

## 基于ARM®32位的Cortex™-M4的无线蓝牙微控制器，带256 K字节闪存、sLib、USBFS、11个定时器、1个ADC、2个比较器、6个通信接口

### ■ 无线蓝牙模块

- 符合蓝牙®技术联盟5.0双模规范
- 低功耗2.4GHz收发
- 时钟：16 MHz晶振、64 MHz PLL、32 kHz内部时钟
- 外设：8个GPIO带2通道PWM和2个UART，其中UART21与MCU USART3连接

### ■ 内核：ARM®32位的Cortex™-M4 CPU

- 最高150 MHz工作频率，带存储器保护单元(MPU)，内建单周期乘法和硬件除法
- 具有DSP指令集

### ■ 存储器

- 256 K字节的内部闪存存储器
- 18 K字节的启动程序存储器可作启动加载程序(Bootloader)用外，也可一次性配置成一般用户程序和数据区
- sLib：将指定之主存储区设为执行代码安全库区，此区代码仅能调用无法读取
- 32 K字节的SRAM

### ■ 电源管理

- 2.6至3.6伏供电
- 上电/下电复位(POR/LVR)，可编程电压监测器(PVM)
- 低功耗模式：睡眠、深度睡眠和待机模式
- V<sub>BAT</sub>为LEXT、ERTC和20个32位电池供电寄存器(BPR)供电

### ■ 时钟和复位管理

- 外部主时钟输入
- 内嵌经出厂校准的48 MHz时钟(HICK)(25 °C达1%精度，-40 °C至+105 °C达2.5%精度)，带自动时钟校准功能(ACC)
- PLL可灵活配置31 ~ 500倍频和1 ~ 15分频系数
- 32 kHz晶振(LEXT)
- 40 kHz晶振(LICK)

### ■ 1个12位A/D转换器，0.5 μs转换时间(多达16个输入通道)

- 转换范围：0至3.6 V
- 一组采样和保持功能

- 温度传感器

### ■ 2个比较器

### ■ DMA：12通道DMA控制器

- 支持的外设：定时器、ADC、SPI、I<sup>2</sup>C、和USART

### ■ 调试模式

- 串行单线调试(SWD)

### ■ MCU快速GPIO端口

- 所有I/O口可以映像到16个外部中断
- 几乎所有I/O口可容忍5V输入信号
- 所有I/O口均为快速I/O，寄存器存取速度最高f<sub>AHB</sub>

### ■ 多达11个定时器

- 6个16位定时器+2个32位定时器，每个定时器有多达4个用于输入捕获/输出比较/PWM或脉冲计数的通道并支持编码器模式
- 2个看门狗定时器
- 系统嘀嗒定时器：24位自减型计数器

### ■ ERTC：增强型ERTC

### ■ 6个通信接口

- I<sup>2</sup>C接口(支持SMBus/PMBus)
- 3个USART(支持ISO7816, LIN, IrDA接口和调制解调控制)
- SPI接口
- CAN接口(2.0B主动)
- USB2.0全速设备/主机/OTG控制器

### ■ CRC计算单元

### ■ 96位的芯片唯一代码 (UID)

### ■ 封装

- QFN48 7 x 7 mm

### ■ 选型列表

内部闪存存储器	型号
256 K 字节	AT32WB415CCU7-7

## 目 录

1 系统构架.....	23
1.1 系统概述 .....	25
1.1.1 ARM Cortex™-M4处理器 .....	25
1.1.2 位带 .....	25
1.1.3 中断和异常向量 .....	27
1.1.4 系统嘀嗒定时器（SysTick） .....	30
1.1.5 复位流程 .....	30
1.2 寄存器的缩写说明 .....	31
1.3 器件特征信息 .....	32
1.3.1 闪存容量寄存器 .....	32
1.3.2 器件电子签名 .....	32
2 存储器资源 .....	33
2.1 内部存储器地址映射 .....	33
2.2 Flash存储器 .....	33
2.3 SRAM存储器 .....	34
2.4 外设地址映射 .....	34
3 电源控制（PWC） .....	36
3.1 简介 .....	36
3.2 主要特点 .....	36
3.3 上电下电复位 .....	36
3.4 电压监测器（PVM） .....	37
3.5 电源域划分 .....	37
3.6 省电模式 .....	38
3.7 PWC寄存器 .....	39
3.7.1 电源控制寄存器（PWC_CTRL） .....	39
3.7.2 电源控制及状态寄存器（PWC_CTRLSTS） .....	40
4 时钟和复位管理（CRM） .....	41
4.1 时钟 .....	41
4.1.1 时钟源 .....	41
4.1.2 系统时钟 .....	42
4.1.3 外设时钟 .....	42
4.1.4 时钟失效检测 .....	42
4.1.5 自动滑顺频率切换 .....	42
4.1.6 内部时钟输出 .....	42
4.1.7 中断 .....	42
4.2 复位 .....	43
4.2.1 系统复位 .....	43
4.2.2 电池供电域复位 .....	43
4.3 CRM寄存器 .....	43
4.3.1 时钟控制寄存器（CRM_CTRL） .....	44
4.3.2 时钟配置寄存器（CRM_CFG） .....	45

4.3.3	时钟中断寄存器（CRM_CLKINT）	47
4.3.4	APB2外设复位寄存器（CRM_APB2RST）	48
4.3.5	APB1外设复位寄存器（CRM_APB1RST）	49
4.3.6	AHB外设时钟使能寄存器（CRM_AHBEN）	50
4.3.7	APB2外设时钟使能寄存器（CRM_APB2EN）	51
4.3.8	APB1外设时钟使能寄存器（CRM_APB1EN）	52
4.3.9	电池供电域控制寄存器（CRM_BPDC）	53
4.3.10	控制/状态寄存器（CRM_CTRLSTS）	54
4.3.11	AHB外设复位寄存器（CRM_AHBRST）	55
4.3.12	PLL配置寄存器（CRM_PLL）	55
4.3.13	额外寄存器（CRM_MISC1）	56
4.3.14	OTG_FS扩展控制寄存器（CRM_OTG_EXTCTRL）	56
4.3.15	额外寄存器（CRM_MISC2）	56
5	内嵌闪存控制器（FLASH）	58
5.1	FLASH介绍	58
5.2	主存储器操作	59
5.2.1	解锁/锁定	59
5.2.2	擦除	59
5.2.3	编程	61
5.2.4	读取	62
5.3	主存扩展区操作	62
5.4	用户系统数据区操作	63
5.4.1	解锁/锁定	63
5.4.2	擦除	63
5.4.3	编程	64
5.4.4	读取	65
5.5	闪存保护	65
5.5.1	访问保护	65
5.5.2	擦写保护	66
5.6	特殊功能	67
5.6.1	安全库区设定	67
5.6.2	启动程序代码区域作为主存扩展使用	68
5.6.3	CRC校验	68
5.7	FLASH寄存器	68
5.7.1	闪存性能选择寄存器（FLASH_PSR）	69
5.7.2	闪存解锁寄存器（FLASH_UNLOCK）	69
5.7.3	闪存用户系统数据解锁寄存器（FLASH_USD_UNLOCK）	69
5.7.4	闪存状态寄存器（FLASH_STS）	70
5.7.5	闪存控制寄存器（FLASH_CTRL）	70
5.7.6	闪存地址寄存器（FLASH_ADDR）	71
5.7.7	用户系统数据寄存器（FLASH_USD）	71
5.7.8	擦除编程保护状态寄存器（FLASH_EPPS）	71

5.7.9	闪存安全库区状态寄存器0 (SLIB_STS0) .....	71
5.7.10	闪存安全库区状态寄存器1 (SLIB_STS1) .....	72
5.7.11	闪存安全库区密码清除寄存器 (SLIB_PWD_CLR) .....	72
5.7.12	闪存安全库区额外状态寄存器 (SLIB_MISC_STS) .....	72
5.7.13	闪存CRC校验地址寄存器 (FLASH_CRC_ADDR) .....	73
5.7.14	闪存CRC校验控制寄存器 (FLASH_CRC_CTRL) .....	73
5.7.15	闪存CRC校验结果寄存器 (FLASH_CRC_CHK) .....	73
5.7.16	闪存安全库区密码设定寄存器 (SLIB_SET_PWD) .....	73
5.7.17	闪存安全库区地址设定寄存器 (SLIB_SET_RANGE) .....	73
5.7.18	主存扩展存储区域安全库区设定寄存器 (EM_SLIB_SET) .....	74
5.7.19	启动程序代码区模式设定寄存器 (BTM_MODE_SET) .....	74
5.7.20	闪存安全库区解锁寄存器 (SLIB_UNLOCK) .....	75
<b>6</b>	<b>通用功能输入输出 (GPIO)</b> .....	<b>76</b>
6.1	简介 .....	76
6.2	功能描述 .....	76
6.2.1	GPIO结构 .....	76
6.2.2	GPIO复位状态 .....	76
6.2.3	通用功能输入配置 .....	76
6.2.4	模拟输入/输出配置 .....	77
6.2.5	通用功能输出配置 .....	77
6.2.6	I/O端口保护 .....	77
6.3	GPIO寄存器 .....	77
6.3.1	GPIO配置低寄存器 (GPIOx_CFGLR) (x=A..F) .....	78
6.3.2	GPIO配置高寄存器 (GPIOx_CFGHR) (A..F) .....	78
6.3.3	GPIO输入数据寄存器 (GPIOx_IDT) (x=A..F) .....	78
6.3.4	GPIO输出数据寄存器 (GPIOx_ODT) (x=A..F) .....	78
6.3.5	GPIO设置/清除寄存器 (GPIOx_SCR) (x=A..F) .....	79
6.3.6	GPIO清除寄存器 (GPIOx_CLR) (x=A..F) .....	79
6.3.7	GPIO写保护寄存器 (GPIOx_WPR) (x=A..F) .....	79
<b>7</b>	<b>复用功能输入输出 (IOMUX)</b> .....	<b>80</b>
7.1	简介 .....	80
7.2	功能描述 .....	80
7.2.1	IOMUX结构 .....	80
7.2.2	复用功能输入配置 .....	80
7.2.3	复用功能输出或双向复用功能配置 .....	81
7.2.4	外设复用功能管脚配置 .....	81
7.2.5	IOMUX映射优先级 .....	81
7.2.5.1	硬件抢占功能 .....	81
7.2.5.2	调试端口优先 .....	81
7.2.5.3	其他外设输出优先级关系 .....	82
7.2.6	外部中断/唤醒线 .....	82
7.3	IOMUX寄存器 .....	82

7.3.1	事件输出控制寄存器 (IOMUX_EVTOUT) .....	83
7.3.2	IO复用重映射寄存器 (IOMUX_REMAP) .....	83
7.3.3	复用外部中断配置寄存器1 (IOMUX_EXINTC1) .....	84
7.3.4	复用外部中断配置寄存器2 (IOMUX_EXINTC2) .....	85
7.3.5	复用外部中断配置寄存器3 (IOMUX_EXINTC3) .....	85
7.3.6	复用外部中断配置寄存器4 (IOMUX_EXINTC4) .....	86
7.3.7	IO复用重映射寄存器2 (IOMUX_REMAP2) .....	87
7.3.8	IO复用重映射寄存器3 (IOMUX_REMAP3) .....	87
7.3.9	IO复用重映射寄存器4 (IOMUX_REMAP4) .....	87
7.3.10	IO复用重映射寄存器5 (IOMUX_REMAP5) .....	88
7.3.11	IO复用重映射寄存器6 (IOMUX_REMAP6) .....	88
7.3.12	IO复用重映射寄存器7 (IOMUX_REMAP7) .....	89
7.3.13	IO复用重映射寄存器8 (IOMUX_REMAP8) .....	89
<b>8</b>	<b>外部中断/事件控制器 (EXINT) .....</b>	<b>91</b>
8.1	EXINT介绍 .....	91
8.2	功能描述和配置流程 .....	91
8.3	EXINT寄存器描述 .....	92
8.3.1	中断使能寄存器 (EXINT_INTEN) .....	92
8.3.2	事件使能寄存器 (EXINT_EVTEN) .....	92
8.3.3	极性配置寄存器1 (EXINT_POLCFG1) .....	92
8.3.4	极性配置寄存器2 (EXINT_POLCFG2) .....	92
8.3.5	软件触发寄存器 (EXINT_SWTRG) .....	93
8.3.6	中断状态寄存器 (EXINT_INTSTS) .....	93
<b>9</b>	<b>DMA控制器 (DMA) .....</b>	<b>94</b>
9.1	简介 .....	94
9.2	特性 .....	94
9.3	功能描述 .....	94
9.3.1	通道配置 .....	94
9.3.2	握手机制 .....	95
9.3.3	仲裁 .....	95
9.3.4	可编程数据传输宽度 .....	95
9.3.5	错误事件 .....	96
9.3.6	中断 .....	96
9.3.7	DMA固定请求映射 .....	97
9.3.8	DMA弹性请求映射 .....	97
9.4	DMA寄存器 .....	98
9.4.1	DMA状态寄存器 (DMA_STS) .....	99
9.4.2	DMA状态清除寄存器 (DMA_CLR) .....	101
9.4.3	DMA通道x配置寄存器 (DMA_CxCTRL) (x = 1...7) .....	103
9.4.4	DMA通道x数据传输量寄存器 (DMA_CxDTCNT) (x = 1...7) .....	104
9.4.5	DMA通道x外设地址寄存器 (DMA_CxPADDR) (x = 1...7) .....	104
9.4.6	DMA通道x存储器地址寄存器 (DMA_CxMADDR) (x = 1...7) .....	104

9.4.7 通道来源寄存器0 (DMA_SRC_SEL0) .....	105
9.4.8 通道来源寄存器1 (DMA_SRC_SEL1) .....	105
<b>10 CRC计算单元 (CRC) .....</b>	<b>106</b>
10.1 CRC介绍 .....	106
10.2 CRC寄存器 .....	106
10.2.1 数据寄存器 (CRC_DT) .....	106
10.2.2 通用数据寄存器 (CRC_CDT) .....	106
10.2.3 控制寄存器 (CRC_CTRL) .....	106
10.2.4 初始化寄存器 (CRC_IDT) .....	107
<b>11 I<sup>2</sup>C接口 .....</b>	<b>108</b>
11.1 I <sup>2</sup> C简介 .....	108
11.2 I <sup>2</sup> C主要特点 .....	108
11.3 I <sup>2</sup> C总线特性 .....	108
11.4 I <sup>2</sup> C接口 .....	108
11.4.1 I <sup>2</sup> C从机通信流程 .....	111
11.4.2 I <sup>2</sup> C主机通信流程 .....	112
11.4.3 利用DMA传输 .....	117
11.4.4 SMBus .....	118
11.4.5 I <sup>2</sup> C中断请求 .....	119
11.4.6 I <sup>2</sup> C调试模式 .....	120
11.5 I <sup>2</sup> C寄存器描述 .....	120
11.5.1 控制寄存器1(I2C_CTRL1) .....	120
11.5.2 控制寄存器2(I2C_CTRL2) .....	121
11.5.3 自身地址寄存器1(I2C_OADDR1) .....	122
11.5.4 自身地址寄存器2(I2C_OADDR2) .....	122
11.5.5 数据寄存器(I2C_DT) .....	122
11.5.6 状态寄存器1(I2C_STS1) .....	123
11.5.7 状态寄存器2(I2C_STS2) .....	124
11.5.8 时钟控制寄存器(I2C_CLKCTRL) .....	125
<b>12 通用同步异步收发器 (USART) .....</b>	<b>126</b>
12.1 USART介绍 .....	126
12.2 全双工半双工选择器简述和配置流程 .....	127
12.3 模式选择器简述和配置流程 .....	127
12.3.1 模式选择器简述 .....	127
12.3.2 模式选择器配置方法 .....	128
12.4 USART帧格式简述和配置流程 .....	128
12.5 DMA传输简述和配置流程 .....	128
12.5.1 DMA发送配置流程 .....	128
12.5.2 DMA接收配置流程 .....	129
12.6 波特率发生器简述及配置流程 .....	129
12.6.1 波特率发生器简述 .....	129
12.6.2 波特率发生器配置方法 .....	129

12.7	发送器简述和配置流程	129
12.7.1	发送器简述	129
12.7.2	发送器配置流程	130
12.8	接收器简述和配置流程	130
12.8.1	接收器简述	130
12.8.2	接收器配置流程	130
12.8.3	起始侦测和噪声检测	131
12.9	中断	132
12.10	I/O管脚控制	132
12.11	USART寄存器描述	132
12.11.1	状态寄存器 ( USART_STS )	133
12.11.2	数据寄存器 ( USART_DT )	134
12.11.3	波特比率寄存器 ( USART_BAUDR )	134
12.11.4	控制寄存器1 ( USART_CTRL1 )	134
12.11.5	控制寄存器2 ( USART_CTRL2 )	135
12.11.6	控制寄存器3 ( USART_CTRL3 )	136
12.11.7	保护时间和预分频寄存器 ( USART_GDIV )	137
13	串行外设接口 ( SPI )	138
13.1	串行外设接口 ( SPI ) 简介	138
13.2	SPI功能描述	138
13.2.1	SPI简述	138
13.2.2	全双工半双工选择器简述和配置流程	139
13.2.3	CS控制器简述和配置流程	140
13.2.4	SPI_SCK控制器简述和配置流程	141
13.2.5	CRC简述和配置流程	141
13.2.6	DMA传输简述和配置流程	142
13.2.7	发送器简述和配置流程	142
13.2.8	接收器简述和配置流程	143
13.2.9	Motorola模式通信时序	143
13.2.10	中断	146
13.2.11	IO管脚控制	146
13.2.12	注意事项	146
13.3	SPI寄存器	146
13.3.1	SPI控制寄存器1 ( SPI_CTRL1 )	146
13.3.2	SPI控制寄存器2 ( SPI_CTRL2 )	148
13.3.3	SPI状态寄存器 ( SPI_STS )	148
13.3.4	SPI数据寄存器 ( SPI_DT )	149
13.3.5	SPICRC多项式寄存器 ( SPI_CPOLY )	149
13.3.6	SPIRxCRC寄存器 ( SPI_RCRC )	149
13.3.7	SPITxCRC寄存器 ( SPI_TCRC )	150
14	定时器 ( TIMER )	151
14.1	通用定时器 ( TMR2到TMR5 )	151

14.1.1 TMRx简介 .....	151
14.1.2 TMRx主要功能 .....	151
14.1.3 TMRx功能描述 .....	152
14.1.3.1 计数时钟 .....	152
14.1.3.2 计数模式 .....	154
14.1.3.3 TMR输入部分 .....	156
14.1.3.4 TMR输出部分 .....	157
14.1.3.5 定时器同步 .....	159
14.1.3.6 调试模式 .....	161
14.1.4 TMRx寄存器描述 .....	162
14.1.4.1 控制寄存器1 (TMRx_CTRL1) .....	162
14.1.4.2 控制寄存器2 (TMRx_CTRL2) .....	163
14.1.4.3 次定时器控制寄存器 (TMRx_STCTRL) .....	163
14.1.4.4 DMA/中断使能寄存器 (TMRx_IDEN) .....	164
14.1.4.5 中断状态寄存器 (TMRxISTS) .....	165
14.1.4.6 软件事件寄存器 (TMRx_SWEVT) .....	166
14.1.4.7 通道模式寄存器1 (TMRx_CM1) .....	166
14.1.4.8 通道模式寄存器2 (TMRx_CM2) .....	168
14.1.4.9 通道控制寄存器 (TMRx_CCTRL) .....	169
14.1.4.10 计数值 (TMRx_CVAL) .....	170
14.1.4.11 分频系数 (TMRx_DIV) .....	170
14.1.4.12 周期寄存器 (TMRx_PR) .....	170
14.1.4.13 通道1数据寄存器 (TMRx_C1DT) .....	170
14.1.4.14 通道2数据寄存器 (TMRx_C2DT) .....	172
14.1.4.15 通道3数据寄存器 (TMRx_C3DT) .....	172
14.1.4.16 通道4数据寄存器 (TMRx_C4DT) .....	172
14.1.4.17 DMA控制寄存器 (TMRx_DMACTRL) .....	172
14.1.4.18 DMA数据寄存器 (TMRx_DMADT) .....	173
14.2 通用定时器 (TMR9到TMR11) .....	174
14.2.1 TMRx简介 .....	174
14.2.2 TMRx主要特性 .....	174
14.2.2.1 TMR9主要特性 .....	174
14.2.2.2 TMR10、TMR11主要特性 .....	174
14.2.3 TMRx功能描述 .....	175
14.2.3.1 计数时钟 .....	175
14.2.3.2 计数模式 .....	176
14.2.3.3 TMR输入部分 .....	177
14.2.3.4 TMR输出部分 .....	178
14.2.3.5 TMR同步 .....	179
14.2.3.6 调试模式 .....	180
14.2.4 TMR9寄存器描述 .....	181
14.2.4.1 控制寄存器 1 (TMR9_CTRL1) .....	181

14.2.4.2 次定时器控制寄存器 (TMR9_STCTRL) .....	181
14.2.4.3 DMA/中断使能寄存器 (TMR9_IDEN) .....	182
14.2.4.4 中断状态寄存器 (TMR9ISTS) .....	182
14.2.4.5 软件事件寄存器 (TMR9_SWEVT) .....	183
14.2.4.6 通道模式寄存器1 (TMR9_CM1) .....	183
14.2.4.7 通道控制寄存器 (TMR9_CCTRL) .....	185
14.2.4.8 计数器 (TMR9_CVAL) .....	186
14.2.4.9 预分频器 (TMR9_DIV) .....	186
14.2.4.10 周期寄存器 (TMR9_PR) .....	186
14.2.4.11 通道1数据寄存器 (TMR9_C1DT) .....	186
14.2.4.12 通道2数据寄存器 (TMR9_C2DT) .....	186
14.2.5 TMR10、TMR11寄存器描述 .....	186
14.2.5.1 控制寄存器1 (TMRx_CTRL1) .....	187
14.2.5.2 DMA/中断使能寄存器 (TMRx_IDEN) .....	187
14.2.5.3 中断状态寄存器 (TMRxISTS) .....	187
14.2.5.4 软件事件寄存器 (TMRx_SWEVT) .....	188
14.2.5.5 通道模式寄存器1 (TMRx_CM1) .....	188
14.2.5.6 通道控制寄存器 (TMRx_CCTRL) .....	190
14.2.5.7 计数值 (TMRx_CVAL) .....	190
14.2.5.8 预分频器 (TMRx_DIV) .....	190
14.2.5.9 周期寄存器 (TMRx_PR) .....	190
14.2.5.10 通道1数据寄存器 (TMRx_C1DT) .....	190
14.3 高级控制定时器 (TMR1) .....	192
14.3.1 TMR1简介 .....	192
14.3.2 TMR1主要特性 .....	192
14.3.3 TMR1功能描述 .....	192
14.3.3.1 计数时钟 .....	192
14.3.3.2 计数模式 .....	194
14.3.3.3 TMR输入部分 .....	196
14.3.3.4 TMR输出部分 .....	197
14.3.3.5 TMR刹车功能 .....	200
14.3.3.6 TMR同步 .....	201
14.3.3.7 调试模式 .....	202
14.3.4 TMR1寄存器描述 .....	202
14.3.4.1 TMR1控制寄存器1 (TMR1_CTRL1) .....	203
14.3.4.2 TMR1控制寄存器2 (TMR1_CTRL2) .....	203
14.3.4.3 TMR1次定时器控制寄存器 (TMR1_STCTRL) .....	204
14.3.4.4 TMR1 DMA/中断使能寄存器 (TMR1_IDEN) .....	205
14.3.4.5 TMR1中断状态寄存器 (TMR1ISTS) .....	206
14.3.4.6 TMR1软件事件寄存器 (TMR1_SWEVT) .....	207
14.3.4.7 TMR1通道模式寄存器1 (TMR1_CM1) .....	208
14.3.4.8 TMR1通道模式寄存器2 (TMR1_CM2) .....	210

14.3.4.9 TMR1通道控制寄存器 (TMR1_CCTRL) .....	211
14.3.4.10 TMR1计数值 (TMR1_CVAL) .....	212
14.3.4.11 TMR1预分频器 (TMR1_DIV) .....	212
14.3.4.12 TMR1周期寄存器 (TMR1_PR) .....	213
14.3.4.13 TMR1重复周期寄存器 (TMR1_RPR) .....	213
14.3.4.14 TMR1通道1数据寄存器 (TMR1_C1DT) .....	213
14.3.4.15 TMR1通道2数据寄存器 (TMR1_C2DT) .....	213
14.3.4.16 TMR1通道3数据寄存器 (TMR1_C3DT) .....	213
14.3.4.17 TMR1通道4数据寄存器 (TMR1_C4DT) .....	213
14.3.4.18 TMR1刹车寄存器 (TMR1_BRK) .....	213
14.3.4.19 TMR1 DMA控制寄存器 (TMR1_DMACTRL) .....	214
14.3.4.20 TMR1 DMA数据寄存器 (TMR1_DMADT) .....	215
<b>15 窗口看门狗 (WWDT) .....</b>	<b>216</b>
15.1 WWDT简介 .....	216
15.2 WWDT主要特性 .....	216
15.3 WWDT功能描述 .....	216
15.4 调试模式 .....	217
15.5 WWDT寄存器 .....	217
15.5.1 控制寄存器 (WWDT_CTRL) .....	217
15.5.2 配置寄存器 (WWDT_CFG) .....	217
15.5.3 状态寄存器 (WWDT_STS) .....	218
<b>16 看门狗 (WDT) .....</b>	<b>219</b>
16.1 WDT简介 .....	219
16.2 WDT主要特性 .....	219
16.3 WDT功能描述 .....	219
16.4 调试模式 .....	220
16.5 WDT寄存器 .....	220
16.5.1 命令寄存器 (WDT_CMD) .....	220
16.5.2 预分频寄存器 (WDT_DIV) .....	220
16.5.3 重装载寄存器 (WDT_RLD) .....	221
16.5.4 状态寄存器 (WDT_STS) .....	221
<b>17 实时时钟 (ERTC) .....</b>	<b>222</b>
17.1 ERTC简介 .....	222
17.2 ERTC主要特性 .....	222
17.3 ERTC功能说明 .....	222
17.3.1 ERTC时钟 .....	222
17.3.2 ERTC初始化 .....	223
17.3.3 周期性自动唤醒 .....	224
17.3.4 ERTC校准 .....	225
17.3.5 参考时钟检测 .....	225
17.3.6 时间戳 .....	225
17.3.7 入侵检测 .....	226

17.3.8 复用功能输出 .....	226
17.3.9 ERTC唤醒 .....	226
17.4 ERTC寄存器 .....	227
17.4.1 ERTC时间寄存器(ERTC_TIME) .....	228
17.4.2 ERTC日期寄存器(ERTC_DATE) .....	228
17.4.3 ERTC控制寄存器(ERTC_CTRL) .....	228
17.4.4 ERTC初始化和状态寄存器(ERTC_STS) .....	230
17.4.5 ERTC预分频器寄存器(ERTC_DIV) .....	231
17.4.6 ERTC唤醒定时器寄存器(ERTC_WAT) .....	231
17.4.7 ERTC粗校准寄存器(ERTC_CCAL) .....	231
17.4.8 ERTC闹钟A寄存器(ERTC_ALA) .....	231
17.4.9 ERTC闹钟B寄存器(ERTC_ALB) .....	232
17.4.10 ERTC写保护寄存器(ERTC_WP) .....	232
17.4.11 ERTC亚秒寄存器(ERTC_SBS) .....	233
17.4.12 ERTC时间微调寄存器(ERTC_TADJ) .....	233
17.4.13 ERTC时间戳时间寄存器(ERTC_TSTM) .....	233
17.4.14 ERTC时间戳日期寄存器(ERTC_TSDT) .....	233
17.4.15 ERTC时间戳亚秒寄存器(ERTC_TSSBS) .....	233
17.4.16 ERTC精密校准寄存器(ERTC_SCAL) .....	234
17.4.17 ERTC入侵配置寄存器(ERTC_TAMP) .....	234
17.4.18 ERTC闹钟A亚秒寄存器(ERTC_ALASBS) .....	235
17.4.19 ERTC闹钟B亚秒寄存器(ERTC_ALBSBS) .....	235
17.4.20 ERTC电池供电数据寄存器(ERTC_BPRx) .....	235
18 模拟/数字转换 (ADC) .....	236
18.1 ADC简介 .....	236
18.2 ADC主要特征 .....	236
18.3 ADC架构 .....	237
18.4 ADC功能介绍 .....	237
18.4.1 通道管理 .....	237
18.4.1.1 内部温度传感器 .....	238
18.4.1.2 内部参考电压 .....	238
18.4.2 ADC操作流程 .....	238
18.4.2.1 上电与校准 .....	238
18.4.2.2 触发 .....	239
18.4.2.3 采样与转换时序 .....	240
18.4.3 转换顺序管理 .....	240
18.4.3.1 序列模式 .....	240
18.4.3.2 抢占自动转换模式 .....	240
18.4.3.3 反复模式 .....	240
18.4.3.4 分割模式 .....	241
18.4.4 数据管理 .....	241
18.4.4.1 数据内容处理 .....	241

18.4.4.2	数据获取	242
18.4.5	电压监测	242
18.4.6	状态标志与中断	242
18.5	ADC寄存器	242
18.5.1	ADC状态寄存器 (ADC_STS)	243
18.5.2	ADC控制寄存器1 (ADC_CTRL1)	243
18.5.3	ADC控制寄存器2 (ADC_CTRL2)	245
18.5.4	ADC采样时间寄存器1 (ADC_SPT1)	246
18.5.5	ADC采样时间寄存器2 (ADC_SPT2)	248
18.5.6	ADC抢占通道数据偏移寄存器x (ADC_PCDTOx) (x=1..4)	250
18.5.7	ADC电压监测高边界寄存器 (ADC_VMHB)	250
18.5.8	ADC电压监测低边界寄存器 (ADC_VMLB)	251
18.5.9	ADC普通序列寄存器1 (ADC_OSQ1)	251
18.5.10	ADC普通序列寄存器2 (ADC_OSQ2)	251
18.5.11	ADC普通序列寄存器3 (ADC_OSQ3)	252
18.5.12	ADC抢占序列寄存器 (ADC_PSQ)	252
18.5.13	ADC抢占数据寄存器x (ADC_PDTx) (x= 1..4)	253
18.5.14	ADC普通数据寄存器 (ADC_ODT)	253
19	CAN总线控制器	254
19.1	简介	254
19.2	主要特性	254
19.3	波特率设置	254
19.4	中断管理	257
19.5	设计提示	257
19.6	功能描述	258
19.6.1	整体功能描述	258
19.6.2	工作模式	258
19.6.3	测试方法	259
19.6.4	报文过滤	259
19.6.5	报文发送	261
19.6.6	报文接收	262
19.6.7	出错管理	263
19.7	CAN寄存器	263
19.7.1	CAN控制和状态寄存器	265
19.7.1.1	CAN主控制寄存器 (CAN_MCTRL)	265
19.7.1.2	CAN主状态寄存器 (CAN_MSTS)	266
19.7.1.3	CAN发送状态寄存器 (CAN_TSTS)	267
19.7.1.4	CAN接收FIFO 0寄存器 (CAN_RF0)	270
19.7.1.5	CAN接收FIFO 1寄存器 (CAN_RF1)	270
19.7.1.6	CAN中断使能寄存器 (CAN_INTEN)	271
19.7.1.7	CAN错误状态寄存器 (CAN_ESTS)	272
19.7.1.8	CAN位时序寄存器 (CAN_BTMG)	273

19.7.2 CAN邮箱寄存器 .....	273
19.7.2.1 发送邮箱标识符寄存器 (CAN_TMIx) (x=0..2) .....	274
19.7.2.2 发送邮箱数据长度和时间戳寄存器 (CAN_TMCx) (x=0..2) ...	274
19.7.2.3 发送邮箱低字节数据寄存器 (CAN_TMDTLx) (x=0..2) .....	275
19.7.2.4 发送邮箱高字节数据寄存器 (CAN_TMDTHx) (x=0..2) .....	275
19.7.2.5 接收FIFO邮箱标识符寄存器 (CAN_RFIx) (x=0..1) .....	275
19.7.2.6 接收FIFO邮箱数据长度和时间戳寄存器 (CAN RFCx) (x=0..1)	275
19.7.2.7 接收FIFO邮箱低字节数据寄存器 (CAN_RFDTLx) (x=0..1) ...	275
19.7.2.8 接收FIFO邮箱高字节数据寄存器 (CAN_RFDTThx) (x=0..1) ..	276
19.7.3 CAN过滤器寄存器 .....	276
19.7.3.1 CAN过滤器控制寄存器 (CAN_FCTRL) .....	276
19.7.3.2 CAN过滤器模式配置寄存器 (CAN_FMCFG) .....	276
19.7.3.3 CAN过滤器位宽配置寄存器 (CAN_FBWCFG) .....	276
19.7.3.4 CAN过滤器FIFO关联寄存器 (CAN_FRF) .....	277
19.7.3.5 CAN过滤器激活控制寄存器 (CAN_FACFG) .....	277
19.7.3.6 CAN过滤器组i的过滤位寄存器x (CAN_FiFBx) (其中i=0..13; x=1..2)	
	277
<b>20 USB OTG全速(OTGFS) .....</b>	<b>278</b>
20.1 OTGFS系统结构框图 .....	278
20.2 OTGFS 功能概述 .....	278
20.3 OTGFS时钟与管脚配置 .....	279
20.3.1 OTGFS时钟配置 .....	279
20.3.2 OTGFS管脚配置 .....	279
20.4 OTGFS中断 .....	279
20.5 OTGFS 功能操作 .....	280
20.5.1 OTGFS初始化 .....	280
20.5.2 OTGFS FIFO配置 .....	281
20.5.2.1 设备模式 .....	281
20.5.2.2 主机模式 .....	282
20.5.2.3 刷新控制器发送FIFO .....	283
20.5.3 OTGFS主机模式 .....	283
20.5.3.1 主机初始化 .....	283
20.5.3.2 OTGFS 通道初始化 .....	283
20.5.3.3 中止通道 .....	284
20.5.3.4 队列深度 .....	284
20.5.3.5 特殊情况处理 .....	286
20.5.3.6 主机HFIR功能 .....	286
20.5.3.7 初始化批量IN传输/控制IN传输 .....	288
20.5.3.8 初始化批量和控制OUT/SETUP传输 .....	289
20.5.3.9 初始化中断IN传输 .....	291
20.5.3.10 初始化中断OUT传输 .....	292
20.5.3.11 初始化同步IN传输 .....	294

20.5.3.12 初始化同步OUT传输 .....	295
20.5.4 OTGFS设备模式 .....	296
20.5.4.1 设备初始化 .....	296
20.5.4.2 USB复位时初始化 .....	297
20.5.4.3 枚举完成时初始化 .....	297
20.5.4.4 SetAddress指令时初始化 .....	297
20.5.4.5 SetConfiguration/SetInterface时初始化 .....	297
20.5.4.6 激活端点 .....	298
20.5.4.7 USB 端点失效操作 .....	298
20.5.4.8 控制写传输(SETUP/Data OUT/Status IN) .....	298
20.5.4.9 控制读传输(SETUP/Data IN/Status OUT) .....	299
20.5.4.10 两个阶段的控制传输(SETUP/Status IN) .....	299
20.5.4.11 读取FIFO包 .....	299
20.5.4.12 OUT数据传输 .....	300
20.5.4.13 IN数据传输 .....	302
20.5.4.14 非周期性（批量和控制）IN数据传输 .....	303
20.5.4.15 非同步OUT数据传输 .....	303
20.5.4.16 同步OUT数据传输 .....	305
20.5.4.17 使能同步端点 .....	306
20.5.4.18 未完成同步OUT数据传输 .....	307
20.5.4.19 未完成同步IN数据传输 .....	308
20.5.4.20 周期性IN（中断和同步）数据传输 .....	309
20.6 OTGFS控制和状态寄存器 .....	310
20.6.1 CSR寄存器映像 .....	310
20.6.2 OTGFS寄存器地址映象 .....	311
20.6.3 OTGFS全局寄存器 .....	314
20.6.3.1 OTGFS状态控制器(OTGFS_GOTGCTL) .....	314
20.6.3.2 OTGFS OTG中断状态控制器(OTGFS_GOTGINT) .....	314
20.6.3.3 OTGFS AHB配置寄存器(OTGFS_GAHBCFG) .....	314
20.6.3.4 OTGFS_USB配置寄存器(OTGFS_GUSBCFG) .....	315
20.6.3.5 OTGFS复位寄存器(OTGFS_GRSTCTL) .....	316
20.6.3.6 OTGFS中断寄存器(OTGFS_GINTSTS) .....	317
20.6.3.7 OTGFS中断屏蔽寄存器(OTGFS_GINTMSK) .....	320
20.6.3.8 OTGFS接收状态调试读/OTG状态读和POP寄存器(OTGFS_GRXSTSR / OTGFS_GRXSTSP) .....	322
20.6.3.9 OTGFS接收FIFO长度寄存器(OTGFS_GRXFSIZ) .....	323
20.6.3.10 OTGFS非周期性TX FIFO长度寄存器(OTGFS_GNPTXFSIZ)/端点0 TX FIFO长度寄存器(OTGFS_DIEPTXF0) .....	323
20.6.3.11 OTGFS非周期性TX FIFO/请求队列状态寄存器(OTGFS_GNPTXSTS) .....	323
20.6.3.12 OTGFS通用控制器配置寄存器(OTGFS_GCCFG) .....	324
20.6.3.13 OTGFS控制器ID寄存器(OTGFS_GUID) .....	324
20.6.3.14 OTGFS主机周期性发送FIFO长度寄存器(OTGFS_HPTXFSIZ) .....	324

20.6.3.15 OTGFS设备IN端点发送FIFO长度寄存器(OTGFS_DIEPTXF <sub>n</sub> )(其中n是FIFO的编号, x=1...3) .....	325
20.6.4 主机模式下的寄存器.....	325
20.6.4.1 OTGFS主机模式配置寄存器(OTGFS_HCFG) .....	325
20.6.4.2 OTGFS主机帧间隔寄存器(OTGFS_HFIR) .....	325
20.6.4.3 OTGFS主机帧号/帧时间剩余寄存器(OTGFS_HFNUM) .....	326
20.6.4.4 OTGFS主机周期性发送FIFO/请求队列寄存器(OTGFS_HPTXSTS) .....	326
20.6.4.5 OTGFS主机所有通道中断寄存器(OTGFS_HAINT) .....	326
20.6.4.6 OTGFS主机所有通道中断屏蔽寄存器(OTGFS_HAINTMSK) .....	327
20.6.4.7 OTGFS主机端口控制和状态寄存器(OTGFS_HPRT).....	327
20.6.4.8 OTGFS主机通道x特性寄存器(OTGFS_HCCHAR <sub>x</sub> )(此处x代码通道号, x = 0...8) .....	328
20.6.4.9 OTGFS主机通道x中断寄存器(OTGFS_HCINT <sub>x</sub> )(其中x代表通道号, x=0...8) .....	329
20.6.4.10 OTGFS主机通道x中断屏蔽寄存器(OTGFS_HCINTMSK <sub>x</sub> )(其中x为通道号, x=0...8) .....	330
20.6.4.11 OTGFS主机通道x传输长度寄存器(OTGFS_HCTSIZ <sub>x</sub> )(其中x为通道号, x=0...8) .....	330
20.6.5 设备模式下的寄存器.....	331
20.6.5.1 OTGFS设备配置寄存器(OTGFS_DCFG) .....	331
20.6.5.2 OTGFS设备控制寄存器(OTGFS_DCTL).....	331
20.6.5.3 OTGFS设备状态寄存器(OTGFS_DSTS) .....	332
20.6.5.4 OTGFS设备OTGFSIN端点通用中断屏蔽寄存器(OTGFS_DIEPMSK) .....	333
20.6.5.5 OTGFS设备OUT端点通用中断屏蔽寄存器(OTGFS_DOEPM <sub>SK</sub> ) .....	334
20.6.5.6 OTGFS设备所有端点中断寄存器(OTGFS_DAINT) .....	334
20.6.5.7 OTGFS所有端点中断屏蔽寄存器(OTGFS_DAINTMSK) .....	335
20.6.5.8 OTGFS设备IN端点FIFO空中断屏蔽寄存器(OTGFS_DIEPEMPMSK) .....	335
20.6.5.9 OTGFS设备控制IN端点0控制寄存器(OTGFS_DIEPCTL0) .....	335
20.6.5.10 OTGFS设备IN端点x控制寄存器(OTGFS_DIEPCTL <sub>x</sub> )(其中x为端点号, x=1...3) .....	336
20.6.5.11 OTGFS设备控制OUT端点0控制寄存器(OTGFS_DOEPCTL0) .....	338
20.6.5.12 OTGFS设备OUT端点x控制寄存器(OTGFS_DOEPCTL <sub>x</sub> )(其中x为端点号, x=1...3) .....	339
20.6.5.13 OTGFS设备IN端点x中断寄存器(OTGFS_DIEPINT <sub>x</sub> )(其中x为端点号, x=0...3) .....	340
20.6.5.14 OTGFS设备OUT端点x中断寄存器(OTGFS_DOEPINT <sub>x</sub> )(其中x为端点号, x=0...3) .....	341
20.6.5.15 OTGFS设备IN端点0传输长度寄存器(OTGFS_DIEPTSIZ0) .....	341
20.6.5.16 OTGFS设备OUT端点0传输长度寄存器(OTGFS_DOEPTSIZ0) .....	342
20.6.5.17 OTGFS设备IN端点x传输长度寄存器(OTGFS_DIEPTSIZ <sub>x</sub> )( 其中 x 为端点号, x=1...3) .....	342
20.6.5.18 OTGFS设备IN端点传输FIFO状态寄存器(OTGFS_DTXFSTS <sub>x</sub> )(其中x为端点号, x=0...3) .....	343

20.6.5.19 OTGFS设备OUT端点x传输长度寄存器(OTGFS_DOEPTSIZx)( 其中x 为端点号, x=1...3).....	343
20.6.6 供电和时钟控制寄存器 .....	343
20.6.6.1 OTGFS电源和时钟门控寄存器(OTGFS_PCGCCTL).....	343
<b>21 比较器(CMP).....</b>	<b>345</b>
21.1 简介 .....	345
21.2 主要特性 .....	345
21.3 中断管理 .....	345
21.4 设计提示 .....	346
21.5 功能描述 .....	346
21.5.1 模拟比较器 .....	346
21.6 CMP寄存器 .....	346
21.6.1 比较器控制和状态寄存器1 ( CMP_CTRLSTS1 ) .....	347
21.6.2 比较器控制和状态寄存器2(CMP_CTRLSTS2) .....	349
<b>22 调试DEBUG.....</b>	<b>350</b>
22.1 简介 .....	350
22.2 调试与跟踪功能 .....	350
22.3 I/O控制 .....	350
22.4 DEBUG寄存器 .....	351
22.4.1 DEBUG设备ID ( DEBUG_IDCODE ) .....	351
22.4.2 DEBUG控制寄存器 ( DEBUG_CTRL ) .....	351
<b>23 版本历史 .....</b>	<b>353</b>

## 图目录

图 1-1 AT32WB415 系列微控制器系统架构 .....	24
图 1-2 Cortex <sup>TM</sup> -M4 内部框图 .....	25
图 1-3 位带区与位带别名区的膨胀关系图 A .....	25
图 1-4 位带区与位带别名区的膨胀关系图 B .....	26
图 1-5 复位流程 .....	30
图 1-6 MSP 及 PC 初始化的一个范例 .....	31
图 2-1 AT32WB415 地址映射 .....	33
图 3-1 各电源域框图 .....	36
图 3-2 上电/下电复位波形图 .....	37
图 3-3 PVM 的阈值与输出 .....	37
图 4-1 AT32WB415 时钟结构图 .....	41
图 4-2 系统复位电路 .....	43
图 5-1 主存储器扇区擦除流程图 .....	60
图 5-2 主存储器整片擦除流程图 .....	61
图 5-3 主存储器编程流程图 .....	62
图 5-4 系统数据区擦除图 .....	64
图 5-5 系统数据区编程图 .....	65
图 6-1 GPIO 基本结构 .....	76
图 7-1 IOMUX 基本结构 .....	80
图 8-1 外部中断/事件控制器框图 .....	91
图 9-1 DMA 框图 .....	94
图 9-2 请求/应答对后重新仲裁 .....	95
图 9-3 PWIDHT: byte, MWIDHT: half-word .....	96
图 9-4 PWIDHT: half-word, MWIDHT: word .....	96
图 9-5 PWIDHT: word, MWIDHT: byte .....	96
图 11-1 I <sup>2</sup> C 总线协议 .....	108
图 11-2 I <sup>2</sup> C 的功能框图 .....	109
图 11-3 从发送器的传送序列图 .....	111
图 11-4 从接收器的传送序列图 .....	112
图 11-5 主发送器传送序列图 .....	113
图 11-6 主接收器传送序列图 .....	114
图 11-7 N>2 主接收器传送序列图 .....	115
图 11-8 N=2 主接收器传送序列图 .....	116
图 11-9 N=1 主接收器传送序列图 .....	117
图 12-1 USART 框图 .....	126
图 12-2 USART 中断映像图 .....	132
图 13-1 SPI 框图 .....	138
图 13-2 SPI 双线单向全双工连接示意图 .....	139
图 13-3 SPI 作主机单线单向只收连接示意图 .....	139
图 13-4 SPI 作从机单线单向只收连接示意图 .....	140
图 13-5 SPI 作单线双向半双工连接示意图 .....	140
图 13-6 主机全双工通信 .....	144
图 13-7 从机全双工通信 .....	144
图 13-8 主机半双工发送通信 .....	144
图 13-9 从机半双工接收通信 .....	145
图 13-10 从机半双工发送通信 .....	145
图 13-11 主机半双工接收通信 .....	145
图 13-12 SPI 中断 .....	146
图 14-1 通用定时器框图 .....	152
图 14-2 使用 CK_INT 且分频系数为 1 .....	152
图 14-3 外部时钟模式 A 框图 .....	152

图 14-4 使用外部时钟模式 A 计数.....	153
图 14-5 外部时钟模式 B 框图.....	153
图 14-6 使用外部时钟模式 B 计数.....	153
图 14-7 当预分频器的参数从 1 变到 4 时, 计数器的时序图.....	154
图 14-8 PRBEN=0 时的溢出事件.....	154
图 14-9 PRBEN=1 时的溢出事件.....	154
图 14-10 计数器时序图, 内部时钟分频因子为 4.....	155
图 14-11 计数器时序图, 内部时钟分频因子为 1, TMRx_PR=0x32.....	155
图 14-12 编码模式计数实例 (编码器模式 C) .....	156
图 14-13 输入/输出通道 1 的主电路 .....	156
图 14-14 通道 1 输入部分 .....	156
图 14-15 捕获/比较通道的输出部分 (通道 1 至 4) .....	157
图 14-16 计数值与 C1DT 值匹配时翻转 C1ORAW .....	158
图 14-17 向上计数下 PWM 模式 A .....	158
图 14-18 中央双向对齐计数下 PWM 模式 A .....	158
图 14-19 单周期模式 .....	159
图 14-20 EXT 清除 CxORAW(PWM 模式 A).....	159
图 14-21 复位模式例子 .....	159
图 14-22 挂起模式下例子 .....	160
图 14-23 触发器模式例子 .....	160
图 14-24 主/次定时器连接框图 .....	160
图 14-25 主定时器启动次定时器例子 .....	161
图 14-26 外部触发同时启动主、次定时器 .....	161
图 14-27 通用定时器 TMR9 框图 .....	174
图 14-28 通用定时器 TMR10/11 框图 .....	175
图 14-29 使用 CK_INT 且分频系数为 1 .....	175
图 14-30 外部时钟模式 A 框图 .....	175
图 14-31 使用外部时钟模式 A 计数 .....	176
图 14-32 当预分频器的参数从 1 变到 4 时, 计数器的时序图 .....	176
图 14-33 PRBEN=0 时的溢出事件 .....	177
图 14-34 PRBEN=1 时的溢出事件 .....	177
图 14-35 输入/输出通道 1 的主电路 .....	177
图 14-36 通道 1 输入部分 .....	177
图 14-37 捕获/比较通道的输出部分 (通道 1) .....	178
图 14-38 计数值与 C1DT 值匹配时翻转 C1ORAW .....	179
图 14-39 向上计数下 PWM 模式 A .....	179
图 14-40 单周期模式 .....	179
图 14-41 复位模式例子 .....	180
图 14-42 挂起模式下例子 .....	180
图 14-43 触发器模式例子 .....	180
图 14-44 高级控制定时器框图 .....	192
图 14-45 使用 CK_INT 且分频系数为 1 .....	193
图 14-46 外部时钟模式 A 框图 .....	193
图 14-47 使用外部时钟模式 A 计数 .....	193
图 14-48 外部时钟模式 B 框图 .....	193
图 14-49 使用外部时钟模式 B 计数 .....	194
图 14-50 当预分频器的参数从 1 变到 4 时, 计数器的时序图 .....	194
图 14-51 PRBEN=0 时的溢出事件 .....	195
图 14-52 PRBEN=1 时的溢出事件 .....	195
图 14-53 计数器时序图, 内部时钟分频因子为 4 .....	195
图 14-54 计数器时序图, 内部时钟分频因子为 1, TMRx_PR=0x32 .....	195
图 14-55 RPR=2 时的 OVIF.....	196

图 14-56 编码模式计数实例（编码器模式 C） .....	196
图 14-57 输入/输出通道 1 的主电路 .....	196
图 14-58 通道 1 输入部分 .....	197
图 14-59 通道 1 至 3 输出部分 .....	197
图 14-60 通道 4 输出部分 .....	197
图 14-61 计数值与 C1DT 值匹配时翻转 C1ORAW .....	198
图 14-62 向上计数下 PWM 模式 A .....	199
图 14-63 中央双向对齐计数下 PWM 模式 .....	199
图 14-64 单周期模式 .....	199
图 14-65 EXT 清除 CxORAW(PWM 模式 A) .....	200
图 14-66 带死区插入的互补输出 .....	200
图 14-67 TMR 刹车功能的例子 .....	201
图 14-68 复位模式例子 .....	201
图 14-69 挂起模式下例子 .....	202
图 14-70 触发器模式例子 .....	202
图 15-1 窗口看门狗框图 .....	216
图 15-2 窗口看门狗时序图 .....	217
图 16-1 看门狗框图 .....	219
图 17-1 ERTC 框图 .....	222
图 18-1 ADC1 框图 .....	237
图 18-2 ADC 基础操作流程 .....	238
图 18-3 ADC 上电与校准 .....	239
图 18-4 序列模式 .....	240
图 18-5 抢占自动转换模式 .....	240
图 18-6 反复模式 .....	241
图 18-7 分割模式 .....	241
图 18-8 数据内容处理 .....	242
图 19-1 位时序 .....	254
图 19-2 发送中断的产生 .....	256
图 19-3 发送中断的产生 .....	257
图 19-4 接收中断 0 的产生 .....	257
图 19-5 接收中断 1 的产生 .....	257
图 19-6 状态错误中断的产生 .....	257
图 19-7 CAN 框图 .....	258
图 19-8 32 位宽标识符掩码模式 .....	260
图 19-9 32 位宽标识符列表模式 .....	260
图 19-10 16 位宽标识符掩码模式 .....	260
图 19-11 16 位宽标识符列表模式 .....	260
图 19-12 发送邮箱状态转换 .....	262
图 19-13 接收 FIFO 状态 .....	263
图 19-14 发送和接收邮箱 .....	274
图 20-1 OTGFS 的系统结构框图 .....	278
图 20-2 OTGFS 中断结构示意图 .....	280
图 20-3 写入发送 FIFO 流程图 .....	285
图 20-4 读取接收 FIFO 流程图 .....	286
图 20-5 HFIRRLDCTRL 为 0x0 时的 HFIR 行为 .....	287
图 20-6 HFIRRLDCTRL 为 0x1 时的 HFIR 行为 .....	287
图 20-7 普通 Bulk/Control OUT/SETUP 和 Bulk/Control IN 传输例程示例图 .....	290
图 20-8 普通中断 OUT/IN 传输示例图 .....	293
图 20-9 普通同步 OUT/IN 传输示例图 .....	296
图 20-10 读取接收 FIFO .....	300
图 20-11 SETUP 数据包流程图 .....	302

图 20-12 BULK OUT 传输示例图 .....	305
图 20-13 CSR 存储器映像.....	311
图 21-1 比较器的框图.....	345

## 表目录

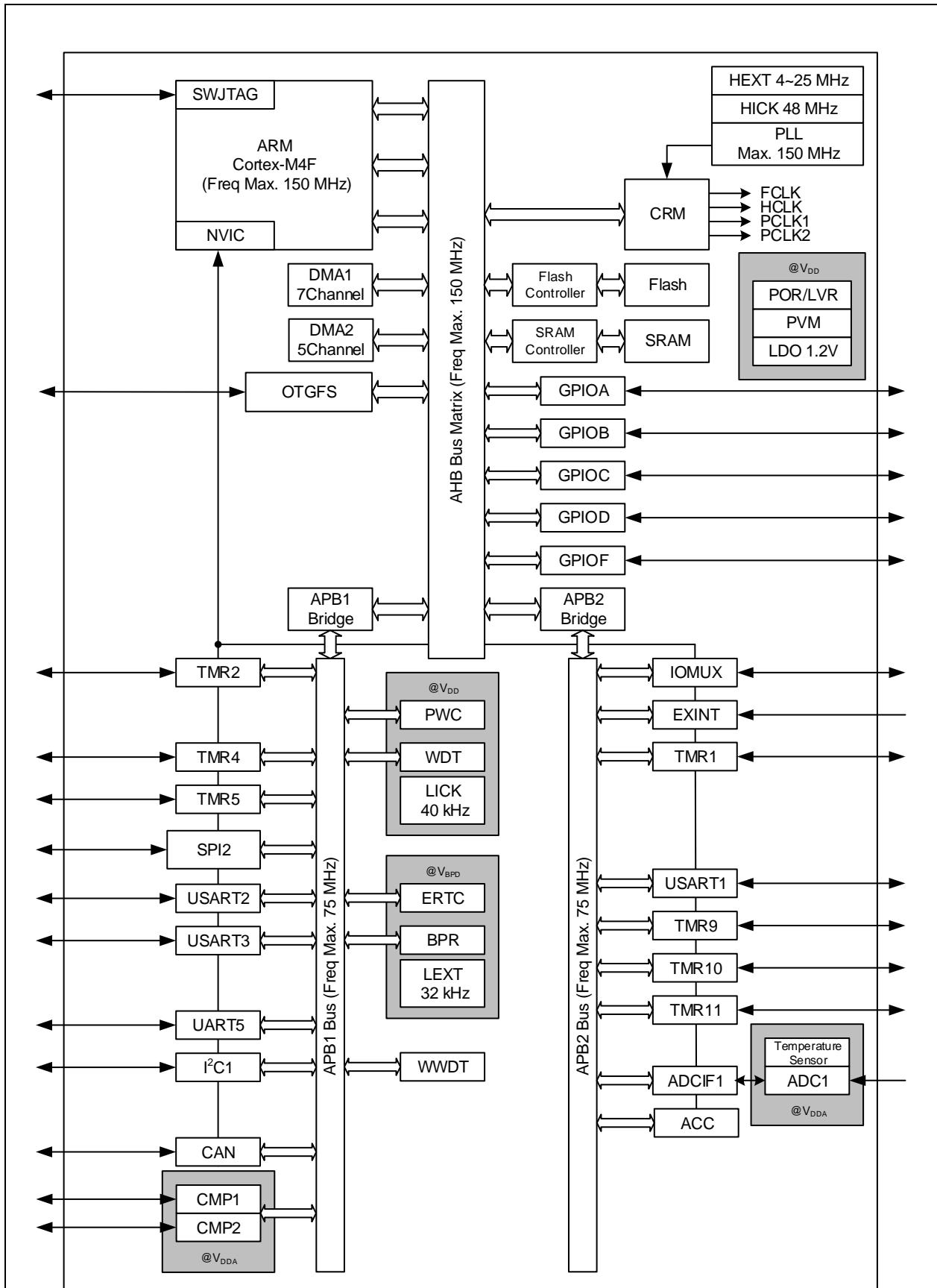
表 1-1 SRAM 区中的位带地址映射 .....	26
表 1-2 外设区中的位带地址映射 .....	27
表 1-3 AT32WB415 产品的向量表 .....	27
表 1-4 寄存器描述缩写说明 .....	31
表 1-5 器件特征信息相关寄存器地址和复位值 .....	32
表 2-1 256KB 闪存模块的组织 .....	33
表 2-2 各外设起始地址 .....	34
表 3-1 PWC 寄存器映像和复位值 .....	39
表 4-1 CRM 寄存器映像和复位值 .....	43
表 5-1 闪存存储结构 (256K) .....	58
表 5-2 用户系统数据说明 .....	58
表 5-3 闪存访问权限 .....	66
表 5-4 闪存接口寄存器映像和复位值 .....	68
表 6-1 GPIO 寄存器映像和复位值 .....	77
表 7-1 复用功能输入配置 .....	81
表 7-2 复用功能输出配置 .....	81
表 7-3 硬件抢占功能 .....	81
表 7-4 调试端口映射 .....	82
表 7-5 IOMUX 寄存器地址映像和复位值 .....	82
表 8-1 外部中断/事件控制器寄存器映像和复位值 .....	92
表 9-1 DMA 错误事件 .....	96
表 9-2 DMA 中断 .....	97
表 9-3 DMA1 各通道的外设请求 .....	97
表 9-4 DMA2 各通道的外设请求 .....	97
表 9-5 DMA 各通道的弹性请求 .....	97
表 9-6 DMA 寄存器的映像和复位值 .....	98
表 10-1 CRC 计算单元寄存器映像 .....	106
表 11-1 I <sup>2</sup> C 寄存器地址映像和复位值 .....	120
表 12-1 检测起始位和噪声的数据采样 .....	131
表 12-2 检测有效数据和噪声的数据采样 .....	131
表 12-3 USART 中断请求 .....	132
表 12-4 USART 寄存器映像和复位值 .....	133
表 13-1 SPI 寄存器映像和复位值 .....	146
表 14-1 TMR 功能对比 .....	151
表 14-2 TMRx 内部触发连接 .....	153
表 14-3 计数方向与编码器信号的关系 .....	155
表 14-4 TMRx – 寄存器映像和复位值 .....	162
表 14-5 标准 CxOUT 通道的输出控制位 .....	170
表 14-6 TMRx 内部触发连接 .....	176
表 14-7 TMR9 寄存器映像和复位值 .....	181
表 14-8 标准 CxOUT 通道的输出控制位 .....	186
表 14-9 TMR10、TMR11 寄存器映像和复位值 .....	186
表 14-10 标准 CxOUT 通道的输出控制位 .....	190
表 14-11 TMRx 内部触发连接 .....	194
表 14-12 计数方向与编码器信号的关系 .....	196
表 14-13 TMR1 寄存器映像和复位值 .....	202
表 14-14 带刹车功能的互补输出通道 CxOUT 和 CxCOUT 的控制位 .....	211
表 15-1 PCLK1 频率为 72MHz 时，最大和最小看门狗超时时间 .....	216
表 15-2 WWDT 寄存器映像和复位值 .....	217
表 16-1 看门狗超时时间 (LICK=40kHz) .....	220
表 16-2 WDT 寄存器映像和复位值 .....	220

表 17-1 ERTC 寄存器配置表.....	223
表 17-2 ERTC 唤醒低功耗模式 .....	227
表 17-3 中断控制位 .....	227
表 17-4 ERTC 寄存器映像和复位值.....	227
表 18-1 ADC 触发来源 .....	239
表 18-2 ADC 寄存器映像和复位值 .....	242
表 19-1 CAN 寄存器映像和复位值 .....	263
表 20-1 OTGFS 输入/输出引脚 .....	279
表 20-2 OTGFS 发送 FIFO SRAM 分配表 .....	281
表 20-3 OTGFS 内部寄存器存储空间分配表 .....	282
表 20-4 OTGFS 模块的寄存器映像及其复位值.....	311
表 20-5 软件断开的最短持续时间.....	332
表 21-1 CMP 寄存器映像和复位值.....	346
表 22-1 跟踪功能使能.....	350
表 22-2 跟踪功能模式.....	350
表 22-3 DEBUG 寄存器地址和复位值.....	351

# 1 系统构架

AT32WB415 系列微控制器内部集成了：32 位 ARM®Cortex™-M4 处理器，多个 16 位和 32 位的定时器，DMA 控制器，实时时钟 ERTC，SPI 通信接口，I2C 通信接口，USART/UART 通信接口，CAN 总线控制器，USB2.0 OTG 全速接口，12 位 ADC 和 PVM 模块等外设。大量的外设和存储器。Cortex™-M4 处理器支持增强的高效 DSP 指令集，包含扩展的单周期 16/32 位乘法累加器（MAC）、双 16 位 MAC 指令、优化的 8/16 位 SIMD 运算及饱和运算指令。系统详细架构见下图。

图 1-1 AT32WB415系列微控制器系统架构



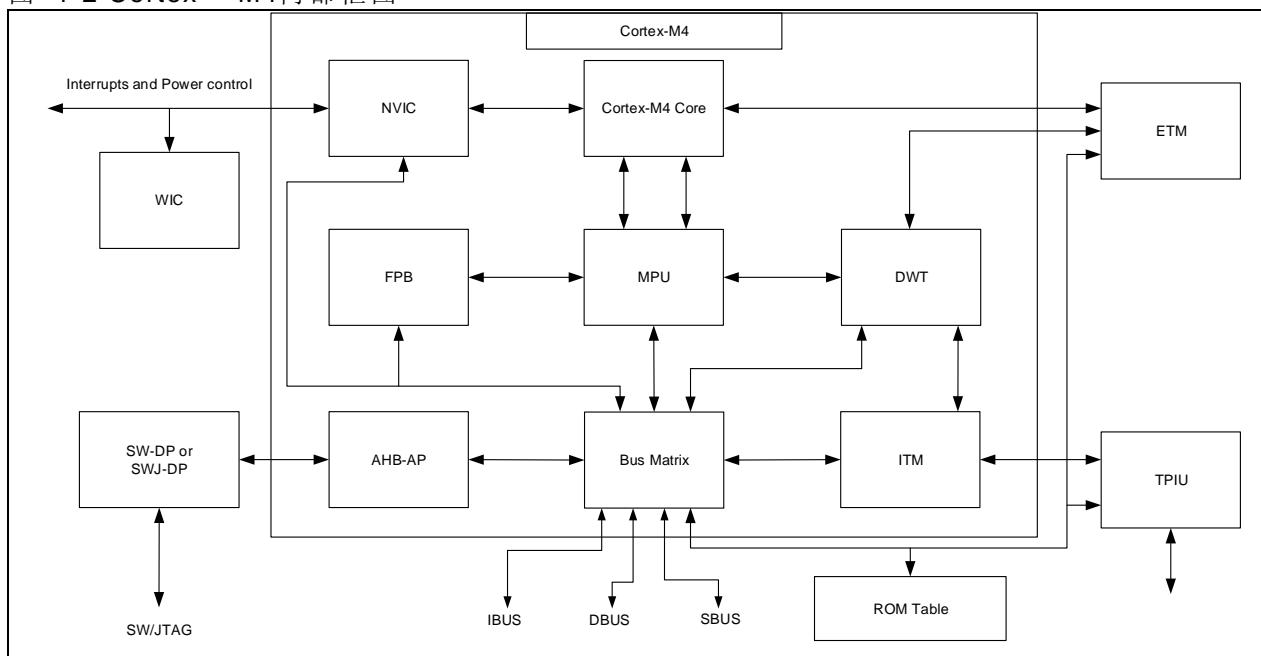
## 1.1 系统概述

### 1.1.1 ARM Cortex™-M4处理器

Cortex™-M4 处理器是一款低功耗处理器，具有低门数，低中断延迟和低成本调试的特点。支持包括 DSP 指令集，特别适合用于深度嵌入式应用程序需要快速中断响应功能。Cortex™-M4 处理器是基于 ARMv7-M 架构，既支持 Thumb 指令集也支持 DSP 指令集。

下图为 Cortex™-M4 处理器的内部框图，请参阅《ARM® Cortex-M4 技术参考手册》了解关于 Cortex™-M4 更详尽信息。

图 1-2 Cortex™-M4 内部框图



### 1.1.2 位带

利用位带操作，可以使用普通的加载/存储操作来对单一比特进行读写访问。在 Cortex™-M4 中提供了两个位带区：SRAM 最低 1M 字节空间和外设区间的最低 1M 字节空间。这两个区中的地址除了可以像普通存储器一样访问外，还可以通过它们各自的位带别名区来快捷访问这两个区中任意地址的任意比特位，位带别名区将位带区每个比特膨胀成一个 32 位的字。当你访问位带别名区的一个地址时，等同于直接访问位带区的一个比特位。

图 1-3 位带区与位带别名区的膨胀关系图 A

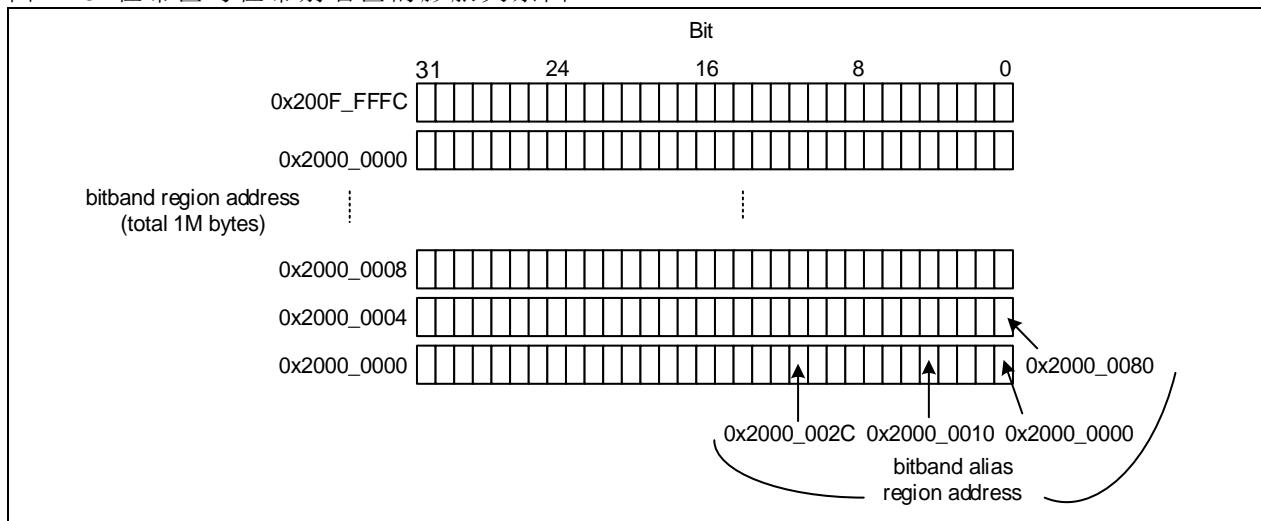
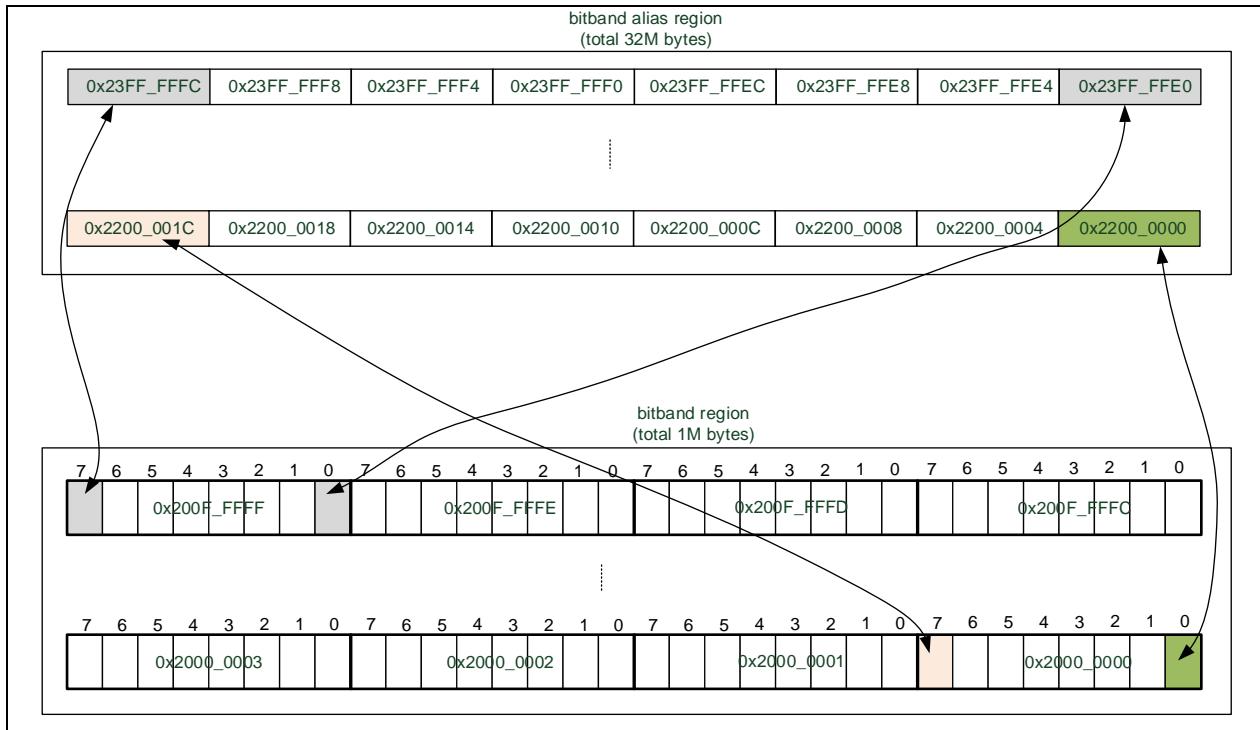


图 1-4 位带区与位带别名区的膨胀关系图B



**位带区：**支持位带操作的地址区

**位带别名区：**对别名区地址的访问最终作用到位带区的访问上

在位带区中，每个比特都映射到别名地址区的一个字（这是只有 **LSB** 有效的字）。当一个位带别名区地址被访问时，会先把该地址转换成位带区地址。对于读操作，读取位带区地址中的一个字，再把需要的位右移到 **LSB**，并把 **LSB** 返回。对于写操作，把需要写的位左移到对应的位序号处，然后执行一个比特级的“读-改-写”过程。

支持位带操作的两个内存区的地址范围为：

SRAM 区中的最低 1M 字节：0x2000\_0000~0x200F\_FFFF

外设区间最低 1M 字节：0x4000\_0000~0x400F\_FFFF

对于 SRAM 位带区的某个比特，如果所在字节地址为 A，位序号为 n(0<=n<=7)，则该比特在别名区的地址为：

$$\text{AliasAddr} = 0x2200_0000 + (A - 0x2000_0000) * 32 + n * 4$$

对于外设区间位带区的某个比特，如果所在字节地址为 A，位序号为 n(0<=n<=7)，则该比特在别名区的地址为：

$$\text{AliasAddr} = 0x4200_0000 + (A - 0x4000_0000) * 32 + n * 4$$

对于 SRAM 区中，位带区与位带别名区的映射如下表所示：

表 1-1 SRAM区中的位带地址映射

位带区	等效别名区地址
0x2000_0000.0	0x2200_0000.0
0x2000_0000.1	0x2200_0004.0
0x2000_0000.2	0x2200_0008.0
...	...
0x2000_0000.31	0x2200_007C.0
0x2000_0004.0	0x2200_0080.0
0x2000_0004.1	0x2200_0084.0
0x2000_0004.2	0x2200_0088.0

...	...
0x200F_FFFC.31	0x23FF_FFFC.0

对于外设区中，位带区与位带别名区的映射如下表所示：

表 1-2 外设区中的位带地址映射

位带区	等效别名区地址
0x4000_0000.0	0x4200_0000.0
0x4000_0000.1	0x4200_0004.0
0x4000_0000.2	0x4200_0008.0
...	...
0x4000_0000.31	0x4200_007C.0
0x4000_0004.0	0x4200_0080.0
0x4000_0004.1	0x4200_0084.0
0x4000_0004.2	0x4200_0088.0
...	...
0x400F_FFFC.31	0x43FF_FFFC.0

位带操作的优越性最容易想到的是通过 GPIO 的管脚来单独控制每盏 LED 的点亮与熄灭。另一方面，也对操作串行接口提供很大的方便。总之，位带操作对于硬件 I/O 密集型的底层程序最有用处。

位带操作还能简化跳转的判断。当跳转依据是某个位时，以前必须这样做：

- 读取整个寄存器
- 屏蔽不需要的位
- 比较并跳转

现在只需要：

- 从位带别名区读取该位的状态
- 比较并跳转

使代码更简洁，这只是位带操作优越性的初步体现，位带操作还有一个重要的好处是在多任务以及多任务环境中，将以前的读-改-写需要的三条指令，做成了一个硬件级别支持的原子操作，消除了以前读-改-写可能被中断，导致出现紊乱的情况。

### 1.1.3 中断和异常向量

下面两个表，分别列出了 AT32WB415 产品的向量表。

表 1-3 AT32WB415 产品的向量表

位置	优先级 类型	名称	说明	地址
-	-	-	保留	0x0000_0000
-3	固定	Reset	复位	0x0000_0004
-2	固定	NMI	CRM 时钟失效检测（CFD）联接到 NMI 向量	0x0000_0008
-1	固定	硬件失效（HardFault）	所有类型的失效	0x0000_000C
0	可设置	存储管理 (MemoryManage)	存储器管理	0x0000_0010
1	可设置	总线错误（BusFault）	预取指失败，存储器访问失败	0x0000_0014
2	可设置	错误应用（UsageFault）	未定义的指令或非法状态	0x0000_0018

-	-	-	保留	0x0000_001C ~0x0000_002B	
3	可设置	SVCall	通过 SWI 指令的系统服务调用	0x0000_002C	
4	可设置	调试监控 (DebugLENonitor)	调试监控器	0x0000_0030	
-	-	-	保留	0x0000_0034	
5	可设置	PendSV	可挂起的系统服务	0x0000_0038	
6	可设置	SysTick	系统嘀嗒定时器	0x0000_003C	
0	7	可设置	WWDT	窗口定时器中断	0x0000_0040
1	8	可设置	PVM	连到 EXINT 线 16 的电源电压检测 (PVD) 中断	0x0000_0044
2	9	可设置	TAMPER	连接到 EXINT 线 21 的侵入检测中断	0x0000_0048
3	10	可设置	ERTC	连接到 EXINT 线 22 的实时时钟 (ERTC) 全局中断	0x0000_004C
4	11	可设置	FLASH	闪存全局中断	0x0000_0050
5	12	可设置	CRM	复位和时钟控制 (CRM) 中断	0x0000_0054
6	13	可设置	EXINT0	EXINT 线 0 中断	0x0000_0058
7	14	可设置	EXINT1	EXINT 线 1 中断	0x0000_005C
8	15	可设置	EXINT2	EXINT 线 2 中断	0x0000_0060
9	16	可设置	EXINT3	EXINT 线 3 中断	0x0000_0064
10	17	可设置	EXINT4	EXINT 线 4 中断	0x0000_0068
11	18	可设置	DMA1 通道 1	DMA1 通道 1 全局中断	0x0000_006C
12	19	可设置	DMA1 通道 2	DMA1 通道 2 全局中断	0x0000_0070
13	20	可设置	DMA1 通道 3	DMA1 通道 3 全局中断	0x0000_0074
14	21	可设置	DMA1 通道 4	DMA1 通道 4 全局中断	0x0000_0078
15	22	可设置	DMA1 通道 5	DMA1 通道 5 全局中断	0x0000_007C
16	23	可设置	DMA1 通道 6	DMA1 通道 6 全局中断	0x0000_0080
17	24	可设置	DMA1 通道 7	DMA1 通道 7 全局中断	0x0000_0084
18	25	可设置	ADC1	ADC1 的全局中断	0x0000_0088
19	26	可设置	CAN1_TX	CAN1 发送中断	0x0000_008C
20	27	可设置	CAN1_RX0	CAN1 接收 0 中断	0x0000_0090
21	28	可设置	CAN1_RX1	CAN1 接收 1 中断	0x0000_0094
22	29	可设置	CAN1_SE	CAN1 状态错误中断	0x0000_0098
23	30	可设置	EXINT9_5	EXINT 线[9: 5]中断	0x0000_009C
24	31	可设置	TMR1_BRK_TMR9	TMR1 停止中断和 TMR9 全局中断	0x0000_00A0
25	32	可设置	TMR1_OVF_TMR10	TMR1 溢出中断和 TMR10 全局中断	0x0000_00A4
26	33	可设置	TMR1_TRG_HALL1	TMR1 触发和 HALL 中断	0x0000_00A8
27	34	可设置	TMR1_CH	TMR1 通道中断	0x0000_00AC

28	35	可设置	TMR2	TMR2 全局中断	0x0000_00B0
29	36	-	-	保留	0x0000_00B4
30	37	可设置	TMR4	TMR4 全局中断	0x0000_00B8
31	38	可设置	I2C1_EVT	I <sup>2</sup> C1 事件中断	0x0000_00BC
32	39	可设置	I2C1_ETR	I <sup>2</sup> C1 错误中断	0x0000_00C0
33	40	-	-	保留	0x0000_00C4
34	41	-	-	保留	0x0000_00C8
35	42	-	-	保留	0x0000_00CC
36	43	可设置	SPI2	SPI2 全局中断	0x0000_00D0
37	44	可设置	USART1	USART1 全局中断	0x0000_00D4
38	45	可设置	USART2	USART2 全局中断	0x0000_00D8
39	46	可设置	USART3	USART3 全局中断	0x0000_00DC
40	47	可设置	EXINT15_10	EXINT 线[15: 10]中断	0x0000_00E0
41	48	可设置	ERTCAlarm	连到 EXINT 的 ERTC 闹钟中断	0x0000_00E4
42	49	可设置	OTGFS1	OTGFS1 唤醒中断	0x0000_00E8
43	50	-	-	保留	0x0000_00EC
44	51	-	-	保留	0x0000_00F0
45	52	-	-	保留	0x0000_00F4
46	53	-	-	保留	0x0000_00F8
47	54	-	-	保留	0x0000_00FC
48	55	-	-	保留	0x0000_0100
49	56	-	-	保留	0x0000_0104
50	57	可设置	TMR5	TMR5 全局中断	0x0000_0108
51	58	-	-	保留	0x0000_010C
52	59	-	-	保留	0x0000_0110
53	60	可设置	UART5	UART5 全局中断	0x0000_0114
54	61	-	-	保留	0x0000_0118
55	62	-	-	保留	0x0000_011C
56	63	可设置	DMA2 通道 1	DMA2 通道 1 全局中断	0x0000_0120
57	64	可设置	DMA2 通道 2	DMA2 通道 2 全局中断	0x0000_0124
58	65	可设置	DMA2 通道 3	DMA2 通道 3 全局中断	0x0000_0128
59	66	可设置	DMA2 通道 4_5	DMA2 通道 4 和 DMA2 通道 5 全局中断	0x0000_012C
60	67	-	-	保留	0x0000_0130
61	68	-	-	保留	0x0000_0134
62	69	-	-	保留	0x0000_0138

63	70	-	-	保留	0x0000_013C
64	71	-	-	保留	0x0000_0140
65	72	-	-	保留	0x0000_0144
66	73	-	-	保留	0x0000_0148
67	74	可设置	OTGFS	OTGFS 中断	0x0000_014C
68	75	-	-	保留	0x0000_0150
69	76	-	-	保留	0x0000_0154
70	77	可设置	CMP1	连到 EXINT 线 19 的 CMP1 中断	0x0000_0158
71	78	可设置	CMP2	连到 EXINT 线 20 的 CMP2 中断	0x0000_015C
72	79	-	-	保留	0x0000_0160
73	80	-	-	保留	0x0000_0164
74	81	-	-	保留	0x0000_0168
75	82	可设置	DMA2 通道 6_7	DMA2 通道 6 和 DMA2 通道 7 全局中断	0x0000_016C

#### 1.1.4 系统嘀嗒定时器 (SysTick)

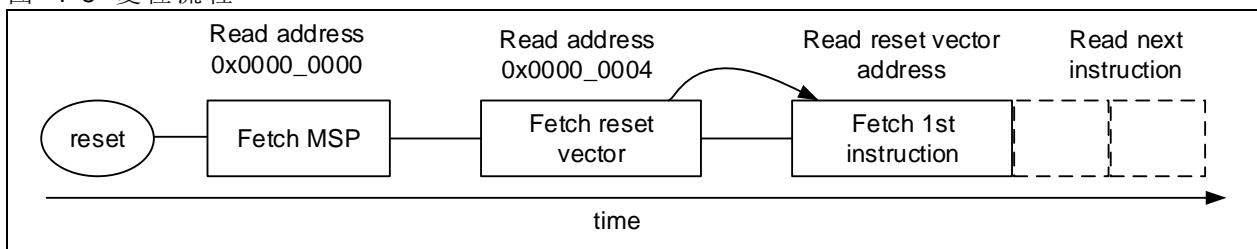
系统嘀嗒定时器是一个 24 位递减计数器，递减至零可自动重载计数初值。可产生周期性异常，用作嵌入式操作系统的多任务调度计数器，或对于无嵌入式操作系统，可用于调用需周期性执行的任务。系统嘀嗒定时器校准值固定值 9000，当系统嘀嗒时钟设定为 9MHz，产生 1ms 时间基准。

#### 1.1.5 复位流程

系统复位后以及处理器开始执行程序前，处理器会从 CODE 存储器中读出前两个字。

- 从地址 0x0000\_0000 处取出主栈指针 (MSP) 的初始值。
- 从地址 0x0000\_0004 处取出程序计数器 (PC) 的初始值，这个值是复位向量，LSB 必须是 1。然后从这个值所对应的地址处取指。

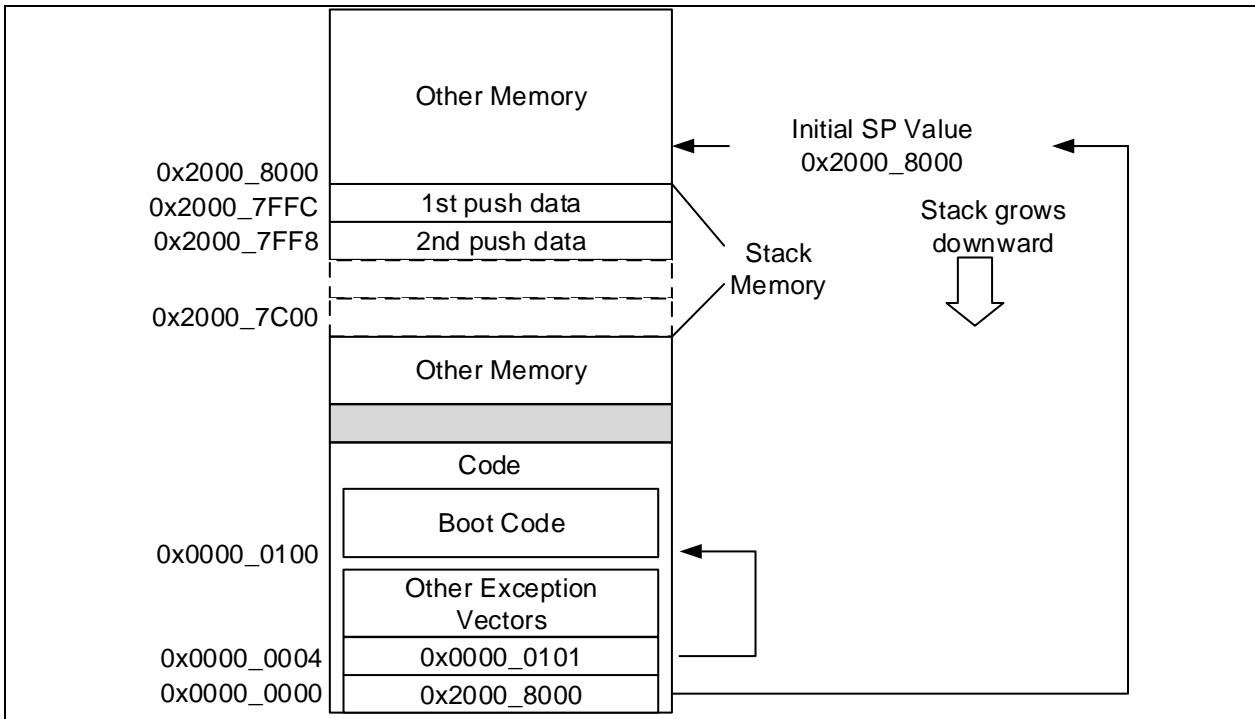
图 1-5 复位流程



Cortex™-M4 使用的是向下生长的满栈，所以 MSP 的初始值必须是堆栈内存的末地址加 1。举例来说，堆栈区域设定在 0x2000\_7C00~0x2000\_7FFF 之间，那么 MSP 的初始值必须是 0x2000\_8000。

向量表跟随在 MSP 的初始值之后。Cortex™-M4 是在 Thumb 状态下执行，所以向量表中的每个数值都必须将 LSB 置 1，所以，下图中使用 0x0000\_0101 来表示地址 0x0000\_0100。当 0x0000\_0100 处的指令得到执行后，就正式开始程序的执行。在此之前初始化 MSP 是必须的，因为可能第一条指令还没执行就会被 NMI 或是其他 fault 打断。MSP 初始化好后就可以为它们的服务程序准备好堆栈空间。

图 1-6 MSP 及 PC 初始化的一个范例



在 AT32WB415 中，可以将主闪存存储器、启动程序存储器或片上 SRAM 这三块存储器重映射到 0x0000\_0000~0x07FF\_FFFF 的 CODE 区，由 BOOT1 和 BOOT0 管脚来设定 CODE 从哪块存储器启动，当{BOOT1, BOOT0}=00/10 时，CODE 从主闪存存储器启动，当{BOOT1, BOOT0}=01 时，CODE 从启动程序存储器启动，当{BOOT1, BOOT0}=11 时，CODE 从片上 SRAM 启动。系统复位后或从待机模式退出时，BOOT1 和 BOOT0 管脚值都会被重新锁存。

启动程序存储器中包含内嵌的 Bootloader 程序，可提供 flash 编程功能，通过 USART1、USART2 或 USB 接口对 flash 进行重新编程；也可以提供通信协议栈等额外的固件，可被软件开发人员通过 API 调用。

## 1.2 寄存器的缩写说明

表 1-4 寄存器描述缩写说明

寄存器类型	说明
rw	可以读或写这些位
ro	只能读这些位
wo	只能写这些位；如果读这些位，则返回它们的复位值
rrc	可以读，读取这些位时，自动清除这些位
rw0c	可以读并写'0'清除这些位，写'1'将不对该位产生影响
rw1c	可以读并写'1'清除这些位，写'0'将不对该位产生影响
rw1s	可以读并写'1'设置这些位，写'0'将不对该位产生影响
tog	可以读，写'1'将翻转此位值，写'0'将不对该位产生影响
rwt	可以读，写任何值时，将触发事件
resd	保留

## 1.3 器件特征信息

表 1-5 器件特征信息相关寄存器地址和复位值

寄存器简称	基地址	复位值
F_SIZE	0x1FFF F7E0	0xFFFF
UID[31:0]	0x1FFF F7E8	0xFFFF XXXX
UID[63:32]	0x1FFF F7EC	0xFFFF XXXX
UID[95:64]	0x1FFF F7F0	0xFFFF XXXX

### 1.3.1 闪存容量寄存器

闪存容量寄存器提供该芯片闪存容量信息，用户可透过该寄存器取得闪存容量。

域	简称	复位值	类型	功能
位 15:0	F_SIZE	0xFFFF	ro	闪存容量，以 KByte 为单位 例如：0x0080 = 128KByte

### 1.3.2 器件电子签名

器件电子签名包含产品容量信息和器件唯一 ID（96 位 UID），它位于闪存的信息区块中。96 位器件唯一 ID 对任何器件来说都是独一无二的，且用户不可更改。ID 可以用来作为下列用途：

- 序列号；例如 USB 字串序列
- 或者做为密钥的一部分

域	简称	复位值	类型	功能
位 31:0	UID[31:0]	0xFFFF	ro	UID 的 bit31 到 bit0 信息

域	简称	复位值	类型	功能
位 31:0	UID[63:32]	0xFFFF	ro	UID 的 bit63 到 bit32 信息

域	简称	复位值	类型	功能
位 31:0	UID[95:64]	0xFFFF	ro	UID 的 bit95 到 bit64 信息

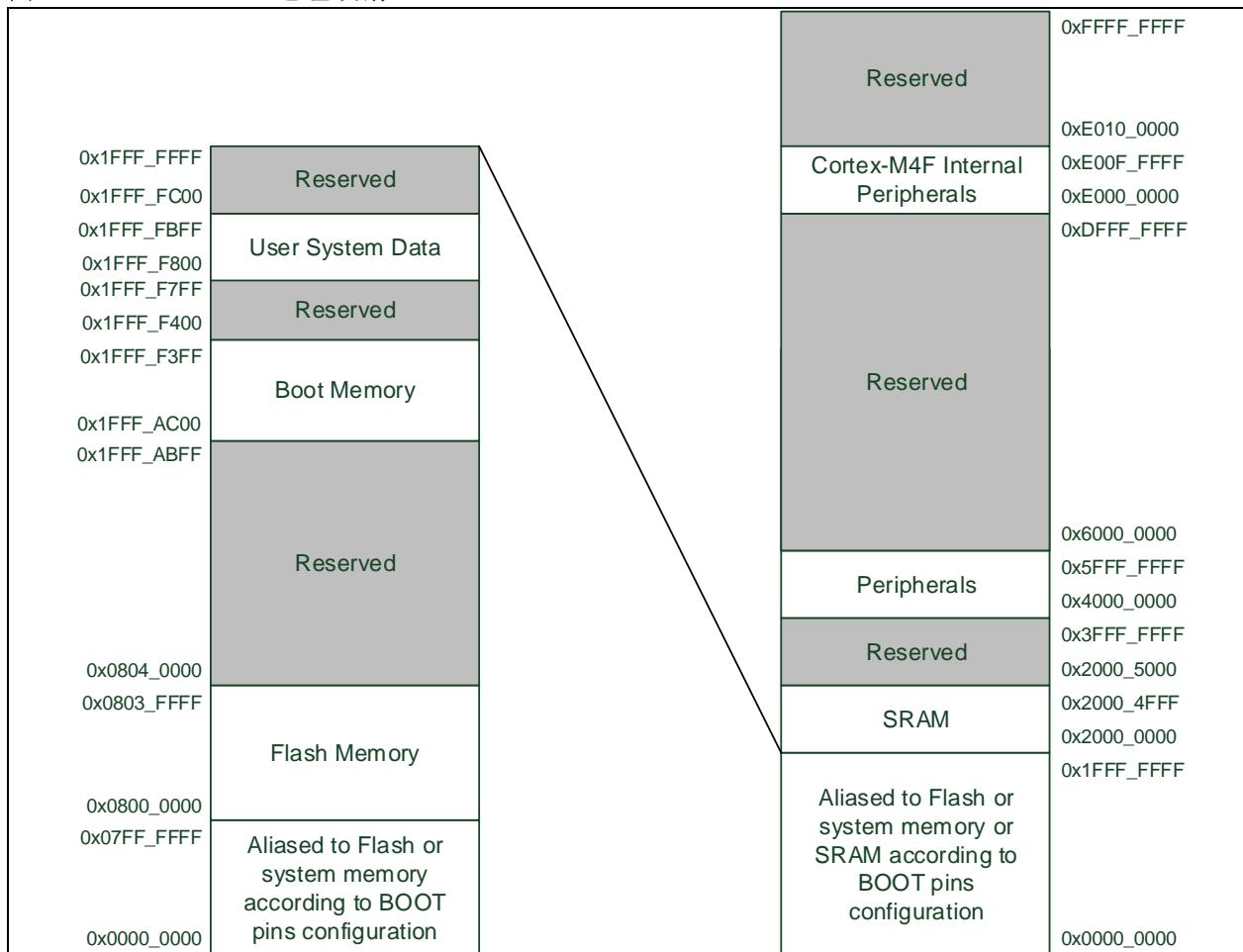
注意：UID[95:88]为 Series ID，AT32WB415 为 0x011。

## 2 存储器资源

### 2.1 内部存储器地址映射

芯片内部存储器包括程序存储器 flash，数据存储器 SRAM，外设寄存器和内核寄存器等。各区域地址映射如下图：

图 2-1 AT32WB415 地址映射



### 2.2 Flash存储器

AT32WB415 系列提供最大 256KB 的片上闪存，支持单周期最大 32 位读取操作。

闪存存储器由闪存控制器操作，有关闪存控制器的操作与寄存器配置信息请参考第 5 章节。

#### 闪存存储结构（256K）

主存储器只有闪存容量为 256K 字节的片 1 闪存，包含 128 个扇区，每扇区大小为 2K 字节。

表 2-1 256KB 闪存模块的组织

结构	名称	地址范围
主存储器	扇区 0	0x0800 0000 – 0x0800 07FF
	扇区 1	0x0800 0800 – 0x0800 0FFF
	扇区 2	0x0800 1000 – 0x0800 17FF
	扇区 3	0x0800 1800 – 0x0800 1FFF
	扇区 4	0x0800 2000 – 0x0800 27FF
	.	.
	扇区 127	0x0803 F800 – 0x0803 FFFF
信息块	启动程序代码区	0x1FFF AC00 – 0x1FFF F3FF

## 2.3 SRAM存储器

AT32WB415 系列内置 32K 字节的片上 SRAM，起始地址为 0x2000\_0000。它可以以字节、半字（16 位）或字（32 位）访问。

## 2.4 外设地址映射

表 2-2 各外设起始地址

总线	起始地址	外设
AHB	0x6000 0000 - 0xFFFF FFFF	保留
	0x5004 0000 - 0x5FFF FFFF	保留
	0x5000 0000 - 0x5003 FFFF	OTGFS
	0x4002 8000 - 0x4FFF FFFF	保留
	0x4002 3400 - 0x4002 7FFF	保留
	0x4002 3000 - 0x4002 33FF	CRC
	0x4002 2000 - 0x4002 23FF	闪存存储器接口 (FLASH)
	0x4002 1400 - 0x4002 1FFF	保留
	0x4002 1000 - 0x4002 13FF	时钟和复位管理 (CRM)
	0x4002 0800 - 0x4002 0FFF	保留
	0x4002 0400 - 0x4002 07FF	DMA2
	0x4002 0000 - 0x4002 03FF	DMA1
	0x4001 8400 - 0x4001 7FFF	保留
	0x4001 8000 - 0x4001 83FF	保留
APB2	0x4001 6000 - 0x4001 7FFF	保留
	0x4001 5800 - 0x4001 5BFF	保留
	0x4001 5400 - 0x4001 57FF	TMR11 定时器
	0x4001 5000 - 0x4001 53FF	TMR10 定时器
	0x4001 4C00 - 0x4001 4FFF	TMR9 定时器
	0x4001 3C00 - 0x4001 4BFF	保留
	0x4001 3800 - 0x4001 3BFF	USART1
	0x4001 3400 - 0x4001 37FF	保留
	0x4001 3000 - 0x4001 33FF	保留
	0x4001 2C00 - 0x4001 2FFF	TMR1 定时器
	0x4001 2800 - 0x4001 2BFF	保留
	0x4001 2400 - 0x4001 27FF	ADC1
	0x4001 2000 - 0x4001 23FF	保留
	0x4001 1C00 - 0x4001 1FFF	GPIO 端口 F
	0x4001 1800 - 0x4001 1BFF	保留
	0x4001 1400 - 0x4001 17FF	GPIO 端口 D
	0x4001 1000 - 0x4001 13FF	GPIO 端口 C
	0X4001 0C00 - 0x4001 0FFF	GPIO 端口 B
	0x4001 0800 - 0x4001 0BFF	GPIO 端口 A

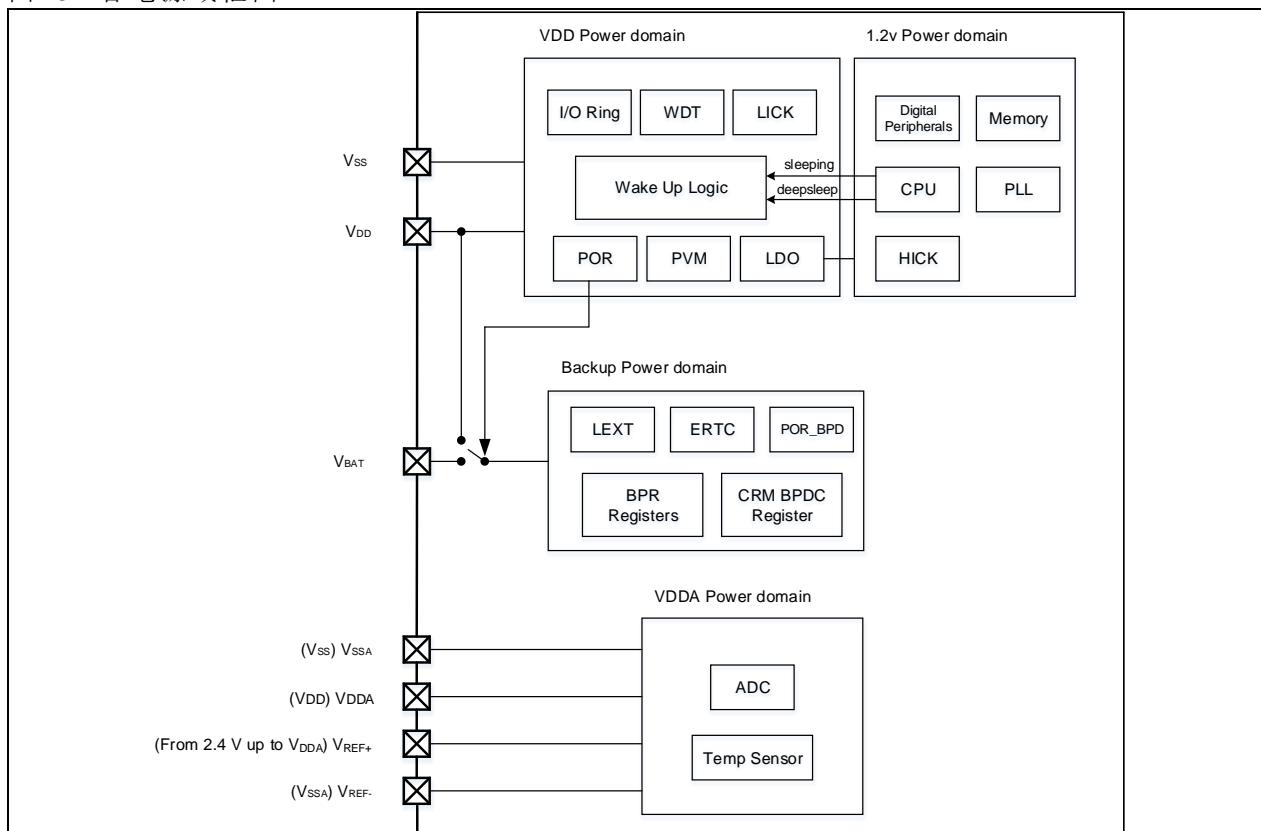
总线	起始地址	外设
APB1	0x4001 0400 - 0x4001 07FF	EXINT
	0x4001 0000 - 0x4001 03FF	IOMUX
	0x4000 8000 - 0x4000 FFFF	保留
	0x4000 7C00 - 0x4000 7FFF	保留
	0x4000 7800 - 0x4000 7BFF	保留
	0x4000 7400 - 0x4000 77FF	保留
	0x4000 7000 - 0x4000 73FF	电源控制 (PWC)
	0x4000 6800 - 0x4000 6FFF	保留
	0x4000 6400 - 0x4000 67FF	CAN1
	0x4000 6000 - 0x4000 63FF	保留
	0x4000 5C00 - 0x4000 5FFF	保留
	0x4000 5800 - 0x4000 5BFF	保留
	0x4000 5400 - 0x4000 57FF	I <sup>2</sup> C1
	0x4000 5000 - 0x4000 53FF	UART5
	0x4000 4C00 - 0x4000 4FFF	保留
	0x4000 4800 - 0x4000 4BFF	USART3
	0x4000 4400 - 0x4000 47FF	USART2
	0x4000 4000 - 0x4000 43FF	保留
	0x4000 3C00 - 0x4000 3FFF	保留
	0x4000 3800 - 0x4000 3BFF	SPI2
	0x4000 3400 - 0x4000 37FF	保留
	0x4000 3000 - 0x4000 33FF	看门狗 (WDT)
	0x4000 2C00 - 0x4000 2FFF	窗口看门狗 (WWDT)
	0x4000 2800 - 0x4000 2BFF	ERTC
	0x4000 2400 - 0x4000 27FF	CMP 控制器
	0x4000 2000 - 0x4000 23FF	保留
	0x4000 1C00 - 0x4000 1FFF	保留
	0x4000 1800 - 0x4000 1BFF	保留
	0x4000 1400 - 0x4000 17FF	保留
	0x4000 1000 - 0x4000 13FF	保留
	0x4000 0C00 - 0x4000 0FFF	TMR5 定时器
	0x4000 0800 - 0x4000 0BFF	TMR4 定时器
	0x4000 0400 - 0x4000 07FF	保留
	0x4000 0000 - 0x4000 03FF	TMR2 定时器

## 3 电源控制 (PWC)

### 3.1 简介

AT32WB415 系列设备工作电压范围为 2.6V 至 3.6V，正常工作温度范围为 -40~+105°C。AT32WB415 系列设备为了降低功耗，使用户可以在 CPU 运行时间要求、速度和功耗进行折中取舍，提供了三种省电模式——睡眠模式，深度睡眠模式和待机模式。AT32WB415 系列设备有三个电源域——VDD/VDDA 域，1.2V 域和电池供电域。其中 VDD/VDDA 域由电源直接供电，1.2V 域由 VDD/VDDA 域中嵌入的 LDO 供电，电池供电域由 V<sub>BAT</sub> 引脚供电。

图 3-1 各电源域框图



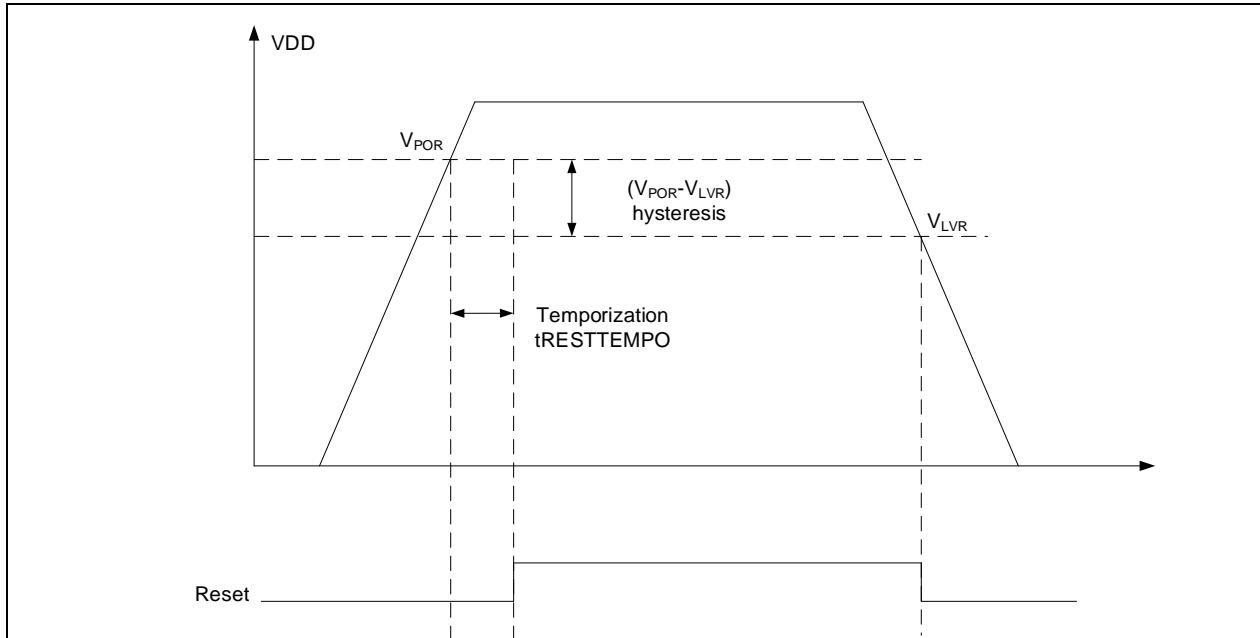
### 3.2 主要特点

- 具备三个电源域：VDD/VDDA 域、1.2V 内核域和电池供电域。
- 支持三种省电模式：睡眠模式、深度睡眠模式和待机模式。
- 内建电压调节器，提供 1.2V 给内核域。
- 提供电压监测器，能在电压低于或高于阈值时发出中断或事件。
- 当 VDD 供电关闭时，电池供电域由电池 V<sub>BAT</sub> 供电。
- VDD/VDDA 采用独立的数字和模拟地，用于隔离电源噪声。

### 3.3 上电下电复位

VDD/VDDA 域内置一个 POR 模拟模块用于产生电源复位，当 VDD 由 0V 上升至工作电压过程中，电源复位信号在 V<sub>POR</sub> 时刻被上电释放。当 VDD 由工作电压下降至 0V 过程中，电源复位信号在 V<sub>LVR</sub> 时刻被下电复位。上电复位过程，复位信号的释放相较于 VDD 升压过程存在一定的时间延迟，同时上电下电复位具有一定迟滞。

图 3-2 上电/下电复位波形图

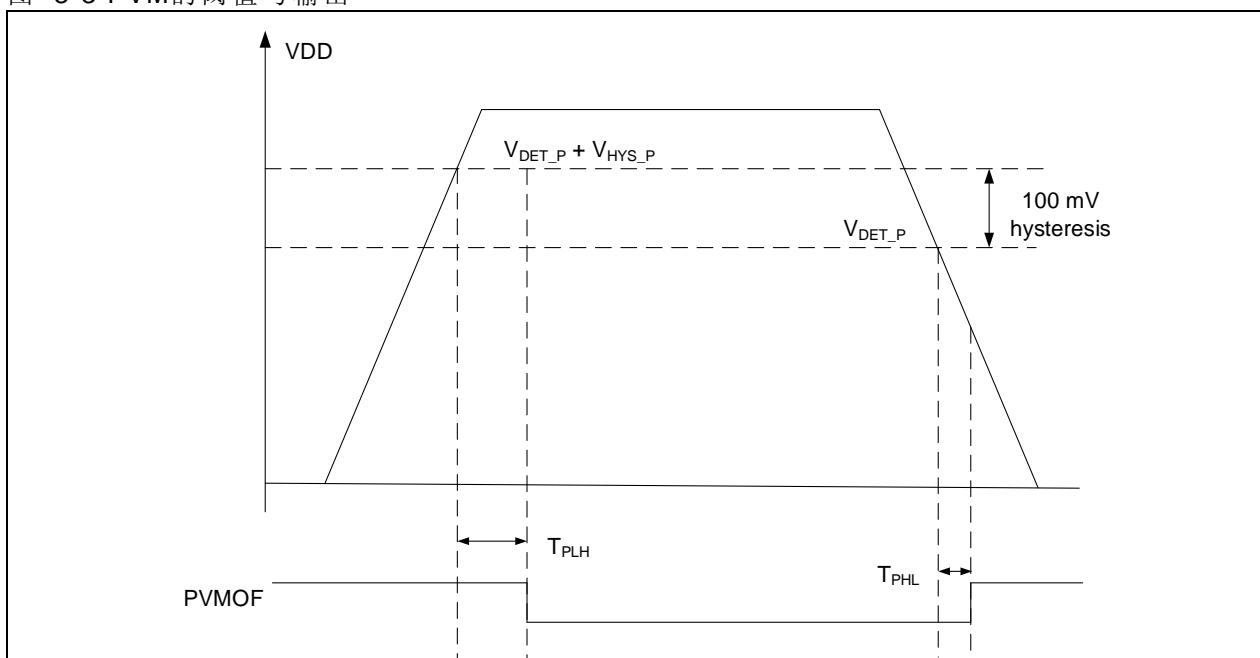


### 3.4 电压监测器 (PVM)

电压监测器 PVM 主要用来监控供电电源的跳变，可通过电源控制寄存器 (PWC\_CTRL) 中的 PVMEN 位开启电压监测功能，并通过 PVMSEL[2: 0]来选择监控阈值。

电压监测器开启后，电源控制及状态寄存器 (PWC\_CTRLSTS) 中的 PVMOF 位会指示 VDD 与设定阈值比较的结果，迟滞电压 VHYS\_P 为 100mV。当 VDD 越过 PVM 阈值边界时，产生的 PVMOF 位电平变化可以通过外部中断第 16 号线产生 PVM 中断。

图 3-3 PVM 的阈值与输出



### 3.5 电源域划分

#### 1.2V 域

1.2V 内核域包括 CPU 内核、存储器 SRAM、内嵌数字外设以及时钟锁相环 PLL，由 LDO (电压调节器) 供电，在三种省电模式下可以配置为全供电状态、低功耗状态或关闭状态。

### VDD/VDDA 域

VDD/VDDA 域包括 VDD 域和 VDDA 域两部分。VDD 域包括 I/O 电路、省电模式唤醒电路、看门狗 WDT、上电/掉电复位 (POR/LVR)、电压调节器 LDO 以及除 PC13、PC14 和 PC15 之外的所有 PAD 电路等。VDDA 域包括 ADC (AD 转换器)、温度传感器 Temp Sensor 等。

一般来说，为保证低电压时 ADC 的高精度，数字电路由 VDD 供电，模拟电路由 VDDA 独立供电，外部参考电压 VREF+连接至 VDDA 引脚，VREF-连接至 VSSA 引脚。

### 电池供电域

电池供电域包括 ERTC 电路、LEXT 振荡器、PC13、PC14 以及 PC15。电池供电域由 VDD 或 VBAT 引脚供电，当 VDD 主电源被切断时，电池供电域切换至 VBAT 引脚供电，ERTC 可以正常工作。

- 1) 当电池供电域由 VDD 供电时，PC13 可以作为通用 I/O 口、TAMPER 引脚、ERTC 校准时钟、ERTC 闹钟或秒输出，PC14 和 PC15 可以用于 GPIO 或 LEXT 引脚。（PC13 至 PC15 作为 I/O 口的速度必须限制在 2MHz 以下，最大负载为 30pF，而且这些 I/O 口绝对不能当作电流源）。
- 2) 当电池供电域由 VBAT 供电时，PC13 可以作为 TAMPER 引脚、ERTC 闹钟或秒输出，PC14 和 PC15 只能用于 LEXT 引脚。

电池供电域的电源开关不会因为 VDD 在上升阶段或是因为 VDD 掉电复位而断开与 VBAT 的连接。当主电源上 VDD 上电较快，电源开关还未切换至主电源 VDD 时，为防止电流从 VDD 注入到 VBAT，推荐在 VDD 与 VBAT 之间接一个低压降二极管。若应用中没有外部电池，VBAT 最好连接一个 100nF 的陶瓷滤波电容并在外部连接到 VDD。

## 3.6 省电模式

当 CPU 无需继续运行时，AT32WB415 提供三种低功耗模式（睡眠模式、深度睡眠模式、待机模式）可以实现更低的功耗。用户可以在启动时间，唤醒源，电源消耗等方面进行折中。此外在运行模式下，还可以通过降低系统时钟或关闭 APB 和 AHB 总线上未被使用的外设时钟来降低功耗。

### 睡眠模式 (Sleep Mode)

执行 WFI 或 WFE 指令可以进入睡眠状态。根据 Cortex™-M4 系统控制寄存器中的 SLEEPONEXIT 位有两种进入睡眠模式的机制：

当 SLEEPDEEP=0, SLEEPONEXIT=0 时，执行 WFI 或 WFE 指令，可立即进入睡眠模式，即 SLEEP-NOW 模式；

当 SLEEPDEEP=0, SLEEPONEXIT=1 时，执行 WFI 指令，当系统从最低优先级的中断处理程序中退出时，可立即进入睡眠模式，即 SLEEP-ON-EXIT 模式。

在睡眠模式下，CPU 时钟关闭，其他时钟均正常工作，电压调节器正常工作，所有的 I/O 引脚都保持它们在运行模式时的状态，调节器 LDO 以正常功耗模式提供 1.2V 电源 (CPU 内核、内存和内嵌外设)。

- 1) 执行 WFI 指令进入睡眠模式时，只要产生外设中断，都能使系统退出睡眠模式。
- 2) 执行 WFE 指令进入睡眠模式时，存在两种方式的唤醒事件，使系统退出睡眠模式：
  - 使能任一外设中断（未在 NVIC 中使能）且使能 SEVONPEND 位可以产生唤醒事件。

系统唤醒后，需清除外设中断挂起位和 NVIC 通道挂起位。

- 配置内部 EXINT 线为事件模式来产生唤醒事件。

从执行 WFE 指令进入睡眠模式唤醒所需的时间最短，因为没有时间损失在中断的进入或退出上。

### 深度睡眠模式 (Deepsleep Mode)

通过设置 Cortex™-M4 系统控制寄存器中的 SLEEPDEEP 位，清除电源控制寄存器 (PWC\_CTRL) 中的 LPSEL 位，执行 WFI 或 WFE 指令才能进入深度睡眠模式。

还可以通过设置电源控制寄存器 (PWC\_CTRL) 中 VRSEL 位选择深度睡眠模式下电压调节器的工作状态。当 VRSEL=0，电压调节器正常工作，当 VRSEL=1，电压调节器处于低功耗模式。

在深度睡眠模式下，所有 1.2V 时钟关闭，HICK 和 HEXT 振荡器都被关闭，电压调节器以正常工作或低功耗工作状态给 1.2V 域供电，所有 I/O 管脚都保持它们在运行模式时的状态，SRAM 和寄存器内容保持。

- 1) 执行 WFI 指令进入深度睡眠模式，任一外部中断线在中断模式下产生的中断，即可使系统退出深度睡眠模式。
- 2) 如果执行 WFE 指令进入深度睡眠模式，任一外部中断线在事件模式下产生的事件，即可使系统退出深度睡眠模式。

系统从深度睡眠模式退出时，HICK RC 振荡器开启并在稳定后被选为系统时钟。当电压调节器处于低功耗模式时，退出深度睡眠模式时，需要额外等待电压调节器稳定，从而会增加一段额外的唤醒时间。

### 待机模式（Standby Mode）

待机模式可最大限度的降低系统功耗，在该模式下，电压调节器关闭，只有电池供电的寄存器和待机电路维持供电，其他的 1.2V 供电区域，PLL、HICK 和 HEXT 振荡器都被断电。寄存器和 SRAM 中的内容也会丢失。

通过设置 Cortex™-M4 系统控制寄存器中的 SLEEPDEEP 位，设置电源控制寄存器(PWC\_CTRL)中 LPSEL 位，并清除电源控制及状态寄存器 (PWC\_CTRLSTS) 中的 SWEF 位的情况下，最后执行 WFI 或 WFE 指令即可进入待机模式。

在待机模式下，除了复位管脚、被设置为防侵入或校准输出时的 TAMPER 管脚和被使能的唤醒管脚之外，所有的 I/O 管脚处于高阻态。

当产生 WKUP 管脚的上升沿、ERTC 闹钟事件的上升沿、ERTC 入侵事件、ERTC 时间戳、ERTC 周期性自动唤醒、NRST 管脚上外部复位、WDT 复位时，微控制器将退出待机模式。

### 调试配置

默认情况下，在进行调试时，微处理器一旦进入深度睡眠或待机模式，会因为 Cortex™-M4 的内核失去了时钟而失去调试连接。只需通过设置 DEBUG 控制寄存器 (DEBUG\_CTRL) 中的某些配置位，就可以在低功耗模式下继续调试软件。

## 3.7 PWC 寄存器

必须用字（32 位）的方式操作这些外设寄存器。

表 3-1 PWC 寄存器映像和复位值

寄存器简称	基址偏移量	复位值
PWC_CTRL	0x00	0x0000 0000
PWC_CTRLSTS	0x04	0x0000 0000

### 3.7.1 电源控制寄存器（PWC\_CTRL）

域	简称	复位值	类型	功能
位 31: 9	保留	0x0000 00	resd	保持默认值。
位 8	BPWEN	0	rw	电池供电区域的写入使能（Battery powered domain write enable） 0: 关闭； 1: 开启。 注： 复位后，电池供电区域禁止写入。要对电池供电区域进行写操作的话，需先设置这位为允许写入状态。
位 7: 5	PVMSEL	0x0	rw	电压监测临界值选择（Power voltage monitoring boundary select） 000: 未用，禁止配置； 001: 2.3V； 010: 2.4V； 011: 2.5V； 100: 2.6V； 101: 2.7V； 110: 2.8V； 111: 2.9V。
位 4	PVMEN	0	rw	电压监测使能（Power voltage monitoring enable） 0: 关闭； 1: 开启。
位 3	CLSEF	0	wo	清除 SEF 标志（Clear SEF flag） 0: 无效； 1: 清除 SEF 标志。

位 2	CLSWEF	0	wo	注：该位在清除 SEF 后由硬件将其清零，且任何时刻读取该位返回值均是零。 清除 SWEF 标志 (Clear SWEF flag) 0: 无效； 1: 清除 SWEF 标志。 注： 实际 SWEF 标志的清除大约需要 2 个系统时钟周期； 该位在清除 SWEF 后由硬件将其清零，且任何时刻读取该位返回值均是零。
位 1	LPSEL	0	rw	SLEEPDEEP 状态下的低功耗模式选择位 (Low power mode select when Cortex™-M4 sleepdeep) 0: 进入 DEEPSLEEP 模式； 1: 进入待机模式。
位 0	VRSEL	0	rw	DEEPSLEEP 模式下电压调节器状态选择 (Voltage regulator state select when deepsleep mode) 0: 正常开启； 1: 低功耗模式。

### 3.7.2 电源控制及状态寄存器 (PWC\_CTRLSTS)

与标准的 APB 读相比，读此寄存器需要额外的 APB 周期

域	简称	复位值	类型	功能
位 31: 9	保留	0x000000	resd	保持默认值。
位 8	SWPEN	0	rw	待机唤醒引脚使能 (Standby wake-up pin enable) 0: 关闭 (该引脚可用作通用 I/O)； 1: 开启 (该引脚被强置为输入下拉模式，且无法再用作通用 I/O)。 注：在系统复位时硬件将清除这一位。
位 7: 3	保留	0x00	resd	保持默认值。
位 2	PVMOF	0	ro	电源电压监测输出标志 (Power voltage monitoring output flag) 0: 电源电压高于临界值； 1: 电源电压低于临界值。 注：待机模式下电压监测停止工作。
位 1	SEF	0	ro	进入待机模式标志 (Standby mode entry flag) 0: 未进过待机模式； 1: 有进过待机模式。 注：该位被硬件置起（进入待机模式时），由 POR/LVR 或写 CLSEF 位将其清零。
位 0	SWEF	0	ro	待机唤醒事件标志 (Standby wake-up event flag) 0: 无唤醒事件产生； 1: 有唤醒事件产生。 注： 该位被硬件置起（产生唤醒事件时），由 POR/LVR 或写 CLSWEF 位将其清零。 唤醒事件将由以下几种情况产生： 在待机唤醒引脚上出现上升沿时，将产生唤醒事件； 出现 RTC 闹钟事件时，将产生唤醒事件； 待机唤醒引脚保持高电平期间使能该待机唤醒引脚，将产生唤醒事件。

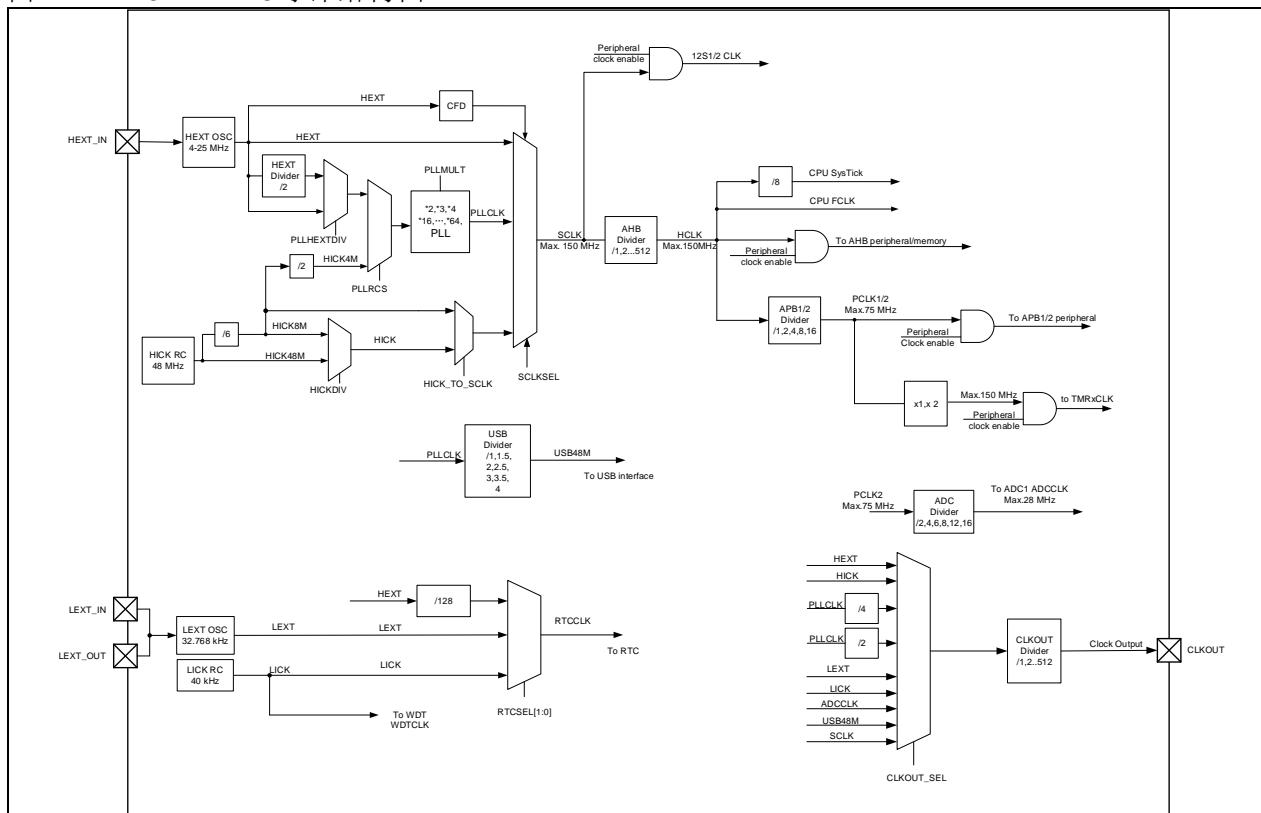
## 4 时钟和复位管理 (CRM)

4.1 时钟

AT32WB415 的时钟源

包含：HEXT 振荡器时钟，HICK 振荡器时钟，PLL 时钟，LEXT 振荡器时钟和 LICK 振荡器时钟。时钟结构如下：

图 4-1 AT32WB415时钟结构图



AHB、APB1 和 APB2 的频率都支持多种分频。AHB 域的最大频率是 150MHz，APB1 和 APB2 域的最大允许频率是 75MHz。

#### 4.1.1 时钟源

- #### ● HEXT 振荡器时钟,

**HEXT** 端口时钟可以提供频率高达 25MHz 的外部时钟。外部时钟信号必须连到 **HEXT IN** 引脚。

- ## ● HICK 振荡器时钟

**HICK** 振荡器时钟由芯片内的高速 **RC** 振荡器提供。**HICK** 时钟的内部频率为 **48MHz**，频率精度较差，每颗芯片的 **HICK** 时钟频率在出厂前已经被校准到 **1%** (**25° C**)，工厂校准值被装载到时钟控制寄存器的 **HICKCAL[7: 0]**位。考虑不同的电压或环境温度对 **HICK** 的 **RC** 振荡器的影响，用户可以通过时钟控制寄存器里的 **HICKTRIM[5: 0]**位来调整 **HICK** 频率。

HICK 时钟直到稳定后才会被释放出来。

- ### ● PLL 时钟

**PLL** 的输入时钟源可以选择 **HICK** 时钟或 **HEXT** 时钟且输入时钟范围为 2M~16MHz 之间。使用 **PLL** 前，一定要先选择输入时钟源和倍频因子，否则，**PLL** 使能后，这些参数将无法改动。**PLL** 时钟直到稳定后才会被释放出来。

- #### ● LEXT 振荡器时钟

**LEXT** 振荡器时钟包括 **LEXT** 晶体/陶瓷谐振器和 **LEXT** 旁路时钟两个时钟源。

## LEXT 晶体/陶瓷谐振器

**LEXT** 晶体/陶瓷谐振器提供一个低功耗且精确的 32.768KHz 低速时钟源。**LEXT** 时钟直到稳定后，才会被释放出来。

### LEXT 旁路时钟

在 LEXT 旁路模式下，可以提供最高频率达 32.768kHz 的外部时钟源。外部时钟信号必须连到 LEXT\_IN 引脚，并且 LEXT\_OUT 引脚也一定要保持悬空。

### ● LICK 振荡器时钟

LICK 振荡器时钟由芯片内的低速 RC 振荡器提供，作为一个频率为大约 40kHz (30kHz 和 60kHz 之间) 的低功耗时钟源，它可以为看门狗和自动唤醒单元提供时钟，并能在深度睡眠和待机模式下保持运行。LICK 时钟直到稳定后，才会被释放出来。

## 4.1.2 系统时钟

系统复位以后，系统时钟使用 HICK 时钟作为默认时钟。系统时钟可在 HICK 振荡器时钟、HEXT 振荡器时钟和 PLL 时钟之间进行灵活切换，只有当目标时钟源稳定后，系统时钟切换才会发生。当 HICK 振荡器时钟直接作为系统时钟或间接通过 PLL 作为系统时钟时，它将无法被停止。

## 4.1.3 外设时钟

大多数外设使用系统时钟 HCLK、PCLK1 或 PCLK2 时钟。个别外设还有专用时钟。

系统滴答定时器 (SysTick) 使用 HCLK 或 HCLK 的 8 分频作为时钟。

ADC 使用 APB2 时钟的 2、4、6、8、12、16 分频作为时钟。

定时器使用 APB1/2 作为时钟，特别地，当 APB 预分频系数是 1 时，定时器的时钟频率等于 APB1/2 的时钟频率；当 APB 预分频系数不为 1 时，定时器的时钟频率等于 APB1/2 时钟频率的 2 倍。

USB 时钟只能选择 PLL 分频时钟，当选 PLL 分频时钟时，USB 分频器提供 48MHz 的 USBCLK 时钟，PLL 需设置为  $48 \times N \times 0.5$  MHz ( $N=2,3,4,5,\dots$ )。

ERTC 的时钟源有：HEXT 振荡器 128 分频时钟，LEXT 振荡器时钟 LICK 振荡器时钟。ERTC 的时钟源一旦选择后就不可再更改，只有将电池供电域复位后才能重新配置 ERTC 时钟源。当 VDD 掉电时，ERTC 使用 LEXT 作为时钟的话，ERTC 可以继续工作，但 ERTC 使用 HEXT 或 LICK 作为时钟源时，由于 HEXT 和 LICK 均掉电，会导致 ERTC 状态不定。

看门狗使用 LICK 振荡器时钟作为时钟源。硬件选项或软件开启看门狗后，将强制打开 LICK 振荡器，LICK 振荡器稳定后，才给看门狗提供时钟。

## 4.1.4 时钟失效检测

当 HEXT 时钟直接或间接作为系统时钟时，为防止 HEXT 时钟出现故障，特设计了时钟失效检测模块 (CFD)。当 HEXT 时钟出现故障，CFD 侦测到失效后，将时钟失效事件送到 TMR1 的刹车输入端，并产生 CFD 中断，此 CFD 中断直接连到 CPU 的 NMI 中断，供软件完成营救操作。NMI 中断将一直重复执行，直到 CFD 中断挂起位被清除为止，所以在 NMI 的处理程序中必须清除 CFD 中断。当 HEXT 时钟出现故障时，将导致系统时钟切换到 HICK 时钟，同时关闭 CFD，关闭 HEXT 时钟，如果 HEXT 时钟通过 PLL 做为系统时钟时，也会关闭 PLL 模块。

## 4.1.5 自动滑顺频率切换

当系统时钟源从其他时钟切换到 PLL 或是 AHB 预分频由大切换到小时，为了使系统稳定顺滑切换，特设计了自动顺滑频率切换功能，当系统频率操作目标大于 108MHz 时，建议开启自动顺滑频率切换功能。当自动顺滑频率切换功能开启时，硬件会暂停 AHB 总线，直到整个自动顺滑频率切换才恢复。此期间 DMA 仍正常工作，中断事件会被记忆并待 AHB 总线恢复后 NVIC 即可处理。

## 4.1.6 内部时钟输出

微控制器允许输出内部时钟信号到外部 CLKOUT 引脚。ADC CLK、USB48M、SCLK、LICK、LEXT、HICK、HEXT、除 2 的 PLL 时钟以及除 4 的 PLL 时钟这 9 个时钟信号可输出到 CLKOUT。

## 4.1.7 中断

微控制器为每个时钟源设计了一个稳定标志，当用户开启一个时钟源后，可查询对应的时钟源的稳定标志来判断时钟是否稳定。当用户开启对应时钟源的中断使能的话，将产生中断请求。

当 HEXT 时钟出现故障，CFD 侦测到失效后，将产生 CFD 中断，此中断直接连到 CPU 的 NMI 中断。

## 4.2 复位

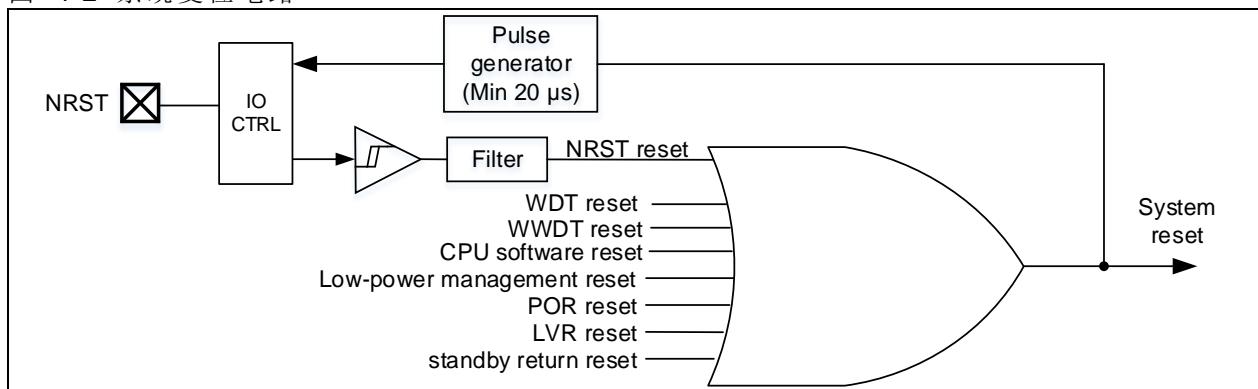
### 4.2.1 系统复位

AT32WB415 系统复位包括以下复位源:

- NRST 复位: 外部 NRST 管脚复位
- WDT 复位: 看门狗溢出复位
- WWDT 复位: 窗口看门狗溢出复位
- CPU 软件复位: Cortex™-M4 软件复位
- 低功耗管理复位: 将用户选择字节中的 nSTDBY\_RST 位清 0 并进入待机模式, 将产生低功耗管理复位; 将用户选择字节中的 nDEPSLP\_RST 位清 0 并进入深度睡眠模式, 也将产生低功耗管理复位。
- POR 复位: 上电复位
- LVR 复位: 掉电复位
- 从待机模式中返回等事件产生复位。

NRST 复位, WDT 复位, WWDT 复位, 软件复位和低功耗管理复位将复位所有寄存器至它们的复位状态, 时钟控制器的 CRM\_CTRLSTS 寄存器和电池供电域中的寄存器除外; 上电复位、掉电复位或者从待机模式中返回等事件产生复位会复位所有寄存器至复位状态, 电池供电寄存器除外。

图 4-2 系统复位电路



### 4.2.2 电池供电域复位

电池供电域复位包括以下复位源:

- 电池供电域软件复位: 设置电池供电域控制寄存器 (CRM\_BPDC) 中的 BPDRST 位来产生复位
  - 在 VDD 和 VBAT 两者掉电的前提下, VDD 或 VBAT 再上电将产生复位。
- 电池供电域软件复位只影响电池供电域。

## 4.3 CRM 寄存器

下表列出了 CRM 寄存器的映像和复位值。

可以用字节 (8 位)、半字 (16 位) 或字 (32 位) 的方式操作这些外设寄存器。

表 4-1 CRM 寄存器映像和复位值

寄存器简称	基址偏移量	复位值
CRM_CTRL	0x000	0x0000 XX83
CRM_CFG	0x004	0x0000 0000
CRM_CLKINT	0x008	0x0000 0000
CRM_APB2RST	0x00C	0x0000 0000
CRM_APB1RST	0x010	0x0000 0000
CRM_AHBEN	0x014	0x0000 0014

CRM_APB2EN	0x018	0x0000 0000
CRM_APB1EN	0x01C	0x0000 0000
CRM_BPDC	0x020	0x0000 0000
CRM_CTRLSTS	0x024	0x0C00 0000
CRM_AHBRST	0x028	0x0000 0000
CRM_PLL	0x02C	0x0000 1F10
CRM_MISC1	0x030	0x0000 0000
CRM_OTG_EXTCTRL	0x044	0x0000 0000
CRM_MISC2	0x054	0x0000 000D

### 4.3.1 时钟控制寄存器 (CRM\_CTRL)

访问：无等待状态，字，半字和字节访问

域	简称	复位值	类型	功能
位 31: 26	保留	0x00	resd	请保持默认值。
位 25	PLLSTBL	0x0	ro	PLL 时钟稳定 (PLL clock stable) 该位待 PLL 稳定后由硬件置起。 0: 未稳定; 1: 已稳定。
位 24	PLLEN	0x0	rw	PLL 使能 (PLL enable) 该位可由软件置起或清除，也可在进入待机或深度睡眠模式时，由硬件清除。当系统时钟为 PLL 时钟时，该位无法清除。 0: 关闭; 1: 开启。
位 23: 20	保留	0x0	resd	保持默认值。
位 19	CFDEN	0x0	rw	时钟失效检测使能 (Clock Failure Detection enable) 0: 关闭; 1: 开启。
位 18	HEXTBYPS	0x0	rw	HEXT 旁路使能 (High speed external crystal bypass) 只有在 HEXT 关闭时，软件才能操作该位。 0: 关闭; 1: 开启。
位 17	HEXTSTBL	0x0	ro	HEXT 时钟稳定 (High speed external crystal stable) 该位待 HEXT 稳定后由硬件置起。 0: 未稳定; 1: 已稳定。
位 16	HEXTEN	0x0	rw	HEXT 使能 (High speed external crystal enable) 该位可由软件置起或清除，也可在进入待机或深度睡眠模式时，由硬件清除。当系统时钟有用到 HEXT 时，该位无法清除。 0: 关闭; 1: 开启。

位 15: 8	HICKCAL	0xXX	rw	HICK 时钟校准值 (High speed internal clock calibration) 默认值为出厂校准初始值。 HICK 输出频率为 48 MHZ 时, 每 HICKCAL 数值的变化对应频率调整 240 kHz (设计值); HICK 输出频率是 8 MHZ 时, 每 HICKCAL 数值的变化对应频率调整 40 kHz (设计值)。 注意: 此位只有在 HICKCAL_KEY[7:0] 为 0x5A 的时候可被写入。
位 7: 2	HICKTRIM	0x20	rw	HICK 时钟调整值 (High speed internal clock trimming) 该数值和 HICKCAL[7: 0] 数值共同决定 HICK 振荡器的频率, 默认数值为 32, 可以把 HICK 调整到精度±1%。
位 1	HICKSTBL	0x1	ro	HICK 时钟稳定 (High speed internal clock stable) 该位待 HICK 稳定后由硬件置起。 0: 未稳定; 1: 已稳定。
位 0	HICKEN	0x1	rw	HICK 使能 (High speed internal clock enable) 该位可由软件置起或清除, 在退出待机或深度睡眠模式, 或 HEXT 发生故障时, 该位也可被硬件置起。当系统时钟有用到 HICK 时, 该位无法清除。 0: 关闭; 1: 开启。

### 4.3.2 时钟配置寄存器 (CRM\_CFG)

访问: 0 到 2 个等待周期, 字, 半字和字节访问, 只有当访问发生在时钟切换时, 才会插入 1 或 2 个等待周期。

域	简称	复位值	类型	功能
位 31	保留	0x0	resd	请保持默认值。
位 26: 24	CLKOUT_SEL	0x0	rw	内部时钟输出选择 (Clock output selection) CLKOUT_SEL[3] 在 CRM_MISC1 寄存器的位 16。 0000: 无; 0001: 保留; 0010: LICK; 0011: LEXT; 0100: SCLK; 0101: HICK; 0110: HEXT; 0111: PLL/2; 1100: PLL/4; 1101: USB; 1110: ADC。
位 27 位 23: 22	USBDIV	0x0	rw	USB 分频因子 (USB division) PLL 时钟分频后作为 USB 时钟。 000: 1.5 倍分频; 001: 不分频; 010: 2.5 倍分频; 011: 2 倍分频; 100: 3.5 倍分频; 101: 3 倍分频; 110: 4 倍分频; 111: 4 倍分频。

				PLL 倍频系数 (PLL multiplication factor) { 位 30: 29, 位 21: 18} 000000: 2 倍频 000010: 4 倍频 ..... 001100: 14 倍频 001110: 16 倍频 010000: 17 倍频 010010: 19 倍频 ..... 111110: 63 倍频 111111: 64 倍频。 注意: PLLRANGE 位须搭配 PLL 倍频后的频率值进行设置
位 30: 29 位 21: 18	PLLMULT	0x00	rw	HEXT 分频后作为 PLL 输入时钟源 (HEXT division selection for PLL entry clock) 0: 不分频; 1: 分频系数为 2。
位 17	PLLHEXTDIV	0x0	rw	PLL 输入时钟选择 (PLL reference clock select) 0: HICK 分频时钟 (4MHz) 作为 PLL 输入时钟; 1: HEXT 时钟作为 PLL 输入时钟源。
位 16	PLLRCSCS	0x0	rw	ADC 分频因子 (ADC division) PCLK 分频后作为 ADC 时钟。 000: 2 分频; 001: 4 分频; 010: 6 分频; 011: 8 分频; 100: 2 分频; 101: 12 分频; 110: 8 分频; 111: 16 分频。
位 28 位 15: 14	ADCDIV	0x0	rw	APB2 分频因子 (APB2 division) HCLK 分频后作为 APB2 时钟。 0XX: 不分频; 100: 2 分频; 101: 4 分频; 110: 8 分频; 111: 16 分频。 注意: 软件必须保证 APB2 时钟频率不超过 75MHz。
位 13: 11	APB2DIV	0x0	rw	APB1 分频因子 (APB1 division) HCLK 分频后作为 APB1 时钟。 0XX: 不分频; 100: 2 分频; 101: 4 分频; 110: 8 分频; 111: 16 分频。 注意: 软件必须保证 APB1 时钟频率不超过 75MHz。
位 10: 8	APB1DIV	0x0	rw	AHB 分频因子 (AHB division) SCLK 分频后作为 AHB 时钟。 0XXX: 不分频; 1000: 2 分频; 1001: 4 分频; 1010: 8 分频; 1011: 16 分频; 1100: 64 分频; 1101: 128 分频; 1110: 256 分频; 1111: 512 分频。
位 7: 4	AHBDIV	0x0	rw	

位 3: 2	SCLKSTS	0x0	ro	系统时钟选择状态位 (System clock select status) 00: HICK; 01: HEXT; 10: PLL; 11: 保留, 保持默认值。
位 1: 0	SCLKSEL	0x0	rw	系统时钟选择 (System clock select) 00: HICK; 01: HEXT; 10: PLL; 11: 保留, 保持默认值。

### 4.3.3 时钟中断寄存器 (CRM\_CLKINT)

访问: 无等待周期, 字, 半字和字节访问

域	简称	复位值	类型	功能
位 31: 24	保留	0x00	resd	保持默认值。
位 23	CFDFC	0x0	wo	清除时钟失效标志 (Clock failure detection flag clear) 由软件写'1'清除 CFDF。 0: 不清除; 1: 清除。
位 22: 21	保留	0x0	resd	保持默认值。
位 20	PLLSTBLFC	0x0	wo	清除 PLL 稳定标志 (PLL stable flag clear) 由软件写'1'清除 PLLSTBLF。 0: 不清除; 1: 清除。
位 19	HEXTSTBLFC	0x0	wo	清除 HEXT 稳定标志 (HEXT stable flag clear) 由软件写'1'清除 HEXTSTBLF。 0: 不清除; 1: 清除。
位 18	HICKSTBLFC	0x0	wo	清除 HICK 稳定标志 (HICK stable flag clear) 由软件写'1'清除 HICKSTBLF。 0: 不清除; 1: 清除。
位 17	LEXTSTBLFC	0x0	wo	清除 LEXT 稳定标志 (LEXT stable flag clear) 由软件写'1'清除 LEXTSTBLF。 0: 不清除; 1: 清除。
位 16	LICKSTBLFC	0x0	wo	清除 LICK 稳定标志 (LICK stable flag clear) 由软件写'1'清除 LICKSTBLF。 0: 不清除; 1: 清除。
位 15: 13	保留	0x0	resd	保持默认值。
位 12	PLLSTBLIEN	0x0	rw	PLL 稳定中断使能 (PLL stable interrupt enable) 0: 关闭; 1: 开启。
位 11	HEXTSTBLIEN	0x0	rw	HEXT 稳定中断使能 (HEXT stable interrupt enable) 0: 关闭; 1: 开启。
位 10	HICKSTBLIEN	0x0	rw	HICK 稳定中断使能 (HICK stable interrupt enable) 0: 关闭; 1: 开启。
位 9	LEXTSTBLIEN	0x0	rw	LEXT 稳定中断使能 (LEXT stable interrupt enable) 0: 关闭; 1: 开启。

位 8	LICKSTBLIEN	0x0	rw	LICK 稳定中断使能 (LICK stable interrupt enable) 0: 关闭; 1: 开启。
位 7	CFDF	0x0	ro	时钟失效标志 (Clock Failure Detection flag) 在 HEXT 时钟出现故障时, 由硬件置起。 0: 未出现; 1: 出现。
位 6: 5	保留	0x0	resd	保持默认值。
位 4	PLLSTBLF	0x0	ro	PLL 稳定标志 (PLL stable flag) 由硬件置起。 0: 未稳定; 1: 已稳定。
位 3	HEXTSTBLF	0x0	ro	HEXT 稳定标志 (HEXT stable flag) 由硬件置起。 0: 未稳定; 1: 已稳定。
位 2	HICKSTBLF	0x0	ro	HICK 稳定标志 (HICK stable flag) 由硬件置起。 0: 未稳定; 1: 已稳定。
位 1	LEXTSTBLF	0x0	ro	LEXT 稳定标志 (LEXT stable flag) 由硬件置起。 0: 未稳定; 1: 已稳定。
位 0	LICKSTBLF	0x0	ro	LICK 稳定中断标志 (LICK stable flag) 由硬件置起。 0: 未稳定; 1: 已稳定。

#### 4.3.4 APB2外设复位寄存器 (CRM\_APB2RST)

访问: 无等待周期, 字, 半字和字节访问

域	简称	复位值	类型	功能
位 31: 22	保留	0x000	resd	保持默认值。
位 21	保留	0x0	resd	保持默认值。
位 20	TMR10RST	0x0	rw	TMR10 复位 (TMR10 reset) 0: 无复位; 1: 复位。
位 19	TMR9RST	0x0	rw	TMR9 复位 (TMR9 reset) 0: 无复位; 1: 复位。
位 18: 15	保留	0x0	resd	保持默认值。
位 14	USART1RST	0x0	rw	USART1 复位 (USART1 reset) 0: 无复位; 1: 复位。
位 13	保留	0x0	resd	保持默认值。
位 12	保留	0x0	resd	保持默认值。
位 11	TMR1RST	0x0	rw	TMR1 复位 (TMR1 reset) 0: 无复位; 1: 复位。
位 10	保留	0x0	resd	保持默认值。

位 9	ADC1RST	0x0	rw	ADC1 复位 (ADC1 reset) 0: 无复位; 1: 复位。
位 8	保留	0x0	resd	保持默认值。
位 7	GPIOFRST	0x0	rw	GPIOF 复位 (GPIOF reset) 0: 无复位; 1: 复位。
位 6	保留	0x0	resd	保持默认值。
位 5	GPIODRST	0x0	rw	GPIOD 复位 (GPIOD reset) 0: 无复位; 1: 复位。
位 4	GPIOCRST	0x0	rw	GPIOC 复位 (GPIOC reset) 0: 无复位; 1: 复位。
位 3	GPIOBRST	0x0	rw	GPIOB 复位 (GPIOB reset) 0: 无复位; 1: 复位。
位 2	GPIOARST	0x0	rw	GPIOA 复位 (GPIOA reset) 0: 无复位; 1: 复位。
位 1	EXINTRST	0x0	rw	EXINT 复位 (EXINT reset) 0: 无复位; 1: 复位。保持默认值。
位 0	IOMUXRST	0x0	rw	IOMUX 复位 (IOMUX reset) 0: 无复位; 1: 复位。

### 4.3.5 APB1外设复位寄存器 (CRM\_APB1RST)

访问：无等待周期，字，半字和字节访问

域	简称	复位值	类型	功能
位 31:29	保留	0x0	resd	保持默认值。
位 28	PWCRST	0x0	rw	PWC 复位 (PWC reset) 0: 无复位; 1: 复位。
位 27: 26	保留	0x0	resd	保持默认值。
位 25	CAN1RST	0x0	rw	CAN1 复位 (CAN1 reset) 0: 无复位; 1: 复位。
位 24: 23	保留	0x0	resd	保持默认值。
位 22	保留	0x0	resd	保持默认值。
位 21	I2C1RST	0x0	rw	I2C1 复位 (I2C1 reset) 0: 无复位; 1: 复位。
位 20	UART5RST	0x0	rw	UART5 复位 (UART5 reset) 0: 无复位; 1: 复位。

位 19	保留	0x0	resd	保持默认值。
位 18	USART3RST	0x0	rw	USART3 复位 (USART3 reset) 0: 无复位; 1: 复位。
位 17	USART2RST	0x0	rw	USART2 复位 (USART2 reset) 0: 无复位; 1: 复位。
位 16: 15	保留	0x0	resd	保持默认值。
位 14	SPI2RST	0x0	rw	SPI2 复位 (SPI2 reset) 0: 无复位; 1: 复位。
位 13: 12	保留	0x0	resd	保持默认值。
位 11	WWDTRST	0x0	rw	窗口看门狗复位 (WWDT reset) 0: 无复位; 1: 复位。
位 10	保留	0x0	resd	保持默认值。
位 9	CMPRST	0x0	rw	CMP 复位 (CMP reset) 0: 无复位; 1: 复位。
位 8: 4	保留	0x00	resd	保持默认值。
位 3	TMR5RST	0x0	rw	TMR5 复位 (TMR5 reset) 0: 无复位; 1: 复位。
位 2	TMR4RST	0x0	rw	TMR4 复位 (TMR4 reset) 0: 无复位; 1: 复位。
位 1	保留	0x0	resd	保持默认值。
位 0	TMR2RST	0x0	rw	TMR2 复位 (TMR2 reset) 0: 无复位; 1: 复位。

#### 4.3.6 AHB外设时钟使能寄存器 (CRM\_AHBEN)

访问：无等待周期，字、半字和字节访问

域	简称	复位值	类型	功能
位 31: 13	保留	0x00000	resd	保持默认值。
位 12	OTGFS1EN	0x0	rw	USB 时钟使能 (USB clock enable) 0: 关闭; 1: 开启。
位 11	保留	0x0	resd	保持默认值。
位 10	保留	0x0	resd	保持默认值。
位 9: 7	保留	0x0	rw	保持默认值。
位 6	CRCEN	0x0	rw	CRC 时钟使能 (CRC clock enable) 0: 关闭; 1: 开启。
位 5	保留	0x0	resd	保持默认值。

				闪存时钟使能 (Flash clock enable)
位 4	FLASHEN	0x1	rw	该位配置睡眠或深度睡眠模式下闪存时钟使能。 0: 关闭; 1: 开启。
位 3	保留	0x0	resd	保持默认值。
位 2	SRAMEN	0x1	rw	SRAM 时钟使能 (SRAM clock enable) 该位配置睡眠或深度睡眠模式下 SRAM 时钟使能。 0: 关闭; 1: 开启。
位 1	DMA2EN	0x0	rw	DMA2 时钟使能 (DMA2 clock enable) 0: 关闭; 1: 开启。
位 0	DMA1EN	0x0	rw	DMA1 时钟使能 (DMA1 clock enable) 0: 关闭; 1: 开启。

#### 4.3.7 APB2外设时钟使能寄存器 (CRM\_APB2EN)

域	简称	复位值	类型	功能
位 31: 22 保留		0x00	resd	保持默认值。
位 21	保留	0x0	resd	保持默认值。
位 20	TMR10EN	0x0	rw	TMR10 时钟使能 (TMR10 clock enable) 0: 关闭; 1: 开启。
位 19	TMR9EN	0x0	rw	TMR9 时钟使能 (TMR9 clock enable) 0: 关闭; 1: 开启。
位 18: 15 保留		0x0	resd	保持默认值。
位 14	USART1EN	0x0	rw	USART1 时钟使能 (USART1 clock enable) 0: 关闭; 1: 开启。
位 13	保留	0x0	resd	保持默认值。
位 12	保留	0x0	resd	保持默认值。
位 11	TMR1EN	0x0	rw	TMR1 时钟使能 (TMR1 clock enable) 0: 关闭; 1: 开启。
位 10	保留	0x0	resd	保持默认值。
位 9	ADC1EN	0x0	rw	ADC1 时钟使能 (ADC 1 clock enable) 0: 关闭; 1: 开启。
位 8	保留	0x0	resd	保持默认值。
位 7	GPIOFEN	0x0	rw	GPIOF 时钟使能 (GPIOF clock enable) 0: 关闭; 1: 开启。
位 6	保留	0x0	resd	保持默认值。

