基于ARM®32位的Cortex™-M4F微控制器+FPU，带256K字节至1024K字节内部闪存、sLib、USB、以太网、2个CAN、17个定时器、3个ADC、21个通信接口

功能

- **内核：带有FPU的ARM®32位的Cortex™-M4F CPU**
  - 最高240 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
  - 内建浮点运算(FPU)，具有DSP指令集
- **存储器**
  - 从256 K字节至1024 K字节的内部闪存程序/数据存储器
  - 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
  - SPIM接口：额外提供高达16 M字节外部SPI闪存程序数据存储器接口
  - 高达96+128 K字节的SRAM
  - 带2个片选外部存储器控制器(XMC)，支持复用信号的SRAM/NOR/PSRAM和NAND存储器
  - 并行LCD接口，兼容8080/6800模式
- **时钟、复位和电源管理**
  - 2.6至3.6伏供电和I/O引脚
  - 上电/断电复位(POR/PDR)、可编程电压监测器(PVD)
  - 4至25 MHz晶体振荡器
  - 内嵌经出厂调校的8 MHz的RC振荡器(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）
  - 内嵌带校准的40 kHz的RC振荡器
  - 带校准功能为RTC的32 kHz振荡器
- **低功耗**
  - 睡眠、停机和待机模式
  - V_{BAT}为RTC和42个16位后备寄存器供电
- **3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
  - 三组采样和保持功能
  - 温度传感器
- **2个12位D/A转换器**
- **DMA：14通道DMA控制器**
  - 支持的外设：定时器、ADC、SDIO、I²S、SPI、I²C和USART
- **调试模式**
  - 串行单线调试(SWD)和JTAG接口
  - Cortex™-M4F内嵌跟踪模块(ETM™)
- **多达80个快速I/O端口**
  - 37/51/80个多功能双向的I/O口，所有I/O口可以映像到16个外部中断；几乎所有端口均可容忍5V输入信号
  - 所有I/O口均为快速I/O，寄存器访问速度最高f_{AHB}
- **多达17个定时器**
  - 多达8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或脉冲计数的通道和增量编码器输入
  - 多达2个16位带死区控制和紧急刹车，用于电机控制的PWM高级控制定时器
  - 2个看门狗定时器(独立的和窗口型的)
  - 系统时间定时器：24位自减型计数器
  - 2个16位基本定时器用于驱动DAC
- **多达21个通信接口**
  - 多达3个I²C接口(支持SMBus/PMBus)
  - 多达8个USART接口(支持ISO7816，LIN，IrDA接口和调制解调控制)
  - 多达4个SPI接口(50 M位/秒)，4个均可复用为I²S接口，其中I2S2/I2S3支持全双工
  - 多达2个CAN接口(2.0B主动)
  - USB2.0全速设备接口，支持无晶振(crystal-less)
  - 多达2个SDIO接口
  - 10/100M以太网MAC：具有专用DMA和4 K字节SRAM，支持IEEE 1588硬件，MII/RMII接口
- **CRC计算单元，96位的芯片唯一代码**
- **封装**
  - LQFP100 14 x 14 mm
  - LQFP64 10 x 10 mm
  - LQFP48 7 x 7 mm
  - QFN48 6 x 6 mm
- **选型列表**

内部闪存存储器	型号
1024 K字节	AT32F403ACGU7, AT32F403ACGT7, AT32F403ARGT7, AT32F403AVGT7, AT32F407RGT7, AT32F407VGT7
512 K字节	AT32F403ACEU7, AT32F403ACET7, AT32F403ARET7, AT32F403AVET7, AT32F407RET7, AT32F407VET7
256 K字节	AT32F403ACCU7, AT32F403ACCT7, AT32F403ARCT7, AT32F403AVCT7, AT32F407RCT7, AT32F407VCT7

目 录

1 系统架构 46
1.1 系统概述 46
1.1.1 总线架构 51
1.1.2 ARM Cortex™-M4F处理器 51
1.2 地址映射 52
1.2.1 寄存器映像 53
1.2.2 位绑定 55
1.2.3 片上SRAM 56
1.2.4 片上Flash 56
1.3 引导配置 59
1.4 器件特征信息 60
1.4.1 寄存器的缩写说明 60
1.4.2 闪存容量寄存器 60
1.4.3 器件电子签名 60
2 电源控制（PWR） 62
2.1 简介 62
2.2 主要特点 62
2.3 功能描述 62
2.3.1 电源域 62
2.3.1.1 VDD/VDDA电源域 63
2.3.1.2 内核电源域 64
2.3.2 低功耗模式 65
2.3.2.1 睡眠模式 66
2.3.2.2 停止模式 67
2.3.2.3 待机模式 68
2.3.2.4 调试模式 68
2.3.3 自动唤醒 69
2.4 PWR寄存器 69

2.4.1 电源控制寄存器（PWR_CTRL）..... 70
2.4.2 电源控制/状态寄存器（PWR_CTRLSTS）..... 71
3 复位和时钟控制（RCC）..... 72
3.1 复位..... 72
3.1.1 系统复位..... 72
3.1.2 电源复位..... 72
3.1.3 备份域复位..... 73
3.2 时钟..... 73
3.2.1 HSE时钟..... 76
3.2.2 HSI时钟..... 76
3.2.3 PLL..... 77
3.2.4 LSE时钟..... 77
3.2.5 LSI时钟..... 77
3.2.6 系统时钟（SYSCLK）选择..... 77
3.2.7 时钟失效检测（CFD）..... 78
3.2.8 RTC时钟..... 78
3.2.9 看门狗时钟..... 78
3.2.10 时钟输出..... 78
3.3 RCC寄存器描述..... 80
3.3.1 时钟控制寄存器（RCC_CTRL）..... 82
3.3.2 时钟配置寄存器（RCC_CFG）..... 83
3.3.3 时钟中断寄存器 （RCC_CLKINT）..... 85
3.3.4 APB2外设复位寄存器（RCC_APB2RST）..... 87
3.3.5 APB1外设复位寄存器（RCC_APB1RST）..... 89
3.3.6 AHB外设时钟使能寄存器（RCC_AHBEN）..... 92
3.3.7 APB2外设时钟使能寄存器（RCC_APB2EN）..... 94
3.3.8 APB1外设时钟使能寄存器（RCC_APB1EN）..... 96

3.3.9 备份域控制寄存器（RCC_BDC）..... 99
3.3.10 控制/状态寄存器（RCC_CTRLSTS）..... 100
3.3.11 AHB外设复位寄存器（RCC_AHBRST）..... 101
3.3.12 额外寄存器（RCC_MISC）..... 101
3.3.13 测试寄存器（RCC_TEST）..... 102
3.3.14 额外寄存器（RCC_MISC2）..... 103
3.3.15 额外寄存器（RCC_INTCTRL）..... 104
4 备份寄存器（BKPR）..... 106
4.1 BKPR简介..... 106
4.2 BKPR特性..... 106
4.3 BKPR功能描述..... 106
4.3.1 侵入检测..... 106
4.3.2 RTC校准..... 106
4.4 BKPR寄存器描述..... 106
4.4.1 备份数据寄存器x（BKPR_DRx）（x = 1 … 42）..... 109
4.4.2 RTC时钟校准寄存器（BKPR_RTCCAL）..... 109
4.4.3 备份控制寄存器（BKPR_CTRL）..... 110
4.4.4 备份控制/状态寄存器（BKPR_CTRLSTS）..... 110
5 闪存控制器（FMC）..... 112
5.1 FMC简介..... 112
5.2 主要特点..... 112
5.2.1 闪存模块组织..... 112
5.2.2 外部闪存模块组织..... 115
5.3 功能描述..... 116
5.3.1 读操作..... 116
5.3.1.1 取指令..... 116
5.3.1.2 D-Code接口..... 117
5.3.1.3 闪存访问控制器..... 117

5.3.2 闪存编程和擦除控制器（FPEC） 117
5.3.2.1 键值 118
5.3.2.2 解除闪存锁 118
5.3.2.3 主闪存编程 118
5.3.2.4 闪存擦除 119
5.3.2.5 选择字节编程 120
5.3.3 保护 121
5.3.3.1 写保护 121
5.3.3.2 读保护 121
5.3.3.3 选择字节块写保护 122
5.3.4 选择字节说明 122
5.3.5特殊功能 123
5.3.5.1 安全库区设定 123
5.4 FMC寄存器 125
5.4.1 闪存访问控制寄存器（FLASH_ACR） 127
5.4.2 FPEC键寄存器（FLASH_FCKEY） 127
5.4.3 闪存OPTKEY寄存器（FLASH_OPTKEYR） 128
5.4.4 闪存状态寄存器（FLASH_STS） 128
5.4.5 闪存控制寄存器（FLASH_CTRL） 129
5.4.6 闪存地址寄存器（FLASH_ADDR） 130
5.4.7 选择字节寄存器（FLASH_UOB） 130
5.4.8 写保护寄存器（FLASH_WRPRT） 131
5.4.9 FPEC键寄存器2（FLASH_FCKEY2） 131
5.4.10 闪存状态寄存器2（FLASH_STS2） 131
5.4.11 闪存控制寄存器2（FLASH_CTRL2） 132
5.4.12 闪存地址寄存器2（FLASH_ADDR2） 133
5.4.13 FPEC键寄存器3（FLASH_FCKEY3） 133
5.4.14 闪存选择寄存器（FLASH_SELECT） 134
5.4.15 闪存状态寄存器3（FLASH_STS3） 134

5.4.16 闪存控制寄存器3（FLASH_CTRL3） 135
5.4.17 闪存地址寄存器3（FLASH_ADDR3） 135
5.4.18 闪存解密地址寄存器（FLASH_DA） 136
5.4.19 闪存安全库区状态寄存器0（SLIB_CDR0） 136
5.4.20 闪存安全库区状态寄存器1（SLIB_CDR1） 137
5.4.21 闪存安全库区密码寄存器（SLIB_PSW） 137
5.4.22 闪存安全库区密码设定状态寄存器（SLIB_PSW_STS） 138
5.4.23 闪存安全库区密码设定寄存器（SLIB_SET_PSW） 138
5.4.24 闪存安全库区地址设定寄存器（SLIB_SET_RANGE） 139
5.4.25 闪存安全库区键值设定寄存器（SLIB_KEYR） 139
5.4.26 闪存CRC校验控制寄存器（CRC_DR） 140
5.4.27 闪存CRC校验结果寄存器（CRC_OUTR） 140
6 CRC计算单元（CRC） 142
6.1 CRC简介 142
6.2 CRC主要特性 142
6.3 CRC功能描述 142
6.4 CRC寄存器 143
6.4.1 数据寄存器（CRC_DR） 143
6.4.2 独立数据寄存器（CRC_IDR） 143
6.4.3 控制寄存器（CRC_CTRL） 144
7 通用和复用功能I/O（GPIO和AFIO） 145
7.1 简介 145
7.2 主要特征 145
7.3 功能描述 145
7.3.1 GPIO引脚配置 145
7.3.2 外部中断/唤醒线 147
7.3.3 输入配置 147

7.3.4 模拟输入配置 148
7.3.5 输出配置 149
7.3.6 GPIO锁定机制 150
7.3.7 复用功能（AF） 150
7.4 IO映射功能配置 155
7.4.1 把OSC32_IN/OSC32_OUT作为GPIO端口PC14/PC15 155
7.4.2 把OSC_IN/OSC_OUT引脚作为GPIO端口PD0/PD1 155
7.4.3 CAN复用功能重映射 155
7.4.4 JTAG/SWD复用功能重映射 155
7.4.5 ADC复用功能重映射 156
7.4.6 定时器复用功能重映射 156
7.4.7 USART/UART复用功能重映射 158
7.4.8 I2C复用功能重映射 159
7.4.9 SPI/I2S复用功能重映射 159
7.4.10 SDIO复用功能重映射 160
7.4.11 外部SPIF复用功能重映射 161
7.4.12 Ethernet复用功能重映射 161
7.4.13 XMC复用功能重映射 161
7.5 GPIO与AFIO寄存器 162
7.5.1 端口配置低寄存器（GPIOx_CTRLL）（x=A..E） 164
7.5.2 端口配置高寄存器（GPIOx_CTRLH）（A..E） 165
7.5.3 端口输入数据寄存器（GPIOx_IPTDT）（x=A..E） 165
7.5.4 端口输出数据寄存器（GPIOx_OPTDT）（x=A..E） 166
7.5.5 端口位设置/清除寄存器（GPIOx_BSRE）（x=A..E） 166
7.5.6 端口位清除寄存器（IOx_BRE）（x=A..E） 167
7.5.7 端口配置锁定寄存器（GPIOx_LOCK）（x=A..E） 167
7.5.8 电压转换速率切换控制寄存器（GPIOx_SRCTR）（x=A..E） 168

7.5.9 极大电流推动/吸入能力切换控制寄存器（GPIOx_HDRV）（x=A..E）..... 168
7.5.10 复用事件控制寄存器（AFIO_EVCTRL）..... 168
7.5.11 复用重映射和调试I/O配置寄存器（AFIO_MAP）..... 170
7.5.12 复用外部中断配置寄存器1（AFIO_EXTIC1）..... 173
7.5.13 复用外部中断配置寄存器2（AFIO_EXTIC2）..... 173
7.5.14 复用外部中断配置寄存器3（AFIO_EXTIC3）..... 173
7.5.15 复用外部中断配置寄存器4（AFIO_EXTIC4）..... 174
7.5.16 复用重映射和调试I/O配置寄存器2（AFIO_MAP2）..... 174
7.5.17 复用重映射和调试I/O配置寄存器3（AFIO_MAP3）..... 175
7.5.18 复用重映射和调试I/O配置寄存器4（AFIO_MAP4）..... 175
7.5.19 复用重映射和调试I/O配置寄存器5（AFIO_MAP5）..... 176
7.5.20 复用重映射和调试I/O配置寄存器6（AFIO_MAP6）..... 177
7.5.21 复用重映射和调试I/O配置寄存器7（AFIO_MAP7）..... 178
7.5.22 复用重映射和调试I/O配置寄存器8（AFIO_MAP8）..... 180
8 中断和事件 182
8.1 嵌套向量中断控制器 182
8.1.1 系统嘀嗒（SysCNTRick）校准值寄存器 182
8.1.2 中断和异常向量 182
8.2 外部中断/事件控制器（EXTI）..... 186
8.2.1 主要特性 186
8.2.2 框图 186
8.2.3 唤醒事件管理 186
8.2.4 功能说明 186
8.2.5 外部中断/事件线路映像 187
8.3 EXTI寄存器描述 188
8.3.1 中断屏蔽寄存器（EXTI_INTEN）..... 189
8.3.2 事件屏蔽寄存器（EXTI_EVTEN）..... 189

8.3.3 上升沿触发选择寄存器（EXTI_RTRSEL） 190
8.3.4 下降沿触发选择寄存器（EXTI_FTRSEL） 190
8.3.5 软件中断事件寄存器（EXTI_SWIE） 191
8.3.6 挂起寄存器（EXTI_PND） 191
9 DMA控制器（DMA） 192
9.1 DMA简介 192
9.2 DMA主要特性 192
9.3 功能描述 193
9.3.1 DMA处理 193
9.3.2 仲裁器 194
9.3.3 DMA通道 194
9.3.4 可编程的数据传输宽度、对齐方式和数据大小端 195
9.3.5 错误管理 196
9.3.6 中断 196
9.3.7 DMA固定请求映像 196
9.3.8 DMA弹性请求映像 199
9.4 DMA寄存器 200
9.4.1 DMA中断状态寄存器（DMA_ISTS） 204
9.4.2 DMA中断标志清除寄存器（DMA_ICLR） 204
9.4.3 DMA通道x配置寄存器（DMA_CHCTRLx）（x = 1…7） 205
9.4.4 DMA通道x传输数量寄存器（DMA_TCNTx）（x = 1…7） 207
9.4.5 DMA通道x外设地址寄存器（DMA_CPBAx）（x = 1…7） 207
9.4.6 DMA通道x存储器地址寄存器（DMA_CMBAx）（x = 1…7） 208
9.4.7 通道来源寄存器0（DMA_SRC_SEL0） 208
9.4.8 通道來源寄存器1（DMA_SRC_SEL1） 208
10 定时器（TIMER） 210
10.1 基本定时器（TMR6和TMR7） 210

10.1.1 TMR6和TMR7简介 210
10.1.2 TMR6和TMR7的主要特性 211
10.1.3 TMR6和TMR7的功能 211
10.1.3.1 时基单元 211
10.1.3.2 预分频器 212
10.1.3.3 计数模式 213
10.1.3.4 时钟源 216
10.1.3.5 调试模式 216
10.1.4 TMR6和TMR7寄存器 216
10.1.4.1 TMR6 和TMR7控制寄存器1（TMRx_CTRL1） 217
10.1.4.2 TMR6和TMR7控制寄存器2（TMRx_CTRL2） 218
10.1.4.3 TMR6 和TMR7 DMA/中断使能寄存器（TMRx_DIE） 219
10.1.4.4 TMR6和TMR7状态寄存器（TMRx_STS） 219
10.1.4.5 TMR6和TMR7事件产生寄存器（TMRx_EVEG） 219
10.1.4.6 TMR6和TMR7计数器（TMRx_CNT） 220
10.1.4.7 TMR6和TMR7预分频器（TMRx_DIV） 220
10.1.4.8 TMR6和TMR7自动重装载寄存器（TMRx_AR） 220
10.2 通用定时器（TMR2到TMR5） 221
10.2.1 TMRx简介 221
10.2.2 TMRx主要功能 221
10.2.3 TMRx功能描述 222
10.2.3.1 时基单元 222
10.2.3.2 计数器模式 223
10.2.3.3 时钟选择 231
10.2.3.4 捕获/比较通道 233
10.2.3.5 输入捕获模式 235
10.2.3.6 PWM输入模式 235
10.2.3.7 强置输出模式 236
10.2.3.8 输出比较模式 236
10.2.3.9 PWM模式 237
10.2.3.10 单脉冲模式 239
10.2.3.11 在外部事件时清除OCxREF信号 240

10.2.3.12 编码器接口模式 241
10.2.3.13 定时器输入异或功能 243
10.2.3.14 定时器和外部触发的同步 243
10.2.3.15 定时器同步 245
10.2.3.16 调试模式 250
10.2.4 TMRx寄存器描述 250
10.2.4.1 控制寄存器1（TMRx_CTRL1） 252
10.2.4.2 控制寄存器2（TMRx_CTRL2） 253
10.2.4.3 从模式控制寄存器（TMRx_SMC） 254
10.2.4.4 DMA/中断使能寄存器（TMRx_DIE） 256
10.2.4.5 状态寄存器（TMRx_STS） 257
10.2.4.6 事件产生寄存器（TMRx_EVEG） 258
10.2.4.7 捕获/比较模式寄存器1（TMRx_CCM1） 259
10.2.4.8 捕获/比较模式寄存器2（TMRx_CCM2） 261
10.2.4.9 捕获/比较使能寄存器（TMRx_CCE） 262
10.2.4.10 计数器（TMRx_CNT） 263
10.2.4.11 预分频器（TMRx_DIV） 264
10.2.4.12 自动重装载寄存器（TMRx_AR） 264
10.2.4.13 捕获/比较寄存器1（TMRx_CC1） 264
10.2.4.14 捕获/比较寄存器2（TMRx_CC2） 265
10.2.4.15 捕获/比较寄存器3（TMRx_CC3） 265
10.2.4.16 捕获/比较寄存器4（TMRx_CC4） 266
10.2.4.17 DMA控制寄存器（TMRx_DMAC） 266
10.2.4.18 连续模式的DMA地址（TMRx_DMABA） 267
10.3 通用定时器（TMR9到TMR14） 268
10.3.1 TMRx简介 268
10.3.2 TMRx主要功能 268
10.3.2.1 TMR9和TMR12主要功能 268
10.3.2.2 TMR10、TMR11、TMR13和 TMR14主要功能 269
10.3.3 TMRx功能描述 270
10.3.3.1 时基单元 270
10.3.3.2 计数器模式 271
10.3.3.3 时钟选择 274

10.3.3.4 捕获/比较通道 276
10.3.3.5 输入捕获模式 277
10.3.3.6 PWM输入模式 277
10.3.3.7 强置输出模式 278
10.3.3.8 输出比较模式 278
10.3.3.9 PWM模式 279
10.3.3.10 单脉冲模式 280
10.3.3.11 定时器和外部触发的同步 281
10.3.3.12 定时器同步 283
10.3.3.13 调试模式 283
10.3.4 TMR9和TMR12寄存器描述 283
10.3.4.1 控制寄存器 1（TMRx_CTRL1） 284
10.3.4.2 从模式控制寄存器（TMRx_SMC） 285
10.3.4.3 DMA/中断使能寄存器（TMRx_DIE） 286
10.3.4.4 状态寄存器（TMRx_STS） 287
10.3.4.5 事件产生寄存器（TMRx_EVEG） 288
10.3.4.6 捕获/比较模式寄存器1（TMRx_CCM1） 288
10.3.4.7 捕获/比较使能寄存器（TMRx_CCE） 290
10.3.4.8 计数器（TMRx_CNT） 291
10.3.4.9 预分频器（TMRx_DIV） 292
10.3.4.10 自动重装载寄存器（TMRx_AR） 292
10.3.4.11 捕获/比较寄存器1（TMRx_CC1） 292
10.3.4.12 捕获/比较寄存器2（TMRx_CC2） 293
10.3.5 TMR10、TMR11、TMR13和TMR14寄存器描述 293
10.3.5.1 控制寄存器1（TMRx_CTRL1） 294
10.3.5.2 DMA/中断使能寄存器（TMRx_DIE） 295
10.3.5.3 状态寄存器（TMRx_STS） 295
10.3.5.4 事件产生寄存器（TMRx_EVEG） 296
10.3.5.5 捕获/比较模式寄存器1（TMRx_CCM1） 296
10.3.5.6 捕获/比较使能寄存器（TMRx_CCE） 298
10.3.5.7 计数器（TMRx_CNT） 299
10.3.5.8 预分频器（TMRx_DIV） 299
10.3.5.9 自动重装载寄存器（TMRx_AR） 299
10.3.5.10 捕获/比较寄存器1（TMRx_CC1） 300

10.4 高级控制定时器（TMR1、TMR8） 301
10.4.1 TMR1、TMR8简介 301
10.4.2 TMR1、TMR8主要特性 301
10.4.3 TMR1、TMR8功能描述 302
10.4.3.1 时基单元 302
10.4.3.2 计数器模式 303
10.4.3.3 重复计数器 311
10.4.3.4 时钟选择 312
10.4.3.5 捕获/比较通道 315
10.4.3.6 输入捕获模式 317
10.4.3.7 PWM输入模式 317
10.4.3.8 强置输出模式 318
10.4.3.9 输出比较模式 318
10.4.3.10 PWM模式 319
10.4.3.11 互补输出和死区插入 321
10.4.3.12 使用刹车功能 323
10.4.3.13 在外部事件时清除OCxREF信号 324
10.4.3.14 产生六步PWM输出 325
10.4.3.15 单脉冲模式 326
10.4.3.16 编码器接口模式 327
10.4.3.17 定时器输入异或功能 329
10.4.3.18 与霍尔传感器的接口 329
10.4.3.19 TMRx定时器和外部触发的同步 331
10.4.3.20 定时器同步 334
10.4.3.21 调试模式 334
10.4.4 TMR1、TMR8寄存器描述 334
10.4.4.1 TMR1、TMR8 控制寄存器1（TMRx_CTRL1） 336
10.4.4.2 TMR1 、TMR8控制寄存器2（TMRx_CTRL2） 337
10.4.4.3 TMR1、TMR8从模式控制寄存器（TMRx_SMC） 339
10.4.4.4 TMR1、TMR8 DMA/中断使能寄存器（TMRx_DIE） 340
10.4.4.5 TMR1、TMR8状态寄存器（TMRx_STS） 341
10.4.4.6 TMR1、TMR8事件产生寄存器（TMRx_EVEG） 343
10.4.4.7 TMR1、TMR8捕获/比较模式寄存器1（TMRx_CCM1） 343

10.4.4.8 TMR1、TMR8捕获/比较模式寄存器2（TMRx_CCM2）.....346
10.4.4.9 TMR1、TMR8捕获/比较使能寄存器（TMRx_CCE）.....347
10.4.4.10 TMR1、TMR8计数器（TMRx_CNT）.....349
10.4.4.11 TMR1、TMR8预分频器（TMRx_DIV）.....350
10.4.4.12 TMR1、TMR8自动重装载寄存器（TMRx_AR）.....350
10.4.4.13 TMR1、TMR8重复计数寄存器（TMRx_RC）.....350
10.4.4.14 TMR1、TMR8捕获/比较寄存器 1（TMRx_CC1）.....351
10.4.4.15 TMR1、TMR8捕获/比较寄存器2（TMRx_CC2）.....351
10.4.4.16 TMR1、TMR8捕获/比较寄存器3（TMRx_CC3）.....351
10.4.4.17 TMR1、TMR8捕获/比较寄存器4（TMRx_CC4）.....352
10.4.4.18 TMR1、TMR8刹车和死区寄存器（TMRx_BRKDT）.....352
10.4.4.19 TMR1、TMR8DMA控制寄存器（TMRx_DMAC）.....353
10.4.4.20 TMR1、TMR8连续模式的DMA地址（TMRx_DMABA）.....354
11 看门狗.....355
11.1 窗口看门狗（WWDG）.....355
11.1.1 WWDG简介.....355
11.1.2 WWDG主要特性.....355
11.1.3 WWDG功能描述.....355
11.1.4 如何编写看门狗超时程序.....356
11.1.5 调试模式.....357
11.1.6 寄存器描述.....358
11.1.6.1 控制寄存器（WWDG_CTRL）.....358
11.1.6.2 配置寄存器（WWDG_CFG）.....359
11.1.6.3 状态寄存器（WWDG_STS）.....359
11.2 独立看门狗（IWDG）.....360
11.2.1 简介.....360
11.2.2 IWDG主要性能.....360
11.2.3 IWDG功能描述.....360
11.2.3.1 硬件看门狗.....360
11.2.3.2 寄存器访问保护.....360
11.2.3.3 调试模式.....360

11.2.4 IWDG寄存器描述 362
11.2.4.1 键寄存器（IWDG_KEY）..... 362
11.2.4.2 预分频寄存器（IWDG_PR）..... 363
11.2.4.3 重装载寄存器（IWDG_RLD）..... 363
11.2.4.4 状态寄存器（IWDG_STS）..... 364
12 实时时钟（RTC）..... 365
12.1 RTC简介 365
12.2 主要特性 365
12.3 功能描述 365
12.3.1 概述 365
12.3.2 复位过程 366
12.3.3 读RTC寄存器 366
12.3.4 配置RTC寄存器 367
12.3.5 RTC标志的设置 367
12.4 RTC寄存器描述 368
12.4.1 RTC控制寄存器高位（RTC_CTRLH）..... 368
12.4.2 RTC控制寄存器低位（RTC_CTRLL）..... 369
12.4.3 RTC预分频装载寄存器（RTC_DIVH/RTC_DIVL）..... 370
12.4.4 RTC预分频器余数寄存器（RTC_DIVCNTH / RTC_DIVCN TL）..... 371
12.4.5 RTC计数器寄存器（RTC_CNTH / RTC_CNTL）..... 371
12.4.6 RTC闹钟寄存器（RTC_ALAH/RTC_ALAL）..... 372
13 模拟/数字转换（ADC）..... 373
13.1 ADC介绍 373
13.2 ADC主要特征 373
13.3 ADC功能描述 374
13.3.1 ADC开关控制 375
13.3.2 ADC时钟 375
13.3.3 通道选择 375

13.3.4 单次转换模式 375
13.3.5 连续转换模式 376
13.3.6 时序图 376
13.3.7 模拟看门狗 376
13.3.8 扫描模式 377
13.3.9 注入通道管理 377
13.3.10 间断模式 378
13.3.11 校准 379
13.3.12 数据对齐 379
13.3.13 可编程的通道采样时间 380
13.3.14 外部触发转换 380
13.3.15 DMA请求 382
13.3.16 双ADC模式 382
13.3.16.1 同步注入模式 383
13.3.16.2 同步规则模式 384
13.3.16.3 快速交叉模式 384
13.3.16.4 慢速交叉模式 385
13.3.16.5 交替触发模式 386
13.3.16.6 独立模式 386
13.3.16.7 混合的规则/注入同步模式 386
13.3.16.8 混合的同步规则+交替触发模式 387
13.3.16.9 混合同步注入 +交叉模式 387
13.3.17 温度传感器 388
13.3.18 ADC中断 389
13.4 ADC寄存器 389
13.4.1 ADC状态寄存器（ADC_STS） 391
13.4.2 ADC控制寄存器1（ADC_CTRL1） 391
13.4.3 ADC控制寄存器2（ADC_CTRL2） 393
13.4.4 ADC采样时间寄存器1（ADC_SMPT1） 396

13.4.5 ADC采样时间寄存器2（ADC_SMPT2） 396
13.4.6 ADC注入通道数据偏移寄存器x（ADC_JOFSx）（x=1..4） 397
13.4.7 ADC看门狗高阀值寄存器（ADC_WHTR） 397
13.4.8 ADC看门狗低阀值寄存器（ADC_WLTR） 398
13.4.9 ADC规则序列寄存器1（ADC_RSQ1） 398
13.4.10 ADC规则序列寄存器2（ADC_RSQ2） 398
13.4.11 ADC规则序列寄存器3（ADC_RSQ3） 399
13.4.12 ADC注入序列寄存器（ADC_JSQ） 399
13.4.13 ADC 注入数据寄存器x（ADC_JDORx）（x= 1..4） 400
13.4.14 ADC规则数据寄存器（ADC_RDOR） 400
14 数字/模拟转换（DAC） 402
14.1 DAC简介 402
14.2 DAC主要特征 402
14.3 DAC功能描述 403
14.3.1 使能DAC通道 403
14.3.2 使能DAC输出缓存 403
14.3.3 DAC数据格式 403
14.3.4 DAC转换 405
14.3.5 DAC输出电压 405
14.3.6 选择DAC触发 405
14.3.7 DMA请求 406
14.3.8 噪声生成 406
14.3.9 三角波生成 407
14.4 双DAC通道转换 408
14.4.1 不使用波形发生器的独立触发 408
14.4.2 使用相同LFSR的独立触发 408
14.4.3 使用不同LFSR的独立触发 409

14.4.4 产生相同三角波的独立触发 409
14.4.5 产生不同三角波的独立触发 409
14.4.6 同时软件激活 409
14.4.7 不使用波形发生器的同时触发 410
14.4.8 使用相同LFSR的同时触发 410
14.4.9 使用不同LFSR的同时触发 410
14.4.10 使用相同三角波发生器的同时触发 410
14.4.11 使用不同三角波发生器的同时触发 411
14.5 DAC寄存器 411
14.5.1 DAC控制寄存器（DAC_CTRL） 412
14.5.2 DAC软件触发寄存器（DAC_SWTRG） 414
14.5.3 DAC信道1的12位右对齐数据保持寄存器（DAC_HDR12R1） 415
14.5.4 DAC信道1的12位左对齐数据保持寄存器（DAC_HDR12L 1） 415
14.5.5 DAC信道1的8位右对齐数据保持寄存器（DAC_HDR8R1） 416
14.5.6 DAC信道2的12位右对齐数据保持寄存器（DAC_HDR12 R2） 416
14.5.7 DAC信道2的12位左对齐数据保持寄存器（DAC_HDR12L 2） 417
14.5.8 DAC信道2的8位右对齐数据保持寄存器（DAC_HDR8R2） 417
14.5.9 双DAC的12位右对齐数据保持寄存器（DAC_HDR12RD） 417
14.5.10 双DAC的12位左对齐数据保持寄存器（DAC_HDR12LD） 418
14.5.11 双DAC的8位右对齐数据保持寄存器（DAC_HDR8RD） 418
14.5.12 DAC信道1数据输出寄存器（DAC_ODT1） 419
14.5.13 DAC信道2数据输出寄存器（DAC_ODT2） 419
15 I²C接口 420
15.1 I²C简介 420
15.2 I²C主要特点 420
15.3 I²C功能描述 421
15.3.1 模式选择 421

15.3.2 I²C从模式 422
15.3.3 I²C主模式 424
15.3.4 错误条件 428
15.3.5 SDA/SCL线控制 429
15.3.6 SMBus 430
15.3.7 DMA请求 431
15.3.8 包错误校验(PEC) 432
15.3.9 I²C中断请求 433
15.3.10 I²C调试模式 434
15.4 I²C寄存器描述 434
15.4.1 控制寄存器1(I²C_CTRL1) 436
15.4.2 控制寄存器2(I²C_CTRL2) 437
15.4.3 自身地址寄存器1(I²C_OADDR1) 438
15.4.4 自身地址寄存器2(I²C_OADDR2) 439
15.4.5 数据寄存器(I²C_DT) 439
15.4.6 状态寄存器1(I²C_STS1) 440
15.4.7 状态寄存器2(I²C_STS2) 442
15.4.8 时钟控制寄存器(I²C_CLKCTRL) 443
15.4.9 TMRISE寄存器(I²C_TMRISE) 444
16 通用同步异步收发器（USART） 445
16.1 USART介绍 445
16.2 USART主要特性 445
16.3 USART功能概述 446
16.3.1 USART特性描述 447
16.3.2 发送器 448
16.3.2.1 字符发送 448
16.3.2.2 可配置的停止位 448

16.3.2.3 单字节通信 449
16.3.2.4 断开帧 450
16.3.2.5 空闲符号 450
16.3.3 接收器 450
16.3.3.1 起始位侦测 451
16.3.3.2 字符接收 451
16.3.3.3 断开帧 452
16.3.3.4 空闲符号 452
16.3.3.5 溢出错误 452
16.3.3.6 帧错误 453
16.3.3.7 接收期间可配置的停止位 453
16.3.4 分数波特率的产生 454
16.3.4.1 如何从USART_BAUDR寄存器值得到USARTDIV 454
16.3.5 USART接收器容忍时钟的变化 456
16.3.6 多处理器通信 456
16.3.6.1 空闲总线检测（WUMODE=0） 456
16.3.6.2 地址标记（address mark）检测（WUMODE=1） 457
16.3.7 校验控制 457
16.3.8 LIN（局域互联网）模式 458
16.3.8.1 LIN发送 458
16.3.8.2 LIN接收 458
16.3.9 USART同步模式 460
16.3.10 单线半双工通信 462
16.3.11 智能卡 463
16.3.12 IrDA SIR ENDEC功能模块 464
16.3.13 利用DMA连续通信 466
16.3.13.1 利用DMA发送 466
16.3.13.2 利用DMA接收 467
16.3.13.3 多缓冲器通信中的错误标志和中断产生 468
16.3.14 硬件流控制 468
16.3.14.1 RTS流控制 468

16.3.14.2 CTS流控制 469
16.4 USART中断请求 469
16.5 USART模式配置 470
16.6 USART寄存器描述 470
16.6.1 USART寄存器地址映象 470
16.6.2 状态寄存器（USART_STS） 472
16.6.3 数据寄存器（USART_DT） 473
16.6.4 波特比率寄存器（USART_BAUDR） 474
16.6.5 控制寄存器1（USART_CTRL1） 474
16.6.6 控制寄存器2（USART_CTRL2） 476
16.6.7 控制寄存器3（USART_CTRL3） 477
16.6.8 保护时间和预分频寄存器（GTP） 479
17 串行外设接口（SPI） 480
17.1 SPI简介 480
17.2 主要特点 480
17.2.1 SPI特点 480
17.2.2 I²S功能 480
17.3 功能描述 481
17.3.1 SPI功能描述 481
17.3.1.1 概述 481
17.3.1.2 配置SPI为从模式 485
17.3.1.3 配置SPI为主模式 486
17.3.1.4 配置SPI为单工通信 487
17.3.1.5 数据发送与接收过程 487
17.3.1.6 CRC计算 492
17.3.1.7 状态标志 493
17.3.1.8 关闭SPI 494
17.3.1.9 利用DMA的SPI通信 495
17.3.1.10 错误标志 496
17.3.1.11 SPI中断 497

17.3.2 I²S功能描述 497
17.3.2.1 I²S功能描述 497
17.3.2.2 I2S 全双工 499
17.3.2.3 支持的音频协议 499
17.3.2.4 时钟发生器 505
17.3.2.5 I²S主模式 508
17.3.2.6 I²S从模式 509
17.3.2.7 状态标志位 510
17.3.2.8 错误标志位 510
17.3.2.9 I²S中断 511
17.3.2.10 DMA功能 511
17.4 SPI寄存器 511
17.4.1 SPI控制寄存器1（SPI_CTRL1）（I²S模式下不使用） 513
17.4.2 SPI控制寄存器2（SPI_CTRL2） 514
17.4.3 SPI状态寄存器（SPI_STS） 515
17.4.4 SPI数据寄存器（SPI_DT） 516
17.4.5 SPICRC多项式寄存器（SPI_CPOLY）（I²S模式下不使用） 516
17.4.6 SPIRxCRC寄存器（SPI_RCRC）（I²S模式下不使用） 517
17.4.7 SPITxCRC寄存器（SPI_TCRC） 517
17.4.8 SPI_I2S配置寄存器（SPI_I2SCTRL） 517
17.4.9 SPI_I2S预分频寄存器（SPI_I2SCLKP） 518
18 CAN总线控制器 520
18.1 简介 520
18.2 主要特点 520
18.3 功能描述 520
18.3.1 CAN整体功能描述 520
18.3.2 工作模式 522
18.3.2.1 初始化模式 522
18.3.2.2 正常模式 522
18.3.2.3 睡眠模式（低功耗） 523

18.3.3 测试模式 523
18.3.3.1 静默模式 523
18.3.3.2 环回模式 524
18.3.3.3 环回静默模式 524
18.3.4 AT32F403A/407系列处于调试模式时 525
18.3.5 发送处理 525
18.3.6 时间触发通信模式 526
18.3.7 接收管理 526
18.3.8 标识符过滤 528
18.3.9 报文存储 531
18.3.10 出错管理 532
18.3.11 位时间特性 533
18.3.12 bxCAN中断 536
18.4 CAN 寄存器 537
18.4.1 寄存器访问保护 539
18.4.2 CAN控制和状态寄存器 539
18.4.2.1 CAN主控制寄存器（CAN_MCTRL） 539
18.4.2.2 CAN主状态寄存器 （CAN_MSTS） 540
18.4.2.3 CAN发送状态寄存器 （CAN_TSTS） 541
18.4.2.4 CAN接收FIFO 0寄存器 （CAN_RF0） 543
18.4.2.5 CAN接收FIFO 1寄存器（CAN_RF1） 544
18.4.2.6 CAN中断使能寄存器 （CAN_INTEN） 544
18.4.2.7 CAN错误状态寄存器 （CAN_ESTS） 545
18.4.2.8 CAN位时序寄存器 （CAN_BTMG） 546
18.4.3 CAN邮箱寄存器 547
18.4.3.1 发送邮箱标识符寄存器 （CAN_TMIx） （x=0..2） 548
18.4.3.2 发送邮箱数据长度和时间戳寄存器 （CAN_TDTx） （x=0..2） 548
18.4.3.3 发送邮箱低字节数据寄存器 （CAN_TDLx） （x=0..2） 549
18.4.3.4 发送邮箱高字节数据寄存器 （CAN_TDHx） （x=0..2） 549
18.4.3.5 接收FIFO邮箱标识符寄存器 （CAN_RFIx） （x=0..1） 550

18.4.3.6 接收FIFO邮箱数据长度和时间戳寄存器 （CAN_RDTx） （x=0..1）550
18.4.3.7 接收FIFO邮箱低字节数据寄存器 （CAN_RDLx） （x=0..1） 551
18.4.3.8 接收FIFO邮箱高字节数据寄存器 （CAN_RDHx） （x=0..1） 551
18.4.4 CAN过滤器寄存器 552
18.4.4.1 CAN 过滤器主控寄存器 （CAN_FM） 552
18.4.4.2 CAN过滤器模式寄存器 （CAN_FM1） 552
18.4.4.3 CAN 过滤器位宽寄存器 （CAN_FS1） 553
18.4.4.4 CAN 过滤器FIFO关联寄存器 （CAN_FFA1） 553
18.4.4.5 CAN过滤器激活寄存器 （CAN_FA1） 553
18.4.4.6 CAN 过滤器组i的寄存器x （CAN_FBiRx） （其中i＝0..13；x=1..2）554
19 外部存储控制器（XMC） 555
19.1 简介 555
19.2 主要特点 555
19.2.1 框图 555
19.2.2 AHB接口 556
19.2.3 支持的存储器和操作 556
19.3 功能描述 557
19.3.1 地址映射 557
19.3.1.1 NOR和PSRAM地址映射 557
19.3.1.2 NAND地址映射 558
19.3.2 NOR闪存/PSRAM控制器 559
19.3.2.1 外部存储器接口信号 560
19.3.2.2 支持的存储器及其操作 561
19.3.2.3 时序规则 561
19.3.2.4 NOR闪存和PSRAM控制器时序图 561
19.3.2.5 同步的成组读 573
19.3.3 NAND闪存控制器 577
19.3.3.1 外部存储器接口信号 577
19.3.3.2 NAND闪存支持的存储器及其操作 578
19.3.3.3 NAND闪存时序图 578
19.3.3.4 NAND闪存操作 579

19.3.3.5 NAND闪存预等待功能 580
19.3.3.6 NAND闪存的纠错码ECC计算（NAND闪存） 580
19.4 XMC寄存器 581
19.4.1 NOR闪存和PSRAM控制器寄存器 581
19.4.1.1 SRAM/NOR闪存片选控制寄存器1,4（XMC_BK1CTRL1,4） 581
19.4.1.2 SRAM/NOR闪存片选时序寄存器1,4（XMC_BK1TMG1,4） 583
19.4.1.3 SRAM/NOR闪存写时序寄存器1,4（XMC_BK1TMGWR1,4） 585
19.4.1.4 SRAM/NOR额外时序寄存器1,4（XMC_EXT1,4） 586
19.4.2 NAND闪存控制器寄存器 586
19.4.2.1 NAND闪存控制寄存器2（XMC_BK2CTRL） 587
19.4.2.2 FIFO状态和中断寄存器2（XMC_BK2STS） 588
19.4.2.3 通用存储空间时序寄存器2（XMC_BK2TMGMEM） 589
19.4.2.4 属性存储空间时序寄存器2（XMC_BK2TMGATT） 589
19.4.2.5 ECC结果寄存器2（XMC_BK2ECC） 590
20 SDIO接口 592
20.1 简介 592
20.2 主要特点 592
20.3 功能描述 594
20.3.1 SDIO功能描述 594
20.3.1.1 SDIO适配器 595
20.3.1.2 SDIO AHB接口 603
20.3.2 卡功能描述 603
20.3.2.1 卡识别模式 603
20.3.2.2 卡复位 603
20.3.2.3 操作电压范围确认 604
20.3.2.4 卡识别过程 604
20.3.2.5 写数据块 605
20.3.2.6 读数据块 605
20.3.2.7 数据流操作，数据流写入和数据流读出（只适用于多媒体卡） 605
20.3.2.8 擦除：成组擦除和扇区擦除 606
20.3.2.9 宽总线选择和解除选择 606
20.3.2.10 保护管理 607

20.3.2.11 卡状态寄存器 609
20.3.2.12 SD状态寄存器 611
20.3.2.13 SD的I/O模式 614
20.3.2.14 命令与响应 614
20.3.3 响应格式 617
20.3.3.1 R1（普通响应命令） 617
20.3.3.2 R1b 617
20.3.3.3 R2（CID、CSD寄存器） 617
20.3.3.4 R3（OCR寄存器） 618
20.3.3.5 R4（快速I/O） 618
20.3.3.6 R4b 618
20.3.3.7 R5（中断请求） 619
20.3.3.8 R6（中断请求） 619
20.3.4 SDIO I/O卡特定的操作 620
20.3.4.1 使用SDIO_D2信号线的SDIO I/O读等待操作 620
20.3.4.2 使用停止SDIO_CK的SDIO读等待操作 620
20.3.4.3 SDIO暂停/恢复操作 620
20.3.4.4 SDIO中断 620
20.3.5 硬件流控制 620
20.4 SDIO寄存器 621
20.4.1 SDIO电源控制寄存器（SDIO_POWER） 622
20.4.2 SDIO时钟控制寄存器（SDIO_CLKCTRL） 623
20.4.3 SDIO参数寄存器（SDIO_ARG） 623
20.4.4 SDIO命令寄存器（SDIO_CMD） 624
20.4.5 SDIO命令响应寄存器（SDIO_RSPCMD） 625
20.4.6 SDIO响应1..4寄存器（SDIO_RSPx） 625
20.4.7 SDIO数据定时器寄存器（SDIO_DTTMR） 626
20.4.8 SDIO数据长度寄存器（SDIO_DTLEN） 626
20.4.9 SDIO数据控制寄存器（SDIO_DTCTRL） 626
20.4.10 SDIO数据计数器寄存器（SDIO_DTCNTR） 628

20.4.11 SDIO状态寄存器（SDIO_STS） 628
20.4.12 SDIO清除中断寄存器（SDIO_INTCLR） 629
20.4.13 SDIO中断屏蔽寄存器（SDIO_INTEN） 630
20.4.14 SDIOBUF计数器寄存器（SDIO_BUFCNTR） 632
20.4.15 SDIO数据BUF寄存器（SDIO_BUF） 633
21 通用串行总线全速设备接口（USBDEV） 634
21.1 简介 634
21.2 USB主要特点 634
21.3 功能描述 635
21.3.1 USB功能模块描述 636
21.3.2 通用USB设备编程 637
21.3.2.1 系统复位和上电复位 637
21.3.2.2 双缓冲端点 640
21.3.2.3 同步传输 641
21.3.2.4 挂起/恢复事件 642
21.3.2.5 SOF输出 643
21.4 USB寄存器 643
21.4.1 通用寄存器 645
21.4.1.1 USB控制寄存器（USB_CTRL） 645
21.4.1.2 USB中断状态寄存器（USB_INTSTS） 647
21.4.1.3 USB帧编号寄存器（USB_FRNUM） 649
21.4.1.4 USB设备地址寄存器（USB_DEVADR） 649
21.4.1.5 USB分组缓冲区描述表地址寄存器（USB_BUFTBL） 649
21.4.1.6 USB SOF控制寄存器（USB_SOFEN） 650
21.4.2 端点寄存器 650
21.4.2.1 USB端点n寄存器（USB_EPTn），n=[0..7] 650
21.4.3 缓冲区描述表 653
21.4.3.1 发送缓冲区地址寄存器 n（USB_ADRn_TX） 653
21.4.3.2 发送数据字节数寄存器 n（USB_CNTn_TX） 653
21.4.3.3 接收缓冲区地址寄存器 n（USB_ADRn_RX） 653

21.4.3.4 接收数据字节数寄存器 n（USB_CNTn_RX）.....654
22 MCU调试（MCUDBG）.....655
22.1 简介.....655
22.2 功能描述.....656
22.2.1 低功耗模式的调试支持.....656
22.2.2 支持定时器、看门狗、bxCAN和I2C的调试.....656
22.2.3 ID代码.....656
22.2.4 SWJ调试端口脚.....656
22.2.5 JTAG脚上的内部上拉和下拉.....656
22.2.6 跟踪脚的分配控制.....657
22.3 MCUDBG寄存器.....657
22.3.1 MCUDBG控制寄存器（MCUDBG_CTRL）.....658
23 HSI自动时钟校准（HSI Auto Clock Calibration）.....660
23.1 ACC介绍.....660
23.2 ACC主要特性.....660
23.3 ACC功能概述.....660
23.3.1 ACC特性描述.....661
23.4 ACC中断请求.....662
23.5 ACC寄存器描述.....662
23.5.1 ACC寄存器地址映象.....663
23.5.2 状态寄存器（ACC_STS）.....663
23.5.3 控制寄存器1（ACC_CTRL1）.....664
23.5.4 控制寄存器2（ACC_CTRL2）.....664
23.5.5 比较值1（ACC_C1）.....665
23.5.6 比较值2（ACC_C2）.....665
23.5.7 比较值3（ACC_C3）.....666
24 以太网（ETH）：具有DMA控制器的介质访问控制（MAC）.....667
24.1 以太网模块介绍.....667

24.2 以太网模块主要功能 667
24.2.1 MAC控制器功能 667
24.2.2 DMA功能 668
24.2.3 PTP功能 669
24.3 以太网模块引脚和内部信号 669
24.4 以太网模块功能描述：SMI、MII和RMII 669
24.4.1 站点管理接口(SMI) 670
24.4.2 独立于介质的接口：MII 672
24.4.3 精简的独立于介质的接口：RMII 673
24.4.4 MII/RMII的选择 674
24.5 以太网模块功能描述：MAC 802.3 675
24.5.1 MAC 802.3 帧格式 675
24.5.2 MAC帧的传输 678
24.5.3 MAC帧的接收 683
24.5.4 MAC中断 686
24.5.5 MAC过滤 686
24.5.6 MAC自循环模式 688
24.5.7 MAC管理计数器：MMC 688
24.5.8 电源管理：PMT 689
24.5.9 精确时间协议(IEEE1588 PTP) 691
24.6 以太网功能描述：DMA控制器操作 695
24.6.1 使用DMA发送的初始化步骤 696
24.6.2 主机总线突发访问 697
24.6.3 主机数据缓存对齐 697
24.6.4 缓冲区大小计算 697
24.6.5 DMA仲裁器 698
24.6.6 DMA错误响应 698

24.6.7 发送DMA设置 698
24.6.8 接收DMA设置 707
24.6.9 DMA中断 713
24.7 以太网中断 714
24.8 以太网寄存器 716
24.8.1 MAC寄存器描述 719
24.8.2 MMC寄存器描述 730
24.8.3 IEEE 1588 时间戳寄存器 734
24.8.4 DMA寄存器描述 740
25 版本历史 751

图目录

图 1-1 AT32F403A 系列微控制器系统架构 47
图 1-2 AT32F407 系列微控制器系统架构 49
图 1-3 Cortex™-M4F 内部框图 52
图 1-4 AT32F403A 地址配置 53
图 2-1 各电源域框图 63
图 2-2 上电复位和掉电复位的波形图 64
图 3-1 复位电路 73
图 3-2 AT32F403A 时钟树 74
图 3-3 AT32F407 时钟树 75
图 3-4 HSE/LSE 时钟源 76
图 5-1 编程过程 116
图 5-2 外部存储器参考线路 116
图 5-3 编程过程 118
图 5-4 闪存页擦除过程 119
图 5-5 闪存整片擦除过程 120
图 6-1 CRC 单元框 142
图 7-1 I/O 端口位的基本结构 146
图 7-2 伏兼容 I/O 端口位的基本结构 146
图 7-3 输入浮空/上拉/下拉配置 148
图 7-4 高阻抗的模拟输入配置 149
图 7-5 输出配置 150
图 7-6 复用功能配置 151
图 8-1 外部中断/事件控制器框图 186
图 8-2 外部中断通用 I/O 映像 188
图 9-1 DMA 框图 193
图 9-2 DMA1 请求映像 197
图 9-3 DMA2 请求映像 198
图 10-1 基本定时器框图 211
图 10-2 预分频系数从 1 变到 2 的计数器时序图 212
图 10-3 预分频系数从 1 变到 4 的计数器时序图 213
图 10-4 计数器时序图，内部时钟分频系数为 1 214
图 10-5 计数器时序图，内部时钟分频系数为 2 214
图 10-6 计数器时序图，内部时钟分频系数为 4 214
图 10-7 计数器时序图，内部时钟分频系数为 N 215
图 10-8 计数器时序图，当 ARPEN=0 时的更新事件（TMRx_AR 没有预装载） 215
图 10-9 计数器时序图，当 ARPEN=1 时的更新事件（预装载 TMRx_AR） 216
图 10-10 普通模式时序图，内部时钟分频系数为 1 216
图 10-11 通用定时器框图 222
图 10-12 当预分频器的参数从 1 变到 2 时，计数器的时序图 223

图 10-13 当预分频器的参数从 1 变到 4 时，计数器的时序图 223
图 10-14 计数器时序图，内部时钟分频因子为 1 224
图 10-15 计数器时序图，内部时钟分频因子为 2 224
图 10-16 计数器时序图，内部时钟分频因子为 4 225
图 10-17 计数器时序图，内部时钟分频因子为 N 225
图 10-18 计数器时序图，当 ARPEN=0 时的更新事件（没有预装入了 TMRx_AR） 225
图 10-19 计数器时序图，当 ARPEN=1 时的更新事件（预装入了 TMRx_AR） 226
图 10-20 计数器时序图，内部时钟分频因子为 1 227
图 10-21 计数器时序图，内部时钟分频因子为 2 227
图 10-22 计数器时序图，内部时钟分频因子为 4 227
图 10-23 计数器时序图，内部时钟分频因子为 N 228
图 10-24 计数器时序图，当 ARPEN=0 时的更新事件 228
图 10-25 计数器时序图，内部时钟分频因子为 1，TMRx_AR=0x6 229
图 10-26 计数器时序图，内部时钟分频因子为 2 229
图 10-27 计数器时序图，内部时钟分频因子为 4，TMRx_AR=0x36 230
图 10-28 计数器时序图，内部时钟分频因子为 N 230
图 10-29 计数器时序图，ARPEN=1 时的更新事件（计数器下溢） 230
图 10-30 计数器时序图，ARPEN=1 时的更新事件（计数器溢出） 231
图 10-31 一般模式下的控制电路，内部时钟分频因子为 1 231
图 10-32 TI2 外部时钟连接例子 232
图 10-33 外部时钟模式 1 下的控制电路 232
图 10-34 外部触发输入框图 233
图 10-35 外部时钟模式 2 下的控制电路 233
图 10-36 捕获/比较通道（如：通道 1 输入部分） 234
图 10-37 捕获/比较通道 1 的主电路 234
图 10-38 捕获/比较通道的输出部分（通道 1） 234
图 10-39 PWM 输入模式时序 236
图 10-40 输出比较模式，翻转 OC1 237
图 10-41 边沿对齐的 PWM 波形（AR=8） 238
图 10-42 中央对齐的 PWM 波形（AP=8） 239
图 10-43 单脉冲模式的例子 240
图 10-44 清除 TMRx 的 OCxREF 241
图 10-45 编码器模式下的计数器操作实例 242
图 10-46 IC1FP1 反相的编码器接口模式实例 242
图 10-47 复位模式下的控制电路 243
图 10-48 门控模式下的控制电路 244
图 10-49 触发器模式下的控制电路 244
图 10-50 外部时钟模式 2＋触发模式下的控制电路 245
图 10-51 主/从定时器的例子 246
图 10-52 定时器 1 的 OC1REF 控制定时器 2 247
图 10-53 通过使能定时器 1 可以控制定时器 2 248

图 10-54 使用定时器 1 的更新触发定时器 2 248
图 10-55 利用定时器 1 的使能触发定时器 2 249
图 10-56 使用定时器 1 的 TI1 输入触发定时器 1 和定时器 2 250
图 10-57 通用定时器 TMR9/12 框图 269
图 10-58 通用定时器 TMR10/11/13/14 框图 270
图 10-59 当预分频器的参数从 1 变到 2 时，计数器的时序图 271
图 10-60 当预分频器的参数从 1 变到 4 时，计数器的时序图 271
图 10-61 计数器时序图，内部时钟分频因子为 1 272
图 10-62 计数器时序图，内部时钟分频因子为 2 272
图 10-63 计数器时序图，内部时钟分频因子为 4 273
图 10-64 计数器时序图，内部时钟分频因子为 N 273
图 10-65 计数器时序图，当 ARPEN=0 时的更新事件（TMRx_AR 没有预装入） 273
图 10-66 计数器时序图，当 ARPEN=1 时的更新事件（预装入了 TMRx_AR） 274
图 10-67 一般模式下的控制电路，内部时钟分频因子为 1 274
图 10-68 TI2 外部时钟连接例子 275
图 10-69 外部时钟模式 1 下的控制电路 275
图 10-70 捕获/比较通道（如：通道 1 输入部分） 276
图 10-71 捕获/比较通道 1 的主电路 276
图 10-72 捕获/比较通道的输出部分（通道 1） 277
图 10-73 PWM 输入模式时序 278
图 10-74 输出比较模式，翻转 OC1 279
图 10-75 边沿对齐的 PWM 波形（AR=8） 280
图 10-76 单脉冲模式的例子 280
图 10-77 复位模式下的控制电路 282
图 10-78 门控模式下的控制电路 282
图 10-79 触发器模式下的控制电路 283
图 10-80 高级控制定时器框图 302
图 10-81 当预分频器的参数从 1 变到 2 时，计数器的时序图 303
图 10-82 当预分频器的参数从 1 变到 4 时，计数器的时序图 303
图 10-83 计数器时序图，内部时钟分频因子为 1 304
图 10-84 计数器时序图，内部时钟分频因子为 2 304
图 10-85 计数器时序图，内部时钟分频因子为 4 305
图 10-86 计数器时序图，内部时钟分频因子为 N 305
图 10-87 计数器时序图，当 ARPEN=0 时的更新事件（TMRx_AR 没有预装入） 305
图 10-88 计数器时序图，当 ARPEN=1 时的更新事件（预装入了 TMRx_AR） 306
图 10-89 计数器时序图，内部时钟分频因子为 1 307
图 10-90 计数器时序图，内部时钟分频因子为 2 307
图 10-91 计数器时序图，内部时钟分频因子为 4 307
图 10-92 计数器时序图，内部时钟分频因子为 N 308
图 10-93 计数器时序图，当没有使用自动装载时的更新事件 308
图 10-94 计数器时序图，内部时钟分频因子为 1，TMRx_AR=0x6 309

图 10-95 计数器时序图，内部时钟分频因子为 2 309
图 10-96 计数器时序图，内部时钟分频因子为 4，TMRx_AR=0x36 310
图 10-97 计数器时序图，内部时钟分频因子为 N 310
图 10-98 计数器时序图，ARPEN=1 时的更新事件（计数器下溢） 311
图 10-99 计数器时序图，ARPEN=1 时的更新事件（计数器溢出） 311
图 10-100 不同模式下更新速率的例子，及 TMRx_RC 的寄存器设置 312
图 10-101 一般模式下的控制电路，内部时钟分频因子为 1 313
图 10-102 TI2 外部时钟连接例子 313
图 10-103 外部时钟模式 1 下的控制电路 314
图 10-104 外部触发输入框图 314
图 10-105 外部时钟模式 2 下的控制电路 315
图 10-106 捕获/比较通道（如：通道 1 输入部分） 315
图 10-107 捕获/比较通道 1 的主电路 316
图 10-108 捕获/比较通道的输出部分（通道 1 至 3） 316
图 10-109 捕获/比较通道的输出部分（通道 4） 316
图 10-110 PWM 输入模式时序 318
图 10-111 输出比较模式，翻转 OC1 319
图 10-112 边沿对齐的 PWM 波形（AR=8） 320
图 10-113 中央对齐的 PWM 波形（AR=8） 321
图 10-114 带死区插入的互补输出 322
图 10-115 死区波形延迟大于负脉冲 322
图 10-116 死区波形延迟大于正脉冲 322
图 10-117 响应刹车的输出 324
图 10-118 清除 TMRx 的 OCxREF 325
图 10-119 产生六步 PWM，使用 HALL 的例子（OSIMR=1） 326
图 10-120 单脉冲模式的例子 327
图 10-121 编码器模式下的计数器操作实例 329
图 10-122 IC1FP1 反相的编码器接口模式实例 329
图 10-123 霍尔传感器接口的实例 331
图 10-124 复位模式下的控制电路 332
图 10-125 门控模式下的控制电路 333
图 10-126 触发器模式下的控制电路 333
图 10-127 外部时钟模式 2＋触发模式下的控制电路 334
图 11-1 看门狗框图 356
图 11-2 窗口看门狗时序图 357
图 11-3 独立看门狗框图 360
图 12-1 简化的 RTC 框图 366
图 12-2 RTC 秒和闹钟波形图示例，PR=0003，ALARM=00004 367
图 12-3 RTC 溢出波形图示例，PR=0003 367
图 13-1 单个 ADC 框图 374
图 13-2 时序图 376

图 13-3 模拟看门狗警戒区 377
图 13-4 注入转换延时 378
图 13-5 校准时序图 379
图 13-6 数据右对齐 380
图 13-7 数据左对齐 380
图 13-8 双 ADC 框图(1) 383
图 13-9 在 4 个通道上的同步注入模式 384
图 13-10 在 16 个通道上的同步规则模式 384
图 13-11 在 1 个通道上连续转换模式下的快速交叉模式 385
图 13-12 在 1 个通道上的慢速交叉模式 385
图 13-13 交替触发：每个 ADC1 的注入通道组 386
图 13-14 交替触发：在间断模式下每个 ADC 上的 4 个注入通道 386
图 13-15 交替+规则同步 387
图 13-16 触发事件发生在注入转换期间 387
图 13-17 交叉的单通道转换被注入序列 CH11 和 CH12 中断 388
图 13-18 温度传感器和 V_{REFINT} 通道框图 388
图 14-1 DAC 信道模块框图 403
图 14-2 单 DAC 信道模式的数据寄存器 404
图 14-3 双 DAC 信道模式的数据寄存器 404
图 14-4 TEN=0 触发禁止时转换的时间框图 405
图 14-5 DAC LFSR 寄存器算法 406
图 14-6 带 LFSR 波形生成的 DAC 转换（使能软件触发） 407
图 14-7 DAC 三角波生成 407
图 14-8 带三角生成的 DAC 转换（使能软件触发） 408
图 15-1 I^2C 总线协议 421
图 15-2 I^2C 的功能框图 422
图 15-3 从发送器的传送序列图 423
图 15-4 从接收器的传送序列图 424
图 15-5 主发送器传送序列图 426
图 15-6 主接收器传送序列图 427
图 15-7 N>2 时主收流程图 427
图 15-8 N=2 时主收流程图 428
图 15-9 N=1 时主收流程图 428
图 15-10 I^2C 中断映射图 434
图 16-1 USART 框图 447
图 16-2 字长设置 448
图 16-3 配置停止位 449
图 16-4 发送时 TRAC/TDE 的变化情况 450
图 16-5 起始位侦测 451
图 16-6 检测噪声的数据采样 453

图 16-7 利用空闲总线检测的静默模式 457
图 16-8 利用地址标记检测的静默模式 457
图 16-9 LIN 模式下的断开检测（11 位断开长度 - 设置了 LBDLEN 位） 459
图 16-10 LIN 模式下的断开检测与帧错误的检测 460
图 16-11 USART 同步传输的例子 461
图 16-12 USART 数据时钟时序示例（LEN=0） 461
图 16-13 USART 数据时钟时序示例（LEN=1） 462
图 16-14 RX 数据采样/保持时间 462
图 16-15 ISO7816-3 异步协议 463
图 16-16 使用 1.5 停止位检测奇偶检验错 464
图 16-17 IrDA SIR ENDEC 框图 466
图 16-18 IrDA 数据调制（3/16） - 普通模式 466
图 16-19 利用 DMA 发送 467
图 16-20 利用 DMA 接收 468
图 16-21 两个 USART 间的硬件流控制 468
图 16-22 RTS 流控制 469
图 16-23 CTS 流控制 469
图 16-24 USART 中断映像图 470
图 17-1 SPI 框图 481
图 17-2 单主和单从应用 482
图 17-3 硬件/软件的从选择管理 484
图 17-4 数据时钟时序图 485
图 17-5 主模式、全双工模式下（BDMODE=0 并且 RONLY=0）连续输出时，TE/RNE/BSY 的变化示意图 489
图 17-6 从模式、全双工模式下（BDMODE=0 并且 RONLY=0）连续传输时，TE/RNE/BSY 的变化示意图。 490
图 17-7 主设备只发送模式（BDMODE=0 并且 RONLY=0）下连续传输时，TE/BSY 变化示意图 491
图 17-8 从设备只发送模式（BDMODE=0 并且 RONLY=0）下连续传输时，TE/BSY 变化示意图 491
图 17-9 只接收模式（BDMODE=0 并且 RONLY=1）下连续传输时，RNE 变化示意图 492
图 17-10 非连续传输发送（BDMODE=0 并且 RONLY=0）时，TE/BSY 变化示意图 492
图 17-11 使用 DMA 发送 496
图 17-12 使用 DMA 接收 496
图 17-13 I^2S 框图 498
图 17-14 I2S 全双工结构图 499
图 17-15 I^2S 飞利浦协议波形（16/32 位全精度，CPOL=0） 500
图 17-16 I^2S 飞利浦协议标准波形（24 位帧，CPOL=0） 500
图 17-17 发送 0x8EAA33 500
图 17-18 接收 0x8EAA33 501
图 17-19 I^2S 飞利浦协议标准波形（16 位扩展至 32 位包帧，CPOL=0） 501
图 17-20 示例 501
图 17-21 MSB 对齐 16 位或 32 位全精度，CPOL=0 502

图 17-22 MSB 对齐 24 位数据，CPOL=0 502
图 17-23 MSB 对齐 16 位数据扩展到 32 位包帧，CPOL=0 502
图 17-24 LSB 对齐 16 位或 32 位全精度，CPOL=0 503
图 17-25 LSB 对齐 24 位数据，CPOL=0 503
图 17-26 要求发送 0x3478AE 的操作 503
图 17-27 要求接收 0x3478AE 的操作 503
图 17-28 LSB 对齐 16 位数据扩展到 32 位包帧，CPOL=0 504
图 17-29 示例 504
图 17-30 PCM 标准波形（16 位） 504
图 17-31 PCM 标准波形（16 位扩展到 32 位包帧） 505
图 17-32 音频采样频率定义 505
图 17-33 I^2S 时钟发生器结构 506
图 18-1 CAN 网拓扑结构 521
图 18-2 CAN 框图 522
图 18-3 bxCAN 工作模式 523
图 18-4 bxCAN 工作在静默模式 524
图 18-5 bxCAN 工作在环回模式 524
图 18-6 bxCAN 工作在环回静默模式 525
图 18-7 发送邮箱状态 526
图 18-8 接收 FIFO 状态 527
图 18-9 过滤器组位宽设置－寄存器组织 529
图 18-10 过滤器编号的例子 530
图 18-11 过滤器机制的例子 531
图 18-12 CAN 错误状态图 532
图 18-13 位时序 534
图 18-14 各种 CAN 帧 535
图 18-15 事件标志和中断产生 536
图 18-16 发送和接收邮箱 547
图 19-1 XMC 框图 555
图 19-2 XMC 存储块 557
图 19-3 模式 1 读操作 562
图 19-4 模式 1 写操作 562
图 19-5 模式 A 读操作 563
图 19-6 模式 A 写操作 564
图 19-7 模式 2/B 读操作 565
图 19-8 模式 2 写操作 566
图 19-9 模式 B 写操作 566
图 19-10 模式 C 读操作 568
图 19-11 模式 C 写操作 568
图 19-12 模式 D 读操作 570
图 19-13 模式 D 写操作 570

图 19-14 复用读操作 572
图 19-15 复用写操作 572
图 19-16 同步的总线复用读模式 - NOR,PSRAM（CRAM） 574
图 19-17 同步的总线复用写模式 - PSRAM（CRAM） 576
图 19-18 NAND 控制器通用存储空间的访问时序 579
图 19-19 操作 CE 敏感型 NAND 闪存 580
图 20-1 SDIO“无响应”和“无数据”操作 593
图 20-2 SDIO（多）数据块读操作 593
图 20-3 SDIO（多）数据块写操作 593
图 20-4 SDIO 连续读操作 594
图 20-5 SDIO 连续写操作 594
图 20-6 SDIO 框图 595
图 20-7 SDIO 适配器 596
图 20-8 控制单元 597
图 20-9 SDIO 适配器命令通道 597
图 20-10 命令通道状态机（CPSM） 598
图 20-11 SDIO 命令传输 598
图 20-12 数据通道 600
图 20-13 数据通道状态机（DPSM） 601
图 21-1 USB 设备框图 635
图 21-2 分组缓冲区对应的缓冲区描述表项定位 638
图 22-1 AT32F403A/407 级别和 Cortex™-M4F 级别的调试框图 655
图 23-1 ACC 框图 661
图 23-2 cross-return 策略 661
图 23-3 ACC 中断映像图 662
图 24-1 ETH 框图 670
图 24-2 SMI 接口信号 670
图 24-3 MDIO 时序和帧格式图－写操作 671
图 24-4 MDIO 时序和帧格式图－读操作 671
图 24-5 独立于介质的接口(MII)信号线 672
图 24-6 MII 时钟源 673
图 24-7 精简的独立于介质的接口信号 674
图 24-8 RMII 时钟源 674
图 24-9 内部时钟结构 675
图 24-10 地址域格式 676
图 24-11 MAC 帧格式 677
图 24-12 带标签的 MAC 帧格式 678
图 24-13 位传输顺序 682
图 24-14 无冲突时的传输 682
图 24-15 有冲突时的传输 682
图 24-16 MII 和 RMII 模式下的帧传输 683

图 24-17 接收位顺序 685
图 24-18 接收时无错误 685
图 24-19 接收时有错误 686
图 24-20 接收时有假载波指示 686
图 24-21 MAC 控制器中断屏蔽示意图 686
图 24-22 唤醒帧过滤寄存器 689
图 24-23 网络时钟同步 692
图 24-24 用精调的方式更新系统时间 693
图 24-25 PTP 触发输出连接到 TMR2 的 ITR1 695
图 24-26 PPS 输出 695
图 24-27 描述符的环结构和链结构 696
图 24-28 默认模式下发送 DMA 的操作 699
图 24-29 OPSF 模式下发送 DMA 的操作 700
图 24-30 传输描述符 701
图 24-31 IEEE1588 时间戳使能时的发送描述符格式 704
图 24-32 接收 DMA 操作 708
图 24-33 接收 DMA 描述符 709
图 24-34 IEEE1588 时间戳使能时的接收描述符格式 713
图 24-35 DMA 中断示意图 714
图 24-36 以太网 MAC 唤醒帧过滤器寄存器(ETH_MACRWFF) 725

表目录

表 1-1 寄存器组起始地址 53
表 1-2 启动模式 59
表 1-3 寄存器缩写表 60
表 2-1 SLP-NOW 模式 66
表 2-2 SLP-ON-EXIT 模式 67
表 2-3 停止模式 67
表 2-4 待机模式 68
表 3-1 RCC 寄存器的映像和复位值 80
表 4-1 BKPR 寄存器映像和复位值 107
表 5-1 闪存模块组织（1024K） 112
表 5-2 闪存模块组织（512K） 113
表 5-3 闪存模块组织（256K） 114
表 7-1 端口位配置表 146
表 7-2 输出模式位 147
表 7-3 高级定时器 TMR1/8 151
表 7-4 通用定时器 TMR2-5/TMR9-14 151
表 7-5 USART 152
表 7-6 SPI 152
表 7-7 I²S 152
表 7-8 I²C 接口 153
表 7-9 BxCAN 153
表 7-10 USB 153
表 7-11 SDIO 153
表 7-12 ADC/DAC 153
表 7-13 Ethernet 153
表 7-14 XMC 154
表 7-15 其它 I/O 功能 154
表 7-16 CAN1 复用功能重映射 155
表 7-17 CAN2 复用功能重映射 155
表 7-18 调试接口信号 155
表 7-19 调试端口映射 156
表 7-20 ADC1 外部触发注入转换复用功能重映射 156
表 7-21 ADC1 外部触发规则转换复用功能重映射 156
表 7-22 ADC2 外部触发注入转换复用功能重映射 156
表 7-23 ADC2 外部触发规则转换复用功能重映射 156
表 7-24 TMR1 复用功能重映射 156
表 7-25 TMR2 复用功能重映射 157
表 7-26 TMR3 复用功能重映射 157
表 7-27 TMR4 复用功能重映射 157

表 7-28 TMR5 复用功能重映射 157
表 7-29 TMR9 复用功能重映射 158
表 7-30 USART1 重映射 158
表 7-31 USART2 重映射 158
表 7-32 USART3 重映射 158
表 7-33 UART4 重映射 158
表 7-34 UART5 重映射 159
表 7-35 USART6 重映射 159
表 7-36 UART7 重映射 159
表 7-37 UART8 重映射 159
表 7-38 I²C1 重映射 159
表 7-39 I²C3 重映射 159
表 7-40 SPI1/I²S1 重映射 160
表 7-41 SPI2/I²S2 重映射 160
表 7-42 SPI3/I²S3/I²SEXT3 重映射 160
表 7-43 SPI4/IS24 重映射 160
表 7-44 SDIO2 Dx 复用功能重映射 160
表 7-45 SDIO2 CK/CMD 复用功能重映射 161
表 7-46 外部 SPIF 复用功能重映射 161
表 7-47 Ethernet 复用功能重映射 161
表 7-48 XMC 复用功能重映射 161
表 7-49 GPIO 寄存器地址映射和复位值 162
表 7-50 AFIO 寄存器地址映射和复位值 163
表 8-1 AT32F403A/407 产品的向量表 182
表 8-2 外部中断/事件控制器寄存器映像和复位值 189
表 9-1 可编程的数据传输宽度和大小端操作（当 PINC = MINC = 1） 195
表 9-2 DMA 中断请求 196
表 9-3 各个通道的 DMA1 请求一览 197
表 9-4 各个通道的 DMA2 请求一览 198
表 9-5 各个信道的 DMA 弹性请求一览 199
表 9-6 DMA 寄存器映像和复位值 200
表 10-1 TMR6 和 TMR7– 寄存器图和复位值 217
表 10-2 计数方向与编码器信号的关系 241
表 10-3 TMRx – 寄存器图和复位值 250
表 10-4 TMRx 内部触发连接(1) 255
表 10-5 标准 OCx 通道的输出控制位 263
表 10-6 TMRx – 寄存器图和复位值 283
表 10-7 TMRx 内部触发连接（1） 286
表 10-8 标准 OCx 通道的输出控制位 291
表 10-9 TMRx – 寄存器图和复位值 293
表 10-10 标准 OCx 通道的输出控制位 299

表 10-11 计数方向与编码器信号的关系 328
表 10-12 TMR1、TMR8 – 寄存器图和复位值 334
表 10-13 TMRx 内部触发连接 340
表 10-14 带刹车功能的互补输出通道 OCx 和 OCxN 的控制位 349
表 11-1 WWDG 寄存器映像和复位值 358
表 11-2 看门狗超时时间（40kHz 的输入时钟（LSI））(1) 361
表 12-1 RTC-寄存器映像和复位值 368
表 13-1 ADC 引脚 375
表 13-2 模拟看门狗通道选择 377
表 13-3 ADC1 和 ADC2 用于规则通道的外部触发 380
表 13-4 ADC1 和 ADC2 用于注入通道的外部触发 381
表 13-5 ADC3 用于规则通道的外部触发 381
表 13-6 ADC3 用于注入通道的外部触发 381
表 13-7 ADC 中断 389
表 13-8 ADC 寄存器映像和复位值 389
表 14-1 DAC 引脚 402
表 14-2 外部触发 405
表 14-3 DAC 寄存器映射 411
表 15-1 SMBus 与 I2C 的比较 430
表 15-2 I2C 中断请求表 433
表 15-3 I²C 寄存器地址映像和复位值 434
表 16-1 检测噪声的数据采样 453
表 16-2 设置波特率时的误差计算 455
表 16-3 当 DIV_Decimal 等于 0 时，USART 接收器的容忍度 456
表 16-4 当 DIV_Decimal 不等于 0 时，USART 接收器的容忍度 456
表 16-5 帧格式 457
表 16-6 USART 中断请求 469
表 16-7 USART 模式设置 (1) 470
表 17-1 SPI 中断请求 497
表 17-2 使用系统时钟得到精确的音频频率 506
表 17-3 I2S 中断请求 511
表 17-4 SPI 寄存器列表及其复位值 511
表 18-1 发送邮箱寄存器列表 532
表 18-2 接收邮箱寄存器列表 532
表 18-3 537
表 19-1 NOR/PSRAM 存储块选择 558
表 19-2 外部存储器地址 558
表 19-3 存储器映射和时序寄存器 558
表 19-4 NAND 存储块选择 558
表 19-5 可编程的 NOR/PSRAM 访问参数 559

表 19-6 非复用信号的 NOR 闪存接口 560
表 19-7 复用 NOR 闪存接口 560
表 19-8 非复用信号的 PSRAM 接口 560
表 19-9 XMC 支持的 NOR 闪存/PSRAM 存储器和操作方式 561
表 19-10 XMC_BK1CTRLx 位域（模式 1） 563
表 19-11 XMC_BK1TMGx 位域（模式 1） 563
表 19-12 XMC_BK1CTRLx 位域（模式 A） 564
表 19-13 XMC_BK1TMGx 位域（模式 A） 564
表 19-14 XMC_BK1TMGWRx 位域（模式 A） 565
表 19-15 XMC_BK1CTRLx 位域（模式 2/B） 567
表 19-16 XMC_BK1TMGx 位域（模式 2/B） 567
表 19-17 XMC_BK1TMGWRx 位域（模式 2/B） 567
表 19-18 XMC_BK1CTRLx 位域（模式 C） 568
表 19-19 XMC_BK1TMGx 位域（模式 C） 569
表 19-20 XMC_BK1TMGWRx 位域（模式 C） 569
表 19-21 XMC_BK1CTRLx 位域（模式 D） 571
表 19-22 XMC_BK1TMGx 位域（模式 D） 571
表 19-23 XMC_BK1TMGWRx 位域（模式 D） 571
表 19-24 XMC_BK1CTRLx 位域（复用模式） 573
表 19-25 XMC_BK1TMGx 位域（复用模式） 573
表 19-26 XMC_BK1CTRLx 位域（同步模式） 575
表 19-27 XMC_BK1TMGx 位域（同步模式） 575
表 19-28 XMC_BK1CTRLx 位域（同步模式） 576
表 19-29 XMC_BK1TMGx 位域（同步模式） 576
表 19-30 可编程 NAND 操作参数 577
表 19-31 8 位 NAND 闪存 577
表 19-32 16 位 NAND 闪存 577
表 19-33 支持的存储器及其操作 578
表 19-34 XMC 寄存器地址映像 581
表 19-35 ECC 结果有效位 591
表 20-1 SDIO 引脚定义 595
表 20-2 命令格式 599
表 20-3 短响应格式 599
表 20-4 长响应格式 599
表 20-5 命令通道状态标志 599
表 20-6 数据令牌格式 602
表 20-7 发送 BUF 状态标志 602
表 20-8 接收 BUF 状态标志 602
表 20-9 卡状态 609
表 20-10 SD 状态 611
表 20-11 速度类型代码 612

表 20-12 移动性能代码 612
表 20-13 AU_SIZE 代码 613
表 20-14 最大的 AU 长度 613
表 20-15 ERASE_SIZE 代码 613
表 20-16 擦除超时代码 613
表 20-17 擦除偏移代码 614
表 20-18 基于块传输的写命令 615
表 20-19 基于块传输的写保护命令 616
表 20-20 擦除命令 616
表 20-21 I/O 模式命令 616
表 20-22 上锁命令 616
表 20-23 应用相关命令 617
表 20-24 R1 响应 617
表 20-25 R2 响应 618
表 20-26 R3 响应 618
表 20-27 R4 响应 618
表 20-28 R4b 响应 618
表 20-29 R5 响应 619
表 20-30 R6 响应 619
表 20-31 SDIO 寄存器映像 621
表 20-32 响应类型和 SDIO_RSPx 寄存器 625
表 21-1 分组缓冲区配置表 638
表 21-2 双缓冲批量端点缓冲区标识定义 641
表 21-3 双缓冲批量端点的缓冲区使用标识 641
表 21-4 同步端点的缓冲区使用标识 642
表 21-5 唤醒事件检测 643
表 21-6 接收状态编码 652
表 21-7 端点类型编码 652
表 21-8 端点特殊类型定义 652
表 21-9 发送状态编码 652
表 21-10 分组缓冲区大小的定义 654
表 22-1 灵活的跟踪引脚分配 657
表 22-2 MCUDBG 寄存器地址映像和复位值 657
表 23-1 ACC 中断请求 662
表 24-1 以太网模块引脚配置 669
表 24-2 管理帧格式 670
表 24-3 时钟范围 672
表 24-4 发送接口信号编码 673
表 24-5 接收接口信号编码 673
表 24-6 帧的状态 684
表 24-7 目的地址过滤器结果列表 687

表 24-8 源地址过滤器结果列表 688
表 24-9 接收描述符 0..... 711
表 24-10 以太网寄存器映像及其复位值 716

1 系统架构

AT32F403A 系列微控制器包括 ARM® Cortex™-M4F 处理器内核、总线架构、外设以及存储器构成。Cortex™-M4F 处理器是一种新时代的内核，拥有许多先进功能。对比于 Cortex™-M3，Cortex™-M4F 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，并且具有单精度（IEEE-754）浮点运算单元（FPU）。当设计中使用带 DSP 功能的 Cortex™-M4F 时就能格外节能，比软件解决方案更快，使 Cortex™-M4F 适用于那些要求微控制器提供高效能与低功耗的产品市场。

1.1 系统概述

AT32F403A 主系统基于 AHB 总线矩阵整合而成。AHB 总线矩阵是基于 AMBA3.0 AHB-LITE 的多层级 AHB，可使系统中多个主设备与从设备之间建立并行的访问路径，确保总线带宽的有效利用。AHB 总线矩阵包含五个主设备：Cortex™-M4F 内核的 ICode 总线、DCode 总线、系统总线，以及两个 DMA 控制器。总线中包含六个从设备：闪存控制器（FMC）、片上 SRAM、SDIO 控制器以及两个 APB 总线网桥。系统总线主要用于加载/储存数据以及调试访问系统存储区。系统存储区可划分为片上 SRAM 区、外部存储区以及外设映像区。

系统 AHB 总线与所有的 AHB 外设相连，此外还包含两条 AHB-APB 总线桥接器，这样可以在系统 AHB 总线以及两个 APB 总线之间实现完全同步连接。两条 APB 总线则与所有的 APB 外设相连。APB1/APB2 总线的最高速度限制为 120MHz，以上所述设备通过多层级 AHB 总线架构相互连接，如图 1-1 所示：

图 1-1 AT32F403A系列微控制器系统架构

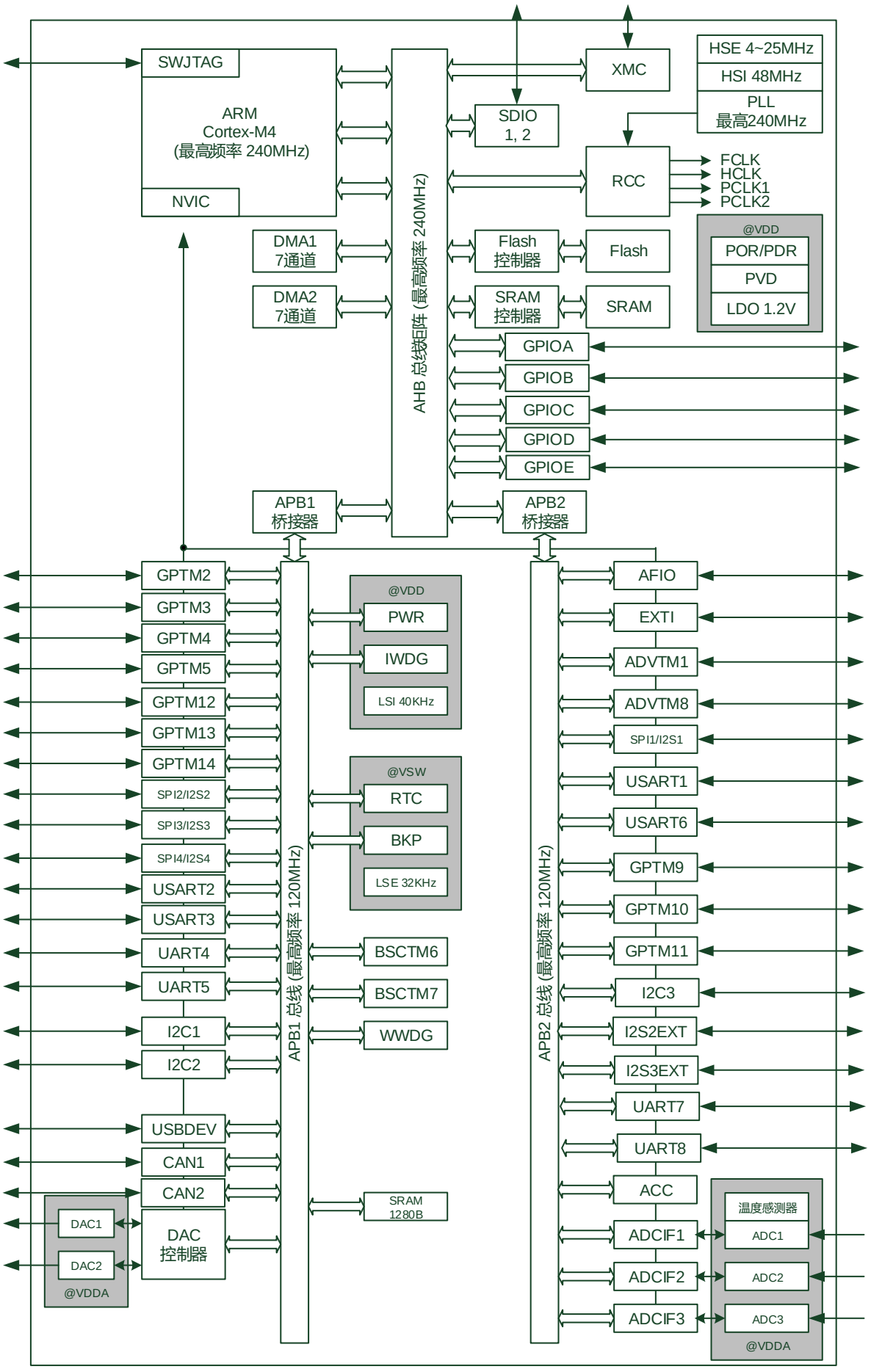
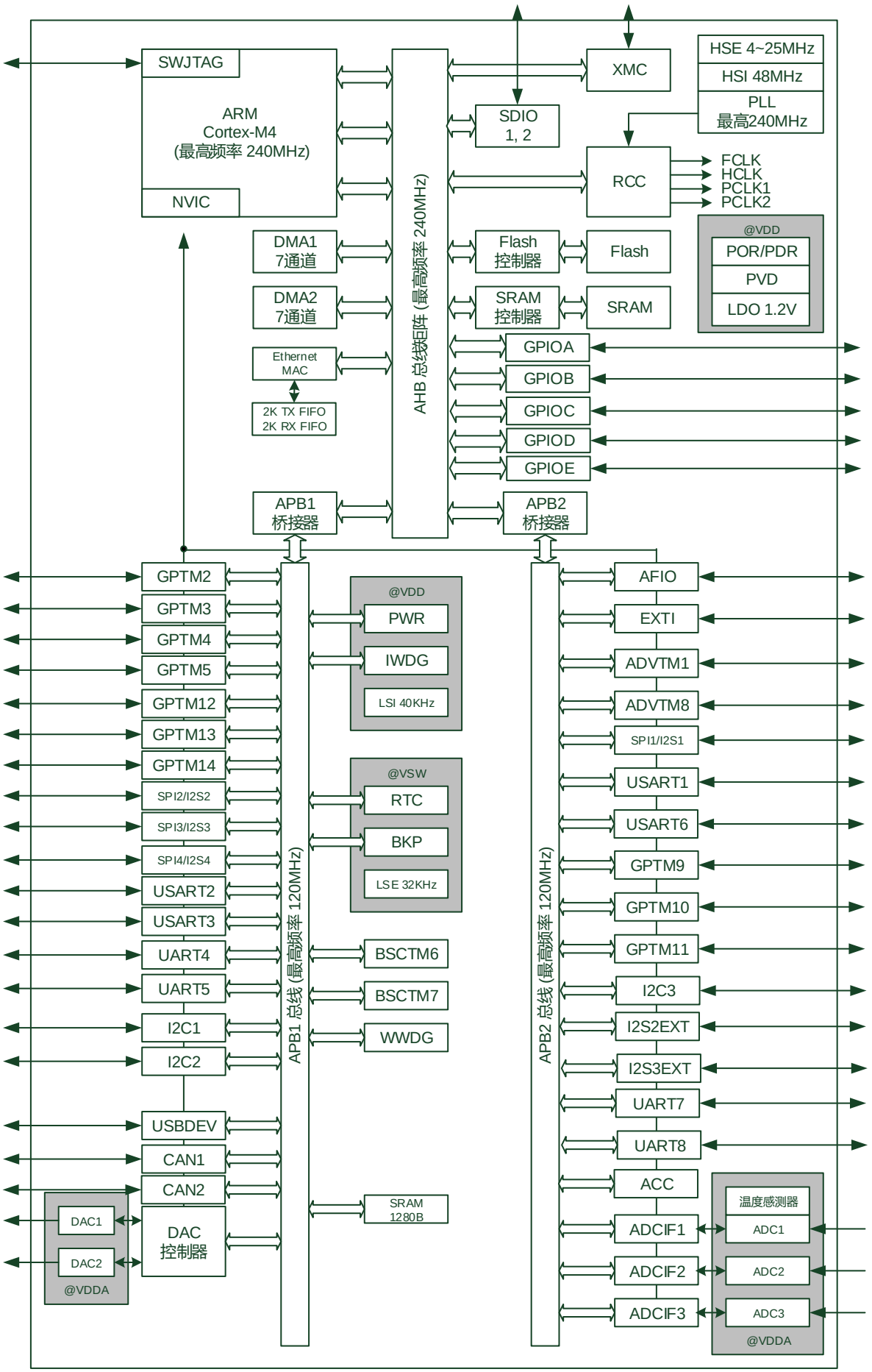

SWJTAG
ARM
Cortex-M4
(最高频率 240MHz)
NVIC
AHB 总线矩阵 (最高频率 240MHz)
XMC
SDIO
1, 2
HSE 4~25MHz
HSI 48MHz
PLL
最高240MHz
RCC
FCLK
HCLK
PCLK1
PCLK2
@VDD
POR/PDR
PVD
LDO 1.2V
DMA1
7通道
DMA2
7通道
Flash
控制器
Flash
SRAM
控制器
SRAM
GPIOA
GPIOB
GPIOC
GPIOD
GPIOE
APB1
桥接器
APB2
桥接器
GPTM2
GPTM3
GPTM4
GPTM5
GPTM12
GPTM13
GPTM14
SPI2/I2S2
SPI3/I2S3
SPI4/I2S4
USART2
USART3
UART4
UART5
I2C1
I2C2
USBDEV
CAN1
CAN2
DAC1
DAC2
@VDDA
DAC
控制器
APB1 总线 (最高频率 120MHz)
@VDD
PWR
IWDG
LSI 40KHz
@VSW
RTC
BKP
LSE 32KHz
BSCTM6
BSCTM7
WWDG
SRAM
1280B
AFIO
EXTI
ADVTM1
ADVTM8
SPI1/I2S1
USART1
USART6
GPTM9
GPTM10
GPTM11
I2C3
I2S2EXT
I2S3EXT
UART7
UART8
ACC
ADCIF1
ADCIF2
ADCIF3
APB2 总线 (最高频率 120MHz)
温度感测器
ADC1
ADC2
ADC3
@VDDA

图 1-2 AT32F407系列微控制器系统架构


SWJTAG
ARM
Cortex-M4
(最高频率 240MHz)
NVIC
AHB 总线矩阵 (最高频率 240MHz)
XMC
SDIO
1, 2
HSE 4~25MHz
HSI 48MHz
PLL
最高240MHz
RCC
FCLK
HCLK
PCLK1
PCLK2
@VDD
POR/PDR
PVD
LDO 1.2V
DMA1
7通道
DMA2
7通道
Flash
控制器
Flash
SRAM
控制器
SRAM
Ethernet
MAC
2K TX FIFO
2K RX FIFO
GPIOA
GPIOB
GPIOC
GPIOD
GPIOE
APB1
桥接器
APB2
桥接器
GPTM2
GPTM3
GPTM4
GPTM5
GPTM12
GPTM13
GPTM14
SPI2/I2S2
SPI3/I2S3
SPI4/I2S4
USART2
USART3
UART4
UART5
I2C1
I2C2
USBDEV
CAN1
CAN2
DAC1
DAC2
@VDDA
DAC
控制器
APB1 总线 (最高频率 120MHz)
@VDD
PWR
IWDG
LSI 40KHz
@VSW
RTC
BKP
LSE 32KHz
BSCTM6
BSCTM7
WWDG
SRAM
1280B
AFIO
EXTI
ADVTM1
ADVTM8
SPI1/I2S1
USART1
USART6
GPTM9
GPTM10
GPTM11
I2C3
I2S2EXT
I2S3EXT
UART7
UART8
ACC
ADCIF1
ADCIF2
ADCIF3
APB2 总线 (最高频率 120MHz)
温度感测器
ADC1
ADC2
ADC3
@VDDA

1.1.1 总线架构

ICode 总线

- 该总线将Cortex™-M4F内核的指令总线与闪存指令接口相连接。指令预取在此总线上完成。

DCode 总线

- 该总线将Cortex™-M4F内核的DCode总线与闪存存储器的数据接口相连接（常量加载和调试访问）。

系统总线

- 此总线连接Cortex™-M4F内核的系统总线（外设总线）到总线矩阵，总线矩阵协调着内核和DMA间的访问。

DMA 总线

- 此总线将DMA的AHB主控接口与总线矩阵相联，总线矩阵协调着CPU的DCode和DMA到SRAM、闪存和外设的访问。

总线矩阵

- 总线矩阵协调内核系统总线和DMA主控总线之间的访问仲裁，仲裁利用轮换算法。AHB外设通过总线矩阵与系统总线相连，允许DMA访问。

AHB/APB 桥接器（APB）

两个 AHB/APB 桥在 AHB 和 2 个 APB 总线间提供同步连接。APB1、APB2 操作速度限于 120MHz。有关连接到每个桥的不同外设的地址映像请参考 1.2 地址映射。在每一次复位以后，所有除 SRAM 和 FMC 以外的外设都被关闭，在使用一个外设之前，必须设置寄存器 RCC_AHBEN 来打开该外设的时钟。

注意：当对 APB 寄存器进行 8 位或者 16 位访问时，该访问会被自动转换成 32 位的访问，桥接器会自动将 8 位或者 32 位的数据扩展以配合 32 位的向量。

1.1.2 ARM Cortex™-M4F处理器

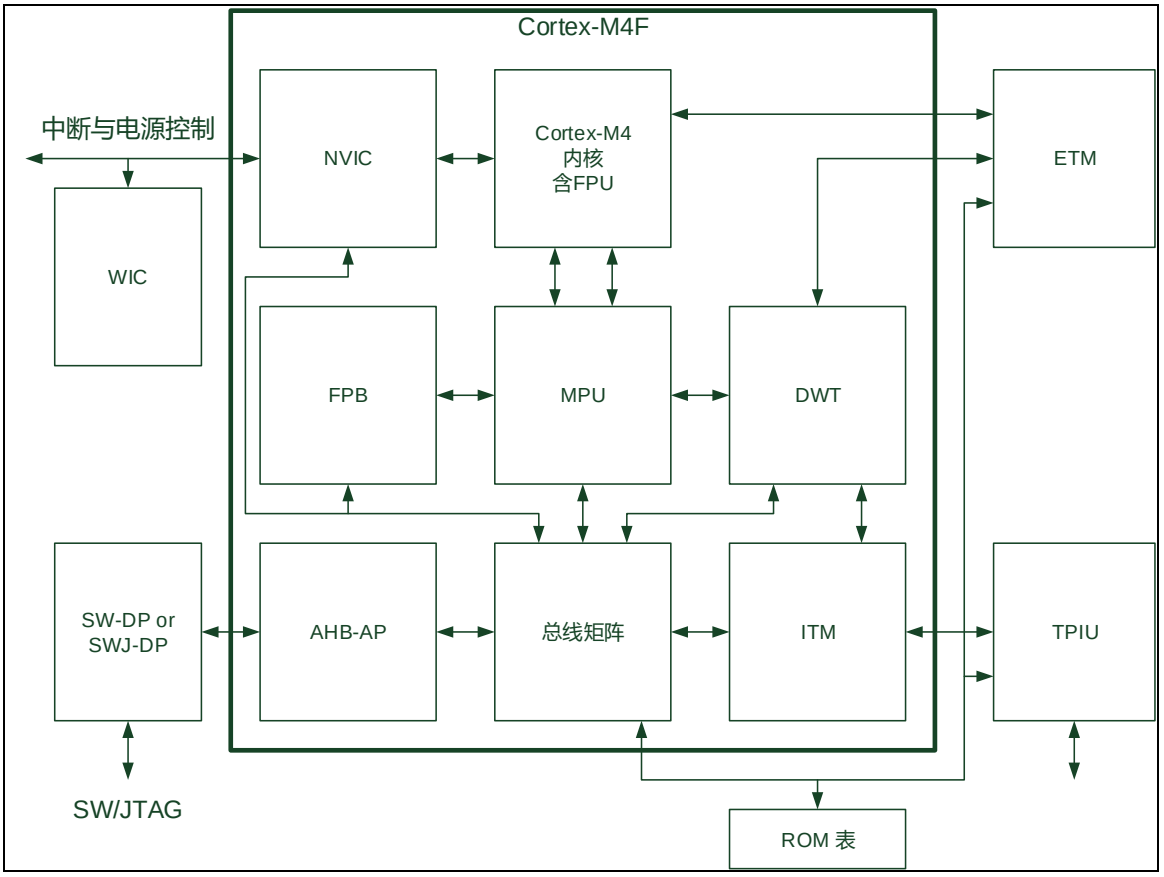
Cortex™-M4F 处理器是一款低功耗处理器，具有低门数，低中断延迟和低成本调试。支持包括 DSP 指令集与浮点运算功能，特别适合用于深度嵌入式应用程序需要快速中断响应功能。Cortex™-M4F 处理器是基于 ARMv7-M 架构，既支持 Thumb 指令集也支持 DSP 指令集。Cortex™-M4F 处理器并提供以下系统级外设：

内部总线矩阵，用于实现 ICode 总线、DCode 总线、系统总线、专用外设总线（PPB）以及调试专用总线（AHB-AP）的互联：

- 嵌套式向量型中断控制器（Nested Vectored Interrupt Controller，简写为NVIC）
- 闪存地址重载及断点单元（Flash Patch and Breakpoint，简写为FPB）
- 内存保护单元（Memory Protection Unit，简写为MPU）
- 数据观测点及跟踪单元（Data Watchpoint and Trace，简写为DWT）
- 跟踪仪器宏单元（Instrument Trace Macrocell，简写为ITM）
- 串行线JTAG 调试接口（Serial Wire JTAG Debug Port，简写为SWJ-DP）
- 内嵌跟踪宏单元（Embedded Trace Macrocell，简写为ETM）
- 跟踪端口接口单元（Trace Port Interface Unit，简写为TPIU）

下图为 Cortex™-M4F 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™-M4F 更详尽信息。

图 1-3 Cortex™-M4F内部框图

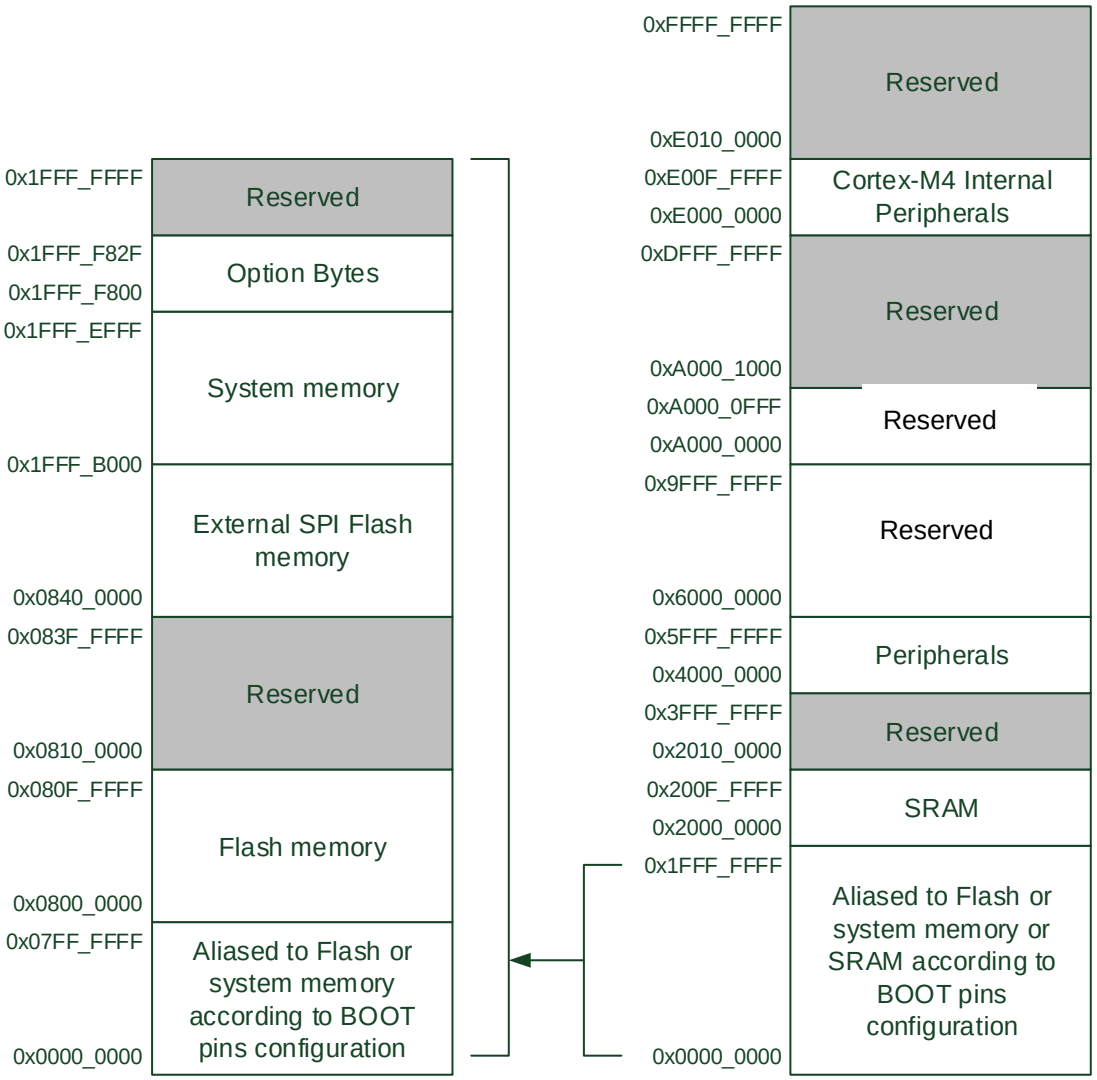


1.2 地址映射

程序存储器、数据存储器、寄存器和输入输出端口被组织在同一个 4GB 的线性地址空间内。数据字节以小端格式存放在存储器中。一个字里的最低地址字节被认为是该字的最低有效字节，而最高地址字节是最高有效字节。

外设寄存器的映像请参考相关章节。可访问的存储器空间被分成 8 个主要块，每个块为 512MB。其他所有没有分配给片上存储器和外设的存储器空间都是保留的地址空间，请参考相应器件的数据手册中的存储器映像图。

图 1-4 AT32F403A地址配置



1.2.1 寄存器映像

请参考相应器件的数据手册中的存储器映像图。表 1-1 寄存器组起始地址列出了所用 AT32F403A/407 中外设的起始地址。

表 1-1 寄存器组起始地址

起始地址	外设	总线	寄存器映像
0A000 1000 - 0xFFFF FFFF	保留	AHB	
0A000 0000 - 0xA000 0FFF	XMC_REG		参见 19.4 节
0x6000 0000 - 0x9FFF FFFF	XMC_MEM		参见 19.4 节
0x4002 A000 - 0x5FFF FFFF	保留		
0x4002 8000 - 0x4002 9FFF	ETHERNET		参见 24.8 以太网寄存器
0x4002 3400 - 0x4002 7FFF	SDIO2		参见 20.4 节
0x4002 3000 - 0x4002 33FF	CRC		参见 6.4 节
0x4002 2000 - 0x4002 23FF	闪存存储器接口（FMC）		参见 5.4 节
0x4002 1400 - 0x4002 1FFF	保留		
0x4002 1000 - 0x4002 13FF	复位和时钟控制（RCC）		参见 3.3 节

0x4002 0800 - 0x4002 0FFF	保留		
0x4002 0400 - 0x4002 07FF	DMA2		参见 9.4 节
0x4002 0000 - 0x4002 03FF	DMA1		参见 9.4 节
0x4001 8400 - 0x4001 7FFF	保留		
0x4001 8000 - 0x4001 83FF	SDIO		参见 20.4 节
0x4001 7400 - 0x4001 7FFF	保留		
0x4001 7000 - 0x4001 73FF	I2S3EXT		参见 17.4 节
0x4001 6C00 - 0x4001 6FFF	I2S2EXT		参见 17.4 节
0x4001 6800 - 0x4001 6BFF	UART8		参见 16.6 节
0x4001 6400 - 0x4001 67FF	UART7		参见 16.6 节
0x4001 6000 - 0x4001 63FF	USART6		参见 16.6 节
0x4001 5C00 - 0x4001 5FFF	I2C3		参见 15.4 节
0x4001 5800 - 0x4001 5BFF	ACC		参见 23.5 节
0x4001 5400 - 0x4001 57FF	TMR11 定时器		参见 10.3.5 节
0x4001 5000 - 0x4001 53FF	TMR10 定时器		参见 10.3.5 节
0x4001 4C00 - 0x4001 4FFF	TMR9 定时器		参见 10.3.4 节
0x4001 4400 - 0x4001 4BFF	保留		
0x4001 4000 - 0x4001 43FF	保留		
0x4001 3C00 - 0x4001 3FFF	ADC3		参见 13.4 节
0x4001 3800 - 0x4001 3BFF	USART1	APB2	参见 16.6 节
0x4001 3400 - 0x4001 37FF	TMR8 定时器		参见 10.4.4 节
0x4001 3000 - 0x4001 33FF	SPI1/I2S1		参见 17.4 节
0x4001 2C00 - 0x4001 2FFF	TMR1 定时器		参见 10.4.4 节
0x4001 2800 - 0x4001 2BFF	ADC2		参见 13.4 节
0x4001 2400 - 0x4001 27FF	ADC1		参见 13.4 节
0x4001 2000 - 0x4001 23FF	保留		
0x4001 1C00 - 0x4001 1FFF	保留		
0x4001 1800 - 0x4001 1BFF	GPIO 端口 E		参见 7.5 节
0x4001 1400 - 0x4001 17FF	GPIO 端口 D		参见 7.5 节
0x4001 1000 - 0x4001 13FF	GPIO 端口 C		参见 7.5 节
0X4001 0C00 - 0x4001 0FFF	GPIO 端口 B		参见 7.5 节
0x4001 0800 - 0x4001 0BFF	GPIO 端口 A		参见 7.5 节
0x4001 0400 - 0x4001 07FF	EXTI		参见 8.3 节
0x4001 0000 - 0x4001 03FF	AFIO		参见 7.5 节
0x4000 8400 - 0x4000 FFFF	保留		
0x4000 7800 - 0x4000 83FF	USB 1280 字节 SRAM（1）		参见 21.3.2 节
0x4000 7400 - 0x4000 77FF	DAC		
0x4000 7000 - 0x4000 73FF	电源控制（PWR）	APB1	参见 2.4 节
0x4000 6C00 - 0x4000 6FFF	后备寄存器（BKPR）		参见 4.4 节
0x4000 6800 - 0x4000 6BFF	bxCAN2		参见 18.4 节
0x4000 6400 - 0x4000 67FF	bxCAN1		参见 18.4 节

0x4000 6000 - 0x4000 63FF	USB 512 字节 SRAM (1)		参见 21.3.2 节
0x4000 5C00 - 0x4000 5FFF	USB 全速设备寄存器		参见 21.4 节
0x4000 5800 - 0x4000 5BFF	I2C2		参见 15.4 节
0x4000 5400 - 0x4000 57FF	I2C1		参见 15.4 节
0x4000 5000 - 0x4000 53FF	UART5		参见 16.6 节
0x4000 4C00 - 0x4000 4FFF	UART4		参见 16.6 节
0x4000 4800 - 0x4000 4BFF	USART3		参见 16.6 节
0x4000 4400 - 0x4000 47FF	USART2		参见 16.6 节
0x4000 4000 - 0x4000 43FF	SPI4/I2S4		参见 17.4 节
0x4000 3C00 - 0x4000 3FFF	SPI3/I2S3		参见 17.4 节
0x4000 3800 - 0x4000 3BFF	SPI2/I2S2		参见 17.4 节
0x4000 3400 - 0x4000 37FF	保留		
0x4000 3000 - 0x4000 33FF	独立看门狗（IWDG）		参见 11.2.4 节
0x4000 2C00 - 0x4000 2FFF	窗口看门狗（WWDG）		参见 11.1.6 节
0x4000 2800 - 0x4000 2BFF	RTC		参见 12.4 节
0x4000 2400 - 0x4000 27FF	保留		
0x4000 2000 - 0x4000 23FF	TMR14 定时器		参见 10.3.5 节
0x4000 1C00 - 0x4000 1FFF	TMR13 定时器		参见 10.3.5 节
0x4000 1800 - 0x4000 1BFF	TMR12 定时器		参见 10.3.5 节
0x4000 1400 - 0x4000 17FF	TMR7 定时器		参见 10.1.4 节
0x4000 1000 - 0x4000 13FF	TMR6 定时器		参见 10.1.4 节
0x4000 0C00 - 0x4000 0FFF	TMR5 定时器		参见 10.2.4 节
0x4000 0800 - 0x4000 0BFF	TMR4 定时器		参见 10.2.4 节
0x4000 0400 - 0x4000 07FF	TMR3 定时器		参见 10.2.4 节
0x4000 0000 - 0x4000 03FF	TMR2 定时器		参见 10.2.4 节

注意： *1、当 USB768B=0 SRAM 大小为 512 字节时，其地址范围为 0x4000 6000~0x4000 63FF。当 USB768B=1 SRAM 配置为 768~1280 字节时，其地址范围为 0x4000 7800~0x4000 83FF。CAN1 和 CAN2 都不使能时，USB 分组缓冲区最大可设定到 1280 字节，CAN1 或 CAN2 任意一个使能，USB 分组缓冲区最大可设定到 1024 字节，CAN1 和 CAN2 都使能时，USB 分组缓冲区最大可设定到 768 字节。*

2、仅 AT32F407 支持 ETHERNET 模块。

1.2.2 位绑定

Cortex™-M4F 存储器映像包括两个位段（bit-band）区。这两个位段区将别名存储器区中的每个字映射到位段存储器区的一个位，在别名存储区写入一个字具有对位段区的目标位执行读-改-写操作的相同效果。

在 AT32F403A/407 系列里，外设寄存器和 SRAM 都被映射到一个位段区里，这允许执行单一的位段的写和读操作。

下面的映射公式给出了别名区中的每个字是如何对应位带区的相应位的：

位别名地址 = 位绑定地址 + （字节偏移×32） + （位偏移×4）。

其中：

位别名地址：位别名存储器区中字的地址，它映像到某个目标位。

位绑定地址：位别名区的起始地址。

字节偏移：包含目标位的字节在位段里的序号。

位偏移：目标位所在位置（0-31）。

举例说明，若想对映射别名区中 SRAM 地址为 0x20000200 的字节中的位 2 作位级操作：

位别名地址 = 0x2200_0000 + （0x200×32） + （2×4） = 0x2200_4008

如果对 0x2200_4008 地址的写操作，与对 SRAM 中地址 0x2000_0200 字节的位 2 执行读-改-写操作有着相同的效果。如果对 0x2200_4008 地址进行读操作，将返回 SRAM 中地址 0x2000_0200 字节的位 2 的值（0x0000_0001 或 0x0000_0000）。请参阅《ARM®Cortex-M4 技术参考手册》了解关于位绑定更详尽信息。

1.2.3 片上SRAM

AT32F403A/407 系列内置 96K 字节的片上 SRAM，起始地址为 0x2000_0000。它可以以字节、半字（16 位）或全字（32 位）访问。AT32F403A/407 系列另外提供一个特别的模式能使片上 SRAM 扩充为 224K 字节，使用者可透过设定扩充的选择字节 EOPB 位 0 来使用此扩充模式(参阅 5.3.4 节)。在 224KB 扩充模式下，零等待延迟(zero wait state)的闪存容量限制为 128K 字节。

1.2.4 片上Flash

AT32F403A/407 系列提供最大 1024KB 的片上闪存，支持零等待延时的单周期 32 位读取操作。Flash 存储器的组织可划分为主存储块与信息块：主存储块用于存放应用程序代码，可按字节（8 位对齐）访问、半字节（16 位对齐）、全字（32 位对齐）访问。存储器可单次做一个字节（8 位）、半字节（16 位）或一个全字（32 位）编程。主存储块由每页为 2KB（256K 及以上容量）的分页组成，每个分页都可以个别抹除，闪存存储器也支持单次全片抹除（但不会抹除信息块）。信息块则包含系统存储器以及选项字节两个区段。系统存储器用于存储引导加载程序，于出厂时已编程，用户不可更改。而选项字节则存放用户可修改的选项。详细组织可参阅下表。

闪存存储器由闪存控制器操作，该控制器提供预取缓冲、选项字节加载、闪存编程与抹除以及读取/写入保护等功能。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。

闪存模块组织（1024K）

块		名称	地址范围	长度（字节）
主存储器	块 1 (Bank1) 512KB	页 0	0x0800 0000 – 0x0800 07FF	2K
		页 1	0x0800 0800 – 0x0800 0FFF	2K
		页 2	0x0800 1000 – 0x0800 17FF	2K
		页 3	0x0800 1800 – 0x0800 1FFF	2K
		页 4	0x0800 2000 – 0x0800 27FF	2K
		. .	. .	. .
		页 255	0x0807 F800 – 0x0807 FFFF	2K
	块 2 (Bank2) 512KB	页 256	0x0808 0000 – 0x0808 07FF	2K
		页 257	0x0808 0800 – 0x0808 0FFF	2K
		页 258	0x0808 1000 – 0x0808 17FF	2K
		页 259	0x0808 1800 – 0x0808 1FFF	2K
		页 260	0x0808 2000 – 0x0808 27FF	2K
		. .	. .	. .
		页 511	0x080F F800 – 0x080F FFFF	2K
外部存储器	块 3 (Bank3)		0x0840 0000 – 0x093F FFFF	16M
信息块		启动程序代码	0x1FFF B000 – 0x1FFF EFFF	16K
		用户选择字节	0x1FFF F800 – 0x1FFF F82F	48
闪存存储器		FLASH_ACR	0x4002 2000 – 0x4002 2003	4

接口寄存器	FLASH_FCKEY	0x4002 2004 – 0x4002 2007	4
	FLASH_OPTKEYR	0x4002 2008 – 0x4002 200B	4
	FLASH_STS	0x4002 200C – 0x4002 200F	4
	FLASH_CTRL	0x4002 2010 – 0x4002 2013	4
	FLASH_ADDR	0x4002 2014 – 0x4002 2017	4
	保留	0x4002 2018 – 0x4002 201B	4
	FLASH_UOB	0x4002 201C – 0x4002 201F	4
	FLASH_WRPRT	0x4002 2020 – 0x4002 2023	4
	保留	0x4002 2024 - 0x4002 2043	32
	FLASH_FCKEY2	0x4002 2044 - 0x4002 2047	4
	保留	0x4002 2048 - 0x4002 204B	4
	FLASH_STS2	0x4002 204C - 0x4002 204F	4
	FLASH_CTRL2	0x4002 2050 - 0x4002 2053	4
	FLASH_ADDR2	0x4002 2054 - 0x4002 2057	4
	保留	0x4002 2058 - 0x4002 2083	44
	FLASH_FCKEY3	0x4002 2084 - 0x4002 2087	4
	FLASH_SELECT	0x4002 2088 - 0x4002 208B	4
	FLASH_STS3	0x4002 208C - 0x4002 208F	4
	FLASH_CTRL3	0x4002 2090 - 0x4002 2093	4
	FLASH_ADDR3	0x4002 2094 - 0x4002 2097	4
	FLASH_DA	0x4002 2098 – 0x4002 209B	4
	保留	0x4002 209C - 0x4002 20CB	48
	SLIB_CDR0	0x4002 20CC - 0x4002 20CF	4
	SLIB_CDR1	0x4002 20D0 - 0x4002 20D3	4
	SLIB_PSW	0x4002 20D4 - 0x4002 20D7	4
	SLIB_PSW_STS	0x4002 20D8 - 0x4002 20DB	4
	SLIB_SET_PSW	0x4002 20DC - 0x4002 20DF	4
	SLIB_SET_RANGE	0x4002 20E0 - 0x4002 20E3	4
	保留	0x4002 20E4 - 0x4002 20EF	12
	SLIB_KEYR	0x4002 20F0 - 0x4002 20F3	4
	CRC_DR	0x4002 20F4 - 0x4002 20F7	4
	CRC_OUTR	0x4002 20F8 - 0x4002 2FB	4

闪存模块组织（512K）

块		名称	地址范围	长度（字节）
主存储器	块 1 (Bank1) 512KB	页 0	0x0800 0000 – 0x0800 07FF	2K
		页 1	0x0800 0800 – 0x0800 0FFF	2K
		页 2	0x0800 1000 – 0x0800 17FF	2K
		页 3	0x0800 1800 – 0x0800 1FFF	2K
		页 4	0x0800 2000 – 0x0800 27FF	2K
		. .	. .	. .
		页 255	0x0807 F800 – 0x0807 FFFF	2K
外部存储器	块 3 (Bank3)		0x0840 0000 – 0x093F FFFF	16M
信息块		启动程序代码	0x1FFF B000 – 0x1FFF EFFF	16K
		用户选择字节	0x1FFF F800 – 0x1FFF F82F	48
闪存存储器接口寄存器		FLASH_ACR	0x4002 2000 – 0x4002 2003	4
		FLASH_FCKEY	0x4002 2004 – 0x4002 2007	4

	FLASH_OPTKEYR	0x4002 2008 – 0x4002 200B	4
	FLASH_STS	0x4002 200C – 0x4002 200F	4
	FLASH_CTRL	0x4002 2010 – 0x4002 2013	4
	FLASH_ADDR	0x4002 2014 – 0x4002 2017	4
	保留	0x4002 2018 – 0x4002 201B	4
	FLASH_UOB	0x4002 201C – 0x4002 201F	4
	FLASH_WRPRT	0x4002 2020 – 0x4002 2023	4
	保留	0x4002 2024 - 0x4002 2083	96
	FLASH_FCKEY3	0x4002 2084 - 0x4002 2087	4
	FLASH_SELECT	0x4002 2088 - 0x4002 208B	4
	FLASH_STS3	0x4002 208C - 0x4002 208F	4
	FLASH_CTRL3	0x4002 2090 - 0x4002 2093	4
	FLASH_ADDR3	0x4002 2094 - 0x4002 2097	4
	FLASH_DA	0x4002 2098 – 0x4002 209B	4
	保留	0x4002 209C - 0x4002 20CB	48
	SLIB_CDR0	0x4002 20CC - 0x4002 20CF	4
	SLIB_CDR1	0x4002 20D0 - 0x4002 20D3	4
	SLIB_PSW	0x4002 20D4 - 0x4002 20D7	4
	SLIB_PSW_STS	0x4002 20D8 - 0x4002 20DB	4
	SLIB_SET_PSW	0x4002 20DC - 0x4002 20DF	4
	SLIB_SET_RANGE	0x4002 20E0 - 0x4002 20E3	4
	保留	0x4002 20E4 - 0x4002 20EF	12
	SLIB_KEYR	0x4002 20F0 - 0x4002 20F3	4
	CRC_DR	0x4002 20F4 - 0x4002 20F7	4
	CRC_OUTR	0x4002 20F8 - 0x4002 2FB	4

闪存模块组织（256K）

块		名称	地址范围	长度（字节）
主存储器	块 1 (Bank1) 256KB	页 0	0x0800 0000 – 0x0800 07FF	2K
		页 1	0x0800 0800 – 0x0800 0FFF	2K
		页 2	0x0800 1000 – 0x0800 17FF	2K
		页 3	0x0800 1800 – 0x0800 1FFF	2K
		页 4	0x0800 2000 – 0x0800 27FF	2K
		. .	. .	. .
		页 127	0x0803 F800 – 0x0803 FFFF	2K
外部存储器	块 3 (Bank3)		0x0840 0000 – 0x093F FFFF	16M
信息块		启动程序代码	0x1FFF B000 – 0x1FFF EFFF	16K
		用户选择字节	0x1FFF F800 – 0x1FFF F82F	48
闪存存储器 接口寄存器		FLASH_ACR	0x4002 2000 – 0x4002 2003	4
		FLASH_FCKEY	0x4002 2004 – 0x4002 2007	4
		FLASH_OPTKEYR	0x4002 2008 – 0x4002 200B	4
		FLASH_STS	0x4002 200C – 0x4002 200F	4
		FLASH_CTRL	0x4002 2010 – 0x4002 2013	4
		FLASH_ADDR	0x4002 2014 – 0x4002 2017	4
		保留	0x4002 2018 – 0x4002 201B	4
		FLASH_UOB	0x4002 201C – 0x4002 201F	4
		FLASH_WRPRT	0x4002 2020 – 0x4002 2023	4

保留	0x4002 2024 - 0x4002 2083	96
FLASH_FCKEY3	0x4002 2084 - 0x4002 2087	4
FLASH_SELECT	0x4002 2088 - 0x4002 208B	4
FLASH_STS3	0x4002 208C - 0x4002 208F	4
FLASH_CTRL3	0x4002 2090 - 0x4002 2093	4
FLASH_ADDR3	0x4002 2094 - 0x4002 2097	4
FLASH_DA	0x4002 2098 – 0x4002 209B	4
保留	0x4002 209C - 0x4002 20CB	48
SLIB_CDR0	0x4002 20CC - 0x4002 20CF	4
SLIB_CDR1	0x4002 20D0 - 0x4002 20D3	4
SLIB_PSW	0x4002 20D4 - 0x4002 20D7	4
SLIB_PSW_STS	0x4002 20D8 - 0x4002 20DB	4
SLIB_SET_PSW	0x4002 20DC - 0x4002 20DF	4
SLIB_SET_RANGE	0x4002 20E0 - 0x4002 20E3	4
保留	0x4002 20E4 - 0x4002 20EF	12
SLIB_KEYR	0x4002 20F0 - 0x4002 20F3	4
CRC_DR	0x4002 20F4 - 0x4002 20F7	4
CRC_OUTR	0x4002 20F8 - 0x4002 2FB	4

1.3 引导配置

在 AT32F403A/407 里，可以通过 BOOT[1：0]引脚选择三种不同引导模式，如下表所示。

表 1-2 启动模式

启动模式	说明	引脚配置	
		BOOT1	**BOOT0**
主闪存存储器	主闪存存储器被选为启动区域	X	0
系统存储器	系统存储器被选为启动区域	0	1
片上 SRAM	内置 SRAM 被选为启动区域	1	1

在系统复位后，BOOT 引脚的值将被锁存。用户可以通过设置 BOOT1 和 BOOT0 引脚的状态，来选择在复位后的启动模式。从待机模式退出时，BOOT 引脚的值将被被重新锁存；因此，在待机模式下 BOOT 引脚应保持为需要的启动配置。

根据选定的启动模式，主闪存存储器、系统存储器或 SRAM 可以按照以下方式访问：

- 从主闪存存储器启动：主闪存存储器被映射到启动空间（0x0000 0000），但仍然能够在它原有的地址（0x0800 0000）访问它，即闪存存储器的内容可以在两个地址区域访问，0x0000 0000 或 0x0800 0000。
- 从系统存储器启动：系统存储器被映射到启动空间（0x0000 0000），但仍然能够在它原有的地址（0x1FFF B000）访问它。
- 从内置 SRAM 启动：只能在 0x2000 0000 开始的地址区访问 SRAM。

在启动延迟之后，CPU 从地址 0x0000 0000 获取堆栈顶的地址，并从启动存储器的 0x0000_0004 指示的地址开始执行代码。因为固定的存储器映像，代码区始终从地址 0x0000_0000 开始（通过 ICode 和 DCode 总线访问），而数据区（SRAM）始终从地址 0x2000_0000 开始（通过系统总线访问）。Cortex™-M4F 的 CPU 始终从 ICode 总线获取复位向量，即启动仅适合于从代码区开始（典型地从 Flash 启动）。

AT32F403A/407 系列微控制器实现了一个特殊的机制，可以从片上 SRAM 启动。当从片上 SRAM 启动，在应用程序的初始化代码中，必须使用 NVIC 的异常表和偏移寄存器，重新映射向量表之 SRAM 中。

系统存储器中包含内嵌的引导加载程序，可用于对闪存存储器编程。引导加载程序是通过 USART1 接口对闪存存储器进行重新编成。

1.4 器件特征信息

1.4.1 寄存器的缩写说明

本手册中对寄存器的描述中将使用下述缩写：

表 1-3 寄存器缩写表

读/写（rw）	软件可以读或写这些位。
只读（r）	软件只能读这些位。
只写（w）	软件只能写这些位；如果读这些位，则返回它们的复位值。
读/清除（rc_w0）	软件可以读并写'0'清除这些位，写'1'将不对该位产生影响。
读/设置（rs）	软件可以读并写'1'设置这些位，写'0'将不对该位产生影响。
保留（res）	保留位，必须保持在复位状态。

1.4.2 闪存容量寄存器

闪存容量寄存器提通该芯片闪存容量信息，用户可透过该寄存器取得闪存容量。

基地址：0x1FFF F7E0

复位值：0xXXXX （由生产厂设置）

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
F_SIZE															
r															

位15：0	**F_SIZE**：闪存容量，以 KByte 为单位 例如：0x0080 = 128KByte

1.4.3 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），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

基地址：0x1FFF F7E8

复位值：0xXXXX XXXX （由生产厂设置）

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
U_ID[31：16]															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
U_ID[15：0]															
r															

基地址：0x1FFF F7EC

复位值：0xXXXX XXXX （由生产厂设置）

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
U_ID[63：48]															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
U_ID[47：32]															
r															

基地址：0x1FFF F7F0

复位值：0xXXXX XXXX （由生产厂设置）

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
U_ID[95：80]															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
U_ID[79：64]															
r															

2 电源控制（PWR）

2.1 简介

功耗是 AT32F403A/407 系列设备中最重要的问题之一，AT32F403A/407 系列设备可工作电压范围为 2.6V 至 3.6V，且可在-40~+105℃温度范围内正常工作。为了减少功耗，且使得应用程序可以在 CPU 运行时间要求、速度和功耗的相互冲突中获得最佳折衷，电源控制中提供了三种省电模式，包括睡眠模式，停止模式和待机模式。AT32F403A/407 系列设备有三个电源域，包括 VDD/VDDA 域，1.2V 域和备份域。VDD/VDDA 域由电源直接供电，但 VDDA 和 VSSA 提供分离的模拟模块电源，降低电源噪声干扰。在 VDD/VDDA 域中嵌入了一个 LDO，用来为 1.2V 域供电。

2.2 主要特点

具备三个电源：VDD/VDDA 域、内核域和备份域。

支持三种省电模式：睡眠模式、停止模式和待机模式。

内建电压调节器提供 1.2V 给内核域。

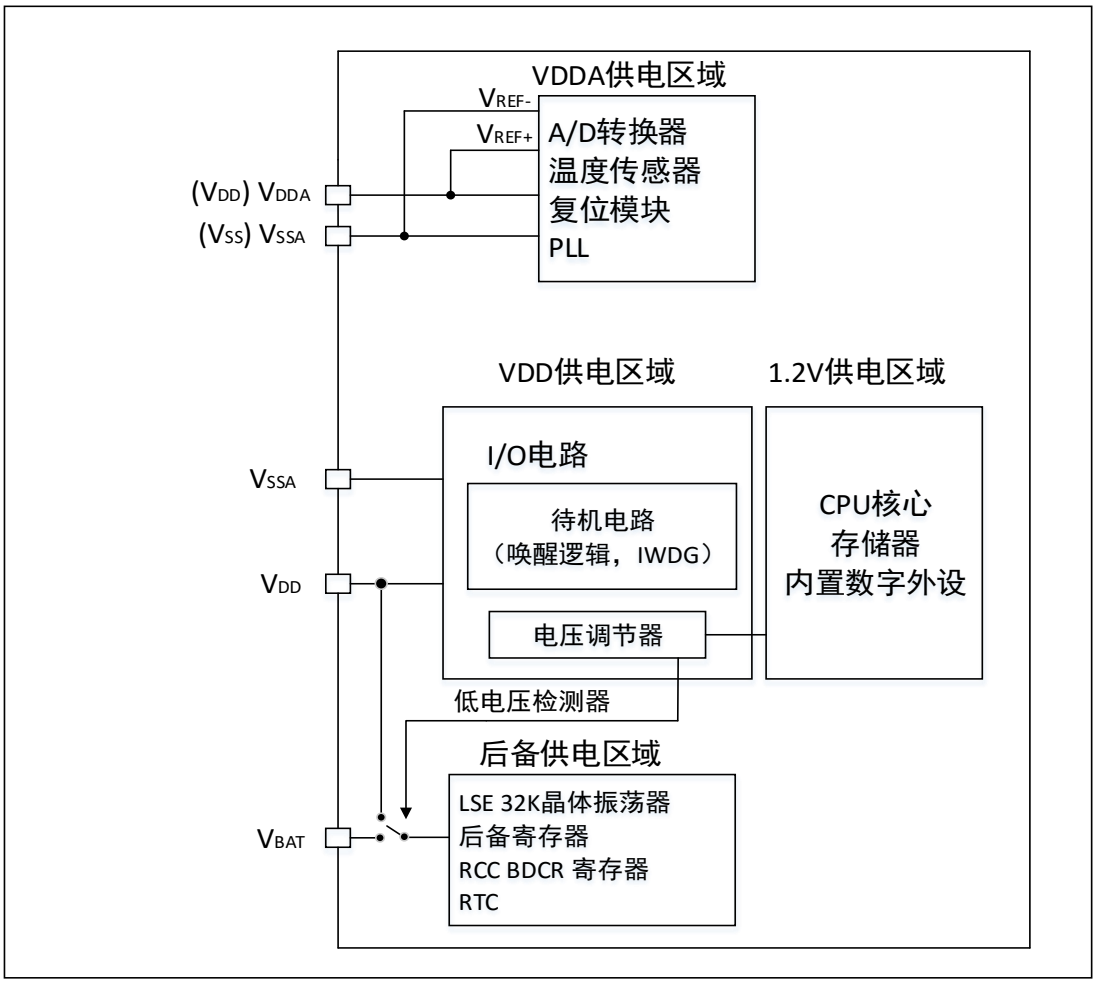
提供电压检测器，能于电压低于阀值时发出中断。

2.3 功能描述

2.3.1 电源域

AT32F403A/407 的工作电压（VDD）为 2.6～3.6V。通过内置的电压调节器提供所需的 1.2V 电源。

图 2-1 各电源域框图



2.3.1.1 VDD/VDDA电源域

AT32F403A/407 的工作电压（VDD）为 2.6～3.6V。通过内置的电压调节器提供所需的 1.2V 电源。

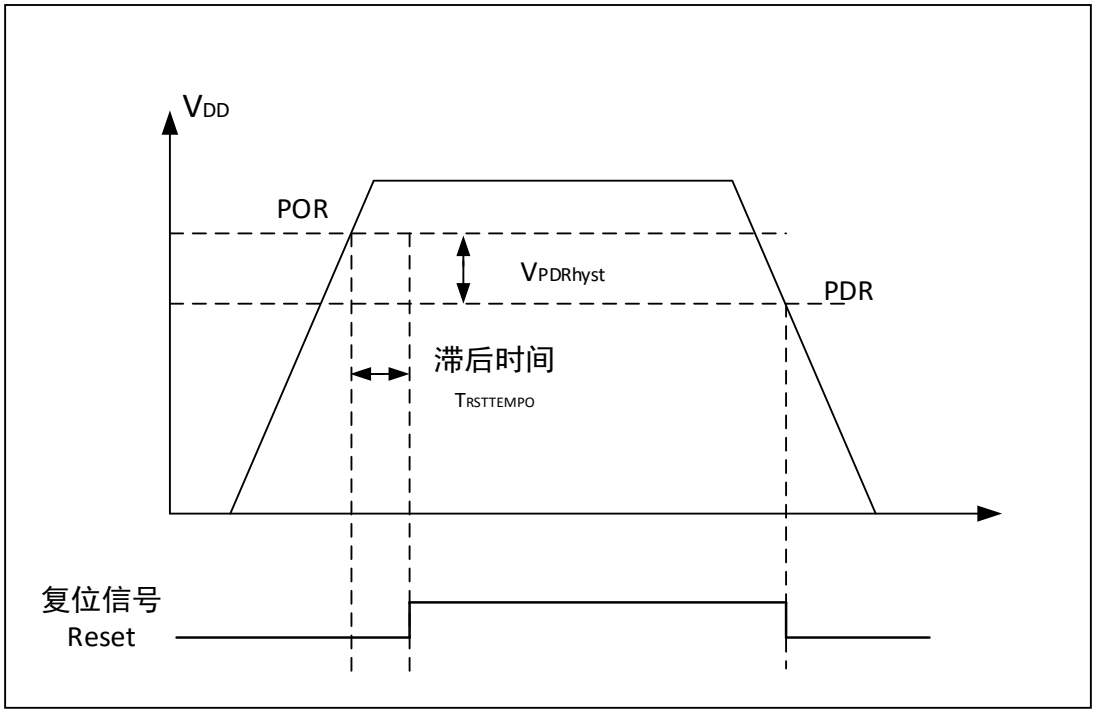
为了提高转换的精确度，ADC 使用一个独立的电源供电，过滤和屏蔽来自印刷电路板上的毛刺干扰。

ADC 的电源引脚为 VDDA，独立的电源地 VSSA。

VREF+和 VREF-引脚，他们在芯片内部与 ADC 的电源（VDDA）和地（VSSA）相连。

AT32F403A/407 内部有一个完整的上电复位（POR）和掉电复位（PDR）电路，当供电电压达到 2.6V 时系统既能正常工作。当 VDD/VDDA 低于指定的限位电压 VPOR/VPDR 时，系统保持为复位状态，而无需外部复位电路。关于上电复位和掉电复位的细节请参考数据手册的电气特性部分。

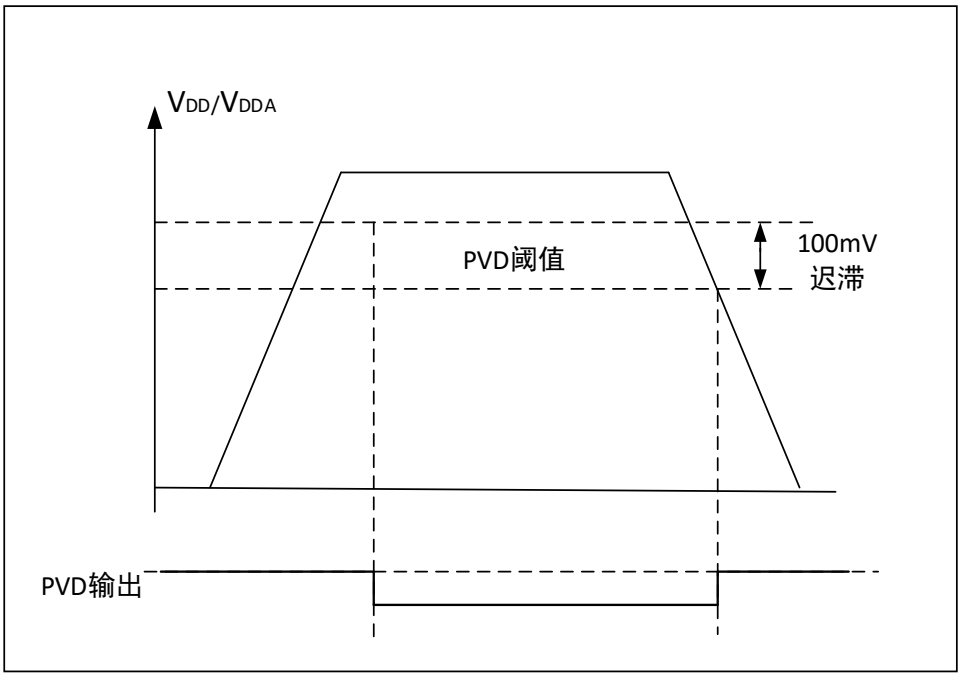
图 2-2 上电复位和掉电复位的波形图



AT32F403A/407 提供一个程序电压侦测器 PVD，用户可以利用 PVD 对 VDD 电压与电源控制寄存器（PWR_CTRL）中的 PVDS[2：0]位进行比较来监控电源，选择监控电压的阀值。

通过设置 PVDEN 位来使能 PVD。电源控制/状态寄存器（PWR_CTRLSTS）中的 PVD 标志用来表明 VDD 是高于还是低于 PVD 的电压阀值。该事件在内部连接到外部中断的第 16 线，如果该中断在外部中断寄存器中是使能的，该事件就会产生中断。当 VDD 下降到 PVD 阀值以下和（或）当 VDD 上升到 PVD 阀值之上时，根据外部中断第 16 线的上升/下降边沿触发设置，就会产生 PVD 中断。例如，这一特性可用于执行紧急关闭任务。

图2-1 PVD的阀值与输出



2.3.1.2 内核电源域

内核电源域包含 CPU 核心、内存与内置数字外设。该电源域由一电压调节器供电。调节器于复位后总是使能的。根据应用方式它以 3 种不同的模式工作。

运转模式：调节器以正常功耗模式提供 1.2V 电源（内核，内存和外设）。

停止模式：调节器以低功耗模式提供 1.2V 电源，以保存寄存器和 SRAM 的内容。

待机模式：调节器停止供电。除了备用电路和备份域外，寄存器和 SRAM 的内容全部丢失。

电池备份域

VBAT 脚也为 RTC、LSE 振荡器和 PC13 至 PC15 供电。

注意： *在 VDD 上升阶段（$t_{RSTTEMPO}$）或者探测到 PVD 之后，VBAT 和 VDD 之间的电源开关仍会保持连接在 VBAT。在 VDD 上升阶段，如果 VDD 在小于 $t_{RSTTEMPO}$ 的时间内达到稳定状态（关于 $t_{RSTTEMPO}$ 可参考数据手册中的相关部分），且 VDD>VBAT + 0.6V 时，电流可能通过 VDD 和 VBAT 之间的内部二极管注入到 VBAT。*

如果与 VBAT 连接的电源或者电池不能承受这样的注入电流，强烈建议在外部 VBAT 和电源之间连接一个低压降二极管。

建议 VBAT 在外部通过一个 100nF 的陶瓷电容与 VDD 相连。

当备份区域由 VDD（内部模拟开关连到 VDD）供电时，下述功能可用：

- PC14和PC15可以用于GPIO或LSE引脚；
- PC13可以作为通用I/O口、TAMPER引脚、RTC校准时钟、RTC闹钟或秒输出（参见第4章备份寄存器（BKPR））。

注意： *因为模拟开关只能通过少量的电流（3mA），在输出模式下使用 PC13 至 PC15 的 I/O 口功能是有限制的：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（如驱动 LED）。*

后备区域由 VBAT 供电时可以使用下述功能：

- PC14和PC15只能用于LSE引脚
- PC13可以作为TAMPER引脚、RTC闹钟或秒输出（参见第4.4.2节： RTC时钟校准寄存器（BKPR_RTCCAL））。

2.3.2 低功耗模式

在系统或电源复位以后，微控制器处于运行状态。当 CPU 不需继续运行时，提供了多种低功耗模式来节省功耗，例如等待某个外部事件时。用户需要根据最低电源消耗、最快速启动时间和可用的唤醒源等条件，选定一个最佳的低功耗模式。

AT32F403A/407 有三种低功耗模式：

- 睡眠模式（Cortex™-M4F内核停止，所有外设（包括Cortex™-M4F核心的外设，如NVIC、系统时钟（SysTick）等）仍在运行）；
- 停止模式（所有的时钟都已停止，SysTick除外）；
- 待机模式（1.2V电源关闭）。

模式	进入	唤醒	对 1.2V 区域时钟的影响	对 VDD 区域时钟的影响	电压调节器
睡眠（SLP-NOW 或 SLP-ON-EXIT）	WFI	任一中断	CPU 时钟关闭，对其他外设时钟和 ADC 时钟无影响	无	开
	WFE	唤醒事件			
停机	PDDS 位 +SLEEPDEEP 位 +WFI 或 WFE	任一外部中断（在外部中断寄存器中设置）	关闭所有 1.2V 区域的时钟	HSI 和 HSE 的振荡器关闭	开启或处于低功耗模式（依据电源控制寄存器（PWR_CTRL）的设定）

待机	PDDS 位 +SLEEPDEEP 位 +WFI 或 WFE	WKUP 引脚的上升沿、RTC 闹钟事件、NRST 引脚上的外部复位、IWDG 复位			关

此外，在运行模式下，可以通过以下方式中的一种降低功耗：

- 降低系统时钟。在运行模式下，通过对预分频寄存器进行编程，可以降低任意一个系统时钟（SYSCLK、HCLK、PCLK1、PCLK2）的速度。进入睡眠模式前，也可以利用预分频器来降低外设的时钟。详见第3.3.2节：时钟配置寄存器（RCC_CFG）。
- 关闭APB和AHB总线上未被使用的外设时钟。在运行模式下，任何时候都可以通过停止为外设和内存提供时钟（HCLK和PCLKx）来减少功耗。为了在睡眠模式下更多地减少功耗，可在执行WFI或WFE指令前关闭所有外设的时钟。通过设置AHB外设时钟使能寄存器（RCC_AHBEN）、APB2外设时钟使能寄存器（RCC_APB2EN）和APB1外设时钟使能寄存器（RCC_APB1EN）来开关各个外设模块的时钟。

2.3.2.1 睡眠模式

进入睡眠模式：

通过执行 WFI 或 WFE 指令进入睡眠状态。根据 Cortex™-M4F 系统控制寄存器中的 SLEEPONEXIT 位的值，有两种选项可用于选择睡眠模式进入机制：

- SLP-NOW：如果SLEEPONEXIT位被清除，当WFI或WFE被执行时，微控制器立即进入睡眠模式。
- SLP-ON-EXIT：如果SLEEPONEXIT位被置位，系统从最低优先级的中断处理程序中退出时，微控制器就立即进入睡眠模式。

在睡眠模式下，所有的 I/O 引脚都保持它们在运行模式时的状态。关于如何进入睡眠模式，更多的细节参考表 2-1 和表 2-2。

退出睡眠模式：

如果执行 WFI 指令进入睡眠模式，任意一个被嵌套向量中断控制器响应的外设中断都能将系统从睡眠模式唤醒。

如果执行 WFE 指令进入睡眠模式，则一旦发生唤醒事件时，微处理器都将从睡眠模式退出。唤醒事件可以通过下述方式产生：

- 在外设控制寄存器中使能一个中断，而不是在NVIC（嵌套向量中断控制器）中使能，并且在Cortex™-M4F系统控制寄存器中使能SEVONPEND位。当MCU从WFE中唤醒后，外设的中断挂起位和外设的NVIC中断通道挂起位（在NVIC中断清除挂起寄存器中）必须被清除。
- 配置一个外部或内部的EXTI线为事件模式。当MCU从WFE中唤醒后，因为与事件线对应的挂起位未被设置，不必清除外设的中断挂起位或外设的NVIC中断通道挂起位。

该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。关于如何退出睡眠模式，更多的细节参考表 2-1 SLP-NOW 模式和表 2-2 SLP-ON-EXIT 模式。

表 2-1 SLP-NOW模式

SLP-NOW 模式	说明
进入	在以下条件下执行 WFI（等待中断）或 WFE（等待事件）指令： – SLEEPDEEP = 0 和 – SLEEPONEXIT = 0 参考 Cortex™-M4F 系统控制寄存器。

退出	如果执行 WFI 进入睡眠模式： 中断：参考中断向量表（表 8-1） 如果执行 WFE 进入睡眠模式： 唤醒事件：参考唤醒事件管理（第 8.2.3 节）
唤醒延时	无

表 2-2 SLP-ON-EXIT模式

SLP-ON_EXIT 模式	说明
进入	在以下条件下执行 WFI 指令： – SLEEPDEEP = 0 和 – SLEEPONEXIT = 1 参考 Cortex™-M4F 系统控制寄存器
退出	中断：参考中断向量表（表 8-1）
唤醒延时	无

2.3.2.2 停止模式

停止模式是在 Cortex™-M4F 的深睡眠模式基础上结合了外设的时钟控制机制，在停止模式下电压调节器可运行在正常或低功耗模式。此时在 1.2V 供电区域的的所有时钟都被停止，SysTick 除外，PLL、HSI 和 HSE RC 振荡器的功能被禁止，SRAM 和寄存器内容被保留下来。

在停止模式下，所有的 I/O 引脚都保持它们在运行模式时的状态。

进入停止模式

关于如何进入停止模式，详见表 2-3。

如果正在进行闪存编程，直到对内存访问完成，系统才进入停止模式。如果正在进行对 APB 的访问，直到对 APB 访问完成，系统才进入停止模式。可以通过对独立的控制位进行编程，可选择以下功能：

- 独立看门狗（IWDG）：可通过写入看门狗的键寄存器或硬件选择来启动IWDG。一旦启动了独立看门狗，除了系统复位，它不能再被停止。详见11.2.3节。
- 实时时钟（RTC）：通过备份域控制寄存器（RCC_BDC）的RTCEN位来设置。
- 内部RC振荡器（LSI RC）：通过控制/状态寄存器（RCC_CTRLSTS）的LSIEN位来设置。
- 外部32.768kHz振荡器（LSE）：通过备份域控制寄存器（RCC_BDC）的LSEEN位设置。在停止模式下，如果在进入该模式前ADC和DAC没有被关闭，那么这些外设仍然消耗电流。通过设置寄存器ADC_CTRL2的ADON位和寄存器DAC_CTRL的CHENx位为0可关闭这2个外设。

退出停止模式

关于如何退出停止模式，详见下表。当一个中断或唤醒事件导致退出停止模式时，HSI RC 振荡器被选为系统时钟。

当电压调节器处于低功耗模式下，当系统从停止模式退出时，将会有一段额外的启动延时。如果在停止模式期间保持内部调节器开启，则退出启动时间会缩短，但相应的功耗会增加。

表 2-3 停止模式

停止模式	说明
进入	在以下条件下执行 WFI（等待中断）或 WFE（等待事件）指令： – 设置 Cortex™-M4F 系统控制寄存器中的 SLEEPDEEP 位 – 清除电源控制寄存器（PWR_CTRL）中的 PDDS 位 注：为了进入停止模式，所有的外部中断的请求位（挂起寄存器（EXTI_PR））和 RTC 的闹钟标志都必须被清除，否则停止模式的进入流程将会被跳过，程序继续运行。

退出	如果执行 WFI 进入停止模式： 设置任一外部中断线为中断模式（在 NVIC 中必须使能相应的外部中断向量）。参见中断向量表（表 8-1 AT32F403A/407 产品的向量表）。 如果执行 WFE 进入停止模式： 设置任一外部中断线为事件模式。参见唤醒事件管理（第 8.2.3 节）。
唤醒延时	HSI RC 唤醒时间

2.3.2.3 待机模式

待机模式可实现系统的最低功耗。该模式是在 Cortex™-M4F 深睡眠模式时关闭电压调节器。整个 1.2V 供电区域被断电。PLL、HSI 和 HSE 振荡器也被断电。SRAM 和寄存器内容丢失。只有备份的寄存器和待机电路维持供电（见图 2-1）。

进入待机模式

关于如何进入待机模式，表 2-4。可以通过设置独立的控制位，选择以下待机模式的功能：

- 独立看门狗（IWDG）：可通过写入看门狗的键寄存器或硬件选择来启动IWDG。一旦启动了独立看门狗，除了系统复位，它不能再被停止。详见11.2.3节。
- 实时时钟（RTC）：通过备用区域控制寄存器（RCC_BDC）的RTCEN位来设置。
- 内部RC振荡器（LSI RC）：通过控制/状态寄存器（RCC_CTRLSTS）的LSIEN位来设置。
- 外部32.768kHz振荡器（LSE）：通过备用区域控制寄存器（RCC_BDC）的LSEEN位设置。

退出待机模式

当一个外部复位（NRST 引脚）、IWDG 复位、WKUPF 引脚上的上升沿或 RTC 闹钟事件的上升沿发生时（见图 12-1：简化的 RTC 框图），微控制器从待机模式退出。从待机唤醒后，除了电源控制/状态寄存器（PWR_CTRLSTS）（见第 2.4.2 节），所有寄存器被复位。

从待机模式唤醒后的代码执行等同于复位后的执行（采样启动模式引脚、读取复位向量等）。电源控制/状态寄存器（PWR_CTRLSTS）（见第 2.4.2 节）将会指示内核由待机状态退出。

待机模式下的输入/输出端口状态

在待机模式下，所有的 I/O 引脚处于高阻态，除了以下的引脚：

- 复位引脚（始终有效）
- 当被设置为防侵入或校准输出时的TAMPER引脚
- 被使能的唤醒引脚

表 2-4 待机模式

待机模式	说明
进入	在以下条件下执行 WFI（等待中断）或 WFE（等待事件）指令： – 设置 Cortex™-M4F 系统控制寄存器中的 SLEEPDEEP 位 – 设置电源控制寄存器（PWR_CTRL）中的 PDDS 位 – 清除电源控制/状态寄存器（PWR_CTRLSTS）中的 WUF 位
退出	WKUP 引脚的上升沿、RTC 闹钟事件的上升沿、NRST 引脚上外部复位、IWDG 复位。
唤醒延时	复位持续时间 $T_{RSTTEMPO}$

2.3.2.4 调试模式

默认情况下，如果在进行调试微处理器时，使微处理器进入停止或待机模式，将失去调试连接。这是因为 Cortex™-M4F 的内核失去了时钟。然而，通过设置 DBGMCU_CTRL 寄存器中的某些配置位，可以在使用低功耗模式下调试软件。

更多的细节请参考第 22.2.1 节低功耗模式的调试支持。

2.3.3 自动唤醒

RTC 可以在不需要依赖外部中断的情况下唤醒低功耗模式下的微控制器（自动唤醒模式）。RTC 提供一个可编程的时间基数，用于周期性从停止或待机模式下唤醒。通过对备份区域控制寄存器（RCC_BDC）的 RTCSEL[1：0]位的编程，三个 RTC 时钟源中的二个时钟源可以选作实现此功能。

低功耗 32.768kHz 外部晶振（LSE）

该时钟源提供了一个低功耗且精确的时间基准。（在典型情形下消耗小于 1μA）

低功耗内部 RC 振荡器（LSI RC）

使用该时钟源，节省了一个 32.768kHz 晶振的成本。

为了用 RTC 闹钟事件将系统从停止模式下唤醒，必须进行如下操作：

配置外部中断线 17 为上升沿触发。

配置 RTC 使其可产生 RTC 闹钟事件。

如果要从待机模式中唤醒，不必配置外部中断线 17。

2.4 PWR寄存器

可以用半字（16 位）或字（32 位）的方式操作这些外设寄存器。

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	PWR_CTRL	保留																							DBP	PVDS[2：0]			PVDEN	CLSBF	CLWUF	PDDS	LPDS
	复位值																								0	0	0	0	0	0	0	0	0
0x04	PWR_CTRLSTS	保留																							WUPEN	保留					PVD	SBF	WUF
	复位值																								0						0	0	0

2.4.1 电源控制寄存器（PWR_CTRL）

地址偏移：0x000

复位值：0x0000 0000 （从待机模式唤醒时清除）

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															
res															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留							DBP	PVDS[2：0]			PVDEN	CLSBF	CLWUF	PDDS	LPDS
res							rw	rw			rw	rc_w1	rc_w1	rw	rw

位 31：9	保留。始终读为 0。
位 8	**DBP：**取消后备区域的写保护 在复位后，RTC 和后备寄存器处于被保护状态以防意外写入。设置这位允许写入这些寄存器。 0：禁止写入 RTC 和后备寄存器 1：允许写入 RTC 和后备寄存器 注：如果 RTC 的时钟是 HSE/128，该位必须保持为'1'。
位 7：5	**PVDS[2：0]：**PVD 电平选择 这些位用于选择电源电压监测器的电压阀值 000：**保留** 100：2.6V 001：2.3V 101：2.7V 010：2.4V 110：2.8V 011：2.5V 111：2.9V 注：详细说明参见数据手册中的电气特性部分。
位 4	**PVDEN：**电源电压监测器（PVD）使能 0：禁止 PVD 1：开启 PVD
位 3	**CLSBF：**清除待机位 始终读出为 0 0：无功效 1：清除 SBF 待机位（写）
位 2	**CLWUF：**清除唤醒位 始终读出为 0 0：无功效 1：2 个系统时钟周期后清除 WUF 唤醒位（写）
位 1	**PDDS：**掉电深睡眠 0：CPU 进入深睡眠时进入停机模式 1：CPU 进入深睡眠时进入待机模式
位 0	**LPDS：**深睡眠下的低功耗 PDDS=0 时，与 PDDS 位协同操作 0：在停机模式下电压调压器开启 1：在停机模式下电压调压器处于低功耗模式

2.4.2 电源控制/状态寄存器（PWR_CTRLSTS）

地址偏移：0x004

复位值：0x0000 0000 （从待机模式唤醒时不被清除）

与标准的 APB 读相比，读此寄存器需要额外的 APB 周期

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															
res															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留							WUPEN	保留					PVD	SBF	WUF
res							rw	res					r	r	r

位 31：9	保留。始终读为 0。
位 8	**WUPEN：**使能 WKUPF 引脚 0：WKUPF 引脚为通用 I/O。WKUPF 引脚上的事件不能将 CPU 从待机模式唤醒 1：WKUPF 引脚用于将 CPU 从待机模式唤醒，WKUPF 引脚被强置为输入下拉的配置（WKUPF 引脚 上的上升沿将系统从待机模式唤醒） 注：在系统复位时清除这一位。
位 7：3	保留。始终读为 0。
位 2	**PVD：**PVD 输出 当 PVD 被 PVDEN 位使能后该位才有效 0：VDD/VDDA 高于由 PVDS[2：0]选定的 PVD 阀值 1：VDD/VDDA 低于由 PVDS[2：0]选定的 PVD 阀值 注：在待机模式下 PVD 被停止。因此，待机模式后或复位后，直到设置 PVDEN 位之前，该位为 0。
位 1	**SBF：**待机标志 该位由硬件设置，并只能由 POR/PDR（上电/掉电复位）或设置电源控制寄存器（PWR_CTRL）的 CLSBF 位清除。 0：系统不在待机模式 1：系统进入待机模式
位 0	**WUF：**唤醒标志 该位由硬件设置，并只能由 POR/PDR（上电/掉电复位）或设置电源控制寄存器（PWR_CTRL）的 CLWUF 位清除。 0：没有发生唤醒事件 1：在 WKUPF 引脚上发生唤醒事件或出现 RTC 闹钟事件。 注：当 WKUPF 引脚已经是高电平时，在（通过设置 WUPEN 位）使能 WKUPF 引脚时，会检测到一个额外的事件。

3 复位和时钟控制（RCC）

3.1 复位

AT32F403A/407 支持三种复位形式，分别为系统复位、上电复位和备份区域复位。

3.1.1 系统复位

除了时钟控制器的 RCC_CTRLSTS 寄存器中的复位标志位和备份区域中的寄存器（见图 2-1 各电源域框图）以外，系统复位将复位所有寄存器至它们的复位状态。

当发生以下任一事件时，产生一个系统复位：

1. NRST 引脚上的低电平（外部复位）
2. 窗口看门狗计数终止（WWDG 复位）
3. 独立看门狗计数终止（IWDG 复位）
4. 软件复位（SW 复位）
5. 低功耗管理复位可通过查看 RCC_CTRLSTS 控制状态寄存器中的复位状态标志位识别复位事件来源。

软件复位：

通过将 Cortex™-M4F 中断应用和复位控制寄存器中的 SYSRESETREQ 位置’1’，可实现软件复位。请参考 Cortex™-M4 技术参考手册获得进一步信息。

低功耗管理复位：

在以下两种情况下可产生低功耗管理复位：

1. 在进入待机模式时产生低功耗管理复位：通过将用户选择字节中的 nSTDBY_RST 位置’0’将使能该复位。这时，即使执行了进入待机模式的过程，系统将被复位而不是进入待机模式。
2. 在进入停止模式时产生低功耗管理复位：通过将用户选择字节中的 nSTP_RST 位置’0’将使能该复位。这时，即使执行了进入停机模式的过程，系统将被复位而不是进入停机模式。

3.1.2 电源复位

当以下事件中之一发生时，产生电源复位：

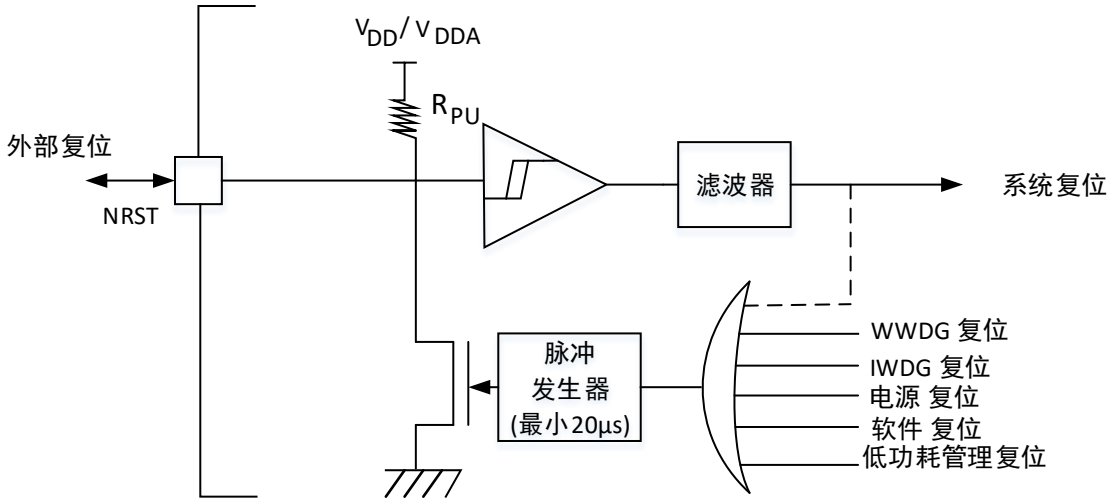
1. 上电/掉电复位（POR/PDR 复位）
2. 从待机模式中返回

电源复位将复位除了备份区域外的所有寄存器。（见图 2-1）

图中复位源将最终作用于 RESET 引脚，并在复位过程中保持低电平。复位入口矢量被固定在地址 0x0000_0004。

芯片内部的复位信号会在 NRST 引脚上输出，脉冲发生器保证每一个（外部或内部）复位源都能有至少 20μs 的脉冲延时；当 NRST 引脚被拉低产生外部复位时，它将产生复位脉冲。

图 3-1 复位电路



3.1.3 备份域复位

备份区域拥有两个专门的复位，它们只影响备份区域（见图 2-1）。当以下事件中之一发生时，产生备份区域复位。

1. 软件复位，备份区域复位可由设置备份域控制寄存器（RCC_BDC）（见 3.3.9 节）中的 BDRST 位产生。
2. 在 VDD 和 VBAT 两者掉电的前提下，VDD 或 VBAT 上电将引发备份区域复位。

3.2 时钟

三种不同的时钟源可被用来驱动系统时钟（SYSCLK）：

- HSI 振荡器时钟
- HSE 振荡器时钟
- PLL 时钟

这些设备有以下 2 种二级时钟源：

- 40kHz 低速内部 RC，可以用于驱动独立看门狗和通过程序选择驱动 RTC。RTC 用于从停机/待机模式下自动唤醒系统。
- 32.768kHz 低速外部晶体也可用来通过程序选择驱动 RTC（RTCCLK）。当不被使用时，任一个时钟源都可被独立地启动或关闭，由此优化系统功耗。

图 3-2 AT32F403A时钟树

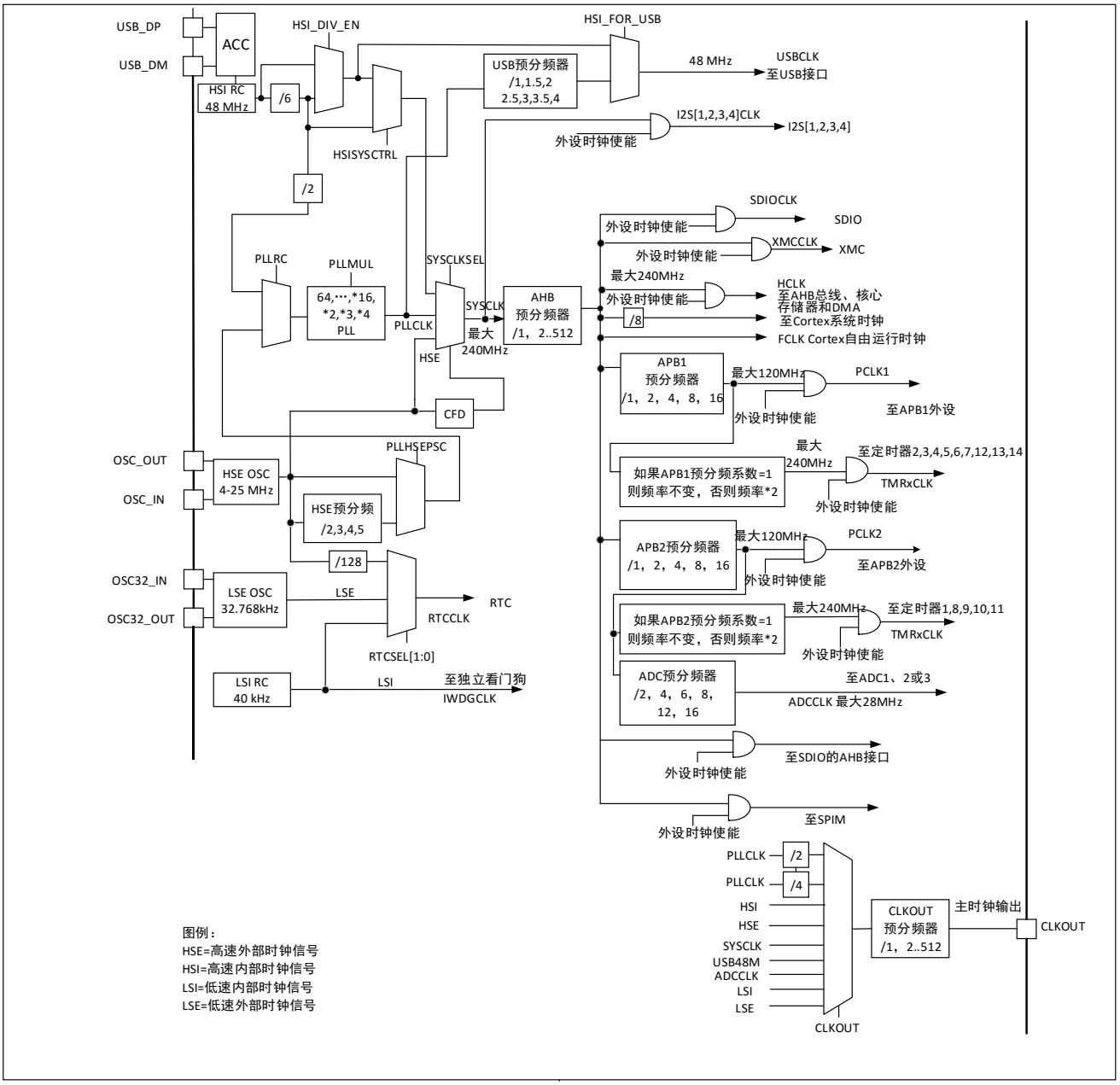
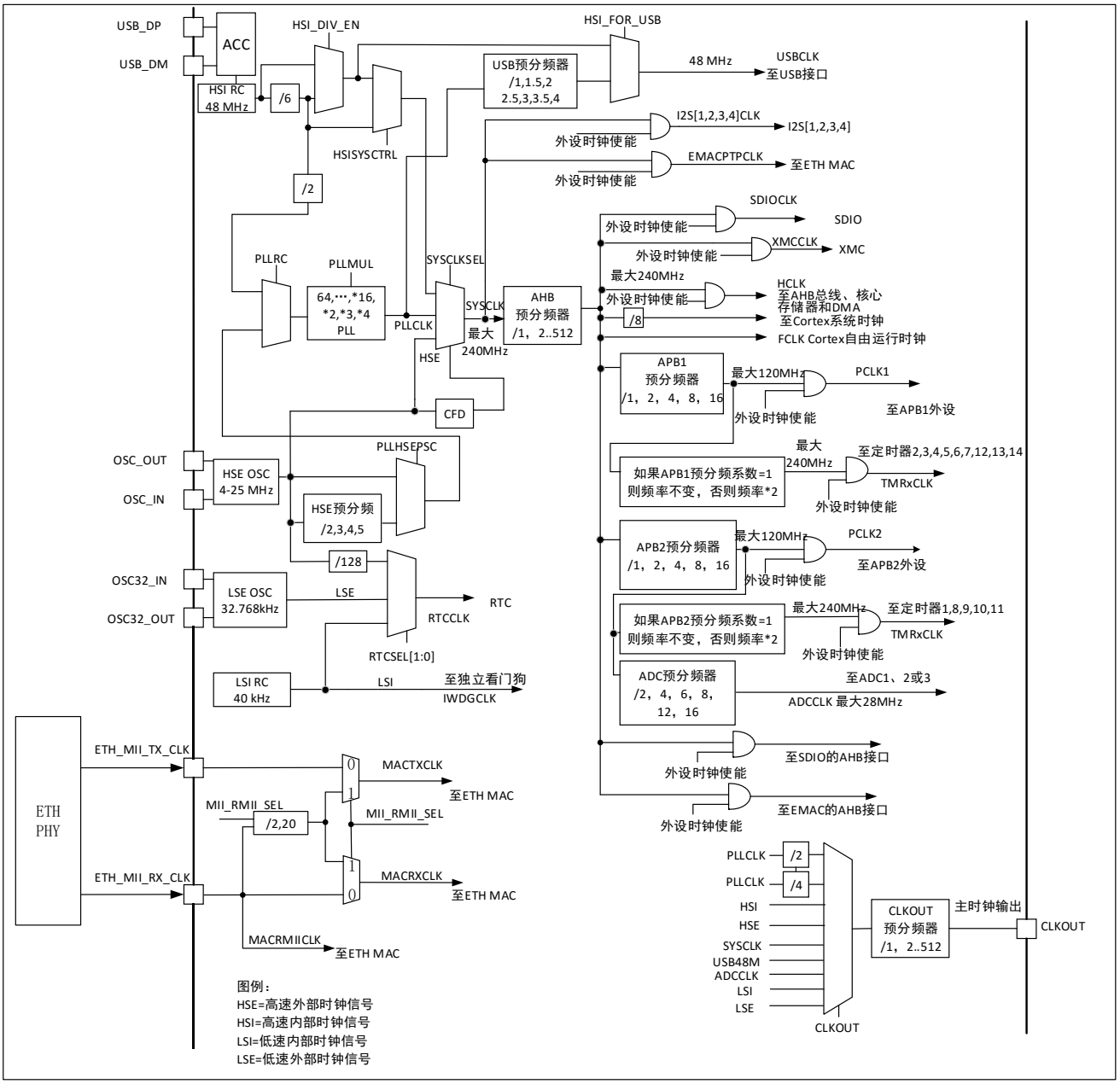


图 3-3 AT32F407时钟树



1. 当 HSI 被用于作为 PLL 时钟的输入时，系统时钟能得到的最大频率是 240MHz。
2. 对于内部和外部时钟源的特性，请参考相应产品数据手册中“电气特性”章节。

用户可通过多个预分频器配置 AHB、高速 APB（APB2）和低速 APB（APB1）域的频率。AHB 域的最大频率是 240MHz。APB1 和 APB2 域的最大允许频率是 120MHz。SDIO 接口的时钟频率固定为 HCLK。

RCC 通过 AHB 时钟（HCLK）8 分频后作为 Cortex 系统定时器（SysTick）的外部时钟。通过对 SysTick 控制与状态寄存器的设置，可选择上述时钟或 Cortex（HCLK）时钟作为 SysTick 时钟。ADC 时钟由高速 APB2 时钟经 2、4、6、8、12、16 分频后获得。

定时器时钟频率分配由硬件按以下 2 种情况自动设置：

1. 如果相应的 APB 预分频系数是 1，定时器的时钟频率与所在 APB 总线频率一致。
2. 否则，定时器的时钟频率被设为与其相连的 APB 总线频率的 2 倍。

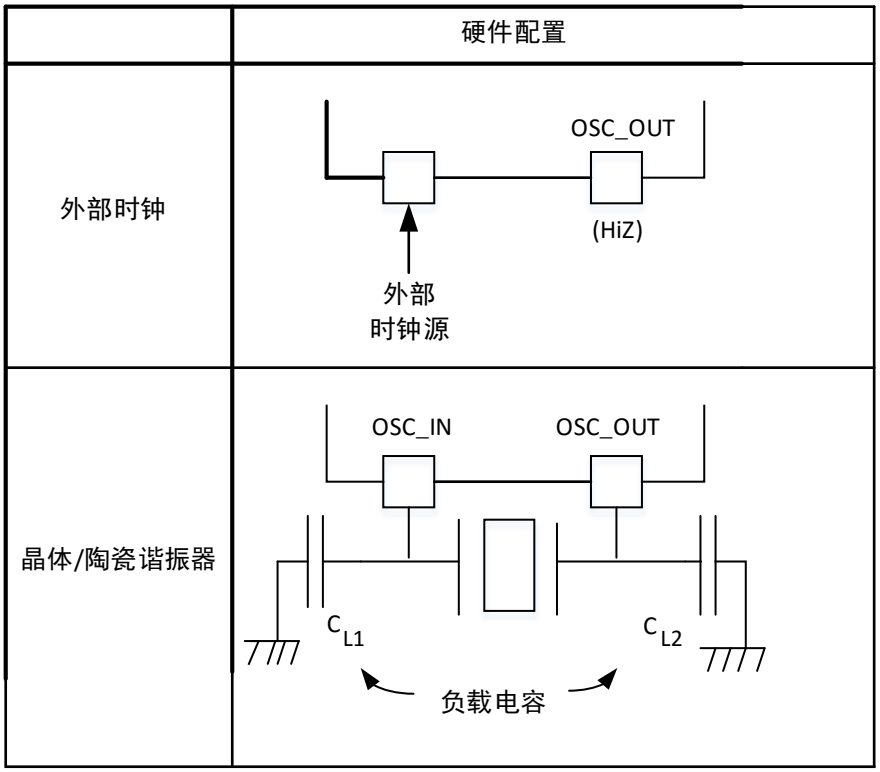
FCLK 是 Cortex™-M4F 的自由运行时钟。详情见 ARM 的 Cortex™-M4 技术参考手册。

3.2.1 HSE时钟

高速外部时钟信号（HSE）由以下两种时钟源产生：

- HSE外部晶体/陶瓷谐振器
- HSE用户外部时钟为了减少时钟输出的失真和缩短启动稳定时间，晶体/陶瓷谐振器和负载电容器必须尽可能地靠近振荡器引脚。负载电容值必须根据所选择的振荡器来调整。

图 3-4 HSE/LSE时钟源



外部时钟源（HSE 旁路）

在这个模式里，必须提供外部时钟。它的频率最高可达 25MHz。用户可通过设置在时钟控制寄存器中的 HSEBYPS 和 HSEEN 位来选择这一模式。外部时钟信号（50%占空比的方波、正弦波或三角波）必须连到 OSC_IN 引脚，同时保证 OSC_OUT 引脚悬空。见图 3-4。

外部晶体/陶瓷谐振器（HSE 晶体）

4~25MHz 外部振荡器可为系统提供更为精确的主时钟。相关的硬件配置可参考图 3-4，进一步信息可参考数据手册的电气特性部分。

在时钟控制寄存器 RCC_CTRL 中的 HSESTBL 位用来指示高速外部振荡器是否稳定。在启动时，直到这一位被硬件置’1’，时钟才被释放出来。如果在时钟中断寄存器 RCC_CLKINT 中允许产生中断，将会产生相应中断。

HSE 晶体可以通过设置时钟控制寄存器里 RCC_CTRL 中的 HSEEN 位被启动和关闭。

3.2.2 HSI时钟

HSI 时钟信号由内部 48MHz 的 RC 振荡器产生，可直接作为系统时钟或在 2 分频后作为 PLL 输入。

HSI RC 振荡器能够在不需要任何外部器件的条件下提供系统时钟。它的启动时间比 HSE 晶体振荡器短。然而，即使在校准之后它的时钟频率精度仍较差。

校准制造工艺决定了不同芯片的 RC 振荡器频率会不同，这就是为什么每个芯片的 HSI 时钟频率在出厂前已经被 ATK 校准到 1%（25° C）的原因。系统复位时，工厂校准值被装载到时钟控制寄存器的 HSICAL[7:0]位。HSICAL[7:0]可被软件进行再配置（需要把 HSICAL_KEY 设成 0x5A），一旦经由软件配置过后，需

重置系统才能重新装载工厂校准值。

如果用户的应用基于不同的电压或环境温度，这将会影响 RC 振荡器的精度。可以通过时钟控制寄存器里的 HSITWK[5：0]位来调整 HSI 频率。

时钟控制寄存器中的 HSISTBL 位用来指示 HSI RC 振荡器是否稳定。在时钟启动过程中，直到这一位被硬件置‘1’，HSI RC 输出时钟才被释放。HSI RC 可由时钟控制寄存器中的 HSIEN 位来启动和关闭。

如果 HSE 晶体振荡器失效，HSI 时钟会被作为备用时钟源。参考 3.2.7 时钟失效检测（CFD）。

3.2.3 PLL

内部 PLL 可以用来倍频 HSI RC 的输出时钟或 HSE 晶体输出时钟。参考图 3-2 和时钟控制寄存器。

PLL 的设置（选择 HSI 振荡器除 2 或 HSE 振荡器为 PLL 的输入时钟，和选择倍频因子）必须在其被激活前完成。一旦 PLL 被激活，这些参数就不能被改动。PLL 的输入时钟范围为 2M~16MHz 之间。

如果 PLL 中断在时钟中断寄存器里被允许，当 PLL 准备就绪时，可产生中断申请。如果需要在应用中使用 USB 接口，PLL 必须被设置为输出 48、72、96、120、144、168 或 192MHZ 时钟，用于提供 48MHz 的 USBCLK 时钟。

3.2.4 LSE时钟

LSE 晶体是一个 32.768kHz 的低速外部晶体或陶瓷谐振器。它为实时时钟或者其他定时功能提供一个低功耗且精确的时钟源。

LSE 晶体通过在备份域控制寄存器（RCC_BDC）里的 LSEEN 位启动和关闭。在备份域控制寄存器（RCC_BDC）里的 LSESTBL 指示 LSE 晶体振荡是否稳定。在启动阶段，直到这个位被硬件置’1’后，LSE 时钟信号才被释放出来。如果在时钟中断寄存器里被允许，可产生中断申请。

外部时钟源（LSE 旁路）

在这个模式里必须提供一个 32.768kHz 频率的外部时钟源。你可以通过设置在备份域控制寄存器（RCC_BDC）里的 LSEBYPS 和 LSEEN 位来选择这个模式。具有 50%占空比的外部时钟信号（方波、正弦波或三角波）必须连到 OSC32_IN 引脚，同时保证 OSC32_OUT 引脚悬空，见图 3-4。

3.2.5 LSI时钟

LSI RC 担当一个低功耗时钟源的角色，它可以在停机和待机模式下保持运行，为独立看门狗和自动唤醒单元提供时钟。LSI 时钟频率大约 40kHz（在 30kHz 和 60kHz 之间）。进一步信息请参考数 据手册中有关电气特性部分。

LSI RC 可以通过控制/状态寄存器（RCC_CTRLSTS）里的 LSIEN 位来启动或关闭。在控制/状态寄存器（RCC_CTRLSTS）里的 LSISTBL 位指示低速内部振荡器是否稳定。在启动阶段，直到这个位被硬件设置为’1’后，此时钟才被释放。如果在时钟中断寄存器（RCC_CLKINT）里被允许，将产生 LSI 中断申请。

LSI 校准可以通过校准内部低速振荡器 LSI 来补偿其频率偏移，从而获得精度可接受的 RTC 时间基数，以及独立看门狗（IWDG）的超时时间（当这些外设以 LSI 为时钟源）。

校准可以通过使用 TMR5 的输入时钟（TMR5_CLK）测量 LSI 时钟频率实现。测量以 HSE 的精度为保证，软件可以通过调整 RTC 的 20 位预分频器来获得精确的 RTC 时钟基数，以及通过计算得到精确的独立看门狗（IWDG）的超时时间。

LSI 校准步骤如下：

1. 打开 TMR5，设置通道 4 为输入捕获模式；
2. 设置 AFIO_MAP 的 TMR5_CH4_IREMAP 位为’1’，在内部把 LSI 连接到 TMR5 的通道 4；
3. 通过 TMR5 的捕获/比较 4 事件或者中断来测量 LSI 时钟频率；
4. 根据测量结果和期望的 RTC 时间基数和独立看门狗的超时时间，设置 20 位预分频器。

3.2.6 系统时钟（SYSCLK）选择

系统复位后，HSI 振荡器被选为系统时钟。当时钟源被直接或通过 PLL 间接作为系统时钟时，它将不能被停止。

只有当目标时钟源准备就绪了（经过启动稳定阶段的延迟或 PLL 稳定），从一个时钟源到另一个时钟源的切换才会发生。在被选择时钟源没有就绪时，系统时钟的切换不会发生。直至目标时钟源就绪，才发生切换。

在时钟控制寄存器（RCC_CTRL）里的状态位指示哪个时钟已经准备好了，哪个时钟目前被用作系统时钟。

3.2.7 时钟失效检测（CFD）

时钟失效检测可以通过软件被激活。一旦其被激活，时钟监测器将在 HSE 振荡器启动延迟后被使能，并在 HSE 时钟关闭后关闭。

如果 HSE 时钟发生故障，HSE 振荡器被自动关闭，时钟失效事件将被送到高级定时器（TMR1 和 TMR8）的刹车输入端，并产生时钟安全中断 CFDI，允许软件完成营救操作。此 CFDI 中断连接到 Cortex™-M4F 的 NMI 中断（不可屏蔽中断）。

注意： 一旦 CFD 被激活，并且 HSE 时钟出现故障，CFD 中断就产生，并且 NMI 也自动产生。NMI 将被不断执行，直到 CFD 中断挂起位被清除。因此，在 NMI 的处理程序中必须通过设置时钟中断寄存器（RCC_CLKINT）里的 CFDFC 位来清除 CFD 中断。

如果 HSE 振荡器被直接或间接地作为系统时钟，（间接的意思是：它被作为 PLL 输入时钟，并且 PLL 时钟被作为系统时钟），时钟故障将导致系统时钟自动切换到 HSI 振荡器，同时外部 HSE 振荡器被关闭。在时钟失效时，如果 HSE 振荡器时钟（被分频或未被分频）是用作系统时钟的 PLL 的输入时钟，PLL 也将被关闭。

3.2.8 RTC时钟

通过设置备份域控制寄存器（RCC_BDC）里的 RTCSEL[1：0]位，RTCCLK 时钟源可以由 HSE/128、LSE 或 LSI 时钟提供。除非备份域复位，此选择不能被改变。

LSE 时钟在备份域里，但 HSE 和 LSI 时钟不是。因此：

- 如果 LSE 被选为 RTC 时钟：
  - 只要 VBAT 维持供电，RTC 仍继续工作。
- 如果 LSI 被选为自动唤醒单元（AWU）时钟：
  - 如果 VDD 供电被切断，AWU 状态不能被保证。有关 LSI 校准，详见 3.2.5 节 LSI 时钟。
- 如果 HSE 时钟 128 分频后作为 RTC 时钟：
  - 如果 VDD 供电被切断或内部电压调压器被关闭（1.2V 域的供电被切断），则 RTC 状态不确定。
  - 必须设置电源控制寄存器（见 2.4.1 节）的 DPB 位（取消后备区域的写保护）为’1’。

3.2.9 看门狗时钟

如果独立看门狗已经由硬件选项或软件启动，LSI 振荡器将被强制在打开状态，并且不能被关闭。在 LSI 振荡器稳定后，时钟供应给 IWDG。

3.2.10 时钟输出

微控制器允许输出时钟信号到外部 CLKOUT 引脚。相应的 GPIO 端口寄存器必须被配置为相应功能。以下 9 个时钟信号可被选作 CLKOUT 时钟：

- ADC CLK
- USB48M
- SYSCLK
- LSI
- LSE
- HSI
- HSE

- 除2的PLL时钟。
- 除4的PLL时钟。

3.3 RCC寄存器描述

下表列出了 RCC 寄存器的映像和复位值。

表 3-1 RCC寄存器的映像和复位值

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	RCC_CTRL	保留						PLLSTBL	PLLEN	保留				CFDEN	HSEBYPS	HSESTBL	HSEEN	HSICAL[7：0]								HSITWK[5：0]						HSISTBL	HSION
	复位值							0	0					0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	1
0x04	RCC_CFG	PLLRANGE	PLLMUL[5：4]		ADCPSC[2]	USBPSC[2]	CLKOUT [2：0]			USBPSC[1：0]		PLLMUL [3：0]				PLLHSEPSC	PLLRC	ADCPSC[1：0]		APB2PSC [2：0]			APB1PSC [2：0]			AHBPSC [3：0]				SYSCLKSTS [1：0]		SYSCLKSEL [1：0]	
	复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08	RCC_CLKINT	保留								CFDFC	保留			PLLSTBLFC	HSESTBLFC	HSISTBLFC	LSESTBLFC	LSISTBLFC	保留			PLLSTBLIE	HSESTBLIE	HSISTBLIE	LSESTBLIE	LSISTBLIE	CFDF	保留		PLLSTBLF	HSESTBLF	HSISTBLF	LSESTBLF	LSISTBLF
	复位值									0				0	0	0	0	0				0	0	0	0	0	0			0	0	0	0	0
0x0C	RCC_APB2RSTR	保留					UART8RST	UART7RST	USART6RST	I2C3RST	ACCRST	TMR11RST	TMR10RST	TMR9RST	保留			ADC3RST	USART1RST	TMR8RST	SPI1RST	TMR1RST	ADC2RST	ADC1RST	保留			GPIOERST	GPIODRST	GPIOCRST	GPIOBRST	GPIOARST	EXTIRST	AFIORST
	复位值						0	0	0	0	0	0	0	0				0	0	0	0	0	0	0				0	0	0	0	0	0	0
0x10	RCC_APB1RSTR	保留	DACRST	PWRRST	BKPRST	CAN2RST	CANRST	保留		USBRST	I2C2RST	I2C1RST	UART5RST	UART4RST	USART3RST	USART2RST	SPI4RST	SPI3RST	SPI2RST	保留		WWDGRST	保留			TMR14RST	TMR13RST	TMR12RST	TMR7RST	TMR6RST	TMR5RST	TMR4RST	TMR3RST	TMR2RST
	复位值		0	0	0	0	0			0	0	0	0	0	0	0		0	0			0				0	0	0	0	0	0	0	0	0
0x14	RCC_AHBEN	保留		ETHMACPTPEN	保留												ETHMACRXEN	ETHMACTXEN	ETHMACEN	保留		SDIO2EN	SDIO1EN	保留	XMCEN	保留	CCE	保留	FLASHEN	保留	SRAMEN	DMA2EN	DMA1EN
	复位值			0													0	0	0			0	0		0		0		1		1	0	0
0x18	RCC_APB2EN	保留					UART8EN	UART7EN	USART6EN	I2C3EN	ACCEN	TMR11EN	TMR10EN	TMR9EN	保留			ADC3EN	USART1EN	TMR8EN	SPI1EN	TMR1EN	ADC2EN	ADC1EN	保留			GPIOEEN	GPIODEN	GPIOCEN	GPIOBEN	GPIOAEN	保留	AFIOEN
	复位值						0	0	0	0	0	0	0	0				0	0	0	0	0	0	0				0	0	0	0	0		0
0x1C	RCC_APB1EN	保留	DACEN	PWREN	BKPEN	CAN2EN	CAN1EN	保留		USBEN	I2C2EN	I2C1EN	UART5EN	UART4EN	USART3EN	USART2EN	SPI4EN	SPI3EN	SPI2EN	保留		WWDGEN	保留			TMR14EN	TMR13EN	TMR12EN	TMR7EN	TMR6EN	TMR5EN	TMR4EN	TMR3EN	TMR2EN
	复位值		0	0	0	0	0			0	0	0	0	0	0	0	0	0	0			0				0	0	0	0	0	0	0	0	0

0x20	RCC_BDC	保留																	BDRST	RTCEN	保留								RTCSEL[1：0]		保留							LSEBYPS	LSESTBL	LSEEN
	复位值																		0	0									0	0							0	0	0
0x24	RCC_CTRLSTS	LPRSTF	WWDGRSTF	IWDGRSTF	SWRSTF	PORSTF	PINRSTF	保留	RSTFC	保留																								LSISTBL	LSIEN
	复位值	0	0	0	0	1	1		0																									0	0
0x28	RCC_AHBRST	保留																															ETHMACRST	保留								
	复位值																																0								
0x30	RCC_MISC	MCOPRE				保留	HSI_DIV_EN	USB768B	保留										CLKOUT[3]	保留													HSICAL_KEY[7:0]							
	复位值	0	0	0	0		0	0											0														0	0	0	0	0	0	0	0
0x50	RCC_TEST	保留																	MCO2TMR_EN	保留																				
	复位值																		0																						
0x54	RCC_MISC2	保留																				EMAC_PPS_SEL	保留	HSE_DIV_CTRL		保留			HSI_SYS_CTRL	HSI_FOR_USB	保留		AUTO_STEP_EN		保留						
	复位值																					0		0	0				0	0			0	0							
0x5C	RCC_INTCTRL	保留																																						USB_INT_CTRL
	复位值																																							0

3.3.1 时钟控制寄存器（RCC_CTRL）

偏移地址：0x00

复位值：0x0000 XX83，X 代表未定义

访问：无等待状态，字，半字和字节访问

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留						PLL STBL	PLL EN	保留				CFD EN	HSE BYPS	HSE STBL	HSE EN
res						r	rw	res				rw	rw	r	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HSICAL[7：0]								HSITWK[5：0]						HSI STBL	HSI EN
rw								rw						r	rw

位 30：26	保留，始终读为 0。
位 25	**PLLSTBL**：PLL 时钟就绪标志（PLL clock ready flag） PLL 锁定后由硬件置'1'。 0：PLL 未锁定； 1：PLL 锁定。
位 24	**PLLEN**：PLL 使能（PLL enable） 由软件置'1'或清零。当进入待机和停止模式时，该位由硬件清零。当 PLL 时钟被用作或被选择将要作为系统时钟时，该位不能被清零。 0：PLL 关闭； 1：PLL 使能。
位 23：20	保留，始终读为 0。
位 19	**CFDEN**：时钟失效检测使能（Clock Failure Detection enable） 由软件置'1'或清零以使能时钟监测器。 0：时钟监测器关闭； 1：如果外部 4-25MHz 振荡器就绪，时钟监测器开启。
位 18	**HSEBYPS**：外部高速时钟旁路（External high-speed clock bypass） 在调试模式下由软件置'1'或清零来旁路外部晶体振荡器。只有在外部 4-25MHz 振荡器关闭的情况下，才能写入该位。 0：外部 4-25MHz 振荡器没有旁路； 1：外部 4-25MHz 外部晶体振荡器被旁路。
位 17	**HSESTBL**：外部高速时钟就绪标志（External high-speed clock ready flag） 由硬件置'1'来指示外部 4-25MHz 振荡器已经稳定。在 HSEEN 位清零后，该位需要 6 个外部 4-25MHz 振荡器周期清零。 0：外部 4-25MHz 振荡器没有就绪； 1：外部 4-25MHz 振荡器就绪。
位 16	**HSEEN**：外部高速时钟使能（External high-speed clock enable） 由软件置'1'或清零。 当进入待机和停止模式时，该位由硬件清零，关闭 4-25MHz 外部振荡器。当外部 4-25MHz 振荡器被用作或被选择将要作为系统时钟时，该位不能被清零。 0：HSE 振荡器关闭； 1：HSE 振荡器开启。

位 15：8	**HSICAL[7：0]：**内部高速时钟校准（Internal high-speed clock calibration） 在系统启动时，这些位被自动初始化。 HSI 输出频率为 48MHZ 时，每步 HSICAL 的变化调整 240kHz（设计值）；HSI 输出频率是 8MHZ 时，每步 HSICAL 的变化调整 40kHz（设计值）。 **此字段只有在 HSICAL_KEY[7:0]为 0x5A 的时候可被写入。**
位 7：2	**HSITWK[5：0]：**内部高速时钟调整（Internal high-speed clock trimming） 由软件写入来调整内部高速时钟，它们被叠加在 HSICAL[7：0]数值上。这些位在 HSICAL[7：0]的基础上，让用户可以输入一个调整数值，根据电压和温度的变化调整内部 HSI RC 振荡器的频率。 默认数值为 32，可以把 HSI 调整到 48MHz±1%；
位 1	**HSISTBL：**内部高速时钟就绪标志（Internal high-speed clock ready flag） 由硬件置'1'来指示内部 48MHz 振荡器已经稳定。在 HSIEN 位清零后，该位需要 6 个内部 48MHz 振荡器周期清零。 0：内部 48MHz 振荡器没有就绪； 1：内部 48MHz 振荡器就绪。
位 0	**HSIEN：**内部高速时钟使能（Internal high-speed clock enable） 由软件置'1'或清零。 当从待机和停止模式返回或用作系统时钟的外部 4-25MHz 振荡器发生故障时，该位由硬件置'1'来启动内部 48MHz 的 RC 振荡器。当内部 48MHz 振荡器被直接或间接地用作或被选择将要作为系 统时钟时，该位不能被清零。 0：内部 48MHz 振荡器关闭； 1：内部 48MHz 振荡器开启。

3.3.2 时钟配置寄存器（RCC_CFG）

偏移地址：0x04

复位值：0x0000 0000

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。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PLL RANGE	PLLMUL [5：4]		ADC PSC[2]	USB PSC[2]	CLKOUT[2：0]			USBPSC [1：0]		PLLMUL[3：0]				PLLHSE PSC	PLL RC
rw	rw		rw	rw	rw			rw		rw				rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADCPSC[1：0]		APB2PSC[2：0]			APB1PSC[2：0]			AHBPSC[3：0]				SYSCLKSTS [1：0]		SYSCLKSEL [1：0]	
rw		rw			rw			rw				r		rw	

位 31	**PLLRANGE：**PLL Clock 输出范围 0： PLL output ≦ 72 MHz 1： PLL output > 72 MHz

位 26：24	**CLKOUT[3:0]：**微控制器时钟输出（Microcontroller clock output） CLKOUT[3] 在 RCC_MISC 寄存器位 16 由软件置'1'或清零。 0000：没有时钟输出； 0001: 保留 0010: LSI 时钟输出； 0011: LSE 时钟输出； 0100：系统时钟（SYSCLK）输出； 0101：内部 RC 振荡器时钟（HSI）输出； 0110：外部振荡器时钟（HSE）输出； 0111：PLL 时钟 2 分频后输出。 1100：PLL 时钟 4 分频后输出。 1101：USB 时钟输出。 1110：ADC 时钟输出。 注意：- 该时钟输出在启动和切换 CLKOUT 时钟源时可能会被截断。 - 在系统时钟作为输出至 CLKOUT 引脚时，请保证输出时钟频率不超过 50MHz（I/O 口最高频率）。
位 27 位 23：22	**USBPSC[2：0]：**USB 预分频（USB prescaler） 由软件置'1'或清'0'来产生 48MHz 的 USB 时钟。在 RCC_APB1EN 寄存器中使能 USB 时钟之前，必须保证该位已经有效。如果 USB 时钟被使能，该位不能被清零。 000：PLL 时钟 1.5 倍分频作为 USB 时钟 001：PLL 时钟直接作为 USB 时钟 010：PLL 时钟 2.5 倍分频作为 USB 时钟 011：PLL 时钟 2 倍分频作为 USB 时钟 100：PLL 时钟 3.5 倍分频作为 USB 时钟 101：PLL 时钟 3 倍分频作为 USB 时钟 110：PLL 时钟 4 倍分频作为 USB 时钟 111：PLL 时钟 4 倍分频作为 USB 时钟
位 30：29 位 21：18	**PLLMUL[5：0]：**PLL 倍频系数（PLL multiplication factor）{ 位 30：29，位 21：18} 由软件设置来确定 PLL 倍频系数。只有在 PLL 关闭的情况下才可被写入。 注意：PLLRANGE 寄存器须搭配 PLL 输出频率设置 000000：PLL 2 倍频输出　　000001：PLL 3 倍频输出 000010：PLL 4 倍频输出　　000011：PLL 5 倍频输出 …… 001100：PLL 14 倍频输出　　001101：PLL 15 倍频输出 001110：PLL 16 倍频输出　　001111：PLL 16 倍频输出 010000：PLL 17 倍频输出　　010001：PLL 18 倍频输出 010010：PLL 19 倍频输出　　010011：PLL 20 倍频输出 …… 111110：PLL 63 倍频输出　　111111：PLL 64 倍频输出
位 17	**PLLHSEPSC：**HSE 分频器作为 PLL 输入（HSE divider for PLL entry） 由软件置'1'或清'0'来分频 HSE 后作为 PLL 输入时钟。只能在关闭 PLL 时才能写入此位。 0：HSE 不分频 1：HSE 分频，分频系数由 HSE_DIV_CTRL 设定
位 16	**PLLRC：**PLL 输入时钟源（PLL entry clock source） 由软件置'1'或清'0'来选择 PLL 输入时钟源。只能在关闭 PLL 时才能写入此位。 0：HSI 振荡器时钟经 2 分频后作为 PLL 输入时钟 1：HSE 时钟作为 PLL 输入时钟。

位 28 位 15：14	**ADCPSC[2：0]：**ADC 预分频（ADC prescaler） ADCPSC[2]， 由位 28 来做设定 由软件置'1'或清'0'来确定 ADC 时钟频率 000：PCLK2 2 分频后作为 ADC 时钟 001：PCLK2 4 分频后作为 ADC 时钟 010：PCLK2 6 分频后作为 ADC 时钟 011：PCLK2 8 分频后作为 ADC 时钟 100：PCLK2 2 分频后作为 ADC 时钟 101：PCLK2 12 分频后作为 ADC 时钟 110：PCLK2 8 分频后作为 ADC 时钟 111：PCLK2 16 分频后作为 ADC 时钟
位 13：11	**APB2PSC[2：0]：**高速 APB 预分频（APB2）（APB high-speed prescaler（APB2）） 由软件置'1'或清'0'来控制高速 APB2 时钟（PCLK2）的预分频系数。 警告：软件必须保证 APB2 时钟频率不超过 120MHz。 0xx：HCLK 不分频 100：HCLK 2 分频 101：HCLK 4 分频 110：HCLK 8 分频 111：HCLK 16 分频
位 10：8	**APB1PSC[2：0]：**低速 APB 预分频（APB1）（APB low-speed prescaler（APB1）） 由软件置'1'或清'0'来控制低速 APB1 时钟（PCLK1）的预分频系数。 警告：软件必须保证 APB1 时钟频率不超过 120MHz。 0xx：HCLK 不分频 100：HCLK 2 分频 101：HCLK 4 分频 110：HCLK 8 分频 111：HCLK 16 分频
位 7：4	**AHBPSC[3：0]：**AHB 预分频（AHB Prescaler） 由软件置'1'或清'0'来控制 AHB 时钟的预分频系数。 0xxx：SYSCLK 不分频 1000：SYSCLK 2 分频　1100：SYSCLK 64 分频 1001：SYSCLK 4 分频　1101：SYSCLK 128 分频 1010：SYSCLK 8 分频　1110：SYSCLK 256 分频 1011：SYSCLK 16 分频　1111：SYSCLK 512 分频 注意：当 AHB 时钟的预分频系数大于 1 时，必须开启预取缓冲器。
位 3：2	**SYSCLKSTS[1：0]：**系统时钟切换状态（System clock switch status） 由硬件置'1'或清'0'来指示哪一个时钟源被作为系统时钟。 00：HSI 作为系统时钟； 01：HSE 作为系统时钟； 10：PLL 输出作为系统时钟； 11：不可用。
位 1：0	**SYSCLKSEL[1：0]：**系统时钟切换（System clock switch） 由软件置'1'或清'0'来选择系统时钟源。在从停止或待机模式中返回时或直接或间接作为系统时钟的 HSE 出现故障时，由硬件强制选择 HSI 作为系统时钟（如果时钟安全系统已经启动） 00：HSI 作为系统时钟； 01：HSE 作为系统时钟； 10：PLL 输出作为系统时钟； 11：不可用。

3.3.3 时钟中断寄存器 （RCC_CLKINT）

偏移地址：0x08

复位值：0x0000 0000

访问：无等待周期，字，半字和字节访问

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留								CFDFC	保留		PLLSTB LFC	HSEST BLFC	HSIST BLFC	LSES TBLFC	LSIST BLFC
res								w	res		w	w	w	w	w

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留			PLLS TBLIE	HSES TBLIE	HSIS TBLIE	LSES TBLIE	LSIST BLIE	CFDF	保留		PLLS TBLF	HSES TBLF	HSIS TBLF	LSES TBLF	LSIS TBLF
res			rw	rw	rw	rw	rw	r	res		r	r	r	r	r

位 31：24	保留，始终读为 0。
位 23	**CFDFC**：清除时钟失效中断（Clock failure detection interrupt clear） 由软件置'1'来清除 CFDF 安全系统中断标志位 CFDF。 0：无作用； 1：清除 CFDF 安全系统中断标志位。
位 22：21	保留，始终读为 0。
位 20	**PLLSTBLFC**：清除 PLL 就绪中断（PLL ready interrupt clear） 由软件置'1'来清除 PLL 就绪中断标志位 PLLSTBLF。 0：无作用； 1：清除 PLL 就绪中断标志位 PLLSTBLF。
位 19	**HSESTBLFC**：清除 HSE 就绪中断（HSE ready interrupt clear） 由软件置'1'来清除 HSE 就绪中断标志位 HSESTBLF。 0：无作用； 1：清除 HSE 就绪中断标志位 HSESTBLF。
位 18	**HSISTBLFC**： 清除 HSI 就绪中断（HSI ready interrupt clear） 由软件置'1'来清除 HSI 就绪中断标志位 HSISTBLF。 0：无作用； 1：清除 HSI 就绪中断标志位 HSISTBLF。
位 17	**LSESTBLFC**：清除 LSE 就绪中断（LSE ready interrupt clear） 由软件置'1'来清除 LSE 就绪中断标志位 LSESTBLF。 0：无作用； 1：清除 LSE 就绪中断标志位 LSESTBLF。
位 16	**LSISTBLFC**：清除 LSI 就绪中断（LSI ready interrupt clear） 由软件置'1'来清除 LSI 就绪中断标志位 LSISTBLF。 0：无作用； 1：清除 LSI 就绪中断标志位 LSISTBLF。
位 15：13	保留，始终读为 0。
位 12	**PLLSTBLIE**：PLL 就绪中断使能（PLL ready interrupt enable） 由软件置'1'或清'0'来使能或关闭 PLL 就绪中断。 0：PLL 就绪中断关闭； 1：PLL 就绪中断使能。
位 11	**HSESTBLIE**：HSE 就绪中断使能（HSE ready interrupt enable） 由软件置'1'或清'0'来使能或关闭外部 4-25MHz 振荡器就绪中断。 0：HSE 就绪中断关闭； 1：HSE 就绪中断使能。
位 10	**HSISTBLIE**：HSI 就绪中断使能（HSI ready interrupt enable） 由软件置'1'或清'0'来使能或关闭内部 48MHz RC 振荡器就绪中断。 0：HSI 就绪中断关闭； 1：HSI 就绪中断使能。
位 9	**LSESTBLIE**：LSE 就绪中断使能（LSE ready interrupt enable） 由软件置'1'或清'0'来使能或关闭外部 32kHz RC 振荡器就绪中断。 0：LSE 就绪中断关闭； 1：LSE 就绪中断使能。

位 8	**LSISTBLIE**：LSI 就绪中断使能（LSI ready interrupt enable） 由软件置'1'或清'0'来使能或关闭内部 40kHz RC 振荡器就绪中断。 0：LSI 就绪中断关闭； 1：LSI 就绪中断使能。
位 7	**CFDF**：时钟失效中断标志（Clock Failure Detection interrupt flag） 在外部 4-25MHz 振荡器时钟出现故障时，由硬件置'1'。 由软件通过置'1' CFDFC 位来清除。 0：无 HSE 时钟失效产生的安全系统中断； 1：HSE 时钟失效导致了时钟安全系统中断。
位 6：5	保留，始终读为 0。
位 4	**PLLSTBLF**：PLL 就绪中断标志（PLL ready interrupt flag） 在 PLL 就绪且 PLLSTBLIE 位被置'1'时，由硬件置'1'。 由软件通过置'1' PLLSTBLFC 位来清除。 0：无 PLL 上锁产生的时钟就绪中断； 1：PLL 上锁导致时钟就绪中断。
位 3	**HSESTBLF**：HSE 就绪中断标志（HSE ready interrupt flag） 在外部低速时钟就绪且 HSESTBLIE 位被置'1'时，由硬件置'1'。 由软件通过置'1' HSESTBLFC 位来清除。 0：无外部 4-25MHz 振荡器产生的时钟就绪中断； 1：外部 4-25MHz 振荡器导致时钟就绪中断。
位 2	**HSISTBLF**：HSI 就绪中断标志（HSI ready interrupt flag） 在内部高速时钟就绪且 HSISTBLIE 位被置'1'时，由硬件置'1'。 由软件通过置'1' HSISTBLFC 位来清除。 0：无内部 48MHz RC 振荡器产生的时钟就绪中断； 1：内部 48MHz RC 振荡器导致时钟就绪中断。
位 1	**LSESTBLF**：LSE 就绪中断标志（LSE ready interrupt flag） 在外部低速时钟就绪且 LSESTBLIE 位被置'1'时，由硬件置'1'。 由软件通过置'1' LSESTBLFC 位来清除。 0：无外部 32kHz 振荡器产生的时钟就绪中断； 1：外部 32kHz 振荡器导致时钟就绪中断。
位 0	**LSISTBLF**：LSI 就绪中断标志（LSI ready interrupt flag） 在内部低速时钟就绪且 LSISTBLIE 位被置'1'时，由硬件置'1'。由软件通过置'1' LSISTBLFC 位来清除。 0：无内部 40kHz RC 振荡器产生的时钟就绪中断； 1：内部 40kHz RC 振荡器导致时钟就绪中断。

3.3.4 APB2外设复位寄存器（RCC_APB2RST）

偏移地址：0x0C

复位值：0x0000 0000

访问：无等待周期，字，半字和字节访问

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留					UART8 RST	UART7 RST	USART 6RST	I2C3 RST	ACC RST	TMR11 RST	TMR10 RST	TMR9 RST	保留		
res					rw	rw	rw	rw	rw	rw	rw	rw	res		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADC3 RST	USAR T1RST	TMR8 RST	SPI1 RST	TMR1 RST	ADC2 RST	ADC1 RST	保留		GPIOE RST	GPIOD RST	GPIOC RST	GPIOB RST	GPIOA RST	EXTI RST	AFIO RST
rw	rw	rw	rw	rw	rw	rw	res		res	rw	rw	rw	rw	res	rw

位 31：27	保留，始终读为 0。

位 26	**UART8RST**：UART8 复位（UART8 reset） 由软件置'1'或清'0' 0：无作用； 1：复位 UART8。
位 25	**UART7RST**：UART7 复位（UART7 reset） 由软件置'1'或清'0' 0：无作用； 1：复位 UART7。
位 24	**UART6RST**：UART6 复位（UART6 reset） 由软件置'1'或清'0' 0：无作用； 1：复位 UART6。
位 23	**I2C3RST**：I2C3 复位（I2C3reset） 由软件置'1'或清'0' 0：无作用； 1：复位 I2C3。
位 22	**ACCRST**：ACC 复位（ACC reset） 由软件置'1'或清'0' 0：无作用； 1：复位 ACC 定时器。
位 21	**TMR11RST**：TMR11 定时器复位（TMR11 timer reset） 由软件置'1'或清'0' 0：无作用； 1：复位 TMR11 定时器。
位 20	**TMR10RST**：TMR10 定时器复位（TMR10 timer reset） 由软件置'1'或清'0' 0：无作用； 1：复位 TMR10 定时器。
位 19	**TMR9RST**：TMR9 定时器复位（TMR9 timer reset） 由软件置'1'或清'0' 0：无作用； 1：复位 TMR9 定时器。
位 18:16	保留，始终读为 0。
位 15	**ADC3RST**：ADC3 接口复位（ADC3 interface reset） 由软件置'1'或清'0' 0：无作用； 1：复位 ADC3 接口。
位 14	**USART1RST**：USART1 复位（USART1 reset） 由软件置'1'或清'0' 0：无作用； 1：复位 USART1。
位 13	**TMR8RST**：TMR 8 定时器复位（TMR8 timer reset） 由软件置'1'或清'0' 0：无作用； 1：复位 TMR 8 定时器。
位 12	**SPI1RST**：SPI1 复位（SPI 1 reset） 由软件置'1'或清'0' 0：无作用； 1：复位 SPI1。

位 11	**TMR1RST**：TMR1 定时器复位（TMR1 timer reset） 由软件置'1'或清'0' 0：无作用； 1：复位 TMR 1 定时器。
位 10	**ADC2RST**：ADC2 接口复位（ADC 2 interface reset） 由软件置'1'或清'0' 0：无作用； 1：复位 ADC2 接口。
位 9	**ADC1RST**：ADC1 接口复位（ADC 1 interface reset） 由软件置'1'或清'0' 0：无作用； 1：复位 ADC1 接口。
位 8:7	保留，始终读为 0。
位 6	**GPIOERST**：IO 端口 E 复位（IO port E reset） 由软件置'1'或清'0' 0：无作用； 1：复位 IO 端口 E。
位 5	**GPIODRST**：IO 端口 D 复位（IO port D reset） 由软件置'1'或清'0' 0：无作用； 1：复位 IO 端口 D。
位 4	**GPIOCRST**：IO 端口 C 复位（IO port C reset） 由软件置'1'或清'0' 0：无作用； 1：复位 IO 端口 C。
位 3	**GPIOBRST**：IO 端口 B 复位（IO port B reset） 由软件置'1'或清'0' 0：无作用； 1：复位 IO 端口 B。
位 2	**GPIOARST**：IO 端口 A 复位（IO port A reset） 由软件置'1'或清'0' 0：无作用； 1：复位 IO 端口 A。
位 1	**EXTIRST**：EXTI 复位（EXTI reset） 由软件置'1'或清'0' 0：无作用； 1：复位 EXTI。 **注意：该控制为只写位，读一直为 0**
位 0	**AFIORST**：辅助功能 IO 复位（Alternate function I/O reset） 由软件置'1'或清'0' 0：无作用； 1：复位辅助功能。

3.3.5 APB1外设复位寄存器（RCC_APB1RST）

偏移地址：0x10

复位值：0x0000 0000

访问：无等待周期，字，半字和字节访问

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留		DAC RST	PWR RST	BKP RST	CAN2 RST	CAN RST	保留	USB RST	I2C2 RST	I2C1 RST	UART5 RST	UART4 RST	USART3 RST	USART2 RST	SPI4 RST
res		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPI3 RST	SPI2 RST	保留		WWDG RST	保留		TMR14 RST	TMR13 RST	TMR12 RST	TMR7 RST	TMR6 RST	TMR5 RST	TMR4 RST	TMR3 RST	TMR2 RST
rw	rw	res		rw	res		rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:30	保留，始终读为 0。
位 29	**DACRST**：DAC 接口复位（DAC interface reset） 由软件置’1’或清’0’ 0：无作用； 1：复位 DAC 接口。
位 28	**PWRRST**：电源接口复位（Power interface reset） 由软件置’1’或清’0’ 0：无作用； 1：复位电源接口。
位 27	**BKPRST**：备份接口复位（Backup interface reset） 由软件置’1’或清’0’ 0：无作用； 1：复位备份接口。
位 26	**CAN2RST**：CAN2 复位（CAN 2reset） 由软件置’1’或清’0’ 0：无作用； 1：复位 CAN2。
位 25	**CANRST**：CAN 复位（CAN reset） 由软件置’1’或清’0’ 0：无作用； 1：复位 CAN。
位 24	保留，始终读为 0。
位 23	**USBRST**：USB 复位（USB reset） 由软件置’1’或清’0’ 0：无作用； 1：复位 USB。
位 22	**I2C2RST**：I2C 2 复位（I2C 2 reset） 由软件置’1’或清’0’ 0：无作用； 1：复位 I2C 2。
位 21	**I2C1RST**：I2C 1 复位（I2C 1 reset） 由软件置’1’或清’0’ 0：无作用； 1：复位 I2C 1。
位 20	**UART5RST**：UART5 复位（UART 5 reset） 由软件置’1’或清’0’ 0：无作用； 1：复位 UART5。
位 19	**UART4RST**：UART4 复位（UART 4 reset） 由软件置’1’或清’0’ 0：无作用； 1：复位 UART4。
位 18	**USART3RST**：USART3 复位（USART 3 reset） 由软件置’1’或清’0’ 0：无作用； 1：复位 USART3。

位 17	**USART2RST**：USART2 复位（USART 2 reset） 由软件置'1'或清'0' 0：无作用； 1：复位 USART2。
位 16	**SPI4RST** SPI4 复位（SPI 4 reset） 由软件置'1'或清'0' 0：无作用； 1：复位 SPI4。
位 15	**SPI3RST** SPI3 复位（SPI 3 reset） 由软件置'1'或清'0' 0：无作用； 1：复位 SPI3。
位 14	**SPI2RST**：SPI2 复位（SPI 2 reset） 由软件置'1'或清'0' 0：无作用； 1：复位 SPI2。
位 13：12	保留，始终读为 0。
位 11	**WWDGRST**：窗口看门狗复位（WCNTRindow watchdog reset） 由软件置'1'或清'0' 0：无作用； 1：复位窗口看门狗。
位 10：9	保留，始终读为 0。
位 8	**TMR14RST**：定时器 14 复位（Timer 14 reset） 由软件置'1'或清'0' 0：无作用； 1：复位 TMR 14 定时器。
位 7	**TMR13RST**：定时器 13 复位（Timer 13 reset） 由软件置'1'或清'0' 0：无作用； 1：复位 TMR 13 定时器。
位 6	**TMR12RST**：定时器 12 复位（Timer 12 reset） 由软件置'1'或清'0' 0：无作用； 1：复位 TMR 12 定时器。
位 5	**TMR7RST**：定时器 7 复位（Timer 7 reset） 由软件置'1'或清'0' 0：无作用； 1：复位 TMR 7 定时器。
位 4	**TMR6RST**：定时器 6 复位（Timer 6 reset） 由软件置'1'或清'0' 0：无作用； 1：复位 TMR 6 定时器。
位 3	**TMR5RST**：定时器 5 复位（Timer 5 reset） 由软件置'1'或清'0' 0：无作用； 1：复位 TMR 5 定时器。
位 2	**TMR4RST**：定时器 4 复位（Timer 4 reset） 由软件置'1'或清'0' 0：无作用； 1：复位 TMR 4 定时器。
位 1	**TMR3RST**：定时器 3 复位（Timer 3 reset） 由软件置'1'或清'0' 0：无作用； 1：复位 TMR 3 定时器。

位 0	**TMR2RST**：定时器 2 复位（Timer 2 reset） 由软件置'1'或清'0' 0：无作用； 1：复位 TMR 2 定时器。

3.3.6 AHB外设时钟使能寄存器（RCC_AHBEN）

偏移地址：0x14

复位值：0x0000 0014

访问：无等待周期，字、半字和字节访问

注意：当外设时钟没有启用时，软件不能读出外设寄存器的数值，返回的数值始终是 0x0。

AT32F403A 该寄存器定义如下：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															
res															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留				SDIO2EN	SDIO1EN	保留	XMCEN	保留	CCE	保留	FLASHEN	保留	SRAMEN	DMA2EN	DMA1EN
res				rw	rw	res	rw	res	rw	res	rw	res	rw	rw	rw

位 31：12	保留，始终读为 0。
位 11	**SDIO2EN**：SDIO2 时钟使能（SDIO2 clock enable） 由软件置'1'或清'0'。 0：SDIO2 时钟关闭； 1：SDIO2 时钟开启。
位 10	**SDIO1EN**：SDIO1 时钟使能（SDIO1 clock enable） 由软件置'1'或清'0'。 0：SDIO1 时钟关闭； 1：SDIO1 时钟开启。
位 9	保留，始终读为 0。
位 8	**XMCEN**：XMC 时钟使能（XMC clock enable） 由软件置'1'或清'0'。 0：XMC 时钟关闭； 1：XMC 时钟开启。
位 7	保留，始终读为 0。
位 6	**CCE**：CRC 时钟使能（CRC clock enable） 由软件置'1'或清'0'。 0：CRC 时钟关闭； 1：CRC 时钟开启。
位 5	保留，始终读为 0。
位 4	**FLASHEN**：闪存接口电路时钟使能（FLITF clock enable） 由软件置'1'或清'0'来开启或关闭睡眠模式时闪存接口电路时钟。 0：睡眠模式时闪存接口电路时钟关闭； 1：睡眠模式时闪存接口电路时钟开启。
位 3	保留，始终读为 0。
位 2	**SRAMEN**：SRAM 时钟使能（SRAM interface clock enable） 由软件置'1'或清'0'来开启或关闭睡眠模式时 SRAM 时钟。 0：睡眠模式时 SRAM 时钟关闭； 1：睡眠模式时 SRAM 时钟开启。

位 1	**DMA2EN**：DMA2 时钟使能（DMA2 clock enable） 由软件置'1'或清'0'。 0：DMA2 时钟关闭； 1：DMA2 时钟开启。
位 0	**DMA1EN**：DMA1 时钟使能（DMA1 clock enable） 由软件置'1'或清'0'。 0：DMA1 时钟关闭； 1：DMA1 时钟开启。

AT32F407 该寄存器定义如下：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留			ETHMACPTPEN	保留											ETHMACRXEN
res			rw	res											rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ETHMACTXEN	ETHMACEN	保留		SDIO2EN	SDIO1EN	保留	XMCEN	保留	CCE	保留	FLASHEN	保留	SRAMEN	DMA2EN	DMA1EN
res	res	res		rw	rw	res	rw	res	rw	res	rw	res	rw	rw	rw

位 31：29	保留，始终读为 0。
位 28	ETHMACPTPEN：以太网 MAC PTP 时钟使能 (Ethernet MAC PTP clock enable) 0：以太网 MAC 接收时钟关闭； 1：以太网 MAC 接收时钟开启。
位 27：17	保留，始终读为 0。
位 16	**ETHMACRXEN**：以太网 MAC 接收时钟使能 (Ethernet MAC RX clock enable) 0：以太网 MAC 接收时钟关闭； 1：以太网 MAC 接收时钟开启。 注：在 RMII 模式下，如果使能了这个时钟，MAC 的 RMII 时钟也被使能。
位 15	**ETHMACTXEN**：以太网 MAC 发送时钟使能 (Ethernet MAC TX clock enable) 由软件置'1'或清'0'。 0：以太网 MAC 发送时钟关闭； 1：以太网 MAC 发送时钟开启。 注：在 RMII 模式下，如果使能了这个时钟，MAC 的 RMII 时钟也被使能。
位 14	**ETHMACEN**：以太网 MAC 时钟使能 (Ethernet MAC TX clock enable) 由软件置'1'或清'0'。在使能 MAC 时钟之前必须先选定 PHY 接口(MII/RMII)。 0：以太网 MAC 时钟关闭； 1：以太网 MAC 时钟开启。
位 13：12	保留，始终读为 0。
位 11	**SDIO2EN**：SDIO2 时钟使能（SDIO2 clock enable） 由软件置'1'或清'0'。 0：SDIO2 时钟关闭； 1：SDIO2 时钟开启。
位 10	**SDIO1EN**：SDIO1 时钟使能（SDIO1 clock enable） 由软件置'1'或清'0'。 0：SDIO1 时钟关闭； 1：SDIO1 时钟开启。
位 9	保留，始终读为 0。
位 8	**XMCEN**：XMC 时钟使能（XMC clock enable） 由软件置'1'或清'0'。 0：XMC 时钟关闭； 1：XMC 时钟开启。
位 7	保留，始终读为 0。

位 6	**CCE**：CRC 时钟使能（CRC clock enable） 由软件置'1'或清'0'。 0：CRC 时钟关闭； 1：CRC 时钟开启。
位 5	保留，始终读为 0。
位 4	**FLASHEN**：闪存接口电路时钟使能（FLITF clock enable） 由软件置'1'或清'0'来开启或关闭睡眠模式时闪存接口电路时钟。 0：睡眠模式时闪存接口电路时钟关闭； 1：睡眠模式时闪存接口电路时钟开启。
位 3	保留，始终读为 0。
位 2	**SRAMEN**：SRAM 时钟使能（SRAM interface clock enable） 由软件置'1'或清'0'来开启或关闭睡眠模式时 SRAM 时钟。 0：睡眠模式时 SRAM 时钟关闭； 1：睡眠模式时 SRAM 时钟开启。
位 1	**DMA2EN**：DMA2 时钟使能（DMA2 clock enable） 由软件置'1'或清'0'。 0：DMA2 时钟关闭； 1：DMA2 时钟开启。
位 0	**DMA1EN**：DMA1 时钟使能（DMA1 clock enable） 由软件置'1'或清'0'。 0：DMA1 时钟关闭； 1：DMA1 时钟开启。

3.3.7 APB2外设时钟使能寄存器（RCC_APB2EN）

偏移地址：0x18

复位值：0x0000 0000

访问：字，半字和字节访问

通常无访问等待周期。

但在 APB2 总线上的外设被访问时，将插入等待状态直到 APB2 的外设访问结束。

注意：当外设时钟没有启用时，软件不能读出外设寄存器的数值，返回的数值始终是 0x0。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留					UART8 EN	UART7 EN	USART6 EN	I2C3 EN	ACC EN	TMR11 EN	TMR10 EN	TMR9 EN	保留		
res					rw	rw	rw	rw	rw	rw	rw	rw	res		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADC3 EN	USART1 EN	TMR8 EN	SPI1 EN	TMR1 EN	ADC2 EN	ADC1 EN	保留		GPIOE EN	GPIOD EN	GPIOC EN	GPIOB EN	GPIOA EN	保留	AFIO EN
res	rw	rw	rw	rw	rw	rw	res		rw	rw	rw	rw	rw	res	rw

位 31：27	保留，始终读为 0。
位 26	**UART8EN**：UART8 时钟使能（UART8 clock enable） 由软件置'1'或清'0' 0：UART8 时钟关闭； 1：UART8 时钟开启。
位 25	**UART7EN**：UART7 时钟使能（UART7 clock enable） 由软件置'1'或清'0' 0：UART7 时钟关闭； 1：UART7 时钟开启。

位 24	**USART6EN**：USART6 时钟使能（USART6 clock enable） 由软件置'1'或清'0' 0：USART6 时钟关闭； 1：USART6 时钟开启。
位 23	**I2C3EN**：I2C3 时钟使能（I2C 3 clock enable） 由软件置'1'或清'0' 0：I2C 3 时钟关闭； 1：I2C 3 时钟开启。
位 22	**ACCEN**: ACC 时钟使能（ACC clock enable） 由软件置'1'或清'0' 0：ACC 时钟关闭； 1：ACC 时钟开启。
位 21	**TMR11EN**：TMR11 定时器时钟使能（Timer 11 clock enable） 由软件置'1'或清'0' 0：TMR11 定时器时钟关闭； 1：TMR11 定时器时钟开启。
位 20	**TMR10EN**：TMR10 定时器时钟使能（Timer 10 clock enable） 由软件置'1'或清'0' 0：TMR10 定时器时钟关闭； 1：TMR10 定时器时钟开启。
位 19	**TMR9EN**：TMR9 定时器时钟使能（Timer 9 clock enable） 由软件置'1'或清'0' 0：TMR9 定时器时钟关闭； 1：TMR9 定时器时钟开启。
位 18：16	保留，始终读为 0。
位 15	**ADC3EN**：ADC3 接口时钟使能（ADC 3 interface clock enable） 由软件置'1'或清'0' 0：ADC3 接口时钟关闭； 1：ADC3 接口时钟开启。
位 14	**USART1EN**：USART1 时钟使能（USART1 clock enable） 由软件置'1'或清'0' 0：USART1 时钟关闭； 1：USART1 时钟开启。
位 13	**TMR8EN**：TMR8 定时器时钟使能（Timer 8 clock enable） 由软件置'1'或清'0' 0：TMR8 定时器时钟关闭； 1：TMR8 定时器时钟开启。
位 12	**SPI1EN**：SPI1 时钟使能（SPI 1 clock enable） 由软件置'1'或清'0' 0：SPI1 时钟关闭； 1：SPI1 时钟开启。
位 11	**TMR1EN**：TMR1 定时器时钟使能（Timer 1 clock enable） 由软件置'1'或清'0' 0：TMR1 定时器时钟关闭； 1：TMR1 定时器时钟开启。
位 10	**ADC2EN**：ADC2 接口时钟使能（ADC 2 interface clock enable） 由软件置'1'或清'0' 0：ADC2 接口时钟关闭； 1：ADC2 接口时钟开启。

位 9	**ADC1EN**：ADC1 接口时钟使能（ADC 1 interface clock enable） 由软件置'1'或清'0' 0：ADC1 接口时钟关闭； 1：ADC1 接口时钟开启。
位 8：7	保留，始终读为 0。
位 6	**GPIOEEN**：IO 端口 E 时钟使能（I/O port E clock enable） 由软件置'1'或清'0' 0：IO 端口 E 时钟关闭； 1：IO 端口 E 时钟开启。
位 5	**GPIODEN**：IO 端口 D 时钟使能（I/O port D clock enable） 由软件置'1'或清'0' 0：IO 端口 D 时钟关闭； 1：IO 端口 D 时钟开启。
位 4	**GPIOCEN**：IO 端口 C 时钟使能（I/O port C clock enable） 由软件置'1'或清'0' 0：IO 端口 C 时钟关闭； 1：IO 端口 C 时钟开启。
位 3	**GPIOBEN**：IO 端口 B 时钟使能（I/O port B clock enable） 由软件置'1'或清'0' 0：IO 端口 B 时钟关闭； 1：IO 端口 B 时钟开启。
位 2	**GPIOAEN**：IO 端口 A 时钟使能（I/O port A clock enable） 由软件置'1'或清'0' 0：IO 端口 A 时钟关闭； 1：IO 端口 A 时钟开启。
位 1	保留，始终读为 0。
位 0	**AFIOEN**：辅助功能 IO 时钟使能（Alternate function I/O clock enable） 由软件置'1'或清'0' 0：辅助功能 IO 时钟关闭； 1：辅助功能 IO 时钟开启。

3.3.8 APB1外设时钟使能寄存器（RCC_APB1EN）

偏移地址：0x1C

复位值：0x0000 0000

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。

注意： 当外设时钟没有启用时，软件不能读出外设寄存器的数值，返回的数值始终是 0x0。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留		DAC EN	PWR EN	BKP EN	CAN2 EN	CAN EN	保留	USB EN	I2C2 EN	I2C1 EN	UART5 EN	UART4 EN	USART3 EN	USART2 EN	SPI4 EN
res		rw	rw	rw	rw	rw	res	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPI3 EN	SPI2 EN	保留		WWDG EN	保留		TMR14 EN	TMR13 EN	TMR12 EN	TMR7 EN	TMR6 EN	TMR5 EN	TMR4 EN	TMR3 EN	TMR2 EN
rw	rw	res		rw	res		rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31：30	保留，始终读为 0。

位 29	**DACEN**： DAC 接口时钟使能（DAC interface clock enable） 由软件置'1'或清'0' 0：DAC 接口时钟关闭； 1：DAC 接口时钟开启。
位 28	**PWREN**：电源接口时钟使能（Power interface clock enable） 由软件置'1'或清'0' 0：电源接口时钟关闭； 1：电源接口时钟开启。
位 27	**BKPEN**：备份接口时钟使能（Backup interface clock enable） 由软件置'1'或清'0' 0：备份接口时钟关闭； 1：备份接口时钟开启。
位 26	**CAN2EN**：CAN2 时钟使能（CAN2 clock enable） 由软件置'1'或清'0' 0：CAN2 时钟关闭； 1：CAN2 时钟开启。
位 25	**CAN1EN**：CAN1 时钟使能（CAN1 clock enable） 由软件置'1'或清'0' 0：CAN1 时钟关闭； 1：CAN1 时钟开启。
位 24	保留，始终读为 0。
位 23	**USBEN**：USB 时钟使能（USB clock enable） 由软件置'1'或清'0' 0：USB 时钟关闭； 1：USB 时钟开启。
位 22	**I2C2EN**：I2C 2 时钟使能（I2C 2 clock enable） 由软件置'1'或清'0' 0：I2C 2 时钟关闭； 1：I2C 2 时钟开启。
位 21	**I2C1EN**：I2C 1 时钟使能（I2C 1 clock enable） 由软件置'1'或清'0' 0：I2C 1 时钟关闭； 1：I2C 1 时钟开启。
位 20	**UART5EN**：UART5 时钟使能（UART 5 clock enable） 由软件置'1'或清'0' 0：UART5 时钟关闭； 1：UART5 时钟开启。
位 19	**UART4EN**：UART4 时钟使能（UART 4 clock enable） 由软件置'1'或清'0' 0：UART4 时钟关闭； 1：UART4 时钟开启。
位 18	**USART3EN**：USART3 时钟使能（USART 3 clock enable） 由软件置'1'或清'0' 0：USART3 时钟关闭； 1：USART3 时钟开启。
位 17	**USART2EN**：USART2 时钟使能（USART 2 clock enable） 由软件置'1'或清'0' 0：USART2 时钟关闭； 1：USART2 时钟开启。

