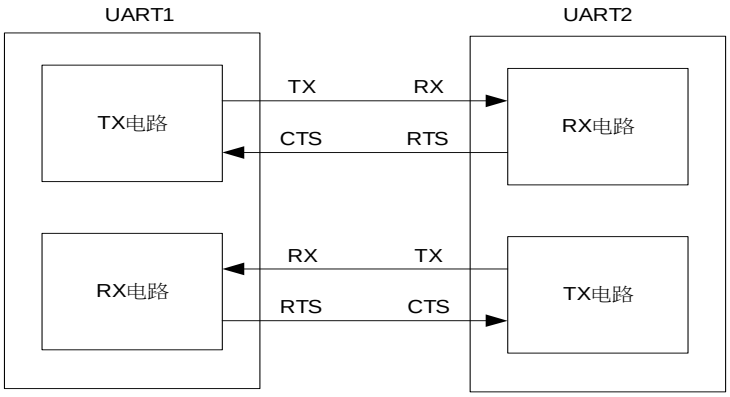


图 9-11　自动流量控制框图

9. 4. 7. 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 阈值的值是: 1，2，4，6 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)，阈值设置为 6 个字节能用一个字节的安全区最大限度的使用 FIFO。

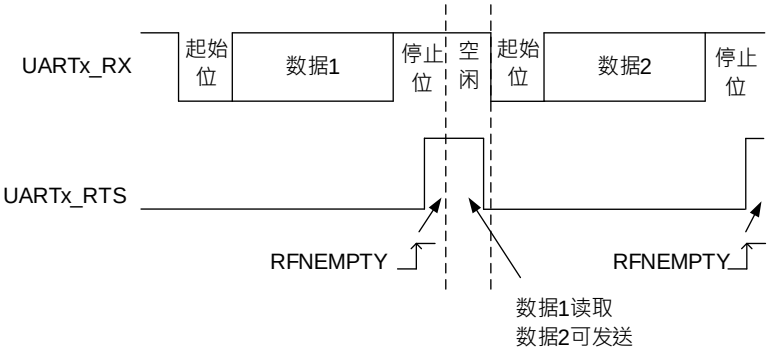


图 9-12 自动 RTSn 控制

9. 4. 7. 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。

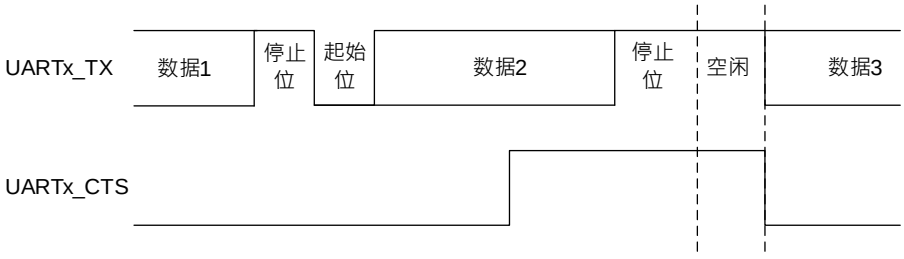


图 9-13 自动 CTSn 控制

9. 4. 7. 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。

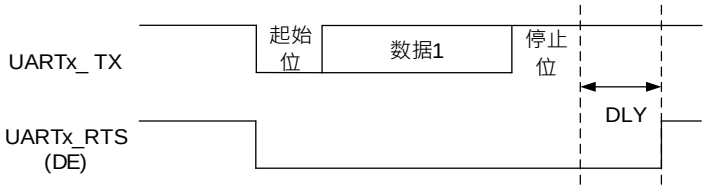


图 9-14　驱动开启当 AADINV=0

9. 4. 8　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 字节后得到通知。

9. 4. 9　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 9-3　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. 10 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。

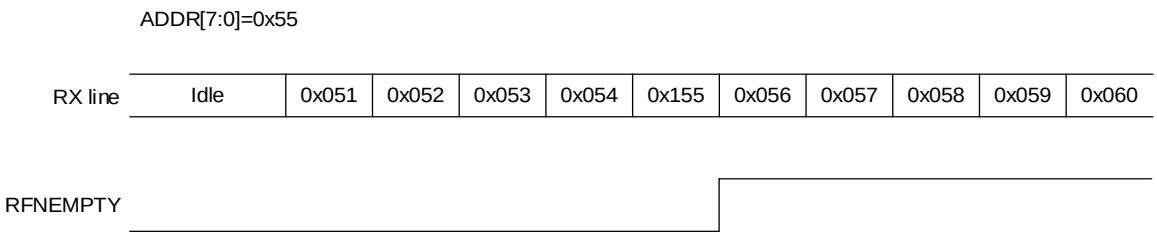


图 9-15　使用地址标示侦测模式

9. 4. 11　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。

多缓冲器通信中的错误标志位和中断产生

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位设置为 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，有分别的错误标志位中断开启位；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. 12　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，主要提供使用者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 16 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
噪声位侦测	NOISE
接收器 FIFO 触发门坎	RFTH
接收器非空	RFNEMPTY
接收器满	RFFULL
接收器溢出	RFOERR
接收器下溢	RFUERR
发送器字节完成	TBC
接收器 FIFO 触发门坎	RFTH
发送器空	TFEMPTY
发送器溢出	TFOERR

表 9-4　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_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:04_H																												
复位值:0x0000 0000																												
31	30	29	28	27	26	25	24	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，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，此功能用于和其他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，采用起始位后面跟着第0位的顺序 1:数据在发送和接收的时候，采用起始位后面跟着最高位的顺序
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)
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					ABRREPT	ABRMOD		ABREN						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引脚输出高电平 1:发送器未发送时，TX引脚开漏
—	Bits 15-12	—	—
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 7-3	—	—
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引脚输出高电平 1:自动流量控制关闭时，RTS引脚输出低电平
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: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<7:0>							

—	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
—	Bits 23-8	—	—
RTO	Bits 7-0	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。

9. 5. 2. 8 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中数据笔数** 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中数据少于或等于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
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
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. 9 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内未满足数据时清除。 0:发送器FIFO未满足数据 1:发送器FIFO满足数据
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写入数据时清除。 0:发送器有数据 1:发送器无数据
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	**接收器溢出错误** 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内未满足数据时清除。 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. 10 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	—	—	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
—	Bit 7-6	—	—
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. 11 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	—	—	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
—	Bit 7-6	—	—
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. 12 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	—	—	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
—	Bit 7-6	—	—
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. 13 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	—	—	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
—	Bit 7-6	—	—
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. 14 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	—	—	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
—	Bit 7-6	—	—
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. 15 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	—	—	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)
—	Bit 7-6	—	—
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 外部中断 (EXTI)

10. 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/内部事件信道，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缘，并且通过状态标志位来标示中断的来源。这些可配置的信道是由 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信道都可独立配置。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。

10. 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。

10. 3 结构图

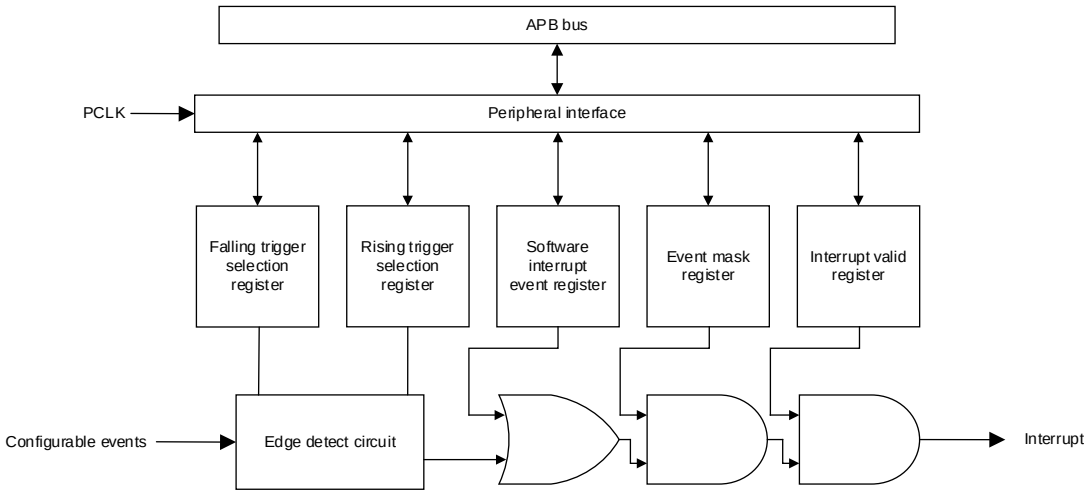


图 10-1　外部中断/事件框图

10. 4 功能描述

10. 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. 使用者可以由 **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。

10. 4. 2 软件中断选择

使用者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. 使用者可以根据需求设定 **EXTI_SWI**，让 EXTI 在某些时间点发出中断。

10. 4. 3 外部和内部中断/事件通道映射

因为 GPIO 有 A/B/C/D 组，每一组最多会有 16 bit 的信号，4 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	通道类型
0 - 15	GPIO	可配置
16	CMP	可配置
17	OPAMP	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 10-1 EXTI 通道连线

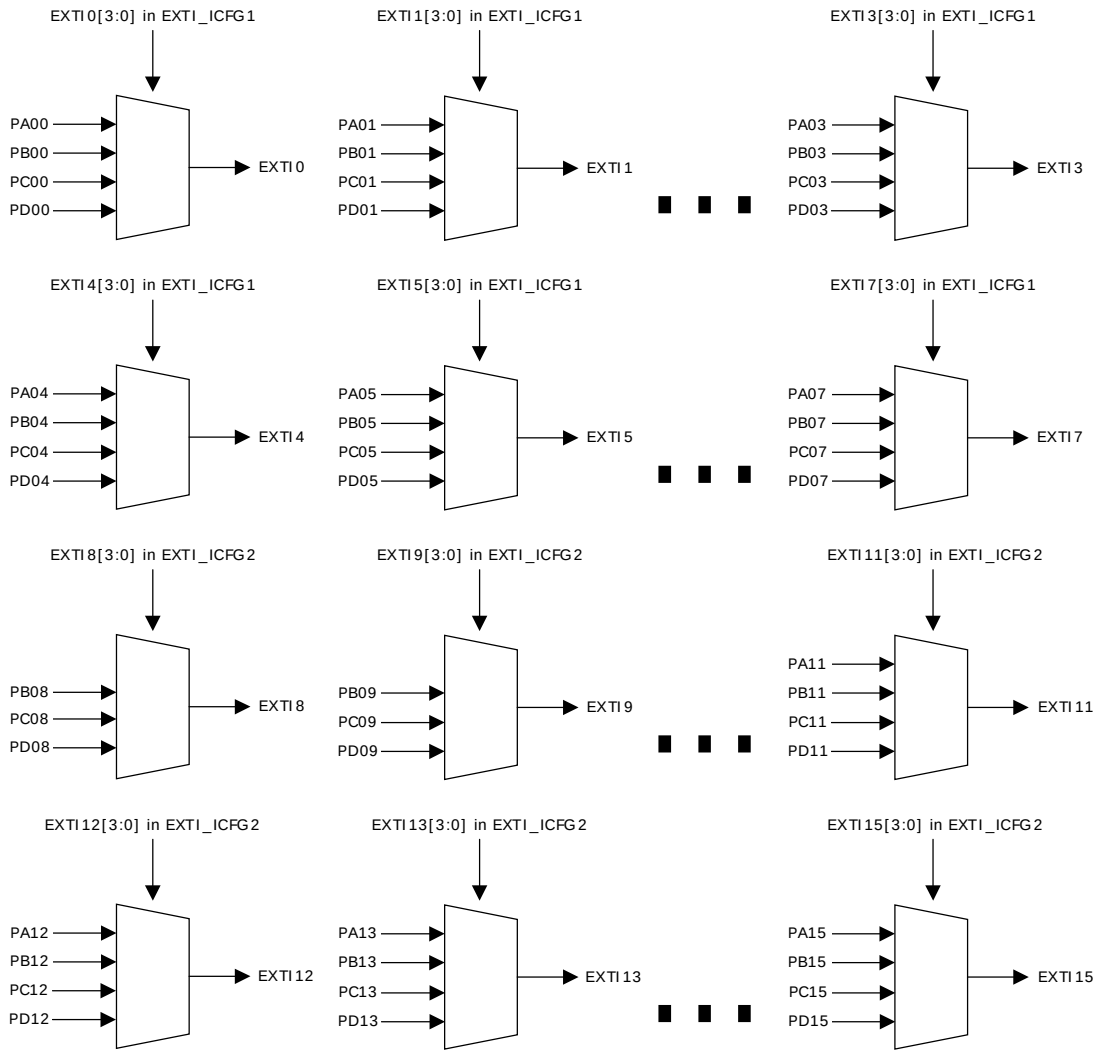


图 10-2　外部中断/事件 GPIO 映射

注:PA08、PA09 无脚位。

10. 5 特殊功能寄存器

10. 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_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

10. 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 10-1 EXTI 通道连线。

10. 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	

10. 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=0至17)。** 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	

10. 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	W1	**EXTI通道y中断使能状态。(y=20至21)。** 0: EXTI通道y中断禁止状态。 1: EXTI通道y中断使能状态。
IV21	Bit 20	W1	
—	Bits 19-18	—	—
IV17	Bit 17	W1	**EXTI通道y中断使能状态。(y=0至17)。** 0: EXTI通道y中断禁止状态。 1: EXTI 通道 y 中断使能状态。
IV16	Bit 16	W1	
IV15	Bit 15	W1	
IV14	Bit 14	W1	
IV13	Bit 13	W1	
IV12	Bit 12	W1	
IV11	Bit 11	W1	
IV10	Bit 10	W1	
IV9	Bit 9	W1	
IV8	Bit 8	W1	
IV7	Bit 7	W1	
IV6	Bit 6	W1	
IV5	Bit 5	W1	
IV4	Bit 4	W1	
IV3	Bit 3	W1	
IV2	Bit 2	W1	
IV1	Bit 1	W1	
IV0	Bit 0	W1	

10. 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	W1	**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	W1	
—	Bits 19-18	—	—
RF17	Bit 17	W1	**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	W1	
RF15	Bit 15	W1	
RF14	Bit 14	W1	
RF13	Bit 13	W1	
RF12	Bit 12	W1	
RF11	Bit 11	W1	
RF10	Bit 10	W1	
RF9	Bit 9	W1	
RF8	Bit 8	W1	
RF7	Bit 7	W1	
RF6	Bit 6	W1	
RF5	Bit 5	W1	
RF4	Bit 4	W1	
RF3	Bit 3	W1	
RF2	Bit 2	W1	
RF1	Bit 1	W1	
RF0	Bit 0	W1	

10. 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: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
										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	W1	**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使能。 1: EXTI通道y中断产生。
IM21	Bit 20	W1	
—	Bits 19-18	—	—
IM17	Bit 17	W1	**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使能。 1: EXTI 通道 y 中断产生。
IM16	Bit 16	W1	
IM15	Bit 15	W1	
IM14	Bit 14	W1	
IM13	Bit 13	W1	
IM12	Bit 12	W1	
IM11	Bit 11	W1	
IM10	Bit 10	W1	
IM9	Bit 9	W1	
IM8	Bit 8	W1	
IM7	Bit 7	W1	
IM6	Bit 6	W1	
IM5	Bit 5	W1	
IM4	Bit 4	W1	
IM3	Bit 3	W1	
IM2	Bit 2	W1	
IM1	Bit 1	W1	
IM0	Bit 0	W1	

10. 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: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
										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	

10. 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至21)** 0:禁止EXTI通道y上升沿触发中断。 1:使能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至17)** 0:禁止EXTI通道y上升沿触发中断。 1:使能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	

10. 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至21)** 0:禁止EXTI通道y下降沿触发中断。 1:使能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至17)** 0:禁止EXTI通道y下降沿触发中断。 1:使能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	

10. 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	

10. 5. 2. 10 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					DBEN15	DBEN14	DBEN13	DBEN12	DBEN11	DBEN10	DBEN9	DBEN8	DBEN7	DBEN6	DBEN5	DBEN4	DBEN3	DBEN2	DBEN1	DBEN0

—	Bits 31-22	—	—
DBEN21	Bit 21	R/W	**DBENy:弹跳消除功能开关。(y=20,21)。** 0:关闭弹跳消除功能。 1:开启弹跳消除功能。
DBEN20	Bit 20	R/W	
—	Bits 19-16	—	—
DBEN15	Bit 15	R/W	**DBENy:弹跳消除功能开关。(y=0…15)。** 0:关闭弹跳消除功能。 1:开启弹跳消除功能。
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	

10. 5. 2. 11 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	**DBPRE: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 EXTI 时钟频率取样。 1:取样频率为 EXTI 时钟频率/2。 … 255:取样频率为 EXTI 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。

10. 5. 2. 12 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为EXTI的外部中断源。 0000: PA[y] 引脚。 0001: PB[y] 引脚。 0010: PC[y] 引脚。 0011: PD[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	

10. 5. 2. 13 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。 0011: PD[y] 引脚。 其他配置保留。 注:PA08、PA09无引脚。
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11章 模拟比较器 (CMP)

11. 1 概述

模拟比较器是一个外围设备，它可以比较两个模拟电压的值，并以逻辑输出的形式显示比较结果。具有中断功能，当产生比较结果时，可产生中断信号，并且可以通过中断触发 ADC，或者通知应用程序开始捕获样本序列。比较器可以有不同的组态配置来对应不同的应用。

11. 2 特性

- 轨对轨输入范围
- 可配置尖峰脉冲过滤器。
- 内建 16 节点的 4 位数字电阻器。
- 可做为低电压侦测器与电容量测。
- 消隐源比较器输出。
- 比较器拥有可配置正端与负端输入。
- 提供触发给其他功能模块使用（ADC、Timer）。
- 可以产生中断信号 (通过 EXTI 控制器)。

11. 3 结构图

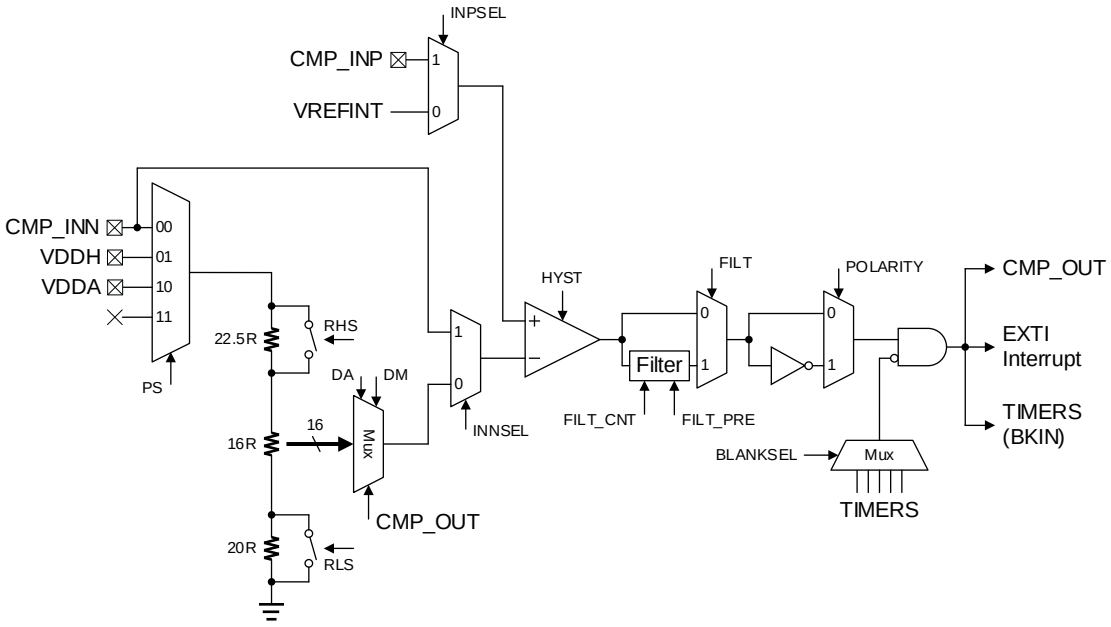


图 11-1 CMP 架构图

11. 4 功能描述

11. 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“多路复用引脚功能选择”表。

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使用。

11. 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。

11. 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(即只能读取)。

CMP 配置完成后，使用 **CMP_CFG1** 的位 31，可以将 LOCK 位设置为 1。这将导致整个 **CMP_CFG1/2** 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。

11. 4. 4 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。

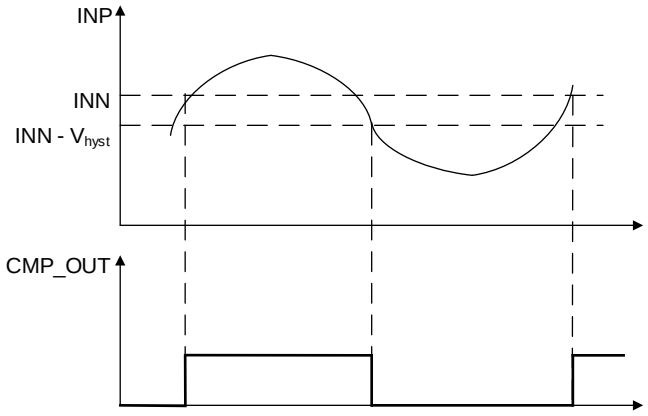


图 11-2 迟滞功能示意图

11. 4. 5 CMP滤波功能

CMP 具有可配置的低通滤波功能，避免转换结果不稳定的现象。如果不需要滤波功能，可以将其禁用。若要使用滤波功能，需搭配 CMP_CFG2.FILT_PRE[7:0]与 CMP_CFG2.FILT_CNT[2:0]配置滤波时间。

11. 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_CFG1.BLANKSEL[4:0]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:

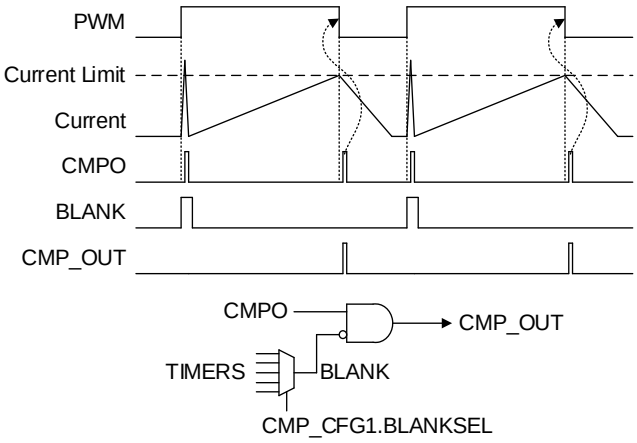


图 11-3 消隐功能示意图

11. 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(EXTI)，并且可以产生中断或事件。

11. 4. 8 CMP多节点电阻器功能

比较器内建一个多节点电阻器，电阻分为三部份：22.5R、16R 与 20R。在 16R 电阻处接入一个 16 段的电阻节点选择器，将 16R 的电阻等分为 16 个节点，可以通过寄存器 CMP_CFG2.DA[3:0]与 CMP_CFG2.DM[3:0]的设置，选择不同的电阻节点，输出不同的电压值到比较器的负端输入通道。而控制位 CMP_CFG2.RHS、CMP_CFG2.RLS 被设置时，可使得 22.5R 与 20R 电阻被短路，此开关可调节电阻节点电压值。多节点电阻器的电压源为 VDDA、VDDH 与外部输入，通过寄存器 CMP_CFG2.PS[1:0]设置选择不同的电压源，增加节点电压的输出范围。

屏蔽节点选择器 CMP_CFG2.DM[3:0]是与节点选择器 CMP_CFG2.DA[3:0]是联动的，屏蔽节点选择器 CMP_CFG2.DM[3:0]的每一位对应控制着节点选择器 CMP_CFG2.DA[3:0]的每一位的屏蔽功能的开启与关闭。当屏蔽节点选择器 CMP_CFG2.DM[3:0]的对应位被设置，则节点选择器 CMP_CFG2.DA[3:0]的对应位就会开启屏蔽功能，且该位的状态值与比较器的输出状态值是一致，即 CMP_CFG2.DA[X]=CMP_OUT。这样就会出现节点选择器的电压输出会在两个节点之间来回切换。

DM[3:0]	CMP_OUT	DA[3:0]	DM[3:0]	CMP_OUT	DA[3:0]
	输出状态	输出切换结果		输出状态	输出切换结果
0000	0	uuuu	1000	0	0uuu
	1	uuuu		1	1uuu
0001	0	uuu0	1001	0	0uu0
	1	uuu1		1	1uu1
0010	0	uu0u	1010	0	0u0u
	1	uu1u		1	1u1u
0011	0	uu00	1011	0	0u00
	1	uu11		1	1u11
0100	0	u0uu	1100	0	00uu
	1	u1uu		1	11uu
0101	0	u0u0	1101	0	00u0
	1	u1u1		1	11u1
0110	0	u00u	1110	0	000u
	1	u11u		1	111u
0111	0	u000	1111	0	0000
	1	u111		1	1111

注:"u" 表示不改变。

11. 5 特殊功能寄存器

11. 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

11. 5. 2 寄存器描述

11. 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 <4:0>					—	FILT	—	HYST	POL	OUTSEL<2:0>			—	—	—	INPSEL	—	—	—	INNSEL	—	—	—	EN

LOCK	Bit 31	R/W	**Comparator 锁定** 0: CMP_CFG1/2[31:0]位可正常读写操作。 1: CMP_CFG1/2[31:0]位仅供读操作。
VALUE	Bit 30	R	**Comparator 输出** 0: CMP输出为低电平。 1: CMP输出为高电平。
—	Bit 29-25	—	—
BLANKSEL	Bit 24-20	R/W	**Comparator 消隐功能** 00000:关闭消隐功能。 xxxx1:开启GP32C4T1_CH4消隐功能。 xxx1x:开启GP16C2T1_CH2消隐功能。 xx1xx:开启GP16C2T2_CH2消隐功能。 x1xxx:开启GP16C2T3_CH2消隐功能。 1xxxx:开启GP16C2T4_CH2消隐功能。 可同时开启多个输入消隐。
—	Bit 19	—	—
FILT	Bit 18	R/W	**Comparator 滤波功能** 若要使用滤波功能，需搭配 CFG2.FILT_PRE[7:0]与

			CFG2.FILT_CNT[2:0]配置。 0:关闭滤波功能。 1:开启滤波功能。
—	Bit 17	—	—
HYST	Bit 16	R/W	**Comparator 迟滞功能** 0:关闭迟滞功能。 1:开启迟滞功能。
POL	Bit 15	R/W	**Comparator 输出极性** 0:关闭输出反相功能。 1:开启输出反相功能。
OUTSEL	Bit 14-12	R/W	**Comparator 输出触发选择** 000:未指定输出。 001:保留。 010: GP16C2T1_BKIN。 011: GP16C2T2_BKIN。 100: GP16C2T3_BKIN。 101: GP16C2T4_BKIN。 110:保留。 111:保留。
—	Bit 11-9	—	—
INPSEL	Bit 8	R/W	**Comparator 正端输入选择** 0:VREFINT电压输入。 1:外部输入。
—	Bit 7-5	—	—
INNSEL	Bit 4	R/W	**Comparator 负端输入选择** 0:多节点电阻分压输入。 1:外部输入。
—	Bit 3-1	—	—
EN	Bit 0	R/W	**Comparator 开关** 0: Comparator 关闭。 1: Comparator 启动。

11. 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 0801																												
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
—	—	—	—	—	FILT_CNT<2:0>			FILT_PRE<7:0>								DM<3:0>				DA<3:0>				—	—	RLS	RHS	—	—	PS<1:0>	

—	Bit 31-27	—	—
FILT_CNT	Bit 26-24	R/W	**Comparator 滤波计数器** 配置滤波功能的计数次数。 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。
FILT_PRE	Bit 23-16	R/W	**Comparator 滤波预分频器** 配置滤波功能的取样时间。 0:无间隔，根据CMP时钟频率取样。 1:取样频率为CMP时钟频率/2。 … 255:取样频率为CMP时钟频率/256。
DM	Bit 15-12	R/W	**Comparator 多节点电阻分压输入掩码选择** 0000:关闭屏蔽功能。 xxx1:配置后会将CMP_OUT取代DA[0]。 xx1x:配置后会将CMP_OUT取代DA[1]。

			x1xx:配置后会将CMP_OUT取代DA[2]。 1xxx:配置后会将CMP_OUT取代DA[3]。 可同时开启多个屏蔽位。
DA	Bit 11-8	R/W	**Comparator 多节点电阻分压输入选择** 0000: 0。 0001: 1/16。 0010: 2/16。 … 1111: 15/16。
—	Bit 7-6	—	—
RLS	Bit 5	R/W	**Comparator 多节点电阻分压低节短路开关** 0:关闭低节电阻短路功能。 1:开启低节电阻短路功能。
RHS	Bit 4	R/W	**Comparator 多节点电阻分压高节短路开关** 0:关闭高节电阻短路功能。 1:开启高节电阻短路功能。
—	Bit 3-2	—	—
PS	Bit 1-0	R/W	**Comparator 多节点电阻分压电源选择** 00:选择外部输入电压。 01:选择VDDH电压。 10:选择VDDA电压。 11:保留。

第12章 液晶驱动控制器 (LCD)

12. 1 概述

LCD 控制器依据用户设置的帧幅频率与种类控制输出波型，此外 LCD 控制器配置了三种闪烁功能提供用户选择，并可以设置闪烁周期。

12. 2 特性

- 内置倍压电路(Regulated charge pump)
- 支持 16 阶可调整 VLCD 电压范围 2.5-4.0V
- 支持 4x38 或 6x36 的 COM/SEG 组合
- 支持 A、B 两种驱动波型
- 支持四阶可调式驱动偏压
  - 静态偏压(Static)
  - 1/2 偏压
  - 1/3 偏压
  - 1/4 偏压
- 支持两种 LCD 输出波型
  - 1/4 占空比
  - 1/6 占空比
- 可选择输入 LCD 频率与可编译的输出频率
- 三种闪烁功能
- LCD 控制信号数字输出

12. 3 结构图

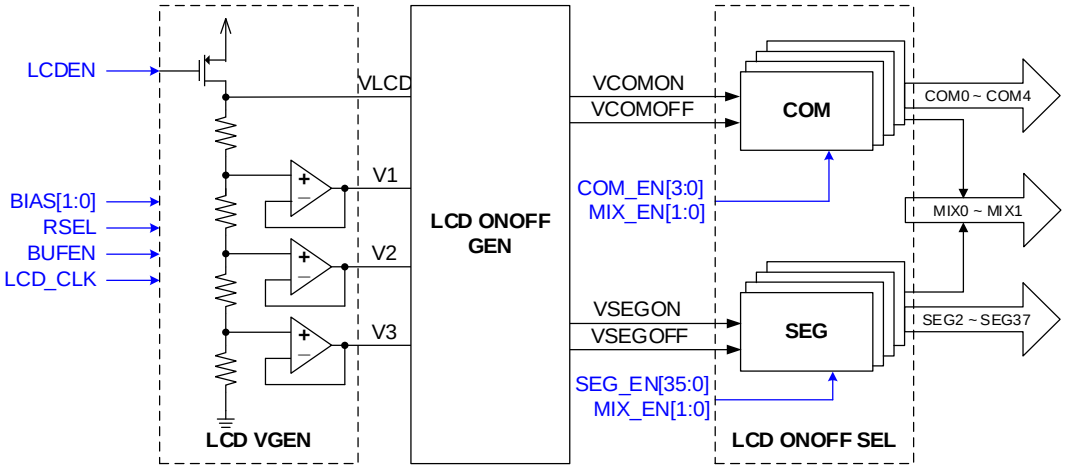


图 12-1　LCD 架构图

12. 4 功能描述

12. 4. 1　时钟配置

LCD 相关的时钟，包括 LCD 时钟(LCDCLK)以及电荷泵时钟(VLCDCLK)。

12. 4. 1. 1　　电荷泵时钟 (VLCDCLK)

VLCDCLK 需要 2MHz。提供三种时钟源，使用者可以通过 **RCU_CFG2**.VLCD_CKCFG 选择时钟源为 HRC、HOSC 或是 PLL，并且通过 **RCU_CFG2**.VLCD_CKDIV 进行分频。

12. 4. 1. 2　　LCD时钟(LCDCLK)

LCD 时钟，控制器配置两种时钟源，使用者可以通过 **RCU_CFG2**.LCD_CKCFG 选择时钟源为 LRC 或 LOSC，并且可通过 **RCU_CFG2**.LCD_CKDIV 设置分频输出 **LCD_CLK**，最低分频为 2，最高分频为 8。

寄存器 **LCD_CTRL**.FRDIV 设有帧幅分频器，最高可达 254 分频。

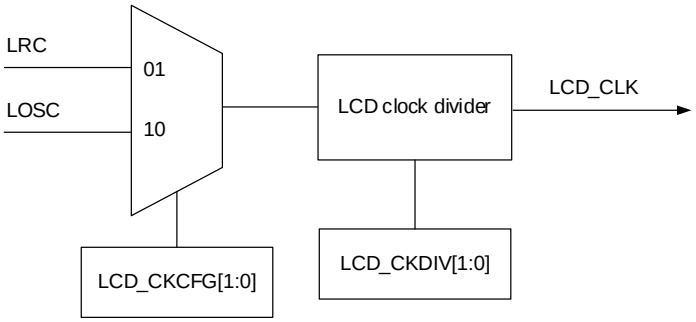


图 12-2　时钟源选择模块

时钟源	LCD_CKCFG[1:0]	LCD_CKDIV[1:0]	来源为 32k Hz，LCD_CLK(Hz)
LRC	0x01	0x00 (2 分频)	16k
		0x01 (4 分频)	8k
LOSC	0x10	0x10 (6 分频)	5.33k
		0x11 (8 分频)	4k

表 12-1　时钟源关系

12. 4. 2　帧幅频率

LCD 的工作频率由 **LCD_CLK** 提供，并设置 **LCD_CTRL**.FRDIV 位来选择帧幅分频器提供适当的工作频率输出 LCD 帧幅频率，而帧幅频率与操作波型需要依据外设的 LCD 显示器的规格设置正确的频率。

$$LCD\_PHY\_CLK=LCD\_CLK\div(2*(FRDIV+1)),\quad FRDIV = 0\ to\ 127$$

12. 4. 3　闪烁功能

LCD 控制器三种闪烁模式选择

- 设置 **LCD_CTRL**.WSBLINK 为 01，字节快速转为显示设置字节再转为未设置字节。
- 设置 **LCD_CTRL**.WSBLINK 为 10，字节快速转为全灭再转为显示设置字节。
- 设置 **LCD_CTRL**.WSBLINK 为 11，选择设置 **LCD_BLINK1** 或是 **LCD_BLINK2**，其中每 8 个位分别控制 COM 的 SEG0-7 输出，当设置为 1 时反转字节。

12. 4. 4　信号偏压、占空比以及型态

12. 4. 4. 1　　占空比

LCD 是使用动态扫描的驱动方式，所以每个 COM 的有效时间与扫描周期的比值也就是所谓的占空比，所以占空比的参数会与 COM 数量相同。

通过设定寄存器 **LCD_MODE**.DUTY，选择 1/4 以及 1/6 的占空比。

12. 4. 4. 2 偏压

LCD 的 SEG/COM 的驱动波形为類比信号，而各文件類比电压相对于 LCD 输出的最高电压的比例称为偏压。

在帧周期(frame cycle)中，对应到的相位拥有最大的振幅 VLCD 或是 VSS，其他没有对应的相位，其信号振幅:

- 选择 1/4 bias，信号振幅会呈现 1/4 或 3/4 VLCD。
- 选择 1/3 bias，信号振幅会呈现 1/3 或 2/3 VLCD。
- 选择 1/2 bias，信号振幅会呈现 1/2 VLCD。

12. 4. 4. 3 AB 型态

提供两种型态供用户选择，可通过寄存器 **LCD_MODE**.TYPE，选择 A 以及 B 型态。

12. 4. 4. 4 GPIO 模式

设定 **LCD_MODE**.LCD_GPIO，让 COMx/MIXx/SEGx 输出信号由对应的脚位改变成为数字输出。并且提供**LCD_MODE**.POLA_SEL，可以反转输出的信号，POLA_SEL[0]以及 POLA_SEL[1]分别控制 COMx、SEGx 数据的极性选择。

12. 4. 5 COM/SEG 信号介绍

输出信号的脚位

- COM[3:0]
- SEG[35:0]
- MIX[1:0]

当使用 IO 进行 LCD 信号输出时，需对将相应的脚位的复用功能设置为 COMx/MIXx/SEGx，以开启信号输出。可以使用 GPIOx_AFL 或者 GPIOx_AFH 来进行相应的复用功能设置。(详细的复用功能选择请参考 datasheet)

SEG[35:0] 所呈现的数值会是寄存器设定 LCD_RAMx 的 SEG37-SEG2。MIX[1:0]会依照占空比 1/4、1/6 配置，来决定该脚位是当做 COM 还是 SEG 功能。使用者设定 **LCD_MODE**.DUTY，便会决定 MIX[1:0]的功能。

- 使用占空比 1/4，MIX[1:0]其功能将会变成 SEG 的功能，其数值为 MIX1/MIX0 对应到 SEG1/SEG0。在此设定下，LCD 将可以提供 4 x 38。
- 使用占空比 1/6，MIX[1:0]其功能将会变成 COM 的功能，其数值为 MIX1/MIX0 对应到 COM5/COM4。在此设定下，LCD 将可以提供 6 x 36。

12. 4. 5. 1 LCD 输出波形

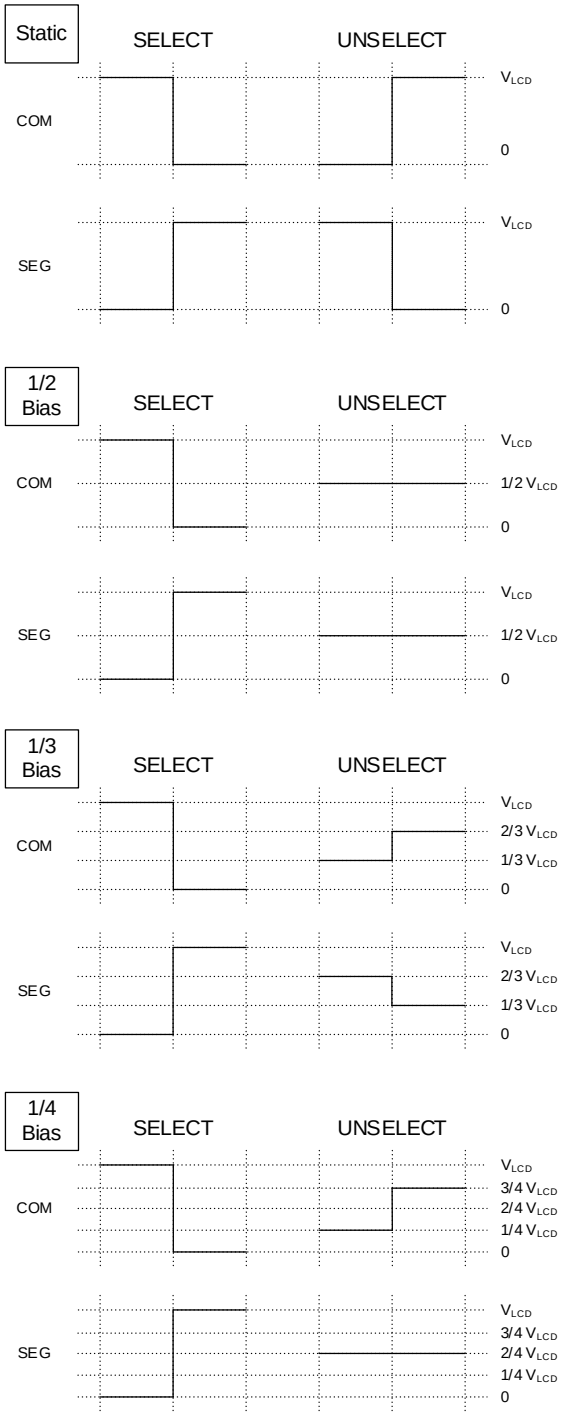


图 12-3 LCD driver 选择波形

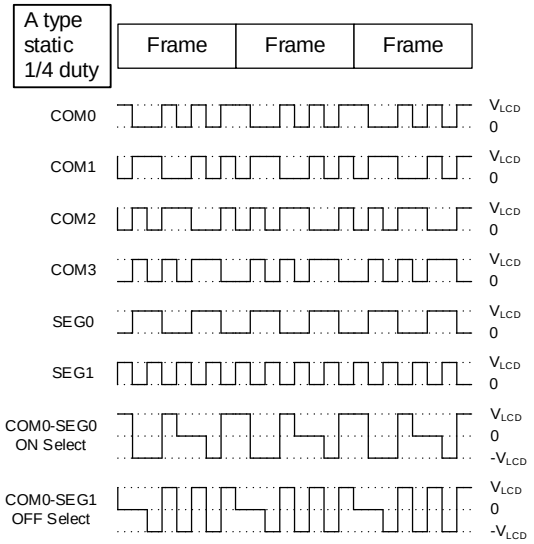


图 12-4　输出波型-静态操作(Static)、1/4 duty、A type

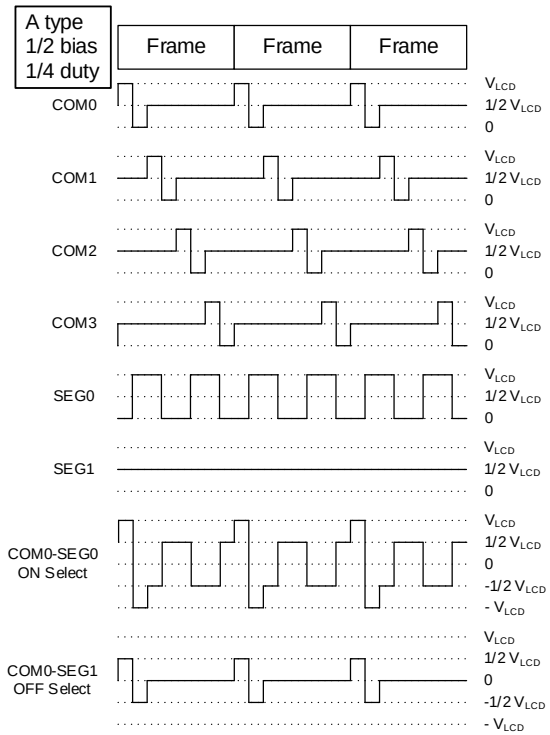


图 12-5　输出波型-1/2 bias、1/4 duty、A type

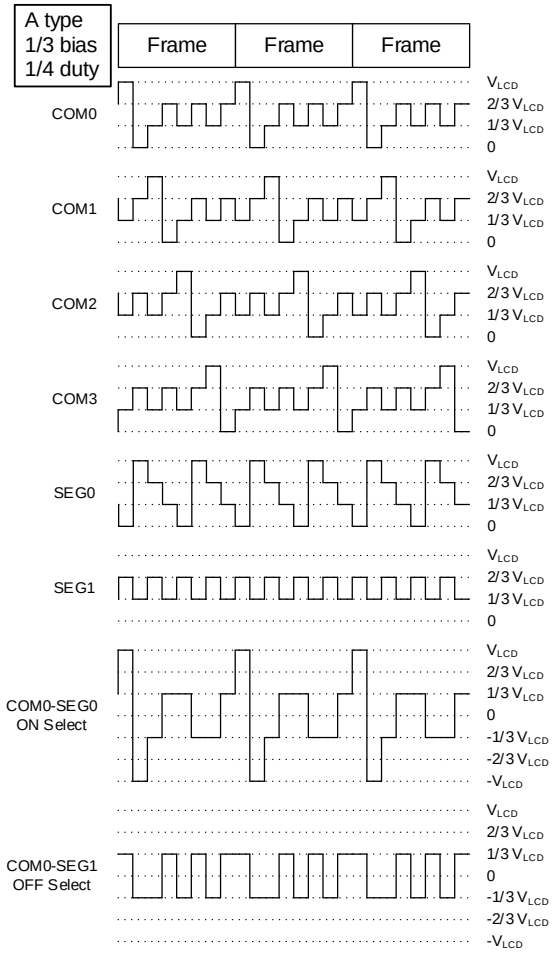


图 12-6　输出波型-1/3 bias、1/4 duty、A type

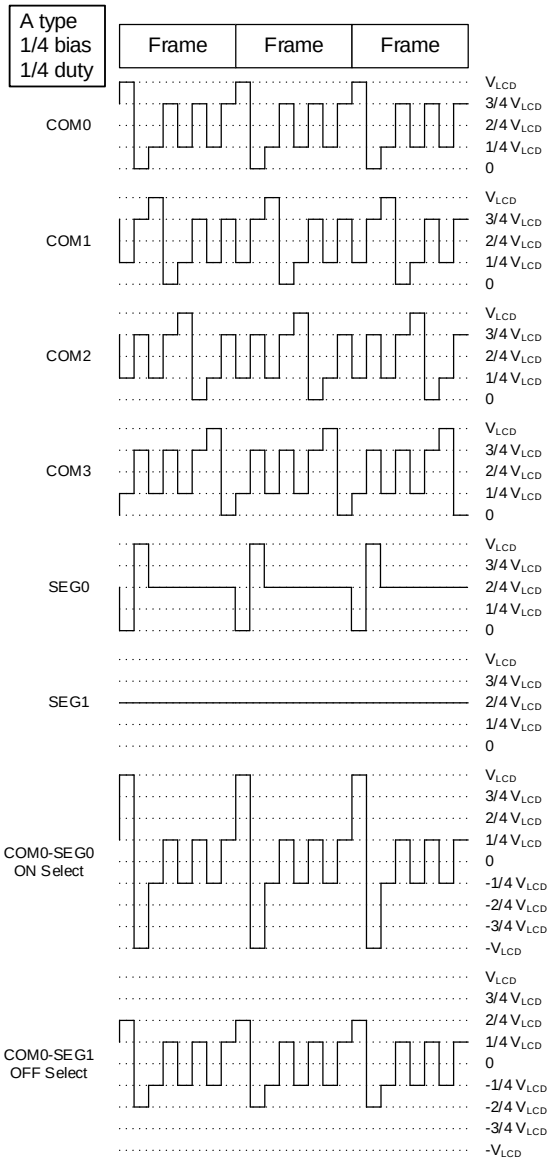


图 12-7　输出波型-1/4 bias、1/4 duty、A type

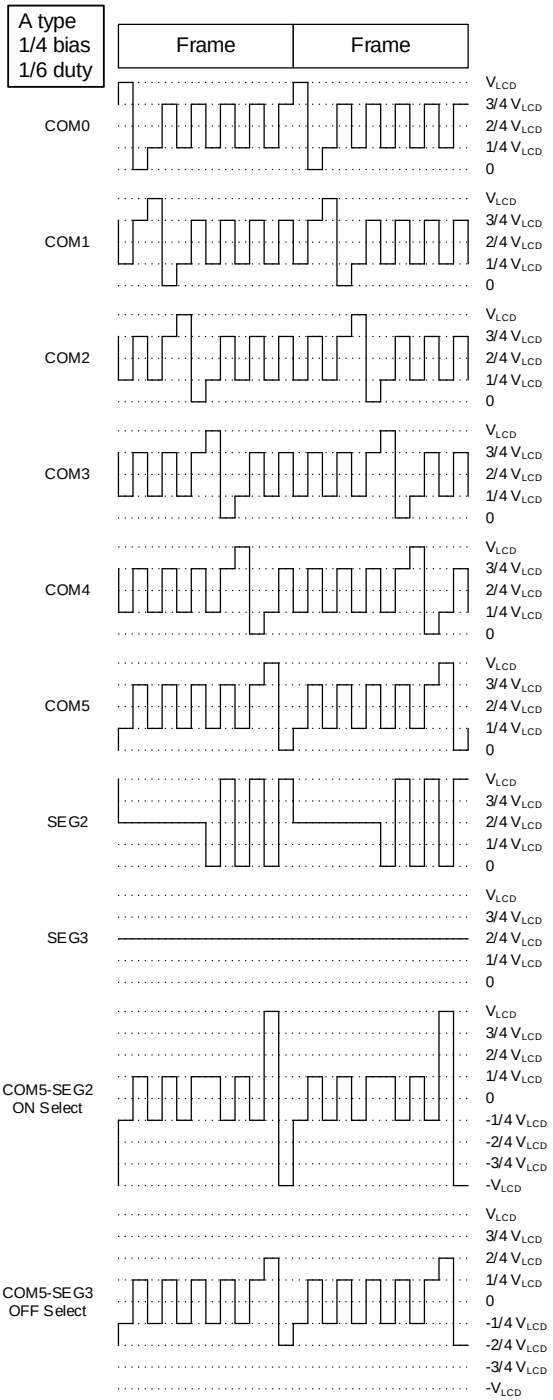


图 12-8　输出波型-1/4 bias、1/6 duty、A type

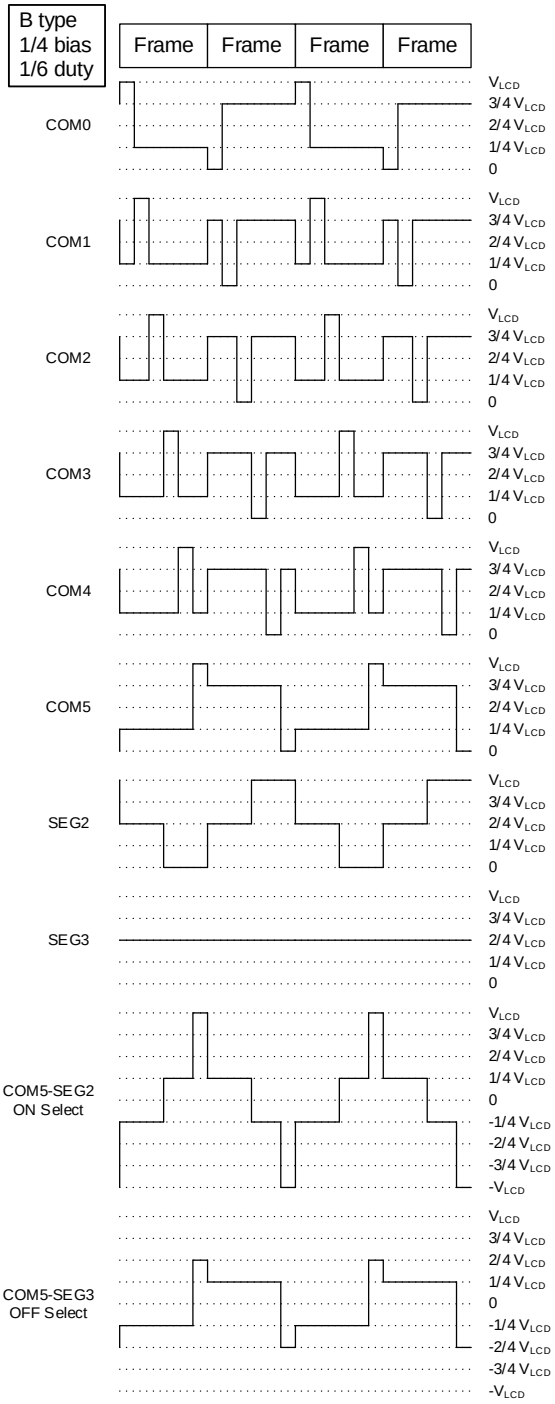


图 12-9　输出波型-1/4 bias、1/6 duty、B type

12. 4. 6 LCD 电压

LCD 电压来源可分为内部可调电荷泵以及由脚位 VLCD 提供外部电压。使用寄存器 **LCD_MODE**.VLCD_SRC 选择电压的来源。

- 选择内部电压，可以通过寄存器 **LCD_MODE**.VLCD_SEL 选择电压范围 2.5V-4.0V。
- 选择外部电压，提供的电压值需要在允许的电压范围内(参考 datasheet)。

使用 LCD 之前，皆要确保先行设定电压的来源以及电压的数值，而电压的相关配置，必须按照下列的顺序。

- 使用内部的可调电荷泵，需要提供电容在 VLCD 脚位。
  - 设定 VLCDCLK 为 2MHz。
  - 设定 **LCD_MODE**.VLCD_SRC 选择内部可调电荷泵。
  - 设定脚位 PB11 复用功能为 VLCD。(详细的复用功能选择请参考 datasheet)
  - 等待连接在 VLCD 脚位的电容充电的时间，如果外部的电容为 1uF，约需要等待 2ms。
  - 设定 **LCD_MODE**.VLCD_SEL 选择电压文件位。
  - 当其他 LCD 控制信号设定完成，最后设定 **LCD_CTRL**.LCDEN 开启 LCD。
- 使用外部电压来源，由 VLCD 脚位输入电压。
  - 设定 **LCD_MODE**.VLCD_SRC 选择外部电压源。
  - 设定脚位 PB11 复用功能为 VLCD。(详细的复用功能选择请参考 datasheet)
  - 当其他 LCD 控制信号设定完成，最后设定 **LCD_CTRL**.LCDEN 开启 LCD。

12. 4. 7 控制流程

LCD 的控制流程介绍

- 初始设定，时钟以及电压配置。此区设定因为有电压源的配置，必须按照顺序进行设定
  - 开启 LCD 以及 GPIO 时钟。
  - 配置 LCDCLK 时钟。如有内部电压的需求，开启 VLCDCLK 时钟。
  - 配置电压源，必须按照电压流程进行设定
- LCD 数值以及模式设定
  - 设定输出脚位的复用功能，开启通道输出。
  - 将初始数据写入 **LCD_RAM**。
  - 设定 **LCD_MODE**.DUTY 以及 **LCD_MODE**.BIAS，决定 LCD 的模式以及输出波形。
- LCD 频率以及中断设定
  - 设定 **LCD_CTRL**.FRDIV，LCD 工作频率(LCD_PHY_CLK)分频选择。
  - 设定 **LCD_CTRL**.UPDCTRL，侦更新完成的模式。
  - 开启更新完成 **LCD_IER**.UPDATE 的中断。
- LCD 开启
  - 设定 **LCD_CTRL**.LCDEN，开启 LCD。

- 确认 **LCD_CTRL**.VLCD_RDY 是否为 1，确保 VLCD 电压稳定。

◆ LCD 更新完成中断的处理

- 设定 **LCD_ICR**.UPDCTRL，清除中断。
- 更新显示数据，重新设定 **LCD_RAM**。
- 设定 **LCD_CTRL**.WSBLINK，更改闪烁模式。

12. 5 特殊功能寄存器

12. 5. 1 寄存器列表

LCD 寄存器列表			
名称	偏移地址	类型	描述
LCD_MODE	0000_H	R/W	LCD 模式控制寄存器
LCD_CTRL	0004_H	R/W	LCD 控制寄存器
LCD_BLINK1	0008_H	R/W	LCD 闪烁控制寄存器 1
LCD_BLINK2	$000C_H$	R/W	LCD 闪烁控制寄存器 2
LCD_IER	0010_H	W1	LCD 中断开启寄存器
LCD_IDR	0014_H	W1	LCD 中断关闭寄存器
LCD_IVS	0018_H	R	LCD 中断功能有效状态寄存器
LCD_RIF	$001C_H$	R	LCD 原始中断状态寄存器
LCD_IFM	0020_H	R	LCD 中断标志位状态寄存器
LCD_ICR	0024_H	C_W1	LCD 中断清除寄存器
LCD_BUSY	0028_H	R	LCD 运行状态寄存器
LCD_RAM0	0030_H	R/W	LCD 显示储存寄存器 0
LCD_RAM1	0034_H	R/W	LCD 显示储存寄存器 1
LCD_RAM2	0038_H	R/W	LCD 显示储存寄存器 2
LCD_RAM3	$003C_H$	R/W	LCD 显示储存寄存器 3
LCD_RAM4	0040_H	R/W	LCD 显示储存寄存器 4
LCD_RAM5	0044_H	R/W	LCD 显示储存寄存器 5
LCD_RAM6	0048_H	R/W	LCD 显示储存寄存器 6
LCD_RAM7	$004C_H$	R/W	LCD 显示储存寄存器 7
LCD_RAM8	0050_H	R/W	LCD 显示储存寄存器 8
LCD_RAM9	0054_H	R/W	LCD 显示储存寄存器 9
LCD_RAM10	0058_H	R/W	LCD 显示储存寄存器 10
LCD_RAM11	$005C_H$	R/W	LCD 显示储存寄存器 11

12. 5. 2 寄存器描述

12. 5. 2. 1 LCD模式控制寄存器 (LCD_MODE)

LCD 模式控制寄存器 (LCD_MODE)																															
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
—	—	—	—	—	—	—	—	—	—	POLA_SEL<1:0>		LCD_GPIO	QTSEL<2:0>			—	RSEL<2:0>			VLCD_SEL<3:0>				—	VLCD_SRC	BUFEN	TYPE	BIAS<1:0>		—	DUTY

—	Bit 31-22	—	—
POLA_SEL	Bit 21-20	R/W	**LCD GPIO极性选择** POLA_SEL[1] 0: SEGx数值正向 1: SEGx数值反向 POLA_SEL[0] 0: COMx数值正向 1: COMx数值反向 注:只有在LCD_GPIO=1的情况下，才能设定此寄存器。
LCD_GPIO	Bit 19	R/W	**LCD GPIO模式选择** 0: LCD COM/MIX/SEG由LCD module提供，IO输出为模拟信号。 1: LCD COM/MIX/SEG由数字提供，使用GPIO输出。
QTSEL	Bit 18-16	R/W	**LCD快速充电时间选择** 000: 1 x t_{LCD_CK} 001: 2 x t_{LCD_CK} 010: 3 x t_{LCD_CK} 011: 4 x t_{LCD_CK} 100: 5 x t_{LCD_CK} 101: 6 x t_{LCD_CK} 110: 7 x t_{LCD_CK} 111: 8 x t_{LCD_CK} $t_{LCD\_CK} = 1/f_{LCD\_CK}$
—	Bit 15	—	—
RSEL	Bit 14-12	R/W	**LCD偏压电阻选择**

			000: 225K 001: 900K 010: 60K 011: 60K 100: 225K or 60K 101: 900K or 60K 110: 保留 111: 保留 注: 只有在关闭LCDEN=0时才能对该位进行设置。
VLCD_SEL	Bit 11-8	R/W	**VLCD电压选择** VLCD电压输出2.5V + 0.1V*VLCD_SEL 注: 只有在关闭LCDEN=0时才能对该位进行设置。
—	Bit 7	—	—
VLCD_SRC	Bit 6	R/W	**VLCD电压来源选择** 0: 使用内部电荷泵 1: 使用外部电压 注: 只有在关闭LCDEN=0时才能对该位进行设置。
BUFEN	Bit 5	R/W	**输出缓冲器开启** 0:关闭输出缓冲器 1:开启输出缓冲器 注:只有在关闭LCDEN=0时才能对该位进行设置。
TYPE	Bit 4	R/W	**LCD工作频率(LCD_PHY_CLK)输出类型选择** 0: A 类型(在每一个COM边界改变相位) 1: B 类型(在每一帧幅边界改变相位) 注:只有在关闭LCDEN=0时才能对该位进行设置。
BIAS	Bit 3-2	R/W	**LCD偏压选择** 00:静态 01: 1/2偏压 10: 1/3偏压 11: 1/4偏压 注:只有在关闭LCDEN=0时才能对该位进行设置。
—	Bit 1	—	—