位 5	GPIODEN	0x0	rw	GPIOD 时钟使能 (GPIOD clock enable) 0: 关闭; 1: 开启。
位 4	GPIOCEN	0x0	rw	GPIOC 时钟使能 (GPIOC clock enable) 0: 关闭; 1: 开启。
位 3	GPIOBEN	0x0	rw	GPIOB 时钟使能 (GPIOB clock enable) 0: 关闭; 1: 开启。
位 2	GPIOAEN	0x0	rw	GPIOA 时钟使能 (GPIOA clock enable) 0: 关闭; 1: 开启。
位 1	保留	0x0	rw	保持默认值。
位 0	IOMUXEN	0x0	rw	IOMUX 时钟使能 (IOMUX clock enable) 0: 关闭; 1: 开启。

### 4.3.8 APB1外设时钟使能寄存器 (CRM\_APB1EN)

访问：字、半字和字节访问

通常无访问等待周期。但在 APB1 总线上的外设被访问时，将插入等待状态直到 APB1 外设访问结束。

域	简称	复位值	类型	功能
位 31: 29 保留		0x0	resd	保持默认值。
位 28	PWCEN	0x0	rw	PWC 时钟使能 (Power control clock enable) 0: 关闭; 1: 开启。
位 27	保留	0x0	resd	保持默认值。
位 26	保留	0x0	resd	保持默认值。
位 25	CAN1EN	0x0	rw	CAN1 时钟使能 (CAN1 clock enable) 0: 关闭; 1: 开启。
位 24: 23 保留		0x0	resd	保持默认值。
位 22	保留	0x0	resd	保持默认值。
位 21	I2C1EN	0x0	rw	I2C1 时钟使能 (I2C1 clock enable) 0: 关闭; 1: 开启。
位 20	UART5EN	0x0	rw	UART5 时钟使能 (UART5 clock enable) 0: 关闭; 1: 开启。
位 19	保留	0x0	resd	保持默认值。
位 18	USART3EN	0x0	rw	USART3 时钟使能 (USART3 clock enable) 0: 关闭; 1: 开启。

位 17	USART2EN	0x0	rw	USART2 时钟使能 (USART2 clock enable) 0: 关闭; 1: 开启。
位 16: 15 保留		0x0	resd	保持默认值。
位 14	SPI2EN	0x0	rw	SPI2 时钟使能 (SPI2 clock enable) 0: 关闭; 1: 开启。
位 13: 12 保留		0x0	resd	保持默认值。
位 11	WWDTEN	0x0	rw	窗口看门狗时钟使能 (WWDT clock enable) 0: 关闭; 1: 开启。
位 10	保留	0x0	resd	保持默认值。
位 9	CMPEN	0x0	rw	CMP 时钟使能 (CMP clock enable) 0: 关闭; 1: 开启。
位 8: 4 保留		0x00	resd	保持默认值。
位 3	TMR5EN	0x0	rw	TMR5 时钟使能 (TMR5 clock enable) 0: 关闭; 1: 开启。
位 2	TMR4EN	0x0	rw	TMR4 时钟使能 (TMR4 clock enable) 0: 关闭; 1: 开启。
位 1	保留	0x0	resd	保持默认值。
位 0	TMR2EN	0x0	rw	TMR2 时钟使能 (TMR2 clock enable) 0: 关闭; 1: 开启。

#### 4.3.9 电池供电域控制寄存器 (CRM\_BPDC)

访问：0 到 3 等待周期，字、半字和字节访问；当连续对该寄存器进行访问时，将插入等待状态。

注意：电池供电域控制寄存器中 (CRM\_BPDC) LEXTEN、LEXTBYPs、ERTCSEL 和 ERTCEN 位处于电池供电域。因此，这些位在复位后处于写保护状态，只有在电源控制寄存器 (PWC\_CTRL) 中的 BPWEN 位置位后才能对这些位进行改动。这些位只能由电池供电域软件复位清除。任何内部或外部复位都不会影响这些位。

域	简称	复位值	类型	功能
位 31: 17 保留		0x0000	resd	保持默认值。
位 16	BPDRST	0x0	rw	电池供电域软件复位 (Battery powered domain software reset) 0: 无复位; 1: 复位。
位 15	ERTCEN	0x0	rw	ERTC 时钟使能 (ERTC clock enable) 由软件置位或清零。 0: 关闭; 1: 开启。
位 14: 10 保留		0x00	resd	保持默认值。

				ERTC 时钟选择 (ERTC clock selection) 确定了 ERTC 时钟选择后, 如果想要再次更改, 必须设置 BPDRST 位复位后, 才能重新改写 ERTC 时钟选择。
位 9: 8	ERTCSEL	0x0	rw	00: 无; 01: LEXT; 10: LICK; 11: HEXT/128。
位 7: 3	保留	0x0	resd	保持默认值。
位 2	LEXTBYP	0x0	rw	LEXT 旁路使能 (Low speed external crystal bypass) 0: 关闭; 1: 开启。
位 1	LEXTSTBL	0x0	ro	LEXT 稳定 (External low-speed oscillator stable) 该位待 LEXT 稳定后由硬件置起。 0: 未稳定; 1: 已稳定。
位 0	LEXTEN	0x0	rw	LEXT 使能 (External low-speed oscillator enable) 0: 关闭; 1: 开启。

#### 4.3.10 控制/状态寄存器 (CRM\_CTRLSTS)

除复位标志外由系统复位清除, 复位标志能由电源复位或写 RSTFC 位进行清除。访问: 0 到 3 等待周期, 字、半字和字节访问; 当连续对该寄存器进行访问时, 将插入等待状态。

域	简称	复位值	类型	功能
位 31	LPRSTF	0x0	ro	低功耗复位标志 (Low-power reset flag) 该位由硬件置起, 软件写 RSTFC 位清除。 0: 无; 1: 有。
位 30	WWDTRSTF	0x0	ro	窗口看门狗复位标志 (WWDT reset flag) 该位由硬件置起, 软件写 RSTFC 位清除。 0: 无; 1: 有。
位 29	WDTRSTF	0x0	ro	看门狗复位标志 (WDT reset flag) 该位由硬件置起, 软件写 RSTFC 位清除。 0: 无; 1: 有。
位 28	SWRSTF	0x0	ro	软件复位标志 (Software reset flag) 该位由硬件置起, 软件写 RSTFC 位清除。 0: 无; 1: 有。
位 27	PORRSTF	0x1	ro	上电/掉电复位标志 (POR/LVR reset flag) 该位由硬件置起, 软件写 RSTFC 位清除。 0: 无; 1: 有。
位 26	NRSTF	0x1	ro	NRST 引脚复位标志 (NRST reset flag) 该位由硬件置起, 软件写 RSTFC 位清除。 0: 无; 1: 有。
位 25	保留	0x0	resd	保持默认值。
位 24	RSTFC	0x0	rw	清除复位标志 (Reset flag clear) 由软件写'1'来清除复位标志。 0: 无作用; 1: 清除复位标志。
位 23: 2	保留	0x000000	resd	保持默认值。

位 1	LICKSTBL	0x0	ro	LICK 稳定 (LICK stable) 0: 未稳定; 1: 已稳定。
位 0	LICKEN	0x0	rw	LICK 使能 (LICK enable) 0: 关闭; 1: 开启。

### 4.3.11 AHB外设复位寄存器 (CRM\_AHBRST)

访问：无等待周期，字，半字和字节访问

域	简称	复位值	类型	功能
位 31: 13 保留		0x00000	resd	保持默认值。
位 12	OTGFS1RST	0x0	rw	USB 复位 (USB reset) 0: 无作用; 1: 复位。
位 11: 0 保留		0x000	resd	保持默认值。

### 4.3.12 PLL配置寄存器 (CRM\_PLL)

访问：无等待周期，字，半字和字节访问

域	简称	复位值	类型	功能
位 31	PLLCFGEN	0x0	rw	PLL 配置使能 (PLL Configure Enable) 0: PLL 使用常规整数倍频配置模式, 使用 PLL_FREF 和 PLLMUL 寄存器配置 PLL 1: PLL 使用灵活配置模式, 使用 PLL_MS/PLL_NS/PLL_FR 寄存器值配置 PLL。
位 30: 27 保留		0x0	resd	保持默认值。 <b>PLL_FREF:</b> PLL 输入时钟选择, 仅在 PLLCFGEN=0 时起作用 000: PLL 使用 3.9 MHz ~ 5 MHz 输入时钟; 001: PLL 使用 5.2 MHz ~ 6.25 MHz 输入时钟; 010: PLL 使用 7.8125 MHz ~ 8.33 MHz 输入时钟; 011: PLL 使用 8.33 MHz ~ 12.5 MHz 输入时钟; 100: PLL 使用 15.625 MHz ~ 20.83 MHz 输入时钟; 101: PLL 使用 20.83 MHz ~ 31.255 MHz 输入时钟; 110: 保留; 111: 保留。
位 26: 24		0x0	rw	
位 23: 17 保留		0x00	resd	保持默认值。
位 16: 8 PLL_NS		0x01F	rw	PLL 倍频系数 PLL_NS 范围 (31~500)
位 7: 4 PLL_MS		0x1	rw	PLL 预分频系数 PLL_MS 范围 (1~15)
位 3 保留		0x0	resd	保持默认值。
位 2: 0 PLL_FR		0x0	rw	PLL 后分频配置值 PLL_FR 范围 (0~5) 000: PLL 后分频系数为 1, 1 分频输出; 001: PLL 后分频系数为 2, 2 分频输出; 010: PLL 后分频系数为 4, 4 分频输出; 011: PLL 后分频系数为 8, 8 分频输出; 100: PLL 后分频系数为 16, 16 分频输出; 101: PLL 后分频系数为 32, 32 分频输出; 其他: 保留 请注意 PLL_FR 值和后分频系数对应关系

### 4.3.13 额外寄存器 (CRM\_MISC1)

域	简称	复位值	类型	功能
位 31: 28	CLKOUTDIV	0x0	rw	CLKOUT 分频因子 (Clock output division) CLKOUT 输出频率的分频值设定。 0xxx: 不分频; 1000: 2 分频; 1001: 4 分频; 1010: 8 分频; 1011: 16 分频; 1100: 64 分频; 1101: 128 分频; 1110: 256 分频; 1111: 512 分频。
位 27: 26	保留	0x0	resd	保持默认值。
位 25	HICKDIV	0x0	rw	HICK 6 分频 (HICK 6 divider selection) 该位选择使用 HICK 时钟还是 HICK 的 6 分频时钟，选择 HICK 的 6 分频时钟的话，时钟频率为 8 MHz，选择不分频的话，时钟频率为 48 MHz。 0: 分频; 1: 不分频。 注意：不论何种情况 HICK 输入到 PLL 时的频率都固定为 4 MHz。
位 24: 21	保留	0x0	resd	保持默认值。
位 20	CLKFMC_SRC	0x0	rw	FMC 时钟来源 (FMC Clock Source) 0: FMC 时钟源是 8M HICK; 1: FMC 时钟源是 HCLK。
位 19: 17	保留	0x0	resd	保持默认值。
位 16	CLKOUT_SEL[3]	0x0	rw	内部时钟输出选择 (Clock output selection) 搭配时钟配置寄存器 (CRM_CFG) 位 26: 24 使用。
位 15: 8	保留	0x00	resd	保持默认值。
位 7: 0	HICKCAL_KEY	0x00	rw	HICKCAL 写入键值 (HICK calibration key) 此字段为 0x5A 时，HICKCAL [7: 0]才可被写入。

### 4.3.14 OTG\_FS扩展控制寄存器 (CRM\_OTG\_EXTCTRL)

此应用程序必需在使能 OTG\_FS 外设之前，配置此寄存器。

域	简称	复位值	类型	功能
位 31	EP3_RMPEN	0x0	rw	端点 3 重配置使能 (endpoint3 remap enable) 0: OTG_FS 模块的端点 3 不支持重配置，仅作为端点 3 与主机通信； 1: OTG_FS 模块的端点 3 支持重配置，可同时支持作为端点 3 和端点 4 与主机通信。
位 30	USBDIV_RST	0x0	rw	USB 分频器复位 (USB Divider Reset) 0: 无作用 1: 复位 USB 分频器
位 29: 0	保留	0x0000 0000	resd	保持默认值。

注：此控制寄存器为 C 版增加功能

### 4.3.15 额外寄存器 (CRM\_MISC2)

域	简称	复位值	类型	功能
位 31: 10	保留	0x0000000	resd	保持默认值。

位 9	HICK_TO_SCLK	0x0	rw	HICK 作为系统时钟的频率选择位 当 SCLKSEL 选择 HICK 为时钟源时，SCLK 的频率为 0: 固定是 8Mhz, 即选择 HICK 时钟的 6 分频； 1: 根据 HICKDIV 设定可能为 48M 或 8M。
位 8: 6	保留	0x0	resd	保持默认值。
位 5: 4	AUTO_STEP_EN	0x0	rw	自动滑顺频率切换使能 为使切换系统时钟源到 PLL 或是切换 AHB 预分频由大到小时平顺(系统频率由小变大)，建议系统频率操作目标大于 108MHz 时启动自动滑顺频率切换。 当自动滑顺频率切换功能作用时，硬件会暂停 AHB 总线，直到整个自动滑顺频率切换完成才恢复。此期间 DMA 仍正常工作，中断事件会被记忆并待 AHB 总线恢复后 NVIC 即可处理。 00: 关闭自动滑顺频率切换； 01: 保留； 10: 保留； 11: 开启自动滑顺频率切换，当 AHBDIV 或 SCLKSEL 两个寄存器被改动时，会自动触发自动滑顺频率切换功能。
位 3: 0	保留	0x0	resd	保持默认值。

# 5 内嵌闪存控制器 (FLASH)

## 5.1 FLASH介绍

闪存由主存储器、信息块、闪存寄存器这三个部分组成。

- 主存储器容量可达 256K 字节
- 信息块由 18K 字节的系统启动程序代码区和用户系统数据区组成。系统启动程序使用 USART1, USART2 或 OTGFS device 模式实现 ISP 编程

主存储器只有闪存容量为 256K 字节的片 1 闪存，包含 128 扇区，每扇区大小为 2K 字节。

表 5-1 闪存存储结构 (256K)

结构		名称	地址范围	
主存储器	片 1 (Bank1) 256KB	扇区 0	0x0800 0000 – 0x0800 07FF	
		扇区 1	0x0800 0800 – 0x0800 0FFF	
		扇区 2	0x0800 1000 – 0x0800 17FF	
		...	...	
		扇区 127	0x0803 F800 – 0x0803 FFFF	
信息块		启动程序代码区 18KB	0x1FFF AC00 – 0x1FFF F3FF	
信息块		用户系统数据区 1KB	0x1FFF F800 – 0x1FFF FBFF	

### 用户系统数据区

每次系统复位后将从闪存信息块中读出系统数据信息并保存在用户系统数据寄存器 (FLASH\_USD) 以及擦除编程保护状态寄存器 (FLASH\_EPPS) 中。

每个系统数据实际占用 2 个字节，低字节对应系统数据的内容，高字节对应系统数据的反码，用于验证选择位的正确性。当读出的高字节不等于低字节的反码时（高字节及低字节均为 0xFF 时除外），系统数据装载器会产生一个系统数据错误的标志 (USDERR)，并把对应的系统数据及其反码都强置为 0xFF。

注意：用户系统数据内容的更新需要一次系统复位才能真正实现。

表 5-2 用户系统数据说明

地址	位	内容
0x1FFF_F800	[7:0]	<b>FAP[7:0]:</b> 闪存访问保护 (访问保护启动/解除结果存放在寄存器 FLASH_USD[1] 以及 [26]) 0xA5: 闪存访问保护解除 0xCC: 高级闪存访问保护启动 其他值: 低级闪存访问保护启动
	[15:8]	<b>nFAP[7:0]:</b> FAP[7:0] 的反码
	[23:16]	<b>SSB[7:0]:</b> 系统配置字节 (存放在寄存器 FLASH_USD[9:2]) <b>位 7:3</b> 保留不用 <b>位 2 (nSTDBY_RST)</b> 0: 进入待机模式时产生复位 1: 进入待机模式时不产生复位 <b>位 1 (nDEPSLP_RST)</b> 0: 进入深度睡眠模式时产生复位 1: 进入深度睡眠模式时不产生复位 <b>位 0 (nWDT_ATO_EN)</b> 0: 看门狗自启动开启 1: 看门狗自启动关闭
	[31:24]	<b>nSSB[7:0]:</b> SSB[7:0] 的反码
	[7:0]	<b>Data0[7:0]:</b> 用户数据 0 (存放在寄存器 FLASH_USD[17:10])
	[15:8]	<b>nData0[7:0]:</b> Data0[7:0] 的反码
	[23:16]	<b>Data1[7:0]:</b> 用户数据 1 (存放在寄存器 FLASH_USD[25:18])
	[31:24]	<b>nData1[7:0]:</b> Data1[7:0] 的反码
	[7:0]	<b>EPP0[7:0]:</b> 闪存擦写保护字节 0 (存放在寄存器 FLASH_EPPS[7:0]) 用于保护主闪存存储器的扇区 0 ~ 扇区 15，每个比特位保护 2 个扇区 0: 擦写保护启动 1: 擦写保护解除
	[15:8]	<b>nEPP0[7:0]:</b> EPP0[7:0] 的反码
0x1FFF_F808	[23:16]	<b>EPP1[7:0]:</b> 闪存擦写保护字节 1 (存放在寄存器 FLASH_EPPS[15:8]) 用于保护主闪存存储器的扇区 16 ~ 扇区 31，每个比特位保护 2 个扇区 0: 擦写保护启动

		1: 擦写保护解除 <b>nEPP1[7:0]: EPP1[7:0]的反码</b>
0x1FFF_F80C	[7:0]	<b>EPP2[7:0]:</b> 闪存擦写保护字节 2 (存放在寄存器 FLASH_EPPS[23:16]) 用于保护主闪存存储器的扇区 32 ~ 扇区 47, 每个比特位保护 2 个扇区 0: 擦写保护启动 1: 擦写保护解除
	[15:8]	<b>nEPP2[7:0]:</b> EPP2[7:0]的反码
	[23:16]	<b>EPP3[7:0]:</b> 闪存擦写保护字节 3 (存放在寄存器 FLASH_EPPS[31:24]) 其中位 6:0 用于保护主闪存存储器的扇区 48 ~ 扇区 61, 每个比特位保护 2 个扇区 位 7 用于保护主闪存存储器的扇区 62 及之后的扇区, 以及主存扩展区 0: 擦写保护启动 1: 擦写保护解除
	[31:24]	<b>nEPP3[7:0]:</b> EPP3[7:0]的反码
	[7:0]	<b>Data2[7:0]:</b> 用户数据 2
	[15:8]	<b>nData2[7:0]:</b> Data2[7:0]的反码
0x1FFF_F810	[23:16]	<b>Data3[7:0]:</b> 用户数据 3
	[31:24]	<b>nData3[7:0]:</b> Data3[7:0]的反码
	[7:0]	<b>Data4[7:0]:</b> 用户数据 4
	[15:8]	<b>nData4[7:0]:</b> Data4[7:0]的反码
0x1FFF_F814	[23:16]	<b>Data5[7:0]:</b> 用户数据 5
	[31:24]	<b>nData5[7:0]:</b> Data5[7:0]的反码
	.	.
	[7:0]	<b>Data504[7:0]:</b> 用户数据 504
0x1FFF_FBFC	[15:8]	<b>nData504[7:0]:</b> Data504[7:0]的反码
	[23:16]	<b>Data505[7:0]:</b> 用户数据 505
	[31:24]	<b>nData505[7:0]:</b> Data505[7:0]的反码

## 5.2 主存储器操作

### 5.2.1 解锁/锁定

复位后, 主存储器默认是被锁定的, 此时不允许配置闪存控制寄存器 (FLASH\_CTRL), 需要对闪存解锁后才能成功实现对闪存的写入与擦除操作。

#### 解锁流程:

对闪存解锁寄存器(FLASH\_UNLOCK)顺序写入键值 KEY1(0x45670123)和键值 KEY2(0xCDEF89AB), 能够解锁对应区域闪存。

注意: 解锁必须顺序写入正确的键值, 否则会产生总线错误并且闪存会被锁死, 直到下一次复位才能恢复。

#### 锁定流程:

软件置起闪存控制寄存器 (FLASH\_CTRL) 中的 OPLK 位, 锁定对应区域闪存。

### 5.2.2 擦除

编程之前必须先进行擦除操作, 主存储器有扇区擦除和整片擦除两种擦除方式。

#### 扇区擦除

主闪存存储器的每一扇区以及主存扩展区都可以使用扇区擦除功能独立擦除。

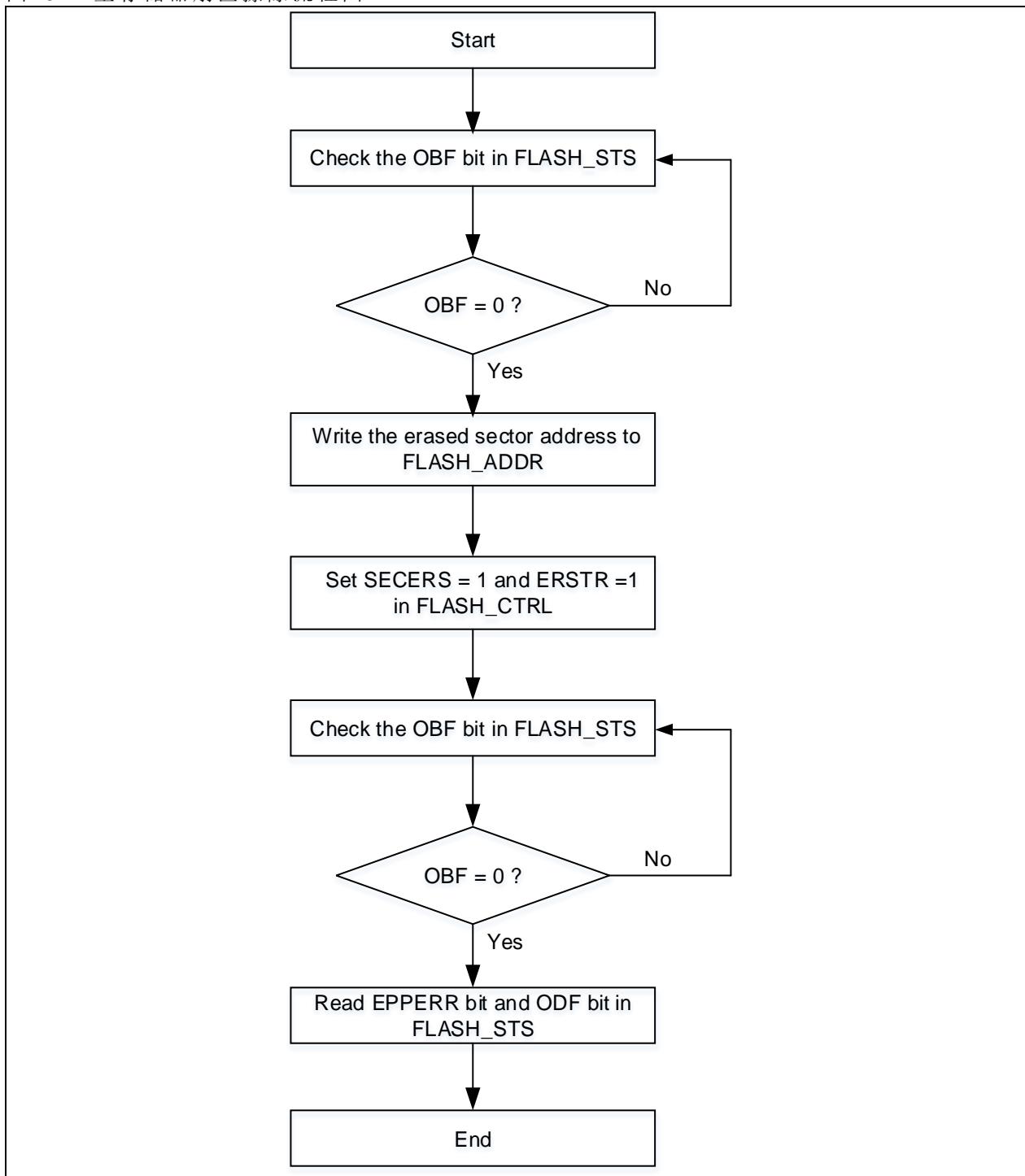
擦除流程如下:

- 检查闪存状态寄存器 (FLASH\_STS) 的 OBF 位, 确认没有正在进行的闪存操作;
- 对闪存地址寄存器 (FLASH\_ADDR) 写入要擦除的扇区地址;
- 对闪存控制寄存器 (FLASH\_CTRL) 的 SECERS 位以及 ERSTR 位均置 1, 启动扇区擦除;
- 等待闪存状态寄存器 (FLASH\_STS) 的 OBF 位变为‘0’, 并查询闪存状态寄存器 (FLASH\_STS) 的 EPPERR 位和 ODF 位, 确认擦除结果。

注意: 当启动程序代码区域是设定为主存扩展区时, 执行扇区擦除实际上是对整个主存扩

展区的擦除。

图 5-1 主存储器扇区擦除流程图



### 整片擦除

主闪存存储器可以使用整片擦除功能直接擦除。

擦除流程如下：

- 检查 **FLASH\_STS** 寄存器的 **OBF** 位，确认没有正在进行的闪存操作；
- 对闪存控制寄存器（**FLASH\_CTRL**）的 **BANKERS** 位以及 **ERSTR** 位均置 1，启动整片擦除；
- 等待 **FLASH\_STS** 寄存器的 **OBF** 位变为‘0’，并查询 **FLASH\_STS** 寄存器的 **EPPERR** 位和 **ODF** 位，确认擦除结果。

注意：

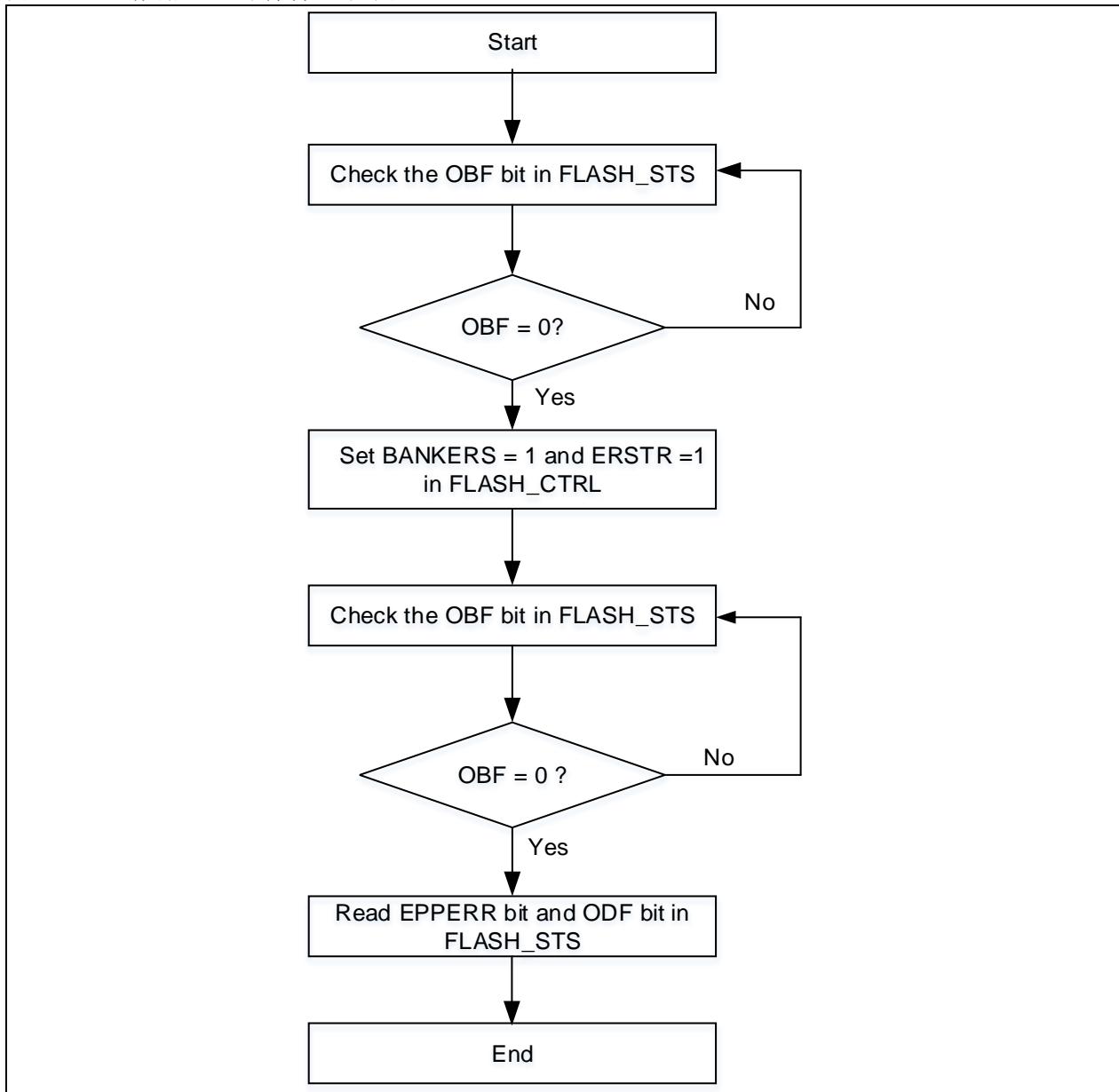
- 1) 当启动程序代码区域是设定为主存扩展区时，执行整片擦除操作会自动擦除主闪存以

及主存扩展区。

2) 擦除期间进行读闪存的操作，将导致 CPU 会被暂停直到擦除完成才处理读闪存操作。

3) 擦除操作前必须保证内部的 HICK 有打开

图 5-2 主存储器整片擦除流程图



### 5.2.3 编程

当想要改写主存储器的内容时，可以通过主存储器编程流程完成一次写入 32 位、16 位或 8 位的数据。主存储器编程流程：

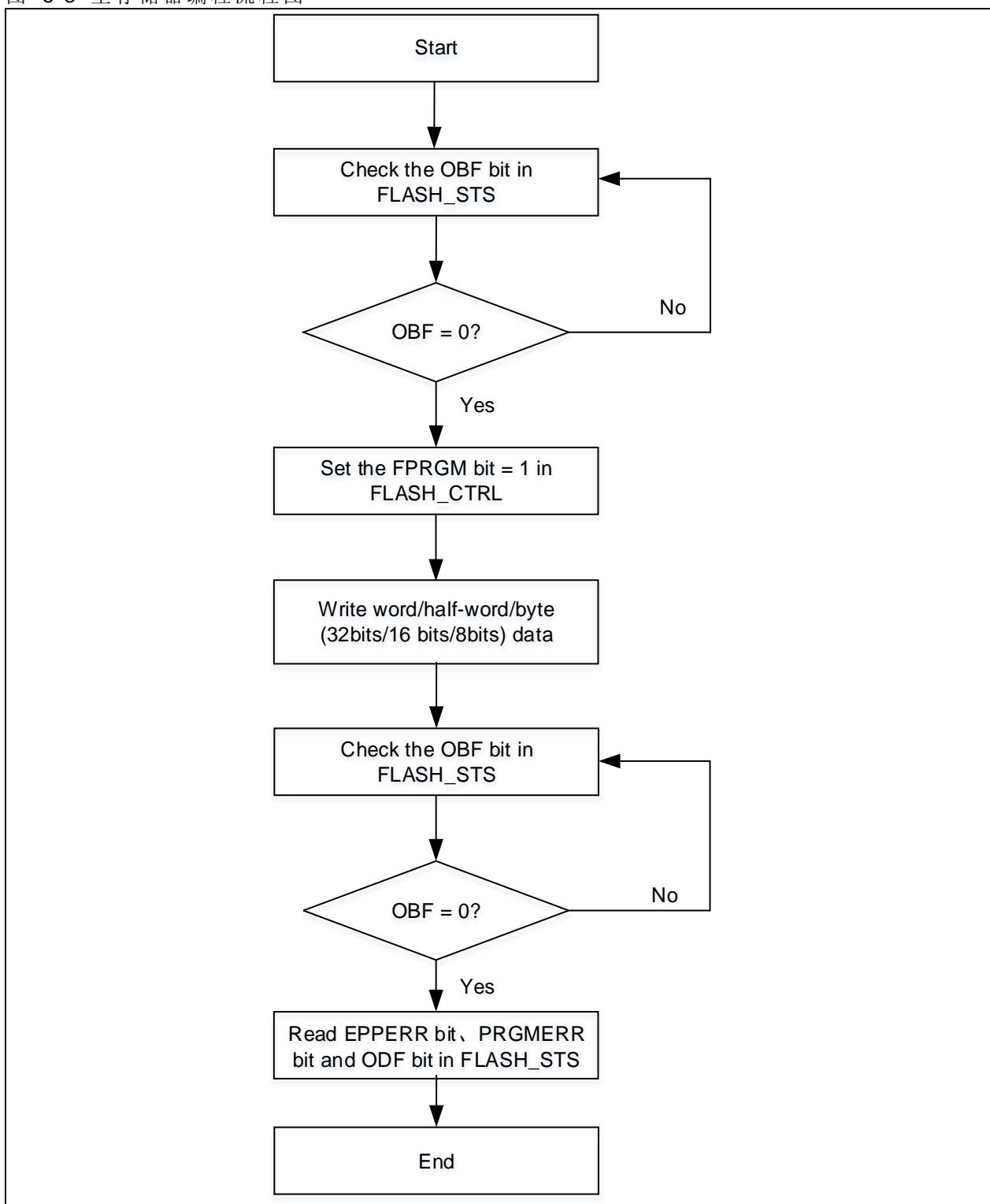
- 检查闪存状态寄存器（FLASH\_STS）的 OBF 位，确认没有正在进行的闪存操作；
- 对闪存控制寄存器（FLASH\_CTRL）的 FPRGM 位置 1，此时可以接受对主闪存的编程指令；
- 对指定的地址写入要编程的数据（任意字/半字/字节）；
- 等待闪存状态寄存器（FLASH\_STS）的 OBF 位变为‘0’，并查询闪存状态寄存器（FLASH\_STS）的 EPPERR 位、PRGMERR 位和 ODF 位，确认编程结果。

注意：

- 1) 当要写入的地址未被提前擦除时，除非要写入的数据值是全 0，否则编程不被执行，并置位闪存状态寄存器（FLASH\_STS）的 PRGMERR 位来告知编程发生错误。
- 2) 编程期间进行读闪存的操作，将导致 CPU 会被暂停直到编程完成才处理读闪存操作。

3) 编程操作前必须保证内部的 HICK 有打开。

图 5-3 主存储器编程流程图



#### 5.2.4 读取

通过 CPU 的 AHB 总线可以直接寻址访问主闪存存储区。

### 5.3 主存扩展区操作

启动程序代码区也可以设定为主存扩展区存放用户应用代码。当作为主存扩展区时，其操作方法，包括读取、解锁、擦除、编程都跟主存储器相同。

## 5.4 用户系统数据区操作

### 5.4.1 解锁/锁定

复位后，用户系统数据区默认是锁定的，需要在闪存解锁后再对用户系统数据区解锁才能成功实现写入与擦除操作。

#### 解锁流程：

对闪存解锁寄存器(FLASH\_UNLOCK)顺序写入键值 KEY1(0x45670123)和键值 KEY2(0xCDEF89AB);对闪存用户系统数据解锁寄存器 (FLASH\_USD\_UNLOCK) 顺序写入键值 KEY1 (0x45670123) 和键值 KEY2 (0xCDEF89AB)，闪存控制寄存器 (FLASH\_CTRL) 中的 USDULKS 位将被硬件自动置起，表示允许对用户系统数据区的写、擦除操作。

**注意：**解锁必须顺序写入正确的键值，否则会产生总线错误并且闪存会被锁死，直到下一次复位才能恢复。

#### 锁定流程：

软件清除闪存控制寄存器 (FLASH\_CTRL) 中的 USDULKS 位，锁定用户系统数据区。

### 5.4.2 擦除

在编程之前必须先进行擦除操作，用户系统数据区域可单独实现擦除功能。

#### 擦除流程如下：

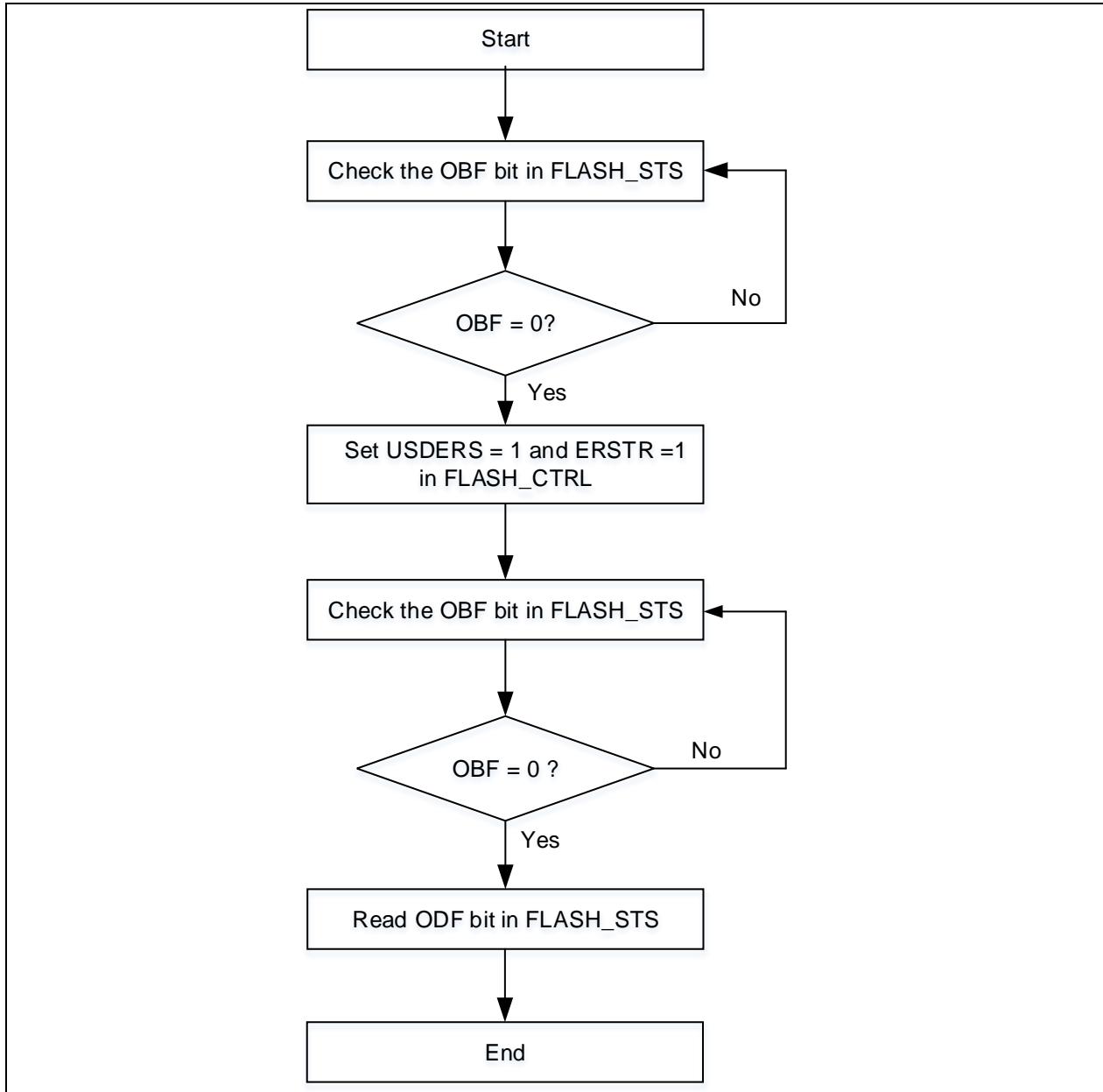
- 检查闪存状态寄存器 (FLASH\_STS) 的 OBF 位，确认没有正在进行的闪存操作；
- 对闪存控制寄存器 (FLASH\_CTRL) 的 USDERS 位以及 ERSTR 位均置 1，启动整块系统数据区擦除；
- 等待闪存状态寄存器 (FLASH\_STS) 的 OBF 位变为‘0’，并查询闪存状态寄存器 (FLASH\_STS) 的 ODF 位，确认擦除结果。

#### 注意：

擦除期间进行读闪存的操作，将导致 CPU 会被暂停直到擦除完成才处理读闪存操作。

擦除操作前必须保证内部的 HICK 有打开

图 5-4 系统数据区擦除图



### 5.4.3 编程

当想要改写用户系统数据区域的内容时，可以通过用户系统数据区编程流程完成一次写入 32 位或 16 位数据。

系统数据区的编程流程：

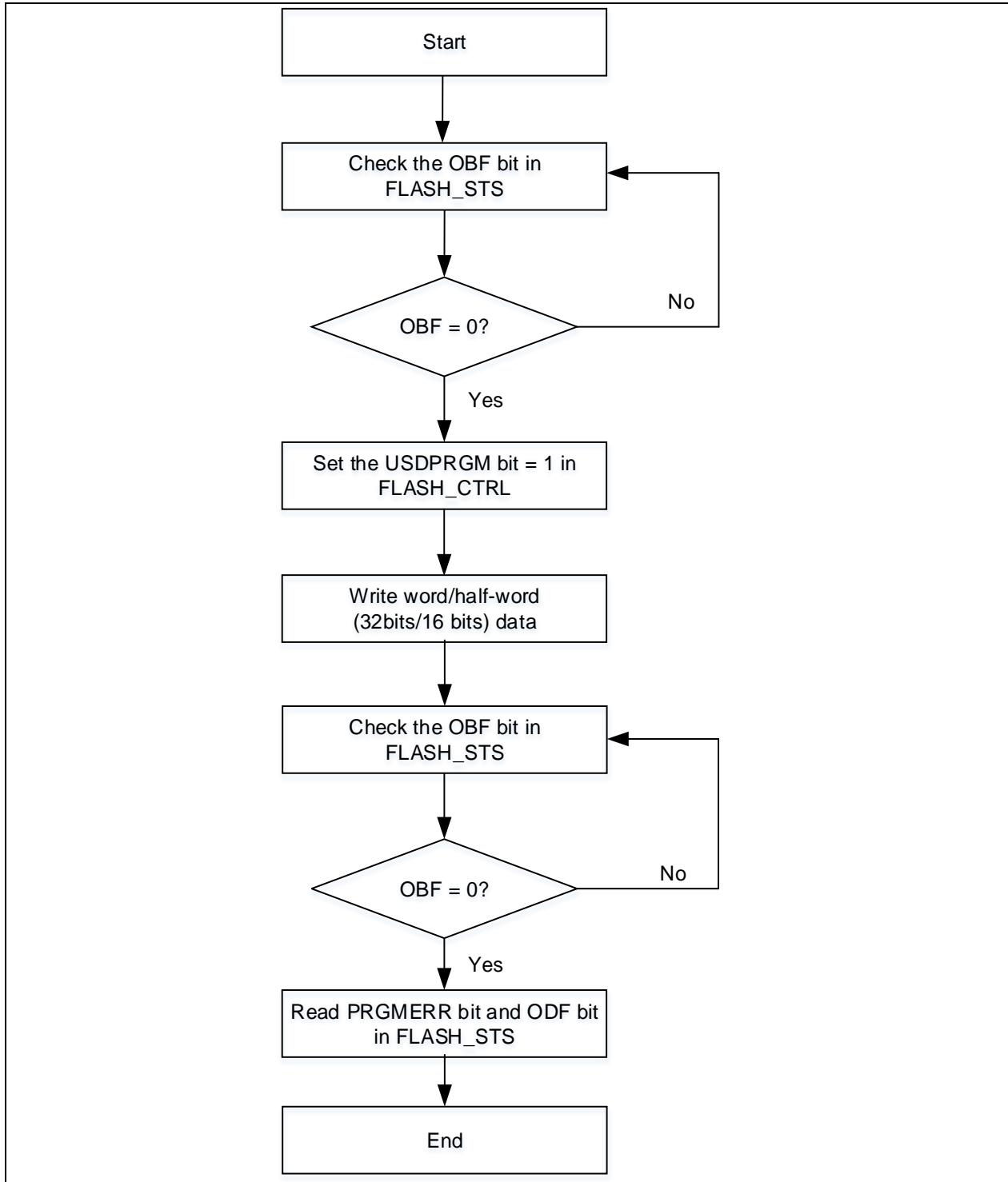
- 检查闪存状态寄存器（FLASH\_STS）的 OBF 位，确认没有正在进行的闪存操作；
- 对闪存控制寄存器（FLASH\_CTRL）的 USDPRGM 位置 1，此时可以接受对用户系统数据区的编程指令；
- 对指定的地址写入要编程的数据（任意字/半字）；
- 等待闪存状态寄存器（FLASH\_STS）的 OBF 位变为‘0’，并查询闪存状态寄存器（FLASH\_STS）的 PRGMERR 位和 ODF 位，确认编程结果。

注意：

编程期间进行读闪存的操作，将导致 CPU 会被暂停直到编程完成才处理读闪存操作。

编程操作前必须保证内部的 HICK 有打开

图 5-5 系统数据区编程图



#### 5.4.4 读取

通过 CPU 的 AHB 总线可以直接寻址访问用户系统数据区。

### 5.5 闪存保护

闪存存储器有访问保护以及擦写保护两种保护方式。

#### 5.5.1 访问保护

闪存访问保护分为两类：闪存低级访问保护，闪存高级访问保护。

访问保护启动后，只允许闪存程序对闪存存储器数据进行读出访问，禁止在调试模式下或是从非主闪存存

储器启动对闪存存储器数据的读出访问。

#### 闪存低级访问保护

当 nFAP 字节和 FAP 字节存放的内容不等于 0x5A 和 0xA5 以及不等于 0x33 和 0xCC 时，闪存在系统复位后，将启动闪存低级访问保护。

此保护下，用户可以重新擦除系统数据区，并对 FAP 字节写入 0xA5 解除闪存低级访问保护（从低级保护状态变为未保护状态，将自动产生对主闪存以及主存扩展区的整片擦除操作），最后进行系统复位，系统数据装载器重新加载系统数据信息，更新闪存访问保护解除信息（FAP 字节）。

#### 闪存高级访问保护

当 nFAP 字节存放的内容等于 0x33，并且 FAP 字节存放的内容等于 0xCC 时，闪存在系统复位后，将启动闪存高级访问保护。

一旦此保护启动后，将禁止用户重新擦除以及写入系统数据区。该保护的解除只能通过设置寄存器 FAP\_HL\_DIS 位来实现硬件自动擦除用户系统数据区。

**注意：**

- 1) 主存扩展区也支持访问保护功能
- 2) 如果访问保护被置位的时候仍然处于调试模式，必须用 POR（上电复位）代替系统复位 清除调试模式，才能恢复闪存程序访问闪存存储器数据的权限。

下表是启动闪存访问保护后，闪存不同区域访问权限说明：

表 5-3 闪存访问权限

区域		保护等级			访问权限		
		调试模式或是从 SRAM 启动以及从启动程序代码区启动			从主闪存启动		
		读	写	擦除	读	写	擦除
主闪存区	低级访问保护	禁止		禁止 (1) (2)	允许		
	高级访问保护 (4)	禁止		允许			
用户系统数据区	低级访问保护	禁止	允许		允许		
	高级访问保护 (4)	禁止	禁止	禁止 (3)	禁止	禁止 (3)	

(1) 主闪存区会在解除闪存访问保护时被硬件自动擦除

(2) 只禁止扇区擦除，允许整片擦除

(3) 用户系统数据区只能通过设置寄存器 FAP\_HL\_DIS 位被硬件自动擦除

(4) 闪存高级访问保护的功能是实现闪存访问及用户系统数据区误擦除保护

## 5.5.2 擦写保护

闪存的擦写保护的基本单位为 2 扇区。通过擦写保护可以防止程序在跑飞时闪存存储器的内容被意外更改。

在下面列出的情况下，擦写将不被允许，并会置位 EPPERR 位：

- 对被设置为擦写保护的扇区（主闪存以及闪存扩展区）做扇区擦除操作以及编程操作将不被允许
- 对存在任一扇区被设置为擦写保护的主闪存以及闪存扩展区做整片擦除将不被允许
- 闪存访问保护启动后，主闪存扇区 0~1 将被自动擦写保护，不允许做扇区擦除操作以及编程操作
- 闪存访问保护启动后，主存储器和主存扩展区在调试模式或是从非主闪存存储器启动下被自动擦写保护，不允许做扇区擦除操作以及编程操作

注意：在 256K 的闪存容量中，扇区单位是 2K 字节；在 256K 以下的闪存容量中，扇区单位是 1K 字节

## 5.6 特殊功能

### 5.6.1 安全库区设定

设定以密码保护主存中指定范围的程式区，即安全库区，此区域仅能被执行，无法读取（I-Code, D-Code 总线除外），以及写入与删除，除非输入指定密码。安全库区划分为 指令安全库区 与 数据安全库区，并可选部分或是整个安全库区存放指令，但不支持整个安全库区存放数据。

#### 设定安全库区的益处：

以密码保护安全库区，方案商可刻录核心算法到此区域；

安全库区仅能执行，无法被读取，除非输入方案商指定密码，也无法删除（包含 ISP/IAP/SWD）；

其余空白程序区可以提供给方案商客户进行二次开发；

方案商可以藉由安全库功能销售核心算法，不需要每个客户都开发完整方案；

设定安全库区，可防止蓄意破坏或更改终端产品应用程序代码。

#### 注意：

安全库区代码必须以扇区为单位进行烧录，并且起始地址与主存地址对齐；

中断向量表会被放置在闪存的第一扇扇区（扇区 0）内，请勿将闪存的第一扇扇区设定为安全库区；

要被安全库区保护的程序代码，不可放置在闪存的第一扇区内；

仅允许 I-Code 总线读取指令安全库区；

仅允许 D-Code 总线读取数据安全库区；

写入或删除安全库区代码，将在闪存状态寄存器（FLASH\_STS）的 EPPERR 位置‘1’提出警告；

执行主存的整片擦除时，将不会擦除安全库区。

默认状态下，安全库区设定寄存器始终是不可读且被锁定的。要想对安全库区设定寄存器进行写操作，首先要对安全库区解锁，对闪存安全库区解锁寄存器（SLIB\_UNLOCK）写入 0xA35F6D24 值，通过查看闪存安全库区密码清除寄存器（SLIB\_MISC\_STS）的位 SLIB\_ULKF 确认解锁成功，随后对安全库区设定寄存器写入设定值。

以扇区为单位对安全库区代码进行可选的 CRC 校验。

启动主存安全库区的流程如下：

- 检查闪存状态寄存器（FLASH\_STS）的 OBF 位，以确认没有其他正在进行的编程操作；
- 对闪存安全库区解锁寄存器（SLIB\_UNLOCK）写入 0xA35F6D24，以进行安全库区解锁；
- 检查闪存安全库区密码清除寄存器（SLIB\_MISC\_STS）的 SLIB\_ULKF 位，以确认解锁成功；
- 如果安全库区设在主闪存内，需要在闪存安全库区地址设定寄存器（SLIB\_SET\_RANGE）设定要保护的扇区，包含指令区与数据区的地址；如果安全库区设在主存扩展区域，需要设定主存扩展存储区域安全库区设定寄存器（EM\_SLIB\_SET）。
- 等待 OBF 位变为‘0’；
- 在闪存安全库区密码设定寄存器（SLIB\_SET\_PWD）设定安全库区密码；
- 等待 OBF 位变为‘0’；
- 烧录将存入安全库区的代码；
- 进行系统复位，重装载安全库区设定字；
- 读出闪存安全库区状态寄存器 0（SLIB\_STS0）/STS1 寄存器用于判断安全库区设定结果。

#### 注意：

不支持同时设定主闪存和主存扩展区域为安全库区；

启动安全库区的流程需要在闪存访问保护未启动时执行

解除安全库区的流程是：

- 在闪存安全库区密码清除寄存器（SLIB\_PWD\_CLR）写入先前设置的安全区域密码；

- 等待 OBF 位变为'0'；
- 进行系统复位，重装载安全库区设定字；
- 读出闪存安全库区状态寄存器 0 (SLIB\_STS0) 用于判断安全库区解除结果。
- 注意：解除安全库区将会自动执行主存及主存扩展区的整片擦除，以及安全库设定块擦除。

## 5.6.2 启动程序代码区域作为主存扩展使用

用户只有一次机会将启动程序代码区域作为主存扩展使用。一旦设定成功，主存扩展区将具有主闪存特性。设定启动程序代码区域作为主存扩展使用的流程是：

- 用户读取闪存安全库区状态寄存器 0 (SLIB\_STS0) 的位 0，获知启动程序代码区域当前的模式
- 对闪存安全库区解锁寄存器 (SLIB\_UNLOCK) 写入 0xA35F6D24，以进行启动程序代码区域模式设定解锁
- 写非 0xFF 到启动程序代码区模式设定寄存器 (BTM\_MODE\_SET) 的位 7-0
- 等待 OBF 位变为'0'；
- 进行系统复位，重装载设定字；
- 读出闪存安全库区状态寄存器 0 (SLIB\_STS0) 用于判断设定结果。

**注意：**启动设定主存扩展区的流程需要在闪存访问保护未启动时执行

## 5.6.3 CRC校验

以扇区为单位对安全库区代码或用户代码进行可选的 CRC 校验。

校验流程如下：

- 检查闪存状态寄存器 (FLASH\_STS) 的 OBF 位，以确认没有其他正在进行的编程操作；
- 在闪存 CRC 校验地址寄存器 (FLASH\_CRC\_ADDR) 设定要校验的代码起始地址；
- 在闪存 CRC 校验控制寄存器 (FLASH\_CRC\_CTR) 位 15-0，设定要校验的代码数量（单位是扇区）；
- 在闪存 CRC 校验控制寄存器 (FLASH\_CRC\_CTR) 设置位 16，启动 CRC 校验；
- 等待 OBF 位变为'0'；
- 读出闪存 CRC 校验结果寄存器 (FLASH\_CRC\_CHK) 用于判断 CRC 校验结果。

**注意：**闪存 CRC 校验地址寄存器 (FLASH\_CRC\_ADDR) 设定值必须与扇区起始地址对齐；不允许跨主存及主存扩展区的 CRC 校验。

## 5.7 FLASH寄存器

下表列出了 FLASH 寄存器的映像和复位值。

必须用字 (32 位) 的方式操作这些外设寄存器。

表 5-4 闪存接口寄存器映像和复位值

寄存器简称	基址偏移量	复位值
FLASH_PSR	0x00	0x0000 0030
FLASH_UNLOCK	0x04	0xFFFF XXXX
FLASH_USD_UNLOCK	0x08	0xFFFF XXXX
FLASH_STS	0x0C	0x0000 0000
FLASH_CTRL	0x10	0x0000 0080
FLASH_ADDR	0x14	0x0000 0000
FLASH_USD	0x1C	0x03FF FFFF
FLASH_EPPS	0x20	0xFFFF FFFF
SLIB_STS0	0x74	0x0000 0000
SLIB_STS1	0x78	0x0000 0000
SLIB_PWD_CLR	0x7C	0xFFFF FFFF

SLIB_MISC_STS	0x80	0x0000 0000
FLASH_CRC_ADDR	0x84	0x0000 0000
FLASH_CRC_CTRL	0x88	0x0000 0000
FLASH_CRC_CHK	0x8C	0x0000 0000
SLIB_SET_PWD	0x160	0x0000 0000
SLIB_SET_RANGE	0x164	0x0000 0000
EM_SLIB_SET	0x168	0x0000 0000
BTM_MODE_SET	0x16C	0x0000 0000
SLIB_UNLOCK	0x170	0x0000 0000

### 5.7.1 闪存性能选择寄存器 (FLASH\_PSR)

域	简称	复位值	类型	功能
位 31:6	保留	0x000000	resd	保持为默认值
位 5	PFT_ENF	0x1	ro	预取使能标志 (Prefetch enabled flag) 该位置起时，表示启动闪存预取缓冲区
位 4	PFT_EN	0x1	rw	预取使能 (Prefetch enable) 0: 闪存预取缓冲区关闭； 1: 闪存预取缓冲区开启。
位 3	HFCYC_EN	0x0	rw	半周期加速访问使能 (Half cycle acceleration access enable) 0: 关闭； 1: 开启。 用于 WTCYC=0 时加速对闪存的访问
位 2:0	WTCYC	0x0	rw	等待周期 (Wait cycle) 需要根据系统时钟大小来设定闪存访问的等待周期，以系统时钟为单位。 0: 零个等待周期 1: 一个等待周期 2: 两个等待周期 3: 三个等待周期 4: 四个等待周期 系统时钟以 32MHz 作为步阶： 系统时钟在第一个 32MHz 步阶可设零等待周期； 系统时钟在第二个 32MHz 步阶可设一个等待周期； 系统时钟在第三个 32MHz 步阶可设两个等待周期； 系统时钟在第四个 32MHz 步阶可设三个等待周期； 系统时钟在第五个 32MHz 步阶可设四个等待周期

### 5.7.2 闪存解锁寄存器 (FLASH\_UNLOCK)

域	简称	复位值	类型	功能
位 31:0	UKVAL	0XXXX XXXX wo		解锁键值 (Unlock key value) 该寄存器用于解锁主闪存及闪存扩展区。

注意：所有这些位是只写的，读出时返回 0。

### 5.7.3 闪存用户系统数据解锁寄存器 (FLASH\_USD\_UNLOCK)

域	简称	复位值	类型	功能
位 31:0	USD_UKVAL	0XXXX XXXX wo		用户系统数据解锁键值 (User system data Unlock key value)

注意：所有这些位是只写的，读出时返回 0。

### 5.7.4 闪存状态寄存器 (FLASH\_STS)

域	简称	复位值	类型	功能
位 31:6	保留	0x0000000	resd	保持为默认值
位 5	ODF	0	rw1c	操作完成标志 (Operation done flag) 当闪存操作 (编程/擦除) 成功完成时, 硬件会置起该位, 软件写'1'可以清除。
位 4	EPPERR	0	rw1c	擦写保护错误 (Erase/Program protection error) 当擦除或编程的闪存地址在擦写保护设定范围内时, 硬件会置起该位, 软件写'1'可以清除。
位 3	保留	0	resd	保持为默认值
位 2	PRGMERR	0	rw1c	编程错误 (Program error) 当编程的闪存地址的值为非擦除状态时, 硬件会置起该位, 软件写'1'可以清除。
位 1	保留	0	resd	保持为默认值
位 0	OBF	0	ro	操作忙标志 (Operation busy flag) 该位置起表示闪存操作正在进行, 该位清除表示操作结束。

### 5.7.5 闪存控制寄存器 (FLASH\_CTRL)

域	简称	复位值	类型	功能
位 31:17	保留	0x0000	resd	保持为默认值
位 16	FAP_HL_DIS	0x0	rw	闪存访问高级保护解除 (Flash access protection high level disable) 设置该位, 将触发硬件自动对用户系统数据区做擦除, 复位后将解除闪存访问高级保护, 维持低级访问保护。 该位置为 1 后, 硬件将自动清除此位。
位 15:13	保留	0x0	resd	保持为默认值
位 12	ODFIE	0	rw	操作完成中断使能 (Operation done flag interrupt enable) 0: 关闭; 1: 开启。
位 11,8,3	保留	0	resd	保持为默认值
位 10	ERRIE	0	rw	错误中断使能 (Error interrupt enable) 开启后 EPPERR 或 PRGMERR 都会产生中断。 0: 关闭; 1: 开启。
位 9	USDULKS	0	rw	用户系统数据解锁成功 (User system data unlock success) 一旦用户系统数据区解锁成功, 该位将被硬件自动置起, 表示允许对用户系统数据的编程/擦除操作。软件写'0'可以清除此位, 重新锁定用户系统数据区。
位 7	OPLK	1	rw	操作锁定 (Operation lock) 该位默认处于置起状态, 表示闪存锁定, 锁定时不允许操作, 解锁成功后, 硬件会自动清除此位, 表示允许闪存编程/擦除操作。软件写'1'可以重新锁定闪存操作。
位 6	ERSTR	0	rw	擦除开始 (Erasing start) 软件置起该位, 开始执行擦除操作。擦除完成后硬件自动清除该位。
位 5	USDERS	0	rw	用户系统数据擦除 (User system data erase) 用户系统数据区擦除。
位 4	USDPRGM	0	rw	用户系统数据编程 (User system data program) 用户系统数据编程。
位 2	BANKERS	0	rw	片擦除 (Bank erase) 擦除片操作。
位 1	SECERS	0	rw	扇区擦除 (Sector erase) 擦除扇区操作。
位 0	FPRGM	0	rw	闪存编程 (Flash program) 编程操作。

### 5.7.6 闪存地址寄存器 (FLASH\_ADDR)

域	简称	复位值	类型	功能
位 31:0	FA	0x0000 0000	wo	闪存地址 (Flash address) 扇区擦除时选择对应的闪存扇区地址。

### 5.7.7 用户系统数据寄存器 (FLASH\_USD)

域	简称	复位值	类型	功能
位 31:27	保留	0x00	resd	保持为默认值
位 26	FAP_HL	0	ro	闪存访问保护高级 (Flash access protection high level) 闪存访问保护状态使用 {位 26, 位 1}联合判断。 0: 未启动访问保护, 且 FAP 值=0xA5 1: 启动低级访问保护, 且 FAP 值非 0xCC 以及 0xA5 2: 保留 3: 启动高级访问保护, 且 FAP 值=0xCC
位 25:18	USER_D1	0xFF	ro	用户数据 1
位 17:10	USER_D0	0xFF	ro	用户数据 0
位 9:2	SSB	0xFF	ro	系统配置字节 (System setting byte) 这里包含加载的用户系统数据区中的系统配置字节 位 9:5: 未用 位 4: nSTDBY_RST 位 3: nDEPSLP_RST 位 2: nWDT_ATO_EN
位 1	FAP	0	ro	闪存访问保护 (Flash access protection)
位 0	USDERR	0	ro	用户系统数据错误 (User system data error) 该位置起表示用户系统数据中某字节和它的反码不匹配。 此时该字节和它的反码读出值将被硬件自动强制置为 0xFF

### 5.7.8 擦除编程保护状态寄存器 (FLASH\_EPPS)

域	简称	复位值	类型	功能
位 31:0	EPPS	0xFFFF FFFF	ro	擦除/编程保护状态 (Erase/program protection status) 该寄存器反映的是加载的用户系统数据中的擦写保护字节状态。

### 5.7.9 闪存安全库区状态寄存器0 (SLIB\_STS0)

专用于闪存安全库区。

域	简称	复位值	类型	功能
位 31:24	保留	0x00	resd	保持为默认值
位 23:16	EM_SLIB_DAT_SS	0x00	ro	主存扩展存储区安全库区数据起始扇区 (Extension memory sLib data start sector) 0: 无效扇区 1: 扇区 1 2: 扇区 2 ... 17: 扇区 17 0xFF: 无数据安全区 其余设定值: 无效
位 15:4	保留	0x000	resd	保持为默认值
位 3	SLIB_ENF	0	ro	sLib 使能标志 (sLib enabled flag) 该位置起时, 表示闪存主存区域部分或是全部 (依照 SLIB_STS1 设定) 作为安全库代码。
位 2	EM_SLIB_ENF	0	ro	主存扩展存储区 sLib 使能标志 (Extension memory sLib enabled flag)

				该位置起时，表示启动程序代码区域是作为主闪存扩展区域（BTM_AP_ENF 置起），并且存放的应用代码为安全库代码
位 1	保留	0	resd	保持为默认值
位 0	BTM_AP_ENF	0	ro	启动程序代码区域存放应用代码使能标志（Boot memory store application code enabled flag） 该位置起时，表示启动程序代码区域可以作为主存扩展区域存放用户应用代码；否则仅用于存放系统启动代码

### 5.7.10 闪存安全库区状态寄存器1（SLIB\_STS1）

专用于闪存安全库区。

域	简称	复位值	类型	功能
位 31:22	SLIB_ES	0x000	ro	主存安全库区结束扇区（sLib end sector） 0: 扇区 0 1: 扇区 1 2: 扇区 2 ... 127: 扇区 127
位 21:11	SLIB_DAT_SS	0x000	ro	主存安全库区数据区起始扇区（sLib data start sector） 0: 无效扇区 1: 扇区 1 2: 扇区 2 ... 127: 扇区 127 0x7FF: 无安全库区数据区
位 10:0	SLIB_SS	0x000	ro	主存安全库区起始扇区（sLib start sector） 0: 扇区 0 1: 扇区 1 2: 扇区 2 ... 127: 扇区 127

### 5.7.11 闪存安全库区密码清除寄存器（SLIB\_PWD\_CLR）

专用于闪存安全库区。

域	简称	复位值	类型	功能
位 31:0	SLIB_PCLR_VAL	0x0000 0000	wo	安全库区密码清除（sLib password clear value） 用于写入正确的安全库区密码，将实现解除安全库区功能。 此寄存器写入状态将在 SLIB_MISC_STS 位 0 与位 1 中体现。

### 5.7.12 闪存安全库区额外状态寄存器（SLIB\_MISC\_STS）

专用于闪存安全库区。

域	简称	复位值	类型	功能
位 31:3	保留	0x0000000	resd	保持为默认值
位 2	SLIB_ULKF	0	ro	SLib 解锁标志（sLib unlock flag） 当该位置起时表示 SLib 相关设定寄存器允许配置。
位 1	SLIB_PWD_OK	0	ro	密码正确（sLib password ok） 当密码正确，该位被硬件置起。
位 0	SLIB_PWD_ERR	0	ro	密码错误（sLib password error） 当密码错误，并且设定的密码清除寄存器的值不等于 0xFFFF FFFF，该位被硬件置起。 注意：当该位置起后，硬件将不再接受重新设定密码清除寄存器，直到再次复位。

### 5.7.13 闪存CRC校验地址寄存器（FLASH\_CRC\_ADDR）

专用于主闪存以及主存扩展区域。

域	简称	复位值	类型	功能
位 31:0	CRC_ADDR	0x0000 0000	wo	CRC 地址 (CRC address) 选择要校验的闪存扇区起始地址。 注意：必须与扇区起始地址对齐

注意：所有这些位是只写的，读出无反应。

### 5.7.14 闪存CRC校验控制寄存器（FLASH\_CRC\_CTRL）

专用于主闪存以及主存扩展区域。

域	简称	复位值	类型	功能
位 31:17	保留	0x0000	wo	保持为默认值
位 16	CRC_STRT	0x0	wo	启动 CRC 校验 (CRC start) 设置该位去启动用户代码或是安全库代码的 CRC 校验功能。 硬件启动 CRC 后，会自动清除该位。 注意： 校验数据从 CRC_ADDR ~ CRC_ADDR+CRC_SN*1 扇区
位 15:0	CRC_SN	0x0000	rw	CRC 校验扇区数量 (CRC sector nubmber) 设定本次 CRC 校验的数据量，单位是扇区

### 5.7.15 闪存CRC校验结果寄存器（FLASH\_CRC\_CHK）

专用于主闪存以及主存扩展区域。

域	简称	复位值	类型	功能
位 31:0	CRC_CHK	0x0000 0000	ro	CRC 校验结果 (CRC check result)

注意：所有这些位是只读的，写入无反应。

### 5.7.16 闪存安全库区密码设定寄存器（SLIB\_SET\_PWD）

专用于闪存安全库区密码设定。

域	简称	复位值	类型	功能
位 31:0	SLIB_PSET_VAL	0x0000 0000	wo	安全库区密码 (sLib password setting value) 注意：在解除安全库区锁定后，此寄存器才允许被写入，用于设定安全库区启动密码。但写入 0xFFFF_FFFF 以及 0x0000_0000 值无效。

注意：所有这些位是只写入，读出为 0。

### 5.7.17 闪存安全库区地址设定寄存器（SLIB\_SET\_RANGE）

专用于主存安全库区地址设定。

域	简称	复位值	类型	功能
位 31:22	SLIB_ES_SET	0x000	wo	主存安全库区结束扇区设定 (sLib end sector setting) 用于设定启动安全库区时的安全库区结束扇区位置 0: 扇区 0 1: 扇区 1 2: 扇区 2 ... 127: 扇区 127
位 21:11	SLIB_DSS_SET	0x000	wo	主存安全库区数据区起始扇区设定 (sLib data start sector setting)

位 10:0	SLIB_SS_SET	0x000	wo	用于设定启动安全库区时的数据区起始扇区位置 0: 无效扇区，设定将导致安全库区无法启动 1: 扇区 1 2: 扇区 2 ... 127: 扇区 127 0x7FF: 无安全库区数据区
				主存安全库区起始扇区设定 (sLib start sector setting) 用于设定启动安全库区时的安全库区起始扇区位置 0: 扇区 0 1: 扇区 1 2: 扇区 2 ... 127: 扇区 127

注意：

所有这些位是只写入，读出为 0。

在解除安全库区锁定后，此寄存器才允许被写入。

超过主闪存存储地址范围均是无效设定。

### 5.7.18 主存扩展存储区域安全库区设定寄存器 (EM\_SLIB\_SET)

专用于主存扩展区域。

域	简称	复位值	类型	功能
位 31:24	保留	0x00	wo	保持为默认值
位 23:16	EM_SLIB_DSS_SET	0x000	wo	主存扩展区安全库区数据区起始扇区设定 (Extension memory sLib data start sector setting) 用于设定启动安全库区时的数据区起始扇区位置 0: 无效扇区，设定将导致安全库区无法启动 1: 扇区 1 2: 扇区 2 ... 17: 扇区 17 0xFF: 无安全库区数据区 其余设定值：无效 注意： 当设为 0xFF，表示主存扩展区从扇区 0 至扇区 17 都为安全库区，且整个安全库区作为指令安全库区；
位 15:0	EM_SLIB_SET	0x000	wo	主存扩展存储区 sLib 设定 (Extension memory sLib setting) 写入 0x5AA5 将启动主存扩展区作为存放安全库区代码功能

注意：所有这些位是只写的，读出无反应。

### 5.7.19 启动程序代码区模式设定寄存器 (BTM\_MODE\_SET)

专用于启动程序代码区域。

域	简称	复位值	类型	功能
位 31:8	保留	0x000000	wo	保持为默认值
位 7:0	BTM_MODE_SET	0x00	wo	启动程序代码区模式设定 (Boot memory mode setting) 0xFF: 启动程序代码区域作为系统区域，存放系统启动代码功能 其他值：启动程序代码区域作为主存扩展区域，存放应用代码功能 注意：此寄存器的设定需要在闪存访问保护未启动下进行

注意：所有这些位是只写的，读出无反应。

### 5.7.20 闪存安全库区解锁寄存器（SLIB\_UNLOCK）

专用于安全库区寄存器的解锁设定。

域	简称	复位值	类型	功能
位 31:0	SLIB_UKVAL	0x0000 0000	wo	安全库区解锁键值（sLib unlock key value） 固定键值 0xA35F_6D24，用于安全库区设定寄存器的解锁。

注意：所有这些位是只写入，读出为 0。

# 6 通用功能输入输出 (GPIO)

## 6.1 简介

AT32WB415 支持多达 28 个双向 I/O 管脚，每个管脚都可以实现与外部的通讯、控制以及数据采集的功能。

每个管脚都支持通用功能输入输出 (GPIO) 或复用功能输入输出 (IOMUX)。本章节详细介绍 GPIO 功能，IOMUX 功能详见复用功能输入输出章节。

每个管脚都可以软件配置成浮空输入、上拉/下拉输入、模拟输入/输出、通用推挽/开漏输出、复用推挽/开漏输出。

每个管脚都可以软件配置输出驱动能力以及输出信号斜率。

每个管脚都可以配置为外部中断输入。

每个管脚都支持配置锁定功能。

## 6.2 功能描述

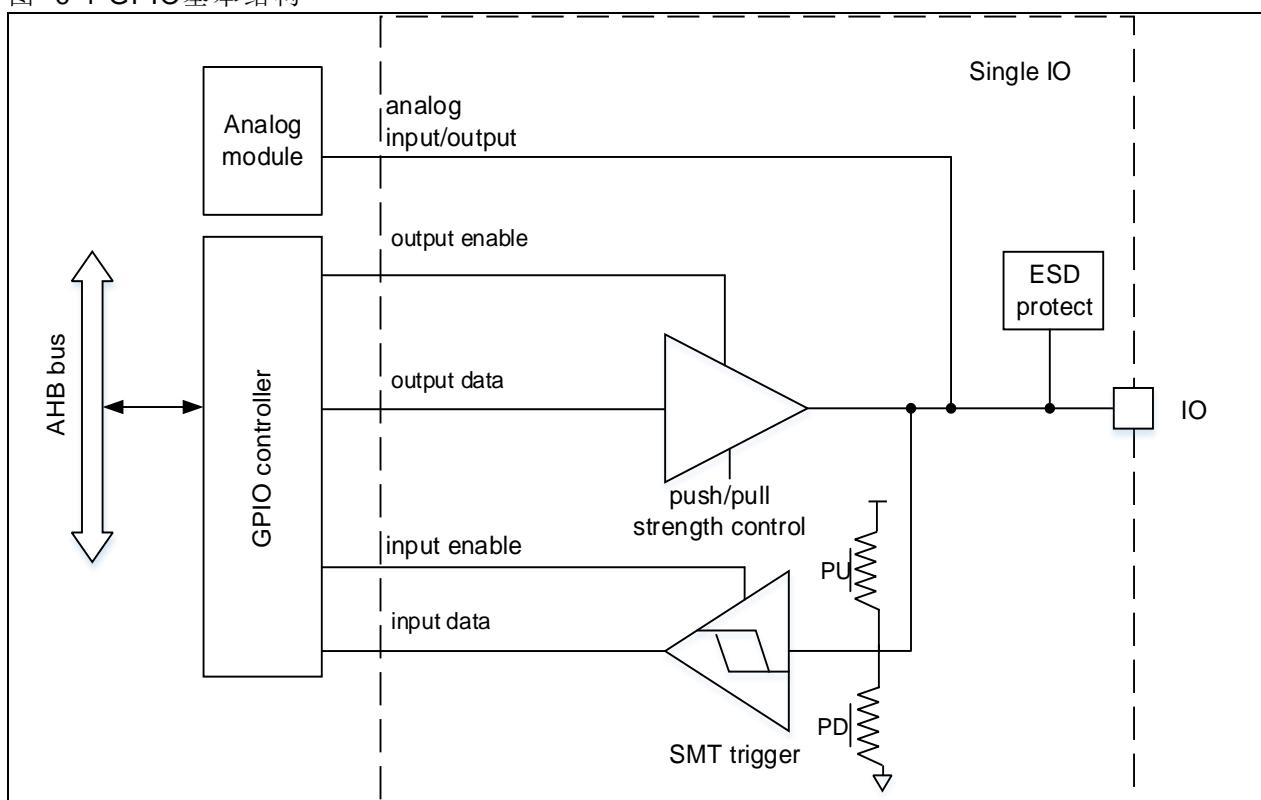
### 6.2.1 GPIO 结构

每个管脚可以由软件配置成四种输入模式（输入浮空、输入上拉、输入下拉、模拟输入）和四种输出模式（开漏输出、推挽式输出、推挽式复用、开漏复用）。

每个 I/O 端口对应的寄存器不允许半字或字节访问，必须按 32 位字被访问，每个 I/O 端口位可以自由编程。

下图给出了一个 I/O 端口位的基本结构。

图 6-1 GPIO 基本结构



### 6.2.2 GPIO 复位状态

系统上电或复位后，所有管脚都被配置为浮空输入模式。

### 6.2.3 通用功能输入配置

配置模式	IOFC	HDRV	IOMC[1]	IOMC[0]	ODT 寄存器
浮空输入	01		000		不使用

下拉输入	10	0
上拉输入		1

当管脚配置为输入时：

管脚状态可通过对输入数据寄存器的读访问得到

可配置管脚为浮空输入、上拉输入或下拉输入

施密特触发器有效

不能对该管脚进行输出。

注意：如果是浮空输入模式，为避免复杂环境下，没有使用的管脚有干扰，导致漏电，建议，如管脚不使用，则配置为模拟输入模式。

#### 6.2.4 模拟输入/输出配置

配置模式	IOFC	HDRV	IOMC[1]	IOMC[0]	ODT 寄存器
模拟输入输出	00		000		不使用

当 GPIO 端口被配置为模拟输入配置时：

施密特触发无效

不能对该管脚进行数字输入输出

对应的管脚，无任何上拉/下拉电阻。

#### 6.2.5 通用功能输出配置

配置模式	IOFC	HDRV	IOMC[1]	IOMC[0]	ODT 寄存器
推挽 (Push-Pull)	00		000/100: 输入模式 001: 输出模式，较大电流推动/吸入能力 010: 输出模式，适中电流推动/吸入能力 011: 输出模式，适中电流推动/吸入能力 1xx: 输出模式，极大电流推动/吸入能力		0 或 1
开漏 (Open-Drain)	01				0 或 1

当 GPIO 端口被配置为输出时：

施密特触发器有效

可通过输出寄存器让对应管脚输出

上拉和下拉电阻不能被使用

在开漏模式时，可强输出 0，可用上拉电阻输出 1。

在推挽模式时，可通过输出寄存器输出数字 0/1。

CONF = 10 或 11 时，为复用输出，详情请参考 IOMUX 章节

#### 6.2.6 I/O端口保护

为了防止误操作导致 GPIO 功能混乱，提供每个对应管脚的锁定机制。一旦锁定，在下次复位或者上电之前都不能进行对应管脚的 GPIO 配置。

### 6.3 GPIO寄存器

下面列出了 GPIO 寄存器映像和复位数值。

必须以字（32 位）的方式操作这些外设寄存器。

表 6-1 GPIO 寄存器映像和复位值

寄存器简称	基址偏移量	复位值
GPIOx_CFGLR	0x00	0x4444 4444
GPIOx_CFGHR	0x04	0x4444 4444
GPIOx_IDT	0x08	0x0000 XXXX
GPIOx_ODT	0x0C	0x0000 0000
GPIOx_SCR	0x10	0x0000 0000
GPIOx_CLR	0x14	0x0000 0000
GPIOx_WPR	0x18	0x0000 0000

### 6.3.1 GPIO配置低寄存器 (GPIOx\_CFGLR) (x=A..F)

域	简称	复位值	类型	功能
位 31: 30				GPIOx 功能配置 (y=0~7) (GPIOx function configurate)
位 27: 26				当 IO 模式配置为输入模式 (IOMCy[1: 0]=00) :
位 23: 22				00: 模拟;
位 19: 18	IOFCy	0x1	rw	01: 浮空 (复位后的状态);
位 15: 14				10: 下拉或上拉;
位 11: 10				11: 保留。
位 7: 6				当 IO 模式配置为输出模式 (IOMCy[1: 0]!00) :
位 3: 2				00: 通用推挽;
位 29: 28				01: 通用开漏;
位 25: 24				10: 复用推挽;
位 21: 20				11: 复用开漏。
位 17: 16	IOMCy	0x0	rw	GPIOx 模式配置 (y=0~7) (GPIOx mode configurate)
位 13: 12				00: 输入模式 (复位后的状态);
位 9: 8				01: 输出模式, 较大电流推动/吸入能力
位 5: 4				10: 输出模式, 适中电流推动/吸入能力
位 1: 0				11: 输出模式, 极大电流推动/吸入能力

注意：有些端口寄存器复位值不同。

### 6.3.2 GPIO配置高寄存器 (GPIOx\_CFGHR) (A..F)

域	简称	复位值	类型	功能
位 31: 30				GPIOx 功能配置 (y=8~15) (GPIOx function configurate)
位 27: 26				当 IO 模式配置为输入模式 (IOMCy[1: 0]=00) :
位 23: 22				00: 模拟;
位 19: 18	IOFCy	0x1	rw	01: 浮空 (复位后的状态);
位 15: 14				10: 下拉或上拉;
位 11: 10				11: 保留。
位 7: 6				当 IO 模式配置为输出模式 (IOMCy[1: 0]!00) :
位 3: 2				00: 通用推挽;
位 29: 28				01: 通用开漏;
位 25: 24				10: 复用推挽;
位 21: 20				11: 复用开漏。
位 17: 16	IOMCy	0x0	rw	GPIOx 模式配置 (y=8~15) (GPIOx mode configurate)
位 13: 12				00: 输入模式 (复位后的状态);
位 9: 8				01: 输出模式, 较大电流推动/吸入能力
位 5: 4				10: 输出模式, 适中电流推动/吸入能力
位 1: 0				11: 输出模式, 极大电流推动/吸入能力

注意：有些端口寄存器复位值不同。

### 6.3.3 GPIO输入数据寄存器 (GPIOx\_IDT) (x=A..F)

域	简称	复位值	类型	功能
位 31: 16	保留	0x0000	resd	保持默认值。
位 15: 0	IDT	0xFFFF	ro	GPIOx 输入的数据 (GPIOx input data) GPIOx 对应 IO 口的输入电平状态, 每一位对应 GPIOx 的一个 IO。

### 6.3.4 GPIO输出数据寄存器 (GPIOx\_ODT) (x=A..F)

域	简称	复位值	类型	功能
---	----	-----	----	----

位 31: 16 保留	0x0000	resd	保持默认值。 GPIOx 输出的数据 (GPIOx output data)。 每一位对应 GPIOx 的一个 IO。
位 15: 0 ODT	0x0000	rw	做输出功能时: GPIOx 对应 IO 口的输出电平状态。 0: 低电平; 1: 高电平。 做输入功能时: GPIOx 对应 IO 口的上拉/下拉状态。 0: 下拉; 1: 上拉。

### 6.3.5 GPIO设置/清除寄存器 (GPIOx\_SCR) (x=A..F)

域	简称	复位值	类型	功能
位 31: 16 IOCB		0x0000	wo	清除 GPIOx 位 (GPIOx clear bit) 写'1'的位其对应 ODT 寄存器位会清除, 写'0'的位其对应 ODT 寄存器位维持不变, 相当于 ODT 寄存器的位操作。 0: 对应位不变; 1: 对应位清除。
位 15: 0 IOSB		0x0000	wo	设置 GPIOx 位 (GPIOx set bit) 写'1'的位其对应 ODT 寄存器位会置起, 写'0'的位其对应 ODT 寄存器位维持不变, 相当于 ODT 寄存器的位操作。 如果 IOCB 和 IOSB 同一个位都写'1', 那么优先级更高的 IOSB 会生效。 0: 对应位不变; 1: 对应位置起。

### 6.3.6 GPIO清除寄存器 (GPIOx\_CLR) (x=A..F)

域	简称	复位值	类型	功能
位 31: 16 保留		0x0000	resd	保持默认值。
位 15: 0 IOCB		0x0000	wo	清除 GPIOx 的位 (GPIOx clear bit) 写'1'的位其对应 ODT 寄存器位会清除, 写'0'的位其对应 ODT 寄存器位维持不变, 相当于 ODT 寄存器的位操作。 0: 对应位不变; 1: 对应位清除。

### 6.3.7 GPIO写保护寄存器 (GPIOx\_WPR) (x=A..F)

域	简称	复位值	类型	功能
位 31: 17 保留		0x0000	resd	保持默认值。
位 16 WPSEQ		0x0	rw	写保护使能序列 (Write protect sequence) 想保护某些 IO 位不被写入, 需配合同时操作写保护使能序列位和 WPEN 位。 写保护使能位操作按照以下方式操作 4 次, 写'1' -> 写'0' -> 写'1' -> 读, 操作期间 WPSEL 位值不可修改。
位 15: 0 WPEN		0x0000	rw	写保护使能 (Write protect enable) 每一位对应 GPIOx 的一个 IO。 0: 无写保护; 1: 写保护。

# 7 复用功能输入输出 (IOMUX)

## 7.1 简介

AT32WB415 支持多达 28 个双向 I/O 管脚，每个管脚都可以实现与外部的通讯、控制以及数据采集的功能。

每个管脚都支持通用功能输入输出 (GPIO) 或复用功能输入输出 (IOMUX)。本章节详细介绍 IOMUX 功能，GPIO 功能详见通用功能输入输出章节。

每个管脚都通过软件配置 GPIO 配置低寄存器(GPIOx\_CFGLR)或 GPIO 配置高寄存器(GPIOx\_CFGHR)寄存器设定成复用功能输入输出端口。

大多数管脚支持多个外设的输出功能映射，可通过 IOMUX 章节寄存器来选择不同的外设输入输出功能。每个管脚都支持外部中断功能。

## 7.2 功能描述

### 7.2.1 IOMUX结构

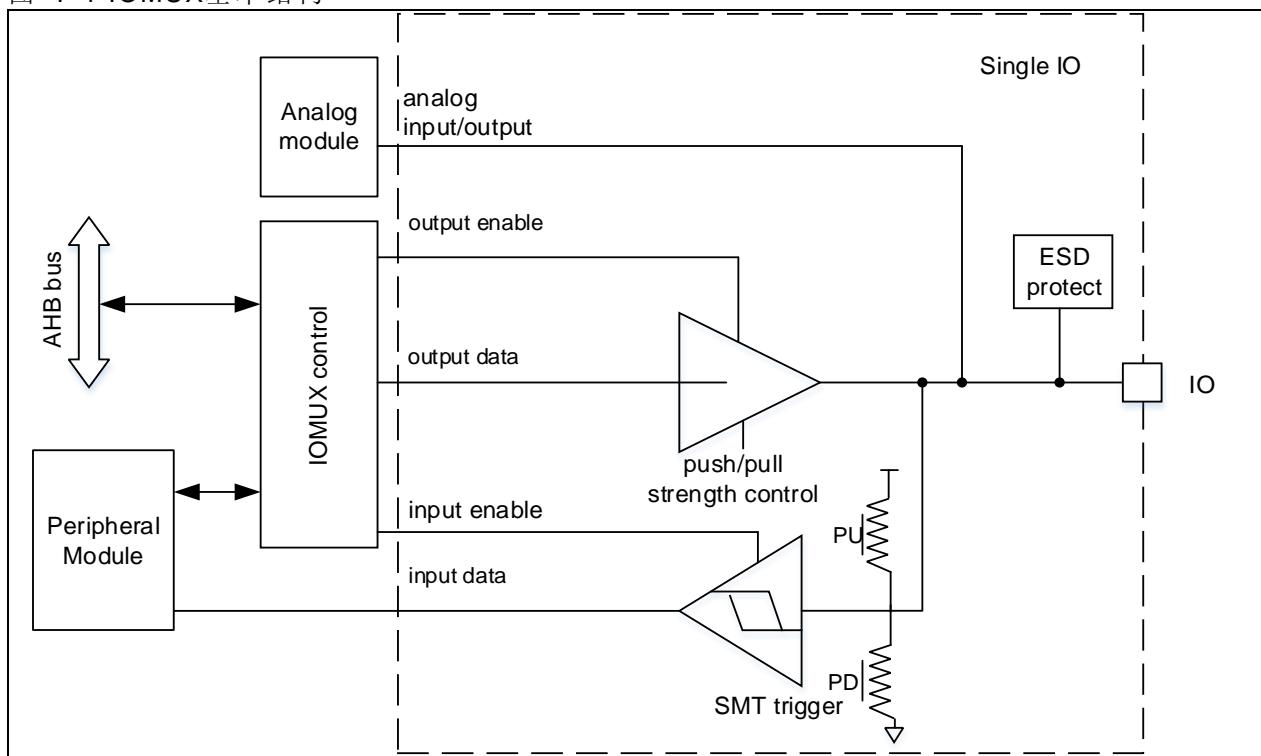
管脚作为复用输入功能时，与通用输入功能一样，端口配置成输入模式（浮空、上拉、下拉）。

要实现复用输出功能，必须配置 GPIO 配置低寄存器 (GPIOx\_CFGLR) 或 GPIO 配置高寄存器 (GPIOx\_CFGHR) 将该端口设定为复用功能输出模式（推挽或开漏）。此时管脚和 GPIO 控制器断开，由 IOMUX 控制器进行控制。

要实现双向复用功能，与复用输出功能一样，将该端口设定为复用功能输出模式（推挽或开漏）即可。由 IOMUX 控制器进行控制。

管脚作为复用输出功能时，一个管脚可能持多个外设的输出功能映射，需要通过配置 IOMUX 相关寄存器来选择复用输出功能。当管脚配置为复用输出功能但对应外设没有被激活的话，该管脚的输出将不确定。

图 7-1 IOMUX 基本结构



### 7.2.2 复用功能输入配置

当 I/O 端口配置为复用功能输入时：

管脚状态可通过对输入数据寄存器的读访问得到

可配置管脚为浮空输入、上拉输入或下拉输入

施密特触发器有效

不能对该管脚进行输出。

表 7-1 复用功能输入配置

配置模式	IOFC	HDRV	IOMC[1]	IOMC[0]	ODT 寄存器
浮空输入	01				不使用
下拉输入			000		0
上拉输入	10				1

### 7.2.3 复用功能输出或双向复用功能配置

当 I/O 端口配置为复用功能输出或双向复用功能时：

管脚输出由外设决定

施密特触发器有效

上拉和下拉电阻均关闭

如果管脚被误配成多个复用功能输出，管脚将按映射优先级输出，详见下面小节。

开漏模式时，读输入数据寄存器时可得到 I/O 口状态

推挽模式时，读输入数据寄存器时可得到 I/O 口状态

一些外设的输出功能可以重映射到不同的管脚，因此可以在不同封装中来选择 I/O 外设复用功能的数量达到最优化。通过配置复用重映射寄存器 (IOMUX\_REMAP) 或复用重映射寄存器 x (IOMUX\_REMAPx) (x=2,3...8) 来实现管脚的重新映射。

表 7-2 复用功能输出配置

配置模式	IOFC	HDRV	IOMC[1]	IOMC[0]
推挽 (Push-Pull)	10		001: 输出模式，较大电流推动/吸入能力	
			010: 输出模式，适中电流推动/吸入能力	
开漏 (Open-Drain)	11		011: 输出模式，适中电流推动/吸入能力	
			1xx: 输出模式，极大电流推动/吸入能力	

注意：配置为复用功能输出或双向复用功能时，必须满足  $IOMC[1:0] > 00$

### 7.2.4 外设复用功能管脚配置

当外设需要使用 IOMUX 复用功能时：

如果外设管脚需要作为复用输出则对应的管脚配置成复用推挽/开漏输出

如果外设管脚需要作为复用输入则对应的管脚配置成浮空输入/上拉输入/下拉输入

ADC 外设需要将模拟通道对应的管脚配置为模拟输入/输出模式

I2C 外设需要对应管脚作为双向复用功能时，需把对应的管脚配置复用开漏模式

### 7.2.5 IOMUX 映射优先级

单个管脚可能有多个外设复用映射，当多个外设复用映射到同一个管脚时，外设遵循以下优先级规则：

硬件抢占功能优先

非 timer 外设复用映射优先于 timer 外设。

多个非 timer 外设之间复用映射无优先级关系，复用功能叠加到同一个管脚。

#### 7.2.5.1 硬件抢占功能

某些管脚不管 GPIO 配置为任何模式，都会被特定的硬件功能占用。

表 7-3 硬件抢占功能

管脚名字	抢占使能位	说明
PA0	PWC_CTRLSTS[8]=1	抢占使能位有效之后，PA0 管脚直接作为 PWC 的 WKUP 功能使用
PA11	CRM_APB1EN[23]=1	抢占使能位有效之后，PA11 作为 USB_DM 通道使用
PA12	CRM_APB1EN[23]=1	抢占使能位有效之后，PA12 作为 USB_DP 通道使用
PC13	CRM_APB1EN[27]=1 & (BPR_CTRL[0]=1   BPR_RTCCAL[8]=1   BPR_RTCCAL[7]=1)	抢占使能位有效之后，PC13 作为 RTC 通道使用
PC14	CMR_BPDC[0]=1	抢占使能位有效之后，PC14 作为 LEXT 通道使用
PC15	CMR_BPDC[0]=1	抢占使能位有效之后，PC15 作为 LEXT 通道使用

#### 7.2.5.2 调试端口优先

在进行芯片调试时，为了防止其他外设对调试端口的干扰，导致不能被调试，配置好后的调试端口管脚，

不管对应管脚的 GPIO 寄存器被配置为任何模式，都会一直保持为调试端口。

为了在调试期间可以使用更多管脚，通过设置复用重映射和调试 I/O 配置寄存器 (IOMUX\_REMAP) 的 SWJTAG\_MUX [2: 0]位或复用重映射和调试 I/O 配置寄存器 7 (IOMUX\_REMAP7) 的 SWJTAG\_GMUX [2: 0]位，可以改变上述重映射配置。

表 7-4 调试端口映射

SWJTAG_MUX [2: 0]或 SWJTAG_GMUX [2: 0]	SWJ/I/O 管脚分配				
	PA13/JTMS/ SWDIO	PA14/JTCK/ SWCLK	PA15/JTDI	PB3/JTDO/ TRACESWO	PB4/NJTRST
000	√	√	√	√	√
001	√	√	√	√	×
010	√	√	×	×	×
100	×	×	×	×	×
其它	-	-	-	-	-

注意：√ 表示该管脚被强制分配给调试端口，× 表示该管脚可以释放给其他外设使用。

### 7.2.5.3 其他外设输出优先级关系

除了硬件抢占功能和端口调试功能以外，其他外设输出优先级如下：

非 timer 外设输出优先于 timer 外设，即其他外设和 timer 同时映射到某一管脚，timer 不能输出。

多个非 timer 外设输出如果映射到同一管脚，则这些外设输出会叠加输出到该管脚。

### 7.2.6 外部中断/唤醒线

每个管脚都支持作为外部中断的输入，对应的管脚须配置为输入模式。

## 7.3 IOMUX寄存器

下面列出了 IOMUX 寄存器映像和复位数值。

必须以字(32 位)的方式操作这些外设寄存器。

注：MCU PA9, PA10, PA15, PB0, PB1, PB3~5, PB10~12, PC0~11, PD2, PF4~7未连接建议软件将这些引脚全部设置为输出模式低电平以强化抗干扰能力并避免额外漏电，相应的IOMUX remap 也未出。

表 7-5 IOMUX寄存器地址映像和复位值

寄存器简称	基址偏移量	复位值
IOMUX_EVTOUT	0x00	0x0000 0000
IOMUX_REMAP	0x04	0x0000 0000
IOMUX_EXINTC1	0x08	0x0000
IOMUX_EXINTC2	0x0C	0x0000
IOMUX_EXINTC3	0x10	0x0000
IOMUX_EXINTC4	0x14	0x0000
IOMUX_REMAP2	0x1C	0x0000 0000
IOMUX_REMAP3	0x20	0x0000 0000
IOMUX_REMAP4	0x24	0x0000 0000
IOMUX_REMAP5	0x28	0x0000 0000
IOMUX_REMAP6	0x2C	0x0000 0000
IOMUX_REMAP7	0x30	0x0000 0000
IOMUX_REMAP8	0x34	0x0000 0000

注意：对寄存器 IOMUX\_EVTOUT, IOMUX\_REMAPx 和 IOMUX\_EXINTCx 进行读写操作前，应当首先打开 IOMUX 的时钟。

### 7.3.1 事件输出控制寄存器 (IOMUX\_EVTOUT)

域	简称	复位值	类型	功能
位 31: 8	保留	0x000000	resd	保持默认值。
位 7	EVOEN	0x0	rw	事件输出使能 (Event output enable) 使能后, Cortex-M 的 EVENTOUT 信号将连接到配置的 I/O 口。
位 6: 4	SELPORT	0x0	rw	选择 IO 端口 (Selection IO port) 选择输出 EVENTOUT 信号的 GPIO 端口: 000: GPIOA; 001: GPIOB; 010: GPIOC; 011: GPIOD; 101: GPIOF。
位 3: 0	SELPIN	0x0	rw	选择 IO 管脚 (x=A...E) (Selection IO pin) 选择输出 EVENTOUT 信号的 GPIOx 的 I/O 管脚: 0000: 管脚 0 0001: 管脚 1 0010: 管脚 2 0011: 管脚 3 0100: 管脚 4 0101: 管脚 5 0110: 管脚 6 0111: 管脚 7 1000: 管脚 8 1001: 管脚 9 1010: 管脚 10 1011: 管脚 11 1100: 管脚 12 1101: 管脚 13 1110: 管脚 14 1111: 管脚 15

### 7.3.2 IO复用重映射寄存器 (IOMUX\_REMAP)

域	简称	复位值	类型	功能
位 31	保留	0x0	resd	保持默认值。
位 30:27	保留	0x0	resd	保持默认值。
位 26: 24	SWJTAG_MUX	0x0	rw	SWD JTAG 复用 (SWD JTAG muxing) 配置 SWJTAG 接口相关的 IO 是否作为 GPIO 使用。 000: 支持 SWD 和 JTAG, 所有 SWJTAG 管脚不可作 GPIO; 001: 支持 SWD 和 JTAG, 禁用 NJTRST, PB4 可作 GPIO; 010: 支持 SWD, 禁用 JTAG, PA15/PB3/PB4 可作 GPIO; 100: 禁用 SWD 和 JTAG, 所有 SWJTAG 管脚均可作 GPIO; 其它: 无作用。
位 23:19	保留	0x0	resd	保持默认值。
位 18	ADC1_ETO_MUX	0x0	rw	ADC1 普通转换外部触发复用 (ADC1 external trigger ordinary conversion muxing) 选择 ADC1 普通转换外部触发输入。 0: ADC1 普通转换外部触发连接到 EXINT11; 1: ADC1 普通转换外部触发连接到 TMR8_TRGO。
位 17	ADC1_ETP_MUX	0x0	rw	ADC1 抢占转换外部触发复用 (ADC1 external trigger preempted conversion muxing) 选择 ADC1 抢占转换外部触发输入。 1: ADC1 抢占转换外部触发连接到 TMR1 通道 4。
位 16	TMR5CH4_MUX	0x0	rw	TMR5 通道 4 的内部复用 (TMR5 channel4 muxing) 选择 TMR5 通道 4 的内部映射。 0: TMR5_CH4 连接到 PA3; 1: TMR5_CH4 连接到 LICK 低速内部时钟, 可对 LICK 进行校准。
位 15	保留	0x0	rw	保持默认值。
位 14: 13	CAN_MUX	0x0	rw	CAN 的 IO 复用 (CAN IO muxing) 选择 CAN_TX 和 CAN_RX 的 IO 复用功能。 00: RX/PA11、TX/PA12; 01: 不使用;

				10: RX/ PB8、TX/ PB9; 11: 不使用；
位 12	保留	0x0	resd	保持默认值。
位 11: 10	保留	0x0	resd	保持默认值。
位 9: 8	TMR2_MUX	0x0	rw	定时器 2 的 IO 复用 (TMR2 IO muxing) 选择 TMR2 的 IO 复用功能。 00: CH1/EXT/PA0, CH2/PA1, CH3/PA2, CH4/PA3; 01: CH1/EXT/PA15, CH2/PB3, CH3/PA2, CH4/PA3; 10: CH1/EXT/PA0, CH2/PA1, CH3/PB10, CH4/PB11; 11: CH1/EXT/PA15, CH2/PB3, CH3/PB10, CH4/PB11。
位 7: 6	TMR1_MUX	0x0	rw	定时器 1 的 IO 复用 (TMR1 IO muxing) 选择 TMR1 的 IO 复用功能。 00: EXT/PA12, CH1/PA8, CH2/PA9, CH3/PA10, CH4/PA11, BRK/PB12, CH1C/PB13, CH2C/PB14, CH3C/PB15; 01: EXT/PA12, CH1/PA8, CH2/PA9, CH3/PA10, CH4/PA11, BRK/PA6, CH1C/PA7, CH2C/PB0, CH3C/PB1; 10: 不使用; 11: 不使用;
位 5: 4	USART3_MUX	0x0	rw	USART3 的 IO 复用 (USART3 IO muxing) 选择 USART3 的 IO 复用功能。 00: TX/PB10, RX/PB11, CK/PB12, CTS/PB13, RTS/PB14; 01: TX/PC10, RX/PC11, CK/PC12, CTS/PB13, RTS/PB14; 10: TX/PA7, RX/PA6, CK/PA5, CTS/PB1, RTS/PB0 11: 不使用；
位 3	保留	0x0	resd	保持默认值。
位 2	USART1_MUX	0x0	rw	USART1 的 IO 复用 (USART1 IO muxing) 选择 USART1 的 IO 复用功能。 0: TX/PA9, RX/PA10; 1: TX/PB6, RX/PB7。
位 1	I2C1_MUX	0x0	rw	I2C1 的 IO 复用 (I2C1 IO muxing) 选择 I2C1 的 IO 复用功能。 0: SCL/PB6, SDA/PB7 SMBA/PB5; 1: SCL/PB8, SDA/PB9 SMBA/PB5。
位 0	保留	0x0	resd	保持默认值。

### 7.3.3 复用外部中断配置寄存器1 (IOMUX\_EXINTC1)

域	简称	复位值	类型	功能
位 31: 16	保留	0x0000	resd	保持默认值。
位 15: 12	EXINT3	0x0000	rw	配置 EXINT3 的输入源 (configure EXINT3 source) 选择 EXINT3 外部中断的输入源。 0000: GPIOA 管脚 3 0001: GPIOB 管脚 3 0010: GPIOC 管脚 3 0011: GPIOD 管脚 3 0101: GPIOF 管脚 3 其他: 保留
位 11: 8	EXINT2	0x0000	rw	配置 EXINT2 的输入源 (configure EXINT2 source) 选择 EXINT2 外部中断的输入源。 0000: GPIOA 管脚 2 0001: GPIOB 管脚 2 0010: GPIOC 管脚 2 0011: GPIOD 管脚 2 0101: GPIOF 管脚 2

				其他: 保留
位 7: 4	EXINT1	0x0000	rw	配置 EXINT1 的输入源 (configure EXINT1 source) 选择 EXINT1 外部中断的输入源。 0000: GPIOA 管脚 1 0001: GPIOB 管脚 1 0010: GPIOC 管脚 1 0011: GPIOD 管脚 1 0101: GPIOF 管脚 1 其他: 保留
位 3: 0	EXINT0	0x0000	rw	配置 EXINT0 的输入源 (configure EXINT0 source) 选择 EXINT0 外部中断的输入源。 0000: GPIOA 管脚 0 0001: GPIOB 管脚 0 0010: GPIOC 管脚 0 0011: GPIOD 管脚 0 0101: GPIOF 管脚 0 其他: 保留

### 7.3.4 复用外部中断配置寄存器2 (IOMUX\_EXINTC2)

域	简称	复位值	类型	功能
位 31: 16	保留	0x0000	resd	保持默认值。
位 15: 12	EXINT7	0x0000	rw	配置 EXINT7 的输入源 (configure EXINT7 source) 选择 EXINT7 外部中断的输入源。 0000: GPIOA 管脚 7 0001: GPIOB 管脚 7 0010: GPIOC 管脚 7 0011: GPIOD 管脚 7 0101: GPIOF 管脚 7 其他: 保留
位 11: 8	EXINT6	0x0000	rw	配置 EXINT6 的输入源 (configure EXINT6 source) 选择 EXINT6 外部中断的输入源。 0000: GPIOA 管脚 6 0001: GPIOB 管脚 6 0010: GPIOC 管脚 6 0011: GPIOD 管脚 6 0101: GPIOF 管脚 6 其他: 保留
位 7: 4	EXINT5	0x0000	rw	配置 EXINT5 的输入源 (configure EXINT5 source) 选择 EXINT5 外部中断的输入源。 0000: GPIOA 管脚 5 0001: GPIOB 管脚 5 0010: GPIOC 管脚 5 0011: GPIOD 管脚 5 0101: GPIOF 管脚 5 其他: 保留
位 3: 0	EXINT4	0x0000	rw	配置 EXINT4 的输入源 (configure EXINT4 source) 选择 EXINT4 外部中断的输入源。 0000: GPIOA 管脚 4 0001: GPIOB 管脚 4 0010: GPIOC 管脚 4 0011: GPIOD 管脚 4 0101: GPIOF 管脚 4 其他: 保留

### 7.3.5 复用外部中断配置寄存器3 (IOMUX\_EXINTC3)

域	简称	复位值	类型	功能
位 31: 16	保留	0x0000	resd	保持默认值。
位 15: 12	EXINT11	0x0000	rw	配置 EXINT11 的输入源 (configure EXINT11 source) 选择 EXINT11 外部中断的输入源。 0000: GPIOA 管脚 11

				0001: GPIOB 管脚 11 0010: GPIOC 管脚 11 0011: GPIOD 管脚 11 0101: GPIOF 管脚 11 其他: 保留
位 11: 8	EXINT10	0x0000	rw	配置 EXINT10 的输入源 (configure EXINT10 source) 选择 EXINT10 外部中断的输入源。 0000: GPIOA 管脚 10 0001: GPIOB 管脚 10 0010: GPIOC 管脚 10 0011: GPIOD 管脚 10 0101: GPIOF 管脚 10 其他: 保留
位 7: 4	EXINT9	0x0000	rw	配置 EXINT9 的输入源 (configure EXINT9 source) 选择 EXINT9 外部中断的输入源。 0000: GPIOA 管脚 9 0001: GPIOB 管脚 9 0010: GPIOC 管脚 9 0011: GPIOD 管脚 9 0101: GPIOF 管脚 9 其他: 保留
位 3: 0	EXINT8	0x0000	rw	配置 EXINT8 的输入源 (configure EXINT8 source) 选择 EXINT8 外部中断的输入源。 0000: GPIOA 管脚 8 0001: GPIOB 管脚 8 0010: GPIOC 管脚 8 0011: GPIOD 管脚 8 0101: GPIOF 管脚 8 其他: 保留

### 7.3.6 复用外部中断配置寄存器4 (IOMUX\_EXINTC4)

域	简称	复位值	类型	功能
位 31: 16	保留	0x0000	resd	保持默认值。
位 15: 12	EXINT15	0x0000	rw	配置 EXINT15 的输入源 (configure EXINT15 source) 选择 EXINT15 外部中断的输入源。 0000: GPIOA 管脚 15 0001: GPIOB 管脚 15 0010: GPIOC 管脚 15 0011: GPIOD 管脚 15 0101: GPIOF 管脚 15 其他: 保留
位 11: 8	EXINT14	0x0000	rw	配置 EXINT14 的输入源 (configure EXINT14 source) 选择 EXINT14 外部中断的输入源。 0000: GPIOA 管脚 14 0001: GPIOB 管脚 14 0010: GPIOC 管脚 14 0011: GPIOD 管脚 14 0101: GPIOF 管脚 14 其他: 保留
位 7: 4	EXINT13	0x0000	rw	配置 EXINT13 的输入源 (configure EXINT13 source) 选择 EXINT13 外部中断的输入源。 0000: GPIOA 管脚 13 0001: GPIOB 管脚 13 0010: GPIOC 管脚 13 0011: GPIOD 管脚 13 0101: GPIOF 管脚 13 其他: 保留
位 3: 0	EXINT12	0x0000	rw	配置 EXINT12 的输入源 (configure EXINT12 source) 选择 EXINT12 外部中断的输入源。 0000: GPIOA 管脚 12

0001: GPIOB 管脚 12  
0010: GPIOC 管脚 12  
0011: GPIOD 管脚 12  
0101: GPIOF 管脚 12  
其他: 保留

### 7.3.7 IO复用重映射寄存器2 (IOMUX\_REMAP2)

域	简称	复位值	类型	功能
位 31: 28	保留	0x000	resd	保持默认值。
位 27: 26	CMP_MUX	0x0	w	CMP_MUX: CMP 内部重复用 (CMP internal remap) 该位可由软件置'1'或置'0'。它控制 CMP 的内部复用。 当该位置'00'时, CMP1_OUT 与 PA0 相连, CMP2_OUT 与 PA2 相连; 当该位置'01'时, CMP1_OUT 与 PA6 相连, CMP2_OUT 与 PA7 相连; 当该位置'10'时, CMP1_OUT 与 PA11 相连, CMP2_OUT 与 PA12 相连; 其它, 保留。
位 25: 0	保留	0x000	resd	保持默认值。

### 7.3.8 IO复用重映射寄存器3 (IOMUX\_REMAP3)

域	简称	复位值	类型	功能
位 31: 12	保留	0x0000000	resd	保持默认值。
位 11: 8	TMR11_GMUX	0x0	rw	TMR11 的 IO 全局复用 (TMR11 general muxing) 选择 TMR11 的 IO 复用功能。 0000: CH1/PB9 0010: CH1/PA7
位 7: 4	TMR10_GMUX	0x0	rw	TMR10 的 IO 全局复用 (TMR10 general muxing) 选择 TMR10 的 IO 复用功能。 0000: CH1/PB8 0010: CH1/PA6
位 3: 0	TMR9_GMUX	0x0	rw	TMR9 的 IO 全局复用 (TMR9 general muxing) 选择 TMR9 的 IO 复用功能。 0000: CH1/PA2 CH2/PA3 0010: CH1/PB14 CH2/PB15

### 7.3.9 IO复用重映射寄存器4 (IOMUX\_REMAP4)

域	简称	复位值	类型	功能
位 31: 20	保留	0x000	resd	保持默认值。
位 19	TMR5CH4_GMUX	0x0	rw	TMR5 通道 4 全局复用 (TMR5 channel4 general muxing) 选择 TMR5 通道 4 内部复用。 0: TMR5_CH4 与 PA3 相连; 1: LICK 内部振荡器与 TMR5_CH4 相连, 目的是对 LICK 进行校准。
位 18: 16	TMR5_GMUX	0x0	rw	TMR5 的 IO 全局复用 (TMR5 IO general muxing) 选择 TMR5 的 IO 复用功能。 0000: CH1/PA0 CH2/PA1 CH3/PA2 CH4/PA3 0001: CH1/PF4 CH2/PF5 CH3/PA2 CH4/PA3
位 15: 12	保留	0x0	resd	保持默认值。
位 11: 8	保留	0x0	resd	保持默认值。
位 7	保留	0x0	resd	保持默认值。
位 6: 4	TMR2_GMUX	0x0	rw	TMR2 的 IO 全局复用 (TMR2 IO general muxing) 选择 TMR2 的 IO 复用功能。 000: CH1_EXT/PA0 CH2/PA1 CH3/PA2 CH4/PA3 001: CH1_EXT/PA15 CH2/PB3 CH3/PA2 CH4/PA3 010: CH1_EXT/PA0 CH2/PA1 CH3/PB10 CH4/PB11 011: CH1_EXT/PA15 CH2/PB3 CH3/PB10 CH4/PB11
位 3: 0	TMR1_GMUX	0x0	rw	TMR1 的 IO 全局复用 (TMR1 IO general muxing)

选择 TMR1 的 IO 复用功能。  
 0000: EXT/PA12, CH1/PA8, CH2/PA9, CH3/PA10,  
 CH4/PA11, BRK/PB12, CH1C/PB13, CH2C/PB14,  
 CH3C/PB15;  
 0001: EXT/PA12, CH1/PA8, CH2/PA9, CH3/PA10,  
 CH4/PA11, BRK/PA6, CH1C/PA7, CH2C/PB0,  
 CH3C/PB1;  
 0010: EXT/PA0, CH1/PC6, CH2/PC7, CH3/PC8,  
 CH4/PC9, BRK/PA6, CH1C/PA7, CH2C/PB0,  
 CH3C/PB1;  
 其他: 不使用

### 7.3.10 IO复用重映射寄存器5 (IOMUX\_REMAP5)

域	简称	复位值	类型	功能
位 31: 24	保留	0x00	resd	保持默认值。
位 23: 20	SPI2_GMUX	0x0	rw	SPI2 的 IO 全局复用 (SPI2 IO general muxing) 选择 SPI2 的 IO 复用功能。 0000: CS/PB12, SCK/PB13, MISO/PB14, MOSI/PB15 MCK/PC6 . 0001: CS/PA15, SCK/PB3, MISO/PB4, MOSI/PB5 MCK/PC7。 其他: 不使用
位 19: 16	保留	0x0	resd	保持默认值。
位 15: 12	保留	0x0	resd	保持默认值。
位 11: 8	保留	0x0	resd	保持默认值。
位 7: 4	I2C1_GMUX	0x0	rw	I2C1 的 IO 全局复用 (I2C1 IO general muxing) 选择 I2C1 的 IO 复用功能。 0000: SCL/PB6, SDA/PB7, SMBA/PB5; 0001: SCL/PB8, SDA/PB9, SMBA/PB5. 0010: SCL/PF6, SDA/PF7, SMBA/PB5。 其他: 不使用
位 3: 0	保留	0x0	resd	保持默认值。

### 7.3.11 IO复用重映射寄存器6 (IOMUX\_REMAP6)

域	简称	复位值	类型	功能
位 31: 28	保留	0x0	resd	保持默认值。
位 27: 24	USART3_GMUX	0x0	rw	USART3 的 IO 全局复用 (USART3 IO general muxing) 选择 USART3 的 IO 复用功能。 0000: TX/PB10, RX/PB11, CK/PB12, CTS/PB13, RTS/PB14; 0001: TX/PC10, RX/PC11, CK/PC12, CTS/PB13, RTS/PB14; 0010: TX/PA7, RX/PA6, CK/PA5, CTS/PB1, RTS/PB0; 其他: 不使用
位 23: 20	保留	0x0	resd	保持默认值。
位 19: 16	USART1_GMUX	0x0	rw	USART1 的 IO 全局复用 (USART1 IO general muxing) 选择 USART1 的 IO 复用功能。 0000: TX/PA9, RX/PA10; 0001: TX/PB6, RX/PB7。 其他: 不使用
位 15: 12	保留	0x0	resd	保持默认值。
位 11: 8	保留	0x0	resd	保持默认值。
位 7: 4	保留	0x0	resd	保持默认值。

位 3: 0	CAN1_GMUX	0x0	rw	CAN1 的 IO 全局复用 (CAN1 IO general muxing) 选择 CAN1 的 IO 复用功能。 00: RX/PA11、TX/PA12; 10: RX/ PB8、TX/ PB9; 其他: 不使用
--------	-----------	-----	----	--

### 7.3.12 IO复用重映射寄存器7 (IOMUX\_REMAP7)

域	简称	复位值	类型	功能
位 31: 21 保留		0x000	resd	保持默认值。
位 20	PD01_GMUX	0x0	rw	PDO 复用到 HEXT_IN (PD0 muxing on HEXT_IN) 选择 PD0 和 PD1 的 GPIO 功能复用。 此功能只适用于 48 和 64 管脚的封装。 0: 无复用; 1: PDO 复用到 HEXT_IN。
位 19	保留	0x0	resd	保持默认值。
位 18: 16 SWJTAG_GMUX		0x0	rw	SWD JTAG 的 IO 全局复用 (SWD JTAG IO general muxing) 配置 SWJTAG 接口相关的 IO 是否作为 GPIO 使用。 000: 支持 SWD 和 JTAG, 所有 SWJTAG 管脚不可作 GPIO; 001: 支持 SWD 和 JTAG, 禁用 NJTRST, PB4 可作 GPIO; 010: 支持 SWD, 禁用 JTAG, PA15/PB3/PB4 可作 GPIO; 100: 禁用 SWD 和 JTAG, 所有 SWJTAG 管脚均可作 GPIO; 其它: 无作用。
位 15: 10 保留		0x00	resd	保持默认值。
位 9: 6 保留		0x0	resd	保持默认值。
位 5	ADC1ETO_GMUX	0x0	rw	ADC1 普通转换外部触发重复用 (ADC1 external trigger ordinary conversion general muxing) 选择 ADC1 普通转换外部触发输入。 0: ADC1 普通转换外部触发连接到 EXINT11; 1: ADC1 普通转换外部触发连接到 TMR8_TRGO。
位 4	ADC1ETP_GMUX	0x0	rw	ADC1 抢占转换外部触发重复用 (ADC1 external trigger preempted conversion general muxing) 该位可由软件置'1'或置'0'。它控制外部触发相连的触发输入。当该位置'1'时, ADC1 注入转换外部触发与 TMR1 通道 4 相连。
位 3: 0 保留		0x0	resd	保持默认值。

### 7.3.13 IO复用重映射寄存器8 (IOMUX\_REMAP8)

域	简称	复位值	类型	功能
位 31: 8 保留		0x0000 00	resd	保持默认值。
位 7: 6 保留		0x0	resd	保持默认值。
位 5: 4 UX	TMR2_CH4_CMP_GM	0x0	rw	控制 TMR2 通道 4 内部复用。 00, 01:由 TMR3_GMUX 控制选择的 IO 信号连接到 TMR2 的通道 4 10:由 CMP 的输出信号连接到 TMR2 的通道 4 11: CMP 的输出信号或上 TMR2_GMUX 控制选择的 IO 信号连接到 TMR2 通道 4
位 3: 2 UX	TMR1_CH1_CMP_GM	0x0	rw	控制 TMR1 通道 1 内部复用。 00, 01:由 TMR1_GMUX 控制选择的 IO 信号连接到 TMR1 的通道 1 10: 连接到 CMP 的输出信号连接到 TMR1 的通道 1 11: CMP 的输出信号或上 TMR1_GMUX 控制选择的 IO 信号连接到 TMR1 通道 1
位 1: 0 UX	TMR1_BK1_CMP_GM	0x0	rw	控制 TMR1 刹车输入通道 1 内部复用。 00, 01:由 TMR1_GMUX 控制选择的 IO 信号连接到 TMR1 的刹车输入通道 1

10: 由 CMP 输出信号连接到 TMR1 的刹车输入通道 1  
由 CMP 的输出信号连接到 TMR1 的刹车输入通道 1

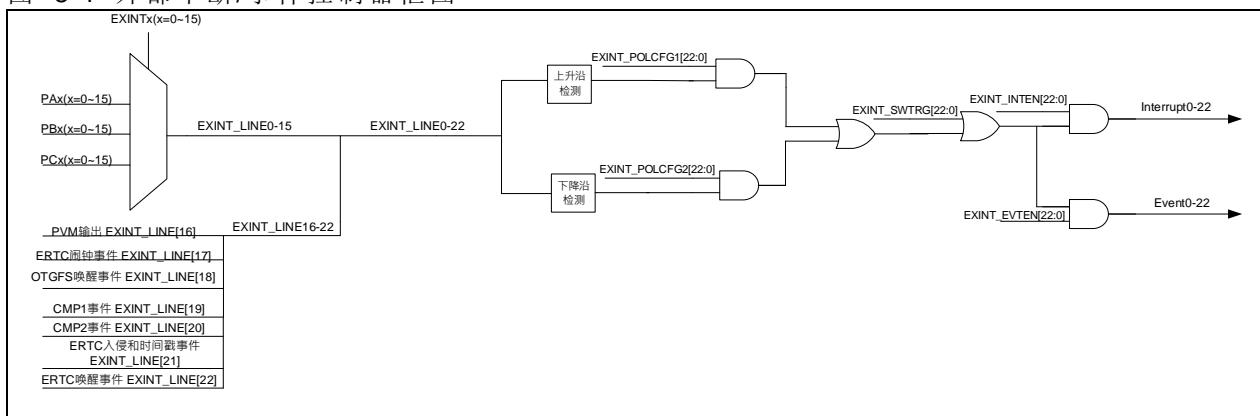
11: CMP 的输出信号或上 TMR1\_GMUX 控制选择的 IO 信号连接到 TMR1 刹车 输入通道 1

# 8 外部中断/事件控制器 (EXINT)

## 8.1 EXINT介绍

EXINT 共计有 23 条中断线 EXINT\_LINE[22:0], 每条中断线均支持通过边沿检测触发和软件触发来产生中断或事件。EXINT 可以根据软件配置，独立的使能或禁止中断或事件，并采取不同的边沿检测方式（检测上升沿或检测下降沿或同时检测上升沿和下降沿）以及触发方式（边沿检测触发或软件触发或边沿检测和软件同时触发）响应触发源独立的产生中断或事件。

图 8-1 外部中断/事件控制器框图



### EXINT 控制器的主要特性：

- 中断线 0~15 所映射的 IO 可以独立的配置
- 每个中断线都有独立的触发方式选择
- 每个中断都有独立的使能位
- 每个事件都有独立的使能位
- 共 23 个可独立产生和清除的软件触发
- 每个中断都有独立的状态位
- 每个中断都可以被独立的清除

## 8.2 功能描述和配置流程

EXINT 共计有 23 条中断线 EXINT\_LINE[22:0], 可以通过边沿检测的方式分别检测来自 GPIO 的外部中断源以及包括 PVM 输出, ERTC 闹钟事件, OTGFS 唤醒事件, CMP1 唤醒事件, CMP2 唤醒事件, ERTC 入侵和时间戳事件以及 ERTC 唤醒事件共七种芯片内部的中断源，其中来自 GPIO 的中断源可以通过软件编程配置 IOMUX 中的复用外部中断配置寄存器 x (IOMUX\_EXINTCx) 灵活的选择，需要注意的是这些输入源是互斥的，例如 EXINT\_LINE0 只能选择 PA0/PB0/PC0 中的某一个，而不能同时选择 PA0 和 PB0 作为输入源。

EXINT 支持多种边沿检测方式，每条中断线可以通过软件编程配置极性配置寄存器 1 (EXINT\_POLCFG1) 和极性配置寄存器 2 (EXINT\_POLCFG2) 独立的选择上升沿检测或下降沿检测或同时进行上升沿和下降沿检测，中断线上检测到的有效边沿触发可以用于产生事件或中断。

EXINT 支持独立的软件触发产生中断或事件，即除了来自中断线上的有效边沿外，用户可以通过软件编程配置软件触发寄存器 (EXINT\_SWTRG) 对应位来产生对应的中断或事件。

EXINT 具备独立的中断和事件使能位，用户可以通过软件编程配置中断使能寄存器 (EXINT\_INTEN) 和事件使能寄存器 (EXINT\_EVTEN) 来使能或关闭对应的中断或事件，这意味着无论是通过边沿检测还是软件触发产生中断或事件，都需要提前使能对应的中断或事件。

EXINT 具备独立的中断状态位，用户可以通过中断状态寄存器 (EXINT\_INTSTS) 读取对应的中断状态并通过对该寄存器相应位写 1 来清除已置位的状态标志。

### 中断初始化流程

- 选择中断源，即配置复用外部中断配置寄存器 x (IOMUX\_EXINTCx) (如果需要使用 GPIO 作为中断源需要该步骤)
- 选择触发方式，即配置极性配置寄存器 1 (EXINT\_POLCFG1) 和极性配置寄存器 2 (EXINT\_POLCFG2)

- 使能中断或事件，即配置中断使能寄存器（EXINT\_INTEN）和事件使能寄存器（EXINT\_EVTEN）
  - 产生软件触发，即配置软件触发寄存器（EXINT\_SWTRG）产生软件触发（此步骤仅适用于需软件触发产生中断的应用）
- 中断清除流程**
- 清除标志，即对中断状态寄存器（EXINT\_INTSTS）对应位写 1 来清除已产生的中断，同时该操作会同步清除软件触发寄存器（EXINT\_SWTRG）中的对应位。

## 8.3 EXINT 寄存器描述

下表列出了 EXINT 寄存器的映像和复位值。

必须以字（32 位）的方式操作这些外设寄存器。

表 8-1 外部中断/事件控制器寄存器映像和复位值

寄存器简称	基址偏移量	复位值
EXINT_INTEN	0x00	0x0000 0000
EXINT_EVTEN	0x04	0x0000 0000
EXINT_POLCFG1	0x08	0x0000 0000
EXINT_POLCFG2	0x0C	0x0000 0000
EXINT_SWTRG	0x10	0x0000 0000
EXINT_INTSTS	0x14	0x0000 0000

### 8.3.1 中断使能寄存器（EXINT\_INTEN）

域	简称	复位值	类型	功能
位 31: 23	保留	0x000	resd	硬件强制为 0。
位 22: 0	INTENx	0x00000	rw	线 x 上的中断使能/禁止位（Interrupt enable or disable on line x） 0: 禁止中断请求； 1: 使能中断请求。

### 8.3.2 事件使能寄存器（EXINT\_EVTEN）

域	简称	复位值	类型	功能
位 31: 23	保留	0x000	resd	硬件强制为 0。
位 22: 0	EVTENx	0x00000	rw	线 x 上的事件使能/禁止位（Event enable or disable on line x） 0: 禁止事件请求； 1: 使能事件请求。

### 8.3.3 极性配置寄存器1（EXINT\_POLCFG1）

域	简称	复位值	类型	功能
位 31: 23	保留	0x000	resd	硬件强制为 0。
位 22: 0	RPx	0x00000	rw	线 x 上的上升沿触发事件配置位（Rising polarity configuration bit of line x） 这些位用于选择线 x 由上升沿触发中断和事件。 0: 禁止上升沿触发； 1: 使能上升沿触发。

### 8.3.4 极性配置寄存器2（EXINT\_POLCFG2）

域	简称	复位值	类型	功能
位 31: 23	保留	0x000	resd	硬件强制为 0。
位 22: 0	FPx	0x00000	rw	线 x 上的下降沿触发事件配置位（Falling polarity event configuration bit of line x） 这些位用于选择线 x 由下降沿触发中断和事件。 0: 禁止下降沿触发； 1: 允许下降沿触发。

### 8.3.5 软件触发寄存器 (EXINT\_SWTRG)

域	简称	复位值	类型	功能
位 31: 23	保留	0x000	resd	硬件强制为 0。 软件触发线 x (Software trigger on line x) 当中断使能寄存器 (EXINT_INTEN) 中的对应位为 1，则 软件写此位硬件将自动置起中断状态寄存器 (EXINT_INTSTS) 中的对应位并产生中断。
位 22: 0	SWTx	0x00000	rw	当事件使能寄存器 (EXINT_EVTEN) 中的对应位为 1， 则软件写此位硬件将自动产生对应中断线上的事件。 0: 默认值； 1: 产生软件触发。 注：通过清除中断状态寄存器 (EXINT_INTSTS) 的对应位 (写入 1)，可以清除该位为 0。

### 8.3.6 中断状态寄存器 (EXINT\_INTSTS)

域	简称	复位值	类型	功能
位 31: 23	保留	0x000	resd	硬件强制为 0。 线 x 状态位 (Line x state bit)
位 22: 0	LINEx	0x00000	rw1c	0: 没有发生中断； 1: 发生了中断。 注：在该位中写入'1'可以清除它。

## 9 DMA 控制器 (DMA)

### 9.1 简介

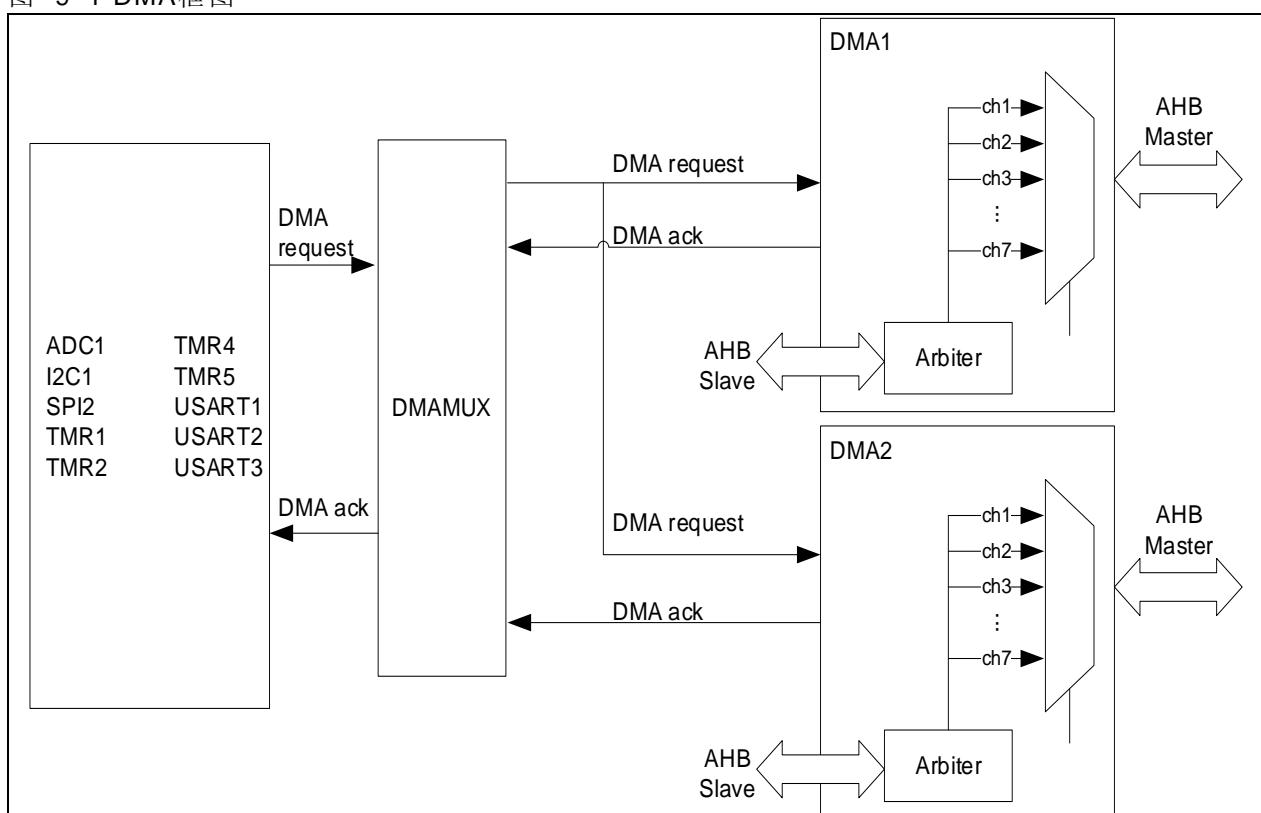
直接存储器访问 (DMA) 控制器，不仅旨在增强系统性能并减少处理器的中断生成，而且还针对 32 位 MCU 应用程序而设计。

一个处理器包含 2 个 DMA 控制器。每个控制器各有 7 个 DMA 通道，每个通道管理来自于外设对存储器访问的请求，并由仲裁器来协调各个 DMA 请求的优先权。

### 9.2 特性

- 符合 AMBA 规范 (Rev. 2.0)
- 仅支持 AHB OKAY 和 ERROR 响应
- 不支持 AHB 主接口的 HBUSREQ 和 HGRANT
- 支持 7 个通道
- 支持外设到存储器，存储器到外设和存储器到存储器的传输
- 支持硬件握手
- 支持 8 位，16 位和 32 位数据宽度传输
- 传输数据长度最大为 65535，可由编程配置
- 支持弹性映射

图 9-1 DMA 框图



注意：根据不同型号，图中 DMA 外设可能会有所减少。

### 9.3 功能描述

#### 9.3.1 通道配置

1. 设置外设地址 (DMA 通道x外设地址寄存器 (DMA\_CxPADDR))  
数据传输的初始外设地址，在传输过程中不会被改变。
2. 设置存储器地址 (DMA 通道x存储器地址寄存器 (DMA\_CxMADDR))  
数据传输的初始存储器地址，在传输过程中不会被改变。

### 3. 配置数据传输量 (DMA通道x数据传输量寄存器 (DMA\_CxDTCNT))

可编程的传输数据长度最大为65535。在传输过程中，该传输数据量的值会逐渐递减。

### 4. 配置通道设定 (DMA通道x配置寄存器 (DMA\_CxCTRL))

包含通道优先级，数据传输的方向、宽度，地址增量模式、循环模式和中断方式。

#### ● 通道优先级 (CHPL)

分为4个等级，最高优先级、高优先级、中等优先级和低优先级。

若有2个通道优先级设定相同，则较低编号的通道有较高的优先权。举例，通道1优先于通道2。

#### ● 数据传输方向 (DTD)

分为存储器到外设 (M2P)，外设到存储器 (P2M)。

#### ● 地址增量模式 (PINCM/MINCM)

当设置为增量模式时，下一笔传输的地址将是前一笔传输地址加上传输宽度 (PWIDHT/MWIDHT)。

#### ● 循环模式 (LM)

当通道配置设定为循环模式时，在最后一次传输后 DMA 通道 x 数据传输量寄存器 (DMA\_CxDTCNT) 的内容会恢复成初始值。

#### ● 存储器到存储器模式 (M2M)

存储器到存储器模式是 DMA 在没有外设请求的情况下进行数据传输。

循环模式与存储器到存储器模式不能同时使用。

### 5. 使能该通道的DMA传输 (DMA\_CxCTRL寄存器的CHEN位)

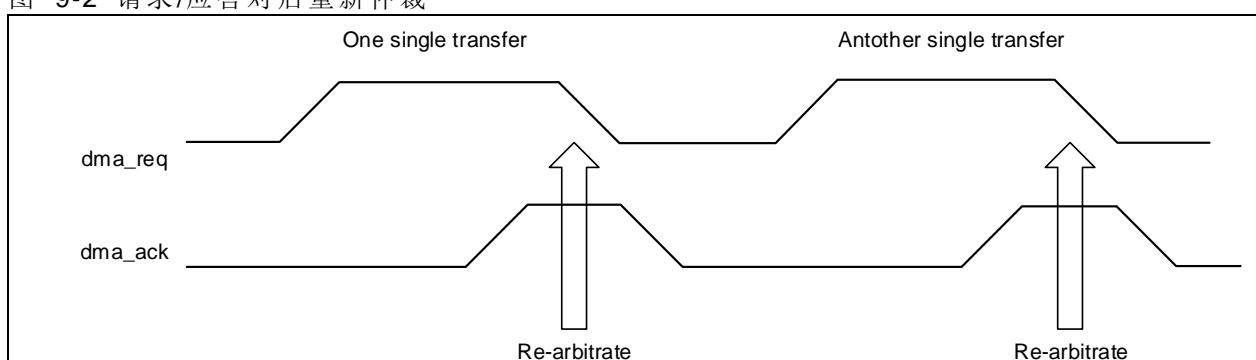
## 9.3.2 握手机制

在 P2M 和 M2P 传输模式，外设需要向 DMA 控制器发送请求信号。该通道将发出外设传输（单次），直到请求信号被应答为止。外设传输完成后，DMA 控制器将应答信号发送到外设。外设从 DMA 控制器获得应答信号后立即释放其请求。一旦外设取消了请求，DMA 控制器将释放应答信号。

## 9.3.3 仲裁

当同时启用多个通道时，仲裁器将在主控制器完全传输数据后重新进行仲裁。优先级最高的通道等待当前占用主控制器的通道完成数据传输后，将具有主控制器使用权。每当通道以外设主控制器的优先级完成一个单次传输后，外设主控制器就会重新仲裁以服务其他通道。

图 9-2 请求/应答对后重新仲裁



## 9.3.4 可编程数据传输宽度

通过 DMA 通道 x 配置寄存器 (DMA\_CxCTRL) 中的 PWIDHT 和 MWIDHT 位可以对源数据和目标数据的数据宽度进行编程，当 PWIDHT 不等于 MWIDHT 时，会依据 PWIDHT/ MWIDHT 设定将资料对齐。

图 9-3 PWIDHT: byte, MWIDTH: half-word

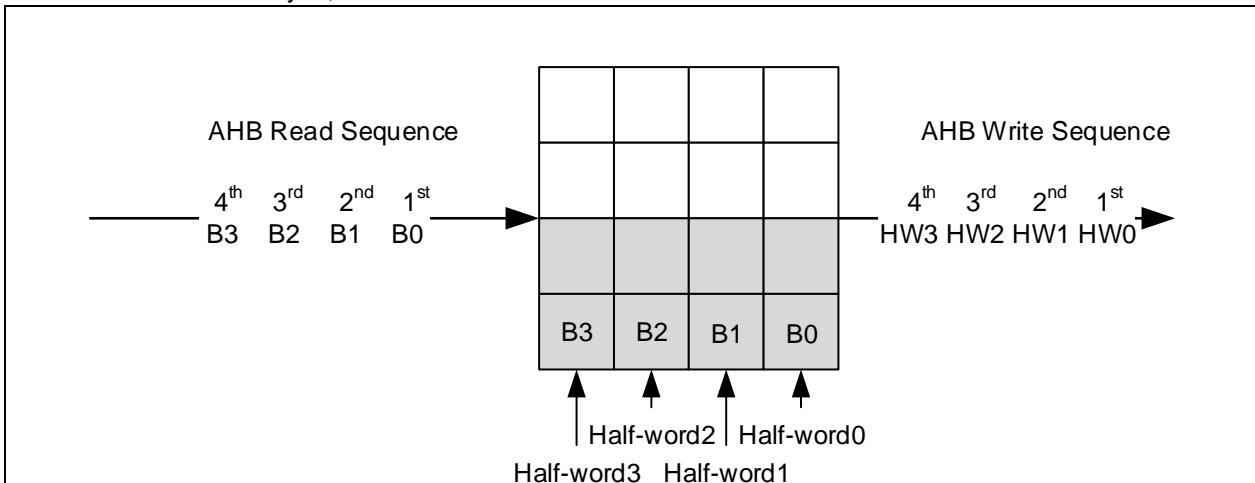


图 9-4 PWIDHT: half-word, MWIDTH: word

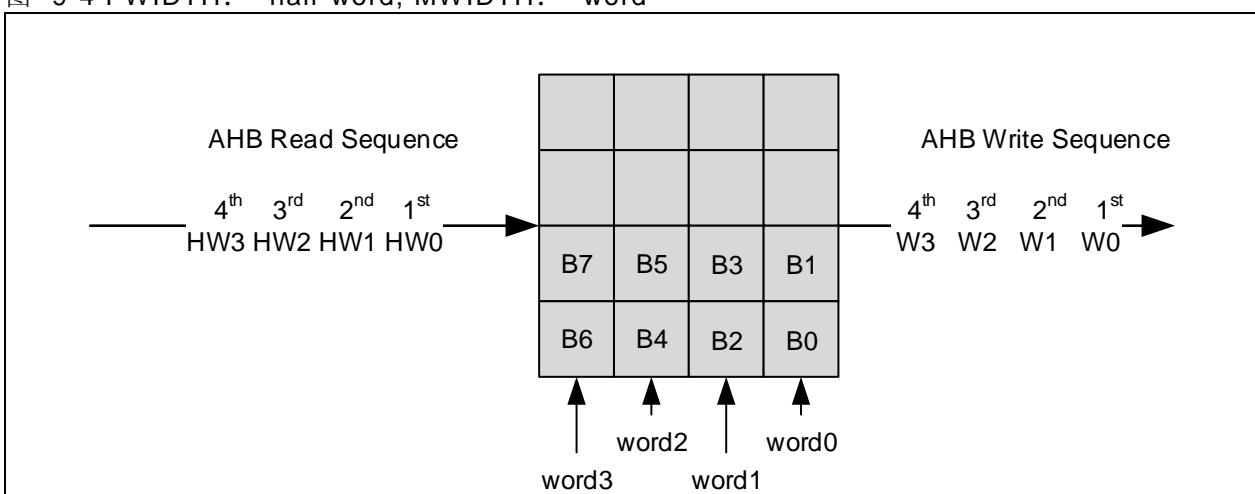
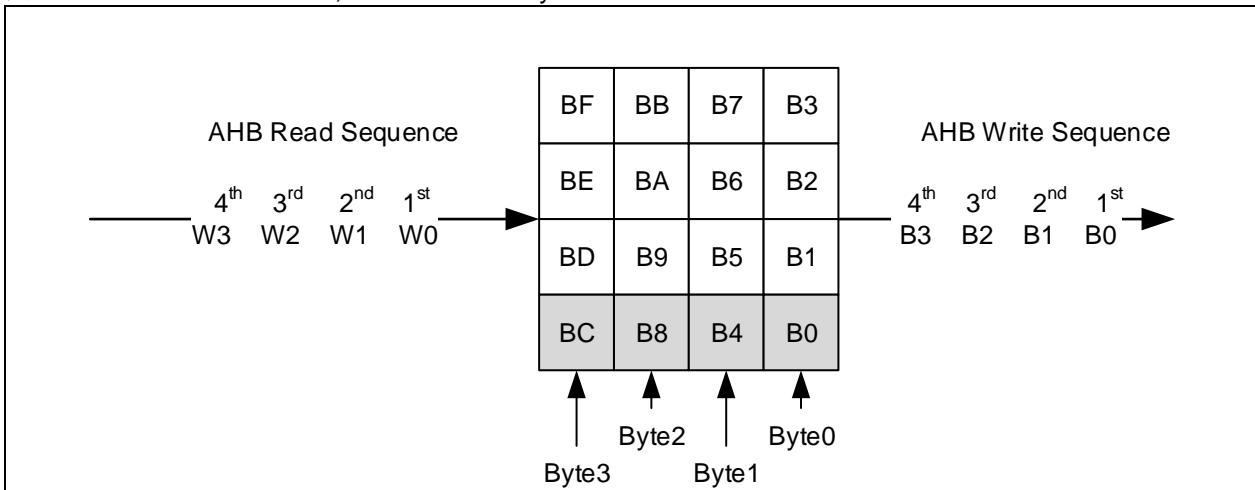


图 9-5 PWIDHT: word, MWIDTH: byte



### 9.3.5 错误事件

表 9-1 DMA错误事件

错误事件	
传输错误	DMA 读/写访问期间发生 AHB 响应错误

### 9.3.6 中断

DMA 可在传输过半、传输完成和传输错误时产生中断。每个通道的中断都有专用标志，清除和使能位如下表所示。

表 9-2 DMA中断

中断事件	事件标志位	清除控制位	使能控制位
半传输	HDTF	HDTFC	HDTIEN
传输完成	FDTF	FDTFC	FDTIEN
传输错误	DTERRF	DTERRFC	DTERRIEN

注意： DMA2 通道 4/通道 5，通道 6/通道 7 的中断被映射在同一个中断向量上。

### 9.3.7 DMA固定请求映射

数个外设请求通过逻辑“OR”运算映像到一个 DMA 通道，用户必须确保在一个通道上一次仅激活一个外设请求。另外，通过设置相应外设寄存器中的控制位，可以独立地开启或关闭外设的 DMA 请求。

表 9-3 DMA1各通道的外设请求

外设	通道 1	通道 2	通道 3	通道 4	通道 5	通道 6	通道 7
ADC1	ADC1						
SPI				SPI2	SPI2		
USART		USART3_TX	USART3_RX	USART1_TX	USART1_RX	USART2_RX	USART2_TX
I <sup>2</sup> C						I2C1_TX	I2C1_RX
TMR1		TMR1_CH1	TMR1_CH2	TMR1_CH4 TMR1_TRIG TMR1_HALL	TMR1_OVERFLOW	TMR1_CH3	
TMR2	TMR2_CH3	TMR2_OVERFLOW			TMR2_CH1		TMR2_CH2 TMR2_CH4
TMR3		TMR3_CH3	TMR3_CH4 TMR3_OVERFLOW			TMR3_CH1 TMR3_TRIG	
TMR4	TMR4_CH1			TMR4_CH2	TMR4_CH3		TMR4_OVERFLOW

表 9-4 DMA2各通道的外设请求

外设	通道 1	通道 2	通道 3	通道 4	通道 5	通道 6	通道 7
TMR5	TMR5_CH4 TMR5_TRIG	TMR5_CH3 TMR5_OVERFLOW			TMR5_CH2	TMR5_CH1	

### 9.3.8 DMA弹性请求映射

当设定弹性模式时 (DMA\_FLEX\_EN = 1)，每个通道的请求来源由 CHx\_SRC 来设定[x=1~7]。

使用例子：假如 DMA 通道 1 指定成 USART3\_TX，通道 3 要指定成 USART3\_RX，其他不使用，则设定上必须是 DMA\_FLEX\_EN=1，CH1\_SRC=30，CH3\_SRC=29，CH[2/4/5/6/7]\_SRC=0。

CHx\_SRC 设定值对应 DMA 来源见下表：

表 9-5 DMA各通道的弹性请求

CHx_SRC	请求来源	CHx_SRC	DMA 来源	CHx_SRC	请求来源	CHx_SRC	请求来源
0	No select	1	ADC1	2	reserved	3	reserved
4	reserved	5	reserved	6	reserved	7	reserved
8	reserved	9	reserved	10	reserved	11	SPI2_RX
12	SPI2_TX	13	reserved	14	reserved	15	reserved
16	reserved	17	reserved	18	reserved	19	reserved
20	reserved	21	reserved	22	reserved	23	reserved
24	reserved	25	USART1_RX	26	USART1_TX	27	USART2_RX
28	USART2_TX	29	USART3_RX	30	USART3_TX	31	reserved
32	reserved	33	UART5_RX	34	UART5_TX	35	reserved
36	reserved	37	reserved	38	reserved	39	reserved
40	reserved	41	I2C1_RX	42	I2C1_TX	43	reserved
44	reserved	45	reserved	46	reserved	47	reserved

48	reserved	49	reserved	50	reserved	51	reserved
52	reserved	53	TMR1_TRIG	54	TMR1_HALL	55	TMR1_OVERFLOW
56	TMR1_CH1	57	TMR1_CH2	58	TMR1_CH3	59	TMR1_CH4
60	reserved	61	TMR2_TRIG	62	reserved	63	TMR2_OVERFLOW
64	TMR2_CH1	65	TMR2_CH2	66	TMR2_CH3	67	TMR2_CH4
68	reserved	69	TMR3_TRIG	70	TMR3_HALL	71	TMR3_OVERFLOW
72	TMR3_CH1	73	TMR3_CH2	74	TMR3_CH3	75	TMR3_CH4
76	reserved	77	TMR4_TRIG	78	reserved	79	TMR4_OVERFLOW
80	TMR4_CH1	81	TMR4_CH2	82	TMR4_CH3	83	TMR4_CH4
84	reserved	85	TMR5_TRIG	86	reserved	87	TMR5_OVERFLOW
88	TMR5_CH1	89	TMR5_CH2	90	TMR5_CH3	91	TMR5_CH4
92	reserved	93	reserved	94	reserved	95	reserved
96	reserved	97	reserved	98	reserved	99	reserved
100	reserved	101	reserved	102	reserved	103	reserved
104	reserved	105	reserved	106	reserved	107	reserved
108	reserved	109	reserved	110	reserved	111	reserved
112	reserved	113	reserved	114	reserved	115	reserved
116	reserved	117	reserved	118	reserved	119	reserved

## 9.4 DMA寄存器

下表列出了 DMA 寄存器的映像和复位值。

可以用字节（8 位）、半字（16 位）或字（32 位）的方式操作这些外设寄存器。

表 9-6 DMA寄存器的映像和复位值

寄存器简称	基址偏移量	复位值
DMA_STS	0x00	0x0000 0000
DMA_CLR	0x04	0x0000 0000
DMA_C1CTRL	0x08	0x0000 0000
DMA_C1DTCNT	0x0C	0x0000 0000
DMA_C1PADDR	0x10	0x0000 0000
DMA_C1MADDR	0x14	0x0000 0000
DMA_C2CTRL	0x1C	0x0000 0000
DMA_C2DTCNT	0x20	0x0000 0000
DMA_C2PADDR	0x24	0x0000 0000
DMA_C2MADDR	0x28	0x0000 0000
DMA_C3CTRL	0x30	0x0000 0000
DMA_C3DTCNT	0x34	0x0000 0000
DMA_C3PADDR	0x38	0x0000 0000
DMA_C3MADDR	0x3C	0x0000 0000
DMA_C4CTRL	0x44	0x0000 0000
DMA_C4DTCNT	0x48	0x0000 0000
DMA_C4PADDR	0x4C	0x0000 0000

DMA_C4MADDR	0x50	0x0000 0000
DMA_C5CTRL	0x58	0x0000 0000
DMA_C5DTCNT	0x5C	0x0000 0000
DMA_C5PADDR	0x60	0x0000 0000
DMA_C5MADDR	0x64	0x0000 0000
DMA_C6CTRL	0x6C	0x0000 0000
DMA_C6DTCNT	0x70	0x0000 0000
DMA_C6PADDR	0x74	0x0000 0000
DMA_C6MADDR	0x78	0x0000 0000
DMA_C7CTRL	0x80	0x0000 0000
DMA_C7DTCNT	0x84	0x0000 0000
DMA_C7PADDR	0x88	0x0000 0000
DMA_C7MADDR	0x8C	0x0000 0000
DMA_SRC_SEL0	0xA0	0x0000 0000
DMA_SRC_SEL1	0xA4	0x0000 0000

注意：在以下列举的所有寄存器中，所有与通道 6 和通道 7 相关的位，对 DMA2 固定请求映像不适用，因为 DMA2 固定请求映像只有 5 个通道。对于 DMA2 的弹性请求映像，则可支持到 7 个通道。

#### 9.4.1 DMA状态寄存器 (DMA\_STS)

访问：无等待状态，字，半字和字节访问

域	简称	复位值	类型	功能
31: 28	保留	0x0	resd	保持默认值。
位 27	DTERRF7	0x0	ro	通道 7 数据传输错误事件标志 (data transfer error event flag) 0: 未发生错误传输事件 1: 发生错误传输事件
位 26	HDTF7	0x0	ro	通道 7 半数据传输事件标志 (half data transfer event flag) 0: 未发生半传输事件 1: 发生半传输事件
位 25	FDTF7	0x0	ro	通道 7 数据传输完成事件标志 (full data transfer event flag) 0: 未发生传输完成事件 1: 发生传输完成事件
位 24	GF7	0x0	ro	通道 7 全局事件标志 (Global event flag) 0: 未发生传输错误、半传输完成或传输完成事件 1: 发生传输错误、半传输完成或传输完成事件
位 23	DTERRF6	0x0	ro	通道 6 数据传输错误事件标志 (data transfer error event flag) 0: 未发生错误传输事件 1: 发生错误传输事件
位 22	HDTF6	0x0	ro	通道 6 半数据传输事件标志 (half data transfer event flag) 0: 未发生半传输事件 1: 发生半传输事件
位 21	FDTF6	0x0	ro	通道 6 数据传输完成事件标志 (full data transfer event flag) 0: 未发生传输完成事件 1: 发生传输完成事件