位 16	**SPI4EN**：SPI 4 时钟使能（SPI 4 clock enable） 由软件置'1'或清'0' 0：SPI 4 时钟关闭； 1：SPI 4 时钟开启。
位 15	**SPI3EN**：SPI 3 时钟使能（SPI 3 clock enable） 由软件置'1'或清'0' 0：SPI 3 时钟关闭； 1：SPI 3 时钟开启。
位 14	**SPI2EN**：SPI 2 时钟使能（SPI 2 clock enable） 由软件置'1'或清'0' 0：SPI 2 时钟关闭； 1：SPI 2 时钟开启。
位 13：12	保留，始终读为 0。
位 11	**WWDGEN**：窗口看门狗时钟使能（Window watchdog clock enable） 由软件置'1'或清'0' 0：窗口看门狗时钟关闭； 1：窗口看门狗时钟开启。
位 10：9	保留，始终读为 0。
位 8	**TMR14EN**：定时器 14 时钟使能（Timer14 clock enable） 由软件置'1'或清'0' 0：定时器 14 时钟关闭； 1：定时器 14 时钟开启。
位 7	**TMR13EN**：定时器 13 时钟使能（Timer13 clock enable） 由软件置'1'或清'0' 0：定时器 13 时钟关闭； 1：定时器 13 时钟开启。
位 6	**TMR12EN**：定时器 12 时钟使能（Timer12 clock enable） 由软件置'1'或清'0' 0：定时器 12 时钟关闭； 1：定时器 12 时钟开启。
位 5	**TMR7EN**：定时器 7 时钟使能（Timer 7 clock enable） 由软件置'1'或清'0' 0：定时器 7 时钟关闭； 1：定时器 7 时钟开启。
位 4	**TMR6EN**：定时器 6 时钟使能（Timer 6 clock enable） 由软件置'1'或清'0' 0：定时器 6 时钟关闭； 1：定时器 6 时钟开启。
位 3	**TMR5EN**：定时器 5 时钟使能（Timer 5 clock enable） 由软件置'1'或清'0' 0：定时器 5 时钟关闭； 1：定时器 5 时钟开启。
位 2	**TMR4EN**：定时器 4 时钟使能（Timer 4 clock enable） 由软件置'1'或清'0' 0：定时器 4 时钟关闭； 1：定时器 4 时钟开启。

位 1	**TMR3EN**：定时器 3 时钟使能（Timer 3 clock enable） 由软件置’1’或清’0’ 0：定时器 3 时钟关闭； 1：定时器 3 时钟开启。
位 0	**TMR2EN**：定时器 2 时钟使能（Timer 2 clock enable） 由软件置’1’或清’0’ 0：定时器 2 时钟关闭； 1：定时器 2 时钟开启。

3.3.9 备份域控制寄存器（RCC_BDC）

偏移地址：0x20

复位值：0x0000 0000，只能由备份域复位有效复位

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： *备份域控制寄存器中（RCC_BDC）LSEEN、LSEBYPS、RTCSEL 和 RTCEN 位处于备份域。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（PWR_CTRL）中的 DBP 位置’1’后才能对这些位进行改动。进一步信息请参考 4.1 节。这些位只能由备份域复位清除（见 3.1.3 节）。任何内部或外部复位都不会影响这些位。*

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															BDRST
res															rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RTCEN	保留					RTCSEL[1:0]		保留					LSE BYPS	LSE STBL	LSEEN
rw	res					rw	rw	res					rw	r	rw

位 31：17	保留，始终读为 0。
位 16	**BDRST**：备份域软件复位（Backup domain software reset） 由软件置’1’或清’0’ 0：复位未激活； 1：复位整个备份域。
位 15	**RTCEN**：RTC 时钟使能（RTC clock enable） 由软件置’1’或清’0’ 0：RTC 时钟关闭； 1：RTC 时钟开启。
位 14：10	保留，始终读为 0。
位 9：8	**RTCSEL[1：0]**：RTC 时钟源选择（RTC clock source selection） 由软件设置来选择 RTC 时钟源。一旦 RTC 时钟源被选定，直到下次后备域被复位，它不能在被 改变。可通过设置 BDRST 位来清除。 00：无时钟； 01：LSE 振荡器作为 RTC 时钟； 10：LSI 振荡器作为 RTC 时钟； 11：HSE 振荡器在 128 分频后作为 RTC 时钟。
位 7：3	保留，始终读为 0。

位 2	**LSEBYPS**：外部低速时钟振荡器旁路（External low-speed oscillator bypass） 在调试模式下由软件置'1'或清'0'来旁路 LSE。只有在外部 32kHz 振荡器关闭时，才能写入该位 0：LSE 时钟未被旁路； 1：LSE 时钟被旁路。
位 1	**LSESTBL**：外部低速 LSE 就绪（External low-speed oscillator ready） 由硬件置'1'或清'0'来指示是否外部 32kHz 振荡器就绪。在 LSEEN 被清零后，该位需要 6 个外部 低速振荡器的周期才被清零。 0：外部 32kHz 振荡器未就绪； 1：外部 32kHz 振荡器就绪。
位 0	**LSEEN**：外部低速振荡器使能（External low-speed oscillator enable） 由软件置'1'或清'0' 0：外部 32kHz 振荡器关闭； 1：外部 32kHz 振荡器开启。

3.3.10 控制/状态寄存器（RCC_CTRLSTS）

偏移地址：0x24

复位值：0x0C00 0000，除复位标志外由系统复位清除，复位标志只能由电源复位清除。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。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LP RSTF	WWDG RSTF	IWDG RSTF	SW RSTF	POR RSTF	PIN RSTF	保留	RST FC	保留							
rw	rw	rw	rw	rw	rw	rw	rw	res							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留														LSI STBL	LSI EN
res														r	rw

位 31	**LPRSTF**：低功耗复位标志（Low-power reset flag） 在低功耗管理复位发生时由硬件置'1'；由软件通过写 RSTFC 位清除。 0：无低功耗管理复位发生； 1：发生低功耗管理复位。 关于低功耗管理复位的详细信息，请参考 3.1.1 节的"低功耗管理复位"。
位 30	**WWDGRSTF**：窗口看门狗复位标志（Window watchdog reset flag） 在窗口看门狗复位发生时由硬件置'1'；由软件通过写 RSTFC 位清除。 0：无窗口看门狗复位发生； 1：发生窗口看门狗复位。
位 29	**IWDGRSTF**：独立看门狗复位标志（Independent watchdog reset flag） 在独立看门狗复位发生在 VDD 区域时由硬件置'1'；由软件通过写 RSTFC 位清除。 0：无独立看门狗复位发生； 1：发生独立看门狗复位。
位 28	**SWRSTF**：软件复位标志（Software reset flag） 在软件复位发生时由硬件置'1'；由软件通过写 RSTFC 位清除。 0：无软件复位发生； 1：发生软件复位。
位 27	**PORRSTF**：上电/掉电复位标志（POR/PDR reset flag） 在上电/掉电复位发生时由硬件置'1'；由软件通过写 RSTFC 位清除。 0：无上电/掉电复位发生； 1：发生上电/掉电复位。

位 26	**PINRSTF**：NRST 引脚复位标志（PIN reset flag） 在 NRST 引脚复位发生时由硬件置'1'；由软件通过写 RSTFC 位清除。 0：无 NRST 引脚复位发生； 1：发生 NRST 引脚复位。
位 25	保留，读操作返回 0
位 24	**RSTFC**：清除复位标志（Reset flag clear） 由软件置'1'来清除复位标志。 0：无作用； 1：清除复位标志。
位 23：2	保留，读操作返回 0
位 1	**LSISTBL**：内部低速振荡器就绪（Internal low-speed oscillator ready） 由硬件置'1 或清'0'来指示内部 40kHz RC 振荡器是否就绪。在 LSIEN 清零后，3 个内部 40kHz RC 振荡器的周期后 LSISTBL 被清零。 0：内部 40kHz RC 振荡器时钟未就绪； 1：内部 40kHz RC 振荡器时钟就绪。
位 0	**LSIEN**：内部低速振荡器使能（Internal low-speed oscillator enable） 由软件置'1 或清'0'。 0：内部 40kHz RC 振荡器关闭； 1：内部 40kHz RC 振荡器开启。

3.3.11 AHB外设复位寄存器（RCC_AHBRST）

偏移地址：0x28

复位值：0x0000 0000，除复位标志外由系统复位清除，复位标志只能由电源复位清除。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。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															
res															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留	ETH MAC RST	保留													
res	rw	res													

位 31:15	保留，读操作返回 0
位 14	**ETHMACRST**：以太网 MAC 复位 (Ethernet MAC reset) 由软件置'1'或清'0'。 0：无作用； 1：复位以太网 MAC 模块。
位 13：0	保留，读操作返回 0

3.3.12 额外寄存器（RCC_MISC）

偏移地址：0x30

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MCOPRE				保留		HSI_DIV_EN	USB 768B	保留							CLK OUT[3]
rw				res		rw	rw	res							rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留								HSICAL_KEY[7:0]							
res								rw							

位 31:28	**MCOPRE: CLKOUT 预分频** 0~7: CLKOUT 输出 8: CLKOUT 2 分频输出 9: CLKOUT 4 分频输出 10: CLKOUT 8 分频输出 11: CLKOUT 16 分频输出 12: CLKOUT 64 分频输出 13: CLKOUT 128 分频输出 14: CLKOUT 256 分频输出 15: CLKOUT 512 分频输出
位 27:26	保留，读操作返回 0
位 25	**HSI_DIV_EN: HSI 6 分频使能** 0: HSI 输出 6 分频时钟为 8M 1: HSI 输出不分频时钟为 48M
位 24	**USB768B:** USB 缓冲区大小 0：缓冲区为 512 字节 1：缓冲区为 768~1280 字节
位 23:17	保留，读操作返回 0
位 16	**CLKOUT[3]**：微控制器时钟输出 （Microcontroller clock output） 搭配 RCC_CFG 寄存器位 26:24 使用
位 15:8	保留，读操作返回 0
位 7:0	**HSICAL_KEY[7:0]**：HSICAL 写入键值。 **此字段为 0x5A 时，HSICAL [7:0]才可被写入。**

3.3.13 测试寄存器（RCC_TEST）

偏移地址：0x50

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															MCO2 TMR_EN
res															rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留															
res															

位 31:17	保留，请保持为 0

位 16	**MCO2TMR_EN: CLKOUT 连接到 Timer10 的 CH0 使能位** 0：CLKOUT 未连接到 Timer10 的 CH0 1：CLKOUT 连接到 Timer10 的 CH0
位 15:0	保留，请保持为 0

3.3.14 额外寄存器（RCC_MISC2）

偏移地址：0x54

复位值：0x0000 000D

AT32F403A 位定义如下

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															
res															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留		HSE_DIV_CTRL		保留	保留	HSI_SYS_CTRL	HSI_FOR_USB	保留		AUTO_STEP_EN		保留			
res		rw		res	res	rw	rw	res		rw		res			

位 31:14	保留，读操作返回 0
位 13：12	HSE_DIV_CTRL: HSE 时钟分频控制位 00：2 分频 01：3 分频 10：4 分频 11：5 分频 注意：请在 HSE 时钟稳定后再配置分频控制位，否则会忽略当前这次的配置。
位 11：10	保留，读操作返回 0
位 9	**HSI_SYS_CTRL: HSI 作为系统时钟的频率选择位** 当 SYSCLKSEL 选择 HSI 为时钟源时，SYSCLK 的频率为 0: 固定是 8Mhz，即选择 HSI 时钟的 6 分频 1: 根据 HSI DIV EN 设定可能为 48M 或 8M
位 8	**HSI_FOR_USB: USB 48M 时钟源选择位** 0: USB 48M 时钟源是 PLL 或是其分频 1: USB 48M 时钟源来自 HSI 或是其 6 分频，**由于 USB 必须工作在 48M，此时必须保证 HSI_DIV_EN=1，使 USB 48M 时钟选择 HSI 的 48MHz 输出.**
位 7：6	保留，读操作返回 0
位 5：4	**AUTO_STEP_EN: 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 AHBPSC 或 SYSCLKSEL 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

AT32F407 位定义如下

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															
res															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EMAC_PPS_SEL	保留	HSE_DIV_CTRL		保留	保留	HSI_SYS_CTRL	HSI_FOR_USB	保留		AUTO_STEP_EN		保留			
rw	res	rw		res	res	rw	rw	res		rw		res			

位 31:16	保留，读操作返回 0
位 15	EMAC_PPS_SEL：ETH MAC PPS 输出脉宽选择位 0：输出脉宽为 125mS 1：输出脉宽为 1 个系统时钟
位 14	保留，读操作返回 0
位 13：12	HSE_DIV_CTRL: HSE 时钟分频控制位 00：2 分频 01：3 分频 10：4 分频 11：5 分频 注意：请在 HSE 时钟稳定后再配置分频控制位，否则会忽略当前这次的配置。
位 11：10	保留，读操作返回 0
位 9	**HSI_SYS_CTRL: HSI 作为系统时钟的频率选择位** 当 SYSCLKSEL 选择 HSI 为时钟源时，SYSCLK 的频率为 0: 固定是 8Mhz，即选择 HSI 时钟的 6 分频 1: 根据 HSI DIV EN 设定可能为 48M 或 8M
位 8	**HSI_FOR_USB: USB 48M 时钟源选择位** 0: USB 48M 时钟源是 PLL 或是其分频 1: USB 48M 时钟源来自 HSI 或是其 6 分频，**由于 USB 必须工作在 48M，此时必须保证 HSI_DIV_EN=1，使 USB 48M 时钟选择 HSI 的 48MHz 输出.**
位 7：6	保留，读操作返回 0
位 5：4	**AUTO_STEP_EN: 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 AHBPSC 或 SYSCLKSEL 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

3.3.15 额外寄存器（RCC_INTCTRL）

偏移地址：0x5C

复位值：0x0000 0000



位 31:1	保留，读操作返回 0
位 0	**USB_INT_CTRL:USBDEV 模块中断重映射控制位** 0：USBDEV 的 USB_HP 使用 19 中断，USB_LP 使用 20 号中断 1：USBDEV 的 USB_HP 使用 73 中断，USB_LP 使用 74 号中断

4 备份寄存器（BKPR）

4.1 BKPR简介

备份寄存器是 42 个 16 位的寄存器，可用来存储 84 个字节的用户应用程序数据。这些寄存器处在备份域里，这些寄存器由 VDD/VBAT 维持供电。当系统在待机模式下被唤醒，或系统复位或电源复位时，这些寄存器也不会被复位。

此外，BKPR 控制寄存器用来管理侵入检测和 RTC 校准功能。复位后，对备份寄存器和 RTC 的访问被禁止，并且备份域被保护以防止可能存在的意外的写操作。执行以下操作可以使能对备份寄存器和 RTC 的访问。

- 通过设置寄存器 RCC_APB1EN 的 PWREN 和 BKPEN 位来打开电源和后备接口的时钟。
- 电源控制寄存器（PWR_CTRL）的 DBP 位来使能对后备寄存器和 RTC 的访问。

4.2 BKPR特性

- 84字节数据后备寄存器
- 用来管理防侵入检测并具有中断功能的状态/控制寄存器
- 用来存储RTC校验值的校验寄存器
- 在PC13引脚（当该引脚不用于侵入检测时）上输出RTC校准时钟，RTC闹钟信号或者秒信号。

4.3 BKPR功能描述

4.3.1 侵入检测

当 TAMPER 引脚上的信号从 0 变成 1 或者从 1 变成 0（取决于备份控制寄存器 BKPR_CTRL 的 TPALV 位），会产生一个侵入检测事件。侵入检测事件将所有数据备份寄存器内容清除。

然而为了避免丢失侵入事件，侵入检测信号是边沿检测的信号与侵入检测允许位的逻辑与，从而在侵入检测引脚被允许前发生的侵入事件也可以被检测到。

- 当TPALV=0时：如果在启动侵入检测TAMPER引脚前（通过设置TPEN位）该引脚已经为高电平，一旦启动侵入检测功能，则会产生一个额外的侵入事件（尽管在TPEN位置’1’后并没有出现上升沿）。
- 当TPALV=1时：如果在启动侵入检测引脚TAMPER前（通过设置TPEN位）该引脚已经为低电平，一旦启动侵入检测功能，则会产生一个额外的侵入事件（尽管在TPEN位置’1’后并没有出现下降沿）。

设置 BKPR_CTRLSTS 寄存器的 TPIEN 位为’1’，当检测到侵入事件时就会产生一个中断。在一个侵入事件被检测到并被清除后，侵入检测引脚 TAMPER 应该被禁止。然后，在再次写入备份数据寄存器前重新用 TPEN 位启动侵入检测功能。这样，可以阻止软件在侵入检测引脚上仍然有侵入事件时对备份数据寄存器进行写操作。这相当于对侵入引脚 TAMPER 进行电平检测。

注意：当 VDD 电源断开时，侵入检测功能仍然有效。为了避免不必要的复位数据备份寄存器，TAMPER 引脚应该在片外连接到正确的电平。

4.3.2 RTC校准

为方便测量，RTC 时钟可以经 64 分频输出到侵入检测引脚 TAMPER 上。通过设置 RTC 校验寄存器（BKPR_RTCCAL）的 OT1CAL 位来开启这一功能。

通过配置 CAL[6：0]位，此时钟可以最多减慢 121ppm。

4.4 BKPR寄存器描述

可以用半字（16 位）或字（32 位）的方式操作这些外设寄存器。

BKPR 寄存器是 16 位的可寻址寄存器。

表 4-1 BKPR寄存器映像和复位值

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	保留																																
0x04	BKPR_DR1	保留																DT[15：0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08	BKPR_DR2	保留																DT[15：0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	BKPR_DR3	保留																DT[15：0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	BKPR_DR4	保留																DT[15：0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14	BKPR_DR5	保留																DT[15：0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	BKPR_DR6	保留																DT[15：0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1C	BKPR_DR7	保留																DT[15：0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20	BKPR_DR8	保留																DT[15：0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x24	BKPR_DR9	保留																DT[15：0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x28	BKPR_DR10	保留																DT[15：0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x2C	BKPR_ RTCCAL	保留																						OT2SE	OT2SE	OT1CA	CAL[6：0]						
	复位值																							0	0	0	0	0	0	0	0	0	0
0x30	BKPR_CTRL	保留																														TPALV	TPEN
	复位值																															0	0
0x34	BKPR _CTRLSTS	保留																						TPIF	TPEF	保留					TPIEN	CTPIF	CTPEF
	复位值																							0	0						0	0	0
0x38	保留																																
0x3C	保留																																
0x40	BKPR_DR11	保留																DT[15：0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x44	BKPR_DR12	保留																DT[15：0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x48	BKPR_DR13	保留																DT[15：0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x4C	BKPR_DR14	保留																DT[15：0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x50	BKPR_DR15	保留																DT[15：0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

0x54	BKPR_DR16	保留	DT[15：0]													
	复位值		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x58	BKPR_DR17	保留	DT[15：0]													
	复位值		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x5C	BKPR_DR18	保留	DT[15：0]													
	复位值		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x60	BKPR_DR19	保留	DT[15：0]													
	复位值		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x64	BKPR_DR20	保留	DT[15：0]													
	复位值		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x68	BKPR_DR21	保留	DT[15：0]													
	复位值		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x6C	BKPR_DR22	保留	DT[15：0]													
	复位值		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x70	BKPR_DR23	保留	DT[15：0]													
	复位值		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x74	BKPR_DR24	保留	DT[15：0]													
	复位值		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x78	BKPR_DR25	保留	DT[15：0]													
	复位值		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x7C	BKPR_DR26	保留	DT[15：0]													
	复位值		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x80	BKPR_DR27	保留	DT[15：0]													
	复位值		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x84	BKPR_DR28	保留	DT[15：0]													
	复位值		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x88	BKPR_DR29	保留	DT[15：0]													
	复位值		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x8C	BKPR_DR30	保留	DT[15：0]													
	复位值		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x90	BKPR_DR31	保留	DT[15：0]													
	复位值		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x94	BKPR_DR32	保留	DT[15：0]													
	复位值		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x98	BKPR_DR33	保留	DT[15：0]													
	复位值		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x9C	BKPR_DR34	保留	DT[15：0]													
	复位值		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xA0	BKPR_DR35	保留	DT[15：0]													
	复位值		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xA4	BKPR_DR36	保留	DT[15：0]													
	复位值		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xA8	BKPR_DR37	保留	DT[15：0]													
	复位值		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xAC	BKPR_DR38	保留	DT[15：0]													
	复位值		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xB0	BKPR_DR39	保留	DT[15：0]													
	复位值		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

0xB4	BKPR_DR40	保留	DT[15：0]													
	复位值		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xB8	BKPR_DR41	保留	DT[15：0]													
	复位值		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xBC	BKPR_DR42	保留	DT[15：0]													
	复位值		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

4.4.1 备份数据寄存器x（BKPR_DRx）（x = 1 … 42）

地址偏移：0x04 到 0x28，0x40 到 0xBC

复位值：0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DT[15：0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 15： 0	**DT[15：0]**：备份数据 这些位可以被用来写入用户数据。 *注意*：BKPR_DRx 寄存器不会被系统复位、电源复位、从待机模式唤醒所复位。它们可以由备份域复位来复位或（如果侵入检测引脚 TAMPER 功能被开启时）由侵入引脚事件复位。

4.4.2 RTC时钟校准寄存器（BKPR_RTCCAL）

地址偏移：0x2C

复位值：0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留				TOGEN	CCOS	OT2SEL	OT2SEN	OT1CAL	CAL[6：0]						
res				rw	rw	rw	rw	rw	rw						

位 15：12	保留，始终读为 0。
位 11	**TOGEN:** 在 OT2SEN 使能后此位可用来决定闹钟或秒输出型式。 0：闹钟或秒输出为脉冲输出型式（输出脉冲的宽度为一个 RTC 时钟的周期） 1：闹钟或秒输出为翻转输出型式（每来一次闹钟或秒输出事件，相对应管脚翻转一次） 注：该位只能被后备区的复位所清除
位 10	**CCOS:** 选择校准时钟输出时，此位可用来选择是输出校准前的 64 分频输出还是校准后的 64 分频输出。 0：校准前的时钟 64 分频输出 1：校准后的时钟 64 分频输出 注：该位只能被后备区的复位所清除
位 9	**OT2SEL**：闹钟或秒输出选择（Alarm or second output selection） 当设置了 OT2SEL 位，OT2SEL 位可用于选择在 TAMPER 引脚上输出的是 RTC 秒事件还是闹钟事件。 0：输出 RTC 闹钟事件 1：输出秒事件 注：该位只能被后备区的复位所清除
位 8	**OT2SEN**：允许输出闹钟或秒脉冲（Alarm or second output enable） 根据 OT2SEL 位的设置，该位允许 RTC 闹钟或秒事件输出到 TAMPER 引脚上。 设置了 OT2SEN 位时不能开启 TAMPER 的功能。 注：该位只能被后备区的复位所清除

位 7	**OT1CAL**：校准时钟输出（Calibration clock output） 0：无影响 1：此位置 1 可以在侵入检测引脚输出经 64 分频后的 RTC 时钟。当 OT1CAL 位置 1 时，必须关闭侵入检测功能以避免检测到无用的侵入信号。 注：当 VDD 供电断开时，该位被清除。
位 6：0	**CAL[6：0]**：校准值（Calibration value） 校准值表示在每 2^{20} 个时钟脉冲内将有多少个时钟脉冲被跳过。这可以用来对 RTC 进行校准，以 1000000/2^{20}ppm 的比例减慢时钟。RTC 时钟可以被减慢 0~121ppm。

4.4.3 备份控制寄存器（BKPR_CTRL）

偏移地址：0x30

复位值：0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留														TPALV	TPEN
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 15：2	保留，始终读为 0。
位 1	**TPALV**：侵入检测 TAMPER 引脚有效电平（TAMPER pin active level） 0：侵入检测 TAMPER 引脚上的高电平会清除所有数据备份寄存器（如果 TPEN 位为 1） 1：侵入检测 TAMPER 引脚上的低电平会清除所有数据备份寄存器（如果 TPEN 位为 1）
位 0	**TPEN**：启动侵入检测 TAMPER 引脚（TAMPER pin enable） 0：侵入检测 TAMPER 引脚作为通用 IO 口使用 1：开启侵入检测引脚作为侵入检测使用

注意：同时设置 TPALV 和 TPEN 位总是安全的。然而，同时清除两者会产生一个假的侵入事件。因此，推荐只在 TPEN 为 0 时才改变 TPALV 位的状态。

4.4.4 备份控制/状态寄存器（BKPR_CTRLSTS）

偏移地址：0x34

复位值：0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留						TPIF	TPEF	保留					TPIEN	CTPIF	CTPEF
rw	rw	rw	rw	rw	rw	r	r	rw	rw	rw	rw	rw	rw	rw	rw

位 15：10	保留，始终读为 0。
位 9	TPIF：侵入中断标志（Tamper interrupt flag） 当检测到有侵入事件且 TPIEN 位为 1 时，此位由硬件置 1。通过向 CTPIF 位写 1 来清除此标志位（同时也清除了中断）。如果 TPIEN 位被清除，则此位也会被清除。 0：无侵入中断 1：产生侵入中断 *注意*：仅当系统复位或由待机模式唤醒后才复位该位
位 8	TPEF：侵入事件标志（Tamper event flag） 当检测到侵入事件时此位由硬件置 1。通过向 CTPEF 位写 1 可清除此标志位 0：无侵入事件 1：检测到侵入事件 注：侵入事件会复位所有的 BKPR_DRx 寄存器。只要 TPEF 为 1，所有的 BKPR_DRx 寄存器就一直保持复位状态。当此位被置 1 时，若对 BKPR_DRx 进行写操作，写入的值不会被保存。
位 7：3	保留，始终读为 0。

位 2	TPIEN：允许侵入 TAMPER 引脚中断（Tamper pin interrupt enable） 0：禁止侵入检测中断 1：允许侵入检测中断（BKPR_CTRL 寄存器的 TPEN 位也必须被置 1） 注 1：侵入中断无法将系统内核从低功耗模式唤醒。 注 2：仅当系统复位或由待机模式唤醒后才复位该位。
位 1	CTPIF：清除侵入检测中断（Clear tamper interrupt） 此位只能写入，读出值为 0。 0：无效 1：清除侵入检测中断和 TPIF 侵入检测中断标志
位 0	CTPEF：清除侵入检测事件（Clear tamper event） 此位只能写入，读出值为 0。 0：无效 1：清除 TPEF 侵入检测事件标志（并复位侵入检测器）

5 闪存控制器（FMC）

5.1 FMC简介

内嵌的闪存存储器可以用于在线编程（ICP）或在程序中编程（IAP）烧写。

在线编程（In-Circuit Programming - ICP）方式用于更新闪存存储器的全部内容，它通过 JTAG、SWD 协议或系统加载程序（Bootloader）下载用户应用程序到微控制器中。ICP 是一种快速有效的编程方法，消除了封装和管座的困扰。

与 ICP 方式对应，在程序中编程（In-Application Programming - IAP）可以使用微控制器支持的任一种通信接口（如 I/O 端口、USB、CAN、UART、I^2C、SPI 等）下载程序或数据到存储器中。IAP 允许用户在程序运行时重新烧写闪存存储器中的内容。然而，IAP 要求至少有一部分程序已经使用 ICP 烧到闪存存储器中。

闪存接口是在 AHB 协议上实现了对指令和数据的访问，它通过对存储器的预取缓存，加快了存储器的访问；闪存接口还实现了在所有工作电压下对闪存编程和擦除所需的逻辑电路，这里还包括访问和写入保护以及选择字节的控制。

5.2 主要特点

- 多达1024K字节闪存
- 总共使用了两片闪存；前512KB容量在第1片闪存（bank1）中，其余的容量在第2片闪存（bank2）中
- 透过SPIM接口提供外部存储器（bank3）闪存接口的特性
- 带预取缓冲器的读接口（4x64位）
- 选择字节加载
- 闪存编程/擦除操作
- 读出/写入保护
- 低功耗模式

5.2.1 闪存模块组织

存储器组织成高达 1024K 的主存储器块和一个信息块：

表 5-1 闪存模块组织（1024K）

块		名称	地址范围	长度（字节）
主存储器	块 1 (Bank1) 512KB	页 0	0x0800 0000 – 0x0800 07FF	2K
		页 1	0x0800 0800 – 0x0800 0FFF	2K
		页 2	0x0800 1000 – 0x0800 17FF	2K
		页 3	0x0800 1800 – 0x0800 1FFF	2K
		页 4	0x0800 2000 – 0x0800 27FF	2K
		. .	. .	. .
		页 255	0x0807 F800 – 0x0807 FFFF	2K
	块 2 (Bank2) 512KB	页 256	0x0808 0000 – 0x0808 07FF	2K
		页 257	0x0808 0800 – 0x0808 0FFF	2K
		页 258	0x0808 1000 – 0x0808 17FF	2K
		页 259	0x0808 1800 – 0x0808 1FFF	2K
		页 260	0x0808 2000 – 0x0808 27FF	2K
		. .	. .	. .
		页 511	0x080F F800 – 0x080F FFFF	2K
外部存储器	块 3 (Bank3)		0x0840 0000 – 0x093F FFFF	16M

信息块	启动程序代码	0x1FFF B000 – 0x1FFF EFFF	16K
	用户选择字节	0x1FFF F800 – 0x1FFF F82F	48
闪存存储器 接口寄存器	FLASH_ACR	0x4002 2000 – 0x4002 2003	4
	FLASH_FCKEY	0x4002 2004 – 0x4002 2007	4
	FLASH_OPTKEYR	0x4002 2008 – 0x4002 200B	4
	FLASH_STS	0x4002 200C – 0x4002 200F	4
	FLASH_CTRL	0x4002 2010 – 0x4002 2013	4
	FLASH_ADDR	0x4002 2014 – 0x4002 2017	4
	保留	0x4002 2018 – 0x4002 201B	4
	FLASH_UOB	0x4002 201C – 0x4002 201F	4
	FLASH_WRPRT	0x4002 2020 – 0x4002 2023	4
	保留	0x4002 2024 - 0x4002 2043	32
	FLASH_FCKEY2	0x4002 2044 - 0x4002 2047	4
	保留	0x4002 2048 - 0x4002 204B	4
	FLASH_STS2	0x4002 204C - 0x4002 204F	4
	FLASH_CTRL2	0x4002 2050 - 0x4002 2053	4
	FLASH_ADDR2	0x4002 2054 - 0x4002 2057	4
	保留	0x4002 2058 - 0x4002 2083	44
	FLASH_FCKEY3	0x4002 2084 - 0x4002 2087	4
	FLASH_SELECT	0x4002 2088 - 0x4002 208B	4
	FLASH_STS3	0x4002 208C - 0x4002 208F	4
	FLASH_CTRL3	0x4002 2090 - 0x4002 2093	4
	FLASH_ADDR3	0x4002 2094 - 0x4002 2097	4
	FLASH_DA	0x4002 2098 – 0x4002 209B	4
	保留	0x4002 209C - 0x4002 20CB	48
	SLIB_CDR0	0x4002 20CC - 0x4002 20CF	4
	SLIB_CDR1	0x4002 20D0 - 0x4002 20D3	4
	SLIB_PSW	0x4002 20D4 - 0x4002 20D7	4
	SLIB_PSW_STS	0x4002 20D8 - 0x4002 20DB	4
	SLIB_SET_PSW	0x4002 20DC - 0x4002 20DF	4
	SLIB_SET_RANGE	0x4002 20E0 - 0x4002 20E3	4
	保留	0x4002 20E4 - 0x4002 20EF	12
	SLIB_KEYR	0x4002 20F0 - 0x4002 20F3	4
	CRC_DR	0x4002 20F4 - 0x4002 20F7	4
	CRC_OUTR	0x4002 20F8 - 0x4002 2FB	4

表 5-2 闪存模块组织（512K）

块		名称	地址范围	长度（字节）
主存储器	块 1 (Bank1) 512KB	页 0	0x0800 0000 – 0x0800 07FF	2K
		页 1	0x0800 0800 – 0x0800 0FFF	2K
		页 2	0x0800 1000 – 0x0800 17FF	2K
		页 3	0x0800 1800 – 0x0800 1FFF	2K
		页 4	0x0800 2000 – 0x0800 27FF	2K
		. .	. .	. .
		页 255	0x0807 F800 – 0x0807 FFFF	2K
外部存储器	块 3 (Bank3)		0x0840 0000 – 0x093F FFFF	16M

信息块	启动程序代码	0x1FFF B000 – 0x1FFF EFFF	16K
	用户选择字节	0x1FFF F800 – 0x1FFF F82F	48
闪存存储器接口寄存器	FLASH_ACR	0x4002 2000 – 0x4002 2003	4
	FLASH_FCKEY	0x4002 2004 – 0x4002 2007	4
	FLASH_OPTKEYR	0x4002 2008 – 0x4002 200B	4
	FLASH_STS	0x4002 200C – 0x4002 200F	4
	FLASH_CTRL	0x4002 2010 – 0x4002 2013	4
	FLASH_ADDR	0x4002 2014 – 0x4002 2017	4
	保留	0x4002 2018 – 0x4002 201B	4
	FLASH_UOB	0x4002 201C – 0x4002 201F	4
	FLASH_WRPRT	0x4002 2020 – 0x4002 2023	4
	保留	0x4002 2024 - 0x4002 2083	96
	FLASH_FCKEY3	0x4002 2084 - 0x4002 2087	4
	FLASH_SELECT	0x4002 2088 - 0x4002 208B	4
	FLASH_STS3	0x4002 208C - 0x4002 208F	4
	FLASH_CTRL3	0x4002 2090 - 0x4002 2093	4
	FLASH_ADDR3	0x4002 2094 - 0x4002 2097	4
	FLASH_DA	0x4002 2098 – 0x4002 209B	4
	保留	0x4002 209C - 0x4002 20CB	48
	SLIB_CDR0	0x4002 20CC - 0x4002 20CF	4
	SLIB_CDR1	0x4002 20D0 - 0x4002 20D3	4
	SLIB_PSW	0x4002 20D4 - 0x4002 20D7	4
	SLIB_PSW_STS	0x4002 20D8 - 0x4002 20DB	4
	SLIB_SET_PSW	0x4002 20DC - 0x4002 20DF	4
	SLIB_SET_RANGE	0x4002 20E0 - 0x4002 20E3	4
	保留	0x4002 20E4 - 0x4002 20EF	12
	SLIB_KEYR	0x4002 20F0 - 0x4002 20F3	4
	CRC_DR	0x4002 20F4 - 0x4002 20F7	4
	CRC_OUTR	0x4002 20F8 - 0x4002 2FB	4

表 5-3 闪存模块组织（256K）

块		名称	地址范围	长度（字节）
主存储器	块 1 (Bank1) 256KB	页 0	0x0800 0000 – 0x0800 07FF	2K
		页 1	0x0800 0800 – 0x0800 0FFF	2K
		页 2	0x0800 1000 – 0x0800 17FF	2K
		页 3	0x0800 1800 – 0x0800 1FFF	2K
		页 4	0x0800 2000 – 0x0800 27FF	2K
		. .	. .	. .
		页 127	0x0803 F800 – 0x0803 FFFF	2K
外部存储器	块 3 (Bank3)		0x0840 0000 – 0x093F FFFF	16M
信息块		启动程序代码	0x1FFF B000 – 0x1FFF EFFF	16K
		用户选择字节	0x1FFF F800 – 0x1FFF F82F	48
闪存存储器接口寄存器		FLASH_ACR	0x4002 2000 – 0x4002 2003	4
		FLASH_FCKEY	0x4002 2004 – 0x4002 2007	4
		FLASH_OPTKEYR	0x4002 2008 – 0x4002 200B	4
		FLASH_STS	0x4002 200C – 0x4002 200F	4
		FLASH_CTRL	0x4002 2010 – 0x4002 2013	4

FLASH_ADDR	0x4002 2014 – 0x4002 2017	4
保留	0x4002 2018 – 0x4002 201B	4
FLASH_UOB	0x4002 201C – 0x4002 201F	4
FLASH_WRPRT	0x4002 2020 – 0x4002 2023	4
保留	0x4002 2024 - 0x4002 2083	96
FLASH_FCKEY3	0x4002 2084 - 0x4002 2087	4
FLASH_SELECT	0x4002 2088 - 0x4002 208B	4
FLASH_STS3	0x4002 208C - 0x4002 208F	4
FLASH_CTRL3	0x4002 2090 - 0x4002 2093	4
FLASH_ADDR3	0x4002 2094 - 0x4002 2097	4
FLASH_DA	0x4002 2098 – 0x4002 209B	4
保留	0x4002 209C - 0x4002 20CB	48
SLIB_CDR0	0x4002 20CC - 0x4002 20CF	4
SLIB_CDR1	0x4002 20D0 - 0x4002 20D3	4
SLIB_PSW	0x4002 20D4 - 0x4002 20D7	4
SLIB_PSW_STS	0x4002 20D8 - 0x4002 20DB	4
SLIB_SET_PSW	0x4002 20DC - 0x4002 20DF	4
SLIB_SET_RANGE	0x4002 20E0 - 0x4002 20E3	4
保留	0x4002 20E4 - 0x4002 20EF	12
SLIB_KEYR	0x4002 20F0 - 0x4002 20F3	4
CRC_DR	0x4002 20F4 - 0x4002 20F7	4
CRC_OUTR	0x4002 20F8 - 0x4002 2FB	4

闪存存储器被组织成 32 位宽的存储器单元，可以存放代码和数据常数。每一个 AT32F403A/407 微控制器的闪存模块都有一个特定的启始地址。

信息块分为两个部分：

- 系统存储器是用于存放在系统存储器自举模式下的启动程序，这个区域只保留给Artery使用，启动程序使用USART1，USART2或者USB（DFU）串行接口实现对闪存存储器的编程；Artery在生产线上对这个区域编程并锁定以防止用户擦写。
- 选择字节

  对主存储器和信息块的写入由内嵌的闪存编程/擦除控制器（FPEC）管理；编程与擦除的高电压由内部产生。

闪存存储器有两种保护方式防止非法的访问（读、写、擦除）：

- 页写入保护
- 读出保护（详情请参考5.3.3节）

在执行闪存写操作时，任何对闪存的读操作都会锁住总线，在写操作完成后读操作才能正确地进行；即在进行写或擦除操作时，不能进行代码或数据的读取操作。

进行闪存编程操作时（写或擦除），必须打开内部的 RC 振荡器（HSI）。闪存存储器可以用 ICP 或 IAP 方式编程。

注意：在低功耗模式下，所有闪存存储器的操作都被中止。

5.2.2 外部闪存模块组织

闪存存储器提供外部闪存存取模块，可以透过 SPIM 传输接口控制外部 SPI 闪存 (bank3)，读写最大容量 16M 字节，操作方式和第一片闪存(bank1)及第二片闪存(bank2)相同，另外增加密文保护功能可透过选择字节决定数据是否加密，并由 FLASH_DA 寄存器控制加密范围。

AHB 时钟（HCLK）是 SPIM 的参考时钟。透过 SPIM 传输接口向外部 SPI 闪存提供 HCLK/2 的时钟。

SPIM = 外部 SPI Flash memory 扩展(程序执行/数据储存/程序与数据可加密)。SPIM 和 USB 接口复用管脚，两种功能无法同时使用。

注意：外部闪存模块组织只支持字或半字的操作。

图 5-1 编程过程

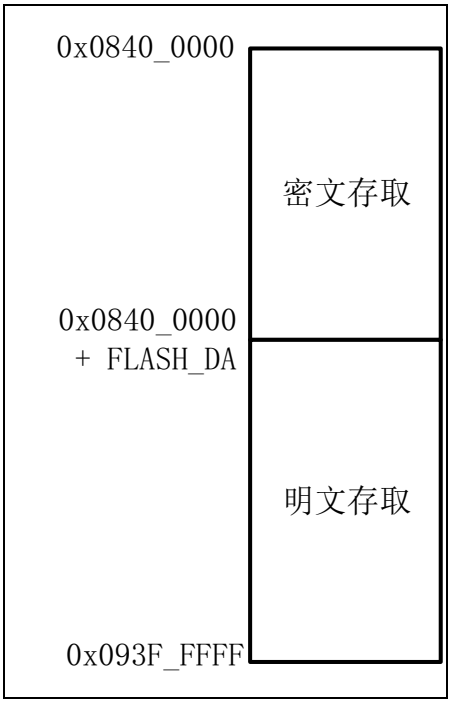
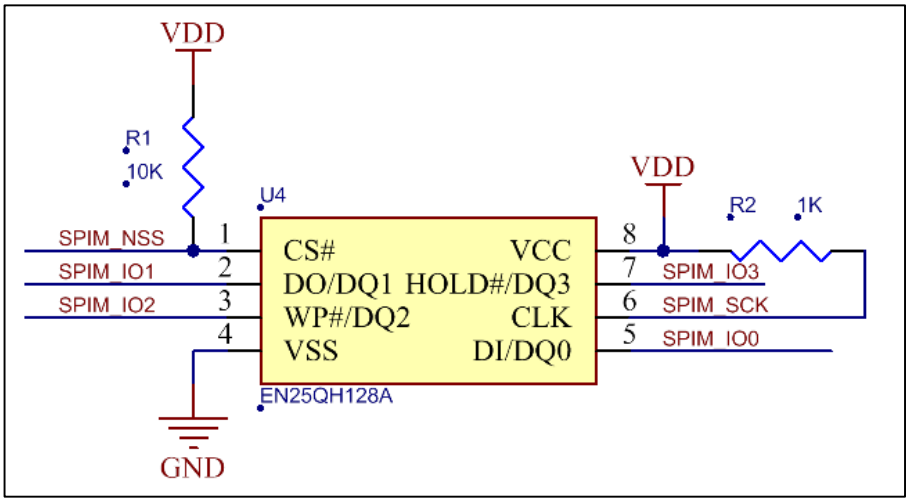


图 5-2 外部存储器参考线路



5.3 功能描述

5.3.1 读操作

内置闪存模块可以在通用地址空间直接寻址，任何 8/16/32 位数据的读操作都能访问闪存模块的内容并得到相应的数据。

读接口在闪存端包含一个读控制器，还包含一个 AHB 接口与 CPU 衔接。这个接口的主要工作是产生读闪存的控制信号并预取 CPU 要求的指令块，预取指令块仅用于在 I-Code 总线上的取指操作，数据常量是通过 D-Code 总线访问的。这两条总线的访问目标是相同的闪存模块，访问 D-Code 将比预取指令优先级高。

5.3.1.1 取指令

Cortex™-M4F 在 I-Code 总线上取指令，在 D-Code 总线上取数据。预取指令块可以有效地提高对 I-Code

总线访问的效率。

预取缓冲器包含两个数据块，每个数据块有 8 个字节；预取指令（数据）块直接映像到闪存中，因为数据块的大小与闪存的宽度相同，所以读取预取指令块可以在一个读周期完成。

设置预取缓冲器可以使 CPU 更快地执行，CPU 读取一个字的同时下一个字已经在预取缓冲器中等候，即当代码跳转的边界为 8 字节的倍数时，闪存的加速比例为 2。

预取控制器根据预取缓冲器中可用的空间决定是否访问闪存，预取缓冲器中有至少一块的空余空间时，预取控制器则启动一次读操作。

清除闪存访问控制寄存器中的一个控制位能够关闭预取缓冲器。芯片复位后预取缓冲器处于开启状态。

注意： *当 AHB 时钟的预分频系数不为’1’时，必须打开预取缓冲器（FLASH_ACR[4]=1）。*

如果在系统中没有高频率的时钟，即 HCLK 频率较低时，闪存的访问只需半个 HCLK 周期（半周期的闪存访问只能在时钟频率低于 8MHz 时进行，使用 HSI 或 HSE 并且关闭 PLL 时可得到这样的频率）；在闪存访问控制寄存器中有一个控制位可以选择这种工作方式。

注意： *当使用了预取缓冲器和 AHB 时钟的预分频系数不为’1’时，不能使用半周期访问方式。*

访问时间调节器为了维持读闪存的控制信号，预取控制器的时钟周期与闪存访问时间的比例由闪存访问控制器控制；这个值给出了能够正确地读取数据时，闪存控制信号所需的时钟周期数目；芯片复位后，该值为’0’，闪存访问仅为一个时钟周期。

5.3.1.2 D-Code接口

D-Code 接口包含 CPU 端简单的 AHB 接口和对闪存访问控制器的仲裁器提出访问请求的逻辑电路。D-Code 的访问优先于预取指令的访问。这个接口使用预取缓冲器的访问时间调节器模块。

5.3.1.3 闪存访问控制器

这个模块就是在 I-Code 上的指令预取请求和 D-Code 接口上读请求的仲裁器。D-Code 接口的请求优先于 I-Code 的请求。

5.3.2 闪存编程和擦除控制器（FPEC）

FPEC 模块处理闪存的编程和擦除操作，它包括 17 个 32 位的寄存器，只要 CPU 不访问闪存，闪存操作不会延缓 CPU 的执行。

- FPEC键寄存器（FLASH_FCKEY）
- 选择字节键寄存器（FLASH_OPTKEYR）
- 闪存状态寄存器（FLASH_STS）
- 闪存控制寄存器（FLASH_CTRL）
- 闪存地址寄存器（FLASH_ADDR）
- 选择字节寄存器（FLASH_UOB）
- 写保护寄存器（FLASH_WRPRT）
- FPEC键寄存器2（FLASH_FCKEY2）
- 闪存状态寄存器2（FLASH_STS2）
- 闪存控制寄存器2（FLASH_CTRL2）
- 闪存地址寄存器2（FLASH_ADDR2）
- FPEC键寄存器3（FLASH_FCKEY3）
- 闪存选择寄存器（FLASH_SELECT）
- 闪存状态寄存器3（FLASH_STS3）
- 闪存控制寄存器3（FLASH_CTRL3）
- 闪存地址寄存器3（FLASH_ADDR3）
- 闪存解密地址寄存器（FLASH_DA）

5.3.2.1 键值

共有三个键值：

- RDPRTEN键 = 0x00A5
- KEY1 = 0x45670123
- KEY2 = 0xCDEF89AB

5.3.2.2 解除闪存锁

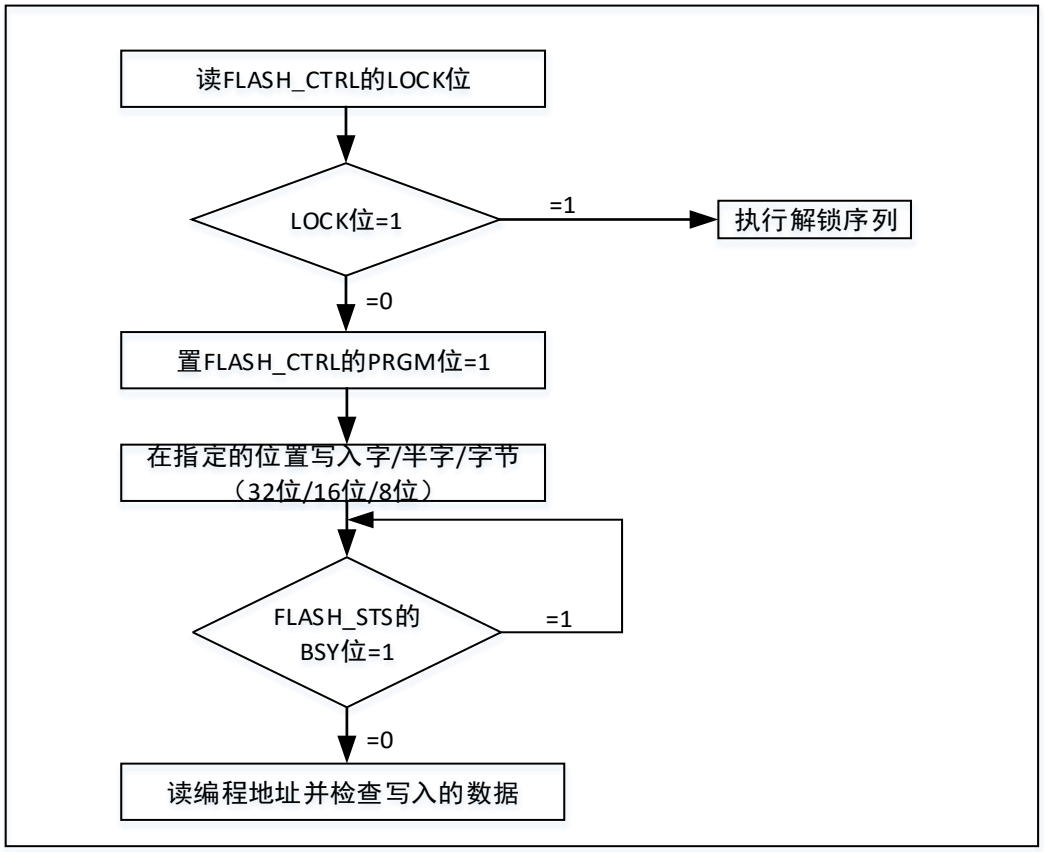
复位后，FPEC 模块是被保护的，不能写入 FLASH_CTRLx 寄存器；通过写入特定的序列到 FLASH_FCKEYx 寄存器可以打开 FPEC 模块，这个特定的序列是在 FLASH_FCKEYx 写入两个键值（KEY1 和 KEY2，见 5.3.2.1 节）；错误的操作序列都会在下次复位前锁死 FPEC 模块和 FLASH_CTRLx 寄存器。

写入错误的键序列还会产生总线错误；总线错误发生在第一次写入的不是 KEY1，或第一次写入的是 KEY1 但第二次写入的不是 KEY2 时；FPEC 模块和 FLASH_CTRLx 寄存器可以由程序设置 FLASH_CTRLx 寄存器中的 LOCK 位锁住，这时可以通过在 FLASH_FCKEYx 中写入正确的键值对 FPEC 解锁。

5.3.2.3 主闪存编程

对主闪存编程每次可以写入 32 位、16 位或 8 位。当 FLASH_CTRLx 寄存器的 PRGM 位为’1’时，在一个闪存地址写入一个字/半字/字节将启动一次编程。在编程过程中（BSY 位为’1’），任何读写闪存的操作都会使 CPU 暂停，直到此次闪存编程结束。

图 5-3 编程过程



标准编程

这种模式下CPU 以标准的写字、半字或字节的方式烧写闪存，FLASH_CTRLx 寄存器的PRGM 位必须置’1’。FPEC先读出指定地址的内容并检查它是否被擦除，如未被擦除则不执行编程并在 FLASH_STSx 寄存器的PRGMFLR 位提出警告（唯一的例外是当要烧写的数值是0x0000_0000 (0x0000或0x00) 时，0x0000_0000(0x0000或0x00) 可被正确烧入且PRGMFLR位不被置位）；如果指

定的地址在FLASH_WRPRT 中设定为写保护，则不执行编程并在FLASH_STSx寄存器的WRPRTFLR位置’1’提出警告。FLASH_STSx 寄存器的PRCDN 为’1’时表示编程结束。

标准的闪存编程顺序如下：

- 检查FLASH_STSx寄存器的BSY位，以确认没有其他正在进行的编程操作；
- 设置FLASH_CTRLx寄存器的PRGM位为’1’；
- 在指定的地址写入要编程的字/半字/字节；
- 等待BSY位变为’0’；
- 读出写入的地址并验证数据。

注意：当 FLASH_STSx 寄存器的 BSY 位为’1’时，不能对任何寄存器执行写操作。

5.3.2.4 闪存擦除

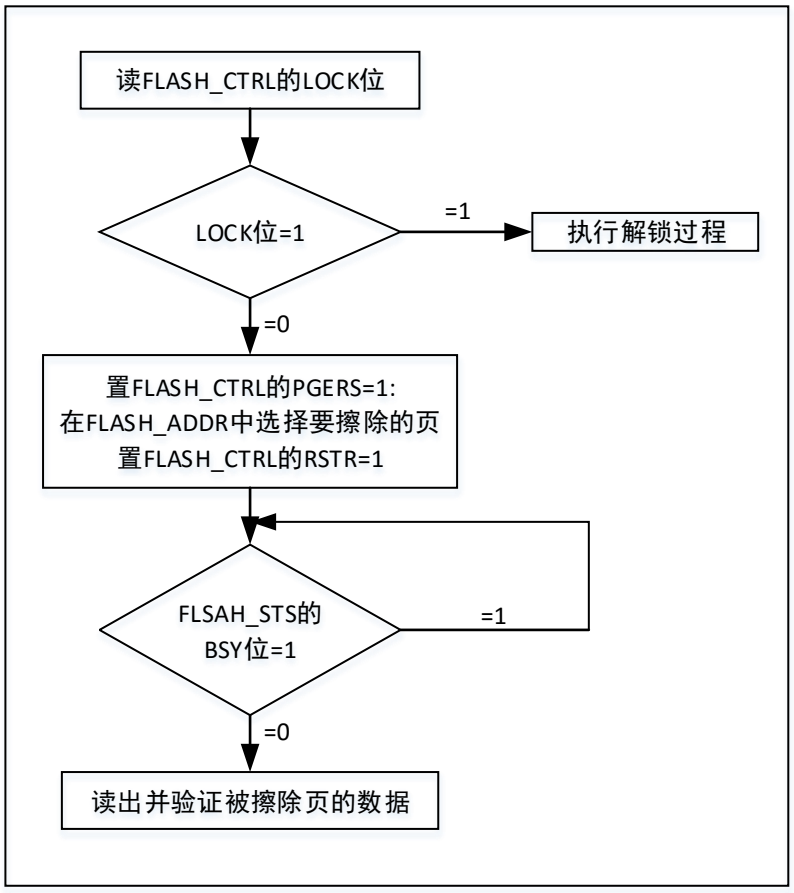
闪存可以按页擦除，也可以整片擦除。

页擦除

闪存的任何一页都可以通过 FPEC 的页擦除功能擦除；擦除一页应遵守下述过程：

- 检查FLASH_STSx寄存器的BSY位，以确认没有其他正在进行的闪存操作；
- 设置FLASH_CTRLx寄存器的PGERS位为’1’；
- 用FLASH_ADDRx寄存器选择要擦除的页；
- 设置FLASH_CTRLx寄存器的RSTR位为’1’；
- 等待BSY位变为’0’；
- 读出被擦除的页并做验证。

图 5-4 闪存页擦除过程

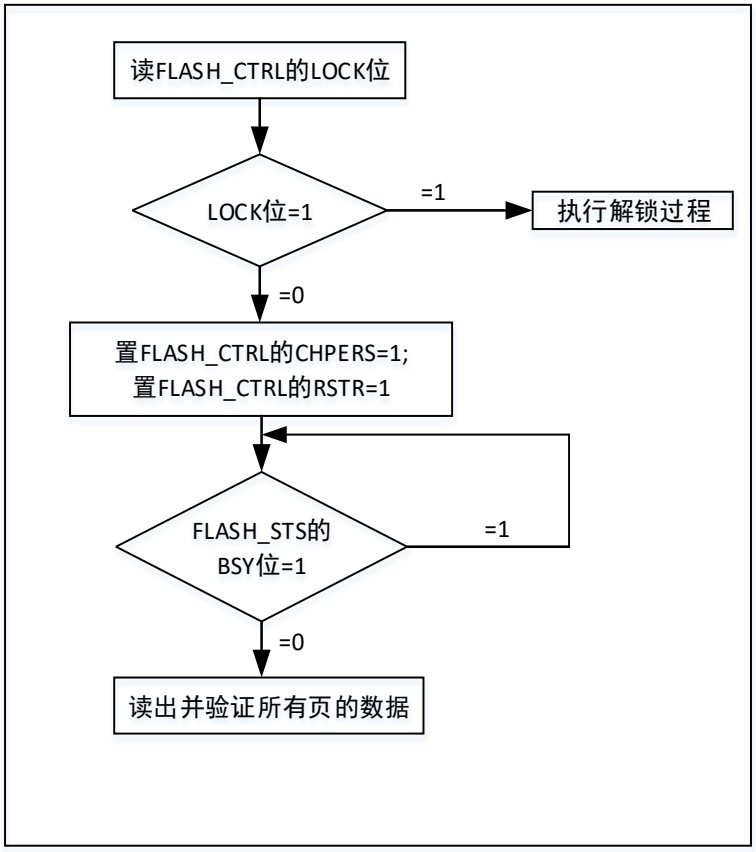


整片擦除

可以用整片擦除功能擦除所有用户区的闪存，信息块不受此操作影响。建议使用下述过程：

- 检查FLASH_STSx寄存器的BSY位，以确认没有其他正在进行的闪存操作；
- 设置FLASH_CTRLx寄存器的CHPERS位为'1'；
- 设置FLASH_CTRLx寄存器的RSTR位为'1'；
- 等待BSY位变为'0'；
- 读出所有页并做验证。

图 5-5 闪存整片擦除过程



5.3.2.5 选择字节编程

对选择字节的编程与普通的用户地址不同。选择字节的数目只有 24 个字节（其中 4 个字节作为写保护，1 个字节作为读保护，1 个字节为配置选项，8 个字节存储用户数据）。对 FPEC 解锁后，必须分别写入 KEY1 和 KEY2（见 5.3.2.1 节）到 FLASH_OPTKEYR 寄存器，FLASH_CTRL 寄存器的 UOBWE 位会被设置为'1'，此时可以对选择字节进行编程：设置 FLASH_CTRL 寄存器的 UOBPRGM 位为'1'后写入半字到指定的地址。

FPEC 先读出指定地址的选择字节内容并检查它是否已经被擦除，如未被擦除则不执行编程并在 FLASH_STS 寄存器的 WRPRTFLR 位提出警告。FLASH_STS 寄存器的 PRCDN 为'1'时表示编程结束。

FPEC 使用半字中的低字节并自动地计算出高字节（高字节为低字节的反码），并开始编程操作，这将保证选择字节和它的反码始终是正确的。

烧写编程的顺序如下：

- 检查FLASH_STS寄存器的BSY位，以确认没有其他正在进行的编程操作；
- 解锁FLASH_CTRL寄存器的UOBWE位；
- 设置FLASH_CTRL寄存器的UOBPRGM位为'1'；
- 写入要编程的半字到指定的地址；

- 等待BSY位变为’ 0’；
- 读出写入的地址并验证数据。

当读闪存保护选项从“保护”变为“未保护”时，在重新设置读保护选项前会自动执行一个整片擦除用户闪存的操作。如果用户要改变读保护之外的选项，则不会出现整片擦除操作。读保护选项上的这一擦除操作保护了闪存中的内容不被非法读出。

擦除过程

选择字节的擦除操作顺序（OPTERASE）如下：

- 检查FLASH_STS寄存器的BSY位，以确认没有其他正在进行的闪存操作；
- 解锁FLASH_CTRL寄存器的UOBWE位；
- 设置FLASH_CTRL寄存器的UOBERS位为’1’；
- 设置FLASH_CTRL寄存器的RSTR位为’1’；
- 等待BSY位变为’0’；
- 读出被擦除的选择字节并做验证。

5.3.3 保护

闪存中的用户代码区可以防止非法的读出；同样可以对闪存区的页加以保护，防止在程序跑飞的情况下不被意外地改变。在 256K 及以上的闪存容量中，写保护的基本单位为 2 页。

5.3.3.1 写保护

如果试图在一个受保护的页面进行编程或擦除操作，在闪存状态寄存器（FLASH_STS）中会返回一个保护错误标志。

解除保护下属步骤用于解除写保护：

- 使用闪存控制寄存器（FLASH_CTRL）的UOBERS位擦除整个选择字节区域；
- 写入正确的RDP代码0xA5，允许读访问；
- 进行系统复位，重装载选择字节（包含新的WRP [3：0]字节）；写保护被解除。

5.3.3.2 读保护

这项保护是通过设置 RDP 选择字节启动的，并通过系统复位加载 RDP 选择字节。当保护字节被写入相应的值以后，实施下述保护：

- 只允许从用户代码中对主闪存存储器的读操作（以非调试方式从主闪存存储器启动）。
- 第0~1页（256K及以上闪存容量）被自动加上了写保护，其它部分的存储器可以通过在主闪存存储器中执行的代码进行编程（实现IAP或数据存储等功能），但不允许在调试模式下或在从内部SRAM启动后执行写或擦除操作（整片擦除除外）。

所有通过 JTAG/SWD 向内置 SRAM 装载代码并执行代码的功能依然有效，亦可以通过 JTAG/SWD 从内置 SRAM 启动，这个功能可以用来解除读保护。当读保护的选择字节转变为存储器未保护的数值时，将会执行主存的整片擦除过程。

当 RDP 选择字节和它的反码包含下列数值对时，闪存被置于保护状态：

表 5-4 闪存存储器保护状态

RDP 字节的值	RDP 反码的值	读保护状态
0xFF	0xFF	保护
RDPRTEN	RDP 字节的反码	未保护
任意值	非 RDP 字节的反码	保护

注意：擦除选择字节块将不会导致自动的整片擦除操作，因为擦除的结果 0xFF 相当于保护状态。

解除读保护的过程是：

- 擦除整个选择字节区域，读保护码（RDP）将变为0xFF，此时读保护仍然有效；

- 写入正确的RDP代码0xA5以解除存储器的保护，该操作将首先导致对所有用户闪存的整片擦除操作；
- 进行复位以重新加载选择字节（和新的RDP代码），此时读保护被解除。

注意：可以使用系统启动程序解除读保护（此时只需执行系统复位即可重新加载选择字节）。

5.3.3.3 选择字节块写保护

默认状态下，选择字节块始终是可以读且被写保护。要想对选择字节块进行写操作（编程/擦除）首先要在 OPTKEYR 中写入正确的键序列（与上锁时一样），随后允许对选择字节块的写操作，FLASH_CTRL 寄存器的 UOBWE 位标示允许写，清除这位将禁止写操作。

5.3.4 选择字节说明

选择字节共有 24 个字节，由用户根据应用的需要配置；例如：可以选择使用硬件模式的看门狗或软件的看门狗。

在选择字节中每个 32 位的字被划分为下述格式：

表 5-5 选择字节格式

位 31~24	位 23~16	位 15~8	位 7~0
选择字节 1 的反码	选择字节 1	选择字节 0 的反码	选择字节 0

选择字节块中选择字节的组织结构如下：

表 5-6 信息块的组织结构

地址	**[31：24]**	**[23：16]**	**[15：8]**	**[7：0]**
0x1FFF_F800	uUSER	USER	nRDP	RDP
0x1FFF_F804	nData1	Data1	nData0	Data0
0x1FFF_F808	nWRP1	WRP1	nWRP0	WRP0
0x1FFF_F80C	nWRP3	WRP3	nWRP2	WRP2
0x1FFF_F810	Reserved	Reserved	nEOPB0	EOPB0
0x1FFF_F814	nData3	Data3	nData2	Data2
0x1FFF_F818	nData5	Data5	nData4	Data4
0x1FFF_F81C	nData7	Data7	nData6	Data6
0x1FFF_F820	nBANK3KEY1	BANK3KEY1	nBANK3KEY0	BANK3KEY0
0x1FFF_F824	nBANK3KEY3	BANK3KEY3	nBANK3KEY2	BANK3KEY2
0x1FFF_F828	nBANK3KEY5	BANK3KEY5	nBANK3KEY4	BANK3KEY4
0x1FFF_F82C	nBANK3KEY7	BANK3KEY7	nBANK3KEY6	BANK3KEY6

表 5-7 用户选择字节说明

RDP：读出保护选择字节 读出保护功能帮助用户保护存在闪存中的代码。该功能由设置信息块中的一个选择字节启用。写入正确的数值（RDPRTEN 键=0x00A5）到这个选择字节后，闪存被开放允许读出访问。 （读保护启动/解除结果存放在寄存器 FLASH_UOB[1]）	
USR：用户选择字节（存放在寄存器 FLASH_UOB[9:2]） 这个字节用于配置下列功能： – 选择主存储块启动：块 1 或块 2 – 选择看门狗事件：硬件或软件 – 进入停机（STOP）模式时的复位事件 – 进入待机模式时的复位事件	
位 23：20	0xF：不用
位19	BTOPT 0：当配置从主存储块启动时，若块 2 无启动程序，从块 1 启动。否则，从块 2 启动 1：当配置从主存储块启动（默认值）时，从块 1 启动
位18	nRST_STDBY 0：当进入待机模式时产生复位 1：进入待机模式时不产生复位

位17	nRST_STOP 0：当进入停机（STOP）模式时产生复位 1：进入停机（STOP）模式时不产生复位
位16	WDG_SW 0：硬件看门狗 1：软件看门狗
WRPx：闪存写保护选择字节（存放在寄存器 FLASH_WRPRT） -- 对于 256K 及以上闪存容量，选择字节 WRPx 中的每一个比特位用于保护主存储器中 2 个存储页（2K 字节/页），但是 WRP3 的位 7 用于保护第 62 及之后的页： 0：实施写保护 1：不实施写保护　四个用户选择字节用于保护总共 256K 字节（或 512K, 1024K）的主存储器以及外部存储。 WRP0：第 0~15 页的写保护（存放在寄存器 FLASH_WRPRT[7:0]） WRP1：第 16~31 页的写保护（存放在寄存器 FLASH_WRPRT[15:8]） WRP2：第 32~47 页的写保护（存放在寄存器 FLASH_WRPRT[23:16]） WRP3：位 0~6 提供第 48~61 页的写保护（存放在寄存器 FLASH_WRPRT[31:24]）；位 7 提供第 62 及之后页的写保护，以及块 3（Bank3）的写保护	
Datax：用户数据 这个地址可以使用选择字节的编程方式编程。 Data7：用户数据 7 Data6：用户数据 6 Data5：用户数据 5 Data4：用户数据 4 Data3：用户数据 3 Data2：用户数据 2 Data1：用户数据 1（存放在寄存器 FLASH_UOB[25:18]） Data0：用户数据 0（存放在寄存器 FLASH_UOB[17:10]）	
EOPB0: 扩充的选择字节	
位 0	0：片上 SRAM 224K 字节 1：片上 SRAM 96K 字节 注意：位 0 由 1 改写成 0 只能在安全库区未启动的情况下实现
BANK3KEYx: 第三片闪存(Bank3)密文存取区加密键值 **不加密的设定条件包括**： -- BANK3KEYx 以及 nBANK3KEYx 均为 0xFF（即默认擦除状态） -- BANK3KEYx 写入 0x00 -- BANK3KEYx 写入 0xFF 即{nBANK3KEYx, BANK3KEYx }均设为 0xFFFF, 0xFF00, 0x00FF	

每次系统复位后，选择字节装载器读出信息块的数据并保存在寄存器中；每个选择位都在信息块中有它的反码位，在装载选择位时反码位用于验证选择位是否正确，如果有任何的差别，将产生一个选择字节错误标志（UOBFLR）。当发生选择字节错误时，对应的选择字节被强置为 0xFF。当选择字节和它的反码均为 0xFF 时（擦除后的状态）——关闭上述验证功能。

所有的选择位（不包括它们的反码位）用于配置该微控制器，CPU 可以读选择寄存器，详见第 5.4 章 FMC 寄存器。

5.3.5特殊功能

5.3.5.1 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：

只可在主存中设置安全库区

安全库区代码必须以页为单位进行烧录，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页(页 0)内，请勿将闪存的第一页设定为安全库区；

要被安全库区保护的程序代码，不可放置在闪存的第一页内；

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 FLASH_STS 寄存器的 WRPRTFLR 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写保护。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 SLIB_KEYR 寄存器写入 0xA35F6D24 值，通过查看 SLIB_PSW_STS 位 SLIB_UNLOCK 确认解锁成功，随后允许对安全库区设定寄存器写入设定值。
以页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过程如下：

- 检查FLASH_STS寄存器的BSY位，以确认没有其他正在进行的编程操作；
- 对SLIB_KEYR寄存器写入0xA35F6D24，以进行安全库区解锁；
- 检查SLIB_PSW_STS寄存器的SLIB_UNLOCK位，以确认解锁成功；
- 在SLIB_SET_RANGE寄存器设定要保护的区域，包含指令区与数据区的地址；
- 等待BSY位变为’0’；
- 在SLIB_SET_PSW寄存器设定安全区域密码；
- 等待BSY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SLIB_CDR0/CDR1寄存器用于判断安全库区设定结果。

解除安全库区的过程是：

- 在SLIB_PSW寄存器写入先前设置的安全区域密码；
- 等待BSY位变为’0’；
- 进行系统复位，重装载安全库区设定字；
- 读出SLIB_CDR0寄存器用于判断安全库区设定结果。

注意：解除安全库区将会执行主存的整片擦除，以及安全库设定块擦除。

5.4 FMC寄存器

表 5-8 闪存接口—寄存器映像和复位值

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	FLASH_ACR	保留																										PRFTBS	PRFTBE	HLFCYA	保留		
	复位值																											1	1	0			
0x04	FLASH_FCKEY	KEY[31：0]																															
	复位值	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0x08	FLASH_OPTKEYR	OPTKEYR[31：0]																															
	复位值	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0x0C	FLASH_STS	保留																										PRCDN	WRPRTFLR	保留	PRGMFLR	保留	BSY
	复位值																											0	0		0		0
0x10	FLASH_CTRL	保留																			PRCDNI	保留	ERRIE	UOBWE	保留	LOCK	RSTR	UOBERS	UOBPRG	保留	CHPERS	PGERS	PRGM
	复位值																				0		0	0		1	0	0	0		0	0	0
0x14	FLASH_ADDR	TA[31：0]																															
	复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	保留																																
0x1C	FLASH_UOB	保留						USER_D1									USER_D0							未使用				BTOPT	nSTDBY_RST	nSTP_RST	SYSCLKSEL_WD	RDPRTEN	UOBFLR
	复位值							1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0
0x20	FLASH_WRPRT	WRPRTBMP[31：0]																															
	复位值	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x44	FLASH_FCKEY2	KEY[31：0]																															
	复位值	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0x4C	FLASH_STS2	保留																										PRCDN	WRPRTFLR	保留	PRGMFLR	保留	BSY
	复位值																											0	0		0		0

0x50	FLASH_CTRL2	保留																					PRCDNIE	保留	ERRIE	保留	LOCK	RSTR	保留			CHPERS	PGERS	PRGM
	复位值																						0		0		1	0				0	0	0
0x54	FLASH_ADDR2	TA[31：0]																																
	复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x84	FLASH_FCKEY3	KEY [31：0]																																
	复位值	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0x88	FLASH_SELECT	SELECT [31：0]																																
	复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x8C	FLASH_STS3	保留																											PRCDN	WRPRTFLR	保留	PRGMFLR	保留	BSY
	复位值																												0	0		0		0
0x90	FLASH_CTRL3	保留																					PRCDNIE	保留	ERRIE	保留	LOCK	RSTR	保留			CHPERS	PGERS	PRGM
	复位值																						0		0		1	0				0	0	0
0x94	FLASH_ADDR3	TA[31：0]																																
	复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x98	FLASH_DA	FDA[31：0]																																
	复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xCC	SLIB_CDR0	**保留**																														FSLIB_EN	**保留**	
	复位值																															0	0	0
0xD0	SLIB_CDR1		SLIB_END_PAGE [9：0]									SLIB _DATA_START_PAGE [10：0]										SLIB_START_PAGE [10：0]											
	复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xD4	SLIB_PSW	FSLIB_PSW[31：0]																																
	复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xD8	SLIB_PSW_STS	保留							SLIB_CNT[8:0]										保留												SLIB_UNLOCK	SLIB_PSW_OK	SLIB_PSW_ERR
	复位值	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xDC	SLIB_SET_PSW	FSLIB_SET_PSW [31：0]																																
	复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

0xE0	SLIB_SET_RANGE	SLIB_SET_END_PAGE [9：0]										SLIB_SET_DATA_START_PAGE [10：0]											SLIB_SET_START_PAGE [10：0]										
	复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xF0	SLIB_KEYR	FSLIB_KEYR [31：0]																														
	复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xF4	CRC_DR	CRC_START	保留									CRC_PAGE_NUM[11:0]											CRC_START_PAGE[11:0]										
	复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xF8	CRC_OUTR	CRC_OUTR [31：0]																														
	复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

5.4.1 闪存访问控制寄存器（FLASH_ACR）

地址偏移：0x00

复位值：0x0000 0030

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															
res															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留										PRFTBS	PRFTBE	HLFCYA	保留		
res										r	rw	rw	res		

位 31~6	保留。必须保持为清除状态’0’
位 5	**PRFTBS**：预取缓冲区状态 该位指示预取缓冲区的状态 0：预取缓冲区关闭； 1：预取缓冲区开启。
位 4	**PRFTBE**：预取缓冲区使能 0：关闭预取缓冲区； 1：启用预取缓冲区。
位 3	**HLFCYA**：闪存半周期访问使能 0：禁止半周期访问； 1：启用半周期访问。
位 2~0	保留

5.4.2 FPEC键寄存器（FLASH_FCKEY）

专用于闪存区块 1。

地址偏移：0x04

复位值：xxxx xxxx

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[31：16]															

w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY[15：0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

注意：　所有这些位是只写的，读出时返回 0。

位 31~0	**KEY：FPEC** 键 这些位用于输入 **FPEC** 的解锁键。

5.4.3 闪存OPTKEY寄存器（FLASH_OPTKEYR）

地址偏移：0x08

复位值：xxxx xxxx

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

注意: 所有这些位是只写的，读出时返回 0

位 31~0	**OPTKEYR**：选择字节键 这些位用于输入选择字节的键以解除 **UOBWE**。

5.4.4 闪存状态寄存器（FLASH_STS）

专用于闪存区块 1。

地址偏移：0x0C

复位值：0x0000 0000

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

位 31~6	保留。必须保持为清除状态’0’
位 5	**PRCDN**：操作结束 当闪存操作（编程/擦除）完成时，硬件设置这位为’1’，写入’1’可以清除这位状态。 注：每次成功的编程或擦除都会设置 PRCDN 状态。
位 4	**WRPRTFLR**：写保护错误 试图对写保护的闪存地址编程时，硬件设置这位为’1’，写入’1’可以清除这位状态。
位 3	保留。必须保持为清除状态’0’

位 2	**PRGMFLR**：编程错误 试图对内容不是'0xFFFF'的地址编程时，硬件设置这位为'1'，写入'1'可以清除这位状态。 注：进行编程操作之前，必须先清除 FLASH_CTRL 寄存器的 RSTR 位。
位 1	保留。必须保持为清除状态'0'
位 0	**BSY**：繁忙标志 该位指示闪存操作正在进行。在闪存操作开始时，该位被设置为'1'；在操作结束或发生错误时该位被清除为'0'。

5.4.5 闪存控制寄存器（FLASH_CTRL）

专用于闪存区块 1。

地址偏移：0x10

复位值：0x0000 0080

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															
res															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留			PRCDNIE	保留	ERRIE	UOBWE	保留	LOCK	RSTR	UOBERS	UOBPRGM	保留	CHPERS	PGERS	PRGM
res			rw	res	rw	rw	res	rw	rw	rw	rw	res	rw	rw	rw

位 31~13	保留。必须保持为清除状态'0'
位 12	**PRCDNIE**：允许操作完成中断 该位允许在 **FLASH_STS** 寄存器中的 **PRCDN** 位变为'1'时产生中断。 0：禁止产生中断； 1：允许产生中断。
位 11，8，3	保留。必须保持为清除状态'0'
位 10	**ERRIE**：允许错误状态中断 该位允许在发生 **FPEC** 错误时产生中断（**FLASH_STS** 寄存器中的 **PRGMFLR/WRPRTFLR** 置为'1'时）。 0：禁止产生中断； 1：允许产生中断。
位 9	**UOBWE**：允许写选择字节 当该位为'1'时，允许对选择字节进行编程操作。当在 **FLASH_OPTKEYR** 寄存器写入正确的键序列后，该位被置为'1'。 软件可清除此位。
位 7	**LOCK**：锁 只能写'1'。 当该位为'1'时表示 **FPEC** 和 **FLASH_CTRL** 被锁住。在检测到正确的解锁序列后，硬件清除此位为'0'。在一次不成功的解锁操作后，下次系统复位前，该位不能再被改变。
位 6	**RSTR**：开始 当该位为'1'时将触发一次擦除操作。该位只可由软件置为'1'并在 **BSY** 变为'0'时清为'0'。
位 5	**UOBERS**：擦除选择字节 擦除选择字节。
位 4	**UOBPRGM**：烧写选择字节 对选择字节编程。

位 2	**CHPERS**：全擦除 选择擦除所有用户页。
位 1	**PGERS**：页擦除 选择擦除页。
位 0	**PRGM**：编程 选择编程操作。

5.4.6 闪存地址寄存器（FLASH_ADDR）

专用于闪存区块 1。

地址偏移：0x14

复位值：0x0000 0000

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

这些位由硬件修改为当前/最后使用的地址。在页擦除操作中，软件必须修改这个寄存器以指定要擦除的页。

位 31~0	**TA**：闪存地址 当进行编程时选择要编程的地址，当进行页擦除时选择要擦除的页。 注意：当 **FLASH_STS** 中的 **BSY** 位为’1’时，不能写这个寄存器

5.4.7 选择字节寄存器（FLASH_UOB）

地址偏移：0x1C

复位值：0x03FF FFFC

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留						USER_D1[7:0]								USER_D0[7:6]	
res						r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
USER_D0[5:0]						未使用				BTOPT	nSTDBY_RST	nSTP_RST	SYSCLKSEL_WDG	RDPRTEN	UOBFLR
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

位 31~26	保留。必须保持为清除状态’0’。
位 25~18	**USER_D1**：用户数据 1
位 17~10	**USER_D0**：用户数据 0
位 9~2	**USR**：用户选择字节 这里包含 OBL 加载的用户选择字节 位[9：6]：未用 位 5：**BTOPT** 位 4：**nSTDBY_RST** 位 3：**nSTP_RST** 位 2：**SYSCLKSEL_WDG**

位 1	**RDPRTEN**：读保护 当设置为 1，表示闪存存储器的读保护有效。 注意：该位为只读。
位 0	**UOBFLR**：选择字节错误 当该位为'1'时表示选择字节和它的反码不匹配。在寄存器 FLASH_UOB 或 FLASH_WRPRT 中，对应错误的字节和它的反码读出为 0xFF 注意：该位为只读。

5.4.8 写保护寄存器（FLASH_WRPRT）

地址偏移：0x20

复位值：0xFFFF FFFF

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

位 31~0	WRPRTBMP：写保护 该寄存器包含由 OBL 加载的写保护选择字节。 0：写保护生效 1：写保护失效 注意：这些位为只读。

5.4.9 FPEC键寄存器2（FLASH_FCKEY2）

专用于闪存区块2。

地址偏移：0x44

复位值：xxxx xxxx

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

注意： 所有这些位是只写的，读出时返回 0。

位 31~0	**KEY**：FPEC 键 这些位用于输入区块 2 **FPEC** 的解锁键。

5.4.10 闪存状态寄存器2（FLASH_STS2）

专用于闪存区块2。

地址偏移：0x4C

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16

保留
res

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留										PRCDN	WRPRTFLR	保留	PRGMFLR	保留	BSY
res										rw	rw	rw	rw	rw	r

位 31~6	保留。必须保持为清除状态’0’
位 5	**PRCDN**：操作结束 当闪存操作（编程/擦除）完成时，硬件设置这位为’1’，写入’1’可以清除这位状态。 注意：每次成功的编程或擦除都会设置 **PRCDN** 状态。
位 4	**WRPRTFLR**：写保护错误 试图对写保护的闪存地址编程时，硬件设置这位为’1’，写入’1’可以清除这位状态。
位 3	保留。必须保持为清除状态’0’
位 2	**PRGMFLR**：编程错误 试图对内容不是’0xFFFF’的地址编程时，硬件设置这位为’1’，写入’1’可以清除这位状态。 注意：进行编程操作之前，必须先清除 **FLASH_CTRL2** 寄存器的 **RSTR** 位。
位 1	保留。必须保持为清除状态’0’
位 0	**BSY**：忙 该位指示闪存操作正在进行。在闪存操作开始时，该位被设置为’1’；在操作结束或发生错误时该位被清除为’0’。

5.4.11 闪存控制寄存器2（FLASH_CTRL2）

专用于闪存区块2。

地址偏移：0x50

复位值：0x0000 0080

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															
res															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留			PRCDNIE	保留	ERRIE	保留		LOCK	RSTR	保留			CHPERS	PGERS	PRGM
res			rw	res	rw	res		rw	rw	res			rw	rw	rw

位 31~13	保留。必须保持为清除状态’0’
位 12	**PRCDNIE**：允许操作完成中断 该位允许在 **FLASH_STS2** 寄存器中的 **PRCDN** 位变为’1’时产生中断。 0：禁止产生中断； 1：允许产生中断。
位 11，9，8	保留。必须保持为清除状态’0’
位 10	**ERRIE**：允许错误状态中断 该位允许在发生 FPEC 错误时产生中断（当 **FLASH_STS2** 寄存器中的 **PRGMFLR/WRPRTFLR** 置为’1’时）。 0：禁止产生中断； 1：允许产生中断。

位 7	**LOCK**：锁 只能写'1'。 当该位为'1'时表示 **FPEC** 和 **FLASH_CTRL2** 被锁住。在检测到正确的解锁序列后，硬件清除此位为'0'。 在一次不成功的解锁操作后，下次系统复位前，该位不能再被改变。
位 6	**RSTR**：开始 当该位为'1'时将触发一次擦除操作。该位只可由软件置为'1'并在 **BSY** 变为'0'时，清为'0'。
位 5~3	保留。必须保持为清除状态'0'
位 2	**CHPERS**：全擦除 选择擦除所有用户页。
位 1	**PGERS**：页擦除 选择擦除页。
位 0	**PRGM**：编程 选择编程操作。

5.4.12 闪存地址寄存器2（FLASH_ADDR2）

专用于闪存区块2。

地址偏移：0x54

复位值：0x0000 0000

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

这些位由硬件修改为当前/最后使用的地址。在页擦除操作中，软件必须修改这个寄存器以指定要擦除的页。

位 31~0	TA：闪存地址 当进行编程时选择要编程的地址，当进行页擦除时选择要擦除的页。 注意：当 **FLASH_STS2** 中的 **BSY** 位为' 1' 时，不能写这个寄存器

5.4.13 FPEC键寄存器3（FLASH_FCKEY3）

专用于闪存区块 3。

地址偏移：0x84

复位值：xxxx xxxx

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

注意：所有这些位是只写的，读出时返回 0。

位 31~0	**KEY**：FPEC 键 这些位用于输入区块 3 **FPEC** 的解锁键。

5.4.14 闪存选择寄存器（FLASH_SELECT）

专用于闪存区块 3。

地址偏移：0x88

复位值：0x0000 0000

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

位 31~0	**SELECT[31:0]** 区块 3(Bank 3)支持扩展 SPI Flash 芯片型号选择 0x0001 : GD25Q127C, GD25Q64C, GD25Q32C, GD25Q16C, GD25Q80C 0x0002: EN25F20A, EN25QH128A, W25Q128V **其他 ： 保留** 注: 支持以上 SPI Flash 型号外，也支持兼容以上型号命令集的 SPI Flash 芯片

5.4.15 闪存状态寄存器3（FLASH_STS3）

专用于闪存区块 3。

地址偏移：0x8C

复位值：0x0000 0000

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

位 31~6	保留。必须保持为清除状态’0’
位 5	**PRCDN**：操作结束 当闪存操作（编程/擦除）完成时，硬件设置这位为’1’，写入’1’可以清除这位状态。 注意：每次成功的编程或擦除都会设置 PRCDN 状态。
位 4	**WRPRTFLR**：写保护错误 试图对写保护的闪存地址编程时，硬件设置这位为’1’，写入’1’可以清除这位状态。
位 3	保留。必须保持为清除状态’0’
位 2	**PRGMFLR**：编程错误 试图对内容不是’0xFFFF’的地址编程时，硬件设置这位为’1’，写入’1’可以清除这位状态。 注：进行编程操作之前，必须先清除 **FLASH_CTRL3** 寄存器的 **RSTR** 位。
位 1	保留。必须保持为清除状态’0’

位 0	**BSY**：忙 该位指示闪存操作正在进行。在闪存操作开始时，该位被设置为'1'；在操作结束或发生错误时该位被清除为'0'。

5.4.16 闪存控制寄存器3（FLASH_CTRL3）

专用于闪存区块 3。

地址偏移：0x90

复位值：0x0000 0080

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															
res															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留			PRCDNIE	保留	ERRIE	保留		LOCK	RSTR	保留			CHPERS	PGERS	PRGM
res			rw	res	rw	res		rw	rw	res			rw	rw	rw

位 31~13	保留。必须保持为清除状态'0'
位 12	**PRCDNIE**：允许操作完成中断 该位允许在 **FLASH_STS3** 寄存器中的 **PRCDN** 位变为'1'时产生中断。 0：禁止产生中断； 1：允许产生中断。
位 11，9，8	保留。必须保持为清除状态'0'
位 10	**ERRIE**：允许错误状态中断 该位允许在发生 **FPEC** 错误时产生中断（当 **FLASH_STS3** 寄存器中的 **PRGMFLR/WRPRTFLR** 置为'1'时）。 0：禁止产生中断； 1：允许产生中断。
位 7	**LOCK**：锁 只能写'1'。 当该位为'1'时表示 **FPEC** 和 **FLASH_CTRL3** 被锁住。在检测到正确的解锁序列后，硬件清除此位为'0'。 在一次不成功的解锁操作后，下次系统复位前，该位不能再被改变。
位 6	**RSTR**：开始 当该位为'1'时将触发一次擦除操作。该位只可由软件置为'1'并在 **BSY** 变为'0'时 清为'0'。
位 5~3	保留。必须保持为清除状态'0'
位 2	**CHPERS**：全擦除 选择擦除所有用户页。
位 1	**PGERS**：页擦除 选择擦除页。
位 0	**PRGM**：编程 选择编程操作。

5.4.17 闪存地址寄存器3（FLASH_ADDR3）

专用于闪存区块 3。

地址偏移：0x94

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TA[31：16]															

w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TA[15：0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

这些位由硬件修改为当前/最后使用的地址。在页擦除操作中，软件必须修改这个寄存器以指定要擦除的页。

位 31~0	**TA**：闪存地址 当进行编程时选择要编程的地址，当进行页擦除时选择要擦除的页。 注意：当 **FLASH_STS3** 中的 **BSY** 位为’1’时，不能写这个寄存器。

5.4.18 闪存解密地址寄存器（FLASH_DA）

专用于闪存区块 3。

地址偏移：0x98

复位值：0x0000 0000

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

位 31~0	**FDA**：在用户程序中需要设置 FLASH_DA 寄存器来设置外部存储器加密范围。 0x0840_0000 ~ (0x0840_0000+FDA-0x1)为外部存储器加密范围 (0x0840_0000 +FDA) ~ 0x093F FFFF 为外部存储器未加密范围 注意： 当 **FLASH_STS3** 中的 **BSY** 位为’1’时，不能写这个寄存器。 FDA 的设定值必须是 4 的倍数。

5.4.19 闪存安全库区状态寄存器0（SLIB_CDR0）

专用于闪存安全库区。

地址偏移：0xCC

复位值：0x0000 0000

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

位 31~4	保留

位 3	**FSLIB_EN**：允许存放安全库代码功能 该位设定时，表示闪存主存区域部分或是全部（依照 FSLIB_CDR1 设定）作为安全库代码。 0：闪存存放安全库代码功能失效； 1：闪存存放安全库代码功能生效。
位 0~2	保留

5.4.20 闪存安全库区状态寄存器1（SLIB_CDR1）

专用于闪存安全库区。

地址偏移：0xD0

复位值：0x0000 0000

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

位 31~22	**SLIB_END_PAGE:** 主存安全库区结束页号 0：页 0 1：页 1 2：页 2 … 511: 页 511
位 21~11	**SLIB_DAT_START_PAGE:** 主存数据安全库区起始页号 0：无效页 1：页 1 2：页 2 … 511: 页 511 0x7FF: 无数据安全库区
位 10~0	**SLIB_ START_PAGE:** 主存安全库区起始页号 0：页 0 1：页 1 2：页 2 … 511: 页 511

5.4.21 闪存安全库区密码寄存器（SLIB_PSW）

专用于闪存安全库区。

地址偏移：0xD4

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FSLIB_PSW[31：16]															

w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FSLIB_PSW[15：0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

位 31~0	**FSLIB_PSW**：安全库区密码寄存器 设定正确的安全库区密码，将解除安全库区功能。

5.4.22 闪存安全库区密码设定状态寄存器（SLIB_PSW_STS）

专用于闪存安全库区。
地址偏移：0xD8
复位值：0x0100 0000

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

位 31~25	保留。必须保持为清除状态’0’
位 24~16	**SLIB_CNT[8:0]:** 剩余的安全库区设定次数 从 256 到 0
位 15~3	保留。必须保持为清除状态’0’
位 2	**SLIB_UNLOCK:** 当该位为’1’时表示寄存器偏 0xDC – 0xE3 可配置。
位 1	**SLIB_PSW_OK**：密码正确 当密码正确，并且设定的密码不等于 0xFFFF FFFF，该位被设置为’1’.
位 0	**SLIB_PSW_ERR**：密码错误 当密码错误，并且设定的密码不等于 0xFFFF FFFF，该位被设置为’1’. 注意：当该位置 1 后，硬件将不再接受重新设定密码寄存器，直到再次复位。

5.4.23 闪存安全库区密码设定寄存器（SLIB_SET_PSW）

专用于闪存安全库区密码设定。
地址偏移：0xDC
复位值：0x0000 0000

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

注意： *所有这些位是只写入，读出为 0。*

位 31~0	**FSLIB_SET_PSW[31:0]:** 配置安全库区密码 注意：在解除 FSLIB Lock 后，此 register 才允许被写入。并且 0xFFFF_FFFF 以及 0x0000_0000 值无效

5.4.24 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。
地址偏移：0xE0
复位值：0x0000 0000

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

注意： *所有这些位是只写入，读出为 0。*

位 31~22	**SLIB_SET_END_PAGE:** 配置主存安全库区结束页号 0：页 0 1：页 1 2：页 2 … 511: 页 511
位 21~11	**SLIB_SET_DAT_START_PAGE:** 配置主存数据安全库区起始页号 0：无效页，设定将导致安全库区无法启动 1：页 1 2：页 2 … 511: 页 511 0x7FF: 无数据安全库区
位 10~0	**SLIB_SET_ START_PAGE:** 配置主存安全库区起始页号 0：页 0 1：页 1 2：页 2 … 511: 页 511

注意：在解除 FSLIB Lock 后，此 register 才允许被写入。

5.4.25 闪存安全库区键值设定寄存器（SLIB_KEYR）

专用于安全库区寄存器的解锁设定。
地址偏移：0xF0
复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FSLIB_KEYR [31:16]															

w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FSLIB_KEYR [15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

注意： *所有这些位是只写入，读出为 0。*

位 31~0	**SLIB_KEY**：0xA35F_6D24 这些位用于输入 **SLIB** 的解锁键。

5.4.26 闪存CRC校验控制寄存器（CRC_DR）

专用于闪存主存区域。

地址偏移：0xF4

复位值：0x0000 0000

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

注意： *所有这些位是只写的，读出无反应。*

位 31	**CRC_START**：启动 CRC 校验 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 校验后，会自动清除该位。
位 30~24	保留。必须保持为清除状态’0’
位 23~12	**CRC_PAGE_NUM[11:0]**：CRC 校验页数量 设定本次 CRC 校验的数据量，单位是页
位 11~0	**CRC_START_PAGE[11:0]:** CRC 校验起始页 设定本次 CRC 校验从哪一页开始 0x0: 页 0 0x1: 页 1 …

5.4.27 闪存CRC校验结果寄存器（CRC_OUTR）

专用于主存或是安全库区。

地址偏移：0xF8

复位值：0x0000 0000

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

CRC_OUT[15：0]
r r r r r r r r r r r r r r r r

注意： *所有这些位是只读的，写入无反应。*

位 31~0	**CRC_OUT**：CRC 校验结果

6 CRC计算单元（CRC）

6.1 CRC简介

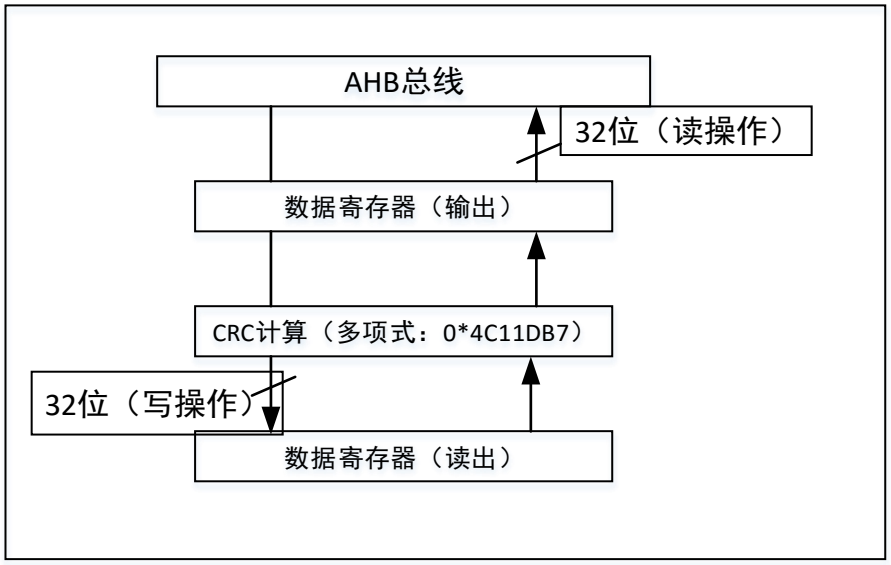
循环冗余校验（CRC）计算单元是根据固定的生成多项式得到任一 32 位全字的 CRC 计算结果。在其他的应用中，CRC 技术主要应用于核实数据传输或者数据存储的正确性和完整性。标准 EN/IEC 60335-1 即提供了一种核实闪存存储器完整性的方法。CRC 计算单元可以在程序运行时计算出软件的标识，之后与在连接时生成的参考标识比较，然后存放在指定的存储器空间。

6.2 CRC主要特性

- 使用CRC-32（以太网）多项式：0x4C11DB7
  - $X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + X^8 + X^7 + X^5 + X^4 + X^2 + X + 1$
- 一个32位数据寄存器用于输入 / 输出
- CRC计算时间：4个AHB时钟周期（HCLK）
- 通用8位寄存器（可用于存放临时数据）

下图为 CRC 计算单元框图

图 6-1 CRC单元框



6.3 CRC功能描述

CRC 计算单元含有 1 个 32 位数据寄存器：

- 对该寄存器进行写操作时，作为输入寄存器，可以输入要进行 CRC 计算的新数据。
- 对该寄存器进行读操作时，返回上一次 CRC 计算的结果。每一次写入数据寄存器，其计算结果是前一次 CRC 计算结果和新计算结果的组合（对整个 32 位字进行 CRC 计算，而不是逐字节地计算）

在 CRC 计算期间会暂停 CPU 的写操作，因此可以对寄存器 CRC_DR 进行背靠背写入或者连续地写-读操作。

可以通过设置寄存器 CRC_CTRL 的 RESET 位来重置寄存器 CRC_DR 为 0xFFFF FFFF。该操作不影响寄存器 CRC_IDR 内的数据。

6.4 CRC寄存器

CRC 计算单元包括 2 个数据寄存器和 1 个控制寄存器

下表列出了 CRC 的寄存器映像和复位值

表 6-1 CRC 计算单元寄存器映像

偏移	寄存器	31~24	23~16	15~8	7	6	5	4	3	2	1	0
0x00	CRC_DR	数据寄存器										
	复位值	0xFFFF FFFF										
0x04	CRC_IDR	保留			独立的数据寄存器							
	复位值				0x00							
0x08	CRC_CTRL	保留										RESET
	复位值											0

6.4.1 数据寄存器（CRC_DR）

地址偏移：0x00

复位值：0xFFFF FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DR[31：16]															
rw															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DR[15：0]															
rw															

位 31： 0	数据寄存器位 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

6.4.2 独立数据寄存器（CRC_IDR）

地址偏移：0x04

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															
res															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留								IDR[7：0]							
res								rw							

位 31：8	保留
位 7：0	通用 8 位数据寄存器位 可用于临时存放 1 字节的数据。 寄存器 CRC_CTRL 的 RESET 位产生的 CRC 复位对本寄存器没有影响

6.4.3 控制寄存器（CRC_CTRL）

地址偏移：0x08

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															
res															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留															RESET
res															r

位 31：1	保留
位 0	RESET 位 复位 CRC 计算单元，设置数据寄存器为 0xFFFF FFFF。 只能对该位写’1’，它由硬件自动清’0’。

7 通用和复用功能I/O（GPIO和AFIO）

7.1 简介

GPIO 接口包括 5 组通用输入/输出端口。
每个 GPIO 组提供 16 个通用输入/输出引脚；每个 GPIO 端口都有相关的控制和配置寄存器来满足特定的功能，GPIO 引脚上的外部中断都有相关的控制和配置寄存器在外部中断控制器，参考章节 8.中断和事件。

GPIO 端口和其他复用功能同用一个引脚，在特定的封装下获得最大的灵活性。GPIO 引脚可以用于复用功能引脚，通过配置相关的寄存器用作复用功能输入和复用功能输出。

每个 GPIO 引脚可通过软件配置为输出（推挽或开漏）,输入（上拉，下拉或无上拉/下拉）或作为外设复用功能。大部分的 GPIO 引脚都有数字或模拟的复用功能。所有的 GPIO 都具备大电流驱动能力。

7.2 主要特征

- 输入/输出方向控制
- 每个引脚都有弱上拉/下拉功能
- 推挽/开漏输出使能控制
- 输出置位/复位控制
- 可编程触发沿的外部中断 - 在外部中断配置寄存器中
- 模拟输入/模拟输出配置
- 复用功能输入/输出配置
- 端口配置锁定

7.3 功能描述

7.3.1 GPIO引脚配置

每个 GPIO 端口有两个 32 位配置寄存器（GPIOx_CTRLL，GPIOx_CTRLH），两个 32 位数据寄存器（GPIOx_IPTDT 和 GPIOx_OPTDT），一个 32 位置位/复位寄存器（GPIOx_BSRE），一个 16 位复位寄存器（GPIOx_BRE）和一个 32 位锁定寄存器（GPIOx_LOCK）。

根据数据手册中列出的每个 I/O 端口的特定硬件特征，GPIO 端口的每个位可以由软件分别配置成多种模式。

- 输入浮空
- 输入上拉
- 输入下拉
- 模拟输入
- 开漏输出
- 推挽式输出
- 推挽式复用功能
- 开漏复用功能

每个 I/O 端口位可以自由编程，然而 I/O 端口寄存器必须按 32 位字被访问（不允许半字或字节访问）。GPIOx_BSRE 和 GPIOx_BRE 寄存器允许对任何 GPIO 寄存器的读/更改的独立访问；这样，在读和更改访问之间产生 IRQ 时不会发生危险。

下图给出了一个 I/O 端口位的基本结构。

图 7-1 I/O端口位的基本结构

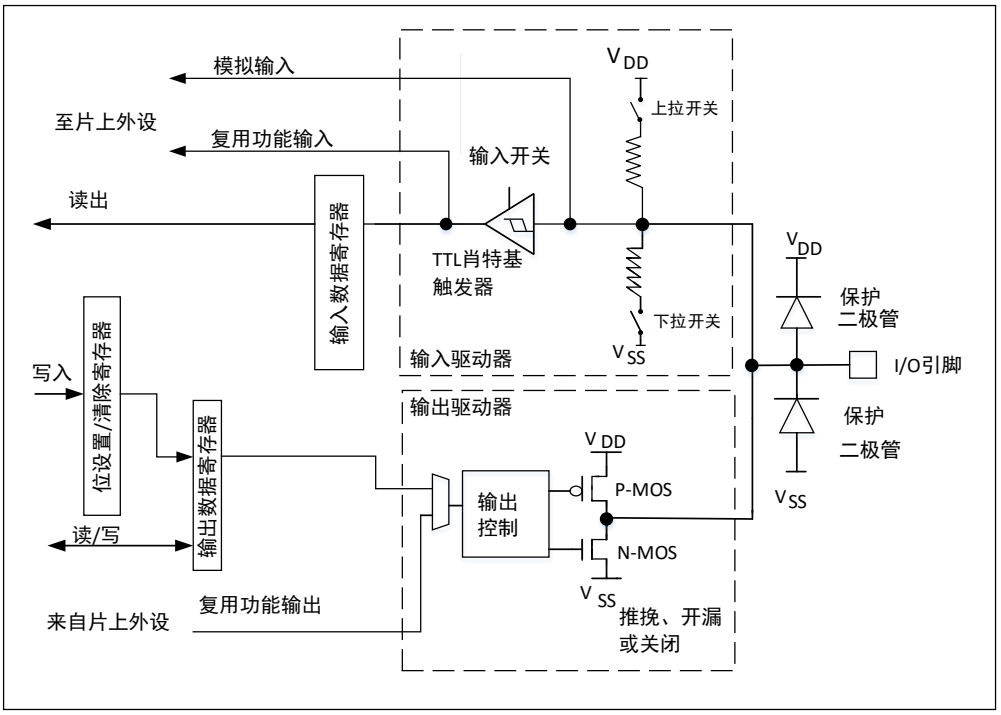
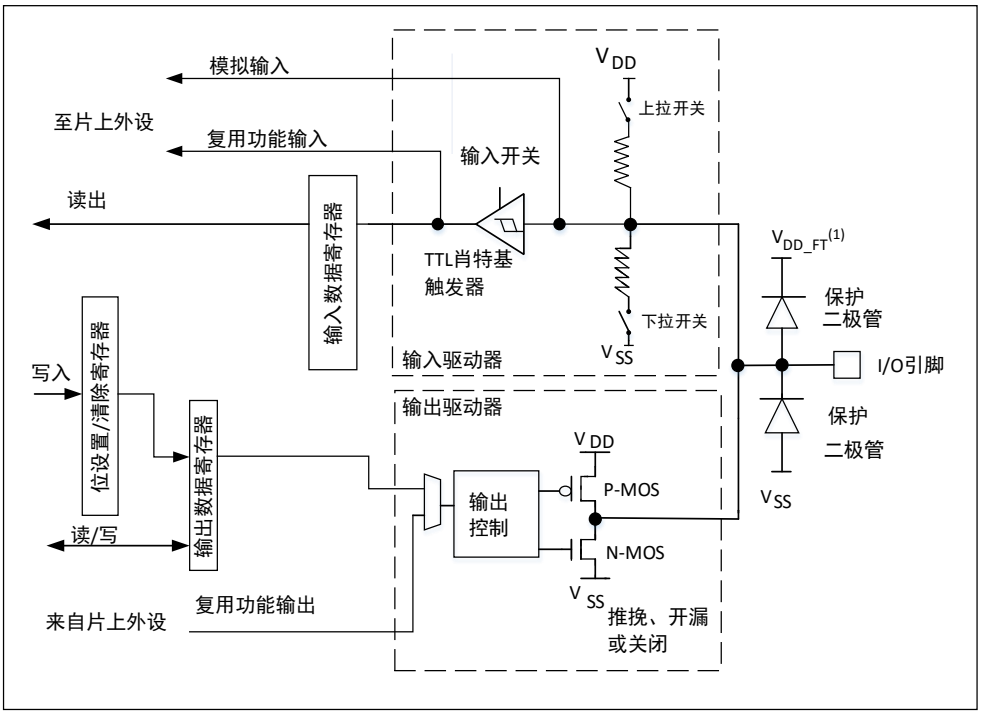


图 7-2 伏兼容I/O端口位的基本结构



注意：V_{DD_FT} 对 5 伏容忍 I/O 脚是特殊的，它与 V_{DD} 不同。

表 7-1 端口位配置表

配置模式		CONF1	CONF0	MDE1	MDE0	PxOPTDT 寄存器
通用输出	推挽（Push-Pull）	0	0	01 10 11 见表 7-2		0 或 1
	开漏（Open-Drain）		1			0 或 1
复用功能输出	推挽（Push-Pull）	1	0			不使用
	开漏（Open-Drain）		1			不使用
输入	模拟输入		0	00		不使用

	浮空输入	0	1		不使用
	下拉输入	1	0		0
	上拉输入				1

表 7-2 输出模式位

MDE[1：0]	意义
00	保留
01	较大电流推动/吸入能力
10	适中电流推动/吸入能力
11	适中电流推动/吸入能力

复位期间和刚复位后，复用功能未开启，I/O 端口被配置成浮空输入模式（CONFx[1：0]=01b，MDEx[1：0]=00b）。

复位后，JTAG 引脚被置于输入上拉或下拉模式：

- PA15：JTDI置于上拉模式
- PA14：JTCK置于下拉模式
- PA13：JTMS置于上拉模式
- PB4：JNTRST置于上拉模式

当作为输出配置时，写到输出数据寄存器（GPIOx_OPTDT）上的值会输出到相应的 I/O 引脚。可以以推挽模式或开漏模式（当输出 0 时，只有 N-MOS 被打开）使用输出驱动器。

输入数据寄存器（GPIOx_IPTDT）在每个 AHB 时钟周期捕捉 I/O 引脚上的数据。所有 GPIO 引脚有一个内部弱上拉和弱下拉，当配置为输入时，它们可以被激活也可以被断开。

当对 GPIOx_OPTDT 的个别位编程时，软件不需要禁止中断：在单次 AHB 写操作里，可以只更改一个或多个位。

这是通过对“置位/复位寄存器”（GPIOx_BSRE，复位是 GPIOx_BRE）中想要更改的位写’1’来实现的。没被选择的位将不被更改。

7.3.2 外部中断/唤醒线

所有端口都有外部中断能力。为了使用外部中断线，端口必须配置成输入模式。更多的关于外部中断的信息，参考 8.2 节外部中断/事件控制器（EXTI）。

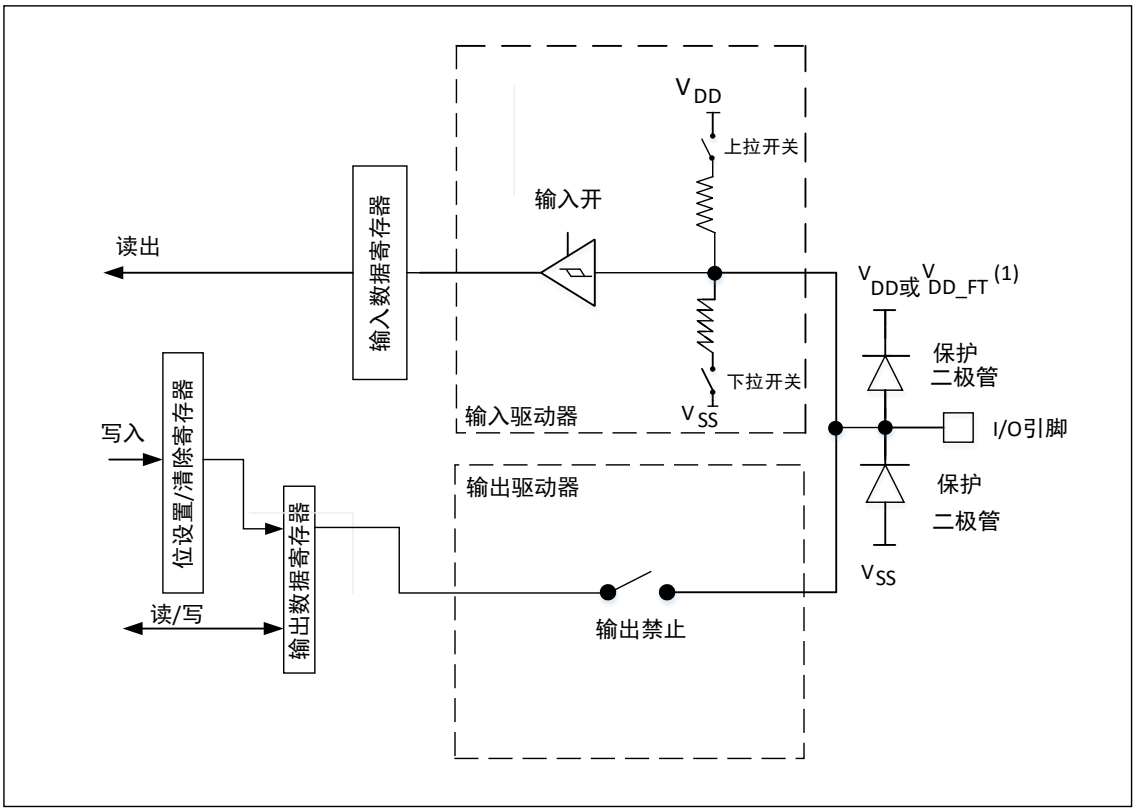
7.3.3 输入配置

当 I/O 端口配置为输入时：

- 输出缓冲器被禁止
- 施密特触发输入被激活
- 根据输入配置（上拉，下拉或浮动）的不同，弱上拉和下拉电阻被连接
- 出现在I/O脚上的数据在每个AHB时钟被采样到输入数据寄存器
- 对输入数据寄存器的读访问可得到I/O状态

下图给出了 I/O 端口位的输入配置

图 7-3 输入浮空/上拉/下拉配置



注意：V_{DD_FT} 对 5 伏容忍 I/O 脚是特殊的，它与 V_{DD} 不同

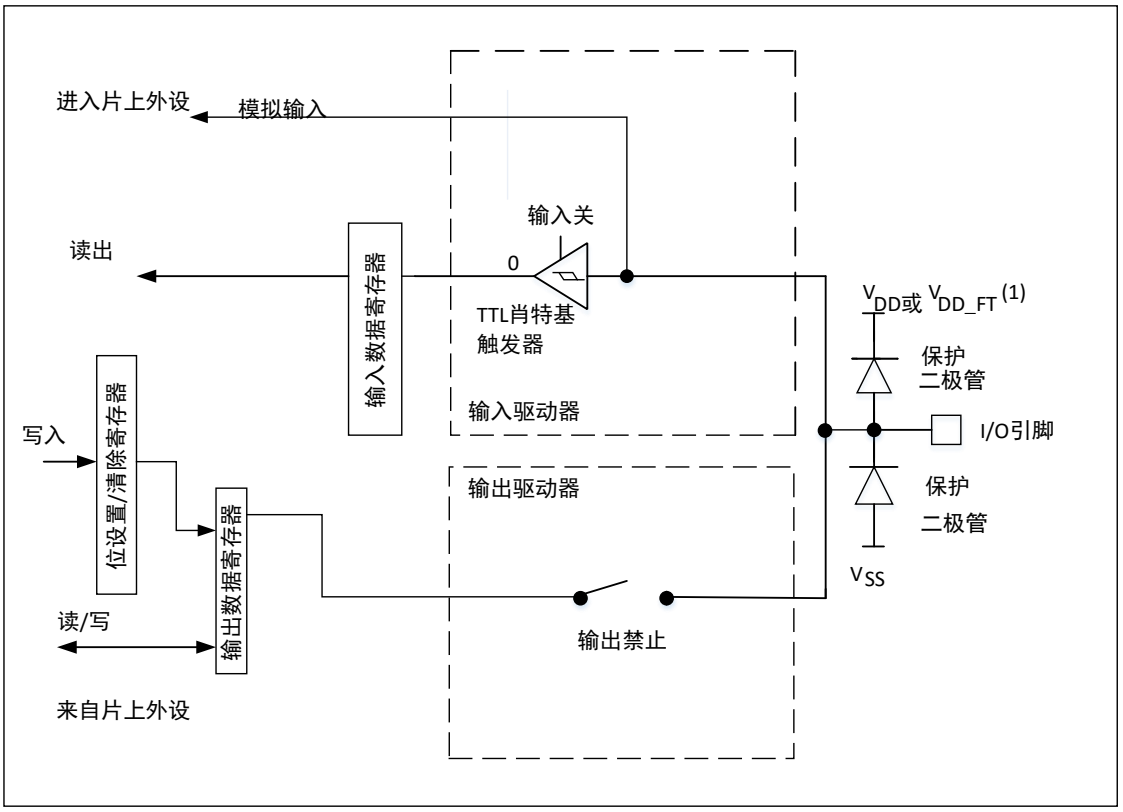
7.3.4 模拟输入配置

当 I/O 端口被配置为模拟输入配置时：

- 输出缓冲器被禁止
- 禁止施密特触发输入，实现了每个模拟I/O引脚上的零消耗。施密特触发输出值被强置为’0’
- 弱上拉和下拉电阻被禁止
- 读取输入数据寄存器时数值为’0’

下图给出了 I/O 端口位的高阻抗模拟输入配置：

图 7-4 高阻抗的模拟输入配置



注意：V_{DD_FT} 对 5 伏兼容 I/O 脚是特殊的，它与 V_{DD} 不同

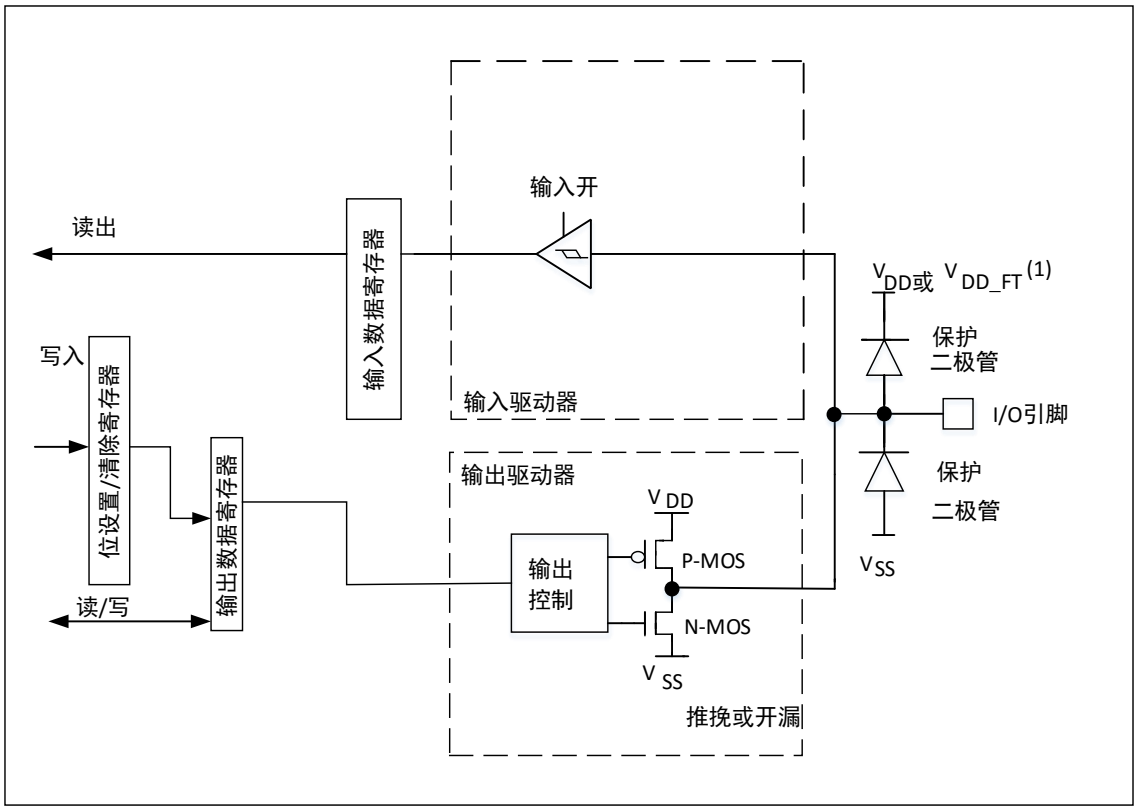
7.3.5 输出配置

当 I/O 端口被配置为输出时：

- 输出缓冲器被激活
  - 开漏模式：输出寄存器上的’0’激活N-MOS，而输出寄存器上的’1’将端口置于高阻状态（P-MOS从不被激活）
  - 推挽模式：输出寄存器上的’0’激活N-MOS，而输出寄存器上的’1’将激活P-MOS
- 施密特触发输入被激活
- 弱上拉和下拉电阻被禁止
- 出现在I/O脚上的数据在每个AHB时钟被采样到输入数据寄存器
- 在开漏模式时，对输入数据寄存器的读访问可得到I/O状态
- 在推挽式模式时，对输出数据寄存器的读访问得到最后一次写的值

下图给出了 I/O 端口位的输出配置。

图 7-5 输出配置



注意：V_{DD_FT} 对 5 伏兼容 I/O 脚是特殊的，它与 V_{DD} 不同

7.3.6 GPIO锁定机制

锁定机制允许冻结 IO 配置。当在一个端口位上执行了锁定（LOCK）程序，在下一次复位之前，将不能再更改端口位的配置。

7.3.7 复用功能（AF）

使用默认复用功能前必须对端口位配置寄存器编程。

- 对于复用的输入功能，端口必须配置成输入模式（浮空、上拉或下拉）且输入引脚必须由外部驱动。

注意：也可以通过软件来模拟复用功能输入引脚，这种模拟可以通过对 GPIO 控制器编程来实现。此时，端口应当被设置为复用功能输出模式。显然，这时相应的引脚不再由外部驱动，而是通过 GPIO 控制器由软件来驱动。

- 对于复用输出功能，端口必须配置成复用功能输出模式（推挽或开漏）。
- 对于双向复用功能，端口位必须配置复用功能输出模式（推挽或开漏）。这时，输入驱动器被配置成浮空输入模式。

如果把端口配置成复用输出功能，则引脚和输出寄存器断开，并和片上外设的输出信号连接。如果软件把一个 GPIO 脚配置成复用输出功能，但是外设没有被激活，它的输出将不确定。

当 I/O 端口被配置为复用功能时：

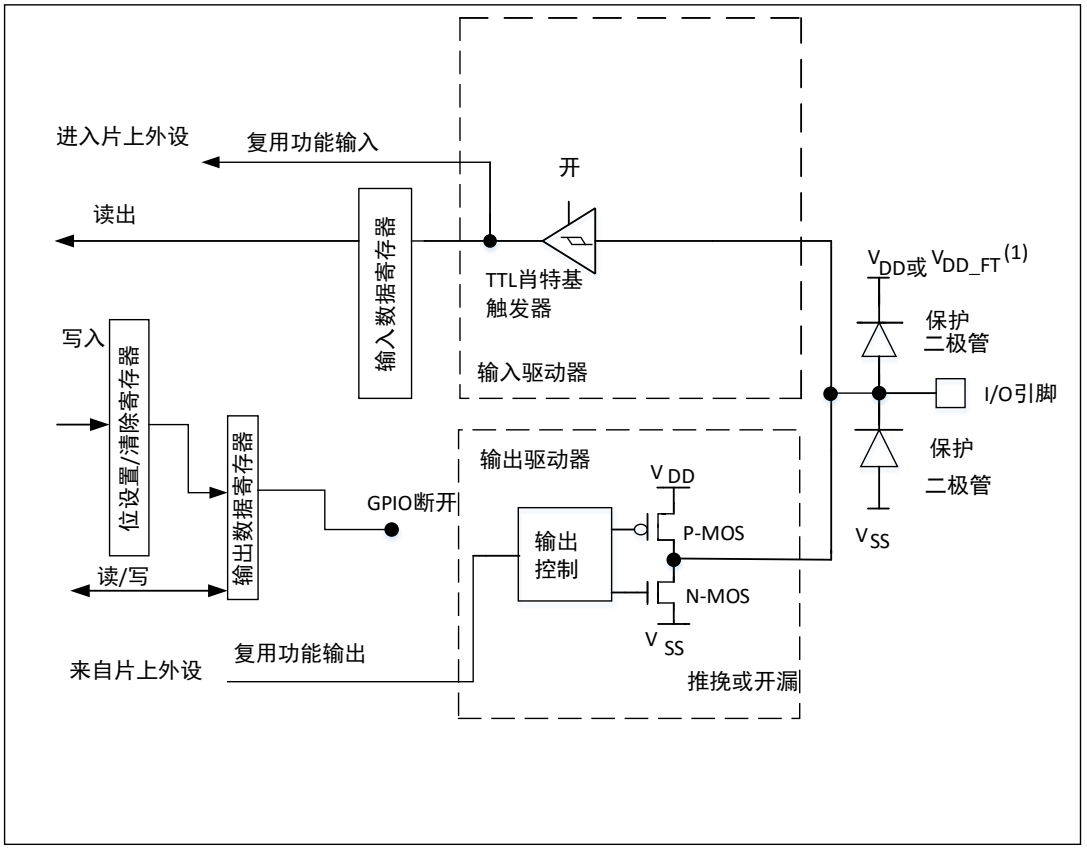
- 在开漏或推挽式配置中，输出缓冲器被打开
- 内置外设的信号驱动输出缓冲器（复用功能输出）
- 施密特触发输入被激活
- 弱上拉和下拉电阻被禁止
- 如果引脚被误配成多个复用功能输出，各复用功能优先级请参考数据手册。

- 在每个AHB时钟周期，出现在I/O脚上的数据被采样到输入数据寄存器
- 开漏模式时，读输入数据寄存器时可得到I/O口状态
- 在推挽模式时，读输出数据寄存器时可得到最后一次写的值

下图示出了 I/O 端口位的复用功能配置。详见 7.4 节 IO 映射功能配置。

一组复用功能 I/O 寄存器允许用户把一些复用功能重新映象到不同的引脚。

图 7-6 复用功能配置



注意：V_{DD_FT} 对 5 伏兼容 I/O 脚是特殊的，它与 V_{DD} 不同

为了使不同器件封装的外设 I/O 功能的数量达到最优，可以把一些复用功能重新映射到其他一些脚上。这可以通过软件配置相应的寄存器来完成（参考第 7.5 节 GPIO 与 AFIO 寄存器）。这时，复用功能就不再映射到它们的原始引脚上了。

下列表格列出了各个外设的引脚配置。

表 7-3 高级定时器TMR1/8

TMR1/8 引脚	配置	GPIO 配置
TMR1/8_CHx	输入捕获通道 x	浮空输入
	输出比较通道 x	推挽复用输出
TMR1/8_CHxN	互补输出通道 x	推挽复用输出
TMR1/8_BKIN	刹车输入	浮空输入
TMR1/8_ETR	外部触发时钟输入	浮空输入

表 7-4 通用定时器TMR2-5/TMR9-14

TMRx 引脚	配置	GPIO 配置
TMRx_CHx	输入捕获通道 x	浮空输入

	输出比较通道 x	推挽复用输出
TMRx_ETR	外部触发时钟输入	浮空输入

表 7-5 USART

USART 引脚	配置	GPIO 配置
USARTx_TX	全双工模式	推挽复用输出
	半双工同步模式	推挽复用输出
USARTx_RX	全双工模式	浮空输入或带上拉输入
	半双工同步模式	未用，可作为通用 I/O
USARTx_CK	同步模式	推挽复用输出
USARTx_RTS	硬件流量控制	推挽复用输出
USARTx_CTS	硬件流量控制	浮空输入或带上拉输入

表 7-6 SPI

SPI 引脚	配置	GPIO 配置
SPIx_SCK	主模式	推挽复用输出
	从模式	浮空输入
SPIx_MOSI	全双工模式/主模式	推挽复用输出
	全双工模式/从模式	浮空输入或带上拉输入
	简单的双向数据线/主模式	推挽复用输出
	简单的双向数据线/从模式	未用，可作为通用 I/O
SPIx_MISO	全双工模式/主模式	浮空输入或带上拉输入
	全双工模式/从模式	推挽复用输出
	简单的双向数据线/主模式	未用，可作为通用 I/O
	简单的双向数据线/从模式	推挽复用输出
SPIx_NSS	硬件主/从模式	浮空输入或带上拉输入或带下拉输入
	硬件主模式/NSS 输出使能	推挽复用输出
	软件模式	未用，可作为通用 I/O

表 7-7 I^2S

I2S 引脚	配置	GPIO 配置
I2Sx_WS	主模式	推挽复用输出
	从模式	浮空输入
I2Sx_CK	主模式	推挽复用输出
	从模式	浮空输入
I2Sx_SD I2Sx_SD_ext	发送器	推挽复用输出
	接收器	浮空输入或带上拉输入或带下拉输入
I2Sx_MCK	主模式	推挽复用输出

	从模式	未用，可作为通用 I/O

表 7-8 I²C接口

I2C 引脚	配置	GPIO 配置
I2Cx_SCL	I2C 时钟	开漏复用输出
I2Cx_SDA	I2C 数据	开漏复用输出
I2Cx_SMBA	SMBUS alert	开漏复用输出

表 7-9 BxCAN

BxCAN 引脚	GPIO 配置
CAN_TX	推挽复用输出
CAN_RX	浮空输入或带上拉输入

表 7-10 USB

USB 引脚	GPIO 配置
USB_DM/USB_DP	一旦使能了 USB 模块，这些引脚会自动连接到内部 USB 收发器
USB_SOF	推挽复用输出

表 7-11 SDIO

SDIO 引脚	GPIO 配置
SDIO_CK	推挽复用输出
SDIO_CMD	推挽复用输出
SDIO[D7：D0]	推挽复用输出

表 7-12 ADC/DAC

ADC/DAC 引脚	GPIO 配置
ADC/DAC	模拟输入

注意：ADC 输入引脚必须配置为模拟输入。

表 7-13 Ethernet

Ethernet 引脚	GPIO 配置
ETH_MDC	推挽复用输出
ETH_MII_TXD2	推挽复用输出
ETH_MII_TX_CLK	浮空输入
ETH_MII_CRS	浮空输入
ETH_MII_RX_CLK ETH_RMII_REF_CLK	浮空输入
ETH_MDIO	推挽复用输出
ETH_MII_COL	浮空输入
ETH_MII_RX_DV ETH_RMII_CRS_DV	浮空输入

ETH_MII_RXD0 ETH_RMII_RXD0	浮空输入
ETH_MII_RXD1 ETH_RMII_RXD1	浮空输入
ETH_MII_RXD2	浮空输入
ETH_MII_RXD3	浮空输入
ETH_MII_RX_ER	浮空输入
ETH_MII_TX_EN ETH_RMII_TX_EN	推挽复用输出
ETH_MII_TXD0 ETH_RMII_TXD0	推挽复用输出
ETH_MII_TXD1 ETH_RMII_TXD1	推挽复用输出
ETH_PPS_OUT	推挽复用输出
ETH_MII_TXD3	推挽复用输出

表 7-14 XMC

XMC 引脚	GPIO 配置
XMC_A[25：0] XMC_D[15：0]	推挽复用输出
XMC_CK	推挽复用输出
XMC_NOE XMC_NWE	推挽复用输出
XMC_NE[4：1] XMC_NCE[3：2] XMC_NCE4_1 XMC_NCE4_2	推挽复用输出
XMC_NWAIT XMC_CD	浮空输入或带上拉输入
XMC_NIOS16 XMC_INTR XMC_INT[3：2]	浮空输入
XMC_NL XMC_NBL[1：0]	推挽复用输出
XMC_NIORD XMC_NIOWR XMC_NREG	推挽复用输出

表 7-15 其它I/O功能

引脚	复用功能	GPIO 配置
TAMPER-RTC	RTC 输出	当配置 BRKP_CTRL 和 BRKP_RTCCAL 寄存器时，由硬件强制设置
	侵入事件输入	
CLKOUT	时钟输出	推挽复用输出
EXTINT 输入线	外部中断输入	浮空输入或带上拉输入或带下拉输入

7.4 IO映射功能配置

为了优化不同封装的外设数目，可以把一些复用功能重新映射到其他引脚上。设置复用重映射和调试 I/O 配置寄存器（AFIO_MAPx）或复用重映射和调试 I/O 配置寄存器 x（AFIO_MAPx）(x=2,3…8)实现引脚的重新映射。这时，复用功能不再映射到它们的原始分配上。

7.4.1 把OSC32_IN/OSC32_OUT作为GPIO端口PC14/PC15

当 LSE 振荡器关闭时，LSE 振荡器引脚 OSC32_IN/OSC32_OUT 可以分别用做 GPIO 的 PC14/PC15，LSE 功能始终优先于通用 I/O 口的功能。

7.4.2 把OSC_IN/OSC_OUT引脚作为GPIO端口PD0/PD1

外部振荡器引脚 OSC_IN/OSC_OUT 可以用做 GPIO 的 PD0/PD1，通过设置复用重映射和调试 I/O 配置寄存器（AFIO_MAP）或复用重映射和调试 I/O 配置寄存器 7（AFIO_MAP7）实现。

7.4.3 CAN复用功能重映射

CAN 信号可以被映射到端口 A、端口 B、或端口 D 上，如下表所示。参阅复用重映射和调试 I/O 配置寄存器（AFIO_MAP）或重映射和调试 I/O 配置寄存器 6（AFIO_MAP6）。

表 7-16 CAN1复用功能重映射

复用功能	CAN_REMAP[1：0]=”00”或 CAN1_GRMP[3:0]=”0000”	CAN_REMAP[1：0]=”10”或 CAN1_GRMP[3:0]=”0010”	CAN_REMAP[1：0]=”11”或 CAN1_GRMP[3:0]=”0011”
CAN_RX	PA11	PB8	PD0
CAN_TX	PA12	PB9	PD1

表 7-17 CAN2复用功能重映射

复用功能	CAN2_REMAP=”0”或 CAN2_GRMP[3:0]=”0000”	CAN2_REMAP=”1”或 CAN2_GRMP[3:0]=”0001”
CAN2_RX	PB12	PB5
CAN2_TX	PB13	PB6

7.4.4 JTAG/SWD复用功能重映射

调试接口信号被映射到 GPIO 端口上，如下表所示。

表 7-18 调试接口信号

复用功能	**GPIO** 端口
JTMS/SWDIO	PA13
JTCK/SWCLK	PA14
JTDI	PA15
JTDO/TRACESWO	PB3
JNTRST	PB4
TRACECK	PE2
TRACED0	PE3
TRACED1	PE4
TRACED2	PE5
TRACED3	PE6

为了在调试期间可以使用更多 GPIOs，通过设置复用重映射和调试 I/O 配置寄存器（AFIO_MAP）的

SWJTAG_CONF[2：0]位或复用重映射和调试 I/O 配置寄存器 7（AFIO_MAP7）的 SWJTAG_GRMP[2：0]位，可以改变上述重映射配置。

表 7-19 调试端口映射

SWJTAG_CONF[2：0]或SWJTAG_GRMP[2：0]	可能的调试端口	SWJI/O 引脚分配				
		PA13/JTMS/SWDIO	PA14/JTCK/SWCLK	PA15/JTDI	PB3/JTDO/TRACESWO	PB4/NJTRST
000	完全 SWJ（JTAG-DP+SW-DP）（复位状态）	I/O 不可用	I/O 不可用	I/O 不可用	I/O 不可用	I/O 不可用
001	完全 SWJ（JTAG-DP+ SW-DP）但没有 JNTRST	I/O 不可用	I/O 不可用	I/O 不可用	I/O 不可用	I/O 可用
010	关闭 JTAG-DP，启用 SW-DP	I/O 不可用	I/O 不可用	I/O 可用	I/O 可用 (1)	I/O 可用
100	关闭 JTAG-DP，关闭 SW-DP	I/O 可用	I/O 可用	I/O 可用	I/O 可用	I/O 可用
其它	禁用	-	-	-	-	-

注意：I/O 口只可在不使用异步跟踪时使用。

7.4.5 ADC复用功能重映射

参阅复用重映射和调试 I/O 配置寄存器（AFIO_MAP）或重映射和调试 I/O 配置寄存器 7（AFIO_MAP7）。

表 7-20 ADC1外部触发注入转换复用功能重映射

复用功能	ADC1_ETI_GRMP=0 或 ADC1_EXTRGINJ_REMAP=0	ADC1_ETI_GRMP=1 或 ADC1_EXTRGINJ_REMAP=1
ADC1 外部触发注入转换	ADC1 外部触发注入转换与 EXTINT15 相连	ADC1 外部触发注入转换与 TMR8_CH4 相连

表 7-21 ADC1外部触发规则转换复用功能重映射

复用功能	ADC1_ETR_GRMP=0 或 ADC1_EXTRGREG_REMAP=0	ADC1_ETR_GRMP=1 或 ADC1_EXTRGREG_REMAP=1
ADC1 外部触发规则转换	ADC1 外部触发规则转换与 EXTINT11 相连	ADC1 外部触发规则转换与 TMR8_TRGO 相连

表 7-22 ADC2外部触发注入转换复用功能重映射

复用功能	ADC2_ETI_GRMP =0 或 ADC2_EXTRGINJ_REMAP=0	ADC2_ETI_GRMP =1 或 ADC2_EXTRGINJ_REMAP=1
ADC2 外部触发注入转换	ADC2 外部触发注入转换与 EXTINT15 相连	ADC2 外部触发注入转换与 TMR8_CH4 相连

表 7-23 ADC2外部触发规则转换复用功能重映射

复用功能	ADC2_ETR_GRMP =0 或 ADC1_EXTRGREG_REMAP=0	ADC2_ETR_GRMP =1 或 ADC2_EXTRGREG_REMAP=1
ADC2 外部触发规则转换	ADC2 外部触发规则转换与 EXTINT11 相连	ADC2 外部触发规则转换与 TMR8_TRGO 相连

7.4.6 定时器复用功能重映射

参阅复用重映射和调试 I/O 配置寄存器（AFIO_MAP）、复用重映射和调试 I/O 配置寄存器 2（AFIO_MAP2）、复用重映射和调试 I/O 配置寄存器 3（AFIO_MAP3）或重映射和调试 I/O 配置寄存器 4（AFIO_MAP4）。

表 7-24 TMR1复用功能重映射

复用功能映射	TMR1_REMAP[1：0]=00 或 TMR1_GRMP[3:0]=0000（没有重映射）	TMR1_REMAP[1：0]=01 或 TMR1_GRMP[3:0]=0001（部分重映射）	TMR1_REMAP[1：0]=11 或 TMR1_GRMP[3:0]=0011（完全重映射）(1)
TMR1_ETR	PA12		PE7

TMR1_CH1	PA8		PE9
TMR1_CH2	PA9		PE11
TMR1_CH3	PA10		PE13
TMR1_CH4	PA11		PE14
TMR1_BKIN	PB12	PA6	PE15
TMR1_CH1N	PB13	PA7	PE8
TMR1_CH2N	PB14	PB0	PE10
TMR1_CH3N	PB15	PB1	PE12

表 7-25 TMR2复用功能重映射

复用功能	TMR2_GRMP[2:0]=000 或 TMR2_REMAP[1:0]=00	TMR2_GRMP[2:0]=001 或 TMR2_REMAP[1:0]=01	TMR2_GRMP[2:0]=010 或 TMR2_REMAP[1:0]=10	TMR2_GRMP[2:0]=011 或 TMR2_REMAP[1:0]=11
TMR2_CH1_ETR(1)	PA0	PA15	PA0	PA15
TMR2_CH2	PA1	PB3	PA1	PB3
TMR2_CH3	PA2		PB10	
TMR2_CH4	PA3		PB11	

注意：TMR2_CH1 和 TMR2_ETR 共用一个引脚，但不能同时使用（因此在此使用这样的标记：TMR2_CH1_ETR）。

表 7-26 TMR3复用功能重映射

复用功能	TMR3_GRMP[3:0]=0000 或 TMR3_REMAP[1：0]=00 （没有重映射）	TMR3_GRMP[3:0]=0010 TMR3_REMAP[1：0]=10 （部分重映射）	TMR3_GRMP[3:0]=0011 TMR3_REMAP[1：0]=11 （完全重映射）(1)
TMR3_CH1	PA6	PB4	PC6
TMR3_CH2	PA7	PB5	PC7
TMR3_CH3	PB0		PC8
TMR3_CH4	PB1		PC9

注意：重映射只适用于 64、100 脚的封装。

表 7-27 TMR4复用功能重映射

复用功能	TMR4_REMAP=0 或 TMR4_GRMP[3:0]=0000	TMR4_REMAP=1 或 TMR4_GRMP[3:]=0001
TMR4_CH1	PB6	PD12
TMR4_CH2	PB7	PD13
TMR4_CH3	PB8	PD14
TMR4_CH4	PB9	PD15

表 7-28 TMR5复用功能重映射

复用功能	TMR5CH4_INTLRE=0 或 TMR5CH4_IRMP=0	TMR5CH4_INTLRE=1 或 TMR5CH4_IRMP=1
TMR5_CH4	TMR5 的通道 4 连至 PA3	LSI 内部时钟连至 TMR5_CH4 输入作为校准使用

表 7-29 TMR9复用功能重映射

复用功能映射	TMR9_GRMP[3:0]=0000 或 TMR9_REMAP=0	TMR9_GRMP[3:0]=0010 或 TMR9_REMAP=1
TMR9_CH1	PA2	PE5
TMR9_CH2	PA3	PE6

7.4.7 USART/UART复用功能重映射

参见复用重映射和调试 I/O 配置寄存器（AFIO_MAP）、复用重映射和调试 I/O 配置寄存器 6（AFIO_MAP6）或复用重映射和调试 I/O 配置寄存器 8（AFIO_MAP8）。

表 7-30 USART1重映射

复用功能	USART1_GRMP[3:0]=0000 或 USART1_REMAP=0	USART1_GRMP[3:0]=0001 或 USART1_REMAP=1
USART1_TX	PA9	PB6
USART1_RX	PA10	PB7

表 7-31 USART2重映射

复用功能	USART2_REMAP=0 或 USART2_GRMP[3:0]=0000	USART2_REMAP=1 (1) USART2_GRMP[3:0]=0001
USART2_CTS	PA0	PD3
USART2_RTS	PA1	PD4
USART2_TX	PA2	PD5
USART2_RX	PA3	PD6
USART2_CK	PA4	PD7

注意：重映射只适用于 100 脚的封装。

表 7-32 USART3重映射

复用功能	USART3_REMAP[1：0]=00 或 USART3_GRMP[3:0]=0000 （没有重映射）	USART3_REMAP[1：0]=01 或 USART3_GRMP[3:0]=0001 （部分重映射）(1)	USART3_REMAP[1：0]=11 或 USART3_GRMP[3:0]=0011 （完全重映射）(2)
USART3_TX	PB10	PC10	PD8
USART3_RX	PB11	PC11	PD9
USART3_CK	PB12	PC12	PD10
USART3_CTS	PB13		PD11
USART3_RTS	PB14		PD12

注意： 1.重映射只适用于 64、100 脚的封装；

2.重映射只适用于 100 脚的封装。

表 7-33 UART4重映射

复用功能	UART4_GRMP[3:0]=0000	UART4_GRMP[3:0]=0010
UART4_TX	PC10	PA0
UART4_RX	PC11	PA1

表 7-34 UART5重映射

复用功能	UART5_GRMP[3:0]=0000	UART5_GRMP[3:0]=0001
UART5_TX	PC12	PB9
UART5_RX	PD2	PB8

表 7-35 USART6重映射

复用功能	USART6_GRMP[3:0]=0000	USART6_GRMP[3:0]=0001
USART6_TX	PC6	PA4
USART6_RX	PC7	PA5

表 7-36 UART7重映射

复用功能	UART7_GRMP[3:0]=0000	UART7_GRMP[3:0]=0001
UART7_TX	PE8	PB4
UART7_RX	PE7	PB3

表 7-37 UART8重映射

复用功能	UART8_GRMP[3:0]=0000	UART8_GRMP[3:0]=0001
UART8_TX	PE1	PC2
UART8_RX	PE0	PC3

7.4.8 I2C复用功能重映射

参见复用重映射和调试 I/O 配置寄存器（AFIO_MAP）或复用重映射和调试 I/O 配置寄存器 5（AFIO_MAP5）。

表 7-38 I²C1重映射

复用功能	I2C1_REMAP=0 或 I2C1_GRMP[3:0]=0000	I2C1_REMAP=1 或 I2C1_GRMP[3:0]=0001
I2C1_SCL	PB6	PB8
I2C1_SDA	PB7	PB9
I2C1_SMBA	PB5	

表 7-39 I²C3重映射

复用功能	I2C3_REMAP=0 或 I2C3_GRMP[3:0]=0000	I2C3_REMAP=1 或 I2C3_GRMP[3:0]=0001
I2C3_SCL	PA8	
I2C3_SDA	PC9 (1)	PB4
I2C3_SMBA	PA9	

7.4.9 SPI/I2S复用功能重映射

参见复用重映射和调试 I/O 配置寄存器（AFIO_MAP）或复用重映射和调试 I/O 配置寄存器 5（AFIO_MAP5）。

表 7-40 SPI1/I²S1重映射

复用功能	SPI1_GRMP[3:0]=0000 或 SPI1_REMAP[1：0]=00	SPI1_GRMP[3:0]=0001 或 SPI1_REMAP[1：0]=01	SPI1_GRMP[3:0]=0010 或 SPI1_REMAP[1：0]=10	SPI1_GRMP[3:0]=0011 或 SPI1_REMAP[1：0]=11
SPI1_NSS/I2S1_WS	PA4	PA15	PA4	PA15
SPI1_SCK/I2S1_CK	PA5	PB3	PA5	PB3
SPI1_MISO	PA6	PB4	PA6	PB4
SPI1_MOSI/I2S1_SD	PA7	PB5	PA7	PB5
I2S1_MCK	PB0		PB6	

表 7-41 SPI2/I²S2重映射

复用功能	SPI2_GRMP[3:0]=0000	SPI2_GRMP[3:0]=0001	SPI2_GRMP[3:0]=0010
I2S2_MCK	PC6	PA3	PA6

表 7-42 SPI3/I²S3/I²SEXT3重映射

复用功能	SPI3_REMAP=0 SPI3_GRMP[3:0]=0000	SPI3_REMAP=1 SPI3_GRMP[3:0]=0001	SPI3_GRMP[3:0]=0010	SPI3_GRMP[3:0]=0011
SPI3_NSS/I2S3_WS	PA15	PA4	PA15	PA4
SPI3_SCK/I2S3_CK	PB3	PC10	PB3	PC10
SPI3_MISO/I2Sext3_SD	PB4	PC11	PB4	PC11
SPI3_MOSI/I2S3_SD	PB5	PC12	PB5	PC12
I2S3_MCK	PC7		PB10	

表 7-43 SPI4/IS24重映射

复用功能	SPI4_REMAP=0 或 SPI4_GRMP[3:0]=0000	SPI4_REMAP=1 或 SPI4_GRMP[3:0]=0001	SPI4_GRMP[3:0]=0010	SPI4_GRMP[3:0]=0011
SPI4_NSS/I2S4_WS	PE4	PE12	PB6	
SPI4_SCK/I2S4_CK	PE2	PE11	PB7	
SPI4_MISO/	PE5	PE13	PB8	
SPI4_MOSI/I2S4_SD	PE6	PE14	PB9	
I2S4_MCK	PC8			PA10

7.4.10 SDIO复用功能重映射

参见复用重映射和调试 I/O 配置寄存器 2（AFIO_MAP2）或复用重映射和调试 I/O 配置寄存器 6（AFIO_MAP6）。

表 7-44 SDIO2 Dx复用功能重映射

复用功能	SDIO2_REMAP[0]=0 或 SDIO2_GRMP[0]=0	SDIO2_REMAP[0]=1 或 SDIO2_GRMP[0]=1
SDIO2_D0	PC0	PA4
SDIO2_D1	PC1	PA5
SDIO2_D2	PC2	PA6
SDIO2_D3	PC3	PA7

SDIO2_D4	PA4	-
SDIO2_D5	PA5	-
SDIO2_D6	PA6	-
SDIO2_D7	PA7	-

表 7-45 SDIO2 CK/CMD复用功能重映射

复用功能	SDIO2_REMAP[1]=0 或 SDIO2_GRMP[1]=0	SDIO2_REMAP[1]=1 或 SDIO2_GRMP[1]=1
SDIO2_CK	PC4	PA2
SDIO2_CMD	PC5	PA3

7.4.11 外部SPIF复用功能重映射

参见复用重映射和调试 I/O 配置寄存器 2（AFIO_MAP2）或复用重映射和调试 I/O 配置寄存器 7（AFIO_MAP7）。

表 7-46 外部SPIF复用功能重映射

复用功能	EXT_SPIF_GEN =0 或 EXT_SPIF_EN=0	（EXT_SPIF_GEN =1 或 EXT_SPIF_EN =1） EXT_SPIF_GRMP[2:0]=000	（EXT_SPIF_GEN =1 或 EXT_SPIF_EN =1） EXT_SPIF_GRMP[2:0]=001
SPIF_SCK	NA	PB1	
SPIF_CS	NA	PA8	
SPIF_IO0	NA	PA11	PB10
SPIF_IO1	NA	PA12	PB11
SPIF_IO2	NA	PB7	
SPIF_IO3	NA	PB6	

7.4.12 Ethernet复用功能重映射

参见复用重映射和调试 I/O 配置寄存器（AFIO_MAP）或复用重映射和调试 I/O 配置寄存器 8（AFIO_MAP8）。

表 7-47 Ethernet复用功能重映射

复用功能	ETH_REMAP =0 或 ETH_GRMP[1:0] =00	ETH_REMAP =1 或 ETH_GRMP[1:0] =01
RX_DV/CRS_DV	PA7	PD8
RXD0	PC4	PD9
RXD1	PC5	PD10
RXD2	PB0	PD11
RXD3	PB1	PD12

7.4.13 XMC复用功能重映射

参见复用重映射和调试 I/O 配置寄存器 7（AFIO_MAP7）。

表 7-48 XMC复用功能重映射

复用功能	**XMC_GRMP[2:0]** =000	**XMC_GRMP[2:0]** =001	**XMC_GRMP[2:0]** =010

XMC_NWE	PD5	PD2	PC2
XMC_D0	PD14	PB14	
XMC_D1	PD15	PC6	
XMC_D2	PD0	PC11	
XMC_D3	PD1	PC12	
XMC_D4	PE7	PA2	
XMC_D5	PE8	PA3	
XMC_D6	PE9	PA4	
XMC_D7	PE10	PA5	
XMC_D13	PD8	PB12	
FSMC_NOE	PD4	PC5	

7.5 GPIO与AFIO寄存器

下面列出了 GPIO 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 7-49 GPIO寄存器地址映射和复位值

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000h	GPIOx_CTRLL	CONF7[1：0]		MDE7[1：0]		CONF6[1：0]		MDE6[1：0]		CONF5[1：0]		MDE5[1：0]		CONF4[1：0]		MDE4[1：0]		CONF3[1：0]		MDE3[1：0]		CONF2[1：0]		MDE2[1：0]		CONF1[1：0]		MDE1[1：0]		CONF0[1：0]		MDE0[1：0]	
	复位值	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0
004h	GPIOx_CTRH	CONF15[1：0]		MDE15[1：0]		CONF14[1：0]		MDE14[1：0]		CONF13[1：0]		MDE13[1：0]		CONF12[1：0]		MDE12[1：0]		CONF11[1：0]		MDE11[1：0]		CONF10[1：0]		MDE10[1：0]		CONF9[1：0]		MDE9[1：0]		CONF8[1：0]		MDE8[1：0]	
	复位值	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0
008h	GPIOx_IPTDT	保留																IPTDT[15：0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
00Ch	GPIOx_OPTDT	保留																OPTDT[15：0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
010h	GPIOx_BSRE	BRE[15：0]																BST[15：0]															
	复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
014h	GPIOx_BRE	保留																BRE[15：0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
018h	GPIOx_LOCK	保留															LOCKK	LOCK[15：0]															
	复位值																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
020h	GPIOx_SRCTR	保留																SRCTR[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
03Ch	GPIOx_HDRV	保留																HDRV[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

下面列出了 AFIO 寄存器映象和复位数值。必须以字（32 位）方式操作这些外设寄存器。

表 7-50 AFIO寄存器地址映射和复位值

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000h	AFIO_EVCTRL	保留																								EVOEN	PORT[2:0]			PIN[3:0]			
	复位值																									0	0	0	0	0	0	0	0
004h	AFIO_MAP	SPI1_REMAP[1]	PTP_PPS_REMAP	TIM2_ITR1_REMAP	SPI3_REMAP	保留	SWJTAG_CONF[2:0]			MII_RMII_SEL	CAN2_REMAP	ETH_REMAP	ADC2_EXTRGREG_REMAP	ADC2_EXTRGINJ_REMAP	ADC1_EXTRGREG_REMAP	ADC1_EXTRGINJ_REMAP	TMR5CH4_INTLRE	PD01_REMAP	CAN_REMAP[1:0]		TMR4_REMAP	TMR3_REMAP[1:0]		TMR2_REMAP[1:0]		TMR1_REMAP[1:0]		USART3_REMAP[1:0]		USART2_REMAP	USART1_REMAP	I2C1_REMAP	SPI1_REMAP[0]
	复位值	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
008H	AFIO_EXTIC1	保留																EXTINT3[3:0]				EXTINT2[3:0]				EXTINT1[3:0]				EXTINT0[3:0]			
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
00CH	AFIO_EXTIC2	保留																EXTINT7[3:0]				EXTINT6[3:0]				EXTINT5[3:0]				EXTINT4[3:0]			
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
010H	AFIO_EXTIC3	保留																EXTINT11[3:0]				EXTINT10[3:0]				EXTINT9[3:0]				EXTINT8[3:0]			
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
014H	AFIO_EXTIC4	保留																EXTINT15[3:0]				EXTINT14[3:0]				EXTINT13[3:0]				EXTINT12[3:0]			
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
01CH	AFIO_MAP2	保留										EXT_SPIF_EN	SDIO2_REMAP			I2C3_REMAP	SPI4_REMAP	保留				XMC_NADV_REMAP	保留			TMR9_REMAP	保留						
	复位值											0	0	0		0	0					0				0							
020H	AFIO_MAP3	保留																												TMR9_GRMP			
	复位值																													0	0	0	0
024H	AFIO_MAP4	保留													TMR5CH4_IRMP	保留		TMR4_GRMP				TMR3_GRMP				TMR2ITR1_IRMP		TMR2_GRMP		TMR1_GRMP			
	复位值														0			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

028H	AFIO_MAP5	SPI4_GRMP				SPI3_GRMP				SPI2_GRMP				SPI1_GRMP				I2C3_GRMP				I2C2_GRMP				I2C1_GRMP				UART5_GRMP			
	复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
02CH	AFIO_MAP6	UART4_GRMP				USART3_GRMP				保留				USART1_GRMP				SDIO1_GRMP				保留				CAN2_GRMP				CAN1_GRMP			
	复位值	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0
030H	AFIO_MAP7	保留				XMC_NADV_GRMP	XMC_GRMP			保留		PD01_GRMP	保留	SWJTAG_GRMP			保留			ADC2_ETR_GRMP	ADC2_ETI_GRMP	保留		ADC1_ETR_GRMP	ADC1_ETI_GRMP	EXT_SPIF_GEN	EXT_SPIF_GRMP						
	复位值					0	0	0	0			0		0	0	0				0	0			0	0	0	0	0	0	0	0		
034H	AFIO_MAP8	UART8_GRMP				UART7_GRMP				USART6_GRMP				PTP_PPS_GRMP	MII_RMII_GRMP	ETH_GRMP	保留																
	复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0																	

注意：对寄存器 AFIO_EVCTRL，AFIO_MAPX 和 AFIO_EXTICX 进行读写操作前，应当首先打开 AFIO 的时钟。参考第 3.3.7 节 APB2 外设时钟使能寄存器（RCC_APB2EN）

7.5.1 端口配置低寄存器（GPIOx_CTRLL）（x=A..E）

偏移地址：0x00

复位值：0x44444444

31 30	29 28	27 26	25 24	23 22	21 20	19 18	17 16
CONF7[1:0]	MDE7[1:0]	CONF6[1:0]	MDE6[1:0]	CONF5[1:0]	MDE5[1:0]	CONF4[1:0]	MDE4[1:0]
rw	rw	rw	rw	rw	rw	rw	rw
15 14	**13 12**	**11 10**	**9 8**	**7 6**	**5 4**	**3 2**	**1 0**
CONF3[1:0]	MDE3[1:0]	CONF2[1:0]	MDE2[1:0]	CONF1[1:0]	MDE1[1:0]	CONF0[1:0]	MDE0[1:0]
rw	rw	rw	rw	rw	rw	rw	rw

位 31:30 27:26 23:22 19:18 15:14 11:10 7:6 3:2	**CONFy[1:0]**:端口 x 配置位（y=0…7）（Portx configuration bits） 软件通过这些位配置相应的 I/O 端口，请参考表 7-1 端口位配置表。 在输入模式（MDE[1:0]=00）： 00:模拟输入模式 01:浮空输入模式（复位后的状态） 10:上拉/下拉输入模式 11:保留 在输出模式（MDE[1:0]>00）： 00:通用推挽输出模式 01:通用开漏输出模式 10:复用功能推挽输出模式 11:复用功能开漏输出模式
位 29:28 25:24 21:20 17:16 13:12 9:8,5:4 1:0	**MDEy[1:0]**:端口 x 的模式位（y=0…7）（Portx Mode bits） 软件通过这些位配置相应的 I/O 端口，请参考表 7-1 端口位配置表。 00:输入模式（复位后的状态） 01: 输出模式，较大电流推动/吸入能力 10: 输出模式，适中电流推动/吸入能力 11: 输出模式，适中电流推动/吸入能力

注意：有些端口寄存器复位值不同，比如 PA 有些引脚默认是 JTAG/SWD 有上拉输入引脚。

7.5.2 端口配置高寄存器（GPIOx_CTRLH）（A..E）

偏移地址:0x04

复位值:0x44444444

31 30	29 28	27 26	25 24	23 22	21 20	19 18	17 16
CONF15[1:0]	MDE15[1:0]	CONF14[1:0]	MDE14[1:0]	CONF13[1:0]	MDE13[1:0]	CONF12[1:0]	MDE12[1:0]
rw	rw	rw	rw	rw	rw	rw	rw
15 14	**13 12**	**11 10**	**9 8**	**7 6**	**5 4**	**3 2**	**1 0**
CONF11[1:0]	MDE11[1:0]	CONF10[1:0]	MDE10[1:0]	CONF9[1:0]	MDE9[1:0]	CONF8[1:0]	MDE8[1:0]
rw	rw	rw	rw	rw	rw	rw	rw

位 31:30 27:26 23:22 19:18 15:14 11:10 7:6 3:2	**CONFy[1:0]**：端口 x 配置位（y=8…15）（Portx configuration bits） 软件通过这些位配置相应的 I/O 端口，请参考表 7-1 端口位配置表。 在输入模式（MDE[1:0]=00）： 00:模拟输入模式 01:浮空输入模式（复位后的状态） 10:上拉/下拉输入模式 11:保留 在输出模式（MDE[1:0]>00）： 00:通用推挽输出模式 01:通用开漏输出模式 10:复用功能推挽输出模式 11:复用功能开漏输出模式
位 29:28 25:24 21:20 17:16 13:12 9:8,5:4 1:0	**MDEy[1:0]**：端口 x 的模式位（y=8…15）（Portx Mode bits） 软件通过这些位配置相应的 I/O 端口，请参考表 7-1 端口位配置表。 00:输入模式（复位后的状态） 01: 输出模式，较大电流推动/吸入能力 10: 输出模式，适中电流推动/吸入能力 11: 输出模式，适中电流推动/吸入能力

注意：有些端口寄存器复位值不同，比如 PB 有些引脚默认是 JTAG/SWD 有上拉输入引脚。

7.5.3 端口输入数据寄存器（GPIOx_IPTDT）（x=A..E）

地址偏移:0x08

复位值:0x0000XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															
res															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IPTDT[15]	IPTDT[14]	IPTDT[13]	IPTDT[12]	IPTDT[11]	IPTDT[10]	IPTDT[9]	IPTDT[8]	IPTDT[7]	IPTDT[6]	IPTDT[5]	IPTDT[4]	IPTDT[3]	IPTDT[2]	IPTDT[1]	IPTDT[0]
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

位 31:16	保留，始终读为 0。
位 15:0	**IPTDTy[15:0]**:端口输入数据（y=0…15）（Port input data） 这些位为只读并只能以字（16 位）的形式读出。读出的值为对应 I/O 口的状态。

7.5.4 端口输出数据寄存器（GPIOx_OPTDT）（x=A..E）

地址偏移:0x0C

复位值:0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															
res															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OPTDT[15]	OPTDT[14]	OPTDT[13]	OPTDT[12]	OPTDT[11]	OPTDT[10]	OPTDT[9]	OPTDT[8]	OPTDT[7]	OPTDT[6]	OPTDT[5]	OPTDT[4]	OPTDT[3]	OPTDT[2]	OPTDT[1]	OPTDT[0]
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:16	保留，始终读为 0。
位 15:0	**OPTDTy[15:0]**:端口输出数据（y=0…15）（Port output data） 这些位可读可写并只能以字（16 位）的形式操作。 注:对 GPIOx_BSRE（x=A…E），可以分别地对各个 OPTDT 位进行独立的设置/清除。

7.5.5 端口位设置/清除寄存器（GPIOx_BSRE）（x=A..E）

地址偏移:0x10

复位值:0x00000000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BST[15]	BST[14]	BST[13]	BST[12]	BST[11]	BST[10]	BST[9]	BST[8]	BST[7]	BST[6]	BST[5]	BST[4]	BST[3]	BST[2]	BST[1]	BST[0]
wo	wo	wo	wo	wo	wo	wo	wo	wo	wo	wo	wo	wo	wo	wo	wo

位 31:16	**BREy**:清除端口 x 的位 y（y=0…15）（Portx Reset bity） 这些位只能写入并只能以字（16 位）的形式操作。 0:对对应的 OPTDTy 位不产生影响 1:清除对应的 OPTDTy 位为 0 注:如果同时设置了 BSTy 和 MREy 的对应位，BSTy 位起作用。
位 15:0	**BSTy**:设置端口 x 的位 y（y=0…15）（Portx Set bity） 这些位只能写入并只能以字（16 位）的形式操作。 0:对对应的 OPTDTy 位不产生影响 1:设置对应的 OPTDTy 位为 1

7.5.6 端口位清除寄存器（IOx_BRE）（x=A..E）

地址偏移:0x14

复位值:0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															
res															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BREy[15]	BREy[14]	BREy[13]	BREy[12]	BREy[11]	BREy[10]	BREy[9]	BREy[8]	BREy[7]	BREy[6]	BREy[5]	BREy[4]	BREy[3]	BREy[2]	BREy[1]	BREy[0]
wo	wo	wo	wo	wo	wo	wo	wo	wo	wo	wo	wo	wo	wo	wo	wo

位 31:16	保留。
位 15:0	**BREy**:清除端口 x 的位 y（y=0…15）（Portx Reset bity） 这些位只能写入并只能以字（16 位）的形式操作。 0:对对应的 OPTDTy 位不产生影响 1:清除对应的 OPTDTy 位为 0

7.5.7 端口配置锁定寄存器（GPIOx_LOCK）（x=A..E）

当执行正确的写序列设置了位 16（LOCKK）时，该寄存器用来锁定端口位的配置。位[15:0]用于锁定 GPIO 端口的配置。在规定的写入操作期间，不能改变 LOCK[15:0]。当对相应的端口位执行了 LOCK 序列后，在下次系统复位之前将不能再更改端口位的配置。

每个锁定位锁定控制寄存器（CTRLL,CTRLH）中相应的 4 个位。

地址偏移:0x18

复位值:0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															LOCKK
res															rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LOCK[15]	LOCK[14]	LOCK[13]	LOCK[12]	LOCK[11]	LOCK[10]	LOCK[9]	LOCK[8]	LOCK[7]	LOCK[6]	LOCK[5]	LOCK[4]	LOCK[3]	LOCK[2]	LOCK[1]	LOCK[0]
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:17	保留。
位 16	**LOCKK:**锁键（Lock key） 该位可随时读出，它只可通过锁键写入序列修改。 0:端口配置锁键位未激活 1:端口配置锁键位被激活，下次系统复位前 GPIOx_LOCK 寄存器被锁住。 锁键的写入序列: 写 1->写 0->写 1->读 0->读 1 最后一个读可省略，但可以用来确认锁键已被激活。 注:在操作锁键的写入序列时，不能改变 LOCK[15:0]的值。 操作锁键写入序列中的任何错误将不能激活锁键。
位 15:0	**LOCKy**:端口 x 的锁位 y（y=0…15）（Portx Lock bit） 这些位可读可写但只能在 LOCKK 位为 0 时写入。 0:不锁定端口的配置 1:锁定端口的配置

7.5.8 电压转换速率切换控制寄存器（GPIOx_SRCTR）（x=A..E）

地址偏移:0x20

复位值:0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															
res															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SRCTR[15]	SRCTR[14]	SRCTR[13]	SRCTR[12]	SRCTR[11]	SRCTR[10]	SRCTR[9]	SRCTR[8]	SRCTR[7]	SRCTR[6]	SRCTR[5]	SRCTR[4]	SRCTR[3]	SRCTR[2]	SRCTR[1]	SRCTR[0]
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:16	保留。
位 15:0	**SRCTRy:** 端口 x 电压转换速率切换控制位 y（y=0…15）（Portx Slew Rate bit） 0：适中转换速率 1：增强转换速率

7.5.9 极大电流推动/吸入能力切换控制寄存器（GPIOx_HDRV）（x=A..E）

地址偏移:0x3C

复位值:0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															
res															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HDRV[15]	HDRV[14]	HDRV13]	HDRV[12]	HDRV11]	HDRV10]	HDRV [9]	HDRV [8]	HDRV [7]	HDRV [6]	HDRV [5]	HDRV [4]	HDRV [3]	HDRV [2]	HDRV [1]	HDRV[0]
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:16	保留。
位 15:0	**HDRVy:** 端口 x 极大电流推动/吸入能力切换控制位 y（y=0…15）（Portx Huge Driven bit） 0：GPIO 配置为较大或适中电流推动/吸入能力，由 MDEy[1:0]配置决定 1：GPIO 配置为极大电流推动/吸入能力，忽略 MDEy[1:0]配置

7.5.10 复用事件控制寄存器（AFIO_EVCTRL）

地址偏移:0x00

复位值:0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															
res															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留								EVOEN	PORT[2:0]			PIN[3:0]			
res								rw	rw	rw	rw	rw	rw	rw	rw

位 31:8	保留。
位 7	**EVOEN**:允许事件输出（Event output enable） 该位可由软件读写。当设置该位后，Cortex 的 EVENTOUT 将连接到由 PORT[2:0]和 PIN[3:0]选定的 I/O 口。
位 6:4	**PORT[2:0]**:端口选择（Port selection） 选择用于输出 Cortex 的 EVENTOUT 信号的端口: 000:选择 PA 001:选择 PB 010:选择 PC 011:选择 PD 100:选择 PE
位 3:0	**PIN[3:0]**:引脚选择（x=A…E）（Pin selection） 选择用于输出 Cortex 的 EVENTOUT 信号的引脚: 0000:选择 Px0 0001:选择 Px1 0010:选择 Px2 0011:选择 Px3 0100:选择 Px4 0101:选择 Px5 0110:选择 Px6 0111:选择 Px7 1000:选择 Px8 1001:选择 Px9 1010:选择 Px10 1011:选择 Px11 1100:选择 Px12 1101:选择 Px13 1110:选择 Px14 1111:选择 Px15

7.5.11 复用重映射和调试I/O配置寄存器（AFIO_MAP）

地址偏移:0x04

复位值:0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SPI1_REMAP[1]	PTP_PPS_REMAP	TIM2ITR1_REMAP	SPI3_REMAP	保留	SWJTAG_CONF[2:0]			MII_RMII_SEL	CAN2_REMAP	ETH_REMAP	ADC2_EXTRG_REG_REMAP	ADC2_EXTRG_INJ_REMAP	ADC1_EXTRG_REG_REMAP	ADC1_EXTRG_INJ_REMAP	TMR5CH4_INTLRE
rw	rw	rw	rw	res	rw			rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PD01_REMAP	CAN_REMAP[1:0]		TMR4_REMAP	TMR3_REMAP[1:0]		TMR2_REMAP[1:0]		TMR1_REMAP[1:0]		USART3_REMAP[1:0]		USART2_REMAP	USART1_REMAP	I2C1_REMAP	SPI1_RE MAP[0]
rw	rw		rw	rw		rw		rw		rw		rw	rw	rw	rw

位 31	**SPI1_REMAP[1]**:SPI1 的重映射 具体定义参考位 0 的 SPI1_REMAP[1:0]。
位 30	**PTP_PPS_REMAP**：以太网 PTP PPS 重映射 (Ethernet PTP PPS remapping) 该位可由软件置'1'或置'0'。它允许以太网 MAC 的 PPS_PTS 输出到 PB5 引脚。 0：PTP_PPS 不输出到 PB5 引脚； 1：PTP_PPS 输出到 PB5 引脚。 注：该位只在互联型产品上有效，其它产品中为保留位
位 29	**TIM2ITR1_REMAP**：TIM2 内部触发 1 重映射 (TIM2 internal trigger 1 remapping) 该位可由软件置'1'或置'0'。它控制 TIM2_ITR1 的内部重映射。 0：使用 TMR8_TRGO 作为 TMR2 ITR1 的输入 1：为了校准的需要，在内部连接 TIM2_ITR1 至以太网的 PTP 输出； 注：该位只在互联型产品上有效，其它产品中为保留位。
位 28	**SPI3_REMAP**：SPI3 重映射 (SPI3 remapping) 该位可由软件置'1'或置'0'。它控制 SPI3 的 NSS、SCK、MISO、MOSI 在 GPIO 端口的复用功能。 0：没有重映射(NSS/PA15、SCK/PB3、MISO/PB4、MOSI/PB5)； 1：重映射(NSS/PA4、SCK/PC10、MISO/PC11、MOSI/PC12)。 注：该位只在互联型产品上有效，其它产品中为保留位。
位 27	**保留**
位 26:24	**SWJTAG_CONF[2:0]**:串行线 JTAG 配置（Serial wire JTAG configuration） 这些位只可由软件写（读这些位，将返回未定义的数值），用于配置 SWJ 和跟踪复用功能的 I/O 口。SWJ（串行线 JTAG）支持 JTAG 或 SWD 访问 Cortex 的调试端口。系统复位后的默认状态是启用 SWJ 但没有跟踪功能，这种状态下可以通过 JTMS/JTCK 脚上的特定信号选择 JTAG 或 SYSCLKSEL（串行线）模式。 000:完全 SWJ（JTAG-DP+SW-DP）:复位状态； 001:完全 SWJ（JTAG-DP+SW-DP）但没有 NJTRST； 010:关闭 JTAG-DP，启用 SW-DP； 100:关闭 JTAG-DP，关闭 SW-DP； 其它组合:无作用。
位 23	**MII_RMII_SEL**：MII 或 RMII 选择 (MII or RMII selection) 该位可由软件置'1'或置'0'。它配置内部的以太网 MAC 使用外部的 MII 接口还是 RMII 接口的收发器(PHY)。 0：配置以太网的 MAC 使用外部 MII 接口的收发器(PHY)； 1：配置以太网的 MAC 使用外部 RMII 接口的收发器(PHY)。 注：该位只在互联型产品上有效，其它产品中为保留位。

位 22	**CAN2_REMAP**：CAN2 的重映射 (CAN2 I/O remapping) 该位可由软件置'1'或置'0'。它控制 CAN2_TX 和 CAN2_RX 引脚的配置。 0：没有重映射(CAN2_RX/PB12、CAN2_TX/PB13) 1：重映射(CAN2_RX/PB5、CAN2_TX/PB6) 注：该位只在互联型产品上有效，其它产品中为保留位。
位 21	**ETH_REMAP**：以太网 MAC 的引脚配置 (Ethernet MAC I/O remapping) 该位可由软件置'1'或置'0'。它控制以太网 MAC 至外部收发器(PHY)的连接。 0：没有重映射(RX_DV/CRS_DV/PA7、RXD0/PC4、RXD1/PC5、RXD2/PB0、RXD3/PB1) 1：重映射(RX_DV/CRS_DV/PD8、RXD0/PD9、RXD1/PD10、RXD2/PD11、RXD3/PD12)。
位 20	**ADC2_EXTRGREG_REMAP**:ADC2 规则转换外部触发重映射（ADC2 external trigger regular conversion remapping） 该位可由软件置'1'或置'0'。它控制与 ADC2 规则转换外部触发相连的触发输入。 当该位置'0'时，ADC2 规则转换外部触发与 EXTINT11 相连； 当该位置'1'时，ADC2 规则转换外部触发与 TMR8_TRGO 相连。
位 19	**ADC2_EXTRGINJ_REMAP**:ADC2 注入转换外部触发重映射（ADC2 external trigger injected conversion remapping） 该位可由软件置'1'或置'0'。它控制与 ADC2 注入转换外部触发相连的触发输入。 当该位置'0'时，ADC2 注入转换外部触发与 EXTINT15 相连； 当该位置'1'时，ADC2 注入转换外部触发与 TMR8 通道 4 相连。
位 18	**ADC1_EXTRGREG_REMAP**:ADC1 规则转换外部触发重映射(ADC1 external trigger regular conversion remapping） 该位可由软件置'1'或置'0'。它控制与 ADC1 规则转换外部触发相连的触发输入。 当该位置'0'时，ADC1 规则转换外部触发与 EXTINT11 相连； 当该位置'1'时，ADC1 规则转换外部触发与 TMR8_TRGO 相连。
位 17	**ADC1_EXTRGINJ_REMAP**:ADC1 注入转换外部触发重映射（ADC1 External trigger injected conversion remapping） 该位可由软件置'1'或置'0'。它控制与 ADC1 注入转换外部触发相连的触发输入。当该位置'0'时，ADC1 注入转换外部触发与 EXTINT15 相连； 当该位置'1'时，ADC1 注入转换外部触发与 TMR8 通道 4 相连。
位 16	**TMR5CH4_INTLRE**:TMR5 通道 4 内部重映射（TMR5 channel4 internal remap） 该位可由软件置'1'或置'0'。它控制 TMR5 通道 4 内部映射。 当该位置'0'时，TMR5_CH4 与 PA3 相连； 当该位置'1'时，LSI 内部振荡器与 TMR5_CH4 相连，目的是对 LSI 进行校准。
位 15	**PD01_REMAP**: 端口 D0/端口 D1 映射到 OSC_IN/OSC_OUT（PortD0/PortD1 mappingon OSC_IN/OSC_OUT） 该位可由软件置'1'或置'0'。它控制 PD0 和 PD1 的 GPIO 功能映射。当不使用主振荡器 HSE 时（系统运行于内部的 48MHz 阻容振荡器），PD0 和 PD1 可以映射到 OSC_IN 和 OSC_OUT 引脚。此功能只能适用于 48 和 64 引脚的封装. 0:不进行 PD0 和 PD1 的重映射； 1:PD0 映射到 OSC_IN，PD1 映射到 OSC_OUT。
位 14:13	**CAN_REMAP[1:0]:**CAN 复用功能重映射（CAN alternate function remapping） 这些位可由软件置'1'或置'0'，在只有单个 CAN 接口的产品上控制复用功能 CAN_RX 和 CAN_TX 的重映射。 00:CAN_RX 映射到 PA11，CAN_TX 映射到 PA12； 01:未用组合； 10:CAN_RX 映射到 PB8，CAN_TX 映射到 PB9； 11: CAN_RX 映射到 PD0，CAN_TX 映射到 PD1；
位 12	**TMR4_REMAP:**定时器 4 的重映射（TMR4 remapping） 该位可由软件置'1'或置'0'，控制将 TMR4 的通道 1-4 映射到 GPIO 端口上。 0:没有重映射（TMR4_CH1/PB6，TMR4_CH2/PB7，TMR4_CH3/PB8，TMR4_CH4/PB9）； 1:完全映射（TMR4_CH1/PD12，TMR4_CH2/PD13，TMR4_CH3/PD14，TMR4_CH4/PD15）。

位 11:10	**TMR3_REMAP[1:0]:**定时器 3 的重映射（TMR3 remapping） 这些位可由软件置'1'或置'0'，控制定时器 3 的通道 1 至 4 在 GPIO 端口的映射。 00:没有重映射（CH1/PA6，CH2/PA7，CH3/PB0，CH4/PB1）； 01:未用组合； 10:部分映射（CH1/PB4，CH2/PB5，CH3/PB0，CH4/PB1）； 11:完全映射（CH1/PC6，CH2/PC7，CH3/PC8，CH4/PC9）。 注:重映射不影响在 PD2 上的 TMR3_ETR。
位 9:8	**TMR2_REMAP[1:0]:**定时器 2 的重映射（TMR2 remapping） 这些位可由软件置'1'或置'0'，控制定时器 2 的通道 1 至 4 和外部触发（ETR）在 GPIO 端口的映射。 00:没有重映射（CH1/ETR/PA0，CH2/PA1，CH3/PA2，CH4/PA3）； 01:部分映射（CH1/ETR/PA15，CH2/PB3，CH3/PA2，CH4/PA3）； 10:部分映射（CH1/ETR/PA0，CH2/PA1，CH3/PB10，CH4/PB11）； 11:完全映射（CH1/ETR/PA15，CH2/PB3，CH3/PB10，CH4/PB11）。
位 7:6	**TMR1_REMAP[1:0]:**定时器 1 的重映射（TMR1 remapping） 这些位可由软件置'1'或置'0'，控制定时器 1 的通道 1 至 4、1N 至 3N、外部触发（ETR）和刹车输入（BKIN）在 GPIO 端口的映射。 00:没有重映射（ETR/PA12，CH1/PA8，CH2/PA9，CH3/PA10，CH4/PA11，BKIN/PB12，CH1N/PB13，CH2N/PB14，CH3N/PB15）； 01:部分映射（ETR/PA12，CH1/PA8，CH2/PA9，CH3/PA10，CH4/PA11，BKIN/PA6，CH1N/PA7，CH2N/PB0，CH3N/PB1）； 10:未用组合； 11: 完全映像(ETR/PE7，CH1/PE9，CH2/PE11，CH3/PE13，CH4/PE14，BKIN/PE15，CH1N/PE8，CH2N/PE10，CH3N/PE12)
位 5:4	**USART3_REMAP[1:0]:**USART3 的重映射（USART3 remapping） 这些位可由软件置'1'或置'0'，控制 USART3 的 CTS、RTS、CK、TX 和 RX 复用功能在 GPIO 端口的映射。 00:没有重映射（TX/PB10，RX/PB11，CK/PB12，CTS/PB13，RTS/PB14）； 01:部分映射（TX/PC10，RX/PC11，CK/PC12，CTS/PB13，RTS/PB14）； 10:未用组合； 11：完全映像（TX/PD8，RX/PD9，CK/PD10，CTS/PD11，RTS/PD12）
位 3	**USART2_REMAP:**USART2 的重映射（USART2 remapping） 这些位可由软件置'1'或置'0'，控制 USART20 的 CTS、RTS、CK、TX 和 RX 复用功能在 GPIO 端口的映射。 0:没有重映射（CTS/PA0，RTS/PA1，TX/PA2，RX/PA3，CK/PA4）； 1:重映射（CTS/PD3，RTS/PD4，TX/PD5，RX/PD6，CK/PD7）；
位 2	**USART1_REMAP:**USART1 的重映射（USART1 remapping） 该位可由软件置'1'或置'0'，控制 USART1 的 TX 和 RX 复用功能在 GPIO 端口的映射。 0:没有重映射（TX/PA9，RX/PA10）； 1:重映射（TX/PB6，RX/PB7）。
位 1	**I2C1_REMAP[0]:**I2C1 的重映射（I2C1 remapping） 该位可由软件置'1'或置'0'，控制 I2C1 的 SCL 和 SDA 复用功能在 GPIO 端口的映射。 0:没有重映射（SCL/PB6，SDA/PB7）； 1:重映射（SCL/PB8，SDA/PB9）。
位 0	**SPI1_REMAP[0]:**SPI1 的重映射 SPI1_REMAP[1]设置于位 31. SPI1_REMAP[1:0]可由软件置'00'，'01'或置'10'，控制 SPI1 的 NSS、SCK、MISO 和 MOSI 复用功能在 GPIO 端口的映射。 00:没有重映射（NSS/PA4，SCK/PA5，MISO/PA6，MOSI/PA7）。 01:重映射（NSS/PA15，SCK/PB3，MISO/PB4，MOSI/PB5）。 10,11：未用组合

7.5.12 复用外部中断配置寄存器1（AFIO_EXTIC1）

地址偏移:0x08

复位值:0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															
res															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXTINT3[3:0]				EXTINT2[3:0]				EXTINT1[3:0]				EXTINT0[3:0]			
rw				rw				rw				rw			

位 31:16	保留。
位 15:0	**EXTINTx[3:0]**:EXTINTx 配置（x=0…3）（EXTINTx configuration） 这些位可由软件读写，用于选择 EXTINTx 外部中断的输入源。 0000:PA[x]引脚　0100: PE[x]引脚 0001:PB[x]引脚　0101:保留 0010:PC[x]引脚　0110:保留 0011:PD[x]引脚

7.5.13 复用外部中断配置寄存器2（AFIO_EXTIC2）

地址偏移:0x0C

复位值:0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															
res															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXTINT7[3:0]				EXTINT6[3:0]				EXTINT5[3:0]				EXTINT4[3:0]			
rw				rw				rw				rw			

位 31:16	保留。
位 15:0	**EXTINTx[3:0]**:EXTINTx 配置（x=4…7）（EXTINTx configuration） 这些位可由软件读写，用于选择 EXTINTx 外部中断的输入源。 0000:PA[x]引脚　0100: PE[x]引脚 0001:PB[x]引脚　0101: 保留 0010:PC[x]引脚　0110:保留 0011: PD[x]引脚

7.5.14 复用外部中断配置寄存器3（AFIO_EXTIC3）

地址偏移:0x10

复位值:0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															
res															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXTINT11[3:0]				EXTINT10[3:0]				EXTINT9[3:0]				EXTINT8[3:0]			
rw				rw				rw				rw			

位 31:16	保留。
位 15:0	**EXTINTx[3:0]**:EXTINTx 配置（x=8…11）（EXTINTx configuration） 这些位可由软件读写，用于选择 EXTINTx 外部中断的输入源。 0000:PA[x]引脚　0100: PE[x]引脚 0001:PB[x]引脚　0101:保留 0010:PC[x]引脚　0110:保留 0011: PD[x]引脚

7.5.15 复用外部中断配置寄存器4（AFIO_EXTIC4）

地址偏移:0x14

复位值:0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															
res															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXTINT15[3:0]				EXTINT14[3:0]				EXTINT13[3:0]				EXTINT12[3:0]			
rw				rw				rw				rw			

位 31:16	保留。
位 15:0	**EXTINTx[3:0]**:EXTINTx 配置（x=12…15）（EXTINTx configuration） 这些位可由软件读写，用于选择 EXTINTx 外部中断的输入源。 0000:PA[x]引脚　0100: PE[x]引脚 0001:PB[x]引脚　0101:保留 0010:PC[x]引脚　0110:保留 0011: PD[x]引脚

7.5.16 复用重映射和调试I/O配置寄存器2（AFIO_MAP2）

地址偏移:0x1C

复位值:0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留										EXT_SPIF_EN	SDIO2_REMAP[1:0]		I2C3_REMAP	SPI4_REMAP	保留
res										rw	rw		rw	rw	res

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留					XMC_NADV_REMAP	保留				TMR9_REMAP	保留				
res					rw	res				rw	res				

位 31:22	保留。
位 21	**EXT_SPIF_EN:**使能外部 SPIFlash 接口。 该位可由软件置'1'或置'0'。控制是否使用外部 SPIFlash。 请参考 7.4.12-“外部 SPIF 复用功能重映射”
位 20:19	**SDIO2_REMAP[1:0]:**SDIO2 内部重映射（SDIO2 internal remap） 该位可由软件置'1'或置'0'。它控制 SDIO2 内部映射。 请参考 7.4.10SDIO 复用功能重映射

位 18	**I2C3_REMAP:**I2C3 内部重映射（I2C3 internal remap） 该位可由软件置'1'或置'0'。它控制 I2C3 内部映射。 请参考 7.4.8I2C 复用功能重映射
位 17	**SPI4_REMAP:**SPI4 内部重映射（SPI4 internal remap） 该位可由软件置'1'或置'0'。它控制 SPI4 内部映射。 请参考 7.4.9SPI/I2S 复用功能重映射
位 16:11	保留。
位 10	**XMC_NADV_REMAP:**XMCNADV 连接。 该位可由软件置'1'或置'0'。控制是否使用 XMC_NADV 信号。 0:XMC_NADV 连接到 pin。（默认） 1:XMC_NADV 不使用，对应的 pin 可被其他外设使用。
位 9：6	保留。
位 5	**TMR9_REMAP**:TMR9 通道 1/2 内部重映射（TMR9 channel1/2 internal remap） 该位可由软件置'1'或置'0'。它控制 TMR9 通道 1/2 内部映射。 当该位置'0'时，TMR9_CH1 与 PA2 相连,TMR9_CH2 与 PA3 相连； 当该位置'1'时，TMR9_CH1 与 PE5 相连,TMR9_CH2 与 PE6 相连；
位 4:0	保留。

7.5.17 复用重映射和调试I/O配置寄存器3（AFIO_MAP3）

地址偏移:0x20

复位值:0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															
res															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留												TMR9_GRMP			
res												rw			

位 31:4	保留
位 3:0	TMR9_GRMP：TMR9 的重映射（TMR9 remapping） TMR9_GRMP[0] 该位无实体寄存器，直接映射到 AFIO_MAP2 寄存器的 bit5(TMR9_REMAP)，对该位的读写等同于对 AFIO_MAP2 寄存器的 bit5(TMR9_REMAP)的读写。 这些位可由软件置'1'或置'0'，控制复用功能 TMR9_CH1 和 TMR9_CH2 的重映射。 0000：TMR9_CH1 与 PA2 相连,TMR9_CH2 与 PA3 相连； 0001: TMR9_CH1 与 PE5 相连,TMR9_CH2 与 PE6 相连 0010~1111：保留，供以后扩展使用，请勿使用。

7.5.18 复用重映射和调试I/O配置寄存器4（AFIO_MAP4）

地址偏移:0x24

复位值:0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留												TMR5CH4_IRMP	保留		

				res								rw		res	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TMR4_GRMP				TMR3_GRMP				TMR2ITR1_IRMP		TMR2_GRMP		TMR1_GRMP			
rw				rw				rw		rw		rw			

位 31:20	保留，供以后扩展使用，请写全 0。
位 19	**TMR5CH4_IRMP**:TMR5 通道 4 内部重映射（TMR5 channel4 internal remap） 该位可由软件置'1'或置'0'。它控制 TMR5 通道 4 内部映射。当该位置'0'时，TMR5_CH4 与 PA3 相连；当该位置'1'时，LSI 内部振荡器与 TMR5_CH4 相连，目的是对 LSI 进行校准。
位 18:16	保留
位 15:12	**TMR4_GRMP:**TMR4 的重映射（TMR4 remapping） 这些位可由软件置'1'或置'0'， 它控制 TMR4 内部映射。 请参考 7.4.6 定时器复用功能重映射 0000~1111:如未使用该位，请写全 0。
位 11:8	**TMR3_GRMP:**TMR3 的重映射（TMR3 remapping） 这些位可由软件置'1'或置'0'， 它控制 TMR3 内部映射。 请参考 7.4.6 定时器复用功能重映射 0000~1111:如未使用该位，请写全 0。
7:6 位	**TMR2ITR1_IRMP[1:0]：** TIM2 内部触发 1 重映射 (TMR2 internal trigger 1 remapping) 该位可由软件设置。它控制 TMR2_ITR1 的内部重映射。 00：使用 TMR8_TRGO 作为 TMR2 ITR1 的输入 01：保留，请勿使用 10：为了校准的需要，在内部连接以太网的 PTP 输出至 TMR2_ITR1； 11： 使用 USBDEV 的 SOF 作为 TMR2_ITR1 的输入（当 TMR2_ITR1 使用此选项时，会导致 TMR2_GRMP/TMR2_REMAP 功能失效，请注意此使用限制）
位 5:4	**TMR2_GRMP:**TMR2 的重映射（TMR2 remapping） 这些位可由软件置'1'或置'0'， 它控制 TMR2 内部映射。 请参考 7.4.6 定时器复用功能重映射 0000~1111:如未使用该位，请写全 0。
位 3:0	**TMR1_GRMP:**TMR1 的重映射（TMR1 remapping） 这些位可由软件置'1'或置'0'， 它控制 TMR1 内部映射。 请参考 7.4.6 定时器复用功能重映射 0000~1111:如未使用该位，请写全 0。

7.5.19 复用重映射和调试I/O配置寄存器5（AFIO_MAP5）

地址偏移:0x28

复位值:0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SPI4_GRMP				SPI3_GRMP				SPI2_GRMP				SPI1_GRMP			
rw				rw				rw				rw			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
I2C3_GRMP				保留				I2C1_GRMP				USART5_GRMP			
rw				res				rw				rw			

位 31:28	**SPI4_GRMP**:SPI4 内部重映射（SPI4 internal remap） 这些位可由软件置'1'或置'0'， 它控制 SPI4 内部映射。 请参考 7.4.9SPI/I2S 复用功能重映射 0000~1111:如未使用该位，请写全 0。
27：24	**SPI3_GRMP**:SPI3 内部重映射（SPI3 internal remap） 这些位可由软件置'1'或置'0'， 它控制 SPI3 内部映射。 请参考 7.4.9SPI/I2S 复用功能重映射 0000~1111:如未使用该位，请写全 0。
位 23:20	**SPI2_GRMP**:SPI2 内部重映射（SPI3 internal remap） 这些位可由软件置'1'或置'0'， 它控制 SPI2 内部映射。 请参考 7.4.9SPI/I2S 复用功能重映射 0000~1111:如未使用该位，请写全 0。
位 19:16	**SPI1_GRMP:**SPI1 的重映射（SPI1 remapping） 这些位可由软件置'1'或置'0'， 它控制 SPI1 内部映射。 请参考 7.4.9SPI/I2S 复用功能重映射 0000~1111:如未使用该位，请写全 0。
位 15:12	**I2C3_GRMP**:I2C3 内部重映射（I2C3 internal remap） 该位可由软件置'1'或置'0'。它控制 I2C3 内部映射。 0000~0011 请参考 7.4.9SPI/I2S 复用功能重映射 0100~1111:保留，供以后扩展使用，请勿使用。
位 11:8	保留，供以后扩展使用，请写全 0。
位 7:4	**I2C1_GRMP:**I2C1 的重映射（I2C1 remapping） 该位可由软件置'1'或置'0'，控制 I2C1 的 SCL 和 SDA 复用功能在 GPIO 端口的映射。 0000~0011 请参考 7.4.8I2C 复用功能重映射 0100~1111:保留，供以后扩展使用，请勿使用。
位 3:0	**USART5_GRMP:**USART5 的重映射（USART5 remapping） 这些位可由软件置'1'或置'0'， 它控制 USART5 内部映射。 请参考 7.4.7USART/UART 复用功能重映射 0000~1111:如未使用该位，请写全 0。

7.5.20 复用重映射和调试I/O配置寄存器6（AFIO_MAP6）

地址偏移:0x2C

复位值:0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UART4_GRMP				USART3_GRMP				USART2_GRMP				USART1_GRMP			
rw				rw				rw				rw			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SDIO2_GRMP				保留				CAN2_GRMP				CAN1_GRMP			
rw				res				rw				rw			

位 31:28	**UART4_GRMP:**UART4 的重映射（UART4 remapping） 这些位可由软件置'1'或置'0'，控制复用功能 UART4_RX 和 UART_TX 的重映射。 请参考 7.4.7USART/UART 复用功能重映射 0000~1111:如未使用该位，请写全 0。

位 27:24	**USART3_GRMP:**USART3 的重映射（USART3 remapping） 这些位可由软件置'1'或置'0'， 它控制 USART3 内部映射。 请参考 7.4.7USART/UART 复用功能重映射 0000~1111:如未使用该位，请写全 0。
位 23:20	**USART2_GRMP:**USART1 的重映射（USART2 remapping） 这些位可由软件置'1'或置'0'， 它控制 USART2 内部映射。 请参考 7.4.7USART/UART 复用功能重映射 0000~1111:如未使用该位，请写全 0。
位 19:16	**USART1_GRMP:**USART1 的重映射（USART1 remapping） 这些位可由软件置'1'或置'0'， 它控制 USART1 内部映射。 请参考 7.4.7USART/UART 复用功能重映射 0000~1111:如未使用该位，请写全 0。
位 15:12	**SDIO2_GRMP**:SDIO2 内部重映射（SDIO2 internal remap） 该位可由软件置'1'或置'0'。它控制 SDIO2 内部映射。 0000~0111 请参考 7.4.10SDIO 复用功能重映射 1000~1111:保留，供以后扩展使用，请勿使用。
位 11:8	**保留**
位 7:4	**CAN2_GRMP:**CAN2 的重映射（CAN2 remapping） 这些位可由软件置'1'或置'0'，控制复用功能 CAN2_RX 和 CAN2_TX 的重映射。 请参考 7.4.3CAN 复用功能重映射 0000~1111:如未使用该位，请写全 0。
位 3:0	**CAN1_GRMP:**CAN1 的重映射（CAN1 remapping） 这些位可由软件置'1'或置'0'， 它控制 CAN1 内部映射。 请参考 7.4.3CAN 复用功能重映射 0000~1111:如未使用该位，请写全 0。

7.5.21 复用重映射和调试I/O配置寄存器7（AFIO_MAP7）

地址偏移:0x30

复位值:0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留				XMC_NADV_GRMP	XMC_GRMP			保留			PD01_GRMP	保留	SWJTAG_GRMP		
res				rw	rw			res			rw	res	rw		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留						ADC2_ETR_GRMP	ADC2_ETI_GRMP	保留		ADC1_ETR_GRMP	ADC1_ETI_GRMP	EXT_SPIF_GEN	EXT_SPIF_GRMP		
res						rw	rw	res		rw	rw	rw	rw		

位 31:28	保留，供以后扩展使用，请写全 0。
位 27	**XMC_NADV_GRMP:**XMCNADV 连接。 该位可由软件置'1'或置'0'。控制是否使用 XMC_NADV 信号。 0:XMC_NADV 连接到 pin。（默认） 1:XMC_NADV 不使用，对应的 pin 可被其他外设使用。

位 26:24	**XMC_GRMP**:XMC 内部重映射（XMC remapping） 这些位可由软件置'1'或置'0'， 它控制 XMC 内部映射。 请参考 7.4.13 XMC 复用功能重映射 000~111:如未使用该位，请写全 0。
位 23:21	保留，供以后扩展使用，请写全 0。
位 20	**PD01_GRMP**:端口 D0/端口 D1 映射到 OSC_IN/OSC_OUT（PortD0/PortD1 mappingon OSC_IN/OSC_OUT） 该位可由软件置'1'或置'0'。它控制 PD0 和 PD1 的 GPIO 功能映射。当不使用主振荡器 HSE 时（系统运行于内部的 48MHz 阻容振荡器），PD0 和 PD1 可以映射到 OSC_IN 和 OSC_OUT 引脚。 0:不进行 PD0 和 PD1 的重映射； 1:PD0 映射到 OSC_IN，PD1 映射到 OSC_OUT。
位 19	保留，供以后扩展使用，请写全 0。
位 18:16	**SWJTAG_GRMP:** 串行线 JTAG 配置（Serial wire JTAG remapping） 这些位只可由软件写（读这些位，将返回未定义的数值），用于配置 SWJ 和跟踪复用功能的 I/O 口。SWJ（串行线 JTAG）支持 JTAG 或 SWD 访问 Cortex 的调试端口。系统复位后的默认状态是启用 SWJ 但没有跟踪功能，这种状态下可以通过 JTMS/JTCK 脚上的特定信号选择 JTAG 或 SYSCLKSEL（串行线）模式。 000:完全 SWJ（JTAG-DP+SW-DP）:复位状态； 001:完全 SWJ（JTAG-DP+SW-DP）但没有 NJTRST； 010:关闭 JTAG-DP，启用 SW-DP； 100:关闭 JTAG-DP，关闭 SW-DP； 其它组合:无作用。
位 15:10	保留，供以后扩展使用，请写全 0。
位 9	**ADC2_ETR_GRMP**:ADC2 规则转换外部触发重映射（ADC2 external trigger regular conversion remapping） 该位可由软件置'1'或置'0'。它控制与 ADC2 规则转换外部触发相连的触发输入。当该位置'0'时，ADC2 规则转换外部触发与 EXTINT11 相连；当该位置'1'时，ADC2 规则转换外部触发与 TMR8_TRGO 相连。
位 8	**ADC2_ETI_GRMP**:ADC2 注入转换外部触发重映射（ADC2 external trigger injected conversion remapping） 该位可由软件置'1'或置'0'。它控制与 ADC2 注入转换外部触发相连的触发输入。当该位置'0'时，ADC2 注入转换外部触发与 EXTINT15 相连；当该位置'1'时，ADC2 注入转换外部触发与 TMR8 通道 4 相连。
位 7:6	保留，供以后扩展使用，请写全 0。
位 5	**ADC1_ETR_GRMP**:ADC1 规则转换外部触发重映射（ADC1 external trigger regular conversion remapping） 该位可由软件置'1'或置'0'。它控制与 ADC1 规则转换外部触发相连的触发输入。当该位置'0'时，ADC1 规则转换外部触发与 EXTINT11 相连；当该位置'1'时，ADC1 规则转换外部触发与 TMR8_TRGO 相连。
位 4	**ADC1_ETI_GRMP**:ADC1 注入转换外部触发重映射（ADC1 External trigger injected conversion remapping） 该位可由软件置'1'或置'0'。它控制与 ADC1 注入转换外部触发相连的触发输入。当该位置'0'时，ADC1 注入转换外部触发与 EXTINT15 相连；当该位置'1'时，ADC1 注入转换外部触发与 TMR8 通道 4 相连。
位 3	**EXT_SPIF_GEN:**使能外部 SPI Flash 接口。 该位可由软件置'1'或置'0'。控制是否使用外部 SPI Flash。 和 EXT_SPIF_EN 是完全映射关系。 请参考 7.4.12-"外部 SPIF 复用功能重映射"

位 2:0	**EXT_SPIF_GRMP**: 该位可由软件置'1'或置'0'。它控制外部 SPIF 接口的重映射。 0000~0001:请参考 1.4.12-"外部 SPIF 复用功能重映射" 0010~1111:保留，供以后扩展使用，请勿使用。

7.5.22 复用重映射和调试I/O配置寄存器8（AFIO_MAP8）

地址偏移:0x34

复位值:0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UART8_GRMP				UART7_GRMP				USART6_GRMP				PTP_PPS_GRMP	MII_RMII_GRMP	ETH_GRMP	
rw				rw				rw				rw	rw	rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留															
res															

位 31:28	**UART8_GRMP:**USART8 的重映射（UART8remapping） 这些位可由软件置'1'或置'0'， 它控制 UART8 内部映射。 请参考 7.4.7 USART/UART 复用功能重映射 0000~1111:如未使用该位，请写全 0。
位 27：24	**UART7_GRMP:**USART7 的重映射（UART7 remapping） 这些位可由软件置'1'或置'0'， 它控制 UART7 内部映射。 请参考 7.4.7USART/UART 复用功能重映射 0000~1111:如未使用该位，请写全 0。
位 23：20	**USART6_GRMP:**USART6 的重映射（USART6 remapping） 这些位可由软件置'1'或置'0'， 它控制 USART6 内部映射。 请参考 7.4.7USART/UART 复用功能重映射 0000~1111:如未使用该位，请写全 0。
位 19	PTP_PPS_GRMP：以太网 PTP PPS 重映射 (Ethernet PTP PPS remapping) 该位可由软件置'1'或置'0'。它允许以太网 MAC 的 PPS_PTS 输出到 PB5 引脚。 0：PTP_PPS 不输出到 PB5 引脚； 1：PTP_PPS 输出到 PB5 引脚。 注：该位只在互联型产品上有效，其它产品中为保留位
位 18	MII_RMII_SEL：MII 或 RMII 选择 (MII or RMII selection) 该位可由软件置'1'或置'0'。它配置内部的以太网 MAC 使用外部的 MII 接口还是 RMII 接口的收发器(PHY)。 0：配置以太网的 MAC 使用外部 MII 接口的收发器(PHY)； 1：配置以太网的 MAC 使用外部 RMII 接口的收发器(PHY)。 注：该位只在互联型产品上有效，其它产品中为保留位。
位 17:16	ETH_GRMP：以太网 MAC 的引脚配置 (Ethernet MAC I/O remapping) 该位可由软件置'1'或置'0'。它控制以太网 MAC 至外部收发器(PHY)的连接。 0：没有重映射(RX_DV-CRS_DV/PA7、RXD0/PC4、RXD1/PC5、RXD2/PB0、RXD3/PB1) 1：重映射(RX_DV-CRS_DV/PD8、RXD0/PD9、RXD1/PD10、RXD2/PD11、RXD3/PD12)。 注：该位只在互联型产品上有效，其它产品中为保留位。转换外部触发与 TMR8 通道 4 相连。

位 15:0	保留，供以后扩展使用，请写全 0。

8 中断和事件

8.1 嵌套向量中断控制器

特性

- 81 个可屏蔽中断通道（不包含 16 个 Cortex™-M4F 的中断线）；
- 16 个可编程的优先等级（使用了 4 位中断优先级）；
- 低延迟的异常和中断处理；
- 电源管理控制；
- 系统控制寄存器的实现；

嵌套向量中断控制器（NVIC）和处理器核的接口紧密相连，可以实现低延迟的中断处理和高效地处理晚到的中断。

嵌套向量中断控制器管理着包括内核异常等中断。

8.1.1 系统嘀嗒（SysCNTRick）校准值寄存器

系统嘀嗒校准值固定为 9000，当系统嘀嗒时钟设定为 9MHz，产生 1ms 时间基准。

8.1.2 中断和异常向量

下面列出了 AT32F403A/407 产品的向量表。

表 8-1 AT32F403A/407产品的向量表

位置	优先 级	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	不可屏蔽中断 RCC 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（Debug Monitor）	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	WWDG	窗口定时器中断	0x0000_0040
1	8	可设置	PVD	连到 EXTI 的电源电压检测（PVD）中断	0x0000_0044
2	9	可设置	TAMPER	侵入检测中断	0x0000_0048

3	10	可设置	RTC	实时时钟（RTC）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	RCC	复位和时钟控制（RCC）中断	0x0000_0054
6	13	可设置	EXTINT0	EXTI 线 0 中断	0x0000_0058
7	14	可设置	EXTINT1	EXTI 线 1 中断	0x0000_005C
8	15	可设置	EXTINT2	EXTI 线 2 中断	0x0000_0060
9	16	可设置	EXTINT3	EXTI 线 3 中断	0x0000_0064
10	17	可设置	EXTINT4	EXTI 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_2	ADC1 和 ADC2 的全局中断	0x0000_0088
19	26	可设置	USB_HP_CAN1_TX	USB 高优先级或 CAN1 发送中断	0x0000_008C
20	27	可设置	USB_LP_CAN1_RX0	USB 低优先级或 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	CAN_SCE	CAN SCE 中断	0x0000_0098
23	30	可设置	EXTINT9_5	EXTI 线[9：5]中断	0x0000_009C
24	31	可设置	TMR1_BRK_TMR9	TMR1 刹车中断和 TMR9 全局中断	0x0000_00A0
25	32	可设置	TMR1_OV_TMR10	TMR1 更新中断和 TMR10 全局中断	0x0000_00A4
26	33	可设置	TMR1_TRG_COM_TMR11	TMR1 触发和通信中断和 TMR11 全局中断	0x0000_00A8
27	34	可设置	TMR1_CC	TMR1 捕获比较中断	0x0000_00AC
28	35	可设置	TMR2	TMR2 全局中断	0x0000_00B0
29	36	可设置	TMR3	TMR3 全局中断	0x0000_00B4
30	37	可设置	TMR4	TMR4 全局中断	0x0000_00B8
31	38	可设置	I2C1_EV	I^2C1 事件中断	0x0000_00BC
32	39	可设置	I2C1_ER	I^2C1 错误中断	0x0000_00C0
33	40	可设置	I2C2_EV	I^2C2 事件中断	0x0000_00C4
34	41	可设置	I2C2_ER	I^2C2 错误中断	0x0000_00C8
35	42	可设置	SPI1	SPI1 全局中断	0x0000_00CC
36	43	可设置	SPI2_I2S2EXT	SPI2 全局中断和 I2S2EXT 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	EXTINT15_10	EXTI 线[15：10]中断	0x0000_00E0
41	48	可设置	RTCAlarm	连到 EXTI 的 RTC 闹钟中断	0x0000_00E4
42	49	可设置	USB 唤醒	连到 EXTI 的从 USB 待机唤醒中断	0x0000_00E8
43	50	可设置	TMR8_BRK_TMR12	TMR8 刹车中断和 TMR12 全局中断	0x0000_00EC
44	51	可设置	TMR8_OV_TMR13	TMR8 更新中断和 TMR13 全局中断	0x0000_00F0
45	52	可设置	TMR8_TRG_COM_TMR 14	TMR8 触发和通信中断和 TMR14 全局中断	0x0000_00F4
46	53	可设置	TMR8_CC	TMR8 捕获比较中断	0x0000_00F8
47	54	可设置	ADC3	ADC3 全局中断	0x0000_00FC
48	55	可设置	XMC	XMC 全局中断	0x0000_0100
49	56	可设置	SDIO	SDIO 全局中断	0x0000_0104
50	57	可设置	TMR5	TMR5 全局中断	0x0000_0108
51	58	可设置	SPI3_I2S3EXT	SPI3 全局中断和 I2S3EXT 全局中断	0x0000_010C
52	59	可设置	UART4	UART4 全局中断	0x0000_0110
53	60	可设置	UART5	UART5 全局中断	0x0000_0114
54	61	可设置	TMR6	TMR6 全局中断	0x0000_0118
55	62	可设置	TMR7	TMR7 全局中断	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	可设置	SDIO2	SDIO2 全局中断	0x0000_0130
61	68	可设置	I2C3_EV	I2C3 事件中断	0x0000_0134
62	69	可设置	I2C3_ER	I2C3 错误中断	0x0000_0138
63	70	可设置	SPI4	SPI4 全局中断	0x0000_013C
64	71	-	-	保留	0x0000_0140
65	72	-	-	保留	0x0000_0144
66	73	-	-	保留	0x0000_0148
67	74	-	-	保留	0x0000_014C
68	75	可设置	CAN2_TX	CAN2 发送中断	0x0000_0150
69	76	可设置	CAN2_RX0	CAN2 接收 0 中断	0x0000_0154
70	77	可设置	CAN2_RX1	CAN2 接收 1 中断	0x0000_0158
71	78	可设置	CAN2_SCE	CAN2 SCE 中断	0x0000_015C
72	79	可设置	ACC	ACC 中断	0x0000_0160

73	80	可设置	USB_HP[1]	USB 高优先级	0x0000_0164
74	81	可设置	USB_LP[1]	USB 低优先级	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
76	83	可设置	USART6	UART6 全局中断	0x0000_0170
77	84	可设置	UART7	UART7 全局中断	0x0000_0174
78	85	可设置	UART8	UART8 全局中断	0x0000_0178
79	86	可设置	ETH[2]	以太网全局中断	0x0000_017C
80	87	可设置	ETH_WKUP[2]	连接 EXTI 的以太网唤醒中断	0x0000_0180

注意 1：USBDEV 模块中断支持重新映射，由 RCC_INTCTRL 的 USB_INT_CTRL 位控制，当 USB_INT_CTRL=0 时，USB_HP 使用 19 号中断，USB_LP 使用 20 号中断；USB_INT_CTRL=1 时，USB_HP 使用 73 号中断，USB_LP 使用 74 号中断。

2: AT32F407 支持 ETH 和 ETH_WKUP 中断，AT32F403A 不支持这两个中断。

8.2 外部中断/事件控制器（EXTI）

外部中断/事件控制器由 20 个产生事件/中断请求的边沿检测器组成。每个输入线可以独立地配置输入类型（事件或中断）和对应的触发事件（上升沿或下降沿或者双边沿都触发）。每个输入线都可以独立地被屏蔽。挂起寄存器保持着状态线的中断请求。

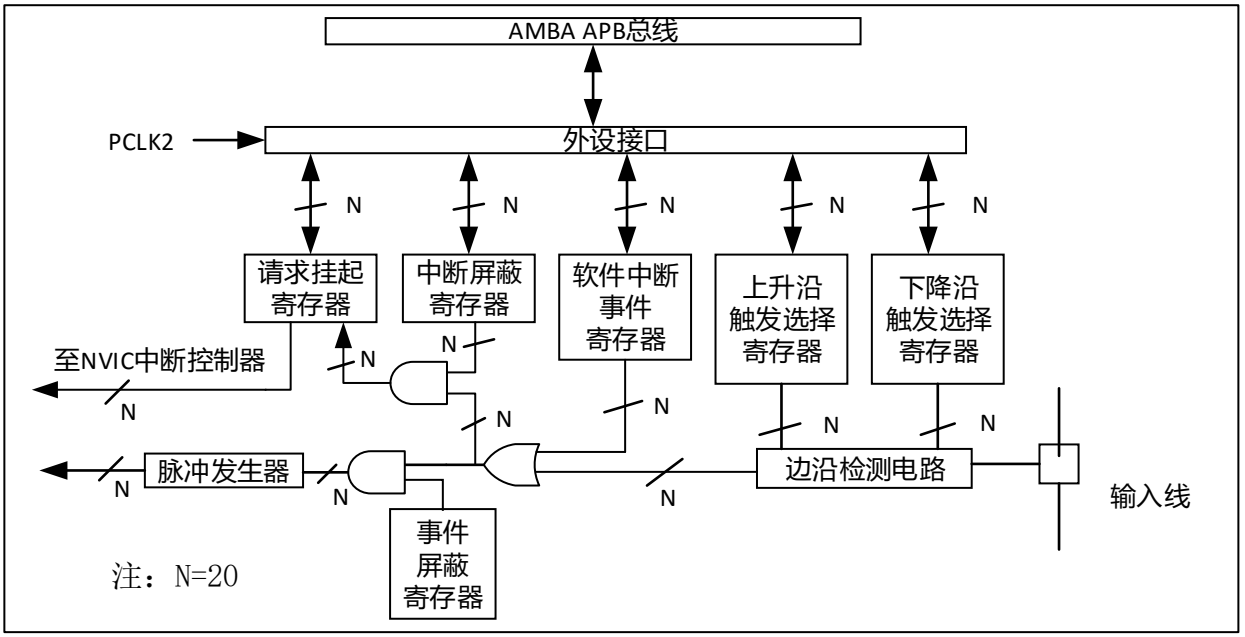
8.2.1 主要特性

EXTI 控制器的主要特性如下：

- 每个中断/事件都有独立的触发和屏蔽
- 每个中断线都有专用的状态位
- 支持多达 20 个软件的中断/事件请求
- 检测脉冲宽度低于 APB2 时钟宽度的外部信号。参见数据手册中电气特性部分的相关参数。

8.2.2 框图

图 8-1 外部中断/事件控制器框图



8.2.3 唤醒事件管理

AT32F403A/407 可以处理外部或内部事件来唤醒内核（WFE）。唤醒事件可以通过下述配置产生：

- 在外设的控制寄存器使能一个中断，但不在 NVIC 中使能，同时在 Cortex™-M4F 的系统控制寄存器中使能 SEVONPEND 位。当 CPU 从 WFE 恢复后，需要清除相应外设的中断挂起位和外设 NVIC 中断通道挂起位（在 NVIC 中断清除挂起寄存器中）。
- 配置一个外部或内部 EXTI 线为事件模式，当 CPU 从 WFE 恢复后，因为对应事件线的挂起位没有被置位，不必清除相应外设的中断挂起位或 NVIC 中断通道挂起位。

在 AT32F407 产品中，以太网唤醒事件同样具有 WFE 唤醒功能。

使用外部 I/O 端口作为唤醒事件，请参见 8.2.4 节功能说明。

8.2.4 功能说明

要产生中断，必须先配置好并使能中断线。根据需要的边沿检测设置 2 个触发寄存器，同时在中断屏蔽寄存器的相应位写'1'允许中断请求。当外部中断线上发生了期待的边沿时，将产生一个中断请求，对应的挂起位也随之被置'1'。在挂起寄存器的对应位写'1'，将清除该中断请求。

如果需要产生事件，必须先配置好并使能事件线。根据需要的边沿检测通过设置 2 个触发寄存器，同时在事件屏蔽寄存器的相应位写'1'允许事件请求。当事件线上发生了需要的边沿时，将产生一个事件请求

脉冲，对应的挂起位不被置'1'。

通过在软件中断/事件寄存器写'1'，也可以通过软件产生中断/事件请求。

硬件中断选择

通过下面的过程来配置 20 个线路做为中断源：

- 配置20个中断线的屏蔽位（EXTI_INTEN）
- 配置所选中断线的触发选择位（EXTI_RTRSEL和EXTI_FTRSEL）；
- 配置对应到外部中断控制器（EXTI）的NVIC中断通道的使能和屏蔽位，使得20个中断线中的请求可以被正确地响应。

硬件事件选择

通过下面的过程，可以配置 20 个线路为事件源

- 配置20个事件线的屏蔽位（EXTI_EVTEN）
- 配置事件线的触发选择位（EXTI_RTRSEL和EXTI_FTRSEL）

软件中断/事件的选择

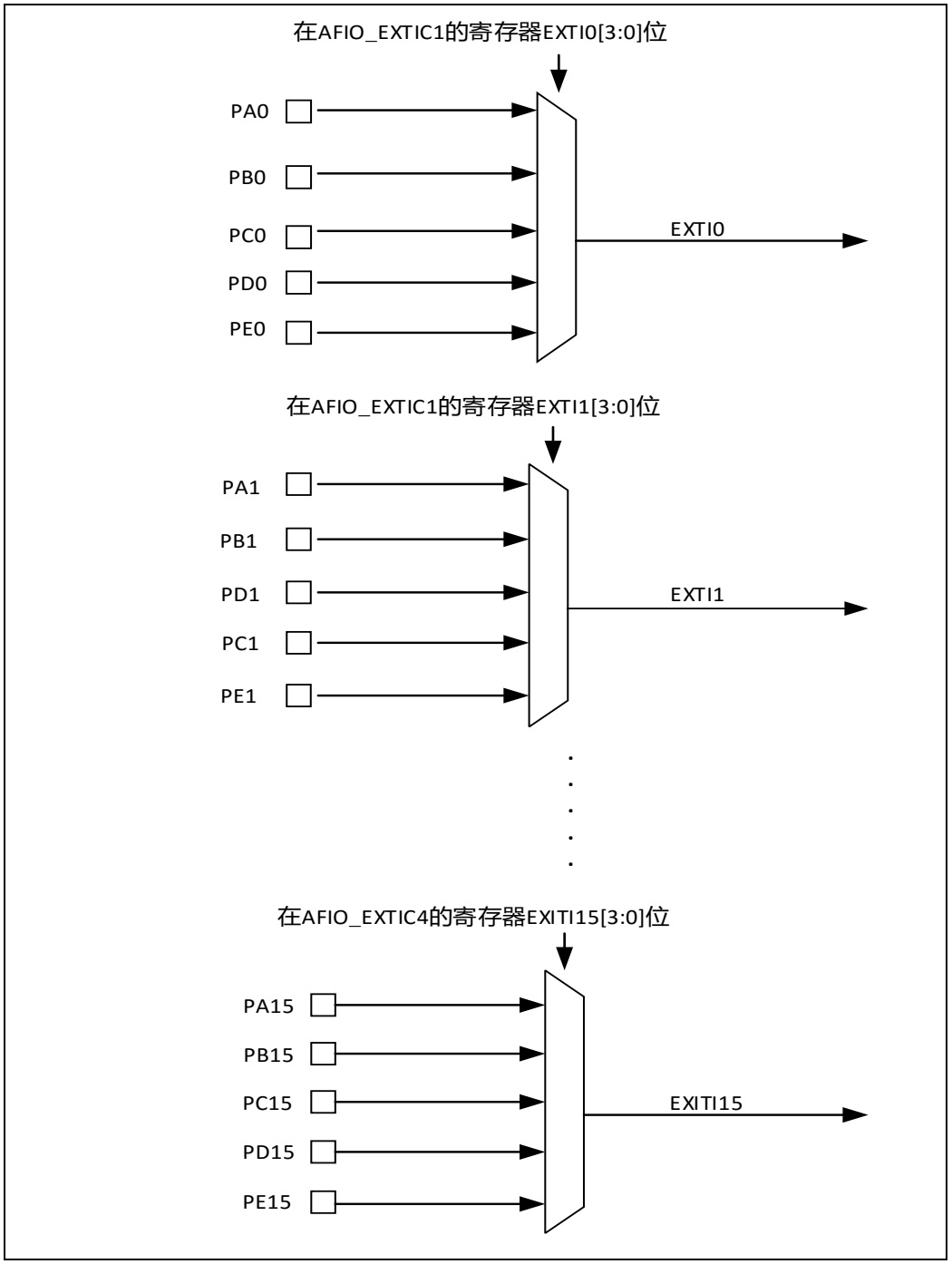
20 个线路可以被配置成软件中断/事件线。下面是产生软件中断的过程：

- 配置20个中断/事件线屏蔽位（EXTI_INTEN, EXTI_EVTEN）
- 设置软件中断寄存器的请求位（EXTI_SWIE）

8.2.5 外部中断/事件线路映像

80 个通用 I/O 端口以下图的方式连接到 16 个外部中断/事件线上：

图 8-2 外部中断通用I/O映像



通过 AFIO_EXTICx 配置 GPIO 线上的外部中断/事件，必须先使能 AFIO 时钟。参见 3.3.7 节。

另外四个 EXTI 线的连接方式如下：

- EXTI 线 16 连接到 PVD 输出
- EXTI 线 17 连接到 RTC 闹钟事件
- EXTI 线 18 连接到 USB 唤醒事件
- EXTI 线 19 连接到以太网唤醒事件（仅适用于 AT32F407，AT32F403A 保留未用）

8.3 EXTI寄存器描述

必须以字（32 位）的方式操作这些外设寄存器。

下表列出了 EXTI 寄存器的映像和复位值。

表 8-2 外部中断/事件控制器寄存器映像和复位值

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	EXTI_INTEN	保留												MR[19：0]																			
	复位值													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	EXTI_EVTEN	保留												MR[19：0]																			
	复位值													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08	EXTI_RTRSEL	保留												TR[19：0]																			
	复位值													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	EXTI_FTRSEL	保留												TR[19：0]																			
	复位值													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	EXTI_SWIE	保留												SWIE[19：0]																			
	复位值													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14	EXTI_PND	保留												PR[19：0]																			
	复位值													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

8.3.1 中断屏蔽寄存器（EXTI_INTEN）

偏移地址：0x00

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留												MR19	MR18	MR17	MR16
res												rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MR15	MR14	MR13	MR12	MR11	MR10	MR9	MR8	MR7	MR6	MR5	MR4	MR3	MR2	MR1	MR0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31：20	保留，必须始终保持为复位状态（0）。
位 19：0	**MRx**：线 x 上的中断屏蔽（Interrupt Mask on line x） 0：屏蔽来自线 x 上的中断请求； 1：开放来自线 x 上的中断请求。 注：位 19 只适用于 AT32F407，对于其它产品为保留位。

8.3.2 事件屏蔽寄存器（EXTI_EVTEN）

偏移地址：0x04

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留												MR19	MR18	MR17	MR16
res												rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MR15	MR14	MR13	MR12	MR11	MR10	MR9	MR8	MR7	MR6	MR5	MR4	MR3	MR2	MR1	MR0

rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31：20	保留，必须始终保持为复位状态（0）。
位 19：0	**MRx**：线 x 上的事件屏蔽（Event Mask on line x） 0：屏蔽来自线 x 上的事件请求； 1：开放来自线 x 上的事件请求。 注：位 19 只适用于 AT32F407，对于其它产品为保留位。

8.3.3 上升沿触发选择寄存器（EXTI_RTSEL）

偏移地址：0x08

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留												TR19	TR18	TR17	TR16
res												rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TR15	TR14	TR13	TR12	TR11	TR10	TR9	TR8	TR7	TR6	TR5	TR4	TR3	TR2	TR1	TR0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31：20	保留，必须始终保持为复位状态（0）。
位 19：0	**TRx**：线 x 上的上升沿触发事件配置位（Rising trigger event configuration bit of line x） 0：禁止输入线 x 上的上升沿触发（中断和事件） 1：允许输入线 x 上的上升沿触发（中断和事件） 注：位 19 只适用于 AT32F407，对于其它产品为保留位。

注意：外部唤醒线是边沿触发的，这些线上不能出现毛刺信号。在写 EXTI_RTRSEL 寄存器时，在外部中断线上的上升沿信号不能被识别，挂起位也不会被置位。在同一中断线上，可以同时设置上升沿和下降沿触发。即任一边沿都可触发中断。

8.3.4 下降沿触发选择寄存器（EXTI_FTRSEL）

偏移地址：0x0C

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留												TR19	TR18	TR17	TR16
res												rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TR15	TR14	TR13	TR12	TR11	TR10	TR9	TR8	TR7	TR6	TR5	TR4	TR3	TR2	TR1	TR0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31：20	保留，必须始终保持为复位状态（0）。
位 19：0	**TRx**：线 x 上的下降沿触发事件配置位（Falling trigger event configuration bit of line x） 0：禁止输入线 x 上的下降沿触发（中断和事件） 1：允许输入线 x 上的下降沿触发（中断和事件） 注：位 19 只适用于 AT32F407，对于其它产品为保留位。

注意：外部唤醒线是边沿触发的，这些线上不能出现毛刺信号。在写 EXTI_FTRSEL 寄存器时，在外部中断线上的下降沿信号不能被识别，挂起位不会被置位。在同一中断线上，可以同时设置上升沿和下降沿触发。即任一边沿都可触发中断。

8.3.5 软件中断事件寄存器（EXTI_SWIE）

偏移地址：0x10

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留												SWIER19	SWIER18	SWIER17	SWIER16
res												rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWIER15	SWIER14	SWIER13	SWIER12	SWIER11	SWIER10	SWIER9	SWIER8	SWIER7	SWIER6	SWIER5	SWIER4	SWIER3	SWIER2	SWIER1	SWIER0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31：20	保留，必须始终保持为复位状态（0）。
位 19：0	**SWIERx**：线 x 上的软件中断（Software interrupt on line x） 如果在 EXTI_INTEN 中这条线的对应位被置“1”，在该位为“0”时对该位写“1”将置起 EXTI_PND 中的对应位，并产生中断请求。 注：通过清除 EXTI_PND 的对应位（写入’1’），可以清除该位为’0’。 注：位 19 只适用于 AT32F407，对于其它产品为保留位。

8.3.6 挂起寄存器（EXTI_PND）

偏移地址：0x14

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留												PR19	PR18	PR17	PR16
res												rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PR15	PR14	PR13	PR12	PR11	PR10	PR9	PR8	PR7	PR6	PR5	PR4	PR3	PR2	PR1	PR0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31：20	保留，必须始终保持为复位状态（0）。
位 19：0	**PRx**： 挂起位（Pending bit） 0：没有发生触发请求 1：发生了选择的触发请求当在外部中断线上发生了选择的边沿事件，该位被置’1’。在该位中写入’1’可以清除它。 注：位 19 只适用于 AT32F407，对于其它产品为保留位。

9 DMA控制器（DMA）

9.1 DMA简介

- 直接存储器存取（DMA）用来提供在外设和存储器之间或者存储器和存储器之间的高速数据传输。无须CPU干预，数据可以通过DMA快速地移动，这就节省了CPU的资源来做其他操作。

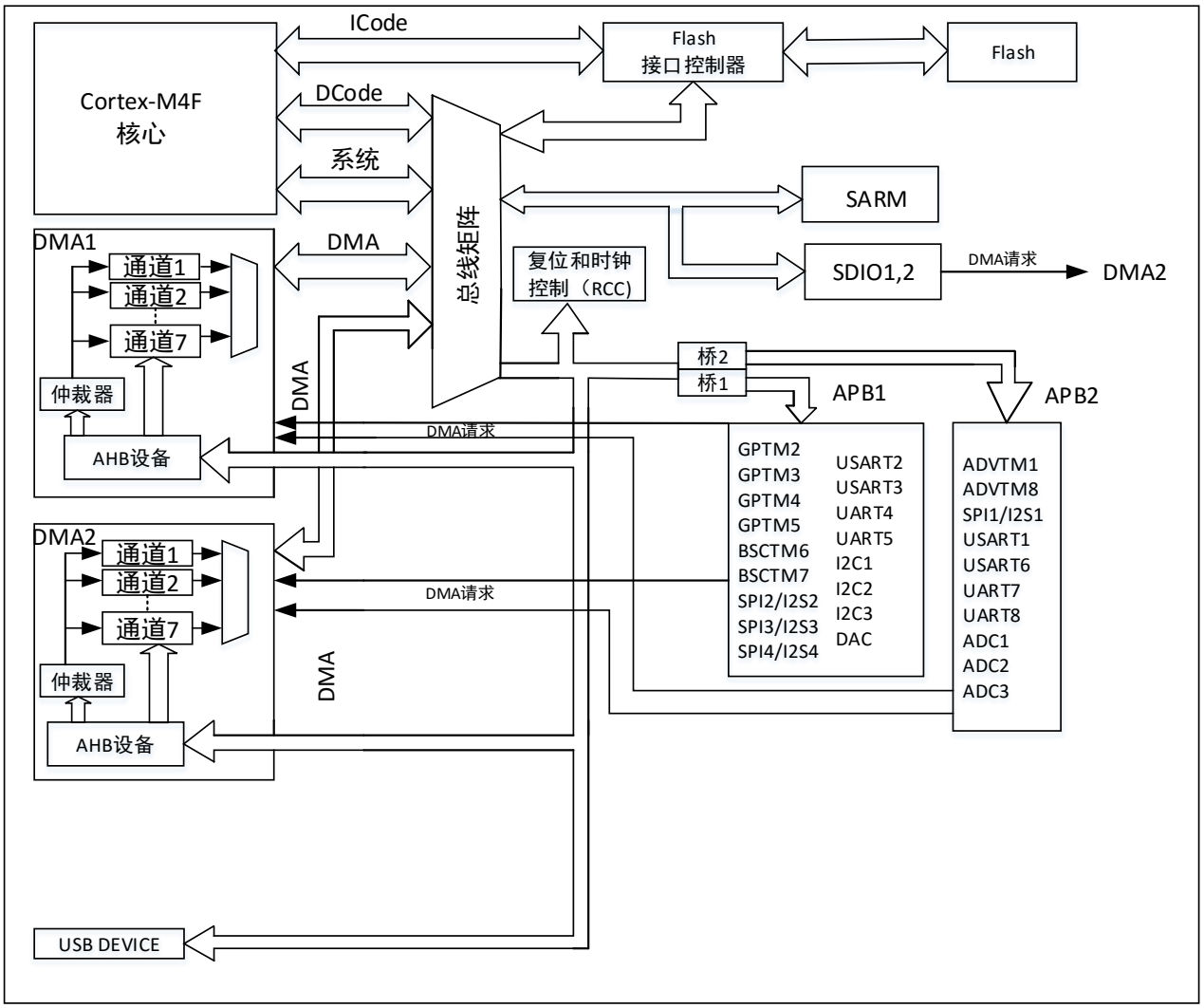
两个 DMA 控制器有 14 个通道（DMA1 有 7 个通道，DMA2 有 7 个通道），每个通道专门用来管理来自于一个或多个外设对存储器访问的请求。还有一个仲裁器来协调各个 DMA 请求的优先权。

9.2 DMA主要特性

- 14个独立的可配置的通道（请求）：DMA1有7个通道，DMA2有7个通道
- 每个通道都直接连接专用的硬件DMA请求，每个通道都同样支持软件触发。这些功能通过软件来配置
- 弹性来源模式支持可随意指定硬件DMA请求到任意通道
- 在同一个DMA模块上，多个请求间的优先权可以通过软件编程设置（共有四级：很高、高、中等和低），优先权设置相等时由硬件决定（请求0优先于请求1，依此类推）
- 独立数据源和目标数据区的传输宽度（字节、半字、全字），模拟打包和拆包的过程。源和目标地址必须按数据传输宽度对齐
- 支持循环的缓冲器管理
- 每个通道都有3个事件标志（DMA半传输、DMA传输完成和DMA传输出错），这3个事件标志逻辑或成为一个单独的中断请求
- 存储器和存储器间的传输
- 外设和存储器、存储器和外设之间的传输
- 闪存、SRAM、外设的SRAM、APB1、APB2和AHB外设均可作为访问的源和目标。
- 可编程的数据传输数目：最大为65535

下面为功能框图：

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

DMA 控制器和 Cortex™-M4F 核心共享系统数据总线，执行直接存储器数据传输。当 CPU 和 DMA 同时访问相同的目标（RAM 或外设）时，DMA 请求会暂停 CPU 访问系统总线达若干个周期，总线仲裁器执行循环调度，以保证 CPU 至少可以得到一半的系统总线（存储器或外设）带宽。