DUTY	Bit 0	R/W	**DUTY输出选择** 00: 1/4 Duty(COM0、COM1、COM2、COM3) 01: 1/6 Duty(COM0、COM1、COM2、COM3、COM4、COM5) 注:只有在关闭LCDEN=0时才能对该位进行设置。

12. 5. 2. 2 LCD控制寄存器 (LCD_CTRL)

LCD 控制寄存器 (LCD_CTRL)

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
VLCD_RDY											WSBLINK<1:0>		FCVALUE<6:0>							FCCTRL	SWUPDATE	UPDCTRL<1:0>		FRDIV<6:0>							LCDEN

VLCD_RDY	Bit 31	R	**LCD电压稳定状态** 0: LCD电压未稳定。 1: LCD电压已稳定。
—	Bit 30-21	—	—
WSBLINK	Bit 20-19	R/W	**全频闪烁选择** 00:关闭闪烁功能 01:闪烁模式0，设定COM0~5寄存器数值反向闪烁 10:闪烁模式1，设定COM0~5寄存器数值闪烁 11:闪烁模式2，设定COM0~5寄存器的0-7位数值闪烁由LCD_BLINK1以及LCD_BLINK2控制
FCVALUE	Bit 18-12	R/W	**帧幅计数器数值** 设置闪烁周期与LCD的数值更新频率 帧幅频率等于f_{LCD}/(FCVALUE<6:0> x 2)时计数器递减。
FCCTRL	Bit 11	R/W	**帧幅计数器开启** 0:关闭帧幅计数器 1:开启帧幅计数器
SWUPDATE	Bit 10	C_W1	**软件生成更新事件** 该位由软件设置，可由硬件在更新完成后自动清零。

			此位设置时，产生更新事件。并在下一个帧幅周期更新COM0~COM5数值。
UPDCTRL	Bit 9-8	R/W	**更新控制选择** 00:当SWUPDATE为1时，下个帧幅周期更新 01:在每个帧幅周期更新 10:当帧幅计数器计数至0时的下个帧幅周期更新 11:保留
FRDIV	Bit 7-1	R/W	**LCD工作频率(LCD_PHY_CLK)分频选择** 设置LCD工作频率 分频系数 ：2*(FRDIV+1)
LCDEN	Bit 0	R/W	**LCD 控制开启** 0:关闭LCD控制 1:开启LCD控制

12. 5. 2. 3 LCD闪烁控制寄存器 1 (LCD_BLINK1)

LCD 闪烁控制寄存器 1 (LCD_BLINK1)																															
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
COM3<7:0>								COM2<7:0>								COM1<7:0>								COM0<7:0>							

COM3	Bit 31-24	R/W	**COM3闪烁选择** 选择COM3的0-7位数值闪烁 注:此位需要在闪烁模式 2(LCD_CTRL.WSBLINK=11)时才有功用。
COM2	Bit 23-16	R/W	**COM2闪烁选择** 选择COM2的0-7位数值闪烁 注:此位需要在闪烁模式 2(LCD_CTRL.WSBLINK=11)时才有功用。
COM1	Bit 15-8	R/W	**COM1闪烁选择** 选择COM1的0-7位数值闪烁 注:此位需要在闪烁模式 2(LCD_CTRL.WSBLINK=11)时才有功用。
COM0	Bit 7-0	R/W	**COM0闪烁选择** 选择COM0的0-7位数值闪烁 注:此位需要在闪烁模式 2(LCD_CTRL.WSBLINK=11)时才有功用。

12. 5. 2. 4 LCD闪烁控制寄存器 2 (LCD_BLINK2)

LCD 闪烁控制寄存器 2 (LCD_BLINK2)																															
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
																COM5<7:0>								COM4<7:0>							

—	Bit 31-16	—	—
COM5	Bit 15-8	R/W	**COM5闪烁选择** 选择COM5的0-7位数值闪烁 注:此位需要在闪烁模式 2(LCD_CTRL.WSBLINK=11)时才有功用。
COM4	Bit 7-0	R/W	**COM4闪烁选择** 选择COM4的0-7位数值闪烁 注:此位需要在闪烁模式 2(LCD_CTRL.WSBLINK=11)时才有功用。

12. 5. 2. 5 LCD中断开启寄存器 (LCD_IER)

LCD 中断开启寄存器 (LCD_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
																															UPDATE

—	Bit 31-1	—	—
UPDATE	Bit 0	W1	**开启更新完成中断功能** 此位设置时，开启中断功能，更新完成时发生中断。

12. 5. 2. 6 LCD中断关闭寄存器 (LCD_IDR)

LCD 中断关闭寄存器 (LCD_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
																															UPDATE

—	Bit 31-1	—	—
UPDATE	Bit 0	W1	**关闭更新完成中断功能** 此位设置时，关闭更新完成中断功能

12. 5. 2. 7 LCD中断功能有效状态寄存器 (LCD_IVS)

LCD 中断功能有效状态寄存器 (LCD_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
																															UPDATE

—	Bit 31-1	—	—
UPDATE	Bit 0	R	**更新完成中断功能开启/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

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

12. 5. 2. 8 LCD原始中断状态寄存器 (LCD_RIF)

LCD 原始中断状态寄存器 (LCD_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
																															UPDATE

—	Bit 31-1	—	—
UPDATE	Bit 0	R	**更新完成时，原始中断状态** 0 :无发生中断 1 :已发生中断

12. 5. 2. 9 LCD中断标志位状态寄存器 (LCD_IFM)

LCD 中断标志位状态寄存器 (LCD_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
																															UPDATE

—	Bit 31-1	—	—
UPDATE	Bit 0	R	**更新完成时，中断标志位状态** 0 :无发生中断 1 :已发生中断

注:LCD_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 LCD_RIF 与 LCD_IVS 进行硬件运算，公式如下:LCD_IFM = LCD_RIF & LCD_IVS

12. 5. 2. 10 LCD中断清除寄存器 (LCD_ICR)

LCD 中断清除寄存器 (LCD_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
																															UPDATE

—	Bit 31-1	—	—
UPDATE	Bit 0	C_W1	**清除更新完成中断状态** 此位设置时，清除中断状态(LCD_RIF 与 LCD_IFM)

注:LCD_ICR 寄存器设置时,将清除 LCD_RIF 与 LCD_IFM 中断标志位状态;此设置不影响中断 LCD_IER、LCD_IDR 与 LCD_IVS 寄存器，只清除标志位状态 LCD_RIF 与 LCD_IFM。此寄存器通过硬件清除中断，公式如下:LCD_RIF = LCD_RIF & !LCD_ICR

12. 5. 2. 11 LCD运行状态寄存器 (LCD_BUSY)

LCD 运行状态寄存器 (LCD_BUSY)																												
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
																															UPDATE

—	Bit 31-1	—	—
UPDATE	Bit 0	R	**更新状态** 当COMx数值由LCD时钟域更新至APB时钟域时由硬件置起，更新完成后自动清除

12. 5. 2. 12 LCD显示储存寄存器 0 (LCD_RAM0)

LCD 显示储存寄存器 0 (LCD_RAM0)																															
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
LCD_RAM0<31:0>																															

LCD_RAM0	Bit 31-0	R/W	**COM0 SEG[31:0]数值** 选择扫描至COM0时，输出SEG0到SEG31状态 0:不点亮(透明) 1:点亮(不透明)

12. 5. 2. 13 LCD显示储存寄存器 1 (LCD_RAM1)

LCD 显示储存寄存器 1 (LCD_RAM1)																															
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
																										LCD_RAM1<5:0>					

—	Bit 31-6	—	—
LCD_RAM1	Bit 5-0	R/W	**COM0 SEG[37:32]数值** 选择扫描至COM0时，输出SEG32到SEG38状态 0:不点亮(透明) 1:点亮(不透明)

12. 5. 2. 14 LCD显示储存寄存器 2 (LCD_RAM2)

LCD 显示储存寄存器 2 (LCD_RAM2)																															
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
LCD_RAM2<31:0>																															

LCD_RAM2	Bit 31-0	R/W	**COM1 SEG[31:0]数值** 选择扫描至COM1时，输出SEG0到SEG31状态 0:不点亮(透明) 1:点亮(不透明)

12. 5. 2. 15 LCD显示储存寄存器 3 (LCD_RAM3)

LCD 显示储存寄存器 3 (LCD_RAM3)																															
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
																										LCD_RAM3<5:0>					

—	Bit 31-6	—	—
LCD_RAM3	Bit 5-0	R/W	**COM1 SEG[37:32]数值** 选择扫描至COM1时，输出SEG32到SEG38状态 0:不点亮(透明) 1:点亮(不透明)

12. 5. 2. 16 LCD显示储存寄存器 4 (LCD_RAM4)

LCD 显示储存寄存器 4 (LCD_RAM4)																															
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
LCD_RAM4<31:0>																															

LCD_RAM4	Bit 31-0	R/W	**COM2 SEG[31:0]数值** 选择扫描至COM2时，输出SEG0到SEG31状态 0:不点亮(透明) 1:点亮(不透明)

12. 5. 2. 17 LCD显示储存寄存器 5 (LCD_RAM5)

LCD 显示储存寄存器 5 (LCD_RAM5)																															
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
																										LCD_RAM5<5:0>					

—	Bit 31-6	—	—
LCD_RAM5	Bit 5-0	R/W	**COM2 SEG[37:32]数值** 选择扫描至COM2时，输出SEG32到SEG38状态 0:不点亮(透明) 1:点亮(不透明)

12. 5. 2. 18 LCD显示储存寄存器 6 (LCD_RAM6)

LCD 显示储存寄存器 6 (LCD_RAM6)																															
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
LCD_RAM6<31:0>																															

LCD_RAM6	Bit 31-0	R/W	**COM3 SEG[31:0]数值** 选择扫描至COM3时，输出SEG0到SEG31状态 0:不点亮(透明) 1:点亮(不透明)

12. 5. 2. 19 LCD显示储存寄存器 7 (LCD_RAM7)

LCD 显示储存寄存器 7 (LCD_RAM7)																															
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
																										LCD_RAM7<5:0>					

—	Bit 31-6	—	—
LCD_RAM7	Bit 5-0	R/W	**COM3 SEG[37:32]数值** 选择扫描至COM3时，输出SEG32到SEG38状态 0:不点亮(透明) 1:点亮(不透明)

12. 5. 2. 20 LCD显示储存寄存器 8 (LCD_RAM8)

LCD 显示储存寄存器 8 (LCD_RAM8)																															
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
LCD_RAM8<31:0>																															

LCD_RAM8	Bit 31-0	R/W	**COM4 SEG[31:0]数值** 选择扫描至COM4时，输出SEG0到SEG31状态 0:不点亮(透明) 1:点亮(不透明)

12. 5. 2. 21 LCD显示储存寄存器 9 (LCD_RAM9)

LCD 显示储存寄存器 9 (LCD_RAM9)																															
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
																										LCD_RAM9<5:0>					

—	Bit 31-6	—	—
LCD_RAM9	Bit 5-0	R/W	**COM4 SEG[37:32]数值** 选择扫描至COM4时，输出SEG32到SEG38状态 0:不点亮(透明) 1:点亮(不透明)

12. 5. 2. 22 LCD显示储存寄存器 10 (LCD_RAM10)

LCD 显示储存寄存器 10 (LCD_RAM10)																															
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
LCD_RAM10<31:0>																															

LCD_RAM10	Bit 31-0	R/W	**COM5 SEG[31:0]数值** 选择扫描至COM5时，输出SEG0到SEG31状态 0:不点亮(透明) 1:点亮(不透明)

12. 5. 2. 23 LCD显示储存寄存器 11 (LCD_RAM11)

LCD 显示储存寄存器 11 (LCD_RAM11)																															
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
—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	LCD_RAM11<5:0>					

—	Bit 31-6	—	—
LCD_RAM11	Bit 5-0	R/W	**COM5 SEG[37:32]数值** 选择扫描至COM5时，输出SEG32到SEG38状态 0:不点亮(透明) 1:点亮(不透明)

第13章 模数转换器(ADC)

13. 1 概述

支持前置滤波器、ADC 输入选择器和温度传感器组成。

13. 2 特性

- 支持仪表放大器增益可配置 0.5/1/2/4/8/16/32/64 倍
- 支持 ADC 输入增益可配置为 1/2/4/8 倍
- 支持 24 位转换结果，最高可达 20 位有效位数(ENOB)
- 支持 ADC 中断标志位
- 支持可配置参考电压，可选择外部或内部参考电压
- 支持可配置 ADC 转换时钟

13. 3 结构图

13. 3. 1 IA结构

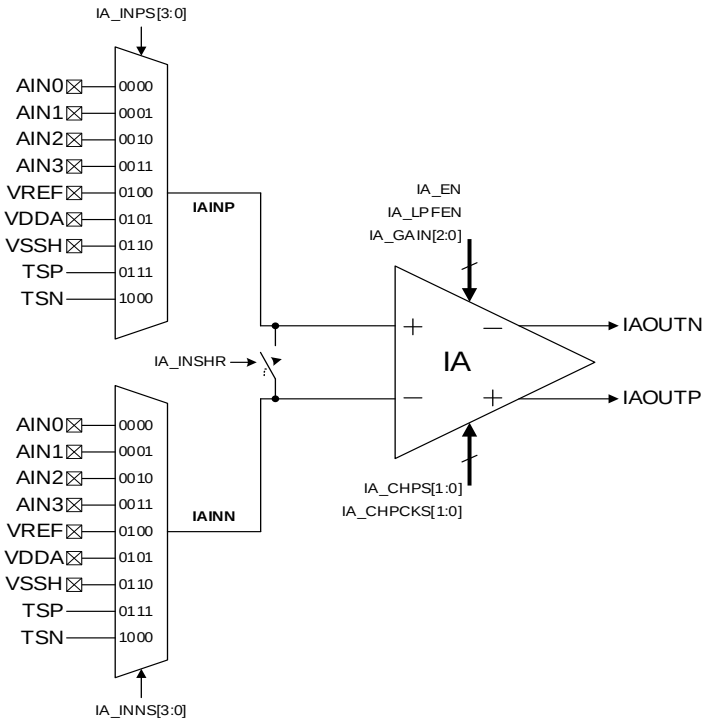


图 13-1 IA 结构图

13. 3. 2 SD-ADC结构

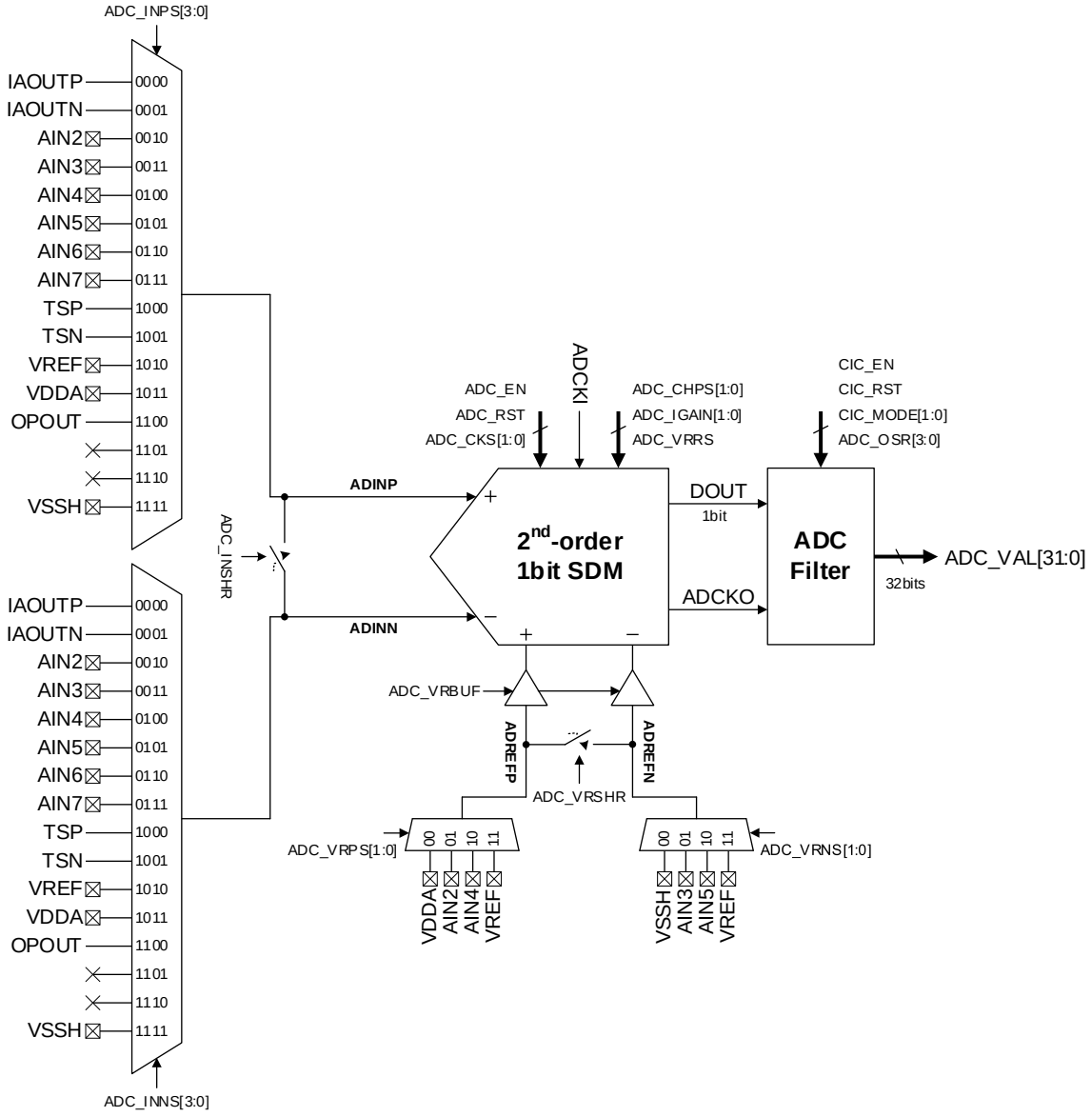


图 13-2 SD-ADC 结构

13. 4 功能描述

13. 4. 1 ADC滤波器(ADC Filter)

ADC 滤波器电路作用为将 SDADC 的 1 位数据进行处理，将高频的量化噪声抑制后再输出多位低速率的数据给用户。

13. 4. 1. 1 SDADC电路建立时间(Settle time of SDADC circuit)

由于 SDADC 启动后(配置 **ADC_CTRL0**.ADC_EN 为 1 到稳定输出需要一个建立时间，因此要避免抓取到不稳定的数值需在建立时间后再启动 ADC 滤波器。

建立时间为 20 个 ADCKO 的时钟周期。

- ADCKO = 1 MHz, Settle Time = 20 us
- ADCKO = 500 kHz, Settle Time = 40 us
- ADCKO = 250 kHz, Settle Time = 80 us

13. 4. 1. 2 滤波器架构

ADC 滤波器使用可配置阶数和过取样率的 CIC 架构。用户可配置 OSR(Oversample Ratio)**ADC_CTRL1**.ADC_OSR ， 从 64 到 32768 。 以及设定阶数(Order)**ADC_CTRL1**.CIC_MODE，可配置 2 到 4 阶。下图为 4 阶的 CIC 滤波器架构示意图：

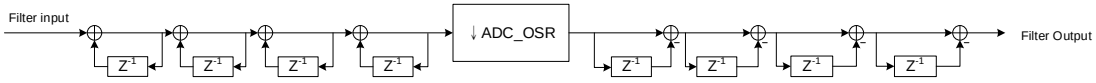
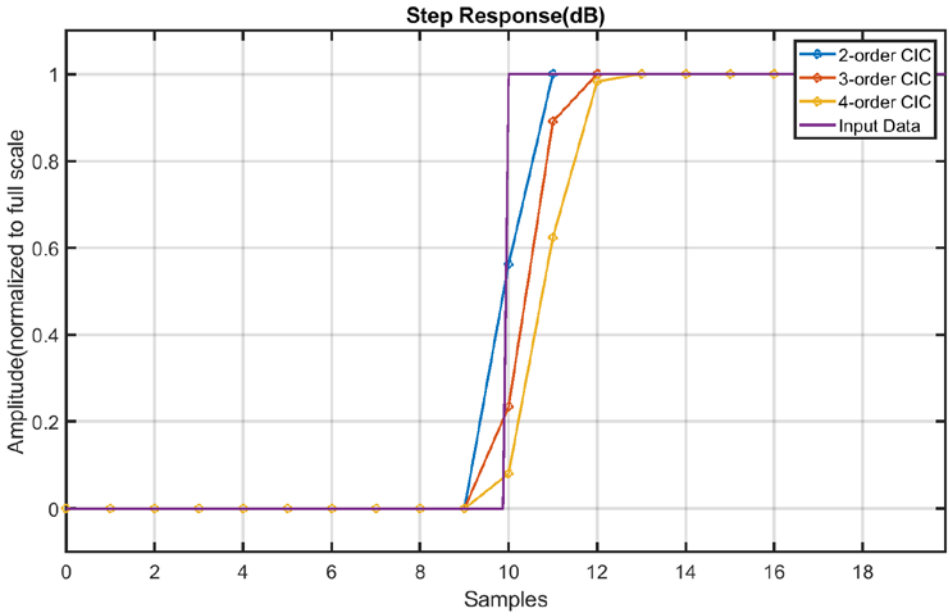


图 13-3 CIC 滤波器架构

13. 4. 1. 3 建立时间和数据转换时间

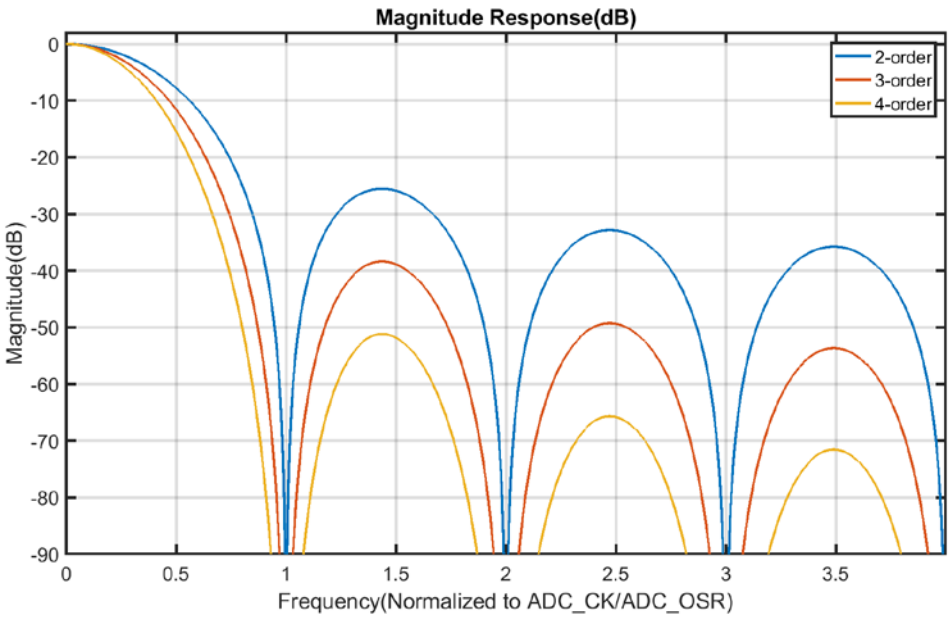
- 单笔转换时间为$\frac{OSR}{ADCKO}$
- 数据建立时间为$\frac{Order \times OSR}{ADCKO}$

下图为不同阶数下，建立时间的比较：



13. 4. 1. 4 频率响应

下图为不同阶数下，滤波器频率响应:



13. 4. 1. 5 ADC数值对应输入电压

ADC_VAL 用 32 位有号整数表示，可表示范围为[-8388608, 8388608]。

其数值除了输入电压、参考电压外和 **ADC_CTRL0** 的 VRRS，INGN 也有相关。

以下列举两个范例计算理论值的 **ADC_VAL**。

- 范例 1
  - 输入电压 : 10 mV
  - 参考电压 : 3.3 V
  - VRRS : 0x0 (ADC 参考电压范围选择 1x)
  - INGN : 0x0 (ADC 输入增益选择 1x)

$$ADC\_VAL=\frac{0.01\times1}{3.3\times1}\times2^{23}=25420$$

- 范例 2
  - 输入电压 : 10 mV
  - 参考电压 : 3.3 V
  - VRRS : 0x1 (ADC 参考电压范围选择 0.5x)
  - INGN : 0x2 (ADC 输入增益选择 4x)

$$ADC\_VAL=\frac{0.01\times4}{3.3\times0.5}\times2^{23}=101680$$

13. 4. 2 系统斩波器

通过硬件周期性变化 IA 的极性并在 ADC 滤波器平均其输出结果后，得到一个抑制 IA 偏移的输出结果。

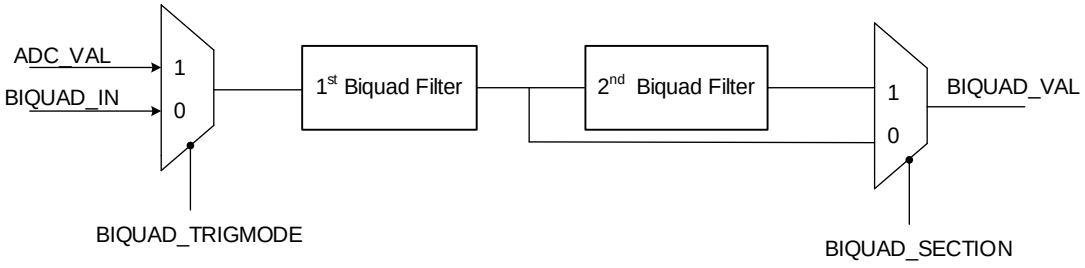
13. 4. 2. 1 建立时间和数据转换时间

- 单笔转换时间为$\frac{Order\times OSR}{ADCKO}$
- 数据建立时间为$2\cdot\frac{Order\times OSR}{ADCKO}$

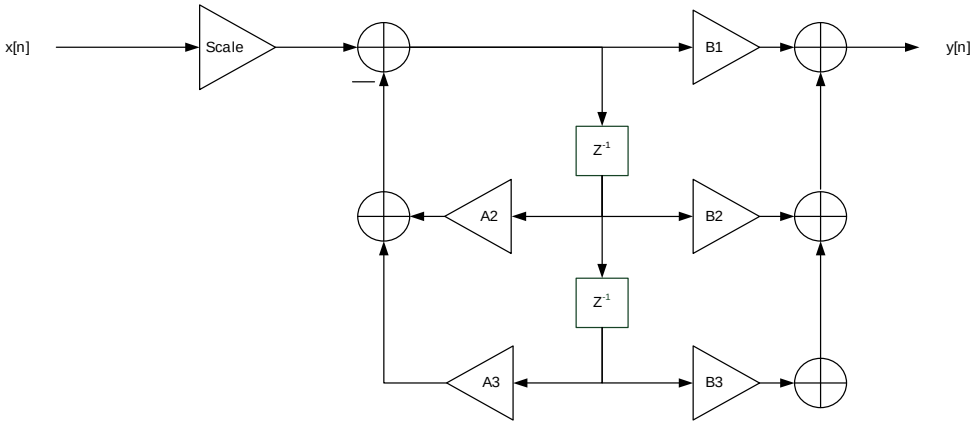
13. 4. 3 双二阶滤波器 (Biqaud Filter)

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 **BIQUAD_CTRL**.BIQUAD_SECTION 配置。一个双二阶滤波器架构如下图所示，由 6 个系数组成。

- 双二阶滤波器结构俯视图



◆ 双二阶滤波器结构



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$Scale\cdot\frac{B_1+B_2Z^{-1}+B_3Z^{-2}}{1+A_2Z^{-1}+A_3Z^{-2}}$

13. 4. 3. 1 数值范围

◆ 输入/输出：28bits 有号数。小数位元数为 23bit，数值范围 [-16, 16) 。

◆ 滤波器系数：28bits 有号数。小数位元数为 23bit，数值范围 [-16, 16) 。

◆ 反馈暂存数值：32bits 有号数。小数位元数为 23bit，数值范围 [-256, 256) 。

13. 4. 3. 2 触发模式选择

共有两种模式可通过寄存器 **BIQUAD_CTRL**.BIQUAD_TRIGMODE 配置。

第一种为使用者对 **BIQUAD_CTRL**.BIQUAD_SWTRIG 写 1 触发运算，输入数据来自 **BIQUAD_IN** 寄存器。

第二种为通过 ADC 滤波器触发运算，其输入数据来自 **ADC_VAL** 寄存器。

13. 4. 4 温度感测

温度传感器由二极管(BJT)组成，其电压信号对温度的变化为已通过 0K 曲线，其具有以下特色：

◆ 温度传感器在环境温度为 0K 为绝对 0 度，其输出的电压值 V_{0k} = 0V。

◆ 通过测量方式可使得模拟数字转换器 ADC 的偏移电压 $V_{ADC-OFFSET}$ 与 BJT 之不对称性自动抵消

TS 初始化配置以及计算斜率

◆ 设定寄存器 **ADC_CTRL0**.ADC_EN 启用 ADC，并且设定 ADC 相关的频率配置，开启 ADC 就会开启 TS 的功能。

- 在同一温度 T_A 下进行量测 ADC 的数值 ADC_{TS0} 以及 ADC_{TS1}
- 量测 ADC_{TS0}，需要将 **ADC_CTRL0**.ADC_INPS 以及 **ADC_CTRL0**.ADC_INNS 皆设定为<1001>。
- 量测 ADC_{TS1}，需要将 **ADC_CTRL0**.ADC_INPS 以及 **ADC_CTRL0**.ADC_INNS 皆设定为<1000>。
- 两数值 ADC_{TS0} 以及 ADC_{TS1} 相减再除 2 得到 $ADC_{TS@TA}$ 可以求得温度 T_A 所得到的电压值。
- TS 的输出电压 V_{TS} 为线性的曲线，可以推导出增益值斜率 G_{TS}。
- 斜率公式为$G_{TS}=\frac{ADC_{TS@TA}}{T_A\text{-}0K}$，0K 的转换会有点误差，可以参考 ADC Datasheet 章节取得。

使用 G_{TS} 计算温度 TX

- 在同一温度 TX 下进行量测 ADC 的数值 ADC_{TX0} 以及 ADC_{TX1}
- 量测 ADC_{TX0}，需要将 **ADC_CTRL0**.ADC_INPS 以及 **ADC_CTRL0**.ADC_INNS 皆设定为<1001>。
- 量测 ADC_{TX1}，需要将 **ADC_CTRL0**.ADC_INPS 以及 **ADC_CTRL0**.ADC_INNS 皆设定为<1000>。
- 两数值 ADC_{TX0} 以及 ADC_{TX1} 相减再除 2 得到 $ADC_{TS@TX}$。
- 将 $ADC_{TS@TX}$ 经过使用 G_{TS} 进行运算，得到对应的温度。
- $TX=\frac{ADC_{TS@TX}}{G_{TS}}+0K$

13. 5 特殊功能寄存器

13. 5. 1 寄存器列表

ADC 寄存器列表			
名称	偏移地址	类型	描述
IA_CTRL	0000_H	R/W	IA 配置寄存器
ADC_CTRL0	0004_H	R/W	ADC 配置寄存器 0
ADC_CTRL1	0008_H	R/W	ADC 配置寄存器 1
ADC_VAL	$000C_H$	R	ADC 数值
BIQUAD_CTRL	0010_H	R/W	BIQUAD 配置寄存器
BIQUAD_VAL	0014_H	R	BIQUAD 计算结果数值
BIQUAD_IN	0018_H	R/W	BIQUAD 滤波器输入数值
ADC_IER	0030_H	W	ADC 中断开启寄存器
ADC_IDR	0034_H	W	ADC 中断关闭寄存器
ADC_IVS	0038_H	R	ADC 中断功能有效状态寄存器
ADC_RIF	$003C_H$	R	ADC 原始中断状态寄存器
ADC_IFM	0040_H	R	ADC 中断标志位状态寄存器
ADC_ICR	0044_H	W	ADC 中断清除寄存器
COEF_SC_1	0050_H	R/W	BIQUAD 第一级增益系数
COEF_B1_1	0054_H	R/W	BIQUAD 第一级 B1 系数
COEF_B2_1	0058_H	R/W	BIQUAD 第一级 B2 系数
COEF_B3_1	$005C_H$	R/W	BIQUAD 第一级 B3 系数
COEF_A2_1	0060_H	R/W	BIQUAD 第一级 A2 系数
COEF_A3_1	0064_H	R/W	BIQUAD 第一级 A3 系数
COEF_SC_2	0068_H	R	BIQUAD 第二级增益系数
COEF_B1_2	$006C_H$	R	BIQUAD 第二级 B1 系数
COEF_B2_2	0070_H	R	BIQUAD 第二级 B2 系数
COEF_B3_2	0074_H	R	BIQUAD 第二级 B3 系数
COEF_A2_2	0078_H	R/W	BIQUAD 第二级 A2 系数
COEF_A3_2	$007C_H$	R/W	BIQUAD 第二级 A3 系数

13. 5. 2 寄存器描述

13. 5. 2. 1 IA配置寄存器 (IA_CTRL)

IA 配置寄存器 (IA_CTRL)																													
偏移地址:0x000																													
复位值:0x0001 0090																													
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
—	—	—	—	—	—	—	—	—	—	IA_POLA_ST	—	—	IA_GAIN<2:0>			IA_INNS<3:0>				IA_INPS<3:0>				IA_CHPCKS<1:0>		IA_CHPS<1:0>		IA_LPFEN	—	IA_INSHR	IA_EN

—	Bits 31-22	—	—
IA_POLA_ST	Bit 21	R	**仪表放大器输入极性状态** 0: 正端与负端信号维持 1: 正端与负端信号对调
—	Bits 20-19	—	—
IA_GAIN	Bit 18-16	R/W	**仪表放大器增益选择** 000: 0.5x 001: 1x 010: 2x 011: 4x 100: 8x 101: 16x 110: 32x 111: 64x
IA_INNS	Bit 15-12	R/W	**仪表放大器负端输入选择** 0000: AIN0 0001: AIN1 0010: AIN2 0011: AIN3 0100: VREF 0101: VDDA 0110: VSSH 0111: TSP 1000: TSN
IA_INPS	Bit 11-8	R/W	**仪表放大器正端输入选择** 0000: AIN0 0001: AIN1

			0010:AIN2 0011: AIN3 0100: VREF 0101: VDDA 0110: VSSH 0111: TSP 1000: TSN
IA_CHPCKS	Bit 7-6	R/W	**仪表放大器斩波器时钟选择** 00: 32 分频(125 kHz) 01: 16 分频(250 kHz) 10: 8 分频(500 kHz) 11: No clock 注: 1. 括号内为 CHOPCLK 被配置为 4MHz 时，对应斩波器内部的时钟频率。 2. CHOPCLK 配置方式请参考 RCU "Chopper 时钟"章节。
IA_CHPS	Bit 5-4	R/W	**仪表放大器斩波器控制** 00：模拟斩波器 OFF，系统斩波器 OFF 01: 模拟斩波器 ON，系统斩波器 OFF 10: 模拟斩波器 OFF，系统斩波器 ON 11: 模拟斩波器 ON，系统斩波器 ON 注: 启用系统斩波器时，ADC_INNS/ADC_INPS 必须同时配置为 4’b0000 或 4’b0001.
IA_LPFEN	Bit 3	R/W	**仪表放大器低通滤波器开关** 0: 禁用 1: 启用
—	Bits 2	—	—
IA_INSHR	Bit 1	R/W	**仪表放大器输入短路控制** 0: 禁用 1: 启用
IA_EN	Bit 0	R/W	**仪表放大器开关** 0: 禁用 1: 启用

13. 5. 2. 2 ADC配置寄存器 0 (ADC_CTRL0)

ADC 配置寄存器 (ADC_CTRL0)																													
偏移地址:0x004																													
复位值:0x0040 00E0																													
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADC_RST	—	—	—	—	—	—	ADC_CURSEL	—	ADC_VRRS	ADC_INGN<1:0>		ADC_VRNS<1:0>		ADC_VRPS<1:0>		ADC_INNS<3:0>				ADC_INPS<3:0>				ADC_CHPS<1:0>		ADC_CKS<1:0>		ADC_VRBUF	ADC_VRSHR	ADC_INSHR	ADC_EN

ADC_RST	Bits 31	W1C	**ADC 复位** 0: 无效 1: 复位
—	Bits 30-25	—	—
ADC_CURSEL	Bit 24	R/W	**ADC 电流选择** 0: 低电流 1: 高电流
—	Bits 23	—	—
ADC_VRRS	Bit 22	R/W	**ADC 参考电压范围选择** 0: 1x 1: 0.5x
ADC_INGN	Bit 21-20	R/W	**ADC 输入增益选择** 00: 1x 01: 2x 10: 4x 11: 8x
ADC_VRNS	Bit 19-18	R/W	**ADC 参考电压负端输入选择** 00: VSSH 01: AIN3 10: AIN5 11: VREF
ADC_VRPS	Bit 17-16	R/W	**ADC 参考电压正端输入选择** 00: VDDA 01: AIN2 10: AIN4 11: VREF
ADC_INNS	Bit 15-12	R/W	**ADC 负端输入选择** 0000: IAOUTP

			0001: IAOUTN 0010: AIN2 0011: AIN3 0100: AIN4 0101: AIN5 0110: AIN6 0111: AIN7 1000: TSP 1001: TSN 1010: VREF 1011: VDDA 1100: OPOUT 1101: 保留 1110: 保留 1111: VSSH
ADC_INPS	Bit 11-8	R/W	**ADC 正端输入选择** 0000: IAOUTP 0001: IAOUTN 0010: AIN2 0011: AIN3 0100: AIN4 0101: AIN5 0110: AIN6 0111: AIN7 1000: TSP 1001: TSN 1010: VREF 1011: VDDA 1100: OPOUT 1101: 保留 1110: 保留 1111: VSSH
ADC_CHPS	Bit 7-6	R/W	**ADC 斩波器选择** 00：斩波禁用,双相关取样禁用 01: 双相关取样启用 10: 斩波启用 11: 斩波启用，双相关取样启用
ADC_CKS	Bit 5-4	R/W	**ADC 时钟选择**

			00 : 16 分频 (250 kHz) 01 : 8 分频 (500 kHz) 10 : 4 分频 (1 MHz) 11 : 保留 注 : 1. 括号内为 ADCKI 被配置为 4MHz 时，对应的 ADCKO 频率。 2. ADCKI 配置方式请参考 RCU "ADC 时钟" 章节。
ADC_VRBUF	Bit 3	R/W	**ADC 参考电压输入缓冲器开关** 0: 禁用 1: 启用
ADC_VRSHR	Bit 2	R/W	**ADC 参考电压短路控制** 0: 禁用 1: 启用
ADC_INSHR	Bit 1	R/W	**ADC 输入短路控制** 0: 禁用 1: 启用
ADC_EN	Bit 0	R/W	**ADC 开关** 0: 禁用 1: 启用

13. 5. 2. 3 ADC配置寄存器 1 (ADC_CTRL1)

ADC 配置寄存器 1 (ADC_CTRL1)																													
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
								READY_FLAG																ADC_OSR<3:0>				CIC_ORDER<1:0>		CIC_RST	CIC_EN

—	Bits 31-24	—	—
READY_FLAG	Bits 23	R	**输出数据稳定标志位** 当 ADC 滤波器启动超过对应的建立时间后会将此标志位置 1。 此建立时间仅考虑 ADC 滤波器本身的稳定时间。 若系统斩波器启动时，则需要 2 倍的建立时间。 0: 尚未稳定 1: 已稳定
—	Bits 22-8	—	—
ADC_OSR	Bit 7-4	R/W	**ADC 过采样输出频率设置** 0000: 64 0001: 128 0010: 256 0011: 512 0100: 1024 0101: 2048 0110: 4096 0111: 8192 1000: 16384 其他:32768
CIC_MODE	Bit 3-2	R/W	**CIC 滤波器阶数选择** 00: 2-order CIC 01: 3-order CIC 10: 4-order CIC 11: 保留
CIC_RST	Bit 1	C_W1	**CIC 滤波器复位** 0: 无作用

			1: 写 1 对滤波器复位，会自动清回 0。
CIC_EN	Bit 0	R/W	**CIC 滤波器开关** 0: 禁用 1: 启用

13. 5. 2. 4 ADC数值(ADC_VAL)

ADC 数值 (ADC_VAL)

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
ADC_VAL<31:0>																															

ADC_VAL	Bits 31-0	R	**ADC 数值** 表示方式为 32bits 有号数。 最小值: 0xFF800000 (-8388608) 最大值: 0x00800000 (+8388608)

13. 5. 2. 5 BIQUAD配置寄存器 (BIQUAD_CTRL)

BIQUAD 配置寄存器 (BIQUAD_CTRL)																												
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
																							BIQUAD_SWTRIG					BIQUAD_TRIGMODE	BIQUAD_SECTION	BIQUAD_RST	BIQAUD_EN

—	Bits 31-9	—	—
BIQUAD_SWTRIG	Bit 8	C_W1	**BIQAUD 滤波器软件触发开关** 0: 无作用 1: 写 1 触发滤波器
—	Bits 7-4	—	—
BIQUAD_TRIGMODE	Bit 3	R/W	**BIQAUD 滤波器触发模式选择** 0: 由用户配置 BIQUAD_SWTRIG 触发运算，并将 BIQUAD_IN 作为滤波器的输入数值 1: 由 ADC 滤波器触发运算，并将 ADC_VAL 值作为滤波器的输入数值
BIQUAD_SECTION	Bit 2	R/W	**BIQAUD 滤波器阶数选择** 0: 1 级 Filter 1: 2 级 Filter 串接
BIQUAD_RST	Bit 1	C_W1	**BIQUAD 滤波器复位** 0: 无作用 1: 写 1 对滤波器复位，会自动清回 0
BIQAUD_EN	Bit 0	R/W	**BIQUAD 滤波器开关** 0: 禁用 1: 启用

13. 5. 2. 6 BIQUAD 计算结果数值 (BIQUAD_VAL)

BIQUAD 计算结果数值 (BIQUAD_VAL)																														
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
BIQUAD_VAL <31:0>																														

BIQUAD_VAL	Bits 31-0	R	**BIQAUD 滤波器输出数值** 32bits 有号数。 最小值 : 0xF8000000 (-134217728) 最大值 : 0x08000000 (+134217728)

13. 5. 2. 7 BIQUAD滤波器输入数值 (BIQUAD_IN)

BIQUAD 滤波器输入数值 (BIQUAD_IN)																														
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

BIQUAD_IN	Bits 31-0	R/W	**BIQAUD 滤波器输入数值** 当设定 BIQUAD_TRIGMODE=0 时，作为 BIQUAD 滤波器的输入。 32bits 有号数。 最小值 : 0xF8000000 (-134217728) 最大值 : 0x08000000 (+134217728)

13. 5. 2. 8 ADC中断开启寄存器(ADC_IER)

ADC 中断开启寄存器(ADC_IER)																														
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
																														BIQUAD_DONE	ADC_DONE

—	Bits 31-2	—	—
BIQUAD_DONE	Bit 1	W1	**BIQUAD 完成中断使能** 0: 写 0 无效 1: 开启 BIQUAD 完成中断
ADC_DONE	Bit 0	W1	**ADC 完成中断使能** 0: 写 0 无效 1: 开启 ADC 完成中断

13. 5. 2. 9 ADC中断关闭寄存器(ADC_IDR)

ADC 中断关闭寄存器(ADC_IDR)																														
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
																														BIQUAD_DONE	ADC_DONE

—	Bits 31-2	—	—
BIQUAD_DONE	Bit 1	W1	**BIQUAD 完成中断禁止** 0: 写 0 无效 1: 禁止 BIQUAD 完成中断
ADC_DONE	Bit 0	W1	**ADC 完成中断禁止** 0: 写 0 无效 1: 禁止 ADC 完成中断

13. 5. 2. 10 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: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
																														BIQUAD_DONE	ADC_DONE

—	Bits 31-2	—	—
BIQUAD_DONE	Bit 1	R	**BIQUAD 完成中断状态** 0: 关闭 BIQUAD 完成中断状态 1: 开启 BIQUAD 完成中断状态
ADC_DONE	Bit 0	R	**ADC 完成中断状态** 0: 关闭 ADC 完成中断状态 1: 开启 ADC 完成中断状态

13. 5. 2. 11 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: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
																														BIQUAD_DONE	ADC_DONE

—	Bits 31-2	—	—
BIQUAD_DONE	Bit 1	R	**BIQUAD 完成，原始中断状态** 0: 无发生中断 1: BIQUAD 完成中断状态。
ADC_DONE	Bit 0	R	**ADC 完成，原始中断状态** 0: 无发生中断 1: ADC 完成中断状态。

13. 5. 2. 12 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: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
																														BIQUAD_DONE	ADC_DONE

—	Bits 31-2	—	—
BIQUAD_DONE	Bit 1	R	**BIQUAD 完成，中断状态标志位** 0: 未发生中断或中断未使能 1: BIQUAD 完成中断
ADC_DONE	Bit 0	R	**ADC 完成，中断状态标志位** 0: 未发生中断或中断未使能 1: ADC 完成中断

13. 5. 2. 13 ADC中断清除寄存器(ADC_ICR)

ADC 中断清除寄存器(ADC_ICR)																													
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
																														BIQUAD_DONE	ADC_DONE

—	Bits 31-2	—	—
BIQUAD_DONE	Bit 1	W1	**BIQUAD 完成，原始中断状态(ADC_RIF 与 ADC_IFM)** 0：写 0 无效 1：清除中断事件与中断
ADC_DONE	Bit 0	W1	**ADC 完成，原始中断状态(ADC_RIF 与 ADC_IFM)** 0：写 0 无效 1：清除中断事件与中断

13. 5. 2. 14 BIQUAD第一级增益系数 (COEF_SC_1)

BIQUAD 第一级增益系数 (COEF_SC_1)																															
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
COEF_SC_1<31:0>																															

COEF_SC_1	Bits 31-0	R/W	**BIQAUD 第一级增益系数** 32bits 有号数。 最小值 : 0xF8000000 (-134217728) 最大值 : 0x08000000 (+134217728) 系数数值 $=\frac{BIQUAND\_IN}{2^{23}}$，数值可表示范围为[-16, 16)。

13. 5. 2. 15 BIQUAD第一级B1 系数 (COEF_B1_1)

BIQUAD 第一级 B1 系数 (COEF_B1_1)																															
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
COEF_B1_1<31:0>																															

COEF_B1_1	Bits 31-0	R/W	**BIQAUD 第一级 B1 系数** 数值范围请参考 COEF_SC_1

13. 5. 2. 16 BIQUAD第一级B2 系数 (COEF_B2_1)

BIQUAD 第一级 B2 系数 (COEF_B2_1)																															
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
COEF_B2_1<31:0>																															

COEF_B2_1	Bits 31-0	R/W	**BIQAUD 第一级 B2 系数** 数值范围请参考 COEF_SC_1

13. 5. 2. 17 BIQUAD第一级B3 系数 (COEF_B3_1)

BIQUAD 第一级 B3 系数 (COEF_B3_1)																															
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
COEF_B3_1<31:0>																															

COEF_B3_1	Bits 31-0	R/W	**BIQAUD 第一级 B3 系数** 数值范围请参考 COEF_SC_1

13. 5. 2. 18 BIQUAD第一级A2 系数 (COEF_A2_1)

BIQUAD 第一级 A2 系数 (COEF_A2_1)																															
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
COEF_A2_1<31:0>																															

COEF_A2_1	Bits 31-0	R/W	**BIQAUD 第一级 A2 系数** 数值范围请参考 COEF_SC_1

13. 5. 2. 19 BIQUAD第一级A3 系数 (COEF_A3_1)

BIQUAD 第一级 A3 系数 (COEF_A3_1)																															
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
COEF_A3_1<31:0>																															

COEF_A3_1	Bits 31-0	R/W	**BIQAUD 第一级 A3 系数** 数值范围请参考 COEF_SC_1

13. 5. 2. 20 BIQUAD第二级增益系数 (COEF_SC_2)

BIQUAD 第二级增益系数 (COEF_SC_2)																															
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
COEF_SC_2<31:0>																															

COEF_SC_2	Bits 31-0	R/W	**BIQAUD 第二级增益系数** 数值范围请参考 COEF_SC_1

13. 5. 2. 21 BIQUAD第二级B1 系数 (COEF_B1_2)

BIQUAD 第二级 B1 系数 (COEF_B1_2)																															
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
COEF_B1_2<31:0>																															

COEF_B1_2	Bits 31-0	R/W	**BIQAUD 第二级 B1 系数** 数值范围请参考 COEF_SC_1

13. 5. 2. 22 BIQUAD第二级B2 系数 (COEF_B2_2)

BIQUAD 第二级 B2 系数 (COEF_B2_2)																															
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
COEF_B2_2<31:0>																															

COEF_B2_2	Bits 31-0	R/W	**BIQAUD 第二级 B2 系数** 数值范围请参考 COEF_SC_1

13. 5. 2. 23 BIQUAD第二级B3 系数 (COEF_B3_2)

BIQUAD 第二级 B3 系数 (COEF_B3_2)																															
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
COEF_B3_2<31:0>																															

COEF_B3_2	Bits 31-0	R/W	**BIQAUD 第二级 B3 系数** 数值范围请参考 COEF_SC_1

13. 5. 2. 24 BIQUAD第二级A2 系数 (COEF_A2_2)

BIQUAD 第二级 A2 系数 (COEF_A2_2)																															
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
COEF_A2_2<31:0>																															

COEF_A2_2	Bits 31-0	R/W	**BIQAUD 第二级 A2 系数** 数值范围请参考 COEF_SC_1

13. 5. 2. 25 BIQUAD第二级A3 系数 (COEF_A3_2)

BIQUAD 第二级 A3 系数 (COEF_A3_2)																															
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
COEF_A3_2<31:0>																															

COEF_A3_2	Bits 31-0	R/W	**BIQAUD 第二级 A3 系数** 数值范围请参考 COEF_SC_1

第14章　轨对轨运算放大器 (OPAMP)

14. 1 概述

芯片嵌入一个轨对轨运算放大器网络(Rail-to-Rail OPAMP)，主要用于模拟信号处理。输入范围和输出范围都是从 VSS 至 VDDA。输入信号范围为 VSS +0.1V 和 VDDA-0.1V 之间时，开环增益为 80dB 以上。

14. 2 特性

- 轨对轨输入范围，以及轨对轨输出范围
- 在 22pF 负载情况下，它有 1MHz 单位的增益带宽和 60 相位裕量
- 直流增益 80dB 以上
- 1mA 推挽输出驱动能力
- 正输入端有 7 个独立的选择开关，负输入端有 7 独立的选择开关
- 内置 10pF 电容
- 可作为比较器使用，作为比较器时具有斩波器功能
- 内置尖峰脉冲数字低通滤波器

14. 3 结构图

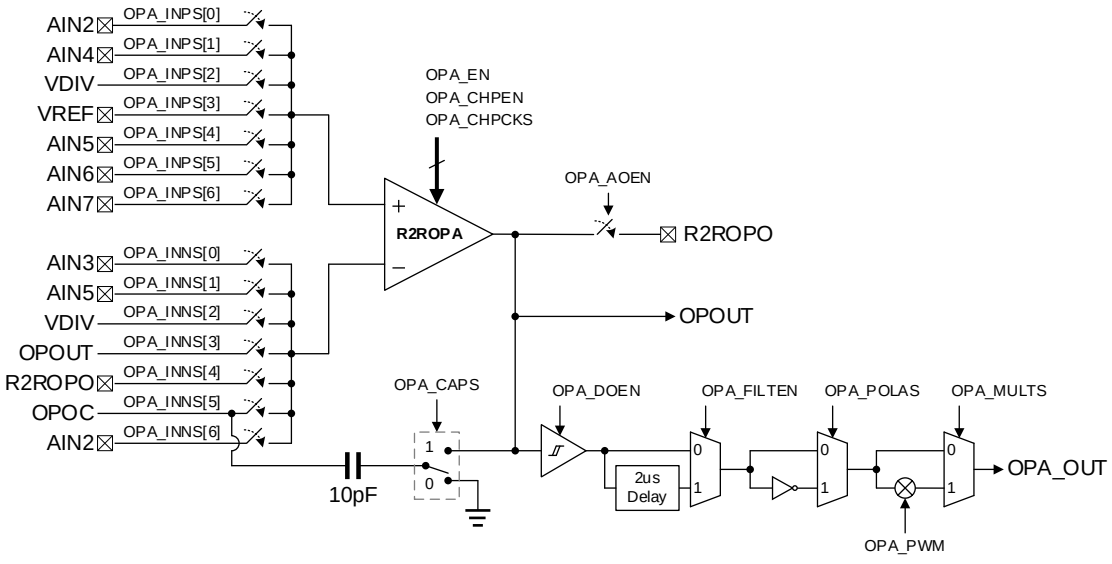


图 14-1　OPAMP 结构图

14. 4 功能描述

14. 4. 1 输入通道选择开关

运算放大器的输入通道皆是独立的选择开关，正端以及负端的皆由 7 个独立开关控制。

正端输入由 **OPA_CTRL**.OPA_INPS 控制开关，输入信号为 AIN2、AIN4、VDIV、VREF、AIN5、AIN6、AIN7。

负端输入由 **OPA_CTRL**.OPA_INNS 控制开关，输入信号为 AIN3、AIN5、VDIV、OPOUT、R2ROPO、OPOC、AIN2。

需要先将脚位 PA03、PA04 以及 PA05 设定为复用功能，选择 AIN5、AIN6 以及 AIN7，才能通过该信道输入信号传入运算放大器。(详细的复用功能选择请参考相关的 datasheet)

14. 4. 2 模拟输出(R2ROPO)

运算放大器的模拟输出脚位为 PA07，需要将该脚位的复用功能设置为 R2ROPO。此外，还需要将 OPA_CTRL.AOEN 设置为 1。(详细的复用功能选择请参考相关的 datasheet)

14. 4. 3 斩波器(Chopper)

运算放大器有斩波器的功能，**OPA_CTRL**.OPA_CHPEN。当要使用斩波器功能，需要提供斩波器时钟正确的频率。可以通过 **RCU_CFG2**.CHOP_CKCFG 以及 **RCU_CFG2**.CHOP_CKDIV 的配置，提供斩波器时钟 4MHz。

14. 4. 4 内置 10pF电容

OPAMP 内置一个 10pF 的电容，它在不同设置下有不同的功能。电容的上端连接到 OPOC，它可以连接到运算放大器的负输入端，该开关是由控制位 **OPA_CTRL**.INNS 设置；电容的下端可连接到 OPOUT 或 VSS，可通过控制位 **OPA_CTRL**.CAPS 的设置选择。

有两种方法来采样模拟输入。

- 开环采样技术
- 死循环采样技术

14. 4. 5 比较器功能

若将运放配置为开环功能，则轨对轨运算放大器可以作为一个比较器使用。通过 **OPA_CTRL**.OPA_DOEN 输出数字信号 0/1。若正向输入大于负向输入，则 **OPA_CTRL**.OPA_OUT 输出 1;若正向输入小于负向输入,则 **OPA_CTRL**.OPA_OUT 输出 0。

为了防止尖峰脉冲干扰，数字信号输出还可以经过 2us 的低通滤波器。如果任何尖峰脉冲小于 2us，比较器输出结果不会改变。比较器数字信号相位可以通过控制位 **OPA_CTRL**.POLAS 的设置来改变。

比较器的输出结果还可以通过 **OPA_CTRL**.OPA_MULTS 以及 **OPA_CTRL**.OPA_MULTSRC 选择 TIMER 输出的信号进行相乘，输出一个高频信号，可以作为 LED 驱动器。比较器的输出 OPA_OUT 也可以连接到 I/O 引脚。(输出的脚位可参考 Datasheet)

OPA_MULTSRC[2:0]	OPA_PWM
000	GP32C4T1_CH1
001	GP32C4T1_CH2
010	GP32C4T1_CH3
011	GP32C4T1_CH4
100	GP16C2T1_CH1
101	GP16C2T2_CH1
110	GP16C2T3_CH1
111	GP16C2T4_CH1

- 数字滤波器

  可调低通滤波器，使用弹跳消除取样(debounce)的功能，可以经过时钟频率的分频以及取样 OPAMP 的输出，达到防止尖峰脉冲干扰。

  用户设定 OPAMP 系统时钟，时间为 T_S，**OPA_CTRL**.OPA_FILTEN 开启滤波器功能，设置滤波器的参数 **OPA_DBC**.DBCNT 以及 **OPA_DBC**.DBPRE。Debounce 的公式为:

  $$\text{Debounce Time}=\{(\text{DBPRE}+1)\times\text{DBCNT}\}\times T_S$$

  举例 2us 低通滤波器，在 OPAMP 系统时钟为 48 MHz，T_S =2.0833e−08 us 条件下的配置流程

- 设置 **OPA_CTRL**.OPA_FILTEN 开启滤波器功能。
- 设置滤波器参数 **OPA_DBC**.DBCNT 以及 **OPA_DBC**.DBPRE，使用不同的设定值，可以得到相同的滤波器。
  - ◇ 设置 **OPA_DBC**.DBCNT = 96，**OPA_DBC**.DBPRE = 0。
  - ◇ 设置 **OPA_DBC**.DBCNT = 6，**OPA_DBC**.DBPRE = 7。

14. 5 特殊功能寄存器

14. 5. 1 寄存器列表

OPAMP 寄存器列表			
名称	偏移地址	类型	描述
OPA_CTRL	0000_H	R/W	OPA 配置寄存器
OPA_DBC	0004_H	R/W	OPA 弹跳消除取样率控制寄存器

14. 5. 2 寄存器描述

14. 5. 2. 1 OPA配置寄存器(OPA_CTRL)

OPA 配置寄存器(OPA_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
OPA_OUT	OPA_MULTSRC<2:0>			OPA_MULTS	OPA_POLAS	OPA_FILTEN	OPA_DOEN	-	OPA_INNS<6:0>							-	OPA_INPS<6:0>							-	-	OPA_AOEN	OPA_CAPS	-	OPA_CHPCKS	OPA_CHPEN	OPA_EN

OPA_OUT	Bit 31	R	**OPAMP数字输出值** 0:负端输入端信号>正端输入端信号 1:正端输入端信号>负端输入端信号
OPA_MULTSRC	Bit 30-28	R/W	**OPA_PWM来源选择** 000: GP32C4T1_CH1。 001: GP32C4T1_CH2。 010: GP32C4T1_CH3。 011: GP32C4T1_CH4。 100: GP16C2T1_CH1。 101: GP16C2T2_CH1。 110: GP16C2T3_CH1。 111: GP16C2T4_CH1。
OPA_MULTS	Bit 27	R/W	**OPAMP输出经过OPA_PWM多任务器选择** 0:不经过多功能器 1:经过多功能器，OPAMP输出选加上OPA_PWM
OPA_POLAS	Bit 26	R/W	**OPAMP数字输出相位选择** 0:正相输出 1:反相输出