位 20	GF6	0x0	ro	通道 6 全局事件标志 (Global event flag) 0: 未发生传输错误、半传输完成或传输完成事件 1: 发生传输错误、半传输完成或传输完成事件
位 19	DTERRF5	0x0	ro	通道 5 数据传输错误事件标志 (data transfer error event flag) 0: 未发生错误传输事件 1: 发生错误传输事件
位 18	HDTF5	0x0	ro	通道 5 半数据传输事件标志 (half data transfer event flag) 0: 未发生半传输事件 1: 发生半传输事件
位 17	FDTF5	0x0	ro	通道 5 数据传输完成事件标志 (full data transfer event flag) 0: 未发生传输完成事件 1: 发生传输完成事件
位 16	GF5	0x0	ro	通道 5 全局事件标志 (Global event flag) 0: 未发生传输错误、半传输完成或传输完成事件 1: 发生传输错误、半传输完成或传输完成事件
位 15	DTERRF4	0x0	ro	通道 4 数据传输错误事件标志 (data transfer error event flag) 0: 未发生错误传输事件 1: 发生错误传输事件
位 14	HDTF4	0x0	ro	通道 4 半数据传输事件标志 (half data transfer event flag) 0: 未发生半传输事件 1: 发生半传输事件
位 13	FDTF4	0x0	ro	通道 4 数据传输完成事件标志 (full data transfer event flag) 0: 未发生传输完成事件 1: 发生传输完成事件
位 12	GF4	0x0	ro	通道 4 全局事件标志 (Global event flag) 0: 未发生传输错误、半传输完成或传输完成事件 1: 发生传输错误、半传输完成或传输完成事件
位 11	DTERRF3	0x0	ro	通道 3 数据传输错误事件标志 (data transfer error event flag) 0: 未发生错误传输事件 1: 发生错误传输事件
位 10	HDTF3	0x0	ro	通道 3 半数据传输事件标志 (half data transfer event flag) 0: 未发生半传输事件 1: 发生半传输事件
位 9	FDTF3	0x0	ro	通道 3 数据传输完成事件标志 (full data transfer event flag) 0: 未发生传输完成事件 1: 发生传输完成事件
位 8	GF3	0x0	ro	通道 3 全局事件标志 (Global event flag) 0: 未发生传输错误、半传输完成或传输完成事件 1: 发生传输错误、半传输完成或传输完成事件
位 7	DTERRF2	0x0	ro	通道 2 数据传输错误事件标志 (data transfer error event flag) 0: 未发生错误传输事件 1: 发生错误传输事件
位 6	HDTF2	0x0	ro	通道 2 半数据传输事件标志 (half data transfer event flag) 0: 未发生半传输事件 1: 发生半传输事件

位 5	FDTF2	0x0	ro	通道 2 数据传输完成事件标志 (full data transfer event flag) 0: 未发生传输完成事件 1: 发生传输完成事件
位 4	GF2	0x0	ro	通道 2 全局事件标志 (Global event flag) 0: 未发生传输错误、半传输完成或传输完成事件 1: 发生传输错误、半传输完成或传输完成事件
位 3	DTERRF1	0x0	ro	通道 1 数据传输错误事件标志 (data transfer error event flag) 0: 未发生错误传输事件 1: 发生错误传输事件
位 2	HDTF1	0x0	ro	通道 1 半数据传输事件标志 (half data transfer event flag) 0: 未发生半传输事件 1: 发生半传输事件
位 1	FDTF1	0x0	ro	通道 1 数据传输完成事件标志 (full data transfer event flag) 0: 未发生传输完成事件 1: 发生传输完成事件
位 0	GF1	0x0	ro	通道 1 全局事件标志 (Global event flag) 0: 未发生传输错误、半传输完成或传输完成事件 1: 发生传输错误、半传输完成或传输完成事件

#### 9.4.2 DMA状态清除寄存器 (DMA\_CLR)

访问：无等待状态，字，半字和字节访问

域	简称	复位值	类型	功能
31: 28	保留	0x0	resd	保持默认值。
位 27	DTERRFC7	0x0	rw1c	清除通道 7 的数据传输错误标志 (data transfer error flag clear) 0: 无效 1: 清除 DMA_STS 寄存器中 DTERRF7 标志
位 26	HDTFC7	0x0	rw1c	清除通道 7 的半数据传输标志 (half data transfer flag clear) 0: 无效 1: 清除 DMA_STS 寄存器中 HDTF7 标志
位 25	FDTFC7	0x0	rw1c	清除通道 7 的数据传输完成标志 (full data transfer flag clear) 0: 无效 1: 清除 DMA_STS 寄存器中 FDTF7 标志
位 24	GFC7	0x0	rw1c	清除通道 7 的全局中断标志 (Global flag clear) 0: 无效 1: 清除 DMA_STS 寄存器中 DTERRF7、HDTF7、FDTF7 和 GF7 标志
位 23	DTERRFC6	0x0	rw1c	清除通道 6 的数据传输错误标志 (data transfer error flag clear) 0: 无效 1: 清除 DMA_STS 寄存器中 DTERRF6 标志
位 22	HDTFC6	0x0	rw1c	清除通道 6 的半数据传输标志 (half data transfer flag clear) 0: 无效 1: 清除 DMA_STS 寄存器中 HDTF6 标志
位 21	FDTFC6	0x0	rw1c	清除通道 6 的数据传输完成标志 (full data transfer flag clear) 0: 无效 1: 清除 DMA_STS 寄存器中 FDTF6 标志

位 20	GFC6	0x0	rw1c	清除通道 6 的全局中断标志 (Global flag clear) 0: 无效 1: 清除 DMA_STS 寄存器中 DTERRF6、HDTF6 FDTF6 和 GF6 标志
位 19	DTERRFC5	0x0	rw1c	清除通道 5 的数据传输错误标志 (data transfer error flag clear) 0: 无效 1: 清除 DMA_STS 寄存器中 DTERRF5 标志
位 18	HDTFC5	0x0	rw1c	清除通道 5 的半数据传输标志 (half data transfer flag clear) 0: 无效 1: 清除 DMA_STS 寄存器中 HDTF5 标志
位 17	FDTFC5	0x0	rw1c	清除通道 5 的数据传输完成标志 (full data transfer flag clear) 0: 无效 1: 清除 DMA_STS 寄存器中 FDTF5 标志
位 16	GFC5	0x0	rw1c	清除通道 5 的全局中断标志 (Global flag clear) 0: 无效 1: 清除 DMA_STS 寄存器中 DTERRF5、HDTF5 FDTF5 和 GF5 标志
位 15	DTERRFC4	0x0	rw1c	清除通道 4 的数据传输错误标志 (data transfer error flag clear) 0: 无效 1: 清除 DMA_STS 寄存器中 DTERRF4 标志
位 14	HDTFC4	0x0	rw1c	清除通道 4 的半数据传输标志 (half data transfer flag clear) 0: 无效 1: 清除 DMA_STS 寄存器中 HDTF4 标志
位 13	FDTFC4	0x0	rw1c	清除通道 4 的数据传输完成标志 (full data transfer flag clear) 0: 无效 1: 清除 DMA_STS 寄存器中 FDTF4 标志
位 12	GFC4	0x0	rw1c	清除通道 4 的全局中断标志 (Global flag clear) 0: 无效 1: 清除 DMA_STS 寄存器中 DTERRF4、HDTF4 FDTF4 和 GF4 标志
位 11	DTERRFC3	0x0	rw1c	清除通道 7 的数据传输错误标志 (data transfer error flag clear) 0: 无效 1: 清除 DMA_STS 寄存器中 DTERRF7 标志
位 10	HDTFC3	0x0	rw1c	清除通道 7 的半数据传输标志 (half data transfer flag clear) 0: 无效 1: 清除 DMA_STS 寄存器中 HDTF7 标志
位 9	FDTFC3	0x0	rw1c	清除通道 3 的数据传输完成标志 (full data transfer flag clear) 0: 无效 1: 清除 DMA_STS 寄存器中 FDTF3 标志
位 8	GFC3	0x0	rw1c	清除通道 3 的全局中断标志 (Global flag clear) 0: 无效 1: 清除 DMA_STS 寄存器中 DTERRF3、HDTF3 FDTF3 和 GF3 标志
位 7	DTERRFC2	0x0	rw1c	清除通道 2 的数据传输错误标志 (data transfer error flag clear) 0: 无效 1: 清除 DMA_STS 寄存器中 DTERRF2 标志

位 6	HDTFC2	0x0	rw1c	清除通道 2 的半数据传输标志 (half data transfer flag clear) 0: 无效 1: 清除 DMA_STS 寄存器中 HDTF2 标志
位 5	FDTFC2	0x0	rw1c	清除通道 2 的数据传输完成标志 (full data transfer flag clear) 0: 无效 1: 清除 DMA_STS 寄存器中 FDTF2 标志
位 4	GFC2	0x0	rw1c	清除通道 2 的全局中断标志 (Global flag clear) 0: 无效 1: 清除 DMA_STS 寄存器中 DTERRF2、HDTF2 FDTF2 和 GF2 标志
位 3	DTERRFC1	0x0	rw1c	清除通道 1 的数据传输错误标志 (data transfer error flag clear) 0: 无效 1: 清除 DMA_STS 寄存器中 DTERRF1 标志
位 2	HDTFC1	0x0	rw1c	清除通道 1 的半数据传输标志 (half data transfer flag clear) 0: 无效 1: 清除 DMA_STS 寄存器中 HDTF1 标志
位 1	FDTFC1	0x0	rw1c	清除通道 1 的数据传输完成标志 (full data transfer flag clear) 0: 无效 1: 清除 DMA_STS 寄存器中 FDTF1 标志
位 0	GFC1	0x0	rw1c	清除通道 1 的全局中断标志 (Global flag clear) 0: 无效 1: 清除 DMA_STS 寄存器中 DTERRF1、HDTF1 FDTF1 和 GF1 标志

### 9.4.3 DMA通道x配置寄存器 (DMA\_CxCTRL) (x = 1…7)

访问：无等待状态，字，半字和字节访问

域	简称	复位值	类型	功能
位 31: 15	保留	0x00000	resd	保持默认值。
位 14	M2M	0x0	rw	存储器到存储器模式 (Memory to memory mode) 0: 关闭 1: 开启
位 13: 12	CHPL	0x0	rw	通道优先级 (Channel preemptive level) 00: 低优先级 01: 中优先级 10: 高优先级 11: 最高优先级
位 11: 10	MWIDTH	0x0	rw	存储器数据宽度 (Memory data bit width) 00: 8 bit 位宽 01: 16 bit 位宽 10: 32 bit 位宽 11: 保留
位 9: 8	PWIDTH	0x0	rw	外设数据宽度 (Peripheral data bit width) 00: 8 bit 位宽 01: 16 bit 位宽 10: 32 bit 位宽 11: 保留
位 7	MINCM	0x0	rw	存储器地址递增模式 (Memory address increment mode) 0: 关闭 1: 开启

位 6	PINCM	0x0	rw	外设地址递增模式 (Peripheral address increment mode) 0: 关闭 1: 开启
位 5	LM	0x0	rw	循环模式 (Loop mode) 0: 关闭 1: 开启
位 4	DTD	0x0	rw	数据传输方向 (Data transfer direction) 0: 外设为源 1: 存储器为源
位 3	DTERRIEN	0x0	rw	允许数据传输错误中断 (data transfer error interrupt enable) 0: 禁止数据传输错误中断 1: 允许数据传输错误中断
位 2	HDTIEN	0x0	rw	允许半数据传输中断 (half data transfer interrupt enable) 0: 禁止半数据传输中断
位 1	FDTIEN	0x0	rw	允许数据传输完成中断 (full data transfer interrupt enable) 0: 禁止数据传输完成中断 1: 允许数据传输完成中断
位 0	CHEN	0x0	rw	通道使能 (Channel enable) 0: 关闭 1: 开启

#### 9.4.4 DMA通道x数据传输量寄存器 (DMA\_CxDTCNT) (x = 1…7)

访问: 无等待状态, 字, 半字和字节访问

域	简称	复位值	类型	功能
位 31: 16	保留	0x0000	resd	保持默认值。
位 15: 0	CNT	0x0000	rw	DMA 通道数据传输个数 (Number of data to transfer) DMA 通道传输数据个数范围为 0x0~0xFFFF, 在更改 DMA 通道传输数据个数时需要确保对应通道的 CHEN 位为 0, 否则无法写入; DMA 控制器每传输完一笔数据, 此值会硬件减 1。 注: 此寄存器为传输数据个数, 不是传输数据量大小; 传输数据量大小需要根据数据宽度换算得到。

#### 9.4.5 DMA通道x外设地址寄存器 (DMA\_CxPADDR) (x = 1…7)

访问: 无等待状态, 字, 半字和字节访问

域	简称	复位值	类型	功能
位 31: 0	PADDR	0x0000 0000	rw	外设端基址 (Peripheral base address) 外设数据寄存器的地址, 作为数据传输的源或目标。 注: 确保对应通道的 CHEN 位为 0, 否则无法写入。

#### 9.4.6 DMA通道x存储器地址寄存器 (DMA\_CxMADDR) (x = 1…7)

访问: 无等待状态, 字, 半字和字节访问

域	简称	复位值	类型	功能
位 31: 0	MADDR	0x0000 0000	rw	储存器端基址 (Memory base address) 储存器地址作为数据传输的源或目标。 注: 确保对应通道的 CHEN 位为 0, 否则无法写入。

### 9.4.7 通道来源寄存器0 (DMA\_SRC\_SEL0)

访问：无等待状态，字，半字和字节访问

域	简称	复位值	类型	功能
位 31: 24	CH4_SRC	0x00	rw	CH4 来源的选择位 (CH4 source select) 当 DMA_FLEX_EN=1 时, 由 CH4_SRC 选择通道 4 来源, 详见 <a href="#">9.3.8 DMA 弹性请求映射</a>
位 23: 16	CH3_SRC	0x00	rw	CH3 来源的选择位 (CH3 source select) 当 DMA_FLEX_EN=1 时, 由 CH3_SRC 选择通道 3 来源, 详见 <a href="#">9.3.8 DMA 弹性请求映射</a>
位 15: 8	CH2_SRC	0x00	rw	CH2 来源的选择位 (CH2 source select) 当 DMA_FLEX_EN=1 时, 由 CH2_SRC 选择通道 2 来源, 详见 <a href="#">9.3.8 DMA 弹性请求映射</a>
位 7: 0	CH1_SRC	0x00	rw	CH1 来源的选择位 (CH1 source select) 当 DMA_FLEX_EN=1 时, 由 CH1_SRC 选择通道 1 来源, 详见 <a href="#">9.3.8 DMA 弹性请求映射</a>

### 9.4.8 通道来源寄存器1 (DMA\_SRC\_SEL1)

访问：无等待状态，字，半字和字节访问

域	简称	复位值	类型	功能
位 31: 25	保留	0x00	resd	保持默认值。
位 24	DMA_FLEX_EN:	0x0	rw	DMA 请求映像模式选择位。 (DMA flexible mapping enable) 0: DMA 请求映像模式为固定模式 1: DMA 请求映像模式为弹性模式
位 23: 16	CH7_SRC	0x00	rw	CH7 来源的选择位 (CH7 source select) 当 DMA_FLEX_EN=1 时, 由 CH7_SRC 选择通道 7 来源, 详见 <a href="#">9.3.8 DMA 弹性请求映射</a>
位 15: 8	CH6_SRC	0x00	rw	CH6 来源的选择位 (CH6 source select) 当 DMA_FLEX_EN=1 时, 由 CH6_SRC 选择通道 6 来源, 详见 <a href="#">9.3.8 DMA 弹性请求映射</a>
位 7: 0	CH5_SRC	0x00	rw	CH5 来源的选择位 (CH5 source select) 当 DMA_FLEX_EN=1 时, 由 CH5_SRC 选择通道 5 来源, 详见 <a href="#">9.3.8 DMA 弹性请求映射</a>

# 10 CRC 计算单元 (CRC)

## 10.1 CRC介绍

CRC计算单元是一个独立的具备CRC计算功能的外设，CRC计算单元采用CRC32标准。用户可以通过软件编程配置控制寄存器(CRC\_CTRL)选择是否进行输入数据翻转(全字翻转, REVOD=1)或输出数据翻转(字节翻转, REVID=01; 半字翻转, REVID=10; 全字翻转, REVID=11), CRC计算单元还提供初始化功能, 每次RESET操作后, CRC计算单元会将CRC\_IDT中的值搬入数据寄存器(CRC\_DT)。

用户通过写和读数据寄存器(CRC\_DT)的方式, 写入想要进行计算的值, 读出计算的结果, 注意每次的CRC计算结果是前一次计算结果与当前待计算值的组合。

### CRC主要特性:

- 采用CRC-32标准
- 一次CRC计算需要4个HCLK
- 输入输出数据格式可翻转
- 待计算值的写入和计算结果的读出都通过写和读数据寄存器(CRC\_DT)实现
- 配置初始化寄存器(CRC\_IDT)写入初始化值, 在每次CRC复位后该值会加载到数据寄存器(CRC\_DT)

## 10.2 CRC寄存器

除CRC\_DT可以用字节(8位)、半字(16位)或字(32位)的方式操作之外, 其他寄存器必须以字(32位)的方式操作。

表 10-1 CRC计算单元寄存器映像

寄存器简称	基址偏移量	复位值
CRC_DT	0x00	0xFFFF FFFF
CRC_CDT	0x04	0x0000 0000
CRC_CTRL	0x08	0x0000 0000
CRC_IDT	0x10	0xFFFF FFFF

### 10.2.1 数据寄存器 (CRC\_DT)

域	简称	复位值	类型	功能
位 31: 0	DT	0xFFFF FFFF	rw	数据寄存器位(Data value) 写入CRC计算器的新数据时, 作为输入寄存器读取时返回CRC计算的结果。

### 10.2.2 通用数据寄存器 (CRC\_CDT)

域	简称	复位值	类型	功能
位 31: 8	保留	0x000000	resd	保持默认值。
位 7: 0	CDT	0x00	rw	通用8位数据寄存器位(Common 8-bit data value) 可用于临时存放1字节的数据。寄存器CRC_CTRL的RST位产生的CRC复位对本寄存器没有影响。

### 10.2.3 控制寄存器 (CRC\_CTRL)

域	简称	复位值	类型	功能
位 31: 8	保留	0x000000	resd	保持默认值。
位 7	REVOD	0x0	resd	输出数据翻转(Reverse output data) 由软件置起或清零。该位控制是否翻转输出数据。 0: 不翻转; 1: 全字翻转。
位 6: 5	REVID	0x0	rw	输入数据翻转(Reverse input data) 由软件置起或清零。该位控制如何翻转输入数据。

				00: 不翻转; 01: 字节翻转; 10: 半字翻转; 11: 全字翻转。
位 4: 1	保留	0x0	resd	保持默认值。
位 0	RST	0x0	rw	RESET 位 (Reset CRC calculation unit) 由软件置起, 由硬件自动清零。复位 CRC 计算单元, 设置数据寄存器为 0xFFFF FFFF。 0: 无作用; 1: 复位。

#### 10.2.4 初始化寄存器 (CRC\_IDT)

域	简称	复位值	类型	功能
位 31: 0	IDT	0xFFFF FFFF	rw	初始化数据寄存器 (Initial data value) 当 CRC_CTRL 寄存器的 RST 位产生的 CRC 复位时, 初始化寄存器中的数值将作为 CRC_DT 寄存器的初始值写入。

# 11 I<sup>2</sup>C 接口

## 11.1 I<sup>2</sup>C 简介

I<sup>2</sup>C 总线接口处理微控制器和串行 I<sup>2</sup>C 总线之间的通信，支持主机和从机模式，最大通信速度为 400kbit/s。

## 11.2 I<sup>2</sup>C 主要特点

- I<sup>2</sup>C 总线
  - 主机和从机模式
  - 多主机功能
  - 标准模式(100kHz)和快速模式(400kHz)
  - 7-bit 和 10-bit 地址模式
  - 广播呼叫模式
  - 状态标志
  - 错误标志
  - 时钟延展功能
  - 通讯事件中断
  - 错误中断
- 支持 DMA 传输
- 支持部分 SMBus2.0 协议
  - PEC 产生及检查
  - SMBus 提醒功能
  - ARP(地址解析协议)
  - 超时机制
- PMBus

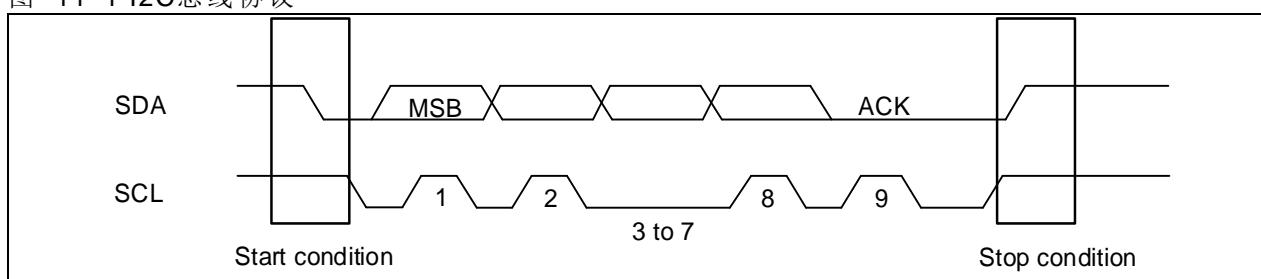
## 11.3 I<sup>2</sup>C 总线特性

I<sup>2</sup>C 总线是由数据线 SDA 和时钟线 SCL 构成，在标准模式下通信速度可达到 100kHz，快速模式下则可以达到 400kHz，一帧数据传输从开始信号开始，在结束信号后停止。在收到开始信号后总线的状态被认为是繁忙的，当收到结束信号后，总线被认为再次空闲。

开始信号：SCL 为高电平时，SDA 由高电平变为低电平。

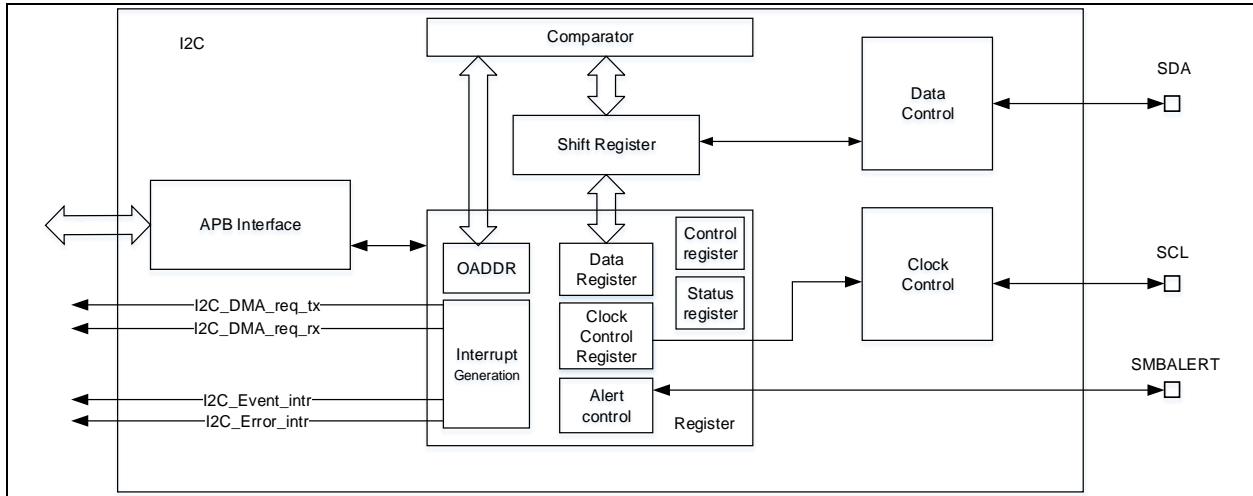
结束信号：SCL 为高电平时，SDA 由低电平变为高电平。

图 11-1 I<sup>2</sup>C 总线协议



## 11.4 I<sup>2</sup>C 接口

I<sup>2</sup>C 接口的功能框图示于下图。

图 11-2 I<sup>2</sup>C 的功能框图

### 1. I<sup>2</sup>C时钟

I<sup>2</sup>C 时钟由 APB1 或者 APB2 提供，可通过设置控制寄存器 2 (I<sup>2</sup>C\_CTRL2) 的 CLKFREQ[7: 0]对 I<sup>2</sup>C 时钟进行分频，在不同模式下对时钟有最低的速率要求，标准模式必须设定至少 2MHz，而快速模式下则是至少要 4MHz。

### 2. 接口工作模式

I<sup>2</sup>C 总线接口可以工作在主机模式与从机模式，并且可以相互切换。默认情况下处于从机模式，当设置了 GENSTART=1 产生了一个起始信号后，I<sup>2</sup>C 总线接口切换成主模式，当数据传输完成之后，也就是结束信号产生了之后，I<sup>2</sup>C 总线接口自动返回为从机模式。

- 主机发送模式
- 主机接收模式
- 从机发送模式
- 从机接收模式

### 3. 通信流程

- 主机模式通信流程：
  1. 产生开始信号
  2. 发送地址
  3. 发送或接收数据
  4. 产生结束信号
  5. 通信结束
- 从机模式通信流程：
  1. 等待地址匹配
  2. 发送或接收数据
  3. 等待结束信号产生
  4. 通信结束

### 4. 地址控制

主机和从机都支持 7 位和 10 位地址模式

#### 从机地址模式：

- 7 位地址模式
  - 单地址模式 ADDR2EN=0：此时只匹配OADDR1
  - 双地址模式 DUALEN=1：此时匹配OADDR1和OADDR2
- 10 位地址模式
  - 只匹配OADDR1

#### 从机特殊地址支持：

- 广播地址 (0b0000000x)：当 GCAEN=1 时该地址启用

- **SMBus** 设备默认地址 (0b1100001x)：当在 **SMBus** 设备模式下该地址启用，该地址用于 **SMBus** 地址解析协议
- **SMBus** 主机默认地址 (0b0001000x)：当在 **SMBus** 主机模式下该地址启用，该地址用于 **SMBus** 主机通知协议
- **SMBus** 提醒地址 (0b0001100x)：当在 **SMBus** 主机模式下并且 **SMBALERT = 1** 下该地址启用，该地址用于 **SMBus** 提醒响应协议  
关于 **SMBus** 协议更详细的信息请参考 **SMBus2.0** 协议。

#### 从机地址匹配流程：

- 收到开始信号
- 匹配地址
- 若地址成功匹配，从机回一个 ACK
- 此时 ADDR7F 置 1，DIRF 指示传输方向
  - 如果DIRF=0从机进入接收模式，开始接收数据
  - 如果DIRF=1从机进入发送模式，开始发送数据

### 5. 时钟延展功能

时钟延展的功能的主要作用是当从机因为某些情况下不能及时的处理数据时，从机通过主动拉低 **SCL** 线，使通信暂停，避免数据丢失，软件可以通过设定控制寄存器 1(I2C\_CTRL1)的 **STRETCH** 位选择是否允许时钟延展。

- 在发送器模式：
  - 允许时钟延展：在发送下个字节（下一个数据的第一个 **SCL** 上升沿）前，若没有新数据写入数据寄存器(I2C\_DT)，则 I<sup>2</sup>C 接口拉低 **SCL** 总线，等待数据写入数据寄存器(I2C\_DT)
  - 不允许时钟延展：发送下个字节（下一个数据的第一个 **SCL** 上升沿）前，若没有新数据写入数据寄存器(I2C\_DT)，则发生欠载错误。
- 在接收器模式
  - 允许时钟延展：数据寄存器(I2C\_DT)内的数据未被读出，然后移位寄存器又接收完一个字节，I<sup>2</sup>C 接口拉低 **SCL** 总线，等待读取数据寄存器(I2C\_DT)
  - 不允许时钟延展：数据寄存器(I2C\_DT)内的数据未被读出，然后移位寄存器又接收完一个字节，此时如果又接收到一个数据，则发生过载错误。

## 11.4.1 I<sup>2</sup>C从机通信流程

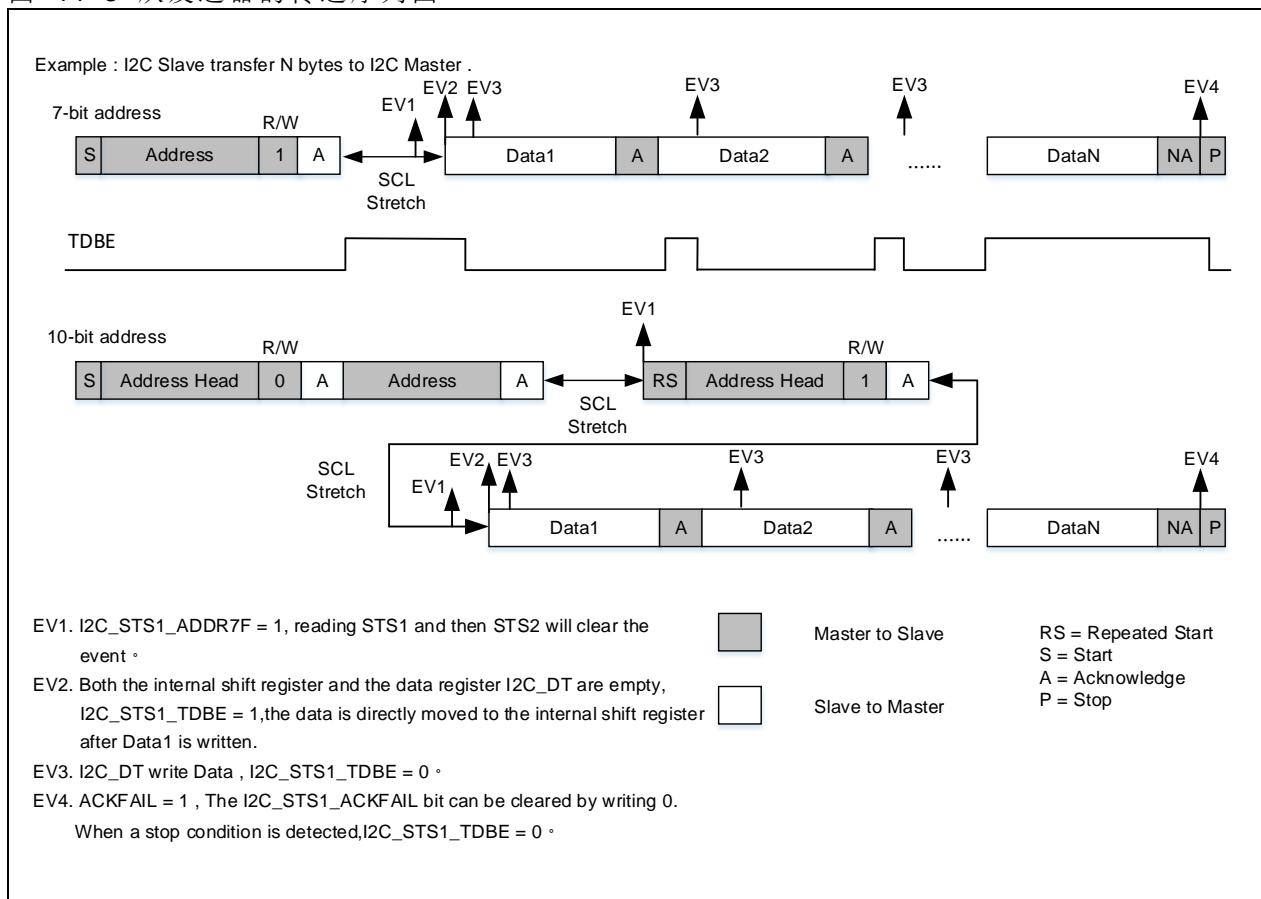
### 初始化

使能 I<sup>2</sup>C 外设时钟并配置控制寄存器 2 (I2C\_CTRL2) 中时钟相关寄存器确保正确的时序，接着等待 I<sup>2</sup>C 主机发送开始讯号。

### 发送器

从机发送数据主要有以下操作流程，初始化后软件可以参照以下步骤进行操作：

图 11-3 从发送器的传送序列图



### 7位地址模式：

1. 等待主机发送地址
2. EV1: 成功匹配到地址 (ADDR7F=1)，从机将SCL总线拉低，软件先读取STS1，再读取STS2 清除ADDR7F位，此时进入发送阶段，DT寄存器和内部移位寄存器皆为空，硬件将TDBE位置1
3. EV2: 向DT寄存器写入数据，此时数据会被立即送到移位寄存器并释放SCL总线，此时TDBE仍然为1
4. EV3: 此时DT数据寄存器空，移位寄存器非空，向DT寄存器写入数据，此时TDBE清零
5. EV4: 收到主机发送的ACKFAIL事件，此时ACKFAIL=1，向ACKFAIL写0清除该事件
6. 通信结束

### 10位地址模式：

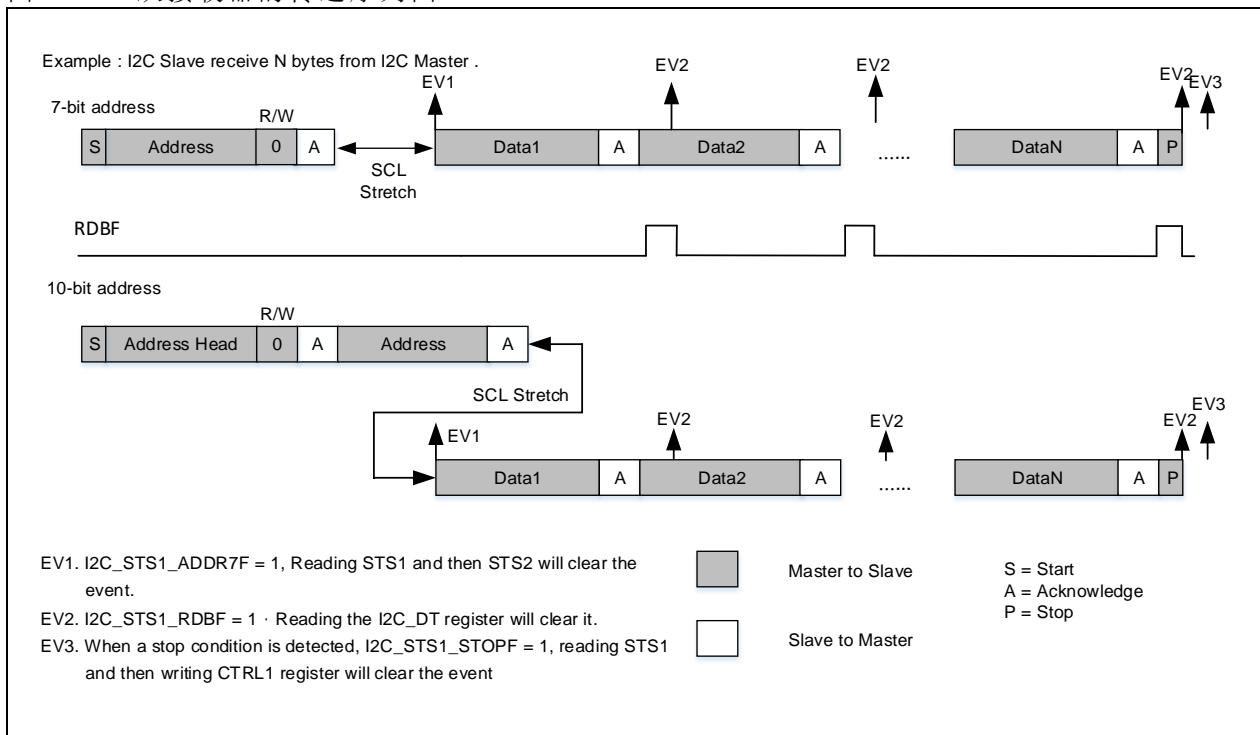
1. 等待主机发送地址
2. EV1: 成功匹配到地址 (ADDR7F=1)，从机将SCL总线拉低，软件先读取STS1，再读取STS2 清除ADDR7F位，等待主机发送重复开始信号
3. EV1: 成功匹配到地址 (ADDR7F=1)，软件先读取STS1，再读取STS2再次清除ADDR7F位，此时进入发送阶段，此时DT寄存器和内部移位寄存器皆为空，硬件将TDBE位置1
4. EV2: 向DT寄存器写入数据，此时数据会被立即送到移位寄存器并释放SCL总线，此时TDBE仍然为1
5. EV3: 此时DT数据寄存器空，移位寄存器非空，向DT寄存器写入数据，此时TDBE清零

6. EV4: 收到主机发送的ACKFAIL事件, 此时ACKFIAL=1, 向ACKFIAL写0清除该事件
7. 通信结束

### 从接收器

从机接收数据主要有以下操作流程, 初始化后软件可以参照以下步骤进行操作 :

图 11-4 从接收器的传送序列图



### 7位地址模式:

1. 等待主机发送地址
2. EV1: 成功匹配到地址 (ADDR7F=1), 从机将SCL总线拉低, 软件可通过读取STS1在读取STS2清除ADDR7F位, 此时从机释放SCL总线, 进入接收阶段
3. 从机内部移位寄存器接收来自总在线的数据, 并存入DT寄存器
4. EV2: 在接收到字节后, RDBF位被置1, 软件读取数据寄存器(I2C\_DT), RDBF位被清0
5. EV3: 收到主机发送的结束信号, STOPF=1, 软件读取STS1, 再写CTRL1寄存器清除该事件
6. 通信结束

### 10位地址模式:

1. 等待主机发送地址
2. EV1: 成功匹配到地址 (ADDR7F=1), 从机将SCL总线拉低, 软件先读取STS1, 再读取STS2清除ADDR7F位, 此时从机释放SCL总线进入接收阶段
3. 从机内部移位寄存器接收来自总在线的数据, 并存入DT寄存器
4. EV2: 在接收到字节后, RDBF位被置1, 软件读取数据寄存器(I2C\_DT), RDBF位被清0
5. EV3: 收到主机发送的结束信号, STOPF=1, 软件读取STS1, 再写CTRL1寄存器清除该事件
6. 通信结束

## 11.4.2 I<sup>2</sup>C主机通信流程

### 主机模式初始化

1. 设置输入时钟以产生正确的时序 (控制寄存器2 (I2C\_CTRL2) 中的CLKFREQ位) ;
2. 设置I<sup>2</sup>C的通信速度 (时钟控制寄存器(I2C\_CLKCTRL)) ;
3. 设置总线最大上升时间 (I2C\_TMRISE寄存器) ;
4. 设置控制寄存器1 (I2C\_CTRL1) ;
5. 启动外设, 若设置GENSTART位在启动外设时会在总在线产生开始信号, 设备会进入主机模式从机地址发送

从机地址可分为 7 位和 10 位地址模式，主机会根据送出的地址最低位决定进入发送器模式或是接收器模式。

- 7 位地址模式：

发送模式：发送的地址最低位为 0 时，进入发送器模式；

接收模式：发送的地址最低位为 1 时，进入接收器模式。

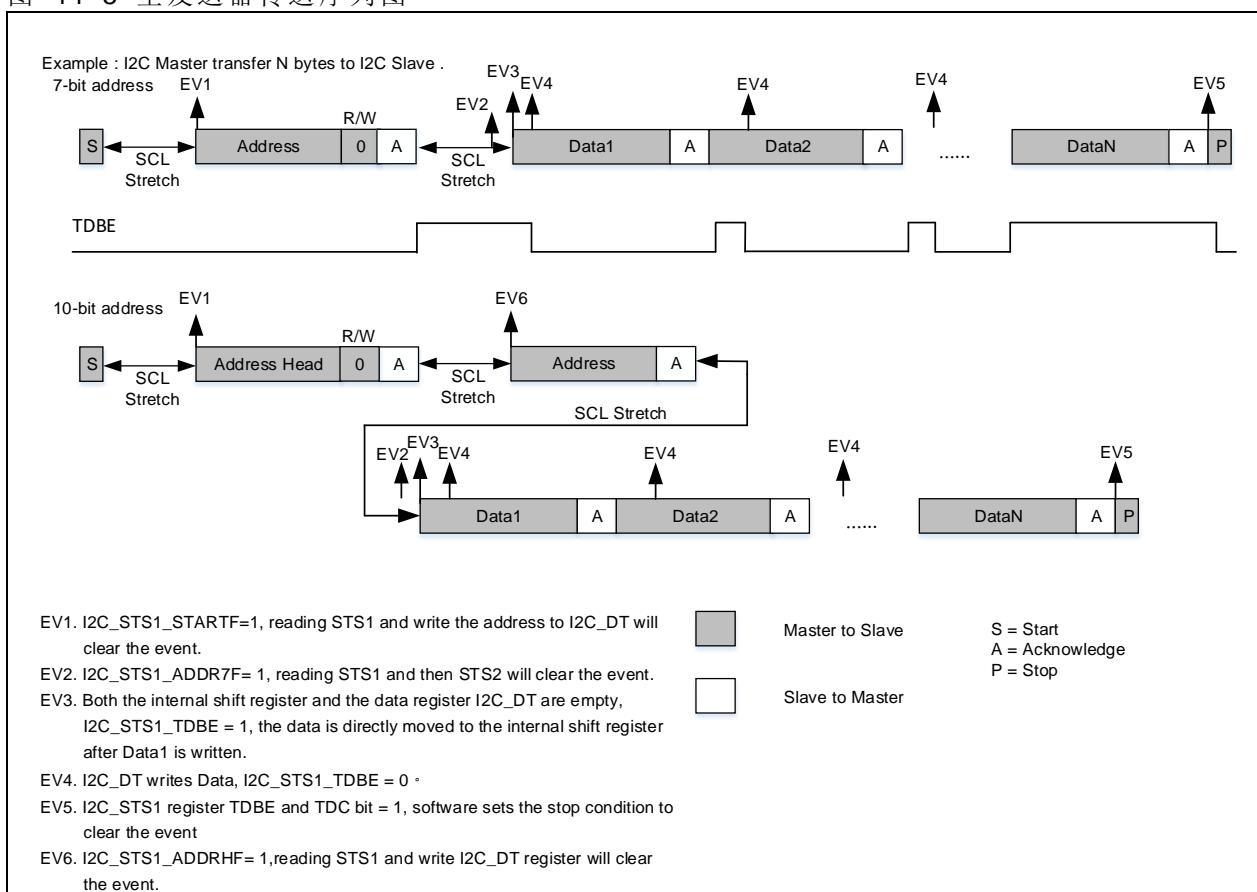
- 10 位地址模式：

发送模式：先发送从机地址头 0b11110xx0 (xx 为地址[9: 8])，再发送从机地址[7: 0]，主机进入发送器模式；

接收模式：先发送从机地址头 0b11110xx0 (xx 为地址[9: 8])，再发送从机地址[7: 0]，再发送从机地址头 0b11110xx1 (xx 为地址[9: 8])，主机进入接收器模式。

### 主发送器

图 11-5 主发送器传送序列图



- 7 位地址模式：

1. 发送开始信号 (GENSTART=1)
2. EV1: 开始信号产生完成 (STARTF=1)，软件先读取STS1，然后将地址写入DT寄存器
3. EV2: 成功匹配到地址 (ADDR7F=1)，软件先读取STS1，再读取STS2清除ADDR7F位，此时主机进入发送阶段，DT寄存器和内部移位寄存器皆为空，硬件将TDBE位置1
4. EV3: 向DT寄存器写入数据，此时数据会被立即送到移位寄存器并释放SCL总线，此时TDBE仍然为1
5. EV4: 此时DT数据寄存器空，移位寄存器非空，向DT寄存器写入数据，此时TDBE清零
6. TDBE位在倒数第二个字节发送完成后置起
7. EV5: TDC=1，字节发送结束，主机发送结束信号 (STOPF=1)，硬件自动清除TDBE位和TDC位
8. 通信结束

- 10 位地址模式：

1. 发送开始信号 (GENSTART=1)
2. EV1: 开始信号产生完成，STARTF=1，软件先读取STS1，然后将地址写入DT寄存器
3. EV6: 10位地址头序列已发送，软件可通过读取STS1再写入DT寄存器清除ADDRHF位

4. EV2: 成功匹配到地址 (ADDR7F=1), 软件先读取STS1, 再读取STS2清除ADDR7F位, 此时主机进入发送阶段, DT寄存器和内部移位寄存器皆为空, 硬件将TDBE位置1
5. EV3: 向DT寄存器写入数据, 此时数据会被立即送到移位寄存器并释放SCL总线, 此时TDBE仍然为1
6. EV4: 此时DT数据寄存器空, 移位寄存器非空, 向DT寄存器写入数据, 此时TDBE清零
7. TDBE位在倒数第二个字节发送完成后置起
8. EV5: TDC=1, 字节发送结束, 主机发送结束信号 (STOPF=1), 硬件自动清除TDBE位和TDC位
9. 通信结束

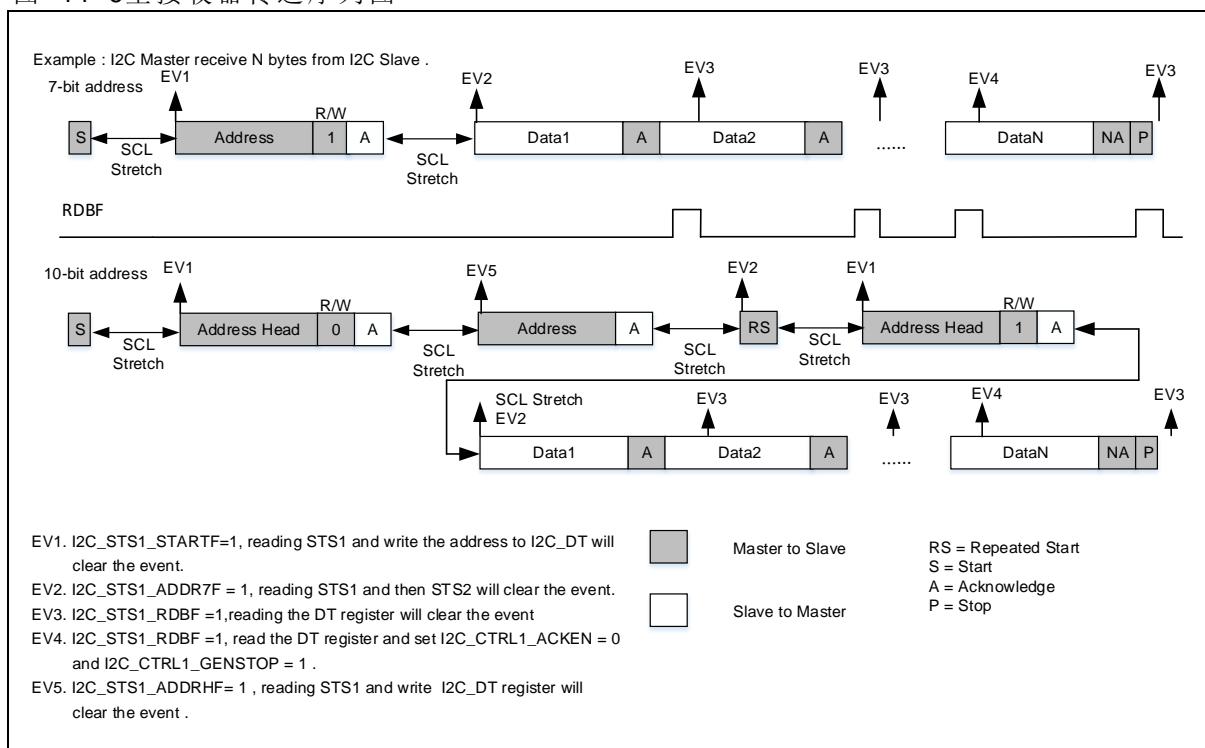
### 主接收器

主机接收数据可依 I<sup>2</sup>C 中断优先级分为几种情况:

#### 1. I<sup>2</sup>C中断为最高优先级

- 在读倒数第二个字节后需清除控制寄存器 1 (I2C\_CTRL1) 的 ACKEN 位并设置同一寄存器的 GENSTOP 位以产生结束信号。
- 若只接收一个字节时, 需在清除 ADDR7F 标志后设置控制寄存器 1 (I2C\_CTRL1) 的 ACKEN 和 GENSTOP 位。
- 接收到字节后硬件会将 I2C\_STS1\_RDBF 位置 1, 在软件读数据寄存器(I2C\_DT)后会被清 0。

图 11-6 主接收器传送序列图



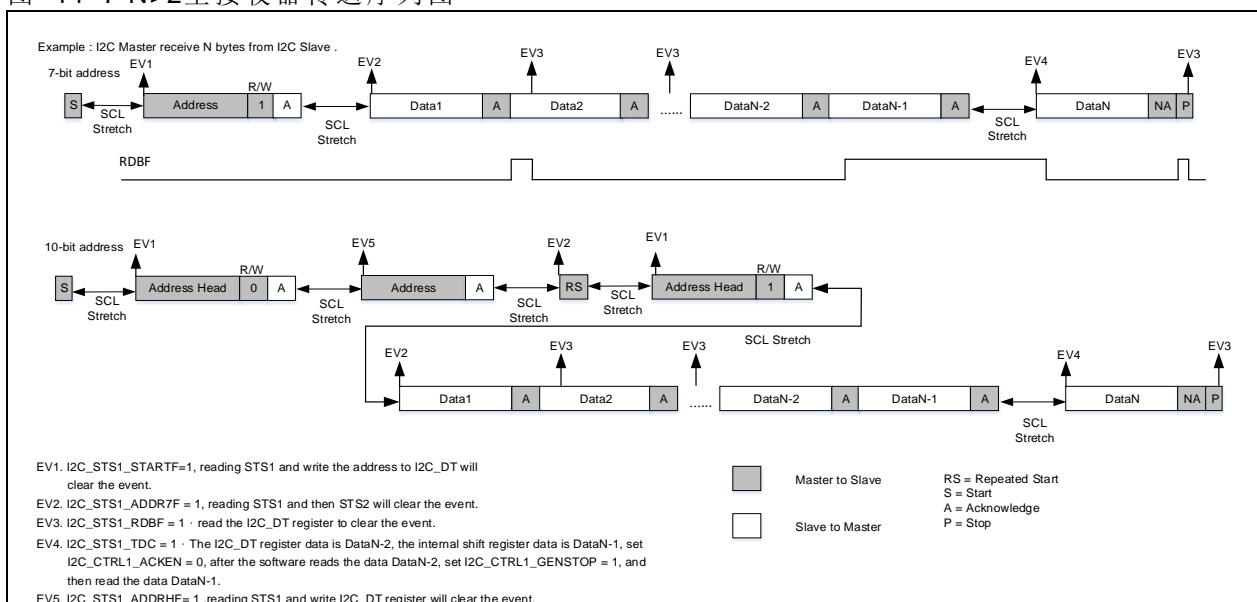
- 7 位地址模式:
  1. 发送开始信号 (GENSTART=1)
  2. EV1: 开始信号产生完成 (STARTF=1), 软件先读取STS1, 然后将地址写入DT寄存器
  3. EV2: 成功匹配到地址 (ADDR7F=1), 软件先读取STS1, 再读取STS2清除ADDR7F位, 此时主机进入接收阶段
  4. EV3: 在接收到字节后, RDBF位被置1, 软件读取数据寄存器(I2C\_DT), RDBF位被清0
  5. EV4: 接收完倒数第二个字节后, 软件需立即将ACKEN位清0, GENSTOP位置1
  6. EV3: 在接收到字节后, RDBF位被置1, 软件读取数据寄存器(I2C\_DT), RDBF位被清0
  7. 通信结束
- 10 位地址模式:
  1. 发送开始信号 (GENSTART=1)
  2. EV1: 开始信号产生完成 (STARTF=1), 软件先读取STS1, 然后将地址写入DT寄存器
  3. EV5: 10位地址头序列已发送, 软件可通过读取STS1再写入DT寄存器清除ADDRHF位

4. EV2: 成功匹配到地址 (ADDR7F=1), 软件先读取STS1, 再读取STS2清除ADDR7F位, 主机发送重复开始信号 (GENSTART=1)
5. EV1: 重复开始信号产生完成 (STARTF=1), 软件先读取STS1, 然后将地址写入DT寄存器
6. EV2: 成功匹配到地址 (ADDR7F=1), 软件先读取STS1, 再读取STS2清除ADDR7F位, 此时主机进入接收阶段
7. EV3: 在接收到字节后, RDBF位被置1, 软件读取数据寄存器(I2C\_DT), RDBF位被清0
8. EV4: 接收完倒数第二个字节后, 软件需立即将ACKEN位清0, GENSTOP位置1
9. EV3: 在接收到字节后, RDBF位被置1, 软件读取数据寄存器(I2C\_DT), RDBF位被清0
10. 通信结束

## 2. I<sup>2</sup>C中断非最高优先级且要接收的字节数大于2

- 在接收到倒数第三个字节(N-2)时不进行读取, 待收到倒数第二个字节(N-1)时, 清除I2C\_CTRL1寄存器的ACKEN位, 接着读取倒数第三个字节(N-2), 设置I2C\_CTRL1寄存器的GENSTOP位后读取倒数第二个字节(N-1), 接着总线开始接收最后字节。

图 11-7 N>2主接收器传送序列图

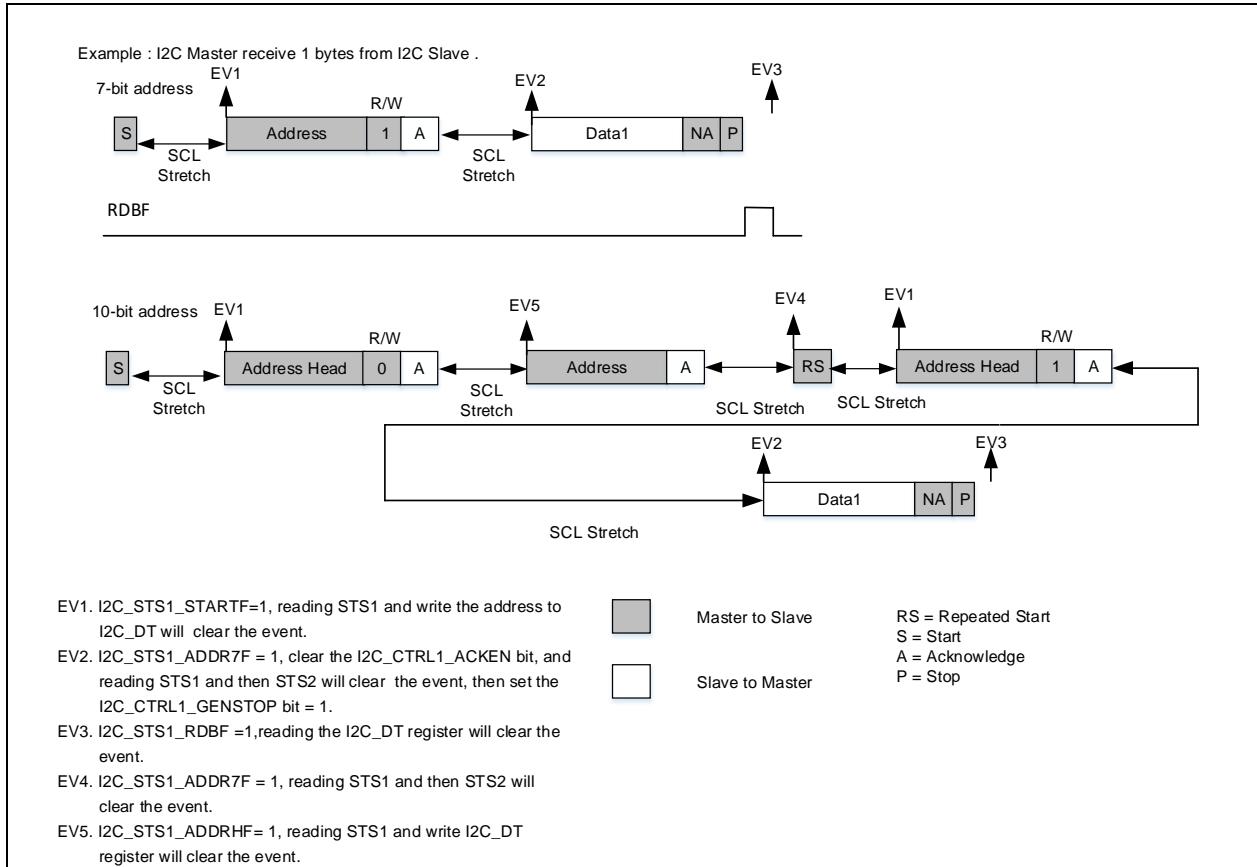


- 7位地址模式:
  1. 发送开始信号 (GENSTART=1)
  2. EV1: 开始信号产生完成 (STARTF=1), 软件先读取STS1, 然后将地址写入DT寄存器
  3. EV2: 成功匹配到地址 (ADDR7F=1), 软件先读取STS1, 再读取STS2清除ADDR7F位, 此时主机进入接收阶段
  4. EV3: 在接收到字节后, RDBF位被置1, 软件读取I2C\_DT寄存器, RDBF位被清0
  5. EV4: TDC=1, 数据寄存器(I2C\_DT)内容为N-2, 移位寄存器内容为数据N-1, 软件将ACKEN位置0并读取读数据N-2, 接着设置GENSTOP=1, 然后读数据N-1
  6. EV3: 在接收到字节后, RDBF位被置1, 软件读取数据寄存器(I2C\_DT), RDBF位被清0
  7. 通信结束
- 10位地址模式:
  1. 发送开始信号 (GENSTART=1)
  2. EV1: 开始信号产生完成 (STARTF=1), 软件先读取STS1, 然后将地址写入DT寄存器
  3. EV5: 10位地址头序列已发送, 软件可通过读取STS1再写入DT寄存器清除ADDRHF位
  4. EV2: 成功匹配到地址 (ADDR7F=1), 软件先读取STS1, 再读取STS2清除ADDR7F位, 主机发送重复开始信号 (GENSTART=1)
  5. EV1: 重复开始信号产生完成, STARTF=1, 软件先读取STS1, 然后将地址写入DT寄存器
  6. EV2: 成功匹配到地址 (ADDR7F=1), 软件先读取STS1, 再读取STS2清除ADDR7F位, 此时主机进入接收阶段
  7. EV3: 在接收到字节后, RDBF位被置1, 软件读取数据寄存器(I2C\_DT), RDBF位被清0
  8. EV4: TDC=1, 数据寄存器(I2C\_DT)内容为N-2, 移位寄存器内容为数据N-1, 软件将ACKEN

位置0并读取读数据N-2，接着设置GENSTOP=1，然后读数据N-1

9. EV3: 在接收到字节后，RDBF位被置1，软件读取数据寄存器(I2C\_DT)，RDBF位被清0
  10. 通信结束
3. **I<sup>2</sup>C中断非最高优先级且要接收的字节数等于2**
- 在接收数据前设置控制寄存器1(I2C\_CTRL1)的MACKCTRL位，待地址匹配后，先清除ACKEN位，后清除ADDR7F位，待TDC位置1后设置控制寄存器1(I2C\_CTRL1)的GENSTOP位，接着读取DT寄存器。

图 11-8 N=2主接收器传送序列图



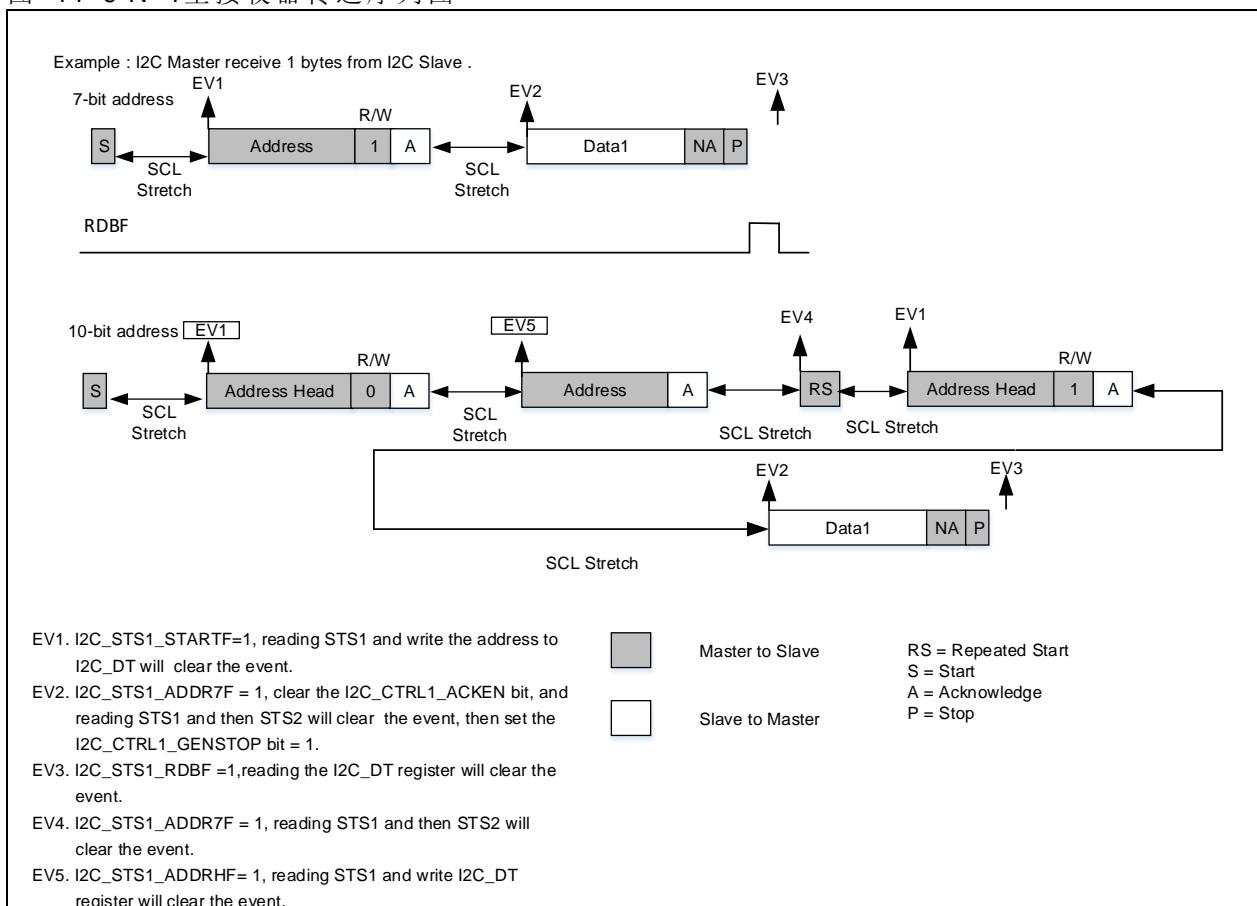
- 7位地址模式：
  1. 设置I2C\_CTRL1寄存器的MACKCTRL=1
  2. 发送开始信号(GENSTART=1)
  3. EV1: 开始信号产生完成(STARTF=1)，软件先读取STS1，然后将地址写入DT寄存器
  4. EV2: 成功匹配到地址(ADDR7F=1)，首先清除ACKEN位，然后先读取STS1，再读取STS2清除ADDR7F位，此时主机进入接收阶段
  5. EV2: TDC=1，接着设置GENSTOP=1然后读数据寄存器(I2C\_DT)两次
  6. 通信结束
- 10位地址模式：
  1. 设置I2C\_CTRL1寄存器的MACKCTRL=1
  2. 发送开始信号(GENSTART=1)
  3. EV1: 开始信号产生完成(STARTF=1)，软件先读取STS1，然后将地址写入DT寄存器
  4. EV4: 10位地址头序列已发送，软件可通过读取STS1再写入DT寄存器清除ADDRHF位
  5. EV2: 成功匹配到地址(ADDR7F=1)，软件先读取STS1，再读取STS2清除ADDR7F位，主机发送重复开始信号(GENSTART=1)
  6. EV1: 重复开始信号产生完成，STARTF=1，软件先读取STS1，然后将地址写入DT寄存器
  7. EV2: 成功匹配到地址(ADDR7F=1)，首先清除ACKEN位，然后先读取STS1，再读取STS2清除ADDR7F位，此时主机进入接收阶段
  8. EV3: TDC=1，接着设置GENSTOP=1然后读数据寄存器(I2C\_DT)两次
  9. 通信结束

#### 4. I<sup>2</sup>C中断非最高优先级且要接收的字节数等于1

- 待地址匹配后，先清除 ACKEN 位，后清除 ADDR7F 位，接着设置控制寄存器 1 (I2C\_CTRL1) 的 GENSTOP 位，待 RDBF 位置 1 后读取 DT 寄存器内字节。

主机接收数据主要有以下操作流程，主机模式初始化后软件可以参照以下步骤进行操作：

图 11-9 N=1 主接收器传送序列图



- 7 位地址模式：

1. 发送开始信号 (GENSTART=1)
2. EV1：开始信号产生完成 (STARTF=1)，软件先读取STS1，然后将地址写入DT寄存器
3. EV2：成功匹配到地址 (ADDR7F=1)，首先清除ACKEN位，然后先读取STS1，再读取STS2 清除ADDR7F位，接着设置GENSTOP=1，此时主机进入接收阶段
4. EV3：RDBF=1，读取I2C\_DT寄存器，RDBF位被清0
5. 通信结束

- 10 位地址模式：

1. 发送开始信号 (GENSTART=1)
2. EV1：开始信号产生完成 (STARTF=1)，软件先读取STS1，然后将地址写入DT寄存器
3. EV5：10位地址头序列已发送，软件可通过读取STS1再写入DT寄存器清除ADDRHF位
4. EV4：成功匹配到地址 (ADDR7F=1)，软件先读取STS1，再读取STS2清除ADDR7F位，机发送重复开始信号 (GENSTART=1)
5. EV1：重复开始信号产生完成，STARTF=1，软件先读取STS1，然后将地址写入DT寄存器
6. EV2：成功匹配到地址 (ADDR7F=1)，首先清除ACKEN位，然后先读取STS1，再读取STS2 清除ADDR7F位，接着设置GENSTOP=1，此时主机进入接收阶段
7. EV3：RDBF=1，读取数据寄存器(I2C\_DT)，RDBF位被清0
8. 通信结束

### 11.4.3 利用 DMA 传输

I<sup>2</sup>C 可以使用 DMA 进行数据传输，可通过使能传输完成中断位产生中断，当利用 DMA 进行传输时，控制寄存器 2 (I2C\_CTRL2) 的 DATAIEN 位需设为 0，以下说明软件利用 DMA 进行数据传输的操作流程。

**DMA 发送**

1. 设置外设地址 (DMA通道x外设地址寄存器 (DMA\_CxPADDR) = 数据寄存器 (I2C\_DT) 地址)
2. 设置数据存储地址 (DMA通道x存储器地址寄存器 (DMA\_CxMADDR) = 数据存储地址)
3. 设置传输方向为内存到外设 (DMA\_CHCTRL的DTD=1)
4. 设置传输字节数 (DMA通道x数据传输量寄存器 (DMA\_CxDTCNT))
5. 设置DMA通道的其他配置, 例如: 优先级、存储器数据宽度、外设数据宽度、中断等 (DMA\_CHCTRL)
6. 使能DMA通道 (DMA通道x配置寄存器 (DMA\_CxCTRL) 的CHEN=1)。
7. 使能I<sup>2</sup>C DMA请求 (控制寄存器2 (I2C\_CTRL2) 的DMAEN=1), 当状态寄存器1 (I2C\_STS1) 的TDBE位被置1时, DMA将数据从内存地址传输到数据寄存器 (I2C\_DT)
8. 等待传输字节数DMA通道x数据传输量寄存器 (DMA\_CxDTCNT) =0时, 数据传输完成, (可以通过DMA传输完成中断来等待)。
9. 主机发送模式: 等待TDC标志置1, 产生STOP条件, 传输完成。  
从机发送模式: 等待ACKFAIL标志置1, 清除ACKFAIL标志, 传输完成。

**DMA 接收**

1. 设置外设地址 (DMA通道x外设地址寄存器 (DMA\_CxPADDR) = 数据寄存器 (I2C\_DT) 地址)
2. 设置数据存储地址 (DMA通道x存储器地址寄存器 (DMA\_CxMADDR) = 数据存储地址)
3. 设置传输方向为外设到内存 (DMA\_CHCTRL的DTD=0)
4. 设置传输字节数 (DMA通道x数据传输量寄存器 (DMA\_CxDTCNT))
5. 设置DMA通道的其他配置, 例如: 优先级、存储器数据宽度、外设数据宽度、中断等 (DMA\_CHCTRL)
6. 使能DMA通道 (DMA通道x配置寄存器 (DMA\_CxCTRL) 的CHEN=1)。
7. 使能I<sup>2</sup>C DMA请求 (控制寄存器2 (I2C\_CTRL2) 的DMAEN=1), 当状态寄存器1 (I2C\_STS1) 的RDBF位被置1时, DMA将数据从I2C\_DT寄存器传输到数据存储地址。
8. 等待传输字节数DMA\_TCNTx=0时, 数据传输完成, (可以通过DMA传输完成中断来等待)。
9. 主机接收模式: 清除ACKFAIL标志, 产生STOP条件, 传输完成(当传输数据>=2, 且DMAEND=1时, 当数据传输完成了之后 (DMA\_CxDTCNT=0), 将会自动产生一个NACK)。  
从机接收模式: 等待STOPF标志置1, 清除STOPF标志, 传输完成。

#### 11.4.4 SMBus

SMBus 即系统管理总线是一双线制总线, 基于 I<sup>2</sup>C 的操作原理, 系统中各设备之间通过 SMBus 总线传送和接收讯息, 通过 SMBus 总线, 设备可以提供制造商信息, 告诉系统型号, 报告不同类型错误, 接受控制参数等。关于 SMBus 更加详细的信息请参考 SMBus2.0 协议。

**SMBus 和 I<sup>2</sup>C 的差异**

1. SMBus需维持最低 10kHz 以上的运作频率主要为了管理监控, 只要在保持一定传速运作的情况下加入参数, 就可轻松获知总线目前是否处于闲置 (Idle) 中, 省去逐一侦测传输过程中的停断 (STOP) 信号, 或持续保有停断侦测并辅以额外参数侦测, I<sup>2</sup>C则无
2. SMBus 传输速度从最小 10kHz 到最大 100kHz, I<sup>2</sup>C则是无最小传输速度, 根据不同模式有不同的最大传输速度, 分为标准模式(100kHz)和快速模式(400kHz)
3. SMBus 对接口被重制(Reset)后的恢复时间(Timeout)是 35ms, I<sup>2</sup>C则无时间限制

**SMBus 使用流程**

1. 将I<sup>2</sup>C接口设置SMBus模式, 控制寄存器1 (I2C\_CTRL1) 的PERMODE=1

2. 选择SMBus模式:

SMBMODE=1: SMBus主机

SMBMODE=0: SMBus设备

3. 其他配置和I<sup>2</sup>C使用配置一样

各种 SMBus 协议需要由软件来实现, I<sup>2</sup>C 接口只提供了这些协议的地址识别。

**SMBus 地址解析协议(ARP)**

通过 ARP 协议可以给总线上的设备动态的分配一个唯一的新地址, 解决地址冲突问题。关于 ARP 协议更详细的信息请参考 SMBus2.0 协议。

通过使能 ARPEN 位, 可以使能 I<sup>2</sup>C 接口对设备默认地址 (0b1100001x) 的识别, 但是像唯一设备标

识（UDID）以及具体的协议实现过程，需要由软件来处理。

### SMBus 主机通知协议

通过 SMBus 主机通知协议，可让从设备发送数据到主设备，例如从机可以通过此协议通知主机进行 ARP。关于 SMBus 主机通知协议更详细的信息请参考 SMBus2.0 协议。

当使能了 ARP 模式（ARPEN=1）以及在主机模式（SMBMODE=1）下，I<sup>2</sup>C 接口使能对主机默认地址（0b0001000x）的识别。

### SMBus 提醒协议（SMBus Alert）

SMBALERT 是一个可选信号，连接主机和从机的 ALERT 引脚，用于从机通知主机访问从机，SMBALERT 是一个线与信号。关于 SMBus 提醒协议更详细的信息请参考 SMBus2.0 协议。

操作流程如下：

#### SMBus 主机

1. 启用 SMBus 提醒模式（SMBALERT=1）
2. 根据实际需求启用 ALERT 中断
3. 当 ALERT 引脚上产生了提醒事件时（ALERT 引脚电平由高变低）
4. 如果使能了中断，主机将产生 ALERT 中断
5. 主机处理该中断并向从机发送提醒响应地址 ARA（Alert Response Address）地址（0001100x），访问所有设备，获取从机地址，只有那些将 SMBALERT 拉低的设备才会应答
6. 主机通过获取到的从机地址进行下一步操作。

#### SMBus 从机

1. 产生提醒事件，ALERT 引脚由高变低（SMBALERT=1），此时从机响应 ARA（Alert Response Address）地址（0001100x）
2. 根据实际需求启用 ALERT 中断（当收到 ARA 地址时会产生中断）
3. 等待主机通过发送 ARA 地址获取从机地址
4. 上报自己的地址，如果发生了仲裁丢失，继续等待
5. 地址上报成功，释放 ALERT 引脚（SMBALERT=0）

### 包错误校验(PEC)

包错误校验(PEC)用于保证数据传输的正确性和完成性，使用 CRC-8 进行校验，多项式为：

$$C(x) = x^8 + x^2 + x + 1$$

当 PECEN=1 时启动 PEC 计算，检验数据包括地址以及数据，当在仲裁丢失时 PEC 计算会失效。

PEC 发送：

- 正常模式：最后一次 TDBE 事件后设置 PECTRA=1，让 PEC 在最后一个字节后被发送
- DMA 模式：在最后一个字节传输完成后自动发送 PEC，例如：传输的数据为 8 个那么设置 DMA\_TCNTx=8

PEC 接收：

- 正常模式：最后一个 RDBF 事件后设置 PECTRA 位，PECTRA 位必须在接收当前字节的 ACK 脉冲之前被设置
- DMA 模式：接收时会自动把最后一个字节当作 PECVAL 并检查，例如：传输的数据为 8 个那么设置 DMA\_TCNTx=9

在接收模式下，当 PEC 校验失败时，将产生一个 NACK。

## 11.4.5 I<sup>2</sup>C 中断请求

下表列出了所有的 I<sup>2</sup>C 中断请求。

中断事件	事件标志	使能位
已发送起始条件(主机)	STARTF	EVTIEN
地址已发送(主机)或地址匹配(从机)	ADDR7F	
10 位地址头已发送(主机)	ADDRHF	
数据传输完成	TDC	
收到停止条件(从机)	STOPF	

发送缓冲区空	TDBE	EVTIEN 和 DATAIEN  ERRIEN
接收缓冲区非空	RDBF	
SMBus 提醒	ALERTF	
超时错误	TMOUT	
PEC 错误	PECERR	
过载/欠载	OUF	
应答失败	ACKFAIL	
仲裁丢失	ARLOST	
总线错误	BUSERR	

#### 11.4.6 I<sup>2</sup>C 调试模式

当微控制器进入调试模式 (Cortex<sup>TM</sup>-M4 核心处于停止状态) 时，根据 DEBUG 模块中的 I2Cx\_SMBUS\_TIMEOUT 配置位，SMBUS 超时控制或者继续正常工作或者可以停止。

### 11.5 I<sup>2</sup>C 寄存器描述

必须以字（32 位）的方式操作这些外设寄存器。

表 11-1 I<sup>2</sup>C 寄存器地址映像和复位值

寄存器简称	基址偏移量	复位值
I2C_CTRL1	0x00	0x0000
I2C_CTRL2	0x04	0x0000
I2C_OADDR1	0x08	0x0000
I2C_OADDR2	0x0C	0x0000
I2C_DT	0x10	0x0000
I2C_STS1	0x14	0x0000
I2C_STS2	0x18	0x0000
I2C_CLKCTRL	0x1C	0x0000
I2C_TMRISE	0x20	0x0002

#### 11.5.1 控制寄存器1(I2C\_CTRL1)

域	简称	复位值	类型	功能
位 15	RESET	0x0	rw	I <sup>2</sup> C 外设复位 (I <sup>2</sup> C peripheral reset) 0: 不复位; 1: 复位。 注: 该位可以用于 BUSYF 位为'1', 在总线上又没有检测到停止条件时。
位 14	保留	0x0	rstd	保持默认值。
位 13	SMBALERT	0x0	rw	SMBus 提醒引脚设置 (SMBus alert pin set) 软件可以使其置 1 或清为 0; 当 I2CEN=0 时, 由硬件清除。 0: 置高; 1: 置低。
位 12	PECTEN	0x0	rw	请求 PEC 传输使能 (Request PEC transmission enable) 软件可以使其置 1 或清为 0; 当传送 PECTEN 后, 开始或结束信号时, 由硬件清除。 0: 停止传输; 1: 启动传输。

				主机接收模式应答控制 (Master receiving mode acknowledge control) 0: ACKEN 位效果作用于当前传字节; 1: ACKEN 位效果作用于第二个传输字节。 该位只在主机接收两个字节模式下使用，目的是为了让主机及时的回 ACK。
位 11	MACKCTRL	0x0	rw	应答使能(Acknowledge enable) 软件可以使置 1 或清为 0; 0: 关闭, 不发送应答; 1: 开启。
位 10	ACKEN	0x0	rw	产生停止条件 (Generate stop condition) 软件可以使置 1 或清为 0; 或当检测到结束信号时，由硬件清除； 当检测到超时错误时，硬件将其置位。 0: 未产生; 1: 产生。 如果在从模式下当设置了此位，从机将释放 SCL 和 SDA 总线。
位 9	GENSTOP	0x0	rw	产生起始条件 (Generate start condition) 软件可以使置 1 或清为 0; 或当起始条件发出后，由硬件清除。 0: 未产生; 1: 产生。
位 8	GENSTART	0x0	rw	时钟延展模式 (Clock stretching mode) 0: 开启; 1: 关闭。
位 7	STRETCH	0x0	rw	广播地址使能 (General call address enable) 0: 开启; 1: 关闭。
位 6	GCAEN	0x0	rw	PEC 计算使能 (PEC calculation enable) 0: 关闭; 1: 开启。
位 5	PECEN	0x0	rw	SMBus ARP 协议使能 (SMBus address resolution protocol enable) 0: 关闭; 1: 开启。 SMBus 主机: 响应主机地址 0001000x; SMBus 设备: 响应设备默认地址 0001100x。
位 4	ARPEN	0x0	rw	SMBus 设备模式 (SMBus device mode) 0: SMBus 设备; 1: SMBus 主机。
位 3	SMBMODE	0x0	rw	I <sup>2</sup> C 外设模式 (I <sup>2</sup> C peripheral mode) 0: I <sup>2</sup> C 模式; 1: SMBus 模式。
位 2	保留	0x0	resd	硬件强制为 0。
位 1	PERMODE	0x0	rw	I <sup>2</sup> C 外设使能 (I <sup>2</sup> C peripheral enable) 0: 关闭; 1: 开启。
位 0	I2CEN	0x0	rw	在通讯结束后发生 I2CEN=0，所有的位被清除。 在主模式下，通讯结束之前，绝不能清除该位。

注意：当 GENSTART、GENSTOP 或 PECTEN 设置后，软件应该在相应位被硬件清零后写控制寄存器 1 (I2C\_CTRL1)，否则有可能产生第二次 GENSTART、GENSTOP 或 PECTEN 请求。

## 11.5.2 控制寄存器2(I2C\_CTRL2)

域	简称	复位值	类型	功能
位 15: 13	保留	0x0	resd	硬件强制为 0
位 12	DMAEND	0x0	rw	DMA 传输结束指示 (DMA transfer end indication) 0: 将要传输不是最后一笔数据; 1: 将要传输最后一笔数据。

位 11	DMAEN	0x0	rw	启动 DMA 传输 (DMA transfer enable) 0: 关闭; 1: 开启。
位 10	DATAIEN	0x0	rw	数据传输中断使能 (Data transmission interrupt enable) TDBE 或 RDBF 位置 1 时产生中断 0: 关闭; 1: 开启。
位 9	EVTIEN	0x0	rw	事件中断使能 (Event interrupt enable) 0: 关闭; 1: 开启。 在下列条件下, 将产生该中断: - STARTF = 1 (主模式) - ADDR7F = 1 (主/从模式) - ADDRHF = 1 (主模式) - STOPF = 1 (从模式) - TDC = 1, 但是没有 TDBE 或 RDBF 事件 - 如果 DATAIEN = 1, TDBE 事件为 1 - 如果 DATAIEN = 1, RDBF 事件为 1
位 8	ERRIEN	0x0	rw	错误中断使能 (Error interrupt enable) 0: 关闭; 1: 开启。 在下列条件下, 将产生该中断: - BUSERR = 1 - ARLOST = 1 - ACKFAIL = 1 - OVER = 1 - PECERR = 1 - TMOUT = 1 - ALERTF = 1
位 7: 0	CLKFREQ	0x00	rw	I <sup>2</sup> C 输入时钟频率 (I <sup>2</sup> C input clock frequency) 必须设置正确的输入时钟频率以产生正确的时序, 允许的范围在 2~120MHz 之间: 范围 2~120MHz。 2: 2MHz; 3: 3MHz; ..... 120: 120MHz。

### 11.5.3 自身地址寄存器1(I2C\_OADDR1)

域	简称	复位值	类型	功能
位 15	ADDR1MODE	0x0	rw	地址模式 (Address mode) 0: 7 位地址; 1: 10 位地址。
位 14: 10	保留	0x00	resd	保持默认值。
位 9: 0	ADDR1	0x000	rw	本机地址 1 (Own address) 当在 7 位地址模式下时 BIT0 以及 BIT[9: 8]不关心。

### 11.5.4 自身地址寄存器2(I2C\_OADDR2)

域	简称	复位值	类型	功能
位 15: 8	保留	0x00	resd	保持默认值。
位 7: 1	ADDR2	0x00	rw	本机地址 2 (Own address 2) 7 位地址。
位 0	ADDR2EN	0x0	rw	本机地址 2 使能 (Own address 2 enable) 0: 在 7 位地址模式下, 只有 OADDR1 被识别; 1: 在 7 位地址模式下, OADDR1 和 OADDR2 都被识别。

### 11.5.5 数据寄存器(I2C\_DT)

域	简称	复位值	类型	功能
位 15: 8	保留	0x00	resd	保持默认值。

位 7: 0	DT[7: 0]	0x00	rw	<p>用于存放接收到或待发送的数据 发送器模式：当写一个字节至 DT 寄存器时，自动启动数据传输。一旦传输开始(TDE=1)，如果能及时把下一个需传输的数据写入 DT 寄存器，I2C 模块将保持连续的数据流。 接收器模式：接收到的字节被拷贝到 DT 寄存器 (RDNE=1)。在接收到下一个字(RDNE=1)之前读出数据寄存器，即可实现连续的数据传送。 注：如果在处理 ACK 脉冲时发生 ARLOST 事件，接收到的字节不会被拷贝到数据寄存器里，因此不能读到它。</p>
--------	----------	------	----	--

## 11.5.6 状态寄存器1(I2C\_STS1)

域	简称	复位值	类型	功能
位 15	ALERTF	0x0	rw0c	<p>SMBus 提醒标志 (SMBus alert flag) 在 SMBus 主机模式下： 0: 未收到； 1: 收到。 SMBus 从机：指示设备默认地址接收 (0001100x) 0: 未收到； 1: 收到。 软件可以使其清为 0；当 I2CEN=0 时，由硬件清除。</p>
位 14	TMOUT	0x0	rw0c	<p>SMBus 超时标志 (SMBus timeout flag) 0: 无超时错误； 1: 超时。 软件可以使其清为 0；当 I2CEN=0 时，由硬件清除。 注：这个功能仅在 SMBUS 模式下有效</p>
位 13	保留	0x0	resd	保持默认值。
位 12	PECERR	0x0	rw0c	<p>PEC 接收错误标志 (PEC receive error flag) 0: 正确； 1: 错误。 软件可以使其清为 0。</p>
位 11	OUF	0x0	rw0c	<p>溢出标志 (Overload / underload flag) 当传输方向为发送数据时： 0: 正常； 1: 欠载。 当传输方向为接收数据时： 0: 正常； 1: 过载。 软件可以使其清为 0；当 I2CEN=0 时，由硬件清除。</p>
位 10	ACKFAIL	0x0	rw0c	<p>应答失败标志 (Acknowledge failure flag) 0: 正常； 1: 失败。 当没有返回应答时，硬件将置该位为'1' 软件可以使其清为 0；当 I2CEN=0 时，由硬件清除。</p>
位 9	ARLOST	0x0	rw0c	<p>仲裁丢失标志 (Arbitration lost flag) 0: 正常； 1: 仲裁丢失。 软件可以使其清为 0；当 I2CEN=0 时，由硬件清除。 在 ARLOST 事件之后，I<sup>2</sup>C 接口自动切换回从模式。</p>
位 8	BUSERR	0x0	rw0c	<p>总线错误标志 (Bus error flag) 0: 正常； 1: 错误。 当接口检测到错误的起始或停止条件，硬件将该位置'1'。 软件可以使其清为 0；当 I2CEN=0 时，由硬件清除</p>
位 7	TDBE	0x0	ro	<p>发送缓冲器空标志 (Transmit data buffer empty flag) 0: 数据正在从数据寄存器 (DT) 发送到移位寄存器，数据寄存器还装着数据； 1: 数据已经从数据寄存器 (DT) 发送到移位寄存器，数据寄存器空。</p>

				当 DT 为空的时候，该标志值起，向 DT 写数据时，此标志清除。 注：在写入第 1 个要发送的数据后，或设置了 TDC 时写入数据，都不能清除 TDRE 位，这是因为数据寄存器仍然为空。
位 6	RDBF	0x0	ro	接收数据缓冲器满标志 (Receive data buffer full flag) 0: 数据寄存器 (DT) 未接收到数据； 1: 数据寄存器 (DT) 接收到数据。 读取 DT 寄存器时，此标志清除。 在发生 ARLOST 事件时，RDBF 不被置位。
位 5	保留	0x0	resd	保持默认值。
位 4	STOPF	0x0	ro	停止条件产生完成标志 (Stop condition generation complete flag) 0: 未产生； 1: 已产生。 当 ACKEN=1，从设备在总线上检测到停止条件时，硬件将该位置'1' 先读取 STS1 寄存器，然后写 CTRL1 寄存器清除标志。
位 3	ADDRHF	0x0	ro	主机 9~8 位地址头匹配标志 (master 9~8 bit address header match flag) 0: 未匹配； 1: 已匹配。 在 10 位地址模式下，当主设备已经将第一个字节发送出去时，硬件将该位置'1' 软件读取 STS1 寄存器后，对 CTRL1 寄存器的写操作将清除该位，或当 PEN=0 时，硬件清除该位 注：收到一个 NACK 后，ADDR10F 位不被置位。
位 2	TDC	0x0	ro	数据传输完成标志 (Transmit data complete flag) 0: 未完成 (移位寄存器还有数据)； 1: 已完成 (移位寄存器空闲)。 读或写 DT 寄存器，或者收到开始或结束信号自动清除。 当 STRETCH=0 接收时收到一个新字节(包括 ACK 脉冲)且数据寄存器还未被读取(RDBF=1) 发送时，当一个新数据将被发送且数据寄存器还未被写入新的数据 (TDRE=1) 上述两种情况 TDC 位会置 1
位 1	ADDR7F	0x0	ro	0~7 位地址匹配标志 (0~7 bit address match flag) 0: 未产生； 1: 地址在主机模式下被发送或从机模式下接收到匹配地址。 在软件读取 STS1 寄存器后，对 STS2 寄存器的读操作将清除该位 注：在收到 NACK 后，ADDR7F 位不会被置位。
位 0	STARTF	0x0	ro	起始条件产生完成标志 (Start condition generation complete flag) 0: 未产生； 1: 已产生。 先读取 STS1 寄存器，然后写 DT 寄存器清除标志。

### 11.5.7 状态寄存器2(I2C\_STS2)

域	简称	复位值	类型	功能
位 15: 8	PECVAL	0x00	ro	PEC 值 (PEC value) 当 PECEN 重置时清零。
位 7	ADDR2F	0x0	ro	接收到地址 2 标志 (Received address 2 flag) 0: 接收到的地址与 OADDR1 内的内容相匹配； 1: 接收到的地址与 OADDR2 内的内容相匹配。 当收到 STOP/START 条件自动清除，或 I2CEN=0 时，硬件将该位清除

				SMBus 主机地址接收标志 (SMBus host address receiving flag) 0: 未接收; 1: 已接收。 当收到 STOP/START 条件自动清除, 或 I2CEN=0 时, 硬件将该位清除
位 6	HOSTADDRF	0x0	ro	SMBus 设备地址接收标志 (SMBus device address receiving flag) 0: 未接收; 1: 已接收。 当收到 STOP/START 条件自动清除, 或 I2CEN=0 时, 硬件将该位清除
位 5	DEVADDRF	0x0	ro	广播地址接收标志 (General call address reception flag) 0: 未接收; 1: 已接收。 当收到 STOP/START 条件自动清除, 或 I2CEN=0 时, 硬件将该位清除
位 4	GCADDRF	0x0	ro	保留默认值。
位 3	保留	0x0	resd	传输方向标志 (Transmission direction flag) 0: 接收数据; 1: 发送数据。 当收到 STOP 条件自动清除。
位 2	DIRF	0x0	ro	总线忙标志 (Bus busy flag transmission mode) 0: 空闲; 1: 忙。 当检测到 SDA/SCL 变低时置起, 检测到停止条件清零。
位 1	BUSYF	0x0	ro	传输模式 (Transmission mode) 0: 从机; 1: 主机。 当设置了 GENSTART 并发出 START 后, 该位置起, 当 检测到停止时, 该位清零。
位 0	TRMODE	0x0	ro	

### 11.5.8 时钟控制寄存器(I2C\_CLKCTRL)

域	简称	复位值	类型	功能
位 15	SPEEDMODE	0x0	rw	速度模式选择 (Speed mode selection) 0: 标准模式 (最快 100 kHz); 1: 快速模式 (最快 400 kHz)。 在快速模式下, 当 I <sup>2</sup> C 时钟为 10MHz 整数倍时, 可以产生准确的 400kHz 时钟
位 14	DUTYMODE	0x0	rw	快速模式占空比 (Fast mode duty cycle) 0: 高电平与低电平比值为 1: 2; 1: 低电平与高电平比值为 9: 16。
位 13: 12	保留	0x0	resd	保持默认值。
位 11: 0	SPEED	0x000	rw	I <sup>2</sup> C 总线速度配置 (I <sup>2</sup> C bus speed config) 在标准模式下: 高电平= SPEED × T <sub>I2C_CLK</sub> ; 低电平= SPEED × T <sub>I2C_CLK</sub> ; 在快速模式下: DUTYMODE = 0: 高电平= SPEED × T <sub>I2C_CLK</sub> × 1; 低电平= SPEED × T <sub>I2C_CLK</sub> × 2; DUTYMODE = 1: 高电平= SPEED × T <sub>I2C_CLK</sub> × 9; 低电平= SPEED × T <sub>I2C_CLK</sub> × 16。 标准模式下最小值为 4, 快速模式下最小值为 1。 只有在关闭 I <sup>2</sup> C 时(I2CEN=0)才能设置 CLKCTRL 寄存器;

注意: 只有当 I<sup>2</sup>C 被关闭时(I2CEN=0)才能设置时钟控制寄存器(I2C\_CLKCTRL)。

# 12 通用同步异步收发器 (USART)

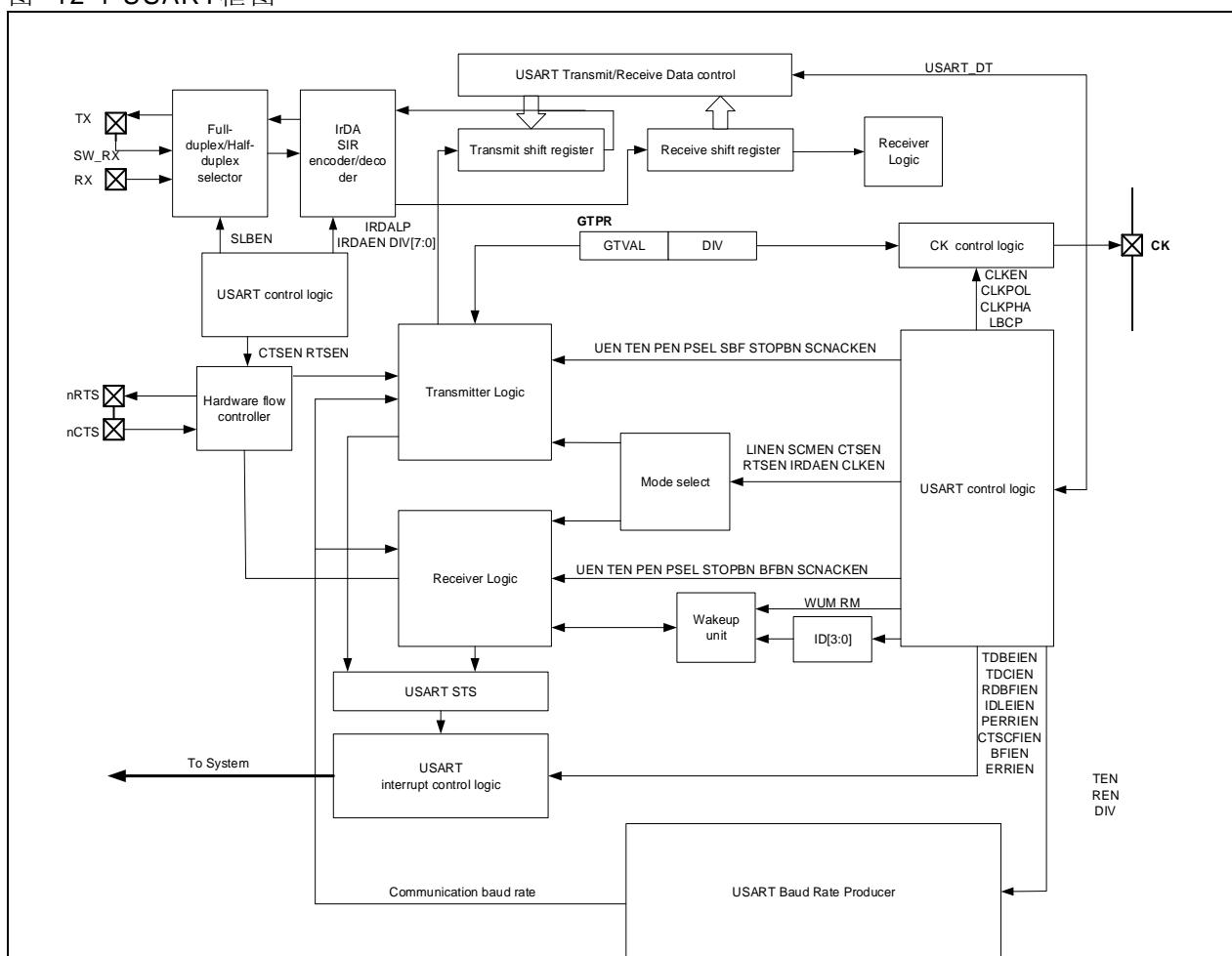
## 12.1 USART介绍

通用同步异步收发器 (USART) 是一个能通过多种不同的配置与使用不同的数据格式的外设进行通信的通用接口，同时支持异步全双工，异步半双工以及同步传输。USART 提供了可编程的波特率发生器，根据系统频率以及分频系数的不同，可产生高达 9.375MBits/s 的波特率，用户可以通过配置系统时钟以及分频系数以此产生所需要的特定通信频率。

USART 除了支持标准的 NRZ 异步以及同步收发通信协议外，还支持一些常用的其他类型的串行通信协议，如 LIN(局域互联网)，IrDA (红外数据组织) SIRENDEC 规范，ISO7816-3 标准的异步智能卡协议，以及 CTS/RTS (Clear To Send/Request To Send) 硬件流操作。

USART 还支持多处理器通信，以及可配置通过空闲帧或 ID 匹配唤醒的静默模式，以此搭建 USART 网络，并且同时支持使用 DMA 进行数据的收发，以此实现高速通信。

图 12-1 USART 框图



### USART 主要特性如下所列：

- 可编程配置的全双工或半双工通信
  - 全双工异步通信
  - 单线半双工通信
- 可编程配置的通信模式
  - NRZ标准格式 (Mark/Space)
  - LIN (局域互联网)：LIN主机有发送间隔帧的能力以及LIN从机有检测间隔帧的能力
  - IrDA SIR (串行红外)：在普通模式下持续时间为3/16位，在红外低功耗模式下位持续时间可调
  - ISO7816-3标准里定义的异步智能卡协议：智能卡模式支持0.5或1.5个停止位
  - RS-232 CTS/RTS (Clear To Send/Request To Send) 硬件流操作

- 通过静默模式实现多处理器通信(具有 ID 匹配和总线空闲两种可编程配置的唤醒方式)
- 同步模式
- 可编程配置的波特率发生器
  - 发送和接收共用的可编程波特率，最高达 9.375MBits/s
- 可编程配置的帧格式
  - 可编程的数据位位数（8位或9位）
  - 可编程的停止位位数-支持1或2个停止位
  - 可编程的校验控制：发送方具备发送校验位的能力，接收方具备对接收到的数据进行校验的能力
- 可编程配置的 DMA 多缓冲器通信
- 可编程配置的独立的发送器和接收器使能位
- 可编程配置的输出 CLK 的相位和极性以及频率
- 检测标志
  - 接收缓冲器满
  - 发送缓冲器空
  - 传输完成标志
- 四个错误检测标志
  - 溢出错误
  - 噪声错误
  - 帧错误
  - 校验错误
- 可编程配置的 10 个带标志的中断源
  - CTSF 改变
  - LIN 间隔帧检测
  - 发送数据寄存器空
  - 发送完成
  - 接收数据寄存器满
  - 检测到总线为空闲
  - 溢出错误
  - 帧错误
  - 噪声错误
  - 校验错误

## 12.2 全双工半双工选择器简述和配置流程

USART 全双工半双工选择器通过软件编程配置相应寄存器的方式，使得 USART 可以采用全双工或半双工的方式和外设进行数据交换。

USART 默认选择使用双线单向全双工时，TX 管脚用于数据输出，RX 管脚用于数据输入，USART 接收器和发送器相互独立，这使得 USART 可以同时进行数据发送和数据接收，以此实现全双工通信。

USART 在 HALFSEL 位置 1 时选择使用单线双向半双工的方式进行数据通信，在此条件下，LINEN 位，CLKEN 位，SCMEN 位以及 IRDAEN 位需置 0，此时在 USART 内部，RX 管脚无效，TX 管脚和 SW\_RX 管脚互连，对 USART 来说，TX 管脚用于数据输出，SW\_RX 用于数据输入，对外设来说，数据都从 TX 管脚映射的 IO 双向传输。

## 12.3 模式选择器简述和配置流程

### 12.3.1 模式选择器简述

USART 模式选择器通过软件编程配置相应寄存器的方式，使得 USART 可以根据软件的不同配置工作在不同的工作模式下，以此能与使用不同通信协议的外设之间实现数据交换。

USART 默认支持 NRZ 标准格式 (Mark/Space)，根据 USART 模式选择器配置的不同，USART 还可以支持 LIN (局域互联网)，IrDA SIR (串行红外)，ISO7816-3 标准里定义的异步智能卡协议，RS-232 CTS/RTS (Clear To Send/Request To Send) 硬件流操作以及静默模式和同步模式。

### 12.3.2 模式选择器配置方法

用户可以通过不同的配置以此选择不同的工作模式，配置方法分列如下所列，请将如下配置方法配合本章后述的接收器和发送器配置方法结合使用以完成 USART 初始化配置。

1. LIN模式：LINEN位置1，CLKEN位置0，STOPBN[1: 0]位置0，SCMEN位置0，SLHDEN位置0，IRDAEN位置0，DBN位置0。可以选择BFBN位置1或0来选择是11位还是10位间隔帧检测，可以使用SBF位置1发送13位低电平的LIN同步间隔帧。
2. 智能卡模式：SCMEN位置1，LINEN位置0，SLHDEN位置0，IRDAEN位置0，CLKEN位置1，DBN位置1，PEN位置1，STOPBN[1: 0]=11。可以选择配置CLKPOL位和CLKPHA位以及LBCP位以满足不同的时钟极性以及时钟相位和时钟脉冲个数，具体可见同步模式部分。可以通过配置SCGT[7: 0]位选择保护时间。可以通过配置SCNACKEN位选择是否在校验出错时发送NACK。
3. 红外模式：IRDAEN位置1，CLKEN位置0，STOPBN[1: 0]位置0，SCMEN位置0，SLHDEN位置0。可以选择IRDALP位置1以开启红外低功耗模式，并配合ISDIV[7: 0]配置想要产生的低功耗频率。
4. 硬件流控制模式：RTSEN位置1和CTSEN位置1可以分别开启RTS和CTS流控制。
5. 静默模式：RM位置1进入静默模式，根据WUM位置1和置0，可以分别通过ID匹配和空闲总线从静默模式中唤醒，其中ID号ID[3: 0]可编程配置，当选择ID匹配时，数据位的MSB为1表示当前数据是ID，4个LSB表示ID值。
6. 同步模式：CLKEN位置1开启同步模式并使能时钟管脚输出，通过配置CLKPOL位置1或0可以选择空闲状态下CK管脚上的电平为高或低，通过配置CLKPHA位置1或0可以选择在时钟的第二个或第一个边沿开始采样数据，通过配置LBCP位置1或0可以选择最后一位数据是否输出时钟，通过配置ISDIV[4: 0]可以选择想要输出的时钟频率。

## 12.4 USART帧格式简述和配置流程

USART一笔数据帧由起始位，数据位，停止位依次组成，最后一位数据位可以作为校验位。

USART一笔空闲帧的长度等于当前配置下数据帧的长度，但所有位都为1。

USART一笔间隔帧的长度等于当前配置下数据帧的长度加上停止位，停止位之前的所有位都等于0。

通过DBN位配置8位(DBN=0)或9位(DBN=1)数据位。

通过STOPBN位配置1位(STOPBN=00)，0.5位(STOPBN=01)，2位(STOPBN=10)，1.5位(STOPBN=11)停止位。

通过PEN位置“1”配置校验控制使能，通过PSEL位配置奇校验(PSEL=1)或偶校验(PSEL=0)，校验控制使能后数据位的MSB将由奇偶校验位替代，即有效数据位减少一位。

## 12.5 DMA传输简述和配置流程

USART可以使用DMA操作发送数据缓冲器和接收数据缓冲区以实现高速连续传输，USART的DMA传输需要配合DMA使用，下方会简述配置流程，但具体和DMA配置相关部分请参见DMA章节的描述。

### 12.5.1 DMA发送配置流程

1. 选择DMA传输通道：在DMA章节DMA通道映射表中选择用于当前所用USART的DMA通道。
2. 配置DMA传输目标地址：在DMA控制寄存器(TMRx\_DMACTRL)中DMA传输目的地址位写入当前所使用的USART的数据寄存器(USART\_DT)地址，DMA将会在接收到发送请求后将代发送的数据写入该地址。
3. 配置DMA传输源地址：在DMA控制寄存器(TMRx\_DMACTRL)中DMA传输源地址位写入代发送数据存放的地址，DMA将会在接收到发送请求后将该地址内的数据写入到目标地址中，即写入到当前所使用的USART的数据寄存器(USART\_DT)中。
4. 配置DMA传输字节个数：在DMA控制寄存器(TMRx\_DMACTRL)相关位置配置期望传输的字节个数
5. 配置DMA传输通道优先级：在DMA控制寄存器(TMRx\_DMACTRL)相关位置配置当前所使用通道的USART的DMA传输通道优先级。
6. 配置DMA中断产生时机：在DMA控制寄存器(TMRx\_DMACTRL)相关位置配置是在传输完成或传输完成一半时产生DMA中断。
7. 使能DMA传输通道：在DMA控制寄存器(TMRx\_DMACTRL)相关位置使能当前所选用的

DMA通道

### 12.5.2 DMA接收配置流程

1. 选择DMA传输通道：在DMA章节DMA通道映射表中选择用于当前所用USART的DMA通道。
2. 配置DMA传输目标地址：在DMA控制寄存器（TMRx\_DMACTRL）中DMA传输目的地址位写入期望存放接收数据的地址，DMA将会在接收到接收请求后，将当前所使用的USART的数据寄存器（USART\_DT）中的数据存放在目的地址中。
3. 配置DMA传输源地址：在DMA控制寄存器（TMRx\_DMACTRL）中DMA传输源地址位写入当前所使用的USART的数据寄存器（USART\_DT）的地址，DMA将会在接收到接收请求后将该地址内的数据写入到目标地址中，即写入到期望存放接收数据的地址。
4. 配置DMA传输字节个数：在DMA控制寄存器（TMRx\_DMACTRL）相关位置配置期望传输的字节个数
5. 配置DMA传输通道优先级：在DMA控制寄存器（TMRx\_DMACTRL）相关位置配置当前所使用通道的USART的DMA传输通道优先级。
6. 配置DMA中断产生时机：在DMA控制寄存器（TMRx\_DMACTRL）相关位置配置是在传输完成或传输完成一半时产生DMA中断。
7. 使能DMA传输通道：在DMA控制寄存器（TMRx\_DMACTRL）相关位置使能当前所选用的DMA通道

## 12.6 波特率发生器简述及配置流程

### 12.6.1 波特率发生器简述

USART 波特率发生器通过使用内部计数器，以 PCLK 为基准，DIV(波特比率寄存器(USART\_BAUDR) [15: 0]) 即为该计数器的溢出值，该计数器计满一次代表一位数据，所以每位数据位宽为 DIV 个 PCLK 周期。

由于 USART 的接收器和发送器共用同一个波特率发生器，并且接收器将每位数据拆分为 16 份等长的部分以此来实现过采样，所以数据位宽不得小于 16 个 PCLK 周期，即 DIV 中的值必须大于 16。

### 12.6.2 波特率发生器配置方法

用户可通过配置不同的系统时钟以及在波特比率寄存器（USART\_BAUDR）中写入不同的值以此产生特定的波特率，具体的运算关系见如下公式

$$\text{TX/RX 波特率} = \frac{f_{CK}}{\text{DIV}}$$

这里的 $f_{CK}$ 是指 USART 的系统时钟（即对应的 PCLK1/PCLK2）

注：1. 波特比率寄存器（USART\_BAUDR）中的值需要在 UEN 之前写入，且 UEN=1 时，不可更改这些位。

2. 关闭 USART 接收器或发送器会使内部计数器复位，波特率发生中断。

## 12.7 发送器简述和配置流程

### 12.7.1 发送器简述

USART 发送器具有独立的使能位 TEN，发送器与接收器共用同一个波特率且该波特率可编程配置，USART 具有一个发送数据缓冲器（TDR）和一个发送移位寄存器，当发送数据缓冲器（TDR）为空时，TDBE 置起，如果设置了 TDBEIEN 将会产生中断。

软件写入的值会先存储在发送数据缓冲器（TDR）中，当发送移位寄存器为空时，USART 会将发送数据缓冲器中的值移到发送移位寄存器，USART 发送器将以 LSB 的方式将发送移位寄存器中的数据从 TX 脚输出，具体的输出格式取决于软件配置的帧格式。

如若选择了同步传输或者配置了时钟输出，USART 发送器将时钟脉冲从 CK 脚输出，如若选择了硬件流控制，USART 发送器将控制信号将从 CTS 管脚输入。

注意：1. 在数据传输期间不能复位 TEN 位，否则将破坏 TX 脚上的数据。

2. TEN 位被激活后，USART 将自动发送一个空闲帧。

## 12.7.2 发送器配置流程

1. USART使能: UEN位置1。
2. 全双工半双工配置: 具体参见全双工半双工选择器配置部分。
3. 模式配置: 具体参见模式选择器配置部分。
4. 帧格式配置: 具体参见帧格式配置部分。
5. 中断配置: 具体参见中断发生器配置部分。
6. DMA发送配置: 如果选择使用DMA发送, DMATEN位(控制寄存器3 (USART\_CTRL3) [7])置1, 并按照DMA传输中的描述配置DMA寄存器。
7. 波特率配置: 具体参见波特率发生器配置部分。
8. 发送器使能: TEN位置1, 置1后USART发送器会自动发送一个空闲帧。
9. 数据写入: 等待TDBE位置起后, 将要发送的数据写入数据寄存器 (USART\_DT) (此操作会清除TDBE位), 在非DMA模式下, 重复此操作。
10. 在写入最后一个期望传输的数据后, 等待TDC位置起, 这表示最后一个数据帧的传输结束, 在该标志置起前, 禁止关闭USART, 否则传输可能出错。
11. 在TDC=1后, 可以采用先读一次状态寄存器 (USART\_STS), 再写一次数据寄存器 (USART\_DT) 的方式来清除TDC; 也可以采用软件对它写'0'来清除, 但此方法只推荐在DMA模式下使用。

## 12.8 接收器简述和配置流程

### 12.8.1 接收器简述

USART接收器具有独立的接收器使能位 REN(控制寄存器 1 (USART\_CTRL1) [2]), 接收器和发送器共用同一个波特率且该波特率可编程配置, USART 具有一个接收数据缓冲器 (RDR) 和一个接收移位寄存器。

数据从 USART 的 RX 脚输入, 当接收器判断到一个有效的起始位后, 接收器会以 LSB 的方式将接收到的数据依次移入接收移位寄存器, 并根据软件配置的帧格式, 在接收到一个完整的数据帧后将接收移位寄存器中的值移入接收数据缓冲器并置起 RDBF, 如果设置了 RDBFIEN 将会产生中断。

如若选择了硬件流控制, USART 接收器将控制信号将从 RTS 管脚输出。

在数据接收过程中, USART 接收器会根据软件的配置检测帧错误, 溢出错误, 奇偶校验错误以及噪声错误, 并根据相应的中断使能位是否置位来判断是否产生相应的中断。

### 12.8.2 接收器配置流程

配置步骤:

1. USART使能: UEN位置1。
2. 全双工半双工配置: 具体参见全双工半双工选择器配置部分。
3. 模式配置: 具体参见模式选择器配置部分。
4. 帧格式配置: 具体参见帧格式配置部分。
5. 中断配置: 具体参见中断发生器配置部分。
6. DMA接收配置: 如果选择使用DMA接收, DMAREN位置1, 并按照DMA传输中的描述配置DMA寄存器。
7. 波特率配置: 具体参见波特率发生器配置部分。
8. 接收器使能: REN位置1。

当一字符被接收到时:

- RDBF 位被置位。它表明移位寄存器的内容被转移到 RDR (Receiver Data Register)。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)。
- 如果 RDBFIEN 位被设置, 则产生中断。
- 在接收期间如果检测到帧错误, 噪声或溢出错误, 错误标志将被置起。
- 在 DMA 传输时, RDNE 在每个字节接收后被置起, 并由 DMA 对数据寄存器的读操作而清零。
- 在非 DMA 传输时, 由软件读数据寄存器 (USART\_DT) 完成对 RDBF 位清除。RDBF 标志也可以通过对它写 0 来清除。RDBF 位必须在下一帧数据接收结束前被清零, 以避免溢出错误。

当一个间隔帧被接收到时：

- 非 LIN 模式：USART 接收器按照帧错误处理，并置起 FERR 位，若相应中断使能，中断产生，具体可见下方错误帧的描述。
- LIN 模式：USART 接收器按间隔帧处理，并置起 BFF 位，若 BFIEN 置位，则中断产生。

当一个空闲帧被接收到时：

- USART 接收器按数据帧处理，并置起 IDLEF 位，若 IDLEIEN 置位，则中断产生。

当一个帧错误产生时：

- FERR 位置位。
- USART 接收器将错误的数据从接收移位寄存器转移到接收数据缓冲器。
- 在非 DMA 传输时，这个位和 RDBF 位同时置起，后者将产生中断。在 DMA 传输时，如果 ERRIEN 置位的话，将产生中断。

当一个溢出错误产生时：

- ROERR 位被置位。
- 接收数据缓冲器中的数据不会被覆盖，读数据寄存器（USART\_DT）仍能得到先前的数据。
- 接收移位寄存器中的内容会被覆盖，随后接收到的数据都将丢失。
- 如果 RDBFIEN 位置位或 ERRIEN 和 DMAREN 位都被置位，中断产生。
- 先读状态寄存器（USART\_STS），再读数据寄存器（USART\_DT），可清除 ROERR。

注意：当 ROERR 置位时，表明至少有 1 个数据已经丢失。有两种可能性：

如果 RDBF=1，上一个有效数据还存储在接收数据缓冲器中，可以被读出。如果 RDBF=0，这意味着上一个有效数据已经从接收数据缓冲器中读走。

注意：在接收数据时，REN 位不应该被复位。如果 REN 位在接收时被清零，当前字节的接收被丢失。

### 12.8.3 起始侦测和噪声检测

USART 接收器在 REN 位置位后便开始侦测起始位，USART 接收器通过过采样技术，在第 3、5、7、8、9、10 位共 6 个点进行数据采样，以此侦测有效起始位以及识别噪声，具体的噪声和有效起始位的判别方式可以参见下方表检测起始位和噪声的数据采样。

表 12-1 检测起始位和噪声的数据采样

采样值 (3 · 5 · 7)	采样值 (8 · 9 · 10)	NERR 位	起始位有效性
000	000	0	有效
001/010/100	001/010/100	1	有效
001/010/100	000	1	有效
000	001/010/100	1	有效
111/110/101/011	任意值	1	无效
任意值	111/110/101/011	1	无效

注意：如果在第 3、5、7、8、9、10 位的采样值满足不了上表任意一种组合，则 USART 接收器认为没有接受到正确的起始位，将退出起始位侦测并回到空闲状态等待下降沿。

USART 接收器具备噪声检测功能，在非同步模式时，使用过采样技术，在第 7、8、9 采样点，根据不同的采样值，区别有效输入数据和噪声，并恢复数据和置起噪声错误标志位 NERR。具体的采样方法以及噪声和有效数据的判别方式可以参见下方表检测有效数据和噪声的数据采样。

表 12-2 检测有效数据和噪声的数据采样

采样值	NERR 位	接收的位	数据有效性
000	0	0	有效
001	1	0	无效
010	1	0	无效
011	1	1	无效
100	1	0	无效
101	1	1	无效
110	1	1	无效
111	0	1	有效

当 USART 接收器在数据帧中检测到噪声时：

- 在 RDBF 位置起的同时置起 NERR 位。
  - USART 接收器将错误数据从接收移位寄存器转移到接收数据缓冲器。
  - 在非 DMA 传输时，没有噪声中断产生。然而，因为 NERR 位和 RDBF 位是同时置位，RDBF 将产生中断。在 DMA 传输时，如果 ERRIEN 位置位，中断产生。
- 先读状态寄存器 (USART\_STS)，再读数据寄存器 (USART\_DT)，将清除 NERR 位。

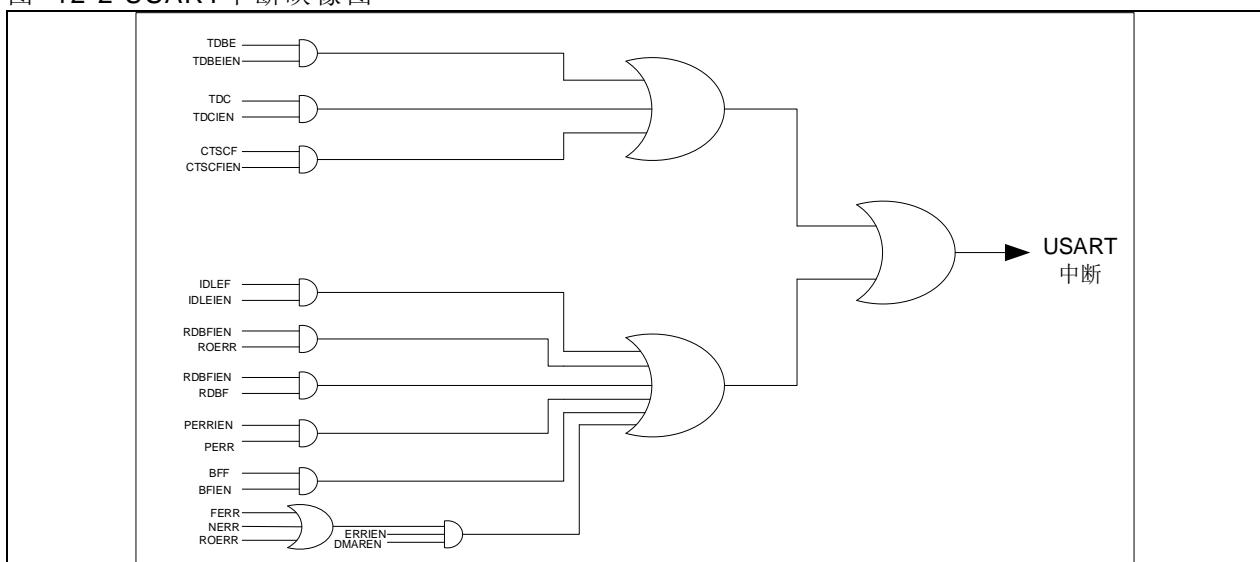
## 12.9 中断

USART 中断发生器是 USART 中断的控制中枢，USART 中断产生器会实时监测 USART 内部的中断源，并根据软件配置的相应中断源的中断使能位，以此决定是否产生中断，下表所示为 USART 的中断源以及相应的中断使能位，对相应的中断使能位置 1 时，即可在相应事件出现后产生中断。

表 12-3 USART 中断请求

中断事件	事件标志	使能位
发送数据寄存器空	TDBE	TDBEIEN
CTS 标志	CTSCF	CTSCFIEN
发送完成	TDC	TDCIEN
接收数据就绪可读	RDBF	RDBFIEN
检测到数据溢出	ROERR	
检测到空闲线路	IDLEF	IDLEIEN
奇偶检验错	PERR	PERRIEN
断开标志	BFF	BFIEN
噪声标志，多缓冲通信中的溢出错误和帧错误	NERR 或 ROERR 或 FERR	ERRIEN <sup>(1)</sup>

图 12-2 USART 中断映像图



## 12.10 I/O 管脚控制

USART 通过五个接口外部设备进行通信，管脚定义如下：

**RX:** 串行数据输入端。

**TX:** 串行数据输出端。在单线半双工模式和智能卡模式里，TX 脚作为 I/O 使用，即用于发送数据也用于接收数据。

**CK:** 发送器时钟输出。输出的 CLK 相位和极性以及频率均可编程配置。

**CTS:** 发送器输入端，硬件流控制模式发送使能信号。

**RTS:** 接收器输出端，硬件流控制模式发送请求信号。

## 12.11 USART 寄存器描述

必须以字（32 位）的方式操作这些外设寄存器。

表 12-4 USART寄存器映像和复位值

寄存器简称	基址偏移量	复位值
USART_STS	0x00	0x0000 00C0
USART_DT	0x04	0x0000 0000
USART_BAUDR	0x08	0x0000 0000
USART_CTRL1	0x0C	0x0000 0000
USART_CTRL2	0x10	0x0000 0000
USART_CTRL3	0x14	0x0000 0000
USART_GDIV	0x18	0x0000 0000

### 12.11.1 状态寄存器 (USART\_STS)

域	简称	复位值	类型	功能
位 31: 10	保留	0x000000	resd	硬件强制为 0。
位 9	CTSCF	0x0	rw0c	CTS 变化标志 (CTS change flag) 当 CTS 线发送变化时，该位被硬件置起，由软件将其清零。 0: 无； 1: 有。
位 8	BFF	0x0	rw0c	间隔帧标志 (break frame flag) 当检测到间隔帧时，该位被硬件置起，由软件将其清零。 0: 无； 1: 有。
位 7	TDBE	0x1	ro	发送缓冲器空 (Transmit data buffer empty) 当发送缓冲器为空，可以再次写入数据时，该位被硬件置起。对数据寄存器 (USART_DT) 的写操作，将清零该位。 0: 非空； 1: 空。
位 6	TDC	0x1	rw0c	发送数据完成 (Transmit data complete) 当发送数据完成，该位被硬件置起，由软件将其清零（方式 1：先读状态寄存器 (USART_STS)，再写数据寄存器 (USART_DT)；方式 2：操作该位写'0'）。 0: 未完成； 1: 完成。
位 5	RDBF	0x0	rw0c	接收数据缓冲器满 (Receive data buffer full) 当接收到数据时，该位被硬件置起，由软件将其清零（方式 1：读数据寄存器 (USART_DT)；方式 2：操作该位写'0'）。 0: 未收到； 1: 收到。
位 4	IDLEF	0x0	ro	总线空闲 (Idle flag) 当检测到总线空闲时，该位被硬件置起，由软件将其清零（先读状态寄存器 (USART_STS)，再读数据寄存器 (USART_DT)）。 0: 无； 1: 有。
位 3	ROERR	0x0	ro	接收器溢出错误 (Receiver overflow error) 当 RDNE 仍然置起没有清除的时候，如果此时又收到数据，该位被硬件置起，由软件将其清零（先读状态寄存器 (USART_STS)，再读 USART_DT）。 0: 无； 1: 有。 注意：该位被置位时，DT 寄存器中的数据不会丢失，但是后续的数据会被覆盖。
位 2	NERR	0x0	ro	噪声错误 (Noise error)

位 1	FERR	0x0	ro	接收到的数据有杂讯时，该位被硬件置起，由软件将其清零（先读状态寄存器（USART_STS），再读数据寄存器（USART_DT））。 0: 无； 1: 有。
位 0	PERR	0x0	ro	帧错误（Framing error） 当检测到停止位异常（检测到低电平）、过多的杂讯噪声或者检测到间隔帧，该位被硬件置起，由软件将其清零（先读状态寄存器（USART_STS），再读数据寄存器（USART_DT））。 0: 无； 1: 有。
				校验错误（Parity error） 接收如果出现奇偶校验错误，该位被硬件置起，由软件将其清零（先读状态寄存器（USART_STS），再读数据寄存器（USART_DT））。 0: 无； 1: 有。

## 12.11.2 数据寄存器（USART\_DT）

域	简称	复位值	类型	功能
位 31: 9	保留位	0x000000	resd	硬件强制为 0。
位 8: 0	DT	0x00	rw	数据值（Data value） 该寄存器包含读和写的功能。当奇偶校验位使能，发送操作时，写到 MSB 的值会被校验位取代。接收操作时，读到的 MSB 位是接收到的校验位。

## 12.11.3 波特比率寄存器（USART\_BAUDR）

注意：如果 TE 或 RE 被分别禁止，波特计数器停止计数。

域	简称	复位值	类型	功能
位 31: 16	保留位	0x0000	resd	硬件强制为 0。
位 15: 0	DIV	0x0000	rw	分频系数（Division） 这 16 位定义了 USART 分频系数。

## 12.11.4 控制寄存器1（USART\_CTRL1）

域	简称	复位值	类型	功能
位 31: 14	保留位	0x000000	resd	硬件强制为 0。
位 13	UEN	0x0	rw	USART 使能（USART enable） 0: 关闭； 1: 启开。
位 12	DBN	0x0	rw	数据位个数（Data bit num） 该位定义了数据位的个数。 0: 8 位； 1: 9 位。
位 11	WUM	0x0	rw	唤醒方式（Wake up mode） 该位定义静默状态下被唤醒的方式。 0: 空闲帧唤醒； 1: ID 匹配唤醒。
位 10	PEN	0x0	rw	奇偶校验使能（Parity enable） 该位定义使能硬件奇偶校验（对于发送来说就是校验位的产生；对于接收来说就是校验位的检测）。当使能了该位，硬件将发送数据的最高位替换成校验位；对接收到的数据检查其校验位是否正确。 0: 关闭； 1: 启开。
位 9	PSEL	0x0	rw	奇偶校验选择（Parity selection） 该位定义是采用奇校验还是偶校验。 0: 偶校验； 1: 奇校验。

位 8	PERRIEN	0x0	rw	PERR 中断使能 (PERR interrupt enable) 0: 关闭; 1: 开启。
位 7	TDBEIEN	0x0	rw	发送数据缓冲器空中断使能 (TDBE interrupt enable) 0: 关闭; 1: 开启。
位 6	TDCIEN	0x0	rw	发送数据完成中断使能 (TDC interrupt enable) 0: 关闭; 1: 开启。
位 5	RDBFIEN	0x0	rw	接收数据缓冲器满中断使能 (RDNE interrupt enable) 0: 关闭; 1: 开启。
位 4	IDLEIEN	0x0	rw	总线空闲中断使能 (IDLE interrupt enable) 0: 关闭; 1: 开启。
位 3	TEN	0x0	rw	发送使能 (Transmitter enable) 该位定义发送端的使能。 0: 关闭; 1: 开启。
位 2	REN	0x0	rw	接收使能 (Receiver enable) 该位定义接收端的使能。 0: 关闭; 1: 开启。
位 1	RM	0x0	rw	接收静默 (Receiver mute) 该位定义接收端静默的开启，可由软件置起或清零。当配置为空闲帧唤醒时，唤醒后硬件也会将其清零，当配置为匹配地址唤醒时，收到匹配地址唤醒后硬件会将其清零，收到不匹配地址后硬件会再次将其置起进入静默状态。 0: 普通; 1: 静默。
位 0	SBF	0x0	rw	发送间隔帧 (Send break frame) 使用该位来发送间隔帧。该位可以由软件置起或清零。常规用法是软件置起该位，间隔帧发送完成后，由硬件将该位清零。 0: 无; 1: 发送。

### 12.11.5 控制寄存器2 (USART\_CTRL2)

域	简称	复位值	类型	功能
位 31: 15	保留位	0x00000	resd	硬件强制为 0。
位 14	LINEN	0x0	rw	LIN 模式使能 (LIN mode enable) 0: 关闭; 1: 开启。
位 13: 12	STOPBN	0x0	rw	停止位个数 (STOP bit num) 这 2 位用来设置停止位的个数 00: 1 位; 01: 0.5 位; 10: 2 位; 11: 1.5 位;
位 11	CLKEN	0x0	rw	时钟使能 (Clock enable) 该位用来使能同步模式或智能卡模式的时钟管脚。 0: 关闭; 1: 开启。
位 10	CLKPOL	0x0	rw	时钟极性 (Clock polarity) 在同步模式或智能卡模式下，可以用该位选择时钟管脚上总线空闲时时钟输出的极性。 0: 低电平; 1: 高电平。
位 9	CLKPHA	0x0	rw	时钟相位 (Clock phase)

				在同步模式或智能卡模式下，可以用该位选择时钟管脚上时钟输出的相位。 0: 第一个边沿进行数据捕获； 1: 第二个边沿进行数据捕获。
位 8	LBCP	0x0	rw	最后一位时钟脉冲 (Last bit clock pulse) 在同步模式下，使用该位来控制是否在时钟管脚上输出数据的最后一位对应的时钟脉冲 0: 不输出； 1: 输出。
位 7	保留位	0x0	resd	保持默认值。
位 6	BFIEN	0x0	rw	间隔帧中断使能 (break frame interrupt enable) 0: 关闭； 1: 开启。
位 5	BFBN	0x0	rw	间隔帧位数 (break frame bit num) 该位用来选择是 11 位还是 10 位的间隔帧。 0: 10 位； 1: 11 位。
位 4	保留位	0x0	resd	保持默认值。
位 3: 0	ID	0x0	rw	USART 的 ID 号 (USART identification) 可配置的 USART 的 ID 号。

注意： 在使能发送后不能改写这三个位 (CLKPOL、CLKPHA、LBCP)。

### 12.11.6 控制寄存器3 (USART\_CTRL3)

域	简称	复位值	类型	功能
位 31: 11	保留位	0x000000	resd	硬件强制为 0。
位 10	CTSCFIEN	0x0	rw	CTSCF 中断使能 (CTSCF interrupt enable) 0: 关闭； 1: 开启。
位 9	CTSEN	0x0	rw	CTS 使能 (CTS enable) 0: 关闭； 1: 开启。
位 8	RTSEN	0x0	rw	RTS 使能 (RTS enable) 0: 关闭； 1: 开启。
位 7	DMATEN	0x0	rw	DMA 发送使能 (DMA transmit enable) 0: 关闭； 1: 开启。
位 6	DMAREN	0x0	rw	DMA 接收使能 (DMA receiver enable) 0: 关闭； 1: 开启。
位 5	SCMEN	0x0	rw	智能卡模式使能 (Smart card mode enable) 0: 关闭； 1: 开启。
位 4	SCNACKEN	0x0	rw	智能卡 NACK 使能 (Smart card NACK enable) 该位用于配置校验错误出现时，发送 NACK。 0: 不发送； 1: 发送。
位 3	SLBEN	0x0	rw	单线双向半双工模式使能 (Single line bidirectional half-duplex enable) 0: 关闭； 1: 开启。
位 2	IRDALP	0x0	rw	红外低功耗模式配置 (IrDA low-power mode) 该位用来配置红外低功耗模式。 0: 关闭； 1: 开启。
位 1	IRDAEN	0x0	rw	红外功能使能 (IrDA enable) 0: 关闭； 1: 开启。
位 0	ERRIEN	0x0	rw	错误中断使能 (Error interrupt enable) 当有帧错误、接收溢出错误或者杂讯错误时产生中断。

0: 关闭;  
1: 开启。

### 12.11.7 保护时间和预分频寄存器 (USART\_GDIV)

域	简称	复位值	类型	功能
位 31: 16	保留位	0x0000	resd	硬件强制为 0。
位 15: 8	SCGT	0x00	rw	智能卡保护时间值 (Smart card guard time) 在智能卡模式下, 当保护时间过去后, 才会设置发送完成标志, 这几位配置保护时间值。
位 7: 0	ISDIV	0x00	rw	红外或者智能卡分频系数 (IrDA/smartcard division) 红外 (IrDA) 模式: 8 位[7: 0]有效, 普通模式无效且只能设置为 00000001, 低功耗模式分频系数对外设时钟进行分频, 作为脉冲宽度的基数周期; 00000000: 保留 - 不要写入该值; 00000001: 1 分频; 00000010: 2 分频; ..... 智能卡模式: 低 5 位[4: 0]有效, 分频系数对外设时钟进行分频, 给智能卡提供时钟。可以设置为如下值: 00000: 保留 - 不要写入该值; 00001: 2 分频; 00010: 4 分频; 00011: 6 分频; .....