9.3.1 DMA处理

在发生一个事件后，外设向 DMA 控制器发送一个请求信号。DMA 控制器根据通道的优先权处理请求。当 DMA 控制器开始访问发出请求的外设时，DMA 控制器立即发送给它一个应答信号。当从 DMA 控制器得到应答信号时，外设立即释放它的请求。一旦外设释放了这个请求，DMA 控制器同时撤销应答信号。如果有更多的请求时，外设可以启动下一个周期。

总之，每次 DMA 传送由 3 个操作组成：

- 从外设数据寄存器或者从当前外设/存储器地址寄存器指示的存储器地址取数据，第一次传输时的开始地址是DMA_CPBAx或DMA_CMBAx寄存器指定的外设基地址或存储器单元
- 存数据到外设数据寄存器或者当前外设/存储器地址寄存器指示的存储器地址，第一次传输时的开始地址是DMA_CPBAx或DMA_CMBAx寄存器指定的外设基地址或存储器单元

- 执行一次DMA_TCNTx寄存器的递减操作，该寄存器包含未完成的操作数目

9.3.2 仲裁器

仲裁器根据通道请求的优先级来启动外设/存储器的访问。

优先权管理分 2 个阶段：

- 软件：每个通道的优先权可以在DMA_CHCTRLx寄存器中设置，有4个等级
  - 最高优先级
  - 高优先级
  - 中等优先级
  - 低优先级
- 硬件：如果2个请求有相同的软件优先级，则较低编号的通道比较高编号的通道有较高的优先权。举个例子，通道2优先于通道4

注意： DMA1 控制器拥有高于 DMA2 控制器的优先级。

9.3.3 DMA通道

每个通道都可以在有固定地址的外设寄存器和存储器地址之间执行 DMA 传输。DMA 传输的数据宽度是可编程的，最大达到 65535。包含要传输的数据项数量的寄存器，在每次传输后递减。

可编程的数据宽度

外设和存储器的传输数据宽度可以通过 DMA_CHCTRLx 寄存器中的 PWIDTH 和 MWIDTH 编程。

指针增量

通过设置 DMA_CHCTRLx 寄存器中的 PINC 和 MINC 标志位，外设和存储器的指针在每次传输后可以有选择地完成自动增量。当设置为增量模式时，下一个要传输的地址将是前一个地址加上增量值，增量值取决与所选的数据宽度为 1、2 或 4。第一个传输的地址是存放在 DMA_CPBAx/DMA_CMBAx 寄存器中地址。在传输过程中，这些寄存器保持它们初始的数值，软件不能改变和读出当前正在传输的地址（它在内部的当前外设/存储器地址寄存器中）。

当通道配置为非循环模式时，传输结束后（即传输计数变为 0）将不再产生 DMA 操作。要开始新的 DMA 传输，需要在关闭 DMA 通道的情况下，在 DMA_TCNTx 寄存器中重新写入传输数目。

注意： 如果一个 DMA 通道关闭使能，DMA 寄存器值不会被复位。DMA 通道寄存器（DMA_CCRx, DMA_CPARx 和 DMA_CMARx）保持上一次的设置值。

在循环模式下，最后一次传输结束时，DMA_TCNTx 寄存器的内容会自动地被重新加载为其初始数值，内部的当前外设/存储器地址寄存器也被重新加载为 DMA_CPBAx/DMA_CMBAx 寄存器设定的初始基地址。

通道配置过程

下面是配置 DMA 通道 x 的过程（x 代表通道号）：

1. 在DMA_CPBAx寄存器中设置外设寄存器的地址。发生外设数据传输请求时，这个地址将是数据传输的源或目标。
2. 在DMA_CMBAx寄存器中设置数据存储器的地址。发生外设数据传输请求时，传输的数据将从这个地址读出或写入这个地址。
3. 在DMA_TCNTx寄存器中设置要传输的数据宽度。在每个数据传输后，这个数值递减。
4. 在DMA_CHCTRLx寄存器的CHPL[1：0]位中设置通道的优先级。
5. 在DMA_CHCTRLx寄存器中设置数据传输的方向、循环模式、外设和存储器的增量模式、外设和存储器的数据宽度、传输一半产生中断或传输完成产生中断。
6. 设置DMA_CHCTRLx寄存器的ENABLE位，启动该通道。

一旦启动了 DMA 通道，它既可响应连到该通道上的外设的 DMA 请求。

当传输一半的数据后，半传输标志（HTIF）被置 1，当设置了允许半传输中断位（HTIE）时，将产生一

个中断请求。在数据传输结束后，传输完成标志（TCIF）被置 1，当设置了允许传输完成中断位（TCIE）时，将产生一个中断请求。

循环模式

循环模式用于处理循环缓冲区和连续的数据传输（如 ADC 的扫描模式）。在 DMA_CHCTRLx 寄存器中的 CIRM 位用于开启这一功能。当启动了循环模式，数据传输的数目变为 0 时，将会自动地被恢复成配置通道时设置的初值，DMA 操作将会继续进行。

存储器到存储器模式

DMA 通道的操作可以在没有外设请求的情况下进行，这种操作就是存储器到存储器模式。

当设置了 DMA_CHCTRLx 寄存器中的 MEMTOMEM 位之后，在软件设置了 DMA_CHCTRLx 寄存器中的 CHEN 位启动 DMA 通道时，DMA 传输将马上开始。当 DMA_TCNTx 寄存器变为 0 时，DMA 传输结束。存储器到存储器模式不能与循环模式同时使用。

9.3.4 可编程的数据传输宽度、对齐方式和数据大小端

当 PWIDTH 和 MWIDTH 不相同时，DMA 模块按照下表进行数据对齐。

表 9-1 可编程的数据传输宽度和大小端操作（当PINC = MINC = 1）

源端宽度	目标宽度	传输数目	源：地址/数据	传输操作	目标：地址/数据
8	8	4	0x0 / B0 0x1 / B1 0x2 / B2 0x3 / B3	1：在 0x0 读 B0[7：0]，在 0x0 写 B0[7：0] 2：在 0x1 读 B1[7：0]，在 0x1 写 B1[7：0] 3：在 0x2 读 B2[7：0]，在 0x2 写 B2[7：0] 4：在 0x3 读 B3[7：0]，在 0x3 写 B3[7：0]	0x0 / B0 0x1 / B1 0x2 / B2 0x3 / B3
8	16	4	0x0 / B0 0x1 / B1 0x2 / B2 0x3 / B3	1：在 0x0 读 B0[7：0]，在 0x0 写 00B0[15：0] 2：在 0x1 读 B1[7：0]，在 0x2 写 00B1[15：0] 3：在 0x2 读 B2[7：0]，在 0x4 写 00B2[15：0] 4：在 0x3 读 B3[7：0]，在 0x6 写 00B3[15：0]	0x0 / 00B0 0x2 / 00B1 0x4 / 00B2 0x6 / 00B3
8	32	4	0x0 / B0 0x1 / B1 0x2 / B2 0x3 / B3	1：在 0x0 读 B0[7：0]，在 0x0 写 000000B0[31：0] 2：在 0x1 读 B1[7：0]，在 0x4 写 000000B1[31：0] 3：在 0x2 读 B2[7：0]，在 0x8 写 000000B2[31：0] 4：在 0x3 读 B3[7：0]，在 0xC 写 000000B3[31：0]	0x0 / 000000B0 0x4 / 000000B1 0x8 / 000000B2 0xC / 000000B3
16	8	4	0x0 / B1B0 0x2 / B3B2 0x4 / B5B4 0x6 / B7B6	1：在 0x0 读 B1B0[15：0]，在 0x0 写 B0[7：0] 2：在 0x2 读 B3B2[15：0]，在 0x1 写 B2[7：0] 3：在 0x4 读 B5B4[15：0]，在 0x2 写 B4[7：0] 4：在 0x6 读 B7B6[15：0]，在 0x3 写 B6[7：0]	0x0 / B0 0x1 / B2 0x2 / B4 0x3 / B6
16	16	4	0x0 / B1B0 0x2 / B3B2 0x4 / B5B4 0x6 / B7B6	1：在 0x0 读 B1B0[15：0]，在 0x0 写 B1B0[15：0] 2：在 0x2 读 B3B2[15：0]，在 0x2 写 B3B2[15：0] 3：在 0x4 读 B5B4[15：0]，在 0x4 写 B5B4[15：0] 4：在 0x6 读 B7B6[15：0]，在 0x6 写 B7B6[15：0]	0x0 / B1B0 0x2 / B3B2 0x4 / B5B4 0x6 / B7B6
16	32	4	0x0 / B1B0 0x2 / B3B2 0x4 / B5B4 0x6 / B7B6	1：在 0x0 读 B1B0[15：0]，在 0x0 写 0000B1B0[31：0] 2：在 0x2 读 B3B2[15：0]，在 0x4 写 0000B3B2[31：0] 3：在 0x4 读 B5B4[15：0]，在 0x8 写 0000B5B4[31：0] 4：在 0x6 读 B7B6[15：0]，在 0xC 写 0000B7B6[31：0]	0x0 / 0000B1B0 0x4 / 0000B3B2 0x8 / 0000B5B4 0xC / 0000B7B6
32	8	4	0x0 / B3B2B1B0 0x4 / B7B6B5B4 0x8 / BBBAB9B8 0xC / BFBEBDBC	1：在 0x0 读 B3B2B1B0[31：0]，在 0x0 写 B0[7：0] 2：在 0x4 读 B7B6B5B4[31：0]，在 0x1 写 B4[7：0] 3：在 0x8 读 BBBAB9B8[31：0]，在 0x2 写 B8[7：0] 4：在 0xC 读 BFBEBDBC[31：0]，在 0x3 写 BC[7：0]	0x0 / B0 0x1 / B4 0x2 / B8 0x3 / BC
32	16	4	0x0 / B3B2B1B0 0x4 / B7B6B5B4 0x8 / BBBAB9B8 0xC / BFBEBDBC	1：在 0x0 读 B3B2B1B0[31：0]，在 0x0 写 B1B0[15：0] 2：在 0x4 读 B7B6B5B4[31：0]，在 0x2 写 B5B4[15：0] 3：在 0x8 读 BBBAB9B8[31：0]，在 0x4 写 B9B8[15：0] 4：在 0xC 读 BFBEBDBC[31：0]，在 0x6 写 BDBC[15：0]	0x0 / B1B0 0x2 / B5B4 0x4 / B9B8 0x6 / BDBC
32	32	4	0x0 / B3B2B1B0 0x4 / B7B6B5B4 0x8 / BBBAB9B8 0xC / BFBEBDBC	1：在 0x0 读 B3B2B1B0[31：0]，在 0x0 写 B3B2B1B0[31：0] 2：在 0x4 读 B7B6B5B4[31：0]，在 0x4 写 B7B6B5B4[31：0] 3：在 0x8 读 BBBAB9B8[31：0]，在 0x8 写 BBBAB9B8[31：0] 4：在 0xC 读 BFBEBDBC[31：0]，在 0xC 写 BFBEBDBC[31：0]	0x0 / B3B2B1B0 0x4 / B7B6B5B4 0x8 / BBBAB9B8 0xC / BFBEBDBC

操作一个不支持字节或半字写的 AHB 设备

当 DMA 模块开始一个 AHB 的字节或半字写操作时，数据将在 HWDATA[31：0]总线中未使用的部分重复。因此，如果 DMA 以字节或半字写入不支持字节或半字写操作的 AHB 设备时（即 HSIZE 不适于该模块），不会发生错误，DMA 将按照下面两个例子写入 32 位 HWDATA 数据：

- 当HSIZE=半字时，写入半字’0xABCD’，DMA将设置HWDATA总线为’0xABCDABCD’
- 当HSIZE=字节时，写入字节’0xAB’，DMA将设置HWDATA总线为’0xABABABAB’

假定 AHB/APB 桥是一个 AHB 的 32 位从设备，它不处理 HSIZE 参数，它将按照下述方式把任何 AHB 上的字节或半字按 32 位传送到 APB 上：

- 一个AHB上对地址0x0（或0x1、0x2或0x3）的写字节数据’0xB0’操作，将转换到APB上对地址0x0的写字数据’0xB0B0B0B0’操作
- 一个AHB上对地址0x0（或0x2）的写半字数据’0xB1B0’操作，将转换到APB上对地址0x0的写字数据’0xB1B0B1B0’操作

例如，如果要写入 APB 后备寄存器（与 32 位地址对齐的 16 位寄存器），需要配置存储器数据源宽度（MWIDTH）为’16 位’，外设目标数据宽度（(PWIDTH）为’32 位’。

9.3.5 错误管理

读写一个保留的地址区域，将会产生 DMA 传输错误。当在 DMA 读写操作时发生 DMA 传输错误时，硬件会自动地清除发生错误的通道所对应的通道配置寄存器（DMA_CHCTRLx）的 CHEN 位，该通道操作被停止。此时，在 DMA_IFR 寄存器中对应该通道的传输错误中断标志位（ERRIF）将被置位，如果在 DMA_CHCTRLx 寄存器中设置了传输错误中断允许位，则将产生中断。

9.3.6 中断

每个 DMA 通道都可以在 DMA 传输过半、传输完成和传输错误时产生中断。为应用的灵活性考虑，通过设置寄存器的不同位来打开这些中断。

表 9-2 DMA中断请求

中断事件	事件标志位	使能控制位
传输过半	HTIF	HTIE
传输完成	TCIF	TCIE
传输错误	ERRIF	ERRIE

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

9.3.7 DMA固定请求映像

DMA1 控制器

从外设（TMRx[x=1、2、3、4]、ADC1、SPI1、SPI/I2S2、I2Cx[x=1、2]和 USARTx[x=1、2、3]）产生的 7 个请求，通过逻辑或输入到 DMA1 控制器，这意味着同时只能有一个请求有效。参见下图的 DMA1 请求映像。

外设的 DMA 请求，可以通过设置相应外设寄存器中的控制位，被独立地开启或关闭。

图 9-2 DMA1请求映像

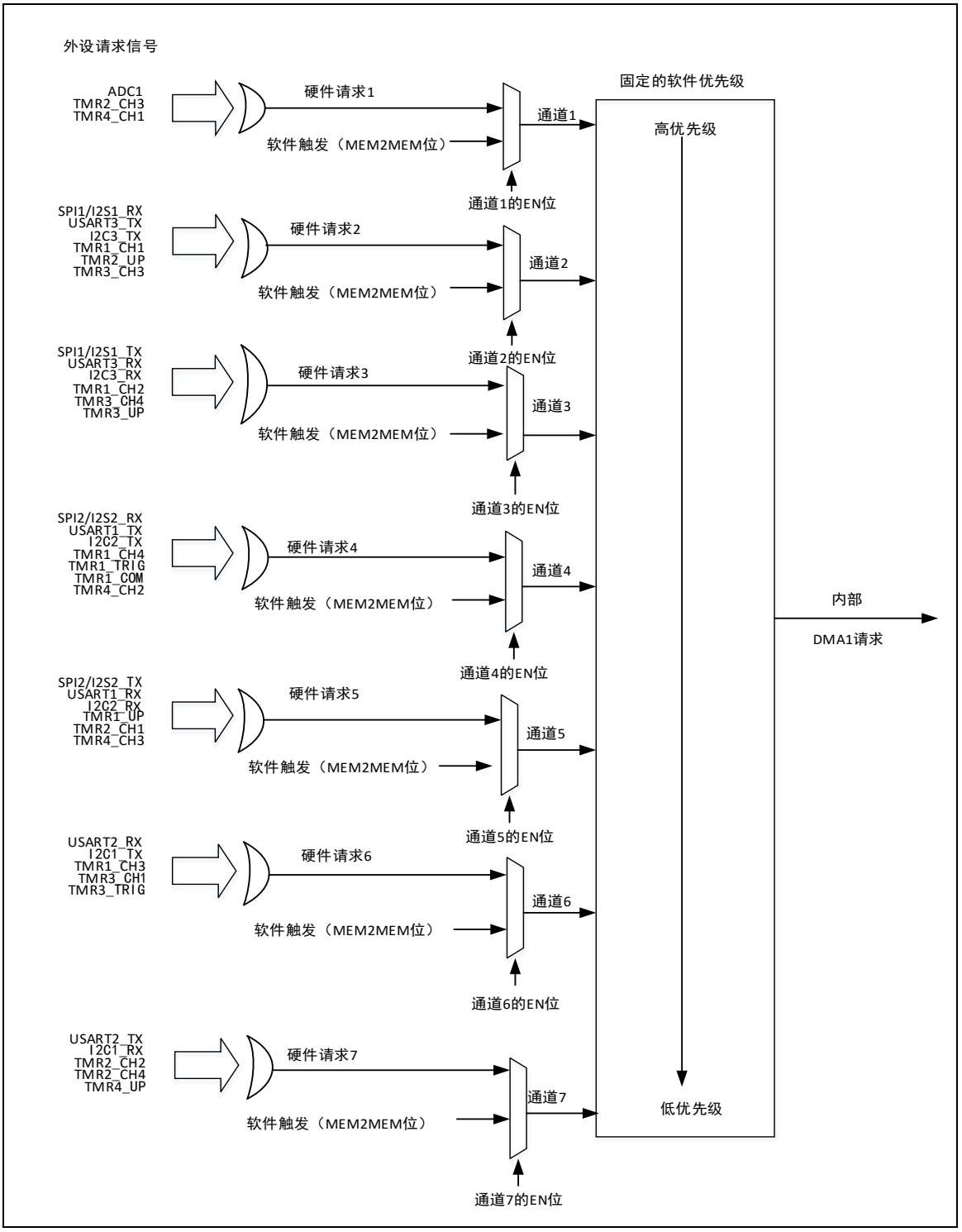


表 9-3 各个通道的DMA1请求一览

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/I^2S		SPI1/I2S1_RX	SPI1/I2S1_TX	SPI2/I2S2_RX	SPI2/I2S2_TX		
USART		USART3_TX	USART3_RX	USART1_TX	USART1_RX	USART2_RX	USART2_TX

I^2C		I2C3_TX	I2C3_RX	I2C2_TX	I2C2_RX	I2C1_TX	I2C1_RX
TMR1		TMR1_CH1	TMR1_CH2	TMR1_CH4 TMR1_TRIG TMR1_COM	TMR1_UP	TMR1_CH3	
TMR2	TMR2_CH3	TMR2_UP			TMR2_CH1		TMR2_CH2 TMR2_CH4
TMR3		TMR3_CH3	TMR3_CH4 TMR3_UP			TMR3_CH1 TMR3_TRIG	
TMR4	TMR4_CH1			TMR4_CH2	TMR4_CH3		TMR4_UP

DMA2 控制器

从外设（TMRx[5、8]、UART4 和 SDIO）产生的 5 个请求，经逻辑或输入到 DMA2 控制器，这意味着同时只能有一个请求有效。参见下图的 DMA2 请求映像。

外设的 DMA 请求，可以通过设置相应外设寄存器中的 DMA 控制位，被独立地开启或关闭。

图 9-3 DMA2请求映像

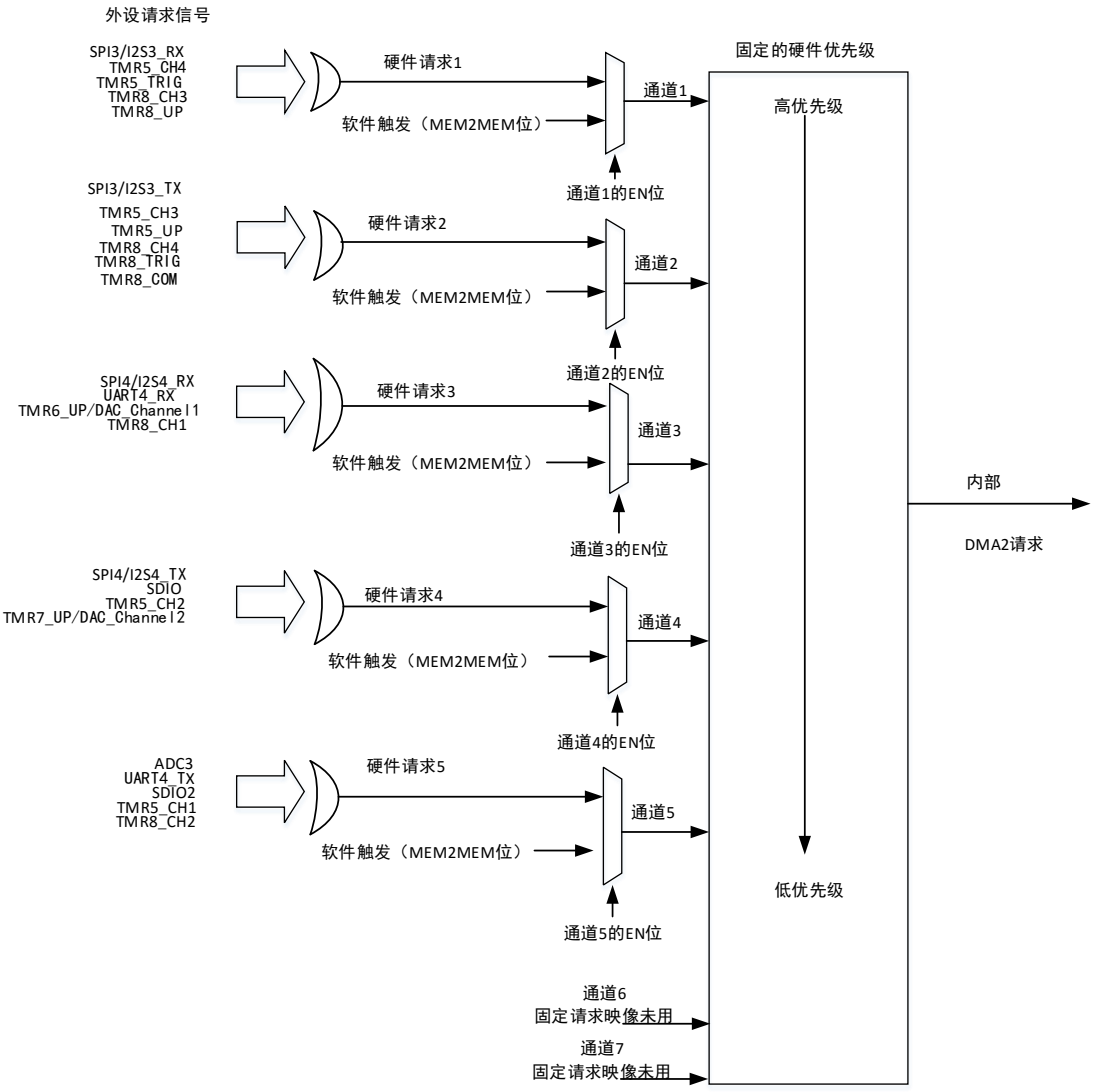


表 9-4 各个通道的DMA2请求一览

外设	通道 **1**	通道 **2**	通道 **3**	通道 **4**	通道 **5**
ADC3					ADC3

SPI/I2S	SPI3/I2S3_RX	SPI3/I2S3_TX	SPI4/I2S4_RX	SPI4/I2S4_TX	
UART4			UART4_RX		UART4_TX
SDIO				SDIO	
SDIO2					SDIO2
TMR5	TMR5_CH4 TMR5_TRIG	TMR5_CH3 TMR5_UP		TMR5_CH2	TMR5_CH1
TMR6/ DAC 通道 1			TMR6_UP/ DAC 通道 1		
TMR7/ DAC 通道 2				TMR7_UP/ DAC 通道 2	
TMR8	TMR8_CH3 TMR8_UP	TMR8_CH4 TMR8_TRIG TMR8_COM	TMR8_CH1		TMR8_CH2

9.3.8 DMA弹性请求映像

当 DMA_FLEX_EN 设定弹性模式时，每个通道的请求来源由 CHx_SRC 来设定[x=1、2、3、4、5、6、7]。CHx_SRC 设定值对应 DMA 来源见下表:

表 9-5 各个信道的DMA弹性请求一览

CHx_SRC 设定值	DMA 来源	CHx_SRC 设定值	DMA 来源	CHx_SRC 设定值	DMA 来源	CHx_SRC 设定值	DMA 来源
0	No select	1	ADC1	2	reserved	3	ADC3
4	reserved	5	DAC1	6	DAC2	7	reserved
8	reserved	9	SPI1_RX	10	SPI1_TX	11	SPI2_RX
12	SPI2_TX	13	SPI3_RX	14	SPI3_TX	15	SPI4_RX
16	SPI4_TX	17	I2S2EXT_RX	18	I2S2EXT_TX	19	I2S3EXT_RX
20	I2S3EXT_TX	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	UART4_RX
32	UART4_TX	33	UART5_RX	34	UART5_TX	35	USART6_RX
36	USART6_TX	37	UART7_RX	38	UART7_TX	39	UART8_RX
40	UART8_TX	41	I2C1_RX	42	I2C1_TX	43	I2C2_RX
44	I2C2_TX	45	I2C3_RX	46	I2C3_TX	47	reserved
48	reserved	49	SDIO1	50	SDIO2	51	reserved
52	reserved	53	TIM1_TRIG	54	TIM1_COM	55	TIM1_UP
56	TIM1_CH1	57	TIM1_CH2	58	TIM1_CH3	59	TIM1_CH4
60	reserved	61	TIM2_TRIG	62	reserved	63	TIM2_UP
64	TIM2_CH1	65	TIM2_CH2	66	TIM2_CH3	67	TIM2_CH4
68	reserved	69	TIM3_TRIG	70	reserved	71	TIM3_UP
72	TIM3_CH1	73	TIM3_CH2	74	TIM3_CH3	75	TIM3_CH4
76	reserved	77	TIM4_TRIG	78	reserved	79	TIM4_UP
80	TIM4_CH1	81	TIM4_CH2	82	TIM4_CH3	83	TIM4_CH4
84	reserved	85	TIM5_TRIG	86	reserved	87	TIM5_UP

88	TIM5_CH1	89	TIM5_CH2	90	TIM5_CH3	91	TIM5_CH4
92	reserved	93	reserved	94	reserved	95	TIM6_UP
96	reserved	97	reserved	98	reserved	99	reserved
100	reserved	101	reserved	102	reserved	103	TIM7_UP
104	reserved	105	reserved	106	reserved	107	reserved
108	reserved	109	TIM8_TRIG	110	TIM8_COM	111	TIM8_UP
112	TIM8_CH1	113	TIM8_CH2	114	TIM8_CH3	115	TIM8_CH4
116	reserved	117	reserved	118	reserved	119	reserved
120	reserved	121	reserved	122	reserved	123	reserved
124	reserved	125	reserved	126	reserved	127	reserved
128	reserved	129	reserved	130	reserved	131	reserved
132	reserved	133	reserved	134	reserved	135	reserved
136	reserved	137	reserved	138	reserved	139	reserved
140	reserved	141	reserved	142	reserved	143	reserved
144	reserved	145	reserved	146	reserved	147	reserved
148	reserved	149	reserved	150	reserved	151	reserved
152	reserved	153	reserved	154	reserved	155	reserved
156	reserved	157	reserved	158	reserved	159	reserved
160	reserved	161	reserved	162	reserved	163	reserved
164	reserved	165	reserved	166	reserved	167	reserved
168	reserved	169	reserved	170	reserved	171	reserved
172	reserved	173	reserved	174	reserved	175	reserved

当 DMA1/2 的 DMA_FLEX_EN 必须要同时设定为 1 或 0 时，DMA1/2 的映像模式必须一致。无法 DMA1 是固定式映像，DMA2 是弹性式映像。

使用例子： 假如 DMA 通道 1 要指定成 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。

9.4 DMA寄存器

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

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。

表 9-6 DMA寄存器映像和复位值

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000h	DMA_ISTS	保留				ERRIF7	HTIF7	TCIF7	GIF7	ERRIF6	HTIF6	TCIF6	GIF6	ERRIF5	HTIF5	TCIF5	GIF5	ERRIF4	HTIF4	TCIF4	GIF4	ERRIF3	HTIF3	TCIF3	GIF3	ERRIF2	HTIF2	TCIF2	GIF2	ERRIF1	HTIF1	TCIF1	GIF1
	复位值					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
004h	DMA_ICLR	保留				CERRIF7	CHTIF7	CTCIF7	CGIF7	CERRIF6	CHTIF6	CTCIF6	CGIF6	CERRIF5	CHTIF5	CTCIF5	CGIF5	CERRIF4	CHTIF4	CTCIF4	CGIF4	CERRIF3	CHTIF3	CTCIF3	CGIF3	CERRIF2	CHTIF2	CTCIF2	CGIF2	CERRIF1	CHTIF1	CTCIF1	CGIF1
	复位值					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

008h	DMA_CHCTRL1	保留																	保留	MEMTOMEM	CHPL[1：0]		MWIDTH[1：0]		PWIDTH[1：0]		MINC	PINC	CIRM	DIR	ERRIE	HTIE	TCIE	CHEN
	复位值																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
00Ch	DMA_TCNT1	保留																	CNT[15：0]															
	复位值																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
010h	DMA_CPBA1	PA[31：0]																															
	复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
014h	DMA_CMBA1	MA[31：0]																															
	复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
018h	保留																																
01Ch	DMA_CHCTRL2	保留																		MEMTOMEM	CHPL[1：0]		MWIDTH[1：0]		PWIDTH[1：0]		MINC	PINC	CIRM	DIR	ERRIE	HTIE	TCIE	CHEN
	复位值																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
020h	DMA_TCNT2	保留																	CNT[15：0]															
	复位值																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
024h	DMA_CPBA2	PA[31：0]																															
	复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
028h	DMA_CMBA2	MA[31：0]																															
	复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
02Ch	保留																																
030h	DMA_CHCTRL3	保留																		MEMTOMEM	CHPL[1：0]		MWIDTH[1：0]		PWIDTH[1：0]		MINC	PINC	CIRM	DIR	ERRIE	HTIE	TCIE	CHEN
	复位值																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
034h	DMA_TCNT3	保留																	CNT[15：0]															
	复位值																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
038h	DMA_CPBA3	PA[31：0]																															
	复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
03Ch	DMA_CMBA3	MA[31：0]																															
	复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
040h	保留																																
044h	DMA_CHCTRL4	保留																		MEMTOMEM	CHPL[1：0]		MWIDTH[1：0]		PWIDTH[1：0]		MINC	PINC	CIRM	DIR	ERRIE	HTIE	TCIE	CHEN
	复位值																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
048h	DMA_TCNT4	保留																	CNT[15：0]															
	复位值																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
04Ch	DMA_CPBA4	PA[31：0]																															

	复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
050h	DMA_CMBA4	MA[31：0]																													
	复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
054h	保留																														
058h	DMA_CHCTRL5	保留																MEMTOMEM	CHPL[1：0]		MWIDTH[1：0]		PWIDTH[1：0]		MINC	PINC	CIRM	DIR	ERRIE	HTIE	TCIE	CHEN
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0
05Ch	DMA_TCNT5	保留																CNT[15：0]													
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0
060h	DMA_CPBA5	PA[31：0]																													
	复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
064h	DMA_CMBA5	MA[31：0]																													
	复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
068h	保留																														
06Ch	DMA_CHCTRL6	保留																MEMTOMEM	CHPL[1：0]		MWIDTH[1：0]		PWIDTH[1：0]		MINC	PINC	CIRM	DIR	ERRIE	HTIE	TCIE	CHEN
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0
070h	DMA_TCNT6	保留																CNT[15：0]													
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0
074h	DMA_CPBA6	PA[31：0]																													
	复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
078h	DMA_CMBA6	MA[31：0]																													
	复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
07Ch	保留																														
080h	DMA_CHCTRL7	保留																MEMTOMEM	CHPL[1：0]		MWIDTH[1：0]		PWIDTH[1：0]		MINC	PINC	CIRM	DIR	ERRIE	HTIE	TCIE	CHEN
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0
084h	DMA_TCNT7	保留																CNT[15：0]													
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0
088h	DMA_CPBA7	PA[31：0]																													
	复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
08Ch	DMA_CMBA7	MA[31：0]																													
	复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
090h ~09Ch	保留																														
0A0h	DMA_SRC_SEL0	CH4_SRC								CH3_SRC								CH2_SRC								CH1_SRC					
	复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

0A4h	DMA_SRC_SEL1	保留							DMA_FLEX_EN	CH7_SRC								CH6_SRC								CH5_SRC							
	复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0A8h	保留																														

9.4.1 DMA中断状态寄存器（DMA_ISTS）

偏移地址：0x00

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留				ERRIF7	HTIF7	TCIF7	GIF7	ERRIF6	HTIF6	TCIF6	GIF6	ERRIF5	HTIF5	TCIF5	GIF5
				r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ERRIF4	HTIF4	TCIF4	GIF4	ERRIF3	HTIF3	TCIF3	GIF3	ERRIF2	HTIF2	TCIF2	GIF2	ERRIF1	HTIF1	TCIF1	GIF1
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

位 31：28	保留，始终读为 0。
位 27，23，19，15，11，7，3	**ERRIFx**：通道 x 的传输错误标志（x = 1 … 7） （Channel x transfer error flag） 硬件设置这些位。在 DMA_ICLR 寄存器的相应位写入’1’可以清除这里对应的标志位。 0：在通道 x 没有传输错误（TE）； 1：在通道 x 发生了传输错误（TE）。
位 26，22，18，14，10，6，2	**HTIFx**：通道 x 的半传输标志（x = 1 … 7） （Channel x half transfer flag） 硬件设置这些位。在 DMA_ICLR 寄存器的相应位写入’1’可以清除这里对应的标志位。 0：在通道 x 没有半传输事件（HT）； 1：在通道 x 产生了半传输事件（HT）。
位 25，21，17，13，9，5，1	**TCIFx**：通道 x 的传输完成标志（x = 1 … 7） （Channel x transfer complete flag） 硬件设置这些位。在 DMA_ICLR 寄存器的相应位写入’1’可以清除这里对应的标志位。 0：在通道 x 没有传输完成事件（TC）； 1：在通道 x 产生了传输完成事件（TC）。
位 24，20，16，12，8，4，0	**GIFx**：通道 x 的全局中断标志（x = 1 … 7） （Channel x global interrupt flag） 硬件设置这些位。在 DMA_ICLR 寄存器的相应位写入’1’可以清除这里对应的标志位。 0：在通道 x 没有 TE、HT 或 TC 事件； 1：在通道 x 产生了 TE、HT 或 TC 事件。

9.4.2 DMA中断标志清除寄存器（DMA_ICLR）

偏移地址：0x04

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留				CERRIF7	CHTIF7	CTCIF7	CGIF7	CERRIF6	CHTIF6	CTCIF6	CGIF6	CERRIF5	CHTIF5	CTCIF5	CGIF5
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CERRIF4	CHTIF4	CTCIF4	CGIF4	CERRIF3	CHTIF3	CTCIF3	CGIF3	CERRIF2	CHTIF2	CTCIF2	CGIF2	CERRIF1	CHTIF1	CTCIF1	CGIF1
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31：28	保留，始终读为 0。
位 27，23，19，15，11，7，3	**CERRIFx**：清除通道 x 的传输错误标志（x = 1 … 7） （Channel x transfer error clear） 这些位由软件设置和清除。 0：不起作用 1：清除 DMA_ISTS 寄存器中的对应 ERRIF 标志。

位 26，22，18，14，10，6，2	**CHTIFx**：清除通道 x 的半传输标志（x = 1 … 7） （Channel x half transfer clear） 这些位由软件设置和清除。 0：不起作用 1：清除 DMA_ISTS 寄存器中的对应 HTIF 标志。
位 25，21，17 ，13 ，9，5，1	**CTCIFx**：清除通道 x 的传输完成标志（x = 1 … 7） （Channel x transfer complete clear） 这些位由软件设置和清除。 0：不起作用 1：清除 DMA_ISTS 寄存器中的对应 TCIF 标志。
位 24，20，16，12，8，4，0	**CGIFx**：清除通道 x 的全局中断标志（x = 1 … 7） （Channel x global interrupt clear） 这些位由软件设置和清除。 0：不起作用 1：清除 DMA_ISTS 寄存器中的对应的 GIF、ERRIF、HTIF 和 TCIF 标志。

9.4.3 DMA通道x配置寄存器（DMA_CHCTRLx）（x = 1…7）

偏移地址：0x08 + 20 x （通道编号 – 1）

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留	MEMTOMEM	CHPL[1:00]		MWIDTH[1：0]		PWIDTH[1：0]		MINC	PINC	CIRM	DIR	ERRIE	HTIE	TCIE	CHEN
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31：15	保留，始终读为 0。
位 14	**MEMTOMEM**：存储器到存储器模式I（Memory to memory mode） 该位由软件设置和清除。 0：非存储器到存储器模式； 1：启动存储器到存储器模式。
位 13：12	**CHPL[1：0]**：通道优先级（Channel priority level） 这些位由软件设置和清除。 00：低 01：中 10：高 11：最高
位 11：10	**MWIDTH[1：0]**：存储器数据宽度（Memory size） 这些位由软件设置和清除。 00：8 位 01：16 位 10：32 位 11：保留
位 9：8	**PWIDTH[1：0]**：外设数据宽度（Peripheral size） 这些位由软件设置和清除。 00：8 位 01：16 位 10：32 位 11：保留
位 7	**MINC**：存储器地址增量模式（Memory increment mode） 该位由软件设置和清除。 0：不执行存储器地址增量操作 1：执行存储器地址增量操作

位 6	**PINC**：外设地址增量模式（Peripheral increment mode） 该位由软件设置和清除。 0：不执行外设地址增量操作 1：执行外设地址增量操作
位 5	**CIRM**：循环模式（Circular mode） 该位由软件设置和清除。 0：不执行循环操作 1：执行循环操作
位 4	**DIR**：数据传输方向（Data transfer direction） 该位由软件设置和清除。 0：从外设读 1：从存储器读
位 3	**ERRIE**：允许传输错误中断（Transfer error interrupt enable） 该位由软件设置和清除。 0：禁止 TE 中断 1：允许 TE 中断
位 2	**HTIE**：允许半传输中断（Half transfer interrupt enable） 该位由软件设置和清除。 0：禁止 HT 中断 1：允许 HT 中断
位 1	**TCIE**：允许传输完成中断（Transfer complete interrupt enable） 该位由软件设置和清除。 0：禁止 TC 中断 1：允许 TC 中断
位 0	**CHEN**：通道使能（Channel enable） 该位由软件设置和清除。 0：通道关闭 1：通道使能

9.4.4 DMA通道x传输数量寄存器（DMA_TCNTx）（x = 1…7）

偏移地址：0x0C + 20 x （通道编号– 1）

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT[15：0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31：16	保留，始终读为 0。
位 15：0	**CNT[15：0]**：数据传输数量（Number of data to transfer） 数据传输数量为 0 至 65535。这个寄存器只能在通道不工作（DMA_CHCTRLx 的 CHEN=0）时写入。通道开启后该寄存器变为只读，指示剩余的待传输字节数目。寄存器内容在每次 DM A 传输后递减。 数据传输结束后，寄存器的内容或者变为 0；或者当该通道配置为自动重加载模式时，寄存器的内容将被自动重新加载为之前配置时的数值。 当寄存器的内容为 0 时，无论通道是否开启，都不会发生任何数据传输。

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

偏移地址：0x10 + 20 x （通道编号 – 1）

复位值：0x0000 0000

当开启通道（DMA_CHCTRLx 的 CHEN=1）时不能写该寄存器。

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

位 31：0	**PA[31：0]**：外设地址（Peripheral address） 外设数据寄存器的基地址，作为数据传输的源或目标。 当 PWIDTH=’01’（16 位），不使用 PA[0]位。操作自动地与半字地址对齐。 当 PWIDTH=’10’（32 位），不使用 PA[1：0]位。操作自动地与字地址对齐。

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

偏移地址：0x14 + 20 x （通道编号 – 1）

复位值：0x0000 0000

当开启通道（DMA_CHCTRLx 的 CHEN=1）时不能写该寄存器。

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

位 31：0	**MA[31：0]**：储存器地址 储存器地址作为数据传输的源或目标。 当 MWIDTH=’01’（16 位），不使用 MA[0]位。操作自动地与半字地址对齐。 当 MWIDTH=’10’（32 位），不使用 MA[1：0]位。操作自动地与字地址对齐。

9.4.7 通道来源寄存器0（DMA_SRC_SEL0）

偏移地址：0xA0

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CH4_SRC								CH3_SRC							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CH2_SRC								CH1_SRC							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:24	**CH4_SRC: CH4 来源的选择位** 当 DMA_FLEX_EN=1 时，有 CH4_SRC 选择 CH4 来源，详见 9.3.8DMA 弹性请求映像
位 23:16	**CH3_SRC: CH3 来源的选择位** 当 DMA_FLEX_EN=1 时，有 CH3_SRC 选择 CH3 来源，详见 9.3.8DMA 弹性请求映像
位 15:8	**CH2_SRC: CH2 来源的选择位** 当 DMA_FLEX_EN=1 时，有 CH2_SRC 选择 CH2 来源，详见 9.3.8DMA 弹性请求映像
位 7:0	**CH1_SRC: CH1 来源的选择位** 当 DMA_FLEX_EN=1 时，有 CH1_SRC 选择 CH1 来源，详见 9.3.8DMA 弹性请求映像

9.4.8 通道來源寄存器1（DMA_SRC_SEL1）

偏移地址：0xA4

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
							DMA_FLEX_EN	CH7_SRC							
			res				rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CH6_SRC								CH5_SRC							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:24	**DMA_FLEX_EN: DMA 请求映像模式选择位。** 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: CH7 来源的选择位** 当 DMA_FLEX_EN=1 时，有 CH7_SRC 选择 CH7 来源，详见 9.3.8DMA 弹性请求映像
位 15:8	**CH6_SRC: CH6 来源的选择位** 当 DMA_FLEX_EN=1 时，有 CH6_SRC 选择 CH6 来源，详见 9.3.8DMA 弹性请求映像
位 7:0	**CH5_SRC: CH5 来源的选择位** 当 DMA_FLEX_EN=1 时，有 CH5_SRC 选择 CH5 来源，详见 9.3.8DMA 弹性请求映像

10 定时器（TIMER）

AT32F403A/407 定时器可分为通用定时器与高级控制定时器，详细功能模式可参考 10.1~10.4 节说明，下表为各种类型定时器的功能总表。

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	ETR 输入	刹车输入
高级控制定时器	TMR1 TMR8	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	支持	不支持
	TMR3 TMR4	16	向上 向下 向上/向下	不支持	1~65535	支持	4	支持	支持	不支持
	TMR9 TMR12	16	向上	不支持	1~65535	支持	2	支持	不支持	不支持
	TMR10 TMR11 TMR13 TMR14	16	向上	不支持	1~65535	支持	1	不支持	不支持	不支持
基本定时器	TMR6 TMR7	16	向上	不支持	1~65535	支持	不支持	不支持	不支持	不支持

Timer 类型	Timer	计数位数	计数方式	PWM 输出	单脉冲输出	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 TMR8	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
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
	TMR3 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
	TMR9 TMR12	16	向上	支持	支持	不支持	不支持	不支持	不支持	定时器同步 ADC/DAC
	TMR10 TMR11 TMR13 TMR14	16	向上	支持	支持	不支持	不支持	不支持	不支持	无
基本定时器	TMR6 TMR7	16	向上	不支持	不支持	不支持	不支持	不支持	不支持	DAC

10.1基本定时器（TMR6和TMR7）

10.1.1 TMR6和TMR7简介

基本定时器 TMR6 和 TMR7 各包含一个 16 位自动装载计数器，由各自的可编程预分频器驱动。它们可

以作为通用定时器提供时间基准，特别地可以为数模转换器（DAC）提供时钟。实际上，它们在芯片内部直接连接到 DAC 并通过触发输出直接驱动 DAC。

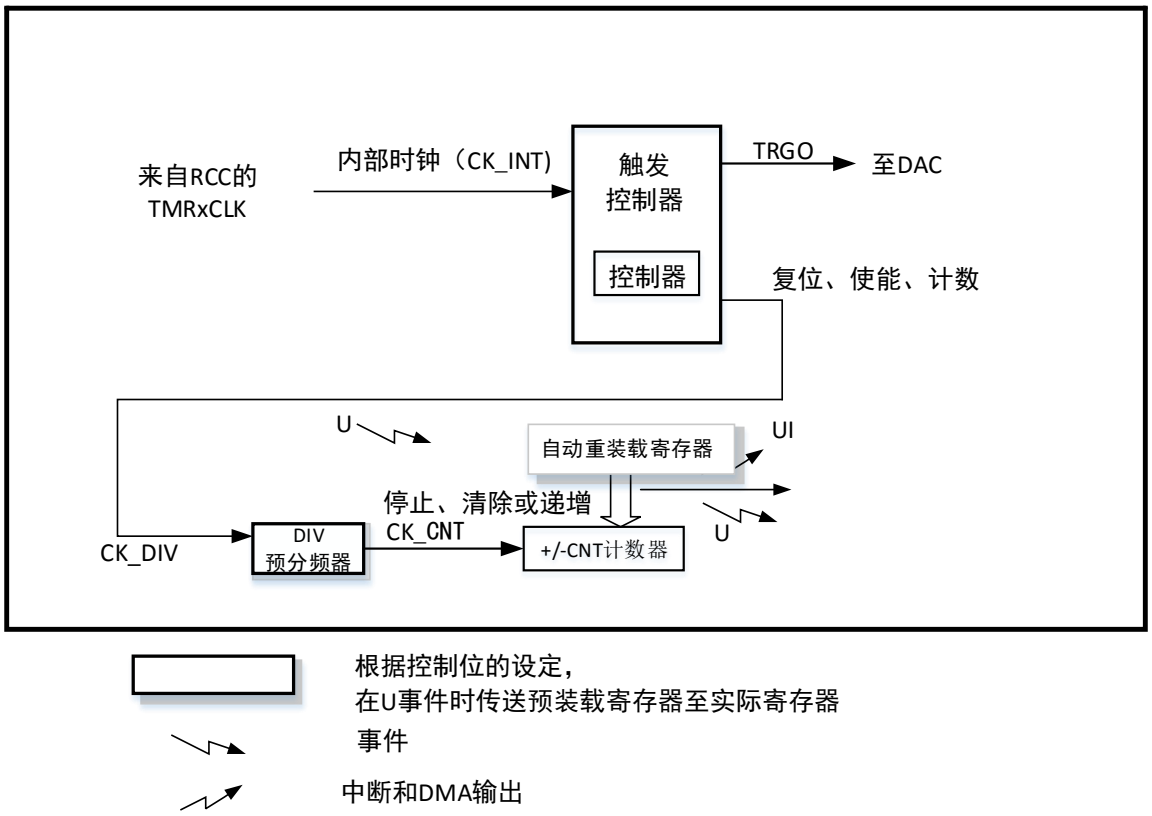
这 2 个定时器是互相独立的，不共享任何资源。

10.1.2 TMR6和TMR7的主要特性

TMR6 和 TMR7 定时器的主要功能包括：

- 16位自动重装载累加计数器
- 16位可编程（可实时修改）预分频器，用于对输入时钟按系数为1～65536之间的任意数值分频
- 触发DAC的同步电路
- 在更新事件（计数器溢出）时产生中断/DMA请求

图 10-1 基本定时器框图



10.1.3 TMR6和TMR7的功能

10.1.3.1 时基单元

这个可编程定时器的主要部分是一个带有自动重装载的 16 位累加计数器，计数器的时钟通过一个预分频器得到。

软件可以读写计数器、自动重装载寄存器和预分频寄存器，即使计数器运行时也可以操作。

时基单元包含：

- 计数器寄存器（TMRx_CNT）
- 预分频寄存器（TMRx_DIV）
- 自动重装载寄存器（TMRx_AR）

自动重装载寄存器是预加载的，每次读写自动重装载寄存器时，实际上是通过读写预加载寄存器实现。根据 TMRx_CTRL1 寄存器中的自动重装载预加载使能位（ARPEN），写入预加载寄存器的内容能够立即或在每次更新事件时，传送到它的影子寄存器。当 TMRx_CTRL1 寄存器的 UEVDIS 位为’0’，则每当计

数器达到溢出值时，硬件发出更新事件；软件也可以产生更新事件；关于更新事件的产生，随后会有详细的介绍。

计数器由预分频输出 CK_CNT 驱动，设置 TMRx_CTRL1 寄存器中的计数器使能位（CNTEN）使能计数器计数。

注意: 实际的设置计数器使能信号 CNT_EN 相对于 CNTEN 滞后一个时钟周期。

10.1.3.2 预分频器

预分频可以以系数介于 1 至 65536 之间的任意数值对计数器时钟分频。它是通过一个 16 位寄存器（TMRx_DIV）的计数实现分频。因为 TMRx_DIV 控制寄存器具有缓冲，可以在运行过程中改变它的数值，新的预分频数值将在下一个更新事件时起作用。

以下两图是在运行过程中改变预分频系数的例子。

图 10-2 预分频系数从1变到2的计数器时序图

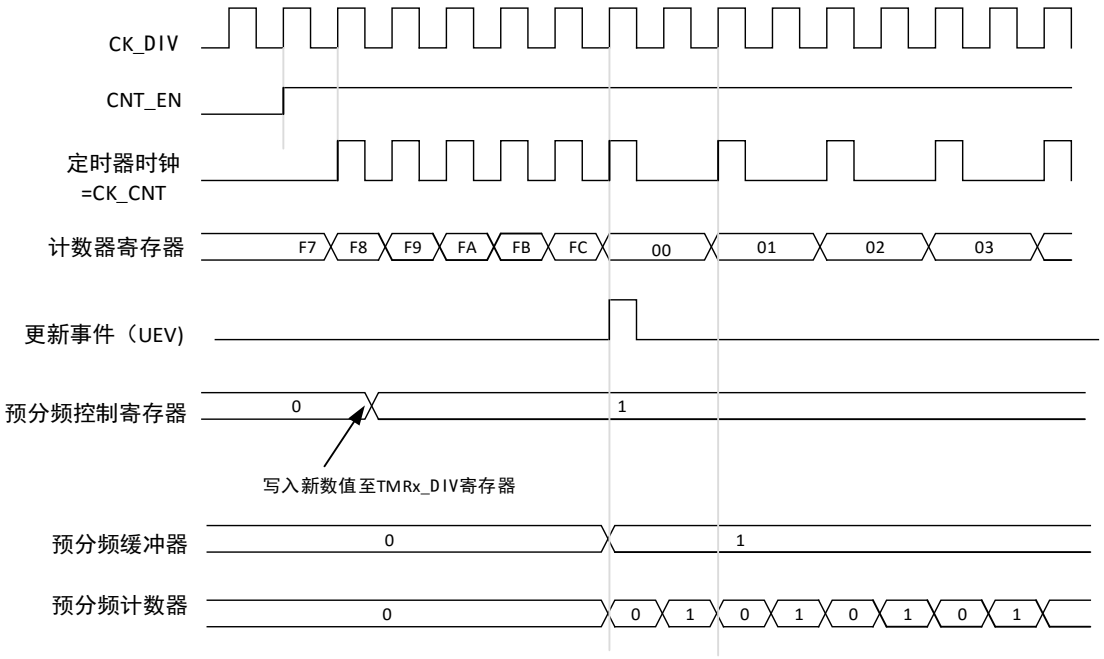
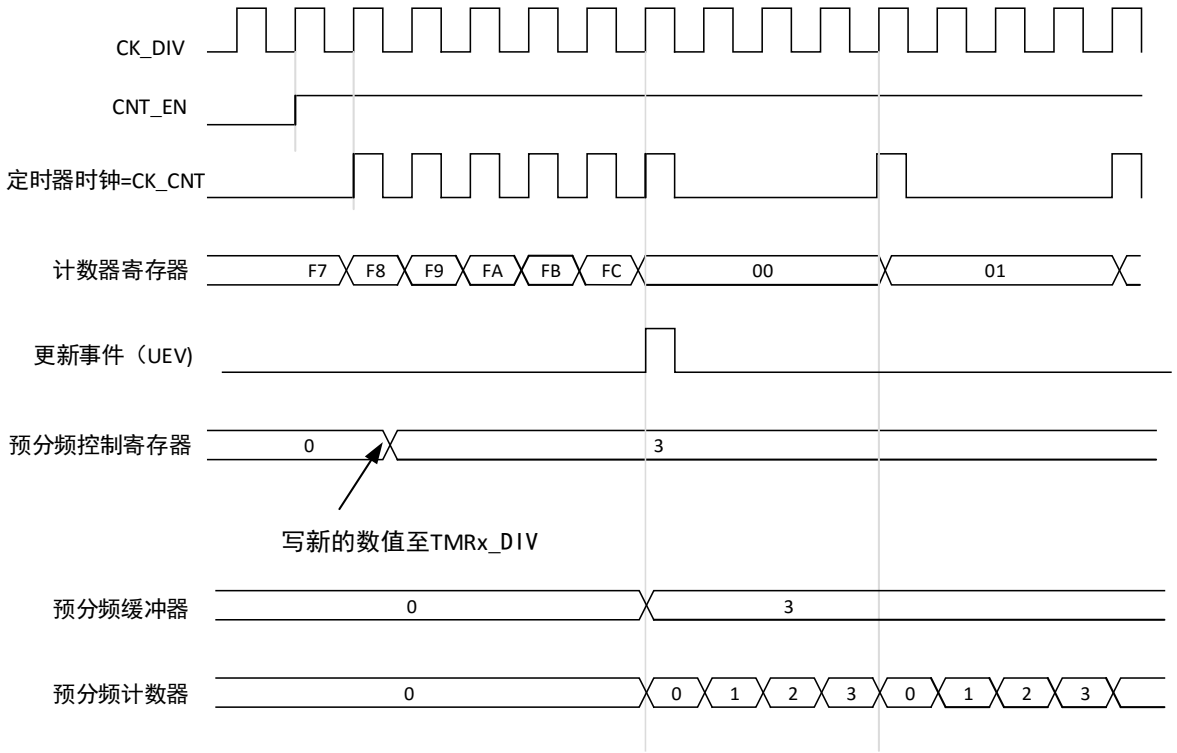


图 10-3 预分频系数从1变到4的计数器时序图



10.1.3.3 计数模式

计数器从 0 累加计数到自动重装载数值（TMRx_AR 寄存器），然后重新从 0 开始计数并产生一个计数器溢出事件。

每次计数器溢出时可以产生更新事件；（通过软件或使用从模式控制器）设置 TMRx_EVEG 寄存器的 UEVG 位也可以产生更新事件。

设置 TMRx_CTRL1 中的 UEVDIS 位可以禁止产生 UEV 事件，这可以避免在写入预加载寄存器时更改影子寄存器。在清除 UEVDIS 位为’0’之前，将不再产生更新事件，但计数器和预分频器依然会在应产生更新事件时重新从 0 开始计数(但预分频系数不变)。另外，如果设置了 TMRx_CTRL1 寄存器中的 UVERS（选择更新请求），设置 UEVG 位可以产生一次更新事件 UEV，但不设置 UEVIF 标志（即没有中断或 DMA 请求）。

当发生一次更新事件时，所有寄存器会被更新并（根据 UVERS 位）设置更新标志（TMRx_STS 寄存器的 UEVIF 位）：

- 传送预装载值（TMRx_DIV寄存器的内容）至预分频器的缓冲区。
- 自动重装载影子寄存器被更新为预装载值（TMRx_AR）。

以下是一些在 TMRx_AR=0x36 时不同时钟频率下计数器工作的图示例子。

图 10-4 计数器时序图，内部时钟分频系数为1

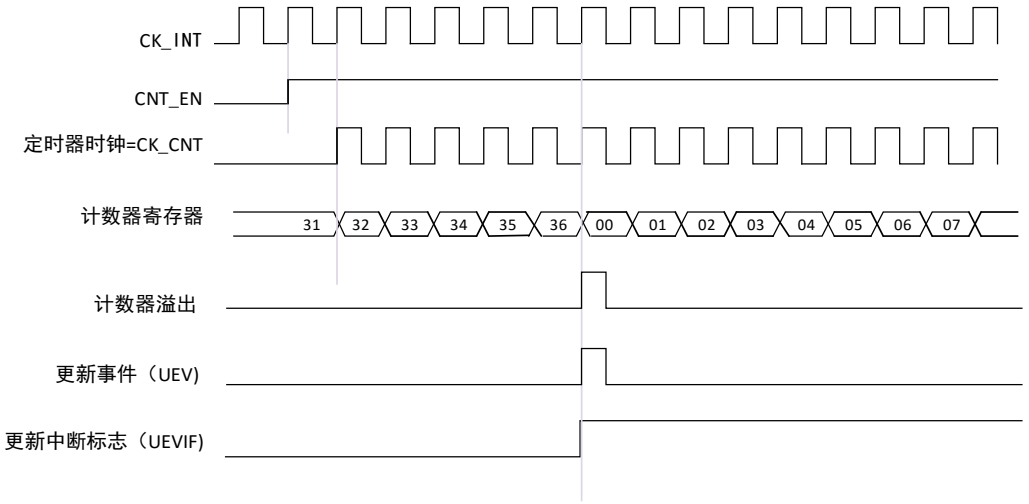


图 10-5 计数器时序图，内部时钟分频系数为2

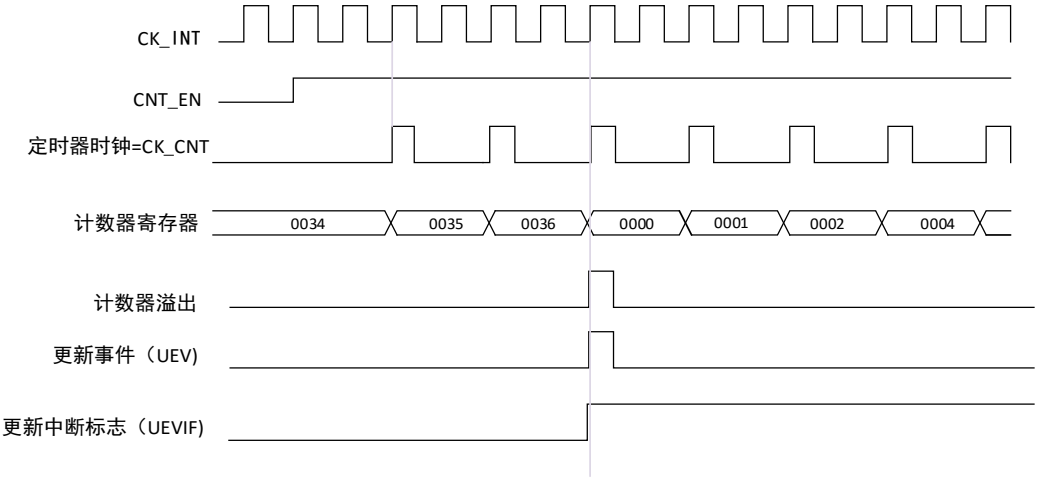


图 10-6 计数器时序图，内部时钟分频系数为4

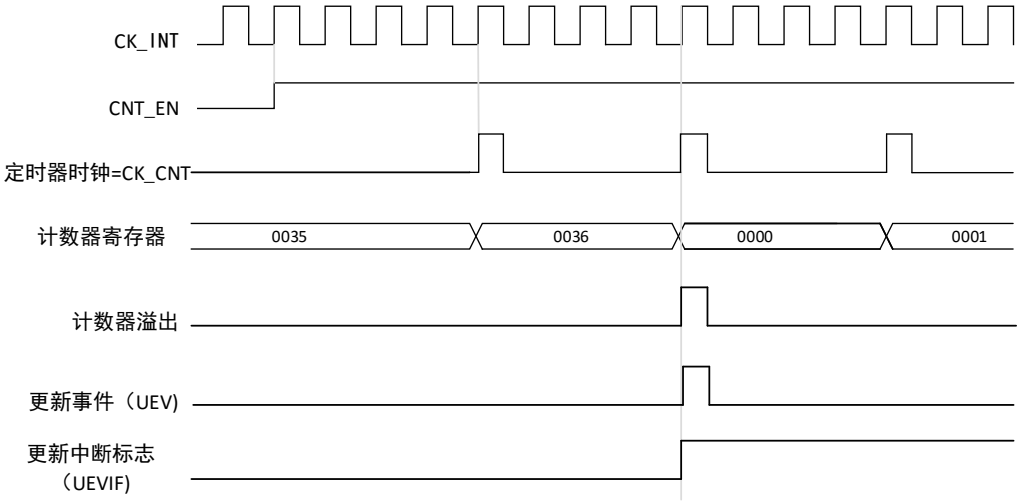


图 10-7 计数器时序图，内部时钟分频系数为N

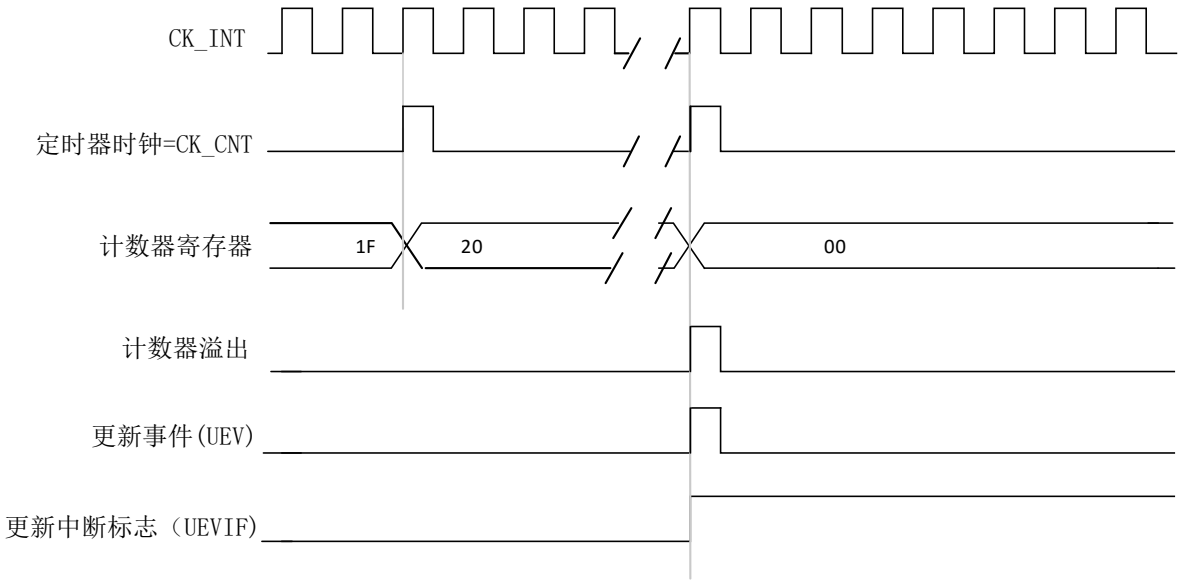


图 10-8 计数器时序图，当ARPEN=0时的更新事件（TMRx_AR没有预装载）

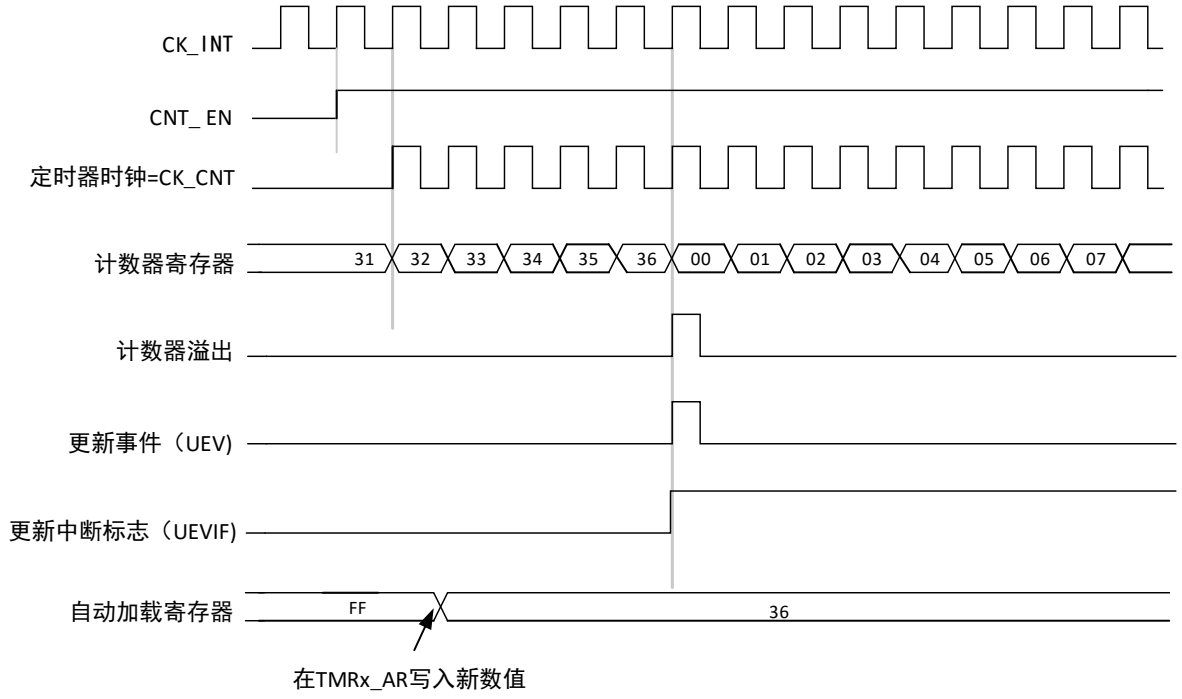
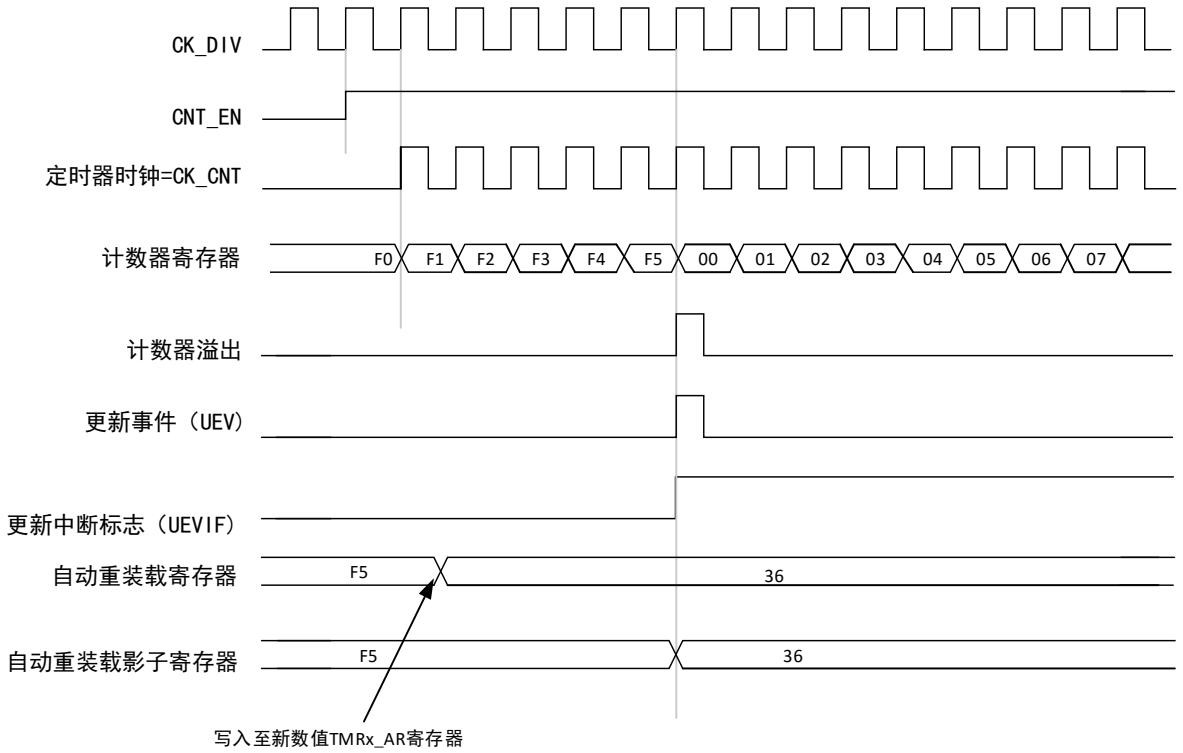


图 10-9 计数器时序图，当ARPEN=1时的更新事件（预装载TMRx_AR）



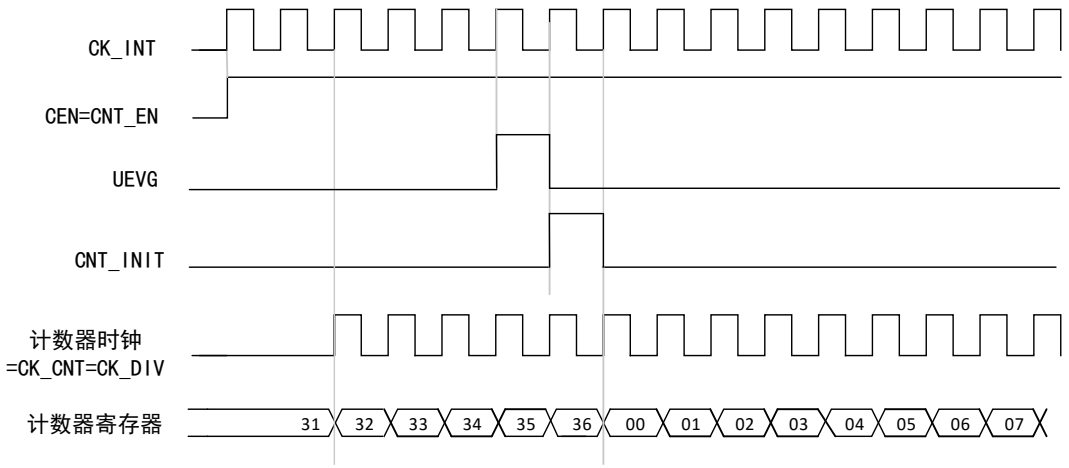
10.1.3.4 时钟源

计数器的时钟由内部时钟（CK_INT）提供。

TMRx_CTRL1 寄存器的 CNTEN 位和 TMRx_EVEG 寄存器的 UEVG 位是实际的控制位，（除了 UEVG 位被自动清除外）只能通过软件改变它们。一旦置 CNTEN 位为’1’，内部时钟即向预分频器提供时钟。

下图示出控制电路和向上计数器在普通模式下，没有预分频器时的操作。

图 10-10 普通模式时序图，内部时钟分频系数为1



10.1.3.5 调试模式

当微控制器进入调试模式（Cortex™-M4F 核心停止）时，根据 DBG 模块中的配置位 DBG_TMRx_STOP 的设置，TMRx 计数器或者继续计数或者停止工作。详见第 22.2.2 节。

10.1.4 TMR6和TMR7寄存器

可以用半字（16 位）或字（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 10-1 TMR6和TMR7 寄存器图和复位值

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	TMRx_CTRL1	保留																								ARPEN	保留				UVERS	UEVDIS	CNTEN
	复位值																									0					0	0	0
0x04	TMRx_CTRL2	保留																										MMSEL[2：0]			保留			
	复位值																											0	0	0				
0x0C	TMRx_DIE	保留																								UEVDE	保留						UEVIE
	复位值																									0							0
0x10	TMRx_STS	保留																															UEVIF
	复位值																																0
0x14	TMRx_EVEG	保留																															UEVG
	复位值																																0
0x24	TMRx_CNT	保留																CNT[15：0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x28	TMRx_DIV	保留																DIV[15：0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x2C	TMRx_AR	保留																AR[15：0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

10.1.4.1TMR6 和TMR7控制寄存器1（TMRx_CTRL1）

偏移地址：0x00

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留								ARP EN	保留			OPM ODE	UVE RS	UEV DIS	CNT EN
res								rw	res			rw	rw	rw	rw

位 15：8	保留，始终读为 0。
位 7	**ARPEN**：自动重装载预装载使能 （Auto-reload preload enable） 0：TMRx_AR 寄存器没有缓冲 1：TMRx_AR 寄存器具有缓冲
位 6：4	保留，始终读为 0。
位 3	**OPMODE**：单脉冲模式 （One-pulse mode） 0：在发生更新事件时，计数器不停止 1：在发生下次更新事件时，计数器停止计数（清除 CNTEN 位）。

位 2	**UVERS**：更新请求源 （Update request source） 该位由软件设置和清除，以选择 UEV 事件的请求源。 0：如果使能了中断或 DMA，以下任一事件可以产生一个更新中断或 DMA 请求： - 计数器溢出 - 设置 UEVG 位 - 通过从模式控制器产生的更新 1：如果使能了中断或 DMA，只有计数器上溢或下溢可以产生更新中断或 DMA 请求。
位 1	**UEVDIS**：禁止更新 （Update disable） 该位由软件设置和清除，以使能或禁止 UEV 事件的产生。 0：UEV 使能。更新事件（UEV）可以由下列事件产生： - 计数器溢出 - 设置 UEVG 位 - 通过从模式控制器产生的更新产生更新事件后，带缓冲的寄存器被加载为预加载数值。 1：禁止 UEV。不产生更新事件（UEV），影子寄存器保持它的内容（AR、DIV）。但是如果设置了 UEVG 位或从模式控制器产生了一个硬件复位，则计数器和预分频器将被重新初始化。
位 0	**CNTEN**：计数器使能 （Counter enable） 0：关闭计数器 1：使能计数器 *注：门控模式只能在软件已经设置了 CNTEN 位时有效，而触发模式可以自动地由硬件设置 CNTEN 位。在单脉冲模式下，当产生更新事件时 CNTEN 被自动清除。*

10.1.4.2TMR6和TMR7控制寄存器2（TMRx_CTRL2）

偏移地址：0x04

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留									MMSEL			保留			
res									rw	rw	rw	res			

位 15：7	保留，始终读为 0。
位 6：4	**MMSEL**：主模式选择 （Master mode selection） 这些位用于选择在主模式下向从定时器发送的同步信息（TRGO），有以下几种组合： 000：复位 – 使用 TMRx_EVEG 寄存器的 UEVG 位作为触发输出（TRGO）。如果触发输入产生了复位（从模式控制器配置为复位模式），则相对于实际的复位信号，TRGO 上的信号有一定的延迟。 001：使能 – 计数器使能信号 CNT_EN 被用作为触发输出（TRGO）。它可用于在同一时刻启动多个定时器，或控制使能从定时器的时机。计数器使能信号是通过 CNTEN 控制位和配置为门控模式时的触发输入的'逻辑或'产生。 当计数器使能信号是通过触发输入控制时，在 TRGO 输出上会有一些延迟，除非选择了主/从模式（见 TMRx_SMC 寄存器的 MSMODE 位）。 010：更新 – 更新事件被用作为触发输出（TRGO）。例如一个主定时器可以作为从定时器的预分频器使用。
位 3：0	保留，始终读为 0。

10.1.4.3 TMR6 和TMR7 DMA/中断使能寄存器（TMRx_DIE）

偏移地址：0x0C

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留							UEVDE	保留							UEVIE
res							rw	res							rw

位 15：9	保留，始终读为 0。
位 8	**UEVDE：**更新 DMA 请求使能 （Update DMA request enable） 0：禁止更新 DMA 请求 1：使能更新 DMA 请求
位 7：1	保留，始终读为 0。
位 0	**UEVIE：**更新中断使能 （Update interrupt enable） 0：禁止更新中断 1：使能更新中断

10.1.4.4 TMR6和TMR7状态寄存器（TMRx_STS）

偏移地址：0x10

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留															UEVIF
res															rc_w0

位 15：1	保留，始终读为 0。
位 0	**UEVIF：**更新中断标志（Update interrupt flag） 硬件在更新中断时设置该位，它由软件清除。 0：没有产生更新。 1：产生了更新中断。下述情况下由硬件设置该位： – 计数器产生溢出并且 TMRx_CTRL1 中的 UEVDIS=0； – 如果 TMRx_CTRL1 中的 UVERS=0 并且 UEVDIS=0，置位 UEVG（TMRx_EVEG）位重新初始化计数器时。

10.1.4.5 TMR6和TMR7事件产生寄存器（TMRx_EVEG）

偏移地址：0x14

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留															UEVG
res															rc_w0

位 15：1	保留，始终读为 0。
位 0	**UEVG**：产生更新事件 （Update generation） 该位由软件设置，由硬件自动清除。 0：无作用 1：重新初始化定时器的计数器并产生对寄存器的更新。注意：预分频器也被清除（但预分频系数不变）。

10.1.4.6 TMR6和TMR7计数器（TMRx_CNT）

偏移地址：0x24

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT[15：0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 15：0	**CNT[15：0]**：计数器数值 （Counter value）

10.1.4.7 TMR6和TMR7预分频器（TMRx_DIV）

偏移地址：0x28

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DIV[15：0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 15：0	**DIV[15：0]**：预分频器数值 （Prescaler value） 计数器的时钟频率 CK_CNT 等于 fCK_DIV/（DIV[15：0]+1）。 在每一次更新事件时，DIV 的数值被传送到实际的预分频寄存器中。

10.1.4.8 TMR6和TMR7自动重装载寄存器（TMRx_AR）

偏移地址：0x2C

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AR[15：0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 15：0	**AR[15：0]**：自动重装载数值 （Auto reload value） AR 的数值将传送到实际的自动重装载寄存器中。 关于 AR 的更新和作用，详见 10.1.3.1 节。 如果自动重装载数值为 0，则计数器停止。

10.2 通用定时器（TMR2到TMR5）

10.2.1 TMRx简介

通用定时器是一个通过可编程预分频器驱动的16位自动装载计数器构成。它适用于多种场合，包括测量输入信号的脉冲长度（输入捕获）或者产生输出波形（输出比较和PWM）。

使用定时器预分频器和RCC时钟控制器预分频器，脉冲长度和波形周期可以在几个微秒到几个毫秒间调整。

每个定时器都是完全独立的，没有互相共享任何资源。它们可以一起同步操作，参见10.2.3.15节。

10.2.2 TMRx主要功能

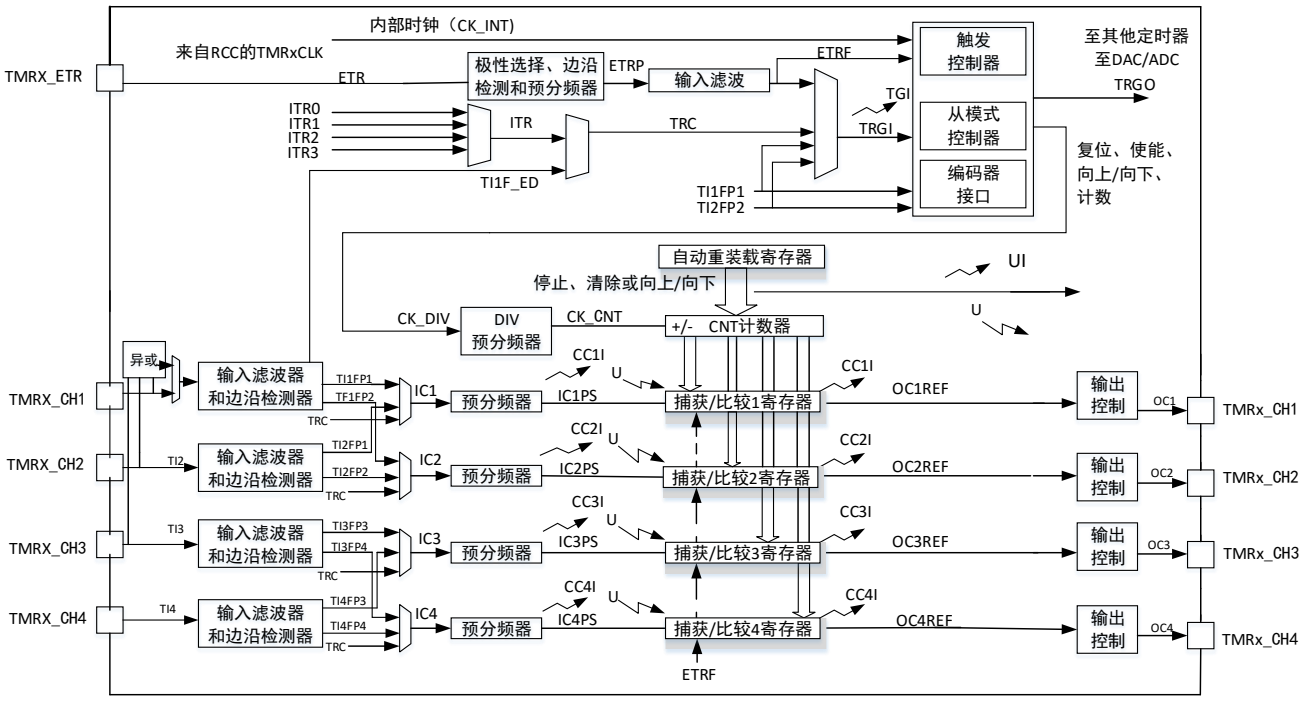
通用TMRx（TMR2、TMR3、TMR4和TMR5）定时器功能包括：

- 16位向上、向下、向上/向下自动装载计数器

注意： *TMR2和TMR5可选32位，详见10.2.4.1节控制寄存器1（TMRx_CTRL1）*

- 16位可编程（可以实时修改）预分频器，计数器时钟频率的分频系数为1～65536之间的任意数值
- 4个独立通道：
  - 输入捕获
  - 输出比较
  - PWM生成（边缘或中间对齐模式）
  - 单脉冲模式输出
- 使用外部信号控制定时器和定时器互连的同步电路
- 如下事件发生时产生中断/DMA：
  - 更新：计数器向上溢出/向下溢出，计数器初始化（通过软件或者内部/外部触发）
  - 触发事件（计数器启动、停止、初始化或者由内部/外部触发计数）
  - 输入捕获
  - 输出比较
- 支持针对定位的增量（正交）编码器和霍尔传感器电路
- 触发输入作为外部时钟或者按周期的电流管理

图 10-11 通用定时器框图



注意： *根据控制位的设定，在 U 事件时传送预加载寄存器的内容至工作寄存器*

事件

中断和 DMA 输出

10.2.3 TMRx功能描述

10.2.3.1 时基单元

可编程通用定时器的主要部分是一个 16 位计数器和与其相关的自动装载寄存器。这个计数器可以向上计数、向下计数或者向上向下双向计数。此计数器时钟由预分频器分频得到。

计数器、自动装载寄存器和预分频器寄存器可以由软件读写，在计数器运行时仍可以读写。

时基单元包含：

- 计数器寄存器（TMRx_CNT）
- 预分频器寄存器（TMRx_DIV）
- 自动装载寄存器（TMRx_AR）

自动装载寄存器是预先装载的，写或读自动重装载寄存器将访问预装载寄存器。根据在 TMRx_CTRL1 寄存器中的自动装载预装载使能位（ARPEN）的设置，预装载寄存器的内容被立即或在每次的更新事件 UEV 时传送到影子寄存器。当计数器达到溢出条件（向下计数时的下溢条件）并当 TMRx_CTRL1 寄存器中的 UEVDIS 位等于'0'时，产生更新事件。更新事件也可以由软件产生。随后会详细描述每一种配置下更新事件的产生。

计数器由预分频器的时钟输出 CK_CNT 驱动，仅当设置了计数器 TMRx_CTRL1 寄存器中的计数器使能位（CNTEN）时，CK_CNT 才有效。（有关计数器使能的细节，请参见控制器的从模式描述）。

注意： *真正的计数器使能信号 CNT_EN 是在 CNTEN 的一个时钟周期后被设置。*

预分频器

预分频器可以将计数器的时钟频率按 1 到 65536 之间的任意值分频。它是基于一个（在 TMRx_DIV 寄存器中的）16 位寄存器控制的 16 位计数器。这个控制寄存器带有缓冲器，它能够在工作时被改变。新的预分频器参数在下一次更新事件到来时被采用。

图 10-12 和图 10-13 给出了在预分频器运行时，更改计数器参数的例子。

图 10-12 当预分频器的参数从1变到2时，计数器的时序图

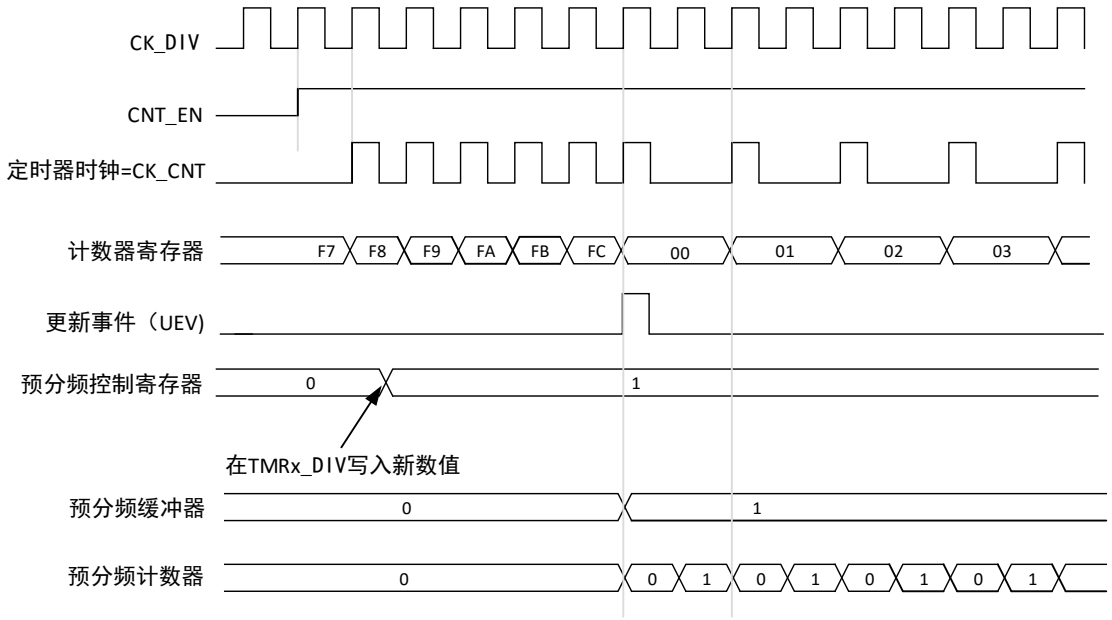
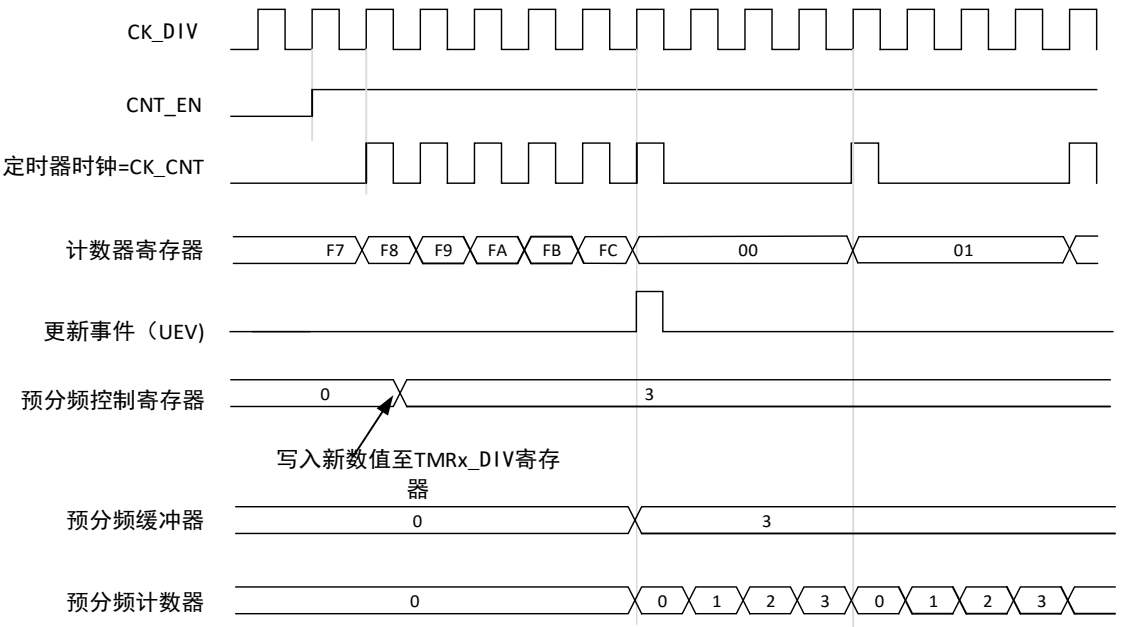


图 10-13 当预分频器的参数从1变到4时，计数器的时序图



10.2.3.2 计数器模式

向上计数模式

在向上计数模式中，计数器从 0 计数到自动加载值（TMRx_AR 计数器的内容），然后重新从 0 开始计数并且产生一个计数器溢出事件。

每次计数器溢出时可以产生更新事件，在 TMRx_EVEG 寄存器中（通过软件方式或者使用从模式控制器）设置 UEVG 位也同样可以产生一个更新事件。

设置 TMRx_CTRL1 寄存器中的 UEVDIS 位，可以禁止更新事件；这样可以避免在向预装载寄存器中写入新值时更新影子寄存器。在 UEVDIS 位被清'0'之前，将不产生更新事件。但是在应该产生更新事件时，计数器仍会被清'0'，同时预分频器的计数也被清 0（但预分频系数不变）。此外，如果设置了 TMRx_CTRL1 寄存器中的 UEVRS 位（选择更新请求），设置 UEVG 位将产生一个更新事件 UEV，但硬件不设置 UEVIF 标志（即不产生中断或 DMA 请求）；这是为了避免在捕获模式下清除计数器时，同时产生更新和捕获中断。

当发生一个更新事件时，所有的寄存器都被更新，硬件同时（依据 UEVRS 位）设置更新标志位（TMRx_STS

寄存器中的 UEVIF 位）。

- 预分频器的缓冲区被置入预装载寄存器的值（TMRx_DIV寄存器的内容）。
- 自动装载影子寄存器被重新置入预装载寄存器的值（TMRx_AR）。

下图给出一些例子，当 TMRx_AR=0x36 时计数器在不同时钟频率下的动作。

图 10-14 计数器时序图，内部时钟分频因子为1

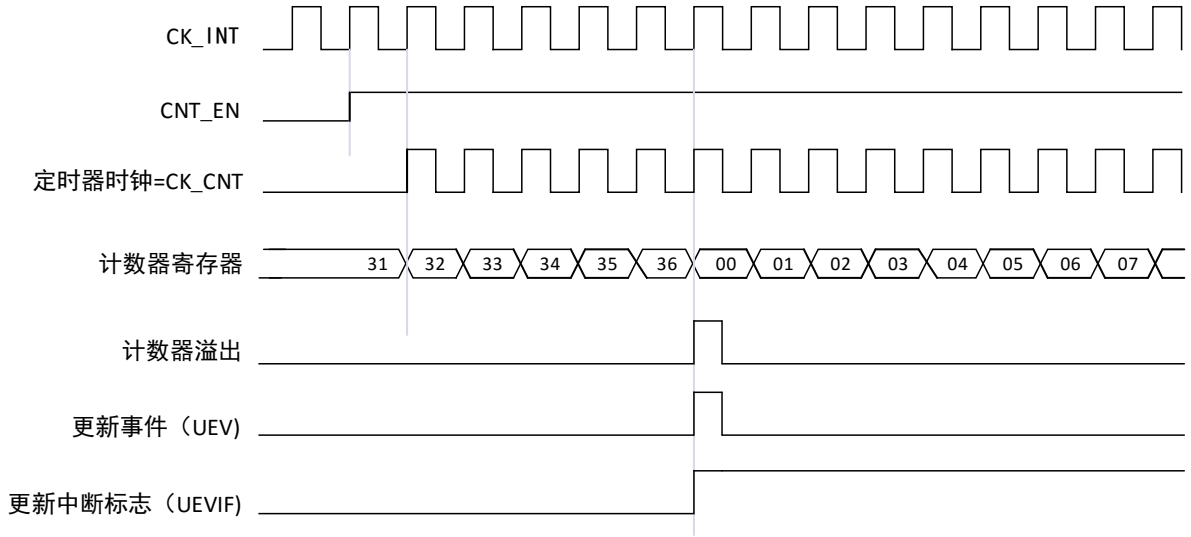


图 10-15 计数器时序图，内部时钟分频因子为2

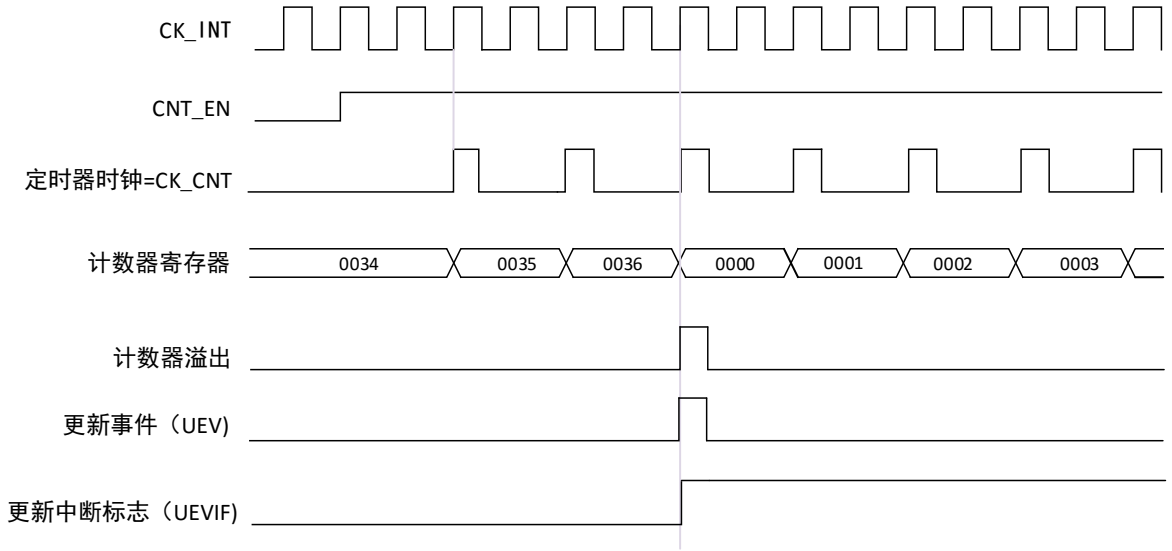


图 10-16 计数器时序图，内部时钟分频因子为4

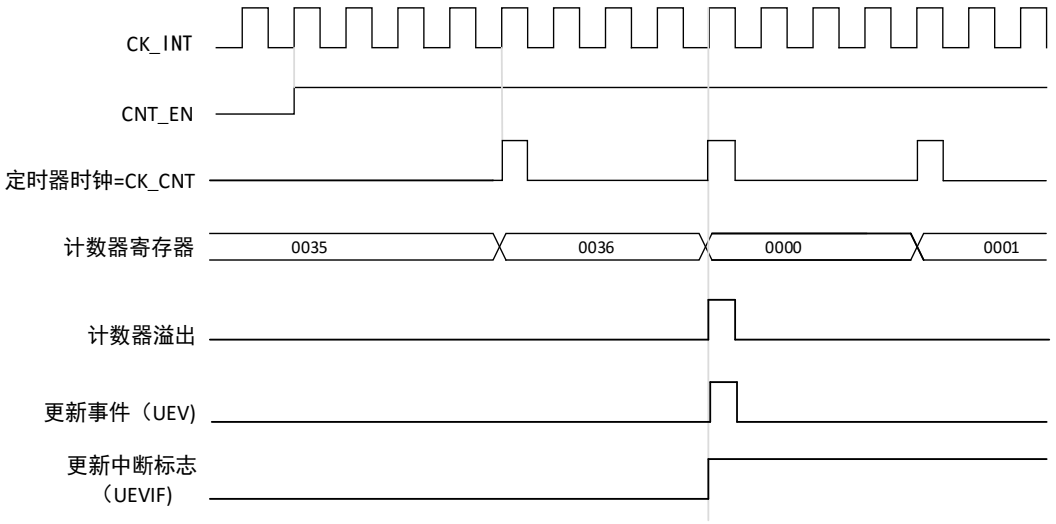


图 10-17 计数器时序图，内部时钟分频因子为N

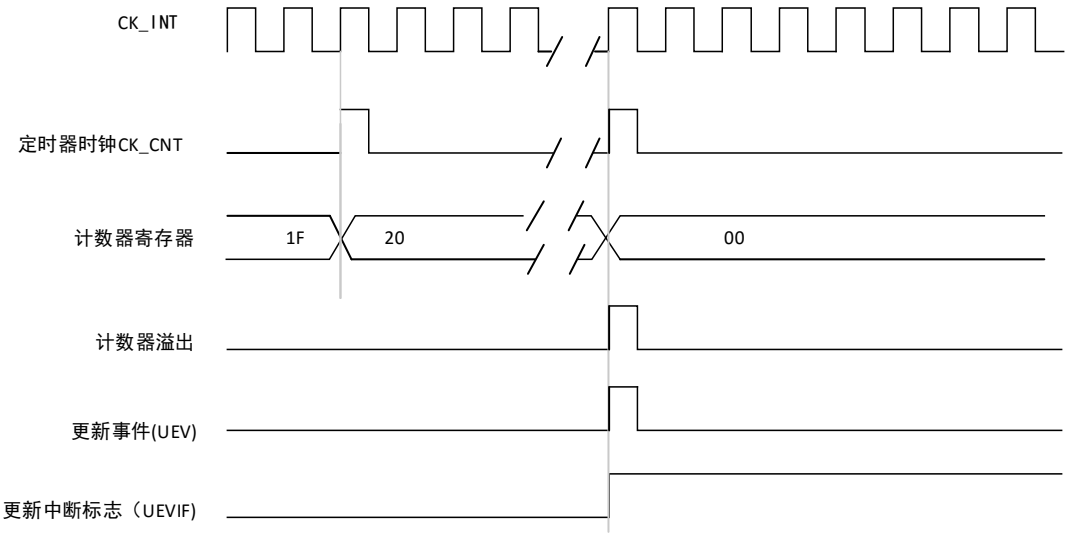


图 10-18 计数器时序图，当ARPEN=0时的更新事件（没有预装入了TMRx_AR）

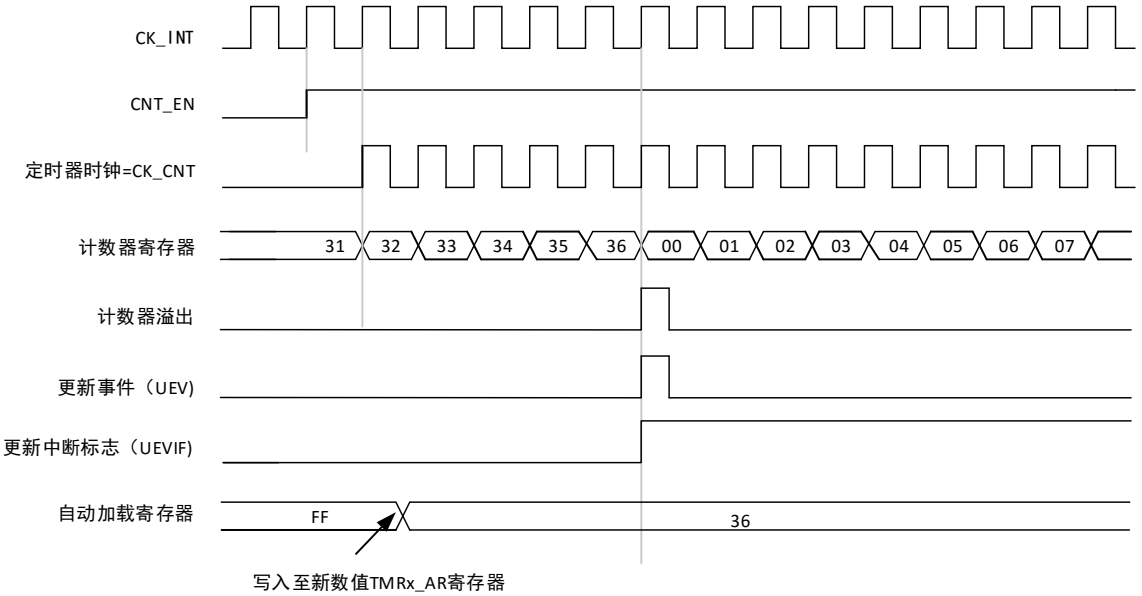
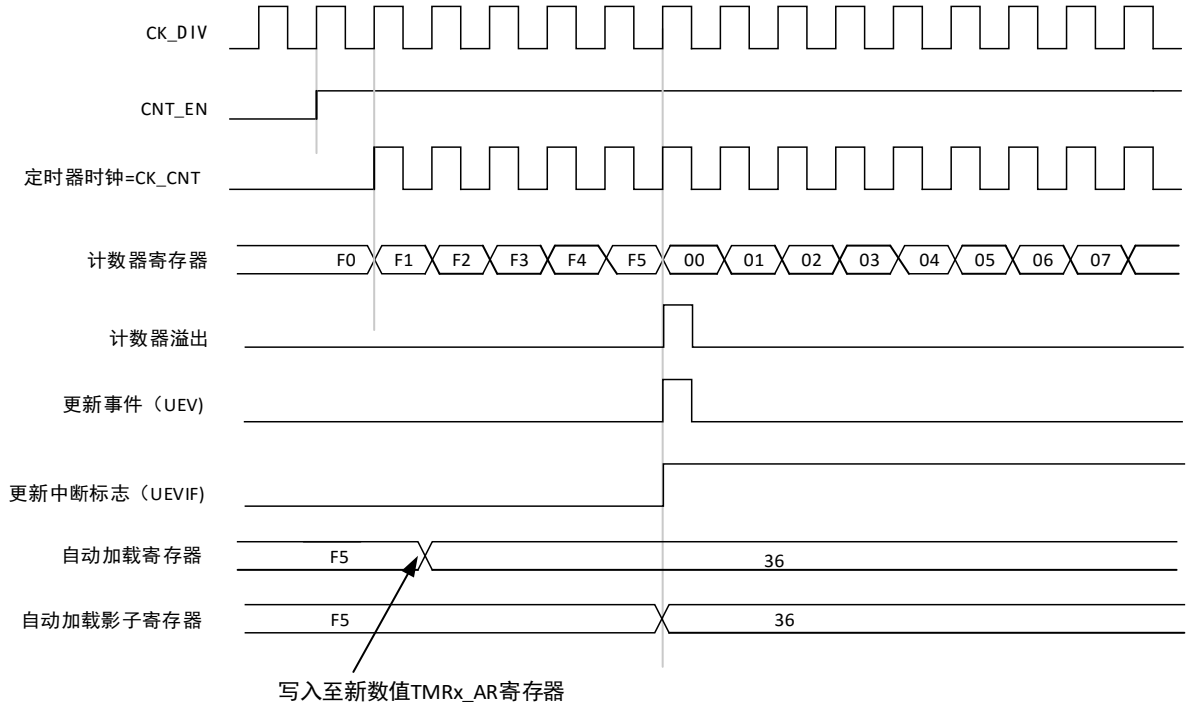


图 10-19 计数器时序图，当ARPEN=1时的更新事件（预装入了TMRx_AR）



向下计数模式

在向下模式中，计数器从自动装入的值（TMRx_AR 计数器的值）开始向下计数到 0，然后从自动装入的值重新开始并且产生一个计数器向下溢出事件。

每次计数器溢出时可以产生更新事件，在 TMRx_EVEG 寄存器中（通过软件方式或者使用从模式控制器）设置 UEVG 位，也同样可以产生一个更新事件。

设置 TMRx_CTRL1 寄存器的 UEVDIS 位可以禁止 UEV 事件。这样可以避免向预装载寄存器中写入新值时更新影子寄存器。因此 UEVDIS 位被清为'0'之前不会产生更新事件。然而，计数器仍会从当前自动加载值重新开始计数，同时预分频器的计数器重新从 0 开始（但预分频系数不变）。此外，如果设置了 TMRx_CTRL1 寄存器中的 UEVRS 位（选择更新请求），设置 UEVG 位将产生一个更新事件 UEV 但不设置 UEVIF 标志（因此不产生中断和 DMA 请求），这是为了避免在发生捕获事件并清除计数器时，同时产生更新和捕获中断。

当发生更新事件时，所有的寄存器都被更新，并且（根据 UEVRS 位的设置）更新标志位（TMRx_STS 寄存器中的 UEVIF 位）也被设置。

- 预分频器的缓存器被置入预装载寄存器的值（TMRx_DIV寄存器的值）。
- 当前的自动加载寄存器被更新为预装载值（TMRx_AR寄存器中的内容）。

注意：　自动装载在计数器重载入之前被更新，因此下一个周期将是预期的值。

以下是一些当 TMRx_AR=0x36 时，计数器在不同时钟频率下的操作例子。

图 10-20 计数器时序图，内部时钟分频因子为1

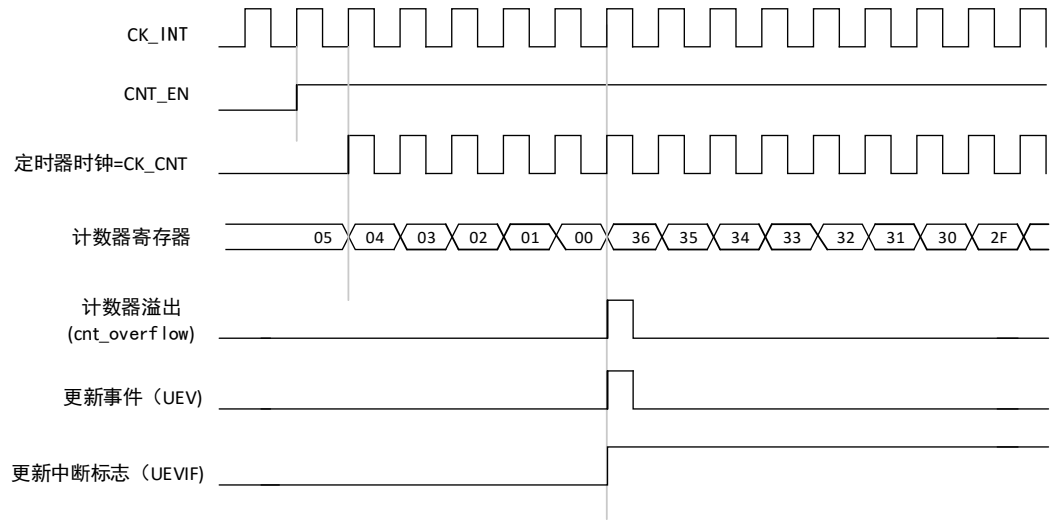


图 10-21 计数器时序图，内部时钟分频因子为2

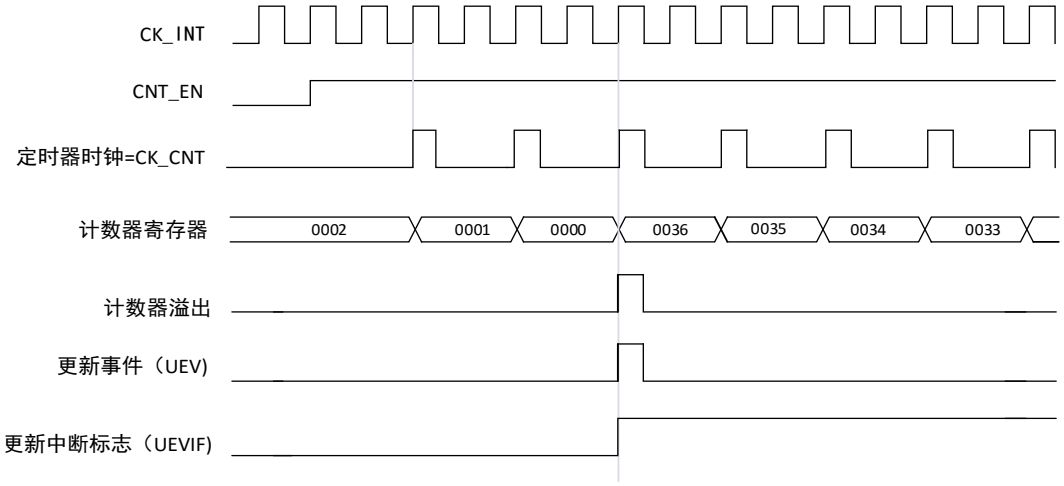


图 10-22 计数器时序图，内部时钟分频因子为4

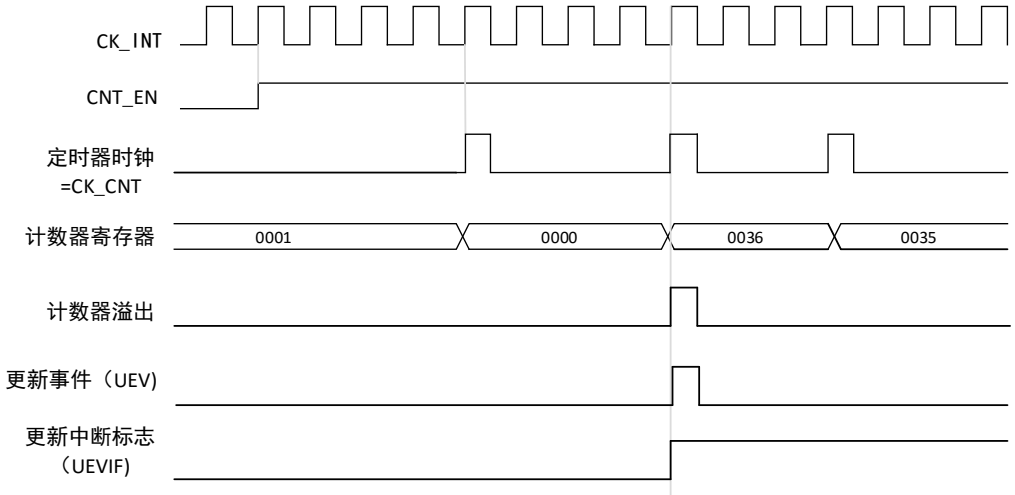


图 10-23 计数器时序图，内部时钟分频因子为N

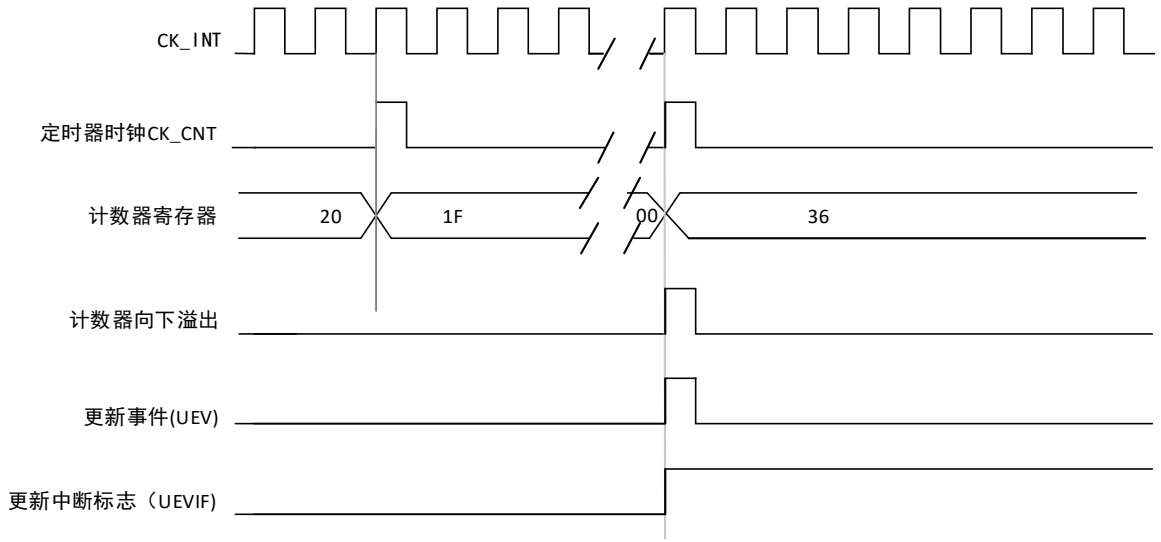
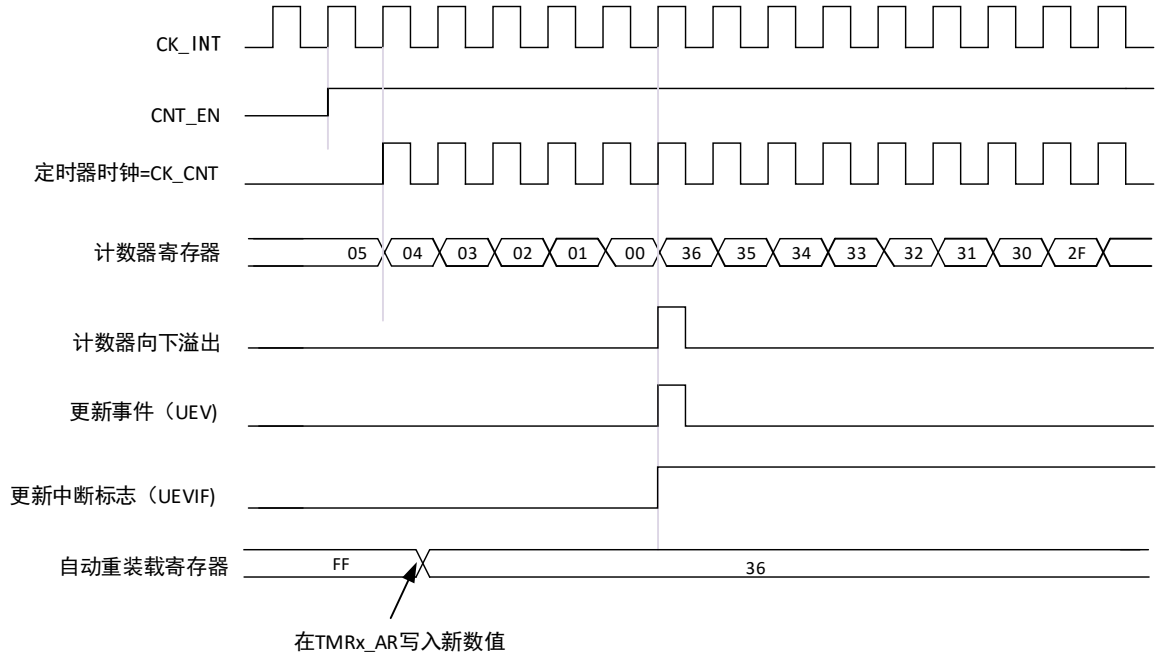


图 10-24 计数器时序图，当ARPEN=0时的更新事件



中央对齐模式（向上/向下计数）

在中央对齐模式，计数器从 0 开始计数到自动加载的值（TMRx_AR 寄存器）−1，产生一个计数器溢出事件，然后向下计数到 1 并且产生一个计数器下溢事件；然后再从 0 开始重新计数。

在这个模式，不能写入 TMRx_CTRL1 中的 DIR 方向位。它由硬件更新并指示当前的计数方向。

可以在每次计数上溢和每次计数下溢时产生更新事件；也可以通过（软件或者使用从模式控制器）。设置 TMRx_EVEG 寄存器中的 UEVG 位产生更新事件。然后，计数器重新从 0 开始计数，预分频器也重新从 0 开始计数。

设置 TMRx_CTRL1 寄存器中的 UEVDIS 位可以禁止 UEV 事件。这样可以避免在向预装载寄存器中写入新值时更新影子寄存器。因此 UEVDIS 位被清为'0'之前不会产生更新事件。然而，计数器仍会根据当前自动重加载的值，继续向上或向下计数。此外，如果设置了 TMRx_CTRL1 寄存器中的 UEVRS 位（选择更新请求），设置 UEVG 位将产生一个更新事件 UEV 但不设置 UEVIF 标志（因此不产生中断和 DMA 请求），这是为了避免在发生捕获事件并清除计数器时，同时产生更新和捕获中断。

当发生更新事件时，所有的寄存器都被更新，并且（根据 UEVRS 位的设置）更新标志位（TMRx_STS 寄存器中的 UEVIF 位）也被设置。

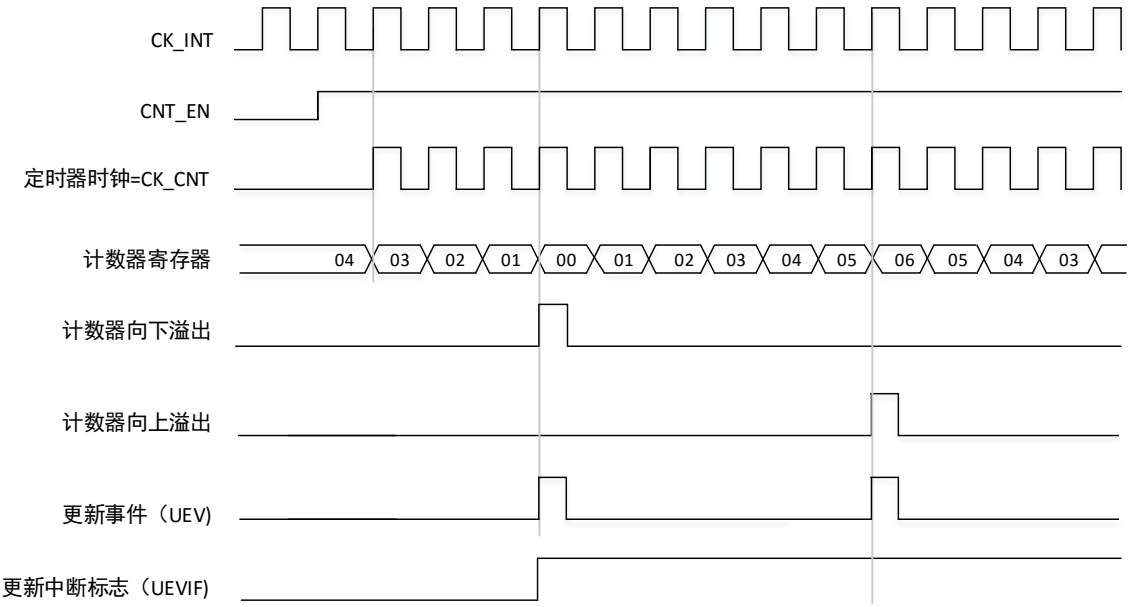
- 预分频器的缓存器被加载为预装载（TMRx_DIV寄存器）的值。

- 当前的自动加载寄存器被更新为预装载值（TMRx_AR寄存器中的内容）。

注意： *如果因为计数器溢出而产生更新，自动重装载将在计数器重载入之前被更新，因此下一个周期将是预期的值（计数器被装载为新的值）。*

以下是一些计数器在不同时钟频率下的操作的例子：

图 10-25 计数器时序图，内部时钟分频因子为1，TMRx_AR=0x6



注意： *这里使用了中心对齐模式1（寄存器的配置详见 10.2.4.1 节）。*

图 10-26 计数器时序图，内部时钟分频因子为2

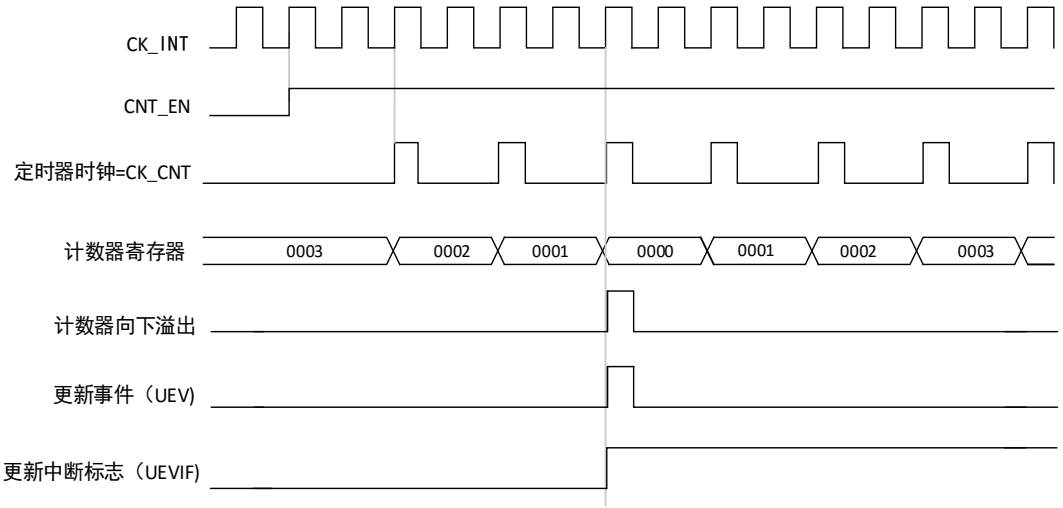


图 10-27 计数器时序图，内部时钟分频因子为4，TMRx_AR=0x36

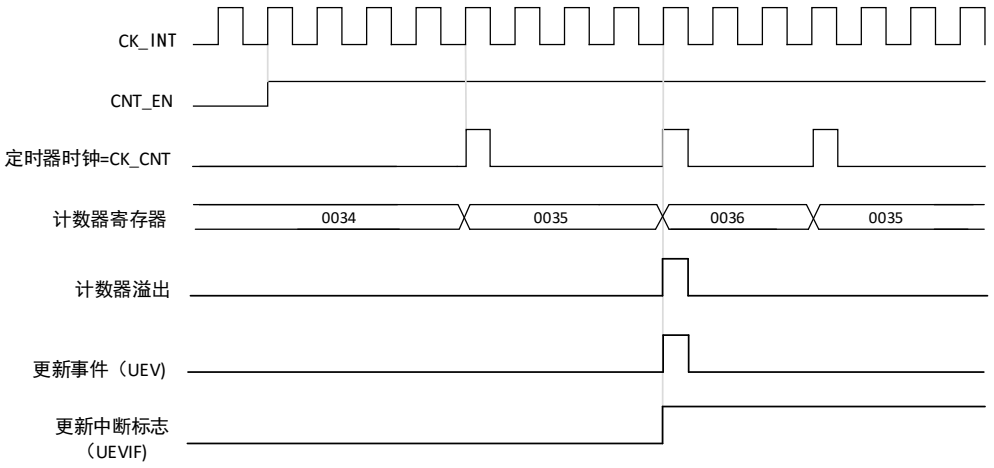


图 10-28 计数器时序图，内部时钟分频因子为N

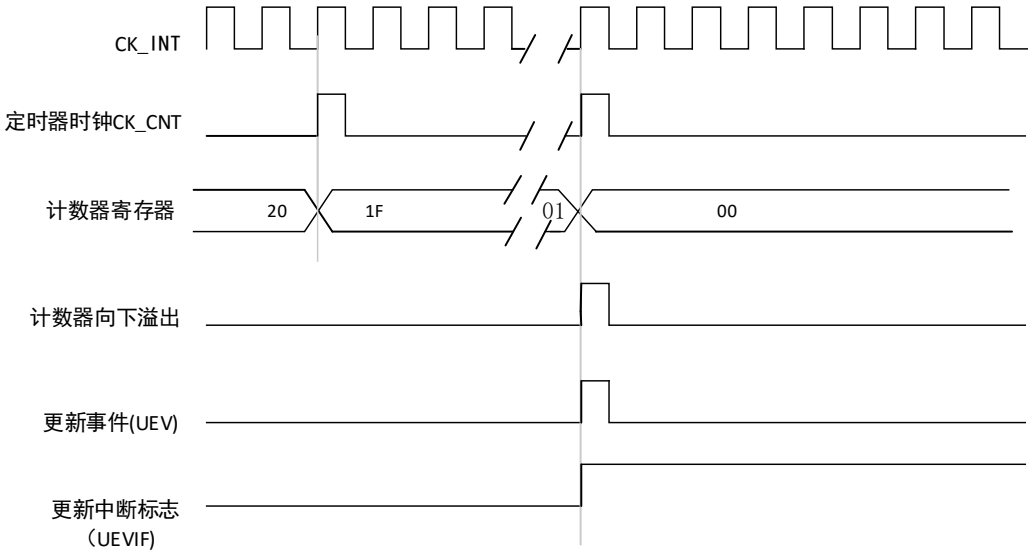


图 10-29 计数器时序图，ARPEN=1时的更新事件（计数器下溢）

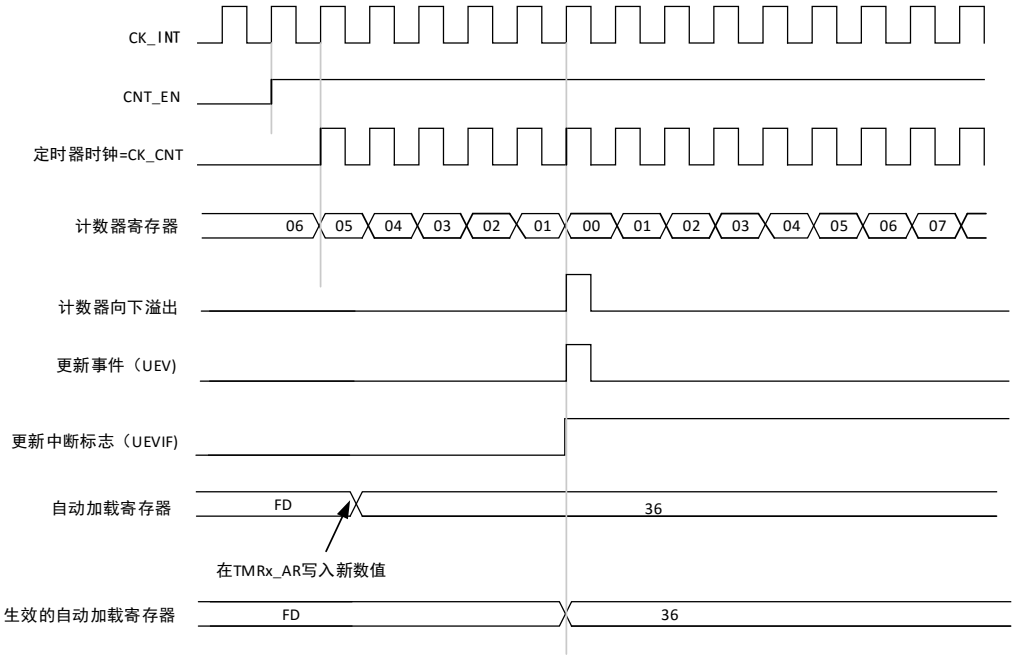
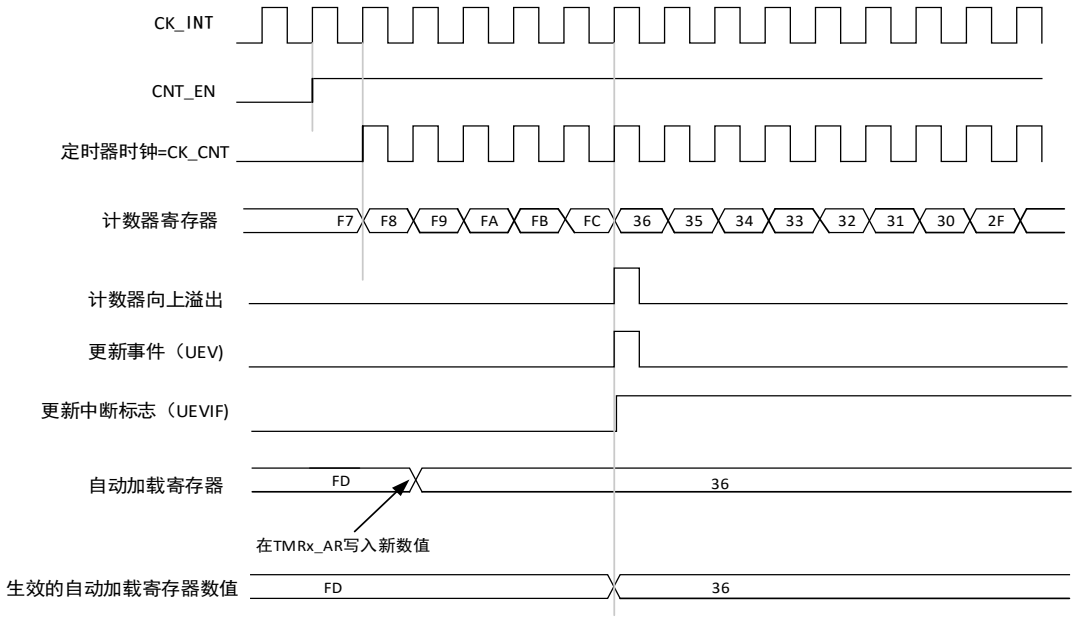


图 10-30 计数器时序图，ARPEN=1时的更新事件（计数器溢出）



10.2.3.3 时钟选择

计数器时钟可由下列时钟源提供：

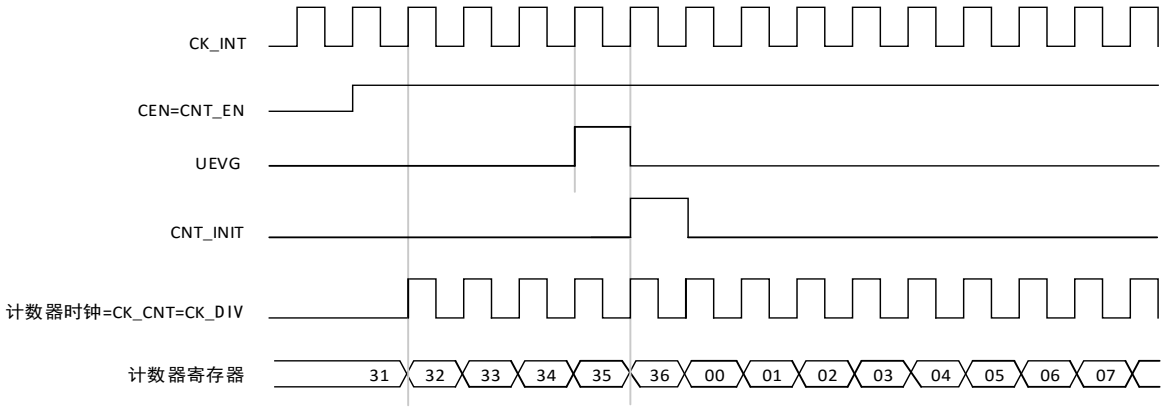
- 内部时钟（CK_INT）
- 外部时钟模式1：外部输入脚（TIx）
- 外部时钟模式2：外部触发输入（ETR）
- 内部触发输入（ITRx）：使用一个定时器作为另一个定时器的预分频器，如可以配置一个定时器Timer1而作为另一个定时器Timer2的预分频器。参见10.2.3.15。

内部时钟源（CK_INT）

如果禁止了从模式控制器（TMRx_SMC 寄存器的 SMSEL=000），则 CNTEN、DIR（TMRx_CTRL1 寄存器）和 UEVG 位（TMRx_EVEG 寄存器）是事实上的控制位，并且只能被软件修改（UEVG 位仍被自动清除）。只要 CNTEN 位被写成'1'，预分频器的时钟就由内部时钟 CK_INT 提供。

下图显示了控制电路和向上计数器在一般模式下，不带预分频器时的操作。

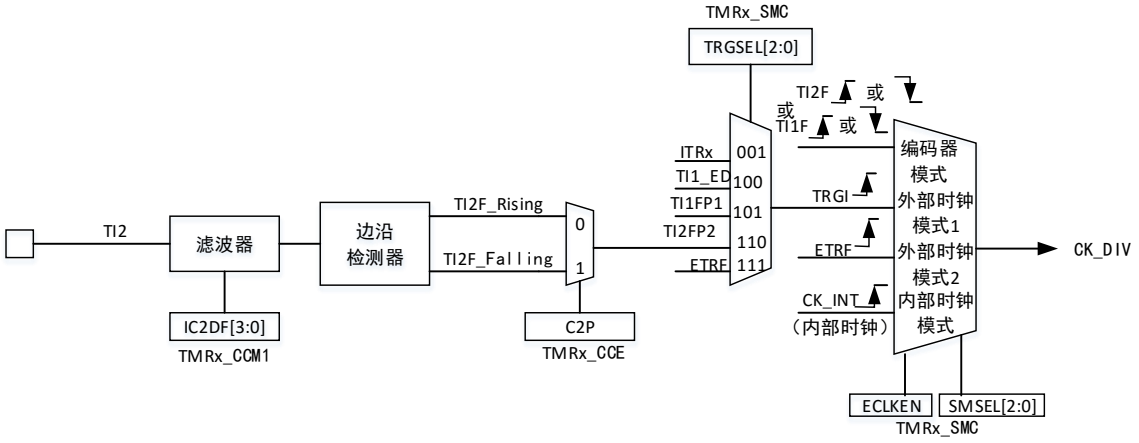
图 10-31 一般模式下的控制电路，内部时钟分频因子为1



外部时钟源模式 1

当 TMRx_SMC 寄存器的 SMSEL=111 时，此模式被选中。计数器可以在选定输入端的每个上升沿或下降沿计数。

图 10-32 TI2外部时钟连接例子



例如，要配置向上计数器在 T12 输入端的上升沿计数，使用下列步骤：

1. 配置 TMRx_CCM1 寄存器 C2SEL=’01’，配置通道 2 检测 TI2 输入的上升沿。
2. 配置 TMRx_CCM1 寄存器的 IC2DF[3：0]，选择输入滤波器带宽（如果不需要滤波器，保持 IC2DF=0000）。

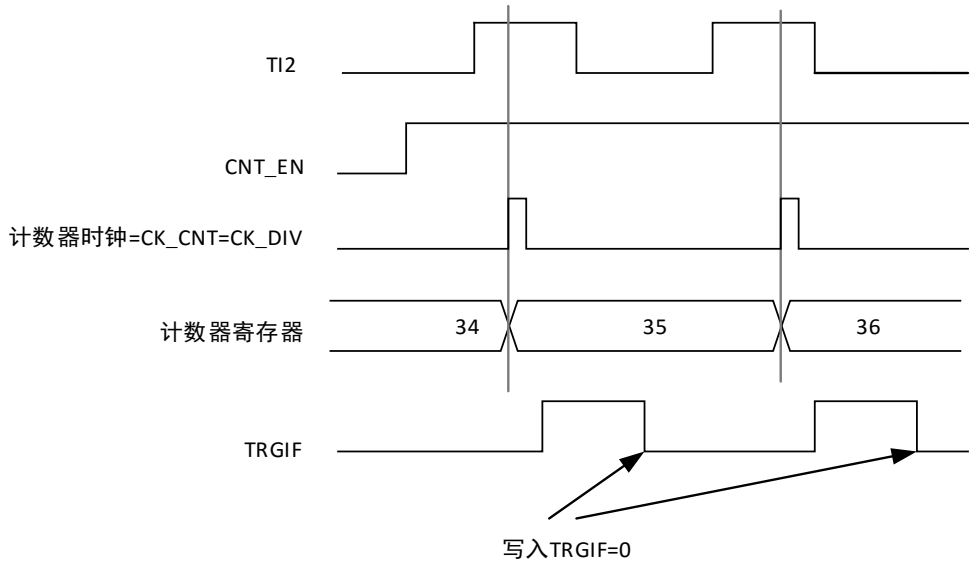
注意： 捕获预分频器不用作触发，所以不需要对它进行配置

3. 配置 TMRx_CCE 寄存器的 C2P=’0’，选定上升沿极性
4. 配置 TMRx_SMC 寄存器的 SMSEL=’111’，选择定时器外部时钟模式 1
5. 配置 TMRx_SMC 寄存器中的 TRGSEL=’110’，选定 TI2 作为触发输入源
6. 设置 TMRx_CTRL1 寄存器的 CNTEN=’1’，启动计数器

当上升沿出现在 TI2，计数器计数一次，且 TRGIF 标志被设置。

在 TI2 的上升沿和计数器实际时钟之间的延时，取决于在 TI2 输入端的重新同步电路。

图 10-33 外部时钟模式1下的控制电路



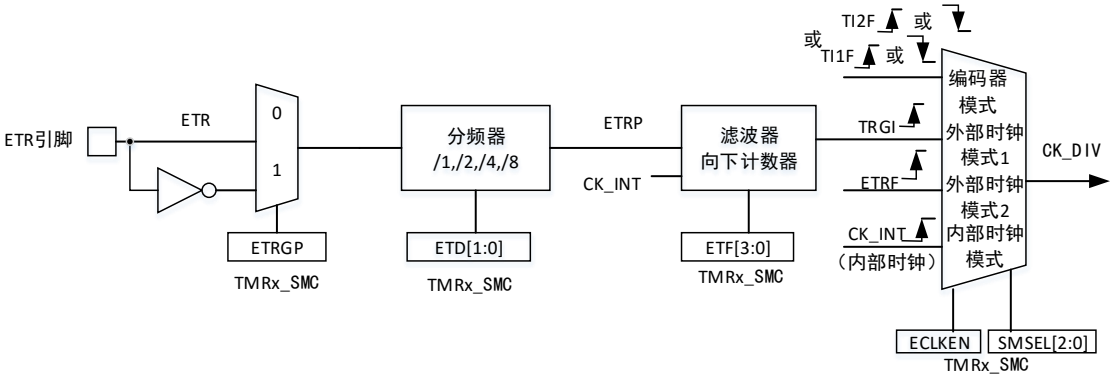
外部时钟源模式 2

选定此模式的方法为：令 TMRx_SMC 寄存器中的 ECLKEN=1

计数器能够在外部触发 ETR 的每一个上升沿或下降沿计数。

下图是外部触发输入的框图。

图 10-34 外部触发输入框图



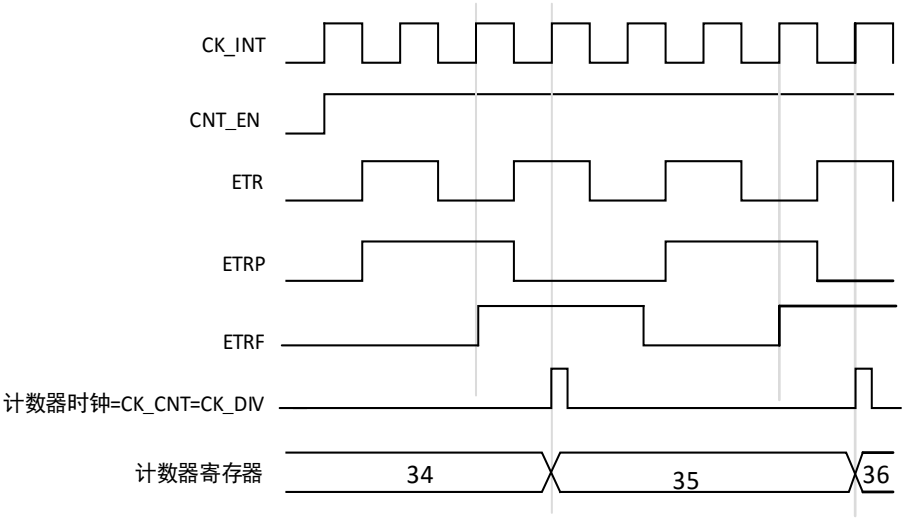
例如，要配置在 ETR 下每 2 个上升沿计数一次的向上计数器，使用下列步骤：

1. 本例中不需要滤波器，置 TMRx_SMC 寄存器中的 ETDF[3：0]=0000，
2. 设置预分频器，置 TMRx_SMC 寄存器中的 ETD[1：0]=01，
3. 设置在 ETR 的上升沿检测，置 TMRx_SMC 寄存器中的 ETRGP=0，
4. 开启外部时钟模式 2，置 TMRx_SMC 寄存器中的 ECLKEN=1，
5. 启动计数器，置 TMRx_CTRL1 寄存器中的 CNTEN=1

计数器在每 2 个 ETR 上升沿计数一次。

在 ETR 的上升沿和计数器实际时钟之间的延时取决于在 ETRP 信号端的重新同步电路。

图 10-35 外部时钟模式2下的控制电路

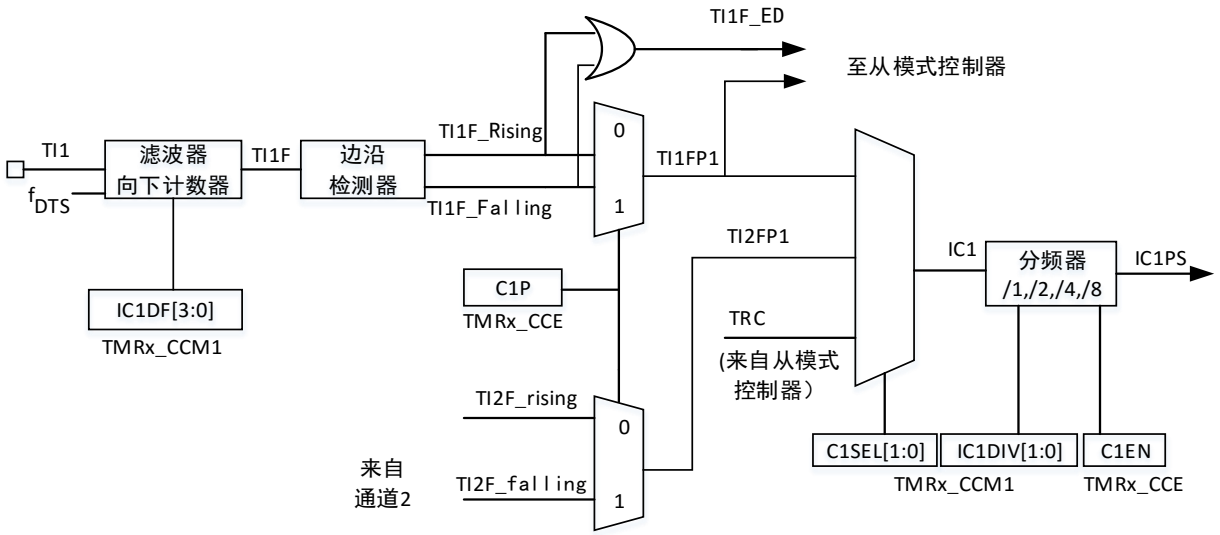


10.2.3.4 捕获/比较通道

每一个捕获/比较通道都是围绕着一个捕获/比较寄存器（包含影子寄存器），包括捕获的输入部分（数字滤波、多路复用和预分频器），和输出部分（比较器和输出控制）。下面几张图是一个捕获/比较通道概览。

输入部分对相应的 TIx 输入信号采样，并产生一个滤波后的信号 TIxF。然后，一个带极性选择的边缘检测器产生一个信号（TIxFPx），它可以作为从模式控制器的输入触发或者作为捕获控制。该信号通过预分频进入捕获寄存器。

图 10-36 捕获/比较通道（如：通道1输入部分）



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

图 10-37 捕获/比较通道1的主电路

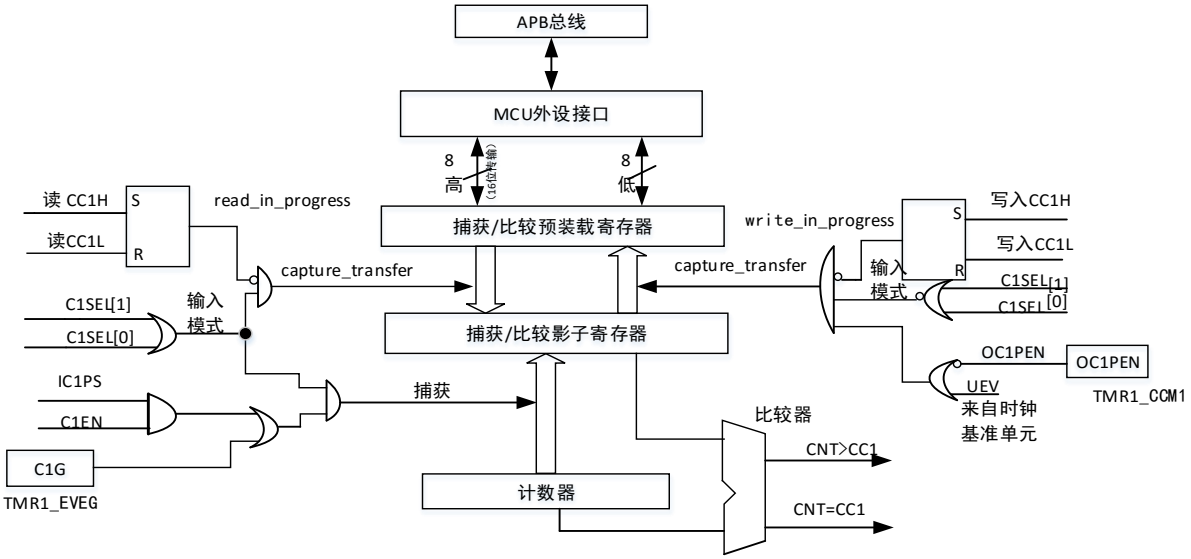
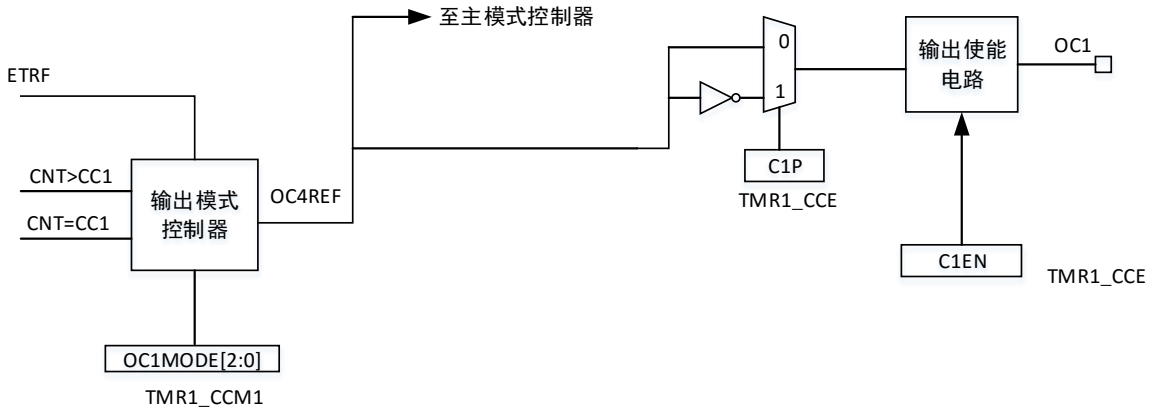


图 10-38 捕获/比较通道的输出部分（通道1）



捕获/比较模块由一个预装载寄存器和一个影子寄存器组成。读写过程仅操作预装载寄存器。

在捕获模式下，捕获发生在影子寄存器上，然后再复制到预装载寄存器中。

在比较模式下，预装载寄存器的内容被复制到影子寄存器中，然后影子寄存器的内容和计数器进行比较。

10.2.3.5 输入捕获模式

在输入捕获模式下，当检测到 ICx 信号上相应的边沿后，计数器的当前值被锁存到捕获/比较寄存器（TMRx_CCx）中。当捕获事件发生时，相应的 CxIF 标志（TMRx_STS 寄存器）被置'1'，如果使能了中断或者 DMA 操作，则将产生中断或者 DMA 操作。如果捕获事件发生时 CxIF 标志已经为高，那么重复捕获标志 CxOF（TMRx_STS 寄存器）被置'1'。写 CxIF=0 可清除 CxIF，或读取存储在 TMRx_CCx 寄存器中的捕获数据也可清除 CxIF。写 CxOF=0 可清除 CxOF。

以下例子说明如何在 TI1 输入的上升沿时捕获计数器的值到 TMRx_CC1 寄存器中，步骤如下：

- 选择有效输入端：TMRx_CC1必须连接到TI1输入，所以写入TMRx_CCM1寄存器中的C1SEL=01，只要C1SEL不为'00'，通道被配置为输入，并且TM1_CC1寄存器变为只读。
- 根据输入信号的特点，配置输入滤波器为所需的带宽（即输入为TIx时，输入滤波器控制位是TMRx_CCMx寄存器中的ICxDF位）。假设输入信号在最多5个内部时钟周期的时间内抖动，我们须配置滤波器的带宽长于5个时钟周期。因此我们可以（以f_{CK_INT}频率）连续采样8次，以确认在TI1上一次真实的边沿变换，即在TMRx_CCM1寄存器中写入IC1DF=0011。
- 选择TI1通道的有效转换边沿，在TMRx_CCE寄存器中写入C1P=0（上升沿）。
- 配置输入预分频器。在本例中，我们希望捕获发生在每一个有效的电平转换时刻，因此预分频器被禁止（写TMRx_CCM1寄存器的IC1DIV=00）。
- 设置TMRx_CCE寄存器的C1EN=1，允许捕获计数器的值到捕获寄存器中。
- 如果需要，通过设置TMRx_DIE寄存器中的C1IE位允许相关中断请求，通过设置TMRx_DIE寄存器中的C1DE位允许DMA请求。

当发生一个输入捕获时：

- 产生有效的电平转换时，计数器的值被传送到TMRx_CC1寄存器。
- C1IF标志被设置（中断标志）。当发生至少2个连续的捕获时，而C1IF未曾被清除，C1OF也被置'1'。
- 如设置了C1IE位，则会产生一个中断。
- 如设置了C1DE位，则还会产生一个DMA请求。

为了处理重复捕获，建议在读出重复捕获标志之前读取数据，这是为了避免丢失在读出重复捕获标志之后和读取数据之前可能产生的重复捕获信息。

注意： *设置 TMRx_EVEG 寄存器中相应的 CxG 位，可以通过软件产生输入捕获中断和/或 DMA 请求。*

10.2.3.6 PWM输入模式

该模式是输入捕获模式的一个特例，除下列区别外，操作与输入捕获模式相同：

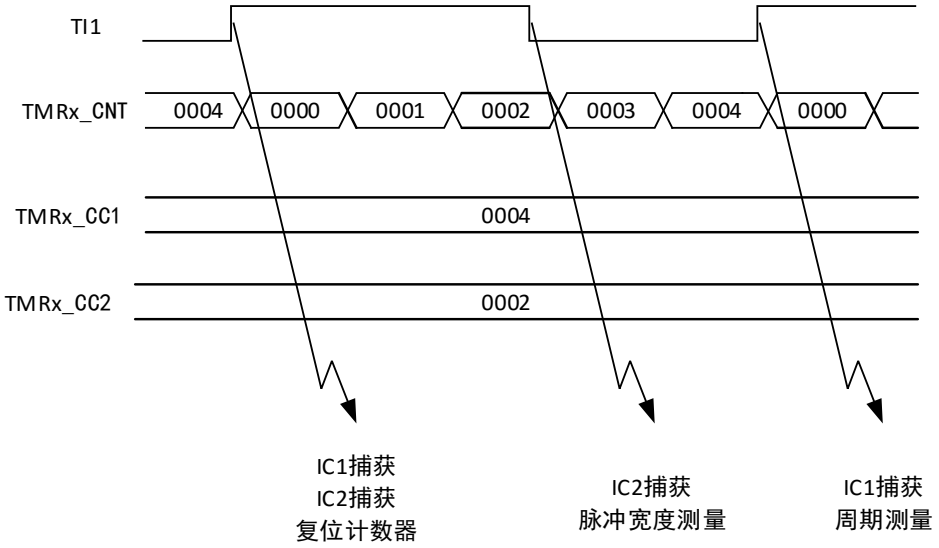
- 两个ICx信号被映射至同一个TIx输入。
- 这2个ICx信号为边沿有效，但是极性相反。
- 其中一个TIxFP信号被作为触发输入信号，而从模式控制器被配置成复位模式。

例如，你需要测量输入到TI1上的PWM信号的长度（TMRx_CC1寄存器）和占空比（TMRx_CC2寄存器），具体步骤如下（取决于CK_INT的频率和预分频器的值）。

- 选择TMRx_CC1的有效输入：置TMRx_CCM1寄存器的C1SEL=01（选择TI1）。
- 选择TI1FP1的有效极性（用来捕获数据到TMRx_CC1中和清除计数器）：置C1P=0（上升沿有效）。
- 选择TMRx_CC2的有效输入：置TMRx_CCM1寄存器的C2SEL=10（选择TI1）。

- 选择TI1FP2的有效极性（捕获数据到TMRx_CC2）：置C2P=1（下降沿有效）。
- 选择有效的触发输入信号：置TMRx_SMC寄存器中的TRGSEL=101（选择TI1FP1）。
- 配置从模式控制器为复位模式：置TMRx_SMC中的SMSEL=100。
- 使能捕获：置TMRx_CCE寄存器中C1EN=1且C2EN=1。

图 10-39 PWM输入模式时序



由于只有 TI1FP1 和 TI2FP2 连到了从模式控制器，所以 PWM 输入模式只能使用 TMRx_CH1/TMRx_CH2 信号。

10.2.3.7 强置输出模式

在输出模式（TMRx_CCMx 寄存器中 CxSEL=00）下，输出比较信号（OCxREF 和相应的 OCx）能够直接由软件强置为有效或无效状态，而不依赖于输出比较寄存器和计数器间的比较结果。

置 TMRx_CCMx 寄存器中相应的 OCxMODE=101，即可强置输出比较信号（OCxREF/OCx）为有效状态。这样 OCxREF 被强置为高电平（OCxREF 始终为高电平有效），同时 OCx 得到 CxP 极性位相反的值。

例如：CxP=0（OCx 高电平有效），则 OCx 被强置为高电平。

置 TMRx_CCMx 寄存器中的 OCxMODE=100，可强置 OCxREF 信号为低。

该模式下，在 TMRx_CCx 影子寄存器和计数器之间的比较仍然在进行，相应的标志也会被修改。因此仍然会产生相应的中断和 DMA 请求。这将会在下面的输出比较模式一节中介绍。

10.2.3.8 输出比较模式

此项功能是用来控制一个输出波形，或者指示一段给定的的时间已经到时。

当计数器与捕获/比较寄存器的内容相同时，输出比较功能做如下操作：

- 将输出比较模式（TMRx_CCMx寄存器中的OCxMODE位）和输出极性（TMRx_CCE寄存器中的CxP位）定义的值输出到对应的引脚上。在比较匹配时，输出引脚可以保持它的电平（OCxMODE=000）、被设置成有效电平（OCxMODE=001）、被设置成无效电平（OCxMODE=010）或进行翻转（OCxMODE=011）。
- 设置中断状态寄存器中的标志位（TMRx_STS寄存器中的CxIF位）。
- 若设置了相应的中断屏蔽（TMRx_DIE寄存器中的CxIE位），则产生一个中断。
- 若设置了相应的使能位（TMRx_DIE寄存器中的CxDE位，TMRx_CTRL2寄存器中的CDSEL位选择DMA请求功能），则产生一个DMA请求。

TMRx_CCMx 中的 OCxPEN 位选择 TMRx_CCx 寄存器是否需要使用预装载寄存器。

在输出比较模式下，更新事件 UEV 对 OCxREF 和 OCx 输出没有影响。

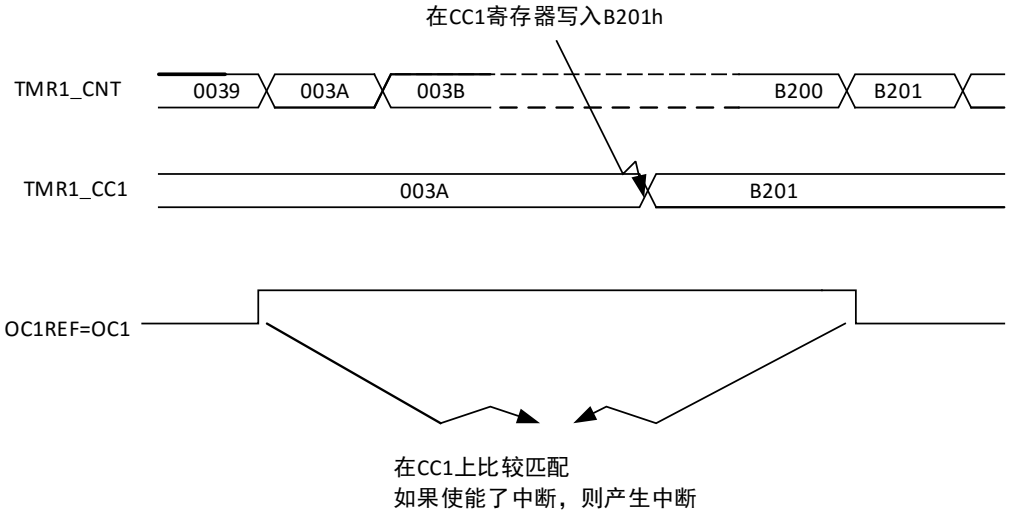
同步的精度可以达到计数器的一个计数周期。输出比较模式（在单脉冲模式下）也能用来输出一个单脉冲。

输出比较模式的配置步骤：

1. 选择计数器时钟（内部，外部，预分频器）。
2. 将相应的数据写入 TMRx_AR 和 TMRx_CCx 寄存器中。
3. 如果要产生一个中断请求和/或一个 DMA 请求，设置 CxIE 位和/或 CxDE 位。
4. 选择输出模式，例如当计数器 CNT 与 CCx 匹配时翻转 OCx 的输出引脚，CCx 预装载未用，开启 OCx 输出且高电平有效，则必须设置 OCxMODE='011'、OCxPEN='0'、CxP='0'和 CxEN='1'。
5. 设置 TMRx_CTRL1 寄存器的 CNTEN 位启动计数器

TMRx_CCx 寄存器能够在任何时候通过软件进行更新以控制输出波形，条件是未使用预装载寄存器（OCxPEN='0'，否则 TMRx_CCx 影子寄存器只能在发生下一次更新事件时被更新）。下图给出了一个例子。

图 10-40 输出比较模式，翻转OC1



10.2.3.9 PWM模式

脉冲宽度调制模式可以产生一个由 TMRx_AR 寄存器确定频率、由 TMRx_CCx 寄存器确定占空比的信号。

在 TMRx_CCMx 寄存器中的 OCxMODE 位写入'110'（PWM 模式 1）或'111'（PWM 模式 2），能够独立地设置每个 OCx 输出通道产生一路 PWM。必须设置 TMRx_CCMx 寄存器 OCxPEN 位以使能相应的预装载寄存器，最后还要设置 TMRx_CTRL1 寄存器的 ARPEN 位，（在向上计数或中心对称模式中）使能自动重装载的预装载寄存器。

仅当发生一个更新事件的时候，预装载寄存器才能被传送到影子寄存器，因此在计数器开始计数之前，必须通过设置 TMRx_EVEG 寄存器中的 UEVG 位来初始化所有的寄存器。OCx 的极性可以通过软件在 TMRx_CCE 寄存器中的 CxP 位设置，它可以设置为高电平有效或低电平有效。TMRx_CCE 寄存器中的 CxEN 位控制 OCx 输出使能。详见 TMRx_CCE 寄存器的描述。

在 PWM 模式（模式 1 或模式 2）下，TMRx_CNT 和 TMRx_CCx 始终在进行比较，（依据计数器的计数方向）以确定是否符合 TMRx_CCx ≤ TMRx_CNT 或者 TMRx_CNT ≤ TMRx_CCx 。然而为了与 OCREF_CLR 的功能（在下一个 PWM 周期之前，ETR 信号上的一个外部事件能够清除 OCxREF）一致，OCxREF 信号只能在下述条件下产生：

- 当比较的结果改变
- 当输出比较模式（TMRx_CCMx寄存器中的OCxMODE位）从“冻结”（无比较，OCxMODE='000'）切换到某个PWM模式（OCxMODE='110'或'111'）。

这样在运行中可以通过软件强置 PWM 输出。

根据 TMRx_CTRL1 寄存器中 CMSEL 位的状态，定时器能够产生边沿对齐的 PWM 信号或中央对齐的

PWM 信号。

PWM 边沿对齐模式

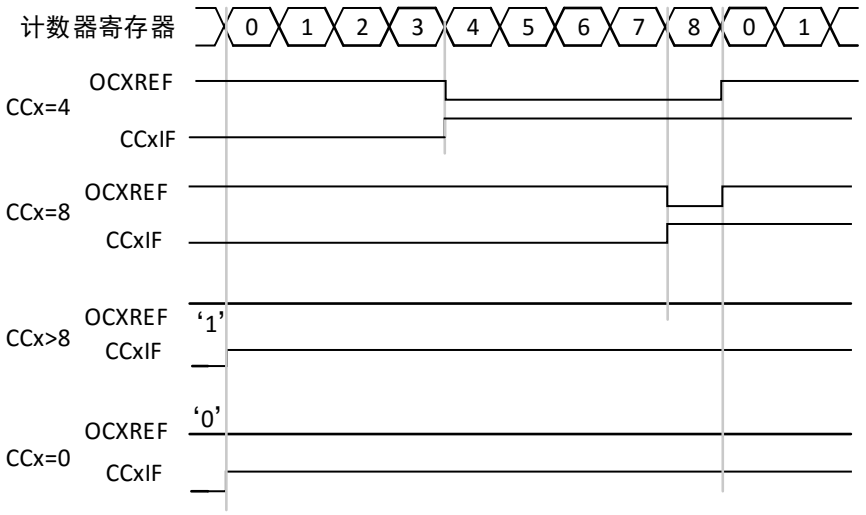
向上计数配置

当 TMRx_CTRL1 寄存器中的 DIR 位为低的时候执行向上计数。参看 10.2.3.2 节。

下面是一个 PWM 模式 1 的例子。当 TMRx_CNT<TMRx_CCx 时 PWM 信号参考 OCxREF 为高，否则为低。如果 TMRx_CCx 中的比较值大于自动重装载值（TMRx_AR），则 OCxREF 保持为'1'。

如果比较值为 0，则 OCxREF 保持为'0'。下图为 TMRx_AR=8 时边沿对齐的 PWM 波形实例。

图 10-41 边沿对齐的PWM波形（AR=8）



向下计数的配置

当 TMRx_CTRL1 寄存器的 DIR 位为高时执行向下计数。参看 10.2.3.2 节。

在 PWM 式 1，当 TMRx_CNT>TMRx_CCx 时参考信号 OCxREF 为低，否则高。如果 TMRx_CCx 中的比较值大于 TMRx_AR 中的自动重装载值，则 OCxREF 保持为'1'。该模式下不能产生 0%的 PWM 波形。

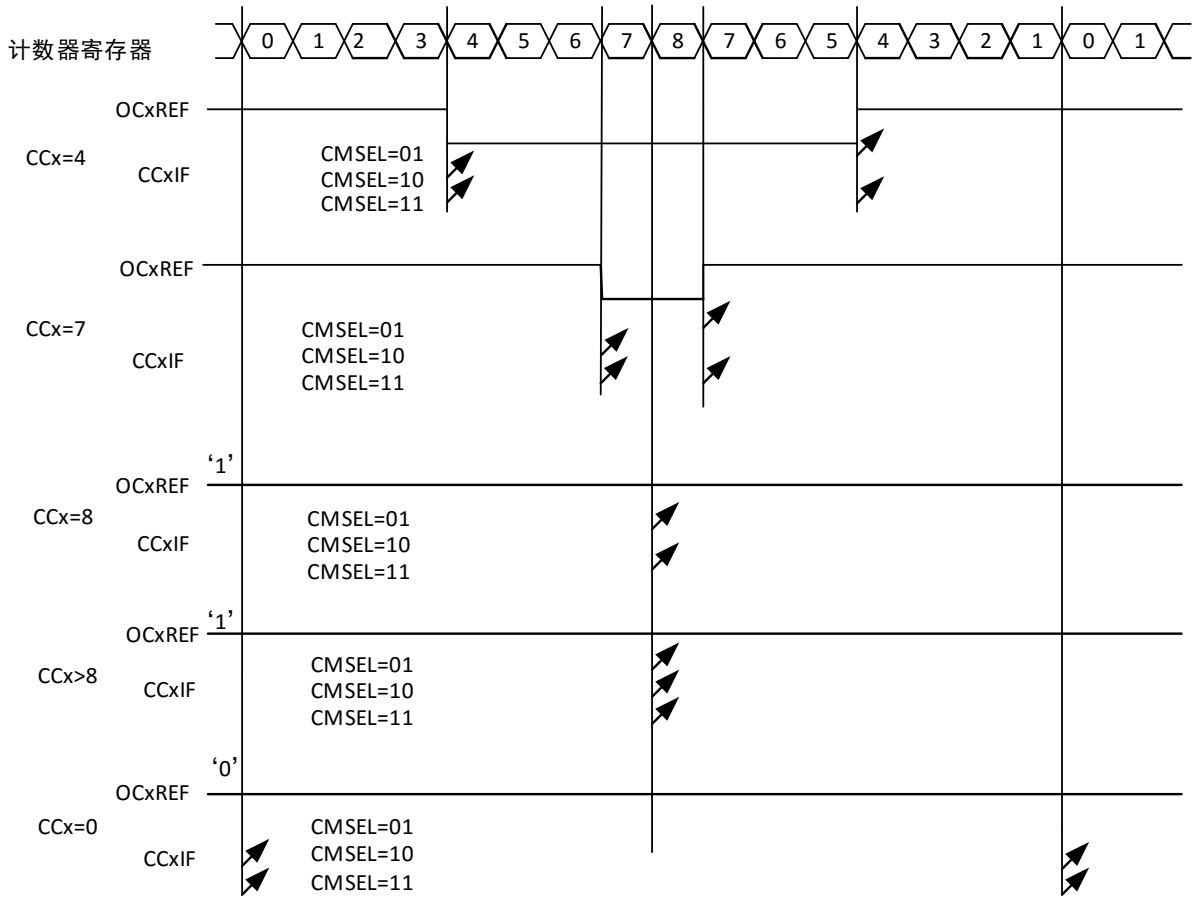
PWM 中央对齐模式

当 TMRx_CTRL1 寄存器中的 CMSEL 位不为'00'时，为中央对齐模式（所有其他的配置对 OCxREF/OCx 信号都有相同的作用）。根据不同的 CMSEL 位设置，比较标志可以在计数器向上计数时被置'1'、在计数器向下计数时被置'1'、或在计数器向上和向下计数时被置'1'。TMRx_CTRL1 寄存器中的计数方向位（DIR）由硬件更新，不要用软件修改它。参看 10.2.3.2 节的中央对齐模式。

下图给出了一些中央对齐的 PWM 波形的例子

- TMRx_AR=8
- PWM模式1
- TMRx_CTRL1寄存器中的CMSEL=01，在中央对齐模式1时，当计数器向下计数时设置比较标志。

图 10-42 中央对齐的PWM波形（AP=8）



使用中央对齐模式的提示：

- 进入中央对齐模式时，使用当前的向上/向下计数配置：这就意味着计数器向上还是向下计数取决于TMRx_CTRL1寄存器中DIR位的当前值。此外，软件不能同时修改DIR和CMSEL位。
- 不推荐当运行在中央对齐模式时改写计数器，因为这会产生不可预知的结果。特别地：
  - 如果写入计数器的值大于自动重加载的值（TMRx_CNT > TMRx_AR），则方向不会被更新。例如，如果计数器正在向上计数，它就会继续向上计数。
  - 如果将0或者TMRx_AR的值写入计数器，方向被更新，但不产生更新事件UEV。
- 使用中央对齐模式最保险的方法，就是在启动计数器之前产生一个软件更新（设置TMRx_EVEG位中的UEVG位），不要在计数进行过程中修改计数器的值。

10.2.3.10 单脉冲模式

单脉冲模式（OPM）是前述众多模式的一个特例。这种模式允许计数器响应一个激励，并在一个程序可控的延时之后，产生一个脉宽可程序控制的脉冲。

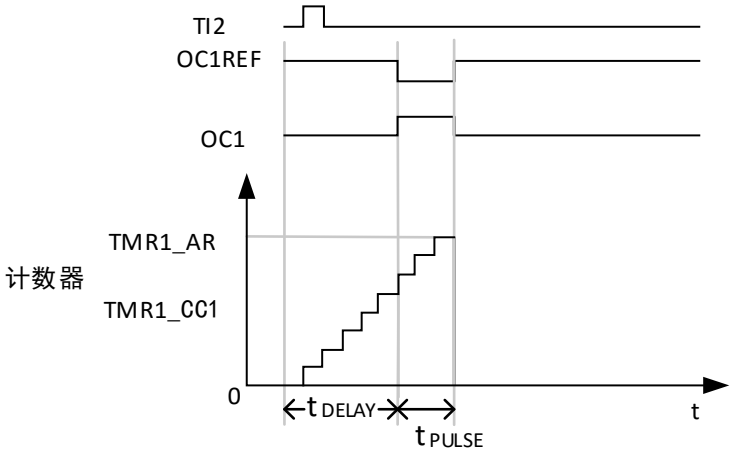
可以通过从模式控制器启动计数器，在输出比较模式或者 PWM 模式下产生波形。设置 TMRx_CTRL1 寄存器中的 OPMODE 位将选择单脉冲模式，这样可以让计数器自动地在产生下一个更新事件 UEV 时停止。

仅当比较值与计数器的初始值不同时，才能产生一个脉冲。启动之前（当定时器正在等待触发），必须如下配置：

向上计数方式：CNT < CCx ≤ AR （特别地，0 < CCx），

向下计数方式：CNT > CCx。

图 10-43 单脉冲模式的例子



例如，你需要在从 TI2 输入脚上检测到一个上升沿开始，延迟 t_{DELAY} 之后，在 OC1 上产生一个长度为 t_{PULSE} 的正脉冲。

假定 TI2FP2 作为触发 1：

- 置TMRx_CCM1寄存器中的C2SEL='01'，把TI2FP2映像到TI2。
- 置TMRx_CCE寄存器中的C2P='0'，使TI2FP2能够检测上升沿。
- 置TMRx_SMC寄存器中的TRGSEL='110'，TI2FP2作为从模式控制器的触发（TRGI）。
- 置TMRx_SMC寄存器中的SMSEL='110'（触发模式），TI2FP2被用来启动计数器。

OPM 波形由写入比较寄存器的数值决定（要考虑时钟频率和计数器预分频器）

- t_{DELAY}由写入TMRx_CC1寄存器中的值定义。
- t_{PULSE}由自动装载值和比较值之间的差值定义（TMRx_AR - TMRx_CC1）。
- 假定当发生比较匹配时要产生从'0'到'1'的波形，当计数器到达预装载值时要产生一个从'1'到'0'的波形；首先要置TMRx_CCM1寄存器的OC1MODE='111'，进入PWM模式2；根据需要有选择地使能预装载寄存器：置TMRx_CCM1中的OC1PEN='1'和TMRx_CTRL1寄存器中的 ARPEN；然后在TMRx_CC1寄存器中填写比较值，在TMRx_AR寄存器中填写自动装载值，修改UEVG位来产生一个更新事件，然后等待在TI2上的一个外部触发事件。本例中，C1P='0'。

在这个例子中，TMRx_CTRL1 寄存器中的 DIR 和 CMSEL 位应该置低。因为只需一个脉冲，所以必须设置 TMRx_CTRL1 寄存器中的 OPMODE='1'，在下一个更新事件（当计数器从自动装载值翻转到 0）时停止计数。

特殊情况：OCx 快速使能：

在单脉冲模式下，在 TIx 输入脚的边沿检测逻辑设置 CNTEN 位以启动计数器。然后计数器和比较值间的比较操作产生了输出的转换。但是这些操作需要一定的时钟周期，因此它限制了可得到的最小延时 t_{DELAY}。

如果要以最小延时输出波形，可以设置 TMRx_CCMx 寄存器中的 OCxFEN 位；此时 OCxREF（和 OCx）被强制响应激励而不再依赖比较的结果，输出的波形与比较匹配时的波形一样。OCxFEN 只在通道配置为 PWM1 和 PWM2 模式时起作用。

10.2.3.11 在外部事件时清除OCxREF信号

对于一个给定的通道，设置 TMRx_CCMx 寄存器中对应的 CxDIS 位为'1'，能够用 ETRF 输入端的高电平把 OCxREF 信号拉低，OCxREF 信号将保持为低直到发生下一次的更新事件 UEV。

该功能只能用于输出比较和 PWM 模式，而不能用于强置模式。

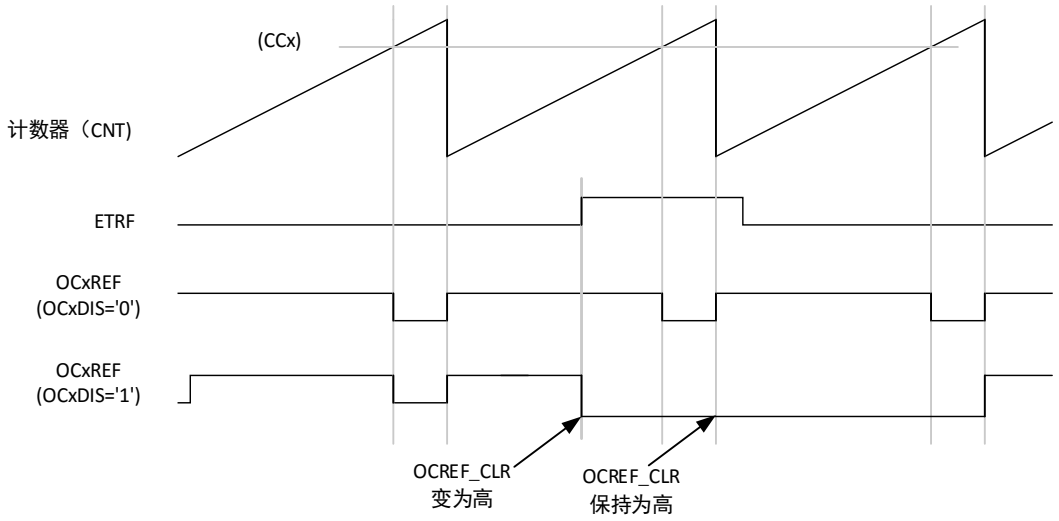
例如，OCxREF 信号可以联到一个比较器的输出，用于控制电流。这时，ETR 必须配置如下：

1. 外部触发预分频器必须处于关闭：TMRx_SMC 寄存器中的 ETD[1：0]='00'。

2. 必须禁止外部时钟模式 2：TMRx_SMC 寄存器中的 ECLKEN=’0’。
3. 外部触发极性（ETRGP）和外部触发滤波器（ETDF）可以根据需要配置。

下图显示了当 ETRF 输入变为高时，对应不同 CxDIS 的值，OCxREF 信号的动作。在这个例子中，定时器 TMRx 被置于 PWM 模式。

图 10-44 清除TMRx的OCxREF



10.2.3.12 编码器接口模式

选择编码器接口模式的方法是：如果计数器只在 TI1 的边沿计数，则置 TMRx_SMC 寄存器中的 SMSEL=001；如果只在 TI2 边沿计数，则置 SMSEL=010；如果计数器同时在 TI1 和 TI2 边沿计数，则置 SMSEL=011。

通过设置 TMRx_CCE 寄存器中的 C1P 和 C2P 位，可以选择 TI1 和 TI2 极性；如果需要，还可以对输入滤波器编程。

两个输入 TI1 和 TI2 被用来作为增量编码器的接口。参看表 10-2，假定计数器已经启动（TMRx_CTRL1 寄存器中的 CNTEN=’1’），计数器由每次在 TI1FP1 或 TI2FP2 上的有效跳变驱动。TI1FP1 和 TI2FP2 是 TI1 和 TI2 在通过输入滤波器和极性控制后的信号；如果没有滤波和变相，则 TI1FP1=TI1，TI2FP2=TI2。根据两个输入信号的跳变顺序，产生了计数脉冲和方向信号。依据两个输入信号的跳变顺序，计数器向上或向下计数，同时硬件对 TMRx_CTRL1 寄存器的 DIR 位进行相应的设置。

不管计数器是依靠 TI1 计数、依靠 TI2 计数或者同时依靠 TI1 和 TI2 计数。在任一输入端（TI1 或者 TI2）的跳变都会重新计算 DIR 位。

编码器接口模式基本上相当于使用了一个带有方向选择的外部时钟。这意味着计数器只在 0 到 TMRx_AR 寄存器的自动装载值之间连续计数（根据方向，或是 0 到 AR 计数，或是 AR 到 0 计数）。所以在开始计数之前必须配置 TMRx_AR；同样，捕获器、比较器、预分频器、触发输出特性等仍工作如常。

在这个模式下，计数器依照增量编码器的速度和方向被自动的修改，因此计数器的内容始终指示着编码器的位置。计数方向与相连的传感器旋转的方向对应。下表列出了所有可能的组合，假设 TI1 和 TI2 不同时变换。

表 10-2 计数方向与编码器信号的关系

有效边沿	相对信号的电平（TI1FP1 对应 TI2，TI2FP2 对应 TI1）	TI1FP1 信号		TI2FP2 信号	
		上升	下降	上升	下降
仅在 TI1 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 TI2 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

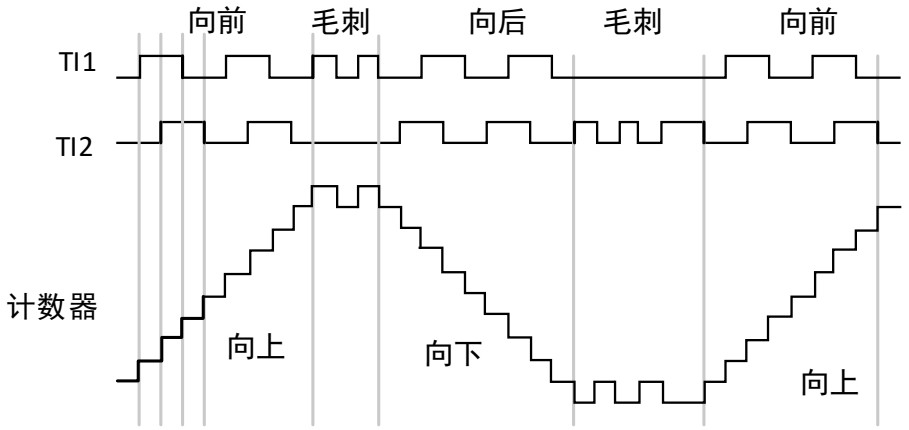
在 TI1 和 TI2 上计数	高	向下计数	向上计数	向上计数	向下计数
	低	向上计数	向下计数	向下计数	向上计数

一个外部的增量编码器可以直接与 MCU 连接而不需要外部接口逻辑。但是，一般会使用比较将编码器的差动输出转换到数字信号，这大大增加了抗噪声干扰能力。编码器输出的第三个信号表示机械零点，可以把它连接到一个外部中断输入并触发一个计数器复位。

下图是一个计数器操作的实例，显示了计数信号的产生和方向控制。它还显示了当选择了双边沿时，输入抖动是如何被抑制的；抖动可能会在传感器的位置靠近一个转换点时产生。在这个例子中，我们假定配置如下：

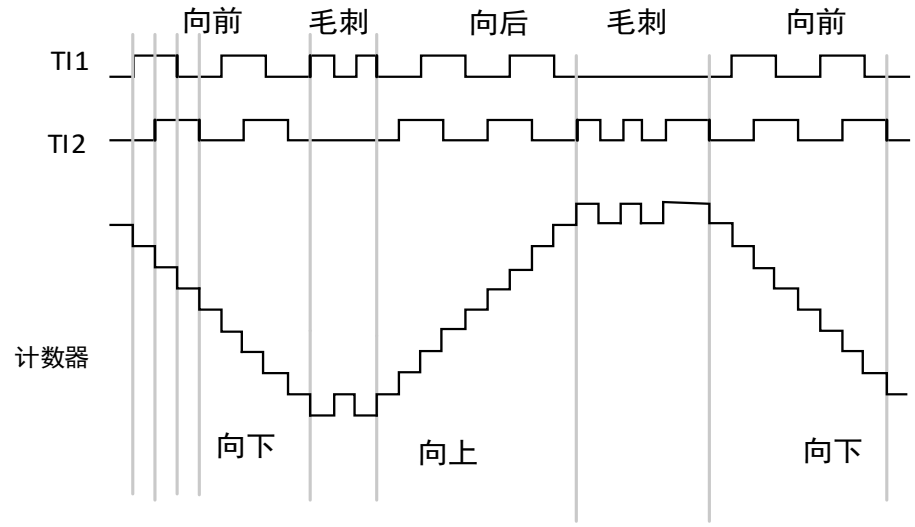
- C1SEL='01' （TMRx_CCM1寄存器，IC1FP1映射到TI1）。
- C2SEL='01' （TMRx_CCM1寄存器，IC2FP2映射到TI2）。
- C1P='0' （TMRx_CCE寄存器，IC1FP1不反相，IC1FP1=TI1）。
- C2P='0' （TMRx_CCE寄存器，IC2FP2不反相，IC2FP2=TI2）。
- SMSEL='011' （TMRx_SMC寄存器，所有的输入均在上升沿和下降沿有效）。
- CNTEN='1' （TMRx_CTRL1寄存器，计数器使能）。

图 10-45 编码器模式下的计数器操作实例



下图为当 IC1FP1 极性反相时计数器的操作实例（C1P='1'，其他配置与上例相同）

图 10-46 IC1FP1反相的编码器接口模式实例



当定时器配置成编码器接口模式时，提供传感器当前位置的信息。使用第二个配置在捕获模式的定时器，可以测量两个编码器事件的间隔，获得动态的信息（速度，加速度，减速度）。指示机械零点的编码器输出可被用做此目的。根据两个事件间的间隔，可以按照固定的时间读出计数器。如果可能的话，你可以把

计数器的值锁存到第三个输入捕获寄存器（捕获信号必须是周期的并且可以由另一个定时器产生）；也可以通过一个由实时时钟产生的 DMA 请求来读取它的值。

10.2.3.13 定时器输入异或功能

TMRx_CTRL2 寄存器中的 TI1SEL 位，允许通道 1 的输入滤波器连接到一个异或门的输出端，异或门的 3 个输入端为 TMRx_CH1、TMRx_CH2 和 TMRx_CH3。

异或输出能够被用于所有定时器的输入功能，如触发或输入捕获。本章的 10.4.3.18 节给出了此特性用于连接霍尔传感器的例子。

10.2.3.14 定时器和外部触发的同步

TMRx 定时器能够在多种模式下和一个外部的触发同步：复位模式、门控模式和触发模式。

从模式：复位模式

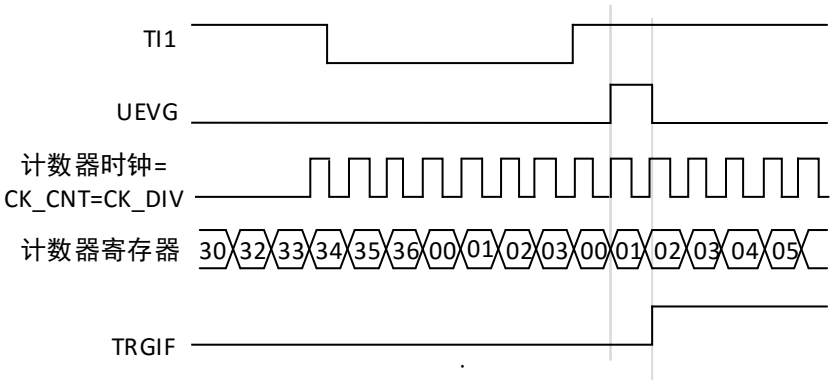
在发生一个触发输入事件时，计数器和它的预分频器能够重新被初始化；同时，如果 TMRx_CTRL1 寄存器的 UEVRS 位为低，还会产生一个更新事件 UEV；然后所有的预装载寄存器（TMRx_AR，TMRx_CCx）都会被更新。

在下面的例子中，TI1 输入端的上升沿导致向上计数器被清零：

- 配置通道1以检测TI1的上升沿。配置输入滤波器的带宽（在本例中，不需要任何滤波器，因此保持IC1DF=0000）。触发操作中不使用捕获预分频器，所以不需要配置它。C1SEL位只选择输入捕获源，即TMRx_CCM1寄存器中C1SEL=01。置TMRx_CCE寄存器中C1P=0以确定极性（只检测上升沿）。
- 置TMRx_SMC寄存器中SMSEL=100，配置定时器为复位模式；置TMRx_SMC寄存器中TRGSEL =101，选择TI1作为输入源。
- 置TMRx_CTRL1寄存器中CNTEN=1，启动计数器。

计数器开始依据内部时钟计数，然后正常运转直到 TI1 出现一个上升沿；此时，计数器被清零然后从 0 重新开始计数。同时，触发标志（TMRx_STS 寄存器中的 TRGIF 位）被设置，根据 TMRx_DIE 寄存器中 TRGIE（中断使能）位和 TRGDE（DMA 使能）位的设置，产生一个中断请求或一个 DMA 请求。下图显示当自动重装载寄存器 TMRx_AR=0x36 时的动作。在 TI1 上升沿和计数器的实际复位之间的延时，取决于 TI1 输入端的重同步电路。

图 10-47 复位模式下的控制电路



从模式：门控模式

按照选中的输入端电平使能计数。

在如下的例子中，计数器只在 TI1 为低时向上计数：

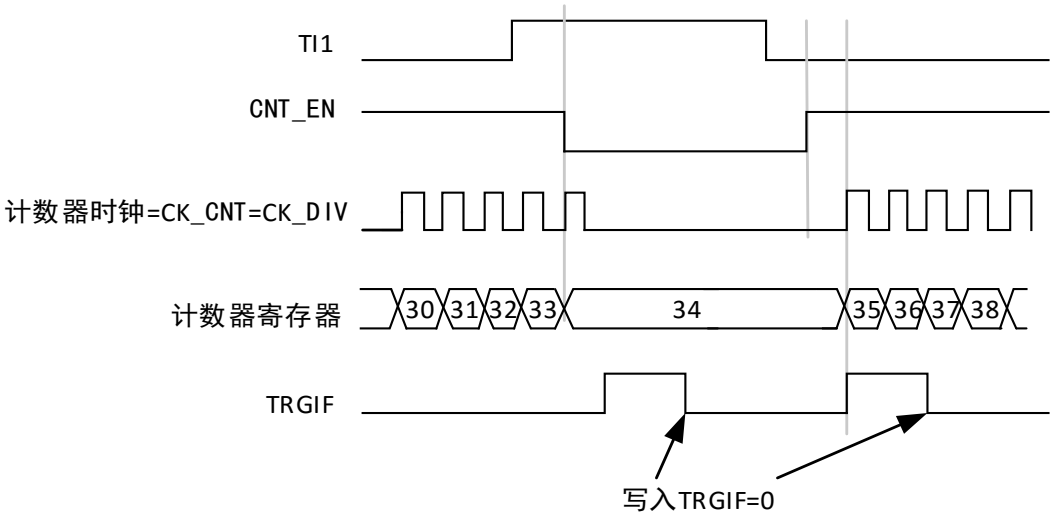
- 配置通道1以检测TI1上的低电平。配置输入滤波器带宽（本例中，不需要滤波，所以保持IC1DF=0000）。触发操作中不使用捕获预分频器，所以不需要配置。C1SEL位用于选择输入捕获源，置TMRx_CCM1寄存器中C1SEL=01。置TMRx_CCE寄存器中C1P=1以确定极性（只检测低电平）。
- 置TMRx_SMC寄存器中SMSEL=101，配置定时器为门控模式；置TMRx_SMC寄存器中TRGSEL=101，选择 TI1作为输入源。

- 置TMRx_CTRL1寄存器中CNTEN=1，启动计数器。在门控模式下，如果CNTEN=0，则计数器不能启动，不论触发输入电平如何。

只要 TI1 为低，计数器开始依据内部时钟计数，在 TI1 变高时停止计数。当计数器开始或停止时都设置 TMRx_STS 中的 TRGIF 标置。

TI1 上升沿和计数器实际停止之间的延时，取决于 TI1 输入端的重同步电路。

图 10-48 门控模式下的控制电路



从模式：触发模式

输入端上选中的事件使能计数器。

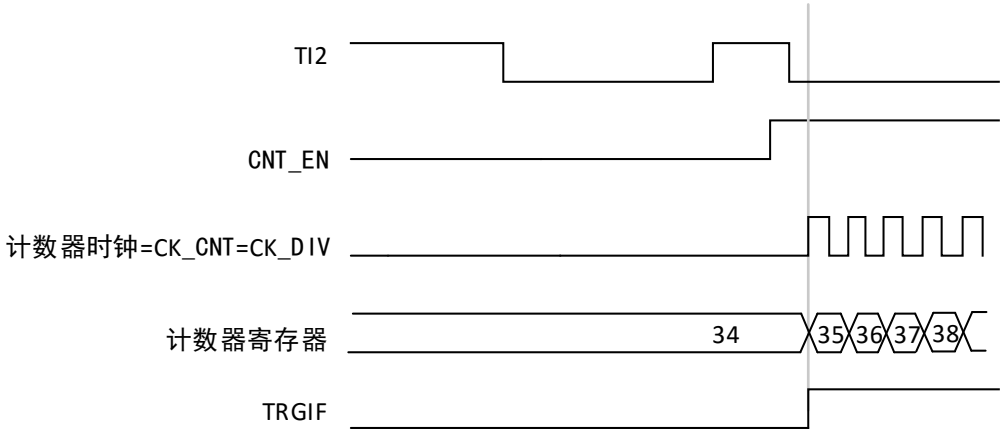
在下面的例子中，计数器在 TI2 输入的上升沿开始向上计数：

- 配置通道2检测TI2的上升沿。配置输入滤波器带宽（本例中，不需要任何滤波器，保持IC2DF=0000）。触发操作中不使用捕获预分频器，不需要配置。C2SEL位只用于选择输入捕获源，置TMRx_CCM1寄存器中C2SEL=01。置TMRx_CCE寄存器中C2P=0以确定极性（只检测低电平）。
- 置TMRx_SMC寄存器中SMSEL=110，配置定时器为触发模式；置TMRx_SMC寄存器中TRGSEL=110，选择TI2作为输入源。

当 TI2 出现一个上升沿时，计数器开始在内部时钟驱动下计数，同时设置 TRGIF 标志。

TI2 上升沿和计数器启动计数之间的延时，取决于 TI2 输入端的重同步电路。

图 10-49 触发器模式下的控制电路



从模式：外部时钟模式 2 ＋ 触发模式

外部时钟模式 2 可以与另一种从模式（外部时钟模式 1 和编码器模式除外）一起使用。这时，ETR 信号

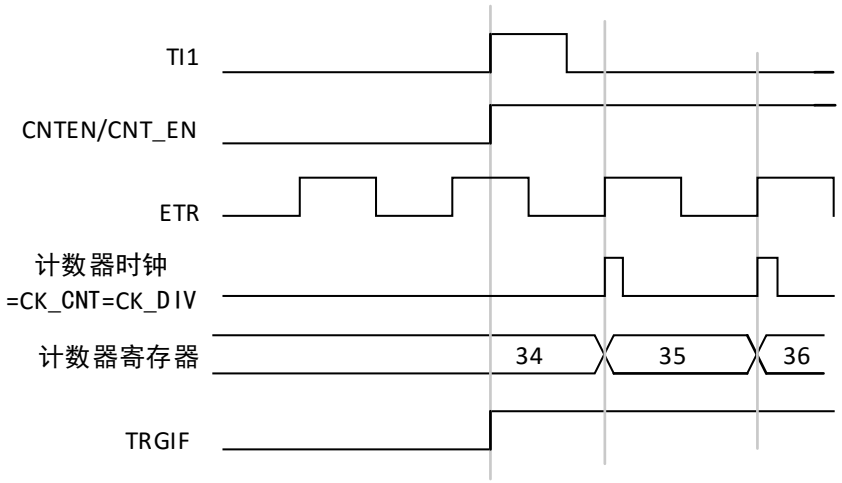
被用作外部时钟的输入，在复位模式、门控模式或触发模式时可以选择另一个输入作为触发输入。不建议使用 TMRx_SMC 寄存器的 TRGSEL 位选择 ETR 作为 TRGI。

下面的例子中，TI1 上出现一个上升沿之后，计数器即在 ETR 的每一个上升沿向上计数一次：

1. 通过 TMRx_SMC 寄存器配置外部触发输入电路：
   - ETDF=0000：没有滤波
   - ETD=00：不用预分频器
   - ETRGP=0：检测 ETR 的上升沿，置 ECLKEN=1 使能外部时钟模式 2
2. 按如下配置通道 1，检测 TI 的上升沿：
   - IC1DF=0000：没有滤波
   - 触发操作中不使用捕获预分频器，不需要配置
   - 置 TMRx_CCM1 寄存器中 C1SEL=01，选择输入捕获源
   - 置 TMRx_CCE 寄存器中 C1P=0 以确定极性（只检测上升沿）
3. 置 TMRx_SMC 寄存器中 SMSEL=110，配置定时器为触发模式。置 TMRx_SMC 寄存器中 TRGSEL=101，选择 TI1 作为输入源。

当 TI1 上出现一个上升沿时，TRGIF 标志被设置，计数器开始在 ETR 的上升沿计数。ETR 信号的上升沿和计数器实际复位间的延时，取决于 ETRP 输入端的重同步电路。

图 10-50 外部时钟模式2＋触发模式下的控制电路



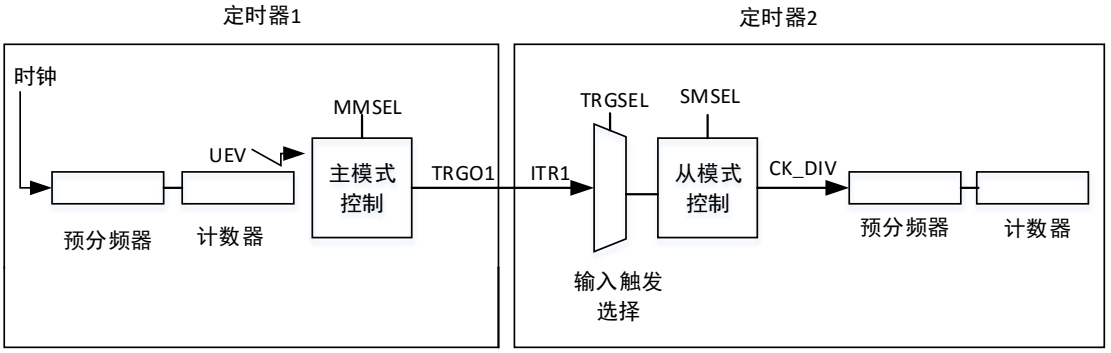
10.2.3.15 定时器同步

所有 TMRx 定时器在内部相连，用于定时器同步或链接。当一个定时器处于主模式时，它可以对另一个处于从模式的定时器的计数器进行复位、启动、停止或提供时钟等操作。

下图显示了触发选择和主模式选择模块的概况。

使用一个定时器作为另一个定时器的预分频器

图 10-51 主/从定时器的例子



如：可以配置定时器 1 作为定时器 2 的预分频器。参考图 10-51，进行下述操作：

- 配置定时器1为主模式，它可以在每一个更新事件UEV时输出一个周期性的触发信号。在TMR1_CTRL2寄存器的MMSEL=’010’时，每当产生一个更新事件时在TRGO1上输出一个上升沿信号。
- 连接定时器1的TRGO1输出至定时器2，设置TMR2_SMC寄存器的TRGSEL=’000’，配置定时器2为使用ITR1作为内部触发的从模式。
- 然后把从模式控制器置于外部时钟模式1（TMR2_SMC寄存器的SMSEL=111）；这样定时器2即可由定时器1周期性的上升沿（即定时器1的计数器溢出）信号驱动。
- 最后，必须设置相应（TMRx_CTRL1寄存器）的CNTEN位分别启动两个定时器。

注意： *如果 OCx 已被选中为定时器 1 的触发输出（MMSEL=1xx），它的上升沿用于驱动定时器 2 的计数器。*

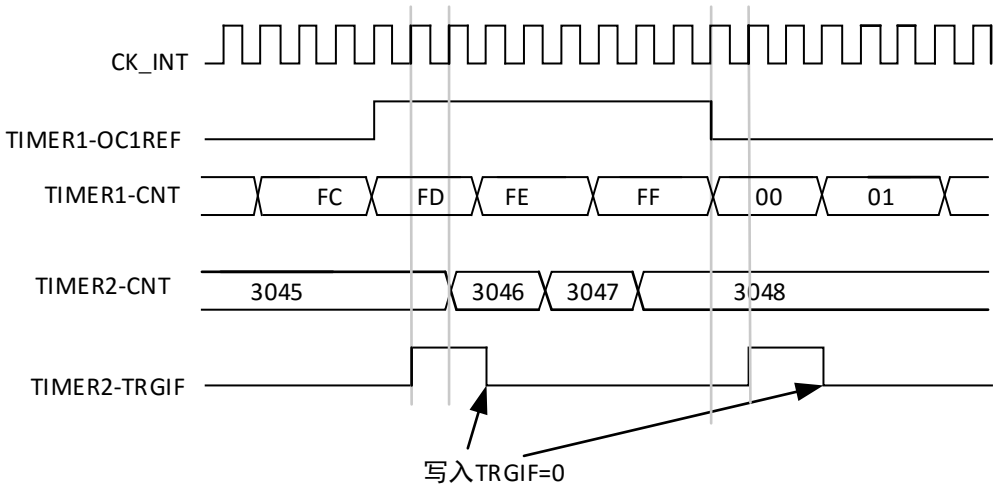
使用一个定时器使能另一个定时器

在这个例子中，定时器 2 的使能由定时器 1 的输出比较控制。参考图 10-51 的连接。只当定时器 1 的 OC1REF 为高时，定时器 2 才对分频后的内部时钟计数。两个定时器的时钟频率都是由预分频器对 CK_INT 除以 3（$f_{CK_CNT}=f_{CK_INT}/3$）得到。

- 配置定时器1为主模式，送出它的输出比较参考信号（OC1REF）为触发输出（TMR1_CTRL2寄存器的MMSEL=100）
- 配置定时器1的OC1REF波形（TMR1_CCM1寄存器）
- 配置定时器2从定时器1获得输入触发（TMR2_SMC寄存器的TRGSEL=000）
- 配置定时器2为门控模式（TMR2_SMC寄存器的SMSEL=101）
- 置TMR2_CTRL1寄存器的CNTEN=1以使能定时器2
- 置TMR1_CTRL1寄存器的CNTEN=1以启动定时器1

注意： *定时器 2 的时钟不与定时器 1 的时钟同步，这个模式只影响定时器 2 计数器的使能信号。*

图 10-52 定时器1的OC1REF控制定时器2

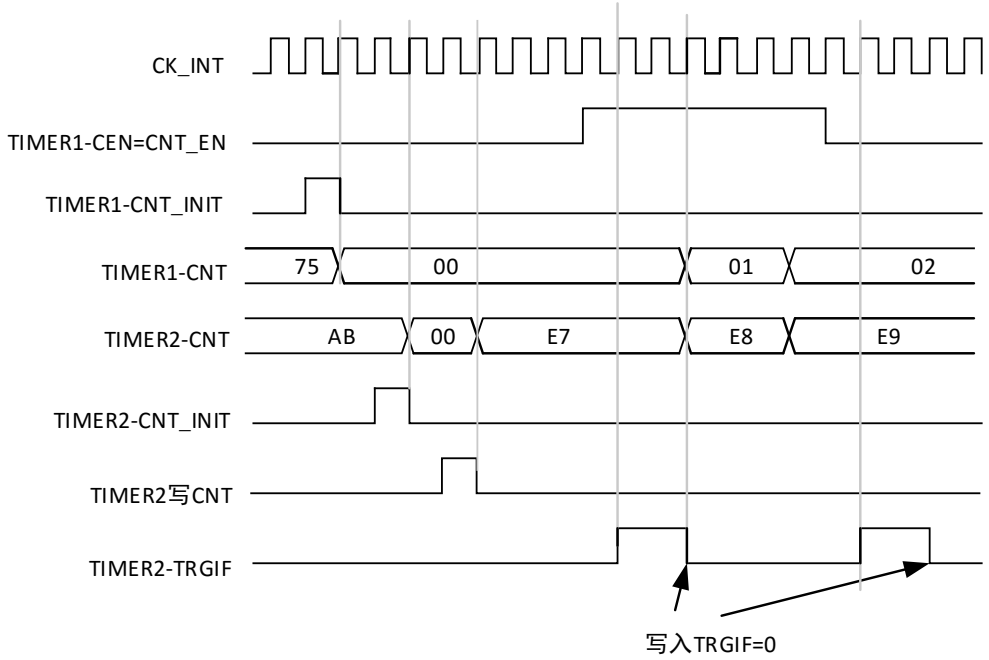


在图 10-52 的例子中，在定时器 2 启动之前，它们的计数器和预分频器未被初始化，因此它们从当前的数值开始计数。可以在启动定时器 1 之前复位 2 个定时器，使它们从给定的数值开始，即在定时器计数器中写入需要的任意数值。写 TMRx_EVEG 寄存器的 UEVG 位即可复位定时器。

在下一个例子中，需要同步定时器 1 和定时器 2。定时器 1 是主模式并从 0 开始，定时器 2 是从模式并从 0xE7 开始；2 个定时器的预分频器系数相同。写'0'到 TMR1_CTRL1 的 CNTEN 位将禁止定时器 1，定时器 2 随即停止。

- 配置定时器1为主模式，送出输出比较1参考信号（OC1REF）做为触发输出（TMR1_CTRL2寄存器的MMSEL=100）。
- 配置定时器1的OC1REF波形（TMR1_CCM1寄存器）。
- 配置定时器2从定时器1获得输入触发（TMR2_SMC寄存器的TRGSEL=000）
- 配置定时器2为门控模式（TMR2_SMC寄存器的SMSEL=101）
- 置TMR1_EVEG寄存器的UEVG='1'，复位定时器1。
- 置TMR2_EVEG寄存器的UEVG='1'，复位定时器2。
- 写'0xE7'至定时器2的计数器（TMR2_CNT），初始化它为0xE7。
- 置TMR2_CTRL1寄存器的CNTEN='1'以使能定时器2。
- 置TMR1_CTRL1寄存器的CNTEN='1'以启动定时器1。
- 置TMR1_CTRL1寄存器的CNTEN='0'以停止定时器1。

图 10-53 通过使能定时器1可以控制定时器2

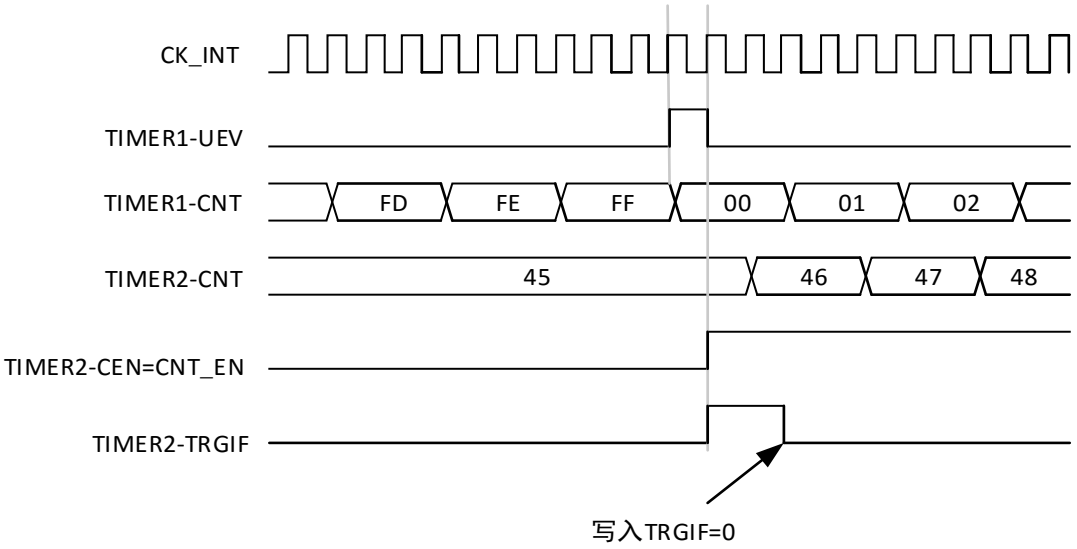


使用一个定时器去启动另一个定时器

在这个例子中，使用定时器 1 的更新事件使能定时器 2。参考图 10-51 的连接。一旦定时器 1 产生更新事件，定时器 2 即从它当前的数值（可以是非 0）按照分频的内部时钟开始计数。在收到触发信号时，定时器 2 的 CNTEN 位被自动地置'1'，同时计数器开始计数直到写'0'到 TMR2_CTRL1 寄存器的 CNTEN 位。两个定时器的时钟频率都是由预分频器对 CK_INT 除以 3（fCK_CNT=fCK_INT/3）。

- 配置定时器1为主模式，送出它的更新事件（UEV）做为触发输出（TMR1_CTRL2寄存器的MMSEL=010）。
- 配置定时器1的周期（TMR1_AR寄存器）。
- 配置定时器2从定时器1获得输入触发（TMR2_SMC寄存器的TRGSEL=000）
- 配置定时器2为触发模式（TMR2_SMC寄存器的SMSEL=110）
- 置TMR1_CTRL1寄存器的CNTEN=1以启动定时器1。

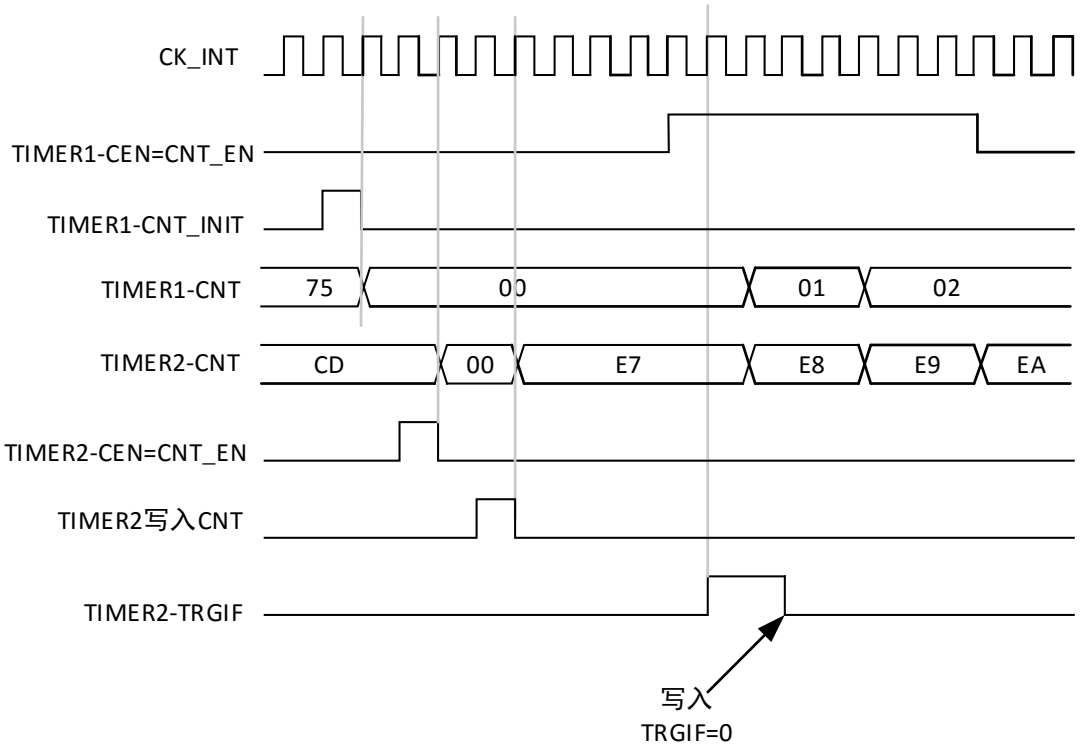
图 10-54 使用定时器1的更新触发定时器2



在上一个例子中，可以在启动计数之前初始化两个计数器。图 10-55 显示在与上例相同配置情况下，使

用触发模式而不是门控模式（TMR2_SMC 寄存器的 SMSEL=110）的动作。

图 10-55 利用定时器1的使能触发定时器2



使用一个定时器作为另一个的预分频器

这个例子使用定时器 1 作为定时器 2 的预分频器。参考图 10-51 的连接，配置如下：

- 配置定时器1为主模式，送出它的更新事件UEV做为触发输出（TMR1_CTRL2寄存器的MMSEL='010'）。然后每次计数器溢出时输出一个周期信号。
- 配置定时器1的周期（TMR1_AR寄存器）。
- 配置定时器2从定时器1获得输入触发（TMR2_SMC寄存器的TRGSEL=000）。
- 配置定时器2使用外部时钟模式（TMR2_SMC寄存器的SMSEL=111）。
- 置TMR1_CTRL2寄存器的CNTEN=1以启动定时器2。
- 置TMR1_CTRL1寄存器的CNTEN=1以启动定时器1。

使用一个外部触发同步地启动 2 个定时器。

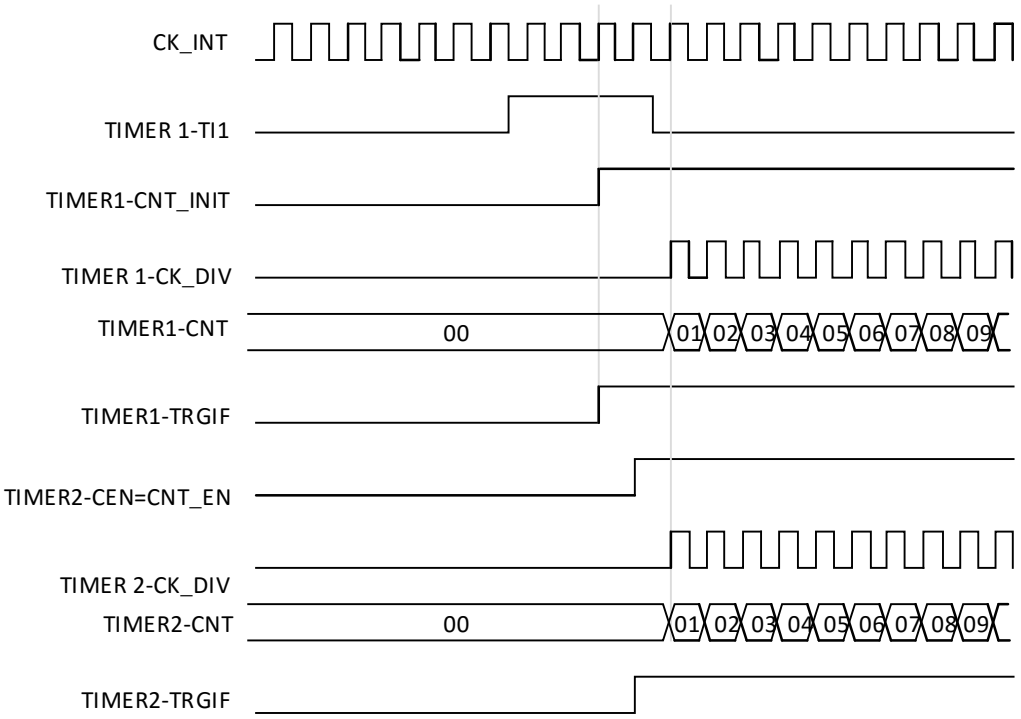
这个例子中当定时器 1 的 TI1 输入上升时使能定时器 1，使能定时器 1 的同时使能定时器 2，参见图 10-51。为保证计数器的对齐，定时器 1 必须配置为主/从模式（对于 TI1 为从，对于定时器 2 为主）：

- 配置定时器1为主模式，送出它的使能做为触发输出（TMR1_CTRL2寄存器的MMSEL='001'）。
- 配置定时器1为从模式，从TI1获得输入触发（TMR1_SMC寄存器的TRGSEL='100'）。
- 配置定时器1为触发模式（TMR1_SMC寄存器的SMSEL='110'）。
- 配置定时器1为主/从模式，TMR1_SMC寄存器的MSMODE='1'。
- 配置定时器2从定时器1获得输入触发（TMR2_SMC寄存器的TRGSEL=000）
- 配置定时器2为触发模式（TMR2_SMC寄存器的SMSEL='110'）。当定时器1的TI1上出现一个上升沿时，两个定时器同步地按照内部时钟开始计数，两个TRGIF标志也同时被设置。

注意：在这个例子中，在启动之前两个定时器都被初始化（设置相应的 UEVG 位），两个计数器都从 0 开始，但可以通过写入任意一个计数器寄存器（TMRx_CNT）在定时器间

插入一个偏移。下图中能看到主/从模式下在定时器 1 的 CNT_EN 和 CK_DIV 之间有个延迟。

图 10-56 使用定时器1的TI1输入触发定时器1和定时器2



10.2.3.16 调试模式

当微控制器进入调试模式（Cortex™-M4F 核心停止），根据 DBG 模块中 DBG_TMRx_STOP 的设置，TMRx 计数器或者继续正常操作，或者停止。详见随后第 22.2.2 节：支持定时器、看门狗、bxCAN 和 I2C 的调试。

10.2.4 TMRx寄存器描述

可以用半字（16 位）或字（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 10-3 TMRx – 寄存器图和复位值

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	TMRx_CTRL1	保留																					PMEN	CLKDIV[1:0]		ARPEN	CMSEL[1:0]		DIR	OPMODE	UVERS	UEVDIS	CNTEN
	复位值																						0	0	0	0	0	0	0	0	0	0	0
0x04	TMRx_CTRL2	保留																								TI1SEL	MMSEL[2:0]			CDSEL	保留		
	复位值																									0	0	0	0	0			
0x08	TMRx_SMC	保留																ETRGP	ECLKEN	ETD[1:0]		ETDF[3:0]				MSMODE	TRGSEL[2:0]			保留	SMSEL[2:0]		
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0

| | | |

	复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x40	TMRx_CC4	CC4[31：16]																CC4[15：0]														
	复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x48	TMRx_ DMAC	保留																		DBLEN[4：0]					保留	ADDR[4：0]					
	复位值																			0	0	0	0	0		0	0	0	0	0	
0x4C	TMRx_ DMABA	保留																DMABA[15：0]														
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0

10.2.4.1 控制寄存器1（TMRx_CTRL1）

偏移地址：0x00

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留					PMEN	CLKDIV [1：0]		ARPEN	CMSEL[1：0]		DIR	OPM ODE	UEVRS	UEVDIS	CNTEN
res					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 15：11	保留，始终读为 0。
位 10	**PMEN：** 增强模式使能（Plus Mode Enable） 开启 TMRx 增强模式，该模式下 TMRx_CNT， TMRx_AR， TMRx_CC1/2/3/4 由 16 位扩展为 32 位。 0：不开启增强模式； 1：开启增强模式。 *注：TMR2 和 TMR5 才具有此功能，其它 TMR 设置此位无效。*
位 9：8	**CLKDIV[1：0]：** 时钟分频因子（Clock division） 定义在定时器时钟（CK_INT）频率与数字滤波器（ETR，TIx）使用的采样频率之间的分频比例。 00：t_{DTS} = t_{CK_INT} 01：t_{DTS} = 2 xt_{CK_INT} 10：t_{DTS} = 4 x t_{CK_INT} 11：保留
位 7	**ARPEN：** 自动重装载预装载允许位 （Auto-reload preload enable） 0：TMRx_AR 寄存器没有缓冲； 1：TMRx_AR 寄存器被装入缓冲器。
位 6：5	**CMSEL[1：0]：** 选择中央对齐模式 （Center-aligned mode selection） 00：边沿对齐模式。计数器依据方向位（DIR）向上或向下计数。 01：中央对齐模式 1。计数器交替地向上和向下计数。配置为输出的通道（TMRx_CCMx 寄存器中 CxSEL=00）的输出比较中断标志位，只在计数器向下计数时被设置。 10：中央对齐模式 2。计数器交替地向上和向下计数。配置为输出的通道（TMRx_CCMx 寄存器中 CxSEL=00）的输出比较中断标志位，只在计数器向上计数时被设置。 11：中央对齐模式 3。计数器交替地向上和向下计数。配置为输出的通道（TMRx_CCMx 寄存器中 CxSEL=00）的输出比较中断标志位，在计数器向上和向下计数时均被设置。 *注： 在计数器开启时（CNTEN=1），不允许从边沿对齐模式转换到中央对齐模式。*
位 4	**DIR：** 方向 （Direction） 0：计数器向上计数； 1：计数器向下计数。 *注：当计数器配置为中央对齐模式或编码器模式时，该位为只读。*

位 3	**OPMODE**：单脉冲模式 （One pulse mode） 0：在发生更新事件时，计数器不停止； 1：在发生下一次更新事件（清除 CNTEN 位）时，计数器停止。
位 2	**UEVRS**：更新请求源 （Update request source） 软件通过该位选择 UEV 事件的源 0：如果使能了更新中断或 DMA 请求，则下述任一事件产生更新中断或 DMA 请求： – 计数器溢出/下溢 – 设置 UEVG 位 – 从模式控制器产生的更新 1：如果使能了更新中断或 DMA 请求，则只有计数器溢出/下溢才产生更新中断或 DMA 请求。
位 1	**UEVDIS**：禁止更新 （Update disable） 软件通过该位允许/禁止 UEV 事件的产生 0：允许 UEV。更新（UEV）事件由下述任一事件产生： – 计数器溢出/下溢 – 设置 UEVG 位 – 从模式控制器产生的更新 具有缓存的寄存器被装入它们的预装载值。 1：禁止 UEV。不产生更新事件，影子寄存器（AR、DIV、CCx）保持它们的值。如果设置 UEVG 位或从模式控制器发出了一个硬件复位，则计数器和预分频器被重新初始化。
位 0	**CNTEN**：使能计数器 0：禁止计数器； 1：使能计数器。 *注：在软件设置了 CNTEN 位后，外部时钟、门控模式和编码器模式才能工作。触发模式可以自动地通过硬件设置 CNTEN 位。* 在单脉冲模式下，当发生更新事件时，CNTEN 被自动清除。

10.2.4.2 控制寄存器2（TMRx_CTRL2）

偏移地址：0x04

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留								TI1SEL	MMSEL[2：0]			CDSEL	保留		
res								rw	rw	rw	rw	rw	res		

位 15：8	保留，始终读为 0。
位 7	**TI1SEL**：TI1 选择 （TI1 selection） 0：TMRx_CH1 引脚连到 TI1 输入； 1：TMRx_CH1、TMRx_CH2 和 TMRx_CH3 引脚经异或后连到 TI1 输入。 见本章 10.4.3.18 的与霍尔传感器的接口一节。

位 6：4	**MMSEL[2：0]：**主模式选择 （Master mode selection） 这 3 位用于选择在主模式下送到从定时器的同步信息（TRGO）。可能的组合如下： 000：**复位** – TMRx_EVEG 寄存器的 UEVG 位被用于作为触发输出（TRGO）。如果是触发输入产生的复位（从模式控制器处于复位模式），则 TRGO 上的信号相对实际的复位会有一个延迟。 001：**使能** – 计数器使能信号 CNT_EN 被用于作为触发输出（TRGO）。有时需要在同一时间启动多个定时器或控制在一段时间内使能从定时器。计数器使能信号是通过 CNTEN 控制位和门控模式下的触发输入信号的逻辑或产生。 当计数器使能信号受控于触发输入时，TRGO 上会有一个延迟，除非选择了主/从模式（见 TMRx_SMC 寄存器中 MSMODE 位的描述）。 010：**更新** – 更新事件被选为触发输入（TRGO）。例如，一个主定时器的时钟可以被用作一个从定时器的预分频器。 011：比较脉冲 – 在发生一次捕获或一次比较成功时，当要设置 C1IF 标志时（即使它已经为高），触发输出送出一个正脉冲（TRGO）。 100：**比较** – OC1REF 信号被用于作为触发输出（TRGO）。 101：**比较** – OC2REF 信号被用于作为触发输出（TRGO）。 110：**比较** – OC3REF 信号被用于作为触发输出（TRGO）。 111：**比较** – OC4REF 信号被用于作为触发输出（TRGO）。
位 3	**CDSEL：**捕获/比较的 DMA 选择 0：当发生 CCx 事件时，送出 CCx 的 DMA 请求； 1：当发生更新事件时，送出 CCx 的 DMA 请求。
位 2：0	保留，始终读为 0。

10.2.4.3 从模式控制寄存器（TMRx_SMC）

偏移地址：0x08

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ETRGP	ECLKEN	ETD[1：0]		ETDF[3：0]				MSMODE	TRGSEL[2：0]			保留	SMSEL[2：0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	res	rw	rw	rw

位 15	**ETRGP：**外部触发极性（External trigger polarity） 该位选择是用 ETR 还是 ETR 的反相来作为触发操作 0：ETR 不反相，高电平或上升沿有效； 1：ETR 被反相，低电平或下降沿有效。
位 14	**ECLKEN：**外部时钟使能位 （External clock enable） 该位启用外部时钟模式 2 0：禁止外部时钟模式 2； 1：使能外部时钟模式 2。计数器由 ETRF 信号上的任意有效边沿驱动。 *注 1：设置 ECLKEN 位与选择外部时钟模式 1 并将 TRGI 连到 ETRF（SMSEL=111 和 TRGSEL=111）具有相同功效。* *注 2：下述从模式可以与外部时钟模式 2 同时使用：复位模式、门控模式和触发模式；但是，这时 TRGI 不能连到 ETRF（TRGSEL 位不能是'111'）。* *注 3：外部时钟模式 1 和外部时钟模式 2 同时被使能时，外部时钟的输入是 ETRF。*
位 13：12	**ETD[1：0]：**外部触发预分频 （External trigger prescaler） 外部触发信号 ETRP 的频率必须最多是 CK_INT 频率的 1/4。当输入较快的外部时钟时，可以使用预分频降低 ETRP 的频率。 00：关闭预分频； 01：ETRP 频率除以 2； 10：ETRP 频率除以 4； 11：ETRP 频率除以 8。

位 11：8	**ETDF[3：0]：**外部触发滤波（External trigger filter） 这些位定义了对 ETRP 信号采样的频率和对 ETRP 数字滤波的带宽。实际上，数字滤波器是一个事件计数器，它记录到 N 个事件后会产生一个输出的跳变。 0000：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 7	**MSMODE：**主/从模式 （Master/slave mode） 0：无作用； 1：触发输入（TRGI）上的事件被延迟了，以允许在当前定时器（通过 TRGO）与它的从定时器间的完美同步。这对要求把几个定时器同步到一个单一的外部事件时是非常有用的。
位 6：4	**TRGSEL[2：0]：**触发选择 （Trigger selection） 这 3 位选择用于同步计数器的触发输入。 000：内部触发 0（ITR0），TMR1　　100：TI1 的边沿检测器（TI1F_ED） 001：内部触发 1（ITR1），TMR2　　101：滤波后的定时器输入 1（TI1FP1） 010：内部触发 2（ITR2），TMR3　　110：滤波后的定时器输入 2（TI2FP2） 011：内部触发 3（ITR3），TMR4　　111：外部触发输入（ETRF） 关于每个定时器中 ITRx 的细节，参见表 10-4。 *注：这些位只能在未用到（如 SMSEL=000）时被改变，以避免在改变时产生错误的边沿检测。*
位 3	保留，始终读为 0。
位 2：0	**SMSEL[2：0]：**从模式选择 （Slave mode selection） 当选择了外部信号，触发信号（TRGI）的有效边沿与选中的外部输入极性相关（见输入控制寄存器和控制寄存器的说明） 000：关闭从模式 – 如果 CNTEN=1，则预分频器直接由内部时钟驱动。 001：编码器模式 1 – 根据 TI2FP2 的电平，计数器在 TI1FP1 的边沿向上/下计数。 010：编码器模式 2 – 根据 TI1FP1 的电平，计数器在 TI2FP2 的边沿向上/下计数。 011：编码器模式 3 – 根据另一个信号的输入电平，计数器在 TI1FP1 和 TI2FP2 的边沿向上/下计数。 100：复位模式 – 选中的触发输入（TRGI）的上升沿重新初始化计数器，并且产生一个更新寄存器的信号。 101：门控模式 – 当触发输入（TRGI）为高时，计数器的时钟开启。一旦触发输入变为低，则计数器停止（但不复位）。计数器的启动和停止都是受控的。 110：触发模式 – 计数器在触发输入 TRGI 的上升沿启动（但不复位），只有计数器的启动是受控的。 111：外部时钟模式 1 – 选中的触发输入（TRGI）的上升沿驱动计数器。 *注：如果 TI1F_EN 被选为触发输入（TRGSEL=100）时，不要使用门控模式。这是因为，TI1F_ED 在每次 TI1F 变化时输出一个脉冲，然而门控模式是要检查触发输入的电平。*

表 10-4 TMRx内部触发连接(1)

从定时器	ITR0 （TRGSEL = 000）	ITR1 （TRGSEL = 001）	ITR2 （TRGSEL = 010）	ITR3 （TRGSEL = 011）
TMR2	TMR1	TMR8/USB_SOF(2)	TMR3	TMR4
TMR3	TMR1	TMR2	TMR5	TMR4
TMR4	TMR1	TMR2	TMR3	TMR8
TMR5	TMR2	TMR3	TMR4	TMR8

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 ITRx 也不存在。

注意 2：ITR1 可以选择 TMR8 或 USB_SOF，由 AFIO_MAP4 的 TMR2ITR1_IRMP 位控制。

10.2.4.4 DMA/中断使能寄存器（TMRx_DIE）

偏移地址：0x0C

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留	TRGDE	保留	C4DE	C3DE	C2DE	C1DE	UEVDE	保留	TuRGIE	保留	C4IE	C3IE	C2IE	C1IE	UEVIE
res	rw	res	rw	rw	rw	rw	rw	res	rw	res	rw	rw	rw	rw	rw

位 15	保留，始终读为 0。
位 14	**TRGDE**：允许触发 DMA 请求 （Trigger DMA request enable） 0：禁止触发 DMA 请求； 1：允许触发 DMA 请求。
位 13	保留，始终读为 0。
位 12	**C4DE**：允许捕获/比较 4 的 DMA 请求 （Capture/Compare 4 DMA request enable） 0：禁止捕获/比较 4 的 DMA 请求； 1：允许捕获/比较 4 的 DMA 请求。
位 11	**C3DE**：允许捕获/比较 3 的 DMA 请求 （Capture/Compare 3 DMA request enable） 0：禁止捕获/比较 3 的 DMA 请求； 1：允许捕获/比较 3 的 DMA 请求。
位 10	**C2DE**：允许捕获/比较 2 的 DMA 请求 （Capture/Compare 2 DMA request enable） 0：禁止捕获/比较 2 的 DMA 请求； 1：允许捕获/比较 2 的 DMA 请求。
位 9	**C1DE**：允许捕获/比较 1 的 DMA 请求 （Capture/Compare 1 DMA request enable） 0：禁止捕获/比较 1 的 DMA 请求； 1：允许捕获/比较 1 的 DMA 请求。
位 8	**UEVDE**：允许更新的 DMA 请求 （Update DMA request enable） 0：禁止更新的 DMA 请求； 1：允许更新的 DMA 请求。
位 7	保留，始终读为 0。
位 6	**TRGIE**：触发中断使能 （Trigger interrupt enable） 0：禁止触发中断； 1：使能触发中断。
位 5	保留，始终读为 0。
位 4	**C4IE**：允许捕获/比较 4 中断 （Capture/Compare 4 interrupt enable） 0：禁止捕获/比较 4 中断； 1：允许捕获/比较 4 中断。
位 3	**C3IE**：允许捕获/比较 3 中断 （Capture/Compare 3 interrupt enable） 0：禁止捕获/比较 3 中断； 1：允许捕获/比较 3 中断。
位 2	**C2IE**：允许捕获/比较 2 中断 （Capture/Compare 2 interrupt enable） 0：禁止捕获/比较 2 中断； 1：允许捕获/比较 2 中断。
位 1	**C1IE**：允许捕获/比较 1 中断 （Capture/Compare 1 interrupt enable） 0：禁止捕获/比较 1 中断； 1：允许捕获/比较 1 中断。

位 0	**UEVIE**：允许更新中断 （Update interrupt enable） 0：禁止更新中断； 1：允许更新中断。

10.2.4.5 状态寄存器（TMRx_STS）

偏移地址：0x10

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留			C4OF	C3OF	C2OF	C1OF	保留		TRGIF	保留	C4IF	C3IF	C2IF	C1IF	UEVIF
res			rc_w0	rc_w0	rc_w0	rc_w0	res		rc_w0	res	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0

位 15：13	保留，始终读为 0。
位 12	**C4OF**：捕获/比较 4 重复捕获标记 （Capture/Compare 4 overcapture flag） 参见 C1OF 描述。
位 11	**C3OF**：捕获/比较 3 重复捕获标记 （Capture/Compare 3 overcapture flag） 参见 C1OF 描述。
位 10	**C2OF**：捕获/比较 2 重复捕获标记 （Capture/Compare 2 overcapture flag） 参见 C1OF 描述。
位 9	**C1OF**：捕获/比较 1 重复捕获标记 （Capture/Compare 1 overcapture flag） 仅当相应的通道被配置为输入捕获时，该标记可由硬件置’1’。写’0’可清除该位。 0：无重复捕获产生； 1：当计数器的值被捕获到 TMRx_CC1 寄存器时，C1IF 的状态已经为’1’。
位 8：7	保留，始终读为 0。
位 6	**TRGIF**：触发器中断标记 （Trigger interrupt flag） 当发生触发事件（当从模式控制器处于除门控模式外的其它模式时，在 TRGI 输入端检测到有效边沿，或门控模式下的任一边沿）时由硬件对该位置’1’。它由软件清’0’。 0：无触发器事件产生； 1：触发器中断等待响应。
位 5	保留，始终读为 0。
位 4	**C4IF**：捕获/比较 4 中断标记 （Capture/Compare 4 interrupt flag） 参考 C1IF 描述。
位 3	**C3IF**：捕获/比较 3 中断标记 （Capture/Compare 3 interrupt flag） 参考 C1IF 描述。
位 2	**C2IF**：捕获/比较 2 中断标记 （Capture/Compare 2 interrupt flag） 参考 C1IF 描述。
位 1	**C1IF**：捕获/比较 1 中断标记 （Capture/Compare 1 interrupt flag） **如果通道 CC1 配置为输出模式：** 当计数器值与比较值匹配时该位由硬件置’1’，但在中心对称模式下除外（参考 TMRx_CTRL1 寄存器 的 CMSEL 位）。它由软件清’0’。 0：无匹配发生； 1：TMRx_CNT 的值与 TMRx_CC1 的值匹配。 **如果通道 CC1 配置为输入模式：** 当捕获事件发生时该位由硬件置’1’，它由软件清’0’或通过读 TMRx_CC1 清’0’。 0：无输入捕获产生； 1：计数器值已被捕获（拷贝）至 TMRx_CC1（在 IC1 上检测到与所选极性相同的边沿）。

位 0	**UEVIF**：更新中断标记 （Update interrupt flag） 当产生更新事件时该位由硬件置'1'。它由软件清'0'。 0：无更新事件产生； 1：更新中断等待响应。当寄存器被更新时该位由硬件置'1'： – 若 TMRx_CTRL1 寄存器的 UEVDIS=0、UEVRS=0，当 TMRx_EVEG 寄存器的 UEVG=1 时产生更新事件（软件对计数器 CNT 重新初始化）； – 若 TMRx_CTRL1 寄存器的 UEVDIS=0、UEVRS=0，当计数器 CNT 被触发事件重初始化时产生更新事件。（参考同步控制寄存器的说明）

10.2.4.6 事件产生寄存器（TMRx_EVEG）

偏移地址：0x14

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留									TRGG	保留	C4G	C3G	C2G	C1G	UEVG
res									rw	res	w	w	w	w	w

位 15：7	保留，始终读为 0。
位 6	**TRGG**：产生触发事件 （Trigger generation） 该位由软件置'1'，用于产生一个触发事件，由硬件自动清'0'。 0：无动作； 1：TMRx_STS 寄存器的 TRGIF=1，若开启对应的中断和 DMA，则产生相应的中断和 DMA。
位 5	保留，始终读为 0。
位 4	**C4G**：产生捕获/比较 4 事件 （Capture/compare 4 generation） 参考 C1G 描述。
位 3	**C3G**：产生捕获/比较 3 事件 （Capture/compare 3 generation） 参考 C1G 描述。
位 2	**C2G**：产生捕获/比较 2 事件 （Capture/compare 2 generation） 参考 C1G 描述。
位 1	**C1G**：产生捕获/比较 1 事件 （Capture/compare 1 generation） 该位由软件置'1'，用于产生一个捕获/比较事件，由硬件自动清'0'。 0：无动作； 1：在通道 CC1 上产生一个捕获/比较事件： **若通道 CC1 配置为输出：** 设置 C1IF=1，若开启对应的中断和 DMA，则产生相应的中断和 DMA。 **若通道 CC1 配置为输入：** 当前的计数器值捕获至 TMRx_CC1 寄存器；设置 C1IF=1，若开启对应的中断和 DMA，则产生相应的中断和 DMA。若 C1IF 已经为 1，则设置 C1OF=1。
位 0	**UEVG**：产生更新事件 （Update generation） 该位由软件置'1'，由硬件自动清'0'。 0：无动作； 1：重新初始化计数器，并产生一个更新事件。注意预分频器的计数器也被清'0'（但是预分频系数不变）。若在中心对称模式下或 DIR=0（向上计数）则计数器被清'0'，若 DIR=1（向下计数）则计数器取 TMRx_AR 的值。

10.2.4.7 捕获/比较模式寄存器1（TMRx_CCM1）

偏移地址：0x18

复位值：0x0000

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 CxSEL 定义。该寄存器其它位的作用在输入和输出模式下不同。OCxx 描述了通道在输出模式下的功能，ICxx 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：

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC2DIS	OC2MODE[2：0]			OC2PEN	OC2FEN	C2SEL[1：0]		OC1DIS	OC1MODE[2：0]			OC1PEN	OC1FEN	C1SEL[1：0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 15	**OC2DIS**：输出比较 2 清 0 使能 （Output compare 2 clear enable）
位 14：12	**OC2MODE[2：0]**：输出比较 2 模式 （Output compare 2 mode）
位 11	**OC2PEN**：输出比较 2 预装载使能 （Output compare 2 preload enable）
位 10	**OC2FEN**：输出比较 2 快速使能 （Output compare 2 fast enable）
位 9：8	**C2SEL[1：0]**：捕获/比较 2 选择 （Capture/Compare 2 selection） 该位定义通道的方向（输入/输出），及输入脚的选择： 00：CC2 通道被配置为输出； 01：CC2 通道被配置为输入，IC2 映射在 TI2 上； 10：CC2 通道被配置为输入，IC2 映射在 TI1 上； 11：CC2 通道被配置为输入，IC2 映射在 TRC 上。此模式仅工作在内部触发器输入被选中时（由 TMRx_SMC 寄存器的 TRGSEL 位选择）。 *注：C2SEL 仅在通道关闭时（TMRx_CCE 寄存器的 C2EN='0'）才是可写的。*
位 7	**OC1DIS**：输出比较 1 清 0 使能 （Output compare 1 clear enable） 0：OC1REF 不受 ETRF 输入的影响； 1：一旦检测到 ETRF 输入高电平，清除 OC1REF=0。
位 6：4	**OC1MODE[2：0]**：输出比较 1 模式 （Output compare 1 enable） 该 3 位定义了输出参考信号 OC1REF 的动作，而 OC1REF 决定了 OC1 的值。OC1REF 是高电平有效，而 OC1 的有效电平取决于 C1P 位。 000：冻结。输出比较寄存器 TMRx_CC1 与计数器 TMRx_CNT 间的比较对 OC1REF 不起作用； 001：匹配时设置通道 1 为有效电平。当计数器 TMRx_CNT 的值与捕获 / 比较寄存器 1 （TMRx_CC1）相同时，强制 OC1REF 为高。 010：匹配时设置通道 1 为无效电平。 当计数器 TMRx_CNT 的值与捕获 / 比较寄存器 1 （TMRx_CC1）相同时，强制 OC1REF 为低。 011：翻转。当 TMRx_CC1=TMRx_CNT 时，翻转 OC1REF 的电平。 100：强制为无效电平。强制 OC1REF 为低。 101：强制为有效电平。强制 OC1REF 为高。 110：PWM 模式 1— 在向上计数时，一旦 TMRx_CNT<TMRx_CC1 时通道 1 为有效电平，否则为无效电平；在向下计数时，一旦 TMRx_CNT>TMRx_CC1 时通道 1 为无效电平（OC1REF=0），否则为有效电平（OC1REF=1）。 111：PWM 模式 2— 在向上计数时，一旦 TMRx_CNT<TMRx_CC1 时通道 1 为无效电平，否则为有效电平；在向下计数时，一旦 TMRx_CNT>TMRx_CC1 时通道 1 为有效电平，否则为无效电平。 *注：在 PWM 模式 1 或 PWM 模式 2 中，只有当比较结果改变了或在输出比较模式中从冻结模式切换到 PWM 模式时，OC1REF 电平才改变。*

位 3	**OC1PEN**：输出比较 1 预装载使能 （Output compare 1 preload enable） 0：禁止 TMRx_CC1 寄存器的预装载功能，可随时写入 TMRx_CC1 寄存器，并且新写入的数值立即起作用。 1：开启 TMRx_CC1 寄存器的预装载功能，读写操作仅对预装载寄存器操作，TMRx_CC1 的预装载值在更新事件到来时被传送至当前寄存器中。 *注：仅在单脉冲模式下（TMRx_CTRL1 寄存器的 OPMODE='1'），可以在未确认预装载寄存器情况下使用 PWM 模式，否则其动作不确定。*
位 2	**OC1FEN**：输出比较 1 快速使能（Output compare 1 fast enable） 该位用于加快 CC 输出对触发器输入事件的响应。 0：根据计数器与 CC1 的值，CC1 正常操作，即使触发器是打开的。当触发器的输入出现一个有效沿时，激活 CC1 输出的最小延时为 5 个时钟周期。 1：输入到触发器的有效沿的作用就像发生了一次比较匹配。因此，OC 被设置为比较电平而与比较结果无关。采样触发器的有效沿和 CC1 输出间的延时被缩短为 3 个时钟周期。 该位只在通道被配置成 PWM1 或 PWM2 模式时起作用。
位 1：0	**C1SEL[1：0]**：捕获/比较 1 选择 （Capture/Compare 1 selection） 这 2 位定义通道的方向（输入/输出），及输入脚的选择： 00：CC1 通道被配置为输出； 01：CC1 通道被配置为输入，IC1 映射在 TI1 上； 10：CC1 通道被配置为输入，IC1 映射在 TI2 上； 11：CC1 通道被配置为输入，IC1 映射在 TRC 上。此模式仅工作在内部触发器输入被选中时（由 TMRx_SMC 寄存器的 TRGSEL 位选择）。 *注：C1SEL 仅在通道关闭时（TMRx_CCE 寄存器的 C1EN='0'）才是可写的。*

输入捕获模式：

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IC2DF[3：0]				IC2DIV[1：0]		C2SEL[1：0]		IC1DF[3：0]				IC1DIV[1：0]		C1SEL[1：0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 15：12	**IC2DF[3：0]**：输入捕获 2 滤波器 （Input capture 2 filter）
位 11：10	**IC2DIV[1：0]**：输入/捕获 2 预分频器 （input capture 2 prescaler）
位 9：8	**C2SEL[1：0]**：捕获/比较 2 选择 （Capture/compare 2 selection） 这 2 位定义通道的方向（输入/输出），及输入脚的选择： 00：CC2 通道被配置为输出； 01：CC2 通道被配置为输入，IC2 映射在 TI2 上； 10：CC2 通道被配置为输入，IC2 映射在 TI1 上； 11：CC2 通道被配置为输入，IC2 映射在 TRC 上。此模式仅工作在内部触发器输入被选中时（由 TMRx_SMC 寄存器的 TRGSEL 位选择）。 *注：C2SEL 仅在通道关闭时（TMRx_CCE 寄存器的 C2EN='0'）才是可写的。*
位 7：4	**IC1DF[3：0]**：输入捕获 1 滤波器 （Input capture 1 filter） 这几位定义了 TI1 输入的采样频率及数字滤波器长度。数字滤波器由一个事件计数器组成，它记录到 N 个事件后会产生一个输出的跳变： 0000：无滤波器，以f_{DTS}采样 0001：采样频率 $f_{SAMPLING}=f_{CK_INT}$，N=2 0010：采样频率 $f_{SAMPLING}=f_{CK_INT}$，N=4 0011：采样频率 $f_{SAMPLING}=f_{CK_INT}$，N=8 0100：采样频率 $f_{SAMPLING}=f_{DTS}/2$，N=6 0101：采样频率 $f_{SAMPLING}=f_{DTS}/2$，N=8 0110：采样频率 $f_{SMPLING}=f_{DTS}/4$，N=6 0111：采样频率 $f_{SAMPLING}=f_{DTS}/4$，N=8 1000：采样频率$f_{SAMPLING}=f_{DTS}/8$，N=6 1001：采样频率$f_{SAMPLING}=f_{DTS}/8$，N=8 1010：采样频率$f_{SAMPLING}=f_{DTS}/16$，N=5 1011：采样频率$f_{SAMPLING}=f_{DTS}/16$，N=6 1100：采样频率$f_{SAMPLING}=f_{DTS}/16$，N=8 1101：采样频率$f_{SAMPLING}=f_{DTS}/32$，N=5 1110：采样频率$f_{SAMPLING}=f_{DTS}/32$，N=6 1111：采样频率$f_{SAMPLING}=f_{DTS}/32$，N=8

位 3：2	**IC1DIV[1：0]**：输入/捕获 1 预分频器 （Input capture 1 prescaler） 这 2 位定义了 CC1 输入（IC1）的预分频系数。 一旦 C1EN=’0’（TMRx_CCE 寄存器中），则预分频器复位。 00：无预分频器，捕获输入口上检测到的每一个边沿都触发一次捕获； 01：每 2 个事件触发一次捕获； 10：每 4 个事件触发一次捕获； 11：每 8 个事件触发一次捕获。
位 1：0	**C1SEL[1：0]**：捕获/比较 1 选择 （Capture/Compare 1 selection） 这 2 位定义通道的方向（输入/输出），及输入脚的选择： 00：CC1 通道被配置为输出； 01：CC1 通道被配置为输入，IC1 映射在 TI1 上； 10：CC1 通道被配置为输入，IC1 映射在 TI2 上； 11：CC1 通道被配置为输入，IC1 映射在 TRC 上。此模式仅工作在内部触发器输入被选中时（由 TMRx_SMC 寄存器的 TRGSEL 位选择）。 *注：C1SEL 仅在通道关闭时（TMRx_CCE 寄存器的 C1EN=’0’）才是可写的。*

10.2.4.8 捕获/比较模式寄存器2（TMRx_CCM2）

偏移地址：0x1C

复位值：0x0000 参看以上 CCM1 寄存器的描述

输出比较模式：

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC4 DIS	OC4MODE[2：0]			OC4 PEN	OC4 FEN	C4SEL[1：0]		OC3 DIS	OC3MODE[2：0]			OC3 PEN	OC3 FEN	C3SEL[1：0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 15	**OC4DIS**：输出比较 4 清 0 使能 （Output compare 4 clear enable）
位 14：12	**OC4MODE[2：0]**：输出比较 4 模式 （Output compare 4 mode）
位 11	**OC4PEN**：输出比较 4 预装载使能 （Output compare 4 preload enable）
位 10	**OC4FEN**：输出比较 4 快速使能 （Output compare 4 fast enable）
位 9：8	**C4SEL[1：0]**：捕获/比较 4 选择 （Capture/Compare 4 selection） 这 2 位定义通道的方向（输入/输出），及输入脚的选择： 00：CC4 通道被配置为输出； 01：CC4 通道被配置为输入，IC4 映射在 TI4 上； 10：CC4 通道被配置为输入，IC4 映射在 TI3 上； 11：CC4 通道被配置为输入，IC4 映射在 TRC 上。此模式仅工作在内部触发器输入被选中时（由 TMRx_SMC 寄存器的 TRGSEL 位选择）。 *注：C4SEL 仅在通道关闭时（TMRx_CCE 寄存器的 C4EN=’0’）才是可写的。*
位 7	**OC3DIS**：输出比较 3 清 0 使能 （Output compare 3 clear enable）
位 6：4	**OC3MODE[2：0]**：输出比较 3 模式 （Output compare 3 mode）
位 3	**OC3PEN**：输出比较 3 预装载使能 （Output compare 3 preload enable）
位 2	**OC3FEN**：输出比较 3 快速使能 （Output compare 3 fast enable）
位 1：0	**C3SEL[1：0]**：捕获/比较 3 选择 （Capture/Compare 3 selection） 这 2 位定义通道的方向（输入/输出），及输入脚的选择： 00：CC3 通道被配置为输出； 01：CC3 通道被配置为输入，IC3 映射在 TI3 上； 10：CC3 通道被配置为输入，IC3 映射在 TI4 上； 11：CC3 通道被配置为输入，IC3 映射在 TRGI 上。此模式仅工作在内部触发器输入被选中时（由 TMRx_SMC 寄存器的 TRGSEL 位选择）。 *注：C3SEL 仅在通道关闭时（TMRx_CCE 寄存器的 C3EN=’0’）才是可写的。*

输入捕获模式：

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IC4DF[3：0]				IC4DIV[1：0]		C4SEL[1：0]		IC3DF[3：0]				IC3DIV[1：0]		C3SEL[1：0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 15：12	**IC4DF[3：0]：**输入捕获 4 滤波器 （Input capture 4 filter）
位 11：10	**IC4DIV[1：0]：**输入/捕获 4 预分频器 （input capture 4 prescaler）
位 9：8	**C4SEL[1：0]：**捕获/比较 4 选择 （Capture/compare 4 selection） 这 2 位定义通道的方向（输入/输出），及输入脚的选择： 00：CC4 通道被配置为输出； 01：CC4 通道被配置为输入，IC4 映射在 TI4 上； 10：CC4 通道被配置为输入，IC4 映射在 TI3 上； 11：CC4 通道被配置为输入，IC4 映射在 TRC 上。此模式仅工作在内部触发器输入被选中时（由 TMRx_SMC 寄存器的 TRGSEL 位选择）。 *注：C4SEL 仅在通道关闭时（TMRx_CCE 寄存器的 C4EN='0'）才是可写的。*
位 7：4	**IC3DF[3：0]：**输入捕获 3 滤波器 （Input capture 3 filter）
位 3：2	**IC3DIV[1：0]：**输入/捕获 3 预分频器 （Input capture 3 prescaler）
位 1：0	**C3SEL[1：0]：**捕获/比较 3 选择 （Capture/Compare 3 selection） 这 2 位定义通道的方向（输入/输出），及输入脚的选择： 00：CC3 通道被配置为输出； 01：CC3 通道被配置为输入，IC3 映射在 TI3 上； 10：CC3 通道被配置为输入，IC3 映射在 TI4 上； 11：CC3 通道被配置为输入，IC3 映射在 TRC 上。此模式仅工作在内部触发器输入被选中时（由 TMRx_SMC 寄存器的 TRGSEL 位选择）。 *注：C3SEL 仅在通道关闭时（TMRx_CCE 寄存器的 C3EN='0'）才是可写的。*

10.2.4.9 捕获/比较使能寄存器（TMRx_CCE）

偏移地址：0x20

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留		C4P	C4EN	保留		C3P	C3EN	保留		C2P	C2EN	保留		C1P	C1EN
res		rw	rw	res		rw	rw	res		rw	rw	res		rw	rw

位 15：14	保留，始终读为 0。
位 13	**C4P：**输入/捕获 4 输出极性 （Capture/Compare 4 output polarity） 参考 C1P 的描述。
位 12	**C4EN：**输入/捕获 4 输出使能 （Capture/Compare 4 output enable） 参考 C1EN 的描述。
位 11：10	保留，始终读为 0。
位 9	**C3P：**输入/捕获 3 输出极性 （Capture/Compare 3 output polarity） 参考 C1P 的描述。
位 8	**C3EN：**输入/捕获 3 输出使能 （Capture/Compare 3 output enable） 参考 C1EN 的描述。
位 7：6	保留，始终读为 0。

位 5	**C2P**：输入/捕获 2 输出极性 （Capture/Compare 2 output polarity） 参考 C1P 的描述。
位 4	**C2EN**：输入/捕获 2 输出使能 （Capture/Compare 2 output enable） 参考 C1EN 的描述。
位 3：2	保留，始终读为 0。
位 1	**C1P**：输入/捕获 1 输出极性 （Capture/Compare 1 output polarity） **CC1 通道配置为输出：** 0：OC1 高电平有效 1：OC1 低电平有效 **CC1 通道配置为输入：** 该位选择是 IC1 还是 IC1 的反相信号作为触发或捕获信号。 0：不反相：捕获发生在 IC1 的上升沿；当用作外部触发器时，IC1 不反相。 1：反相：捕获发生在 IC1 的下降沿；当用作外部触发器时，IC1 反相。
位 0	**C1EN**：输入/捕获 1 输出使能 （Capture/Compare 1 output enable） **CC1 通道配置为输出：** 0： 关闭－ OC1 禁止输出。 1： 开启－ OC1 信号输出到对应的输出引脚。 **CC1 通道配置为输入：** 该位决定了计数器的值是否能捕获入 TMRx_CC1 寄存器。 0：捕获禁止； 1：捕获使能。

表 10-5 标准OCx通道的输出控制位

CxEN 位	OCx 输出状态
0	禁止输出（OCx=0，OCx_EN=0）
1	OCx = OCxREF + 极性，OCx_EN=1

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

10.2.4.10 计数器（TMRx_CNT）

偏移地址：0x24

复位值：0x0000

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

位 31：16	**CNT[31：16]**：计数器的值 （Counter value） 当 TMR2 或 TMR5 开启增强模式时（TMR_CTRL1 中的 PMEN 位），CNT 被扩展为 32 位。
位 15：0	**CNT[15：0]**：计数器的值 （Counter value）

10.2.4.11 预分频器（TMRx_DIV）

偏移地址：0x28

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DIV[15：0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 15：0	**DIV[15：0]：** 预分频器的值 （Prescaler value） 计数器的时钟频率 CK_CNT 等于 f_{CK_DIV}/（DIV[15：0]+1）。 DIV 包含了当更新事件产生时装入当前预分频器寄存器的值。

10.2.4.12 自动重装载寄存器（TMRx_AR）

偏移地址：0x2C

复位值：0x0000

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

位 31：16	**AR[31：16]：** 自动重装载的值 （Auto reload value） 当 TMR2 或 TMR5 开启增强模式时（TMR_CTRL1 中的 PMEN 位），AR 被扩展为 32 位。
位 15：0	**AR[15：0]：** 自动重装载的值 （Auto reload value） AR 包含了将要传送至实际的自动重装载寄存器的数值。当自动重装载的值为空时，计数器不工作。 详细参考 10.2.3.1：有关 AR 的更新和动作。

10.2.4.13 捕获/比较寄存器1（TMRx_CC1）

偏移地址：0x34

复位值：0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CC1[31：16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CC1[15：0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31：16	**CC1[31：16]：** 捕获/比较 1 的值 （Capture/Compare 1 value） 当 TMR2 或 TMR5 开启增强模式时（TMR_CTRL1 中的 PMEN 位），CC1 被扩展为 32 位。

位 15：0	**CC1[15：0]：** 捕获/比较 1 的值 （Capture/Compare 1 value） **若 CC1 通道配置为输出：** CC1 包含了装入当前捕获/比较 1 寄存器的值（预装载值）。 如果在 TMRx_CCM1 寄存器（OC1PEN 位）中未选择预装载特性，写入的数值会被立即传输至当前寄存器中。否则只有当更新事件发生时，此预装载值才传输至当前捕获/比寄存器 1 中。 当前捕获/比较寄存器参与同计数器 TMRx_CNT 的比较，并在 OC1 端口上产生输出信号。 **若 CC1 通道配置为输入：** CC1 包含了由上一次输入捕获 1 事件（IC1）传输的计数器值。

10.2.4.14 捕获/比较寄存器2（TMRx_CC2）

偏移地址：0x38

复位值：0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CC2[31：16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CC2[15：0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31：16	**CC2[31：16]：**捕获/比较 2 的值 （Capture/Compare 2 value） 当 TMR2 或 TMR5 开启增强模式时（TMR_CTRL1 中的 PMEN 位），CC2 被扩展为 32 位。
位 15：0	**CC2[15：0]：** 捕获/比较 2 的值 （Capture/Compare 2value） **若 CC2 通道配置为输出：** CC2 包含了装入当前捕获/比较 2 寄存器的值（预装载值）。 如果在 TMRx_CCM2 寄存器（OC2PEN 位）中未选择预装载特性，写入的数值会被立即传输至当前寄存器中。否则只有当更新事件发生时，此预装载值才传输至当前捕获/比较 2 寄存器中。当前捕获/比较寄存器参与同计数器 TMRx_CNT 的比较，并在 OC2 端口上产生输出信号。 **若 CC2 通道配置为输入：** CC2 包含了由上一次输入捕获 2 事件（IC2）传输的计数器值。

10.2.4.15 捕获/比较寄存器3（TMRx_CC3）

偏移地址：0x3C

复位值：0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CC3[31：16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CC3[15：0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31：16	**CC3[31：16]：**捕获/比较 3 的值 （Capture/Compare3 value） 当 TMR2 或 TMR5 开启增强模式时（TMR_CTRL1 中的 PMEN 位），CC3 被扩展为 32 位。

位 15：0	**CC3[15：0]：**捕获/比较 3 的值 （Capture/Compare 3value） **若 CC3 通道配置为输出：** CC3 包含了装入当前捕获/比较 3 寄存器的值（预装载值）。 如果在 TMRx_CCM3 寄存器（OC3PEN 位）中未选择预装载特性，写入的数值会被立即传输至当前寄存器中。否则只有当更新事件发生时，此预装载值才传输至当前捕获/比较寄存器 3 中。当前捕获/比较寄存器参与同计数器 TMRx_CNT 的比较，并在 OC3 端口上产生输出信号。 **若 CC3 通道配置为输入：** CC3 包含了由上一次输入捕获 3 事件（IC3）传输的计数器值。

10.2.4.16 捕获/比较寄存器4（TMRx_CC4）

偏移地址：0x40

复位值：0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CC4[31：16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CC4[15：0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31：16	**CC4[31：16]：**捕获/比较 4 的值 （Capture/Compare4 value） 当 TMR2 或 TMR5 开启增强模式时（TMR_CTRL1 中的 PMEN 位），CC4 被扩展为 32 位。
位 15：0	**CC4[15：0]：** 捕获/比较 4 的值 （Capture/Compare 4value） **若 CC4 通道配置为输出：** CC4 包含了装入当前捕获/比较 4 寄存器的值（预装载值）。 如果在 TMRx_CCM4 寄存器（OC4PEN 位）中未选择预装载特性，写入的数值会被立即传输至当前寄存器中。否则只有当更新事件发生时，此预装载值才传输至当前捕获/比较寄存器 4 中。当前捕获/比较寄存器参与同计数器 TMRx_CNT 的比较，并在 OC4 端口上产生输出信号。 **若 CC4 通道配置为输入：** CC4 包含了由上一次输入捕获 4 事件（IC4）传输的计数器值。

10.2.4.17 DMA控制寄存器（TMRx_DMAC）

偏移地址：0x48

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留			DBLEN[4：0]					保留			ADDR[4：0]				
res			rw	rw	rw	rw	rw	res			rw	rw	rw	rw	rw

位 15：13	保留，始终读为 0。
位 12：8	**DBLEN[4：0]：**DMA 连续传送长度 （DMA burst length） 这些位定义了 DMA 在连续模式下的传送长度（当对 TMRx_DMABA 寄存器进行读或写时，定时器则进行一次连续传送），即：定义传输的字节数目： 00000：1 个字节 00001：2 个字节 00010：3 个字节 .... 10001：18 个字节

位 7：5	保留，始终读为 0。
位 4：0	**ADDR[4：0]：**DMA 基地址 （DMA base address） 这些位定义了 DMA 在连续模式下的基地址（当对 TMRx_DMABA 寄存器进行读或写时），ADDR 定义为从 TMRx_CTRL1 寄存器所在地址开始的偏移量： 00000：TMRx_CTRL1， 00001：TMRx_CTRL2， 00010：TMRx_SMC， .....