OPA_FILTEN	Bit 25	R/W	**OPAMP数字滤波器开关** 0:禁用 1:启用
OPA_DOEN	Bit 24	R/W	**OPAMP数字输出开关** 0:禁用 1:启用
—	Bit 23	—	—
OPA_INNS	Bit 22-16	R/W	**OPAMP 负端通道输入开关选择** 每个bit的设置皆对应到通道开关。 OPA_INNS[x], x = 0-6。 0: 禁用 1: 启用 OPA_INNS[0] : AIN3 OPA_INNS[1] : AIN5 OPA_INNS[2] : VDIV OPA_INNS[3] : OPOUT OPA_INNS[4] : R2ROPO OPA_INNS[5] : OPOC OPA_INNS[6] : AIN2
—	Bit 15	—	—
OPA_INPS	Bit 14-8	R/W	**OPAMP正端通道输入开关选择** 每个bit的设置皆对应到通道开关。 OPA_INPS[x], x = 0-6。 0: 禁用 1: 启用 OPA_INPS[0] : AIN2 OPA_INPS[1] : AIN4 OPA_INPS[2] : VDIV OPA_INPS[3] : VREF OPA_INPS[4] : AIN5 OPA_INPS[5] : AIN6 OPA_INPS[6] : AIN7
—	Bit 7-6	—	—
OPA_AOEN	Bit 5	R/W	**OPAMP模拟输出开关** 0:禁用 1:启用
OPA_CAPS	Bit 4	R/W	**OPAMP内置电容用途设置** 0:电容作为采样电容，下端连接至VSSA

			1:电容作为集成电容器，下端连接至OPOUT
—	Bit 3	-	—
OPA_CHPCKS	Bit 2	R/W	**OPAMP斩波器时钟选择** 0: 1 MHz 1: 2 MHz
OPA_CHPEN	Bit 1	R/W	**OPAMP 斩波器开关** 0:禁用 1:启用
OPA_EN	Bit 0	R/W	**OPAMP 开关** 0:禁用 1:启用

14. 5. 2. 2 弹跳消除取样率控制寄存器(OPA_DBC)

弹跳消除取样率控制寄存器 (OPA_DBC)

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
																					DBCNT<2:0>			DBPRE<7:0>									

—	Bit 31-11	—	—
DBCNT	Bit 10-8	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。
DBPRE	Bit 7-0	R/W	**DBPRE: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。

第15章 模拟电源控制 (ANPWR)

15. 1 概述

模拟电源结构通常可以分为三个区块：模拟稳压电源(VDDA)、参考电压源(VREF)和 LCD 稳压电源(VLCD)。在本章节中，我们将着重介绍 VDDA 和 VREF 的产生方式。

VDDA 主要用于提供模拟/数字转换器(SD-ADC)、仪表放大器(IA)、参考电压(VREF)和轨到轨运算放大器(OPAMP)的供电，或作为这些组件的输入电压。VREF 提供了模拟/数字转换器(SD-ADC)、仪表放大器(IA)和轨对轨运算放大器(OPAMP)的输入电压。

15. 2 特性

- 模拟电路使用的稳压电路 LDO 输出电压 VDDA。
- VDDA 有四种不同的操作模式: VDDH 短路、接地放电、高阻态以及可调节稳压。
- VREF 为 ADC 需要的参考电压。
- VREF 可选择外部或内部的电源。
- VREF 的内部电源可以选择 VBG 以及 VDDA 的分压。

15. 3 结构图

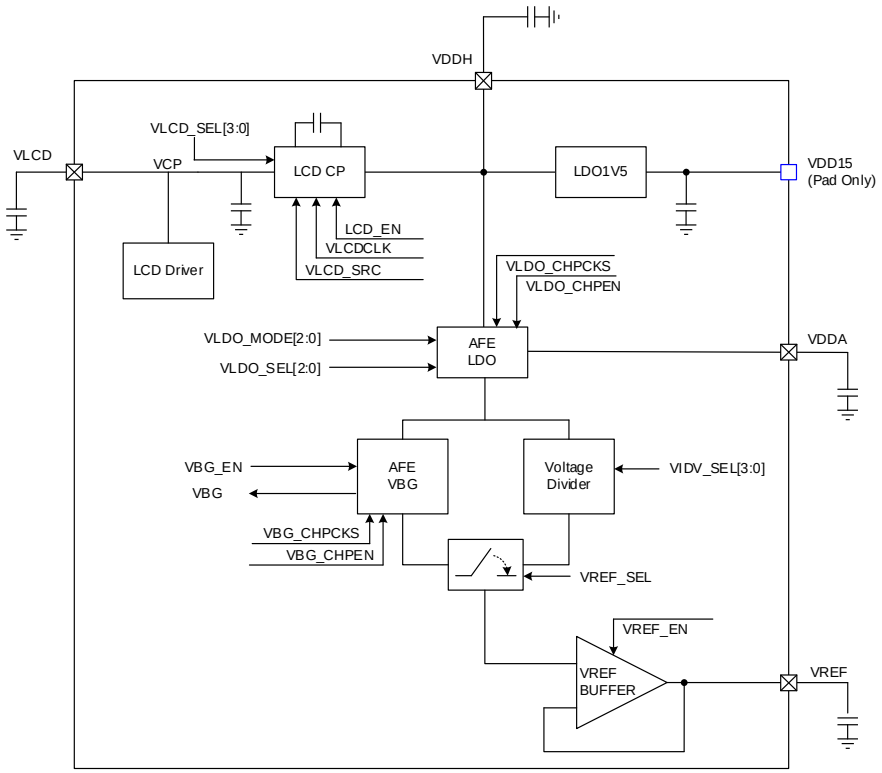


图 15-1 模拟电压源结构图

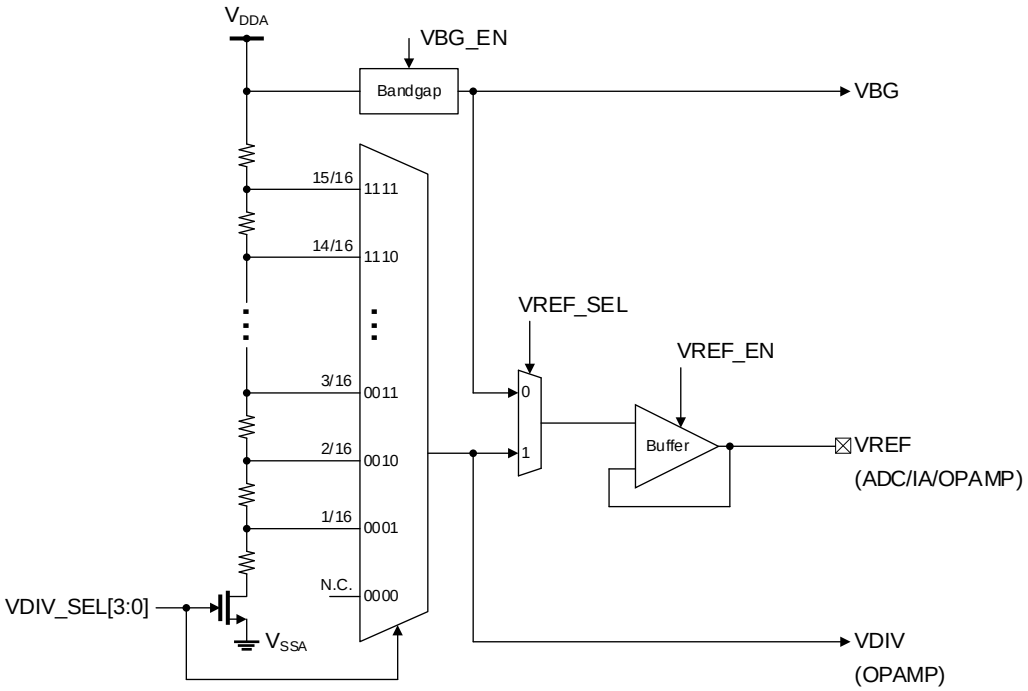


图 15-2　VREF 电压结构图

15. 4 功能描述

15. 4. 1 VDDA 电压

芯片带有一个转为模拟电路使用的稳压电路 LDO:VDDA，通过设定寄存器 **PWR_CTRL**.VLDO_MODE 设置不同的操作模式和不同的输出电压。

有四个不同的操作模式：

- 高阻态(Floating)，可从外部灌入电压给 VDDA，但是外部灌入的电压应该不可超过 VDDH。VDDH 短路(Bypass)，此时 VDDA 接近 VDDHV。
- 接地放电(Weak Pull Down)，此时 VDDA 输出接近 VSS 电位。
- 可调节稳压模式 LDO，此模式 VDDA 输出五个不同的电压：2.2V、2.4V、2.7V、3.0V 以及 3.3V。

15. 4. 2 参考电压源(VREF)

参考电压源 VREF，提供模拟/数字转换器(SD-ADC)、仪表放大器(IA)与轨对轨运算放大器(OPAMP)使用。此模块提供四种电源输出，种类如下：

电压可选择：

- 外部电源。
- 电阻分压电源(VDIV)：提供 15 阶 VDDA 分压选择。
- 带隙电压(VBG)：由 VDDA 产生出的带隙电压 1.2V。
- 参考电压(VREF)：可配置选择两种电压输出至 VREF 引脚，电阻分压电源(VDIV)或带隙电压(VBG)，具备+/-1mA 推拉驱动能力。

参考电压的输出脚位为 PA06，需要设该脚位的复用功能为 VREF。(详细的复用功能选择请参考 datasheet)

15. 4. 3 斩波器(Chopper)

模拟稳压电路以及带隙电路皆有展波器的功能，分别由寄存器开关 **PWR_CTRL**.VBG_CHPEN 以及 **PWR_CTRL**.VLDO_CHPEN 进行控制。当要使用斩波器功能，需要提供斩波器时钟正确的频率。可以通过 **RCU_CFG2**.CHOP_CKCFG 以及 **RCU_CFG2**.CHOP_CKDIV 的配置，提供斩波器时钟 4MHz。

15. 4. 4 电压开关流程

应用中如果需要开启模拟电路，电压的设置需要依照下列步骤开启寄存器

- 首先设定寄存器 **PWR_CTRL**.BIAS_EN，电路所需要的电流开关。
- 等待开启电流的稳定时间，需要 10 us。
- 接着设定 **PWR_CTRL**.VLDO_MODE，决定 VDDA 的电压模式。
- 如果设定了内部的可调节稳压模式，设定 **PWR_CTRL**.VLDO_SEL，决定 VDDA 的电压数值。
- 等待 VDDA 的稳定时间，需要 200 us。
- 最后 VDDA 的稳定时间等待结束，便可以开始设定其他 AFE 电路的寄存器。

15. 5 特殊功能寄存器

15. 5. 1 寄存器列表

ANPWR 寄存器列表			
名称	偏移地址	类型	描述
PWR_CTRL	0000_H	R/W	模拟电压源配置寄存器

15. 5. 2 寄存器描述

15. 5. 2. 1 模拟电源方案寄存器(PWR_CTRL)

模拟电源方案寄存器 (PWR_CTRL)

偏移地址:0x00

复位值:0x0008 0030

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
—	—	—	—	—	—	—	—	—	—	—	—	VDIV_SEL<3:0>				—	—	VREF_EN	VREF_SEL	TSFLTEN	VBG_CHPCKS	VBG_CHPEN	VBG_EN	BIAS_EN	VLDO_SEL<2:0>			VLDO_CHPCKS	VLDO_CHPEN	VLDO_MODE<1:0>	

—	Bit 31-20	—	—
VDIV_SEL	Bit 19-16	R/W	**VDIV分压选择** 0000:禁用 0001: VDDA*(1/16) 0010: VDDA*(2/16) 0011: VDDA*(3/16) 0100: VDDA*(4/16) 0101: VDDA*(5/16) 0110: VDDA*(6/16) 0111: VDDA*(7/16) 1000: VDDA*(8/16) 1001: VDDA*(9/16) 1010: VDDA*(10/16) 1011: VDDA*(11/16) 1100: VDDA*(12/16) 1101: VDDA*(13/16) 1110: VDDA*(14/16) 1111: VDDA*(15/16)
—	Bit 15-14	—	—
VREF_EN	Bit 13	R/W	**VREF开关**

			0: 禁用 1: 启用
VREF_SEL	Bit 12	R/W	**VREF选择** 0: VBG 1: VDIV
TSFLTEN	Bit 11	R/W	**温度感测脚位RC滤波器开关** 0: 禁用 1: 启用 注:当TSP/TSN信号须经由仪表放大器(IA)，再连接到ADC通道时，方可开启此功能。 (IA_CTRL.IA_INNS=0111/1000, IA_CTRL.IA_INPS=0111/1000)
VBG_CHPCKS	Bit 10	R/W	**VBG斩波器时钟选择** 0: 1 MHz 1: 2 MHz 注:建议配置为 2 MHz。
VBG_CHPEN	Bit 9	R/W	**VBG斩波器开关** 0: 禁用 1: 启用
VBG_EN	Bit 8	R/W	**VBG开关** 0: 禁用 1: 启用
BIAS_EN	Bit 7	R/W	**Bias开关** 0: 禁用 1: 启用
VLDO_SEL	Bit 6-4	R/W	**VLDO可调节电压选择** 000: 2.2V 001: 2.4V 010: 2.7V 011: 3.0V 1XX: 3.3V
VLDO_CHPCKS	Bit 3	R/W	**VLDO斩波器时钟选择** 0: 1 MHz 1: 2 MHz 注:建议配置为 2 MHz。
VLDO_CHPEN	Bit 2	R/W	**VLDO 斩波器开关** 0: 禁用 1: 启用

VLDO_MODE	Bit 1-0	R/W	**VLDO 模式选择** 00: 高阻态(Floating) 01: 短路 VDDH(Bypass) 10: 接地放电 (Weakly Pull-down) 11: 可调节电压模式

第16章 通用定时器 32 位 4 通道(GP32C4T)

16. 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。

16. 2 特性

- 三种 32 位元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
- 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:
  - 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
- 支持增量(正交)编码
- 外部时钟输入触发计数器

16. 3结构图

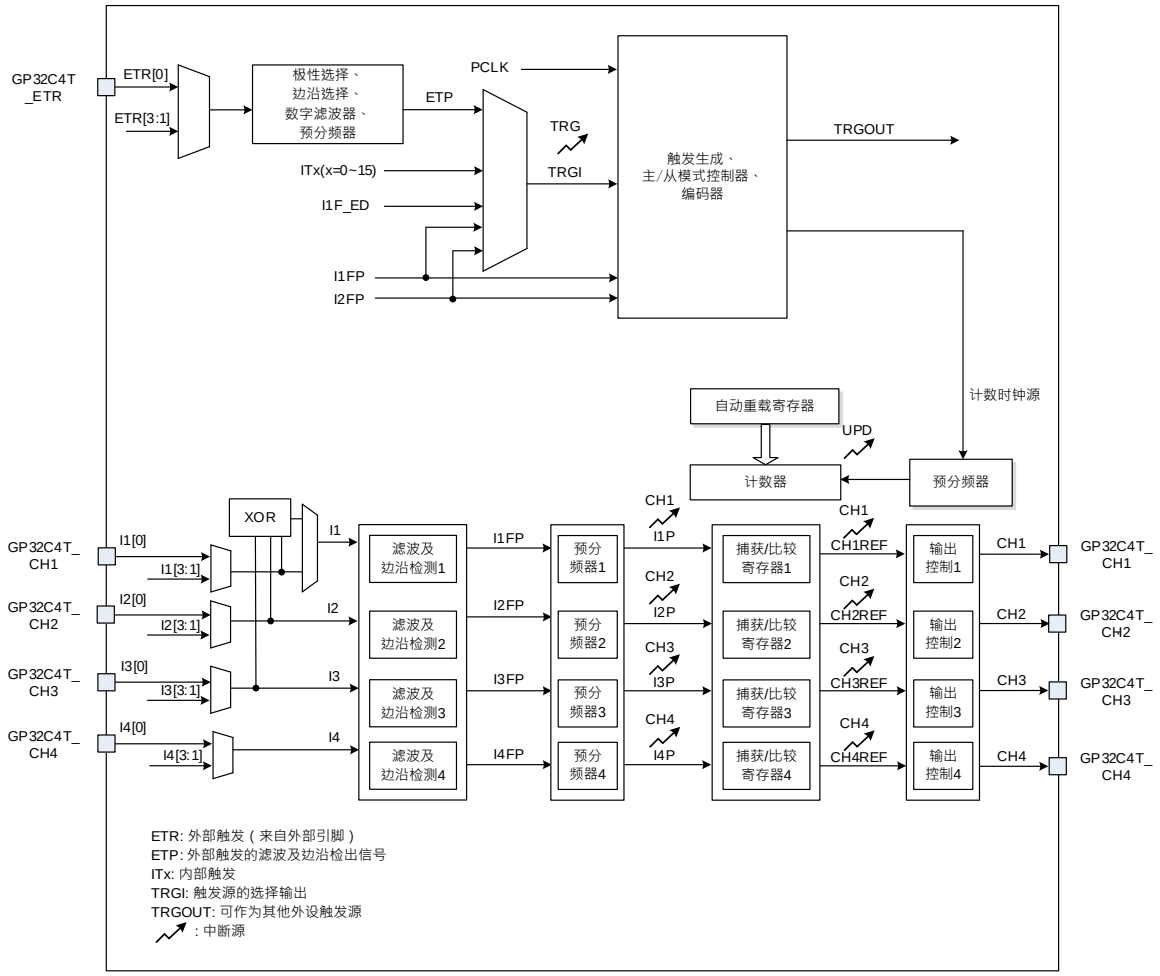


图 16-1　GP32C4T 定时器结构框图

16. 4 功能描述

16. 4. 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** 的重载值会被立即反应到影子寄存器中；而设置 ARPEN 位为 1 时，**GP32C4T_AR** 寄存器具有缓冲功能，当产生更新事件(UPD)时，**GP32C4T_AR** 寄存器的重载值才会被更新到影子寄存器中。

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 APB 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。

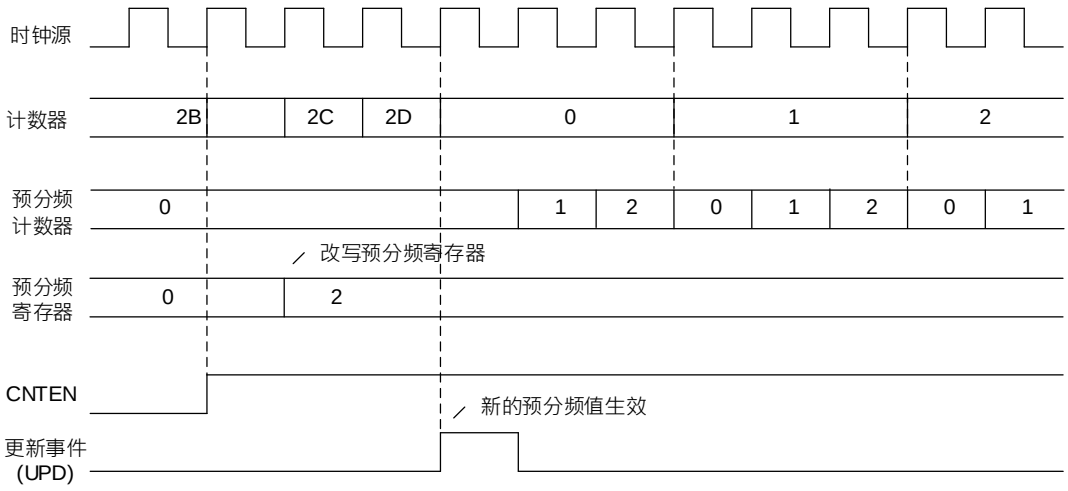


图 16-2 预分频值计数时序图

16. 4. 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(I1、I2)、外部时钟源 2(ETR)与内部触发输入(IT0~IT15)。

16. 4. 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 000b)，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。

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。

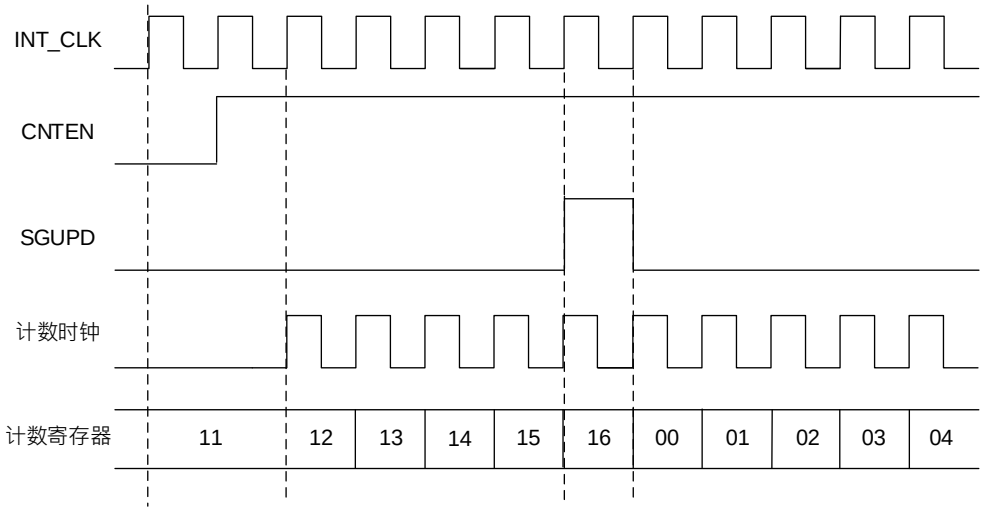


图 16-3 采用内部时钟计数

16. 4. 2. 2 外部时钟源 1

GP32C4T_SMCON 寄存器的 SMODS 位為 111b 时，可选择外部时钟源 1。计数器可根据选定的上升沿或下降沿计数。

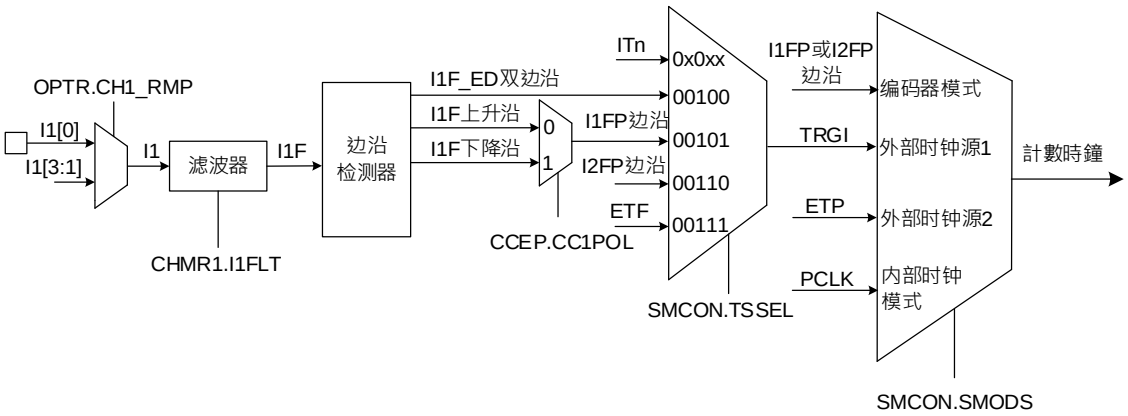


图 16-4 外部时钟连接

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_OPTR** 寄存器的 CH1_RMP 位为 00b，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 0000)。
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 TSSEL1 位与 TSSEL2 位为 00101b，选择外部时钟源为 I1。
6. 设置 **GP32C4T_SMCON** 寄存器的 SMODS 位为 111b，选择定时器外部时钟模式 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。

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时，取决于 I1 输入的同步电路。

16. 4. 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。

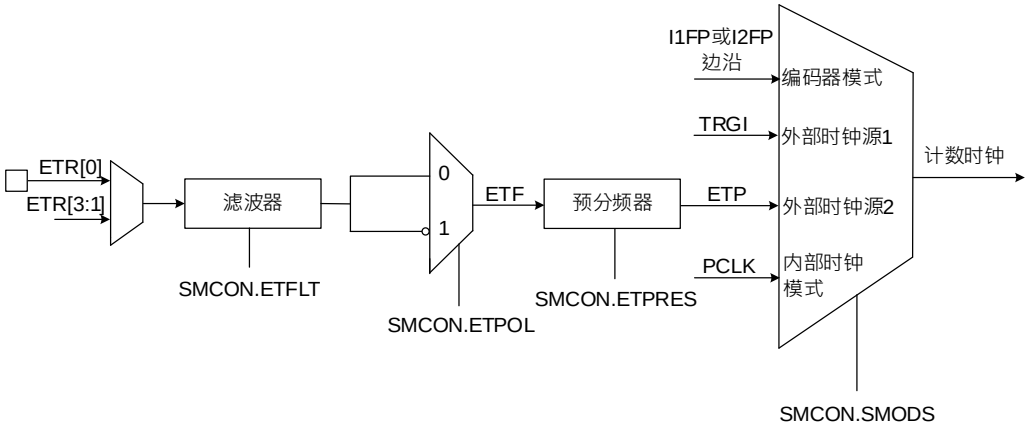


图 16-5 外部触发输入模块

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 0000)。
2. 设置 **GP32C4T_SMCON** 寄存器的 ETPOL 位，检测 ETR 引脚上升沿或下降沿。
3. 设置 **GP32C4T_SMCON** 寄存器的 ECM2EN 位為 1，开启外部时钟模式 2。
4. 设置 **GP32C4T_CON1** 寄存器的 CNTEN 位為 1，开启计数器。

计数器在每个 ETR 上升沿计数一次。

16. 4. 2. 4　内部触发输入(ITn)

设置 **GP32C4T_SMCON** 寄存器的 SMODS 位为 111b，外部时钟模式 1。计数器根据选定的内部输入端(ITn)的上升沿计数。

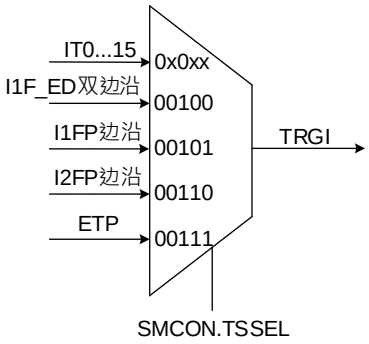


图 16-6　ITn 内部时钟连接

设置计数器外部时钟源为 ITn 输入，并在 ITn 上升沿时计数，步骤如下:

1. 设置 **GP32C4T_SMCON** 寄存器的 TSSEL1 位与 TSSEL2 位,选定 ITn 作为外部时钟源。
2. 设置 **GP32C4T_SMCON** 寄存器的 SMODS 位为 111b，设置外部时钟模式 1。
3. 设置 **GP32C4T_CON1** 寄存器的 CNTEN 位为 1，开启计数器。

ITn 产生上升沿时，计数器计数一次。

16. 4. 3　计数模式

16. 4. 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。

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)更新影子寄存器。在设置 DISUE 位为 0 之前都不会产生更新事件(UPD)，计数器和预分频器都会重新从 0 开始计数。

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。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影子寄存器:

- 更新 **GP32C4T_AR** 寄存器数值到影子寄存器
- 更新 **GP32C4T_PRES** 寄存器数值到影子寄存器

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。

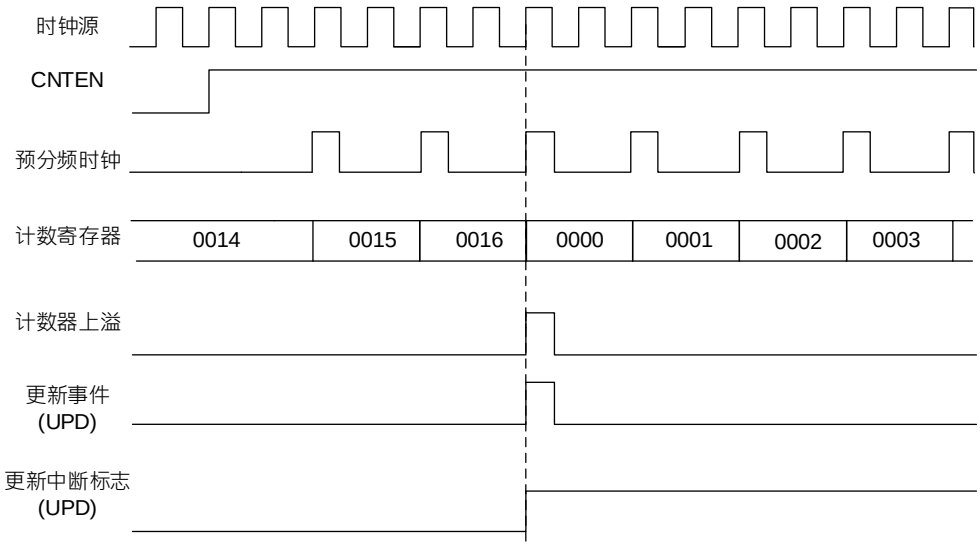


图 16-7　计数器递增计数时序图

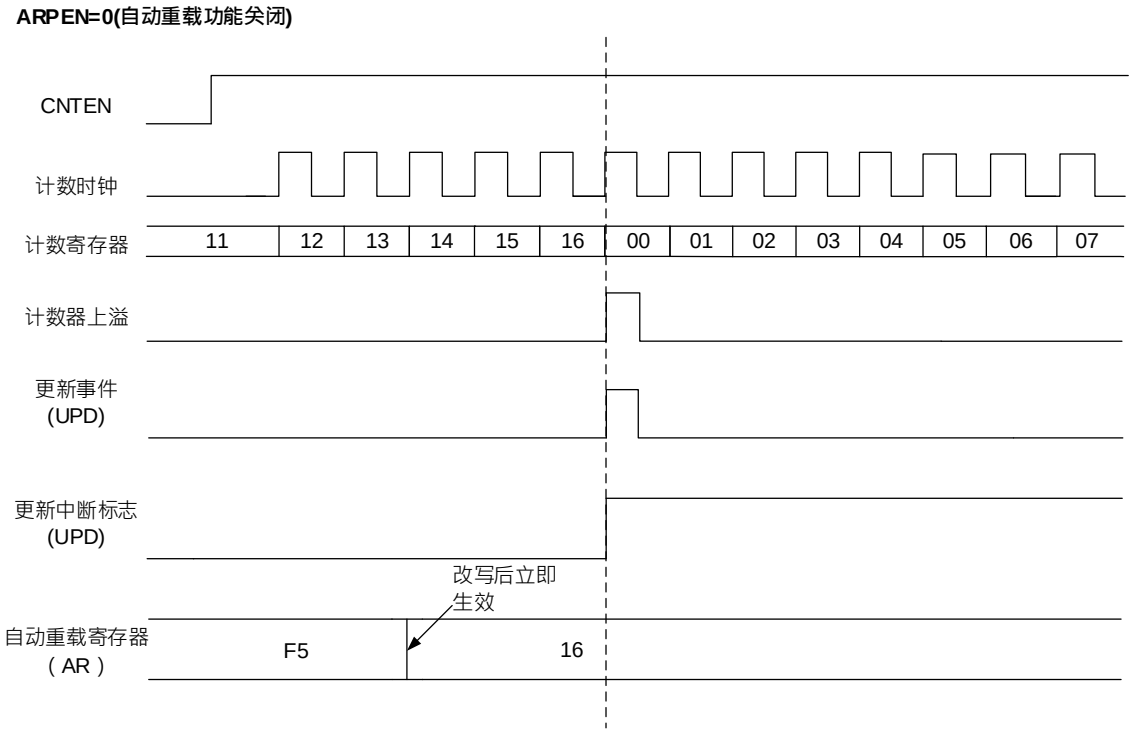


图 16-8　设置 ARPEN 位为 0 时计数器时序图

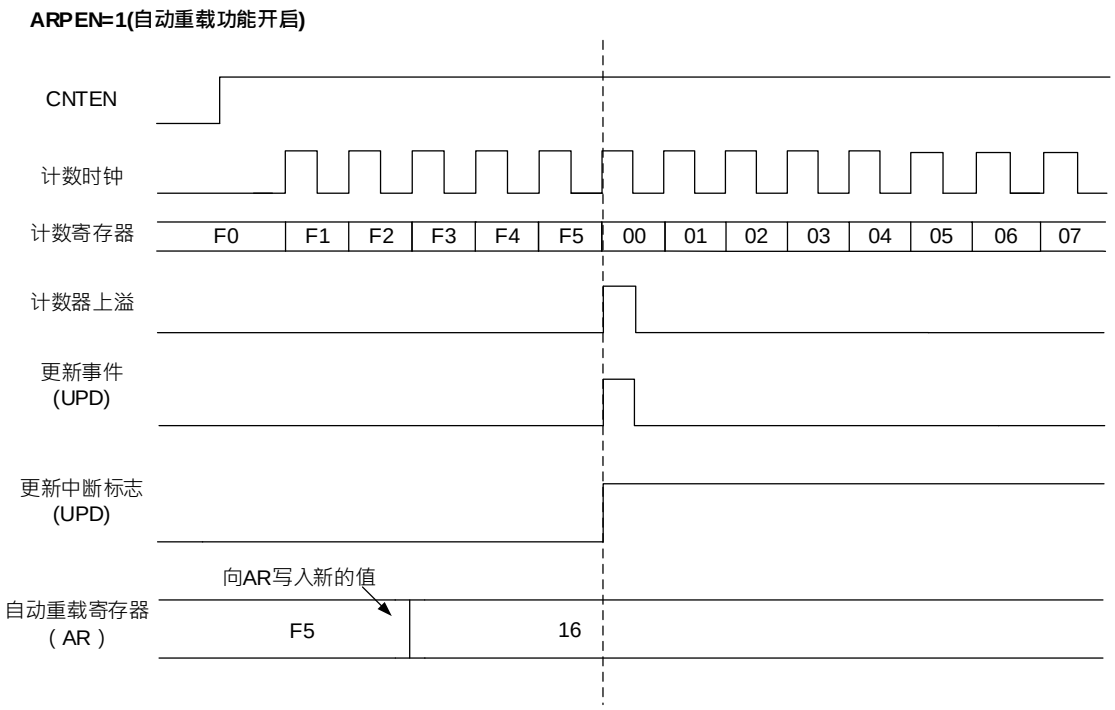


图 16-9　设置 ARPEN 位为 1 时计数器时序图

16. 4. 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。

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)更新影子寄存器。在设置 DISUE 位为 0 之前都不会产生更新事件(UPD)，计数器会重新从当前自动重载值开始计数，而预分频器从 0 开始计数。

此外，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。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影子寄存器。

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。

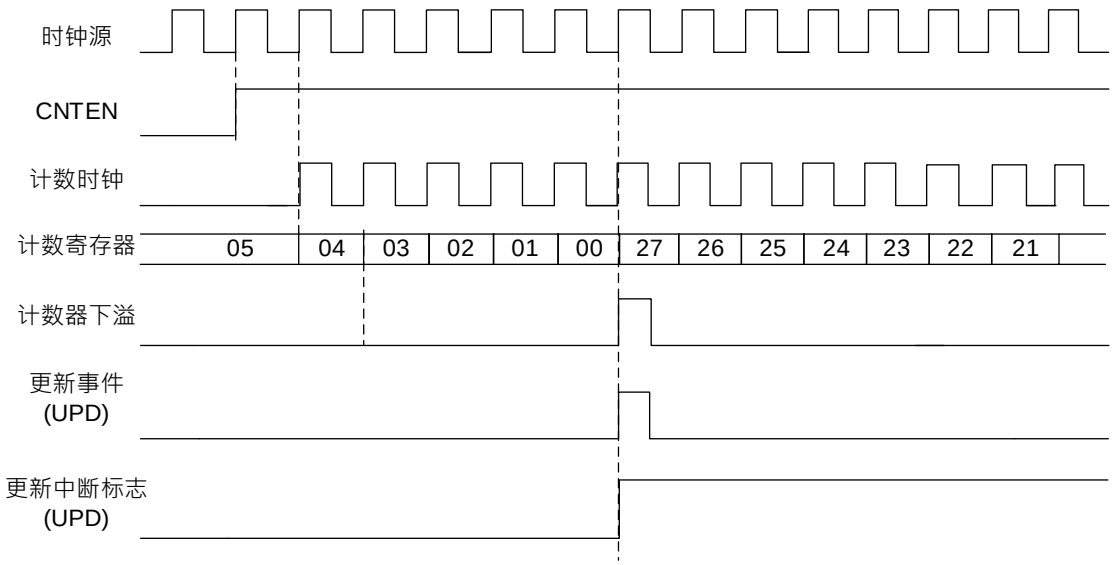


图 16-10　计数器递减计数时序图

16. 4. 3. 3　　中心对齐模式

设置 **GP32C4T_CON1** 寄存器的 CMSEL 位数值不等于 00 时，定时器工作在中心对齐模式。定时器设置为中心对齐模式时，计数器先从 0 开始递增至 **GP32C4T_AR** 寄存器数值减 1，并产生更新事件(UPD)；接着计数器从 **GP32C4T_AR** 寄存器数值递减至 1，并产生更新事件，之后从 0 开始重新计数，因此方式循环计数。将信道配置为输出模式时，当计数器递减计数(中心对称模式 1，设置 CMSEL 位为 01)、计数器递增计数(中心对称模式 2，设置 CMSEL 位为 10)、计数器递增和递减计数(中心对称模式 3，设置 CMSEL 位为 11)时，其将设置输出比较中断标志位为 1。

在中心对齐模式下，**GP32C4T_CON1** 寄存器的 DIRSEL 位无法进行写操作，该位由硬件自动更新指示当前计数方向。

计数上溢、下溢或者设置 **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。设置 **GP32C4T_CON1** 寄存器的 DIRSEL 位为 0 时，计数器由 0 开始递增。设置 **GP32C4T_CON1** 寄存器的 DIRSEL 位为 1 时计数器由 **GP32C4T_AR** 寄存器数值开始递减，而预分频器都是从 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)更新影子寄存器。在设置 DISUE 位为 0 之前都不会产生更新事件(UPD)，在正常产生更新事件时，计数器和预分频器都会重新从 0 开始计数。

此外，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。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影子寄存器。

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

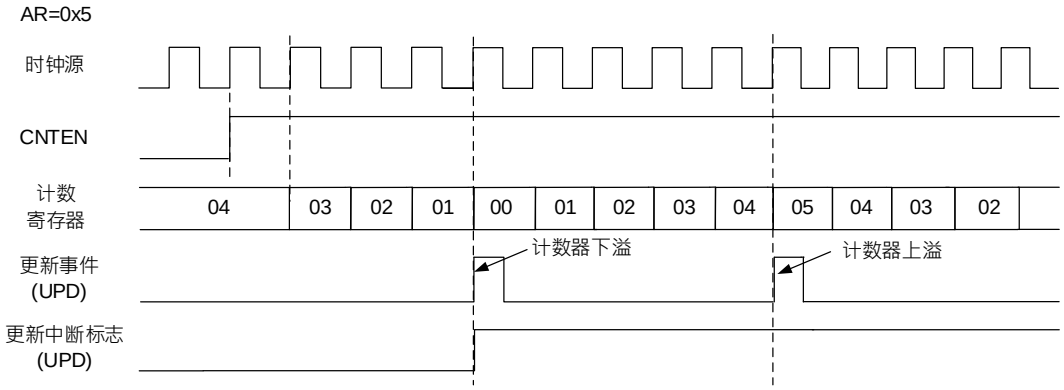


图 16-11　计数器递增减计数时序图

16. 4. 4 捕获或比较通道

输入电路对 In 输入端的信号进行采样，产生一个经过滤波的信号 InF。之后一个可极性选择的边沿检测器产生 In 边沿检测信号，该信号可作为从模式控制器的触发输入或作为捕获控制命令，且该信号经过分频后进入捕获寄存器。

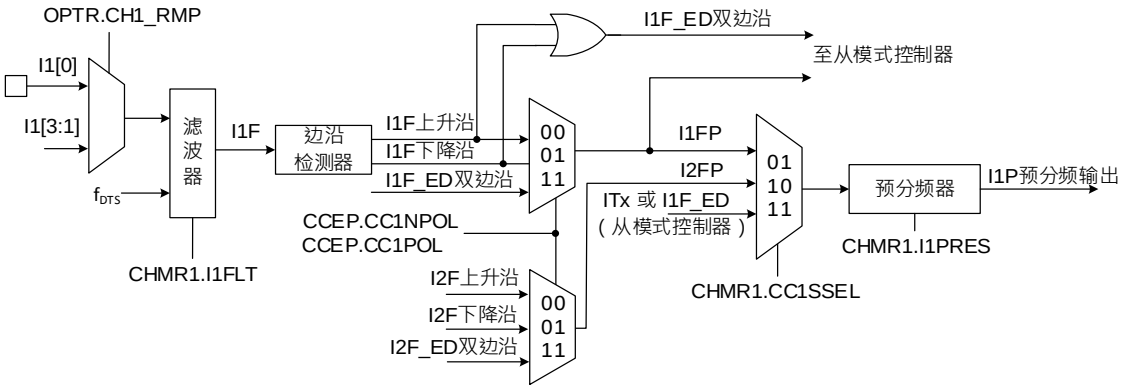


图 16-12 捕获或比较通道

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

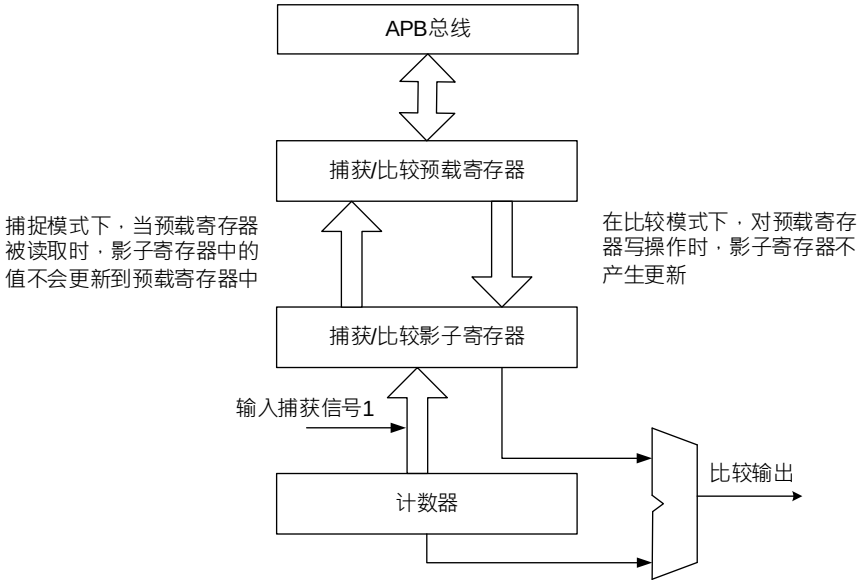


图 16-13 捕获或比较通道结构图

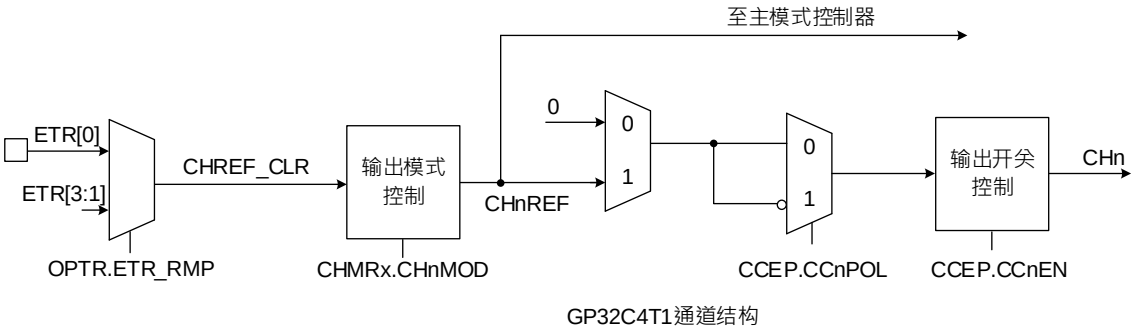


图 16-14　捕获或比较通道的输出部分

16. 4. 5　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。当捕获发生时，**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(如果有开启)。

当 **GP32C4T_RIF** 寄存器中相应的 CHn 标志位已经为 1，又发生捕获事件时，**GP32C4T_RIF** 寄存器中相应的过捕获 CHnOV 标志位也会被设定为 1，表示发生过捕获事件。

通过软件设置 **GP32C4T_ICR** 寄存器的 CHn 位与 CHnOV 位为 1，清除 **GP32C4T_RIF** 寄存器中 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，连续 8 次采样才确认电平变化有效。
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. 设置 **GP32C4T_CCEP** 寄存器的 CC1EN 位为 1，开启捕获计数器。
6. 如有需要，设置 **GP32C4T_IER** 寄存器的 CH1 位为 1，开启中断请求。

当发生输入捕获时:

1. 有效边沿产生，**GP32C4T_CCVAL1** 寄存器获取计数器数值。
2. 硬件自动设置 CH1 标志位为 1(中断标志位)。若至少 2 个连续的捕获发生，但标志位没有及时清除，则会设置 CH1OV 位为 1。
3. 中断的产生取决于 **GP32C4T_IER** 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前先读取捕获数据。避免丢失在读过捕获标志位到读捕获数据之间的重复捕获信息。

注:捕获中断请求可由软件设置 GP32C4T_SGE 寄存器的 SGCHn 位产生。

16. 4. 6 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_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，连续 8 次采样才确认电平变化有效。
3. 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** 寄存器和计数器清零。
4. 设置 **GP32C4T_CHMR1** 寄存器的 CC2SSEL 位为 10b，通道 2 选择 I1 为有效输入端。
5. 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。
6. 设置 **GP32C4T_SMCON** 寄存器的 TSSEL1 位与 TSSEL2 位为 00101b，选择 I1 滤波后信号为有效的触发输入。
7. 设置 **GP32C4T_SMCON** 寄存器的 SMODS 位为 100b，选择从模式控制器为复位模式。
8. 设置 **GP32C4T_CCEP** 寄存器的 CC1EN 位和 CC2EN 位为 1，开启捕获。

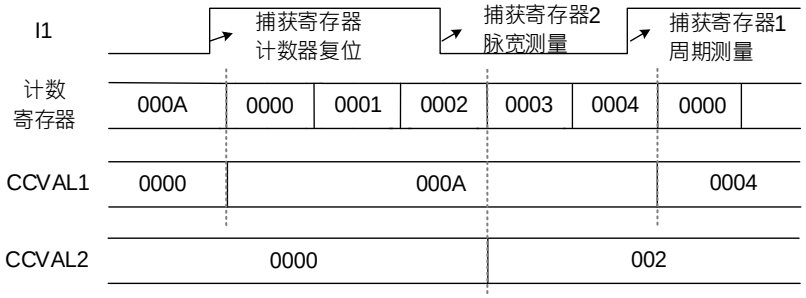


图 16-15 PWM 输入模式时序

- **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 = (GP32C4T_PRES + 1) * (GP32C4T_CCVALn + 1) / f_{INT_CLK}

16. 4. 7 PWM输出模式

脉宽调制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。

每个个信道的 PWM 模式是相互独立的(每个 CHn 输出一个 PWM)，只需设置 **GP32C4T_CHMRn** 寄存器的 CHnMOD 位为 110(PWM 模式 1)或为 111(PWM 模式 2)。

可通过设置 **GP32C4T_CHMRn** 寄存器的 CHnPEN 位为 1 来开启相应的预装载寄存器，及设置 **GP32C4T_CON1** 寄存器的 ARPEN 位为 1 来开启自动重载功能。

开启预装载、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写入到影子寄存器中，因此在开启计数前，必须通过设置 **GP32C4T_SGE** 寄存器的 SGUPD 位为 1 来初始化所有的寄存器。

CHn 的极性可通过 **GP32C4T_CCEP** 寄存器的 CCnPOL 位设置，有效电平可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波形是边沿对齐或中心对齐，取决于GP32C4T_CON1寄存器的CMSEL位。

16. 4. 7. 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。

下图以 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 110b，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，当计数器上数到 8 后重载。
5. 设置 **GP32C4T_CON1** 寄存器的 CNTEN 位为 1，开启计数器。

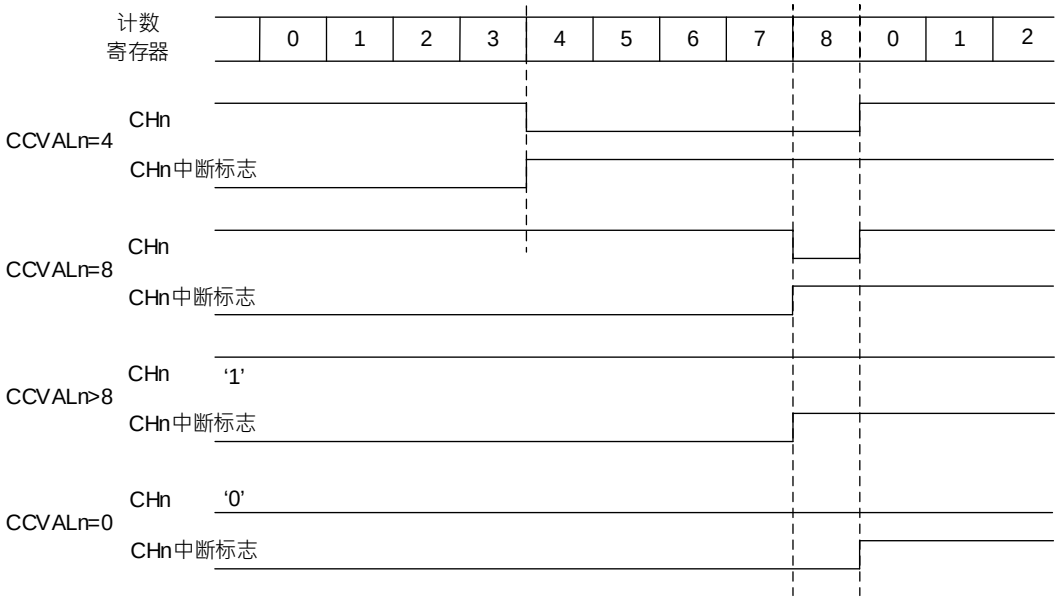


图 16-16 边沿对齐递增计数 PWM 波形(AR=8)

- **GP32C4T_COUNT< GP32C4T_CCVAL1** 时，CH1 为高电平。
- **GP32C4T_COUNT>= GP32C4T_CCVAL1** 时，CH1 为低电平。

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；若 **GP32C4T_COUNT** = 0 时，CH1 会永远输出低电平。

◆ 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

下图以 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，当计数器下数到 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 110b，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。

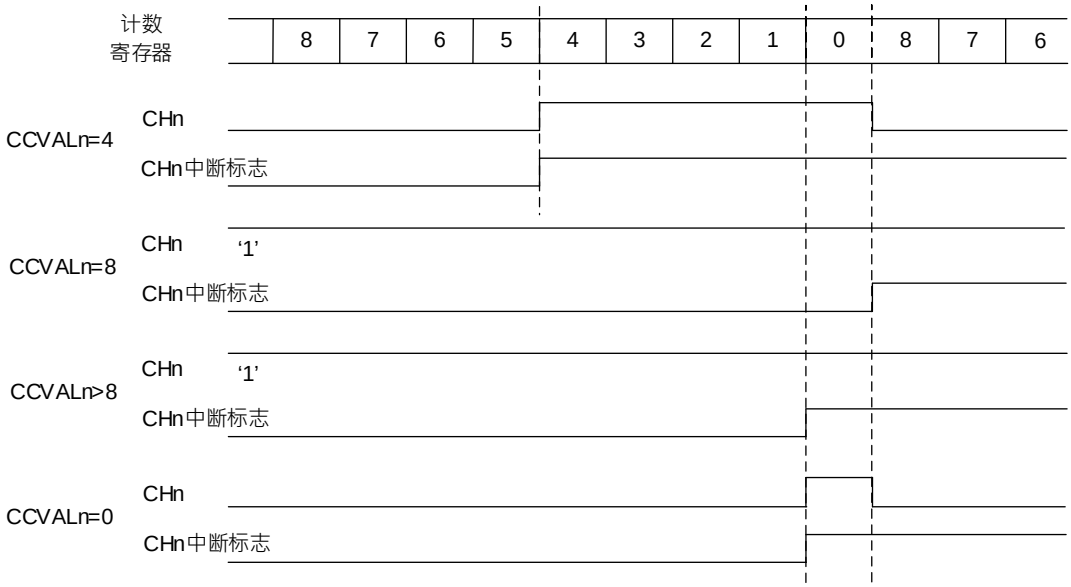


图 16-17　边沿对齐递减计数 PWM 波形(AR=8)

◆ **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。此模式下不可能产生 0%的 PWM 波形。

16. 4. 7. 2 PWM中心对齐模式

设置 **GP32C4T_CON1** 寄存器的 CMSEL 位不为 00 时，中心对齐模式有效。根据 CMSEL 位的设置，计数器可以在递增、递减计数分别设置 **GP32C4T_RIF** 寄存器的比较标志位为 1 或是在递增递减设置比较标志位为 1。**GP32C4T_CON1** 寄存器的 DIRSEL 位控制计数方向由硬件更新，软件无法修改。

下图为中心对齐模式 2 下，CH1 输出 PWM 模式为例，相关配置流程如下:

1. 设置 **GP32C4T_CON1** 寄存器的 CMSEL 位为 10b，选择中心对齐模式 2，计数器只有在递增计数时才会设置 **GP32C4T_RIF** 寄存器的比较匹配标志位为 1。
2. 设置 **GP32C4T_AR** 寄存器的 ARV 位为 3Fh，当计数器下数到 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 110b，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 3Dh
7. 设置 **GP32C4T_CON1** 寄存器的 CNTEN 位为 1，开启计数器。

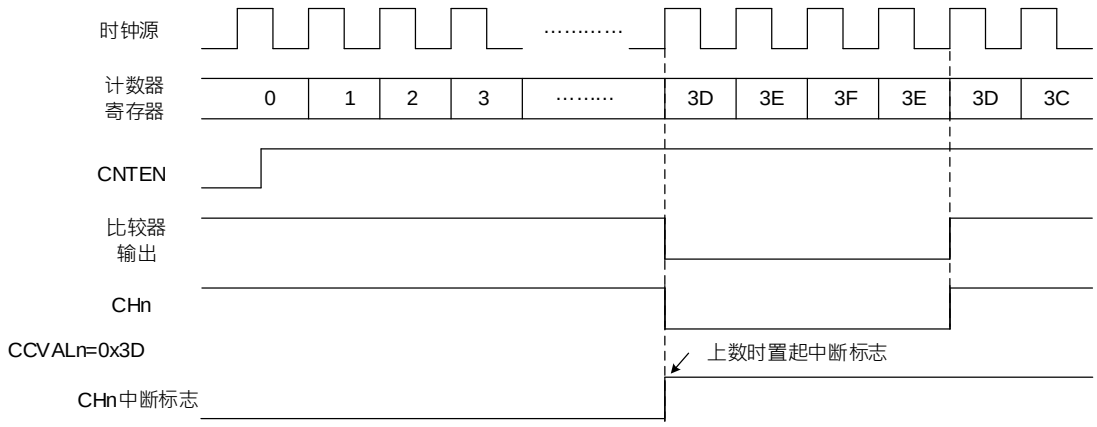


图 16-18 中心对齐 PWM 波形(AR 位为 3Fh，CCRV 位为 3Dh)

当计数器递增计数到 3Dh 时，PWM 输出低电平，同时设置 **GP32C4T_RIF** 寄存器的比较匹配标志位；递减计数到 3Dh 时，PWM 输出回到高电平。

中心对齐模式的使用技巧:

- 当进入中心对齐模式后，当前递增或递减设置生效。**GP32C4T_COUNT** 递增或递减计数取决于 **GP32C4T_CON1** 寄存器的 DIRSEL 位数值。此外，软件不得对 DIRSEL 和 CMSEL 位同时进行修改。
- 计数器在中心对齐模式下运行时，对 **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)且在计数器运行过程中不对 **GP32C4T_COUNT** 寄存器写值。

16. 4. 8 输出比较模式

该功能用于控制输出波形或指示周期时间的结束。

当捕获或比较寄存器和 **GP32C4T_COUNT** 寄存器数值匹配时，输出比较功能:

- 设置 **GP32C4T_CHMRn** 寄存器的 CHnMOD 位选择输出模式，输出极性由 **GP32C4T_CCEP** 寄存器的 CCnPOL 位控制 :
  - ◇ 设置 CHnMOD 位为 000b:当计数器匹配比较器时输出保持其电平。
  - ◇ 设置 CHnMOD 位为 001b: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 010b: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 011b: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。

设置 **GP32C4T_CHMRn** 寄存器的 CHnPEN 位数值可决定 **GP32C4T_CCVALn** 寄存器是否带有预装载寄存器。

在输出比较模式中，更新事件 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 011，当 CNTV 与 CCRVALn 匹配时，CHn 输出翻转。
   - 设置 CHnPEN 位为 0，关闭预装载寄存器。
   - 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影子寄存器。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。

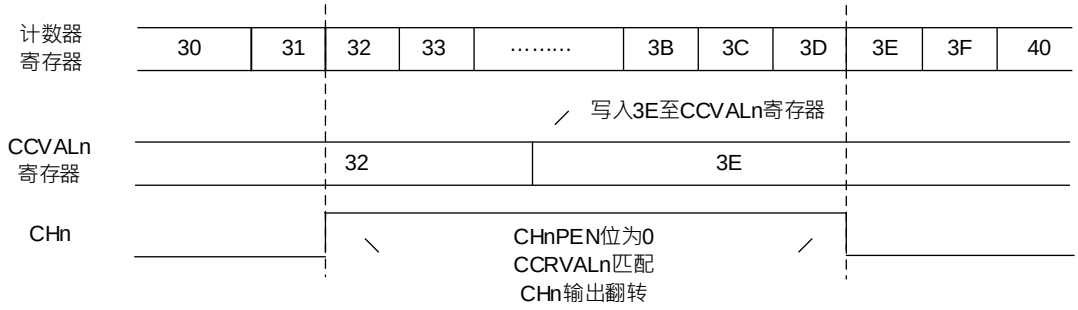


图 16-19　输出比较模式，触发 CHn

16. 4. 8. 1　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 ETR 输入端为高电平时，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。

选择 ETR 时，ETR 配置如下:

1. 设置 **GP32C4T_SMCON** 寄存器中的 ETPRES 位为 00b，关闭外部触发预分频器。
2. 设置 **GP32C4T_SMCON** 寄存器的 ECM2EN 位为 0，关闭外部时钟源 2。
3. 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。