# 13 串行外设接口 (SPI)

## 13.1 串行外设接口 (SPI) 简介

SPI 接口提供软件编程配置选项，本章将介绍 SPI 的功能特性以及配置流程。

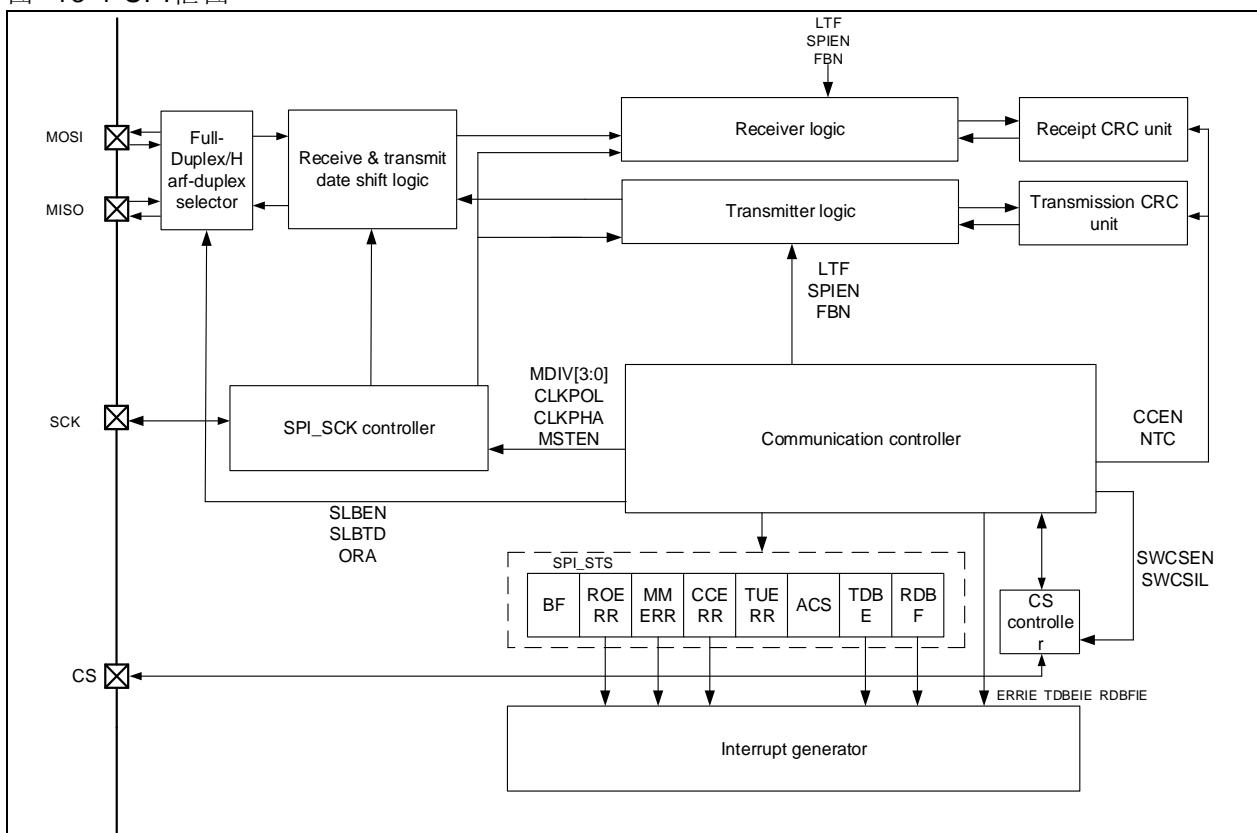
## 13.2 SPI 功能描述

### 13.2.1 SPI 简述

串行外设接口 (SPI) 根据软件编程配置的方式不同，可以分别作为主机和从机使用，又可以分别工作在全双工，全双工只收，半双工只发/只收四种不同的模式下，并且还提供 DMA 传输，SPI 内部硬件自动 CRC 计算和校验等功能。

**SPI 的架构框图见下图：**

图 13-1 SPI 框图



**SPI 接口作为 SPI 使用时主要特征如下：**

- 可编程配置的全双工或半双工通信
  - 全双工同步通信（可以选择全双工只收以此释放用于发送的 IO）
  - 半双工同步通信（可以根据软件编程配置选择传输方向：发送或接收）
- 可编程配置主/从模式
- 可编程配置的 CS 信号处理方式
  - 硬件处理 CS
  - 软件处理 CS
- 可编程配置的 8 位或 16 位帧位数
- 可编程配置的通信频率以及分频系数（最大分频系数为  $f_{PCLK}/2$ ）
- 可编程配置的时钟极性和相位
- 可编程配置的数据传输顺序(先发 MSB/LSB)
- 可编程配置的错误中断标志（接收器溢出错误，主模式错误，CRC 校验错误）
- 可编程配置的发送数据缓冲器空中断以及接收数据缓冲器满中断
- 支持 DMA 发送和接收

- 支持硬件 CRC 发送和校验
- 具备通信忙标志

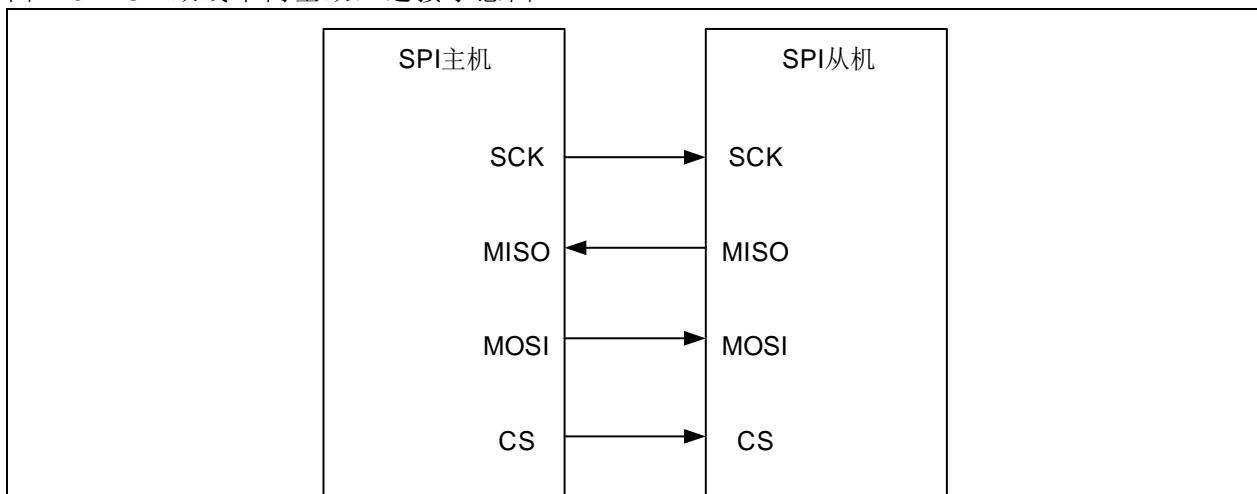
### 13.2.2 全双工半双工选择器简述和配置流程

SPI 全双工半双工选择器通过软件编程配置的方式，可以使 SPI 接口作为 SPI 使用时，可以工作在双线单向全双工，单线单向只收，单线双向半双工发送和单线双向半双工接收四种同步模式。

**双线单向全双工模式配置方式以及 SPI IO 连接方式如下：**

SLBEN 位置 0, ORA 置 0 时，SPI 工作在双线单向全双工，此时 SPI 可以同时进行数据的收发，IO 连接方式如下图。

图 13-2 SPI 双线单向全双工连接示意图



SPI 作主机或从机在此模式下，关闭 SPI 或进入省电模式（或关闭 SPI 系统时钟）之前需要等待 RDBF 置位，TDBE 置位，并等待 BF=0。

**单线单向只收模式配置方式以及 SPI IO 连接方式如下：**

SLBEN 位置 0, ORA 置 1 时，SPI 工作在单线单向只收模式，此时 SPI 只能作为数据接收方，无法发送数据。作为主机时使用 MISO 接收数据，MOSI 管脚所映射的 IO 释放。作为从机时使用 MOSI 接收数据，MISO 管脚所映射的 IO 释放。

图 13-3 SPI 作主机单线单向只收连接示意图

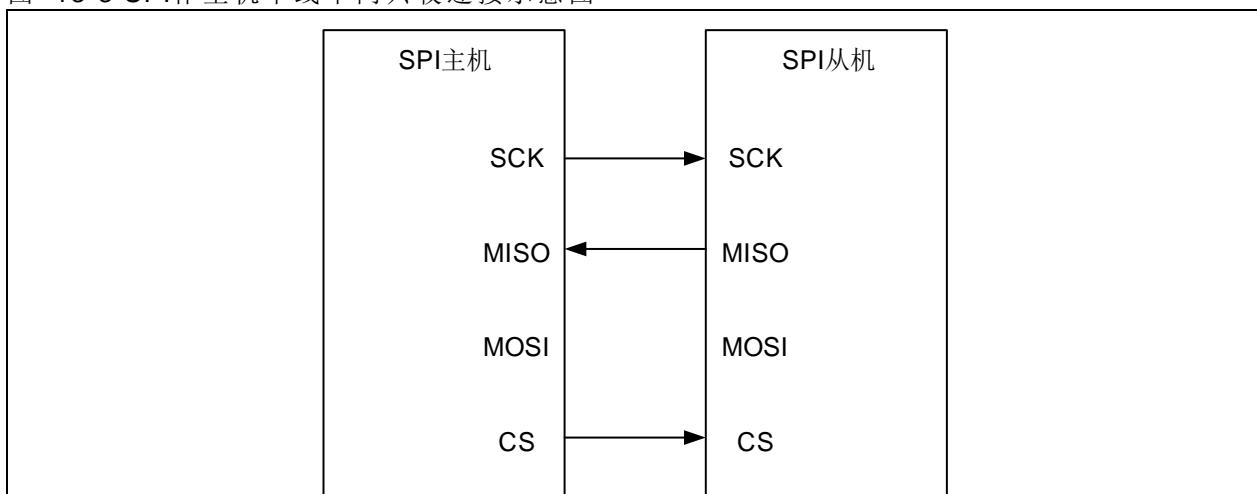
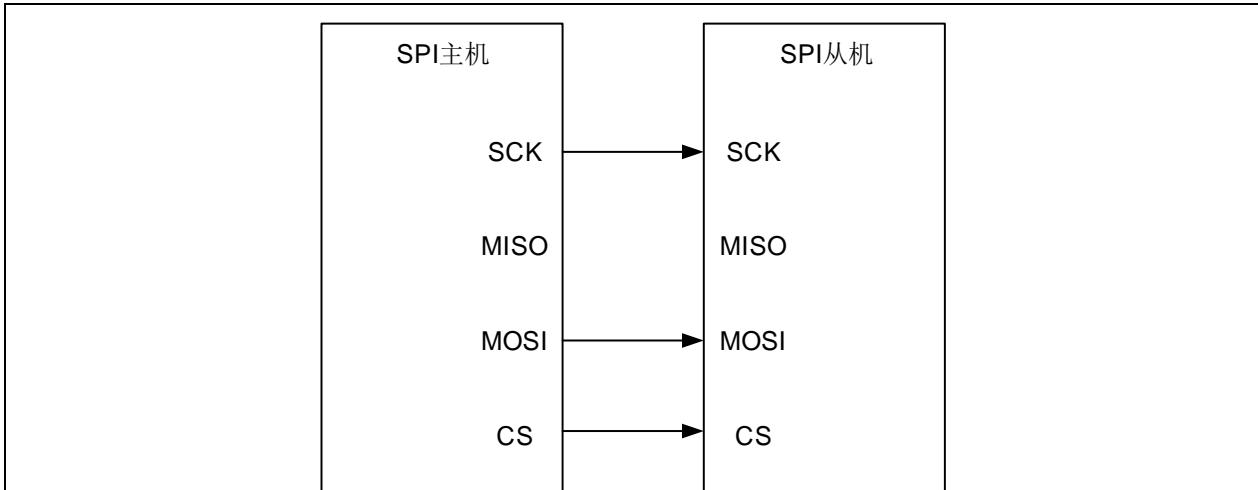


图 13-4 SPI作从机单线单向只收连接示意图



SPI 作主机时，在此模式下，需要等待倒数第二个 RDBF 置起，关闭 SPI 之前等待一个 SPI\_SCK 周期，在进入省电模式(或关闭 SPI 系统时钟)之前等待最后一个 RDBF=1。

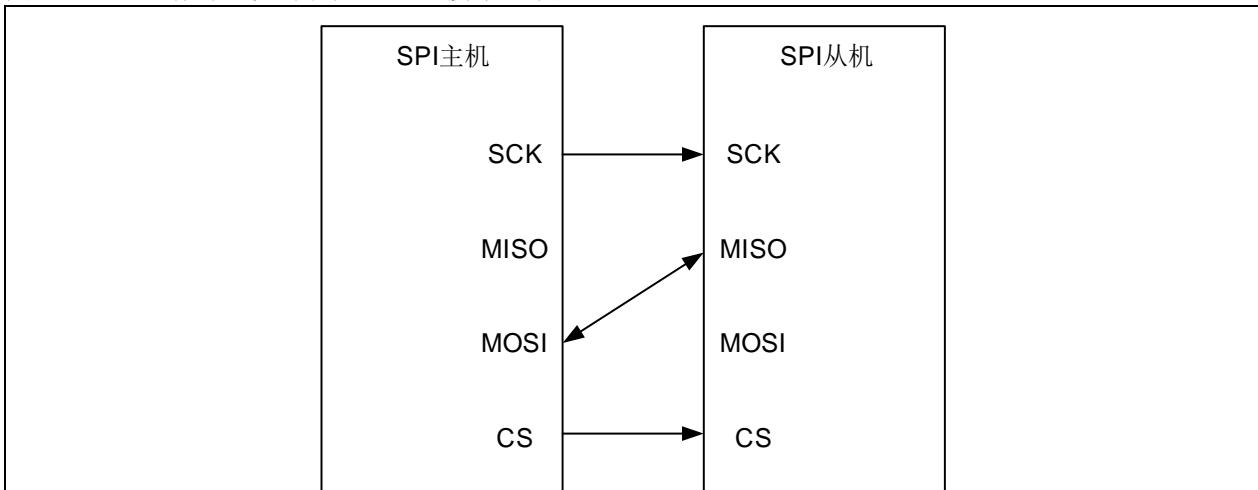
SPI 作从机时，在此模式下，关闭 SPI 无需判断任何标志，但是在进入省电模式(或关闭 SPI 系统时钟)之前需要等待 BF=0。

#### 单线双向半双工模式配置方式以及 SPI IO 连接方式如下：

SLBEN 位置 1 时，SPI 工作在单线双向半双工模式，此时 SPI 可以分时进行数据收发。作为主机时使用 MOSI 收发数据，MISO 管脚所映射的 IO 释放。作为从机时使用 MISO 收发数据，MOSI 管脚所映射的 IO 释放。

软件通过编程控制 SLBTD 位控制传输方向，SLBTD 位置 1 时，SPI 只能发送数据，SLBTD 位置 0 时，SPI 只能接收数据。

图 13-5 SPI作单线双向半双工连接示意图



SPI 作主机或从机时，在单线双向半双工，传输方向选择为发送时，需要等待 TDBE 置位，BF=0 后才能关闭 SPI，在关闭 SPI 后才可以进入省电模式(或关闭 SPI 系统时钟)。

SPI 作主机时，在单线双向半双工，传输方向选择为接收时，需要等待倒数第二个 RDBF 置起，关闭 SPI 之前等待一个 SPI\_SCK 周期，在进入省电模式(或关闭 SPI 系统时钟)之前等待最后一个 RDBF=1。

SPI 作从机时，在单线双向半双工，传输方向选择为接收时，关闭 SPI 无需判断任何标志，但是在进入省电模式(或关闭 SPI 系统时钟)之前需要等待 BF=0。

### 13.2.3 CS控制器简述和配置流程

SPI 的 CS 控制器提供通过软件可编程配置的方式选择硬件控制 CS 信号或软件控制 CS 信号，以此实现 CS 信号的控制，用于多处理器模式下主机从机选择，以及通过 CS 信号后于 SCK 信号使能，有效地屏蔽总线上的干扰，下面将分软件 CS 以及硬件 CS 来介绍 CS 控制器的配置流程，并会简述在主机和从机模式下软件和硬件 CS 的输入输出方式。

#### 硬件 CS 配置流程：

当 SPI 作主机，硬件 CS 输出时，HWCSOE 位置 1，SWCSEN 置 0，开启硬件 CS 控制，SPI 在使能之

后会在 CS 管脚上输出低电平，在 SPI 关闭并且发送完成后，释放 CS 信号。

当 SPI 作主机，硬件 CS 输入时，HWCSOE 位置 0，SWCSEN 置 0，开启硬件 CS 控制，此时一旦主机 SPI 检测到 CS 管脚为低电平时，SPI 硬件自动关闭 SPI 并进入从机模式，模式错误标志 MMERR 同时置位，若使能了错误中断（ERRIE=1），将会产生中断，在 MMERR 置位期间，硬件不允许软件置位 SPIEN 和 MSTEN 位，通过读或写 SPI 状态寄存器（SPI\_STS）再写 SPI 控制寄存器 1（SPI\_CTRL1）可以清除 MMERR。

当 SPI 作从机，硬件 CS 输入时，HWCSOE 位置 0，SWCSEN 置 0，开启硬件 CS 控制，从机根据 CS 管脚上的电平判断是否发送或接收数据，只有 CS 管脚上为低电平时，从机才会被选中并进行数据的收发。

#### 软件 CS 配置流程：

当 SPI 作主机，软件 CS 输入时，SWCSEN 位置 1，开启软件 CS 控制，当 SWCSIL 位置 0 时，SPI 硬件自动关闭 SPI 并进入从机模式，模式错误标志 MMERR 同时置位，若使能了错误中断（ERRIE=1），将会产生中断，在 MMERR 置位期间，硬件不允许软件置位 SPIEN 和 MSTEN 位，通过读或写 SPI 状态寄存器（SPI\_STS）再写 SPI 控制寄存器 1（SPI\_CTRL1）可以清除 MMERR。

当 SPI 作从机，软件 CS 输入时，SWCSEN 位置 1，开启软件 CS 控制，SPI 根据 SWCSIL 位判断 CS 信号电平，不使用 CS 管脚，当 SWCSIL=0 时，从机才会被选中并进行数据的收发。

### 13.2.4 SPI\_SCK控制器简述和配置流程

SPI 协议采用同步传输，所以 SPI 接口在作为 SPI 使用时，作主机时，需要产生通信时钟用于 SPI 接口的数据收发，并且需要将该通信时钟通过 IO 输出给从机，用于从机的数据收发；作从机时，需要外设提供通信时钟从 IO 输入到 SPI 接口内部作为通信时钟使用，所以实际上，SPI\_SCK 控制器便是扮演着产生 SPI\_SCK 以及分配 SPI\_SCK 的角色，详细的配置方法如下所述。

#### SPI\_SCK 控制器配置流程：

- 时钟极性相位选择：配置 CLKPOL,CLKPHA 选择需要的极性和相位。
- 时钟分频系数选择：配置 CRM 选择需要的 PCLK 频率，配置 MDIV[3: 0]选择需要的分频系数。
- 主机或从机选择：配置 MSTEN 选择 SPI 作主机或从机使用，注意主机只收模式在 SPI 使能后就会开始输出时钟，直到 SPI 被关闭且接收完成。

### 13.2.5 CRC简述和配置流程

SPI 接口内部具有独立的发送和接收 CRC 计算单元，通过软件编程配置，SPI 接口在作为 SPI 使用时，可以同时在用户使用 DMA 读写数据或 CPU 读写数据的情况下，自动进行 CRC 计算以及 CRC 校验，如果在传输过程中，硬件检测到接收到的数据与 SPIRxCRC 寄存器（SPI\_RCRC）中的数据不符，且该笔数据又是 CRC 数据时，CCERR 位会置起，若使能了错误中断（ERRIE=1），将会产生中断。

下面分 DMA 和 CPU 操作数据寄存器分别描述 SPI 的 CRC 功能以及 CRC 配置流程。

#### CRC 配置流程

- CRC 计算多项式配置：配置 SPICRC 多项式寄存器（SPI\_CPOLY）选择 CRC 计算多项式。
- 使能 CRC：置起 CCEN 位使能 CRC 计算，该操作将会复位 SPIRxCRC 寄存器（SPI\_RCRC）以及 SPITxCRC 寄存器（SPI\_TCRC）。
- 根据 DMA 或 CPU 操作数据寄存器选择是否以及何时置位 NTC 位，具体请参见下方描述。

#### DMA 发送模式：

在采用 DMA 写入待发送的数据时，当使能 CCEN 后，硬件会根据 SPICRC 多项式寄存器（SPI\_CPOLY）中的值以及每笔发送的数据自动计算 CRC 值，并在最后一笔数据发送完成后自动发送 CRC 值，该值即 SPITxCRC 寄存器（SPI\_TCRC）中的值。

#### DMA 接收模式：

在采用 DMA 读取待接收的数据时，当使能 CCEN 后，硬件会根据 SPICRC 多项式寄存器（SPI\_CPOLY）中的值以及每笔接收的数据自动计算 CRC 值，并在最后一笔数据接收完成后等待 CRC 数据接收完成，并将收到的 CRC 值和 SPIRxCRC 寄存器（SPI\_RCRC）中的值作比较，若校验出错，会置起 CCERR 标志，若使能了 ERRIE 位，则产生错误中断。

#### CPU 发送模式：

相较于 DMA 发送模式，该模式需要软件在写入最后一笔待发送的数据后，在最后一笔数据发送完成之前

置起 NTC 位。

#### CPU 接收模式

在双线单向全双工模式下，按照 CPU 发送模式操作 NTC 位，CPU 接收模式的 CRC 计算和校验会自动完成，在单线单向只接收以及单线双向只接收模式下，相较于 DMA 接收需要软件在接收到倒数第二笔数据之后，接收到最后一笔数据之前置起 NTC 位。

### 13.2.6 DMA 传输简述和配置流程

SPI 接口支持使用 DMA 进行发送数据的写入，接收数据的读取，具体配置流程分别见下述的 DMA 发送配置流程以及 DMA 接收配置流程。

需要特别注意的是，在开启CRC计算和校验时，DMA发送数据的个数配置为待发送的数据个数，DMA读取数据的个数配置为待接收的数据个数，此时硬件在所有数据传输完毕后自动进行CRC传输，且接收方还会自动进行CRC校验，需要注意，接收到的CRC数据，硬件会搬到SPI数据寄存器（SPI\_DT）中，并置位RDBF，以及在开启了DMA传输时发出DMA读请求，所以这里推荐当CRC接收完毕后软件要去读DT寄存器来取走CRC值，防止后续传输出错。

#### DMA 发送配置流程：

- 选择 DMA 传输通道：在 DMA 章节 DMA 通道映射表中选择用于当前所用 SPI 的 DMA 通道。
- 配置 DMA 传输目标地址：在 DMA 控制寄存器（TMRx\_DMACTRL）中 DMA 传输目的地址位写入当前所使用的 SPI 的 SPI 数据寄存器（SPI\_DT）地址，DMA 将会在接收到发送请求后将待发送的数据写入该地址。
- 配置 DMA 传输源地址：在 DMA 控制寄存器（TMRx\_DMACTRL）中 DMA 传输源地址位写入待发送数据存放的地址，DMA 将会在接收到发送请求后将该地址内的数据写入到目标地址中，即写入到当前所使用的 SPI 的 SPI 数据寄存器（SPI\_DT）中。
- 配置 DMA 传输数据个数：在 DMA 控制寄存器（TMRx\_DMACTRL）相关位置配置期望传输的数据个数
- 配置 DMA 传输通道优先级：在 DMA 控制寄存器（TMRx\_DMACTRL）相关位置配置当前所使用通道的 SPI 的 DMA 传输通道优先级。
- 配置 DMA 中断产生时机：在 DMA 控制寄存器（TMRx\_DMACTRL）相关位置配置是在传输完成或传输完成一半时产生 DMA 中断。
- 使能 DMA 传输通道：在 DMA 控制寄存器（TMRx\_DMACTRL）相关位置使能当前所选用的 DMA 通道

（TMRx\_DMACTRL）

#### DMA 接收配置流程：

- 选择 DMA 传输通道：在 DMA 章节 DMA 通道映射表中选择用于当前所用 SPI 的 DMA 通道。
- 配置 DMA 传输目标地址：在 DMA 控制寄存器（TMRx\_DMACTRL）中 DMA 传输目的地址位写入期望存放接收数据的地址，DMA 将会在接收到接收请求后，将当前所使用的 SPI 的 SPI 数据寄存器（SPI\_DT）中的数据存放在目的地址中。
- 配置 DMA 传输源地址：在 DMA 控制寄存器（TMRx\_DMACTRL）中 DMA 传输源地址位写入当前所使用的 SPI 的 SPI 数据寄存器（SPI\_DT）的地址，DMA 将会在接收到接收请求后将该地址内的数据写入到目标地址中，即写入到期望存放接收数据的地址。
- 配置 DMA 传输数据个数：在 DMA 控制寄存器（TMRx\_DMACTRL）相关位置配置期望传输的数据个数。
- 配置 DMA 传输通道优先级：在 DMA 控制寄存器（TMRx\_DMACTRL）相关位置配置当前所使用通道的 SPI 的 DMA 传输通道优先级。
- 配置 DMA 中断产生时机：在 DMA 控制寄存器（TMRx\_DMACTRL）相关位置配置是在传输完成或传输完成一半时产生 DMA 中断。
- 使能 DMA 传输通道：在 DMA 控制寄存器（TMRx\_DMACTRL）相关位置使能当前所选用的 DMA 通道

### 13.2.7 发送器简述和配置流程

SPI 发送器时钟由 SPI\_SCK 控制器提供，根据软件编程配置，发送器可以输出不同的数据帧格式，SPI 具有一个数据缓冲寄存器 SPI\_DT，软件需要将待发送的数据先写入 SPI\_DT，发送器在有时钟时，会把 SPI 数据寄存器（SPI\_DT）中的数据保存到发送器中的数据缓冲器(有别于 SPI 数据寄存器（SPI\_DT），

SPI 发送器中的数据缓冲器由 SPI\_SCK 驱动, 且硬件自动控制, 软件不可操作), 并按照配置好的帧格式将数据依次发出。

用户可以选择 DMA 或 CPU 来控制数据的写入, 若选择 DMA 传输, 详细配置请参见 DMA 传输章节, 若选择 CPU 传输, 则用户需要判断 TDBE 位, 该位复位值为 1, 代表 SPI\_DT 为空, 若 TDBEIE 置位, 则产生中断, 数据写入后, TDBE 拉低, 直到数据被同步到发送器中的数据缓冲器后, TDBE 再次被拉起, 即用户只可以在 TDBE 置位时写入待发送的数据。

发送器配置完成并使能 SPI 后, SPI 将进入数据发送状态, 所以此之前, 应需要参考全双工半双工章节配置通信选用的是全双工或半双工等, 并参考 CS 控制器章节配置选用的 CS 控制模式, 还需要参考 SPI\_SCK 控制章节配置通信时钟, 若使用了 CRC 以及 DMA, 还需参考 CRC 以及 DMA 传输章节配置 CRC 以及 DMA, 如下为推荐的发送器配置流程。

#### 发送器配置流程:

- 配置全双工半双工选择器。
- 配置 CS 控制器
- 配置 SPI\_SCK 控制器
- 配置 CRC (若需要使用 CRC 自动计算和校验功能)
- 配置 DMA 传输 (若需要使用 DMA 传输功能)
- 若没有选择 DMA 传输功能, 软件需要判断 TDBE 位, 软件需要根据需求判断是否要打开发送数据中断, 即置位 TDBEIE。
- 配置帧格式: 配置 LTF 位选择 MSB/LSB 格式, 配置 FBN 选择 8/16 位数据。
- 置位 SPIEN 位使能 SPI

### 13.2.8 接收器简述和配置流程

SPI 接收器时钟由 SPI\_SCK 控制器提供, 根据软件编程配置, 接收器可以接收不同的数据帧格式, SPI 接收器具有一个接收数据缓冲寄存器, 该寄存器由 SPI\_SCK 驱动, 在每笔传输的最后一个 CLK, 数据从移位寄存器压入该接收数据缓冲寄存器, 随后发送器会给出数据接收完成的标志给到 SPI 的控制逻辑, SPI 的控制逻辑在检测到该标志后会自动把接收器中的数据缓冲器中的值压入 SPI\_DT, RDBF 随之置起, 这意味着有数据被收到, 且该数据已被压入 SPI 数据寄存器(SPI\_DT), 此时读 SPI 数据寄存器(SPI\_DT)可以读出该笔数据, 同时 RDBF 随之清除。

用户可以选择 DMA 或 CPU 来控制数据的读出, 若选择 DMA 传输, 详细配置请参见 DMA 传输章节, 若选择 CPU 传输, 则用户需要判断 RDBF 位, 该位复位值为 0, 代表 SPI\_DT 为空, 当有数据被接收到, 且数据被移入 SPI 数据寄存器 (SPI\_DT) 时, RDBF 置位, 代表 SPI 数据寄存器 (SPI\_DT) 内有数据等待读取, 此时若 RDBFIE 置位则产生中断。

若在下一笔接收器接收到的数据准备压入 SPI 数据寄存器(SPI\_DT)时, 之前接收到的数据仍未被读走, 即 RDBF 仍为 1, 则代表数据溢出, 在此之前接收到的数据不会丢失, 但之后的数据都将丢失, 此时 ROERR 置起, 若 ERRIE 置位, 则产生错误中断, 依次读 SPI 数据寄存器(SPI\_DT)和 SPI 状态寄存器(SPI\_STS)可将 ROERR 清除, 如下为推荐的接收器配置流程。

#### 接收器配置流程:

- 配置全双工半双工选择器。
- 配置 CS 控制器。
- 配置 SPI\_SCK 控制器。
- 配置 CRC (若需要使用 CRC 自动计算和校验功能)。
- 配置 DMA 传输 (若需要使用 DMA 传输功能)。
- 若没有选择 DMA 传输功能, 软件需要判断 RDBF 位, 软件需要根据需求判断是否要打开接收数据中断, 即置位 RDBFIE。
- 配置帧格式: 配置 LTF 位选择 MSB/LSB 格式, 配置 FBN 选择 8/16 位数据。
- 置位 SPIEN 位使能 SPI。

### 13.2.9 Motorola模式通信时序

本节介绍 SPI 通信时序, 包括全双工和半双工的主/从通信时序。

#### 全双工通信-主机通信时序

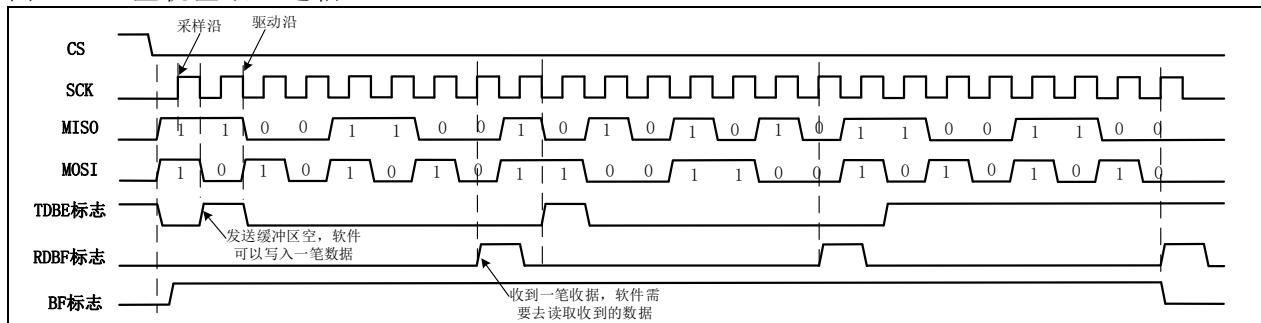
其中主机端配置如下:

MSTEN=1: 设备为主机;

SLBEN=0: 全双工模式;

CLKPOL=0, CLKPHA=0: SCK 空闲输出低电平, 第一个边沿作为采样边沿;  
 FBN=0: 帧数据的长度为 8 位;  
 主机发送数据 (MOSI): 0xaa, 0xcc, 0xaa  
 从机发送数据 (MISO): 0xcc, 0xaa, 0xcc

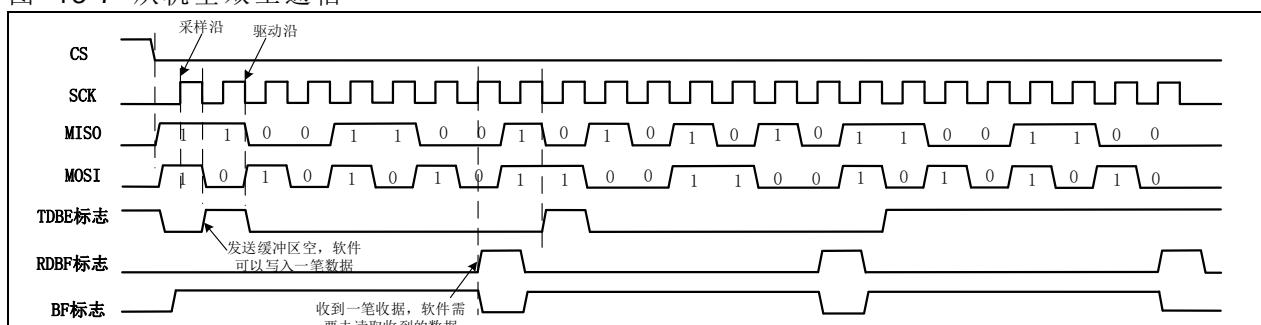
图 13-6 主机全双工通信



### 全双工通信-从机通信时序

其中从机端配置如下:  
 MSTEN=0: 设备为从机;  
 SLBEN=0: 全双工模式;  
 CLKPOL=0, CLKPHA=0: SCK 空闲输出低电平, 第一个边沿作为采样边沿;  
 FBN=0: 帧数据的长度为 8 位;  
 主机发送数据 (MOSI): 0xaa, 0xcc, 0xaa  
 从机发送数据 (MISO): 0xcc, 0xaa, 0xcc

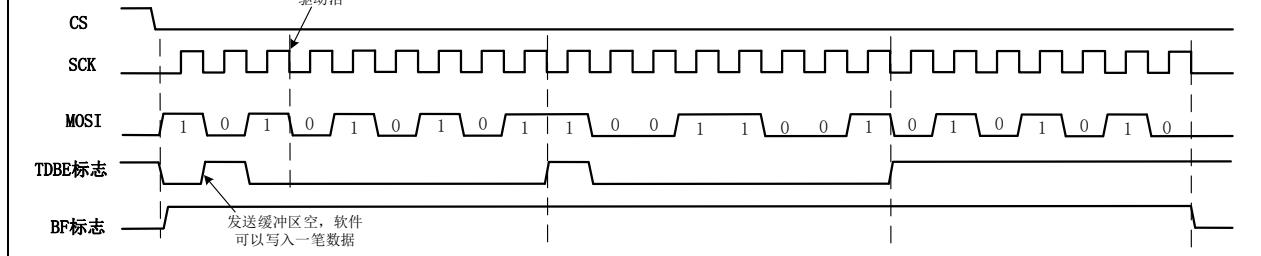
图 13-7 从机全双工通信



### 半双工通信-主机发送时序

MSTEN=1: 设备为主机;  
 SLBEN=1: 单线双向模式;  
 SLBTD=1: 发送模式;  
 CLKPOL=0, CLKPHA=0: SCK 空闲输出低电平, 第一个边沿为采样边沿;  
 FBN=0: 帧数据的长度为 8 位;  
 主机发送数据: 0xaa, 0xcc, 0xaa

图 13-8 主机半双工发送通信



### 半双工通信-从机接收时序

MSTEN=0: 设备为从机;

SLBEN=1: 单线双向模式;

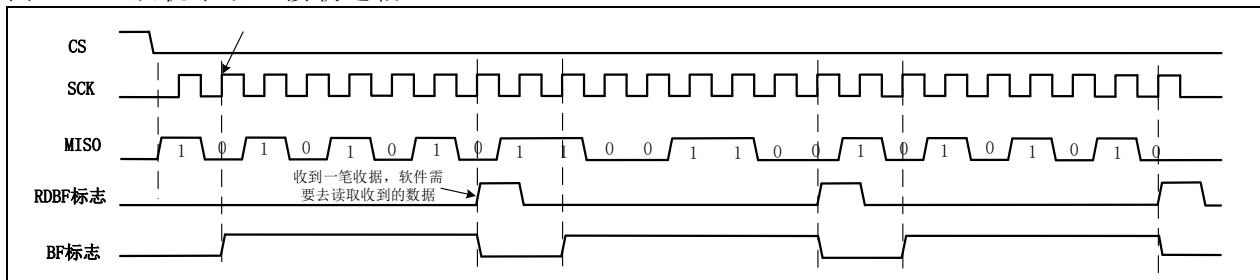
SLBTD=0: 接收模式;

CLKPOL=0, CLKPHA=0: SCK 空闲输出低电平, 第一个边沿为采样边沿;

FBN=0: 帧数据的长度为 8 位;

从机接收数据: 0xaa, 0xcc, 0xaa

图 13-9 从机半双工接收通信



#### 半双工通信-从机发送时序

MSTEN=0: 设备为从机;

SLBEN=1: 单线双向模式;

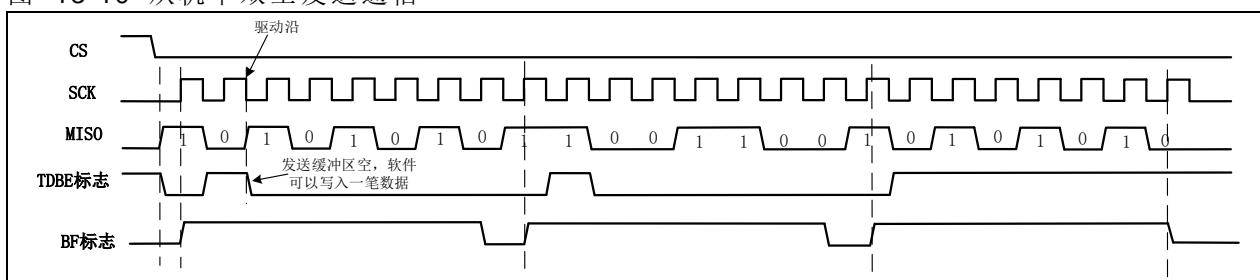
SLBTD=1: 发送模式;

CLKPOL=0, CLKPHA=0: SCK 空闲输出低电平, 第一个边沿为采样边沿;

FBN=0: 帧数据的长度为 8 位;

从机发送数据: 0xaa, 0xcc, 0xaa

图 13-10 从机半双工发送通信



#### 半双工通信-主机接收时序

MSTEN=1: 设备为主机;

SLBEN=1: 单线双向模式;

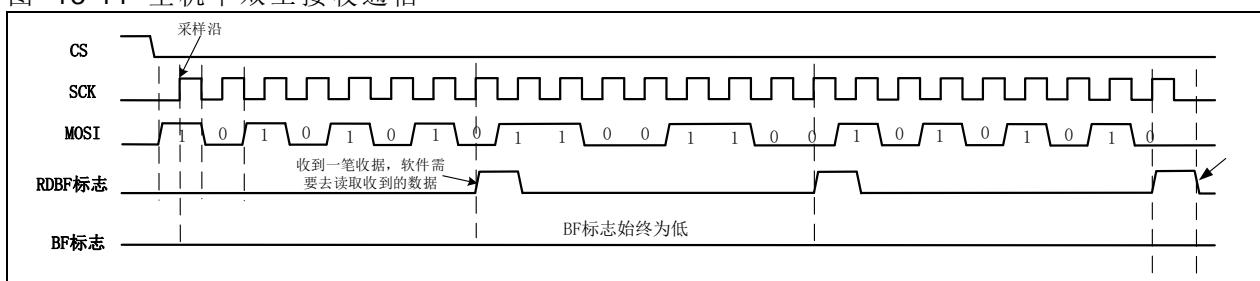
SLBTD=0: 接收模式;

CLKPOL=0, CLKPHA=0: SCK 空闲输出低电平, 第一个边沿为采样边沿;

FBN=0: 帧数据的长度为 8 位;

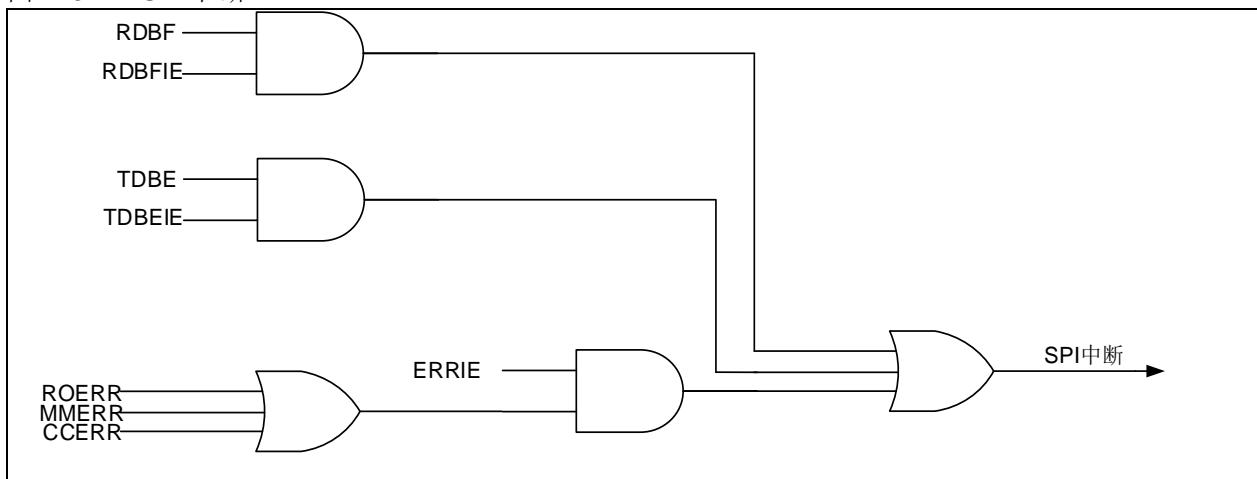
主机接收数据: 0xaa, 0xcc, 0xaa

图 13-11 主机半双工接收通信



### 13.2.10 中断

图 13-12 SPI中断



### 13.2.11 IO管脚控制

SPI 接口作为 SPI 使用时最多可有 4 根管脚与外设相连，各管脚的使用方法可以参见全双工半双工选择器简述和配置流程以及 CS 控制器简述和配置流程章节，各管脚的定义如下。

- **MISO:** 主机输入/从机输出管脚。在 SPI 接口作 SPI 主机使用时，从机送出的数据从该管脚输入。在 SPI 接口作 SPI 从机使用时，从机待发送的数据从该管脚输出。
- **MOSI:** 主设备输出/从设备输入管脚。在 SPI 接口作 SPI 主机使用时，主机待发送的数据从该管脚输出。在 SPI 接口作 SPI 从机使用时，主机送出的数据从该管脚输入。
- **SCK:** SPI 的通信时钟管脚。在 SPI 接口作 SPI 主机使用时，通信时钟从此管脚输出送给外设。在 SPI 接口作 SPI 从机使用时，主机提供的通信时钟从该管脚输入以作为 SPI 接口的通信时钟。
- **CS:** 片选信号。这是一个可选的管脚，用来选中主/从设备，具体使用方式可以参见 CS 控制器章节。

### 13.2.12 注意事项

CRC 接收完成后要软件读 DT 寄存器来读出 CRC 值。

## 13.3 SPI寄存器

必须以字（32 位）的方式操作这些外设寄存器。

表 13-1 SPI寄存器映像和复位值

寄存器简称	基址偏移量	复位值
SPI_CTRL1	0x00	0x0000
SPI_CTRL2	0x04	0x0000
SPI_STS	0x08	0x0002
SPI_DT	0x0C	0x0000
SPI_CPOLY	0x10	0x0007
SPI_RCRC	0x14	0x0000
SPI_TCRC	0x18	0x0000

### 13.3.1 SPI控制寄存器1 (SPI\_CTRL1)

域	简称	复位值	类型	功能
位 15	SLBEN	0x0	rw	单线双向半双工模式使能 (Single line bidirectional half-duplex enable) 0: 关闭;

				1: 开启。 单线双向半双工模式传输方向 (Single line bidirectional half-duplex transmission direction) 和 SLBEN 位一起决定在“单线双向半双工”模式下数据的传输方向 0: 接收模式; 1: 发送模式。
位 14	SLBTD	0x0	rw	CRC 校验使能 (CRC calculation enable) 0: 关闭; 1: 开启。
位 13	CCEN	0x0	rw	下一笔传输数据为 CRC (Next transmission CRC) 该位置起表示下一笔传输的数据为 CRC 数据。 0: 普通数据; 1: CRC 数据。
位 12	NTC	0x0	rw	帧位个数 (frame bit num) 该位配置发送/接收时数据帧位个数。 0: 8 位; 1: 16 位。
位 11	FBN	0x0	rw	仅接收有效 (Only receive active) 在“双线单向”模式时，该位置起表示只有接收有效，发送被禁止。 0: 发送和接收; 1: 仅接收。
位 10	ORA	0x0	rw	软件 CS 模式使能 (Software CS enable) 当该位被置起时，CS 管脚上的电平由 SWCSIL 位的值决定，此时在 CS 管脚上的 I/O 电平状态无效。 0: 关闭; 1: 开启。
位 9	SWCSEN	0x0	rw	软件 CS 内部电平 (Software CS internal level) 该位只在 SWCSEN 位置起时有意义，它决定了 CS 上的内部电平状态。 做主设备时，该位必须设置置起。 0: 低电平; 1: 高电平。
位 8	SWCSIL	0x0	rw	LSB 先传输 (LSB transmit first) 该位用于选择数据先传输 MSB 还是 LSB。 0: MSB; 1: LSB。
位 7	LTF	0x0	rw	SPI 使能 (SPI enable) 0: 关闭; 1: 开启。
位 6	SPIEN	0x0	rw	主模式时钟频率分频系数 (Master clock frequency division) 作主模式时，分频系数对外设时钟进行分频，作为 SPI 时钟，MDIV[3]位在 SPI 控制寄存器 2 (SPI_CTRL2) , MDIV[3: 0]: 0000: 2 分频 0001: 4 分频 0010: 8 分频 0011: 16 分频 0100: 32 分频 0101: 64 分频 0110: 128 分频 0111: 256 分频 1000: 512 分频 1001: 1024 分频
位 5: 3	MDIV	0x0	rw	主模式使能 (Master enable) 0: 关闭 (从设备); 1: 开启 (主设备)。
位 2	MSTEN	0x0	rw	时钟极性 (Clock polarity) 空闲时时钟输出的极性。
位 1	CLKPOL	0x0	rw	

				0: 低电平; 1: 高电平。
位 0	CLKPHA	0x0	rw	时钟相位 (Clock phase) 0: 第一个边沿进行数据捕获; 1: 第二个边沿进行数据捕获。

### 13.3.2 SPI控制寄存器2 (SPI\_CTRL2)

域	简称	复位值	类型	功能
位 15: 9	保留位	0x00	resd	硬件强制为 0
位 8	MDIV	0x0	rw	主模式时钟频率分频系数 (Master clock frequency division) 详见 MDIV[2: 0]在 SPI 控制寄存器 1 (SPI_CTRL1)。
位 7	TDBEIE	0x0	rw	发送数据缓冲器空中断使能 (Transmit data buffer empty interrupt enable) 0: 关闭; 1: 开启。
位 6	RDBFIE	0x0	rw	接收数据缓冲器满中断使能 (Receive data buffer full interrupt enable) 0: 关闭; 1: 开启。
位 5	ERRIE	0x0	rw	错误中断使能 (Error interrupt enable) 当错误 (CCERR、MMERR、ROERR、TUERR) 产生时, 该位控制是否产生中断 0: 关闭; 1: 开启。
位 4: 3	保留位	0x0	resd	保持默认值。
位 2	HWCSONE	0x0	rw	硬件 CS 输出使能 (Hardware CS output enable) 该位做主设备时才有意义, 设置为'1'时, CS 脚 I/O 口输出低电平, 设置为'0'时, 必须保证 CS 脚 I/O 口输入为高电平。 0: 关闭; 1: 开启。
位 1	DMATEN	0x0	rw	DMA 发送使能 (DMA transmit enable) 0: 关闭; 1: 开启。
位 0	DMAREN	0x0	rw	DMA 接收使能 (DMA receive enable) 0: 关闭; 1: 开启。

### 13.3.3 SPI状态寄存器 (SPI\_STS)

域	简称	复位值	类型	功能
位 15: 8	保留位	0x00	resd	硬件强制为 0
位 7	BF	0x0	ro	通信忙标志 (Busy flag) 0: 通信空闲; 1: 通信忙。
位 6	ROERR	0x0	ro	接收器溢出错误 (Receiver overflow error) 0: 无; 1: 有。
位 5	MMERR	0x0	ro	主模式错误 (Master mode error) 该位由硬件置位, 软件清除 (先读或写 SPI 状态寄存器 (SPI_STS), 再写 SPI 控制寄存器 1 (SPI_CTRL1))。 0: 无; 1: 有。
位 4	CCERR	0x0	rw0c	CRC 校验错误 (CRC calculation error) 该位由硬件置起, 由软件清除。 0: 正确; 1: 错误。

位 3	TUERR	0x0	ro	发送器欠载错误 (Transmitter underload error) 该位由硬件置起，软件清除（读 SPI 状态寄存器 (SPI_STS)）。 0: 无； 1: 有。
位 2	ACS	0x0	ro	音频通道状态 (Audio channel state) 该位表示当前传输的音频左右声道状态。 0: 左声道； 1: 右声道。
位 1	TDBE	0x1	ro	发送数据缓冲器空 (Transmit data buffer empty) 0: 非空； 1: 空。
位 0	RDBF	0x0	ro	接收数据缓冲器满 (Receive data buffer full) 0: 未满； 1: 满。

### 13.3.4 SPI数据寄存器 (SPI\_DT)

域	简称	复位值	类型	功能
位 15: 0	DT	0x0000	rw	数据值 (Data value) 该寄存器包含读和写的功能，当数据位配置为 8 位时，该寄存器只有低 8 位[7: 0]有效。

### 13.3.5 SPICRC多项式寄存器 (SPI\_CPOLY)

域	简称	复位值	类型	功能
位 15: 0	CPOLY	0x0007	rw	CRC 多项式寄存器 (CRC polynomial) 该寄存器为 CRC 计算时用到的多项式，可以根据应用设置。

### 13.3.6 SPIRxCRC寄存器 (SPI\_RCRC)

域	简称	复位值	类型	功能
位 15: 0	RCRC	0x0000	ro	接收 CRC 寄存器 (receive CRC) CRC 使能后，该寄存器值为根据接收到的数据计算得到的 CRC 值，要复位该寄存器，需操作 SPI 控制寄存器 1 (SPI_CTRL1) 的 CCEN 位先清除再置起。 当数据位配置为 8 位时，该寄存器只有低 8 位[7: 0]有效，按照 CRC8 计算；当数据位配置为 16 位时，按照 CRC16 计算。

### 13.3.7 SPITxCRC寄存器 (SPI\_TCRC)

域	简称	复位值	类型	功能
位 15: 0	TCRC	0x0000	ro	发送 CRC 寄存器 (transmit CRC) CRC 使能后, 该寄存器值为根据发送的数据计算得到的 CRC 值。要复位该寄存器, 需操作 SPI 控制寄存器 1 (SPI_CTRL1) 的 CCEN 位先清除再置起。 当数据位配置为 8 位时, 该寄存器只有低 8 位[7: 0]有 效, 按照 CRC8 计算; 当数据位配置为 16 位时, 按照 CRC16 计算。

## 14 定时器 (TIMER)

AT32WB415 定时器种类有基本定时器、通用定时器、高级控制定时器，详细功能模式可参考 [14.1~14.3](#) 节说明，下表为各种类型定时器的功能总表。

表 14-1 TMR功能对比

Timer	类型	Timer	计数位数	计数方式	重复计数器	预分频系数	DMA 请求产生	捕获/比较通道	PWM 输入模式	EXT 输入	刹车输入
高级控制定时器		TMR1	16	向上 向下 向上/向下	8 位	1~65535	支持	4	支持	支持	支持
通用定时器	TMR2 TMR5	16/32		向上 向下 向上/向下	不支持	1~65535	支持	4	支持	仅 TMR2 支持	不支持
				向上 向下 向上/向下						仅 TMR3 支持	不支持
	TMR4	16		向上 向下 向上/向下	不支持	1~65535	支持	4	支持	不支持	不支持
	TMR9	16		向上	不支持	1~65535	不支持	2	支持	不支持	不支持
	TMR10 TMR11	16		向上	不支持	1~65535	不支持	1	不支持	不支持	不支持
Timer	类型	Timer	计数位数	计数方式	PWM 输出	单周期输出	互补输出	死区	编码器接口连接	霍尔传感器接口连接	连动外设
高级控制定时器		TMR1	16	向上 向下 向上/向下	支持	支持	支持	支持	支持	支持	定时器同步/ADC
通用定时器	TMR2 TMR5	16/32		向上 向下 向上/向下	支持	支持	不支持	不支持	支持	支持	定时器同步/ADC
				向上 向下 向上/向下							定时器同步/ADC
	TMR4	16		向上 向下 向上/向下	支持	支持	不支持	不支持	支持	支持	定时器同步/ADC
	TMR9	16		向上	支持	支持	不支持	不支持	不支持	不支持	定时器同步
	TMR10 TMR11	16		向上	支持	支持	不支持	不支持	不支持	不支持	无

### 14.1 通用定时器 (TMR2到TMR5)

#### 14.1.1 TMRx简介

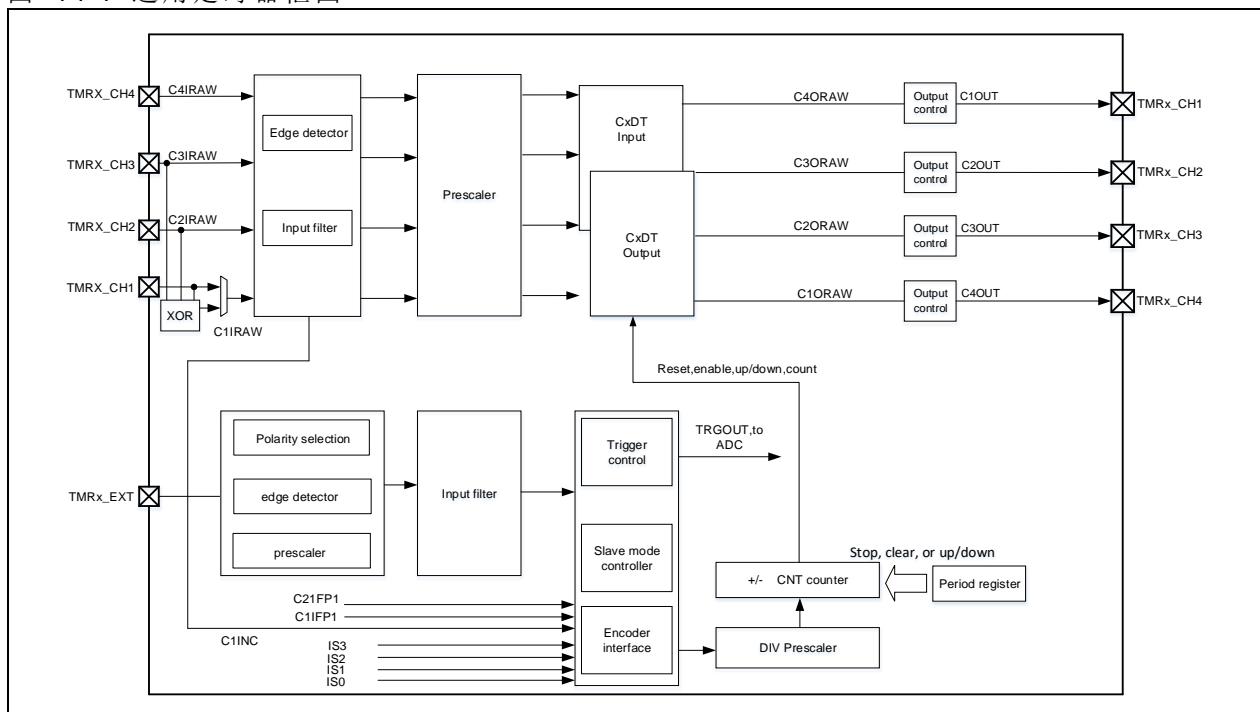
通用定时器 TMR2 到 TMR5 包含一个支持向上、向下、中央双向对齐计数的 16 位计数器、4 个捕获/比较寄存器、4 组独立的通道。可实现输入捕获、可编程 PWM 输出。

#### 14.1.2 TMRx主要功能

- 可选内部、外部、内部触发输入用作计数时钟
- 16 位支持向上、向下、双向、编码器模式的计数器 (TMR2/5 可扩展至 32 位)
- 4 组独立通道，支持输入捕获、输出比较、PWM 生成、单周期模式。
- 定时器之间可互联同步

- 支持溢出事件、触发事件、通道事件触发中断/DMA
- 支持 TMR burst DMA 传输

图 14-1 通用定时器框图



## 14.1.3 TMRx功能描述

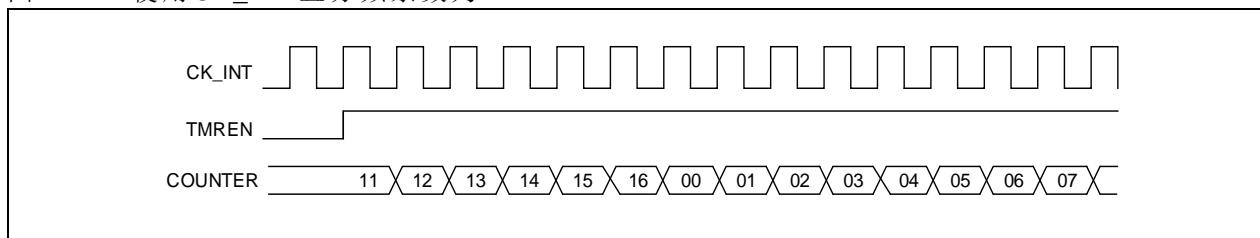
### 14.1.3.1 计数时钟

TMR2 至 5 计数时钟可从内部时钟 (CK\_INT)、外部时钟 (外部时钟模式 A、B)、内部触发输入 (ISx) 这些时钟源提供。

#### 内部时钟 (CK\_INT)

默认下使用 CK\_INT 经由预分频器驱动计数器计数。

图 14-2 使用 CK\_INT 且分频系数为 1



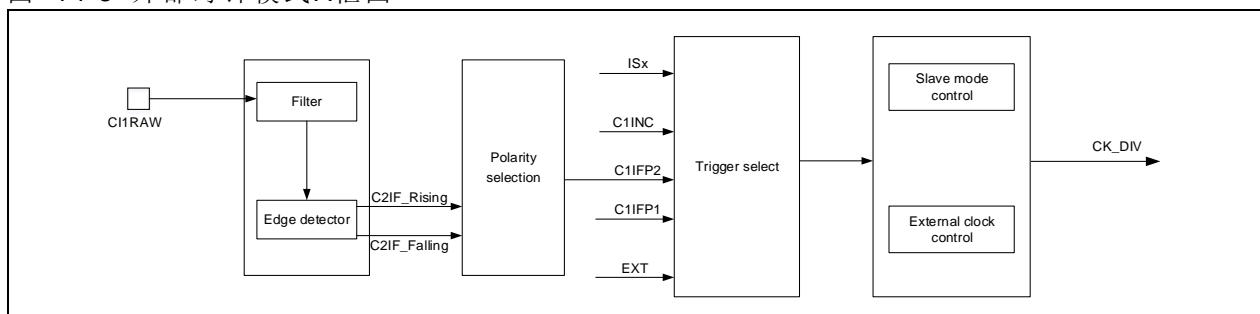
#### 外部时钟 (TRGIN/EXT)

计数时钟可由两种外部时钟源提供，分别为 TRGIN 和 EXT 信号。

当 SMSEL=3'111 时，外部时钟模式 A 被选中，配置 STIS[2: 0]来选择 TRGIN 信号驱动计数器计数。

当 ECMBEN=1 时，外部时钟模式 B 被选中，计数器由 EXT 信号驱动计数。

图 14-3 外部时钟模式A框图



注：由于同步逻辑，输入端信号与计数器实际时钟之间存在一定延时。

图 14-4 使用外部时钟模式A计数

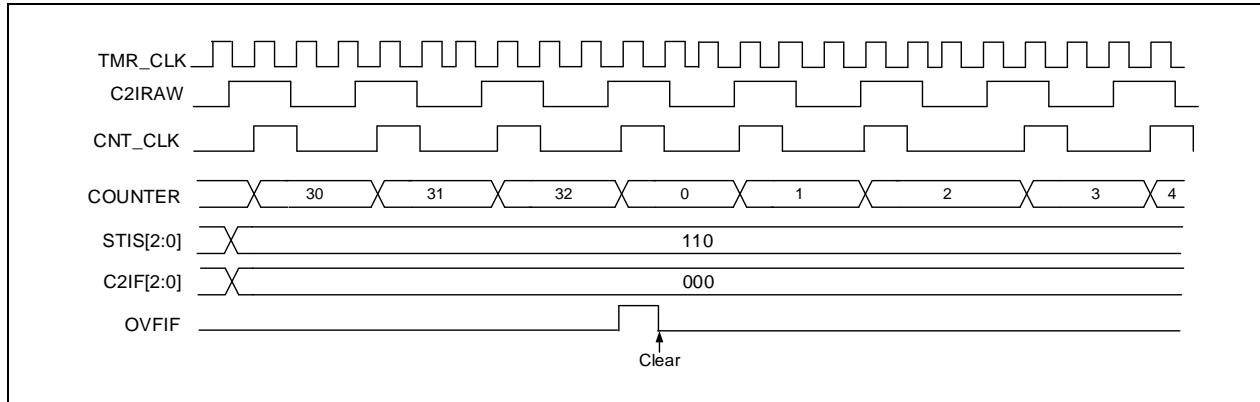
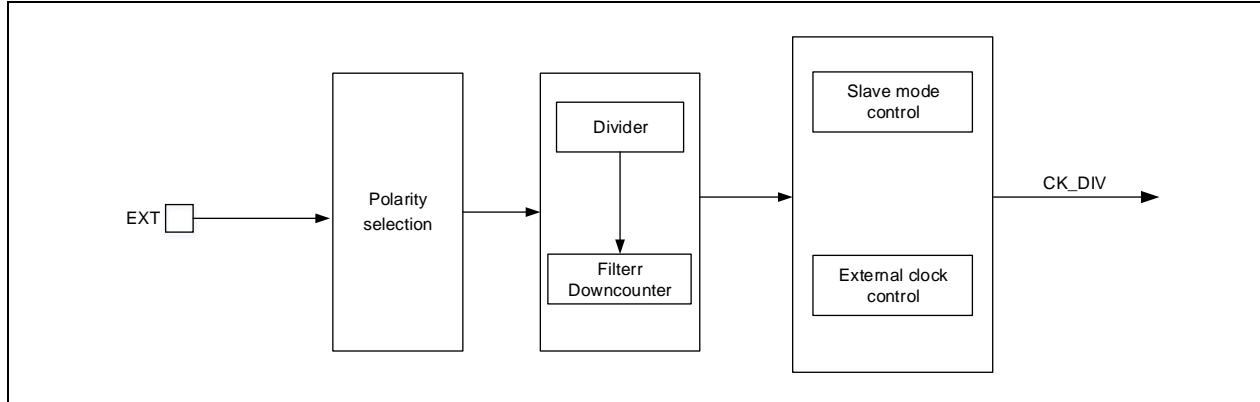
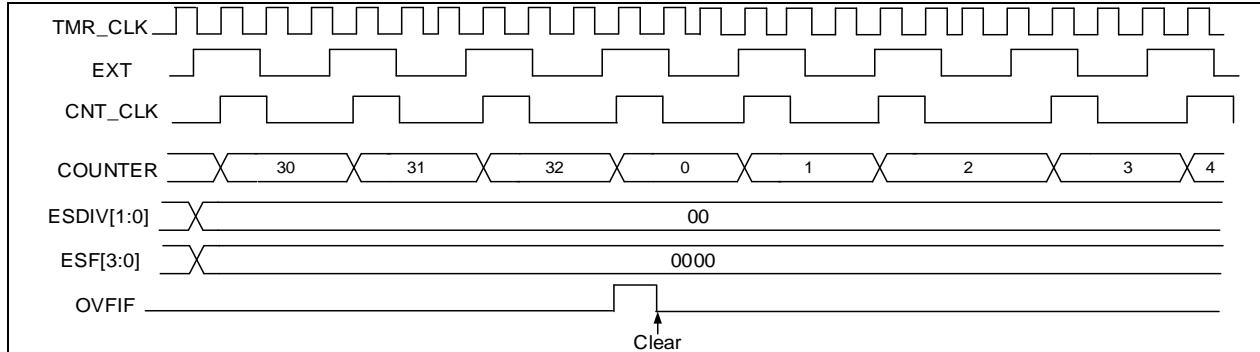


图 14-5 外部时钟模式B框图



注：由于同步逻辑。输入端 EXT 信号与计数器实际时钟之间存在一定延时。

图 14-6 使用外部时钟模式B计数



### 内部触发输入 (ISx)

定时器之间支持互联同步，因此一个定时器的 TMR\_CLK 可由另一个定时器输出信号 TRGOUT 提供。配置 STIS[2: 0]选择内部触发信号驱动计数器计数。

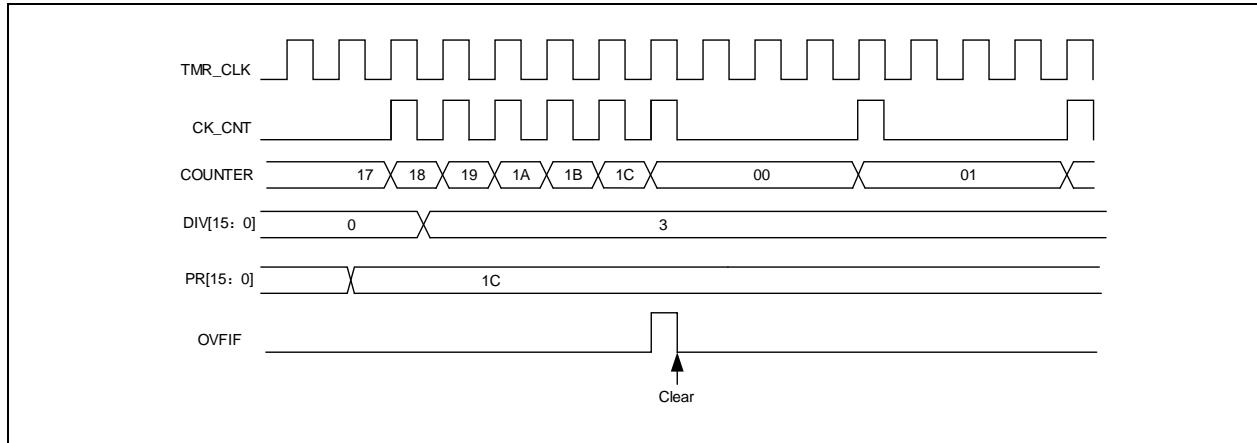
TMR2 至 5 定时器内含一个 16 位预分频器，用于产生驱动计数器计数的时钟 CK\_CNT，通过配置 TMRx\_DIV 寄存器值，可灵活调整 CK\_CNT 与 TMR\_CLK 之间的分频关系。预分频值可在任何时刻修改，但只在下一个溢出事件发生时，新值才会生效。

表 14-2 TMRx内部触发连接

次定时器	IS0 (STIS = 000)	IS1 (STIS = 001)	IS2 (STIS = 010)	IS3 (STIS = 011)
TMR2	TMR1	USB_SOF	-	TMR4
-	TMR1	TMR2	TMR5	TMR4
TMR4	TMR1	TMR2	-	-
TMR5	TMR2	-	TMR4	-

注意 1：如果某个产品中没有相应的定时器，则对应的触发信号 ISx 也不存在。

图 14-7 当预分频器的参数从1变到4时，计数器的时序图



### 14.1.3.2 计数模式

TMR2 至 5 定时器提供了多种计数模式，用来满足不同的应用场景。其内部拥有一个支持 16 位向上计、向下、中央双向对齐计数的计数器，TMR2/5 可通过将 PMEN 位置 1 扩展至 32 位，计数器的值可由周期寄存器（TMRx\_PR）载入。默认下，周期寄存器（TMRx\_PR）值会立即传入它的影子寄存器；当开启周期缓冲功能后（PRBEN 置 1），周期寄存器（TMRx\_PR）值会在溢出事件发生时传入它的影子寄存器。溢出事件由 OVFEN、OVFS 位配置。

将 TMREN 位置 1 可使能定时器计数，由于同步逻辑，实际驱动计数器的使能信号 TMR\_EN 相对于 TMREN 延迟一个时钟周期。

#### 向上计数模式

向上计数模式中，当计数值达到周期寄存器（TMRx\_PR）值时，重新从 0 向上计数，计数器上溢并产生溢出事件，同时 OVFIF 位置 1。若禁止产生溢出事件，则计数器溢出后不再重载预分频值和周期值，否则预分频值和周期值将在溢出事件后更新。

图 14-8 PRBEN=0 时的溢出事件

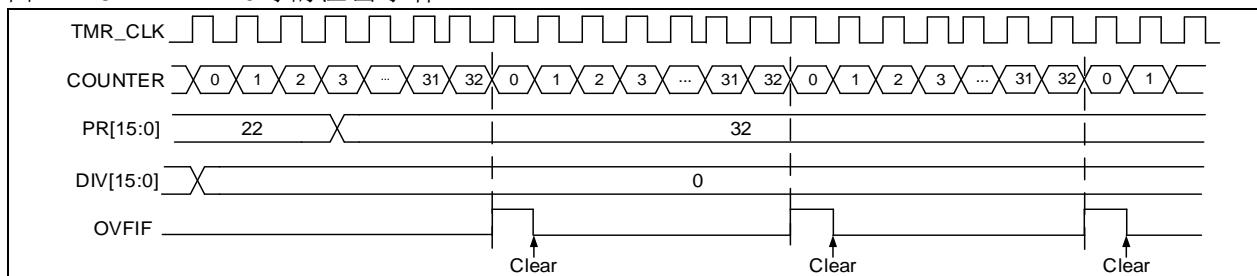
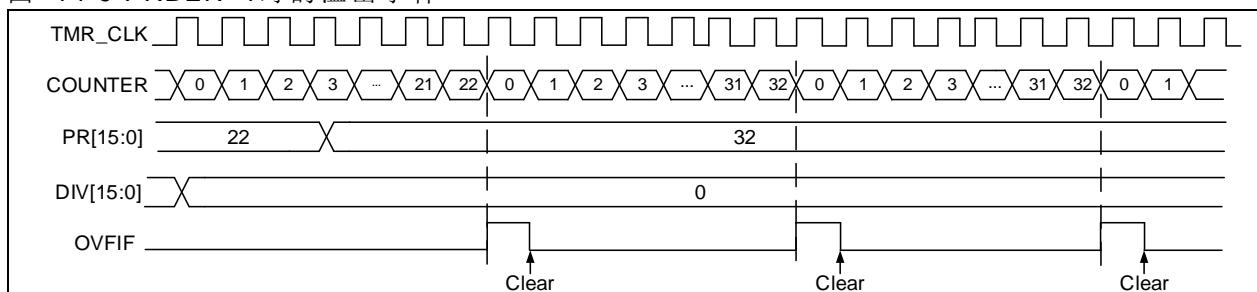


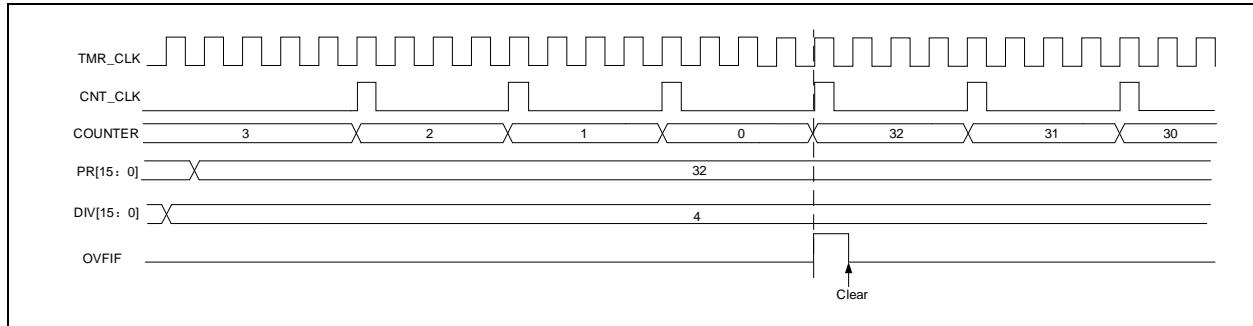
图 14-9 PRBEN=1 时的溢出事件



#### 向下计数模式

向下计数模式中，当计数值达到 0 值时，重新从周期寄存器（TMRx\_PR）向上下数，计数器下溢并产生溢出事件。

图 14-10 计数器时序图，内部时钟分频因子为4

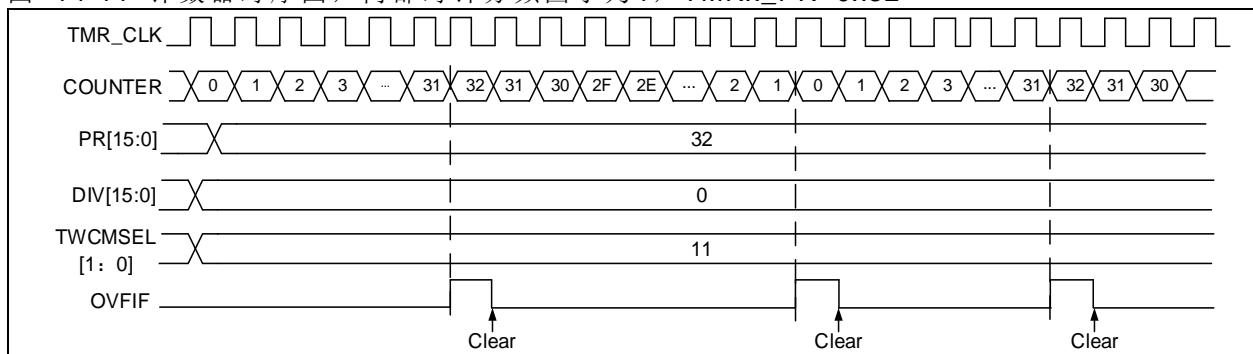


### 中央双向对齐计数模式

中央双向对齐计数模式中，计数器交替向上、向下计数。当计数值从周期寄存器（TMRx\_PR）值向下计数到1值时，产生下溢事件，然后从0开始向上计数；当向上计数到周期寄存器（TMRx\_PR）值-1时，产生上溢事件，之后从周期寄存器（TMRx\_PR）值向下计数。计数器计数方向可由计数器方向控制位（OWCDIR）实时查看。

注意：中央双向对齐计数模式下，OWCDIR位为只读位。

图 14-11 计数器时序图，内部时钟分频因子为1，TMRx\_PR=0x32



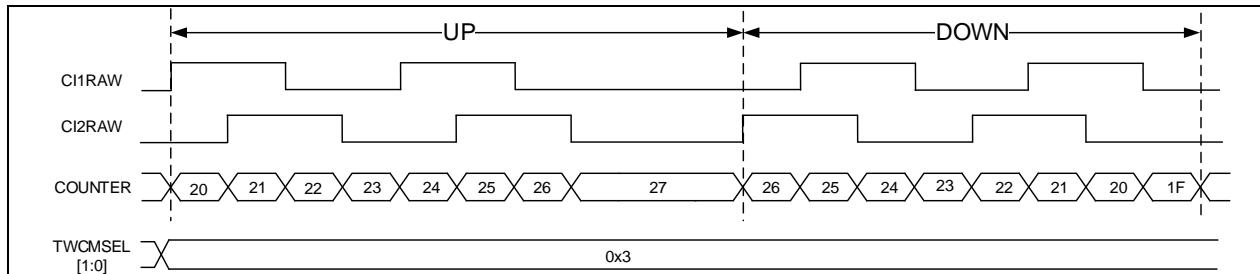
### 编码器模式

将SMSEL[2:0]配置为3'b001/3'b010/3'b011可开启编码模式，编码模式下需提供两个输入(C1IN/C2IN)，根据一个输入的电平值，计数器将在另一个输入的边沿向上或向下计数。计数方向将由OWCDIR值指示。编码模式下计数器计数方向如下表所示：

表 14-3 计数方向与编码器信号的关系

有效边沿	相对信号的电平 (C1INFP1 对应 C2IN, C2INFP2 对应 C1IN)	C1INFP1 信号		C2INFP2 信号	
		上升	下降	上升	下降
仅在 C1IN 计数	高	向下计数	向上计数	不计数	不计数
	低	向上计数	向下计数	不计数	不计数
仅在 C2IN 计数	高	不计数	不计数	向上计数	向下计数
	低	不计数	不计数	向下计数	向上计数
在 C1IN 和 C2IN 上计数	高	向下计数	向上计数	向上计数	向下计数
	低	向上计数	向下计数	向下计数	向上计数

图 14-12 编码模式计数实例（编码器模式 C）



### 14.1.3.3 TMR 输入部分

TMR2 至 5 拥有 4 个独立通道，每个通道可配置为输入或输出，当配置位输入时，可用于对输入信号的滤波、选择、分频和输入捕获功能。

图 14-13 输入/输出通道 1 的主电路

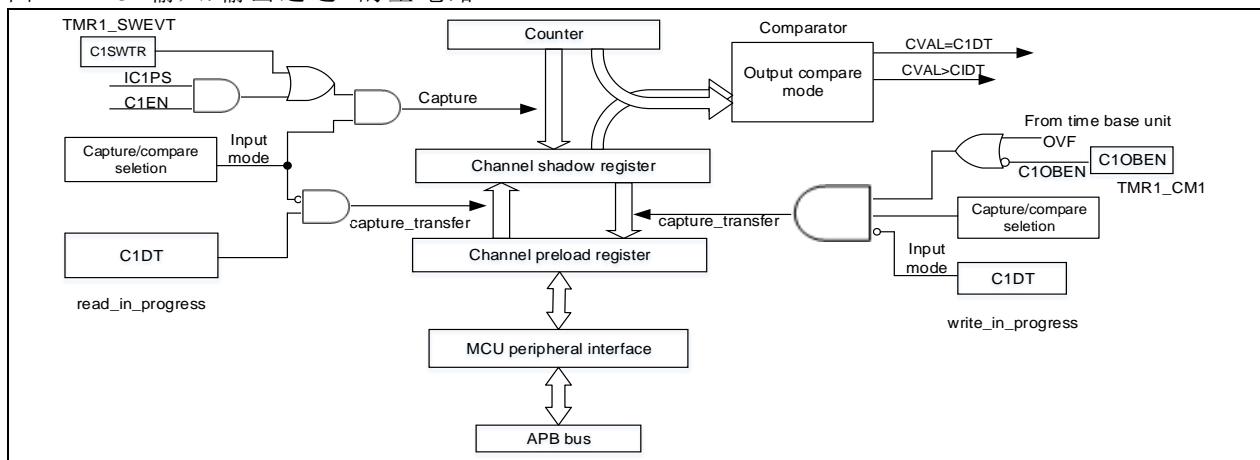
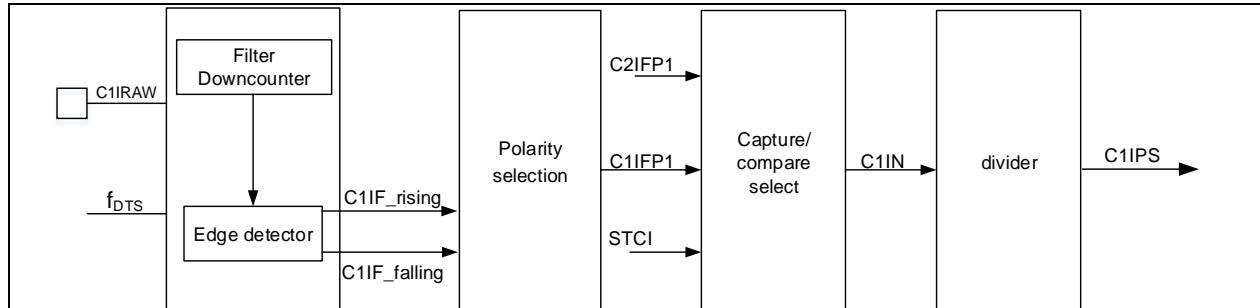


图 14-14 通道 1 输入部分



#### 输入模式

此模式下，当选中的触发信号被检测到时，通道寄存器 (TMRx\_CxDT) 会记录当前计数器计数值，并将捕获比较中断标志位 (CxIF) 置 1，若已使能通道中断 (CxIEN)、通道 DMA 请求 (CxDEN) 则产生相应的中断和 DMA 请求。若在 CxIF 已置 1 后检测到选中的触发信号，则将 CxRF 位置 1。

若要捕获 C1IN 输入的上升沿，可按如下进行配置：

- 将通道寄存器 (TMRx\_CxDT) 中的 C1C 位配置为 01，选择 C1IN 作为通道 1 输入。
- 配置 C1IN 信号滤波器带宽 (CxDF[3: 0])。
- 配置 C1IN 通道的有效沿，在通道控制寄存器 (TMRx\_CCTRL) 中写入 C1P=0 (上升沿)。
- 配置 C1IN 信号捕获频率 (C1DIV[1: 0])。
- 使能通道 1 输入捕获 (C1EN=1)。
- 根据需要设置 DMA/中断使能寄存器 (TMRx\_IDEN) 中的 C1IEN 为、DMA/中断使能寄存器 (TMRx\_IDEN) 中的 C1DEN 位，选择中断请求或 DMA 请求。

#### 多输入异或

通道 1 的输入端可选择 TMRx\_CH1、TMRx\_CH2 和 TMRx\_CH3 经异或逻辑后输入。将控制寄存器 2 (TMRx\_CTRL2) 中的 C1INSEL 位置 1 可开启此功能。

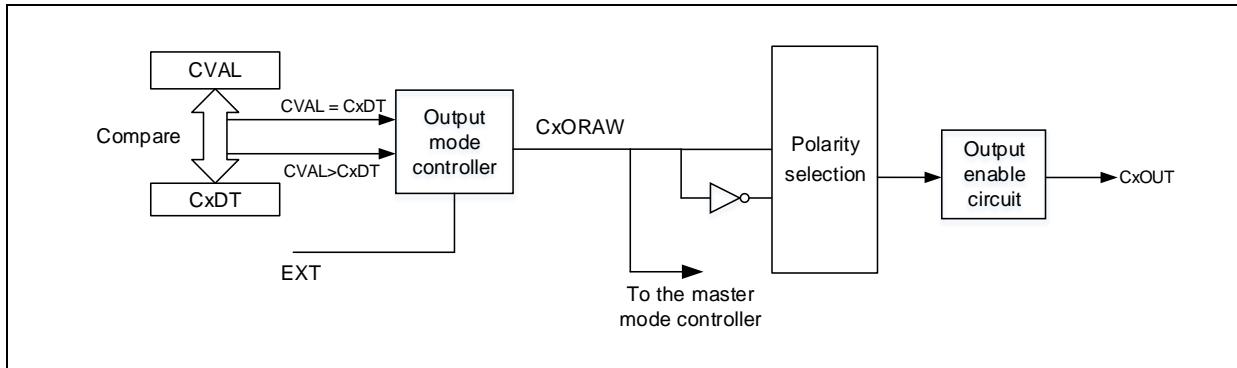
多输入异或功能可用于连接霍尔传感器，例如，将异或输入的三个输入端分别连接到三个霍尔传感器，通

过分析三路霍尔传感器信号可计算出转子的位置和速度。

#### 14.1.3.4 TMR输出部分

TMR 的输出部分由比较器和输出控制构成，用于编程输出信号的周期、占空比、极性。

图 14-15 捕获/比较通道的输出部分（通道 1 至 4）



##### 输出模式

配置  $CxC[2: 0] \neq 2'b00$  将通道配置为输出可实现多种输出模式，此时，计数器计数值将与通道寄存器（ $TMRx_CxDT$ ）值比较，并根据  $CxOCTRL[2: 0]$  位配置的输出模式，产生中间信号  $CxORAW$ ，再经过输出控制逻辑处理后输送到 IO。输出信号的周期由周期寄存器（ $TMRx_PR$ ）值配置，占空比则由通道寄存器（ $TMRx_CxDT$ ）值配置。

输出比较模式有以下子类：

- PWM 模式：**  $CxOCTRL=3'b110/111$  时，开启 PWM 模式，每个通道可独立配置并输出一路 PWM 信号。此时，输出信号的周期由  $TMRx_PR$  配置，占空比由  $CxDT$  值配置，计数器值与通道寄存器（ $TMRx_CxDT$ ）值进行比较，根据计数方向输出指定电平信号，关于 PWM 模式 A/B 详见  $CxOCTRL[2: 0]$  位描述。当计数模式为中央双向对齐计数时，可根据  $OWCDIR$  位指示计数方向。
- 强制输出模式：**  $CxOCTRL=3'b100/101$  时，开启强制输出模式。此时， $CxORAW$  信号的电平被强制输出为配置的电平，而与计数值无关。虽然输出信号不依赖于比较结果，但通道标志位和 DMA 请求仍依赖于比较结果。
- 输出比较模式：**  $CxOCTRL=3'b001/010/011$  时，开启输出比较模式。此时，当计数值与  $CxDT$  值匹配时， $CxORAW$  被强制为高电平、低电平或进行电平翻转。
- 单周期模式：** PWM 模式的特例，将  $OCMEN$  位置 1 可开启单周期模式，此模式下，仅在当前计数周期中进行比较匹配，完成当前计数后， $TMREN$  位清 0，因此仅输出一个脉冲。当配置为向上计数模式时，需要严格配置  $CVAL < CxDT \leq PR$ ；向下计数时，需严格配置  $CVAL > CxDT$ 。
- 快速输出模式：** 将  $CxOEN$  位置 1 可开启此功能，开启后  $CxORAW$  电平值不再在计数值与  $CxDT$  匹配时变化，而是在当前计数周期开始时，也就是说，比较结果被提前了，计数器值与通道寄存器（ $TMRx_CxDT$ ）的比较结果将会提前决定  $CxORAW$  的电平。

图 14-16 展示了输出比较模式（翻转）的例子， $C1DT=0x3$ ，当计数值等于  $0x3$  时，输出电平  $C1OUT$  被翻转。

图 14-17 展示了计数器向上计数与 PWM 模式 A 配合的例子， $PR=0x32$ ， $CxDT$  配置为不同的值时输出时输出信号的翻转情况。

图 14-18 展示了计数器中央双向对齐计数与 PWM 模式 A 配合的例子， $PR=0x32$ ， $CxDT$  配置为不同的值时输出时输出信号的翻转情况。

图 14-19 展示了计数器向上计数与单周期模式下 PWM 模式 B 配合的例子，计数器仅计数了一个周期，输出信号在这个周期中只输出了一个脉冲。

图 14-16 计数值与C1DT值匹配时翻转C1ORAW

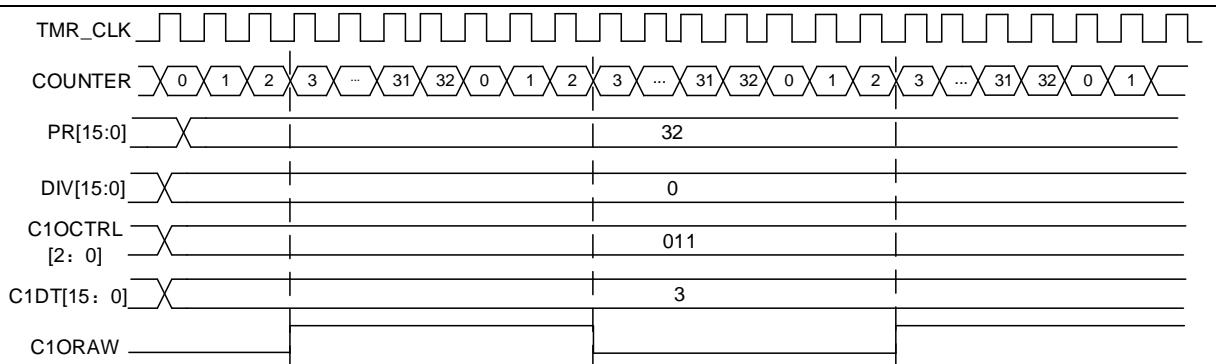


图 14-17 向上计数下 PWM 模式 A

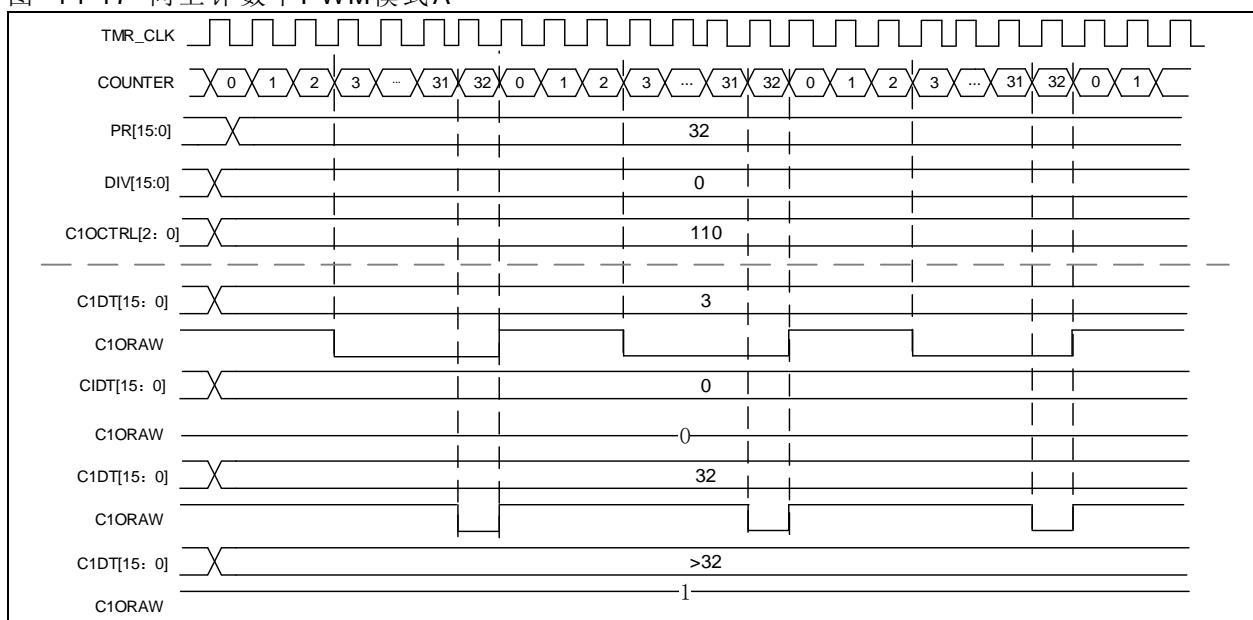


图 14-18 中央双向对齐计数下 PWM 模式 A

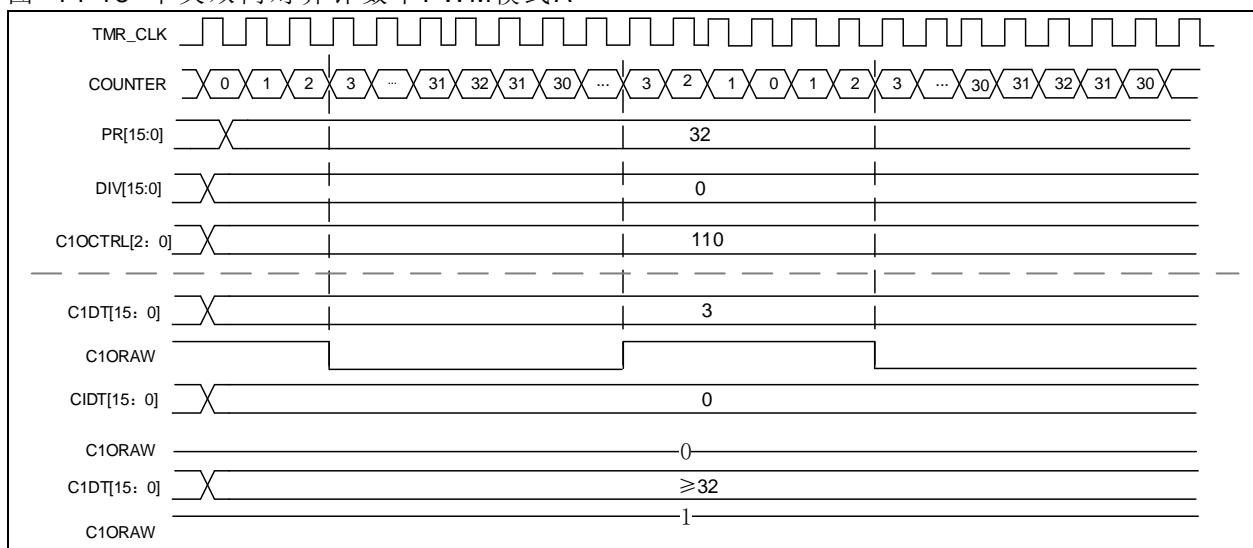
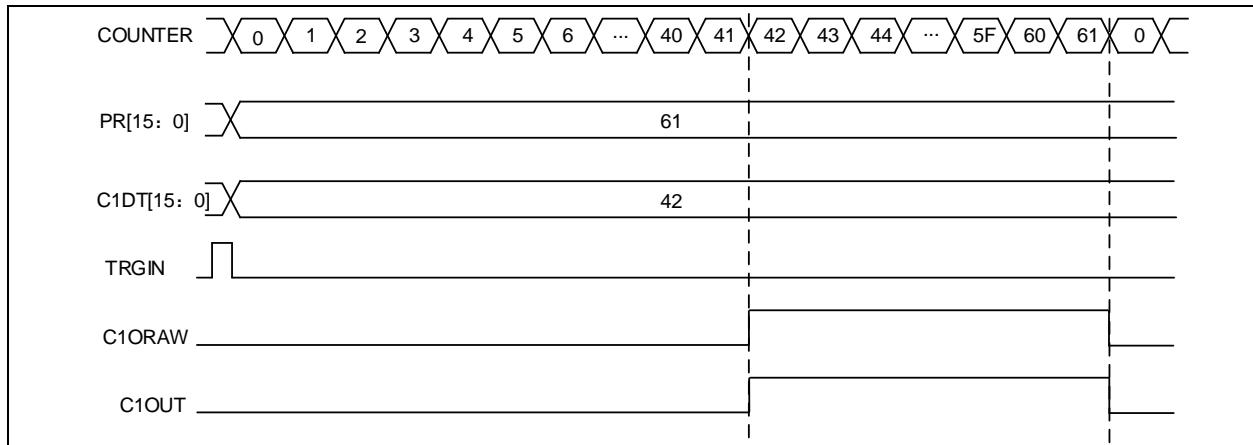


图 14-19 单周期模式

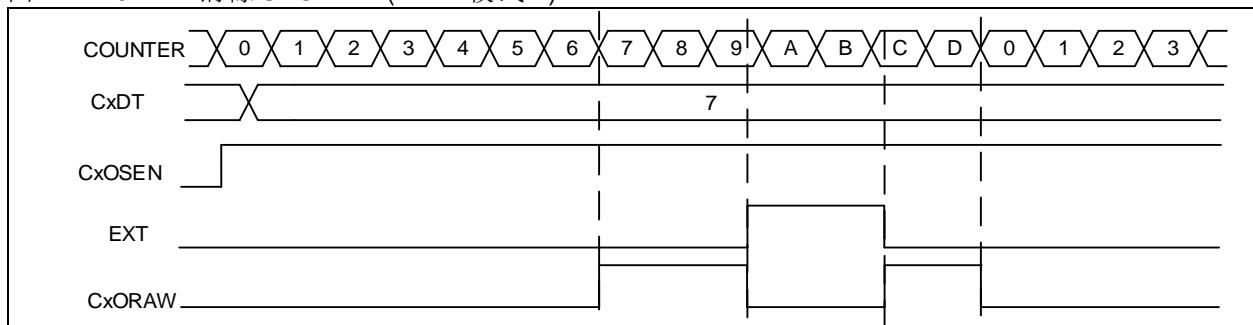


### CxORAW 信号清除

将 CxOSEN 位置 1 后, 指定通道的 CxORAW 信号由 EXT 高电平清 0, 在下一次溢出事件发生前 CxORAW 信号无法被改变。

强制模式时, CxORAW 信号清除功能不可用, 只有在输出比较模式或 PWM 模式, 此功能有效。下图显示了使用 EXT 信号清除 CxORAW 的例子, 当 EXT 为高电平期间, 原本为高电平的 CxORAW 信号被拉低, 当 EXT 为低电平时, CxORAW 根据计数值和 CxDT 比较结果输出电平。

图 14-20 EXT 清除 CxORAW(PWM 模式 A)



### 14.1.3.5 定时器同步

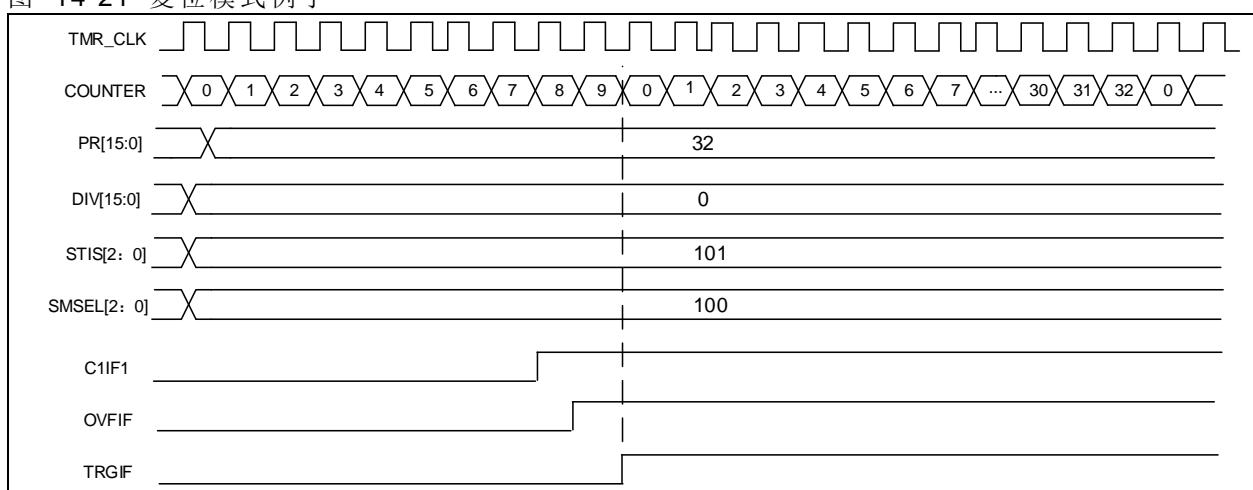
主次定时器之间可由内部连接信号进行同步。主定时器可由 PTOS[2: 0]位选择主定时器输出, 即同步信息; 次定时器由 SMSEL[2: 0]位选择从模式, 即次定时器的工作模式。

定时器从模式有以下几种:

#### 从模式: 复位模式

选中的触发信号将复位计数器和预分频器, 若 OVFS 位为 0, 将产生一个溢出事件。

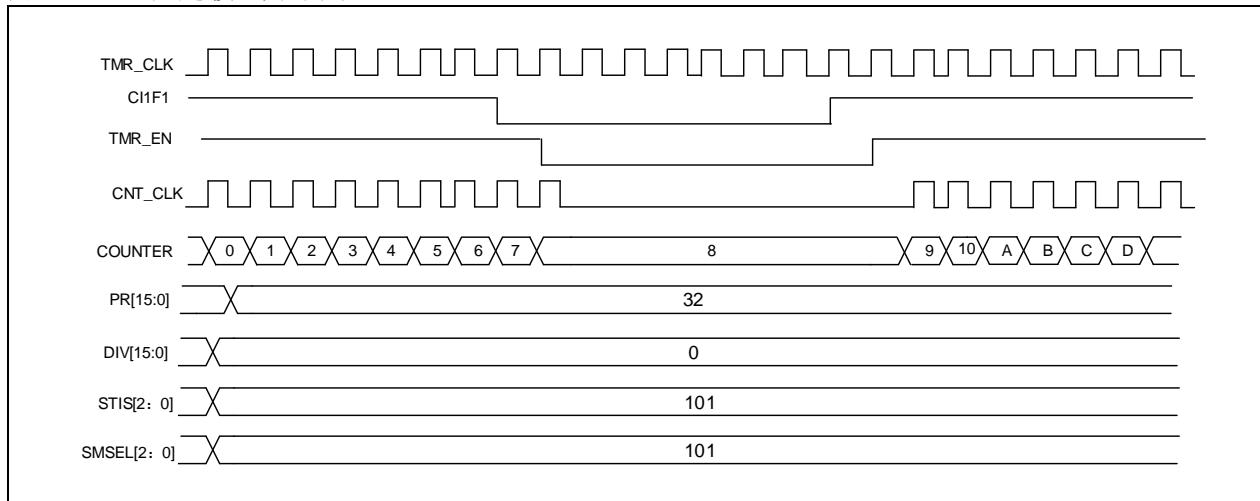
图 14-21 复位模式例子



### 从模式：挂起模式

挂起模式下，计数的计数和停止受选中触发输入信号控制，当触发输入为高电平时计数器开始计数；当为低电平时，计数器暂停计数。

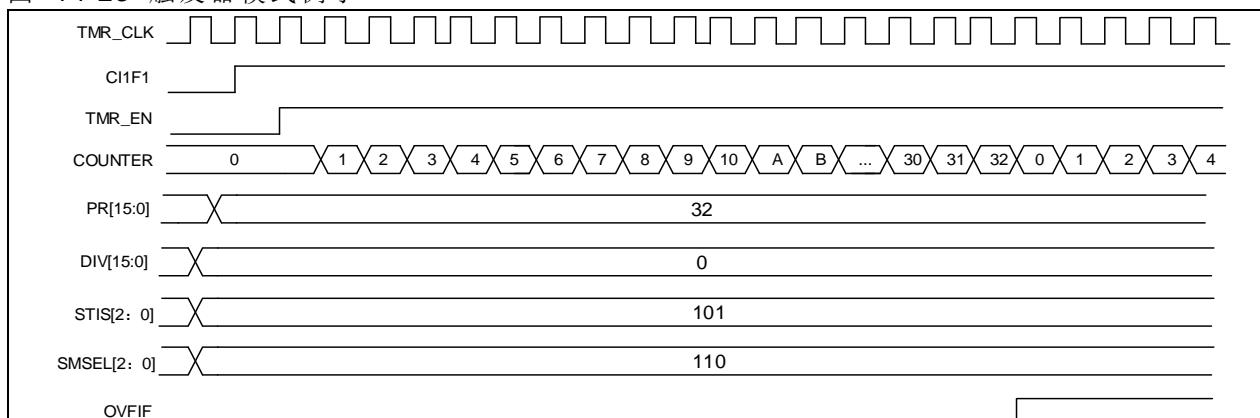
图 14-22 挂起模式下例子



### 从模式：触发模式

计数器将在选中的触发输入上升沿启动计数（将 TMR\_EN 置 1）。

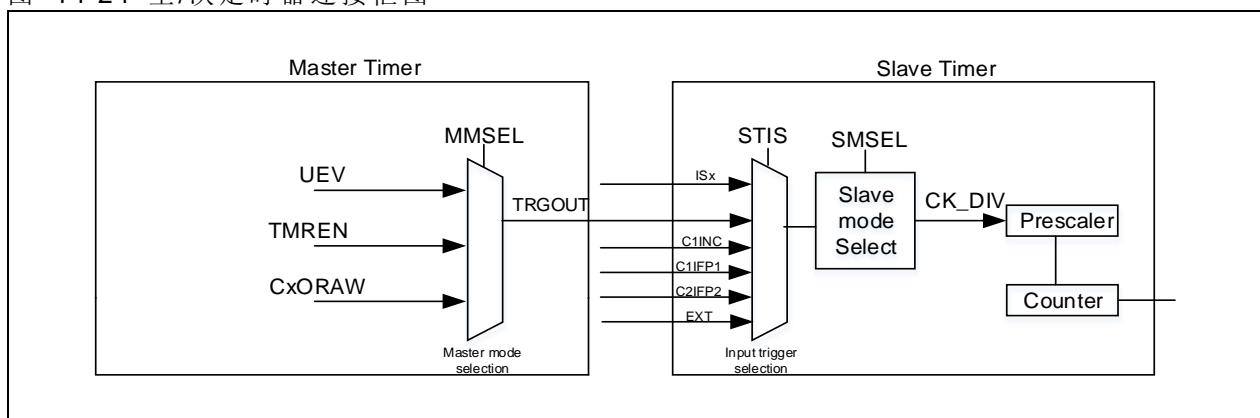
图 14-23 触发器模式例子



### 主/次定时器互联实例

主/次定时器可分别配置不同的主模式和从模式，两者搭配可实现多种功能，一下提供了一些定时器互联的例子。

图 14-24 主/次定时器连接框图



主定时器为次定时器提供时钟：

- 配置主定时器输出信号 TRGOUT 为溢出事件，配置 PTOS[2: 0]=3'b010，主定时器每次计数器溢出输出一个脉冲信号，用作次定时器计数时钟。
- 配置主定时器计数周期（周期寄存器（TMRx\_PR））。
- 配置次定时器触发输入信号 TRGIN 为主定时器输出（次定时器控制寄存器

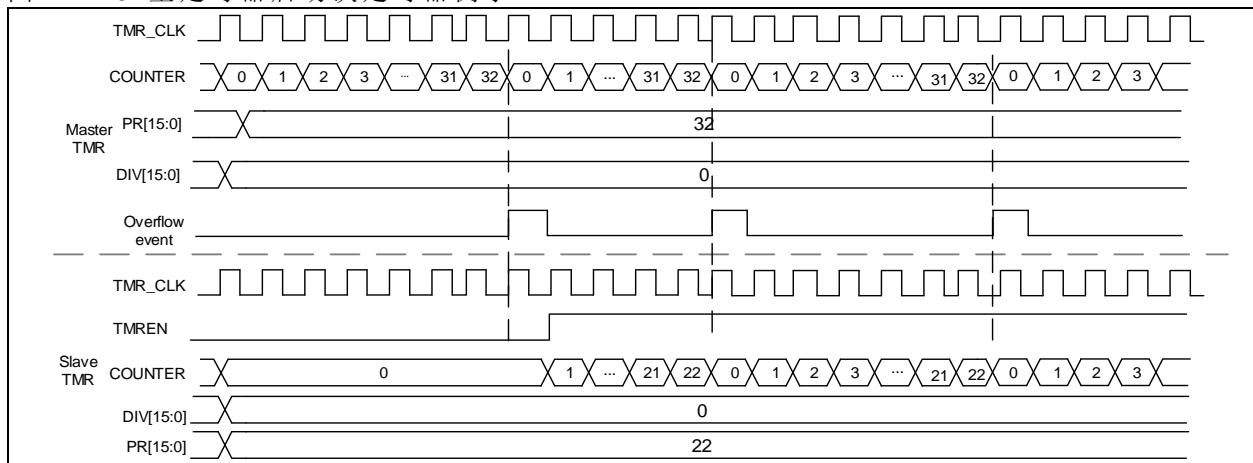
(TMRx\_STCTRL) 的 STIS[2: 0] )。

- 配置次定时器使用外部时钟模式 A (次定时器控制寄存器 (TMRx\_STCTRL) 的 SMSEL[2: 0]=3'b111)。
- 将主定时器和次定时器的 TMREN 位置 1 启动定时器。

主定时器启动次定时器：

- 配置主定时器输出信号 TRGOUT 为溢出事件，配置 PTOS[2: 0]=3'b010，主定时器每次计数器溢出输出一个脉冲信号，用作次定时器计数时钟。
- 配置主定时器计数周期（周期寄存器 (TMRx\_PR)）。
- 配置次定时器触发输入 TRGIN 为主定时器输出。
- 配置次定时器为触发模式 (TMR2\_STCTRL 寄存器的 SMSEL=3'b110)
- 置主定时器 TMREN=1 以启动主定时器。

图 14-25 主定时器启动次定时器例子

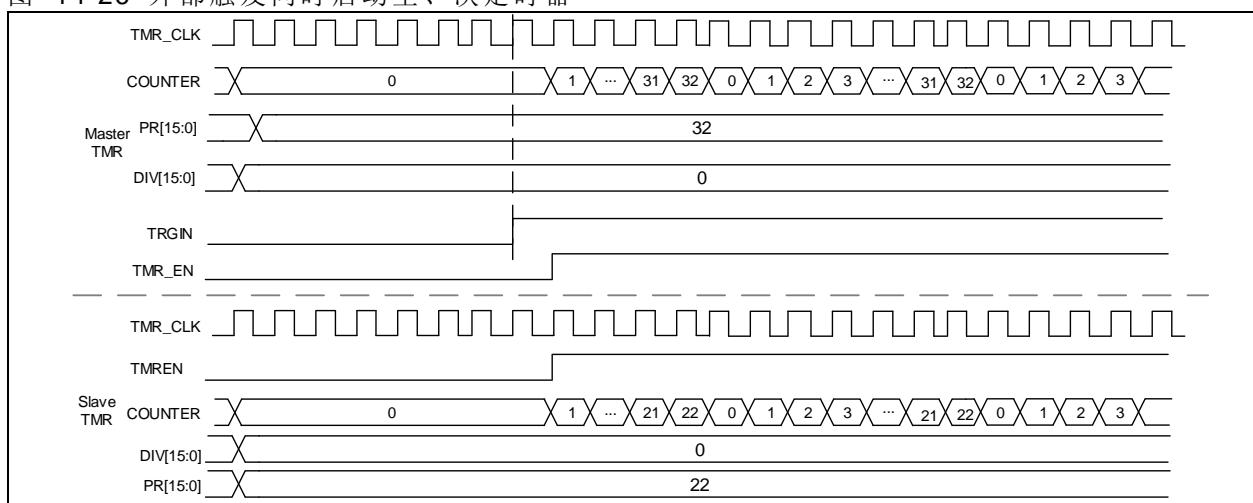


外部触发信号同步启动主、次定时器：

这个例子中，主定时器同时作为主定时器和次定时器，将主定时器的次定时器同步功能开启，此模式用于将主定时器和次定时器保持同步。

- 配置主定时器 STS 位为 1。
- 配置主定时器输出信号 TRGOUT 为溢出事件，配置 PTOS[2: 0]=3'b010，主定时器每次计数器溢出输出一个脉冲信号，用作次定时器计数时钟。
- 配置主定时器的次定时模式为触发模式，触发源选择 C1IN。
- 配置次定时器触发输入 TRGIN 为主定时器输出。
- 配置次定时器为触发模式 (TMR2\_STCTRL 寄存器的 SMSEL=3'b110)。

图 14-26 外部触发同时启动主、次定时器



### 14.1.3.6 调试模式

当微控制器进入调试模式 (Cortex™-M4 核心停止) 时，将 DEBUG 模块中的 TMRx\_PAUSE 置 1，可以使 TMRx 计数器暂停计数。

### 14.1.4 TMRx寄存器描述

必须以字（32位）的方式操作这些外设寄存器。

下表中将 TMRx 的所有寄存器映射到一个 16 位可寻址（编址）空间。

表 14-4 TMRx – 寄存器映像和复位值

寄存器简称	基址偏移量	复位值
TMRx_CTRL1	0x00	0x0000
TMRx_CTRL2	0x04	0x0000
TMRx_STCTRL	0x08	0x0000
TMRx_IDEN	0x0C	0x0000
TMRxISTS	0x10	0x0000
TMRx_SWEVT	0x14	0x0000
TMRx_CM1	0x18	0x0000
TMRx_CM2	0x1C	0x0000
TMRx_CCTRL	0x20	0x0000
TMRx_CVAL	0x24	0x0000 0000
TMRx_DIV	0x28	0x0000
TMRx_PR	0x2C	0x0000 0000
TMRx_C1DT	0x34	0x0000 0000
TMRx_C2DT	0x38	0x0000 0000
TMRx_C3DT	0x3C	0x0000 0000
TMRx_C4DT	0x40	0x0000 0000
TMRx_DMACTRL	0x48	0x0000
TMRx_DMADT	0x4C	0x0000

#### 14.1.4.1 控制寄存器1 (TMRx\_CTRL1)

域	简称	复位值	类型	功能
位 15: 11	保留	0x00	resd	保持默认值。
位 10	PMEN	0x0	rw	增强模式使能 (Plus Mode Enable) 开启 TMRx 增强模式，该模式下 TMRx_CVAL, TMRx_PR, TMRx_CxDT 由 16 位扩展为 32 位。 0: 关闭; 1: 启开。 注: TMR2 和 TMR5 才具有此功能, 其它 TMR 设置此位无效。
位 9: 8	CLKDIV	0x0	rw	时钟除频 (Clock divider) 00: 无除频; 01: 2 除频; 10: 4 除频; 11: 保留。
位 7	PRBEN	0x0	rw	周期缓冲使能 (Period buffer enable) 0: 缓冲关闭; 1: 缓冲开启。
位 6: 5	TWCMSEL	0x0	rw	中央双向对齐计数模式选择 (Two-way count mode selection) 00: 单向对齐计数模式, 方向由 OWCDIR 配置; 01: 中央双向对齐计数模式 1, 上下交替计数, 输出标志位只在计数器向下计数时被置起; 10: 中央双向对齐计数模式 2, 上下交替计数, 输出标志位只在计数器向上计数时被置起;

				11: 中央双向对齐计数模式 3, 上下交替计数, 输出标志位在计数器向上和向下计数时皆被置起。
位 4	OWCDIR	0x0	rw	单向对齐计数模式计数方向 (One-way count direction) 0: 向上; 1: 向下。
位 3	OCMEN	0x0	rw	单周期使能 (One cycle mode enable) 该功能用于选择更新事件后, 计数器是否停止。 0: 关闭; 1: 开启。
位 2	OVFS	0x0	rw	溢出事件源选择 (Overflow event source) 配置溢出事件或 DMA 请求来源。 0: 来源于计数器溢出、设置 OVFSWTR 位或次定时器控制器产生的溢出事件; 1: 只能来源于计数器溢出。
位 1	OVFEN	0x0	rw	溢出事件使能 (Overflow event enable) 0: 开启; 1: 关闭。
位 0	TMREN	0x0	rw	使能定时器 (TMR enable) 0: 关闭; 1: 开启。

#### 14.1.4.2 控制寄存器2 (TMRx\_CTRL2)

域	简称	复位值	类型	功能
位 15: 8	保留	0x00	resd	保持默认值。
位 7	C1INSEL	0x0	rw	C1IN 选择 (C1IN selection) 0: CH1 管脚连到 C1IRAW 输入; 1: CH1、CH2 和 CH3 管脚异或结果连到 C1IRAW 输入。
位 6: 4	PTOS	0x0	rw	主定时器输出信号选择 (Primary TMR output selection) TMRx 输出到次定时器的信号选择: 000: 复位; 001: 使能; 010: 更新; 011: 比较脉冲; 100: C1ORAW 信号; 101: C2ORAW 信号; 110: C3ORAW 信号; 111: C4ORAW 信号。
位 3	DRS	0x0	rw	DMA 请求源 (DMA request source) DMA 请求来源。 0: 捕获/比较事件; 1: 溢出事件。
位 2: 0	保留	0x0	resd	保持默认值。

#### 14.1.4.3 次定时器控制寄存器 (TMRx\_STCTRL)

域	简称	复位值	类型	功能
位 15	ESP	0x0	rw	外部信号极性 (External signal polarity) 用于选择外部方式。 0: 高电平或上升沿; 1: 低电平或下降沿。
位 14	ECMBEN	0x0	rw	外部时钟模式 B 使能 (External clock mode B enable) 用于启用外部时钟模式 B 0: 关闭; 1: 开启。
位 13: 12	ESDIV	0x0	rw	外部信号除频 (External signal divide) 用于选择降低外部触发频率的除频。 00: 关闭分频; 01: 2 分频;

				10: 4 分频; 11: 8 分频。
位 11: 8	ESF	0x0	rw	外部信号滤波 (External signal filter) 用于过滤外部信号，当外部信号产生了 N 次之后才能被采样。 0000: 无滤波器，以 $f_{DTS}$ 采样 0001: $f_{SAMPLING} = f_{CK\_INT}$ , N=2; 0010: $f_{SAMPLING} = f_{CK\_INT}$ , N=4; 0011: $f_{SAMPLING} = f_{CK\_INT}$ , N=8; 0100: $f_{SAMPLING} = f_{DTS}/2$ , N=6; 0101: $f_{SAMPLING} = f_{DTS}/2$ , N=8; 0110: $f_{SAMPLING} = f_{DTS}/4$ , N=6; 0111: $f_{SAMPLING} = f_{DTS}/4$ , N=8; 1000: $f_{SAMPLING} = f_{DTS}/8$ , N=6; 1001: $f_{SAMPLING} = f_{DTS}/8$ , N=8; 1010: $f_{SAMPLING} = f_{DTS}/16$ , N=5; 1011: $f_{SAMPLING} = f_{DTS}/16$ , N=6; 1100: $f_{SAMPLING} = f_{DTS}/16$ , N=8; 1101: $f_{SAMPLING} = f_{DTS}/32$ , N=5; 1110: $f_{SAMPLING} = f_{DTS}/32$ , N=6; 1111: $f_{SAMPLING} = f_{DTS}/32$ , N=8。
位 7	STS	0x0	rw	次定时器同步 (Subordinate TMR synchronization) 该位开启后，主次定时器可实现高度同步。 0: 关闭; 1: 开启。
位 6: 4	STIS	0x0	rw	次定时器输入选择 (Subordinate TMR input selection) 用于次定时器的输入选择。 000: 内部选择 0 (IS0) ; 001: 内部选择 1 (IS1) ; 010: 内部选择 2 (IS2) ; 011: 内部选择 3 (IS3) ; 100: C1IRAW 的输入检测器 (C1INC) ; 101: 滤波输入 1 (C1IF1) ; 110: 滤波输入 2 (C2IF2) ; 111: 外部输入 (EXT) 。 关于每个定时器中 ISx 的细节，参见表 14-2。
位 3	保留	0x0	resd	保持默认值。
位 2: 0	SMSEL	0x0	rw	次定时器模式选择 (Subordinate TMR mode selection) 000: 关闭从模式; 001: 编码模式 A; 010: 编码模式 B; 011: 编码模式 C; 100: 复位模式 - TRGIN 输入上升沿时，重新初始化计数器; 101: 挂起模式 - TRGIN 输入高电平时，计数器计数; 110: 触发模式 - TRGIN 输入上升沿时，产生触发事件; 111: 外部时钟模式 A - TRGIN 输入上升沿时，提供时钟; 注：编码模式 A/B/C 配置方法请查看计数模式章节。

#### 14.1.4.4 DMA/中断使能寄存器 (TMRx\_IDEN)

域	简称	复位值	类型	功能
位 15	保留	0x0	resd	保持默认值。
位 14	TDEN	0x0	rw	触发 DMA 请求使能 (Trigger DMA request enable) 0: 关闭; 1: 开启。
位 13	保留	0x0	resd	保持默认值。

位 12	C4DEN	0x0	rw	通道 4 的 DMA 请求使能(Channel 4 DMA request enable) 0: 关闭; 1: 开启。
位 11	C3DEN	0x0	rw	通道 3 的 DMA 请求使能(Channel 3 DMA request enable) 0: 关闭; 1: 开启。
位 10	C2DEN	0x0	rw	通道 2 的 DMA 请求使能(Channel 2 DMA request enable) 0: 关闭; 1: 开启。
位 9	C1DEN	0x0	rw	通道 1 的 DMA 请求使能(Channel 1 DMA request enable) 0: 关闭; 1: 开启。
位 8	OVFDEN	0x0	rw	溢出事件的 DMA 请求使能 (overflow event DMA request enable) 0: 关闭; 1: 开启。
位 7	保留	0x0	resd	保持默认值。
位 6	TIEN	0x0	rw	触发中断使能 (Trigger interrupt enable) 0: 关闭; 1: 开启。
位 5	保留	0x0	resd	保持默认值。
位 4	C4IEN	0x0	rw	通道 4 中断使能 (Channel 4 interrupt enable) 0: 关闭; 1: 开启。
位 3	C3IEN	0x0	rw	通道 3 中断使能 (Channel 3 interrupt enable) 0: 关闭; 1: 开启。
位 2	C2IEN	0x0	rw	通道 2 中断使能 (Channel 2 interrupt enable) 0: 关闭; 1: 开启。
位 1	C1IEN	0x0	rw	通道 1 中断使能 (Channel 1 interrupt enable) 0: 关闭; 1: 开启。
位 0	OVFIEN	0x0	rw	溢出中断使能 (overflow interrupt enable) 0: 关闭; 1: 开启。

#### 14.1.4.5 中断状态寄存器 (TMRxISTS)

域	简称	复位值	类型	功能
位 15: 13	保留	0x0	resd	保持默认值。
位 12	C4RF	0x0	rw0c	通道 4 再捕获标记 (Channel 4 recapture flag) 见 C1RF 的描述。
位 11	C3RF	0x0	rw0c	通道 3 再捕获标记 (Channel 3 recapture flag) 见 C1RF 的描述。
位 10	C2RF	0x0	rw0c	通道 2 再捕获标记 (Channel 2 recapture flag) 见 C1RF 的描述。
位 9	C1RF	0x0	rw0c	通道 1 再捕获标记 (Channel 1 recapture flag) C1IF 的状态已经为'1'时是否再次发生了捕获, 由硬件置'1', 写'0'清除。 0: 无捕获发生; 1: 捕获发生。
位 8: 7	保留	0x0	resd	保持默认值。
位 6	TRGIF	0x0	rw0c	触发中断标记 (Trigger interrupt flag) 当发生触发事件时由硬件置'1', 写'0'清除。 0: 无触发事件发生; 1: 发生触发事件。 触发事件: 在 TRGIN 接收到有效边沿, 或挂起模式下接收到任意边沿。
位 5	保留	0x0	resd	保持默认值。
位 4	C4IF	0x0	rw0c	通道 4 中断标记 (Channel 4 interrupt flag)

				参考 C1IF 描述。
位 3	C3IF	0x0	rw0c	通道 3 中断标记 (Channel 3 interrupt flag) 参考 C1IF 描述。
位 2	C2IF	0x0	rw0c	通道 2 中断标记 (Channel 2 interrupt flag) 参考 C1IF 描述。
位 1	C1IF	0x0	rw0c	通道 1 中断标记 (Channel 1 interrupt flag) 若通道 1 为输入模式时： 捕获事件发生时由硬件置'1'，由软件清'0'或读 TMRx_C1DT 清'0'。 0: 无捕获事件发生； 1: 发生捕获事件。 若通道 1 为输出模式时： 比较事件发生时由硬件置'1'，由软件清'0'。 0: 无比较事件发生； 1: 发生比较事件。
位 0	OVFIF	0x0	rw0c	溢出中断标记 (Overflow interrupt flag) 当溢出事件发生时由硬件置'1'，由软件清'0'。 0: 无溢出事件发生； 1: 发生溢出事件，若 TMRx_CTRL1 的 OVFEN=0、 OVFS=0 时： - 当 TMRx_SWEVE 寄存器的 OVFG=1 时产生溢出事件； - 当计数值 CVAL 被触发事件重初始化时产生溢出事件。

#### 14.1.4.6 软件事件寄存器 (TMRx\_SWEVT)

域	简称	复位值	类型	功能
位 15: 7	保留	0x000	resd	保持默认值。
位 6	TRGSWTR	0x0	rw	软件触发触发事件 (Trigger event triggered by software) 通过软件触发一个触发事件。 0: 无作用； 1: 制造一个触发事件。
位 5	保留	0x0	resd	保持默认值。
位 4	C4SWTR	0x0	wo	软件触发通道 4 事件 (Channel 4 event triggered by software) 见 C1M 的描述。
位 3	C3SWTR	0x0	wo	软件触发通道 3 事件 (Channel 3 event triggered by software) 见 C1M 的描述。
位 2	C2SWTR	0x0	wo	软件触发通道 2 事件 (Channel 2 event triggered by software) 见 C1M 的描述。
位 1	C1SWTR	0x0	wo	软件触发通道 1 事件 (Channel 1 event triggered by software) 通过软件触发一个通道 1 事件。 0: 无作用； 1: 制造一个通道 1 事件。
位 0	OVFSWTR	0x0	wo	软件触发溢出事件 (Overflow event triggered by software) 通过软件触发一个溢出事件。 0: 无作用； 1: 制造一个溢出事件。

#### 14.1.4.7 通道模式寄存器1 (TMRx\_CM1)

输出比较模式：

域	简称	复位值	类型	功能
位 15	C2OSEN	0x0	rw	通道 2 输出开关使能 (Channel 2 output switch enable)
位 14: 12	C2OCTRL	0x0	rw	通道 2 输出控制 (Channel 2 output control)
位 11	C2OBEN	0x0	rw	通道 2 输出缓存使能 (Channel 2 output buffer enable)

位 10	C2OIEN	0x0	rw	通道 2 输出立即使能 (Channel 2 output immediately enable)
位 9: 8	C2C	0x0	rw	通道 2 配置 (Channel 2 configure) 当 C2EN=0'时, 这些位用于选择通道 2 为输出或输入, 以及输入时的映射选择: 00: 输出; 01: 输入, C2IN 映射在 C2IRAW 上; 10: 输入, C2IN 映射在 C1IRAW 上; 11: 输入, C2IN 映射在 STCI 上, 只有在 STIS 选择内部触发输入时才工作。
位 7	C1OSEN	0x0	rw	通道 1 输出开关使能 (Channel 1 output switch enable) 0: EXT 输入不影响 C1ORAW; 1: 当 EXT 输入高电平时, 将 C1ORAW 清 0。
位 6: 4	C1OCTRL	0x0	rw	通道 1 输出控制 (Channel 1 output control) 这些位用于设置原始信号 C1ORAW 的工作状态。 000: 断开。断开 C1ORAW 到 C1OUT 的输出; 001: 设置 C1ORAW 为高: TMRx_CVAL=TMRx_C1DT 时。 010: 设置 C1ORAW 为低: TMRx_CVAL=TMRx_C1DT 时。 011: 切换 C1ORAW 的电平: 当 TMRx_CVAL=TMRx_C1DT 时。 100: 固定 C1ORAW 为低。 101: 固定 C1ORAW 为高。 110: PWM 模式 A -OWCDIR=0, 若 TMRx_C1DT>TMRx_CVAL 时设置 C1ORAW 为高, 否则为低; -OWCDIR=1, 若 TMRx_C1DT < TMRx_CVAL 时设置 C1ORAW 为低, 否则为高。 111: PWM 模式 B -OWCDIR=0, 若 TMRx_C1DT > TMRx_CVAL 时设置 C1ORAW 为低, 否则为高; -OWCDIR=1, 若 TMRx_C1DT < TMRx_CVAL 时设置 C1ORAW 为高, 否则为低。 注: 除'000'外, 其余配置下 C1OUT 将连接到 C1ORAW, C1OUT 的输出电平除了会根据 C1ORAW 变化外, 还与 CCTRL 所配置的输出极性有关。
位 3	C1OBEN	0x0	rw	通道 1 输出缓存使能 (Channel 1 output buffer enable) 0: 关闭 TMRx_C1DT 的缓存功能, 写入 TMRx_C1DT 的内容会立即生效。 1: 启用 TMRx_C1DT 的缓存功能, 写入 TMRx_C1DT 的内容将保存到缓存寄存器中, 当发生溢出事件时再更新到 TMRx_C1DT 中。
位 2	C1OIEN	0x0	rw	通道 1 输出立即使能 (Channel 1 output immediately enable) 在 PWM 模式 A 或模式 B 下, 该位能够缩短触发事件到通道 1 的输出响应间的时间。 0: 需要比较 CVAL 与 C1DT 的值之后再产生输出。 1: 无需比较 CVAL 与 C1DT 的值, 当发生触发事件时立即产生输出。
位 1: 0	C1C	0x0	rw	通道 1 配置 (Channel 1 configure) 当 C1EN=0'时, 这些位用于选择通道 1 为输出或输入, 以及输入时的映射选择: 00: 输出; 01: 输入, C1IN 映射在 C1IRAW 上; 10: 输入, C1IN 映射在 C2IRAW 上; 11: 输入, C1IN 映射在 STCI 上, 只有在 STIS 选择内部触发输入时才工作。

输入模式:

域	简称	复位值	类型	功能
位 15: 12	C2DF	0x0	rw	通道 2 滤波器 (Channel 2 digital filter)
位 11: 10	C2IDIV	0x0	rw	通道 2 分频系数 (Channel 2 input divider)
位 9: 8	C2C	0x0	rw	通道 2 配置 (Channel 2 configure) 当 C2EN='0'时, 这些位用于选择通道 2 为输出或输入, 以及输入时的映射选择: 00: 输出; 01: 输入, C2IN 映射在 C2IRAW 上; 10: 输入, C2IN 映射在 C1IRAW 上; 11: 输入, C2IN 映射在 STCI 上, 只有在 STIS 选择内部触发输入时才工作。
位 7: 4	C1DF	0x0	rw	通道 1 滤波器 (Channel 1 digital filter) 这些位用于配置通道 1 的滤波器。滤波的个数为 N, 则表示发生了 N 次采样事件后输入边沿才能通过滤波器: 0000: 无滤波器, 以 $f_{DTS}$ 采样 1000: 采样频率 $f_{SAMPLING} = f_{DTS}/8$ , N=6 0001: 采样频率 $f_{SAMPLING} = f_{CK\_INT}$ , N=2 1001: 采样频率 $f_{SAMPLING} = f_{DTS}/8$ , N=8 0010: 采样频率 $f_{SAMPLING} = f_{CK\_INT}$ , N=4 1010: 采样频率 $f_{SAMPLING} = f_{DTS}/16$ , N=5 0011: 采样频率 $f_{SAMPLING} = f_{CK\_INT}$ , N=8 1011: 采样频率 $f_{SAMPLING} = f_{DTS}/16$ , N=6 0100: 采样频率 $f_{SAMPLING} = f_{DTS}/2$ , N=6 1100: 采样频率 $f_{SAMPLING} = f_{DTS}/16$ , N=8 0101: 采样频率 $f_{SAMPLING} = f_{DTS}/2$ , N=8 1101: 采样频率 $f_{SAMPLING} = f_{DTS}/32$ , N=5 0110: 采样频率 $f_{SAMPLING} = f_{DTS}/4$ , N=6 1110: 采样频率 $f_{SAMPLING} = f_{DTS}/32$ , N=6 0111: 采样频率 $f_{SAMPLING} = f_{DTS}/4$ , N=8 1111: 采样频率 $f_{SAMPLING} = f_{DTS}/32$ , N=8
位 3: 2	C1IDIV	0x0	rw	通道 1 分频系数 (Channel 1 input divider) 这些位定义了通道 1 的分频系数。 00: 不分频, 每一个有效的边沿都会产生一次输入; 01: 每 2 个有效的边沿产生一次输入; 10: 每 4 个有效的边沿产生一次输入; 11: 每 8 个有效的边沿产生一次输入。 注: C1EN='0'时, 分频系数复位。
位 1: 0	C1C	0x0	rw	通道 1 配置 (Channel 1 configure) 当 C1EN='0'时, 这些位用于选择通道 1 为输出或输入, 以及输入时的映射选择: 00: 输出; 01: 输入, C1IN 映射在 C1IRAW 上; 10: 输入, C1IN 映射在 C2IRAW 上; 11: 输入, C1IN 映射在 STCI 上, 只有在 STIS 选择内部触发输入时才工作。

#### 14.1.4.8 通道模式寄存器2 (TMRx\_CM2)

输出比较模式:

域	简称	复位值	类型	功能
位 15	C4OSEN	0x0	rw	通道 4 输出开关使能 (Channel 4 output switch enable)
位 14: 12	C4OCTRL	0x0	rw	通道 4 输出控制 (Channel 4 output control)
位 11	C4OBEN	0x0	rw	通道 4 输出缓存使能 (Channel 4 output buffer enable)
位 10	C4OIEN	0x0	rw	通道 4 输出立即使能 (Channel 4 output immediately enable)
位 9: 8	C4C	0x0	rw	通道 4 配置 (Channel 4 configure) 当 C4EN='0'时, 这些位用于选择通道 4 为输出或输入, 以及输入时的映射选择: 00: 输出; 01: 输入, C4IN 映射在 C4IRAW 上; 10: 输入, C4IN 映射在 C3IRAW 上;

位 7	C3OSEN	0x0	rw	11: 输入, C4IN 映射在 STCI 上, 只有在 STIS 选择内部触发输入时才工作。 通道 3 输出开关使能 (Channel 3 output switch enable)
位 6: 4	C3OCTRL	0x0	rw	通道 3 输出控制 (Channel 3 output control)
位 3	C3OBEN	0x0	rw	通道 3 输出缓存使能 (Channel 3 output buffer enable)
位 2	C3OIEN	0x0	rw	通道 3 输出立即使能 (Channel 3 output immediately enable)
位 1: 0	C3C	0x0	rw	通道 3 配置 (Channel 3 configure) 当 C3EN=0'时, 这些位用于选择通道 3 为输出或输入, 以及输入时的映射选择: 00: 输出; 01: 输入, C3IN 映射在 C3IRAW 上; 10: 输入, C3IN 映射在 C4IRAW 上; 11: 输入, C3IN 映射在 STCI 上, 只有在 STIS 选择内部触发输入时才工作。

**输入模式:**

域	简称	复位值	类型	功能
位 15: 12	C4DF	0x0	rw	通道 4 滤波器 (Channel 4 digital filter)
位 11: 10	C4IDIV	0x0	rw	通道 4 分频系数 (Channel 4 input divider)
位 9: 8	C4C	0x0	rw	通道 4 配置 (Channel 4 configure) 当 C4EN=0'时, 这些位用于选择通道 4 为输出或输入, 以及输入时的映射选择: 00: 输出; 01: 输入, C4IN 映射在 C4IRAW 上; 10: 输入, C4IN 映射在 C3IRAW 上; 11: 输入, C4IN 映射在 STCI 上, 只有在 STIS 选择内部触发输入时才工作。
位 7: 4	C3DF	0x0	rw	通道 3 滤波器 (Channel 3 digital filter)
位 3: 2	C3IDIV	0x0	rw	通道 3 分频系数 (Channel 3 input divider)
位 1: 0	C3C	0x0	rw	通道 3 配置 (Channel 3 configure) 当 C3EN=0'时, 这些位用于选择通道 3 为输出或输入, 以及输入时的映射选择: 00: 输出; 01: 输入, C3IN 映射在 C3IRAW 上; 10: 输入, C3IN 映射在 C4IRAW 上; 11: 输入, C3IN 映射在 STCI 上, 只有在 STIS 选择内部触发输入时才工作。

#### 14.1.4.9 通道控制寄存器 (TMRx\_CCTRL)

域	简称	复位值	类型	功能
位 15: 14	保留	0x0	resd	保持默认值。
位 13	C4P	0x0	rw	通道 4 极性 (Channel 4 polarity) 见 C1P 的描述。
位 12	C4EN	0x0	rw	通道 4 使能 (Channel 4 enable) 见 C1EN 的描述。
位 11	C3CP	0x0	rw	通道 3 互补极性 (Channel 3 complementary polarity) 定义输入信号的有效沿, 详见 C1P 位描述。
位 10	保留	0x0	resd	保持默认值。
位 9	C3P	0x0	rw	通道 3 极性 (Channel 3 polarity) 见 C1P 的描述。
位 8	C3EN	0x0	rw	通道 3 使能 (Channel 3 enable) 见 C1EN 的描述。
位 7	C2CP	0x0	rw	通道 2 互补极性 (Channel 2 complementary polarity) 定义输入信号的有效沿, 详见 C1P 位描述。
位 6	保留	0x0	resd	保持默认值。
位 5	C2P	0x0	rw	通道 2 极性 (Channel 2 polarity) 见 C1P 的描述。
位 4	C2EN	0x0	rw	通道 2 使能 (Channel 2 enable) 见 C1EN 的描述。

位 3	C1CP	0x0	rw	通道 1 互补极性 (Channel 1 complementary polarity) 定义输入信号的有效沿, 详见 C1P 位描述。
位 2	保留	0x0	resd	保持默认值。
位 1	C1P	0x0	rw	通道 1 极性 (Channel 1 polarity) 通道 1 配置为输出: 0: C1OUT 的有效电平为高 1: C1OUT 的有效电平为低 通道 1 配置为输入: C1CP/C1P 位共同定义输入信号有效沿。 00: C1IN 的有效边沿为上升沿; 作为外部触发使用时, C1IN 不反相。 01: C1IN 的有效边沿为下降沿; 作为外部触发使用时, C1IN 反相。 10: 保留 11: C1IN 的有效边沿为上升沿和下降沿; 作为外部触发使用时, C1IN 不反相。
位 0	C1EN	0x0	rw	通道 1 使能 (Channel 1 enable) 0: 禁止输入或输出; 1: 使能输入或输出。

表 14-5 标准CxOUT通道的输出控制位

CxEN 位	CxOUT 输出状态
0	禁止输出 (CxOUT=0, Cx_EN=0)
1	CxOUT = CxORAW + 极性, Cx_EN=1

注意: 连接到标准 CxOUT 通道的外部 I/O 管脚状态, 取决于 CxOUT 通道状态和 GPIO 以及 IOMUX 寄存器。

#### 14.1.4.10 计数值 (TMRx\_CVAL)

域	简称	复位值	类型	功能
位 31: 16	CVAL	0x0000	rw	计数值 (Counter value) 当 TMR2 或 TMR5 开启增强模式时 (TMR_CTRL1 中的 PMEN 位), CVAL 被扩展为 32 位。
位 15: 0	CVAL	0x0000	rw	计数值 (Counter value)

#### 14.1.4.11 分频系数 (TMRx\_DIV)

域	简称	复位值	类型	功能
位 15: 0	DIV	0x0000	rw	分频系数 (Divider value) 计数器时钟频率 $f_{CK\_CNT} = f_{TMR\_CLK} / (DIV[15: 0]+1)$ 。 DIV 为溢出事件发生时写入的分频系数。

#### 14.1.4.12 周期寄存器 (TMRx\_PR)

域	简称	复位值	类型	功能
位 31: 16	PR	0x0000	rw	周期值 (Period value) 当 TMR2 或 TMR5 开启增强模式时 (TMR_CTRL1 中的 PMEN 位), PR 被扩展为 32 位。
位 15: 0	PR	0x0000	rw	周期值 (Period value) 定时器计数的周期值。当周期值为 0 时, 定时器不工作。

#### 14.1.4.13 通道1数据寄存器 (TMRx\_C1DT)

域	简称	复位值	类型	功能
位 31: 16	C1DT	0x0000	rw	通道 1 数据寄存器值 (Channel 1 data register) 当 TMR2 或 TMR5 开启增强模式时 (TMR_CTRL1 中的 PMEN 位), C1DT 被扩展为 32 位。
位 15: 0	C1DT	0x0000	rw	通道 1 数据寄存器值 (Channel 1 data register) 若通道 1 配置为输入:

---

C1DT 是前一次通道 1 输入事件（C1IN）所保存的 CVAL。

若通道 1 配置为输出：

C1DT 是将要和 CVAL 进行比较的值，写入的值是否会立即生效取决于输出缓存使能位（C1OBEN），并根据设置在 C1OUT 上产生相应的输出。

---

#### 14.1.4.14 通道2数据寄存器 (TMRx\_C2DT)

域	简称	复位值	类型	功能
位 31: 16	C2DT	0x0000	rw	通道 2 数据寄存器值 (Channel 2 data register) 当 TMR2 或 TMR5 开启增强模式时 (TMR_CTRL1 中的 PMEN 位), C2DT 被扩展为 32 位。
位 15: 0	C2DT	0x0000	rw	通道 2 数据寄存器值 (Channel 2 data register) 若通道 1 配置为输入: C2DT 是前一次通道 2 输入事件 (C2IN) 所保存的 CVAL。 若通道 2 配置为输出: C2DT 是将要和 CVAL 进行比较的值, 写入的值是否会立即生效取决于输出缓存使能位 (C2OBEN), 并根据设置在 C2OUT 上产生相应的输出。

#### 14.1.4.15 通道3数据寄存器 (TMRx\_C3DT)

域	简称	复位值	类型	功能
位 31: 16	C3DT	0x0000	rw	通道 3 数据寄存器值 (Channel 3 data register) 当 TMR2 或 TMR5 开启增强模式时 (TMR_CTRL1 中的 PMEN 位), C3DT 被扩展为 32 位。
位 15: 0	C3DT	0x0000	rw	通道 3 数据寄存器值 (Channel 3 data register) 若通道 3 配置为输入: C3DT 是前一次通道 3 输入事件 (C3IN) 所保存的 CVAL。 若通道 3 配置为输出: C3DT 是将要和 CVAL 进行比较的值, 写入的值是否会立即生效取决于输出缓存使能位 (C3OBEN), 并根据设置在 C3OUT 上产生相应的输出。

#### 14.1.4.16 通道4数据寄存器 (TMRx\_C4DT)

域	简称	复位值	类型	功能
位 31: 16	C4DT	0x0000	rw	通道 4 数据寄存器值 (Channel 4 data register) 当 TMR2 或 TMR5 开启增强模式时 (TMR_CTRL1 中的 PMEN 位), C4DT 被扩展为 32 位。
位 15: 0	C4DT	0x0000	rw	通道 4 数据寄存器值 (Channel 4 data register) 若通道 4 配置为输入: C4DT 是前一次通道 4 输入事件 (C4IN) 所保存的 CVAL。 若通道 4 配置为输出: C4DT 是将要和 CVAL 进行比较的值, 写入的值是否会立即生效取决于输出缓存使能位 (C4OBEN), 并根据设置在 C4OUT 上产生相应的输出。

#### 14.1.4.17 DMA控制寄存器 (TMRx\_DMACTRL)

域	简称	复位值	类型	功能
位 15: 13	保留	0x0	resd	保持默认值。
位 12: 8	DTB	0x00	rw	DMA 传输字节 (DMA transfer bytes) 这些位定义了传输的字节个数: 00000: 1 个字节      00001: 2 个字节 00010: 3 个字节      00011: 4 个字节 ..... 10000: 17 个字节      10001: 18 个字节
位 7: 5	保留	0x0	resd	保持默认值。
位 4: 0	ADDR	0x00	rw	DMA 传输地址偏移 (DMA transfer address offset) ADDR 定义了从 TMRx_CTRL1 所在地址开始的偏移量: 00000: TMRx_CTRL1, 00001: TMRx_CTRL2, 00010: TMRx_STCTRL, .....

#### 14.1.4.18 DMA数据寄存器 (TMRx\_DMADT)

域	简称	复位值	类型	功能
位 15: 0	DMADT	0x0000	rw	DMA 传输的数据寄存器 (DMA data register) 通过对 DMADT 寄存器的读写能够实现对任意 TMR 寄存器的操作，其操作的寄存器地址范围是： TMRx 外设地址 + ADDR*4 至 TMRx 外设地址 + ADDR*4 + DTB*4。

## 14.2 通用定时器（TMR9到TMR11）

### 14.2.1 TMRx简介

通用定时器 TMR9 到 TMR11 支持 16 位向上计数，可通过同步功能进行互联。

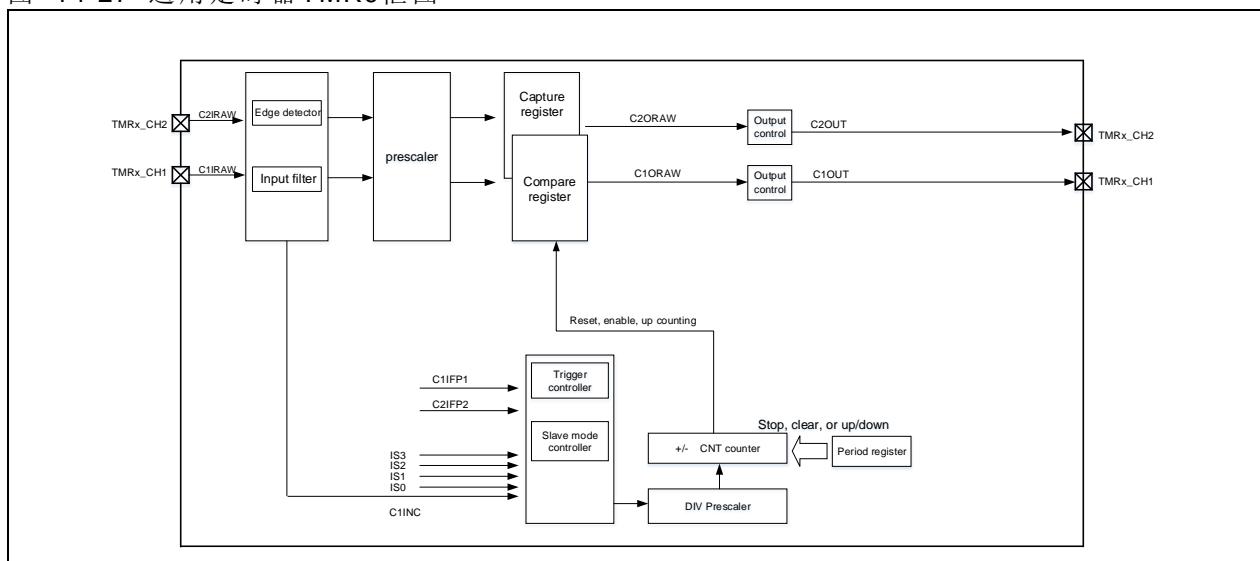
### 14.2.2 TMRx主要特性

#### 14.2.2.1 TMR9主要特性

TMR9 功能包括：

- 可选内部、外部输入用作计数时钟
- 16 位向上计数器
- 2 组独立通道，支持输入捕获、输出比较、PWM 生成、单周期模式
- 定时器之间可互联同步
- 支持溢出事件、触发事件、通道事件触发中断

图 14-27 通用定时器 TMR9 框图

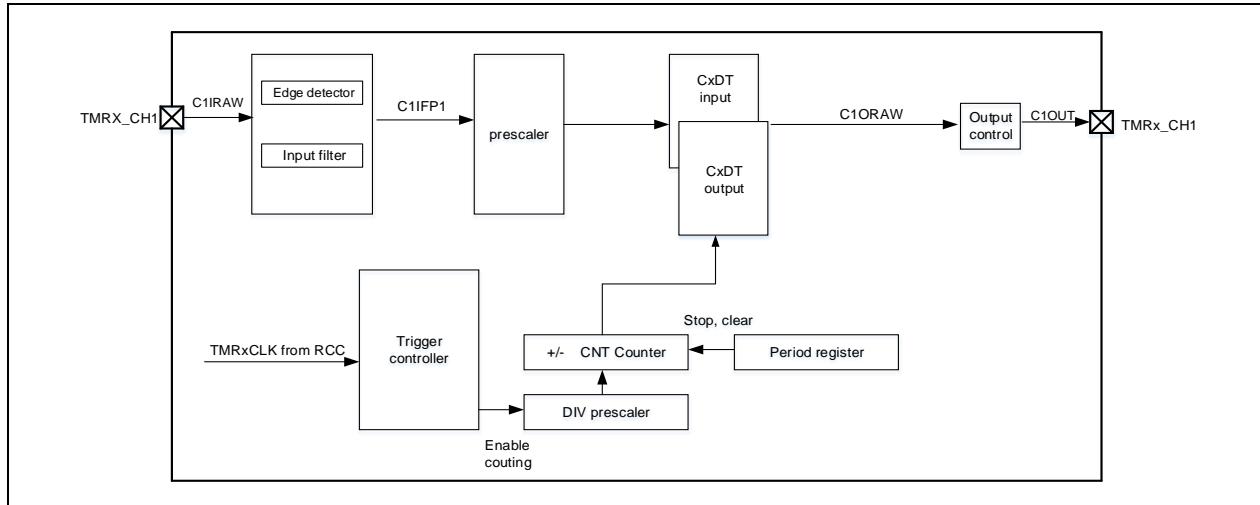


#### 14.2.2.2 TMR10、TMR11主要特性

通用 TMRx (TMR10、TMR11) 定时器功能包括：

- 由内部用作计数时钟
- 16 位向上计数器
- 1 组独立通道，支持输入捕获、输出比较、PWM 生成
- 定时器之间可互联同步
- 支持溢出事件、通道事件触发中断

图 14-28 通用定时器 TMR10/11 框图



## 14.2.3 TMRx 功能描述

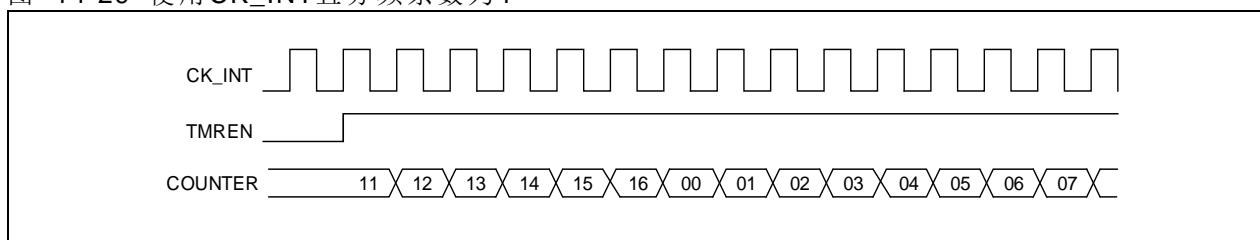
### 14.2.3.1 计数时钟

通用定时器计数时钟可从内部时钟 (CK\_INT)、外部时钟 (外部时钟模式 A)、内部触发输入 (ISx) 这些时钟源提供。

#### 内部时钟 (CK\_INT)

默认下使用 CK\_INT 经由预分频器驱动计数器计数。

图 14-29 使用 CK\_INT 且分频系数为 1

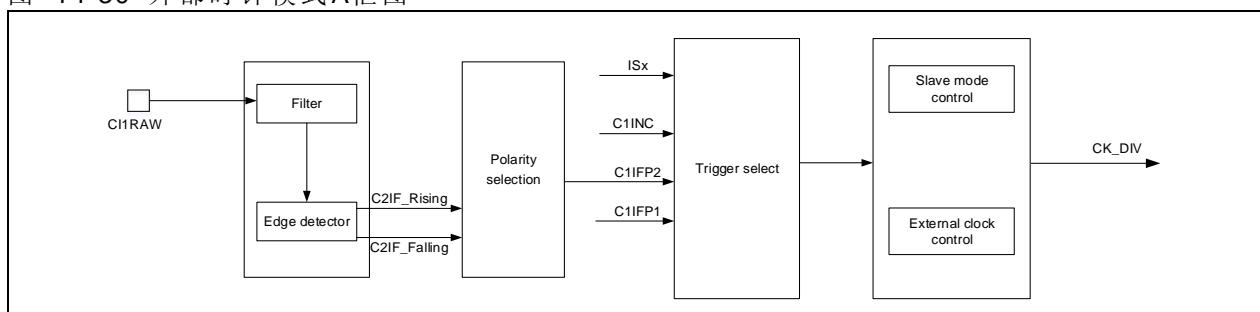


#### 外部时钟 (仅 TMR9)

计数时钟可由选择的 TRGIN 信号提供。

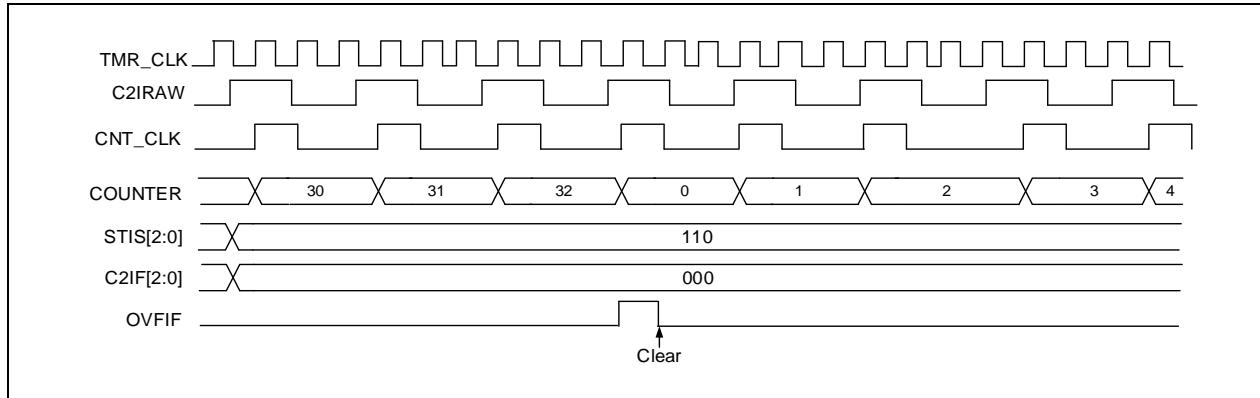
当 SMSEL=3'111 时，外部时钟模式 A 被选中，配置 STIS[2: 0]来选择 TRGIN 信号驱动计数器计数。

图 14-30 外部时钟模式 A 框图



注：由于同步逻辑，输入端信号与计数器实际时钟之间存在一定延时。

图 14-31 使用外部时钟模式A计数



### 内部触发输入 (ISx)

定时器之间支持互联同步，因此一个定时器的 TMR\_CLK 可由另一个定时器输出信号 TRGOUT 提供。配置 STIS[2: 0]选择内部触发信号驱动计数器计数。

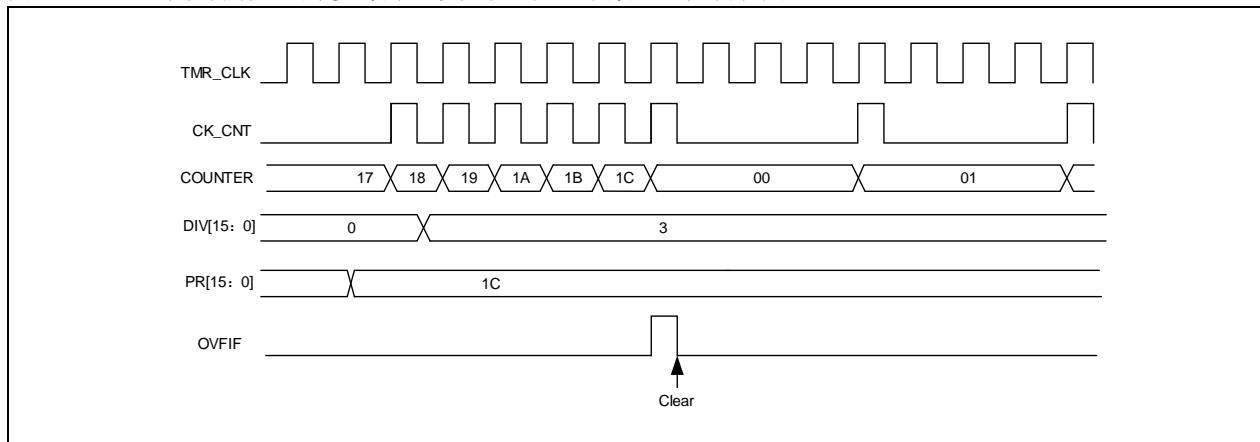
高级定时器内含一个 16 位预分频器，用于产生驱动计数器计数的时钟 CK\_CNT，通过配置 TMRx\_DIV 寄存器值，可灵活调整 CK\_CNT 与 TMR\_CLK 之间的分频关系。预分频值可在任何时刻修改，但只在下一个溢出事件发生时，新值才会生效。

表 14-6 TMRx 内部触发连接

次定时器	ISO (STIS = 000)	IS1 (STIS = 001)	IS2 (STIS = 010)	IS3 (STIS = 011)
TMR9	TMR2	-	TMR10_OC	TMR11_OC

注意：如果某个产品中没有相应的定时器，则对应的触发信号 ISx 也不存在。

图 14-32 当预分频器的参数从 1 变到 4 时，计数器的时序图



### 14.2.3.2 计数模式

通用定时器仅提供向上计数模式，其内部拥有一个支持 16 位计数的计数器，计数器的值可由周期寄存器 (TMRx\_PR) 载入。默认下，周期寄存器 (TMRx\_PR) 值会立即传入它的影子寄存器；当开启周期缓冲功能后 (PRBEN 置 1)，周期寄存器 (TMRx\_PR) 值会在溢出事件发生时传入它的影子寄存器。溢出事件由 OVFEN、OVFS 位配置。

将 TMREN 位置 1 可使能定时器计数，由于同步逻辑，实际驱动计数器的使能信号 TMR\_EN 相对于 TMREN 延迟一个时钟周期。

#### 向上计数模式

向上计数模式中，当计数值达到 TMRx\_PR 值时，重新从 0 向上计数，计数器上溢并产生溢出事件，同时 OVFIF 位置 1。若禁止产生溢出事件，则计数器溢出后不再重载预分频值和周期值，否则预分频值和周期值将在溢出事件后更新。

图 14-33 PRBEN=0时的溢出事件

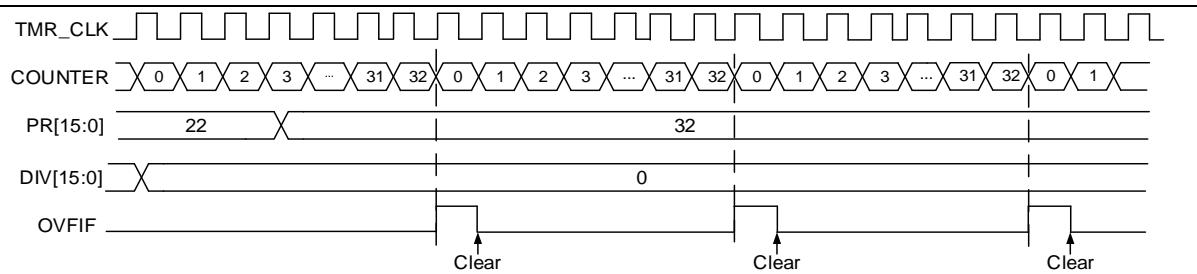
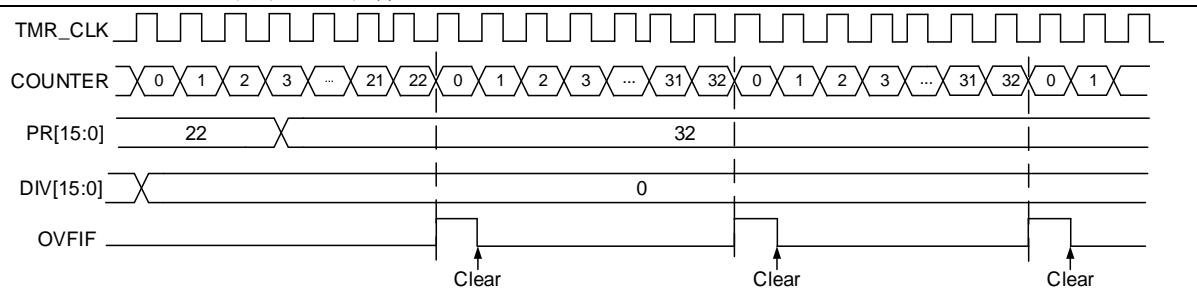


图 14-34 PRBEN=1时的溢出事件



### 14.2.3.3 TMR输入部分

TMR9 拥有两个独立通道，TMR10、11 拥有一个独立通道。每个通道可配置为输入或输出，当配置位输入时，可用于对输入信号的滤波、选择、分频和输入捕获功能。

图 14-35 输入/输出通道1的主电路

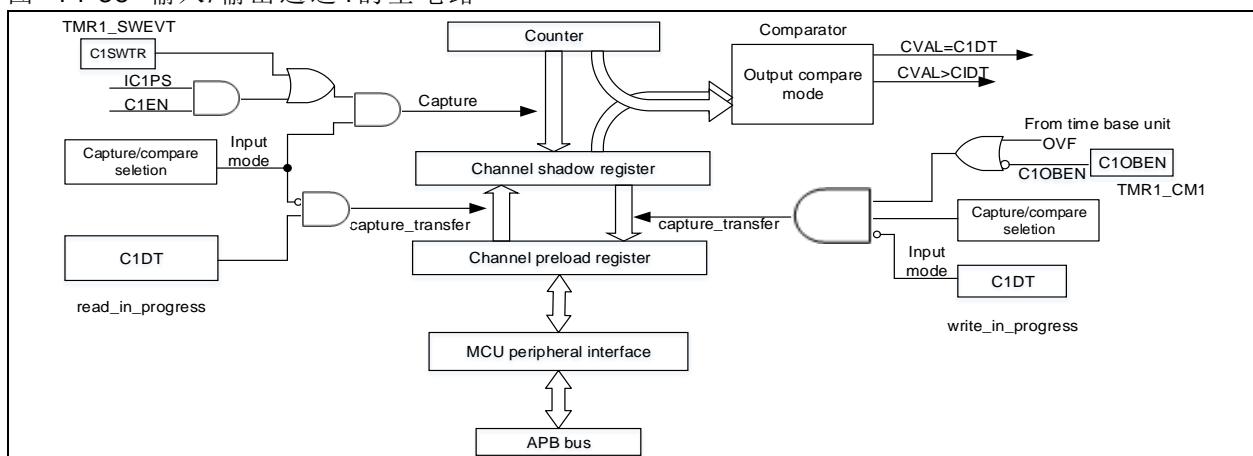
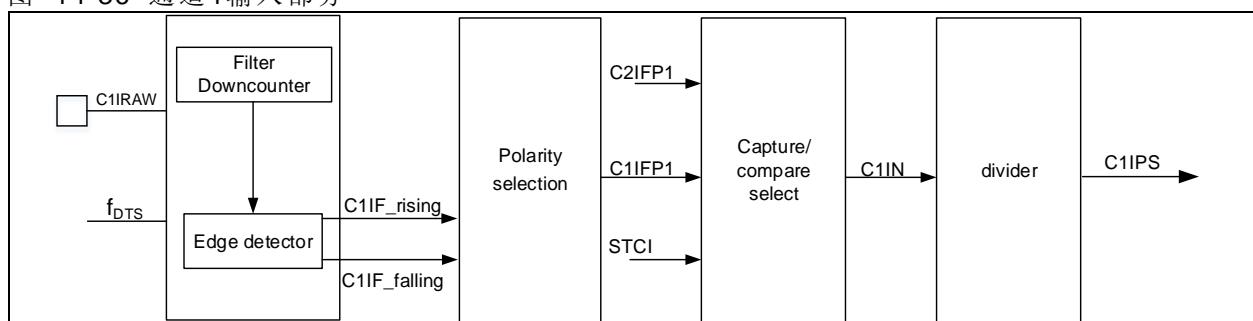


图 14-36 通道1输入部分



#### 输入模式

此模式下，当选中的触发信号被检测到时，通道寄存器（TMRx\_CxDT）会记录当前计数器计数值，并将捕获比较中断标志位（CxIF）置 1，若已使能通道中断（CxIEN）则产生相应的中断请求。若在 CxIF 已置 1 后检测到选中的触发信号，则将 CxRF 置位 1。

若要捕获 C1IN 输入的上升沿，可按如下进行配置：

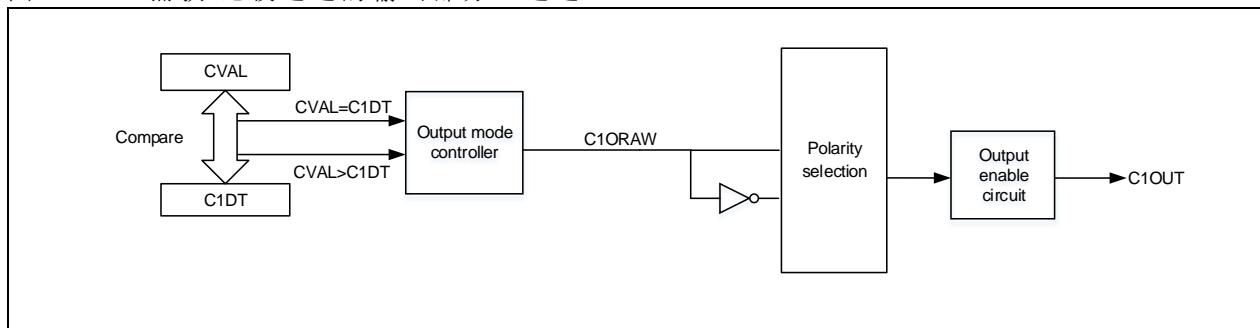
- 将通道寄存器（TMRx\_CxDT）中的 C1C 位配置为 01，选择 C1IN 作为通道 1 输入。
- 配置 C1IN 信号滤波器带宽（CxDF[3: 0]）。

- 配置 C1IN 通道的有效沿，在通道控制寄存器（TMRx\_CCTRL）中写入 C1P=0（上升沿）。
- 配置 C1IN 信号捕获频率（C1DIV[1: 0]）。
- 使能通道 1 输入捕获（C1EN=1）。
- 根据需要设置 DMA/中断使能寄存器（TMRx\_IDEN）中的 C1IEN 位，选择中断请求。

#### 14.2.3.4 TMR输出部分

TMR 的输出部分由比较器和输出控制构成，用于编程输出信号的周期、占空比、极性。

图 14-37 捕获/比较通道的输出部分（通道1）



##### 输出模式

配置  $CxC[2: 0]\neq 2'b00$  将通道配置为输出可实现多种输出模式，此时，计数器计数值将与通道寄存器（TMRx\_CxDT）值比较，并根据  $CxOCTRL[2: 0]$  位配置的输出模式，产生中间信号  $CxORAW$ ，再经过输出控制逻辑处理后输送到 IO。输出信号的周期由周期寄存器（TMRx\_PR）值配置，占空比则由通道寄存器（TMRx\_CxDT）值配置。

输出比较模式有以下子类：

- **PWM 模式：**  $CxOCTRL=3'b110/111$  时，开启 PWM 模式，每个通道可独立配置并输出一路 PWM 信号。此时，输出信号的周期由 TMRx\_PR 配置，占空比由  $CxDT$  值配置，计数器值与通道寄存器（TMRx\_CxDT）值进行比较，根据计数方向输出指定电平信号，关于 PWM 模式 A/B 详见  $CxOCTRL[2: 0]$  位描述。当计数模式为中央双向对齐计数时，可根据 OWCDIR 位指示计数方向。
- **强制输出模式：**  $CxOCTRL=3'b100/101$  时，开启强制输出模式。此时， $CxORAW$  信号的电平被强制输出为配置的电平，而与计数值无关。虽然输出信号不依赖于比较结果，但通道标志位和 DMA 请求仍依赖于比较结果。
- **输出比较模式：**  $CxOCTRL=2'b001/010/011$  时，开启输出比较模式。此时，当计数值与  $CxDT$  值匹配时， $CxORAW$  被强制为高电平、低电平或进行电平翻转。
- **单周期模式(仅 TMR9/12)：** PWM 模式的特例，将 OCMEN 位置 1 可开启单周期模式，此模式下，仅在当前计数周期中进行比较匹配，完成当前计数后，TMREN 位清 0，因此仅输出一个脉冲。当配置为向上计数模式时，需要严格配置  $CVAL < CxDT \leq PR$ ；向下计数时，需严格配置  $CVAL > CxDT$ 。
- **快速输出模式(仅 TMR9/12)：** 将 CxOIEN 位置 1 可开启此功能，开启后  $CxORAW$  电平值不再在计数值与  $CxDT$  匹配时变化，而是在当前计数周期开始时，也就是说，比较结果被提前了，计数器值与通道寄存器（TMRx\_CxDT）的比较结果将会提前决定  $CxORAW$  的电平。

图 14-38 展示了输出比较模式（翻转）的例子， $C1DT=0x3$ ，当计数值等于  $0x3$  时，输出电平  $C1OUT$  被翻转。

图 14-39 展示了计数器向上计数与 PWM 模式 A 配合的例子， $PR=0x32$ ， $CxDT$  配置为不同的值时输出时输出信号的翻转情况。

图 14-40 展示了计数器向上计数与单周期模式下 PWM 模式 B 配合的例子，计数器仅计数了一个周期，输出信号在这个周期中只输出了一个脉冲。

图 14-38 计数值与C1DT值匹配时翻转C1ORAW

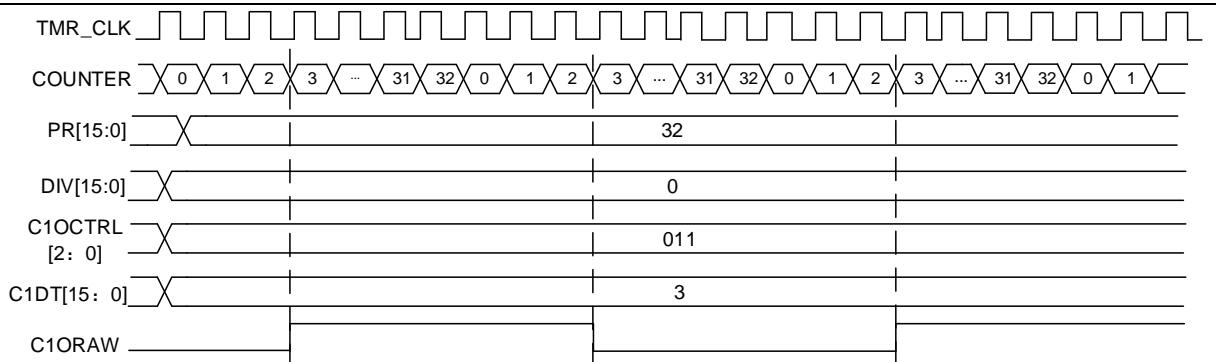


图 14-39 向上计数下 PWM 模式 A

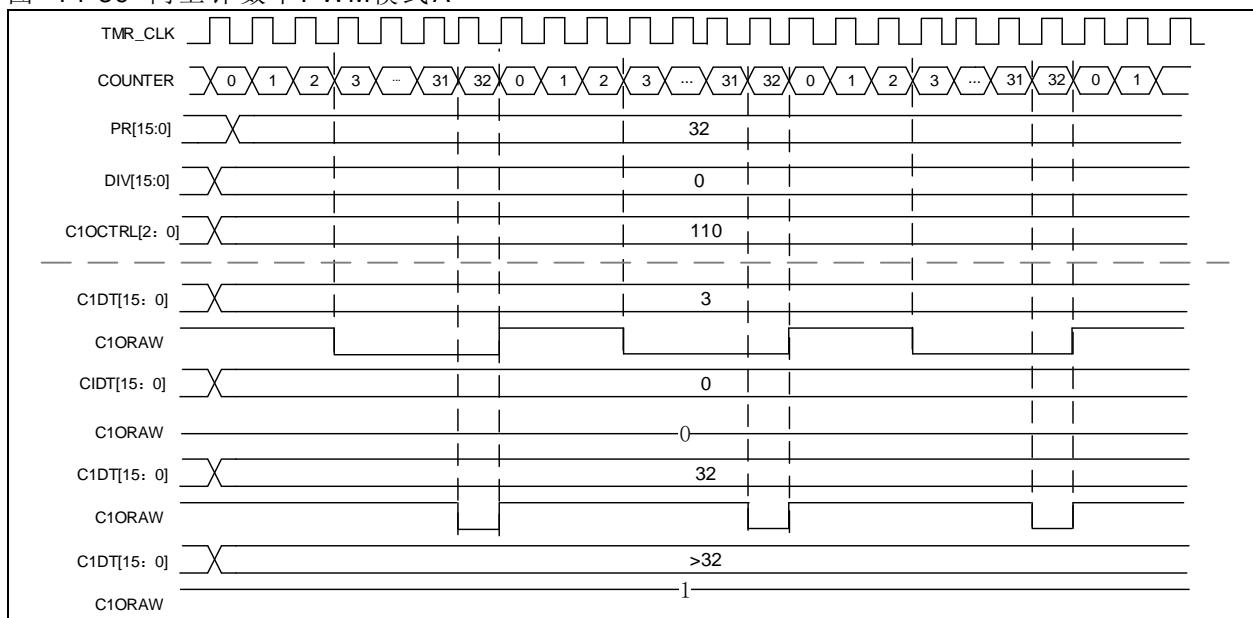
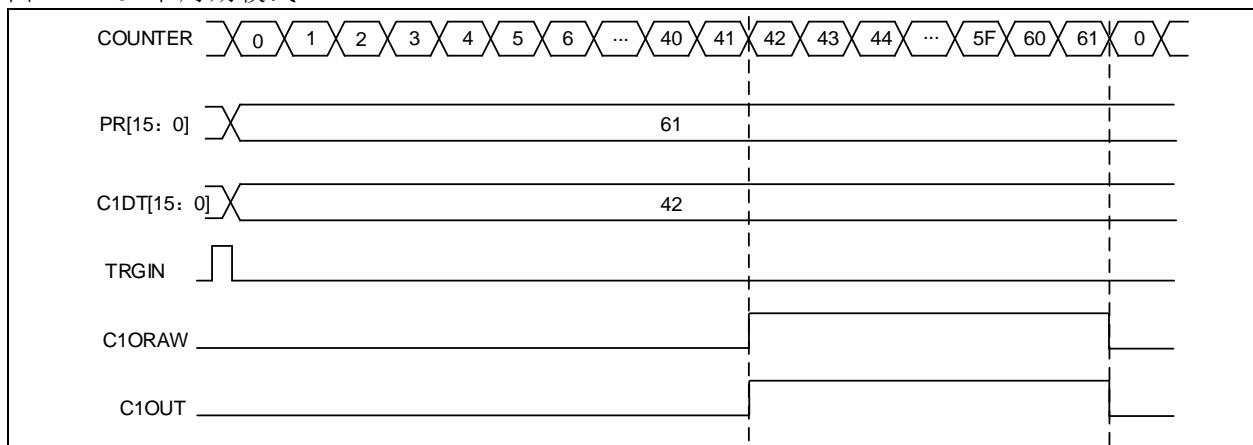