10.2.4.18 连续模式的DMA地址（TMRx_DMABA）

偏移地址：0x4C

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DMABA[15：0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 15：0	**DMABA[15：0]：**DMA 连续传送寄存器（DMA register for burst accesses） 对 TMRx_DMABA 寄存器的读或写会导致对以下地址所在寄存器的存取操作：TMRx_CTRL1 地址 + ADDR + DMA 索引，其中： “TMRx_CTRL1 地址”是控制寄存器 1（TMRx_CTRL1）所在的地址； “ADDR”是 TMRx_DMAC 寄存器中定义的基地址； “DMA 索引”是由 DMA 自动控制的偏移量，它取决于 TMRx_DMAC 寄存器中定义的 DBLEN。

10.3 通用定时器（TMR9到TMR14）

10.3.1 TMRx简介

通用定时器是一个通过可编程预分频器驱动的16位自动装载计数器构成。它适用于多种场合，包括测量输入信号的脉冲长度（输入捕获）或者产生输出波形（输出比较和PWM）。

使用定时器预分频器和RCC时钟控制器预分频器，脉冲长度和波形周期可以在几个微秒到几个毫秒间调整。

每个定时器都是完全独立的，没有互相共享任何资源。它们可以一起同步操作，参见10.3.3.12节。

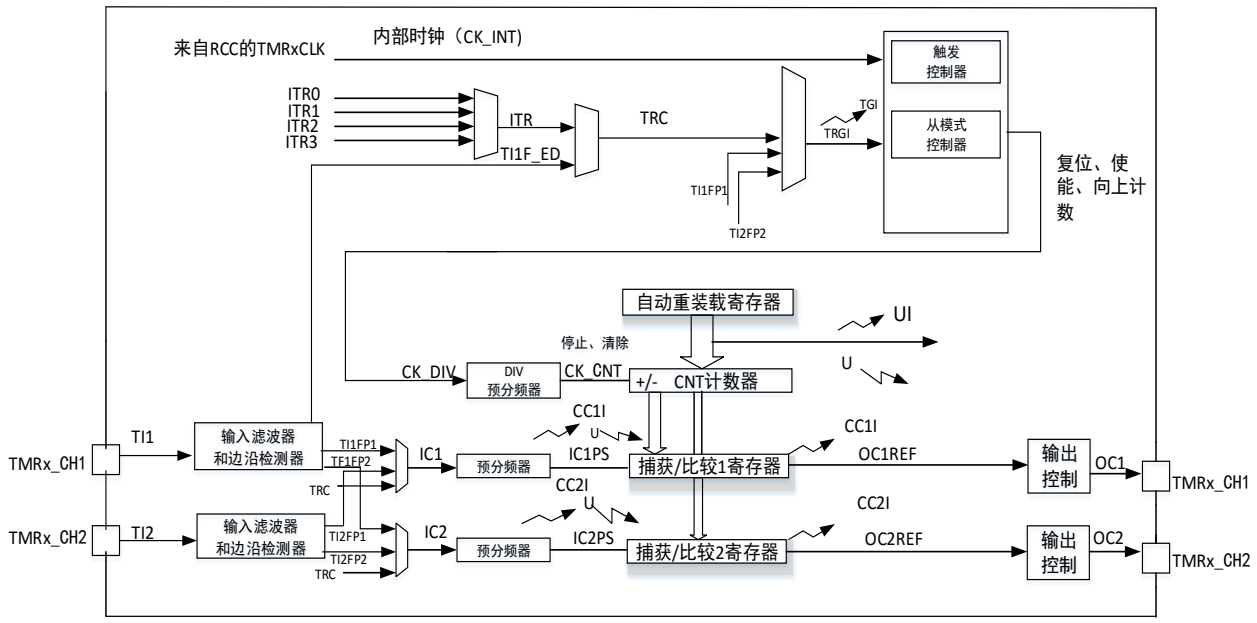
10.3.2 TMRx主要功能

10.3.2.1 TMR9和TMR12主要功能

通用TMRx （TMR9和TMR12）定时器功能包括：

- 16位向上自动装载计数器
- 16位可编程（可以实时修改）预分频器，计数器时钟频率的分频系数为1～65536之间的任意数值
- 2个独立通道：
  - 输入捕获
  - 输出比较
  - PWM生成（边缘对齐模式）
  - 单脉冲模式输出
- 使用外部信号控制定时器和定时器互连的同步电路
- 如下事件发生时产生中断：
  - 更新：计数器向上溢出，计数器初始化（通过软件或者内部）
  - 触发事件（计数器启动、停止、初始化或者由内部触发计数）
  - 输入捕获
  - 输出比较

图 10-57 通用定时器TMR9/12框图



注意： Reg *根据控制位的设定，在 U 事件时传送预加载寄存器的内容至工作寄存器*

事件

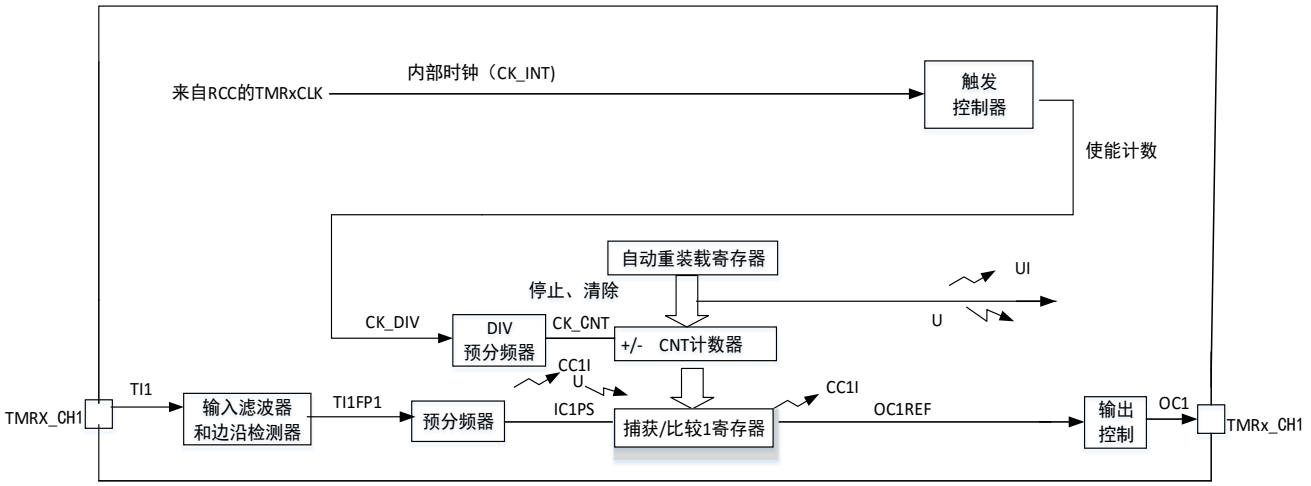
中断

10.3.2.2 TMR10、TMR11、TMR13和 TMR14主要功能

通用 TMRx （TMR10、TMR11、TMR13、TMR14）定时器功能包括：

- 16位向上自动装载计数器
- 16位可编程（可以实时修改）预分频器，计数器时钟频率的分频系数为1～65536之间的任意数值
- 1个独立通道：
  - 输入捕获
  - 输出比较
  - PWM生成（边缘对齐模式）
  - 单脉冲模式输出
- 如下事件发生时产生中断：
  - 更新：计数器向上溢出，计数器初始化（通过软件或者内部触发）
  - 输入捕获
  - 输出比较

图 10-58 通用定时器TMR10/11/13/14框图



注意： Reg *根据控制位的设定，在 U 事件时传送预加载寄存器的内容至工作寄存器*

事件

中断和 DMA 输出

10.3.3 TMRx功能描述

10.3.3.1 时基单元

可编程通用定时器的主要部分是一个 16 位计数器和与其相关的自动装载寄存器。这个计数器可以向上计数。此计数器时钟由预分频器分频得到。

计数器、自动装载寄存器和预分频器寄存器可以由软件读写，在计数器运行时仍可以读写。

时基单元包含：

- 计数器寄存器（TMRx_CNT）
- 预分频器寄存器 （TMRx_DIV）
- 自动装载寄存器 （TMRx_AR）

自动装载寄存器是预先装载的，写或读自动重装载寄存器将访问预装载寄存器。根据在 TMRx_CTRL1 寄存器中的自动装载预装载使能位（ARPEN）的设置，预装载寄存器的内容被立即或在每次的更新事件 UEV 时传送到影子寄存器。当计数器达到溢出条件并当 TMRx_CTRL1 寄存器中的 UEVDIS 位等于'0'时，产生更新事件。更新事件也可以由软件产生。随后会详细描述每一种配置下更新事件的产生。

计数器由预分频器的时钟输出 CK_CNT 驱动，仅当设置了计数器 TMRx_CTRL1 寄存器中的计数器使能位（CNTEN）时，CK_CNT 才有效。（有关计数器使能的细节，请参见控制器的从模式描述）。

注意： *真正的计数器使能信号 CNT_EN 是在 CNTEN 的一个时钟周期后被设置。*

预分频器

预分频器可以将计数器的时钟频率按 1 到 65536 之间的任意值分频。它是基于一个（在 TMRx_DIV 寄存器中的）16 位寄存器控制的 16 位计数器。这个控制寄存器带有缓冲器，它能够在工作时被改变。新的预分频器参数在下一次更新事件到来时被采用。

图 10-59 和图 10-60 给出了在预分频器运行时，更改计数器参数的例子。

图 10-59 当预分频器的参数从1变到2时，计数器的时序图

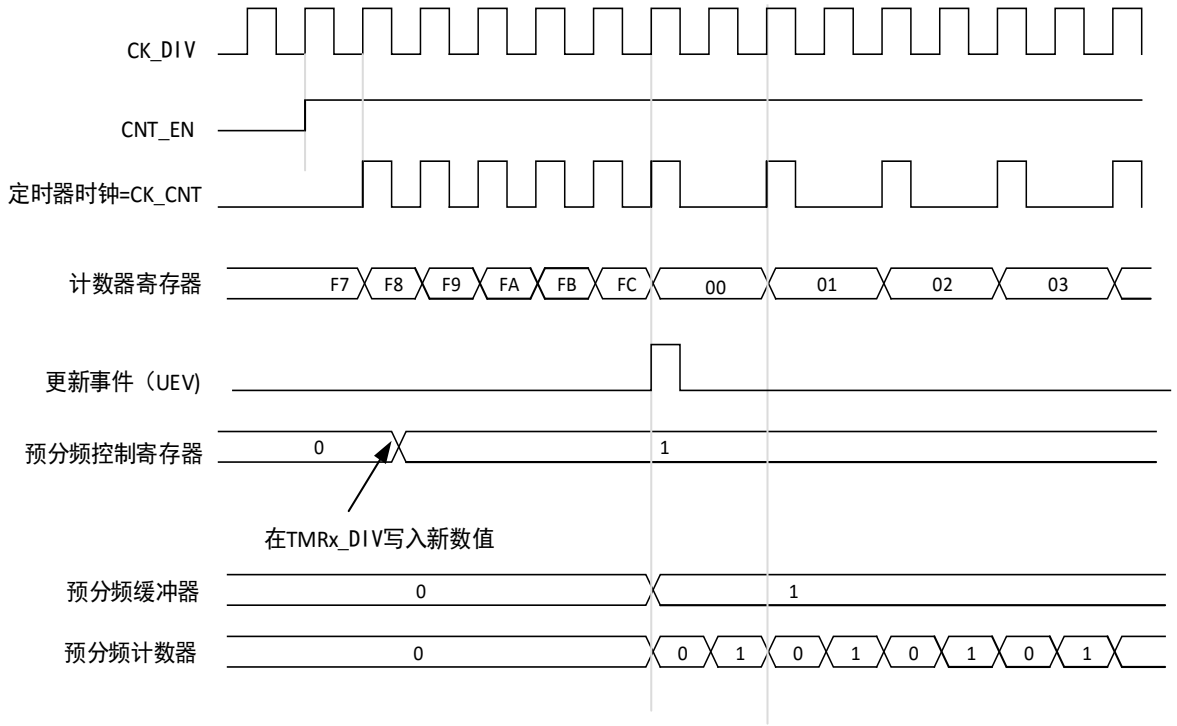
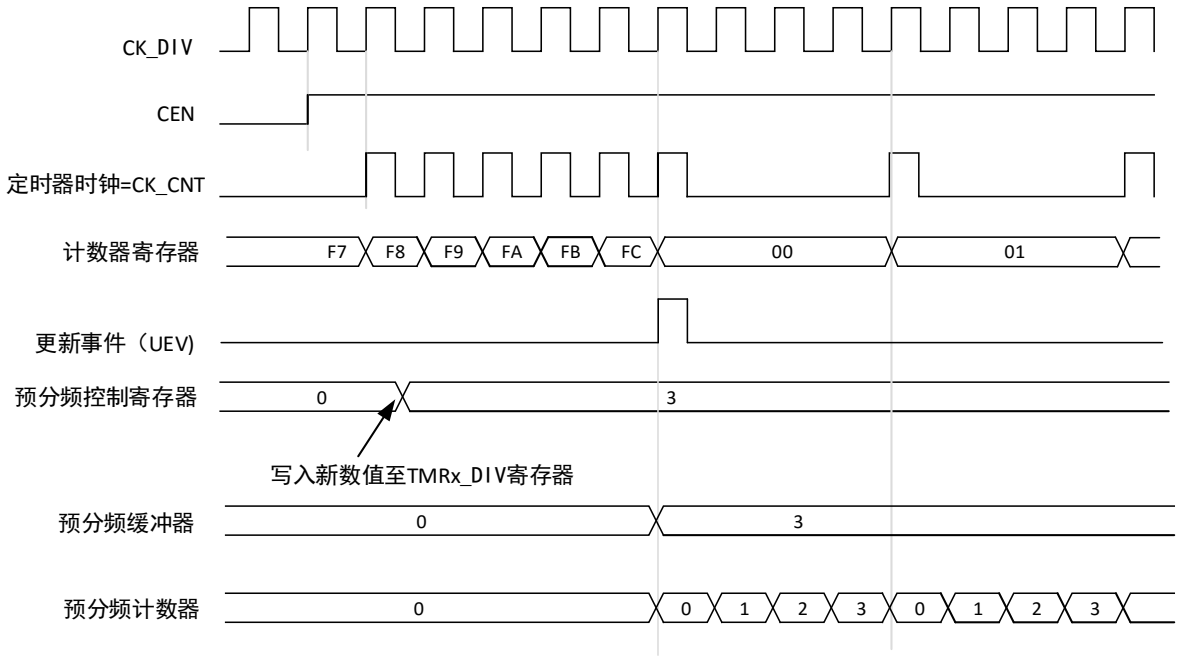


图 10-60当预分频器的参数从1变到4时，计数器的时序图



10.3.3.2 计数器模式

向上计数模式

在向上计数模式中，计数器从 0 计数到自动加载值（TMRx_AR 计数器的内容），然后重新从 0 开始计数并且产生一个计数器溢出事件。

每次计数器溢出时可以产生更新事件，在 TMRx_EVEG 寄存器中（通过软件方式,TMR9 和 12 还可以使用从模式控制器）设置 UEVG 位也同样可以产生一个更新事件。

设置 TMRx_CTRL1 寄存器中的 UEVDIS 位，可以禁止更新事件；这样可以避免在向预装载寄存器中写入新值时更新影子寄存器。在 UEVDIS 位被清'0'之前，将不产生更新事件。但是在应该产生更新事件时，

计数器仍会被清'0'，同时预分频器的计数也被清 0（但预分频系数不变）。此外，如果设置了 TMRx_CTRL1 寄存器中的 UEVRS 位（选择更新请求），设置 UEVG 位将产生一个更新事件 UEV，但硬件不设置 UEVIF 标志（即不产生中断或 DMA 请求）；这是为了避免在捕获模式下清除计数器时，同时产生更新和捕获中断。

当发生一个更新事件时，所有的寄存器都被更新，硬件同时（依据 UEVRS 位）设置更新标志位（TMRx_STS 寄存器中的 UEVIF 位）。

- 预分频器的缓冲区被置入预装载寄存器的值（TMRx_DIV寄存器的内容）。
- 自动装载影子寄存器被重新置入预装载寄存器的值（TMRx_AR）。

下图给出一些例子，当 TMRx_AR=0x36 时计数器在不同时钟频率下的动作。

图 10-61 计数器时序图，内部时钟分频因子为1

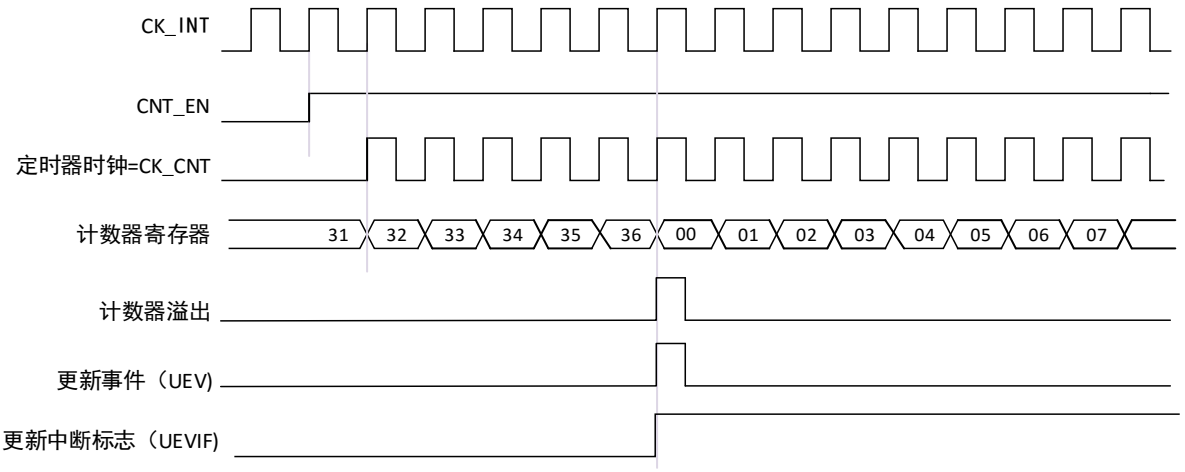


图 10-62 计数器时序图，内部时钟分频因子为2

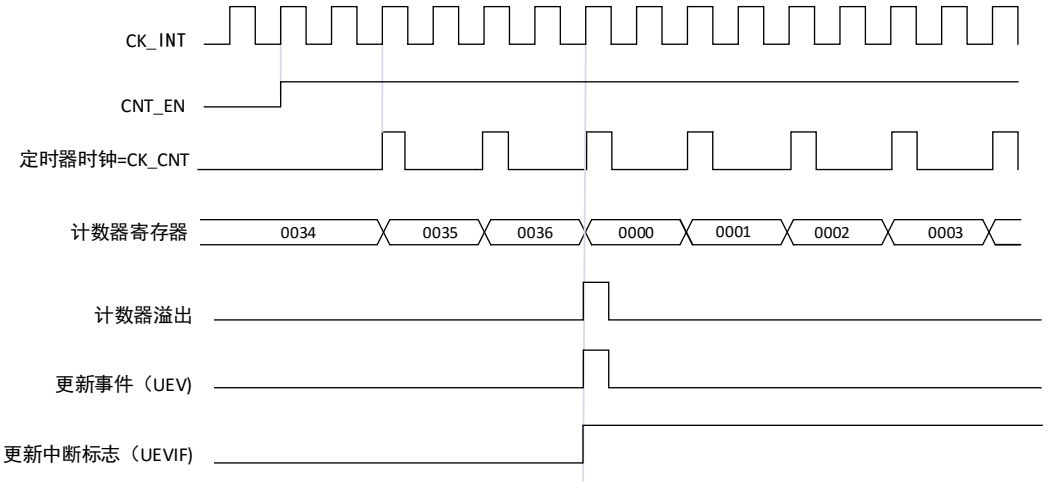


图 10-63 计数器时序图，内部时钟分频因子为4

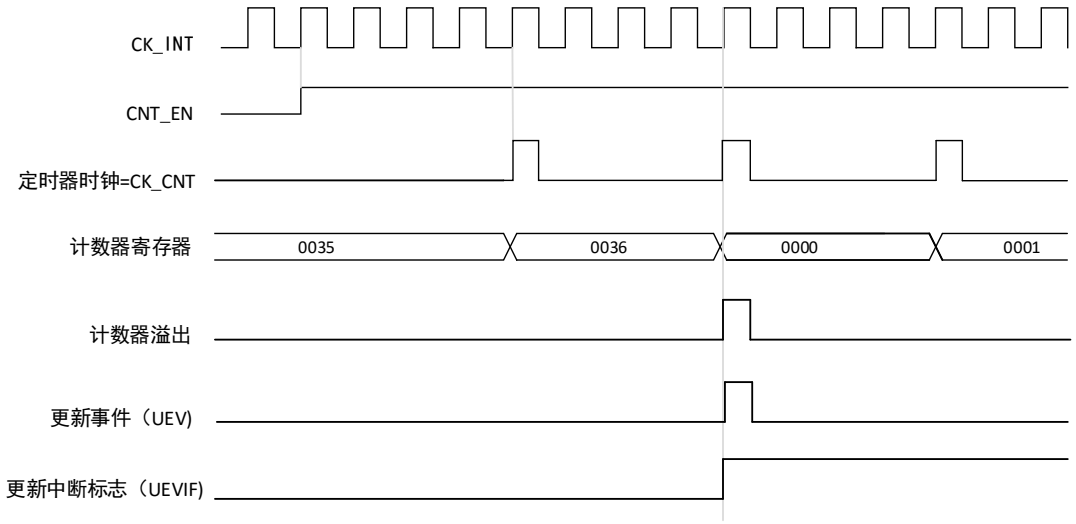


图 10-64 计数器时序图，内部时钟分频因子为N

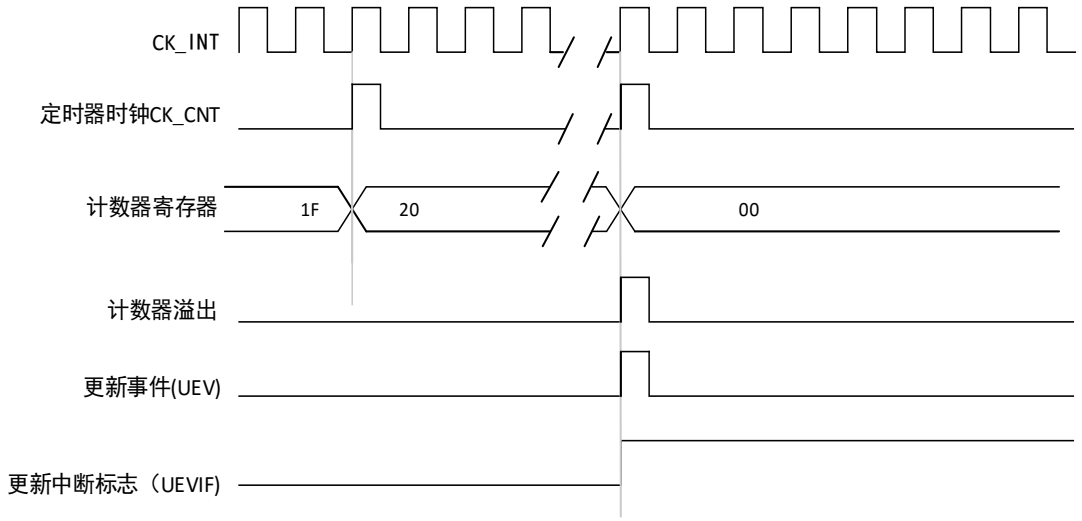


图 10-65 计数器时序图，当ARPEN=0时的更新事件（TMRx_AR没有预装入）

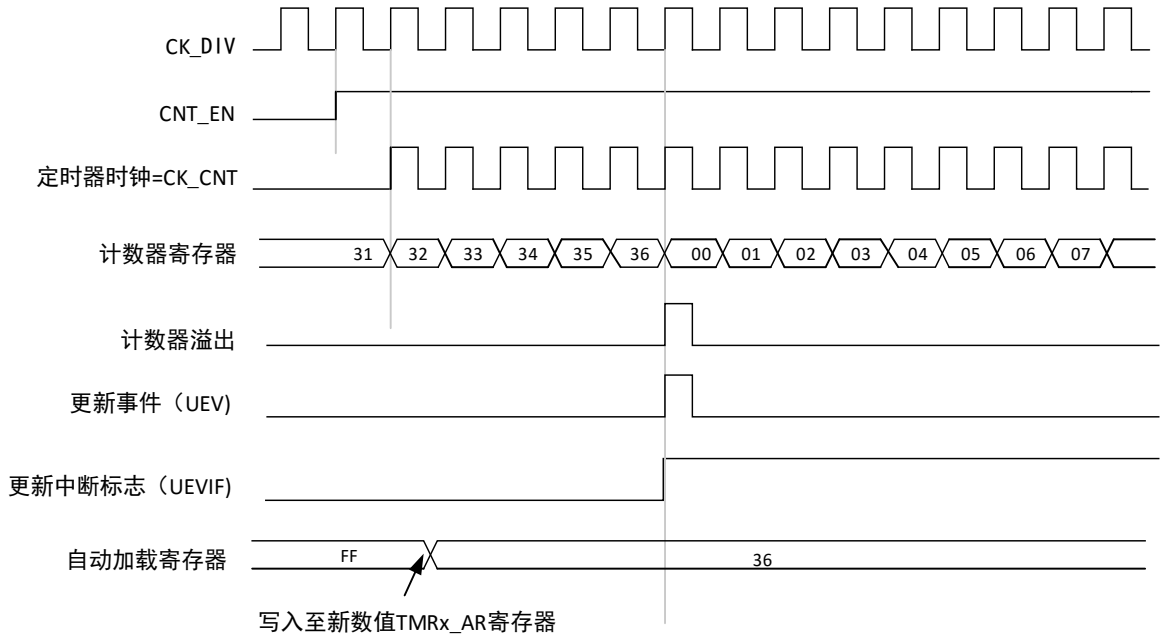
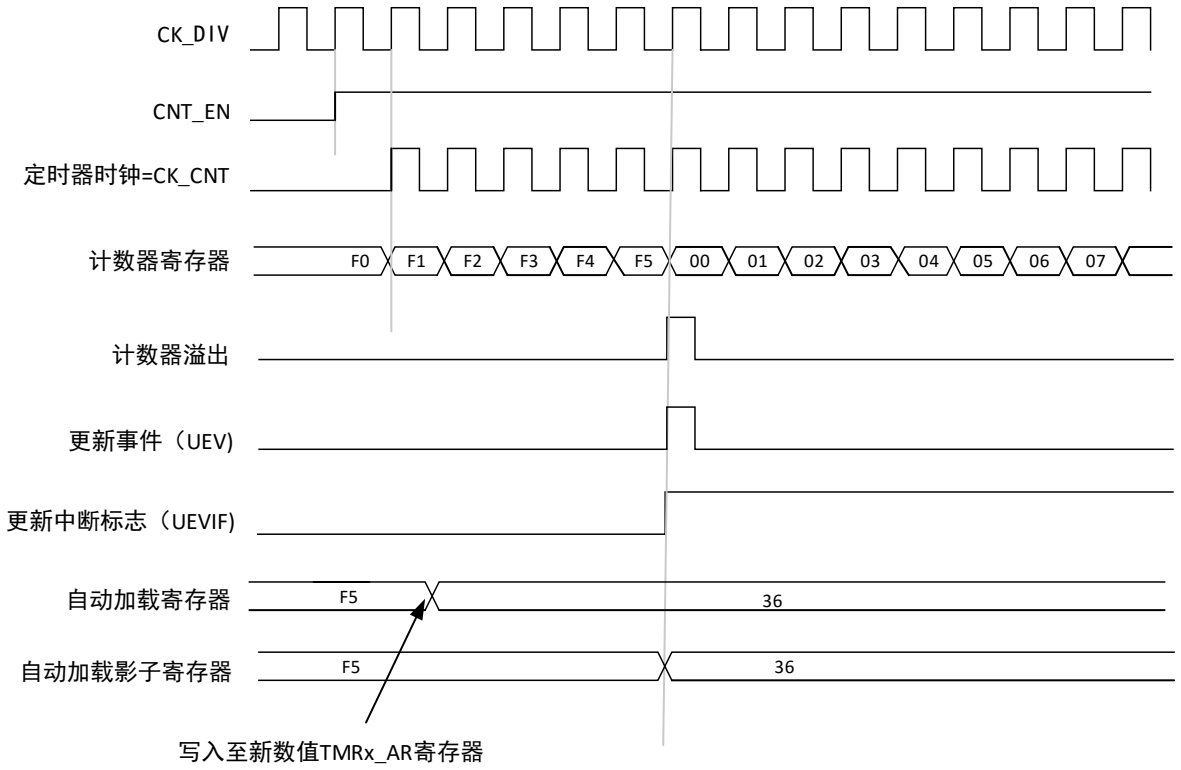


图 10-66 计数器时序图，当ARPEN=1时的更新事件（预装入了TMRx_AR）



10.3.3.3 时钟选择

计数器时钟可由下列时钟源提供：

- 内部时钟（CK_INT）
- 外部时钟模式1（仅对TMR9和TMR12）：外部输入脚（TIx）
- 内部触发输入（ITRx）（仅对TMR9和TMR12）：连接另外一个定时器的触发信号。如可以配置一个定时器Timer1而作为另一个定时器Timer2的预分频器。参见10.3.3.12。

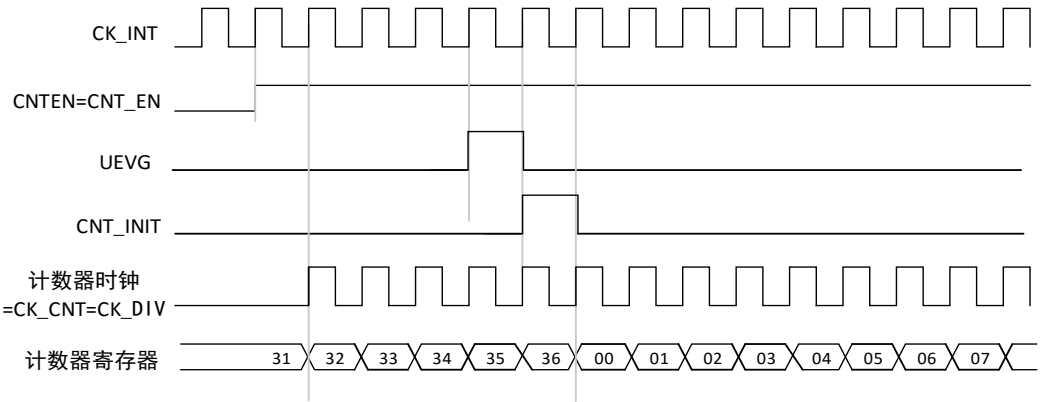
内部时钟源（CK_INT）

对于 TMR10/TMR11 和 TMR13/TMR14，内部时钟源是默认的时钟源。

对于 TMR9 和 TMR12，如果禁止了从模式控制器（TMRx_SMC 寄存器的 SMSEL=000），则 CNTEN、DIR（TMRx_CTRL1 寄存器）和 UEVG 位（TMRx_EVEG 寄存器）是事实上的控制位，并且只能被软件修改（UEVG 位仍被自动清除）。只要 CNTEN 位被写成'1'，预分频器的时钟就由内部时钟 CK_INT 提供。

下图显示了控制电路和向上计数器在一般模式下，不带预分频器时的操作。

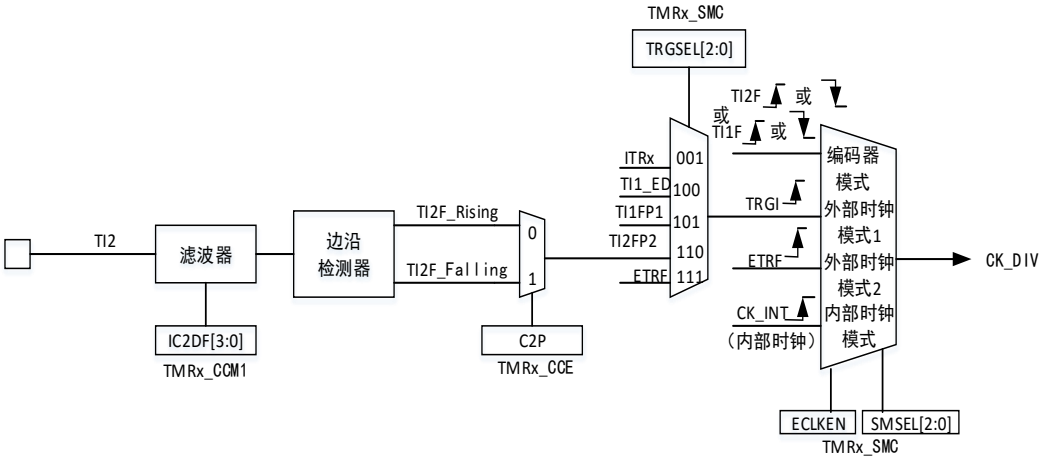
图 10-67 一般模式下的控制电路，内部时钟分频因子为1



外部时钟源模式 1（TMR9 和 TMR12）

当 TMRx_SMC 寄存器的 SMSEL=111 时，此模式被选中。计数器可以在选定输入端的每个上升沿或下降沿计数。

图 10-68 TI2外部时钟连接例子



例如，要配置向上计数器在 T12 输入端的上升沿计数，使用下列步骤：

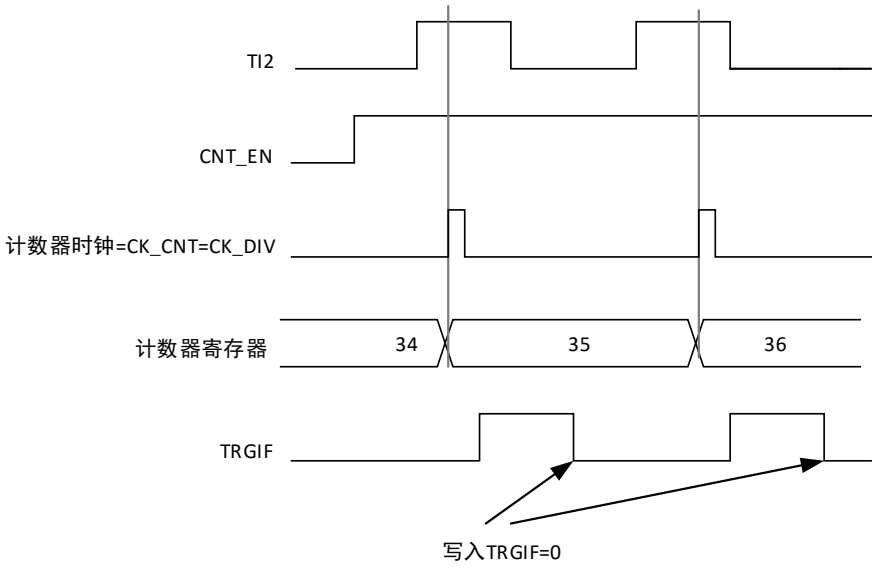
1. 配置 TMRx_CCM1 寄存器 C2SEL='01'，配置通道 2 检测 TI2 输入的上升沿；
2. 配置 TMRx_CCM1 寄存器的 IC2DF[3：0]，选择输入滤波器带宽（如果不需要滤波器，保持 IC2DF=0000）；

注意： *捕获预分频器不用作触发，所以不需要对它进行配置；*

3. 配置 TMRx_CCE 寄存器的 C2P='0'，选定上升沿极性；
4. 配置 TMRx_SMC 寄存器的 SMSEL='111'，选择定时器外部时钟模式 1；
5. 配置 TMRx_SMC 寄存器中的 TRGSEL='110'，选定 TI2 作为触发输入源；
6. 设置 TMRx_CTRL1 寄存器的 CNTEN='1'，启动计数器

当上升沿出现在 TI2，计数器计数一次，且 TRGIF 标志被设置。在 TI2 的上升沿和计数器实际时钟之间的延时，取决于在 TI2 输入端的重新同步电路。

图 10-69 外部时钟模式1下的控制电路

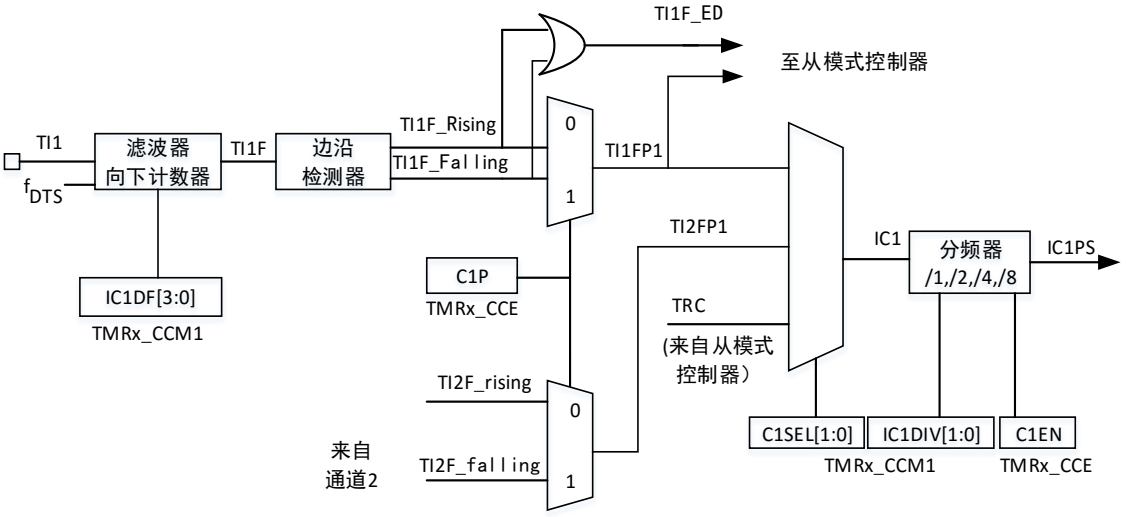


10.3.3.4 捕获/比较通道

每一个捕获/比较通道都是围绕着一个捕获/比较寄存器（包含影子寄存器），包括捕获的输入部分（数字滤波、多路复用和预分频器）和输出部分（比较器和输出控制）。

下面几张图是一个捕获/比较通道概览。输入部分对相应的 TIx 输入信号采样，并产生一个滤波后的信号 TIxF。然后，一个带极性选择的边缘检测器产生一个信号（TIxFPx），它可以作为从模式控制器的输入触发或者作为捕获控制。该信号通过预分频进入捕获寄存器。

图 10-70 捕获/比较通道（如：通道1输入部分）



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

图 10-71 捕获/比较通道1的主电路

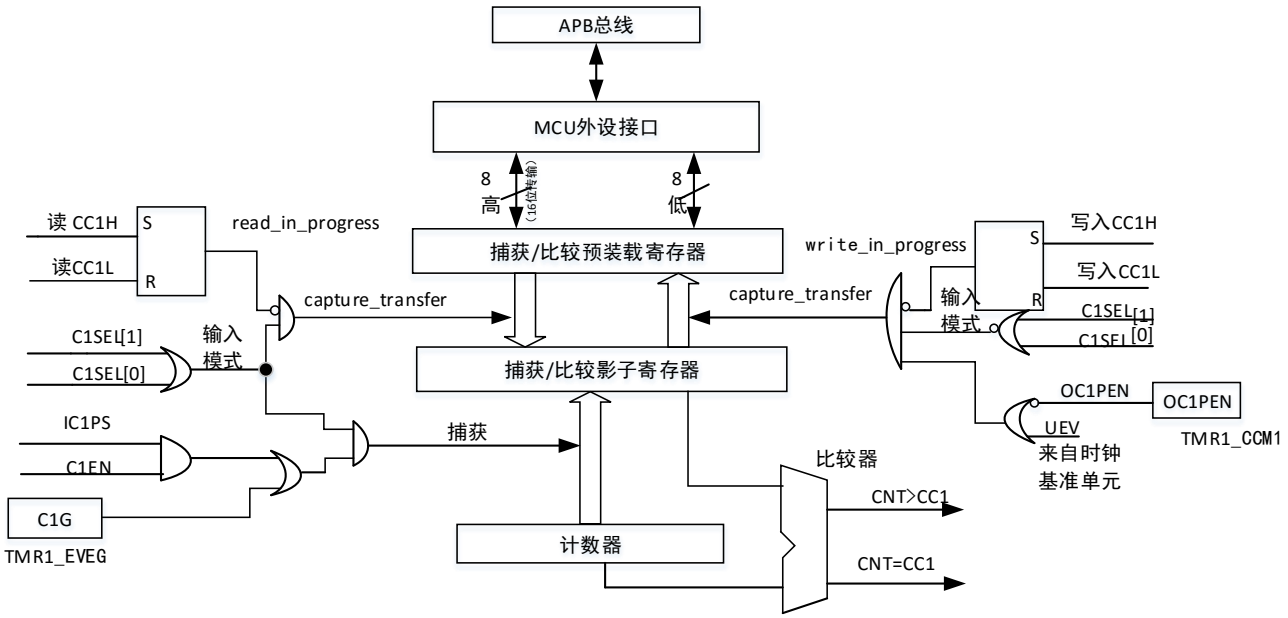
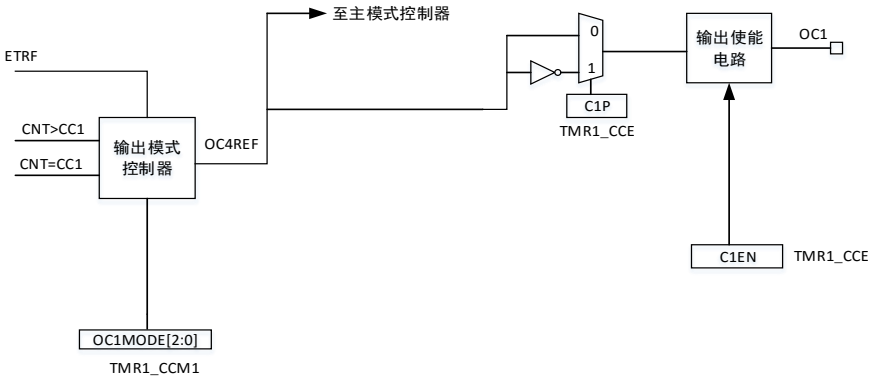


图 10-72 捕获/比较通道的输出部分（通道1）



捕获/比较模块由一个预装载寄存器和一个影子寄存器组成。读写过程仅操作预装载寄存器。

在捕获模式下，捕获发生在影子寄存器上，然后再复制到预装载寄存器中。

在比较模式下，预装载寄存器的内容被复制到影子寄存器中，然后影子寄存器的内容和计数器进行比较。

10.3.3.5 输入捕获模式

在输入捕获模式下，当检测到 ICx 信号上相应的边沿后，计数器的当前值被锁存到捕获/比较寄存器（TMRx_CCx）中。当捕获事件发生时，相应的 CxIF 标志（TMRx_STS 寄存器）被置'1'，如果使能了中断或者 DMA 操作，则将产生中断或者 DMA 操作。如果捕获事件发生时 CxIF 标志已经为高，那么重复捕获标志 CxOF（TMRx_STS 寄存器）被置'1'。写 CxIF=0 可清除 CxIF，或读取存储在 TMRx_CCx 寄存器中的捕获数据也可清除 CxIF。写 CxOF=0 可清除 CxOF。

以下例子说明如何在 TI1 输入的上升沿时捕获计数器的值到 TMRx_CC1 寄存器中，步骤如下：

- 选择有效输入端：TMRx_CC1必须连接到TI1输入，所以写入TMRx_CCM1寄存器中的C1SEL=01，只要C1SEL不为'00'，通道被配置为输入，并且TM1_CC1寄存器变为只读。
- 根据输入信号的特点，配置输入滤波器为所需的带宽（即输入为TIx时，输入滤波器控制位是 TMRx_CCMx寄存器中的ICxDF位）。假设输入信号在最多5个内部时钟周期的时间内抖动，我们须配置滤波器的带宽长于5个时钟周期。因此我们可以（以f_{CK_INT}频率）连续采样8次，以确认在TI1上一次真实的边沿变换，即在TMRx_CCM1寄存器中写入IC1DF=0011。
- 选择TI1通道的有效转换边沿，在TMRx_CCE寄存器中写入C1P=0（上升沿）。
- 配置输入预分频器。在本例中，我们希望捕获发生在每一个有效的电平转换时刻，因此预分频器被禁止（写TMRx_CCM1寄存器的IC1DIV=00）。
- 设置TMRx_CCE寄存器的C1EN=1，允许捕获计数器的值到捕获寄存器中。
- 如果需要，通过设置TMRx_DIE寄存器中的C1IE位允许相关中断请求。

当发生一个输入捕获时：

- 产生有效的电平转换时，计数器的值被传送到TMRx_CC1寄存器。
- C1IF标志被设置（中断标志）。当发生至少2个连续的捕获时，而C1IF未曾被清除，C1OF也被置'1'。
- 如设置了C1IE位，则会产生一个中断。
- 为了处理重复捕获，建议在读出重复捕获标志之前读取数据，这是为了避免丢失在读出重复捕获标志之后和读取数据之前可能产生的重复捕获信息。

注意：　设置 TMRx_EVEG 寄存器中相应的 CxG 位，可以通过软件产生输入捕获中断。

10.3.3.6 PWM输入模式

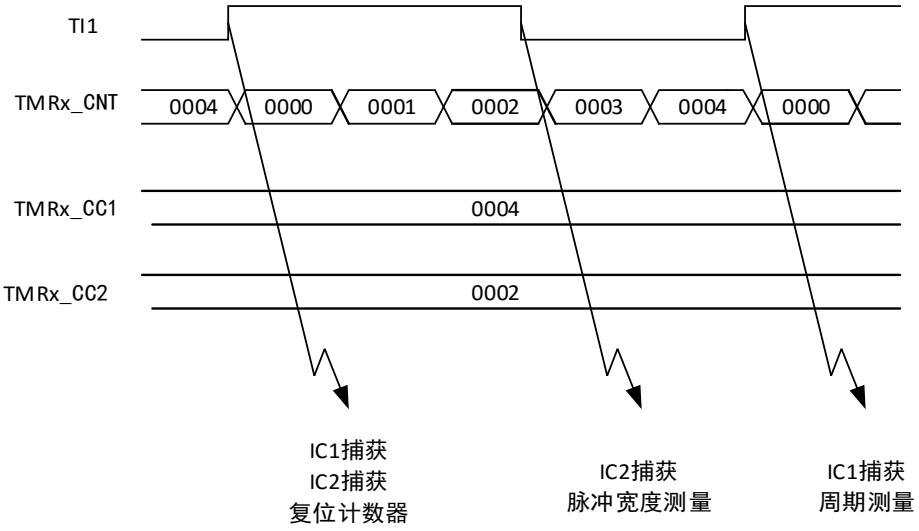
该模式是输入捕获模式的一个特例，除下列区别外，操作与输入捕获模式相同：

- 两个ICx信号被映射至同一个TIx输入。
- 这2个ICx信号为边沿有效，但是极性相反。
- 其中一个TIxFP信号被作为触发输入信号，而从模式控制器被配置成复位模式。

例如，你需要测量输入到 TI1 上的 PWM 信号的长度（TMRx_CC1 寄存器）和占空比（TMRx_CC2 寄存器），具体步骤如下（取决于 CK_INT 的频率和预分频器的值）。

- 选择TMRx_CC1的有效输入：置TMRx_CCM1寄存器的C1SEL=01（选择TI1）。
- 选择TI1FP1的有效极性（用来捕获数据到TMRx_CC1中和清除计数器）：置C1P=0（上升沿有效）。
- 选择TMRx_CC2的有效输入：置TMRx_CCM1寄存器的C2SEL=10（选择TI1）。
- 选择TI1FP2的有效极性（捕获数据到TMRx_CC2）：置C2P=1（下降沿有效）。
- 选择有效的触发输入信号：置TMRx_SMC寄存器中的TRGSEL=101（选择TI1FP1）。
- 配置从模式控制器为复位模式：置TMRx_SMC中的SMSEL=100。
- 使能捕获：置TMRx_CCE寄存器中C1EN=1且C2EN=1。

图 10-73 PWM输入模式时序



由于只有 TI1FP1 和 TI2FP2 连到了从模式控制器，所以 PWM 输入模式只能使用 TMRx_CH1/TMRx_CH2 信号。

10.3.3.7 强置输出模式

在输出模式（TMRx_CCMx 寄存器中 CxSEL=00）下，输出比较信号（OCxREF 和相应的 OCx）能够直接由软件强置为有效或无效状态，而不依赖于输出比较寄存器和计数器间的比较结果。

置 TMRx_CCMx 寄存器中相应的 OCxMODE=101，即可强置输出比较信号（OCxREF/OCx）为有效状态。这样 OCxREF 被强置为高电平（OCxREF 始终为高电平有效），同时 OCx 得到 CxP 极性位相反的值。

例如：CxP=0（OCx 高电平有效），则 OCx 被强置为高电平。

置 TMRx_CCMx 寄存器中的 OCxMODE=100，可强置 OCxREF 信号为低。

该模式下，在 TMRx_CCx 影子寄存器和计数器之间的比较仍然在进行，相应的标志也会被修改。因此仍然会产生相应的中断和 DMA 请求。这将会在下面的输出比较模式一节中介绍。

10.3.3.8 输出比较模式

此项功能是用来控制一个输出波形，或者指示一段给定的的时间已经到时。

当计数器与捕获/比较寄存器的内容相同时，输出比较功能做如下操作：

- 将输出比较模式（TMRx_CCMx寄存器中的OCxMODE位）和输出极性（TMRx_CCE寄存器中的CxP位）定义的值输出到对应的引脚上。在比较匹配时，输出引脚可以保持它的电平（OCxMODE=000）、被设置成有效电平（OCxMODE=001）、被设置成无效电平（OCxMODE=010）或进行翻转（OCxMODE=011）。
- 设置中断状态寄存器中的标志位（TMRx_STS寄存器中的CxIF位）。
- 若设置了相应的中断屏蔽（TMRx_DIE寄存器中的CxIE位），则产生一个中断。

TMRx_CCMx 中的 OCxPEN 位选择 TMRx_CCx 寄存器是否需要使用预装载寄存器。

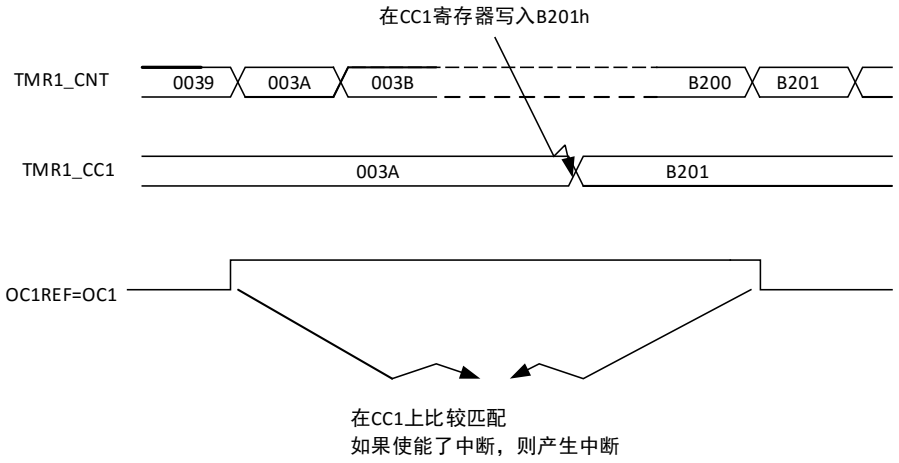
在输出比较模式下，更新事件 UEV 对 OCxREF 和 OCx 输出没有影响。同步的精度可以达到计数器的一个计数周期。输出比较模式（在单脉冲模式下）也能用来输出一个单脉冲。

输出比较模式的配置步骤：

1. 选择计数器时钟（内部，外部，预分频器）。
2. 将相应的数据写入 TMRx_AR 和 TMRx_CCx 寄存器中。
3. 如果要产生一个中断请求和/或一个 DMA 请求，设置 CxIE 位和/或 CxDE 位。
4. 选择输出模式，例如当计数器 CNT 与 CCx 匹配时翻转 OCx 的输出引脚，CCx 预装载未用，开启 OCx 输出且高电平有效，则必须设置 OCxMODE=’011’、OCxPEN=’0’、CxP=’0’和 CxEN=’1’。
5. 设置 TMRx_CTRL1 寄存器的 CNTEN 位启动计数器

TMRx_CCx 寄存器能够在任何时候通过软件进行更新以控制输出波形，条件是未使用预装载寄存器（OCxPEN=’0’，否则 TMRx_CCx 影子寄存器只能在发生下一次更新事件时被更新）。下图给出了一个例子。

图 10-74 输出比较模式，翻转OC1



10.3.3.9 PWM模式

脉冲宽度调制模式可以产生一个由 TMRx_AR 寄存器确定频率、由 TMRx_CCx 寄存器确定占空比的信号。

在 TMRx_CCMx 寄存器中的 OCxMODE 位写入’110’（PWM 模式 1）或’111’（PWM 模式 2），能够独立地设置每个 OCx 输出通道产生一路 PWM。必须设置 TMRx_CCMx 寄存器 OCxPEN 位以使能相应的预装载寄存器，最后还要设置 TMRx_CTRL1 寄存器的 ARPEN 位，（在向上计数或中心对称模式中）使能自动重装载的预装载寄存器。

仅当发生一个更新事件的时候，预装载寄存器才能被传送到影子寄存器，因此在计数器开始计数之前，必须通过设置 TMRx_EVEG 寄存器中的 UEVG 位来初始化所有的寄存器。OCx 的极性可以通过软件在 TMRx_CCE 寄存器中的 CxP 位设置，它可以设置为高电平有效或低电平有效。TMRx_CCE 寄存器中的 CxEN 位控制 OCx 输出使能。详见 TMRx_CCE 寄存器的描述。

在 PWM 模式（模式 1 或模式 2）下，TMRx_CNT 和 TMRx_CCx 始终在进行比较，以确定是否符合 TMRx_CCx ≤ TMRx_CNT 或者 TMRx_CNT ≤ TMRx_CCx。

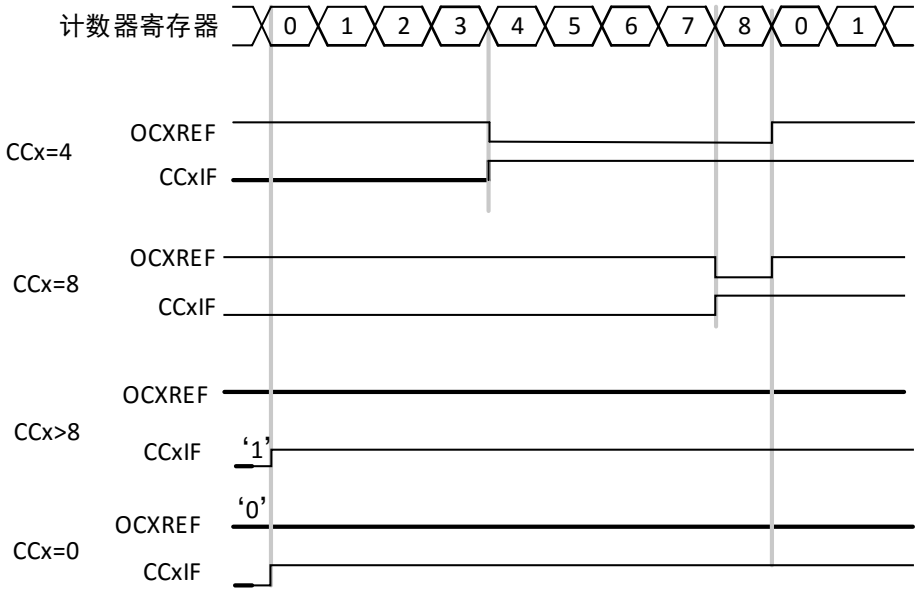
定时器在边沿对齐模式仅当计数器向上计数时能够产生 PWM。

PWM 边沿对齐模式

下面是一个 PWM 模式 1 的例子。当 TMRx_CNT<TMRx_CCx 时 PWM 信号参考 OCxREF 为高，否则为低。如果 TMRx_CCx 中的比较值大于自动重装载值（TMRx_AR），则 OCxREF 保持为'1'。

如果比较值为 0，则 OCxREF 保持为'0'。下图为 TMRx_AR=8 时边沿对齐的 PWM 波形实例。

图 10-75 边沿对齐的PWM波形（AR=8）



10.3.3.10 单脉冲模式

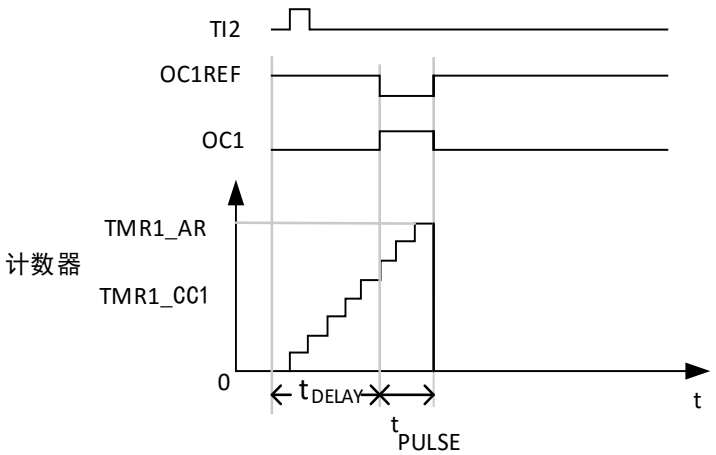
单脉冲模式（OPM）是前述众多模式的一个特例。这种模式允许计数器响应一个激励，并在一个程序可控的延时之后，产生一个脉宽可程序控制的脉冲。

可以通过从模式控制器启动计数器，在输出比较模式或者 PWM 模式下产生波形。设置 TMRx_CTRL1 寄存器中的 OPMODE 位将选择单脉冲模式，这样可以让计数器自动地在产生下一个更新事件 UEV 时停止。

仅当比较值与计数器的初始值不同时，才能产生一个脉冲。启动之前（当定时器正在等待触发），必须如下配置：

CNT < CCx ≤ AR（特别地，0 < CCx）。

图 10-76 单脉冲模式的例子



例如，你需要在从 TI2 输入脚上检测到一个上升沿开始，延迟 t_{DELAY} 之后，在 OC1 上产生一个长度为 t_{PULSE} 的正脉冲。

假定 TI2FP2 作为触发 1：

- 置TMRx_CCM1寄存器中的C2SEL='01'，把TI2FP2映像到TI2。

- 置TMRx_CCE寄存器中的C2P=’0’，使TI2FP2能够检测上升沿。
- 置TMRx_SMC寄存器中的TRGSEL=’110’，TI2FP2作为从模式控制器的触发（TRGI）。
- 置TMRx_SMC寄存器中的SMSEL=’110’（触发模式），TI2FP2被用来启动计数器。

OPM 波形由写入比较寄存器的数值决定（要考虑时钟频率和计数器预分频器）

- t_{DELAY}由写入TMRx_CC1寄存器中的值定义。
- t_{PULSE}由自动装载值和比较值之间的差值定义（TMRx_AR - TMRx_CC1）。
- 假定当发生比较匹配时要产生从’0’到’1’的波形，当计数器到达预装载值时要产生一个从’1’到’0’的波形；首先要置TMRx_CCM1寄存器的OC1MODE=’111’，进入PWM模式2；根据需要有选择地使能预装载寄存器：置TMRx_CCM1中的OC1PEN=’1’和TMRx_CTRL1寄存器中的ARPEN；然后在TMRx_CC1寄存器中填写比较值，在TMRx_AR寄存器中填写自动装载值，修改UEVG位来产生一个更新事件，然后等待在TI2上的一个外部触发事件。本例中， C1P=’0’。

在这个例子中，因为只需一个脉冲，所以必须设置 TMRx_CTRL1 寄存器中的 OPMODE=’1’，在下一个更新事件（当计数器从自动装载值翻转到 0）时停止计数。

当 OPMODE=‘0’的时候，退出单脉冲模式。

特殊情况：OCx 快速使能：

在单脉冲模式下，在 TIx 输入脚的边沿检测逻辑设置 CNTEN 位以启动计数器。然后计数器和比较值间的比较操作产生了输出的转换。但是这些操作需要一定的时钟周期，因此它限制了可得到的最小延时 t_{DELAY}。

如果要以最小延时输出波形，可以设置 TMRx_CCMx 寄存器中的 OCxFEN 位；此时 OCxREF（和 OCx）被强制响应激励而不再依赖比较的结果，输出的波形与比较匹配时的波形一样。OCxFEN 只在通道配置为 PWM1 和 PWM2 模式时起作用。

10.3.3.11 定时器和外部触发的同步

TMRx 定时器能够在多种模式下和一个外部的触发同步：复位模式、门控模式和触发模式。

从模式：复位模式

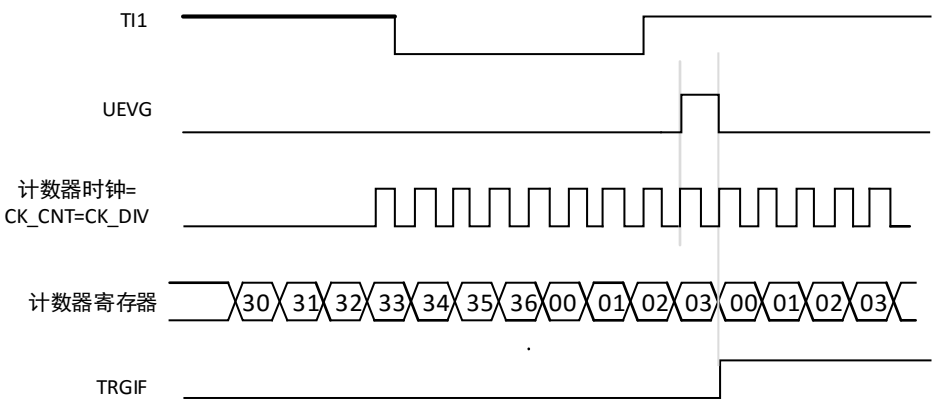
在发生一个触发输入事件时，计数器和它的预分频器能够重新被初始化；同时，如果 TMRx_CTRL1 寄存器的 UEVRS 位为低，还会产生一个更新事件 UEV；然后所有的预装载寄存器（TMRx_AR，TMRx_CCx）都会被更新。

在下面的例子中，TI1 输入端的上升沿导致向上计数器被清零：

- 配置通道1以检测TI1的上升沿。配置输入滤波器的带宽（在本例中，不需要任何滤波器，因此保持IC1DF=0000）。触发操作中不使用捕获预分频器，所以不需要配置它。C1SEL位只选择输入捕获源，即TMRx_CCM1寄存器中C1SEL=01。置TMRx_CCE寄存器中C1P=0以确定极性（只检测上升沿）。
- 置TMRx_SMC寄存器中SMSEL=100，配置定时器为复位模式；置TMRx_SMC寄存器中TRGSEL =101，选择TI1作为输入源。
- 置TMRx_CTRL1寄存器中CNTEN=1，启动计数器。

计数器开始依据内部时钟计数，然后正常运转直到 TI1 出现一个上升沿；此时，计数器被清零然后从 0 重新开始计数。同时，触发标志（TMRx_STS 寄存器中的 TRGIF 位）被设置，根据 TMRx_DIE 寄存器中 TRGIE（中断使能）位的设置，产生一个中断请求。下图显示当自动重装载寄存器 TMRx_AR=0x36 时的动作。在 TI1 上升沿和计数器的实际复位之间的延时，取决于 TI1 输入端的重同步电路。

图 10-77 复位模式下的控制电路



从模式：门控模式

按照选中的输入端电平使能计数。

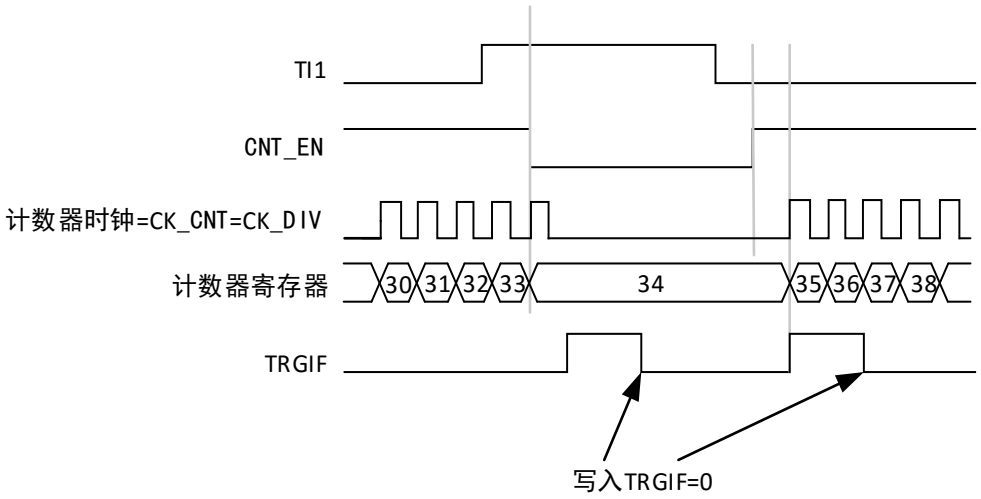
在如下的例子中，计数器只在 TI1 为低时向上计数：

- 配置通道1以检测TI1上的低电平。配置输入滤波器带宽（本例中，不需要滤波，所以保持IC1DF=0000）。触发操作中不使用捕获预分频器，所以不需要配置。C1SEL位用于选择输入捕获源，置TMRx_CCM1寄存器中C1SEL=01。置TMRx_CCE寄存器中C1P=1以确定极性（只检测低电平）。
- 置TMRx_SMC寄存器中SMSEL=101，配置定时器为门控模式；置TMRx_SMC寄存器中TRGSEL=101，选择 TI1作为输入源。
- 置TMRx_CTRL1寄存器中CNTEN=1，启动计数器。在门控模式下，如果CNTEN=0，则计数器不能启动，不论触发输入电平如何。

只要 TI1 为低，计数器开始依据内部时钟计数，在 TI1 变高时停止计数。当计数器开始或停止时都设置 TMRx_STS 中的 TRGIF 标置。

TI1 上升沿和计数器实际停止之间的延时，取决于 TI1 输入端的重同步电路。

图 10-78 门控模式下的控制电路



从模式：触发模式

输入端上选中的事件使能计数器。

在下面的例子中，计数器在 TI2 输入的上升沿开始向上计数：

- 配置通道2检测TI2的上升沿。配置输入滤波器带宽（本例中，不需要任何滤波器，保持IC2DF=0000）。触发操作中不使用捕获预分频器，不需要配置。C2SEL位只用于

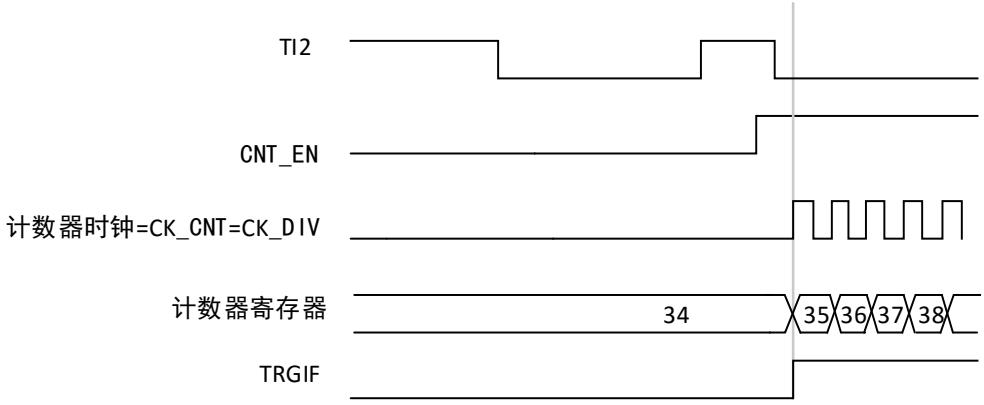
选择输入捕获源，置TMRx_CCM1寄存器中C2SEL=01。置TMRx_CCE寄存器中C2P=1以确定极性（只检测低电平）。

- 置TMRx_SMC寄存器中SMSEL=110，配置定时器为触发模式；置TMRx_SMC寄存器中TRGSEL=110，选择TI2作为输入源。

当 TI2 出现一个上升沿时，计数器开始在内部时钟驱动下计数，同时设置 TRGIF 标志。

TI2 上升沿和计数器启动计数之间的延时，取决于 TI2 输入端的重同步电路。

图 10-79 触发器模式下的控制电路



10.3.3.12 定时器同步

所有 TMRx 定时器在内部相连，用于定时器同步或链接。详细内容请参考 10.2.3.15 节。

注意：在接受主定时器的触发信号前，必须先使能从定时器的时钟，当接收来自主定时器触发时，不要在运行时改变它。

10.3.3.13 调试模式

当微控制器进入调试模式（Cortex™-M4F 核心停止），根据 DBG 模块中 DBG_TMRx_STOP 的设置，TMRx 计数器或者继续正常操作，或者停止。详见随后第 22.2.2 节：支持定时器、看门狗、bxCAN 和 I2C 的调试。

10.3.4 TMR9和TMR12寄存器描述

可以用半字（16 位）或字（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 10-6 TMRx – 寄存器图和复位值

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	TMRx_CTRL1	保留																						CLKDIV[1：0]		ARPEN	保留			OPMODE	UEVRS	UEVDIS	CNTEN
	0x0000																							0	0	0				0	0	0	0
0x08	TMRx_SMC	保留																									TRGSEL[2：0]			保留	SMSEL[2：0]		
	0x0000																										0	0	0		0	0	0
0x0C	TMRx_DIE	保留																									TRGIE	保留			C2IE	C1IE	UEVIE
	0x0000																										0				0	0	0

| | | |

位 15：10	保留，始终读为 0。
位 9：8	**CLKDIV[1：0]：** 时钟分频因子 （Clock division） 定义在定时器时钟（CK_INT）频率与数字滤波器（ETR，TIx）使用的采样频率之间的分频比例。 00：t_{DTS} =t_{CK_INT} 01：t_{DTS} = 2 xt_{CK_INT} 10：t_{DTS} = 4 x t_{CK_INT} 11：保留
位 7	**ARPEN：**自动重装载预装载允许位 （Auto-reload preload enable） 0：TMRx_AR 寄存器没有缓冲； 1：TMRx_AR 寄存器被装入缓冲器。
位 6：4	保留，始终读为 0。
位 3	**OPMODE：**单脉冲模式 （One pulse mode） 0：在发生更新事件时，计数器不停止； 1：在发生下一次更新事件（清除 CNTEN 位）时，计数器停止。
位 2	**UEVRS：**更新请求源 （Update request source） 软件通过该位选择 UEV 事件的源 0：如果使能了更新中断，则下述任一事件产生更新中断： – 计数器溢出 – 设置 UEVG 位 1：如果使能了更新中断请求，则只有计数器溢出才产生更新中断。
位 1	**UEVDIS：**禁止更新 （Update disable） 软件通过该位允许/禁止 UEV 事件的产生 0：允许 UEV。更新（UEV）事件由下述任一事件产生： – 计数器溢出 – 设置 UEVG 位 1：禁止 UEV。不产生更新事件，影子寄存器（AR、DIV、CCx）保持它们的值。如果设置了 UEVG 位或从模式控制器发出了一个硬件复位，则计数器和预分频器被重新初始化。
位 0	**CNTEN：**使能计数器 0：禁止计数器； 1：使能计数器。 在单脉冲模式下，当发生更新事件时，CNTEN 被自动清除。

10.3.4.2 从模式控制寄存器（TMRx_SMC）

偏移地址：0x08

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留									TRGSEL[2：0]			保留	SMSEL[2：0]		
RES									RW	RW	RW	RES	RW	RW	RW

位 15：7	保留，始终读为 0。
位 6：4	**TRGSEL[2：0]：**触发选择 （Trigger selection） 这 3 位选择用于同步计数器的触发输入。 000：内部触发 0（ITR0） 100：TI1 的边沿检测器（TI1F_ED） 001：内部触发 1（ITR1） 101：滤波后的定时器输入 1（TI1FP1） 010：内部触发 2（ITR2） 110：滤波后的定时器输入 2（TI2FP2） 011：内部触发 3（ITR3） 111：保留 关于每个定时器中 ITRx 的细节，参见表 10-7。 *注：这些位只能在未用到（如 SMSEL=000）时被改变，以避免在改变时产生错误的边沿检测。*

位 3	保留，始终读为 0。
位 2：0	**SMSEL[2：0]：**从模式选择 （Slave mode selection） 当选择了外部信号，触发信号（TRGI）的有效边沿与选中的外部输入极性相关（见输入控制寄存器和控制寄存器的说明） 000：关闭从模式 – 如果 CNTEN=1，则预分频器直接由内部时钟驱动。 001：保留。 010：保留。 011：保留。 100：复位模式 – 选中的触发输入（TRGI）的上升沿重新初始化计数器，并且产生一个更新寄存器的信号。 101：门控模式 – 当触发输入（TRGI）为高时，计数器的时钟开启。一旦触发输入变为低，则计数器停止（但不复位）。计数器的启动和停止都是受控的。 110：触发模式 – 计数器在触发输入 TRGI 的上升沿启动（但不复位），只有计数器的启动是受控的。 111：外部时钟模式 1 – 选中的触发输入（TRGI）的上升沿驱动计数器。 *注 1： 如果 TI1F_EN 被选为触发输入（TRGSEL=100）时，不要使用门控模式。这是因为，TI1F_ED 在每次 TI1F 变化时输出一个脉冲，然而门控模式是要检查触发输入的电平。* *注 2： 在接收主定时器事件之前必须先使能从定时器，而且在接受主定时器触发信号期间不可以改变从定时器的时钟。*

表 10-7 TMRx内部触发连接（1）

从定时器	ITR0 （TRGSEL = 000）	ITR1 （TRGSEL = 001）	ITR2 （TRGSEL = 010）	ITR3 （TRGSEL = 011）
TMR9	TMR2_TRGO	TMR3_TRGO	TMR10_OC	TMR11_OC
TMR12	TMR4_TRGO	TMR5_TRGO	TMR13_OC	TMR14_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 ITRx 也不存在。

10.3.4.3 DMA/中断使能寄存器（TMRx_DIE）

偏移地址：0x0C

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留									TRGIE	保留			C2IE	C1IE	UEVIE
RES									RW	RES			RW	RW	RW

位 15：7	保留，始终读为 0。
位 6	**TRGIE：**触发中断使能 （Trigger interrupt enable） 0：禁止触发中断； 1：使能触发中断。
位 5：3	保留，始终读为 0。
位 2	**C2IE：**允许捕获/比较 2 中断 （Capture/Compare 2 interrupt enable） 0：禁止捕获/比较 2 中断； 1：允许捕获/比较 2 中断。
位 1	**C1IE：**允许捕获/比较 1 中断 （Capture/Compare 1 interrupt enable） 0：禁止捕获/比较 1 中断； 1：允许捕获/比较 1 中断。
位 0	**UEVIE：**允许更新中断 （Update interrupt enable） 0：禁止更新中断； 1：允许更新中断。

10.3.4.4 状态寄存器（TMRx_STS）

偏移地址：0x10

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留					C2OF	C1OF	保留		TRGIF	保留			C2IF	C1IF	UEVIF
RES					RC_W0	RC_W0	RES		RC_W0	RES			RC_W0	RC_W0	RC_W0

位 15：11	保留，始终读为 0。
位 10	**C2OF**：捕获/比较 2 重复捕获标记 （Capture/Compare 2 overcapture flag） 参见 C1OF 描述。
位 9	**C1OF**：捕获/比较 1 重复捕获标记 （Capture/Compare 1 overcapture flag） 仅当相应的通道被配置为输入捕获时，该标记可由硬件置’1’。写’0’可清除该位。 0：无重复捕获产生； 1：当计数器的值被捕获到 TMRx_CC1 寄存器时，C1IF 的状态已经为’1’。
位 8：7	保留，始终读为 0。
位 6	**TRGIF**：触发器中断标记 （Trigger interrupt flag） 当发生触发事件（当从模式控制器处于除门控模式外的其它模式时，在 TRGI 输入端检测到有效 边沿，或门控模式下的任一边沿）时由硬件对该位置’1’。它由软件清’0’。 0：无触发器事件产生； 1：触发器中断等待响应。
位 5：3	保留，始终读为 0。
位 2	**C2IF**：捕获/比较 2 中断标记 （Capture/Compare 2 interrupt flag） 参考 C1IF 描述。
位 1	**C1IF**：捕获/比较 1 中断标记 （Capture/Compare 1 interrupt flag） **如果通道 CC1 配置为输出模式：** 当计数器值与比较值匹配时该位由硬件置’1’，但在中心对称模式下除外（参考 TMRx_CTRL1 寄存器的 CMSEL 位）。它由软件清’0’。 当 TMRx_CC1 的值比 TMRx_AR 的值大的时候，CC1F 位在计数器溢出后置高。 0：无匹配发生； 1：TMRx_CNT 的值与 TMRx_CC1 的值匹配。 **如果通道 CC1 配置为输入模式：** 当捕获事件发生时该位由硬件置’1’，它由软件清’0’或通过读 TMRx_CC1 清’0’。 0：无输入捕获产生； 1：计数器值已被捕获（拷贝）至 TMRx_CC1（在 IC1 上检测到与所选极性相同的边沿）。
位 0	**UEVIF**：更新中断标记 （Update interrupt flag） 当产生更新事件时该位由硬件置’1’。它由软件清’0’。 0：无更新事件产生； 1：更新中断等待响应。当寄存器被更新时该位由硬件置’1’： – 若 TMRx_CTRL1 寄存器的 UEVDIS=0、UEVRS=0，当 TMRx_EVEG 寄存器的 UEVG=1 时产生更新事件（软件对计数器 CNT 重新初始化）； – 若 TMRx_CTRL1 寄存器的 UEVDIS=0、UEVRS=0，当计数器 CNT 被触发事件重初始化时产生更新事件。 当 UEVDIS=0，计数器计数溢出时设置该位。

10.3.4.5 事件产生寄存器（TMRx_EVEG）

偏移地址：0x14

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留									TRGG	保留			C2G	C1G	UEVG
RES									W	RES			W	W	W

位 15：7	保留，始终读为 0。
位 6	**TRGG**：产生触发事件 （Trigger generation） 该位由软件置'1'，用于产生一个触发事件，由硬件自动清'0'。 0：无动作； 1：TMRx_STS 寄存器的 TRGIF=1，若开启对应的中断，则产生相应的中断。
位 5：3	保留，始终读为 0。
位 2	**C2G**：产生捕获/比较 2 事件 （Capture/compare 2 generation） 参考 C1G 描述。
位 1	**C1G**：产生捕获/比较 1 事件 （Capture/compare 1 generation） 该位由软件置'1'，用于产生一个捕获/比较事件，由硬件自动清'0'。 0：无动作； 1：在通道 CC1 上产生一个捕获/比较事件： **若通道 CC1 配置为输出：** 设置 C1IF=1，若开启对应的中断，则产生相应的中断。 **若通道 CC1 配置为输入：** 当前的计数器值捕获至 TMRx_CC1 寄存器；设置 C1IF=1，若开启对应的中断，则产生相应的中断。若 C1IF 已经为 1，则设置 C1OF=1。
位 0	**UEVG**：产生更新事件 （Update generation） 该位由软件置'1'，由硬件自动清'0'。 0：无动作； 1：重新初始化计数器，并产生一个更新事件。注意预分频器的计数器也被清'0'（但是预分频系数不变）。计数器被清'0'。

10.3.4.6 捕获/比较模式寄存器1（TMRx_CCM1）

偏移地址：0x18

复位值：0x0000

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 CxSEL 定义。该寄存器其它位的作用在输入和输出模式下不同。OCxx 描述了通道在输出模式下的功能，ICxx 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：

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留	OC2MODE[2：0]			OC2PEN	OC2FEN	C2SEL[1：0]		保留	OC1MODE[2：0]			OC1PEN	OC1FEN	C1SEL[1：0]	
RES	RW	RW	RW	RW	RW	RW	RW	RES	RW	RW	RW	RW	RW	RW	RW

位 15	保留。
位 14：12	**OC2MODE[2：0]**：输出比较 2 模式 （Output compare 2 mode）
位 11	**OC2PEN**：输出比较 2 预装载使能 （Output compare 2 preload enable）
位 10	**OC2FEN**：输出比较 2 快速使能 （Output compare 2 fast enable）

位 9：8	**C2SEL[1：0]：**捕获/比较 2 选择 （Capture/Compare 2 selection） 该位定义通道的方向（输入/输出），及输入脚的选择： 00：CC2 通道被配置为输出； 01：CC2 通道被配置为输入，IC2 映射在 TI2 上； 10：CC2 通道被配置为输入，IC2 映射在 TI1 上； 11：CC2 通道被配置为输入，IC2 映射在 TRC 上。此模式仅工作在内部触发器输入被选中时（由 TMRx_SMC 寄存器的 TRGSEL 位选择）。 *注：C2SEL 仅在通道关闭时（TMRx_CCE 寄存器的 C2EN='0'）才是可写的。*
位 7	保留。
位 6：4	**OC1MODE[2：0]：**输出比较 1 模式 （Output compare 1 enable） 该 3 位定义了输出参考信号 OC1REF 的动作，而 OC1REF 决定了 OC1 的值。OC1REF 是高电平 有效，而 OC1 的有效电平取决于 C1P 位。 000：冻结。输出比较寄存器 TMRx_CC1 与计数器 TMRx_CNT 间的比较对 OC1REF 不起作用； 001：匹配时设置通道 1 为有效电平。当计数器 TMRx_CNT 的值与捕获 / 比较寄存器 1（TMRx_CC1）相同时，强制 OC1REF 为高。 010：匹配时设置通道 1 为无效电平。当计数器 TMRx_CNT 的值与捕获 / 比较寄存器 1（TMRx_CC1）相同时，强制 OC1REF 为低。 011：翻转。当 TMRx_CC1=TMRx_CNT 时，翻转 OC1REF 的电平。 100：强制为无效电平。强制 OC1REF 为低。 101：强制为有效电平。强制 OC1REF 为高。 110：PWM 模式 1— 在向上计数时，一旦 TMRx_CNT<TMRx_CC1 时通道 1 为有效电平，否则为无效电平。 111：PWM 模式 2— 在向上计数时，一旦 TMRx_CNT<TMRx_CC1 时通道 1 为无效电平，否则为有效电平。 *注 1：在 PWM 模式 1 或 PWM 模式 2 中，只有当比较结果改变了或在输出比较模式中从冻结模式切换到 PWM 模式时，OC1REF 电平才改变。*
位 3	**OC1PEN：**输出比较 1 预装载使能 （Output compare 1 preload enable） 0：禁止 TMRx_CC1 寄存器的预装载功能，可随时写入 TMRx_CC1 寄存器，并且新写入的数值立即起作用。 1：开启 TMRx_CC1 寄存器的预装载功能，读写操作仅对预装载寄存器操作，TMRx_CC1 的 预装载值在更新事件到来时被传送至当前寄存器中。 *注：仅在单脉冲模式下（TMRx_CTRL1 寄存器的 OPMODE='1'），可以在未确认预装载寄存器情况下使用 PWM 模式，否则其动作不确定。*
位 2	**OC1FEN：**输出比较 1 快速使能 （Output compare 1 fast enable） 该位用于加快 CC 输出对触发器输入事件的响应。 0：根据计数器与 CC1 的值，CC1 正常操作，即使触发器是打开的。当触发器的输入出现一个有效沿时，激活 CC1 输出的最小延时为 5 个时钟周期。 1：输入到触发器的有效沿的作用就像发生了一次比较匹配。因此，OC 被设置为比较电平而与比较结果无关。采样触发器的有效沿和 CC1 输出间的延时被缩短为 3 个时钟周期。 该位只在通道被配置成 PWM1 或 PWM2 模式时起作用。
位 1：0	**C1SEL[1：0]：**捕获/比较 1 选择 （Capture/Compare 1 selection） 这 2 位定义通道的方向（输入/输出），及输入脚的选择： 00：CC1 通道被配置为输出； 01：CC1 通道被配置为输入，IC1 映射在 TI1 上； 10：CC1 通道被配置为输入，IC1 映射在 TI2 上； 11：CC1 通道被配置为输入，IC1 映射在 TRC 上。此模式仅工作在内部触发器输入被选中时（由 TMRx_SMC 寄存器的 TRGSEL 位选择）。 *注：C1SEL 仅在通道关闭时（TMRx_CCE 寄存器的 C1EN='0'）才是可写的。*

输入捕获模式：

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IC2DF[3：0]				IC2DIV[1：0]		C2SEL[1：0]		IC1DF[3：0]				IC1DIV[1：0]		C1SEL[1：0]	
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

位 15：12	**IC2DF[3：0]：**输入捕获 2 滤波器 （Input capture 2 filter）
位 11：10	**IC2DIV[1：0]：**输入/捕获 2 预分频器 （input capture 2 prescaler）
位 9：8	**C2SEL[1：0]：**捕获/比较 2 选择 （Capture/compare 2 selection） 这 2 位定义通道的方向（输入/输出），及输入脚的选择： 00：CC2 通道被配置为输出； 01：CC2 通道被配置为输入，IC2 映射在 TI2 上； 10：CC2 通道被配置为输入，IC2 映射在 TI1 上； 11：CC2 通道被配置为输入，IC2 映射在 TRC 上。此模式仅工作在内部触发器输入被选中时（由 TMRx_SMC 寄存器的 TRGSEL 位选择）。 *注：C2SEL 仅在通道关闭时（TMRx_CCE 寄存器的 C2EN='0'）才是可写的。*
位 7：4	**IC1DF[3：0]：**输入捕获 1 滤波器 （Input capture 1 filter） 这几位定义了 TI1 输入的采样频率及数字滤波器长度。数字滤波器由一个事件计数器组成，它记录到 N 个事件后会产生一个输出的跳变： 0000：无滤波器，以f_{DTS}采样 0001：采样频率 $f_{SAMPLING}=f_{CK_INT}$，N=2 0010：采样频率 $f_{SAMPLING}=f_{CK_INT}$，N=4 0011：采样频率 $f_{SAMPLING}=f_{CK_INT}$，N=8 0100：采样频率 $f_{SAMPLING}=f_{DTS}/2$，N=6 0101：采样频率 $f_{SAMPLING}=f_{DTS}/2$，N=8 0110：采样频率 $f_{SAMPLING}=f_{DTS}/4$，N=6 0111：采样频率 $f_{SAMPLING}=f_{DTS}/4$，N=8 1000：采样频率$f_{SAMPLING}=f_{DTS}/8$，N=6 1001：采样频率$f_{SAMPLING}=f_{DTS}/8$，N=8 1010：采样频率$f_{SAMPLING}=f_{DTS}/16$，N=5 1011：采样频率$f_{SAMPLING}=f_{DTS}/16$，N=6 1100：采样频率$f_{SAMPLING}=f_{DTS}/16$，N=8 1101：采样频率$f_{SAMPLING}=f_{DTS}/32$，N=5 1110：采样频率$f_{SAMPLING}=f_{DTS}/32$，N=6 1111：采样频率$f_{SAMPLING}=f_{DTS}/32$，N=8
位 3：2	**IC1DIV[1：0]：**输入/捕获 1 预分频器 （Input capture 1 prescaler） 这 2 位定义了 CC1 输入（IC1）的预分频系数。 一旦 C1EN='0'（TMRx_CCE 寄存器中），则预分频器复位。 00：无预分频器，捕获输入口上检测到的每一个边沿都触发一次捕获； 01：每 2 个事件触发一次捕获； 10：每 4 个事件触发一次捕获； 11：每 8 个事件触发一次捕获。
位 1：0	**C1SEL[1：0]：**捕获/比较 1 选择 （Capture/Compare 1 selection） 这 2 位定义通道的方向（输入/输出），及输入脚的选择： 00：CC1 通道被配置为输出； 01：CC1 通道被配置为输入，IC1 映射在 TI1 上； 10：CC1 通道被配置为输入，IC1 映射在 TI2 上； 11：CC1 通道被配置为输入，IC1 映射在 TRC 上。此模式仅工作在内部触发器输入被选中时（由 TMRx_SMC 寄存器的 TRGSEL 位选择）。 *注：C1SEL 仅在通道关闭时（TMRx_CCE 寄存器的 C1EN='0'）才是可写的。*

10.3.4.7 捕获/比较使能寄存器（TMRx_CCE）

偏移地址：0x20

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留								C2NP	保留	C2P	C2EN	C1NP	保留	C1P	C1EN
RES								RW	RES	RW	RW	RW	RES	RW	RW

位 15：8	保留，始终读为 0。
位 7	**C2NP**：输入/捕获 2 互补输出极性 （Capture/Compare 2 output polarity） 参考 C1NP 的描述。
位 6	保留，始终读为 0。
位 5	**C2P**：输入/捕获 2 输出极性 （Capture/Compare 2 output polarity） 参考 C1P 的描述。
位 4	**C2EN**：输入/捕获 2 输出使能 （Capture/Compare 2 output enable） 参考 C1EN 的描述。
位 3	**C1NP**：输入/捕获 1 互补输出极性 （Capture/Compare 1 complementary output polarity） CC1 通道配置为输出：C1NP 必须保持清除。 CC1 通道配置为输入：C1NP 结合 C1P 定义 TI1FP1/TI2FP1 极性（参考 C1P 描述）
位 2	保留，始终读为 0。
位 1	**C1P**：输入/捕获 1 输出极性 （Capture/Compare 1 output polarity） **CC1 通道配置为输出：** 0：OC1 高电平有效 1：OC1 低电平有效 **CC1 通道配置为输入：** CC1NP/CC1P 位选择 TI1FP1 和 TI2FP1 的极性信号作为触发或捕获信号。 00：不反相/上升沿。电路 TIxFP1 上升沿敏感（捕获，触发中的复位，外部时钟或触发模式），TIxFP1 不反相（触发中的门控模式）。 01：反相/下降沿。电路对 TIxFP1 下降沿敏感（捕获，触发中的复位，外部时钟或触发模式），TIxFP1 反相（触发中的门控模式）。 10：保留。 11：不反相/双沿。电路同时对 TIxFP1 上升沿和下降沿敏感（捕获，触发中的复位，外部时钟或触发模式），TIxFP1 是不反相（触发中门控模式）。
位 0	**C1EN**：输入/捕获 1 输出使能 （Capture/Compare 1 output enable） **CC1 通道配置为输出：** 0： 关闭－ OC1 禁止输出。 1： 开启－ OC1 信号输出到对应的输出引脚。 **CC1 通道配置为输入：** 该位决定了计数器的值是否能捕获入 TMRx_CC1 寄存器。 0：捕获禁止； 1：捕获使能。

表 10-8 标准OCx通道的输出控制位

CxEN 位	OCx 输出状态
0	禁止输出（OCx=0，OCx_EN=0）
1	OCx = OCxREF + 极性，OCx_EN=1

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

10.3.4.8 计数器（TMRx_CNT）

偏移地址：0x24

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT[15：0]															
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

位 15：0	**CNT[15：0]**：计数器的值 （Counter value）

10.3.4.9 预分频器（TMRx_DIV）

偏移地址：0x28

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DIV[15：0]															
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

位 15：0	**DIV[15：0]：**预分频器的值 （Prescaler value） 计数器的时钟频率 CK_CNT 等于 f_{CK_DIV}/（DIV[15：0]+1）。 DIV 包含了当更新事件产生时装入当前预分频器寄存器的值。

10.3.4.10 自动重装载寄存器（TMRx_AR）

偏移地址：0x2C

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AR[15：0]															
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

位 15：0	**AR[15：0]：** 自动重装载的值 （Auto reload value） AR 包含了将要传送至实际的自动重装载寄存器的数值。 详细参考 10.3.3.1 节：有关 AR 的更新和动作当自动重装载的值为空时，计数器不工作。

10.3.4.11 捕获/比较寄存器1（TMRx_CC1）

偏移地址：0x34

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CC1[15：0]															
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

位 15：0	**CC1[15：0]：** 捕获/比较 1 的值 （Capture/Compare 1 value） **若 CC1 通道配置为输出：** CC1 包含了装入当前捕获/比较 1 寄存器的值（预装载值）。 如果在 TMRx_CCM1 寄存器（OC1PEN 位）中未选择预装载特性，写入的数值会被立即传输至当前寄存器中。否则只有当更新事件发生时，此预装载值才传输至当前捕获/比较寄存器 1 中。 当前捕获/比较寄存器参与同计数器 TMRx_CNT 的比较，并在 OC1 端口上产生输出信号。 **若 CC1 通道配置为输入：** CC1 包含了由上一次输入捕获 1 事件（IC1）传输的计数器值。

10.3.4.12 捕获/比较寄存器2（TMRx_CC2）

偏移地址：0x38

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CC2[15：0]															
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

位 15：0	**CC2[15：0]：** 捕获/比较 2 的值 （Capture/Compare 2 value） **若 CC2 通道配置为输出：** CC2 包含了装入当前捕获/比较 2 寄存器的值（预装载值）。 如果在 TMRx_CCM2 寄存器（OC2PEN 位）中未选择预装载特性，写入的数值会被立即传输至当前寄存器中。否则只有当更新事件发生时，此预装载值才传输至当前捕获/比较寄存器 2 中。 当前捕获/比较寄存器参与同计数器 TMRx_CNT 的比较，并在 OC2 端口上产生输出信号。 **若 CC2 通道配置为输入：** CC2 包含了由上一次输入捕获 2 事件（IC2）传输的计数器值。

10.3.5 TMR10、TMR11、TMR13和TMR14寄存器描述

可以用半字（16 位）或字（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 10-9 TMRx – 寄存器图和复位值

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	TMRx_CTRL1	保留																						CLKDIV[1：0]		ARPEN	保留				UEVRS	UEVDIS	CNTEN
	复位值																							0	0	0					0	0	0
0x0C	TMRx_DIE	保留																														C1IE	UEVIE
	复位值																															0	0
0x10	TMRx_STS	保留																						C1OF	保留							C1IF	UEVIF
	复位值																							0								0	0
0x14	TMRx_EVEG	保留																														C1G	UEVG
	复位值																															0	0
0x18	TMRx_CCM1 输出比较模式	保留																									OC1MODE[2：0]			OC1PEN	OC1FEN	C1SEL[1：0]	
	复位值																										0	0	0	0	0	0	0
	TMRx_CCM1 输入捕获模式	保留																								IC1DF[3：0]				IC1DIV[1：0]		C1SEL[1：0]	

	复位值															0	0	0	0	0	0	0	0
0x20	TMRx_CCE	保留													C1NP	保留		C1P	C1EN
	复位值														0			0	0
0x24	TMRx_CNT	保留												CNT[15：0]					
	复位值													0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x28	TMRx_DIV	保留												DIV[15：0]					
	复位值													0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x2C	TMRx_AR	保留												AR[15：0]					
	复位值													0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x34	TMRx_CC1	保留												CC1[15：0]					
	复位值													0	0	0	0	0	0	0	0	0	0	0	0	0	0

10.3.5.1 控制寄存器1（TMRx_CTRL1）

偏移地址：0x00

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留						CLKDIV[1：0]		ARPEN	保留				UEVRS	UEVDIS	CNTEN
res						rw	rw	rw	res				rw	rw	rw

位 15：10	保留，始终读为 0。
位 9：8	**CLKDIV[1：0]：** 时钟分频因子 （Clock division） 定义在定时器时钟（CK_INT）频率与数字滤波器（ETR，TIx）使用的采样频率之间的分频比例。 00：t_{DTS} = t_{CK_INT} 01：t_{DTS} = 2 x t_{CK_INT} 10：t_{DTS} = 4 x t_{CK_INT} 11：保留
位 7	**ARPEN：** 自动重装载预装载允许位 （Auto-reload preload enable） 0：TMRx_AR 寄存器没有缓冲； 1：TMRx_AR 寄存器被装入缓冲器。
位 6：3	保留，始终读为 0。
位 2	**UEVRS：** 更新请求源 （Update request source） 软件通过该位选择 UEV 事件的源 0：如果使能了更新中断，则下述任一事件产生更新中断： – 计数器溢出 – 设置 UEVG 位 1：如果使能了更新中断，则只有计数器溢出才产生更新中断。
位 1	**UEVDIS：** 禁止更新 （Update disable） 软件通过该位允许/禁止 UEV 事件的产生 0：允许 UEV。更新（UEV）事件由下述任一事件产生： – 计数器溢出 – 设置 UEVG 位 1：禁止 UEV。不产生更新事件，影子寄存器（AR、DIV、CCx）保持它们的值。如果设置了 UEVG 位或从模式控制器发出了一个硬件复位，则计数器和预分频器被重新初始化。

位 0	**CNTEN**：使能计数器 0：禁止计数器； 1：使能计数器。 在单脉冲模式下，当发生更新事件时，CNTEN 被自动清除。

10.3.5.2 DMA/中断使能寄存器（TMRx_DIE）

偏移地址：0x0C

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留														C1IE	UEVIE
res														rw	rw

位 15：2	保留，始终读为 0。
位 1	**C1IE**：允许捕获/比较 1 中断 （Capture/Compare 1 interrupt enable） 0：禁止捕获/比较 1 中断； 1：允许捕获/比较 1 中断。
位 0	**UEVIE**：允许更新中断 （Update interrupt enable） 0：禁止更新中断； 1：允许更新中断。

10.3.5.3 状态寄存器（TMRx_STS）

偏移地址：0x10

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留						C1OF	保留							C1IF	UEVIF
res						rw	res							rw	rw

位 15：10	保留，始终读为 0。
位 9	**C1OF**：捕获/比较 1 重复捕获标记 （Capture/Compare 1 overcapture flag） 仅当相应的通道被配置为输入捕获时，该标记可由硬件置’1’。写’0’可清除该位。 0：无重复捕获产生； 1：当计数器的值被捕获到 TMRx_CC1 寄存器时，C1IF 的状态已经为’1’。
位 8：2	保留，始终读为 0。
位 1	**C1IF**：捕获/比较 1 中断标记 （Capture/Compare 1 interrupt flag） **如果通道 CC1 配置为输出模式：** 当计数器值与比较值匹配时该位由硬件置’1’，由软件清’0’。 0：无匹配发生； 1：TMRx_CNT 的值与 TMRx_CC1 的值匹配。当计数溢出时，C1F 只‘1’。 **如果通道 CC1 配置为输入模式：** 当捕获事件发生时该位由硬件置’1’，它由软件清’0’或通过读 TMRx_CC1 清’0’。 0：无输入捕获产生； 1：计数器值已被捕获（拷贝）至 TMRx_CC1（在 IC1 上检测到与所选极性相同的边沿）。

位 0	**UEVIF**：更新中断标记 （Update interrupt flag） 当产生更新事件时该位由硬件置'1'。它由软件清'0'。 0：无更新事件产生； 1：更新中断等待响应。当寄存器被更新时该位由硬件置'1'： – 若 TMRx_CTRL1 寄存器的 UEVDIS=0、UEVRS=0，当 TMRx_EVEG 寄存器的 UEVG=1 时产生更新事件（软件对计数器 CNT 重新初始化）； – 若 TMRx_CTRL1 寄存器的 UEVDIS=0，计数器溢出。

10.3.5.4 事件产生寄存器（TMRx_EVEG）

偏移地址：0x14

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留														C1G	UEVG
res														w	w

位 15：2	保留，始终读为 0。
位 1	**C1G**：产生捕获/比较 1 事件 （Capture/compare 1 generation） 该位由软件置'1'，用于产生一个捕获/比较事件，由硬件自动清'0'。 0：无动作； 1：在通道 CC1 上产生一个捕获/比较事件： **若通道 CC1 配置为输出：** 设置 C1IF=1，若开启对应的中断，则产生相应的中断。 **若通道 CC1 配置为输入：** 当前的计数器值捕获至 TMRx_CC1 寄存器；设置 C1IF=1，若开启对应的中断，则产生相应的中断。若 C1IF 已经为 1，则设置 C1OF=1。
位 0	**UEVG**：产生更新事件 （Update generation） 该位由软件置'1'，由硬件自动清'0'。 0：无动作； 1：重新初始化计数器，并产生一个更新事件。注意预分频器的计数器也被清'0'（但是预分频系数不变）。计数器被清 '0' 。

10.3.5.5 捕获/比较模式寄存器1（TMRx_CCM1）

偏移地址：0x18

复位值：0x0000

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 CxSEL 定义。该寄存器其它位的作用在输入和输出模式下不同。OCx 描述了通道在输出模式下的功能，ICx 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：

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留									OC1MODE[2：0]			OC1PEN	OC1FEN	C1SEL[1：0]	
res									rw	rw	rw	rw	rw	rw	rw

位 15：7	保留，始终读为 0。
位 6：4	**OC1MODE[2：0]：**输出比较 1 模式 （Output compare 1 enable） 该 3 位定义了输出参考信号 OC1REF 的动作，而 OC1REF 决定了 OC1 的值。OC1REF 是高电平有效，而 OC1 的有效电平取决于 C1P 位。 000：冻结。输出比较寄存器 TMRx_CC1 与计数器 TMRx_CNT 间的比较对 OC1REF 不起作用； 001：匹配时设置通道 1 为有效电平。当计数器 TMRx_CNT 的值与捕获 / 比较寄存器 1 （TMRx_CC1）相同时，强制 OC1REF 为高。 010：匹配时设置通道 1 为无效电平。当计数器 TMRx_CNT 的值与捕获 / 比较寄存器 1 （TMRx_CC1）相同时，强制 OC1REF 为低。 011：翻转。当 TMRx_CC1=TMRx_CNT 时，翻转 OC1REF 的电平。 100：强制为无效电平。强制 OC1REF 为低。 101：强制为有效电平。强制 OC1REF 为高。 110：PWM 模式 1－一旦 TMRx_CNT<TMRx_CC1 时通道 1 为有效电平，否则为无效电平； 111：PWM 模式 2－一旦 TMRx_CNT<TMRx_CC1 时通道 1 为无效电平，否则为有效电平； *注：在 PWM 模式 1 或 PWM 模式 2 中，只有当比较结果改变了或在输出比较模式中从冻结模式切换到 PWM 模式时，OC1REF 电平才改变。*
位 3	**OC1PEN：**输出比较 1 预装载使能 （Output compare 1 preload enable） 0：禁止 TMRx_CC1 寄存器的预装载功能，可随时写入 TMRx_CC1 寄存器，并且新写入的数值立即起作用。 1：开启 TMRx_CC1 寄存器的预装载功能，读写操作仅对预装载寄存器操作，TMRx_CC1 的预装载值在更新事件到来时被传送至当前寄存器中。 *注：仅在单脉冲模式下（TMRx_CTRL1 寄存器的 OPMODE='1'），可以在未确认预装载寄存器情况下使用 PWM 模式，否则其动作不确定。*
位 2	**OC1FEN：**输出比较 1 快速使能 （Output compare 1 fast enable） 该位用于加快 CC 输出对触发器输入事件的响应。 0：根据计数器与 CC1 的值，CC1 正常操作，即使触发器是打开的。当触发器的输入出现一个有效沿时，激活 CC1 输出的最小延时为 5 个时钟周期。 1：输入到触发器的有效沿的作用就像发生了一次比较匹配。因此，OC 被设置为比较电平而与比较结果无关。采样触发器的有效沿和 CC1 输出间的延时被缩短为 3 个时钟周期。 该位只在通道被配置成 PWM1 或 PWM2 模式时起作用。
位 1：0	**C1SEL[1：0]：**捕获/比较 1 选择 （Capture/Compare 1 selection） 这 2 位定义通道的方向（输入/输出），及输入脚的选择： 00：CC1 通道被配置为输出； 01：CC1 通道被配置为输入，IC1 映射在 TI1 上； 10：保留； 11：保留。 *注：C1SEL 仅在通道关闭时（TMRx_CCE 寄存器的 C1EN='0'）才是可写的。*

输入捕获模式：

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
res								IC1DF[3：0]				IC1DIV [1：0]		C1SEL [1：0]	
								rw	rw	rw	rw	rw	rw	rw	rw

位 15：8	保留，始终读为 0。
位 7：4	**IC1DF[3：0]：**输入捕获 1 滤波器 （Input capture 1 filter） 这几位定义了 TI1 输入的采样频率及数字滤波器长度。数字滤波器由一个事件计数器组成，它记录到 N 个事件后会产生一个输出的跳变： 0000：无滤波器，以f_{DTS}采样 0001：采样频率 $f_{SAMPLING}=f_{CK_INT}$，N=2 0010：采样频率 $f_{SAMPLING}=f_{CK_INT}$，N=4 0011：采样频率 $f_{SAMPLING}=f_{CK_INT}$，N=8 0100：采样频率 $f_{SAMPLING}=f_{DTS}/2$，N=6 0101：采样频率 $f_{SAMPLING}=f_{DTS}/2$，N=8 0110：采样频率 $f_{SAMPLING}=f_{DTS}/4$，N=6 0111：采样频率 $f_{SAMPLING}=f_{DTS}/4$，N=8 1000：采样频率$f_{SAMPLING}=f_{DTS}/8$，N=6 1001：采样频率$f_{SAMPLING}=f_{DTS}/8$，N=8 1010：采样频率$f_{SAMPLING}=f_{DTS}/16$，N=5 1011：采样频率$f_{SAMPLING}=f_{DTS}/16$，N=6 1100：采样频率$f_{SAMPLING}=f_{DTS}/16$，N=8 1101：采样频率$f_{SAMPLING}=f_{DTS}/32$，N=5 1110：采样频率$f_{SAMPLING}=f_{DTS}/32$，N=6 1111：采样频率$f_{SAMPLING}=f_{DTS}/32$，N=8
位 3：2	**IC1DIV[1：0]：**输入/捕获 1 预分频器 （Input capture 1 prescaler） 这 2 位定义了 CC1 输入（IC1）的预分频系数。 一旦 C1EN=’0’（TMRx_CCE 寄存器中），则预分频器复位。 00：无预分频器，捕获输入口上检测到的每一个边沿都触发一次捕获； 01：每 2 个事件触发一次捕获； 10：每 4 个事件触发一次捕获； 11：每 8 个事件触发一次捕获。
位 1：0	**C1SEL[1：0]：**捕获/比较 1 选择 （Capture/Compare 1 selection） 这 2 位定义通道的方向（输入/输出），及输入脚的选择： 00：CC1 通道被配置为输出； 01：CC1 通道被配置为输入，IC1 映射在 TI1 上； 10：保留； 11：保留。 *注：C1SEL 仅在通道关闭时（TMRx_CCE 寄存器的 C1EN=’0’）才是可写的。*

10.3.5.6 捕获/比较使能寄存器（TMRx_CCE）

偏移地址：0x20

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留												CC1NP	保留	C1P	C1EN
res												rw	res	rw	rw

位 15：4	保留，始终读为 0。
位 3	**C1NP：**输入/捕获 1 互补输出极性 （Capture/Compare 1 complementary output polarity） CC1 通道配置为输出：C1NP 必须保持清除。 CC1 通道配置为输入：C1NP 结合 C1P 定义 TI1FP1 极性（参考 C1P 描述）
位 3	保留，始终读为 0。
位 1	**C1P：**输入/捕获 1 输出极性 （Capture/Compare 1 output polarity） **CC1 通道配置为输出：** 0：OC1 高电平有效 1：OC1 低电平有效 **CC1 通道配置为输入：** CC1NP/CC1P 位选择 TI1FP1 的极性信号作为触发或捕获信号。 00：不反相/上升沿。电路对 TI1FP1 上升沿敏感（捕获模式），TI1FP1 不反相。 01：反相/下降沿。电路对 TI1FP1 下降沿敏感（捕获模式），TI1FP1 反相。 10：保留。 11：不反相/双沿。电路同时对 TI1FP1 上升沿和下降沿敏感（捕获模式），TI1FP1 是不反相。

位 0	**C1EN：**输入/捕获 1 输出使能 （Capture/Compare 1 output enable） CC1 通道配置为输出： 0： 关闭－ OC1 禁止输出。 1： 开启－ OC1 信号输出到对应的输出引脚。 CC1 通道配置为输入： 该位决定了计数器的值是否能捕获入 TMRx_CC1 寄存器。 0：捕获禁止。 1：捕获使能。

表 10-10 标准OCx通道的输出控制位

CxEN 位	OCx 输出状态
0	禁止输出（OCx=0，OCx_EN=0）
1	OCx = OCxREF + 极性，OCx_EN=1

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

10.3.5.7 计数器（TMRx_CNT）

偏移地址：0x24

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT[15：0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 15：0	**CNT[15：0]：**计数器的值 （Counter value）

10.3.5.8 预分频器（TMRx_DIV）

偏移地址：0x28

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DIV[15：0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 15：0	**DIV[15：0]：**预分频器的值 （Prescaler value） 计数器的时钟频率 CK_CNT 等于 f_{CK_DIV}/（DIV[15：0]+1）。 DIV 包含了当更新事件产生时装入当前预分频器寄存器的值。

10.3.5.9 自动重装载寄存器（TMRx_AR）

偏移地址：0x2C

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AR[15：0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 15：0	**AR[15：0]：**自动重装载的值 （Auto reload value） AR 包含了将要传送至实际的自动重装载寄存器的数值。

10.3.5.10 捕获/比较寄存器1（TMRx_CC1）

偏移地址：0x34

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CC1[15：0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 15：0	**CC1[15：0]：**捕获/比较 1 的值（Capture/Compare 1 value） **若 CC1 通道配置为输出：** CC1 包含了装入当前捕获/比较 1 寄存器的值（预装载值）。 如果在 TMRx_CCM1 寄存器（OC1PEN 位）中未选择预装载特性，写入的数值会被立即传输至当前寄存器中。否则只有当更新事件发生时，此预装载值才传输至当前捕获/比较寄存器 1 中。 当前捕获/比较寄存器参与同计数器 TMRx_CNT 的比较，并在 OC1 端口上产生输出信号。 **若 CC1 通道配置为输入：** CC1 包含了由上一次输入捕获 1 事件（IC1）传输的计数器值。

10.4 高级控制定时器（TMR1、TMR8）

10.4.1 TMR1、TMR8简介

高级控制定时器（TMR1、TMR8）由一个 16 位的自动装载计数器组成，它由一个可编程的预分频器驱动。

它适合多种用途，包含测量输入信号的脉冲宽度（输入捕获），或者产生输出波形（输出比较、PWM、嵌入死区时间的互补 PWM 等）。

使用定时器预分频器和 RCC 时钟控制预分频器，可以实现脉冲宽度和波形周期从几个微秒到几个毫秒的调节。

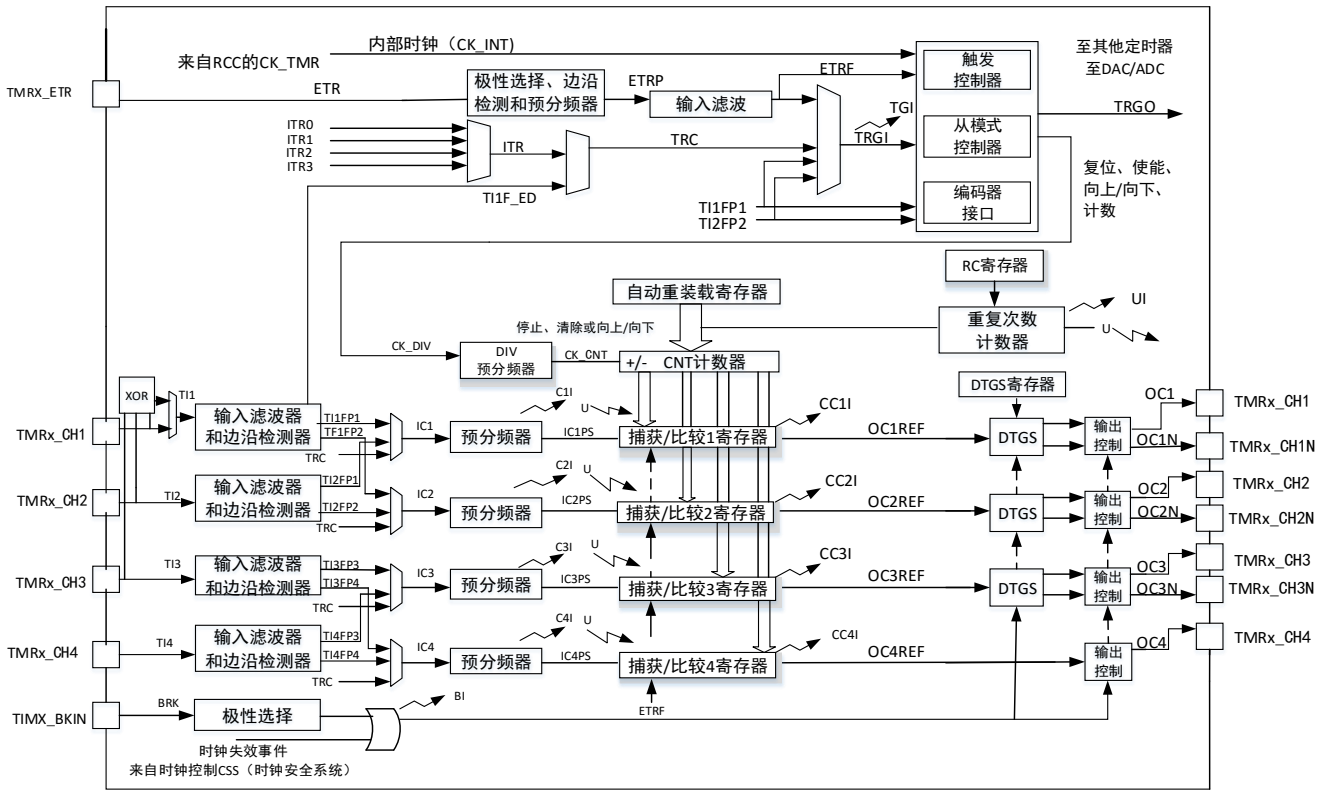
高级控制定时器（TMR1、TMR8）和通用定时器（TMRx）是完全独立的，它们不共享任何资源。它们可以同步操作，具体描述参看 10.4.3.20 节。

10.4.2 TMR1、TMR8主要特性

TMR1、TMR8 定时器的功能包括：

- 16位向上、向下、向上/下自动装载计数器
- 16位可编程（可以实时修改）预分频器，计数器时钟频率的分频系数为1～65536之间的任意数值
- 多达4个独立通道：
  - 输入捕获
  - 输出比较
  - PWM生成（边缘或中间对齐模式）
  - 单脉冲模式输出
- 死区时间可编程的互补输出
- 使用外部信号控制定时器和定时器互联的同步电路
- 允许在指定数目的计数器周期之后更新定时器寄存器的重复计数器
- 刹车输入信号可以将定时器输出信号置于复位状态或者一个已知状态
- 如下事件发生时产生中断/DMA：
  - 更新：计数器向上溢出/向下溢出，计数器初始化（通过软件或者内部/外部触发）
  - 触发事件（计数器启动、停止、初始化或者由内部/外部触发计数）
  - 输入捕获
  - 输出比较
  - 刹车信号输入
- 支持针对定位的增量（正交）编码器和霍尔传感器电路
- 触发输入作为外部时钟或者按周期的电流管理

图 10-80 高级控制定时器框图



注意： Reg *根据控制位的设定，在 U（更新）事件时传送预加载寄存器的内容至工作寄存器*

事件

中断和 DMA 输出

10.4.3 TMR1、TMR8功能描述

10.4.3.1 时基单元

可编程高级控制定时器的主要部分是一个 16 位计数器和与其相关的自动装载寄存器。这个计数器可以向上计数、向下计数或者向上向下双向计数。此计数器时钟由预分频器分频得到。

计数器、自动装载寄存器和预分频器寄存器可以由软件读写，即使计数器还在运行读写仍然有效。

时基单元包含：

- 计数器寄存器（TMRx_CNT）
- 预分频器寄存器 （TMRx_DIV）
- 自动装载寄存器 （TMRx_AR）
- 重复次数寄存器 （TMRx_RC）

自动装载寄存器是预先装载的，写或读自动重装载寄存器将访问预装载寄存器。根据在 TMRx_CTRL1 寄存器中的自动装载预装载使能位（ARPEN）的设置，预装载寄存器的内容被立即或在每次的更新事件 UEV 时传送到影子寄存器。当计数器达到溢出条件（向下计数时的下溢条件）并当 TMRx_CTRL1 寄存器中的 UEVDIS 位等于 0 时，产生更新事件。更新事件也可以由软件产生。随后会详细描述每一种配置下更新事件的产生。

计数器由预分频器的时钟输出 CK_CNT 驱动，仅当设置了计数器 TMRx_CTRL1 寄存器中的计数器使能位（CNTEN）时，CK_CNT 才有效。（更多有关使能计数器的细节，请参见控制器的从模式描述）。

注意，在设置了 TMRx_CTRL 寄存器的 CNTEN 位的一个时钟周期后，计数器开始计数。

预分频器描述

预分频器可以将计数器的时钟频率按 1 到 65536 之间的任意值分频。它是基于一个（在 TMRx_DIV 寄

存器中的）16 位寄存器控制的 16 位计数器。因为这个控制寄存器带有缓冲器，它能够在运行时被改变。新的预分频器的参数在下一次更新事件到来时被采用。

图 10-81 和图 10-82 给出了在预分频器运行时，更改计数器参数的例子。

图 10-81 当预分频器的参数从1变到2时，计数器的时序图

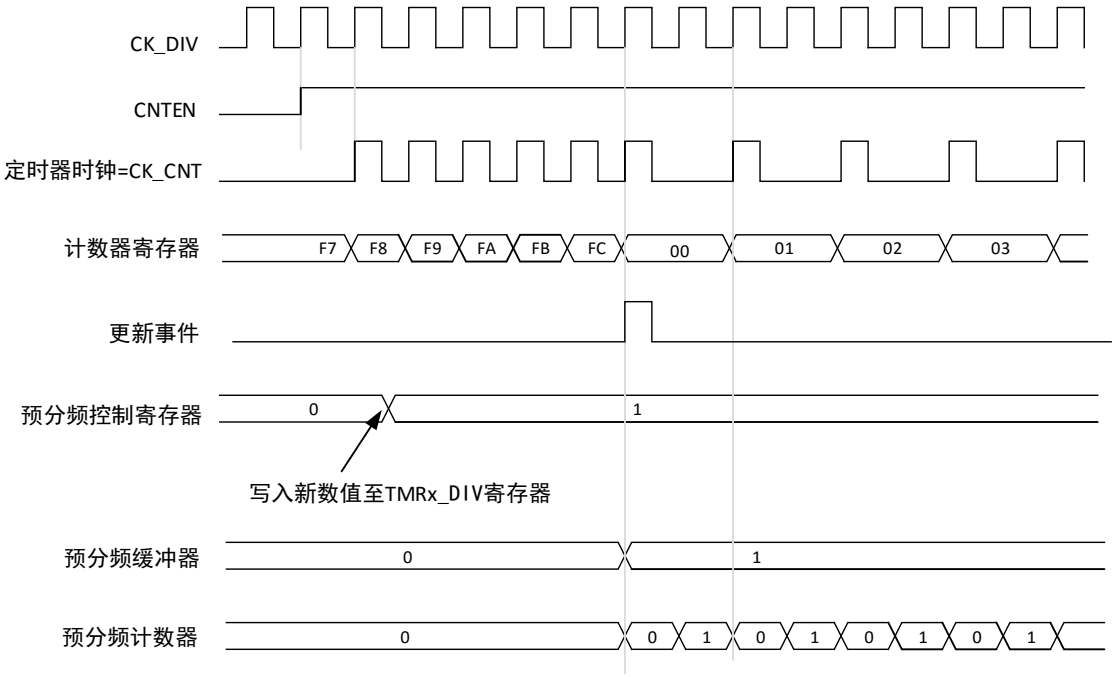
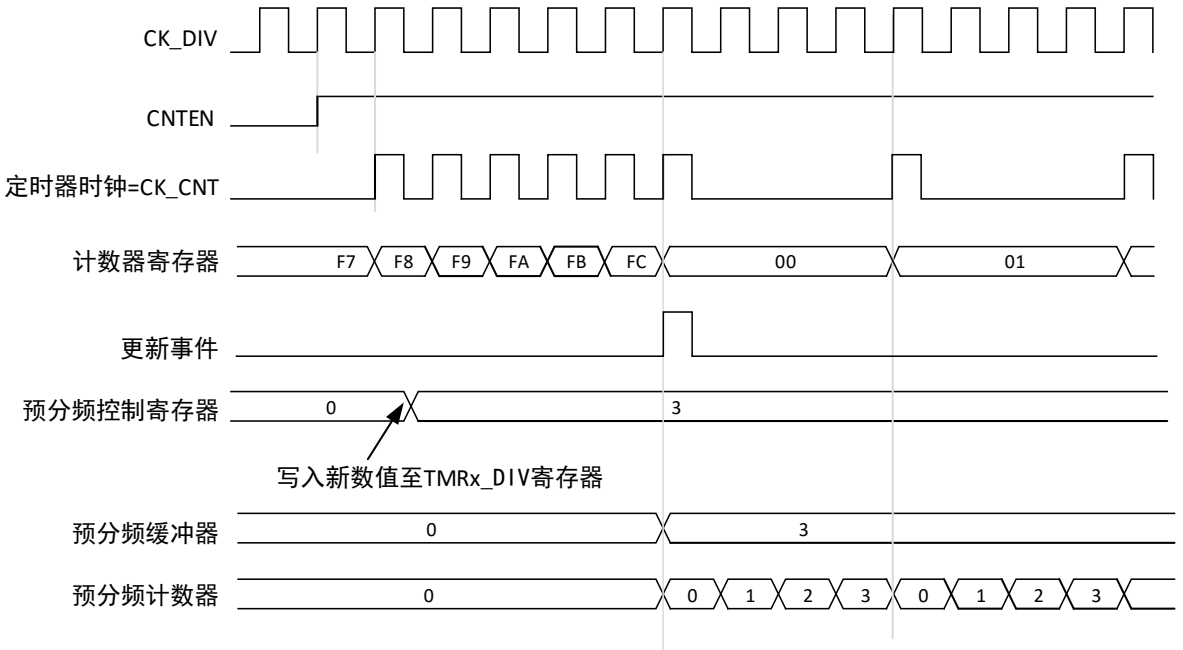


图 10-82 当预分频器的参数从1变到4时，计数器的时序图



10.4.3.2 计数器模式

向上计数模式

在向上计数模式中，计数器从 0 计数到自动加载值（TMRx_AR 计数器的内容），然后重新从 0 开始计数并且产生一个计数器溢出事件。

如果使用了重复计数器功能，在向上计数达到设置的重复计数次数（TMRx_RC）时，才会产生更新事件（UEV）；否则每次计数器溢出时都会产生更新事件。

在 TMRx_EVEG 寄存器中（通过软件方式或者使用从模式控制器）设置 UEVG 位也同样可以产生一个更新事件。

设置 TMRx_CTRL1 寄存器中的 UEVDIS 位，可以禁止更新事件；这样可以避免在向预装载寄存器中写入新值时更新影子寄存器。在 UEVDIS 位被清'0'之前，将不产生更新事件。但是在应该产生更新事件时，计数器仍会被清'0'，同时预分频器的计数也被清 0（但预分频器的数值不变）。此外，如果设置了 TMRx_CTRL1 寄存器中的 UVERS 位（选择更新请求），设置 UEVG 位将产生一个更新事件 UEV，但硬件不设置 UEVIF 标志（即不产生中断或 DMA 请求）。这是为了避免在捕获模式下清除计数器时，同时产生更新和捕获中断。

当发生一个更新事件时，所有的寄存器都被更新，硬件同时（依据 UVERS 位）设置更新标志位（TMRx_STS 寄存器中的 UEVIF 位）。

- 重复计数器被重新加载为TMRx_RC寄存器的内容。
- 自动装载影子寄存器被重新置入预装载寄存器的值（TMRx_AR）。
- 预分频器的缓冲区被置入预装载寄存器的值（TMRx_DIV寄存器的内容）。

下图给出一些例子，当 TMRx_AR=0x36 时计数器在不同时钟频率下的动作。

图 10-83 计数器时序图，内部时钟分频因子为1

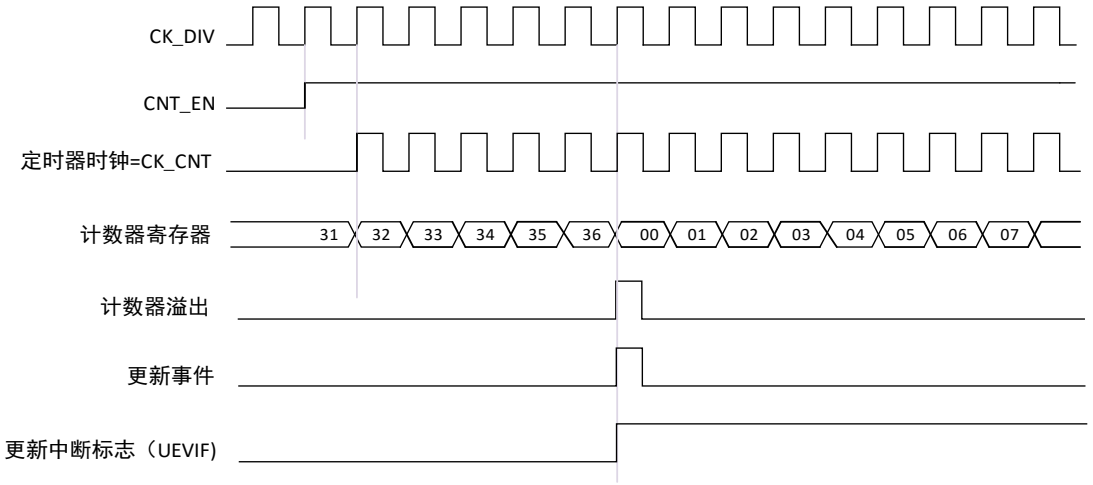


图 10-84 计数器时序图，内部时钟分频因子为2

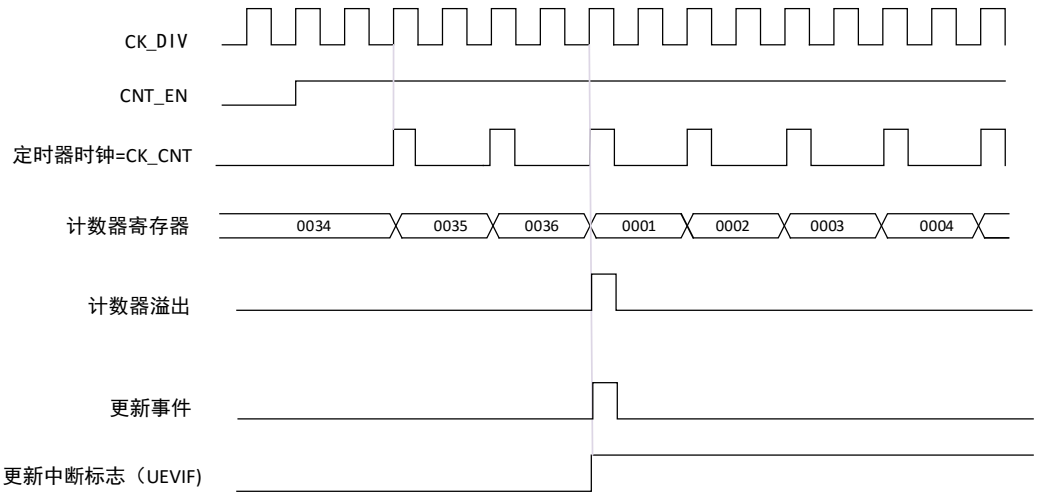


图 10-85 计数器时序图，内部时钟分频因子为4

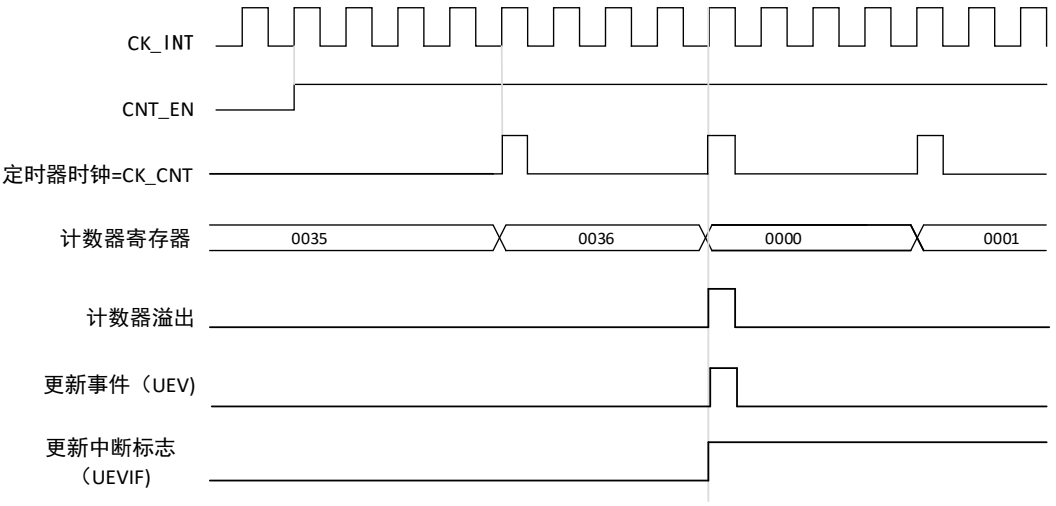


图 10-86 计数器时序图，内部时钟分频因子为N

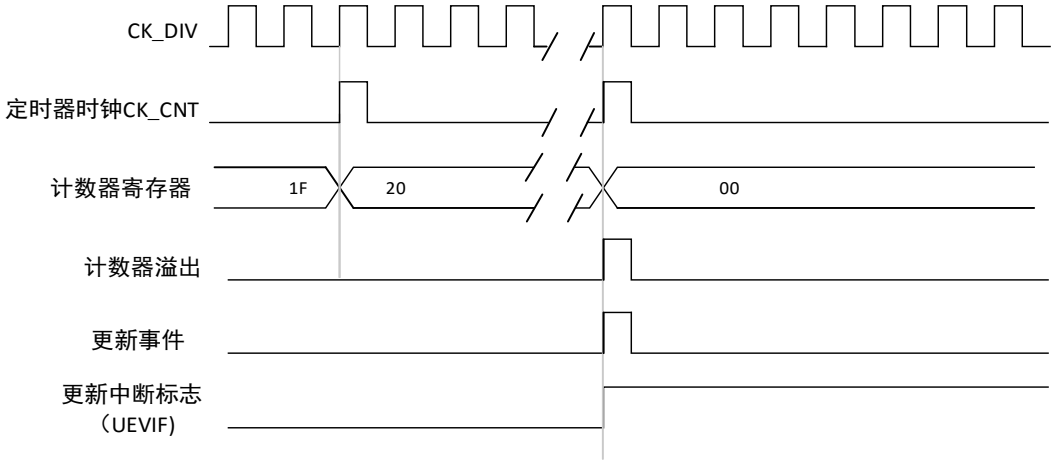


图 10-87 计数器时序图，当ARPEN=0时的更新事件（TMRx_AR没有预装入）

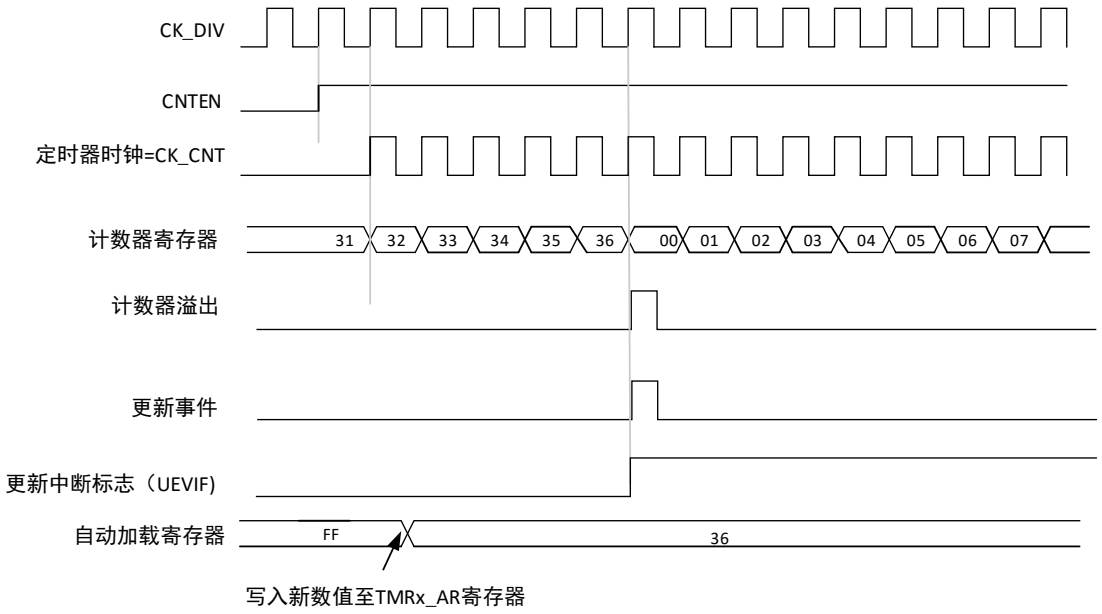
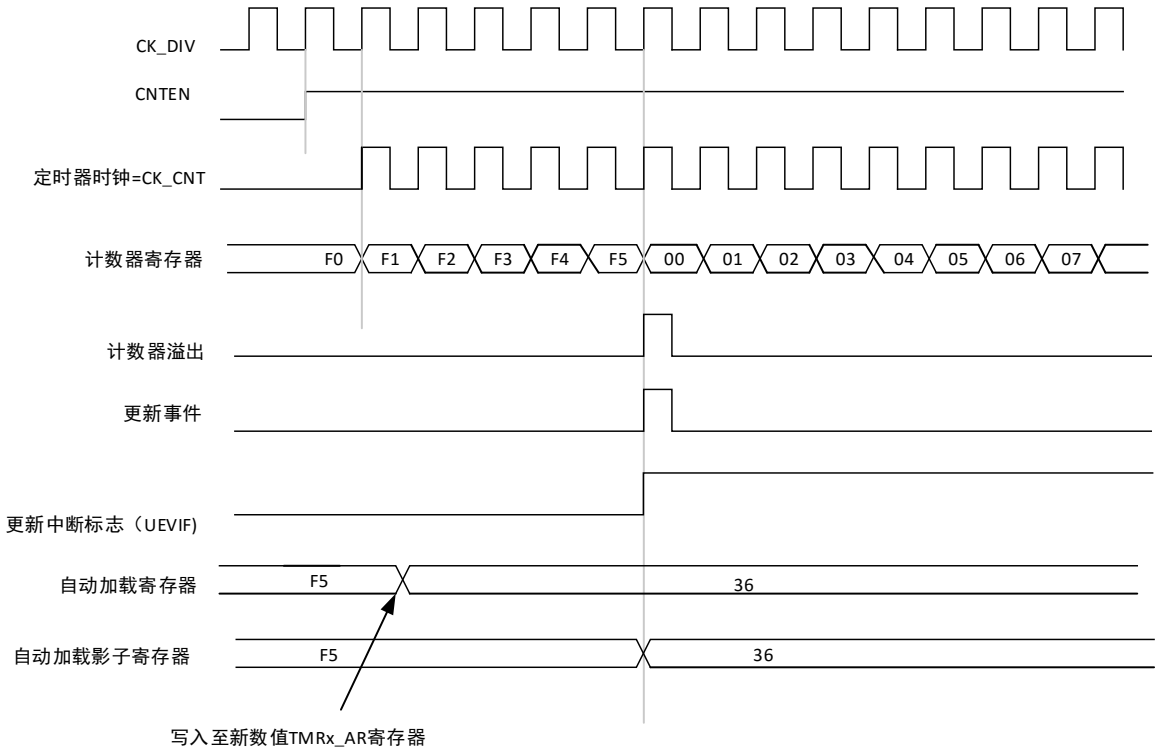


图 10-88 计数器时序图，当ARPEN=1时的更新事件（预装入了TMRx_AR）



向下计数模式

在向下模式中，计数器从自动装入的值（TMRx_AR 计数器的值）开始向下计数到 0，然后从自动装入的值重新开始并且产生一个计数器向下溢出事件。

如果使用了重复计数器，当向下计数达到重复计数寄存器（TMRx_RC）中设定的次数后，才会产生更新事件（UEV），否则每次计数器下溢时都会产生更新事件。

在 TMRx_EVEG 寄存器中（通过软件方式或者使用从模式控制器）设置 UEVG 位，也同样可以产生一个更新事件。

设置 TMRx_CTRL1 寄存器的 UEVDIS 位可以禁止 UEV 事件。这样可以避免向预装载寄存器中写入新值时更新影子寄存器。因此 UEVDIS 位被清为 0 之前不会产生更新事件。然而，计数器仍会从当前自动加载值重新开始计数，并且预分频器的计数器重新从 0 开始（但预分频系数不变）。此外，如果设置了 TMRx_CTRL1 寄存器中的 UVERS 位（选择更新请求），设置 UEVG 位将产生一个更新事件 UEV 但不设置 UEVIF 标志（因此不产生中断和 DMA 请求），这是为了避免在发生捕获事件并清除计数器时，同时产生更新和捕获中断。

当发生更新事件时，所有的寄存器都被更新，并且（根据 UVERS 位的设置）更新标志位（TMRx_STS 寄存器中的 UEVIF 位）也被设置。

- 重复计数器被重置为TMRx_RC寄存器中的内容
- 预分频器的缓存器被加载为预装载的值（TMRx_DIV寄存器的值）。
- 当前的自动加载寄存器被更新为预装载值（TMRx_AR寄存器中的内容）。

注意：　自动装载在计数器重载入之前被更新，因此下一个周期将是预期的值。

以下是一些当 TMRx_AR=0x36 时，计数器在不同时钟频率下的操作例子。

图 10-89 计数器时序图，内部时钟分频因子为1

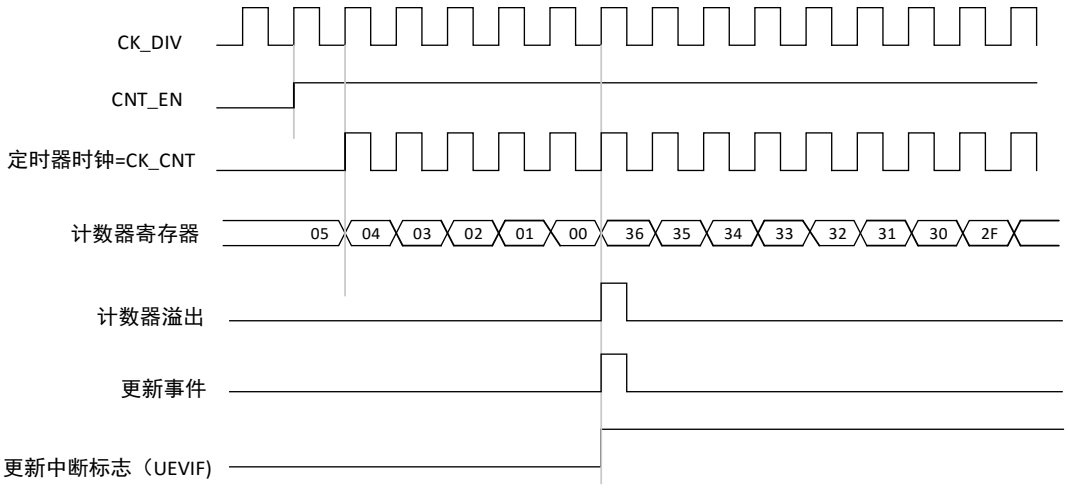


图 10-90 计数器时序图，内部时钟分频因子为2

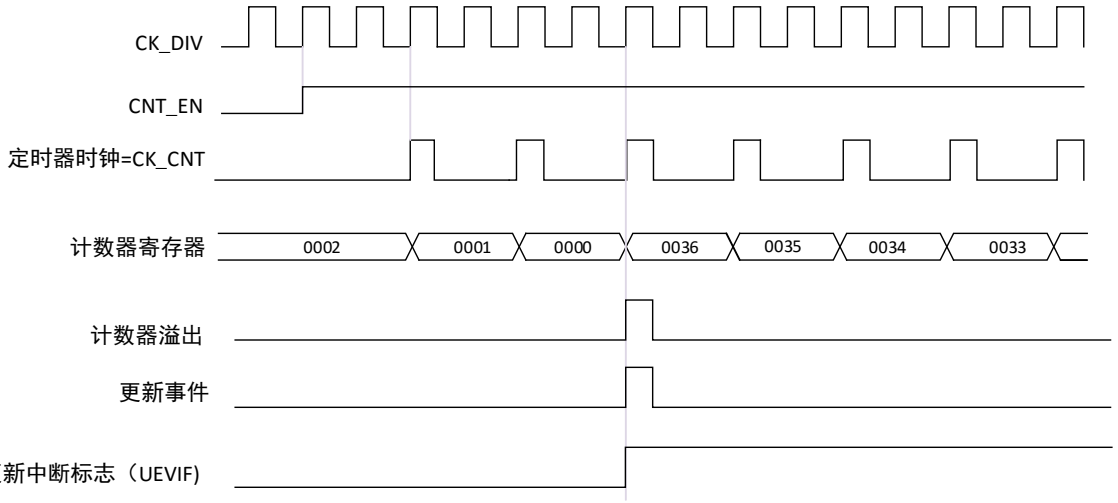


图 10-91 计数器时序图，内部时钟分频因子为4

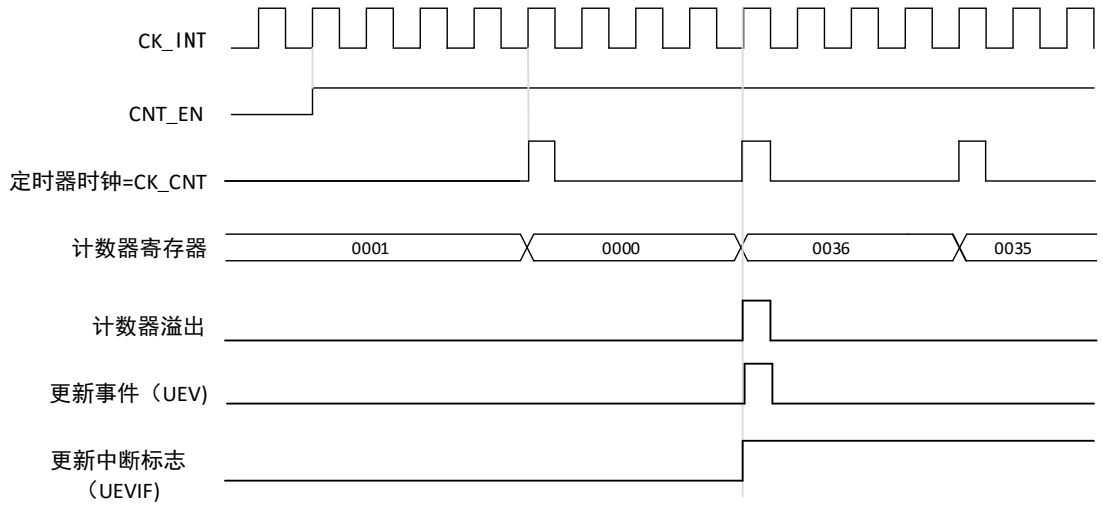


图 10-92 计数器时序图，内部时钟分频因子为N

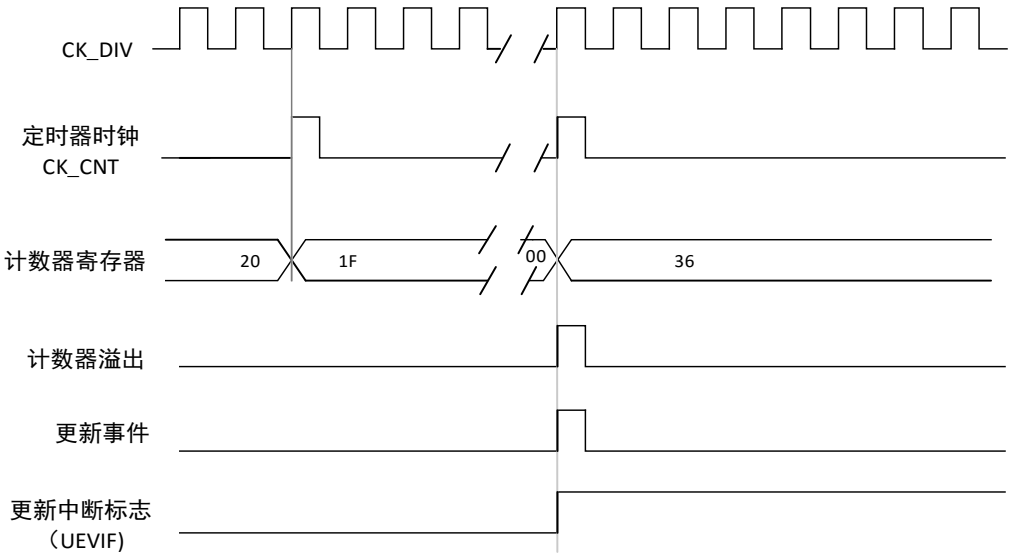
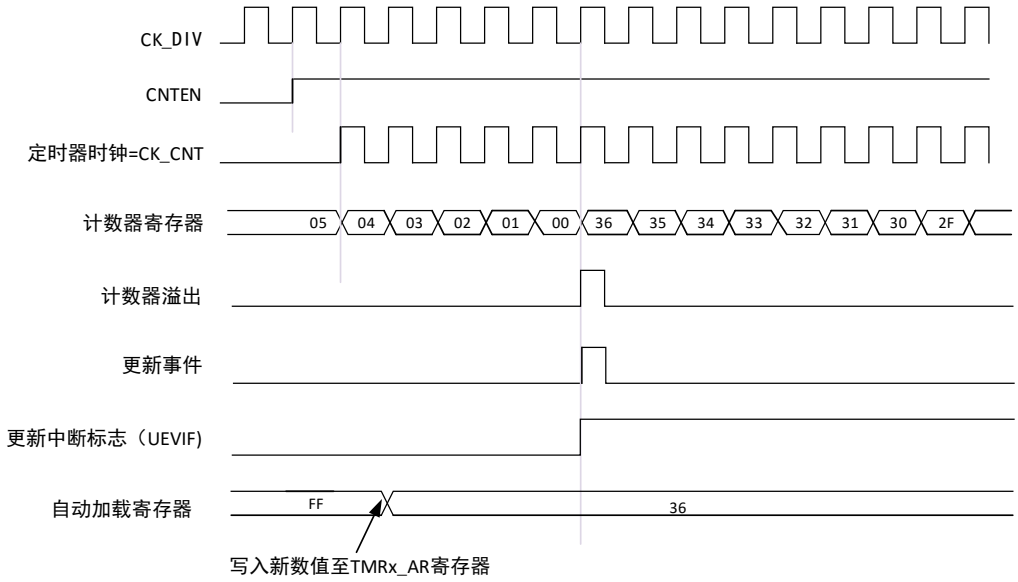


图 10-93 计数器时序图，当没有使用自动装载时的更新事件



中央对齐模式（向上/向下计数）

在中央对齐模式，计数器从 0 开始计数到自动加载的值（TMRx_AR 寄存器）−1，产生一个计数器溢出事件，然后向下计数到 1 并且产生一个计数器下溢事件；然后再从 0 开始重新计数。

在此模式下，不能写入 TMRx_CTRL1 中的 DIR 方向位。它由硬件更新并指示当前的计数方向。

可以在每次计数上溢和每次计数下溢时产生更新事件；也可以通过（软件或者使用从模式控制器）设置 TMRx_EVEG 寄存器中的 UEVG 位产生更新事件。然后，计数器重新从 0 开始计数，预分频器也重新从 0 开始计数。

设置 TMRx_CTRL1 寄存器中的 UEVDIS 位可以禁止 UEV 事件。这样可以避免在向预装载寄存器中写入新值时更新影子寄存器。因此 UEVDIS 位被清为 0 之前不会产生更新事件。然而，计数器仍会根据当前自动重加载的值，继续向上或向下计数。此外，如果设置了 TMRx_CTRL1 寄存器中的 UVERS 位（选择更新请求），设置 UEVG 位将产生一个更新事件 UEV 但不设置 UEVIF 标志（因此不产生中断和 DMA 请求），这是为了避免在发生捕获事件并清除计数器时，同时产生更新和捕获中断。

当发生更新事件时，所有的寄存器都被更新，并且（根据 UVERS 位的设置）更新标志位（TMRx_STS 寄存器中的 UEVIF 位）也被设置。

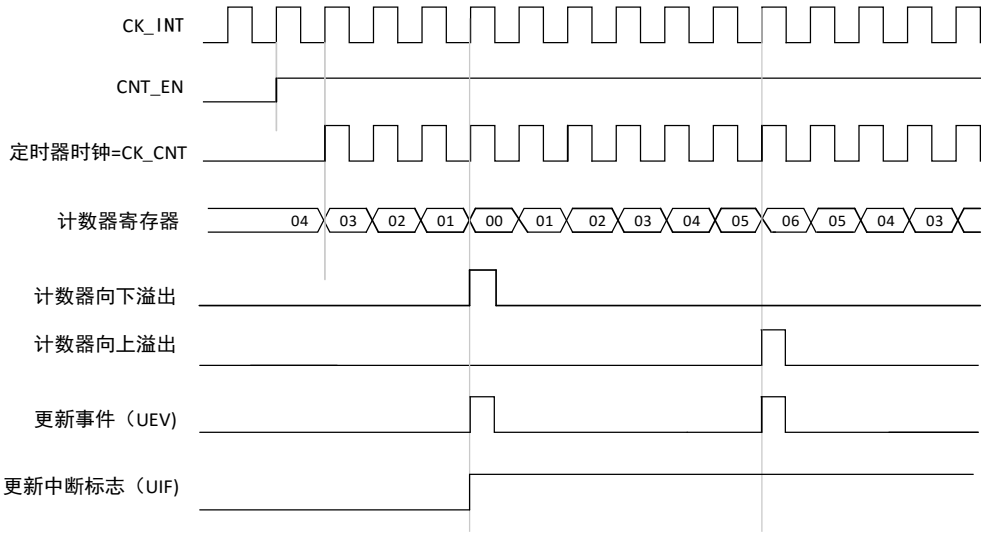
- 重复计数器被重置为TMRx_RC寄存器中的内容
- 预分频器的缓存器被加载为预装载（TMRx_DIV寄存器）的值。

- 当前的自动加载寄存器被更新为预装载值（TMRx_AR寄存器中的内容）。

注意： *如果因为计数器溢出而产生更新，自动重装载将在计数器重载入之前被更新，因此下一个周期将是预期的值（计数器被装载为新的值）。*

以下是一些计数器在不同时钟频率下的操作的例子：

图 10-94 计数器时序图，内部时钟分频因子为1，TMRx_AR=0x6



注意： *这里使用了中心对齐模式 1（详见 10.4.4.1TMR1、TMR8 控制寄存器 1（TMRx_CTRL1））。*

图 10-95 计数器时序图，内部时钟分频因子为2

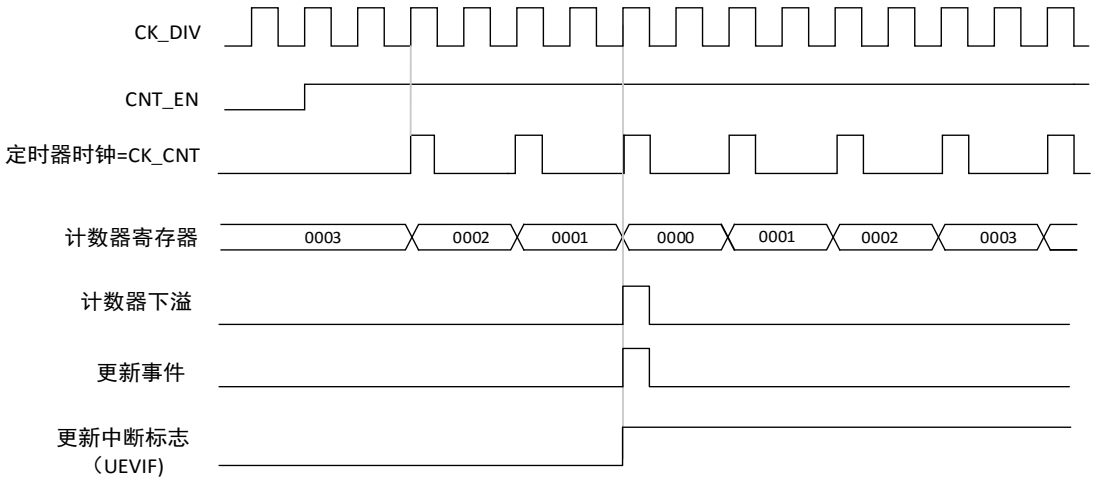
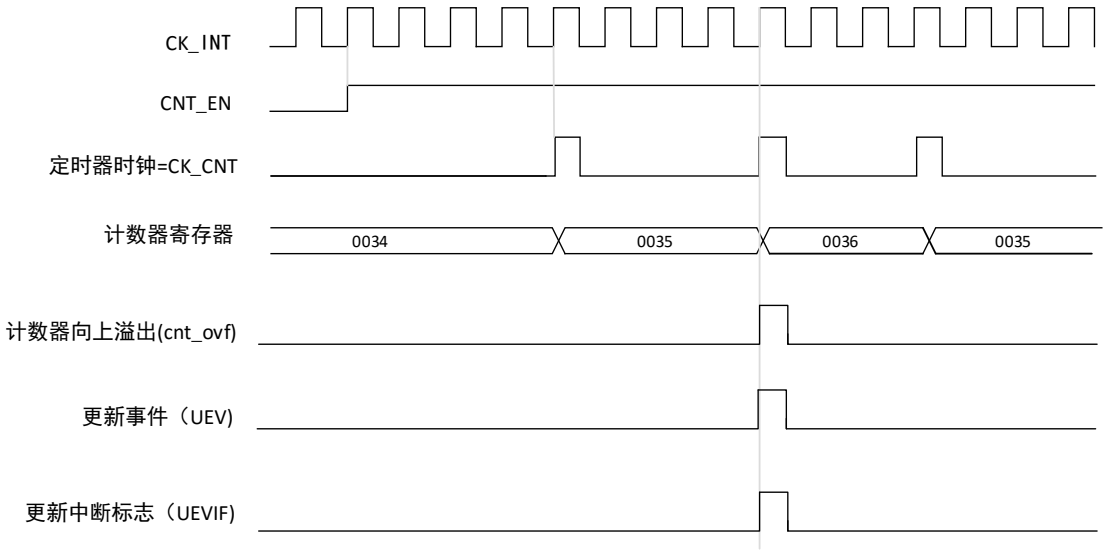


图 10-96 计数器时序图，内部时钟分频因子为4，TMRx_AR=0x36



注：这里使用了中心对齐模式2或3，计数器溢出时设置UEVIF

图 10-97 计数器时序图，内部时钟分频因子为N

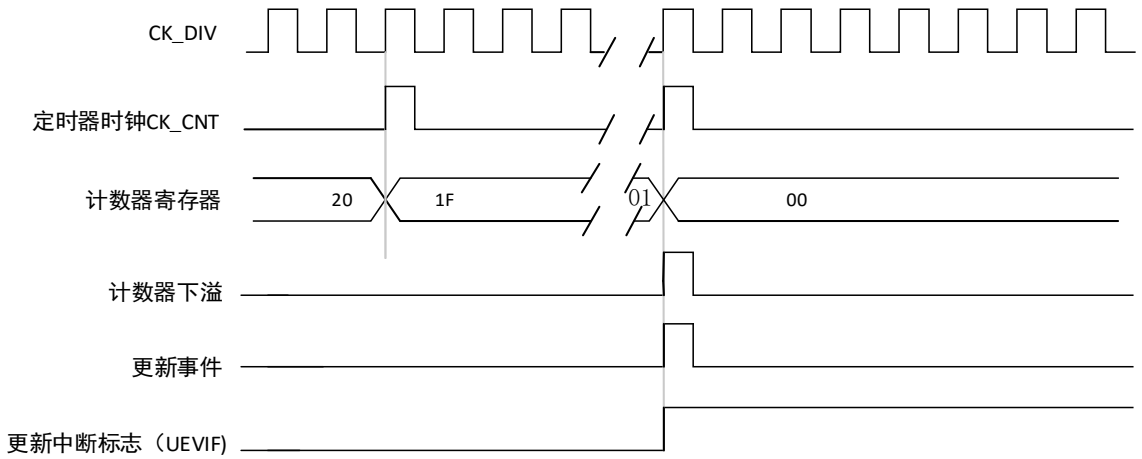


图 10-98 计数器时序图，ARPEN=1时的更新事件（计数器下溢）

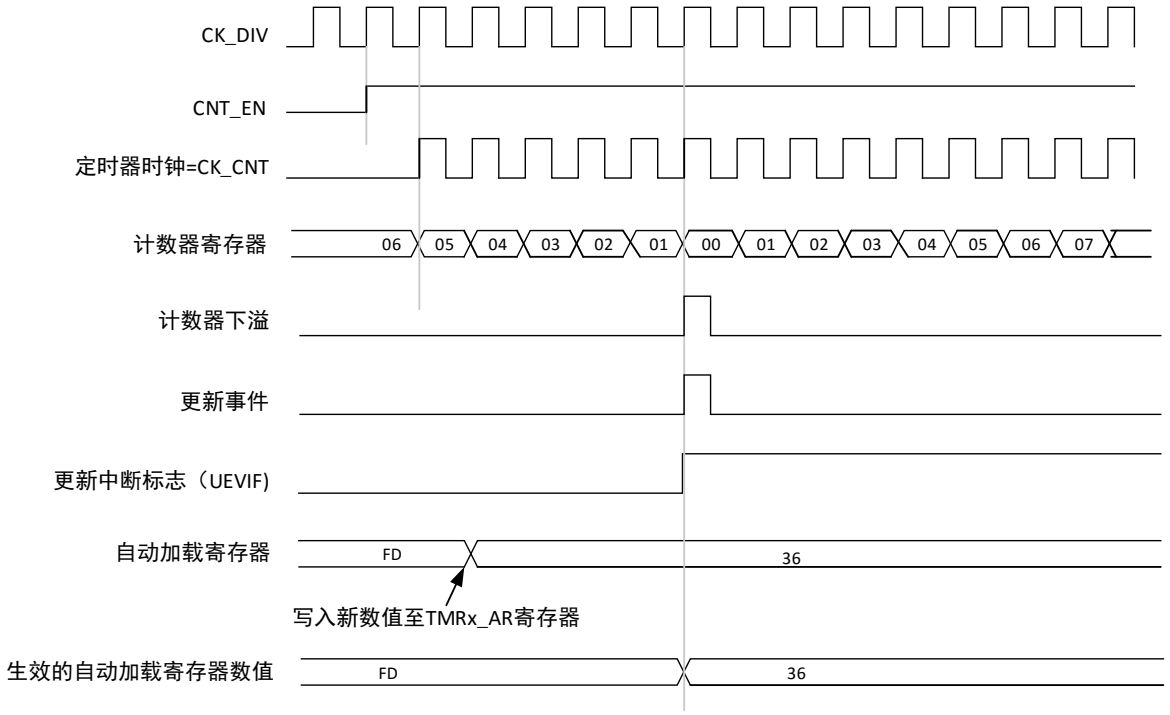
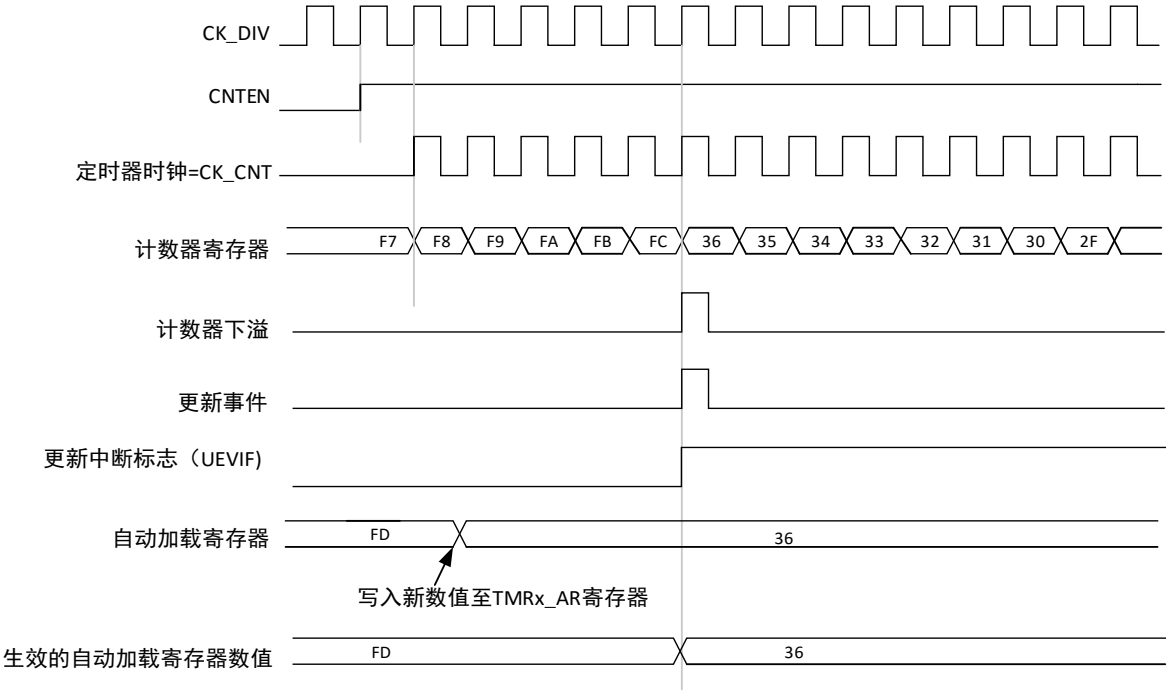


图 10-99 计数器时序图，ARPEN=1时的更新事件（计数器溢出）



10.4.3.3 重复计数器

10.4.3.1 节“时基单元”解释了计数器上溢/下溢时更新事件是如何产生的，然而事实上它只能在重复计数达到 0 的时候产生。这个特性对产生 PWM 信号非常有用。

这意味着在每 N 次计数上溢或下溢时，数据从预装载寄存器传输到影子寄存器（TMRx_AR 自动重载入寄存器，TMRx_DIV 预装载寄存器，还有在比较模式下的捕获/比较寄存（TMRx_CCx），N 是 TMRx_RC 重复计数寄存器中的值。

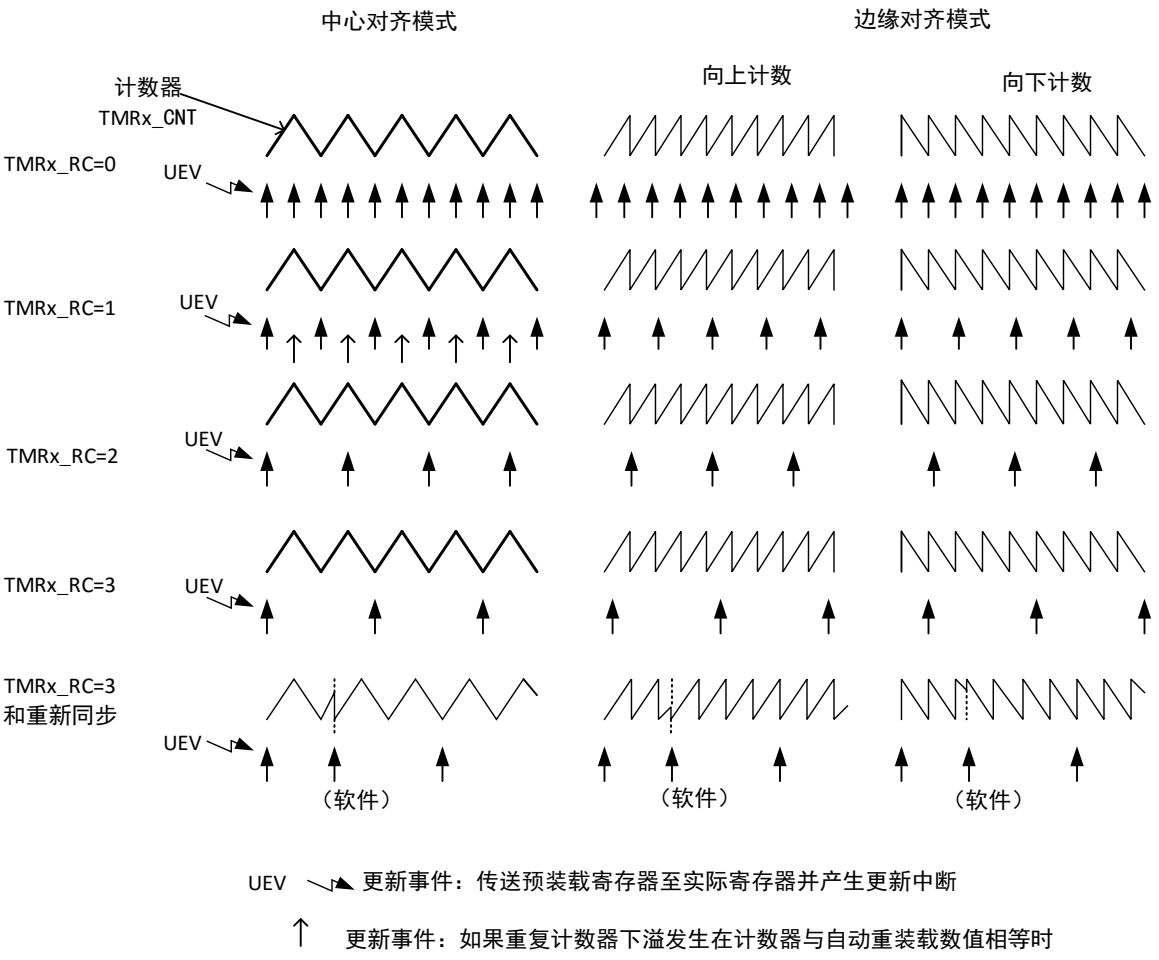
重复计数器在下述任一条件成立时递减：

- 向上计数模式下每次计数器溢出时

- 向下计数模式下每次计数器下溢时
- 中央对齐模式下每次上溢和每次下溢时。虽然这样限制了PWM的最大循环周期为128，但它能够在每个PWM周期2次更新占空比。在中央对齐模式下，因为波形是对称的，如果每个PWM周期中仅刷新一次比较寄存器，则最大的分辨率为2xT_{ck}。

重复计数器是自动加载的，重复速率是由 TMRx_RC 寄存器的值定义（参看图 10-100）。当更新事件由软件产生（通过设置 TMRx_EVEG 中的 UEVG 位）或者通过硬件的从模式控制器产生，则无论重复计数器的值是多少，立即发生更新事件，并且 TMRx_RC 寄存器中的内容被重载入到重复计数器。

图 10-100 不同模式下更新速率的例子，及TMRx_RC的寄存器设置



10.4.3.4 时钟选择

计数器时钟可由下列时钟源提供：

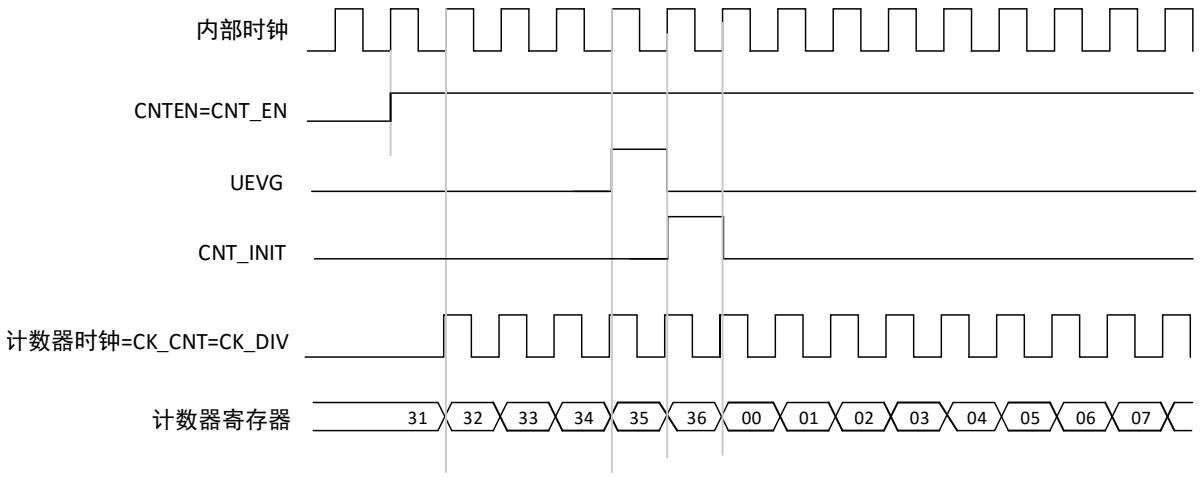
- 内部时钟（CK_INT）
- 外部时钟模式1：外部输入引脚
- 外部时钟模式2：外部触发输入ETR
- 内部触发输入（ITRx）：使用一个定时器作为另一个定时器的预分频器。如可以配置一个定时器Timer1而作为另一个定时器Timer2的预分频器。详见10.4.3.20节。

内部时钟源（CK_INT）

如果禁止了从模式控制器（SMSEL=000），则 CNTEN、DIR（TMRx_CTRL1 寄存器）和 UEVG 位（TMRx_EVEG 寄存器）是实际的控制位，并且只能被软件修改（UEVG 位仍被自动清除）。只要 CNTEN 位被写成'1'，预分频器的时钟就由内部时钟 CK_INT 提供。

下图显示控制电路和向上计数器在一般模式下，不带预分频器时的操作。

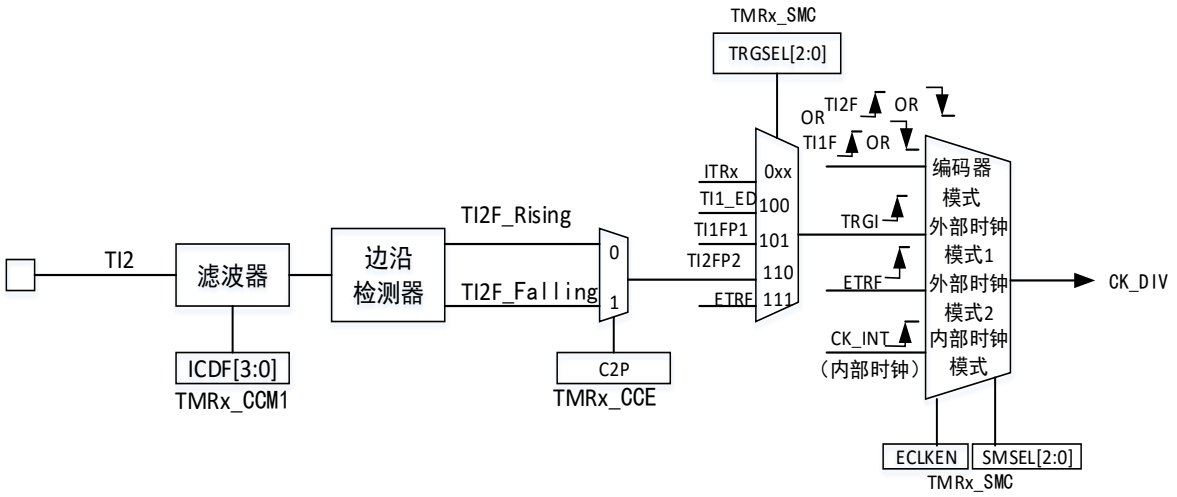
图 10-101 一般模式下的控制电路，内部时钟分频因子为1



外部时钟源模式 1

当 TMRx_SMC 寄存器的 SMSEL=111 时，此模式被选中。计数器可以在选定输入端的每个上升沿或下降沿计数。

图 10-102 TI2外部时钟连接例子



例如，要配置向上计数器在 TI2 输入端的上升沿计数，使用下列步骤：

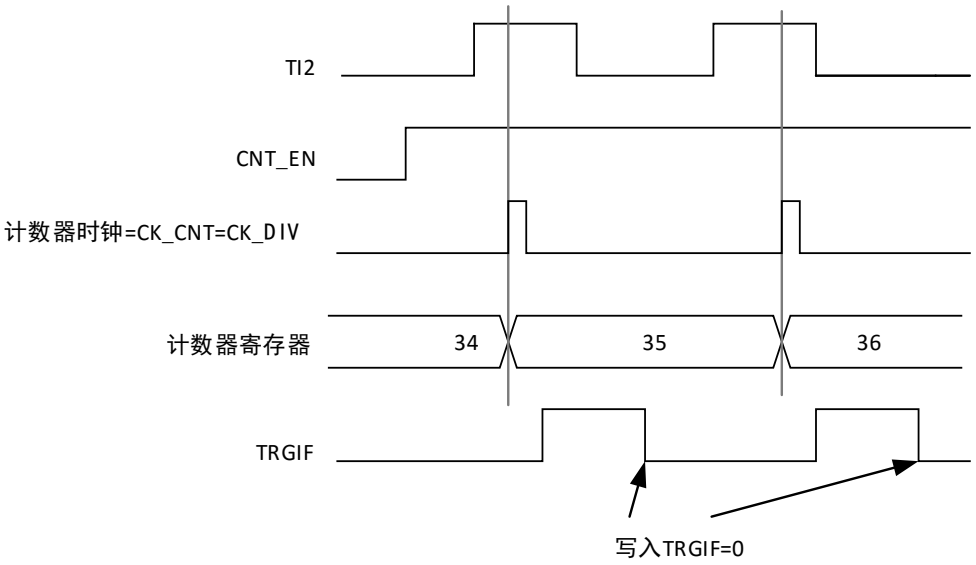
1. 配置 TMRx_CCM1 寄存器 C2SEL=01，配置通道 2 检测 TI2 输入的上升沿。
2. 配置 TMRx_CCM1 寄存器的 IC2DF[3:0],选择输入滤波器带宽（如果不需要滤波器，保持 IC2DF = 0000）。
3. 配置 TMRx_CCE 寄存器的 C2P=0，选定上升沿极性。
4. 配置 TMRx_SMC 寄存器的 SMSEL=111，选择定时器外部时钟模式 1。
5. 配置 TMRx_SMC 寄存器中的 TRGSEL=110，选定 TI2 作为触发输入源。
6. 设置 TMRx_CTRL1 寄存器的 CNTEN=1，启动计数器。

注意：　捕获预分频器不用作触发，所以不需要对它进行配置。

当上升沿出现在 TI2，计数器计数一次，且 TRGIF 标志被设置。

在 TI2 的上升沿和计数器实际时钟之间的延时，取决于在 TI2 输入端的重新同步电路。

图 10-103 外部时钟模式1下的控制电路



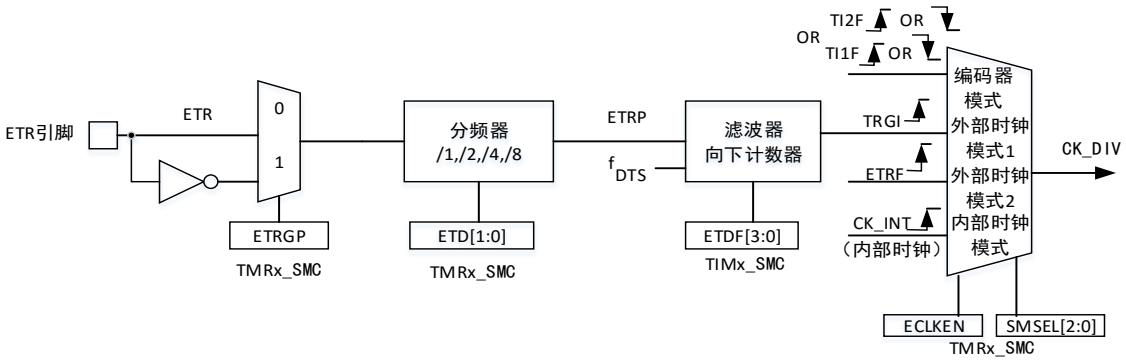
外部时钟源模式 2

选定此模式的方法为：令 TMRx_SMC 寄存器中的 ECLKEN=1。

计数器能够在外部触发 ETR 的每一个上升沿或下降沿计数。

下图是外部触发输入的框图：

图 10-104 外部触发输入框图



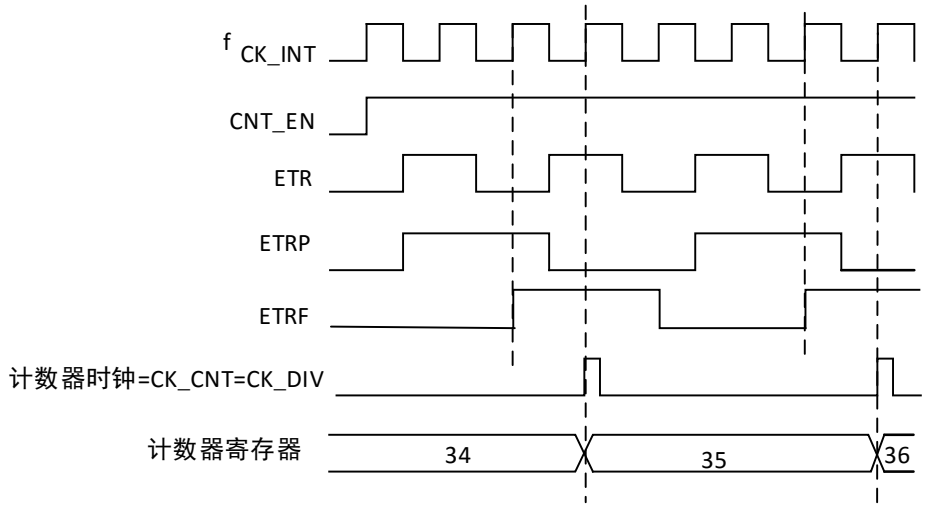
例如，要配置在 ETR 下每 2 个上升沿计数一次的向上计数器，使用下列步骤：

1. 本例中不需要滤波器，设置TMRx_SMC寄存器中的ETDF[3：0]=0000。
2. 设置预分频器，置TMRx_SMC寄存器中的ETD[1：0]=01。
3. 选择ETR的上升沿检测，置TMRx_SMC寄存器中的ETRGP=0。
4. 开启外部时钟模式2，写TMRx_SMC寄存器中的ECLKEN=1。
5. 启动计数器，写TMRx_CTRL1寄存器中的CNTEN=1。

计数器在每 2 个 ETR 上升沿计数一次。

在 ETR 的上升沿和计数器实际时钟之间的延时取决于在 ETRP 信号端的重新同步电路。

图 10-105 外部时钟模式2下的控制电路



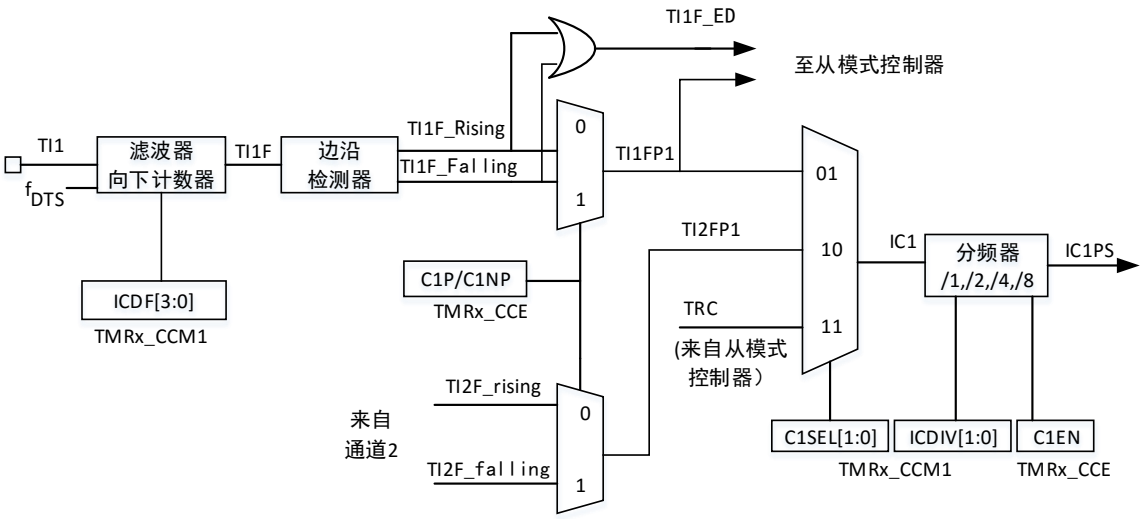
10.4.3.5 捕获/比较通道

每一个捕获/比较通道都是围绕着一个捕获/比较寄存器（包含影子寄存器），包括捕获的输入部分（数字滤波、多路复用和预分频器），和输出部分（比较器和输出控制）。

图 10-106 至图 10-109 是一个捕获/比较通道概览。

输入部分对相应的 TIx 输入信号采样，并产生一个滤波后的信号 TIxF。然后，一个带极性选择的边缘监测器产生一个信号（TIxFPx），它可以作为从模式控制器的输入触发或者作为捕获控制。该信号通过预分频进入捕获寄存器（ICxPS）。

图 10-106 捕获/比较通道（如：通道1输入部分）



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

图 10-107 捕获/比较通道1的主电路

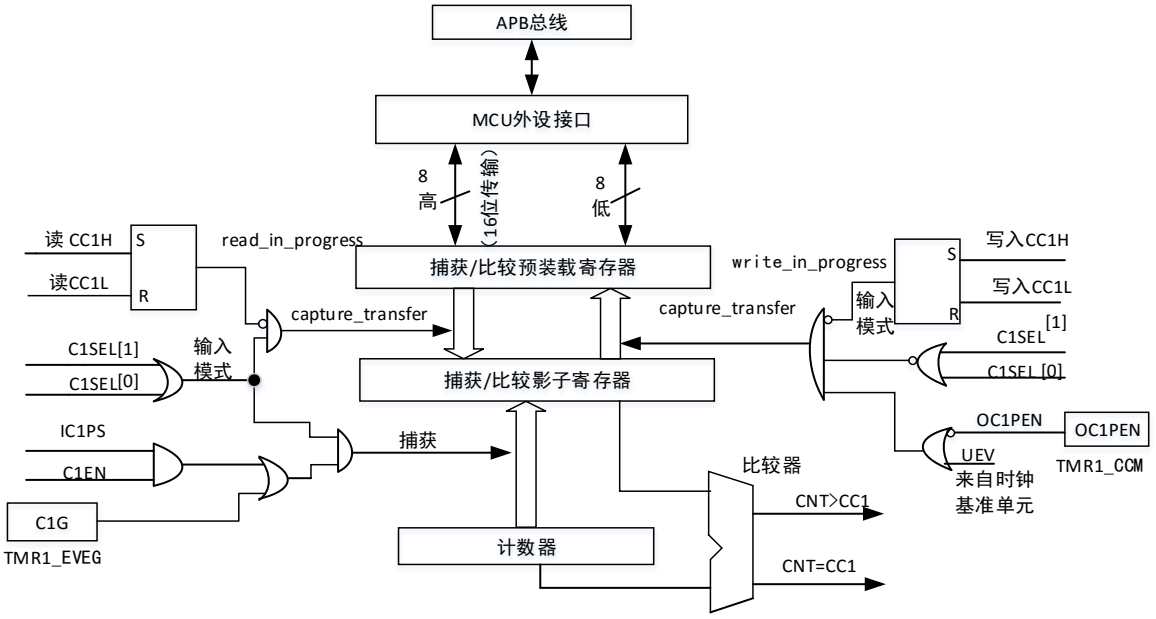


图 10-108 捕获/比较通道的输出部分（通道1至3）

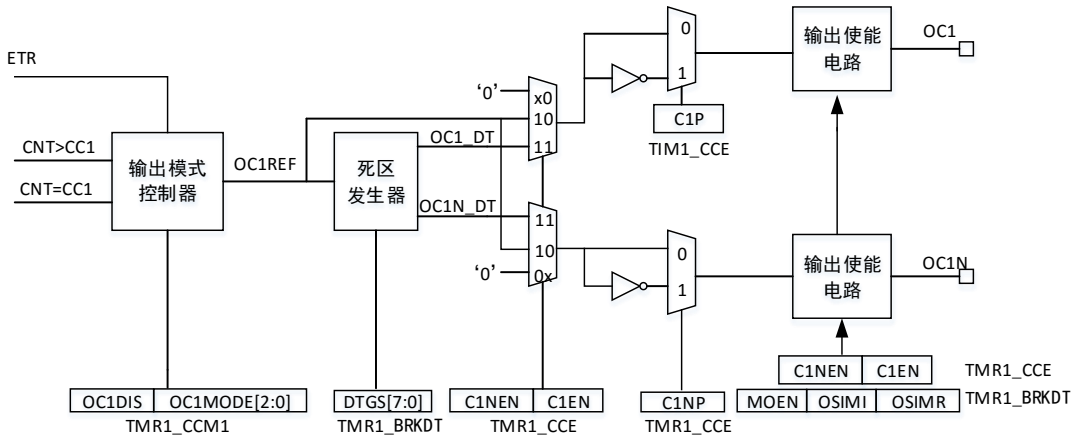
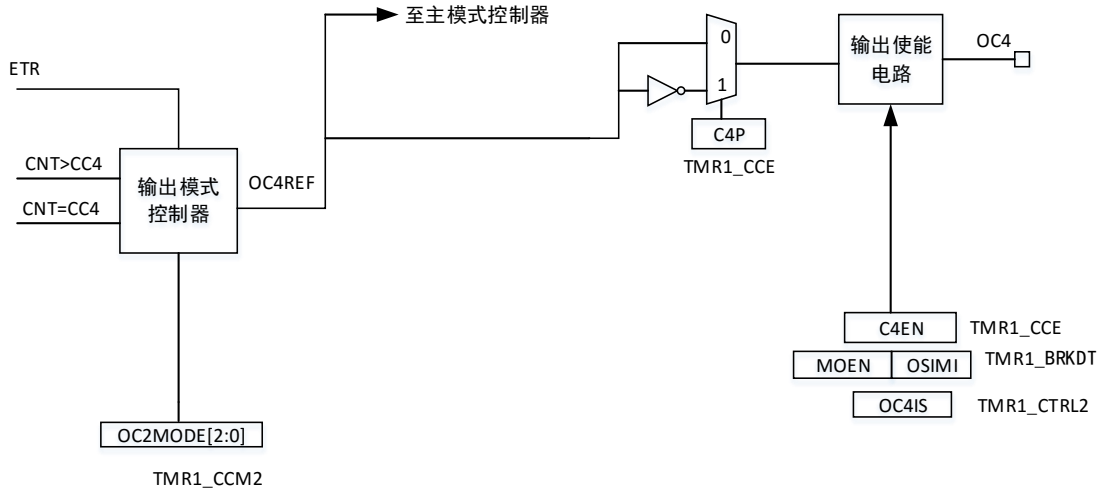


图 10-109 捕获/比较通道的输出部分（通道4）



捕获/比较模块由一个预装载寄存器和一个影子寄存器组成。读写过程仅操作预装载寄存器。

在捕获模式下，捕获发生在影子寄存器上，然后再复制到预装载寄存器中。

在比较模式下，预装载寄存器的内容被复制到影子寄存器中，然后影子寄存器的内容和计数器进行比较。

10.4.3.6 输入捕获模式

在输入捕获模式下，当检测到 ICx 信号上相应的边沿后，计数器的当前值被锁存到捕获/比较寄存器（TMRx_CCx）中。当发生捕获事件时，相应的 CxIF 标志（TMRx_STS 寄存器）被置 1，如果开放了中断或者 DMA 操作，则将产生中断或者 DMA 请求。如果发生捕获事件时 CxIF 标志已经为高，那么重复捕获标志 CxOF（TMRx_STS 寄存器）被置 1。写 CxIF=0 可清除 CxIF，或读取存储在 TMRx_CCx 寄存器中的捕获数据也可清除 CxIF。写 CxOF=0 可清除 CxOF。

以下例子说明如何在 TI1 输入的上升沿时捕获计数器的值到 TMRx_CC1 寄存器中，步骤如下：

- 选择有效输入端：TMRx_CC1必须连接到TI1输入，所以写入TMRx_CCM1寄存器中的C1SEL=01，只要C1SEL不为'00'，通道被配置为输入，并且TMRx_CC1寄存器变为只读。
- 根据输入信号的特点，配置输入滤波器为所需的带宽（即输入为TIx时，输入滤波器控制位是 TMRx_CCMx寄存器中的ICxDF位）。假设输入信号在最多5个内部时钟周期的时间内抖动，我们须配置滤波器的带宽长于5个时钟周期；因此我们可以（以f_{CK_INK}频率）连续采样8次，以确认在TI1上一次真实的边沿变换，即在TMRx_CCM1寄存器中写入IC1DF=0011。
- 选择TI1通道的有效转换边沿，在TMRx_CCE寄存器中写入C1P=0（上升沿）。
- 配置输入预分频器。在本例中，我们希望捕获发生在每一个有效的电平转换时刻，因此预分频器被禁止（写TMRx_CCM1寄存器的IC1DIV=00）。
- 设置TMRx_CCE寄存器的C1EN=1，允许捕获计数器的值到捕获寄存器中。
- 如果需要，通过设置TMRx_DIE寄存器中的C1IE位允许相关中断请求，通过设置TMRx_DIE寄存器中的C1DE位允许DMA请求。

当发生一个输入捕获时：

- 产生有效的电平转换时，计数器的值被传送到TMRx_CC1寄存器。
- C1IF标志被设置（中断标志）。当发生至少2个连续的捕获时，而C1IF未曾被清除，C1OF也被置1。
- 如设置了C1IE位，则会产生一个中断。
- 如设置了C1DE位，则还会产生一个DMA请求。

为了处理重复捕获，建议在读出重复捕获标志之前读取数据，这是为了避免丢失在读出重复捕获标志之后和读取数据之前可能产生的重复捕获信息。

注意： *设置 TMRx_EVEG 寄存器中相应的 CxG 位，可以通过软件产生输入捕获中断和/或 DMA 请求。*

10.4.3.7 PWM输入模式

该模式是输入捕获模式的一个特例，除下列区别外，操作与输入捕获模式相同：

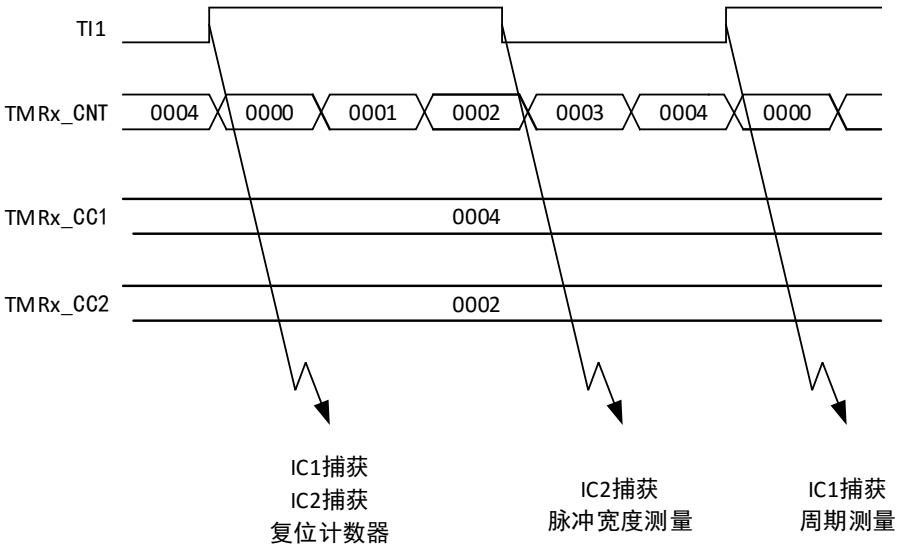
- 两个ICx信号被映射至同一个TIx输入。
- 这2个ICx信号为边沿有效，但是极性相反。
- 其中一个TIxFP信号被作为触发输入信号，而从模式控制器被配置成复位模式。

例如，你需要测量输入到 TI1 上的 PWM 信号的长度（TMRx_CC1 寄存器）和占空比（TMRx_CC2 寄存器），具体步骤如下（取决于 CK_INT 的频率和预分频器的值）

- 选择TMRx_CC1的有效输入：置TMRx_CCM1寄存器的C1SEL=01（选中TI1）。
- 选择TI1FP1的有效极性（用来捕获数据到TMRx_CC1中和清除计数器）：置C1P=0（上升沿有效）。
- 选择TMRx_CC2的有效输入：置TMRx_CCM1寄存器的C2SEL=10（选中TI1）。
- 选择TI1FP2的有效极性（捕获数据到TMRx_CC2）：置C2P=1（下降沿有效）。
- 选择有效的触发输入信号：置TMRx_SMC寄存器中的TRGSEL=101（选择TI1FP1）。

- 配置从模式控制器为复位模式：置TMRx_SMC中的SMSEL=100。
- 使能捕获：置TMRx_CCE寄存器中C1EN=1且C2EN=1。

图 10-110 PWM输入模式时序



注意： 因为只有 TI1FP1 和 TI2FP2 连到了从模式控制器，所以 PWM 输入模式只能使用 TMRx_CH1/TMRx_CH2 信号。

10.4.3.8 强置输出模式

在输出模式（TMRx_CCMx 寄存器中 CxSEL=00）下，输出比较信号（OCxREF 和相应的 OCx/OCxN）能够直接由软件强置为有效或无效状态，而不依赖于输出比较寄存器和计数器间的比较结果。

置 TMRx_CCMx 寄存器中相应的 OCxMODE=101，即可强置输出比较信号（OCxREF/OCx）为有效状态。这样 OCxREF 被强置为高电平（OCxREF 始终为高电平有效），同时 OCx 得到 CxP 极性相反的信号。

例如：CxP=0（OCx 高电平有效），则 OCx 被强置为高电平。

置 TMRx_CCMx 寄存器中的 OCxMODE=100，可强置 OCxREF 信号为低。

该模式下，在 TMRx_CCx 影子寄存器和计数器之间的比较仍然在进行，相应的标志也会被修改。因此仍然会产生相应的中断和 DMA 请求。这将会在下面的输出比较模式一节中介绍。

10.4.3.9 输出比较模式

此项功能是用来控制一个输出波形，或者指示一段给定的的时间已经到时。

当计数器与捕获/比较寄存器的内容相同时，输出比较功能做如下操作：

- 将输出比较模式（TMRx_CCMx寄存器中的OCxMODE位）和输出极性（TMRx_CCE寄存器中的CxP位）定义的值输出到对应的引脚上。在比较匹配时，输出引脚可以保持它的电平（OCxMODE=000）、被设置成有效电平（OCxMODE=001）、被设置成无效电平（OCxMODE=010）或进行翻转（OCxMODE=011）。
- 设置中断状态寄存器中的标志位（TMRx_STS寄存器中的CxIF位）。
- 若设置了相应的中断屏蔽（TMRx_DIE寄存器中的CxIE位），则产生一个中断。
- 若设置了相应的使能位（TMRx_DIE寄存器中的CxDE位，TMRx_CTRL2寄存器中的CDSEL位选择DMA请求功能），则产生一个DMA请求。

TMRx_CCMx 中的 OCxPEN 位选择 TMRx_CCx 寄存器是否需要使用预装载寄存器。

在输出比较模式下，更新事件 UEV 对 OCxREF 和 OCx 输出没有影响。

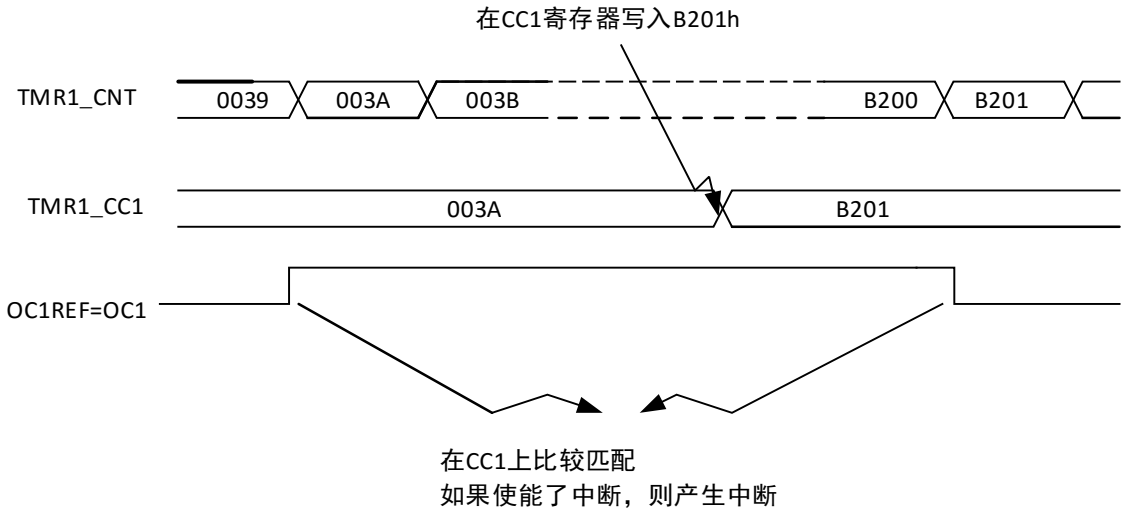
同步的精度可以达到计数器的一个计数周期。输出比较模式（在单脉冲模式下）也能用来输出一个单脉冲。