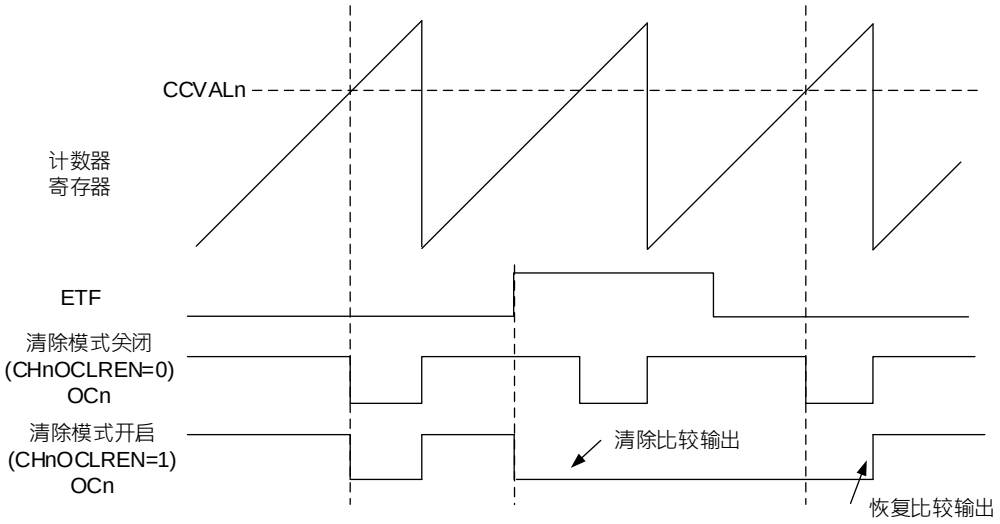


图 16-20　清除比较输出 CHn

16. 4. 8. 2 强制输出模式

设置 **GP32C4T_CHMRn** 寄存器的 CCnSSEL 位为 00b 开启输出模式，在此模式下通过软件设置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 101b，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 100b 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。

16. 4. 9 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配置的波形。

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** 寄存器和 **GP32C4T_COUNT** 寄存器数值不同时，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

基于 PWM 模式设置单脉冲输出波形的步骤如下:

1. 设置 **GP32C4T_CHMRn** 寄存器的 CHnMOD 位，选择 PWM 模式 1 或 2。
2. 设置 **GP32C4T_CCEP** 寄存器的 CCnPOL 位，选择通道 CHn 的输出极性。
3. 设置 **GP32C4T_CON1** 寄存器的 DIRSEL，选择计数器为递增或递减计数。
4. 设置 **GP32C4T_CON1** 寄存器的 SPMEN 位为 1，开启单脉冲模式。
5. 设置 **GP32C4T_CHMR1** 寄存器的 CH1PEN 位为 1，**GP32C4T_CON1** 寄存器的 ARPEN 位为 1，开启比较寄存器和计数重载寄存器的缓冲功能(也可以根据实际情况关闭缓冲)。
6. 设置 **GP32C4T_CCVALn** 寄存器和 **GP32C4T_AR** 寄存器，设置单脉冲输出延时和脉宽时间。
7. 设置 **GP32C4T_SGE** 寄存器的 SGUPD 位为 1 来产生一个更新事件。
8. 设置 **GP32C4T_CON1** 寄存器的 CNTEN 位为 1 来开启计数器，也可以在触发模式下，通过外部触发输入信号来触发硬件自动设置 CNTEN 位为 1。

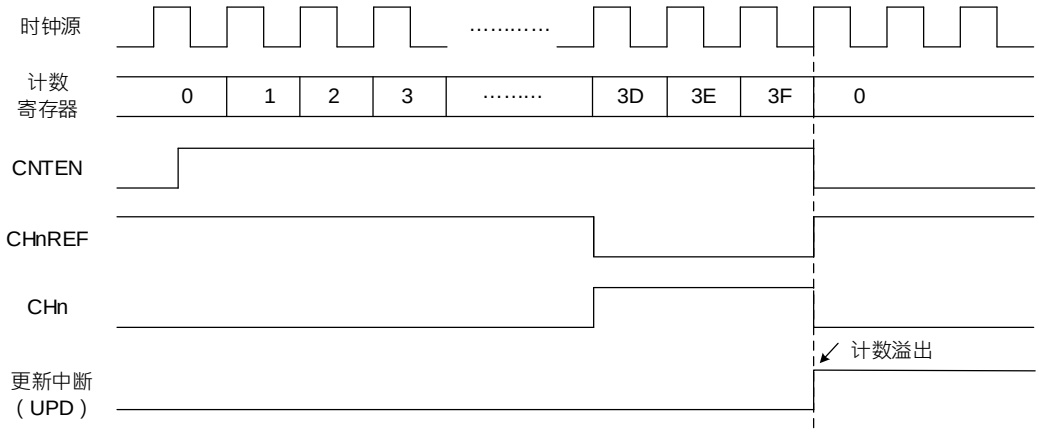


图 16-21　单脉冲模式

CHn 快速开启模式

在单脉冲模式下，In 输入的有效边沿会开启计数器(自动设置 CNTEN 位为 1)，在比较计数器数值后输出信号。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 PWM1 或 PWM2 模式时才能使用。

16. 4. 10 编码器接口模式

编码器接口模式的三种设置:若设置 **GP32C4T_SMCON** 寄存器的 SMODS 位为 001b，则计数器只根据 I2 上的边沿计数；若设置 **GP32C4T_SMCON** 寄存器的 SMODS 位为 010b，则计数器只根据 I1 上的边沿计数；若设置 **GP32C4T_SMCON** 寄存器的 SMODS 位为 011b，则计数器同时根据 I1 和 I2 上的边沿计数。

设置 **GP32C4T_CCEP** 寄存器的 CC1POL 和 CC2POL 位数值可选择 I1 和 I2 的极性。如果需要，也可以设置输入滤波器。

CH1 和 CH2 输入作为增量编码器的接口。当计数器开启时(设置 **GP32C4T_CON1** 寄存器的 CNTEN 位为 1)，计数器时钟是由 I1 或 I2 上滤波后的有效电平转换提供。I1 和 I2 滤波后的有效信号转换序列会产生计数脉冲及方向信号。计数器是递增或递减计数由信号的转换序列决定，**GP32C4T_CON1** 寄存器的 DIRSEL 位数目方向位由硬件自动更新。

编码器接口模式的工作方式类似于一个带有方向选择的外部时钟。计数器在 0 到 **GP32C4T_AR** 寄存器的自动重载值之间连续计数。因此必须在开始计数前设置 **GP32C4T_AR** 寄存器。在此模式下捕获器、预分频器、触发输出的功能皆可正常工作。设定编码模式和选择外部时钟源 2 不兼容，不可以同时选择。

该模式下，计数器会根据增量式编码器的速度和方向自动修改，计数器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。

有效边沿	有效边沿相对信号的电平 (I1 滤波信号对应 I2,I2 滤波信号对应 I1)	I1 滤波信号		I2 滤波信号	
		上升	下降	上升	下降
仅在 I1 计数	高电平	递减	递增	不计数	不计数
	低电平	递增	递减	不计数	不计数
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
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

表 16-1　计数方向与编码器信号的关系

外部增量编码器可直接与 MCU 连接，无需外部接口逻辑。而比较器通常用于将编码器的差分输出转换为数字信号，这样大幅提高抗噪声能力。编码器的第三个输出端用于指示机械零点，可以连接到外部中断输入引脚以触发一次计数复位。

下图给出了计数信号的产生和方向控制的例子。同样给出了选择双边沿时，输入抖动如何被补偿。输入抖动可能发生在传感器靠近切换点处。

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 011b，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。

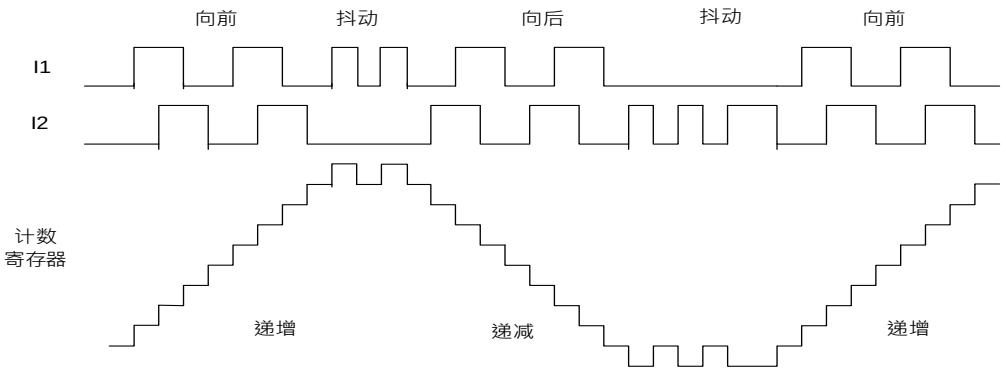


图 16-22　编码器接口模式下的计数操作

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)

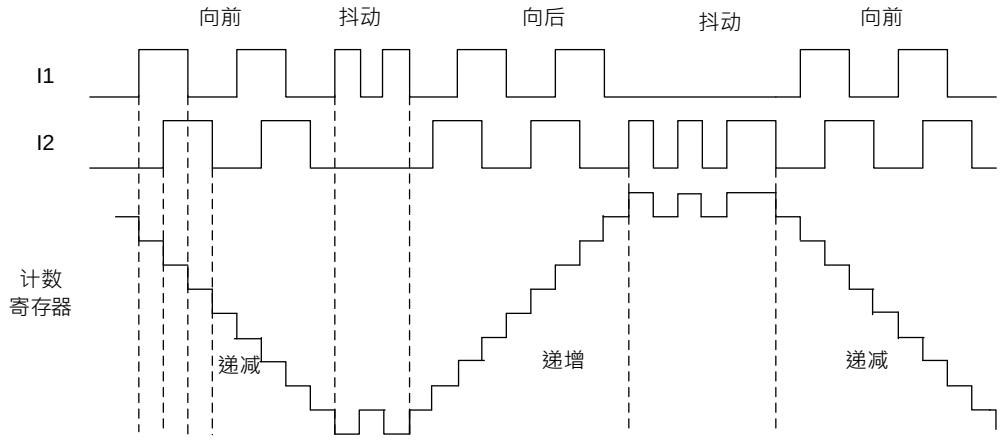


图 16-23　滤波后极性反相时编码器接口例子

当设置为编码器接口模式时，定时器可提供传感器的当前位置信息。设置另一个定时器为捕获模式，用于测量两个编码器事件的间隔，根据间隔时长获取动态信息(速度、加速度、减速度)。编码器用于指示机械零点的输出就是此用处。根据编码器两个事件间隔，可以周期性的读取计数器数值。应用上可以将计数器值锁存到第三个输入捕获寄存器(捕获信号必须是周期性的且可由另一个定时器产生)。

16. 4. 11 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。

16. 4. 12 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。

16. 4. 12. 1　复位模式

计数器及其预分频器可以在响应触发输入事件时重新初始化。此外，若 **GP32C4T_CON1** 寄存器的 UERSEL 位为 0 时会产生一次更新事件 UPD。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 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 100b，选择复位模式。
7. 设置 **GP32C4T_CON1** 寄存器的 CNTEN 位为 1，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)，如果中断开启(取决于 **GP32C4T_IER** 寄存器的 TRGI 位)，会发送中断。

下图给出了设置自动重载寄存器 **GP32C4T_AR** 为 0x36 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。

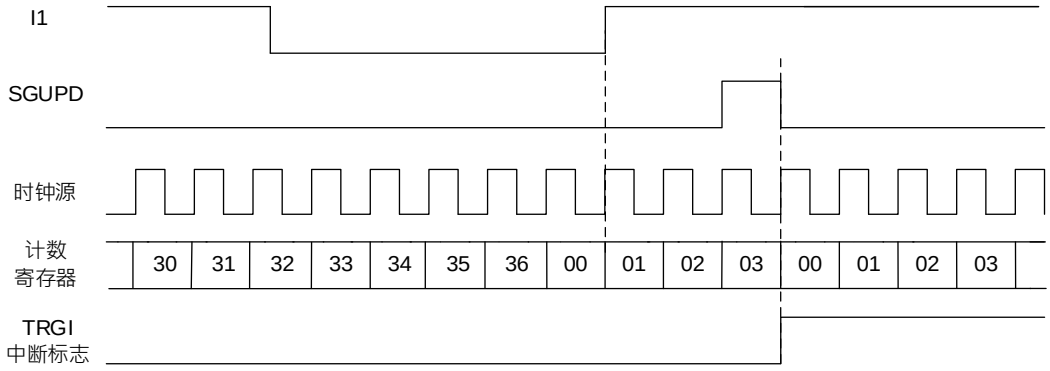


图 16-24 复位模式控制电路

16. 4. 12. 2 门控模式

计数器根据选中的输入电平被开启。

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时。

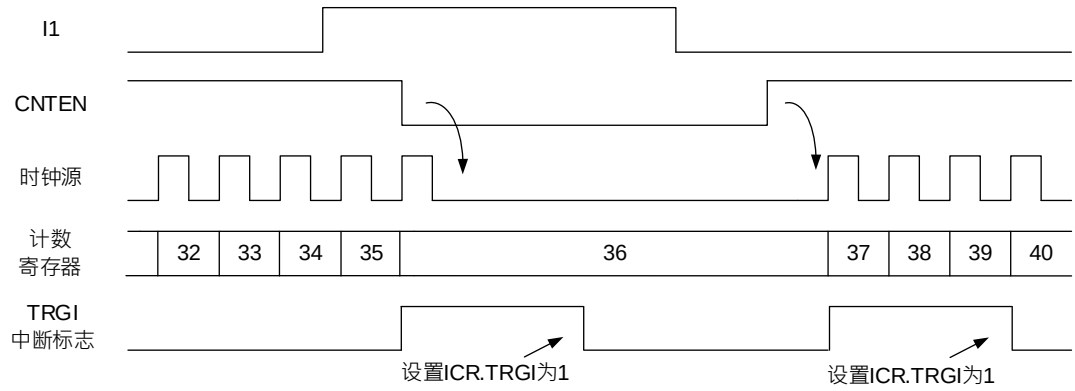


图 16-25　门控模式控制电路

16. 4. 12. 3　触发模式

输入端选中的事件可以开启计数器。

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 1，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时。

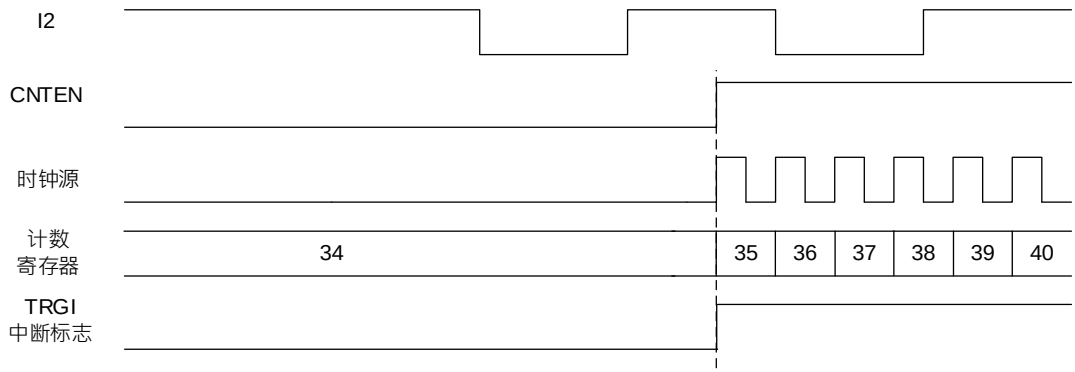


图 16-26　触发模式控制电路

16. 4. 12. 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 1，开启计数器。

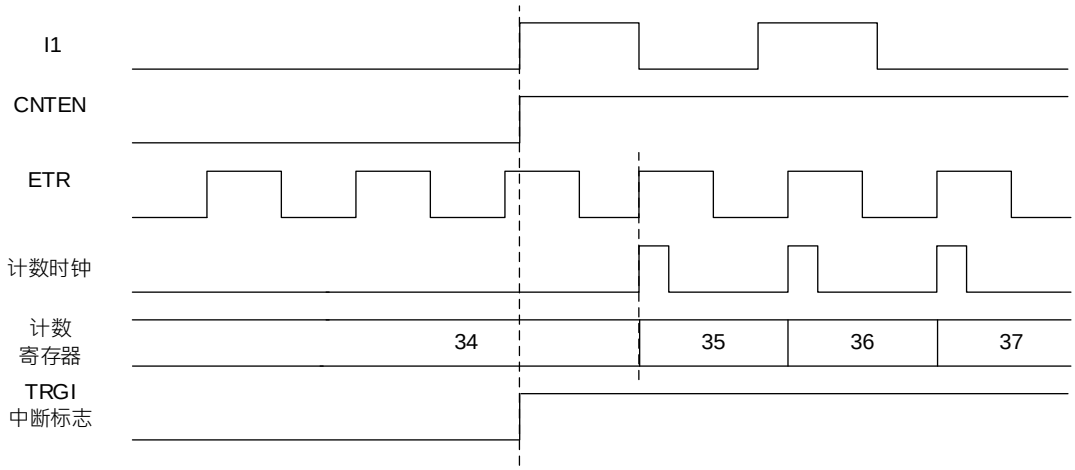


图 16-27　外部时钟源 2＋触发模式下的控制电路

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时。

16. 4. 13 定时器同步

所有定时器在内部相连，用于定时器同步或链接。当一个定时器处于主模式时，它可以对另一个处于从模式的定时器的计数器进行复位、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。

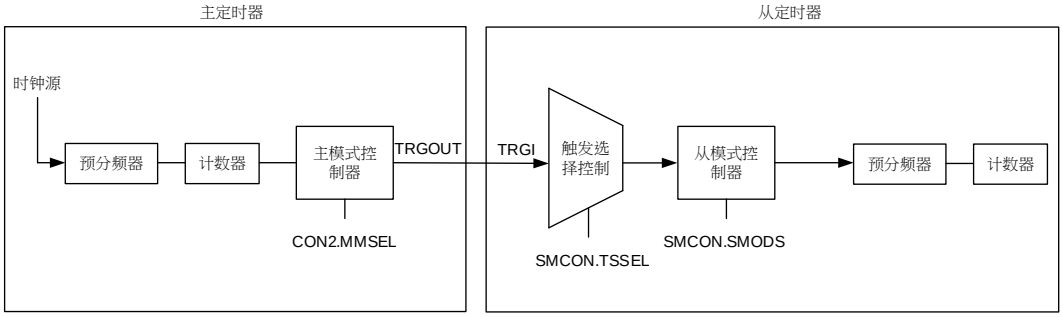


图 16-28　主/从定时器范例

16. 4. 13. 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 01001b(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，当计数器上数到 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。

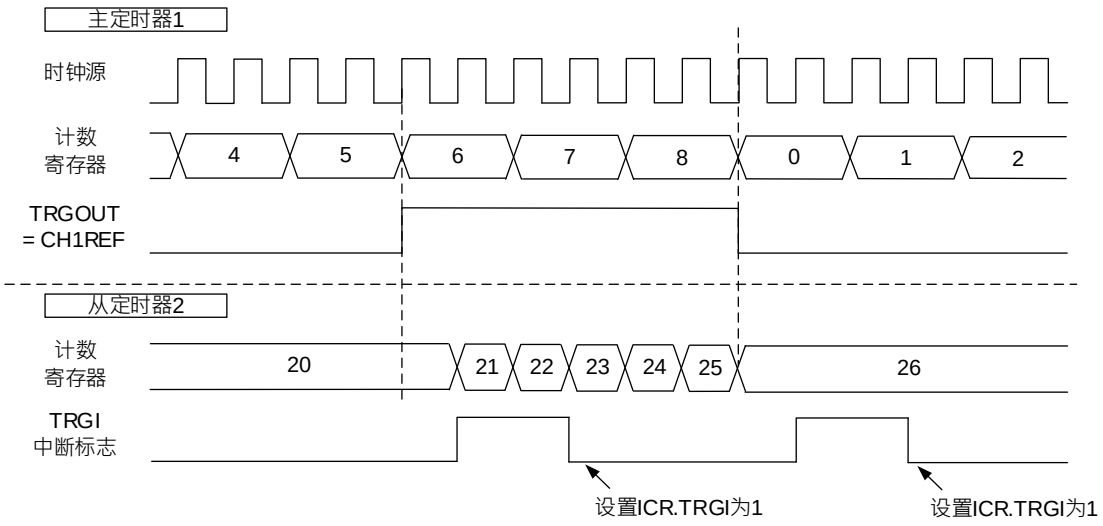


图 16-29　门控从定时器使用主定时器 CH1REF

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。

16. 4. 13. 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，当计数器上数到 2 后重载。
2. 设置 **GP32C4T_SMCON** 寄存器的 TSSEL1 位与 TSSEL2 位为 01001b(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，当计数器上数到 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。

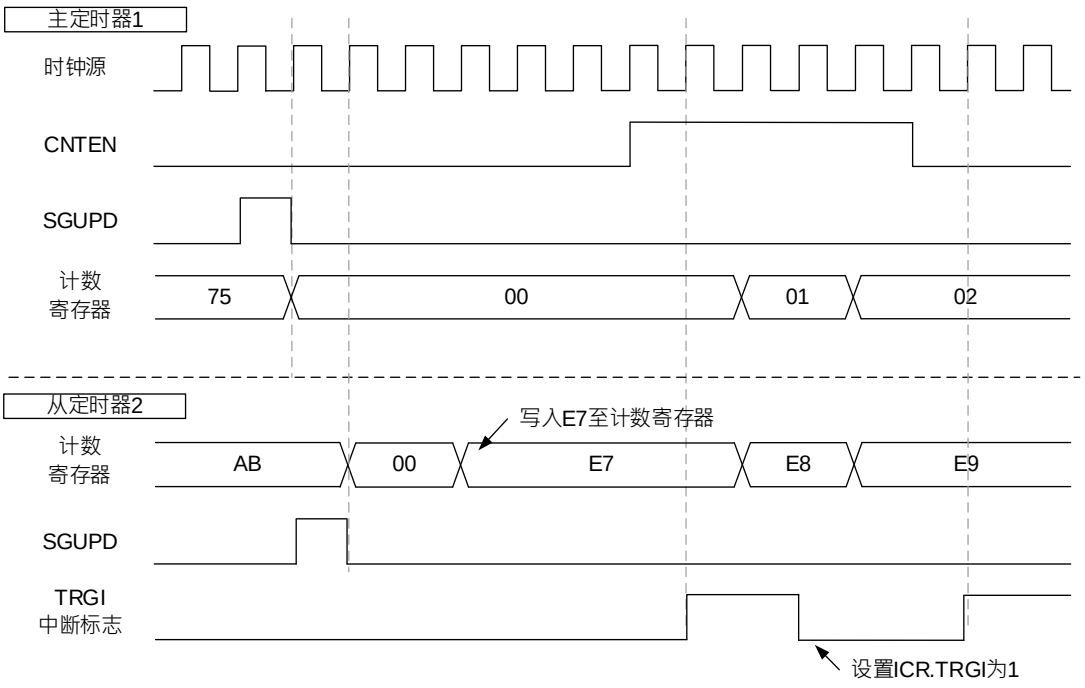


图 16-30　使用主定时器更新事件触发从定时器计数

16. 4. 13. 3　使用外部触发同步开启两个定时器

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)，参见图 16-31。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 01001b(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 GP16C2T1 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。

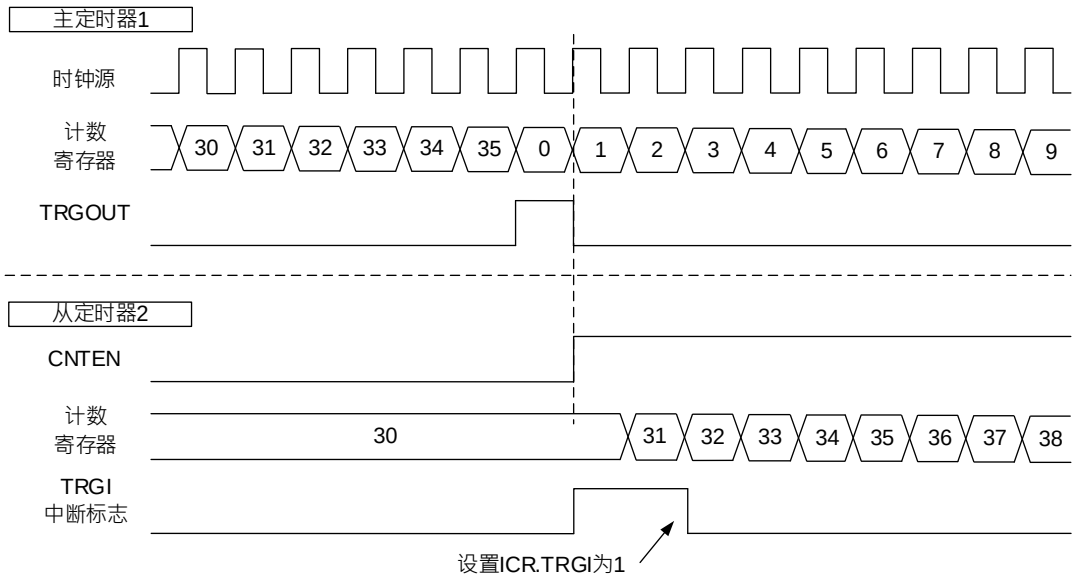


图 16-31　使用定时器 1 的 I1 输入触发定时器 1 和定时器 2

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。

16. 4. 14 调试模式

当微控制器进入调试模式(Cortex™-M0 核停止运行)，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。

为了安全起见，当进入调试模式时，设置 **GP32C4T_CON1** 寄存器的 DBGSEL 位，选择继续输出电平或是输出关闭(切换成输入)。当输出关闭时可以通过 GPIO 控制器选择输出电平，若未设定则为悬空输入。

16. 5 特殊功能寄存器

16. 5. 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_OPTR	$005C_H$	R/W	输入选择寄存器

16. 5. 2 寄存器描述

16. 5. 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
																DBGSEL						DFCKSEL<1:0>		ARPEN	CMSEL<1:0>		DIRSEL	SPMEN	UERSEL	DISUE	CNTEN

—	Bits 31-16	—	—
DBGSEL	Bit 15	R/W	**调试模式下，通道状态选择** 在输出模式中，选择通道为输入或持续输出 0:强制为输入 1:保持当前配置 注：此位仅在 SYSCFG_CFG 寄存器的 DBGHEN位配置GP32C4T时有效
—	Bit 14-10	—	—
DFCKSEL	Bit 9-8	R/W	**时钟预分频器** 设置定时器的频率(INT_CLK)，死区产生器与数字滤波器(ETR，In)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	**自动重载缓冲功能开启** 发生更新事件时，将设定的值载入至影子寄存器中 0:GP32C4T_AR 寄存器未缓冲 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信道

			(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信道(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信道(GP32C4T_CHMRn 寄存器中 CCnSSEL=00)的比较匹配中断请求。
DIRSEL	Bit 4	R/W	**计数方向选择** 当计数器配置为中心对齐模式时，此位只能读取计数器的计数方向 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	**更新事件请求来源选择** 设置更新事件(UPD)的来源 0:下列事件都会产生更新中断: －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
DISUE	Bit 1	R/W	**更新事件关闭** 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影子寄存器载入预装载值 －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 寄存器的影子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

16. 5. 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
																								I1SEL	MMSEL<2:0>						

—	Bits 31-8	—	—
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 引脚的异或组合(XOR)输出
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(TRGOUT)与 ADC 输入 000:复位 - 设置 GP32C4T_SGE 寄存器中的 UPD 位为触发输出(TRGOUT)。如果复位由触发输入生成(从模式控制器配置为复位模式)，则 TRGOUT 上的信号与实际复位信号之间会有延迟 001:开启 - 设置 GP32C4T_CON1 寄存器中的 CNTEN 位为触发输出(TRGOUT)，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 010:更新事件 – 更新事件被用于触发输出(TRGOUT)。一个主定时器的更新事件可当作从定时器的预分频器时钟 011:比较脉冲 – 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(TRGOUT)会送出一个正脉冲 100:比较信号 – CH1REF 信号用于触发输出(TRGOUT) 101:比较信号 – CH2REF 信号用于触发输出(TRGOUT) 110:比较信号 – CH3REF 信号用于触发输出(TRGOUT) 111:比较信号 – CH4REF 信号用于触发输出(TRGOUT)
—	Bit3-0	—	—

16. 5. 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
—	—	—	—	—	—	—	—	—	—	TSSEL2 <1:0>		—	—	—	—	ETPOL	ECM2EN	ETPRES <1:0>		ETFLT<3:0>				MSCFG	TSSEL1<2:0>			—	SMODS<2:0>		

—	Bits 31-22	—	—
TSSEL2	Bit 21-20	R/W	**触发选择2** 参照TSSEL1描述
—	Bit 19-16	—	—
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** 设置外部时钟模式2 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TRGI 连 到 ETP(SMODS=111 和 TSSEL=00111) 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	**外部触发时钟分频器** 外部触发输入信号ETRP频率不得超过1/4的PCLK，分频器可以降低ETP的频率，有效应用于快速的外部时钟源。 00:分频器关闭 01:ETRP频率分频2 10:ETRP频率分频4

			11:ETRP频率分频8
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}，滤波器关闭 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** 0:写入 0 无效 1:延迟触发输入(TRGI)上的事件来允许当前计时器和其从器件之间的同步。该设置有效用于使用单个外部事件来同步多个计时器。
TSSEL1	Bit 6-4	R/W	**触发选择 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 00110: I2 滤波后信号 00111:外部触发输入

			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) 01100:内部触发 8 (IT8) 其他:内部触发 n (ITn) 注:此位需要在使用前设定(SMODS=000)，以避免产生错误的上升/下降沿至计数器
—	Bit 3	—	—
SMODS	Bit 2-0	R/W	**从模式选择** 000:从模式关闭 – 当 GP32C4T_CON1 寄存器 CNTEN 位为 1 时，分频器时钟由内部时钟提供 001:编码器模式 1 – 计数器根据 I1 边沿检测电平在 I2 边沿检测边沿递增或递减计数 010:编码器模式 2 – 计数器根据 I2 边沿检测电平在 I1 边沿检测边沿递增或递减计数 011:编码器模式 3 – 计数器在 I1 边沿检测和 I2 边沿检测的边沿计数，计数的方向取决于另一个输入的电平 100:复位模式 – 选中的触发输入(TRGI)的上升沿重新初始化计数器，并且产生一次更新事件 101: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 110:触发模式 – 计数器在触发输入 TRGI 的上升沿启动(但不复位)，只有计数器的启动是被控制 111: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 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

从定时器	IT4(TSSEL =01000)	IT5(TSSEL =01001)	IT6(TSSEL =01010)	IT7(TSSEL =01011)	IT8(TSSEL =01100)
GP32C4T	保留	GP16C2T1	GP16C2T2	GP16C2T3	GP16C2T4

表 16-2　GP32C4T 内部触发连接

16. 5. 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
—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	CH4O	CH3OV	CH2OV	CH1OV	—	—	TRGI	—	CH4	CH3	CH2	CH1	UPD

—	Bits 31-13	—	—
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

16. 5. 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
																			CH4OV	CH3OV	CH2OV	CH1OV			TRGI		CH4	CH3	CH2	CH1	UPD

—	Bits 31-13	—	—
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

16. 5. 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
—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	CH4OV	CH3OV	CH2OV	CH1OV	—	—	TRGI	—	CH4	CH3	CH2	CH1	UPD

—	Bits 31-13	—	—
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

16. 5. 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
																			CH4OV	CH3OV	CH2OV	CH1OV			TRGI		CH4	CH3	CH2	CH1	UPD

—	Bits 31-13	—	—
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

16. 5. 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
—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	CH4OV	CH3OV	CH2OV	CH1OV	—	—	TRGI	—	CH4	CH3	CH2	CH1	UPD

—	Bits 31-13	—	—
CH4OV	Bit 12	R	**通道4捕获溢出，标志位中断状态** 0:无发生中断 1:已发生中断
CH3OV	Bit 11	R	**通道 3 捕获溢出，标志位中断状态** 0:无发生中断 1:已发生中断
CH2OV	Bit 10	R	**通道 2 捕获溢出，标志位中断状态** 0:无发生中断 1:已发生中断
CH1OV	Bit 9	R	**通道 1 捕获溢出，标志位中断状态** 0:无发生中断 1:已发生中断
—	Bit 8-7	—	—
TRGI	Bit 6	R	**触发，标志位中断状态** 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，标志位中断状态** 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，标志位中断状态** 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，标志位中断状态** 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，标志位中断状态** 0:无发生中断

			1:已发生中断
UPD	Bit 0	R	**更新，标志位中断状态** 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

16. 5. 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
																			CH4OV	CH3OV	CH2OV	CH1OV			TRGI		CH4	CH3	CH2	CH1	UPD

—	Bits 31-13	—	—
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

16. 5. 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	T_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
—	Bit 5	—	—
SGCH4	Bit 4	T_W1	**软件生成通道 4 捕获或比较事件** 参照 SGCH1 描述
SGCH3	Bit 3	T_W1	**软件生成通道 3 捕获或比较事件** 参照 SGCH1 描述
SGCH2	Bit 2	T_W1	**软件生成通道 2 捕获或比较事件**

			参照 SGCH1 描述
SGCH1	Bit 1	T_W1	**软件生成通道 1 捕获或比较事件** 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，则产生中断。 **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，若开启中断，则产生中断。
SGUPD	Bit 0	T_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(递增计数)，则计数器将清零；否则如果 DIRSEL=1(递减计数)，则将使用自动重载寄存器值(GP32C4T_AR)。

16. 5. 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
																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 31-16	—	—
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 或 I1 的双边沿检测
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 CHREF_CLR 为高电平时清除(设置 GP32C4T_OPTR 寄存器的 ETR_RMP 位选择来源)
CH1MOD	Bit 6-4	R/W	**输出比较通道 1 模式**

			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GP32C4T_CCEP寄存器中的CC1POL位和CC1NPOL位。 000:关闭－无作用 001: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1 010: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0 011:匹配时设置翻转电平－当计数器(GP32C4T_COUNT)匹配GP32C4T_CCVAL1寄存器时，CH1REF设置翻转电平(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) 100:强制低电平－CH1REF强制设置低电平 101:强制高电平－CH1REF强制设置高电平 110: PWM模式1－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 111: 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
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影子寄存器中。 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 只在信道被配置成 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 或 I1 的双边沿检测

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 或 I1 的双边沿检测
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}，滤波器关闭 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 或 I1 的双边沿检测

16. 5. 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
																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 31-16	—	—
CH4OCLREN	Bit 15	R/W	**输出比较通道 4 清除开启** 参照 CH1OCLREN 描述
CH4MOD	Bit 14-12	R/W	**输出比较通道 4 模式** 参照 CH1MOD 描述
CH4PEN	Bit 11	R/W	**输出比较通道 4 预装载开启** 参照 CH1PEN 描述
CH4FEN	Bit 10	R/W	**输出比较通道 4 快速开启** 参照 CH1FEN 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 或 I1 的双边沿检测
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 CH1OMOD 描述
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 或 I1 的双边沿检测

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I1FLT描述
I4PRES	Bit 11-10	R/W	**输入捕获通道 4 预分频器** 参照 IC1PRES 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 或 I1 的双边沿检测
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 或 I1 的双边沿检测

16. 5. 2. 13 捕获或比较开启极性寄存器(GP32C4T_CCEP)

捕获或比较开启寄存器(GP32C4T_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
—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	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:**

			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。
—	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。 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

16. 5. 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
CNTV<31:0>																															

CNTV	Bits 31-0	R/W	**计数器数值**

16. 5. 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数值载入影子寄存器中

16. 5. 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: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<31:0>																												

ARV	Bits 31-0	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，设置数值为 0 时计数器停止计数

16. 5. 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。 如果在 GP32C4T_CHMR1 寄存器中的预载功能没有选中，CCRV1 中的值将被永久载入；否则每当发生更新事件，预载值将会复制到有效的捕获或比较寄存器中。有效捕获或比较寄存器的值将会与 GP32C4T_COUNT 中的值进行比较，并在 CH1 上输出。 **通道 CH1 配置为输入:** CCRV1为由上一个输入捕获事件(I1)发生时的计数器数值。

16. 5. 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** **通道CH2配置为输出:** CCRV2是捕获或比较寄存器的预装载值。 如果在GP32C4T_CHMR1寄存器中的预载功能没有选中，CCRV2中的值将被永久载入;否则每当发生更新事件，预载值将会复制到有效的捕获或比较寄存器中。有效捕获或比较寄存器的值将会与GP32C4T_COUNT中的值进行比较，并在CH2上输出。 **通道CH2配置为输入:** CCRV2为由上一个输入捕获事件(I2)发生时的计数器数值。

16. 5. 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** **通道CH3配置为输出:** CCRV3是捕获或比较寄存器的预装载值。 如果在GP32C4T_CHMR2寄存器中的预载功能没有选中，CCRV3中的值将被永久载入;否则每当发生更新事件，预载值将会复制到有效的捕获或比较寄存器中。有效捕获或比较寄存器的值将会与GP32C4T_COUNT中的值进行比较，并在CH3上输出。 **通道CH3配置为输入:** CCRV3为由上一个输入捕获事件(I3)发生时的计数器数值。

16. 5. 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** **通道CH4配置为输出:** CCRV4是捕获或比较寄存器的预装载值。 如果在GP32C4T_CHMR2寄存器中的预载功能没有选中，CCRV4中的值将被永久载入;否则每当发生更新事件，预载值将会复制到有效的捕获或比较寄存器中。有效捕获或比较寄存器的值将会与GP32C4T_COUNT中的值进行比较，并在CH4上输出。 **通道CH1配置为输入:** CCRV4为由上一个输入捕获事件(I)发生时的计数器数值。

16. 5. 2. 21　输入选择寄存器(GP32C4T_OPTR)

输入选择寄存器(GP32C4T_OPTR)																											
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
																						ETR_RMP<1:0>		CH4_RMP<1:0>		CH3_RMP<1:0>		CH2_RMP<1:0>		CH1_RMP<1:0>	

—	Bits 31-10	—	—
ERT_RMP	Bit 9-8	R/W	**ETR输入选择** 00:GP32C4T_ETR输入 01:CMP_OUT 其他:保留
CH4_RMP	Bit 7-6	R/W	**CH4输入选择** 00:GP32C4T_CH4输入 其他:保留
CH3_RMP	Bit 5-4	R/W	**CH3输入选择** 00:GP32C4T_CH3输入 其他:保留
CH2_RMP	Bit 3-2	R/W	**CH2输入选择** 00:GP32C4T_CH2输入 01:保留 10:RTC 1Hz 11:保留
CH1_RMP	Bit 1-0	R/W	**CH1 输入选择** 00:GP32C4T_CH1 输入 01:CMP_OUT 10:RTC wakeup 11: MCO

第17章 通用定时器 16 位 2 通道 (GP16C2T)

17. 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。

17. 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
  - 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:
  - 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

17. 3 结构图

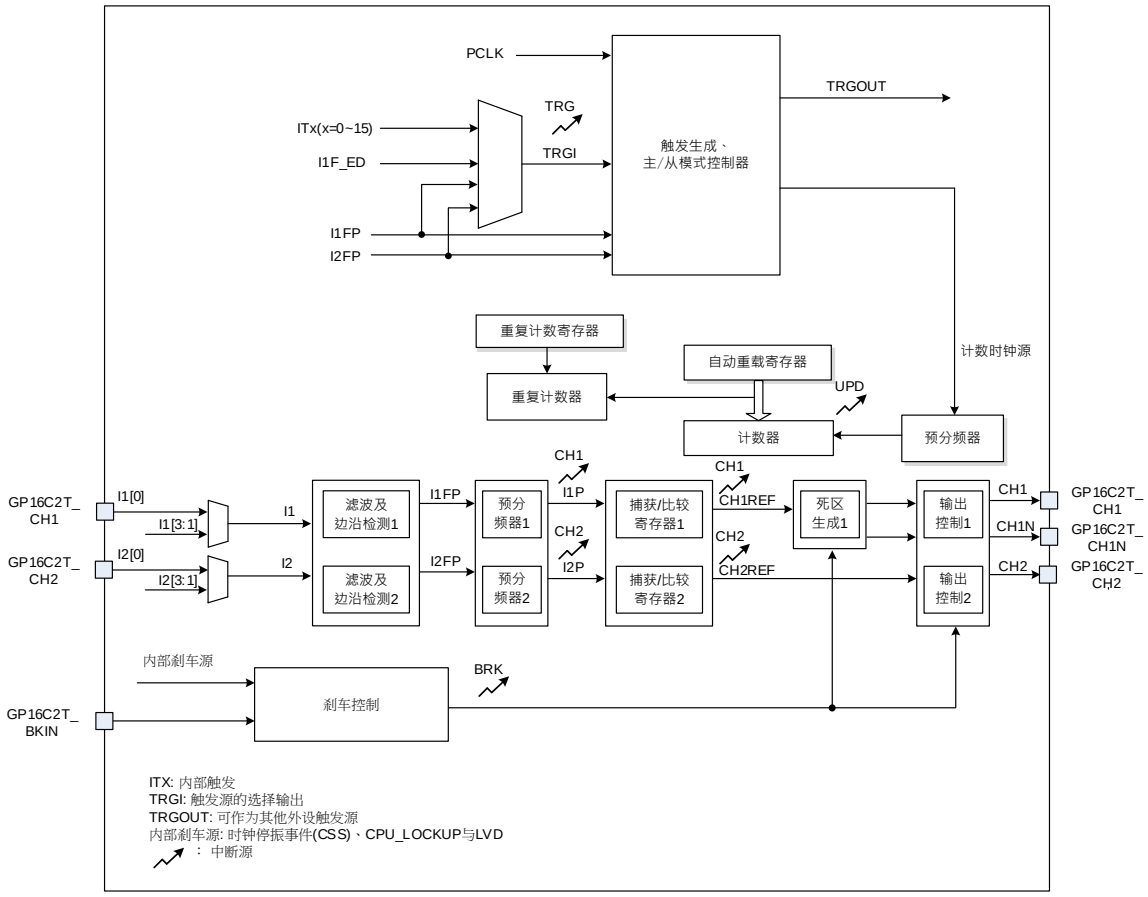


图 17-1　GP16C2T 定时器结构框图

17. 4 功能描述

17. 4. 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** 的重载值会被立即反应到影子寄存器中；而设置 ARPEN 位为 1 时，**GP16C2T_AR** 寄存器具有缓冲功能，当产生更新事件(UPD)时，**GP16C2T_AR** 寄存器的重载值才会被更新到影子寄存器中。

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 APB 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。

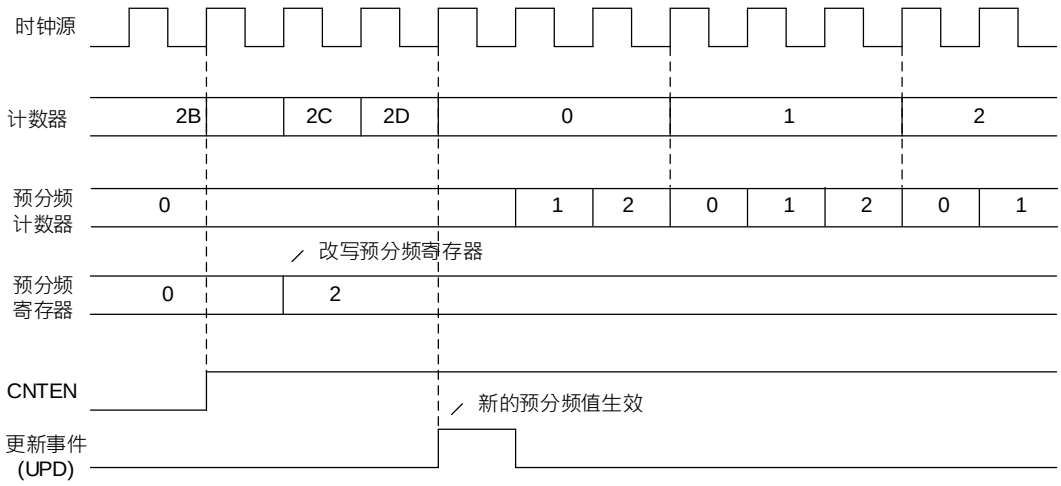


图 17-2 预分频值计数时序图

17. 4. 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，缓冲寄存器数值会立即更新到重复计数器的影子寄存器。

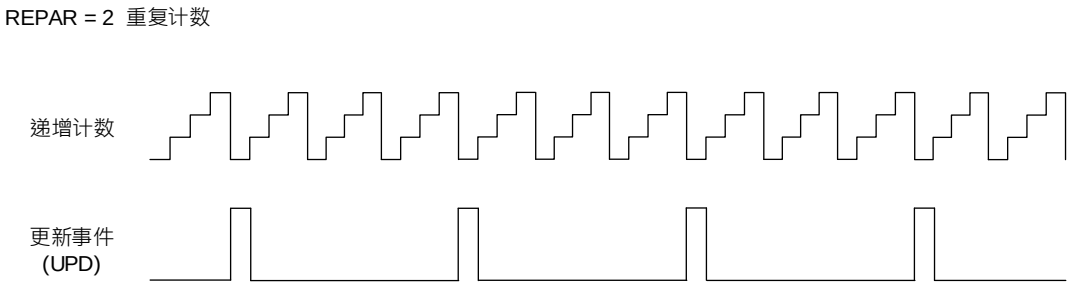


图 17-3 重复计数器工作模式

注意:置位 GP16C2T_SGE 寄存器中的 SGUPD 位为 1，也可以产生更新事件。

17. 4. 3 时钟源

计数器工作时钟可以选择内部时钟(INT_CLK)、外部时钟源 1(I1、I2) 与内部触发输入(IT0~IT15)。

17. 4. 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 000b)，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。

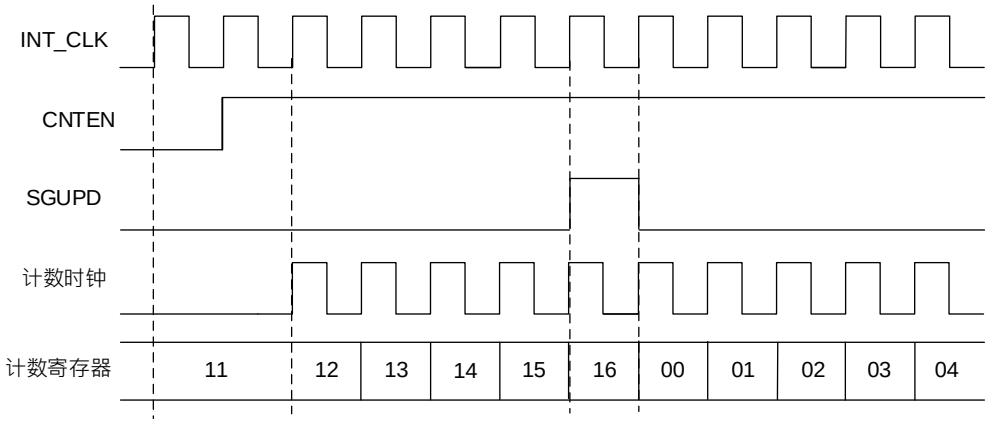


图 17-4 采用内部时钟计数

17. 4. 3. 2 外部时钟源 1

GP16C2T_SMCON 寄存器的 SMODS 位为 111b 时，可选择外部时钟源 1。计数器可根据选定的上升沿或下降沿计数。

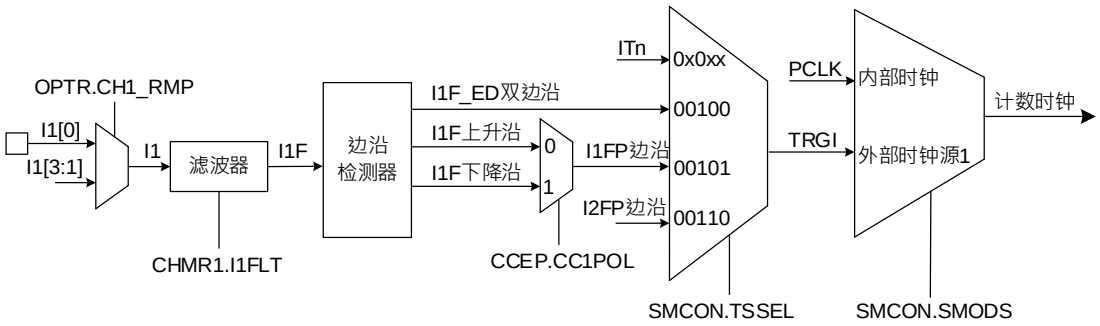


图 17-5 外部时钟连接

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_OPTR** 寄存器的 CH1_RMP 位为 00b，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 0000)。
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 TSSEL1 位与 TSSEL2 位为 00101b，选择外部时钟源为 I1。
6. 设置 **GP16C2T_SMCON** 寄存器的 SMODS 位为 111b，选择定时器外部时钟模式 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时，取决于 I1 输入的同步电路。

17. 4. 3. 3 内部触发输入(ITn)

设置 **GP16C2T_SMCON** 寄存器的 SMODS 位为 111b，外部时钟模式 1。计数器根据选定的内部输入端(ITn)的上升沿计数。

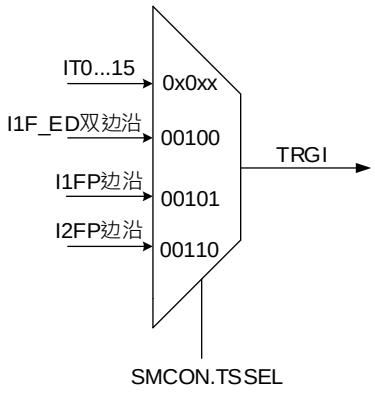


图 17-6 ITn 内部时钟连接

设置计数器外部时钟源为 ITn 输入，并在 ITn 上升沿时计数，步骤如下:

1. 设置 **GP16C2T_SMCON** 寄存器的 TSSEL1 位与 TSSEL2 位，选定 ITn 作为外部时钟源。
2. 设置 **GP16C2T_SMCON** 寄存器的 SMODS 位为 111b，设置外部时钟模式 1。
3. 设置 **GP16C2T_CON1** 寄存器的 CNTEN 位为 1，开启计数器。

ITn 产生上升沿时，计数器计数一次。

17. 4. 4 计数模式

17. 4. 4. 1 递增计数模式

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。

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)更新影子寄存器。在设置 DISUE 位为 0 之前都不会产生更新事件(UPD)，计数器和预分频器都会重新从 0 开始计数。

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。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影子寄存器:

- 更新 **GP16C2T_REPAR** 寄存器数值到影子寄存器
- 更新 **GP16C2T_AR** 寄存器数值到影子寄存器
- 更新 **GP16C2T_PRES** 寄存器数值到影子寄存器

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。

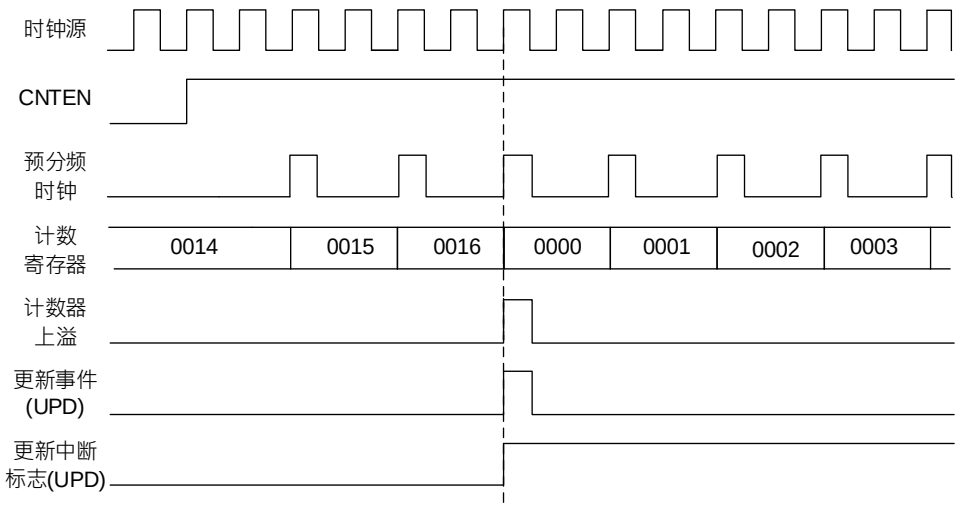


图 17-7 计数器递增计数时序图

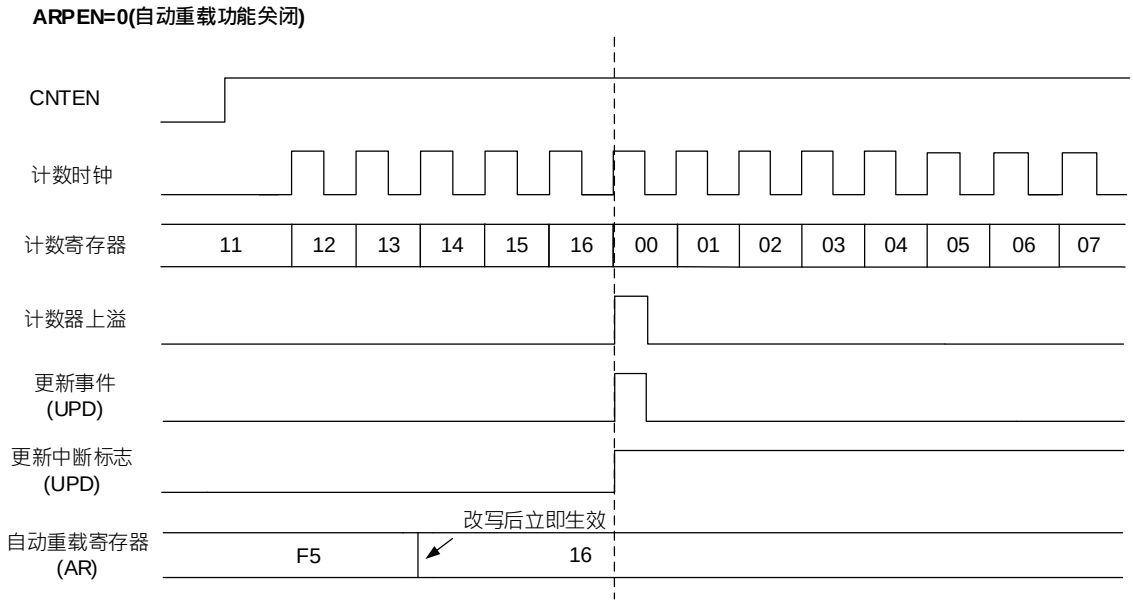


图 17-8　设置 ARPEN 位为 0 时计数器时序图

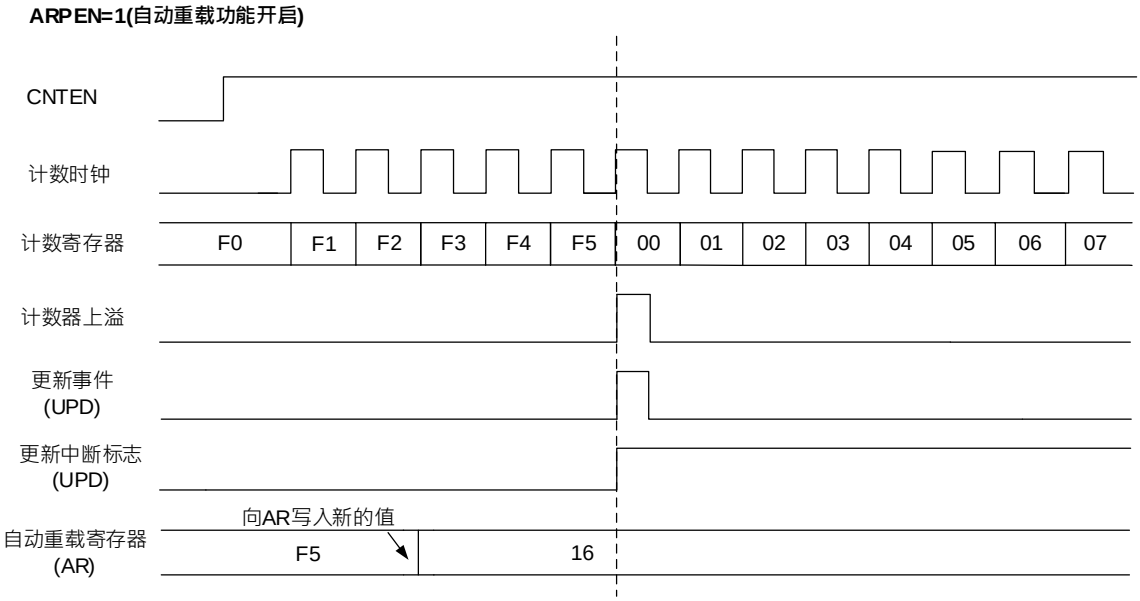


图 17-9　设置 ARPEN 位为 1 时计数器时序图

17. 4. 5 捕获或比较通道

输入电路对 In 输入端的信号进行采样，产生一个经过滤波的信号 InF。之后一个可极性选择的边沿检测器产生 In 边沿检测信号，该信号可作为从模式控制器的触发输入或作为捕获控制命令，且该信号经过分频后进入捕获寄存器。