图 14-40 单周期模式



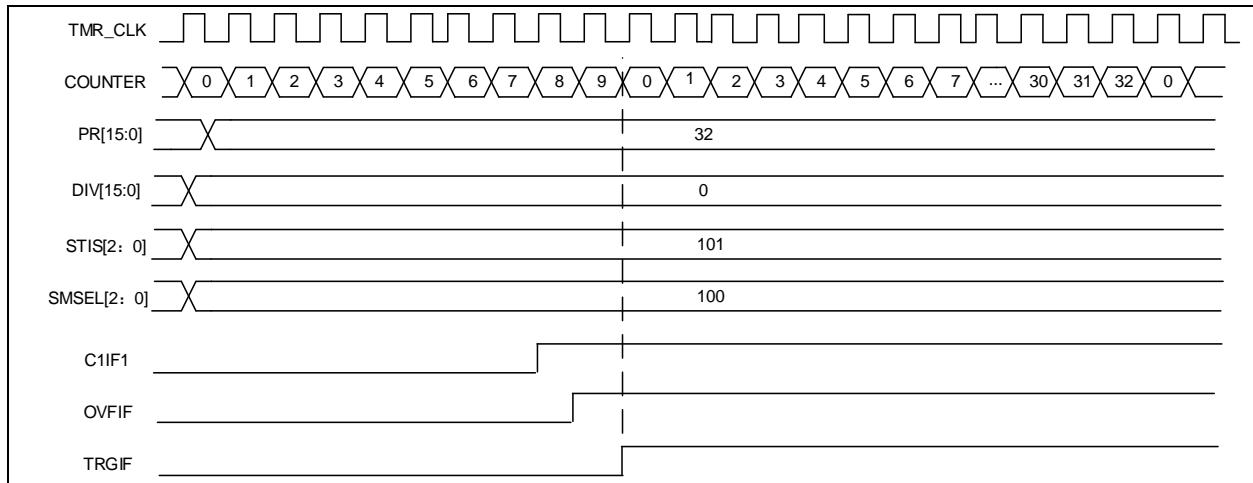
### 14.2.3.5 TMR同步

TMR9 可作为次定时器与主定时器由内部信号进行同步，次定时器由 **SMSEL[2: 0]**位选择从模式，即次定时器的工作模式。

#### 从模式：复位模式

选中的触发信号将复位计数器和预分频器，若 OVFS 位为 0，将产生一个溢出事件。

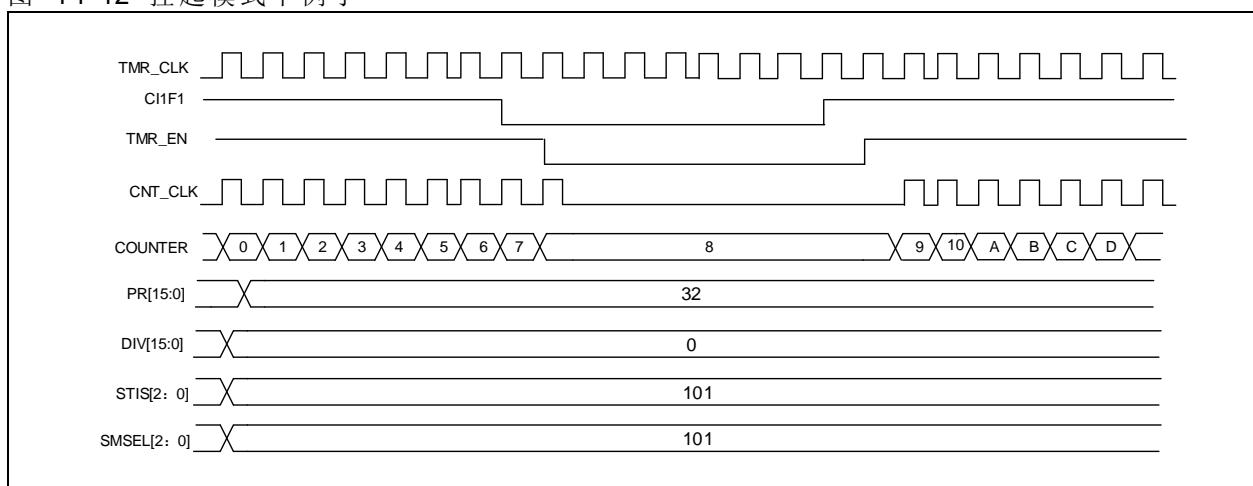
图 14-41 复位模式例子



### 从模式：挂起模式

挂起模式下，计数的计数和停止受选中触发输入信号控制，当触发输入为高电平时计数器开始计数；当为低电平时，计数器暂停计数。

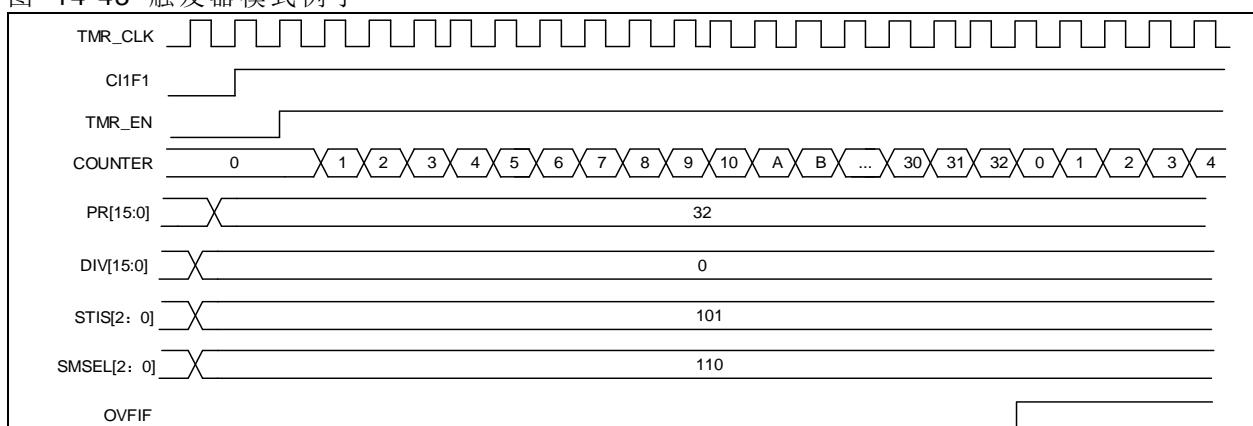
图 14-42 挂起模式下例子



### 从模式：触发模式

计数器将在选中的触发输入上升沿启动计数（将 TMR\_EN 置 1）。

图 14-43 触发器模式例子



定时器的同步的更多实例详见 [14.2.3.5 节](#)。

### 14.2.3.6 调试模式

当微控制器进入调试模式（Cortex™-M4 核心停止）时，将 DEBUG 模块中的 TMRx\_PAUSE 置 1，可以使 TMRx 计数器暂停计数。

#### 14.2.4 TMR9寄存器描述

必须以字（32位）的方式操作这些外设寄存器。

下表中将TMR9的所有寄存器映射到一个16位可寻址（编址）空间。

表 14-7 TMR9寄存器映像和复位值

寄存器简称	基址偏移量	复位值
TMR9_CTRL1	0x00	0x0000
TMR9_STCTRL	0x08	0x0000
TMR9_IDEN	0x0C	0x0000
TMR9ISTS	0x10	0x0000
TMR9_SWEVT	0x14	0x0000
TMR9_CM1	0x18	0x0000
TMR9_CCTRL	0x20	0x0000
TMR9_CVAL	0x24	0x0000
TMR9_DIV	0x28	0x0000
TMR9_PR	0x2C	0x0000
TMR9_C1DT	0x34	0x0000 0000
TMR9_C2DT	0x38	0x0000 0000

##### 14.2.4.1 控制寄存器 1 (TMR9\_CTRL1)

域	简称	复位值	类型	功能
位 15: 10	保留	0x00	resd	保持默认值。
位 9: 8	CLKDIV	0x0	rw	时钟除频 (Clock divider) 00: 无除频; 01: 2 除频; 10: 4 除频; 11: 保留。
位 7	PRBEN	0x0	rw	周期缓冲使能 (Period buffer enable) 0: 缓冲关闭; 1: 缓冲开启。
位 6: 4	保留	0x0	resd	保持默认值。
位 3	OCMEN	0x0	rw	单周期使能 (One cycle mode enable) 该功能用于选择溢出事件后，计数器是否停止。 0: 关闭; 1: 开启。
位 2	OVFS	0x0	rw	溢出事件源选择 (Overflow event source) 配置溢出事件或 DMA 请求来源。 0: 来源于计数器溢出、设置 OVFSWTR 位或次定时器控制器产生的溢出事件; 1: 只能来源于计数器溢出。
位 1	OVFEN	0x0	rw	溢出事件使能 (Overflow event enable) 0: 开启; 1: 关闭。
位 0	TMREN	0x0	rw	使能定时器 (TMR enable) 0: 关闭; 1: 开启。

##### 14.2.4.2 次定时器控制寄存器 (TMR9\_STCTRL)

域	简称	复位值	类型	功能
位 15: 7	保留	0x000	resd	保持默认值。
位 6: 4	STIS	0x0	rw	次定时器输入选择 (Subordinate TMR input selection) 用于次定时器的输入选择。 000: 内部选择 0 (ISO);

位 3	保留	0x0	resd	001: 内部选择 1 (IS1) ; 010: 内部选择 2 (IS2) ; 011: 内部选择 3 (IS3) ; 100: C1IRAW 的输入检测器 (C1INC) ; 101: 滤波输入 1 (C1IF1) ; 110: 滤波输入 2 (C2IF2) ; 111: 保留。 关于每个定时器中 ISx 的细节，参见表 14-7。
位 2: 0	SMSEL	0x0	rw	次定时器模式选择 (Subordinate TMR mode selection) 000: 关闭从模式; 001: 编码模式 A; 010: 编码模式 B; 011: 编码模式 C; 100: 复位模式 - TRGIN 输入上升沿时，重新初始化计数器; 101: 挂起模式 - TRGIN 输入高电平时，计数器计数; 110: 触发模式 - TRGIN 输入上升沿时，产生触发事件; 111: 外部时钟模式 A - TRGIN 输入上升沿时，提供时钟; 注：编码器模式 A/B/C 配置方法请查看计数模式章节。

#### 14.2.4.3 DMA/中断使能寄存器 (TMR9\_IDEN)

域	简称	复位值	类型	功能
位 15: 7	保留	0x000	resd	保持默认值。
位 6	TIEN	0x0	rw	触发中断使能 (Trigger interrupt enable) 0: 关闭; 1: 开启。
位 5: 3	保留	0x0	resd	保持默认值。
位 2	C2IEN	0x0	rw	通道 2 中断使能 (Channel 2 interrupt enable) 0: 关闭; 1: 开启。
位 1	C1IEN	0x0	rw	通道 1 中断使能 (Channel 1 interrupt enable) 0: 关闭; 1: 开启。
位 0	OVIEN	0x0	rw	溢出中断使能 (overflow interrupt enable) 0: 关闭; 1: 开启。

#### 14.2.4.4 中断状态寄存器 (TMR9ISTS)

域	简称	复位值	类型	功能
位 15: 11	保留	0x00	resd	保持默认值。
位 10	C2RF	0x0	rw0c	通道 2 再捕获标记 (Channel 2 recapture flag) 见 C1RF 的描述。
位 9	C1RF	0x0	rw0c	通道 1 再捕获标记 (Channel 1 recapture flag) C1IF 的状态已经为'1'时是否再次发生了捕获，由硬件置'1'，写'0'清除。 0: 无捕获发生; 1: 捕获发生。
位 8: 7	保留	0x0	resd	保持默认值。
位 6	TRGIF	0x0	rw0c	触发中断标记 (Trigger interrupt flag) 当发生触发事件时由硬件置'1'，写'0'清除。 0: 无触发事件发生; 1: 发生触发事件。 触发事件：在 TRGIN 接收到有效边沿，或挂起模式下接收到任意边沿。
位 5: 3	保留	0x0	resd	保持默认值。

位 2	C2IF	0x0	rw0c	通道 2 中断标记 (Channel 2 interrupt flag) 参考 C1IF 描述。
位 1	C1IF	0x0	rw0c	通道 1 中断标记 (Channel 1 interrupt flag) 若通道 1 为输入模式时： 捕获事件发生时由硬件置'1'，由软件清'0'或读 TMR9_C1DT 清'0'。 0: 无捕获事件发生； 1: 发生捕获事件。 若通道 1 为输出模式时： 比较事件发生时由硬件置'1'，由软件清'0'。 0: 无比较事件发生； 1: 发生比较事件。
位 0	OVFIF	0x0	rw0c	溢出中断标记 (Overflow interrupt flag) 当溢出事件发生时由硬件置'1'，由软件清'0'。 0: 无溢出事件发生； 1: 发生溢出事件；

#### 14.2.4.5 软件事件寄存器 (TMR9\_SWEVT)

域	简称	复位值	类型	功能
位 15: 7	保留	0x000	resd	保持默认值。
位 6	TRGSWTR	0x0	rw	软件触发触发事件 (Trigger event triggered by software) 通过软件触发一个触发事件。 0: 无作用； 1: 制造一个触发事件。
位 5: 3	保留	0x0	resd	保持默认值。
位 2	C2SWTR	0x0	wo	软件触发通道 2 事件 (Channel 2 event triggered by software) 见 C1M 的描述。
位 1	C1SWTR	0x0	wo	软件触发通道 1 事件 (Channel 1 event triggered by software) 通过软件触发一个通道 1 事件。 0: 无作用； 1: 制造一个通道 1 事件。
位 0	OVFSWTR	0x0	wo	软件触发溢出事件 (Overflow event triggered by software) 通过软件触发一个溢出事件。 0: 无作用； 1: 制造一个溢出事件。

#### 14.2.4.6 通道模式寄存器1 (TMR9\_CM1)

通道可用于输入（捕获模式）或输出（比较模式），通道的方向由相应的 CxC 定义。该寄存器其它位的作用在输入和输出模式下不同。CxOx 描述了通道在输出模式下的功能，CxIx 描述了通道在输出模式下的功能。因此必须注意，同一个位在输出模式和输入模式下的功能是不同的。

##### 输出比较模式：

域	简称	复位值	类型	功能
位 15	保留	0x0	resd	保持默认值
位 14: 12	C2OCTRL	0x0	rw	通道 2 输出控制 (Channel 2 output control)
位 11	C2OBEN	0x0	rw	通道 2 输出缓存使能 (Channel 2 output buffer enable)
位 10	C2OIEN	0x0	rw	通道 2 输出立即使能 (Channel 2 output immediately enable)
位 9: 8	C2C	0x0	rw	通道 2 配置 (Channel 2 configure) 当 C2EN='0'时，这些位用于选择通道 2 为输出或输入， 以及输入时的映射选择： 00: 输出； 01: 输入，C2IN 映射在 C2IRAW 上； 10: 输入，C2IN 映射在 C1IRAW 上； 11: 输入，C2IN 映射在 STCI 上，只有在 STIS 选择内部 触发输入时才工作。
位 7	保留	0x0	resd	保持默认值
位 6: 4	C1OCTRL	0x0	rw	通道 1 输出控制 (Channel 1 output control)

这些位用于设置原始信号 C1ORAW 的工作状态。

000: 断开。断开 C1ORAW 到 C1OUT 的输出;

001: 设置 C1ORAW 为高: TMR9\_CVAL=TMR9\_C1DT 时。

010: 设置 C1ORAW 为低: TMR9\_CVAL=TMR9\_C1DT 时。

011: 切换 C1ORAW 的电平: 当 TMR9\_CVAL=TMR9\_C1DT 时。

100: 固定 C1ORAW 为低。

101: 固定 C1ORAW 为高。

110: PWM 模式 A

—OWCDIR=0, 若 TMR9\_C1DT>TMR9\_CVAL 时设置 C1ORAW 为高, 否则为低;

—OWCDIR=1, 若 TMR9\_C1DT < TMR9\_CVAL 时设置 C1ORAW 为低, 否则为高。

111: PWM 模式 B

—OWCDIR=0, 若 TMR9\_C1DT > TMR9\_CVAL 时设置 C1ORAW 为低, 否则为高;

—OWCDIR=1, 若 TMR9\_C1DT < TMR9\_CVAL 时设置 C1ORAW 为高, 否则为低。

注: 除'000'外, 其余配置下 C1OUT 将连接到 C1ORAW, C1OUT 的输出电平除了会根据 C1ORAW 变化外, 还与 CCTRL 所配置的输出极性有关。

位 3	C1OBEN	0x0	rw	通道 1 输出缓存使能 (Channel 1 output buffer enable) 0: 关闭 TMR9_C1DT 的缓存功能, 写入 TMR9_C1DT 的内容会立即生效。 1: 启用 TMR9_C1DT 的缓存功能, 写入 TMR9_C1DT 的内容将保存到缓存寄存器中, 当发生溢出事件时再更新到 TMR9_C1DT 中。
位 2	C1OIEN	0x0	rw	通道 1 输出立即使能 (Channel 1 output immediately enable) 在 PWM 模式 A 或模式 B 下, 该位能够缩短触发事件到通道 1 的输出响应间的时间。 0: 需要比较 CVAL 与 C1DT 的值之后再产生输出。 1: 无需比较 CVAL 与 C1DT 的值, 当发生触发事件时立即产生输出。
位 1: 0	C1C	0x0	rw	通道 1 配置 (Channel 1 configure) 当 C1EN=0'时, 这些位用于选择通道 1 为输出或输入, 以及输入时的映射选择: 00: 输出; 01: 输入, C1IN 映射在 C1IRAW 上; 10: 输入, C1IN 映射在 C2IRAW 上; 11: 输入, C1IN 映射在 STCI 上, 只有在 STIS 选择内部触发输入时才工作。

#### 输入模式:

域	简称	复位值	类型	功能
位 15: 12	C2DF	0x0	rw	通道 2 滤波器 (Channel 2 digital filter)
位 11: 10	C2IDIV	0x0	rw	通道 2 分频系数 (Channel 2 input divider)
位 9: 8	C2C	0x0	rw	通道 2 配置 (Channel 2 configure) 当 C2EN=0'时, 这些位用于选择通道 2 为输出或输入, 以及输入时的映射选择: 00: 输出; 01: 输入, C2IN 映射在 C2IRAW 上; 10: 输入, C2IN 映射在 C1IRAW 上; 11: 输入, C2IN 映射在 STCI 上, 只有在 STIS 选择内部触发输入时才工作。
位 7: 4	C1DF	0x0	rw	通道 1 滤波器 (Channel 1 digital filter) 这些位用于配置通道 1 的滤波器。滤波的个数为 N, 则表示发生了 N 次采样事件后输入边沿才能通过滤波器: 0000: 无滤波器, 以 $f_{DTS}$ 采样

				1000: 采样频率 $f_{SAMPLING} = f_{DTS}/8$ , N=6 0001: 采样频率 $f_{SAMPLING} = f_{CK\_INT}$ , N=2 1001: 采样频率 $f_{SAMPLING} = f_{DTS}/8$ , N=8 0010: 采样频率 $f_{SAMPLING} = f_{CK\_INT}$ , N=4 1010: 采样频率 $f_{SAMPLING} = f_{DTS}/16$ , N=5 0011: 采样频率 $f_{SAMPLING} = f_{CK\_INT}$ , N=8 1011: 采样频率 $f_{SAMPLING} = f_{DTS}/16$ , N=6 0100: 采样频率 $f_{SAMPLING} = f_{DTS}/2$ , N=6 1100: 采样频率 $f_{SAMPLING} = f_{DTS}/16$ , N=8 0101: 采样频率 $f_{SAMPLING} = f_{DTS}/2$ , N=8 1101: 采样频率 $f_{SAMPLING} = f_{DTS}/32$ , N=5 0110: 采样频率 $f_{SAMPLING} = f_{DTS}/4$ , N=6 1110: 采样频率 $f_{SAMPLING} = f_{DTS}/32$ , N=6 0111: 采样频率 $f_{SAMPLING} = f_{DTS}/4$ , N=8 1111: 采样频率 $f_{SAMPLING} = f_{DTS}/32$ , N=8
位 3: 2	C1IDIV	0x0	rw	通道 1 分频系数 (Channel 1 input divider) 这些位定义了通道 1 的分频系数。 00: 不分频, 每一个有效的边沿都会产生一次输入; 01: 每 2 个有效的边沿产生一次输入; 10: 每 4 个有效的边沿产生一次输入; 11: 每 8 个有效的边沿产生一次输入。 注: C1EN='0'时, 分频系数复位。
位 1: 0	C1C	0x0	rw	通道 1 配置 (Channel 1 configure) 当 C1EN='0'时, 这些位用于选择通道 1 为输出或输入, 以及输入时的映射选择: 00: 输出; 01: 输入, C1IN 映射在 C1IRAW 上; 10: 输入, C1IN 映射在 C2IRAW 上; 11: 输入, C1IN 映射在 STCI 上, 只有在 STIS 选择内部触发输入时才工作。

#### 14.2.4.7 通道控制寄存器 (TMR9\_CCTRL)

域	简称	复位值	类型	功能
位 15: 8	保留	0x00	resd	保持默认值。
位 7	C2CP	0x0	rw	通道 2 互补极性 (Channel 2 complementary polarity) 定义输入信号的有效沿, 详见 C1P 位描述。
位 6	保留	0x0	resd	保持默认值。
位 5	C2P	0x0	rw	通道 2 极性 (Channel 2 polarity) 见 C1P 的描述。
位 4	C2EN	0x0	rw	通道 2 使能 (Channel 2 enable) 见 C1EN 的描述。
位 3	C1CP	0x0	rw	通道 1 互补极性 (Channel 1 complementary polarity) 定义输入信号的有效沿, 详见 C1P 位描述。
位 2	保留	0x0	resd	保持默认值。
位 1	C1P	0x0	rw	通道 1 极性 (Channel 1 polarity) 通道 1 配置为输出: 0: C1OUT 的有效电平为高 1: C1OUT 的有效电平为低 通道 1 配置为输入: C1CP/C1P 位共同定义输入信号有效沿。 00: C1IN 的有效边沿为上升沿; 作为外部触发使用时, C1IN 不反相。 01: C1IN 的有效边沿为下降沿; 作为外部触发使用时, C1IN 反相。 10: 保留 11: C1IN 的有效边沿为上升沿和下降沿; 作为外部触发使用时, C1IN 不反相。
位 0	C1EN	0x0	rw	通道 1 使能 (Channel 1 enable) 0: 禁止输入或输出; 1: 使能输入或输出。

表 14-8 标准CxOUT通道的输出控制位

CxEN 位	CxOUT 输出状态
0	禁止输出 (CxOUT=0)
1	CxOUT = CxORAW + 极性

注意：连接到标准CxOUT通道的外部I/O管脚状态，取决于CxOUT通道状态和GPIO以及IOMUX寄存器。

#### 14.2.4.8 计数器 (TMR9\_CVAL)

域	简称	复位值	类型	功能
位 15: 0	CVAL	0x0000	rw	计数值 (Counter value)

#### 14.2.4.9 预分频器 (TMR9\_DIV)

域	简称	复位值	类型	功能
位 15: 0	DIV	0x0000	rw	分频系数 (Divider value) 计数器时钟频率 $f_{CK\_CNT} = f_{TMR\_CLK} / (DIV[15:0]+1)$ 。 DIV 为溢出事件发生时写入的分频系数。

#### 14.2.4.10 周期寄存器 (TMR9\_PR)

域	简称	复位值	类型	功能
位 15: 0	PR	0x0000	rw	周期值 (Period value) 定时器计数的周期值。当周期值为 0 时，定时器不工作。

#### 14.2.4.11 通道1数据寄存器 (TMR9\_C1DT)

域	简称	复位值	类型	功能
位 31: 16	保留	0x0000	resd	保持默认值 通道 1 数据寄存器值 (Channel 1 data register) 若通道 1 配置为输入： C1DT 是前一次通道 1 输入事件 (C1IN) 所保存的 CVAL。
位 15: 0	C1DT	0x0000	rw	若通道 1 配置为输出： C1DT 是将要和 CVAL 进行比较的值，写入的值是否会立即生效取决于输出缓存使能位 (C1OBEN)，并根据设置在 C1OUT 上产生相应的输出。

#### 14.2.4.12 通道2数据寄存器 (TMR9\_C2DT)

域	简称	复位值	类型	功能
位 31: 16	保留	0x0000	resd	保持默认值 通道 2 数据寄存器值 (Channel 2 data register) 若通道 1 配置为输入： C2DT 是前一次通道 2 输入事件 (C2IN) 所保存的 CVAL。
位 15: 0	C2DT	0x0000	rw	若通道 2 配置为输出： C2DT 是将要和 CVAL 进行比较的值，写入的值是否会立即生效取决于输出缓存使能位 (C2OBEN)，并根据设置在 C2OUT 上产生相应的输出。

#### 14.2.5 TMR10、TMR11寄存器描述

必须以字（32 位）的方式操作这些外设寄存器。

下表中将 TMR10、TMR11 的所有寄存器映射到一个 16 位可寻址（编址）空间。

表 14-9 TMR10、TMR11寄存器映像和复位值

寄存器简称	基址偏移量	复位值
TMRx_CTRL1	0x00	0x0000
TMRx_IDEN	0x0C	0x0000

TMRxISTS	0x10	0x0000
TMRxSWEVT	0x14	0x0000
TMRxCM1	0x18	0x0000
TMRxCCTRL	0x20	0x0000
TMRxCVAL	0x24	0x0000
TMRxDIV	0x28	0x0000
TMRxPR	0x2C	0x0000
TMRxC1DT	0x34	0x0000

#### 14.2.5.1 控制寄存器1 (TMRx\_CTRL1)

域	简称	复位值	类型	功能
位 15: 10	保留	0x00	resd	保持默认值。
位 9: 8	CLKDIV	0x0	rw	时钟除频 (Clock divider) 00: 无除频; 01: 2 除频; 10: 4 除频; 11: 保留。
位 7	PRBEN	0x0	rw	周期缓冲使能 (Period buffer enable) 0: 缓冲关闭; 1: 缓冲开启。
位 6: 3	保留	0x0	resd	保持默认值。
位 2	OVFS	0x0	rw	溢出事件源选择 (Overflow event source) 配置溢出事件或 DMA 请求来源。 0: 来源于计数器溢出、设置 OVFSWTR 位或次定时器控制器产生的溢出事件; 1: 只能来源于计数器溢出。
位 1	OVFEN	0x0	rw	溢出事件使能 (Overflow event enable) 0: 开启; 1: 关闭。
位 0	TMREN	0x0	rw	使能定时器 (TMR enable) 0: 关闭; 1: 开启。

#### 14.2.5.2 DMA/中断使能寄存器 (TMRx\_IDEN)

域	简称	复位值	类型	功能
位 15: 2	保留	0x0000	resd	保持默认值。
位 1	C1IEN	0x0	rw	通道 1 中断使能 (Channel 1 interrupt enable) 0: 关闭; 1: 开启。
位 0	OVFIEN	0x0	rw	溢出中断使能 (overflow interrupt enable) 0: 关闭; 1: 开启。

#### 14.2.5.3 中断状态寄存器 (TMRxISTS)

域	简称	复位值	类型	功能
位 15: 10	保留	0x00	resd	保持默认值。
位 9	C1RF	0x0	rw0c	通道 1 再捕获标记 (Channel 1 recapture flag) C1IF 的状态已经为'1'时是否再次发生了捕获, 由硬件置'1', 写'0'清除。 0: 无捕获发生; 1: 捕获发生。
位 8: 2	保留	0x00	resd	保持默认值。
位 1	C1IF	0x0	rw0c	通道 1 中断标记 (Channel 1 interrupt flag) 若通道 1 为输入模式时:

位 0	OVFIF	0x0	rw0c	<p>捕获事件发生时由硬件置'1'，由软件清'0'或读 TMRx_C1DT 清'0'。</p> <p>0: 无捕获事件发生； 1: 发生捕获事件。 若通道 1 为输出模式时： 比较事件发生时由硬件置'1'，由软件清'0'。 0: 无比较事件发生； 1: 发生比较事件。</p> <p>溢出中断标记（Overflow interrupt flag） 当溢出事件发生时由硬件置'1'，由软件清'0'。 0: 无溢出事件发生； 1: 发生溢出事件，若 TMRx_CTRL1 的 OVFEN=0、 OVFS=0 时： - 当 TMRx_SWEVE 寄存器的 OVFG=1 时产生溢出事件； - 当计数值 CVAL 被触发事件重初始化时产生溢出事件。</p>
-----	-------	-----	------	--

#### 14.2.5.4 软件事件寄存器 (TMRx\_SWEVT)

域	简称	复位值	类型	功能
位 15: 2	保留	0x0000	resd	保持默认值。
位 1	C1SWTR	0x0	wo	<p>软件触发通道 1 事件 (Channel 1 event triggered by software)</p> <p>通过软件触发一个通道 1 事件。 0: 无作用； 1: 制造一个通道 1 事件。</p>
位 0	OVFSWTR	0x0	wo	<p>软件触发溢出事件 (Overflow event triggered by software)</p> <p>通过软件触发一个溢出事件。 0: 无作用； 1: 制造一个溢出事件。</p>

#### 14.2.5.5 通道模式寄存器1 (TMRx\_CM1)

通道可用于输入（捕获模式）或输出（比较模式），通道的方向由相应的 CxC 定义。该寄存器其它位的作用在输入和输出模式下不同。CxOx 描述了通道在输出模式下的功能，CxIx 描述了通道在输出模式下的功能。因此必须注意，同一个位在输出模式和输入模式下的功能是不同的。

##### 输出比较模式：

域	简称	复位值	类型	功能
位 15: 7	保留	0x000	resd	保持默认值。
位 6: 4	C1OCTRL	0x0	rw	<p>通道 1 输出控制 (Channel 1 output control) 这些位用于设置原始信号 C1ORAW 的工作状态。 000: 断开。断开 C1ORAW 到 C1OUT 的输出； 001: 设置 C1ORAW 为高：TMRx_CVAL=TMRx_C1DT 时。 010: 设置 C1ORAW 为低：TMRx_CVAL=TMRx_C1DT 时。 011: 切换 C1ORAW 的电平：当 TMRx_CVAL=TMRx_C1DT 时。</p> <p>100: 固定 C1ORAW 为低。 101: 固定 C1ORAW 为高。 110: PWM 模式 A - OWCDIR=0, 若 TMRx_C1DT&gt;TMRx_CVAL 时设置 C1ORAW 为高，否则为低； - OWCDIR=1, 若 TMRx_C1DT &lt; TMRx_CVAL 时设置 C1ORAW 为低，否则为高。 111: PWM 模式 B - OWCDIR=0, 若 TMRx_C1DT &gt; TMRx_CVAL 时设置 C1ORAW 为低，否则为高；</p>

					-OWCDIR=1, 若 TMRx_C1DT < TMRx_CVAL 时设置 C1ORAW 为高, 否则为低。 注: 除'000'外, 其余配置下 C1OUT 将连接到 C1ORAW, C1OUT 的输出电平除了会根据 C1ORAW 变化外, 还与 CCTRL 所配置的输出极性有关。
位 3	C1OBEN	0x0	rw		通道 1 输出缓存使能 (Channel 1 output buffer enable) 0: 关闭 TMRx_C1DT 的缓存功能, 写入 TMRx_C1DT 的内容会立即生效。 1: 启用 TMRx_C1DT 的缓存功能, 写入 TMRx_C1DT 的内容将保存到缓存寄存器中, 当发生溢出事件时再更新到 TMRx_C1DT 中。
位 2	C1OIEN	0x0	rw		通道 1 输出立即使能 (Channel 1 output immediately enable) 在 PWM 模式 A 或模式 B 下, 该位能够缩短触发事件到通道 1 的输出响应间的时间。 0: 需要比较 CVAL 与 C1DT 的值之后再产生输出。 1: 无需比较 CVAL 与 C1DT 的值, 当发生触发事件时立即产生输出。
位 1: 0	C1C	0x0	rw		通道 1 配置 (Channel 1 configure) 当 C1EN=0'时, 这些位用于选择通道 1 为输出或输入, 以及输入时的映射选择: 00: 输出; 01: 输入, C1IN 映射在 C1IRAW 上; 10: 输入, C1IN 映射在 C2IRAW 上; 11: 输入, C1IN 映射在 STCI 上, 只有在 STIS 选择内部触发输入时才工作。

**输入模式:**

域	简称	复位值	类型	功能
位 15: 8	保留	0x00	resd	保持默认值
位 7: 4	C1DF	0x0	rw	通道 1 滤波器 (Channel 1 digital filter) 这些位用于配置通道 1 的滤波器。滤波的个数为 N, 则表示发生了 N 次采样事件后输入边沿才能通过滤波器: 0000: 无滤波器, 以 $f_{DTS}$ 采样 1000: 采样频率 $f_{SAMPLING} = f_{DTS}/8$ , N=6 0001: 采样频率 $f_{SAMPLING} = f_{CK\_INT}$ , N=2 1001: 采样频率 $f_{SAMPLING} = f_{DTS}/8$ , N=8 0010: 采样频率 $f_{SAMPLING} = f_{CK\_INT}$ , N=4 1010: 采样频率 $f_{SAMPLING} = f_{DTS}/16$ , N=5 0011: 采样频率 $f_{SAMPLING} = f_{CK\_INT}$ , N=8 1011: 采样频率 $f_{SAMPLING} = f_{DTS}/16$ , N=6 0100: 采样频率 $f_{SAMPLING} = f_{DTS}/2$ , N=6 1100: 采样频率 $f_{SAMPLING} = f_{DTS}/16$ , N=8 0101: 采样频率 $f_{SAMPLING} = f_{DTS}/2$ , N=8 1101: 采样频率 $f_{SAMPLING} = f_{DTS}/32$ , N=5 0110: 采样频率 $f_{SAMPLING} = f_{DTS}/4$ , N=6 1110: 采样频率 $f_{SAMPLING} = f_{DTS}/32$ , N=6 0111: 采样频率 $f_{SAMPLING} = f_{DTS}/4$ , N=8 1111: 采样频率 $f_{SAMPLING} = f_{DTS}/32$ , N=8
位 3: 2	C1IDIV	0x0	rw	通道 1 分频系数 (Channel 1 input divider) 这些位定义了通道 1 的分频系数。 00: 不分频, 每一个有效的边沿都会产生一次输入; 01: 每 2 个有效的边沿产生一次输入; 10: 每 4 个有效的边沿产生一次输入; 11: 每 8 个有效的边沿产生一次输入。 注: C1EN=0'时, 分频系数复位。
位 1: 0	C1C	0x0	rw	通道 1 配置 (Channel 1 configure) 当 C1EN=0'时, 这些位用于选择通道 1 为输出或输入, 以及输入时的映射选择: 00: 输出;

01: 输入, C1IN 映射在 C1IRAW 上;  
 10: 输入, C1IN 映射在 C2IRAW 上;  
 11: 输入, C1IN 映射在 STCI 上, 只有在 STIS 选择内部触发输入时才工作。

#### 14.2.5.6 通道控制寄存器 (TMRx\_CCTRL)

域	简称	复位值	类型	功能
位 15: 4	保留	0x0	resd	保持默认值。
位 3	C1CP	0x0	rw	通道 1 互补极性 (Channel 1 complementary polarity) 定义输入信号的有效沿, 详见 C1P 位描述。
位 2	保留	0x0	resd	保持默认值。
位 1	C1P	0x0	rw	通道 1 极性 (Channel 1 polarity) 通道 1 配置为输出: 0: C1OUT 的有效电平为高 1: C1OUT 的有效电平为低 通道 1 配置为输入: C1CP/C1P 位共同定义输入信号有效沿。 00: C1IN 的有效边沿为上升沿; 作为外部触发使用时, C1IN 不反相。 01: C1IN 的有效边沿为下降沿; 作为外部触发使用时, C1IN 反相。 10: 保留 11: C1IN 的有效边沿为上升沿和下降沿; 作为外部触发使用时, C1IN 不反相。
位 0	C1EN	0x0	rw	通道 1 使能 (Channel 1 enable) 0: 禁止输入或输出; 1: 使能输入或输出。

表 14-10 标准CxOUT通道的输出控制位

CxEN 位	CxOUT 输出状态
0	禁止输出 (CxOUT=0)
1	CxOUT = CxORAW + 极性

注意: 连接到标准 CxOUT 通道的外部 I/O 管脚状态, 取决于 CxOUT 通道状态和 GPIO 以及 IOMUX 寄存器。

#### 14.2.5.7 计数值 (TMRx\_CVAL)

域	简称	复位值	类型	功能
位 15: 0	CVAL	0x0000	rw	计数值 (Counter value)

#### 14.2.5.8 预分频器 (TMRx\_DIV)

域	简称	复位值	类型	功能
位 15: 0	DIV	0x0000	rw	分频系数 (Divider value) 计数器时钟频率 $f_{CK\_CNT} = f_{TMR\_CLK} / (DIV[15:0]+1)$ 溢出事件发生时该寄存器值被传送到实际的预分频寄存器中。

#### 14.2.5.9 周期寄存器 (TMRx\_PR)

域	简称	复位值	类型	功能
位 15: 0	PR	0x0000	rw	周期值 (Period value) 定时器计数的周期值。当周期值为 0 时, 定时器不工作。

#### 14.2.5.10 通道1数据寄存器 (TMRx\_C1DT)

域	简称	复位值	类型	功能
位 15: 0	C1DT	0x0000	rw	通道 1 数据寄存器值 (Channel 1 data register) 若通道 1 配置为输入:

---

C1DT 是前一次通道 1 输入事件(C1IN)所保存的 CVAL。  
若通道 1 配置为输出：  
C1DT 是将要和 CVAL 进行比较的值，写入的值是否会立即生效取决于输出缓存使能位 (C1OBEN)，并根据设置在 C1OUT 上产生相应的输出。

---

## 14.3 高级控制定时器 (TMR1)

### 14.3.1 TMR1简介

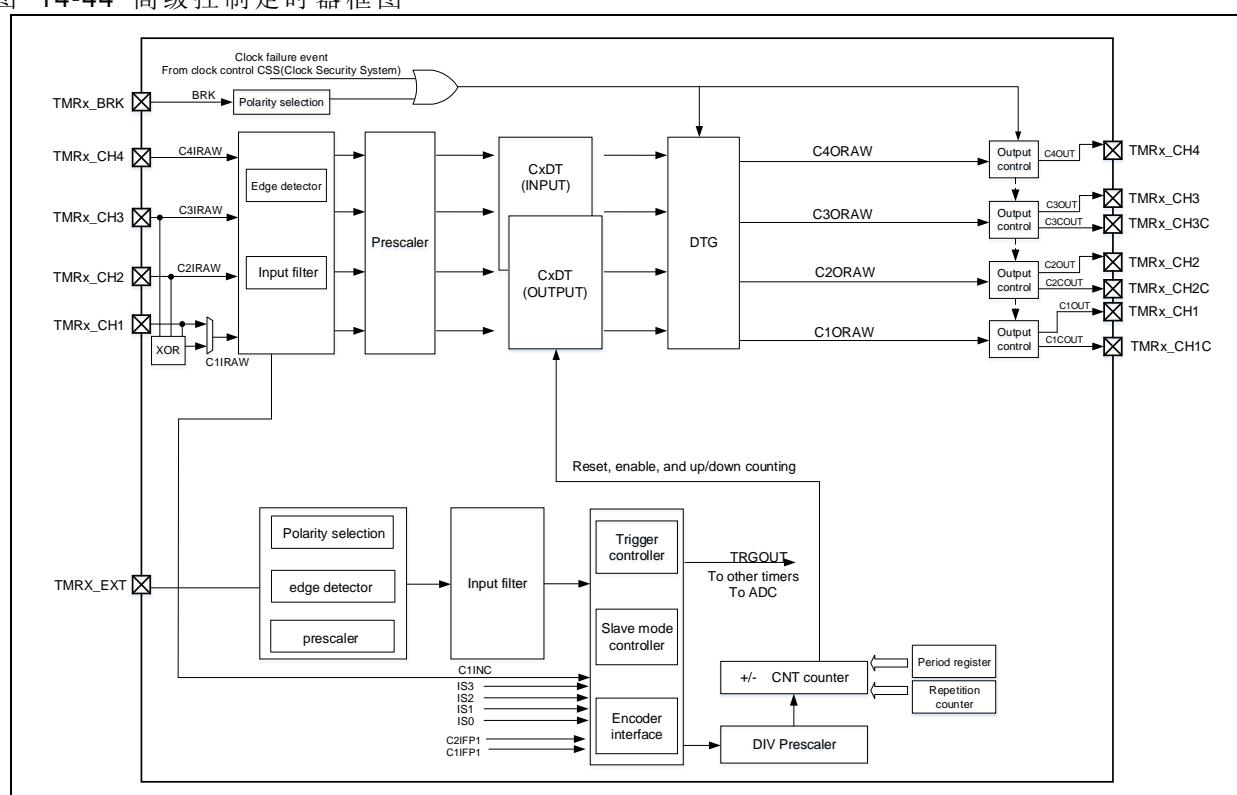
高级定时器 TMR1 包含一个支持向上、向下、中央双向对齐计数的 16 位计数器、4 个通道寄存器、4 组独立的通道。可实现嵌入死区、输入捕获、可编程 PWM 输出。

### 14.3.2 TMR1主要特性

TMR1 定时器的功能包括：

- 可选内部、外部、内部触发输入用作计数时钟
- 16 位支持向上、向下、双向、重复计数、编码器模式的计数器
- 4 组独立通道，支持输入捕获、输出比较、PWM 生成、单周期模式、死区插入。
- 3 组支持互补输出的独立通道
- 支持 TMR 刹车功能
- 定时器之间可互联同步
- 支持溢出事件、触发事件、刹车输入、通道事件触发中断/DMA
- 支持 TMR burst DMA 传输

图 14-44 高级控制定时器框图



### 14.3.3 TMR1功能描述

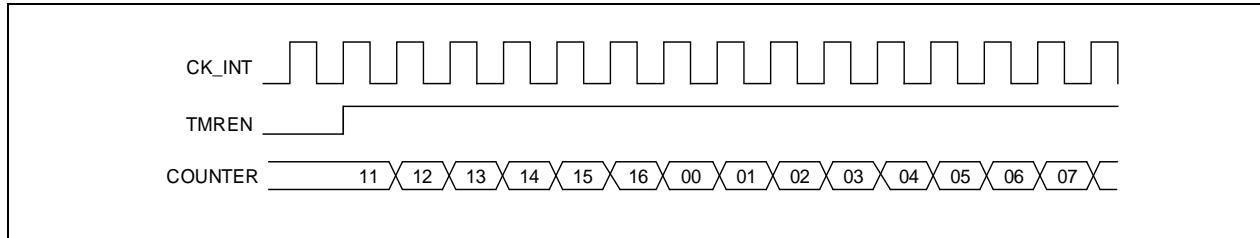
#### 14.3.3.1 计数时钟

TMR1 计数时钟可从内部时钟 (CK\_INT)、外部时钟 (外部时钟模式 A、B)、内部触发输入 (ISx) 这些时钟源提供。

##### 内部时钟 (CK\_INT)

默认下使用 CK\_INT 经由预分频器驱动计数器计数。

图 14-45 使用 CK\_INT 且分频系数为 1



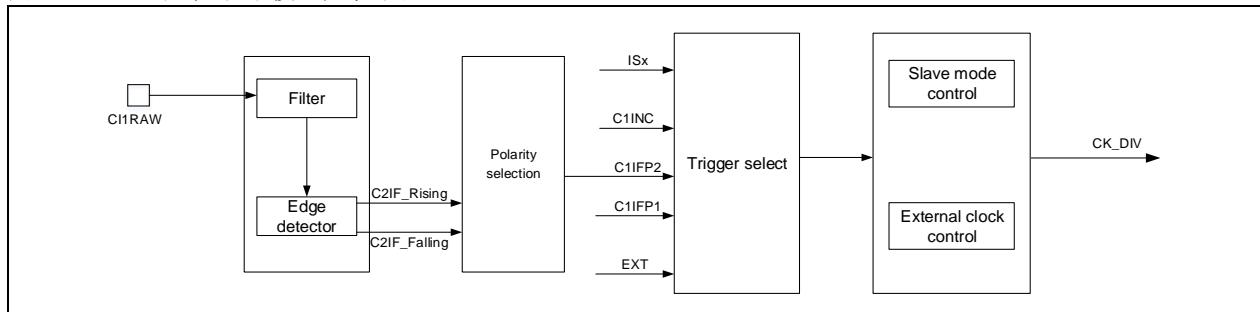
### 外部时钟 (TRGIN/EXT)

计数时钟可由两种外部时钟源提供，分别为 TRGIN 和 EXT 信号。

当 SMSEL=3'111 时，外部时钟模式 A 被选中，配置 STIS[2: 0]来选择 TRGIN 信号驱动计数器计数。

当 ECMBEN=1 时，外部时钟模式 B 被选中，计数器由 EXT 信号驱动计数。

图 14-46 外部时钟模式 A 框图



注：由于同步逻辑，输入端信号与计数器实际时钟之间存在一定延时。

图 14-47 使用外部时钟模式 A 计数

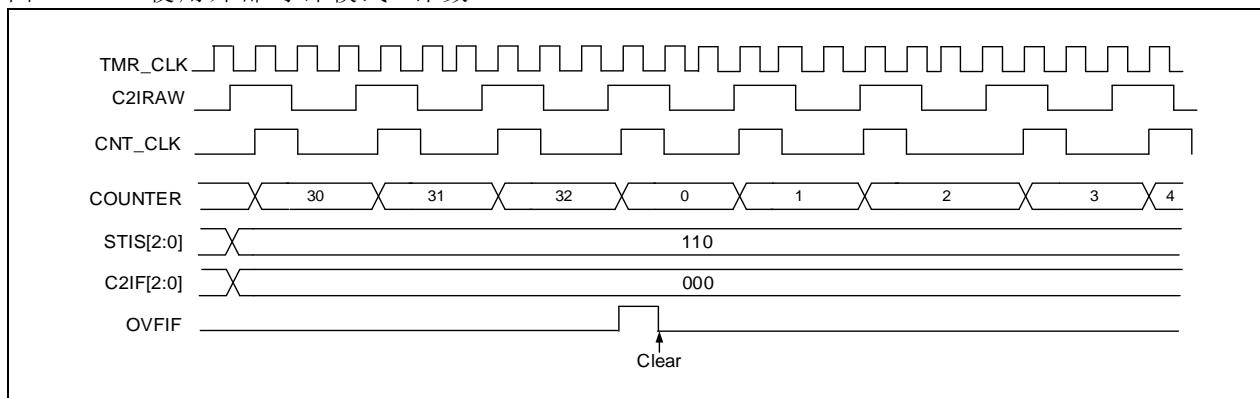
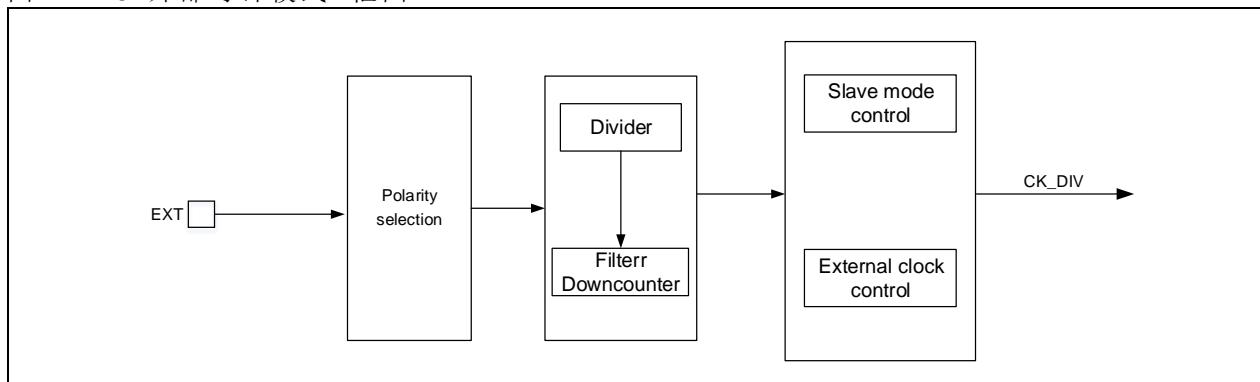
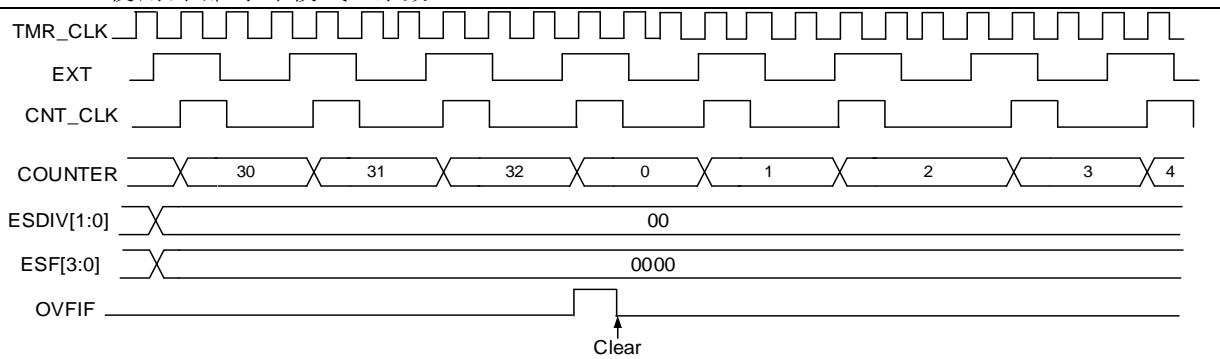


图 14-48 外部时钟模式 B 框图



注：由于同步逻辑。输入端 EXT 信号与计数器实际时钟之间存在一定延时。

图 14-49 使用外部时钟模式B计数



### 内部触发输入 (ISx)

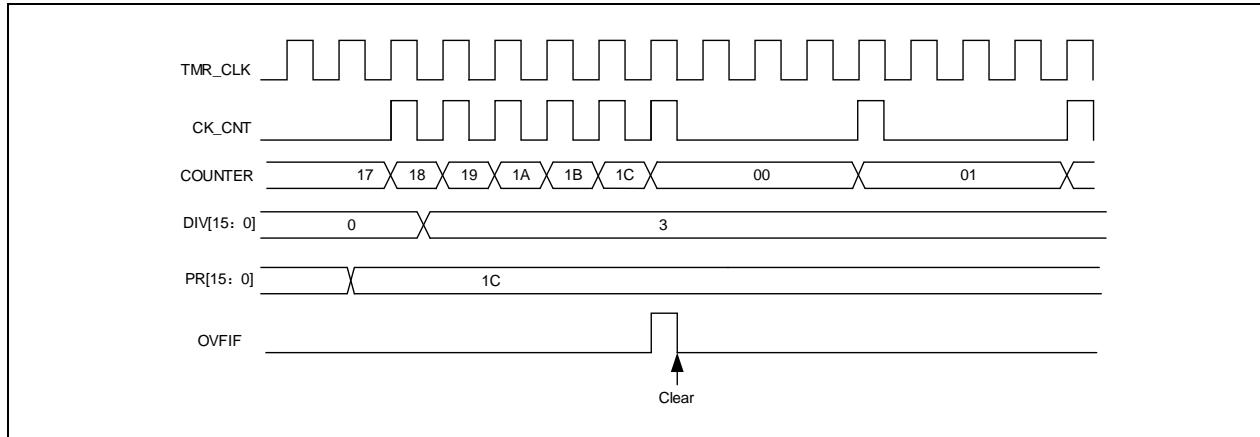
定时器之间支持互联同步，因此一个定时器的 TMR\_CLK 可由另一个定时器输出信号 TRGOUT 提供。配置 STIS[2: 0]选择内部触发信号驱动计数器计数。

高级定时器内含一个 16 位预分频器，用于产生驱动计数器计数的时钟 CK\_CNT，通过配置 TMRx\_DIV 寄存器值，可灵活调整 CK\_CNT 与 TMR\_CLK 之间的分频关系。预分频值可在任何时刻修改，但只在下一个溢出事件发生时，新值才会生效。

表 14-11 TMRx 内部触发连接

次定时器	IS0 (STIS=000)	IS1 (STIS=001)	IS2 (STIS=010)	IS3 (STIS=011)
TMR1	TMR5	TMR2	-	TMR4

图 14-50 当预分频器的参数从 1 变到 4 时，计数器的时序图



### 14.3.3.2 计数模式

高级定时器提供了多种计数模式，用来满足不同的应用场景。其内部拥有一个支持 16 位向上计、向下、中央双向对齐计数的计数器，计数器的值可由周期寄存器 (TMRx\_PR) 载入。默认下，周期寄存器 (TMRx\_PR) 值会立即传入它的影子寄存器；当开启周期缓冲功能后 (PRBEN 置 1)，周期寄存器 (TMRx\_PR) 值会在溢出事件发生时传入它的影子寄存器。溢出事件由 OVREN、OVFS 位配置。

将 TMREN 位置 1 可使能定时器计数，由于同步逻辑，实际驱动计数器的使能信号 TMR\_EN 相对于 TMREN 延迟一个时钟周期。

#### 向上计数模式

向上计数模式中，当计数值达到 TMRx\_PR 值时，重新从 0 向上计数，计数器上溢并产生溢出事件，同时 OVFIF 位置 1。若禁止产生溢出事件，则计数器溢出后不再重载预分频值和周期值，否则预分频值和周期值将在溢出事件后更新。

图 14-51 PRBEN=0时的溢出事件

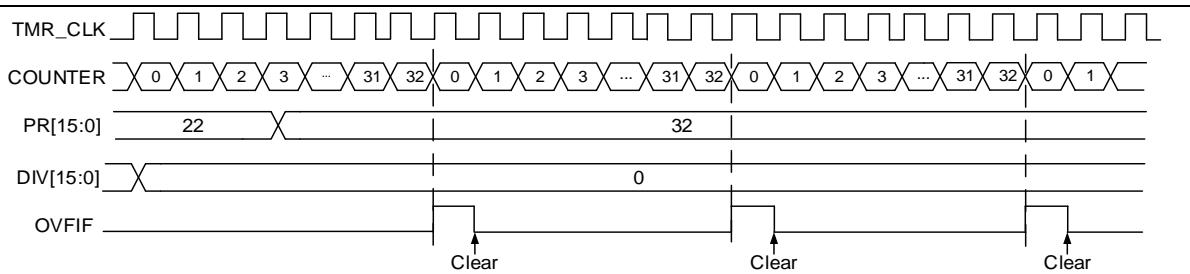
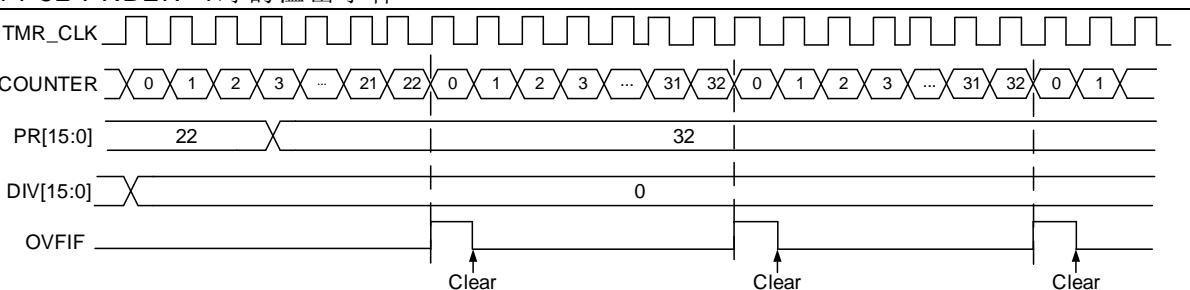


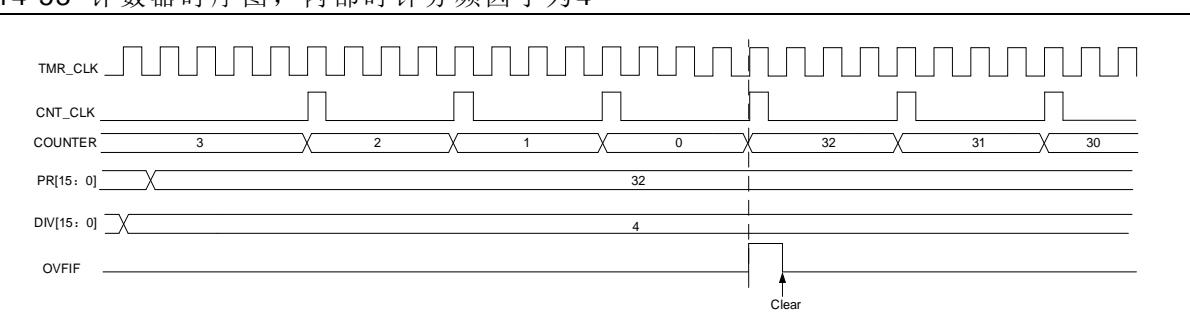
图 14-52 PRBEN=1时的溢出事件



### 向下计数模式

向下计数模式中，当计数值达到 0 值时，重新从 **TMRx\_PR** 向上下数，计数器下溢并产生溢出事件。

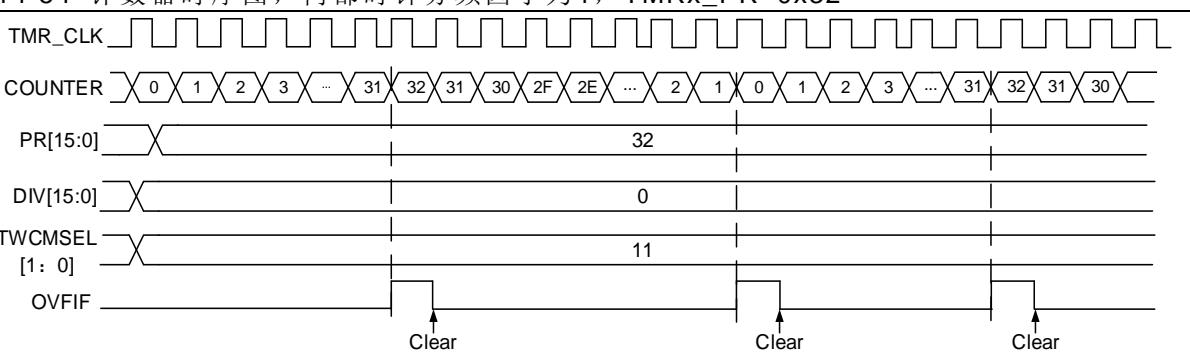
图 14-53 计数器时序图，内部时钟分频因子为4



### 中央双向对齐计数模式

中央双向对齐计数模式中，计数器交替向上、向下计数。当计数值从 **TMRx\_PR** 值向下计数到 1 值时，产生下溢事件，然后从 0 开始向上计数；当向上计数到 **TMRx\_PR** 值-1 时，产生上溢事件，之后从 **TMRx\_PR** 值向下计数。计数器计数方向可由计数器方向控制位（**OWCDIR**）实时查看。

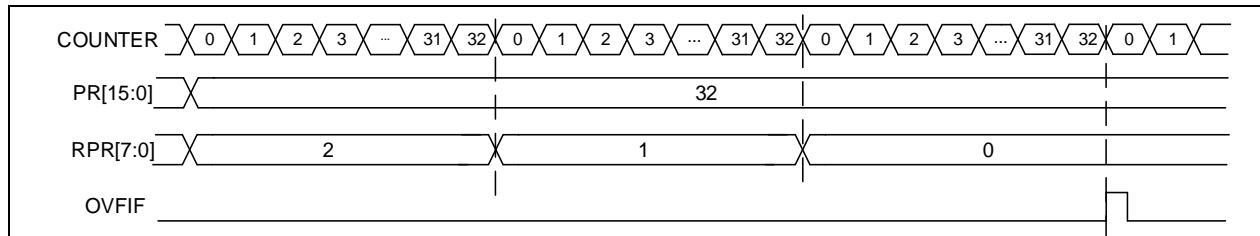
注意： 中央双向对齐计数模式下，**OWCDIR** 位为只读位。

图 14-54 计数器时序图，内部时钟分频因子为1，**TMRx\_PR**=0x32

### 重复计数模式：

重复计数器为非 0 值时，将开启重复计数功能。此时每次计数器溢出，重复计数器递减，当重复计数器达到 0 时，将产生溢出事件。通过配置不同重复计数器值，可调整溢出事件产生的频率。

图 14-55 RPR=2时的OVFIF

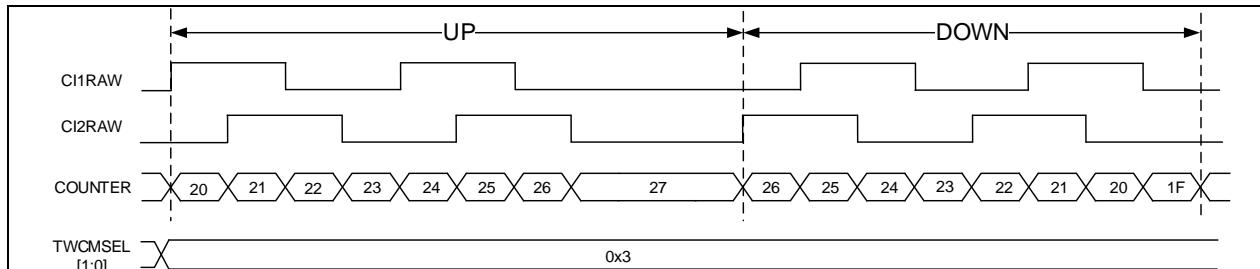
**编码器模式**

将SMSEL[2:0]配置为3'b001/3'b010/3'b011可开启编码模式，编码模式下需提供两个输入(C1IN/C2IN)，根据一个输入的电平值，计数器将在另一个输入的边沿向上或向下计数。计数方向将由OWCDIR值指示。编码模式下计数器计数方向如下表所示：

表 14-12 计数方向与编码器信号的关系

有效边沿	相对信号的电平 (C1INFP1 对应 C2IN, C2INFP2 对应 C1IN)	C1INFP1 信号		C2INFP2 信号	
		上升	下降	上升	下降
仅在 C1IN 计数	高	向下计数	向上计数	不计数	不计数
	低	向上计数	向下计数	不计数	不计数
仅在 C2IN 计数	高	不计数	不计数	向上计数	向下计数
	低	不计数	不计数	向下计数	向上计数
在 C1IN 和 C2IN 上计数	高	向下计数	向上计数	向上计数	向下计数
	低	向上计数	向下计数	向下计数	向上计数

图 14-56 编码模式计数实例（编码器模式C）

**14.3.3.3 TMR输入部分**

TMR1 拥有 4 个独立通道，每个通道可配置为输入或输出，当配置位输入时，可用于对输入信号的滤波、选择、分频和输入捕获功能。

图 14-57 输入/输出通道 1 的主电路

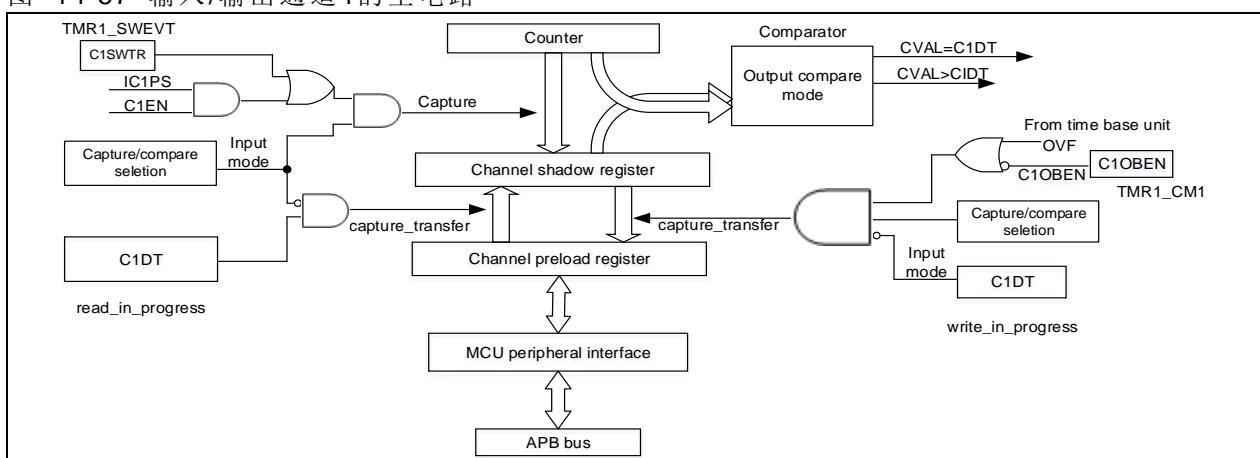
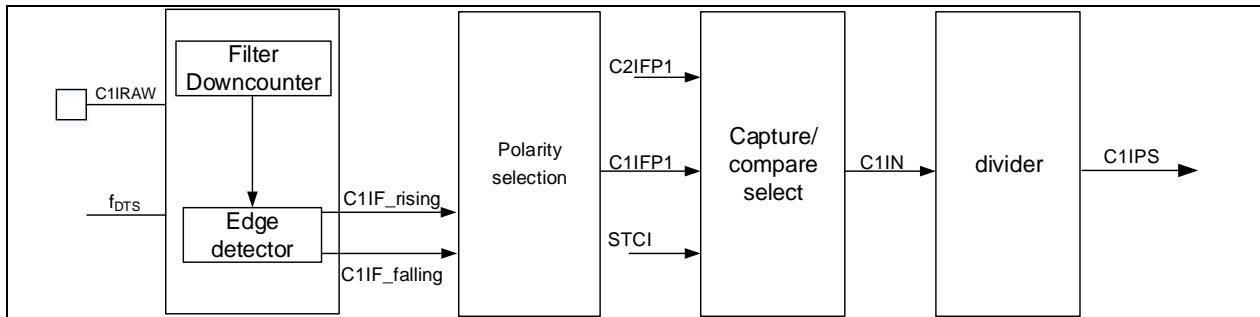


图 14-58 通道1输入部分



### 输入模式

此模式下，当选中的触发信号被检测到时，通道寄存器（TMRx\_CxDT）会记录当前计数器计数值，并将捕获比较中断标志位（CxIF）置 1，若已使能通道中断（CxIEN）、通道 DMA 请求（CxDEN）则产生相应的中断和 DMA 请求。若在 CxIF 已置 1 后检测到选中的触发信号，则将 CxRF 位置 1。

以若要捕获 C1IN 输入的上升沿，可按如下进行配置：

- 将 TMRx\_通道寄存器（TMRx\_CxDT）中的 C1C 位配置为 01，选择 C1IN 作为通道 1 输入。
- 配置 C1IN 信号滤波器带宽（CxDF[3: 0]）。
- 配置 C1IN 通道的有效沿，在通道控制寄存器（TMRx\_CCTRL）中写入 C1P=0（上升沿）。
- 配置 C1IN 信号捕获频率（C1DIV[1: 0]）。
- 使能通道 1 输入捕获（C1EN=1）。
- 根据需要设置 DMA/中断使能寄存器（TMRx\_IDEN）中的 C1IEN 为、DMA/中断使能寄存器（TMRx\_IDEN）中的 C1DEN 位，选择中断请求或 DMA 请求。

### 多输入异或

T 通道 1 的输入端可选择 TMRx\_CH1、TMRx\_CH2 和 TMRx\_CH3 经异或逻辑后输入。将控制寄存器 2（TMRx\_CTRL2）中的 C1INSEL 位置 1 可开启此功能。

多输入异或功能可用于连接霍尔传感器，例如，将异或输入的三个输入端分别连接到三个霍尔传感器，通过分析三路霍尔传感器信号可计算出转子的位置和速度。

### 14.3.3.4 TMR输出部分

TMR 的输出部分由比较器和输出控制构成，用于编程输出信号的周期、占空比、极性。高级定时器的输出部分在不同通道上有所不同，如下图所示：

图 14-59 通道1至3输出部分

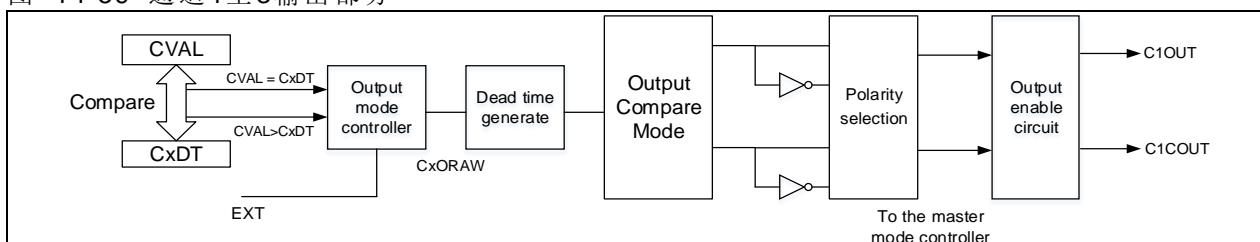
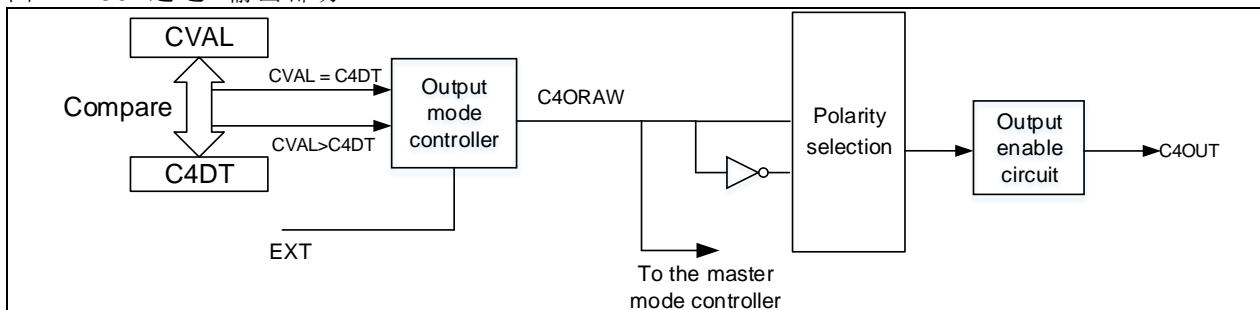


图 14-60 通道4输出部分



## 输出模式

配置  $CxC[2: 0] \neq 2'b00$  将通道配置为输出可实现多种输出模式，此时，计数器计数值将与通道寄存器（ $TMRx_CxDT$ ）值比较，并根据  $CxOCTRL[2: 0]$  位配置的输出模式，产生中间信号  $CxORAW$ ，再经过输出控制逻辑处理后输送到 IO。输出信号的周期由周期寄存器（ $TMRx_PR$ ）值配置，占空比则由通道寄存器（ $TMRx_CxDT$ ）值配置。

输出比较模式有以下子类：

- **PWM 模式：**  $CxOCTRL=2'b110/111$  时，开启 PWM 模式，每个通道可独立配置并输出一路 PWM 信号。此时，输出信号的周期由  $TMRx_PR$  配置，占空比由  $CxDT$  值配置，计数器值与通道寄存器（ $TMRx_CxDT$ ）值进行比较，根据计数方向输出指定电平信号，关于 PWM 模式 A/B 详见  $CxOCTRL[2: 0]$  位描述。当计数模式为中央双向对齐计数时，可根据  $OWCDIR$  位指示计数方向。
- **强制输出模式：**  $CxOCTRL=2'b100/101$  时，开启强制输出模式。此时， $CxORAW$  信号的电平被强制输出为配置的电平，而与计数值无关。虽然输出信号不依赖于比较结果，但通道标志位和 DMA 请求仍依赖于比较结果。
- **输出比较模式：**  $CxOCTRL=2'b001/010/011$  时，开启输出比较模式。此时，当计数值与  $CxDT$  值匹配时， $CxORAW$  被强制为高电平、低电平或进行电平翻转。
- **单周期模式：** PWM 模式的特例，将  $OCMEN$  位置 1 可开启单周期模式，此模式下，仅在当前计数周期中进行比较匹配，完成当前计数后， $TMREN$  位清 0，因此仅输出一个脉冲。当配置为向上计数模式时，需要严格配置  $CVAL < CxDT \leq PR$ ；向下计数时，需严格配置  $CVAL > CxDT$ 。
- **快速输出模式：** 将  $CxOIEN$  位置 1 可开启此功能，开启后  $CxORAW$  电平值不再在计数值与  $CxDT$  匹配时变化，而是在当前计数周期开始时，也就是说，比较结果被提前了，计数器值与通道寄存器（ $TMRx_CxDT$ ）的比较结果将会提前决定  $CxORAW$  的电平。

图 14-61 展示了输出比较模式（翻转）的例子， $C1DT=0x3$ ，当计数值等于 0x3 时，输出电平  $C1OUT$  被翻转。

图 14-62 展示了计数器向上计数与 PWM 模式 A 配合的例子， $PR=0x32$ ， $CxDT$  配置为不同的值时输出时输出信号的翻转情况。

图 14-63 展示了计数器中央双向对齐计数与 PWM 模式 A 配合的例子， $PR=0X32$ ， $CxDT$  配置为不同的值时输出时输出信号的翻转情况。

图 14-64 展示了计数器向上计数与单周期模式下 PWM 模式 B 配合的例子，计数器仅计数了一个周期，输出信号在这个周期中只输出了一个脉冲。

图 14-61 计数值与  $C1DT$  值匹配时翻转  $C1ORAW$

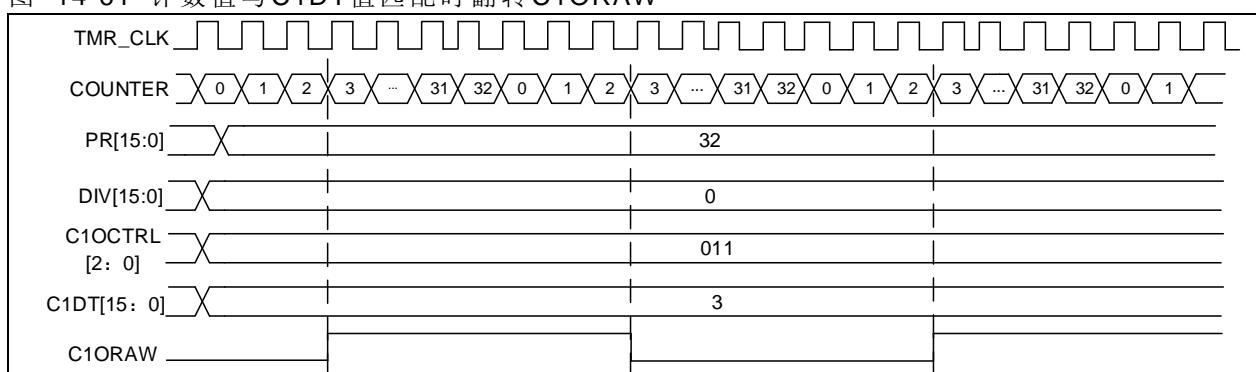


图 14-62 向上计数下 PWM 模式 A

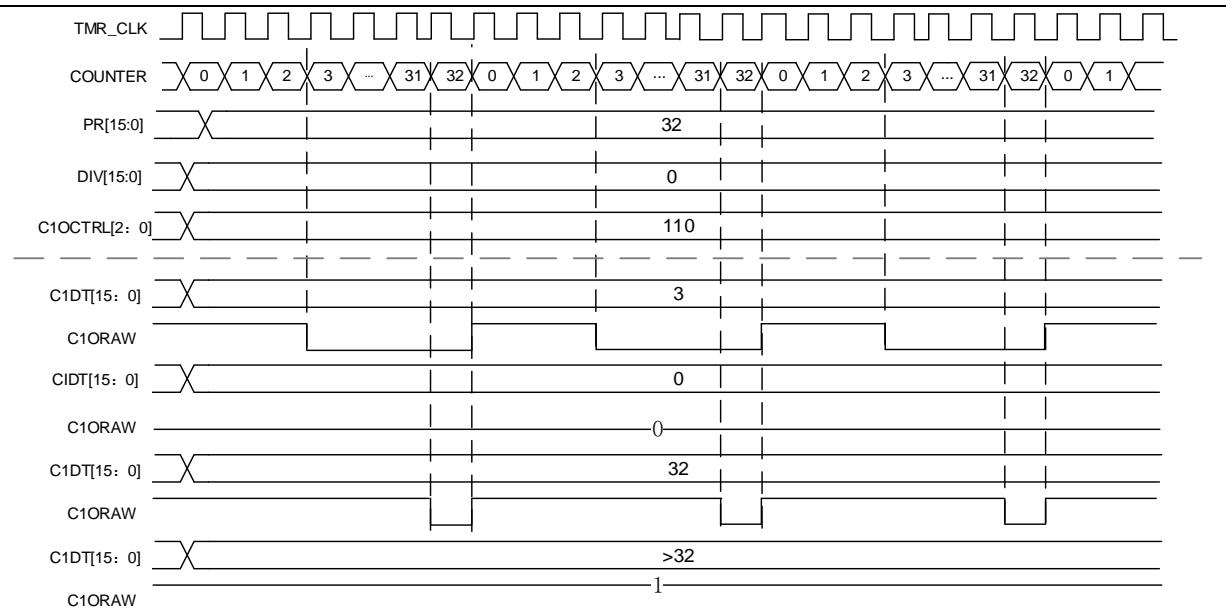


图 14-63 中央双向对齐计数下 PWM 模式

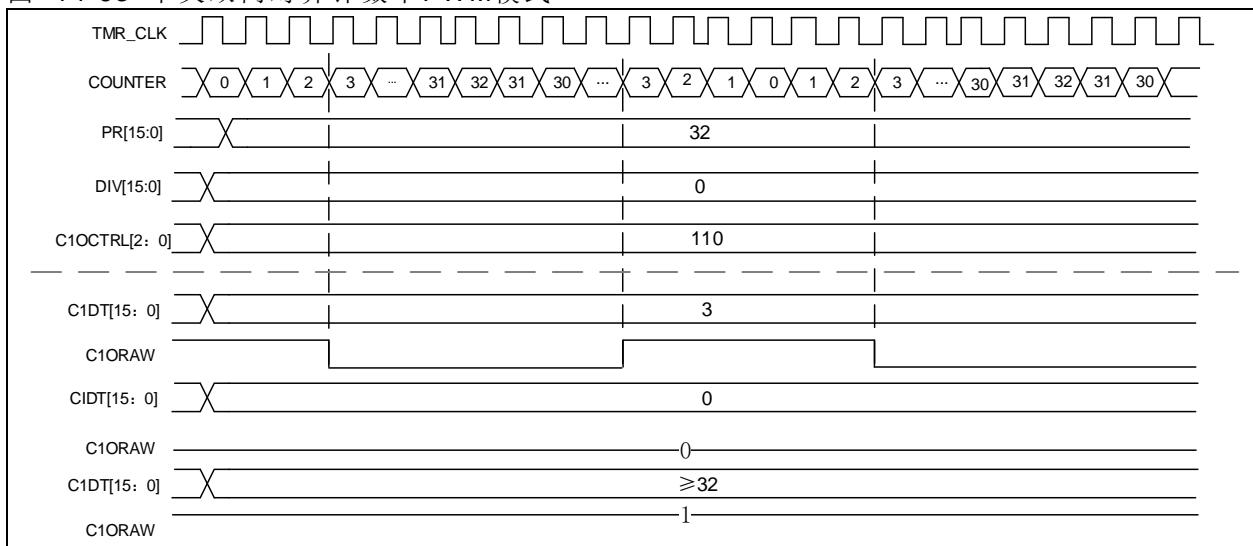
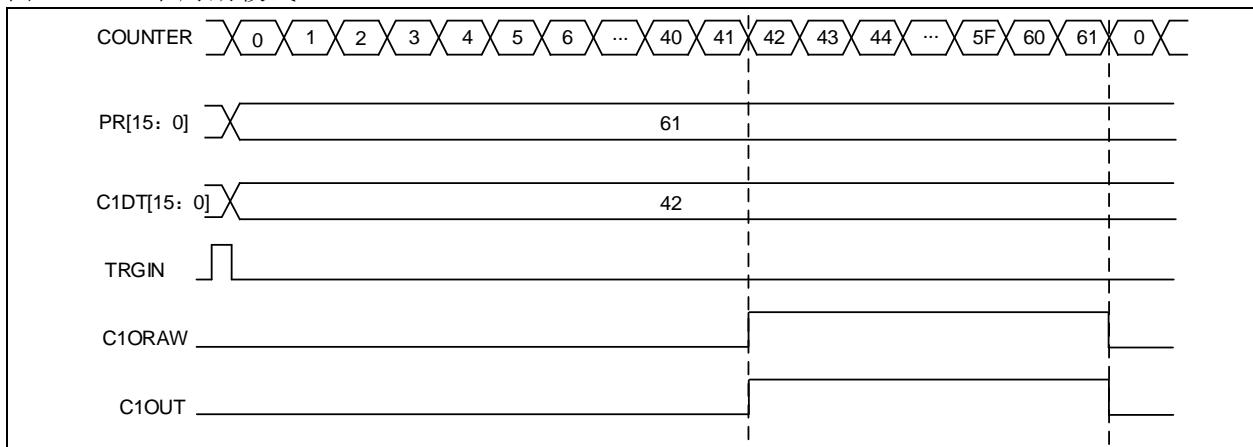


图 14-64 单周期模式

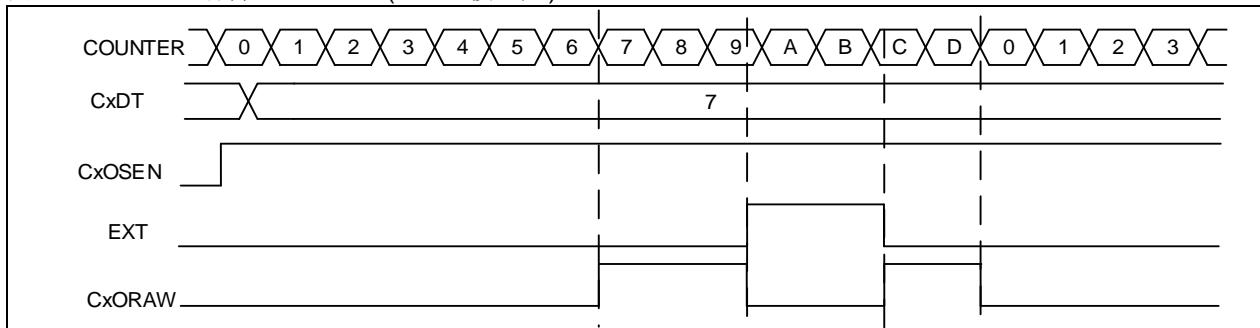


### CxORAW 信号清除

对将 CxOSEN 位置 1 后，指定通道的 CxORAW 信号由 EXT 高电平清 0，在下一次溢出事件发生前 CxORAW 信号无法被改变。

制模式时，CxORAW 信号清除功能不可用，只有在输出比较模式或 PWM 模式，此功能有效。该功能只能用于输出比较和。下图显示了使用 EXT 信号清除 CxORAW 的例子，当 EXT 为高电平期间，原本为高电平的 CxORAW 信号被拉低，当 EXT 为低电平时，CxORAW 根据计数值和 CxDT 比较结果输出电平。

图 14-65 EXT 清除 CxORAW(PWM 模式 A)



### 死区插入

高级定时器通道 1 至 3 包含一组反向通道输出，通过 CxCEN 使能，通过 CxCP 配置极性。CxOUT 和 CxCOUT 的输出状态见表 14-14。

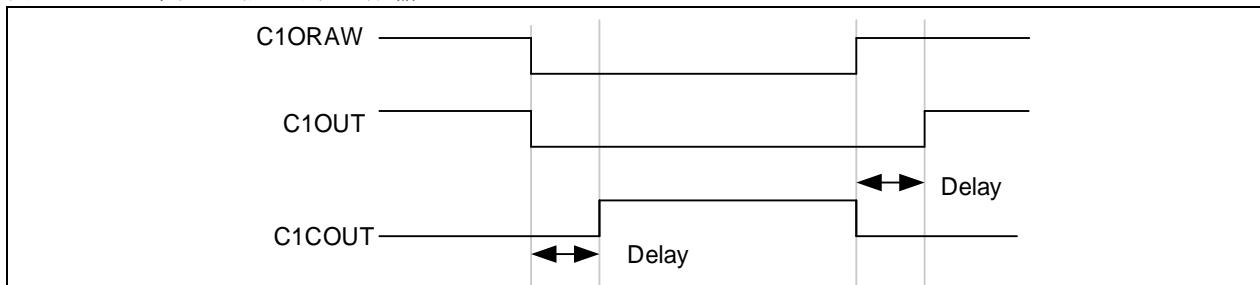
当转换为 IDLEF 状态，即 OEN 下降到 0，死区被激活。

将 CxEN 和 CxCEN 位置 1 后，通过配置 DTC[7: 0]死区发生器，可插入不同时长的死区。插入死区后，CxOUT 的上升沿延迟于参考信号的上升沿；CxCOUT 的上升沿延迟于参考信号的下降沿。

如果延迟大于当前有效的输出宽度如果 C1OUT 和 C1COUT 要产生相应的脉冲，死区时间应小于有效的输出宽度。

下列图显示了 CxP=0、CxCP=0、OEN=1、CxEN=1 并且 CxCEN=1 时死区插入的例子

图 14-66 带死区插入的互补输出



### 14.3.3.5 TMR刹车功能

开启刹车功能后 (BRKEN 位置 1)，CxOUT 和 CxCOUT 由 OEN、FCSODIS、FCSOEN、CxIOS 和 CxCIOS 共同控制。但 CxOUT 和 CxCOUT 输出总是不能同时处于有效电平上的。详见表 14-14 带刹车功能的互补输出通道 CxOUT 和 CxCOUT 的控制位。

刹车信号来源可以是刹车输入管脚、时钟失效事件，刹车输入信号的极性由 BRKV 位控制。

当发生刹车事件时，有下述动作：

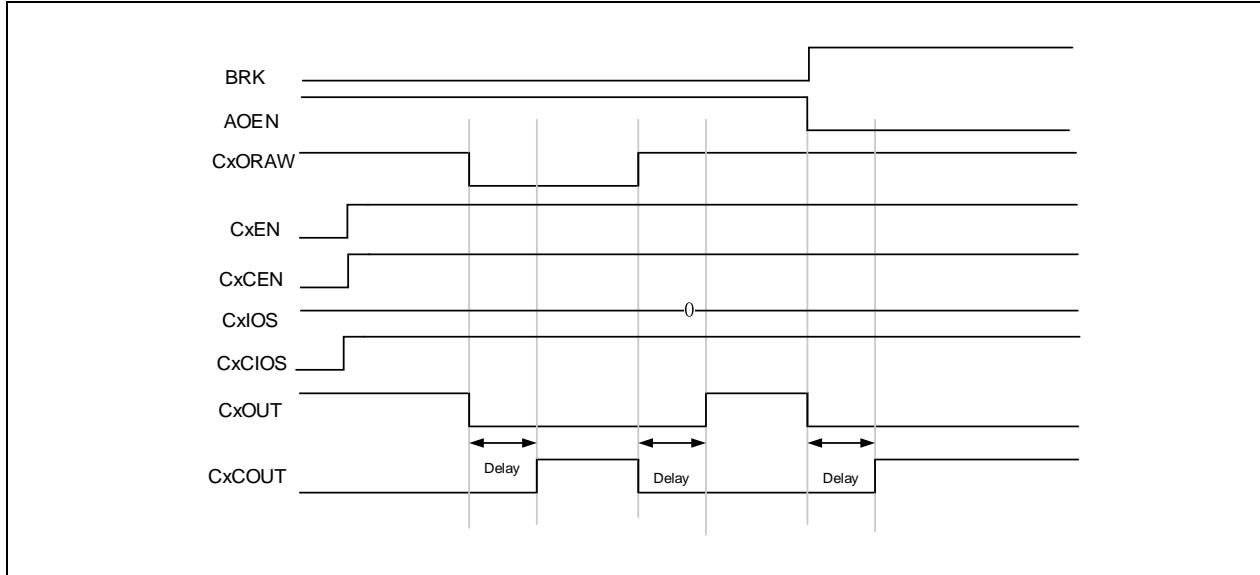
- OEN 位异步清零，通道输出状态由 FCSODIS 位选择。关闭 MCU 的振荡器不影响该功能。
  - OEN 被清零后，通道输出电平由 CxIOS 位设定。如果 FCSODIS=0，则定时器输出使能被禁止，否则输出使能始终为高。
  - 当使用互补输出时：
    - 输出最开始处于复位状态，也就是无效的状态（取决于极性）。这是异步操作，定时器有无时钟并不影响此功能。
    - 定时器的时钟如果有效，会开启死区生成功能，CxIOS 和 CxCIOS 位用来配置死区之后的电平。即使在这种情况下，CxOUT 和 CxCOUT 也不能被同时驱动到有效的电平。
- 注意，由于 OEN 位同步逻辑，死区时间较通常情况会延长一段时间（大约 2 个

*clk\_tmr* 的时钟周期)。

- 如果 FCSODIS=0，定时器释放使能输出，否则保持使能输出；或一旦 CxEN 与 CxCEN 之一变高时，使能输出变为高。
- 如果开启了刹车中断或 DMA 功能，刹车状态标志将置 1，并产生刹车中断或 DMA 请求。
- 如果将 AOEN 位置 1，在下一个溢出事件时 OEN 位被自动置 1。

注意：刹车输入电平有效时，OEN 不能被设置，状态标志 BRKIF 也不能被清除。

图 14-67 TMR刹车功能的例子



#### 14.3.3.6 TMR同步

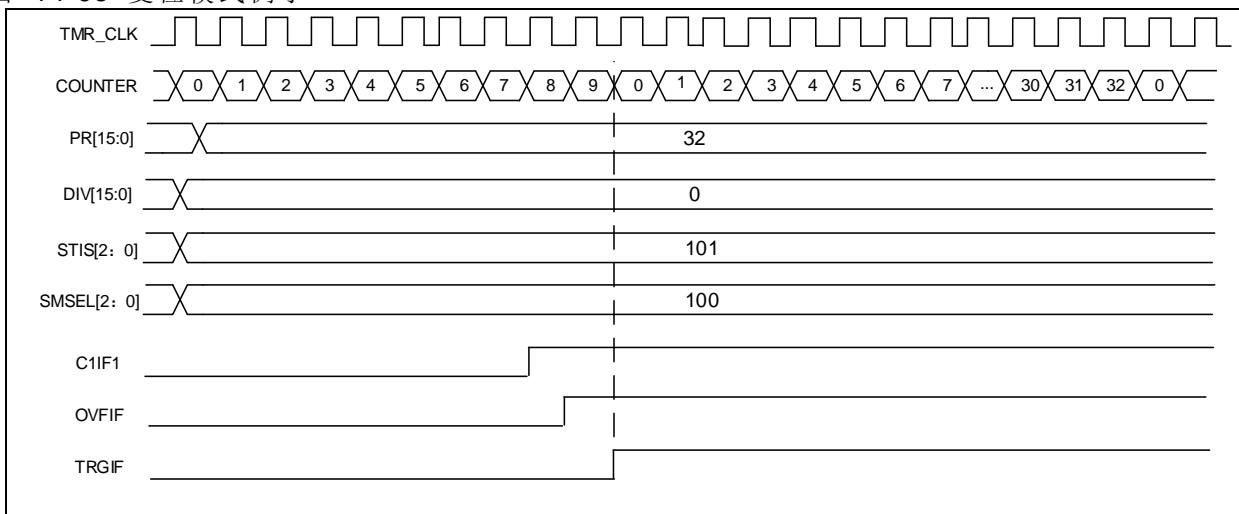
主次定时器之间可由内部连接信号进行同步。主定时器可由 PTOS[2: 0]位选择主定时器输出，即同步信息；次定时器由 SMSEL[2: 0]位选择从模式，即次定时器的工作模式。

定时器从模式有以下几种：

##### 从模式：复位模式

选中的触发信号将复位计数器和预分频器，若 OVFS 位为 0，将产生一个溢出事件。

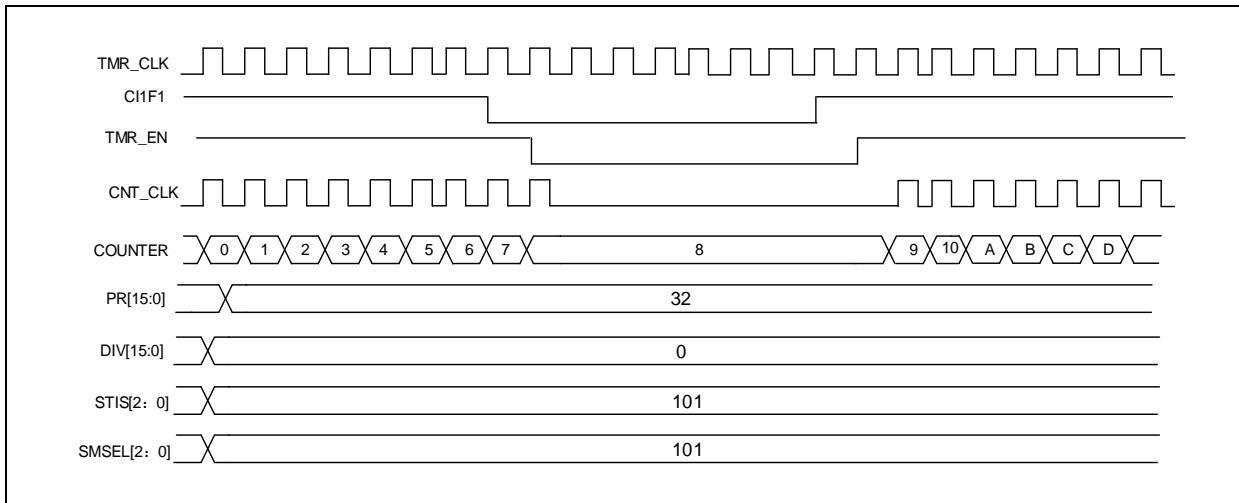
图 14-68 复位模式例子



##### 从模式：挂起模式

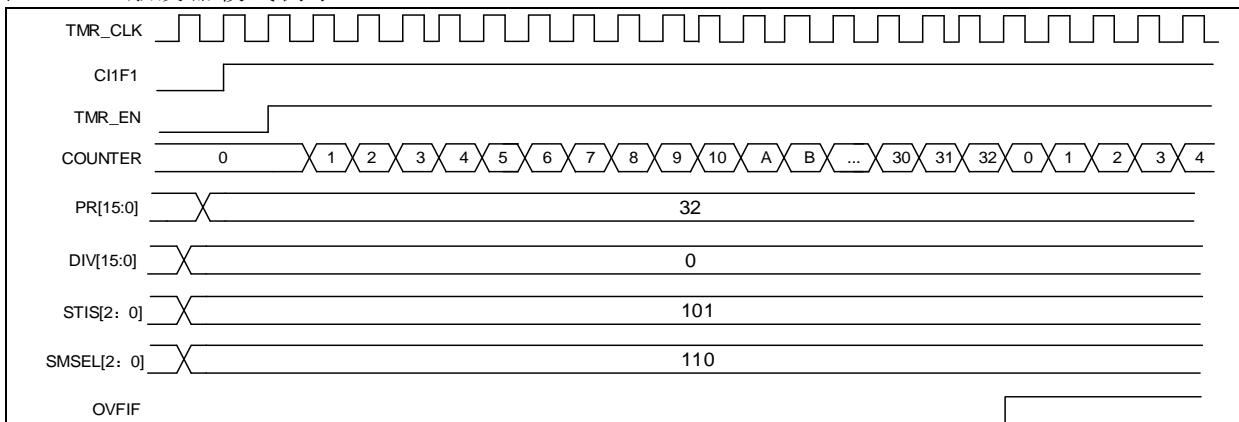
挂起模式下，计数的计数和停止受选中触发输入信号控制，当触发输入为高电平时计数器开始计数；当为低电平时，计数器暂停计数。

图 14-69 挂起模式下例子

**从模式：触发模式**

计数器将在选中的触发输入上升沿启动计数（将 TMR\_EN 置 1）。

图 14-70 触发器模式例子



定时器的同步的更多实例详见 [14.2.3.5](#) 节。

**14.3.3.7 调试模式**

当微控制器进入调试模式（Cortex™-M4 核心停止）时，将 DEBUG 模块中的 TMRx\_PAUSE 置 1，可以使 TMRx 计数器暂停计数。

**14.3.4 TMR1寄存器描述**

必须以字（32 位）的方式操作这些外设寄存器。

下表中将 TMR1 的所有寄存器映射到一个 16 位可寻址（编址）空间

表 14-13 TMR1寄存器映像和复位值

寄存器简称	基址偏移量	复位值
TMR1_CTRL1	0x00	0x0000
TMR1_CTRL2	0x04	0x0000
TMR1_STCTRL	0x08	0x0000
TMR1_IDEN	0x0C	0x0000
TMR1ISTS	0x10	0x0000
TMR1_SWEVT	0x14	0x0000
TMR1_CM1	0x18	0x0000
TMR1_CM2	0x1C	0x0000
TMR1_CCTRL	0x20	0x0000

TMR1_CVAL	0x24	0x0000
TMR1_DIV	0x28	0x0000
TMR1_PR	0x2C	0x0000
TMR1_RPR	0x30	0x0000
TMR1_C1DT	0x34	0x0000
TMR1_C2DT	0x38	0x0000
TMR1_C3DT	0x3C	0x0000
TMR1_C4DT	0x40	0x0000
TMR1_BRK	0x44	0x0000
TMR1_DMACTRL	0x48	0x0000
TMR1_DMADT	0x4C	0x0000

#### 14.3.4.1 TMR1控制寄存器1 (TMR1\_CTRL1)

域	简称	复位值	类型	功能
位 15: 10	保留	0x00	resd	保持默认值。
位 9: 8	CLKDIV	0x0	rw	时钟除频 (Clock divider) 00: 无除频; 01: 2 除频; 10: 4 除频; 11: 保留。
位 7	PRBEN	0x0	rw	周期缓冲使能 (Period buffer enable) 0: 缓冲关闭; 1: 缓冲开启。
位 6: 5	TWCMSEL	0x0	rw	中央双向对齐计数模式选择 (Two-way count mode selection) 00: 单向对齐计数模式, 方向由 OWCDIR 配置; 01: 中央双向对齐计数模式 1, 上下交替计数, 输出标志位只在计数器向下计数时被置起; 10: 中央双向对齐计数模式 2, 上下交替计数, 输出标志位只在计数器向上计数时被置起; 11: 中央双向对齐计数模式 3, 上下交替计数, 输出标志位在计数器向上和向下计数时皆被置起。
位 4	OWCDIR	0x0	rw	单向对齐计数方向 (One-way count direction) 0: 向上; 1: 向下。
位 3	OCMEN	0x0	rw	单周期使能 (One cycle mode enable) 该功能用于选择溢出事件后, 计数器是否停止。 0: 关闭; 1: 开启。
位 2	OVFS	0x0	rw	溢出事件源选择 (Overflow event source) 配置溢出事件或 DMA 请求来源。 0: 来源于计数器溢出、设置 OVFSWTR 位或次定时器控制器产生的溢出事件; 1: 只能来源于计数器溢出。
位 1	OVFEN	0x0	rw	溢出事件使能 (Overflow event enable) 0: 开启; 1: 关闭。
位 0	TMREN	0x0	rw	使能定时器 (TMR enable) 0: 关闭; 1: 开启。

#### 14.3.4.2 TMR1控制寄存器2 (TMR1\_CTRL2)

域	简称	复位值	类型	功能
位 15	保留	0x0	resd	保持默认值。

位 14	C4IOS	0x0	rw	通道 4 空闲输出状态 (Channel 4 idle output state)
位 13	C3CIOS	0x0	rw	通道 3 互补空闲输出状态 (Channel 3 complementary idle output state)
位 12	C3IOS	0x0	rw	通道 3 空闲输出状态 (Channel 3 idle output state)
位 11	C2CIOS	0x0	rw	通道 2 互补空闲输出状态 (Channel 2 complementary idle output state)
位 10	C2IOS	0x0	rw	通道 2 空闲输出状态 (Channel 2 idle output state)
				通道 1 互补空闲输出状态 (Channel 1 complementary idle output state)
位 9	C1CIOS	0x0	rw	输出关闭 (OEN = 0), 死区发生后: 0: C1OUTL=0; 1: C1OUTL=1。
				通道 1 空闲输出状态 (Channel 1 idle output state)
位 8	C1IOS	0x0	rw	输出关闭 (OEN = 0), 死区发生后: 0: C1OUT=0。 1: C1OUT=1。
				C1IN 选择 (C1IN selection) 0: CH1 管脚连到 C1IRAW 输入; 1: CH1、CH2 和 CH3 管脚异或结果连到 C1IRAW 输入。
位 7	C1INSEL	0x0	rw	
				主定时器输出信号选择 (Primary TMR output selection) TMR1 输出到次定时器的信号选择: 000: 复位; 001: 使能; 010: 溢出; 011: 比较脉冲; 100: C1ORAW 信号; 101: C2ORAW 信号; 110: C3ORAW 信号; 111: C4ORAW 信号。
位 6: 4	PTOS	0x0	rw	
				DMA 请求源 (DMA request source) DMA 请求来源。 0: 通道事件; 1: 溢出事件。
位 3	DRS	0x0	rw	
				通道控制位刷新选择 (Channel control bit refresh select) 对具有互补输出的通道, 如果通道控制位有缓存时: 0: 通过设置 HALL 位刷新控制位; 1: 通过设置 HALL 位或 TRGIN 的上升沿刷新控制位。
位 2	CCFS	0x0	rw	
位 1	保留	0x0	resd	保持默认值。
位 0	CBCTRL	0x0	rw	通道缓存控制 (Channel buffer control) 对具有互补输出的通道: 0: CxEN, CxCEN 和 CxOCTRL 位无缓存; 1: CxEN, CxCEN 和 CxOCTRL 位有缓存。

#### 14.3.4.3 TMR1次定时器控制寄存器 (TMR1\_STCTRL)

域	简称	复位值	类型	功能
位 15	ESP	0x0	rw	外部信号极性 (External signal polarity) 用于选择外部方式。 0: 高电平或上升沿; 1: 低电平或下降沿。
位 14	ECMBEN	0x0	rw	外部时钟模式 B 使能 (External clock mode B enable) 用于启用外部时钟模式 B 0: 关闭; 1: 启开。
位 13: 12	ESDIV	0x0	rw	外部信号除频 (External signal divide) 用于选择降低外部触发频率的除频。 00: 关闭分频; 01: 2 分频;

				10: 4 分频; 11: 8 分频。
位 11: 8	ESF	0x0	rw	外部信号滤波 (External signal filter) 用于过滤外部信号, 当外部信号产生了 N 次之后才能被采样。 0000: 无滤波器, 以 $f_{DTS}$ 采样 0001: $f_{SAMPLING} = f_{CK\_INT}$ , N=2; 0010: $f_{SAMPLING} = f_{CK\_INT}$ , N=4; 0011: $f_{SAMPLING} = f_{CK\_INT}$ , N=8; 0100: $f_{SAMPLING} = f_{DTS}/2$ , N=6; 0101: $f_{SAMPLING} = f_{DTS}/2$ , N=8; 0110: $f_{SAMPLING} = f_{DTS}/4$ , N=6; 0111: $f_{SAMPLING} = f_{DTS}/4$ , N=8; 1000: $f_{SAMPLING} = f_{DTS}/8$ , N=6; 1001: $f_{SAMPLING} = f_{DTS}/8$ , N=8; 1010: $f_{SAMPLING} = f_{DTS}/16$ , N=5; 1011: $f_{SAMPLING} = f_{DTS}/16$ , N=6; 1100: $f_{SAMPLING} = f_{DTS}/16$ , N=8; 1101: $f_{SAMPLING} = f_{DTS}/32$ , N=5; 1110: $f_{SAMPLING} = f_{DTS}/32$ , N=6; 1111: $f_{SAMPLING} = f_{DTS}/32$ , N=8.
位 7	STS	0x0	rw	次定时器同步 (Subordinate TMR synchronization) 该位开启后, 主次定时器可实现高度同步。 0: 关闭; 1: 开启。
位 6: 4	STIS	0x0	rw	次定时器输入选择 (Subordinate TMR input selection) 用于次定时器的输入选择。 000: 内部选择 0 (IS0); 001: 内部选择 1 (IS1); 010: 内部选择 2 (IS2); 011: 内部选择 3 (IS3); 100: C1IRAW 的输入检测器 (C1INC); 101: 滤波输入 1 (C1IF1); 110: 滤波输入 2 (C2IF2); 111: 外部输入 (EXT)。 关于每个定时器中 ISx 的细节, 参见表 14-11。
位 3	保留	0x0	resd	保留, 保持默认值。
位 2: 0	SMSEL	0x0	rw	次定时器模式选择 (Subordinate TMR mode selection) 000: 关闭从模式; 001: 编码模式 A; 010: 编码模式 B; 011: 编码模式 C; 100: 复位模式 - TRGIN 输入上升沿时, 重新初始化 CVAL; 101: 挂起模式 - TRGIN 输入高电平时, CVAL 计数; 110: 触发模式 - TRGIN 输入上升沿时, 产生触发事件; 111: 外部时钟模式 A - TRGIN 输入上升沿时, 提供时钟; 注: 编码器模式 A/B/C 配置方法请查看计数模式章节。

#### 14.3.4.4 TMR1 DMA/中断使能寄存器 (TMR1\_IDEN)

域	简称	复位值	类型	功能
位 15	保留	0x0	resd	保持默认值。
位 14	TDEN	0x0	rw	触发 DMA 请求使能 (Trigger DMA request enable) 0: 关闭; 1: 开启。
位 13	HALLDE	0x0	rw	HALL DMA 请求使能 (HALL DMA request enable)

				0: 关闭; 1: 开启。
位 12	C4DEN	0x0	rw	通道 4 的 DMA 请求使能 (Channel 4 DMA request enable) 0: 关闭; 1: 开启。
位 11	C3DEN	0x0	rw	通道 3 的 DMA 请求使能 (Channel 3 DMA request enable) 0: 关闭; 1: 开启。
位 10	C2DEN	0x0	rw	通道 2 的 DMA 请求使能 (Channel 2 DMA request enable) 0: 关闭; 1: 开启。
位 9	C1DEN	0x0	rw	通道 1 的 DMA 请求使能 (Channel 1 DMA request enable) 0: 关闭; 1: 开启。
位 8	OVFDEN	0x0	rw	溢出事件的 DMA 请求使能 (overflow event DMA request enable) 0: 关闭; 1: 开启。
位 7	BRKIE	0x0	rw	刹车中断使能 (Brake interrupt enable) 0: 关闭; 1: 开启。
位 6	TIEN	0x0	rw	触发中断使能 (Trigger interrupt enable) 0: 关闭; 1: 开启。
位 5	HALLIEN	0x0	rw	HALL 中断使能 (HALL interrupt enable) 0: 关闭; 1: 开启。
位 4	C4IEN	0x0	rw	通道 4 中断使能 (Channel 4 interrupt enable) 0: 关闭; 1: 开启。
位 3	C3IEN	0x0	rw	通道 3 中断使能 (Channel 3 interrupt enable) 0: 关闭; 1: 开启。
位 2	C2IEN	0x0	rw	通道 2 中断使能 (Channel 2 interrupt enable) 0: 关闭; 1: 开启。
位 1	C1IEN	0x0	rw	通道 1 中断使能 (Channel 1 interrupt enable) 0: 关闭; 1: 开启。
位 0	OVFIEN	0x0	rw	更新中断使能 (Update interrupt enable) 0: 关闭; 1: 开启。

#### 14.3.4.5 TMR1中断状态寄存器 (TMR1\_ISTS)

域	简称	复位值	类型	功能
位 15: 13	保留	0x0	resd	保持默认值。
位 12	C4RF	0x0	rw0c	通道 4 再捕获标记 (Channel 4 recapture flag) 见 C1RF 的描述。
位 11	C3RF	0x0	rw0c	通道 3 再捕获标记 (Channel 3 recapture flag) 见 C1RF 的描述。
位 10	C2RF	0x0	rw0c	通道 2 再捕获标记 (Channel 2 recapture flag) 见 C1RF 的描述。
位 9	C1RF	0x0	rw0c	通道 1 再捕获标记 (Channel 1 recapture flag) C1IF 的状态已经为'1'时是否再次发生了捕获, 由硬件置'1', 写'0'清除。 0: 无捕获发生;

位 8	保留	0x0	resd	1: 捕获发生。 保持默认值。
位 7	BRKIF	0x0	rw0c	刹车中断标记 (Brake interrupt flag) 用于标记刹车输入的电平是否有效, 由硬件置'1', 写'0'清除。 0: 无效; 1: 有效。
位 6	TRGIF	0x0	rw0c	触发中断标记 (Trigger interrupt flag) 当发生触发事件时由硬件置'1', 写'0'清除。 0: 无触发事件发生; 1: 发生触发事件。 触发事件: 在 TRGIN 接收到有效边沿, 或挂起模式下接收到任意边沿。
位 5	HALLIF	0x0	rw0c	HALL 中断标记 (HALL interrupt flag) 当发生触发事件时由硬件置'1', 写'0'清除。 0: 无 HALL 事件发生; 1: 发生 HALL 事件。 HALL 事件: CxEN、CxCEN、CxOCTRL 已被更新。
位 4	C4IF	0x0	rw0c	通道 4 中断标记 (Channel 4 interrupt flag) 见 C1IF 的描述。
位 3	C3IF	0x0	rw0c	通道 3 中断标记 (Channel 3 interrupt flag) 见 C1IF 的描述。
位 2	C2IF	0x0	rw0c	通道 2 中断标记 (Channel 2 interrupt flag) 见 C1IF 的描述。
位 1	C1IF	0x0	rw0c	通道 1 中断标记 (Channel 1 interrupt flag) 若通道 1 为输入模式时: 捕获事件发生时由硬件置'1', 由软件清'0'或读 TMR1_C1DT 清'0'。 0: 无捕获事件发生; 1: 发生捕获事件。 若通道 1 为输出模式时: 比较事件发生时由硬件置'1', 由软件清'0'。 0: 无比较事件发生; 1: 发生比较事件。
位 0	OVFIF	0x0	rw0c	溢出中断标记 (Overflow interrupt flag) 当溢出事件发生时由硬件置'1', 由软件清'0'。 0: 无溢出事件发生; 1: 发生溢出事件, 若 TMR1_CTRL1 的 OVFEN=0、 OVFS=0 时: - 当 TMR1_SWEVE 寄存器的 OVFG=1 时产生溢出事件; - 当计数值 CVAL 被触发事件重初始化时产生溢出事件。

#### 14.3.4.6 TMR1软件事件寄存器 (TMR1\_SWEVT)

域	简称	复位值	类型	功能
位 15: 8	保留	0x00	resd	保持默认值。
位 7	BRKSWTR	0x0	wo	软件触发刹车事件 (Brake event triggered by software) 通过软件触发一个刹车事件。 0: 无作用; 1: 制造一个刹车事件。
位 6	TRGSWTR	0x0	wo	软件触发触发事件 (Trigger event triggered by software) 通过软件触发一个触发事件。 0: 无作用; 1: 制造一个触发事件。
位 5	HALLSWTR	0x0	wo	软件触发 HALL 事件 (HALL event triggered by software) 通过软件产生一个 HALL 事件。 0: 无作用; 1: 产生一个 HALL 事件。 注: 该位只对拥有互补输出的通道有效。

位 4	C4SWTR	0x0	wo	软件触发通道 4 事件 (Channel 4 event triggered by software) 见 C1M 的描述。
位 3	C3SWTR	0x0	wo	软件触发通道 3 事件 (Channel 3 event triggered by software) 见 C1M 的描述。
位 2	C2SWTR	0x0	wo	软件触发通道 2 事件 (Channel 2 event triggered by software) 见 C1M 的描述。
位 1	C1SWTR	0x0	wo	C1SWTR: 软件触发通道 1 事件 (Channel 1 event triggered by software) 通过软件触发一个通道 1 事件。 0: 无作用; 1: 制造一个通道 1 事件。
位 0	OVFSWTR	0x0	wo	软件触发溢出事件 (Overflow event triggered by software) 通过软件触发一个溢出事件。 0: 无作用; 1: 制造一个溢出事件。

#### 14.3.4.7 TMR1通道模式寄存器1 (TMR1\_CM1)

通道可用于输入（捕获模式）或输出（比较模式），通道的方向由相应的 CxC 位定义。该寄存器其它位的作用在输入和输出模式下不同。CxOx 描述了通道在输出模式下的功能，CxIx 描述了通道在输入模式下的功能。因此必须注意，同一个位在输出模式和输入模式下的功能是不同的。

##### 输出比较模式

域	简称	复位值	类型	功能
位 15	C2OSEN	0x0	rw	通道 2 输出开关使能 (Channel 2 output switch enable)
位 14: 12	C2OCTRL	0x0	rw	通道 2 输出控制 (Channel 2 output control)
位 11	C2OBEN	0x0	rw	通道 2 输出缓存使能 (Channel 2 output buffer enable)
位 10	C2OIEN	0x0	rw	通道 2 输出立即使能 (Channel 2 output immediately enable)
位 9: 8	C2C	0x0	rw	通道 2 配置 (Channel 2 configure) 当 C2EN='0'时，这些位用于选择通道 2 为输出或输入， 以及输入时的映射选择： 00: 输出; 01: 输入，C2IN 映射在 C2IRAW 上; 10: 输入，C2IN 映射在 C1IRAW 上; 11: 输入，C2IN 映射在 STI 上，只有在 STIS 选择内部 触发输入时才工作。
位 7	C1OSEN	0x0	rw	通道 1 输出开关使能 (Channel 1 output switch enable) 0: EXT 输入不影响 C1ORAW; 1: 当 EXT 输入高电平时，将 C1ORAW 清 0。
位 6: 4	C1OCTRL	0x0	rw	通道 1 输出控制 (Channel 1 output control) 这些位用于设置原始信号 C1ORAW 的工作状态。 000: 断开。断开 C1ORAW 到 C1OUT 的输出; 001: 设置 C1ORAW 为高: TMR1_CVAL=TMR1_C1DT 时。 010: 设置 C1ORAW 为低: TMR1_CVAL=TMR1_C1DT 时。 011: 切换 C1ORAW 的电平: 当 TMR1_CVAL=TMR1_C1DT 时。 100: 固定 C1ORAW 为低。 101: 固定 C1ORAW 为高。 110: PWM 模式 A —OWCDIR=0, 若 TMR1_C1DT>TMR1_CVAL 时设置 C1ORAW 为高，否则为低; —OWCDIR=1, 若 TMR1_C1DT < TMR1_CVAL 时设置 C1ORAW 为低，否则为高。 111: PWM 模式 B —OWCDIR=0, 若 TMR1_C1DT > TMR1_CVAL 时设置 C1ORAW 为低，否则为高;

					-OWCDIR=1, 若 TMR1_C1DT < TMR1_CVAL 时设置 C1ORAW 为高, 否则为低。 注: 除'000'外, 其余配置下 C1OUT 将连接到 C1ORAW, C1OUT 的输出电平除了会根据 C1ORAW 变化外, 还与 CCTRL 所配置的输出极性有关。
位 3	C1OBEN	0x0	rw		通道 1 输出缓存使能 (Channel 1 output buffer enable) 0: 关闭 TMR1_C1DT 的缓存功能, 写入 TMR1_C1DT 的内容会立即生效。 1: 启用 TMR1_C1DT 的缓存功能, 写入 TMR1_C1DT 的内容将保存到缓存寄存器中, 当发生溢出事件时再更新到 TMR1_C1DT 中。
位 2	C1OIEN	0x0	rw		通道 1 输出立即使能 (Channel 1 output immediately enable) 在 PWM 模式 A 或模式 B 下, 该位能够缩短触发事件到通道 1 的输出响应间的时间。 0: 需要比较 CVAL 与 C1DT 的值之后再产生输出。 1: 无需比较 CVAL 与 C1DT 的值, 当发生触发事件时立即产生输出。
位 1: 0	C1C	0x0	rw		通道 1 配置 (Channel 1 configure) 当 C1EN=0'时, 这些位用于选择通道 1 为输出或输入, 以及输入时的映射选择: 00: 输出; 01: 输入, C1IN 映射在 C1IRAW 上; 10: 输入, C1IN 映射在 C2IRAW 上; 11: 输入, C1IN 映射在 STCI 上, 只有在 STIS 选择内部触发输入时才工作。

### 输入模式

域	简称	复位值	类型	功能
位 15: 12	C2DF	0x0	rw	通道 2 滤波器 (Channel 2 digital filter)
位 11: 10	C2IDIV	0x0	rw	通道 2 分频系数 (Channel 2 input divider)
位 9: 8	C2C	0x0	rw	通道 2 配置 (Channel 2 configure) 当 C2EN=0'时, 这些位用于选择通道 2 为输出或输入, 以及输入时的映射选择: 00: 输出; 01: 输入, C2IN 映射在 C2IRAW 上; 10: 输入, C2IN 映射在 C1IRAW 上; 11: 输入, C2IN 映射在 STCI 上, 只有在 STIS 选择内部触发输入时才工作。
位 7: 4	C1DF	0x0	rw	通道 1 滤波器 (Channel 1 digital filter) 这些位用于配置通道 1 的滤波器。滤波的个数为 N, 则表示发生了 N 次采样事件后输入边沿才能通过滤波器: 0000: 无滤波器, 以 $f_{DTS}$ 采样 1000: 采样频率 $f_{SAMPLING} = f_{DTS}/8$ , N=6 0001: 采样频率 $f_{SAMPLING} = f_{CK\_INT}$ , N=2 1001: 采样频率 $f_{SAMPLING} = f_{DTS}/8$ , N=8 0010: 采样频率 $f_{SAMPLING} = f_{CK\_INT}$ , N=4 1010: 采样频率 $f_{SAMPLING} = f_{DTS}/16$ , N=5 0011: 采样频率 $f_{SAMPLING} = f_{CK\_INT}$ , N=8 1011: 采样频率 $f_{SAMPLING} = f_{DTS}/16$ , N=6 0100: 采样频率 $f_{SAMPLING} = f_{DTS}/2$ , N=6 1100: 采样频率 $f_{SAMPLING} = f_{DTS}/16$ , N=8 0101: 采样频率 $f_{SAMPLING} = f_{DTS}/2$ , N=8 1101: 采样频率 $f_{SAMPLING} = f_{DTS}/32$ , N=5 0110: 采样频率 $f_{SAMPLING} = f_{DTS}/4$ , N=6 1110: 采样频率 $f_{SAMPLING} = f_{DTS}/32$ , N=6 0111: 采样频率 $f_{SAMPLING} = f_{DTS}/4$ , N=8 1111: 采样频率 $f_{SAMPLING} = f_{DTS}/32$ , N=8
位 3: 2	C1IDIV	0x0	rw	通道 1 分频系数 (Channel 1 input divider) 这些位定义了通道 1 的分频系数。

位 1: 0	C1C	0x0	rw	00: 不分频, 每一个有效的边沿都会产生一次输入; 01: 每 2 个有效的边沿产生一次输入; 10: 每 4 个有效的边沿产生一次输入; 11: 每 8 个有效的边沿产生一次输入。 注: C1EN='0'时, 分频系数复位。
位 1: 0	C1C	0x0	rw	通道 1 配置 (Channel 1 configure) 当 C1EN='0'时, 这些位用于选择通道 1 为输出或输入, 以及输入时的映射选择: 00: 输出; 01: 输入, C1IN 映射在 C1IRAW 上; 10: 输入, C1IN 映射在 C2IRAW 上; 11: 输入, C1IN 映射在 STCI 上, 只有在 STIS 选择内部触发输入时才工作。

#### 14.3.4.8 TMR1通道模式寄存器2 (TMR1\_CM2)

参看以上 CM1 寄存器描述

##### 输出比较模式

域	简称	复位值	类型	功能
位 15	C4OSEN	0x0	rw	通道 4 输出开关使能 (Channel 4 output switch enable)
位 14: 12	C4OCTRL	0x0	rw	通道 4 输出控制 (Channel 4 output control)
位 11	C4OBEN	0x0	rw	通道 4 输出缓存使能 (Channel 4 output buffer enable)
位 10	C4OIEN	0x0	rw	通道 4 输出立即使能 (Channel 4 output immediately enable)
位 9: 8	C4C	0x0	rw	通道 4 配置 (Channel 4 configure) 当 C4EN='0'时, 这些位用于选择通道 4 为输出或输入, 以及输入时的映射选择: 00: 输出; 01: 输入, C4IN 映射在 C4IRAW 上; 10: 输入, C4IN 映射在 C3IRAW 上; 11: 输入, C4IN 映射在 STCI 上, 只有在 STIS 选择内部触发输入时才工作。
位 7	C3OSEN	0x0	rw	通道 3 输出开关使能 (Channel 3 output switch enable)
位 6: 4	C3OCTRL	0x0	rw	通道 3 输出控制 (Channel 3 output control)
位 3	C3OBEN	0x0	rw	通道 3 输出缓存使能 (Channel 3 output buffer enable)
位 2	C3OIEN	0x0	rw	通道 3 输出立即使能 (Channel 3 output immediately enable)
位 1: 0	C3C	0x0	rw	通道 3 配置 (Channel 3 configure) 当 C3EN='0'时, 这些位用于选择通道 3 为输出或输入, 以及输入时的映射选择: 00: 输出; 01: 输入, C3IN 映射在 C3IRAW 上; 10: 输入, C3IN 映射在 C4IRAW 上; 11: 输入, C3IN 映射在 STCI 上, 只有在 STIS 选择内部触发输入时才工作。

##### 输入模式

域	简称	复位值	类型	功能
位 15: 12	C4DF	0x0	rw	通道 4 滤波器 (Channel 4 digital filter)
位 11: 10	C4IDIV	0x0	rw	通道 4 分频系数 (Channel 4 input divider)
位 9: 8	C4C	0x0	rw	通道 4 配置 (Channel 4 configure) 当 C4EN='0'时, 这些位用于选择通道 4 为输出或输入, 以及输入时的映射选择: 00: 输出; 01: 输入, C4IN 映射在 C4IRAW 上; 10: 输入, C4IN 映射在 C3IRAW 上; 11: 输入, C4IN 映射在 STCI 上, 只有在 STIS 选择内部触发输入时才工作。
位 7: 4	C3DF	0x0	rw	通道 3 滤波器 (Channel 3 digital filter)
位 3: 2	C3IDIV	0x0	rw	通道 3 分频系数 (Channel 3 input divider)

位 1: 0	C3C	0x0	rw	通道 3 配置 (Channel 3 configure) 当 C3EN=’0’时，这些位用于选择通道 3 为输出或输入， 以及输入时的映射选择： 00: 输出； 01: 输入，C3IN 映射在 C3IRAW 上； 10: 输入，C3IN 映射在 C4IRAW 上； 11: 输入，C3IN 映射在 STCI 上，只有在 STIS 选择内部 触发输入时才工作。
--------	-----	-----	----	---

#### 14.3.4.9 TMR1通道控制寄存器 (TMR1\_CCTRL)

域	简称	复位值	类型	功能
位 15: 14	保留	0x0	resd	保持默认值。
位 13	C4P	0x0	rw	通道 4 极性 (Channel 4 polarity) 见 C1P 的描述。
位 12	C4EN	0x0	rw	通道 4 使能 (Channel 4 enable) 见 C1EN 的描述。
位 11	C3CP	0x0	rw	通道 3 互补极性 (Channel 3 complementary polarity) 见 C1P 的描述。
位 10	C3CEN	0x0	rw	通道 3 互补使能 (Channel 3 complementary enable) 见 C1EN 的描述。
位 9	C3P	0x0	rw	通道 3 极性 (Channel 3 polarity) 见 C1P 的描述。
位 8	C3EN	0x0	rw	通道 3 使能 (Channel 3 enable) 见 C1EN 的描述。
位 7	C2CP	0x0	rw	通道 2 互补极性 (Channel 2 complementary polarity) 见 C1P 的描述。
位 6	C2CEN	0x0	rw	通道 2 互补使能 (Channel 2 complementary enable) 见 C1EN 的描述。
位 5	C2P	0x0	rw	通道 2 极性 (Channel 2 polarity) 见 C1P 的描述。
位 4	C2EN	0x0	rw	通道 2 使能 (Channel 2 enable) 见 C1EN 的描述。
位 3	C1CP	0x0	rw	通道 1 互补极性 (Channel 1 complementary polarity) 0: C1COUT 的有效电平为高 1: C1COUT 的有效电平为低
位 2	C1CEN	0x0	rw	通道 1 互补使能 (Channel 1 complementary enable) 0: 禁止输出； 1: 使能输出。
位 1	C1P	0x0	rw	通道 1 极性 (Channel 1 polarity) 通道 1 配置为输出： 0: C1OUT 的有效电平为高 1: C1OUT 的有效电平为低 通道 1 配置为输入： C1CP/C1P 位共同定义输入信号有效沿。 00: C1IN 的有效边沿为上升沿；作为外部触发使用时， C1IN 不反相。 01: C1IN 的有效边沿为下降沿；作为外部触发使用时， C1IN 反相。 10: 保留 11: C1IN 的有效边沿为上升沿和下降沿；作为外部触发 使用时，C1IN 不反相。
位 0	C1EN	0x0	rw	通道 1 使能 (Channel 1 enable) 0: 禁止输入或输出； 1: 使能输入或输出。

表 14-14 带刹车功能的互补输出通道CxOUT和CxCOUT的控制位

控制位	输出状态 (1)
-----	----------

OEN 位	FCSODIS 位	FCSOEN 位	CxEN 位	CxCEN 位	CxOUT 输出状态	CxCOUT 输出状态
1	X	0	0	0	输出禁止 (与定时器断开) CxOUT=0, Cx_EN=0	输出禁止(与定时器断开) CxOUT=0, CxCEN=0
		0	0	1	输出禁止 (与定时器断开) CxOUT=0, Cx_EN=0	CxORAW + 极性, CxOUT=CxORAW xor CxCP, CxCEN=1
		0	1	0	CxORAW+极性, CxOUT=CxORAW xor CxP, Cx_EN=1	输出禁止(与定时器断开) CxOUT=0, CxCEN=0
		0	1	1	CxORAW+极性+死区, Cx_EN=1	CxORAW 反相+极性+死区, CxCEN=1
		1	0	0	输出禁止(与定时器断开) CxOUT=CxP, Cx_EN=0	输出禁止(与定时器断开) CxOUT=CxP, CxCEN=0
		1	0	1	关闭状态 (输出使能且为无效电平) CxOUT=CxP, Cx_EN=1	CxORAW + 极性, CxOUT=CxORAW xor CxCP, CxCEN=1
		1	1	0	CxORAW + 极性, CxOUT=CxORAW xor CxP, Cx_EN=1	关闭状态 (输出使能且为无效电平) CxOUT=CxCP, CxCEN=1
		1	1	1	CxORAW+极性+死区, Cx_EN=1	CxORAW 反相+极性+死区, CxCEN=1
0	X	0	0	0	输出禁止(与定时器断开) 异步地: CxOUT=CxP , Cx_EN=0 , CxOUT=CxCP , CxCEN=0; 若时钟存在: 经过一个死区时间后 CxOUT=CxIOS, CxOUT=CxCIOS, 假设 CxIOS 与 CxCIOS 并不都对应 CxOUT 和 CxCOUT 的有效电平。	
		0	0	1		
		0	1	0		
		0	1	1		
		1	0	0		
		1	0	1		
		1	1	0		
		1	1	1		

注意: 如果一个通道的 2 个输出都没有使用 ( $CxEN = CxCEN = 0$ ) , 那么  $CxIOS$ ,  $CxCIOS$ ,  $CxP$  和  $CxCP$  都必须清零。

注意: 管脚连接到互补的  $CxOUT$  和  $CxCOUT$  通道的外部 I/O 管脚的状态, 取决于  $CxOUT$ 、 $CxCOUT$  通道状态和 GPIO 以及 IOMUX 寄存器。

#### 14.3.4.10 TMR1计数值 (TMR1\_CVAL)

域	简称	复位值	类型	功能
位 15: 0	CVAL	0x0000	rw	计数值 (Counter value)

#### 14.3.4.11 TMR1预分频器 (TMR1\_DIV)

域	简称	复位值	类型	功能
位 15: 0	DIV	0x0000	rw	分频系数 (Divider value)

计数器时钟频率  $f_{CK\_CNT} = f_{TMR\_CLK} / (\text{DIV}[15: 0]+1)$   
溢出事件发生时该寄存器值被传送到实际的预分频寄存器中。

#### 14.3.4.12 TMR1周期寄存器 (TMR1\_PR)

域	简称	复位值	类型	功能
位 15: 0	PR	0x0000	rw	周期值 (Period value) 定时器计数的周期值。当周期值为 0 时，定时器不工作。

#### 14.3.4.13 TMR1重复周期寄存器 (TMR1\_RPR)

域	简称	复位值	类型	功能
位 15: 8	保留	0x00	resd	保持默认值。
位 7: 0	RPR	0x00	rw	重复周期的次数 (Repetition of period value) 这些位用于减慢溢出事件发生的速率，当重复周期的次数减为 0 时才会发生溢出事件。

#### 14.3.4.14 TMR1通道1数据寄存器 (TMR1\_C1DT)

域	简称	复位值	类型	功能
位 15: 0	C1DT	0x0000	rw	通道 1 数据寄存器值 (Channel 1 data register) 若通道 1 配置为输入： C1DT 是前一次通道 1 输入事件 (C1IN) 所保存的 CVAL。 若通道 1 配置为输出： C1DT 是将要和 CVAL 进行比较的值，写入的值是否会立即生效取决于输出缓存使能位 (C1OBEN)，并根据设置在 C1OUT 上产生相应的输出。

#### 14.3.4.15 TMR1通道2数据寄存器 (TMR1\_C2DT)

域	简称	复位值	类型	功能
位 15: 0	C2DT	0x0000	rw	通道 2 数据寄存器值 (Channel 2 data register) 若通道 2 配置为输入： C2DT 是前一次通道 2 输入事件 (C2IN) 所保存的 CVAL。 若通道 2 配置为输出： C2DT 是将要和 CVAL 进行比较的值，写入的值是否会立即生效取决于输出缓存使能位 (C2OBEN)，并根据设置在 C2OUT 上产生相应的输出。

#### 14.3.4.16 TMR1通道3数据寄存器 (TMR1\_C3DT)

域	简称	复位值	类型	功能
位 15: 0	C3DT	0x0000	rw	通道 3 数据寄存器值 (Channel 3 data register) 若通道 3 配置为输入： C3DT 是前一次通道 3 输入事件 (C3IN) 所保存的 CVAL。 若通道 3 配置为输出： C3DT 是将要和 CVAL 进行比较的值，写入的值是否会立即生效取决于输出缓存使能位 (C3OBEN)，并根据设置在 C3OUT 上产生相应的输出。

#### 14.3.4.17 TMR1通道4数据寄存器 (TMR1\_C4DT)

域	简称	复位值	类型	功能
位 15: 0	C4DT	0x0000	rw	通道 4 数据寄存器值 (Channel 4 data register) 若通道 4 配置为输入： C4DT 是前一次通道 4 输入事件 (C4IN) 所保存的 CVAL。 若通道 4 配置为输出： C4DT 是将要和 CVAL 进行比较的值，写入的值是否会立即生效取决于输出缓存使能位 (C4OBEN)，并根据设置在 C4OUT 上产生相应的输出。

#### 14.3.4.18 TMR1刹车寄存器 (TMR1\_BRK)

域	简称	复位值	类型	功能
位 15	OEN	0x0	rw	输出使能 (Output enable) 对配置为输出的通道，该位用于使能 CxOUT 和 CxCOUT 的输出。 0: 关闭; 1: 开启。
位 14	AOEN	0x0	rw	输出自动使能(Automatic output enable) 用于溢出事件时将 OEN 自动置'1' 0: 关闭; 1: 开启
位 13	BRKV	0x0	rw	刹车输入信号的有效性 (Brake input validity) 用于选择刹车输入信号的输入有效电平： 0: 低电平; 1: 高电平。
位 12	BRKEN	0x0	rw	刹车功能使能 (Brake enable) 用于开启刹车功能。 0: 关闭; 1: 开启。
位 11	FCSOEN	0x0	rw	总输出开时的冻结状态 (Frozen channel status when holistic output enable) 该位用于配置具有互补输出的通道，在定时器不工作且 MOEN=1 时的通道状态。 0: 关闭 CxOUT/CxCOUT 输出; 1: 开启 CxOUT/CxCOUT 输出，输出为无效电平。
位 10	FCSODIS	0x0	rw	总输出关时的冻结状态 (Frozen channel status when holistic output disable) 该位用于配置具有互补输出的通道，在定时器不工作且 MOEN=0 时的通道状态。 0 : 关闭 CxOUT/CxCOUT 输出; 1 : 开启 CxOUT/CxCOUT 输出，输出为空闲电平。
位 9: 8	WPC	0x0	rw	写保护配置 (Write protected configuration) 该位用于配置写保护。 00: 写保护关闭; 01: 3 级写保护，以下位受写保护: TMR1_BRK: DTC、BRKEN、BRKV 和 AOEN TMR1_CTRL2: CxIOS 和 CxCIOs 10: 2 级写保护，除 3 级写保护的内容外，以下位也受写保护: TMR1_CCTRL: CxP 和 CxCP TMR1_BRK: FCSODIS 和 FCSOEN 11: 1 级写保护，除 2 级写保护的内容外，以下位也受写保护: TMR1_CMx: C2OCTRL 和 C2OBEN 注：WPC>0 时将无法再次被修改，直到系统复位。
位 7: 0	DTC	0x00	rw	死区配置 (Dead-time configuration) 这些位用于配置死区时间。取 DTC[7: 0]的高 3 位为功能选择位： 0xx: DT = DTC [7: 0] * TDTS; 10x: DT = (64+ DTC [5: 0]) * TDTS * 2; 110: DT = (32+ DTC [4: 0]) * TDTS * 8; 111: DT = (32+ DTC [4: 0]) * TDTS * 16;

注意：根据锁定设置，AOEN、BRKV、BRKEN、FCSODIS、FCSOEN 和 DTC[7: 0]位均可被写保护，有必要在第一次写入 TMR1\_BRK 寄存器时对它们进行配置。

#### 14.3.4.19 TMR1 DMA控制寄存器 (TMR1\_DMACTRL)

域	简称	复位值	类型	功能
位 15: 13	保留	0x0	resd	保持默认值。
位 12: 8	DTB	0x00	rw	DMA 传输字节 (DMA transfer bytes) 这些位定义了传输的字节个数： 00000: 1 个字节      00001: 2 个字节

				00010: 3个字节	00011: 4个字节
				.....	.....
				10000: 17个字节	10001: 18个字节
位 7: 5	保留	0x0	resd	保持默认值。	
位 4: 0	ADDR	0x00	rw	DMA 传输地址偏移 (DMA transfer address offset) ADDR 定义了从 TMR1_CTRL1 所在地址开始的偏移量: 00000: TMR1_CTRL1, 00001: TMR1_CTRL2, 00010: TMR1_STCTRL, .....	