输出比较模式的配置步骤：

1. 选择计数器时钟（内部，外部，预分频器）。
2. 将相应的数据写入 TMRx_AR 和 TMRx_CCx 寄存器中。
3. 如果要产生一个中断请求，设置 CCxIE 位。
4. 选择输出模式，例如：
   - 要求计数器与 CCx 匹配时翻转 OCx 的输出引脚，设置 OCxMODE=011
   - 置 OCxPEN = 0 禁用预装载寄存器
   - 置 CxP = 0 选择极性为高电平有效
   - 置 CxEN = 1 使能输出
5. 设置 TMRx_CTRL1 寄存器的 CNTEN 位启动计数器

TMRx_CCx 寄存器能够在任何时候通过软件进行更新以控制输出波形，条件是未使用预装载寄存器（OCxPEN='0'，否则 TMRx_CCx 的影子寄存器只能在发生下一次更新事件时被更新）。下图给出了一个例子。

图 10-111 输出比较模式，翻转OC1



10.4.3.10 PWM模式

脉冲宽度调制模式可以产生一个由 TMRx_AR 寄存器确定频率、由 TMRx_CCx 寄存器确定占空比的信号。

在 TMRx_CCMx 寄存器中的 OCxMODE 位写入'110'（PWM 模式 1）或'111'（PWM 模式 2），能够独立地设置每个 OCx 输出通道产生一路 PWM。必须通过设置 TMRx_CCMx 寄存器的 OCxPEN 位使能相应的预装载寄存器，最后还要设置 TMRx_CTRL1 寄存器的 ARPEN 位，（在向上计数或中心对称模式中）使能自动重装载的预装载寄存器。

仅当发生一个更新事件的时候，预装载寄存器才能被传送到影子寄存器，因此在计数器开始计数之前，必须通过设置 TMRx_EVEG 寄存器中的 UEVG 位来初始化所有的寄存器。

OCx 的极性可以通过软件在 TMRx_CCE 寄存器中的 CxP 位设置，它可以设置为高电平有效或

低电平有效。OCx 的输出使能通过（TMRx_CCE 和 TMRx_BRKDT 寄存器中）CxEN、CxNEN、MOEN、OSIMI 和 OSIMR 位的组合控制。详见 TMRx_CCE 寄存器的描述。

在 PWM 模式（模式 1 或模式 2）下，TMRx_CNT 和 TMRx_CCx 始终在进行比较，（依据计数器的计数方向）以确定是否符合 TMRx_CCx≤TMRx_CNT 或者 TMRx_CNT≤TMRx_CCx。

根据 TMRx_CTRL1 寄存器中 CMSEL 位的状态，定时器能够产生边沿对齐的 PWM 信号或中央对齐的 PWM 信号。

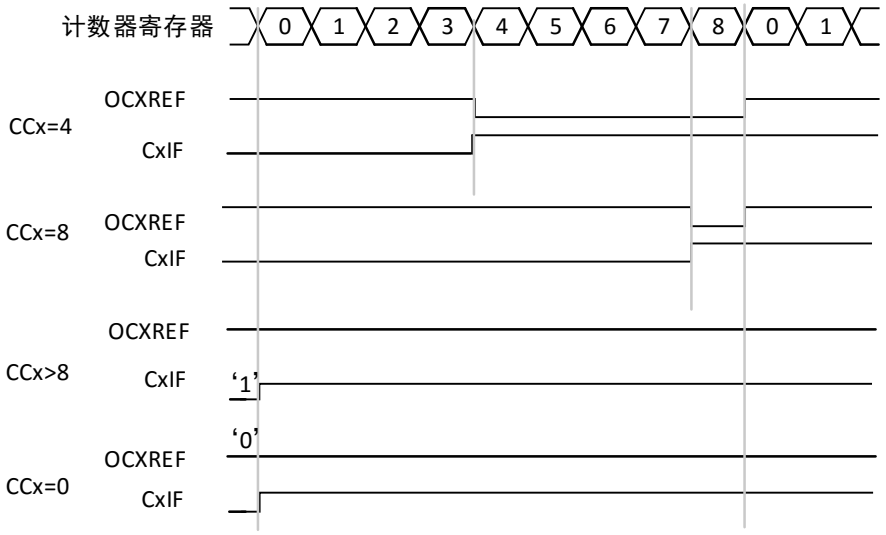
PWM 边沿对齐模式

- 向上计数配置

当 TMRx_CTRL1 寄存器中的 DIR 位为低的时候执行向上计数。参看 10.4.3.2 节。

下面是一个 PWM 模式 1 的例子。当 TMRx_CNT<TMRx_CCx 时，PWM 参考信号 OCxREF 为高，否则为低。如果 TMRx_CCx 中的比较值大于自动重装载值（TMRx_AR），则 OCxREF 保持为'1'。如果比较值为 0，则 OCxREF 保持为'0'。下图为 TMRx_AR=8 时边沿对齐的 PWM 波形实例。

图 10-112 边沿对齐的PWM波形（AR=8）



- 向下计数的配置

  当 TMRx_CTRL1 寄存器的 DIR 位为高时执行向下计数。参看 10.4.3.2 节。

  在 PWM 模式 1，当 TMRx_CNT>TMRx_CCx 时参考信号 OCxREF 为低，否则为高。如果 TMRx_CCx 中的比较值大于 TMRx_AR 中的自动重装载值，则 OCxREF 保持为'1'。该模式下不能产生 0%的 PWM 波形。

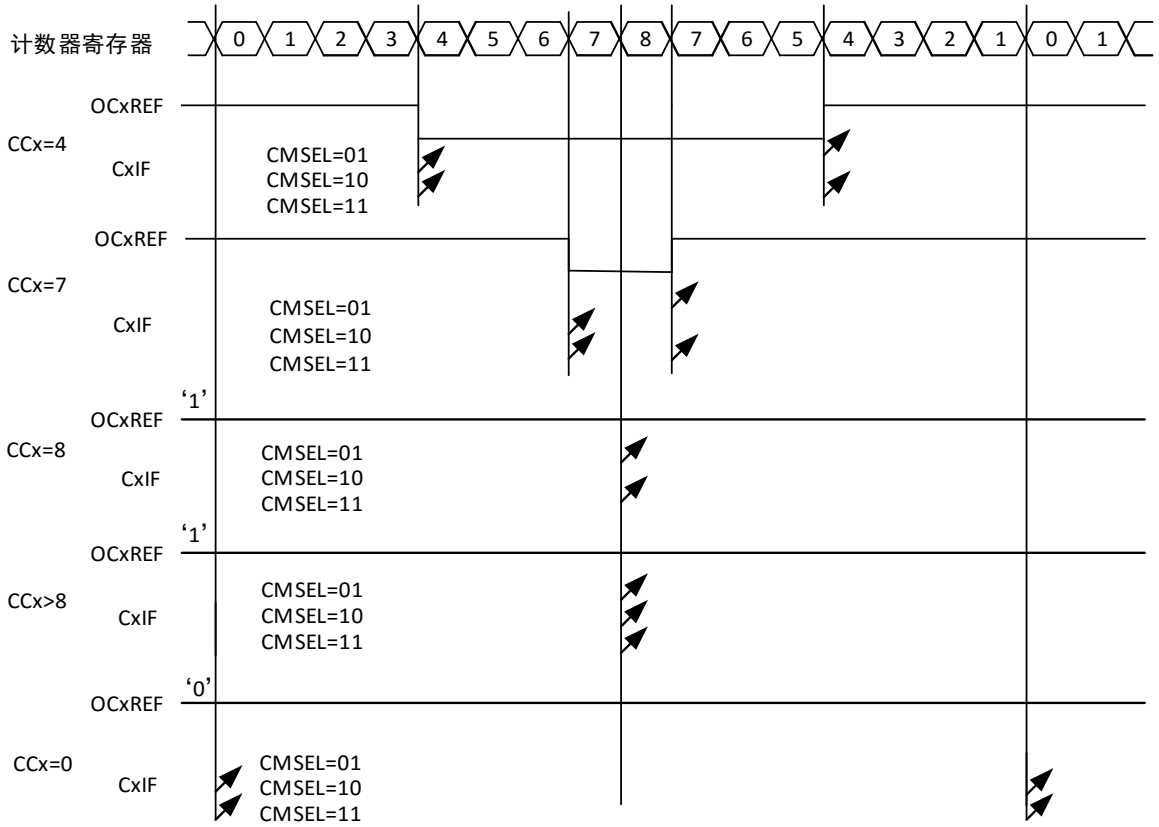
PWM 中央对齐模式

当 TMRx_CTRL1 寄存器中的 CMSEL 位不为'00'时为中央对齐模式（所有其他的配置对 OCxREF/OCx 信号都有相同的作用）。根据不同的 CMSEL 位设置，比较标志可以在计数器向上计数时被置 1、在计数器向下计数时被置 1、或在计数器向上和向下计数时被置 1。TMRx_CTRL1 寄存器中的计数方向位（DIR）由硬件更新，不要用软件修改它。参看 10.4.3.2 节的中央对齐模式。

下图给出了一些中央对齐的 PWM 波形的例子

- TMRx_AR=8
- PWM模式1
- TMRx_CTRL1寄存器的CMSEL=01，在中央对齐模式1下，当计数器向下计数时设置比较标志。

图 10-113 中央对齐的PWM波形（AR=8）



使用中央对齐模式的提示：

- 进入中央对齐模式时，使用当前的向上/向下计数配置：这就意味着计数器向上还是向下计数取决于TMRx_CTRL1寄存器中DIR位的当前值。此外，软件不能同时修改DIR和CMSEL位。
- 不推荐当运行在中央对齐模式时改写计数器，因为这会产生不可预知的结果。特别地：
  - 如果写入计数器的值大于自动重加载的值（TMRx_CNT>TMRx_AR），则方向不会被更新。例如，如果计数器正在向上计数，它就会继续向上计数。
  - 如果将 0 或者 TMRx_AR 的值写入计数器，方向被更新，但不产生更新事件 UEV。
- 使用中央对齐模式最保险的方法，就是在启动计数器之前产生一个软件更新（设置TMRx_EVEG位中的UEVG位），并且不要在计数进行过程中修改计数器的值。

10.4.3.11 互补输出和死区插入

高级控制定时器（TMR1、TMR8）能够输出两路互补信号，并且能够管理输出的瞬时关断和接通。这段时间通常被称为死区，用户应该根据连接的输出器件和它们的特性（电平转换的延时、电源开关的延时等）来调整死区时间。

配置 TMRx_CCE 寄存器中的 CxP 和 CxNP 位，可以为每一个输出独立地选择极性（主输出 OCx 或互补输出 OCxN）。

互补信号 OCx 和 OCxN 通过下列控制位的组合进行控制：TMRx_CCE 寄存器的 CxEN 和 CxNEN 位，TMRx_BRKDT 和 TMRx_CTRL2 寄存器中的 MOEN、OCxIS、OCxNIS、OSIMI 和 OSIMR 位，详见表 10-14 带刹车功能的互补输出通道 OCx 和 OCxN 的控制位。特别的是，在转换到 IDLEF 状态时（MOEN 下降到 0）死区被激活。

同时设置 CxEN 和 CxNEN 位将插入死区，如果存在刹车电路，则还要设置 MOEN 位。每一个通道都有一个 10 位的死区发生器。参考信号 OCxREF 可以产生 2 路输出 OCx 和 OCxN。如果 OCx 和 OCxN 为高有效：

- OCx输出信号与参考信号相同，只是它的上升沿相对于参考信号的上升沿有一个延迟。
- OCxN输出信号与参考信号相反，只是它的上升沿相对于参考信号的下降沿有一个延迟。

如果延迟大于当前有效的输出宽度（OCx 或者 OCxN），则不会产生相应的脉冲。

下列几张图显示了死区发生器的 出信号和当前参考信号 OCxREF 之间的关系。(假设 CxP=0、CxNP=0、MOEN=1、CxEN=1 并且 CxNEN=1）

图 10-114 带死区插入的互补输出

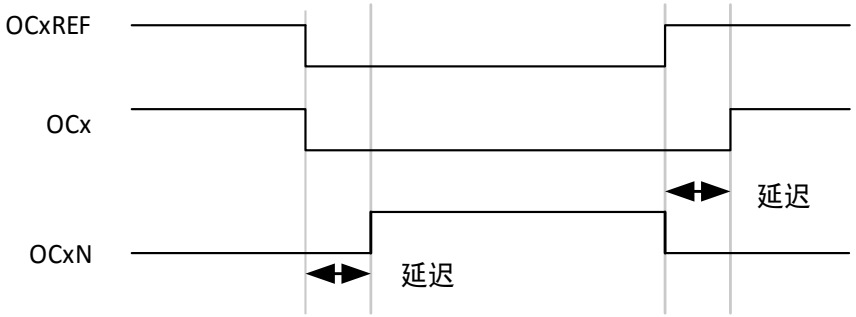


图 10-115 死区波形延迟大于负脉冲

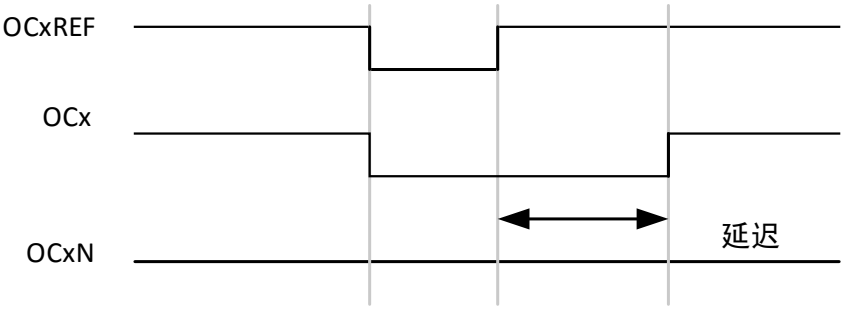
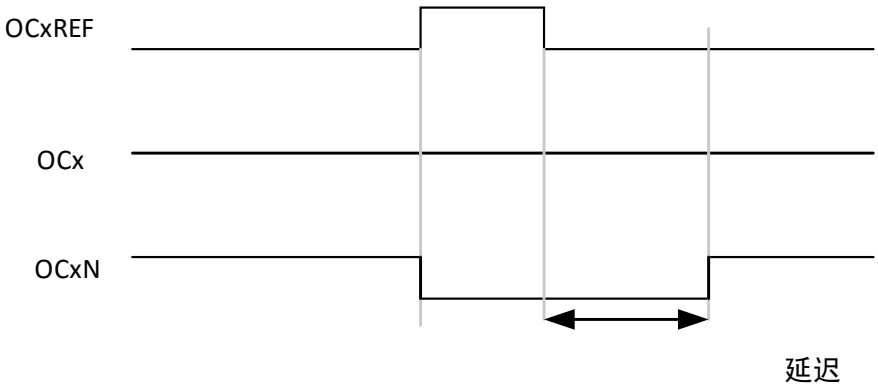


图 10-116 死区波形延迟大于正脉冲



每一个通道的死区延时都是相同的，是由 TMRx_BRKDT 寄存器中的 DTGS 位编程配置。详见 10.4.4.18TMR1、TMR8 刹车和死区寄存器（TMRx_BRKDT）中的延时计算。

重定向 OCxREF 到 OCx 或 OCxN

在输出模式下(强置、输出比较或 PWM)，通过配置 TMRx_CCE 寄存器的 CxEN 和 CxNEN 位，OCxREF 可以被重定向到 OCx 或者 OCxN 的输出。

这个功能可以在互补输出处于无效电平时，在某个输出上送出一个特殊的波形（例如 PWM 或者静态有效电平）。另一个作用是，让两个输出同时处于无效电平，或处于有效电平和带死区的互补输出。

注意： 当只使能 OCxN（CxEN=0，CxNEN=1）时，它不会反相，当 OCxREF 有效时立即变高。例如，如果 CxNP=0，则 OCxN=OCxREF。另一方面，当 OCx 和 OCxN 都被使能时（CxEN=CxNEN=1），当 OCxREF 为高时 OCx 有效；而 OCxN 相反，当 OCxREF 低时 OCxN 变为有效。

10.4.3.12 使用刹车功能

当使用刹车功能时，依据相应的控制位（TMRx_BRKDT 寄存器中的 MOEN、OSIMI 和 OSIMR 位，TMRx_CTRL2 寄存器中的 OCxIS 和 OCxNIS 位），输出使能信号和无效电平都会被修改。但无论何时，OCx 和 OCxN 输出不能在同一时间同时处于有效电平上。详见表 10-14 带刹车功能的互补输出通道 OCx 和 OCxN 的控制位。

刹车源既可以是刹车输入引脚又可以是一个时钟失败事件。时钟失败事件由复位时钟控制器中的时钟安全系统产生，详见 3.2.7 时钟失效检测（CFD）。

系统复位后，刹车电路被禁止，MOEN 位为低。设置 TMRx_BRKDT 寄存器中的 BRKEN 位可以使能刹车功能，刹车输入信号的极性可以通过配置同一个寄存器中的 BRKP 位选择。BRKEN 和 BRKP 可以同时被修改。当写入 BRKEN 和 BRKP 位时，在真正写入之前会有 1 个 APB 时钟周期的延迟，因此需要等待一个 APB 时钟周期之后，才能正确地读回写入的位。

因为 MOEN 下降沿可以是异步的，在实际信号（作用在输出端）和同步控制位（在 TMRx_BRKDT 寄存器中）之间设置了一个再同步电路。这个再同步电路会在异步信号和同步信号之间产生延迟。特别的，如果当它为低时写 MOEN=1，则读出它之前必须先插入一个延时（空指令）才能读到正确的值。这是因为写入的是异步信号而读的是同步信号。

当发生刹车时（在刹车输入端出现选定的电平），有下述动作：

- MOEN位被异步地清除，将输出置于无效状态、空闲状态或者复位状态（由OSIMI位选择）。这个特性在MCU的振荡器关闭时依然有效。
- 一旦MOEN=0，每一个输出通道输出由TMRx_CTRL2寄存器中的OCxIS位设定的电平。如果OSIMI=0，则定时器释放使能输出，否则使能输出始终为高。
- 当使用互补输出时：
  - 输出首先被置于复位状态即无效的状态（取决于极性）。这是异步操作，即使定时器没有时钟时，此功能也有效。
  - 如果定时器的时钟依然存在，死区生成器将会重新生效，在死区之后根据 OCxIS 和 OCxNIS 位指示的电平驱动输出端口。即使在这种情况下，OCx 和 OCxN 也不能被同时驱动到有效的电平。

    注意，因为重新同步 MOEN，死区时间比通常情况下长一些（大约 2 个 ck_tim 的时钟周期）。
  - 如果 OSIMI=0，定时器释放使能输出，否则保持使能输出；或一旦 CxEN 与 CxNEN 之一变高时，使能输出变为高。
- 如果设置了TMRx_DIE寄存器中的BRKIE位，当刹车状态标志（TMRx_STS寄存器中的BRKIF位）为'1'时，则产生一个中断。如果设置了TMRx_DIE寄存器中的BDE位，则产生一个DMA请求。
- 如果设置了TMRx_BRKDT寄存器中的AOEN位，在下一个更新事件UEV时MOEN位被自动置位；例如，这可以用来进行整形。否则，MOEN始终保持低直到被再次置'1'；此时，这个特性可以被用在安全方面，你可以把刹车输入连到电源驱动的报警输出、热敏传感器或者其他安全器件上。

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

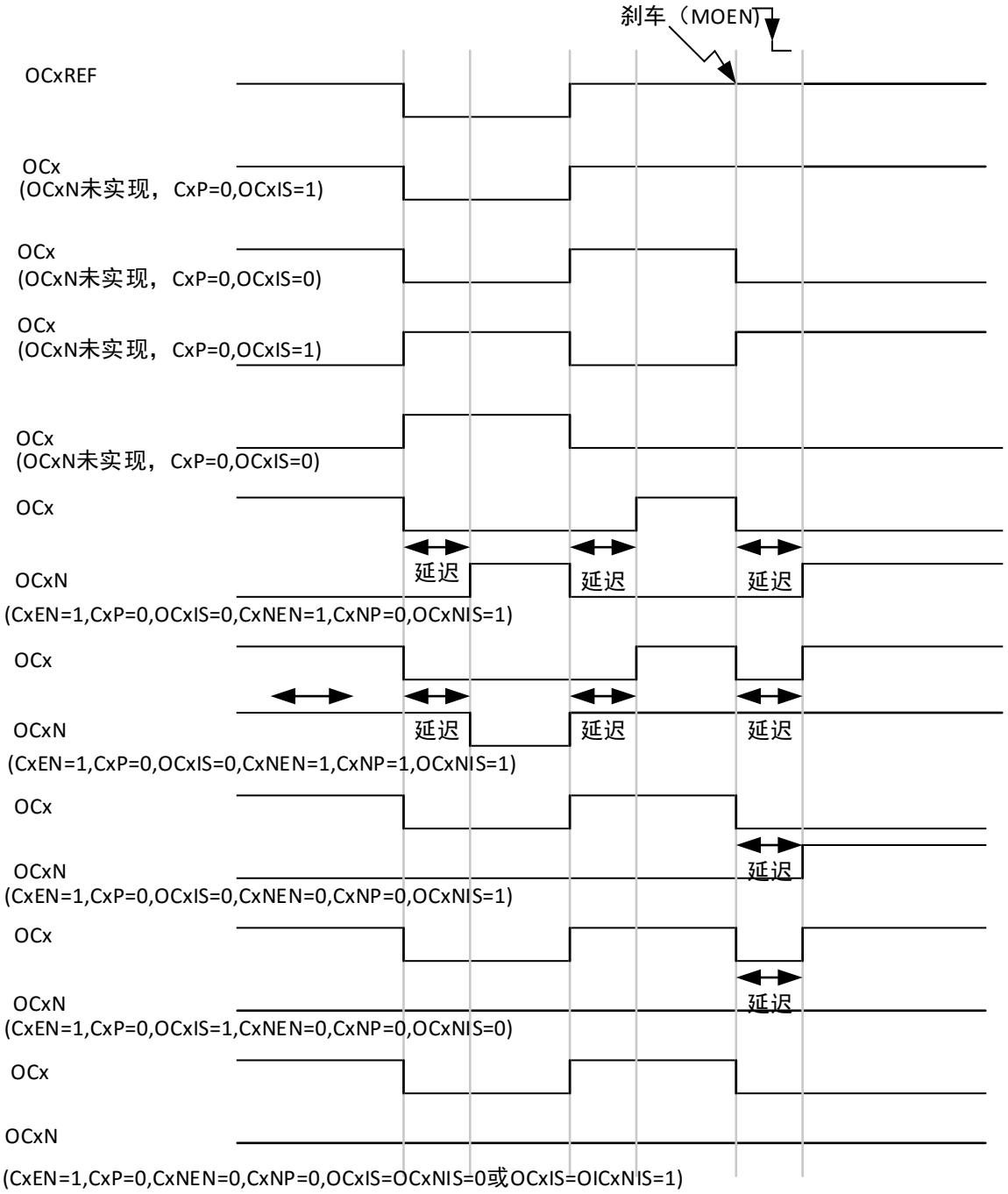
刹车由 BRK 输入产生，它的有效极性是可编程的，且由 TMRx_BRKDT 寄存器中的 BRKEN 位开启。

除了刹车输入和输出管理，刹车电路中还实现了写保护以保证应用程序的安全。它允许用户冻结几个配置参数（死区长度，OCx/OCxN 极性和被禁止的状态，OCxMODE 配置，刹车使能和极性）。

用户可以通过 TMRx_BRKDT 寄存器中的 LOCKC 位，从三级保护中选择一种，参看 10.4.4.18 节 TMR1、TMR8 刹车和死区寄存器（TMRx_BRKDT）。在 MCU 复位后 LOCKC 位只能被修改一次。

下图显示响应刹车的输出实例。

图 10-117 响应刹车的输出



10.4.3.13 在外部事件时清除OCxREF信号

对于一个给定的通道，设置 TMRx_CCMx 寄存器中对应的 OCxDIS 位为'1'，能够用 ETRF 输入端的高电平把 OCxREF 信号拉低，OCxREF 信号将保持为低直到发生下一次的更新事件 UEV。

该功能只能用于输出比较和 PWM 模式，而不能用于强置模式。

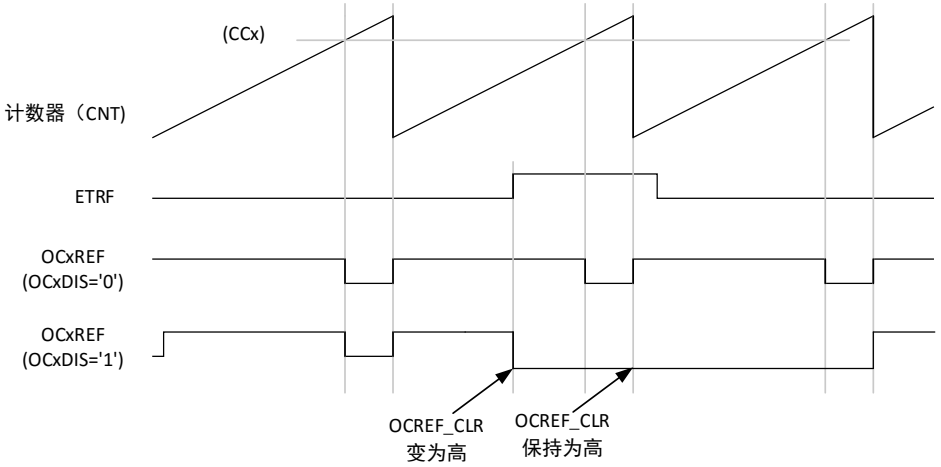
例如，OCxREF 信号可以联到一个比较器的输出，用于控制电流。这时，ETR 必须配置如下：

1. 外部触发预分频器必须处于关闭：TMRx_SMC 寄存器中的 ETD[1：0]=00。
2. 必须禁止外部时钟模式 2：TMRx_SMC 寄存器中的 ECLKEN=0。

3. 外部触发极性（ETRGP）和外部触发滤波器（ETDF）可以根据需要配置。

下图显示了当 ETRF 输入变为高时，对应不同 OCxDIS 的值，OCxREF 信号的动作。在这个例子中，定时器 TMRx 被置于 PWM 模式。

图 10-118 清除TMRx的OCxREF



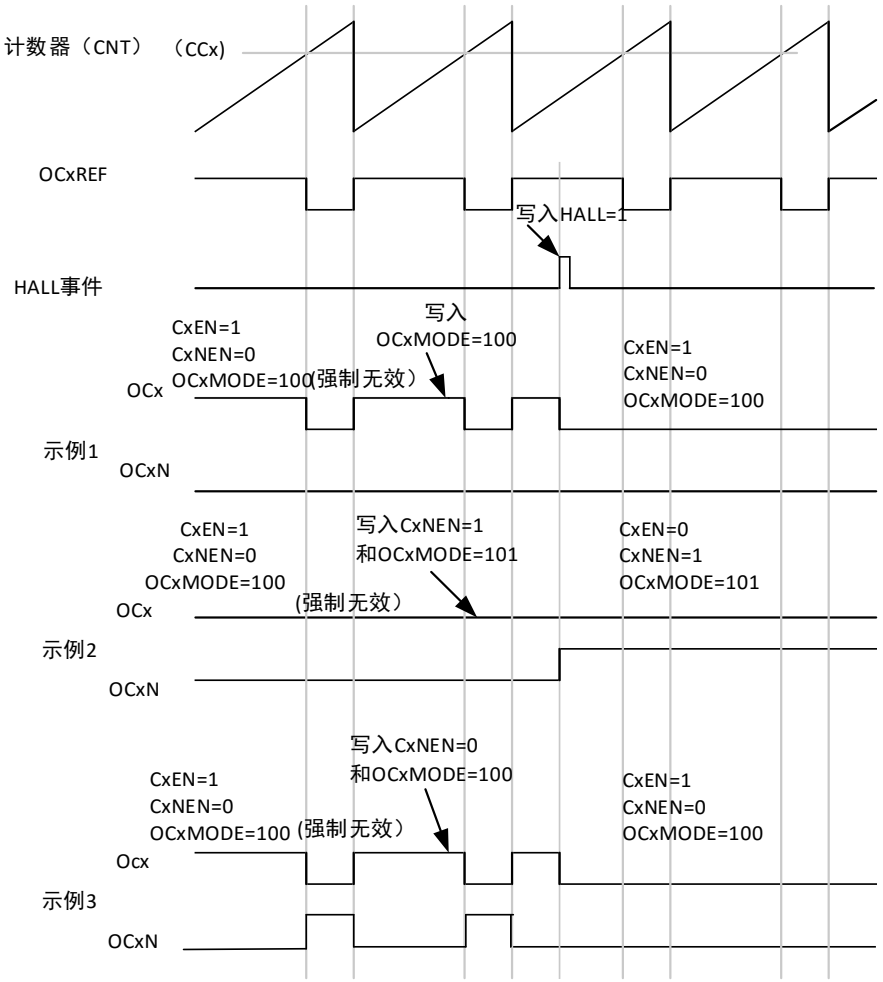
10.4.3.14 产生六步PWM输出

当在一个通道上需要互补输出时，预装载位有 OCxMODE、CxEN 和 CxNEN。在发生 HALL 换相事件时，这些预装载位被传送到影子寄存器位。这样你就可以预先设置好下一步骤配置，并在同一个时刻同时修更改所有通道的配置。HALL 可以通过设置 TMRx_EVEG 寄存器的 HALL 位由软件产生，或在 TRGI 上升沿由硬件产生。

当发生 HALL 事件时会设置一个标志位（TMRx_STS 寄存器中的 HALLIF 位），这时如果已设置了 TMRx_DIE 寄存器的 HALLIE 位，则产生一个中断；如果已设置了 TMRx_DIE 寄存器的 HALLDE 位，则产生一个 DMA 请求。

下图显示当发生 HALL 事件时，三种不同配置下 OCx 和 OCxN 输出。

图 10-119 产生六步PWM，使用HALL的例子（OSIMR=1）



10.4.3.15 单脉冲模式

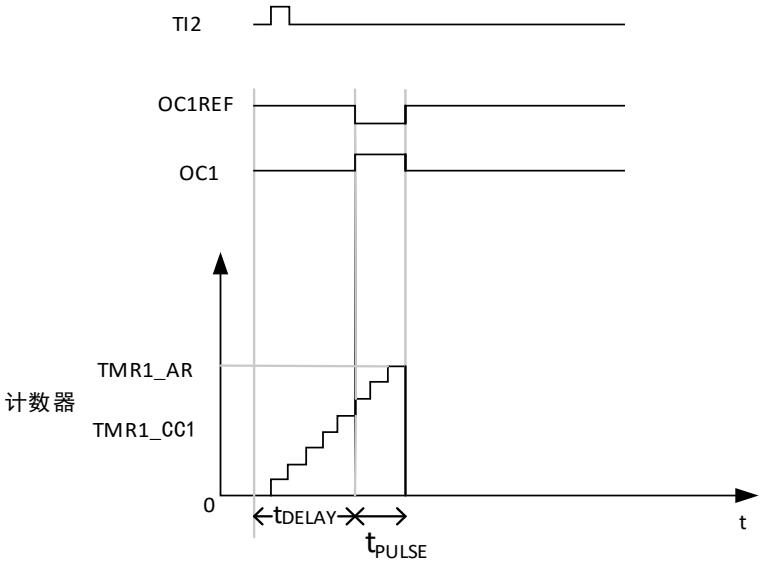
单脉冲模式（OPMODE）是前述众多模式的一个特例。这种模式允许计数器响应一个激励，并在一个程序可控的延时之后产生一个脉宽可程序控制的脉冲。

可以通过从模式控制器启动计数器，在输出比较模式或者 PWM 模式下产生波形。设置 TMRx_CTRL1 寄存器中的 OPMODE 位将选择单脉冲模式，这样可以让计数器自动地在产生下一个更新事件 UEV 时停止。

仅当比较值与计数器的初始值不同时，才能产生一个脉冲。启动之前（当定时器正在等待触发），必须如下配置：

- 向上计数方式：计数器CNT < CCx ≤ AR （特别地，0 < CCx）；
- 向下计数方式：计数器CNT > CCx。

图 10-120 单脉冲模式的例子



例如，你需要在从 TI2 输入脚上检测到一个上升沿开始，延迟 t_{DELAY} 之后，在 OC1 上产生一个长度为 t_{PULSE} 的正脉冲。

假定 TI2FP2 作为触发 1：

- 置TMRx_CCM1寄存器中的C2SEL=01，把TI2FP2映像到TI2。
- 置TMRx_CCE寄存器中的C2P=0，使TI2FP2能够检测上升沿。
- 置TMRx_SMC寄存器中的TRGSEL=110，TI2FP2作为从模式控制器的触发（TRGI）。
- 置TMRx_SMC寄存器中的SMSEL=110（触发模式），TI2FP2被用来启动计数器。

OPMODE 的波形由写入比较寄存器的数值决定（要考虑时钟频率和计数器预分频器）

- t_{DELAY}由TMRx_CC1寄存器中的值定义。
- t_{PULSE}由自动装载值和比较值之间的差值定义（TMRx_AR - TMRx_CC1）。
- 假定当发生比较匹配时要产生从0到1的波形，当计数器达到预装载值时要产生一个从1到0的波形；首先要置TMRx_CCM1寄存器的OC1MODE=111，进入PWM模式2；根据需要有选择地使能预装载寄存器：置TMRx_CCM1中的OC1PEN=1和TMRx_CTRL1寄存器中的ARPEN；然后在TMRx_CC1寄存器中填写比较值，在TMRx_AR寄存器中填写自动装载值，设置UEVG位来产生一个更新事件，然后等待在TI2上的一个外部触发事件。本例中，C1P=0。

在这个例子中，TMRx_CTRL1 寄存器中的 DIR 和 CMSEL 位应该置低。因为只需要一个脉冲，所以必须设置 TMRx_CTRL1 寄存器中的 OPMODE=1，在下一个更新事件（当计数器从自动装载值翻转到 0）时停止计数。

特殊情况：OCx 快速使能：

在单脉冲模式下，在 TIx 输入脚的边沿检测逻辑设置 CNTEN 位以启动计数器。然后计数器和比较值间的比较操作产生了输出的转换。但是这些操作需要一定的时钟周期，因此它限制了可得到的最小延时 t_{DELAY}。

如果要以最小延时输出波形，可以设置 TMRx_CCMx 寄存器中的 OCxFEN 位；此时 OCxREF（和 OCx）直接响应激励而不再依赖比较的结果，输出的波形与比较匹配时的波形一样。OCxFEN 只在通道配置为 PWM1 和 PWM2 模式时起作用。

10.4.3.16 编码器接口模式

选择编码器接口模式的方法是：如果计数器只在 TI1 的边沿计数，则置 TMRx_SMC 寄存器中的 SMSEL=001；如果只在 TI2 边沿计数，则置 SMSEL=010；如果计数器同时在 TI1 和 TI2 边沿计数，则置 SMSEL=011。

通过设置 TMRx_CCE 寄存器中的 C1P 和 C2P 位，可以选择 TI1 和 TI2 极性；如果需要，还可以对输入滤波器编程。

两个输入 TI1 和 TI2 被用来作为增量编码器的接口。参看表 10-11，假定计数器已经启动（TMRx_CTRL1 寄存器中的 CNTEN=1），则计数器由每次在 TI1FP1 或 TI2FP2 上的有效跳变驱动。TI1FP1 和 TI2FP2 是 TI1 和 TI2 在通过输入滤波器和极性控制后的信号；如果没有滤波和变相，则 TI1FP1=TI1，TI2FP2=TI2。根据两个输入信号的跳变顺序，产生了计数脉冲和方向信号。依据两个输入信号的跳变顺序，计数器向上或向下计数，同时硬件对 TMRx_CTRL1 寄存器的 DIR 位进行相应的设置。不管计数器是依靠 TI1 计数、依靠 TI2 计数或者同时依靠 TI1 和 TI2 计数，在任一输入端（TI1 或者 TI2）的跳变都会重新计算 DIR 位。

编码器接口模式基本上相当于使用了一个带有方向选择的外部时钟。这意味着计数器只在 0 到 TMRx_AR 寄存器的自动装载值之间连续计数（根据方向，或是 0 到 AR 计数，或是 AR 到 0 计数）。所以在开始计数之前必须配置 TMRx_AR；同样，捕获器、比较器、预分频器、重复计数器、触发输出特性等仍工作如常。编码器模式和外部时钟模式 2 不兼容，因此不能同时操作。

在这个模式下，计数器依照增量编码器的速度和方向被自动的修改，因此计数器的内容始终指示着编码器的位置。计数方向与相连的传感器旋转的方向对应。下表列出了所有可能的组合，假设 TI1 和 TI2 不同时变换。

表 10-11 计数方向与编码器信号的关系

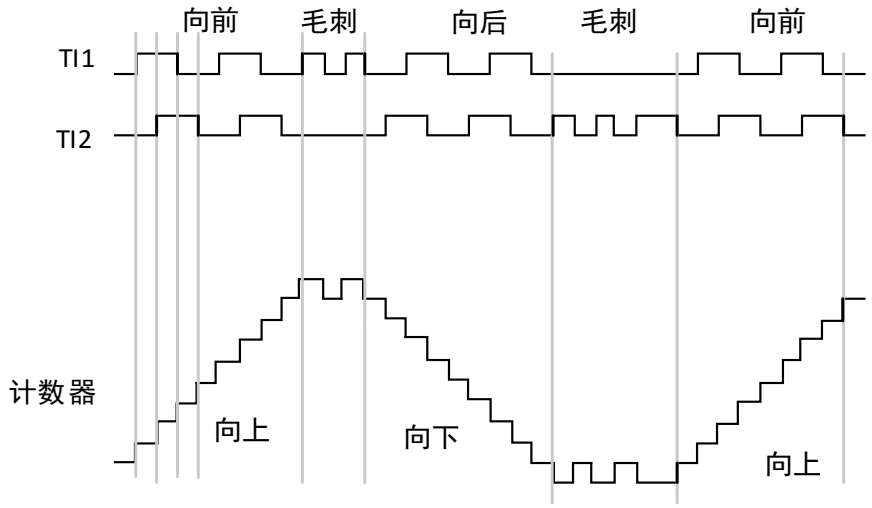
有效边沿	相对信号的电平（TI1FP1 对应 TI2，TI2FP2 对应 TI1）	TI1FP1 信号		TI2FP2 信号	
		上升	下降	上升	下降
仅在 TI1 计数	高	向下计数	向上计数	不计数	不计数
	低	向上计数	向下计数	不计数	不计数
仅在 TI2 计数	高	不计数	不计数	向上计数	向下计数
	低	不计数	不计数	向下计数	向上计数
在 TI1 和 TI2 上计数	高	向下计数	向上计数	向上计数	向下计数
	低	向上计数	向下计数	向下计数	向上计数

一个外部的增量编码器可以直接与 MCU 连接而不需要外部接口逻辑。但是，一般会使用比较器将编码器的差动输出转换到数字信号，这大大增加了抗噪声干扰能力。编码器输出的第三个信号表示机械零点，可以把它连接到一个外部中断输入并触发一个计数器复位。

下图是一个计数器操作的实例，显示了计数信号的产生和方向控制。它还显示了当选择了双边沿时，输入抖动是如何被抑制的；抖动可能会在传感器的位置靠近一个转换点时产生。在这个例子中，我们假定配置如下：

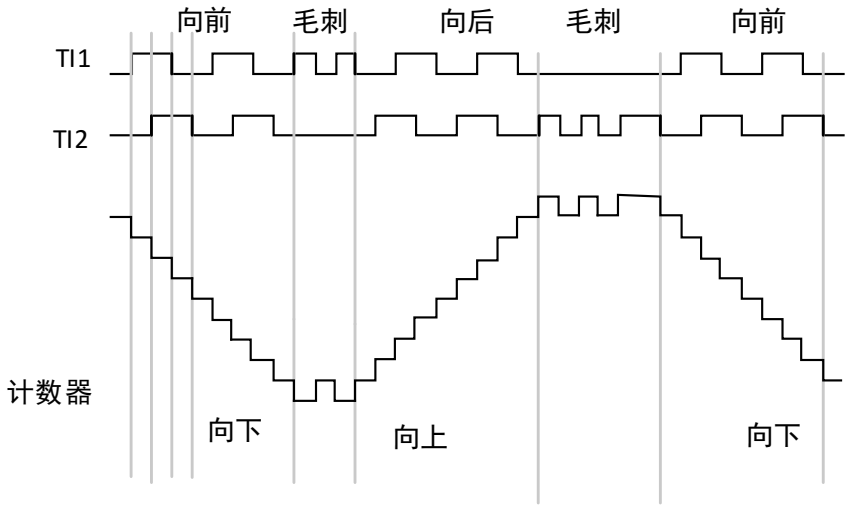
- C1SEL='01' （TMRx_CCM1寄存器，IC1FP1映射到TI1）
- C2SEL='01' （TMRx_CCM1寄存器，IC2FP2映射到TI2）
- C1P='0' （TMRx_CCE寄存器，IC1FP1不反相，IC1FP1=TI1）
- C2P='0' （TMRx_CCE寄存器，IC2FP2不反相，IC2FP2=TI2）
- SMSEL='011' （TMRx_SMC寄存器，所有的输入均在上升沿和下降沿有效）.
- CNTEN='1' （TMRx_CTRL1寄存器，计数器使能）

图 10-121 编码器模式下的计数器操作实例



下图为当 IC1FP1 极性反相时计数器的操作实例（C1P='1'，其他配置与上例相同）。

图 10-122 IC1FP1反相的编码器接口模式实例



当定时器配置成编码器接口模式时，提供传感器当前位置的信息。使用第二个配置在捕获模式的定时器，可以测量两个编码器事件的间隔，获得动态的信息（速度，加速度，减速度）。指示机械零点的编码器输出可被用做此目的。根据两个事件间的间隔，可以按照固定的时间读出计数器。如果可能的话，你可以把计数器的值锁存到第三个输入捕获寄存器（捕获信号必须是周期的并且可以由另一个定时器产生）；也可以通过一个由实时时钟产生的 DMA 请求来读取它的值。

10.4.3.17 定时器输入异或功能

TMRx_CTRL2 寄存器中的 TI1SEL 位，允许通道 1 的输入滤波器连接到一个异或门的输出端，异或门的 3 个输入端为 TMRx_CH1、TMRx_CH2 和 TMRx_CH3。

异或输出能够被用于所有定时器的输入功能，如触发或输入捕获。下一节给出了此特性用于连接霍尔传感器的例子。

10.4.3.18 与霍尔传感器的接口

使用高级控制定时器（TMR1、TMR8）产生 PWM 信号驱动马达时，可以用另一个通用 TMRx（TMR2、TMR3、TMR4 或 TMR5）定时器作为“接口定时器”来连接霍尔传感器，见图 10-123 霍尔传感器接口的实例，3 个定时器输入脚（CC1、CC2、CC3）通过一个异或门连接到 TI1 输入通道（通过设置 TMRx_CTRL2 寄存器中的 TI1SEL 位来选择），“接口定时器”捕获这个信号。

从模式控制器被配置于复位模式，从输入是 TI1F_ED。每当 3 个输入之一变化时，计数器重新从 0 开始

计数。这样产生一个由霍尔输入端的任何变化而触发的时间基准。

“接口定时器”上的捕获/比较通道 1 配置为捕获模式，捕获信号为 TRC（见图 10-106 捕获/比较通道（如：通道 1 输入部分））。捕获值反映了两个输入变化间的时间延迟，给出了马达速度的信息。

“接口定时器”可以用来在输出模式产生一个脉冲，这个脉冲可以（通过触发一个 HALL 事件）用于改变高级定时器 TMR1、TMR8 各个通道的属性，而高级控制定时器产生 PWM 信号驱动马达。

因此“接口定时器”通道必须编程为在一个指定的延时（输出比较或 PWM 模式）之后产生一个正脉冲，这个脉冲通过 TRGO 输出被送到高级控制定时器 TMR1、TMR8。

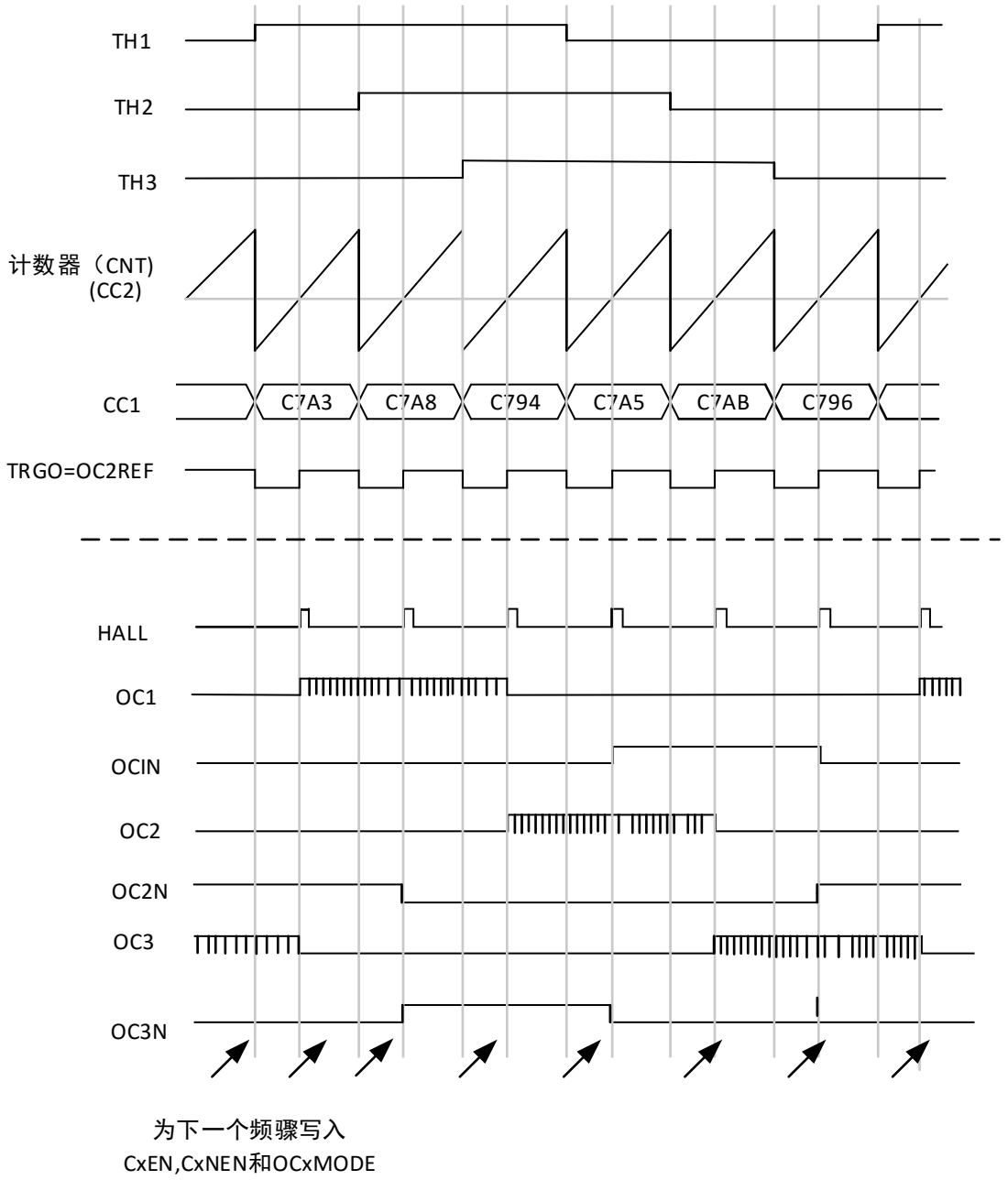
举例：霍尔输入连接到 TMRx 定时器，要求每次任一霍尔输入上发生变化之后的一个指定的时刻，改变高级控制定时器 TMRx 的 PWM 配置。

- 置TMRx_CTRL2寄存器的TI1SEL位为’1’，配置三个定时器输入逻辑或到TI1输入，
- 时基编程：置TMRx_AR为其最大值（计数器必须通过TI1的变化清零）。设置预分频器得到一个最大的计数器周期，它长于传感器上的两次变化的时间间隔。
- 设置通道1为捕获模式（选中TRC）：置TMRx_CCM1寄存器中C1SEL=11，如果需要，还可以设置数字滤波器。
- 设置通道2为PWM2模式，并具有要求的延时：置TMRx_CCM1寄存器中的OC2MODE=111和C2SEL=00。
- 选择OC2REF作为TRGO上的触发输出：置TMRx_CTRL2寄存器中的MMSEL=101。

在高级控制寄存器 TMR1 中，正确的 ITR 输入必须是触发器输入，定时器被编程为产生 PWM 信号，捕获/比较控制信号为预装载的（TMRx_CTRL2 寄存器中 CPC=1），同时触发输入控制 HALL 事件（TMRx_CTRL2 寄存器中 CUSEL=1）。在一次 HALL 事件后，写入下一步的 PWM 控制位（CxEN、OCxMODE），这可以在处理 OC2REF 上升沿的中断子程序里实现。

下图显示了这个实例

图 10-123 霍尔传感器接口的实例



10.4.3.19 TMRx定时器和外部触发的同步

TMRx 定时器能够在多种模式下和一个外部的触发同步：复位模式、门控模式和触发模式。

从模式：复位模式

在发生一个触发输入事件时，计数器和它的预分频器能够重新被初始化。同时，如果 TMRx_CTRL1 寄存器的 UVERS 位为低，还产生一个更新事件 UEV 。然后所有的预装载寄存器（TMRx_AR ，TMRx_CCx）都被更新了。

在以下的例子中，TI1 输入端的上升沿导致向上计数器被清零：

- 配置通道1以检测TI1的上升沿。配置输入滤波器的带宽（在本例中，不需要任何滤波器，因此保持IC1DF=0000）。触发操作中不使用捕获预分频器，所以不需要配置。C1SEL位只选择输入捕获源，即TMRx_CCM1寄存器中C1SEL=01。置TMRx_CCE寄存器中C1P=0以确定极性（只检测上升沿）。
- 置TMRx_SMC寄存器中SMSEL=100，配置定时器为复位模式；置TMRx_SMC寄存器

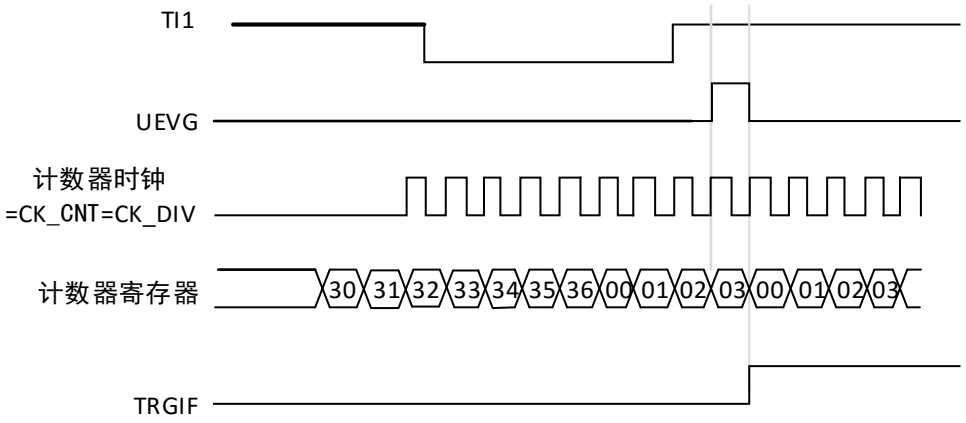
中TRGSEL=101，选择TI1作为输入源。

- 置TMRx_CTRL1寄存器中CNTEN=1，启动计数器。

计数器开始依据内部时钟计数，然后正常运转直到 TI1 出现一个上升沿；此时，计数器被清零然后从 0 重新开始计数。同时，触发标志（TMRx_STS 寄存器中的 TRGIF 位）被设置，根据 TMRx_DIE 寄存器中 TRGIE（中断使能）位和 TRGDE（DMA 使能）位的设置，产生一个中断请求或一个 DMA 请求。

下图显示当自动重装载寄存器 TMRx_AR=0x36 时的动作。在 TI1 上升沿和计数器的实际复位之间的延时取决于 TI1 输入端的重同步电路。

图 10-124 复位模式下的控制电路



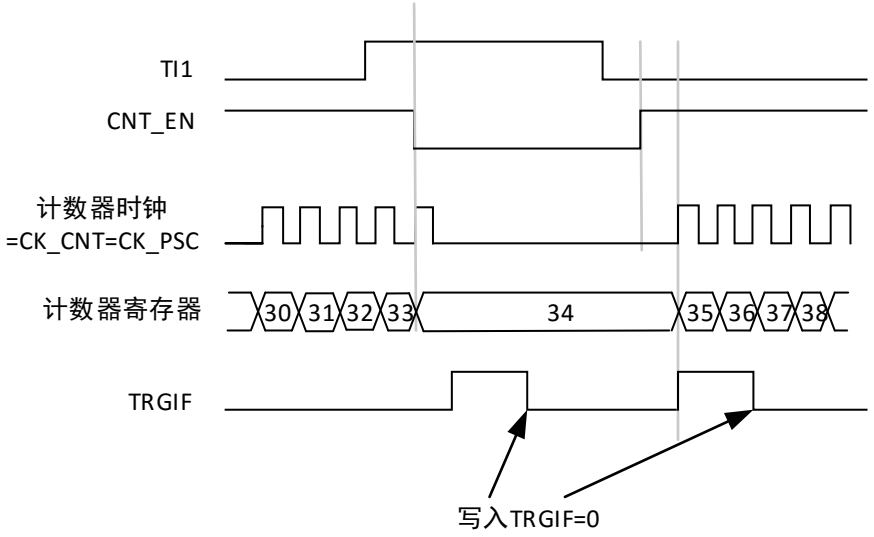
从模式：门控模式

按照选中的输入端电平使能计数器。在如下的例子中，计数器只在 TI1 为低时向上计数：

- 配置通道1以检测TI1上的低电平。配置输入滤波器带宽（本例中，不需要滤波，所以保持IC1DF=0000）。触发操作中不使用捕获预分频器，所以不需要配置。C1SEL位用于选择输入捕获源，置TMRx_CCM1寄存器中C1SEL=01。置TMRx_CCE寄存器中C1P=1以确定极性（只检测低电平）。
- 置TMRx_SMC寄存器中SMSEL=101，配置定时器为门控模式；置TMRx_SMC寄存器中TRGSEL=101，选择TI1作为输入源。
- 置TMRx_CTRL1寄存器中CNTEN=1，启动计数器。在门控模式下，如果CNTEN=0，则计数器不能启动，不论触发输入电平如何。只要TI1为低，计数器开始依据内部时钟计数，一旦TI1变高则停止计数。当计数器开始或停止时都设置TMRx_STS中的TRGIF标置。

TI1 上升沿和计数器实际停止之间的延时取决于 TI1 输入端的重同步电路。

图 10-125 门控模式下的控制电路



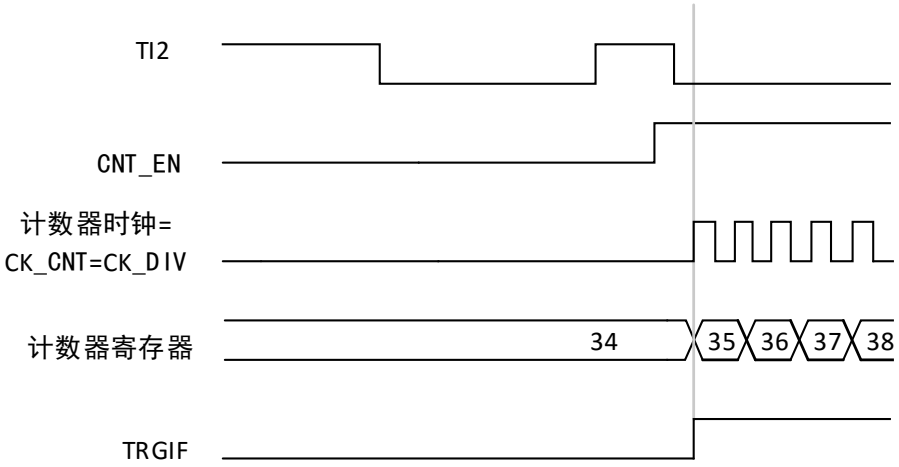
从模式：触发模式

输入端上选中的事件使能计数器。在下面的例子中，计数器在 TI2 输入的上升沿开始向上计数：

- 配置通道2检测TI2的上升沿。配置输入滤波器带宽（本例中，不需要任何滤波器，保持IC2DF=0000）。触发操作中不使用捕获预分频器，不需要配置。C2SEL位只用于选择输入捕获源，置TMRx_CCM1寄存器中C2SEL=01。置TMRx_CCE寄存器中C2P=0以确定极性（只检测低电平）。
- 置TMRx_SMC寄存器中SMSEL=110，配置定时器为触发模式；置TMRx_SMC寄存器中TRGSEL=110，选择TI2作为输入源。

当 TI2 出现一个上升沿时，计数器开始在内部时钟驱动下计数，同时设置 TRGIF 标志。TI2 上升沿和计数器启动计数之间的延时，取决于 TI2 输入端的重同步电路。

图 10-126 触发器模式下的控制电路



从模式：外部时钟模式 2 ＋ 触发模式

外部时钟模式 2 可以与另一种从模式（外部时钟模式 1 和编码器模式除外）一起使用。这时，ETR 信 号被用作外部时钟的输入，在复位模式、门控模式或触发模式可以选择另一个输入作为触发输入。不建议使用 TMRx_SMC 寄存器的 TRGSEL 位选择 ETR 作为 TRGI。

在下面的例子中，一旦在 TI1 上出现一个上升沿，计数器即在 ETR 的每一个上升沿向上计数一次：

1. 通过 TMRx_SMC 寄存器配置外部触发输入电路：
   - ETDF=0000：没有滤波
   - ETD=00：不用预分频器

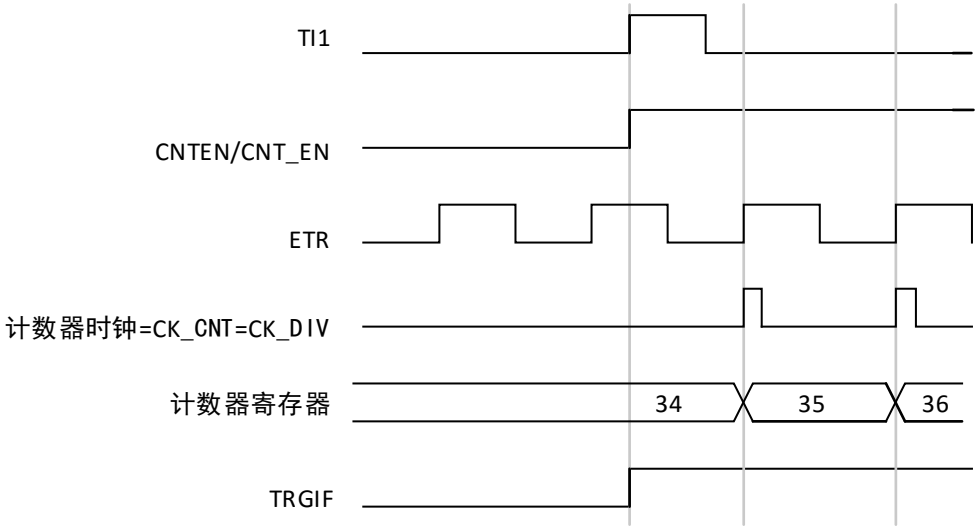
— ETRGP=0：检测 ETR 的上升沿，置 ECLKEN=1 使能外部时钟模式 2。

2. 按如下配置通道 1，检测 TI 的上升沿：
   — IC1DF=0000：没有滤波
   — 触发操作中不使用捕获预分频器，不需要配置
   — 置 TMRx_CCM1 寄存器中 C1SEL=01，选择输入捕获源
   — 置 TMRx_CCE 寄存器中 C1P=0 以确定极性（只检测上升沿）
3. 置 TMRx_SMC 寄存器中 SMSEL=110，配置定时器为触发模式。置 TMRx_SMC 寄存器中 TRGSEL=101，选择 TI1 作为输入源。

当 TI1 上出现一个上升沿时，TRGIF 标志被设置，计数器开始在 ETR 的上升沿计数。

ETR 信号的上升沿和计数器实际复位间的延时，取决于 ETRP 输入端的重同步电路。

图 10-127 外部时钟模式2＋触发模式下的控制电路



10.4.3.20 定时器同步

所有 TMR 定时器在内部相连，用于定时器同步或链接。详见 10.2.3.5 节。

10.4.3.21 调试模式

当微控制器进入调试模式时（Cortex™-M4F 核心停止），根据 DBG 模块中 DBG_TMRx_STOP 的设置，TMRx 计数器可以或者继续正常操作，或者停止。详见 22.2.2 节。

10.4.4 TMR1、TMR8寄存器描述

可以用半字（16 位）或字（32 位）的方式操作这些外设寄存器。

下表中将 TMR1、TMR8 的所有寄存器映射到一个 16 位可寻址（编址）空间

表 10-12 TMR1、TMR8 – 寄存器图和复位值

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	TMRx_CTRL1	保留																						CLKDIV[1：0]		ARPEN	CMSEL[1：0]		DIR	OPMODE	UVERS	UEVDIS	CNTEN
	复位值																							0	0	0	0	0	0	0	0	0	0

| | | |

0x28	TMRx_DIV	保留	DIV[15：0]														
	复位值		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x2C	TMRx_AR	保留	AR[15：0]														
	复位值		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x30	TMRx_RC	保留													RC[7：0]		
	复位值														0	0	0	0	0	0	0	0
0x34	TMRx_CC1	保留	CC1[15：0]														
	复位值		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x38	TMRx_CC2	保留	CC2[15：0]														
	复位值		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x3C	TMRx_CC3	保留	CC3[15：0]														
	复位值		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x40	TMRx_CC4	保留	CC4[15：0]														
	复位值		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x44	TMRx_ BRKDT	保留	MOEN	AOEN	BRKP	BRKEN	OSIMR	OSIMI	LOCKC[1：0]		DTGS[7：0]						
	复位值		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x48	TMRx_ DMAC	保留						DBLEN[4：0]				保留	ADDR[4：0]				
	复位值							0	0	0	0	0	0	0	0	0	0	0
0x4C	TMRx_ DMABA	保留	DMABA[15：0]														
	复位值		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

10.4.4.1 TMR1、TMR8 控制寄存器1（TMRx_CTRL1）

偏移地址：0x00

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留						CLKDIV[1：0]		ARP EN	CMSEL[1：0]		DIR	OPMO DE	UVE RS	UEV DIS	CNT EN
res						rw	rw	rw	rw		rw	rw	rw	rw	rw

位 15：10	保留，始终读为 0。
位 9：8	**CLKDIV[1：0]：**时钟分频因子 （Clock division） 这 2 位定义在定时器时钟（CK_INT）频率、死区时间和由死区发生器与数字滤波器（ETR，TIx）所用的采样时钟之间的分频比例。 00：t_{DTS} = t_{CK_IT} 01：t_{DTS} = 2 x t_{CK_INT} 10：t_{DTS} = 4 x t_{CK_INT} 11：保留，不要使用这个配置
位 7	**ARPEN：**自动重装载预装载允许位 （Auto-reload preload enable） 0：TMRx_AR 寄存器没有缓冲； 1：TMRx_AR 寄存器被装入缓冲器。

位 6：5	**CMSEL[1：0]**：选择中央对齐模式 （Center-aligned mode selection） 00：边沿对齐模式。计数器依据方向位（DIR）向上或向下计数。 01：中央对齐模式 1。计数器交替地向上和向下计数。配置为输出的通道（TMRx_CCMx 寄存器中 CxSEL=00）的输出比较中断标志位，只在计数器向下计数时被设置。 10：中央对齐模式 2。计数器交替地向上和向下计数。配置为输出的通道（TMRx_CCMx 寄存器中 CxSEL=00）的输出比较中断标志位，只在计数器向上计数时被设置。 11：中央对齐模式 3。计数器交替地向上和向下计数。配置为输出的通道（TMRx_CCMx 寄存器中 CxSEL=00）的输出比较中断标志位，在计数器向上和向下计数时均被设置。 *注：在计数器开启时（CNTEN=1），不允许从边沿对齐模式转换到中央对齐模式。*
位 4	**DIR**：方向（Direction） 0：计数器向上计数； 1：计数器向下计数。 *注：当计数器配置为中央对齐模式或编码器模式时，该位为只读。*
位 3	**OPMODE**：单脉冲模式 （One pulse mode） 0：在发生更新事件时，计数器不停止； 1：在发生下一次更新事件（清除 CNTEN 位）时，计数器停止。
位 2	**UVERS**：更新请求源 （Update request source） 软件通过该位选择 UEV 事件的源 0：如果使能了更新中断或 DMA 请求，则下述任一事件产生更新中断或 DMA 请求： – 计数器溢出/下溢 – 设置 UEVG 位 – 从模式控制器产生的更新 1：如果使能了更新中断或 DMA 请求，则只有计数器溢出/下溢才产生更新中断或 DMA 请求。
位 1	**UEVDIS**：禁止更新 （Update disable） 软件通过该位允许/禁止 UEV 事件的产生 0：允许 UEV。更新（UEV）事件由下述任一事件产生： – 计数器溢出/下溢 – 设置 UEVG 位 – 从模式控制器产生的更新具有缓存的寄存器被装入它们的预装载值。（更新影子寄存器） 1：禁止 UEV。不产生更新事件，影子寄存器（AR、DIV、CCx）保持它们的值。如果设置了 UEVG 位或从模式控制器发出了一个硬件复位，则计数器和预分频器被重新初始化。
位 0	**CNTEN**：使能计数器 （Counter enable） 0：禁止计数器； 1：使能计数器。 *注：在软件设置了 CNTEN 位后，外部时钟、门控模式和编码器模式才能工作。触发模式可以自动地通过硬件设置 CNTEN 位。*

10.4.4.2 TMR1 、TMR8控制寄存器2（TMRx_CTRL2）

偏移地址：0x04

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留	OC4IS	OC3NIS	OC3IS	OC2NIS	OC2IS	OC1NIS	OC1IS	TI1SEL	MMSEL[2：0]			CDSEL	CUSEL	保留	CPC
res	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	res	rw

位 15	保留，始终读为 0。
位 14	**OC4IS**：输出空闲状态 4（OC4 输出）。参见 OC1IS 位。
位 13	**OC3NIS**：输出空闲状态 3（OC3N 输出）。参见 OC1NIS 位。
位 12	**OC3IS**：输出空闲状态 3（OC3 输出）。参见 OC1IS 位。

位 11	**OC2NIS**：输出空闲状态 2（OC2N 输出）。参见 OC1NIS 位。
位 10	**OC2IS**：输出空闲状态 2（OC2 输出）。参见 OC1IS 位。
位 9	**OC1NIS**：输出空闲状态 1（OC1N 输出） （Output Idle state 1） 0：当 MOEN=0 时，死区后 OC1N=0； 1：当 MOEN=0 时，死区后 OC1N=1。 *注：已经设置了 LOCK（TMRx_BKR 寄存器）级别 1、2 或 3 后，该位不能被修改。*
位 8	**OC1IS**：输出空闲状态 1（OC1 输出） （Output Idle state 1） 0：当 MOEN=0 时，如果实现了 OC1N，则死区后 OC1=0。 1：当 MOEN=0 时，如果实现了 OC1N，则死区后 OC1=1。 *注：已经设置了 LOCK（TMRx_BKR 寄存器）级别 1、2 或 3 后，该位不能被修改。*
位 7	**TI1SEL**：TI1 选择 （TI1 selection） 0：TMRx_CH1 引脚连到 TI1 输入。 1：TMRx_CH1、TMRx_CH2 和 TMRx_CH3 引脚经异或后连到 TI1 输入。
位 6：4	**MMSEL[2：0]**：主模式选择 （Master mode selection） 这 3 位用于选择在主模式下送到从定时器的同步信息（TRGO）。可能的组合如下： 000：复位 – TMRx_EVEG 寄存器的 UEVG 位被用于作为触发输出（TRGO）。如果是触发输入产生的复位（从模式控制器处于复位模式），则 TRGO 上的信号相对实际的复位会有一个延迟。 001：使能–计数器使能信号 CNT_EN 被用于作为触发输出（TRGO）。有时需要在同一时间启动多个定时器或控制在一段时间内使能从定时器。计数器使能信号是通过 CNTEN 控制位和门控模式下的触发输入信号的逻辑或产生。当计数器使能信号受控于触发输入时，TRGO 上会有一个延迟，除非选择了主/从模式（见 TMRx_SMC 寄存器中 MSMODE 位的描述）。 010：更新 – 更新事件被选为触发输入（TRGO）。例如，一个主定时器的时钟可以被用作一个从定时器的预分频器。 011：比较脉冲 – 在发生一次捕获或一次比较成功时，当要设置 C1IF 标志时（即使它已经为高），触发输出送出一个正脉冲（TRGO）。 100：比较 – OC1REF 信号被用于作为触发输出（TRGO）。 101：比较 – OC2REF 信号被用于作为触发输出（TRGO）。 110：比较 – OC3REF 信号被用于作为触发输出（TRGO）。 111：比较 – OC4REF 信号被用于作为触发输出（TRGO）。
位 3	**CDSEL**：捕获/比较的 DMA 选择 （Capture/compare DMA selection） 0：当发生 CCx 事件时，送出 CCx 的 DMA 请求； 1：当发生更新事件时，送出 CCx 的 DMA 请求。
位 2	**CUSEL**：捕获/比较控制更新选择 （Capture/compare control update selection） 0：如果捕获/比较控制位是预装载的（CPC=1），只能通过设置 HALL 位更新它们； 1：如果捕获/比较控制位是预装载的（CPC=1），可以通过设置 HALL 位或 TRGI 上的一个上升沿更新它们。 *注：该位只对具有互补输出的通道起作用。*
位 1	保留，始终读为 0。
位 0	**CPC**：捕获/比较预装载控制位 （Capture/compare preloaded control） 0：CxEN，CxNEN 和 OCxMODE 位不是预装载的； 1：CxEN，CxNEN 和 OCxMODE 位是预装载的；设置该位后，它们只在发生一个 HALL 事件的时候（设置了 HALL 位或检测到 TRGI 的上升沿，依据 CUSEL 位）被更新。 *注：该位只对具有互补输出的通道起作用。*

10.4.4.3 TMR1、TMR8从模式控制寄存器（TMRx_SMC）

偏移地址：0x08

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ETRGP	ECLKEN	ETD[1：0]		ETDF[3：0]				MSMODE	TRGSEL[2：0]			保留	SMSEL[2：0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	res	rw	rw	rw

位 15	**ETRGP：**外部触发极性 （External trigger polarity） 该位选择是用 ETR 还是 ETR 的反相来作为触发操作 0：ETR 不反相，高电平或上升沿有效； 1：ETR 被反相，低电平或下降沿有效。
位 14	**ECLKEN：**外部时钟使能位 （External clock enable） 该位启用外部时钟模式 2 0：禁止外部时钟模式 2； 1：使能外部时钟模式 2。计数器由 ETRF 信号上的任意有效边沿驱动。 *注 1：设置 ECLKEN 位与选择外部时钟模式 1 并将 TRGI 连到 ETRF（SMSEL=111 和 TRGSEL=111）具有相同功效。* *注 2：下述从模式可以与外部时钟模式 2 同时使用：复位模式，门控模式和触发模式；但是，这时 TRGI 不能连到 ETRF（TRGSEL 位不能是'111'）。* *注 3：外部时钟模式 1 和外部时钟模式 2 同时被使能时，外部时钟的输入是 ETRF。*
位 13：12	**ETD[1：0]：**外部触发预分频 （External trigger divide） 外部触发信号 ETRP 的频率必须最多是 TMRxCLK 频率的 1/4。当输入较快的外部时钟时，可以使用预分频降低 ETRP 的频率。 00：关闭预分频； 01：ETRP 频率除以 2； 10：ETRP 频率除以 4； 11：ETRP 频率除以 8。
位 11：8	**ETDF[3：0]：**外部触发滤波 （External trigger filter） 这些位定义了对 ETRP 信号采样的频率和对 ETRP 数字滤波的带宽。实际上，数字滤波器是一个事件计数器，它记录到 N 个事件后会产生一个输出的跳变。 0000：无滤波器，以f_{DTS}采样 0001：采样频率 $f_{SAMPLING}=f_{CK_INT}$，N=2 0010：采样频率 $f_{SAMPLING}=f_{CK_INT}$，N=4 0011：采样频率 $f_{SAMPLING}=f_{CK_INT}$，N=8 0100：采样频率 $f_{SAMPLING}=f_{DTS}/2$，N=6 0101：采样频率 $f_{SAMPLING}=f_{DTS}/2$，N=8 0110：采样频率 $f_{SAMPLING}=f_{DTS}/4$，N=6 0111：采样频率 $f_{SAMPLING}=f_{DTS}/4$，N=8 1000：采样频率$f_{SAMPLING}=f_{DTS}/8$，N=6 1001：采样频率$f_{SAMPLING}=f_{DTS}/8$，N=8 1010：采样频率$f_{SAMPLING}=f_{DTS}/16$，N=5 1011：采样频率$f_{SAMPLING}=f_{DTS}/16$，N=6 1100：采样频率$f_{SAMPLING}=f_{DTS}/16$，N=8 1101：采样频率$f_{SAMPLING}=f_{DTS}/32$，N=5 1110：采样频率$f_{SAMPLING}=f_{DTS}/32$，N=6 1111：采样频率$f_{SAMPLING}=f_{DTS}/32$，N=8
位 7	**MSMODE：**主/从模式 （Master/slave mode） 0：无作用； 1：触发输入（TRGI）上的事件被延迟了，以允许在当前定时器（通过 TRGO）与它的从定时器间的完美同步。这对要求把几个定时器同步到一个单一的外部事件时是非常有用的。
位 6：4	**TRGSEL[2：0]：**触发选择 （Trigger selection） 这 3 位选择用于同步计数器的触发输入。 000：内部触发 0（ITR0）　100：TI1 的边沿检测器（TI1F_ED） 001：内部触发 1（ITR1）　101：滤波后的定时器输入 1（TI1FP1） 010：内部触发 2（ITR2）　110：滤波后的定时器输入 2（TI2FP2） 011：内部触发 3（ITR3）　111：外部触发输入（ETRF） 更多有关 ITRx 的细节，参见表 10-13。 *注：这些位只能在未用到（如 SMSEL=000）时被改变，以避免在改变时产生错误的边沿检测。*
位 3	保留，始终读为 0。

位 2：0	**SMSEL[2：0]：**从模式选择 （Slave mode selection） 当选择了外部信号，触发信号（TRGI）的有效边沿与选中的外部输入极性相关（见输入控制寄存器和控制寄存器的说明） 000：关闭从模式 – 如果 CNTEN=1，则预分频器直接由内部时钟驱动。 001：编码器模式 1 – 根据 TI2FP2 的电平，计数器在 TI1FP1 的边沿向上/下计数。 010：编码器模式 2 – 根据 TI1FP1 的电平，计数器在 TI2FP2 的边沿向上/下计数。 011：编码器模式 3 – 根据另一个信号的输入电平，计数器在 TI1FP1 和 TI2FP2 的边沿向上/下计数。 100：复位模式 – 选中的触发输入（TRGI）的上升沿重新初始化计数器，并且产生一个更新寄存器的信号。 101：门控模式 – 当触发输入（TRGI）为高时，计数器的时钟开启。一旦触发输入变为低，则计数器停止（但不复位）。计数器的启动和停止都是受控的。 110：触发模式 – 计数器在触发输入 TRGI 的上升沿启动（但不复位），只有计数器的启动是受控的。 111：外部时钟模式 1 – 选中的触发输入（TRGI）的上升沿驱动计数器。 *注：如果 TI1F_EN 被选为触发输入（TRGSEL=100）时，不要使用门控模式。这是因为，TI1F_ED 在每次 TI1F 变化时输出一个脉冲，然而门控模式是要检查触发输入的电平。*

表 10-13 TMRx内部触发连接

从定时器	ITR0（TRGSEL=000）	ITR1（TRGSEL=001）	ITR2（TRGSEL=010）	ITR3（TRGSEL=011）
TMR1	TMR5	TMR2	TMR3	TMR4
TMR8	TMR1	TMR2	TMR4	TMR5
保留	TMR8	TMR2	TMR4	TMR5

10.4.4.4 TMR1、TMR8 DMA/中断使能寄存器（TMRx_DIE）

偏移地址：0x0C

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留	TRGDE	HALLDE	C4DE	C3DE	C2DE	C1DE	UEVDE	BRKIE	TRGIE	HALLIE	C4IE	C3IE	C2IE	C1IE	UEVIE
res	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 15	保留，始终读为 0。
位 14	**TRGDE：**允许触发 DMA 请求 （Trigger DMA request enable） 0：禁止触发 DMA 请求； 1：允许触发 DMA 请求。
位 13	**HALLDE：**允许 HALL 的 DMA 请求 （HALL DMA request enable） 0：禁止 HALL 的 DMA 请求； 1：允许 HALL 的 DMA 请求。
位 12	**C4DE：**允许捕获/比较 4 的 DMA 请求 （Capture/Compare 4 DMA request enable） 0：禁止捕获/比较 4 的 DMA 请求； 1：允许捕获/比较 4 的 DMA 请求。
位 11	**C3DE：**允许捕获/比较 3 的 DMA 请求 （Capture/Compare 3 DMA request enable） 0：禁止捕获/比较 3 的 DMA 请求； 1：允许捕获/比较 3 的 DMA 请求。
位 10	**C2DE：**允许捕获/比较 2 的 DMA 请求 （Capture/Compare 2 DMA request enable） 0：禁止捕获/比较 2 的 DMA 请求； 1：允许捕获/比较 2 的 DMA 请求。

位 9	**C1DE：**允许捕获/比较 1 的 DMA 请求 （Capture/Compare 1 DMA request enable） 0：禁止捕获/比较 1 的 DMA 请求； 1：允许捕获/比较 1 的 DMA 请求。
位 8	**UEVDE：**允许更新的 DMA 请求 （Update DMA request enable） 0：禁止更新的 DMA 请求； 1：允许更新的 DMA 请求。
位 7	**BRKIE：**允许刹车中断 （Break interrupt enable） 0：禁止刹车中断； 1：允许刹车中断。
位 6	**TRGIE：**触发中断使能 （Trigger interrupt enable） 0：禁止触发中断； 1：使能触发中断。
位 5	**HALLIE：**允许 HALL 中断 （HALL interrupt enable） 0：禁止 HALL 中断； 1：允许 HALL 中断。
位 4	**C4IE：**允许捕获/比较 4 中断 （Capture/Compare 4 interrupt enable） 0：禁止捕获/比较 4 中断； 1：允许捕获/比较 4 中断。
位 3	**C3IE：**允许捕获/比较 3 中断 （Capture/Compare 3 interrupt enable） 0：禁止捕获/比较 3 中断； 1：允许捕获/比较 3 中断。
位 2	**C2IE：**允许捕获/比较 2 中断 （Capture/Compare 2 interrupt enable） 0：禁止捕获/比较 2 中断； 1：允许捕获/比较 2 中断。
位 1	**C1IE：**允许捕获/比较 1 中断 （Capture/Compare 1 interrupt enable） 0：禁止捕获/比较 1 中断； 1：允许捕获/比较 1 中断。
位 0	**UEVIE：**允许更新中断 （Update interrupt enable） 0：禁止更新中断； 1：允许更新中断。

10.4.4.5 TMR1、TMR8状态寄存器（TMRx_STS）

偏移地址：0x10

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留			C4OF	C3OF	C2OF	C1OF	保留	BRKIF	TRGIF	HALLIF	C4IF	C3IF	C2IF	C1IF	UEVIF
res			rc_w0	rc_w0	rc_w0	rc_w0	res	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0

位 15：13	保留，始终读为 0。
位 12	**C4OF：**捕获/比较 4 重复捕获标记 （Capture/Compare 4 overcapture flag） 参见 C1OF 描述。
位 11	**C3OF：**捕获/比较 3 重复捕获标记 （Capture/Compare 3 overcapture flag） 参见 C1OF 描述。
位 10	**C2OF：**捕获/比较 2 重复捕获标记 （Capture/Compare 2 overcapture flag） 参见 C1OF 描述。

位 9	**C1OF**：捕获/比较 1 重复捕获标记 （Capture/Compare 1 overcapture flag） 仅当相应的通道被配置为输入捕获时，该标记可由硬件置 1。写 0 可清除该位。 0：无重复捕获产生； 1：计数器的值被捕获到 TMRx_CC1 寄存器时，C1IF 的状态已经为'1'。
位 8	保留，始终读为 0。
位 7	**BRKIF**：刹车中断标记 （Break interrupt flag） 一旦刹车输入有效，由硬件对该位置'1'。如果刹车输入无效，则该位可由软件清'0'。 0：无刹车事件产生； 1：刹车输入上检测到有效电平。
位 6	**TRGIF**：触发器中断标记 （Trigger interrupt flag） 当发生触发事件（当从模式控制器处于除门控模式外的其它模式时，在 TRGI 输入端检测到有效边沿，或门控模式下的任一边沿）时由硬件对该位置'1'。它由软件清'0'。 0：无触发器事件产生； 1：触发中断等待响应。
位 5	**HALLIF**：HALL 中断标记 （HALL interrupt flag） 一旦产生 HALL 事件（当捕获/比较控制位：CxEN、CxNEN、OCxMODE 已被更新）该位由硬件置'1'。它由软件清'0'。 0：无 HALL 事件产生； 1：HALL 中断等待响应。
位 4	**C4IF**：捕获/比较 4 中断标记 （Capture/Compare 4 interrupt flag） 参考 C1IF 描述。
位 3	**C3IF**：捕获/比较 3 中断标记 （Capture/Compare 3 interrupt flag） 参考 C1IF 描述。
位 2	**C2IF**：捕获/比较 2 中断标记 （Capture/Compare 2 interrupt flag） 参考 C1IF 描述。
位 1	**C1IF**：捕获/比较 1 中断标记 （Capture/Compare 1 interrupt flag） **如果通道 CC1 配置为输出模式：** 当计数器值与比较值匹配时该位由硬件置 1，但在中心对称模式下除外（参考 TMRx_CTRL1 寄存器 的 CMSEL 位）。它由软件清'0'。 0：无匹配发生； 1：TMRx_CNT 的值与 TMRx_CC1 的值匹配。 当 TMRx_CC1 的内容大于 TMRx_AR 的内容时，在向上或向上/下计数模式时计数器溢出，或向下计数模式时的计数器下溢条件下，C1IF 位变高。 **如果通道 CC1 配置为输入模式：** 当捕获事件发生时该位由硬件置'1'，它由软件清'0'或通过读 TMRx_CC1 清'0'。 0：无输入捕获产生； 1：计数器值已被捕获（拷贝）至 TMRx_CC1（在 IC1 上检测到与所选极性相同的边沿）。
位 0	**UEVIF**：更新中断标记 （Update interrupt flag） 当产生更新事件时该位由硬件置'1'。它由软件清'0'。 0：无更新事件产生； 1：更新中断等待响应。当寄存器被更新时该位由硬件置'1'： – 若 TMRx_CTRL1 寄存器的 UEVDIS=0，当重复计数器数值上溢或下溢时（重复计数器=0 时产生更新事件）。 – 若 TMRx_CTRL1 寄存器的 UVERS=0、UEVDIS=0，当设置 TMRx_EVEG 寄存器的 UEVG=1 时产生更新事件，通过软件对计数器 CNT 重新初始化时。 – 若 TMRx_CTRL1 寄存器的 UVERS=0、UEVDIS=0，当计数器 CNT 被触发事件重新初始化时。（参考 10.4.4.3TMR1、TMR8 从模式控制寄存器（TMRx_SMC））。

10.4.4.6 TMR1、TMR8事件产生寄存器（TMRx_EVEG）

偏移地址：0x14

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留								BRKG	TRGG	HALLG	C4G	C3G	C2G	C1G	UEVG
res								w	w	w	w	w	w	w	w

位 15：8	保留，始终读为 0。
位 7	**BRKG**：产生刹车事件 （Break generation） 该位由软件置'1'，用于产生一个刹车事件，由硬件自动清'0'。 0：无动作； 1：产生一个刹车事件。此时 MOEN=0、BRKIF=1，若开启对应的中断和 DMA，则产生相应的中断和 DMA。
位 6	**TRGG**：产生触发事件 （Trigger generation） 该位由软件置'1'，用于产生一个触发事件，由硬件自动清'0'。 0：无动作； 1：TMRx_STS 寄存器的 TRGIF=1，若开启对应的中断和 DMA，则产生相应的中断和 DMA。
位 5	**HALLG**：捕获/比较事件，产生控制更新 （Capture/Compare control update generation） 该位由软件置'1'，由硬件自动清'0'。 0：无动作； 1：当 CPC=1，允许更新 CxEN、CxNEN、OCxMODE 位。 *注：该位只对拥有互补输出的通道有效。*
位 4	**C4G**：产生捕获/比较 4 事件 （Capture/Compare 4 generation） 参考 C1G 描述。
位 3	**C3G**：产生捕获/比较 3 事件 （Capture/Compare 3 generation） 参考 C1G 描述。
位 2	**C2G**：产生捕获/比较 2 事件 （Capture/Compare 2 generation） 参考 C1G 描述。
位 1	**C1G**：产生捕获/比较 1 事件 （Capture/Compare 1 generation） 该位由软件置'1'，用于产生一个捕获/比较事件，由硬件自动清'0'。 0：无动作； 1：在通道 CC1 上产生一个捕获/比较事件： **若通道 CC1 配置为输出：** 设置 C1IF=1，若开启对应的中断和 DMA，则产生相应的中断和 DMA。 **若通道 CC1 配置为输入：** 当前的计数器值被捕获至 TMRx_CC1 寄存器；设置 C1IF=1，若开启对应的中断和 DMA，则产生相应的中断和 DMA。若 C1IF 已经为 1，则设置 C1OF=1。
位 0	**UEVG**：产生更新事件 （Update generation） 该位由软件置'1'，由硬件自动清'0'。 0：无动作； 1：重新初始化计数器，并产生一个更新事件。注意预分频器的计数器也被清'0'（但是预分频系数不变）。若在中心对称模式下或 DIR=0（向上计数）则计数器被清'0'；若 DIR=1（向下计数）则取计数器取 TMRx_AR 的值。

10.4.4.7 TMR1、TMR8捕获/比较模式寄存器1（TMRx_CCM1）

偏移地址：0x18

复位值：0x0000

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 CxSEL 位定义。该寄存器其它位

的作用在输入和输出模式下不同。OCxx 描述了通道在输出模式下的功能，ICxx 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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC2 DIS	OC2MODE[2：0]			OC2 PEN	OC2 FEN	C2SEL[1：0]		OC1 DIS	OC1MODE[2：0]			OC1 PEN	OC1 FEN	C1SEL[1：0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 15	**OC2DIS**：输出比较 2 清 0 使能 （Output Compare 2 clear enable）
位 14：12	**OC2MODE[2：0]**：输出比较 2 模式 （Output Compare 2 mode）
位 11	**OC2PEN**：输出比较 2 预装载使能 （Output Compare 2 preload enable）
位 10	**OC2FEN**：输出比较 2 快速使能 （Output Compare 2 fast enable）
位 9：8	**C2SEL[1：0]**：捕获/比较 2 选择。（Capture/Compare 2 selection） 该位定义通道的方向（输入/输出），及输入脚的选择： 00：CC2 通道被配置为输出； 01：CC2 通道被配置为输入，IC2 映射在 TI2 上； 10：CC2 通道被配置为输入，IC2 映射在 TI1 上； 11：CC2 通道被配置为输入，IC2 映射在 TRC 上。此模式仅工作在内部触发器输入被选中时（由 TMRx_SMC 寄存器的 TRGSEL 位选择）。 *注：C2SEL 仅在通道关闭时（TMRx_CCE 寄存器的 C2NEN=0）才是可写的。*
位 7	**OC1DIS**：输出比较 1 清'0'使能 （Output Compare 1 clear enable） 0：OC1REF 不受 ETRF 输入的影响； 1：一旦检测到 ETRF 输入高电平，清除 OC1REF=0。
位 6：4	**OC1MODE[2：0]**：输出比较 1 模式 （Output Compare 1 mode） 该 3 位定义了输出参考信号 OC1REF 的动作，而 OC1REF 决定了 OC1、OC1N 的值。OC1REF 是高电平有效，而 OC1、OC1N 的有效电平取决于 C1P、C1NP 位。 000：冻结。输出比较寄存器 TMRx_CC1 与计数器 TMRx_CNT 间的比较对 OC1REF 不起作用； 001：匹配时设置通道 1 为有效电平。当计数器 TMRx_CNT 的值与捕获 / 比较寄存器 1（TMRx_CC1）相同时，强制 OC1REF 为高。 010：匹配时设置通道 1 为无效电平。当计数器 TMRx_CNT 的值与捕获 / 比较寄存器 1（TMRx_CC1）相同时，强制 OC1REF 为低。 011：翻转。当 TMRx_CC1=TMRx_CNT 时，翻转 OC1REF 的电平。 100：强制为无效电平。强制 OC1REF 为低。 101：强制为有效电平。强制 OC1REF 为高。 110：PWM 模式 1－ 在向上计数时，一旦 TMRx_CNT<TMRx_CC1 时通道 1 为有效电平，否则为无效电平；在向下计数时，一旦 TMRx_CNT>TMRx_CC1 时通道 1 为无效电平（OC1REF=0），否则为有效电平（OC1REF=1）。 111：PWM 模式 2－在向上计数时，一旦 TMRx_CNT<TMRx_CC1 时通道 1 为无效电平，否则为有效电平；在向下计数时，一旦 TMRx_CNT>TMRx_CC1 时通道 1 为有效电平，否则为无效电平。 *注 1：一旦 LOCK 级别设为 3（TMRx_BRKDT 寄存器中的 LOCKC 位）并且 C1SEL=00（该通道配置成输出）则该位不能被修改。* *注 2：在 PWM 模式 1 或 PWM 模式 2 中，只有当比较结果改变了或在输出比较模式中从冻结模式切换到 PWM 模式时，OC1REF 电平才改变。*

位 3	**OC1PEN**：输出比较 1 预装载使能 （Output Compare 1 preload enable） 0：禁止 TMRx_CC1 寄存器的预装载功能，可随时写入 TMRx_CC1 寄存器，并且新写入的数值立即起作用。 1：开启 TMRx_CC1 寄存器的预装载功能，读写操作仅对预装载寄存器操作，TMRx_CC1 的预装载值在更新事件到来时被加载至当前寄存器中。 *注 1：一旦 LOCK 级别设为 3（TMRx_BRKDT 寄存器中的 LOCKC 位）并且 C1SEL=00（该通道配置成输出）则该位不能被修改。* *注 2：仅在单脉冲模式下（TMRx_CTRL1 寄存器的 OPMODE=1），可以在未确认预装载寄存器情况下使用 PWM 模式，否则其动作不确定。*
位 2	**OC1FEN**：输出比较 1 快速使能 （Output Compare 1 fast enable） 该位用于加快 CC 输出对触发输入事件的响应。 0：根据计数器与 CC1 的值，CC1 正常操作，即使触发器是打开的。当触发器的输入有一个有效沿时，激活 CC1 输出的最小延时为 5 个时钟周期。 1：输入到触发器的有效沿的作用就像发生了一次比较匹配。因此，OC 被设置为比较电平而与比较结果无关。采样触发器的有效沿和 CC1 输出间的延时被缩短为 3 个时钟周期。 OC1FEN 只在通道被配置成 PWM1 或 PWM2 模式时起作用。
位 1：0	**C1SEL[1：0]**：捕获/比较 1 选择。（Capture/Compare 1 selection） 这 2 位定义通道的方向（输入/输出），及输入脚的选择： 00：CC1 通道被配置为输出； 01：CC1 通道被配置为输入，IC1 映射在 TI1 上； 10：CC1 通道被配置为输入，IC1 映射在 TI2 上； 11：CC1 通道被配置为输入，IC1 映射在 TRC 上。此模式仅工作在内部触发器输入被选中时（由 TMRx_SMC 寄存器的 TRGSEL 位选择）。 *注：C1SEL 仅在通道关闭时（TMRx_CCE 寄存器的 C1EN=0）才是可写的。*

输入捕获模式

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IC2DF[3：0]				IC2DIV[1：0]		C2SEL[1：0]		IC1DF[3：0]				IC1DIV[1：0]		C1SEL[1：0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 15：12	**IC2DF[3：0]**：输入捕获 2 滤波器 （Input capture 2 filter）
位 11：10	**IC2DIV[1：0]**：输入/捕获 2 预分频器 （Input capture 2 prescaler）
位 9：8	**C2SEL[1：0]**：捕获/比较 2 选择 （Capture/Compare 2 selection） 这 2 位定义通道的方向（输入/输出），及输入脚的选择： 00：CC2 通道被配置为输出； 01：CC2 通道被配置为输入，IC2 映射在 TI2 上； 10：CC2 通道被配置为输入，IC2 映射在 TI1 上； 11：CC2 通道被配置为输入，IC2 映射在 TRC 上。此模式仅工作在内部触发器输入被选中时（由 TMRx_SMC 寄存器的 TRGSEL 位选择）。 *注：C2SEL 仅在通道关闭时（TMRx_CCE 寄存器的 C2EN=0）才是可写的。*
位 7：4	**IC1DF [3：0]**：输入捕获 1 滤波 （Input capture 1 filter） 这几位定义了TI1 输入的采样频率及数字滤波器长度。数字滤波器由一个事件计数器组成，它记录到 N 个事件后会产生一个输出的跳变： 0000：无滤波器，以f_{DTS}采样 0001：采样频率 $f_{SAMPLING}=f_{CK_INT}$，N=2 0010：采样频率 $f_{SAMPLING}=f_{CK_INT}$，N=4 0011：采样频率 $f_{SAMPLING}=f_{CK_INT}$，N=8 0100：采样频率 $f_{SAMPLING}=f_{DTS}/2$，N=6 0101：采样频率 $f_{SAMPLING}=f_{DTS}/2$，N=8 0110：采样频率 $f_{SAMPLING}=f_{DTS}/4$，N=6 0111：采样频率 $f_{SAMPLING}=f_{DTS}/4$，N=8 1000：采样频率$f_{SAMPLING}=f_{DTS}/8$，N=6 1001：采样频率$f_{SAMPLING}=f_{DTS}/8$，N=8 1010：采样频率$f_{SAMPLING}=f_{DTS}/16$，N=5 1011：采样频率$f_{SAMPLING}=f_{DTS}/16$，N=6 1100：采样频率$f_{SAMPLING}=f_{DTS}/16$，N=8 1101：采样频率$f_{SAMPLING}=f_{DTS}/32$，N=5 1110：采样频率$f_{SAMPLING}=f_{DTS}/32$，N=6 1111：采样频率$f_{SAMPLING}=f_{DTS}/32$，N=8

位 3：2	**IC1DIV[1：0]**：输入/捕获 1 预分频器 （Input capture 1 prescaler） 这 2 位定义了 CC1 输入（IC1）的预分频系数.一旦 C1EN=0（TMRx_CCE 寄存器中），则预分频器复位。 00：无预分频器，捕获输入口上检测到的每一个边沿都触发一次捕获； 01：每 2 个事件触发一次捕获； 10：每 4 个事件触发一次捕获； 11：每 8 个事件触发一次捕获。
位 1：0	**C1SEL[1：0]**：捕获/比较 1 选择 （Capture/Compare 1 Selection） 这 2 位定义通道的方向（输入/输出），及输入脚的选择： 00：CC1 通道被配置为输出； 01：CC1 通道被配置为输入，IC1 映射在 TI1 上； 10：CC1 通道被配置为输入，IC1 映射在 TI2 上； 11：CC1 通道被配置为输入，IC1 映射在 TRC 上。此模式仅工作在内部触发器输入被选中时（由 TMRx_SMC 寄存器的 TRGSEL 位选择）。 *注：C1SEL 仅在通道关闭时（TMRx_CCE 寄存器的 C1EN=0）才是可写的。*

10.4.4.8 TMR1、TMR8捕获/比较模式寄存器2（TMRx_CCM2）

偏移地址：0x1C

复位值：0x0000

参看以上 CCM1 寄存器描述

输出比较模式

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC4 DIS	OC4MODE[2：0]			OC4 PE	OC4 FEN	C4SEL[1：0]		OC3 DIS	OC3MODE[2：0]			OC3 PEN	OC3 FEN	C3SEL[1：0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 15	**OC4DIS**：输出比较 4 清 0 使能 （Output compare 4 clear enable）
位 14：12	**OC4MODE[2：0]**：输出比较 4 模式 （Output compare 4 mode）
位 11	**OC4PEN**：输出比较 4 预装载使能 （Output compare 4 preload enable）
位 10	**OC4FEN**：输出比较 4 快速使能 （Output compare 4 fast enable）
位 9：8	**C4SEL[1：0]**：捕获/比较 4 选择 （Capture/Compare 4 selection） 该 2 位定义通道的方向（输入/输出），及输入脚的选择： 00：CC4 通道被配置为输出； 01：CC4 通道被配置为输入，IC4 映射在 TI4 上； 10：CC4 通道被配置为输入，IC4 映射在 TI3 上； 11：CC4 通道被配置为输入，IC4 映射在 TRC 上。此模式仅工作在内部触发器输入被选中时（由 TMRx_SMC 寄存器的 TRGSEL 位选择）。 *注：CC4S 仅在通道关闭时（TMRx_CCE 寄存器的 C4EN=0）才是可写的。*
位 7	**OC3DIS**：输出比较 3 清 0 使能 （Output compare 3 clear enable）
位 6：4	**OC3MODE[2：0]**：输出比较 3 模式 （Output compare 3 mode）
位 3	**OC3PEN**：输出比较 3 预装载使能 （Output compare 3 preload enable）
位 2	**OC3FEN**：输出比较 3 快速使能 （Output compare 3 fast enable）

位 1：0	**C3SEL[1：0]：**捕获/比较 3 选择 （Capture/Compare 3 selection） 这 2 位定义通道的方向（输入/输出），及输入脚的选择： 00：CC3 通道被配置为输出； 01：CC3 通道被配置为输入，IC3 映射在 TI3 上； 10：CC3 通道被配置为输入，IC3 映射在 TI4 上； 11：CC3 通道被配置为输入，IC3 映射在 TRC 上。此模式仅工作在内部触发器输入被选中时（由 TMRx_SMC 寄存器的 TRGSEL 位选择）。 *注：C3SEL 仅在通道关闭时（TMRx_CCE 寄存器的C3EN=0）才是可写的。*

输入捕获模式

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IC4DF[3：0]				IC4DIV[1：0]		C4S[1：0]		IC3DF[3：0]				IC3DIV[1：0]		C3SEL[1：0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 15：12	**IC4DF[3：0]：**输入捕获 4 滤波器 （Input capture 4 filter）
位 11：10	**IC4DIV[1：0]：**输入/捕获 4 预分频器 （Input capture 4 prescaler）
位 9：8	**C4S[1：0]：**捕获/比较 4 选择 （Capture/Compare 4 selection） 这 2 位定义通道的方向（输入/输出），及输入脚的选择： 00：CC4 通道被配置为输出； 01：CC4 通道被配置为输入，IC4 映射在 TI4 上； 10：CC4 通道被配置为输入，IC4 映射在 TI3 上； 11：CC4 通道被配置为输入，IC4 映射在 TRC 上。此模式仅工作在内部触发器输入被选中时（由 TMRx_SMC 寄存器的 TRGSEL 位选择）。 *注：CC4S 仅在通道关闭时（TMRx_CCE 寄存器的C4EN=0）才是可写的。*
位 7：4	**IC3DF[3：0]：**输入捕获 3 滤波器 （Input capture 3 filter）
位 3：2	**IC3DIV[1：0]：**输入/捕获 3 预分频器 （Input capture 3 prescaler）
位 1：0	**C3SEL[1：0]：**捕获/比较 3 选择 （Capture/compare 3 selection） 这 2 位定义通道的方向（输入/输出），及输入脚的选择： 00：CC3 通道被配置为输出； 01：CC3 通道被配置为输入，IC3 映射在 TI3 上； 10：CC3 通道被配置为输入，IC3 映射在 TI4 上； 11：CC3 通道被配置为输入，IC3 映射在 TRC 上。此模式仅工作在内部触发器输入被选中时（由 TMRx_SMC 寄存器的 TRGSEL 位选择）。 *注：C3SEL 仅在通道关闭时（TMRx_CCE 寄存器的C3EN=0）才是可写的。*

10.4.4.9 TMR1、TMR8捕获/比较使能寄存器（TMRx_CCE）

偏移地址：0x20

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留		C4P	C4EN	C3NP	C3NEN	C3P	C3EN	C2NP	C2NEN	C2P	C2EN	C1NP	C1NEN	C1P	C1EN
res		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 15：14	保留，始终读为 0。
位 13	**C4P：**输入/捕获 4 输出极性 （Capture/Compare 4 output polarity） 参考 C1P 的描述。
位 12	**C4EN：**输入/捕获 4 输出使能 （Capture/Compare 4 output enable） 参考 C1EN 的描述。

位 11	**C3NP**：输入/捕获 3 互补输出极性（Capture/Compare 3 complementary output polarity） 参考 C1NP 的描述。
位 10	**C3NEN**：输入/捕获 3 互补输出使能（Capture/Compare 3 complementary output enable） 参考 C1NEN 的描述。
位 9	**C3P**：输入/捕获 3 输出极性（Capture/Compare 3 output polarity） 参考 C1P 的描述。
位 8	**C3EN**：输入/捕获 3 输出使能 （Capture/Compare 3 output enable） 参考 C1EN 的描述。
位 7	**C2NP**：输入/捕获 2 互补输出极性（Capture/Compare 2 complementary output polarity） 参考 C1NP 的描述。
位 6	**C2NEN**：输入/捕获 2 互补输出使能（Capture/Compare 2 complementary output enable） 参考 C1NEN 的描述。
位 5	**C2P**：输入/捕获 2 输出极性 （Capture/Compare 2 output polarity） 参考 C1P 的描述。
位 4	**C2EN**：输入/捕获 2 输出使能 （Capture/Compare 2 output enable） 参考 C1EN 的描述。
位 3	**C1NP**：输入/捕获 1 互补输出极性 （Capture/Compare 1 complementary output polarity） **CC1 通道配置为输出：** 0：OC1N 高电平有效； 1：OC1N 低电平有效。 **CC1 通道配置为输入：** C1NP 结合 C1P 定义 TI1FP1/TI2FP1 极性（参考 C1P 描述） *注：一旦 LOCK 级别（TMRx_BRKDT 寄存器中的 LOCKC 位）设为 3 或 2 且 C1SEL=00（通道配置为输出）则该位不能被修改。*
位 2	**C1NEN**：输入/捕获 1 互补输出使能（Capture/Compare 1 complementary output enable） 0：关闭－ OC1N 禁止输出，因此 OC1N 的电平依赖于 MOEN、OSIMI、OSIMR、OC1IS、OC1NIS 和 C1EN 位的值。 1：开启－ OC1N 信号输出到对应的输出引脚，其输出电平依赖于 MOEN、OSIMI、OSIMR 、OC1IS、OC1NIS 和 C1EN 位的值。
位 1	**C1P**：输入/捕获 1 输出极性 （Capture/Compare 1 output polarity） **CC1 通道配置为输出：** 0：OC1 高电平有效 1：OC1 低电平有效 **CC1 通道配置为输入：** CC1NP/CC1P 位选择 TI1FP1 和 TI2FP1 的极性信号作为触发或捕获信号。 00：不反相/上升沿。电路 TIxFP1 上升沿敏感（捕获，触发中的复位，外部时钟或触发模式），TIxFP1 不反相（触发中的门控模式）。 01：反相/下降沿。电路对 TIxFP1 下降沿敏感（捕获，触发中的复位，外部时钟或触发模式），TIxFP1 反相（触发中的门控模式）。 10：保留。 11：不反相/双沿。电路同时对 TIxFP1 上升沿和下降沿敏感（捕获，触发中的复位，外部时钟或触发模式），TIxFP1 是不反相（触发中门控模式）。 *注：一旦 LOCK 级别（TMRx_BRKDT 寄存器中的 LOCKC 位）设为 3 或 2，则该位不能被修改。*
位 0	**C1EN**：输入/捕获 1 输出使能 （Capture/Compare 1 output enable） **CC1 通道配置为输出：** 0 ：关闭－ OC1 禁止输出，因此 OC1 的输出电平依赖于 MOEN、OSIMI、OSIMR、OC1IS、OC1NIS 和 C1NEN 位的值。 1 ：开启－ OC1 信号输出到对应的输出引脚，其输出电平依赖于 MOEN 、OSIMI 、OSIMR 、OC1IS、OC1NIS 和 C1NEN 位的值。 **CC1 通道配置为输入：** 该位决定了计数器的值是否能捕获入 TMRx_CC1 寄存器。 0：捕获禁止； 1：捕获使能。

表 10-14 带刹车功能的互补输出通道OCx和OCxN的控制位

控制位					输出状态（1）	
MOEN 位	OSIMI 位	OSIMR 位	CxEN 位	CxNEN 位	OCx 输出状态	OCxN 输出状态
1	X	0	0	0	输出禁止（与定时器断开） OCx=0，OCx_EN=0	输出禁止（与定时器断开） OCxN=0，OCxN_EN=0
		0	0	1	输出禁止（与定时器断开） OCx=0，OCx_EN=0	OCxREF + 极性， OCxN= OCxREF xor CxNP， OCxN_EN=1
		0	1	0	OCxREF+极性， OCx= OCxREF xor CxP， OCx_EN=1	输出禁止（与定时器断开） OCxN=0，OCxN_EN=0
		0	1	1	OCxREF+极性+死区， OCx_EN=1	OCxREF 反相+极性+死区， OCxN_EN=1
		1	0	0	输出禁止（与定时器断开） OCx=CxP，OCx_EN=0	输出禁止（与定时器断开） OCxN=CxNP，OCxN_EN=0
		1	0	1	关闭状态 （输出使能且为无效电平） OCx=CxP，OCx_EN=1	OCxREF + 极性， OCxN= OCxREF xor CxNP， OCxN_EN=1
		1	1	0	OCxREF + 极性， OCx= OCxREF xor CxP， OCx_EN=1	关闭状态 （输出使能且为无效电平） OCxN=CxNP，OCxN_EN=1
		1	1	1	OCxREF+极性+死区， OCx_EN=1	OCxREF 反相+极性+死区， OCxN_EN=1
0	0	X	0	0	输出禁止（与定时器断开） 异步地： OCx=CxP ，OCx_EN=0 ，OCxN=CxNP ，OCxN_EN=0； 若时钟存在：经过一个死区时间后 OCx=OC1IS，OCxN=OCxNIS，假设 OC1IS 与 OCxNIS 并不都对应 OCx 和 OCxN 的有效电平。	
	0		0	1		
	0		1	0		
	0		1	1		
	1		0	0	关闭状态（输出使能且为无效电平） 异步地： OCx=CxP ， OCx_EN=1 ， OCxN=CxNP ，OCxN_EN=1； 若时钟存在：经过一个死区 时间后 OCx=OC1IS，OCxN=OCxNIS，假设 OC1IS 与 OCxNIS 并不都对应 OCx 和 OCxN 的有效电平。	
	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 = CxNEN = 0)，那么 OC1IS， OCxNIS， CxP 和 CxNP 都必须清零。

注意： 引脚连接到互补的 OCx 和 OCxN 通道的外部 I/O 引脚的状态，取决于 OCx 和 OCxN 通道状态和 GPIO 以及 AFIO 寄存器。

10.4.4.10 TMR1、TMR8计数器（TMRx_CNT）

偏移地址：0x24

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT[15：0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 15：0	**CNT[15：0]：**计数器的值 （Counter value）

10.4.4.11 TMR1、TMR8预分频器（TMRx_DIV）

偏移地址：0x28

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DIV[15：0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 15：0	**DIV[15：0]：**预分频器的值 （Prescaler value） 计数器的时钟频率（CK_CNT）等于 f_{CK_DIV} / （ DIV[15：0]+1）。 DIV 包含了每次当更新事件产生时，装入当前预分频器寄存器的值；更新事件包括计数器被 TMR_EVEG 的 UEVG 位清’0’或被工作在复位模式的从控制器清’0’。

10.4.4.12 TMR1、TMR8自动重装载寄存器（TMRx_AR）

偏移地址：0x2C

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AR[15：0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 15：0	**AR[15：0]：**自动重装载的值 （Prescaler value） AR 包含了将要装载入实际的自动重装载寄存器的值。 详细参考 10.4.3.1 节：有关 AR 的更新和动作。 当自动重装载的值为空时，计数器不工作。

10.4.4.13 TMR1、TMR8重复计数寄存器（TMRx_RC）

偏移地址：0x30

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留								RC[7：0]							
res								rw	rw	rw	rw	rw	rw	rw	rw

位 15：8	保留，始终读为 0。
位 7：0	**RC[7：0]：** 重复计数器的值 （Repetition counter value） 开启了预装载功能后，这些位允许用户设置比较寄存器的更新速率（即周期性地从预装载寄存器传输到当前寄存器）。如果允许产生更新中断，则会同时影响产生更新中断的速率。每次向下计数器 RC_CNT 达到 0，会产生一个更新事件并且计数器 RC_CNT 重新从 RC 值开始计数。由于 RC_CNT 只有在周期更新事件 U_RC 发生时才重载 RC 值，因此对 TMRx_RC 寄存器写入的新值只在下次周期更新事件发生时才起作用。 这意味着在 PWM 模式中，（RC+1）对应着： — 在边沿对齐模式下，PWM 周期的数目； — 在中心对称模式下，PWM 半周期的数目；

10.4.4.14 TMR1、TMR8捕获/比较寄存器 1（TMRx_CC1）

偏移地址：0x34

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CC1[15：0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 15：0	**CC1[15：0]：**捕获/比较通道 1 的值 （Capture/Compare 1 value） **若 CC1 通道配置为输出：** CC1 包含了装入当前捕获/比较 1 寄存器的值（预装载值）。如果在 TMRx_CCM1 寄存器（OC1PEN 位）中未选择预装载功能，写入的数值会立即传输至当前寄存器中。否则只有当更新事件发生时，此预装载值才传输至当前捕获/比较 1 寄存器中。当前捕获/比较寄存器参与同计数器 TMRx_CNT 的比较，并在 OC1 端口上产生输出信号。 **若 CC1 通道配置为输入：** CC1 包含了由上一次输入捕获 1 事件（IC1）传输的计数器值。

10.4.4.15 TMR1、TMR8捕获/比较寄存器2（TMRx_CC2）

偏移地址：0x38

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CC2[15：0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 15：0	**CC2[15：0]：** 捕获/比较通道 2 的值 （Capture/Compare 2 value） **若 CC2 通道配置为输出：** CC2 包含了装入当前捕获/比较 2 寄存器的值（预装载值）。如果在 TMRx_CCM2 寄存器（OC2PEN 位）中未选择预装载特性，写入的数值会立即传输至当前寄存器中。否则只有当更新事件发生时，此预装载值才传输至当前捕获/比较 2 寄存器中。当前捕获/比较寄存器参与同计数器 TMRx_CNT 的比较，并在 OC2 端口上产生输出信号。 **若 CC2 通道配置为输入：** CC2 包含了由上一次输入捕获 2 事件（IC2）传输的计数器值。

10.4.4.16 TMR1、TMR8捕获/比较寄存器3（TMRx_CC3）

偏移地址：0x3C

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CC3[15：0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 15：0	**CC3[15：0]：**捕获/比较通道 3 的值 （Capture/Compare 3 value） **若 CC3 通道配置为输出：** CC3 包含了装入当前捕获/比较 3 寄存器的值（预装载值）。 如果在 TMRx_CCM3 寄存器（OC3PEN 位）中未选择预装载特性，写入的数值会立即传输至当前寄存器中。否则只有当更新事件发生时，此预装载值才传输至当前捕获/比较 3 寄存器中。当前捕获/比较寄存器参与同计数器 TMRx_CNT 的比较，并在 OC3 端口上产生输出信号。 **若 CC3 通道配置为输入：** CC3 包含了由上一次输入捕获 3 事件（IC3）传输的计数器值。

10.4.4.17 TMR1、TMR8捕获/比较寄存器4（TMRx_CC4）

偏移地址：0x40

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CC4[15：0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 15：0	**CC4[15：0]：**捕获/比较通道 4 的值 （Capture/Compare 4 value） **若 CC4 通道配置为输出：** CC4 包含了装入当前捕获/比较 4 寄存器的值（预装载值）。 如果在 TMRx_CCM4 寄存器（OC4PEN 位）中未选择预装载特性，写入的数值会立即传输至当前寄存器中。否则只有当更新事件发生时，此预装载值才传输至当前捕获/比较 4 寄存器中。当前捕获/比较寄存器参与同计数器 TMRx_CNT 的比较，并在 OC4 端口上产生输出信号。 **若 CC4 通道配置为输入：** CC4 包含了由上一次输入捕获 4 事件（IC4）传输的计数器值。

10.4.4.18 TMR1、TMR8刹车和死区寄存器（TMRx_BRKDT）

偏移地址：0x44

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MOEN	AOEN	BRKP	BRKEN	OSIMR	OSIMI	LOCKC[1：0]		DTGS[7：0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

注意： 根据锁定设置，AOEN、BRKP、BRKEN、OSIMI、OSIMR 和 DTGS[7：0]位均可被写保护，有必要在第一次写入 TMRx_BRKDT 寄存器时对它们进行配置。

位 15	**MOEN：** 主输出使能 （Main output enable） 一旦刹车输入有效，该位被硬件异步清'0'。根据 AOEN 位的设置值，该位可以由软件清'0'或被自动置 1。它仅对配置为输出的通道有效。 0：禁止 OC 和 OCN 输出或强制为空闲状态； 1：如果设置了相应的使能位（TMRx_CCE 寄存器的 CxEN、CxNEN 位），则开启 OC 和 OCN 输出。 有关 OC/OCN 使能的细节，参见 10.4.4.9 节：TMR1、TMR8 捕获/比较使能寄存器（TMRx_CCE）。
位 14	**AOEN：** 自动输出使能 （Automatic output enable） 0：MOEN 只能被软件置'1'； 1：MOEN 能被软件置'1'或在下一个更新事件被自动置'1'（如果刹车输入无效）。 *注：一旦 LOCK 级别（TMRx_BRKDT 寄存器中的 LOCKC 位）设为'1'，则该位不能被修改。*
位 13	**BRKP：** 刹车输入极性 （Break polarity） 0：刹车输入低电平有效； 1：刹车输入高电平有效。 *注：一旦 LOCK 级别（TMRx_BRKDT 寄存器中的 LOCKC 位）设为'1'，则该位不能被修改。* *注：任何对该位的写操作都需要一个 APB 时钟的延迟以后才能起作用。*
位 12	**BRKEN：** 刹车功能使能 （Break enable） 0：禁止刹车输入（BRK 及 CFD 时钟失效事件）； 1：开启刹车输入（BRK 及 CFD 时钟失效事件）。 *注：一旦 LOCK 级别（TMRx_BRKDT 寄存器中的 LOCKC 位）设为'1'，则该位不能被修改。* *注：任何对该位的写操作都需要一个 APB 时钟的延迟以后才能起作用。*

位 11	**OSIMR：** 运行模式下“关闭状态”选择 （Off-state selection for Run mode） 该位用于当 MOEN=1 且通道为互补输出时。没有互补输出的定时器中不存在 OSIMR 位。参考 OC/OCN 使能的详细说明（10.4.4.9 节，TMR1、TMR8 捕获/比较使能寄存器（TMRx_CCE））。 0：当定时器不工作时，禁止 OC/OCN 输出（OC/OCN 使能输出信号=0）； 1：当定时器不工作时，一旦 CxEN=1 或 CxNEN=1，首先开启 OC/OCN 并输出无效电平，然后置 OC/OCN 使能输出信号=1。 *注：一旦 LOCK 级别（TMRx_BRKDT 寄存器中的 LOCKC 位）设为 2，则该位不能被修改。*
位 10	**OSIMI：**空闲模式下“关闭状态”选择 （Off-state selection for Idle mode） 该位用于当 MOEN=0 且通道设为输出时。参考 OC/OCN 使能的详细说明（10.4.4.9 节，TMR1、TMR8 捕获/比较使能寄存器（TMRx_CCE））。 0 ：当定时器不工作时，禁止 OC/OCN 输出（OC/OCN 使能输出信号=0）； 1 ：当定时器不工作时，一旦 CxEN=1 或 CxNEN=1 ，OC/OCN 首先输出其空闲电平，然后 OC/OCN 使能输出信号=1。 *注：一旦 LOCK 级别（TMRx_BRKDT 寄存器中的 LOCKC 位）设为 2，则该位不能被修改。*
位 9：8	**LOCKC[1：0]：** 锁定设置 （LENock configuration） 该位为防止软件错误而提供写保护。 00：锁定关闭，寄存器无写保护； 01：锁定级别 1，不能写入 TMRx_BRKDT 寄存器的 DTGS、BRKEN、BRKP、AOEN 位和 TMRx_CTRL2 寄存器的 OCXIS/OCxNIS 位； 10：锁定级别 2，不能写入锁定级别 1 中的各位，也不能写入 CC 极性位（一旦相关通道通过 CxSEL 位设为输出，CC 极性位是 TMRx_CCE 寄存器的 CxP/CCNxP 位）以及 OSIMR/OSIMI 位； 11：锁定级别 3，不能写入锁定级别 2 中的各位，也不能写入 CC 控制位（一旦相关通道通过 CxSEL 位设为输出，CC 控制位是 TMRx_CCMx 寄存器的 OCxMODE/OCxPEN 位）； *注：在系统复位后，只能写一次 LOCKC 位，一旦写入 TMRx_BRKDT 寄存器，则其内容冻结直至复位。*
位 7：0	**DTGS[7：0]：** 死区发生器设置 （Dead-time generator setup） 这些位定义了插入互补输出之间的死区持续时间。假设 DT 表示其持续时间： DTGS[7：5]=0xx => DT=DTGS[7：0] ×Tdtg， Tdtg = T_{DTS}； DTGS[7：5]=10x => DT=（64+DTGS[5：0]） ×Tdtg， Tdtg = 2 × T_{DTS}； DTGS[7：5]=110 => DT=（32+DTGS[4：0]） ×Tdtg， Tdtg = 8 × T_{DTS}； DTGS[7：5]=111 => DT=（32+DTGS[4：0]） ×Tdtg， Tdtg = 16 × T_{DTS}； 例：若 T_{DTS} = 125ns（8MHZ），可能的死区时间为： 0 到 15875ns，若步长时间为 125ns； 16us 到 31750ns，若步长时间为 250ns； 32us 到 63us，若步长时间为 1us； 64us 到 126us，若步长时间为 2us； *注：一旦 LOCK 级别（TMRx_BRKDT 寄存器中的 LOCKC 位）设为 1、2 或 3，则不能修改这些位。*

10.4.4.19 TMR1、TMR8DMA控制寄存器（TMRx_DMAC）

偏移地址：0x48

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留			DBLEN[4：0]					保留			ADDR[4：0]				
res			rw	rw	rw	rw	rw	res			rw	rw	rw	rw	rw

位 15：13	保留，始终读为 0。
位 12：8	**DBLEN[4：0]：** DMA 连续传送长度 （DMA burst length） 这些位定义了 DMA 在连续模式下的传送长度（当对 TMRx_DMABA 寄存器进行读或写时，定时器则 进行一次连续传送），即：定义传输的次数，传输可以是半字（双字节）或字节： 00000：1 次传输 00001：2 次传输 00010：3 次传输 .... 10001：18 次传输 例：我们考虑这样的传输：DBLEN=7，ADDR=TMR2_CTRL1 - 如果 DBLEN=7，ADDR=TMR2_CTRL1 表示待传输数据的地址，那么传输的地址由下式给出：（TMRx_CTRL1 的地址） + ADDR + （DMA 索引），其中 DMA 索引= DBLEN 其中（TMRx_CTRL1 的地址） + ADDR 再加上 7，给出了将要写入或者读出数据的地址，这样数据的传输将发生在从地址（TMRx_CTRL1 的地址） + ADDR 开始的 7 个寄存器。 根据 DMA 数据长度的设置，可能发生以下情况： - 如果设置数据为半字（16 位），那么数据就会传输给全部 7 个寄存器。 - 如果设置数据为字节，数据仍然会传输给全部 7 个寄存器：第一个寄存器包含第一个 MSB 字节，第二个寄存器包含第一个 LSB 字节，以此类推。因此对于定时器，用户必须指定由 DMA 传输的数据宽度。
位 7：5	保留，始终读为 0。
位 4：0	**ADDR[4：0]：** DMA 基地址 （DMA base address） 这些位定义了 DMA 在连续模式下的基地址（当对 TMRx_DMAR 寄存器进行读或写时），ADDR 定义为从 TMRx_CTRL1 寄存器所在地址开始的偏移量： 00000：TMRx_CTRL1， 00001：TMRx_CTRL2， 00010：TMRx_SMC， .....

10.4.4.20 TMR1、TMR8连续模式的DMA地址（TMRx_DMABA）

偏移地址：0x4C

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DMABA[15：0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 15：0	**DMABA[15：0]：** DMA 连续传送寄存器 （DMA register for burst accesses） 对 TMRx_DMAC 寄存器的读或写会导致对以下地址所在寄存器的存取操作： TMRx_CTRL1 地址 + ADDR + DMA 索引，其中： “TMRx_CTRL1 地址”是控制寄存器 1（TMRx_CTRL1）所在的地址； “ADDR”是 TMRx_DMAC 寄存器中定义的基地址； “DMA 索引”是由 DMA 自动控制的偏移量，它取决于 TMRx_DMAC 寄存器中定义的 DBLEN。

11 看门狗

11.1 窗口看门狗（WWDG）

11.1.1 WWDG简介

AT32F403A/407 内置两个看门狗，提供了更高的安全性、时间的精确性和使用的灵活性。两个看门狗设备（独立看门狗和窗口看门狗）可用来检测和解决由软件错误引起的故障；当计数器达到给定的超时值时，触发一个中断（仅适用于窗口型看门狗）或产生系统复位。

独立看门狗（IWDG）由专用的低速时钟（LSI）驱动，即使主时钟发生故障它也仍然有效。窗口看门狗由从 APB1 时钟分频后得到的时钟驱动，通过可配置的时间窗口来检测应用程序非正常的过迟或过早的操作。

窗口看门狗通常被用来监测，由外部干扰或不可预见的逻辑条件造成的应用程序背离正常的运行序列而产生的软件故障。除非递减计数器的值在 CNTR6 位变成 0 前被刷新，看门狗电路在达到预置的时间周期时，会产生一个 MCU 复位。在递减计数器达到窗口寄存器数值之前，如果 7 位的递减 计数器数值（在控制寄存器中）被刷新，那么也将产生一个 MCU 复位。这表明递减计数器需要在一个有限的时间窗口中被刷新。

IWDG 最适合应用于那些需要看门狗作为一个在主程序之外，能够完全独立工作，并且对时间精度要求较低的场合。WWDG 最适合那些要求看门狗在精确计时窗口起作用的应用程序。

关于独立看门狗的详情，请参看第 11.2 节。

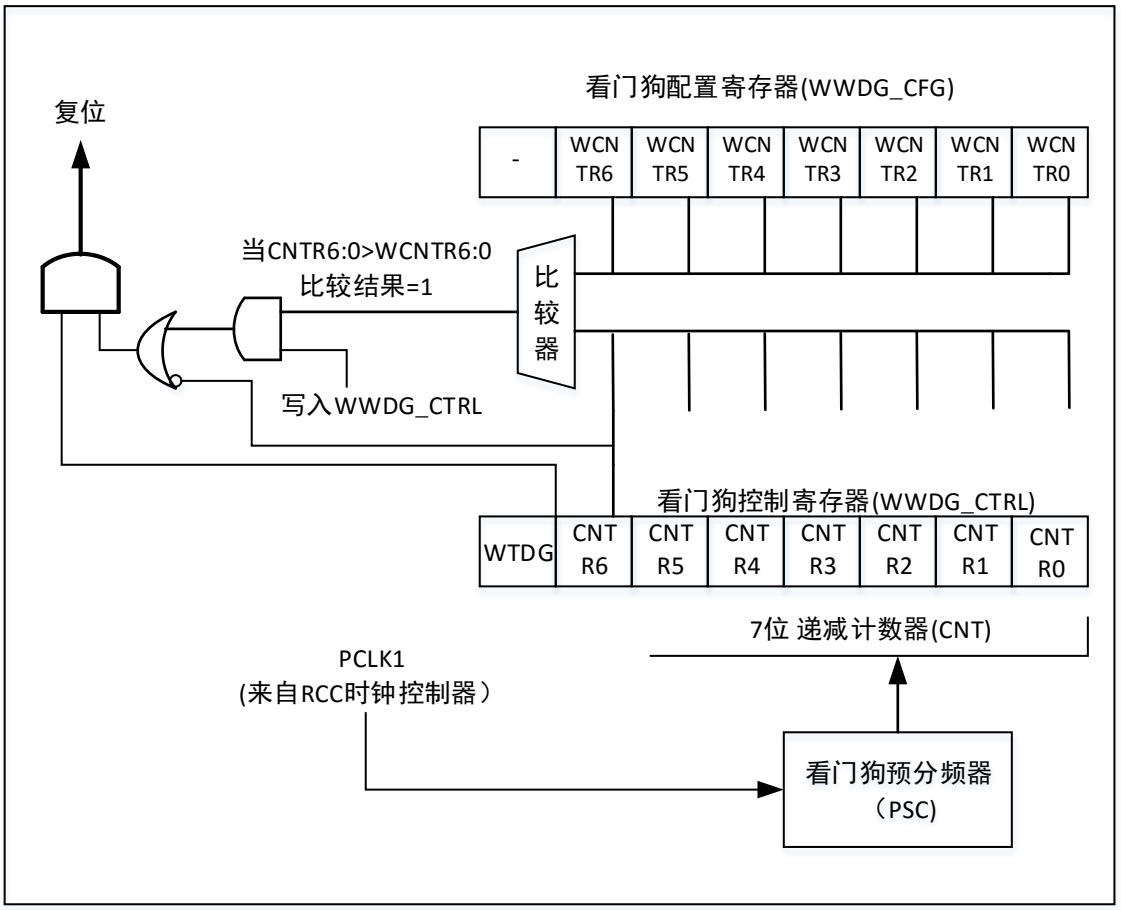
11.1.2 WWDG主要特性

- 可编程的自由运行递减计数器
- 条件复位
  - 当递减计数器的值小于0x40，（若看门狗被启动）则产生复位。
  - 当递减计数器在窗口外被重新装载，（若看门狗被启动）则产生复位。
- 如果启动了看门狗并且允许中断，当递减计数器等于0x40时产生早期唤醒中断（EWIEN），它可以被用于重装载计数器以避免WWDG复位。

11.1.3 WWDG功能描述

如果看门狗被启动（WWDG_CTRL 寄存器中的 EN 位被置’1’），并且当 7 位（CNTR[6：0]）递减计数器从 0x40 翻转到 0x3F（CNTR6 位清零）时，则产生一个复位。如果软件在计数器值大于窗口寄存器中的数值时重新装载计数器，将产生一个复位。

图 11-1 看门狗框图



应用程序在正常运行过程中必须定期地写入 WWDG_CTRL 寄存器以防止 MCU 发生复位。只有当计数器值小于窗口寄存器的值时，才能进行写操作。储存在 WWDG_CTRL 寄存器中的数值必须在 0xFF 和 0xC0 之间：

- 启动看门狗

  在系统复位后，看门狗总是处于关闭状态，设置WWDG_CTRL寄存器的EN位能够开启看门狗，随后它不能再被关闭，除非发生复位。

- 控制递减计数器

  递减计数器处于自由运行状态，即使看门狗被禁止，递减计数器仍继续递减计数。当看门狗被启用时，CNTR6位必须被设置，以防止立即产生一个复位。CNTR[5：0]位包含了看门狗产生复位之前的计时数目；复位前的延时时间在一个最小值和一个最大值之间变化，这是因为写入WWDG_CTRL寄存器时，预分频值是未知的。配置寄存器（WWDG_CFG）中包含窗口的上限值：要避免产生复位，递减计数器必须在其值小于窗口寄存器的数值并且大于0x3F时被重新装载，图11-2描述了窗口寄存器的工作过程。另一个重装载计数器的方法是利用早期唤醒中断（EWIEN）。设置WWDG_CFG寄存器中的EWIEN位开启该中断。当递减计数器到达0x40时，则产生此中断，相应的中断服务程序（ISTS）可以用来加载计数器以防止WWDG复位。在WWDG_STS寄存器中写’0’可以清除该中断。

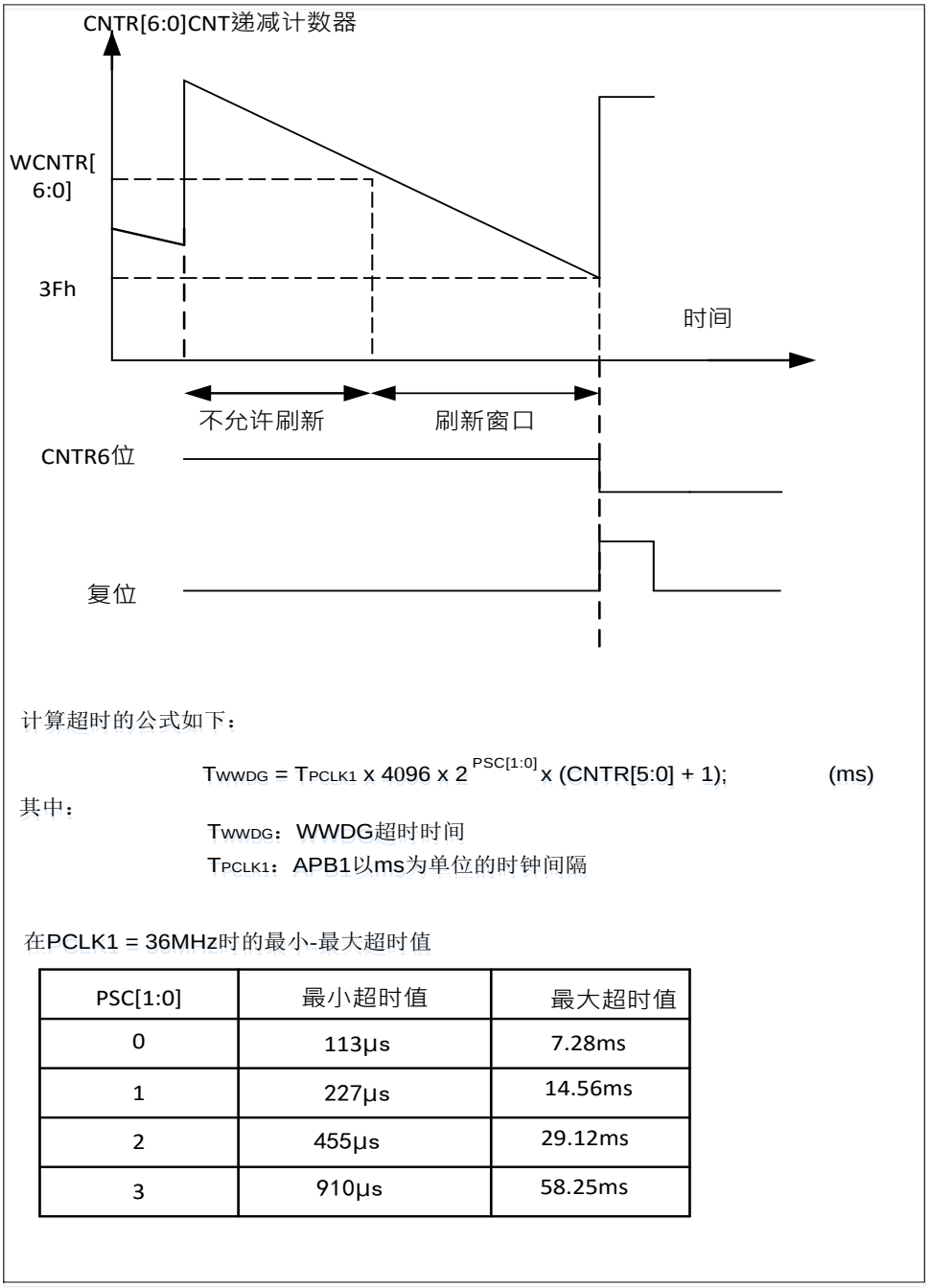
注意： 可以用 CNTR6 位产生一个软件复位（设置 EN 位为’1’，CNTR6 位为’0’）。

11.1.4 如何编写看门狗超时程序

可以使用图 11-2 提供的公式计算窗口看门狗的超时时间。

警告：当写入 WWDG_CTRL 寄存器时，始终置 CNTR6 位为’1’以避免立即产生一个复位。

图 11-2 窗口看门狗时序图



计算超时的公式如下：

$$T_{WWDG} = T_{PCLK1} \times 4096 \times 2^{PSC[1:0]} \times (CNTR[5:0] + 1); \quad (ms)$$

其中：

T_{WWDG}：WWDG超时时间

T_{PCLK1}：APB1以ms为单位的时钟间隔

在PCLK1 = 36MHz时的最小-最大超时值

PSC[1:0]	最小超时值	最大超时值
0	113μs	7.28ms
1	227μs	14.56ms
2	455μs	29.12ms
3	910μs	58.25ms

11.1.5 调试模式

当微控制器进入调试模式时（Cortex™-M4F 核心停止），根据调试模块中的 DBG_WWDG_STOP 配置位的状态，WWDG 的计数器能够继续工作或停止。详见第 22.2.2 节。

11.1.6 寄存器描述

可以用半字（16 位）或字（32 位）的方式操作这些外设寄存器。

表 11-1 WWDG寄存器映像和复位值

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	WWDG_CTRL	保留																								EN	CNTR[6：0]						
	复位值																									0	1	1	1	1	1	1	1
0x04	WWDG_CFG	保留																						EWIEN	PSC[1：0]		WCNTR[6：0]						
	复位值																							0	0	0	1	1	1	1	1	1	1
0x08	WWDG_STS	保留																															EWIF
	复位值																																0

11.1.6.1 控制寄存器（WWDG_CTRL）

地址偏移量：0x00

复位值：0x7F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															
res															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留								EN	CNTR[6：0]						
res								rs	rw						

位 31：8	保留。
位 7	**EN**： 激活位（Activation bit） 此位由软件置'1'，但仅能由硬件在复位后清'0'。当 EN=1 时，看门狗可以产生复位。 0：禁止看门狗 1：启用看门狗
位 6：0	**CNTR[6：0]**： 7 位计数器（MSB 至 LSB） （7-bit counter） 这些位用来存储看门狗的计数器值。每（4096x2^{PSC}）个 PCLK1 周期减 1。当计数器值从 40h 变为 3Fh 时（CNTR6 变成 0），产生看门狗复位。

11.1.6.2 配置寄存器（WWDG_CFG）

地址偏移量：0x04

复位值：0x7F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															
res															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留						EWIEN	PSC[1：0]		WCNTR[6：0]						
res						rs	rw		rw						

位 31：8	保留。
位 9	**EWIEN：** 提前唤醒中断（Early wakeup interrupt） 此位若置'1'，则当计数器值达到 40h，即产生中断。此中断只能由硬件在复位后清除。
位 8：7	**PSC[1：0]：** 时基 预分频器的时基可以设置如下： 00： CK 计时器时钟（PCLK1 除以 4096）除以 1 01： CK 计时器时钟（PCLK1 除以 4096）除以 2 10： CK 计时器时钟（PCLK1 除以 4096）除以 4 11： CK 计时器时钟（PCLK1 除以 4096）除以 8
位 6：0	**WCNTR[6：0]：** 7 位窗口值（7-bit window value） 这些位包含了用来与递减计数器进行比较用的窗口值。

11.1.6.3 状态寄存器（WWDG_STS）

地址偏移量：0x08

复位值：0x00

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															
res															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留															EWIF
res															rc_w 0

位 31：1	保留。
位 0	**EWIF：** 提前唤醒中断标志（Early wakeup interrupt flag） 当计数器值达到 40h 时，此位由硬件置'1'。它必须通过软件写'0'来清除。对此位写'1'无效。若中断未被使能，此位也会被置'1'。

11.2 独立看门狗（IWDG）

11.2.1 简介

独立看门狗（IWDG）由专用的低速时钟（LSI）驱动，即使主时钟发生故障它也仍然有效。

IWDG 最适合应用于那些需要看门狗作为一个在主程序之外，能够完全独立工作，并且对时间精度要求较低的场合。

11.2.2 IWDG主要性能

- 自由运行的递减计数器
- 时钟由独立的RC振荡器提供（可在停止和待机模式下工作）
- 看门狗被激活后，则在计数器计数至0x000时产生复位

11.2.3 IWDG功能描述

图 11-3 为独立看门狗模块的功能框图。在键寄存器（IWDG_KEY）中写入 0xCCCC，开始启用独立看门狗；此时计数器开始从其复位值 0xFFF 递减计数。当计数器计数到末尾 0x000 时，会产生一个复位信号（IWDG_RESET）。

无论何时，只要在键寄存器 IWDG_KEY 中写入 0xAAAA，IWDG_RLD 中的值就会被重新加载到计数器，从而避免产生看门狗复位。

11.2.3.1 硬件看门狗

如果用户在选择字节中启用了“硬件看门狗”功能，在系统上电复位后，看门狗会自动开始运行；如果在计数器计数结束前，若软件没有向键寄存器写入相应的值，则系统会产生复位。

11.2.3.2 寄存器访问保护

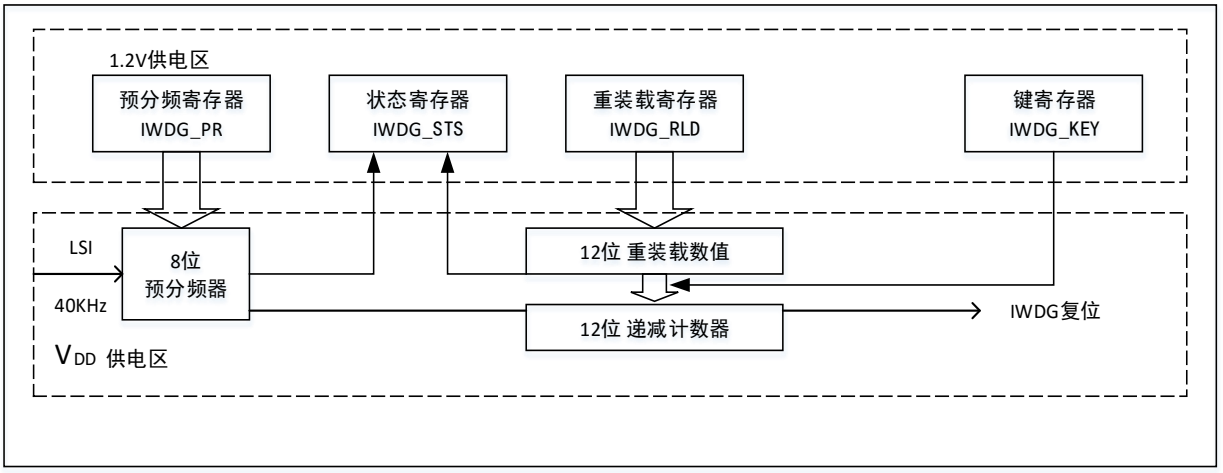
IWDG_PR 和 IWDG_RLD 寄存器具有写保护功能。要修改这两个寄存器的值，必须先向 IWDG_KEY 寄存器中写入 0x5555。以不同的值写入这个寄存器将会打乱操作顺序，寄存器将重新被保护。重装载操作（即写入 0xAAAA）也会启动写保护功能。

状态寄存器指示预分频值和递减计数器是否正在被更新。

11.2.3.3 调试模式

当微控制器进入调试模式时（Cortex™-M4F 核心停止），根据调试模块中的 DBG_IWDG_STOP 配置位的状态，IWDG 的计数器能够继续工作或停止。详见有关调试模块的章节。

图 11-3 独立看门狗框图



注意： 看门狗功能处于 VDD 供电区，即在停机和待机模式时仍能正常工作。

表 11-2 看门狗超时时间（40kHz的输入时钟（LSI））(1)

预分频系数	PR[2：0]位	最短时间（ms） RLD[11：0] = 0x000	最长时间（ms） RLD[11：0] = 0xFFF
/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

注意： 这些时间是按照 40kHz 时钟给出。实际上，MCU 内部的 RC 频率会在 30kHz 到 60kHz 之间变化。此外，即使 RC 振荡器的频率是精确的，确切的时序仍然依赖于 APB 接口时钟与 RC 振荡器时钟之间的相位差，因此总会有一个完整的 RC 周期是不确定的。

通过对 LSI 进行校准可获得相对精确的看门狗超时时间。有关 LSI 校准的问题，详见 3.2.5 节。

11.2.4 IWDG寄存器描述

关于在寄存器描述里面所用到的缩写，详见表 1-3。可以用半字（16 位）或字（32 位）的方式操作这些外设寄存器。

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	IWDG_KEY	保留																KEY[15：0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	IWDG_PR	保留																													PR[2：0]		
	复位值																														0	0	0
0x08	IWDG_RLD	保留																RLD[11：0]															
	复位值																	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x08	IWDG_STS	保留																														RLDF	PSCF
	复位值																															0	0

11.2.4.1 键寄存器（IWDG_KEY）

地址偏移：0x00

复位值：0x0000 0000（在待机模式复位）

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

位 31：16	保留，始终读为 0。
位 15：0	**KEY[15：0]**：键值（只写寄存器，读出值为 0x0000）（Key value） 软件必须以一定的间隔写入 0xAAAA，否则，当计数器为 0 时，看门狗会产生复位。 写入 0x5555 表示允许访问 IWDG_PR 和 IWDG_RLD 寄存器。（见 11.2.3.2 节）写入 0xCCCC，启动看门狗工作（若选择了硬件看门狗则不受此命令字限制）。

11.2.4.2 预分频寄存器（IWDG_PR）

地址偏移：0x04

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															
res															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留													PR[2：0]		
res													rw	rw	rw

位 31：3	保留，始终读为 0。
位 2：0	**PR[2：0]：** 预分频因子（Prescaler divider） 这些位具有写保护设置，参见 11.2.3.2 节。通过设置这些位来选择计数器时钟的预分频因子。要改变预分频因子，IWDG_STS 寄存器的 PSCF 位必须为 0。 000： 预分频因子=4 100： 预分频因子=64 001： 预分频因子=8 101： 预分频因子=128 010： 预分频因子=16 110： 预分频因子=256 011： 预分频因子=32 111： 预分频因子=256 注意：对此寄存器进行读操作，将从 VDD 电压域返回预分频值。如果写操作正在进行，则读回的值可能是无效的。因此，只有当 IWDG_STS 寄存器的 PSCF 位为 0 时，读出的值才有效。

11.2.4.3 重装载寄存器（IWDG_RLD）

地址偏移：0x08

复位值：0x0000 0FFF（待机模式时复位）

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															
res															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留				RLD[11：0]											
res				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31：12	保留，始终读为 0。
位 11：0	**RLD[11：0]：** 看门狗计数器重装载值（Watchdog counter reload value） 这些位具有写保护功能，参看 11.2.3.2 节。用于定义看门狗计数器的重装载值，每当向 IWDG_KEY 寄存器写入 0xAAAA 时，重装载值会被传送到计数器中。随后计数器从这个值开始递减计数。看门狗超时周期可通过此重装载值和时钟预分频值来计算，参照表 11-2。 只有当 IWDG_STS 寄存器中的 RLDF 位为 0 时，才能对此寄存器进行修改。 注意：对此寄存器进行读操作，将从 VDD 电压域返回重装载值。如果写操作正在进行，则读回的值可能是无效的。因此，只有当 IWDG_STS 寄存器的 RLDF 位为 0 时，读出的值才有效。

11.2.4.4 状态寄存器（IWDG_STS）

地址偏移：0x0C

复位值：0x0000 0000（待机模式时不复位）

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															
res															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留														RLDF	PSCF
res														r	r

位 31：2	保留。
位 1	**RLDF：** 看门狗计数器重装载值更新（Watchdog counter reload value update） 此位由硬件置'1'用来指示重装载值的更新正在进行中。当在 VDD 域中的重装载更新结束后，此位由硬件清'0'（最多需 5 个 40kHz 的 RC 周期）。重装载值只有在 RLDF 位被清'0'后才可更新。
位 0	**PSCF：** 看门狗预分频值更新（Watchdog prescaler value update） 此位由硬件置'1'用来指示预分频值的更新正在进行中。当在 VDD 域中的预分频值更新结束后，此位由硬件清'0'（最多需 5 个 40kHz 的 RC 周期）。预分频值只有在 PSCF 位被清'0'后才可更新。

注意： *如果在应用程序中使用了多个重装载值或预分频值，则必须在 RLDF 位被清除后才能重新改变预装载值，在 PSCF 位被清除后才能重新改变预分频值。然而，在预分频和/或重装值更新后，不必等待 RLDF 或 PSCF 复位，可继续执行下面的代码。（即是在低功耗模式下，此写操作仍会被继续执行完成。）*

12 实时时钟（RTC）

12.1 RTC简介

实时时钟是一个独立的定时器。RTC 模块拥有一组连续计数的计数器，在相应软件配置下，可提供时钟日历的功能。修改计数器的值可以重新设置系统当前的时间和日期。

RTC 模块和时钟配置系统（RCC_BDC 寄存器）处于后备区域，即在系统复位或从待机模式唤醒后，RTC 的设置和时间维持不变。

系统复位后，对后备寄存器和 RTC 的访问被禁止，这是为了防止对后备区域（BRKP）的意外写操作。执行以下操作将使能对后备寄存器和 RTC 的访问：

- 设置寄存器RCC_APB1EN的PWREN和BKPEN位，使能电源和后备接口时钟
- 设置寄存器PWR_CTRL的DBP位，使能对后备寄存器和RTC的访问。

12.2 主要特性

- 可编程的预分频系数：分频系数最高为2^20。
- 32位的可编程计数器，可用于较长时间段的测量。
- 2个分离的时钟：用于APB1接口的PCLK1和RTC时钟（RTC时钟的频率必须小于PCLK1时钟频率的四分之一以上）。
- 可以选择以下三种RTC的时钟源：
  - HSE时钟除以128
  - LSE振荡器时钟
  - LSI振荡器时钟（详见3.2.8节RTC时钟）
- 2个独立的复位类型：
  - APB1接口由系统复位；
  - RTC核心（预分频器、闹钟、计数器和分频器）只能由后备域复位（详见3.1.3节）。
- 3个专门的可屏蔽中断：
  - 闹钟中断，用来产生一个软件可编程的闹钟中断。
  - 秒中断，用来产生一个可编程的周期性中断信号（最长可达1秒）。
  - 溢出中断，指示内部可编程计数器溢出并回转为0的状态。

12.3 功能描述

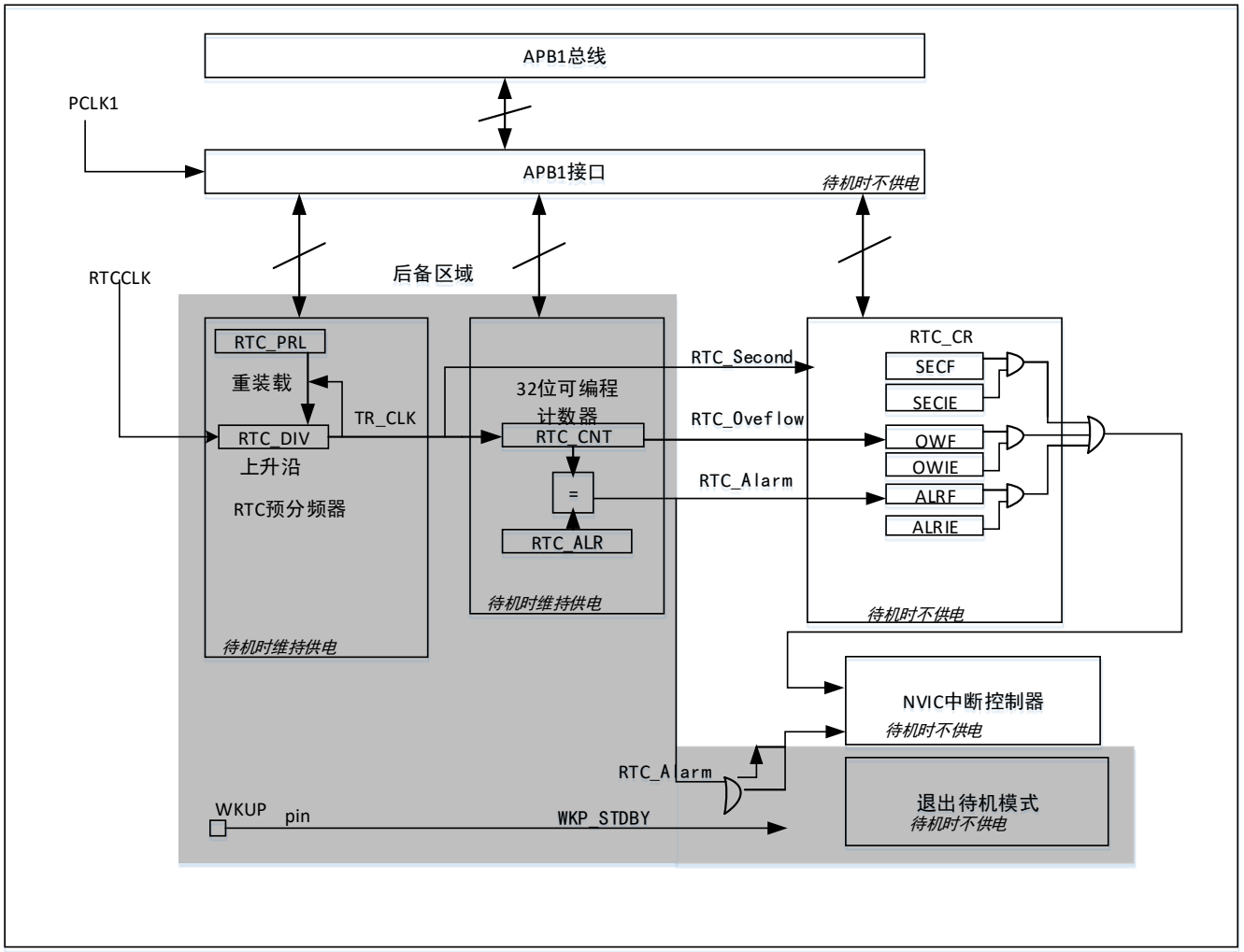
12.3.1 概述

RTC 由两个主要部分组成（参见下图）。第一部分（APB1 接口）用来和 APB1 总线相连。此单元还包含一组 16 位寄存器，可通过 APB1 总线对其进行读写操作（参见 12.4 节）。APB1 接口由 APB1 总线时钟驱动，用来与 APB1 总线接口。

另一部分（RTC 核心）由一组可编程计数器组成，分成两个主要模块。第一个模块是 RTC 的预分频模块，它可编程产生最长为 1 秒的 RTC 时间基准 LN_CLK。RTC 的预分频模块包含了一个 20 位的可编程分频器（RTC 预分频器）。如果在 RTC_CTRL 寄存器中设置了相应的允许位，则在每个 LN_CLK 周期中 RTC 产生一个中断（秒中断）。第二个模块是一个 32 位的可编程计数器，可被初始化为当前的系统时间。系

统时间按 LN_CLK 周期累加并与存储在 ALA 寄存器中的可编程时间相比较，如果 RTC_CTRL 控制寄存器中设置了相应允许位，比较匹配时将产生一个闹钟中断。

图 12-1 简化的RTC框图



12.3.2 复位过程

除了 DIV、ALA、CNT 和 DIVCNT 寄存器外，所有的系统寄存器都由系统复位或电源复位进行异步复位。

DIV、ALA、CNT 和 DIVCNT 寄存器仅能通过备份域复位信号复位，详见第 3.1.3 节。

12.3.3 读RTC寄存器

RTC 核完全独立于 RTC APB1 接口。 软件通过 APB1 接口访问 RTC 的预分频值、计数器值和闹钟值。但是，相关的可读寄存器只在与 RTC APB1 时钟进行重新同步的 RTC 时钟的上升沿被更新。RTC 标志也是如此的。

这意味着，如果 APB1 接口曾经被关闭，而读操作又是在刚刚重新开启 APB1 之后，则在第一次的内部寄存器更新之前，从 APB1 上读出的 RTC 寄存器数值可能被破坏了（通常读到 0）。下述几种情况下能够发生这种情形：

- 发生系统复位或电源复位
- 系统刚从待机模式唤醒（参见第 2.3.2 节低功耗模式。
- 系统刚从停机模式唤醒（参见第 2.3.2 节低功耗模式）。所有以上情况中，APB1 接口被禁止时（复位、无时钟或断电）RTC 核仍保持运行状态。因此，若在读取 RTC 寄存器时，RTC 的 APB1 接口曾经处于禁止状态，则软件首先必须等待 RTC_CTRLL 寄存器中的 RSYNF 位（寄存器同步标志）被硬件置’1’。

注意： RTC 的 APB1 接口不受 WFI 和 WFE 等低功耗模式的影响。

12.3.4 配置RTC寄存器

必须设置 RTC_CTRLL 寄器中的 CMF 位，使 RTC 进入配置模式后，才能写入 DIV、CNT、ALA 寄存器。

另外，对 RTC 任何寄存器的写操作，都必须在前一次写操作结束后进行。可以通过查询 RTC_CTRLL 寄存器中的 RTF 状态位，判断 RTC 寄存器是否处于更新中。仅当 RTF 状态位是’1’时，才可以写入 RTC 寄存器。

配置过程：

1. 查询 RTF 位，直到 RTF 的值变为’1’
2. 置 CMF 值为 1，进入配置模式
3. 对一个或多个 RTC 寄存器进行写操作
4. 清除 CMF 标志位，退出配置模式
5. 查询 RTF，直至 RTF 位变为’1’以确认写操作已经完成。仅当 CMF 标志位被清除时，写操作才能进行，这个过程至少需要 3 个 RTCCLK 周期。

12.3.5 RTC标志的设置

在每一个 RTC 核心的时钟周期中，更改 RTC 计数器之前设置 RTC 秒标志（PACEF）。 在计数器到达 0x0000 之前的最后一个 RTC 时钟周期中，设置 RTC 溢出标志（OVF）。 在计数器的值到达闹钟寄存器的值加 1（ALA+1）之前的 RTC 时钟周期中，设置 RTC_Alarm 和 RTC 闹钟标志（ALAF）。对 RTC 闹钟的写操作必须使用下述过程之一与 RTC 秒标志同步：

- 使用 RTC 闹钟中断，并在中断处理程序中修改 RTC 闹钟和/或 RTC 计数器。
- 等待 RTC 控制寄存器中的 PACEF 位被设置，再更改 RTC 闹钟和/或 RTC 计数器。

图 12-2 RTC秒和闹钟波形图示例，PR=0003，ALARM=00004

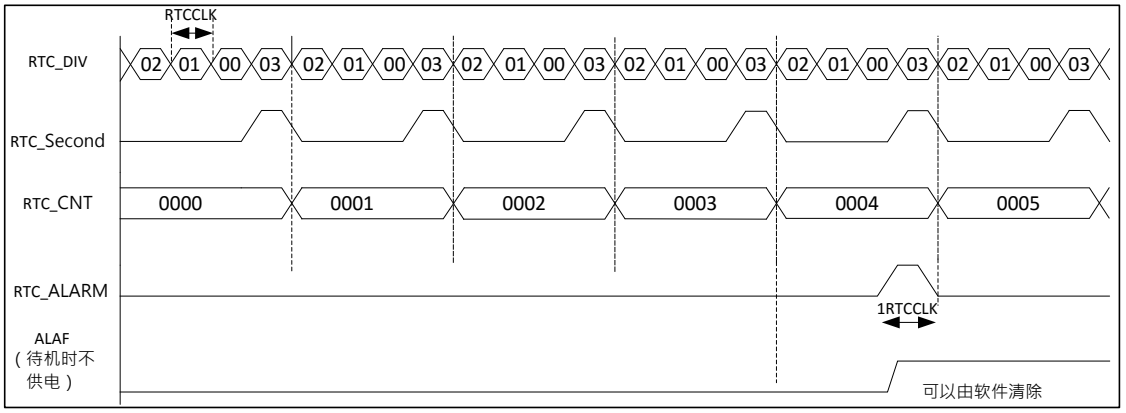
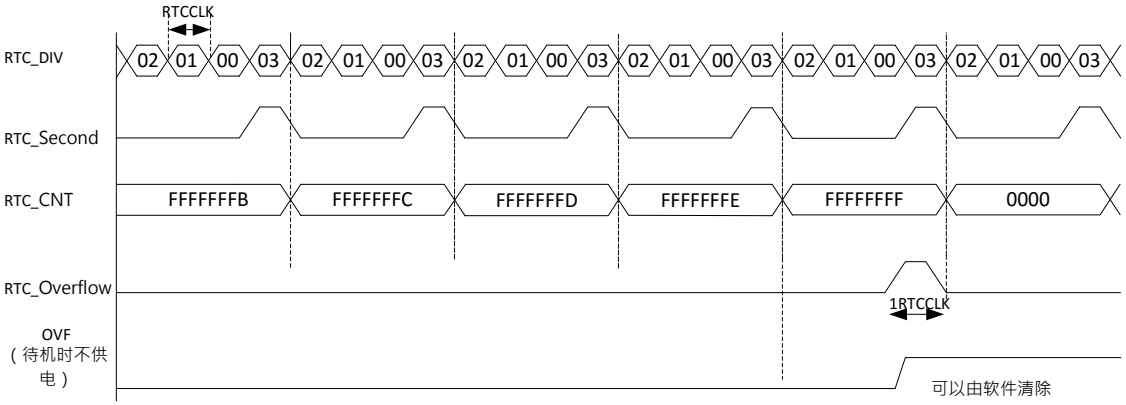


图 12-3 RTC溢出波形图示例，PR=0003



12.4 RTC寄存器描述

可以用半字（16 位）或字（32 位）的方式操作这些外设寄存器。

RTC 寄存器是 16 位可寻址寄存器，具体描述如下：

表 12-1 RTC-寄存器映像和复位值

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000h	RTC_CTRLH	保留																													OVIEN	ALAIEN	PACEIEN
	复位值																														0	0	0
004h	RTC_CTRLL	保留																										RTF	CMF	RSYNF	OVF	ALAF	PACEF
	复位值																											0	0	0	0	0	0
008h	RTC_DIVH	保留																												DIV[19：16]			
	复位值																													0	0	0	0
00Ch	RTC_DIVL	保留																DIV[15：0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
010h	RTC_DIVCNTH	保留																DIVCNT[31：16]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
014h	RTC_DIVCNTL	保留																DIVCNT [15：0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
018h	RTC_CNTH																	CNT[31：16]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
01Ch	RTC_CNTL																	CNT[15：0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
020h	RTC_ALAH																	ALA[31：16]															
	复位值																	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
024h	RTC_ALAL																	ALA[15：0]															
	复位值																	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

12.4.1 RTC控制寄存器高位（RTC_CTRLH）

地址偏移量：0x00

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留													OVIEN	ALAIEN	PACEIEN
res													rw	rw	rw

位 15：3	保留，被硬件强制为 0。
位 2	**OVIEN**：允许溢出中断位（Overflow interrupt enable） 0：屏蔽（不允许）溢出中断 1：允许溢出中断
位 1	**ALAIEN**：允许闹钟中断（Alarm interrupt enable） 0：屏蔽（不允许）闹钟中断 1：允许闹钟中断

位 0	**PACEIEN**：允许秒中断（Second interrupt enable） 0：屏蔽（不允许）秒中断 1：允许秒中断

这些位用来屏蔽中断请求。

注意： 系统复位后所有的中断被屏蔽，因此可通过写 RTC 寄存器来确保在初始化后没有挂起的中断请求。当外设正在完成前一次写操作时（标志位 RTF=0），不能对 RTC_CTRLH 寄存器进行写操作。

RTC 功能由这个控制寄存器控制。一些位的写操作必须经过一个特殊的配置过程来完成（见 12.3.4 节）。

12.4.2 RTC控制寄存器低位（RTC_CTRLL）

偏移地址：0x04

复位值：0x0020

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留										RTF	CMF	RSYNF	OVF	ALAF	PACEF
res										r	rw	rc w0	rc w0	rc w0	rc w0

位 15：6	保留，被硬件强制为 0。
位 5	**RTF**：RTC 操作关闭 （RTC operation OFF） RTC 模块利用这位来指示对其寄存器进行的最后一次操作的状态，指示操作是否完成。若此位为'0'，则表示无法对任何的 RTC 寄存器进行写操作。此位为只读位。 0：上一次对 RTC 寄存器的写操作仍在进行; 1：上一次对 RTC 寄存器的写操作已经完成。
位 4	**CMF**：配置标志 （Configuration flag） 此位必须由软件置'1'以进入配置模式，从而允许向 **CNT**、**ALA** 或 **DIVCNT** 寄存器写入数据。只有当此位在被置'1'并重新由软件清'0'后，才会执行写操作。 0：退出配置模式（开始更新 RTC 寄存器）； 1：进入配置模式。
位 3	**RSYNF**：寄存器同步标志 （Registers synchronized flag） 每当 CNT 寄存器和 **DIVCNT** 寄存器由软件更新或清'0'时，此位由硬件置'1'。在 **APB1** 复位后，或 **APB1** 时钟停止后，此位必须由软件清'0'。要进行任何的读操作之前，用户程序必须等待这位被硬件置'1'，以确保 **CNT**、**ALA** 或 **DIVCNT** 已经被同步。 0：寄存器尚未被同步； 1：寄存器已经被同步。
位 2	**OVF**：溢出标志 （Overflow flag） 当 32 位可编程计数器溢出时，此位由硬件置'1'。如果 **RTC_CTRLH** 寄存器中 **OVIEN**=1，则产生中断。此位只能由软件清'0'。对此位写'1'是无效的。 0：无溢出； 1：32 位可编程计数器溢出。
位 1	**ALAF**：闹钟标志 （Alarm flag） 当 32 位可编程计数器达到 ALA 寄存器所设置的预定值，此位由硬件置'1'。如果 **RTC_CTRLH** 寄存器中 ALAIEN=1，则产生中断。此位只能由软件清'0'。对此位写'1'是无效的。 0：无闹钟； 1：有闹钟。

位 0	**PACEF**：秒标志 （Second flag） 当 32 位可编程预分频器溢出时，此位由硬件置'1'同时 RTC 计数器加 1。因此，此标志为分辨率可编程的 RTC 计数器提供一个周期性的信号（通常为 1 秒）。如果 **RTC_CTRLH** 寄存器中 **PACEIEN**=1，则产生中断。此位只能由软件清除。对此位写'1'是无效的。 0：秒标志条件不成立； 1：秒标志条件成立。

RTC 的功能由这个控制寄存器控制。当前一个写操作还未完成时（RTF=0 时，详见 *12.3.4 节*），不能写 RTC_CTRL 寄存器。

注意： *1.任何标志位都将保持挂起状态，直到适当的 RTC_CTRL 请求位被软件复位，表示所请求的中断已经被接受。*

2.在复位时禁止所有中断，无挂起的中断请求，可以对 RTC 寄存器进行写操作。

3.当 APB1 时钟不运行时，OVF、ALAF、PACEF 和 RSYNF 位不被更新。

4.OVF、ALAF、PACEF 和 RSYNF 位只能由硬件置位，由软件来清零。

5.若 ALAF=1 且 ALAIEN=1，则允许产生 RTC 全局中断。如果在 EXTI 控制器中允许产生 EXTI 线 17 中断，则允许产生 RTC 全局中断和 RTC 闹钟中断。

6.若 ALAF=1，如果在 EXTI 控制器中设置了 EXTI 线 17 的中断模式，则允许产生 RTC 闹钟中断；如果在 EXTI 控制器中设置了 EXTI 线 17 的事件模式，则这条线上会产生一个脉冲（不会产生 RTC 闹钟中断）。

12.4.3 RTC预分频装载寄存器（RTC_DIVH/RTC_DIVL）

预分频装载寄存器用来保存 RTC 预分频器的周期计数值。它们受 RTC_CTRL 寄存器的 RTF 位保护，仅当 RTF 值为’1’时允许进行写操作。

RTC 预分频装载寄存器高位（RTC_DIVH）

偏移地址：0x08

只写（参见 12.3.4 节）

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留												DIV[19：16]			
res												w	w	w	w

位 15：4	保留
位 3：0	**DIV[19：16]**：RTC 预分频装载值高位（RTC prescaler reload value high） 根据以下公式，这些位用来定义计数器的时钟频率 $f_{LN_CLK} = f_{RTCCLK}$/（DIV[19：0]+1）

RTC 预分频装载寄存器低位（RTC_DIVL）

偏移地址：0x0C

只写（参见 12.3.4 节）

复位值：0x8000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DIV[15：0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

位 15：0	**DIV[15：0]**：RTC 预分频装载值低位 根据以下公式，这些位用来定义计数器的时钟频率： $f_{LN_CLK} = f_{RTCCLK}/（DIV[19：0]+1）$ 注：不推荐使用 0 值，否则无法正确的产生 RTC 中断和标志位。

注意：如果输入时钟频率是 32.768kHz（fRTCCLK），这个寄存器中写入 7FFFh 可获得周期为 1 秒钟的信号。

12.4.4 RTC预分频器余数寄存器（RTC_DIVCNTH / RTC_DIVCNTL）

在 LN_CLK 的每个周期里，RTC 预分频器中计数器的值都会被重新设置为 DIVCNT 寄存器的值。用户可通过读取 DIVCNT 寄存器，以获得预分频计数器的当前值，而不停止分频计数器的工作，从而获得精确的时间测量。此寄存器是只读寄存器，其值在 RTC_DIV 或 RTC_CNT 寄存器中的值发生改变后，由硬件重新装载。

RTC 预分频器余数寄存器高位（RTC_DIVCNTH）

偏移地址：0x10

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留												DIVCNT[19：16]			
res												r	r	r	r

位 15：4	保留
位 3：0	**DIVCNT[19：16]**：RTC 时钟分频器余数高位RTC clock divider high）

RTC 预分频器余数寄存器低位（RTC_DIVCNTL）

偏移地址：0x14

复位值：0x8000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DIVCNT[15：0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

位 15：0	**DIVCNT[15：0]**：RTC 时钟器余数低位（RTC clock divider low）

12.4.5 RTC计数器寄存器（RTC_CNTH / RTC_CNTL）

RTC 核有一个 32 位可编程的计数器，可通过两个 16 位的寄存器访问。计数器以预分频器产生的 LN_CLK 时间基准为参考进行计数。CNT 寄存器用来存放计数器的计数值。他们受 RTC_CTRL 的位 RTF 写保护，仅当 RTF 值’1’时，允许写操作。在高或低寄存器（RTC_CNTH 或 RTC_CNTL）上的写操作，能够直接装载到相应的可编程计数器，并且重新装载 RTC 预分频器。当进行读操作时，直接返回计数器内的计数值（系统时间）。

RTC 计数器寄存器高位（RTC_CNTH）

偏移地址：0x18

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT[31：16]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

位 15：0	**CNT[31：16]**：RTC 计数器高位（RTC counter high） 可通过读 **RTC_CNTH** 寄存器来获得 RTC 计数器当前值的高位部分。要对此寄存器进行写操作前，必须先进入配置模式（参见 12.3.4 节）。

RTC 计数器寄存器低位（RTC_CNTL）

偏移地址：0x1C

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT[15：0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

位 15：0	**CNT[15：0]**：RTC 计数器低位。 可通过读 **RTC_CNTL** 寄存器来获得 RTC 计数器当前值的低位部分。要对此寄存器进行写操作，必须先进入配置模式（参见 12.3.4 节）。

12.4.6 RTC闹钟寄存器（RTC_ALAH/RTC_ALAL）

当可编程计数器的值与 ALA 中的 32 位值相等时，即触发一个闹钟事件，并且产生 RTC 闹钟中断。此寄存器受 RTC_CTRL 寄存器里的 RTF 位写保护，仅当 RTF 值为’1’时，允许写操作。

RTC 闹钟寄存器高位（RTC_ALAH）

偏移地址：0x20

只写（参见 12.3.4 节）

复位值：0xFFFF

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ALA[31：16]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

位 15：0	**ALA[31：16]**：RTC 闹钟值高位 （RTC alarm high） 此寄存器用来保存由软件写入的闹钟时间的低位部分。要对此寄存器进行写操作，必须先进入配置模式（参见 12.3.4 节）

RTC 闹钟寄存器低位（RTC_ALAL）

偏移地址：0x24

只写（参见 12.3.4 节）

复位值：0xFFFF

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ALA[15：0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

位 15：0	**ALA[15：0]**：RTC 闹钟值低位 （RTC alarm low） 此寄存器用来保存由软件写入的闹钟时间的低位部分。要对此寄存器进行写操作，必须先进入配置模式（参见 12.3.4 节）

13 模拟/数字转换（ADC）

13.1 ADC介绍

12 位 ADC 是一种逐次逼近型模拟数字转换器。它有多达 18 个通道，可测量 16 个外部和 2 个内部信号源。各通道的 A/D 转换可以单次、连续、扫描或间断模式执行。ADC 的结果可以左对齐或右对齐方式存储在 16 位数据寄存器中。

模拟看门狗特性允许应用程序检测输入电压是否超出用户定义的高/低阈值。

ADC 的输入时钟不得超过 28 MHz，它是由 PCLK2 经分频产生，参见图 3-2。

13.2 ADC主要特征

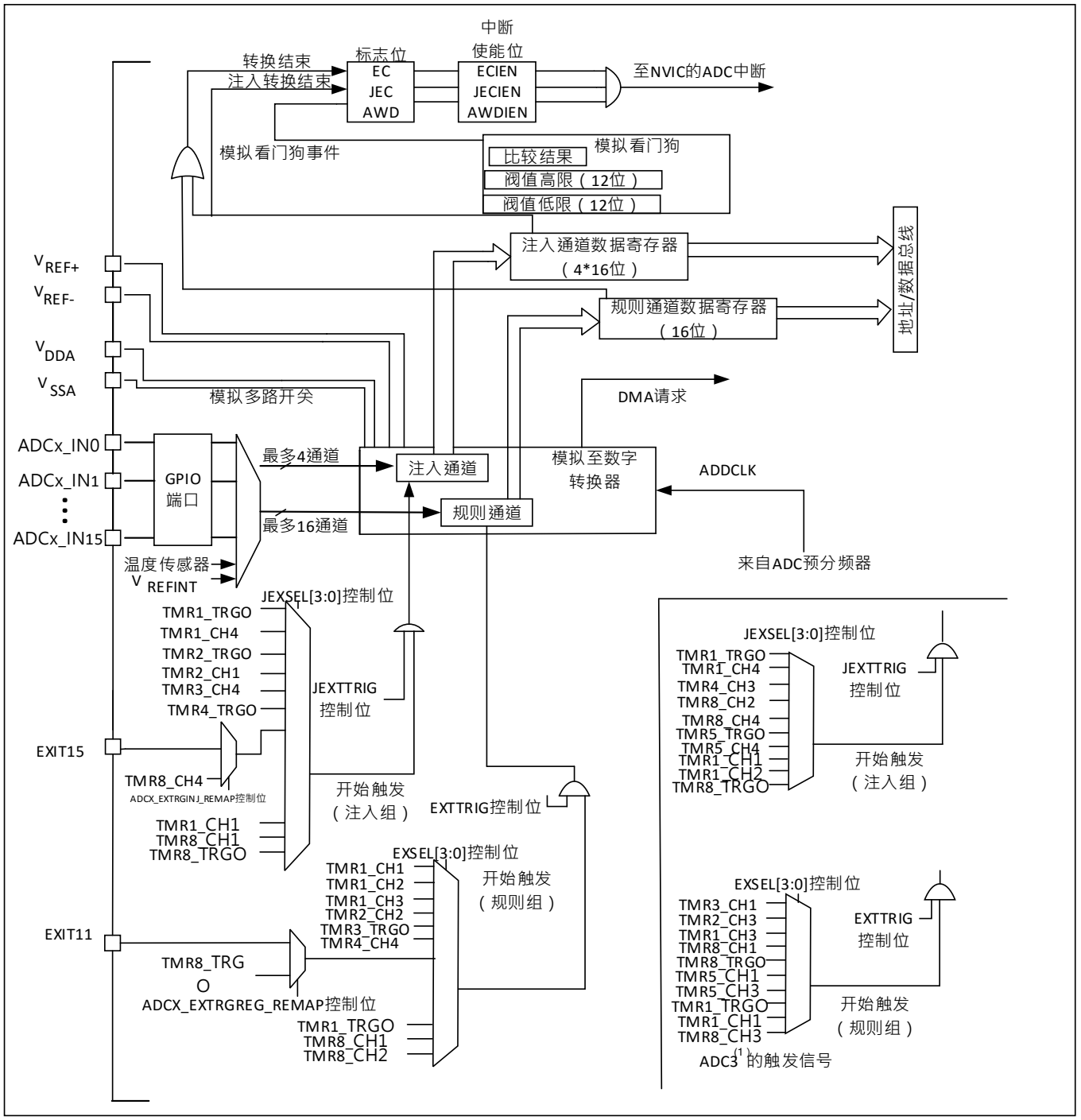
- 12位分辨率
- 转换结束、注入转换结束和发生模拟看门狗事件时产生中断
- 单次和连续转换模式
- 从通道0到通道n的自动扫描模式
- 自校准时间: 154个ADC时钟周期
- 带内嵌数据一致性的数据对齐
- 采样间隔可以按通道分别编程
- 规则转换和注入转换均有外部触发选项
- 间断模式
- 双重模式（带2个或以上ADC的器件）
- ADC转换时间
  - ADC时钟为28MHz时为0.5 μs（系统时钟为240MHz时，ADC时钟可配置为20M,转换时间为0.7 μs）
- ADC供电要求：2.6V到3.6V
- ADC输入范围：$V_{REF-} \leq V_{IN} \leq V_{REF+}$
- 规则通道转换期间有DMA请求产生

注意： 如果有 V_{REF-} 引脚（取决于封装），必须和 V_{SSA} 相连接。

13.3 ADC功能描述

下图为一个 ADC 模块的框图，表 13-1 为 ADC 引脚的说明。

图 13-1 单个ADC框图



注意： ADC3 的规则转换和注入转换触发与 ADC1 和 ADC2 的不同。

表 13-1 ADC引脚

名称	信号类型	注解
V_{REF+}	输入，模拟参考正极	ADC 使用的高端/正极参考电压，2.6V ≤ V_{REF+} ≤ V_{DDA}
V_{DDA} (1)	输入，模拟电源	等效于 V_{DD} 的模拟电源且：2.6V ≤ V_{DDA} ≤ V_{DD}（3.6V）
V_{REF-}	输入，模拟参考负极	ADC 使用的低端/负极参考电压，V_{REF-} = V_{SSA}
V_{SSA} (1)	输入，模拟电源地	等效于 V_{SS} 的模拟电源地
ADCx_IN[15：0]	模拟输入信号	16 个模拟输入通道

注意： V_{DDA} 和 V_{SSA} 应该分别连接到 V_{DD} 和 V_{SS}。

13.3.1 ADC开关控制

通过设置 ADC_CTRL2 寄存器的 ADON 位可给 ADC 上电。当第一次设置 ADON 位时，它将 ADC 从断电状态下唤醒。

ADC 上电延迟一段时间后（t_{STAB}），再次设置 ADON 位时开始进行转换。通过清除 ADON 位可以停止转换，并将 ADC 置于断电模式。在这个模式中，ADC 几乎不耗电（仅几个 μA）。

13.3.2 ADC时钟

由时钟控制器提供的 ADCCLK 时钟和 PCLK2（APB2 时钟）同步。RCC 控制器为 ADC 时钟提供一个专用的可编程预分频器，详见第 3 章复位和时钟控制（RCC）章节。

13.3.3 通道选择

有 18 个通道。可以把转换组织成两组：规则组和注入组。在任意多个通道上以任意顺序进行的一系列转换构成成组转换。例如，可以如下顺序完成转换：通道 3、通道 8、通道 2、通道 2、通道 0、通道 2、通道 2、通道 15。

- **规则组**由多达16个转换组成。规则通道和它们的转换顺序在ADC_RSQx寄存器中选择。规则组中转换的总数应写入ADC_RSQ1寄存器的LEN[3：0]位中
- **注入组**由多达4个转换组成。注入通道和它们的转换顺序在ADC_JSQ寄存器中选择。注入组里的转换总数目应写入ADC_JSQ寄存器的LEN[1：0]位中

如果 ADC_RSQx 或 ADC_JSQ 寄存器在转换期间被更改，当前的转换被清除，一个新的启动脉冲将发送到 ADC 以转换新选择的组。

温度传感器 VREFINT 内部通道

温度传感器和通道 ADC1_IN16 相连接，内部参照电压 V_{REFINT} 和 ADC1_IN17 相连接。可以按注入或规则通道对这两个内部通道进行转换。

注意： 温度传感器和 VREFINT 只能出现在主 ADC1 中。

13.3.4 单次转换模式

单次转换模式下，ADC 只执行一次转换。该模式既可通过设置 ADC_CTRL2 寄存器的 ADON 位（只适用于规则通道）启动也可通过外部触发启动（适用于规则通道或注入通道），这时 CON 位为 0。

一旦选择通道的转换完成：

- 如果一个规则通道被转换
  - 转换数据被储存在16位ADC_RDOR寄存器中
  - EC（转换结束）标志被设置
  - 如果设置了ECIEN，则产生中断
- 如果一个注入通道被转换
  - 转换数据被储存在16位的ADC_JDOR1寄存器中

- JEC（注入转换结束）标志被设置
- 如果设置了JECIEN位，则产生中断

然后 ADC 停止。

13.3.5 连续转换模式

在连续转换模式中，当前面 ADC 转换一结束马上就启动另一次转换。此模式可通过外部触发启动或通过设置 ADC_CTRL2 寄存器上的 ADON 位启动，此时 CON 位是 1。

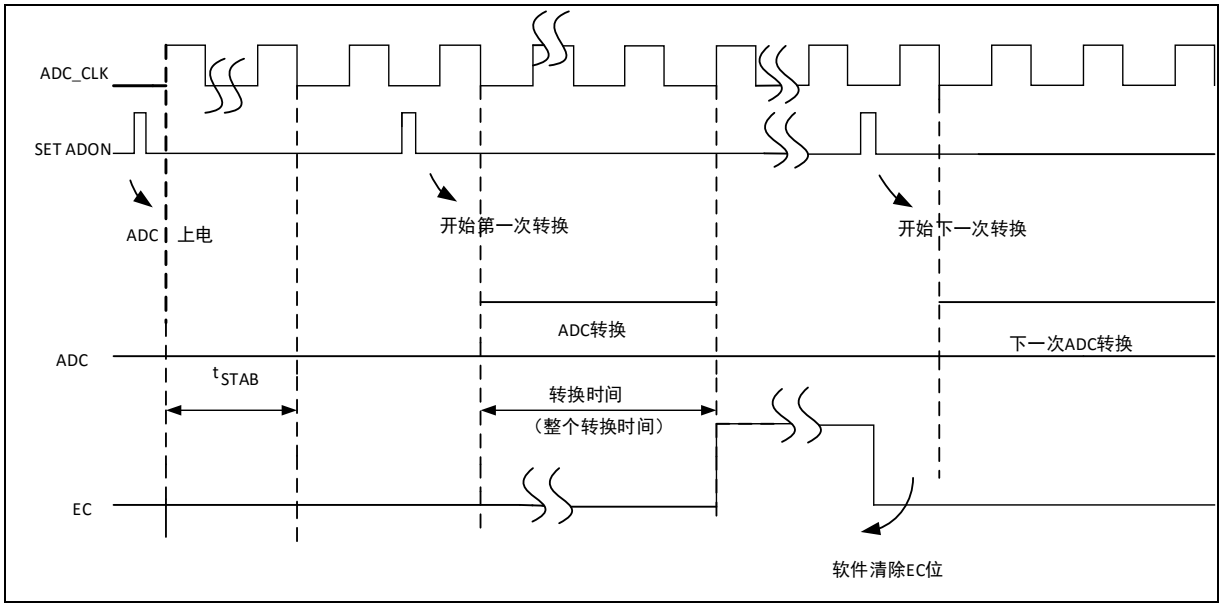
每个转换后：

- 如果一个规则通道被转换
  - 转换数据被储存在16位的ADC_RDOR寄存器中
  - EC（转换结束）标志被设置
  - 如果设置了ECIEN，则产生中断
- 如果一个注入通道被转换
  - 转换数据被储存在16位的ADC_JDOR1寄存器中
  - JEC（注入转换结束）标志被设置
  - 如果设置了JECIEN位，则产生中断

13.3.6 时序图

如下图所示，ADC 在开始精确转换前需要一个稳定时间 t_{STAB}。在开始 ADC 转换和 14 个时钟周期后，EC 标志被设置，16 位 ADC 数据寄存器包含转换的结果。

图 13-2 时序图



13.3.7 模拟看门狗

如果被 ADC 转换的模拟电压低于低阀值或高于高阀值，AWD 模拟看门狗状态位被设置。阀值位于 ADC_WHTR 和 ADC_WLTR 寄存器的最低 12 个有效位中。通过设置 ADC_CTRL1 寄存器的 AWDIEN 位以允许产生相应中断。

阀值独立于由 ADC_CTRL2 寄存器上的 DALIGN 位选择的数据对齐模式。比较是在对齐之前完成的（见 13.3.12 节）。

通过配置 ADC_CTRL1 寄存器，模拟看门狗可以作用于 1 个或多个通道，如表 13-2 所示。

图 13-3 模拟看门狗警戒区



表 13-2 模拟看门狗通道选择

模拟看门狗警戒的通道	ADC_CTRL1 寄存器控制位		
	AWDSGE 位	AWDEN 位	JAWDEN 位
无	任意值	0	0
所有注入通道	0	0	1
所有规则通道	0	1	0
所有注入和规则通道	0	1	1
单一的 (1) 注入通道	1	0	1
单一的 (1) 规则通道	1	1	0
单一的 (1) 注入或规则通道	1	1	1

注意： 由 AWDCS[4：0]位选择。

13.3.8 扫描模式

此模式用来扫描一组模拟通道。

扫描模式可通过设置 ADC_CTRL1 寄存器的 SCN 位来选择。一旦这个位被设置，ADC 扫描所有被 ADC_RSQX 寄存器（对规则通道）或 ADC_JSQ（对注入通道）选中的所有通道。在每个组的每个通道上执行单次转换。在每个转换结束时，同一组的下一个通道被自动转换。如果设置了 CON 位，转换不会在选择组的最后一个通道上停止，而是再次从选择组的第一个通道继续转换。

如果设置了 DMAEN 位，在每次 DOR 寄存器更新后，DMA 控制器把规则组通道的转换数据传输到 SRAM 中。而注入通道转换的数据总是存储在 ADC_JDORx 寄存器中。

13.3.9 注入通道管理

触发注入

清除 ADC_CTRL1 寄存器的 JAUT 位，并且设置 SCN 位，即可使用触发注入功能。

1. 利用外部触发或通过设置 ADC_CTRL2 寄存器的 ADON 位，启动一组规则通道的转换。
2. 如果在规则通道转换期间产生一外部注入触发，当前转换被复位，注入通道序列被以单次扫描方式进行转换。
3. 然后，恢复上次被中断的规则组通道转换。如果在注入转换期间产生一规则事件，注入转换不会被中断，但是规则序列将在注入序列结束后被执行。图 13-4 是其定时图。

注意： 当使用触发的注入转换时，必须保证触发事件的间隔长于注入序列。例如：序列长度为 28 个 ADC 时钟周期（即 2 个具有 1.5 个时钟间隔采样时间的转换），触发之间最小的间隔必须是 29 个 ADC 时钟周期。

自动注入

如果设置了 JAUT 位，在规则组通道之后，注入组通道被自动转换。这可以用来转换在 ADC_RSQx 和 ADC_JSQ 寄存器中设置的多至 20 个转换序列。

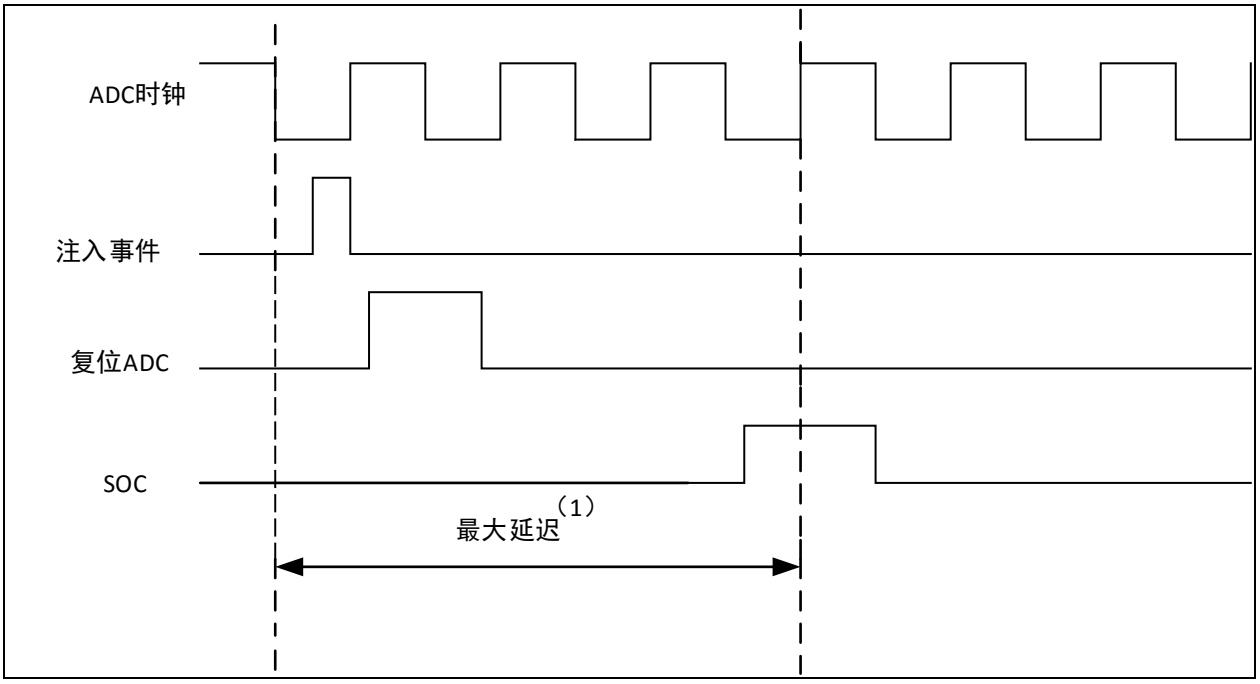
在此模式里，必须禁止注入通道的外部触发。

如果除 JAUT 位外还设置了 CON 位，规则通道至注入通道的转换序列被连续执行。

对于 ADC 时钟预分频系数为 4 至 8 时，当从规则转换切换到注入序列或从注入转换切换到规则序列时，会自动插入 1 个 ADC 时钟间隔；当 ADC 时钟预分频系数为 2 时，则有 2 个 ADC 时钟间隔的延迟。

注意： 不可能同时使用自动注入和间断模式。

图 13-4 注入转换延时



注意： 最大延迟数值请参考 AT32F403A/407 系列数据手册中有关电气特性部分。

13.3.10 间断模式

规则组

此模式通过设置 ADC_CTRL1 寄存器上的 RDISEN 位激活。它可以用来执行一个短序列的 n 次转换（n<=8），此转换是 ADC_RSQx 寄存器所选择的转换序列的一部分。数值 n 由 ADC_CTRL1 寄存器的 DISN[2：0]位给出。

一个外部触发信号可以启动 ADC_RSQx 寄存器中描述的下一轮 n 次转换，直到此序列所有的转换完成为止。总的序列长度由 ADC_RSQ1 寄存器的 LEN[3：0]定义。

举例：

n=3，被转换的通道 = 0、1、2、3、6、7、9、10

第一次触发：转换的序列为 0、1、2, 并产生 EC 事件

第二次触发：转换的序列为 3、6、7, 并产生 EC 事件

第三次触发：转换的序列为 9、10，并产生 EC 事件

第四次触发：转换的序列 0、1、2, 并产生 EC 事件

注意： 当以间断模式转换一个规则组时，转换序列结束后不自动从头开始。当所有子组被转换完成，下一次触发启动第一个子组的转换。在上面的例子中，第四次触发重新转换第一子组的通道 0、1 和 2。

注入组

此模式通过设置 ADC_CTRL1 寄存器的 JDISEN 位激活。在一个外部触发事件后，该模式按通道顺序逐个转换 ADC_JSQ 寄存器中选择的序列。

一个外部触发信号可以启动 ADC_JSQ 寄存器选择的下一个通道序列的转换，直到序列中所有的转换完成为止。总的序列长度由 ADC_JSQ 寄存器的 JLEN[1：0]位定义。

例子：

n=1，被转换的通道 = 1、2、3

第一次触发：通道 1 被转换

第二次触发：通道 2 被转换

第三次触发：通道 3 被转换，并且产生 EC 和 JEC 事件

第四次触发：通道 1 被转换

注意： 1.当完成所有注入通道转换，下个触发启动第 1 个注入通道的转换。在上述例子中，第四个触发重新转换第 1 个注入通道 1。

2.不能同时使用自动注入和间断模式。

3.必须避免同时为规则和注入组设置间断模式。间断模式只能作用于一组转换。

13.3.11 校准

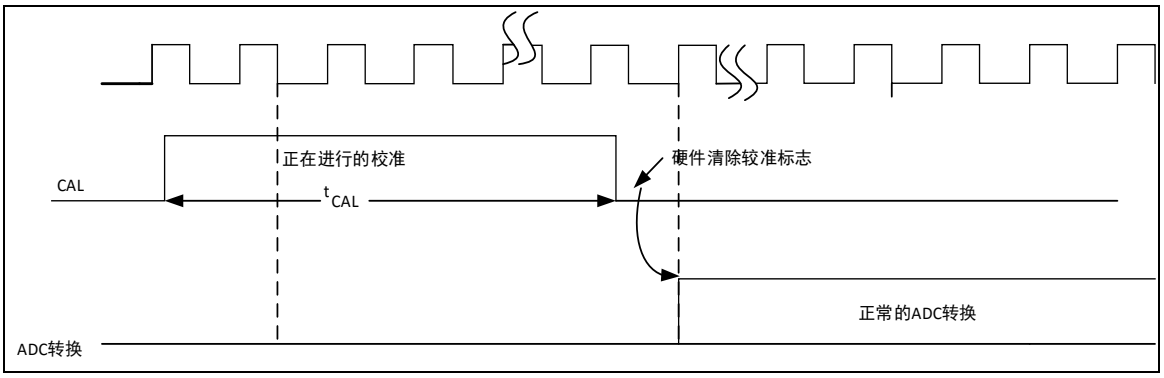
ADC 有一个内置自校准模式。校准可大幅减小因内部电容器组的变化而造成的准精度误差。在校准期间，在每个电容器上都会计算出一个误差修正码（数字值），这个码用于消除在随后的转换中每个电容器上产生的误差。

通过设置 ADC_CTRL2 寄存器的 CAL 位启动校准。一旦校准结束，CAL 位被硬件复位，可以开始正常转换。建议在上电时执行一次 ADC 校准。校准阶段结束后，校准码储存在 ADC_RDOR 中。

注意： 1.建议在每次上电后执行一次校准。

2.启动校准前，ADC 必须处于上电状态（ADON=’1’）超过至少两个 ADC 时钟周期。

图 13-5 校准时序图



13.3.12 数据对齐

ADC_CTRL2 寄存器中的 DALIGN 位选择转换后数据储存的对齐方式。数据可以左对齐或右对齐，如图 13-6 和图 13-7 所示。

注入组通道转换的数据值已经减去了在 ADC_JOFSx 寄存器中定义的偏移量，因此结果可以是一个负值。SEXT 位是扩展的符号值。

对于规则组通道，不需减去偏移值，因此只有 12 个位有效。

图 13-6 数据右对齐

注入组

SEXT	SEXT	SEXT	SEXT	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0

规则组

0	0	0	0	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0

图 13-7 数据左对齐

注入组

SEXT	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	0	0	0

规则组

D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	0	0	0	0

13.3.13 可编程的通道采样时间

ADC 使用若干个 ADC_CLK 周期对输入电压采样，采样周期数目可以通过 ADC_SMPT1 和 ADC_SMPT2 寄存器中的 SMP[2：0]位更改。每个通道可以分别用不同的时间采样。总转换时间如下计算：

T_{CONV} = 采样时间+ 12.5 个周期

例如：

当 ADCCLK=14MHz，采样时间为 1.5 周期

T_{CONV} = 1.5 + 12.5 = 14 周期 = 1 μs

13.3.14 外部触发转换

转换可以由外部事件触发（例如定时器捕获，EXTI 线）。如果设置了 EXTTRIG 控制位，则外部事件就能够触发转换。EXSEL[3：0]和 JEXSEL[3：0]控制位允许应用程序选择 16 个可能的事件中的某一个，可以触发规则和注入组的采样。

注意： 当外部触发信号被选为 ADC 规则或注入转换时，只有它的上升沿可以启动转换。

表 13-3 ADC1和ADC2用于规则通道的外部触发

触发源	连接类型	**EXSEL[3：0]**
TMR1_CC1 事件	来自片上定时器的内部信号	0000
TMR1_CC2 事件		0001
TMR1_CC3 事件		0010
TMR2_CC2 事件		0011
TMR3_TRGO 事件		0100
TMR4_CC4 事件		0101
EXTI 线 11/TMR8_TRGO 事件 (1)	外部引脚/来自片上定时器的内部信号	0110
SWSTR	软件控制位	0111
TMR1_TRGO 事件	来自片上定时器的内部信号	1101
TMR8_CC1 事件		1110
TMR8_CC2 事件		1111

注意： 1.对于规则通道，选中 EXTI 线路 11 或 TMR8_TRGO 作为外部触发事件，可以分别通过设置 ADC1 和 ADC2 的 ADC1_ETRGREG_REMAP 位和 ADC2_ETRGREG_REMAP 位实现。

表 13-4 ADC1和ADC2用于注入通道的外部触发

触发源	连接类型	**JEXSEL[3：0]**
TMR1_TRGO 事件	来自片上定时器的内部信号	0000
TMR1_CC4 事件		0001
TMR2_TRGO 事件		0010
TMR2_CC1 事件		0011
TMR3_CC4 事件		0100
TMR4_TRGO 事件		0101
EXTI 线 15/TMR8_CC4 事件 (1)	外部引脚/来自片上定时器的内部信号	0110
JSWSTR	软件控制位	0111
TMR1_CC1 事件	来自片上定时器的内部信号	1101
TMR8_CC1 事件		1110
TMR8_TRGO 事件		1111

注意： 1.对于注入通道，选中 EXTI 线路 15 和 TMR8_CC4 作为外部触发事件，可以分别通过设置 ADC1 和 ADC2 的 ADC1_ETRGINJ_REMAP 位和 ADC2_ETRGINJ_REMAP 位实现。

表 13-5 ADC3用于规则通道的外部触发

触发源	连接类型	**EXSEL[3：0]**
TMR3_CC1 事件	来自片上定时器的内部信号	0000
TMR2_CC3 事件		0001
TMR1_CC3 事件		0010
TMR8_CC1 事件		0011
TMR8_TRGO 事件		0100
TMR5_CC1 事件		0101
TMR5_CC3 事件		0110
SWSTR	软件控制位	0111
TMR1_TRGO 事件	来自片上定时器的内部信号	1101
TMR1_CC1 事件		1110
TMR8_CC3 事件		1111

表 13-6 ADC3用于注入通道的外部触发

触发源	连接类型	**JEXSEL[3：0]**
TMR1_TRGO 事件	来自片上定时器的内部信号	0000
TMR1_CC4 事件		0001

TMR4_CC3 事件		0010
TMR8_CC2 事件		0011
TMR8_CC4 事件		0100
TMR5_TRGO 事件		0101
TMR5_CC4 事件		0110
JSWSTR	软件控制位	0111
TMR1_CC1 事件	来自片上定时器的内部信号	1101
TMR1_CC2 事件		1110
TMR8_TRGO 事件		1111

软件触发事件可以通过对寄存器 ADC_CTRL2 的 SWSTR 或 JSWSTR 位置’1’产生。

规则组的转换可以被注入触发打断。

13.3.15 DMA请求

因为规则通道转换的值储存在一个仅有的数据寄存器中，所以当转换多个规则通道时需要使用 DMA，这可以避免丢失已经存储在 ADC_RDOR 寄存器中的数据。

只有在规则通道的转换结束时才产生 DMA 请求，并将转换的数据从 ADC_RDOR 寄存器传输到用户指定的目的地址。

注意：只有 ADC1 和 ADC3 拥有 DMA 功能。由 ADC2 转化的数据可以通过双 ADC 模式，利用 ADC1 的 DMA 功能传输。

13.3.16 双ADC模式

在有 2 个或以上 ADC 模块的产品中，可以使用双 ADC 模式（见图 13-8 双 ADC 框图）。

在双 ADC 模式里，根据 ADC1_CTRL1 寄存器中 DUALM[3：0]位所选的模式，转换的启动可以是 ADC1 主和 ADC2 从的交替触发或同步触发。

注意： 在双 ADC 模式里，当转换配置成由外部事件触发时，用户必须将其设置成仅触发主 ADC，从 ADC 设置成软件触发，这样可以防止意外的触发从转换。但是，主和从 ADC 的外部触发必须同时被激活。

共有 6 种可能的模式：

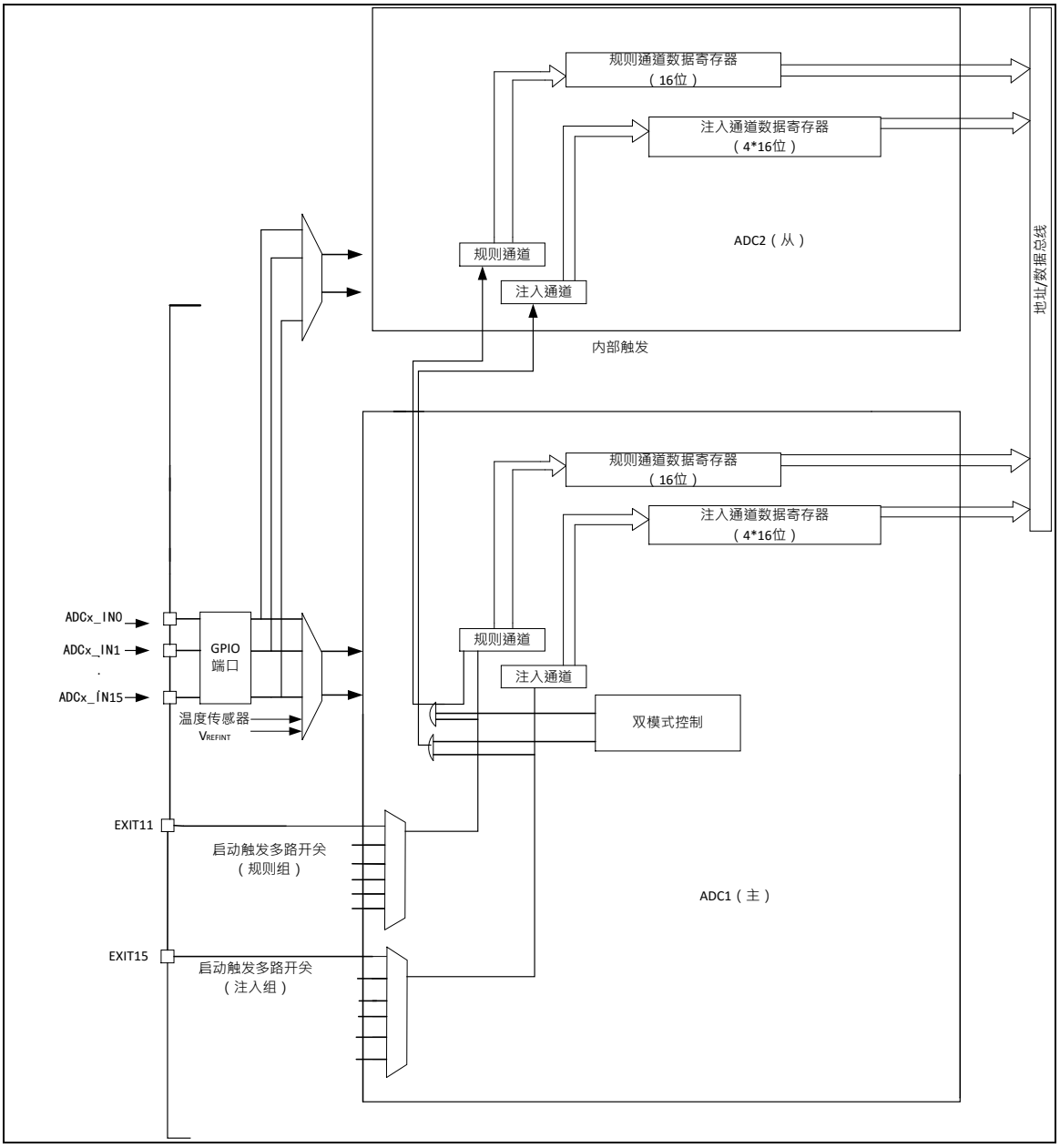
- 同步注入模式
- 同步规则模式
- 快速交叉模式
- 慢速交叉模式
- 交替触发模式
- 独立模式

还有可以用下列方式组合使用上面的模式：

- 同步注入模式 + 同步规则模式
- 同步规则模式 + 交替触发模式
- 同步注入模式 + 交叉模式

注意： 在双 ADC 模式里，为了在主数据寄存器上读取从转换数据，必须使能 DMAEN 位，即使不使用 DMA 传输规则通道数据。

图 13-8 双ADC框图（1）



注意： 1.外部触发信号作用于ADC2，但在本图中没有显示。

2.在某些双ADC模式中，在完整的ADC1数据寄存器（ADC1_RDOR）中包含了ADC1和ADC2的规则转换数据。

13.3.16.1 同步注入模式

此模式转换一个注入通道组。外部触发来自 ADC1 的注入组多路开关（由 ADC1_CTRL2 寄存器的 JEXSEL[3：0]选择），它同时给 ADC2 提供同步触发。

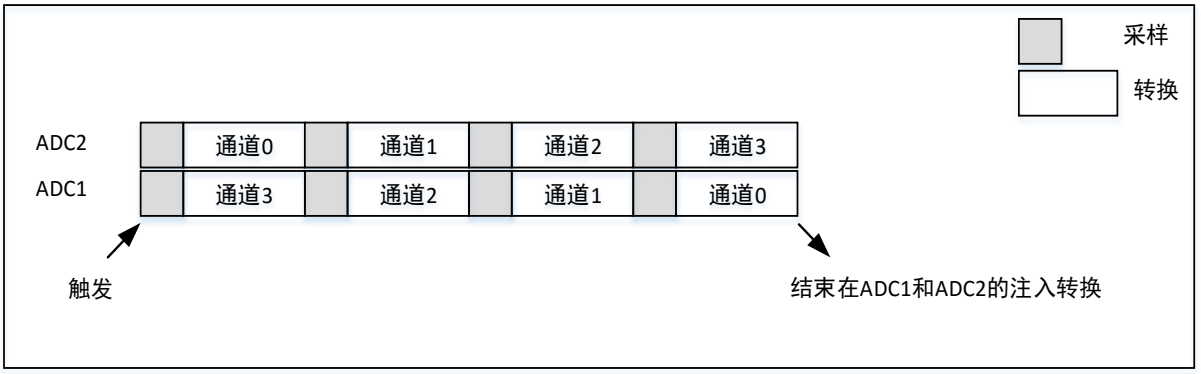
注意： 不要在 2 个 ADC 上转换相同的通道（两个 ADC 在同一个通道上的采样时间不能重叠）。

在 ADC1 或 ADC2 的转换结束时：

- 转换的数据存储在每个ADC接口的ADC_JDORx寄存器中
- 当所有ADC1/ADC2注入通道都被转换时，产生JEC中断（若任一ADC接口开放了中断）

注意： 在同步模式中，必须转换具有相同采样时间长度的序列，或保证触发的间隔比 2 个序列中较长的序列长,否则当较长序列的转换还未完成时，具有较短序列的 ADC 转换可能会被重启。

图 13-9 在4个通道上的同步注入模式



13.3.16.2 同步规则模式

此模式在规则通道组上执行。外部触发来自 ADC1 的规则组多路开关（由 ADC1_CTRL2 寄存器的 EXSEL[3：0]选择），它同时给 ADC2 提供同步触发。

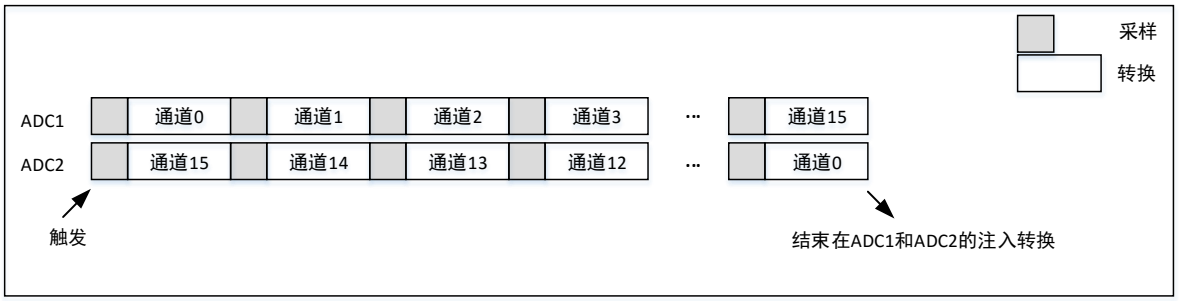
注意： 不要在 2 个 ADC 上转换相同的通道（两个 ADC 在同一个通道上的采样时间不能重叠）。

在 ADC1 或 ADC2 的转换结束时：

- 产生一个32位DMA传输请求（如果设置了DMAEN位），32位的ADC1_RDOR寄存器内容传输到SRAM中，它上半个字包含ADC2的转换数据，低半个字包含ADC1的转换数据
- 当所有ADC1/ADC2规则通道都被转换完时，产生EC中断（若任一ADC接口开放了中断）

注意： 在同步规则模式中，必须转换具有相同采样时间长度的序列，或保证触发的间隔比 2 个序列中较长的序列长，否则当较长序列的转换还未完成时，具有较短序列的 ADC 转换可能会被重启。

图 13-10 在16个通道上的同步规则模式



13.3.16.3 快速交叉模式

此模式只适用于规则通道组（通常为一个通道）。外部触发来自 ADC1 的规则通道多路开关。外部触发产生后：

- ADC2立即启动并且
- ADC1在延迟7个ADC时钟周期后启动

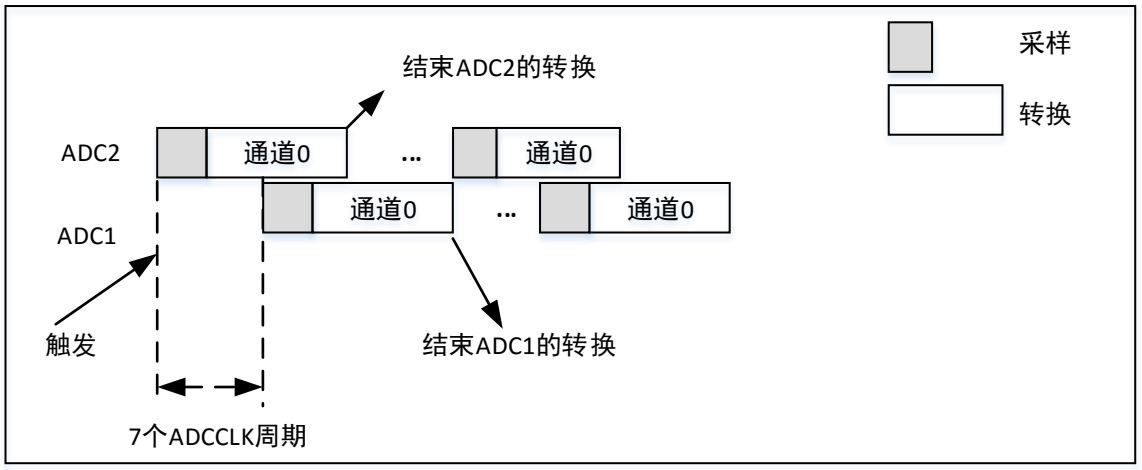
如果同时设置了 ADC1 和 ADC2 的 CON 位，所选的两个 ADC 规则通道将被连续地转换。

ADC1 产生一个 EC 中断后（由 ECIEN 使能），产生一个 32 位的 DMA 传输请求（如果设置了 DMAEN 位），ADC1_RDOR 寄存器的 32 位数据被传输到 SRAM，ADC1_RDOR 的上半个字包含 ADC2 的转换

数据，低半个字包含 ADC1 的转换数据。

注意： 最大允许采样时间<7 个 ADCCLK 周期，避免 ADC1 和 ADC2 转换相同通道时发生两个采样周期的重叠。

图 13-11 在1个通道上连续转换模式下的快速交叉模式



13.3.16.4 慢速交叉模式

此模式只适用于规则通道组（只能为一个通道）。外部触发来自 ADC1 的规则通道多路开关。外部触发产生后：

- ADC2立即启动并且
- ADC1在延迟14个ADC时钟周期后启动
- 在延迟第二次14个ADC周期后ADC2再次启动，如此循环

注意： 最大允许采样时间<14 个 ADCCLK 周期，以避免和下个转换重叠。

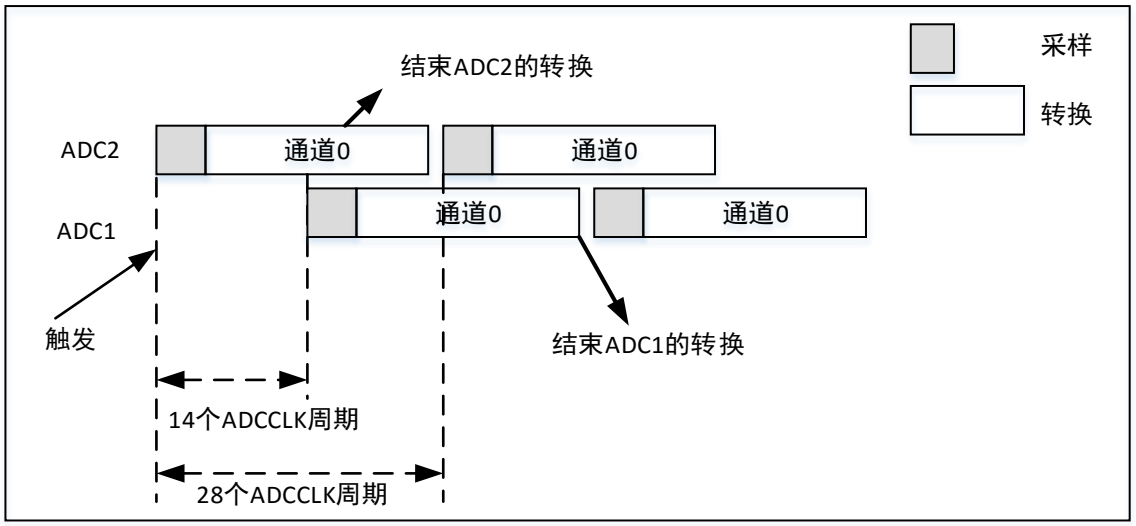
ADC1 产生一个 EC 中断后（由 ECIEN 使能），产生一个 32 位的 DMA 传输请求（如果设置了 DMAEN 位），ADC1_RDOR 寄存器的 32 位数据被传输到 SRAM，ADC1_RDOR 的上半个字包含 ADC2 的转换数据，低半个字包含 ADC1 的转换数据。

在 28 个 ADC 时钟周期后自动启动新的 ADC2 转换。

在这个模式下不能设置 CON 位，因为它将连续转换所选择的规则通道。

注意： 应用程序必须确保当使用交叉模式时，不能有注入通道的外部触发产生。

图 13-12 在1个通道上的慢速交叉模式



13.3.16.5 交替触发模式

此模式只适用于注入通道组。外部触发源来自 ADC1 的注入通道多路开关。

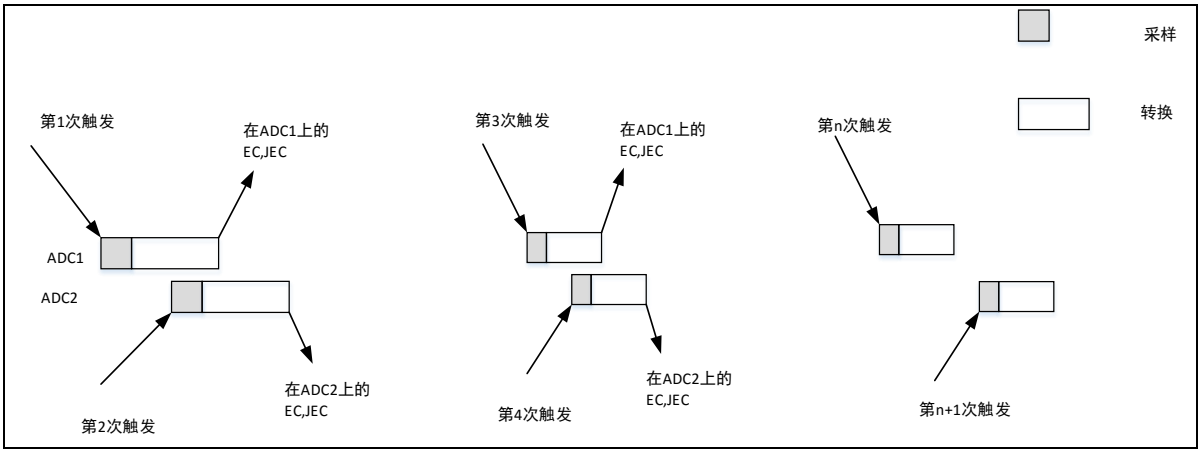
- 当第一个触发产生时，ADC1上的所有注入组通道被转换
- 当第二个触发到达时，ADC2上的所有注入组通道被转换
- 如此循环……

如果允许产生 JEC 中断，在所有 ADC1 注入组通道转换后产生一个 JEC 中断。

如果允许产生 JEC 中断，在所有 ADC2 注入组通道转换后产生一个 JEC 中断。

当所有注入组通道都转换完后，如果又有另一个外部触发，交替触发处理从转换 ADC1 注入组通道重新开始。

图 13-13 交替触发：每个ADC1的注入通道组



如果 ADC1 和 ADC2 上同时使用了注入间断模式：

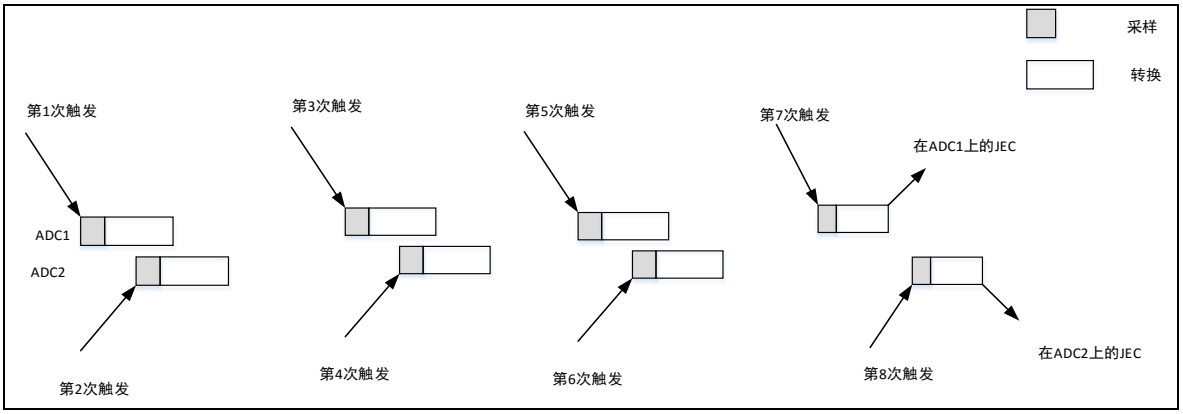
- 当第一个触发产生时，ADC1上的第一个注入通道被转换
- 当第二个触发到达时，ADC2上的第一个注入通道被转换
- 如此循环……

如果允许产生 JEC 中断，在所有 ADC1 注入组通道转换后产生一个 JEC 中断。

如果允许产生 JEC 中断，在所有 ADC2 注入组通道转换后产生一个 JEC 中断。

当所有注入组通道都转换完后，如果又有另一个外部触发，则重新开始交替触发过程。

图 13-14 交替触发：在间断模式下每个ADC上的4个注入通道



13.3.16.6 独立模式

此模式里，双 ADC 同步不工作，每个 ADC 接口独立工作。

13.3.16.7 混合的规则/注入同步模式

规则组同步转换可以被中断，以启动注入组的同步转换。

注意：在混合的规则/注入同步模式中，必须转换具有相同采样时间长度的序列，或保证触发的间隔比 2 个序列中较长的序列长，否则当较长序列的转换还未完成时，具有较短序列的 ADC 转换可能会被重启。

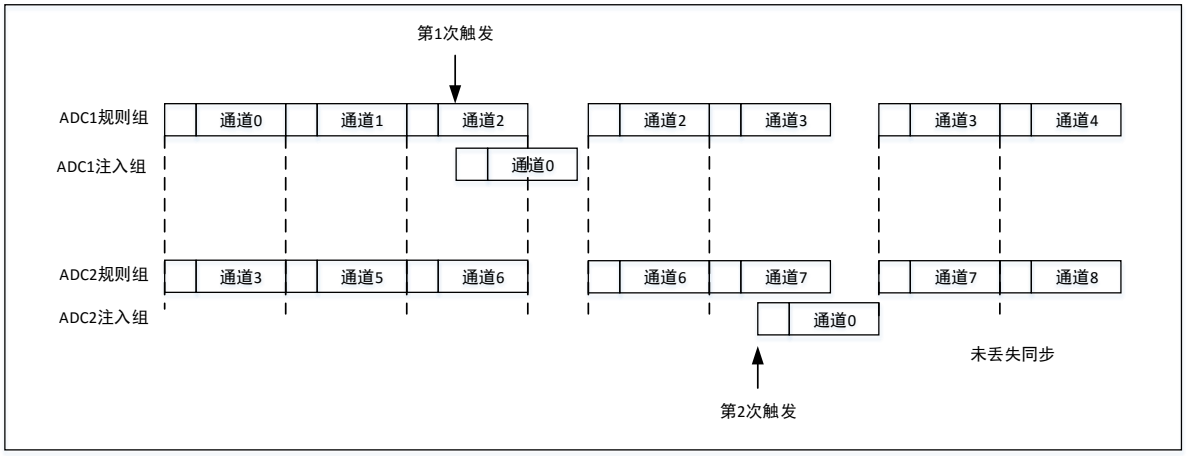
13.3.16.8 混合的同步规则+交替触发模式

规则组同步转换可以被中断，以启动注入组交替触发转换。图 13-15 显示了一个规则同步转换被交替触发所中断。

注入交替转换在注入事件到达后立即启动。如果规则转换已经在运行，为了在注入转换后确保同步，所有的 ADC（主和从）的规则转换被停止，并在注入转换结束时同步恢复。

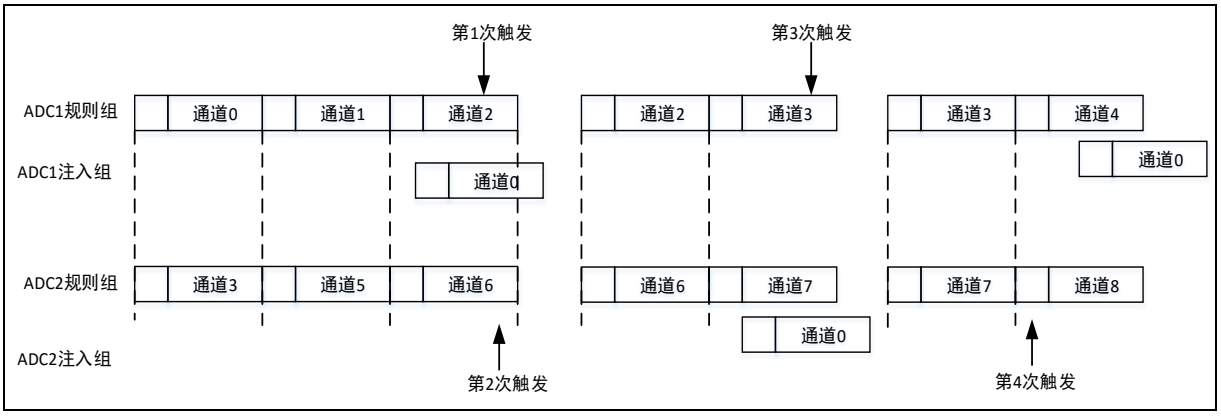
注意：在混合的同步规则+交替触发模式中，必须转换具有相同采样时间长度的序列，或保证触发的间隔比 2 个序列中较长的序列长，否则当较长序列的转换还未完成时，具有较短序列的 ADC 转换可能会被重启。

图 13-15 交替+规则同步



如果触发事件发生在一个中断了规则转换的注入转换期间，这个触发事件将被忽略。下图示出了这种情况的操作（第 2 个触发被忽略）。

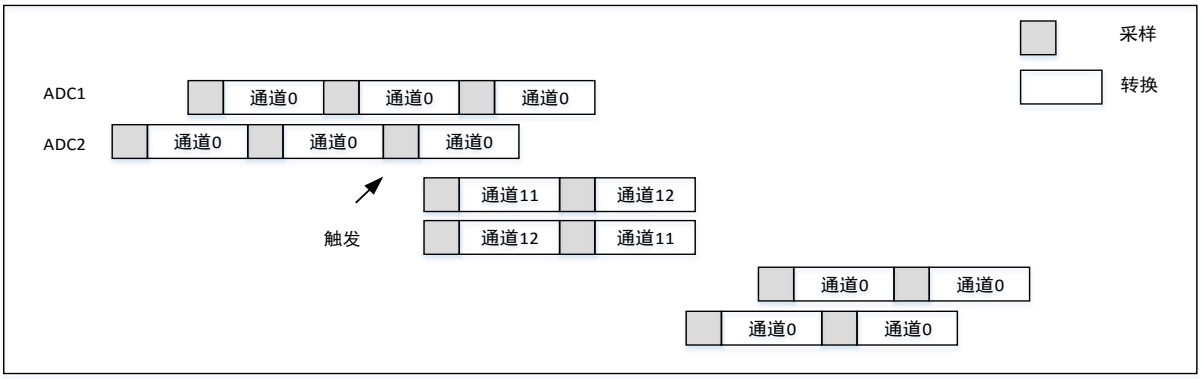
图 13-16 触发事件发生在注入转换期间



13.3.16.9 混合同步注入 +交叉模式

一个注入事件可以中断一个交叉转换。这种情况下，交叉转换被中断，注入转换被启动，在注入序列转换结束时，交叉转换被恢复。下图是这种情况的一个例子。

图 13-17 交叉的单通道转换被注入序列CH11和CH12中断



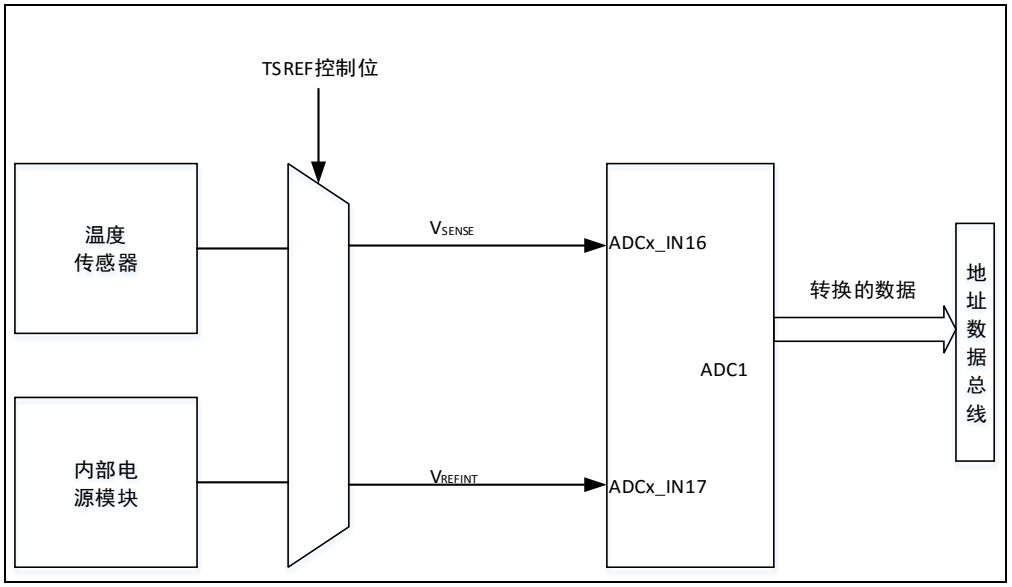
13.3.17 温度传感器

温度传感器可以用来测量器件周围的温度（T_A）。温度传感器在内部和 ADC1_IN16 输入通道相连接，此通道把传感器输出的电压转换成数字值。温度传感器模拟输入推荐采样时间是 8.6 μs。图 13-18 是温度传感器的方框图。当没有被使用时，传感器可以置于关电模式。