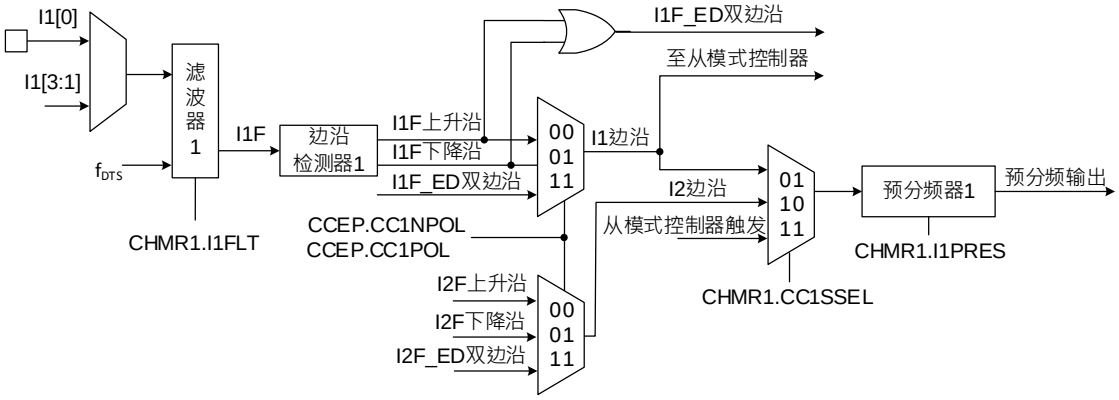


图 17-10　捕获或比较通道

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

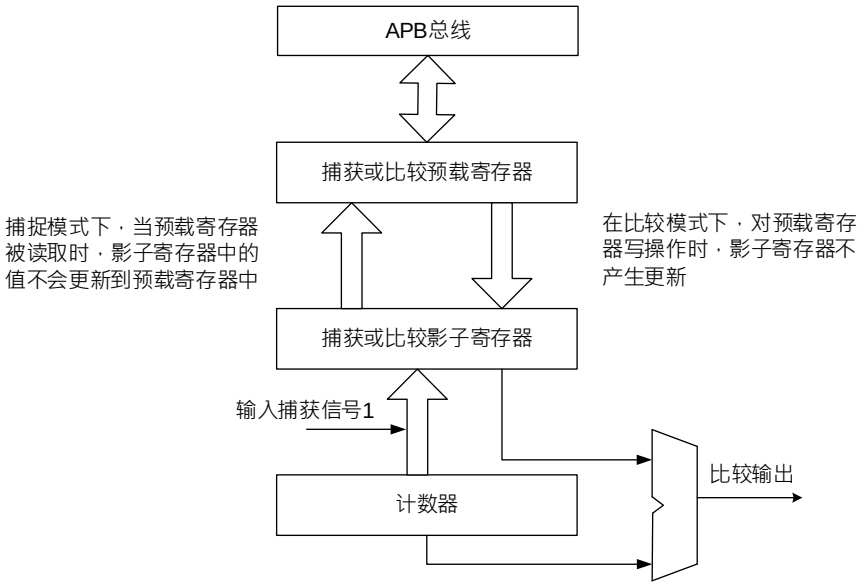


图 17-11　捕获或比较通道结构图

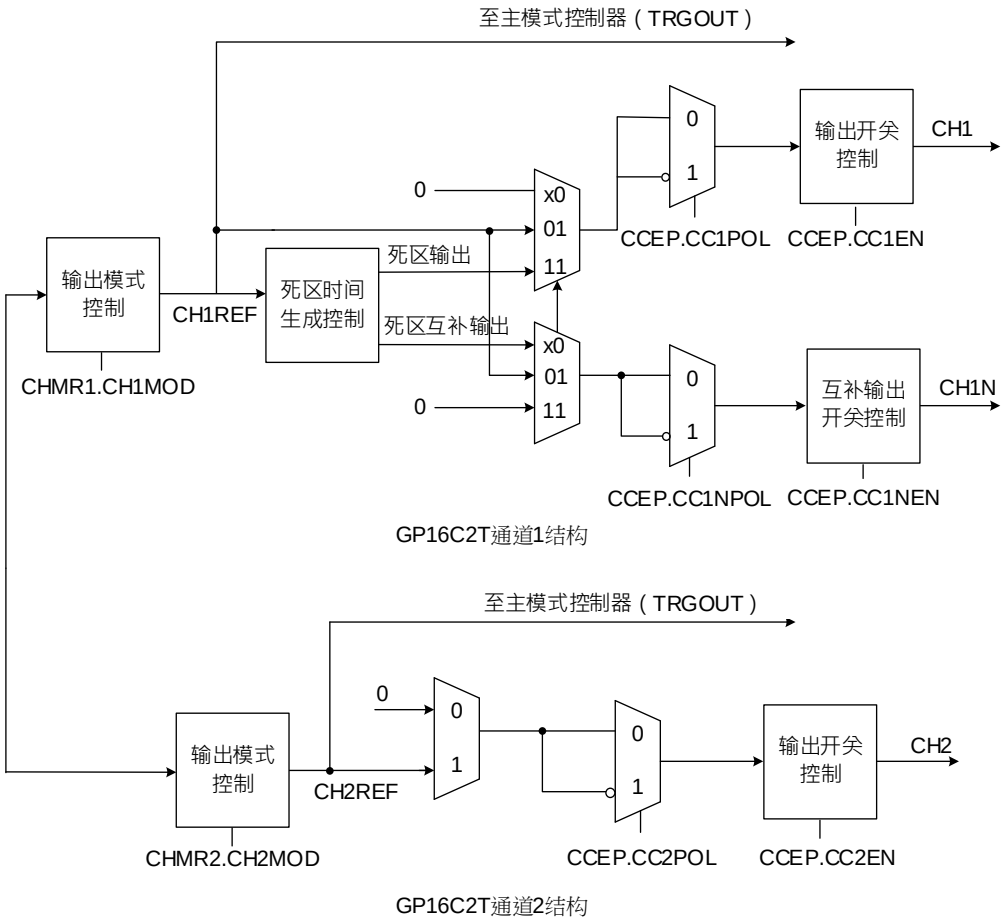


图 17-12　捕获或比较通道的输出部分

17. 4. 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。当捕获发生时，**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(如果有开启)。

当 **GP16C2T_RIF** 寄存器中相应的 CHn 标志位已经为 1，又发生捕获事件时，**GP16C2T_RIF** 寄存器中相应的过捕获 CHnOV 标志位也会被设定为 1，表示发生过捕获事件。

通过软件设置 **GP16C2T_ICR** 寄存器的 CHn 位与 CHnOV 位为 1，清除 **GP16C2T_RIF** 寄存器中 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，连续 8 次采样才确认电平变化有效。
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. 设置 **GP16C2T_CCEP** 寄存器的 CC1EN 位为 1，开启捕获计数器。
6. 如有需要，设置 **GP16C2T_IER** 寄存器的 CH1 位为 1，开启中断请求。

当发生输入捕获时:

1. 有效边沿产生，**GP16C2T_CCVAL1** 寄存器获取计数器数值。
2. 硬件自动设置 CH1 标志位为 1(中断标志位)。若至少 2 个连续的捕获发生，但标志位没有及时清除，则会设置 CH1OV 位为 1。
3. 中断的产生取决于 **GP16C2T_IER** 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前先读取捕获数据。避免丢失在读过捕获标志位到读捕获数据之间的重复捕获信息。

注:捕获中断请求可由软件设置 GP16C2T_SGE 寄存器的 SGCHn 位产生。

17. 4. 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_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，连续 8 次采样才确认电平变化有效。
3. 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** 寄存器和计数器清零。
4. 设置 **GP16C2T_CHMR1** 寄存器的 CC2SSEL 位为 10b，通道 2 选择 I1 为有效输入端。
5. 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。
6. 设置 **GP16C2T_SMCON** 寄存器的 TSSEL1 位与 TSSEL2 位为 00101b，选择 I1 滤波后信号为有效的触发输入。
7. 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。
8. 设置 **GP16C2T_CCEP** 寄存器的 CC1EN 位和 CC2EN 位为 1，开启捕获。

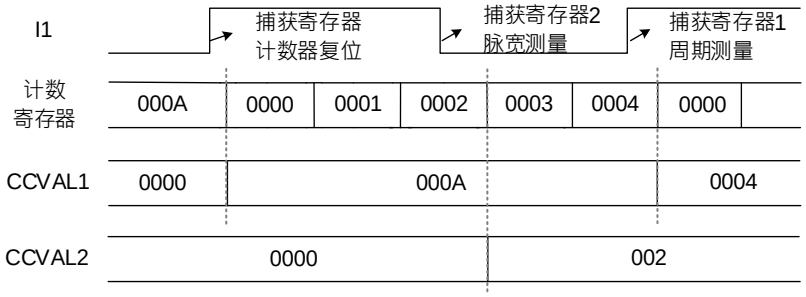


图 17-13 PWM 输入模式时序

- **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 = (GP16C2T_PRES + 1) * (GP16C2T_CCVALn + 1) / f_{INT_CLK}

17. 4. 8 PWM输出模式

脉宽调制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。

每个信道的 PWM 模式是相互独立的(每个 CHn 输出一个 PWM)，只需设置 **GP16C2T_CHMRn** 寄存器的 CHnMOD 位为 110(PWM 模式 1)或为 111(PWM 模式 2)。

可通过设置 **GP16C2T_CHMRn** 寄存器的 CHnPEN 位为 1 来开启相应的预装载寄存器，及设置 **GP16C2T_CON1** 寄存器的 ARPEN 位为 1 来开启自动重载功能。

开启预装载、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写入到影子寄存器中，因此在开启计数前，必须通过设置 **GP16C2T_SGE** 寄存器的 SGUPD 位为 1 来初始化所有的寄存器。

CHn 的极性可通过 **GP16C2T_CCEP** 寄存器的 CCnPOL 位设置，有效电平可设置为高电平或低电平。CHn 的输出由 CCnEN、CCnNEN、GOEN、OFFSSI 和 OFFSSR 位(**GP16C2T_CCEP** 和 **GP16C2T_BDCFG** 寄存器)组合控制。

在 PWM 模式(1 或 2)中，**GP16C2T_COUNT** 会持续与 **GP16C2T_CCVALn** 寄存器数值比较，以确定 **GP16C2T_COUNT** 是否< GP16C2T_CCVALn。

17. 4. 8. 1 PWM边沿对齐模式

- 递增计数配置

下图以 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 110b，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，当计数器上数到 8 后重载。
5. 设置 **GP16C2T_CON1** 寄存器的 CNTEN 位为 1，开启计数器。

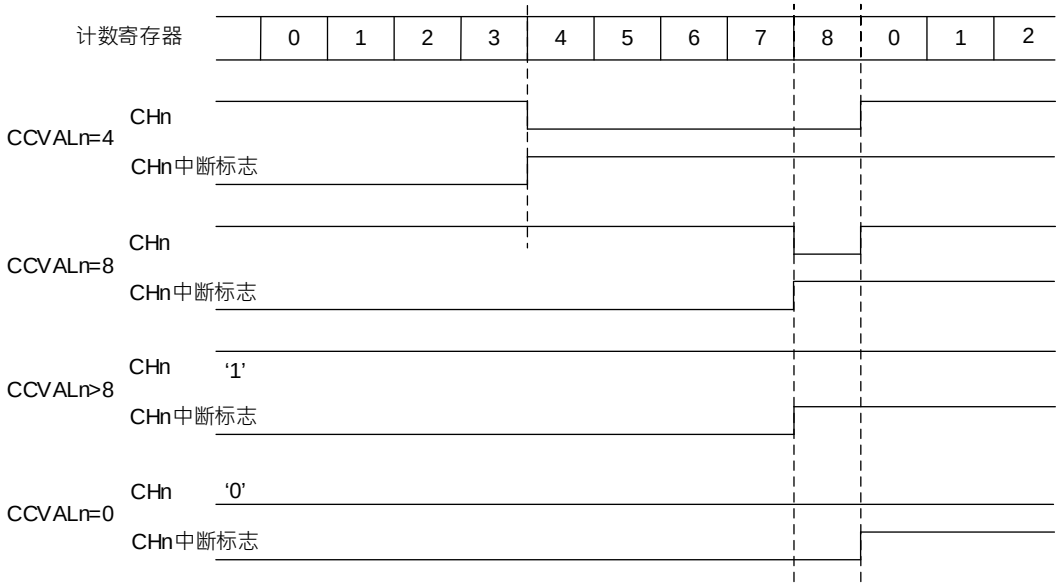


图 17-14 边沿对齐递增计数 PWM 波形(AR=8)

- **GP16C2T_COUNT<GP16C2T_CCVAL1** 时，CH1 为高电平。
- **GP16C2T_COUNT >= GP16C2T_CCVAL1** 时，CH1 为低电平。

其中比较特别的是，若设定 **GP16C2T_CCVAL1>GP16C2T_AR** 时，CH1 会永远输出高电平；若 **GP16C2T_COUNT** = 0 时，CH1 会永远输出低电平。

17. 4. 9 输出比较模式

该功能用于控制输出波形或指示周期时间的结束。

当捕获或比较寄存器和 **GP16C2T_COUNT** 寄存器数值匹配时，输出比较功能:

- 设置 **GP16C2T_CHMRn** 寄存器的 CHnMOD 位选择输出模式，输出极性由 **GP16C2T_CCEP** 寄存器的 CCnPOL 位控制 ：
  - ◇ 设置 CHnMOD 位为 000b:当计数器匹配比较器时输出保持其电平。
  - ◇ 设置 CHnMOD 位为 001b: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 010b: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 011b: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。

设置 **GP16C2T_CHMRn** 寄存器的 CHnPEN 位数值可决定 **GP16C2T_CCVALn** 寄存器是否带有预装载寄存器。

在输出比较模式中，更新事件 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 011b，当 CNTV 与 CCRVALn 匹配时，CHn 输出翻转。
   - 设置 CHnPEN 位为 0，关闭预装载寄存器。
   - 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影子寄存器。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。

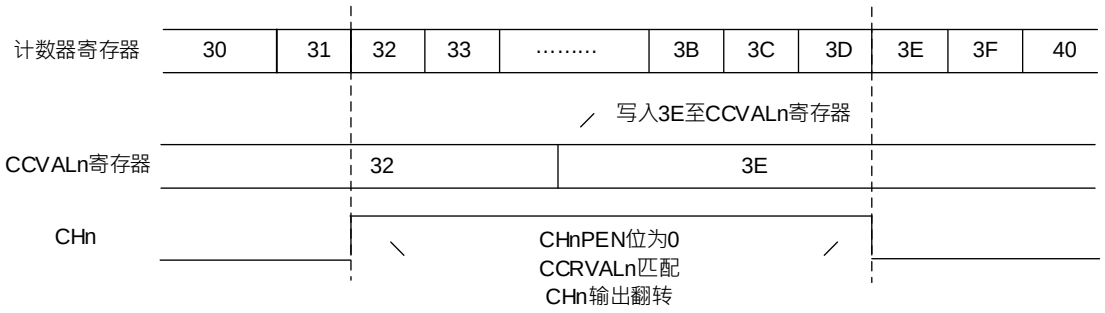


图 17-15　输出比较模式，触发 CHn

17. 4. 9. 1　　强制输出模式

设置 **GP16C2T_CHMRn** 寄存器的 CCnSSEL 位为 00b 开启输出模式，在此模式下通过软件设置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 101b，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 100b 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。

17. 4. 10 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配置的波形。

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** 寄存器和 **GP16C2T_COUNT** 寄存器数值不同时，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)

基于 PWM 模式设置单脉冲输出波形的步骤如下:

1. 设置 **GP16C2T_CHMRn** 寄存器的 CHnMOD 位，选择 PWM 模式 1 或 2。
2. 设置 **GP16C2T_CCEP** 寄存器的 CCnPOL 位，选择通道 CHn 的输出极性。
3. 设置 **GP16C2T_CON1** 寄存器的 SPMEN 位为 1，开启单脉冲模式。
4. 设置 **GP16C2T_CHMR1** 寄存器的 CH1PEN 位为 1，**GP16C2T_CON1** 寄存器的 ARPEN 位为 1，开启比较寄存器和计数重载寄存器的缓冲功能(也可以根据实际情况关闭缓冲)。

5. 设置 **GP16C2T_CCVALn** 寄存器和 **GP16C2T_AR** 寄存器，设置单脉冲输出延时和脉宽时间
6. 设置 **GP16C2T_SGE** 寄存器的 SGUPD 位为 1 来产生一个更新事件
7. 设置 **GP16C2T_CON1** 寄存器的 CNTEN 位为 1 来开启计数器，也可以在触发模式下，通过外部触发输入信号来触发硬件自动设置 CNTEN 位为 1。

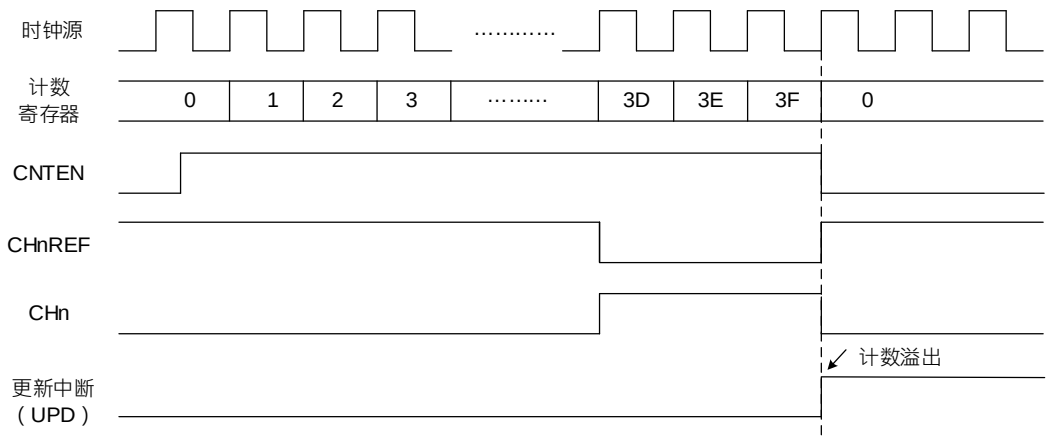


图 17-16　单脉冲模式

CHn 快速开启模式

在单脉冲模式下，In 输入的有效边沿会开启计数器(自动设置 CNTEN 位为 1)，在比较计数器数值后输出信号。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 PWM1 或 PWM2 模式时才能使用。

17. 4. 11 互补输出与死区时间

两个互补的通道输出信号，可以用来控制输出的瞬时开关。这个瞬时的延迟即为死区时间。

每个输出可独立选择输出极性(主输出 CHn 或互补输出 CHnN)，该操作可通过写 **GP16C2T_CCEP** 寄存器的 CCnPOL 和 CCnNPOL 位完成。

互补信号 CHn 和 CHnN 由几个控制位共同控制，分别是 **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。特别是切换为空闲状态(GOEN 变为 0)后，死区时间依然有效。

设置 CCnEN 和 CCnNEN 位为 1，开启死区时间插入，若有刹车电路，同样需要设置 GOEN 位为 1。**GP16C2T_BDCFG** 寄存器的 DT[7:0]可以控制所有通道的死区时间的产生。根据比较输出波形，产生 CHn 和 CHnN 两路输出。若 CHn 和 CHnN 有效电平为高:

- CHn 的输出信号与参考信号(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。

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

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

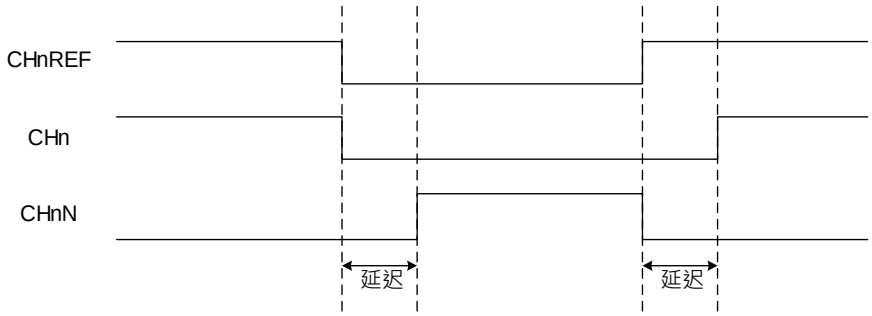


图 17-17　互补输出含死区时间插入

任何情况下，CHn 与 CHnN 的输出信号都不能同时为有效电平。

17. 4. 12 刹车功能

刹车功能模式由以下几个位控制输出开启信号和无效电平:

- **GP16C2T_BDCFG** 寄存器的 GOEN、OFFSSI 和 OFFSSR 位。
- **GP16C2T_CON2** 寄存器的 OISSn 和 OISSnN 位。

刹车源可以是刹车输入引脚(BKIN)、时钟停振事件、CPU_LOCKUP 与 LVD 以及软件控制 **GP16C2T_SGE** 寄存器的 SGBRK 位。时钟停振事件由时钟控制器(RCU)中的时钟安全系统(CSS)产生。时钟安全系统(CSS)详细信息可参考时钟安全系统章节。

系统复位后，刹车电路被关闭且 GOEN 位被复位。设置 **GP16C2T_BDCFG** 寄存器的 BRKEN 位为 1 可开启刹车功能，BRKP 位可选择刹车输入信号的极性。

由于 GOEN 的下降沿是异步信号，在 BKIN 输入和 **GP16C2T_BDCFG** 寄存器之间插入了一个同步电路。这也导致了异步和同步信号之间会产生一些延迟。

当发生刹车请求时(刹车输入端侦测到有效电平):

1. GOEN 位被清除。
2. CHn/CHnN 输出端根据 **GP16C2T_BDCFG** 寄存器中 OFFSSI 位，处于禁止输出、无效状态或空闲状态。
   - OFFSSI = 0 : CHn/CHnN 禁止输出，关闭输出开启信号，此时输出不由计数器控制。
   - OFFSSI = 1 : CHn/CHnN 输出空闲电平，由 **GP16C2T_CON2** 寄存器的 OISSn 位与 OISSnN 位决定。当使用互补输出且定时器时钟仍然存在时，CHn 与 CHnN 会在死区时间后输出相应的电平。在这种情况下，CHn 与 CHnN 的输出信号一样不能同时为有效电平。

3. 设置刹车状态标志位(**GP16C2T_RIF** 寄存器的 BRK 位)为 1 时，若设置 **GP16C2T_IER** 寄存器的 BRK 位为 1，则产生中断。
4. 若设置 **GP16C2T_BDCFG** 寄存器的 AOEN 位为 1，在下次更新事件(UPD)发生时，会自动设置 GOEN 位为 1。否则，GOEN 位会保持为 0，直到对其写为 1 操作，该特性可用于安全方面的应用，可以将刹车输入端接到一个电源驱动的报警端、热敏传感器或其他安全器件上。

注:当刹车输入有效电平时，不能设置 GOEN 位为 1(自动地或者通过软件)。同时状态标志位 BRK 不能被清除。

除刹车输入和输出管理，为保证应用程序的安全，内部刹车电路具有写保护功能。用户可冻结几个设置参数(死区时间，CHn/CHnN 极性和关闭时状态，CHnMOD 设置，刹车开启和极性)。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 位写入一次。

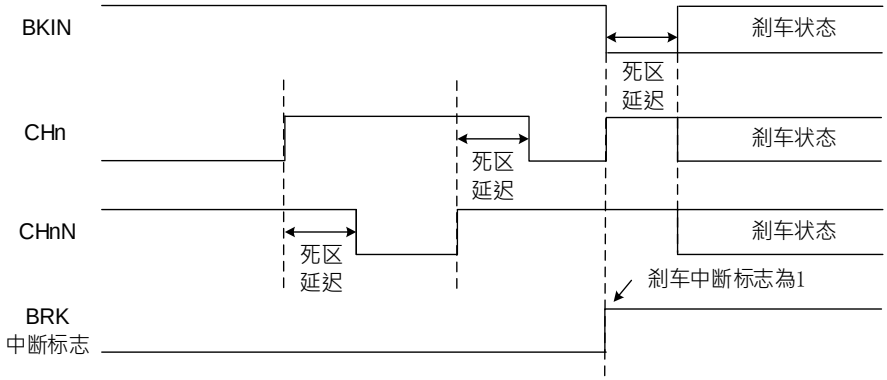


图 17-18　刹车输出行为

17. 4. 13 生成 6 步PWM

当通道使用互补功能时，设置 CHnMOD、CCnEN、CCnNEN 位为 1 提供预装载位，由设置 **GP16C2T_CON2** 寄存器的 CCUSEL 位为 1 开启预装载功能，当发生 COM 事件时将预装载寄存器的数值载入至影子寄存器。因此用户可以预先将配置写入寄存器，通过预装载寄存器可以在发生 COM 事件同时更改全部的通道配置。COM 事件可以通过设置 **GP16C2T_SGE** 寄存器的 SGCOM 位为 1 产生，也可以通过 TRGI 的上升沿产生。

当发生 COM 事件时，硬件自动设置 **GP16C2T_RIF** 寄存器的 COM 位为 1。如果设置 **GP16C2T_IER** 寄存器的 COM 位为 1，则产生中断。

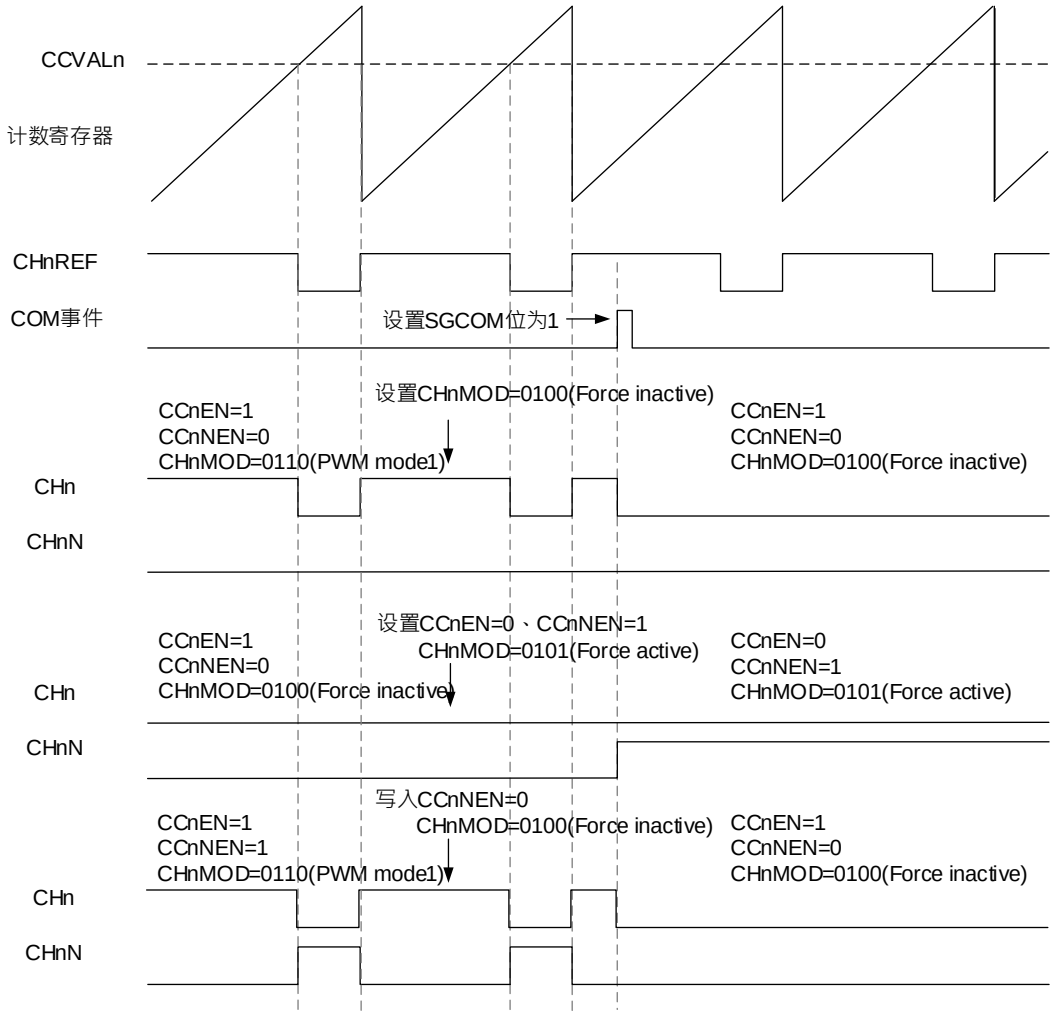


图 17-19　COM 事件生成 6 步 PWM

17. 4. 14 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。

17. 4. 14. 1 复位模式

计数器及其预分频器可以在回应触发输入事件时重新初始化。此外，若 **GP16C2T_CON1** 寄存器的 UERSEL 位为 0 时会产生一次更新事件 UPD。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 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 100b，选择复位模式。
7. 设置 **GP16C2T_CON1** 寄存器的 CNTEN 位为 1，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)，如果中断开启(取决于 **GP16C2T_IER** 寄存器的 TRGI 位)，会发送中断。

下图给出了设置自动重载寄存器 **GP16C2T_AR** 为 0x36 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。

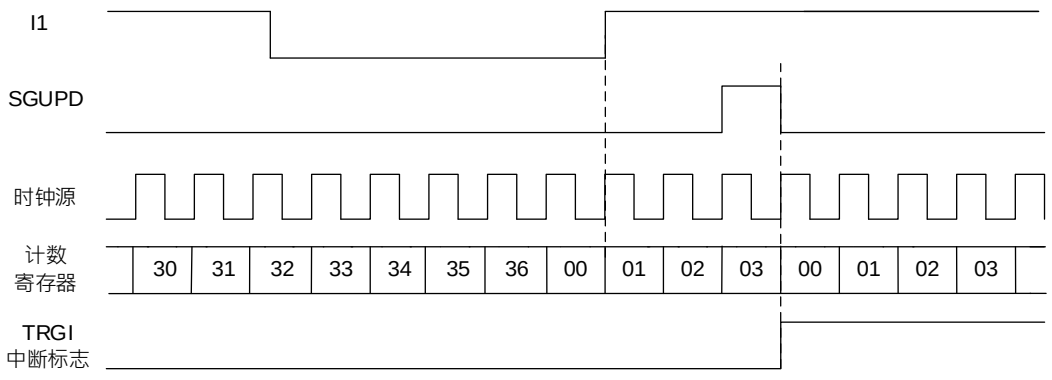


图 17-20 复位模式控制电路

17. 4. 14. 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时。

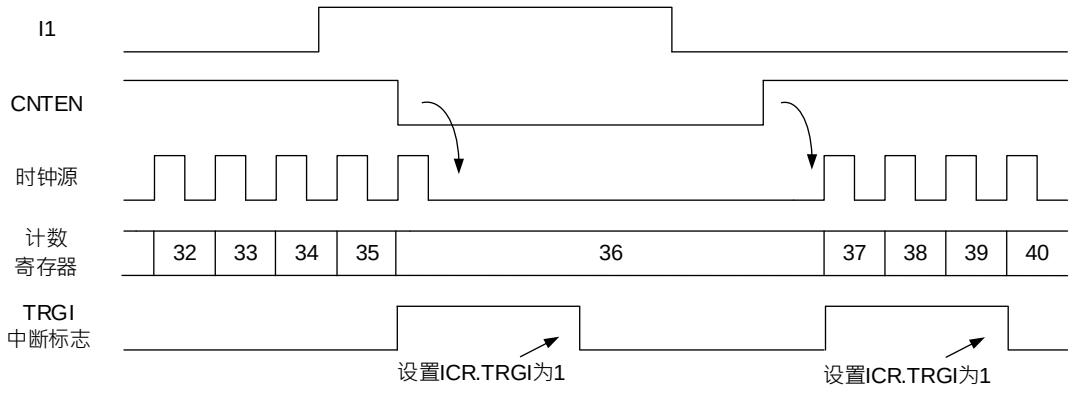


图 17-21　门控模式控制电路

17. 4. 14. 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 1，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时。

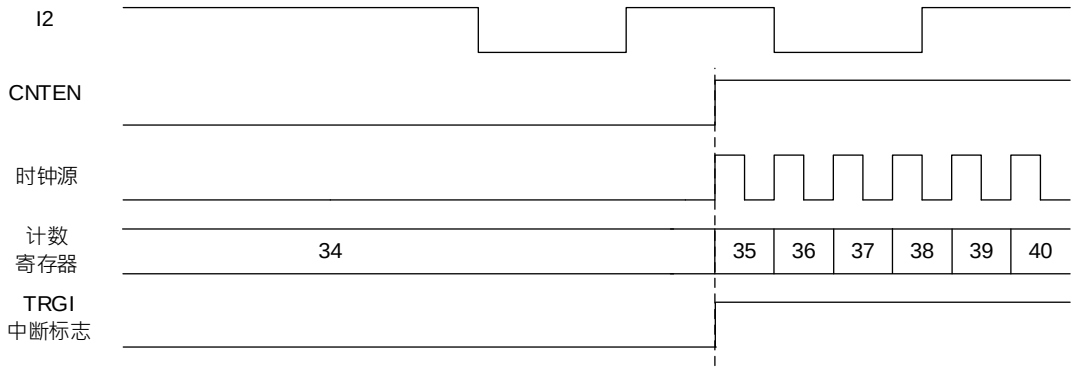


图 17-22　触发模式控制电路

17. 4. 15 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。

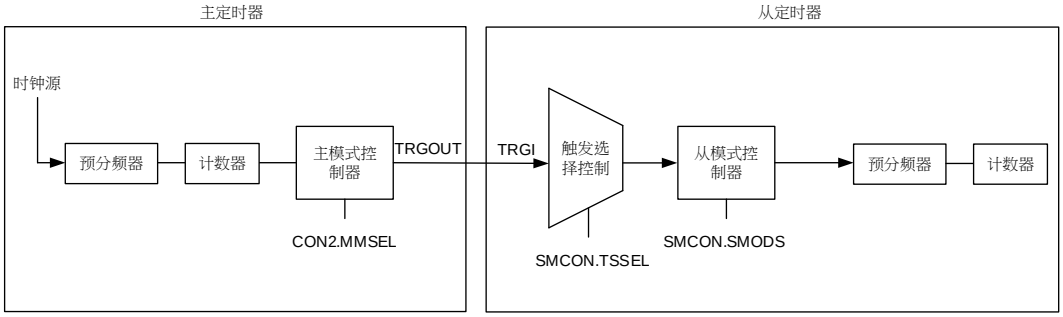


图 17-23　主/从定时器范例

17. 4. 15. 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(GP16C2T2)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 01010b(GP16C2T2)，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 **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 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，当计数器上数到 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。

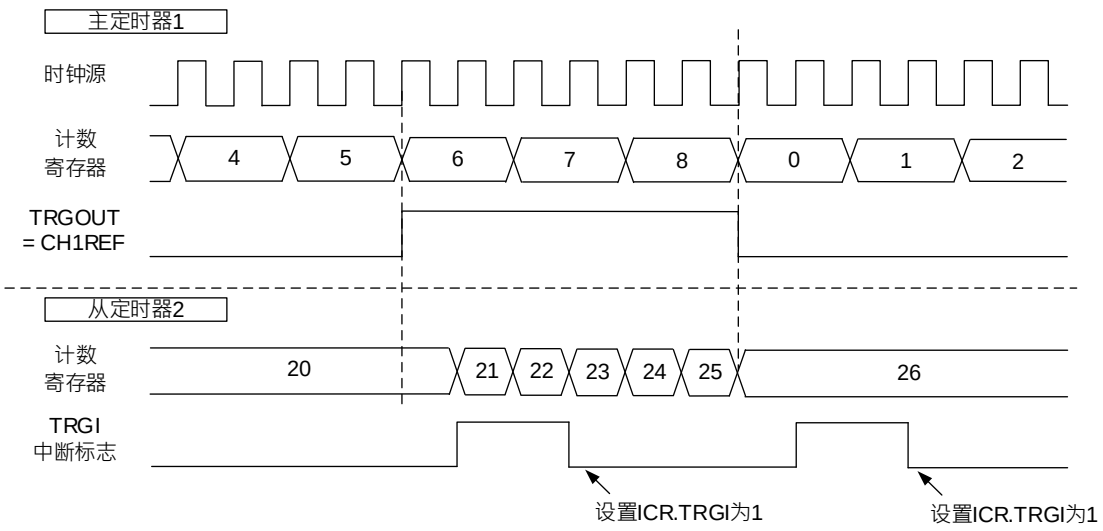


图 17-24　门控从定时器使用主定时器 CH1REF

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。

17. 4. 15. 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(GP16C2T2)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，当计数器上数到 2 后重载。
2. 设置 **GP16C2T_SMCON** 寄存器的 TSSEL1 位与 TSSEL2 位为 01010b(GP16C2T2)，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 **GP16C2T_AR** 寄存器的 ARV 位为 03h，当计数器上数到 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。

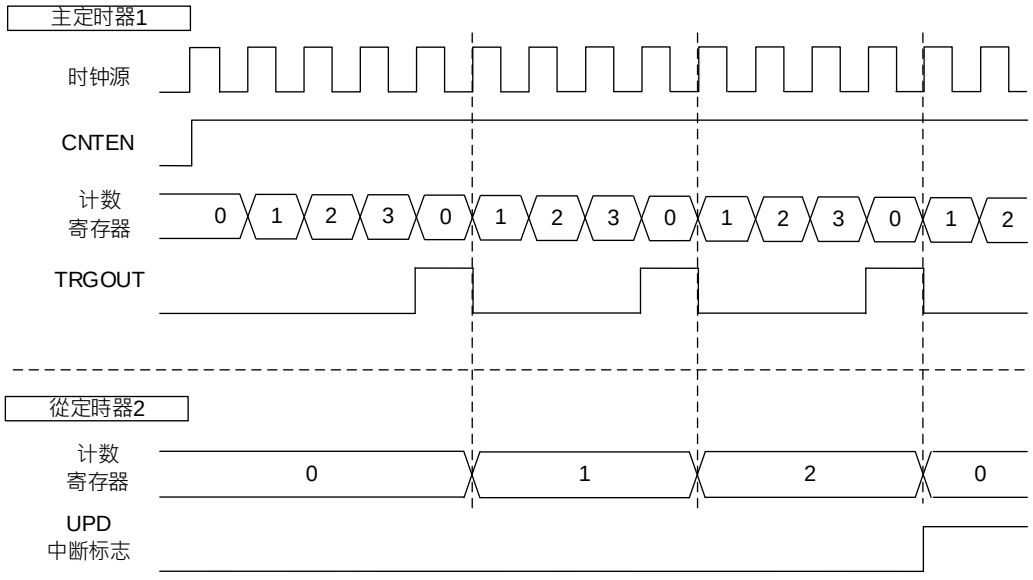


图 17-25　使用主定时器更新事件触发从定时器计数

17. 4. 15. 3 使用外部触发同步开启两个定时器

这个例子中当定时器 1(GP16C2T2)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)，参见图 17-26。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 01010b(GP16C2T2)，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 GP16C2T2 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。

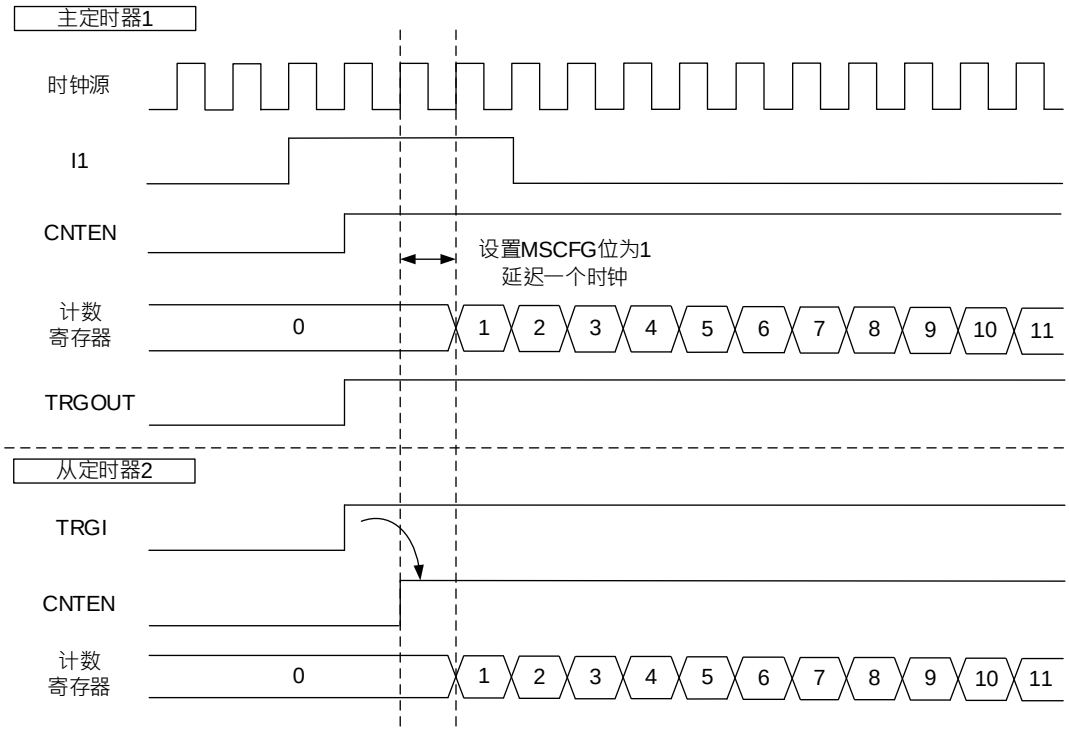


图 17-26 使用定时器 1 的 I1 输入触发定时器 1 和定时器 2

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。

17. 4. 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，选择将计数器继续正常工作或停止计数。

为了安全起见，当进入调试模式时，设置 **GP16C2T_CON1** 寄存器的 DBGSEL 位，选择继续输出电平或是输出关闭(切换成输入)。当输出关闭时可以通过 GPIO 控制器选择输出电平，若未设定则为悬空输入。

17. 5 特殊功能寄存器

17. 5. 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_OPTR	$005C_H$	R/W	输入选择寄存器

17. 5. 2 寄存器描述

17. 5. 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
																DBGSEL						DFCKSEL<1:0>		ARPEN				SPMEN	UERSEL	DISUE	CNTEN

—	Bit 31-16	—	—
DBGSEL	Bit 15	R/W	**调试模式下，通道状态选择** 在输出模式中，选择通道为输入或持续输出 0:强制为输入 1:保持当前配置 注：此位仅在 SYSCFG_CFG 寄存器的 DBGHEN位配置GP16C2T时有效
—	Bit 14-10	—	—
DFCKSEL	Bit 9-8	R/W	**时钟预分频器** 设置定时器的频率(INT_CLK)，死区产生器与数字滤波器(ETR、In)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	**自动重载缓冲功能开启** 发生更新事件时，将设定的值载入至影子寄存器中 0:GP16C2T_AR 寄存器未缓冲 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	**更新事件请求来源选择**

			设置更新事件(UPD)的来源 0:下列事件都会产生更新中断: －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
DISUE	Bit 1	R/W	**更新事件关闭** 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影子寄存器载入预装载值 －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 寄存器的影子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

17. 5. 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，经过死区时间后，CH1N=0 1:当 GOEN=0，经过死区时间后，CH1N=1 注:当设置 GP16C2T_BDCFG 寄存器中的 LOCKLVL 位为锁定级别 1、2 或 3 后，此位无法更改。
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:当设置 GP16C2T_BDCFG 寄存器中的 LOCKLVL 位为锁定级别 1、2 或 3 后，此位无法更改。
—	Bit 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(TRGOUT)与 ADC 输入 000:复位 - 设置 GP16C2T_SGE 寄存器中的 UPD 位为触发输出(TRGOUT)。如果复位由触发输入生成(从模式控制器配置为复位模式)，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(TRGOUT)，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(TRGOUT)。一个主定时器的更新事件可当作从定时器的预分频器时钟 011:比较脉冲 – 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(TRGOUT)会送出一个正脉冲 100:比较信号 – CH1REF 信号用于触发输出(TRGOUT) 101:比较信号 – CH2REF 信号用于触发输出(TRGOUT) 110:保留 111:保留
—	Bit 3	—	—
CCUSEL	Bit 2	R/W	**捕获或比较更新控制选择** 此功能只在有互补输出通道作用 0:在捕获或比较预装载时(CCPCEN =1)，只能通过 GP16C2T_SGE 寄存器的 SGCOM 位为 1 时更新 1:在捕获或比较预装载时(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	**捕获或比较预装载控制** 设置后只在换向事件(COM)，即对 GP16C2T_SGE 寄存器的 SGCOM 位置 1 或 TRGI 的上升沿时更新 0:GP16C2T_CCEP 寄存器中的 CCnEN、CCnNEN 和 GP16C2T_CHMRn 寄存器中的 CHnMOD 位预装载关闭 1:GP16C2T_CCEP 寄存器中的 CCnEN、CCnNEN 和 GP16C2T_CHMRn 寄存器中的 CHnMOD 位预装载开启

17. 5. 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	R/W	**触发选择2** 参照TSSEL1描述
—	Bit 19-8	—	—
MSCFG	Bit 7	R/W	**主/从模式** 0:写入0无效 1:延迟触发输入(TRGI)上的事件来允许当前计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。
TSSEL1	Bit 6-4	R/W	**触发选择 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 00110: I2 滤波后信号 00111:保留 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) 01100:内部触发 8 (IT8) 其他:内部触发 n (ITn) 注:此位需要在使用前设定(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** 000:从模式关闭 - 当 GP16C2T_CON1 寄存器 CNTEN 位为 1 时，分频器时钟由内部时钟提供 001:保留 010:保留 011:保留 100:复位模式 - 选中的触发输入(TRGI)的上升沿重新初始化计数器，并且产生一次更新事件 101: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 110:触发模式 - 计数器在触发输入 TRGI 的上升沿启动(但不复位)，只有计数器的启动是被控制 111: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 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

从定时器	IT4(TSSEL =01000)	IT5(TSSEL =01001)	IT6(TSSEL =01010)	IT7(TSSEL =01011)	IT8(TSSEL =01100)
GP16C2T1	GP32C4T	保留	GP16C2T2	GP16C2T3	GP16C2T4
GP16C2T2	GP32C4T	GP16C2T1	保留	GP16C2T3	GP16C2T4
GP16C2T3	GP32C4T	GP16C2T1	GP16C2T2	保留	GP16C2T4
GP16C2T4	GP32C4T	GP16C2T1	GP16C2T2	GP16C2T3	保留

表 17-1 GP16C2T 内部触发连接

17. 5. 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，硬件侦测信道 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，硬件侦测信道 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

17. 5. 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

17. 5. 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

17. 5. 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

17. 5. 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，标志位中断状态** 0:无发生中断 1:已发生中断
CH1OV	Bit 9	R	**通道 1 捕获溢出，标志位中断状态** 0:无发生中断 1:已发生中断
—	Bit 8	—	—
BRK	Bit 7	R	**刹车，标志位中断状态** 0:无发生中断 1:已发生中断
TRGI	Bit 6	R	**触发，标志位中断状态** 0:无发生中断 1:已发生中断
COM	Bit 5	R	**COM，标志位中断状态** 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，标志位中断状态** 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，标志位中断状态** 0:无发生中断 1:已发生中断
UPD	Bit 0	R	**更新，标志位中断状态** 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

17. 5. 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

17. 5. 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	T_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	T_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
SGCOM	Bit 5	T_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。当设置 CCPCEN 为 1，则可更新 CCnEN, CCnNEN 和 CHnMOD。 注:此位只有用作于有互补输出的通道
—	Bit 4-3	—	—
SGCH2	Bit 2	T_W1	**软件生成通道 2 捕获或比较事件**

			参照 SGCH1 描述
SGCH1	Bit 1	T_W1	**软件生成通道 1 捕获或比较事件** 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，则产生中断。 **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，若开启中断，则产生中断。
SGUPD	Bit 0	T_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)。

17. 5. 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
																	CH2MOD<2:0>			CH2PEN	CH2FEN	CC2SSEL<1:0>			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 31-15	—	—
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 或 I1 的双边沿检测
—	Bit 7	—	—
CH1MOD	Bit 6-4	R/W	**输出比较信道 1 模式** 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。 000:关闭 – 无作用

			001: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 1 010: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 0 011:匹配时设置翻转电平 – 当计数器(GP16C2T_COUNT)匹配 GP16C2T_CCVAL1 寄存器时，CH1REF 设置翻转电平(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) 100:强制低电平 – CH1REF 强制设置低电平 101:强制高电平 – CH1REF 强制设置高电平 110:PWM 模式 1 – 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。 111: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。
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影子寄存器中。 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 只在信道被配置成 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 或 I1 的双边沿检测

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 或 I1 的双边沿检测
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}，滤波器关闭 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 或 I1 的双边沿检测

17. 5. 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，CC1NP 影子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，CC1NE 影子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 影子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 影子位才会设置为预载值中新的值。

17. 5. 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
																CNTV <15:0>															

—	Bits 31-16	—	—
CNTV	Bits 15-0	R/W	**计数器数值**

17. 5. 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** 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数值载入影子寄存器中

17. 5. 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	**自动重载数值** 设置计数器的递增计数的边界重载值，设置数值为 0 时计数器停止计数

17. 5. 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，该位允许使用者设置比较寄存器的更新率(例如:预载到有效寄存器的周期性传输)，同样也可以设置更新中断生成率。每次当REPV_CNT的相关递减计数器递减至0，会产生更新事件，会从REPV值重新计数。因为只有当发生重复更新事件时，REPV_CNT才会重新载入REPV值，所以只有在发生下一次重复更新事件时，写入GP16C2T_REPAR寄存器的值才会生效。 在PWM模式下，(REPV+1)相当于: - 在边沿对齐模式下，(REPV+1)对应的是PWM的周期数

17. 5. 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。 如果在 GP16C2T_CHMR1 寄存器中的预载功能没有选中，CCRV1 中的值将被永久载入；否则每当发生更新事件，预载值将会复制到有效的捕获或比较寄存器中。有效捕获或比较寄存器的值将会与 GP16C2T_COUNT 中的值进行比较，并在 CH1 上输出。 **通道 CH1 配置为输入:** CCRV1为由上一个输入捕获事件(I1)发生时的计数器数值。

17. 5. 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** **信道CH2配置为输出:** CCRV2是捕获或比较寄存器的预装载值。 如果在GP16C2T_CHMR1寄存器中的预载功能没有选中，CCRV2中的值将被永久载入;否则每当发生更新事件，预载值将会复制到有效的捕获或比较寄存器中。有效捕获或比较寄存器的值将会与GP16C2T_COUNT中的值进行比较，并在CH2上输出。 **信道CH2配置为输入:** CCRV2为由上一个输入捕获事件(I2)发生时的计数器数值。

17. 5. 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
																GOEN	AOEN	BRKP	BRKEN	OFFSSR	OFFSSI	LOCKLVL<1:0>		DT<7:0>							

—	Bits 31-16	—	—
GOEN	Bit 15	R/W	**通道主要输出开启** 一旦刹车输入有效，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:CHn和CHnN输出关闭或强制为空闲状态。 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，则该位无法修改。
BRKP	Bit 13	R/W	**选择刹车极性** 0:刹车输入 BRK 为低电平有效 1:刹车输入 BRK 为高电平有效 注 1:当 GP16C2T_BDCFG 寄存器中的 LOCKLVL 位已被设置为锁定级别 1，则该位无法修改。
BRKEN	Bit 12	R/W	**开启刹车** 0:刹车输入(BRK 和 CCS 时钟失效事件)关闭 1:刹车输入(BRK 和 CCS 时钟失效事件)开启 注:当 GP16C2T_BDCFG 寄存器中的 LOCKLVL 位已被设置为锁定级别 1，则该位无

			法修改。
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:当定时器关闭时，输出关闭(CHn/CHnN 开启输出信号=0) 1:当定时器关闭时，当 CCnEN 为 1 或 CCnNEN 为 1 时，便将 CHn/CHnN 输出设为无效电平。然后设置 CHn/CHnN 开启输出信号为 1。 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** 此位在 GOEN 为 0 且设置为输出模式并具有互补输出的通道。 0:当定时器关闭时，输出关闭(CHn 或 CHnN 开启输出信号为 0) 1:当定时器关闭时，当 CCnEN 为 1 或 CCnNEN 为 1 时，便将 CHn/CHnN 输出强制为空闲电平。然后设置 CHn/CHnN 开启输出信号为 1。 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。
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 时，无法对 GP16C2T_BDCFG 寄存器中的 DT 位、BRKEN、BRKP 和 AOEN 位，以及 GP16C2T_CON2 寄存器中的 OISSx 和 OISSxN 位执行写操作。 10:锁定级别 2 时，无法对锁定级别 1 与 CH 极性位(GP16C2T_CCEP 寄存器中的 CCnPOL 与 CCnNPOL 位，只要相关信道由 CCnSSEL 配置为输出)以及 OFFSSR 和 OFFSSI 执行写操作。 11:锁定级别 3 =锁定级别 2 与 CH 控制位(GP16C2T_CHMRn 寄存器中的 CHnMOD 和 CHnPEN 位，只要相关信道由 CCnSSEL 配置

			为输出)执行写操作。 注:锁定配置位仅在复位后可写一次。一旦对 GP16C2T_BDCFG 寄存器中 LOCKLVL 位写入非 00 数值，其设置内容在下一个复位前都处于冻结状态。
DT	Bit 7-0	R/W	**死区延时** 设置值该位定义了互补输出之间插入的死区时间。DT 对应的就是该时间段。 DT[7:5]=0xx ⇒ DT=DT[7:0] x t_{dtg}，式中 $t_{dtg}=t_{DTS}$。 DT[7:5]=10x ⇒ DT=(64+DT[5:0])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，则该位无法修改

17. 5. 2. 20　输入选择寄存器(GP16C2T_OPTR)

输入选择寄存器(GP16C2T_OPTR)																														
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
																												CH2_RMP<1:0>		CH1_RMP<1:0>	

—	Bits 31-4	—	—
CH2_RMP	Bit 3-2	R/W	**CH2输入选择** 00:GP16C2T_CH2输入 其他:保留
CH1_RMP	Bit 1-0	R/W	**CH1 输入选择** 00:GP16C2T_CH1 输入 其他:CMP_OUT

第18章　基本定时器 16 位(BS16T)

18. 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。

18. 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:
  - ◇ 更新事件:计数器上溢，计数器初始化(通过软件)

18. 3 结构图

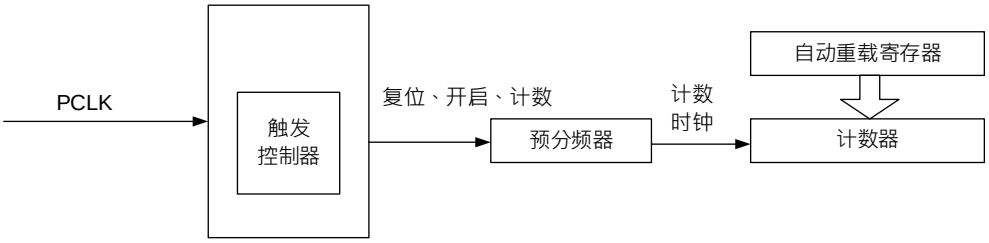


图 18-1　BS16T 定时器结构框图

18. 4 功能描述

18. 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 的重载值会被立即反应到影子寄存器中；而设置 ARPEN 位为 1 时，BS16T_AR 寄存器具有缓冲功能，当产生更新事件(UPD)时，BS16T_AR 寄存器的重载值才会被更新到影子寄存器中。

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 APB 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。

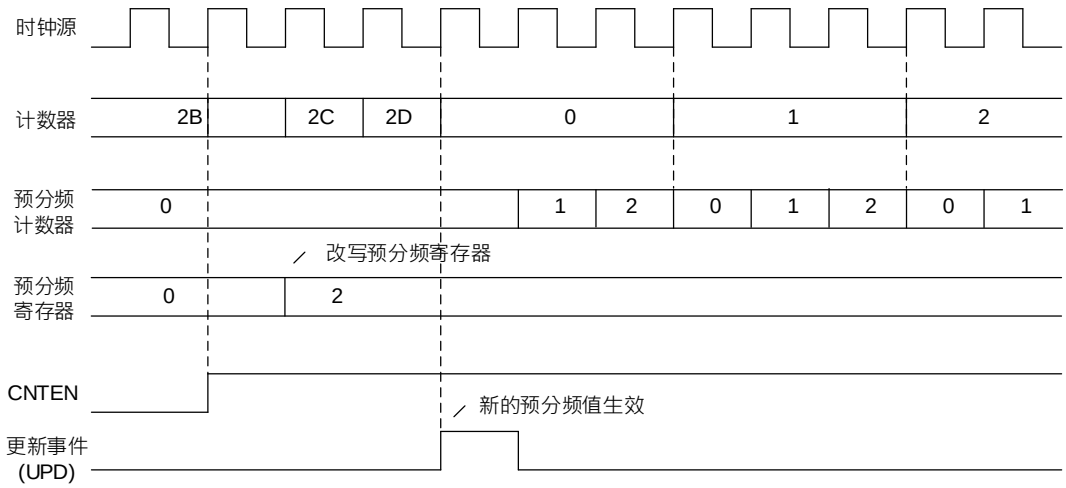


图 18-2 预分频值计数时序图

18. 4. 2 时钟源

计数器工作时钟可以选择内部时钟(INT_CLK)

18. 4. 2. 1 内部时钟源(INT_CLK)

BS16T_CON1 寄存器的 CNTEN 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。

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。

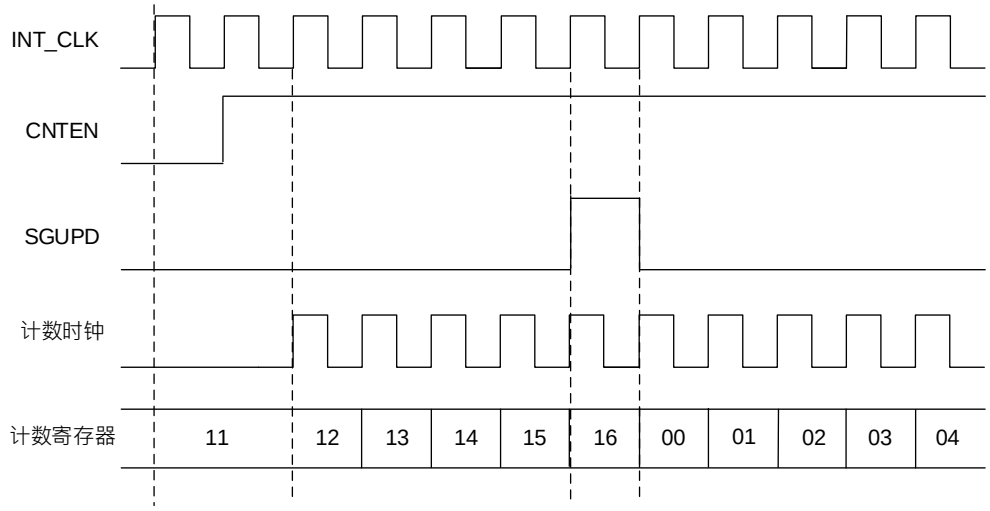


图 18-3 采用内部时钟计数

18. 4. 3 计数模式

18. 4. 3. 1 递增计数模式

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)产生更新事件。

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)更新影子寄存器。在设置 DISUE 位为 0 之前都不会产生更新事件(UPD)，计数器和预分频器都会重新从 0 开始计数。

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。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影子寄存器:

- 更新 **BS16T_AR** 寄存器数值到影子寄存器
- 更新 **BS16T_PRES** 寄存器数值到影子寄存器

下图为设置 **BS16T_AR** 寄存器为 0x16，预分频设为 2 分频时的计数器时序。

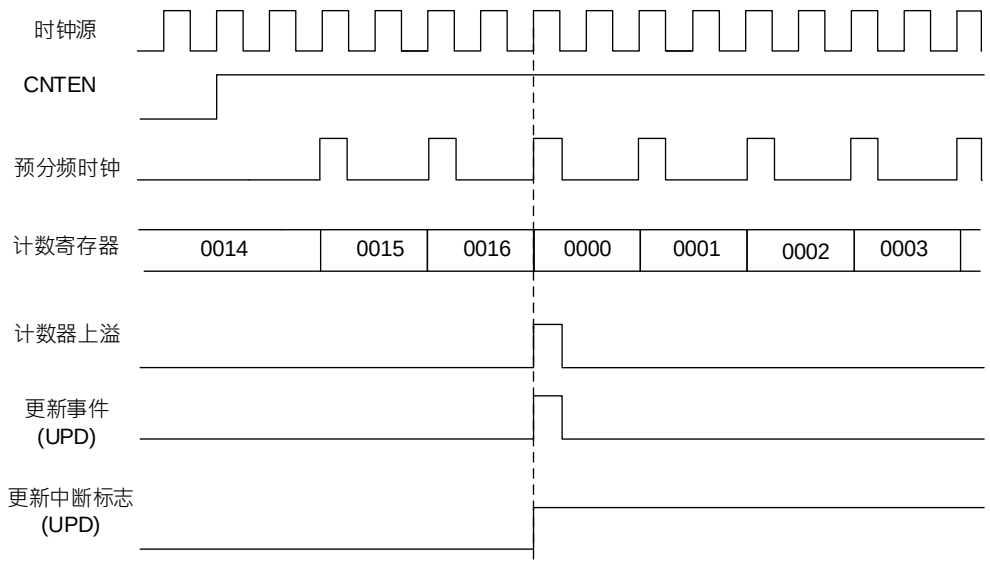


图 18-4　计数器递增计数时序图

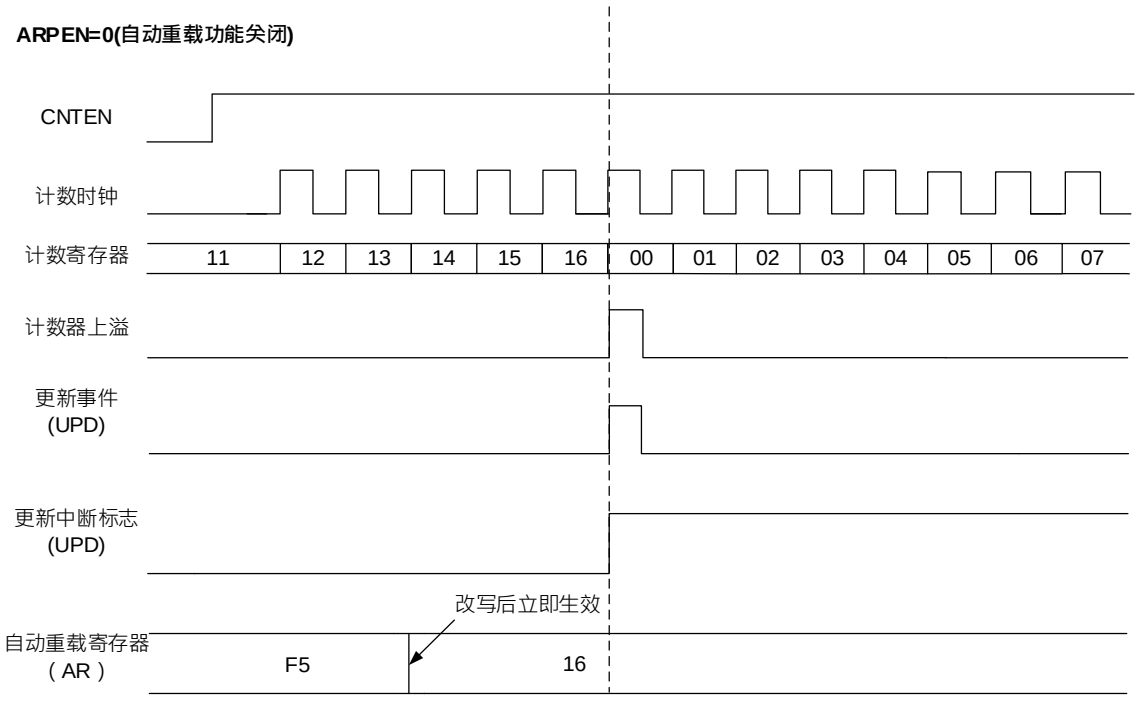


图 18-5　设置 ARPEN 位为 0 时计数器时序图

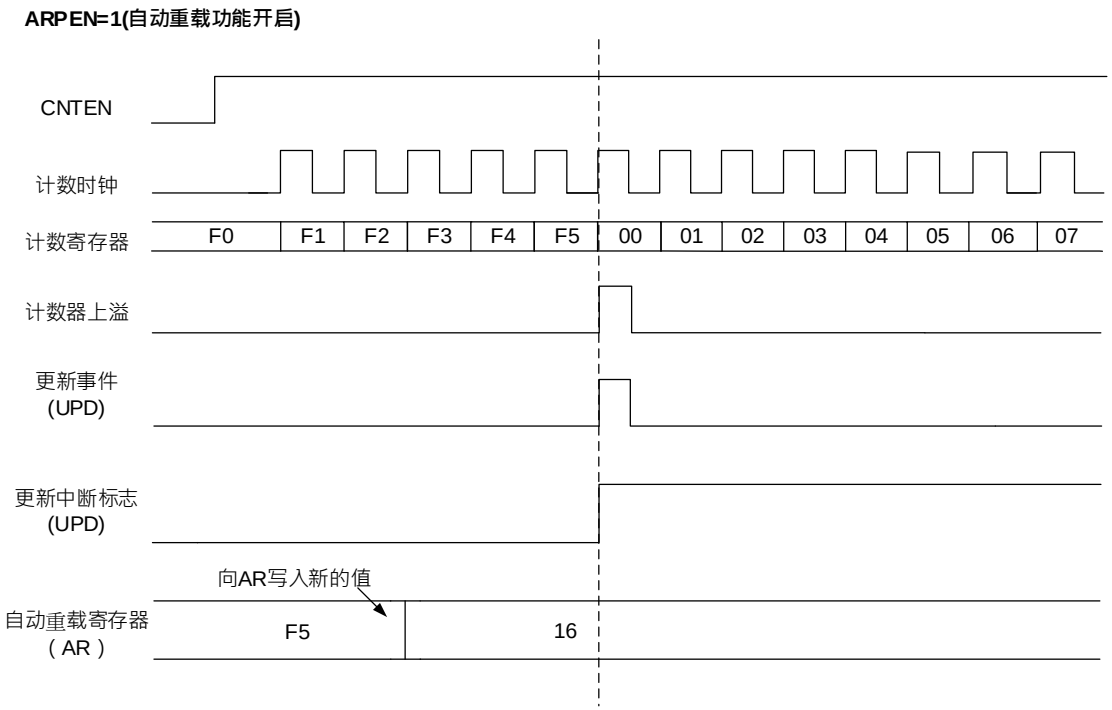


图 18-6　设置 ARPEN 位为 1 时计数器时序图

18. 4. 4　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。

18. 5 特殊功能寄存器

18. 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_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

18. 5. 2 寄存器描述

18. 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
																								ARPEN				SPMEN	UERSEL	DISUE	CNTEN

—	Bits 31-8	—	—
ARPEN	Bit 7	R/W	**自动重载缓冲功能开启** 发生更新事件时，将设定的值载入至影子寄存器中 0:BS16T_AR 寄存器未缓冲 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	**更新事件请求来源选择** 设置更新事件(UPD)的来源 0:下列事件都会产生更新中断: －计数器上溢 －设置 BS16T_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
DISUE	Bit 1	R/W	**更新事件关闭** 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影子寄存器加载预装载值 －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 与 BS16T_PRES 寄存器的影子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** 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设置 CNTEN 位为 1 0:计数器关闭 1:计数器开启

18. 5. 2. 2　　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

18. 5. 2. 3 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

18. 5. 2. 4 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

18. 5. 2. 5 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

18. 5. 2. 6 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，标志位中断状态** 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

18. 5. 2. 7 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

18. 5. 2. 8 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	T_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)。

18. 5. 2. 9 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
																CNTV<15:0>															

—	Bits 31-16	—	—
CNTV	Bits 15-0	R/W	**计数器数值**

18. 5. 2. 10 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数值载入影子寄存器中

18. 5. 2. 11 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	**自动重载数值** 设置计数器的递增计数的边界重载值，设置数值为 0 时计数器停止计数

第19章 独立看门狗(IWDT)

19. 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，且使用者无法通过软件关闭 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。

19. 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。

19. 3 结构图

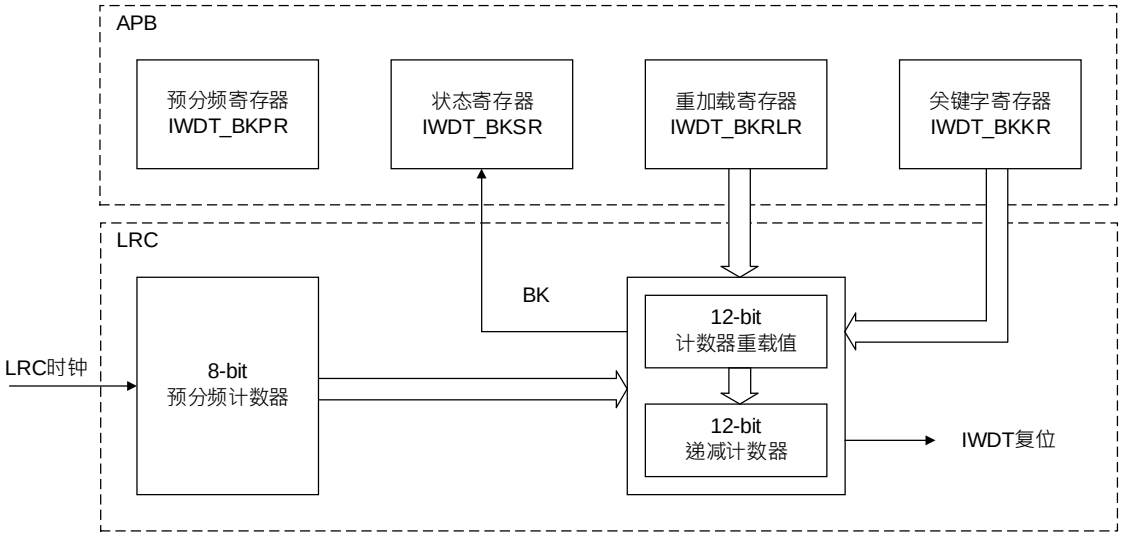


图 19-1　IWDT 架构图

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

19. 4 功能描述

19. 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)。

19. 4. 2 低功耗模式下的行为

使能 IWDT 后，则无法停止 IWDT。

19. 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，可以通过标志寄存器得知。

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。

19. 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。

19. 5 特殊功能寄存器

19. 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

19. 5. 2 寄存器描述

19. 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，读取为 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。

19. 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

19. 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。

19. 5. 2. 4 IWDT标志寄存器 (IWDT_BKFR)

此寄存器只允许使用 32 位存取。

IWDT 标志寄存器 (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	—	—

19. 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。

19. 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20章 窗口看门狗 (WWDT)

20. 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。

20. 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。

20. 3 结构图

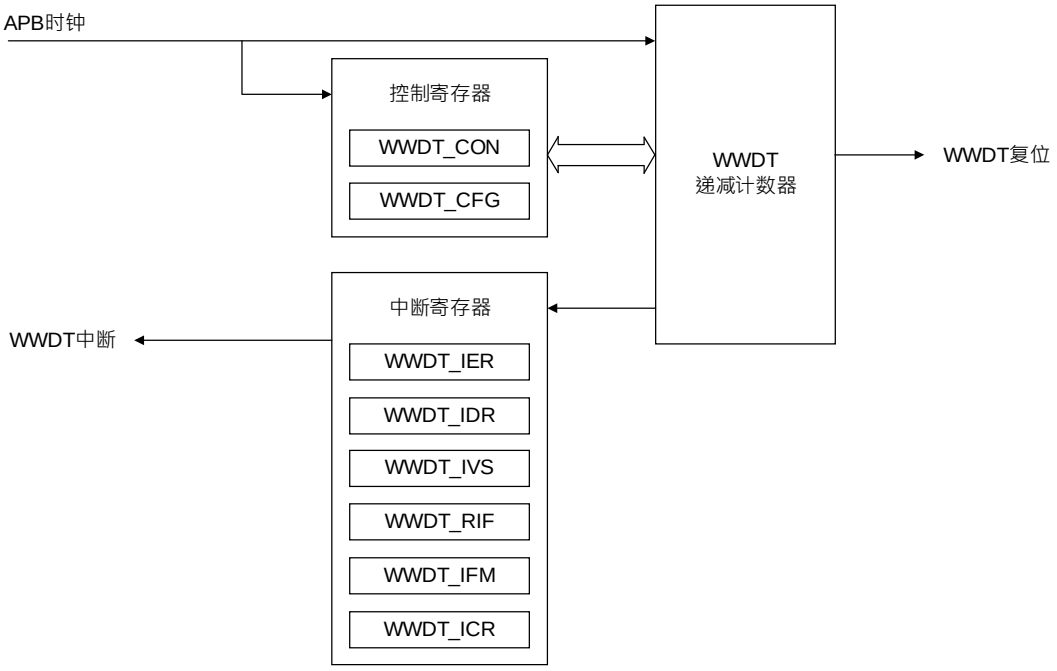


图 20-1　WWDT 架构图

20. 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。

20. 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。

20. 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。

20. 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。

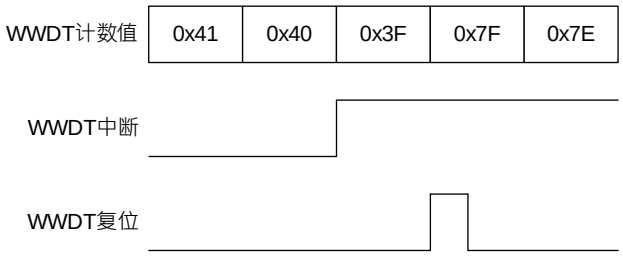


图 20-2 WWDT 中断示意图

20. 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

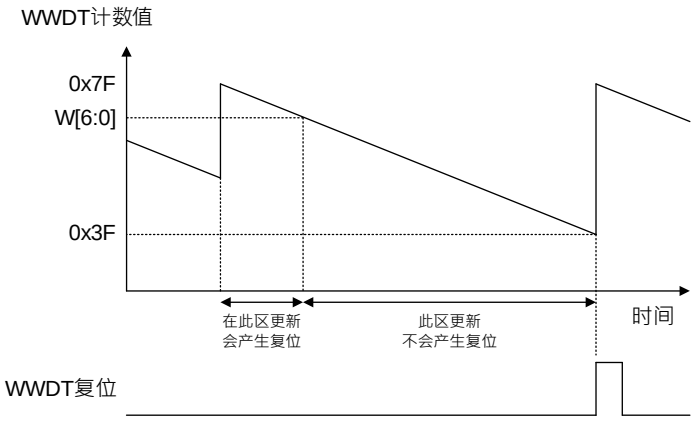


图 20-3 WWDT 时序图

20. 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。

20. 5 特殊功能寄存器

20. 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 中断功能有效位状态寄存器
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

20. 5. 2 寄存器描述

20. 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。

20. 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

20. 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

20. 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

20. 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

20. 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。

20. 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。

20. 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21章 实时时钟 (RTC)

21. 1 概述

Real Time Clock (RTC)可提供用户准确的时间以及日期信息，这些信息皆以 BCD 的格式存储在 RTC 控制寄存器内部，同时提供用户自定义设定闹铃功能与硬件自动定时校准功能。RTC 的时钟来源分为 2 种供用户选择，分别为外部 32.768 KHz 的晶体振荡器以及内部约 32 KHz 的 RC 振荡器。此外，RTC 也支持将系统从低功耗模式 STOP0 与 STOP1 唤醒的功能，在低功耗模式期间 RTC 仍会持续计时。

21. 2 特性

- 支持日历显示:年、月、日。
- 支持时间显示:星期、小时、分钟、秒。
- 日历与时间皆以 BCD 格式储存。
- 支持闰年侦测功能。
- 支持用户设置 RTC 计数器校准功能，最高支持±0.00745ppm 的高精度校准。
- 支持用户自行设定闹铃中断功能。
- 支持年、月、日、星期、小时、分钟、秒的翻转(Rollover)中断。
- 支持在低功耗 STOP0 与 STOP1 模式下持续计数。
- 支持用户自行配置睡眠计数器。
- 支持最长 16777216 秒(194 天)的睡眠计数。
- 支持侦测日历、时间是否已被清除。

21. 3 结构图

RTC 主要分为 2 个部分，第一个部分为 RTC 计数器，第二个部分为 RTC 时间存储与闹铃。RTC 计数器的部分主要负责计算日期以及时间，并提供将系统从低功耗 STOP0 与 STOP1 下唤醒的功能。而另一个部分则是提供日期以及时间的读取，同时根据用户所设定的闹钟发出中断。

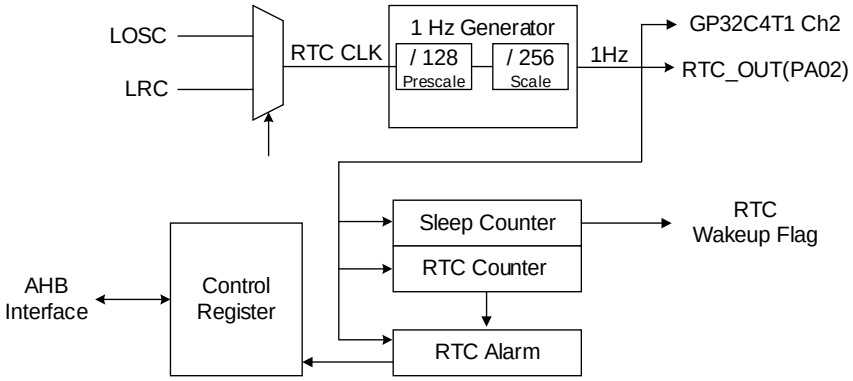


图 21-1　RTC 架构图

21. 4 功能描述

21. 4. 1　设定并开启RTC

在开启 RTC 计数之前，需确认 RTC 内已填入正确的时间与日期信息，同时 RTC 的 1Hz 信号产生器已被正确的配置。RTC 的配置步骤建议如下:

- 配置 **RTC_CTRL** 内的 RTCEN 数值为 0 关闭 RTC 计数，读取确认 **RTC_CTRL** 内的 RTCEN 数值为 0。
- 对 **RTC_CAL** 填入日期信息，格式必须为 BCD 码。
- 对 **RTC_TIME** 填入时间信息，格式必须为 BCD 码。
- 设定 **RTC_CTRL** 内的 CKSEL 选择 RTC 的时钟。
- 设定 **RTC_CTRL** 内的 PSCALE 与 SCALE 让计数器能够产生正确的 1Hz 信号。
- 若有校准相关需求，请参阅后续章节描述。
- 配置 **RTC_CTRL** 内的 RTCEN 数值为 1 开启 RTC 计数。需注意开启 RTC 计数功能以后，不再支持用户修改 **RTC_CAL** 与 **RTC_TIME** 内的内容。

21. 4. 2　读取RTC日期与时间

RTC 可通过读取 **RTC_CAL** 与 **RTC_TIME** 获取当前的日期与时间，**RTC_CAL** 与 **RTC_TIME** 的数值会在每一秒发生后同步进行更新。当系统进入低功耗 STOP0 或 STOP1 模式时，此时 1.5V 域的电源虽然不会被关闭，但是除了 LOSC 与 LRC 以外的时钟会被关闭，此时会无法更新 **RTC_CAL** 与 **RTC_TIME** 的数值，待系统被唤醒后才会更新日期与时间。

21. 4. 3 RTC校准

由于RTC时钟的来源并非准确的32.768 kHz，因此需要针对RTC所使用的时钟进行频率补偿，以确保能计算出准确的1秒。校准数值的计算方式如下:

- 方法1:在RTC不开启校准功能的情况下，直接量测RTC连续计时数日后与标准时间的误差值(ppm)。
- 方法2:利用RTC 1Hz信号触发GP32C4T1计数，利用Timer的计数数值推算出RTC 1Hz的误差值(ppm)。在使用此量测方式前，建议先关闭RTC校准功能。举例来说，当系统频率为8MHz，利用RTC 1Hz触发Timer计数，则Timer每一秒所计算出的数值会接近8000000，因此可利用8000000减去Timer的计数数值，即可推算出误差值(ppm)。在使用此方法时，需注意Timer所推算出的误差值会受限于系统时钟的精准度，因此建议在使用此方式进行校准时，系统时钟不要使用内部RC时钟。

当计算完RTC 1Hz的误差值(ppm)以后，将误差值(ppm)先后乘上32768(RTC时钟产生1Hz的计数周期)与4096(RTC校准周期)，即可得到RTC的校准数值。将此数值填入**RTC_CALIB**内的CALIB，并设定MODE选择使用RTC正校准模式或是RTC负校准模式，最后再设定CALIBEN开启硬件自动定时校准功能。

21. 4. 3. 1 RTC校准示例1(实际频率高于预期频率)

预期频率:32768 Hz。

实际频率:32770.5 Hz。

准数值:(32770.5 – 32768) X 4096 = +10240。

校准步骤如下:

1. 设定 **RTC_CALIB** 内CALIB的数值为10240。
2. 设定**RTC_CALIB**内MODE为0开启正校准功能。
3. 设定**RTC_CALIB**内CALIB开启RTC校准功能。

21. 4. 3. 2 RTC校准示例2(实际频率低于预期频率)

预期频率:32768 Hz。

实际频率:32766.5 Hz。

准数值:(32766.5 – 32768) X 4096 = -6144。

校准步骤如下:

1. 设定 **RTC_CALIB** 内CALIB的数值为6144。
2. 设定 **RTC_CALIB** 内MODE为1开启负校准功能。
3. 设定**RTC_CALIB** 内CALIB开启RTC校准功能。

21. 4. 4 RTC 唤醒计数器

RTC的唤醒计数器可以自行设定**RTC_WKUP**内部的WKSEL来决定开启计数的时间点，当用

户设定 WKSEL 为 0x0 时代表不使用 RTC 的唤醒计数器；当设定 WKSEL 为 0x1 时，此时唤醒计数器可当成一般计数器使用，用户可以设定 WKCAL 来决定要计数多少秒，当计数完成时可发出中断信号通知；当设定 WKSEL 为 0x2 时，唤醒计数器会在系统进入低功耗模式以后开始计数，并于计数完成时唤醒系统。可设定至 16777216 秒，即 194 天。

当设定 RTC 为进入低功耗模式以后开始计数时，需注意 RTC 在低功耗模式下仅支持 STOP0 与 STOP1 模式，当系统进入低功耗 STANDBY 或是 SHUTDOWN 模式时，便不再支持 RTC 计数。

21. 4. 5 RTC闹铃

RTC 支持用户自行配置闹钟功能。用户可设定 **RTC_ALEN** 开启各类型的 RTC 闹钟，并于 **RTC_ALTIME** 以及 **RTC_ALCAL** 设定触发闹钟的时间，当 RTC 时间与日期符合 **RTC_ALTIME** 与 **RTC_ALCAL** 内的设定值时便会触发中断事件。

21. 5 特殊功能寄存器

21. 5. 1 寄存器列表

RTC 寄存器列表			
名称	偏移地址	类型	描述
RTC_CTRL	0000_H	R/W	RTC 控制寄存器
RTC_WKUP	0004_H	R/W	RTC 唤醒寄存器
RTC_TIME	0008_H	R/W	RTC 时间寄存器
RTC_CAL	$000C_H$	R/W	RTC 日期寄存器
RTC_CALIB	0010_H	R/W	RTC 校准寄存器
RTC_ALTIME	0020_H	R/W	RTC 时间闹钟寄存器
RTC_ALCAL	0024_H	R/W	RTC 日期闹钟寄存器
RTC_ALEN	0028_H	R/W	RTC 闹钟开关寄存器
RTC_IER	0030_H	W1	RTC 中断开启寄存器
RTC_IDR	0034_H	W1	RTC 中断关闭寄存器
RTC_IVS	0038_H	R	RTC 中断功能有效状态寄存器
RTC_RIF	$003C_H$	R	RTC 原始中断状态寄存器
RTC_IFM	0040_H	R	RTC 中断标志位状态寄存器
RTC_ICR	0044_H	C_W1	RTC 中断清除寄存器
RTC_STA	0050_H	R	RTC 状态寄存器

21. 5. 2 寄存器描述

21. 5. 2. 1 RTC 控制寄存器(RTC_CTRL)

RTC 控制寄存器(RTC_CTRL)																													
偏移地址:0x00																													
复位值:0x00FF 7F00																													
31	30	29	28	27	26	25	24	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<7:0>								—	PSCALE<6:0>							—	—	—	—	CKSEL<1:0>		—	RTCEN

—	Bit 31-24	—	—
SCALE	Bits 23-16	R/W	**RTC 计数分频** SCALE 控制 RTC 时钟的分频。当 PSCALE 内的数值计数至 0 时，SCALE 的数值自动减 1。此分频计数器所能计数的最大值为 (SCALE+1)。
—	Bit 15	—	—
PSCALE	Bits 14-8	R/W	**RTC 计数预分频** PSCALE 控制 RTC 时钟的预分频计数器。当 PSCALE 计数至 0 时，SCALE 的数值自动减 1。此分频计数器所能计数的最大值为 (PSCALE+1)。
—	Bits 7-4	—	—
CKSEL	Bits 3-2	R/W	**RTC 时钟源选择** 配置 CKSEL 选择 RTC 时钟的来源。 0x1:选择 LOSC 当作 RTC 时钟源 0x2:选择 LRC 当作 RTC 时钟源。
—	Bit 1	—	—
RTCEN	Bit 0	W1	**RTC 启动** 配置此位开启 RTC。当 RTCEN 为 1 以后，不再允许修改 CKSEL、PSCALE 与 SCALE 的数值。

21. 5. 2. 2 RTC 唤醒寄存器(RTC_WKUP)

RTC 唤醒寄存器(RTC_WKUP)																															
偏移地址:0x04																															
复位值:0x00FF 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
						WKSEL<1:0>		WKCAL<23:0>																							

—	Bits 31-26	—	—
WKSEL	Bits 25-24	R/W	**RTC 唤醒计数启动事件选择** 0x0:关闭 RTC 唤醒计数器。 0x1:软件启动 RTC 唤醒计数器。 0x2:进入低功耗模式后启动 RTC 唤醒计数器。
WKCAL	Bits 23-0	R/W	**RTC 唤醒计数值** 以秒为单位进行配置，最多可配置(WKCAL+1)秒。

21. 5. 2. 3 RTC 时间寄存器(RTC_TIME)

RTC 时间寄存器(RTC_TIME)																															
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
					WEEK<2:0>					HOUR_T<1:0>		HOUR_U<3:0>					MIN_T<2:0>			MIN_U<3:0>				SEC_T<2:0>			SEC_U<3:0>			

—	Bits 31-27	—	—
WEEK	Bits 26-24	R/W	**星期值** 以 BCD 格式储存，数值介于 1 ~ 7，代表周一至周日。
—	Bits 23-22	—	—
HOUR_T	Bits 21-20	R/W	**小时(十位数)** 以 BCD 格式储存，数值介于 0 ~ 2。
HOUR_U	Bits 19-16	R/W	**小时(个位数)** 以 BCD 格式储存，数值介于 0 ~ 9。

—	Bit 15	—	—
MIN_T	Bits 14-12	R/W	**分(十位数)** 以 BCD 格式储存，数值介于 0 ~ 5。
MIN_U	Bits 11-8	R/W	**分(个位数)** 以 BCD 格式储存，数值介于 0 ~ 9。
—	Bit 7	—	—
SEC_T	Bits 6-4	R/W	**秒(十位数)** 以 BCD 格式储存，数值介于 0 ~ 5。
SEC_U	Bits 3-0	R/W	**秒(个位数)** 以 BCD 格式储存，数值介于 0 ~ 9。

21. 5. 2. 4 RTC 日期寄存器(RTC_CAL)

RTC 日期寄存器(RTC_CAL)

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
—	—	—	—	—	—	—	—	YEAR_T<3:0>				YEAR_U<3:0>				—	—	—	MON_T	MON_U<3:0>				—	—	DATE_T<1:0>		DATE_U<3:0>			

—	Bits 31-24	—	—
YEAR_T	Bits 23-20	R/W	**年(十位数)** 以 BCD 格式储存，数值介于 0 ~ 9。
YEAR_U	Bits 19-16	R/W	**年(个位数)** 以 BCD 格式储存，数值介于 0 ~ 9。
—	Bits 15-13	—	—
MON_T	Bit 12	R/W	**月(十位数)** 以 BCD 格式储存，数值介于 0 ~ 1。
MON_U	Bits 11-8	R/W	**月(个位数)** 以 BCD 格式储存，数值介于 0 ~ 9。
—	Bits 7-6	—	—
DATE_T	Bits 5-4	R/W	**日(十位数)** 以 BCD 格式储存，数值介于 0 ~ 3。
DATE_U	Bits 3-0	R/W	**日(个位数)** 以 BCD 格式储存，数值介于 0 ~ 9。

21. 5. 2. 5 RTC 校准寄存器(RTC_CALIB)

RTC 校准寄存器(RTC_CALIB)																													
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
CALIB <15:0>																														MODE	CALIBEN

CALIB	Bits 31-16	R/W	**RTC 1Hz 校准值** RTC 1Hz 校准值。此数值的计算方式为: 32768 x LOSC/LRC 误差值(ppm) x 4096。举例来说，若 LOSC 的误差为 15ppm，则 CALIB 的数值应为 32768 x 15 x 10^{-6} x 4096 = 2013。需注意的是，校准值仅有在 RTC 校准功能未被开启前才允许修改。
—	Bits 15-2	—	—
MODE	Bits 1	R/W	**RTC 校准模式** 校准模式仅有在 RTC 校准功能未被开启前才允许修改。 0x0:正校准。当 RTC 时钟频率高于 32.768KHz 时使用此模式。 0x1:负校准。当 RTC 时钟频率低于 32.768KHz 时使用此模式。
CALIBEN	Bits 0	R/W	**RTC 校准开关** 设定为 1 开启 RTC 校准功能。开启校准功能以后，RTC 依据 CALIB 内的数值自动定期对 RTC 1Hz 计数器校准。

21. 5. 2. 6 RTC 时间闹钟寄存器(RTC_ALTIME)

RTC 时间闹钟寄存器(RTC_ALTIME)																												
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
—	—	—	—	—	WEEK<2:0>			—	—	HOUR_T<1:0>		HOUR_U<3:0>				—	MIN_T<2:0>			MIN_U<3:0>				—	SEC_T<2:0>			SEC_U<3:0>			

—	Bits 31-27	—	—
WEEK	Bits 26-24	R/W	**星期值闹钟** 以 BCD 格式储存，数值介于 1 ~ 7，代表周一至周日。
—	Bits 23-22	—	—
HOUR_T	Bits 21-20	R/W	**小时(十位数)闹钟** 以 BCD 格式储存，数值介于 0 ~ 2。
HOUR_U	Bits 19-16	R/W	**小时(个位数)闹钟** 以 BCD 格式储存，数值介于 0 ~ 9。
—	Bit 15	—	—
MIN_T	Bits 14-12	R/W	**分(十位数)闹钟** 以 BCD 格式储存，数值介于 0 ~ 5。
MIN_U	Bits 11-8	R/W	**分(个位数)闹钟** 以 BCD 格式储存，数值介于 0 ~ 9。
—	Bit 7	—	—
SEC_T	Bits 6-4	R/W	**秒(十位数)闹钟** 以 BCD 格式储存，数值介于 0 ~ 5。
SEC_U	Bits 3-0	R/W	**秒(个位数)闹钟** 以 BCD 格式储存，数值介于 0 ~ 9。

21. 5. 2. 7 RTC日期闹钟寄存器(RTC_ALCAL)

RTC 日期闹钟寄存器 (RTC_ALCAL)

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
—	—	—	—	—	—	—	—	YEAR_T<3:0>				YEAR_U<3:0>				—	—	—	MON_T	MON_U<3:0>				—	—	DATE_T<1:0>		DATE_U<3:0>			

—	Bits 31-24	—	—
YEAR_T	Bits 23-20	R/W	**年(十位数)闹钟** 以 BCD 格式储存，数值介于 0 ~ 9。
YEAR_U	Bits 19-16	R/W	**年(个位数)闹钟** 以 BCD 格式储存，数值介于 0 ~ 9。
—	Bits 15-13	—	—
MON_T	Bit 12	R/W	**月(十位数)闹钟** 以 BCD 格式储存，数值介于 0 ~ 1。
MON_U	Bits 11-8	R/W	**月(个位数)闹钟** 以 BCD 格式储存，数值介于 0 ~ 9。
—	Bits 7-6	—	—
DATE_T	Bits 5-4	R/W	**日(十位数)闹钟** 以 BCD 格式储存，数值介于 0 ~ 3。
DATE_U	Bits 3-0	R/W	**日(个位数)闹钟** 以 BCD 格式储存，数值介于 0 ~ 9。

21. 5. 2. 8 RTC 闹钟开关寄存器(RTC_ALEN)

RTC 闹钟开关寄存器(RTC_ALEN)																												
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
																									YEAR	MONTH	DATE	WEEK	HOUR	MIN	SEC

—	Bits 31-7	—	—
YEAR	Bit 6	R/W	**年闹钟开关** 设定为 1 开启年闹钟。当 RTC_CAL 内年的数值与 RTC_ALCAL 内年的数值相同时可触发 RTC 中断。
MONTH	Bit 5	R/W	**月闹钟开关** 设定为 1 开启月闹钟。当 RTC_CAL 内月的数值与 RTC_ALCAL 内月的数值相同时可触发 RTC 中断。
DATE	Bit 4	R/W	**日闹钟开关** 设定为 1 开启日闹钟。当 RTC_CAL 内日的数值与 RTC_ALCAL 内日的数值相同时可触发 RTC 中断。
WEEK	Bit 3	R/W	**星期闹钟开关** 设定为 1 开启星期闹钟。当 RTC_TIME 内星期的数值与 RTC_ALTIME 内星期的数值相同时可触发 RTC 中断。
HOUR	Bit 2	R/W	**小时闹钟开关** 设定为 1 开启小时闹钟。当 RTC_TIME 内小时的数值与 RTC_ALTIME 内小时的数值相同时可触发 RTC 中断。
MIN	Bit 1	R/W	**分闹钟开关** 设定为 1 开启分闹钟。当 RTC_TIME 内分的数值与 RTC_ALTIME 内分的数值相同时可触发 RTC 中断。
SEC	Bit 0	R/W	**秒闹钟开关** 设定为 1 开启秒闹钟。当 RTC_TIME 内秒的数值与 RTC_ALTIME 内秒的数值相同时可触

			发 RTC 中断。

21. 5. 2. 9 RTC 中断开启寄存器(RTC_IER)

RTC 中断开启寄存器(RTC_IER)																													
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
															WKTM	F1HZ	RYEAR	RMON	RDATE	RWEEK	RHOUR	RMIN	RSEC	AMALL	AYEAR	AMONTH	ADATE	AWEEK	AHOUR	AMIN	ASEC

—	Bits 31-17	—	—
WKTM	Bit 16	W1	**开启 RTC 唤醒计数器计数完毕中断功能** 此位设置时，开启中断功能。
F1HZ	Bit 15	W1	**开启 RTC 1Hz 中断功能** 此位设置时，开启中断功能。
RYEAR	Bit 14	W1	**开启 RTC 跨世纪中断功能** 此位设置时，开启中断功能。
RMON	Bit 13	W1	**开启 RTC 跨年中断功能** 此位设置时，开启中断功能。
RDATE	Bit 12	W1	**开启 RTC 跨月中断功能** 此位设置时，开启中断功能。
RWEEK	Bit 11	W1	**开启 RTC 跨周中断功能** 此位设置时，开启中断功能。
RHOUR	Bit 10	W1	**开启 RTC 跨日中断功能** 此位设置时，开启中断功能。
RMIN	Bit 9	W1	**开启 RTC 跨小时中断功能** 此位设置时，开启中断功能。
RSEC	Bit 8	W1	**开启 RTC 跨分中断功能** 此位设置时，开启中断功能。
AMALL	Bit 7	W1	**开启 RTC 全闹钟满足中断功能** 此位设置时，开启中断功能。
AYEAR	Bit 6	W1	**开启年闹钟中断功能** 此位设置时，开启中断功能。
AMONTH	Bit 5	W1	**开启月闹钟中断功能** 此位设置时，开启中断功能。
ADATE	Bit 4	W1	**开启日闹钟中断功能**

			此位设置时，开启中断功能。
AWEEK	Bit 3	W1	**开启星期闹钟中断功能** 此位设置时，开启中断功能。
AHOUR	Bit 2	W1	**开启小时闹钟中断功能** 此位设置时，开启中断功能。
AMIN	Bit 1	W1	**开启分闹钟中断功能** 此位设置时，开启中断功能。
ASEC	Bit 0	W1	**开启秒闹钟中断功能** 此位设置时，开启中断功能。

21. 5. 2. 10 RTC 中断关闭寄存器(RTC_IDR)

RTC 中断关闭寄存器(RTC_IDR)

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
|	|	|	|	|	|	|	|	|	|	|	|	|	|	|	WKTM	F1HZ	RYEAR	RMON	RDATE	RWEEK	RHOUR	RMIN	RSEC	AMALL	AYEAR	AMONTH	ADATE	AWEEK	AHOUR	AMIN	ASEC

—	Bits 31-17	—	—
WKTM	Bit 16	W1	**关闭 RTC 唤醒计数器计数完毕中断功能** 此位设置时，关闭中断功能。
F1HZ	Bit 15	W1	**关闭 RTC 1Hz 中断功能** 此位设置时，关闭中断功能。
RYEAR	Bit 14	W1	**关闭 RTC 跨世纪中断功能** 此位设置时，关闭中断功能。
RMON	Bit 13	W1	**关闭 RTC 跨年中断功能** 此位设置时，关闭中断功能。
RDATE	Bit 12	W1	**关闭 RTC 跨月中断功能** 此位设置时，关闭中断功能。
RWEEK	Bit 11	W1	**关闭 RTC 跨周中断功能** 此位设置时，关闭中断功能。
RHOUR	Bit 10	W1	**关闭 RTC 跨日中断功能** 此位设置时，关闭中断功能。
RMIN	Bit 9	W1	**关闭 RTC 跨小时中断功能** 此位设置时，关闭中断功能。
RSEC	Bit 8	W1	**关闭 RTC 跨分中断功能**

			此位设置时，关闭中断功能。
AMALL	Bit 7	W1	**关闭 RTC 全闹钟满足中断功能** 此位设置时，关闭中断功能。
AYEAR	Bit 6	W1	**关闭年闹钟中断功能** 此位设置时，关闭中断功能。
AMONTH	Bit 5	W1	**关闭月闹钟中断功能** 此位设置时，关闭中断功能。
ADATE	Bit 4	W1	**关闭日闹钟中断功能** 此位设置时，关闭中断功能。
AWEEK	Bit 3	W1	**关闭星期闹钟中断功能** 此位设置时，关闭中断功能。
AHOUR	Bit 2	W1	**关闭小时闹钟中断功能** 此位设置时，关闭中断功能。
AMIN	Bit 1	W1	**关闭分闹钟中断功能** 此位设置时，关闭中断功能。
ASEC	Bit 0	W1	**关闭秒闹钟中断功能** 此位设置时，关闭中断功能。

21. 5. 2. 11 RTC中断功能有效状态寄存器(RTC_IVS)

RTC 中断功能有效状态寄存器(RTC_IVS)

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
															WKTM	F1HZ	RYEAR	RMON	RDATE	RWEEK	RHOUR	RMIN	RSEC	AMALL	AYEAR	AMONTH	ADATE	AWEEK	AHOUR	AMIN	ASEC

—	Bits 31-17	—	—
WKTM	Bit 16	R	**RTC 唤醒计数器计数完毕中断功能开启/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。
F1HZ	Bit 15	R	**RTC 1Hz 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。
RYEAR	Bit 14	R	**RTC 跨世纪中断功能开启/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。
RMON	Bit 13	R	**RTC 跨年中断功能开启/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。
RDATE	Bit 12	R	**RTC 跨月中断功能开启/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。
RWEEK	Bit 11	R	**RTC 跨周中断功能开启/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。
RHOUR	Bit 10	R	**RTC 跨日中断功能开启/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。
RMIN	Bit 9	R	**RTC 跨小时中断功能开启/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。
RSEC	Bit 8	R	**RTC 跨分中断功能开启/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。
AMALL	Bit 7	R	**RTC 全闹钟满足中断功能开启/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。
AYEAR	Bit 6	R	**年闹钟中断功能开启/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。
AMONTH	Bit 5	R	**月闹钟中断功能开启/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。
ADATE	Bit 4	R	**日闹钟中断功能开启/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。
AWEEK	Bit 3	R	**星期闹钟中断功能开启/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。
AHOUR	Bit 2	R	**小时闹钟中断功能开启/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。
AMIN	Bit 1	R	**分闹钟中断功能开启/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。
ASEC	Bit 0	R	**秒闹钟中断功能开启/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。

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

21. 5. 2. 12 RTC原始中断状态寄存器(RTC_RIF)

RTC 原始中断状态寄存器(RTC_RIF)

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
															WKTM	F1HZ	RYEAR	RMON	RDATE	RWEEK	RHOUR	RMIN	RSEC	AMALL	AYEAR	AMONTH	ADATE	AWEEK	AHOUR	AMIN	ASEC

—	Bits 31-17	—	—
WKTM	Bit 16	R	**RTC 唤醒计数器计数完毕，原始中断状态** 0:无发生中断。 1:已发生中断。
F1HZ	Bit 15	R	**RTC 1Hz，原始中断状态** 0:无发生中断。 1:已发生中断。
RYEAR	Bit 14	R	**RTC 跨世纪，原始中断状态** 0:无发生中断。 1:已发生中断。
RMON	Bit 13	R	**RTC 跨年，原始中断状态** 0:无发生中断。 1:已发生中断。
RDATE	Bit 12	R	**RTC 跨月，原始中断状态** 0:无发生中断。 1:已发生中断。
RWEEK	Bit 11	R	**RTC 跨周，原始中断状态** 0:无发生中断。 1:已发生中断。

RHOUR	Bit 10	R	**RTC 跨日，原始中断状态** 0:无发生中断。 1:已发生中断。
RMIN	Bit 9	R	**RTC 跨小时，原始中断状态** 0:无发生中断。 1:已发生中断。
RSEC	Bit 8	R	**RTC 跨分，原始中断状态** 0:无发生中断。 1:已发生中断。
AMALL	Bit 7	R	**RTC 全闹钟满足，原始中断状态** 0:无发生中断。 1:已发生中断。
AYEAR	Bit 6	R	**年闹钟，原始中断状态** 0:无发生中断。 1:已发生中断。
AMONTH	Bit 5	R	**月闹钟，原始中断状态** 0:无发生中断。 1:已发生中断。
ADATE	Bit 4	R	**日闹钟，原始中断状态** 0:无发生中断。 1:已发生中断。
AWEEK	Bit 3	R	**星期闹钟，原始中断状态** 0:无发生中断。 1:已发生中断。
AHOUR	Bit 2	R	**小时闹钟，原始中断状态** 0:无发生中断。 1:已发生中断。
AMIN	Bit 1	R	**分闹钟，原始中断状态** 0:无发生中断。 1:已发生中断。
ASEC	Bit 0	R	**秒闹钟，原始中断状态** 0:无发生中断。 1:已发生中断。

21. 5. 2. 13 RTC中断标志位状态寄存器(RTC_IFM)

RTC 中断标志位状态寄存器(RTC_IFM)																														
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
															WKTM	F1HZ	RYEAR	RMON	RDATE	RWEEK	RHOUR	RMIN	RSEC	AMALL	AYEAR	AMON	ADATE	AWEEK	AHOUR	AMIN	ASEC

—	Bits 31-17	—	—
WKTM	Bit 16	R	**RTC 唤醒计数器计数完毕，标志位中断状态** 0:无发生中断。 1:已发生中断。
F1HZ	Bit 15	R	**RTC 1Hz，标志位中断状态** 0:无发生中断。 1:已发生中断。
RYEAR	Bit 14	R	**RTC 跨世纪，标志位中断状态** 0:无发生中断。 1:已发生中断。
RMON	Bit 13	R	**RTC 跨年，标志位中断状态** 0:无发生中断。 1:已发生中断。
RDATE	Bit 12	R	**RTC 跨月，标志位中断状态** 0:无发生中断。 1:已发生中断。
RWEEK	Bit 11	R	**RTC 跨周，标志位中断状态** 0:无发生中断。 1:已发生中断。
RHOUR	Bit 10	R	**RTC 跨日，标志位中断状态** 0:无发生中断。 1:已发生中断。
RMIN	Bit 9	R	**RTC 跨小时，标志位中断状态** 0:无发生中断。 1:已发生中断。
RSEC	Bit 8	R	**RTC 跨分，标志位中断状态** 0:无发生中断。 1:已发生中断。

AMALL	Bit 7	R	**RTC 全闹钟满足，标志位中断状态** 0:无发生中断。 1:已发生中断。
AYEAR	Bit 6	R	**年闹钟，标志位中断状态** 0:无发生中断。 1:已发生中断。
AMONTH	Bit 5	R	**月闹钟，标志位中断状态** 0:无发生中断。 1:已发生中断。
ADATE	Bit 4	R	**日闹钟，标志位中断状态** 0:无发生中断。 1:已发生中断。
AWEEK	Bit 3	R	**星期闹钟，标志位中断状态** 0:无发生中断。 1:已发生中断。
AHOUR	Bit 2	R	**小时闹钟，标志位中断状态** 0:无发生中断。 1:已发生中断。
AMIN	Bit 1	R	**分闹钟，标志位中断状态** 0:无发生中断。 1:已发生中断。
ASEC	Bit 0	R	**秒闹钟，标志位中断状态** 0:无发生中断。 1:已发生中断。

RTC_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 RTC_RIF 与 RTC_IVS 进行硬件运算，公式如下:RTC_IFM = RTC_RIF & RTC_IVS

21. 5. 2. 14 RTC 中断清除寄存器(RTC_ICR)

RTC 中断清除寄存器(RTC_ICR)																														
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
															WKTM	F1HZ	RYEAR	RMON	RDATE	RWEEK	RHOUR	RMIN	RSEC	AMALL	AYEAR	AMONTH	ADATE	AWEEK	AHOUR	AMIN	ASEC

—	Bits 31-17	—	—
WKTM	Bit 16	C_W1	**清除 RTC 唤醒计数器计数完毕中断状态** 此位设置时，清除中断状态(RTC_RIF 与 RTC_IFM)。
F1HZ	Bit 15	C_W1	**清除 RTC 1Hz 中断状态** 此位设置时，清除中断状态(RTC_RIF 与 RTC_IFM)。
RYEAR	Bit 14	C_W1	**清除 RTC 跨世纪中断状态** 此位设置时，清除中断状态(RTC_RIF 与 RTC_IFM)。
RMON	Bit 13	C_W1	**清除 RTC 跨年中断状态** 此位设置时，清除中断状态(RTC_RIF 与 RTC_IFM)。
RDATE	Bit 12	C_W1	**清除 RTC 跨月中断状态** 此位设置时，清除中断状态(RTC_RIF 与 RTC_IFM)。
RWEEK	Bit 11	C_W1	**清除 RTC 跨周中断状态** 此位设置时，清除中断状态(RTC_RIF 与 RTC_IFM)。
RHOUR	Bit 10	C_W1	**清除 RTC 跨日中断状态** 此位设置时，清除中断状态(RTC_RIF 与 RTC_IFM)。
RMIN	Bit 9	C_W1	**清除 RTC 跨小时中断状态** 此位设置时，清除中断状态(RTC_RIF 与 RTC_IFM)。
RSEC	Bit 8	C_W1	**清除 RTC 跨分中断状态** 此位设置时，清除中断状态(RTC_RIF 与 RTC_IFM)。

AMALL	Bit 7	C_W1	**清除 RTC 全闹钟满足中断状态** 此位设置时，清除中断状态(RTC_RIF 与 RTC_IFM)。
AYEAR	Bit 6	C_W1	**清除年闹钟中断状态** 此位设置时，清除中断状态(RTC_RIF 与 RTC_IFM)。
AMONTH	Bit 5	C_W1	**清除月闹钟中断状态** 此位设置时，清除中断状态(RTC_RIF 与 RTC_IFM)。
ADATE	Bit 4	C_W1	**清除日闹钟中断状态** 此位设置时，清除中断状态(RTC_RIF 与 RTC_IFM)。
AWEEK	Bit 3	C_W1	**清除星期闹钟中断状态** 此位设置时，清除中断状态(RTC_RIF 与 RTC_IFM)。
AHOUR	Bit 2	C_W1	**清除小时闹钟中断状态** 此位设置时，清除中断状态(RTC_RIF 与 RTC_IFM)。
AMIN	Bit 1	C_W1	**清除分闹钟中断状态** 此位设置时，清除中断状态(RTC_RIF 与 RTC_IFM)。
ASEC	Bit 0	C_W1	**清除秒闹钟中断状态** 此位设置时，清除中断状态(RTC_RIF 与 RTC_IFM)。

RTC_ICR 寄存器设置时，将清除 RTC_RIF 与 RTC_IFM 中断标志位状态；此设置不影响中断 RTC_IER、RTC_IDR 与 RTC_IVS 寄存器，只清除标志位状态 RTC_RIF 与 RTC_IFM。此寄存器通过硬件清除中断，公式如下:
RTC_RIF = RTC_RIF & ~RTC_ICR

21. 5. 2. 15 RTC 状态寄存器(RTC_STAT)

RTC 状态寄存器 (RTC_STAT)																												
偏移地址:0x50																												
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
																															EMPTY

—	Bits 31-1	—	—
EMPTY	Bit 0	R	**RTC 日历寄存器状态** 0:未对 RTC_CAL 寄存器填入数值。 1: RTC_CAL 寄存器内数值不为 0。

第22章 串行通讯 (I2C)

22. 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。

标准 I2C 是一个多主机总线且包括冲突检测与仲裁，如果有两个或两个以上的主机试图同时控制总线时，其仲裁可以防止数据损坏。提供了标准模式(Sm)、快速模式(Fm)与极快速模式(Fm+)供用户选择。并且也提供 SMBus(系统管理总线)与 PMBus(电源管理总线)。

22. 2 特性

- 配置为主机或从机
- 多主机模式
- 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 位与 10 位地址模式
- 提供 2 组 7 位从机地址(2 个地址，其中一个包括屏蔽功能)
- 提供所有 7 位地址接听模式
- 提供广播模式
- 可配置的建立时间和保持时间
- 可选择时钟延长
- 可配置数字滤波器
- 提供 SMBus 标准功能
- 硬件 PEC(封包错误检查)产生与 ACK 控制
- 命令与数据应答控制
- 提供地址解析通讯协议
- 提供可选择为主机或从设备
- 提供 SMBus 警报
- 提供侦测超时与空闲功能
- 提供 PMBus rev 1.1 标准功能

22. 3 结构图

关于 I2C 界面的结构如下图所示:

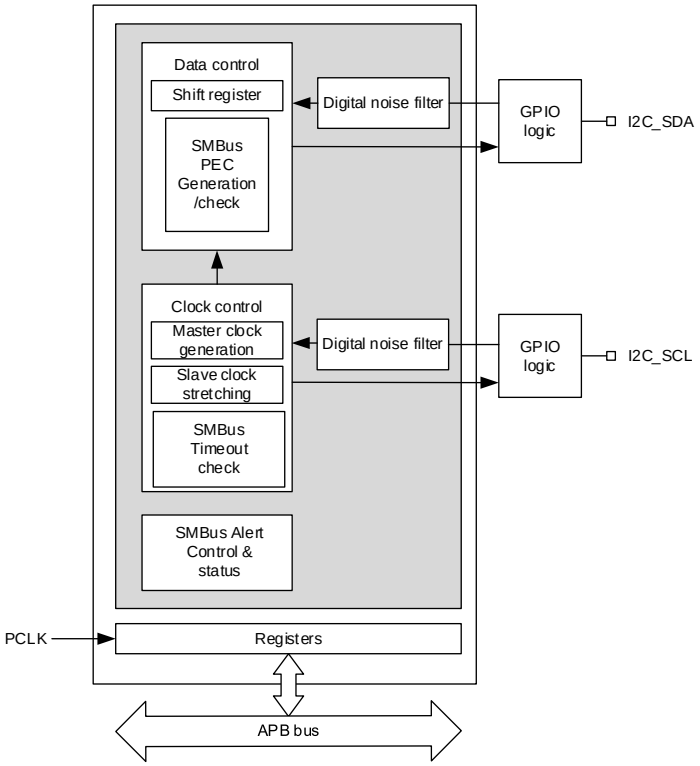


图 22-1　I2C 结构图

22. 4 功能描述

除了接收和发送数据外，该接口还将其从串行格式转换为并行格式，反之亦然。中断由软件开启或关闭。该接口通过数据引脚(SDA)和时钟引脚(SCL)连接到 I2C 总线。它可以连接标准模式(最高 100 kHz)，快速模式(最高 400 kHz)或极快速模式(最高 1 MHz)的 I2C 总线。

该接口也可通过数据引脚(SDA)和时钟引脚(SCL)连接到 SMBus。

如果支持 SMBus 功能:可以使用 SMBus 警报引脚(SMBA)。

22. 4. 1 I2C总线协议

I2C 是一种双线双向串行总线，可在设备之间提供简单有效的数据交换方法。标准 I2C 是真正的多主机总线，包括冲突检测和仲裁，如果两个或多个主机同时尝试控制总线，可防止数据损坏。数据在主机和从机之间逐字节同步传输到串行数据(SDA)和串行时钟(SCL)在线，每个数据字节长度为 8 位。

22. 4. 1. 1 START和STOP条件协议

I2C 规范将起始条件定义为 SDA 从高状态到低状态的转换，且 SCL 为高电平。起始条件由主机产生，表示总线从空闲状态转换为活动状态。每个数据位有一个 SCL 时钟脉冲，首先发送 MSB。每个传送的字节后面都有一个应答位。当 SCL 为高电平时，SDA 上的转换被解释为命令(START 或 STOP)。在 SCL 的高电平期间对每个位进行采样; 因此，SDA 仅可以在 SCL 的低电平期间改变，并且必须在 SCL 的高电平期间保持稳定。

当总线空闲时，SCL 和 SDA 都通过总线上的外部上拉电阻拉高。当主机想要在总线上开始传输时，主机发出 START 信号。这被定义为 SDA 从高到低的转换，且 SCL 为 1。当主机想要停止传输时，主机发出 STOP 条件。这被定义为 SDA 从低到高的转换，且 SCL 为 1。下图表示了 START 和 STOP 条件的时序。当数据在总线上传输时，当 SCL 为 1 时，SDA 必须稳定。

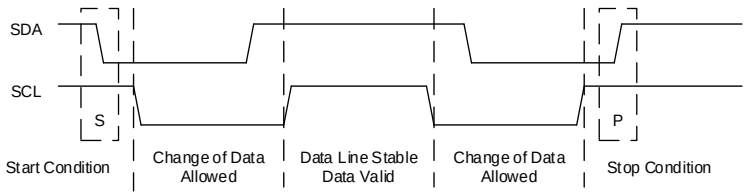


图 22-2 START 和 STOP 条件

22. 4. 1. 2 应答位

数据的传输必需带有应答位。应答位相关的时钟脉冲由主机产生。主机在应答时钟脉冲期间释放 SDA(高电平)。从机必须在应答时钟脉冲低电平期间下拉 SDA，以便在时钟脉冲的高电平期间保持稳定的低电平。当然，还必须考虑建立和保持时间。通常，已经被寻址的从机必须在接收到每个字节后产生应答，除非数据以 CBUS 地址开始。

当从机不应答从机地址时(例如，由于它正在执行某些实时功能而无法接收或发送)，从机必须将数据线保持为高电平。然后，主机可以产生 STOP 条件以中止传输，或者重复 START 条件以开启新传输。

如果从机应答了从机地址，但是，传输后的某个时间不能再接收数据字节，则主机必须中止传输。这由从机在随后的第一个字节上产生非应答来表示。从机使数据线保持高电平，主机产生 STOP 或重复 START 条件。

如果主机涉及接收传输，它必须通过不在从时钟输出的最后一个字节上产生非应答来向从机发送数据结束信号。从机必须释放数据线，以允许主机产生 STOP 或重复 START 条件。

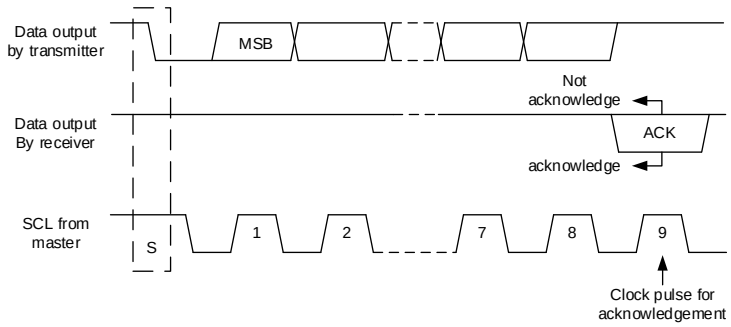


图 22-3 I2C 总线上的应答

22. 4. 1. 3 I2C寻址从协议

有两种地址格式:7 位地址格式和 10 位地址格式。在 7 位地址格式期间，第一个字节的前 7 位(位 7:1)设置从地址，LSB 位(位 0)是 R/W 位。当位 8 设置为 0 时，主机写入从机。当位 8 设置为 1 时，主机从从机读取。数据首先传输最高有效位(MSB)。在 10 位寻址期间，传输两个字节以设置 10 位地址。第一个字节的传输包含以下位定义。前 5 位(位 7:3)通知从机这是一个 10 位传输，接着是后两位(位 2:1)，它为从机地址位 9:8，LSB 位(位 8)是 R/W 位。传输的第二个字节设置从地址的位 7:0。

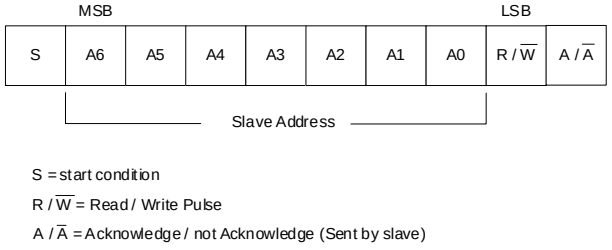


图 22-4 7 位地址格式

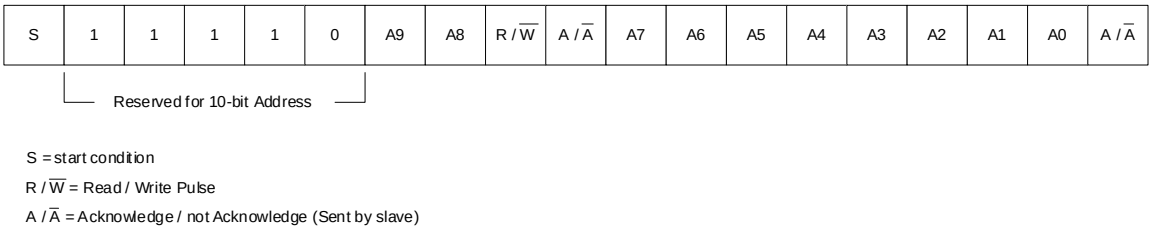


图 22-5　10 位地址格式

从地址	R/W位	描述
0000 000	0	广播地址
0000 000	1	START字节
0000 001	X	CBUS地址
1111 0XX	X	10位从机寻址

表 22-1　第一个字节中位的定义

22. 4. 1. 4　I2C发送和接收协议

所有数据都以字节格式传输，对每次传输的字节数没有限制。在主机发送地址和 R/W 位或主机向从机发送一个字节数据后，接收的从机必须响应应答信号。当接收的从机没有响应应答脉冲时，主机通过发出 STOP 条件来中止传输。从机应使 SDA 保持高电平，以便主机可以中止传输。如果主机正在发送数据，则接收的从机在接收到每个数据字节后用应答脉冲响应主机。传输格式如下:

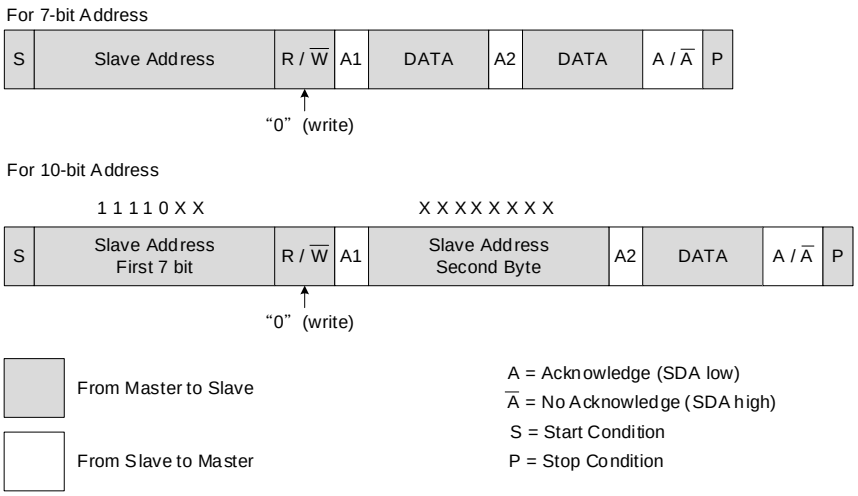


图 22-6　主机 - 发送协议

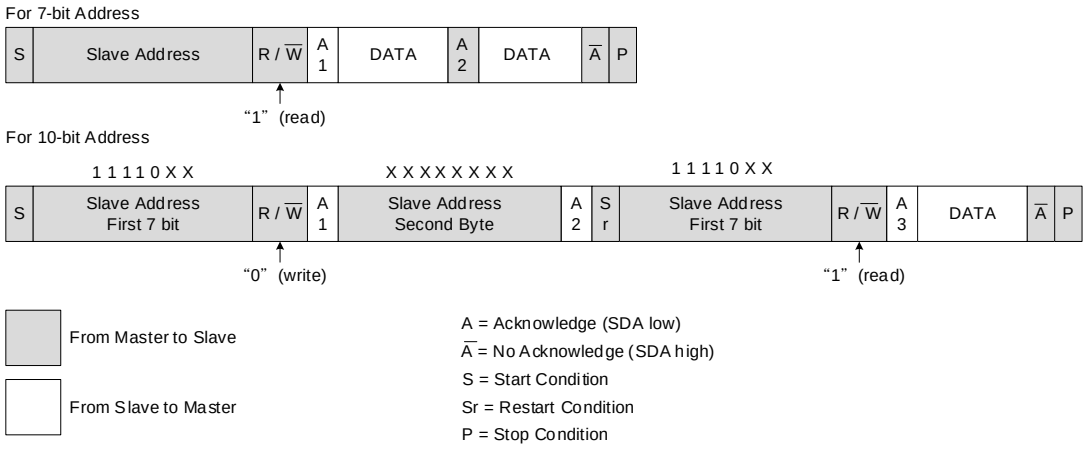


图 22-7　主机 - 接收协议

22. 4. 2　I2C时钟要求

I2C 内核由 I2CCLK 提供时钟。

I2CCLK 周期 T_{I2CCLK} 必须符合以下条件:

$T_{I2CCLK} < (T_{LOW} - T_{Filter}) / 4$ 与 $T_{I2CCLK} < T_{HIGH}$

关于:

T_{LOW}:SCL 低电平时间。

T_{HIGH}:SCL 高电平时间。

T_{Filter}:开启时，数字滤波器带来的延迟总和。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:

$T_{PCLK} < 4/3\ T_{SCL}$

T_{SCL}:SCL 周期。

22. 4. 3 数据传输

SDA 上的每个字节必须为 8 位长。每次传输可以传输的字节数不受限制，每个字节后面都必须有一个应答位，使用最高有效位(MSB)传输数据。如果从机不能接收或发送另一个完整的数据字节，则它可以将 SCL 保持为低电平以强制主机进入等待状态，直到它执行完了某些其他功能，例如服务于内部中断。当从机准备好接收另一个数据字节并释放 SCL 时，数据传输继续。

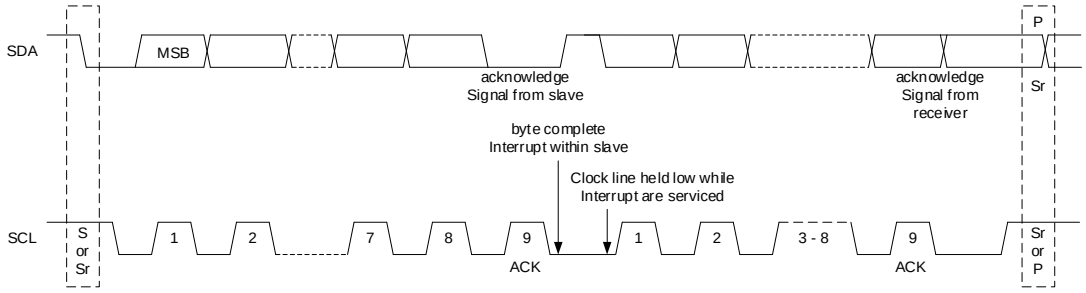


图 22-8 I2C 总线上的数据传输

接收

SDA 输入移位寄存器。在第 8 个 SCL 脉冲之后(当接收到完整的数据字节时)，如果 RXNE=1，表示尚未读取先前接收的数据字节，此时 SCL 会在第 9 个 SCL 脉冲之后延长，直到读取 **I2C_RXDATA** 寄存器，使得接收器为空。

传输

如果发送器不为空(TXE=0)，则在第 9 个 SCL 脉冲(应答脉冲)之后将其内容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，表示发送器尚未写入数据，SCL 将被拉低，直到通过写入 **I2C_TXDATA** 寄存器到发送器。延长 SCL 在第 9 个 SCL 脉冲之后完成。

硬件传输管理

I2C 在硬件中的带有字节计数器，以便管理字节传输并以各种模式关闭通信，例如:在主模式下产生 NACK、STOP 和 RESTART，从接收器模式下的 ACK 控制，支持 SMBus 功能时的 PEC 产生/检查。

字节计数器在主模式下使用。默认情况下，它在从机模式下关闭，但可以通过设置 **I2C_CON1** 寄存器中的 SBC 位(从机字节控制)位由软件开启。设置 **I2C_CON1** 寄存器和 **I2C_CON2** 寄存器中的 NBYTES[15:0]位选择要传输的字节数。如果要传送的字节数(NBYTES)大于 65535，或者接收的字节数(NBYTES)大于 255，或者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，当传送到 NBYTES 中所设置的字节数时，设置 TCR 位为 1，如果 **I2C_IER** 寄存器中的 TCR 位为 1，则会产生中断。只要设置了 TCR 位，SCL 就会被延长。当软件对 NBYTES 写入非零值时，硬件会自动清除 TCR。

当 NBYTES 计数器重载最后一个字节数时，必须清除 RELOAD 位。

当主模式下 RELOAD 位=0 时，计数器可用于 2 种模式:

- **自动结束模式(I2C_CON2** 寄存器中的 AUTOEND 位为 1)。在此模式下，一旦传输了 NBYTES[15:0]位的字节数，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，一旦传输了 NBYTES[15:0]位的字节数，就会产生软件操作; 设置 TC 位为 1，如果 **I2C_IER** 寄存器中的 TC 位为 1，则会产生中断。只要设置了 TC 位，SCL 就会被延长。当设置 **I2C_CON2** 寄存器中的 START 或 STOP 位为 1 时，TC 位由软件清零。当主机要发送 RESTART 条件时，必须使用此模式。

22. 4. 4 I2C从机模式

I2C 从机初始化

为了在从机模式下工作，用户必须至少开启一个从机地址。两个地址 **I2C_ADDR1** 和 **I2C_ADDR2** 寄存器可用于设置从机自身地址 OA1 和 OA2。

- 通过设置 **I2C_ADDR1** 寄存器中的 OA1MODE 位为 1，可以在 7 位模式(默认情况下)或 10 位寻址模式下设置 OA1。通过设置 **I2C_ADDR1** 寄存器中的 OA1EN 位为 1 开启 OA1。
- 如果需要额外的从地址，可以设置第二个从地址 OA2。通过设置 **I2C_ADDR2** 寄存器中的 OA2MSK[2:0]位，可以屏蔽最多 7 个 OA2 LSB。因此，对于设置为 1 至 6 的 OA2MSK 位，仅使用 OA2[7:2]、OA2[7:3]、OA2[7:4]、OA2[7:5]、OA2[7:6]或 OA2[7]与收到的地址相比较。一旦 OA2MSK 位不等于 0，OA2 的地址比较器就会排除未被应答的 I2C 保留地址(0000 XXX 和 1111 XXX)。如果 OA2MSK=7，则应答所有接收的 7 位地址(保留地址除外)。OA2 只能是 7 位地址。

  当 OA2MSK=0 的情况下且在 **I2C_ADDR1** 或 **I2C_ADDR2** 寄存器中的设置特定开启位，则可应答这些保留地址。通过设置 **I2C_ADDR2** 寄存器中的 OA2EN 位为 1 开启 OA2。

- 通过设置 **I2C_CON1** 寄存器中的 GCEN 位为 1 开启广播地址。

当其中一个开启地址选择 I2C 时，地址匹配中断标志位状态设置为 1，如果 **I2C_IER** 寄存器中的 ADDR 位为 1，则会产生中断。

默认情况下，从机使用其时钟延长功能，这表示它在需要时将 SCL 拉到低电平，以执行软件操作。如果主机不支持时钟延长，则必须在 **I2C_CON1** 寄存器中将 I2C 设置 NOSTRETCH 位为 1。

收到地址匹配中断后，如果开启了多个地址，用户必须读取 **I2C_STAT** 寄存器中的 ADDCODE[6:0]位，以检查匹配的地址以及检查 DIR 位以了解传输方向。

从机时钟延长(NOSTRETCH=0)

在默认模式下，I2C 从机在以下情况下延长 SCL 时钟:

- 在传输过程中，如果先前的数据传输完成且没有在 **I2C_TXDATA** 寄存器中写入新数据。当数据写入 **I2C_TXDATA** 寄存器时，将释放此延长。

- 在接收过程中，尚未读取 **I2C_RXDATA** 寄存器。读取 **I2C_RXDATA** 寄存器将释放此延长。
- 从机字节控制模式下，当发生 TCR=1 时，重载模式(SBC 位=1 且 RELOAD 位=1)，表示数据字节已被传输。此时通过在 NBYTES[15:0]字段中写入非零值清除 TCR 时，释放该延长。
- 从机应答控制模式下，当收到地址或数据时，每个字节的第 8 和第 9 个 SCL 脉冲之间会延长 SCL。当设置应答位更新时，将会释放该延长并响应应答脉冲。
- 在 SCL 下降沿检测后，I2C 拉低电平延长 SCL 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** 寄存器中的 NOSTRETCH 位为 1 时，I2C 从机不会延长 SCL。

- 在传输过程中，必须在与传输相对应的第一个 SCL 脉冲之前将数据写入 **I2C_TXDATA** 寄存器。如果不是，则发生下溢错误，在 **I2C_STAT** 寄存器中的 TXUD 位被设置为 1，如果 **I2C_IER** 寄存器中的 TXUD 位为 1，则会产生中断。
- 在接收过程中，如果接收器非空时，必须在下一个数据字节的第 9 个 SCL 脉冲(应答脉冲)之前从 **I2C_RXDATA** 寄存器中读取数据。如果发生溢出，在 **I2C_STAT** 寄存器中的 RXOV 位被设置为 1，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

为了在从机接收模式下允许字节 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。这需要符合 SMBus 标准。

必须选择重载模式，以便在从机接收模式(RELOAD 位=1)中允许字节控制。要控制每个字节，必须在 ADDR 中断子程序中将 NBYTES 初始化为 0x1，并在每个接收到的字节后重载到 0x1。当接收到该字节时，TCR 位被设置为 1，在第 9 个 SCL 脉冲之后将 SCL 拉低。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 位决定下一个数据是否应答。通过将 NBYTES 设置为非零值来释放 SCL 延伸:发送应答或不应答。

NBYTES 可以加载大于 0x1 的值，在这种情况下，接收流程在 NBYTES 数据接收期间是连续的。

注:当关闭 I2C 时，才可设置 SBC 位。当 TCR=1 时，此时可以更改 RELOAD 位。

注: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。

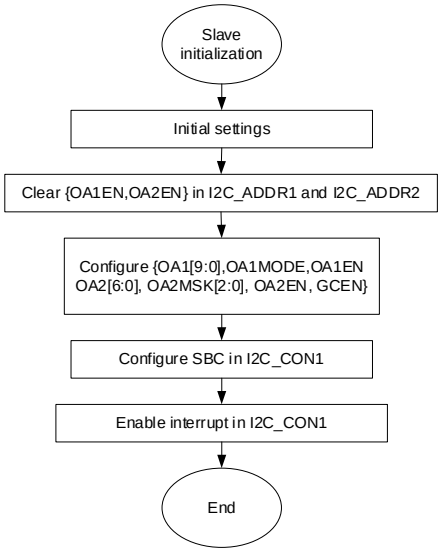


图 22-9　从机初始化流程图

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，此时若 TX 已准备好要发送的数据时，从机会通过内部移位寄存器将 TX 中的字节发送到 SDA。若 TX 此时为空，从机会延长 SCL 低电平时间，直到 TX 通过 **I2C_TXDATA** 寄存器写入发送数据为止。

当发送成功，主机会回应答信号，当主机响应 NACK 时，此时 I2C_RIF 寄存器中的 NACK 位被设置为 1，此时从机会自动释放 SCL 与 SDA 总线让主机能够发送后续的 STOP 或 RESTART 命令。

当主机响应 STOP 时，此时 **I2C_RIF** 寄存器中的 STOP 位被设置为 1，并结束通信，等待下一次收到匹配的地址。然而，若 TX 内还有尚未传送的字节，使用者可以选择下一次地址匹配时，继续传送数据。

当从机字节控制开启时，设置 **I2C_CON1** 寄存器和 **I2C_CON2** 寄存器中的 NBYTES 位选择需要传送多少字节。

注意:当 NOSTRETCH 模式开启时，由于 SCL 时钟无法被延长，因此需要传送的字节需要被提前写入 I2C_TXDATA 寄存器。

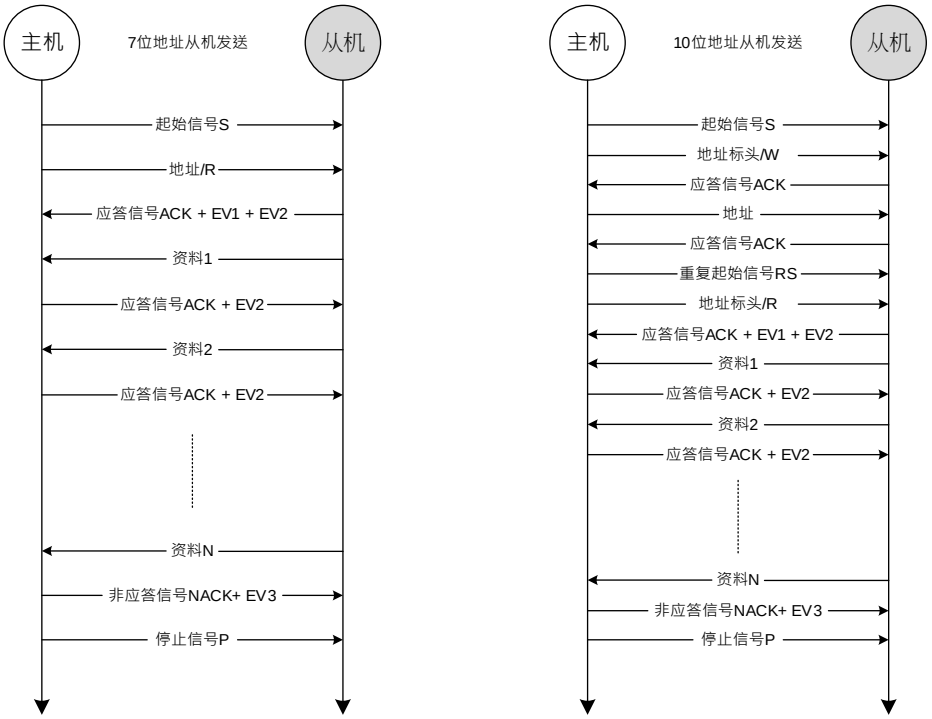


图 22-10　从机发送的传输序列图

注 1:S=起始位，RS=重复起始位，P=停止位，ACK=应答，NACK=非应答

注 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。

注 3:EV2=判断 I2C_STAT 寄存器中的 TXE 位，若为空写入数据至 I2C_TXDATA 寄存器。

注 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 TX 是否还有尚未发送的字节，若有则软件需要额外处理。

注 5:如果软件列在当前传送字节传输结束之前尚未写入下一个字节，导致 TX 为空时，EV2 事件将会延长 SCL 时钟低电平时间。

从机接收

在接收到匹配的地址时，**I2C_RIF** 寄存器中的 ADDR 位被设置为 1，从机此时会通过内部移位寄存器将 SDA 上的字节保存到 RX 中，无论 RX 是否为空，从机都会自动发送应答信号，差异仅在应答信号发送后，非空时会延长 SCL 时钟低电平时间，等待软件读取 RX 字节后，才会释放 SCL 时钟。

当主机发送 STOP 时，此时 **I2C_RIF** 寄存器中的 STOP 位被设置为 1，并结束通信，等待下一次收到匹配的地址。

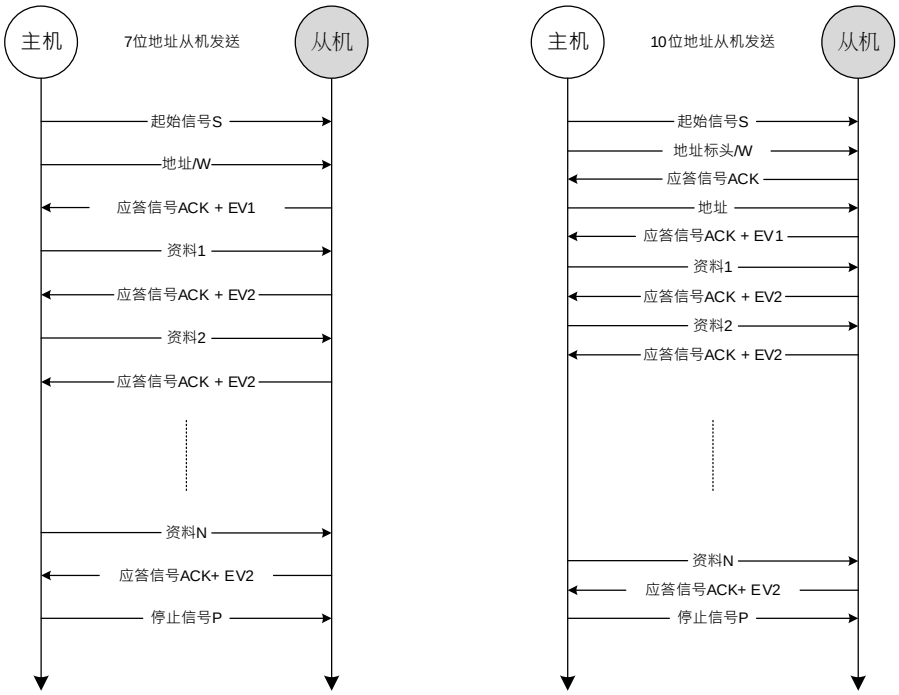


图 22-11　从机接收的传输序列图

注 1:S=起始位，RS=重复起始位，P=停止位，ACK=应答，NACK=非应答

注 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。

注 3:EV2=判断 I2C_STAT 寄存器中的 RXNE 位，若非空则读取 I2C_RXDATA 寄存器数据。

注 4:如果软件在当前传送字节传输结束时，导致 RX 非空，EV2 事件将会延长 SCL 时钟低电平时间。

22. 4. 5 I2C主机模式

I2C 主机初始化

在开启外设之前，必须通过设置 **I2C_TIMINGR** 寄存器中的 SCLH 位和 SCLL 位来配置 I2C 主机时钟。实现同步时钟机制以支持多主机环境和从机时钟延长。

为了允许同步时钟:

- 从 SCL 低电平内部检测开始，使用 SCLL 计数器计数低电平时钟。
- 从 SCL 高电平内部检测开始，使用 SCLH 计数器计数高电平时钟。

根据 SCL 下降沿，I2C 在 T_{SYNC} 延迟后检测到自己的 SCL 低电平。一旦 SCLL 计数器达到 **I2C_TIMINGR** 寄存器中的 SCLL[7:0]位中设置的值，I2C 就会将 SCL 释放为高电平，同步 SCL 输入数字噪声滤波器和 SCL 与 I2CCLK 时钟。

在 T_{SYNC} 产生延迟后，I2C 会检测自己的 SCL 高电平，具体取决于 SCL 上升沿，同步 SCL 输入数字噪声滤波器和 SCL 与 I2CCLK 时钟。

一旦 SCLH 计数器达到设置 **I2C_TIMINGR** 寄存器中的 SCLH[7:0]位的数值，I2C 就会将 SCL 拉到低电平。

因此，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
- 启用时，数字滤波器引起的输入延迟:DNF x T_{I2CCLK}
- 由于同步 SCL 与 I2CCLK 时钟(2 至 3 个 I2CCLK 周期)导致的延迟

T_{SYNC2} 的持续时间取决于以下参数:

- SCL 上升斜率
- 开启时，数字滤波器引起的输入延迟:DNF x TI2CCLK
- 由于同步 SCL 与 I2CCLK 时钟(2 至 3 个 I2CCLK 周期)导致的延迟

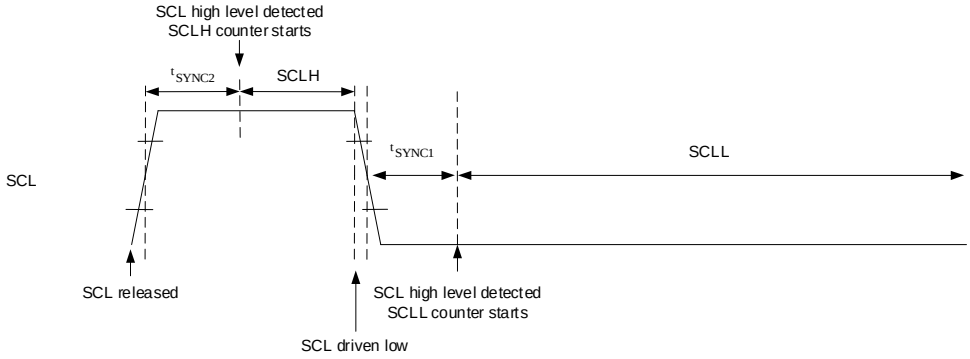


图 22-12 主机时钟产生

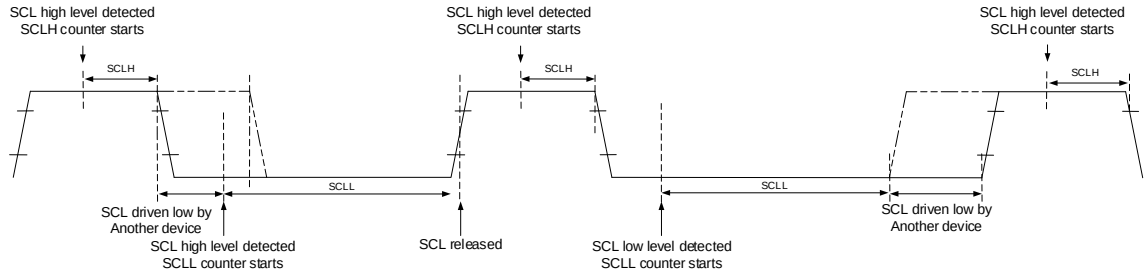


图 22-13　SCL 主机时钟同步

主机通信初始化(地址阶段)

为了开启通信，用户必须设置 **I2C_CON2** 寄存器中的寻址的从机参数:

- 寻址模式(7 位或 10 位):ADD10
- 要发送的从机地址:SADD [9:0]
- 传输方向:RD_WRN
- 如果是 10 位地址读取:HEAD10R 位。必须配置 HEAD10R 位以指示是否必须发送完整的地址序列，或者仅在方向改变时发送地址标头。
- 要传输的字节数:NBYTES[15:0]。
  - ◇ 在传送模式: 如果字节数等于或大于 65535 个字节，则 NBYTES[15:0]可以先填入 0xFFFF。
  - ◇ 在接收模式: 如果字节数等于或大于 255 个字节，则 NBYTES[15:0]可以先填入 0x00FF。

然后，使用者必须设置 **I2C_CON2** 寄存器中的 START 位为 1。当 START 位为 1 时，不允许更改所有上述位。一旦检测到总线空闲(BUSY=0)后，主机就会自动发送 START 条件，再发送从机地址。

在仲裁丢失的情况下，主机自动切换回从机模式，如果它作为寻址从机，则可以应答自己的地址。

注:无论接收到的应答值是什么，当总线上的从机地址发送时，START 位由硬件清除。如果发生仲裁丢失，则 START 位也由硬件清除。在 10 位寻址模式下，当从机地址前 7 位被从机 NACK 时，主机将自动重复起始从机地址传输，直到收到 ACK。如果在 START 位为 1 时 I2C 被寻址为从机(ADDR=1)，则 I2C 切换到从机模式，当 ADDR 位为 1 时，START 位清零。

注:对重复起始条件应用相同的过程。在这种情况下，BUSY=1。

初始化主机 10 位从地址寻址接收模式

- 如果从机地址采用 10 位格式，用户可以通过清零 I2C_CON2 寄存器中的 HEAD10R 位来选择发送完整的读序列。在这种情况下，主机在 START 位为 1 后自动发送以下完整序列:起始+从机地址 10 位标头写+从机地址第 2 字节+重复起始+从机地址 10 位标头读取；
- 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。

主机传送

在主机开始传送之前，需先设置 **I2C_CON1** 寄存器和 **I2C_CON2** 寄存器中的 NBYTES 位，当传输字节大于 65535 时，需要额外设置 RELOAD 位。在此配置下，当 NBYTES 配置的笔数传送完成后，**I2C_RIF** 寄存器中的 TCR 位被设置为 1，此时 SCL 时钟会保持低电平，直到 NBYTES 位重新写入新的数值以及对 **I2C_TXDATA** 寄存器写入发送数据后，才会继续进行传输。

当从机回应 NACK，**I2C_RIF** 寄存器中的 NACK 位被设置为 1，并且接下来主机会自动发送停止信号 STOP。

当从机响应 ACK，且 RELOAD 位=0、NBYTES 所配置的笔数都已经传完时，会有以下情况:

- 若 AUTOEND=1，此时会自动发送停止信号 STOP。
- 若 AUTOEND=0，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信号。
  - STOP:设置 **I2C_CON2** 寄存器中的 STOP 位为 1，此时会发送停止信号。

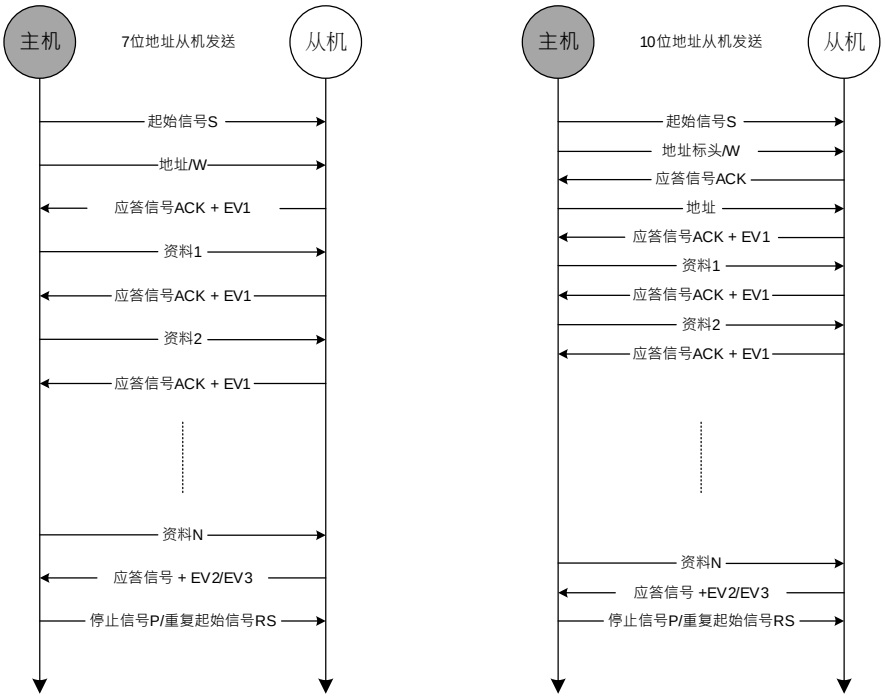


图 22-14　主机发送的传输序列图

注 1:S=起始位，RS=重复起始位，P=停止位，ACK=应答，NACK=非应答

注 2:EV1=传输尚未完成事件，判断 I2C_STAT 寄存器中的 TXE 位，若为空写入数据至 I2C_TXDATA 寄存器。

注 3:EV2=当收到应答信号 ACK 时，若传输已完成，后续操作为停止信号 STOP 或重复起始信号 RESTART。

注 4:EV3=当收到非应答信号 NACK 时，后续操作为停止信号 STOP。

注 5:如果软件在当前传送字节传输结束之前尚未写入下一个字节，导致 TX 为空时，EV1 事件将会延长 SCL

时钟低电平时间。

主机接收

在主机开始接收之前，需先设置 **I2C_CON1** 寄存器和 **I2C_CON2** 寄存器中的 NBYTES 位，当传输字节大于 255 时，需要额外配置 RELOAD 位。在此配置下，当 NBYTES 配置的笔数传送完成后，**I2C_RIF** 寄存器中的 TCR 位被设置为 1，此时 SCL 时钟会保持低电平，直到 NBYTES 位重新写入新的数值后，才会继续进行传输。

当 RELOAD 位=0 并且 NBYTES 所配置的笔数都已经传完时，会有以下情况:

- 若 AUTOEND=1，此时会自动发送非应答信号 NACK 与停止信号 STOP。
- 若 AUTOEND=0，此时会自动发送非应答信号 NACK，并将 SCL 时钟保持低电平，等待软件控制后续操作:
  - RESTART:对 **I2C_CON2** 寄存器中的 START 位设置为 1，此时会发送重复起始信号。
  - STOP:对 **I2C_CON2** 寄存器中的 STOP 位设置为 1，此时会发送停止信号。

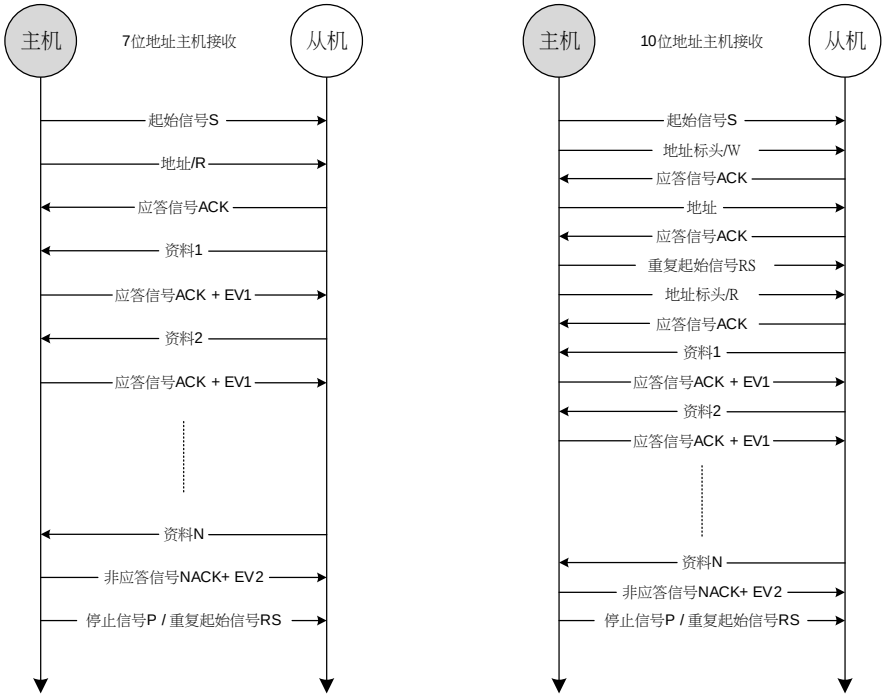


图 22-15　主机接收的传输序列图

注 1:S=起始位，RS=重复起始位，P=停止位，ACK=应答，NACK=非应答

注 2:EV1=传输尚未完成事件，判断 I2C_STAT 寄存器中的 RXNE 位，若非空则读取 I2C_RXDATA 寄存器数据。

注 3:EV2=传输完成事件，此时会自动发送非应答信号 NACK，后续根据软件配置来决定是发送停止信号 STOP 还是重复起始信号 RESTART。

注 4:如果软件列在当前传送字节传输结束之前尚读取数据，导致 RX 非空时，EV1 事件将会延长 SCL 时钟低

电平时间。

22. 4. 6 I2C_TIMINGR寄存器中配置的例子

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+)
	10 kHz	100 kHz	400 kHz	500 kHz
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 22-2 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+)
	10 kHz	100 kHz	400 kHz	1000kHz
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 22-3 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+)

	10 kHz	100 kHz	400 kHz	1000kHz
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 = 1250 ns	5x250 = 1250 ns	4x125 = 500 ns	2x125 = 250 ns

表 22-4 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。

22. 4. 7 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 rev 2.0(http://smbus.org)兼容。

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。
- 主设备用于发出命令，产生时钟并停止传输的设备。
- 主机用于专用主机，为系统的 CPU 提供主接口。主机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机。该外设可以配置为主设备或从设备，也可以配置为主机。

SMBUS 基于 I2C 规范 rev 2.1。

总线协议

对于任何给定的设备，有 11 种可能的命令协议。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、传送的字节、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，请参阅 SMBus 规范 2.0 版(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)。这个 128 位数字由软件实现。

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 从设备默认地址(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，请参阅 SMBus 规范 2.0 版(http://smbus.org)。

收到命令和数据应答控制

SMBus 接收器必须能够 NACK 每个接收到的命令或数据。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。

主机通知协议

该外设通过设置 **I2C_CON1** 寄存器中的 SMBHEN 位为 1 来支持主机通知协议。在这种情况下，主机将应答 SMBus 主机地址(0b0001 000)。使用此协议时，设备充当主机，主机充当从机。

SMBus 警报

支持 SMBus ALERT 可选信号。仅从设备可以通过 SMBALERT＃引脚向主机发送信号选择通信。主机处理中断并同时通过警报响应地址(0b0001 100)访问所有 SMBALERT＃设备。只有拉低 SMBALERT＃的设备才会应答警报响应地址。

当配置为从设备(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。警报响应地址同时开启。

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，当 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报错误检查机制，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分组错误代码(PEC)来实现分组错误检查。通过在所有消息字节(包括地址和读/写位)上使用 $C_{(x)} = X^8 + X^2 + X + 1$ CRC-8 多项式来计算 PEC。外设嵌入了硬件 PEC 计算器，当接收到的字节与硬件计算的 PEC 不匹配时，允许自动发送非应答。

超时

该外设嵌入了硬件定时器，以符合 SMBus 规范 2.0 版中定义的 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(从设备)	-	25	ms
$T_{LOW:MEXT}$(2)	累积时钟低电平延长时间(主设备)	-	10	ms

表 22-5　SMBus 超时规格

1. $T_{LOW:SEXT}$ 是允许给定从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。另一个从设备或主设备也可能延长时钟，导致组合时钟低电平延长时间大于 $T_{LOW:SEXT}$。因此，该参数是在从设备作为全速主设备的唯一目标的情况下测量的。
2. $T_{LOW:MEXT}$ 是允许主设备在从 START 到 ACK，ACK 到 ACK 或 ACK 到 STOP 定义消息的每个字节内延长其时钟周期的累积时间。从设备或另一个主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，使用全速从设备作为主设备的唯一目标来测量该参数。

总线空闲检测

如果总线检测到时钟和数据信号已经持续高电平并且 T_{IDLE} 大于 $T_{HIGH.MAX}$，则主设备可以认为总线是空闲的。

该时序参数涵盖了主机已动态添加到总线并且可能未检测到 SMBCLK 或 SMBDAT 线路上的状态转换的情况。在这种情况下，主设备必须等待足够长的时间以确保当前没有进行传输。外设支持硬件总线空闲检测。

22. 4. 8　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控制(从机模式)

SMBus 接收器必须能够 NACK 每个接收到的命令或数据。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。

特定地址(从机模式)

如果需要，应开启特定的 SMBus 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(0b1100 001)。

通过设置 **I2C_CON1** 寄存器中的 SMBHEN 位为 1 开启 SMBus 主机从机地址(0b0001 000)。

通过设置 **I2C_CON1** 寄存器中的 ALERTEN 位为 1 开启报警响应地址(0b0001100)。

数据包错误检查

通过设置 **I2C_CON1** 寄存器中的 PECEN 位为 1 开启 PEC 计算。然后在硬件字节计数器下管理 PEC 传输:**I2C_CON1** 寄存器和 **I2C_CON2** 寄存器中的 NBYTES[15:0]位。必须先设置 PECEN 位，然后再开启 I2C。

PEC 传输由硬件字节计数器管理，因此在从机模式下连接 SMBus 时必须设置 SBC 位。在设置 PECBYTE 位为 1 且 RELOAD 位清零后，在传输 NBYTES-1 数据后传输 PEC。如果设置了 RELOAD 位，则 PECBYTE 无效。

注:开启 I2C 时，不允许更改 PECEN 设置。

超时检测

通过设置 **I2C_TIMEOUTR** 寄存器中的 TIMEOUTEN 位和 TEXTEN 位为 1 来开启超时检测。定时器必须如以下的方式设置，即它们在 SMBus 规范版本 2.0 中给出的最大检测到超时。

◆ $T_{TIMEOUT}$ 检查

为了开启 $T_{TIMEOUT}$ 检查，必须设置 TIMEOUTA[11:0]位为定时器的重载值，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 位来开启定时器。如果 SCL 在大于(TIMEOUTA + 1)x 2048 x T_{I2CCLK} 的时间内被拉低，则在 **I2C_RIF** 寄存器中的 TOUT 位被设置为 1。

注:当设置 TIMEOUTEN 位为 1 时，不允许更改设置 TIMEOUTA[11:0]位和 TIDLE 位。

◆ $T_{LOW:SEXT}$ 和 $T_{LOW:MEXT}$ 检查

根据外设是配置为主机还是从机，必须设置 TIMEOUTB[11:0]位为定时器的重载值，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，因此用户可以为两者选择相同的值。然后，通过设置 **I2C_TIMEOUTR** 寄存器中的 TEXTEN 位为 1 来开启定时器。

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 TIMEOUTA[11:0]位为定时器的重载值，以获得 T_{IDLE} 参数。必须设置 TIDLE 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 都保持高电平的时间大于(TIMEOUTA + 1)x 4 x T_{I2CCLK}，则在 **I2C_RIF** 寄存器中的 TOUT 位被设置为 1。

注:设置 TIMEOUTEN 时，不允许更改设置 TIMEOUTA 位和 TIDLE 位。

22. 4. 9 SMBus: I2C_TIMEOUTR 寄存器配置的例子

◆ 将 $T_{TIMEOUT}$ 的最大持续时间配置为 25 ms:

F_{I2CCLK}	TIMEOUT[11:0] bits	TIDLE bit	TIMEOUTEN bit	$T_{TIMEOUT}$
8 MHz	0x61	0	1	98x2048x125 ns=25 ms
16 MHz	0xC3	0	1	196x2048x62.5 ns=25 ms
32 MHz	0x186	0	1	391x2048x31.25 ns=25 ms

表 22-6 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] bits	TIMEOUTEN bit	$T_{LOW:SEXT}$
8 MHz	0x1F	1	32x2048x125 ns=8 ms
16 MHz	0x3F	1	64x2048x62.5 ns=8 ms
32 MHz	0x7C	1	125x2048x31.25 ns=8 ms

表 22-7 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] bits	TIDLE bit	TIMEOUTEN bit	T_{IDLE}
8 MHz	0x63	1	1	100x4x125 ns=50 µs
16 MHz	0xC7	1	1	200x4x62.5 ns=50 µs
32 MHz	0x18F	1	1	400x4x31.25 ns=50 µs

表 22-8 各种 I2CCLK 频率的 TIMEOUTA 设置示例(最大 T_{IDLE} =50 µs)

22. 4. 10 错误情况

以下是可能导致通信失败的错误情况。

总线错误(BERR)

总线错误是指在总线上不在 9 个 SCL 时钟脉冲的倍数之后检测到 START 或 STOP 条件。仅当 I2C 作为主机或寻址从机进行传输时(不在从机模式下的地址阶段)，才会设置总线错误标志位。

如果在从机模式下检测到错误的 START 或 RESTART，则 I2C 会进入地址识别状态，就像正确的 START 条件一样。

检测到总线错误时，**I2C_RIF** 寄存器中的 BERR 位被设置为 1，如果 **I2C_IER** 寄存器中的 BERR 位为 1，则会产生中断。

仲裁丢失(ARLO)

仲裁丢失为当在 SDA 上发送高电平时，但在 SCL 上升沿上采样到低电平。

◆ 在主机模式下，在地址阶段，数据阶段和数据应答阶段检测仲裁丢失。在这种情况下，SDA 和 SCL 被释放，START 控制位由硬件清零，主机自动切换到从机模式。

- 在从机模式下，会在数据阶段和数据应答阶段检测仲裁丢失。在这种情况下，传输停止，SCL 和 SDA 被释放。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。

接收溢出 / 发送下溢错误(RXOV / TXUD)

当 NOSTRETCH = 1 时，在从机模式下检测到溢出或下溢错误:

- 当接收到新字节且接收非空时。新接收的字节将会丢失，并且自动发送 NACK 作为对新字节的应答。
- 在传输中:
  - 当应发送新字节且尚未写入发送器时，将发送 0xFF。

当检测到接收溢出或发送下溢错误时，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。

22. 4. 11 I2C中断

I2C 中的中断由一组六个寄存器控制。

- 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**)是一个只能读取的寄存器，用于读取模块的中断状态。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:
  - ◇ SMBus 警报
  - ◇ 发生超时
  - ◇ PEC 错误
  - ◇ 仲裁丢失
  - ◇ 总线错误
  - ◇ 传送完成并重载
  - ◇ 传送完成
  - ◇ 检测 STOP
  - ◇ 收到非应答
  - ◇ 地址匹配
  - ◇ 接收器不为空
  - ◇ 发送器为空
  - ◇ 发生接收溢出或发送下溢
- 中断标志位状态寄存器(IFM)
  - ◇ I2C 中断标志位状态寄存器(**I2C_IFM**)用于读取模块的标志位中断状态，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。

22. 4. 12 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 SMBus 超时计数器继续正常工作或停止计数。

22. 5 特殊功能寄存器

22. 5. 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 时钟寄存器
I2C_TIMEOUTR	0014_H	R/W	I2C 超时寄存器
I2C_STAT	0018_H	R	I2C 状态寄存器
I2C_PECR	0020_H	R/W	I2C PEC 寄存器
I2C_RXDATA	0024_H	R	I2C 接收器数据寄存器
I2C_TXDATA	0028_H	W	I2C 发送器数据寄存器
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

22. 5. 2 寄存器描述

22. 5. 2. 1 I2C控制寄存器 1 (I2C_CON1)

I2C 控制寄存器 1 (I2C_CON1)																													
偏移地址: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
NBYTES<15:8>								PECEN	ALERTEN	SMBDEN	SMBHEN	GCEN	-	NOSTRETCH	SBC	-	-	-	-	DNF<3:0>				-	-	-	-	-	-	-	PE

NBYTES	Bit 31-24	R/W	**字节数** 参照CON2.NBYTES描述 注: 在接收模式且RELOAD开启时，该位域必须保持为'0'。
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** 设备模式(SMBHEN = 0): 0:释放 SMBA 引脚为高电平且关闭警报响应地址标头: 0001100x 回应非应答。 1:将 SMBA 引脚驱动为低电平且开启警报响应地址标头: 0001100x 回应应答。 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 设备从机地址启用** 0:关闭设备从机地址。地址 0b1100001x 回应非应答。 1:开启设备从机地址。地址 0b1100001x 回应应答。 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 主机地址启用** 0:关闭主机地址。地址 0b0001000x 回应非应答。 1:开启主机地址。地址 0b0001000x 回应应答。 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。地址 0b00000000 回应非应答。 1:开启广播呼叫。地址 0b00000000 回应应答。
—	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除状态。 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:关闭从字节控制 1:开启从字节控制
—	Bit 15-12	—	—
DNF	Bit 11-8	R/W	**数字噪声滤波器** 该位用于配置 SDA 和 SCL 输入上的数字噪声滤波器。数字滤波器将过滤长度高达 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 SCL 和 SDA 被释放。内部状态机和状态位将恢复为其复位值。清零后，PE 必须保持低电平至少 3 个 APB 时钟周期。

22. 5. 2. 2 I2C控制寄存器 2 (I2C_CON2)

此寄存器必须按字(32 位)访问。

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	**数据包错误检查字节** 该位由软件设置，当传输 PEC 时，或者当接收到 STOP 条件、匹配的地址或当 PE=0 时，由硬件清零。 0:没有 PEC 传输。 1:请求 PEC 发送/接收 注:向该位写'0’无效。 设置 RELOAD 时，该位无效。 当 SBC=0 时，该位对从机模式无效。 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)** 该位由软件设置和清除。 0:软件结束模式:发送完 NBYTES 笔数据时设置 TC 标志位，将 SCL 拉低。 1:自动结束模式:发送完 NBYTES 笔数据时自动发送器 STOP 条件。 注:该位在从机模式或设置 RELOAD 位为 1 时无效。
RELOAD	Bit 24	R/W	**NBYTES 重载模式** 该位由软件设置和清除。 0:在 NBYTES 笔数据传输之后完成传输(接下

			来是 STOP 或 RESTART)。 1: NBYTES 笔数据传输后将继续传输(NBYTES 将重载)。发送完 NBYTES 笔数据后设置 TCR 标志位，将 SCL 拉低。
NBYTES	Bit 23-16	R/W	**字节数** 此位与 CON1.NBYTES 组合成 16 位，NBYTES={CON1.NBYTES[7:0],CON2.NBYTES[7:0]} 在此设置要发送/接收的字节数。 在 SBC=0 的从机模式下，该位无效。 注:不允许在设置 START 位时更改该位。
NACK	Bit 15	R/W1	**NACK 产生(从机模式)** 该位由软件设置，在发送器 NACK 时由硬件清零，或在接收到 STOP 条件或地址匹配时，或当 PE=0 时由硬件清零。 0:在下一个接收的字节发送 ACK。 1:在下一个接收的字节发送 NACK。 注:向该位写'0’无效。 该位仅用于从机模式:在主机接收模式下，无论 NACK 位值如何，在 STOP 或 RESTART 条件之前的最后一个字节之后自动产生 NACK。当从机接收 NOSTRETCH 模式发生溢出时，无论 NACK 位值如何，都会自动产生 NACK。当开启硬件 PEC 检查(PECBYTE=1)时，PEC 确认值不依赖于 NACK 值。
STOP	Bit 14	R/W1	**STOP 产生(主机模式)** 该位由软件设置，当检测到停止条件时，或当 PE=0 时由硬件清零。 在主机下: 0:无 STOP 产生。 1:当前字节传输后 STOP 产生。 注:向该位写'0’无效。
START	Bit 13	R/W1	**START 产生** 该位由软件设置，并在起始位后跟随地址序列发送器、仲裁丢失、检测超时错误或 PE=0 时由硬件清零。 0:无 START 产生。

			1: RESTART/START 产生: - 如果 I2C 处于主机且 AUTOEND=0，则在 NBYTES 传输结束后，当 RELOAD=0 时，将该位设置为 1 会并产生重复起始条件。 - 否则，一旦总线空闲，将该位设置为 1 产生 START 条件。 注:向该位写'0’无效。 即使总线忙，也可以设置 START 位。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	**10 位地址标头仅读取方向(主机接收模式)** 0:主机发送完整的 10 位从机地址读取序列:在写入方向上起始 + 2 字节 10 位地址 + 在读取方向上重新起始 + 10 位地址的第 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:不允许在设置 START 位时更改该位。
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:不允许在设置 START 位时更改该位。
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:不允许在设置 START 位时更改该位。
SADD	Bit 9-0	R/W	**从机地址位 0(主机模式)** 在 7 位地址模式下(ADD10=0):无效 在 10 位地址模式下(ADD10=1):写入发送的从地址第 0 位 **从机地址位 7:1(主机模式)** 在 7 位地址模式下(ADD10=0):写入发送的 7 位从地址 在 10 位地址模式下(ADD10=1):写入发送的从地址第 7:0 位 **从机地址位 9:8(主机模式)** 在 7 位地址模式下(ADD10=0):无效

			在 10 位地址模式下(ADD10=1):写入发送的从地址的第 9:8 位 注:不允许在设置 START 位时更改该位。

22. 5. 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。接收到的从机地址 OA1 响应非应答。 1:开启本机地址 1。接收到的从机地址 OA1 响应应答。
—	Bit 14-11	—	—
OA1MODE	Bit 10	R/W	**本机地址 1，10 位地址模式启用** 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	**OA1[0]:本机地址 1** 7 位地址模式:无效 10 位地址模式:地址的第 0 位 **OA1[7:1]:本机地址 1** 7 位地址模式: 7 位地址 10 位地址模式: 10 位地址的第 7:1 位 **OA1[9:8]:本机地址 1** 7 位地址模式:无效 10 位地址模式:地址的第 9:8 位 注:当 OA1EN=0 时才能写入该位。

22. 5. 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。接收到的从机地址 OA2 响应非应答。 1:开启本机地址 2。接收到的从机地址 OA2 响应应答。
—	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，并且应答所有(保留位除外)7 位接收地址。 注:当 OA2EN=0 时才能写入该位。 一旦 OA2MSK 不等于 0，则 I2C 即使比较匹配，也不会应答保留地址 (0b0000xxx 和 0b1111xxx)。
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。
—	Bit 0	—	—

22. 5. 2. 5 I2C时钟寄存器 (I2C_TIMINGR)

I2C 时钟寄存器 (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** 该位用于预分频 I2CCLK，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} 只有在关闭 I2C(PE=0)时才能对该位进行设置。
—	Bit 27-24	—	—
SCLDEL	Bit 23-20	R/W	**数据设置时间** 该位用于在 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，NOSTRETCH=0 时，SCL 线在 T_{SCLDEL} 期间拉低。 T_{SCLDEL} =(SCLDEL + 1)x T_{PRESC} 注: T_{SCLDEL} 用于产生 $T_{SU:DAT}$ 时序。 只有在关闭 I2C(PE=0)时才能对该位进行设置。
SDADEL	Bit 19-16	R/W	**数据保持时间** 该位用于在 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，NOSTRETCH=0 时，SCL 线在 T_{SDADEL} 期间拉低。 T_{SDADEL}= SDADEL x T_{PRESC} 注: SDADEL 用于产生 $T_{HD:DAT}$ 时序。 只有在关闭 I2C(PE=0)时才能对该位进行设置。
SCLH	Bit 15-8	R/W	**SCL 高电平期间(主机模式)** 该位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。 只有在关闭 I2C(PE=0)时才能对该位进行设置。

SCLL	Bit 7-0	R/W	**SCL 低电平周期(主机模式)** 该位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 TBUF 和 $T_{SU:STA}$ 时序。 只有在关闭 I2C(PE=0)时才能对该位进行设置。

22. 5. 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	**累积时钟延长超时开启** 0:关闭累积时钟延长超时检测 1:开启累积时钟延长超时检测。 当 I2C 接口完成累积 SCL 延长超过 $T_{LOW:EXT}$ 时，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延长时间($T_{LOW:MEXT}$)。 在从机模式下，检测到从机累积时钟低延长时间($T_{LOW:SEXT}$)。 $T_{LOW:EXT}$ = (TIMEOUTB + 1)x 2048 x T_{I2CCLK} 注:当 TEXTEN=0 时才能写入该位。
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，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 低超时 1: TIMEOUTA 用于检测 SCL 和 SDA 高超时(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** 该位用于配置: - 当 TIDLE=0 时，SCL 低超时条件 $T_{TIMEOUT}$ $T_{TIMEOUT}$ =(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。

22. 5. 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，使用接收的地址更新该位。在 10 位地址的情况下，ADDCODE 提供 10 位标头与地址的 2 个 MSB。
DIR	Bit 16	R	**传输方向(从机模式)** 发生地址匹配事件(ADDRRI = 1)时更新此标志位。 0:写入传输，从机进入接收模式。 1:读取传输，从机进入发送模式。
BUSY	Bit 15	R	**总线忙** 该标志位表示总线上正在进行通信。检测到 START 条件时由硬件设置为 1。当检测到停止条件或 PE=0 时，由硬件清零。
—	Bit 14-12	—	—

TCR	Bit 11	R	**传输完成并重载** 当 RELOAD=1 且已将 NBYTES 笔数据传输完成时，此标志位由硬件设置为 1。当 NBYTES 位写入非零值时，由硬件清零。 注:当 PE=0 时，该位由硬件清零。该标志位仅用于主机，或者用于从模式且设置 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，该标志位由硬件设置为 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	**接收器下溢** 读取 I2C_STAT 后，由硬件清零。 0:接收器没有下溢 1:接收器下溢
RXOV	Bit 7	R/C_R	**接收器溢出** 读取 I2C_STAT 后，由硬件清零。 0:接收器没有溢出 1:接收器溢出
—	Bit 6	—	—
RXNE	Bit 5	R	**接收器非空** 0:接收器空 1:接收器非空
—	Bit 4	—	—
TXUD	Bit 3	R/C_R	**发送器下溢** 读取 I2C_STAT 后，由硬件清零。 0:发送器没有下溢 1:发送器下溢
TXOV	Bit 2	R/C_R	**发送器溢出** 读取 I2C_STAT 后，由硬件清零。 0:发送器没有溢出 1:发送器溢出
—	Bit 1	—	—
TXE	Bit 0	R	**发送器空** 0:发送器没有空

			1:发送器空

22. 5. 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	**数据包错误检查寄存器** 当 PECEN = 1 时，该位包含内部 PEC。 当 PECEN = 0 时，PEC 由硬件清零。

22. 5. 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 位接收器数据** 从 I2C 总线接收的数据字节。

22. 5. 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 位发送器数据** 要传输到 I2C 总线的数据字节。

22. 5. 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	**开启 SMBus 报警中断功能** 此位设置时，开启中断功能，硬件侦测 SMBus 报警事件时发生中断
TOUT	Bit 19	W1	**开启超时中断功能** 此位设置时，开启中断功能，硬件侦测超时事件时发生中断
PECE	Bit 18	W1	**开启 PEC 错误中断功能** 此位设置时，开启中断功能，硬件侦测 PEC 错误事件时发生中断
ARLO	Bit 17	W1	**开启仲裁丢失中断功能** 此位设置时，开启中断功能，硬件侦测仲裁丢失事件时发生中断
BERR	Bit 16	W1	**开启总线错误中断功能**

			此位设置时，开启中断功能，硬件侦测总线错误事件时发生中断
—	Bit 15	—	—
STOP	Bit 14	W1	**开启检测停止中断功能** 此位设置时，开启中断功能，硬件侦测检测停止位事件时发生中断
NACK	Bit 13	W1	**开启接收器 NACK 中断功能** 此位设置时，开启中断功能，硬件侦测接收器 NACK 事件时发生中断
ADDR	Bit 12	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
TCR	Bit 11	W1	**开启传输完成并重载中断功能** 此位设置时，开启中断功能，硬件侦测传输完成并重载事件时发生中断
TC	Bit 10	W1	**开启传输完成中断功能** 此位设置时，开启中断功能，硬件侦测传输完成事件时发生中断
—	Bit 9	—	—
RXUD	Bit 8	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
RXOV	Bit 7	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
—	Bit 6	—	—
RXNE	Bit 5	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
—	Bit 4	—	—
TXUD	Bit 3	W1	**开启发送器下溢中断功能** 此位设置时，开启中断功能，硬件侦测发送器下溢事件时发生中断
TXOV	Bit 2	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 1	—	—
TXE	Bit 0	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

22. 5. 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	**关闭 SMBus 报警中断功能** 此位设置时，关闭 SMBus 报警中断功能
TOUT	Bit 19	W1	**关闭超时中断功能** 此位设置时，关闭超时中断功能
PECE	Bit 18	W1	**关闭 PEC 错误中断功能** 此位设置时，关闭 PEC 错误中断功能
ARLO	Bit 17	W1	**关闭仲裁丢失中断功能** 此位设置时，关闭仲裁丢失中断功能
BERR	Bit 16	W1	**关闭总线错误中断功能** 此位设置时，关闭总线错误中断功能
—	Bit 15	—	—
STOP	Bit 14	W1	**关闭检测停止中断功能** 此位设置时，关闭检测停止位中断功能
NACK	Bit 13	W1	**关闭接收器 NACK 中断功能** 此位设置时，关闭接收器 NACK 中断功能
ADDR	Bit 12	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
TCR	Bit 11	W1	**关闭传输完成并重载中断功能** 此位设置时，关闭传输完成并重载中断功能
TC	Bit 10	W1	**关闭传输完成中断功能** 此位设置时，关闭传输完成中断功能
—	Bit 9	—	—
RXUD	Bit 8	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

RXOV	Bit 7	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
—	Bit 6	—	—
RXNE	Bit 5	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
—	Bit 4	—	—
TXUD	Bit 3	W1	**关闭发送器下溢中断功能** 此位设置时，关闭发送器下溢中断功能
TXOV	Bit 2	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 1	—	—
TXE	Bit 0	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

22. 5. 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 报警中断功能状态** 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	**检测停止位中断功能状态** 0:中断功能处于关闭状态 1:中断功能处于开启状态
NACK	Bit 13	R	**接收器 NACK 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	**传输完成并重载中断功能状态** 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 6	—	—
RXNE	Bit 5	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
—	Bit 4	—	—
TXUD	Bit 3	R	**发送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 1	—	—
TXE	Bit 0	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

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

22. 5. 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 报警，原始中断状态** 0:无发生中断 1:已发生中断
TOUT	Bit 19	R	**超时，原始中断状态** 0:无发生中断 1:已发生中断
PECE	Bit 18	R	**PEC 错误，原始中断状态** 0:无发生中断 1:已发生中断
ARLO	Bit 17	R	**仲裁丢失，原始中断状态** 0:无发生中断 1:已发生中断
BERR	Bit 16	R	**总线错误，原始中断状态** 0:无发生中断 1:已发生中断
—	Bit 15	—	—
STOP	Bit 14	R	**检测停止位，原始中断状态** 0:无发生中断 1:已发生中断
NACK	Bit 13	R	**接收器 NACK，原始中断状态** 0:无发生中断