#### 14.3.4.20 TMR1 DMA数据寄存器 (TMR1\_DMADT)

域	简称	复位值	类型	功能
位 15: 0	DMADT	0x0000	rw	DMA 传输的数据寄存器 (DMA data register) 通过对 DMADT 寄存器的读写能够实现对任意 TMR 寄存器的操作，其操作的寄存器地址范围是： TMR1 外设地址 + ADDR*4 至 TMR1 外设地址 + ADDR*4 + DTB*4。

# 15 窗口看门狗 (WWDT)

## 15.1 WWDT简介

当程序正常运行时，需在一个有限的时间窗口内重载窗口看门狗递减计数器，用来避免看门狗电路产生系统复位，以此来监测系统是否正常运行。

窗口看门狗时钟由 APB1\_CLK 分频而来，由于 APB1\_CLK 的精确性，窗口看门狗可对有限的时间窗口精确控制。

## 15.2 WWDT主要特性

- 7 位递减计数器
- 启动看门狗后，当递减计数器的值小于 0x40 或是在窗口外被重新装载产系统生复位。
- 可以通过重载计数器中断重装载计数器。

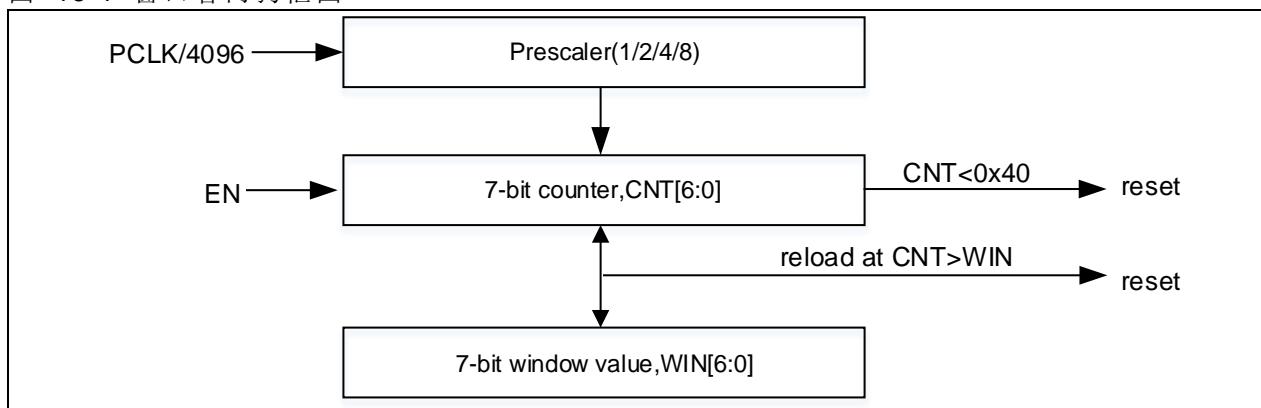
## 15.3 WWDT功能描述

启动窗口看门狗后，窗口看门狗可在以下两种情况下产生系统复位：

第一种，7 位递减计数器值由 0x40 变为 0x3F。

第二种，7 位递减计数器值大于 7 位窗口值时，重载计数器值。

图 15-1 窗口看门狗框图



为避免重载计数器值时产生复位，应在计数器值小于窗口值大于 0x40 时重载计数器值。

WWDT 计数器时钟由 APB1\_CLK 分频得到，分频系数可通过配置配置寄存器 (WWDT\_CFG) DIV[1:0] 改变。计数器值决定了 WWDT 复位前的最大计数周期数，结合 WIN[6:0] 可灵活的调整重载窗口。

WWDT 提供了重载计数器中断功能，开启后，WWDT 将在计数值达到 0x40h 时将 RLDF 标志位置 1，同时产生重载计数器中断，可在中断服务程序中重载计数器值，以避免发生系统复位。需要注意的是，若在 CNT[6] 为 0 时，将 WWDTEN 置 1 会产生一个系统复位，因此当写入控制寄存器 (WWDT\_CTRL) 时，应始终保持 CNT[6] 为 1，避免使能窗口看门狗后立即产生一个系统复位。

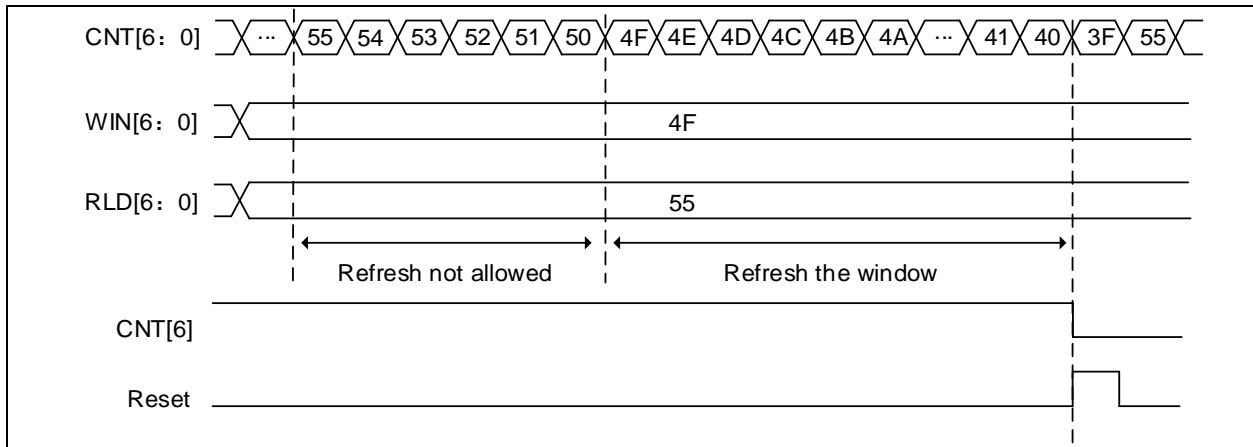
窗口看门狗超时时间  $T_{WWDT}$  可由一下公式计算，其中  $T_{PCLK1}$  为 APB1 时钟周期，单位为 ms:

$$T_{WWDT} = T_{PCLK1} \times 4096 \times 2^{DIV[1:0]} \times (CNT[5:0] + 1); \quad (\text{ms})$$

表 15-1 PCLK1 频率为 72MHz 时，最大和最小看门狗超时时间

时钟预分频值	最小超时时间	最大超时时间
0	56.5 μs	3.64ms
1	113.5 μs	7.28ms
2	227.5 μs	14.56ms
3	455 μs	29.12ms

图 15-2 窗口看门狗时序图



## 15.4 调试模式

微控制器处于调试模式时，意味着 Cortex™-M4 核心停止。将 DEBUG 模块中 WWDT\_PAUSE 位置 1 可将 WWDT 计数器计数暂停。

## 15.5 WWDT 寄存器

必须以字（32 位）的方式操作这些外设寄存器。

表 15-2 WWDT 寄存器映像和复位值

寄存器简称	基址偏移量	复位值
WWDT_CTRL	0x00	0x7F
WWDT_CFG	0x04	0x7F
WWDT_STS	0x08	0x00

### 15.5.1 控制寄存器 (WWDT\_CTRL)

域	简称	复位值	类型	功能
位 31: 8	保留	0x000000	resd	保持默认值。
位 7	WWDTEN	0x0	rw1s	窗口看门狗使能 (Window watchdog enable) 0: 关闭; 1: 开启。 该位由软件置起，只能在复位后自动清零。
位 6: 0	CNT	0x7F	rw	递减计数器 (Decrement counter) 当计数器递减到 0x3F 时产生复位。

### 15.5.2 配置寄存器 (WWDT\_CFG)

域	简称	复位值	类型	功能
位 31: 10	保留	0x000000	resd	保持默认值。
位 9	RLDIEN	0x0	rw	重载计数器中断 (Reload counter interrupt) 0: 关闭; 1: 开启。
位 8: 7	DIV	0x0	rw	时钟预分频值 (Clock division value) 00: PCLK1 除以 4096; 01: PCLK1 除以 8192; 10: PCLK1 除以 16384; 11: PCLK1 除以 32768。
位 6: 0	WIN	0x7F	rw	窗口值 (Window value) 当计数器值大于窗口值时，此时重载计数器会产生复位，重载计数器区间为 0x40~WIN[6: 0]

### 15.5.3 状态寄存器 (WWDT\_STS)

域	简称	复位值	类型	功能
位 31: 1	保留	0x0000 0000	resd	保持默认值。
位 0	RLDF	0x0	rw0c	重载计数器中断标志 (Reload counter interrupt flag) 当递减计数器为 0x40 时，该标志会置位。 该位被硬件置起，由软件将其清零。'

# 16 看门狗 (WDT)

## 16.1 WDT简介

看门狗由专用低速时钟 (LICK) 驱动，由于 LICK 时钟精度较低，因此看门狗适用于低时间精度、能够独立于主程序之外的应用

## 16.2 WDT主要特性

- 12 位递减计数器
- 计数器由 LICK 时钟驱动（可在深度睡眠和待机模式下工作）
- 看门狗使能后，将在计数器计数至 0 时产生 WDT 系统复位

## 16.3 WDT功能描述

### WDT 启动方式:

WDT 的启动方式有两种，分别为软件启动和硬件启动。软件启动通过向命令寄存器 (WDT\_CMD) 写入 0xCCCC 实现；硬件启动则需通过配置用户系统数据区来实现，使能硬件看门狗后，看门狗将在上电复位后自动开始运行。

### WDT 复位条件:

当 WDT 计数器值递减至 0 时将产生 WDT 系统复位，因此需定时向命令寄存器 (WDT\_CMD) 写入 0xAAAA 重载计数器值。

### WDT 写保护:

预分频寄存器 (WDT\_DIV)、重装载寄存器 (WDT\_RLD) 受写保护，向命令寄存器 (WDT\_CMD) 写入 0x5555 可解锁寄存器写保护，之后可对其进行配置。这两个寄存器的更新状态分别由状态寄存器 (WDT\_STS) 中 DIVF、RLDF 指示。向命令寄存器 (WDT\_CMD) 写入其它值将重新启动预分频寄存器 (WDT\_DIV)、重装载寄存器 (WDT\_RLD) 写保护。向命令寄存器 (WDT\_CMD) 写入 0xAAAA 也会启动寄存器写保护。

### WDT 时钟:

WDT 计数器由 LICK 时钟驱动，LICK 是内部 RC 时钟，其典型值为 40kHz，范围为 30kHz~60kHz 之间，所以超时时间也是在一定区间内，使用时应注意在超时时间配置上应该留有余量，如果需要获得较为精确的看门狗超时时间，可对 LICK 进行校准，有关 LICK 校准的问题，详见 [4.1.1 节](#)。

图 16-1 看门狗框图

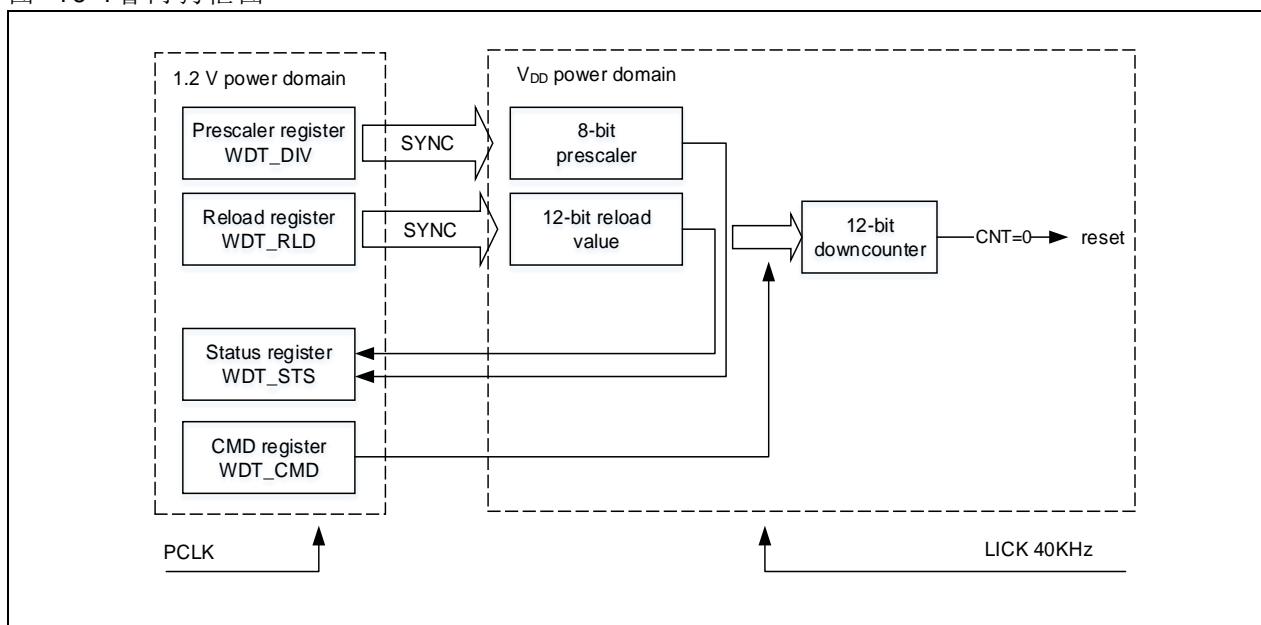


表 16-1 看门狗超时时间 (LICK=40kHz)

预分频系数	DIV[2: 0]位	最短时间 (ms) RLD[11: 0] = 0x000	最长时间 (ms) RLD[11: 0] = 0xFFFF
/4	0	0.1	409.6
/8	1	0.2	819.2
/16	2	0.4	1638.4
/32	3	0.8	3276.8
/64	4	1.6	6553.6
/128	5	3.2	13107.2
/256	(6 或 7)	6.4	26214.4

## 16.4 调试模式

微控制器处于调试模式时，意味着 Cortex™-M4 核心停止。此时将 DEBUG 模块中 WDT\_PAUSE 位置 1 会暂停 WDT 计数器计数。

## 16.5 WDT 寄存器

必须以字（32 位）的方式操作这些外设寄存器。

表 16-2 WDT 寄存器映像和复位值

寄存器简称	基址偏移量	复位值
WDT_CMD	0x00	0x0000 0000
WDT_DIV	0x04	0x0000 0000
WDT_RLD	0x08	0x0000 0FFF
WDT_STS	0x0C	0x0000 0000

### 16.5.1 命令寄存器 (WDT\_CMD)

(在待机模式复位)

域	简称	复位值	类型	功能
位 31: 16	保留	0x0000	resd	保持默认值。 命令寄存器 (Command register) 0xAEEE: 重载计数器； 0x5555: 解锁预分频寄存器 (WDT_DIV) 和重装载寄存器 (WDT_RLD) 写保护； 0xCCCC: 启动看门狗，如果使能了硬件看门狗，则不需要执行此操作。
位 15: 0	CMD	0x0000	wo	

### 16.5.2 预分频寄存器 (WDT\_DIV)

(待机模式时复位)

域	简称	复位值	类型	功能
位 31: 3	保留	0x0000 0000	resd	保持默认值。 递减计数器时钟预分频值 (Clock division value) 000: LICK 除以 4; 001: LICK 除以 8; 010: LICK 除以 16; 011: LICK 除以 32; 100: LICK 除以 64; 101: LICK 除以 128; 110: LICK 除以 256; 111: LICK 除以 256。
位 2: 0	DIV	0x0	rw	

只有解锁写保护后才能写此寄存器，只有当 DIVF 为 0 时，才能读取此寄存器。

### 16.5.3 重装载寄存器 (WDT\_RLD)

(待机模式时复位)

域	简称	复位值	类型	功能
位 31: 12	保留	0x00000	resd	保持默认值。
位 11: 0	RLD	0xFFFF	rw	重载值 (Reload value) 只有解锁写保护后才能写此寄存器，只有当 RLDF 为 0 时，才能读取此寄存器。

### 16.5.4 状态寄存器 (WDT\_STS)

(待机模式时复位)

域	简称	复位值	类型	功能
位 31: 2	保留	0x0000 0000	resd	保持默认值。
位 1	RLDF	0x0	ro	重载值更新完成标志 (Reload value update complete flag) 0: 更新完成; 1: 正在更新。 只有当 RLDF 为 0 时才能写重装载寄存器 (WDT_RLD)。
位 0	DIVF	0x0	ro	分频值更新完成标志 (Division value update complete flag) 0: 更新完成; 1: 正在更新。 只有当 DIVF 为 0 时才能写预分频寄存器 (WDT_DIV)。

# 17 实时时钟 (ERTC)

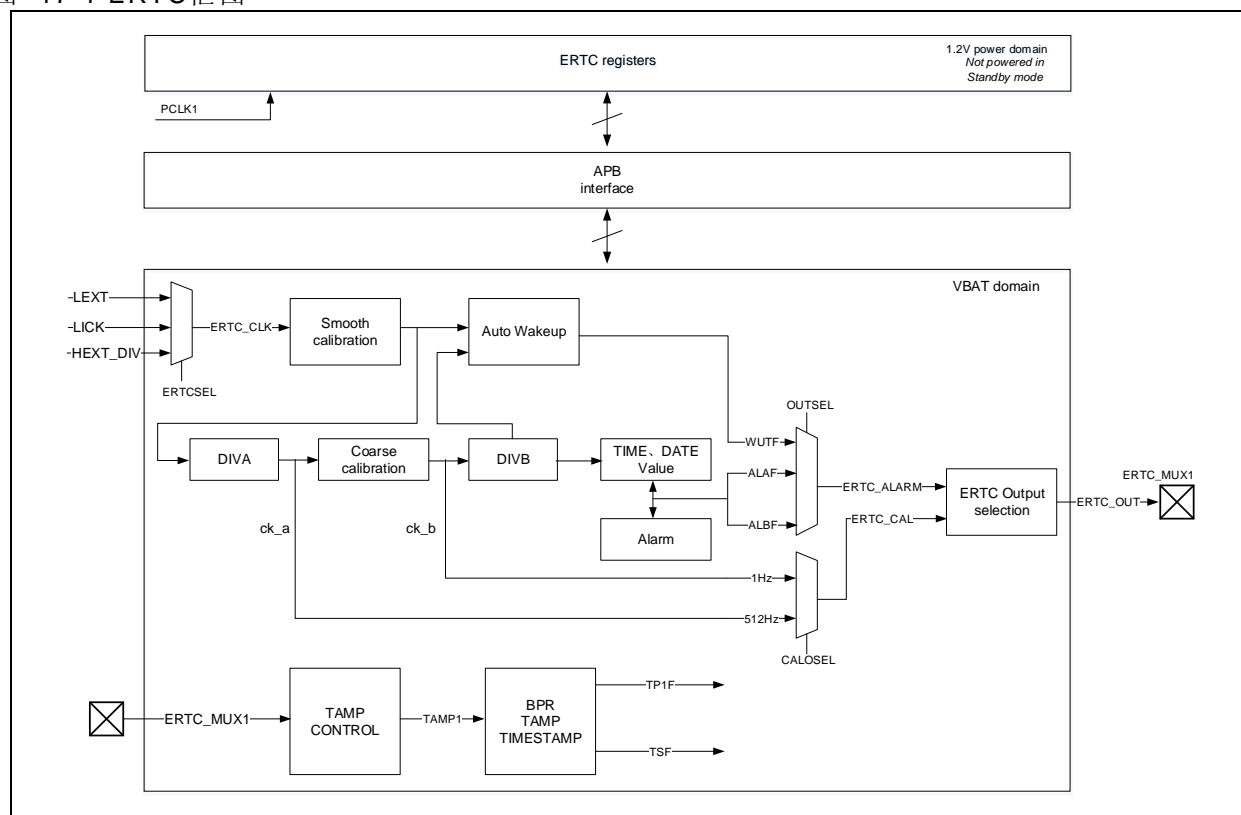
## 17.1 ERTC简介

实时时钟用于配置日历时钟，修改 ERTC 中日历寄存器值可以修改系统的当前时间和日期。ERTC 计数逻辑位于电池供电域，只要电池供电域有电，ERTC 便会一直运行，不受系统复位以及 VDD 掉电影响。

## 17.2 ERTC主要特性

- 功能强大的实时日历，支持两组闹钟
- 周期性唤醒
- 参考时钟检测
- 两组可配置入侵检测，支持时间戳功能
- 支持精密校准、粗略校准两种校准
- 20 个电池供电寄存器
- 5 组中断：闹钟 A、闹钟 B、周期性唤醒、入侵检测、时间戳
- 复用功能输出，校准时钟输出、闹钟事件或唤醒事件
- 复用功能输入，参考时钟输入、一路入侵检测、时间戳

图 17-1 ERTC 框图



## 17.3 ERTC功能说明

### 17.3.1 ERTC时钟

ERTC\_CLK 可从 LEXT、LICK、分频后的 HEXT 中选择（由电池供电域控制寄存器（CRM\_BPDC）ERTCSEL[1: 0]配置）。

ERTC 内置分频器 A 和分频器 B，分别由 DIVA[6: 0]、DIVB[14: 0]配置，推荐 DIVA 配置为较高的值，以最大程度降低功耗。ERTC\_CLK 依次经由分频器 A、分频器 B 处理，得到 ck\_a、ck\_b 时钟，ck\_a 用于更新亚秒，ck\_b 用于更新日历和周期性唤醒。ck\_a、ck\_b 时钟频率可由下式计算：

$$f_{ck\_a} = \frac{f_{ERTC\_CLK}}{DIVA + 1}$$

$$f_{ck\_b} = \frac{f_{ERTC\_CLK}}{(DIVB + 1) \times (DIVA + 1)}$$

当配置 DIVA=127, DIVB=255, 且 ERTC\_CLK 选用 32.768kHz 的 LEXT 时, 可得到 1Hz 的 ck\_b, 用于更新日历。

### 17.3.2 ERTC初始化

#### 寄存器解锁:

上电复位后所有 ERTC 寄存器处于写保护状态, 需要先解除写保护, 才能写配置 ERTC 寄存器(除 ERTC 初始化和状态寄存器 (ERTC\_STS) [14: 8]、ERTC 入侵配置寄存器 (ERTC\_TAMP) 和 ERTC 电池供电数据寄存器 (RTC\_BPRx) 外, 其它寄存器位均受写保护, 且写保护不受系统复位影响)。

#### 解锁步骤:

- 1、使能电源接口时钟: APB1 外设时钟使能寄存器 (CRM\_APB1EN) 的 PWCEN=1
- 2、解锁电池供电域写保护: 电源控制寄存器 (PWC\_CTRL) 的 BPWEN=1
- 3、依次向 ERTC 写保护寄存器 (ERTC\_WP) 写入 0xCA, 0x53, 若向 ERTC 写保护寄存器 (ERTC\_WP) 写入错误的值, 将重新激活写保护。

下表列出了需要解除写保护和进入初始化模式才可以配置的 ERTC 寄存器:

表 17-1 ERTC 寄存器配置表

寄存器简称	是否受 ERTC_WP 写保护	是否需要进入初始化模式	其它
ERTC_TIME	是	是	-
ERTC_DATE	是	是	-
ERTC_CTRL	是	位 7、6、4 需要	-
ERTC_STS	除位[14: 8]外	-	-
ERTC_DIV	是	是	-
ERTC_WAT	是	否	WATWF 为 1 时可配置
ERTC_CCAL	是	是	-
ERTC_ALA	是	否	ALAWF 为 1 时可配置
ERTC_ALB	是	否	ALBWF 为 1 时可配置
ERTC_WP	-	-	-
ERTC_SBS	-	-	-
ERTC_TADJ	是	否	TADJF 为 0 时可配置
ERTC_TSTM	-	-	-
ERTC_TSDT	-	-	-
ERTC_TSSBS	-	-	-
ERTC_SCAL	是	否	CALUPDF 为 0 时可配置
ERTC_TAMP	否	否	-
ERTC_ALASBS	是	否	ALAWF 为 1 时可配置
ERTC_ALBSBS	是	否	ALBWF 为 1 时可配置
ERTC_BPRx	否	否	-

#### 时钟、日历初始化:

寄存器解锁后, 时钟和日历的初始化配置可按以下步骤进行:

1. 将 IMEN 位置 1 进入初始化模式。
2. 等待初始化标志位 INITF 置 1。
3. 依次配置 DIVB、DIVA。

4. 配置时钟和日历值。
5. 将 IMEN 位清 0 退出初始化模式，等待 UPDF 置 1，表明白天同步完成，日历开始计数。

为了方便时间微调，ERTC 还提供了夏令时和时间调整功能。

夏令时功能：用于增加（ADD1H=1）或减小（DEC1H=1）1 小时，而无需重新进行初始化配置。

时间调整功能：用于精确的调整当前时钟。若只配置 DECSBS[14: 0]值，该值将会加到分频器 B 计数器值中，时钟因此产生延迟；若只将 ADD1S 位置 1，当前时钟将增加 1 秒；若同时配置 DECSBS[14: 0]，ADD1S 位，时钟将增加零点几秒。

延迟时间（ADD1S=0）：延迟时间=DECSBS/(DIVB+1)；

提前时间（ADD1S=1）：提前时间=1-(DECSBS/(DIVB+1))。

注：设置时间调整寄存器前，必须先确认 SBS[15]=0，以免亚秒发生上溢；参考时钟检测与粗略数字校准功能不能同时使用：当 RCDEN=1 时，粗略数字校准功能不可用。

#### 日历读取：

ERTC 提供两种日历访问方式，分别为同步读取（DREN=0）和异步读取（DREN=1）。

同步读取时，ERTC 通过 PCLK1 访问同步的影子寄存器来获取时钟和日历值。影子寄存器值由位于电池供电域的 ERTC 日历值同步而来，同步周期为两个 ERTC\_CLK，同步完成后 UPDF 将置 1。影子寄存器由系统复位来复位。为保证读取的 ERTC 亚秒寄存器（ERTC\_SBS）、ERTC 时间寄存器（ERTC\_TIME）、ERTC 日期寄存器（ERTC\_DATE）值来自同一时刻，读取低阶寄存器时会将高阶寄存器值锁定，直到读取 ERTC 日期寄存器（ERTC\_DATE）。例如读取 ERTC 亚秒寄存器（ERTC\_SBS），会将 ERTC 时间寄存器（ERTC\_TIME）、ERTC 日期寄存器（ERTC\_DATE）值锁定。

异步读取时，ERTC 通过 PCLK1 直接读取位于电池供电域的 ERTC 时钟和日历值，这样避免了由于同步时间带来的误差。异步读取时，UPDF 标志将由硬件清 0。为保证异步读取时钟和日历值的准确性，软件必须两次读取时钟和日历值，并比较两次结果是否一致，如果不一致应该再读，直到两次结果一致，另外，也可以只比较两次结果的最低位来判定。

注：在 STANDBY 和 DEEPSLEEP 模式下，当前日历值不会复制到影子寄存器中，当从这两种模式唤醒时，必须先设置 UPDF=0，然后等待 UPDF=1，以保证读取的日历值是最新的；在同步读取时，需保证 PCLK1 频率至少为 ERTC\_CLK 频率 7 倍；异步读取时，需要额外一个 APB 周期才能完成读取日历寄存器的指令。

#### 闹钟初始化：

ERTC 包含闹钟 A、闹钟 B 两个闹钟，并分别提供了闹钟 A 中断、闹钟 B 中断。

闹钟值由 ERTC\_ALASBS/ERTC\_ALA（ERTC\_ALBSBS/ERTC\_ALB）设定，开启闹钟后，当设定的闹钟值匹配日历值时将触发闹钟事件。通过 MASKx 位，可选择性的屏蔽日历字段，被屏蔽的字段不参与闹钟匹配。

闹钟 A、B 的配置可按以下步骤进行：

1. 关闭闹钟 A 或闹钟 B（设置 ALAEN=0 或 ALBEN=0）。
2. 等待闹钟 A 或闹钟 B 寄存器允许写（等待 ALAWF 或 ALBWF 位置 1）。
3. 配置闹钟 A 或闹钟 B 寄存器（ERTC\_ALA/ERTC\_ALASBS、ERTC\_ALB/ERTC\_ALBSBS）。
4. 使能闹钟 A 或闹钟 B（设置 ALAEN=1 或 ALBEN=1）。

注：当 ERTC 闹钟 A 寄存器（ERTC\_ALA）或 ERTC 闹钟 B 寄存器（ERTC\_ALB）中 MASK1 为 0 时，DIVB 至少为 3 才能使闹钟正常工作。

### 17.3.3 周期性自动唤醒

周期性唤醒功能用于 ERTC 周期性的自动唤醒低功耗模式，唤醒周期由 VAL[15: 0]设定（WATCLK[2]=1 时扩展为 17 位，唤醒计数值为  $VAL + 2^{16}$ ）。当唤醒计数器值由 VAL 值递减至 0 时，WATF 标志置 1，产生唤醒事件，同时唤醒计数器值重载 VAL 值。若使能周期性唤醒中断，将产生周期性唤醒中断。

驱动唤醒定时器的时钟通过 WATCLK[2: 0]设定，可选 16/8/4/2 分频后的 ERTC\_CLK 或 ck\_b(通常 1Hz)，结合唤醒计数值可灵活调整唤醒周期。

周期性唤醒功能可按以下步骤进行配置：

1. 关闭周期性唤醒（设置 WATEN=0）。
2. 等待唤醒自动重载定时器和 WATCLK[2: 0]位允许写（WATWF 位置 1）。
3. 配置唤醒定时器计数值和唤醒时钟（VAL[15: 0]、WATCLK[2: 0]位）。
4. 使能定时器（设置 WATEN=1）。

注：系统复位以及低功耗模式（睡眠、深睡眠和待机）对唤醒定时器没有任何影响。

### 17.3.4 ERTC校准

ERTC 提供了两种校准方法：粗略校准和精密校准。但两种校准方法不能同时使用。

#### 粗略数字校准：

粗略数字校准通过增加或减少 ck\_a 周期值来实现提前或推迟更新日历值的功能。

正校准时 (CALDIR=0)，在 64 分钟的前 2xCALVAL 分钟时间内，每分钟 (约 15360 个 ck\_a 周期) 插入 2 个 ck\_a 周期，相当于提前更新日历。

负校准时 (CALDIR=1)，在 64 分钟前的 2xCALVAL 分钟时间内，每分钟 (约 15360 个 ck\_a 周期) 忽略 1 个 ck\_a 周期，相当于推迟更新日历。

注：粗略数字校准至少要将 DIVA 值设置为 6。

#### 精密数字校准：

区别于粗略数字校准，精密校准的校准效果更好且校准更加均匀。开启精密校准校准功能后，将均匀增加或减少 ERTC\_CLK 来达到校准的目的。

当 ERTC\_CLK 为 32.768kHz 时，精密校准周期约为  $2^{20}$  个 ERTC\_CLK(32 秒)。DEC[8: 0]值指定了  $2^{20}$  个 ERTC\_CLK 中忽略的脉冲数，最多可忽略 511 个脉冲；将 ADD 置 1，可在  $2^{20}$  个 ERTC\_CLK 中插入 512 个脉冲。两者搭配使用，可在  $2^{20}$  个 ERTC\_CLK 周期进行-511~+512 的调整。

有效校准频率  $F_{SCAL}$ ：

$$F_{SCAL} = F_{ERTC\_CLK} \times [1 + \frac{ADD \times 512 - DEC}{2^{20} + DEC - ADD \times 512}]$$

当分频器 A 值小于 3 时，会按照 ADD 等于 0 校准。此时应降低分频器 B 值来实现每秒增加 8 个 ERTC\_CLK，也就是 32 秒增加 256 个 ERTC\_CLK 搭配 DEC[8: 0]位，可在  $2^{20}$  个 ERTC\_CLK 周期进行-255~+256 的调整。

此时有效校准频率  $F_{SCAL}$ ：

$$F_{SCAL} = F_{ERTC\_CLK} \times [1 + \frac{256 - DEC}{2^{20} + DEC - 256}]$$

精密数字校准的校准周期还可选择 8 秒或 16 秒(由 CAL8 和 CAL16 配置)，8 秒校准周期的优先级更高，同时使能 8 秒和 16 秒校准周期，将优先选择 8 秒校准周期。

ERTC 提供了 CALUPDF 标志用来指示校准值的状态，当配置 ERTC 精密校准寄存器 (ERTC\_SCAL) 时，CALUPDF 标志位将置 1，指示校准值正在更新；当校准值被成功应用后，标志位自动清 0，指示校准值更新完成。

### 17.3.5 参考时钟检测

为保证日历长时间运行的精确性，ERTC 提供了时钟同步功能（低功耗模式不可用），用精度更高的参考时钟（一般用 50Hz 或者 60Hz 的市电）校准更新日历的 1Hz 时钟。

参考时钟检测功能开启后，在每次更新日历值的前 7 个 ck\_a 周期检测参考时钟边沿，若检测到边沿，将使用此边沿更新日历值，后续采用 3 个 ck\_a 周期检测参考时钟边沿。每一次检测到参考时钟边沿时，都会将分频器 A 的值进行重载，这会使得内部 1Hz 的日历时钟与参考时钟边沿刚好对齐，当内部 1Hz 时钟出现微小偏移时，利用更精确的参考时钟，将 1Hz 时钟微调至与参考时钟边沿对齐。当没有检测到参考时钟边沿时，ERTC 会利用原来的时钟源更新日历。

需要注意的是，使能参考时钟功能后，需要将 DIVA、DIVB 设置为复位值 (0x7F、0xFF)，并且时钟同步功能不能与粗校准功能同时开启。

### 17.3.6 时间戳

时间戳功能用于在发生时间戳事件时（入侵引脚检测到有效边沿），将当前的日历值保存到时间戳寄存器中。

当发生时间戳时，TSF 位置 1，此时若再次发生时间戳事件，TSOF 标志位将置 1，但时间戳寄存器并不会更新，可以通过 TSIEN 位设置是否使能时间戳中断。

时间戳用法有两种

1. 单独的时间戳功能，此时入侵检测引脚用来检测时间戳，使用步骤：
  - 选择上升沿还是下降沿触发（设置 TSEDG）
  - 使能时间戳（设置 TSEN=1）
2. 发生入侵事件时保存时间戳，使用步骤：

- 配置入侵检测相关寄存器
- 使能发生入侵事件时保存时间戳（设置 TPTSEN=1）

注：发生时间戳事件后，TSF 在两个 ck\_a 周期后置 1，建议在 TSF 已置 1 的情况下轮询 TSOF 位

### 17.3.7 入侵检测

ERTC 提供了一组入侵检测 TAMP1，可配置为滤波后的电平检测或边沿检测。TAMP1 连接到入侵引脚 ERTC\_MUX1。

当检测到有效的入侵事件后，TP1F 将置 1，若已使能了入侵检测中断，将产生对应的中断；若 TPTSEN 位已置 1，将同时产生时间戳事件。为保证位于电池供电域中的电池供电寄存器数据安全，入侵事件发生时将复位电池供电寄存器。

**边沿入侵检测配置步骤：**

1. 选择入侵检测方式为边沿检测 (TPFLT=00)，并选择有效沿 (TP1EDG)。
2. 根据需要配置是否在入侵事件时激活时间戳 (TPTSEN 置 1)。
3. 根据需要开启入侵中断使能 (TPIEN 置 1)。
4. 将 TAMP1 使能 (TP1EN 置 1)。

**电平入侵检测配置步骤：**

1. 选择入侵检测方式为电平检测，并选择有效电平采样次数 (TPFLT≠00)。
2. 选择入侵有效电平 (TP1EDG)。
3. 选择入侵检测采样频率 (TPFREQ)。
4. 根据需要开启入侵检测上拉 (TPPU 置 1)，若开启，还需配置上拉电阻预充电时间 (TPPR)。
5. 根据需要配置是否在入侵事件时激活时间戳 (TPTSEN 置 1)。
6. 根据需要开启入侵中断使能 (TPIEN 置 1)。
7. 将 TAMP1 使能 (TP1EN 置 1)。

当配置为边沿检测时，有以下两点要注意：

1. 在使能入侵检测前，若入侵检测已被配置为上升沿有效，且入侵检测引脚已为高电平，在使能入侵检测后会立刻产生一个入侵检测事件。
2. 在使能入侵检测前，若入侵检测已被配置为下降沿有效，且入侵检测引脚已为低电平，在使能入侵检测后会立刻产生一个入侵检测事件。

注：当 VDD 电源关闭时，入侵检测仍有效。

### 17.3.8 复用功能输出

ERTC 提供了一组复用功能输出，可以输出以下事件：

1. 校准后的时钟 (OUTSEL=0, CALOEN=1)
  - 输出 512Hz (CALOSEL=0)
  - 输出 1Hz (CALOSEL=1)
2. 闹钟A (OUTSEL=1)
3. 闹钟B (OUTSEL=2)
4. 唤醒事件 (OUTSEL=3)

当输出闹钟事件或者唤醒事件时 (OUTSEL≠0)，可以通过 OUTTYPE 选择输出类型为开漏或是推挽，可以通过 OUTP 配置输出极性。

### 17.3.9 ERTC唤醒

ERTC 可由闹钟、周期性唤醒、时间戳、入侵事件进行唤醒，使能 ERTC 中断可按以下操作配置：

1. 将 ERTC 对应中断的 EXINT 线配置为中断模式并使能，有效沿选择上升沿。
2. 使能 ERTC 中断对应的 NVIC 通道。
3. 使能对应的 ERTC 中断。

下表说明了 ERTC 时钟源、事件以及中断对唤醒低功耗模式的影响：

表 17-2 ERTC唤醒低功耗模式

时钟源	事件	唤醒 SLEEP	唤醒 DEEPSLEEP	唤醒 STANDBY
HEXT	闹钟 A	√	×	×
	闹钟 B	√	×	×
	周期性唤醒	√	×	×
	时间戳	√	×	×
	入侵事件	√	×	×
LICK	闹钟 A	√	√	√
	闹钟 B	√	√	√
	周期性唤醒	√	√	√
	时间戳	√	√	√
	入侵事件	√	√	√
LEXT	闹钟 A	√	√	√
	闹钟 B	√	√	√
	周期性唤醒	√	√	√
	时间戳	√	√	√
	入侵事件	√	√	√

表 17-3 中断控制位

中断事件	事件标志	中断使能位	EXINT 线
闹钟 A	ALAF	ALAIEN	17
闹钟 B	ALBF	ALBIEN	17
周期性唤醒	WATF	WATIEN	22
时间戳	TSF	TSIEN	21
入侵事件	TP1F	TPIEN	21

## 17.4 ERTC 寄存器

必须以字（32 位）的方式操作这些外设寄存器。

ERTC 寄存器是 16 位可寻址寄存器，具体描述如下：

表 17-4 ERTC 寄存器映像和复位值

寄存器简称	基址偏移量	复位值
ERTC_TIME	0x00	0x0000 0000
ERTC_DATE	0x04	0x0000 2101
ERTC_CTRL	0x08	0x0000 0000
ERTC_STS	0x0C	0x0000 0007
ERTC_DIV	0x10	0x007F 00FF
ERTC_WAT	0x14	0x0000 FFFF
ERTC_CCAL	0x18	0x0000 0000
ERTC_ALA	0x1C	0x0000 0000
ERTC_ALB	0x20	0x0000 0000
ERTC_WP	0x24	0x0000 0000
ERTC_SBS	0x28	0x0000 0000
ERTC_TADJ	0x2C	0x0000 0000
ERTC_TSTM	0x30	0x0000 0000
ERTC_TS DT	0x34	0x0000 000D
ERTC_TSSBS	0x38	0x0000 0000
ERTC_SCAL	0x3C	0x0000 0000
ERTC_TAMP	0x40	0x0000 0000
ERTC_ALASBS	0x44	0x0000 0000

ERTC_ALBSBS	0x48	0x0000 0000
ERTC_BPRx	0x50-0x9C	0x0000 0000

### 17.4.1 ERTC时间寄存器(ERTC\_TIME)

域	简称	复位值	类型	功能
位 31: 23	保留	0x000	resd	保持默认值。
位 22	AMPM	0x0	rw	上午/下午 (AM/PM) 0: 上午; 1: 下午。 注: 该位只用于 12 小时制, 24 小时制保持为 0。
位 21: 20	HT	0x0	rw	小时十位 (Hour tens)
位 19: 16	HU	0x0	rw	小时个位 (Hour units)
位 15	保留	0x0	resd	保持默认值。
位 14: 12	MT	0x0	rw	分钟十位 (Minute tens)
位 11: 8	MU	0x0	rw	分钟个位 (Minute units)
位 7	保留	0x0	resd	保持默认值
位 6: 4	ST	0x0	rw	秒钟十位 (Second tens)
位 3: 0	SU	0x0	rw	秒钟个位 (Second units)

### 17.4.2 ERTC日期寄存器(ERTC\_DATE)

域	简称	复位值	类型	功能
位 31: 24	保留	0x00	resd	保持默认值。
位 23: 20	YT	0x0	rw	年份十位 (Year tens)
位 19: 16	YU	0x0	rw	年份个位 (Year units)
位 15: 13	WK	0x1	rw	星期 (Week) 0: 禁用; 1: 星期一; 2: 星期二; 3: 星期三; 4: 星期四; 5: 星期五; 6: 星期六; 7: 星期日。
位 12	MT	0x0	rw	月份十位 (Month tens)
位 11: 8	MU	0x1	rw	月份个位 (Month units)
位 7: 6	保留	0x0	resd	保持默认值。
位 5: 4	DT	0x0	rw	日期十位 (Date tens)
位 3: 0	DU	0x1	rw	日期个位 (Date units)

### 17.4.3 ERTC控制寄存器(ERTC\_CTRL)

域	简称	复位值	类型	功能
位 31: 24	保留	0x00	resd	保持默认值。
位 23	CALOEN	0x0	rw	校准输出使能 (Calibration output enable) 0: 关闭; 1: 开启。
位 22: 21	OUTSEL	0x0	rw	输出源选择 (Output source selection) 00: 关闭; 01: 闹钟 A; 10: 闹钟 B; 11: 唤醒事件。
位 20	OUTP	0x0	rw	输出极性 (Output polarity) 0: 高; 1: 低。
位 19	CALOSEL	0x0	rw	校准输出选择 (Calibration output selection) 0: 512Hz; 1: 1Hz。
位 18	BPR	0x0	rw	电池供电域数据寄存器 (Battery power domain data register)

				该位在电池供电域，不受系统复位影响，可用来存储夏令时操作或者一些其他需要一直保存的数据。
位 17	DEC1H	0x0	wo	减少 1 小时 (Decrease 1 hour) 0: 无作用; 1: 减少一小时。 注：当小时不为 0 时才有效，该位置 1 后（不要在小时递增时置 1），下一秒生效。
位 16	ADD1H	0x0	wo	增加 1 小时 (Add 1 hour) 0: 无作用; 1: 增加一小时。 注：该位置 1 后（不要在小时递增时置 1），下一秒生效。
位 15	TSIEN	0x0	rw	时间戳中断使能 (Timestamp interrupt enable) 0: 关闭; 1: 开启。
位 14	WATIEN	0x0	rw	唤醒定时器中断使能 (Wakeup timer interrupt enable) 0: 关闭; 1: 开启。
位 13	ALBIEN	0x0	rw	闹钟 B 中断使能 (Alarm B interrupt enable) 0: 关闭; 1: 开启。
位 12	ALAIEN	0x0	rw	闹钟 A 中断使能 (Alarm A interrupt enable) 0: 关闭; 1: 开启。
位 11	TSEN	0x0	rw	时间戳使能 (Timestamp enable) 0: 关闭; 1: 开启。
位 10	WATEN	0x0	rw	唤醒定时器使能 (Wakeup timer enable) 0: 关闭; 1: 开启。
位 9	ALBEN	0x0	rw	闹钟 B 使能 (Alarm B enable) 0: 关闭; 1: 开启。
位 8	ALAEN	0x0	rw	闹钟 A 使能 (Alarm A enable) 0: 关闭; 1: 开启。
位 7	CCALEN	0x0	rw	粗略校准使能 (Coarse calibration enable) 0: 关闭; 1: 开启。
位 6	HM	0x0	rw	小时模式 (Hour mode) 0: 24 小时制; 1: 12 小时制。
位 5	DREN	0x0	rw	日期/时间寄存器直接读取使能 (Date/time register direct read enable) 0: 关闭，ERTC_TIME、ERTC_DATE、ERTC_SBS 值从同步寄存器获取，每两个 ERTC_CLK 更新一次; 1: 开启，ERTC_TIME、ERTC_DATE、ERTC_SBS 值从电池供电域获取。
位 4	RCDEN	0x0	rw	参考时钟检测使能 (Reference clock detection enable) 0: 关闭; 1: 开启。
位 3	TSEDG	0x0	rw	时间戳触发边沿 (Timestamp trigger edge) 0: 上升沿; 1: 下降沿。
位 2: 0	WATCLK	0x0	rw	唤醒定时器时钟选择 (Wakeup timer clock selection) 000: ERTC_CLK/16; 001: ERTC_CLK/8; 010: ERTC_CLK/4; 011: ERTC_CLK/2; 10x: ck_a; 11x: ck_a, 唤醒计数值增加 $2^{16}$ , 唤醒时间 = ERTC_WAT + $2^{16}$ 。

注：在 WATEN=0 且 WATWF=1 时可对这些位进行写操作。

#### 17.4.4 ERTC初始化和状态寄存器(ERTC\_STS)

域	简称	复位值	类型	功能
位 31: 17	保留	0x0000	resd	保持默认值。
位 16	CALUPDF	0x0	ro	校准值更新完成标志 (Calibration value update completed flag) 0: 更新完成; 1: 正在更新。 当对 ERTC 精密校准寄存器 (ERTC_SCAL) 写时，该位自动置 1，当 ERTC 使用新的校准值时，该位自动清零，当该位为 1 时，不能写 ERTC 精密校准寄存器 (ERTC_SCAL)。
位 15: 14	保留	0x0	resd	保持默认值。
位 13	TP1F	0x0	rw0c	入侵检测 1 标志 (Tamper detection 1 flag) 0: 无入侵事件发生; 1: 有入侵事件发生。
位 12	TSOF	0x0	rw0c	时间戳溢出标志 (Timestamp overflow flag) 0: 正常; 1: 溢出。 当产生了时间戳事件 (TSF 置 1) 时，又产生了时间戳事件，该标志置 1。
位 11	TSF	0x0	rw0c	时间戳标志 (Timestamp flag) 0: 无时间戳事件发生; 1: 有时间戳事件发生。 当读取了时间戳，并清除了 TSF 时，建议再检查 TSOF 标志，因为可能会存在当正在清除 TSF 时又产生了时间戳事件。 注：该位清 0 后 2 个 APB_CLK 后生效。
位 10	WATF	0x0	rw0c	唤醒定时器标志 (Wakeup timer flag) 0: 无唤醒事件发生; 1: 有唤醒事件发生。 注：该位清 0 后 2 个 APB_CLK 后生效。
位 9	ALBF	0x0	rw0c	闹钟 B 标志 (Alarm clock B flag) 0: 无闹钟事件发生; 1: 有闹钟事件发生。 注：该位清 0 后 2 个 APB_CLK 后生效。
位 8	ALAF	0x0	rw0c	闹钟 A 标志 (Alarm clock A flag) 0: 无闹钟事件发生; 1: 有闹钟事件发生。 注：该位清 0 后 2 个 APB_CLK 后生效。
位 7	IMEN	0x0	rw	初始化模式使能 (Initialization mode enable) 0: 关闭; 1: 开启。 当进入了初始化模式后，日历处于停止状态。
位 6	IMF	0x0	ro	进入初始化模式标志 (Enter initialization mode flag) 0: 未进入; 1: 进入。 当使能了初始化模式 (INITEN=1)，并进入了初始化模式 (INITEF=1) 时，才能更改 ERTC_TIME、ERTC_DATE、ERTC_DIV 寄存器。
位 5	UPDF	0x0	rw0c	日历更新标志 (Calendar update flag) 0: 正在更新; 1: 更新完成。 每当从电池供电域将日历更新到 ERTC_TIME、ERTC_DATE、ERTC_SBS 同步寄存器，该标志置 1，每两个 ERTC_CLK 更新一次。
位 4	INITF	0x0	ro	日历初始化标志 (Calendar initialization flag) 0: 未初始化;

				1: 已初始化。 当检测到 ERTC_DATE 里面的年份不为 0 时该位置 1，当年份为 0 时，该位清 0。
位 3	TADJF	0x0	ro	时间调整标志 (Time adjustment flag) 0: 无操作； 1: 正在执行时间调整。 当对时间调整存器 (ERTC_TADJ) 写时，该位自动置 1，当时间调整结束后，该位自动清零。
位 2	WATWF	0x1	ro	唤醒定时器寄存器允许写标志 (Wakeup timer register allows write flag) 0: 不允许写； 1: 允许写。
位 1	ALBWF	0x1	ro	闹钟 B 允许写标志 (Alarm B register allows write flag) 0: 不允许写； 1: 允许写。
位 0	ALAWF	0x1	ro	闹钟 A 允许写标志 (Alarm A register allows write flag) 0: 不允许写； 1: 允许写。

#### 17.4.5 ERTC预分频器寄存器(ERTC\_DIV)

域	简称	复位值	类型	功能
位 31: 23	保留	0x000	resd	保持默认值。
位 22: 16	DIVA	0x7F	rw	分频器 A (Diveder A)
位 15	保留	0x0	resd	保持默认值。
位 14: 0	DIVB	0x00FF	rw	分频器 B (Diveder B) 日历时钟=ERTC_CLK/((DIVA+1)x(DIVB+1))。

#### 17.4.6 ERTC唤醒定时器寄存器(ERTC\_WAT)

域	简称	复位值	类型	功能
位 31: 16	保留	0x0000	resd	保持默认值。
位 15: 0	VAL	0xFFFF	rw	唤醒定时器重载值 (Wakeup timer reload value)

#### 17.4.7 ERTC粗校准寄存器(ERTC\_CCAL)

域	简称	复位值	类型	功能
位 31: 8	保留	0x000000	resd	保持默认值。
位 7	CALDIR	0x0	rw	校准方向 (Calibration direction) 0: 正校准； 1: 负校准。
位 6: 5	保留	0x0	resd	保持默认值。
位 4: 0	CALVAL	0x00	rw	校准值 (Calibration value) 正校准： 00000: +0 ppm 00001: +4 ppm 00010: +8 ppm 11111: +126 ppm 负校准： 00000: -0 ppm 00001: -2 ppm 00010: -4 ppm ... 11111: -63 ppm

#### 17.4.8 ERTC闹钟A寄存器(ERTC\_ALA)

域	简称	复位值	类型	功能
位 31	MASK4	0x0	rw	日期/星期屏蔽 (Date/week mask) 0: 无屏蔽； 1: 闹钟和日期/星期无关。
位 30	WKSEL	0x0	rw	日期/星期选择 (Date/week mode select)

				0: 日期; 1: 星期 (DT[1: 0]不使用)。
位 29: 28 DT	0x0	rw	日期十位 (Date tens)	
位 27: 24 DU	0x0	rw	日期/星期个位 (Date/week units)	
			小时屏蔽 (Hour mask)	
位 23 MASK3	0x0	rw	0: 无屏蔽; 1: 闹钟和小时无关。	
			上午/下午 (AM/PM)	
位 22 AMPM	0x0	rw	0: 上午; 1: 下午。 注: 该位只用于 12 小时制, 24 小时制保持为 0。	
位 21: 20 HT	0x0	rw	小时十位 (Hour tens)	
位 19: 16 HU	0x0	rw	小时个位 (Hour units)	
			分钟屏蔽 (Minute mask)	
位 15 MASK2	0x0	rw	0: 无屏蔽; 1: 闹钟和分钟无关。	
位 14: 12 MT	0x0	rw	分钟十位 (Minute tens)	
位 11: 8 MU	0x0	rw	分钟个位 (Minute units)	
			秒钟屏蔽 (Second mask)	
位 7 MASK1	0x0	rw	0: 无屏蔽; 1: 闹钟和秒钟无关。	
位 6: 4 ST	0x0	rw	秒钟十位 (Second tens)	
位 3: 0 SU	0x0	rw	秒钟个位 (Second units)	

#### 17.4.9 ERTC闹钟B寄存器(ERTC\_ALB)

域	简称	复位值	类型	功能
位 31	MASK4	0x0	rw	日期/星期屏蔽 (Date/week mask) 0: 无屏蔽; 1: 闹钟和日期/星期无关。
位 30	WKSEL	0x0	rw	日期/星期选择 (Date/week mode select) 0: 日期; 1: 星期 (DT[1: 0]不使用)。
位 29: 28 DT	0x0	rw	日期十位 (Date tens)	
位 27: 24 DU	0x0	rw	日期/星期个位 (Date/week units)	
			小时屏蔽 (Hour mask)	
位 23 MASK3	0x0	rw	0: 无屏蔽; 1: 闹钟和小时无关。	
			上午/下午 (AM/PM)	
位 22 AMPM	0x0	rw	0: 上午; 1: 下午。 注: 该位只用于 12 小时制, 24 小时制保持为 0。	
位 21: 20 HT	0x0	rw	小时十位 (Hour tens)	
位 19: 16 HU	0x0	rw	小时个位 (Hour units)	
			分钟屏蔽 (Minute mask)	
位 15 MASK2	0x0	rw	0: 无屏蔽; 1: 闹钟和分钟无关。	
位 14: 12 MT	0x0	rw	分钟十位 (Minute tens)	
位 11: 8 MU	0x0	rw	分钟个位 (Minute units)	
			秒钟屏蔽 (Second mask)	
位 7 MASK1	0x0	rw	0: 无屏蔽; 1: 闹钟和秒钟无关。	
位 6: 4 ST	0x0	rw	秒钟十位 (Second tens)	
位 3: 0 SU	0x0	rw	秒钟个位 (Second units)	

#### 17.4.10 ERTC写保护寄存器(ERTC\_WP)

域	简称	复位值	类型	功能
位 31: 8 保留		0x000000	resd	保持默认值。
位 7: 0 CMD		0x00	wo	命令寄存器 (Command register)

依次写入 0xCA、0x53 解锁所有 ERTC 寄存器写保护，当写一个其他值时，将重新开启写保护。

#### 17.4.11 ERTC 亚秒寄存器(ERTC\_SBS)

域	简称	复位值	类型	功能
位 31: 16 保留		0x0000	resd	保持默认值。
位 15: 0 SBS		0x0000	ro	亚秒值 (Sub-second value) 亚秒为分频器 DIVB 的计数值，时钟频率为 ERTC_CLK/(DIVA+1)。

#### 17.4.12 ERTC 时间微调寄存器(ERTC\_TADJ)

域	简称	复位值	类型	功能
位 31	ADD1S	0x0	wo	增加一秒 (Add 1 second) 0: 无效果； 1: 增加 1 秒。 当 TADJF=0 时，才能写此寄存器，通常 ADD1S 与 DECSBS 配合使用，达到微调时间的效果。
位 30: 15 保留		0x0000	resd	保持默认值。
位 14: 0 DECSBS		0x0000	wo	DECSBS[14: 0]: 减少亚秒值 (Decrease sub-second value) 延迟时间 (ADD1S=0)：延迟=DECSBS/(DIVB+1)； 提前时间 (ADD1S=1)：延迟=1-(DECSBS/(DIVB+1))。

#### 17.4.13 ERTC 时间戳时间寄存器(ERTC\_TSTM)

域	简称	复位值	类型	功能
位 31: 23 保留		0x000	resd	保持默认值。
位 22	AMPM	0x0	ro	上午/下午 (AM/PM) 0: 上午； 1: 下午。 注：该位只用于 12 小时制，24 小时制保持为 0。
位 21: 20 HT		0x0	ro	小时十位 (Hour tens)
位 19: 16 HU		0x0	ro	小时个位 (Hour units)
位 15	保留	0x0	resd	保持默认值
位 14: 12 MT		0x0	ro	分钟十位 (Minute tens)
位 11: 8 MU		0x0	ro	分钟个位 (Minute units)
位 7	保留	0x0	resd	保持默认值。
位 6: 4 ST		0x0	ro	秒钟十位 (Second tens)
位 3: 0 SU		0x0	ro	秒钟个位 (Second units)

注意：仅当 ERTC 初始化和状态寄存器 (ERTC\_STS) 中的 TSF 置 1 时，该寄存器的内容才有效。当 TSF 位复位时，清零该寄存器。

#### 17.4.14 ERTC 时间戳日期寄存器(ERTC\_TSDT)

域	简称	复位值	类型	功能
位 31: 16 保留		0x0000	resd	保持默认值。
位 15: 13 WK		0x0	ro	星期 (Week)
位 12	MT	0x0	ro	月十位 (Month tens)
位 11: 8 MU		0x0	ro	月个位 (Month units)
位 7: 6 保留		0x0	resd	保持默认值
位 5: 4 DT		0x0	ro	日期十位 (Date tens)
位 3: 0 DU		0x0	ro	日期个位 (Date units)

注意：仅当 ERTC 初始化和状态寄存器 (ERTC\_STS) 中的 TSF 置 1 时，该寄存器的内容才有效。当 TSF 位复位时，清零该寄存器。

#### 17.4.15 ERTC 时间戳亚秒寄存器(ERTC\_TSSBS)

域	简称	复位值	类型	功能
位 31: 16 保留		0x0000	resd	保持默认值。
位 15: 0 SBS		0x0000	ro	亚秒值 (Sub-second value)

注意：仅当 ERTC 初始化和状态寄存器 (ERTC\_STS) /TSF 置 1 时，该寄存器的内容才有效。当 ERTC 初始化和状态寄存器 (ERTC\_STS) /TSF 位复位时，清零该寄存器。

#### 17.4.16 ERTC精密校准寄存器(ERTC\_SCAL)

域	简称	复位值	类型	功能
位 31: 16	保留	0x0000	resd	保持默认值。
位 15	ADD	0x0	rw	增加 ERTC 时钟 (Add ERTC clock) 0: 无操作; 1: 每 $2^{11}$ 个 ERTC_CLK, 插入一个 ERTC_CLK。
位 14	CAL8	0x0	rw	8 秒校准周期 (8-second calibration period) 0: 无效果; 1: 8 秒校准周期。
位 13	CAL16	0x0	rw	16 秒校准周期 (16 second calibration period) 0: 无效果; 1: 16 秒校准周期。
位 12: 9	保留	0x0	resd	保持默认值
位 8: 0	DEC	0x000	rw	减少 ERTC 时钟 (Decrease ERTC clock) 在 $2^{20}$ 个 ERTC_CLK 周期内, 屏蔽 DEC 个 ERTC_CLK。 通常和 ADD 配合使用, 当 ADD 为 1 时, 在 $2^{20}$ 个 ERTC_CLK 周期内, 实际的 ERTC_CLK 个数为 $2^{20}+512-DEC$ 。

#### 17.4.17 ERTC入侵配置寄存器(ERTC\_TAMP)

域	简称	复位值	类型	功能
位 31: 19	保留	0x0000	resd	保持默认值。
位 18	OUTTYPE	0x0	rw	输出类型 (Output type) 0: 开漏; 1: 推挽。
位 17: 16	保留	0x0	resd	保持默认值
位 15	TPPU	0x0	rw	入侵检测上拉 (Tamper detection pull-up) 0: 开启; 1: 关闭。
位 14: 13	TPPR	0x0	rw	入侵检测预充电时间 (Tamper detection pre-charge time) 0: 1 个 ERTC_CLK; 1: 2 个 ERTC_CLK; 2: 4 个 ERTC_CLK; 3: 8 个 ERTC_CLK。
位 12: 11	TPFLT	0x0	rw	入侵检测滤波时间 (Tamper detection filter time) 0: 无滤波; 1: 连续 2 次采样有效, 判定入侵事件发生; 2: 连续 4 次采样有效, 判定入侵事件发生; 3: 连续 8 次采样有效, 判定入侵事件发生。
位 10: 8	TPFREQ	0x0	rw	入侵检测检测频率 (Tamper detection frequency) 0: ERTC_CLK/32768; 1: ERTC_CLK/16384; 2: ERTC_CLK/8192; 3: ERTC_CLK/4096; 4: ERTC_CLK/2048; 5: ERTC_CLK/1024; 6: ERTC_CLK/512; 7: ERTC_CLK/256。
位 7	TPTSEN	0x0	rw	入侵检测时间戳使能 (Tamper detection timestamp enable) 0: 关闭; 1: 开启, 当产生入侵事件时, 保持时间戳。
位 6: 3	保留	0x0	resd	保持默认值。
位 2	TPIEN	0x0	rw	入侵检测中断使能 (Tamper detection interrupt enable) 0: 关闭;

				1: 开启。 入侵检测 1 有效边沿 (Tamper detection 1 valid edge) 当无滤波时 (TPFLT=0)： 0: 上升沿; 1: 下降沿。 当有滤波时 (TPFLT>0)： 0: 低电平; 1: 高电平。
位 1	TP1EDG	0x0	rw	入侵检测 1 使能 (Tamper detection 1 enable) 0: 关闭; 1: 开启。

#### 17.4.18 ERTC闹钟A亚秒寄存器(ERTC\_ALASBS)

域	简称	复位值	类型	功能
位 31: 28 保留		0x0	resd	保持默认值。
位 27: 24 SBSMSK		0x0	rw	亚秒屏蔽 (Sub-second mask) 0: 不匹配亚秒, 闹钟与亚秒无关; 1: 只匹配 SBS[0]; 2: 只匹配 SBS[1: 0]; 3: 只匹配 SBS[2: 0]; ... 14: 只匹配 SBS[13: 0]; 15: 匹配 SBS[14: 0]。
位 23: 15 保留		0x000	rw	保持默认值。
位 14: 0 SBS		0x0000	rw	亚秒值 (Sub-second value)

#### 17.4.19 ERTC闹钟B亚秒寄存器(ERTC\_ALBSBS)

域	简称	复位值	类型	功能
位 31: 28 保留		0x0	resd	保持默认值。
位 27: 24 SBSMSK		0x0	rw	亚秒屏蔽 (Sub-second mask) 0: 不匹配亚秒, 闹钟与亚秒无关; 1: 只匹配 SBS[0]; 2: 只匹配 SBS[1: 0]; 3: 只匹配 SBS[2: 0]; ... 14: 只匹配 SBS[13: 0]; 15: 匹配 SBS[14: 0]。
位 23: 15 保留		0x000	rw	保持默认值。
位 14: 0 SBS		0x0000	rw	亚秒值 (Sub-second value)

#### 17.4.20 ERTC电池供电数据寄存器(ERTC\_BPRx)

域	简称	复位值	类型	功能
位 31: 0 DT		0x0000 0000	rw	电池供电域数据 (Battery powered domain data) BPR_DT <sub>x</sub> 寄存器, 可以在只由电池供电下保存数据, 不会被系统复位所复位, 只能通过电池供电域复位或入侵事件进行复位。

# 18 模拟/数字转换 (ADC)

## 18.1 ADC简介

ADC 是一个将模拟输入信号转换为 12 位数字信号的外设。采样率最高可达 2MSPS。多达 10 个通道源可进行采样及转换。

## 18.2 ADC主要特征

模拟方面有以下特征：

- 支持分辨率 12 位的转换
- 自校准时间： 172 个 ADC 时钟周期
- ADC 转换时间
- ADC 时钟在最大频率 28MHz 时转换时间为 0.5  $\mu$ s
- ADC 供电要求：请参考 Datasheet
- ADC 输入范围： $V_{REF-} \leq V_{IN} \leq V_{REF+}$

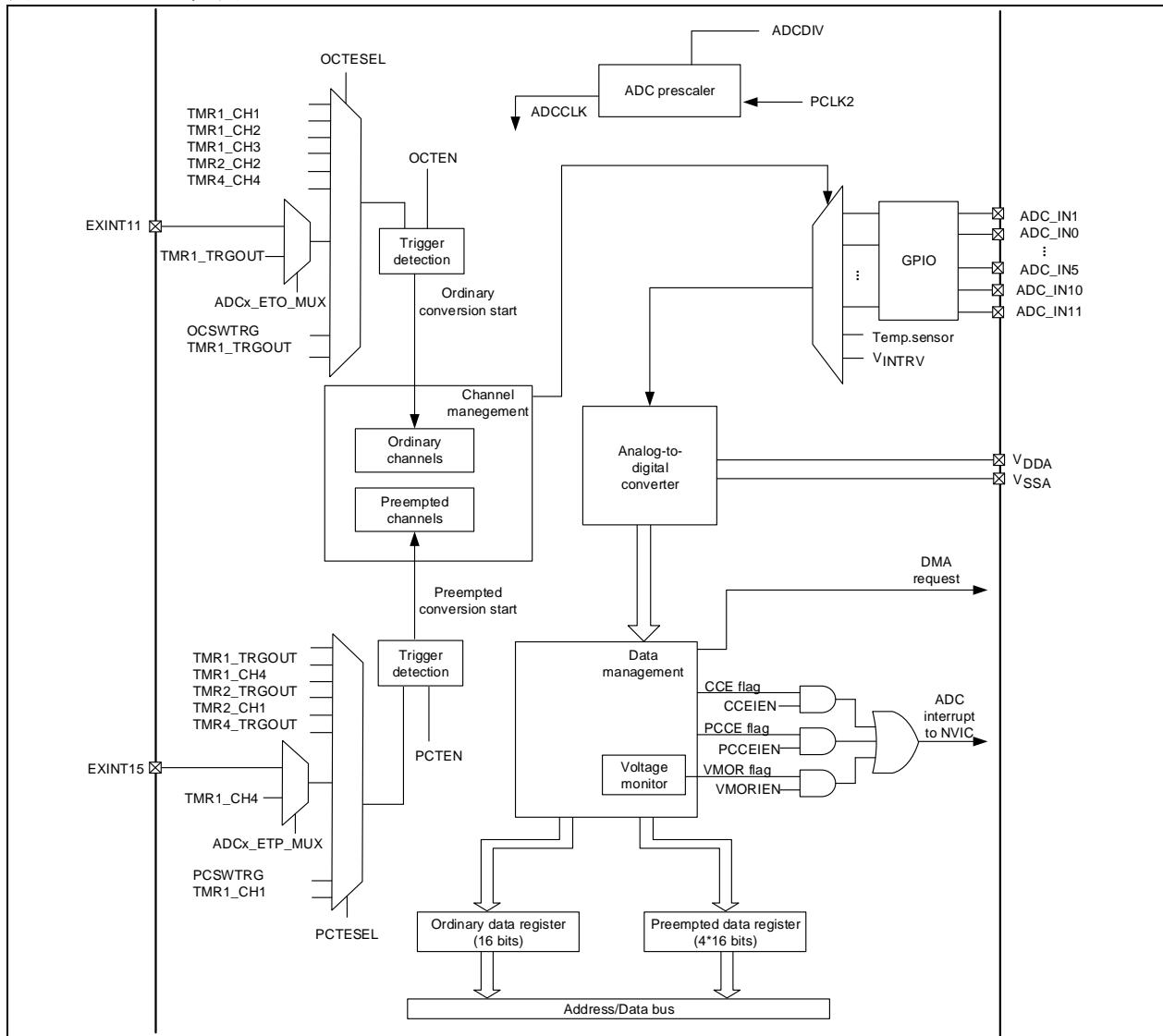
数字控制方面有以下特征：

- 通道管理区分优先权不同的普通通道与抢占通道
- 普通通道与抢占通道具备各自独立的触发侦测电路
- 各通道均可独立配置采样时间
- 转换顺序管理支持多种不同的多通道转换
- 可选择的数据对齐方式
- 可配置的电压监测边界
- 支持 DMA 传输的普通通道数据
- 可设定以下事件发生时响应中断
  - 抢占通道组转换结束
  - 通道转换结束
  - 电压监测超出范围

## 18.3 ADC架构

ADC1的架构如下图所示：

图 18-1 ADC1框图



输入管脚介绍：

- V<sub>DDA</sub>：模拟电源，ADC 模拟电源
- V<sub>SSA</sub>：模拟电源地，ADC 模拟电源地
- ADC<sub>INx</sub>：模拟输入信号通道

输入管脚的连接与电压范围限制请参考 Datasheet

## 18.4 ADC功能介绍

### 18.4.1 通道管理

#### 模拟信号通道输入

每个 ADC 拥有多达 8 个模拟信号通道输入，以 ADC<sub>INx</sub> 表示，x=0 至 5、10、11。

- ADC1\_IN0 至 ADC1\_IN5、ADC1\_IN10、ADC1\_IN11 为外部模拟输入，ADC1\_IN16 为内部温度传感器，ADC1\_IN17 为内部参考电压。

#### 通道转换

转换区分为普通通道转换与抢占通道转换，抢占通道的转换优先权高于普通通道。

抢占通道触发若发生于普通通道转换途中，优先进行抢占通道的转换，普通通道于抢占通道转换结束后重新开始转换被打断的通道。普通通道触发若发生于抢占通道转换途中，普通通道的转换会等待抢占通道转

换完成后才开始。

将通道 (ADC\_INx) 编排进普通通道序列 (ADC\_OSQx) 以及抢占通道序列 (ADC\_PSQ)，相同通道可重复编排，序列总数由 OCLEN 与 PCLEN 定义，接着即可启动普通通道转换或抢占通道转换。

#### 18.4.1.1 内部温度传感器

温度传感器接到 ADC1\_IN16，必须先使能 ADC 控制寄存器 2 (ADC\_CTRL2) 的 ITSRVEN 位并且等待上电时间后才可对温度传感通道进行转换。

转换后获得的数据，搭配数据手册的电气特性章节提供的 25° C 的电压值与数据对温度斜率 (Avg\_Slope)，即可推算温度。

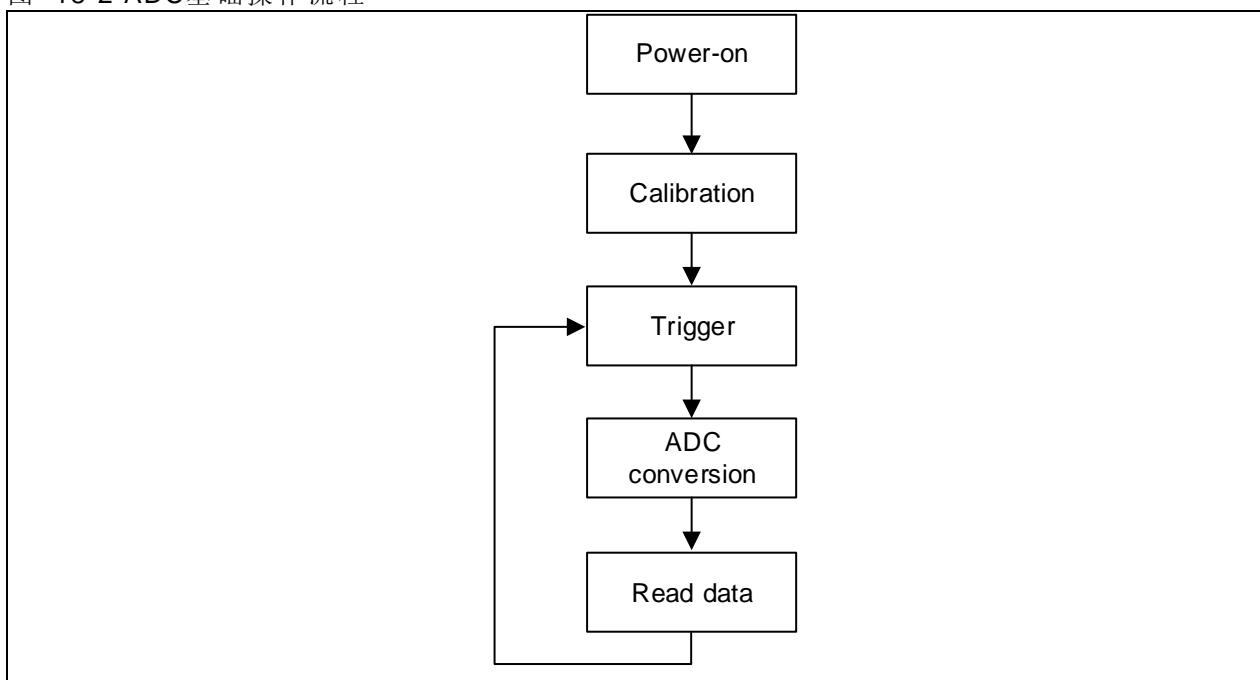
#### 18.4.1.2 内部参考电压

典型值 1.2V 的内部参考电压接到 ADC1\_IN17，必须先使能 ADC 控制寄存器 2 (ADC\_CTRL2) 的 ITSRVEN 位后才可对内部参考电压通道进行转换。此通道的转换数据可用于推算外部参考电压。

### 18.4.2 ADC操作流程

ADC 的基础操作流程如图所示，建议第一次上电后进行校准，以提升采样与转换准确度。待校准完成后可靠触发引起 ADC 采样转换，转换结束后即可读取数据。

图 18-2 ADC基础操作流程



#### 18.4.2.1 上电与校准

##### 上电

用户须先使能 APB2 外设时钟使能寄存器 (CRM\_APB2EN) 的 ADC1EN，以使能 ADC 的时钟：PCLK2 与 ADCCLK。

时钟使能后必须配置 ADC 预分频器(CRM\_CFG 的 ADCDIV)，将 ADCCLK 调整至需求的频率。ADCCLK 由 PCLK2 除频而来。

**注意：** ADCCLK 不可大于 28MHz。

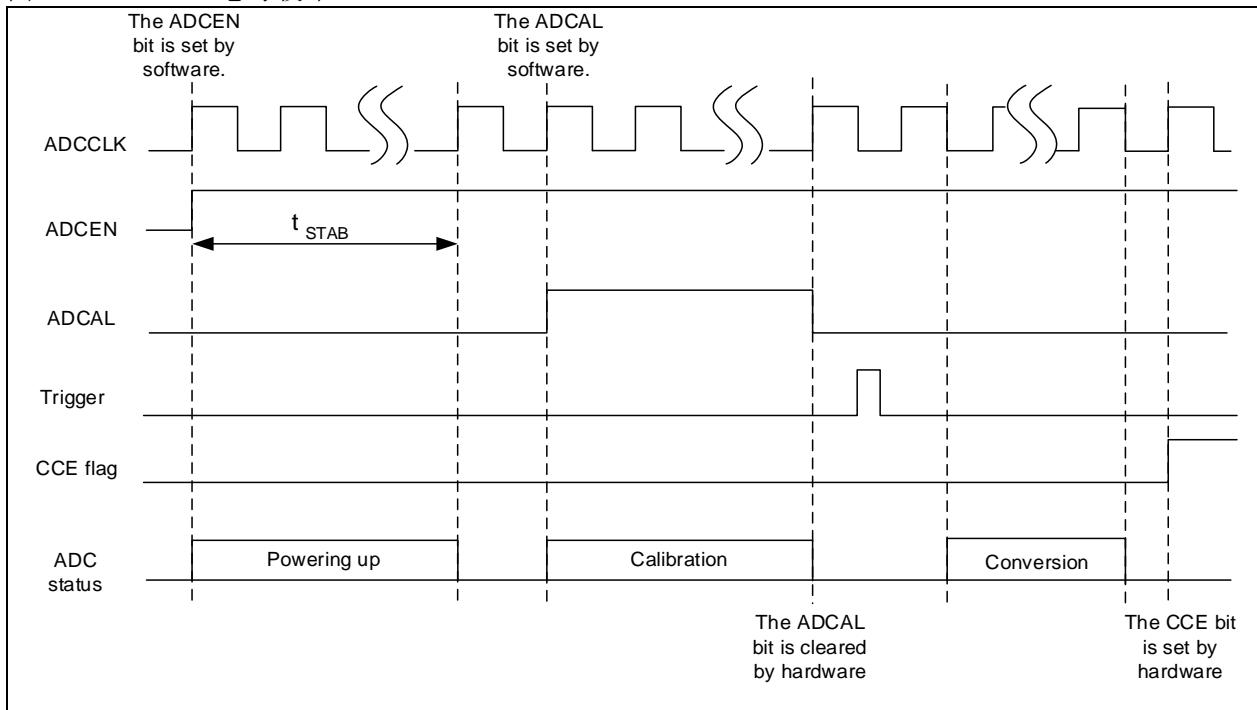
ADCCLK 频率调整完后，即可使能 ADC 控制寄存器 2 (ADC\_CTRL2) 的 ADCEN 位使 ADC 上电，等待 t<sub>STAB</sub> 后才可对 ADC 进行后续操作。清除 ADCEN 会使 ADC 的转换中止并复位，同时 ADC 被断电以达到省电的效果。

##### 校准

上电完成后可设置 ADC 控制寄存器 2 (ADC\_CTRL2) 的 ADCAL 使 ADC 进行校准，校准完成后硬件清除 ADCAL 位，软件即可触发以进行转换。

每次校准后，校准值会被存放至 ADC 普通数据寄存器 (ADC\_ODT) 中，这个校准值自动反馈回 ADC 内部，以消除电容误差。该校准值的存放不会置位 CCE 标志，不会产生中断或 DMA 请求。

图 18-3 ADC上电与校准



#### 18.4.2.2 触发

ADC 触发分为普通通道触发与抢占通道触发，普通通道触发引发普通通道转换，抢占通道触发引发抢占通道转换。使能 ADC 控制寄存器 2 (ADC\_CTRL2) 的 OCTEN 或 PCTEN 后，ADC 才会检测触发来源的上升沿并响应转换。

触发来源可分为软件写寄存器触发 (ADC 控制寄存器 2 (ADC\_CTRL2) 的 OCSWTRG 与 PCSWTRG) 以及外部触发，外部触发包含定时器触发与管脚触发，由 ADC 控制寄存器 2 (ADC\_CTRL2) 的 OCTESEL 与 PCTESEL 选择触发来源，如下表所示。普通通道还有一种特殊的触发来源，即重复使能 ADCEN 触发转换。此种情况下不需要使能 ADC 控制寄存器 2 (ADC\_CTRL2) 的 OCTEN 也可导致普通通道响应转换。

表 18-1 ADC触发来源

OCTESEL	触发来源	PCTESEL	触发来源
0000	TMR1_CH1 event	0000	TMR1_TRGOUT event
0001	TMR1_CH2 event	0001	TMR1_CH4 event
0010	TMR1_CH3 event	0010	TMR2_TRGOUT event
0011	TMR2_CH2 event	0011	TMR2_CH1 event
0100	保留	0100	保留
0101	TMR4_CH4 event	0101	TMR4_TRGOUT event
0110	ADC1_ETO_MUX=0 EXINT line11 external pin	0110	ADC1_ETP_MUX=0 EXINT line15 external pin
	ADC1_ETO_MUX=1 TMR1_TRGOUT event		TMR1_CH4 event
0111	OCSWTRG bit	0111	PCSWTRG bit
1000	保留	1000	保留
1001	保留	1001	保留
1010	保留	1010	保留
1011	保留	1011	保留
1100	保留	1100	保留
1101	TMR1_TRGOUT event	1101	TMR1_CH1 event
1110	保留	1110	保留
1111	保留	1111	保留

### 18.4.2.3 采样与转换时序

用户可于 ADC 采样时间寄存器 1 (ADC\_SPT1) 与 ADC 采样时间寄存器 2 (ADC\_SPT2) 的 CSPTx 配置各个通道 (ADC\_INx) 的采样周期。一次转换所需的时间可利用以下公式推得：

$$\text{一次转换所需的时间(ADCCLK 的周期)} = \text{采样时间} + 12.5$$

示例：

CSPTx 选择 1.5 周期，一次转换需要  $1.5+12.5=14$  个 ADCCLK 周期。

CSPTx 选择 7.5 周期，一次转换需要  $7.5+12.5=20$  个 ADCCLK 周期。

### 18.4.3 转换顺序管理

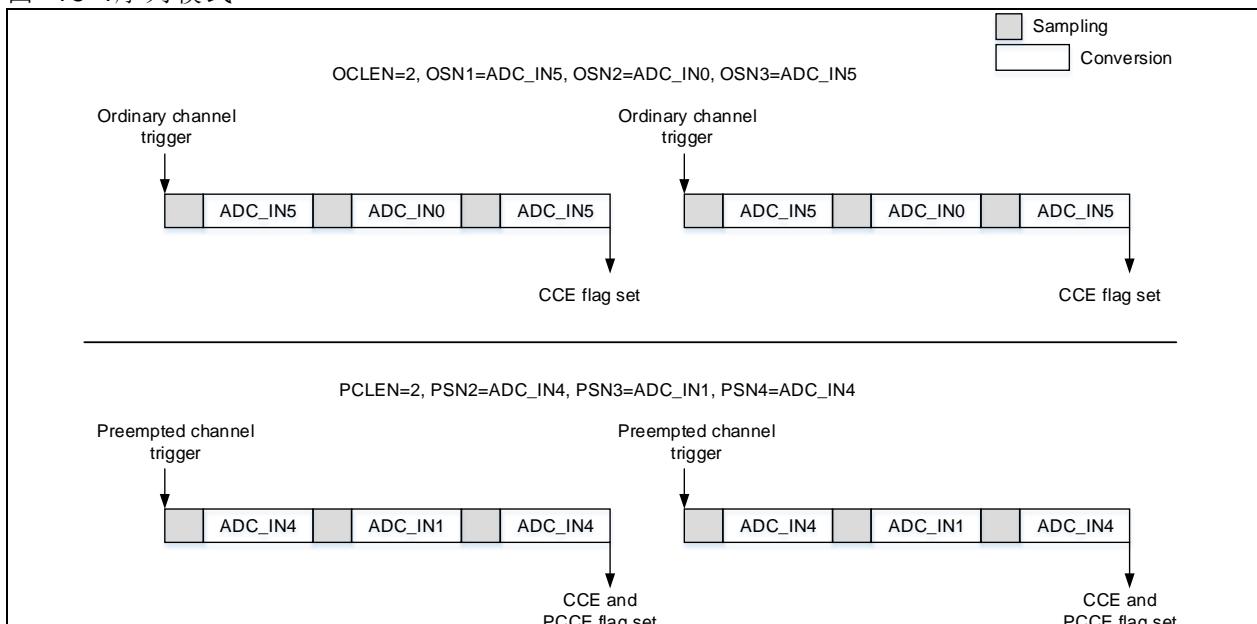
默认模式下，每次触发只会转换单个通道，即 OSN1 (普通触发) 或 PSN4 (抢占触发) 记录的通道。

下面介绍不同的转换顺序模式，即可使多个通道以特定顺序做转换。

#### 18.4.3.1 序列模式

使能 ADC 控制寄存器 1 (ADC\_CTRL1) 的 SQEN，即开启序列模式，用户于 ADC\_OSQx 配置普通通道顺序与总数，于 ADC 抢占序列寄存器 (ADC\_PSQ) 配置抢占通道顺序与总数，开启序列模式后，一次触发将序列中的通道依次转换一次。普通通道从 OSN1 开始转换起，抢占通道是从 PSN<sub>x</sub> 开始转换起， $x=4-PCLEN$ ，下图示范了序列模式的行为。

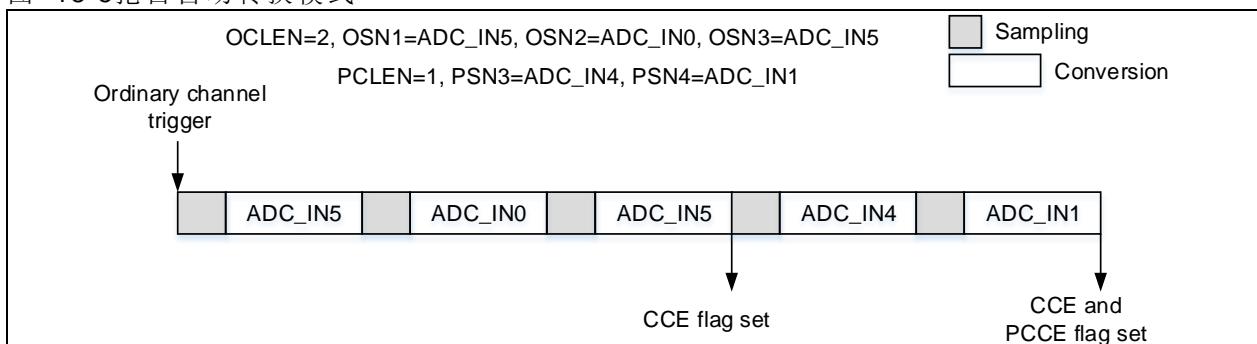
图 18-4 序列模式



#### 18.4.3.2 抢占自动转换模式

使能 ADC 控制寄存器 1 (ADC\_CTRL1) 的 PCAUTOEN，即开启抢占自动转换模式，当普通通道转换完成后，抢占通道将自动接续着转换。可与序列模式共用，当普通通道序列完成后，即会自动开始抢占序列的转换。下图示范了与序列模式共用的抢占自动转换模式行为。

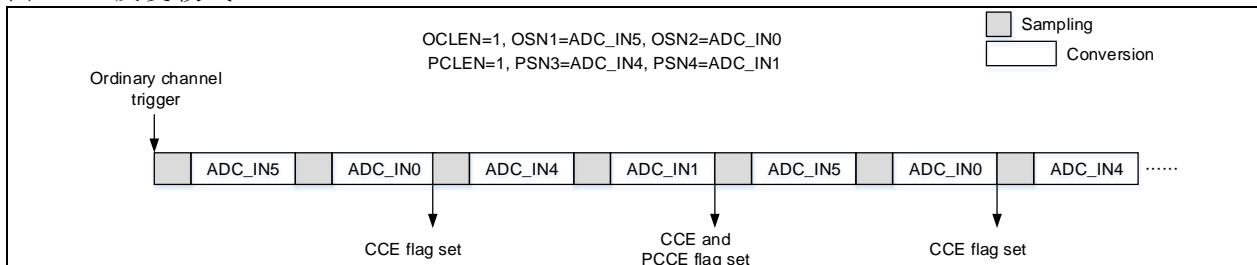
图 18-5 抢占自动转换模式



#### 18.4.3.3 反复模式

使能 ADC 控制寄存器 2 (ADC\_CTRL2) 的 RPEN，即开启反复模式。当普通通道检测到触发后就即会反复不断地转换。可与序列模式下的普通通道转换共用，将反复地转换普通通道序列。也可与抢占自动转换模式共用，将依次反复地转换普通通道序列与注入通道序列。下图示范了与序列模式及抢占自动转换模式共用的反复模式行为。

图 18-6 反复模式



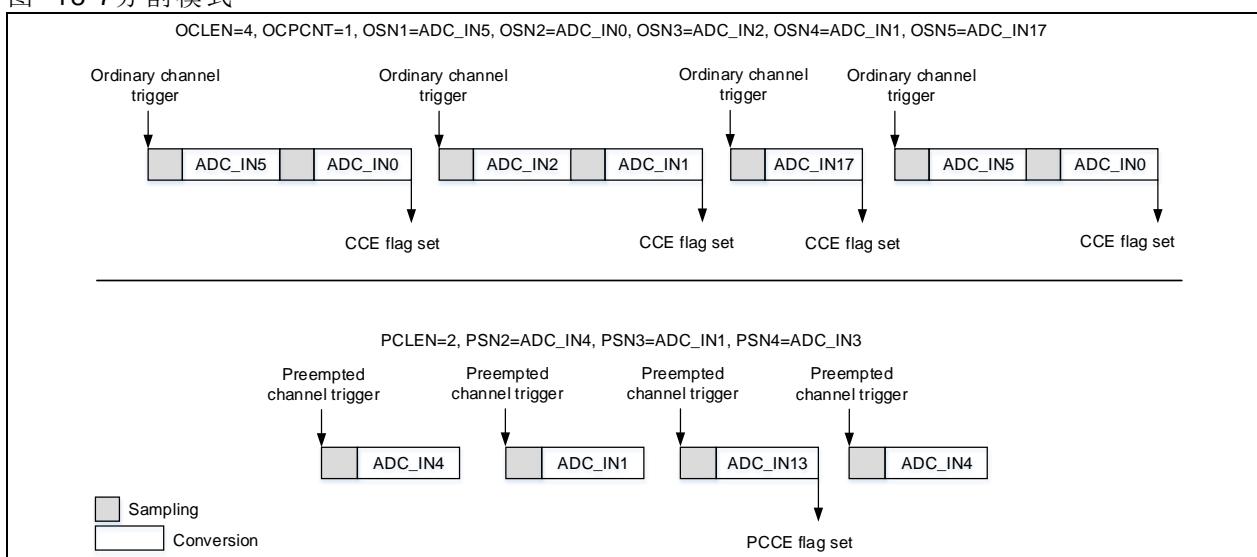
#### 18.4.3.4 分割模式

使能 ADC 控制寄存器 1 (ADC\_CTRL1) 的 OCPEN，即开启普通通道的分割模式，此模式将 ADC 普通序列寄存器 1 (ADC\_OSQ1) 的 OCLEN 的序列长度分割成长度较小的子组别，子组别的通道数于 ADC 控制寄存器 1 (ADC\_CTRL1) 的 OCPCNT 配置，一次触发将转换子组别中的所有通道。每次触发会依序选择不同的子组别。

使能 ADC 控制寄存器 1 (ADC\_CTRL1) 的 PCPEN，即开启抢占通道的分割模式，此模式将 ADC 普通序列寄存器 1 (ADC\_OSQ1) 的 PCLEN 的序列长度分割成只有一个通道的子组别，一次触发将转换子组别中的通道。每次触发会依序选择不同的子组别。

分割模式与反复模式不可共用。下图分别示范了普通分割与抢占分割模式的行为。

图 18-7 分割模式



#### 18.4.4 数据管理

普通通道转换完成后数据存储于 ADC 普通数据寄存器 (ADC\_ODT)，抢占通道转换完成后数据存储于 ADC 抢占数据寄存器 x (ADC\_PDTx)。

##### 18.4.4.1 数据内容处理

由 ADC 控制寄存器 2 (ADC\_CTRL2) 的 DTALIGN 选择转换数据靠右或是靠左对齐放置于数据寄存器，除此之外，抢占通道的数据还会减去 ADC 抢占通道数据偏移寄存器 x (ADC\_PCDTOx) 的偏移量，因此抢占通道数据有可能为负值，以 SIGN 作为符号。如下图所示。

图 18-8 数据内容处理

Ordinary channel data 12 bits															
Right-alignment								DT[11]	DT[10]	DT[9]	DT[8]	DT[7]	DT[6]	DT[5]	DT[4]
Left-alignment								DT[11]	DT[10]	DT[9]	DT[8]	DT[7]	DT[6]	DT[5]	DT[4]
Preempted channel data 12 bits															
Right-alignment								SIGN	SIGN	SIGN	SIGN	DT[11]	DT[10]	DT[9]	DT[8]
Left-alignment								SIGN	DT[11]	DT[10]	DT[9]	DT[8]	DT[7]	DT[6]	DT[5]

#### 18.4.4.2 数据获取

普通通道转换数据可藉由 CPU 或 DMA 读取 ADC 普通数据寄存器 (ADC\_ODT) 获得。抢占通道数据只可藉由 CPU 读取 ADC 抢占数据寄存器 x (ADC\_PDTx) 获得。

使能 ADC 控制寄存器 2 (ADC\_CTRL2) 的 OCDMAEN 后, ADC 会在每次 ADC 普通数据寄存器 (ADC\_ODT) 更新时请求 DMA。

#### 18.4.5 电压监测

使能 ADC 控制寄存器 1 (ADC\_CTRL1) 的 OCVMEN (普通通道) 或 PCVMEN (抢占通道) 即可通过转换结果的判定来实现电压监测。当转换结果大于高边界 ADC 电压监测高边界寄存器 (ADC\_VMHB) 或是小于低边界 ADC 电压监测低边界寄存器 (ADC\_VMLB) 时, 电压监测超出标志 VMOR 会置起。

透过 VMSGGEN 选择对单一特定通道或是所有通道监测。对单一通道监测的话, 由 VMCSEL 配置通道。电压监测一律以转换的原始数据与 12 位边界寄存器做比较, 无视 PCDTOx 与 DTALIGN 位的设定。

#### 18.4.6 状态标志与中断

每个 ADC 拥有自己的 ADC 状态寄存器 (ADCx\_STS): 普通通道转换开始标志 (OCCS)、抢占通道转换开始标志 (PCCS)、抢占通道组转换结束标志 (PCCE)、通道转换结束标志 (CCE) 及电压监测超出标志 (VMOR)。

其中抢占通道组转换结束标志、通道转换结束标志及电压监测超出标志拥有对应中断使能位, 只要将中断使能, 标志置起时便会对 CPU 发出中断。

### 18.5 ADC 寄存器

下表列出了 ADC 寄存器的映像和复位值。

必须以字 (32 位) 的方式操作这些外设寄存器。

表 18-2 ADC 寄存器映像和复位值

寄存器简称	基址偏移量	复位值
ADC_STS	0x000	0x0000 0000
ADC_CTRL1	0x004	0x0000 0000
ADC_CTRL2	0x008	0x0000 0000
ADC_SPT1	0x00C	0x0000 0000
ADC_SPT2	0x010	0x0000 0000
ADC_PCDTO1	0x014	0x0000 0000
ADC_PCDTO2	0x018	0x0000 0000
ADC_PCDTO3	0x01C	0x0000 0000
ADC_PCDTO4	0x020	0x0000 0000

ADC_VMHB	0x024	0x0000 0FFF
ADC_VMLB	0x028	0x0000 0000
ADC_OSQ1	0x02C	0x0000 0000
ADC_OSQ2	0x030	0x0000 0000
ADC_OSQ3	0x034	0x0000 0000
ADC_PSQ	0x038	0x0000 0000
ADC_PDT1	0x03C	0x0000 0000
ADC_PDT2	0x040	0x0000 0000
ADC_PDT3	0x044	0x0000 0000
ADC_PDT4	0x048	0x0000 0000
ADC_ODT	0x04C	0x0000 0000

### 18.5.1 ADC状态寄存器 (ADC\_STS)

访问：字访问

域	简称	复位值	类型	功能
位 31: 5	保留	0x00000000	resd	请保持默认值。
位 4	OCCS	0x0	rw0c	普通通道转换开始标志 (Ordinary channel conversion start flag) 该位被硬件置起，由软件将其清零（对自身写零）。 0: 未开始； 1: 已开始。
位 3	PCCS	0x0	rw0c	抢占通道转换开始标志 (Preempted channel conversion start flag) 该位被硬件置起，由软件将其清零（对自身写零）。 0: 未开始； 1: 已开始。
位 2	PCCE	0x0	rw0c	抢占通道组转换结束标志 (Preempted channels conversion end flag) 该位被硬件置起，由软件将其清零（对自身写零）。 0: 未结束； 1: 已结束。
位 1	CCE	0x0	rw0c	通道转换结束标志 (Channels conversion end flag) 该位被硬件置起，由软件将其清零（对自身写零），或由读取 ADC 普通数据寄存器 (ADC_ODT) 清零。 0: 未结束； 1: 已结束。 注：普通或抢占通道组转换结束均会置位此标志。
位 0	VMOR	0x0	rw0c	电压监测超出范围标志 (Voltage monitoring out of range flag) 该位被硬件置起，由软件将其清零（对自身写零）。 0: 无超出； 1: 有超出。

### 18.5.2 ADC控制寄存器1 (ADC\_CTRL1)

访问：字访问

域	简称	复位值	类型	功能
位 31: 24	保留	0x00	resd	请保持默认值。

位 23	OCVMEN	0x0	rw	普通通道的电压监测使能 (Voltage monitoring enable on ordinary channels) 0: 关闭; 1: 开启。
位 22	PCVMEN	0x0	rw	抢占通道的电压监测使能 (Voltage monitoring enable on preempted channels) 0: 关闭; 1: 开启。
位 21: 16	保留	0x0	resd	请保持默认值。
位 15: 13	OCPCNT	0x0	rw	分割模式下每次触发转换的普通通道个数 (Partitioned mode conversion count of ordinary channels) 000: 1 个通道; 001: 2 个通道; ..... 111: 8 个通道。 注： 抢占组在分割模式下每次触发固定只转换一个通道。
位 12	PCPEN	0x0	rw	抢占通道上的分割模式使能 (Partitioned mode enable on preempted channels) 0: 关闭; 1: 开启。
位 11	OCPEN	0x0	rw	普通通道上的分割模式使能 (Partitioned mode enable on ordinary channels) 该位由软件设置和清除，用于开启或关闭规则通道组上的分割模式 0: 关闭; 1: 开启。
位 10	PCAUTOEN	0x0	rw	普通组转换结束后的抢占组自动转换使能 (Preempted group automatic conversion enable after ordinary group) 0: 关闭; 1: 开启。
位 9	VMSGEN	0x0	rw	单个通道的电压监测使能 (Voltage monitoring enable on a single channel) 0: 关闭 (电压监测所有通道); 1: 开启 (电压监测单一通道)。
位 8	SQEN	0x0	rw	序列模式使能 (Sequence mode enable) 0: 关闭 (转换选择的单一通道); 1: 开启 (转换设定的多个通道)。 注： 如果开启多通道模式，且开启了 CCEIEN 或 PCCEIEN 位，则只在最后一个通道转换完毕后才会产生 CCE 或 PCCE 中断。
位 7	PCCEIEN	0x0	rw	抢占通道组转换结束中断使能 (conversion end interrupt enable for Preempted channels) 0: 关闭; 1: 开启。
位 6	VMORIEN	0x0	rw	电压监测超出范围中断使能 (Voltage monitoring out of range interrupt enable) 0: 关闭; 1: 开启。

位 5	CCEIEN	0x0	rw	通道转换结束中断使能 (Channel conversion end interrupt enable) 0: 关闭; 1: 开启。
位 4: 0	VMCSEL	0x00	rw	电压监测通道选择 (Voltage monitoring channel select) 仅在 VMSGEN 开启时有效。 00000: ADC_IN0 通道; 00001: ADC_IN1 通道; ..... 01011: ADC_IN11 通道; 10000: ADC_IN16 通道; 10001: ADC_IN17 通道。  00110~01001、01100~01111、10010~11111: 未用，禁止配置。

### 18.5.3 ADC控制寄存器2 (ADC\_CTRL2)

访问: 字访问

域	简称	复位值	类型	功能
位 30: 26	保留	0x00	resd	请保持默认值。
位 23	ITSRVEN	0x0	rw	内部温度传感器及 VINTRV 使能 (Internal temperature sensor and VINTRV enable) 0: 关闭; 1: 开启。
位 22	OCSWTRG	0x0	rw	软件触发普通通道转换 (Conversion trigger by software of ordinary channels) 0: 不触发; 1: 触发转换 (可由软件清除, 或在转换开始后由硬件自动清除)。
位 21	PCSWTRG	0x0	rw	软件触发抢占通道转换 (Conversion trigger by software of preempted channels) 0: 不触发; 1: 触发转换 (可由软件清除, 或在转换开始后由硬件自动清除)。
位 20	OCTEN	0x0	rw	普通通道组转换的触发模式使能 (Trigger mode enable for ordinary channels conversion) 0: 关闭; 1: 开启。
位 25 位 19: 17	OCTESEL	0x0	rw	普通通道组转换的触发事件选择 (trigger event select for ordinary channels conversion) ADC 的触发配置如 18.4.2.2
位 16	保留	0x0	resd	请保持默认值。
位 15	PCTEN	0x0	rw	抢占通道组转换的触发模式使能 (Trigger mode enable for preempted channels conversion) 0: 关闭; 1: 开启。
位 24 位 14: 12	PCTESEL	0x0	rw	抢占通道组转换的触发事件选择 (trigger event select for preempted channels conversion) ADC 的触发配置如 18.4.2.2

位 11	DTALIGN	0x0	rw	数据对齐方式 (Data alignment) 0: 右对齐; 1: 左对齐。
位 10: 9	保留	0x0	resd	请保持默认值。
位 8	OCDMAEN	0x0	rw	普通通道转换数据的 DMA 传输使能 (DMA transfer enable of ordinary channels) 0: 关闭; 1: 开启。
位 7: 4	保留	0x0	resd	请保持默认值。
位 3	ADCALINIT	0x0	rw	A/D 初始化校准 (initialize A/D calibration) 该位由软件设置并由硬件清除。在校准寄存器被初始化后该位将被清除。 0: 校准寄存器无初始化执行或初始化结束; 1: 校准寄存器初始化或初始化进行中。
位 2	ADCAL	0x0	rw	A/D 校准 (A/D Calibration) 0: 无校准执行或校准结束; 1: 开始校准或校准进行中。
位 1	RPEN	0x0	rw	反复模式使能 (Repeat mode enable) 0: 关闭 SQEN=0 时, 每次触发转换单个通道, SQEN=1 时, 每次触发转换一组通道; 1: 开启 SQEN =0 时, 一次触发后将反复转换单个通道, SQEN =1 时, 一次触发后将反复转换一组通道。直到 ADCEN 被清零。
位 0	ADCEN	0x0	rw	A/D 转换器使能 (A/D converter enable) 0: 关闭 (ADC 进入断电模式); 1: 开启。 注: 当该位为关闭状态时, 写入开启命令将把 ADC 从断电模式下唤醒。 当该位为开启状态时, 再写入开启命令时同寄存器其它位未改变, 则重复该开启命令将启动普通通道组的转换。 应用程序需注意, 在转换器上电至转换开始有一个延迟 tSTAB。

#### 18.5.4 ADC采样时间寄存器1 (ADC\_SPT1)

访问: 字访问

域	简称	复位值	类型	功能
位 31: 24	保留	0x00	resd	请保持默认值。
位 23: 21	CSPT17	0x0	rw	选择 ADC_IN17 通道的采样时间 (Selection sample time of channel ADC_IN17) 000: 1.5 周期; 001: 7.5 周期; 010: 13.5 周期; 011: 28.5 周期; 100: 41.5 周期; 101: 55.5 周期; 110: 71.5 周期; 111: 239.5 周期。

---

位 20: 18 CSPT16

0x0

rw

选择 ADC\_IN16 通道的采样时间 (Selection sample time of channel ADC\_IN16)

000: 1.5 周期;  
001: 7.5 周期;  
010: 13.5 周期;  
011: 28.5 周期;  
100: 41.5 周期;  
101: 55.5 周期;  
110: 71.5 周期;  
111: 239.5 周期。

---

位 17: 15 保留

0x00

resd

请保持默认值。

---

位 14: 12 保留

0x00

resd

请保持默认值。

---

位 11: 9 保留

0x00

resd

请保持默认值。

---

位 8: 6 保留

0x00

resd

请保持默认值。

---

位 5: 3 CSPT11

0x0

rw

选择 ADC\_IN11 通道的采样时间 (Selection sample time of channel ADC\_IN11)

000: 1.5 周期;  
001: 7.5 周期;  
010: 13.5 周期;  
011: 28.5 周期;  
100: 41.5 周期;  
101: 55.5 周期;  
110: 71.5 周期;  
111: 239.5 周期。

位 2: 0 CSPT10

0x0

rw

选择 ADC\_IN10 通道的采样时间 (Selection sample time of channel ADC\_IN10)

000: 1.5 周期;  
001: 7.5 周期;  
010: 13.5 周期;  
011: 28.5 周期;  
100: 41.5 周期;  
101: 55.5 周期;  
110: 71.5 周期;  
111: 239.5 周期。

### 18.5.5 ADC采样时间寄存器2 (ADC\_SPT2)

访问: 字访问

域	简称	复位值	类型	功能
位 31: 30	保留	0x0	resd	请保持默认值。

位 29: 27 保留

0x0

resd

请保持默认值。

位 26: 24 保留

0x0

resd

请保持默认值。

---

位 23: 21 保留 0x0 resd 请保持默认值。

---

位 20: 18 保留 0x0 resd 请保持默认值。

---

位 17: 15 CSPT5 0x0 rw 选择 ADC\_IN5 通道的采样时间 (Selection sample time of channel ADC\_IN5)  
000: 1.5 周期;  
001: 7.5 周期;  
010: 13.5 周期;  
011: 28.5 周期;  
100: 41.5 周期;  
101: 55.5 周期;  
110: 71.5 周期;  
111: 239.5 周期。

---

位 14: 12 CSPT4 0x0 rw 选择 ADC\_IN4 通道的采样时间 (Selection sample time of channel ADC\_IN4)  
000: 1.5 周期;  
001: 7.5 周期;  
010: 13.5 周期;  
011: 28.5 周期;  
100: 41.5 周期;  
101: 55.5 周期;  
110: 71.5 周期;  
111: 239.5 周期。

---

位 11: 9 CSPT3 0x0 rw 选择 ADC\_IN3 通道的采样时间 (Selection sample time of channel ADC\_IN3)  
000: 1.5 周期;  
001: 7.5 周期;  
010: 13.5 周期;  
011: 28.5 周期;  
100: 41.5 周期;  
101: 55.5 周期;  
110: 71.5 周期;  
111: 239.5 周期。

---

位 8: 6	CSPT2	0x0	rw	选择 ADC_IN2 通道的采样时间 (Selection sample time of channel ADC_IN2) 000: 1.5 周期; 001: 7.5 周期; 010: 13.5 周期; 011: 28.5 周期; 100: 41.5 周期; 101: 55.5 周期; 110: 71.5 周期; 111: 239.5 周期。
位 5: 3	CSPT1	0x0	rw	选择 ADC_IN1 通道的采样时间 (Selection sample time of channel ADC_IN1) 000: 1.5 周期; 001: 7.5 周期; 010: 13.5 周期; 011: 28.5 周期; 100: 41.5 周期; 101: 55.5 周期; 110: 71.5 周期; 111: 239.5 周期。
位 2: 0	CSPT0	0x0	rw	选择 ADC_IN0 通道的采样时间 (Selection sample time of channel ADC_IN0) 000: 1.5 周期; 001: 7.5 周期; 010: 13.5 周期; 011: 28.5 周期; 100: 41.5 周期; 101: 55.5 周期; 110: 71.5 周期; 111: 239.5 周期。

---

### 18.5.6 ADC抢占通道数据偏移寄存器x (ADC\_PCDTOx)

(x=1..4)

访问：字访问

域	简称	复位值	类型	功能
位 31: 12	保留	0x00000	resd	请保持默认值。
位 11: 0	PCDTOx	0x000	rw	抢占通道 x 的数据偏移量设定 (Data offset for Preempted channel x) ADC_PDTx 内存放的转换数据 = 原始转换数据 - ADC_PCDTOx

---

### 18.5.7 ADC电压监测高边界寄存器 (ADC\_VMHb)

访问：字访问

域	简称	复位值	类型	功能
位 31: 12	保留	0x00000	resd	请保持默认值。
位 11: 0	VMHB	0xFFFF	rw	电压监测高边界设定 (Voltage monitoring high boundary)

---

### 18.5.8 ADC电压监测低边界寄存器 (ADC\_VMLB)

访问：字访问

域	简称	复位值	类型	功能
位 31: 12	保留	0x00000	resd	请保持默认值。
位 11: 0	VMLB	0x000	rw	电压监测低边界设定 (Voltage monitoring low boundary)

### 18.5.9 ADC普通序列寄存器1 (ADC\_OSQ1)

访问：字访问

域	简称	复位值	类型	功能
位 31: 24	保留	0x00	resd	请保持默认值。
位 23: 20	OCLEN	0x0	rw	普通转换序列长度 (Ordinary conversion sequence length) 0000: 1 个转换; 0001: 2 个转换; ..... 1111: 16 个转换。
位 19: 15	OSN16	0x00	rw	普通序列中第 16 个转换通道的编号 (number of 16th conversion in ordinary sequence)
位 14: 10	OSN15	0x00	rw	普通序列中第 15 个转换通道的编号 (number of 15th conversion in ordinary sequence)
位 9: 5	OSN14	0x00	rw	普通序列中第 14 个转换通道的编号 (number of 14th conversion in ordinary sequence)
位 4: 0	OSN13	0x00	rw	普通序列中第 13 个转换通道的编号 (number of 13th conversion in ordinary sequence) 注：编号可设定 0~17，示例：设定为 3 就代表第 13 个转换的是 ADC_IN3 通道。

### 18.5.10 ADC普通序列寄存器2 (ADC\_OSQ2)

访问：字访问

域	简称	复位值	类型	功能
位 31: 30	保留	0x0	resd	请保持默认值。
位 29: 25	OSN12	0x00	rw	普通序列中第 12 个转换通道的编号 (number of 12th conversion in ordinary sequence)
位 24: 20	OSN11	0x00	rw	普通序列中第 11 个转换通道的编号 (number of 11th conversion in ordinary sequence)
位 19: 15	OSN10	0x00	rw	普通序列中第 10 个转换通道的编号 (number of 10th conversion in ordinary sequence)
位 14: 10	OSN9	0x00	rw	普通序列中第 9 个转换通道的编号 (number of 9th conversion in ordinary sequence)
位 9: 5	OSN8	0x00	rw	普通序列中第 8 个转换通道的编号 (number of 8th conversion in ordinary sequence)
位 4: 0	OSN7	0x00	rw	普通序列中第 7 个转换通道的编号 (number of 7th conversion in ordinary sequence) 注：编号可设定 0~17，示例：设定为 8 就代表第 7 个转换的是 ADC_IN8 通道。

### 18.5.11 ADC普通序列寄存器3 (ADC\_OSQ3)

访问：字访问

域	简称	复位值	类型	功能
位 31: 30	保留	0x0	resd	请保持默认值。
位 29: 25	OSN6	0x00	rw	普通序列中第 6 个转换通道的编号 (number of 6th conversion in ordinary sequence)
位 24: 20	OSN5	0x00	rw	普通序列中第 5 个转换通道的编号 (number of 5th conversion in ordinary sequence)
位 19: 15	OSN4	0x00	rw	普通序列中第 4 个转换通道的编号 (number of 4th conversion in ordinary sequence)
位 14: 10	OSN3	0x00	rw	普通序列中第 3 个转换通道的编号 (number of 3rd conversion in ordinary sequence)
位 9: 5	OSN2	0x00	rw	普通序列中第 2 个转换通道的编号 (number of 2nd conversion in ordinary sequence)
位 4: 0	OSN1	0x00	rw	普通序列中第 1 个转换通道的编号 (number of 1st conversion in ordinary sequence) 注：编号可设定 0~17，示例：设定为 17 就代表第 1 个转换的是 ADC_IN17 通道。

### 18.5.12 ADC抢占序列寄存器 (ADC\_PSQ)

访问：字访问

域	简称	复位值	类型	功能
位 31: 30	保留	0x0	resd	请保持默认值。
位 21: 20	PCLEN	0x0	rw	抢占转换序列长度 (Preempted conversion sequence length) 00: 1 个转换; 01: 2 个转换; 10: 3 个转换; 11: 4 个转换。
位 19: 15	PSN4	0x00	rw	抢占序列中第 4 个转换通道的编号 (number of 4th conversion in Preempted sequence)
位 14: 10	PSN3	0x00	rw	抢占序列中第 3 个转换通道的编号 (number of 3rd conversion in Preempted sequence)
位 9: 5	PSN2	0x00	rw	抢占序列中第 2 个转换通道的编号 (number of 2nd conversion in Preempted sequence)
位 4: 0	PSN1	0x00	rw	抢占序列中第 1 个转换通道的编号 (number of 1st conversion in Preempted sequence) 注： 编号可设定 0~17，比如设定为 3 时其代表的就是 ADC_IN3 通道。 若 PCLEN 小于 4，则转换的序列顺序是从 (4-PCLEN) 开始。例如：ADC_PSQ[21: 0] = 10 00011 00101 00100 00011，意味着扫描转换将按下列通道顺序执行：4、5、3，而不是 3、4、5。

### 18.5.13 ADC抢占数据寄存器x (ADC\_PDTx) (x=1..4)

访问：字访问

域	简称	复位值	类型	功能
位 31: 16	保留	0x0000	resd	请保持默认值。
位 15: 0	PDTx	0x0000	ro	抢占通道的转换数据 (Conversion data of preempted channel)

### 18.5.14 ADC普通数据寄存器 (ADC\_ODT)

访问：字访问

域	简称	复位值	类型	功能
位 31: 16	保留	0x0000	resd	请保持默认值。
位 15: 0	ODT	0x0000	ro	普通通道的转换数据 (Conversion data of ordinary channel)

# 19 CAN 总线控制器

## 19.1 简介

CAN (Controller Area Network) 是一种实现各节点之间实时、可靠数据通信的分布式串行通信协议，支持 CAN 协议 2.0A 和 2.0B。

## 19.2 主要特性

- 波特率最高可达 1M bit/s/
- 支持时间触发通信
- 中断使能和屏蔽
- 自动重传功能可配发送
- 3 个发送邮箱
- 发送优先级可配置
- 支持发送时间戳
- 接收
- 2 个深度为 3 的 FIFO
- 14 组过滤器组
- 支持标识符列表模式
- 支持标识符掩码模式
- 支持 FIFO 溢出管理
- 时间触发通信模式
- 16 位定时器
- 发送时间戳

## 19.3 波特率设置

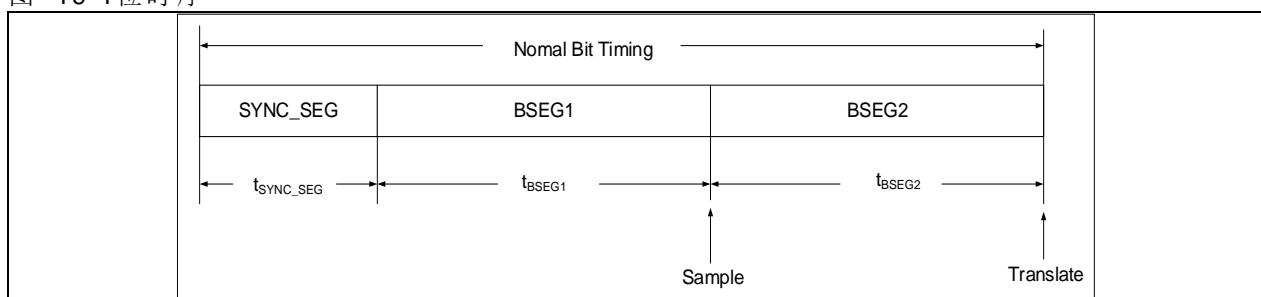
CAN 总线的额定位时间由 3 部分组成。

同步段(SYNC\_SEG)，该段占用 1 时间单元，时间长度由 CAN 位时序寄存器 (CAN\_BTMG) 的 BRDIV[11:0]位定义。

位段 1 (BIT SEGMENT 1)，包括 CAN 标准里的 PROP\_SEG 和 PHASE\_SEG1，记为 BSEG1，该段占用 1 至 16 时间单元，时间单元个数由 BTS1[3: 0]位定义。

位段 2 (BIT SEGMENT 2)，包括 CAN 标准里的 PHASE\_SEG2，记为 BSEG2，该段占用 1 至 8 时间单元，时间单元个数由 BTS2[2: 0]位定义。

图 19-1 位时序



### 波特率计算公式

$$\text{BaudRate} = \frac{1}{\text{Nomal Bit Timimg}}$$

$$\text{Nomal Bit Timimg} = t_{SYNC\_SEG} + t_{BSEG1} + t_{BSEG2}$$

其中

$$t_{SYNC\_SEG} = 1 \times t_q$$

$$t_{BSEG1} = (1 + \text{BTS1}[3: 0]) \times t_q$$

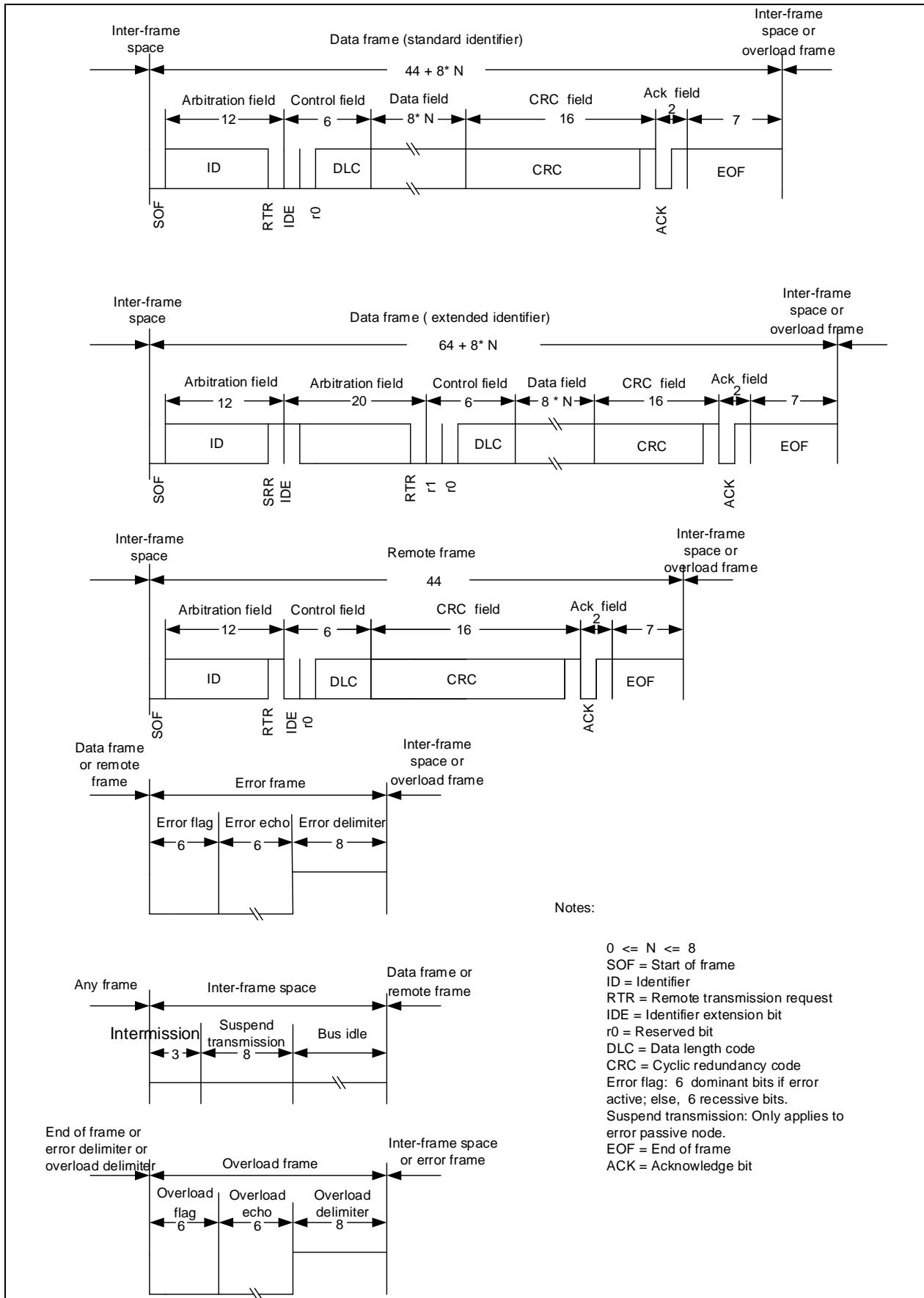
$$t_{BSEG2} = (1 + BTS2[2: 0]) \times t_q$$

$$t_q = (1 + BRDIV[11: 0]) \times t_{pclk}$$

### 硬同步和重同步

默认情况下,CAN 节点的每一位的起始位置总是在同步段内,同时在位段 1 和位段 2 临界位置进行采样。但是由于节点振荡器漂移, 网络节点之间的传播延迟以及噪声干扰等, 实际的传输过程中, CAN 节点的每一位会存在一定的相位误差。为避免相位误差对通讯造成影响, 可以通过帧起始位置的边沿以及后面的下降沿进行硬同步或者重同步, 同步补偿的时间长度最长不超过重新同步调整宽度 (1 至 4 个时间单元, RSAW[1: 0]位设置)。

图 19-2 发送中断的产生



## 19.4 中断管理

CAN 控制器具有 4 个中断向量，通过配置 CAN 中断使能寄存器（CAN\_INTEN），可以控制相应的中断开启或关闭。

图 19-3 发送中断的产生

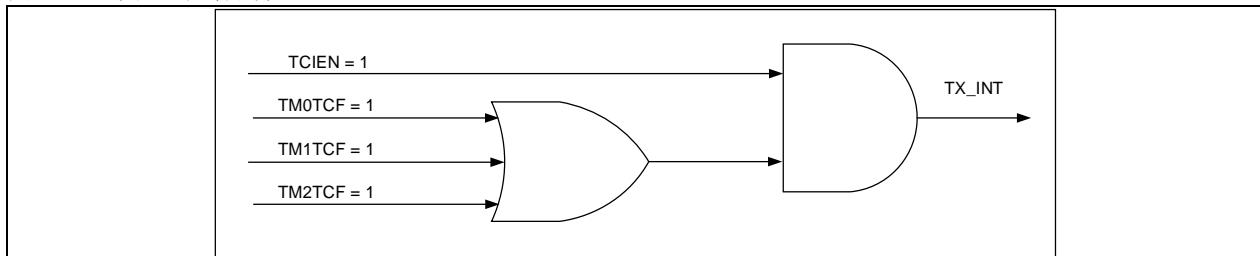


图 19-4 接收中断 0 的产生

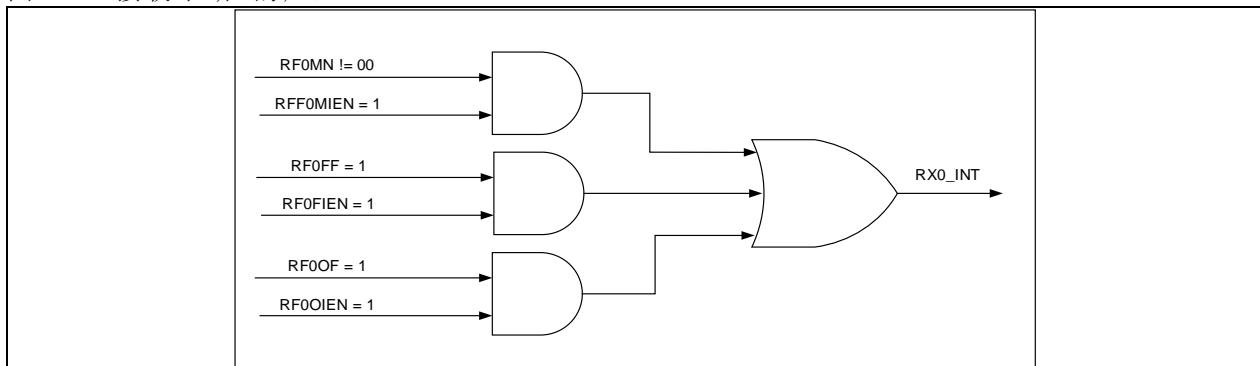


图 19-5 接收中断 1 的产生

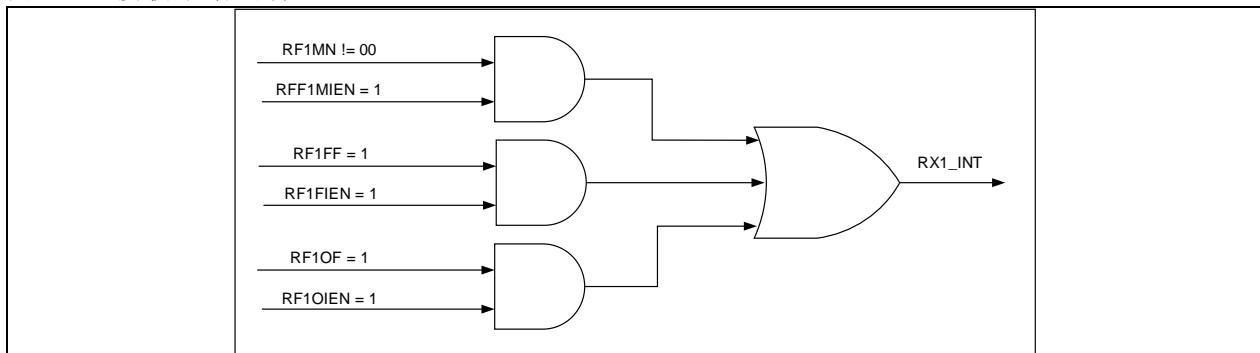
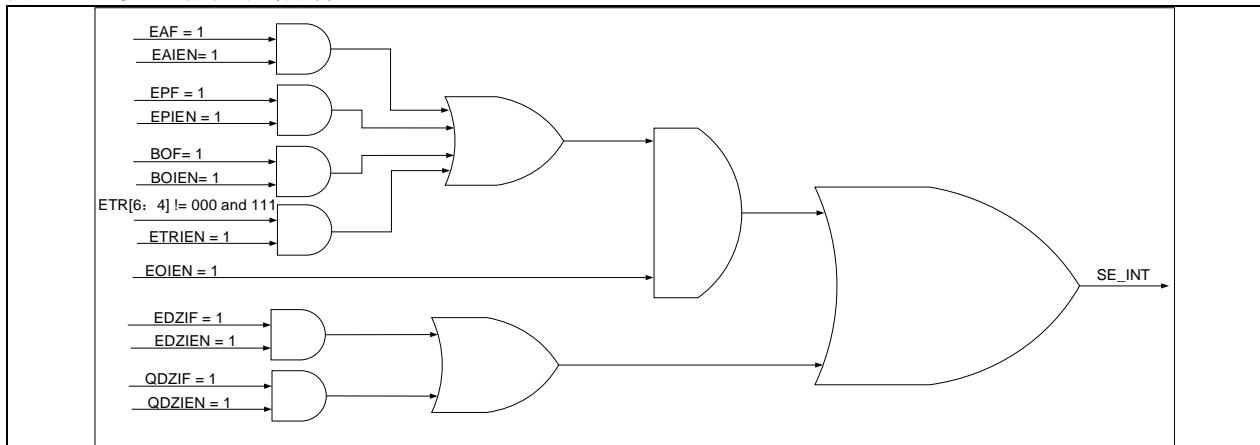


图 19-6 状态错误中断的产生



## 19.5 设计提示

为便于 CAN 应用开发，设计时建议参考如下提示。

- 调试控制

当系统进入调试模式时，可以通过控制 MCU 调试寄存器（DEBUG\_CTRL）的

CANx\_PAUSE 以及 CAN 主控制寄存器 (CAN\_MCTRL) 的 PTD 位控制 CAN 控制器处于停止状态或者正常发送接收状态。

- 时间触发通信

时间触发通信用于提高系统的实时性，避免总线竞争。当 CAN 主控制寄存器 (CAN\_MCTRL) 的 TTCEN 位置 ‘1’，CAN 控制器的时间触发通信即被激活。内部 16 位定时器在每个 CAN 位累加，在帧起始位置被采样，生成时间戳，存储在接收 FIFO 邮箱数据长度和时间戳寄存器 (CAN\_RFCx) /发送邮箱数据长度和时间戳寄存器 (CAN\_TMCx) 中。

- 寄存器访问保护

CAN 位时序寄存器 (CAN\_BTMG) 只能在冻结工作模式下进行修改。

CAN 节点发送错误数据对网络层不会带来问题，但却会对应用程序造成严重影响，因此只能在发送邮箱为空时改变它。

只有在设置过滤器为配置模式下（即 FCS=1），才能修改过滤器的设置，即修改 CAN 过滤器模式配置寄存器 (CAN\_FMCFG)，CAN 过滤器位宽配置寄存器 (CAN\_FBWCFG)，CAN 过滤器 FIFO 关联寄存器 (CAN\_FRF)。过滤位寄存器 x (CAN\_FiFBx) 只有在过滤器配置模式下（即 FCS=1）或者相应过滤器关闭情况下（即 FAENx=0）才能进行修改。

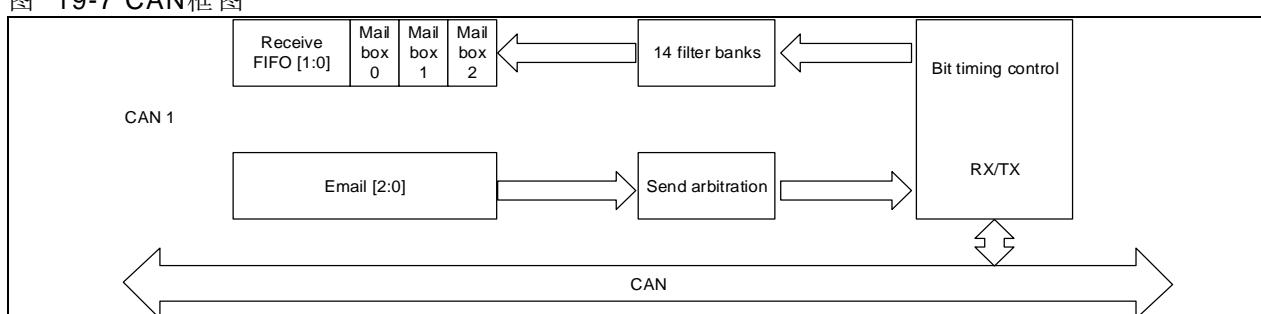
## 19.6 功能描述

### 19.6.1 整体功能描述

随着 CAN 网络节点和报文数量的增加，需要一个增强的过滤机制处理各种类型的报文，减少接收报文的处理时间，采用 FIFO 的方案，使得 CPU 可以长时间处理应用层任务而不会丢失报文。同时发送报文由硬件配置发送优先级顺序，并且完全支持标准标识符（11 位）和扩展标识符（29 位）。

基于以上考虑，CAN 控制器提供 14 组位宽可变/可配置的标识符过滤器组，2 个接收 FIFO，每个 FIFO 都可以存放 3 个完整的报文，且完全由硬件管理，共有 3 个发送邮箱，发送调度器决定发送优先级顺序。

图 19-7 CAN 框图



### 19.6.2 工作模式

CAN 控制器有 3 种工作模式：

- 睡眠模式

系统复位之后，CAN 控制器处于睡眠模式，该模式下 CAN 的时钟停止，因此可以节省电能，但软件仍然可以访问邮箱寄存器，同时内部上拉电阻被禁用。

软件通过对 CAN 主控制寄存器 (CAN\_MCTRL) 的 DZEN 位置‘1’，可以请求 CAN 进入睡眠模式，并且硬件对 CAN 主状态寄存器 (CAN\_MSTS) 的 DZC 位置‘1’进行确认。

有两种方式退出睡眠模式：配置 CAN 主控制寄存器 (CAN\_MCTRL) 的 AEDEN 位为‘1’，一旦检测到 CAN 总线的活动，硬件自动对 DZEN 位清‘0’来唤醒 CAN 控制器。或者软件对 DZEN 位清‘0’可以退出睡眠状态。

睡眠工作模式进入冻结工作模式：对 CAN 主控制寄存器 (CAN\_MCTRL) 的 FZEN 位置‘1’，并且同时对 DZEN 位清‘0’，然后硬件对 CAN 主状态寄存器 (CAN\_MSTS) 的 FZC 位置‘1’来进行确认。

睡眠工作模式进入通讯工作模式：对 CAN 主控制寄存器 (CAN\_MCTRL) FZEN 和 DZEN 位清‘0’，并且 CAN 控制器必须跟总线取得同步，即在 CANRX 管脚上监测到 11 个连续的隐性位。

- 冻结模式

软件对 CAN 控制器的初始化，只能在冻结模式下进行，包括位 CAN 位时序寄存器 (CAN\_BTMG) 和

CAN 主控制寄存器 (CAN\_MCTRL) 这 2 个寄存器。对 CAN 控制器的 14 组过滤器组 (包括模式、位宽、FIFO 关联、激活和过滤器值) 进行初始化可以在非冻结模式下进行。当 CAN 处于冻结模式时, 禁止报文的接收和发送。

冻结工作模式进入通讯工作模式: 对 CAN 主控制寄存器 (CAN\_MCTRL) FZEN 位清 ‘0’, 硬件对 CAN 主状态寄存器 (CAN\_MSTS) 的 FZC 位清‘0’就确认了冻结模式的退出, 并且 CAN 控制器必须跟总线取得同步。

冻结工作模式进入睡眠工作模式: 对 CAN 主控制寄存器 (CAN\_MCTRL) DZEN 位置 ‘1’ , CAN 主控制寄存器 (CAN\_MCTRL) FZEN 位清 ‘0’, 并且硬件对 CAN 主状态寄存器 (CAN\_MSTS) 的 DZC 位置 ‘1’ 进行确认。

- 通讯模式

在冻结工作模式配置完成 CAN 位时序寄存器 (CAN\_BTMG) 和 CAN 主控制寄存器 (CAN\_MCTRL) 这两个寄存器后, 控制 CAN 进入通讯工作模式, 开始报文收发过程。

通讯工作模式进入睡眠工作模式: 对 CAN 主控制寄存器 (CAN\_MCTRL) DZEN 位置 ‘1’, 并等待当前 CAN 总线传输完成。

通讯工作模式进入冻结工作模式: 对 CAN 主控制寄存器 (CAN\_MCTRL) FZEN 位置 ‘1’, 并等待当前 CAN 总线传输完成。

### 19.6.3 测试方法

CAN 控制器定义了三种方法用于测试分析, 包括只听方式、回环方式以及回环只听方式, 可以通过 CAN 位时序寄存器 (CAN\_BTMG) 的 LOEN 位和 LBEN 位进行配置。

- 当 CAN\_BTMG[31]位为 ‘1’ 时采用只听方式, 此时 CAN 可以正常接收数据, 但发送端 CANTX 固定隐性位输出。同时, 发送端 CANTX 发出的显性位可以被接收端侦测到, 但是不会影响到 CAN 总线。
- 当 CAN\_BTMG[30]位为 ‘1’ 时采用回环方式, 此时 CAN 只会接收本节点发送端 CANTX 的电平信号, 同时 CAN 可以发送数据至外部总线, 回环方式主要用于本节点的自我检测。
- 当 CAN\_BTMG[31: 30]位为 ‘11’ 时, 只听方式和回环方式同时有效, 此时 CAN 与总线网络断开, 发送端 CANTX 固定隐性位输出, 并且发送端直接与接收端相连。

### 19.6.4 报文过滤

在接受到的报文会根据其标识符 (ID) 进行过滤, 通过过滤的报文会存储在对应的 FIFO 中, 没有通过的报文则会被丢弃, 整个过程由硬件自动完成, 不会占用 CPU 开销。

#### 过滤器的位宽

每个 CAN 控制器提供 14 个位宽可变、可配置的过滤器组 (0~13), 每个过滤器组由 2 个 32 位寄存器, CAN\_FiFB1 和 CAN\_FiFB2 组成, 通过配置 CAN 过滤器位宽配置寄存器 (CAN\_FBWCFG) 的对应位, 设置过滤器位宽为 2 个 16 位或者单个 32 位。

32 位宽的过滤器寄存器 CAN\_FiFBx 包括: SID[10: 0]、EID[17: 0]、IDT 和 RTR 位。

CAN_FiFB1[31: 21]	CAN_FiFB1[20: 3]	CAN_FiFB1[2: 0]		
CAN_FiFB2[31: 21]	CAN_FiFB2[20: 3]	CAN_FiFB2[2: 0]		
SID[10: 0]/EID[28: 18]	EID[17: 0]	IDT	RTR	0

2 个 16 位宽的过滤器寄存器 (CAN\_FiFBx) 包括: SID[10: 0]、IDT、RTR 和 EID[17: 15]位。

CAN_FiFB1[31: 21]	CAN_FiFB1 [20: 19]	CAN_FiFB1 [18: 16]	CAN_FiFB1[15: 5]	CAN_FiFB1 [4: 3]	CAN_FiFB1 [2: 0]		
CAN_FiFB2[31: 21]	CAN_FiFB2 [20: 19]	CAN_FiFB2 [18: 16]	CAN_FiFB2[15: 5]	CAN_FiFB2 [4: 3]	CAN_FiFB2 [2: 0]		
SID[10: 0]	IDT	RTR	EID[17: 15]	SID[10: 0]	IDT	RTR	EID[17: 15]

#### 过滤器模式

通过设置 CAN 过滤器模式配置寄存器 (CAN\_FMCFG) 的 FMSELx 位可以设置过滤器寄存器工作在标识符掩码模式或者标识符列表模式, 掩码模式用来指定哪些位与预设标识符相同, 哪些位无需比较, 列表

模式表示标识符 (ID 号) 必须与预设标识符一致。两种模式与过滤器位宽配合使用，可以有以下四种过滤方式：

图 19-8 32位宽标识符掩码模式

ID	CAN_FiFB1[31:21]	CAN_FiFB1[20:3]	CAN_FiFB1[2:0]	
Mask	CAN_FiFB2[31:21]	CAN_FiFB2[20:3]	CAN_FiFB2[2:0]	
Mapping	SID[10:0]	EID[17:0]	IDT	RTR

图 19-9 32位宽标识符列表模式

ID	CAN_FiFB1[31:21]	CAN_FiFB1[20:3]	CAN_FiFB1[2:0]	
ID	CAN_FiFB2[31:21]	CAN_FiFB2[20:3]	CAN_FiFB2[2:0]	
Mapping	SID[10:0]	EID[17:0]	IDT	RTR

图 19-10 16位宽标识符掩码模式

ID	CAN_FiFB1[15:5]	CAN_FiFB1[4:0]		
	CAN_FiFB1[31:21]	CAN_FiFB1[20:16]		
ID	CAN_FiFB2[15:5]	CAN_FiFB2[4:0]		
	CAN_FiFB2[31:21]	CAN_FiFB2[20:16]		
Mapping	SID[10:0]	RTR	IDT	EID[17:15]

图 19-11 16位宽标识符列表模式

ID	CAN_FiFB1[15:8]	CAN_FiFB1[7:0]		
	CAN_FiFB1[31:24]	CAN_FiFB1[23:16]		
ID	CAN_FiFB2[15:8]	CAN_FiFB2[7:0]		
	CAN_FiFB2[31:24]	CAN_FiFB2[23:16]		
Mapping	SID[10:0]	RTR	IDT	EID[17:15]

### 过滤器匹配序号

14 组过滤器组根据位宽模式的不同，具有不同的过滤效果，例如 32 位宽标识符掩码模式包含序号为 n 的过滤器，而 16 位宽标识符列表模式包含序号为 n、n+1、n+2 以及 n+3 的过滤器。一帧报文通过了某个序号 (Filter Nnumber) N 的过滤器，则该帧的接收 FIFO 邮箱数据长度和时间戳寄存器 (CAN\_RFCx) RFFMN[7: 0]位存储该序号 N，过滤器序号的分配不关心对应的过滤器组是否处于激活状态。

下表为过滤器匹配序号的示例。

Filter bank	FIFO0	Active	Filter number	Filter bank	FIFO1	Active	Filter number
0	CAN_F0FB1[31: 0]-ID	Yes	0	3	CAN_F3FB1[15: 0]-ID	Yes	0
	CAN_F0FB2[31: 0]-ID		1		CAN_F3FB1[31: 16]-ID		1
1	CAN_F1FB1[15: 0]-ID	Yes	2	4	CAN_F3FB2[15: 0]-ID	Yes	2
	CAN_F1FB1[31: 16]-ID		3		CAN_F3FB2[31: 16]-ID		3
	CAN_F1FB2[15: 0]-ID		4		CAN_F4FB1[31: 0]-ID	Yes	4
	CAN_F1FB2[31: 16]-ID		5		CAN_F4FB2[31: 0]-Mask		

2	CAN_F2FB1[31: 0]-ID	Yes	6	5	CAN_F5FB1[15: 0]-ID	No	5
	CAN_F2FB2[31: 0]-Mask				CAN_F5FB1[31: 16]-Mask		
6	CAN_F6FB1[15: 0]-ID	No	7	7	CAN_F5FB2[15: 0]-ID	No	6
	CAN_F6FB1[31: 16]-Mask		8		CAN_F5FB2[31: 16]-Mask		7
	CAN_F6FB2[15: 0]-ID				CAN_F7FB1[15: 0]-ID	No	8
	CAN_F6FB2[31: 16]-Mask		9		CAN_F7FB1[31: 16]-ID		9
9	CAN_F9FB1[31: 0]-ID	No	10	7	CAN_F7FB2[15: 0]-ID	No	10
	CAN_F9FB2[31: 0]-ID		11		CAN_F7FB2[31: 16]-ID		11
10	CAN_F10FB1[15: 0]-ID	Yes	11	8	CAN_F8FB1[31: 0]-ID	Yes	11
	CAN_F10FB1[31: 16]-Mask				CAN_F8FB2[31: 0]-Mask		
	CAN_F10FB2[15: 0]-ID		12	11	CAN_F11FB1[31: 0]-ID	Yes	12
	CAN_F10FB2[31: 16]-Mask		13		CAN_F11FB2[31: 0]-ID		13
12	CAN_F12FB1[15: 0]-ID	No	13	13	CAN_F13FB1[15: 0]-ID	Yes	14
	CAN_F12FB1[31: 16]-ID		14		CAN_F13FB1[31: 16]-ID		15
	CAN_F12FB2[15: 0]-ID		15		CAN_F13FB2[15: 0]-ID	Yes	16
	CAN_F12FB2[31: 16]-ID		16		CAN_F13FB2[31: 16]-ID		17

### 优先级匹配规则

CAN 控制器接收一帧报文，有可能能够通过多个过滤器的过滤，在这种情况下，存放在接收邮箱中的过滤器匹配序号，根据以下优先级规则确定。

- 位宽为 32 位的过滤器，优先级高于位宽为 16 位的过滤器。
- 在相同位宽的情况下，标识符列表模式的优先级高于标识符掩码模式。
- 在位宽和标识符模式都相同的情况下，标号越小的过滤器具有更高的优先级。

### 过滤器配置

- 将 CAN 过滤器控制寄存器（CAN\_FCTRL）FCS 位置 ‘1’，允许配置 CAN 过滤器。
- 写 CAN 过滤器模式配置寄存器（CAN\_FMCFG）FMSELx 位，控制过滤器工作模式为标识符掩码模式或者列表模式。
- 写 CAN 过滤器位宽配置寄存器（CAN\_FBWCFG）FBWSELx 位，控制过滤器位宽为 2 个 16 位或者单个 32 位。
- 写 CAN 过滤器 FIFO 关联寄存器（CAN\_FRF）FRFSELx 位，关联过滤器 x 到 FIFO0 或者 FIFO1。
- 将 CAN 过滤器激活控制寄存器（CAN\_FACFG）FAENx 位置 ‘1’，激活对应的过滤器组 x。
- 写 CAN\_FiFBx（其中 i=0...13; x=1,2），配置 0~13 组过滤器组。
- 将 CAN 过滤器控制寄存器（CAN\_FCTRL）FCS 位置 ‘0’，完成 CAN 过滤器配置过程。

## 19.6.5 报文发送

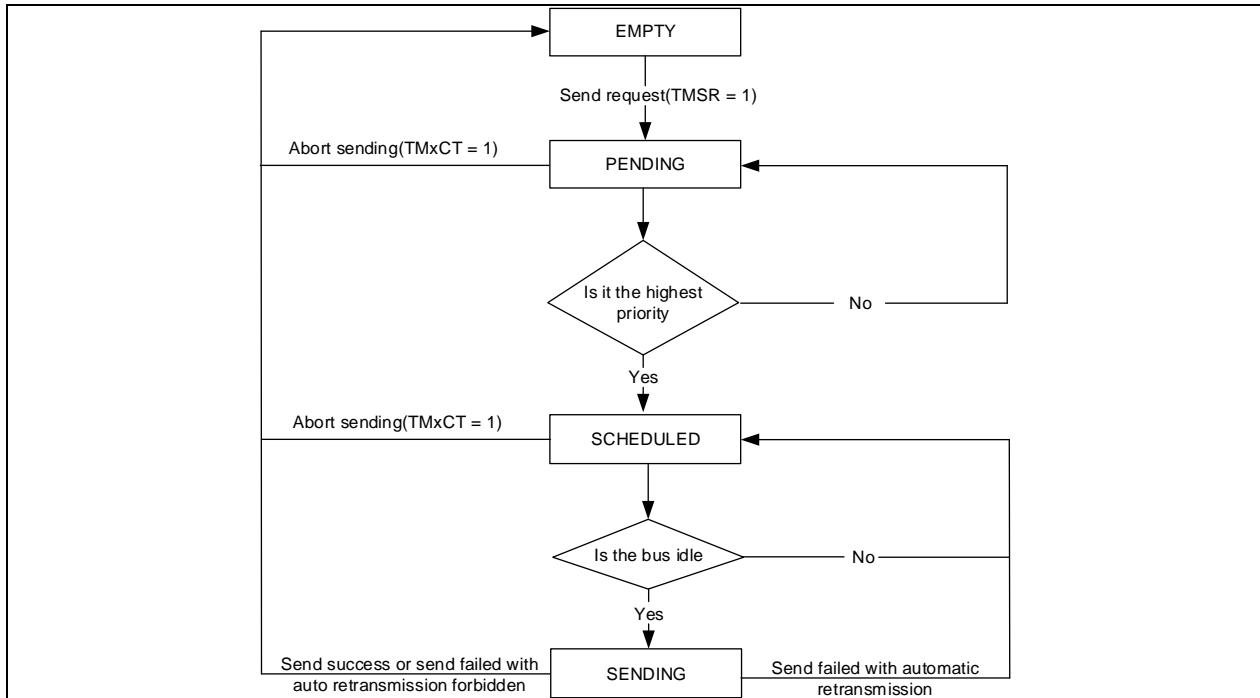
### 寄存器配置

数据发送首先需要选择发送邮箱进行配置，对应的寄存器为发送邮箱标识符寄存器（CAN\_TMIx）、发送邮箱数据长度和时间戳寄存器（CAN\_TMCx）、发送邮箱低字节数据寄存器（CAN\_TMDTLx）以及发送邮箱高字节数据寄存器（CAN\_TMDTHx）。当邮箱配置完成后，对发送邮箱标识符寄存器（CAN\_TMIx）TMSR 位置 ‘1’ 控制 CAN 启动发送流程。

### 报文发送

当对应的邮箱配置完成且 CAN 控制器接收到发送请求后，该邮箱进入 PENDING 状态，此时 CAN 控制器会检查该邮箱是否处于最高优先级状态，如果是则进入 SCHEDULED 状态，否则停下等待该邮箱获取最高优先级。处于 SCHEDULED 状态的邮箱会实时监控 CAN 总线状态，只要总线空闲，预定发送邮箱中的报文就马上被发送。发送完成，该邮箱进入 EMPTY 状态。

图 19-12 发送邮箱状态转换



### 发送优先级配置

当有两个及以上发送邮箱处于 PENDING 状态时，需要决定邮箱的发送优先级。

由标识符决定：当 CAN 主控制寄存器 (CAN\_MCTRL) 的 MMSSR 位置 ‘0’，发送顺序由邮箱中报文的标识符决定。标识符数值低的报文具有更高优先级，相同标识符的，邮箱号小的报文优先发送。

由发送请求顺序决定：当 CAN 主控制寄存器 (CAN\_MCTRL) 的 MMSSR 位置 ‘1’，发送优先级由各邮箱的发送请求次序决定。

### 发送状态及错误信息

CAN 发送状态寄存器 (CAN\_TSTS) 中的 TMxTCF、TMxTSF、TMxALF、TMxTEF 以及 TMxEF 用于显示发送状态和错误信息。

TMxTCF 位：发送完成标志。表示本次数据发送完成，置 ‘1’ 有效。

TMxTSF 位：无错误发送完成标志。表示本次数据发送完成且无错误，置 ‘1’ 有效。

TMxALF 位：发送仲裁丢失标志。表示本次数据发送仲裁失败，置 ‘1’ 有效。

TMxTEF 位：发送错误标志。表示本次数据发送检测到总线错误，且发送错误帧，置 ‘1’ 有效。

TMxEF 位：邮箱空标志。表示本次数据发送完成，邮箱变为空状态，置 ‘1’ 有效。

### 数据发送中止

可以通过将 CAN 发送状态寄存器 (CAN\_TSTS) 的 TMxCT 位置 ‘1’ 中止当前邮箱的发送，具体情况需要分类讨论。

当前邮箱发送失败或者丢失仲裁，假如报文自动重传功能被禁止，则发送邮箱进入 EMPTY 状态；假如报文自动重传功能被使能，则发送邮箱进入 SCHEDULED 状态，接着邮箱发送被中止，进入 EMPTY 状态。

当前邮箱本次数据发送完成且无错误，邮箱进入 EMPTY 状态。

## 19.6.6 报文接收

### 寄存器配置

用户程序通过读接收 FIFO 邮箱标识符寄存器 (CAN\_RFIx)、接收 FIFO 邮箱数据长度和时间戳寄存器 (CAN\_RFCLx)、接收 FIFO 邮箱低字节数据寄存器 (CAN\_RFDTLx) 以及接收 FIFO 邮箱高字节数据寄存器 (CAN\_RFDTHx) 获取接收到的有效报文。

### 报文接收

CAN 控制器具有两个深度为 3 的 FIFO 用于接收报文，采用先进先出的原则。当报文被正确接收且通过了标识符过滤，则被认为是有效报文并存储在对应的 FIFO 中。接收 FIFO 每接收到一帧有效报文，CAN\_RFx 寄存器中的报文数目 RFxMN[1: 0]就加 1，当 RFxMN[1: 0]等于 3 的同时又接收到一帧有效报文，此时控制器会根据 CAN 主控制寄存器 (CAN\_MCTRL) 的 MDRSEL 位选择覆盖接收到的原有的报文或者丢弃该报文。

同时，当用户每次读出一帧报文且控制 CAN\_RFx 寄存器 RFxR 位置 ‘1’，则对应 FIFO 释放一个深度空间，并且 CAN\_RFx 寄存器中的报文数目 RFxMN[1: 0]减 1。

### 接收 FIFO 状态

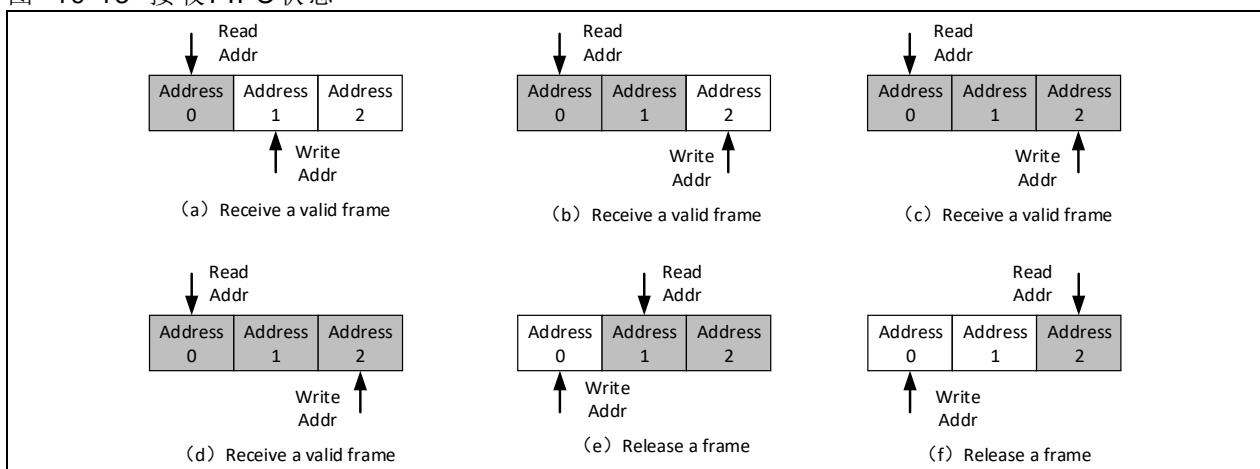
寄存器 RFx 中的 RFxMN[1: 0]、RFxFF 以及 RFxOF 用于显示接收 FIFO 的状态信息。

RFxMN[1: 0]: 表示 FIFOx 中当前存储有效报文的数目。

RFxFF: 表示 FIFOx 中当前存储 3 帧有效报文处于 ‘满’ 状态，如图 (c) 所示。

RFxOF: 表示 FIFOx 中当前有 3 帧有效报文同时又接收到一帧有效报文，处于溢出状态，如图 (d) 所示。

图 19-13 接收 FIFO 状态



## 19.6.7 出错管理

CAN 总线的状态可以根据发送错误计数值 TEC 和接收错误计数值 REC 表明当前 CAN 节点所处的状态，同时 CAN 错误状态寄存器 (CAN\_ESTS) 的 ETR[2: 0]位用于记录上次错误的原因，这些错误状态在 CAN 中断使能寄存器 (CAN\_INTEN) 控制下产生中断。

- 主动错误状态：当 TEC 且 REC 计数值都小于 128 时，系统处于主动错误状态，当检测到错误时发送主动错误标志。
- 被动错误状态：当 TEC 或 REC 计数值大于 127 时，系统处于被动错误状态，当检测到错误时发送被动错误标志。
- 离线状态：当 TEC 大于 255 时，系统进入离线状态，处于离线状态的节点不能发送接收报文，从离线状态恢复分两种情况。当 CAN 主控制寄存器 (CAN\_MCTRL) AEBOEN 位为 ‘0’ 时，通信模式下软件先请求进入冻结模式，然后退出冻结模式，接着 CAN 节点 RX 检测到 128 次 11 个连续隐性位，随后从离线状态恢复。当 AEBOEN 位为 ‘1’ 时，通信模式下 CAN 节点 RX 检测到 128 次 11 个连续隐性位，随后自动从离线状态恢复。

## 19.7 CAN寄存器

必须以字（32 位）的方式操作这些外设寄存器。

表 19-1 CAN寄存器映像和复位值

寄存器简称	基址偏移量	复位值
MCTRL	000h	0x0001 0002
MSTS	004h	0x0000 0C02
TSTS	008h	0x1C00 0000
RF0	00Ch	0x0000 0000
FR1	010h	0x0000 0000

INTEN	014h	0x0000 0000
ESTS	018h	0x0000 0000
BTMG	01Ch	0x0123 0000
保留	020h~17Fh	xx
TMI0	180h	0xXXXX XXXX
TMC0	184h	0xXXXX XXXX
TMDTL0	188h	0xXXXX XXXX
TMDTH0	18Ch	0xXXXX XXXX
TMI1	190h	0xXXXX XXXX
TMC1	194h	0xXXXX XXXX
TMDTL1	198h	0xXXXX XXXX
TMDTH1	19Ch	0xXXXX XXXX
TMI2	1A0h	0xXXXX XXXX
TMC2	1A4h	0xXXXX XXXX
TMDTL2	1A8h	0xXXXX XXXX
TMDTH2	1ACh	0xXXXX XXXX
RFI0	1B0h	0xXXXX XXXX
RFC0	1B4h	0xXXXX XXXX
RFDTL0	1B8h	0xXXXX XXXX
RFDTL0	1BCh	0xXXXX XXXX
RFI1	1C0h	0xXXXX XXXX
RFC1	1C4h	0xXXXX XXXX
RFDTL1	1C8h	0xXXXX XXXX
RFDTL1	1CCh	0xXXXX XXXX
保留	1D0h~1FFh	xx
FCTRL	200h	0x2A1C 0E01
FMCFG	204h	0x0000 0000
保留	208h	xx
FSCFG	20Ch	0x0000 0000
保留	210h	xx
FRF	214h	0x0000 0000
保留	218h	xx
FACFG	21Ch	0x0000 0000
保留	220h~23Fh	xx
FB0F1	240h	0xXXXX XXXX
FB0F2	244h	0xXXXX XXXX
FB1F1	248h	0xXXXX XXXX
FB1F2	24Ch	0xXXXX XXXX
...	...	...
FB13F1	2A8h	0xXXXX XXXX
FB13F2	2ACh	0xXXXX XXXX

## 19.7.1 CAN控制和状态寄存器

### 19.7.1.1 CAN主控制寄存器 (CAN\_MCTRL)

域	简称	复位值	类型	功能
位 31: 17	保留	0x0000	resd	请保持默认值。
位 16	PTD	0x1	rw	<p>调试时禁止收发 (Prohibit trans when debug) 0: 不禁止; 1: 禁止。仍然可以正常地读写和控制接收 FIFO。 注: 仅 PTD 及 DEBUG_CTRL 寄存器的 CANx_PAUSE 同时置位时, 才会实现禁止收发的效果。</p>
位 15	SPRST	0x0	rw1s	<p>部分软复位 (Software partial reset) 0: 不复位; 1: 部分复位。 注: SPRST 只复位接收 FIFO 及 MCTRL 寄存器。 复位后 CAN 进入睡眠模式。此后硬件自动对该位清零。</p>
位 14: 8	保留	0x00	resd	请保持默认值。
位 7	TTCEN	0x0	rw	<p>时间触发通信模式使能 (Time triggered communication mode enable) 0: 关闭; 1: 开启。</p>
位 6	AEBOEN	0x0	rw	<p>自动退出离线状态使能 (Automatic exit bus-off enable) 0: 关闭; 1: 开启。 注: 当开启时, 硬件只需检测到 CAN 总线上出现退出时序就自动退出; 当关闭时, 需要软件执行一次额外的冻结模式的进入以及退出动作, 接着在 CAN 总线上检测到退出时序时才会退出离线状态。</p>
位 5	AEDEN	0x0	rw	<p>自动退出睡眠模式使能 (Automatic exit doze mode enable) 0: 关闭; 1: 开启。 注: 当关闭时, 需软件写清睡眠请求命令来退出; 当开启时, 无需软件干预, 只要检测到 CAN 总线上出现报文时就立即退出睡眠模式。</p>
位 4	PRSFEN	0x0	rw	<p>发送失败时禁止重传使能 (Prohibit retransmission when sending fails enable) 0: 关闭; 1: 开启。</p>
位 3	MDRSEL	0x0	rw	<p>接收溢出时报文丢弃规则选择 (Message discarding rule select when overflow) 0: 上一帧收到的报文被丢弃; 1: 新收到的报文被丢弃。</p>
位 2	MMSSR	0x0	rw	<p>多报文发送顺序规则选择 (Multiple message sending sequence rule) 0: 标识符最小的最先被发送; 1: 最先请求的最先被发送。</p>
位 1	DZEN	0x1	rw	<p>睡眠模式使能 (Doze mode enable) 0: 关闭; 1: 开启。 注: 当设置了 AEDEN, 且检测到 CAN 总线上出现报文时, 硬件会自动退出睡眠模式; 在 CAN 复位或部分软复位后, 该位被硬件强制置位, 即 CAN 默认将处于睡眠模式。</p>
位 0	FZEN	0x0	rw	冻结模式使能 (Freeze mode enable) 0: 关闭;

1: 开启。

注:

当写关闭命令时,会在检测到接收管脚上出现连续的11个隐性位才会实际退出。因此软件需等待FZC被硬件清零来确认。

当写开启命令时,会在当前的CAN活动(发送或接收)结束后才会实际进入。因此软件需等待FZC被硬件置位来确认。

### 19.7.1.2 CAN主状态寄存器 (CAN\_MSTS)

域	简称	复位值	类型	功能
位 31: 12	保留	0x00000	resd	请保持默认值。
位 11	REALRX	0x1	ro	接收管脚实时电平(Real time level of RX pin) 0: 低电平; 1: 高电平。
位 10	LSAMP RX	0x1	ro	接收管脚上次采样电平>Last sample level of RX pin) 0: 低电平; 1: 高电平。 注:此值会跟随REALRX实时更新。
位 9	CURS	0x0	ro	当前的接收状态(Currently receiving status) 0: 未接收; 1: 正在接收。 注:在CAN开始接收时硬件置位此标志,接收完毕后硬件自动清除。
位 8	CUSS	0x0	ro	当前的发送状态(Currently sending status) 0: 未发送; 1: 正在发送。 注:在CAN开始发送时硬件置位此标志,发送完毕后硬件自动清除。
位 7: 5	保留	0x0	resd	请保持默认值。
位 4	EDZIF	0x0	rw1c	进入睡眠模式的中断标志(Enter doze mode interrupt flag) 0: 未进入或无标志置位条件; 1: 已进入。 注: 只有当EDZIEN=1,且CAN进入睡眠模式时才会由硬件置位此标志。此标志的置位将会产生一个状态改变中断。此标志可由软件清零(对自身写一),或当DZC位被清零时硬件自动对本标志清零。
位 3	QDZIF	0x0	rw1c	退出睡眠模式的中断标志(Quit doze mode interrupt flag) 0: 未退出或无退出条件; 1: 已退出或产生了退出条件。 注: 该位由软件将其清零(对自身写一)。 退出条件为检测到总线上出现帧起始位(SOF)。 如果QDZIEN=1,此标志的置位将会产生一个状态改变中断。
位 2	EOIF	0x0	rw1c	出现错误的中断标志(Error occur Interrupt flag) 0: 未出现或无标志置位条件; 1: 已出现。 注: 该位由软件将其清零(对自身写一)。 仅当CAN错误状态寄存器(CAN_ESTS)中的某位被置位,且其对应的CAN中断使能寄存器(CAN_INTEN)的相应中断使能位处于使能状态时,该标志才会被硬件置位。EOIEN=1时,此标志的置位将会产生一个状态改变中断。
位 1	DZC	0x1	ro	睡眠模式确认(Doze mode confirm) 0: 未处于睡眠模式;

---

				1: 正处于睡眠模式中。 注: 该位用于确定 CAN 当前是否处于睡眠模式，是对软件请求进入睡眠模式的确认。 当进入睡眠模式时，会在当前的 CAN 活动（发送或接收）结束后才会实际进入。因此软件需等待本标志被硬件置位来确认进入睡眠模式。 当退出睡眠模式（即软件写关闭睡眠模式命令，或者自动退出睡眠模式使能状态下检测到 CAN 总线上出现报文）时，会在检测到 CAN 的 RX 管脚上出现连续的 11 位隐性位时才会实际退出。因此软件需等待本标志被硬件清零来确认退出睡眠模式。
位 0	FZC	0x0	ro	冻结模式确认 (Freeze mode confirm) 0: 未处于冻结模式; 1: 正处于冻结模式中。 注: 该位用于确定 CAN 当前是否处于冻结模式，是对软件请求进入冻结模式的确认。 当进入冻结模式时，会在当前的 CAN 活动（发送或接收）结束后才会实际进入。因此软件需等待本标志被硬件置位来确认进入冻结模式。 当退出冻结模式时，会在检测到 CAN 的 RX 管脚上出现连续的 11 位隐性位时才会实际退出。因此软件需等待本标志被硬件清零来确认退出冻结模式。

---

### 19.7.1.3 CAN发送状态寄存器 (CAN\_TSTS)

域	简称	复位值	类型	功能
位 31	TM2LPF	0x0	ro	邮箱 2 优先级最低标志 (Transmit mailbox 2 lowest priority flag) 0: 非最低优先级; 1: 最低优先级（表明多个邮箱在等待发送报文时，邮箱 2 的优先级最低）。
位 30	TM1LPF	0x0	ro	邮箱 1 优先级最低标志 (Transmit mailbox 1 lowest priority flag) 0: 非最低优先级; 1: 最低优先级（表明多个邮箱在等待发送报文时，邮箱 1 的优先级最低）。
位 29	TM0LPF	0x0	ro	邮箱 0 最低优先级标志 (Transmit mailbox 0 lowest priority flag) 0: 非最低优先级; 1: 最低优先级（表明多个邮箱在等待发送报文时，邮箱 0 的优先级最低）。
位 28	TM2EF	0x1	ro	发送邮箱 2 空标志 (Transmit mailbox 2 empty flag) 当发送邮箱 2 中没有等待发送的报文时，硬件置位该位。
位 27	TM1EF	0x1	ro	发送邮箱 1 空标志 (Transmit mailbox 1 empty flag) 当发送邮箱 1 中没有等待发送的报文时，硬件置位该位。
位 26	TM0EF	0x1	ro	发送邮箱 0 空标志 (Transmit mailbox 0 empty flag) 当发送邮箱 0 中没有等待发送的报文时，硬件置位该位。
位 25: 24	TMNR	0x0	ro	发送邮箱号记录 (Transmit Mailbox number record) 注: 当有发送邮箱为空时，这两位表示接下来将要使用的空置邮箱号。 示例: CAN 空闲状态下，写一个报文的发送命令后，这 2 位的值将变为 01。 当没有发送邮箱为空时，这两位表示优先级最低的那个发送邮箱号。 示例: 3 个报文待发，报文标识符依次为，邮箱 0 为 0x400，邮箱 1 为 0x433，邮箱 2 为 0x411，此时这 2 位的值将变为 01。
位 23	TM2CT	0x0	rw1s	邮箱 2 取消发送 (Transmit mailbox 2 cancel transmit)