注意： 必须设置 TSREF 位激活内部通道：ADC1_IN16（温度传感器）和 ADC1_IN17（VREFINT）的转换。

温度传感器输出电压随温度线性变化，由于生产过程的变化，温度变化曲线的偏移在不同芯片上会有不同（最多相差 45°C）。内部温度传感器更适合于检测温度的变化，而不是测量绝对的温度。如果需要测量精确的温度，应该使用一个外置的温度传感器。

图 13-18 温度传感器和V_{REFINT}通道框图



读温度

为使用传感器：

1. 选择 ADC1_IN16 输入通道。
2. 选择采样时间为 8.6 μs。
3. 设置 ADC 控制寄存器 2（ADC_CTRL2）的 TSREF 位，以唤醒关电模式下的温度传感器。
4. 通过设置 ADON 位启动 ADC 转换（或用外部触发）。
5. 读 ADC 数据寄存器上的 V_{SENSE} 数据结果。
6. 利用下列公式得出温度。

温度（° C） = {（V_{25} - V_{SENSE}） / Avg_Slope} + 25

这里：

V_{25} = V_{SENSE} 在 25° C 时的数值

Avg_Slope ＝ 温度与 V_{SENSE} 曲线的平均斜率（单位为 mV/° C 或 μV/° C）

参考数据手册的电气特性章节中 V_{25} 和 Avg_Slope 的实际值。

注意： 传感器从关电模式唤醒后到可以输出正确水平的 VSENSE 前，有一个建立时间。ADC 在上电后也有一个建立时间，因此为了缩短延时，应该同时设置 ADON 和 TSREF 位。

13.3.18 ADC中断

规则和注入组转换结束时能产生中断，当模拟看门狗状态位被设置时也能产生中断。它们都有独立的中断使能位。

注意： ADC1 和 ADC2 的中断映射在同一个中断向量上，而 ADC3 的中断有自己的中断向量。

ADC 寄存器中有 2 个其他标志，但是它们没有相关联的中断：

- JSTR（注入组通道转换的启动）
- RSTR（规则组通道转换的启动）

表 13-7 ADC中断

中断事件	事件标志	使能控制位
规则组转换结束	EC	ECIEN
注入组转换结束	JEC	JECIEN
设置了模拟看门狗状态位	AWD	AWDIEN

13.4 ADC寄存器

必须以字（32 位）的方式操作这些外设寄存器。

表 13-8 ADC寄存器映像和复位值

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000h	ADC_STS	保留																											RSTR	JSTR	JEC	EC	AWD
	复位值																												0	0	0	0	0
004h	ADC_CTRL1	保留								AWDEN	JAWDEN	保留		DUALM[3：0]				DISN[2：0]			JDISEN	RDISEN	JAUT	AWDSGE	SCN	JECIEN	AWDIEN	ECIEN	AWDCS[4：0]				
	复位值									0	0			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
008h	ADC_CTRL2	保留						EXSEL[3]	JEXSEL[3]	TSREF	SWSTR	JSWSTR	EXTTRIG	EXSEL[2：0]			保留	JEXTTRIG	JEXSEL[2：0]			DALIGN	保留		DMAEN	保留				RSTCAL	CAL	CON	ADON
	复位值							0	0	0	0	0	0	0	0	0		0	0	0	0	0			0					0	0	0	0
偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
00Ch	ADC_SMPT1	保留								SMP17[2：0]			SMP16[2：0]			SMP15[2：0]			SMP14[2：0]			SMP13[2：0]			SMP12[2：0]			MP11[2：0]			SMP10[2：0]		
	复位值									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

| | | |

13.4.1 ADC状态寄存器（ADC_STS）

地址偏移：0x00

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															
res															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留											RSTR	JSTR	JEC	EC	AWD
res											rc w0	rc w0	rc w0	rc w0	rc w0

位 31：15	保留。必须保持为 0。
位 4	**RSTR**：规则通道开始位（Regular channel Start flag） 该位由硬件在规则通道转换开始时设置，由软件清除。 0：规则通道转换未开始； 1：规则通道转换已开始。
位 3	**JSTR**：注入通道开始位（Injected channel Start flag） 该位由硬件在注入通道组转换开始时设置，由软件清除。 0：注入通道组转换未开始； 1：注入通道组转换已开始。
位 2	**JEC**：注入通道转换结束位（Injected channel end of conversion） 该位由硬件在所有注入通道组转换结束时设置，由软件清除 0：转换未完成； 1：转换完成。
位 1	**EC**：转换结束位（End of conversion） 该位由硬件在（规则或注入）通道组转换结束时设置，由软件清除或由读取 ADC_RDOR 时清除 0：转换未完成； 1：转换完成。
位 0	**AWD**：模拟看门狗标志位（Analog watchdog flag） 该位由硬件在转换的电压值超出了 ADC_WLTR 和 ADC_WHTR 寄存器定义的范围时设置，由软件清除。 0：没有发生模拟看门狗事件； 1：发生模拟看门狗事件。

13.4.2 ADC控制寄存器1（ADC_CTRL1）

地址偏移：0x04

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留								AWDEN	JAWDEN	保留		DUALM[3：0]			
res								rw	rw	res		rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DISN[2：0]			JDISEN	RDISEN	JAUT	AWDSGE	SCN	JECIEN	AWDIEN	ECIEN	AWDCS[4：0]				
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31：24	保留。必须保持为 0。
位 23	**AWDEN**：在规则通道上开启模拟看门狗（Analog watchdog enable on regular channels） 该位由软件设置和清除。 0：在规则通道上禁用模拟看门狗； 1：在规则通道上使用模拟看门狗。
位 22	**JAWDEN**：在注入通道上开启模拟看门狗（Analog watchdog enable on injected channels） 该位由软件设置和清除。 0：在注入通道上禁用模拟看门狗； 1：在注入通道上使用模拟看门狗。
位 21：20	保留。必须保持为 0。
位 19：16	**DUALM[3：0]**：双模式选择（Dual mode selection） 软件使用这些位选择操作模式。 0000：独立模式 0001：混合的同步规则+注入同步模式 0010：混合的同步规则+交替触发模式 0011：混合同步注入+快速交叉模式 0100：混合同步注入+慢速交叉模式 0101：注入同步模式 0110：规则同步模式 0111：快速交叉模式 1000：慢速交叉模式 1001：交替触发模式 **注**：在 ADC2 和 ADC3 这些位为保留位 在双模式中，改变通道的配置会产生一个重新开始的条件，这将导致同步丢失。建议在进行任何配置改变前关闭双模式。
位 15：13	**DISN[2：0]**：间断模式通道计数（Discontinuous mode channel count） 软件通过这些位定义在间断模式下，收到外部触发后转换规则通道的数目 000：1 个通道 001：2 个通道 …… 111：8 个通道
位 12	**JDISEN**：在注入通道上的间断模式（Discontinuous mode on injected channels） 该位由软件设置和清除，用于开启或关闭注入通道组上的间断模式 0：注入通道组上禁用间断模式； 1：注入通道组上使用间断模式。
位 11	**RDISEN**：在规则通道上的间断模式（Discontinuous mode on regular 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间断模式 0：规则通道组上禁用间断模式； 1：规则通道组上使用间断模式。
位 10	**JAUT**：自动的注入通道组转换（Automatic Injected Group conversion） 该位由软件设置和清除，用于开启或关闭规则通道组转换结束后自动的注入通道组转换 0：关闭自动的注入通道组转换； 1：开启自动的注入通道组转换。
位 9	**AWDSGE**：扫描模式中在一个单一的通道上使用看门狗 （Enable the watchdog on a single channel in scan mode） 该位由软件设置和清除，用于开启或关闭由 AWDCS[4：0]位指定的通道上的模拟看门狗功能 0：在所有的通道上使用模拟看门狗； 1：在单一通道上使用模拟看门狗。

位 8	**SCN**：扫描模式（Scan mode） 该位由软件设置和清除，用于开启或关闭扫描模式。在扫描模式中，转换由 ADC_RSQx 或 ADC_JSQx 寄存器选中的通道。 0：关闭扫描模式； 1：使用扫描模式。 **注**：如果分别设置了 ECIEN或 JECIEN位，只在最后一个通道转换完毕后才会产生 EC或 JEC 中断。
位 7	**JECIEN**：允许产生注入通道转换结束中断（Interrupt enable for injected channels） 该位由软件设置和清除，用于禁止或允许所有注入通道转换结束后产生中断。 0：禁止 JEC 中断； 1：允许 JEC 中断。当硬件设置 JEC 位时产生中断。
位 6	**AWDIEN**：允许产生模拟看门狗中断（Analog watchdog interrupt enable） 该位由软件设置和清除，用于禁止或允许模拟看门狗产生中断。在扫描模式下，如果看门狗检测到超范围的数值时，只有在设置了该位时扫描才会中止。 0：禁止模拟看门狗中断； 1：允许模拟看门狗中断。
位 5	**ECIEN**：允许产生 EC 中断（Interrupt enable for EC） 该位由软件设置和清除，用于禁止或允许转换结束后产生中断。 0：禁止 EC 中断； 1：允许 EC 中断。当硬件设置 EC 位时产生中断。
位 4：0	**AWDCS[4：0]**：模拟看门狗通道选择位（Analog watchdog channel select bits） 这些位由软件设置和清除，用于选择模拟看门狗保护的输入通道。 00000：ADC 模拟输入通道 0 00001：ADC 模拟输入通道 1 …… 01111：ADC 模拟输入通道 15 10000：ADC 模拟输入通道 16 10001：ADC 模拟输入通道 17 保留所有其他数值。 **注**：ADC1 的模拟输入通道 16 和通道 17 在芯片内部分别连到了温度传感器和 V_{REFINT}。 ADC2 的模拟输入通道 16 和通道 17 在芯片内部连到了 V_{SS}。 ADC3 模拟输入通道 4-9、14、15、16、17 与 V_{SS} 相连。

13.4.3 ADC控制寄存器2（ADC_CTRL2）

地址偏移：0x08

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留						EXSEL[3]	JEXSEL[3]	TSREF	SWSTR	JSWSTR	EXTTRIG	EXSEL[2:0]			保留
res						rw	rw	rw	rw	rw	rw	rw	rw	rw	res

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
JEXTTRIG	JEXSEL[2:0]			DALIGN	保留		DMAEN	保留				RSTCAL	CAL	CON	ADON
rw	rw	rw	rw	rw	res		rw	res				rw	rw	rw	rw

位 31：24	保留。必须保持为 0。
位 25	**EXSEL[3]**，参考 EXSEL[3：0] 定义。
位 24	**JEXSEL[3]**，参考 JEXSEL[3：0]定义。

位 23	**TSREF**：温度传感器和 V_{REFINT} 使能（Temperature sensor and V_{REFINT} enable） 该位由软件设置和清除，用于开启或禁止温度传感器和 V_{REFINT} 通道。在多于 1 个 ADC 的器件中，该位仅出现在 ADC1 中。 0：禁止温度传感器和 V_{REFINT}； 1：启用温度传感器和 V_{REFINT}。
位 22	**SWSTR**：开始转换规则通道（Start conversion of regular channels） 由软件设置该位以启动转换，转换开始后硬件马上清除此位。如果在 EXSEL[3：0]位中选择了 SWSTR 为触发事件，该位用于启动一组规则通道的转换， 0：复位状态； 1：开始转换规则通道。
位 21	**JSWSTR**：开始转换注入通道（Start conversion of injected channels） 由软件设置该位以启动转换，软件可清除此位或在转换开始后硬件马上清除此位。如果在 JEXSEL[3：0]位中选择了 JSWSTR 为触发事件，该位用于启动一组注入通道的转换。 0：复位状态； 1：开始转换注入通道。
位 20	**EXTTRIG**：规则通道的外部触发转换模式（External trigger conversion mode for regular channels） 该位由软件设置和清除，用于开启或禁止可以启动规则通道组转换的外部触发事件。 0：不用外部事件启动转换； 1：使用外部事件启动转换。
位 25 位 19：17	**EXSEL[3：0]**：选择启动规则通道组转换的外部事件（External event select for regular group） 这些位选择用于启动规则通道组转换的外部事件 **ADC1 和 ADC2 的触发配置如下** 0000：定时器 1 的 CC1 事件 0001：定时器 1 的 CC2 事件 0010：定时器 1 的 CC3 事件 0011：定时器 2 的 CC2 事件 0100：定时器 3 的 TRGO 事件 0101：定时器 4 的 CC4 事件 0110：EXTI 线 11/ TMR8_TRGO 事件 0111：SWSTR 1000：保留 1001：保留 1010：保留 1011：保留 1100：保留 1101：定时器 1 的 TRGO 事件 1110：定时器 8 的 CC1 事件 1111：定时器 8 的 CC2 事件 **ADC3 的触发配置如下** 0000：定时器 3 的 CC1 事件 0001：定时器 2 的 CC3 事件 0010：定时器 1 的 CC3 事件 0011：定时器 8 的 CC1 事件 0100：定时器 8 的 TRGO 事件 0101：定时器 5 的 CC1 事件 0110：定时器 5 的 CC3 事件 0111：SWSTR 1000：保留 1001：保留 1010：保留 1011：保留 1100：保留

	1101：定时器 1 的 TRGO 事件 1110：定时器 1 的 CC1 事件 1111：定时器 8 的 CC3 事件
位 16	保留。必须保持为 0。
位 15	**JEXTTRIG**：注入通道的外部触发转换模式 （External trigger conversion mode for injected channels） 该位由软件设置和清除，用于开启或禁止可以启动注入通道组转换的外部触发事件。 0：不用外部事件启动转换； 1：使用外部事件启动转换。
位 24 位 14：12	**JEXSEL[3：0]**：选择启动注入通道组转换的外部事件 （External event select for injected group） 这些位选择用于启动注入通道组转换的外部事件。 **ADC1 和 ADC2 的触发配置如下** 0000：定时器 1 的 TRGO 事件 0001：定时器 1 的 CC4 事件 0010：定时器 2 的 TRGO 事件 0011：定时器 2 的 CC1 事件 0100：定时器 3 的 CC4 事件 0101：定时器 4 的 TRGO 事件 0110：EXTI 线 15/TMR8_CC4 事件 0111：JSWSTR 1000：保留 1001：保留 1010：保留 1011：保留 1100：保留 1101：定时器 1 的 CC1 事件 1110：定时器 8 的 CC1 事件 1111：定时器 8 的 TRGO 事件 **ADC3 的触发配置如下** 0000：定时器 1 的 TRGO 事件 0001：定时器 1 的 CC4 事件 0010：定时器 4 的 CC3 事件 0011：定时器 8 的 CC2 事件 0100：定时器 8 的 CC4 事件 0101：定时器 5 的 TRGO 事件 0110：定时器 5 的 CC4 事件 0111：JSWSTR 1000：保留 1001：保留 1010：保留 1011：保留 1100：保留 1101：定时器 1 的 CC1 事件 1110：定时器 1 的 CC2 事件 1111：定时器 8 的 TRGO 事件
位 11	**DALIGN**：数据对齐（Data alignment） 该位由软件设置和清除。参考图 13-6 和图 13-7。 0：右对齐； 1：左对齐。
位 10：9	保留。必须保持为 0。
位 8	**DMAEN**：直接存储器访问模式（Direct memory access mode） 该位由软件设置和清除。详见第 9 章 DMA 控制器（DMA）。 0：不使用 DMA 模式； 1：使用 DMA 模式。 **注**：只有 ADC1 和 ADC3 能产生 DMA 请求。

位 7：4	保留。必须保持为 0。
位 3	**RSTCAL**：复位校准（Reset 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：校准寄存器已初始化； 1：初始化校准寄存器。 **注**：如果正在进行转换时设置 RSTCAL，清除校准寄存器需要额外的周期。
位 2	**CAL**：A/D 校准（A/D Calibration） 该位由软件设置以开始校准，并在校准结束时由硬件清除。 0：校准完成； 1：开始校准。
位 1	**CON**：连续转换（Continuous conversion） 该位由软件设置和清除。如果设置了此位，则转换将连续进行直到该位被清除。 0：单次转换模式； 1：连续转换模式。
位 0	**ADON：开/关 A/D 转换器 （A/D converter ON / OFF）** 该位由软件设置和清除。当该位为'0'时，写入'1'将把 ADC 从断电模式下唤醒。 当该位为'1'时，写入'1'将启动转换。应用程序需注意，在转换器上电至转换开始有一个延迟 t_{STAB}，参见图 13-2。 0：关闭 ADC 转换/校准，并进入断电模式； 1：开启 ADC 并启动转换。 **注**：如果在这个寄存器中与 ADON 一起还有其他位被改变，则转换不被触发。这是为了防止触发错误的转换。

13.4.4 ADC采样时间寄存器1（ADC_SMPT1）

地址偏移：0x0C

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留								SMP17[2：0]			SMP16[2：0]			SMP15[2：1]	
res								rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SMP15 [0]	SMP14[2：0]			SMP13[2：0]			SMP12[2：0]			SMP11[2：0]			SMP10[2：0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31：24	保留。必须保持为 0。
位 23：0	**SMPx[2：0]**：选择通道 x 的采样时间（Channel x Sample time selection） 这些位用于独立地选择每个通道的采样时间。在采样周期中通道选择位必须保持不变。 000：1.5 周期　　100：41.5 周期 001：7.5 周期　　101：55.5 周期 010：13.5 周期　　110：71.5 周期 011：28.5 周期　　111：239.5 周期 **注**：　ADC1 的模拟输入通道 16 和通道 17 在芯片内部分别连到了温度传感器和 V_{REFINT}。 ADC2 的模拟输入通道 16 和通道 17 在芯片内部连到了 V_{ss}。 ADC3 模拟输入通道 4-9、14、15、16、17 与 V_{ss} 相连

13.4.5 ADC采样时间寄存器2（ADC_SMPT2）

地址偏移：0x10

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16

保留	SMP9[2：0]	SMP8[2：0]	SMP7[2：0]	SMP6[2：0]	SMP5[2：1]
res	rw rw rw	rw rw rw	rw rw rw	rw rw rw	rw rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SMP5[0]	SMP4[2：0]			SMP3[2：0]			SMP2[2：0]			SMP1[2：0]			SMP0[2：0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31：30	保留。必须保持为 0。
位 29：0	**SMPx[2：0]**：选择通道 x 的采样时间（Channel x Sample time selection） 这些位用于独立地选择每个通道的采样时间。在采样周期中通道选择位必须保持不变。 000：1.5 周期　100：41.5 周期 001：7.5 周期　101：55.5 周期 010：13.5 周期　110：71.5 周期 011：28.5 周期　111：239.5 周期

13.4.6 ADC注入通道数据偏移寄存器x（ADC_JOFSx）（x=1..4）

地址偏移：0x14-0x20

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															
res															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留				JOFSTx[11：0]											
res				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31：12	保留。必须保持为 0。
位 11：0	**JOFSTx[11：0]**：注入通道 x 的数据偏移（Data offset for injected channel x） 当转换注入通道时，这些位定义了用于从原始转换数据中减去的数值。转换的结果可以在 ADC_JDORx 寄存器中读出。

13.4.7 ADC看门狗高阀值寄存器（ADC_WHTR）

地址偏移：0x24

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															
res															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留				AWHT[11：0]											
res				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31：12	保留。必须保持为 0。
位 11：0	**AWHT[11：0]**：模拟看门狗高阀值（Analog watchdog high threshold） 这些位定义了模拟看门狗的阀值高限。

13.4.8 ADC看门狗低阀值寄存器（ADC_WLTR）

地址偏移：0x28

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															
res															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留				AWLT[11：0]											
res				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31：12	保留。必须保持为 0。
位 11：0	**AWLT[11：0]**：模拟看门狗低阀值（Analog watchdog low threshold） 这些位定义了模拟看门狗的阀值低限。

13.4.9 ADC规则序列寄存器1（ADC_RSQ1）

地址偏移：0x2C

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留								LEN[3：0]				SQ16[4：1]			
res								rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SQ16[0]	SQ15[4：0]					SQ14[4：0]					SQ13[4：0]				
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31：24	保留。必须保持为 0。
位 23：20	**LEN[3：0]**：规则通道序列长度 （Regular channel sequence length） 这些位由软件定义在规则通道转换序列中的通道数目。 0000：1 个转换 0001：2 个转换 …… 1111：16 个转换
位 19：15	**SQ16[4：0]**：规则序列中的第 16 个转换（16th conversion in regular sequence） 这些位由软件定义转换序列中的第 16 个转换通道的编号（0~17）。
位 14：10	**SQ15[4：0]**：规则序列中的第 15 个转换（15th conversion in regular sequence）
位 9：5	**SQ14[4：0]**：规则序列中的第 14 个转换（14th conversion in regular sequence）
位 4：0	**SQ13[4：0]**：规则序列中的第 13 个转换（13th conversion in regular sequence）

13.4.10 ADC规则序列寄存器2（ADC_RSQ2）

地址偏移：0x30

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留		SQ12[4：0]					SQ11[4：0]					SQ10[4：1]			

res		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SQ10[0]		SQ9[4：0]				SQ8[4：0]					SQ7[4：0]				
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31：30	保留。必须保持为 0。
位 29：25	**SQ12[4：0]**：规则序列中的第 12 个转换（12th conversion in regular sequence） 这些位由软件定义转换序列中的第 12 个转换通道的编号（0~17）。
位 24：20	**SQ11[4：0]**：规则序列中的第 11 个转换（11th conversion in regular sequence）
位 19：15	**SQ10[4：0]**：规则序列中的第 10 个转换（10th conversion in regular sequence）
位 14：10	**SQ9[4：0]**：规则序列中的第 9 个转换（9th conversion in regular sequence）
位 9：5	**SQ8[4：0]**：规则序列中的第 8 个转换（8th conversion in regular sequence）
位 4：0	**SQ7[4：0]**：规则序列中的第 7 个转换（7th conversion in regular sequence）

13.4.11 ADC规则序列寄存器3（ADC_RSQ3）

地址偏移：0x34

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留		SQ6[4：0]					SQ5[4：0]					SQ4[4：1]			
res		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SQ4[0]		SQ3[4：0]				SQ2[4：0]					SQ1[4：0]				
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31：30	保留。必须保持为 0。
位 29：25	**SQ6[4：0]**：规则序列中的第 6 个转换（6th conversion in regular sequence） 这些位由软件定义转换序列中的第 6 个转换通道的编号（0~17）。
位 24：20	**SQ5[4：0]**：规则序列中的第 5 个转换（5th conversion in regular sequence）
位 19：15	**SQ4[4：0]**：规则序列中的第 4 个转换（4th conversion in regular sequence）
位 14：10	**SQ3[4：0]**：规则序列中的第 3 个转换（3rd conversion in regular sequence）
位 9：5	**SQ2[4：0]**：规则序列中的第 2 个转换（2nd conversion in regular sequence）
位 4：0	**SQ1[4：0]**：规则序列中的第 1 个转换（1st conversion in regular sequence）

13.4.12 ADC注入序列寄存器（ADC_JSQ）

地址偏移：0x38

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留										JLEN[3：0]		JSQ4[4：0]			
res										rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
JSQ4[0]	JSQ3[4：0]					JSQ2[4：0]					JSQ1[4：0]				
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31：22	保留。必须保持为 0。
位 21：20	**JLEN[1：0]**：注入通道序列长度（Injected sequence length） 这些位由软件定义在注入通道转换序列中的通道数目。 00：1 个转换 01：2 个转换 10：3 个转换 11：4 个转换
位 19：15	**JSQ4[4：0]**：注入序列中的第 4 个转换（4th conversion in injected sequence） 这些位由软件定义转换序列中的第 4 个转换通道的编号（0~17）。 **注**：不同于规则转换序列，如果 JLEN[1：0]的长度小于 4，则转换的序列顺序是从（4-JLEN）开始。例如：ADC_JSQ[21：0] = 10 00011 00011 00111 00010，意味着扫描转换将按下列通道顺序转换：7、3、3，而不是 2、7、3。
位 14：10	**JSQ3[4：0]**：注入序列中的第 3 个转换（3rd conversion in injected sequence）
位 9：5	**JSQ2[4：0]**：注入序列中的第 2 个转换（2nd conversion in injected sequence）
位 4：0	**JSQ1[4：0]**：注入序列中的第 1 个转换（1st conversion in injected sequence）

13.4.13 ADC 注入数据寄存器x（ADC_JDORx）（x= 1..4）

地址偏移：0x3C － 0x48

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															
res															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
JD[15：0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

位 31：16	保留。必须保持为 0。
位 21：20	**JD[15：0]**：注入转换的数据（Injected data） 这些位为只读，包含了注入通道的转换结果。数据是左对齐或右对齐，如图 13-6 和图 13-7 所示。

13.4.14 ADC规则数据寄存器（ADC_RDOR）

地址偏移：0x4C

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADC2D[15：0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
D[15：0]															

r r r r r r r r r r r r r r r r

位 31：16	**ADC2D[15：0]**：ADC2 转换的数据（ADC2 data） - 在 ADC1 中：双模式下，这些位包含了 ADC2 转换的规则通道数据。见 13.3.16：双 ADC 模式。 - 在 ADC2 和 ADC3 中：不使用这些位。
位 15：0	**D[15：0]**：规则转换的数据（Regular data） 这些位为只读，包含了规则通道的转换结果。数据是左对齐或右对齐，如图 13-6 和图 13-7 所示。

14 数字/模拟转换（DAC）

14.1 DAC简介

数字/模拟转换模块（DAC）是 12 位数字输入，电压输出的数字/模拟转换器。DAC 可以配置为 8 位或 12 位模式，也可以与 DMA 控制器配合使用。DAC 工作在 12 位模式时，数据可以设置成左对齐或右对齐。DAC 模块有 2 个输出信道，每个信道都有单独的转换器。在双 DAC 模式下，2 个信道可以独立地进行转换，也可以同时进行转换并同步地更新 2 个通道的输出。DAC 可以通过引脚输入参考电压 V_{REF+}以获得更精确的转换结果。

14.2 DAC主要特征

- 2 个 DAC 转换器：每个转换器对应 1 个输出通道
- 8 位或者 12 位单调输出
- 12 位模式下数据左对齐或者右对齐
- 同步更新功能
- 噪声波形生成
- 三角波形生成
- 双 DAC 通道同时或者分别转换
- 每个信道都有 DMA 功能
- 外部触发转换
- 输入参考电压 V_{REF+}

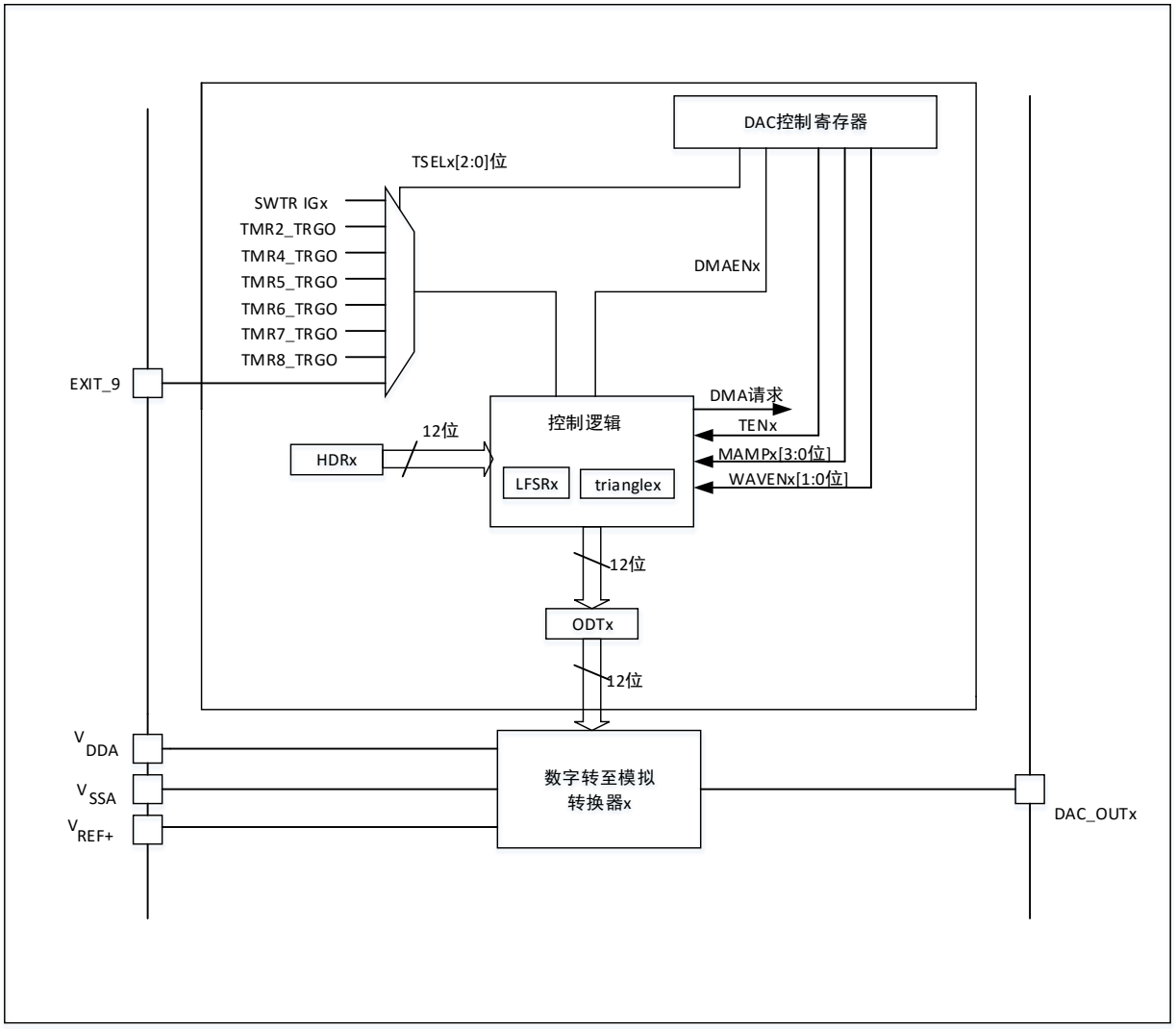
单个 DAC 通道的框图如下图，表 14-1 给出了引脚的说明。

表 14-1 DAC引脚

名称	型号类型	注释
V_{REF+}	输入，正模拟参考电压	DAC 使用的高端/正极参考电压， 2.4V ≤ V_{REF+} ≤ V_{DDA} （3.3V）
V_{DDA}	输入，模拟电源	模拟电源
V_{SSA}	输入，模拟电源地	模拟电源的地线
DAC_OUTx	模拟输出信号	DAC 通道 x 的模拟输出

注意：一旦使能 DACx 通道，相应的 GPIO 引脚（PA4 或者 PA5）就会自动与 DAC 的模拟输出相连（DAC_OUTx）。为了避免寄生的干扰和额外的功耗，引脚 PA4 或者 PA5 在之前应当设置成模拟输入（AIN）。

图 14-1 DAC信道模块框图



14.3 DAC功能描述

14.3.1 使能DAC通道

将 DAC_CTRL 寄存器的 CHENx 位置’1’即可打开对 DAC 通道 x 的供电。经过一段启动时间 tWAKEUP，DAC 通道 x 即被使能。

注意： CHENx 位只会使能 DAC 通道 x 的模拟部分，即便该位被置’0’，DAC 信道 x 的数字部分仍然工作。

14.3.2 使能DAC输出缓存

DAC 集成了 2 个输出缓存，可以用来减少输出阻抗，无需外部运放即可直接驱动外部负载。每个 DAC 通道输出缓存可以通过设置 DAC_CTRL 寄存器的 BFFx 位来使能或者关闭。

14.3.3 DAC数据格式

根据选择的配置模式，数据按照下文所述写入指定的寄存器：

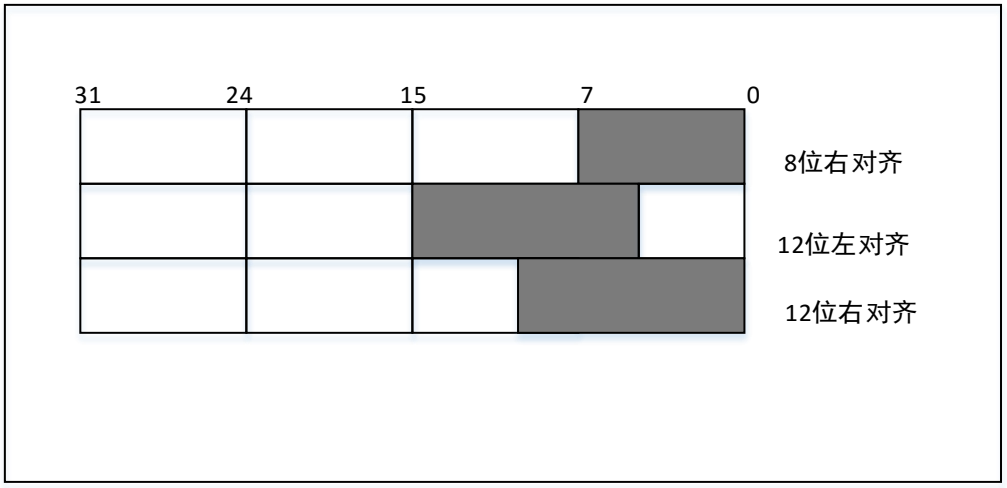
- 单DAC通道x，有3种情况：
  - 8位数据右对齐：用户须将数据写入寄存器DAC_HDR8Rx[7：0]位（实际是存入寄存器HDRx[11：4]位）
  - 12位数据左对齐：用户须将数据写入寄存器DAC_HDR12Lx[15：4]位（实际是存

入寄存器HDRx[11：0]位）

— 12位数据右对齐：用户须将数据写入寄存器DAC_HDR12Rx[11：0]位（实际是存入寄存器HDRx[11：0]位）

根据对 DAC_HDRyyyx 寄存器的操作，经过相应的移位元后，写入的资料被转存到 HDRx 寄存器中（HDRx 是内部的数据保存寄存器 x）。随后，HDRx 寄存器的内容或被自动地传送到 ODTx 寄存器，或通过软件触发或外部事件触发被传送到 ODTx 寄存器。

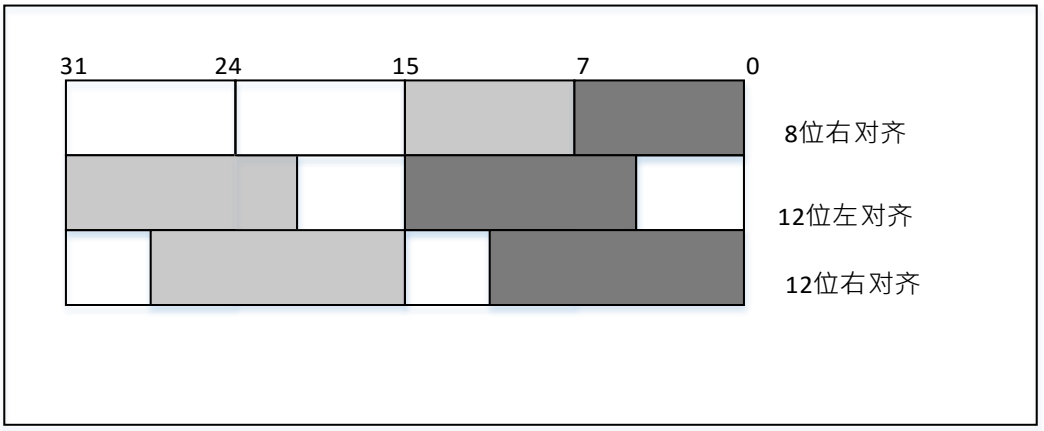
图 14-2 单DAC信道模式的数据寄存器



- 双DAC通道，有3种情况：
  — 8位数据右对齐：用户须将DAC信道1数据写入寄存器DAC_HDR8RD[7：0]位（实际是存入寄存器HDR1[11：4]位），将DAC信道2数据写入寄存器DAC_HDR8RD[15：8]位（实际是存入 寄存器HDR2[11：4]位）
  — 12位数据左对齐：用户须将DAC信道1数据写入寄存器DAC_HDR12LD[15：4]位（实际是存入寄存器HDR1[11：0]位），将DAC信道2数据写入寄存器DAC_HDR12LD[31：20]位（实际是存入寄存器HDR2[11：0]位）
  — 12位数据右对齐：用户须将DAC信道1数据写入寄存器DAC_HDR12RD[11：0]位（实际是存 入寄存器HDR1[11：0]位），将DAC信道2数据写入寄存器DAC_HDR12RD[27：16]位（实际是存入寄存器HDR2[11：0]位）

根据对 DAC_HDRyyyD 寄存器的操作，经过相应的移位元后，写入的资料被转存到 HDR1 和 HDR2 寄存器中（HDR1 和 HDR2 是内部的数据保存寄存器 x）。随后，HDR1 和 HDR2 的内容被自动地传送到 ODTx 寄存器，或通过软件触发或外部事件触发被传送到 ODTx 寄存器。

图 14-3 双DAC信道模式的数据寄存器



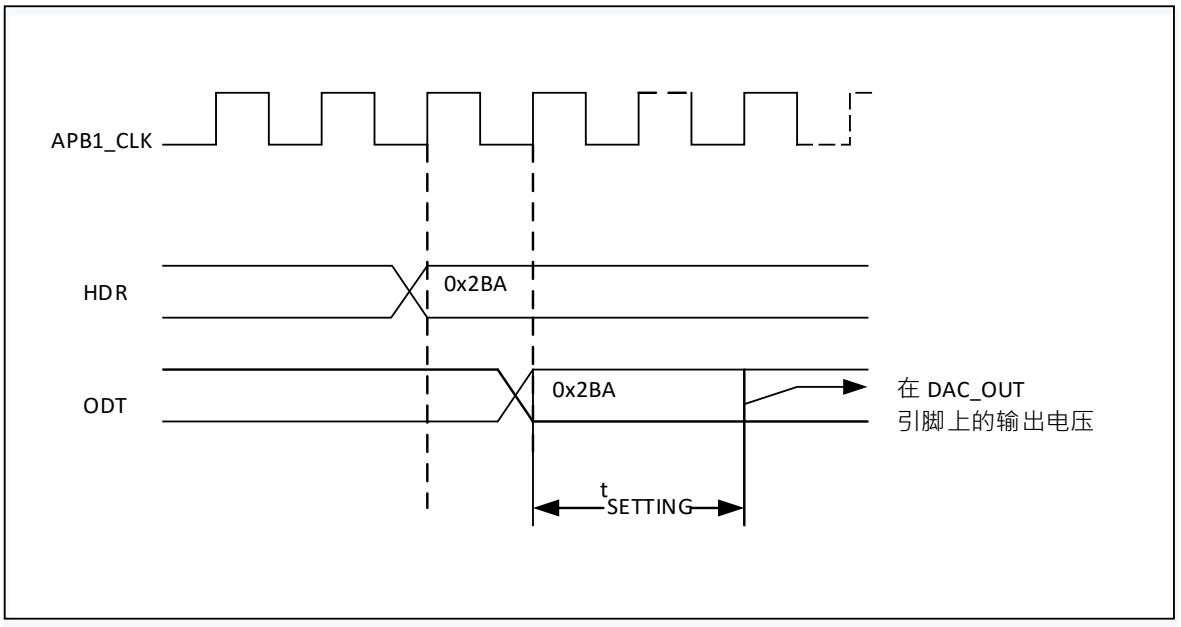
14.3.4 DAC转换

不能直接对寄存器 DAC_ODTx 写入数据，任何输出到 DAC 信道 x 的数据都必须写入 DAC_HDRx 寄存器（数据实际写入 DAC_HDR8Rx、DAC_HDR12Lx、DAC_HDR12Rx、DAC_HDR8RD、DAC_HDR12LD、或者 DAC_HDR12RD 寄存器）。

如果没有选中硬件触发（寄存器 DAC_CTRL 的 TENx 位置’0’），存入寄存器 DAC_HDRx 的数据会在一个 APB1 时钟周期后自动传至寄存器 DAC_ODTx。如果选中硬件触发（寄存器 DAC_CTRL 的 TENx 位置’1’），数据传输在触发发生以后 3 个 APB1 时钟周期后完成。

一旦数据从 DAC_HDRx 寄存器装入 DAC_ODTx 寄存器，在经过时间 $t_{SETTLING}$ 之后，输出即有效，这段时间的长短依电源电压和模拟输出负载的不同会有所变化。

图 14-4 TEN=0触发禁止时转换的时间框图



14.3.5 DAC输出电压

数字输入经过 DAC 被线性地转换为模拟电压输出，其范围为 0 到 V_{REF+}。任一 DAC 信道引脚上的输出电压满足下面的关系：

DAC 输出 $= V_{REF}$ x （ODT / 4095）。

14.3.6 选择DAC触发

如果 TENx 位被置 1，DAC 转换可以由某外部事件触发（定时器计数器、外部中断线）。配置控制位 TGSELx[2：0]可以选择 8 个触发事件之一触发 DAC 转换。

表 14-2 外部触发

触发源	类型	TGSELx[2：0]
定时器 6 TRGO 事件	来自片上定时器的内部信号	000
定时器 8 TRGO 事件		001
定时器 7 TRGO 事件		010
定时器 5 TRGO 事件		011
定时器 2 TRGO 事件		100
定时器 4 TRGO 事件		101
EXTI 线路 9	外部引脚	110

SWTRG（软件触发）	软件控制位	111

每次 DAC 接口侦测到来自选中的定时器 TRGO 输出，或者外部中断线 9 的上升沿，最近存放在寄 存器 DAC_HDRx 中的数据会被传送到寄存器 DAC_ODTx 中。在 3 个 APB1 时钟周期后，寄存器 DAC_ODTx 更新为新值。

如果选择软件触发，一旦 SWTRG 位置’1’，转换即开始。在数据从 DAC_HDRx 寄存器传送到 DAC_ODTx 寄存器后，SWTRG 位由硬件自动清’0’。

注意： *1.不能在 CHENx 为’1’时改变 TGSELx[2：0]位。*

2.如果选择软件触发，数据从寄存器 DAC_HDRx 传送到寄存器 DAC_ODTx 只需要一个 APB1 时 钟周期。

14.3.7 DMA请求

任一 DAC 通道都具有 DMA 功能。2 个 DMA 通道可分别用于 2 个 DAC 通道的 DMA 请求。如果 DMAENx 位置’1’，一旦有外部触发发生，则产生一个 DMA 请求，然后 DAC_HDRx 寄存器的数据被传送到 DAC_ODTx 寄存器。

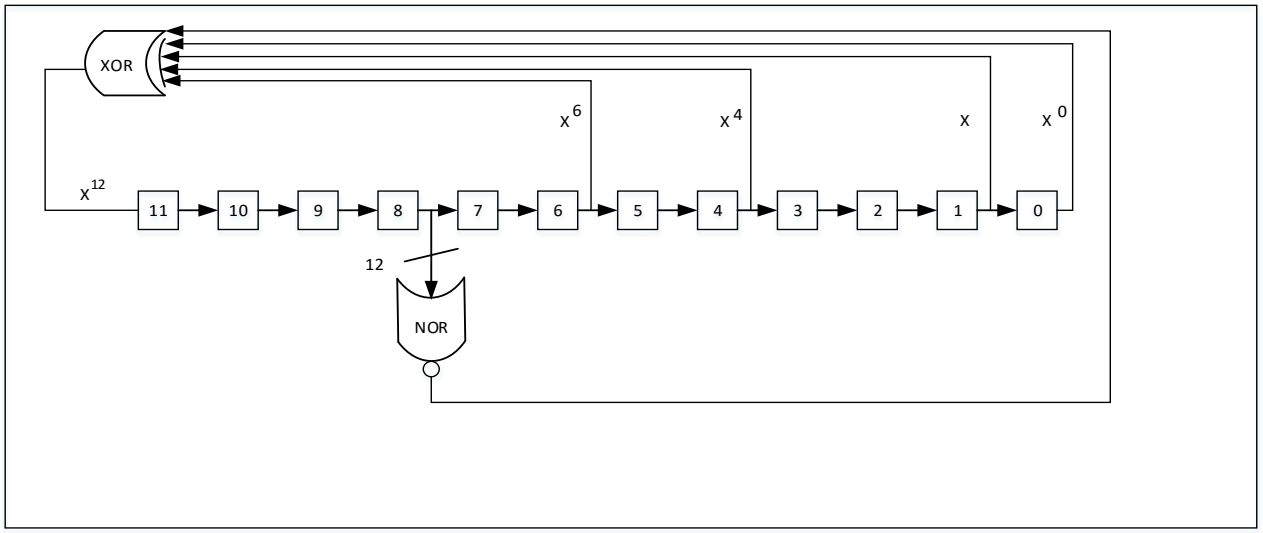
在双 DAC 模式下，如果 2 个通道的 DMAENx 位都为’1’，则会产生 2 个 DMA 请求。如果实际只需要一个 DMA 传输，则应只选择其中一个 DMAENx 位置’1’。这样，程序可以在只使用一个 DMA 请求，一个 DMA 通道的情况下，处理工作在双 DAC 模式的 2 个 DAC 信道。

DAC 的 DMA 请求不会累计，因此如果第 2 个外部触发发生在响应第 1 个外部触发之前，则不能处理第 2 个 DMA 请求，也不会报告错误。

14.3.8 噪声生成

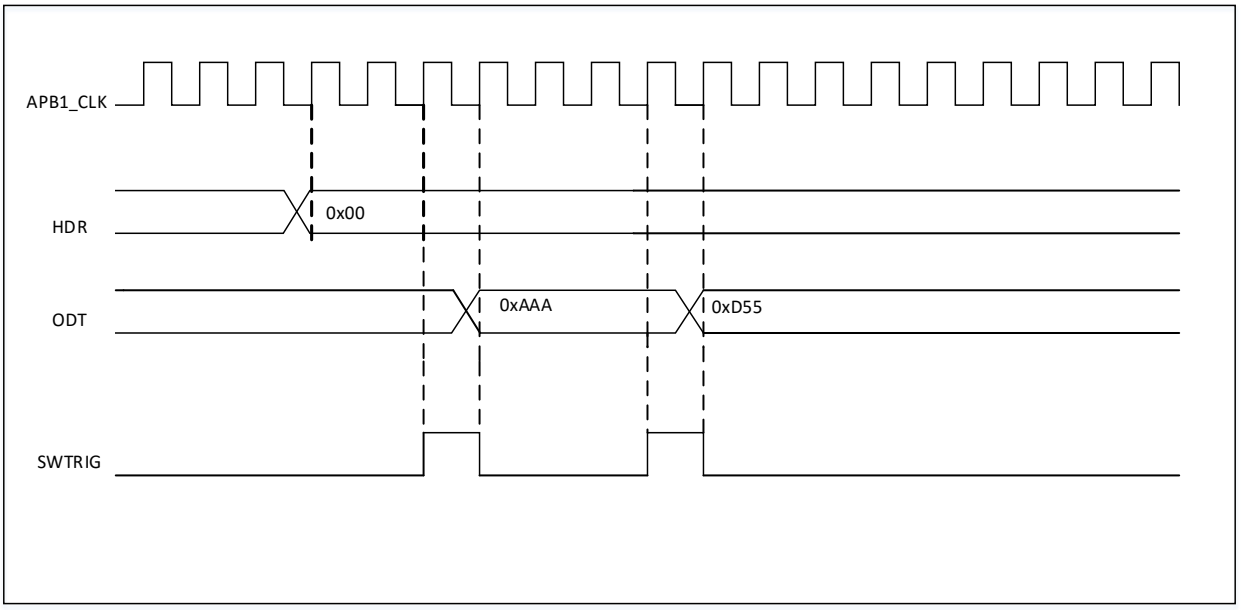
可以利用线性回馈移位寄存器（LENinear Feedback Shift Register LFSR）产生幅度变化的伪噪声。设置 WAVE[1：0]位为’01’选择 DAC 噪声生成功能。寄存器 LFSR 的预装入值为 0xAAA。按照特定算法，在每次触发事件后 3 个 APB1 时钟周期之后更新该寄存器的值。

图 14-5 DAC LFSR寄存器算法



设置 DAC_CTRL 寄存器的 MAMSx[3：0]位可以屏蔽部分或者全部 LFSR 的数据，这样的得到的 LSFR 值与 DAC_HDRx 的数值相加，去掉溢出位之后即被写入 DAC_ODTx 寄存器。 如果寄存器 LFSR 值为 0x000，则会注入’1’（防锁定机制）。将 WAVEx[1：0]位置’0’可以复位 LFSR 波形的生成算法。

图 14-6 带LFSR波形生成的DAC转换（使能软件触发）



注意： 为了产生噪声，必须使能 DAC 触发，即设 DAC_CTRL 寄存器的 TENx 位为’1’。

14.3.9 三角波生成

可以在 DC 或者缓慢变化的信号上加上一个小幅度的三角波。设置 WAVEx[1：0]位为’10’选择 DAC 的三角波生成功能。设置 DAC_CTRL 寄存器的 MAMSx[3：0]位来选择三角波的幅度。内部的三角波计数器每次触发事件之后 3 个 APB1 时钟周期后累加 1。计数器的值与 DAC_HDRx 寄存器的数值相加并丢弃溢出位后写入 DAC_ODTx 寄存器。在传入 DAC_ODTx 寄存器的数值小于 MAMS[3：0] 位定义的最大幅度时，三角波计数器逐步累加。一旦达到设置的最大幅度，则计数器开始递减，达到 0 后再开始累加，周而复始。

将 WAVEx[1：0]位置’0’可以复位三角波的生成。

图 14-7 DAC三角波生成

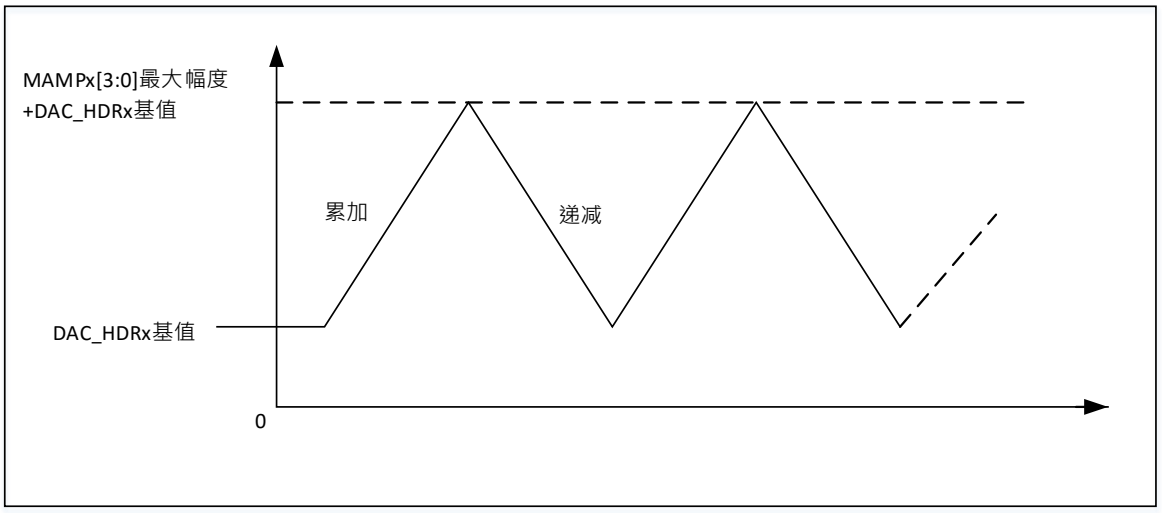
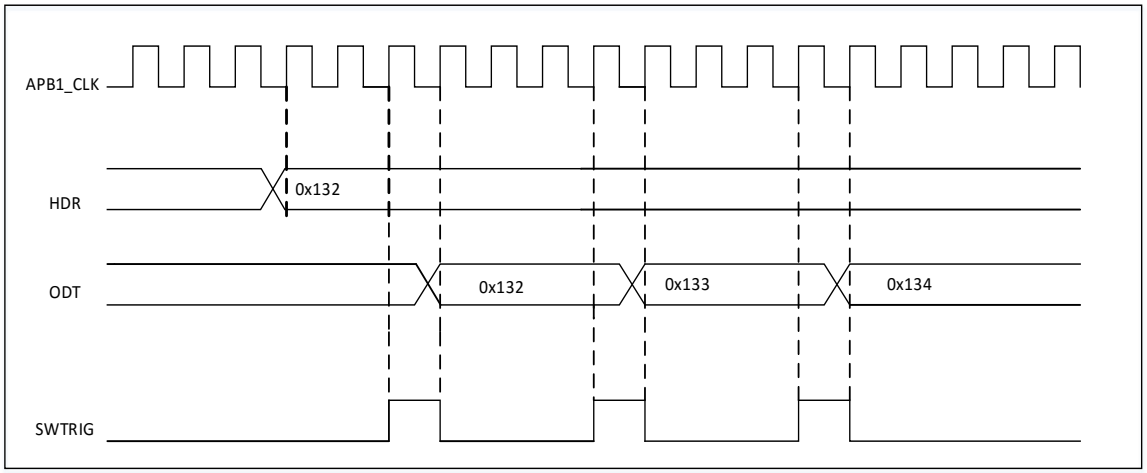


图 14-8 带三角生成的DAC转换（使能软件触发）



注意： 1.为了产生三角波，必须使能 DAC 触发，即设 DAC_CTRL 寄存器的 TENx 位为’1’。

2.MAMS[3：0]位必须在使能 DAC 之前设置，否则其值不能修改。

14.4 双DAC通道转换

在需要 2 个 DAC 同时工作的情况下，为了更有效地利用总线带宽，DAC 集成了 3 个供双 DAC 模式使用的寄存器：HDR8RD、HDR12RD 和 HDR12LD，只需要访问一个寄存器即可完成同时驱动 2 个 DAC 信道的操作。对于双 DAC 通道转换和这些专用寄存器，共有 11 种转换模式可用。这些转换模式在只使用一个 DAC 通道的情况下，仍然可通过独立的 HDRx 寄存器操作。所有模式详述于以下章节。

14.4.1 不使用波形发生器的独立触发

按照下列顺序设置 DAC 工作在此转换模式：

- 分别设置2个DAC通道的触发使能位TEN1和TEN2为’1’；
- 通过设置TGSL1[2：0]和TGSL2[2：0]位为不同值，分别配置2个DAC信道的不同触发源；
- 将双DAC信道转换数据装入所需的HDR寄存器（HDR12RD、HDR12LD或HDR8RD）。

当发生 DAC 通道 1 触发事件时，（延迟 3 个 APB1 时钟周期后）寄存器 HDR1 的值传入寄存器 DAC_ODT1。

当发生 DAC 通道 2 触发事件时，（延迟 3 个 APB1 时钟周期后）寄存器 HDR2 的值传入寄存器 DAC_ODT2。

14.4.2 使用相同LFSR的独立触发

按照下列顺序设置 DAC 工作在此转换模式：

- 分别设置2个DAC通道的触发使能位TEN1和TEN2为’1’；
- 通过设置TGSL1[2：0]和TGSL2[2：0]位为不同值，分别配置2个DAC信道的不同触发源；
- 设置2个DAC通道的WAVEx[1：0]位为”01”，并设置MAMSx[3：0]为相同的LFSR屏蔽值；
- 将双DAC信道转换数据装入所需的HDR寄存器（HDR12RD、HDR12LD或HDR8RD）。当发生DAC通道1触发事件时，具有相同屏蔽的LFSR1计数器值与HDR1寄存器数值相加，（延迟3个APB1时钟周期后）结果传入寄存器DAC_ODT1，然后更新LFSR1计数器。

当发生 DAC 通道 2 触发事件时，具有相同屏蔽的 LFSR2 计数器值与 HDR2 寄存器数值相加，（延迟 3 个 APB1 时钟周期后）结果传入寄存器 DAC_ODT2，然后更新 LFSR2 计数器。

14.4.3 使用不同LFSR的独立触发

按照下列顺序设置 DAC 工作在此转换模式：

- 分别设置2个DAC通道的触发使能位TEN1和TEN2为’1’；
- 通过设置TGSL1[2：0]和 TGSL2[2：0]位为不同值，分别配置2个DAC信道的不同触发源；
- 设置2个DAC通道的WAVEx[1：0]位为”01”，并设MAMSx[3：0]为不同的LFSR屏蔽值；
- 将双DAC信道转换数据装入所需的HDR寄存器（HDR12RD、HDR12LD 或者HDR8RD）。当发生DAC通道1触发事件时，按照MAMS1[3：0]所设屏蔽的LFSR1计数器值与HDR1寄存器数值相加，（延迟3个APB1时钟周期后）结果传入寄存器DAC_ODT1，然后更新LFSR1计数器。

当发生 DAC 通道 2 触发事件时，按照 MAMS2[3：0]所设屏蔽的 LFSR2 计数器值与 HDR2 寄存器数值相加，（延迟 3 个 APB1 时钟周期后）结果传入寄存器 DAC_ODT2，然后更新 LFSR2 计数器。

14.4.4 产生相同三角波的独立触发

按照下列顺序设置 DAC 工作在此转换模式：

- 分别设置2个DAC通道的触发使能位TEN1和TEN2为’1’；
- 通过设置TGSL1[2：0]和TGSL2[2：0]位为不同值，分别配置2个DAC信道的不同触发源；
- 设置2个DAC通道的WAVEx[1：0]位为”1x”，并设MAMSx[3：0]为相同的三角波幅值；
- 将双DAC信道转换数据装入所需的HDR寄存器（HDR12RD、HDR12LD 或HDR8RD）。当发生DAC通道1触发事件时，相同的三角波幅值加上HDR1寄存器的值，（延迟3个APB1时钟周期后）结果传入寄存器DAC_ODT1，然后更新DAC通道1三角波计数器。

当发生 DAC 通道 2 触发事件时，相同的三角波幅值加上 HDR2 寄存器的值，（延迟 3 个 APB1 时钟周期后）结果传入寄存器 DAC_ODT2，然后更新 DAC 通道 2 三角波计数器。

14.4.5 产生不同三角波的独立触发

按照下列顺序设置 DAC 工作在此转换模式：

- 分别设置2个DAC通道的触发使能位TEN1和TEN2为’1’；
- 通过设置TGSL1[2：0]和TGSL2[2：0]位为不同值，分别配置2个DAC信道的不同触发源；
- 设置2个DAC通道的WAVEx[1：0]位为’1x’，并设MAMSx[3：0]为不同的三角波幅值。
- 将双DAC信道转换数据装入所需的HDR寄存器（HDR12RD、HDR12LD或HDR8RD）。

当发生 DAC 通道 1 触发事件时，MAMS1[3：0]所设的三角波幅值加上 HDR1 寄存器数值，（延迟 3 个 APB1 时钟周期后）结果传入寄存器 DAC_ODT1，然后更新 DAC 通道 1 三角波计数器。当发生 DAC 通道 2 触发事件时，MAMS2[3：0]所设的三角波幅值加上 HDR2 寄存器数值，（延迟 3 个 APB1 时钟周期后）结果传入寄存器 DAC_ODT2，然后更新 DAC 通道 2 三角波计数器。

14.4.6 同时软件激活

按照下列过程设置 DAC 工作在此转换模式：

- 将双DAC信道转换数据装入所需的HDR寄存器（HDR12RD、HDR12LD或HDR8RD）。在此配置下，一个APB1时钟周期后，HDR1和HDR2寄存器的数值即被分别传入DAC_ODT1和DAC_ODT2寄存器。

14.4.7 不使用波形发生器的同时触发

按照下列顺序设置 DAC 工作在此转换模式：

- 分别设置2个DAC通道的触发使能位TEN1和TEN2为’1’；
- 通过设置TGSL1[2：0]和 TGSL2[2：0]位为相同值，分别配置2个DAC信道使用相同触发源；
- 将双DAC信道转换数据装入所需的HDR寄存器（HDR12RD、HDR12LD或HDR8RD）。

当发生触发 事件时，（延迟 3 个 APB1 时钟周期后）HDR1 和 HDR2 寄存器的数值分别传入 DAC_ODT1 和 DAC_ODT2 寄存器。

14.4.8 使用相同LFSR的同时触发

- 分别设置2个DAC通道的触发使能位TEN1和TEN2为’1’；
- 通过设置TGSL1[2：0]和TGSL2[2：0]位为相同值，分别配置2个DAC信道使用相同触发源；
- 设置2个DAC通道的WAVEx[1：0]位为”01”，并设MAMSx[3：0]为相同的LFSR屏蔽值；
- 将双DAC信道转换数据装入所需的HDR寄存器（HDR12RD、HDR12LD 或HDR8RD）；当发生触发事件时，MAMS1[3：0]所设屏蔽的LFSR1计数器值与HDR1寄存器的数值相加，（延迟3个APB1时钟周期后）结果传入DAC_ODT1寄存器，然后更新LFSR1计数器。

同样，MAMS1[3：0]所设屏蔽的 LFSR2 计数器值与 HDR2 寄存器的数值相加，（延迟 3 个 APB1 时钟周期后）结果传入寄存器 DAC_ODT2，然后更新 LFSR2 计数器。

14.4.9 使用不同LFSR的同时触发

按照下列顺序设置 DAC 工作在此转换模式：

- 分别设置2个DAC通道的触发使能位TEN1和TEN2为’1’；
- 通过设置TGSL1[2：0]和TGSL2[2：0]位为相同值，分别配置2个DAC信道使用相同触发源；
- 设置2个DAC通道的WAVEx[1：0]位为’01’，并设MAMSx[3：0]为不同的LFSR屏蔽值；
- 将双DAC信道转换数据装入所需的HDR寄存器（HDR12RD、HDR12LD 或HDR8RD）。当发生触发事件时，具有相同屏蔽的LFSR1计数器值与HDR1寄存器数值相加，（延迟3个APB1时钟周期后）结果传入寄存器DAC_ODT1，然后更新LFSR1计数器。同时，具有相同屏蔽的LFSR2计数器值与HDR2寄存器数值相加，（延迟3个APB1时钟周期后）结果传入寄存器DAC_ODT2，然后更新LFSR2计数器。

14.4.10 使用相同三角波发生器的同时触发

按照下列顺序设置 DAC 工作在此转换模式：

- 分别设置2个DAC通道的触发使能位TEN1和TEN2为’1’；
- 通过设置TGSL1[2：0]和 TGSL2[2：0]位为相同值，分别配置2个DAC信道使用相同触发源。
- 设置2个DAC通道的WAVEx[1：0]位为’1x’，并设MAMSx[3：0]为相同的三角波幅值。
- 将双DAC信道转换数据装入所需的HDR寄存器（HDR12RD、HDR12LD或HDR8RD）。当发生触发事件时，相同的三角波幅值与HDR1寄存器数值相加，（延迟3个APB1时钟周期后）结果传入寄存器DAC_ODT1，然后更新LFSR1计数器。

同时，相同的三角波幅值与 HDR2 寄存器数值相加，（延迟 3 个 APB1 时钟周期后）结果传入寄存器 DAC_ODT2，然后更新 LFSR2 计数器。

14.4.11 使用不同三角波发生器的同时触发

按照下列顺序设置 DAC 工作在此转换模式：

- 分别设置2个DAC通道的触发使能位TEN1和TEN2为’1’；
- 通过设置TGSL1[2：0]和 TGSL2[2：0]位为相同值，分别配置2个DAC信道使用相同触发源。
- 设置2个DAC通道的WAVEx[1：0]位为’1x’，并设MAMSx[3：0]为不同的三角波幅值。
- 将双DAC信道转换数据装入所需的HDR寄存器（HDR12RD、HDR12LD 或HDR8RD）。当发生触发事件时，MAMS1[3：0]所设的三角波幅值与HDR1寄存器数值相加，（延迟3个APB1时钟周期后）结果传入寄存器DAC_ODT1，然后更新LFSR1计数器。

同时，MAMS2[3：0]所设的三角波幅值与 HDR2 寄存器数值相加，（延迟 3 个 APB1 时钟周期后）结果传入寄存器 DAC_ODT2，然后更新 LFSR2 计数器。

14.5 DAC寄存器

必须以字（32 位）的方式操作这些外设寄存器。

下表列出了所有 DAC 寄存器。

表 14-3 DAC寄存器映射

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000h	DAC_CTRL	保留			DMAE	MAMS2 [3：0]				WAVE2 [1：0]		TGSEL2 [2：0]			TEN2	BFF2	EN2	保留			DMAE	MAMS1 [3：0]				WAVE1 [1：0]		TGSEL1 [2：0]			TEN1	BFF1	EN1
	复位值				0	0	0	0	0	0	0	0	0	0	0	0	0				0	0	0	0	0	0	0	0	0	0	0	0	0	0
004h	DAC_SWTRG	保留																														SWTRG2	SWTRG1
	复位值																															0	0
008h	DAC_HDR12R1	保留																D1HDR[11：0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
00Ch	DAC_HDR12L1	保留																D1HDR[11：0]														保留	
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0				
010h	DAC_HDR8R1	保留																								D1HDR[7：0]							
	复位值																									0	0	0	0	0	0	0	0
014h	DAC_HDR12R2	保留																D2HDR[11：0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
018h	DAC_HDR12L2	保留																D2HDR[11：0]														保留	
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0				
01Ch	DAC_HDR8R2	保留																								D2HDR[7：0]							
	复位值																									0	0	0	0	0	0	0	0
020h	DAC_HDR12RD	保留			D2HDR[11：0]													保留			D1HDR[11：0]												
	复位值				0	0	0	0	0	0	0	0	0	0	0	0	0				0	0	0	0	0	0	0	0	0	0	0	0	0	0
024h	DAC_HDR12LD	D2HDR[11：0]													保留			D1HDR[11：0]														保留	
	复位值	0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0				

028h	HDR8RD	保留							D2HDR[7：0]								D1HDR[7：0]
	复位值								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
02Ch	DAC_ODT1	保留															D1ODT[11：0]
	复位值																0	0	0	0	0	0	0	0	0	0	0	0
030h	DAC_ODT2	保留															D2ODT[11：0]
	复位值																0	0	0	0	0	0	0	0	0	0	0	0

14.5.1 DAC控制寄存器（DAC_CTRL）

地址偏移：0x00

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留			DMA EN2	MAMS2[3：0]				WAVE2[2：0]		TGSL2[2：0]			TEN2	BFF2	EN2
res			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留			DMA EN1	MAMS1[3：0]				WAVE1[2：0]		TGSL1[2：0]			TEN1	BFF1	EN1
res			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31：29	保留。
位 28	**DMAEN2**：DAC 通道 2 DMA 使能（DAC channel2 DMA enable） 该位由软件设置和清除。 0：关闭 DAC 信道 2 DMA 模式； 1：使能 DAC 信道 2 DMA 模式。
位 27：24	**MAMS2[3：0]**：DAC 通道 2 屏蔽/幅值选择器（DAC channel2 mask/amplitude selector） 由软件设置这些位，用来在噪声生成模式下选择屏蔽位，在三角波生成模式下选择波形的幅 值。 0000：不屏蔽 LSFR 位 0 / 三角波幅值等于 1； 0001：不屏蔽 LSFR 位[1：0] / 三角波幅值等于 3； 0010：不屏蔽 LSFR 位[2：0] / 三角波幅值等于 7； 0011：不屏蔽 LSFR 位[3：0] / 三角波幅值等于 15； 0100：不屏蔽 LSFR 位[4：0] / 三角波幅值等于 31； 0101：不屏蔽 LSFR 位[5：0] / 三角波幅值等于 63； 0110：不屏蔽 LSFR 位[6：0] / 三角波幅值等于 127； 0111：不屏蔽 LSFR 位[7：0] / 三角波幅值等于 255； 1000：不屏蔽 LSFR 位[8：0] / 三角波幅值等于 511； 1001：不屏蔽 LSFR 位[9：0] / 三角波幅值等于 1023； 1010：不屏蔽 LSFR 位[10：0] / 三角波幅值等于 2047； ≥1011：不屏蔽 LSFR 位[11：0] / 三角波幅值等于 4095。
位 23：22	**WAVE2[1：0]**：DAC 信道 2 噪声/三角波生成使能（DAC channel2 noise/triangle wave generation enable） 该 2 位由软件设置和清除。 00：关闭波形发生器； 01：使能噪声波形发生器； 1x：使能三角波发生器。

位 21：19	**TGSL2[2：0]**：DAC 通道 2 触发选择（DAC channel2 trigger selection） 该 3 位用于选择 DAC 通道 2 的外部触发事件。 000：**TMR6 TRGO** 事件； 001：**TMR8 TRGO** 事件； 010：**TMR7 TRGO** 事件； 011：**TMR5 TRGO** 事件； 100：**TMR2 TRGO** 事件； 101：**TMR4 TRGO** 事件； 110：**外部中断线 9**； 111：**软件触发**。 注意：该 3 位只能在 **TEN2** = 1（DAC 通道 2 触发使能）时设置。
位 18	**TEN2**：DAC 通道 2 触发使能（DAC channel2 trigger enable） 该位由软件设置和清除，用来使能/关闭 DAC 通道 2 的触发。 0：关闭DAC 通道2 触发，写入**DAC_HDRx** 寄存器的资料在1 个APB1 时钟周期后传入 **DAC_ODT2** 寄存器； 1：使能DAC 通道2 触发，写入**DAC_HDRx** 寄存器的资料在3 个APB1 时钟周期后传入 **DAC_ODT2** 寄存器。 注意：如果选择软件触发，写入寄存器 **DAC_HDRx** 的资料只需要 1 个 APB1 时钟周期就可以传入 寄存器 **DAC_ODT2**。
位 17	**BFF2**：关闭 DAC 通道 2 输出缓存（DAC channel2 output buffer disable） 该位由软件设置和清除，用来使能/关闭 DAC 通道 2 的输出缓存。 0：使能 DAC 通道 2 输出缓存； 1：关闭 DAC 通道 2 输出缓存。
位 16	**EN2**：DAC 通道 2 使能（DAC channel2 enable） 该位由软件设置和清除，用来使能/关闭 DAC 通道 2。 0：关闭 DAC 通道 2； 1：使能 DAC 通道 2。
位 15：13	保留。
位 12	**DMAEN1**：DAC 通道 1 DMA 使能（DAC channel1 DMA enable） 该位由软件设置和清除。 0：关闭 DAC 信道 1 **DMA** 模式； 1：使能 DAC 信道 1 **DMA** 模式。
位 11：8	**MAMS1[3：0]**：DAC 通道 1 屏蔽/幅值选择器（DAC channel1 mask/amplitude selector） 由软件设置这些位，用来在噪声生成模式下选择屏蔽位，在三角波生成模式下选择波形的幅 值。 0000：不屏蔽 LSFR 位 0 / 三角波幅值等于 1； 0001：不屏蔽 LSFR 位[1：0] / 三角波幅值等于 3； 0010：不屏蔽 LSFR 位[2：0] / 三角波幅值等于 7； 0011：不屏蔽 LSFR 位[3：0] / 三角波幅值等于 15； 0100：不屏蔽 LSFR 位[4：0] / 三角波幅值等于 31； 0101：不屏蔽 LSFR 位[5：0] / 三角波幅值等于 63； 0110：不屏蔽 LSFR 位[6：0] / 三角波幅值等于 127； 0111：不屏蔽 LSFR 位[7：0] / 三角波幅值等于 255； 1000：不屏蔽 LSFR 位[8：0] / 三角波幅值等于 511； 1001：不屏蔽 LSFR 位[9：0] / 三角波幅值等于 1023； 1010：不屏蔽 LSFR 位[10：0] / 三角波幅值等于 2047； ≥1011：不屏蔽 LSFR 位[11：0] / 三角波幅值等于 4095。

位 7：6	**WAVE1[1：0]**：DAC 信道 1 噪声/三角波生成使能（DAC channel1 noise/triangle wave generation enable） 该 2 位由软件设置和清除。 00：关闭波形生成； 01：使能噪声波形发生器； 1x：使能三角波发生器。
位 5：3	**TGSL1[2：0]**：DAC 通道 1 触发选择（DAC channel1 trigger selection） 该位用于选择 DAC 通道 1 的外部触发事件。 000：**TMR6 TRGO** 事件； 001：**TMR8 TRGO** 事件； 010：**TMR7 TRGO** 事件； 011：**TMR5 TRGO** 事件； 100：**TMR2 TRGO** 事件； 101：**TMR4 TRGO** 事件； 110：**外部中断线 9**； 111：**软件触发**。 注意：该位只能在 **TEN1**= 1（DAC 通道 1 触发使能）时设置。
位 2	**TEN1**：DAC 通道 1 触发使能（DAC channel1 trigger enable） 该位由软件设置和清除，用来使能/关闭 DAC 通道 1 的触发。 0：关闭 DAC 通道 1 触发，写入寄存器 **DAC_HDRx** 的数据在 1 个 APB1 时钟周期后传入寄存器 **DAC_ODT1**； 1：使能 DAC 通道 1 触发，写入寄存器 DAC_HDRx 的数据在 3 个 APB1 时钟周期后传入寄存器 **DAC_ODT1**。 注意：如果选择软件触发，写入寄存器 **DAC_HDRx** 的资料只需要 1 个 APB1 时钟周期就可以传入寄存器 **DAC_ODT1**。
位 1	**BFF1**：关闭 DAC 通道 1 输出缓存（DAC channel1 output buffer disable） 该位由软件设置和清除，用来使能/关闭 DAC 通道 1 的输出缓存。 0：使能 DAC 通道 1 输出缓存； 1：关闭 DAC 通道 1 输出缓存。
位 0	**EN1**：DAC 通道 1 使能（DAC channel1 enable） 该位由软件设置和清除，用来使能/失能 DAC 通道 1。 0：关闭 DAC 通道 1； 1：使能 DAC 通道 1。

14.5.2 DAC软件触发寄存器（DAC_SWTRG）

地址偏移：0x04

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															
res															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留														SWTRG2	SWTRG1
res														rw	rw

位 31：2	保留。
位 1	**SWTRG2**：DAC 信道 2 软件触发（DAC channel2 software trigger） 该位由软件设置和清除，用来使能/关闭软件触发。 0：关闭 DAC 信道 2 软件触发； 1：使能 DAC 信道 2 软件触发。 注意：一旦寄存器 **DAC_HDR2** 的数据传入寄存器 **DAC_ODT2**，（1 个 APB1 时钟周期后）该位由硬件置'0'。
位 0	**SWTRG1**：DAC 信道 1 软件触发 （DAC channel1 software trigger） 该位由软件设置和清除，用来使能/关闭软件触发。 0：关闭 DAC 信道 1 软件触发； 1：使能 DAC 信道 1 软件触发。 注意：一旦寄存器 **DAC_HDR1** 的数据传入寄存器 **DAC_ODT1**，（1 个 APB1 时钟周期后）该位由硬件置’0’。

14.5.3 DAC信道1的12位右对齐数据保持寄存器（DAC_HDR12R1）

地址偏移：0x08

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															
res															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留				D1HDR[11：0]											
res				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31：12	保留。
位 11：0	**D1HDR[11：0]**：DAC 信道 1 的 12 位右对齐数据（DAC channel1 12-bit right-aligned data） 该位由软件写入，表示 DAC 信道 1 的 12 位数据。

14.5.4 DAC信道1的12位左对齐数据保持寄存器（DAC_HDR12L 1）

地址偏移：0x0C

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															
res															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
D1HDR[11：0]												保留			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	res			

位 31：16	保留。

位 15：4	**D1HDR[11：0]**：DAC 信道 1 的 12 位左对齐数据（DAC channel1 12-bit left-aligned data） 该位由软件写入，表示 DAC 信道 1 的的 12 位数据.
位 3：0	保留。

14.5.5 DAC信道1的8位右对齐数据保持寄存器（DAC_HDR8R1）

地址偏移：0x10

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															
res															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留								D1HDR[7：0]							
res								rw	rw	rw	rw	rw	rw	rw	rw

位 31：18	保留。
位 7：0	**D1HDR[7：0]**：DAC 信道 1 的 8 位右对齐数据（DAC channel1 8-bit right-aligned data） 该位由软件写入，表示 DAC 信道 1 的的 8 位数据。

14.5.6 DAC信道2的12位右对齐数据保持寄存器（DAC_HDR12 R2）

地址偏移：0x14

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															
res															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留				D2HDR[11：0]											
res				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31：12	保留。
位 11：0	**D2HDR[11：0]**：DAC 信道 2 的 12 位右对齐数据（DAC channel2 12-bit right-aligned data） 该位由软件写入，表示 DAC 信道 2 的 12 位数据。

14.5.7 DAC信道2的12位左对齐数据保持寄存器（DAC_HDR12L 2）

地址偏移：0x18

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															
res															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
D2HDR[11：0]												保留			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	res			

位 31：16	保留。
位 15：4	**DAC2HDR[11：0]**：DAC 信道 2 的 12 位左对齐数据（DAC channel2 12-bit left-aligned data） 该位由软件写入，表示 DAC 信道 2 的的 12 位数据。
位 3：0	保留。

14.5.8 DAC信道2的8位右对齐数据保持寄存器（DAC_HDR8R2）

地址偏移：0x1C

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															
res															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留								D2HDR[7：0]							
res								rw	rw	rw	rw	rw	rw	rw	rw

位 31：18	保留。
位 7：0	**D2HDR[7：0]**：DAC 信道 2 的 8 位右对齐数据（DAC channel2 8-bit right-aligned data）该位由软件写入，表示 DAC 信道 2 的的 8 位数据。

14.5.9 双DAC的12位右对齐数据保持寄存器（DAC_HDR12RD）

地址偏移：0x20

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留				D2HDR[11：0]											
res				RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留				D1HDR[11：0]											
res				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31：28	保留。
位 27：16	**D2HDR[11：0]**：DAC 信道 2 的 12 位右对齐数据（DAC channel2 12-bit right-aligned data） 该位由软件写入，表示 DAC 信道 2 的 12 位数据。
位 15：12	保留。
位 11：0	**D1HDR[11：0]**：DAC 信道 1 的 12 位右对齐数据（DAC channel1 12-bit right-aligned data） 该位由软件写入，表示 DAC 信道 2 的 12 位数据。

14.5.10 双DAC的12位左对齐数据保持寄存器（DAC_HDR12LD）

地址偏移：0x24

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
D2HDR[11：0]												保留			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	res			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
D1HDR[11：0]												保留			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	res			

位 31：20	**D2HDR[11：0]**：DAC 信道 2 的 12 位左对齐数据（DAC channel2 12-bit left-aligned data） 该位由软件写入，表示 DAC 信道 2 的 12 位数据。
位 19：16	保留。
位 15：4	**D1HDR[11：0]**：DAC 信道 1 的 12 位左对齐数据（DAC channel1 12-bit left-aligned data） 该位由软件写入，表示 DAC 信道 1 的 12 位数据。
位 3：0	保留。

14.5.11 双DAC的8位右对齐数据保持寄存器（DAC_HDR8RD）

地址偏移：0x28

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															
res															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
D2HDR[7：0]								D1HDR[7：0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31：16	保留。
位 15：8	**D2HDR[7：0]**：DAC 信道 2 的 8 位右对齐数据（DAC channel2 8-bit right-aligned data） 该位由软件写入，表示 DAC 信道 2 的的 8 位数据。
位 7：0	**D1HDR[7：0]**：DAC 信道 1 的 8 位右对齐数据（DAC channel1 8-bit right-aligned data） 该位由软件写入，表示 DAC 信道 1 的的 8 位数据。

14.5.12 DAC信道1数据输出寄存器（DAC_ODT1）

地址偏移：0x2C

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															
res															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留				D1ODT[11：0]											
res				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31：12	保留。
位 11：0	**D1ODT[11：0]**：DAC 信道 1 输出数据（DAC channel1 data output） 该位由软件写入，表示 DAC 信道 1 的输出数据。

14.5.13 DAC信道2数据输出寄存器（DAC_ODT2）

地址偏移：0x30

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															
res															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留				D2ODT[11：0]											
res				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31：12	保留。
位 11：0	**D2ODT[11：0]**：DAC 信道 2 输出数据（DAC channel2 data output） 该位由软件写入，表示 DAC 信道 2 的输出数据。

15 I²C接口

15.1 I²C简介

I²C(芯片间)总线接口连接微控制器和串行 I²C 总线。它提供多主机功能，控制所有 I²C 总线特定的时序、协议、仲裁和定时。支持标准和快速两种模式，同时与 SMBus2.0 兼容。I²C 模块有多种用途，包括 CRC 码的生成和校验、SMBus(系统管理总线—System Management Bus)和 PMBus(电源管理总线—Power Management Bus)。根据特定设备的需要，可以使用 DMA 以减轻 CPU 的负担。

15.2 I²C主要特点

- 并行总线/I²C总线协议转换器
- 多主机功能：该模块既可做主设备也可做从设备
- I²C主设备功能
  - 产生时钟
  - 产生起始和停止信号
- I²C从设备功能
  - 可编程的I²C地址检测
  - 可响应2个从地址的双地址能力
  - 停止位检测
- 产生和检测7位/10位地址和广播呼叫
- 支持不同的通讯速度
  - 标准速度(高达100 kHz)
  - 快速(高达400 kHz)
- 状态标志：
  - 发送器/接收器模式标志
  - 字节发送结束标志
  - I²C总线忙标志
- 错误标志
  - 主模式时的仲裁丢失
  - 地址/数据传输后的应答(ACK)错误
  - 检测到错位的起始或停止条件
  - 上溢或下溢(在禁止时钟延长的情况下)
- 2个中断向量
  - 1个中断用于地址/数据通讯成功
  - 1个中断用于错误
- 可选的禁止时钟延长功能
- 具单字节缓冲器的DMA
- 可配置的PEC(信息包错误检测)的产生或校验
  - 发送模式中PEC值可以作为最后一个字节传输
  - 用于最后一个接收字节的PEC错误校验
- 兼容SMBus 2.0
  - 25 ms时钟低超时延时
  - 10 ms主设备累积时钟低扩展时间
  - 25 ms从设备累积时钟低扩展时间

- 带ACK控制的硬件PEC产生/校验
- 支持地址分辨协议(ARP)

- 兼容PMBus

15.3 I²C功能描述

I²C 模块接收和发送数据，并将数据从串行转换成并行，或并行转换成串行。可以开启或禁止中断。接口通过数据引脚(SDA)和时钟引脚(SCL)连接到 I²C 总线。允许连接到标准(高达 100kHz)或快速(高达 400kHz)的 I²C 总线。

15.3.1 模式选择

接口可以运行下述模式中的一种：

- 从发送器模式
- 从接收器模式
- 主发送器模式
- 主接收器模式

在默认状态下，接口工作于从模式。接口在生成起始条件后自动地从从模式切换到主模式；当仲裁丢失或产生停止信号时，则从主模式切换到从模式。允许多主机功能。

通信流

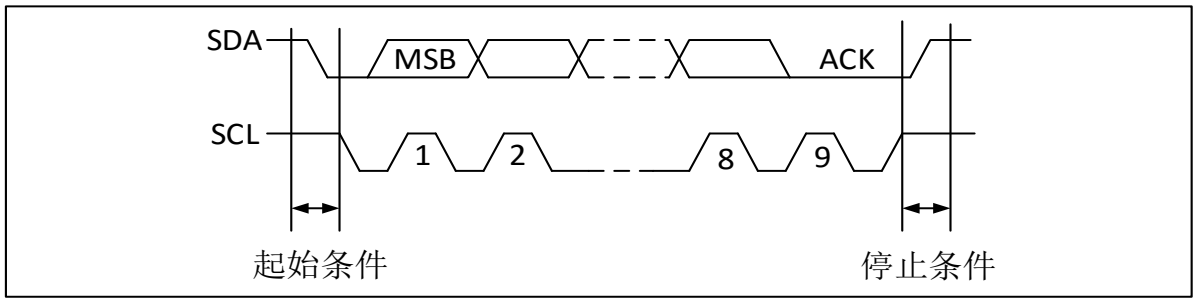
主模式时，I²C 接口启动数据传输并产生时钟信号。串行数据传输总是以起始条件开始并以停止条件结束。起始条件和停止条件都是在主模式下由软件控制产生。

从模式时，I²C 接口能识别它自己的地址(7 位或 10 位)和广播呼叫地址。软件能够开启或禁止广播呼叫地址的识别功能。

数据和地址按 8 位(也即一个字节)进行传输，高位在前。跟在起始条件后的 1 或 2 个字节是地址(7 位模式为 1 个字节，10 位模式为 2 个字节)。地址只在主模式发送。

在一个字节传输的 8 个时钟后的第 9 个时钟期间，接收器必须回送一个应答位(ACK)给发送器。参考下图。

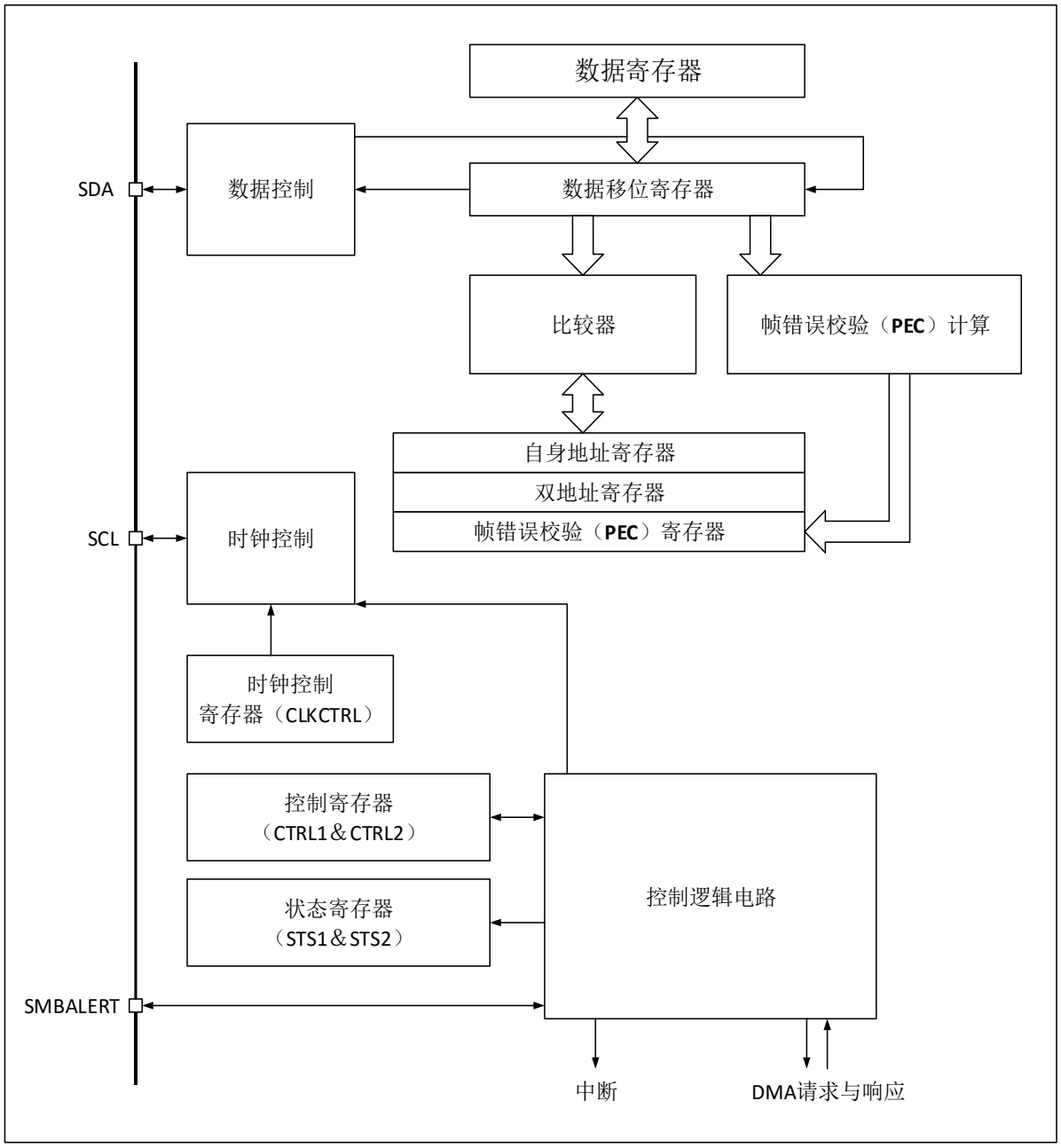
图 15-1 I²C总线协议



软件可以开启或禁止应答(ACKEN)，并可以设置 I²C 接口的地址(7 位、10 位地址或广播呼叫地址)。

I²C 接口的功能框图示于下图。

图 15-2 I²C的功能框图



注意： 在 SMBus 模式下，SMBALERT 是可选信号。如果禁止了 SMBus，则不能使用该信号。

15.3.2 I²C从模式

默认情况下，I²C 接口总是工作在从模式。从从模式切换到主模式，需要产生一个起始条件。为了产生正确的时序，必须在 I2C_CTRL2 寄存器中设定该模块的输入时钟。输入时钟的频率必须至少是：

- 标准模式下为：2MHz
- 快速模式下为：4MHz

一旦检测到起始条件，在 SDA 线上接收到的地址被送到移位寄存器。然后与芯片自己的地址 OADDR1 和 OADDR2(当 DUALEN=1)或者广播呼叫地址(如果 GCEN=1)相比较。

注意： 在 10 位地址模式时，比较包括头段序列(11110xx0)，其中的 xx 是地址的两个最高有效位。

头段或地址不匹配：I²C 接口将其忽略并等待另一个起始条件。

头段匹配(仅 10 位模式)：如果 ACKEN 位被置'1'，I²C 接口产生一个应答脉冲并等待 8 位从地址。

地址匹配：I²C 接口产生以下时序：

- 如果ACKEN被置'1'，则产生一个应答脉冲
- 硬件设置ADDRF位；如果设置了EVTITEN位，则产生一个中断
- 如果DUALEN=1，软件必须读DUALF位，以确认响应了哪个从地址

在 10 位模式，接收到地址序列后，从设备总是处于接收器模式。当接收到重复的起始条件后，在收到与地址匹配的头序列并且最低位为'1'(即 11110xx1)时，将进入发送器模式。

从模式下 TRF 位指示当前是处于接收器模式还是发送器模式。

从发送器

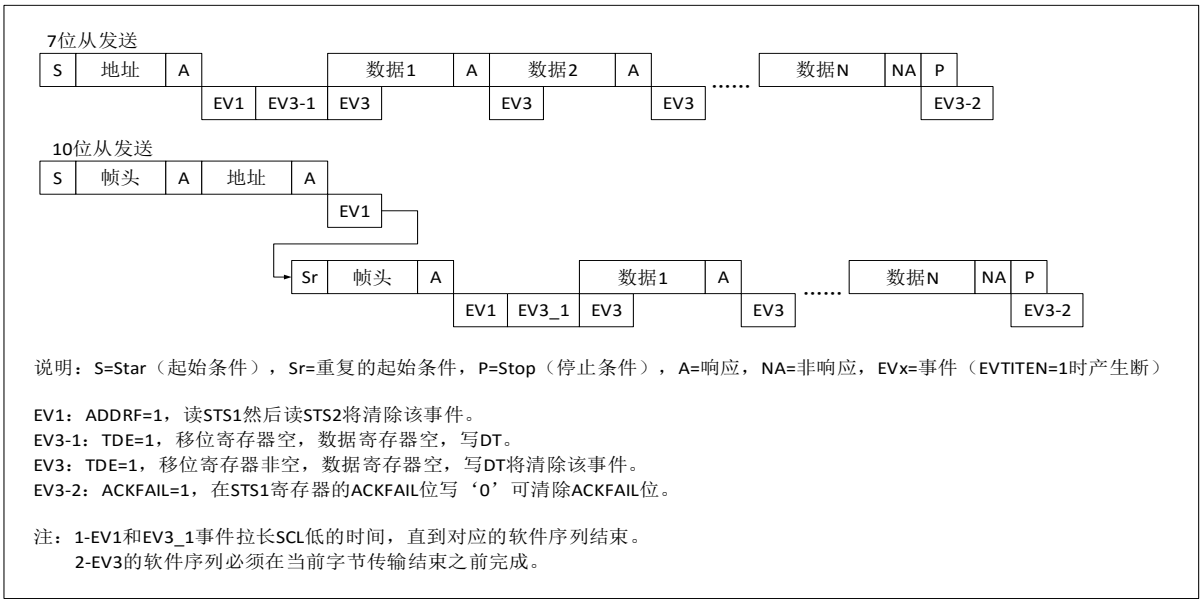
在接收到地址和清除 ADDRF 位后，从发送器将字节从 DT 寄存器经由内部移位寄存器发送到 SDA 线上。

从设备保持 SCL 为低电平，直到 ADDRF 位被清除并且待发送数据已写入 DT 寄存器。(见图 15-3 中的 EV1 和 EV3)。

当收到应答脉冲时：

- TDE位被硬件置位，如果设置了EVTITEN和BUFITEN位，则产生一个中断如果TDE位被置位，但在下一个数据发送结束之前没有新数据写入到I2C_DT寄存器，则BTFF位被置位，在清除BTFF之前I²C接口将保持SCL为低电平；读出I2C_STS1之后再写入I2C_DT寄存器将清除BTFF位。

图 15-3 从发送器的传送序列图



说明：S=Star（起始条件），Sr=重复的起始条件，P=Stop（停止条件），A=响应，NA=非响应，EVx=事件（EVTITEN=1时产生断）

EV1：ADDRF=1，读STS1然后读STS2将清除该事件。
EV3-1：TDE=1，移位寄存器空，数据寄存器空，写DT。
EV3：TDE=1，移位寄存器非空，数据寄存器空，写DT将清除该事件。
EV3-2：ACKFAIL=1，在STS1寄存器的ACKFAIL位写'0'可清除ACKFAIL位。

注：1-EV1和EV3_1事件拉长SCL低的时间，直到对应的软件序列结束。
2-EV3的软件序列必须在当前字节传输结束之前完成。

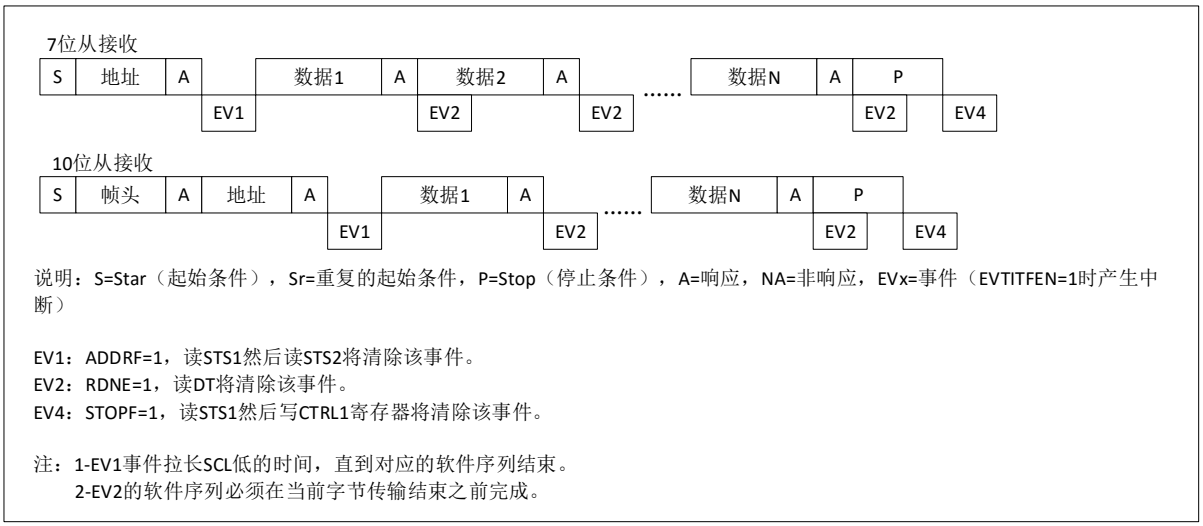
从接收器

在接收到地址并清除 ADDRF 后，从接收器将通过内部移位寄存器从 SDA 线接收到的字节存进 DT 寄存器。I²C 接口在接收到每个字节后都执行下列操作：

- 如果设置了ACKEN位，则产生一个应答脉冲
- 硬件设置RDNE=1如果设置了EVTITEN和BUFITEN位，则产生一个中断。

如果 RDNE 被置位，并且在接收新的数据结束之前 DT 寄存器未被读出，BTFF 位被置位，在清除 BTFF 之前 I²C 接口将保持 SCL 为低电平；读出 I2C_STS1 之后再读出 I2C_DT 寄存器将清除 BTFF 位。

图 15-4 从接收器的传送序列图



关闭从通信

在传输完最后一个数据字节后，主设备产生一个停止条件，I²C 接口检测到这一条件时：

- 设置STOPF=1，如果设置了EVTITEN位，则产生一个中断

然后 I²C 接口等待读 STS1 寄存器，再写 CR1 寄存器，完成这个操作后 STOPF 位被清零。(见图 15-4 中的 EV4)。

15.3.3 I²C主模式

在主模式时，I²C 接口启动数据传输并产生时钟信号。串行数据传输总是以起始条件开始并以停止条件结束。当通过 STARTGEN 位在总线上产生了起始条件，设备就进入了主模式。

以下是主模式所要求的操作顺序：

- 在I2C_CTRL2寄存器中设定该模块的输入时钟以产生正确的时序
- 配置时钟控制寄存器
- 配置上升时间寄存器
- 编程I2C_CTRL1寄存器启动外设
- 置I2C_CTRL1寄存器中的STARTGEN位为1，产生起始条件

I²C 模块的输入时钟频率必须至少是：

- 标准模式下为：2MHz
- 快速模式下为：4MHz

主机时钟的产生

使用 CLKCTRL 来产生 I²C 时钟的高低电平，它从时钟的上升沿或者下降沿开始。由于从机可能会拉低时钟线，因此外设此后等待一个 TMRISE 寄存器设置的最大上升时间后去检测时钟线的电平。

- 如果时钟线为低，说明从机拉低了总线，并且计数高电平的计数器停止计数直到检测到时钟线变高。这样可以保证时钟的高电平宽度
- 如果时钟线为高，高电平计数器持续计数

起始条件

当 BUSYF=0 时，设置 STARTGEN=1，I²C 接口将产生一个开始条件并切换至主模式(MSF 位置位)。

注意：在主模式下，设置 STARTGEN 位将在当前字节传输完后由硬件产生一个重开始条件。

一旦发出开始条件：

- STARTF位被硬件置位，如果设置了EVTITEN位，则会产生一个中断

然后主设备等待读 STS1 寄存器，紧跟着将从地址写入 DT 寄存器(见图 15-5 和图 15-6 的 EV5)。

从地址的发送

从地址通过内部移位寄存器被送到 SDA 线上。

- 在10位地址模式时，发送一个头段序列产生以下事件
  - ADDR10F位被硬件置位，如果设置了EVTITEN位，则产生一个中断。

  然后主设备等待读STS1寄存器，再将第二个地址字节写入DT寄存器(见图 15-5和图 15-6)
  - ADDRF位被硬件置位，如果设置了EVTITEN位，则产生一个中断。

  随后主设备等待一次读STS1寄存器，跟着读STS2寄存器(见图 15-5和图 15-6)
- 在7位地址模式时，只需送出一个地址字节

  一旦该地址字节被送出，
  - ADDRF位被硬件置位，如果设置了EVTITEN位，则产生一个中断

  随后主设备等待一次读STS1寄存器，跟着读STS2寄存器(见图 15-5和图 15-6)。

根据送出从地址的最低位，主设备决定进入发送器模式还是进入接收器模式。

- 在7位地址模式时
  - 要进入发送器模式，主设备发送从地址时置最低位为'0'
  - 要进入接收器模式，主设备发送从地址时置最低位为'1'
- 在10位地址模式时
  - 要进入发送器模式，主设备先送头字节(11110xx0)，然后送从地址。(这里xx代表10位地址中的最高2位。)
  - 要进入接收器模式，主设备先送头字节(11110xx0)，然后送从地址。然后再重新发送一个开始条件，后面跟着头字节(11110xx1)(这里xx代表10位地址中的最高2位。)

TRF 位指示主设备是在接收器模式还是发送器模式。

主发送器

在发送了地址和清除了 ADDRF 位后，主设备通过内部移位寄存器将字节从 DT 寄存器发送到 SDA 线上。

主设备等待，直到数据写入 DT 寄存器，TDE 被清除，(见图 15-5 中的 EV8)。

当收到应答脉冲时：

- TDE位被硬件置位，如果设置了EVTITEN和BUFITEN位，则产生一个中断

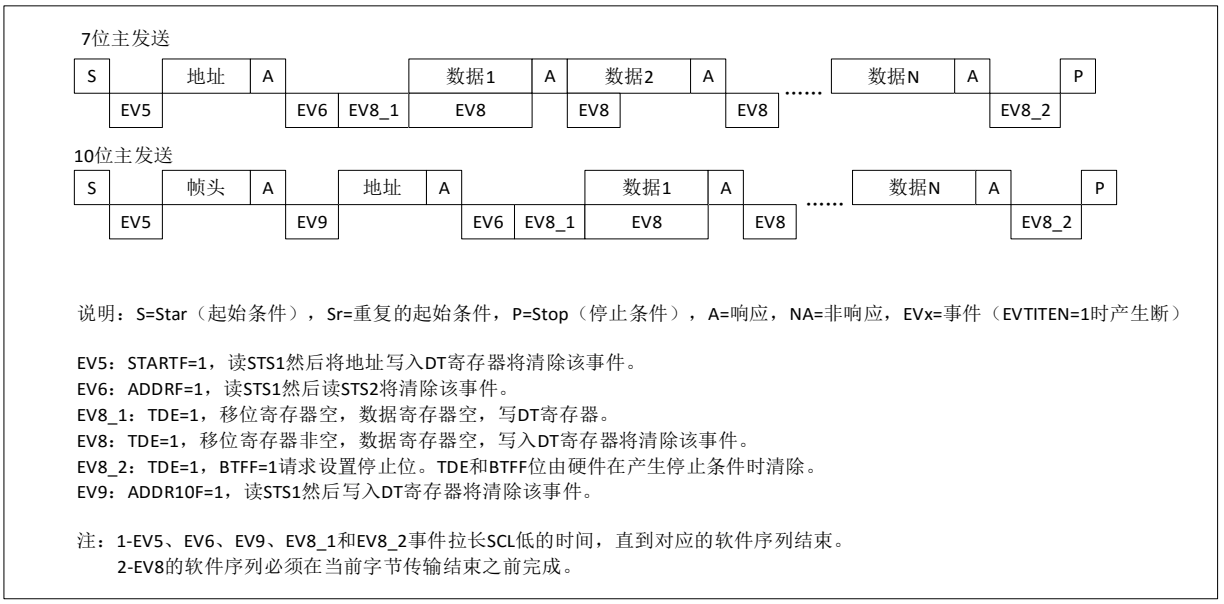
如果 TDE 被置位并且在上一次数据发送结束之前没有写新的数据字节到 DT 寄存器，则 BTFF 被硬件置位，在清除 BTFF 之前 I^2C 接口将保持 SCL 为低电平；读出 I2C_STS1 之后再写入 I2C_DT 寄存器将清除 BTFF 位。

关闭通信

在 DT 寄存器中写入最后一个字节后，通过设置 STOPGEN 位产生一个停止条件(见图 15-5 的 EV8_2)，然后 I^2C 接口将自动回到从模式(MSF 位清除)。

注意：当 TDE 或 BTFF 位置位时，停止条件应安排在出现 EV8_2 事件时。

图 15-5 主发送器传送序列图



说明：S=Star（起始条件），Sr=重复的起始条件，P=Stop（停止条件），A=响应，NA=非响应，EVx=事件（EVTITEN=1时产生断）

EV5：STARTF=1，读STS1然后将地址写入DT寄存器将清除该事件。
EV6：ADDRF=1，读STS1然后读STS2将清除该事件。
EV8_1：TDE=1，移位寄存器空，数据寄存器空，写DT寄存器。
EV8：TDE=1，移位寄存器非空，数据寄存器空，写入DT寄存器将清除该事件。
EV8_2：TDE=1，BTFF=1请求设置停止位。TDE和BTFF位由硬件在产生停止条件时清除。
EV9：ADDR10F=1，读STS1然后写入DT寄存器将清除该事件。

注：1-EV5、EV6、EV9、EV8_1和EV8_2事件拉长SCL低的时间，直到对应的软件序列结束。
2-EV8的软件序列必须在当前字节传输结束之前完成。

主接收器

在发送地址和清除 ADDRF 之后，I²C 接口进入主接收器模式。在此模式下，I²C 接口从 SDA 线接收数据字节，并通过内部移位寄存器送至 DT 寄存器。在每个字节后，I²C 接口依次执行以下操作：

- 如果ACKEN位被置位，发出一个应答脉冲
- 硬件设置RDNE=1，如果设置了EVTITEN和BUFITEN位，则会产生一个中断(见图15-6中的EV7)

如果 RDNE 位被置位，并且在接收新数据结束前，DT 寄存器中的数据没有被读走，硬件将设置 BTFF=1，在清除 BTFF 之前 I²C 接口将保持 SCL 为低电平；读出 I2C_STS1 之后再读出 I2C_DT 寄存器将清除 BTFF 位。

关闭通信

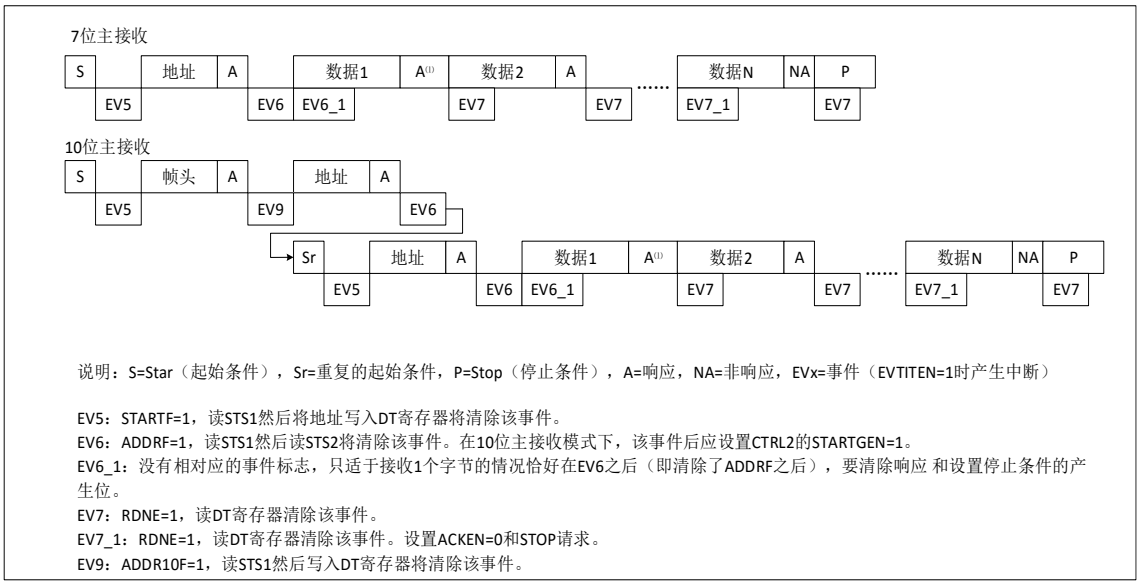
第一种情况：I2C 中断设为最高优先级

主设备在从从设备接收到最后一个字节后发送一个 NACK。接收到 NACK 后，从设备释放对 SCL 和 SDA 线的控制；主设备就可以发送一个停止/重起始条件。

- 为了在收到最后一个字节后产生一个NACK脉冲，在读倒数第二个数据字节之后(在倒数第二个RDNE事件之后)必须清除ACKEN位
- 为了产生一个停止/重起始条件，软件必须在读倒数第二个数据字节之后(在倒数第二个RDNE事件之后)设置STOPGEN/STARTGEN位
- 只接收一个字节时，刚好在EV6之后(EV6_1时，清除ADDRF之后)要关闭应答和停止条件的产生位

在产生了停止条件后，I²C 接口自动回到从模式(MSF 位被清除)。

图 15-6 主接收器传送序列图



说明：S=Star（起始条件），Sr=重复的起始条件，P=Stop（停止条件），A=响应，NA=非响应，EVx=事件（EVTITEN=1时产生中断）

EV5：STARTF=1，读STS1然后将地址写入DT寄存器将清除该事件。
EV6：ADDRF=1，读STS1然后读STS2将清除该事件。在10位主接收模式下，该事件后应设置CTRL2的STARTGEN=1。
EV6_1：没有相对应的事件标志，只适于接收1个字节的情况恰好在EV6之后（即清除了ADDRF之后），要清除响应 和设置停止条件的产生位。
EV7：RDNE=1，读DT寄存器清除该事件。
EV7_1：RDNE=1，读DT寄存器清除该事件。设置ACKEN=0和STOP请求。
EV9：ADDR10F=1，读STS1然后写入DT寄存器将清除该事件。

注意：

1.如果收到一个单独的字节，则是 NA。

2.EV5、EV6 和 EV9 事件拉长 SCL 低电平，直到对应的软件序列结束。

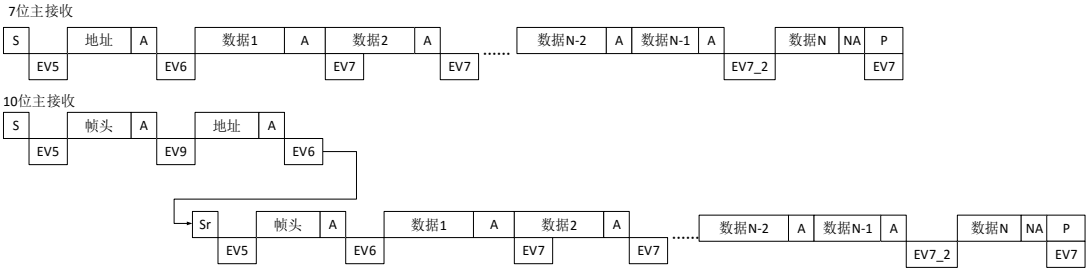
3.EV7 的软件序列必须在当前字节传输结束前完成。

4.EV6_1 或 EV7_1 的软件序列必须在当前传输字节的 ACK 脉冲之前完成。

第二种情况：I2C 中断未设为最高优先级并且接收字节总数 N>2

在这种条件下，接收到数据 N-2 时并不读取，当 N-1 也收到后，时钟被拉低，通信暂停。此时清 ACKEN 位，然后依次读数据 N-2，设置 STOPGEN/STARTGEN，读数据 N-1，再待 RDNE 置位后，读数据 N，如下图所示：

图 15-7 N>2时主收流程图



说明：S=Star（起始条件），Sr=重复的起始条件，P=Stop（停止条件），A=响应，NA=非响应，EVx=事件（EVTITEN=1时产生中断）

EV5：STARTF=1，读STS1然后将地址写入DT寄存器将清除该事件。
EV6：ADDRF=1，读STS1然后读STS2将清除该事件。在10位主接收模式下，该事件后应设置CTRL2的STARTGEN=1。
EV7：RDNE=1，读DT寄存器清除该事件。
EV7_2：BTFF=1，这时DT中为N-2，移位寄存器中为N-1，设置ACKEN=0，读DT中的数据N-2，设置STOPGEN=1然后读N-1
EV9：ADDR10F=1，读STS1然后写入DT寄存器将清除该事件。

注：1-EV5、EV6和EV9事件拉长SCL低的时间，直到对应的软件序列结束。
2-EV7的软件操作流程必须在当前字节传输结束前完成，否则要使用BTFF标志来读取数据以降低通信速度

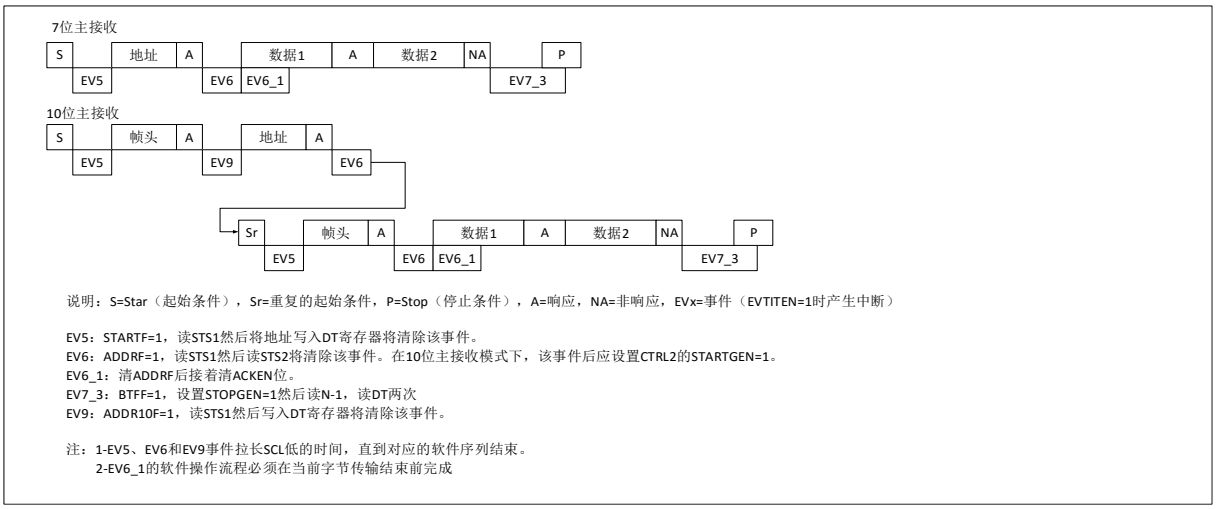
当剩下三个字节时软件流程如下：

- RDNE=1 => 不做任何操作
- 收到数据N-1
- BTFF=1，DT中N-2，移位寄存器中N-1，时钟被拉低
- 清ACKEN位
- 读数据N-2，然后总线会开始接收数据N
- 收到数据N，并回复NACK
- 设置STARTGEN/STOPGEN位
- RDNE=1
- 读数据N

第三种情况：I2C 中断未设为最高优先级并且接收字节总数 N=2 或 N=1

- 接收一个字节
  - ADDRF事件时清ACKEN位
  - 清ADDRF位
  - 设置STARTGEN/STOPGEN位
  - RDNE置位时读取数据
- 接收两个字节
  - 设置POSEN和ACKEN位
  - 等待ADDRF置位
  - 清ADDRF位
  - 清ACKEN位
  - 等待BTFF置位
  - 置STOPGEN位
  - 读两次DT寄存器

图 15-8 N=2时主收流程图

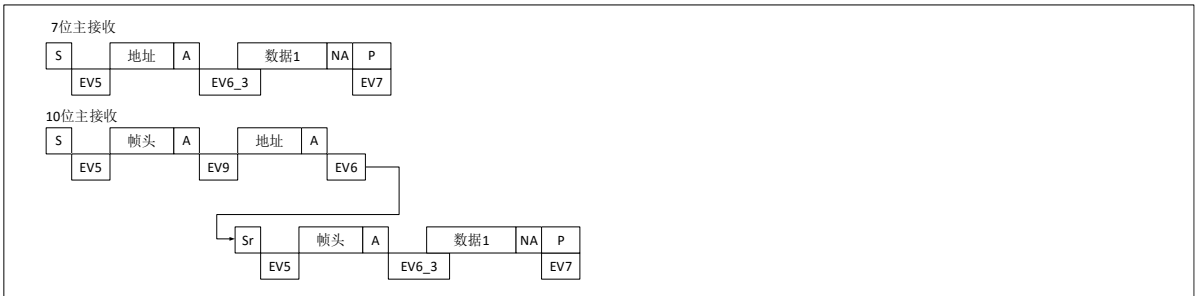


说明：S=Star（起始条件），Sr=重复的起始条件，P=Stop（停止条件），A=响应，NA=非响应，EVx=事件（EVTITEN=1时产生中断）

EV5：STARTF=1，读STS1然后将地址写入DT寄存器将清除该事件。
EV6：ADDRF=1，读STS1然后读STS2将清除该事件。在10位主接收模式下，该事件后应设置CTRL2的STARTGEN=1。
EV6_1：清ADDRF后接着清ACKEN位。
EV7_3：BTFF=1，设置STOPGEN=1然后读N-1，读DT两次
EV9：ADDR10F=1，读STS1然后写入DT寄存器将清除该事件。

注：1-EV5、EV6和EV9事件拉长SCL低的时间，直到对应的软件序列结束。
2-EV6_1的软件操作流程必须在当前字节传输结束前完成

图 15-9 N=1时主收流程图



说明：S=Star（起始条件），Sr=重复的起始条件，P=Stop（停止条件），A=响应，NA=非响应，EVx=事件（EVTITEN=1时产生中断）

EV5：STARTF=1，读STS1然后将地址写入DT寄存器将清除该事件。
EV6：ADDRF=1，读STS1然后读STS2将清除该事件。在10位主接收模式下，该事件后应设置CTRL2的STARTGEN=1。
EV6_3：清ADDRF后接着清ACKEN位，设置STOPGEN=1。
EV7：RDEN=1，读DT寄存器将清除该事件。
EV9：ADDR10F=1，读STS1然后写入DT寄存器将清除该事件。

注：1-EV5、EV6和EV9事件拉长SCL低的时间，直到对应的软件序列结束。
2-EV6_3的软件操作流程必须在当前字节传输结束前完成

15.3.4 错误条件

以下条件可能造成通讯失败。

总线错误(BUSERR)

在一个地址或数据字节传输期间，当 I²C 接口检测到一个外部的停止或起始条件则产生总线错误。此时：

- BUSERR位被置位为'1'；如果设置了ERRITEN位，则产生一个中断
- 在从模式情况下，数据被丢弃，硬件释放总线
  - 如果是错误的开始条件，从设备必须清BUSERR位，然后等待下一次开始条件
  - 如果是错误的停止条件，从设备按正常的停止条件操作，同时硬件释放总线
- 在主模式情况下，硬件不释放总线，同时不影响当前的传输状态。此时由软件决定是否要中止当前的传输

应答错误(ACKFAIL)

当接口检测到一个无应答位时，产生应答错误。此时：

- ACKFAIL位被置位，如果设置了ERRITEN位，则产生一个中断
- 当发送器接收到一个NACK时，必须复位通讯
  - 如果是处于从模式，硬件释放总线
  - 如果是处于主模式，软件必须生成一个停止条件或者重复的起始条件

仲裁丢失(ARLOST)

当 I²C 接口检测到仲裁丢失时产生仲裁丢失错误，此时：

- ARLOST位被硬件置位，如果设置了ERRITEN位，则产生一个中断
- I2C接口自动回到从模式(MSF位被清除)。当I2C接口丢失了仲裁，则它无法在同一个传输中响应它的从地址，但它可以在赢得总线的主设备发送重起始条件之后响应
- 硬件释放总线

过载/欠载错误(OVRUN)

在从模式下，如果禁止时钟延长，I²C接口正在接收数据时，当它已经接收到一个字节(RDNE=1)，但在 DT 寄存器中前一个字节数据还没有被读出，则发生过载错误。此时：

- 最后接收的数据被丢弃
- 在过载错误时，软件应清除RDNE位，发送器应该重新发送最后一次发送的字节

在从模式下，如果禁止时钟延长，I2C 接口正在发送数据时，在下一个字节的时钟到达之前，新的数据还未写入 DT 寄存器(TDE=1)，则发生欠载错误。此时：

- 在DT寄存器中的前一个字节将被重复发出
- 用户应该确定在发生欠载错时，接收端应丢弃重复接收到的数据。发送端应按I2C总线标准在规定的时间更新DT寄存器

在发送第一个字节时，必须在清除 ADDRF 之后并且第一个 SCL 上升沿之前写入 DT 寄存器；如果不能做到这点，则接收方应该丢弃第一个数据。

15.3.5 SDA/SCL线控制

- 如果允许时钟延长：
  - 发送器模式：如果TDE=1且BTFF=1：I2C接口在传输前保持时钟线为低，以等待软件读取STS1，然后把数据写进数据寄存器(缓冲器和移位寄存器都是空的)
  - 接收器模式：如果RDNE=1且BTFF=1：I2C接口在接收到数据字节后保持时钟线为低，以等待软件读STS1，然后读数据寄存器DT(缓冲器和移位寄存器都是满的)
- 如果在从模式中禁止时钟延长
  - 如果RDNE=1，在接收到下个字节前DT还没有被读出，则发生过载错。接收到的最后一个字节丢失
  - 如果TDE=1，在必须发送下个字节之前却没有新数据写进DT，则发生欠载错。相同的字节将被重复发出
  - 不控制重复写冲突

15.3.6 SMBus

介绍

系统管理总线(SMBus)是一个双线接口。通过它，各设备之间以及设备与系统的其他部分之间可以互相通信。它基于 I²C 操作原理。SMBus 为系统和电源管理相关的任务提供一条控制总线。一个系统利用 SMBus 可以和多个设备互传信息，而不需使用独立的控制线路。

系统管理总线(SMBus)标准涉及三类设备。从设备：接收或响应命令的设备。主设备：用来发送命令、产生时钟和终止发送的设备。主机：一种专用的主设备，它提供与系统 CPU 的主接口。主机必须具有主-从机功能并且必须支持 SMBus 提醒协议。一个系统里只允许有一个主机。

SMBus 和 I²C 之间的相似点

- 2条线的总线协议(1个时钟，1个数据) + 可选的SMBus提醒线
- 主-从通信，主设备提供时钟
- 多主机功能
- SMBus数据格式类似于I2C的7位地址格式(见图 15-1)

SMBus 和 I2C 之间的不同点

下表列出了 SMBus 和 I²C 的不同点。

表 15-1 SMBus与I²C的比较

SMBus	I²C
最大传输速度 100kHz	最大传输速度 400kHz
最小传输速度 10kHz	无最小传输速度
35ms 时钟低超时	无时钟超时
固定的逻辑电平	逻辑电平由 VDD 决定
不同的地址类型(保留的、动态的等)	7 位、10 位和广播呼叫从地址类型
不同的总线协议(快速命令、处理呼叫等)	无总线协议

SMBus 应用用途

利用系统管理总线，设备可提供制造商信息，告诉系统它的型号/部件号，保存暂停事件的状态，报告不同类型的错误，接收控制参数，和返回它的状态。SMBus 为系统和电源管理相关的任务提供控制总线。

设备标识

在系统管理总线上，任何一个作为从模式的设备都有一个唯一的地址，叫做从地址。保留的从地址表请参考 2.0 版的 SMBus 规范(*http：//smbus.org/specs/*)。

总线协议

SMBus 技术规范支持 9 个总线协议。有关这些协议的详细资料和 SMBus 地址类型，请参考 2.0 版 的 SMBus 规范(*http：//smbus.org/specs/*)。这些协议由用户的软件来执行。

地址解析协议(ARP)

通过给每个从设备动态地分配一个新的唯一地址，可以解决 SMBus 的从地址冲突。地址解析协 议(ARP)具有以下的特性：

- 使用标准SMBus物理层仲裁机制分配地址
- 当设备维持供电期间，分配的地址仍保持不变，也允许设备在断电后保留其地址
- 在地址分配后，没有额外的SMBus的打包开销(也就是说访问分配地址的设备与访问固定地址的设备所用时间是一样的)
- 任何一个SMBus主设备可以遍历总线

唯一的设备标识符(UDID)

为了分配地址，需要一种区分每个设备的机制，每个设备必须拥有一个唯一的设备标识符。

关于在 ARP 上 128 位的 UDID 的详细信息，参考 2.0 版的 SMBus 规范(http：//smbus.org/specs/)。

SMBus 提醒模式

SMBus 提醒是一个带中断线的可选信号，用于那些希望扩展它们的控制能力而牺牲一个引脚的设备。SMBALERT 和 SCL、SDA 信号一样，是一种线与信号。SMBALERT 通常和 SMBus 广播呼叫地址一起使用。与 SMBus 有关的消息为 2 字节。

一个只具有从功能的设备，可以通过设置 I2C_CTRL1 寄存器上的 SMBALERT 位，使用 SMBALERT 给主机发信号表示它希望进行通信。主机处理该中断并通过提醒响应地址 ARA(Alert Response Address，地址值为 0001100x)访问所有 SMBALERT 设备。只有那些将 SMBALERT 拉低的设备能应答 ARA。此状态是由 I2C_STS1 寄存器中的 SMBALERTF 状态标记来标识的。主机执行一个修改过的接收字节操作。由从发送设备提供的 7 位设备地址被放在字节的 7 个最高位上，第八个位可以是'0'或'1'。

如果多个设备把 SMBALERT 拉低，最高优先级设备(最小的地址)将在地址传输期间通过标准仲裁赢得通信权。在确认从地址后，此设备不得再拉低它的 SMBALERT，如果当信息传输完成后，主机仍看到 SMBALERT 低，就知道需要再次读 ARA。

没有实现 SMBALERT 信号的主机可以定期访问 ARA。

有关 SMBus 提醒模式的更多详细资料，请参考 2.0 版的 SMBus 规范(*http：//smbus.org/specs/*)。

超时错误

在定时规范上 I^2C 和 SMBus 之间有很多差别。

SMBus 定义了一个时钟低超时，35ms 的超时。SMBus 规定 TLOW：SEXT 为从设备的累积时钟低扩展时间。SMBus 规定 TLOW：MEXT 为主设备的累积时钟低扩展时间。更多超时细节请参考 2.0 版的 SMBus 规范(*http：//smbus.org/specs/*)。

I2C_STS1 中的状态标志 Timeout 或 Tlow 错误表明了这个特性的状态。

如何使用 SMBus 模式的接口

为了从 I^2C 模式切换到 SMBus 模式，应该执行下列步骤：

- 设置I2C_CTRL1寄存器中的SMBMODE位
- 按应用要求配置I2C_CTRL1寄存器中的SMBTYPE和ARPEN位

如果要把设备配置成主设备，产生起始条件的步骤见 15.3.3 节 I^2C 主模式。否则，参见 15.3.2 节 I^2C 从模式。

软件程序必须处理多种 SMBus 协议。

- 如果ARPEN=1且SMBTYPE=0，使用SMB设备默认地址
- 如果ARPEN=1且SMBTYPE=1，使用SMB主设备头字段
- 如果SMBALERTF=1，使用SMB提醒响应地址

15.3.7 DMA请求

DMA 请求(当被使能时)仅用于数据传输。发送时数据寄存器变空或接收时数据寄存器变满，则产生 DMA 请求。DMA 请求必须在当前字节传输结束之前被响应。当为相应 DMA 通道设置的数据传输量已经完成时，DMA 控制器发送传输结束信号 EOT 到 I^2C 接口，并且在中断允许时产生一个传输完成中断：

- 主发送器：在EOT中断服务程序中，需禁止DMA请求，然后在等到BTFF事件后设置停止条件
- 主接收器：当要接收的数据数目大于或等于2时，DMA控制器发送一个硬件信号EOT_1，它对应DMA传输(字节数－1)。如果在I2C_CTRL2寄存器中设置了DMALAST位，硬件在发送完 EOT_1后的下一个字节，将自动发送NACK。在中断允许的情况下，用户可以在DMA传输完成的中断服务程序中产生一个停止条件

利用 DMA 发送

通过设置 I2C_CTRL2 寄存器中的 DMAEN 位可以激活 DMA模式。只要 TDE位被置位，数据将由 DMA 从预置的存储区装载进 I2C_DT 寄存器。为 I^2C 分配一个 DMA 通道，须执行以下步骤(x 是通道号)：

1. 在 DMA_CPBAx 寄存器中设置 I2C_DT 寄存器地址。数据将在每个 TDE 事件后从存储器传送至这个地址。
2. 在 DMA_CMBAx 寄存器中设置存储器地址。数据在每个 TDE 事件后从这个存储区传送至 I2C_DT。
3. 在 DMA_TCNTx 寄存器中设置所需的传输字节数。在每个 TDE 事件后，此值将被递减。
4. 利用 DMA_CHCTRLx 寄存器中的 CHPL[0：1]位配置通道优先级。
5. 设置 DMA_CHCTRLx 寄存器中的 DIR 位，并根据应用要求可以配置在整个传输完成一半或全部完成时发出中断请求。
6. 通过设置 DMA_CHCTRLx 寄存器上的 CHEN 位激活通道。

当 DMA 控制器中设置的数据传输数目已经完成时，DMA 控制器给 I2C 接口发送一个传输结束的 EOT / EOT_1 信号。在中断允许的情况下，将产生一个 DMA 中断。

注意： 如果使用 DMA 进行发送时，不要设置 I2C_CTRL2 寄存器的 BUFITEN 位。

利用 DMA 接收

通过设置 I2C_CTRL2 寄存器中的 DMAEN 位可以激活 DMA 接收模式。每次接收到数据字节时，将由 DMA 把 I2C_DT 寄存器的数据传送到设置的存储区(参考 DMA 说明)。设置 DMA 通道进行 I^2C 接收，须执行以下步骤(x 是通道号)：

1. 在 DMA_CPBAx 寄存器中设置 I2C_DT 寄存器的地址。数据将在每次 RDNE 事件后从此地址传送到存储区。
2. 在 DMA_CMBAx 寄存器中设置存储区地址。数据将在每次 RDNE 事件后从 I2C_DT 寄存器传送到此存储区。
3. 在 DMA_TCNTx 寄存器中设置所需的传输字节数。在每个 RDNE 事件后，此值将被递减。
4. 利用 DMA_CHCTRLx 寄存器中的 CHPL[0：1]配置通道优先级。
5. 清除 DMA_CHCTRLx 寄存器中的 DIR 位，根据应用要求可以设置在数据传输完成一半或全部完成时发出中断请求。
6. 设置 DMA_CHCTRLx 寄存器中的 CHEN 位激活该通道。

当 DMA 控制器中设置的数据传输数目已经完成时，DMA 控制器给 I2C 接口发送一个传输结束的 EOT/ EOT_1 信号。在中断允许的情况下，将产生一个 DMA 中断。

注意： 如果使用 DMA 进行接收时，不要设置 I2C_CTRL2 寄存器的 BUFITEN 位。

15.3.8 包错误校验(PEC)

包错误校验(PEC)计算器是用于提高通信的可靠性，这个计算器使用下述 CRC-8 多项式对每一位串行数据进行计算：

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

- PEC计算由I2C_CTRL1寄存器的PECEN位激活。PEC使用CRC-8算法对所有信息字节进行计算，包括地址和读/写位在内
  - 在发送时：在最后一个TDE事件后设置I2C_CTRL1寄存器的PECTRA传输位，PEC将在最后一个字节后被发送
  - 在接收时：在最后一个RDNE事件之后设置I2C_CTRL1寄存器的PECTRA位，如果下个接收到的字节不等于内部计算的PECVAL，接收器发送一个NACK。如果是主接收器，不管校对的结果如何， PEC后都将发送NACK。PECTRA位必须在接收当前字节的ACK脉冲之前设置
- 在I2C_STS1寄存器中可获得PECERR错误标记/中断
- 如果DMA和PEC计算器都被激活
  - 在发送时：当I2C接口从DMA控制器处接收到EOT信号时，它在最后一个字节后自动发送PECVAL
  - 在接收时：当I2C接口从DMA处接收到一个EOT_1信号时，它将自动把下一个字

节作为PECVAL，并且将检查它。在接收到PECVAL后产生一个DMA请求

- 为了允许中间PEC传输，在I2C_CTRL2寄存器中有一个控制位(DMALAST位)用于判别是否真是最后一个DMA传输。如果确实是最后一个主接收器的DMA请求，在接收到最后一个字节后自动发送NACK
- 仲裁丢失时PEC计算失效

15.3.9 I²C中断请求

下表列出了所有的 I²C 中断请求。

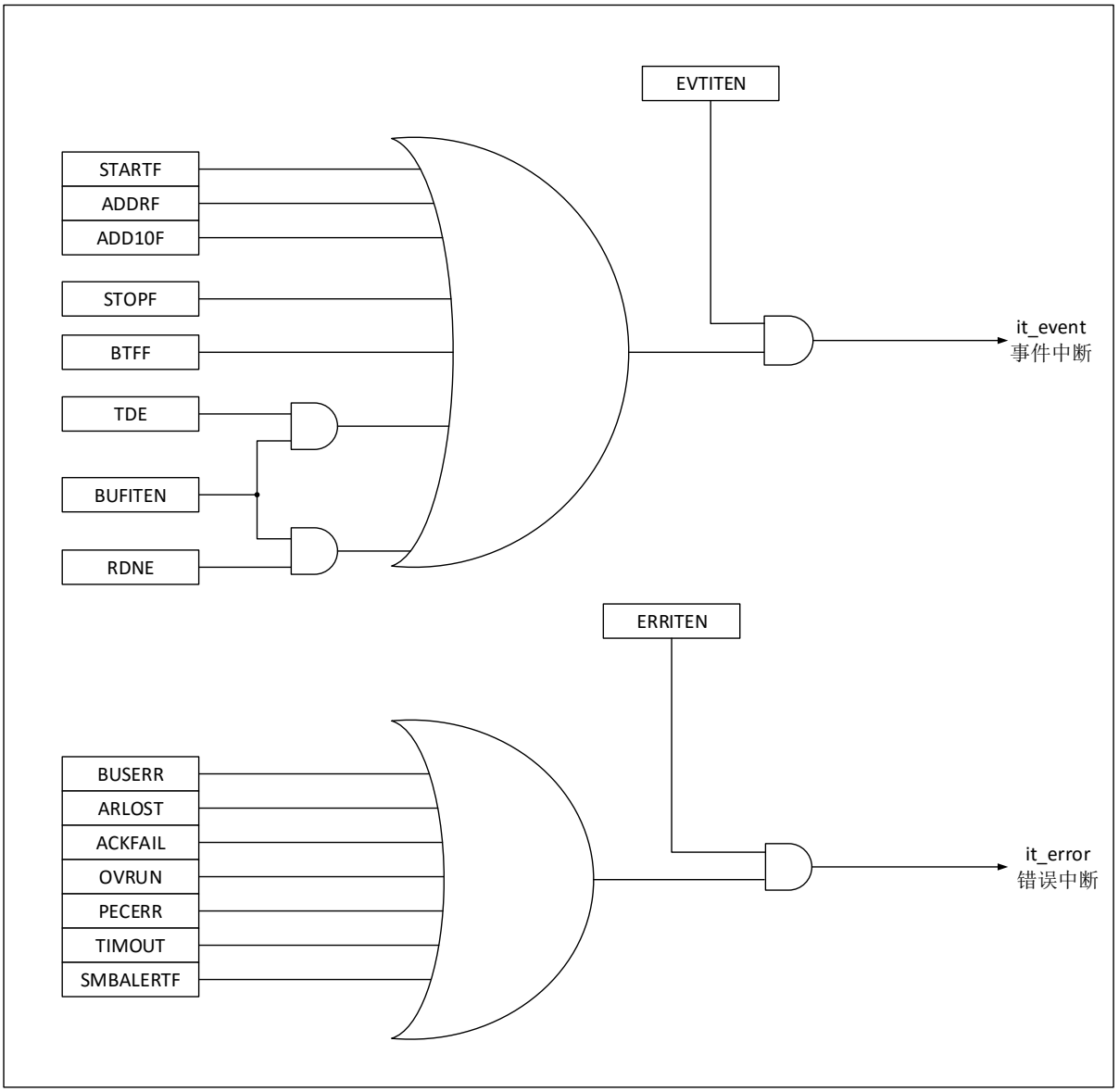
表 15-2 I²C中断请求表

中断事件	事件标志	开启控制位
起始位已发送(主)	STARTF	EVTITEN
地址已发送(主)或地址匹配(从)	ADDRF	
10 位头段已发送(主)	ADDR10F	
已收到停止(从)	STOPF	
数据字节传输完成	BTFF	
接收缓冲区非空	RDNE	EVTITEN 和 BUFITEN
发送缓冲区空	TDE	
总线错误	BUSERR	ERRITEN
仲裁丢失(主)	ARLOST	
响应失败	ACKFAIL	
过载/欠载	OVRUN	
PEC 错误	PECERR	
超时/Tlow 错误	TIMOUT	
SMBus 提醒	SMBALERTF	

注意： 1.STARTF、ADDRF、ADDR10F、STOPF、BTFF、RDNE 和 TDE 通过逻辑或汇到同一个中断通道中。

2.BUSERR、ARLOST、ACKFAIL、OVRUN、PECERR、TIMOUT 和 SMBALERTF 通过逻辑或汇到同一个中断通道中。

图 15-10 I²C中断映射图



15.3.10 I²C调试模式

当微控制器进入调试模式(Cortex™-M4F 核心处于停止状态)时，根据 DBG 模块中的 DBG_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。详见第 22.2.2 节。

15.4 I²C寄存器描述

可以用半字(16 位)或字(32 位)的方式操作这些外设寄存器。

表 15-3 I²C寄存器地址映像和复位值

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

000h	I2C_CTRL1	保留																											SWRESET	保留	SMBALERT	PECTRA	POSEN	ACKEN	STOPGEN	STARTGEN	NOCLKSTRECH	GCEN	PECEN	ARPEN	SMB TYPE	保留	SMBMODE	PEN
	复位值																										0		0	0	0	0	0	0	0	0	0	0	0		0	0
004h	I2C_CTRL2	保留																											保留															
	复位值																														DMALAST	DMAEN	BUFIT EN	EVTIT EN	ERRIT EN	CLKFREQ[7：0]								
																															0	0	0	0	0	0	0	0	0	0	0	0	0	0
008h	I2C_OADDR 1	保留																											ADDRMOD	保留					ADDR[9：8]		ADDR[7：1]							ADDR0
	复位值																												0						0	0	0	0	0	0	0	0	0	0
00Ch	I2C_OADDR 2	保留																											保留								ADDR2[7：1]							DUALEN
	复位值																																			0	0	0	0	0	0	0	0
010h	I2C_DT	保留																																		DT[7：0]							
	复位值																																			0	0	0	0	0	0	0	0
014h	I2C_STS1	保留																											SMBALERTF	TIMOUT	保留	PEC ERR	OVRUN	ACKFAIL	ARLOST	BUSERR	TDE	RDNE	保留	STOPF	ADDR10F	BTFF	ADDRF	STARTF
	复位值																												0	0		0	0	0	0	0	0	0		0	0	0	0	0
018h	I2C_STS2	保留																											PECVAL[7：0]								DUALF	SMB HOST	SMBDEFTADDRF	GCADDRF	保留	TRF	BUSYF	MSF
	复位值																												0	0	0	0	0	0	0	0	0	0	0	0		0	0	0
01Ch	I2C_CLKCT RL	保留																											F/SMODE	FMDUTY	保留	CLKCTRL[11：0]												
	复位值																												0	0		0	0	0	0	0	0	0	0	0	0	0	0	0
020h	I2C_TMRIS E	保留																																							TMRISE[5：0]					
	复位值																																							0	0	0	0	1	0

15.4.1 控制寄存器1(I²C_CTRL1)

地址偏移：0x00

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SW RESET	保留	SMB ALERT	PEC TRA	POS EN	ACK EN	STOP GEN	START GEN	NOCL KSTR ETCH	GCEN	PEC EN	ARP EN	SMB TYPE	保留	SMB MODE	PEN
rw	res	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	res	rw	rw

位 15	**SWRESET**：软件复位(Software reset) 当被置位时，I²C 处于复位状态。在复位该位前确信 I²C 的引脚被释放，总线是空的。 0：I²C 模块不处于复位状态； 1： I²C 模块处于复位状态。 **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	保留位，硬件强制为 0
位 13	**SMBALERT**：SMBus 提醒(SMBus alert) 软件可以设置或清除该位；当 PEN=0 时，由硬件清除。 0：释放 SMBAlert 引脚使其变高。提醒响应地址头紧跟在 NACK 信号后面； 1：驱动 SMBAlert 引脚使其变低。提醒响应地址头紧跟在 ACK 信号后面。
位 12	**PECTRA**：数据包出错检测(Packet error checking) 软件可以设置或清除该位；当传送 PECVAL 后，或起始或停止条件时，硬件将其清除。 0：无 PEC 传输； 1：PEC 传输(在发送或接收模式)。 **注**：仲裁丢失时，PEC 的计算失效。
位 11	**POSEN**：应答/PEC 位置(用于数据接收) (Acknowledge/PEC Position (for data reception)) 软件可以设置或清除该位。 0：ACKEN 位控制当前移位寄存器内正在接收的字节的(N)ACK。 PECTRA 位表明当前移位寄存器内的字节是 PECVAL； 1：ACKEN 位控制在移位寄存器里接收的下一个字节的(N)ACK。PECTRA 位表明在移位寄存器里接收的下一个字节是 PECVAL。 **注**：POSEN 位只能用在 2 字节的接收配置中，必须在接收数据之前配置。 为了 NACK 第 2 个字节，必须在清除 ADDRF 为之后清除 ACKEN 位。 为了检测第 2 个字节的 PECVAL，必须在配置了 POSEN 位之后，拉伸 ADDRF、事件时设置 PECTRA 位。
位 10	**ACKEN**：应答使能(Acknowledge enable) 软件可以设置或清除该位。 0：无应答返回； 1：在接收到一个字节后返回一个应答(匹配的地址或数据)。
位 9	**STOPGEN**：停止条件产生(Stop generation) 软件可以设置或清除该位；或当检测到停止条件时，由硬件清除；当检测到超时错误时，硬件将其置位。 在主模式下： 0：无停止条件产生； 1：当数据寄存器和移位寄存器传输完成后或在当前起始条件发出后产生停止条件。 在从模式下： 0：无停止条件产生； 1：在当前字节传输完成之后释放 SCL 和 SDA 线。 **注**：当设置了 STOPGEN、STARTGEN 或 PECTRA 位，在硬件清除这个位之前，软件不要执行任何对 I2C_CTRL1 的写操作；否则有可能会第 2 次设置 STOPGEN、STARTGEN 或 PECTRA 位。

位 8	**STARTGEN**：起始条件产生(Start generation) 软件可以设置或清除该位，或当起始条件发出后，由硬件清除。 在主模式下： 0：无起始条件产生； 1：重复产生起始条件。 在从模式下： 0：无起始条件产生； 1：当总线空闲时，产生起始条件。
位 7	**NOCLKSTRETCH**：禁止时钟延长(从模式) (Clock stretching disable (Slave mode)) 该位用于当 ADDRF 或 BTFF 标志被置位，在从模式下禁止时钟延长，直到它被软件复位。 0：允许时钟延长； 1：禁止时钟延长。
位 6	**GCEN**：广播呼叫使能(General call enable) 0：禁止广播呼叫。不应答地址 00h； 1：允许广播呼叫. 应答地址 00h。
位 5	**PECEN**：PEC 使能(PEC enable) 0：禁止 PEC 计算； 1：开启 PEC 计算。
位 4	**ARPEN**：ARP 使能(ARP enable) 0：禁止 ARP； 1：使能 ARP。 如果 SMBTYPE=0，使用 SMBus 设备的默认地址。 如果 SMBTYPE=1，使用 SMBus 的主地址。
位 3	**SMBTYPE**：SMBus 类型(SMBus type) 0：SMBus 设备； 1：SMBus 主机。
位 2	保留位，硬件强制为 0。
位 1	**SMBMODE**：SMBus 模式(SMBus mode) 0：I²C 模式； 1：SMBus 模式。
位 0	**PEN**：I²C 模块使能(Peripheral enable) 0：禁用 I²C 模块； 1：启用 I²C 模块：根据 SMBus 位的设置，相应的 I/O 口需配置为复用功能。 **注**：如果清除该位时通讯正在进行，在当前通讯结束后，I²C 模块被禁用并返回空闲状态。 由于在通讯结束后发生 PEN＝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 STARTGEN、STOPGEN 或 PECTRA 设置后，软件应该在相应位被硬件清零后写 I2C_CTRL1，否则有可能产生第二次 STARTGEN、STOPGEN 或 PECTRA 请求。

15.4.2 控制寄存器2(I²C_CTRL2)

地址偏移：0x04

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留			DMA LAST	DM AEN	BUF ITEN	EVT ITEN	ERR ITEN	CLKFREQ[7：0]							
res			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 15：13	保留位，硬件强制为 0
位 12	**DMALAST**：DMA 最后一次传输(DMA last transfer) 0：下一次 DMA 的 EOT 不是最后的传输； 1：下一次 DMA 的 EOT 是最后的传输。 **注**：该位在主接收模式使用，使得在最后一次接收数据时可以产生一个 NACK。
位 11	**DMAEN**：DMA 请求使能(DMA requests enable) 0：禁止 DMA 请求； 1：当 TDE=1 或 RDNE =1 时，允许 DMA 请求。
位 10	**BUFITEN**：缓冲器中断使能(Buffer interrupt enable) 0：当 TDE=1 或 RDNE=1 时，不产生任何中断； 1：当 TDE=1 或 RDNE=1 时，产生事件中断(不管 DMAEN 是何种状态)。
位 9	**EVTITEN**：事件中断使能(Event interrupt enable) 0：禁止事件中断； 1：允许事件中断。 在下列条件下，将产生该中断： – STARTF = 1 (主模式) – ADDRF = 1 (主/从模式) – ADDR10F= 1 (主模式) – STOPF = 1 (从模式) – BTFF = 1，但是没有 TDE 或 RDNE 事件 – 如果 BUFITEN = 1，TDE 事件为 1 – 如果 BUFITEN = 1，RDNE 事件为 1
位 8	**ERRITEN**：出错中断使能(Error interrupt enable) 0：禁止出错中断； 1：允许出错中断。 在下列条件下，将产生该中断： – BUSERR = 1 – ARLOST = 1 – ACKFAIL = 1 – OVRUN = 1 – PECERR = 1 – TIMOUT = 1 – SMBALERTF = 1
位 7：0	**CLKFREQ[7：0]**：I^2C 模块时钟频率(Peripheral 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： 00000000：禁用 00000001：禁用 00000010：2MHz ... 01111000：120MHz

15.4.3 自身地址寄存器1(I^2C_OADDR1)

地址偏移：0x08

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDRMODE	保留	保留				ADDR[9：8]		ADDR[7：1]							ADDR0
rw	res	res				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 15	**ADDRMODE**：寻址模式(从模式) (Addressing mode (slave mode)) 0：7 位从地址(不响应 10 位地址)； 1：10 位从地址(不响应 7 位地址)。
位 14：10	保留位，硬件强制为 0。
位 9：8	**ADDR[9：8]**：接口地址(Interface address) 7 位地址模式时不用关心。 10 位地址模式时为地址的 9~8 位。
位 7：1	**ADDR[7：1]**：接口地址(Interface address) 地址的 7~1 位。
位 0	**ADDR0**：接口地址 (Interface address) 7 位地址模式时不用关心。 10 位地址模式时为地址第 0 位。

15.4.4 自身地址寄存器2(I²C_OADDR2)

地址偏移：0x0C

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留								ADDR2[7：1]							DUALEN
res								rw	rw	rw	rw	rw	rw	rw	rw

位 15：8	保留位，硬件强制为 0
位 7：1	**ADDR2[7：1]**：接口地址(Interface address) 在双地址模式下地址的 7~1 位。
位 0	**DUALEN**：双地址模式使能位(Dual addressing mode 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。

15.4.5 数据寄存器(I²C_DT)

地址偏移：0x10

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留								DT[7：0]							
res								rw	rw	rw	rw	rw	rw	rw	rw

位 15：8	保留位，硬件强制为 0
位 7：0	**DT[7：0]**：8 位数据寄存器(8-bit data register) 用于存放接收到的数据或放置用于发送到总线的数据 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**：在从模式下，地址不会被拷贝进数据寄存器 DT； **注**：硬件不管理写冲突(如果 TDE=0，仍能写入数据寄存器)； **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。

15.4.6 状态寄存器1(I²C_STS1)

地址偏移：0x14

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SMBALERTF	TIMOUT	保留	PECERR	OVRUN	ACKFAIL	ARLOST	BUSERR	TDE	RDNE	保留	STOPF	ADDR10F	BTFF	ADDRF	STARTF
rcw0	rcw0	res	rcw0	rcw0	rcw0	rcw0	rcw0	r	r	res	r	r	r	r	r

位 15	**SMBALERTF**：SMBus 提醒(SMBus alert) 在 SMBus 主机模式下： 0：无 SMBus 提醒； 1：在引脚上产生 SMBALERTF 提醒事件。 在 SMBus 从机模式下： 0：没有 SMBAlert 响应地址头序列； 1：收到 SMBAlert 响应地址头序列至 SMBAlert 变低。 – 该位由软件写'0'清除，或在 PEN=0 时由硬件清除。
位 14	**TIMOUT**：超时或 Tlow 错误(Timeout or Tlow error) 0：无超时错误； 1：SCL 处于低已达到25ms(超时)；或者主机低电平累积时钟扩展时间超过 10ms(Tlow：mext)；或从设备低电平累积时钟扩展时间超过 25ms(Tlow：sext)。 – 当在从模式下设置该位：从设备复位通讯，硬件释放总线 – 当在主模式下设置该位：硬件发出停止条件 – 该位由软件写'0'清除，或在 PEN=0 时由硬件清除 **注**：这个功能仅在 SMBUS 模式下有效
位 13	保留位，硬件强制为 0。
位 12	**PECERR**：在接收时发生 PEC 错误(PEC Error in reception) 0： 无 PEC 错误：接收到 PEC 后接收器返回 ACK(如果 ACKEN=1)； 1： 有 PEC 错误：接收到 PEC 后接收器返回 NACK(不管 ACKEN 是什么值)。 – 该位由软件写'0'清除
位 11	**OVRUN**：过载/欠载(Overrun/Underrun) 0： 无过载/欠载； 1： 出现过载/欠载。 – 当 NOCLKSTRETCH=1 时，在从模式下该位被硬件置位，同时 – 在接收模式中当收到一个新的字节时(包括 ACK 应答脉冲)，数据寄存器里的内容还未被读出，则新接收的字节将丢失 – 在发送模式中当要发送一个新的字节时，却没有新的数据写入数据寄存器，同样的字节将被发送两次 – 该位由软件写'0'清除，或在 PEN=0 时由硬件清除 **注**：如果数据寄存器的写操作发生时间非常接近 SCL 的上升沿，发送的数据是不确定的，并发生保持时间错误。
位 10	**ACKFAIL**：应答失败(Acknowledge failure) 0：没有应答失败； 1：应答失败。 – 当没有返回应答时，硬件将置该位为'1' – 该位由软件写'0'清除，或在 PEN=0 时由硬件清除

位 9	**ARLOST**：仲裁丢失 (主模式) (Arbitration lost (master mode)) 0：没有检测到仲裁丢失； 1：检测到仲裁丢失。当接口失去对总线的控制给另一个主机时，硬件将将置该位为'1'。 – 该位由软件写'0'清除，或在 PEN=0 时由硬件清除 在 ARLOST 事件之后，I^2C 接口自动切换回从模式(M/SL=0)。 **注**：在 SMBUS 模式下，在从模式下对数据的仲裁仅仅发生在数据阶段，或应答传输区间(不包括地址的应答)。
位 8	**BUSERR**：总线出错(Bus error) 0：无起始或停止条件出错； 1：起始或停止条件出错。 – 当接口检测到错误的起始或停止条件，硬件将该位置'1' – 该位由软件写'0'清除，或在PEN=0时由硬件清除。 在发生BUSERR错误时，软件应及时清除该标志位，然后才能保证重新正常通信。
位 7	**TDE**：数据寄存器为空 (发送时) (Data register empty (transmitters)) 0：数据寄存器非空； 1：数据寄存器空。 – 在发送数据时，数据寄存器为空时该位被置'1'，在发送地址阶段不设置该位 – 软件写数据到 DT 寄存器可清除该位；或在发生一个起始或停止条件后，由硬件自动清除 如果收到一个 NACK，或下一个要发送的字节是 PECVAL(PECTRA=1)，该位不被置位。 **注**：在写入第 1 个要发送的数据后，或设置了 BTFF 时写入数据，都不能清除 TDE 位，这是因为数据寄存器仍然为空。
位 6	**RDNE**：数据寄存器非空 (接收时) (Data register not empty (receivers)) 0：数据寄存器为空； 1：数据寄存器非空。 – 在接收时，当数据寄存器不为空，该位被置'1'。在接收地址阶段，该位不被置位 – 软件对数据寄存器的读写操作清除该位，或当 PEN=0 时由硬件清除 在发生 ARLOST 事件时，RDNE 不被置位。 **注**：当设置了 BTFF 时，读取数据不能清除 RDNE 位，因为数据寄存器仍然为满。
位 5	保留位，硬件强制为 0
位 4	**STOPF**：停止条件检测位 (从模式) (Stop detection (slave mode)) 0：没有检测到停止条件； 1：检测到停止条件。 – 在一个应答之后(如果 ACKEN=1)，当从设备在总线上检测到停止条件时，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**：在收到 NACK 后，STOPF 位不被置位。
位 3	**ADDR10F**：10 位头序列已发送 (主模式) (10-bit header sent (Master mode)) 0：没有 ADDR10F 事件发生； 1：主设备已经将第一个地址字节发送出去。 – 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 CR1 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	**BTFF**：字节发送结束(Byte transfer finished) 0：字节发送未完成； 1：字节发送结束。 当 NOCLKSTRETCH=0 时，在下列情况下硬件将该位置'1'： – 在接收时，当收到一个新字节(包括 ACK 脉冲)且数据寄存器还未被读取(RDNE=1) – 在发送时，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(TDE=1) – 在软件读取 STS1 寄存器后，对数据寄存器的读或写操作将清除该位；或在传输中发送一个起始或停止条件后，由硬件清除该位 **注**：在收到一个 NACK 后，BTFF 位不会被置位。 如果下一个要传输的字节是 PECVAL(I2C_STS2 寄存器中 TRF 为'1'，同时 I2C_CTRL1 寄存器中 PECTRA 为'1')，BTFF 位不会被置位。
位 1	**ADDRF**：地址已被发送(主模式)/地址匹配(从模式)(Address sent (master mode)/matched(slave mode)) 在软件读取 STS1 寄存器后，对 STS2 寄存器的读操作将清除该位 **地址匹配(从模式)** 0：地址不匹配或没有收到地址； 1：收到的地址匹配。 – 当收到的从地址与 OADDR 寄存器中的内容相匹配、或发生广播呼叫、或 SMBus 设备默认地址 或 SMBus 主机识别出 SMBus 提醒时，硬件就将该位置'1'(当对应的设置被使能时) **地址已被发送(主模式)** 0：地址发送没有结束； 1：地址发送结束。 – 10 位地址模式时，当收到地址的第二个字节的 ACK 后该位被置'1' – 7 位地址模式时，当收到地址的 ACK 后该位被置'1' **注**：在收到 NACK 后，ADDRF 位不会被置位。
位 0	**STARTF**：起始位(主模式)(Start bit(Master mode)) 0：未发送起始条件； 1：起始条件已发送。 – 当发送出起始条件时该位被置'1' – 软件读取 STS1 寄存器后，写数据寄存器的操作将清除该位

15.4.7 状态寄存器2(I²C_STS2)

地址偏移：0x18

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PECVAL[7：0]								DUALF	SMBHOSTADDRF	SMBDEFTADDRF	GCADDRF	保留	TRF	BUSYF	MSF
r	r	r	r	r	r	r	r	r	r	r	r	res	r	r	r

位 15：8	**PECVAL[7：0]**：数据包出错检测(Packet error checking register) 当 PECEN=1 时，PECVAL[7：0]存放内部的 PEC 的值，当 PECEN 重置时清零。
位 7	**DUALF**：双标志(从模式)(Dual flag (Slave mode)) 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。 – 在产生一个停止条件或一个重复的起始条件时，或 PEN=0 时，硬件将该位清除

位 6	**SMBHOSTADDRF**： SMBus 主机头系列(从模式) (SMBus host header (Slave mode)) 0：未收到 SMBus 主机的地址； 1：当 SMBTYPE=1 且 ARPEN=1 时，收到 SMBus 主机地址。 – 在产生一个停止条件或一个重复的起始条件时，或 PEN=0 时，硬件将该位清除
位 5	**SMBDEFTADDRF**：SMBus 设备默认地址(从模式) (SMBus device default address (Slave mode)) 0：未收到 SMBus 设备的默认地址； 1：当 ARPEN=1 时，收到 SMBus 设备的默认地址。 – 在产生一个停止条件或一个重复的起始条件时，或 PEN=0 时，硬件将该位清除
位 4	**GCADDRF**：广播呼叫地址(从模式) (General call address (Slave mode)) 0：未收到广播呼叫地址； 1：当 GCEN=1 时，收到广播呼叫的地址。 – 在产生一个停止条件或一个重复的起始条件时，或 PEN=0 时，硬件将该位清除
位 3	保留位，硬件强制为 0
位 2	**TRF**：发送/接收(Transmitter/receiver) 0：接收到数据； 1：数据已发送； 在整个地址传输阶段的结尾，该位根据地址字节的 R/W 位来设定 在检测到停止条件(STOPF=1)、重复的起始条件或总线仲裁丢失(ARLOST=1)后，或当 PEN=0 时，硬件将其清除。
位 1	**BUSYF**：总线忙(Bus busy) 0：在总线上无数据通讯； 1：在总线上正在进行数据通讯。 – 在检测到 SDA 或 SCl 为低电平时，硬件将该位置’1’ – 当检测到一个停止条件时，硬件将该位清除 该位指示当前正在进行的总线通讯，当接口被禁用(PEN=0)时该信息仍然被更新。
位 0	**MSF**：主从模式(Master/slave) 0：从模式； 1：主模式。 – 当接口处于主模式(STARTF=1)时，硬件将该位置位 – 当总线上检测到一个停止条件、仲裁丢失(ARLOST=1 时)、或当 PEN=0 时，硬件清除该位

15.4.8 时钟控制寄存器(I²C_CLKCTRL)

地址偏移：0x1C

复位值：0x0000

注意 1.要求 FPCLK1 应当是 10MHz 的整数倍，这样可以正确地产生 400KHz 的快速时钟。

2.CLKCTRL 寄存器只有在关闭 I²C 时(PEN=0)才能设置

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
F/S MODE	FM DUTY	保留		CLKCTRL[11：0]											
rw	rw	res		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 15	**F/SMODE**：I²C 主模式选项 (I²C master mode selection) 0：标准模式的 I²C； 1：快速模式的 I²C。
位 14	**FMDUTY**：快速模式时的占空比 (Fast mode duty cycle) 0：快速模式下：T_{low}/T_{high} = 2； 1：快速模式下：T_{low}/T_{high} = 16/9(见 CLKCTRL)。

位 13：12	保留位，硬件强制为 0。
位 11：0	**CLKCTRL[11：0]**：快速 / 标准模式下的时钟控制分频系数(主模式)(Clock control register in Fast/Standard mode (Master mode)) 该分频系数用于设置主模式下的 SCL 时钟，从模式不用配置或者根据公式频率配置和主模式相同。 在 I²C 标准模式或 SMBus 模式下： T_{high} = CLKCTRL ×T_{PCLK1} T_{low} = CLKCTRL×T_{PCLK1} 在 I²C 快速模式下： 如果 FMDUTY = 0： T_{high} = CLKCTRL ×T_{PCLK1} T_{low} = 2 × CLKCTRL × T_{PCLK1} 如果 FMDUTY = 1： (速度达 400kHz) T_{high} = 9 × CLKCTRL ×T_{PCLK1} T_{low} = 16 × CLKCTRL ×T_{PCLK1} 例如：在标准模式下，产生 100kHz 的 SCL 的频率： 如果 CLKFREQ = 08，T_{PCLK1} = 125ns，则 CLKCTRL 必须写入 0x28(40×125ns = 5000ns)。 **注**：1. 允许设定的最小值为 0x04，在快速 DUTY 模式下允许的最小值为 0x01； 2. $T_{high}=t_{r(SCL)}+t_{w(SCLH)}$，详见数据手册中对这些参数的定义； 3. $T_{low}=t_{f(SCL)}+t_{w(SCLL)}$，详见数据手册中对这些参数的定义； 4. 这些延时没有过滤器； 5. 只有在关闭 I²C 时(PEN = 0)才能设置 CLKCTRL 寄存器； 6. f_{CK} 应当是 10MHz 的整数倍，这样可以正确产生 400kHz 的快速时钟。

15.4.9 TMRISE寄存器(I²C_TMRISE)

地址偏移：0x20

复位值：0x0002

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留										TMRISE[5：0]					
res										rw					

位 15：6	保留位，硬件强制为 0
位 5：0	**TMRISE[5：0]**：在快速/标准模式下的最大上升时间(主模式)(Maximum rise time in Fast/Standard mode (Master mode)) 这些位必须设置为 I²C 总线规范里给出的最大的 SCL 上升时间，增长步幅为 1。 例如：标准模式中最大允许 SCL 上升时间为 1000ns。如果在 I2C_CTRL2 寄存器中 CLKFREQ[7：0]中的值等于 0x08 且 T_{PCLK1}=125ns，故 TMRISE[5：0]中必须写入 09h(1000ns/125 ns = 8+1)。滤波器的值也可以加到 TMRISE[5：0]内。 如果结果不是一个整数，则将整数部分写入 TMRISE[5：0]以确保 t_{HIGH} 参数。 **注**：只有当 I²C 被禁用(PEN=0)时，才能设置 TMRISE[5：0]。

16 通用同步异步收发器（USART）

16.1 USART介绍

通用同步异步收发器（USART）提供了一种灵活的方法与使用工业标准 NRZ 异步串行数据格式的外部设备之间进行全双工数据交换。USART 利用分数波特率发生器提供宽范围的波特率选择。

它支持同步单向通信和半双工单线通信，也支持 LIN（局部互连网），智能卡协议和 IrDA（红外数据组织）SIRENDEC 规范，以及 CTS/RTS（Clear To Send/Request To Send）硬件流操作。它还允许多处理器通信。

使用多缓冲器配置的 DMA 方式，可以实现高速数据通信。

16.2 USART主要特性

- 全双工异步通信
- 单线半双工通信
- NRZ 标准格式（Mark/Space）
- 可编程的波特率发生器
  - 发送和接收共用的可编程波特率，最高达 6.25MBits/s
- 可编程数据字长度（8 位或 9 位）
- 可配置的停止位-支持 1 或 2 个停止位
- LIN 主机有发送断开符的能力以及 LIN 从机有检测断开符的能力
  - 当 USART 硬件配置成 LIN 时，生成 13 位断开符；检测 10/11 位断开符
- 发送方为同步传输提供时钟
- IrDA SIR 编码器解码器
  - 在普通模式下支持 3/16 位的持续时间
- 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
- 可配置的DMA多缓冲器通信
  - 利用 DMA 缓冲接收/发送数据
- 单独的发送器和接收器使能位
- 检测标志
  - 接收缓冲器满
  - 发送缓冲器空
  - 传输结束标志
- 校验控制
  - 发送校验位
  - 对接收数据进行校验
- 四个错误检测标志
  - 溢出错误
  - 噪音错误
  - 帧错误
  - 校验错误
- 10 个带标志的中断源

— CTSF 改变
— LIN 断开符检测
— 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
— 噪音错误
— 校验错误

- 多处理器通信--如果地址不匹配，则进入静默模式
- 从静默模式中唤醒（通过空闲总线检测或地址检测）
- 两种唤醒接收器的方式：地址位（MSB，第9位），总线空闲

16.3 USART功能概述

接口通过三个引脚与其他设备连接在一起（见图 16-1）。任何 USART 双向通信至少需要两个脚：接收数据输入（RX）和发送数据输出（TX）。

RX：串行数据输入端。利用过采样技术识别数据和噪音以恢复数据。

TX：串行数据输出端。当发送器被禁止时，输出引脚恢复到它的 I/O 端口配置。当发送器被激活，并且不发送数据时，TX 引脚处于高电平。在单线和智能卡模式里，此 I/O 口被同时用于数据的发送和接收。

- 总线在发送或接收前应处于空闲状态
- 一个起始位
- 一个数据字（8 或 9 位），最低有效位在前
- 0.5，1，1.5，2 个停止位，表示数据帧的结束
- 使用分数波特率发生器——12 位整数和 4 位小数的表示方法。
- 一个状态寄存器（USART_STS）
- 数据寄存器（USART_DT）
- 一个波特率寄存器（USART_BAUDR），12 位的整数和 4 位小数
- 一个智能卡模式下的保护时间寄存器（GTVAL）

关于以上寄存器中每个位的具体定义，请参考寄存器描述第 16.6 节：USART 寄存器描述。

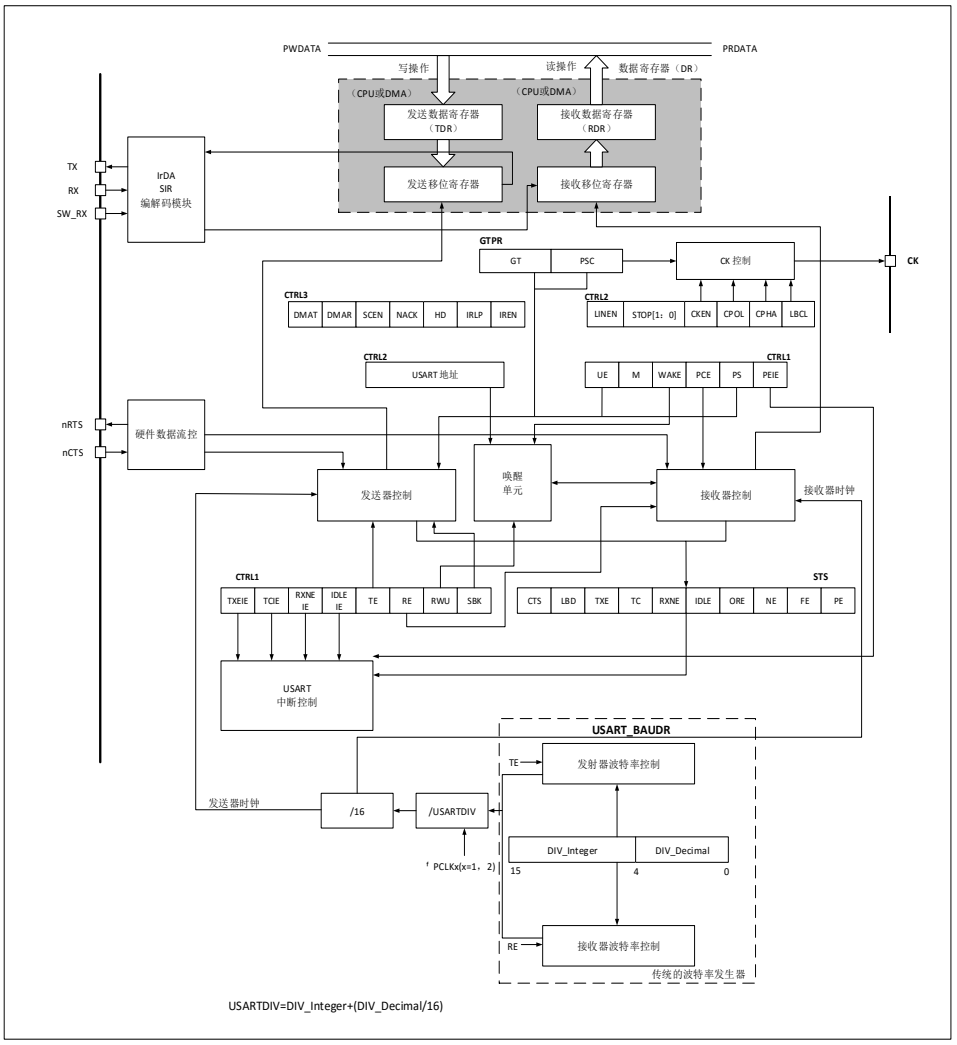
在同步模式中需要下列引脚：

- CK：发送器时钟输出。此引脚输出用于同步传输的时钟，（在 Start 位和 Stop 位上没有时钟脉冲，软件可选地，可以在最后一个数据位送出一个时钟脉冲）。数据可以在 RX 上同步被接收。这可以用来控制带有移位寄存器的外部设备（例如 LCD 驱动器）。时钟相位和极性都是软件可编程的。在智能卡模式里，CK 可以为智能卡提供时钟。

在硬件流控模式中需要下列引脚：

- CTS：接收清零，若为低电平，表明在当前数据传输结束时可继续下一次的数据发送；若为高电平，在当前数据传输结束时阻断下一次的数据发送。
- RTS：发送请求，若为低电平，表明 USART 准备好接收数据。

图 16-1 USART框图



16.3.1 USART特性描述

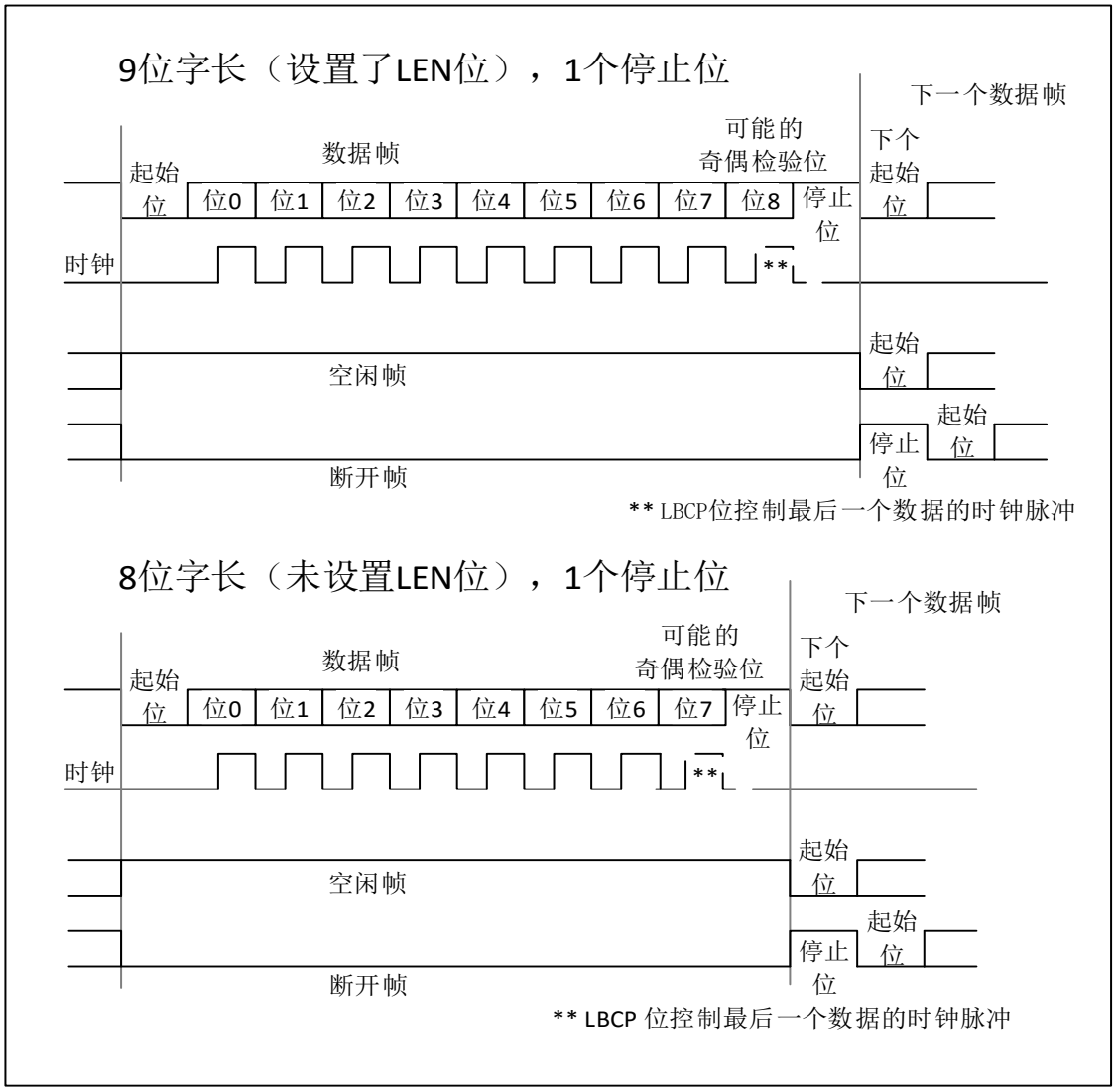
字长可以通过编程 USART_CTRL1 寄存器中的 LEN 位，选择成 8 或 9 位（见图 16-2）。在起始位期间，TX 脚处于低电平，在停止位期间处于高电平。

空闲帧：全部由’1’组成的一个完整的数据帧，也包含了数据的停止位。例如：如果 LEN=0，则空闲帧由 10 个’1’组成；如果 LEN=1，则空闲帧由 11 个’1’组成。

断开帧：被视为在一个帧周期内全部收到’0’（包括停止位期间，也是’0’）。在断开帧结束时，发送器再插入 1 或 2 个停止位（‘1’）来应答起始位。

发送和接收均由一个共用的波特时钟发生器驱动，当发送器和接收器的使能位分别置位时，分别为其产生波特时钟。

图 16-2 字长设置



注意： 在本章中，若未特殊说明，置位均表示某个寄存器被置为状态’1’,复位或清零均表示某个寄存器被置为状态’0’；硬件或者程序均可能置位或者清零某个寄存器，请参考本章具体内容。

16.3.2 发送器

发送器根据 LEN 位的状态发送 8 位或 9 位的数据字。当发送使能位（TEN）被置位时，发送移位寄存器中的数据在 TX 脚上输出；通过配置，相应的时钟脉冲在 CK 脚上输出。

16.3.2.1 字符发送

在 USART 发送期间，在 TX 引脚上首先移出数据的最低有效位。在此模式里，USART_DT 寄存器包含了一个内部总线和发送移位寄存器之间的缓冲器（见图 16-1）。

每个字符之前都有一个低电平的起始位；之后跟着的停止位，其数目可配置。

USART 支持多种停止位的配置：0.5、1、1.5 和 2 个停止位。

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，因为波特率计数器停止计数。正在传输的当前数据将丢失。

2. TEN 位被激活后,USART 将自动发送一个空闲帧。

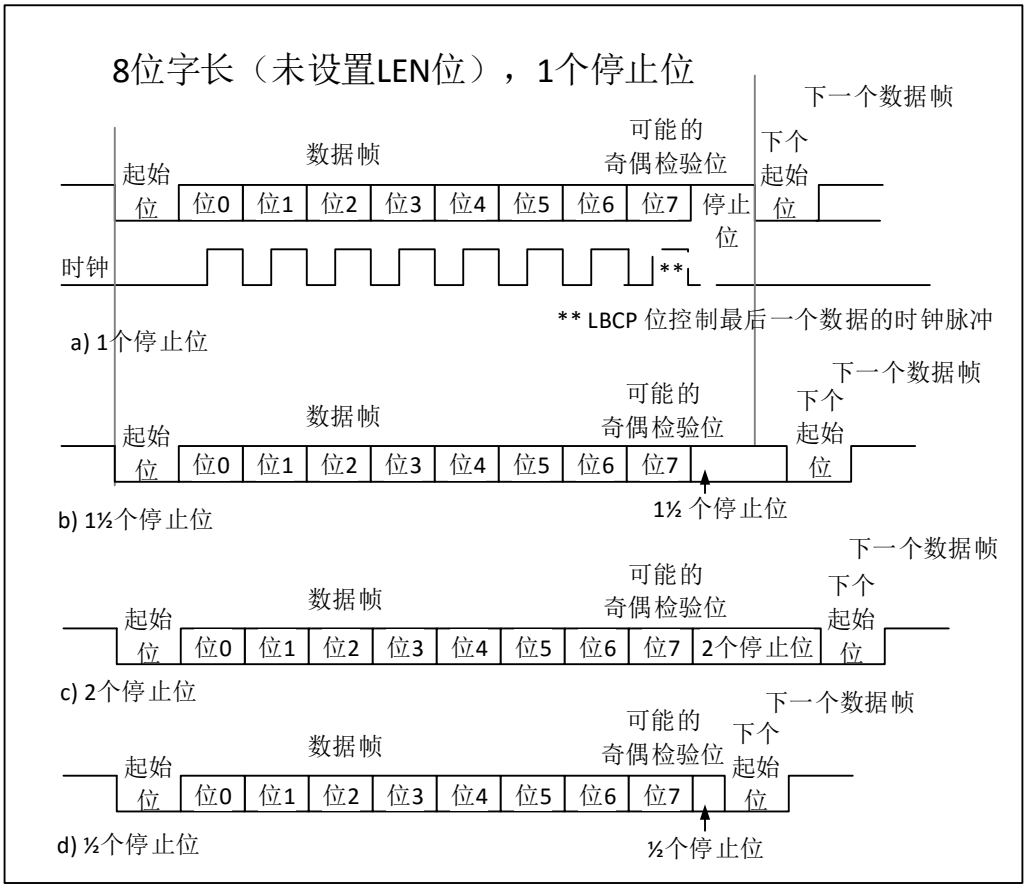
16.3.2.2 可配置的停止位

每个字符发送的停止位的位数可以通过控制寄存器 2（USART_CTRL2）的位 13、12 进行编程。

1. 1 个停止位：停止位位数的默认值。
2. 2 个停止位：可用于常规 USART 模式、单线模式以及调制解调器模式。
3. 0.5 个停止位：在智能卡模式下接收数据时使用。
4. 1.5 个停止位：在智能卡模式下发送和接收数据时使用。空闲帧包括了停止位。

断开帧是 10 位低电平，后跟停止位（当 LEN=0 时）；或者 11 位低电平，后跟停止位（LEN=1 时）。不可能传输更长的断开帧（长度大于 10 或者 11 位）。

图 16-3 配置停止位



配置步骤：

1. 通过对 USART_CTRL1 寄存器的 UEN 位写入 1 来激活 USART。
2. 配置 USART_CTRL1 的 LEN 位来定义字长。
3. 配置 USART_CTRL2 的 STOPB 位来定义停止位。
4. 如果采用多缓冲器通信，配置 USART_CTRL3 中的 DMA 使能位（DMATEN）。按多缓冲器通信中的描述配置 DMA 寄存器。
5. 配置 USART_BAUDR 寄存器来定义波特率。
6. 配置 USART_CTRL1 中的 TEN 位（也即向 TEN 位写入 1），USART 会自动发送一个空闲帧作为第一次数据发送。
7. 把要发送的数据写进 USART_DT 寄存器（此动作清除 TDE 位）。在只有一个缓冲器的情况下，对每个待发送的数据重复步骤 7。
8. 在 USART_DT 寄存器中写入最后一个数据字后，要等待 TRAC=1，它表示最后一个数据帧的传输结束。当需要关闭 USART 或需要进入停机模式之前，需要确认传输结束，避免破坏最后一次数据传输。

16.3.2.3 单字节通信

清零 TDE 位总是通过对数据寄存器的写操作来完成的。

TDE 位由硬件（也即 USART IP 核）来置位，这表明：

- 数据已经从 TDR 寄存器（Transmitter Data Register）移送到移位寄存器，数据发送已经开始。
- TDR 寄存器被清空。
- 下一个数据可以被写进 USART_DT 寄存器而不会覆盖先前的数据。

如果 TDEIEN 位被设置，此标志将产生一个中断。如果此时 USART 正在发送数据，对 USART_DT 寄存器的写操作把数据存入 TDR 寄存器，并在当前传输结束时把该数据复制到移位寄存器。

如果此时 USART 没有发送数据，处于空闲状态，对 USART_DT 寄存器的写操作直接把数据放进移位寄存器，数据传输开始，TDE 位立即被置起（也即写入“1”）。

当一帧发送完成时（停止位发送后）并且设置了 TDE 位，TRAC 位被置起，如果 USART_CTRL1 寄存器中的 TRACIEN 位被置起时，则会产生中断。

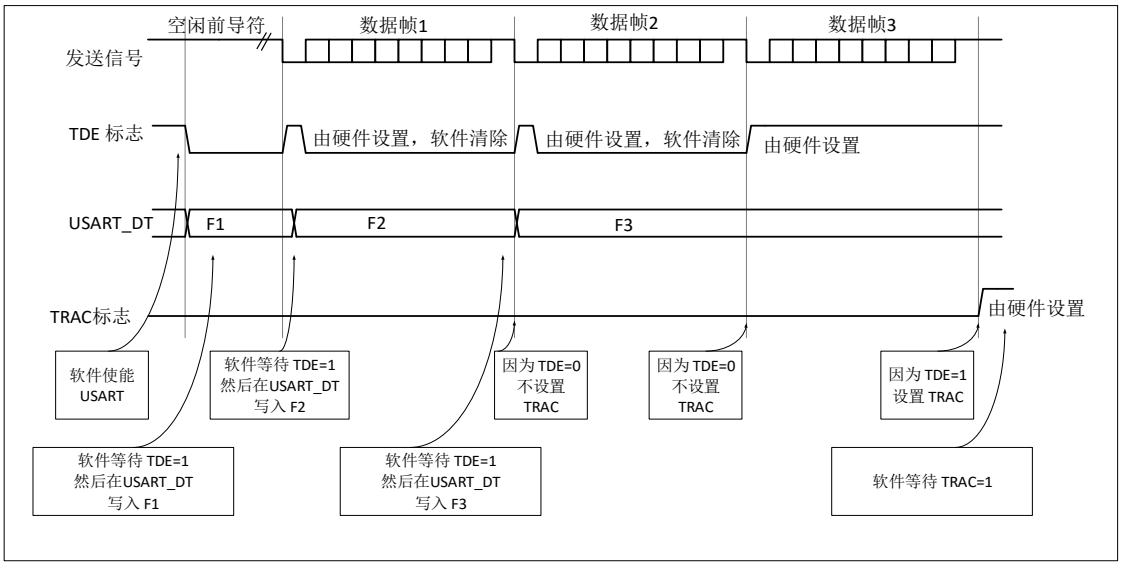
在 USART_DT 寄存器中写入了最后一个数据字后，在关闭 USART 模块之前或设置微控制器进入低功耗模式（见图 16-4）之前，必须先等待 TRAC=1。

使用下列软件过程清除 TRAC 位：

1. 读一次 USART_STS 寄存器；
2. 写一次 USART_DT 寄存器。

注意：　TRAC 位也可以通过软件对它写’0’来清除。此清零方式只推荐在 DMA 多缓冲器通信模式下使用。

图 16-4 发送时TRAC/TDE的变化情况



16.3.2.4 断开帧

设置 SBRK 可发送一个断开帧。断开帧长度取决 LEN 位（见图 16-2）。如果设置 SBRK=1，在完成当前数据发送后，将在 TX 线上发送一个断开帧。断开字符发送完成时（在断开帧的停止位时）SBRK 被硬件复位。USART 在最后一个断开帧的结束处插入一逻辑’1’，以保证能识别下一帧的起始位。

注意：　若在开始发送断开帧之前，软件又复位了 SBRK 位，断开帧将不被发送。如果要发送两个连续的断开帧，SBRK 位应该在前一个断开帧的停止位之后置起。

16.3.2.5 空闲符号

置位 TEN 将使得 USART 在第一个数据帧前发送一空闲帧。

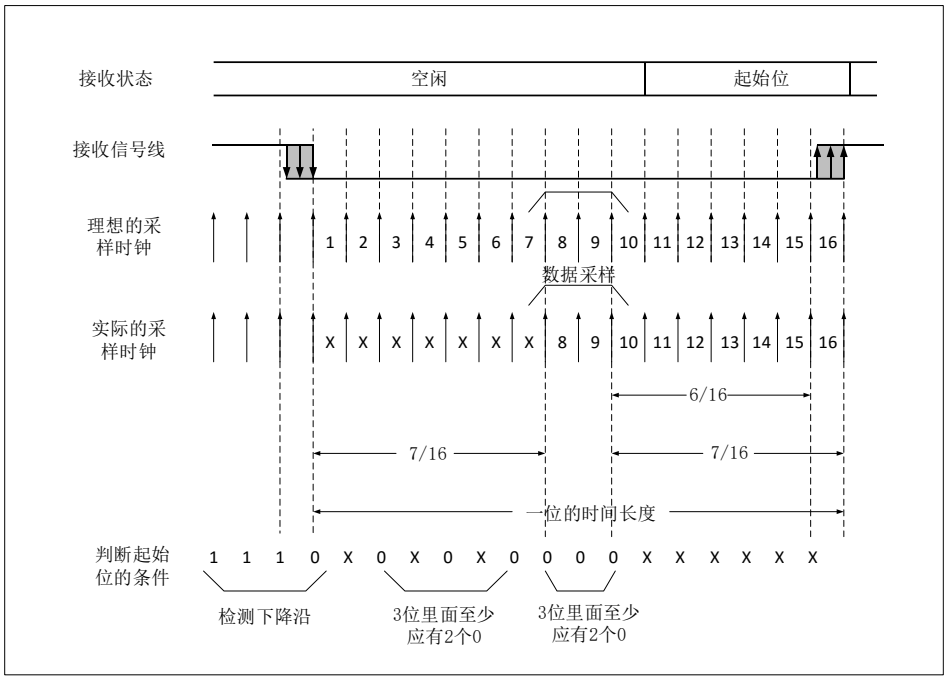
16.3.3 接收器

USART 可以根据 USART_CTRL1 的 LEN 位接收 8 位或 9 位的数据字。

16.3.3.1 起始位侦测

在 USART 中，如果辨认出一个特殊的采样序列，那么就认为侦测到一个起始位。该序列为：1110X0X0X0000。

图 16-5 起始位侦测



注意：如果该序列不完整，那么接收端将退出起始位侦测并回到空闲状态（不设置任何标志位）等待下降沿。

1. 在第 3、5、7 位的采样，以及在第 8、9、10 位的采样都为’0’（也即 6 个’0’），则确认收到起始位，并且不会置位 NERR 噪声标志位。

2. 第 3、5、7 位的采样有两个’0’,与此同时，第 8、9、10 位的采样有两个’0’点，那么确认收到起始位，但是会置位 NERR 噪声标志位。

3. 第 3、5、7 位的采样有两个’0’,与此同时，第 8、9、10 位的采样有三个’0’点，那么确认收到起始位，但是会置位 NERR 噪声标志位。

4. 第 3、5、7 位的采样有三个’0’,与此同时，第 8、9、10 位的采样有两个’0’点，那么确认收到起始位，但是会置位 NERR 噪声标志位。

5. 如果在第 3、5、7、8、9、10 位的采样值满足不了上面四种要求，则该 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。

16.3.3.2 字符接收

在 USART 接收期间，数据的最低有效位首先从 RX 脚移进。在此模式里，USART_DT 寄存器包含的缓冲器位于内部 APB 总线和接收移位寄存器之间。

配置步骤：

1. 通过对 USART_CTRL1 寄存器的 UEN 位写入 1 来激活 USART。
2. 配置 USART_CTRL1 的 LEN 位来定义字长。
3. 配置 USART_CTRL2 的 STOPB 位来定义停止位。
4. 如果采用多缓冲器通信，配置 USART_CTRL3 中的 DMA 使能位（DMAREN）。按多缓冲器通信中的描述配置 DMA 寄存器。
5. 配置 USART_BAUDR 寄存器来定义波特率。

6. 配置 USART_CTRL1 中的 REN 位（也即向 REN 位写入 1）。激活接收器，使它开始寻找起始位。

当一字符被接收到时：

- RDNE 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 RDNEIEN 位被设置，则产生中断。
- 在接收期间如果检测到帧错误，噪音或溢出错误，错误标志将被置起。
- 在多缓冲器通信时，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。
- 在单缓冲器模式里，由软件读 USART_DT 寄存器完成对 RDNE 位清除。RDNE 标志也可以通过对它写 0 来清除。RDNE 位必须在下一帧数据接收结束前被清零，以避免溢出错误。

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。

16.3.3.3 断开帧

当接收到一个断开帧时，USART 按处理帧错误一样处理它。

备注：在 LIN 模式下，当接收到一个断开帧时，将按断开帧处理，LBDF 将会置起。

16.3.3.4 空闲符号

当一空闲帧被检测到时，其处理步骤和接收到普通数据帧一样，但如果 IDLEIEN 位被设置将产生一个中断。

16.3.3.5 溢出错误

如果接收的数据未及时被读走，导致 RDNE 没有及时被复位，又接收到一个字符，则发生溢出错误。数据只有当 RDNE 位被清零后才能从移位寄存器转移到 RDR 寄存器。RDNE 标记是接收到每个字节后被硬件置位的。如果下一个数据已被收到或先前 DMA 请求还没被服务时，RDNE 标志仍是置起的，溢出错误产生。

当溢出错误产生时：

- ORERR 位被置位。
- RDR 内容将不会丢失。读 USART_DT 寄存器仍能得到先前的数据。
- 移位寄存器中的内容将被覆盖。随后接收到的数据都将丢失。
- 如果 RDNEIEN 位被设置或 ERRIEN 和 DMAREN 位都被设置，中断产生。
- 顺序执行对 USART_STS 和 USART_DT 寄存器的读操作，可复位 ORERR。

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

- *如果 RDNE=1，上一个有效数据还在接收寄存器 RDR 上，可以被读出。*
- *如果 RDNE=0，这意味着上一个有效数据已经被读走，RDR 已经没有东西可读。当上一个有效数据在 RDR 中被读取的同时又接收到新的（也就是丢失的）数据时，此种情况可能发生。在读序列期间（在 USART_STS 寄存器读访问和 USART_DT 读访问之间）接收到新的数据，此种情况也可能发生。*

噪音错误使用过采样技术（同步模式除外），通过区别有效输入数据和噪音来进行数据恢复。

图 16-6 检测噪声的数据采样

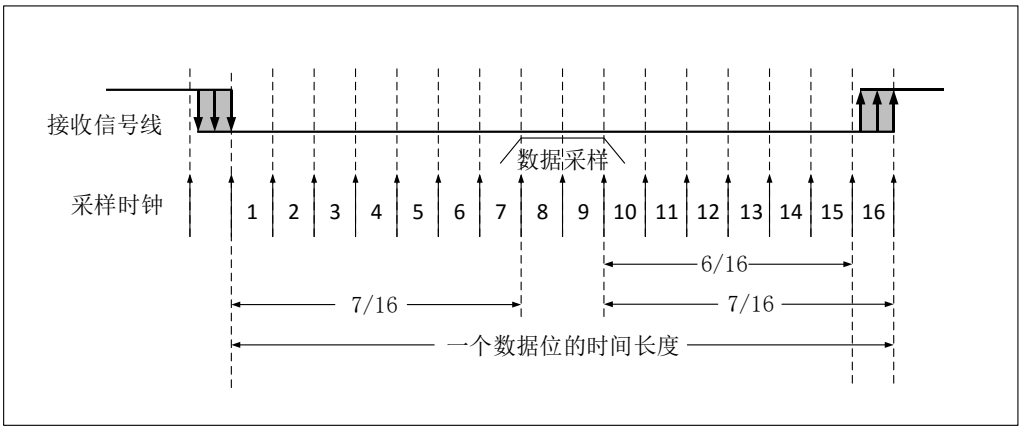


表 16-1 检测噪声的数据采样

采样值	**NERR** 状态	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

当在接收帧中检测到噪音时：

- 在RDNE位的上升沿设置NERR标志。
- 无效数据从移位寄存器传送到USART_DT寄存器。
- 在单个字节通信情况下，没有噪声中断产生。然而，因为NERR标志位和RDNE标志位是同时被置位，RDNE将产生中断。在多缓冲器通信情况下，如果已经设置了USART_CTRL3寄存器中ERRIEN位，将产生一个中断。

顺序读 USART_STS，再读 USART_DT 寄存器，将清除 NERR 标志位。

16.3.3.6 帧错误

当以下情况发生时检测到帧错误：由于数据未同步或大量噪音的原因，停止位没有在预期的时间上接收和识别出来。当帧错误被检测到时：

- FERR位被硬件置起
- 无效数据从移位寄存器传送到USART_DT寄存器。
- 在单字节通信时，没有帧错误中断产生。然而，这个位和RDNE位同时置起，后者将产生中断。在多缓冲器通信情况下，如果USART_CTRL3寄存器中ERRIEN位被置位的话，将产生中断。

顺序执行对 USART_STS 和 USART_DT 寄存器的读操作，可复位 FERR 位。

16.3.3.7 接收期间可配置的停止位

在数据接收期间，数据停止位的个数可以通过控制寄存器 2（USART_CTRL2 中的 STOPB）的控制位来配置，在普通模式时，可以是 1 或 2 个。在智能卡模式里可能是 0.5 或 1.5 个。

1. 0.5 个停止位（智能卡模式中的接收）：不对 0.5 个停止位进行采样。因此，如果选择 0.5 个停止位则不能检测帧错误和断开帧。
2. 1 个停止位：对 1 个停止位的采样在第 8，第 9 和第 10 采样点上进行。
3. 1.5 个停止位（智能卡模式）：当以智能卡模式发送时，器件必须检查数据是否被正确的发送出去。所以接收器功能块必须被激活（USART_CTRL1 寄存器中的 REN=1），并且在停止位的发送期间采样数据线上的信号。如果出现校验错误，智能卡会在发送方采样 NACK 信号时，即总线上停止位对应的时间内时，拉低数据线，以此表示出现了帧错误。发送端的 FERR 在 1.5 个停止位结束时被置起。对 1.5 个停止位的采样是在第 16，第 17 和第 18 采样点进行的。对于智能卡发送端而言，1.5 个的停止位可以被分成 2 部分：一个是 0.5 个数据位的周期，期间不做任何事情；随后是 1 个数据位的周期的停止位，在这段时间的中点处采样。对于智能卡接收端而言，1.5 个的停止位也可以被分成 2 部分：一个是 0.5 个数据位的周期，判断是否有奇偶校验错误；如果有奇偶校验错误，则在随后的 1 个数据位的周期拉低数据总线；如果没有奇偶校验错误，则在随后的 1 个数据位的周期，释放数据总线（数据总线处于高电平）。详见第 16.3.11 节：智能卡。
4. 2 个停止位：对 2 个停止位的采样是在第一停止位的第 8，第 9 和第 10 个采样点完成的。如果第一个停止位期间检测到一个帧错误，帧错误标志将被设置。第二个停止位不再检查帧错误。在第一个停止位结束时 RDNE 标志将被设置。

16.3.4 分数波特率的产生

接收器和发送器的波特率在 USARTDIV 的整数和小数寄存器中的值是相同的。

$$\text{TX/RX 波特率} = \frac{f_{CK}}{16 * \mathbf{USARTDIV}}$$

这里的f_{CK}是给外设的时钟（PCLK1 用于 USART2、3 和 UART4、5，PCLK2 用于 USART1、6 和 UART7、8）

USARTDIV 是一个无符号的定点数。该值设置在 USART_BAUDR 寄存器。

注意： *在写入 USART_BAUDR 之后，波特率计数器会被波特率寄存器的新值替换。因此，不要在通信过程中改变波特率寄存器的数值。*

16.3.4.1 如何从USART_BAUDR寄存器值得到USARTDIV

例 1：

如果 DIV_Integer=27，DIV_Decimal=12（USART_BAUDR=0x1BC），于是

DIV_Integer（USARTDIV）=27

DIV_Decimal（USARTDIV）=12/16=0.75

所以 USARTDIV=27.75

例 2：

要求 USARTDIV=25.62,就有：

DIV_Decimal=16*0.62=9.92

最接近的整数是：10=0x0A

DIV_Integer= DIV_Integer（25.620）=25=0x19

于是，USART_BAUDR=0x19A

例 3：

要求 USARTDIV=50.99 就有：

DIV_Decimal=16*0.99=15.84

最接近的整数是：16=0x10=> DIV_Decimal[3：0]溢出=>进位必须加到小数部分

DIV_Integer= DIV_Integer（0d50.990+进位）=51=0x33

于是：USART_BAUDR=0x330，USARTDIV=0d51.000

表 16-2 设置波特率时的误差计算

波特率		fPCLK=36MHz			fPCLK=72MHz		
序号	Kbps	实际	置于波特率寄存器中的值	误差%	实际	置于波特率寄存器中的值	误差%
1	2.4	2.4	937.5	0%	2.4	1875	0%
2	9.6	9.6	234.375	0%	9.6	468.75	0%
3	19.2	19.2	117.1875	0%	19.2	234.375	0%
4	57.6	57.6	39.0625	0%	57.6	78.125	0%
5	115.2	115.384	19.5	0.15%	115.2	39.0625	0%
6	230.4	230.769	9.75	0.16%	230.769	19.5	0.16%
7	460.8	461.538	4.875	0.16%	461.538	9.75	0.16%
8	921.6	923.076	2.4375	0.16%	923.076	4.875	0.16%
9	2250	2250	1	0%	2250	2	0%
10	4500	不可能	不可能	不可能	4500	1	0%
11	6250	不可能	不可能	不可能	不可能	不可能	不可能
12	7500	不可能	不可能	不可能	不可能	不可能	不可能

波特率		fPCLK=100MHz			fPCLK=120MHz		
序号	Kbps	实际	置于波特率寄存器中的值	误差%	实际	置于波特率寄存器中的值	误差%
1	2.4	2.40004	2604.125	0%	2.4	3125	0%
2	9.6	9.60061	651	0%	9.6	781.25	0%
3	19.2	19.2012	325.5	0%	19.2	390.625	0%
4	57.6	57.6037	108.5	0%	57.61	130.1875	0%
5	115.2	115.207	54.25	0%	115.16	65.125	0%
6	230.4	230.415	27.125	0.01%	230.33	32.5625	0%
7	460.8	460.829	13.5625	0.01%	461.54	16.25	0.16%
8	921.6	925.926	6.75	0.47%	923.08	8.125	0.16%
9	2250	2272.73	2.75	1%	2264.15	3.3125	0.63%
10	4500	4545.45	1.375	1%	4444.44	1.6875	1.23%
11	6250	6250	1	0%	6315.79	1.1875	1.05%
12	7500	不可能	不可能	不可能	7500	1	0%

注意： *1. CPU 的时钟频率越低，则某一特定波特率的误差也越低。可以达到的波特率上限可以由这组数据得到。*

2. USART1、6 和 UART7、8 *使用 PCLK2（最高 120MHz）。其它 USART2*、3 和 UART4、5 *使用 PCLK1（最高 120MHz）。*

16.3.5 USART接收器容忍时钟的变化

只有当整体的时钟系统地变化小于 USART 异步接收器能够容忍的范围，USART 异步接收器才能正常地工作。影响这些变化的因素有：

- DTRA：由于发送器误差而产生的变化（包括发送器端振荡器的变化）
- DQUANT：接收器端波特率取整所产生的误差
- DREC：接收器端振荡器的变化
- DTCL：由于传输线路产生的变化（通常是由于收发器在由低变高的转换时序，与由高变低转换时序之间的不一致性所造成）。

需要满足：DTRA + DQUANT + DREC + DTCL < USART 接收器的容忍度

对于正常接收数据，USART 接收器的容忍度等于最大能容忍的变化，它依赖于下述选择：

- 由 USART_CTRL1 寄存器的 LEN 位定义的 10 或 11 位字符长度
- 是否使用分数波特率产生

表 16-3 当DIV_Decimal等于0时，USART接收器的容忍度

M 位	认为 NF 是错误	不认为 NF 是错误
0	3.75%	4.375%
1	3.41%	3.97%

表 16-4 当DIV_Decimal不等于0时，USART接收器的容忍度

M 位	认为 NF 是错误	不认为 NF 是错误
0	3.33%	3.88%
1	3.03%	3.53%

注意： 在特殊的情况下，即当收到的帧包含一些在 LEN=0 时，正好是 10 位（LEN=1 时是 11 位）的空闲帧，上面 2 个表格中的数据可能会有少许出入。

16.3.6 多处理器通信

通过 USART 可以实现多处理器通信（将几个 USART 连在一个网络里）。例如某个 USART 设备可以是主，它的 TX 输出和其他 USART 从设备的 RX 输入相连接；USART 从设备各自的 TX 输出逻辑地与在一起，并且和主设备的 RX 输入相连接。

在多处理器配置中，通常希望只有被寻址的接收器才被激活，来接收随后的数据，这样就可以减少由未被寻址的接收器的参与带来的多余的 USART 服务开销。未被寻址的设备可启用其静默功能置于静默模式。在静默模式里：

- 任何接收状态位都不会被置位。
- 所有接收中断被禁止。
- USART_CTRL1 寄存器中的 RECMUTE 位被置 1。RECMUTE 可以被硬件自动控制或在某个条件下由软件写入。

根据 USART_CTRL1 寄存器中的 WUMODE 位状态，USART 可以用二种方法进入或退出静默模式。

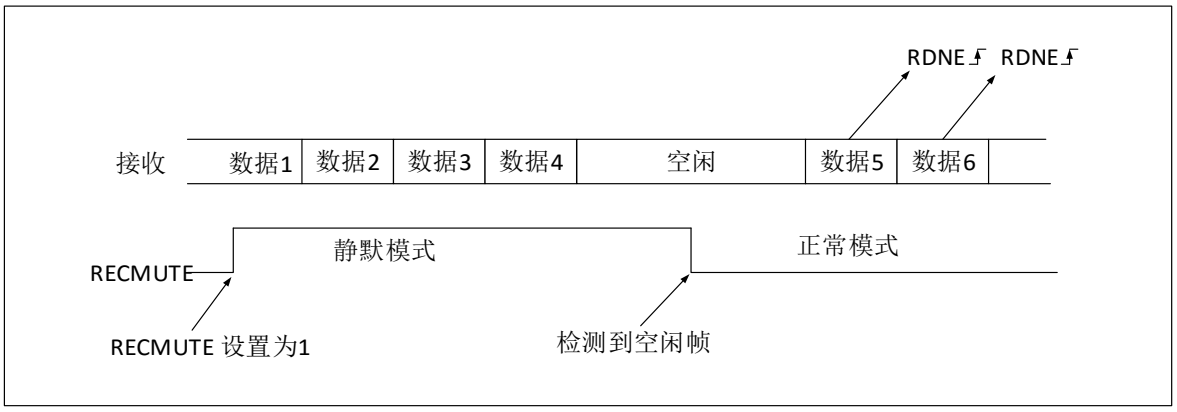
- 如果 WUMODE 位被复位：进行空闲总线检测。
- 如果 WUMODE 位被置位：进行地址标记检测。

16.3.6.1 空闲总线检测（WUMODE=0）

当 RECMUTE 位被写 1 时，USART 进入静默模式。当检测到一空闲帧时，它被唤醒。然后，RECMUTE 被硬件清零，但是 USART_STS 寄存器中的 IDLEF 位并不置起。RECMUTE 还可以被软件写 0。下图给

出利用空闲总线检测来唤醒和进入静默模式的一个例子。

图 16-7 利用空闲总线检测的静默模式



16.3.6.2 地址标记（address mark）检测（WUMODE=1）

在这个模式里，如果 MSB 是 1，该字节被认为是地址，否则被认为是数据。在一个地址字节中，目标接收器的地址被放在 4 个 LSB 中。这个 4 位地址被接收器同它自己地址做比较，接收器的地址被编程在 USART_CTRL2 寄存器的 ADDR。

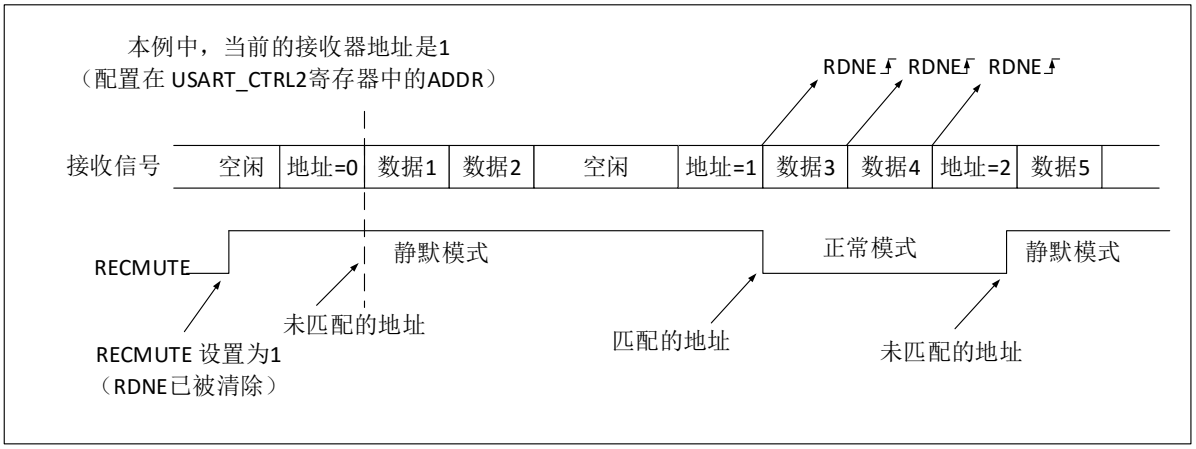
关闭奇偶校验功能时，当 LEN=0 时，MSB 是 USART_DT[7]，当 LEN=1 时，MSB 是 USART_DT[8]；开启奇偶校验功能时，当 LEN=0 时，MSB 是 USART_DT[6]，当 LEN=1 时，MSB 是 USART_DT[7]。

如果接收到的字节与它的编程地址不匹配时，USART 进入静默模式。此时，硬件设置 RECMUTE 位。接收该字节既不会设置 RDNE 标志也不会产生中断或发出 DMA 请求，因为 USART 已经在静默模式。

当接收到的字节与接收器内编程地址匹配时，USART 退出静默模式。然后 RECMUTE 位被清零，随后的字节被正常接收。收到这个匹配的地址字节时将设置 RDNE 位，因为 RECMUTE 位已被清零。

当接收缓冲器不包含数据时（USART_STS 的 RDNE=0），RECMUTE 位可以被写 0 或 1。否则，该次写操作被忽略。下图给出利用地址标记检测来唤醒和进入静默模式的例子。

图 16-8 利用地址标记检测的静默模式



16.3.7 校验控制

设置 USART_CTRL1 寄存器上的 PCEN 位，可以使能奇偶控制（发送时生成一个奇偶位，接收时进行奇偶校验）。根据 LEN 位定义的帧长度，可能的 USART 帧格式列在下表中。

表 16-5 帧格式

M 位	**PCEN** 位	**USART** 帧
0	0	|起始位|8 位数据|停止位|

0	1	|起始位|7 位数据|奇偶检验位|停止位|
1	0	|起始位|9 位数据|停止位|
1	1	|起始位|8 位数据|奇偶检验位|停止位|

注意：地址标记唤醒设备时，地址的匹配只考虑到数据的 MSB 位，而不用关心校验位。（MSB 是数据位中最后发出的，后面紧跟校验位或者停止位。）

偶校验：校验位使得一帧中的7或8个LSB数据以及校验位中’1’的个数为偶数。例如：数据=00110101，有4个’1’，如果选择偶校验（在USART_CTRL1中的PSEL＝0），校验位将是’0’。

奇校验：此校验位使得一帧中的7或8个LSB数据以及校验位中’1’的个数为奇数。例如：数据=00110101，有4个’1’，如果选择奇校验（在USART_CTRL1中的PSEL＝1），校验位将是’1’。

传输模式：如果USART_CTRL1的PCEN位被置位，写进数据寄存器的数据的MSB位被校验位替换后发送出去（如果选择偶校验偶数个’1’，如果选择奇校验奇数个’1’）。如果奇偶校验失败，USART_STS寄存器中的PERR标志被置’1’，并且如果USART_CTRL1寄存器的PERRIEN在被预先设置的话，中断产生。

16.3.8 LIN（局域互联网）模式

LIN 模式是通过设置 USART_CTRL2 寄存器的 LINEN 位选择。在 LIN 模式下，下列控制寄存器必须保持为 0：

- USART_CTRL2 寄存器的 CLKEN 位，STOPB[1：0]，
- USART_CTRL3 寄存器的 SCMEN，HALFSEL 和 IRDAEN

16.3.8.1 LIN发送

16.3.2 节里所描述的同样步骤适用于 LIN 主发送，但和普通 USART 发送有以下区别：

- 清零 LEN 位以配置 8 位字长
- 置位 LINEN 位以进入 LIN 模式。这时，置位 SBRK 将发送 13 位’0’作为断开帧；断开帧包括 13 位’0’和一位’1’，以允许对下一个开始位的检测。

16.3.8.2 LIN接收

当 LIN 模式被使能时，断开帧检测电路被激活。该检测完全独立于 USART 接收器。断开帧只要一出现就能检测到，不管是在总线空闲时还是在接收某数据帧期间（例如，数据帧还未完成，又插入了断开帧的发送）。

当接收器被激活时（USART_CTRL1 的 REN=1），电路监测 RX 上的起始信号。监测起始位的方法同检测断开帧或数据帧是一样的。当起始位被检测到后，电路对每个接下来的位，在每个位的第 8，9，10 个过采样时钟点上进行采样。如果 10 个（当 USART_CTRL2 的 LBDLEN=0）或 11 个（当 USART_CTRL2 的 LBDLEN=1）连续位都是’0’，并且又跟着一个定界符（定界符，也即是 RX 的上升沿），USART_STS 的 LBDF 标志被设置。如果 LBDIEN 位=1，中断产生。在确认断开帧前，要检查定界符，因为它意味 RX 线已经回到高电平。

如果在第 10 或 11 个采样点之前采样到了’1’，检测电路取消当前检测并重新寻找起始位。如果 LIN 模式被禁止，接收器继续如普通 USART 那样工作，并不检测断开帧。如果 LIN 模式没有被激活（LINEN=0），接收器仍然正常工作于 USART 模式，不会进行断开检测。如果 LIN 模式被激活（LINEN=1），只要一发生帧错误（也就是停止位检测到’0’，这种情况出现在断开帧），接收器就停止，直到断开帧检测电路接收到一个’1’（这种情况发生于断开帧没有完整的发出来），或一个定界符（这种情况发生于已经检测到一个完整的断开帧）。图 16-9 说明了断开帧检测器状态机的行为和断开帧标志的关系。图 16-10 给出了一个断开帧的例子。

图 16-9 LIN模式下的断开检测（11位断开长度 - 设置了LBDLEN位）

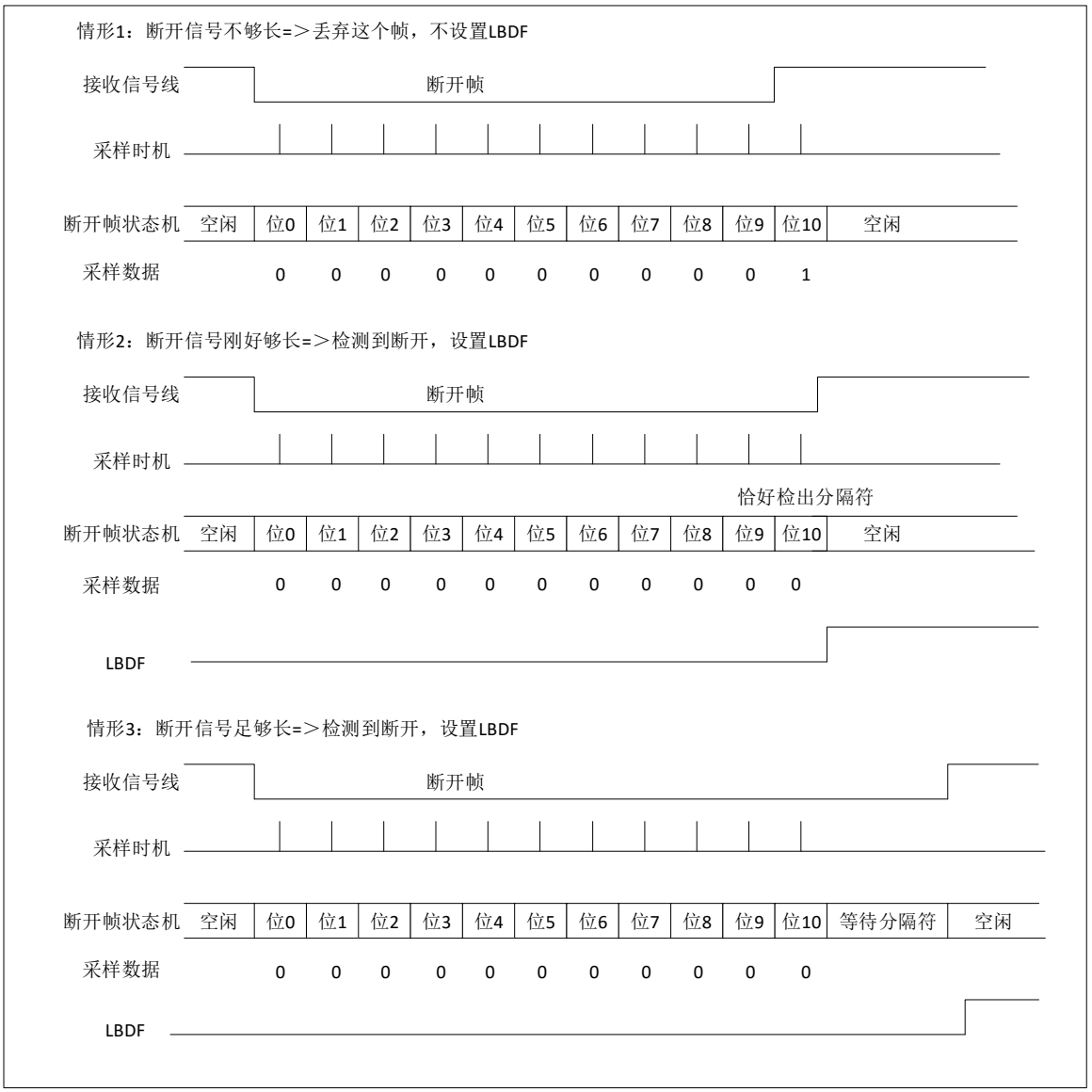
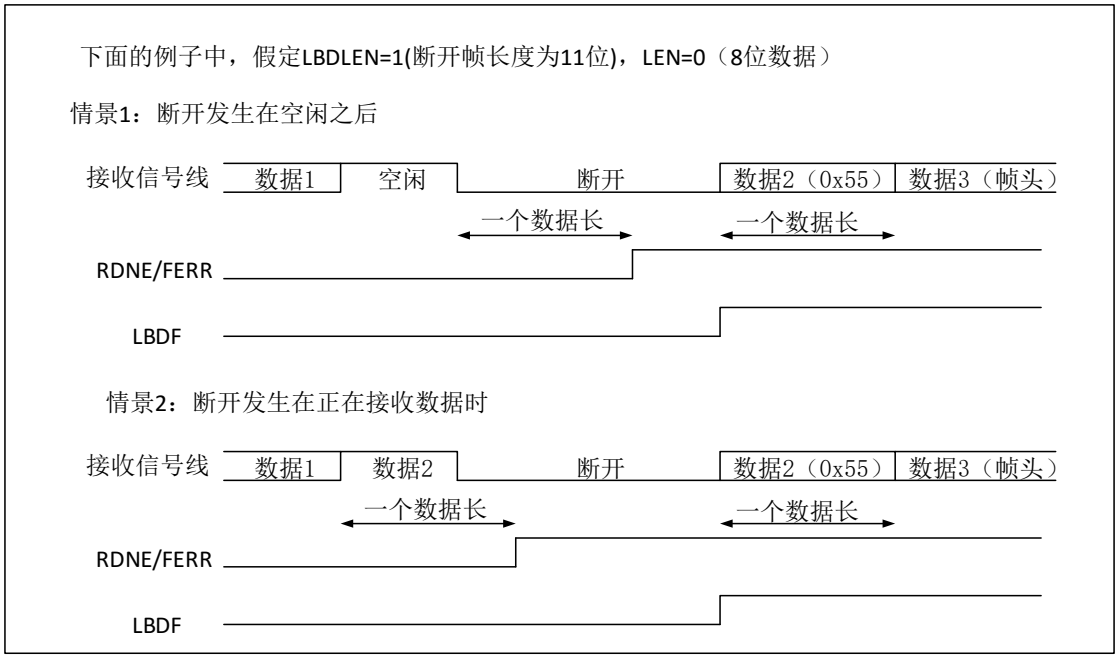


图 16-10 LIN模式下的断开检测与帧错误的检测



16.3.9 USART同步模式

通过在 USART_CTRL2 寄存器上写 CLKEN 位选择同步模式。在同步模式里，下列控制位必须保持清零状态：

- USART_CTRL2 寄存器中的 LINEN 位
- USART_CTRL3 寄存器中的 SCMEN,HALFSEL 和 IRDAEN 位

USART 允许用户以主模式方式控制双向同步串行通信。CK 脚是 USART 发送器时钟的输出。在起始位和停止位期间，CK 脚上没有时钟脉冲。根据 USART_CTRL2 寄存器中 LBCP 位的状态，决定在最后一个有效数据位期间产生或不产生时钟脉冲。USART_CTRL2 寄存器的 CLKPOL 位允许用户选择时钟极性，USART_CTRL2 寄存器上的 CLKPHA 位允许用户选择外部时钟的相位（见图 16-11、图 16-12 和图 16-13）。在总线空闲期间，实际数据到来之前以及发送断开帧的时候，外部 CK 时钟不被激活。同步模式时，USART 发送器和异步模式里工作一模一样。但是因为 CK 是与 TX 同步的（根据 CLKPOL 和 CLKPHA），所以 TX 上的数据是随 CK 同步发出的。

同步模式的 USART 接收器工作方式与异步模式不同。如果 REN=1，数据在 CK 上采样（根据 CLKPOL 和 CLKPHA 决定在上升沿还是下降沿），不需要任何的过采样。但必须考虑建立时间和持续时间（取决于波特率，1/16 位时间）。

注意：1.CK 脚同 TX 脚一起联合工作。因而，只有在使能了发送器（TEN＝1），并且发送数据时（写入数据至 USART_DT 寄存器）才提供时钟。这意味着在没有发送数据时是不可能接收一个同步数据的。

2.LBCP,CLKPOL 和 CLKPHA 位的配置，应该在发送器和接收器都被禁止时；当使能了发送器或接收器时，这些位不能被改变。

3.建议在同一条指令中设置 TEN 和 REN，以减少接收器的建立时间和保持时间。

4.USART 只支持主模式：它不能用来自其他设备的输入时钟接收或发送数据（CK 永远是输出）。

图 16-11 USART同步传输的例子

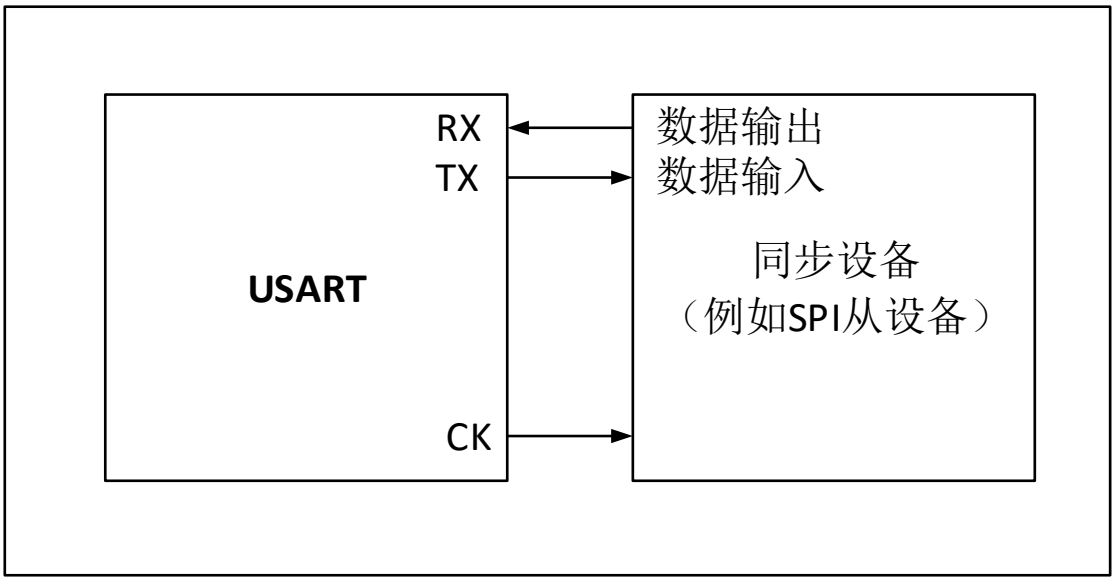


图 16-12 USART数据时钟时序示例（LEN=0）

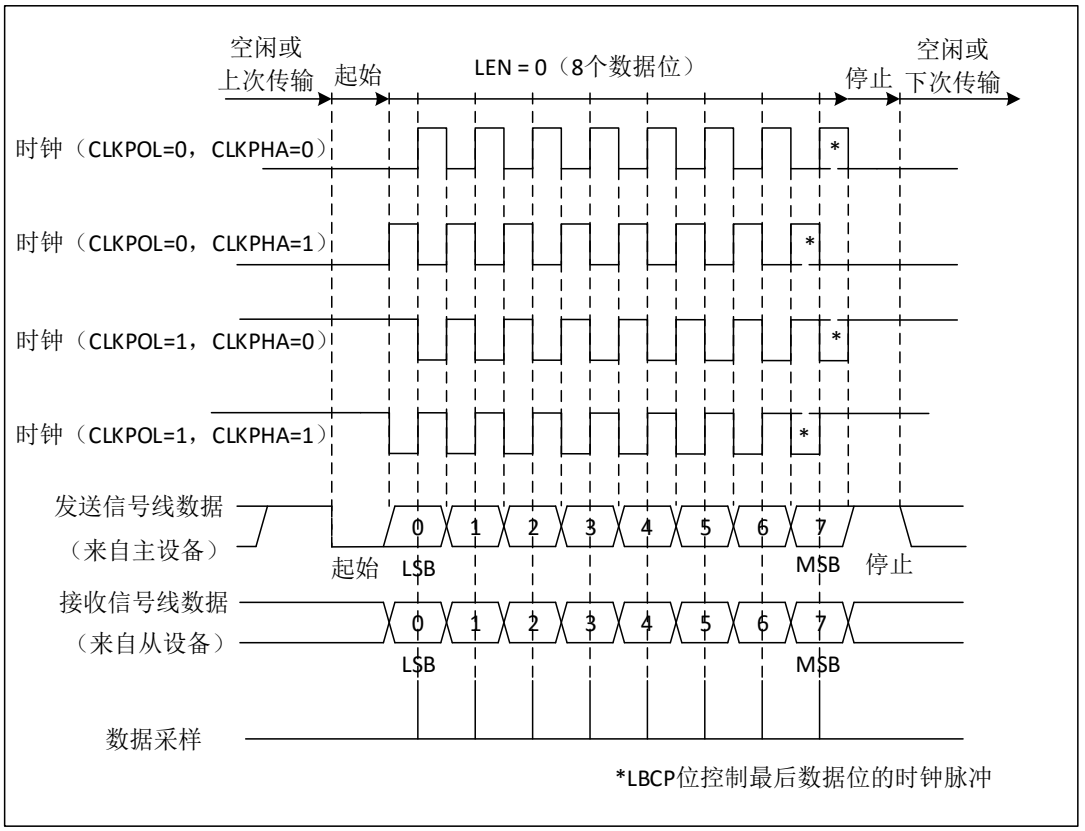


图 16-13 USART数据时钟时序示例（LEN=1）

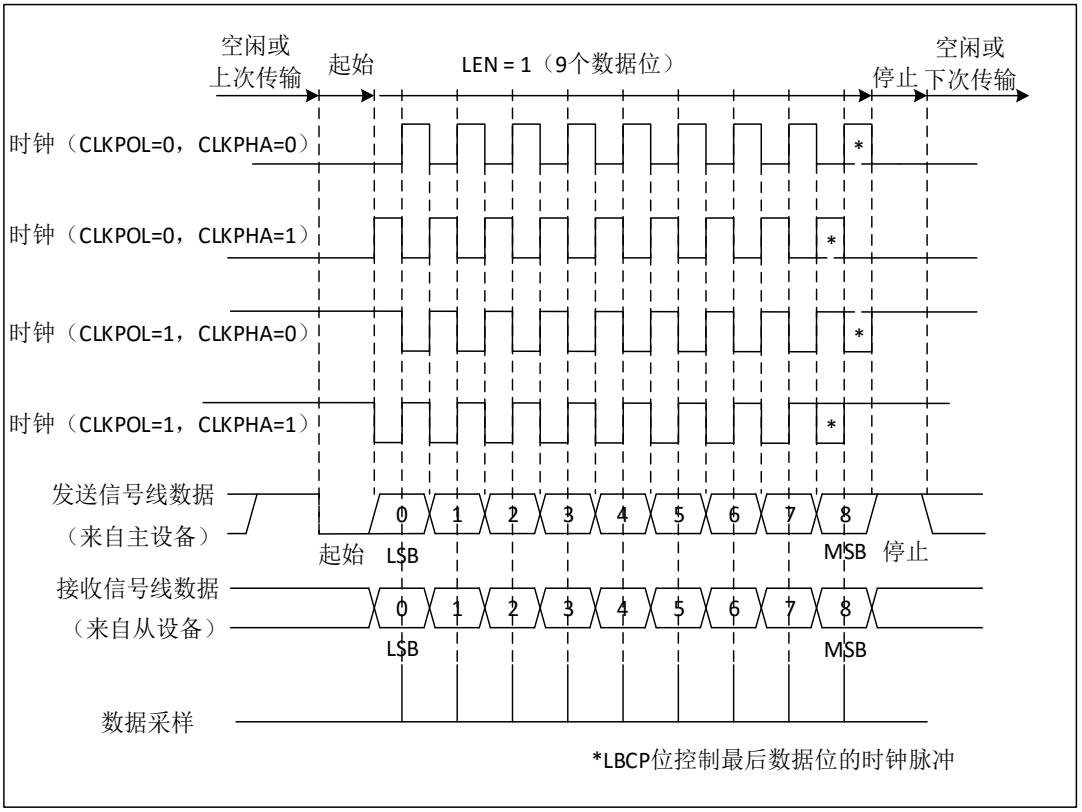
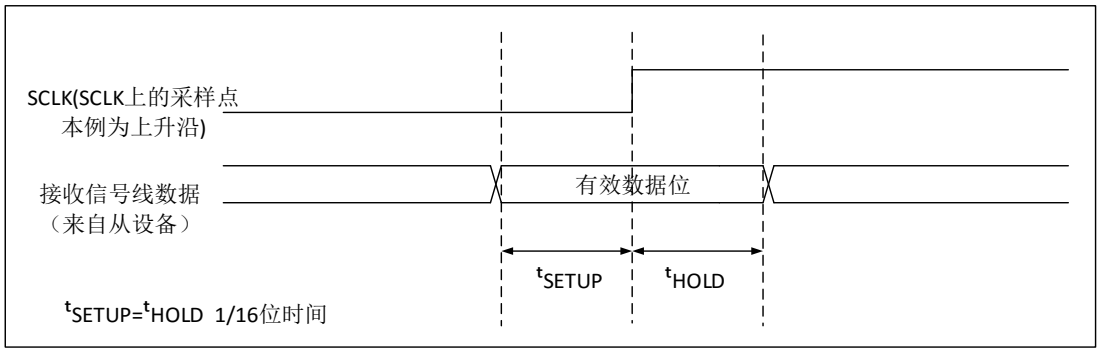