			1:已发生中断
ADDR	Bit 12	R	**地址匹配，原始中断状态** 0:无发生中断 1:已发生中断
TCR	Bit 11	R	**传输完成并重载，原始中断状态** 0:无发生中断 1:已发生中断
TC	Bit 10	R	**传输完成，原始中断状态** 0:无发生中断 1:已发生中断
—	Bit 9	—	—
RXUD	Bit 8	R	**接收器下溢，原始中断状态** 0:无发生中断 1:已发生中断
RXOV	Bit 7	R	**接收器溢出，原始中断状态** 0:无发生中断 1:已发生中断
—	Bit 6	—	—
RXNE	Bit 5	R	**接收器非空，原始中断状态** 0:无发生中断 1:已发生中断
—	Bit 4	—	—
TXUD	Bit 3	R	**发送器下溢，原始中断状态** 0:无发生中断 1:已发生中断
TXOV	Bit 2	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 1	—	—
TXE	Bit 0	R	**发送器空，原始中断状态** 0:无发生中断 1:已发生中断

22. 5. 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 报警，标志位中断状态** 0:无发生中断 1:已发生中断
TOUT	Bit 19	R	**超时，标志位中断状态** 0:无发生中断 1:已发生中断
PECE	Bit 18	R	**PEC 错误，标志位中断状态** 0:无发生中断 1:已发生中断
ARLO	Bit 17	R	**仲裁丢失，标志位中断状态** 0:无发生中断 1:已发生中断
BERR	Bit 16	R	**总线错误，标志位中断状态** 0:无发生中断 1:已发生中断
—	Bit 15	—	—
STOP	Bit 14	R	**检测停止位，标志位中断状态** 0:无发生中断 1:已发生中断
NACK	Bit 13	R	**接收器 NACK，标志位中断状态** 0:无发生中断 1:已发生中断
ADDR	Bit 12	R	**地址匹配，标志位中断状态** 0:无发生中断 1:已发生中断
TCR	Bit 11	R	**传输完成并重载，标志位中断状态** 0:无发生中断

			1:已发生中断
TC	Bit 10	R	**传输完成，标志位中断状态** 0:无发生中断 1:已发生中断
—	Bit 9	—	—
RXUD	Bit 8	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
RXOV	Bit 7	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
—	Bit 6	—	—
RXNE	Bit 5	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
—	Bit 4	—	—
TXUD	Bit 3	R	**发送器下溢，标志位中断状态** 0:无发生中断 1:已发生中断
TXOV	Bit 2	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 1	—	—
TXE	Bit 0	R	**发送器空，标志位中断状态** 0:无发生中断 1:已发生中断

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

22. 5. 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 报警中断清除** 此位设置时，清除中断状态(I2C_RIF 与 I2C_IFM)
TOUT	Bit 19	C_W1	**超时中断清除** 此位设置时，清除中断状态(I2C_RIF 与 I2C_IFM)
PECE	Bit 18	C_W1	**PEC 错误中断清除** 此位设置时，清除中断状态(I2C_RIF 与 I2C_IFM)
ARLO	Bit 17	C_W1	**仲裁丢失中断清除** 此位设置时，清除中断状态(I2C_RIF 与 I2C_IFM)
BERR	Bit 16	C_W1	**总线错误中断清除** 此位设置时，清除中断状态(I2C_RIF 与 I2C_IFM)
—	Bit 15	—	—
STOP	Bit 14	C_W1	**检测停止位中断清除** 此位设置时，清除中断状态(I2C_RIF 与 I2C_IFM)
NACK	Bit 13	C_W1	**接收器 NACK 中断清除** 此位设置时，清除中断状态(I2C_RIF 与 I2C_IFM)
ADDR	Bit 12	C_W1	**地址匹配中断清除** 此位设置时，清除中断状态(I2C_RIF 与 I2C_IFM)
TCR	Bit 11	C_W1	**传输完成并重载中断清除** 此位设置时，清除中断状态(I2C_RIF 与

			I2C_IFM)
TC	Bit 10	C_W1	**传输完成中断清除** 此位设置时，清除中断状态(I2C_RIF 与 I2C_IFM)
—	Bit 9	—	—
RXUD	Bit 8	C_W1	**接收器下溢中断清除** 此位设置时，清除中断状态(I2C_RIF 与 I2C_IFM)
RXOV	Bit 7	C_W1	**接收器溢出中断清除** 此位设置时，清除中断状态(I2C_RIF 与 I2C_IFM)
—	Bit 6	—	—
RXNE	Bit 5	C_W1	**接收器非空中断清除** 此位设置时，清除中断状态(I2C_RIF 与 I2C_IFM)
—	Bit 4	—	—
TXUD	Bit 3	C_W1	**发送器下溢中断清除** 此位设置时，清除中断状态(I2C_RIF 与 I2C_IFM)
TXOV	Bit 2	C_W1	**发送器溢出中断清除** 此位设置时，清除中断状态(I2C_RIF 与 I2C_IFM)
—	Bit 1	—	—
TXE	Bit 0	C_W1	**发送器空中断清除** 此位设置时，清除中断状态(I2C_RIF 与 I2C_IFM)

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23章 串行外设接口(SPI)

23. 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。

23. 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

23. 3 SPI实现

本手册介绍了 SPI 实现的全部功能。

SPI 模式/特性	SPI1
Rx 和 TxFIFO 大小(N)[x 8 位]	4

表 23-1 SPI 特性

23. 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。

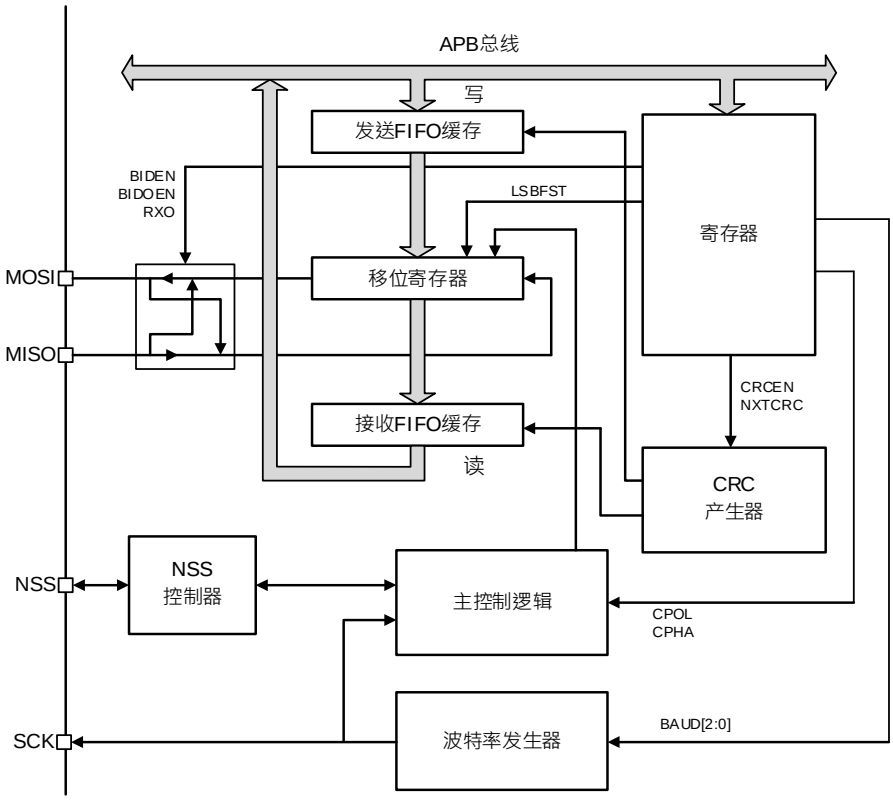


图 23-1 SPI 电路结构框图

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 23. 5. 13. 11)。

23. 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。

23. 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。

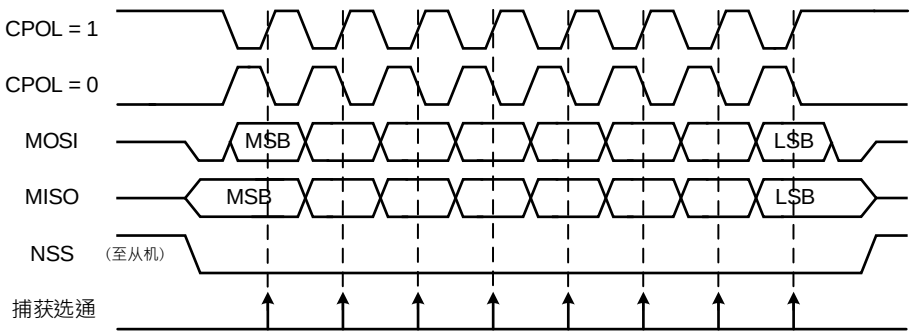
使用者通过组合 **SPI_CON1**.CPOL 和 **SPI_CON1**.CPHA 位来选择数据捕获的时钟边沿。

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

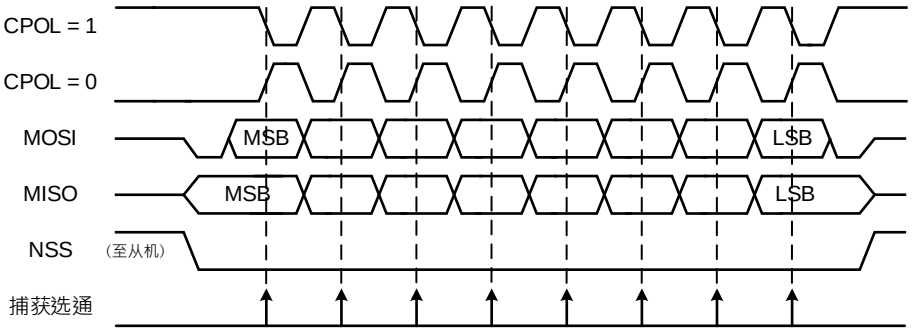


图 23-2　SPI 格式

注意:上图中显示的是当 SPI_CON1.LSBFST 处于复位状态时的时序图。

23. 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。

23. 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。

23. 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。

23. 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。

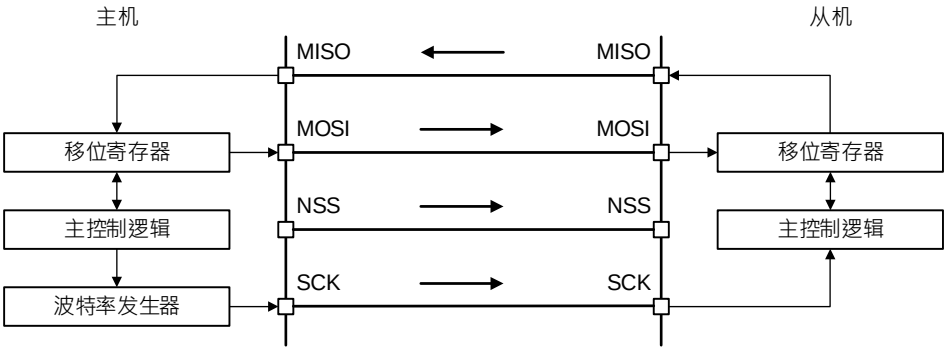


图 23-3 全双工通信

23. 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。

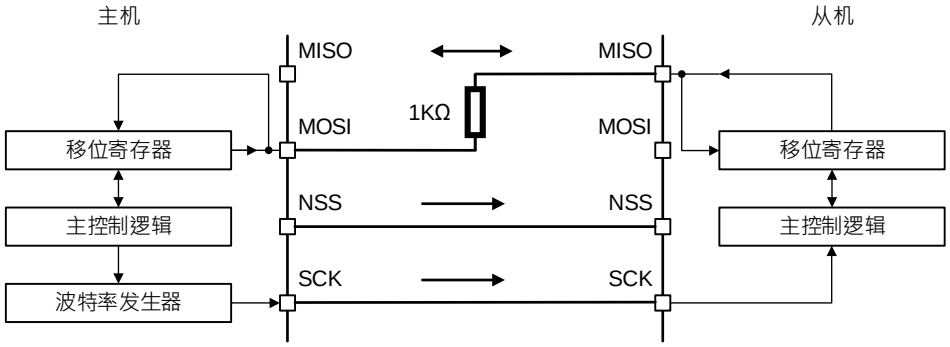


图 23-4 半双工通信

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 23. 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竞争时。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。

23. 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 标志位，因为开始通信后此标志位一直为 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。

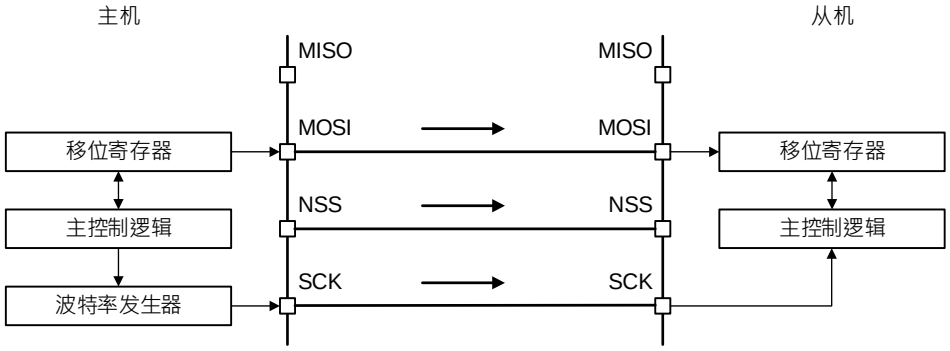


图 23-5 单工通信(主机模式下的只发送与从机模式下的只接收)

23. 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。

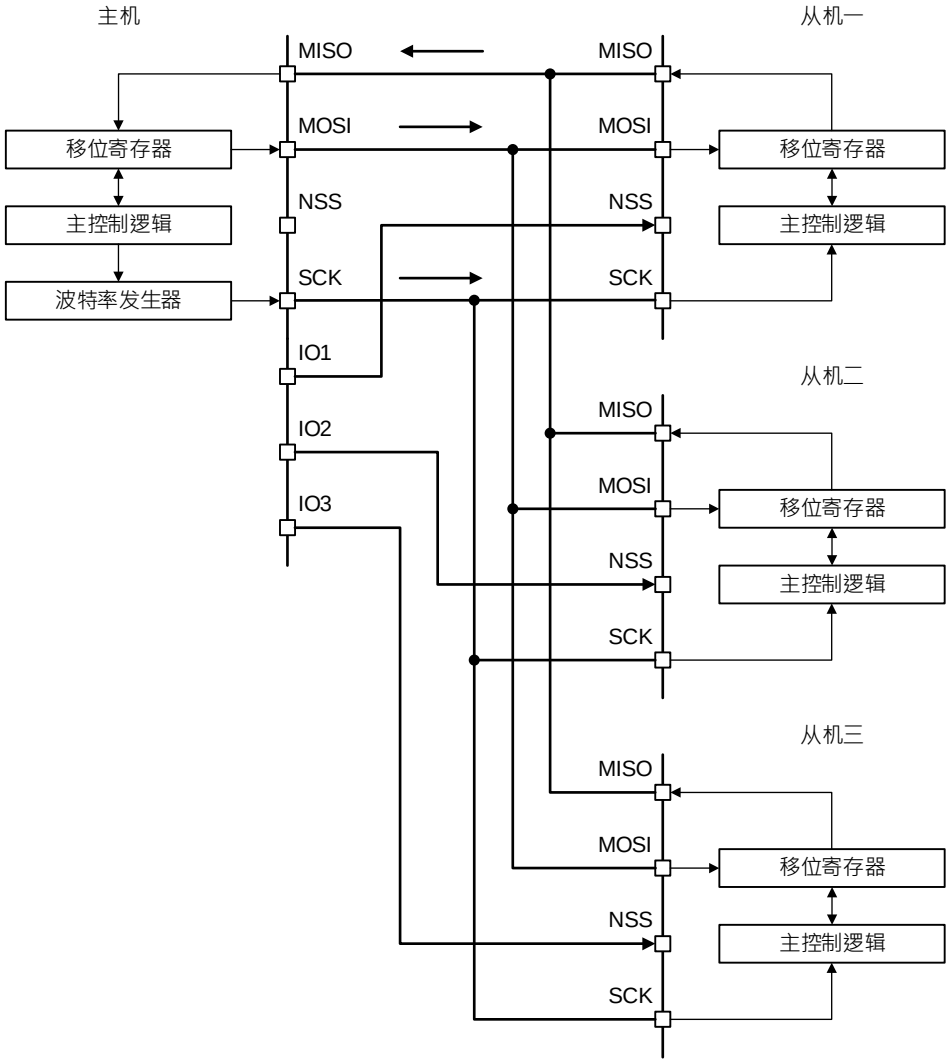


图 23-6 多从机通讯(一个主机和三个从机)

23. 5. 6 多主机通讯应用

多主机模式仅支持两个 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 23. 5. 13. 11)。使用者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)。

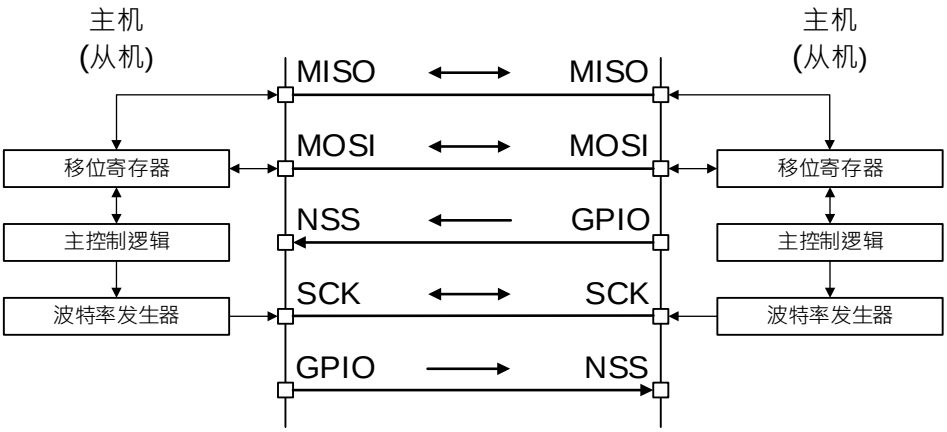


图 23-7 多主机通讯应用

23. 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 23. 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 标志位在发送 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。

23. 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 23-2)。
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 标志位在发送 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。

23. 5. 9 数据发送和接收

23. 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。

23. 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。

23. 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。

23. 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 23-8):

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 23-9):

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。

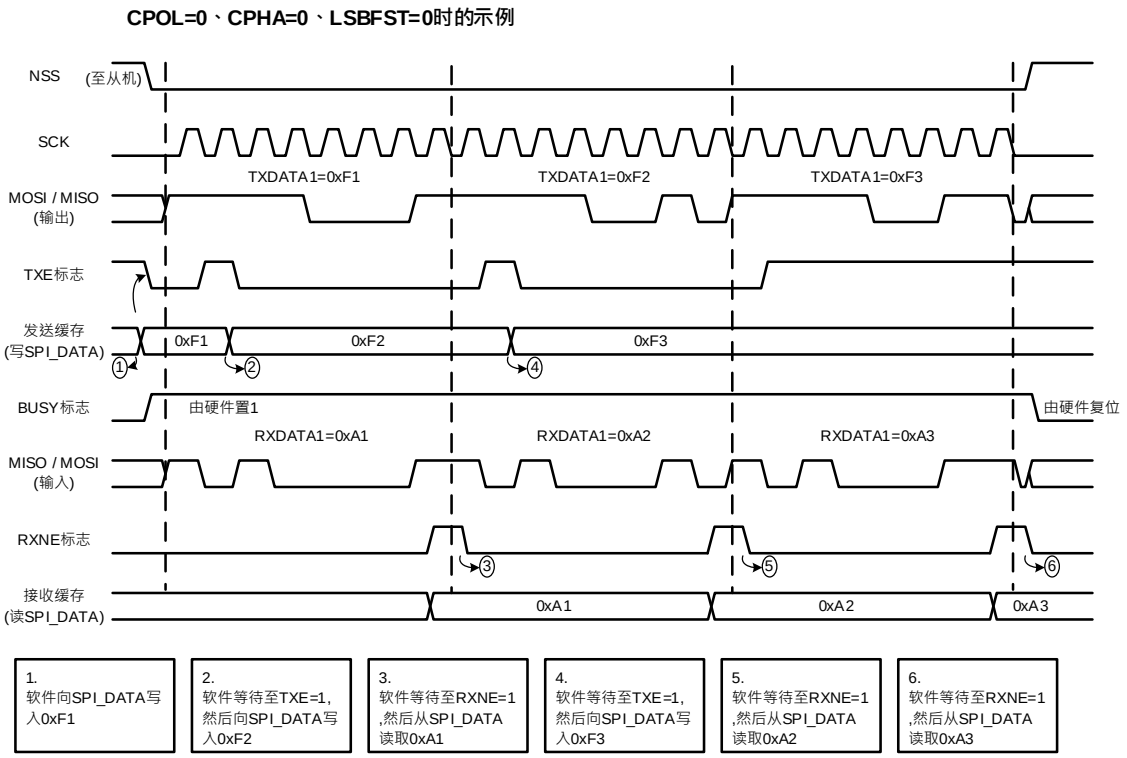


图 23-8 全双工通信(SPI_CON1.BIDEN=0 且 SPI_CON1.RXO=0)的 TXE、RXNE、BUSY 行为(直接存取操作模式在连续传输的情况下)

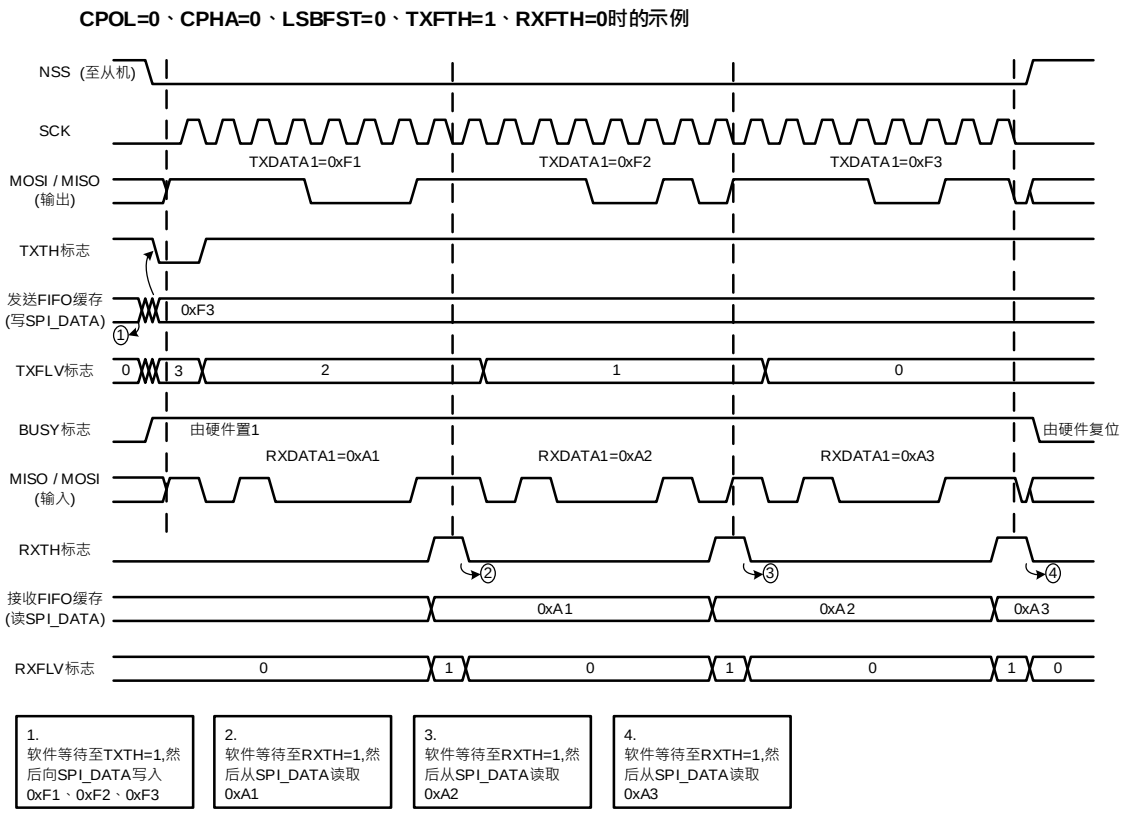


图 23-9　全双工通信(SPI_CON1.BIDEN=0 且 SPI_CON1.RXO=0)的 TXTH、RXTH、TXFLV、RXFLV、BUSY 行为(FIFO 缓存操作模式在连续传输的情况下)

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 23-10):

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 23-11):

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 标志位将置 1，因为始终不会读取接收的数据。

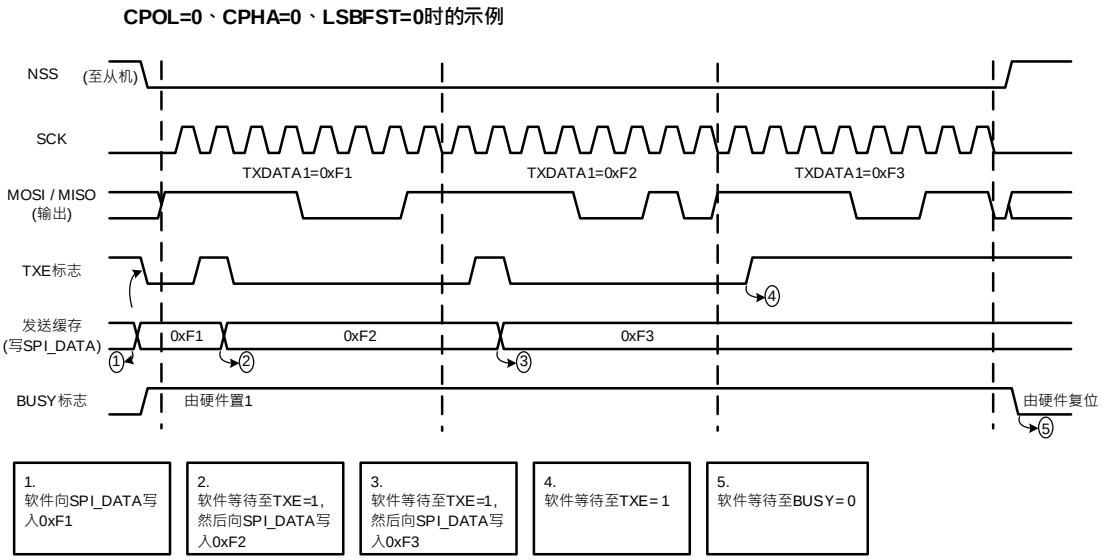


图 23-10 单工通信-只发送模式(SPI_CON1.BIDEN=0 且 SPI_CON1.RXO=0)的 TXE、BUSY 行为(直接存取操作模式在连续传输的情况下)

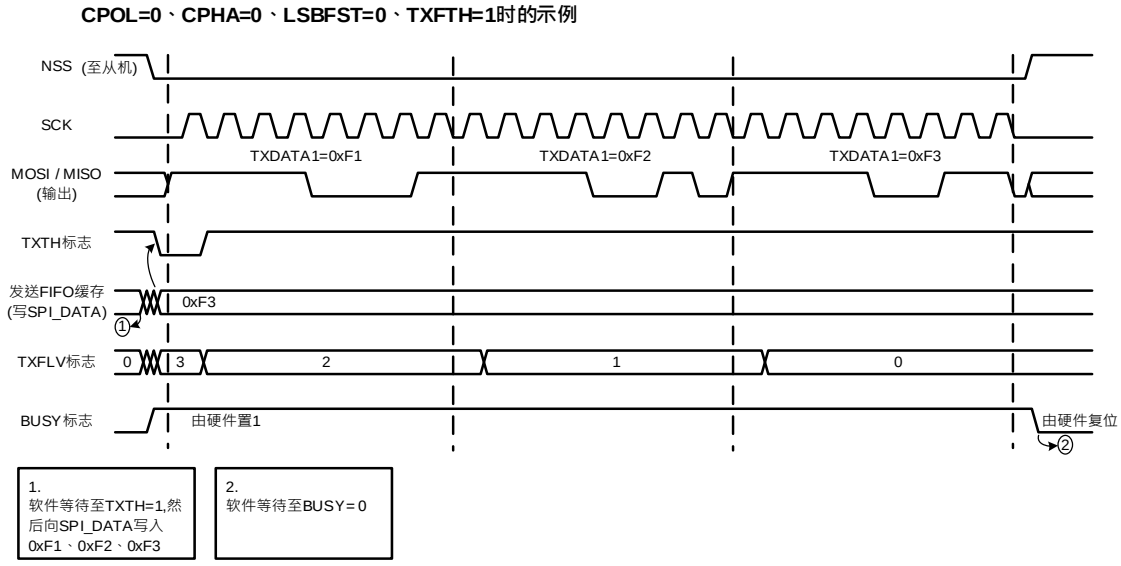


图 23-11 单工通信-只发送模式(SPI_CON1.BIDEN=0 且 SPI_CON1.RXO=0)的 TXTH、TXFLV、BUSY 行为(FIFO 缓存操作模式在连续传输的情况下)

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程相似，但是在 SPI 模块使能前，必须将 **SPI_CON1**.BIDEN 位和 **SPI_CON1**.BIDOEN 位置 1。

单工通信-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 23-12):

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 23-13):

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 22.4.10 SPI 关闭流程

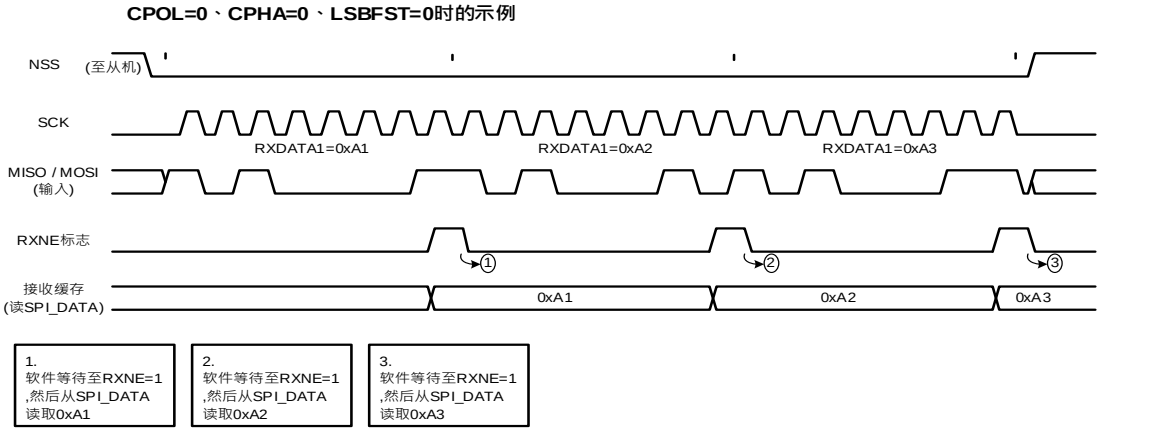


图 23-12　单工通信-只接收模式(SPI_CON1.BIDEN=0 且 SPI_CON1.RXO=1)的 RXNE 行为(直接存取操作模式在连续传输的情况下)

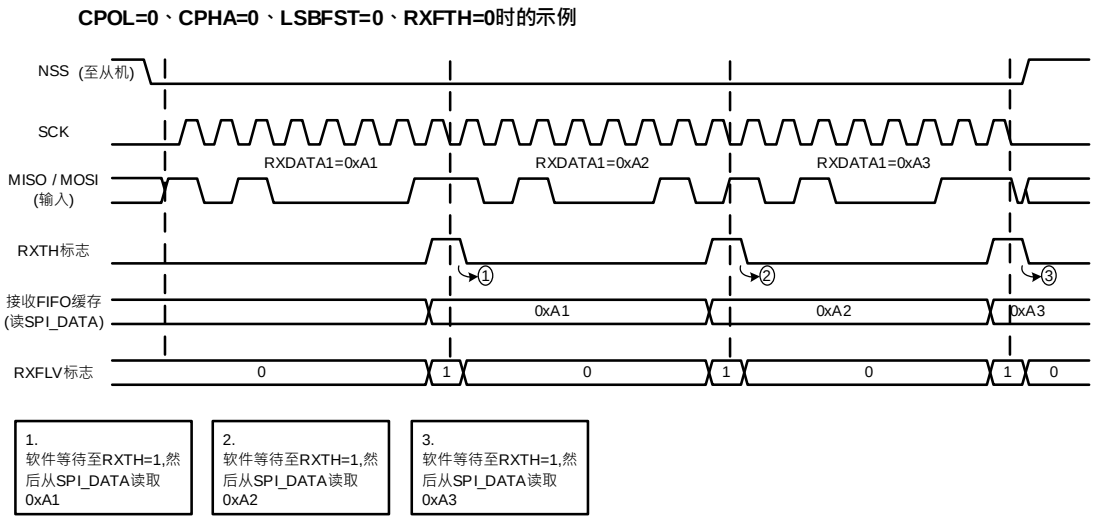


图 23-13　单工通信-只接收模式(SPI_CON1.BIDEN=0 且 SPI_CON1.RXO=1)的 RXTH、RXFLV 行为(FIFO 缓存操作模式在连续传输的情况下)

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。

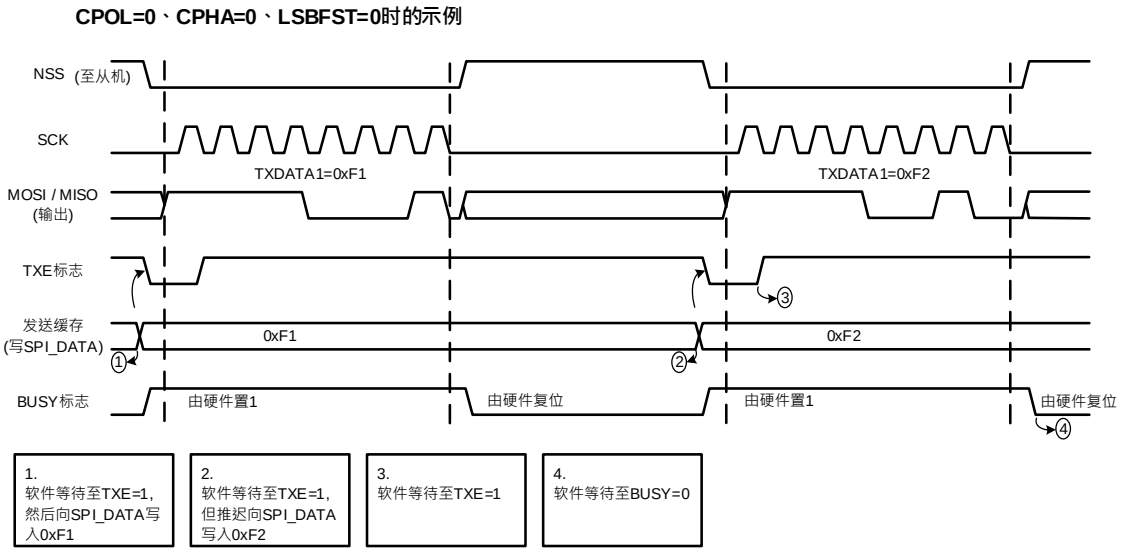


图 23-14　发送时(SPI_CON1.BIDEN =0 且 SPI_CON1.RXO=0)的 TXE、BUSY 行为(在间断传输的情况下)

23. 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:

23. 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)。

23. 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)。

23. 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** 寄存器中的 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)。

23. 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)。

23. 5. 11 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。

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。

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，以重新同步主机和从机双方的 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。

23. 5. 12 SPI状态标志位

23. 5. 12. 1 发送FIFO缓存为空(TXE)

此标志位置 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 标志位清零。

23. 5. 12. 2 发送FIFO缓存为满(TXF)

此标志位置 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 标志位清零。

23. 5. 12. 3 发送FIFO缓存上溢(TXOV)

当发送 FIFO 缓存已满时，使用者对 **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，并将此标志位置 1。对 **SPI_STAT** 寄存器执行读访问时，将 TXOV 标志位清零。

23. 5. 12. 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,并将此标志位置 1。对 **SPI_STAT** 寄存器执行读访问时，将 TXUD 标志位清零。

23. 5. 12. 5 发送FIFO缓存阈值(TXTH)

此标志位置 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 标志位清零。

23. 5. 12. 6 接收FIFO缓存为非空(RXNE)

此标志位置 1 时，表示接收 FIFO 缓存中存在有效的已接收数据。此时使用者可读取 **SPI_DATA** 寄存器，当读取后接收 FIFO 缓存中没有效数据时，此标志位会被清零。

23. 5. 12. 7 接收FIFO缓存为满(RXF)

此标志位置 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 标志位清零。

23. 5. 12. 8 接收FIFO缓存上溢(RXOV)

当接收 FIFO 缓存已满时，使用者没对 **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，同时将此标志位置 1。对 **SPI_STAT** 寄存器执行读访问时，会将 RXOV 标志位清零。

23. 5. 12. 9 接收FIFO缓存下溢(RXUD)

当接收 FIFO 缓存为空时，但使用者对 **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,并将此标志位置 1。对 **SPI_STAT** 寄存器执行读访问时，会将 RXUD 标志位清零。

23. 5. 12. 10 接收FIFO缓存阈值(RXTH)

此标志位置 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 标志位清零。

23. 5. 12. 11 通信忙(BUSY)

BUSY 标志位用于指示 SPI 通信的状态。此标志位由硬件置 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，使用者可检测 **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 标志位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位:

- 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 标志位在各通信之间处于低电平。
当通信连续时，BUSY 标志位在所有传输期间均保持高电平。

注意:请勿使用 BUSY 标志位处理每次数据发送或接收，最好改用 TXTH 标志位和 RXTH 标志位。

23. 5. 13 SPI中断事件

23. 5. 13. 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。

23. 5. 13. 2 发送FIFO缓存上溢(TXOV)

当发送 FIFO 缓存已满(**SPI_STAT**.TXF=1)时，使用者对 **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。

23. 5. 13. 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。

23. 5. 13. 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。

23. 5. 13. 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。

23. 5. 13. 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。

23. 5. 13. 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。

23. 5. 13. 8 接收FIFO缓存下溢(RXUD)

当接收 FIFO 缓存为空时，但使用者对 **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。

23. 5. 13. 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。

23. 5. 13. 10 CRC错误(CRCERR)

当 **SPI_CON1** 寄存器中的 CRCEN 位置 1 时，此标志位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。

23. 5. 13. 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。

23. 5. 13. 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。

23. 5. 14 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。

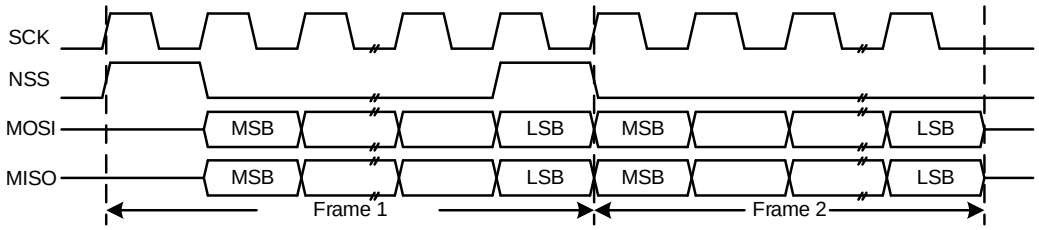


图 23-15　TI 格式

23. 6 特殊功能寄存器

23. 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

23. 6. 2 寄存器描述

23. 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。

23. 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		

—	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
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
—	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。
—	Bits 1-0	—	—

23. 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

23. 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。

23. 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。

23. 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。

23. 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。

23. 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	**帧格式错误中断开启** 0:写入 0 无效 1:开启帧格式错误中断
MODF	Bit 17	W1	**模式故障中断开启** 0:写入 0 无效 1:开启模式故障中断
CRCERR	Bit 16	W1	**CRC 错误中断开启** 0:写入 0 无效 1:开启 CRC 错误中断
—	Bit 15-13	—	—
RXTH	Bit 12	W1	**接收 FIFO 缓存超过阈值中断开启** 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 缓存下溢中断开启** 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 缓存上溢中断开启** 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 缓存满中断开启** 0:写入 0 无效 1:开启接收 FIFO 缓存满中断
RXNE	Bit 8	W1	**接收 FIFO 缓存非空中断开启** 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 缓存低于阈值中断开启**

			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 缓存下溢中断开启** 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缓存上溢中断开启** 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 缓存空中断开启** 0:写入 0 无效 1:开启发送 FIFO 缓存空中断

23. 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	**帧格式错误中断关闭** 0:写入0无效 1:关闭帧格式错误中断
MODF	Bit 17	W1	**模式故障中断关闭** 0:写入 0 无效 1:关闭模式故障中断
CRCERR	Bit 16	W1	**CRC 错误中断关闭** 0:写入 0 无效 1:关闭 CRC 错误中断
—	Bit 15-13	—	—
RXTH	Bit 12	W1	**接收 FIFO 缓存超过阈值中断关闭** 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 缓存下溢中断关闭**

			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 缓存上溢中断关闭** 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 缓存满中断关闭** 0:写入 0 无效 1:关闭接收 FIFO 缓存满中断
RXNE	Bit 8	W1	**接收FIFO缓存非空中断关闭** 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 缓存低于阈值中断关闭** 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缓存下溢中断关闭** 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 缓存上溢中断关闭** 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 缓存空中断关闭** 0:写入 0 无效 1:关闭发送 FIFO 缓存空中断

23. 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: 中断功能处于关闭状态 1: 中断功能处于开启状态
MODF	Bit 17	R	**模式故障中断功能状态** 0: 中断功能处于关闭状态 1: 中断功能处于开启状态
CRCERR	Bit 16	R	**CRC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 15-13	—	—
RXTH	Bit 12	R	**接收FIFO缓存超过阈值中断功能状态** 0: 中断功能处于关闭状态 1: 中断功能处于开启状态
RXUD	Bit 11	R	**接收FIFO缓存下溢中断功能状态** 0: 中断功能处于关闭状态 1: 中断功能处于开启状态
RXOV	Bit 10	R	**接收FIFO缓存上溢中断功能状态** 0: 中断功能处于关闭状态 1: 中断功能处于开启状态
RXF	Bit 9	R	**接收 FIFO 缓存满中断功能状态** 0: 中断功能处于关闭状态 1: 中断功能处于开启状态
RXNE	Bit 8	R	**接收FIFO缓存非空中断功能状态** 0: 中断功能处于关闭状态 1: 中断功能处于开启状态
—	Bit 7-5	—	—
TXTH	Bit 4	R	**发送FIFO缓存低于阈值中断功能状态**

			0: 中断功能处于关闭状态 1: 中断功能处于开启状态
TXUD	Bit 3	R	**发送FIFO缓存下溢中断功能状态** 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FIFO缓存上溢中断功能状态** 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FIFO缓存空中断功能状态** 0: 中断功能处于关闭状态 1: 中断功能处于开启状态

23. 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

23. 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开启 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开启 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开启 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开启 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开启 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开启 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开启 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开启 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开启 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开启 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开启 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开启 1:产生中断

23. 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 使用者指南(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:

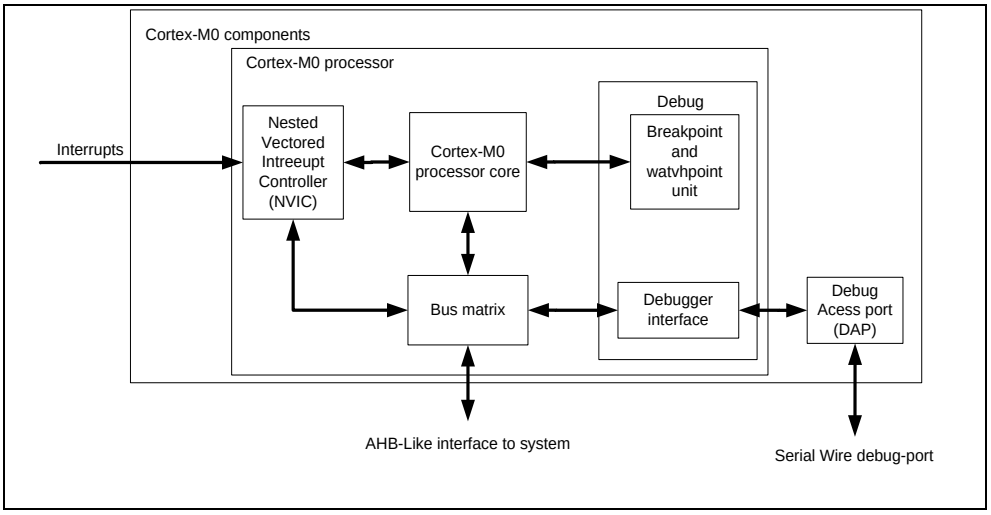
◇ 结构简单，容易学习和编程

◇ 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**。

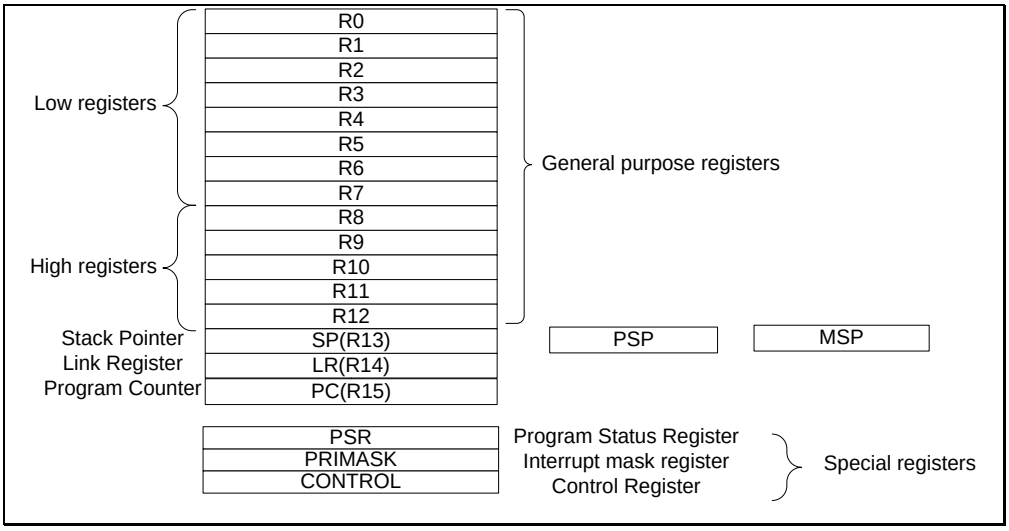
在线程模式下，CONTROL 寄存器控制着处理器使用主堆栈还是进程堆栈，见**处理器－控制寄存器章节**。在处理器模式下，处理器总是使用主堆栈。处理器操作的选择如下:

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

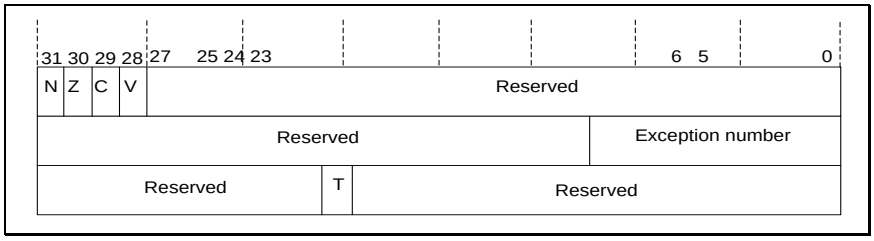
程序状态寄存器(PSR)由下列三种寄存器组合而成:

◇ 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

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”

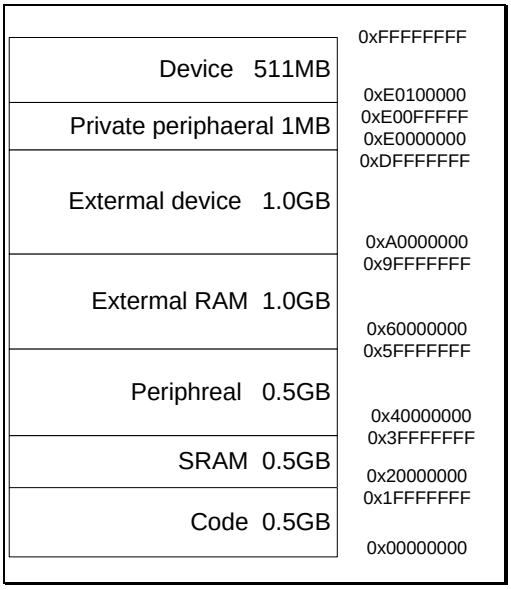
◇ 内部函数“Intrinsic functions”

◇ 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。

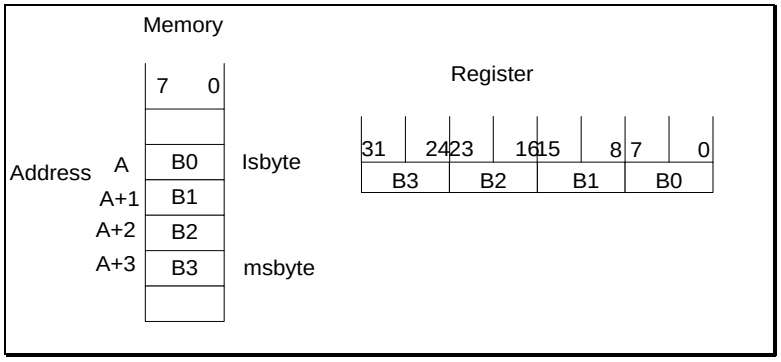
处理器保留与所有其他事务相关的事务顺序。

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—超级使用者调用(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:

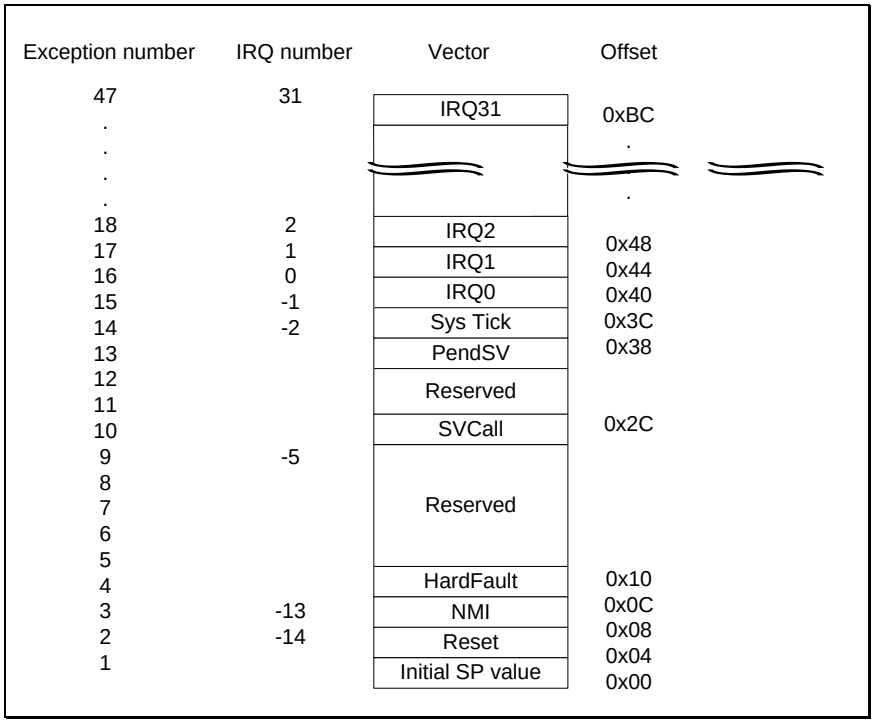
中断服务程序(ISR)—中断 IRQ0~IRQ31 是由 ISR 来处理的异常

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

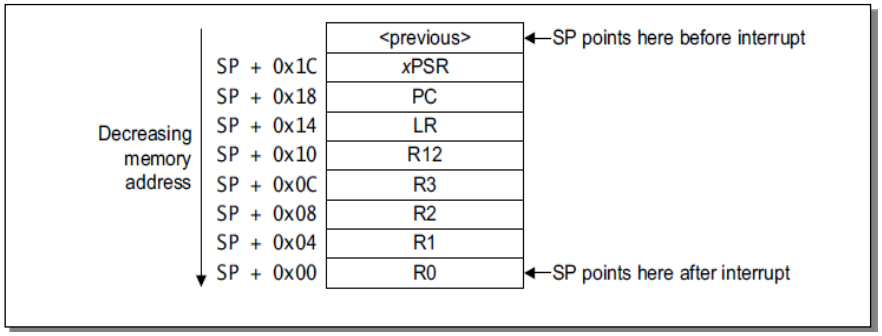
当有一个优先级足够高的挂起异常存在，并且满足下面的任何一个条件，就进入异处理:

◇　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	超级使用者调用	-	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，则使用者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，或者存在着使用者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

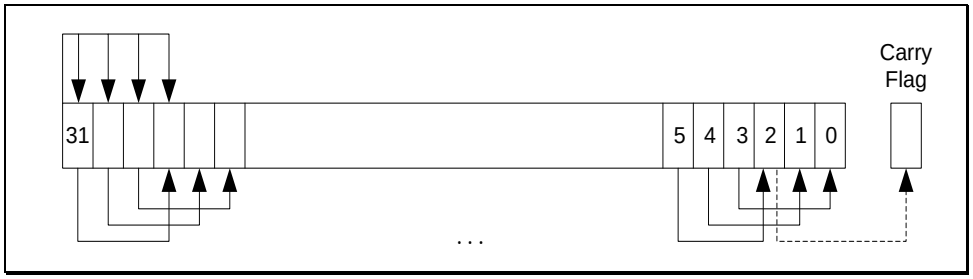
算术右移 n 位的操作是将 Rm 寄存器左边的 32-n 个位向右移动 n 位，结果是寄存器右边有 32-n 个位，然后再将寄存器位[31]的原始值复制到结果寄存器左边的 n 位中，请看图片 **ASR #3**。

使用者可以使用 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。

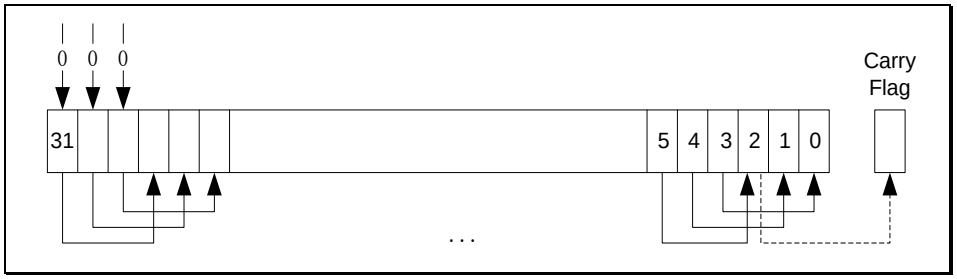
如果寄存器 Rm 值为无符号的整数，使用者可以使用 LSR 操作来对其值进行除以 2^n 的操作。

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。

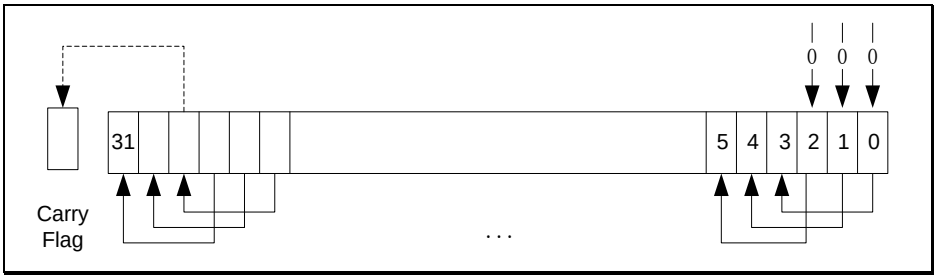
如果寄存器 Rm 值为无符号的整数或是有符号 2 的补码整数值，使用者可以使用 LSL 操作来对其值与 2^n 进行乘法操作。溢出会在无警告提示下发生。

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

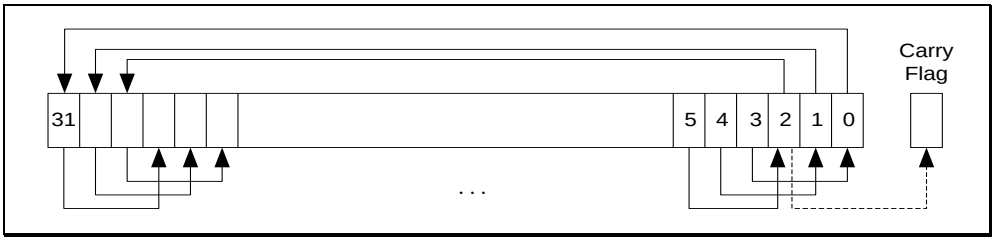
循环右移 n 位的操作是将 Rm 寄存器左边的 32-n 个位向右移动 n 位，结果是寄存器右边有 32-n 个位，然后再将寄存器右边的 n 个位移动到结果寄存器的左边的 n 个位中。请看图片 ROR #3。

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，使用者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。

如果使用者使用 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	超级使用者调用	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

超级使用者调用。

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; 超级使用者调用(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

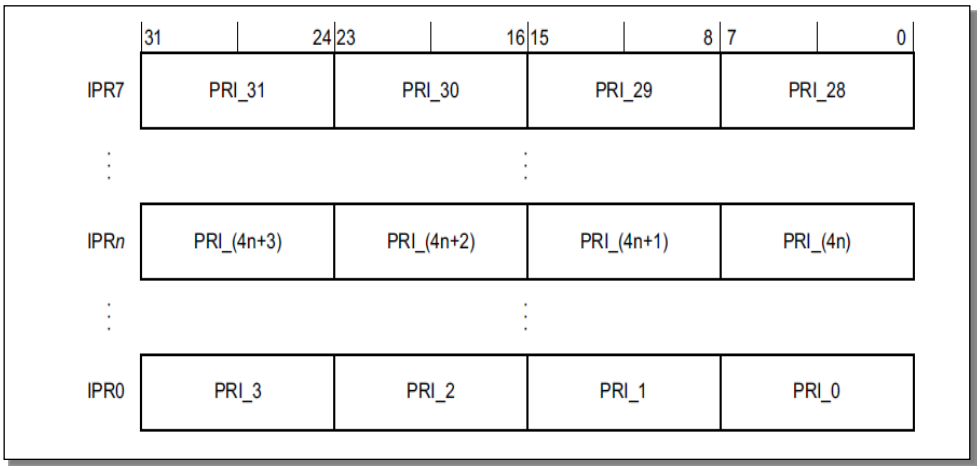
附录表 1-29　ICPR 位分配

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。使用者可以将 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	超级使用者调用	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	2024-12-09	初版发布
V1.1	2025-04-24	删除 ES32F0943LT 型号，修正描述格式，及部分笔误等