				0: 无意义; 1: 取消发送。 注: 软件设置此位可中断邮箱 2 的发送, 硬件清除邮箱 2 的发送报文后同步清除该位。若邮箱 2 为空置邮箱时, 软件置位该位没有任何意义。
位 22: 20 保留		0x0	resd	保持默认值。 邮箱 2 发送错误标志 (Transmit mailbox 2 transmission error flag)
位 19 TM2TEF		0x0	rw1c	0: 无错误; 1: 出现错误。 注: 当邮箱 2 出现发送错误时置位该位。 可由软件对该位写一清零。或在启动下一次发送时由硬件清除此标志。
位 18 TM2ALF		0x0	rw1c	邮箱 2 仲裁丢失标志 (Transmit mailbox 2 arbitration lost flag) 0: 无仲裁问题; 1: 出现仲裁丢失。 注: 当邮箱 2 因仲裁丢失导致发送失败时置位该位。 可由软件对该位写一清零。或在启动下一次发送时由硬件清除此标志。
位 17 TM2TSF		0x0	rw1c	邮箱 2 发送成功标志 (Transmit mailbox 2 transmission success flag) 0: 发送失败; 1: 发送成功。 注: 该位实时指示每次邮箱 2 的发送结果。可由软件对该位写一清零。
位 16 TM2TCF		0x0	rw1c	邮箱 2 发送完成标志 (transmit mailbox 2 transmission completed flag) 0: 正在发送; 1: 发送完成。 注: 每次对邮箱 2 的请求 (发送或中止) 完成后, 由硬件置位该位。 该位可由软件写一清零。或当接收到新的发送请求时由硬件自动清除。 当该位被清除时, 邮箱 2 的 TM2TSF、TM2ALF、TM2TEF 也会同步被硬件清除。
位 15 TM1CT		0x0	rw1s	邮箱 1 取消发送 (Transmit mailbox 1 cancel transmit) 0: 无意义; 1: 取消发送。 注: 软件设置此位可禁止邮箱 1 的发送, 硬件清除邮箱 1 的发送报文后同步清除该位。若邮箱 1 为空置邮箱时, 软件置位该位没有任何意义。
位 14: 12 保留		0x0	resd	保持默认值。
位 11 TM1TEF		0x0	rw1c	邮箱 1 发送错误标志 (Transmit mailbox 1 transmission error flag) 0: 无错误; 1: 出现错误。 注: 当邮箱 1 出现发送错误时置位该位。 可由软件对该位写一清零。或在启动下一次发送时由硬件清除此标志。
位 10 TM1ALF		0x0	rw1c	邮箱 1 仲裁丢失标志 (Transmit mailbox 1 arbitration lost flag) 0: 无仲裁问题; 1: 出现仲裁丢失。 注: 当邮箱 1 因仲裁丢失导致发送失败时置位该位。

				可由软件对该位写一清零。或在启动下一次发送时由硬件清除此标志。
位 9	TM1TSF	0x0	rw1c	<p>邮箱 1 发送成功标志 (Transmit mailbox 1 transmission success flag)</p> <p>0: 发送失败; 1: 发送成功。</p> <p>注: 该位实时指示每次邮箱 1 的发送结果。可由软件对该位写一清零。</p>
位 8	TM1TCF	0x0	rw1c	<p>邮箱 1 发送完成标志 (Transmit mailbox 1 transmission completed flag)</p> <p>0: 正在发送; 1: 发送完成。</p> <p>注: 每次对邮箱 1 的请求 (发送或中止) 完成后, 由硬件置位该位。 该位可由软件写一清零。或当接收到新的发送请求时由硬件自动清除。 当该位被清除时, 邮箱 1 的 TM1TSF、TM1ALF、TM1TEF 也会同步被硬件清除。</p>
位 7	TM0CT	0x0	rw1s	<p>邮箱 0 取消发送 (Transmit mailbox 0 cancel transmit)</p> <p>0: 无意义; 1: 取消发送。</p> <p>注: 软件设置此位可禁止邮箱 0 的发送, 硬件清除邮箱 0 的发送报文后同步清除该位。若邮箱 0 为空置邮箱时, 软件置位该位没有任何意义。</p>
位 6: 4	保留	0x0	resd	保持默认值。
位 3	TM0TEF	0x0	rw1c	<p>邮箱 0 发送错误标志 (Transmit mailbox 0 transmission error flag)</p> <p>0: 无错误; 1: 出现错误。</p> <p>注: 当邮箱 0 出现发送错误时置位该位。 可由软件对该位写一清零。或在启动下一次发送时由硬件清除此标志。</p>
位 2	TM0ALF	0x0	rw1c	<p>邮箱 0 仲裁丢失标志 (Transmit mailbox 0 arbitration lost flag)</p> <p>0: 无仲裁问题; 1: 出现仲裁丢失。</p> <p>注: 当邮箱 0 因仲裁丢失导致发送失败时置位该位。 可由软件对该位写一清零。或在启动下一次发送时由硬件清除此标志。</p>
位 1	TM0TSF	0x0	rw1c	<p>邮箱 0 发送成功标志 (Transmit mailbox 0 transmission success flag)</p> <p>0: 发送失败; 1: 发送成功。</p> <p>注: 该位实时指示每次邮箱 0 的发送结果。可由软件对该位写一清零。</p>
位 0	TM0TCF	0x0	rw1c	<p>邮箱 0 发送完成标志 (Transmit mailbox 0 transmission completed flag)</p> <p>0: 正在发送; 1: 发送完成。</p> <p>注: 每次对邮箱 0 的请求 (发送或中止) 完成后, 由硬件置位该位。 该位可由软件写一清零。或当接收到新的发送请求时由硬件自动清除。 当该位被清除时, 邮箱 0 的 TM0TSF、TM0ALF、TM0TEF 也会同步被硬件清除。</p>

### 19.7.1.4 CAN接收FIFO 0寄存器 (CAN\_RF0)

域	简称	复位值	类型	功能
位 31: 6	保留	0x00000000	resd	保持默认值。
位 5	RF0R	0x0	rw1s	释放接收 FIFO 0 (Receive FIFO 0 release) 0: 无意义; 1: 释放 FIFO。 注: 软件设置此位可释放接收 FIFO 0, 当 FIFO 0 被释放时, 硬件对该位清零。 接收 FIFO 0 为空时, 软件置位该位没有任何意义。 若 FIFO 0 中有 2 个以上的报文时, 软件需要执行一次释 放命令后才能访问第 2 个报文。
位 4	RF0OF	0x0	rw1c	接收 FIFO 0 溢出标志 (Receive FIFO 0 overflow flag) 0: 无溢出; 1: 有溢出。 注: 当 FIFO 0 已满时, 又收到了新的符合过滤条件的报文, 硬件将置位该位。 该位由软件写一清零。
位 3	RF0FF	0x0	rw1c	接收 FIFO 0 满标志 (Receive FIFO 0 full flag) 0: 未满; 1: 已满。 注: 当 FIFO 0 中存储 3 笔待读取的报文时, 硬件将置位该 位。 该位由软件写一清零。
位 2	保留	0x0	resd	保持默认值。
位 1: 0	RF0MN	0x0	ro	FIFO 0 报文数目 (Receive FIFO 0 message num) 注: 这 2 位表示存储在 FIFO 0 中的待读取或者处理的报文数 目。 当 FIFO 0 未满时, 每收到了一笔新的符合过滤条件的报 文, 硬件就对 RF0ML 加 1。 每当软件对 RF0R 位写一来释放接收 FIFO 0 时, RF0ML 就会被减 1, 直到其值为 0。

### 19.7.1.5 CAN接收FIFO 1寄存器 (CAN\_RF1)

域	简称	复位值	类型	功能
位 31: 6	保留	0x00000000	resd	保持默认值。
位 5	RF1R	0x0	rw1s	释放接收 FIFO 1 (Receive FIFO 1 release) 0: 无意义; 1: 释放 FIFO。 注: 软件设置此位可释放接收 FIFO 1, 当 FIFO 1 被释放时, 硬件对该位清零。 接收 FIFO 1 为空时, 软件置位该位没有任何意义。 若 FIFO 1 中有 2 个以上的报文时, 软件需要执行一次释 放命令后才能访问第 2 个报文。
位 4	RF1OF	0x0	rw1c	接收 FIFO 1 溢出标志 (Receive FIFO 1 overflow flag) 0: 无溢出; 1: 有溢出。 注: 当 FIFO 1 已满时, 又收到了新的符合过滤条件的报文, 硬件将置位该位。 该位由软件写一清零。
位 3	RF1FF	0x0	rw1c	接收 FIFO 1 满标志 (Receive FIFO 1 full flag) 0: 未满; 1: 已满。 注:

位 2	保留	0x0	resd	当 FIFO 1 中存储 3 笔待读取的报文时，硬件将置位该位。 该位由软件写一清零。
位 1: 0	RF1MN	0x0	ro	FIFO 1 报文数目 (Receive FIFO 1 message num) 注： 这 2 位表示存储在 FIFO 1 中的待读取或者处理的报文数目。 当 FIFO 1 未满时，每收到了一笔新的符合过滤条件的报文，硬件就对 RF1ML 加 1。 每当软件对 RF1R 位写一来释放接收 FIFO 1 时，RF1ML 就会被硬件减 1，直到其值为 0。

### 19.7.1.6 CAN中断使能寄存器 (CAN\_INTEN)

域	简称	复位值	类型	功能
位 31: 18	保留	0x0000	resd	保持默认值。
位 17	EDZIEN	0x0	rw	进入睡眠模式的中断使能 (Enter doze mode interrupt enable) 0: 关闭; 1: 开启。 注：此中断对应的标志位为 EDZIF，故仅本中断使能且 EDZIF 被置位时才会产生中断。
位 16	QDZIEN	0x0	rw	退出睡眠模式的中断使能 (Quit doze mode interrupt enable) 0: 关闭; 1: 开启。 注：此中断对应的标志位为 QDZIF，故仅本中断使能且 QDZIF 被置位时才会产生中断。
位 15	EOIEN	0x0	rw	出现错误的中断使能 (Error occur interrupt enable) 0: 关闭; 1: 开启。 注：此中断对应的标志位为 EOIF，故仅本中断使能且 EOIF 被置位时才会产生中断。
位 14: 12	保留	0x0	resd	保持默认值。
位 11	ETRIEN	0x0	rw	错误类型记录中断使能 (Error type record interrupt enable) 0: 关闭; 1: 开启。 注：只有此中断使能后，硬件设置 ETR[2: 0]时，才会同步设置 EOIF 位为'1'。
位 10	BOIEN	0x0	rw	总线关闭中断使能 (Bus-off interrupt enable) 0: 关闭; 1: 开启。 注：只有此中断使能后，硬件设置 BOF 时，才会同步设置 EOIF 位为'1'。
位 9	EPIEN	0x0	rw	错误被动中断使能 (Error passive interrupt enable) 0: 关闭; 1: 开启。 注：只有此中断使能后，硬件设置 EPF 时，才会同步设置 EOIF 位为'1'。
位 8	EAIEN	0x0	rw	错误警告中断使能 (Error active interrupt enable) 0: 关闭; 1: 开启。 注：只有此中断使能后，硬件设置 EAF 时，才会同步设置 EOIF 位为'1'。
位 7	保留	0x0	resd	保持默认值。
位 6	RF1OIEN	0x0	rw	接收 FIFO 1 溢出中断使能 (Receive FIFO 1 overflow interrupt enable) 0: 关闭; 1: 开启。

				注：此中断对应的标志位为 RF1OF，故仅本中断使能且 RF1OF 被置位时才会产生中断。
位 5	RF1FIEN	0x0	rw	接收 FIFO 1 满中断使能（Receive FIFO 1 full interrupt enable） 0: 关闭； 1: 开启。 注：此中断对应的标志位为 RF1FF，故仅本中断使能且 RF1FF 被置位时才会产生中断。
位 4	RF1MIEN	0x0	rw	接收 FIFO 1 报文接收中断使能（FIFO 1 receive message interrupt enable） 0: 关闭； 1: 开启。 注：此中断对应的标志位为 RF1MN，故仅本中断使能且 RF1MN 为非零时才会产生中断。
位 3	RF0OIEN	0x0	rw	接收 FIFO 0 溢出中断使能（Receive FIFO 0 overflow interrupt enable） 0: 关闭； 1: 开启。 注：此中断对应的标志位为 RF0OF，故仅本中断使能且 RF0OF 被置位时才会产生中断。
位 2	RF0FIEN	0x0	rw	接收 FIFO 0 满中断使能（Receive FIFO 0 full interrupt enable） 0: 关闭； 1: 开启。 注：此中断对应的标志位为 RF0FF，故仅本中断使能且 RF0FF 被置位时才会产生中断。
位 1	RF0MIEN	0x0	rw	接收 FIFO 0 报文接收中断使能（FIFO 0 receive message interrupt enable） 0: 关闭； 1: 开启。 注：此中断对应的标志位为 RF0MN，故仅本中断使能且 RF0MN 为非零时才会产生中断。
位 0	TCIEN	0x0	rw	发送邮箱发送完成中断使能（Transmit mailbox empty interrupt enable） 0: 关闭； 1: 开启。 注：此中断对应的标志位为 TMxTCF，故仅本中断使能且 TMxTCF 被置位时才会产生中断。

### 19.7.1.7 CAN错误状态寄存器 (CAN\_ESTS)

域	简称	复位值	类型	功能
位 31: 24	REC	0x00	ro	接收错误计数器（Receive error counter） 这个计数器按照 CAN 协议的故障界定机制的接收部分实现。
位 23: 16	TEC	0x00	ro	发送错误计数器（Transmit error counter） 这个计数器按照 CAN 协议的故障界定机制的发送部分实现。
位 15: 7	保留	0x00	resd	保持默认值。
位 6: 4	ETR	0x0	rw	错误类型记录（Error type record） 000: 没有错误； 001: 位填充错误； 010: 格式错误； 011: 确认错误； 100: 隐性位错误； 101: 显性位错误； 110: CRC 错误； 111: 由软件设置。 注：

位 3	保留	0x0	resd	这三位用于记录最新错误类型，由硬件根据 CAN 总线上的出错情况设置。当报文被正确发送或接收后，硬件自动将这三位清零。 硬件没有使用错误代码 7，软件可以设置该值，从而可以检测代码的更新。
位 2	BOF	0x0	ro	总线关闭标志 (Bus-off flag) 0: 未处于总线关闭状态； 1: 处于总线关闭状态。 注：当发送错误计数器 TEC 溢出（即大于 255）时，CAN 进入总线关闭状态，硬件对该位置‘1’。
位 1	EPF	0x0	ro	错误被动标志 (Error passive flag) 0: 未处于错误被动状态； 1: 处于错误被动状态。 注：当前记录的出错次数达到错误被动状态（即接收错误计数器或发送错误计数器的值>127）时，硬件对该位置‘1’。
位 0	EAF	0x0	ro	错误主动标志 (Error active flag) 0: 未处于错误主动状态； 1: 处于错误主动状态。 注：当前记录的出错次数达到错误主动状态（即接收错误计数器或发送错误计数器的值≥96）时，硬件对该位置‘1’。

### 19.7.1.8 CAN位时序寄存器 (CAN\_BTMG)

域	简称	复位值	类型	功能
位 31	LOEN	0x0	rw	只听模式使能 (Listen-Only mode) 0: 关闭； 1: 开启。
位 30	LBEN	0x0	rw	回环模式使能 (Loop back mode) 0: 关闭； 1: 开启。
位 29: 26	保留	0x0	resd	保持默认值。
位 25: 24	RSAW	0x1	rw	重新同步调整宽度 (Resynchronization width) $tRSAW = tCAN \times (RSAW[1: 0] + 1)$ 。 注：该位域定义了 CAN 硬件在每位中可以延长或缩短多少个时间单元的上限。
位 23	保留	0x0	resd	保持默认值。
位 22: 20	BTS2	0x2	rw	位时间段 2 (Bit time segment 2) $tBTS2 = tCAN \times (BTS2[2: 0] + 1)$ 。 注：该位域定义了位时间段 2 占用了多少个时间单元。
位 19: 16	BTS1	0x3	rw	位时间段 1 (Bit time segment 1) $tBTS1 = tCAN \times (BTS1[3: 0] + 1)$ 。 注：该位域定义了位时间段 1 占用了多少个时间单元。
位 15: 12	保留	0x0	resd	保持默认值。
位 11: 0	BRDIV	0x000	rw	波特率分频器 (Baud rate division) $tq = (BRDIV[11: 0]+1) \times tPCLK$ 注：该位域定义了时间单元 (tq) 的时间长度。

### 19.7.2 CAN邮箱寄存器

本节描述发送和接收邮箱寄存器。关于寄存器映像的详细信息，请参考 19.6.5 节报文。

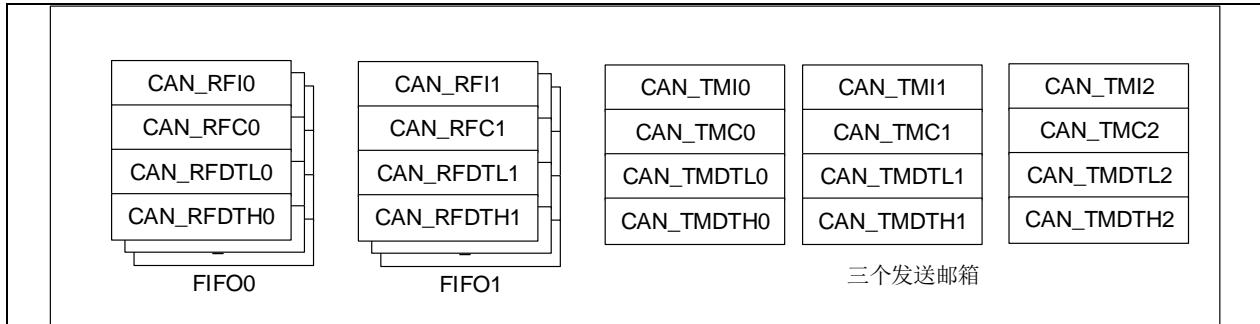
除了下述例外，发送和接收邮箱几乎一样：

- 接收 FIFO 邮箱数据长度和时间戳寄存器 (CAN\_RFCx) 的 RFFMN 域；
- 接收邮箱是只读的；
- 发送邮箱只有在它为空时才是可写的，CAN 发送状态寄存器 (CAN\_TSTS) 的相应 TMxEF 位为‘1’，表示发送邮箱为空。

共有 3 个发送邮箱和 2 个接收邮箱。每个接收邮箱为 3 级深度的 FIFO，并且只能访问 FIFO 中最先收到的报文。

每个邮箱包含 4 个寄存器。

图 19-14 发送和接收邮箱



### 19.7.2.1 发送邮箱标识符寄存器 (CAN\_TMlx) (x=0..2)

注意：1.当其所属的邮箱处在等待发送的状态时，该寄存器是写保护的。

2.该寄存器实现了发送请求控制功能（第 0 位）一复位值为 0。

域	简称	复位值	类型	功能
位 31: 21	TMSID/ TMEID	0xXXX	rw	发送邮箱标准标识符或扩展标识符高字节 (Transmit mailbox standard identifier or extended identifier high bytes) 注：这 11 位为标准标识符或扩展标识符的高 11 位。
位 20: 3	TMEID	0XXXXXX	rw	发送邮箱扩展标识符低字节 (Transmit mailbox extended identifier) 注：这 18 位为扩展标识符的低 18 位。
位 2	TMIDSEL	0xX	rw	标识符类型选择 (Transmit mailbox identifier type select) 0: 标准标识符； 1: 扩展标识符。
位 1	TMFRSEL	0xX	rw	发送邮箱帧类型选择 (Transmit mailbox frame type select) 0: 数据帧； 1: 远程帧。
位 0	TMSR	0x0	rw	发送邮箱的数据发送请求 (transmit mailbox send request) 0: 无意义； 1: 请求发送。 注：当数据发送完成，邮箱为空时，硬件对其清'0'。

### 19.7.2.2 发送邮箱数据长度和时间戳寄存器 (CAN\_TMCx) (x=0..2)

当邮箱不在空置状态时，该寄存器的所有位为写保护。

域	简称	复位值	类型	功能
位 31: 16	TMTS	0XXXX	rw	发送邮箱的报文时间戳 (Transmit mailbox time stamp) 注：该时间戳为报文发送帧起始时刻采样到的 CAN 内部定时器的值。
位 15: 9	保留	0XX	resd	保持默认值。
位 8	TMTSTEN	0X	rw	时间戳的发送使能 (Transmit mailbox time stamp transmit enable) 0: 不发送； 1: 发送。 注： 只有时间触发通信模式使能后，该位才有意义。 时间戳 MTS[15: 0]中，MTS[7: 0]存放于 TMDT7， MTS[15: 8]存放于 TMDT6。故为发送时间戳，发送数据长度需要被设定为 8。
位 7: 4	保留	0X	resd	保持默认值。
位 3: 0	TMDTBL	0X	rw	发送数据长度 (Transmit mailbox data byte length) 注：该域指定了发送报文的数据长度。其中 1 个报文可包含 0 到 8 个字节数据。

### 19.7.2.3 发送邮箱低字节数据寄存器 (CAN\_TMDTx) (x=0..2)

当邮箱不在空置状态时，该寄存器的所有位为写保护。

域	简称	复位值	类型	功能
位 31: 24	TMDT3	0xFF	rw	发送邮箱数据字节 3 (Transmit mailbox data byte 3)
位 23: 16	TMDT2	0xFF	rw	发送邮箱数据字节 2 (Transmit mailbox data byte 2)
位 15: 8	TMDT1	0xFF	rw	发送邮箱数据字节 1 (Transmit mailbox data byte 1)
位 7: 0	TMDT0	0xFF	rw	发送邮箱数据字节 0 (Transmit mailbox data byte 0)

### 19.7.2.4 发送邮箱高字节数据寄存器 (CAN\_TMDTHx) (x=0..2)

当邮箱不在空置状态时，该寄存器的所有位为写保护。

域	简称	复位值	类型	功能
位 31: 24	TMDT7	0xFF	rw	发送邮箱数据字节 7 (Transmit mailbox data byte 7)
位 23: 16	TMDT6	0xFF	rw	发送邮箱数据字节 6 (Transmit mailbox data byte 6) 注：若时间触发通信模式使能，且对应的时间戳的发送使能，则此位将被 TMS[15: 8] 替代。
位 15: 8	TMDT5	0xFF	rw	发送邮箱数据字节 5 (Transmit mailbox data byte 5)
位 7: 0	TMDT4	0xFF	rw	发送邮箱数据字节 4 (Transmit mailbox data byte 4)

### 19.7.2.5 接收FIFO邮箱标识符寄存器 (CAN\_RF1x) (x=0..1)

注意：所有接收邮箱寄存器都是只读的。

域	简称	复位值	类型	功能
位 31: 21	RFSID/RFEID	0xFFFF	ro	接收 FIFO 的标准标识符或扩展标识符 (Receive FIFO standard identifier or receive FIFO extended identifier) 注：这 11 位为标准标识符或扩展标识符的高 11 位。
位 20: 3	RFEID	0xFFFF	ro	接收 FIFO 的扩展标识符 (Receive FIFO extended identifier) 注：这 18 位为扩展标识符的低 18 位。
位 2	RFIDI	0X	ro	接收 FIFO 的标识符类型指示 (Receive FIFO identifier type indication) 0：使用标准标识符； 1：使用扩展标识符。
位 1	RFFRI	0X	ro	接收 FIFO 的帧类型指示 (Receive FIFO frame type indication) 0：数据帧； 1：远程帧。
位 0	保留	0x0	resd	保持默认值。

### 19.7.2.6 接收FIFO邮箱数据长度和时间戳寄存器 (CAN\_RFCx) (x=0..1)

注意：所有接收邮箱寄存器都是只读的。

域	简称	复位值	类型	功能
位 31: 16	RFTS	0xFFFF	ro	接收邮箱的报文时间戳 (Receive FIFO time stamp) 注：该时间戳为报文接收帧起始时刻采样到的 CAN 内部定时器的值。
位 15: 8	RFFMN	0xFF	ro	过滤器匹配序号 (Receive FIFO filter match number) 注：此处存放的是报文通过的那个过滤器序号。
位 7: 4	保留	0X	resd	保持默认值。
位 3: 0	RFDTL	0X	ro	接收数据长度 (Receive FIFO data length) 注：该域指定了接收报文的数据长度。其中 1 个报文可包含 0 到 8 个字节数据。对于远程帧，数据长度 RFDTL 固定为 0。

### 19.7.2.7 接收FIFO邮箱低字节数据寄存器 (CAN\_RFDTLx) (x=0..1)

注意：所有接收邮箱寄存器都是只读的。

域	简称	复位值	类型	功能
位 31: 16	RFDTL	0x0	ro	接收数据长度 (Receive FIFO data length) 注：该域指定了接收报文的数据长度。其中 1 个报文可包含 0 到 8 个字节数据。对于远程帧，数据长度 RFDTL 固定为 0。

位 31: 24	RFDT3	0xXX	ro	接收 FIFO 数据字节 3 (Receive FIFO data byte 3)
位 23: 16	RFDT2	0xXX	ro	接收 FIFO 数据字节 2 (Receive FIFO data byte 2)
位 15: 8	RFDT1	0xXX	ro	接收 FIFO 数据字节 1 (Receive FIFO data byte 1)
位 7: 0	RFDT0	0xXX	ro	接收 FIFO 数据字节 0 (Receive FIFO data byte 0)

### 19.7.2.8 接收 FIFO 邮箱高字节数据寄存器 (CAN\_RFDTx) (x=0..1)

注意：所有接收邮箱寄存器都是只读的。

域	简称	复位值	类型	功能
位 31: 24	RFDT7	0xXX	ro	接收 FIFO 数据字节 7 (Receive FIFO data byte 7)
位 23: 16	RFDT6	0xXX	ro	接收 FIFO 数据字节 6 (Receive FIFO data byte 6)
位 15: 8	RFDT5	0xXX	ro	接收 FIFO 数据字节 5 (Receive FIFO data byte 5)
位 7: 0	RFDT4	0xXX	ro	接收 FIFO 数据字节 4 (Receive FIFO data byte 4)

### 19.7.3 CAN 过滤器寄存器

#### 19.7.3.1 CAN 过滤器控制寄存器 (CAN\_FCTRL)

注意：该寄存器的非保留位完全由软件控制。

域	简称	复位值	类型	功能
位 31: 1	保留	0x160E0700	resd	保持默认值。
位 0	FCS	0x1	rw	过滤器组配置控制开关 (Filters configure switch) 0: 关闭 (过滤器组处于工作状态); 1: 开启 (过滤器组处于配置状态)。 注：过滤器组的初始化配置必须要在过滤器组工作在配置状态下进行。

#### 19.7.3.2 CAN 过滤器模式配置寄存器 (CAN\_FMCFG)

注意：只有在设置 CAN\_FCTRL (FCS=1)，使过滤器处于配置模式下，才能对该寄存器写入。

域	简称	复位值	类型	功能
位 31: 14	保留	0x00000	resd	保持默认值。
位 13: 0	FMSELx	0x0000	rw	过滤器组的模式选择 (Filter mode select) 每一位对应于一个过滤器组 0: 掩码模式; 1: 列表模式。

#### 19.7.3.3 CAN 过滤器位宽配置寄存器 (CAN\_FBWCFG)

注意：只有在设置 CAN\_FCTRL (FCS=1)，使过滤器处于配置模式下，才能对该寄存器写入。

域	简称	复位值	类型	功能
位 31: 14	保留	0x00000	resd	保持默认值。
位 13: 0	FBWSELx	0x0000	rw	过滤器组的位宽选择 (Filter bit width select) 每一位对应于一个过滤器组 0: 2 个 16 位; 1: 单个 32 位。

### 19.7.3.4 CAN过滤器FIFO关联寄存器 (CAN\_FRF)

注意：只有在设置 CAN\_FCTRL (FCS=1)，使过滤器处于初始化模式下，才能对该寄存器写入。

域	简称	复位值	类型	功能
位 31: 14	保留	0x00000	resd	保持默认值。
位 13: 0	FRFSELx	0x0000	rw	过滤器组关联 FIFO 选择 (Filter relation FIFO select) 每一位对应于一个过滤器组 0: 关联 FIFO0; 1: 关联 FIFO1。

### 19.7.3.5 CAN过滤器激活控制寄存器 (CAN\_FACFG)

域	简称	复位值	类型	功能
位 31: 14	保留	0x00000	resd	保持默认值。
位 13: 0	FAENx	0x0000	rw	过滤器组激活使能 (Filter active enable) 每一位对应于一个过滤器组 0: 关闭; 1: 开启。

### 19.7.3.6 CAN过滤器组*i*的过滤位寄存器*x* (CAN\_FiFBx) (其中*i*=0..13; *x*=1..2)

注意：共有 14 组过滤器：*i*=0..13。每组过滤器由 2 个 32 位的寄存器，CAN\_FiFB[2: 1] 组成。

只有在 CAN 过滤器激活控制寄存器 (CAN\_FACFG) 相应的 FAENx 位清'0'，或 CAN 过滤器控制寄存器 (CAN\_FCTRL) 的 FCS 位为'1'时，才能修改相应的过滤器寄存器。

域	简称	复位值	类型	功能
位 31: 0	FFDB	0XXXX XXXX	rw	过滤器过滤数据位 (Filters filter data bit) 列表模式： 寄存器配置值跟总线上接收到的数据对应位的电平完全一致（如果标准帧则忽略扩展帧对应位数值）。 掩码模式： 只有寄存器配置值为'1'的位才跟总线上接收到的数据对应位的电平一致，寄存器配置值为'0'的位不关心。

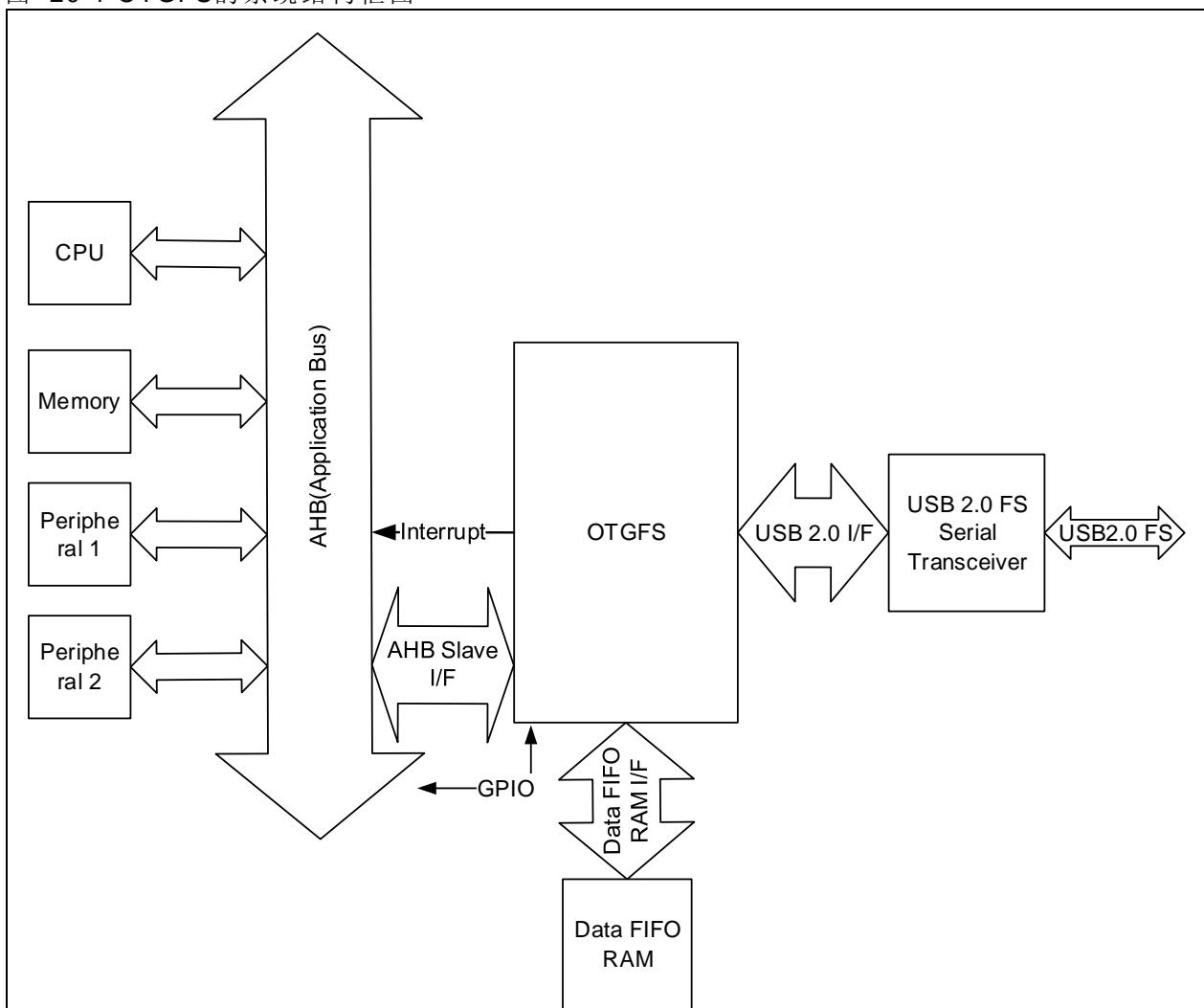
## 20 USB OTG 全速(OTGFS)

OTGFS 部分版权归 Synopsys 公司所有，经许可使用。OTGFS 为全速双重角色设备，符合 Universal Serial Bus Specification, Revision2.0 标准。

### 20.1 OTGFS系统结构框图

下图为 OTGFS 的系统结构框图，OTGFS 模块挂载到 AHB 上，拥有独立的 1280Byte SRAM。

图 20-1 OTGFS的系统结构框图



### 20.2 OTGFS 功能概述

OTGFS 模块由 OTGFS controller、内置物理层（PHY）以及独立 1280 字节 SRAM 组成。

OTGFS 支持控制传输、大容量、中断和同步传输。

OTGFS 是 USB 全速双角色设备（DRD）控制器，由 OTGFS\_USB 配置寄存器（OTGFS\_GUSBCFG）的 FDEVMODE 设定为设备模式；由 OTGFS\_USB 配置寄存器（OTGFS\_GUSBCFG）的 FHSTMODE 设定为主机模式。OTG PHY 内置了 1.5KΩ 上拉电阻和 15KΩ 下拉电阻，以满足双角色设备需要。

OTGFS 作为设备时，支持 1 个双向控制端点，3 个 IN 端点、3 个 OUT 端点，其中端点 3 可以重配置为端点 4；做为主机时，支持 8 个主机通道。

OTGFS 支持 SOF 脉冲功能：发生 SOF 包时产生 SOF 脉冲，SOF 脉冲可输出到定时器 2。

OTGFS 支持挂起模式，进入挂起模式后 OTGFS 处于省电模式。

OTGFS 作为设备时，为所有 OUT 端点分配一个统一的 FIFO 缓存，为每个 IN 端点各自分配一个单独的 FIFO 缓存；OTGFS 作为主机时，为所有的接受通道分配了一个统一的接收 FIFO，为所有非周期性发送

通道分配一个统一的发送 FIFO，为所有周期性发送通道分配一个统一的发送 FIFO。

OTGFS 支持挂起模式，在 OTGFS 处于低功耗模式；OTGFS 还可以通过配置 OTGFS 通用控制器配置寄存器(OTGFS\_GCCFG)的 PWRDOWN 位或 OTGFS 电源和时钟门控寄存器(OTGFS\_PCGCCTL)的 STOPPCLK 位达到低功耗效果。

## 20.3 OTGFS时钟与管脚配置

### 20.3.1 OTGFS时钟配置

OTGFS 接口中存在两个时钟：USB 控制时钟和 AHB 总线时钟。USB 全速设备总线速度标准为  $12\text{Mb/s} \pm 0.25\%$ 。因此需要给 OTGFS 提供  $48\text{MHz} \pm 0.25\%$  的时钟频率用于 USB 总线采样。

OTGFS 48M 时钟来源：

- 通过 PLL 分频

注意 PLL 的输出频率要满足 USBDIV（参考时钟配置寄存器（CRM\_CFG））能够正常分频到  $48\text{MHz}$ 。

注意：使用 OTGFS 时，AHB 时钟频率必须大于  $30\text{Mhz}$

### 20.3.2 OTGFS管脚配置

OTGFS 的输入输出均与 GPIO 复用，当满足以下条件时 GPIO 被用于 OTGFS 功能。

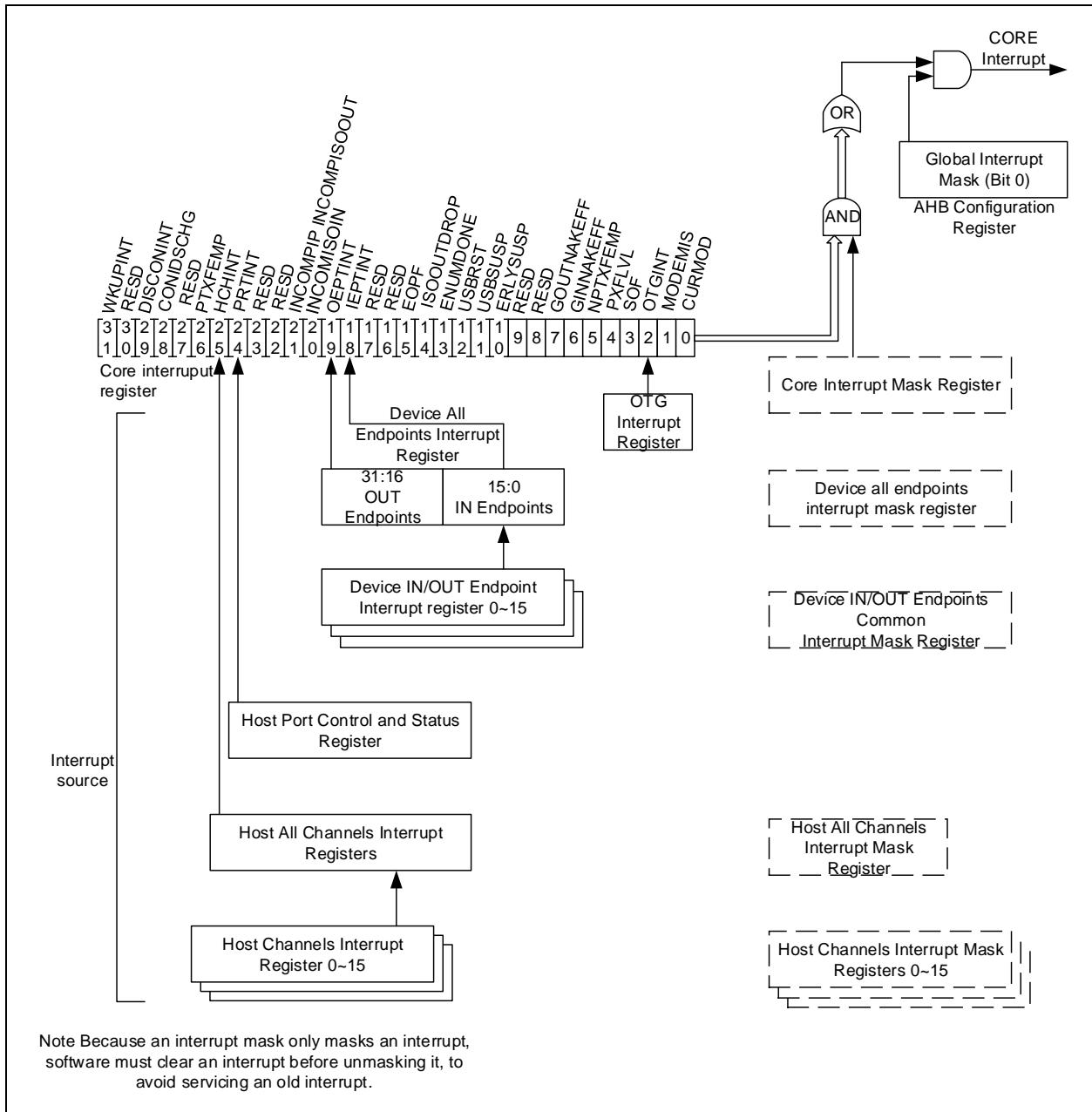
表 20-1 OTGFS输入/输出引脚

信号名称	GPIO	条件
OTGFS_D-	PA11	在 CRM 中使能 OTG 模块，配置 PWRDOWN 位设置为 1
OTGFS_D+	PA12	在 CRM 中使能 OTG 模块，配置 PWRDOWN 位设置为 1

## 20.4 OTGFS中断

OTGFS 中断结构示意图如图 20-2 所示，功能详见 OTGFS 中断寄存器(OTGFS\_GINTSTS)和 OTGFS 中断屏蔽寄存器(OTGFS\_GINTMSK)。

图 20-2 OTGFS 中断结构示意图



## 20.5 OTGFS 功能操作

### 20.5.1 OTGFS 初始化

如果上电时线缆已连接，那么控制器中断寄存器的当前操作模式位（CURMOD）指示当前工作模式。当连接 A 类插头时，OTGFS 控制器工作在主机模式；当连接 B 类插头时，控制器工作在设备模式。本节介绍了上电后控制器的初始化操作。无论控制器处于主机模式还是设备模式，应用程序都必须遵循初始化顺序。所有的控制器全局寄存器都须按照控制器的配置进行初始化操作。

1、在全局 AHB 配置寄存器中设置以下位域：

- 全局中断屏蔽位 = 0x1
- 非周期性发送 FIFO 空级别
- 周期性发送 FIFO 空级别

2、设置全局中断屏蔽寄存器的下列位域：

- OTGFS\_GINTMSK.RXFLVLMSK = 0x0

- 3、设置 OTGFS\_USB 配置寄存器(OTGFS\_GUSBCFG)的下列位域:
  - 全速超时标准位
  - USB 周转时间位
- 4、软件必须解除 OTGFS 中断屏蔽寄存器(OTGFS\_GINTMSK)中的下列位的中断屏蔽:
  - OTG 中断屏蔽
  - 模式不匹配中断屏蔽
- 5、软件通过读取当前运行模式寄存器(OTGFS\_GINTSTS.CURMOD)位来确定 OTGFS 控制器是处于主机模式还是设备模式。

## 20.5.2 OTGFS FIFO配置

### 20.5.2.1 设备模式

在上电复位或 USB 复位时，需要进行动态 FIFO 重新分配。在设备模式下，应用程序在更改 FIFO 数据的 SRAM 分配之前，必须先确保满足下列条件：

- OTGFS\_DIEPCTLx/ OTGFS\_DOEPCTLx.EPENA = 0x0
- OTGFS\_DIEPCTLx/ OTGFS\_DOEPCTLx.NAKSTS = 0x1

通过发送 FIFO 编号寄存器 (OTGFS\_GRSTCTL.TXFNUM) 位刷新控制器中的发送 FIFO。关于如何刷新发送 FIFO，详见“刷新控制器发送 FIFO”章节。

在为 FIFO 分配 SRAM 空间时，必须注意以下几点：

#### (1) 接收 FIFO SRAM 分配

- 用于 SETUP 包的 SRAM：必须预留 13 个 DWORDs 才能接收控制端点发出的 1 个 SETUP 包，这些位置是预留给 SETUP 写数据的，控制器并不会占用。
- 为全局 OUT NAK 预留 1 个 DWORD
- 状态信息与每次收到的包一起写入到 FIFO。因而，必须分配至少一个(包最大长度 /4)+1 这样的空间来接收数据包。如果使能了多个同步端点，那么至少需要分配两个(包最大长度 /4)+1 这样的空间来接收连续的数据包。一般情况下，建议分配两个(包最大长度 /4)+1 的空间以确保当前一个数据包正传输给 AHB 时，USB 可以接收到下一个数据包。如果 AHB 延迟较长，必须配置足够的空间来接收多个数据包，以防止丢失同步数据包。
- 传输完成的状态信息，连同每个端点的最后一个数据包，一起被推送到 FIFO。
- 必须为每个端点的禁止状态位预留一个位置。
- 通常建议为每个 OUT 端点配置两个 DWORDs。

#### (2) 发送 FIFO SRAM 分配

每个 IN 端点发送 FIFO 所需的最小 SRAM 空间就是该 IN 端点的最大数据包长度。发送 IN 端点 FIFO 分配的空间越多，USB 性能越好，并且还能避开 AHB 线上的延迟。

表 20-2 OTGFS发送 FIFO SRAM分配表

FIFO 名称	SRAM 大小
接收 FIFO	rx_fifo_size, 包含 setup 包, OUT 端点控制信息以及数据 OUT 包。
发送 FIFO 0	tx_fifo_size[0]
发送 FIFO 1	tx_fifo_size[1]
发送 FIFO 2	tx_fifo_size[2]
.....	.....
发送 FIFO i	tx_fifo_size[i]

根据以上信息配置下列寄存器：

1. OTGFS 接收 FIFO 长度寄存器(OTGFS\_GRXFSIZ)
  - OTGFS\_GRXFSIZ.RXFDEP = rx\_fifo\_size;
2. 端点 0 TX FIFO 长度寄存器(OTGFS\_DIEPTXF0)
  - OTGFS\_DIEPTXF0.INEPT0TXDEP = tx\_fifo\_size[0];
  - OTGFS\_DIEPTXF0.INEPT0TXSTADDR = rx\_fifo\_size;
3. 设备 IN 端点发送 FIFO#1 长度寄存器(OTGFS\_DIEPTXF1)
  - OTGFS\_DIEPTXF1.INEPTXFSTADDR = OTGFS\_GNPTXFSIZ.NPTXFSTADDR + tx\_fifo\_size[0];
4. 设备 IN 端点发送 FIFO#2 长度寄存器(OTGFS\_DIEPTXF2)

- OTGFS\_DIEPTXF2.INEPTXFSTADDR = OTGFS\_DIEPTXF1.INEPTXFSTADDR + tx\_fifo\_size[1];
- 5. 设备 IN 端点发送 FIFO#i 长度寄存器(OTGFS\_DIEPTXF*i*)
- OTGFS\_DIEPTXF*i*.INEPTXFSTADDR = OTGFS\_DIEPTXF*i*-1.INEPTXFSTADDR + tx\_fifo\_size[i-1];
- 6. 完成 SRAM 分配后必须刷新发送 FIFO 和接收 FIFO，以确保 FIFO 正常运行。
- OTGFS\_GRSTCTL.TXFNUM = 0x10
- OTGFS\_GRSTCTL.TXFFLSH = 0x1
- OTGFS\_GRSTCTL.RXFFLSH = 0x1

应用程序必须要等到 TXFFLSH 位和 RXFFLSH 位清除后才能在控制器上执行其它操作。

### 20.5.2.2 主机模式

在主机模式下，应用程序在更改 FIFO 数据的 SRAM 分配之前，必须先确认下列信息：

- 所有通道已禁用
- 所有 FIFO 为空

重新分配完 FIFO 数据的 SRAM 后，应用程序必须通过发送 FIFO 编号寄存器 (OTGFS\_GRSTCTL.TXFNUM) 刷新位来刷新控制器中的所有 FIFO。

在完成重新分配后，必须通过刷新来复位 FIFO 中的指针以确保 FIFO 正常运行。关于刷新发送 FIFO 的详细信息，请参考“刷新控制器发送 FIFO”。

#### (1) 接收 FIFO SRAM 分配

状态信息与每次接收的数据包一起写入 FIFO。因此，必须分配至少一个(包最大长度 / 4)+2 这样的空间来接收数据包。如果使能了多个同步端点，那么至少需要分配两个(包最大长度 / 4)+2 这样的空间来接收背靠背数据包。一般情况下，建议分配两个(包最大长度 / 4)+2 的空间以便当前一个数据包正传输给 AHB 时，USB 可以接收到下一个数据包。如果 AHB 延迟较长，必须配置足够的空间来接收多个数据包。

传输完成状态信息和通道中止信息会跟着每个主机通道的最后一个数据包一起推送给 FIFO。为此，必须多分配两个 DWORDs。

#### (2) 发送 FIFO SRAM 分配

主机非周期性发送 FIFO 所需要的最小 SRAM 空间就是所有非周期性 OUT 通道的最大包长度。发送非周期性 FIFO 的空间越多，USB 性能越好，并且还能避开 AHB 线上的延迟。通常，建议分配两个最大包长度，以便当前一个数据包正传输给 USB 时，AHB 可以接收到下一个数据包。如果 AHB 延迟较长，必须配置足够的空间来缓冲多个数据包。

主机周期性发送 FIFO 所需的最小 SRAM 空间就是所有周期性 OUT 通道的最大包长度。

#### (3) 内部寄存器存储空间分配

表 20-3 OTGFS 内部寄存器存储空间分配表

FIFO Name	Data SRAM Size
接收数据 FIFO	rx_fifo_size
非周期性发送 FIFO	tx_fifo_size[0]
周期性发送 FIFO	tx_fifo_size[1]

根据这些信息来配置下列寄存器：

1. OTGFS 接收 FIFO 长度寄存器(OTGFS\_GRXFSIZ)
- OTGFS\_GRXFSIZ.RXFDEP = rx\_fifo\_size;
2. OTGFS 非周期性 TX FIFO 长度寄存器(OTGFS\_GNPTXFSIZ)
- OTGFS\_GNPTXFSIZ.NPTXFDEP = tx\_fifo\_size[0];
- OTGFS\_GNPTXFSIZ.NPTXFSTADDR = rx\_fifo\_size;
3. OTGFS 主机周期性发送 FIFO 长度寄存器(OTGFS\_HPTXFSIZ)
- OTGFS\_HPTXFSIZ.PTXFSIZE = tx\_fifo\_size[1];
- OTGFS\_HPTXFSIZ.PTXFSTADDR = OTGFS\_GNPTXFSIZ.NPTXFSTADDR + tx\_fifo\_size[0];
4. 在完成 SRAM 分配之后，必须刷新发送 FIFO 和接收 FIFO 以确保 FIFO 正常运行。
- OTGFS\_GRSTCTL.TXFNUM = 0x10
- OTGFS\_GRSTCTL.TXFFLSH = 0x1
- OTGFS\_GRSTCTL.RXFFLSH = 0x1
- 应用程序必须等到 TXFFLSH 位和 RXFFLSH 位清除之后才能执行其它操作。

### 20.5.2.3 刷新控制器发送 FIFO

应用程序通过 OTGFS\_GRSTCTL.TXFFLSH 来刷新控制器中的所有发送 FIFO，流程如下：

- 检查 OTGFS\_GINTSTS.GINNAKEFF 是否为 0，如果该位已清除，则设置全局 IN NAK 寄存器 (OTGFS\_DCTL.SGNPINNAK) 为 0x1。NAK 有效中断置位表示控制器未读取 FIFO。
- 等待 OTGFS\_GINTSTS.GINNAKEFF = 0x1，表示针对所有 IN 端点的 NAK 设置已生效。
- 轮询 AHB 主机空闲寄存器 (OTGFS\_GRSTCTL.AHBIDLE) 直到其置为 1，AHBIDLE = H 表示控制器没有写入 FIFO。
- 确认 OTGFS\_GRSTCTL.TXFFLSH = 0x0。如果置 0，则将你需要刷新的发送 FIFO 编号写入到发送 FIFO 编号寄存器 (OTGFS\_GRSTCTL.TXFNUM)。
- 设置 OTGFS\_GRSTCTL.TXFFLSH = 0x1，然后等待清除。
- 置起清除全局 IN NAK 寄存器 (OTGFS\_DCTL.CGNPINNAK) 位。

### 20.5.3 OTGFS 主机模式

在 OTGFS 作为主机时，控制器内部不能为 VBUS 提供 5V 电压源，需要外部电压泵持续为 VBUS 供电。

#### 20.5.3.1 主机初始化

应用程序必须遵循以下步骤来初始化控制器：

1. 将主机端口中断屏蔽寄存器 (OTGFS\_GINTMSK.PRTINTMSK) 设置为解除中断屏蔽。
2. 设置 OTGFS 主机模式配置寄存器(OTGFS\_HCFG)。
3. 设置 OTGFS\_HPRT.PRTPOWER 位为 0x1，驱动 USB 线上的 VBUS 供电。
4. 等待端口连接检测寄存器 (OTGFS\_HPRT0.PRTCONDET) 中断，该中断表示设备连接到端口。
5. 设置端口复位寄存器 (OTGFS\_HPRT.PRTRST) 位为 0x1，发起复位操作。
6. 等待至少 10 ms，确保完成复位。
7. 设置端口复位寄存器 (OTGFS\_HPRT.PRTRST) 位为 0x0
8. 等待端口使能/禁止状态改变寄存器 (OTGFS\_HPRT.PRTENCHNG) 中断。
9. 读取端口速度寄存器 (OTGFS\_HPRT.PRTSPD) 位，获取枚举速度。
10. 根据已选择的 PHY 时钟值来配置 HFIR 寄存器。
11. 设置 OTGFS 接收 FIFO 长度寄存器(OTGFS\_GRXFSIZ)，选择接收 FIFO 的长度。
12. 设置 OTGFS 非周期性 TX FIFO 长度寄存器(OTGFS\_GNPTXFSIZ)，选择非周期发送 FIFO 的起始地址和长度。
13. 设置 OTGFS 主机周期性发送 FIFO 长度寄存器(OTGFS\_HPTXFSIZ)，选择周期性发送 FIFO 的起始地址和和长度。

为了与设备通信，应用程序必须按照“OTGFS 通道初始化”的要求使能和初始化至少一个通道。

#### 20.5.3.2 OTGFS 通道初始化

为了与设备通信，应用程序必须使能和初始化至少一个通道。应用程序可遵循下列步骤来使能和初始化通道：

1. 设置 OTGFS 中断屏蔽寄存器(OTGFS\_GINTMSK)解除下列中断屏蔽：
  - 用于 OUT 传输的非周期性发送 FIFO 空
  - 用于 OUT 传输的非周期性发送 FIFO 半空
2. 设置 OTGFS 主机所有通道中断屏蔽寄存器(OTGFS\_HAINTMSK)解除所选通道的中断屏蔽
3. 设置 OTGFS 主机通道 x 中断屏蔽寄存器(OTGFS\_HCINTMSKx)解除主机通道中断寄存器中与传输相关的中断屏蔽
4. 为所选通道的 OTGFS 主机通道 x 传输长度寄存器(OTGFS\_HCTSIZx)配置传输总长度（以字节为单位），以及期望接收的包数目，包括短数据包。应用程序必须根据初始数据 PID (用于首个 OUT 传输的 PID，或者期望从首个 IN 传输收到的数据 PID)来配置 PID 位。
5. 设置传输长度，确保该通道的传输长度是最大包长度的倍数。
6. 根据设备端点特性，比如类型，速度，方向等来配置所选通道的 OTGFS 主机通道 x 特性寄存器 (OTGFS\_HCCHARx) (只有当应用程序准备好传输或接收数据包时，才能通过设置通道使能位为 1 来使能通道)

### 20.5.3.3 中止通道

应用程序可以通过将 OTGFS 主机通道  $x$  特性寄存器(OTGFS\_HCCHAR $x$ )的通道禁止寄存器(HCCHAR $x$ .CHDIS)和通道使能寄存器(OTGFS\_HCCHAR $x$ .CHENA)位设置为 0x1 来中止通道。这个操作将使能主机刷新已提交的请求(如果有的话)并生成“通道中止”中断。应用程序必须等待通道中止寄存器(OTGFS\_HCINT $x$ .CHHLTD)中断生成之后,才能重新为其他传输事务分配通道。主机不会中断已经在 USB 线上启动的传输任务。

应用程序在中止通道前,必须确保非周期性请求队列(当中止非周期性通道时)或者周期性请求队列(当中止周期性通道时)中至少有一个可用空间。当请求队列满额时(在中止通道前),应用程序可以通过将 OTGFS 主机通道  $x$  特性寄存器(OTGFS\_HCCHAR $x$ )中的通道禁止寄存器(HCCHAR $x$ .CHDIS)位设置为 0x1 以及将通道使能寄存器(OTGFS\_HCCHAR $x$ .CHENA)位设置为 0 来刷新已提交的请求。

当请求队列里有传输输入时,控制器会触发 RXFLVL 中断。应用程序必须通过读取/弹出 OTGFS 状态读和 POP 寄存器(OTGFS\_GRXSTSP)来生成“通道中止”中断。

应用程序应当在发生下列情况时中止通道:

- 在非周期性 IN 传输期间检测到了传输完成寄存器(OTGFS\_HCINT $x$ .XFERC)中断;
- 当 IN 或 OUT 通道收到了 OTGFS\_HCINT $x$ .STALL、OTGFS\_HCINT $x$ .XACTERR、OTGFS\_HCINT $x$ .BBLEERR 或者 OTGFS\_HCINT $x$ .DTGLERR 中断信号时。
- 当收到 OTGFS\_GINTSTS.DISCONNINT(设备断开)中断事件时。应用程序必须检查端口连接状态寄存器(OTGFS\_HPRT.PRTCONSTS)信号,这是因为当设备与主机断开时,端口连接状态寄存器(OTGFS\_HPRT.PRTCONSTS)将复位。应用程序必须启动软件复位以确保所有通道已清除。当设备重新连接时,主机必须启动 USB 复位。
- 当应用程序需要在正常传输结束之前中止传输时。

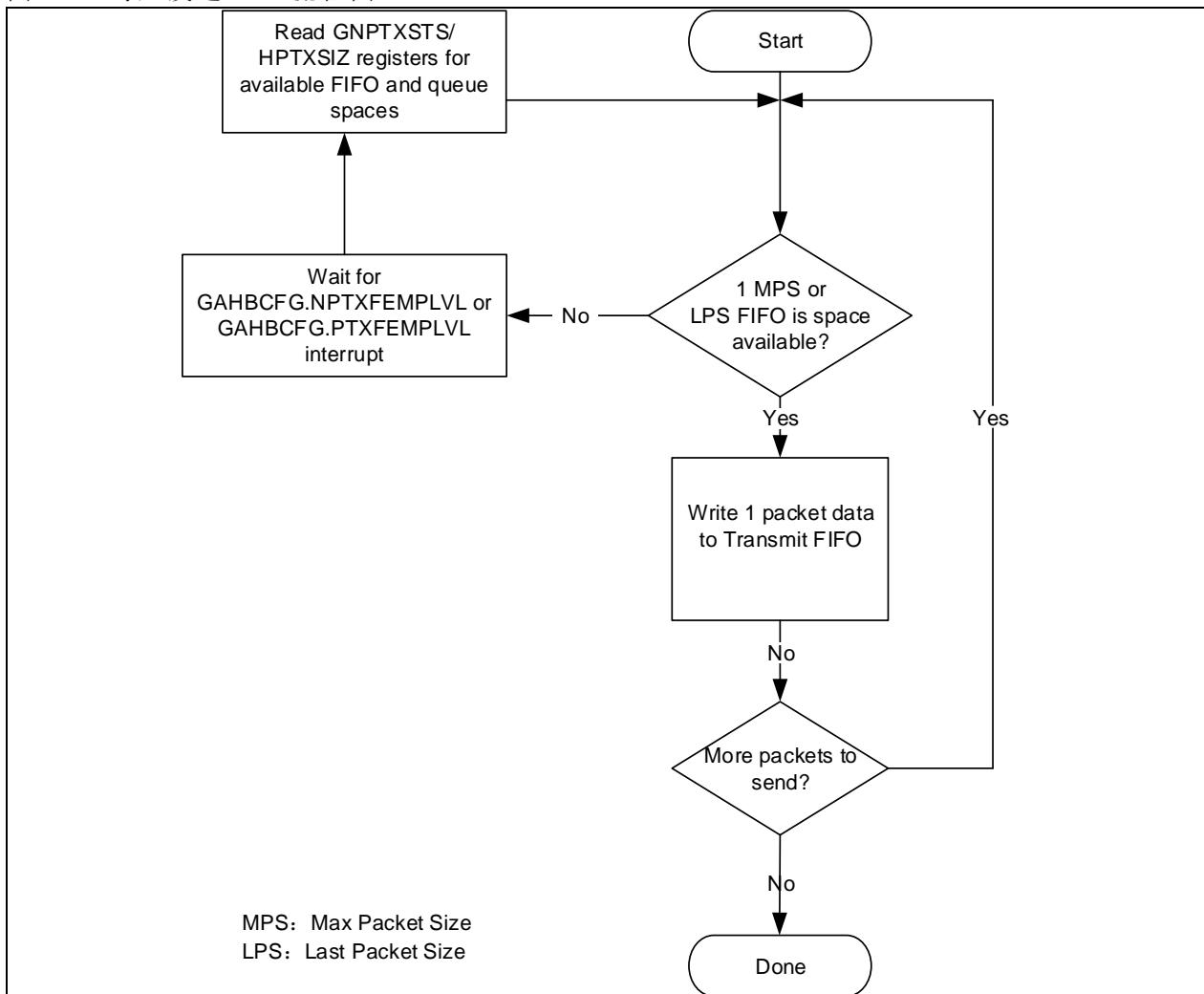
### 20.5.3.4 队列深度

在周期性硬件传输请求队列中支持最多 8 个中断和同步传输请求;在非周期性硬件传输请求队列中支持最多 8 个控制和大容量传输的传输请求。

- 写入发送 FIFO

下图显示写入发送 FIFO 的流程图。OTGFS 主机会在最后一个 DWORD 写数据包时,自动将一个请求(OUT 请求)写入周期性/非周期性请求队列。在开始写入 FIFO 之前,应用程序必须保证周期性/非周期性请求队列里至少有一个可用空间。应用程序只能以 DWORD 方式写入发送 FIFO。如果包长度没有对齐 DWORD,那么应用程序必须填充数据位。OTGFS 主机根据所设置的最大包长度和传输长度来确定实际的包长度。

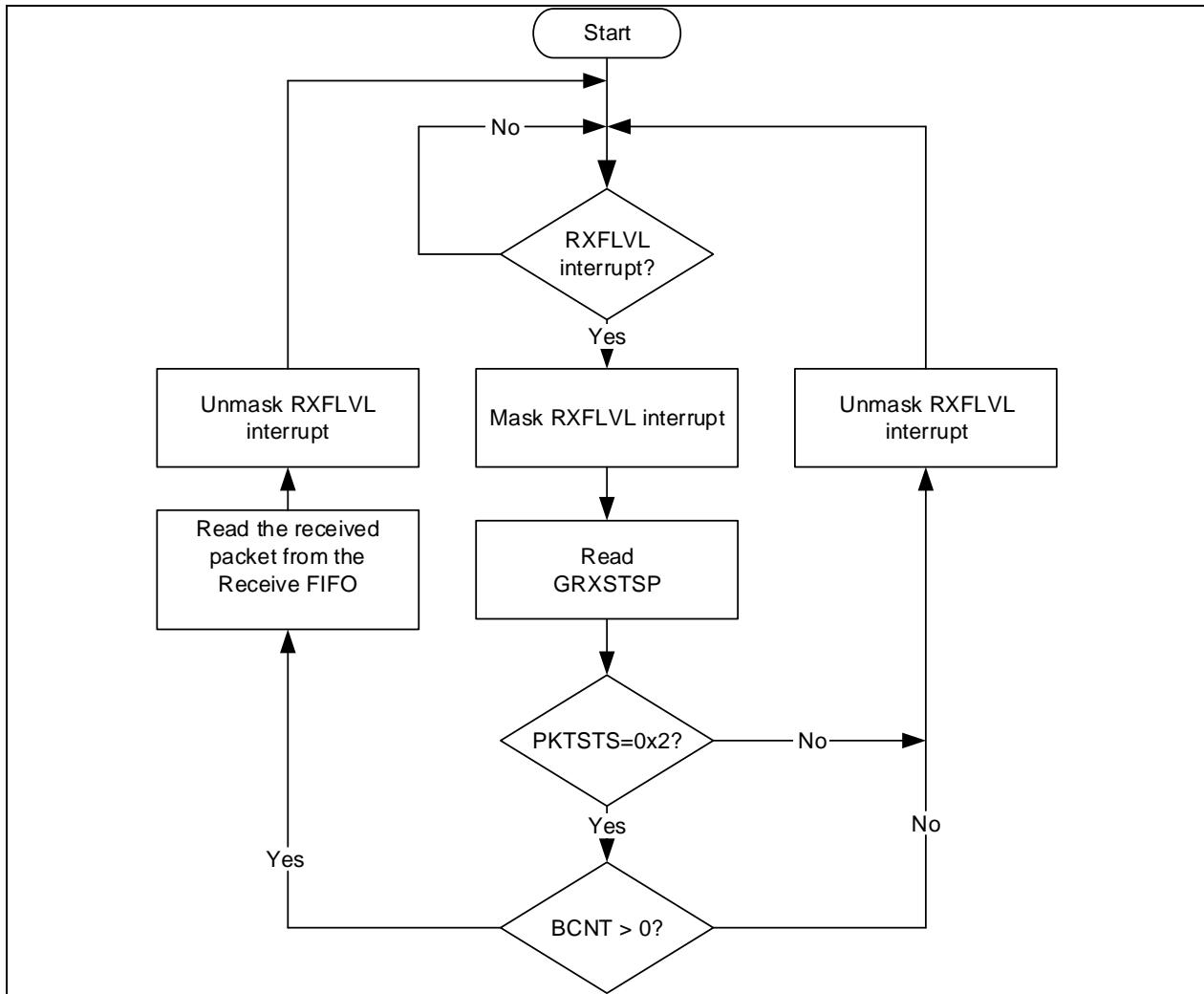
图 20-3 写入发送 FIFO 流程图



### ● 读取接收 FIFO

下图显示读取接收 FIFO 的流程图。应用程序需要忽略除了 IN 数据包(0x0010)以外的所有包状态。

图 20-4 读取接收 FIFO 流程图



### 20.5.3.5 特殊情况处理

#### (1) 处理 Babble 状况

OTGFS 处理两种 babble 情况：即包 babble 和端口 babble。如果设备发出的数据超过该通道的最大包长度时将发生包 babble。如果控制器在 EOF2（即帧 2 末尾，非常接近 SOF）时持续接收设备发出的数据时会产生端口 Babble。

当检测到包 babble 时，OTGFS 将停止写入接收缓冲区并等待包结束。当检测到包结束时，OTGFS 将清空已写入接收缓冲区的数据并生成 Babble 中断。

当检测到端口 babble 时，OTGFS 会清空接收 FIFO 并禁用该端口，然后，控制器生成“端口禁用中断”。一旦收到该中断，应用程序需要通过确认端口过流有效位寄存器 (HPRT.PRTOVRCACT) 信号来确定端口中断并不是因为过流引发的（这是导致端口禁用中断的另一个原因），然后进行软件复位。控制器在检测到端口 babble 信号时不再发送令牌。

#### (2) 处理设备断开

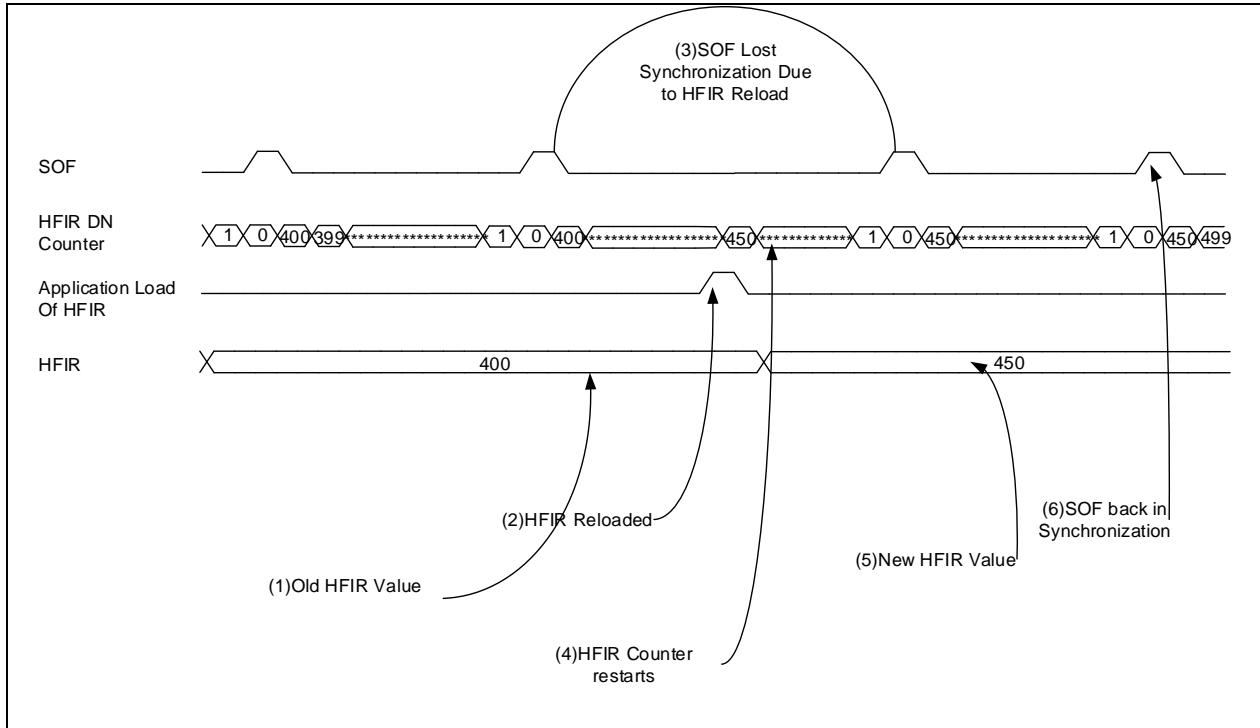
如果设备突然断开，则会生成一个检测到断开事件产生中断寄存器 (OTGFS\_GINTSTS.DISCONINT) 中断。应用程序在收到该中断时，必须通过设置控制器软件复位寄存器 (OTGFS\_GRSTCTL.CSFTRST) 来启动软件复位。

### 20.5.3.6 主机HFIR功能

主机帧间隔寄存器(HFIR)定义了两个连续 SOF(全速)或者 Keep-Alive 令牌之间的间隔。此位域包含了构成所需帧间隔的 PHY 时钟数，主要用于根据 PHY 时钟频率来调整 SOF 持续时间。

将重新加载控制寄存器 (OTGFS\_HFIR.HFIRRLDCTRL) 设置为 0x0 时的 HFIR 行为，如下图。

图 20-5 HFIRRLDCTRL 为 0x0 时的 HFIR 行为

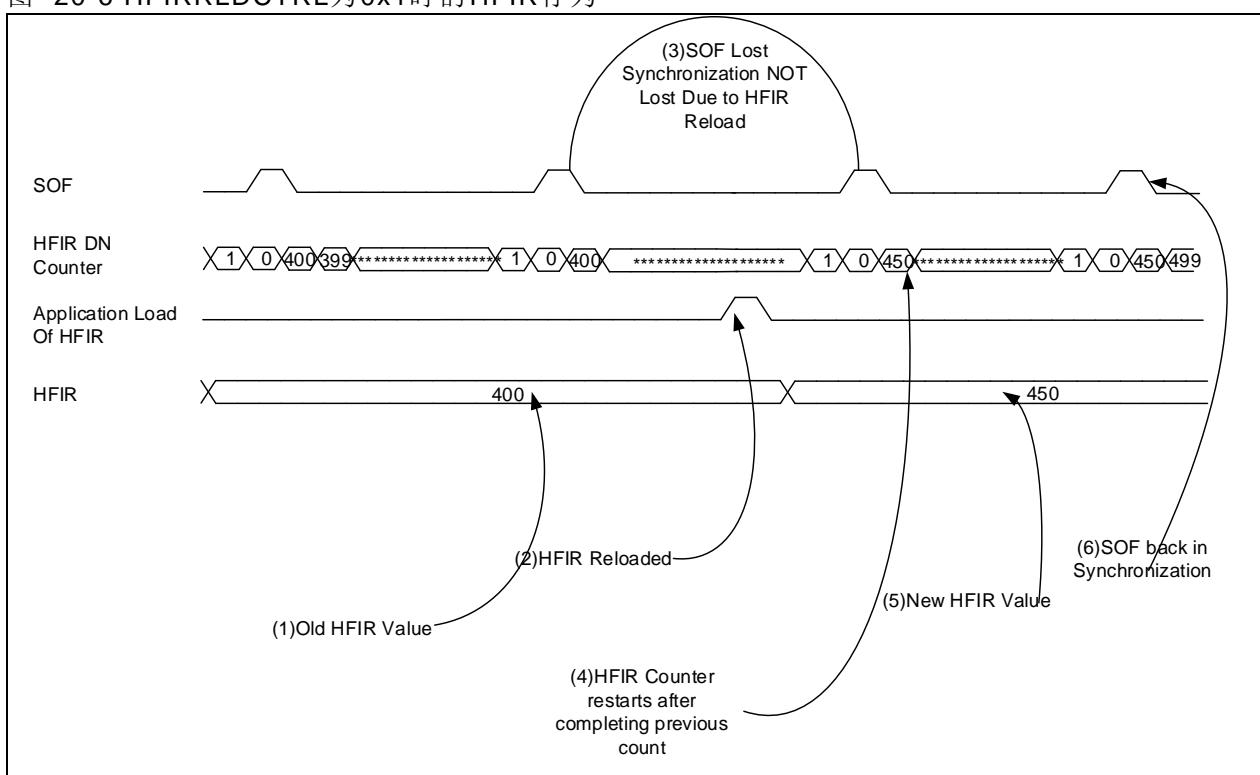


步骤如下所示：

1. 上电复位后，显示应用程序当前设定的 HFIR 值
2. 应用程序加载新值到 HFIR 寄存器
3. 由于 HFIR 向下计数器重新加载，它会立即开始重新计数，从而丢失 SOF 同步
4. 重启 HFIR 计数器
5. HFIR 寄存器接收到新的设定值
6. 使用 HFIR 新功能生成首个 SOF 后，又再次获得 SOF 同步。

当重新加载控制寄存器 (OTGFS\_HFIR.HFIRRLDCTRL) 为 0x1 时的 HFIR 行为，如下图。

图 20-6 HFIRRLDCTRL 为 0x1 时的 HFIR 行为



所示的步骤如下：

- 1.上电复位后，显示应用程序当前设定的 HFIR 值
  - 2.应用程序加载新的 HFIR 值；HFIR 计数器不采用新值，而是继续计数直到计数器达到 0
  - 3.当计数器在使用旧 HFIR 值计数到 0 时，生成 SOF
  - 4.HFIR 计数器采用新值
  - 5.新的 HFIR 值生效
- 经过以上步骤 SOF 恢复同步。

### 20.5.3.7 初始化批量IN传输/控制IN传输

图 20-7 显示典型的批量 IN 传输/控制 IN 传输的操作流程，详见通道 2(ch\_2)。假设：

- 应用程序正在尝试接收两个最大长度的数据包（传输长度为 64 字节）
- 接收 FIFO 包含至少一个最大包长度的数据包以及每个数据包的两个状态 DWORDs（对于全速传输有 72 字节）
- 非周期性请求队列深度为 4

#### (1) 普通批量 IN 传输和控制 IN 传输的操作流程

图 20-7 所示的操作顺序如下：

1. 初始化通道 2（按照“OTGFS 通道初始化”的要求）
2. 置起通道使能寄存器 (OTGFS\_HCCHAR2.CHENA) 位，向非周期性请求队列写入一个 IN 请求
3. 控制器在完成当前 OUT 传输后会发送一个 IN 令牌
4. 一旦将收到的数据包写入接收 FIFO，控制器即生成 RXFLVL 中断。
5. 为处理 RXFLVL 中断，需要先屏蔽 RXFLVL 中断，并读取接收数据包状态以确认收到的字节数，然后再读取接收 FIFO。按照这个步骤可以解除 RXFLVL 中断屏蔽。
6. 传输完成状态被写入接收 FIFO 时控制器会产生 RXFLVL 中断。
7. 应用程序必须读取接收的数据包状态，当接收的数据包不是 IN 数据包时，则不理会它。
8. 控制器在接收到的数据包被读取之后会生成 XFERC 中断。
9. 为处理 XFERC 中断，需要先中止通道（详见“中止通道”），并停止写入 OTGFS 主机通道 2 特性寄存器(OTGFS\_HCCHAR2)。控制器会在 OTGFS 主机通道 2 特性寄存器(OTGFS\_HCCHAR2)被写入后向非周期性请求队列写入通道中止请求。
10. 中止状态被写入接收 FIFO 时控制器会生成 RXFLVL 中断。
11. 读取接收数据包状态，但不予处理。
12. 一旦从接收 FIFO 中读出中止状态，控制器即会生成 CHHLTD 中断。
13. 为响应这个 CHHLTD 中断，处理器不会为其它传输分配这个通道。

#### (2) 处理中断

下列代码描述了批量传输和 IN 传输流程中与通道相关的中断服务程序：

```

Unmask (XACTERR/XFERC/BBLERR/STALL/DATATGLERR)
if (XFERC)
{
    Reset Error Count
    Unmask CHHLTD
    Disable Channel
    Reset Error Count
    Mask ACK
}
else if (XACTERR or BBLERR or STALL)
{
    Unmask CHHLTD
    Disable Channel
    if (XACTERR)
    {
        Increment Error Count
        Unmask ACK
    }
}
else if (ChHltd)
{
    Mask CHHLTD
    if (Transfer Done or (Error_count == 3))
}

```

```
{  
    De-allocate Channel  
}  
else  
{  
    Re-initialize Channel  
}  
}  
else if (ACK)  
{  
    Reset Error Count  
    Mask ACK  
}  
else if (DATATGLERR)  
{  
    Reset Error Count  
}
```

### 20.5.3.8 初始化批量和控制OUT/SETUP传输

图 20-7 介绍了典型的批量传输或控制传输的 OUT/SETUP 操作流程。详见通道 1 (ch\_1)。需要发送 2 个批量传输的 OUT 数据包。控制传输的 SETUP 操作流程也相同，只是只有一个数据包。假设：

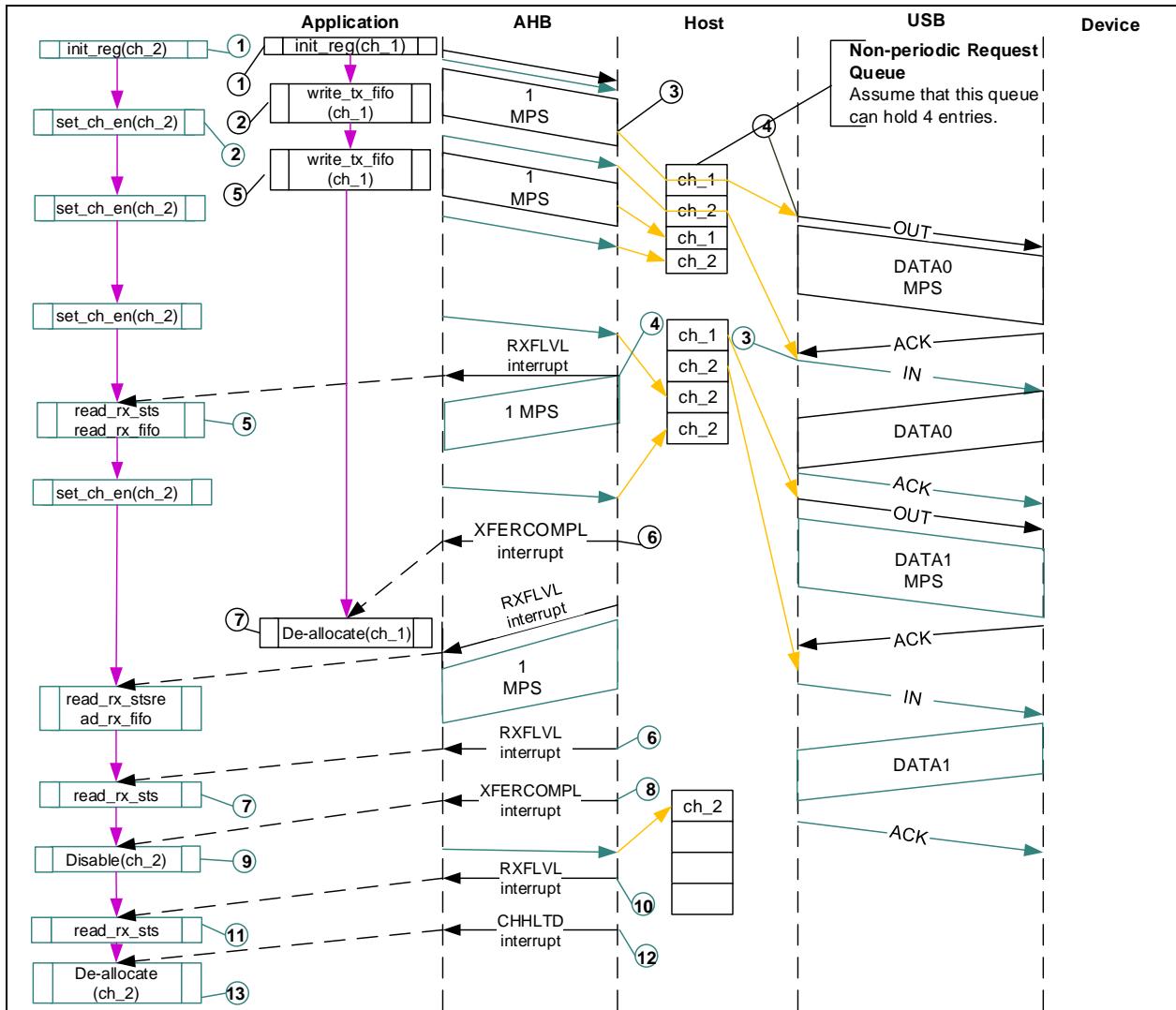
- 应用程序正在尝试发送两个最大包长度的数据包（传输长度为 64 字节）
- 非周期性发送 FIFO 可以保存 2 个数据包（对于全速传输有 128 字节）
- 非周期性请求队列深度为 4。

#### (1) 普通批量传输和控制传输的 OUT/SETUP 操作流程

图 20-7 所示的操作流程如下：

1. 初始化通道 1（按照“OTGFS 通道初始化”步骤）
2. 写通道 1 的第一个数据包
3. 随着最后一个 DWORD 写入，控制器向非周期性请求队列写入一个请求。
4. 一旦非周期性队列变为非空，控制器就会在当前帧帧内发送一个 OUT 令牌。
5. 向通道 1 写入第二个（最后一个）数据包
6. 在前一个传输成功完成后，控制器就会生成 XFERC 中断。
7. 为响应 XFERC 中断，控制器不会为其他传输分配这个通道。

图 20-7 普通 Bulk/Control OUT/SETUP 和 Bulk/Control IN 传输例程示例图



## (2) 处理中断

下列代码给出了批量传输、控制传输 OUT/SETUP 流程中与通道相关的中断服务程序：

```
Unmask (NAK/XACTERR/NYET/STALL/XFERC)
if (XFERC)
{
    Reset Error Count
    Mask ACK
    De-allocate Channel
}
else if (STALL)
{
    Transfer Done = 1
    Unmask CHHLD
    Disable Channel
}
else if (NAK or XACTERR or NYET)
{
    Rewind Buffer Pointers
    Unmask CHHLD
    Disable Channel
    if (XactErr)
    {
        Increment Error Count
        Unmask ACK
    }
    else
}
```

```

    {
        Reset Error Count
    }
}

else if (CHHLTD)
{
    Mask CHHLTD
    if (Transfer Done or (Error_count == 3))
    {
        De-allocate Channel
    }
    else
    {
        Re-initialize Channel (Do ping protocol for HS)
    }
}
else if (ACK)
{
    Reset Error Count
    Mask ACK
}

```

注意事项：

- 应用程序必须在发送 FIFO 和请求队列里有剩余空间时才能向发送 FIFO 写入数据包。应用程序需要通过非周期性发送 FIFO 空寄存器 (OTGFS\_GINTSTS.NPTXFEMP) 中断来检查发送 FIFO 是否有可用空间。
- 应用程序必须在请求队列有可用空间才能写入请求直到收到 XFERC 中断。

### 20.5.3.9 初始化中断IN传输

图 20-8 显示了典型的中断 IN 传输操作流程。详见通道 2(ch\_2)。假设：

- 应用程序正在尝试从奇数帧开始接收最大一个数据包长度的数据包（传输长度为 64 字节）
- 接收 FIFO 可以保存至少一个最大数据包长度的包以及每个数据包的两个状态 DWORDs（对于全速传输有 1031 字节）
- 周期性请求队列深度为 4。

#### (1) 普通的中断 IN 操作流程

图 20-8 (通道 2) 所描述的操作流程如下：

1. 初始化通道 2 (按照“OTGFS 通道初始化”步骤)。应用程序必须设置奇数帧寄存器 (OTGFS\_HCCHAR2.ODDFRM) 位。
2. 设置通道使能寄存器 (OTGFS\_HCCHAR2.CHENA) 位，向周期性请求队列写入 IN 请求。
3. 每次置起 OTGFS 主机通道 2 特性寄存器(OTGFS\_HCCHAR2)的 CHENA 位时，OTGFS 主机都会向周期性请求队列写入一个 IN 请求。
4. OTGFS 主机尝试在下一个 (奇数) 帧时发送一个 IN 令牌。
5. 一旦收到 IN 数据包，并写入接收 FIFO 后，OTGFS 主机就会生成 RXFLVL 中断。
6. 为了处理 RXFLVL 中断，需要先读取接收的数据包状态以确认接收的字节数，然后再读取接收 FIFO。应用程序必须在读取接收 FIFO 之前先屏蔽 RXFLVL 中断，并在读完整个数据包之后解除中断屏蔽。
7. 传输完成状态被写入接收 FIFO 时控制器会产生 RXFLVL 中断。应用程序必须读取接收包状态并在检测到接收包状态不是 IN 数据包时忽略这个包。
8. 一旦读出接收数据包状态后，控制器即生成 XFERC 中断。
9. 为了处理 XFERC 中断，需要先读取包数目寄存器 (OTGFS\_HCTSIZ2.PKTCNT 位)。如果 OTGFS\_HCTSIZ2.PKTCNT 不为 0，则需要中止该通道，然后重新初始化该通道进行下次传输。如果 OTGFS\_HCTSIZ2.PKTCNT == 0，则需要重新初始化该通道进行下一个传输。此时，应用程序必须复位奇数帧寄存器(OTGFS\_HCCHAR2.ODDFRM) 位。

#### (2) 处理中断

下列代码描述了中断 IN 传输流程中与通道相关的中断服务程序。

```

Unmask (NAK/XACTERR/XFERC/BBLERR/STALL/FRMOVRUN/DATATGLERR)
if (XFERC)
{
    Reset Error Count
}

```

```
Mask ACK
if (HCTSIZx.PKTCNT == 0)
{
    De-allocate Channel
}
else
{
    Transfer Done = 1
    Unmask CHHLTD
    Disable Channel
}
}
else if (STALL or FRMOVRUN or NAK or DATATGLERR or BBLERR)
{
    Mask ACK
    Unmask CHHLTD
    Disable Channel
    if (STALL or BBLERR)
    {
        Reset Error Count
        Transfer Done = 1
    }
    else if (!FRMOVRUN)
    {
        Reset Error Count
    }
}
else if (XACTERR)
{
    Increment Error Count
    Unmask ACK
    Unmask CHHLTD
    Disable Channel
}
else if (CHHLTD)
{
    Mask CHHLTD
    if (Transfer Done or (Error_count == 3))
    {
        De-allocate Channel
    }
    else Re-initialize Channel (in next b_interval - 1 uF/F)
}
else if (ACK)
{
    Reset Error Count
    Mask ACK
}
```

应用程序必须在请求队列的剩余空间达到 MC 位域指定数目时才能向同一个通道写入请求，然后再切换到其它通道（如果有的话）。

### 20.5.3.10 初始化中断OUT传输

图 20-8 显示了典型的中断 OUT 操作流程。详见通道 1(ch\_1)。假设：

- 应用程序正在尝试从奇数帧开始每个帧发送一个最大包长度的数据包（传输长度为 64 字节）
- 周期性发送 FIFO 可保存 1 个包（对于全速模式为 1KB）
- 周期性请求队列长度为 4。

#### (1) 普通的中断 OUT 操作

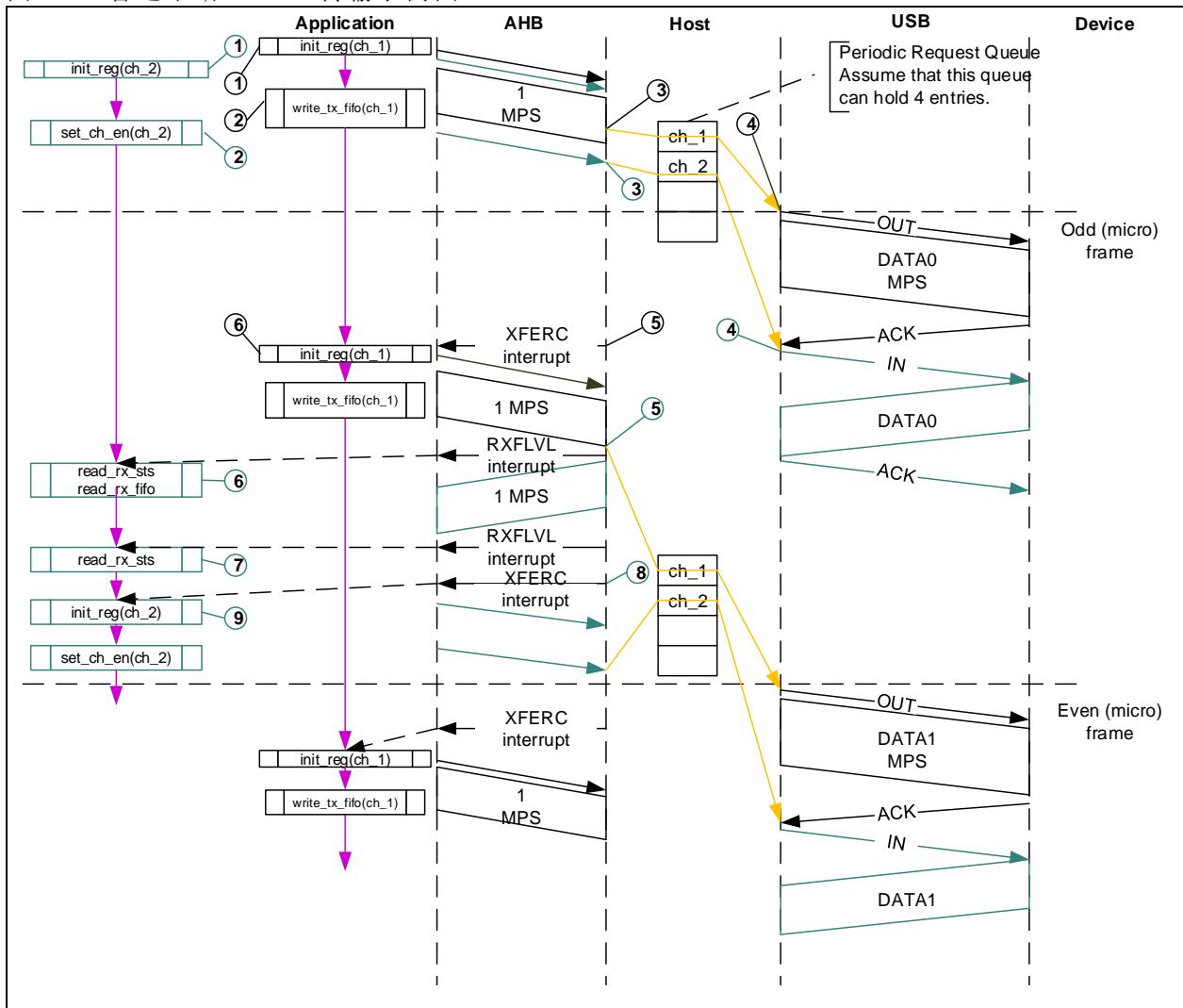
图 20-8 (通道 1) 所描述的操作流程如下：

1. 参考“OTGFS 通道初始化”使能并初始化通道 1。应用程序必须设置奇数帧寄存器 (OTGFS\_HCCHAR1.ODDFRM) 位。
2. 向通道 1 写入第一个数据包。
3. 在写完每个包的最后一个 DWORD 时，主机向周期性请求队列写入一个请求

4. 主机会在下一个帧时(奇数帧)时发送 OUT 令牌
5. 在最后一个包发送成功后，主机会生成 XFERC 中断
6. 为响应 XFERC 中断，需要重新初始化该通道进行下次传输。

## (2) 处理中断

图 20-8 普通中断 OUT/IN 传输示例图



下列代码示例为中断 OUT 传输流程中与通道相关的中断服务程序

```

Unmask (NAK/XACTERR/STALL/XFERC/FRMOVRUN)
if (XFERC)
{
    Reset Error Count
    Mask ACK
    De-allocate Channel
}
else if (STALL or FRMOVRUN)
{
    Mask ACK
    Unmask CHHLTD
    Disable Channel
    if (STALL)
    {
        Transfer Done = 1
    }
}
else if (NAK or XACTERR)
{
    Rewind Buffer Pointers
    Reset Error Count
}

```

```

Mask ACK
Unmask CHHLTD
Disable Channel
}
else if (CHHLTD)
{
Mask CHHLTD
if (Transfer Done or (Error_count == 3))
{
De-allocate Channel
}
else
{
Re-initialize Channel (in next b_interval - 1 uF/F)
}
}
else if (ACK)
{
Reset Error Count
Mask ACK
}

```

在切换到其它通道之前，应用程序需要根据 MC 位域设定的数目，在发送 FIFO 有可用空间时向发送 FIFO 和请求队列写入数据包。应用程序通过非周期性发送 FIFO 空寄存器 (OTGFS\_GINTSTS.NPTXFEMP) 中断来确认发送 FIFO 是否有可用空间。

### 20.5.3.11 初始化同步IN传输

图 20-9 显示了典型的同步 IN 传输流程。详见通道 2(ch\_2)。假设：

- 应用程序正在尝试从下一个奇数帧开始每个帧发送一个最大数据包长度的数据包（传输长度为 1023 字节）
- 接收 FIFO 可保存至少一个最长包长度的数据包和两个状态 DWORDs（对于全速传输有 1031 字节）
- 周期性请求队列深度为 4。

#### (1) 普通的同步 IN 传输

图 20-9 (通道 2) 所描述的操作流程如下：

1. 参考“OTGFS 通道初始化”初始化通道 2。应用程序必须设置奇数帧寄存器 (OTGFS\_HCCHAR2.ODDFRM) 位。
2. 设置通道使能寄存器 (OTGFS\_HCCHAR2.CHENA) 位向周期性请求队列写入 IN 请求。
3. 每次设置 OTGFS 主机通道 2 特性寄存器(OTGFS\_HCCHAR2)的 CHENA 位，主机就会向周期性请求队列写入一个 IN 请求
4. 主机会在下一个奇数帧时发送 IN 令牌
5. 当接收到 IN 数据包并写入接收 FIFO 后，主机会生成 RXFLVL 中断。
6. 为响应 RXFLVL 中断，需要读取接收的数据包状态以确认接收的字节数，然后读取接收 FIFO。应用程序在读取接收 FIFO 之前必须先屏蔽 RXFLVL 中断，并在读取了整个数据包之后解除中断屏蔽。
7. 控制器在传输完成状态输入到接收 FIFO 之后会生成 RXFLVL 中断。此时，应用程序必须读取接收数据包状态，并在检测到接收数据包状态不是 IN 数据包时丢弃这个包。(GRXSTR.PKTSTS!= 0x0010)。
8. 控制器在读取了接收数据包状态后会生成 XFERC 中断。
9. 为响应 XFERC 中断，需要读取包数目寄存器 (OTGFS\_HCTSIZ2.PKTCNT) 位。如果 OTGFS\_HCTSIZ2.PKTCNT 不为 0，然后再重新初始化该通道前中止该通道，然后进行下次传输（如果有的话）。如果 OTGFS\_HCTSIZ2.PKTCNT == 0，需重新初始化该通道以进行下一次传输。此时，应用程序必须读取奇数帧寄存器(OTGFS\_HCCHAR2.ODDFRM) 位。

#### (2) 处理中断

下列代码示例同步 IN 传输流程中与通道相关的中断服务程序。

```

Unmask (XACTERR/XFERC/FRMOVRUN/BBLERR)
if (XFERC or FRMOVRUN)
{
if (XFERC and (HCTSIZx.PKTCNT == 0))
{
Reset Error Count
}
}

```

```
        De-allocate Channel
    }
else
{
    Unmask CHHLTD
    Disable Channel
}
}
else if (XACTERR or BBLERR)
{
    Increment Error Count
    Unmask CHHLTD
    Disable Channel
}
else if (CHHLTD)
{
    Mask CHHLTD
    if (Transfer Done or (Error_count == 3))
    {
        De-allocate Channel
    }
    else
    {
        Re-initialize Channel
    }
}
```

### 20.5.3.12 初始化同步OUT传输

图 20-9 显示了典型的中步 OUT 传输操作流程。详见通道 1(ch\_1)。假设：

- 应用程序正在尝试从下一个奇数帧开始每个帧发送一个最大数据包长度的数据包（传输长度为 1023 字节）
- 周期性发送 FIFO 可保存 1 个数据包（对于全速模式为 1KB）
- 周期性请求队列深度为 4。

#### (1) 普通的同步 OUT 传输操作流程

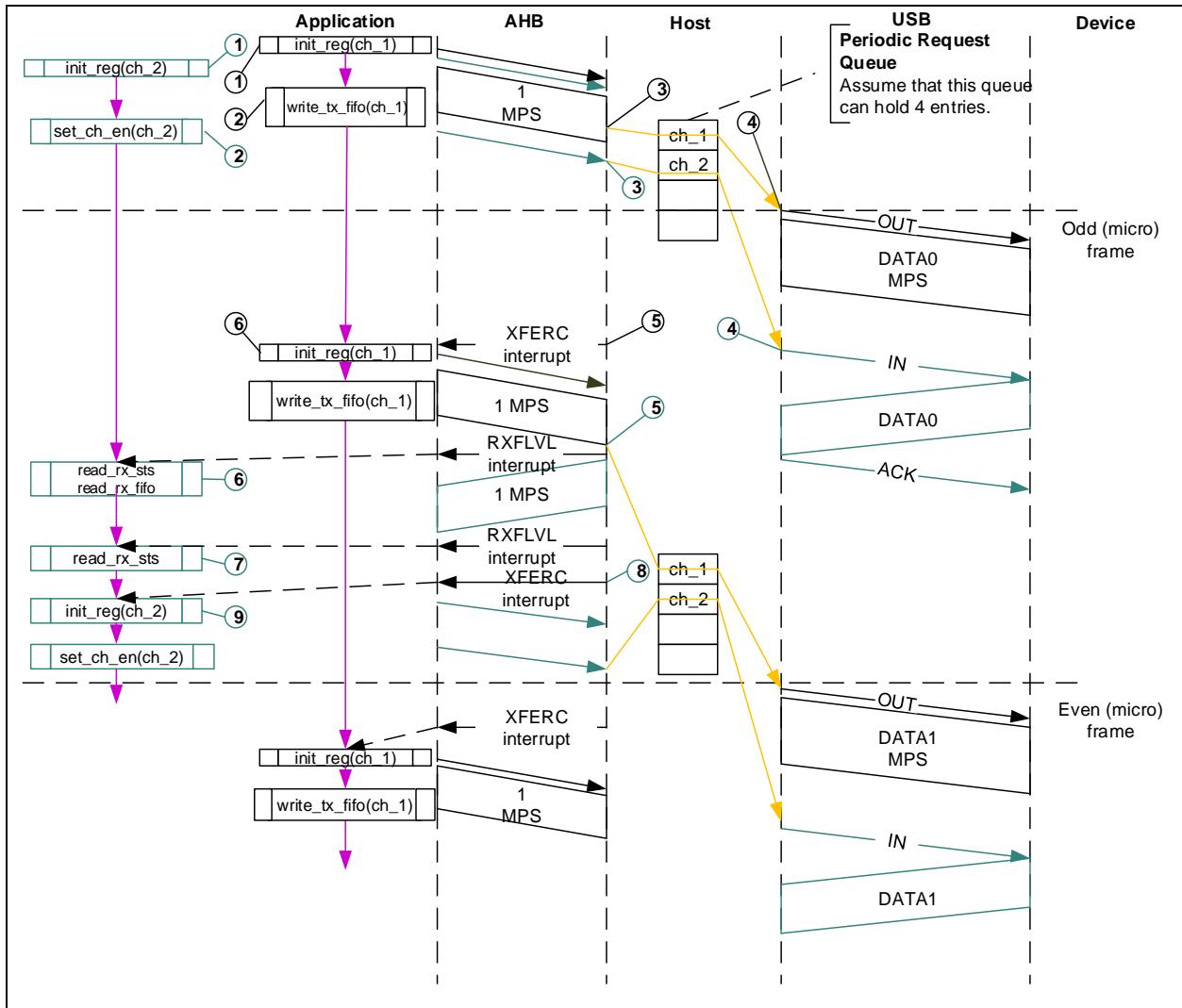
图 20-9 (通道 2) 所描述的操作流程如下：

1. 参考“OTGFS 通道初始化”初始化通道 1。应用程序必须设置奇数帧寄存器(OTGFS\_HCCHAR1.ODDFRM)位。
2. 向通道 1 写入第一个数据包。
3. 在写完每个包的最后一个 DWORD 时，主机向周期性请求队列写入一个请求
4. OTGFS 尝试在下一个帧（奇数帧）发送 OUT 令牌
5. 一旦发送完成最后一个数据包，OTGFS 主机即会生成 XFERC 中断。
6. 为响应 XFERC 中断，需要重新初始化该通道进行下一个传输。

#### (2) 处理中断

图 20-9 显示了同步 OUT 传输操作过程中与通道相关的中断服务程序。

图 20-9 普通同步 OUT/IN 传输示例图



下列示例代码为同步 OUT 传输过程中与通道相关的中断服务程序。

```
Unmask (FRMOVRUN/XFERC)
if (XFERC)
{
    De-allocate Channel
}
else if (FRMOVRUN)
{
    Unmask CHHLTD
    Disable Channel
}
else if (CHHLTD)
{
    Mask CHHLTD
    De-allocate Channel
}
```

## 20.5.4 OTGFS设备模式

### 20.5.4.1 设备初始化

在设备开启时，上电时或者从主机模式切换到设备模式时，应用程序需要遵循下列步骤初始化控制器。

- 配置 OTGFS 设备配置寄存器(OTGFS\_DCFG)的相应位

- 设备速度
- 非零长度状态 OUT 握手

- 周期性帧间隔
- 2. 清除 OTGFS 设备控制寄存器(OTGFS\_DCTL.SFTDISCON)位。控制器在清除此位后会启动连接。
- 3. 设置 OTGFS 中断屏蔽寄存器(OTGFS\_GINTMSK)以解除下列中断屏蔽：
  - USB 复位
  - 枚举完成
  - 早期挂起
  - USB 挂起
  - SOF
- 4. 等待 OTGFS 中断寄存器(OTGFS\_GINTSTS.USBRESET)中断，该中断表示检测到 USB 总线上的复位信号已持续 10ms。应用程序一旦接收此中断，就必须按照“USB 复位时初始化”的步骤操作。
- 5. 等待 OTGFS 中断寄存器(OTGFS\_GINTSTS.ENUMDONE)中断。该中断指示 USB 复位结束。应用程序在接收到此中断后，需读取 OTGFS 设备状态寄存器(OTGFS\_DSTS)来确认枚举速度，并按照“枚举完成时初始化”的步骤来操作。此时，设备准备接受 SOF 包，并在控制端点 0 执行控制传输。

#### 20.5.4.2 USB复位时初始化

本节介绍了当检测到 USB 复位信号时应用程序必须执行的操作。

1. 置起所有 OUT 端点的 NAK 位
- OTGFS\_DOEPCTLx.SNAK = 0x1(针对所有 OUT 端点)
2. 解除下列位的中断屏蔽
  - OTGFS\_DAINTMSK.INEP0 = 0x1(控制 IN 端点 0)
  - OTGFS\_DAINTMSK.OUTEP0 = 0x1(控制 OUT 端点 0)
  - OTGFS\_DOEPMSK.SETUP = 0x1
  - OTGFS\_DOEPMSK.XFERC = 0x1
  - OTGFS\_DIEPMSK.XFERC = 0x1
  - OTGFS\_DIEPMSK.TIMEOUT = 0x1
3. 为了收发数据，设备必须遵守“设备初始化”流程对寄存器进行初始化
4. 为每个端点分配 SRAM
  - 设置 OTGFS 接收 FIFO 长度寄存器(OTGFS\_GRXFSIZ)以确保能接收控制 OUT 数据和 SETUP 数据。所分配的 SRAM 至少是控制端点 0 的 1 个最大包长度+2 个 DWORDs (用于控制 OUT 数据包的状态信息)+10 个 DWORDs (用于 setup 包)。
  - 设置端点 0 TX FIFO 长度寄存器(OTGFS\_DIEPTXF0)，以确保能够发送控制 IN 数据。所分配的 SRAM 大小至少是控制端点 0 的 1 个最大包长度。
5. 复位“设备配置寄存器”的“设备地址”位。
6. 设置与端点控制寄存器中的下列位域，确保控制 OUT 端点 0 能够接收 SETUP 数据包。
  - OTGFS\_DOEPTSIZ0.SUPCNT = 0x3(可接收高达 3 个连续的 SETUP 数据包)此时，用于接收 SETUP 数据包的所有初始化操作就完成了。

#### 20.5.4.3 枚举完成时初始化

本节介绍了当检测到枚举完成中断信号时应用程序必须执行的操作。

- 检测到枚举完成中断信号时，读取 OTGFS 设备状态寄存器(OTGFS\_DSTS)来获取枚举速度。
- 配置 OTGFS 设备控制 IN 端点 0 控制寄存器 (OTGFS\_DIEPCTL0.MPS)位来设置最大包长度。这个操作是用于配置控制端点 0. 控制端点的最大包长度是由枚举速度决定的。
- 解除 SOF 中断屏蔽。

此时，设备准备接收 SOF 包并已经设置好，准备在控制端点 0 执行控制传输。

#### 20.5.4.4 SetAddress指令时初始化

本节介绍了当 SETUP 包收到了 SetAddress 指令时应用程序必须执行的操作。

- 使用 SetAddress 指令收到的设备地址来配置 OTGFS 设备配置寄存器 (OTGFS\_DCFG)。
- 设置控制器，发送 IN 包。

#### 20.5.4.5 SetConfiguration/SetInterface时初始化

本节介绍了在收到 SetConfiguration / SetInterface 命令时应用程序必须执行的操作。

- 在收到 SetConfiguration 命令时，应用程序需根据新配置定义的有效端点的特性来设置端点寄存器。
- 在收到 SetInterface 命令时，应用程序需针对受到该命令影响的端点配置端点寄存器。
- 有些端点在之前的配置中是有效的，但是在新的配置中可能失效。必须停用这些无效端点。
- 关于如何激活或停用某个端点的，详见“激活端点”以及“USB 端点失效操作”
- 解除每个有效端点的中断屏蔽，并屏蔽 DAINTMSK 寄存器中的所有无效端点的中断
- 为每个 FIFO 配置 SRAM。详见“OTGFS FIFO 配置”。
- 在配置完所有端点后，应用程序需要设置控制器来发送状态 IN 包。  
此时，设备控制器已准备好接收和发送任何类型的数据包。

#### 20.5.4.6 激活端点

本节介绍了如何激活一个设备端点或者将现有的设备端点配置为新端点类型。

1. 配置 OTGFS 设备端点 x 控制寄存器(OTGFS\_DIEPCTLx)(对于 IN 或双向端点)或 OTGFS 设备 OUT 端点 x 控制寄存器(OTGFS\_DOEPCTLx) (对于 OUT 或双向端点) 的以下位：

- 最大包长度
- USB 有效端点位 = 0x1
- 端点起始数据翻转（指中断和批量类型的端点）
- 端点类型
- 发送 FIFO 号

2. 一旦激活端点，控制器就开始解析发送到该端点的令牌，并针对该端点收到的每一个有效令牌发出一个握手有效信号

#### 20.5.4.7 USB 端点失效操作

本节介绍的是如何使一个现存的端点失效。必须先终止挂起的传输，才能进行端点失效操作。

- 清除 OTGFS 设备端点 x 控制寄存器(OTGFS\_DIEPCTLx) (对于 IN 或双向端点) 或者 OTGFS 设备 OUT 端点 x 控制寄存器(OTGFS\_DOEPCTLx) (对于 OUT 或双向端点) 的“USB 有效端点”位。
- 一旦端点失效，控制器就会忽略发送到该端点的令牌，这将导致 USB 超时。

#### 20.5.4.8 控制写传输(SETUP/Data OUT/Status IN)

本节介绍了控制写传输的操作流程。

应用程序配置流程为：

1. OTGFS 设备端点 x 中断寄存器(OTGFS\_DOEPINTx.SETUP)包中断置起表示已经向应用程序发送了一个有效 SETUP 包，且数据阶段已启动，详见“OUT 数据传输”。在 SETUP 阶段结束时，应用程序需重新向 OTGFS 设备端点 x 传输长度寄存器(OTGFS\_DOEPTSIZx.SUPCNT)位写 3 来接收下一个 SETUP 包。
2. 如果在 SETUP 中断生成之前收到的最后一个 SETUP 包指示数据 OUT 阶段，需按照“非同步 OUT 数据传输”配置控制器来执行控制 OUT 传输。
3. 对于控制端点 0 的单次 OUT 数据传输，应用程序最多可接收 64 字节数据。如果应用程序期望在数据 OUT 阶段接收不止 64 字节的数据，那么必须重新使能该端点另外再接收 64 字节数据，而且需要持续此类操作直到接收完数据阶段的所有数据。
4. 在最后一个 OUT 传输时置起 OTGFS 设备端点 x 中断寄存器(OTGFS\_DOEPINTx.XFERC)中断表示该控制传输的数据 OUT 阶段结束。
5. 一旦完成数据 OUT 阶段，应用程序必须执行如下操作：
  - 如果需要传输一个新的 SETUP 包，应用程序必须重新使能控制 OUT 端点（参考“OUT 数据传输”）  
 $OTGFS\_DOEPCTLx.EPENA = 0x1$
  - 为了执行接收到的 SETUP 命令，应用程序必须配置控制器中的相应寄存器。这个是可选操作，视所收到的 SETUP 命令类型而定。
6. 在状态 IN 阶段，应用程序必须按照“非周期性（批量和控制）IN 数据传输”来配置寄存器以执行数据 IN 传输。

7. OTGFS 设备端点 x 中断寄存器(OTGFS\_DOEPINTx.XFERC)中断表示控制传输的状态阶段已启动。当接收 FIFO 包状态寄存器的“数据传输完成模式”和“状态阶段开始”位置起时，控制器即会生成该中断。通过设置 OTGFS 设备端点 x 中断寄存器(OTGFS\_DOEPINTx.XFERC)来清除“传输完成”中断。重复上述步骤直至检测到该端点上产生 OTGFS 设备端点 x 中断寄存器(OTGFS\_DIEPINTx.XFERC)中断才表示该控制写传输已完成。

#### 20.5.4.9 控制读传输(SETUP/Data IN/Status OUT)

本节介绍的是控制读传输。应用程序操作流程：

- OTGFS 设备端点 x 中断寄存器(OTGFS\_DOEPINTx.SETUP)包中断表示已向应用程序发送了一个有效的 SETUP 包，而且数据阶段已启动。详见“OUT 数据传输”。应用程序在 SETUP 阶段结束时，必须重新向 OTGFS 设备 OUT 端点 0 传输长度寄存器(OTGFS\_DOEPTSIZ0.SUPCNT)位写 3 来接收下一个 SETUP 包。
- 如果在 SETUP 中断生成之前收到的最后一个 SETUP 包指示数据 IN 阶段，需按照“非周期性 IN 数据传输”配置控制器来执行控制 IN 传输。
- 对于控制端点 0 的单次 IN 数据传输，应用程序最多可接收 64 字节数据。如果应用程序期望在数据 IN 阶段发送不止 64 字节的数据，那么必须重新使能该端点另外再发送 64 字节数据，而且需要持续此类操作直到发送完数据阶段的所有数据。
- 重复上述步骤直至检测到该端点上每个 IN 传输都生成了 OTGFS 设备端点 x 中断寄存器(OTGFS\_DIEPINTx.XFERC)中断。
- 在最后一个 IN 传输时置起 OTGFS 设备端点 x 中断寄存器(OTGFS\_DIEPINTx.XFERC)中断表示该控制传输的数据 OUT 阶段结束
- 如需在状态 OUT 阶段执行数据 OUT 传输，应用程序需要按照“OUT 数据传输”来配置控制器。应用程序必须先正确设置 OTGFS 设备配置寄存器(OTGFS\_DCFG.NZSTSOUTSHSK)握手位，再发送状态阶段的数据 OUT 传输。
- OTGFS 设备端点 x 中断寄存器(OTGFS\_DOEPINTx.XFERC)中断表示该控制传输的状态 OUT 阶段结束，标志着控制读传输成功完成。

#### 20.5.4.10 两个阶段的控制传输(SETUP/Status IN)

本节介绍了两个阶段的控制传输操作。应用程序操作流程：

1. OTGFS 设备端点 x 中断寄存器(OTGFS\_DOEPINTx.SETUP)包中断表示已向应用程序发送了一个有效的 SETUP 包，而且数据阶段已启动。详见“OUT 数据传输”。应用程序在 SETUP 阶段结束时，必须重新向 OTGFS 设备端点 x 传输长度寄存器(OTGFS\_DOEPTSIZx.SUPCNT)位写 3 来接收下一个 SETUP 包。
2. 在 SETUP 中断生成之前对所接收的最后一个 SETUP 包进行解析。如果 SETUP 包指示的两级控制命令，那么应用程序必须执行以下操作：
  - 设置 OTGFS\_DOEPCTLx.EPENA = 0x1
  - 根据所收到的 SETUP 命令类型，应用程序需要配置控制器中的寄存器来执行所收到的 SETUP 命令。
3. 对于状态 IN 阶段，应用程序需按照“非周期性（批量和控制）IN 数据传输”配置寄存器来执行数据 IN 传输。
4. OTGFS 设备端点 x 中断寄存器(OTGFS\_DIEPINTx.XFERC)中断表示该控制传输的状态 IN 阶段已完成。

#### 20.5.4.11 读取FIFO包

本节介绍了如何读取接收 FIFO 数据包（OUT 数据和 SETUP 包）

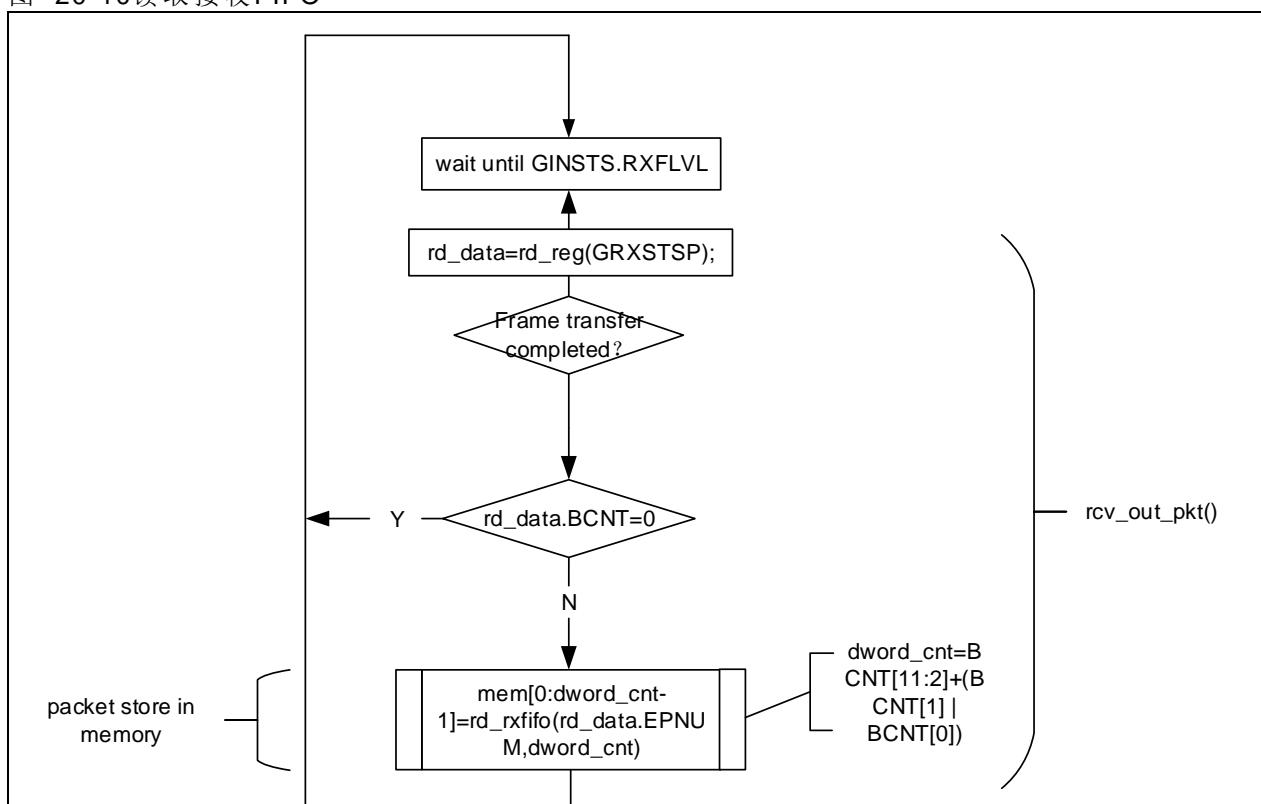
1. 一旦检测到 OTGFS 中断寄存器(OTGFS\_GINTSTS.RXFLVL)中断，应用程序必须读取 OTG 状态读和 POP 寄存器(OTGFS\_GRXSTSP)。
2. 应用程序可以通过设置 OTGFS\_GINTMSK.RXFLVL = 0x0 来屏蔽 OTGFS 中断寄存器(OTGFS\_GINTSTS.RXFLVL)中断，直到从接收 FIFO 读取到数据包。
3. 如果收到的数据包字节数不为 0，那么接收数据 FIFO 会弹出数据字节数并将其储存在存储器。如果所接收的数据包字节数为 0，那么接收数据 FIFO 则不会有数据读出。
4. 接收 FIFO 包状态读数指示发生下列情况。
5. 全局 OUT NAK 模式：PKTSTS = Global OUT NAK, BCNT = 0x000, EPNUM = Dont Care (0x0),

DPID = Dont Care (0x00), 指示全局 OUT NAK 位已生效。

- SETUP 包模式: PKTSTS = SETUP, BCnt = 0x008, EPNUM = Control EP Num, DPID = D0, 表示可以从接收 FIFO 读取某个指定端点的 SETUP 包。
- Setup 阶段完成模式: PKTSTS = Setup Stage Done, BCNT = 0x0, EPNUM = Control EP Num, DPID = Don't Care (0x00), 表示某个指定端点的 Setup 阶段已结束，并开始进入数据阶段。当接收 FIFO 弹出此请求后，控制器会在指定的控制 OUT 端点上触发 Setup 中断。
- 数据 OUT 包模式: PKTSTS = DataOUT, BCnt = 接收的数据 OUT 包的长度( $0 \leq BCNT \leq 1024$ ), EPNUM = 接收数据包的端点号, DPID = 实际的数据 PID。
- 数据传输完成模式: PKTSTS = 数据 OUT 传输完成, BCNT = 0x0, EPNUM = 完成数据传输的 OUT 端点号, DPID = Don't Care (0x00)。这些数据表示某个指定的 OUT 端点的 OUT 数据传输已完成。当接收 FIFO 弹出此请求后，控制器会在指定的 OUT 端点上触发传输完成中断。PKTSTS 代码位于本手册的寄存器章节的“接收状态调试读取/状态读和弹出寄存器”

6. 接收 FIFO 弹出有效数据后，必须解除 OTGFS 中断寄存器(OTGFS\_GINTSTS.RXFLVL)中断屏蔽。
7. 每当应用程序检测到由于 OTGFS 中断寄存器(OTGFS\_GINTSTS.RXFLVL)而产生中断线时需要重复第 1-5 步。读取空接收 FIFO 会导致控制器出现意外后果。流程如下图:

图 20-10 读取接收 FIFO



#### 20.5.4.12 OUT 数据传输

本节介绍了数据 OUT 传输和 SETUP 传输过程中的内部数据流及应用程序的操作流程。

##### (1) 执行 Setup 传输

本节介绍了如何处理 SETUP 数据包以及应用程序处理 SETUP 传输的操作流程。上电复位后，应用程序必须遵循“OTGFS 初始化”流程来初始化控制器。应用程序在与主机通信之前，必须按照“设备初始化”流程来初始化端点，并参考“读取 FIFO 包”

##### 【应用程序要求】

1. 控制 OUT 端点的 OTGFS 设备端点 x 传输长度寄存器(OTGFS\_DOEPTSIZx.SUPCNT)位必须设置为非零值才能接收 SETUP 包。当应用程序将 SUPCNT 位设置为非零值时，控制器会接收到 SETUP 包并将其写入接收 FIFO，不受 OTGFS 设备 OUT 端点 x 控制寄存器(OTGFS\_DOEPCTLx.NAK)状态位和 OTGFS 设备 OUT 端点 x 控制寄存器(OTGFS\_DOEPCTLx.EPENA)位的影响。SUPCNT 位在每次控制

端点接收到一个 SETUP 包时会自动递减。如果在接收 SETUP 包之前, SUPCNT 位的设置值不合适, 那么虽然控制器仍可以接收 SETUP 包并自动递减 SUPCNT 位, 但是应用程序可能就无法确定在控制传输 SETUP 阶段所接收的 SETUP 包有多少个。

- OTGFS\_DOEPTSI<sub>Zx</sub>.SUPCNT = 0x3
- 2. 应用程序必须为接收数据 FIFO 分配一些额外的空间, 以确保能够在一个控制端点接收多达 3 个 SETUP 包。
  - 预留空间为 13DWORDs, 其中 4 个 DWORDs 空间用于 1 个 SETUP 包, 1 个 DWORD 空间用于 Setup 阶段, 8 个 DWORDs 空间用于存入控制端点的两个额外的 SETUP 包。
  - 每个 SETUP 包需要 4 个 DWORDs 空间用于存放 8 个字节的 SETUP 数据, 4 个字节的传输完成状态以及 4 个字节的 SETUP 状态 (SETUP 包模式)。控制器需要为接收数据预留此空间。
  - FIFO 仅用于写 SETUP 数据, 而不会用于数据包
- 3. 应用程序需要从接收 FIFO 读取 2 个 DWORDs 的 SETUP 包。
- 4. 应用程序必须从接收 FIFO 读取传输完成状态 DWORD 并丢弃它。并忽略由于读取而产生的传输完成中断。

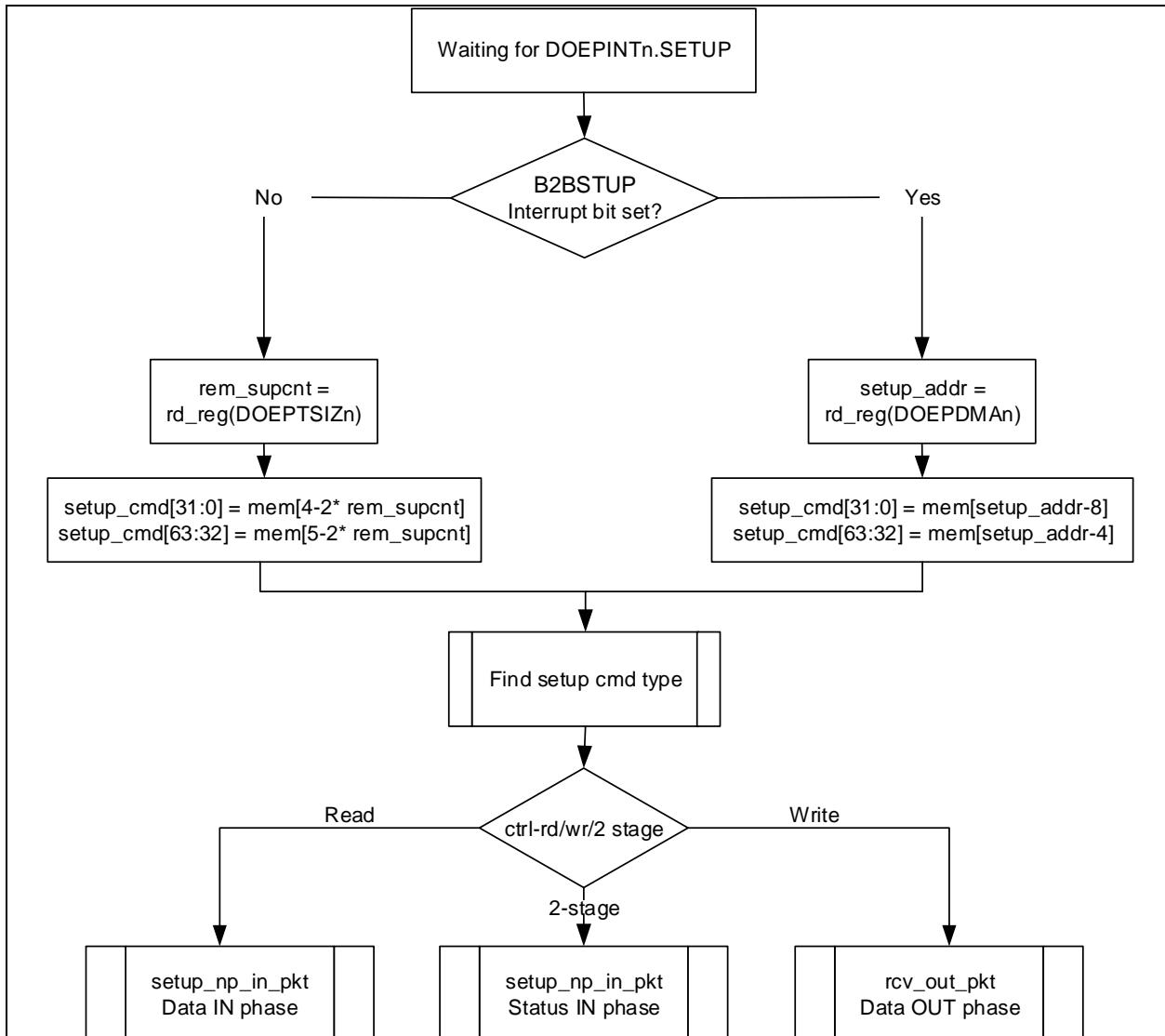
#### 【内部数据流】

- 1.当收到 SETUP 包时, 控制器将接收的数据写入接收 FIFO, 不需要确认接收 FIFO 是否有可用空间, 也不需要检测控制端点的 NAK 和 Stall 位。
  - 控制器会在收到 SETUP 包的控制 IN/OUT 端点上置起 IN NAK 和 OUT NAK 位。
2. USB 线上收到每个 SETUP 包后, 都会写入 3 个 DWORDs 数据到接收 FIFO, SUPCNT 位自动递减 1.
  - 首个 DWORD 包含控制器内部使用的控制信息。
  - 第二个 DWORD 包含 SETUP 命令的前 4 个字节。
  - 第三个 DWORD 包含 SETUP 命令的最后 4 个字节。
3. 当从 SETUP 阶段切换到数据 IN/OUT 阶段时, 控制器会写入 SETUP 状态完成 DWORD 信息到接收 FIFO, 指示 SETUP 阶段已结束。
4. 应用程序通过 AHB 总线读取 SETUP 包。
5. 当应用程序弹出了来自接收 FIFO 的状态阶段完成 DWORD 信息时, 控制器会产生 OTGFS 设备端点 x 中断寄存器(OTGFS\_DOEPINT<sub>x</sub>.SETUP)中断来打断应用程序, 指示应用程序可以开始处理接收的 SETUP 包。
6. 控制器清除控制 OUT 端点的端点使能位。

#### 【应用程序操作流程】

1. 配置 OTGFS 设备端点 x 传输长度寄存器(OTGFS\_DOEPTSI<sub>Zx</sub>)。
  - OTGFS\_DOEPTSI<sub>Zx</sub>.SUPCNT = 0x3
2. 等待 OTGFS 中断寄存器(OTGFS\_GINTSTS.RXFLVL)中断并读取刷新接收 FIFO 的数据包 (参考“读取 FIFO 包”)。可以多次重复此操作。
3. 宣告 OTGFS 设备端点 x 中断寄存器(OTGFS\_DOEPINT<sub>x</sub>.SETUP)中断表示 SETUP 数据传输已完成。在收到此中断后, 应用程序需要读取 OTGFS 设备端点 x 传输长度寄存器(OTGFS\_DOEPTSI<sub>Zx</sub>)来确认收到了多少个 SETUP 包, 并处理最后接收到的一个 SETUP 包。

图 20-11 SETUP数据包流程图



## (2) 处理 3 个以上的连续的 SETUP 包

根据 USB2.0 规范，通常在 SETUP 包出现错误时，主机不会向同一个端点连续发送 3 个以上的 SETUP 包。然而，USB2.0 规范并没有限制主机可以向同一个端点发送连续 SETUP 包的数目。如果发生此类情况，OTGFS 控制器会生成一个中断(OTGFS 设备端点 x 中断寄存器(OTGFS\_DOEPINTx.B2BSTUP))。

### 20.5.4.13 IN数据传输

本节介绍了 IN 数据传输期间的内部数据流和应用程序的操作流程。

1. 应用程序可以选择轮询模式或中断模式。

- 如果选择轮询模式，应用程序将通过读取 OTGFS 设备 IN 端点传输 FIFO 状态寄存器(OTGFS\_DTXFSTSx)来监测端点发送数据 FIFO 的状态以确认数据 FIFO 内是否有足够的空间可用。
  - 如果选择中断模式，应用程序需要等待 OTGFS 设备端点 x 中断寄存器(OTGFS\_DIEPINTx.TXFEMP)中断，然后读取 OTGFS 设备 IN 端点传输 FIFO 状态寄存器(OTGFS\_DTXFSTSx)来确认数据 FIFO 内是否有足够的空间可用。
  - 如果要写一个单独的非零长度的数据包，数据 FIFO 必须要有足够的空间写入整个数据包。
  - 如果要写一个零长度的数据包，应用程序不需要查看 FIFO 空间。
2. 无论使用上述哪种方式，当应用程序确定了有足够的空间可以写入一个发送数据包时，可以先写入端点控制寄存器，再将数据写入数据 FIFO。通常，除了设置端点使能位之外，应用程序必须在 OTGFS 设备端点 x 控制寄存器(OTGFS\_DIEPCTLx)进行读改写设置以防止该寄存器的内容被更改，如果空间足够

大的话，应用程序可以将同一个端点的多个数据包写入发送 FIFO。对于周期性 IN 端点，应用程序只能写入一个帧的数据包。只有在前一个传输发送完成的情况下，才能写入下一个周期性传输。

#### 20.5.4.14 非周期性（批量和控制）IN数据传输

如果要在上电复位后初始化控制器，那么应用程序必须遵循“OTGFS 初始化”的流程。在与主机通讯前，必须先按照“设备初始化”流程对端点进行初始化操作。

##### 【应用程序要求】

1. 对于 IN 传输而言，端点传输长度寄存器中的传输长度位表示的有效数据包含了多个最大包长度的数据包和一个短包。短包在传输结束时被传输。

- 如果需要传输几个最大包长度的数据包和一个短包：

    传输长度[epnum] =  $n * \text{mps}[epnum] + sp$ , (其中  $n$  是  $\geq 0$  的整数，且  $0 \leq sp < \text{mps}[epnum]$ )

    如果( $sp > 0$ ), 那么包数目[epnum] =  $n + 1$ 。否则，包数目[epnum] =  $n$

- 如果需要传输一个单独的零长度的数据包：

    传输长度[epnum] = 0x0

    包数目[epnum] = 0x1

- 如果要传输几个最大包长度的数据包和一个零长度的数据包（在结束时传输），那么应用程序需要分成两部分进行传输。先发送最大包长度的数据包，再单独发送零长度的数据包。

    首次传输：传输长度[epnum] =  $n * \text{mps}[epnum]$ ; 包数目 =  $n$ ;

    二次传输：传输长度[epnum] = 0x0; 包数目 = 0x1;

2. 如果使能了端点进行数据传输，则控制器会更新传输长度寄存器。在 IN 传输结束时，以端点禁用中断为结束标志，此时应用程序需要读取传输长度寄存器来确认 USB 线上发送了多少 FIFO 数据。

3. 发送 FIFO 获取的数据 = 应用程序设置的首个传输长度 - 控制器更新的最终传输长度

- USB 已发送的数据 = (应用程序设置的首个包数目 - 控制器更新的最终包数目) \*  $\text{mps}[epnum]$

- USB 待发送的数据 = 应用程序设置的首个传输长度 - USB 已发送的数据

##### 【内部数据流】

1. 应用程序需要设置端点控制寄存器中的传输长度和包数目位，并使能端点来传输数据。

2. 应用程序还需要将所需数据写入该端点的传输 FIFO。

3. 每当应用程序向发送 FIFO 写入一个数据包时，相应端点的传输长度会随着包长度而自动递减。需要持续写入数据直到该端点的传输长度为 0。将数据写入 FIFO 后，“FIFO 中的包数目”会递增（每个 IN 端点发送 FIFO 数据包是 3bit 数目，由内部控制器保存。对于一个 IN 端点 FIFO，任何时候，控制器能保存的最大包数目为 8）。对于非零长度的数据包，每一个 FIFO 会设置一个单独的标志，FIFO 中不会有数据。

4. 当数据写入发送 FIFO 后，控制器在收到 IN 令牌时会读取该数据。对于每个以 ACK 握手信号结束的非同步 IN 数据包传输，该端点的包数目会自动减 1，直到包数目为 0，包数目不会因为超时而递减。

5. 对于零长度数据包（内部会置起一个零长度标志），控制器会根据 IN 令牌发送零长度数据包，并且包数目位会自动递减。

6. 如果收到了 IN 令牌，但 FIFO 没有数据，且该端点的包数目为 0，那么控制器会生成“当 FIFO 为空时收到了 IN 令牌”的中断，同时不设置该端点的 NAK 位。控制器会以 NAK 握手信号来回应 USB 线上的非同步端点

7. 控制器内部会绕回 FIFO 指针，除了控制 IN 端点外，不会产生超时中断，

8. 当传输长度为 0 并且包数目也为 0 时，会产生传输完成中断，并清除端点使能位。

##### 【应用程序操作流程】

1. 根据传输长度和相应的包数目来设置 OTGFS 设备端点 x 传输长度寄存器(OTGFS\_DIEPTSIz)。

2. 根据端点特性来设置 OTGFS 设备端点 x 控制寄存器(OTGFS\_DIEPCTLx)，并设置 CNAK 和端点使能位。

3. 如果发送非零长度的数据包，则应用程序必须轮询 OTGFS 设备 IN 端点传输 FIFO 状态寄存器(OTGFS\_DTXFSTSx)（其中 n 是指与该端点相关的 FIFO 编号）来确认数据 FIFO 是否有足够的可用空间。应用程序也可以使用 OTGFS 设备端点 x 中断寄存器(OTGFS\_DIEPINTx.TXFEMP)来确定是否写数据。

#### 20.5.4.15 非同步OUT数据传输

如果要在上电复位后初始化控制器，那么应用程序必须遵循“OTGFS 初始化”的流程。在与主机通讯前，必须先按照“端点初始化”流程对端点进行初始化操作，并参考“读取 FIFO 包”。本章节介绍了常规非同步 OUT 传输操作（控制传输，批量传输或中断传输）

### 【应用程序要求】

- 对于 OUT 传输，端点传输长度寄存器的传输长度位必须是该端点最大包长度的倍数，并调整到 DWORD 边界。

```
if (mps[epnum] mod 4) == 0  
transfer size[epnum] = n * (mps[epnum]) //Dword Aligned  
else  
transfer size[epnum] = n * (mps[epnum] + 4 - (mps[epnum] mod 4)) //Non Dword  
Aligned  
packet count[epnum] = n  
n > 0
```

- 发生 OUT 端点中断时，应用程序需要读取端点的传输长度寄存器来计算内存中的数据量。所收到的有效数据长度必须小于设定的传输长度。

- 内存中的有效负载=应用程序设置的首个传输长度-控制器更新的最终传输长度
- 收到此有效负载的 USB 数据包数目=应用程序设置的首个包数目—控制器更新的最终包数目

### 【内部数据流】

- 应用程序需要设置端点控制寄存器的传输长度和包数目位，清除 NAK 位，并使能接收数据的端点
- 一旦清除 NAK 位，只要接收 FIFO 里有可用空间，控制器就开始接收数据并写入接收 FIFO。对于 USB 线收到的每个数据包，需要将数据包和其状态写入接收 FIFO。每次向接收 FIFO 中写入数据包（最大包长度或短包），包数目位会自减 1。
  - 接收 FIFO 将自动清空所收到的含有 Bad Data CRC 的 OUT 数据包。
  - 在 USB 发出数据包 ACK 信号后，控制器不会理会主机（因为未检测到 ACK）重新发送的非同步 OUT 数据包。应用程序没有检测到具有相同数据 PID 的同一个端点上的多个连续 OUT 数据包，此时，包数目不会自动递减。
  - 如果接收 FIFO 里没有空间可用，同步或非同步数据包会被忽略，也不会写入接收 FIFO。另外，非同步 OUT 令牌还会收到一个 NAK 握手信号回应。
  - 以上 3 种情况下，包数目不会递减，因为没有数据写入接收 FIFO。
- 当包数目变为 0 或者当端点收到短包时，该端点会置起 NAK 位。一旦设置 NAK 位，同步或非同步数据包会被忽略，也不会写入接收 FIFO，而且非同步 OUT 令牌还会收到 NAK 握手信号回应。
- 将数据写入接收 FIFO 后，应用程序会读取接收 FIFO 中的数据并写入外部存储器，每个端点 1 次 1 个包。
- 在完成了数据包写入外部存储器之后，该端点的传输长度会随着写入包的长度减少而减少。
- 在发生下列情况时，OUT 端点的 OUT 数据传输完成模式会写入接收 FIFO。
  - 传输长度和包数目均为 0。
  - 写入接收 FIFO 的最后一个 OUT 数据包是一个短包 ( $0 \leq$  数据包长度  $<$  最大包长度)
- 当应用程序弹出该条目（即 OUT 数据传输完成），即会产生“传输完成中断”，且清除该端点使能位。

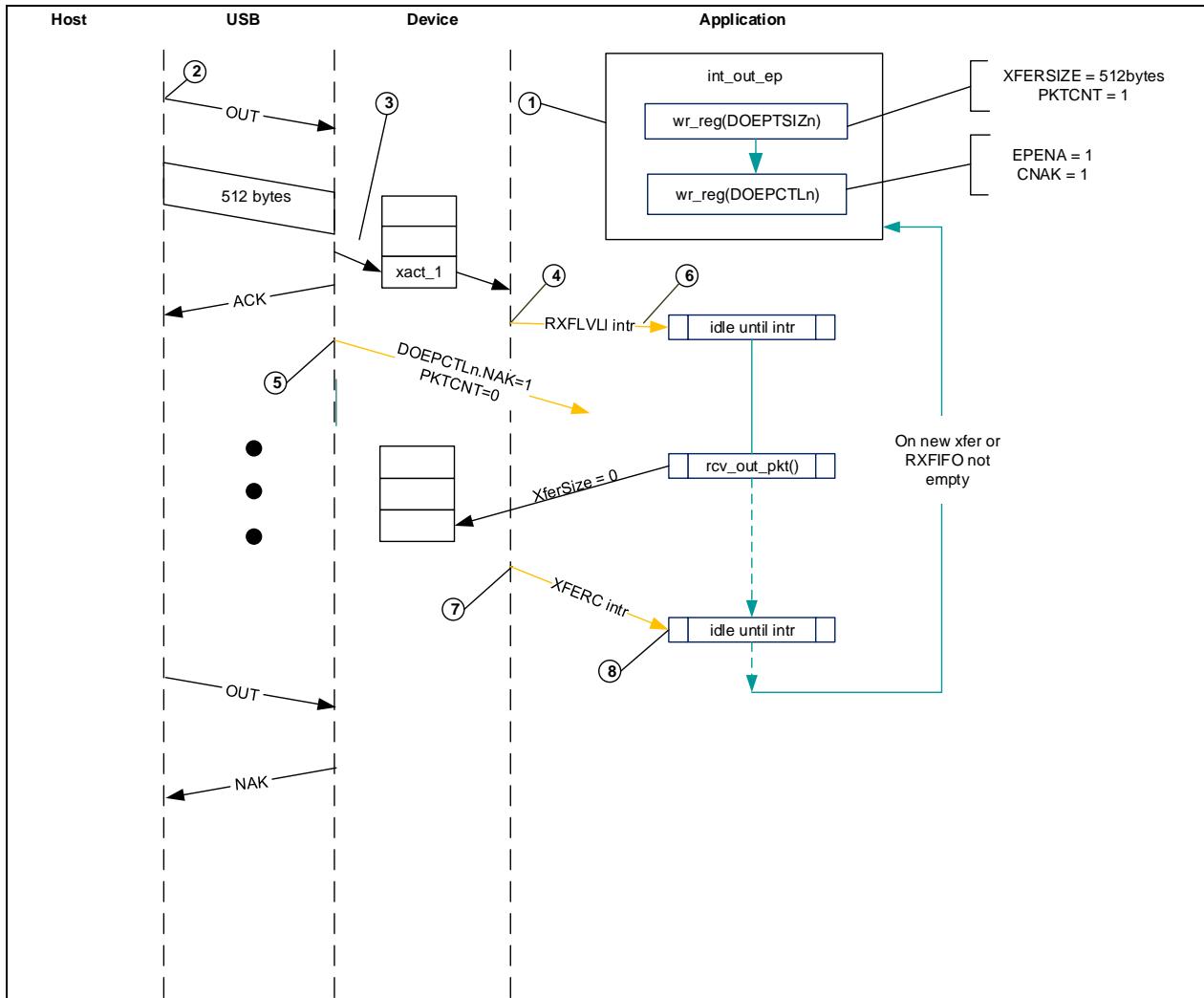
### 【Application Programming Sequence】应用程序操作流程

- 根据传输长度和相应的包数目来设置 OTGFS 设备端点 x 传输长度寄存器(OTGFS\_DOEPTSIZx)。
- 根据端点特性设置 OTGFS 设备 OUT 端点 x 控制寄存器(OTGFS\_DOEPCTLx)，并设置端点使能位和 ClearNAK 位。
  - OTGFS\_DOEPCTLx.EPENA = 0x1
  - OTGFS\_DOEPCTLx.CNAK = 0x1
- 等待 OTGFS 中断寄存器(OTGFS\_GINTSTS.RXFLVL)中断，按照“读取 FIFO 包”的流程来读取接收 FIFO 中的所有数据包。
  - 根据传输长度可重复此步骤
- 置起 OTGFS 设备端点 x 中断寄存器(OTGFS\_DOEPINTx.XFERC)中断即表示成功完成了非同步 OUT 数据传输。读取 OTGFS 设备端点 x 传输长度寄存器(OTGFS\_DOEPTSIZx)来确认收到了多少数据。

### 【批量 OUT 传输】

下图描绘了从 USB 到 AHB 收到一个单独的批量 OUT 数据包，以及该过程所涉及的相关事件。

图 20-12 BULK OUT 传输示例图



在收到 SetConfiguration/SetInterface 命令后，应用程序将通过设置 OYG\_DOEPCTLx.CNAK = 0x1 以及 OTGFS\_DOEPCTLx.EPENA = 0x1 来初始化所有的 OUT 端点，并设置 OTGFS 设备端点 x 传输长度寄存器(OTGFS\_DOEPTSIZx)中的 XFERSIZE 和 PKTCNT 位。

1. 主机尝试给端点发送数据（OUT 令牌）
2. 当控制器收到 USB 发现的 OUT 令牌时，会将数据存放在接收 FIFO，因为接收 FIFO 里有可用空间。
3. 将完整的数据写入接收 FIFO 之后，控制器会解发 OTGFS 中断寄存器(OTGFS\_GINTSTS.RXFLVL) 中断。
4. 在收到 USB 包的包数目后，控制器内部会通过设置该端点的 NAK 位以防止收到更多的数据包。
5. 应用程序处理该中断并从接收 FIFO 读取该数据。
6. 当应用程序读取完所有数据（相当于 XFERSIZE）之后，控制器会生成 OTGFS 设备端点 x 中断寄存器(OTGFS\_DOEPINTx.XFERC)中断。
7. 应用程序处理该中断，并使用 OTGFS 设备端点 x 中断寄存器(OTGFS\_DOEPINTx.XFERC)中位后来判断已完成预期的传输。

#### 20.5.4.16 同步OUT数据传输

如果需要在上电复位后初始化控制器，应用程序需遵循“OTGFS 初始化”的流程操作。在与主机通讯之前，需要按照“端点初始化”流程对端点进行初始化操作，并参考“读取 FIFO 包”。

本节介绍的是常规同步 OUT 数据传输。

##### 【应用要求】

1. 与非同步 OUT 数据传输的应用程序要求相同。
2. 对于同步 OUT 数据传输，传输长度和包数目位必须设置为一个帧所能接收的最大包长度，不能超过这个数。同步 OUT 数据传输不能跨越超过 1 个帧。

- $1 \leq \text{包数目}[\text{epnum}] \leq 3$
- 3. 如果设备支持同步 OUT 端点，则应用程序必须在周期性帧结束之前（OTGFS 中断寄存器(OTGFS\_GINTSTS.EOPF)中断）读取接收 FIFO 中的所有同步 OUT 数据包。
- 4. 如果要在下一个帧接收数据，必须在产生 OTGFS 中断寄存器(OTGFS\_GINTSTS.EOPF)中断以及开始 OTGFS 中断寄存器(OTGFS\_GINTSTS.SOF)信号之前使能同步 OUT 端点。

#### 【内部数据流】

1. 同步 OUT 端点的内部数据流与非同步 OUT 端点是一样的，只有细微区别。
  2. 如果通过设置端点使能位和清除 NAK 位使能了同步 OUT 端点，则奇/偶帧位也要进行适当设置。只有在满足下列条件时，控制器才能在某个帧的同步 OUT 端点接收数据。
- OTGFS 设备 OUT 端点 x 控制寄存器(OTGFS\_DOEPCTLx.Even)/Odd microframe = OTGFS\_DSTS.SOFFN[0]
  - 3. 当应用程序完全读取了接收 FIFO 的同步 OUT 数据包（数据和状态）时，控制器会根据从接收 FIFO 读取的最后一个同步 OUT 数据包的数据 PID 来更新 OTGFS 设备端点 x 传输长度寄存器(OTGFS\_DOEPTSIzX.RXDPID)位。

#### 【应用程序操作流程】

1. 设置 OTGFS 设备端点 x 传输长度寄存器(OTGFS\_DOEPTSIzX)的传输长度和相应包数目。
2. 根据端点特性设置 OTGFS 设备 OUT 端点 x 控制寄存器(OTGFS\_DOEPCTLx)，并设置端点使能位，ClearNAK 和偶/奇帧位
  - 端点使能 = 0x1
  - CNAK = 0x1
  - 偶/奇帧 = (0x0: 偶; 0x1: 奇)
3. 等待 OTGFS 中断寄存器(OTGFS\_GINTSTS.RXFLVL)中断，并读取接收 FIFO 中的所有数据包，详见“读取 FIFO 包”
- 可以根据传输长度重复此动作。
4. OTGFS 设备端点 x 中断寄存器(OTGFS\_DIEPINTx.XFERC)中断表示已完成同步 OUT 数据传输。但是此中断并不一定意味着存储器的数据都是好的。
5. 同步 OUT 传输并不一定能检测到该中断信号，但是应用程序可以检测到 OTGFS 中断寄存器(OTGFS\_GINTSTS.INCOMPISOOUT)同步 OUT 数据中断，详见“不完整的同步 OUT 数据传输”。
6. 读取 OTGFS 设备端点 x 传输长度寄存器(OTGFS\_DOEPTSIzX)来确认所收到的传输长度，并判断帧内所收到的数据是否有效。只有在满足了下列条件时，应用程序才会将存储器收到的数据视为有效数据。
  - OTGFS\_DOEPTSIzX.RXDPID = 0xD0 且收到有效数据的 USB 包数目 = 0x1
  - OTGFS\_DOEPTSIzX.RXDPID = 0xD1 且收到有效数据的 USB 包数目 = 0x2
  - OTGFS\_DOEPTSIzX.RXDPID = 0xD2 且收到有效数据的 USB 包数目 = 0x3

收到有效数据的 USB 包数 = APP 设置的初始包数目 - 控制器更新的最终包数目  
应用程序不理睬无效数据包。

### 20.5.4.17 使能同步端点

主机将设置接口控制命令发送给设备后，会使能同步端点。随后，主机就可以发送任意帧内的首个同步 IN 令牌，然后再按照 BlInterval 的顺序发送。

但是，在 OTGFS 控制器中，同步支持是基于单个传输级。应用程序必须根据每个帧重新配置控制器。OTGFS 控制会使能将要发生传输的帧之前的那个帧的同步端点。

例如，如果数据要在帧 n 上发送，那么必须使能帧 n-1 的端点。另外，OTGFS 通过设置偶/奇帧位来调度同步传输。

#### 【同步 IN 传输中断】

需要处理好下列中断以确保成功调度同步传输。

- OTGFS 设备端点 x 中断寄存器(OTGFS\_DIEPINTx.XFERC) (基于端点)
- OTGFS 中断寄存器(OTGFS\_GINTSTS.INCOMPISOIN) (全局中断)

#### 【处理同步 IN 传输】

需要按照下列步骤来处理同步 IN 传输：

1. 通过设置 OTGFS 中断屏蔽寄存器(OTGFS\_GINTMSK.INCOMISOINMSK)解除 OTGFS 中断寄存器(OTGFS\_GINTSTS.incompISOOUT)中断
2. 通过设置 OTGFS 设备 OTGFSIN 端点通用中断屏蔽寄存器(OTGFS\_DIEPMSK.XFERCMSK)解除 OTGFS 设备端点 x 中断寄存器(OTGFS\_DIEPINTx.XFERC)中断

3. 通过执行下列操作来使能同步端点:

- 配置 OTGFS 设备端点 x 传输长度寄存器(OTGFS\_DIEPTSI<sub>Zx</sub>)

OTGFS\_DIEPTSI<sub>Zx</sub>.XFERSIZE= n \* OTGFS\_DIEPCTL<sub>x</sub>.MPS + sp。其中 0 <= n <= 3, 0 <= sp < OTGFS\_DIEPCTL<sub>x</sub>.MPS。当帧内传输的数据长度小于 OTGFS 设备端点 x 控制寄存器(OTGFS\_DIEPCTL<sub>x</sub>.MPS), n=0。当帧内传输的数据长度是 OTGFS 设备端点 x 控制寄存器(OTGFS\_DIEPCTL<sub>x</sub>.MPS)的倍数时, sp=0。

OTGFS\_DIEPTSI<sub>Zx</sub>.PKTCNT = 1。

OTGFS 设备端点 x 传输长度寄存器(OTGFS\_DIEPTSI<sub>Zx</sub>.MC)的设置值与 OTGFS 设备端点 x 传输长度寄存器(OTGFS\_DIEPTSI<sub>Zx</sub>.PKTCNT)一样。

- 配置 OTGFS 设备端点 x 控制寄存器(OTGFS\_DIEPCTL<sub>x</sub>)

读取 OTGFS 设备状态寄存器(OTGFS\_DSTS)来确定当前的帧号

配置 OTGFS 设备端点 x 控制寄存器(OTGFS\_DIEPCTL<sub>x</sub>.MPS)为最大包长度

配置 OTGFS 设备端点 x 控制寄存器(OTGFS\_DIEPCTL<sub>x</sub>.USBACTEP)为 0x1

配置 OTGFS 设备端点 x 控制寄存器(OTGFS\_DIEPCTL<sub>x</sub>.EPTYPE)为 0x1, 表示同步

配置 OTGFS 设备端点 x 控制寄存器(OTGFS\_DIEPCTL<sub>x</sub>.TXFNUM)为端点的 FIFO 号

配置 OTGFS 设备端点 x 控制寄存器(OTGFS\_DIEPCTL<sub>x</sub>.CNAK)为 0x1

如果 OTGFS\_DSTS.SOFFN[0] = 0x0, 则 OTGFS\_DIEPCTL<sub>x</sub>.SETEVENFR = 0x1 (否则 OTGFS\_DIEPCTL<sub>x</sub>.SETEVENFR = 0x1)

如果 OTGFS\_DSTS.SOFFN[0] = 0x1, 则 OTGFS\_DIEPCTL<sub>x</sub>.SETODDFR = 0x1(否则 OTGFS\_DIEPCTL<sub>x</sub>.SETODDFR = 0x0)

配置 OTGFS\_DIEPCTL<sub>x</sub>.EPENA = 0x1

#### 4. 将端点数据写入相应的发送 FIFO

例如, 写入地址范围

- EP1 对应 0x2000 - 0x2FFC
- EP2 对应 0x3000 - 0x3FFC
- EP3 对应 0x3000 - 0x3FFC
- ...