图 16-14 RX数据采样/保持时间



注意： 在智能卡模式下 CK 的功能不同，有关细节请参考智能卡模式部分。

16.3.10 单线半双工通信

单线半双方模式通过设置 USART_CTRL3 寄存器的 HALFSEL 位选择。在这个模式里，下面的位必须保持清零状态：

- USART_CTRL2 寄存器的 LINEN 和 CLKEN 位
- USART_CTRL3 寄存器的 SCMEN 和 IRDAEN 位

USART 可以配置成遵循单线半双工协议。在单线半双工模式下，TX 和 RX 引脚在芯片内部互连。使用控制位”HALFSEL”（USART_CTRL3 中的 HALFSEL 位）选择半双工和全双工通信。

当 HALFSEL 为’1’时

- RX 不再被使用。
- 当没有数据传输时，TX 总是被释放。因此，它在空闲状态的或接收状态时表现为一个标准 I/O 口。这就意味该 I/O 在不被 USART 驱动时，必须配置成悬空输入（或开漏的输出高）。除此以外，通信与普通 USART 模式类似。由软件来管理线上的冲突（例如通过使用一个中央仲裁器）。

特别的是，发送从不会被硬件所阻碍，也即如果 USART 处于发送状态时，USART 是无法接收数据的。换句话说，在半双工模式下，发送具有比接收有更高的优先权，发送和接收的冲突应该由软件处理。当 TEN 位被设置时，只要数据一写到数据寄存器上，发送就继续。

16.3.11 智能卡

设置 USART_CTRL3 寄存器的 SCMEN 位选择智能卡模式。在智能卡模式下，下列位必须保持清零：

- USART_CTRL2 寄存器的 LINEN 位
- USART_CTRL3 寄存器的 HALFSEL 位和 IRDAEN 位

此外，CLKEN 位可以被设置，以提供时钟给智能卡。

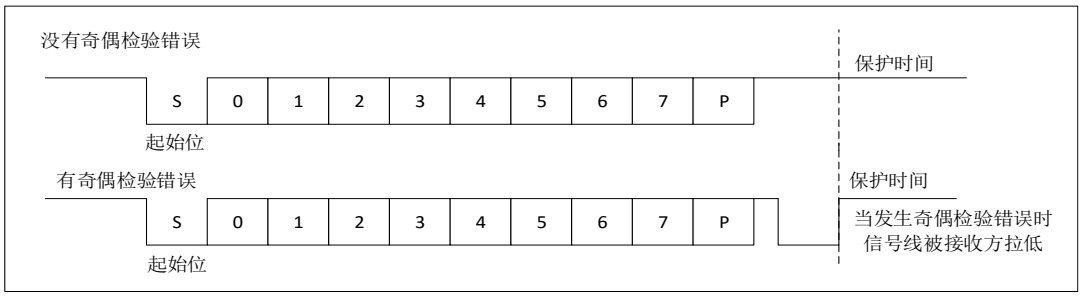
该接口符合 ISO7816-3 标准，支持智能卡异步协议。USART 应该被设置为：

- 8 位数据位加校验位：此时 USART_CTRL1 寄存器中 LEN=1、PCEN=1
- 发送和接收时为 1.5 个停止位：即 USART_CTRL2 寄存器的 STOPB=11

注意： 也可以在接收时选择 0.5 个停止位，但为了避免在 2 种配置间转换，建议在发送和接收时都使用 1.5 个停止位。

图 16-15 给出的例子说明了数据线上，在有校验错误和没校验错误两种情况下的信号。

图 16-15 ISO7816-3异步协议



当与智能卡相连接时，USART 的 TX 驱动一根智能卡也驱动的双向线；TX 需配置成开漏。

智能卡是一个单线半双工通信协议。

从发送移位寄存器把数据发送出去，要被延时最小 1/2 波特时钟。在普通操作时，一个满的发送移位寄存器将在下一个波特时钟沿开始向外移出数据。在智能卡模式下，此发送被延迟 1/2 波特时钟。

- 如果在接收一个设置为 0.5 或 1.5 个停止位的数据帧期间，检测到一奇偶校验错误，在完成接收该帧后（即停止位结束时），发送线被拉低一个波特时钟周期。这是告诉智能卡发送到 USART 的数据没有被正确地接收到。此 NACK 信号（拉低发送线一个波特时钟周期）在发送端将产生一个帧错误（发送端被配置成 1.5 个停止位）。应用程序可以根据协议处理重新发送数据。如果设置了 NACKEN 控制位，发生校验错误时接收器会给出一个 NACK 信号；否则就不会发送 NACK。
- 若发生奇偶校验错误，发送端的 FERR 在 1.5 个停止位结束时被置起。对于智能卡发送端而言，1.5 个的停止位可以被分成 2 部分：一个是 0.5 个数据位的周期，期间不做任何事情；随后是 1 个数据位的周期的停止位，在这段时间的中点处采样。对于智能卡接收端而言，1.5 个的停止位也可以被分成 2 部分：一个是 0.5 个数据位的周期，判断是否有奇偶校验错误；如果有奇偶校验错误，则在随后的 1 个数据位的周期拉低数据总线；如果没有奇偶校验错误，则在随后的 1 个数据位的周期，释放数据总线（数据总线处于高电平）。
- TRAC 标志的置起可以通过编程保护时间寄存器（GTVAL）得以延时。在普通操作时，当发送移位寄存器变空并且没有新的发送请求出现时，TRAC 被立即置起。在智能卡模式里，空的发送移位寄存器将触发保护时间计数器开始向上计数，直到保护时间寄存器（GTVAL）中的值。TRAC 在这段时间（也就是 GTVAL 个 bit 的时间宽度）被强制拉低。当保护时间计数器达到保护时间寄存器中的值时，TRAC 才能置起（也即置 1）。
- TRAC 标志的清零不受智能卡模式的影响。
- 如果发送器检测到一个帧错误（收到接收器的 NACK 信号），发送器的接收功能模块不会把

NACK 信号当作起始位检测。根据 ISO 协议，接收到的 NACK 信号的持续时间可以是 1 或 2 波特时钟周期。

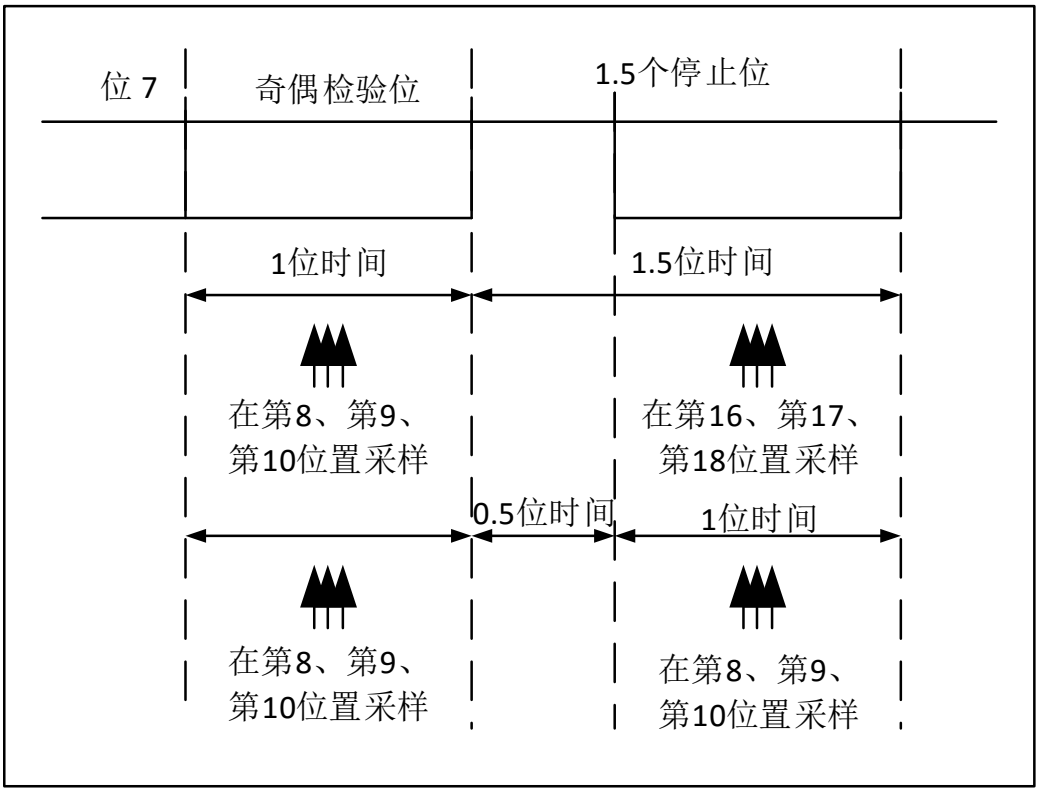
- 在接收器这边，如果一个校验错误被检测到，并且 NACK 被发送，接收器不会把 NACK 检测成起始位。

注意： 1. 断开帧在智能卡模式里没有意义。一个带帧错误的 00h 数据将被当成数据而不是断开帧。

2. 当来回切换 TEN 位时，没有 IDLE 帧被发送。ISO 协议没有定义 IDLE 帧。

下图详述了 USART 是如何采样 NACK 信号的。在这个例子里，USART 正在发送数据，并且被配置成 1.5 个停止位。为了检查数据的完整性和 NACK 信号，USART 的接收功能块被激活。

图 16-16 使用1.5停止位检测奇偶检验错



USART 可以通过 CK 输出为智能卡提供时钟。在智能卡模式里，CK 不和通信直接关联，而是先通过一个 5 位预分频器简单地用内部的外设输入时钟来驱动智能卡的时钟。分频率在预分频寄存器 GTVAL 中配置。CK 频率可以从f_{CK}/2 到f_{CK}/62，这里的f_{CK}是外设输入时钟。

16.3.12 IrDA SIR ENDEC功能模块

通过设置 USART_CTRL3 寄存器的 IRDAEN 位选择 IrDA 模式。在 IrDA 模式里，下列位必须保持清零：

- USART_CTRL2 寄存器的 LINEN,STOPB 和 CLKEN 位
- USART_CTRL3 寄存器的 SCMEN 和 HALFSEL 位。

IrDA SIR 物理层规定使用反相归零调制方案（RZI），该方案用一个红外光脉冲代表逻辑’0’（见图 16-17）。SIR 发送编码器对从 USART 输出的 NRZ（非归零）比特流进行调制。输出脉冲流被传送到一个外部输出驱动器和红外 LED。USART 为 SIRENDEC 最高只支持到 115.2Kbps 速率。在普通模式里，脉冲宽度规定为一个位周期的 3/16。

SIR 接收解码器对来自红外接收器的归零位比特流进行解调，并将接收到的 NRZ 串行比特流输出到 USART。在空闲状态里，解码器输入通常是高（标记状态 marking state）。发送编码器输出的极性和解码器的输入相反。当解码器输入低时，检测到一个起始位。

- IrDA 是一个半双工通信协议。如果发送器忙（也就是 USART 正在送数据给 IrDA 编码器），

IrDA 接收线上的任何数据将被 IrDA 解码器忽视。如果接收器忙（也就是 USART 正在接收从 IrDA 解码器来的解码数据），从 USART 到 IrDA 的 TX 上的数据将不会被 IrDA 编码。当接收数据时，应该避免发送，因为将被发送的数据可能被破坏。

- SIR 发送逻辑把’0’作为高脉冲发送，把’1’作为低电平发送。脉冲的宽度规定为普通模式时位周期的 3/16（见图 16-18）。
- SIR 接收逻辑把高电平状态解释为’1’，把低脉冲解释为’0’。
- 发送编码器输出与解码器输入有着相反的极性。当空闲时，SIR 输出处于低状态。
- SIR 解码器把 IrDA 兼容的接收信号转变成给 USART 的比特流。
- IrDA 规范要求脉冲要宽于 1.41us。脉冲宽度是可编程的。接收器端的尖峰脉冲检测逻辑滤除宽度小于 2 个 DIV 周期的脉冲（DIV 是在 IrDA 低功耗波特率寄存器 GTP 中编程的预分频值）。宽度小于 1 个 DIV 周期的脉冲一定被滤除掉，但是那些宽度大于 1 个而小于 2 个 DIV 周期的脉冲可能被接收或滤除，那些宽度大于 2 个周期的将被视为一个有效的脉冲。当 DIV=0 时，IrDA 编码器/解码器不工作。
- 接收器可以与一低功耗发送器通信。
- 在 IrDA 模式里，USART_CTRL2 寄存器上的 STOPB 位必须配置成 1 个停止位。

IrDA 低功耗模式

发送器在低功耗模式，脉冲宽度不再持续 3/16 个位周期。取而代之，脉冲的宽度是低功耗波特率的 3 倍，它最小可以是 1.42MHz。通常这个值是 1.8432MHz（1.42MHz<DIV<2.12MHz）。一个低功耗模式可编程分频器把系统时钟进行分频以达到这个值。低功耗模式的接收器类似于普通模式的接收器。为了滤除尖峰干扰脉冲，USART 应该滤除宽度短于 1 个 DIV 的脉冲。只有持续时间大于 2 个周期的 IrDA 低功耗波特率时钟（GTP 中的 DIV）的低电平信号才被接受为有效的信号。

注意：1.宽度小于 2 个，但大于 1 个 DIV 周期的脉冲可能会也可能不会被滤除。

2.接收器的建立时间应该由软件管理。IrDA 物理层技术规范规定了在发送和接收之间最小要有 10ms 的延时（IrDA 是一个半双工协议）。

图 16-17 IrDA SIR ENDEC框图

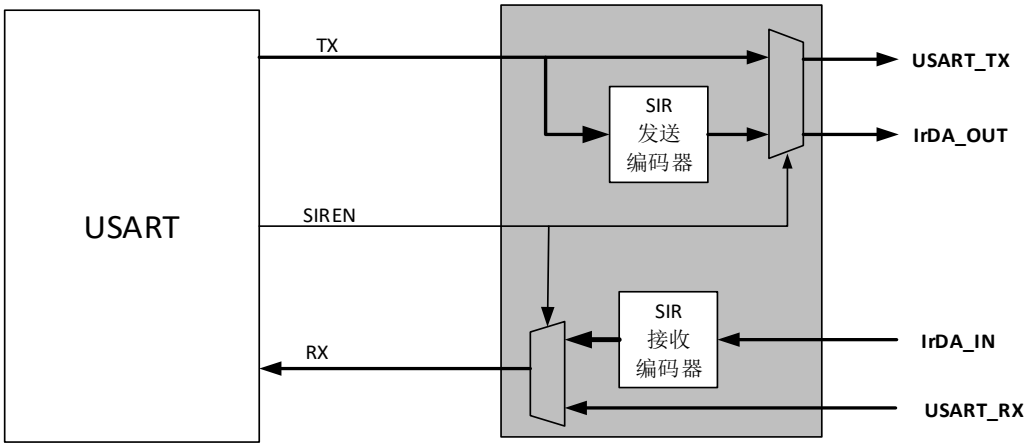
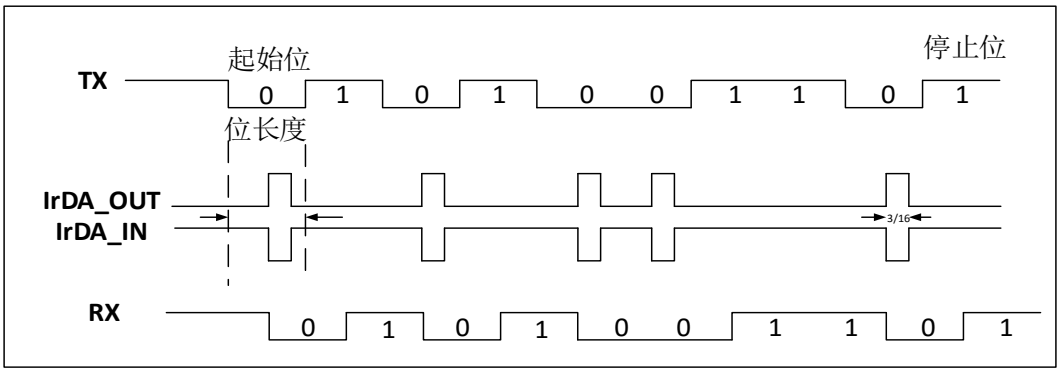


图 16-18 IrDA数据调制（3/16）－普通模式



16.3.13 利用DMA连续通信

USART 可以利用 DMA 连续通信。RX 缓冲器和 TX 缓冲器的 DMA 请求是分别产生的。

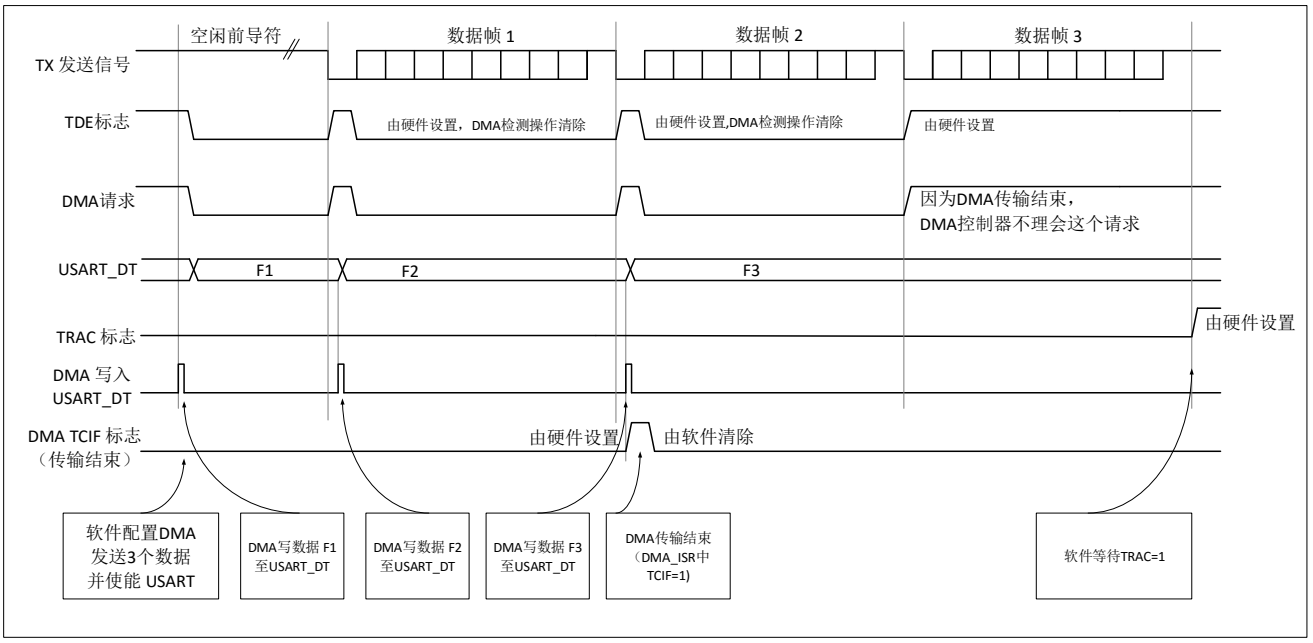
注意：参考产品技术说明以确定是否可用 DMA 控制器。如果所用产品无 DMA 功能，应按 16.3.2 节或 16.3.3 节里所描述的方法使用 USART。在 USART_STS 寄存器里，可以清零 TDE/RDNE 标志来实现连续通信。

16.3.13.1 利用DMA发送

使用 DMA 进行发送，可以通过设置 USART_CTRL3 寄存器上的 DMATEN 位激活。当 TDE 位被置为’1’时，DMA 就从指定的 SRAM 区传送数据到 USART_DT 寄存器。为 USART 的发送分配一个 DMA 通道的步骤如下（x 表示通道号）：

1. 在 DMA 控制寄存器上将 USART_DT 寄存器地址配置成 DMA 传输的目的地址。在每个 TDE 事件后，数据将被传送到这个地址。
2. 在 DMA 控制寄存器上将存储器地址配置成 DMA 传输的源地址。在每个 TDE 事件后，将从此存储器区读出数据并传送到 USART_DT 寄存器。
3. 在 DMA 控制寄存器中配置要传输的总的字节数。
4. 在 DMA 寄存器上配置通道优先级。
5. 根据应用程序的要求，配置在传输完成一半还是全部完成时产生 DMA 中断。
6. 在 DMA 寄存器上激活该通道。当传输完成 DMA 控制器指定的数据宽度时，DMA 控制器在该 DMA 通道的中断向量上产生一中断。在发送模式下，当 DMA 传输完所有要发送的数据时，DMA 控制器设置 DMA_ISTS 寄存器的 TCIF 标志；监视 USART_STS 寄存器的 TRAC 标志可以确认 USART 通信是否结束，这样可以在关闭 USART 或进入停机模式之前避免破坏最后一次传输的数据；软件需要先等待 TDE=1，再等待 TRAC=1。

图 16-19 利用DMA发送

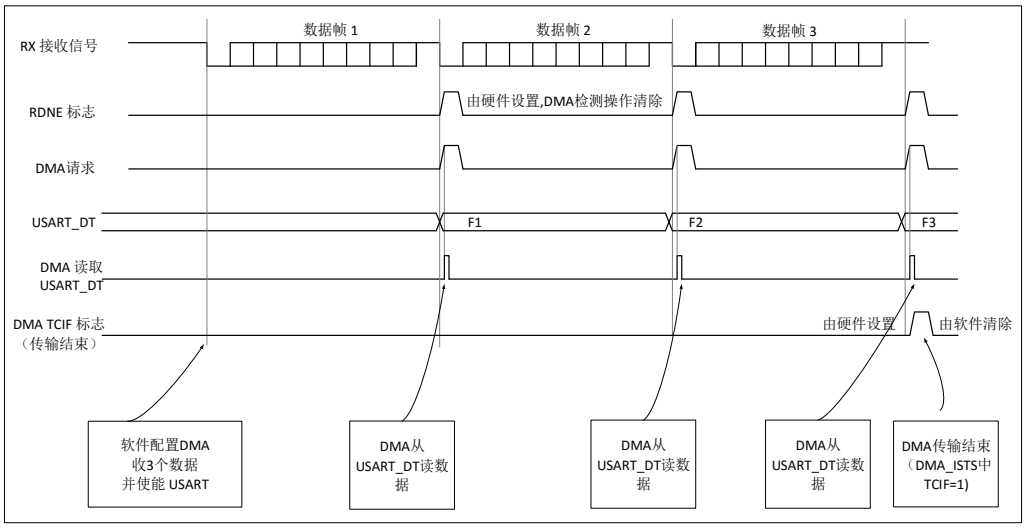


16.3.13.2 利用DMA接收

可以通过设置 USART_CTRL3 寄存器的 DMAREN 位激活使用 DMA 进行接收，每次接收到一个字节，DMA 控制器就就把数据从 USART_DT 寄存器传送到指定的 SRAM 区(参考 DMA 相关说明)。为 USART 的接收分配一个 DMA 通道的步骤如下（x 表示通道号）：

1. 通过 DMA 控制寄存器把 USART_DT 寄存器地址配置成传输的源地址。在每个 RDNE 事件后，将从此地址读出数据并传输到存储器。
2. 通过 DMA 控制寄存器把存储器地址配置成传输的目的地址。在每个 RDNE 事件后，数据将从 USART_DT 传输到此存储器区。
3. 在 DMA 控制寄存器中配置要传输的总的字节数。
4. 在 DMA 寄存器上配置通道优先级。。
5. 根据应用程序的要求配置在传输完成一半还是全部完成时产生 DMA 中断。
6. 在 DMA 控制寄存器上激活该通道。当接收完成 DMA 控制器指定的传输量时，DMA 控制器在该 DMA 通道的中断向量上产生一中断。

图 16-20 利用DMA接收



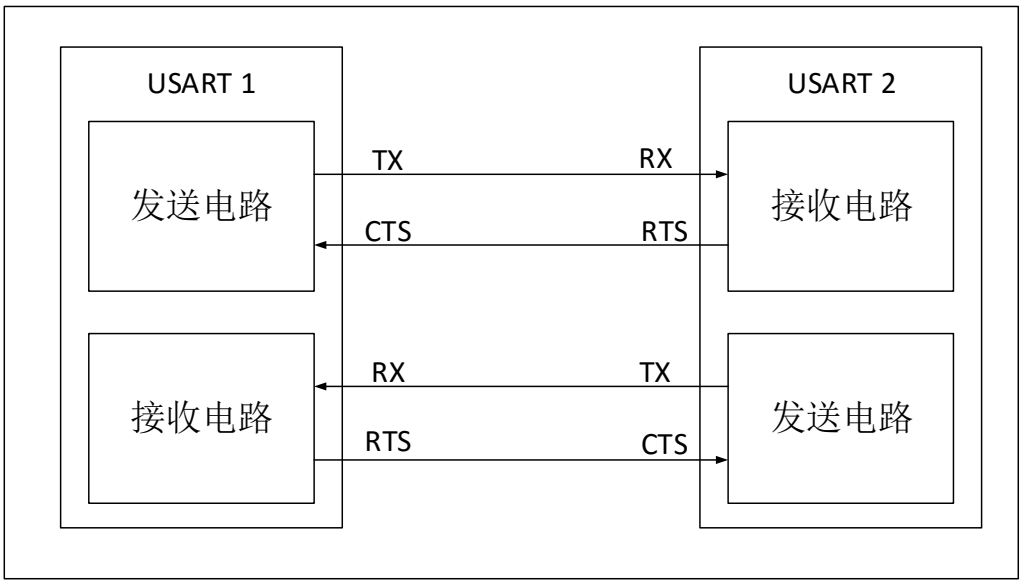
16.3.13.3 多缓冲器通信中的错误标志和中断产生

在多缓冲器通信的情况下，通信期间如果发生任何错误，在当前字节传输后将置起错误标志。如果中断使能位被设置，将产生中断。在单个字节接收的情况下，和 RDNE 一起被置起的帧错误、溢出错误和噪音标志，有单独的错误标志中断使能位；如果设置了，会在当前字节传输结束后，产生中断。

16.3.14 硬件流控制

利用 CTS 输入和 RTS 输出可以控制 2 个设备间的串行数据流。下图表明在这个模式里如何连接 2 个设备。

图 16-21 两个USART间的硬件流控制

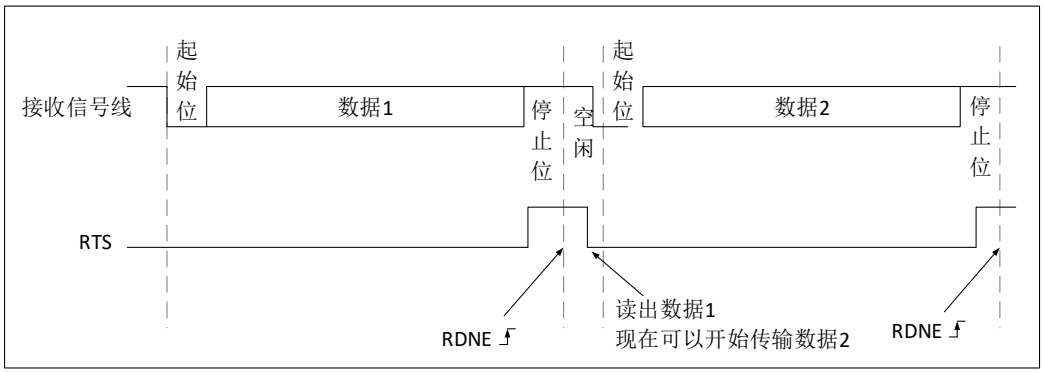


通过将 UASRT_CTRL3 中的 RTSEN 和 CTSEN 置位，可以分别独立地使能 RTS 和 CTS 流控制。

16.3.14.1 RTS流控制

如果 RTS 流控制被使能（RTSEN=1），只要 USART 接收器准备好接收新的数据，RTS 就变成有效（下拉为低电平）。当接收寄存器内有数据到达时（在每个 stop 位开始时），RTS 被置位，由此表明希望在当前帧结束时停止数据传输；当 RDNE 置起后，读 DR 会使得 RDNE 被清零，从而 RTS 也被清零，表示接收机可以接收下一个数据。下图是一个启用 RTS 流控制的通信的例子。

图 16-22 RTS流控制

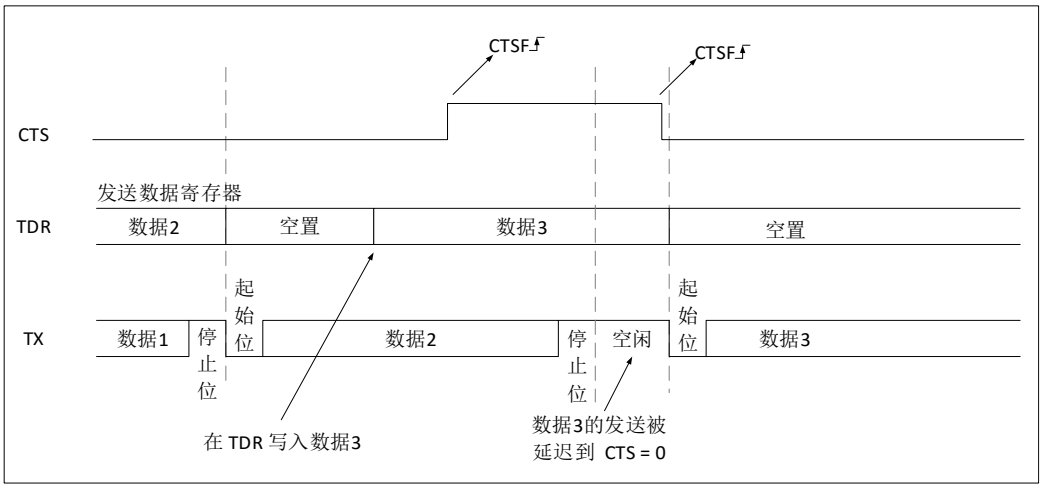


16.3.14.2 CTS流控制

如果 CTS 流控制被使能（CTSEN=1），发送器在发送下一帧前检查 CTS 输入。如果 CTS 有效（也即 CTS 为低电平），则下一个数据被发送（假设那个数据是准备发送的，也就是 TDE=0）；若 CTS 在传输期间被变成无效(也即 CTS 为高电平)，当前的传输完成后停止发送。

当 CTSEN=1 时，只要 CTS 电平发生变化，硬件就自动设置 CTSF 状态位。它表明接收器是否准备好进行通信。如果设置了 USART_CT3 寄存器的 CTSIEN 位，则产生中断。下图是一个启用 CTS 流控制通信的例子。

图 16-23 CTS流控制



16.4 USART中断请求

表 16-6 USART中断请求

中断事件	事件标志	使能位
发送数据寄存器空	TDE	TDEIEN
CTS 标志	CTSF	CTSIEN
发送完成	TRAC	TRACIEN
接收数据就绪可读	RDNE	RDNEIEN
检测到数据溢出	ORERR	
检测到空闲线路	IDLEF	IDLEIEN
奇偶检验错	PERR	PERRIEN
断开标志	LBDF	LBDIEN
噪声标志，多缓冲通信中的溢出错误和帧错误	NERR 或 ORERR 或 FERR	ERRIEN (1)

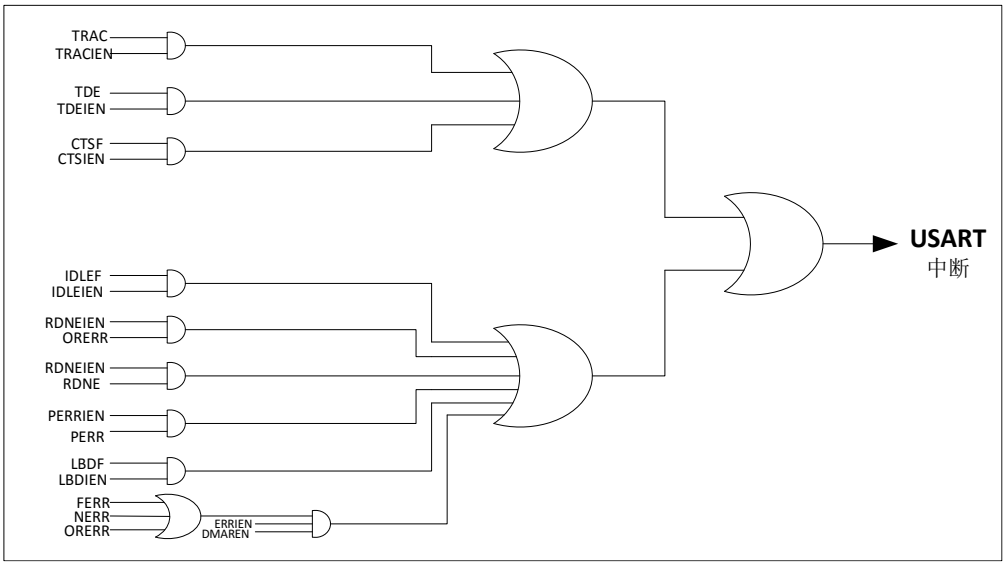
注意： 仅当使用 DMA 接收数据时，才使用这个标志位。

USART 的各种中断事件被连接到同一个中断向量（见下图），有以下各种中断事件：

- 发送期间：发送完成、CTS 变化、发送数据寄存器空。
- 接收期间：空闲总线检测、溢出错误、接收数据寄存器非空、校验错误、LIN 断开帧检测、噪音标志（仅在多缓冲器通信）和帧错误（仅在多缓冲器通信）。

如果设置了对应的使能控制位，这些事件就可以产生各自的中断。

图 16-24 USART中断映像图



16.5 USART模式配置

表 16-7 USART模式设置（1）

USART 模式	USART1	USART2	USART3	UART4	UART5	USART6	UART7	UART8
异步模式	X	X	X	X	X	X	X	X
硬件流控制	X	X	X	NA	NA	NA	NA	NA
多缓存通讯（DMA）	X	X	X	X	X	X	X	X
多处理器通讯	X	X	X	X	X	X	X	X
同步	X	X	X	NA	NA	X	NA	NA
智能卡	X	X	X	NA	NA	X	NA	NA
半双工（单线模式）	X	X	X	X	X	X	X	X
IrDA	X	X	X	X	X	X	X	X
LIN	X	X	X	X	X	X	X	X

注 1：X=支持，NA=不支持该应用。

16.6 USART寄存器描述

有关寄存器描述里所使用的缩写，请参考“寄存器描述表中使用的缩写列表”。

可以用半字（16 位）或字（32 位）的方式操作这些外设寄存器。

16.6.1 USART寄存器地址映象

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

0x00	USART_STS	保留																		CTSF	LBDF	TDE	TRAC	RDNE	IDLEF	ORERR	NERR	FERR	PERR
	0x00C0																			0	0	1	1	0	0	0	0	0	0
0x04	USART_DT	保留																		DT[8：0]									
	0x0000																			0	0	0	0	0	0	0	0	0	0
0x08	USART_BAUDR	保留														DIV_Integer[13：0]												DIV_Decimal [3：0]			
	0x0000															0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	USART_CTRL1	保留																UEN	LEN	WUMODE	PCEN	PSEL	PERRIEN	TDEIEN	TRACIEN	RDNEIEN	IDLEIEN	TEN	REN	RECMUTE	SBRK
	0x0000																	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	USART_CTRL2	保留															LINEN	STOPB [1：0]		CLKEN	CLKPOL	CLKPHA	LBCP	保留	LBDIEN	LBDLEN	保留	ADDR[3：0]			
	0x0000																0	0	0	0	0	0	0		0	0		0	0	0	0
0x14	USART_CTRL3	保留																			CTSIEN	CTSEN	RTSEN	DMATEN	DMAREN	SCMEN	NACKEN	HALFSEL	IRDALP	IRDAEN	ERRIEN
	0x0000																				0	0	0	0	0	0	0	0	0	0	0
0x18	USART_GTP	保留																		GTVAL[7：0]								DIV[7：0]			
	0x0000																			0	0	0	0	0	0	0	0	0	0	0	0

16.6.2 状态寄存器（USART_STS）

地址偏移：0x00

复位值：0x00C0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															
res															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留						CTSF	LBDF	TDE	TRAC	RDNE	IDLEF	ORERR	NERR	FERR	PERR
res						rc_w0	rc_w0	r	rc_w0	rc_w0	r	r	r	r	r

位 31：10	保留位，硬件强制为 0
位 9	**CTSF**：CTSF 标志（CTS flag） 如果设置了 CTSEN 位，当 CTS 输入变化状态时，该位被硬件置高。由软件将其清零。如果 USART_CTRL3 中的 CTSIEN 为'1'，则产生中断。 0：CTS 状态线上没有变化； 1：CTS 状态线上发生变化。 注意：UART4、UART5、USART6、UART7 和 UART8 上不存在这一位。
位 8	**LBDF**：LIN 断开检测标志（LIN break detection flag） 当探测到 LIN 断开时，该位由硬件置'1'，由软件清'0'（向该位写 0）。如果 USART_CTRL3 中的 LBDIEN=1，则产生中断。 0：没有检测到 LIN 断开； 1：检测到 LIN 断开。 注意：若 LBDIEN=1，当 LBDF 为'1'时要产生中断。
位 7	**TDE**：发送数据寄存器空（Transmit data register empty） 当 TDR 寄存器中的数据被硬件转移到移位寄存器的时候，该位被硬件置位。如果 USART_CTRL1 寄存器中的 TDEIEN 为 1，则产生中断。对 USART_DT 的写操作，将清零该位。 0：数据还没有被转移到移位寄存器； 1：数据已经被转移到移位寄存器。 注意：单缓冲器传输中使用该位。
位 6	**TRAC**：发送完成（Transmission complete） 当包含有数据的一帧发送完成后，并且 TDE=1 时，由硬件将该位置'1'。如果 USART_CTRL1 中的 TRACIEN 为'1'，则产生中断。由软件序列清除该位（先读 USART_STS，然后写入 USART_DT）。TRAC 位也可以通过写入'0'来清除，只有在多缓存通讯中才推荐这种清除程序。 0：发送还未完成； 1：发送完成。
位 5	**RDNE**：读数据寄存器非空（Read data register not empty） 当 RDR 移位寄存器中的数据被转移到 USART_DT 寄存器中，该位被硬件置位。如果 USART_CTRL1 寄存器中的 RDNEIEN 为 1，则产生中断。对 USART_DT 的读操作可以将该位清零。RDNE 位也可以通过写入 0 来清除，只有在多缓存通讯中才推荐这种清除程序。 0：数据没有收到； 1：收到数据，可以读出。

位 4	**IDLEF**：监测到总线空闲（IDLE line detected） 当检测到总线空闲（空闲帧）时，该位被硬件置位。如果 USART_CTRL1 中的 IDLEIEN 为'1'，则产生中断。由软件序列清除该位（先读 USART_STS，然后读 USART_DT）。 0：没有检测到空闲总线，也即没有检测到空闲帧； 1：检测到空闲总线。 *注意：IDLEF 位不会再次被置高直到 RDNE 位被置起（即又检测到一次空闲总线）*
位 3	**ORERR**：过载错误（Overrun error） 当 RDNE 仍然是'1'的时候，当前被接收在移位寄存器中的数据，需要传送至 RDR 寄存器时，硬件将该位置位。如果 USART_CTRL1 中的 RDNEIEN 为'1'的话，则产生中断。由软件序列将其清零（先读 USART_STS，然后读 USART_DT）。 0：没有过载错误； 1：检测到过载错误。 注意：该位被置位时，RDR 寄存器中的值不会丢失，但是移位寄存器中的数据会被覆盖。如果设置了 ERRIEN 位，在多缓冲器通信模式下，ORERR 标志置位会产生中断的。
位 2	**NERR**：噪声错误标志（Noise error flag） 在接收到的帧检测到噪音时，由硬件对该位置位。由软件序列对其清零（先读 USART_STS，再读 USART_DT）。 0：没有检测到噪声； 1：检测到噪声。 注意：该位不会产生中断，因为它和 RDNE 一起出现，硬件会在设置 RDNE 标志时产生中断。在多缓冲区通信模式下，如果设置了 ERRIEN 位，则设置 NERR 标志时会产生中断。
位 1	**FERR**：帧错误（Framing error） 当检测到同步错位，过多的噪声或者检测到断开符，该位被硬件置位。由软件序列将其清零（先读 USART_STS，再读 USART_DT）。 0：没有检测到帧错误； 1：检测到帧错误或者断开帧（break frame）。 注意：该位不会产生中断，因为它和 RDNE 一起出现，硬件会在设置 RDNE 标志时产生中断。如果当前传输的数据既产生了帧错误，又产生了过载错误，硬件还是会继续该数据的传输，并且只设置 ORERR 标志位。 在多缓冲区通信模式下，如果设置了 ERRIEN 位，则设置 FERR 标志时会产生中断。
位 0	**PERR**：校验错误（Parity error） 在接收模式下，如果出现奇偶校验错误，硬件对该位置位。由软件序列对其清零（依次读 USART_STS 和 USART_DT）。在清除 PERR 位前，软件必须等待 RDNE 标志位被置'1'。如果 USART_CTRL1 中的 PERRIEN 为'1'，则产生中断。 0：没有奇偶校验错误； 1：奇偶校验错误。

16.6.3 数据寄存器（USART_DT）

地址偏移：0x04

复位值：0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															
res															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留							DT[8：0]								
res							rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31：9	保留位，硬件强制为 0

位 8：0	**DT[8：0]**：数据值（Data value） 包含了发送或接收的数据。由于它是由两个寄存器组成的，一个给发送用（TDR），一个给接收用（RDR），该寄存器兼具读和写的功能。TDR 寄存器提供了内部总线和输出移位寄存器之间的并行接口（参见图 16-1）。RDR 寄存器提供了输入移位寄存器和内部总线之间的并行接口。 当使能校验位（USART_CTRL1 中 PCEN 位被置位）进行发送时，写到 MSB 的值（根据数据的长度不同，MSB 是第 7 位或者第 8 位）会被校验位取代。当使能校验位进行接收时，读到的 MSB 位是接收到的校验位。

16.6.4 波特比率寄存器（USART_BAUDR）

地址偏移：0x08

复位值：0x0000

注意：如果 TE 或 RE 被分别禁止，波特计数器停止计数。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															
res															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DIV_Integer[11：0]												DIV_Decimal[3：0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31：18	保留位，硬件强制为 0
位 15：4	**DIV_ Integer[11：0]**：USARTDIV 的整数部分 这 12 位定义了 USART 分频器除法因子（USARTDIV）的整数部分。
位 3：0	**DIV_Decimal[3：0]**：USARTDIV 的小数部分 这 4 位定义了 USART 分频器除法因子（USARTDIV）的小数部分。

16.6.5 控制寄存器1（USART_CTRL1）

地址偏移：0x0C

复位值：0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															
res															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留		UEN	LEN	WUM ODE	PCEN	PSEL	PER RIEN	TDE IEN	TRA CIEN	RDN EIEN	IDLEI EN	TEN	REN	REC MUTE	SBRK
res		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31：14	保留位，硬件强制为 0。
位 13	**UEN**：USART 使能（USART enable） 当该位被清零，在当前字节传输完成后 USART 的分频器和输出停止工作，以减少功耗。该位由软件设置和清零。 0：USART 分频器、输出和输入均被被禁止； 1：USART 模块使能。

位 12	**LEN**：字长（Word length） 该位定义了数据字的长度，由软件对其设置和清零 0：一个起始位，8 个数据位，n 个停止位； 1：一个起始位，9 个数据位，n 个停止位。 *注意：在数据传输过程中（发送或者接收时），不能修改这个位。*
位 11	**WUMODE**：唤醒方法（Wake up mode） 这位决定了把 USART 唤醒的方法，由软件对该位设置和清零。 0：被空闲帧唤醒； 1：被地址标记唤醒。
位 10	**PCEN**：检验控制使能（Parity control enable） 用该位选择是否进行硬件校验控制（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，在发送数据的最高位（如果 LEN=1，最高位就是第 9 位；如果 LEN=0，最高位就是第 8 位）插入校验位；对接收到的数据检查其校验位。软件对它置'1'或清'0'。一旦设置了该位，当前字节传输完成后，校验控制才生效。 0：禁止校验控制； 1：使能校验控制。
位 9	**PSEL**：校验选择（Parity selection） 当校验控制使能后，该位用来选择是采用偶校验还是奇校验。软件对它置'1'或清'0'。当前字节传输完成后，该选择生效。 0：偶校验； 1：奇校验。
位 8	**PERRIEN**：PERR 中断使能（PERR interrupt enable） 该位由软件设置或清除。 0：禁止产生中断； 1：当 USART_STS 中的 PERR 为'1'时，产生 USART 中断。
位 7	**TDEIEN**：发送缓冲区空中断使能（TDE interrupt enable） 该位由软件设置或清除。 0：禁止产生中断； 1：当 USART_STS 中的 TDE 为'1'时，产生 USART 中断。
位 6	**TRACIEN**：发送完成中断使能（TRAC interrupt enable） 该位由软件设置或清除。 0：禁止产生中断； 1：当 USART_STS 中的 TRAC 为'1'时，产生 USART 中断。
位 5	**RDNEIEN**：接收缓冲区非空中断使能 （RDNE interrupt enable） 该位由软件设置或清除。 0：禁止产生中断； 1：当 USART_SR 中的 RDNE 或者 ORERR 为'1'时，产生 USART 中断。
位 4	**IDLEIEN**：IDLE 中断使能 （IDLE interrupt enable） 该位由软件设置或清除。 0：禁止产生中断； 1：当 USART_STS 中的 IDLEF 为'1'时，产生 USART 中断。
位 3	**TEN**：发送使能 （Transmitter enable） 该位使能发送器。该位由软件设置或清除。 0：禁止发送； 1：使能发送。 *注意：1. 在数据传输过程中，除了在智能卡模式下，如果 TEN 位上有个 0 脉冲（即设置为'0'之后 再设置为'1'），会在当前数据字传输完成后，发送一个"前导符"（空闲总线，也即空闲帧）。* *2. 当 TEN 被设置后，在真正发送开始之前，有一个比特时间的延迟。*

位 2	**REN**：接收使能 （Receiver enable） 该位由软件设置或清除。 0：禁止接收； 1：使能接收，并开始搜寻 RX 引脚上的起始位。
位 1	**RECMUTE**：接收唤醒 （Receiver wakeup） 该位用来决定是否把 USART 置于静默模式。该位由软件设置或清除。当唤醒序列到来时，硬件也会将其清零。 0：接收器处于普通工作模式； 1：接收器处于静默模式。 *注意：1. 在把 USART 置于静默模式（设置 RECMUTE 位）之前，USART 要先接收一个数据。否则在静默模式下，不能被空闲总线检测唤醒。* *2. 当配置成地址标记检测唤醒（WUMODE 位=1），在 RDNE 位被置位时，不能用软件修改 RECMUTE 位。*
位 0	**SBRK**：发送断开帧 （Send break） 使用该位来发送断开字符。该位可以由软件设置或清除。操作过程应该是软件设置该位，然后在断开帧的停止位时，由硬件将该位复位。 0：没有发送断开字符； 1：将要发送断开字符。

16.6.6 控制寄存器2（USART_CTRL2）

地址偏移：0x10

复位值：0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															
res															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留	LIN EN	STOP B[1：0]		CLK EN	CLK POL	CLKP HA	LBCP	保留	LBD IEN	LBD LEN	保留	ADDR[3：0]			
res	rw	rw	rw	rw	rw	rw	rw	res	rw	rw	res	rw	rw	rw	rw

位 31：15	保留位，硬件强制为 0。
位 14	**LINEN**：LIN 模式使能（LIN mode enable） 该位由软件设置或清除。 0：禁止 LIN 模式； 1：使能 LIN 模式。在 LIN 模式下，可以用 USART_CTRL1 寄存器中的 SBRK 位发送 LIN 同步断开符（也即 13 位低电平），以及检测 LIN 同步断开符。
位 13：12	**STOPB**：停止位（STOP bits） 这 2 位用来设置停止位的位数 00：1 个停止位； 01：0.5 个停止位； 10：2 个停止位； 11：1.5 个停止位； 注： UART4、UART5、UART7 和 UART8 不能用 0.5 停止位和 1.5 停止位。
位 11	**CLKEN**：时钟使能（Clock enable） 该位用来使能 CK 引脚 0：禁止 CK 引脚； 1：使能 CK 引脚。 *注意：UART4、UART5、UART7 和 UART8 上不存在这一位。*

位 10	**CLKPOL**：时钟极性（Clock polarity） 在同步模式下，可以用该位选择 SLCK 引脚上时钟输出的极性。和 CLKPHA 位一起配合来产生需要的时钟/数据的采样关系 0：总线空闲时 CK 引脚上保持低电平； 1：总线空闲时 CK 引脚上保持高电平。 注：UART4、UART5、UART7 和 UART8 上不存在这一位。
位 9	**CLKPHA**：时钟相位（Clock phase） 在同步模式下，可以用该位选择 SLCK 引脚上时钟输出的相位。和 CLKPOL 位一起配合来产生需要的时钟/数据的采样关系（参见图 16-12 和图 16-13）。 0：在时钟的第一个边沿进行数据捕获； 1：在时钟的第二个边沿进行数据捕获。 注：UART4、UART5、UART7 和 UART8 上不存在这一位。
位 8	**LBCP**：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 CK 引脚上输出最后发送的那个数据字节（MSB）对应的时钟脉冲 0：最后一位数据的时钟脉冲不从 CK 输出； 1：最后一位数据的时钟脉冲会从 CK 输出。 *注意：1. 最后一个数据位就是第 8 或者第 9 个发送的位（根据 USART_CTRL1 寄存器中的 LEN 位所定义的 8 或者 9 位数据帧格式）。2.* UART4、UART5、UART7 和 UART8 *上不存在这一位。*
位 7	保留位，硬件强制为 0
位 6	**LBDIEN**：LIN 断开符检测中断使能（LIN break detection interrupt enable） 断开符中断屏蔽（使用断开分隔符来检测断开符） 0：禁止中断； 1：只要 USART_STS 寄存器中的 LBDF 为'1'就产生中断。
位 5	**LBDLEN**：LIN 断开符检测长度（LIN break detection length） 该位用来选择是 11 位还是 10 位的断开符检测 0：10 位的断开符检测； 1：11 位的断开符检测。 *注意：LBDLEN 可用于 LIN 模式及其他模式下的断开帧的检测长度控制，且检测长度和 LIN 模式相同。*
位 4	保留位，硬件强制为 0
位 3：0	**ADDR[3：0]**：本设备的 USART 节点地址 该位域给出本设备 USART 节点的地址。这是在多处理器通信下的静默模式中使用的，使用地址标记来唤醒某个 USART 设备。

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

16.6.7 控制寄存器3（USART_CTRL3）

地址偏移：0x14

复位值：0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															
res															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留					CTS IEN	CTS EN	RTS EN	DMA TEN	DMA REN	SCM EN	NAC KEN	HALF SEL	IRD ALP	IRD AEN	ERR IEN
res					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31：11	保留位，硬件强制为 0

位 10	**CTSIEN**：CTSF 中断使能（CTSF interrupt enable） 0：禁止中断； 1：USART_STS 寄存器中的 CTSF 为'1'时生中断。 *注意：UART4、UART5、USART6、UART7 和 UART8 上不存在这一位。*
位 9	**CTSEN**：CTS 使能（CTS enable） 0：禁止 CTS 硬件流控制； 1：CTS 模式使能，只有 CTS 输入信号有效（拉成低电平）时才能发送数据。如果在数据传输的过程中，CTS 信号变成无效，那么发完这个数据后，传输就停止下来。如果当 CTS 为无效时，往数据寄存器里写数据，则要等到 CTS 有效时才会发送这个数据。 *注：UART4、UART5、USART6、UART7 和 UART8 上不存在这一位。*
位 8	**RTSEN**：RTS 使能（RTS enable） 0：禁止 RTS 硬件流控制； 1：RTS 中断使能，只有接收缓冲区内有空余的空间时才请求下一个数据。当前数据发送完成后，发送操作就需要暂停下来。如果可以接收数据了，将 RTS 输出置为有效（拉至低电平）。 *注：UART4、UART5、USART6、UART7 和 UART8 上不存在这一位。*
位 7	**DMATEN**：DMA 使能发送（DMA transmitter enable） 该位由软件设置或清除。 0：禁止发送时的 DMA 模式。 1：使能发送时的 DMA 模式；
位 6	**DMAREN**：DMA 使能接收（DMA receiver enable） 该位由软件设置或清除。 0：禁止接收时的 DMA 模式。 1：使能接收时的 DMA 模式；
位 5	**SCMEN**：智能卡模式使能（Smart card mode enable） 该位用来使能智能卡模式 0：禁止智能卡模式； 1：使能智能卡模式。 *注： UART4、UART5、UART7 和 UART8 上不存在这一位。*
位 4	**NACKEN**：智能卡 NACKEN 使能（Smart card NACK enable） 0：校验错误出现时，不发送 NACK； 1：校验错误出现时，发送 NACK。 *注：UART4、UART5、UART7 和 UART8 上不存在这一位。*
位 3	**HALFSEL**：半双工选择（Half-duplex selection） 选择单线半双工模式 0：不选择半双工模式； 1：选择半双工模式。
位 2	**IRDALP**：红外低功耗（IrDA low-power） 该位用来选择普通模式还是低功耗红外模式 0：普通模式； 1：低功耗模式。
位 1	**IRDAEN**：红外模式使能（IrDA mode enable） 该位由软件设置或清除。 0：不使能红外模式； 1：使能红外模式。

位 0	**ERRIEN**：错误中断使能（Error interrupt enable） 在多缓冲区通信模式下，当有帧错误、过载或者噪声错误时（USART_STS 中的 FERR=1，或者 ORERR=1，或者 NERR=1）产生中断。 0：禁止中断； 1：只要 USART_CTRL3 中的 DMAREN=1，并且 USART_STS 中的 FERR=1，或者 ORERR=1，或者 NERR=1，则产生中断

16.6.8 保护时间和预分频寄存器（GTP）

地址偏移：0x18

复位值：0x0000

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

位 31：16	保留位，硬件强制为 0
位 15：8	**GTVAL[7：0]**：保护时间值（Guard time value） 该位域规定了以波特时钟为单位的保护时间。在智能卡模式下，需要这个功能。当保护时间过去后，才会设置发送完成标志。 *注：UART4 和 UART5 上不存在这一位。*
位 7：0	**DIV[7：0]**：预分频器值（Prescaler value） **-在红外（IrDA）低功耗模式下：** DIV[7：0]=红外低功耗波特率对系统时钟分频以获得低功耗模式下的频率： 源时钟被寄存器中的值（仅有 8 位有效）分频： 00000000：保留–不要写入该值； 00000001：对源时钟 1 分频； 00000010：对源时钟 2 分频； …… **-在红外（IrDA）的普通模式下**：DIV 只能设置为 00000001 **-在智能卡模式下：** DIV[4：0]：预分频值对系统时钟进行分频，给智能卡提供时钟。 寄存器中给出的值（低 5 位有效）乘以 2 后，作为对源时钟的分频因子 00000：保留–不要写入该值； 00001：对源时钟进行 2 分频； 00010：对源时钟进行 4 分频； 00011：对源时钟进行 6 分频； …… *注意：1．位[7：5]在智能卡模式下没有意义。* *2．UART4、UART5、UART7 和 UART8 上不存在这一位。*

17 串行外设接口（SPI）

17.1 SPI简介

SPI 接口可以配置为支持 SPI 协议或者支持 I2S 音频协议。

SPI 接口默认工作在 SPI 方式，可以通过软件把功能从 SPI 模式切换到 I2S 模式。

串行外设接口（SPI）允许芯片与外部设备以半/全双工、同步、串行方式通信。此接口可以被配置成主模式，并为外部从设备提供通信时钟（SCK）。接口还能以多主配置方式工作。它可用于多种用途，包括使用一条双向数据线的双线单工同步传输，还可使用 CRC 校验的可靠通信。

I2S 也是一种 3 引脚的同步串行接口通讯协议。它支持四种音频标准，包括飞利浦 I2S 标准，MSB 和 LSB 对齐标准，以及 PCM 标准。它在半双工通讯中，可以工作在主和从 2 种模式下。当它作为主设备时，通过接口向外部的从设备提供时钟信号。

警告：由于 SPI3/I2S3 的部分引脚与 JTAG 引脚共享（SPI3_NSS/I2S3_WS 与 JTDI，SPI3_SCK/I2S3_CK 与 JTDO），因此这些引脚不受 IO 控制器控制，他们（在每次复位后）被默认保留为 JTAG 用途。如果用户想把引脚配置给 SPI3/I2S3，必须（在调试时）关闭 JTAG 并切换至 SWD 接口，或者（在标准应用时）同时关闭 JTAG 和 SWD 接口。详见章节 7.4.4：JTAG/SWD 复用功能重映射。

17.2 主要特点

17.2.1 SPI特点

- 3线全双工同步传输
- 带或不带第三根双向数据线的双线单工同步传输
- 8或16位传输帧格式选择
- 主或从操作
- 支持多主模式
- 10个主模式波特率预分频系数（最大为$f_{PCLK}/2$）
- 从模式频率（最大为fPCLK/2）
- 主模式和从模式的快速通信
- 主模式和从模式下均可以由软件或硬件进行NSS管理：主/从操作模式的动态改变
- 可编程的时钟极性和相位
- 可编程的数据顺序，MSB在前或LSB在前
- 可触发中断的专用发送和接收标志
- SPI总线忙状态标志
- 支持可靠通信的硬件CRC
  - 在发送模式下，CRC 值可以被作为最后一个字节发送
  - 在全双工模式中对接收到的最后一个字节自动进行 CRC 校验
- 可触发中断的主模式故障、过载以及CRC错误标志
- 支持DMA功能的2字节发送和接收缓冲器：产生发送和接受请求

17.2.2 I^2S功能

- 全双工通信
- 主或者从操作
- 8位线性可编程预分频器，获得精确的音频采样频率（8KHz到192kHz）
- 数据格式可以是16位，24位或者32位

- 音频信道固定数据包帧为16位（16位数据帧）或32位（16、24或32位数据帧）
- 可编程的时钟极性（稳定态）
- 从发送模式下的下溢标志位和主/从接收模式下的上溢标志位
- 16位数据寄存器用来发送和接收，在通道两端各有一个寄存器
- 支持的I²S协议：

  —I2S 飞利浦标准

  —MSB 对齐标准（左对齐）

  —LSB 对齐标准（右对齐）

  —PCM 标准（16 位通道帧上带长或短帧同步或者 16 位数据帧扩展为 32 位通道帧）
- 数据方向总是MSB在先
- 发送和接收都具有DMA能力
- 主时钟可以输出到外部音频设备，比率固定为256xFs（Fs为音频采样频率）

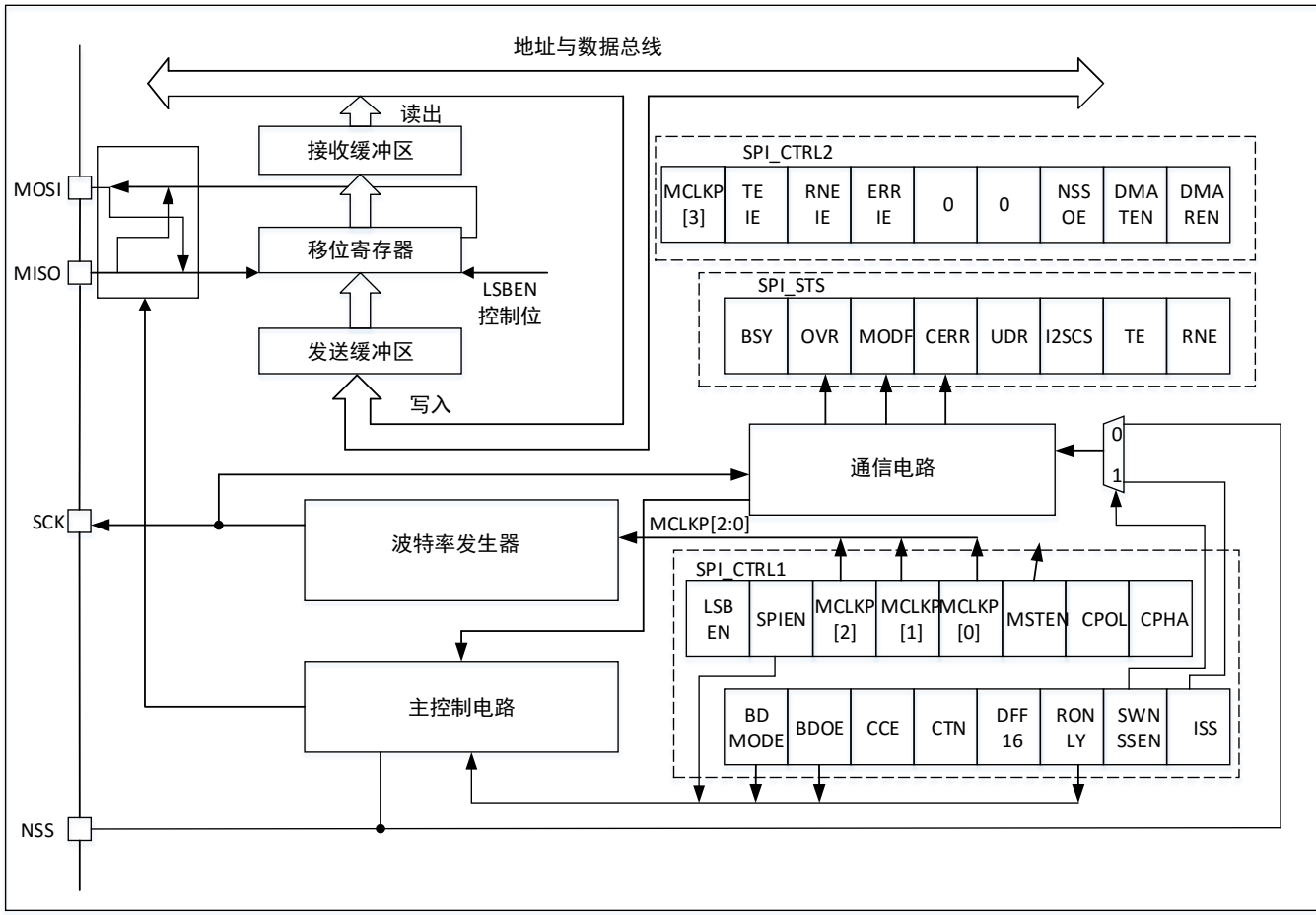
17.3 功能描述

17.3.1 SPI功能描述

17.3.1.1 概述

SPI 的方框图见下图。

图 17-1 SPI框图

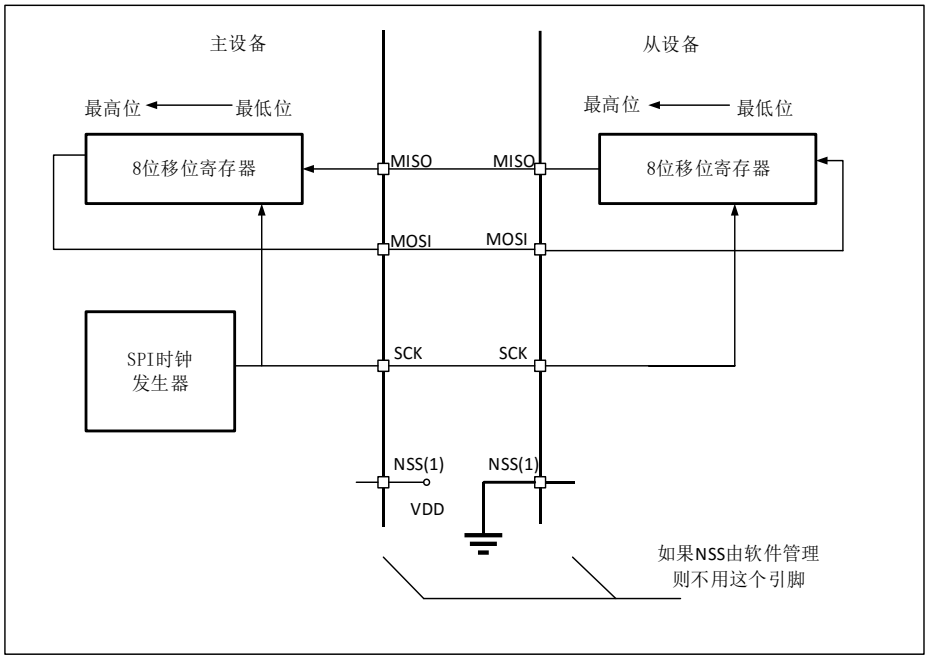


通常 SPI 通过 4 个引脚与外部器件相连：

- MISO：主设备输入/从设备输出引脚。该引脚在从模式下发送数据，在主模式下接收数据。
- MOSI：主设备输出/从设备输入引脚。该引脚在主模式下发送数据，在从模式下接收数据。
- SCK：串口时钟，作为主设备的输出，从设备的输入。
- NSS：从设备选择。这是一个可选的引脚，用来选择主/从设备。它的功能是用来作为“片选引脚”，让主设备可以单独地与特定从设备通讯，避免数据线上的冲突。从设备的NSS引脚可以由主设备的一个标准I/O引脚来驱动。一旦被使能（NSSOE位），NSS引脚也可以作为输出引脚，并在SPI处于主模式时拉低；此时，所有的SPI设备，如果它们的NSS引脚连接到主设备的NSS引脚，则会检测到低电平，如果它们被设置为NSS硬件模式，就会自动进入从设备状态。当配置为主设备、NSS配置为输入引脚（MSTEN=1，NSSOE=0）时，如果NSS被拉低，则这个SPI设备进入主模式失败状态：即MSTEN位被自动清除，此设备进入从模式。

下图是一个单主和单从设备互连的例子。

图 17-2 单主和单从应用



注意： 这里 NSS 引脚设置为输入。

MOSI 脚相互连接，MISO 脚相互连接。这样，数据在主和从之间串行地传输（MSB 位在前）。通信总是由主设备发起。主设备通过 MOSI 脚把数据发送给从设备，从设备通过 MISO 引脚回传数据。这意味全双工通信的数据输出和数据输入是用同一个时钟信号同步的；时钟信号由主设备通过 SCK 脚提供。

从选择（NSS）脚管理

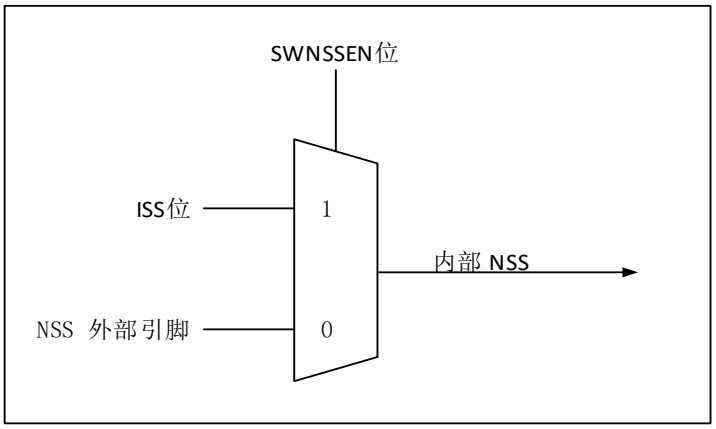
有 2 种 NSS 模式：

- 软件NSS模式：可以通过设置SPI_CTRL1寄存器的SWNSSEN位来使能这种模式（见图 17-3）。在这种模式下NSS引脚可以用作它用，而内部NSS信号电平可以通过写SPI_CTRL1的ISS位来驱动
- 硬件NSS模式，分两种情况：
  - NSS 输出被使能：当 AT32F403A/407 工作为主 SPI，并且 NSS 输出已经通过 SPI_CTRL2 寄存器的 NSSOE 位使能，这时 NSS 引脚被拉低，所有 NSS 引脚与这个主 SPI 的 NSS 引脚相连并配置为硬件 NSS 的 SPI 设备，将自动变成从 SPI 设备。当一个 SPI 设备需要发送广播数据，它必须拉低 NSS 信号，以通知所有其它的设备它是主设备；如果它不能拉低 NSS，

这意味着总线上有另外一个主设备在通信，这时将产生一个硬件失败错误（HardFault）。

— NSS 输出被关闭：允许操作于多主环境。

图 17-3 硬件/软件的从选择管理



时钟信号的相位和极性

SPI_CTRL1 寄存器的 CPOL 和 CPHA 位，能够组合成四种可能的时序关系。CPOL（时钟极性）位控制在没有数据传输时时钟的空闲状态电平，此位对主模式和从模式下的设备都有效。如果 CPOL 被清’0’，SCK 引脚在空闲状态保持低电平；如果 CPOL 被置’1’，SCK 引脚在空闲状态保持高电平。

如果 CPHA（时钟相位）位被置’1’，SCK 时钟的第二个边沿（CPOL 位为 0 时就是下降沿，CPOL 位为’1’时就是上升沿）进行数据位的采样，数据在第二个时钟边沿被锁存。如果 CPHA 位被清’0’，SCK 时钟的第一边沿（CPOL 位为’0’时就是上升沿，CPOL 位为’1’时就是下降沿）进行数据位采样，数据在第一个时钟边沿被锁存。

CPOL 时钟极性和 CPHA 时钟相位的组合选择数据捕捉的时钟边沿。图 17-4 显示了 SPI 传输的 4 种 CPHA 和 CPOL 位组合。此图可以解释为主设备和从设备的 SCK 脚、MISO 脚、MOSI 脚直接连接的主或从时序图。

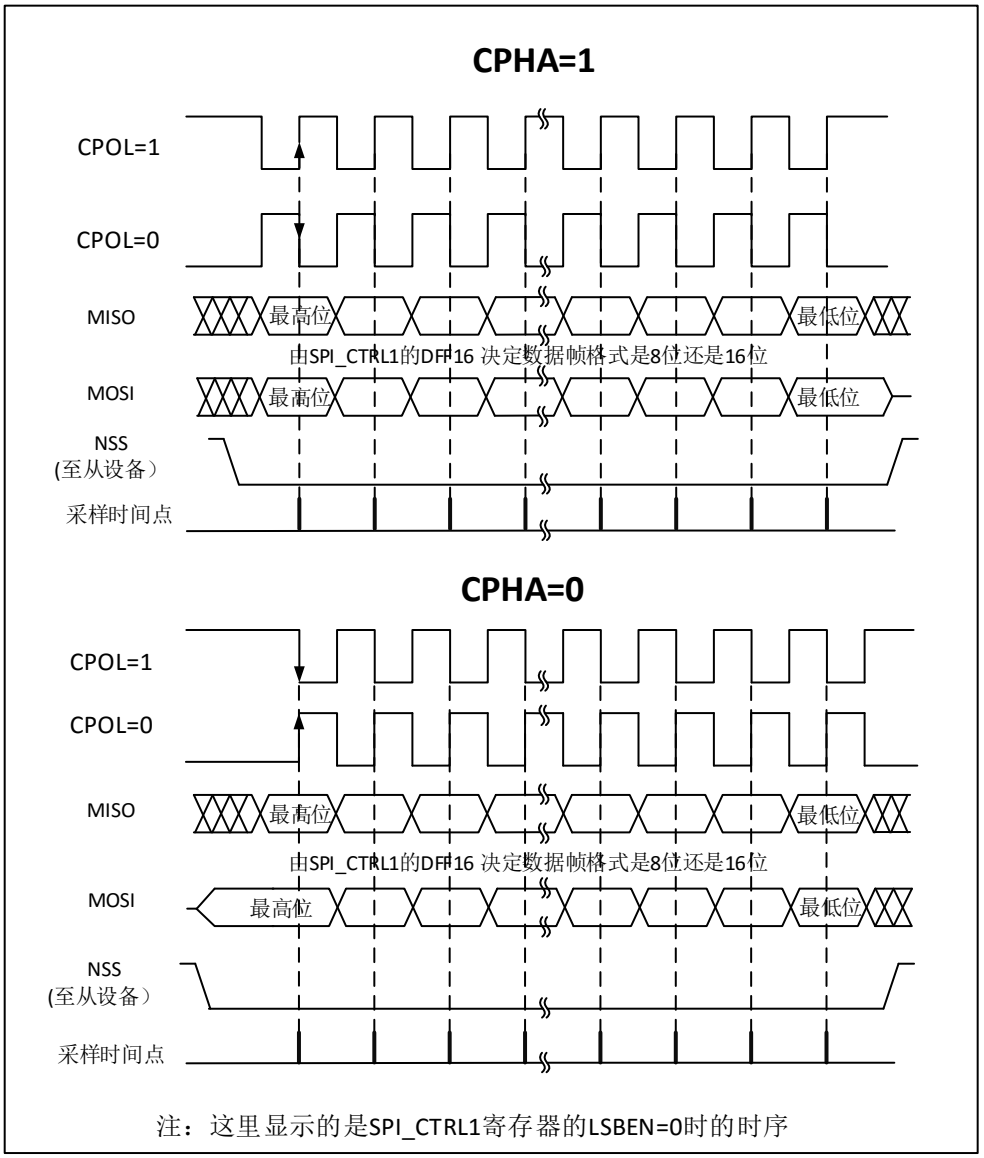
注意： *1.在改变 CPOL/CPHA 位之前，必须清除 SPIEN 位将 SPI 禁止。*

2.主和从必须配置成相同的时序模式。

3.SCK 的空闲状态必须和 SPI_CTRL1 寄存器指定的极性一致（CPOL 为’1’时，空闲时应上拉 SCK 为高电平；CPOL 为’0’时，空闲时应下拉 SCK 为低电平）。

4.数据帧格式（8 位或 16 位）由 SPI_CTRL1 寄存器的 DFF16 位选择，并且决定发送/接收的数据长度。

图 17-4 数据时钟时序图



注：这里显示的是SPI_CTRL1寄存器的LSBEN=0时的时序

数据帧格式

根据SPI_CTRL1寄存器中的LSBEN位，输出数据位时可以MSB在先也可以LSB在先。根据SPI_CTRL1寄存器的DFF16位，每个数据帧可以是8位或是16位。所选择的数据帧格式对发送和/或接收都有效。

17.3.1.2 配置SPI为从模式

在从模式下，SCK引脚用于接收从主设备来的串行时钟。SPI_CTRL1寄存器中MCLKP[3：0]的设置不影响数据传输速率。

注意：建议在主设备发送时钟之前使能SPI从设备，否则可能会发生意外的数据传输。在通信时钟的第一个边沿到来之前或正在进行的通信结束之前，从设备的数据寄存器必须就绪。在使能从设备和主设备之前，通信时钟的极性必须处于稳定的数值。

请按照以下步骤配置SPI为从模式：

配置步骤

1. 设置DFF16位以定义数据帧格式为8位或16位。
2. 选择CPOL和CPHA位来定义数据传输和串行时钟之间的相位关系（见图 17-4）。为保证正确的数据传输，从设备和主设备的CPOL和CPHA位必须配置成相同的方式。
3. 帧格式（SPI_CTRL1寄存器中的LSBEN位定义的”MSB在前”还是”LSB在前”）必须与主设备相同。

4. 硬件模式下（参考从选择（NSS）脚管理部分），在完整的数据帧（8位或16位）传输过程中，NSS引脚必须为低电平。在NSS软件模式下，设置SPI_CTRL1寄存器中的SWNSSEN位并清除ISS位。
5. 清除MSTEN位、设置SPIEN位（SPI_CTRL1寄存器），使相应引脚工作于SPI模式下。在这个配置中，MOSI引脚是数据输入，MISO引脚是数据输出。

数据发送过程

在写操作中，数据被写入发送缓冲器。

当从设备收到时钟信号，并且在 MOSI 引脚上出现第一个数据位时，发送过程开始（注：此时第一个位被发送出去）。余下的位（对于 8 位数据帧格式，还有 7 位；对于 16 位数据帧格式，还有 15 位）被装进移位寄存器。当发送缓冲器中的数据传输到移位寄存器时，SPI_STS 寄存器的 TE 标志被设置，如果设置了 SPI_CTRL2 寄存器的 TEIE 位，将会产生中断。

数据接收过程

对于接收器，当数据接收完成时：

- 移位寄存器中的数据传送到接收缓冲器，SPI_STS寄存器中的RNE标志被设置。
- 如果设置了SPI_CTRL2寄存器中的RNEIE位，则产生中断。

在最后一个采样时钟边沿后，RNE 位被置’1’，移位寄存器中接收到的数据字节被传送到接收缓冲器。当读 SPI_DT 寄存器时，SPI 设备返回这个接收缓冲器的数值。

读 SPI_DT 寄存器时，RNE 位被清除。

17.3.1.3 配置SPI为主模式

在主配置时，在 SCK 脚产生串行时钟。

配置步骤

1. 通过SPI_CTRL1寄存器的MCLKP[3：0]位定义串行时钟波特率。
2. 选择CPOL和CPHA位，定义数据传输和串行时钟间的相位关系（见图 17-4）。
3. 设置DFF16位来定义8位或16位数据帧格式。
4. 配置SPI_CTRL1寄存器的LSBEN位定义帧格式。
5. 如果需要NSS引脚工作在输入模式，硬件模式下，在整个数据帧传输期间应把NSS脚连接到高电平；在软件模式下，需设置SPI_CTRL1寄存器的SWNSSEN位和ISS位。如果NSS引脚工作在输出模式，则只需设置NSSOE位。
6. 必须设置MSTEN位和SPIEN位（只当NSS脚被连到高电平，这些位才能保持置位）。在这个配置中，MOSI引脚是数据输出，而MISO引脚是数据输入。

数据发送过程

当写入数据至发送缓冲器时，发送过程开始。在发送第一个数据位时，数据字被并行地（通过内部总线）传入移位寄存器，而后串行地移出到 MOSI 脚上；MSB 在先还是 LSB 在先，取决于 SPI_CTRL1 寄存器中的 LSBEN 位的设置。数据从发送缓冲器传输到移位寄存器时 TE 标志将被置位，如果设置了 SPI_CTRL1 寄存器中的 TEIE 位，将产生中断。

数据接收过程

对于接收器来说，当数据传输完成时：

- 传送移位寄存器里的数据到接收缓冲器，并且RNE标志被置位。
- 如果设置了SPI_CTRL2寄存器中的RNEIE位，则产生中断。

在最后采样时钟沿，RNE 位被设置，在移位寄存器中接收到的数据字被传送到接收缓冲器。读 SPI_DT 寄存器时，SPI 设备返回接收缓冲器中的数据。

读 SPI_DT 寄存器将清除 RNE 位。一旦传输开始，如果下一个将发送的数据被放进了发送缓冲器，就可以维持一个连续的传输流。在试图写发送缓冲器之前，需确认 TE 标志应该为’1’。

注意： 在 NSS 硬件模式下，从设备的 NSS 输入由 NSS 引脚控制或另一个由软件驱动的 GPIO 引脚控制。

17.3.1.4 配置SPI为单工通信

SPI 模块能够以两种配置工作于单工方式：

- 1条时钟线和1条双向数据线；
- 1条时钟线和1条数据线（只接收或只发送）；

1 条时钟线和 1 条双向数据线（BDMODE=1）

设置 SPI_CTRL1 寄存器中的 BDMODE 位而启用此模式。在这个模式下，SCK 引脚作为时钟，主设备使用 MOSI 引脚而从设备使用 MISO 引脚作为数据通信。传输的方向由 SPI_CTRL1 寄存器里的 BDOE 控制，当这个位是’1’的时候，数据线是输出，否则是输入。

1 条时钟和 1 条单向数据线（BDMODE=0）

在这个模式下，SPI 模块可以或者作为只发送，或者作为只接收。

- 只发送模式类似于全双工模式（BDMODE=0，RONLY=0）：数据在发送引脚（主模式时是MOSI、从模式时是MISO）上传输，而接收引脚（主模式时是MISO、从模式时是MOSI）可以作为通用的I/O使用。此时，软件不必理会接收缓冲器中的数据（如果读出数据寄存器，它不包含任何接收数据）。
- 在只接收模式，可以通过设置SPI_CTRL1寄存器的RONLY位而关闭SPI的输出功能；此时，发送引脚（主模式时是MOSI、从模式时是MISO）被释放，可以作为其它功能使用。

配置并使能 SPI 模块为只接收模式的方式是：

- 在主模式时，一旦使能SPI，通信立即启动，当清除SPIEN位时立即停止当前的接收。在此模式下，不必读取BSY标志，在SPI通信期间这个标志始终为’1’。
- 在从模式时，只要NSS被拉低（或在NSS软件模式时，ISS位为’0’）同时SCK有时钟脉冲，SPI就一直在接收。

17.3.1.5 数据发送与接收过程

接收与发送缓冲器

在接收时，接收到的数据被存放在一个内部的接收缓冲器中；在发送时，在被发送之前，数据将首先被存放在一个内部的发送缓冲器中。

对 SPI_DT 寄存器的读操作，将返回接收缓冲器的内容；写入 SPI_DT 寄存器的数据将被写入发送缓冲器中。

主模式下开始传输

- 全双工模式（BDMODE=0并且RONLY=0）
  - 当写入数据到 SPI_DT 寄存器（发送缓冲器）后，传输开始；
  - 在传送第一位数据的同时，数据被并行地从发送缓冲器传送到 8 位的移位寄存器中，然后按顺序被串行地移位送到 MOSI 引脚上；
  - 与此同时，在 MISO 引脚上接收到的数据，按顺序被串行地移位进入 8 位的移位寄存器中，然后被并行地传送到 SPI_DT 寄存器（接收缓冲器）中。
- 单向的只接收模式（BDMODE=0并且RONLY=1）
  - SPIEN=1 时，传输开始；
  - 只有接收器被激活，在 MISO 引脚上接收到的数据，按顺序被串行地移位进入 8 位的移位寄存器中，然后被并行地传送到 SPI_DT 寄存器（接收缓冲器）中。
- 双向模式，发送时（BDMODE=1并且BDOE=1）
  - 当写入数据到 SPI_DT 寄存器（发送缓冲器）后，传输开始；
  - 在传送第一位数据的同时，数据被并行地从发送缓冲器传送到 8 位的移位寄存器中，然后

按顺序被串行地移位送到 MOSI 引脚上；

— 不接收数据。

- 双向模式，接收时（BDMODE=1并且BDOE=0）
  - — SPIEN=1 并且 BDOE=0 时，传输开始；
  - — 在 MOSI 引脚上接收到的数据，按顺序被串行地移位进入 8 位的移位寄存器中，然后被并行地传送到 SPI_DT 寄存器（接收缓冲器）中。
  - — 不激活发送器，没有数据被串行地送到 MOSI 引脚上。

从模式下开始传输

- 全双工模式（BDMODE=0并且RONLY=0）
  - — 当从设备接收到时钟信号并且第一个数据位出现在它的 MOSI 时，数据传输开始，随后的数据位依次移动进入移位寄存器；
  - — 与此同时，在传输第一个数据位时，发送缓冲器中的数据被并行地传送到 8 位的移位寄存器，随后被串行地发送到 MISO 引脚上。软件必须保证在 SPI 主设备开始数据传输之前在发送寄存器中写入要发送的数据。
- 单向的只接收模式（BDMODE=0并且RONLY=1）
  - — 当从设备接收到时钟信号并且第一个数据位出现在它的 MOSI 时，数据传输开始，随后数据位依次移动进入移位寄存器；
  - — 不启动发送器，没有数据被串行地传送到 MISO 引脚上。
- 双向模式，发送时（BDMODE=1并且BDOE=1）
  - — 当从设备接收到时钟信号并且发送缓冲器中的第一个数据位被传送到 MISO 引脚上的时候，数据传输开始；
  - — 在第一个数据位被传送到 MISO 引脚上的同时，发送缓冲器中要发送的数据被平行地传送到 8 位的移位寄存器中，随后被串行地发送到 MISO 引脚上。软件必须保证在 SPI 主设备开始数据传输之前在发送寄存器中写入要发送的数据；
  - — 不接收数据。
- 双向模式，接收时（BDMODE=1并且BDOE=0）
  - — 当从设备接收到时钟信号并且第一个数据位出现在它的 MISO 时，数据传输开始；
  - — 从 MISO 引脚上接收到的数据被串行地传送到 8 位的移位寄存器中，然后被平行地传送到 SPI_DT 寄存器（接收缓冲器）；
  - — 不启动发送器，没有数据被串行地传送到 MISO 引脚上。

处理数据的发送与接收

当数据从发送缓冲器传送到移位寄存器时，设置 TE 标志（发送缓冲器空），它表示内部的发送缓冲器可以接收下一个数据；如果在 SPI_CTRL2 寄存器中设置了 TEIE 位，则此时会产生一个中断；写入 SPI_DT 寄存器即可清除 TE 位。

注意：在写入发送缓冲器之前，软件必须确认 TE 标志为’1’，否则新的数据会覆盖已经在发送缓冲器中的数据。

在采样时钟的最后一个边沿，当数据被从移位寄存器传送到接收缓冲器时，设置 RNE 标志（接收缓冲器非空）；它表示数据已经就绪，可以从 SPI_DT 寄存器读出；如果在 SPI_CTRL2 寄存器中设置了 RNEIE 位，则此时会产生一个中断；读出 SPI_DT 寄存器即可清除 RXNIE 标志位。

在一些配置中，传输最后一个数据时，可以使用 BSY 标志等待数据传输的结束。

主或从模式下（BDMODE=0 并且 RONLY=0）全双工发送和接收过程模式

软件必须遵循下述过程，发送和接收数据（见图 17-5 和图 17-6）：

1. 设置SPIEN位为’1’，使能SPI模块；
2. 在SPI_DT寄存器中写入第一个要发送的数据，这个操作会清除TE标志；

3. 等待TE=1，然后写入第二个要发送的数据。等待RNE=1，然后读出SPI_DT寄存器并获得第一个接收到的数据，读SPI_DT的同时清除了RNE位。重复这些操作，发送后续的数据同时接收n-1个数据；
4. 等待RNE=1，然后接收最后一个数据；
5. 等待TE=1，在BSY=0之后关闭SPI模块。

也可以在响应 RNE 或 TE 标志的上升沿产生的中断的处理程序中实现这个过程。

图 17-5 主模式、全双工模式下（BDMODE=0并且RONLY=0）连续输出时，TE/RNE/BSY的变化示意图

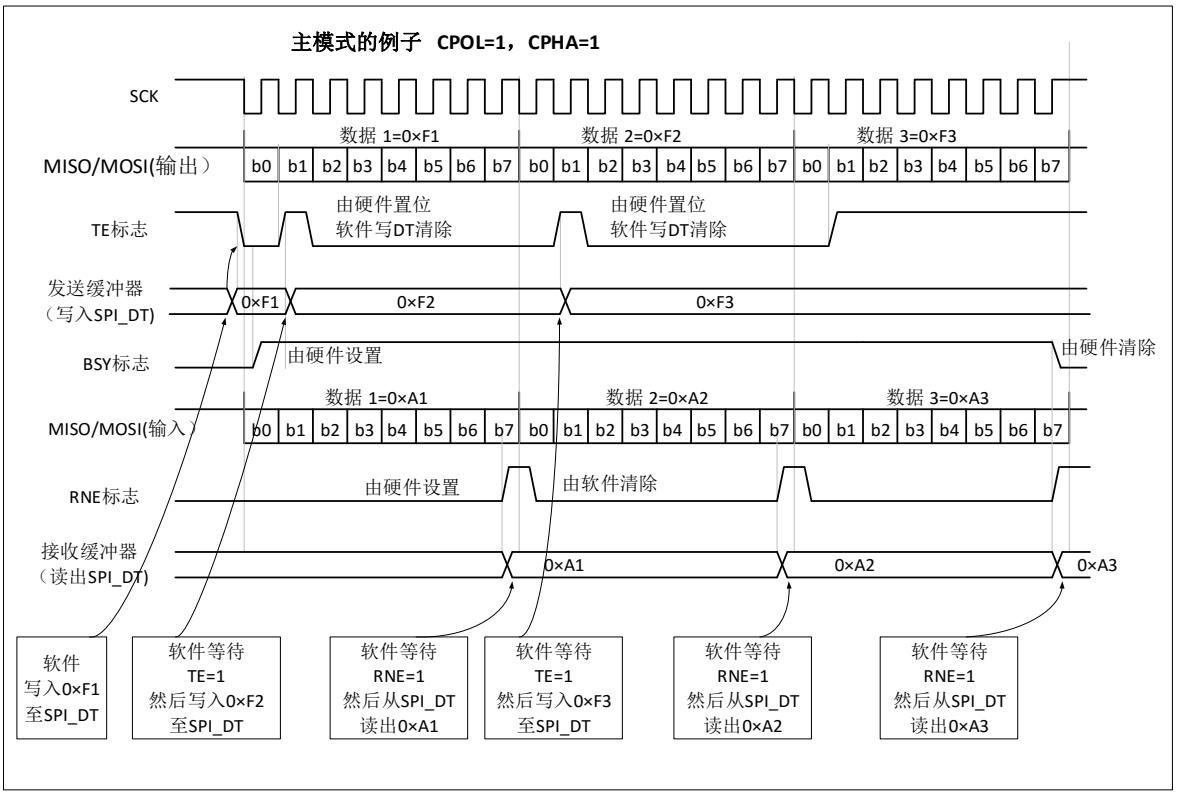
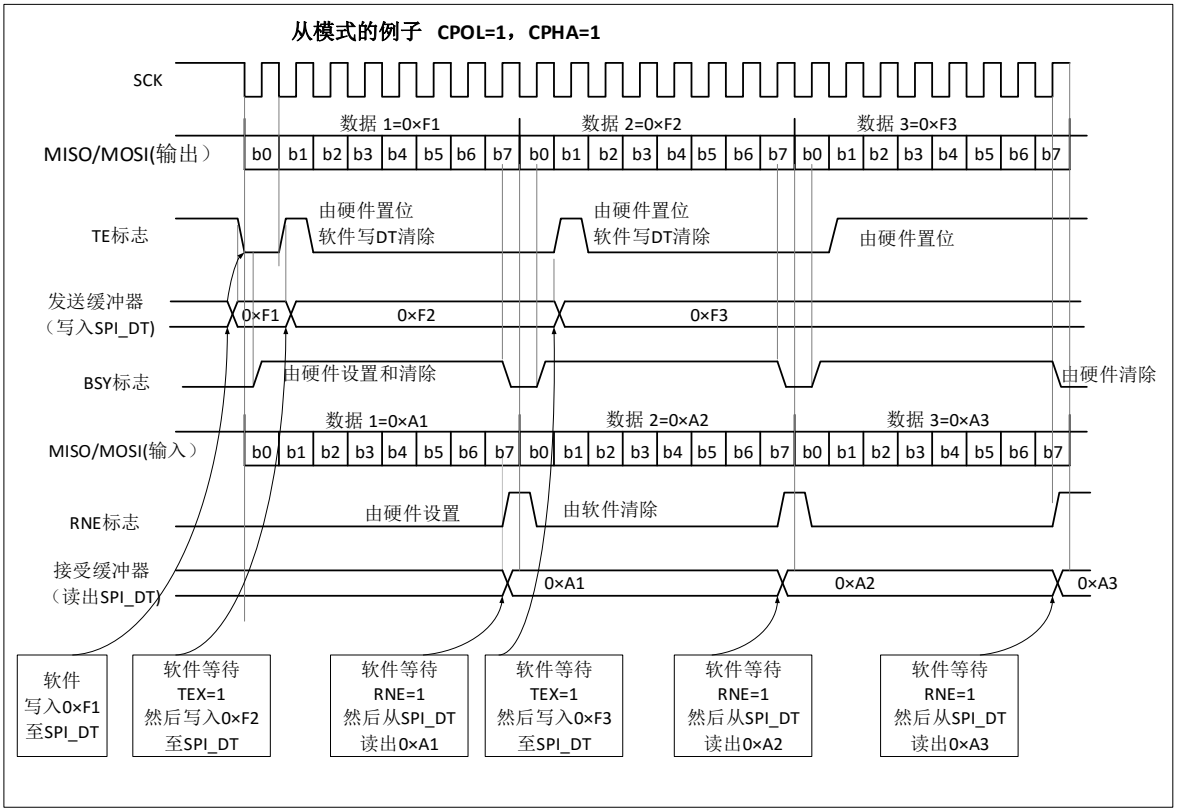


图 17-6 从模式、全双工模式下（BDMODE=0并且RONLY=0）连续传输时，TE/RNE/BSY的变化示意图。



只发送过程（BDMODE=0 并且 RONLY=0）

在此模式下，传输过程可以简要说明如下，使用 BSY 位等待传输的结束（见图 17-7 和图 17-8）：

1. 设置SPIEN位为’1’，使能SPI模块；
2. 在SPI_DT寄存器中写入第一个要发送的数据，这个操作会清除TE标志；
3. 等待TE=1，然后写入第二个要发送的数据。重复这个操作，发送后续的数据；
4. 写入最后一个数据到SPI_DT寄存器之后，等待TE=1；然后等待BSY=0，这表示最后一个数据的传输已经完成。

也可以在响应 TE 标志的上升沿产生的中断的处理程序中实现这个过程。

注意： *1.对于不连续的传输，在写入 SPI_DT 寄存器的操作与设置 BSY 位之间有 2 个 APB 时钟周期的延迟，因此在只发送模式下，写入最后一个数据后，最好先等待 TE=1，然后再等待 BSY=0。*

2.只发送模式下，在传输 2 个数据之后，由于不会读出接收到的数据，SPI_STS 寄存器中的 OVR 位会变为’1’。（注：软件不必理会这个 OVR 标志位）

图 17-7 主设备只发送模式（BDMODE=0并且RONLY=0）下连续传输时，TE/BSY变化示意图

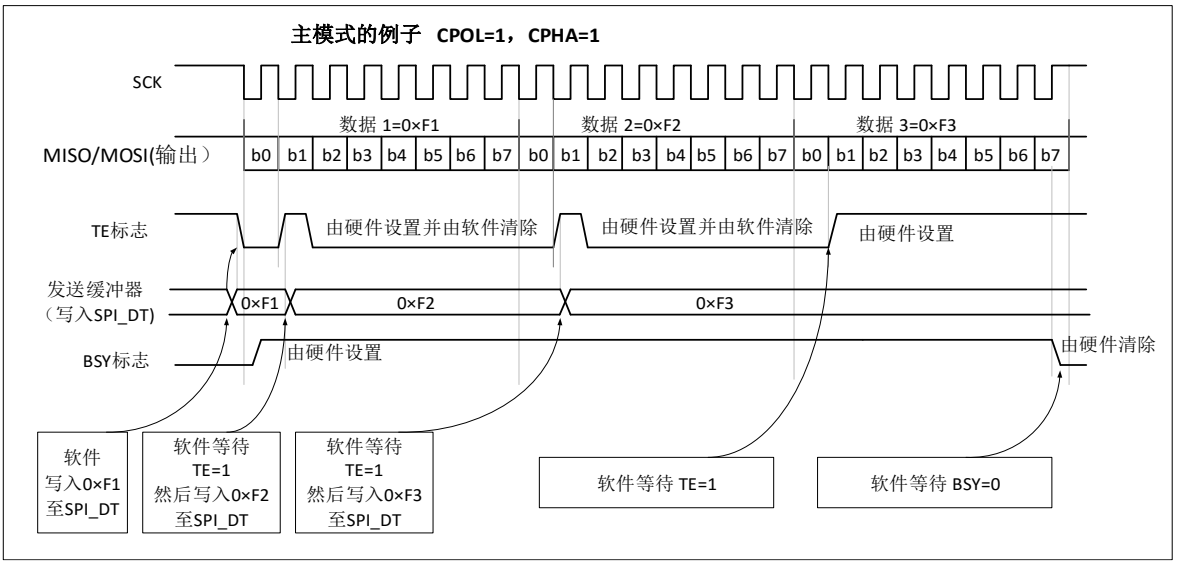
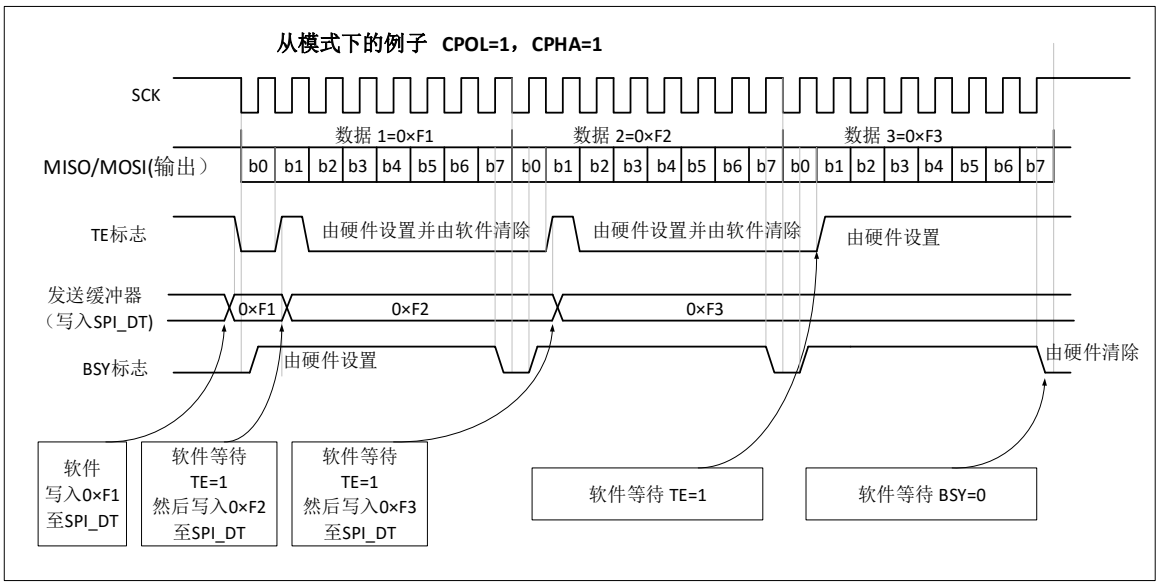


图 17-8 从设备只发送模式（BDMODE=0并且RONLY=0）下连续传输时，TE/BSY变化示意图



双向发送过程（BDMODE=1 并且 BDOE=1）

在此模式下，操作过程类似于只发送模式，不同的是：在使能 SPI 模块之前，需要在 SPI_CTRL1 寄存器中同时设置 BDMODE 和 BDOE 位为’1’。

单向只接收模式（BDMODE=0 并且 RONLY=1）

在此模式下，传输过程可以简要说明如下（见图 17-9）：

1. 在 SPI_CTRL1 寄存器中，设置 RONLY=1；
2. 设置 SPIEN=1，使能 SPI 模块：

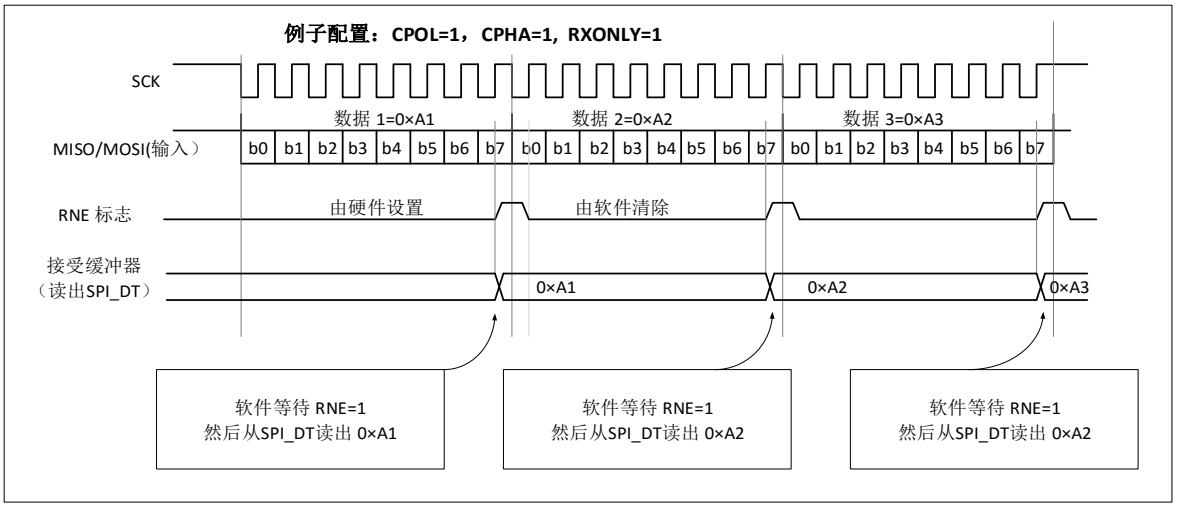
   a）主模式下，立刻产生 SCK 时钟信号，在关闭 SPI（SPIEN=0）之前，不断地接收串行数据；

   b）从模式下，当 SPI 主设备拉低 NSS 信号并产生 SCK 时钟时，接收串行数据。
3. 等待 RNE=1，然后读出 SPI_DT 寄存器以获得收到的数据（同时会清除 RNE 位）。重复这个操作接收所有数据。

也可以在响应 RNE 标志的上升沿产生的中断的处理程序中实现这个过程。

注意： 如果在最后一个数据传输结束后关闭 SPI 模块，请按照第 17.3.1.9 节的建议操作。

图 17-9 只接收模式（BDMODE=0并且RONLY=1）下连续传输时，RNE变化示意图



双向接收过程（BDMODE=1 并且 BDOE=0）

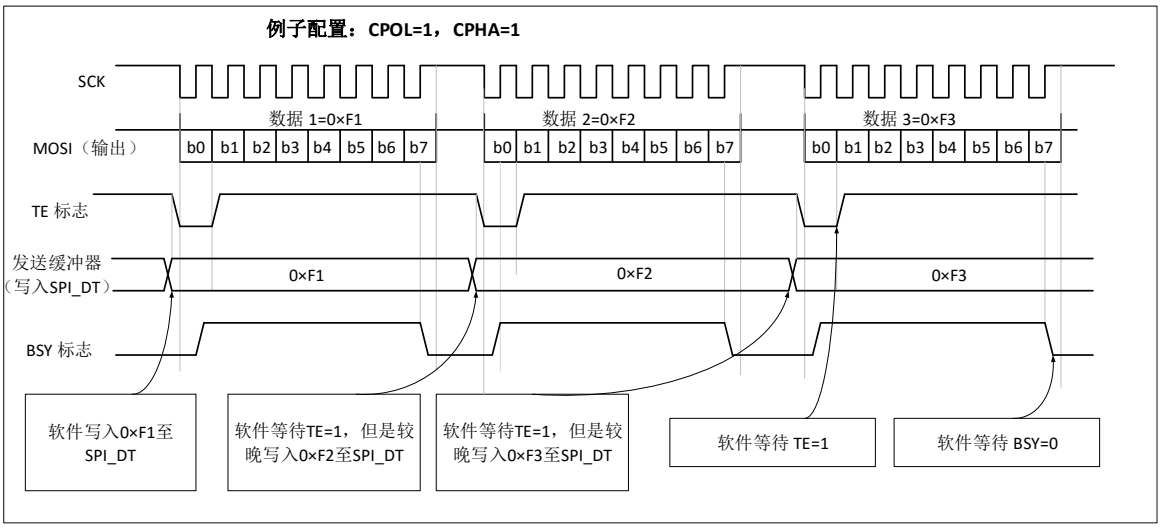
在此模式下，操作过程类似于只接收模式，不同的是：在使能 SPI 模块之前，需要在 SPI_CTRL1 寄存器中设置 BDMODE 为’1’并清除 BDOE 位为’0’。

连续和非连续传输

当在主模式下发送数据时，如果软件足够快，能够在检测到每次 TE 的上升沿（或 TE 中断），并立即在正在进行的传输结束之前写入 SPI_DT 寄存器，则能够实现连续的通信；此时，在每个数据项的传输之间的 SPI 时钟保持连续，同时 BSY 位不会被清除。

如果软件不够快，则会导致不连续的通信；这时，在每个数据传输之间会被清除（见下图）。在主模式的只接收模式下（RONLY=1），通信总是连续的，而且 BSY 标志始终为’1’。在从模式下，通信的连续性由 SPI 主设备决定。不管怎样，即使通信是连续的，BSY 标志会在每个数据项之间至少有一个 SPI 时钟周期为低（见图 17-8）。

图 17-10 非连续传输发送（BDMODE=0并且RONLY=0）时，TE/BSY变化示意图



17.3.1.6 CRC计算

CRC 校验用于保证全双工通信的可靠性。数据发送和数据接收分别使用单独的 CRC 计算器。通过对每一个接收位进行可编程的多项式运算来计算 CRC。CRC 的计算是在由 SPI_CTRL1 寄存器中 CPHA 和 CPOL 位定义的采样时钟边沿进行的。

注意： 该 SPI 接口提供了两种 CRC 计算方法，取决于所选的发送和/或接收的数据帧格式：8 位数据帧采用 CRC8；16 位数据帧采用 CRC16。

CRC 计算是通过设置 SPI_CTRL1 寄存器中的 CCE 位启用的。设置 CCE 位时同时复位 CRC 寄存器（SPI_RCRC 和 SPI_TCRC）。当设置了 SPI_CTRL1 的 CTN 位，SPI_TCRC 的内容将在当前字节发送之后发出。

在传输 SPI_TCRC 的内容时，如果在移位寄存器中收到的数值与 SPI_RCRC 的内容不匹配，则 SPI_STS 寄存器的 CERR 标志位被置 1。

如果在 TX 缓冲器中还有数据，CRC 的数值仅在数据字节传输结束后传送。在传输 CRC 期间，CRC 计算器关闭，寄存器的数值保持不变。

注意： 请参考产品说明书，以确认有此功能（不是所有型号都有此功能）。

SPI 通信可以通过以下步骤使用 CRC：

- 设置CPOL、CPHA、LSBEN、MCLKP、SWNSSEN、ISS和MSTEN的值；
- 在SPI_CPOLY寄存器输入多项式；
- 通过设置SPI_CTRL1寄存器CCE位使能CRC计算，该操作也会清除寄存器SPI_RCRC和SPI_TCRC；
- 设置SPI_CTRL1寄存器的SPIEN位启动SPI功能；
- 启动通信并且维持通信，直到只剩最后一个字节或者半字；
- 在把最后一个字节或半字写进发送缓冲器时，设置SPI_CTRL1的CTN位，指示硬件在发送完成最后一个数据之后，发送CRC的数值。在发送CRC数值期间，停止CRC计算；
- 当最后一个字节或半字被发送后，SPI发送CRC数值，CTN位被清除。同样，接收到的CRC与SPI_RCRC值进行比较，如果比较不相配，则设置SPI_STS上的CERR标志位，当设置了SPI_CTRL2寄存器的ERRIE时，则产生中断。

注意： 1、当 SPI 模块处于从设备模式时，请注意在时钟稳定之后再使能 CRC 计算，否则可能会得到错误的 CRC 计算结果。事实上，只要设置了 CCE 位，只要在 SCK 引脚上有输入时钟，不管 SPIEN 位的状态，都会进行 CRC 的计算。

2、当 SPI 时钟频率较高时，用户在发送 CRC 时必须小心。在 CRC 传输期间，使用 CPU 的时间应尽可能少；为了避免在接收最后的数据和 CRC 时出错，在发送 CRC 过程中应禁止函数调用。必须在发送/接收最后一个数据之前完成设置 CTN 位的操作。

3、当 SPI 时钟频率较高时，因为 CPU 的操作会影响 SPI 的带宽，建议采用 DMA 模式以避免 SPI 降低的速度。

4、当 AT32F403A/407 配置为从模式并且使用了 NSS 硬件模式，NSS 引脚应该在数据传输和 CRC 传输期间保持为低。

当配置 SPI 为从模式并且使用 CRC 的功能，即使 NSS 引脚为高时仍然会执行 CRC 的计算（注：当 NSS 信号为高时，如果 SCK 引脚上有时钟脉冲，则 CRC 计算会继续执行）。例如：当主设备交替地与多个从设备进行通信时，将会出现这种情况（注：此时要想办法避免 CRC 的误操作）。

每次传输或者接收 CRC value 之后会清除 CRC value。

17.3.1.7状态标志

应用程序通过 3 个状态标志可以完全监控 SPI 总线的状态。

发送缓冲器空闲标志（TE）

此标志为’1’时表明发送缓冲器为空，可以写下一个待发送的数据进入缓冲器中。当写入 SPI_DT 时，TE 标志被清除。

接收缓冲器非空（RNE）

此标志为’1’时表明在接收缓冲器中包含有效的接收数据。读 SPI 数据寄存器可以清除此标志。

忙（Busy）标志

BSY 标志由硬件设置与清除（写入此位无效果），此标志表明 SPI 通信层的状态。

当它被设置为’1’时，表明 SPI 正忙于通信，但有一个例外：在主模式的双向接收模式下（MSTEN=1、BDMODE=1 并且 BDOE=0），在接收期间 BSY 标志保持为低。

在软件要关闭 SPI 模块并进入停机模式（或关闭设备时钟）之前，可以使用 BSY 标志检测传输是否结束，这样可以避免破坏最后一次传输，因此需要严格按照下述过程执行。

BSY 标志还可以用于在多主系统中避免写冲突。

除了主模式的双向接收模式（MSTEN=1、BDMODE =1 并且 BDOE=0），当传输开始时，BSY 标志被置’1’。

以下情况时此标志将被清除为’0’：

- 当传输结束（主模式下，如果是连续通信的情况例外）；
- 当关闭SPI模块；
- 当产生主模式失效（MODF=1）。

如果通信不是连续的，则在每个数据项的传输之间，BSY 标志为低。

当通信是连续时：

- 主模式下：在整个传输过程中，BSY标志保持为高；
- 从模式下：在每个数据项的传输之间，BSY标志在一个SPI时钟周期中为低。

注意： 不要使用 BSY 标志处理每一个数据项的发送和接收，最好使用 TE 和 RNE 标志。

17.3.1.8 关闭SPI

当通讯结束，可以通过关闭 SPI 模块来终止通讯。清除 SPIEN 位即可关闭 SPI。在某些配置下，如果在传输还未完成时，就关闭 SPI 模块并进入停机模式，则可能导致当前的传输被破坏，而且 BSY 标志也变得不可信。

为了避免发生这种情况，关闭 SPI 模块时，建议按照下述步骤操作：

在主或从模式下的全双工模式（BDMODE=0，RONLY=0）

1.等待 RNE=1 并接收最后一个数据；

2.等待 TE=1；

3.等待 BSY=0；

4.关闭 SPI（SPIEN=0），最后进入停机模式（或关闭该模块的时钟）。

在主或从模式下的单向只发送模式（BDMODE=0，RONLY=0）或双向的发送模式（BDMODE=1，BDOE=1）

在 SPI_DT 寄存器中写入最后一个数据后：

1. 等待 TE=1；
2. 等待 BSY=0；
3. 关闭 SPI（SPIEN=0），最后进入停机模式（或关闭该模块的时钟）。

在主模式下的单向只接收模式（MSTEN=1，BDMODE=0，RONLY=1）或双向的接收模式（MSTEN=1，BDMODE=1，BDOE=0）

这种情况需要特别地处理，以保证 SPI 不会开始一次新的传输：

1. 等待倒数第二个（第 n-1 个）RNE=1；
2. 在关闭 SPI（SPIEN=0）之前等待一个 SPI 时钟周期（使用软件延迟）；
3. 在进入停机模式（或关闭该模块的时钟）之前等待最后一个 RNE=1。

注意： 在主模式下的双向只接收模式（MSTEN=1，BDMODE =1，BDOE=0）时，传输过程中 BSY 标志始终为低。

在从模式下的只接收模式（MSTEN=0，BDMODE=0，RONLY=1）或双向的接收模式（MSTEN=0，BDMODE=1，BDOE=0）

1. 可以在任何时候关闭 SPI（SPIEN=0），SPI 会在当前的传输结束后被关闭；
2. 如果希望进入停机模式，在进入停机模式（或关闭该模块的时钟）之前必须首先等待 BSY=0。

17.3.1.9 利用DMA的SPI通信

为了达到最大通信速度，需要及时往 SPI 发送缓冲器填数据，同样接收缓冲器中的数据也必须及时读走以防止溢出。为了方便高速率的数据传输，SPI 实现了一种采用简单的请求/应答的 DMA 机制。

当 SPI_CTRL2 寄存器上的对应使能位被设置时，SPI 模块可以发出 DMA 传输请求。发送缓冲器和接收缓冲器亦有各自的 DMA 请求。

- 发送时，在每次TE被设置为’1’时发出DMA请求，DMA控制器则写数据至SPI_DT寄存器，TE标志因此而被清除。
- 接收时，在每次RNE被设置为’1’时发出DMA请求，DMA控制器则从SPI_DT寄存器读出数据，RNE标志因此而被清除。

当只使用 SPI 发送数据时，只需使能 SPI 的发送 DMA 通道。此时，因为没有读取收到的数据，OVR 被置为’1’（译注：软件不必理会这个标志）。当只使用 SPI 接收数据时，只需使能 SPI 的接收 DMA 通道。

在发送模式下，当 DMA 已经传输了所有要发送的数据（DMA_ISTS 寄存器的 TCIF 标志变为’1’）后，可以通过监视 BSY 标志以确认 SPI 通信结束，这样可以避免在关闭 SPI 或进入停止模式时，破坏最后一个数据的传输。因此软件需要先等待 TE=1，然后等待 BSY=0。

注意： *在不连续的通信中，在写数据到 SPI_DT 的操作与 BSY 位被置为’1’之间，有 2 个 APB 时钟周期的延迟，因此，在写完最后一个数据后需要先等待 TE=1 再等待 BSY=0。*

图 17-11 使用DMA发送

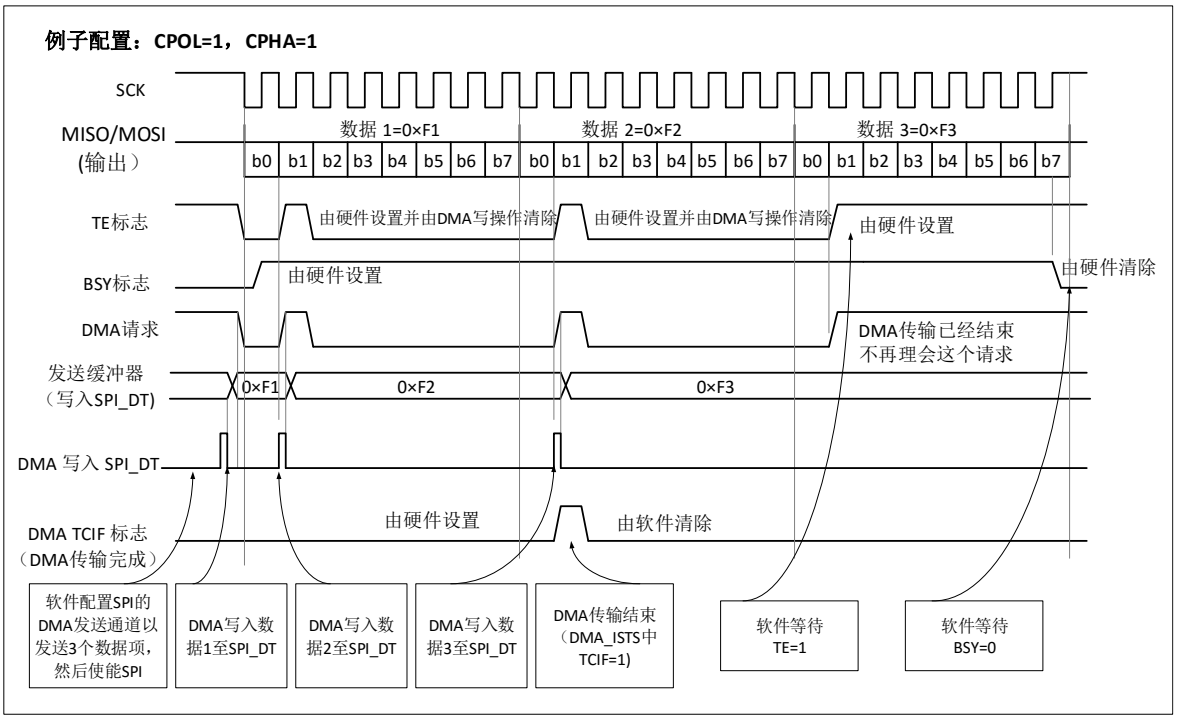
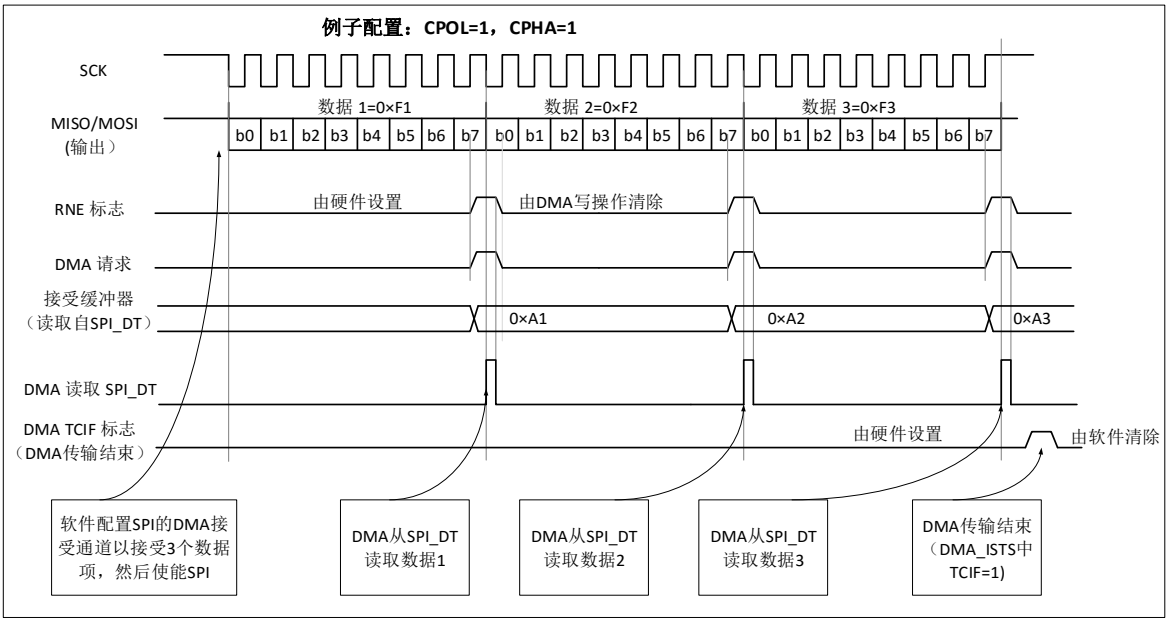


图 17-12 使用DMA接收



带 CRC 的 DMA 功能

当使能 SPI 使用 CRC 检验并且启用 DMA 模式时，在通信结束时，CRC 字节的发送和接收是自动完成的。

数据和 CRC 传输结束时，SPI_STS 寄存器的 CERR 标志为’1’表示在传输期间发生错误。

17.3.1.10 错误标志

主模式失效错误（MODF）

主模式失效仅发生在：NSS 引脚硬件模式管理下，主设备的 NSS 脚被拉低；或者在 NSS 引脚软件模式管理下，ISS 位被置为’0’时；MODF 位被自动置位。主模式失效对 SPI 设备有以下影响：

- MODF位被置为’1’，如果设置了ERRIE位，则产生SPI中断；
- SPIEN位被清为’0’。这将停止一切输出，并且关闭SPI接口；
- MSTEN位被清为’0’，因此强迫此设备进入从模式。

下面的步骤用于清除 MODF 位：

1. 当 MODF 位被置为’1’时，执行一次对 SPI_STS 寄存器的读或写操作；
2. 然后写 SPI_CTRL1 寄存器。

在有多个 MCU 的系统中，为了避免出现多个从设备的冲突，必须先拉高该主设备的 NSS 脚，再对 MODF 位进行清零。在完成清零之后，SPIEN 和 MSTEN 位可以恢复到它们的原始状态。

出于安全的考虑，当 MODF 位为’1’时，硬件不允许设置 SPIEN 和 MSTEN 位。

通常配置下，从设备的 MODF 位不能被置为’1’。然而，在多主配置里，一个设备可以在设置了

MODF 位的情况下，处于从设备模式；此时，MODF 位表示可能出现了多主冲突。中断程序可以执行一个复位或返回到默认状态来从错误状态中恢复。

溢出错误

当主设备已经发送了数据字节，而从设备还没有清除前一个数据字节产生的 RNE 时，即为溢出错误。当产生溢出错误时：

- OVR位被置为’1’；当设置了ERRIE位时，则产生中断。

此时，接收器缓冲器的数据不是主设备发送的新数据，读 SPI_DT 寄存器返回的是之前未读的数据，所有随后传送的数据都被丢弃。

依次读出 SPI_DT 寄存器和 SPI_STS 寄存器可将 OVR 清除。

CRC 错误

当设置了 SPI_CTRL 寄存器上的 CCE 位时，CRC 错误标志用来核对接收数据的有效性。如果移位寄存器中接收到的值（发送方发送的 SPI_TCRC 数值）与接收方 SPI_RCRC 寄存器中的数值不匹配，则 SPI_STS 寄存器上的 CERR 标志被置位为’1’。

17.3.1.11 SPI中断

表 17-1 SPI中断请求

中断事件	事件标志	使能控制位
发送缓冲器空标志	TE	TEIE
接收缓冲器非空标志	RNE	RNEIE
主模式失效事件	MODF	ERRIE
溢出错误	OVR	
CRC 错误标志	CERR	

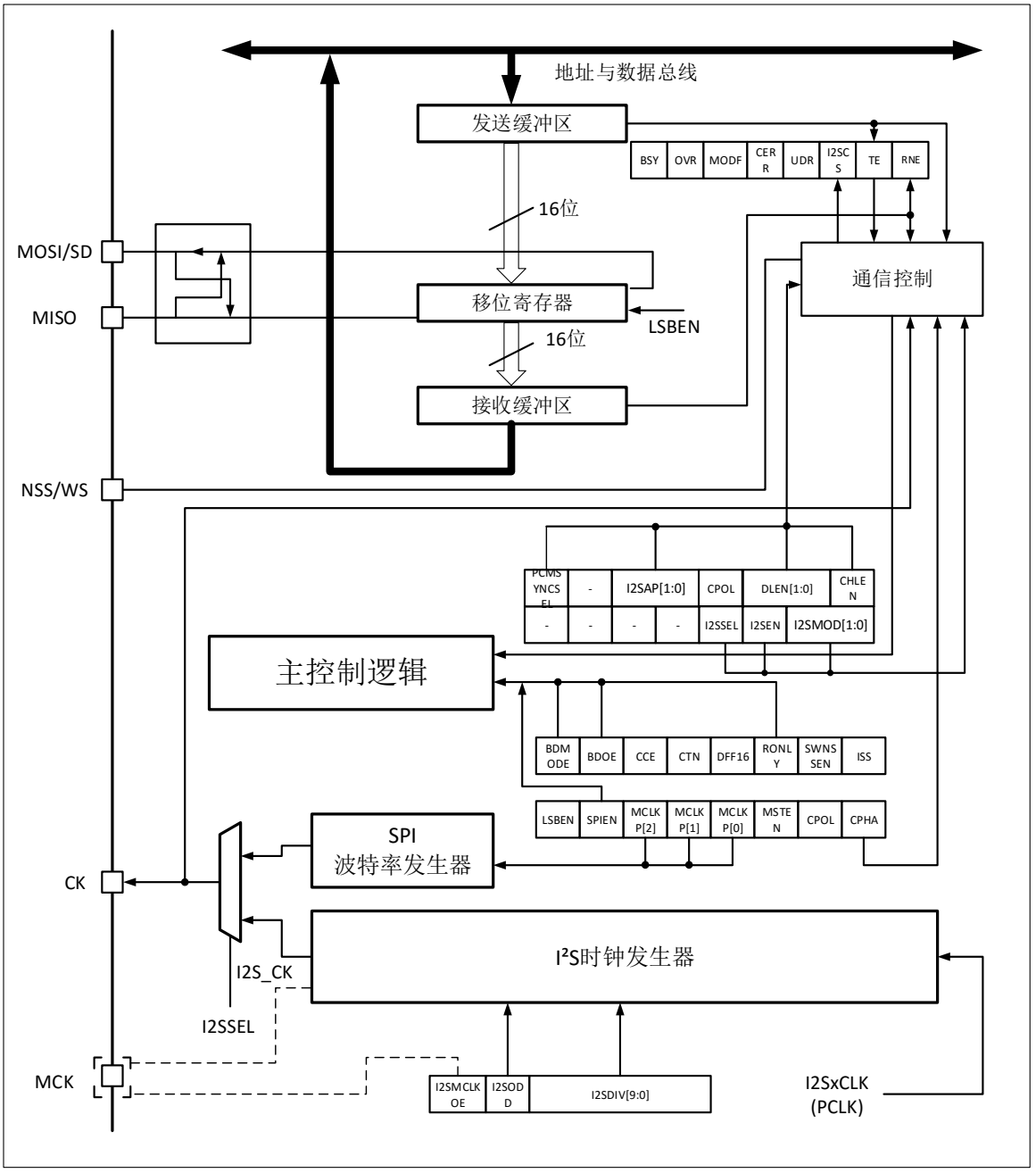
17.3.2 I^2S功能描述

所有的 AT32F403A/407 产品均支持 I^2S 音频协议。

17.3.2.1 I^2S功能描述

I^2S 的框图如下图所示：

图 17-13 I²S框图



通过将寄存器 SPI_I2SCTRL 的 I2SSEL 位置为'1'，即可使能 I²S 功能。此时，可以把 SPI 模块用作 I²S 音频接口。I²S 接口与 SPI 接口使用大致相同的引脚、标志和中断。

I²S 与 SPI 共用 3 个引脚：

- SD：串行数据（映射至MOSI引脚），用来发送和接收2路时分复用通道的数据；
- WS：字选（映射至NSS引脚），主模式下作为数据控制信号输出，从模式下作为输入；
- CK：串行时钟（映射至SCK引脚），主模式下作为时钟信号输出，从模式下作为输入。

  在某些外部音频设备需要主时钟时，可以另有一个附加引脚输出时钟：
- MCLK：主时钟（独立映射），在I2S配置为主模式，寄存器SPI_I2SCLKP的I2SMCLKOE位为'1'时，作为输出额外的时钟信号引脚使用。输出时钟信号的频率预先设置为256×Fs，其中Fs是音频信号的采样频率。

设置成主模式时，I²S 使用自身的时钟发生器来产生通信用的时钟信号。这个时钟发生器也是主时钟输出的时钟源。I²S 模式下有 2 个额外的寄存器，一个是与时钟发生器配置相关的寄存器 SPI_I2SCLKP，另一个是 I²S 通用配置寄存器 SPI_I2SCTRL（可设置音频标准、从/主模式、数据格式、数据包帧、时钟极性等参数）。

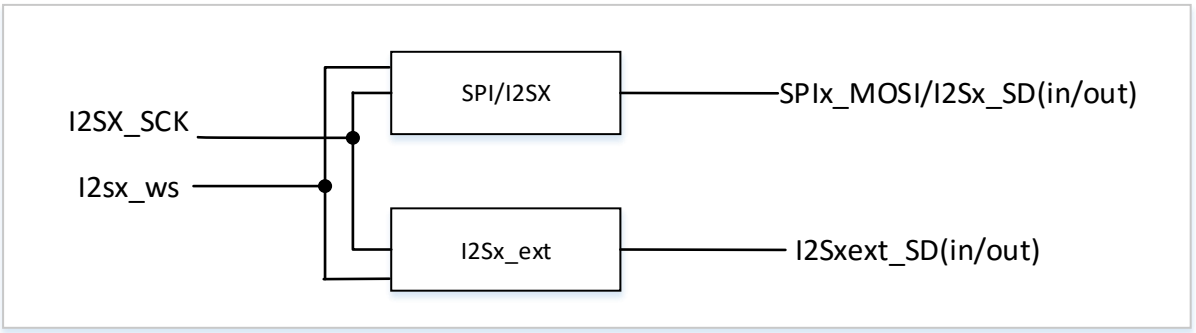
在 I²S 模式下不使用寄存器 SPI_CTRL1 和所有的 CRC 寄存器。同样，I²S 模式下也不使用寄存器 SPI_CTRL2 的 NSSOE 位，和寄存器 SPI_STS 的 MODF 位和 CERR 位。I²S 使用与 SPI 相同的寄存器 SPI_DT 用作 16 位宽模式数据传输。

17.3.2.2 I2S 全双工

为了支持I2S 全双工模式，额外例化了两个I2S 模块（I2S2_ext，I2S3_ext）。I2S2与I2S2_ext组合在一起支持全双工模式，I2S3与I2S3_ext组合在一起支持全双工模式。

注意： I2S2_ext 和I2S3_ext 只用于I2S 全双工模式。

图 17-14 I2S全双工结构图



1. x可以是2或者3

I2Sx 可以作为主机

- 半双工模式下，只有I2Sx可以输出SCK和WS。
- 全双工模式下，只有I2Sx可以输出SCK和WS。

新增的 I2Sx_ext 只能用 I2S 全双工模式下，I2Sx_ext 只可以配置成从机模式。

I2Sx 和 I2Sx_ext 都可以做发送或者接收方。

17.3.2.3 支持的音频协议

三线总线支持 2 个声道上音频数据的时分复用：左声道和右声道，但是只有一个 16 位寄存器用作发送或接收。因此，软件必须在对数据寄存器写入数据时，根据当前传输中的声道写入相应的数据；同样，在读取寄存器数据时，通过检查寄存器 SPI_STS 的 I2SCS 位来判明接收到的数据属于哪个声道。左声道总是先于右声道发送数据（I²SCS 位在 PCM 协议下无意义）。

有四种可用的数据和包帧组合。可以通过以下四种数据格式发送数据：

- 16位数据打包进16位帧
- 16位数据打包进32位帧
- 24位数据打包进32位帧
- 32位数据打包进32位帧

在使用 16 位数据扩展到 32 位帧时，前 16 位（MSB）是有意义的数据，后 16 位（LSB）被强制为 0，该操作不需要软件干预，也不需要有 DMA 请求（仅需要一次读/写操作）。

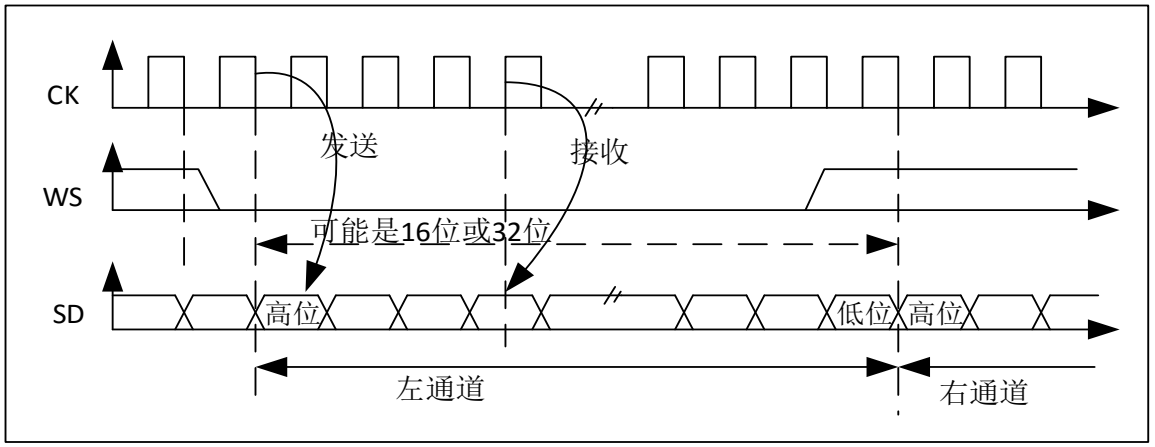
24 位和 32 位数据帧需要 CPU 对寄存器 SPI_DT 进行 2 次读或写操作，在使用 DMA 时，需要 2 次 DMA 传输。对于 24 位数据，扩展到 32 位后，最低 8 位由硬件置 0。对于所有的数据格式和通讯标准，总是先发送最高位（MSB）。

I²S 接口支持四种音频标准，可以通过设置寄存器 SPI_I2SCTRL 的 I2SAP[1：0]位和 PCMSYNCSEL 位来选择。

I²S 飞利浦标准

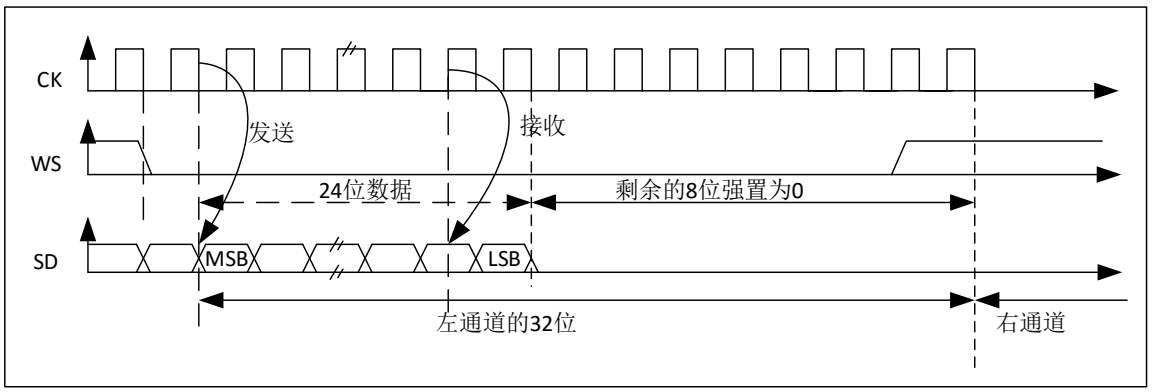
在此标准下，引脚 WS 用来指示正在发送的数据属于哪个声道。在发送第一位数据（MSB）前 1 个时钟周期，该引脚即为有效。

图 17-15 I²S飞利浦协议波形（16/32位全精度，CPOL=0）



发送方在时钟信号（CK）的下降沿改变数据，接收方在上升沿读取数据。WS 信号也在时钟信号的下降沿变化。

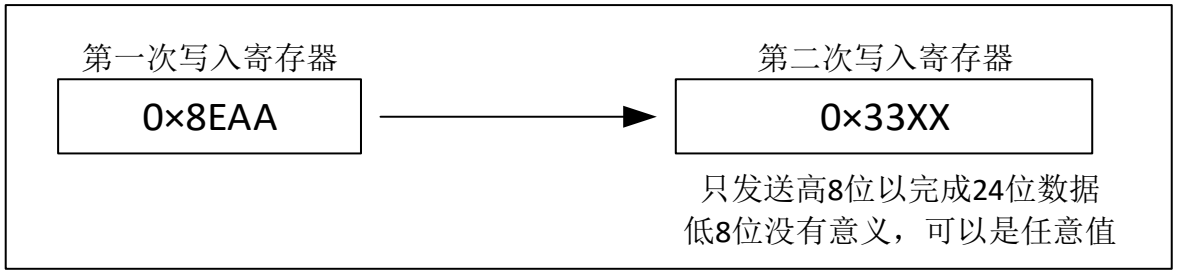
图 17-16 I²S飞利浦协议标准波形（24位帧，CPOL=0）



此模式需要对寄存器 SPI_DT 进行 2 次读或写操作。

- 在发送模式下：如果需要发送0x8EAA33（24位）：

图 17-17 发送0x8EAA33



- 在接收模式下：如果接收0x8EAA33：

图 17-18 接收0x8EAA33

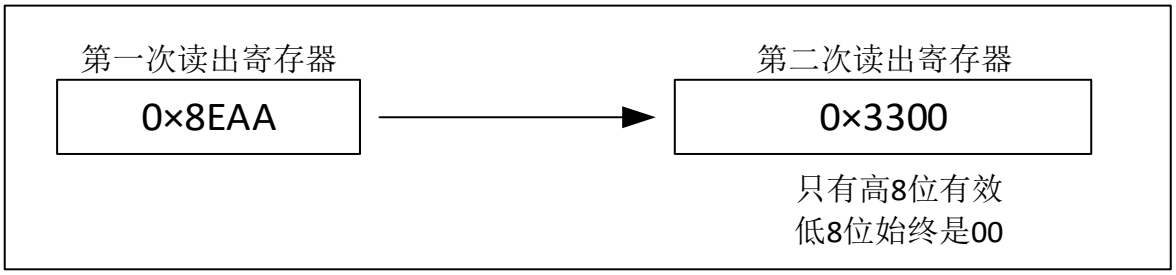
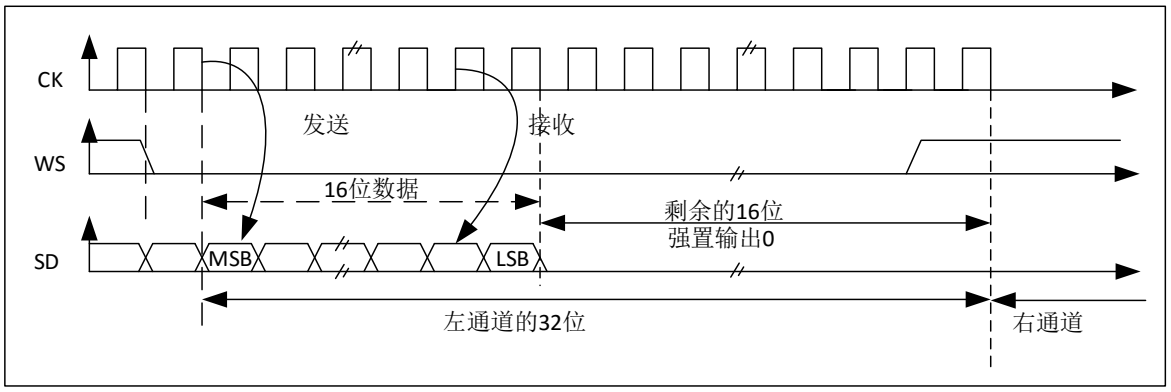


图 17-19 I²S飞利浦协议标准波形（16位扩展至32位包帧，CPOL=0）



在 I2S 配置阶段，如果选择将 16 位数据扩展到 32 位声道帧，只需要访问一次寄存器 SPI_DT。用来扩展到 32 位的低 16 位被硬件置为 0x0000。

如果待传输或者接收的数据是 0x76A3（扩展到 32 位是 0x76A30000），需要的操作如下图所示。

图 17-20 示例



在发送时需要将 MSB 写入寄存器 SPI_DT；标志位 TE 为’1’表示可以写入新的数据，如果允许了相应的中断，则可以产生中断。发送是由硬件完成的，即使还未发送出后 16 位的 0x0000，也会设置 TE 并产生相应的中断。

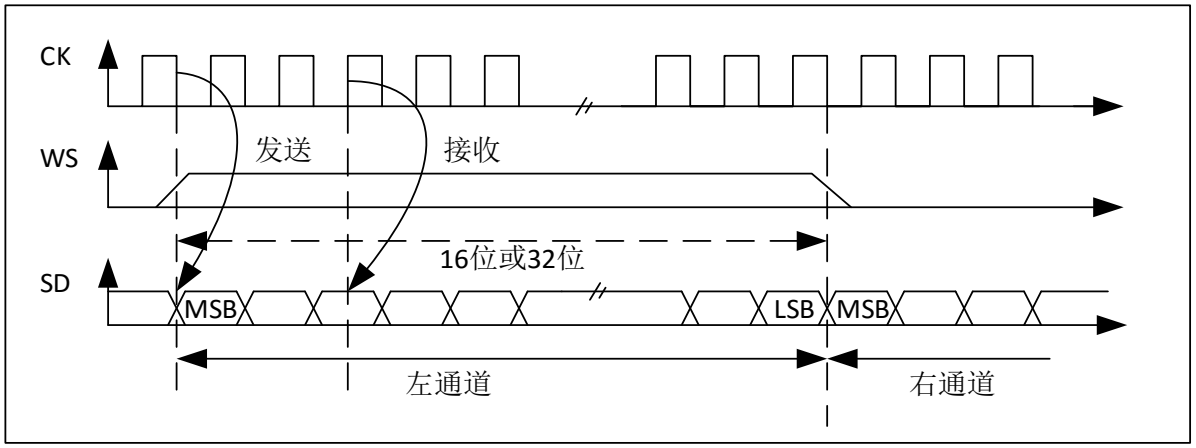
接收时，每次收到高 16 位半字（MSB）后，标志位 RNE 置’1’，如果允许了相应的中断，则可以产生中断。

这样，在 2 次读和写之间有更多的时间，可以防止下溢或者上溢的情况发生。

MSB 对齐标准

在此标准下，WS 信号和第一个数据位，即最高位（MSB）同时产生。

图 17-21 MSB对齐16位或32位全精度，CPOL=0



发送方在时钟信号的下降沿改变数据；接收方是在上升沿读取数据。

图 17-22 MSB对齐24位数据，CPOL=0

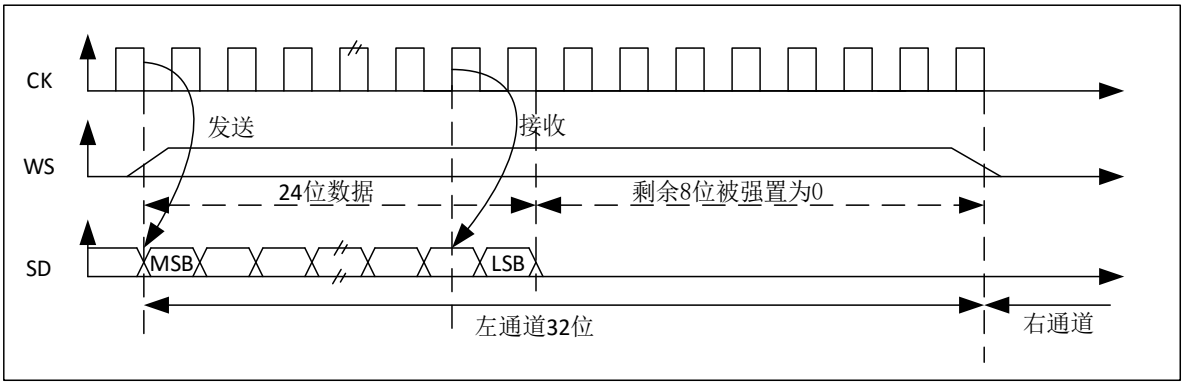
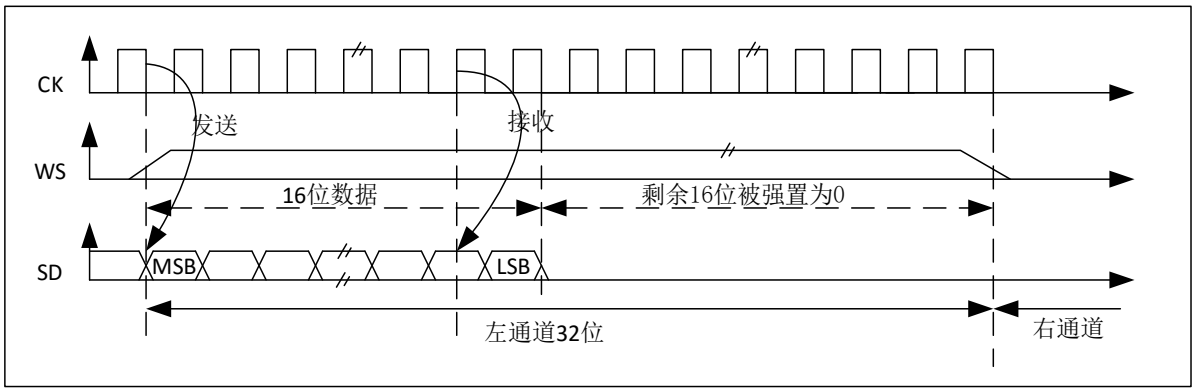


图 17-23 MSB对齐16位数据扩展到32位包帧，CPOL=0



LSB 对齐标准

此标准与 MSB 对齐标准类似（在 16 位或 32 位全精度帧格式下无区别）。

图 17-24 LSB对齐16位或32位全精度，CPOL=0

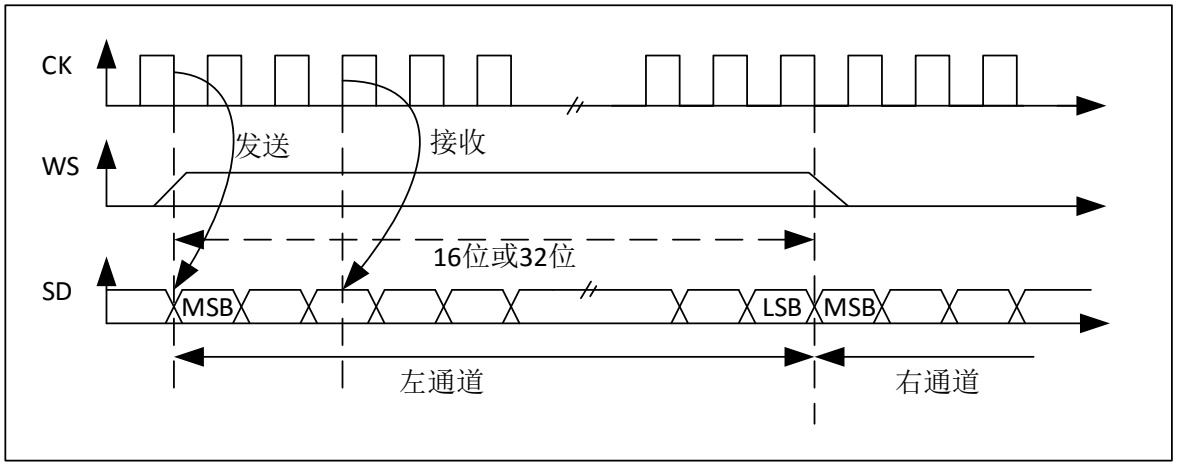
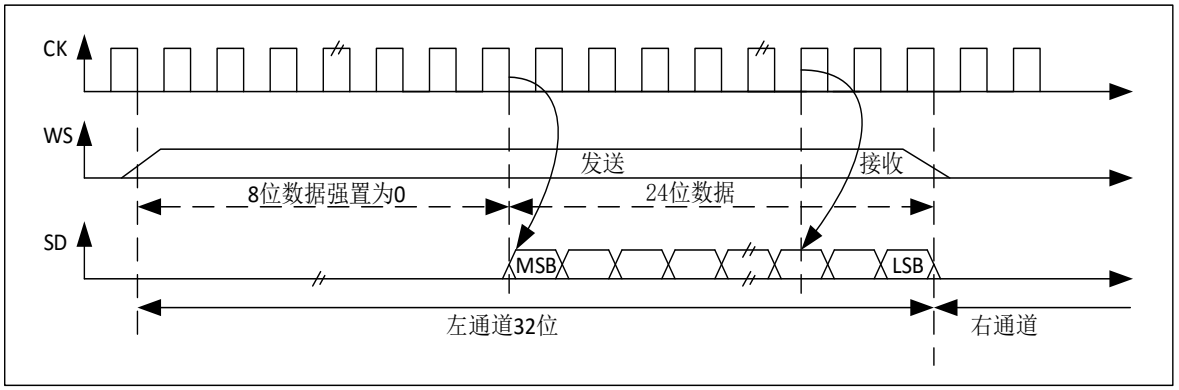


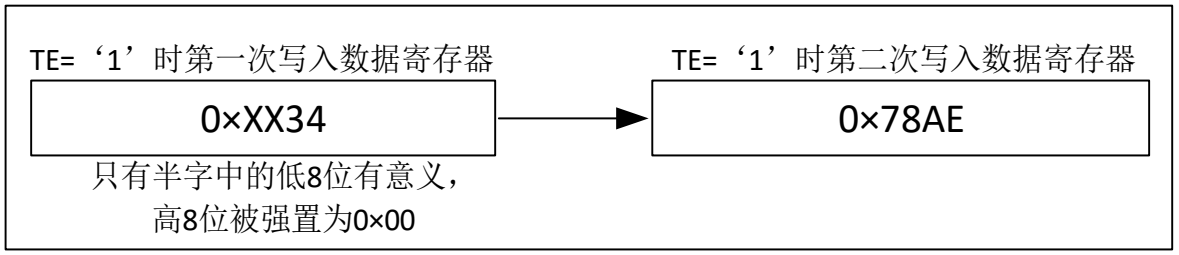
图 17-25 LSB对齐24位数据，CPOL=0



- 在发送模式下

  如果要发送数据 0x3478AE，需要通过软件或者 DMA 对寄存器 SPI_DT 进行 2 次写操作。操作流程如下图所示。

图 17-26 要求发送0x3478AE的操作



- 在接收模式下

  如果要接收数据 0x3478AE，需要在 2 个连续的 RNE 事件发生时，分别对寄存器 SPI_DT 进行 1 次读操作。

图 17-27 要求接收0x3478AE的操作

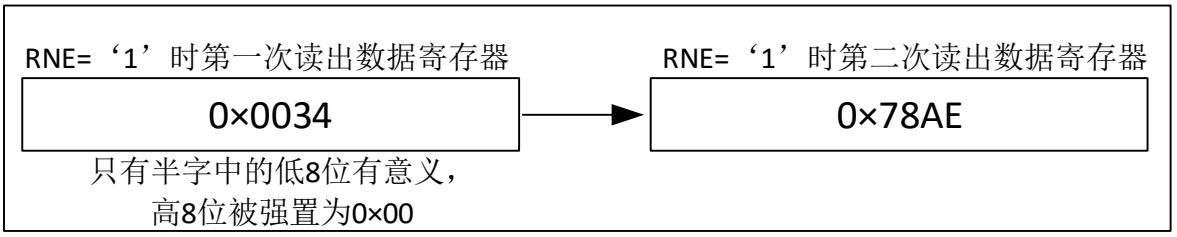
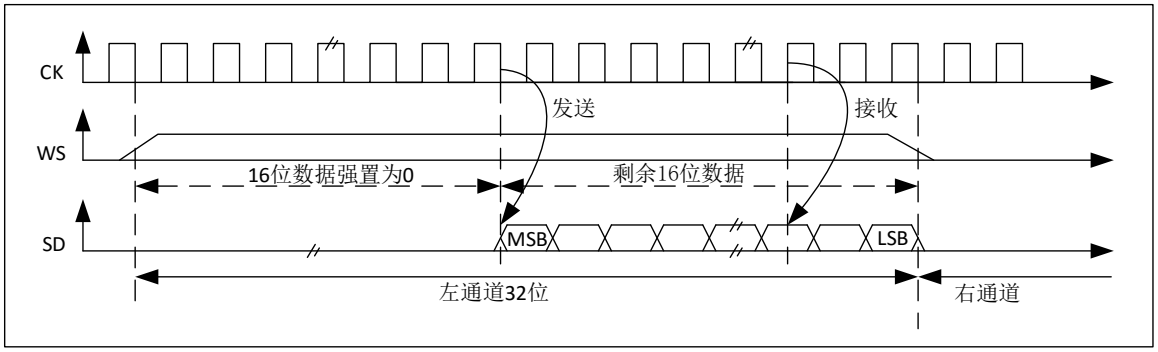


图 17-28 LSB对齐16位数据扩展到32位包帧，CPOL=0



在 I2S 配置阶段，如果选择将 16 位数据扩展到 32 声道帧，只需要访问一次寄存器 SPI_DT。此时，扩展到 32 位后的高半字（16 位 MSB）被硬件置为 0x0000。

如果待传输或者接收的数据是 0x76A3（扩展到 32 位是 0x000076A3），需要的操作如下图所示。

图 17-29 示例

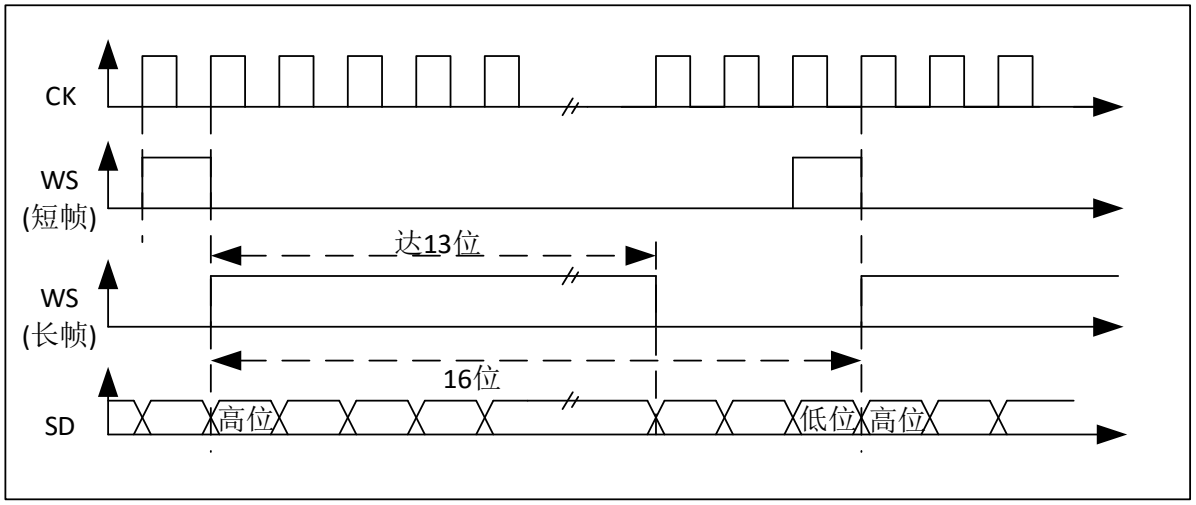


在发送时，如果 TE 为’1’，用户需要写入待发送的数据（即 0x76A3）。用来扩展到 32 位的 0x0000 部分由硬件首先发送出去，一旦有效数据开始从 SD 引脚送出，即发生下一次 TE 事件。在接收时，一旦接收到有效数据（而不是 0x0000 部分），即发生 RNE 事件。这样，在 2 次读和写之间有更多的时间，可以防止下溢或者上溢的情况发生。

PCM 标准

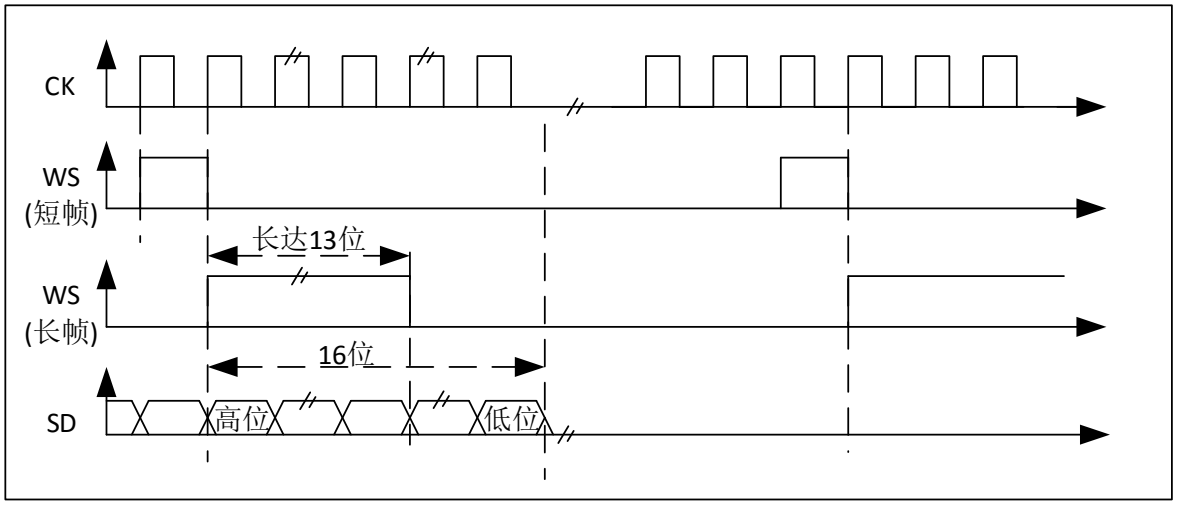
在 PCM 标准下，不存在声道选择的信息。PCM 标准有 2 种可用的帧结构，短帧或者长帧，可以通过设置寄存器 SPI_I2SCTRL 的 PCMSYNCSEL 位来选择。

图 17-30 PCM标准波形（16位）



对于长帧，主模式下，用来同步的 WS 信号有效的时间固定为 13 位。对于短帧，用来同步的 WS 信号长度只有 1 位。

图 17-31 PCM标准波形（16位扩展到32位包帧）



注意： 无论哪种模式（主或从）、哪种同步方式（短帧或长帧），连续的 2 帧数据之间和 2 个同步信号之间的时间差，（即使是从模式）需要通过设置 SPI_I2SCTRL 寄存器的 DLEN 位和 CHLEN 位来确定。

17.3.2.4 时钟发生器

I2S 的比特率即确定了在 I2S 数据线上的数据流和 I2S 的时钟信号频率。

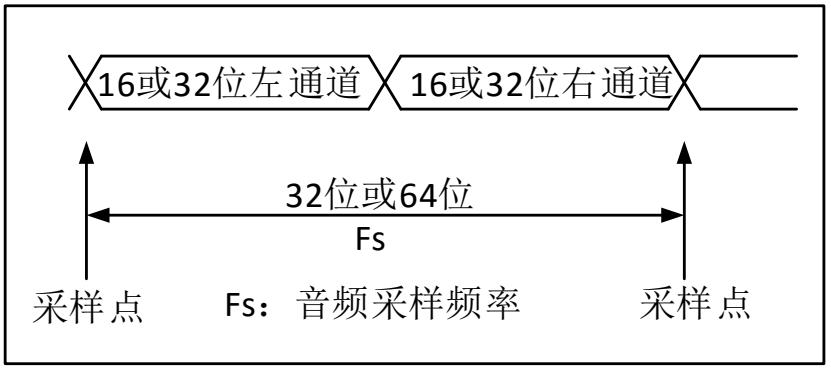
I2S 比特率=每个声道的比特数×声道数目×音频采样频率

对于一个具有左右声道和 16 位音频信号，I2S 比特率计算如下：

I2S 比特率=16×2×Fs

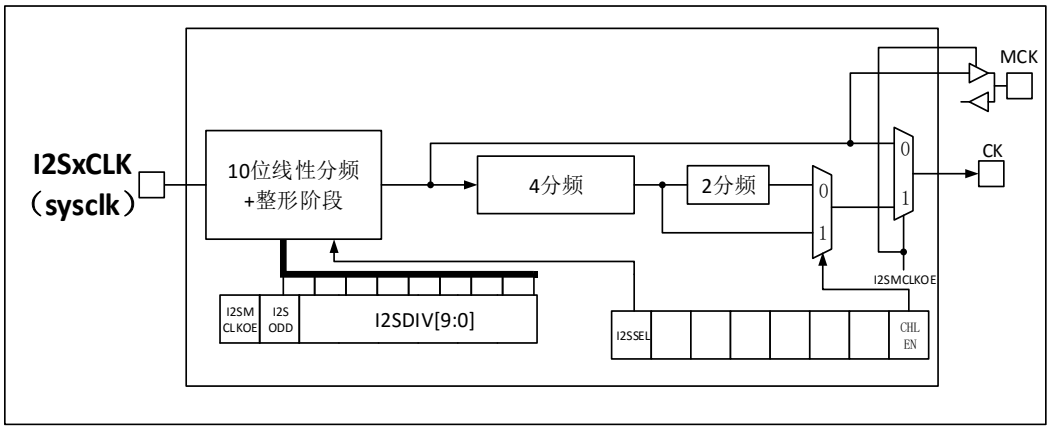
如果包长为 32 位，则有：I2S 比特率=32×2×Fs

图 17-32 音频采样频率定义



在主模式下，为了获得需要的音频频率，需要正确地对线性分频器进行设置。

图 17-33 I²S时钟发生器结构



上图中 I2SxCLK 的时钟源是系统时钟（即驱动 AHB 时钟的 HSI、HSE 或 PLL）。

音频的采样频率可以是 192KHz、96kHz、48kHz、44.1kHz、32kHz、22.05kHz、16kHz、11.025kHz 或者 8kHz（或任何此范围内的数值）。为了获得需要的频率，需按照以下公式设置线性分频器：

当需要生成主时钟时（寄存器 SPI_I2SCLKP 的 I2SMCLKOE 位为’1’）：

声道的帧长为 16 位时，Fs=I2SxCLK/[(16*2)* ((2*I2SDIV) + I2SODD)*8]

声道的帧长为 32 位时，Fs=I2SxCLK/[(32*2)* ((2*I2SDIV) + I2SODD)*4]

当关闭主时钟时 (I2SMCLKOE 位为’0’)：

声道的帧长为 16 位时，Fs=I2SxCLK/[(16*2) * ((2*I2SDIV) + I2SODD)]

声道的帧长为 32 位时，Fs=I2SxCLK/[(32*2) * ((2*I2SDIV) + I2SODD)]

下面 2 张表给出了不同时钟配置时，精确参数的例子。

注意： 可以使用其它配置以达到优化时钟精确度的目的。

表 17-2 使用系统时钟得到精确的音频频率

SysCLK (MHz)	MCLK	TargetFs (Hz)	16bit				32bit			
			I2sDIV	I2S_ODD	RealFs	Error	I2sDIV	I2S_ODD	RealFs	Error
240	NO	192000	19	1	192307	0.16%	10	0	1875000	2.34%
240	NO	96000	39	0	96153	0.16%	19	1	96153	0.16%
240	NO	48000	78	0	48076	0.16%	39	0	48076	0.16%
240	NO	44100	85	0	44117	0.04%	42	1	44117	0.04%
240	NO	32000	117	0	32051	0.16%	58	1	32051	0.16%
240	NO	22050	170	0	22058	0.04%	85	0	22058	0.04%
240	NO	16000	234	1	15991	0.05%	117	0	16025	0.16%
240	NO	11025	340	0	11029	0.04%	170	0	11029	0.04%
240	NO	8000	469	0	7995	0.05%	234	1	7995	0.05%
240	YES	192000	2	1	187500	2.34%	2	1	187500	2.34%
240	YES	96000	5	0	93750	2.34%	5	0	93750	2.34%
240	YES	48000	10	0	46875	2.34%	10	0	46875	2.34%
240	YES	44100	10	1	44642	1.23%	10	1	44642	1.23%
240	YES	32000	14	1	32327	1.02%	14	1	32327	1.02%
240	YES	22050	21	1	21802	1.12%	21	1	21802	1.12%
240	YES	16000	29	1	15889	0.68%	29	1	15889	0.68%
240	YES	11025	42	1	11029	0.04%	42	1	11029	0.04%
240	YES	8000	58	1	8012	0.16%	58	1	8012	0.16%
200	No	192000	16	1	189393.9	1.36%	8	0	195312.5	1.73%

200	No	96000	32	1	96153.85	0.16%	16	1	94696.97	1.36%
200	No	48000	65	0	48076.92	0.16%	32	1	48076.92	0.16%
200	No	44100	71	0	44014.08	0.19%	35	1	44014.08	0.19%
200	No	32000	97	1	32051.28	0.16%	49	0	31887.76	0.35%
200	No	22050	141	1	22084.81	0.16%	71	0	22007.04	0.19%
200	No	16000	195	1	15984.65	0.10%	97	1	16025.64	0.16%
200	No	11025	283	1	11022.93	0.02%	141	1	11042.4	0.16%
200	No	8000	390	1	8002.561	0.03%	195	1	7992.327	0.10%
200	Yes	192000	3	0	130208.3	32.18%	3	0	130208.3	32.18%
200	Yes	96000	4	0	97656.25	1.73%	4	0	97656.25	1.73%
200	Yes	48000	8	0	48828.13	1.73%	8	0	48828.13	1.73%
200	Yes	44100	9	0	43402.78	1.58%	9	0	43402.78	1.58%
200	Yes	32000	12	0	32552.08	1.73%	12	0	32552.08	1.73%
200	Yes	22050	17	1	22321.43	1.23%	17	1	22321.43	1.23%
200	Yes	16000	24	1	15943.88	0.35%	24	1	15943.88	0.35%
200	Yes	11025	35	1	11003.52	0.19%	35	1	11003.52	0.19%
200	Yes	8000	49	0	7971.939	0.35%	49	0	7971.939	0.35%
100	No	192000	8	0	195312.5	1.73%	4	0	195312.5	1.73%
100	No	96000	16	1	94696.97	1.36%	8	0	97656.25	1.73%
100	No	48000	32	1	48076.92	0.16%	16	1	47348.48	1.36%
100	No	44100	35	1	44014.08	0.19%	17	1	44642.86	1.23%
100	No	32000	49	0	31887.76	0.35%	24	1	31887.76	0.35%
100	No	22050	71	0	22007.04	0.19%	35	1	22007.04	0.19%
100	No	16000	97	1	16025.64	0.16%	49	0	15943.88	0.35%
100	No	11025	141	1	11042.4	0.16%	71	0	11003.52	0.19%
100	No	8000	195	1	7992.327	0.10%	97	1	8012.821	0.16%
100	Yes	96000	2	0	97656.25	1.73%	2	0	97656.25	1.73%
100	Yes	48000	4	0	48828.13	1.73%	4	0	48828.13	1.73%
100	Yes	44100	4	1	43402.78	1.58%	4	1	43402.78	1.58%
100	Yes	32000	6	0	32552.08	1.73%	6	0	32552.08	1.73%
100	Yes	22050	9	0	21701.39	1.58%	9	0	21701.39	1.58%
100	Yes	16000	12	0	16276.04	1.73%	12	0	16276.04	1.73%
100	Yes	11025	17	1	11160.71	1.23%	17	1	11160.71	1.23%
100	Yes	8000	24	1	7971.939	0.35%	24	1	7971.939	0.35%
72	No	192000	6	0	187500	2.34%	3	0	187500	2.34%
72	No	96000	11	1	97826.09	1.90%	6	0	93750	2.34%
72	No	48000	32	1	34615.38	27.88%	11	1	48913.04	1.90%
72	No	44100	25	1	44117.65	0.04%	13	0	43269.23	1.88%
72	No	32000	35	0	32142.86	0.45%	17	1	32142.86	0.45%
72	No	22050	51	0	22058.82	0.04%	25	1	22058.82	0.04%
72	No	16000	70	1	15957.45	0.27%	35	0	16071.43	0.45%
72	No	11025	102	0	11029.41	0.04%	51	0	11029.41	0.04%
72	No	8000	140	1	8007.117	0.09%	70	1	7978.723	0.27%
72	Yes	96000	2	0	70312.5	26.76%	2	0	70312.5	26.76%
72	Yes	48000	3	0	46875	2.34%	3	0	46875	2.34%
72	Yes	44100	3	0	46875	6.29%	3	0	46875	6.29%
72	Yes	32000	4	1	31250	2.34%	4	1	31250	2.34%
72	Yes	22050	6	1	21634.62	1.88%	6	1	21634.62	1.88%
72	Yes	16000	9	0	15625	2.34%	9	0	15625	2.34%
72	Yes	11025	13	0	10817.31	1.88%	13	0	10817.31	1.88%
72	Yes	8000	17	1	8035.714	0.45%	17	1	8035.714	0.45%

17.3.2.5 I²S主模式

设置 I²S 工作在主模式，串行时钟由引脚 CK 输出，字选信号由引脚 WS 产生。可以通过设置寄存器 SPI_I2SCLKP 的 I2SMCLKOE 位来选择输出或者不输出主时钟（MCLK）。

流程

1. 设置寄存器 SPI_I2SCLKP 的 I2SDIV[9：0]定义与音频采样频率相符的串行时钟波特率。同时也要定义寄存器 SPI_I2SCLKP 的 I2SODD 位。
2. 设置 CPOL 位定义通信用时钟在空闲时的电平状态。如果需要向外部的 DAC/ADC 音频器件提供主时钟 MCLK，将寄存器 SPI_I2SCLKP 的 I2SMCLKOE 位置为'1'。（按照不同的 MCLK 输出状态，计算 I2SDIV 和 I2SODD 的值，详见 17.3.2.4 节）。
3. 设置寄存器 SPI_I2SCTRL 的 I2SSEL 位为'1'激活 I²S 功能，设置 I2SAP[1：0]和 PCMSYNCSEL 位选择所用的 I²S 标准，设置 CHLEN 选择每个声道的数据位数。还要设置寄存器 SPI_I2SCTRL 的 I2SMOD[1：0]选择 I²S 主模式和方向（发送端还是接收端）。
4. 如果需要，可以通过设置寄存器 SPI_CTRL2 来打开所需的中断功能和 DMA 功能。
5. 必须将寄存器 SPI_I2SCTRL 的 I2SEN 位置为'1'。
6. 引脚 WS 和 CK 需要配置为输出模式。如果寄存器 SPI_I2SCLKP 的 I2SMCLKOE 位为'1'，引脚 MCLK 也要配置成输出模式。

发送流程

当写入 1 个半字（16 位）的数据至发送缓存，发送流程开始。假设第一个写入发送缓存的数据对应的是左声道数据。当数据从发送缓存移到移位寄存器时，标志位 TE 置'1'，这时，要把对应右声道的数据写入发送缓存。标志位 I2SCS 提示了目前待传输的数据对应哪个声道。标志位 I2SCS 的值在 TE 为'1'时更新，因此它在 TE 为'1'时有意义。

在先左声道后右声道的数据都传输完成后，才能被认为是一个完整的数据帧。不可以只传输部分数据帧，如仅有左声道的数据。

当发出第一位数据的同时，半字数据被并行地传送至 16 位移位寄存器，然后后面的位依次按高位在先的顺序从引脚 MOSI/SD 发出。每次数据从发送缓存移至移位寄存器时，标志位 TE 置为'1'，如果寄存器 SPI_CTRL2 的 TEIE 位为'1'，则产生中断。

写入数据的操作取决于所选择的 I²S 标准，详见 17.3.2.3 节。

为了保证连续的音频数据传输，建议在当前传输完成之前，对寄存器 SPI_DT 写入下一个要传输的数据。

当写入最后一个数据，等待 TE=1 及 BSY=0 后再清除 I2SEN 位关闭 I2S 功能。

接收流程

接收流程的配置步骤除了第 3 点外，与发送流程的一致（参见前述的"发送流程"），需要通过配置 I2SMOD[1：0]来选择主接收模式。无论何种数据和声道长度，音频数据总是以 16 位包的形式接收。即每次填满接收缓存后，标志位 RNE 置'1'，如果寄存器 SPI_CTRL2 的 RNEIE 位为'1'，则产生中断。根据配置的数据和声道长度，收到左声道或右声道的数据会需要 1 次或者 2 次把数据传送到接收缓存的过程。

对寄存器 SPI_DT 进行读操作即可清除 RNE 标志位。

每次接收以后即更新 I2SCS。它的值取决于 I²S 单元产生的 WS 信号。

读取数据的操作取决于所选择的 I²S 标准，详见 17.3.2.3 节。

如果前一个接收到的数据还没有被读取，又接收到新数据，即发生上溢，标志位 OVR 被置为'1'，如果寄存器 SPI_CTRL2 的 ERRIE 位为'1'，则产生中断，表示发生了错误。

若要关闭 I²S 功能，需要执行特别的操作，以保证 I²S 模块可以正常地完成传输周期而不会开始新的数据传输。操作过程与数据配置和通道长度、以及音频协议的模式相关：

- 16位数据扩展到32位通道长度（DLEN=00并且CHLEN=1），使用LSB（低位）对齐模式（I2SAP=10）

  a）等待倒数第二个（n-1）RNE=1；

  b）等待 17 个 I²S 时钟周期（使用软件延迟）；

  c）关闭 I²S（I2SEN=0）。

- 16位数据扩展到32位通道长度（DLEN=00并且CHLEN=1），使用MSB（高位）对齐、

I²S或PCM模式（分别为I2SAP=00，I2SAP=01或I2SAP=11）

a）等待最后一个 RNE=1；

b）等待 1 个 I2S 时钟周期（使用软件延迟）；

c）关闭 I2S（I2SEN=0）。

- 所有其它DLEN和CHLEN的组合，I2SAP选择的任意音频模式，使用下述方式关闭I²S：

a）等待倒数第二个（n-1）RNE=1；

b）等待一个 I2S 时钟周期（使用软件延迟）；

c）关闭 I2S（I2SEN=0）。

注意： 在传输期间 BSY 标志始终为低。

17.3.2.6 I²S从模式

在从模式下，I²S 可以设置成发送和接收模式。从模式的配置方式基本遵循和配置主模式一样的流程。在从模式下，不需要 I²S 接口提供时钟。时钟信号和 WS 信号都由外部主 I²S 设备提供，连接到相应的引脚上。因此用户无需配置时钟。

配置步骤列举如下：

1. 设置寄存器 SPI_I2SCTRL 的 I2SSEL 位激活 I²S 功能；设置 I2SAP[1：0]来选择所用的 I²S 标准；设置 DLEN[1：0]选择数据的比特数；设置 CHLEN 选择每个声道的数据位数。设置寄存器 SPI_I2SCTRL 的 I2SMOD[1：0]选择 I²S 从模式的数据方向（发送端还是接收端）。
2. 根据需要，设置寄存器 SPI_CTRL2 打开所需的中断功能和 DMA 功能。
3. 必须设置寄存器 SPI_I2SCTRL 的 I2SEN 位为'1'。

发送流程

当外部主设备发送时钟信号，并且当 NSS_WS 信号请求传输数据时，发送流程开始。必须先使能从设备，并且写入 I²S 数据寄存器之后，外部主设备才能开始通信。

对于 I²S 的 MSB 对齐和 LSB 对齐模式，第一个写入数据寄存器的数据项对应左声道的数据。当开始通信时，数据从发送缓冲器传送到移位寄存器，然后标志位 TE 置为'1'；这时，要把对应右声道的数据项写入 I²S 数据寄存器。

标志位 I2SCS 提示了目前待传输的数据对应哪个声道。与主模式的发送流程相比，在从模式中，I2SCS 取决于来自外部主 I²S 的 WS 信号。这意味着从 I²S 在接收到主端生成的时钟信号之前，就要准备好第一个要发送的数据。对于 I²S 的 MSB 对齐和 LSB 对齐模式，WS 信号为'1'表示先发送左声道。

注意： 设置 I2SEN 位为'1'的时间，应当比 CK 引脚上的主 I2S 时钟信号早至少 2 个 PCLK 时钟周期。

当发出第一位数据的时候，半字数据并行地通过 I²S 内部总线传输至 16 位移位寄存器，然后其它位依次按高位在先的顺序从引脚 MOSI/SD 发出。每次数据从发送缓冲器传送至移位寄存器时，标志位 TE 置'1'，如果寄存器 SPI_CTRL2 的 TEIE 位为'1'，则产生中断。

注意：在对发送缓冲器写入数据前，要确认标志位 TE 为'1'。

写入数据的操作取决于所选中的 I²S 标准，详见 17.3.2.3 节。为了保证连续的音频数据传输，建议在当前传输完成之前，对寄存器 SPI_DT 写入下一个要传输的数据。如果在代表下一个数据传输的第一个时钟边沿到达之前，新的数据仍然没有写入寄存器 SPI_DT，下溢标志位会置'1'，并可能产生中断；它指示软件发送数据错误。如果寄存器 SPI_CTRL2 的 ERRIE 位为'1'，在寄存器 SPI_STS 的标志位 UDR 为高时，就会产生中断。建议在这时关闭 I²S，然后重新从左声道开始发送数据。

当写入最后一个数据，等待 TE=1 及 BSY=0 后再清除 I2SEN 位关闭 I2S 功能。

接收流程

配置步骤除了第 1 点外，与发送流程一致。需要通过配置 I2SMOD[1：0]来选择主接收模式。无论何种数据和声道长度，音频数据总是以 16 位包的形式接收，即每次填满接收缓存，标志位 RNE 置'1'，如果寄存器 SPI_CTRL2 的 RNEIE 位为'1'，则产生中断。按照不同的数据和声道长度设置，收到左声道或者右声道数据会需要 1 次或者 2 次传输数据至接收缓冲器的过程。

每次接收到数据（将要从 SPI_DT 读出）以后即更新 I2SCS，它对应 I²S 单元产生的 WS 信号。

读取 SPI_DT 寄存器，将清除 RNE 位。

读取数据的操作取决于所选中的 I²S 标准，详见 17.3.2.3 节。
在还没有读出前一个接收到的数据，又接收到新数据时，即产生上溢，并设置标志位 OVR 为'1'；如果寄存器 SPI_CTRL2 的 ERRIE 位为'1'，则产生中断，指示发生了错误。
要关闭 I²S 功能时，需要在接收到最后一次 RNE=1 时将 I2SEN 位清'0'。

注意：外部主 I2S 器件需要有通过音频声道发送/接收 16 位或 32 位数据包的功能。

17.3.2.7 状态标志位

有 3 个状态标志位供用户监控 I²S 总线的状态。

忙标志位（BSY）

BSY 标志由硬件设置与清除（写入此位无效果），该标志位指示 I²S 通信层的状态。
该位为'1'时表明 I²S 通讯正在进行中，但有一个例外：主接收模式（I2SMOD=11）下，在接收期间 BSY 标志始终为低。
在软件要关闭 SPI 模块之前，可以使用 BSY 标志检测传输是否结束，这样可以避免破坏最后一次传输，因此需要严格按照下述过程执行。
当传输开始时，BSY 标志被置为'1'，除非 I²S 模块处于主接收模式。
下述情况时，该标志位被清除：

- 当传输结束时（除了主发送模式，这种模式下通信是连续的）；
- 当关闭I²S模块时。

当通信是连续的时候：

- 在主发送模式时，整个传输期间，BSY标志始终为高；
- 在从模式时，每个数据项传输之间，BSY标志在1个I2S时钟周期内变低。

注意：不要使用 BSY 标志处理每一个数据项的发送和接收，最好使用 TE 和 RNE 标志。

发送缓存空标志位（TE）

该标志位为'1'表示发送缓冲器为空，可以对发送缓冲器写入新的待发送数据。在发送缓冲器中已有数据时，标志位清'0'。在 I²S 被关闭时（I2SEN 位为'0'），该标志位也为'0'。

接收缓存非空标志位（RNE）

该标志位置'1'表示在接收缓存里有接收到的有效数据。在读取寄存器 SPI_DT 时，该位清'0'。

声道标志位（I2SCS）

在发送模式下，该标志位在 TE 为高时刷新，指示从 SD 引脚上发送的数据所在的声道。如果在从发送模式下发生了下溢错误，该标志位的值无效，在重新开始通讯前需要把 I²S 关闭再打开。
在接收模式下，该标志位在寄存器 SPI_DT 接收到数据时刷新，指示接收到的数据所在的声道。

注意：如果发生错误（如上溢 OVR），该标志位无意义，需要将 I2S 关闭再打开（同时，如果必要修改 I2S 的配置）。

在 PCM 标准下，无论短帧格式还是长帧格式，这个标志位都没有意义。
如果寄存器 SPI_STS 的标志位 OVR 或 UDR 为'1'，且寄存器 SPI_CTRL2 的 ERRIE 位为'1'，则会产生中断。（中断源已经被清除后）可以通过读寄存器 SPI_STS 来清除中断标志。

17.3.2.8 错误标志位

I²S 单元有 2 个错误标志位。

下溢标志位（UDR）

在从发送模式下，如果数据传输的第一个时钟边沿到达时，新的数据仍然没有写入 SPI_DT 寄存器，该标志位会被置'1'。在寄存器 SPI_I2SCTRL 的 I2SSEL 位置'1'后，该标志位才有效。如果寄存器 SPI_CTRL2 的 ERRIE 位为'1'，就会产生中断。
通过对寄存器 SPI_STS 进行读操作来清除该标志位。

上溢标志位（OVR）

如果还没有读出前一个接收到的数据时，又接收到新的数据，即产生上溢，该标志位置'1'，如果寄存器 SPI_CTRL2 的 ERRIE 位为'1'，则产生中断指示发生了错误。
这时，接收缓存的内容，不会刷新为从发送设备送来的新数据。对寄存器 SPI_DT 的读操作返回最后一个正确接收到的数据。其他所有在上溢发生后由发送设备发出的 16 位数据都会丢失。

通过先读寄存器 SPI_STS 再读寄存器 SPI_DT，来清除该标志位。

17.3.2.9 I²S中断

下表列举了全部 I²S 中断。

表 17-3 I²S中断请求

中断事件	事件标志位	使能标志位
发送缓冲器空标志位	TE	TEIE
接收缓冲器非空标志位	RNE	RNEIE
上溢标志位	OVR	ERRIE
下溢标志位	UDR	

17.3.2.10 DMA功能

DMA 的工作方式在 I²S 模式除了 CRC 功能不可用以外，与在 SPI 模式完全相同。

因为在 I²S 模式下没有数据传输保护系统。

17.4 SPI寄存器

可以用半字（16 位）或字（32 位）的方式操作这些外设寄存器。

表 17-4 SPI寄存器列表及其复位值

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	SPI_CTRL1	保留																BDMODE	BDOE	CCE	CTN	DFF16	RONLY	SWNSSEN	ISS	LSBEN	SPIEN	MCLKP[2：0]			MSTEN	CPOL	CPHA
	0x0000																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	SPI_CTRL2	保留																							MCLKP[3]	TEIE	RNEIE	ERRIE	保留		NSSOE	DMATEN	DMAREN
	0x0000																								0	0	0	0			0	0	0
0x08	SPI_STS	保留																								BSY	OVR	MODF	CERR	UDR	I2SCS	TE	RNE
	0x0002																									0	0	0	0	0	0	1	0
0x0C	SPI_DT	保留																DT[15：0]															
	0x0000																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	SPI_CPOLY	保留																CPOLY[15：0]															
	0x0007																	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1
0x14	SPI_RCRC	保留																RCRC[15：0]															
	0x0000																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	SPI_TCRC	保留																TCRC[15：0]															
	0x0000																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

| | | |

17.4.1 SPI控制寄存器1（SPI_CTRL1）（I²S模式下不使用）

地址偏移：0x00

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BDMODE	BDOE	CCE	CTN	DFF16	RONLY	SWNSSEN	ISS	LSBEN	SPIEN	MCLKP[2：0]			MSTEN	CPOL	CPHA
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 15	**BDMODE**：双向数据模式使能（Bidirectional data mode enable） 0：选择“双线单向”模式； 1：选择“单线双向”模式。 注：I²S 模式下不使用。
位 14	**BDOE**：双向模式下的输出使能（Output enable in bidirectional mode） 和 BDMODE 位一起决定在“单线双向”模式下数据的输出方向 0：输出禁止（只收模式）； 1：输出使能（只发模式）。 这个“单线”数据线在主设备端为 MOSI 引脚，在从设备端为 MISO 引脚。 注：I²S 模式下不使用。
位 13	**CCE**：硬件 CRC 校验使能（Hardware CRC calculation enable） 0：禁止 CRC 计算； 1：启动 CRC 计算。 注：只有在禁止 SPI 时（SPIEN=0），才能写该位，否则出错。该位只能在全双工模式下使用。 注：I²S 模式下不使用。
位 12	**CTN**：下一个发送 CRC（Transmit CRC next） 0：下一个发送的值来自发送缓冲区。 1：下一个发送的值来自发送 CRC 寄存器。 注：在 SPI_DT 寄存器写入最后一个数据后应马上设置该位。 注：I²S 模式下不使用。
位 11	**DFF16**：数据帧格式（Data frame format） 0：使用 8 位数据帧格式进行发送/接收； 1：使用 16 位数据帧格式进行发送/接收。 注：只有当 SPI 禁止（SPIEN=0）时，才能写该位，否则出错。 注：I²S 模式下不使用。
位 10	**RONLY**：只接收（Receive only） 该位和 BDMODE 位一起决定在“双线双向”模式下的传输方向。在多个从设备的配置中，在未被访问的从设备上该位被置 1，使得只有被访问的从设备有输出，从而不会造成数据线上数据冲突。 0：全双工（发送和接收）； 1：禁止输出（只接收模式）。 注：I²S 模式下不使用。
位 9	**SWNSSEN**：软件从设备管理（Software slave management） 当 SWNSSEN 被置位时，NSS 引脚上的电平由 ISS 位的值决定。 0：禁止软件从设备管理； 1：启用软件从设备管理。 注：I²S 模式下不使用。
位 8	**ISS**：内部从设备选择（Internal slave select） 该位只在 SWNSSEN 位为’1’时有意义。它决定了 NSS 上的电平，在 NSS 引脚上的 I/O 操作无效。 注：I²S 模式下不使用。

位 7	**LSBEN**：帧格式（Frame format） 0：先发送 MSB； 1：先发送 LSB。 注：当通信在进行时不能改变该位的值。 注：I^2S 模式下不使用。
位 6	**SPIEN**：SPI 使能（SPI enable） 0：禁止 SPI 设备； 1：开启 SPI 设备。 注：I^2S 模式下不使用。 注：当关闭 SPI 设备时，请按照第 17.3.1.8 节的过程操作。
位 5：3	**MCLKP[2：0]**：波特率控制（Baudrate control） MCLKP[3]位在 SPI_CTRL2 寄存器，MCLKP[3:0]： 0000：f_{PCLK}/2 0001：f_{PCLK}/4 0010：f_{PCLK}/8 0011：f_{PCLK}/16 0100：f_{PCLK}/32 0101：f_{PCLK}/64 0110：f_{PCLK}/128 0111：f_{PCLK}/256 1000：f_{PCLK}/512 1001：f_{PCLK}/1024 当通信正在进行的时候，不能修改这些位，只有在 SPIEN 关闭时才能修改。 注：I^2S 模式下不使用。
位 2	**MSTEN**：主设备选择（Master selection） 0：配置为从设备； 1：配置为主设备。 注：当通信正在进行的时候，不能修改该位。 注：I^2S 模式下不使用。
位 1	**CPOL**：时钟极性（Clock polarity） 0：空闲状态时，SCK 保持低电平； 1：空闲状态时，SCK 保持高电平。 注：当通信正在进行的时候，不能修改该位。 注：I^2S 模式下不使用。
位 0	**CPHA**：时钟相位（Clock phase） 0：数据采样从第一个时钟边沿开始； 1：数据采样从第二个时钟边沿开始。 注：当通信正在进行的时候，不能修改该位。 注：I^2S 模式下不使用。

17.4.2 SPI控制寄存器2（SPI_CTRL2）

地址偏移：0x04

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留							MCLKP[3]	TEIE	RNEIE	ERRIE	保留		NSSOE	DMATEN	DMAREN
res							rw	rw	rw	rw	res		rw	rw	rw

位 15：9	保留位，硬件强制为 0
位 8	**MCLKP[3]**：波特率控制（Baudrate control） 详见 MCLKP[2：0]在 SPI_CTRL1 寄存器。
位 7	**TEIE**：发送缓冲区空中断使能（Rx buffer empty interrupt enable） 0：禁止 TE 中断； 1：允许 TE 中断，当 TE 标志置位为'1'时产生中断请求。
位 6	**RNEIE**：接收缓冲区非空中断使能（RX buffer not empty interrupt enable） 0：禁止 RNE 中断； 1：允许 RNE 中断，当 RNE 标志置位时产生中断请求。

位 5	**ERRIE**：错误中断使能（Error interrupt enable） 当错误（CERR、OVR、MODF）产生时，该位控制是否产生中断 0：禁止错误中断； 1：允许错误中断。
位 4：3	保留位，硬件强制为 0。
位 2	**NSSOE**：SS 输出使能（SS output enable） 0：禁止在主模式下 SS 输出，该设备可以工作在多主设备模式； 1：设备开启时，开启主模式下 SS 输出，该设备不能工作在多主设备模式。 注：I^2S 模式下不使用。
位 1	**DMATEN**：发送缓冲区 DMA 使能（DMA Tx enable） 当该位被设置时，TE 标志一旦被置位就发出 DMA 请求 0：禁止发送缓冲区 DMA； 1：启动发送缓冲区 DMA。
位 0	**DMAREN**：接收缓冲区 DMA 使能（DMA Rx enable） 当该位被设置时，RNE 标志一旦被置位就发出 DMA 请求 0：禁止接收缓冲区 DMA； 1：启动接收缓冲区 DMA。

17.4.3 SPI状态寄存器（SPI_STS）

地址偏移：0x08

复位值：0x0002

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留								BSY	OVR	MODF	CERR	UDR	I2SCS	TE	RNE
res								r	r	r	r_w0	r	r	r	r

位 15：8	保留位，硬件强制为 0
位 7	**BSY**：忙标志（Busy flag） 0：SPI 不忙； 1：SPI 正忙于通信，或者发送缓冲非空。 该位由硬件置位或者复位。 注：使用这个标志时需要特别注意，详见第 17.3.1.7 节和第 17.3.1.8 节。
位 6	**OVR**：溢出标志（Over run flag） 0：没有出现溢出错误； 1：出现溢出错误。 该位由硬件置位，由软件序列复位。 关于软件序列的详细信息，参考 17.3.2.8 节。
位 5	**MODF**：模式错误（Mode fault） 0：没有出现模式错误； 1：出现模式错误。 该位由硬件置位，由软件序列复位。 关于软件序列的详细信息，参考 17.3.1.10 节。 注：I^2S 模式下不使用。
位 4	**CERR**：CRC 错误标志（CRC error flag） 0：收到的 CRC 值和 SPI_RCRC 寄存器中的值匹配； 1：收到的 CRC 值和 SPI_RCRC 寄存器中的值不匹配。 该位由硬件置位，由软件写'0'而复位。 注：I^2S 模式下不使用。

位 3	**UDR**：下溢标志位（Under run flag） 0：未发生下溢； 1：发生下溢。 该标志位由硬件置'1'，由一个软件序列清'0'，详见 17.3.2.8 节。 注：在 SPI 模式下不使用。
位 2	**I2SCS**：声道（Channel side） 0：需要传输或者接收左声道； 1：需要传输或者接收右声道。 注：在 SPI 模式下不使用。在 PCM 模式下无意义。
位 1	**TE**：发送缓冲为空（Transmit buffer empty） 0：发送缓冲非空； 1：发送缓冲为空。
位 0	**RNE**：接收缓冲非空（Receive buffer not empty） 0：接收缓冲为空； 1：接收缓冲非空。

17.4.4 SPI数据寄存器（SPI_DT）

地址偏移：0x0C

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DT[15：0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 15：0	DT[15：0]：数据寄存器（Dataregister） 待发送或者已经收到的数据 数据寄存器对应两个缓冲区：一个用于写（发送缓冲）；另外一个用于读（接收缓冲）。 写操作将数据写到发送缓冲区；读操作将返回接收缓冲区里的数据。 对 SPI 模式的注释：根据 SPI_CTRL1 的 DFF16 位对数据帧格式的选择，数据的发送和接收可以是 8 位或者 16 位的。为保证正确的操作，需要在启用 SPI 之前就确定好数据帧格式。 对于 8 位的数据，缓冲器是 8 位的，发送和接收时只会用到 SPI_DT[7：0]。在接收时，SPI_DT[15：8]被强制为 0。 对于 16 位的数据，缓冲器是 16 位的，发送和接收时会用到整个数据寄存器，即 SPI_DT[15：0]

17.4.5 SPICRC多项式寄存器（SPI_CPOLY）（I²S模式下不使用）

地址偏移：0x10

复位值：0x0007

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CPOLY[15：0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 15：0	CPOLY[15：0]：CRC 多项式寄存器（CRC polynomial egister） 该寄存器包含了 CRC 计算时用到的多项式。其复位值为 0x0007，根据应用可以设置其他数值。 注：在 I2S 模式下不使用。

17.4.6 SPIRxCRC寄存器（SPI_RCRC）（I²S模式下不使用）

地址偏移：0x14

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RCRC[15：0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

位 15：0	**RCRC[15：0]**：接收 CRC 寄存器 在启用 CRC 计算时，RCRC[15：0]中包含了依据收到的字节计算的 CRC 数值。当在 SPI_CTRL1 的 CCE 位写入'1'时，该寄存器被复位。CRC 计算使用 SPI_CPOLY 中的多项式。当数据帧格式被设置为 8 位时，仅低 8 位参与计算，并且按照 CRC8 的方法进行；当数据帧格式为 16 位时，寄存器中的所有 16 位都参与计算，并且按照 CRC16 的标准。 注：当 BSY 标志为'1'时读该寄存器，将可能读到不正确的数值。 注：在 I²S 模式下不使用。

17.4.7 SPITxCRC寄存器（SPI_TCRC）

地址偏移：0x18

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TCRC[15：0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

位 15：0	**TCRC[15：0]**：发送 CRC 寄存器 在启用 CRC 计算时，TCRC[15：0]中包含了依据将要发送的字节计算的 CRC 数值。当在 SPI_CTRL1 中的 CCE 位写入'1'时，该寄存器被复位。CRC 计算使用 SPI_CPOLY 中的多项式。 当数据帧格式被设置为 8 位时，仅低 8 位参与计算，并且按照 CRC8 的方法进行；当数据帧格式为 16 位时，寄存器中的所有 16 个位都参与计算，并且按照 CRC16 的标准。 注：当 BSY 标志为'1'时读该寄存器，将可能读到不正确的数值。 注：在 I²S 模式下不使用。

17.4.8 SPI_I2S配置寄存器（SPI_I2SCTRL）

地址偏移：0x1C

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留				I2SSEL	I2SEN	I2SMOD[1：0]		PCMSYNCSEL	保留	I2SAP[1：0]		CPOL	DLEN[1：0]		CHLEN
res				rw	rw	rw	rw	rw	res	rw	rw	rw	rw	rw	rw

位 15：12	保留位，硬件强制为 0
位 11	**I2SSEL**：I²S 模式选择（I²S mode selection） 0：选择 SPI 模式； 1：选择 I²S 模式。 注：该位只有在关闭了 SPI 或者 I²S 时才能设置。

位 10	**I2SEN**：I^2S 使能（I^2S enable） 0：关闭 I^2S； 1：I^2S 使能。 注：在 SPI 模式下不使用。
位 9：8	I2SMOD[1：0]：I2S 模式设置（I2S configuration mode） 00：从设备发送； 01：从设备接收； 10：主设备发送； 11：主设备接收。 注：该位只有在关闭了 I2S 时才能设置。 在 SPI 模式下不使用。
位 7	**PCMSYNCSEL**：PCM 帧同步（PCM frame synchronization） 0：短帧同步； 1：长帧同步。 注：该位只在 I2SAP=11（使用 PCM 标准）时有意义。 在 SPI 模式下不使用。
位 6	保留位，硬件强制为 0。
位 5：4	**I2SAP**[1：0]：I2S 标准选择（I2S standard selection） 00：I2S 飞利浦标准； 01：高字节对齐标准（左对齐）； 10：低字节对齐标准（右对齐）； 11：PCM 标准。关于 I2S 标准的细节，详见 17.3.2.3 节。 注：为了正确操作，只有在关闭了 I2S 时才能设置该位。 在 SPI 模式下不使用。
位 3	**CPOL**：静止态时钟极性（Steady state clock polarity） 0：I2S 时钟静止态为低电平； 1：I2S 时钟静止态为高电平。 注：为了正确操作，该位只有在关闭了 I2S 时才能设置。 在 SPI 模式下不使用。
位 2：1	**DLEN**[1：0]：待传输数据长度（Data length to be transferred） 00：16 位数据长度； 01：24 位数据长度； 10：32 位数据长度； 11：不允许。 注：为了正确操作，该位只有在关闭了 I2S 时才能设置。 在 SPI 模式下不使用。
位 0	**CHLEN**：声道长度（每个音频声道的数据位数）（Channel length（number of bits per audio channel）） 0：16 位宽； 1：32 位宽。 只有在 DLEN=00 时该位的写操作才有意义，否则声道长度都由硬件固定为 32 位。 注：为了正确操作，该位只有在关闭了 I2S 时才能设置。 在 SPI 模式下不使用。

17.4.9 SPI_I2S预分频寄存器（SPI_I2SCLKP）

地址偏移：0x20

复位值：0x0002

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留				I2SDIV[9：8]		I2SMCLKOE	I2SODD	I2SDIV[7：0]							
res				rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 15：12	保留位，硬件强制为 0
位 12：11	**I2SDIV**[9：8]：I2S 线性预分频（I2S linear prescaler） 描述见 I2SDIV[7：0].
位 9	**I2SMCLKOE**：主设备时钟输出使能（Master clock output enable） 0：关闭主设备时钟输出； 1：主设备时钟输出使能。 注：为了正确操作，该位只有在关闭了 I2S 时才能设置。仅在 I2S 主设备模式下使用该位。在 SPI 模式下不使用。
位 8	**I2SODD**：奇系数预分频（Odd factor for the prescaler） 0：实际分频系数=I2SDIV*2； 1：实际分频系数=（I2SDIV*2）+1。 参见 17.3.2.4 节。 注：为了正确操作，该位只有在关闭了 I2S 时才能设置。仅在 I2S 主设备模式下使用该位。在 SPI 模式下不使用。
位 7：0	**I2SDIV**[7：0]：I2S 线性预分频（I2Slinearprescaler） I2SDIV[9：8]设置在 bit12：11。 禁止设置 I2SDIV[9：0]=0 或者 I2SDIV[9：0]=1 参见 17.3.2.4 节。 注：为了正确操作，该位只有在关闭了 I2S 时才能设置。仅在 I2S 主设备模式下使用该位。在 SPI 模式下不使用。

18 CAN总线控制器

18.1 简介

bxCAN 是基本扩展 CAN（Basic Extended CAN）的缩写，它支持 CAN 协议 2.0A 和 2.0B。它的设计目标是以最小的 CPU 负荷来高效处理大量收到的报文。它也支持报文发送的优先级要求（优先级特性可软件配置）。

对于安全紧要的应用，bxCAN 提供所有支持时间触发通信模式所需的硬件功能。

18.2 主要特点

- 支持 CAN 协议 2.0A 和 2.0B 主动模式
- 波特率最高可达 1 兆位/秒
- 支持时间触发通信功能

发送

- 3 个发送邮箱
- 发送报文的优先级特性可软件配置
- 记录发送 SOF 时刻的时间戳

接收

- 3 级深度的 2 个接收 FIFO
- 可变的过滤器组：
  - 有 14 个过滤器组
- 标识符列表
- FIFO 溢出处理方式可配置
- 记录接收 SOF 时刻的时间戳

时间触发通信模式

- 禁止自动重传模式
- 16 位自由运行定时器
- 可在最后 2 个数据字节发送时间戳

管理

- 中断可屏蔽
- 邮箱占用单独 1 块地址空间，便于提高软件效率

CAN 的 SRAM 存储器

- CAN 负责管理 512 字节的 SRAM 存储器

18.3 功能描述

18.3.1 CAN整体功能描述

在当今的 CAN 应用中，CAN 网络的节点在不断增加，并且多个 CAN 常常通过网关连接起来，因此整个 CAN 网中的报文数量（每个节点都需要处理）急剧增加。除了应用层报文外，网络管理和诊断报文也被引入。

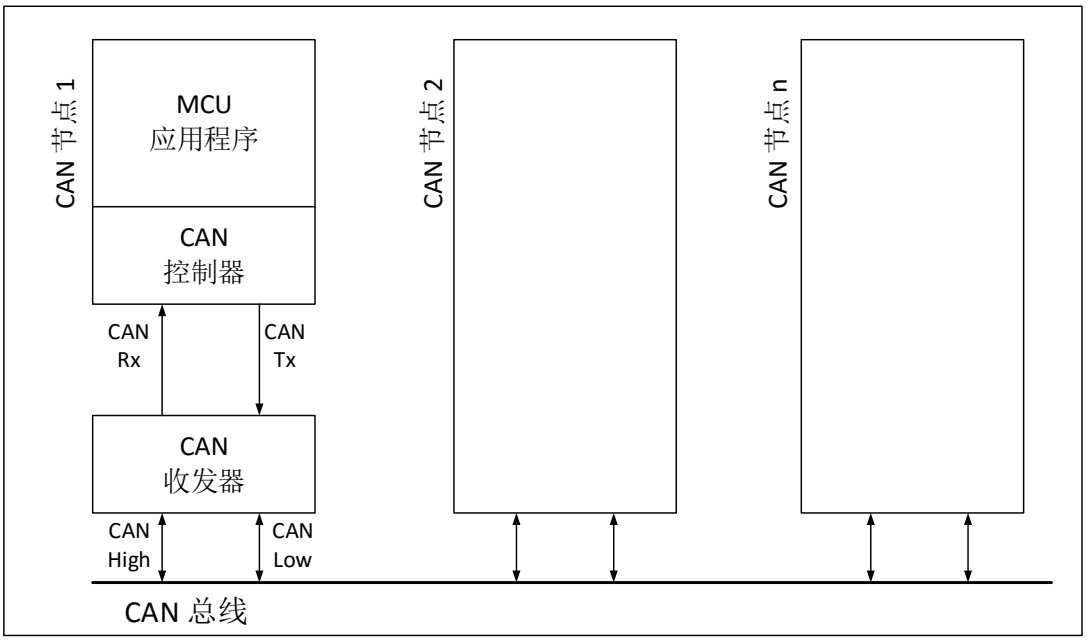
- 需要一个增强的过滤机制来处理各种类型的报文

  此外，由于应用层任务需要占用更多 CPU 时间，为满足实时性的要求，需减少接收报文的处理时间。

- 采用接收 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。

构筑在底层 CAN 驱动程序上的高层协议软件，要求跟 CAN 控制器之间有高效的接口。

图 18-1 CAN网拓扑结构



bxCAN 模块可以完全自动地接收和发送 CAN 报文；且完全支持标准标识符（11 位）和扩展标识符（29 位）。

应用程序通过这些寄存器，可以：

- 配置 CAN 参数，如波特率
- 请求发送报文
- 处理报文接收
- 管理中断
- 获取诊断信息

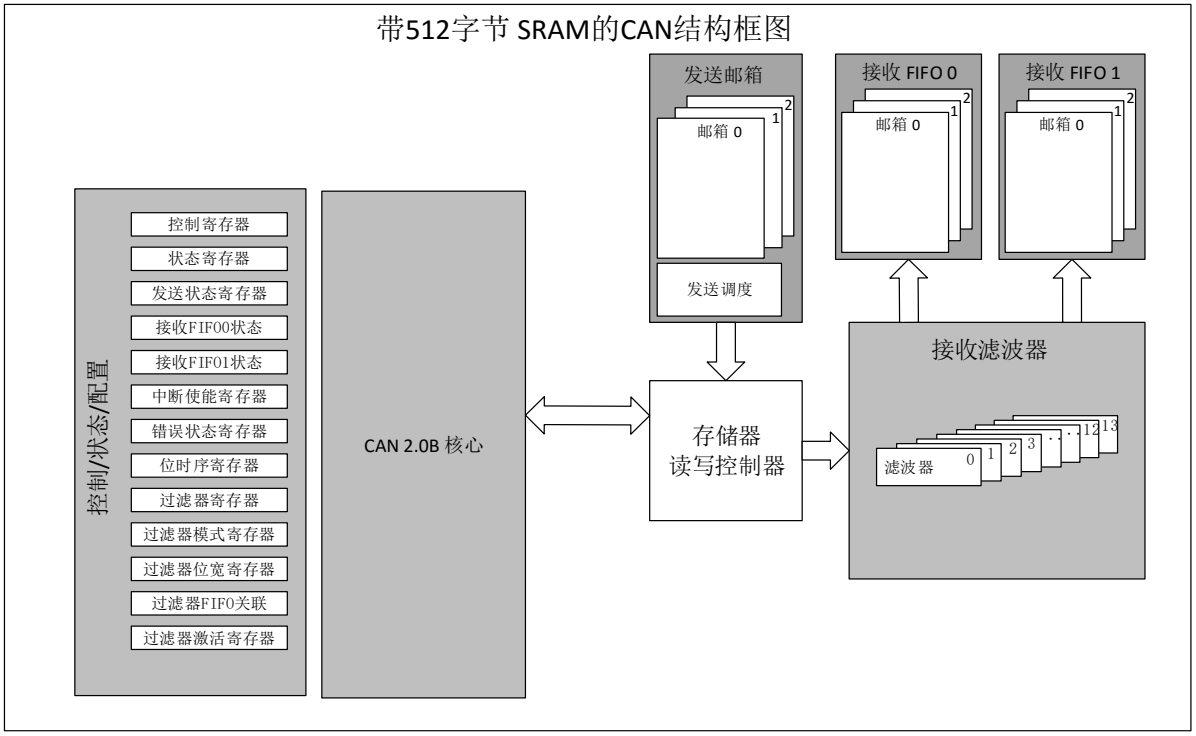
共有 3 个发送邮箱供软件来发送报文。发送调度器根据优先级决定哪个邮箱的报文先被发送。

bxCAN 提供 14 个位宽可变/可配置的标识符过滤器组，软件通过对它们编程，可以在收到的报文中选择它需要的报文，丢掉其它不需要的报文。

接收 FIFO

共有 2 个接收 FIFO，每个 FIFO 都可以存放 3 个完整的报文。它们完全由硬件来管理。

图 18-2 CAN框图



18.3.2 工作模式

bxCAN 有 3 个主要的工作模式：初始化、正常和睡眠模式。在硬件复位后，bxCAN 工作在睡眠模式以节省电能，同时 CANTX 引脚的内部上拉电阻被激活。软件通过对 CAN_MCTRL 寄存器的 INRQ 或 SLP 位置'1'，可以请求 bxCAN 进入初始化或睡眠模式。一旦进入了初始化或睡眠模式，bxCAN 就对 CAN_MSTS 寄存器的 IAK 或 SAK 位置'1'来进行确认，同时内部上拉电阻被禁用。当 IAK 和 SAK 位都为'0'时，bxCAN 就处于正常模式。在进入正常模式前，bxCAN 必须跟 CAN 总线取得同步；为取得同步，bxCAN 要等待 CAN 总线达到空闲状态，即在 CANRX 引脚上监测到 11 个连续的隐性位。

18.3.2.1 初始化模式

软件初始化应该在硬件处于初始化模式时进行。设置 CAN_MCTRL 寄存器的 INRQ 位为'1'，请求 bxCAN 进入初始化模式，然后等待硬件对 CAN_MSTS 寄存器的 IAK 位置'1'来进行确认。清除 CAN_MCTRL 寄存器的 INRQ 位为'0'，请求 bxCAN 退出初始化模式，当硬件对 CAN_MSTS 寄存器的 IAK 位清'0'就确认了初始化模式的退出。

当 bxCAN 处于初始化模式时，禁止报文的接收和发送，并且 CANTX 引脚输出隐性位（高电平）。初始化模式的进入，不会改变配置寄存器。

软件对 bxCAN 的初始化，至少包括位时间特性（CAN_BTMG）和控制（CAN_MCTRL）这 2 个寄存器。在对 bxCAN 的过滤器组（模式、位宽、FIFO 关联、激活和过滤器值）进行初始化前，软件要对 CAN_FM 寄存器的 FINT 位设置'1'。对过滤器的初始化可以在非初始化模式下进行。

注意：当 FINT=1 时，报文的接收被禁止。

可以先对过滤器激活位清'0'（在 CAN_FA1 中），然后修改相应过滤器的值。

如果过滤器组没有使用，那么就应该让它处于非激活状态（保持其 FEN 位为清'0'状态）。

18.3.2.2 正常模式

在初始化完成后，软件应该让硬件进入正常模式，以便正常接收和发送报文。软件可以通过对 CAN_MCTRL 寄存器的 INRQ 位清'0'，来请求从初始化模式进入正常模式，然后要等待硬件对 CAN_MSTS 寄存器的 IAK 位清'0'的确认。在跟 CAN 总线取得同步，即在 CANRX 引脚上监测到 11 个连续的隐性位（等效于总线空闲）后，bxCAN 才能正常接收和发送报文。

不需要在初始化模式下进行过滤器初值的设置，但必须在它处在非激活状态下完成（相应的FEN位为0）。而过滤器的位宽和模式的设置，则必须在初始化模式中进入正常模式前完成。

18.3.2.3 睡眠模式（低功耗）

bxCAN 可工作在低功耗的睡眠模式。软件通过对 CAN_MCTRL 寄存器的 SLP 位置'1'，来请求进入这一模式。在该模式下，bxCAN 的时钟停止了，但软件仍然可以访问邮箱寄存器。

当 bxCAN 处于睡眠模式，软件必须对 CAN_MCTRL 寄存器的 INRQ 位置'1' 并且同时对 SLP 位清'0'，才能进入初始化模式。

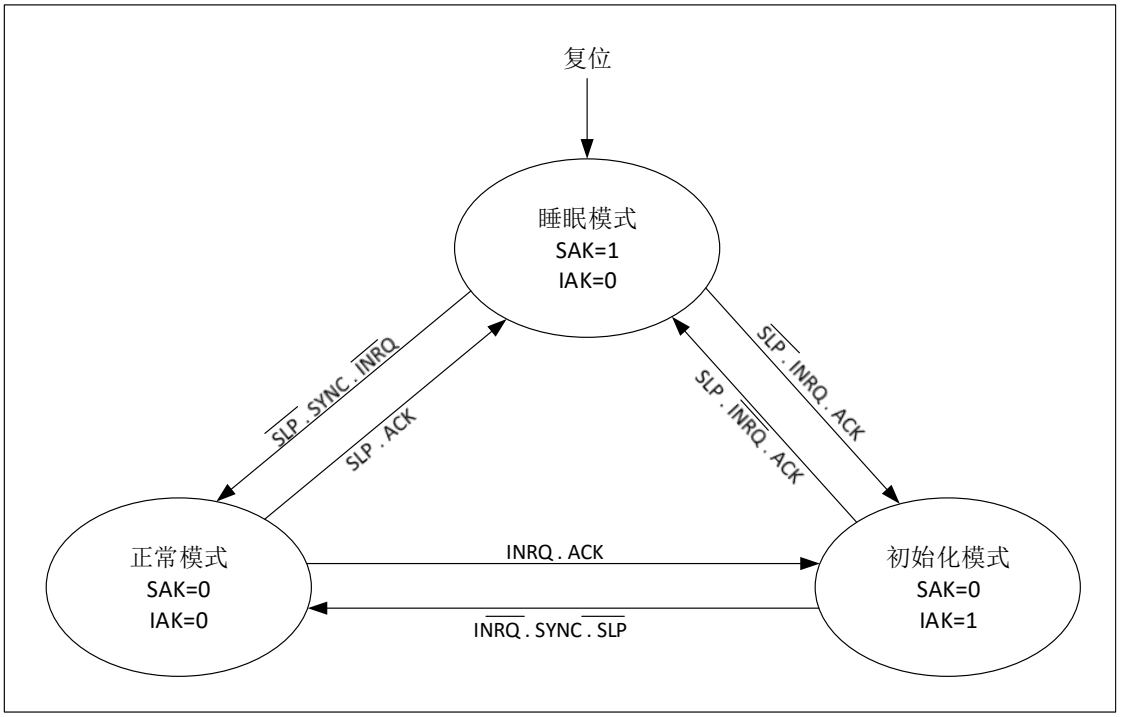
有 2 种方式可以唤醒（退出睡眠模式）bxCAN：通过软件对 SLP 位清'1'，或硬件检测到 CAN 总线的活动。

如果 CAN_MCTRL 寄存器的 AWU 位为'1'，一旦检测到 CAN 总线的活动，硬件就自动对 SLP 位清'0'来唤醒 bxCAN。如果 CAN_MCTRL 寄存器的 AWU 位为'0'，软件必须在唤醒中断里对 SLP 位清'0'才能退出睡眠状态。

注意： *如果唤醒中断被允许（CAN_INTEN 寄存器的 WKIE 位为'1'），那么一旦检测到 CAN 总线活动就会产生唤醒中断，而不管硬件是否会自动唤醒 bxCAN。*

在对 SLP 位清'0'后，睡眠模式的退出必须与 CAN 总线同步，请参考图 18-3：bxCAN 工作模式。当硬件对 SAK 位清'0'时，就确认了睡眠模式的退出。

图 18-3 bxCAN工作模式



注意： *1.ACK = 硬件响应睡眠或初始化请求,而对 CAN_MSTS 寄存器的 IAK 或 SAK 位置 1 的状态*

2.SYNC = bxCAN 等待 CAN 总线变为空闲的状态，即在 CANRX 引脚上检测到连续的 11 个隐性位。

18.3.3 测试模式

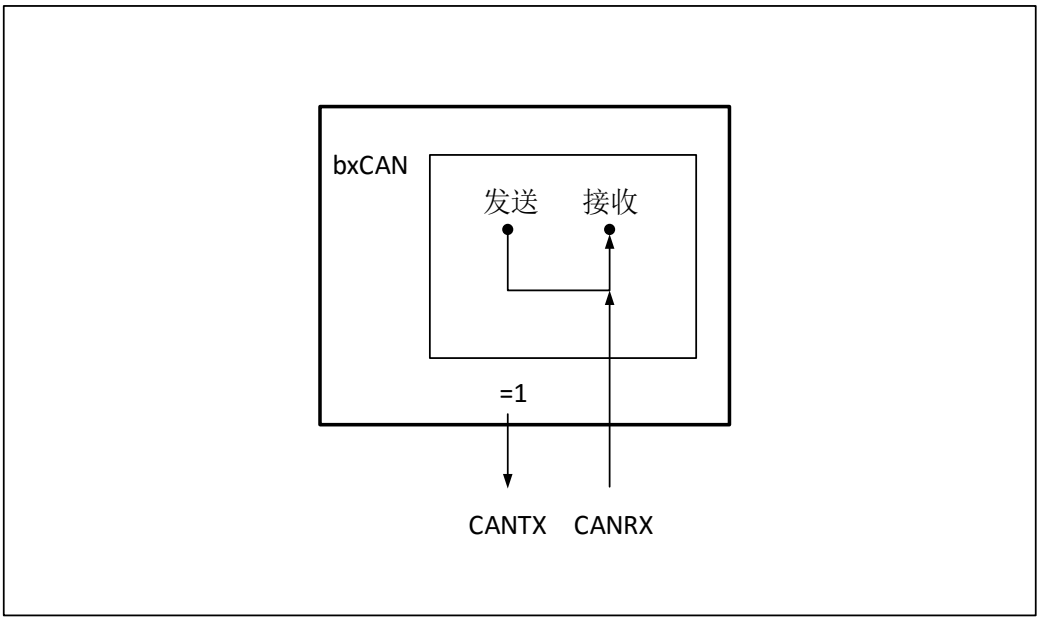
通过对 CAN_BTMG 寄存器的 SIL 和/或 LBK 位置'1'，来选择一种测试模式。只能在初始化模式下，修改这 2 位。在选择了一种测试模式后，软件需要对 CAN_MCTRL 寄存器的 INRQ 位清'0'，来真正进入测试模式。

18.3.3.1 静默模式

通过对 CAN_BTMG 寄存器的 SIL 位置'1'，来选择静默模式。在静默模式下，bxCAN 可以正常地接收数

据帧和远程帧，但只能发出隐性位，而不能真正发送报文。如果 bxCAN 需要发出显性位（确认位、过载标志、主动错误标志），那么这样的显性位在内部被接回来从而可以被 CAN 内核检测到，同时 CAN 总线不会受到影响而仍然维持在隐性位状态。因此，静默模式通常用于分析 CAN 总线的活动，而不会对总线造成影响－显性位（确认位、错误帧）不会真正发送到总线上。

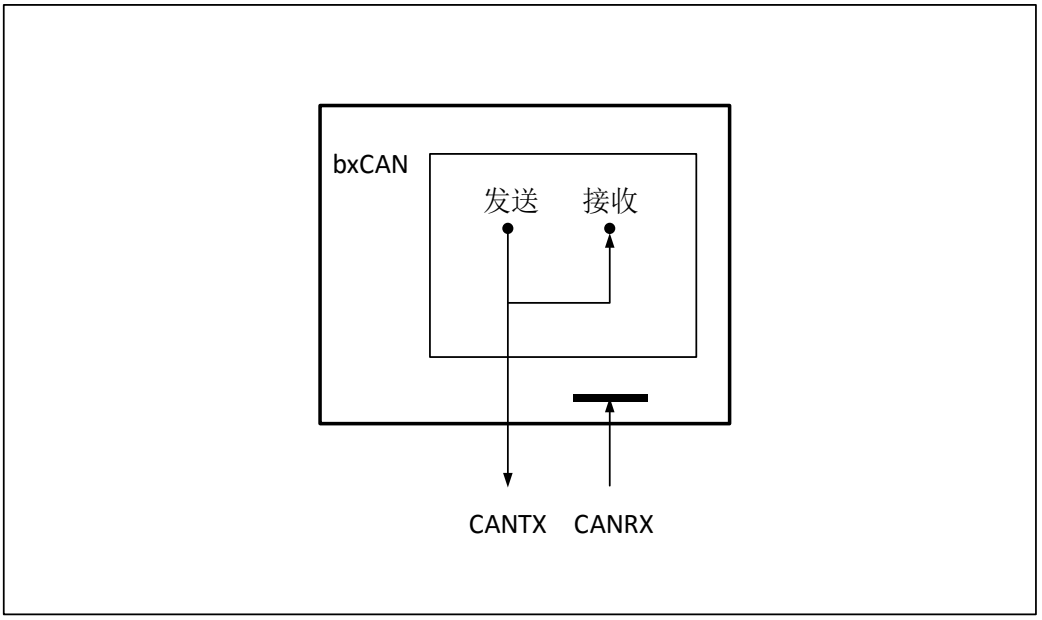
图 18-4 bxCAN工作在静默模式



18.3.3.2 环回模式

通过对 CAN_BTMG 寄存器的 LBK 位置'1'，来选择环回模式。在环回模式下，bxCAN 把发送的报文当作接收的报文并保存（如果可以通过接收过滤）在接收邮箱里。

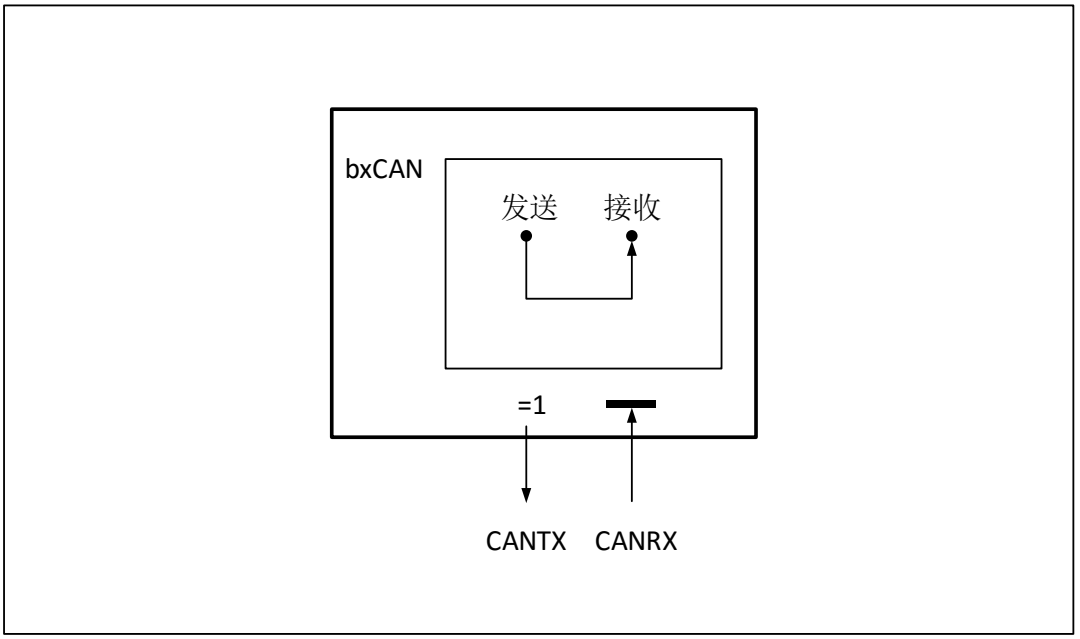
图 18-5 bxCAN工作在环回模式



环回模式可用于自测试。为了避免外部的影响，在环回模式下 CAN 内核忽略确认错误（在数据/远程帧的确认位时刻，不检测是否有显性位）。在环回模式下，bxCAN 在内部把 Tx 输出回馈到 Rx 输入上，而完全忽略 CANRX 引脚的实际状态。发送的报文可以在 CANTX 引脚上检测到。

18.3.3.3 环回静默模式

图 18-6 bxCAN工作在环回静默模式



通过对 CAN_BTMG 寄存器的 LBK 和 SIL 位同时置'1'，可以选择环回静默模式。该模式可用于 "热自测试"，即可以像环回模式那样测试 bxCAN，但却不会影响 CANTX 和 CANRX 所连接的整个 CAN 系统。在环回静默模式下，CANRX 引脚与 CAN 总线断开，同时 CANTX 引脚被驱动到隐性位状态。

18.3.4 AT32F403A/407系列处于调试模式时

当微控制器处于调试模式时，Cortex™-M4F 核心处于暂停状态，依据下述配置位的状态，bxCAN 可以继续正常工作或停止工作：

- 调试（DBG）模块中 CAN1 的 DBG_CAN1_STOP 位。
- CAN_MCTRL 中的 DBF 位。

18.3.5 发送处理

发送报文的流程为：应用程序选择 1 个空置的发送邮箱；设置标识符，数据长度和待发送数据；然后对 CAN_TMIx 寄存器的 TRQ 位置'1'，来请求发送。TRQ 位置'1'后，邮箱就不再是空邮箱；而一旦邮箱不再为空置，软件对邮箱寄存器就不再有写的权限。TRQ 位置 1 后，邮箱马上进入挂号状态，并等待成为最高优先级的邮箱，参见发送优先级。一旦邮箱成为最高优先级的邮箱，其状态就变为预定发送状态。一旦 CAN 总线进入空闲状态，预定发送邮箱中的报文就马上被发送（进入发送状态）。一旦邮箱中的报文被成功发送后，它马上变为空置邮箱；硬件相应地对 CAN_TSTS 寄存器的 RQC 和 TOK 位置 1，来表明一次成功发送。

如果发送失败，由于仲裁引起的就对 CAN_TSTS 寄存器的 ALS 位置'1'，由于发送错误引起的就对 TER 位置'1'。

发送优先级

1. 由标识符决定

   当有超过 1 个发送邮箱在挂号时，发送顺序由邮箱中报文的标识符决定。根据 CAN 协议，标识符数值最低的报文具有最高的优先级。如果标识符的值相等，那么邮箱号小的报文先被发送。

2. 由发送请求次序决定

   通过对 CAN_MCTRL 寄存器的 TFP 位置'1'，可以把发送邮箱配置为发送 FIFO。在该模式下，发送的优先级由发送请求次序决定。该模式对分段发送很有用。

中止

通过对 CAN_TSTS 寄存器的 ARQ 位置'1'，可以中止发送请求。邮箱如果处于挂号或预定状态， 发送请

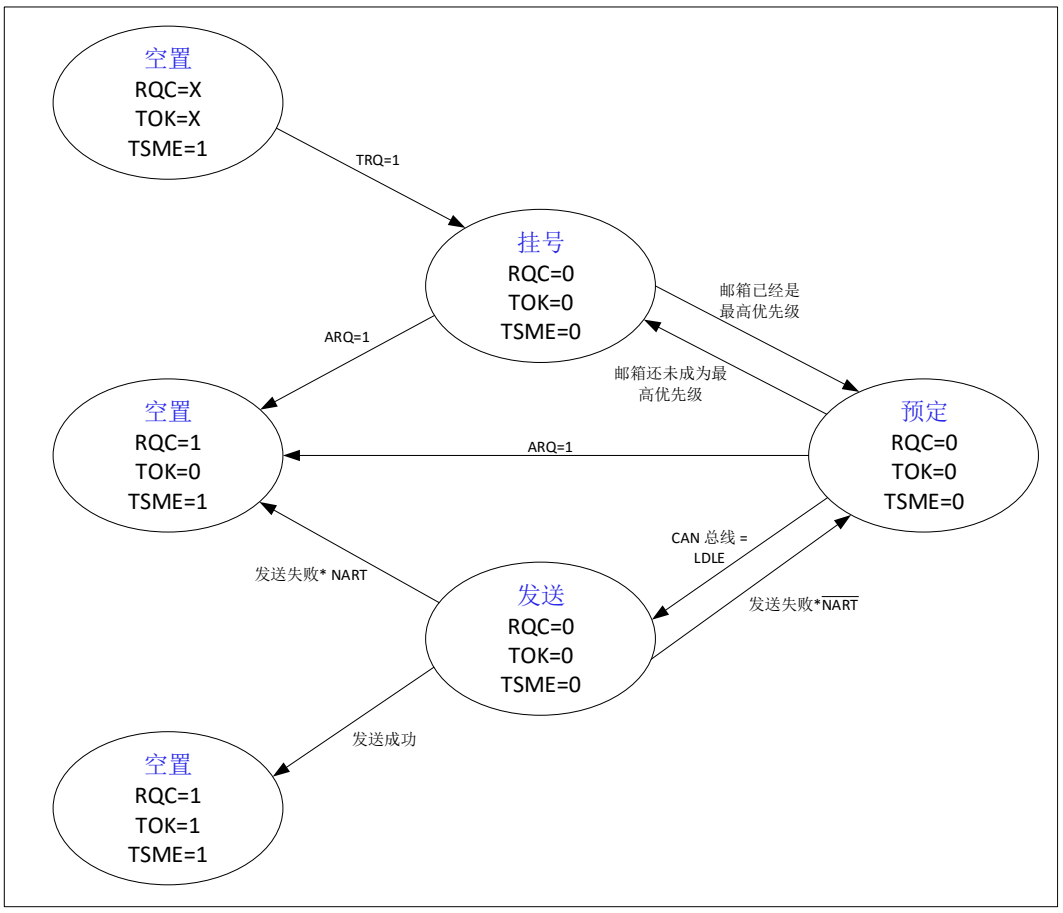
求马上就被中止了。如果邮箱处于发送状态，那么中止请求可能导致两种结果。如果邮箱中的报文被成功发送，那么邮箱变为空置邮箱，并且 CAN_TSTS 寄存器的 TOK 位被硬件置'1'。如果邮箱中的报文发送失败了，那么邮箱变为预定状态，然后发送请求被中止，邮箱变为空置邮箱且 TOK 位被硬件清'0'。因此如果邮箱处于发送状态，那么在发送操作结束后，邮箱都会变为空置邮箱。