#### 5. 等待中断

● 当 OTGFS 设备端点 x 中断寄存器(OTGFS\_DIEPINT<sub>x</sub>.XFERC)中断产生时, 清除 OTGFS 设备端点 x 中断寄存器(OTGFS\_DIEPINT<sub>x</sub>.XFERC); 对于下一个传输任务, 重复步骤 3-5, 直到完成传输。

● 当 OTGFS 中断寄存器(OTGFS\_GINTSTS.INCOMPISOIN)中断产生时, 清除 OTGFS 中断寄存器(OTGFS\_GINTSTS.INCOMPISOIN); 对于任何一个同步 IN 端点, 当奇/偶位与当前帧号位 0 一致并且在端点保持使能的情况下, 控制器在帧结束时会生成该中断。下列情况会导致该中断:

- (1) 帧内没有令牌
- (2) 数据写入接收 FIFO 较晚, 数据还没写完, IN 令牌就到了
- (3) IN 令牌出错。

OTGFS 中断寄存器(OTGFS\_GINTSTS.INCOMPISOIN)是一个全局中断。所以, 当不止一个同步端点处于激活状态时, 应用程序必须判断哪一个同步 IN 端点没有完成数据传输。

为了实现这一步, 需要读取所有同步端点的 DSTS 和 DIEPCTL<sub>x</sub> 位。如果当前端点已使能, 并且 OTGFS 设备状态寄存器(OTGFS\_DSTS.SOFFN)的读取返回值是该端点的目标帧号, 那么该端点就未完成传输。应用程序必须准确地跟踪并更新该同步端点的目标帧号。

如果某个端点未完成传输, 那么需翻转奇/偶位。

接着:

(1) 当 OTGFS 设备端点 x 控制寄存器(OTGFS\_DIEPCTL<sub>x</sub>.DPID)位为 1 (奇帧) 时, 向 OTGFS 设备端点 x 控制寄存器(OTGFS\_DIEPCTL<sub>x</sub>.SETD0PID)写 1 使其成为一个偶帧, 然后, 当下一个帧有 IN 令牌输入时, 就会发送数据。

(2) 当 OTGFS 设备端点 x 控制寄存器(OTGFS\_DIEPCTL<sub>x</sub>.DPID)为 0 (偶帧) 时, 向 OTGFS 设备端点 x 控制寄存器(OTGFS\_DIEPCTL<sub>x</sub>.SETD1PID)写 1 使其变成奇帧, 这样, 当下一个帧有 IN 令牌输入时, 就会发送数据。

### 20.5.4.18 未完成同步 OUT 数据传输

若需要在上电复位时初始化控制器，那么应用程序需要根据“OTGFS 初始化”流程进行操作。在与主机通讯之前，必须先按照“端点初始化”的流程初始化端点。本节介绍了当控制器丢失了同步 OUT 数据包时，应用程序的设置流程。

#### 【内部数据流】

1. 对于同步 OUT 端点来说，并不意味着始终会生成 OTGFS 设备端点 x 中断寄存器(OTGFS\_DOEPINTx.XFERC)中断。如果控制器丢失了同步 OUT 数据包，那么应用程序在遇到下列情况时可能无法检测到 OTGFS 设备端点 x 中断寄存器(OTGFS\_DOEPINTx.XFERC)中断。
  - 当接收 FIFO 无法容纳完整的 ISO OUT 数据包时，控制器会丢失接收到的 ISO OUT 数据。
  - 当接收的同步 OUT 数据包含有 CRC 错误时。
  - 当控制器接收到的同步 OUT 令牌被损坏时。
  - 当应用程序读取接收 FIFO 数据的速度非常慢的时候。
2. 当控制器在传输完成之前检测到所有同步 OUT 端点的周期帧结束时，会生成未完成同步 OUT 数据中断，表示至少有一个同步 OUT 端点没有生成 OTGFS 设备端点 x 中断寄存器(OTGFS\_DOEPINTx.XFERC)中断。此时，虽然未完成数据传输的这个端点保持使能状态，但是该端点并未进行有效传输。

#### 【应用程序操作流程】

1. 宣告未完成同步 OUT 数据中断表示在当前帧内，至少有一个同步 OUT 端点没有完成数据传输。
2. 如果这是因为该端点的同步 OUT 数据没有完全读取导致的，那么应用程序必须读取接收 FIFO 中的所有同步 OUT 数据（包括数据和状态），再进行下一步处理。
  - 在读取完接收 FIFO 中的所有数据后，应用程序会检测到 OTGFS 设备端点 x 中断寄存器(OTGFS\_DOEPINTx.XFERC)中断。此时，应用程序需要按照“控制读传输(SETUP/Data IN/Status OUT)”的流程重新使能该端点来接收下一个帧的同步 OUT 数据。
3. 在收到到未完成同步 OUT 数据中断时，应用程序需要读取所有同步 OUT 端点的控制寄存器来确认当前帧内哪一个端点没有完成传输。如果下列两个条件都满足时，则表示端点就没有完成传输。
  - OTGFS\_DOEPCTLx.偶/奇帧位 = OTGFS\_DSTS.SOFFN[0]
  - OTGFS\_DOEPCTLx.端点使能 = 0x1
4. 必须在检测到 GINTSTS.SOF 中断之前执行上一个步骤，以确保当前帧号不被更改。
5. 对于没有完成传输的同步 OUT 端点，应用程序必须丢弃存储器中的数据，并通过 OTGFS\_DOEPCTLx.端点禁用位来禁用此端点。
6. 等待 OTGFS\_DOEPINTx.端点禁用中断，并按照“控制读传输(SETUP/Data IN/Status OUT)”的流程使能该端点接收下一个帧的新数据。由于控制器需要花一些时间禁用此端点，所以应用程序在接收到错误数据之后可能无法接收到下一个帧的数据。

### 20.5.4.19 未完成同步IN数据传输

本节介绍了当同步 IN 数据传输未完成时应用程序该如何操作。

#### 【内部数据流】

1. 在下列情况下，同步 IN 传输被视为未完成。
  - 控制器在不止一个同步 IN 端点接收到已损坏的同步 IN 令牌。此时应用程序会检测到 GINTSTS.未完成同步 IN 传输中断。
  - 应用程序向发送 FIFO 写入完整数据的速度较慢，且还未完成写入就接收到 IN 令牌。此时，应用程序会检测到 OTGFS 设备端点 x 中断寄存器(OTGFS\_DIEPINTx.INTKNTXFEMP)中断信号。应用程序会忽略此中断，因为这将导致在帧结束时生成 OTGFS\_GINTSTS.未完成同步 IN 传输中断。控制器向 USB 发线一个零长度的数据包来回应接收到的 IN 令牌。
2. 无论处于以上哪种情况，应用程序必须尽快停止向发送 FIFO 写数据。
3. 应用程序必须设置该端点的 NAK 位和禁用位。
4. 控制器禁用此端点，清除禁用位，并触发端点禁用中断。

#### 【应用程序操作流程】

1. 当发送 FIFO 为空时，应用程序会忽略任何一个同步 IN 端点上的 OTGFS 设备端点 x 中断寄存器(OTGFS\_DIEPINTx.INTKNTXFEMP)中断信号，因为这会触发未完成同步 IN 传输中断。
2. OTGFS\_GINTSTS.未完成同步 IN 传输中断表示至少一个同步 IN 端点没有完成同步 IN 传输。
3. 应用程序必须读取所有同步 IN 端点的端点控制寄存器来确认哪一个端点没有完成 IN 数据传输。

4. 应用程序必须向这些端点的周期发送 FIFO 写入数据。
5. 通过设置 OTGFS 设备端点 x 控制寄存器(OTGFS\_DIEPCTLx)的下列位域来禁用该端点
  - OTGFS\_DIEPCTLx.SETNAK = 0x1
  - OTGFS\_DIEPCTLx.端点使能 = 0x1
6. DIEPINTx.端点禁用中断表示控制器已禁用了此断点。
7. 此时，应用程序必须通过使能该端点用于传输下一个帧来清空相关发送 FIFO 中的数据或者覆盖 FIFO 中现有的数据。应用程序必须通过 OTGFS 复位寄存器(OTGFS\_GRSTCTL)来刷新该数据。

### 20.5.4.20 周期性IN（中断和同步）数据传输

本节介绍了典型的周期性 IN 数据传输操作。

如果需要在上电复位后初始化控制器，应用程序必须遵循“OTG 初始化”的流程进行操作。在与主机通讯之前，必须按照“端点初始化”流程来初始化端点。

#### 【应用程序要求】

1. “非周期性（批量和控制）IN 数据传输”应用程序要求也适用于周期性 IN 数据传输，除了第 2 点要求略微不同。
  - 应用程序发送的数据仅限于最大包长度的数据包的倍数包，以及短包。只有在满足下列条件时，才能发送几个最大包长度的数据包和短包。  
传输长度[epnum] = n \* mps[epnum] + sp, (其中 n、i 是 $\geq 0$  的整数，且  $0 \leq sp < mps[epnum]$ )  
如果(sp > 0)，包数[epnum] = n + 1。否则，包数[epnum] = n, mc[epnum] = 包数 [epnum]
    - 应用程序不能在传输结束时发送零长度的数据包。但其本身是可以发送一个单独的零长度的数据包，条件是：传输长度[epnum] = 0; 包数目[epnum] = 1; mc[epnum] = 包数目 [epnum]
  - 2. 应用程序一次只能调度传输 1 帧数据。
    - (OTGFS\_DIEPTSIzX.MC - 1) \* OTGFS\_DIEPCTLx.MPS  $\leq$  OTGFS\_DIEPTSIzX.XFERSIZ  $\leq$  OTGFS\_DIEPTSIzX.MC \* OTGFS\_DIEPCTLx.MPS
    - OTGFS\_DIEPTSIzX.PKTCNT = OTGFS\_DIEPTSIzX.MC
    - 如果 OTGFS\_DIEPTSIzX.XFERSIZ < OTGFS\_DIEPTSIzX.MC \* OTGFS\_DIEPCTLx.MPS，最后一个数据包传输是一个短包。
  - 3. 对于周期性 IN 端点，必须在下一个帧传输之前预取 1 个帧数据。可以通过在调度数据传输的帧之前使能周期性 IN 端点 1 帧来完成这个操作。
  - 4. 应用程序在接收周期性 IN 令牌之前必须将帧要传输的完整数据写入发送 FIFO。即使在收到周期 IN 令牌时，发送 FIFO 里缺失了帧需要传输的 1 DWORD 数据，控制器将 FIFO 为空来处理。当发送 FIFO 为空时，USB 会针对 ISO IN 端点发送零长度数据包。并针对 INTR IN 端点发送 NAK 握手信号。

#### 【内部数据流】

1. 应用程序必须设置端点控制器的传输长度和包数目位，并使能该端点来传输该数据。
2. 应用程序也可以将所需数据写入相关的发送 FIFO。
3. 每当应用程序将发送 FIFO 写入一个包时，该端点的传输长度会减去包长度。需要持续写入数据直到该端点的传输长度变为 0。
4. 当收到某个周期性端点的 IN 令牌时，应用程序会将数据写入 FIFO（如果有的话）。如果 FIFO 里不存在该帧的完整数据，那么控制器会生成 INTKNTXFEMP 中断。
  - USB 针对同步 IN 端点发送一个零长度的数据包。
  - USB 针对中断 IN 端点发送一个 NAK 握手信号。
5. 端点的包数目在遇到下列情况时会自动减 1：
  - 对于同步端点，发送了长度或非零长度的数据包时
  - 对于中断端点，当发送了 ACK 握手信号
  - 当传输长和包数目都变为 0 时，会生成会该端点的传输完成中断，并清除端点使能位。
6. 在“周期性帧间隔”中（通过设置 OTGFS 设备配置寄存器(OTGFS\_DCFG.PERFRINT)），当控制器检测到用于调度当前帧非空的任意一个 IN 端点 FIFO 非空时，控制器会生成 OTGFS 中断寄存器(OTGFS\_GINTSTS.INCOMPISOIN)中断。

#### 【应用程序操作流程（帧传输）】

1. 设置 OTGFS 设备端点 x 传输长度寄存器(OTGFS\_DIEPTSIzX)
2. 根据端点特性设置 OTGFS 设备端点 x 控制寄存器(OTGFS\_DIEPCTLx)并设置 CNAK 和端点使能位。

3. 将下一个帧要传输的数据写入发送 FIFO。
4. 当产生 INTKNTXFEMP 中断表示应用程序尚未将所有待发送的数据写入到发送 FIFO。
5. 如果在检测到该中断时，中断端点已使能，请忽略此中断。如果未使能，请使能该端点以便在下一个 IN 令牌传输数据。如果在检测到该中断时同步端点已使能，详见“未完成同步 IN 数据传输”。
6. 当中断 IN 端点被设置为周期性端点，控制器内部可以处理中断 IN 端点发生的超时情况，并不需要应用程序的介入。因此，应用程序永远无法检测到周期性中断 IN 端点上的 OTGFS 设备端点 x 中断寄存器(OTGFS\_DIEPINTx.TIMEOUT)中断信号。
7. 产生 OTGFS 设备端点 x 中断寄存器(OTGFS\_DIEPINTx.XFERC)中断但是没有产生 INTKNTXFEMP 中断表示成功完成了同步 IN 传输。读取 OTGFS 设备端点 x 传输长度寄存器(OTGFS\_DIEPTSIZx)时，只有当传输长度为 0 且包数目均为 0 才表示 USB 线上的所有数据均已完全传输。
8. 产生 OTGFS 设备端点 x 中断寄存器(OTGFS\_DIEPINTx.XFERC)中断但是不管有没有产生 INTKNTXFEMP 中断表示成功完成了一个中断 IN 传输。读取 OTGFS 设备端点 x 传输长度寄存器(OTGFS\_DIEPTSIZx)时，只有当传输长度为 0 且包数目均为 0，才表示 USB 线上的所有数据均已完全传输。
9. 产生了 INCOMPISOIN 中断但没有产生上述所提到的中断即表示控制器没有接收到当前帧发送的至少 1 个周期性 IN 令牌。关于同步 IN 端点，详见“未完成同步 IN 数据传输”。

## 20.6 OTGFS控制和状态寄存器

应用程序通过 AHB 从接口来读写控制和状态寄存器(CSRx)，从而实现对 OTGFS 控制器的控制。这些寄存器都是 32 位访问的，并且 32 位地址对齐。

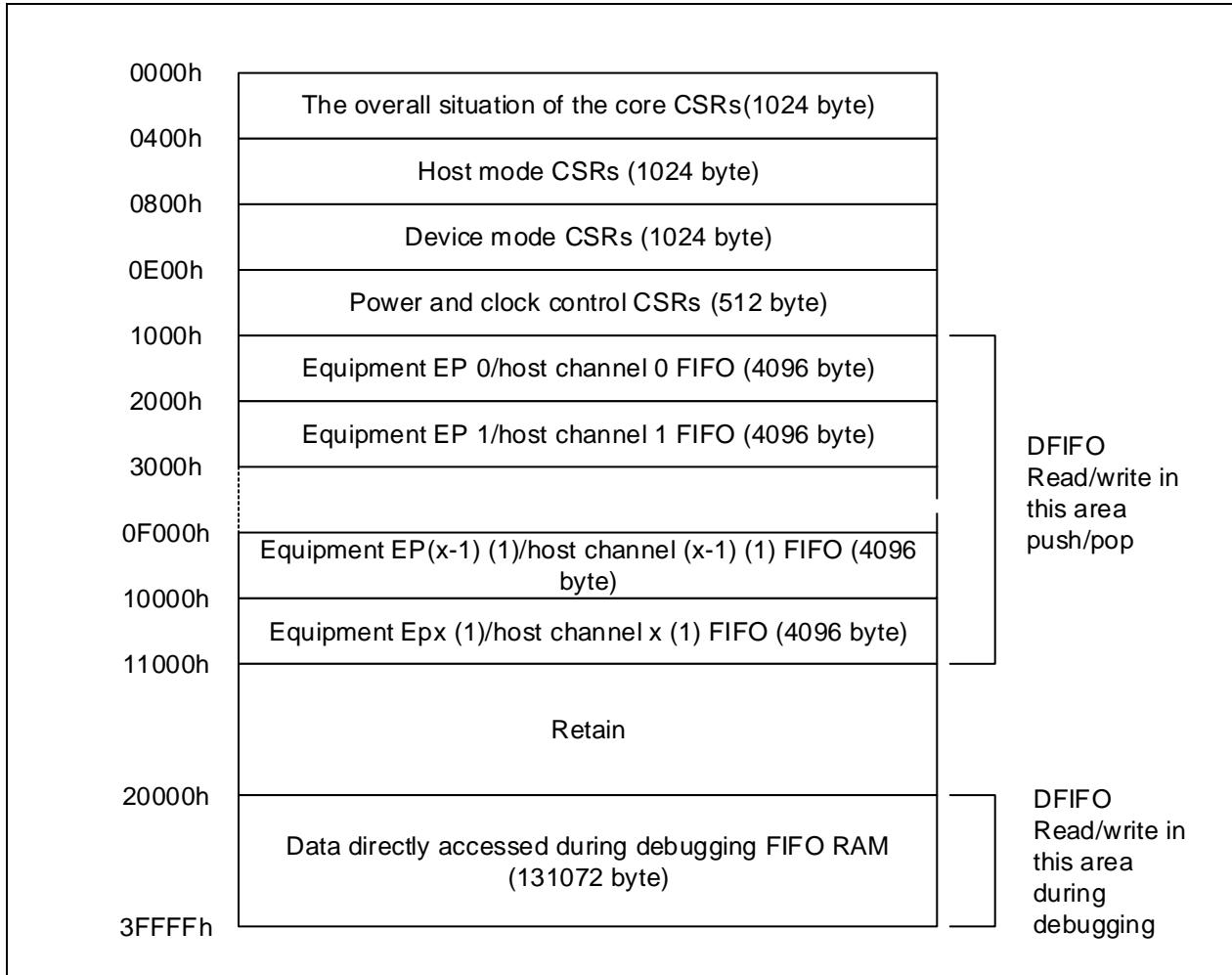
只有控制器全局寄存器，供电和时钟控制寄存器，数据 FIFO 访问寄存器以及主机端口控制和状态寄存器在主机模式和设备模式下都有效。无论 OTGFS 控制器工作在主机模式还是设备模式下，应用程序都不能访问另一种模式下的寄存器组。如果应用程序发生了非法访问，会产生模式不匹配中断并影响控制器中断寄存器的相应位(OTGFS\_GINTSTS 寄存器的 MODMIS 位)。

当控制器从一种模式切换到另一种模式时，新模式下的寄存器组都需要和上电复位时一样的重新初始化。必须以字(32 位)的方式操作这些外设寄存器。

### 20.6.1 CSR寄存器映像

主机模式寄存器和设备模式寄存器占据不同的地址。所有的寄存器都位于 AHB 时钟域。

图 20-13 CSR存储器映像



在设备模式下 x 为 4，在主机模式下 x 为 8

OTGFS 控制和状态寄存器可分为 OTGFS 全局寄存器、主机模式下寄存器、设备模式下寄存器、数据 FIFO(DFIFO)访问寄存器、供电和时钟控制寄存器。

- 1、OTGFS 全局寄存器：这些寄存器在主机模式和设备模式下都有效，寄存器首字母缩写为 G；
- 2、主机模 8 式下寄存器：这些寄存器在每次切换到主机模式时都需要配置，寄存器首字母缩写为 H；
- 3、设备模式下寄存器：这些寄存器在每次切换到设备模式时都需要配置，寄存器首字母缩写为 D；
- 4、数据 FIFO(DFIFO)访问寄存器：这组寄存器列在主机模式和设备模式下都有效，用于读写指定方向的特殊端点或通道的 FIFO。如果一个主机模式下的通道是 IN 类型的，相对应的 FIFO 只能进行读操作。同样地，如果一个主机模式下的通道是 OUT 类型的，相对应的 FIFO 只能进行写操作。
- 5、供电和时钟控制寄存器：只有一个寄存器用来控制供电和时钟控制，此寄存器在设备模式下和主机模式下都有效

## 20.6.2 OTGFS寄存器地址映象

下表给出了 USB OTG 寄存器映像和复位值。

必须以字（32 位）的方式操作这些外设寄存器。

表 20-4 OTGFS模块的寄存器映像及其复位值

寄存器简称	基址偏移量	复位值
OTGFS_GOTGCTL	0x000	0x0001 0000
OTGFS_GOTGINT	0x004	0x0000 0000
OTGFS_GAHBCFG	0x008	0x0000 0000
OTGFS_GUSBCFG	0x00C	0x0000 1400

OTGFS_GRSTCTL	0x010	0x2000 0000
OTGFS_GINTSTS	0x014	0x0400 0020
OTGFS_GINTMSK	0x018	0x0000 0000
OTGFS_GRXSTSR	0x01C	0x0000 0000
OTGFS_GRXSTSP	0x020	0x0000 0000
OTGFS_GRXFISZ	0x024	0x0000 0200
OTGFS_GNPTXFSIZ/ OTGFS_DIEPTXF0	0x028	0x0200 0200
OTGFS_GNPTXSTS	0x02C	0x0008 0200
OTGFS_GCCFG	0x038	0x0000 0000
OTGFS_GUID	0x03C	0x0000 1000
OTGFS_HPTXFSIZ	0x100	0x0000 0000
OTGFS_DIEPTXF1	0x104	0x0000 0000
OTGFS_DIEPTXF2	0x108	0x0000 0000
OTGFS_DIEPTXF3	0x10C	0x0000 0000
OTGFS_HCFG	0x400	0x0000 0000
OTGFS_HFIR	0x404	0x0000 EA60
OTGFS_HFNUM	0x408	0x0000 3FFF
OTGFS_HPTXSTS	0x410	0x0008 0100
OTGFS_HAINT	0x414	0x0000 0000
OTGFS_HAINTMSK	0x418	0x0000 0000
OTGFS_HPRT	0x440	0x0000 0000
OTGFS_HCCHAR0	0x500	0x0000 0000
OTGFS_HCINT0	0x508	0x0000 0000
OTGFS_HCINTMSK0	0x50C	0x0000 0000
OTGFS_HCTSIZ0	0x510	0x0000 0000
OTGFS_HCCHAR1	0x520	0x0000 0000
OTGFS_HCINT1	0x528	0x0000 0000
OTGFS_HCINTMSK1	0x52C	0x0000 0000
OTGFS_HCTSIZ1	0x530	0x0000 0000
OTGFS_HCCHAR2	0x540	0x0000 0000
OTGFS_HCINT2	0x548	0x0000 0000
OTGFS_HCINTMSK2	0x54C	0x0000 0000
OTGFS_HCTSIZ2	0x550	0x0000 0000
OTGFS_HCCHAR3	0x560	0x0000 0000
OTGFS_HCINT3	0x568	0x0000 0000
OTGFS_HCINTMSK3	0x56C	0x0000 0000
OTGFS_HCTSIZ3	0x570	0x0000 0000
OTGFS_HCCHAR4	0x580	0x0000 0000
OTGFS_HCINT4	0x588	0x0000 0000
OTGFS_HCINTMSK4	0x58C	0x0000 0000

OTGFS_HCTSIZ4	0x590	0x0000 0000
OTGFS_HCCHAR5	0x5A0	0x0000 0000
OTGFS_HCINT5	0x5A8	0x0000 0000
OTGFS_HCINTMSK5	0x5AC	0x0000 0000
OTGFS_HCTSIZ5	0x5B0	0x0000 0000
OTGFS_HCCHAR6	0x5C0	0x0000 0000
OTGFS_HCINT6	0x5C8	0x0000 0000
OTGFS_HCINTMSK6	0x5CC	0x0000 0000
OTGFS_HCTSIZ6	0x5D0	0x0000 0000
OTGFS_HCCHAR7	0x5E0	0x0000 0000
OTGFS_HCINT7	0x5E8	0x0000 0000
OTGFS_HCINTMSK7	0x5EC	0x0000 0000
OTGFS_HCTSIZ7	0x5F0	0x0000 0000
OTGFS_HCCHAR8	0x600	0x0000 0000
OTGFS_HCINT8	0x608	0x0000 0000
OTGFS_HCINTMSK8	0x60C	0x0000 0000
OTGFS_HCTSIZ8	0x610	0x0000 0000
OTGFS_DCFG	0x800	0x0220 0000
OTGFS_DCTL	0x804	0x0000 0002
OTGFS_DSTS	0x808	0x0000 0010
OTGFS_DIEPMSK	0x810	0x0000 0000
OTGFS_DOEPMSK	0x814	0x0000 0000
OTGFS_DAINT	0x818	0x0000 0000
OTGFS_DAINTMSK	0x81C	0x0000 0000
OTGFS_DIEPEMPMSK	0x834	0x0000 0000
OTGFS_DIEPCTL0	0x900	0x0000 0000
OTGFS_DIEPINT0	0x908	0x0000 0080
OTGFS_DIEPTSIZ0	0x910	0x0000 0000
OTGFS_DTXFSTS0	0x918	0x0000 0200
OTGFS_DIEPCTL1	0x920	0x0000 0000
OTGFS_DIEPINT1	0x928	0x0000 0080
OTGFS_DIEPTSIZ1	0x930	0x0000 0000
OTGFS_DTXFSTS1	0x938	0x0000 0200
OTGFS_DIEPCTL2	0x940	0x0000 0000
OTGFS_DIEPINT2	0x948	0x0000 0080
OTGFS_DIEPTSIZ2	0x950	0x0000 0000
OTGFS_DTXFSTS2	0x958	0x0000 0200
OTGFS_DIEPCTL3	0x960	0x0000 0000
OTGFS_DIEPINT3	0x968	0x0000 0080
OTGFS_DIEPTSIZ3	0x970	0x0000 0000
OTGFS_DTXFSTS3	0x978	0x0000 0200

OTGFS_DOEPCtl0	0xB00	0x0000 8000
OTGFS_DOEPINT0	0xB08	0x0000 0080
OTGFS_DOEPTSIZ0	0xB10	0x0000 0000
OTGFS_DOEPCtl1	0xB20	0x0000 0000
OTGFS_DOEPINT1	0xB28	0x0000 0080
OTGFS_DOEPTSIZ1	0xB30	0x0000 0000
OTGFS_DOEPCtl2	0xB40	0x0000 0000
OTGFS_DOEPINT2	0xB48	0x0000 0080
OTGFS_DOEPTSIZ2	0xB50	0x0000 0000
OTGFS_DOEPCtl3	0xB60	0x0000 0000
OTGFS_DOEPINT3	0xB68	0x0000 0080
OTGFS_DOEPTSIZ3	0xB70	0x0000 0000
OTGFS_PCGCCTL	0xE00	0x0000 0000
OTGFS_DEP3RMPEN	0xD0C	0x0000 0000
OTGFS_USBDIVRST	0xE10	0x0000 0000

### 20.6.3 OTGFS全局寄存器

该寄存器适用于主机模式和设备模式，在两种模式之间切换时不需要重新配置。

#### 20.6.3.1 OTGFS状态控制器(OTGFS\_GOTGCTL)

OTG 控制和状态寄存器用于控制 OTG 功能并反映其功能状态。

域	简称	复位值	类型	功能
位 31: 22	保留	0x0000	resd	请保持默认值。
位 21	CURMOD	0x0	ro	当前工作模式(Current Mode of Operation) 适用模式：主机模式和设备模式 该位指示当前模式。 0: 设备模式 1: 主机模式
位 20: 16	保留	0x0000	resd	请保持默认值。
位 15: 0	保留	0x0000	resd	请保持默认值。

#### 20.6.3.2 OTGFS OTG中断状态控制器(OTGFS\_GOTGINT)

应用程序可以通过读此寄存器得知发生了何种 OTG 中断，并可以通过写此寄存器清除对应的 OTG 中断。

域	简称	复位值	类型	功能
位 31: 3	保留	0x0000	resd	请保持默认值。
位 2	SESENDDET	0x0	rw1c	会话结束检测(Session End Detected) 当控制器检测到 Bvalid (此芯片即 Vbus) 信号断开时置起此位。该寄存器只能由硬件置 1，软件可通过写 1 清除此位。
位 1: 0	保留	0x0000	resd	请保持默认值。

#### 20.6.3.3 OTGFS AHB配置寄存器(OTGFS\_GAHBCFG)

此寄存器用于在上电或改变控制器模式时配置控制器。此寄存器主要配置一些和 AHB 相关的参数。在初始化配置完成后，不要再修改此寄存器。应用程序在开始向 AHB 或 USB 传送数据前必需先配置好此寄存器。

域	简称	复位值	类型	功能
位 31: 9	保留	0x000000	resd	保持默认值。
位 8	PTXFEMPLVL	0x0	rw	周期性发送 FIFO 空级别(Periodic TxFIFO Empty Level)

				表示何时会触发控制器中断寄存器(GINTSTS.PTXFEMP)的周期性发送 FIFO 空中断位。 0: GINTSTS.PTXFEMP 中断表示周期性发送 FIFO 是半空 1: GINTSTS.PTXFEMP 中断表示周期性发送 FIFO 是全空
位 7	NPTXFEMPLVL	0x0	rw	适用模式：主机模式和设备模式 非周期性发送 FIFO 空级别(Non-Periodic Tx FIFO Empty Level) 在主机模式下，该位表示何时会触发控制中断寄存器(GINTSTS.NPTXFEMP)的非周期性发送 FIFO 空中断位。 在设备模式下，该位表示何时会触发 IN 端点发送 FIFO 空中断(DIEPINTn.TXFEMP)。 0: DIEPINTn.TXFEMP 中断表示 IN 端点发送 FIFO 是半空 1: DIEPINTn.TXFEMP 中断表示 IN 端点发送 FIFO 是全空
位 6: 1	保留	0x00	resd	保持默认值。
位 0	GLBINTMSK	0x0	rw	模式：主机模式和设备模式 全局中断屏蔽(Global Interrupt Mask) 应用程序通过该位来屏蔽/取消屏蔽中断线发给自己的中断。无论该位是否置起，控制器仍会更新中断状态寄存器。 0: 屏蔽对应用程序的中断 1: 不屏蔽对应用程序的中断

#### 20.6.3.4 OTGFS\_USB配置寄存器(OTGFS\_GUSBCFG)

该寄存器用于在上电后或者在切换主机模式或设备模式时配置控制器。该寄存器包含了 USB 和 USB-PHY 相关的参数。应用程序在处理 AHB 或 USB 事务之前，必须先设置该寄存器。初始化配置之后，请不要再修改本寄存器。

域	简称	复位值	类型	功能
位 31	COTXPKT	0x0	rw	适用模式：主机模式和设备模式 发送包损坏(Corrupt Tx packet) 该位仅用于调试。请勿将该位设置为 1。
位 30	FDEVMODE	0x0	rw	适用模式：主机模式和设备模式 强制设备模式(Force Device Mode) 无论 ID 输入引脚的状态如何，向该位写 1 可强制控制器进入设备模式。 0: 普通模式 1: 强制设备模式 将该置起后，应用程序必须等待至少 25ms 才能使设置生效。
位 29	FHSTMODE	0x0	rw	适用模式：主机模式和设备模式 强制主机模式(Force Host Mode) 无论 ID 输入引脚的状态如何，向该位写 1 可强制控制器进入主机模式。 0: 普通模式 1: 强制主机模式 将该置起后，应用程序必须等待至少 25ms 才能使设置生效。
位 28: 15	保留	0x0000	resd	保持默认值。
位 14	保留	0x0	resd	保持默认值。
位 13: 10	USBTRDTIM	0x5	rw	适用模式：仅适用设备模式 USB 周转时间(USB Turnaround Time) 以 PHY 时钟为单位设置周转时间。规定了 MAC 向包 FIFO 控制器 (PFC) 发起请求从 DFIFO (SPRAM) 提取数据的响应时间。这些位必须配置为： 0101: 当 MAC 接口为 16-bit UTMI+时 1001: 当 MAC 接口为 8-bit UTMI+时。

				注意：以上数值是基于最低 30MHz 的 AHB 频率计算得出的。USB 周转时间对于用到长线和 5-Hubs 的认证至关重要，所以如果你需要 AHB 在低于 30 MHz 的频率下运行，而且 USB 周转时间不重要的话，可以将这些位域的值设置大一些。
位 9: 3	保留	0x00	resd	保持默认值。 适用模式：主机模式和设备模式 全速超时校准(FS Timeout Calibration)
位 2: 0	TOUTCAL	0x0	rw	将应用程序在这些位域设置的 PHY 时钟数添加到全速数据包间超时时段内，以补偿 PHY 引起的额外延迟。这个动作可能是必须的，因为 PHY 在生成线路状态条件时引起的延迟会因 PHY 不同而有所不同。 全速模式下，USB 标准超时值是 16~18 个 bit(包含 18)时间。应用程序必须要根据枚举的速度来设置此位域。每个 PHY 时钟增加的 bit 时间为 0.25bit 时间

### 20.6.3.5 OTGFS复位寄存器(OTGFS\_GRSTCTL)

应用程序通过本寄存器来复位控制器的各硬件模块。

域	简称	复位值	类型	功能
位 31	AHBIDLE	0x1	ro	适用模式：主机模式和设备模式 AHB 主机空闲(AHB Master Idle) 指示 AHB 主机状态机处于空闲状态。
位 30: 11	保留	0x000	resd	保持默认值。
位 10: 6	TXFNUM	0x00	rw	适用模式：主机模式和设备模式 发送 FIFO 编号(TxFIFO Number) 此位域表示哪个 FIFO 需要通过发送 FIFO 刷新位(TxFIFO Flush)来刷新。除非控制器已清除 TxFIFO Flush 位，否则不允许更改此位域。 00000: - 主机模式下非周期性 TxFIFO - 设备模式下 Tx FIFO 0 00001: - 主机模式下周期性 TxFIFO - 设备模式下 TXFIFO 1 00010: - 设备模式 TXFIFO 2 ... 01111: - 设备模式下 TXFIFO 15 10000: - 刷新位于设备模式或主机模式下的所有发送 FIFO
位 5	TXFFLSH	0x0	rw1s	适用模式：主机模式和设备模式 发送 FIFO 刷新(TxFIFO Flush) 该位有选择性地刷新单独一个的或所有的发送 FIFO，但是必须确保控制器没有正在处理的事务。 应用程序必须先确认控制器没有读/写 Tx FIFO，才能对此位进行写操作。 通过这些寄存器来验证： 读：NAK 有效中断(NAK Effective Interrupt)确保控制器没有读取 FIFO 写：GRSTCTL.AHBIDLE 确保控制器没有对 FIFO 进行写操作。 在重新配置 FIFO 时，通常建议用户进行刷新操作(Flushing)。 在设备端点禁用期间，也建议使用 FIFO 刷新操作。应用程序必须要等待控制器清除此位之后才能进行其它操作。 该位需要 8 个时钟来完成清除（以 phy_clk 或 hclk 的较慢的那个时钟为准来计算）。
位 4	RXFFLSH	0x0	rw1s	适用模式：主机模式和设备模式 接收 FIFO 刷新(Rx FIFO Flush)

<p>应用程序可以通过该位来刷新整个接收 FIFO，但必须先确保控制器没有正在处理的事务。应用程序必须先确认控制器没有读/写 RxFIFO，才能对此位进行写操作。应用程序必须要等待控制器清除此位之后才能进行其它操作。该位需要 8 个时钟周期来完成清除（以 PHY 或 AHB 最慢的那个时钟为准来计算）。</p>				
位 3	保留	0x0	resd	保持默认值。
位 2	FRMCNTRST	0x0	rw1s	<p>适用模式：仅适用主机模式 主机帧计数器复位(Host Frame Counter Reset) 应用程序通过此位来复位控制器的帧数计数器。将帧计数器复位之后，控制器随后发出的 SOF 的帧号为 0。 如果应用程序向该位写 1，它可能无法读取到这个值，因为该位会在几个时钟周期之后被控制器清除。</p>
位 1	PIUSFTRST	0x0	rw1s	<p>适用模式：主机模式和设备模式 PIU 全速专用控制器软件复位(PIU FS Dedicated Controller Soft Reset) 用于复位 PIU 全速专用控制器 PIU 全速专用控制器中的所有模块状态机即恢复到空闲状态。在发生 PHY 错误（比如操作中断或 babble）导致 PHY 保持在接收状态的时长超过 1 个帧时，可使用此位来复位 PIU 的全速专用控制器。 该位可自动清除，而且控制器中在所有必要逻辑电路复位之后会清除此位。</p>
位 0	CSFTRST	0x0	rw1s	<p>适用模式：主机模式和设备模式 控制器软件复位(Core Soft Reset) 复位 hclk and phy_clock 域，如下所示： 清除所有中断以及 CSR 寄存器，除如下寄存器位之外： - HCFG.FSLSPCS - DCFG.DECSPD - DCTL.SFTDIS 将所有模块状态机（除了 AHB 从机）复位到空闲状态，并清空所有的发送 FIFO 和接收 FIFO。 在完成最后一个阶段的 AHB 数据传输之后，AHB 主机上的所有任务都会尽可能快地结束。USB 上的所有任务会立即停止。 应用程序可以随时对该位进行写操作来复位控制器。此位可自动清除，控制器在将所有必要的逻辑电路复位之后会清除此位，控制器可能需要花几个时钟周期来清除，具体时间取决于控制器的当前所处的状态。清除此位后，应用程序必须要等待至少 3 个 PHY 时钟之后才能访问 PHY 域（同步延迟）。 另外，应用程序在开始其它操作之前，必须要确保本寄存器的位 31 是置 1(AHB 主机是空闲状态)。 通常来讲，使用软件复位的情况为：在进行软件开发，以及当用户对上述所列的 USB 配置寄存器的 PHY 选择位进行了动态修改的情况下，需要通过软件复位。在修改 PHY 时，需要选择相应的 PHY 时钟并运用于 PHY 域。选择了新的时钟之后，必须复位 PHY 域才能进行正常操作。</p>

### 20.6.3.6 OTGFS 中断寄存器(OTGFS\_GINTSTS)

本寄存器会因当前模式（设备模式或主机模式）发生系统事件而中断应用程序，如图 20-2 所示。本寄存器的一些位仅适用于主机模式，一些位仅适用于设备模式。另外，该寄存器会指示当前位于哪种操作模式。

FIFO 状态寄存器中断位是只读的。一旦软件在处理这些中断时读写了 FIFO，则 FIFO 中断条件会自动被清除。

初始化时，应用程序在使能某个中断位之前，必须先清除 GINTSTS 寄存器，以避免在初始化之前产生中断。

域	简称	复位值	类型	功能
位 31	WKUPINT	0x0	rw1c	适用模式：主机模式和设备模式

				检测恢复/远程唤醒信号产生中断(Resume/Remote Wakeup Detected Interrupt) 设备模式：仅当 USB 总线上检测到主机触发的恢复信号时，才产生中断 主机模式：仅当 USB 总线上检测到设备触发的远程唤醒信号时，才产生中断
位 30	保留	0x0	resd	保持默认值。
位 29	DISCONINT	0x0	rw1c	适用模式：仅适用于主机模式 检测到断开事件产生中断(Disconnect Detected Interrupt) 当检测到设备断开时，即产生中断
位 28	CONIDSCHG	0x0	rw1c	适用模式：主机模式和设备模式 连接器 ID 状态变化(Connector ID Status Change) 当检测到连接器 ID 状态发生变化时，控制器将此位置起
位 27	保留	0x0	resd	保持默认值。
位 26	PTXFEMP	0x1	ro	适用模式：仅适用于主机模式 周期性发送 FIFO 空(Periodic TxFIFO Empty) 当周期性发送 FIFO (Periodic Transmit FIFO) 处于半空或全空状态，且在周期性请求队列中有空间可写入一个请求时，即产生中断。具体是半空还是全空，取决于控制器 AHB 配置寄存器的周期性发送 FIFO 空级别位(Periodic TxFIFO Empty Level)。
位 25	HCHINT	0x0	ro	主机通道中断(Host Channels Interrupt) 控制器通过将该位置起，来指示控制器（主机模式下）的某一个通道有待处理的中断。应用程序必须读取主机全通道中断寄存器(Host All Channels Interrupt)以确定产生中断的具体通道编号，然后读取相应的主机通道编号中断寄存器(Host Channel-n Interrupt)以获取中断源。 应用程序必须通过清除 HCINTn (Host All Channels Interrupt)寄存器的相应状态位来清除该位。
位 24	PRTINT	0x0	ro	主机端口中断(Host Port Interrupt) 控制器通过将该位置起，来指示在主机模式下某个端口状态发生改变。应用程序必须通过读取主机端口控制和状态寄存器(Host Port Control and Status)来确认中断源。应用程序必须通过清除主机端口控制和状态寄存器(Host Port Control and Status)来清除此位。
位 23: 22	保留	0x0	resd	保持默认值。
位 21	INCOMPPIP INCOMPISOOUT	0x0	rw1c	未完成周期性传输(Incomplete Periodic Transfer) 适用模式：仅适用于主机模式 在主机模式下，如果当前帧尚有未完成的周期性传输待处理时，控制器会置起此中断位。 未完成同步 OUT 传输(Incomplete Isochronous OUT Transfer) 适用模式：仅适用于设备模式 在设备模式下，控制器置起此中断位以指示当前帧至少存在一个未完成传输的同步 OUT 端点。该中断会随同本寄存器的周期性帧结束中断位(End of Periodic Frame Interrupt)一同生成。
位 20	INCOMPISOIN	0x0	rw1c	适用模式：仅适用于设备模式 未完成同步 IN 传输(Incomplete Isochronous IN Transfer) 控制器置起此中断以指示当前帧至少存在一个未完成传输的同步 IN 端点。该中断会随同本寄存器的周期性帧结束中断位(End of Periodic Frame Interrupt)一同生成。
位 19	OEPTINT	0x0	ro	适用模式：仅适用于设备模式 OUT 端点中断(OUT Endpoints Interrupt) 控制器通过置起此位来指示控制器的某一个 OUT 端点有待处理的中断。应用程序必须通过读取设备全部端点中断寄存器(Device All Endpoints Interrupt)以确定该中断发生在哪一个 OUT 端点号上，接着会读取相应的设备 OUT 端点号中断寄存器(Device OUT Endpoint-n Interrupt)来确定本次中断源。应用程序必须通过清除设备 OUT 端点中断

				寄存器(Device OUT Endpoint-n Interrupt)的相应状态位来清除此位
位 18	IEPTINT	0x0	ro	适用模式: 仅适用于设备模式 <b>IN 端点中断(IN Endpoints Interrupt)</b> 控制器通过置起此位来指示控制器的某一个 IN 端点有待处理的中断(设备模式下)。应用程序必须通过读取设备全部端点中断寄存器(Device All Endpoints Interrupt)以确定该中断发生在哪一个 IN 端点号上, 接着会读取相应的设备 IN 端点号中断寄存器(Device IN Endpoint-n Interrupt)来确定本次中断源。应用程序必须通过清除相应的 Device IN Endpoint-n Interrupt 寄存器的相应状态位来清除此位。
位 17: 16	保留	0x0	resd	保持默认值。
位 15	EOPF	0x0	rw1c	适用模式: 仅适用于设备模式 <b>周期性帧结束中断(End of Periodic Frame Interrupt)</b> 该位表示当前帧已经到达了设备配置寄存器(Device Configuration)的周期帧间隔时间位所设置的周期。
位 14	ISOOUTDROP	0x0	rw1c	适用模式: 仅适用于设备模式 <b>同步 OUT 包丢失中断(Isochronous OUT Packet Dropped Interrupt)</b> 控制器在以下情况会置起该位: 控制器因为接收 FIFO 没有足够的空间来容纳同步 OUT 端点所需的最大容量的数据包而导致其无法向接收 FIFO 写入一个同步 OUT 包。
位 13	ENUMDONE	0x0	rw1c	适用模式: 仅适用于设备模式 <b>完成枚举(Enumeration Done)</b> 控制器将该位置起以表示已完成速度枚举。 应用程序必须通过读取设备状态寄存器(Device Status)来获取枚举的速度信息。
位 12	USBRST	0x0	rw1c	适用模式: 仅适用于设备模式 <b>USB 复位(USB Reset)</b> 控制器将该位置起以表示检测到 USB 总线上的复位信号。
位 11	USBSUSP	0x0	rw1c	适用模式: 仅适用于设备模式 <b>USB 挂起(USB Suspend)</b> 控制器将该位置起以表示检测到 USB 总线上的挂起信号。当总线信号在很长一段时间内没有活动时, 控制器进入挂起状态。
位 10	ERLYSUSP	0x0	rw1c	适用模式: 仅适用于设备模式 <b>早期挂起(Early Suspend)</b> 控制器将该位置起以表示检测到 USB 总线空闲状态已持续 3ms。
位 9: 8	保留	0x0	resd	保持默认值。
位 7	GOUTNAKEFF	0x0	ro	适用模式: 仅适用于设备模式 <b>全局 OUT NAK 生效(Global OUT NAK Effective)</b> 该位表示设备控制寄存器(Device Control)设置 Global OUT NAK 位已生效。通过向设备控制寄存器(Device Control)写入 Clear Global OUT NAK 位可以清除该位。
位 6	GINNAKEFF	0x0	ro	适用模式: 仅适用于设备模式 <b>全局非周期性 IN NAK 生效(Global IN Non-periodic NAK Effective)</b> 该位表示由应用程序所设置的设备控制寄存器(Device Control)的 Set Global Non-periodic IN NAK 位已生效。即, 控制器对应用程序所设置的 Global IN NAK 位进行了采样。通过向设备控制寄存器(Device Control)写入 Clear Global Non-periodic IN NAK 位可以清除该位。此中断并不一定意味着 USB 总线上发送了 NAK 握手信号。STALL 位的优先级高于 NAK 位。
位 5	NPTXFEMP	0x1	ro	适用模式: 主机模式和设备模式 <b>非周期性发送 FIFO 空(Non-periodic Tx FIFO Empty)</b> 当非周期性发送 FIFO 处于半空或全空状态, 并且有足够的空间可以向非周期性发送请求队列(Non-periodic

				Transmit Request Queue)写入至少一个请求时, 即产生中断。具体是半空还是全空, 取决于控制器 AHB 配置寄存器(Core AHB Configuration)的非周期性发送 FIFO 空级别位(Non-periodic TxFIFO Empty Level)。
位 4	RXFLVL	0x0	ro	适用模式: 主机模式和设备模式 接收 FIFO 非空(RxFIFO Non-Empty) 表示接收 FIFO 中至少有一个包等待读取。
位 3	SOF	0x0	rw1c	适用模式: 主机模式和设备模式 帧首包(Start of Frame) 在主机模式下, 控制器将该位置起以指示 USB 总线发送了 SOF (全速)或者 Keep-Alive(低速)信号。 应用程序必须将位置 1 才能清除此中断。 在设备模式下, 控制器将该位置起以表示 USB 总线接收到到了 SOF 令牌。应用程序必须读取设备状态寄存器(Device Status register)来获取当前帧号。只有当控制器工作在全速模式时, 才能产生此中断。该位只能由控制器设置, 应用程序必须写 1 才能清除此位。 注意: 上电复位后立即读取该寄存器可能会返回 0x1。如果在上电复位后立即读取寄存器的值为 0x1, 并不表示已发送 SOF(主机模式下)或已收到 SOF(设备模式下)。只有当主机和设备建立有效连接后, 读取的寄存器的值才是有效的。如果在上电复位后将该位置起, 那么应用程序可清除此位。
位 2	OTGINT	0x0	ro	适用模式: 主机模式和设备模式 OTG 中断(OTG Interrupt) 控制器将该位置起以表示产生了一个 OTG 协议事件。应用程序必须读取 OTG 中断状态寄存器(OTG Interrupt Status)来确定是什么事件引起的中断。应用程序必须通过清除 OTG 中断状态寄存器(OTG Interrupt Status)的相应状态位以清除此位。
位 1	MODEMIS	0x0	rw1c	适用模式: 主机模式和设备模式 模式不匹配中断(Mode Mismatch Interrupt) 当应用程序尝试访问以下内容时, 控制器会置起此位: 当控制器运行在设备模式下时却尝试访问主机模式下的寄存器 当控制器运行在主机模式下时却尝试访问设备模式下的寄存器 在 AHB 上完成对寄存器的访问之后会出现 OKAY 响应, 但控制器内部会忽略此种操作, 所以不会影响到控制器的运行。 此位只能由控制器设置, 应用程序通过写 1 才能清除此位。
位 0	CURMOD	0x0	ro	适用模式: 主机模式和设备模式 当前运行模式(Current Mode of Operation) 此位指示的是当前的运行模式。 0: 设备模式 1: 主机模式

### 20.6.3.7 OTGFS 中断屏蔽寄存器(OTGFS\_GINTMSK)

本寄存器与“中断寄存器”(Interrupt Register)互相配合来中断应用程序。屏蔽某一个中断位后, 就不会生成与该中断位相关的中断。然而与该中断所对应的中断寄存器位仍然是置起状态。

屏蔽中断: 0

解除中断: 1

域	简称	复位值	类型	功能
位 31	WKUPINTMSK	0x0	rw	适用模式: 主机模式和设备模式 屏蔽由恢复/远程唤醒检测到的中断(Resume/Remote Wakeup Detected Interrupt Mask)
位 30	保留	0x0	resd	保持默认值。
位 29	DISCONINTMSK	0x0	rw	适用模式: 主机模式和设备模式

				屏蔽断开事件检测到的中断(Disconnect Detected Interrupt Mask)
位 28	CONIDSCHGMSK	0x0	rw	适用模式：主机模式和设备模式 屏蔽连接器 ID 状态改变(Connector ID Status Change Mask)
位 27	保留	0x0	resd	保持默认值。
位 26	PTXFEMPMASK	0x0	rw	适用模式：仅适用于主机模式 屏蔽周期性发送 FIFO 空(Periodic TxFIFO Empty Mask)
位 25	HCHINTMSK	0x0	rw	适用模式：仅适用于主机模式 主机通道中断屏蔽(Host Channels Interrupt Mask)
位 24	PRTINTMSK	0x0	ro	适用模式：仅适用于主机模式 主机端口中断屏蔽(Host Port Interrupt Mask)
位 23: 22	保留	0x0	resd	保持默认值。
位 21	INCOMPIPMASK INCOMPISOOUTMSK	0x0	rw	未完成周期性传输屏蔽(Incomplete Periodic Transfer Mask) 适用模式：仅适用于主机模式 未完成同步 OUT 传输屏蔽 (Incomplete Isochronous OUT Transfer Mask) 适用模式：仅适用于设备模式
位 20	INCOMISOINMSK	0x0	rw	适用模式：仅适用于设备模式 未完成同步 IN 传输屏蔽(Incomplete Isochronous IN Transfer Mask)
位 19	OEPTINTMSK	0x0	rw	适用模式：仅适用于设备模式 OUT 端点中断屏蔽(OUT Endpoints Interrupt Mask)
位 18	IEPTINTMSK	0x0	rw	适用模式：仅适用于设备模式 IN 端点中断屏蔽(IN Endpoints Interrupt Mask)
位 17	保留	0x0	rw	保持默认值。
位 16	保留	0x0	resd	保持默认值。
位 15	EOPFMSK	0x0	rw	适用模式：仅适用于设备模式 周期性帧结束中断屏蔽(End of Periodic Frame Interrupt Mask)
位 14	ISOOUTDROPMSK	0x0	rw	适用模式：仅适用于设备模式下的同步 OUT 包丢失中断屏蔽(Device only Isochronous OUT Packet Dropped Interrupt Mask)
位 13	ENUMDONEMSK	0x0	rw	适用模式：仅适用于设备模式 枚举完成中断屏蔽(Enumeration Done Mask)
位 12	USBRSTMSK	0x0	rw	适用模式：仅适用于设备模式 USB 复位中断屏蔽(USB Reset Mask)
位 11	USBSUSPMSK	0x0	rw	适用模式：仅适用于设备模式 USB 挂起中断屏蔽(USB Suspend Mask)
位 10	ERLYSUSPMSK	0x0	rw	适用模式：仅适用于设备模式 早期挂起中断屏蔽(Early Suspend Mask)
位 9: 8	保留	0x0	resd	保持默认值。
位 7	GOUTNAKEFFMSK	0x0	rw	适用模式：仅适用于设备模式 全局 OUT NAK 有效中断屏蔽(Global OUT NAK Effective Mask)
位 6	GINNAKEFFMSK	0x0	rw	适用模式：仅适用于设备模式 全局非周期性 IN NAK 有效中断屏蔽(Global Non-periodic IN NAK Effective Mask)
位 5	NPTXFEMPMASK	0x0	rw	适用模式：主机模式和设备模式 非周期性发送 FIFO 空中断屏蔽(Non-periodic TxFIFO Empty Mask)
位 4	RXFLVLMSK	0x0	rw	适用模式：主机模式和设备模式 接收 FIFO 非空中断屏蔽(Receive FIFO Non-Empty Mask)
位 3	SOFMSK	0x0	rw	适用模式：主机模式和设备模式 帧开始中断屏蔽(Start of Frame Mask)
位 2	OTGINTMSK	0x0	rw	适用模式：主机模式和设备模式 OTG 中断屏蔽(OTG Interrupt Mask)。
位 1	MODEMISMSK	0x0	rw	适用模式：主机模式和设备模式 模式不匹配中断屏蔽_Mode Mismatch Interrupt Mask)

位 0	保留	0x0	resd	保持默认值。
-----	----	-----	------	--------

### 20.6.3.8 OTGFS接收状态调试读/OTG状态读和POP寄存器 (OTGFS\_GRXSTSR / OTGFS\_GRXSTSP)

读取“接收状态调试读寄存器(Receive Status Debug Read)”会返回接收 FIFO(Receive FIFO)顶层的数据。读取“接收状态读和 PoP 寄存器(Receive Status Read and Pop)”还会弹出接收 FIFO 顶层的数据。

在主机模式和设备模式下，对接收状态内容的解释并不相同。当接收 FIFO 为空时，控制器将忽略接收状态弹出/读取，并返回 0x0000 0000。当控制器中断寄存器(Core Interrupt register) 的接收 FIFO 非空位被置起时，应用程序才能弹出接收状态 FIFO。

**主机模式：**

域	简称	复位值	类型	功能
位 31: 21	保留	0x000	resd	保持默认值。 数据包状态(Packet Status) 表示接收到的数据包的状态。 0010: 接收到 IN 数据包 0011: 完成 IN 传输(触发中断) 0101: 数据翻转出错(触发中断) 0111: 通道中止(触发中断) 其它: 保留 复位值: 0
位 20: 17	PKTSTS	0x0	ro	数据 PID(Data PID) 表示接收到的数据包的数据 PID 00: DATA0 10: DATA1 01: DATA2 11: MDATA 复位值: 0
位 16: 15	DPID	0x0	ro	字节数(Byte Count) 表示已接收到的 IN 数据包的字节数
位 14: 4	BCNT	0x000	ro	通道数(Channel Number) 表示当前接收到的数据包属于哪一个通道
位 3: 0	CHNUM	0x0	ro	

**设备模式：**

域	简称	复位值	类型	功能
位 31: 25	保留	0x00	resd	保持默认值。 帧号(Frame Number) 表示 USB 总线上所收到的数据包的帧号的最低 4 位。 此位域仅适用于支持同步 OUT 端点功能的情况。
位 24: 21	FN	0x0	ro	数据包状态(Packet Status) 表示接收到的数据包的状态。 0001: 全局 OUT NAK(触发中断) 0010: 收到 OUT 数据包 0011: 完成 OUT 传输(触发中断) 0100: 完成 SETUP 任务(触发中断) 0110: 收到 SETUP 数据包 其它: 保留
位 20: 17	PKTSTS	0x0	ro	数据 PID(Data PID) 表示接收到的 OUT 数据包的数据 PID 00: DATA0 10: DATA1 01: DATA2 11: MDATA
位 16: 15	DPID	0x0	ro	字节数(Byte Count) 表示接收到的数据包的字节数
位 14: 4	BCNT	0x000	ro	
位 3: 0	EPTNUM	0x0	ro	端点号(Endpoint Number)

---

表示当前所收到的数据包属于哪一个端点号

---

### 20.6.3.9 OTGFS接收FIFO长度寄存器(OTGFS\_GRXFSIZ)

应用程序可以对必须要分配给接收 FIFO 的 RAM 长度进行设置。

域	简称	复位值	类型	功能
位 31: 16	保留	0x0000	resd	保持默认值。
位 15: 0	RXFDEP	0x0200	ro/rw	接收 FIFO 深度(RxFifo Depth) 此值以 32 位字为单位。 最小值为 16 最大值为 512 在配置期间, 本寄存器的上电复位值被定义为接收数据 FIFO 的最大深度。

---

### 20.6.3.10 OTGFS非周期性TX FIFO长度寄存器(OTGFS\_GNPTXFSIZ)/端点0 TX FIFO长度寄存器(OTGFS\_DIEPTXF0)

应用程序可设置非周期性发送 FIFO 的 SRAM 长度和存储器起始地址, 该寄存器的位域随主机模式或设备模式而变化。

主机:

域	简称	复位值	类型	功能
位 31: 16	NPTXFDEP	0x0000	ro/rw	非周期性发送 FIFO 深度(Non-periodic TxFIFO depth) 这个数值的单位是 32 位的字 最小值是 16 最大值是 256。
位 15: 0	NPTXFSTADDR	0x0200	ro/rw	非周期性发送 SRAM 起始地址(Non-periodic Transmit SRAM Start Address) 此位域包含了非周期性发送 FIFO SRAM (Non-periodic Transmit FIFO SRAM) 的存储器起始地址。

---

设备:

域	简称	复位值	类型	功能
位 31: 16	INEPT0TXDEP	0x0000	ro/rw	IN 端点发送 FIFO 0 深度(IN Endpoint TxFIFO 0 Depth) 这个数值的单位是 32 位的字 最小值是 16 最大值是 256。
位 15: 0	INEPT0TXSTADDR	0x0200	ro/rw	IN 端点 FIFO 0 发送 SRAM 起始地址(IN Endpoint FIFO0 Transmit SRAM Start Address ) 此位域包含了 IN 端点发送 FIFO 0 的存储器起始地址

---

### 20.6.3.11 OTGFS非周期性TX FIFO/请求队列状态寄存器(OTGFS\_GNPTXSTS)

本寄存器仅适用于主机模式, 本寄存器为只读, 储存了非周期性发送(FIFO Non-periodic TxFIFO) 和非周期性发送请求队列(Non-periodic Transmit Request Queue)的可用空间信息。

域	简称	复位值	类型	功能
位 31	保留	0x0	resd	保持默认值。
位 30: 24	NPTXQTOP	0x00	ro	非周期性发送请求队列的顶部(Top of the Non-periodic Transmit Request Queue) 表示 MAC 正在处理非周期性发送请求队列的请求 位[30: 27]: 通道/端点号 位[26: 25]: 00: IN/OUT 令牌 01: 零长度的发送包 (设备 IN/主机 OUT) 10: PING/CSPLIT 令牌 11: 通道中止指令 位[24]: 结束 (所选通道/端点的最后一个请求)
位 23: 16	NPTXQSPCAVAIL	0x08	ro	非周期性发送请求队列的可用空间(Non-periodic Transmit Request Queue Space Available) 表示非周期性发送请求队列的可用空间。在主机模式下, 此队列既支持 IN 请求也支持 OUT 请求。

---

				00: 非周期性发送请求队列的空间已满 01: 1 个位置可用 02: 2 个位置可用 n: n 个位置可用(0 ≤ n ≤ 8) 其它: 保留 复位值: 可设置
位 15: 0	NPTXFSPCAVAIL	0x0200	ro	非周期性发送 FIFO 的可用空间(Non-periodic TxFIFO Space Avail) 表示非周期性发送 FIFO 的可用空间, 此值以 32 位字节为单位。 00: 非周期性发送 FIFO 已满 01: 1 个字可用 02: 2 个字可用 n: n 个字可用 (0 ≤ n ≤ 256) 其它: 保留 复位值: 可设置

### 20.6.3.12 OTGFS通用控制器配置寄存器(OTGFS\_GCCFG)

域	简称	复位值	类型	功能
位 31: 22	保留	0x000	resd	保持默认值。
位 21	VBUSIG	0x0	rw	VBUS 忽略 (VBUS ignored) 由于本版芯片未出 VBUS 管脚, 用户在使用时, 请务必设置该控制位为 1, 从而 OTGFS 不监测 VBUS 引脚电压, 并且认为在主机和设备模式下, VBUS 电压一直有效。 0: 保留, 请勿使用 1: VBUS 被忽略, 并认为 VBUS 电压一直有效
位 20	SOFOUTEN	0x0	rw	SOF 输出使能(SOF output enable) 0: 不输出 SOF 脉冲; 1: 输出 SOF 脉冲到引脚上。
位 19	BVALIDSESEN	0x0	rw	Bvalid 监测使能 (sense Bvalid enable) 0: 不使能 1: 使能
位 18	AVALIDSESEN	0x0	rw	Avalid 监测使能 (sense Avalid enable) 0: 不使能 1: 使能
位 17	保留	0x000	resd	保持默认值。
位 16	PWRDOWN	0x0	rw	掉电(Power down) 用于在发送和接收时激活收发器, 必需预先配置才能允许 USB 通信 0: 使能掉电; 1: 禁止掉电(收发器激活)。
位 15: 0	保留	0x0000	resd	保持默认值。

### 20.6.3.13 OTGFS控制器ID寄存器(OTGFS\_GUID)

此寄存器只读, 保护产品的 ID。

域	简称	复位值	类型	功能
31: 0	USERID	0x0000 1000	rw	产品 ID (Product ID field) 应用程序可以编程此 ID 位。

### 20.6.3.14 OTGFS主机周期性发送FIFO长度寄存器(OTGFS\_HPTXFSIZ)

此寄存器保存着周期性发送 FIFO 的深度和存储器起始地址

域	简称	复位值	类型	功能
位 31: 16	PTXFSIZE	0x02000	ro/rw	主机周期性发送 FIFO 深度 (Host periodic TxFIFO depth) 此位域的值以 32 位字为单位。 最小值是 16 最大值是 512
位 15: 0	PTXFSTADDR	0x0600	ro/rw	主机周期性发送 FIFO 起始地址(Host Periodic TxFIFO Start Address)

该寄存器的上电复位值指的是接收数据 FIFO 最大深度与非周期性发送数据 FIFO 最大深度之和。

### 20.6.3.15 OTGFS设备IN端点发送FIFO长度寄存器

#### (OTGFS\_DIEPTXF<sub>n</sub>)(其中n是FIFO的编号, x=1…3)

本寄存器保存着在设备模式下进行的 IN 端点发送 FIFO 的深度以及存储器起始地址。每个 FIFO 包含一个 IN 端点数据。本寄存器可重复用于实例化的 IN 端点 FIFO1~15。通过 GNPTXFSIZ 寄存器来设置 IN 端点 FIFO 0 的深度及内存起始地址。

域	简称	复位值	类型	功能
位 31: 16	INEPTXFDEP	0x0200	ro/rw	IN 端点发送 FIFO 深度(IN Endpoint TxFIFO Depth) 以 32 位字为单位。 最小值为 16 最大值为 512 复位值是最大可能的 IN 端点发送 FIFO 的深度。
位 15: 0	INEPTXFSTADDR	0x0400	ro/rw	IN 端点 FIFOOn 发送 SRAM 起始地址(IN Endpoint FIFO Transmit SRAM Start Address) 此值为 IN 端点 n 发送 FIFO 在 SRAM 中的起始地址。

### 20.6.4 主机模式下的寄存器

该寄存器影响主机模式下控制器的运行状况。不支持在设备模式下进行访问(因为设备模式下访问结果未定义)。主机模式下的寄存器分别为:

#### 20.6.4.1 OTGFS主机模式配置寄存器(OTGFS\_HCFG)

该寄存器用于上电后配置控制器。初始化之后，不得再更改本寄存器。

域	简称	复位值	类型	功能
位 31: 3	保留	0x0000 0000	resd	保持默认值。
位 2	FSLSSUPP	0x0	ro	仅支持全速和低速设备(FS- and LS-Only Support) 应用程序通过此位来控制内核的枚举速度。应用程序可通过此位将控制器以全速主机模式来枚举，即使已连接的设备支持高速通信。在初始化设置之后，不得再更改此位域。 0: 全速/低速，具体情况取决于连接设备所支持的最大速度 1: 仅支持全速/低速，即使所连接的设备支持高速模式。
位 1: 0	FSLSPCLKSEL	0x0	rw	全速/低速 PHY 时钟选择(FS/LS PHY Clock Select) 当控制器处于全速主机模式下： 01: PHY 时钟运行频率为 48MHz 其它值：保留 当控制器处于低速主机模式下： 00: 保留； 01: PHY 时钟运行频率为 48MHz。 10: PHY 时钟运行频率为 6MHz。如果选择了 6MHz 时钟，则必须进行软件复位。 11: 保留

#### 20.6.4.2 OTGFS主机帧间隔寄存器(OTGFS\_HFIR)

该寄存器用于设置当前枚举速度的帧间隔。

域	简称	复位值	类型	功能
位 31: 17	保留	0x0000	resd	保持默认值。
位 16	HFIIRLDCTRL	0x0	rw	重新加载控制(Reload Control) 该位允许在运行时对主机帧间隔寄存器进行动态重新加载。 1: 不能动态重新加载主机帧间隔寄存器 0: 可以在运行时动态重新加载主机帧间隔寄存器 该位需在初始化时设置好，并且在运行期间不得更改其值。
位 15: 0	FRINT	0xEA60	rw	帧间隔(Frame Interval)

应用程序通过此位域来设置是两个连续 SOF (全速) 之间的间隔。

此位域中的 PHY 时钟个数即表示帧间隔。只有当主机端口控制及状态寄存器的端口使能被设置后，应用程序才可以写入主机帧间隔寄存器。

如果未设定此位域，控制器将根据主机配置寄存器的 FS/LS PHY 时钟选择位所定义的 PHY 时钟频率来计算其值。初始化配置之后，不得再更改其值。

1 ms \* (全速/低速 PHY 时钟频率)

### 20.6.4.3 OTGFS 主机帧号/帧时间剩余寄存器(OTGFS\_HFNUM)

该寄存器指示当前帧号，以及当前帧剩余时间（以 PHY 时钟个数表示）。

域	简称	复位值	类型	功能
位 31: 16	FTREM	0x0000	ro	帧剩余时间(Frame Time Remaining) 指示当前帧（全速/低速）还剩余多少时间，以 PHY 时钟个数表示。此位域的值会随每个 PHY 时钟个数而自动递减。当值减为 0 时，帧间隔寄存器的值会重新加载到此位域，并且向 USB 总线发出一个新 SOF。
位 15: 0	FRNUM	0x3FFF	ro	帧号(Frame Number) 每当向 USB 总线发出一个新 SOF 时，此位域的值就会递增一次；当值增加到 16'h3FFF 时，此位域即复位为 0。

### 20.6.4.4 OTGFS 主机周期性发送 FIFO/请求队列寄存器(OTGFS\_HPTXSTS)

该寄存器为只读，包含周期性发送 FIFO 和周期性发送请求队列的剩余空间信息。

域	简称	复位值	类型	功能
位 31: 24	PTXQTOP	0x00	ro	周期性发送请求队列的顶层(Top of the Periodic Transmit Request Queue) 表示 MAC 正在处理的周期性发送请求队列的请求。该寄存器用于模块调试。 位[31]: 奇数/偶数帧 0: 偶数帧发送 1: 奇数帧发送 位[30: 27]: 通道/端点号 位[26: 25]: 类型 00: IN/OUT 01: 零长度包 10: 保留 11: 禁止通道指令 位[24]: 终止（所选通道或端点的最后一个请求）
位 23: 16	PTXQSPCAVAIL	0x08	ro	周期性发送请求队列剩余空间(Periodic Transmit Request Queue Space Available) 指示周期性发送请求队列里允许写入的可用空间。此队列包括 IN 和 OUT 请求。 00: 周期性发送请求队列满额 01: 1 个可用空间 10: 2 个可用空间 n: n 个可用空间( $0 \leq n \leq 8$ ) 其它值: 保留
位 15: 0	PTXFSPCAVAIL	0x0100	rw	周期性发送数据 FIFO 剩余空间(Periodic Transmit Data FIFO Space Available) 指示周期性发送 FIFO 的可用空间，以 32 位字为单位 0000: 周期性发送 FIFO 满额 0001: 1 个字可用 0010: 2 个字可用 n: n 个字可用( $0 \leq n \leq 512$ ) 其它值: 保留

### 20.6.4.5 OTGFS 主机所有通道中断寄存器(OTGFS\_HINT)

当某个通道产生了标志事件时，主机所有通道中断寄存器(OTGFS\_HAINT)将通过控制器中断寄存器的主机通道中断位来中断应用程序，如图 20-2 所示。每个通道都有一个中断位，共 16 位。应用程序通过设置或清除相应主机通道 n 中断寄存器的相应位来设置或清除此位。

域	简称	复位值	类型	功能
位 31: 16	保留	0x0000	resd	保持默认值。
位 15: 0	HINT	0x0000	ro	通道中断(Channel Interrupts) 每个通道对应一个位：通道 0 对应位 0，通道 15 对应位 15。

#### 20.6.4.6 OTGFS 主机所有通道中断屏蔽寄存器(OTGFS\_HINTMSK)

主机所有通道中断屏蔽寄存器(OTGFS\_HINTMSK)与主机所有通道中断寄存器(OTGFS\_HINT)配置使用，用于控制在产生事件时是否打断应用程序。每个通道都有一个相对应的中断屏蔽位，共有 16 位。

域	简称	复位值	类型	功能
位 31: 16	保留	0x0000	resd	保持默认值。
位 15: 0	HINTMSK	0x0000	rw	通道中断屏蔽(Channel Interrupt Mask) 每个通道对应一个位：位 0 对应通道 0，位 15 对应通道 15。

#### 20.6.4.7 OTGFS 主机端口控制和状态寄存器(OTGFS\_HPRT)

该寄存器仅适用于主机模式。OTG 主机目前仅支持一个端口。

该寄存器包含 USB 端口相关的信息，比如每个端口的 USB 复位，使能，挂起，唤醒，连接状态以及测试模式等信息，如图 21-2 所示。类型为 rw1c 的寄存器可以通过控制器中断寄存器的主机端口中断位来中断应用程序。端口发生中断时，应用程序必须读取该寄存器，并且清除导致该中断的位。类型为 rw1c 的寄存器，应用程序需要写 1 来清除中断。

域	简称	复位值	类型	功能
位 31: 19	保留	0x0000	resd	保持默认值。
位 18: 17	PRTSPD	0x0	ro	端口速度(Port Speed) 表示连上端口的设备的速度。 00: 保留 01: 全速 10: 低速 11: 保留
位 16: 13	PRTTSTCTL	0x0	rw	端口测试控制(Port Test Control) 应用程序通过向此位域写入一个非零值而将此端口置于测试模式，且端口会发出相应信号。 0000: 测试模式关闭 0001: Test_J 模式 0010: Test_K 模式 0011: Test_SE0_NAK 模式 0100: Test_Packet 模式 0101: Test_Force_Enable 其它值: 保留
位 12	PRTPWR	0x0	rw	端口供电(Port Power) 应用程序通过此位来控制对该端口的供电（写 1 或写 0）。 0: 断电 1: 供电 注意：该位与接口无关。应用程序需按照编程手册中的主机编程流程为各个接口设置此位。
位 11: 10	PRTLNSTS	0x0	ro	端口线状态(Port Line Status) 表示当前 USB 数据线的逻辑状态。 位[10]: D+的逻辑电平 位[11]: D- 的逻辑电平
位 9	保留	0x0	resd	保持默认值。
位 8	PRTRST	0x0	rw	端口复位(Port Reset)

				当应用程序置起此位后，端口会启动复位序列。应用程序必须计算此次复位序列所需时间，并在复位结束后清除此位。 0: 端口不复位 1: 端口复位 应用程序必须使该位保持置起状态至少达到 USB2.0 规范第 7.1.7.5 章节规定的最短持续时间才能启动端口复位。除了最短持续时间外，应用程序在清除此位之前还可以额外增加 10ms，USB 规范没有设定最长持续时间。
位 7	PRTSUSP	0x0	rw1s	端口挂起(Port Suspend) 通过将该位置起，应用程序可使端口进入挂起模式，此时，控制器只停止发送 SOF。应用程序必须通过设置端口时钟停止位才能关闭 PHY 时钟。 读取该位将返回当前端口的挂起状态。 当控制器器检测到远程唤醒信号时会清除此位，或者当应用程序将该寄存器的端口复位位或端口唤醒位置起，或者将控制器中断寄存器的唤醒/远程唤醒检测中断或断开连接检测中断位置起后，也会清除此位。 即便设备未与主机相连，控制器仍然可以清除此位。 0: 端口未处于挂起模式 1: 端口处于挂起模式
位 6	PRTRES	0x0	rw	端口唤醒(Port Resume) 应用程序通过设置此位来驱动端口发出唤醒信号。控制器会持续触发唤醒信号直到应用程序清除此位。如果控制器检测到 USB 远程唤醒序列（按照控制器中断寄存器的端口唤醒/远程唤醒检测中断位的指示），控制器将不受应用程序干预自动触发唤醒信号。 读取此位时将返回控制器是否正在驱动唤醒信号的信息。 0: 有触发唤醒 1: 触发唤醒
位 5	PRTOVRCCHNG	0x0	rw1c	端口过流状态改变(Port Overcurrent Change) 当该寄存器的端口过电流有效位(位 4)的状态发生改变时，控制器将置起此位。只能通过控制器来设置此位，应用程序必须写 1 才能清除此位。
位 4	PRTOVRCACT	0x0	ro	端口过流有效位(Port Overcurrent Active) 表示端口的过流状况 0: 不存在过电流 1: 存有过电流
位 3	PRTENCHNG	0x0	rw1c	端口使能/禁止状态改变(Port Enable/Disable Change) 当该寄存器的端口使能位 2 的状态发生改变时，控制器将此位置起。只能通过控制器将该位置起，应用程序必须写 1 才能清除此位。
位 2	PRTENA	0x0	rw1c	端口使能(Port Enable) 此端口只有在复位序列后才能被控制器使能。在发生过电流，断开连接或者应用程序将此位清除时，即禁用此端口。应用程序不能通过写入寄存器来设置此位，只能通过清除此位来禁用端口。该位不会触发中断。 0: 端口禁用 1: 端口使能
位 1	PRTCONDET	0x0	rw1c	端口连接检测(Port Connect Detected) 控制器在检测到设备连接端口时，会通过控制器中断寄存器的主机端口中断位来设置该位。该位只能由控制器设置，应用程序必须写 1 才能清除此位。
位 0	PRTCONSTS	0x0	ro	端口连接状态(Port Connect Status) 0: 没有设备连接端口 1: 设备连接端口

#### 20.6.4.8 OTGFS主机通道x特性寄存器(OTGFS\_HCCHARx)(此处x代码通道号, x = 0...8)

域	简称	复位值	类型	功能
---	----	-----	----	----

				通道使能(Channel Enable) 此位由应用程序设置，并由 OTG 主机清除。 0: 通道禁止 1: 通道使能
位 31	CHENA	0x0	rw1s	通道禁止(Channel Disable) 应用程序设置此位来停止发送/接收通道上的数据，即便通道传输还没完成。应用程序必须等到通道禁止中断产生，才视该通道已禁止。
位 29	ODDFRM	0x0	rw	奇数帧(Odd Frame) 应用程序通过设置/复位此位来指示 OTG 主机是否以奇数帧来传输。此位仅适用于周期性（同步和中断）传输。 0: 偶数帧 1: 奇数帧
位 28: 22	DEVADDR	0x00	rw	设备地址(Device Address) 此位域用于选择可作为数据源或接收器的指定设备。
位 21: 20	MC	0x0	rw	多次计数 Multi Count (MC) 此位域通知主机该周期性端口的每个帧要执行的传输数目。 00: 保留，此位域生成未定义的结果 01: 1 个事务 10: 每个帧有 2 个事务 11: 每个帧有 3 个事务 此位域至少应该设置为 0x01。
位 19: 18	EPTYPE	0x0	rw	端点类型(Endpoint Type) 表示所选择的传输类型。 00: 控制传输 01: 同步传输 10: 批传输 11: 中断传输
位 17	LSPDDEV	0x0	rw	低速设备(Low-Speed Device) 应用程序通过设置此位来指示该通道正在与低速设备通信。
位 16	保留	0x0	resd	保持默认值。
位 15	EPTDIR	0x0	rw	端点方向(Endpoint Direction) 指示传输是 IN 还是 OUT 0: OUT 1: IN
位 14: 11	EPTNUM	0x0	rw	端点号(Endpoint Number) 指示设备（用作数据源或接收器）的端点号。
位 10: 0	MPS	0x000	rw	最大包长度(Maximum Packet Size) 指示相应端点的最大包长度。

#### 20.6.4.9 OTGFS 主机通道 x 中断寄存器(OTGFS\_HCINTx)(其中 x 代表通道号, x=0...8)

该寄存器包含着与 USB 和 AHB 相关事件的通道状态，如图 20-2 所示。当控制器中断寄存器的主机通道中断位置起时，应用程序必须读取该寄存器。在读寄存器之前，应用程序必须先读取主机所有通道中断寄存器以获理主机通道 n 中断寄存器的确切通道编号。应用程序必须通过清除该寄存器的相应位来清除 OTGFS 主机所有通道中断寄存器(OTGFS\_HAIINT)和 OTGES 中断寄存器 (OTGFS\_GINTSTS) 的相应位。

域	简称	复位值	类型	功能
位 31: 11	保留	0x000000	resd	保持默认值。
位 10	DTGLERR	0x0	rw1c	数据翻转错误(Data Toggle Error) 只能由控制器设置此位，应用程序通过写 1 来清除此位。
位 9	FRMOVRUN	0x0	rw1c	帧溢出(Frame Overrun) 只能由控制器设置此位，应用程序通过写 1 来清除此位。
位 8	BBLERR	0x0	rw1c	Babble Error 只能由控制器设置此位，应用程序通过写 1 来清除此位。
位 7	XACTERR	0x0	rw1c	传输错误(Transaction Error) 表示 USB 总线发生了下列错误：

				CRC 校验失败 传输超时 位填充错误 EOP 错误 只能由控制器设置此位，应用程序通过写 1 来清除此位。
位 6	保留	0x0	resd	保持默认值。
位 5	ACK	0x0	rw1c	ACK 响应已收到/已发送中断(ACK Response Received/Transmitted Interrupt) 只能由控制器设置此位，应用程序通过写 1 来清除此位。
位 4	NAK	0x0	rw1c	NAK 响应已收到中断(NAK Response Received Interrupt) 只能由控制器设置此位，应用程序通过写 1 来清除此位。
位 3	STALL	0x0	rw1c	STALL 响应已收到中断(STALL Response Received Interrupt) 只能由控制器设置此位，应用程序通过写 1 来清除此位。
位 2	保留	0x0	resd	保持默认值。
位 1	CHHLTD	0x0	rw1c	通道中止(Channel Halted) 此位指示传输异常结束，原因是 USB 传输出错或者为响应应用程序因传输完成而发出的中止请求。
位 0	XFERC	0x0	rw1c	传输完成(Transfer Completed) 传输正常结束，未报错。此位只能由控制器设置，应用程序通过写 1 来清除此位。

#### 20.6.4.10 OTGFS 主机通道 x 中断屏蔽寄存器(OTGFS\_HCINTMSKx)(其中 x 为通道号，x=0...8)

本寄存器用于屏蔽上一节所描述的各类通道中断。

域	简称	复位值	类型	功能
位 31: 11	保留	0x000000	resd	保持默认值。
位 10	DTGLERRMSK	0x0	rw	数据翻转错误屏蔽(Data Toggle Error Mask)
位 9	FRMOVRUNMSK	0x0	rw	帧溢出屏蔽(Frame Overrun Mask)
位 8	BBLERRMSK	0x0	rw	Babble 错误屏蔽(Babble Error Mask)
位 7	XACTERRMSK	0x0	rw	传输错误屏蔽(Transaction Error Mask)
位 6	NYETMSK	0x0	rw	NYET 响应已收到中断屏蔽(NYET Response Received Interrupt Mask)
位 5	ACKMSK	0x0	rw	ACK 响应已收到/已发送中断屏蔽(ACK Response Received/Transmitted Interrupt Mask)
位 4	NAKMSK	0x0	rw	NAK 响应已收到中断屏蔽(NAK Response Received Interrupt Mask)
位 3	STALLMSK	0x0	rw	STALL 响应已收到中断屏蔽(STALL Response Received Interrupt Mask)
位 2	保留	0x0	resd	保持默认值。
位 1	CHHLTDMASK	0x0	rw	通道中止屏蔽(Channel Halted Mask)
位 0	XFERCMASK	0x0	rw	传输完成屏蔽(Transfer Completed Mask)

#### 20.6.4.11 OTGFS 主机通道 x 传输长度寄存器(OTGFS\_HCTSIZx)(其中 x 为通道号，x=0...8)

域	简称	复位值	类型	功能
位 31	保留	0x0	resd	保持默认值。
位 30: 29	PID	0x0	rw	PID (Pid) 应用程序在此位域设置了用于首次传输的 PID 类型。主机控制着此位域用于后续的传输任务。 00: DATA0 01: DATA2 10: DATA1 11: MDATA(非控制)/SETUP(控制)
位 28: 19	PKTCNT	0x000	rw	包数目(Packet Count) 应用程序在此位域设置了期望发送或期望接收的包数目。 主机在每次成功发送/接收 OUT/IN 包时会递减包数目。当数据减为 0 时，应用程序将收到中断以指示传输正常结束。

位 18: 0	XFERSIZE	0x00000	rw	传输长度(Transfer Size) 对于 OUT 传输来说，此位域指示主机在传输过程中发送的数据字节数。 对于 IN 传输来说，此位域指示应用程序为传输预留的缓冲区长度。 对于 IN 传输（周期性和非周期性），应用程序需要将此位域设置为最大包长度的整数倍。
---------	----------	---------	----	---

## 20.6.5 设备模式下的寄存器

这些寄存器仅用于设备模式。主机模式不允许访问，因为访问结果未知。其中有些寄存器会影响到所有端点，而有些寄存器只影响某一个端点。

### 20.6.5.1 OTGFS设备配置寄存器(OTGFS\_DCFG)

在复位后、特殊控制命令后或者枚举后，此寄存器用于配置设备模式下的控制器。在初始化之后，请不要再修改此寄存器。

域	简称	复位值	类型	功能
位 31: 13	保留	0x0110	resd	保持默认值。
位 12: 11	PERFRINT	0x0	rw	周期性帧间隔(Periodic frame interval) 此位域表示产生“周期性帧结束中断”位时占某个帧内的时间比。应用程序可以通过此中断来确认帧内的同步传输是否已完成。 00: 80%的帧时间； 01: 85%的帧时间； 10: 90%的帧时间； 11: 95%的帧时间。
位 10: 4	DEVADDR	0x00	rw	设备地址(Device address) 应用程序在每次收到设置地址（SetAddress）控制命令后必须设置此位域。
位 3	保留	0x0	resd	保持默认值。
位 2	NZSTSOUTSHHK	0x0	rw	非零长度的状态 OUT 握手信号(Non-zero-length status OUT handshake) 在控制传输的状态阶段，如果收到了非零长度的数据包，控制器会发出握手信号，应用程序正是通过此位来选择控制器要发送的握手信号。 1: 向非零长度的状态 OUT 传输发送一个 STALL 握手信号，但不向应用程序传送所收到的 OUT 数据包 0: 向应用程序传递所收到的 OUT 数据包（要么零长度，要么非零长度），并且根据设备端点控制寄存器的 NAK 和 STALL 位选择发送握手信号。
位 1: 0	DEVSPD	0x0	rw	设备速度(Device speed) 此位域指示的是应用程序需要控制器进行枚举的速度，或者应用程序可支持的最大速度。然而，总线实际速度只能在整个序列完成之后，才能根据控制器与 USB 主机相连的速度来决定。 00: 保留； 01: 保留； 10: 保留； 11: 全速(USB1.1 收发器，时钟为 48MHz)。

### 20.6.5.2 OTGFS设备控制寄存器(OTGFS\_DCTL)

域	简称	复位值	类型	功能
位 31: 12	保留	0x00000	resd	保持默认值。
位 11	PWROPRGDNE	0x0	wo	上电配置结束(Power-on programming done) 应用程序通过此位来指示从断电模式唤醒后，完成了对该寄存器的配置。
位 10	CGOUTNAK	0x0	wo	清除全局 OUT NAK (Clear global OUT NAK) 对此位进行写操作可以清除全局 OUT NAK 状态。
位 9	SGOUTNAK	0x0	wo	设置全局 OUT NAK (Set global OUT NAK) 对此位进行写操作可以设置全局 OUT NAK 状态。

				应用程序通过此位向所有 OUT 端点发送一个 NAK 握手信号。只有在确认已清除了控制器中断寄存器的全局 OUT NAK 有效位时，应用程序才需要设置此位。
位 8	CGNPINNAK	0x0	wo	清除全局非周期性 IN NAK (Clear Global Non-periodic IN NAK) 对此位写操作可以清除全局非周期性 IN NAK 状态。
位 7	SGNPINNAK	0x0	wo	设置全局非周期性 IN NAK (Set global Non-periodic IN NAK) 对此位写操作可以设置全局非周期性 IN NAK 状态。 应用程序通过此位向所有非周期性 IN 端点发送一个 NAK 握手信号。应用程序只有在确认了控制器中断寄存器的全局 IN NAK 有效信号已清除的情况下才设置此位。
位 6: 4	TSTCTL	0x0	rw	测试控制(Test control) 000: 测试模式禁止; 001: Test_J 模式; 010: Test_K 模式; 011: Test_SE0_NAK 模式; 100: Test_Packet 模式; 101: Test_Force_Enable; 其他: 保留。
位 3	GOUTNAKSTS	0x0	ro	全局 OUT NAK 状态 (Global OUT NAK status) 0: 根据 FIFO 状态, NAK 和 STALL 位的设定来发送握手信号 1: 不管是否有可用空间, 都不会写数据到接收 FIFO。向所有数据包(除了 SETUP 传输)发送一个 NAK 握手信号。丢弃所有同步 OUT 数据包。
位 2	GNPINNAKSTS	0x0	ro	全局非周期性 IN NAK 状态 (Global Non-periodic IN NAK status) 0: 根据发送 FIFO 的数据状况来发送握手信号 1: 不管发送 FIFO 内的数据状况如何, 都会向所有的非周期性 IN 端点发送 NAK 握手信号。
位 1	SFTDISCON	0x1	rw	软件断开(Soft disconnect) 应用程序通过此位指示 OTGFS 控制器进行“软件断开”操作。一旦置起此位, 那么主机会看到设备已断开, 设备也收不到 USB 总线信号。控制器就始终处于断开状态直至应用程序清除此位。 0: 普通操作。当此位在设备软件断开后清除, 控制器会向主机发起一个设备连接事件。USB 主机与设备重新连接后, 会重启设备枚举。
位 0	RWKUPSIG	0x0	rw	远程唤醒信号(Remote wakeup signaling) 应用程序置起此位后, 控制器会启动远程信号来唤醒 USB 主机。应用程序必须置起此位来指示控制器退出挂起状态。 按照 USB2.0 的规定, 应用程序在设置此位 1 - 15 ms 后必须清除该位。

为了使 USB 主机检测到设备断开, 软件断开位在各种状态下保持置起的最短持续时间在表 20-5 给出。为了适应时钟抖动, 建议应用程序在规定的最短持续时间的基础上再添加一些额外的延迟。

表 20-5 软件断开的最短持续时间

操作速度	设备状态	最短持续时间
全速	挂起	1ms + 2.5us
全速	空闲	2.5us
全速	不空闲或挂起(正在执行传输操作)	2.5us

### 20.6.5.3 OTGFS设备状态寄存器(OTGFS\_DSTS)

此寄存器指示控制器与 OTGFS 相关的状态。此寄存器用于在发生设备所有中断(OTGFS\_DAINT)寄存器事件时读取。

域	简称	复位值	类型	功能
位 31: 22	保留	0x000	resd	保持默认值。
位 21: 8	SOFFN	0x0000	ro	接收到的 SOF 的帧号(Frame number of the received SOF) 注意：如果在上电复位后立即读取此位域，可能返回非零值。如果在上电复位后立即读取此位域时返回的是非零值，并不表示已经收到主机发出的 SOF。只有当主机和设备进行连接后，此位域的读取值才是有效的。
位 7: 4	保留	0x1	resd	保持默认值。
位 3	ETICERR	0x0	ro	随机误差(Erratic error) 此类错误会导致控制器挂起，并且控制器中断寄存器的早期挂起位会置起，随后产生中断。如果是因为异常错误而触发控制器早期挂起，那么应用程序只能执行软件断开进行修复解决。
位 2: 1	ENUMSPD	0x0	ro	枚举速度(Enumerated speed) 表示控制器在通过序列进行速度检测后所确定的执行速度。 01: 保留； 10: 保留； 11: 全速(PHY 时钟运行在 48MHz)； 其他：保留。
位 0	SUSPSTS	0x0	ro	挂起状态(Suspend status) 在设备模式下，只要检测到 USB 总线上有挂起条件，此位就会置起。当 USB 总线信号长时间不活动时，控制器即会进入挂起状态。 控制器在遇到下列情况时会退出挂起状态： ■USB 总线信号开始活动 ■应用程序对设备控制寄存器的远程唤醒信号位进行写操作

#### 20.6.5.4 OTGFS设备OTGFSIN端点通用中断屏蔽寄存器(OTGFS\_DIEPMSK)

本寄存器与各个端点的设备 IN 端点中断寄存器配合工作以产生 IN 端点中断。可以通过向设备 IN 端点通用中断屏蔽寄存器的相应位进行写操作来屏蔽设备 IN 端点中断寄存器(OTGFS\_DIEPINTx)中某个特定状态的 IN 端点中断。状态位是默认屏蔽的。

域	简称	复位值	类型	功能
位 31: 10	保留	0x000000	resd	保持默认值。
位 9	BNAINMSK	0x0	rw	BNA 中断屏蔽(BNA interrupt mask) 0: 中断屏蔽； 1: 中断不屏蔽。
位 8	TXFIFOUDRMSK	0x0	rw	FIFO 欠载中断屏蔽(FIFO underrun mask) 0: 中断屏蔽； 1: 中断不屏蔽。
位 7	保留	0x0	resd	保持默认值。
位 6	INEPTNAKMSK	0x0	rw	IN 端点 NAK 状态有效中断屏蔽(IN endpoint NAK effective mask) 0: 中断屏蔽； 1: 中断不屏蔽。
位 5	INTKNEPTMISMSK	0x0	rw	端点收到 IN 命令不匹配中断屏蔽(IN token received with EP mismatch mask) 0: 中断屏蔽； 1: 中断不屏蔽。
位 4	INTKNTXFEMPMSK	0x0	rw	当发送 FIFO 空时收到 IN 命令中断屏蔽(IN token received when Tx FIFO empty mask) 0: 中断屏蔽； 1: 中断不屏蔽。
位 3	TIMEOUTMSK	0x0	rw	超时条件中断屏蔽(非同步端点) (Timeout condition mask (Non-isochronous endpoints)) 0: 中断屏蔽； 1: 中断不屏蔽。

位 2	保留	0x0	resd	保持默认值。
位 1	EPTDISMSK	0x0	rw	端点禁用中断屏蔽(Endpoint disabled interrupt mask) 0: 中断屏蔽; 1: 中断不屏蔽。
位 0	XFERCMSK	0x0	rw	传输完成中断屏蔽(Transfer completed interrupt mask) 0: 中断屏蔽; 1: 中断不屏蔽。

### 20.6.5.5 OTGFS设备OUT端点通用中断屏蔽寄存器(OTGFS\_DOEPMSK)

此寄存器配合设备 OUT 端点中断寄存器(OTGFS\_DOEPINTx)使用，产生 OUT 端点中断。OTGFS 设备 OUT 端点 x 中断寄存器(OTGFS\_DOEPINTx)的每一位都可以通过写此寄存器的相应位来屏蔽。在默认状态下，所有中断都屏蔽。

域	简称	复位值	类型	功能
位 31: 10	保留	0x000000	resd	保持默认值。
位 9	BNAOUTMSK	0x0	rw	BNA 中断屏蔽 (BNA interrupt mask) 0: 中断屏蔽; 1: 中断不屏蔽。
位 8	OUTPERRMSK	0x0	rw	OUT 包错误中断屏蔽 (OUT packet error mask) 0: 中断屏蔽; 1: 中断不屏蔽。
位 7	保留	0x0	resd	保持默认值。
位 6	B2BSETUPMSK	0x0	rw	收到连续的 SETUP 包中断屏蔽 (Back-to-back SETUP packets received mask) 0: 中断屏蔽; 1: 中断不屏蔽。
位 5	保留	0x0	resd	保持默认值。
位 4	OUTTEPDMSK	0x0	rw	当端点被禁止时收到 OUT 命令中断屏蔽 (OUT token received when endpoint disabled mask) 0: 中断屏蔽; 1: 中断不屏蔽。
位 3	SETUPMSK	0x0	rw	SETUP 阶段完成中断屏蔽 (SETUP phase done mask) 仅对控制端点有效 0: 中断屏蔽; 1: 中断不屏蔽。
位 2	保留	0x0	resd	保持默认值。
位 1	EPTDISMSK	0x0	rw	端点被禁止中断屏蔽 (Endpoint disabled interrupt mask) 0: 中断屏蔽; 1: 中断不屏蔽。
位 0	XFERCMSK	0x0	rw	传输结束中断屏蔽 (Transfer completed interrupt mask) 0: 中断屏蔽; 1: 中断不屏蔽。

### 20.6.5.6 OTGFS设备所有端点中断寄存器(OTGFS\_DAINT)

当某个端点发生事件时，设备所有端点中断寄存器(OTGFS\_DAINT)可以通过设置控制器中断寄存器的设备 IN 端点中断位或设备 OUT 端点中断位来中断应用程序。每个端点有一个中断位，其中 OUT 端点和 IN 端点各自拥有多达 8 个中断位。对于双向端点而言，同时使用对应的 IN 中断位和 OUT 中断位。应用程序通过设置和清除对应的设备端点 x 中断寄存器来设置和清除设备所有端点中断寄存器(OTGFS\_DAINT)的相应位。

域	简称	复位值	类型	功能
位 31: 24	保留	0x0000	resd	保持默认值。
位 23: 16	OUTEPTINT	0x0000	ro	OUT 端点中断位 (OUT endpoint interrupt bits) 每个位对应一个 OUT 端点。位 16 对应 OUT 端点 0，位 18 对应 OUT 端点 2。
位 15: 8	保留	0x0000	resd	保持默认值。
位 7: 0	INEPTINT	0x0000	ro	IN 端点中断位 (IN endpoint interrupt bits) 每个位对应一个 IN 端点。位 0 对应 IN 端点 0，位 7 对应 IN 端点 7。

### 20.6.5.7 OTGFS所有端点中断屏蔽寄存器(OTGFS\_DAINTMSK)

当某个设备端点发生事件时，设备端点中断屏蔽寄存器与设备端点中断寄存器共同配合来中断应用程序。尽管如此，与该中断相对应的设备所有端点中断寄存器(OTGFS\_DAINT)仍是置起状态。

域	简称	复位值	类型	功能
位 31: 24	保留	0x0000	resd	保持默认值。
位 23: 16	OUTEPTMSK	0x0000	rw	OUT 端点中断屏蔽寄存器 (OUT EP interrupt mask bits) 每个位对应一个 OUT 端点。 位 16 对应 OUT 端点 0，位 18 对应 OUT 端点 2。 0: 屏蔽中断； 1: 不屏蔽中断。
位 15: 8	保留	0x0000	resd	保持默认值。
位 7: 0	INEPTMSK	0x0000	rw	IN 端点中断屏蔽位 (IN EP interrupt mask bits) 每个位对应一个 IN 端点。 位 0 对应 IN 端点 0，位 7 对应 IN 端点 7。 0: 屏蔽中断； 1: 不屏蔽中断。

### 20.6.5.8 OTGFS设备IN端点FIFO空中断屏蔽寄存器(OTGFS\_DIEPEMPMSK)

此寄存器配合 IN 端点 FIFO 空中断寄存器(TXFE\_OTGFS\_DIEPINTx)产生中断。

域	简称	复位值	类型	功能
位 31: 8	保留	0x0000	resd	保持默认值。
位 7: 0	INEPTXFEMSK	0x0000	rw	IN 端点发送 FIFO 空中断屏蔽位 (IN EP Tx FIFO empty interrupt mask bits) 此位域是设备 IN 端点中断寄存器的屏蔽位。 一个发送 FIFO 空中断位对应一个端点：位 0 对应 IN 端点 0，位 7 对应 IN 端点 7。 0: 屏蔽中断； 1: 不屏蔽中断。

### 20.6.5.9 OTGFS设备控制IN端点0控制寄存器(OTGFS\_DIEPCTL0)

本节介绍了控制 IN 端点 0 控制寄存器。非零控制端点使用端点 1-7 寄存器。

域	简称	复位值	类型	功能
位 31	EPTENA	0x0	rw1s	端点使能(Endpoint enable) 此应用程序设置此位，以启动在端点 0 上的数据传输。 控制器在产生以下端点中断前会先清除此位： ■ 端点禁止； ■ 传输完成。
位 30	EPTDIS	0x0	ro	端点禁用(Endpoint disable) 应用程序通过设置此位可以在完成传输前停止端点上的数据传输。应用程序必须要等到端点禁用中断产生才认为端点已禁用。控制器在生成端点禁用中断之前先清除此位。 只有端点使能的情况下，应用程序才需要置起此位。
位 29: 28	保留	0x0	resd	保持默认值。
位 27	SNAK	0x0	wo	设置 NAK (Set NAK) 通过对该位写操作可以设置端点的 NAK 位。应用程序可通过此位来控制端点上的 NAK 握手信号的发送。当该端点接收到 SETUP 数据包时，控制器也会置起此位。
位 26	CNAK	0x0	wo	清除 NAK (Clear NAK) 对该位写操作可清除端点的 NAK 位。
位 25: 22	TXFNUM	0x0	rw	发送 FIFO 的编号(TxFIFO number) 控制端点 0 只能使用 FIFO0
位 21	STALL	0x0	rw1s	STALL 握手(STALL handshake) 应用程序置起此位，控制器会在收到 SETUP 令牌时清除此位。如果该位连同 NAK 位，全局非周期性 IN NAK 位或者全局 OUT NAK 位同时置起时，STALL 位享有优先级。
位 20	保留	0x0	resd	保持默认值。

位 19: 18	EPTYPE	0x0	ro	端点类型(Endpoint type) 对于控制端点，由硬件置为 00。
位 17	NAKSTS	0x0	ro	NAK 状态 (NAK status) 表示为： 0: 表示控制器根据 FIFO 状态发送非 NAK 握手信号 1: 表示控制器正在发送 NAK 握手信号。 当此位置起时（无论是应用程序设置的还是控制器设置的），控制器都会停止发送数据，即使发送 FIFO 有可用空间。不管该位是否置起，控制器始终以 ACK 握手信号来响应 SETUP 数据包。
位 16	保留	0x0	resd	保持默认值。
位 15	USBACEPT	0x0	ro	USB 有效端点(USB active endpoint) 此位始终置 1，表示不管在什么配置和接口中，控制端点 0 始终有效。
位 14: 2	保留	0x0000	resd	保持默认值。
位 1: 0	MPS	0x0	rw	适用 IN 端点和 OUT 端点。 应用程序通过此位设置当前逻辑端点的最大数据包长度。 00: 64 字节 01: 32 字节 10: 16 字节 11: 8 字节

### 20.6.5.10 OTGFS设备IN端点x控制寄存器(OTGFS\_DIEPCTLx)(其中x为端点号，x=1…3)

应用程序通过这些寄存器控制非 0 端点的操作特性。

域	简称	复位值	类型	功能
位 31	EPTENA	0x0	rw1s	端点使能 (Endpoint enable) 应用程序设置此位，表示该端点准备发送数据 控制器在产生以下端点中断之前会先清除该位： ■ SETUP 阶段完成 ■ 端点禁用 ■ 传输完成
位 30	EPTDIS	0x0	rw1s	端点禁用 (Endpoint disable) 应用程序可通过设置此位停止某个端点上的数据发送/传输，即使传输还没完成。 应用程序需要等到端点禁用中断产生才视为该端点已被禁用。控制器在设置端点禁用中断前会清除此位。应用程序只有在端点已经使用端点时才能设置此位。
位 29	SETD1PID/ SETODDFR	0x0	wo	设置 DATA1 PID (Set DATA1 PID) 仅适用于中断/批量 IN 端点，写此位可以将该寄存器的端点数据 PID 位设置为 DATA1。
位 28	SETD0PID/ SETEVENFR	0x0	rw	设置奇数帧 (Set odd frame) 仅适用于同步 IN 端点，写此位可以将奇偶帧位设置为奇数帧。 0: 关闭 Set DATA1 PID 或者不强制奇数帧 1: 使能 Set DATA1 PID 或者强制奇数帧
位 27	SNAK	0x0	wo	设置 DATA0 PID (Set DATA0 PID) 仅适用于中断/批量 IN 端点，写此位可将该寄存器的端点数据 PID 位设置为 DATA0。
位 26	SETEVENFR	0x0	rw	设置偶数帧 (Set Even frame) 仅适用于同步 IN 端点，写此位可以将偶数帧/奇数帧位设置为偶数帧。 0: 关闭 Set DATA0 PID 或不强制偶数帧 1: 端点数据 PID 设置为 DATA0 或将 EOFNUM 设置为偶数帧
位 27	SNAK	0x0	wo	设置 NAK (Set NAK)

				通过对该位写操作可以设置端点的 NAK 位。应用程序可通过此位来控制端点上的 NAK 握手信号的发送。控制器在接收到 SETUP 数据包或传输结束中断时置起此位。 数值： 0: 未设置 NAK 1: 设置 NAK
位 26	CNAK	0x0	wo	清除 NAK (Clear NAK) 通过对该位写操作可以清除端点的 NAK 位。 0: 不清除 NAK 1: 清除 NAK
位 25: 22	TXFNUM	0x0	rw	发送 FIFO 的编号 (TxFIFO number) 给对应端点分配 FIFO 编号，必须给每个有效的 IN 端点分配一个独立的 FIFO 编号。此位仅对 IN 端点有效。
位 21	STALL	0x0	rw	STALL 握手 (STALL handshake) 适用于非控制非同步 IN 端点和 OUT 端点。 应用程序通过此位停止向该端点发送 USB 主机的令牌。 如果 NAK 位、全局非周期性 IN NAK 位或者全局 OUT NAK 位和该位同时置起时，STALL 位享有优先级。只有应用程序才能清除此位，控制器不能。 0: 停止发送所有无效令牌 1: 停止发送所有有效令牌
位 20	保留	0x0	resd	保持默认值。
位 19: 18	EPTYPE	0x0	rw	端点类型 (Endpoint type) 表示逻辑端点支持的传输类型 00: 控制; 01: 同步; 10: 块传输; 11: 中断。
位 17	NAKSTS	0x0	ro	NAK 状态 (NAK status) 指示以下状态： 0: 表示控制器根据 FIFO 状态发送非 NAK 握手信号 1: 表示控制器正在发送 NAK 握手信号。 当此位置起时（无论是应用程序设置的还是控制器设置的） ■ 控制器停止接收 OUT 端点上的数据，即使接收 FIFO 有剩余空间可以用来容纳传入的数据包。 ■ 对于非同步 IN 端点：控制器停止发送端点上的数据，即使发送 FIFO 还有待发送的数据。 ■ 对于同步 IN 端点：控制器会发出一个零长度的数据包，即使发送 FIFO 还有剩余空间。 不管该位是否置起，控制器始终以 ACK 握手信号来响应 SETUP 数据包。
位 16	DPID/ EOFRNUM	0x0	ro	端点数据 PID (Endpoint data PID) 仅适用于中断/批量 IN 端点。 此位包含的是该端点上要传输或要发送的数据包的 PID 信息。在端点使能后，应用程序需要设置该端点待发送或接收的首个数据包的 PID。应用程序通过该寄存器的 SetD1PID 和 SetD0PID 来设置 DATA0 或 DATA1 PID。 0: DATA0 1: DATA1
				奇/偶数帧 (Even/odd frame) 仅适用于同步 IN 端点。 表示控制器传输或接收该端点的同步数据的帧号。应用程序通过该寄存器的 SETEVNFR 和 SETODDFR 设置希望传输或接收同步数据的偶数帧号/奇数帧号。 0: 偶数帧 1: 奇数帧
位 15	USBACEPT	0x0	rw	USB 活跃端点 (USB active endpoint) 指示在当前配置和接口下，该端点是否活跃。控制器在检测到 USB 复位之后会清除此位(除了端点 0)。 应用程序

				在收到 SetConfiguration 和 SetInterface 命令时必须设置端点寄存器并置起此位。 0: 不活跃 1: 活跃
位 14: 11	保留	0x0	resd	保持默认值。
位 10: 0	MPS	0x000	rw	最大包长度 (Maximum packet size) 应用程序通过此位设置当前逻辑端点的最大包长度, 以字节为单位。

### 20.6.5.11 OTGFS设备控制OUT端点0控制寄存器(OTGFS\_DOEPCTL0)

本节介绍了控制 OUT 端点 0 控制寄存器。非零控制端点使用端点 1-7 寄存器。

域	简称	复位值	类型	功能
位 31	EPTENA	0x0	rw1s	端点使能 (Endpoint enable) 应用程序设置此位, 以启动端点 0 上的数据传输。 控制器在产生以下端点中断之前会先清除该位: ■ SETUP 阶段完成; ■ 端点禁止; ■ 传输完成。
位 30	EPTDIS	0x0	ro	端点禁用 (Endpoint disable) 应用程序不能禁止控制 OUT 端点 0
位 29: 28	保留	0x0	resd	保持默认值。
位 27	SNAK	0x0	wo	设置 NAK (Set NAK) 通过对该位写操作可以设置端点的 NAK 位。应用程序可通过此位来控制端点上的 NAK 握手信号的发送。控制器在接收到 SETUP 数据包或传输结束中断时置起此位。
位 26	CNAK	0x0	wo	清除 NAK (Clear NAK) 通过对该位写操作可以清除端点的 NAK 位。
位 25: 22	保留	0x0	resd	保持默认值。
位 21	STALL	0x0	rw1s	STALL 握手 (STALL handshake) 应用程序置起此位, 控制器会在收到 SETUP 令牌时清除此位。如果该位连同 NAK 位, 全局非周期性 OUT NAK 位同时置起时, STALL 位享有优先级。 不管该位是否置起, 控制器始终以 ACK 握手信号来响应 SETUP 数据包。
位 20	SNP	0x0	rw	监听模式 (Snoop mode) 此位将端点配置为 Snoop 模式。在 Snoop 模式下, 控制器在将 OUT 包传输给应用存储器之前不会检查 OUT 包是否正确。
位 19: 18	EPTYPE	0x0	ro	端点类型 (Endpoint type) 硬件设置将此位设置为 0 用于控制端点类型。
位 17	NAKSTS	0x0	ro	NAK 状态 (NAK status) 表示为: 0: 表示控制器根据 FIFO 状态发送非 NAK 握手信号 1: 表示控制器正在发送 NAK 握手信号。 当此位置起时 (无论是应用程序设置的还是控制器设置的), 控制器都会停止接收数据, 即使接收 FIFO 有可用空间。不管该位是否置起, 控制器始终以 ACK 握手信号来响应 SETUP 数据包。
位 16	保留	0x0	resd	保持默认值。
位 15	USBACCEPT	0x1	ro	USB 有效端点 (USB active endpoint) 此位始终置 1, 表示不管在什么配置和接口中, 控制端点 0 始终有效。
位 14: 2	保留	0x0000	resd	保持默认值。
位 1: 0	MPS	0x0	ro	最大包长度 (Maximum packet size) 控制 OUT 端点 0 的最大包长度与控制器 IN 端点 0 的设置是一样的, 如下: 00: 64 字节; 01: 32 字节; 10: 16 字节; 11: 8 字节。

### 20.6.5.12 OTGFS设备OUT端点x控制寄存器(OTGFS\_DOEPCTLx)(其中x为端点号, x=1…3)

应用程序通过此寄存器来控制除了端点0的其他端点的操作特性。

域	简称	复位值	类型	功能
位 31	EPTENA	0x0	rw1s	端点使能 (Endpoint enable) 此位表示已设置好描述符结构和准备接收数据的数据缓冲区。控制器在产生以下端点中断之前会先清除该位： <ul style="list-style-type: none"> <li>■ SETUP 阶段完成</li> <li>■ 端点禁用</li> <li>■ 传输完成</li> </ul>
位 30	EPTDIS	0x0	ro	端点禁用 (Endpoint disable) 应用程序可通过设置此位来停止某个端点上的数据发送/传输，即使传输还没完成。 应用程序需要等到端点禁用中断产生才视为该端点已被禁用。控制器在设置端点禁用中断前会清除此位。应用程序只有在端点已经使用端点时才能设置此位。 数值： 0：无作用 1：端点禁用
位 29	SETD1PID/ SETODDFR	0x0	rw	设置 DATA1 PID (Set DATA1 PID) 仅适用于中断/批量 OUT 端点，写此位可以将该寄存器的端点数据 PID 位设置为 DATA1。
位 28	SETD0PID/ SETEVENFR	0x0	rw	设置奇数帧 (Set odd frame) 仅适用于同步 OUT 端点，写此位可以将奇偶帧位设置为奇数帧。 0：关闭 Set DATA1 PID 或者不强制奇数帧 1：使能 Set DATA1 PID 或者强制奇数帧
位 27	SNAK	0x0	wo	设置 DATA0 PID (Set DATA0 PID) 仅适用于中断/批量 OUT 端点，写此位可将该寄存器的端点数据 PID 位设置为 DATA0。
位 26	CNAK	0x0	wo	设置偶数帧 (Set Even frame) 仅适用于同步 OUT 端点，写此位可以将偶数帧/奇数帧位设置为偶数帧。 0：关闭 Set DATA0 PID 或不强制偶数帧 1：端点数据 PID 设置为 DATA0 或将 EOFRNUM 设置为偶数帧
位 25: 22	保留	0x0	resd	设置 NAK (Set NAK) 通过对该位写操作可以设置端点的 NAK 位。应用程序可通过此位来控制端点上的 NAK 握手信号的发送。控制器在接收到 SETUP 数据包或传输结束中断时置起此位。 数值： 0：未设置 NAK 1：设置 NAK
位 21	STALL	0x0	rw	清除 NAK (Clear NAK) 通过对该位写操作可以清除端点的 NAK 位。 数值： 0：不清除 NAK 1：清除 NAK
位 20	SNP	0x0	rw	适用于非控制，非同步 IN 端点和 OUT 端点。 应用程序通过此位停止向该端点发送 USB 主机的令牌。 如果 NAK 位，全局非周期性 IN NAK 位或者全局 OUT NAK 位以及该位同时置起时，STALL 位享有优先级。只有应用程序才能清除此位，控制器不能。

				设置此位将使端点进入监听模式。在监听模式下，控制器在将 OUT 数据包写入应用程序缓存区前不检查数据的正确性。
位 19: 18 EPTYPE	0x0	rw		<p>端点类型 (Endpoint type) 表示逻辑端点支持的传输类型。</p> <p>00: 控制 01: 同步 10: 块传输 11: 中断</p>
位 17 NAKSTS	0x0	ro		<p>NAK 状态 (NAK status) 0: 表示控制器根据 FIFO 状态发送非 NAK 握手信号 1: 表示控制器正在发送 NAK 握手信号。 当此位置起时（无论是应用程序设置的还是控制器设置的）</p> <ul style="list-style-type: none"> <li>■ 控制器停止接收 OUT 端点上的数据，即使接收 FIFO 有剩余空间可以用来容纳传入的数据包。</li> <li>■ 对于非同步 IN 端点：控制器停止发送端点上的数据，即使发送 FIFO 还有待发送的数据。</li> <li>■ 对于同步 IN 端点：控制器会发出一个零长度的数据包，即使发送 FIFO 还有剩余空间。</li> </ul> <p>不管该位是否置起，控制器始终以 ACK 握手信号来响应 SETUP 数据包。</p>
位 16 DPID/ EOFRNUM	0x0	ro		<p>端点数据 PID (Endpoint data PID) 仅适用于中断/批量 OUT 端点。</p> <p>此位包含的是该端点上要传输或要发送的数据包的 PID 信息。在端点使能后，应用程序需要设置该端点待发送或接收的首个数据包的 PID。应用程序通过该寄存器的 SetD1PID 和 SetD0PID 来设置 DATA0 或 DATA1 PID。</p> <p>0: DATA0 1: DATA1</p> <p>奇/偶数帧 (Even/odd frame) 仅适用于同步 OUT 端点。</p> <p>表示控制器传输或接收该端点的同步数据的帧号。应用程序通过该寄存器的 SETEVNFR 和 SETODDFR 设置希望传输或接收同步数据的偶数帧号/奇数帧号。</p> <p>0: 偶数帧 1: 奇数帧</p>
位 15 USBACEPT	0x0	rw		<p>USB 活跃端点(USB active endpoint) 指示在当前配置和接口下，该端点是否活跃。控制器在检测到 USB 复位之后会清除此位(除了端点 0)。应用程序在收到 SetConfiguration 和 SetInterface 命令时必须设置端点寄存器并置起此位。</p> <p>0: 不活跃 1: 活跃</p>
位 14: 11 保留	0x0	resd		保持默认值。
位 10: 0 MPS	0x000	rw		最大包长度(Maximum packet size) 应用程序通过此位设置当前逻辑端点的最大包长度，以字节为单位。

### 20.6.5.13 OTGFS设备IN端点x中断寄存器(OTGFS\_DIEPINTx)(其中x为端点号, x=0…3)

本寄存器指示在发生 USB 及 AHB 相关事件时某个端点的状态，具体请参考图 20-2。当控制器中断寄存器的 IN 端点中断位(OTGFS\_GINTSTS 寄存器的 IEPINT 位)为 1 时，程序必需先读 OTGFS 设备所有端点中断寄存器(OTGFS\_DAINT)来获得发生事件的端点号，然后再读相应端点的中断寄存器获得详细信息。应用程序必需通过清除此位来清除 OTGFS 设备所有端点中断寄存器(OTGFS\_DAINT)和 OTCGFS 中断寄存器 (OTGFS\_GINTST) 的对应位。

域	简称	复位值	类型	功能
位 31: 8 保留		0x000000	resd	保持默认值。

位 7	TXFEMP	0x0	ro	发送 FIFO 空 (Transmit FIFO empty) 当端点的发送 FIFO 为全空或半空时，会生成中断。具体是半空还是全空取决于控制器 AHB 配置寄存器的发送 FIFO 空级别位。
位 6	INEPTNAK	0x0	rw1c	IN 端点 NAK 有效 (IN endpoint NAK effective) 在将对应的 DIEPCTLx.CNAK 设置为 1 之前，需要先写 1 来清除此位。 此中断表示应用程序设置的 IN 端点 NAK 位已经生效。 此中断并不能保证已在 USB 线上发送了 NAK 握手信号。 STALL 位的优先级高于 NAK 位。 此位仅适用于端点已使能的情况。
位 5	保留	0x0	resd	保持默认值。
位 4	INTKNTXFEMP	0x0	rw1c	当发送 FIFO 空时收到 IN 命令 (IN token received when TxFIFO is empty) 此位表示当相关的发送 FIFO(不管是周期性或非周期性)时接收到一个 IN 令牌。在收到 IN 令牌的端点上会生成中断。
位 3	TIMEOUT	0x0	rw1c	超时条件 (Timeout condition) 仅适用于控制 IN 端点，此位表示控制器已经侦测到该端点上最后一个 IN 令牌超时。
位 2	保留	0x0	resd	保持默认值。
位 1	EPTDISD	0x0	rw1c	端点禁用中断 (Endpoint disabled interrupt) 表示已经按照应用程序的请求禁用了端点。
位 0	XFERC	0x0	rw1c	传输完成中断 (Transfer completed interrupt) 表示 AHB 以及 USB 已完成该端点设置的传输任务。

#### 20.6.5.14 OTGFS设备OUT端点x中断寄存器(OTGFS\_DOEPINTx)(其中x为端点号, x=0…3)

此寄存器指示相应端点与 USB 和 AHB 相关的事件状态。具体请参考图 20-2。当控制器中断寄存器的 OUT 端点中断位(OTGFS\_GINTSTS 寄存器的 OEPINT 位)为 1 时，应用程序必需先读设备所有端点中断寄存器(OTGFS\_DAINT)来获得发生事件的端点号，然后再读相应端点的中断寄存器获得详细信息。应用程序必需通过清除此位来清除 OTGFS 设备所有端点中断寄存器(OTGFS\_DAINT)和 OTCGFS 中断寄存器 (OTGFS\_GINTST) 的对应位。

域	简称	复位值	类型	功能
位 31: 7	保留	0x00000001	resd	保持默认值。
位 6	B2BSTUP	0x0	rw1c	收到连续的 SETUP 包 (Back-to-back SETUP packets received) 此位表示接收到不止 3 个连接的 SETUP 包。
位 5	保留	0x0	resd	保持默认值。
位 4	OUTTEPD	0x0	rw1c	端点禁用时接收到 OUT 令牌(OUT token received when endpoint disabled) 仅对控制 OUT 端点有效。 此位表示在端点还没有使能的情况下接收到了 OUT 令牌。在收到 OUT 令牌的端点上会生成中断。
位 3	SETUP	0x0	rw1c	SETUP 阶段完成 (SETUP phase done) 仅适用于控制 OUT 端点，此位表示该控制端点的 SETUP 阶段已完成，当前控制传输不再接收连续的 SETUP 数据包。一旦生成该中断，应用程序就可以解析所收到的 SETUP 数据包。
位 2	保留	0x0	resd	保持默认值。
位 1	EPTDISD	0x0	rw1c	端点禁止中断 (Endpoint disabled interrupt) 表示已经按照应用程序的请求禁用了端点。
位 0	XFERC	0x0	rw1c	传输完成中断 (Transfer completed interrupt) 表示 AHB 以及 USB 已完成该端点设置的传输任务。

#### 20.6.5.15 OTGFS设备IN端点0传输长度寄存器(OTGFS\_DIEPTSIZ0)

应用程序必须在使能端点 0 之前设置该寄存器。一旦通过设备控制端点 0 控制寄存器的端点使能位将端点 0 使能后，就只能通过控制器修改本寄存器。一旦控制器清除了端点使能位，则应用程序只能读此寄存器。

域	简称	复位值	类型	功能
位 31: 21	保留	0x000	resd	保持默认值。
位 20: 19	PKTCNT	0x0	rw	包数目(Packet count) 表示 USB 数据包总数目，其构成端点 0 的数据传输长度。
位 18: 7	保留	0x000	resd	每次从发送 FIFO 读取一个数据包时（不管是最大包长度还是短包），此位域就会自动递减。
位 6: 0	XFERSIZE	0x00	rw	保持默认值。 传输长度 (Transfer size) 表示端点 0 上的传输长度（以字节为单位）。当传输长度变为 0 时，控制器就会中断应用程序。在每个包结束后，传输长度可以设置成该端点的最大包长度。 控制器在每次将外部存储器的一个数据包写入发送 FIFO 时，此位域就会自动递减。

### 20.6.5.16 OTGFS设备OUT端点0传输长度寄存器(OTGFS\_DOEPTSIZ0)

应用程序必须在使能端点 0 之前设置该寄存器。一旦通过设备控制端点 0 控制寄存器的端点使能位将端点 0 使能后，就只能通过控制器修改本寄存器。一旦控制器清除了端点使能位，则应用程序只能读此寄存器。

域	简称	复位值	类型	功能
位 31	保留	0x0	resd	保持默认值。
位 30: 29	SUPCNT	0x0	rw	SETUP 包数目 (SETUP packet count) 表示该端点能够收到的连续 SETUP 数据包数。 01: 1 个数据包； 10: 2 个数据包； 11: 3 个数据包。
位 28: 20	保留	0x000	resd	保持默认值。
位 19	PKTCNT	0	rw	包数目 (Packet count) 将数据包写入接收 FIFO 之后，此位自动递减为 0。
位 18: 7	保留	0x000	resd	保持默认值。 传输长度 (Transfer size) 表示端点 0 上的传输长度，以字节为单位。在传输长度变为 0 后，控制器会中断应用程序。传输长度可以设置成该端点的最大包长度，在每个包结束时中断。 控制器在每次将外部存储器的一个数据包写入发送 FIFO 时，此位域就会自动递减。 控制器在每次从接收 FIFO 读取一个数据包并将其写入外部存储器后，此位域就会自动递减。
位 6: 0	XFERSIZE	0x00	rw	

### 20.6.5.17 OTGFS设备IN端点x传输长度寄存器

(OTGFS\_DIEPTSIZx)( 其中 x 为端点号，x=1…3)

应用程序必须在使能端点 x 之前设置该寄存器。一旦通过设备控制端点 x 控制寄存器的端点使能位将端点 x 使能后，就只能通过控制器修改本寄存器。一旦控制器清除了端点使能位，则应用程序只能读此寄存器。

域	简称	复位值	类型	功能
位 31	保留	0x0	resd	保持默认值。
位 30: 29	MC	0x0	rw	帧内包数目 (Multi count) 对于周期性 IN 端点来说，此位域表示 USB 总线上每个帧必须传输的包数目。控制器通过此位域计算同步 IN 端点发送的数据 PID。 01: 1 个数据包 10: 2 个数据包 11: 3 个数据包
位 28: 19	PKTCNT	0x000	rw	包数目 (Packet count) 表示 USB 数据包总数（端点上的数据传输长度）。 每次从发送 FIFO 读取一个数据包时（最大包长度和短包），此位域就会自动递减。
位 18: 0	XFERSIZE	0x00000	rw	传输长度 (Transfer Size)

此位域包含当前端点的传输字节数。控制器会在此位为 0 时产生中断并通知应用程序。可以配置此寄存器为端点的最大传输长度，并在每个数据包结束产生中断。  
每次将外部存储器的一个数据包写入发送 FIFO 时，控制器就会自动递减此位域。

### 20.6.5.18 OTGFS设备IN端点传输FIFO状态寄存器 (OTGFS\_DTXFSTSx)(其中x为端点号, x=0…3)

只读寄存器，储存的是设备 IN 端点发送 FIFO 的可用空间信息。

域	简称	复位值	类型	功能
位 31: 16	保留	0x0000	resd	保持默认值。
位 15: 0	INEPTXFSAV	0x0200	ro	IN 端点发送 FIFO 剩余空间 (IN endpoint TxFIFO space avail) 表示端点发送 FIFO 的可用空间。以 32 位的字为单位。 0x0: 发送 FIFO 满; 0x1: 剩余 1 个字; 0x2: 剩余 2 个字; 0xn: 剩余 n 个字(其中 0 < n < 512); 0x200: 剩余 512 个字; 其他: 保留。

### 20.6.5.19 OTGFS设备OUT端点x传输长度寄存器 (OTGFS\_DOEPTSIZx)( 其中x 为端点号, x=1…3)

应用程序必须在使能端点 x 之前设置该寄存器。一旦通过设备控制端点 x 控制寄存器的端点使能位将端点 x 使能后，就只能通过控制器修改本寄存器。一旦控制器清除了端点使能位，则应用程序只能读此寄存器。

域	简称	复位值	类型	功能
位 31	保留	0x0	resd	保持默认值。
位 30: 29	RXDPID	0x0	ro	收到的数据 PID (Received data PID) 此位仅对同步 OUT 端点有效。表示收到的最后一个数据包的数据 PID: 00: DATA0; 01: DATA2; 10: DATA1; 11: MDATA。
位 28: 19	PKTCNT	0x000	rw	SETUP 包数目 (SETUP packet count) 此位仅对控制 OUT 端点有效。表示端点收到的连续的 SETUP 包的数量: 01: 1 个包; 10: 2 个包; 11: 3 个包。
位 18: 0	XFERSIZE	0x00000	rw	包数目 (Packet count) 指示该端点上传输的 USB 包数目。 每次向接收 FIFO 写入一个数据包时（最大包长度和短包），此位域就会自动递减。

## 20.6.6 供电和时钟控制寄存器

### 20.6.6.1 OTGFS电源和时钟门控寄存器(OTGFS\_PCGCCTL)

此寄存器既适用于主机模式也适用于设备模式。

域	简称	复位值	类型	功能
位 31: 5	保留	0x0000000	resd	保持默认值。
位 4	SUSPENDM	0x0	ro	物理层挂起 (PHY suspend)

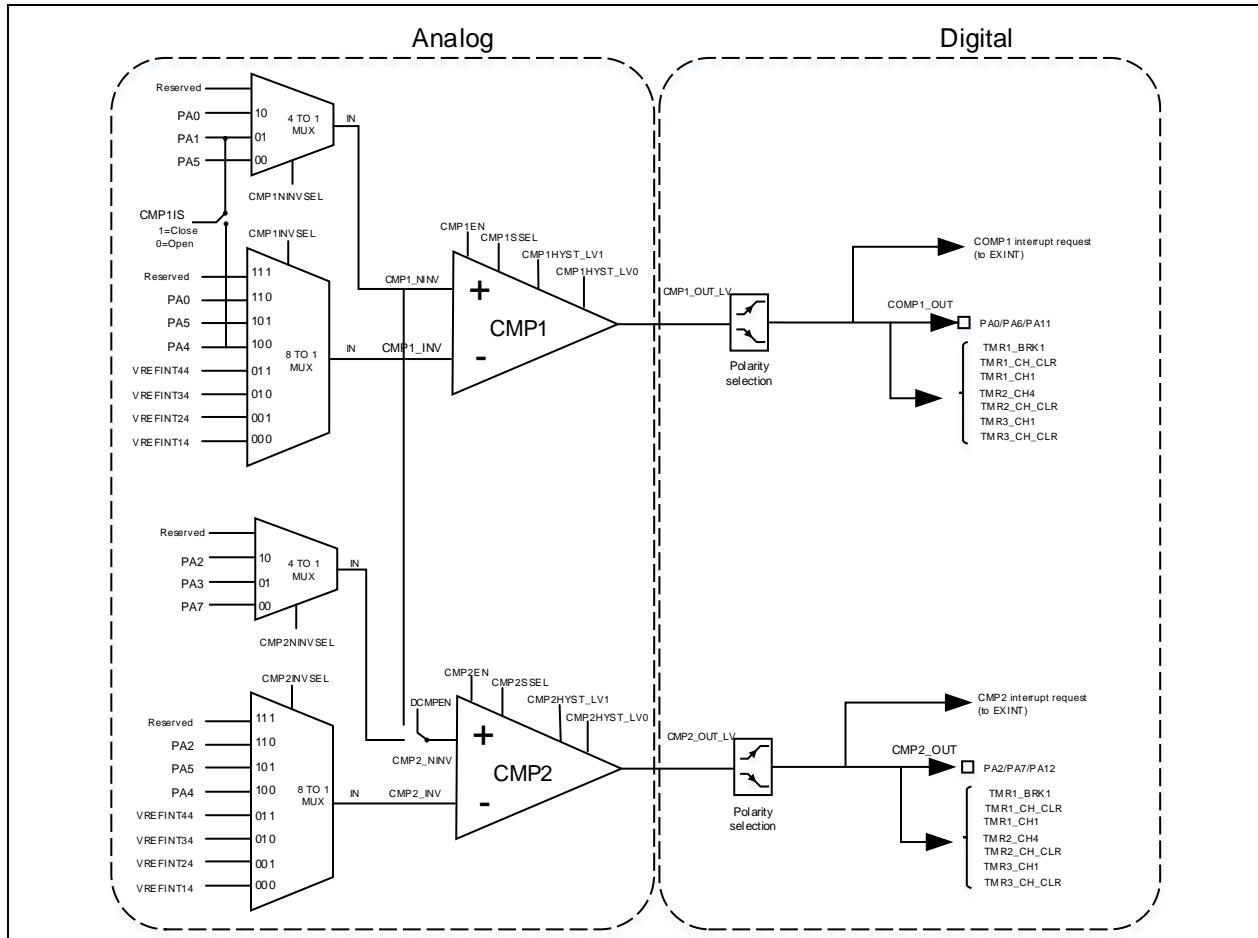
				指示 PHY 处于挂起状态
位 3: 1	保留	0x0	resd	保持默认值。
位 0	STOPPCLK	0x0	rw	停止 PHY 时钟 (Stop PHY clock) 当 USB 挂起或会话无效或者设备断开时，应用程序通过此位停止 PHY 时钟。当 USB 恢复或者有新的会话请求时，应用程序会清除此位。

# 21 比较器(CMP)

## 21.1 简介

AT32WB415 内置两个超低功耗比较器 CMP1 和 CMP2，可以用于多种功能，包括：外部模拟信号的监测控制及从低功耗模式唤醒，与内置定时器结合使用，进行脉冲宽度测量和 PWM 信号控制等。

图 21-1 比较器的框图



## 21.2 主要特性

- 比较器迟滞程度可配
- 比较器输出极性可配
- 比较器输出速度可配
- 比较器同相和反相输入源可选：
  - I/O引脚
  - 内部参考电压和三个系数分压值(1/4, 1/2, 3/4)
- 支持输出重定向功能：
  - 普通I/O
  - 定时器断路输入TMRx\_BRK
  - 定时器输入捕获TMR\_CH
  - 定时器输出比较参考值清零TMR\_CH\_CLR
- 比较器 1 和比较器 2 组合成窗口比较器
- 结合 EXINT 产生中断，从低功耗模式唤醒

## 21.3 中断管理

模拟比较器 1 的输出经过极性选择，输入至 EXINT 第 19 号中断线，产生外部中断或者事件，可以用于

系统从低功耗模式中唤醒。

模拟比较器 2 的输出经过极性选择，输入至 EXINT 第 20 号中断线，产生外部中断或者事件，可以用于系统从低功耗模式中唤醒。

更多详细信息，请参见中断和事件部分。

## 21.4 设计提示

比较器有以下提示仅供设计参考。

- 输入输出配置

I/Os 作为比较器输入时，必须配置为模拟模式。比较器输出可以通过配置寄存器 IOMUX\_REMAP2 的 CMP\_MUX[1: 0] 位重映射到外部 I/O 口。

比较器输出配置

复用配置	CMP_MUX[1: 0]=00	CMP_MUX[1: 0]=01	CMP_MUX[1: 0]=10
CMP1_OUT	PA0	PA6	PA11
CMP2_OUT	PA2	PA7	PA12

- 锁定功能

寄存器 CMP\_CTRLSTS1 具有写保护功能，一旦编程完成，对 CMPxWP 位设置为 1，则寄存器 CMP\_CTRLSTS1 和寄存器 CMP\_CTRLSTS2 的对应位变为只读，包括 CMPxWP 位，只能通过系统复位解除写保护功能，该功能可用于具有特定功能安全要求的应用。

- 低功耗模式配置

比较器时钟源为 PCLK，复位信号采用系统复位。在深度睡眠模式下，比较器依然可以工作，用于 EXINT 的中断源，控制系统从低功耗模式中唤醒。

## 21.5 功能描述

### 21.5.1 模拟比较器

#### 同相反相输入选择

寄存器 CMP\_CTRLSTS1 的 CMPxINVSEL[1: 0] 位控制比较器的同相输入信号来源于 I/O 口，CMPxINVSEL[2: 0] 位控制比较器的反相输入信号来源于内部参考电压和三个系数分压值或者 I/O 口。

#### 迟滞功能

寄存器 CMP\_CTRLSTS1 的 CMPxHYST[1: 0] 控制比较器迟滞输出，该功能可避开噪声信号带来的虚假传输信号，如果不需要迟滞，可以关闭掉。

#### 操作模式配置

寄存器 CMP\_CTRLSTS1 的 CMPxSSEL 位用于控制比较器工作在快速/最大功耗、低速/最低功耗，用于实现比较器在性能与功耗之间的折中。

## 21.6 CMP 寄存器

必须以字（32 位）的方式操作这些外设寄存器。

表 21-1 CMP 寄存器映像和复位值

寄存器简称	基址偏移量	复位值
CMP_CTRLSTS1	0x00	0x0000 0080
CMP_CTRLSTS2	0x04	0x0001 0001

## 21.6.1 比较器控制和状态寄存器1 (CMP\_CTRLSTS1)

域	简称	复位值	类型	功能
位 31	CMP2WP	0x0	rw0c	比较器 2 写保护(Comparator 2 write protect) 0: 关闭; 1: 开启。 注: 用于使能比较器控制寄存器 COMP_CTRLSTS1[31:16]位和 COMP_CTRLSTS2[31:16]位的写保护, 该位只能由系统 复位清零。
位 30	CMP2VALUE	0x0	ro	比较器 2 输出值(Comparator 2 output value) 此位为只读。它反映当前比较器 2 输出的状态 (受到 CMP2P 位的影响)。
位 29:28	CMP2HYST	0x0	rw	比较器 2 迟滞(Comparator2 hysteresis) 00: 没有迟滞 01: 低度迟滞 10: 中度迟滞 11: 高度迟滞 参见迟滞程度的电气特性
位 27	CMP2P	0x00	rw	比较器 2 极性选择位(Comparator2 polarity) 0: 比较器 2 输出值不反相 1: 比较器 2 输出值反相
位 26:24	CMP2TAG	0x0	rw	比较器 2 输出选择位(Comparator output target) 这些位控制比较器 2 的输出目的地 000: 无选择 001: 定时器 1 刹车输入 010: 定时器 1 输入捕获 1 011: 定时器 1 输出比较清除 100: 定时器 2 输入捕获 4 101: 定时器 2 输出比较清除 110: 定时器 3 输入捕获 1 111: 定时器 3 输出比较清除
位 23	DCMPEN	0x0	rw	双比较器模式使能(Double comparator mode enable) 0: 关闭 1: 使能 注: 该位用于使能双比较器模式, 可让 COMP2 和 COMP1 的同相输入端相连。
位 22:20	CMP2INVSEL	0x0	rw	比较器 2 的反相端输入选择(Comparator2 inverting selection) 000: 1/4 VREFINT 001: 1/2 VREFINT 010: 3/4 VREFINT 011: VREFINT 100: PA4 101: PA5 110: PA2 111: Reserved
位 19	保留	0x0	resd	请保持默认值。
位 18	CMP2SSEL	0x0	rw	比较器 2 速度选择(Comparator2 speed selection) 这些位用于控制比较器的操作模式, 允许调整速率和功耗 0: 高速/最大功耗 1: 低速/最低功耗
位 17	保留	0x0	resd	请保持默认值。
位 16	CMP2EN	0x0	rw	比较器 2 使能(Comparator2 enable) 该位控制打开或关闭比较器 0: 关闭比较器 1: 开启比较器
位 15	CMP1WP	0x0	rw0c	比较器 1 写保护(Comparator 1 write protect) 0: 关闭; 1: 开启。

				注：用于使能比较器控制寄存器 COMP_CTRLSTS1[15:0]位和 COMP_CTRLSTS2[15:0] 位的写保护，该位只能由系统复位清零。
位 14	CMP1VALUE	0x0	ro	比较器 1 输出值(Comparator 1 output value) 此位为只读。它反映当前比较器 1 输出的状态（受到 CMP1P 位的影响）。
位 13:12	CMP1HYST	0x0	rw	比较器 1 迟滞(Comparator1 hysteresis) 00: 没有迟滞 01: 低度迟滞 10: 中度迟滞 11: 高度迟滞 参见迟滞程度的电气特性
位 11	CMP1P	0x0	rw	比较器 1 极性选择位(Comparator1 polarity) 0: 比较器 1 输出值不反相 1: 比较器 1 输出值反相
位 10:8	CMP1TAG	0x0	rw	比较器 2 输出选择位(Comparator output target) 这些位控制比较器 2 的输出目的地 000: 无选择 001: 定时器 1 刹车输入 010: 定时器 1 输入捕获 1 011: 定时器 1 输出比较清除 100: 定时器 2 输入捕获 4 101: 定时器 2 输出比较清除 110: 定时器 3 输入捕获 1 111: 定时器 3 输出比较清除
位 7	保留	0x0	resd	请保持默认值。
位 6:4	CMP1INVSEL	0x0	rw	比较器 1 的反相端输入选择(Comparator1 inverting selection) 000: 1/4 VREFINT 001: 1/2 VREFINT 010: 3/4 VREFINT 011: VREFINT 100: PA4 101: PA5 110: PA0 111: Reserved
位 3	保留	0x0	resd	请保持默认值。
位 2	CMP1SSEL	0x0	rw	比较器 1 速度选择(Comparator1 speed selection) 这些位用于控制比较器的操作模式，允许调整速率和功耗 0: 高速/最大功耗 1: 低速/最低功耗
位 1	CMP1IS	0x0	rw	比较器 1 输入切换( Comparator1 input shift) 0: 开关断开 1: 开关闭合 注：该位用于开关比较器 1 同相输入端 PA1 与 PA4 之 间的连接。此开关仅用于重定向信号到高阻输入，例如比 较器 1 的同相输入(高阻开关)
位 0	CMP1EN	0x0	rw	比较器 1 使能(Comparator1 enable) 该位控制打开或关闭比较器 0: 关闭比较器 1: 开启比较器

## 21.6.2 比较器控制和状态寄存器2(CMP\_CTRLSTS2)

域	简称	复位值	类型	功能
位 31: 18	保留	0x0000	resd	请保持默认值。
位 17: 16	COMP2NINVSEL	0x1	rw	比较器 2 同相输入选择(Comparator2 non-inverting input selection) 00: PA7 01: PA3(默认输入) 10: PA2 11: 保留 注: 当 CMP2WP 为 1 时, 这些位为只读。
位 15: 2	保留	0x0000	resd	请保持默认值。
位 1: 0	COMP1NINVSEL	0x1	rw	比较器 1 同相输入选择(Comparator1 non-inverting input selection) 00: PA5 01: PA1(默认输入) 10: PA0 11: 保留 注: 当 CMP1WP 为 1 时, 这些位为只读。

## 22 调试 DEBUG

### 22.1 简介

Cortex™-M4 内核具有丰富的调试特性。除了支持暂停和单步等标准的调试特性外，还可以利用跟踪特性查看程序执行的细节。Cortex™-M4 内核的调试可以通过两种接口实现：串行调试接口。跟踪信息可以由单线的串行线查看接口收集，或者若需要的跟踪带宽较大时，也可以使用 TRACE 接口。跟踪和调试接口可以合并到一个接口中。

ARM Cortex™-M4 内核相关资料，可参考：

- Cortex™-M4 技术参考手册 (TRM)
- ARM 调试接口 V5
- ARM CoreSight 开发工具集(r1p0 版)技术参考手册

### 22.2 调试与跟踪功能

支持不同外设的调试，还可以设置调试时外设的工作状态。对于定时器和看门狗用户可以选择在调试时是否停止或继续计数；对于 CAN，用户可以选择在调试期间是否停止或继续更新接收寄存器；对于 I2C，用户可以选择在调试期间是否停止或继续 SMBUS 超时计数。

另外支持在低功耗模式下调试代码。在睡眠模式下，HCLK 与 FCLK 保持代码配置的时钟继续工作。在深度睡眠模式下，HICK 振荡器将开启并为 FCLK 和 HCLK 提供时钟。

MCU 内部有多个 ID 编码，调试器可通过地址为 0xE0042000 的 DEBUG\_IDCODE 来访问。它是 DEBUG 的一个组成部分，并且映射到外部 PPB 总线上。使用 SW 调试口或通过用户代码都可以访问此编码。即使当 MCU 处于系统复位状态下这个编码也可以被访问。

支持两种跟踪接口模式：串行线查看的单针模式和多针跟踪接口。

### 22.3 I/O 控制

AT32WB415 有的封装里都支持 SWJ-DP 调试，该调试共使用 5 个普通 I/O 口。复位以后，SWJ-DP 作为默认功能可立即供调试器使用。

当用户切换调试接口或不使用调试功能时，可配置 IOMUX\_MAPR/IOMUX\_MAPR7 寄存器来释放这些专用 I/O 口。用户释放相应的调试 I/O 后，GPIO 控制器将取得控制，这些 I/O 口可作为普通的 I/O 口使用。

当用户需要使用跟踪功能时，可以通过设置 DEBUG 控制寄存器 (DEBUG\_CTRL) 的 TRACE\_IOEN 和 TRACE\_MODE 位来使能跟踪功能以及选择跟踪模式。

表 22-1 跟踪功能使能

TRACE_IOEN	功能说明
0	无跟踪（默认状态）
1	开启跟踪功能

表 22-2 跟踪功能模式

TRACE_MODE[1: 0]	PB3/JTDO/TR ACESWO	PE2/TRAC ECK	PE3/TRAC ED[0]	PE4/TRAC ED[1]	PE5/TRACE D[2]	PE6/TRAC ED[3]
00 异步跟踪	TRACES WO	释放（可用作普通 I/O 口）				
01 同步跟踪		TRAC ECK	TRAC ED[0]		释放 (可用作普通 I/O 口)	
10 同步跟踪	释放（可用作普通 I/O 口）	TRAC ECK	TRAC ED[0]	TRAC ED[1]	释放（可用作普通 I/O 口）	
11 同步跟踪		TRACE CK	TRACE D[0]	TRACE D[1]	TRACE D[2]	TRACE D[3]

## 22.4 DEBUG寄存器

下面列出了 DEBUG 寄存器映象和复位数值。

必须以字（32 位）的方式操作这些外设寄存器。

表 22-3 DEBUG寄存器地址和复位值

寄存器简称	基址址	复位值
DEBUG_IDCODE	0xE004 2000	0xFFFF XXXX
DEBUG_CTRL	0xE004 2004	0x0000 0000

### 22.4.1 DEBUG设备ID (DEBUG\_IDCODE)

MCU 集成了 ID code，通过 ID 可以识别 MCU 的版本编号。DEBUG\_IDCODE 寄存器被映射到外部 PPB 总线，基址为 0xE0042000。使用 SW 调试口或用户代码都可以访问此编号。

域	简称	复位值	类型	功能
位 31: 0	PID	0xFFFF XXXX ro		PID 信息
PID [31: 0]	AT32 型号	FLASH 大小	封装	
0x7003_0250	AT32WB415CCU7-7	256KB	QFN48_7x7	

### 22.4.2 DEBUG控制寄存器 (DEBUG\_CTRL)

DEBUG 控制寄存器 (DEBUG\_CTRL) 被映射到外部 PPB 总线，基址为 0xE0042000。寄存器由 PORESET 异步复位（不被系统复位所复位）。当内核处于复位状态下时，调试器可写。

域	简称	复位值	类型	功能
位 31	保留	0x0	resd	保持默认值。
位 30	TMR11_PAUSE	0x0	rw	TMR11 暂停控制位 0: 定时器正常工作； 1: 定时器停止工作。
位 29	TMR10_PAUSE	0x0	rw	TMR10 暂停控制位 0: 定时器正常工作； 1: 定时器停止工作。
位 28	TMR9_PAUSE	0x0	rw	TMR9 暂停控制位 0: 定时器正常工作； 1: 定时器停止工作。
位 27:19	保留	0x000	resd	保持默认值。
位 18	TMR5_PAUSE	0x0	rw	TMR5 暂停控制位 0: 定时器正常工作； 1: 定时器停止工作。
位 17	保留	0x0	resd	保持默认值。
位 16	保留	0x0	resd	保持默认值。
位 15	I2C1_SMBUS_TIMEOUT_UT	0x0	rw	I2C1 暂停控制位。 0: 正常工作； 1: I2C1 SMBUS 的超时控制停止工作。
位 14	CAN1_PAUSE	0x0	rw	CAN1 暂停控制位。 0: CAN1 正常运行； 1: CAN1 的接收寄存器不继续接收数据。
位 13	TMR4_PAUSE	0x0	rw	TMR4 暂停控制位。 0: 定时器正常工作； 1: 定时器停止工作。
位 12	保留	0x0	resd	必须保持为 0。
位 11	TMR2_PAUSE	0x0	rw	TMR2 暂停控制位。 0: 定时器正常工作； 1: 定时器停止工作。
位 10	TMR1_PAUSE	0x0	rw	TMR1 暂停控制位。 0: 定时器正常工作； 1: 定时器停止工作。

位 9	WWDT_PAUSE	0x0	rw	窗口看门狗暂停控制位。 0: 窗口看门狗正常工作; 1: 窗口看门狗停止工作。
位 8	WDT_PAUSE	0x0	rw	看门狗暂停控制位 0: 看门狗正常工作; 1: 看门狗停止工作。
位 7: 6	TRACE_MODE	0x0	rw	跟踪引脚分配控制 00: 异步模式; 01: 同步模式, 数据长度为 1; 10: 同步模式, 数据长度为 2; 11: 同步模式, 数据长度为 4。
位 5	TRACE_IOEN	0x0	rw	跟踪引脚分配使能 0: 无跟踪功能(默认状态); 1: 开启跟踪功能。
位 4: 3	保留	0x0	rwd	必须保持为 0。
位 2	STANDBY_DEBUG	0x0	rw	待机模式调试控制位。 0: 进入待机模式时, 整个 1.2V 数字电路部分都断电; 1: 进入待机模式时, 整个 1.2V 数字电路部分不断电, 系统时钟由内部 RC 振荡器(HICK)提供时钟。
位 1	DEEPSLEEP_DEBUG	0x0	rw	深度睡眠模式调试控制位。 0: 进入深度睡眠模式时, 关闭所有 1.2V 域的时钟, 退出深度睡眠模式时, 系统时钟选择开启内部 RC 振荡器(HICK), 系统时钟选择 HICK 作为系统时钟源, 软件需根据应用需求重新配置系统时钟; 1: 进入深度睡眠模式时, 系统时钟由内部 RC 振荡器(HICK)提供。退出深度睡眠模式时, 系统时钟选择 HICK 作为系统时钟源, 软件需根据应用需求重新配置系统时钟。
位 0	SLEEP_DEBUG	0x0	rw	睡眠模式调试控制位 0: 进入睡眠模式时, CPU HCLK 时钟关闭, 其他时钟均继续运行, 退出睡眠模式时, 不需要重新配置时钟系统; 1: 进入睡眠模式时, 所有时钟都继续运行。

## 23 版本历史

日期	版本	变更
2022.04.13	2.00	新版本发布
2022.06.27	2.01	1、修订图 19-7 框图错误 2、修订 19.6.7 章节描述 3、修订 19.7 章节描述

**重要通知 - 请仔细阅读**

买方自行负责对本文所述雅特力产品和服务的选择和使用，雅特力概不承担与选择或使用本文所述雅特力产品和服务相关的任何责任。

无论之前是否有过任何形式的表示，本文档不以任何方式对任何知识产权进行任何明示或默示的授权或许可。如果本文档任何部分涉及任何第三方产品或服务，不应被视为雅特力授权使用此类第三方产品或服务，或许可其中的任何知识产权，或者被视为涉及以任何方式使用任何此类第三方产品或服务或其中任何知识产权的保证。

除非在雅特力的销售条款中另有说明，否则，雅特力对雅特力产品的使用和 / 或销售不做任何明示或默示的保证，包括但不限于有关适销性、适合特定用途（及其依据任何司法管辖区的法律的对应情况），或侵犯任何专利、版权或其他知识产权的默示保证。

雅特力产品并非设计或专门用于下列用途的产品：(A) 对安全性有特别要求的应用，例如：生命支持、主动植入设备或对产品功能安全有要求的系统；(B) 航空应用；(C) 汽车应用或汽车环境；(D) 航天应用或航天环境，且/或(E) 武器。如果雅特力产品不是为前述应用设计的，而采购商擅自将其用于前述应用，即使采购商向雅特力发出了书面通知，采购商仍将独自承担因此而导致的任何风险，雅特力的产品设计规格明确指定的汽车、汽车安全或医疗工业领域专用产品除外。根据相关政府主管部门的规定，ESCC、QML 或 JAN 正式认证产品适用于航天应用。

经销的雅特力产品如有不同于本文档中提出的声明和 / 或技术特点的规定，将立即导致雅特力针对本文所述雅特力产品或服务授予的任何保证失效，并且不应以任何形式造成或扩大雅特力的任何责任。

© 2019 雅特力科技 (重庆) 有限公司 保留所有权利