禁止自动重传模式

该模式主要用于满足 CAN 标准中，时间触发通信选项的需求。通过对 CAN_MCTRL 寄存器的 NART 位置'1'，来让硬件工作在该模式。 在该模式下，发送操作只会执行一次。如果发送操作失败了，不管是由于仲裁丢失或出错，硬件都不会再自动发送该报文。

在一次发送操作结束后，硬件认为发送请求已经完成，从而对 CAN_TSTS 寄存器的 RQC 位置'1'，同时发送的结果反映在 TOK、ALS 和 TER 位上。

图 18-7 发送邮箱状态



18.3.6 时间触发通信模式

在该模式下，CAN 硬件的内部定时器被激活，并且被用于产生（发送与接收邮箱的）时间戳，分别存储在 CAN_RDTx/CAN_TDTx 寄存器中。内部定时器在每个 CAN 位时间累加。内部定时器在接收和发送的帧起始位的采样点位置被采样，并生成时间戳。

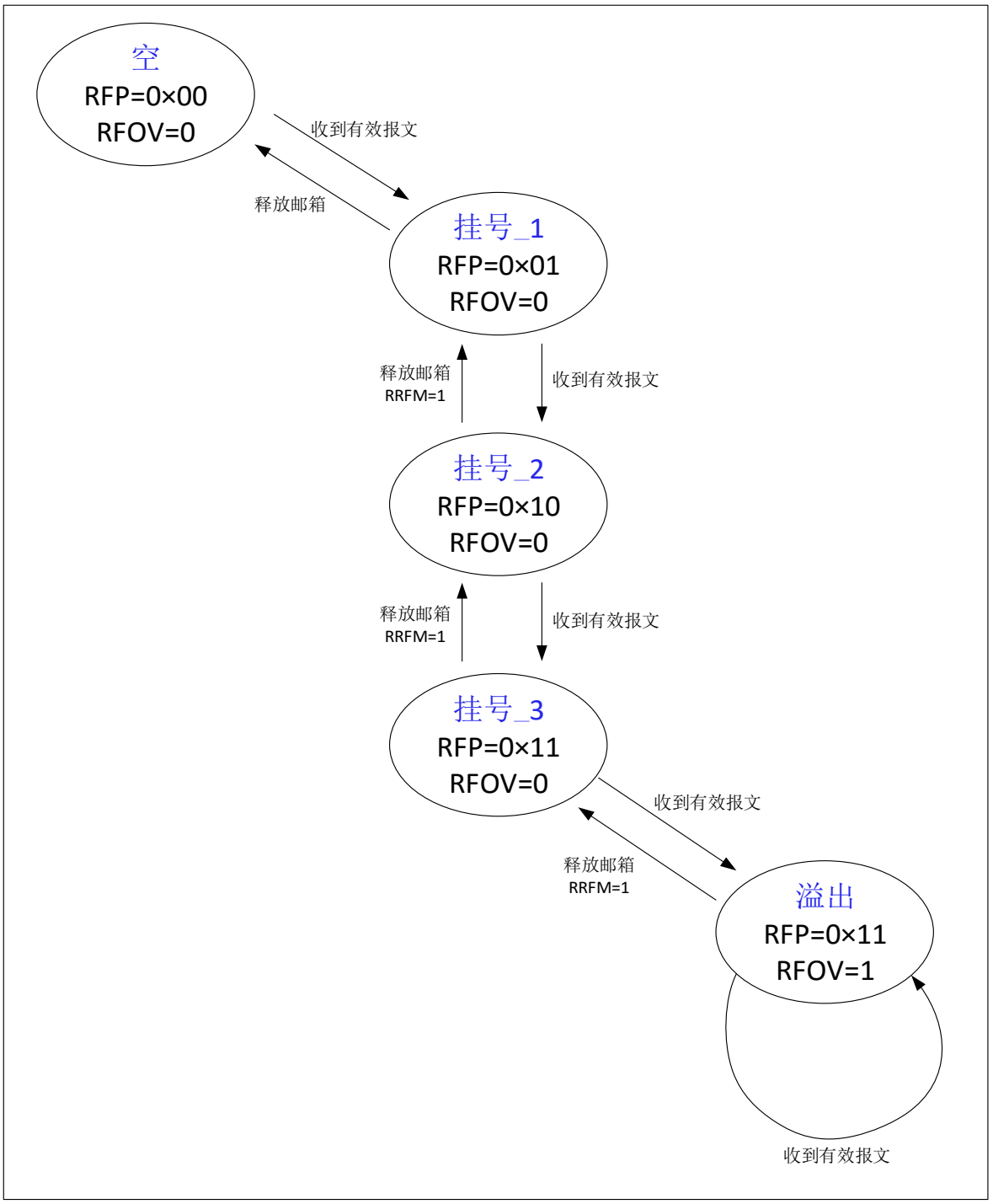
18.3.7 接收管理

接收到的报文，被存储在 3 级邮箱深度的 FIFO 中。FIFO 完全由硬件来管理，从而节省了 CPU 的处理负荷，简化了软件并保证了数据的一致性。应用程序只能通过读取 FIFO 输出邮箱，来读取 FIFO 中最先收到的报文。

有效报文

根据 CAN 协议，当报文被正确接收（直到 EOF 域的最后一位都没有错误），且通过了标识符过滤，那么该报文被认为是有效报文。

图 18-8 接收FIFO状态



FIFO 管理

FIFO 从空状态开始，在接收到第一个有效的报文后，FIFO 状态变为挂号_1（pending_1），硬件相应地把 CAN_RF 寄存器的 RFP[1:0]设置为’01’（二进制 01b）。软件可以读取 FIFO 输出邮箱来读出邮箱中的报文，然后通过对 CAN_RF 寄存器的 RRFM 位设置’1’来释放邮箱，这样 FIFO 又变为空状态了。如果在释放邮箱的同时，又收到了一个有效的报文，那么 FIFO 仍然保留在挂号_1 状态，软件可以读取 FIFO 输出邮箱来读出新收到的报文。

如果应用程序不释放邮箱，在接收到下一个有效的报文后，FIFO 状态变为挂号_2（pending_2），硬件相应地把 RFP[1:0]设置为’10’（二进制 10b）。重复上面的过程，第三个有效的报文把 FIFO 变为挂号_3 状态（RFP[1:0]＝11b）。此时，软件必须对 RRFM 位设置 1 来释放邮箱，以便 FIFO 可以有空间来存放下一个有效的报文；否则，下一个有效的报文到来时就会导致一个报文的丢失。

溢出

当 FIFO 处于挂号_3 状态（即 FIFO 的 3 个邮箱都是满的），下一个有效的报文就会导致溢出，并且一个报文会丢失。此时，硬件对 CAN_RF 寄存器的 RFOV 位进行置'1'来表明溢出情况。至于哪个报文会被丢弃，取决于对 FIFO 的设置：

- 如果禁用了 FIFO 锁定功能（CAN_MCTRL 寄存器的 RFL 位被清'0'），那么 FIFO 中最后收到的报文就被新报文所覆盖。这样，最新收到的报文不会被丢弃掉。
- 如果启用了 FIFO 锁定功能（CAN_MCTRL 寄存器的 RFL 位被置'1'），那么新收到的报文就被丢弃，软件可以读到 FIFO 中最早收到的 3 个报文。

接收相关的中断

一旦往 FIFO 存入一个报文，硬件就会更新 RFP[1：0]位，并且如果 CAN_INTEN 寄存器的 RFPIE 位为'1'，那么就会产生一个中断请求。

当 FIFO 满时（即第 3 个报文被存入），CAN_RF 寄存器的 RFFU 位就被置'1'，并且如果 CAN_INTEN 寄存器的 RFFUIE 位为'1'，那么就会产生一个满中断请求。在溢出的情况下，RFOV 位被置'1'，并且如果 CAN_INTEN 寄存器的 RFOVIE 位为'1'，那么就会产生一个溢出中断请求。

18.3.8 标识符过滤

在 CAN 协议里，报文的标识符不代表节点的地址，而是跟报文的内容相关的。因此，发送者以广播的形式把报文发送给所有的接收者。节点在接收报文时－根据标识符的值－决定软件是否需要该报文；如果需要，就拷贝到 SRAM 里；如果不需要，报文就被丢弃且无需软件的干预。

为满足这一需求，bxCAN 控制器为应用程序提供了 14 个位宽可变的、可配置的过滤器组（13~0），以便只接收那些软件需要的报文。硬件过滤的做法节省了 CPU 开销，否则就必须由软件过滤从而占用一定的 CPU 开销。每个过滤器组 x 由 2 个 32 位寄存器，CAN_FBxR1 和 CAN_FBxR2 组成。

可变的位宽

每个过滤器组的位宽都可以独立配置，以满足应用程序的不同需求。根据位宽的不同，每个过滤器组可提供：

- 1 个 32 位过滤器，包括：SID[10：0]、EID[17：0]、IDT 和 RTR 位
- 2 个 16 位过滤器，包括：SID[10：0]、IDT、RTR 和 EID[17：15]位

此外过滤器可配置为，屏蔽位模式和标识符列表模式。

屏蔽位模式

在屏蔽位模式下，标识符寄存器和屏蔽寄存器一起，指定报文标识符的任何一位，应该按照"必须匹配"或"不用关心"处理。

标识符列表模式

在标识符列表模式下，屏蔽寄存器也被当作标识符寄存器用。因此，不是采用一个标识符加一个屏蔽位的方式，而是使用 2 个标识符寄存器。接收报文标识符的每一位都必须跟过滤器标识符相同。

过滤器组位宽和模式的设置

过滤器组可以通过相应的 CAN_FM 寄存器配置。在配置一个过滤器组前，必须通过清除 CAN_FA1 寄存器的 FEN 位，把它设置为禁用状态。通过设置 CAN_FS1 的相应 FBSx 位，可以配置一个过滤器组的位宽。通过 CAN_FM1 的 FMSx 位，可以配置对应的屏蔽/标识符寄存器的标识符列表模式或屏蔽位模式。

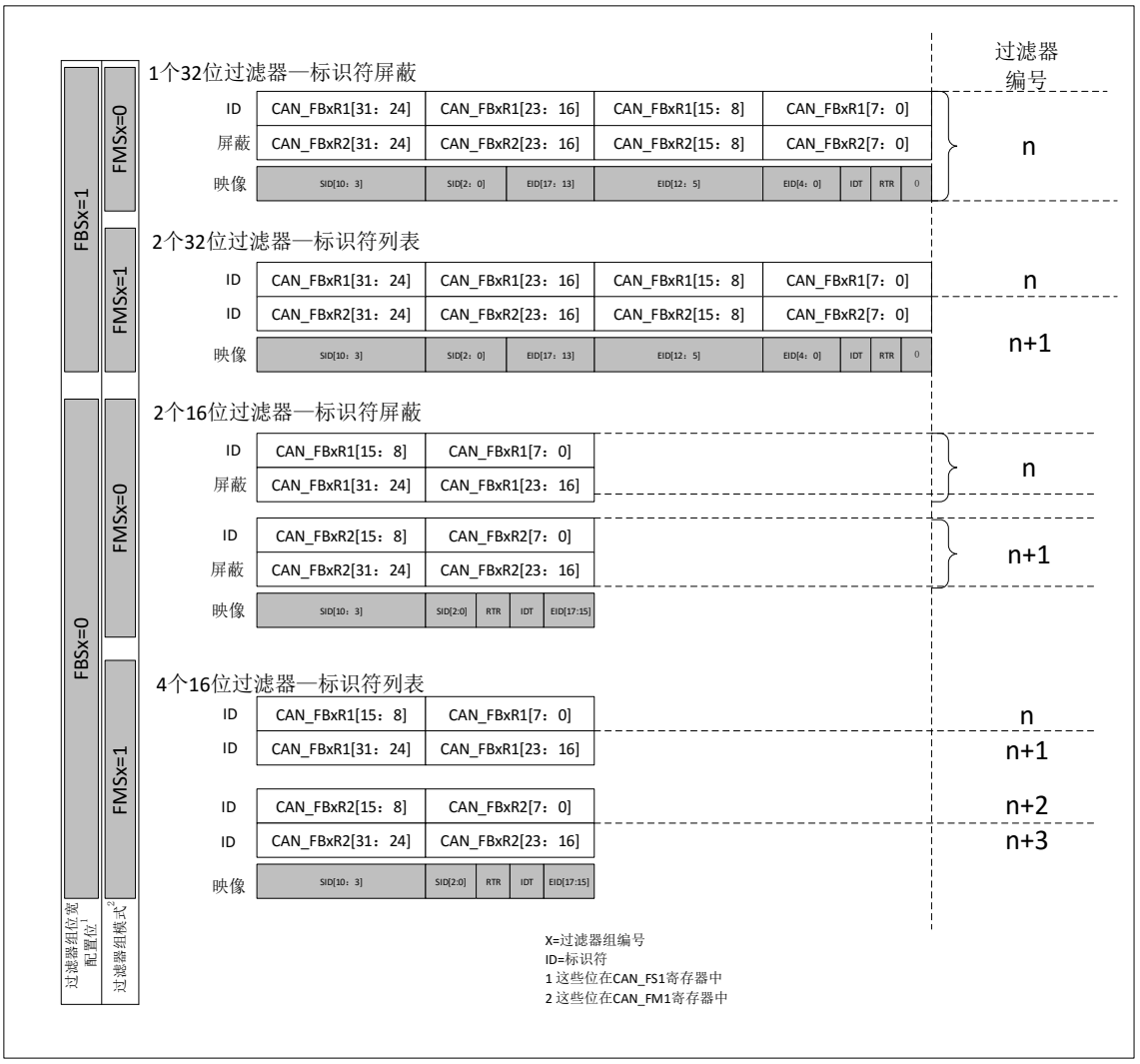
为了过滤出一组标识符，应该设置过滤器组工作在屏蔽位模式。 为了过滤出一个标识符，应该设置过滤器组工作在标识符列表模式。

应用程序不用的过滤器组，应该保持在禁用状态。

过滤器组中的每个过滤器，都被编号为（叫做过滤器号）从 0 开始，到某个最大数值－取决于过滤器组的模式和位宽的设置。

关于过滤器配置，参见下图。

图 18-9 过滤器组位宽设置－寄存器组织



过滤器匹配序号

一旦收到的报文被存入 FIFO，就可被应用程序访问。通常情况下，报文中的数据被拷贝到 SRAM 中；为了把数据拷贝到合适的位置，应用程序需要根据报文的标识符来辨别不同的数据。bxCAN 提供了过滤器匹配序号，以简化这一辨别过程。

根据过滤器优先级规则，过滤器匹配序号和报文一起，被存入邮箱中。因此每个收到的报文，都有与它相关联的过滤器匹配序号。

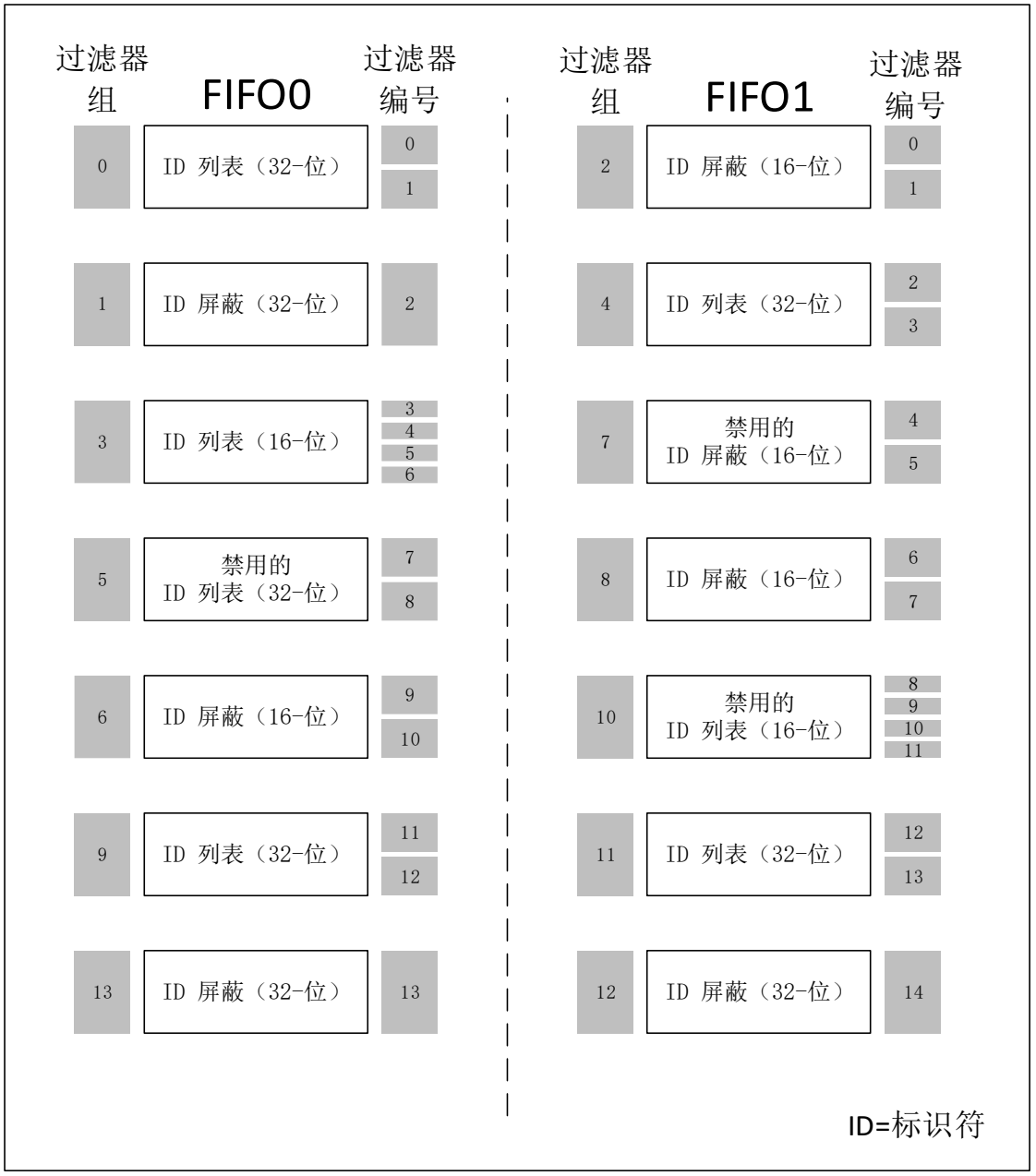
过滤器匹配序号可以通过下面两种方式来使用：

- 把过滤器匹配序号跟一系列所期望的值进行比较
- 把过滤器匹配序号当作一个索引来访问目标地址

对于标识符列表模式下的过滤器（非屏蔽方式的过滤器），软件不需要直接跟标识符进行比较。对于屏蔽位模式下的过滤器，软件只须对需要的那些屏蔽位（必须匹配的位）进行比较即可。

在给过滤器编号时，并不考虑过滤器组是否为激活状态。另外，每个 FIFO 各自对其关联的过滤器进行编号。请参考下图的例子。

图 18-10 过滤器编号的例子

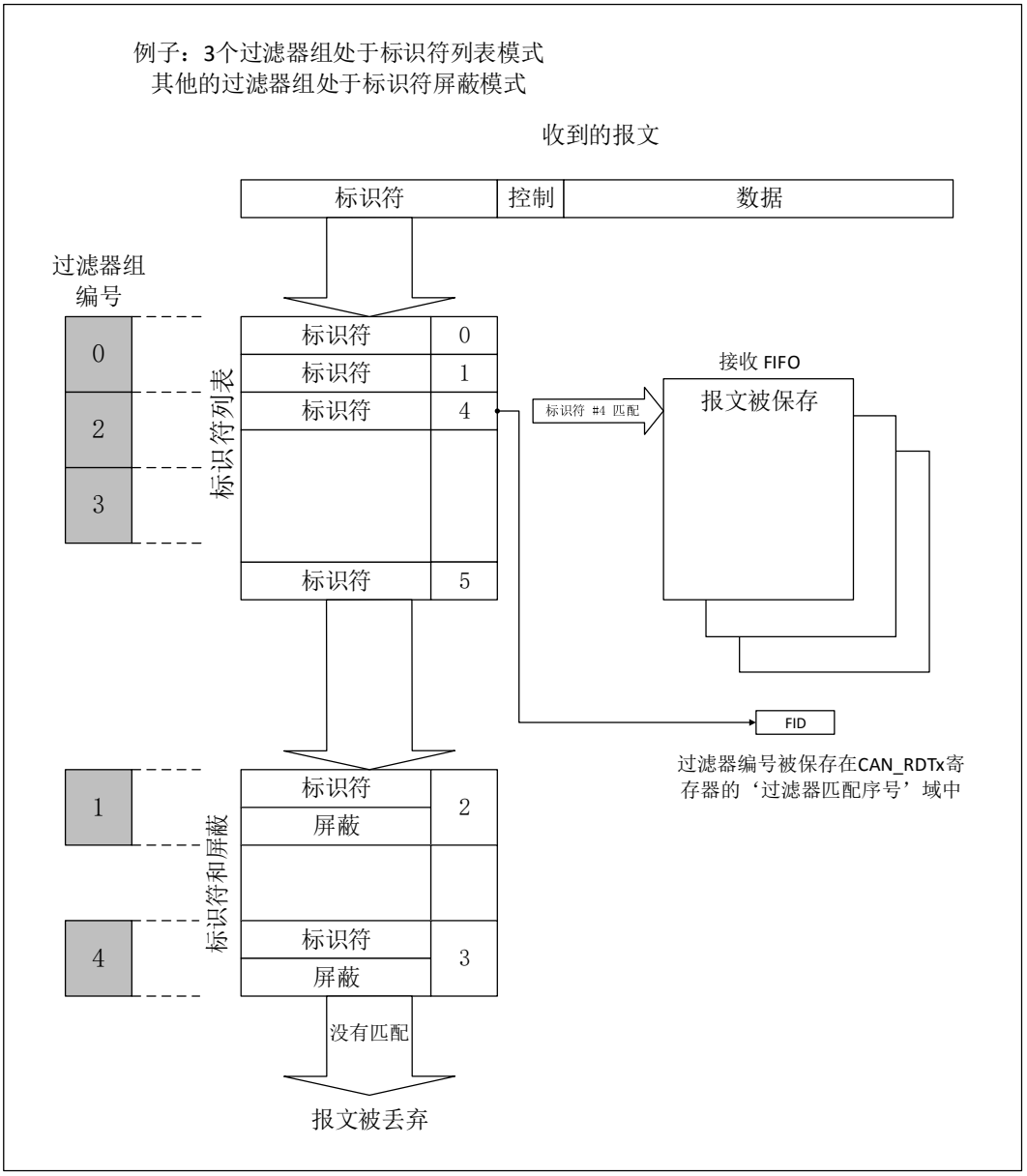


过滤器优先级规则

根据过滤器的不同配置，有可能一个报文标识符能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，根据下列优先级规则来确定：

- 位宽为 32 位的过滤器，优先级高于位宽为 16 位的过滤器
- 对于位宽相同的过滤器，标识符列表模式的优先级高于屏蔽位模式
- 位宽和模式都相同的过滤器，优先级由过滤器号决定，过滤器号小的优先级高

图 18-11 过滤器机制的例子



上面的例子说明了 bxCAN 的过滤器规则：在接收一个报文时，其标识符首先与配置在标识符列表模式下的过滤器相比较；如果匹配上，报文就被存放到相关联的 FIFO 中，并且所匹配的过滤器的序号被存入过滤器匹配序号中。如同例子中所显示，报文标识符跟#4 标识符匹配，因此报文内容和 FID4 被存入 FIFO。

如果没有匹配，报文标识符接着与配置在屏蔽位模式下的过滤器进行比较。如果报文标识符没有跟过滤器中的任何标识符相匹配，那么硬件就丢弃该报文，且不会对软件有任何打扰。

18.3.9 报文存储

邮箱是软件和硬件之间传递报文的接口。邮箱包含了所有跟报文有关的信息：标识符、数据、控制、状态和时间戳信息。

发送邮箱

软件需要在一个空的发送邮箱中，把待发送报文的各种信息设置好（然后再发出发送的请求）。发送的状态可通过查询 CAN_TSTS 寄存器获知。

表 18-1 发送邮箱寄存器列表

相对发送邮箱基地址的偏移量	寄存器名
0	CAN_TMIx
4	CAN_TDTx
8	CAN_TDLx
12	CAN_TDHx

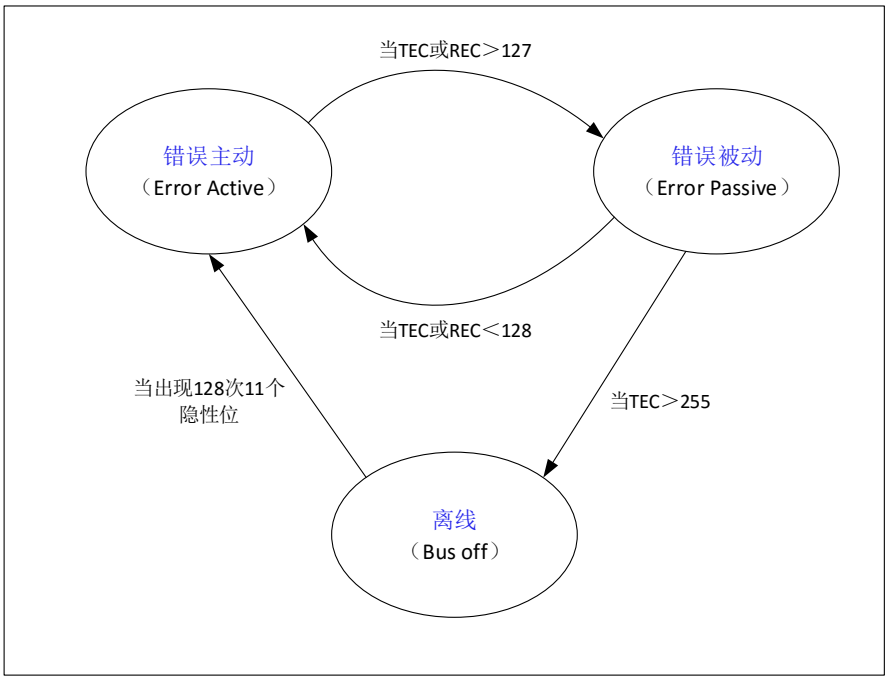
接收邮箱（FIFO）

在接收到一个报文后，软件就可以访问接收 FIFO 的输出邮箱来读取它。一旦软件处理了报文（如把它读出来），软件就应该对 CAN_RFx 寄存器的 RRFM 位进行置'1'，来释放该报文，以便为后面收到的报文留出存储空间。过滤器匹配序号存放在 CAN_RDTx 寄存器的 FID 域中。16 位的时间戳存放在 CAN_RDTx 寄存器的 TS[15：0]域中。

表 18-2 接收邮箱寄存器列表

相对接收邮箱基地址的偏移量	寄存器名
0	CAN_RFIx
4	CAN_RDTx
8	CAN_RDLx
12	CAN_RDHx

图 18-12 CAN错误状态图



18.3.10 出错管理

CAN 协议描述的出错管理，完全由硬件通过发送错误计数器（CAN_ESTS 寄存器里的 TEC 域），和接收错误计数器（CAN_ESTS 寄存器里的 REC 域）来实现，其值根据错误的情况而增加或减少。关于 TEC 和 REC 管理的详细信息，请参考 CAN 标准。

软件可以读出它们的值来判断 CAN 网络的稳定性。

此外，CAN_ESTS 寄存器提供了当前错误状态的详细信息。通过设置 CAN_INTEN 寄存器（比如 ERIE

位），当检测到出错时软件可以灵活地控制中断的产生。

离线恢复

当 TEC 大于 255 时，bxCAN 就进入离线状态，同时 CAN_ESTS 寄存器的 BFF 位被置'1'。在离线状态下，bxCAN 无法接收和发送报文。

根据 CAN_MCTRL 寄存器中 ABO 位的设置，bxCAN 可以自动或在软件的请求下，从离线状态恢复（变为错误主动状态）。在这两种情况下，bxCAN 都必须等待一个 CAN 标准所描述的恢复过程（CAN RX 引脚上检测到 128 次 11 个连续的隐性位）。

如果 ABO 位为'1'，bxCAN 进入离线状态后，就自动开启恢复过程。如果 ABO 位为'0'，软件必须先请求 bxCAN 进入然后再退出初始化模式，随后恢复过程才被开启。

注意：在初始化模式下，bxCAN 不会监视 CAN RX 引脚的状态，这样就不能完成恢复过程。为了完成恢复过程，bxCAN 必须工作在正常模式。

18.3.11 位时间特性

位时间特性逻辑通过采样来监视串行的 CAN 总线，并且通过与帧起始位的边沿进行同步，及通过与后面的边沿进行重新同步，来调整其采样点。

它的操作可以简单解释为，如下所述把名义上的每位时间分为三段：

- 同步段（SYNC_SEG）：通常期望位的变化发生在该时间段内。其值固定为 1 个时间单元（1 x t_{CAN}）。
- 时间段 1（BS1）：定义采样点的位置。它包含 CAN 标准里的 PROP_SEG 和 PHASE_SEG1。其值可以编程为 1 到 16 个时间单元，但也可以被自动延长，以补偿因为网络中不同节点的频率差异所造成的相位的正向漂移。
- 时间段 2（BS2）：定义发送点的位置。它代表 CAN 标准里的 PHASE_SEG2。其值可以编程为 1 到 8 个时间单元，但也可以被自动缩短以补偿相位的负向漂移。

重新同步跳跃宽度（SJW）定义了，在每位中可以延长或缩短多少个时间单元的上限。其值可以编程为 1 到 4 个时间单元。

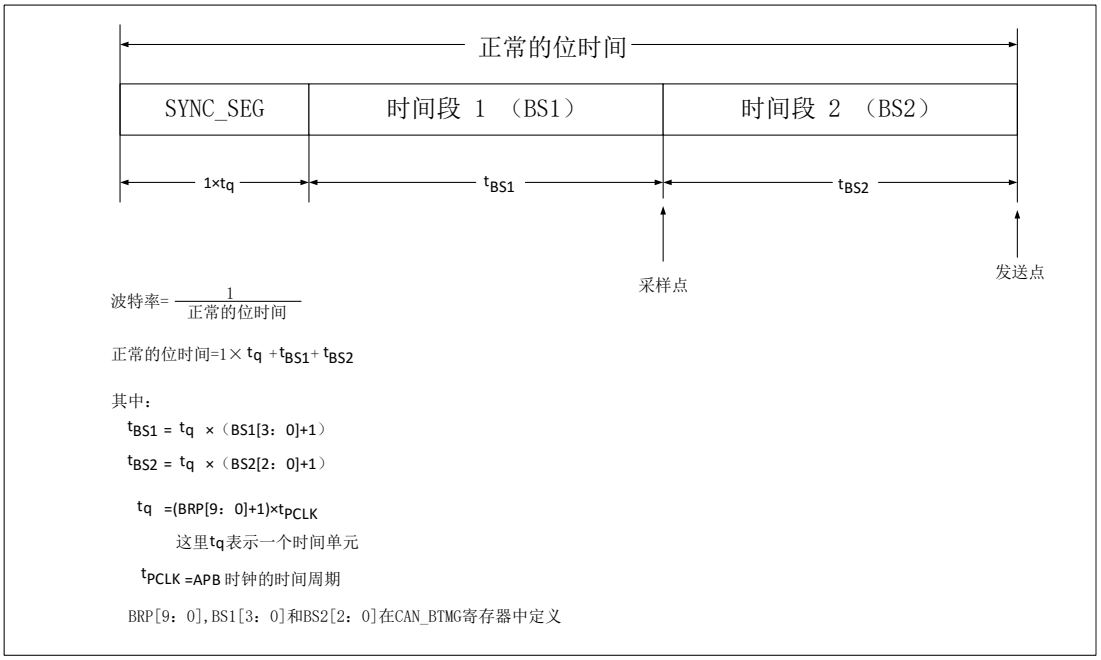
有效跳变被定义为，当 bxCAN 自己没有发送隐性位时，从显性位到隐性位的第 1 次转变。如果在时间段 1（BS1）而不是在同步段（SYNC_SEG）检测到有效跳变，那么 BS1 的时间就被延长最多 SJW 那么长，从而采样点被延迟了。

相反如果在时间段 2（BS2）而不是在 SYNC_SEG 检测到有效跳变，那么 BS2 的时间就被缩短最多 SJW 那么长，从而采样点被提前了。

为了避免软件的编程错误，对位时间特性寄存器（CAN_BTMG）的设置，只能在 bxCAN 处于初始化状态下进行。

注意：关于 CAN 位时间特性和重同步机制的详细信息，请参考 ISO11898 标准。

图 18-13 位时序



波特率= $\frac{1}{\text{正常的位时间}}$

正常的位时间=1× t_q + t_{BS1} + t_{BS2}

其中：

$t_{BS1} = t_q \times$（BS1[3：0]+1）

$t_{BS2} = t_q \times$（BS2[2：0]+1）

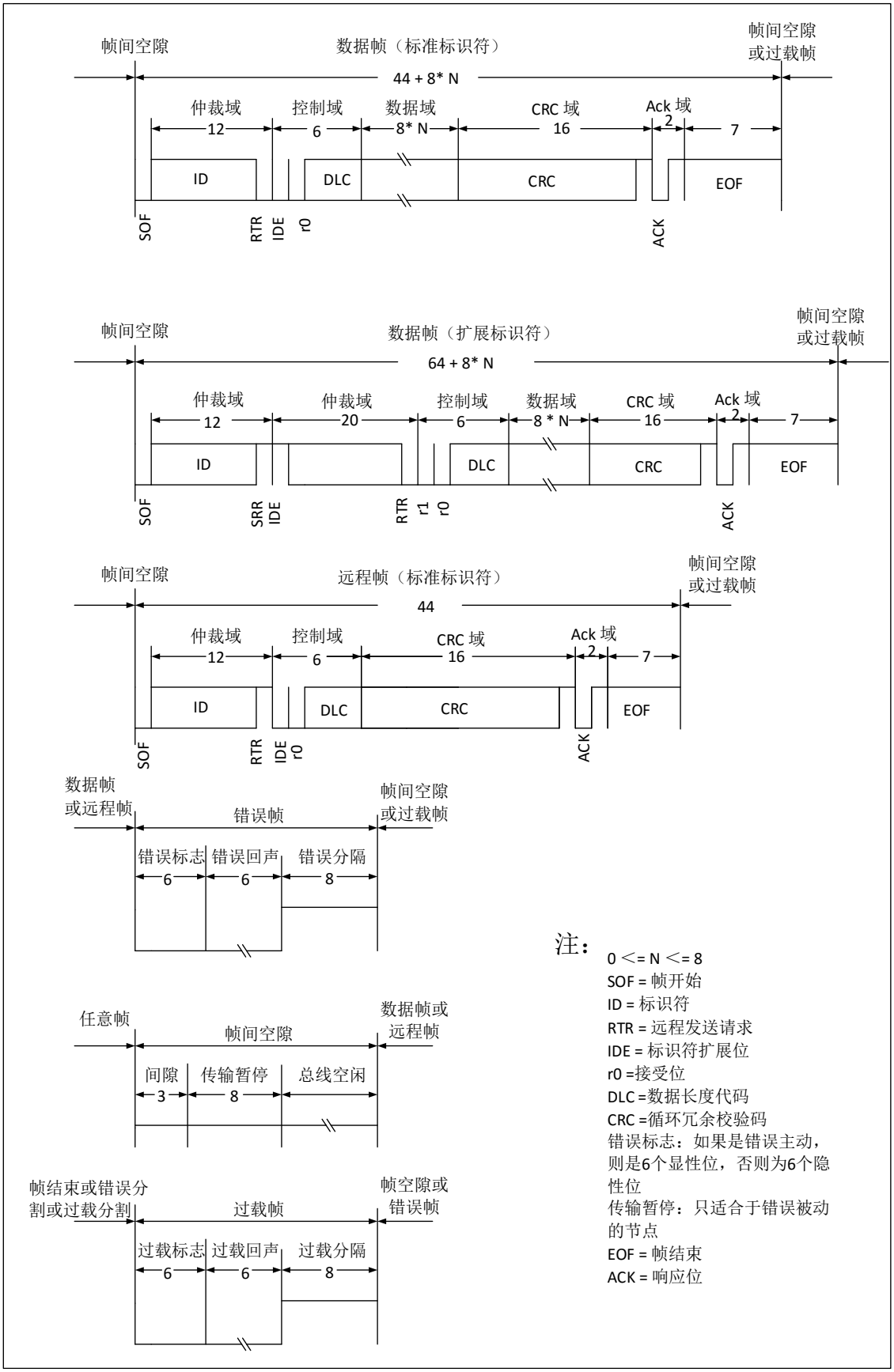
t_q =(BRP[9：0]+1)×t_{PCLK}

这里t_q表示一个时间单元

t_{PCLK} =APB 时钟的时间周期

BRP[9：0],BS1[3：0]和BS2[2：0]在CAN_BTMG寄存器中定义

图 18-14 各种CAN帧



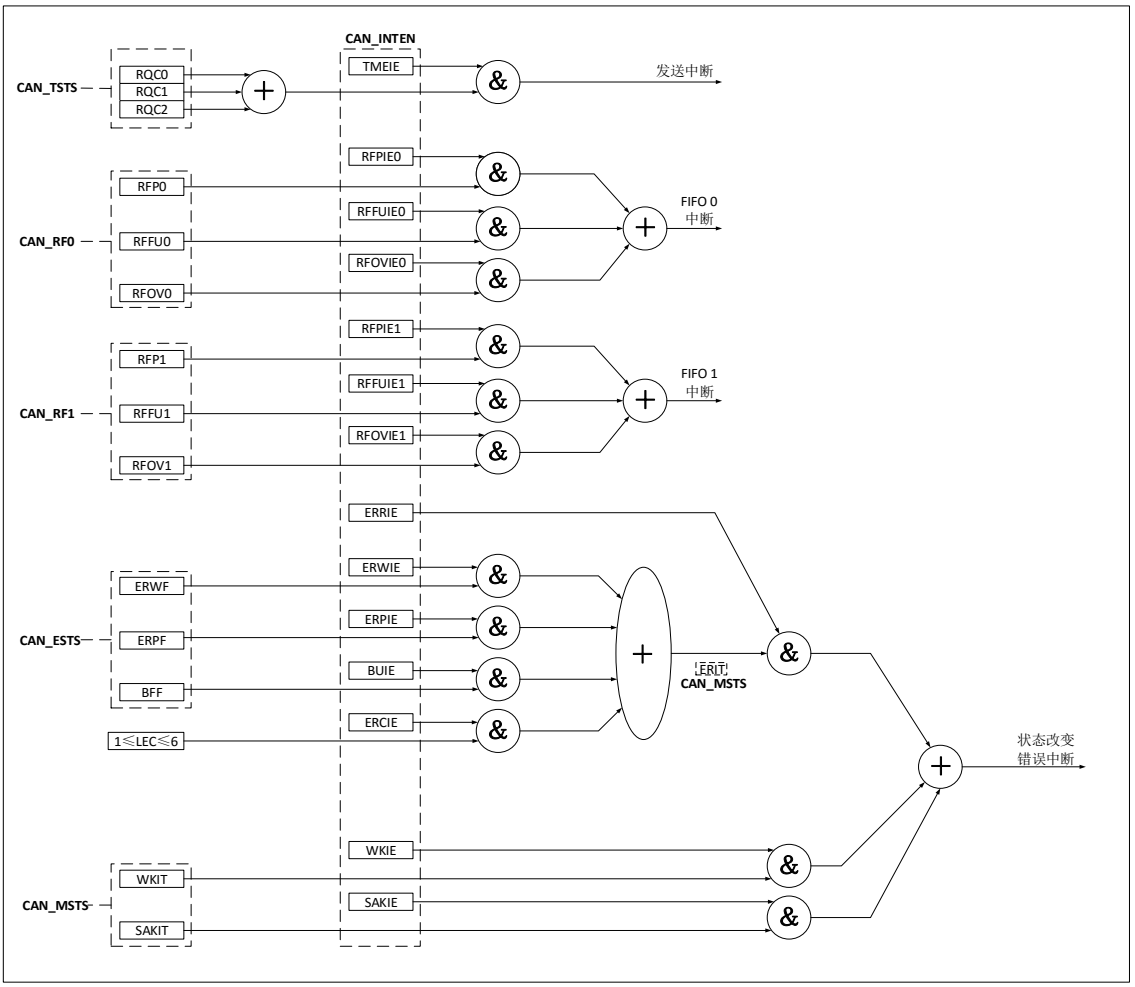
注：

0 <= N <= 8
SOF = 帧开始
ID = 标识符
RTR = 远程发送请求
IDE = 标识符扩展位
r0 =接受位
DLC =数据长度代码
CRC =循环冗余校验码
错误标志：如果是错误主动，则是6个显性位，否则为6个隐性位
传输暂停：只适合于错误被动的节点
EOF = 帧结束
ACK = 响应位

18.3.12 bxCAN中断

bxCAN 占用 4 个专用的中断向量。通过设置 CAN 中断允许寄存器（CAN_INTEN），每个中断源都可以单独允许和禁用。

图 18-15 事件标志和中断产生



- 发送中断可由下列事件产生：
  - 发送邮箱 0 变为空，CAN_TSTS 寄存器的 RQC0 位被置’1’。
  - 发送邮箱 1 变为空，CAN_TSTS 寄存器的 RQC1 位被置’1’。
  - 发送邮箱 2 变为空，CAN_TSTS 寄存器的 RQC2 位被置’1’。
- FIFO0 中断可由下列事件产生：
  - FIFO0 接收到一个新报文，CAN_RF0 寄存器的 RFP0 位不再是’00’。
  - FIFO0 变为满的情况，CAN_RF0 寄存器的 RFFU0 位被置’1’。
  - FIFO0 发生溢出的情况，CAN_RF0 寄存器的 RFOV0 位被置’1’。
- FIFO1 中断可由下列事件产生：
  - FIFO1 接收到一个新报文，CAN_RF1 寄存器的 RFP1 位不再是’00’。
  - FIFO1 变为满的情况，CAN_RF1 寄存器的 RFFU1 位被置’1’。
  - FIFO1 发生溢出的情况，CAN_RF1 寄存器的 RFOV1 位被置’1’。
- 错误和状态变化中断可由下列事件产生：
  - 出错情况，关于出错情况的详细信息请参考 CAN 错误状态寄存器（CAN_ESTS）。
  - 唤醒情况，在 CAN 接收引脚上监视到帧起始位（SOF）。
  - CAN 进入睡眠模式。

18.4 CAN 寄存器

必须以字（32 位）的方式操作这些外设寄存器。

表 18-3

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000h	MCTRL	保留															DBF	RST	保留							TTC	ABO	AWU	NART	RFL	TFP	SLP	INRQ
	复位值																1	0								0	0	0	0	0	0	1	0
004h	MSTS	保留																				RXS	LSAP	RX	TX	保留			SAKIT	WKIT	ERIT	SAK	IAK
	复位值																					1	1	0	0				0	0	0	1	0
008h	TSTS	LPM			TSME			NTM		ARQ2	保留			TER2	ALS2	TOK2	RQC2	ARQ1	保留			TER1	ALS1	TOK1	RQC1	ARQ0	保留			TER0	ALS0	TOK0	RQC0
	复位值	0	0	0	1	1	1	0	0	0				0	0	0	0	0				0	0	0	0	0				0	0	0	0
00Ch	RF0	保留																										RRFM0	RFOV0	RFFU0	保留	RFP0	
	复位值																											0	0	0		0	0
010h	FR1	保留																										RRFM1	RFOV1	RFFU1	保留	RFP1	
	复位值																											0	0	0		0	0
014h	INTEN	保留														SAKIE	WKIE	ERIE	保留			ERCIE	BUIE	ERPIE	ERWIE	保留	RFOVIE1	RFFUIE1	RFPIE1	RFOVIE0	RFFUIE0	RFPIE0	TSMEIE
	复位值															0	0	0				0	0	0	0		0	0	0	0	0	0	0
偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
018h	ESTS	REC[7：0]								TEC[7：0]								保留								ERC			保留	BFF	ERPF	ERWF	
	复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0									0	0	0		0	0	0	
01Ch	BTMG	SIL	LBK	保留				SJW		保留	BS2			BS1				保留				BRP											
	复位值	0	0					0	1		0	1	0	0	0	1	1					0	0	0	0	0	0	0	0	0	0	0	0
020h~17Fh	保留																																
180h	TMI0	SID/EID[28：18]											EID[17：0]																		IDT	RTR	TRQ
	复位值	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
184h	TDT0	TS																	保留						TMEN	保留				DLC			
	复位值	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x								x					x	x	x	x
188h	TDL0	D3								D2								D1								D0							
	复位值	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
18Ch	TDH0	D7								D6								D5								D4							
	复位值	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

190h	TMI1	SID/EID[28：18]										EID[17：0]																		IDT	RTR	TRQ
	复位值	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
194h	TDT1	TS															保留							TMEN	保留					DLC			
	复位值	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
198h	TDL1	D3								D2								D1								D0							
	复位值	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
19Ch	TDH1	D7								D6								D5								D4							
	复位值	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
1A0h	TMI2	SID/EID[28：18]										EID[17：0]																		IDT	RTR	TRQ
	复位值	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
1A4h	TDT2	TS															保留							TMEN	保留					DLC			
	复位值	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x								x						x	x	x	x
1A8h	TDL2	D3								D2								D1								D0							
	复位值	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
1ACh	TDH2	D7								D6								D5								D4							
	复位值	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
1B0h	RFI0	SID/EID[28：18]										EID[17：0]																		IDT	RTR	保留
	复位值	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x		
1B4h	RDT0	TS															FID								保留					DLC			
	复位值	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x					x	x	x	x
偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1B8h	RDL0	D3								D2								D1								D0							
	复位值	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
1BCh	RDH0	D7								D6								D5								D4							
	复位值	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
1C0h	RFI1	SID/EID[28：18]										EID[17：0]																		IDT	RTR	保留
	复位值	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x		
1C4h	RDT1	TS															FID								保留					DLC			
	复位值	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x					x	x	x	x
1C8h	RDL1	D3								D2								D1								D0							
	复位值	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
1CCh	RDH1	D7								D6								D5								D4							
	复位值	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
1D0h~1FFh	保留																																
200h	FM	保留																														FINT	
	复位值																															1	
204h	FM1	保留																		FMS													
	复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
208h	保留																																
20Ch	FS1	保留																		FBS													
	复位值																			0	0	0	0	0	0	0	0	0	0	0	0	0	0
210h	保留																																
214h	FFA1	保留																		FAF													

	复位值																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
218h	保留																														
21Ch	FA1	保留															FEN														
	复位值																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
220h~23Fh	保留																														
240h	FB0R1	FD																													
	复位值	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
244h	FB0R2	FD																													
	复位值	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
248h	FB1R1	FD																													
	复位值	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
24Ch	FB1R2	FD																													
	复位值	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
…	…	……																													
2A8h	FB13R1	FD																													
	复位值	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
2ACh	FB13R2	FD																													
	复位值	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

18.4.1 寄存器访问保护

对某些寄存器的错误访问会导致一个 CAN 节点对整个 CAN 网络的暂时性干扰。因此，软件只能在 CAN 处于初始化模式时修改 CAN_BTMG 寄存器。

虽然错误数据的发送对 CAN 网的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，软件只能在发送邮箱为空的状态改变它。

过滤器的数值只能在关闭对应过滤器组的状态下，或设置 FINT 位为'1'后才能修改。此外，只有在设置整个过滤器为初始化模式下（即 FINT=1），才能修改过滤器的设置，即修改 CAN_FM1，CAN_FS1 和 CAN_FFA1 寄存器。

18.4.2 CAN控制和状态寄存器

18.4.2.1 CAN主控制寄存器（CAN_MCTRL）

地址偏移量：0x00

复位值：0x0001 0002

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															DBF
res															rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RST	保留							TTC	ABO	AWU	NART	RFL	TFP	SLP	INRQ
rs	res							rw	rw	rw	rw	rw	rw	rw	rw

位 31：17	保留，硬件强制为 0。
位 16	**DBF**：调试冻结 （Debug freeze） 0：在调试时，CAN 照常工作 1：在调试时，冻结 CAN 的接收/发送。仍然可以正常地读写和控制接收 FIFO。

位 15	**RST：** bxCAN 软件复位 （bxCAN software reset） 0：本外设正常工作； 1：对 bxCAN 进行强行复位，复位后 bxCAN 进入睡眠模式（FMP 位和 CAN_MCTRL 寄存器被初始化为其复位值）。此后硬件自动对该位清’0’。
位 14：8	保留，硬件强制为 0。
位 7	**TTC：** 时间触发通信模式 （Time triggered communication mode） 0：禁止时间触发通信模式； 1：允许时间触发通信模式。
位 6	**ABO：** 自动离线（Bus-Off）管理 （Automatic bus-off management） 该位决定 CAN 硬件在什么条件下可以退出离线状态。 0：软件请求退出离线状态，首先软件对 CAN_MCTRL 寄存器的 INRQ 位置‘1’，随后清‘0’，一旦硬件检测到 128 次连续 11 位的隐形位，则退出离线状态； 1：硬件自动退出离线状态，当硬件检测到 128 次连续 11 位的隐性位，则自动退出离线状态。
位 5	**AWU：** 自动唤醒模式 （Automatic wakeup mode） 该位决定 CAN 处在睡眠模式时由硬件还是软件唤醒 0：软件唤醒，软件清除 CAN_MCTRL 寄存器的 SLP 位，退出睡眠模式； 1：硬件自动唤醒，当硬件检测到 CAN 报文时，立即退出睡眠模式。唤醒的同时，硬件自动对 CAN_MSTS 寄存器的 SLP 和 SAK 位清’0’ 。
位 4	**NART：** 禁止报文自动重传 （No automatic retransmission） 0：按照 CAN 标准，CAN 硬件在发送报文失败时会一直自动重传直到发送成功； 1：CAN 报文只被发送 1 次，不管发送的结果如何（成功、出错或仲裁丢失）。
位 3	**RFL：** 接收 FIFO 锁定模式 （Receive FIFO locked mode） 0：在接收溢出时 FIFO 未被锁定，当接收 FIFO 的报文未被读出，下一个收到的报文会覆盖原有的报文； 1：在接收溢出时 FIFO 被锁定，当接收 FIFO 的报文未被读出，下一个收到的报文会被丢弃。
位 2	**TFP：** 发送 FIFO 优先级 （Transmit FIFO priority） 当有多个报文同时在等待发送时，该位决定这些报文的发送顺序 0：优先级由报文的标识符来决定； 1：优先级由发送请求的顺序来决定。
位 1	**SLP：** 睡眠模式请求 （Sleep mode request） 软件对该位置’1’可以请求 CAN 进入睡眠模式，一旦当前的 CAN 活动（发送或接收报文）结束，CAN 就进入睡眠。 软件对该位清’0’使 CAN 退出睡眠模式。 当设置了 AWU 位且在 CANRx 信号中检测出 SOF 位时，硬件对该位清’0’。 在复位后该位被置’1’，即 CAN 在复位后处于睡眠模式。
位 0	**INRQ：** 初始化请求 （Initialization request） 软件对该位清’0’可使 CAN 从初始化模式进入正常工作模式：当 CAN 在接收引脚检测到连续的 11 个隐性位后，CAN 就达到同步，并为接收和发送数据作好准备了。为此，硬件相应地对 CAN_MSTS 寄存器的 IAK 位清’0’。 软件对该位置 1 可使 CAN 从正常工作模式进入初始化模式：一旦当前的 CAN 活动（发送或接收）结束，CAN 就进入初始化模式。相应地，硬件对 CAN_MSTS 寄存器的 IAK 位置’1’。

18.4.2.2 CAN主状态寄存器 （CAN_MSTS）

地址偏移量：0x04

复位值：0x0000 0C02

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															
res															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留				RXS	LSAP	RX	TX	保留			SAKIT	WKIT	ERIT	SAK	IAK
res				r	r	r	r	res			rc w1	rc w1	rc w1	r	r

位 31：12	保留位，硬件强制为 0
位 11	**RXS：**CAN 接收电平 （CANRx signal） 该位反映 CAN 接收引脚（CAN_RX）的实际电平。
位 10	**LSAP：**上次采样值 （Last sample point） CAN 接收引脚的上次采样值（对应于当前接收位的值）。
位 9	**RX：**接收模式 （Receive mode） 该位为’1’表示 CAN 当前为接收器。
位 8	**TX：**发送模式 （Transmit mode） 该位为’1’表示 CAN 当前为发送器。
位 7：5	保留位，硬件强制为 0。
位 4	**SAKIT：**睡眠确认中断 （Sleep acknowledge interrupt） 当 SAKIE=1，一旦 CAN 进入睡眠模式硬件就对该位置’1，紧接着相应的中断被触发。当设置该位为’1’时，如果设置了 CAN_INTEN 寄存器中的 SAKIE 位，将产生一个状态改变中断。 软件可对该位清’0’，当 SAK 位被清’0’时硬件也对该位清’0’。 *注： 当 SAKIE=0，不应该查询该位，而应该查询 SAK 位来获知睡眠状态。*
位 3	**WKIT：**唤醒中断挂号 （Wakeup interrupt） 当 CAN 处于睡眠状态，一旦检测到帧起始位（SOF），硬件就置该位为’1’；并且如果 CAN_INTEN 寄存器的 WKIE 位为’1’，则产生一个状态改变中断。 该位由软件清’0’。
位 2	**ERIT：**出错中断挂号 （Error interrupt） 当检测到错误时，CAN_ESTS 寄存器的某位被置’1’，如果 CAN_INTEN 寄存器的相应中断使能位也被置’1’时，则硬件对该位置’1’；如果 CAN_INTEN 寄存器的 ERRIE 位为’1’，则产生状态改变中断。 该位由软件清’0’。
位 1	**SAK：**睡眠模式确认 该位由硬件置’1’，指示软件 CAN 模块正处于睡眠模式。该位是对软件请求进入睡眠模式的确认（对 CAN_MCTRL 寄存器的 SLP 位置’1’）。 当 CAN 退出睡眠模式时硬件对该位清’0’ （需要跟 CAN 总线同步）。这里跟 CAN 总线同步是指，硬件需要在 CAN 的 RX 引脚上检测到连续的 11 位隐性位。 *注： 通过软件或硬件对 CAN_MCTRL 的 SLP 位清’0’，将启动退出睡眠模式的过程。有关清除 SLP 位的详细信息，参见 CAN_MCTRL 寄存器的 AWU 位的描述。*
位 0	**IAK：**初始化确认 该位由硬件置’1’，指示软件 CAN 模块正处于初始化模式。 该位是对软件请求进入初始化模式的确认（对 CAN_MCTRL 寄存器的 INRQ 位置’1’）。 当 CAN 退出初始化模式时硬件对该位清’0’ （需要跟 CAN 总线同步）。这里跟 CAN 总线同步是指，硬件需要在 CAN 的 RX 引脚上检测到连续的 11 位隐性位。

18.4.2.3 CAN发送状态寄存器 （CAN_TSTS）

地址偏移量：0x08

复位值：0x1C00 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LPM2	LPM1	LPM0	TSME2	TSME1	TSME0	NTM[1：0]		ARQ2	保留			TER2	ALS2	TOK2	RQC2

r	r	r	r	r	r	r	r	rs		res		rc w1	rc w1	rc w1	rc w1
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARQ1	保留			TER1	ALS1	TOK1	RQC1	ARQ0	保留			TER0	ALS0	TOK0	RQC0
rs	res			rc w1	rc w1	rc w1	rc w1	rs	res			rc w1	rc w1	rc w1	rc w1

位 31	**LPM2：** 邮箱 2 最低优先级标志 （Lowest priority flag for mailbox 2） 当多个邮箱在等待发送报文，且邮箱 2 的优先级最低时，硬件对该位置'1'。
位 30	**LPM1：** 邮箱 1 最低优先级标志 （Lowest priority flag for mailbox 1） 当多个邮箱在等待发送报文，且邮箱 1 的优先级最低时，硬件对该位置'1'。
位 29	**LPM0：** 邮箱 0 最低优先级标志 （Lowest priority flag for mailbox 0） 当多个邮箱在等待发送报文，且邮箱 0 的优先级最低时，硬件对该位置'1'。 *注：如果只有 1 个邮箱在等待，则 LPM[2：0]被清'0'。*
位 28	**TSME2：** 发送邮箱 2 空 （Transmit mailbox 2 empty） 当邮箱 2 中没有等待发送的报文时，硬件对该位置'1'。
位 27	**TSME1：** 发送邮箱 1 空 （Transmit mailbox 1 empty） 当邮箱 1 中没有等待发送的报文时，硬件对该位置'1'。
位 26	**TSME0：** 发送邮箱 0 空 （Transmit mailbox 0 empty） 当邮箱 0 中没有等待发送的报文时，硬件对该位置'1'。
位 25：24	**NTM[1：0]：** 邮箱号 （Mailbox code） 当有至少 1 个发送邮箱为空时，这 2 位表示下一个空的发送邮箱号。 当所有的发送邮箱都为空时，这 2 位表示优先级最低的那个发送邮箱号。
位 23	**ARQ2：** 邮箱 2 中止发送 （Abort request for mailbox 2） 软件对该位置'1'，可以中止邮箱 2 的发送请求，当邮箱 2 的发送报文被清除时硬件对该位清'0'。 如果邮箱 2 中没有等待发送的报文，则对该位置'1'没有任何效果。
位 22：20	保留位，硬件强制其值为 0
位 19	**TER2：** 邮箱 2 发送失败 （Transmission error of mailbox 2） 当邮箱 2 因为出错而导致发送失败时，对该位置'1'。
位 18	**ALS2：** 邮箱 2 仲裁丢失 （Arbitration lost for mailbox 2） 当邮箱 2 因为仲裁丢失而导致发送失败时，对该位置'1'。
位 17	**TOK2：** 邮箱 2 发送成功 （Transmission OK of mailbox 2） 每次在邮箱 2 进行发送尝试后，硬件对该位进行更新： 0：上次发送尝试失败； 1：上次发送尝试成功。 当邮箱 2 的发送请求被成功完成后，硬件对该位置'1'。
位 16	**RQC2：** 邮箱 2 请求完成 （Request completed mailbox 2） 当上次对邮箱 2 的请求（发送或中止）完成后，硬件对该位置'1'。软件对该位写'1'可以对其清'0'；当硬件接收到发送请求时也对该位清'0'（CAN_TMI2 寄存器的 TRQ 位被置'1'）。 该位被清'0'时，邮箱 2 的其它发送状态位（TOK2, ALS2 和 TER2）也被清'0'。
位 15	**ARQ1：** 邮箱 1 中止发送 （Abort request for mailbox 1） 软件对该位置'1'，可以中止邮箱 1 的发送请求，当邮箱 1 的发送报文被清除时硬件对该位清'0'。 如果邮箱 1 中没有等待发送的报文，则对该位置'1'没有任何效果。
位 14：12	保留位，硬件强制其值为 0
位 11	**TER1：** 邮箱 1 发送失败 （Transmission error of mailbox 1） 当邮箱 1 因为出错而导致发送失败时，对该位置'1'。
位 10	**ALS1：** 邮箱 1 仲裁丢失 （Arbitration lost for mailbox 1） 当邮箱 1 因为仲裁丢失而导致发送失败时，对该位置'1'。

位 9	**TOK1：** 邮箱 1 发送成功 （Transmission OK of mailbox 1） 每次在邮箱 1 进行发送尝试后，硬件对该位进行更新： 0：上次发送尝试失败； 1：上次发送尝试成功。 当邮箱 1 的发送请求被成功完成后，硬件对该位置’1’。
位 8	**RQC1：** 邮箱 1 请求完成 （Request completed mailbox 1） 当上次对邮箱 1 的请求（发送或中止）完成后，硬件对该位置’1’。软件对该位写’1’可以对其清’0’；当硬件接收到发送请求时也对该位清’0’（CAN_TMI1 寄存器的 TRQ 位被置’1’）。 该位被清’0’时，邮箱 1 的其它发送状态位（TOK1, ALS1 和 TER1）也被清’0’。
位 7	**ARQ0：** 邮箱 0 中止发送 （Abort request for mailbox 0） 软件对该位置’1’可以中止邮箱 0 的发送请求，当邮箱 0 的发送报文被清除时硬件对该位清’0’。 如果邮箱 0 中没有等待发送的报文，则对该位置 1 没有任何效果。
位 6：4	保留位，硬件强制其值为 0
位 3	**TER0：** 邮箱 0 发送失败 （Transmission error of mailbox 0） 当邮箱 0 因为出错而导致发送失败时，对该位置’1’。
位 2	**ALS0：** 邮箱 0 仲裁丢失 （Arbitration lost for mailbox 0） 当邮箱 0 因为仲裁丢失而导致发送失败时，对该位置’1’。
位 1	**TOK0：** 邮箱 0 发送成功 （Transmission OK of mailbox 0） 每次在邮箱 0 进行发送尝试后，硬件对该位进行更新： 0：上次发送尝试失败； 1：上次发送尝试成功。当邮箱 0 的发送请求被成功完成后，硬件对该位置’1’。
位 0	**RQC0：** 邮箱 0 请求完成 （Request completed mailbox 0） 当上次对邮箱 0 的请求（发送或中止）完成后，硬件对该位置’1’。 软件对该位写’1’可以对其清’0’；当硬件接收到发送请求时也对该位清’0’（CAN_TMI0 寄存器的 TRQ 位被置’1’）。 该位被清’0’时，邮箱 0 的其它发送状态位（TOK0, ALS0 和 TER0）也被清’0’。

18.4.2.4 CAN接收FIFO 0寄存器 （CAN_RF0）

地址偏移量：0x0C

复位值：0x00

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															
res															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留										RRFM0	RFOV0	RFFU0	保留	RFP0	
res										rs	rc w1	rc w1	res	r	r

位 31：6	保留位，硬件强制为 0
位 5	**RRFM0：** 释放接收 FIFO 0 输出邮箱 （Release FIFO 0 output mailbox） 软件通过对该位置’1’来释放接收 FIFO 的输出邮箱。如果接收 FIFO 为空，那么对该位置’1’没有任何效果，即只有当 FIFO 中有报文时对该位置’1’才有意义。如果 FIFO 中有 2 个以上的报文，由于 FIFO 的特点，软件需要释放输出邮箱才能访问第 2 个报文。 当输出邮箱被释放时，硬件对该位清’0’。
位 4	**RFOV0：** FIFO 0 溢出 （FIFO 0 overrun） 当 FIFO 0 已满，又收到新的报文且报文符合过滤条件，硬件对该位置’1’。 该位由软件清’0’。
位 3	**RFFU0：** FIFO 0 满 （FIFO 0 full） 当 FIFO 0 中有 3 个报文时，硬件对该位置’1’。 该位由软件清’0’。

位 2	保留位，硬件强制其值为 0
位 1：0	**RFP0[1：0]：** FIFO 0 报文数目 （FIFO 0 message pending） FIFO 0 报文数目这 2 位反映了当前接收 FIFO 0 中存放的报文数目。 每当 1 个新的报文被存入接收 FIFO 0，硬件就对 RFP0 加 1。 每当软件对 RRFM0 位写’1’来释放输出邮箱，RFP0 就被减 1，直到其为 0。

18.4.2.5 CAN接收FIFO 1寄存器（CAN_RF1）

地址偏移量：0x10

复位值：0x00

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															
res															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留										RRFM1	RFOV1	RFFU1	保留	RFP1	
res										rs	rc w1	rc w1	res	r	r

位 31：6	保留位，硬件强制为 0
位 5	**RRFM1：** 释放接收 FIFO 1 输出邮箱 （Release FIFO 1 output mailbox） 软件通过对该位置’1’来释放接收 FIFO 的输出邮箱。如果接收 FIFO 为空，那么对该位置’1’没有任何效果，即只有当 FIFO 中有报文时对该位置’1’才有意义。如果 FIFO 中有 2 个以上的报文，由于 FIFO 的特点，软件需要释放输出邮箱才能访问第 2 个报文。 当输出邮箱被释放时，硬件对该位清’0’。
位 4	**RFOV1：** FIFO 1 溢出 （FIFO 1 overrun） 当 FIFO 1 已满，又收到新的报文且报文符合过滤条件，硬件对该位置’1’。 该位由软件清’0’。
位 3	**RFFU1：** FIFO 1 满 （FIFO 1 full） 当 FIFO 1 中有 3 个报文时，硬件对该位置’1’。 该位由软件清’0’。
位 2	保留位，硬件强制其值为 0
位 1：0	**RFP1[1：0]：** FIFO 1 报文数目 （FIFO 1 message pending） FIFO 1 报文数目这 2 位反映了当前接收 FIFO 1 中存放的报文数目。 每当 1 个新的报文被存入接收 FIFO 1，硬件就对 RFP1 加 1。 每当软件对 RRFM1 位写 1 来释放输出邮箱，RFP1 就被减 1，直到其为 0。

18.4.2.6 CAN中断使能寄存器 （CAN_INTEN）

地址偏移量：0x14

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留														SAKIE	WKIE
res														rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ERIE	保留			ERCIE	BUIE	ERPIE	ERWIE	保留	RFOVIE1	RFFUIE1	RFPIE1	RFOVIE0	RFFUIE0	RFPIE0	TSMEIE
rw	res			rw	rw	rw	rw	res	rw	rw	rw	rw	rw	rw	rw

位 31：18	保留位，硬件强制为 0
位 17	**SAKIE：**睡眠中断使能 （Sleep interrupt enable） 0：当 SAKIT 位被置’1’时，不产生中断； 1：当 SAKIT 位被置’1’时，产生中断。
位 16	**WKIE：** 唤醒中断使能 （Wakeup interrupt enable） 0：当 WKIT 位被置’1’时，不产生中断； 1：当 WKIT 位被置’1’时，产生中断。
位 15	**ERIE：** 错误中断使能 （Error interrupt enable） 0：当 CAN_ESTS 寄存器有错误挂号时，不产生中断； 1：当 CAN_ESTS 寄存器有错误挂号时，产生中断。
位 14：12	保留位，硬件强制为 0。
位 11	**ERCIE：** 上次错误号中断使能 （Last error code interrupt enable） 0：当检测到错误，硬件设置 ERC[2：0]时，不设置 ERIT 位； 1：当检测到错误，硬件设置 ERC[2：0]时，设置 ERIT 位为’1’。
位 10	**BUIE：** 离线中断使能 （Bus-off interrupt enable） 0：当 BFF 位被置’1’时，不设置 ERIT 位； 1：当 BFF 位被置’1’时，设置 ERIT 位为’1’。
位 9	**ERPIE：** 错误被动中断使能 （Error Passive Interrupt Enable） 0：当 ERPF 位被置’1’时，不设置 ERIT 位； 1：当 ERPF 位被置’1’时，设置 ERIT 位为’1’。
位 8	**ERWIE：** 错误警告中断使能 （Error warning interrupt enable） 0：当 ERWF 位被置’1’时，不设置 ERIT 位； 1：当 ERWF 位被置’1’时，设置 ERIT 位为’1’。
位 7	保留位，硬件强制为 0
位 6	**RFOVIE1：** FIFO 1 溢出中断使能 （FIFO overrun interrupt enable） 0：当 FIFO 1 的 RFOV 位被置’1’时，不产生中断； 1：当 FIFO 1 的 RFOV 位被置’1’时，产生中断。
位 5	**RFFUIE1：** FIFO 1 满中断使能 （FIFO full interrupt enable） 0：当 FIFO 1 的 RFFU 位被置’1’时，不产生中断； 1：当 FIFO 1 的 RFFU 位被置’1’时，产生中断。
位 4	**RFPIE1：** FIFO 1 消息挂号中断使能 （FIFO message pending interrupt enable） 0：当 FIFO 1 的 RFP[1：0]位为非 0 时，不产生中断； 1：当 FIFO 1 的 RFP[1：0]位为非 0 时，产生中断。
位 3	**RFOVIE0：** FIFO 0 溢出中断使能 （FIFO overrun interrupt enable） 0：当 FIFO 0 的 RFOV 位被置’1’时，不产生中断； 1：当 FIFO 0 的 RFOV 位被置’1’时，产生中断。
位 2	**RFFUIE0：** FIFO 0 满中断使能 （FIFO full interrupt enable） 0：当 FIFO 0 的 RFFU 位被置’1’时，不产生中断； 1：当 FIFO 0 的 RFFU 位被置’1’时，产生中断。
位 1	**RFPIE0：** FIFO 0 消息挂号中断使能 （FIFO message pending interrupt enable） 0：当 FIFO 0 的 RFP[1：0]位为非 0 时，不产生中断； 1：当 FIFO 0 的 RFP[1：0]位为非 0 时，产生中断。
位 0	**TSMEIE：** 发送邮箱空中断使能 （Transmit mailbox empty interrupt enable） 0：当 RQCx 位被置’1’时，不产生中断； 1：当 RQCx 位被置’1’时，产生中断。

18.4.2.7 CAN错误状态寄存器 （CAN_ESTS）

地址偏移量：0x18

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REC[7：0]								TEC[7：0]							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留									ERC			保留	BFF	ERPF	ERWF
res									rw	rw	rw	res	r	r	r

位 31：24	**REC[7：0]：** 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。按照 CAN 的标准，当接收出错时，根据出错的条件，该计数器加 1 或加 8；而在每次接收成功后，该计数器减 1，或当该计数器的值大于 127 时，设置它的值为 120。当该计数器的值超过 127 时，CAN 进入错误被动状态。
位 23：16	**TEC[7：0]：** 9 位发送错误计数器的低 8 位 （Least significant byte of the 9-bit transmit error counter） 与上面相似，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	保留位，硬件强制为 0。
位 6：4	**ERC[2：0]：** 上次错误代码 （Last error code） 在检测到 CAN 总线上发生错误时，硬件根据出错情况设置。当报文被正确发送或接收后，硬件清除其值为'0'。 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。 000： 没有错误； 001： 位填充错； 010： 格式（Form）错； 011： 确认（ACK）错； 100： 隐性位错； 101： 显性位错； 110： CRC 错； 111： 由软件设置。
位 3	保留位，硬件强制为 0。
位 2	**BFF：** 离线标志 （Bus-off flag） 当进入离线状态时，硬件对该位置'1'。当发送错误计数器 TEC 溢出，即大于 255 时，CAN 进入离线状态。
位 1	**ERPF：** 错误被动标志 （Error passive flag） 当出错次数达到错误被动的阈值时，硬件对该位置'1'。 （接收错误计数器或发送错误计数器的值>127）。
位 0	**ERWF：** 错误警告标志 （Error warning flag） 当出错次数达到警告的阈值时，硬件对该位置'1'。 （接收错误计数器或发送错误计数器的值≥96）。

18.4.2.8 CAN位时序寄存器 （CAN_BTMG）

地址偏移量：0x1C

复位值：0x0123 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SIL	LBK	保留				SJW[1：0]		保留	BS2[2：0]			BS1[3：0]			
rw	rw	res				rw	rw	res	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留						BRP[11：0]									

res	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31	**SIL：** 静默模式（用于调试） （Silent mode （debug）） 0： 正常状态； 1： 静默模式。
位 30	**LBK：** 环回模式（用于调试） （Loop back mode （debug）） 0： 禁止环回模式； 1： 允许环回模式。
位 29：26	保留位，硬件强制为 0。
位 25：24	**SJW[1：0]：** 重新同步跳跃宽度 （Resynchronization jump width） 为了重新同步，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。 t_{RJW} = t_{CAN} x （SJW[1：0] + 1）。
位 23	保留位，硬件强制为 0。
位 22：20	**BS2[2：0]：** 时间段 2 （Time segment 2） 该位域定义了时间段 2 占用了多少个时间单元 t_{BS2} = t_{CAN} x （BS2[2：0] + 1）。
位 19：16	**BS1[3：0]：** 时间段 1 （Time segment 1） 该位域定义了时间段 1 占用了多少个时间单元 t_{BS1} = t_{CAN} x （BS1[3：0] + 1）
位 15：12	保留位，硬件强制其值为 0。
位 11：0	**BRP[11：0]：** 波特率分频器 （Baud rate prescaler） 该位域定义了时间单元（t_q）的时间长度 t_q = （BRP[11：0]+1） x t_{PCLK}

18.4.3 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 18.3.9 节报文存储。

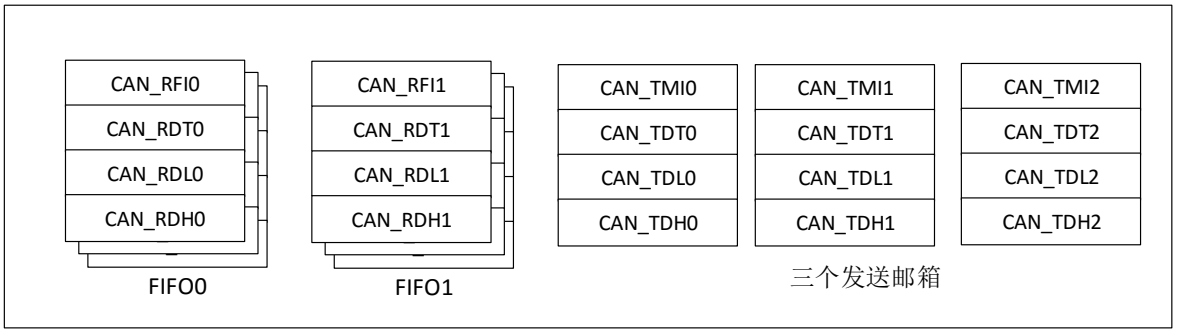
除了下述例外，发送和接收邮箱几乎一样：

- CAN_RDTxR 寄存器的 FID 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_TSTS 寄存器的相应 TSME 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 18-16 发送和接收邮箱



18.4.3.1 发送邮箱标识符寄存器 （CAN_TMIx） （x=0..2）

地址偏移量：0x180，0x190，0x1A0

复位值：0xXXXX XXXX，X=未定义位（除了第 0 位，复位时 TRQ=0）

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。

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

位 31：21	**SID[10：0]/EID[28：18]：**标准标识符或扩展标识符（Standard identifier or extended identifier） 依据 IDT 位的内容，这些位或是标准标识符，或是扩展身份标识的高字节。
位 20：3	**EID[17：0]：** 扩展标识符（Extended identifier） 扩展身份标识的低字节。
位 2	**IDT：** 标识符选择（Identifier extension） 该位决定发送邮箱中报文使用的标识符类型 0：使用标准标识符； 1：使用扩展标识符。
位 1	**RTR：** 远程发送请求（Remote transmission request） 0：数据帧； 1：远程帧。
位 0	**TRQ：** 发送数据请求（Transmit mailbox request） 由软件对其置’1’，来请求发送邮箱的数据。当数据发送完成，邮箱为空时，硬件对其清’0’。

18.4.3.2 发送邮箱数据长度和时间戳寄存器 （CAN_TDTx） （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。

地址偏移量：0x184，0x194，0x1A4

复位值：未定义位

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

位 31：16	**TS[15：0]：** 报文时间戳 （Message time stamp） 该域包含了，在发送该报文 SOF 的时刻，16 位定时器的值。
位 15：9	保留位

位 8	**TMEN：** 发送时间戳 （Transmit global time） 只有在 CAN 处于时间触发通信模式，即 CAN_MCTRL 寄存器的 TTC 位为'1'时，该位才有效。 0：不发送时间戳 TS[15：0]； 1：发送时间戳 TS[15：0]。在长度为 8 的报文中，时间戳 TS[15：0]是最后 2 个发送的字节：TS[7：0]作为第 7 个字节，TS[15：8]为第 6 个字节，它们替换了写入 CAN_TDHxR[31：16]的数据（D6[7：0]和 D7[7：0]）。为了把时间戳的 2 个字节发送出去，DLC 必须编程为 8。
位 7：4	保留位。
位 3：0	**DLC[3：0]：** 发送数据长度 （Data length code） 该域指定了数据报文的数据长度或者远程帧请求的数据长度。1 个报文包含 0 到 8 个字节数据，而这由 DLC 决定。

18.4.3.3 发送邮箱低字节数据寄存器 （CAN_TDLx） （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。

地址偏移量：0x188，0x198，0x1A8

复位值：未定义位

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
D3								D2							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
D1								D0							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31：24	**D3[7：0] ：** 数据字节 3 （Data byte 3） 报文的数据字节 3
位 23：16	**D2[7：0] ：** 数据字节 2 （Data byte 2） 报文的数据字节 2。
位 15：8	**D1[7：0] ：** 数据字节 1 （Data byte 1） 报文的数据字节 1。
位 7：0	**D0[7：0] ：** 数据字节 0 （Data byte 0） 报文的数据字节 0。

18.4.3.4 发送邮箱高字节数据寄存器 （CAN_TDHx） （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。

地址偏移量：0x18C，0x19C，0x1AC

复位值：未定义位

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
D7								D6							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
D5								D4							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31：24	**D7[7：0]** ： 数据字节 7 （Data byte 7） 报文的数据字节 7 *注： 如果 CAN_MCTRL 寄存器的 TTC 位为'1'，且该邮箱的 TMEN 位也为'1'，那么 D7 和 D6 将被 TS 时间戳代替。*
位 23：16	**D6[7：0]** ： 数据字节 6 （Data byte 6） 报文的数据字节 6。
位 15：8	**D5[7：0]** ： 数据字节 5 （Data byte 5） 报文的数据字节 5。
位 7：0	**D4[7：0]** ： 数据字节 4 （Data byte 4） 报文的数据字节 4。

18.4.3.5 接收FIFO邮箱标识符寄存器 （CAN_RFIx） （x=0..1）

地址偏移量：0x1B0，0x1C0

复位值：未定义位

注意： 所有接收邮箱寄存器都是只读的。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SID[10：0]/EID[28：18]														EID[17：16]	
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EID[15：0]													IDT	RTR	保留
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	res

位 31：21	**SID[10：0]/EID[28：18]：** 标准标识符或扩展标识符（Standard identifier or extended identifier） 依据 IDT 位的内容，这些位或是标准标识符，或是扩展身份标识的高字节。
位 20：3	**EID[17：0]：** 扩展标识符 （Extended identifier） 扩展标识符的低字节。
位 2	**IDT：** 标识符选择 （Identifier extension） 该位决定接收邮箱中报文使用的标识符类型 0：使用标准标识符； 1：使用扩展标识符。
位 1	**RTR：** 远程发送请求 （Remote transmission request） 0：数据帧； 1：远程帧。
位 0	保留位。

18.4.3.6 接收FIFO邮箱数据长度和时间戳寄存器 （CAN_RDTx） （x=0..1）

地址偏移量：0x1B4，0x1C4

复位值：未定义

注意： 有接收邮箱寄存器都是只读的。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TS															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FID								保留				DLC			
r	r	r	r	r	r	r	r	res				r	r	r	r

位 31：16	**TS[15：0]：** 报文时间戳 （Message time stamp） 该域包含了，在接收该报文 SOF 的时刻，16 位定时器的值。
位 15：8	**FID[7：0]：** 过滤器匹配序号 （Filter match index） 这里是存在邮箱中的信息传送的过滤器序号。关于标识符过滤的细节，请参考 18.3.8 节中有关过滤器匹配序号。
位 7：4	保留位，硬件强制为 0。
位 3：0	**DLC[3：0]：** 接收数据长度 （Data length code） 该域表明接收数据帧的数据长度（0～8）。对于远程帧请求，数据长度 DLC 恒为 0。

18.4.3.7 接收FIFO邮箱低字节数据寄存器 （CAN_RDLx） （x=0..1）

地址偏移量：0x1B8，0x1C8

复位值：未定义位

注意： 所有接收邮箱寄存器都是只读的。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
D3								D2							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
D1								D0							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

位 31：24	**D3[7：0] ：** 数据字节 3 （Data byte 3） 报文的数据字节 3。
位 23：16	**D2[7：0] ：** 数据字节 2 （Data byte 2） 报文的数据字节 2。
位 15：8	**D1[7：0] ：** 数据字节 1 （Data byte 1） 报文的数据字节 1。
位 7：0	**D0[7：0] ：** 数据字节 0 （Data byte 0） 报文的数据字节 0。 报文包含 0 到 8 个字节数据，且从字节 0 开始。

18.4.3.8 接收FIFO邮箱高字节数据寄存器 （CAN_RDHx） （x=0..1）

地址偏移量：0x1BC，0x1CC

复位值：未定义位

注意： 所有接收邮箱寄存器都是只读的。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
D7								D6							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
D5								D4							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

位 31：24	**D7[7：0] ：** 数据字节 7 （Data byte 7） 报文的数据字节 7
位 23：16	**D6[7：0] ：** 数据字节 6 （Data byte 6） 报文的数据字节 6。
位 15：8	**D5[7：0] ：** 数据字节 5 （Data byte 5） 报文的数据字节 5。
位 7：0	**D4[7：0] ：** 数据字节 4 （Data byte 4） 报文的数据字节 4。

18.4.4 CAN过滤器寄存器

18.4.4.1 CAN 过滤器主控寄存器 （CAN_FM）

地址偏移量：0x200

复位值：0x2A1C 0E01

注意： 该寄存器的非保留位完全由软件控制。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															
res															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留															FINT
res															rw

位 31：1	保留位，强制为复位值。
位 0	**FINT ：** 过滤器初始化模式 （Filter init mode） 针对所有过滤器组的初始化模式设置。 0： 过滤器组工作在正常模式； 1： 过滤器组工作在初始化模式。

18.4.4.2 CAN过滤器模式寄存器 （CAN_FM1）

地址偏移量：0x204

复位值：0x0000 0000

注意： 只有在设置 CAN_FM（FINT=1），使过滤器处于初始化模式下，才能对该寄存器写入。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															
res															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留		FMS 13	FMS 12	FMS 11	FMS 10	FM S9	FM S8	FM S7	FM S6	FM S5	FM S4	FM S3	FM S2	FM S1	FM S0
res		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31：14	保留位，硬件强制为 0
位 13：0	**FMSx ：** 过滤器模式 （Filter mode） 过滤器组 x 的工作模式。 0： 过滤器组 x 的 2 个 32 位寄存器工作在标识符屏蔽位模式； 1： 过滤器组 x 的 2 个 32 位寄存器工作在标识符列表模式。

18.4.4.3 CAN 过滤器位宽寄存器 （CAN_FS1）

地址偏移量：0x20C

复位值：0x0000 0000

注意： 只有在设置 CAN_FM（FINT=1），使过滤器处于初始化模式下，才能对该寄存器写入。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															
res															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留		FBS13	FBS12	FBS11	FBS10	FBS9	FBS8	FBS7	FBS6	FBS5	FBS4	FBS3	FBS2	FBS1	FBS0
res		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31：14	保留位，硬件强制为 0
位 13：0	**FBSx ：** 过滤器位宽设置 过滤器组 x（13～0）的位宽。 0：过滤器位宽为 2 个 16 位； 1：过滤器位宽为单个 32 位。

18.4.4.4 CAN 过滤器FIFO关联寄存器 （CAN_FFA1）

地址偏移量：0x214

复位值：0x0000 0000

注意： 只有在设置 CAN_FM（FINT=1），使过滤器处于初始化模式下，才能对该寄存器写入。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															
res															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留		FAF13	FAF12	FAF11	FAF10	FAF9	FAF8	FAF7	FAF6	FAF5	FAF4	FAF3	FAF2	FAF1	FAF0
res		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31：14	保留位，硬件强制为 0。
位 13：0	**FAFx ：** 过滤器关联 FIFO 设置 报文在通过了某过滤器的过滤后，将被存放到其关联的 FIFO 中。 0：过滤器被关联到 FIFO0； 1：过滤器被关联到 FIFO1。

18.4.4.5 CAN过滤器激活寄存器 （CAN_FA1）

地址偏移量：0x21C

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															
res															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留		FEN13	FEN12	FEN11	FEN10	FEN9	FEN8	FEN7	FEN6	FEN5	FEN4	FEN3	FEN2	FEN1	FEN0
res		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31：14	保留位，硬件强制为 0。
位 13：0	**FENx** ：过滤器激活 （Filter active） 软件对某位设置’1’ 来激活相应的过滤器。只有对 FENx 位清’0’ ，或对 CAN_FM 寄存器的 FINT 位设置’1’后，才能修改相应的过滤器寄存器 x（CAN_FBiRx）。 0：过滤器被禁用； 1：过滤器被激活。

18.4.4.6 CAN 过滤器组i的寄存器x （CAN_FBiRx） （其中i＝0..13；x=1..2）

地址偏移量：0x240h..0x31C

复位值：未定义位

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_FBiR[2：1]组成。

只有在 CAN_FA1 寄存器相应的 FENx 位清’0’，或 CAN_FM 寄存器的 FINT 位为’1’时，才能修改相应的过滤器寄存器。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FD31	FD30	FD29	FD28	FD27	FD26	FD25	FD24	FD23	FD22	FD21	FD20	FD19	FD18	FD17	FD16
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FD15	FD14	FD13	FD12	FD11	FD10	FD9	FD8	FD7	FD6	FD5	FD4	FD3	FD2	FD1	FD0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

在所有的配置情况下：

位 31：0	**FD[31：0]**：过滤器位 （Filter bits） 标识符模式寄存器的每位对应于所期望的标识符的相应位的电平。 0： 期望相应位为显性位； 1： 期望相应位为隐性位。 屏蔽位模式 寄存器的每位指示是否对应的标识符寄存器位一定要与期望的标识符的相应位一致。 0： 不关心，该位不用于比较； 1： 必须匹配，到来的标识符位必须与滤波器对应的标识符寄存器位相一致。

注意： 根据过滤器位宽和模式的不同设置，过滤器组中的两个寄存器的功能也不尽相同。关于过滤器的映射，功能描述和屏蔽寄存器的关联，请参见 18.3.8 节标识符过滤。

屏蔽位模式下的屏蔽/标识符寄存器，跟标识符列表模式下的寄存器位定义相同。

19 外部存储控制器（XMC）

19.1 简介

XMC 模块能够与同步或异步存储器接口，它的主要作用是：

- 将 AHB 传输信号转换到适当的外部设备协议
- 满足访问外部设备的时序要求

所有的外部存储器共享控制器输出的地址、数据和控制信号，每个外部设备可以通过一个唯一的片选信号加以区分。XMC 在任一时刻只访问一个外部设备。

XMC 具有下列主要功能：

- 具复用信号或有额外增加地址锁存功能的静态存储器件，包括：
  - 静态随机存储器（SRAM）
  - NOR 闪存
  - PSRAM（2 个存储器块）
- 一个 NAND 闪存块，支持硬件 ECC 并可检测多达 8K 字节数据
- 支持对同步器件的成组（Burst）访问模式，如 NOR 闪存和 PSRAM
- 8 或 16 位数据总线
- 每一个存储器块都有独立的片选控制
- 每一个存储器块都可以独立配置
- 时序可编程以支持各种不同的器件：
  - 等待周期可编程（多达 15 个周期）
  - 总线恢复周期可编程（多达 15 个周期）
  - 输出使能和写使能延迟可编程（多达 15 周期）
  - 独立的读写时序和协议，可支持宽范围的存储器和时序
- PSRAM 和 SRAM 器件使用的写使能和字节选择输出
- 将 32 位的 AHB 访问请求，转换到连续的 16 位或 8 位，对外部 16 位或 8 位器件的访问
- 具有 2 个字，每个字 32 位宽的写入 FIFO。只缓存 AHB 写突发事件，允许在写入较慢存储器时释放 AHB 进行其它操作。在开始一次新的 XMC 操作前，FIFO 要先被清空。

通常在系统复位或上电时，应该设置好所有定义外部存储器类型和特性的 XMC 寄存器，并保持它们的内容不变；当然，也可以在任何时候改变这些设置。

19.2 主要特点

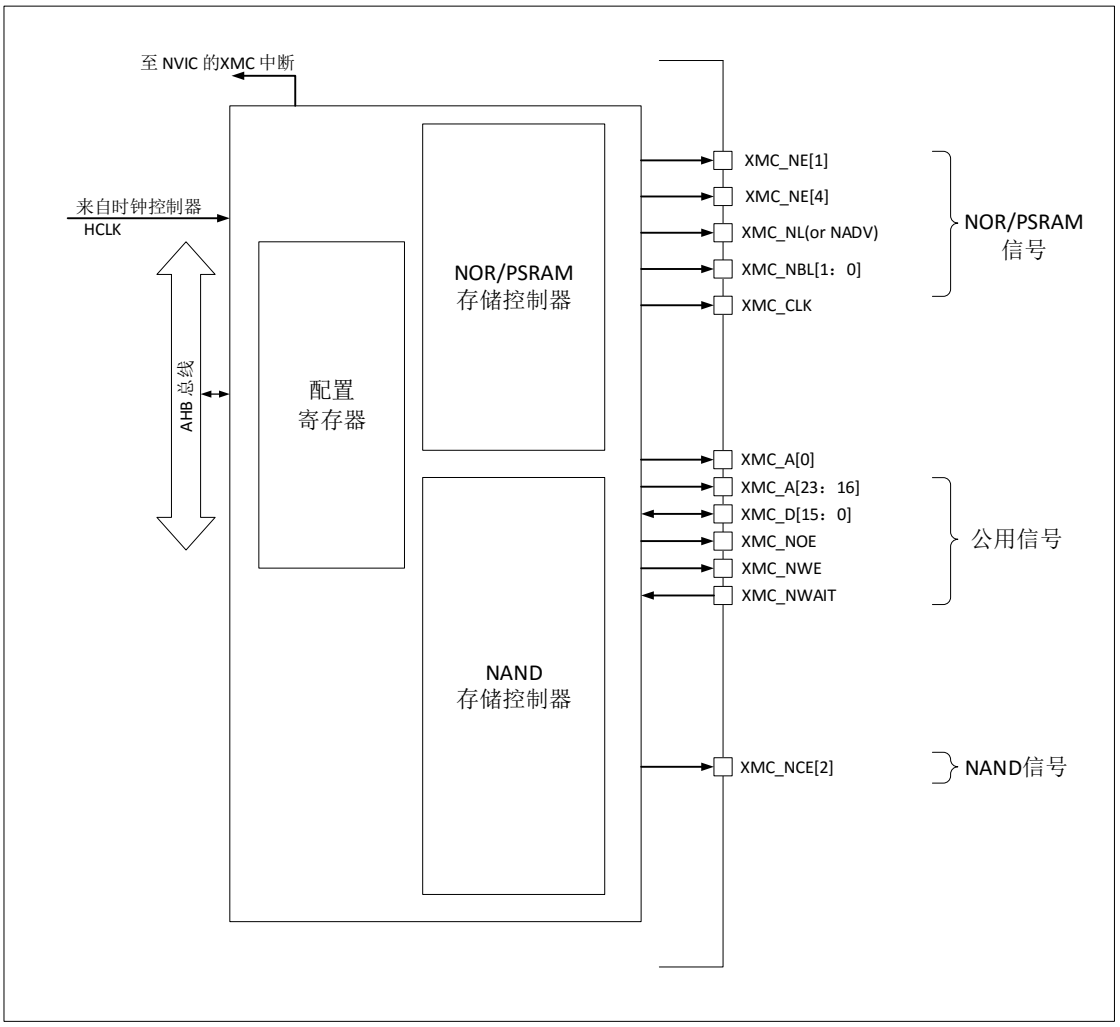
19.2.1 框图

XMC 包含四个主要模块：

- AHB 接口（包含 XMC 配置寄存器）
- NOR 闪存和 PSRAM 控制器
- NAND 闪存控制器
- 外部设备接口

XMC 框图如下：

图 19-1 XMC框图



19.2.2 AHB接口

AHB 接口为内部 CPU 和其它总线控制设备访问外部静态存储器提供了通道。

AHB 操作被转换到外部设备的操作。当选择的外部存储器的数据通道是 16 或 8 位时，在 AHB 上的 32 位数据会被分割成连续的 16 或 8 位的操作。

AHB 时钟（HCLK）是 XMC 的参考时钟。

19.2.3 支持的存储器和操作

一般的操作规则

请求 AHB 操作的数据宽度可以是 8 位、16 位或 32 位，而外部设备则是固定的数据宽度，此时需要保障实现数据传输的一致性。

因此，XMC 执行下述操作规则：

- AHB 操作的数据宽度与存储器数据宽度相同：无数据传输一致性的问题
- AHB 操作的数据宽度大于存储器的数据宽度：此时 XMC 将 AHB 操作分割成几个连续的较小数据宽度的存储器操作，以适应外部设备的数据宽度
- AHB 操作的数据宽度小于存储器的数据宽度：

  依据外部设备的类型，异步的数据传输有可能不一致。

  – 与具有字节选择功能的存储器（SRAM、PSRAM 等）进行异步传输时，XMC 执行读写操作并通过它的字节通道 NBL[1：0]访问正确的数据

  – 与不具有字节选择功能的存储器（NOR 和 16 位 NAND 等）进行异步传输时，即需要对 16 位宽的闪存存储器进行字节访问；显然不能对存储器进行字节模式访问（只允许 16 位的数据传输），因此：

a.不允许进行写操作

b.可以进行读操作（控制器读出完整的 16 位存储器数据，只使用需要的字节）

支持的存储器

此 XMC 型号的地址线仅有 A[23:16]、A[0]，因此 NOR 闪存和 PSRAM 控制器仅支持复用存储器，但是非复用读写功能仍保留着，仍可依照图 19-1 XMC 框图的讯号线来弹性使用。

配置寄存器

XMC 由一组寄存器进行配置。19.4 节详细描述了 NOR 闪存，PSRAM 控制器寄存器，NAND 闪存寄存器。

19.3 功能描述

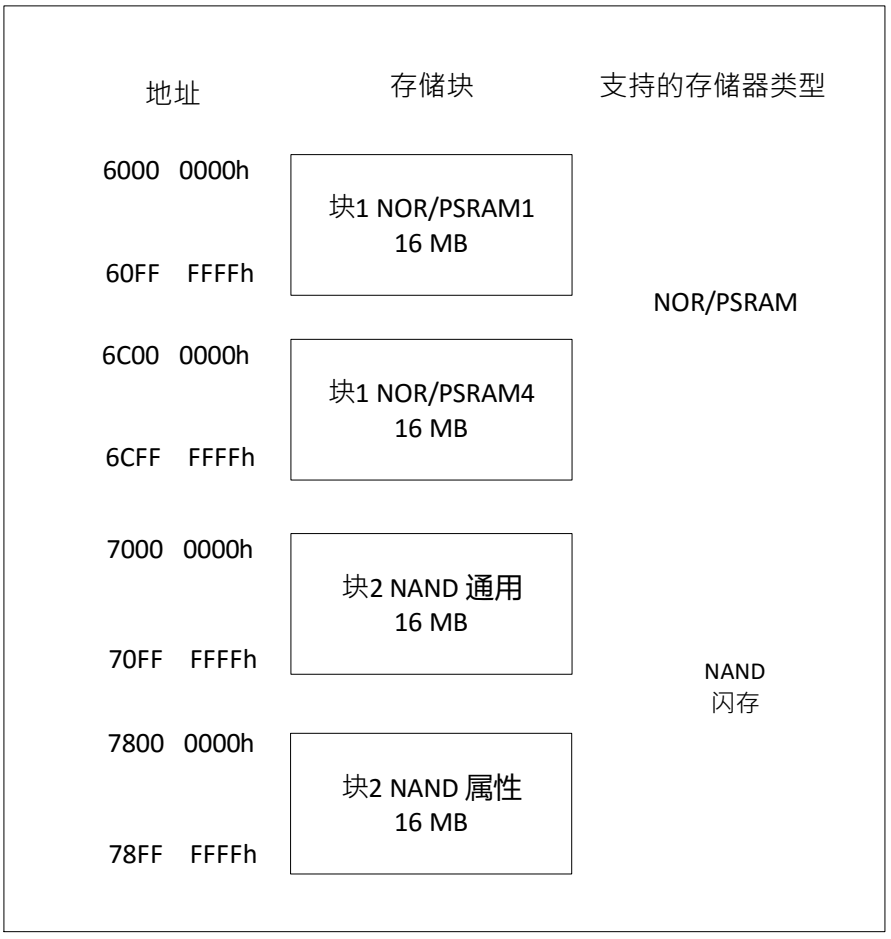
19.3.1 地址映射

从 XMC 的角度看，可以把外部存储器划分为固定大小为 32M 字节的两个存储块，见下图。

- 存储块 1 用于访问最多 2 个 NOR 闪存或 PSRAM 存储设备，这个存储区被划分为 2 个 NOR/PSRAM 区并有 2 个专用的片选
- 存储块 2 用于访问 NAND 闪存设备

每一个存储块上的存储器类型是由用户在配置寄存器中定义的。

图 19-2 XMC存储块



19.3.1.1 NOR和PSRAM地址映射

HADDR[27：26]位用于选择两个存储块之一：

表 19-1 NOR/PSRAM存储块选择

HADDR[27：26]（1）	选择的存储块
00	存储块 1 NOR/PSRAM1
11	存储块 1 NOR/PSRAM4

注意： HADDR 是需要转换到外部存储器的内部 AHB 地址线。

HADDR[23：0]包含外部存储器地址。HADDR 是字节地址，而存储器访问不都是按字节访问，因此接到存储器的地址线依存储器的数据宽度有所不同，如下表：

表 19-2 外部存储器地址

数据宽度（1）	连到存储器的地址线	最大访问存储器空间（位）
8 位	HADDR[23：0]与 XMC_A[23：0]对应相连	16M 字节 x8=128 M 位
16 位	HADDR[23：1]与 XMC_A[22：0]对应相连，HADDR[0]未接	16M 字节/2x16=128 M 位

注意： 对于 16 位宽度的外部存储器，XMC 将在内部使用 HADDR[23：1]产生外部存储器的地址 XMC_A[22：0]。

注意： XMC_A[15:1]未出 pin，在复用模式时低地址 A[15:0]会以 XMC_D[15:0]传送，非复用模式则无法输出 A[15:1]。

NOR 闪存和 PSRAM 的非对齐访问支持

每个 NOR 闪存或 PSRAM 存储器块都可以配置成支持非对齐的数据访问。

在存储器一侧，依据访问的方式是异步或同步，需要考虑两种情况：

- 异步模式：这种情况下，只要每次访问都有准确的地址，完全支持非对齐的数据访问
- 同步模式：这种情况下，XMC 只发出一次地址信号，然后成组的数据传输通过时钟 CLK 顺序进行

某些 NOR 存储器支持线性的非对齐成组访问，固定数目的数据字可以从连续的以 N 为模的地址读取（典型的 N 为 8 或 16，可以通过 NOR 闪存的配置寄存器设置）。此种情况下，可以把存储器的非对齐访问模式设置为与 AHB 相同的模式。

如果存储器的非对齐访问模式不能设置为与 AHB 相同的模式，应该通过 XMC 配置寄存器的相应位禁止非对齐访问，并把非对齐的访问请求分开成两个连续的访问操作。

19.3.1.2 NAND地址映射

这个存储块可以用于 NAND 的操作，被划分为下述访问空间：

表 19-3 存储器映射和时序寄存器

起始地址	结束地址	XMC 存储块	存储空间	时序寄存器
0x78000000	0x78FFFFFF	块 2–NAND 闪存	属性	XMC_BK2TMGATT（0x6C）
0x70000000	0x70FFFFFF		通用	XMC_BK2TMGMEM（0x68）

对于 NAND 闪存存储器，通用和属性空间又可以在低 256K 字节部分划分为 3 个区（见表 19-4）

- 数据区（通用/属性空间的前 64K 字节区域）
- 命令区（通用/属性空间的第 2 个 64K 字节区域）
- 地址区（通用/属性空间的第 2 个 128K 字节区域）

表 19-4 NAND存储块选择

区域名称	HADDR[17：16]	地址范围

地址区	1X	0x020000~0x03FFFF
命令区	01	0x010000~0x01FFFF
数据区	00	0x000000~0x00FFFF

应用软件使用这 3 个区访问 NAND 闪存存储器：

- **发送命令到 NAND 闪存存储器：**软件只需对命令区的任意一个地址写入命令即可
- **指定操作 NAND 闪存存储器的地址：**软件只需对地址区的任意一个地址写入地址值即可。因为一个 NAND 地址可以有 4 或 5 个字节（依实际的存储器容量而定），需要连续地执行对地址区的写才能输出完整的操作地址。
- **读写数据：**软件只需对数据区的任意一个地址写入或读出数据即可

因为 NAND 闪存存储器自动地累加其内部的操作地址，读写数据时没有必要变换数据区的地址，即不必对连续的地址区操作。

19.3.2 NOR闪存/PSRAM控制器

XMC 可以产生适当的信号时序，驱动下述类型的存储器：

- 异步 SRAM
  - 8 位
  - 16 位
  - 32 位
- PSRAM（Cellular RAM）
  - 异步模式
  - 突发模式
- NOR闪存
  - 异步模式或突发模式
  - 复用模式或非复用模式

XMC 对每个存储块输出一个唯一的片选信号 NE[4]、NE[1]，所有其它的（地址、数据和控制）信号则是共享的。

在同步方式中，XMC 向选中的外部设备产生时钟（CLK），该时钟的频率是 HCLK 时钟的整除因子。每个存储块的大小固定为 16M 字节。

每个存储块都有专门的寄存器控制（见 19.4 节）。

可编程的存储器参数包括访问时序（见下表）、是否支持非对齐数据存取和等待周期管理（只针对突发模式下访问 PSRAM 和 NOR 闪存）。

表 19-5 可编程的NOR/PSRAM访问参数

参数	功能	访问方式	单位	最小	最大
地址建立时间	地址建立阶段的时间	异步	AHB 时钟周期（HCLK）	1	16
地址保持时间	地址保持阶段的时间	异步，复用 I/O	AHB 时钟周期（HCLK）	2	16
数据建立时间	数据建立阶段的时间	异步	AHB 时钟周期（HCLK）	2	256
总线恢复时间	总线恢复阶段的时间	异步或同步读	AHB 时钟周期（HCLK）	1	16
时钟分频因子	存储器访问的时钟周期（CLK）与 AHB 时钟周期的比例	同步	AHB 时钟周期（HCLK）	2	16

数据产生时间	突发模式下产生第一个数据所需的时钟数目	同步	存储器时钟周期（CLK）	2	17

19.3.2.1 外部存储器接口信号

表 19-6、表 19-7、表 19-8 列出了与 NOR 闪存和 PSRAM 接口的典型信号。

注意：具有前缀“N”的信号表示低有效信号

NOR 闪存，非复用接口

表 19-6 非复用信号的NOR闪存接口

XMC 信号名称	信号方向	功能
CLK	输出	时钟（同步突发模式使用）
A[23：16]、A[0]	输出	地址总线
D[15：0]	输入/输出	双向数据总线
NE[x]	输出	片选，x=1、4
NOE	输出	输出使能
NWE	输出	写使能
NL（=NADV）	输出	锁存使能（某些 NOR 闪存器件命名该信号为地址有效，NADV）
NWAIT	输入	NOR 闪存要求 XMC 等待的信号

NOR 闪存存储器是按 16 位的字寻址，但是 A[15:1]不输出。

NOR 闪存，复用接口

表 19-7 复用NOR闪存接口

XMC 信号名称	信号方向	功能
CLK	输出	时钟（同步突发模式使用）
A[23：16]	输出	地址总线
AD[15：0]	输入/输出	16 位复用的，双向地址/数据总线
NE[x]	输出	片选，x=1、4
NOE	输出	输出使能
NWE	输出	写使能
NL（=NADV）	输出	锁存使能（某些 NOR 闪存器件命名该信号为地址有效，NADV）
NWAIT	输入	NOR 闪存要求 XMC 等待的信号

NOR 闪存存储器是按 16 位的字寻址，最大容量达 16M 字节（8 条地址线以及 16 条双向地址/数据线）。

PSRAM

表 19-8 非复用信号的PSRAM接口

XMC 信号名称	信号方向	功能
CLK	输出	时钟（同步突发模式使用）
A[23：16]、A[0]	输出	地址总线
D[15：0]	输入/输出	双向数据总线
NE[x]	输出	片选，x=1、4（PSRAM 称其为 NCE（Cellular RAM 即 CRAM））
NOE	输出	输出使能
NWE	输出	写使能
NL（=NADV）	输出	地址有效（存储器信号名称为：NADV）

NWAIT	输入	PSRAM 要求 XMC 等待的信号
NBL[1]	输出	高字节使能（存储器信号名称为：NUB）
NBL[0]	输出	低字节使能（存储器信号名称为：NLB）

PSRAM 存储器是按 16 位的字寻址，但是 A[15:1]不输出。

19.3.2.2 支持的存储器及其操作

下表列出了支持的存储器、访问模式和操作方式，XMC 不支持阴影部分的操作方式。

表 19-9 XMC支持的NOR闪存/PSRAM存储器和操作方式

存储器	模式	读/写	AHB 数据宽度	存储器数据宽度	是否支持	注释
NOR 闪存（总线复用和非总线复用）	异步	读	8	16	支持	
	异步	写	8	16	不支持	
	异步	读	16	16	支持	
	异步	写	16	16	支持	
	异步	读	32	16	支持	分成 2 次 XMC 访问
	异步	写	32	16	支持	分成 2 次 XMC 访问
	异步页	读	-	16	不支持	不支持这种模式
	同步	读	8	16	不支持	
	同步	读	16	16	支持	
	同步	读	32	16	支持	
PSRAM（总线复用和非总线复用）	异步	读	8	16	支持	
	异步	写	8	16	支持	使用字节信号 NBL[1：0]
	异步	读	16	16	支持	
	异步	写	16	16	支持	
	异步	读	32	16	支持	分成 2 次 XMC 访问
	异步	写	32	16	支持	分成 2 次 XMC 访问
	异步页	读	-	16	不支持	不支持这种模式
	同步	读	8	16	不支持	
	同步	读	16	16	支持	
	同步	读	32	16	支持	
	同步	写	8	16	支持	使用字节信号 NBL[1：0]
	同步	写	16/32	16	支持	
SRAM	异步	读	8/16/32	16	支持	使用字节信号 NBL[1：0]
	异步	写	8/16/32	16	支持	使用字节信号 NBL[1：0]

19.3.2.3 时序规则

信号同步

- 所有的控制器输出信号在内部时钟（HCLK）的上升沿变化
- 在同步模式（PSRAM 读或写）下，输出的数据在存储器时钟（CLK）的下降沿变化

19.3.2.4 NOR闪存和PSRAM控制器时序图

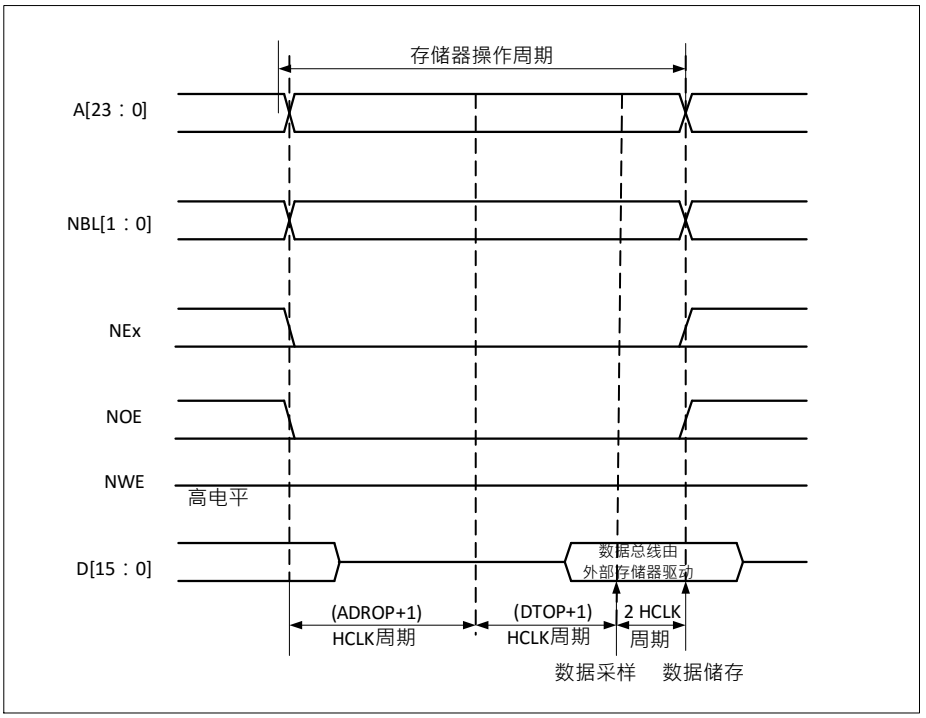
异步静态存储器（NOR 闪存,PSRAM 和 SRAM）

- 所有信号由内部时钟 HCLK 保持同步，但该时钟不会输出到存储器

- XMC 始终在信号 NOE 失效前对数据线采样，这样能够保证符合存储器的数据保持时序（片选失效至数据失效的间隔，通常最小为 0ns）
- 当设置了扩展模式，可以在读和写时混合使用模式 A、B、C 和 D（例如，允许以模式 A 进行读，而以模式 B 进行写）

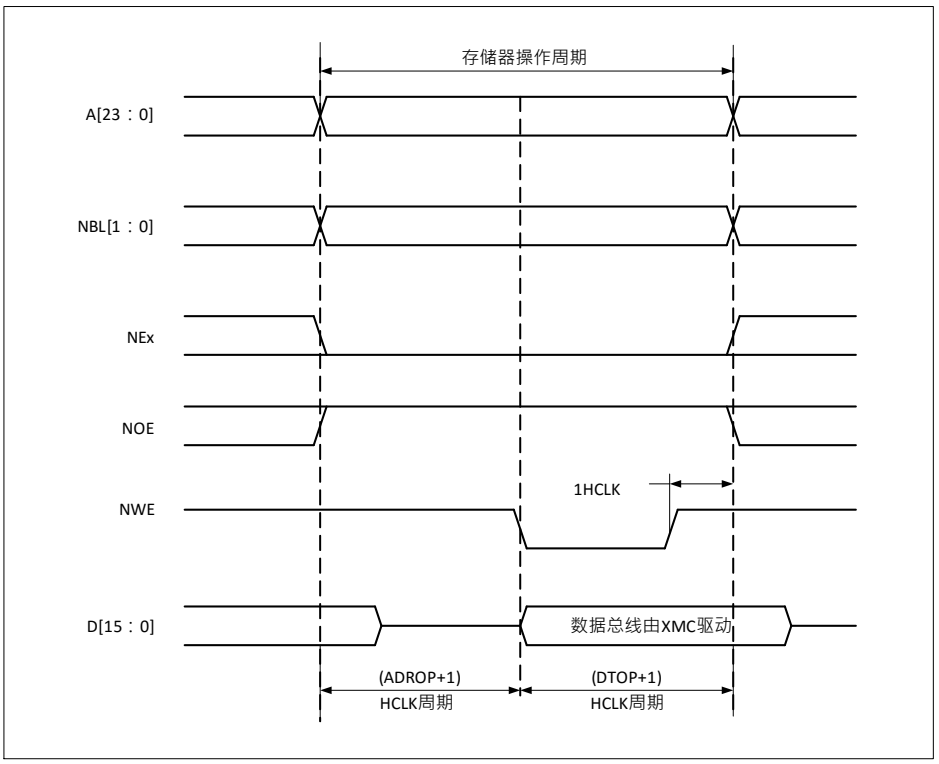
模式 1——SRAM/PSRAM（CRAM）

图 19-3 模式1读操作



注意: A[15:1]不输出。

图 19-4 模式1写操作



注意: A[15:1]不输出。

在写操作的最后一个 HCLK 周期可以保证 NWE 上升沿后地址和数据的保持时间，因为存在这个 HCLK

周期，DTOP 的数值必须大于 0（DTOP>0）。

表 19-10 XMC_BK1CTRLx位域（模式1）

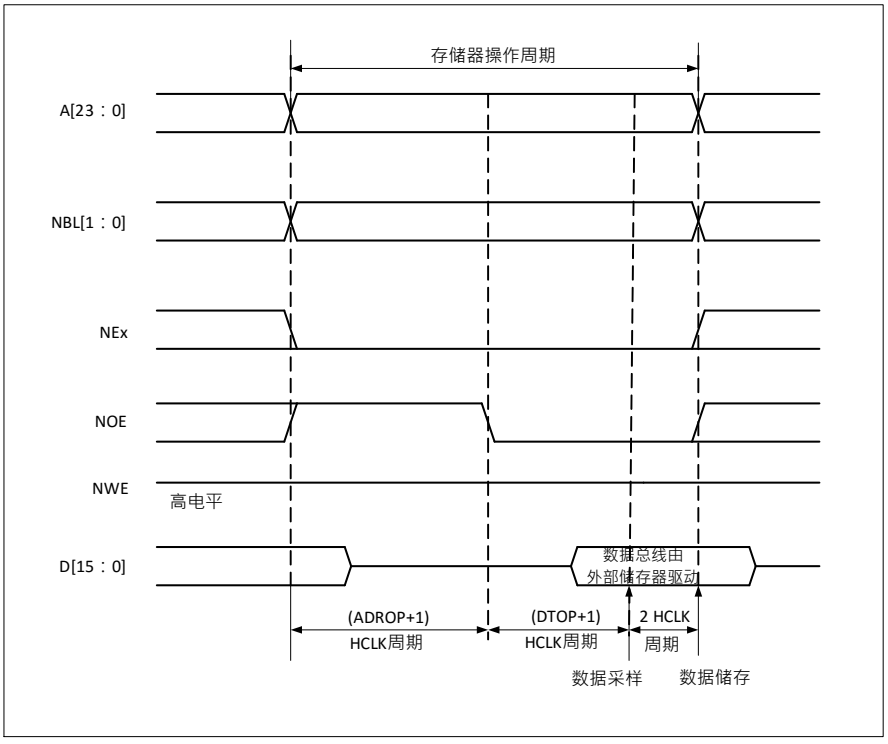
位编号	位名称	设置的数值
31-16		0x0000
15	WAITASYNC	如果存储器支持此功能，设置成 1；否则保持 0。
14-10		0x0
9	WAITALV	仅当位 15 为'1'时有意义。
8	BURSTEN	0x0
7		-
6	NOREN	-
5-4	BUSTYPE	需要时设置
3-2	DEV	需要时设置，不包含 0x2（NOR 闪存）
1	MUXEN	0x0
0	EN	0x1

表 19-11 XMC_BK1TMGx位域（模式1）

位编号	位名称	设置的数值
31-20		0x0000
19-16	INTVLOP	NEx 从高到低的时间（INTVLOP HCLK）
15-8	DTOP	操作的第 2 个阶段的长度，写操作为（DTOP+1 个 HCLK 周期），读操作为（DTOP+3 个 HCLK 周期）。这个域不能为 0，至少为 1。
7-4		0x0
3-0	ADROP	操作的第 1 个阶段的长度（ADROP+1 个 HCLK 周期）。

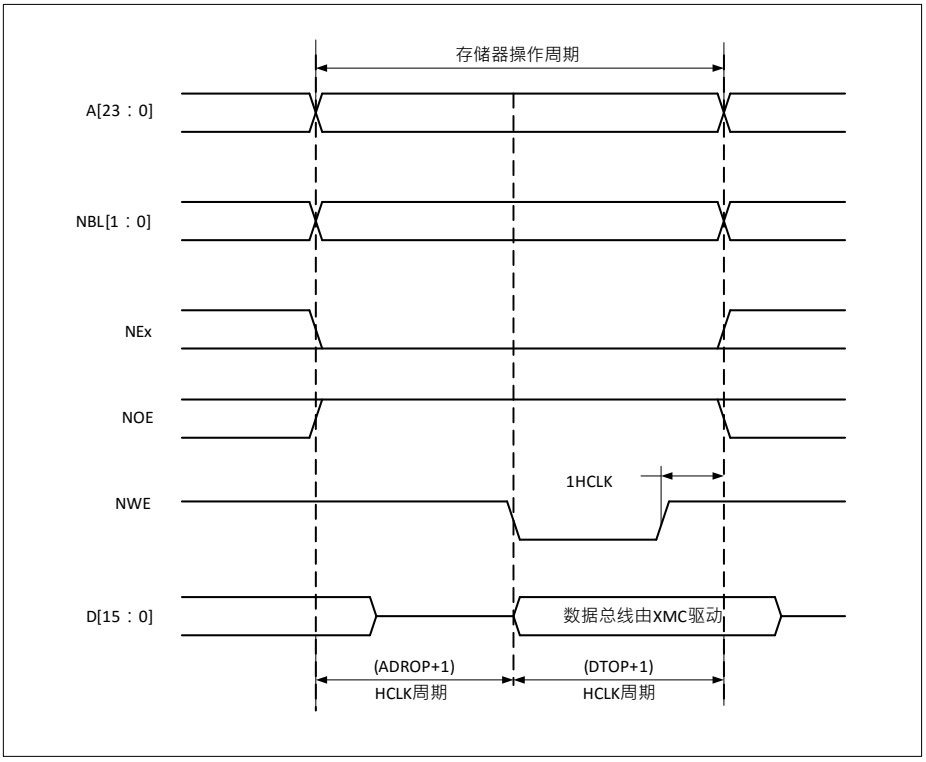
模式 A——SRAM/PSRAM（CRAM） OE 翻转

图 19-5 模式A读操作



注意: A[15:1]不输出。

图 19-6 模式A写操作



注意: A[15:1]不输出。

模式 A 与模式 1 的区别是 NOE 的变化和相互独立的读写时序。

表 19-12 XMC_BK1CTRLx位域（模式A）

位编号	位名称	设置的数值
31-16		0x0000
15	WAITASYNC	如果存储器支持此功能，设置成 1；否则保持 0。
14	TMGWREN	0x1
13-10		0x0
9	WAITALV	仅当位 15 为'1'时有意义。
8	BURSTEN	0x0
7		-
6	NOREN	-
5-4	BUSTYPE	需要时设置
3-2	DEV	需要时设置，不包含 0x2（NOR 闪存）
1	MUXEN	0x0
0	EN	0x1

表 19-13 XMC_BK1TMGx位域（模式A）

位编号	位名称	设置的数值
31-30		0x0
29-28	MODE	0x0

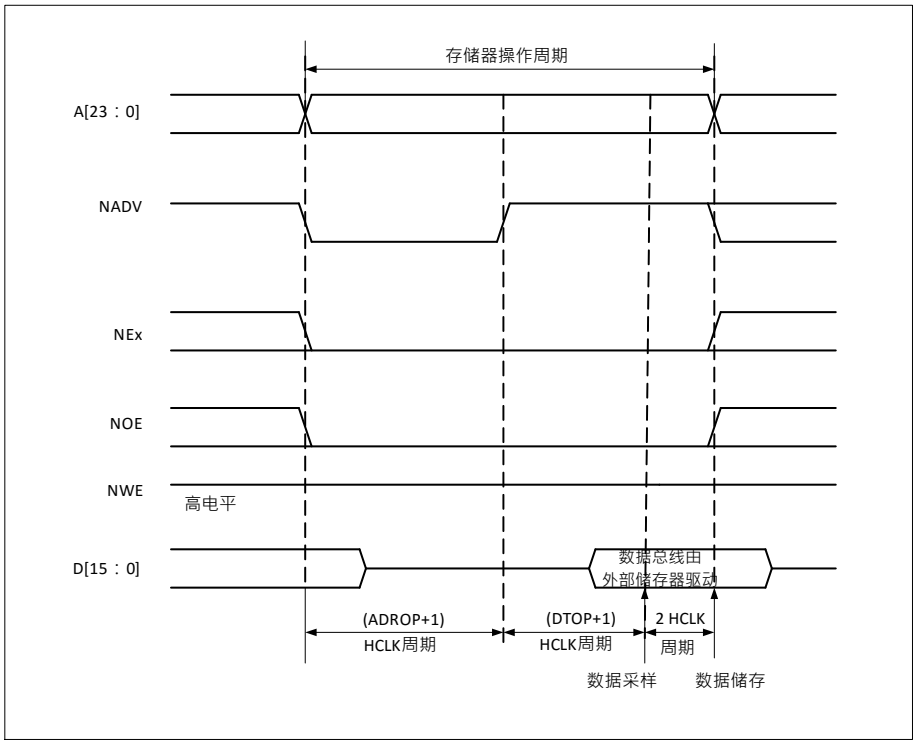
27-20		0x000
19-16	INTVLOP	NEx 从高到低的时间（INTVLOP HCLK）
15-8	DTOP	读操作的第 2 个阶段的长度（DTOP+3 个 HCLK 周期）。这个域不能为 0（至少为 1）。
7-4		0x0
3-0	ADROP	读操作的第 1 个阶段的长度（ADROP+1 个 HCLK 周期）。

表 19-14 XMC_BK1TMGWRx位域（模式A）

位编号	位名称	设置的数值
31-30		0x0
29-28	MODE	0x0
27-20		0x000
19-16	INTVLOP	NEx 从高到低的时间（INTVLOP HCLK）
15-8	DTOP	操作的第 2 个阶段的长度，写操作为（DTOP+1 个 HCLK 周期），读操作为（DTOP+3 个 HCLK 周期）。这个域不能为 0，至少为 1。
7-4		0x0
3-0	ADROP	写操作的第 1 个阶段的长度（ADROP+1 个 HCLK 周期）。

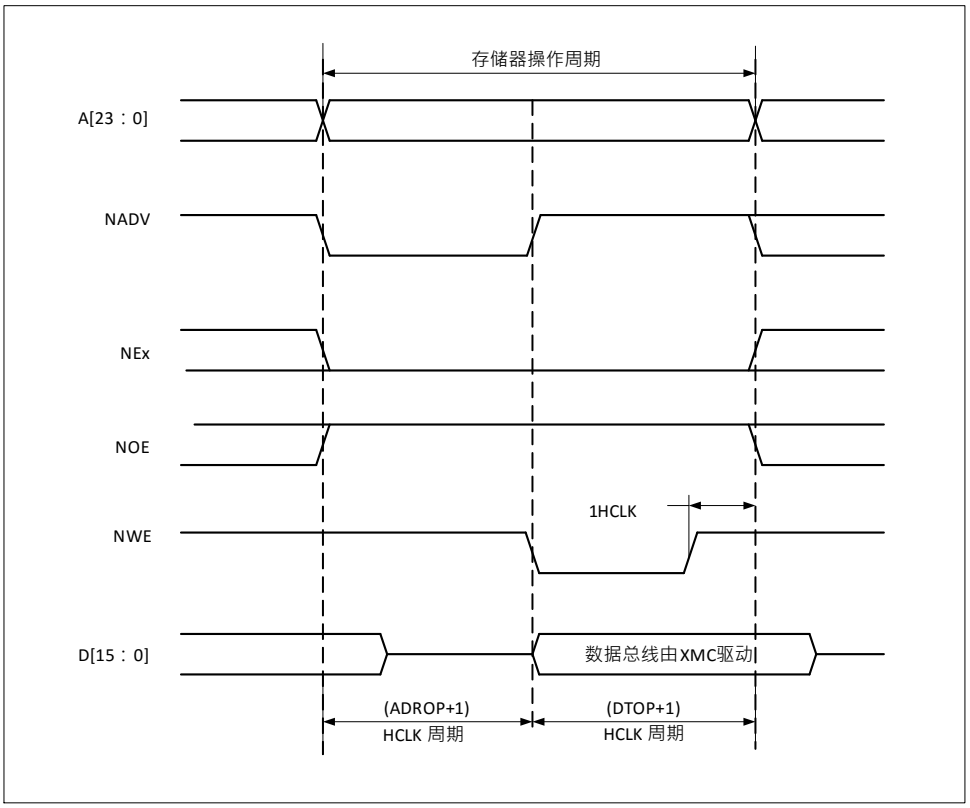
模式 2/B——NOR 闪存

图 19-7 模式2/B读操作



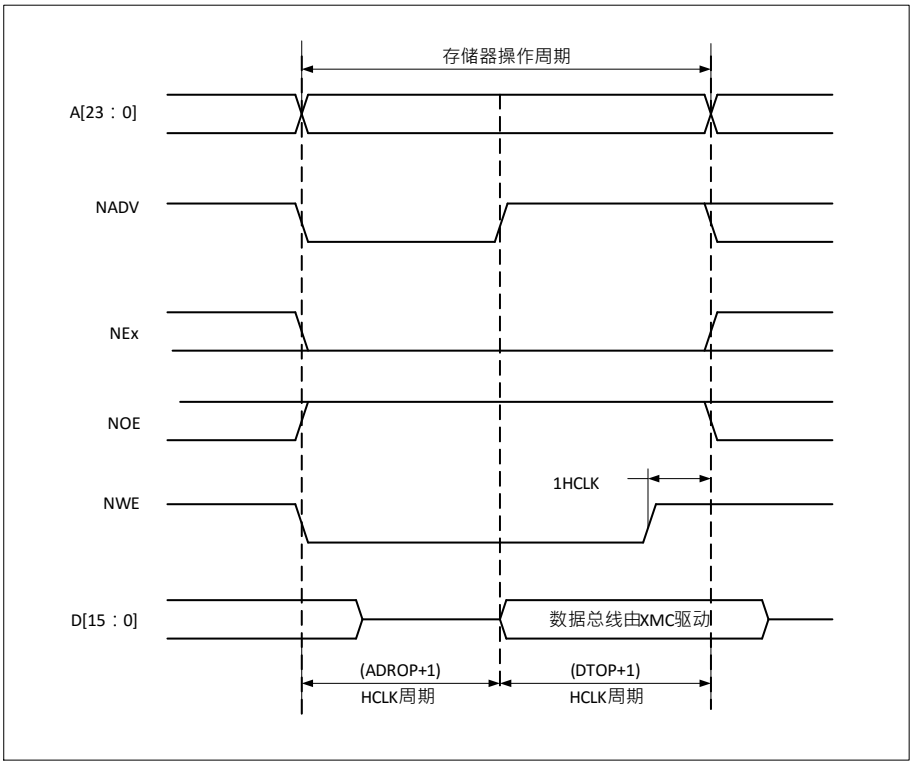
注意: A[15:1]不输出。

图 19-8 模式2写操作



注意: A[15:1]不输出。

图 19-9 模式B写操作



注意: A[15:1]不输出。

模式 2/B 与模式 1 相比较，不同的是 NADV 的变化，且在扩展模式下（模式 B）读写时序相互独立。

表 19-15 XMC_BK1CTRLx位域（模式2/B）

位编号	位名称	设置的数值
31-16		0x0000
15	WAITASYNC	如果存储器支持此功能，设置成 1；否则保持 0。
14	TMGWREN	模式 B：0x1；模式 2：0x0
13-10		0x0
9	WAITALV	仅当位 15 为'1'时有意义。
8	BURSTEN	0x0
7		-
6	NOREN	0x1
5-4	BUSTYPE	需要时设置
3-2	DEV	0x2（NOR 闪存）
1	MUXEN	0x0
0	EN	0x1

表 19-16 XMC_BK1TMGx位域（模式2/B）

位编号	位名称	设置的数值
31-30		0x0
29-28	MODE	0x1
27-20		0x000
19-16	INTVLOP	NEx 从高到低的时间（INTVLOP HCLK）
15-8	DTOP	读操作的第 2 个阶段的长度（DTOP+3 个 HCLK 周期）。这个域不能为 0（至少为 1）。
7-4		0x0
3-0	ADROP	读操作的第 1 个阶段的长度（ADROP+1 个 HCLK 周期）。

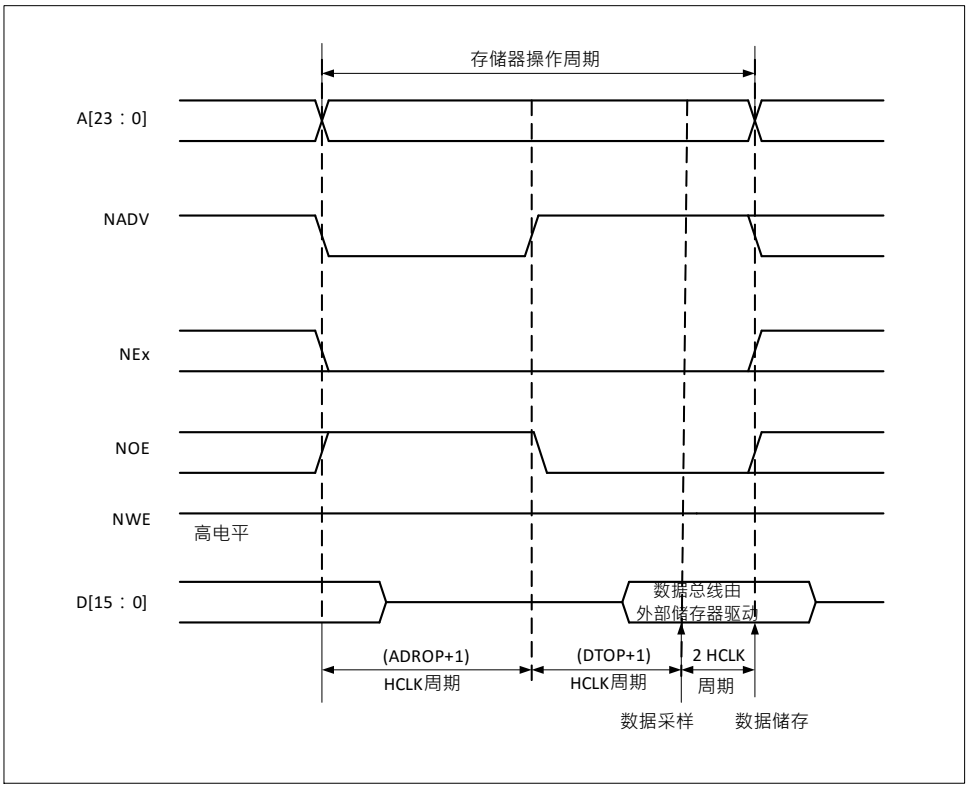
表 19-17 XMC_BK1TMGWRx位域（模式2/B）

位编号	位名称	设置的数值
31-30		0x0
29-28	MODE	0x1
27-20		0x000
19-16	INTVLOP	NEx 从高到低的时间（INTVLOP HCLK）
15-8	DTOP	操作的第 2 个阶段的长度，写操作为（DTOP+1 个 HCLK 周期），读操作为（DTOP+3 个 HCLK 周期）。这个域不能为 0，至少为 1。
7-4		0x0
3-0	ADROP	写操作的第 1 个阶段的长度（ADROP+1 个 HCLK 周期）。

注意： 只有当设置了扩展模式时（模式 B），XMC_BK1TMGWRx 才有效，否则该寄存器的内容不起作用。

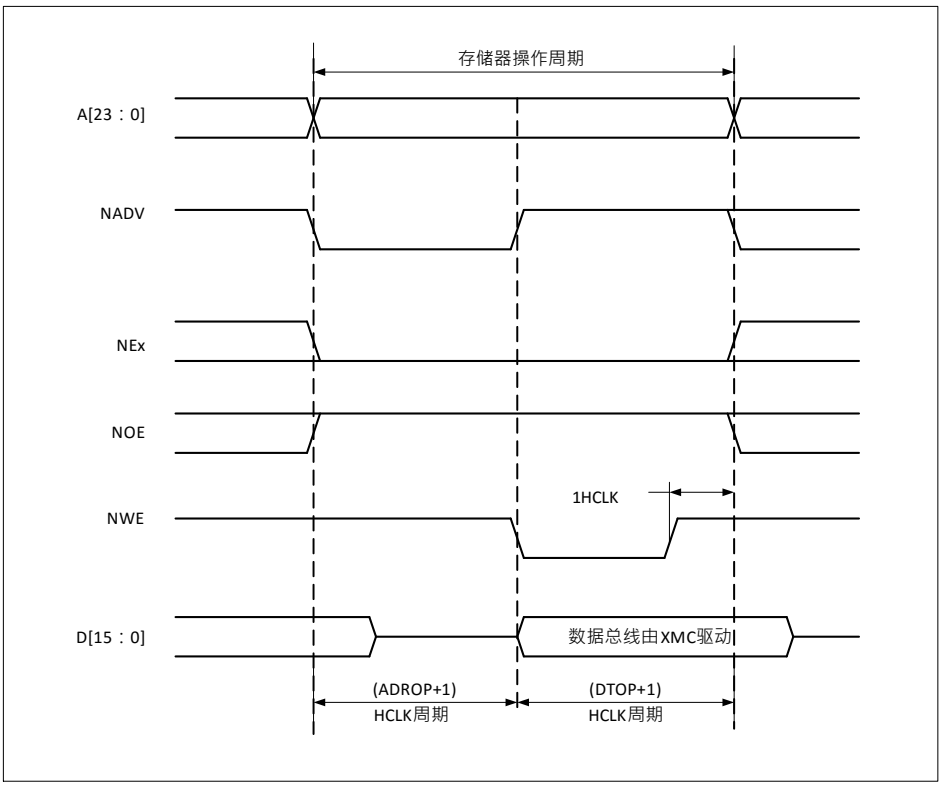
模式 C——NOR 闪存-OE 翻转

图 19-10 模式C读操作



注意: A[15:1]不输出。

图 19-11 模式C写操作



注意: A[15:1]不输出。

模式 C 与模式 1 不同的是，NOE 和 NADV 的翻转变化，以及独立的读写时序。

表 19-18 XMC_BK1CTRLx位域（模式C）

位编号	位名称	设置的数值

31-16		0x0000
15	WAITASYNC	如果存储器支持此功能，设置成 1；否则保持 0。
14	TMGWREN	0x1
13-10		0x0
9	WAITALV	仅当位 15 为'1'时有意义。
8	BURSTEN	0x0
7		-
6	NOREN	0x1
5-4	BUSTYPE	需要时设置
3-2	DEV	0x2（NOR 闪存）
1	MUXEN	0x0
0	EN	0x1

表 19-19 XMC_BK1TMGx位域（模式C）

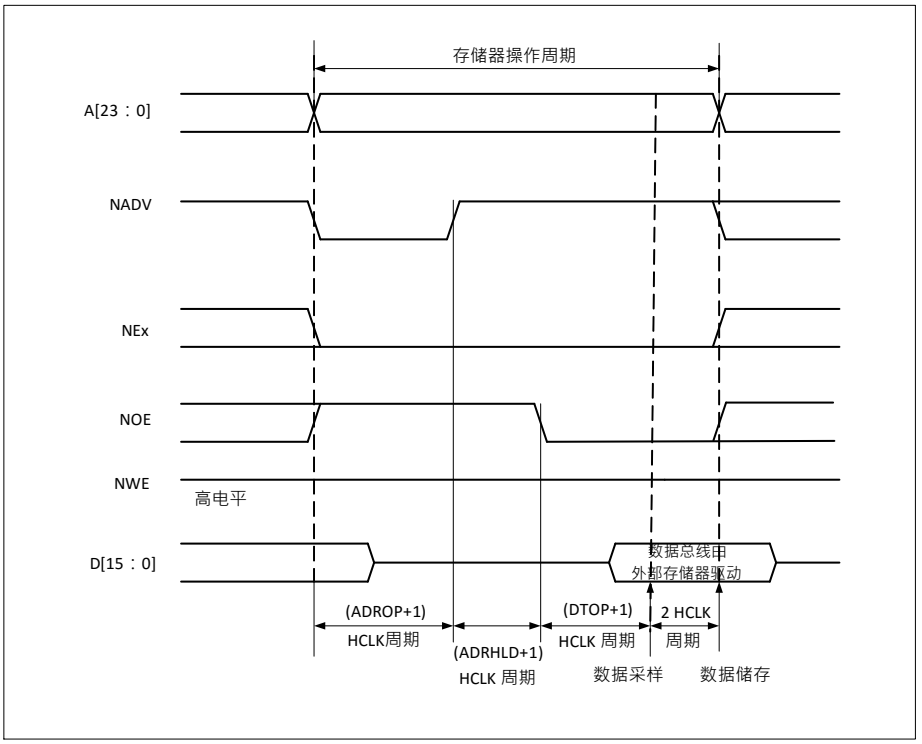
位编号	位名称	设置的数值
31-30		0x0
29-28	MODE	0x2
27-20		0x000
19-16	INTVLOP	NEx 从高到低的时间（INTVLOP HCLK）
15-8	DTOP	读操作的第 2 个阶段的长度（DTOP+3 个 HCLK 周期）。这个域不能为 0（至少为 1）。
7-4		0x0
3-0	ADROP	读操作的第 1 个阶段的长度（ADROP+1 个 HCLK 周期）。

表 19-20 XMC_BK1TMGWRx位域（模式C）

位编号	位名称	设置的数值
31-30		0x0
29-28	MODE	0x2
27-20		0x000
19-16	INTVLOP	NEx 从高到低的时间（INTVLOP HCLK）
15-8	DTOP	操作的第 2 个阶段的长度，写操作为（DTOP+1 个 HCLK 周期），读操作为（DTOP+3 个 HCLK 周期）。这个域不能为 0，至少为 1。
7-4		0x0
3-0	ADROP	写操作的第 1 个阶段的长度（ADROP+1 个 HCLK 周期）。

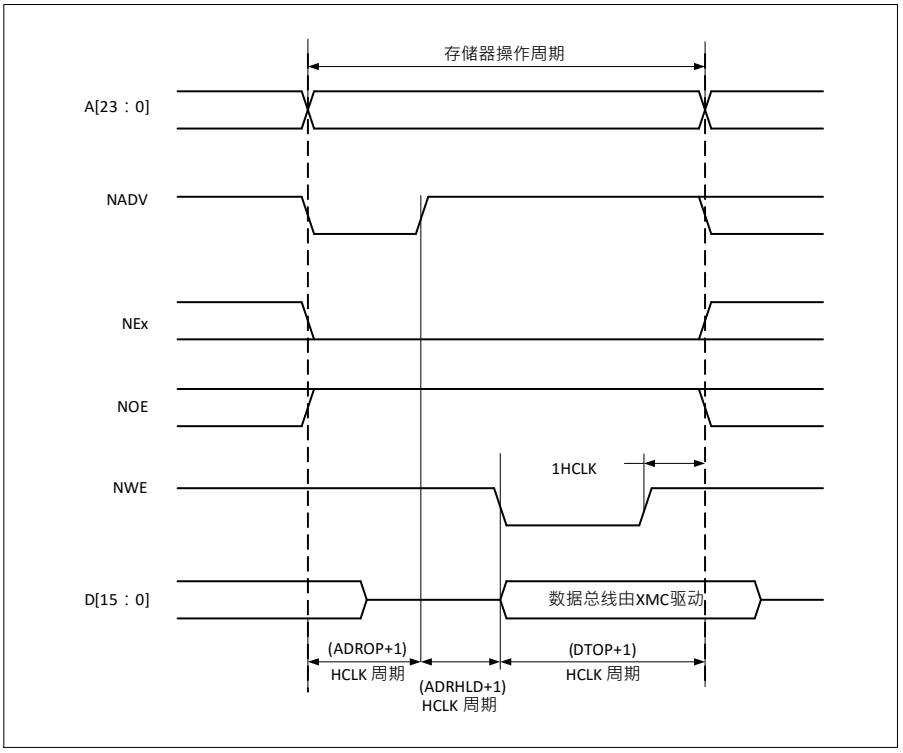
模式 D——带地址扩展的异步操作

图 19-12 模式D读操作



注意: A[15:1]不输出。

图 19-13 模式D写操作



注意: A[15:1]不输出。

模式 D 与模式 1 不同的是 NADV 的翻转变化，NOE 的翻转出现在 NADV 翻转之后，并且具有独立的读写时序。

表 19-21 XMC_BK1CTRLx位域（模式D）

位编号	位名称	设置的数值
31-16		0x0000
15	WAITASYNC	如果存储器支持此功能，设置成 1；否则保持 0。
14	TMGWREN	0x1
13-10		0x0
9	WAITALV	仅当位 15 为'1'时有意义。
8	BURSTEN	0x0
7		-
6	NOREN	根据存储器设置
5-4	BUSTYPE	需要时设置
3-2	DEV	需要时设置
1	MUXEN	0x0
0	EN	0x1

表 19-22 XMC_BK1TMGx位域（模式D）

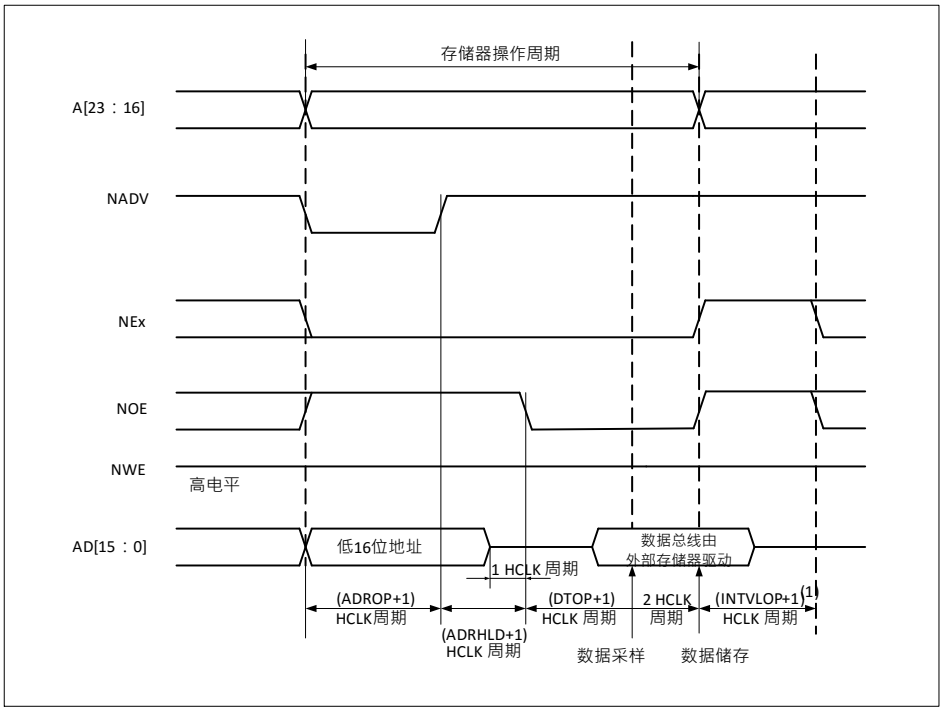
位编号	位名称	设置的数值
31-30		0x0
29-28	MODE	0x3
27-20		0x000
19-16	INTVLOP	NEx 从高到低的时间（INTVLOP HCLK）
15-8	DTOP	读操作的第 2 个阶段的长度（DTOP+3 个 HCLK 周期）。这个域不能为 0（至少为 1）。
7-4	ADRHLD	读操作的中间阶段的长度（ADRHLD+1 个 HCLK 周期）。
3-0	ADROP	读操作的第 1 个阶段的长度（ADROP+1 个 HCLK 周期）。

表 19-23 XMC_BK1TMGWRx位域（模式D）

位编号	位名称	设置的数值
31-30		0x0
29-28	MODE	0x3
27-20		0x000
19-16	INTVLOP	NEx 从高到低的时间（INTVLOP HCLK）
15-8	DTOP	写操作的第 2 个阶段的长度（DTOP+1 个 HCLK 周期）。这个域不能为 0（至少为 1）。
7-4	ADRHLD	写操作的中间阶段的长度（ADRHLD+1 个 HCLK 周期）。
3-0	ADROP	写操作的第 1 个阶段的长度（ADROP+1 个 HCLK 周期）。

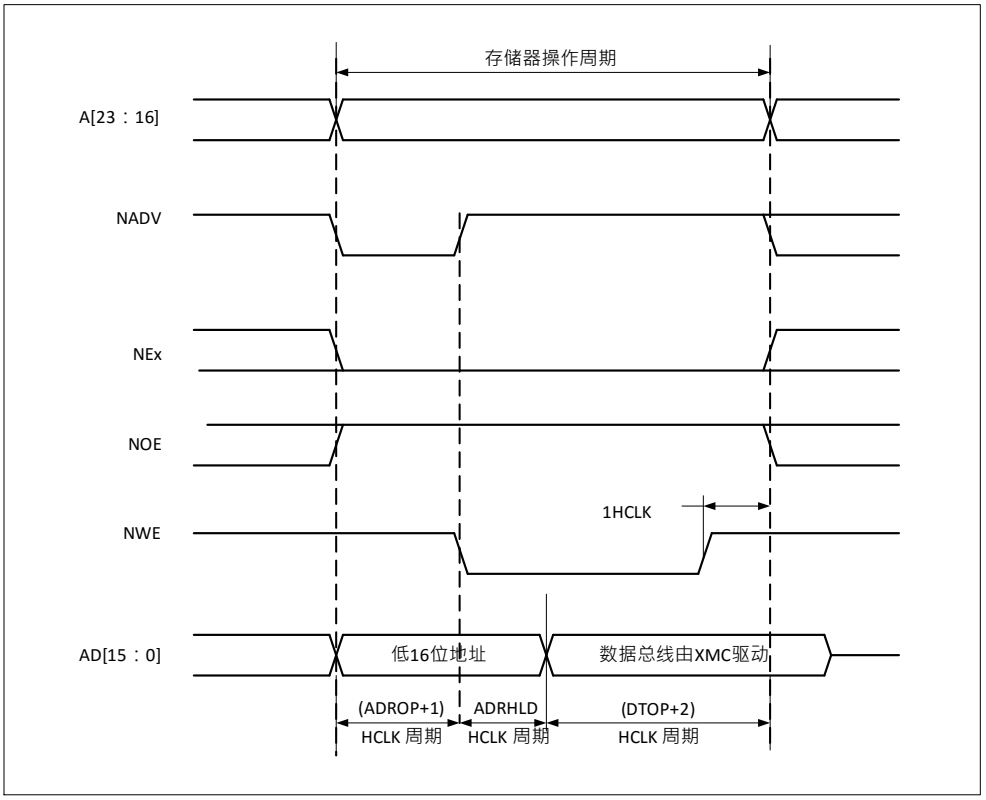
复用模式——地址/数据复用的 NOR 闪存异步操作

图 19-14 复用读操作



注意： 总线恢复延迟（INTVLOP+1）与连续 2 次读操作之间在内部产生的延迟有部分重叠，因此 INTVLOP≤5 时将不影响输出时序。

图 19-15 复用写操作



复用模式与模式 D 不同的是地址的低 16 位出现在数据总线上。

表 19-24 XMC_BK1CTRLx位域（复用模式）

位编号	位名称	设置的数值
31-16		0x0000
15	WAITASYNC	如果存储器支持此功能，设置成 1；否则保持 0。
14	TMGWREN	0x0
13-10		0x0
9	WAITALV	仅当位 15 为'1'时有意义。
8	BURSTEN	0x0
7		-
6	NOREN	0x1
5-4	BUSTYPE	需要时设置
3-2	DEV	0x2（NOR 闪存）
1	MUXEN	0x1
0	EN	0x1

表 19-25 XMC_BK1TMGx位域（复用模式）

位编号	位名称	设置的数值
31-30		0x0
29-20		0x0
19-16	INTVLOP	操作的最后阶段的长度（INTVLOP+1 个 HCLK 周期）。
15-8	DTOP	操作的第 2 个阶段的长度，读操作为（DTOP+3 个 HCLK 周期），写操作为（DTOP+1 个 HCLK 周期）。这个域不能为 0（至少为 1）。
7-4	ADRHLD	操作的中间阶段的长度（ADRHLD+1 个 HCLK 周期）。这个域不能为 0（至少为 1）。
3-0	ADROP	操作的第 1 个阶段的长度（ADROP+1 个 HCLK 周期）。

19.3.2.5 同步的成组读

根据参数 CLKPSC 的不同数值，HCLK 的周期是存储器时钟 CLK 的整数倍。

NOR 闪存存储器有一个从 NADV 有效至 CLK 变高的最小时间限制，为了满足这个限制，在同步访问（NADV 有效之前）的第一个内部时钟周期中，XMC 不会输出时钟到存储器。这样可以保证存储器时钟的上升沿产生于 NADV 低脉冲的中间。

数据延时与 NOR 闪存的延时

数据延时是指在采样数据之前需等待的周期数目，DTSTBL 数值必须与 NOR 闪存配置寄存器中定义的数值相符合。XMC 不把 NADV 为低时的时钟周期包含在数据延时这个参数中。

注意：有些 NOR 闪存把 NADV 为低时的时钟周期包含在数据延时这个参数中，因此 NOR 闪存延时与 XMC 的 DTSTBL 参数的关系可以是：

- *NOR 闪存延时=DTSTBL+2；或*
- *NOR 闪存延时=DTSTBL+3*

有些新出的存储器会在数据延时阶段产生一个 NWAIT 信号，这种情况下可以设置 DTSTBL 为其最小值。XMC 会对 NWAIT 信号采样并等待足够长的时间直到数据有效，在 XMC 检测到存储器结束了延时阶段后，读取正确的数据。

另外一些存储器不在数据延时阶段输出 NWAIT 信号，这时 XMC 和存储器端的数据延迟时间必须设置为相同的数值，否则将得不到正确的数据，或在存储器访问的初始阶段会有数据丢失。

单次成组传输

当选中的存储器块配置为同步成组模式，如果仅需要进行一次 AHB 单次成组传输，如果 AHB 需要传输 16 位数据，则 XMC 会执行一次长度为 1 的成组传输；如果 AHB 需要传输 32 位数据，则 XMC 会分成 2 次 16 位传输，执行一次长度为 2 的成组传输；最后一个数据传输完毕时撤消片选信号。

显然，从效率上讲这种传输方式（与异步读相比）不是最有效的；但是一次随机的异步访问需要重新配置存储器访问模式，这同样需要较长时间。

等待管理

对于同步的 NOR 闪存成组访问，在预置的延迟时间（DTSTBL+2 个 CLK 时钟周期）之后，需检测 NWAIT 信号。

如果检测到 NWAIT 为有效电平时（当 WAITALV=0 时有效电平为低，WAITALV=1 时有效电平为高），XMC 将插入等待周期直到 NWAIT 变为无效电平（当 WAITALV=0 时无效电平为高，WAITALV=1 时无效电平为低）。

当 NWAIT 变为无效时，XMC 认为数据已经有效（WAITCFG=1），或数据将在下一个时钟边沿有效（WAITCFG=0）。

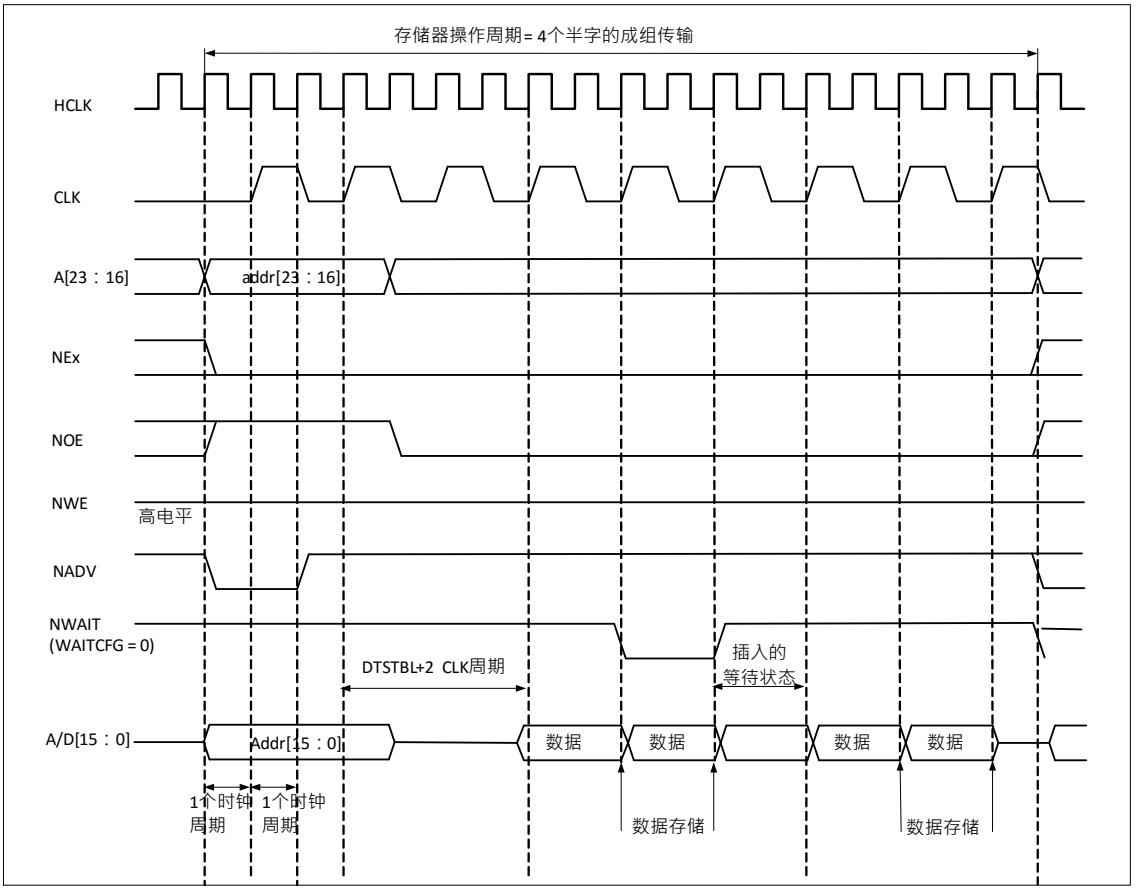
在 NWAIT 信号控制的等待状态插入期间，控制器会连续地向存储器发送时钟脉冲、保持片选信号和输出有效信号，同时忽略无效的数据信号。

在成组传输模式下，NOR 闪存的 NWAIT 信号有 2 种时序配置：

- 闪存存储器在等待状态之前的一个数据周期插入 NWAIT 信号（复位后的默认设置）
- 闪存存储器在等待状态期间插入 NWAIT 信号

通过配置 XMC_BK1CTRLx 寄存器中的 WAITCFG 位，XMC 在每个片选上都支持这 2 种 NOR 闪存的等待状态配置。

图 19-16 同步的总线复用读模式 – NOR,PSRAM（CRAM）



注意： NBL 信号没有显示在图中，对于 NOR 闪存 NBL 应该为高；对于 PSRAM（CRAM），NBL 应该为低。

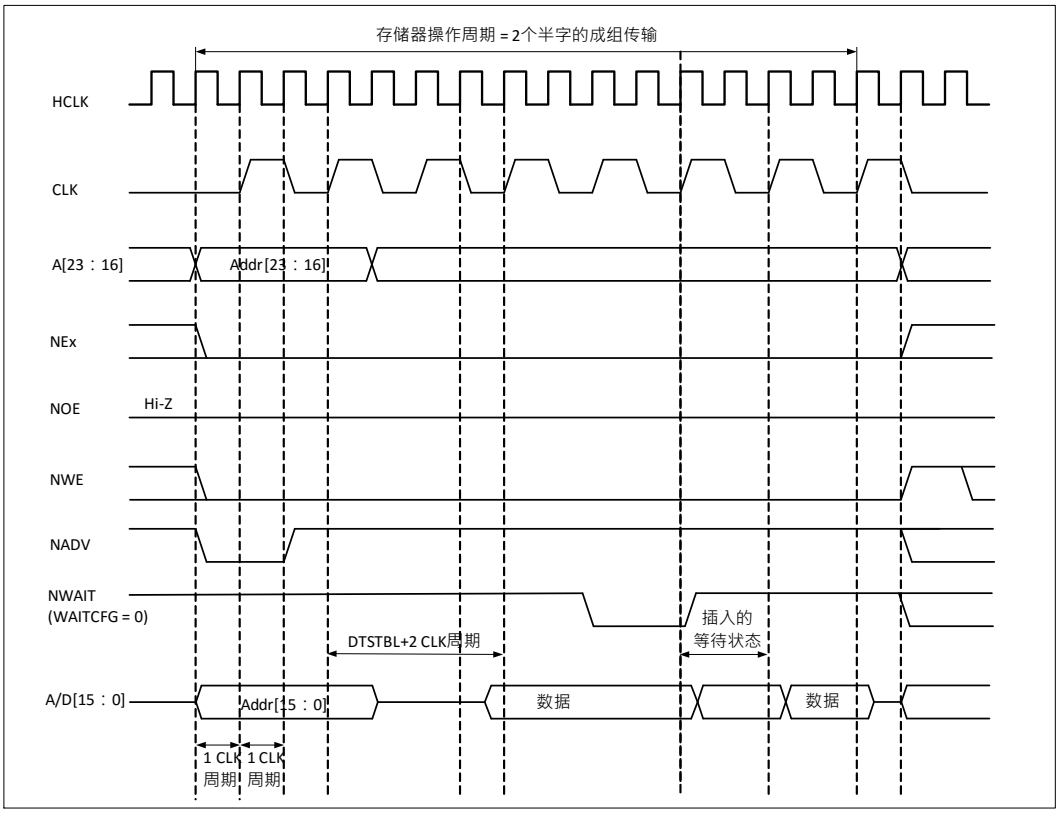
表 19-26 XMC_BK1CTRLx位域（同步模式）

位编号	位名称	设置的数值
31-20		0x0000
19	BURSTWRSYN	对同步读模式不起作用
18-16	PGSIZE	根据需要设置（0x1 为 CRAM 1.5）
14	TMGWREN	0x0
15		0x0
13	WAITEN	如果存储器支持此功能，设置成 1；否则保持 0
12	WREN	对同步读模式不起作用
11	WAITCFG	根据存储器特性设置
10	BRSTSPLTEN	0x0
9	WAITALV	根据存储器特性设置
8	BURSTEN	0x1
7		0x1
6	NOREN	根据存储器设置
5-4	BUSTYPE	根据需要设置
3-2	DEV	0x1 或 0x2
1	MUXEN	根据需要设置
0	EN	0x1

表 19-27 XMC_BK1TMGx位域（同步模式）

位编号	位名称	设置的数值
31-28		0x0
27-24	DTSTBL	数据保持时间
23-20	CLKPSC	0x0–得到 CLK=HCLK（不支持） 0x1–得到 CLK=2xHCLK
19-16	INTVLOP	不起作用
15-8	DTOP	不起作用
7-4	ADRHLD	不起作用
3-0	ADROP	不起作用

图 19-17 同步的总线复用写模式 - PSRAM（CRAM）



注： 1.存储器必须提前一个周期产生 NWAIT 信号，同时 WAITCFG 应配置为 0。

2.字节选择 NBL 输出没有显示在图中，当 NEx 为有效时它们为低。

表 19-28 XMC_BK1CTRLx位域（同步模式）

位编号	位名称	设置的数值
31-20		0x0000
19	BURSTWRSYN	0x1
18-16	PGSIZE	根据需要设置（0x1 为 CRAM 1.5）
15		0x0
14	TMGWREN	0x0
13	WAITEN	如果存储器支持此功能，设置成 1；否则保持 0
12	WREN	0x1
11	WAITCFG	0x0
10	BRSTSPLTEN	0x0
9	WAITALV	根据存储器特性设置
8	BURSTEN	对同步写模式不起作用
7		0x1
6	NOREN	根据存储器设置
5-4	BUSTYPE	根据需要设置
3-2	DEV	0x1
1	MUXEN	根据需要设置
0	EN	0x1

表 19-29 XMC_BK1TMGx位域（同步模式）

位编号	位名称	设置的数值

31-28	-	0x0
27-24	DTSTBL	数据保持时间
23-20	CLKPSC	0x0–得到 CLK=HCLK（不支持） 0x1–得到 CLK=2 x HCLK
19-16	INTVLOP	NEx 从高到低的时间（INTVLOP HCLK）
15-8	DTOP	不起作用
7-4	ADRHLD	不起作用
3-0	ADROP	不起作用

19.3.3 NAND闪存控制器

XMC 可以为下列类型的设备产生合适的信号时序：

- NAND 闪存
  - 8 位
  - 16 位

NAND 控制器能够控制存储块 2。

存储块由专门的寄存器配置（见 19.4 节），可编程的存储器参数包括操作时序和 ECC 配置。

表 19-30 可编程NAND操作参数

参数	功能	操作模式	单位	最小	最大
存储器建立时间	发出命令之前建立地址的（HCLK）时钟周期数目	读/写	AHB 时钟周期（HCLK）	1	256
存储器等待时间	发出命令的最短持续时间（HCLK 周期数目）	读/写		1	255
存储器保持时间	在发送命令结束后保持地址的（HCLK）时钟周期数目，写操作时也是数据的保持时间	读/写		1	255
存储器数据总线高阻时间	启动写操作之后保持数据总线为高阻态的时间	写		0	255

19.3.3.1 外部存储器接口信号

下表列出了用于接口 NAND 闪存的典型信号线。

前缀'N'代表对应的信号线为低电平有效。

8 位 NAND 闪存

表 19-31 8位NAND闪存

XMC 信号名称	输入/输出	功能
A[17]	输出	NAND 闪存地址锁存允许信号（ALE）
A[16]	输出	NAND 闪存命令锁存允许信号（CLE）
D[7：0]	入/出	8 位复用的、双向地址/数据总线
NCE[2]	输出	片选
NOE（=NRE）	输出	输出使能（存储器端信号名称：读使能，NRE）
NWE	输出	写使能
NWAIT	输入	NAND 闪存就绪/繁忙，输入至 XMC 的信号

XMC 可以根据需要产生多个地址周期，理论上 XMC 不限制可以访问的 NAND 容量。

16 位 NAND 闪存

表 19-32 16位NAND闪存

XMC 信号名称	输入/输出	功能
A[17]	输出	NAND 闪存地址锁存允许信号（ALE）

A[16]	输出	NAND 闪存命令锁存允许信号（CLE）
D[15：0]	入/出	16 位复用的、双向地址/数据总线
NCE[2]	输出	片选
NOE（=NRE）	输出	输出使能（存储器端信号名称：读使能，NRE）
NWE	输出	写使能
NWAIT	输入	NAND 闪存就绪/繁忙，输入至 XMC 的信号

XMC 可以根据需要产生多个地址周期，理论上 XMC 不限制可以访问的 NAND 容量。

19.3.3.2 NAND闪存支持的存储器及其操作

下表列出了支持的设备、操作模式和操作方式。在表格中有阴影的部分表示 NAND 闪存控制器不支持对应的操作方式。

表 19-33 支持的存储器及其操作

存储器	模式	读/写	AHB 数据宽度	存储器数据宽度	是否支持	注释
8 位 NAND	异步	读	8	8	支持	
	异步	写	8	8	支持	
	异步	读	16	8	支持	分成 2 次 XMC 访问
	异步	写	16	8	支持	分成 2 次 XMC 访问
	异步	读	32	8	支持	分成 4 次 XMC 访问
	异步	写	32	8	支持	分成 4 次 XMC 访问
16 位 NAND	异步	读	8	16	支持	
	异步	写	8	16	不支持	
	异步	读	16	16	支持	
	异步	写	16	16	支持	
	异步	读	32	16	支持	分成 2 次 XMC 访问
	异步	写	32	16	支持	分成 2 次 XMC 访问

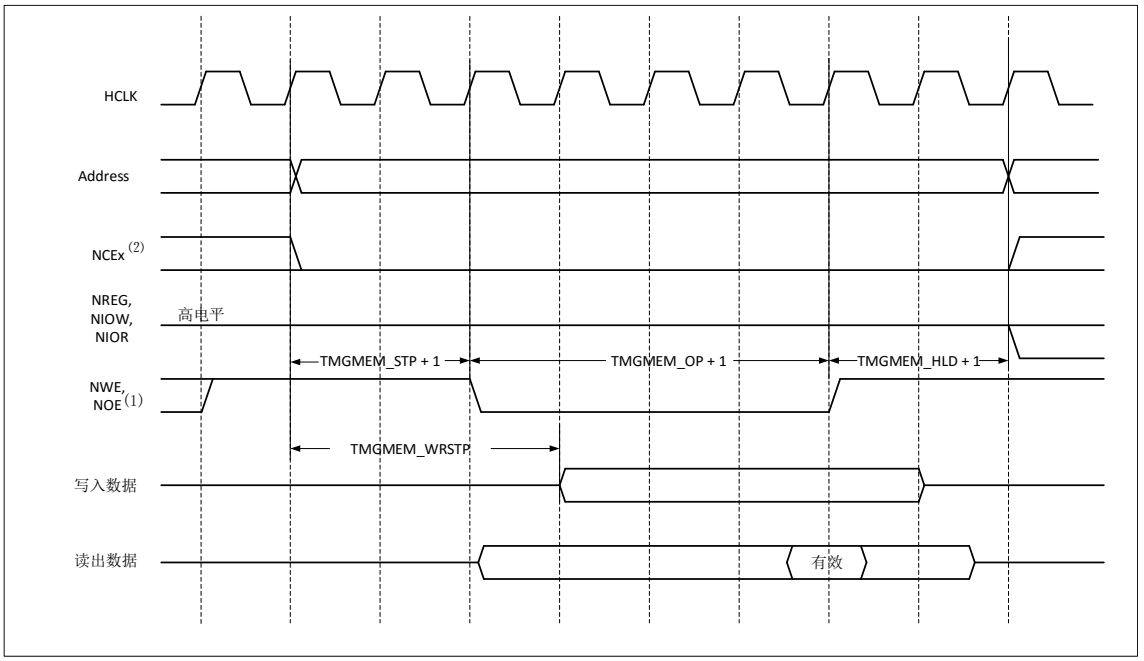
19.3.3.3 NAND闪存时序图

NAND 闪存存储器块是通过一组寄存器管理：

- 控制寄存器：XMC_BK2CTRL
- 中断状态寄存器：XMC_ BK2STS
- ECC 寄存器：XMC_BK2ECC
- 通用存储器空间的时序寄存器：XMC_ BK2TMGMEM
- 属性存储器空间的时序寄存器：XMC_ BK2TMGATT

每一个时序控制寄存器都包含 3 个参数，用于定义 NAND 闪存操作中三个阶段的 HCLK 周期数目，还有一个定义了写操作中 XMC 开始驱动数据总线时机的参数。下图给出了在通用存储空间中操作的时序参数定义，属性存储空间中操作与此相似。

图 19-18 NAND控制器通用存储空间的访问时序



注意： 1.在写操作时 NOE 始终保持高（无效状态），在读操作时 NWE 始终保持高（无效状态）。

2.只要请求 NAND 访问，NCE2 信号就变低并在访问其它存储器块之前保持为低。

19.3.3.4 NAND闪存操作

正如前面所述，NAND 闪存的命令锁存使能（CLE）和地址锁存使能（ALE）信号由 XMC 的地址信号线驱动。这意味着在向 NAND 闪存发送命令或地址时，CPU 需要对存储空间中的特定地址执行写操作。

一个典型的对 NAND 闪存的读操作有如下步骤：

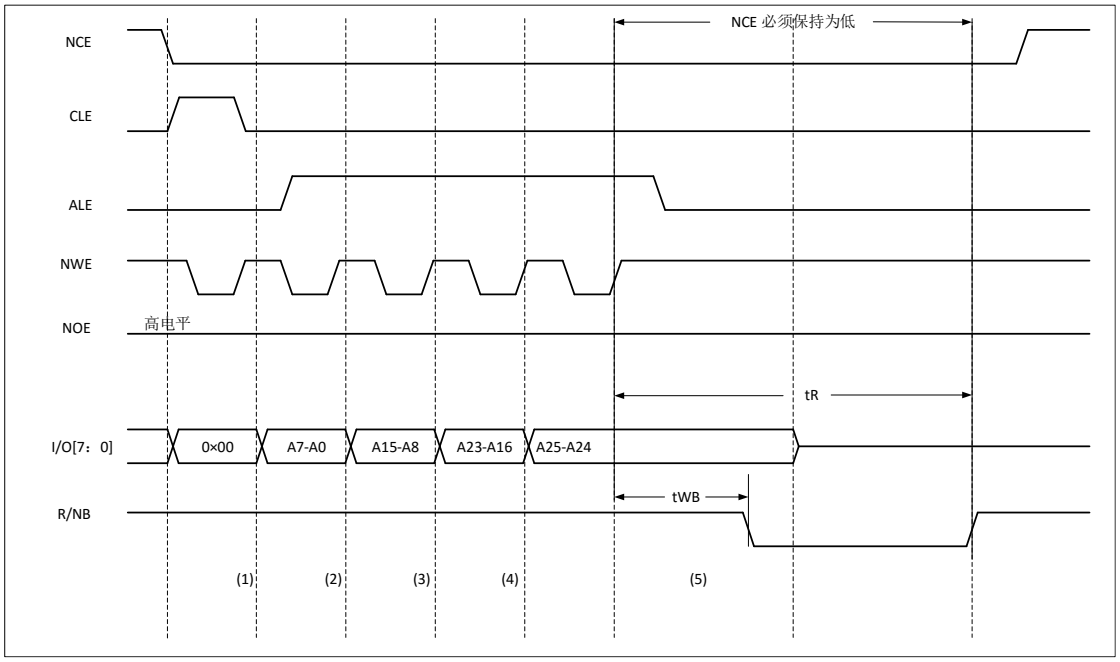
1. 根据NAND闪存的特性，通过XMC_BK2CTRL和XMC_BK2TMGMEM寄存器配置和使能相应的存储器块，对于某些NAND闪存可能还要操作XMC_BK2TMGATT寄存器（见19.3.3.5节NAND闪存预等待功能）。需要配置的位包括：BUSTYPE指示NAND闪存的数据总线宽度，DEV=1，WAITEN=0 或1，参见19.4.2.3节通用存储空间时序寄存器2（XMC_BK2TMGMEM）的时序配置。
2. CPU在通用存储空间写入闪存命令字节（例如对于Samsung的NAND闪存，该字节为0x00），在写信号有效期间（NWE的低脉冲）NAND闪存的CLE输入变为有效（高电平），这时CPU写的字节被NAND闪存识别为一个命令。一旦NAND闪存锁存了这个命令，随后的页读操作不必再发送相同的命令。
3. CPU在通用存储器空间或属性空间写入四个字节（较小容量的NAND闪存可能只需要三个字节）作为读操作的开始地址（STARTAD），以64Mbx8的NAND闪存为例，按照STARTAD[7：0]、STARTAD[15：8]、STARTAD[23：16]和STARTAD[25：24]的顺序写入。在写信号有效期间（NWE的低脉冲）NAND闪存的ALE输入变为有效（高电平），这时CPU写的字节被NAND闪存识别为读操作的开始地址。使用属性存储空间，可以使XMC产生不同的时序，实现某些NAND闪存所需的预等待功能（见19.3.3.5节NAND闪存预等待功能）。
4. 控制器在开始（对相同的或另一个存储器块）新的操作之前等待NAND闪存准备就绪（R/NB信号变为高），在等待期间控制器保持NCE信号有效（低电平）。
5. CPU可以在通用存储空间执行字节读操作，逐字节地读出NAND闪存的存储页（数据域和后备域）
6. 在CPU不写入命令或地址的情况下，NAND闪存的下一个页可以以下述任一种方式读出：

- 按照步骤 5 进行操作
- 返回步骤 3 开始输入一个新的地址
- 返回步骤 2 开始输入一个新的命令

19.3.3.5 NAND闪存预等待功能

某些 NAND 闪存要求在输入最后一个地址字节后，控制器等待 R/NB 信号变低，如下图：

图 19-19 操作CE敏感型NAND闪存



注意： 1.CPU 写字节 0x00 至地址 0x7001 0000

2.CPU 写字节 A7~A0 至地址 0x7002 0000

3.CPU 写字节 A15~A8 至地址 0x7002 0000

4.CPU 写字节 A23~A16 至地址 0x7002 0000

5.CPU 写字节 A25~A24 至地址 0x7802 0000：这时 XMC 使用 XMC_BK2TMGATT 的时序定义执行写操作，此时 TMGATT_HLD≥7（这里：（7+1）xHCLK=112ns>t_{WB} 的最大值）。这样可以保证 R/NB 变低再变高的过程中 NCE 保持为低，只有那些对 CE 敏感型的 NAND 闪存有此要求。

当需要这样的功能时，可以通过配置 TMGMEM_HLD 的数值来保证 t_{WB} 的时序，但是任何随后的 CPU 对 NAND 闪存的读或写操作中，控制器都会在 NWE 信号的上升沿至下一次操作之间插入一个保持延迟，延迟长度为（TMGMEM_HLD+1）个 HCLK 周期。

为了克服这个时序的限制，这里使用了属性存储空间配置 TMGATT_HLD 的数值使之符合 t_{WB} 的时序，同时保持 TMGMEM_HLD 为其最小值。此时，CPU 必须在所有 NAND 闪存的读写操作时使用通用存储空间，只有在写入 NAND 闪存地址的最后一个字节时，CPU 需要写入属性存储空间。

19.3.3.6 NAND闪存的纠错码ECC计算（NAND闪存）

XMC 的 NAND 闪存控制器包含了 1 个纠错码计算的硬件模块，于存储块 2 内。该模块可以用于减小 CPU 在处理纠错码时的软件工作量。

有一个寄存器分别对应存储块 2 包含的硬件 ECC 计算模块。

XMC 中实现的纠错码（ECC）算法可以在读或写 NAND 闪存时，在每 256、512、1024、2048、4096 或 8192 个字节中，矫正 1 个比特位的错误并且检测出 2 个比特位的错误。

每当 NAND 闪存存储器卡被激活时，ECC 模块监测 NAND 闪存的数据总线和读/写信号（NCE 和 NWE）。

该功能的操作如下：

- 当在存储器块 2 访问 NAND 闪存时，出现在 D[15：0]总线上的数据被锁存并用于 ECC 计算
- 当对 NAND 闪存的操作发生在其它地址时，ECC 电路不进行任何操作。因此输出 NAND 闪存命令和地址的写操作不会参与 ECC 计算

当规定数目的字节已经写入 NAND 闪存或从 NAND 闪存读出，软件必须读出 XMC_BK2ECC 寄存器以获得计算的 ECC 数值。读出 ECC 数值后，再次计算 ECC 时需要通过先置 ECCEN 为'0'清除这个寄存器，再在 XMC_BK2CTRL 寄存器的 ECCEN 位写'1'重新使能 ECC 计算。

19.4 XMC寄存器

表 19-34 XMC寄存器地址映像

偏移	寄存器名称	复位值	说明
000h	XMC_BK1CTRL1	0x000030DB	SRAM/NOR 闪存片选控制寄存器 1
004h	XMC_BK1TMG1	0x0FFFFFFF	SRAM/NOR 闪存片选时序寄存器 1
018h	XMC_BK1CTRL4	0x000030D2	SRAM/NOR 闪存片选控制寄存器 4
01Ch	XMC_BK1TMG4	0x0FFFFFFF	SRAM/NOR 闪存片选时序寄存器 4
060h	XMC_BK2CTRL	0x00000018	NAND 闪存控制寄存器 2
064h	XMC_BK2STS	0x00000040	FIFO 状态和中断寄存器 2
068h	XMC_BK2TMGMEM	0xFCFCFCFC	通用存储空间时序寄存器 2
06Ch	XMC_BK2TMGATT	0xFCFCFCFC	属性存储空间时序寄存器 2
074h	XMC_BK2ECC	0x00000000	ECC 结果寄存器 2
104h	XMC_BK1TMGWR1	0x0FFFFFFF	SRAM/NOR 闪存写时序寄存器 1
11Ch	XMC_BK1TMGWR4	0x0FFFFFFF	SRAM/NOR 闪存写时序寄存器 4
220h	XMC_EXT1	0x00000808	SRAM/NOR 额外时序寄存器 1
22Ch	XMC_EXT4	0x00000808	SRAM/NOR 额外时序寄存器 4

19.4.1 NOR闪存和PSRAM控制器寄存器

必须以字（32 位）的方式操作这些外设寄存器。

19.4.1.1 SRAM/NOR闪存片选控制寄存器1,4（XMC_BK1CTRL1,4）

地址偏移：0xA0000000+8*（x-1）,x=1,4

复位值：Bank1 为 0x000030DB，Bank4 为 0x000030D2

这个寄存器包含了每个存储器块的控制信息，可以用于 SRAM、异步或成组传输的 NOR 闪存存储器。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留												BURSTWRSYN	PGSIZE		
res												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WAITASYNC	TMGWREN	WAITEN	WREN	WAITCFG	BRSTSPLTEN	WAITALV	BURSTEN	保留	NORESN	BUSTYPE		DEV		MUXEN	EN
rw	rw	rw	rw	rw	rw	rw	rw	res	rw	rw	rw	rw	rw	rw	rw

位 31：20	保留

位 19	**BURSTWRSYN**：成组写使能位（Write burst enable） 对于 Cellular RAM，该位使能写操作的同步成组传输协议。 对于处于成组传输模式的闪存存储器，这一位允许/禁止通过 NWAIT 信号插入等待状态。读操作的同步成组传输协议使能位是 XMC_BK1CTRLx 寄存器的 BURSTEN 位。 0：写操作始终处于异步模式 1：写操作为同步模式
位 18：16	**PGSIZE**：CRAM 页大小（CRAM page size） 用于 Cellular RAM 1.5，不允许跨页地址边界的成组访问。当配置这些位时，只要达到存储器页大小时，XMC 控制器会自动拆分成组访问（参考存储器参考手册的页大小）。 000：当跨页地址边界时不会拆分成组访问，这是复位后的默认状态。 001：128 字节 010：256 字节 011：512 字节 100：1024 字节 其他：保留
位 15	**WAITASYNC**：在异步传输期间等待信号（Wait signal during asynchronous transfers） 此位使能/禁止 XMC 去使用等待信号，即使在异步协议期间。 0：当运行异步协议时，不考虑 NWAIT 信号，这是复位后的默认状态。 1：在运行异步协议时，考虑 NWAIT 信号。
位 14	**TMGWREN**：扩展模式使能（Extended mode enable） 该位允许 XMC 使用 XMC_BK1TMGWR 寄存器，即允许读和写使用不同的时序。 0：不使用 XMC_BK1TMGWR 寄存器，这是复位后的默认状态。 1：XMC 使用 XMC_BK1TMGWR 寄存器。
位 13	**WAITEN**：等待使能位（Wait enable bit） 当闪存存储器处于同步传输模式时，这一位允许/禁止通过 NWAIT 信号插入等待状态。 0：禁用 NWAIT 信号，在设置的闪存保持周期之后不会检测 NWAIT 信号插入等待状态。 1：使用 NWAIT 信号，在设置的闪存保持周期之后根据 NWAIT 信号插入等待状态；这是复位后的默认状态。
位 12	**WREN**：写使能位（Write enable bit） 该位指示 XMC 是否允许/禁止对存储器的写操作。 0：禁止 XMC 对存储器的写操作，否则产生一个 AHB 错误。 1：允许 XMC 对存储器的写操作；这是复位后的默认状态。
位 11	**WAITCFG**：配置等待时序（Wait timing configuration） 当闪存存储器处于成组传输模式时，NWAIT 信号指示从闪存存储器出来的数据是否有效或是否需要插入等待周期。该位决定存储器是在等待状态之前的一个时钟周期产生 NWAIT 信号，还是在等待状态期间产生 NWAIT 信号。 0：NWAIT 信号在等待状态前的一个数据周期有效；这是复位后的默认状态。 1：NWAIT 信号在等待状态期间有效（不适用于 Cellular RAM）。
位 10	**BRSTSPLTEN**：支持非对齐的成组模式（Wrapped burst mode support） 该位决定控制器是否支持把非对齐的 AHB 成组操作分割成 2 次线性操作；该位仅在存储器的成组模式下有效。 0：不允许直接的非对齐成组操作；这是复位后的默认状态。 1：允许直接的非对齐成组操作。
位 9	**WAITALV**：等待信号极性（Wait signal polarity bit） 设置存储器产生的等待信号的极性；该位仅在存储器的成组模式下有效。 0：NWAIT 等待信号为低时有效；这是复位后的默认状态。 1：NWAIT 等待信号为高时有效。
位 8	**BURSTEN**：成组模式使能（Burst enable bit） 允许对闪存存储器进行成组模式访问；该位仅在闪存存储器的同步成组模式下有效。 0：禁用成组访问模式；这是复位后的默认状态。 1：使用成组访问模式。
位 7	保留。

位 6	**NOREN**：闪存访问使能（Flash access enable） 允许对 NOR 闪存存储器的访问操作。 0：禁止对 NOR 闪存存储器的访问操作。 1：允许对 NOR 闪存存储器的访问操作。
位 5：4	**BUSTYPE**：存储器数据总线宽度（Memory databus width） 定义外部存储器总线的宽度，适用于所有类型的存储器。 00：8 位， 01：16 位（复位后的默认状态） 10：保留，不能用 11：保留，不能用
位 3：2	**DEV**：存储器类型（Memory type） 定义外部存储器的类型： 00：SRAM（存储器块 4 在复位后的默认值） 01：PSRAM（Cellular RAM： CRAM） 10：NOR 闪存（存储器块 1 在复位后的默认值） 11：保留
位 1	**MUXEN**：地址/数据复用使能位（Address/data multiplexing enable bit） 当设置了该位后，地址的低 16 位和数据将共用数据总线，该位仅对 NOR 和 PSRM 存储器有效。 0：地址/数据不复用。 1：地址/数据复用数据总线；这是复位后的默认状态。
位 0	**EN**：存储器块使能位（Memory bank enable bit） 开启对应的存储器块。复位后存储器块 1 是开启的，其它所有存储器块为禁用。访问一个禁用的存储器块将在 AHB 总线上产生一个错误。 0：禁用对应的存储器块。 1：启用对应的存储器块。

19.4.1.2 SRAM/NOR闪存片选时序寄存器1,4（XMC_BK1TMG1,4）

地址偏移：0xA0000000+0x04+8*（x-1）,x=1,4

复位值：0x0FFFFFFF

这个寄存器包含了每个存储器块的控制信息，可以用于 SRAM 和 NOR 闪存存储器。如果 XMC_BK1CTRLx 寄存器中设置了 TMGWREN 位，则有两个时序寄存器分别对应读（本寄存器）和写操作（XMC_BK1TMGWRx 寄存器）。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留		MODE		DTSTBL				CLKPSC				INTVLOP			
res		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DTOP								ADRHLD				ADROP			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31：30	保留
位 29：28	**MODE**：访问模式（Access mode） 定义异步访问模式。这 2 位只在 XMC_BK1CTRLx 寄存器的 TMGWREN 位为 1 时起作用。 00：访问模式 A 01：访问模式 B 10：访问模式 C 11：访问模式 D

位 27：24	**DTSTBL**（见本表格下面的注释）：（同步成组式 NOR 闪存的）数据保持时间（Data latency （for synchronous burst NOR Flash）） 处于同步成组模式的 NOR 闪存，需要定义在读取第一个数据之前等待的存储器周期数目。 这个时间参数不是以 HCLK 表示，而是以闪存时钟（CLK）表示。在访问异步 NOR 闪存或 SRAM 时，这个参数不起作用。操作 CRAM 时，这个参数必须为 0。 0000：第一个数据的保持时间为 2 个 CLK 时钟周期 …… 1111：第一个数据的保持时间为 17 个 CLK 时钟周期（这是复位后的默认数值）
位 23：20	**CLKPSC**：时钟分频比（CLK 信号）（Clock divide ratio（for CLK signal）） 定义 CLK 时钟输出信号的周期，以 HCLK 周期数表示： 0000：保留 0001：1 个 CLK 周期=2 个 HCLK 周期 0010：1 个 CLK 周期=3 个 HCLK 周期 …… 1111：1 个 CLK 周期=16 个 HCLK 周期（这是复位后的默认数值）。在访问异步 NOR 闪存或 SRAM 时，这个参数不起作用。
位 19：16	**INTVLOP**：总线恢复时间（Bus turnaround phase duration） 这些位用于定义一次读操作之后在总线上的延迟（仅适用于总线复用模式的 NOR 闪存操作），一次读操作之后控制器需要在数据总线上为下次操作送出地址，这个延迟就是为了防止总线冲突。如果扩展的存储器系统不包含总线复用模式的存储器，或最慢的存储器可以在 6 个 HCLK 时钟周期内将数据总线恢复到高阻状态，可以设置这个参数为其最小值。 0000：总线恢复时间=1 个 HCLK 时钟周期 …… 1111：总线恢复时间=16 个 HCLK 时钟周期（这是复位后的默认数值）
位 15：8	**DTOP**：数据保持时间（Data-phase duration） 这些位定义数据的保持时间（见图 19-3 至图 19-15），适用于 SRAM 和异步总线复用模式的 NOR 闪存操作。 00000000：保留 00000001：DTOP 保持时间=2 个 HCLK 时钟周期 00000010：DTOP 保持时间=3 个 HCLK 时钟周期 …… 11111111：DTOP 保持时间=256 个 HCLK 时钟周期（这是复位后的默认数值） 对于每一种存储器类型和访问方式的数据保持时间，请参考对应的图表（见图 19-3 至图 19-15）。 例如：模式 1、读操作、DTOP=1：数据保持时间=DTOP+3=4 个 HCLK 时钟周期。
位 7：4	**ADRHLD**：地址保持时间（Address-hold phase duration） 这些位定义地址的保持时间（见图 19-13 至图 19-15），适用于 SRAM 和异步总线复用模式的 NOR 闪存操作。 0000：保留 0001：ADRHLD 保持时间=2 个 HCLK 时钟周期 0010：ADRHLD 保持时间=3 个 HCLK 时钟周期 … 1111：ADRHLD 保持时间=16 个 HCLK 时钟周期（这是复位后的默认数值） 注：在同步操作中，这个参数不起作用，地址保持时间始终是 1 个存储器时钟周期。
位 3：0	**ADROP**：地址建立时间（Address setup phase duration） 这些位定义地址的建立时间（见图 19-3 至图 19-15），适用于 SRAM 和异步总线复用模式的 NOR 闪存操作。 0000：ADROP 建立时间=1 个 HCLK 时钟周期 …… 1111：ADROP 建立时间=16 个 HCLK 时钟周期（这是复位后的默认数值） 对于每一种存储器类型和访问方式的地址建立时间，请参考对应的图表（见图 19-3 至图 19-15）。 例如：模式 2、读操作、ADROP=1：地址建立时间=ADROP+1=2 个 HCLK 时钟周期 注：在同步操作中，这个参数不起作用，地址建立时间始终是 1 个存储器时钟周期。

注意： 因为内部的刷新，PSRAM（CRAM）具有可变的保持延迟，因此这样的存储器会在数据保持期间输出 NWAIT 信号以延长数据的保持时间。

使用 PSRAM（CRAM）时 DTSTBL 域应置为 0，这样 XMC 可以及时地退出自己的保持阶段并开始对存储器发出的 NWAIT 信号进行采样，然后在存储器准备好时开始读或写操作。

这个操作方式还可以用于操作最新的能够输出 NWAIT 信号的同步闪存存储器，详细信息请参考相应的闪存存储器手册。

19.4.1.3 SRAM/NOR闪存写时序寄存器1,4（XMC_BK1TMGWR1,4）

地址偏移：0xA0000000+0x104+8*（x-1）,x=1,4

复位值：0x0FFFFFFF

这个寄存器包含了每个存储器块的控制信息，可以用于 SRAM 和 NOR 闪存存储器。如果 XMC_BK1CTRLx 寄存器中设置了 TMGWREN 位，则这个寄存器对应写操作。

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

位 31：30	保留
位 29：28	**MODE**：访问模式（Access mode） 定义异步访问模式。这 2 位只在 XMC_BK1CTRLx 寄存器的 TMGWREN 位为 1 时起作用。 00：访问模式 A 01：访问模式 B 10：访问模式 C 11：访问模式 D
位 27：20	保留
位 19：16	**INTVLOP**：总线恢复时间（Bus turnaround phase duration） 这些位用于定义一次读操作之后在总线上的延迟（仅适用于总线复用模式的 NOR 闪存操作），一次读操作之后控制器需要在数据总线上为下次操作送出地址，这个延迟就是为了防止总线冲突。如果扩展的存储器系统不包含总线复用模式的存储器，或最慢的存储器可以在 6 个 HCLK 时钟周期内将数据总线恢复到高阻状态，可以设置这个参数为其最小值。 0000：总线恢复时间=1 个 HCLK 时钟周期 …… 1111：总线恢复时间=16 个 HCLK 时钟周期（这是复位后的默认数值）
位 15：8	**DTOP**：数据保持时间（Data-phase duration） 这些位定义数据的保持时间（见图 19-3 至图 19-15），适用于 SRAM 和异步总线复用模式的 NOR 闪存操作。 00000000：保留 00000001：DTOP 保持时间=2 个 HCLK 时钟周期 00000010：DTOP 保持时间=3 个 HCLK 时钟周期 …… 11111111：DTOP 保持时间=256 个 HCLK 时钟周期（这是复位后的默认数值）

位 7：4	**ADRHLD**：地址保持时间（Address-hold phase duration） 这些位定义地址的保持时间（见图 19-12 至图 19-15），适用于 SRAM 和异步总线复用模式的 NOR 闪存操作。 0000：保留 0001：ADRHLD 保持时间=2 个 HCLK 时钟周期 0010：ADRHLD 保持时间=3 个 HCLK 时钟周期 …… 1111：ADRHLD 保持时间=16 个 HCLK 时钟周期（这是复位后的默认数值） 注：在同步 NOR 闪存操作中，这个参数不起作用，地址保持时间始终是 1 个闪存时钟周期。
位 3：0	**ADROP**：地址建立时间（Address setup phase duration） 这些位以 HCLK 周期数定义地址的建立时间（见图 19-3 至图 19-15），适用于 SRAM 和异步总线复用模式的 NOR 闪存操作。 0000：ADROP 建立时间=1 个 HCLK 时钟周期 …… 1111：ADROP 建立时间=16 个 HCLK 时钟周期（这是复位后的默认数值） 注：在同步 NOR 闪存操作中，这个参数不起作用，地址建立时间始终是 1 个闪存时钟周期。

19.4.1.4 SRAM/NOR额外时序寄存器1,4（XMC_EXT1,4）

地址偏移：0xA0000000+0x220+4*（x-1）,x=1,4

复位值：0x00000808

这个寄存器包含了每个存储器块的控制信息，可以用于 SRAM 和 NOR 闪存存储器。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															
res															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BURSTURNR2R								BURSTURNW2W							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31：16	保留
位 15：8	**BURSTURNR2R**：连续读操作恢复时间（Bus turnaround phase for consecutive read duration） 用于定义异步之连续读操作间总线恢复时间。 00000000：连续读操作间隔 1 HCLK 周期 00000001：连续读操作间隔 2 HCLK 周期 …… 00001000：连续读操作间隔 9 HCLK 周期（默认值） …… 11111111：连续读操作间隔 256 HCLK 周期
位 7：0	BURSTURNW2W：连续写操作恢复时间（Bus turnaround phase for consecutive write duration） 用于定义异步之连续写操作间总线恢复时间。 00000000：连续写操作间隔 1 HCLK 周期 00000001：连续写操作间隔 2 HCLK 周期 …… 00001000：连续写操作间隔 9 HCLK 周期（默认值） …… 11111111：连续写操作间隔 256 HCLK 周期

19.4.2 NAND闪存控制器寄存器

必须以字（32 位）的方式操作这些外设寄存器。

19.4.2.1 NAND闪存控制寄存器2（XMC_BK2CTRL）

地址偏移：0xA0000000+0x40+0x20*（x-1）,x=2

复位值：0x00000018

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留												ECCPGSIZE			DLYAR
res												rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DLYAR			DLYCR				保留		ECCEN	BUSTYPE		DEV	EN	WAITEN	保留
rw	rw	rw	rw	rw	rw	rw	res		rw	rw	rw	rw	rw	rw	res

位 31：20	保留
位 19：17	**ECCPGSIZE**：ECC 页面大小（ECC page size） 定义扩展的 ECC 页面大小： 000：256 字节 001：512 字节 010：1024 字节 011：2048 字节 100：4096 字节 101：8192 字节
位 16：13	**DLYAR**：ALE 至 RE 的延迟（ALE to RE delay） 以 AHB 时钟周期（HCLK）为单位设置从 ALE 变低至 RE 变低的时间。 时间计算：t_ar = （DLYAR + SET + 4） x THCLK，这里 THCLK 表示 HCLK 周期长度 0000：1 个 HCLK 周期（默认值） …… 1111：16 个 HCLK 周期 注：根据不同的地址空间，SET 是 TMGMEM_STP 或是 TMGATT_STP。
位 12：9	**DLYCR**：CLE 至 RE 的延迟（CLE to RE delay） 以 AHB 时钟周期（HCLK）为单位设置从 CLE 变低至 RE 变低的时间。 时间计算：t_clr = （DLYCR+SET+4）xTHCLK，这里 THCLK 表示 HCLK 周期长度 0000：1 个 HCLK 周期（默认值） …… 1111：16 个 HCLK 周期 注：根据不同的地址空间，SET 是 TMGMEM_STP 或是 TMGATT_STP。
位 8：7	保留
位 6	**ECCEN**：ECC 计算电路使能位（ECC computation logic enable bit） 0：关闭并复位 ECC 电路（复位后的默认值）； 1：使能 ECC 电路。
位 5：4	**BUSTYPE**：数据总线宽度（Databus width） 定义外部 NAND 闪存数据总线的宽度。 00：8 位（复位后的默认值）； 01：16 位； 10：保留，不要使用； 11：保留，不要使用。
位 3	**DEV**：存储器类型（Memory type） 定义对应的存储器块上连接的存储器类型。 0：保留； 1：NAND 闪存（复位后的默认值）。

位 2	**EN**：NAND 存储器块使能位（NAND Flash memory bank enable bit） 使能存储器块。访问一个未使能的存储器块会产生一个 AHB 总线错误。 0：关闭对应的存储器块（复位后的默认值）； 1：使能对应的存储器块。
位 1	**WAITEN**：等待功能使能位（Wait feature enable bit） 使能 NAND 闪存存储器块的等待功能 0：关闭（复位后的默认值）； 1：使能。
位 0	保留

19.4.2.2 FIFO状态和中断寄存器2（XMC_BK2STS）

地址偏移：0xA0000000+0x44+0x20*（x-1）,x=2

复位值：0x00000040

该寄存器包含 FIFO 状态和中断的信息。XMC 有一个 FIFO，在写存储器时用于保存从 AHB 送来的多达 16 个字的数据。

这个功能可以在操作 XMC 时不过多地占用 AHB，在 XMC 从 FIFO 传送数据到存储器时施放 AHB 的带宽并操作其他外设。

为了计算 ECC 的需要，该寄存器有一个指示位反映了 FIFO 的状态。数据写到存储器时同时进行 ECC 计算，因此软件必须等待 FIFO 变空后才能读到正确的 ECC 数值。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															
res															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留									FIFOE	IFEN	IHLEN	IREN	IFF	IHLF	IRF
res									rw	rw	rw	rw	rw	rw	rw

位 31：7	保留
位 6	**FIFOE**：FIFO 空标志（FIFO empty） 只读位，指示 FIFO 状态 0：FIFO 不空； 1：FIFO 空。
位 5	**IFEN**：中断下降沿检测使能（Interrupt falling edge detection enable bit） 0：关闭中断下降沿检测请求； 1：使能中断下降沿检测请求。
位 4	**IHLEN**：中断高电平检测使能（Interrupt high-level detection enable bit） 0：关闭中断高电平检测请求； 1：使能中断高电平检测请求。
位 3	**IREN**：上升沿中断检测使能（Interrupt rising edge detection enable bit） 0：关闭中断上升沿检测请求； 1：使能中断上升沿检测请求。
位 2	**IFF**：中断下降沿状态（Interrupt falling edge status） 该位由硬件设置，软件清除。 0：没有产生中断下降沿； 1：产生了中断下降沿。
位 1	**IHLF**：中断高电平状态（Interrupt high-level status） 该位由硬件设置，软件清除。 0：没有产生中断高电平； 1：产生了中断高电平。

位 0	**IRF**：中断上升沿状态（Interrupt rising edge status） 该位由硬件设置，软件清除。 0：没有产生中断上升沿； 1：产生了中断上升沿。

19.4.2.3 通用存储空间时序寄存器2（XMC_BK2TMGMEM）

地址偏移：0xA0000000+0x48+0x20*（x-1）,x=2

复位值：0xFCFCFCFC

XMC_BK2TMGMEM 寄存器都包含操作 NAND 闪存存储块 2 的时序参数，这些参数适用于在发送 NAND 闪存的命令、地址和进行数据的读写操作。

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

位 31：24	**WRSTP**：在通用空间数据总线的高阻时间（Common memory databus HiZ time） 当在通用存储空间开始执行对 NAND 闪存的写操作后，数据总线需要保持一段时间的高阻状态，该参数以 HCLK 时钟周期数目（NAND 类型时+1）定义数据总线高阻态的时间。这个参数仅对写操作有效。 00000000：（0x00）NAND 闪存为 1 个 HCLK 周期； …… 11111111：（0xFF）NAND 闪存为 256 个 HCLK 周期。
位 23：16	**HLD**：在通用空间的保持时间（Common memory hold time） 当在通用存储空间对 NAND 闪存进行读或写操作时，该参数以 HCLK 时钟周期数目定义了发送命令（NWE、NOE 变高）后，地址信号（对于写操作则是数据信号）保持的时间。 00000000：保留； 00000001：1 个 HCLK 周期； …… 11111111：255 个 HCLK 周期。
位 15：8	**OP**：在通用空间的等待时间（Common memory wait time） 当在通用存储空间对 NAND 闪存进行读或写操作时，该参数以 HCLK（+1）时钟周期数目定义了保持命令（NWE、NOE 为低）的最小时间。当该参数定义的时间结束时，如果等待信号（NWAIT）有效（低），则命令的保持时间会被拉长。 00000000：保留； 00000001：1 个 HCLK 周期（加上由 NWAIT 信号变低引入的等待周期）； …… 11111111：255 个 HCLK 周期（加上由卡的 NWAIT 信号变低引入的等待周期）。
位 7：0	**STP**：在通用空间的建立时间（Common memory setup time） 当在通用存储空间对 NAND 闪存进行读或写操作时，该参数以 HCLK（操作 NAND 闪存时+2）时钟周期数目定义了发送命令（NWE、NOE 变低）之前建立地址信号的时间。 00000000：NAND 闪存为 2 个 HCLK 周期； …… 11111111：NAND 闪存为 257 个 HCLK 周期。

19.4.2.4 属性存储空间时序寄存器2（XMC_BK2TMGATT）

地址偏移：0xA0000000+0x4C+0x20*（x-1）,x=2

复位值：0xFCFCFCFC

每个 XMC_BK2TMGATT 读/写寄存器都包含操作 NAND 闪存存储块 2 的时序参数，这些参数适用于在 NAND 闪存的最后一个地址写操作的时序与其它操作不同的时候（关于就绪/繁忙的管理，参见 19.3.3.5 节）。

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

位 31：24	**WRSTP**：在属性空间数据总线的高阻时间（Attribute memory databus HiZ time） 当在属性存储空间开始执行对 NAND 闪存的写操作后，数据总线需要保持一段时间的高阻状态，该参数以 HCLK 时钟周期数目定义数据总线高阻态的时间。这个参数仅对写操作有效。 00000000：0 个 HCLK 周期； …… 11111111：255 个 HCLK 周期（复位后的默认值）。
位 23：16	**HLD**：在属性空间的保持时间（Attribute memory x hold time） 当在属性存储空间对 NAND 闪存进行读或写操作时，该参数以 HCLK 时钟周期数目定义了发送命令（NWE、NOE 变高）后，地址信号（对于写操作则是数据信号）保持的时间。 00000000：保留； 00000001：1 个 HCLK 周期； …… 11111111：255 个 HCLK 周期（复位后的默认值）。
位 15：8	**OP**：在属性空间的等待时间（Attribute memory wait time） 当在属性存储空间对 NAND 闪存进行读或写操作时，该参数以 HCLK（+1）时钟周期数目定义了保持命令（NWE、NOE 为低）的最小时间。当该参数定义的时间结束时如果等待信号（NWAIT）有效（低），则命令的保持时间会被拉长。 00000000：保留； 00000001：1 个 HCLK 周期（加上由 NWAIT 信号变低引入的等待周期）； …… 11111111：255 个 HCLK 周期（加上由卡的 NWAIT 信号变低引入的等待周期）（复位后的默认值）。
位 7：0	**STP**：在属性空间的建立时间（Attribute memory setup time） 当在属性存储空间对 NAND 闪存进行读或写操作时，该参数以 HCLK（+1）时钟周期数目定义了发送命令（NWE、NOE 变低）之前建立地址信号的时间。 00000000：1 个 HCLK 周期； …… 11111111：256 个 HCLK 周期（复位后的默认值）

19.4.2.5 ECC结果寄存器2（XMC_BK2ECC）

地址偏移：0xA0000000+0x54+0x20*（x-1）,x=2

复位值：0x00000000

这个寄存器包含了由 XMC 控制器的 ECC 计算模块得到的纠错码的当前数值，每个 NAND 闪存存储器块有一个 ECC 计算模块。当 CPU 在正确的地址（见 19.3.3.6 节）读/写 NAND 闪存的数据时，ECC 模块会自动地处理写入或读出的数据。根据 XMC_BK2CTRL 中 ECCPGSIZE 域的设置，在读出了每页的最后一个字节后，CPU 必须读出 XMC_BK2ECC 寄存器中的 ECC 数值，并与记录在 NAND 闪存后备区域的数据进行比较，据此判断该页的数据是否正确并在可能的情况下，实行矫正。在读出 XMC_BK2ECCR 寄存器的数值后应设置 ECCEN 位为'0'清除它的内容。需要计算一个新的数据页时，再次设置 ECCEN

为'1'。

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

位 31：0	**ECC**：ECC 结果 ECC 计算电路产生的计算结果。下表显示了这些位的内容。

表 19-35 ECC结果有效位

ECCPGSIZE[2：0]	页大小（字节）	ECC 有效位
000	256	ECC[21：0]
001	512	ECC[23：0]
010	1024	ECC[25：0]
011	2048	ECC[27：0]
100	4096	ECC[29：0]
101	8192	ECC[31：0]

20 SDIO接口

20.1 简介

SD/SDIO MMC 卡主机模块（SDIO）在 AHB 外设总线和多媒体卡（MMC）、SD 存储卡、SDIO 卡间提供了操作接口。

多媒体卡系统规格书由 MMCA 技术委员会发布，可以在多媒体卡协会的网站上（www.mmca.org）获得。

SDIO 的主要功能如下：

- 与多媒体卡系统规格书版本 4.2 全兼容。支持三种不同的数据总线模式：1 位（默认）、4 位和 8 位
- 与较早的多媒体卡系统规格版本全兼容（向前兼容）
- 与 SD 存储卡规格版本 2.0 全兼容
- 与 SD I/O 卡规格版本 2.0 全兼容：支持两种不同的数据总线模式：1 位（默认）和 4 位
- 8 位总线模式下数据传输速率可达 50 MHz
- 数据和命令输出使能信号，用于控制外部双向驱动器

注意： *1.SDIO 没有 SPI 兼容的通信模式*

2.在多媒体卡系统规格书版本 2.11 中，定义 SD 存储卡协议是多媒体卡协议的超集。只支持 I/O 模式的 SD 卡或复合卡中的 I/O 部分不能支持 SD 存储设备中很多需要的命令，这里有些命令在 SD I/O 设备中不起作用，如擦除命令，因此 SDIO 不支持这些命令。另外，SD 存储卡和 SD I/O 卡中有些命令是不同的，SDIO 也不支持这些命令。细节可以参考 SD I/O 卡规格书版本 1.0。SDIO 接口的电气和信号定义详见 MMC 参考资料。

多媒体卡/SD 总线将所有卡与控制器相连。

当前版本的 SDIO 在同一时间里只能支持一个 SD/SDIO/MMC 4.2 卡，但可以支持多个 MMC 版本 4.1 或以前版本的卡。

20.2 主要特点

总线上的通信是通过传送命令和数据实现。

在多媒体卡/SD/SD I/O 总线上的基本操作是命令/响应结构，这样的总线操作在命令或总线机制下实现信息交换；另外，某些操作还具有数据令牌。

在 SD/SD IO 存储器卡上传送的数据是以数据块的形式传输；在 MMC 上传送的数据是以数据块或数据流的形式传输。

图 20-1 SDIO“无响应”和“无数据”操作

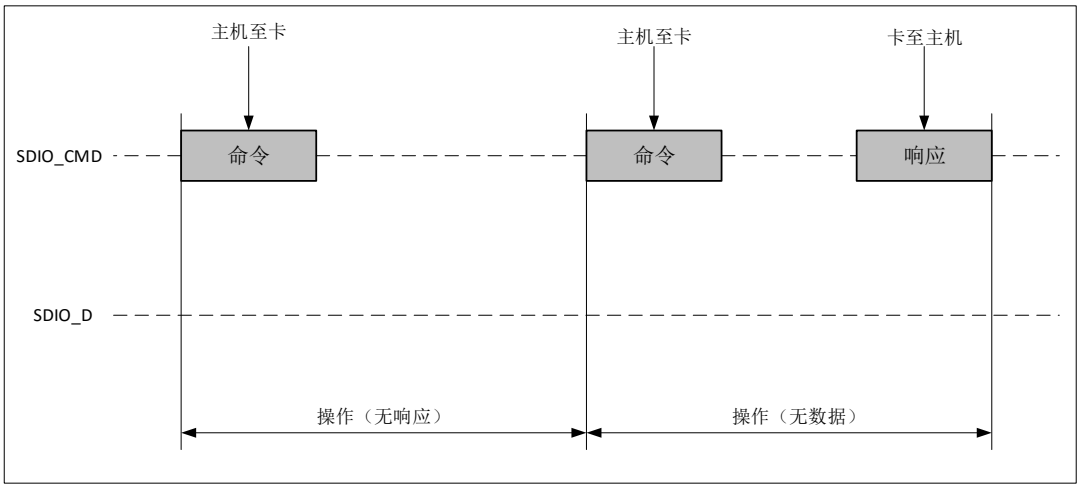


图 20-2 SDIO（多）数据块读操作

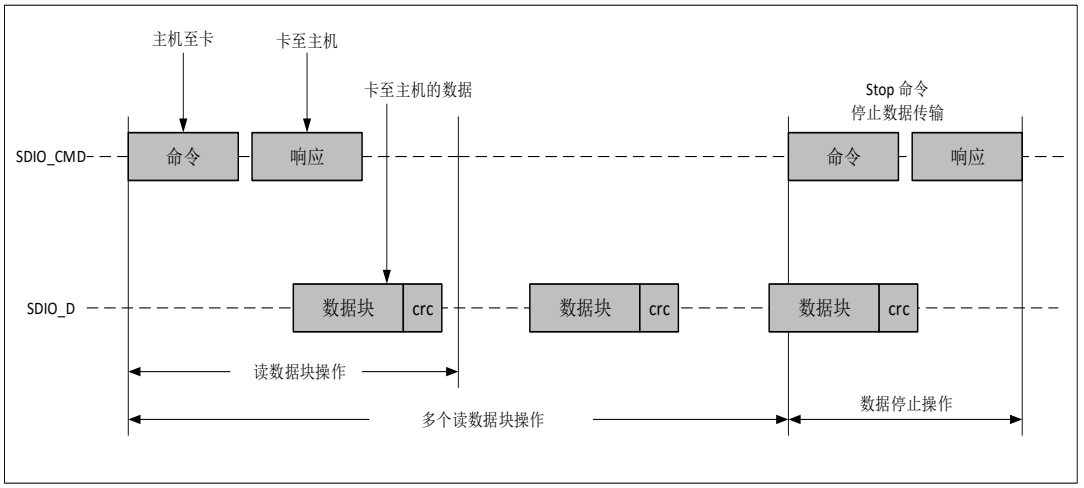
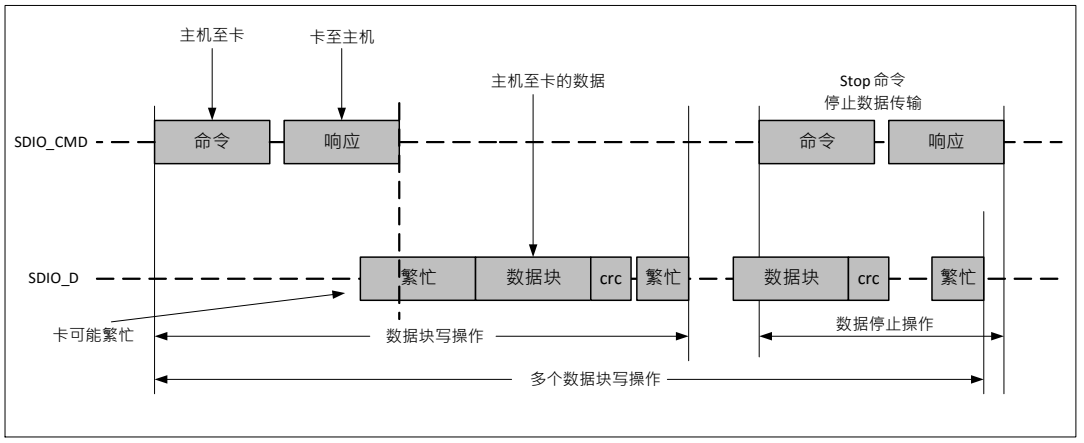


图 20-3 SDIO（多）数据块写操作



注意： 当有 Busy（繁忙）信号时，SDIO（SDIO_D0 被拉低）将不会发送任何数据。

图 20-4 SDIO连续读操作

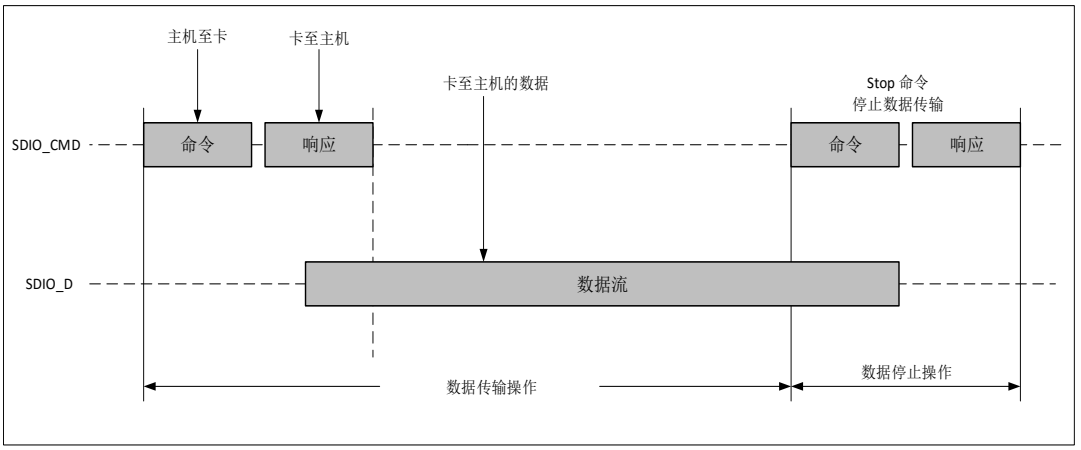
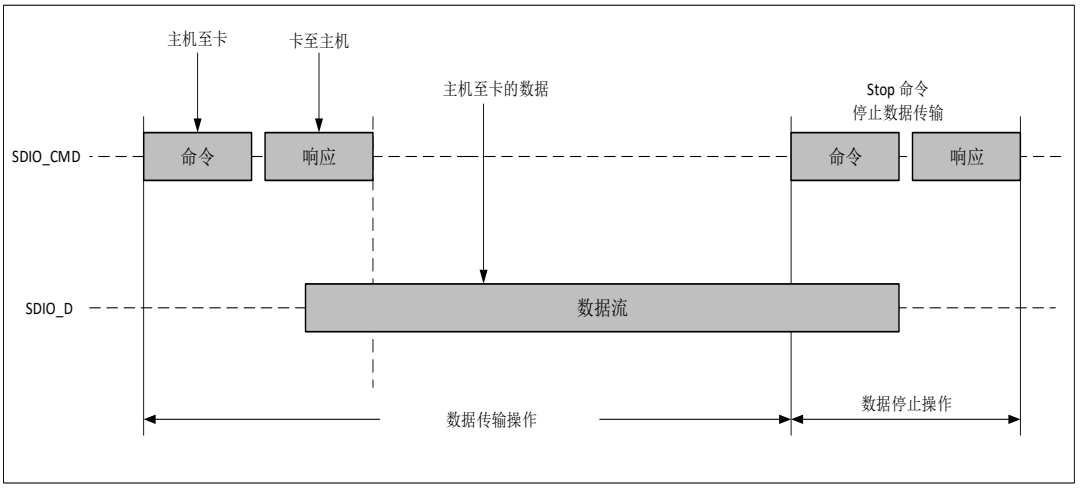


图 20-5 SDIO连续写操作



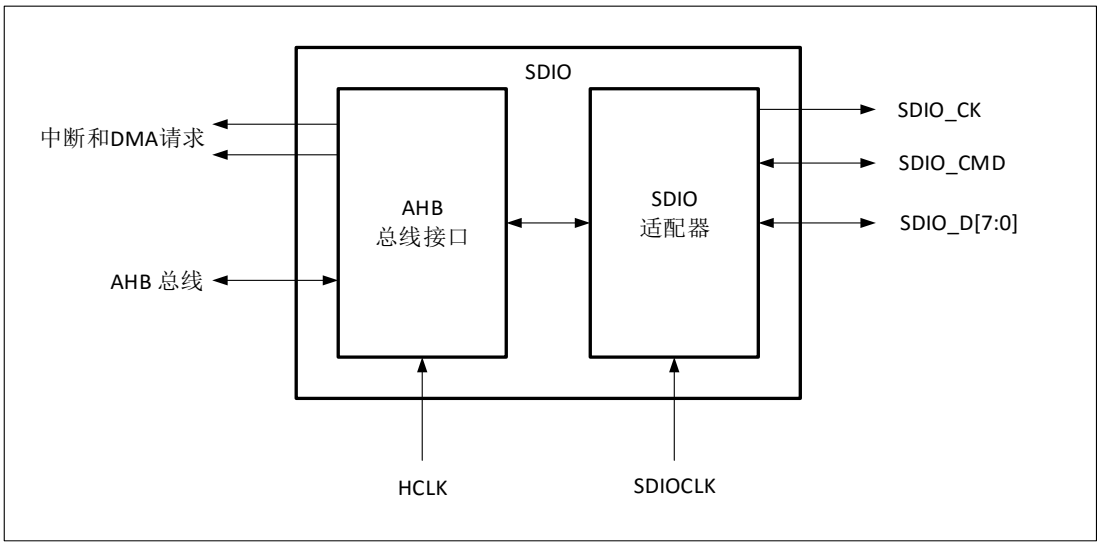
20.3 功能描述

20.3.1 SDIO功能描述

SDIO 包含 2 个部分：

- SDIO 适配器模块：实现所有 MMC/SD/SD I/O 卡的相关功能，如时钟的产生、命令和数据的传送
- AHB 总线接口：操作 SDIO 适配器模块中的寄存器，并产生中断和 DMA 请求信号

图 20-6 SDIO框图



复位后默认情况下 SDIO_D0 用于数据传输。初始化后主机可以改变数据总线的宽度。

如果一个多媒体卡接到了总线上，则 SDIO_D0、SDIO_D[3：0]或 SDIO_D[7：0]可以用于数据传输。MMC 版本 V3.31 和之前版本的协议只支持 1 位数据线，所以只能用 SDIO_D0。

如果一个 SD 或 SD I/O 卡接到了总线上，可以通过主机配置数据传输使用 SDIO_D0 或 SDIO_D[3：0]。所有的数据线都工作在推挽模式。

SDIO_CMD 有两种操作模式：

- 用于初始化时的开路模式（仅用于 MMC 版本 V3.31 或之前版本）
- 用于命令传输的推挽模式（SD/SD I/O 卡和 MMC V4.2 在初始化时也使用推挽驱动）

SDIO_CK 是卡的时钟：每个时钟周期在命令和数据线上传输 1 位命令或数据。对于多媒体卡 V3.31 协议，时钟频率可以在 0 MHz 至 20 MHz 间变化；对于多媒体卡 V4.0/4.2 协议，时钟频率可以在 0 MHz 至 50 MHz 间变化；对于 SD 或 SD I/O 卡，时钟频率可以在 0 MHz 至 50 MHz 间变化。

SDIO 使用一个时钟信号：

- SDIO 适配器时钟（SDIOCLK=HCLK）
- AHB 总线时钟（HCLK）

下表适用于多媒体卡/SD/SD I/O 卡总线：

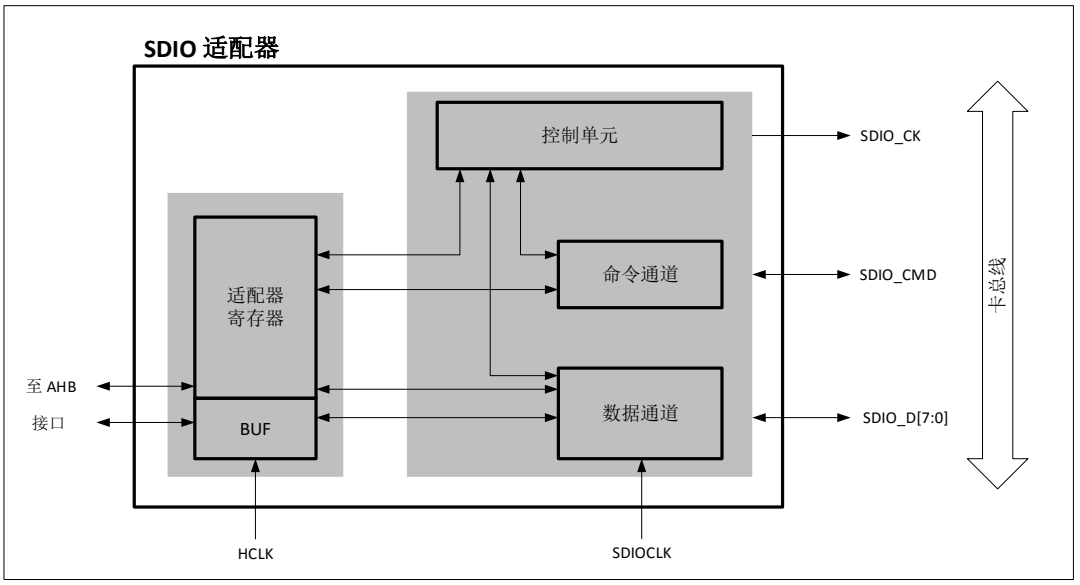
表 20-1 SDIO引脚定义

引脚	方向	说明
SDIO_CK	输出	多媒体卡/SD/SDIO 卡时钟。这是从主机至卡的时钟线。
SDIO_CMD	双向	多媒体卡/SD/SDIO 卡命令。这是双向的命令/响应信号线。
SDIO_D[7：0]	双向	多媒体卡/SD/SDIO 卡数据。这些是双向的数据总线。

20.3.1.1 SDIO适配器

下图是简化的 SDIO 适配器框图：

图 20-7 SDIO适配器



SDIO 适配器是多媒体/加密数字存储卡总线的主设备(主机),用于连接一组多媒体卡或加密数字存储卡,它包含以下 5 个部分:

- 适配器寄存器模块
- 控制单元
- 命令通道
- 数据通道
- 数据 BUF

注意: 适配器寄存器和 BUF 使用 AHB 总线一侧的时钟(HCLK),控制单元、命令通道和数据通道使用 SDIO 适配器一侧的时钟(SDIOCLK)。

适配器寄存器模块

适配器寄存器模块包含所有系统寄存器。该模块还产生清除多媒体卡中静态标记的信号,当在 SDIO 清除寄存器中的相应位写'1'时会产生清除信号。

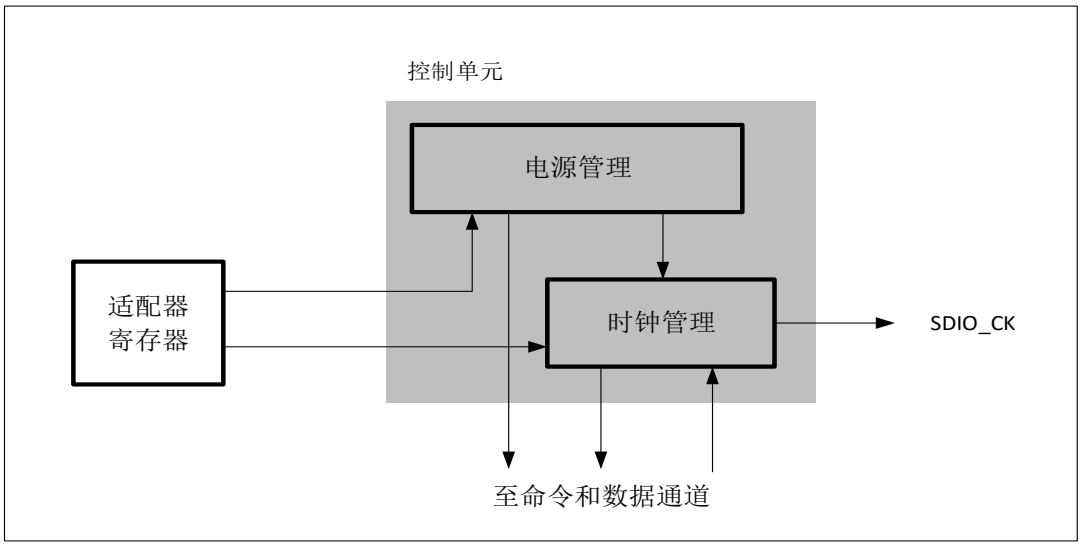
控制单元

控制单元包含电源管理功能和为存储器卡提供的时钟分频。

共有三种电源阶段:

- 电源关闭
- 电源启动
- 电源开

图 20-8 控制单元



上图为控制单元的框图，有电源管理和时钟管理子单元。

在电源关闭和电源启动阶段，电源管理子单元会关闭卡总线上的输出信号。

时钟管理子单元产生和控制 SDIO_CK 信号。SDIO_CK 输出可以使用时钟分频或时钟旁路模式。

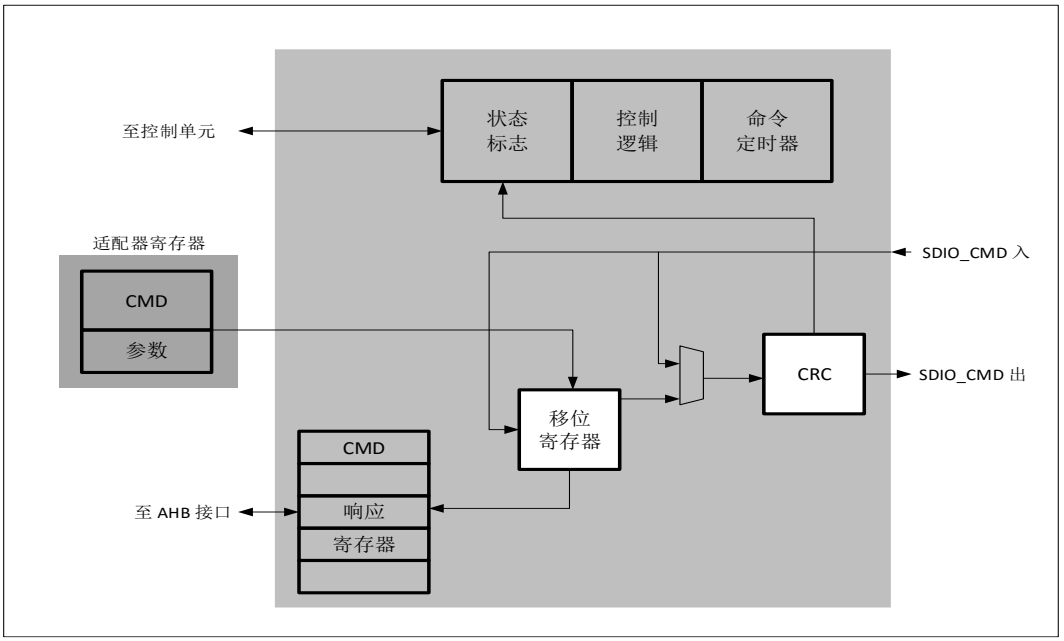
下述情况下没有时钟输出：

- 复位后
- 在电源关闭和电源启动阶段
- 当启动了省电模式并且卡总线处于空闲状态（命令通道和数据通道子单元进入空闲阶段后的 8 个时钟周期）

命令通道

命令通道单元向卡发送命令并从卡接收响应。

图 20-9 SDIO适配器命令通道

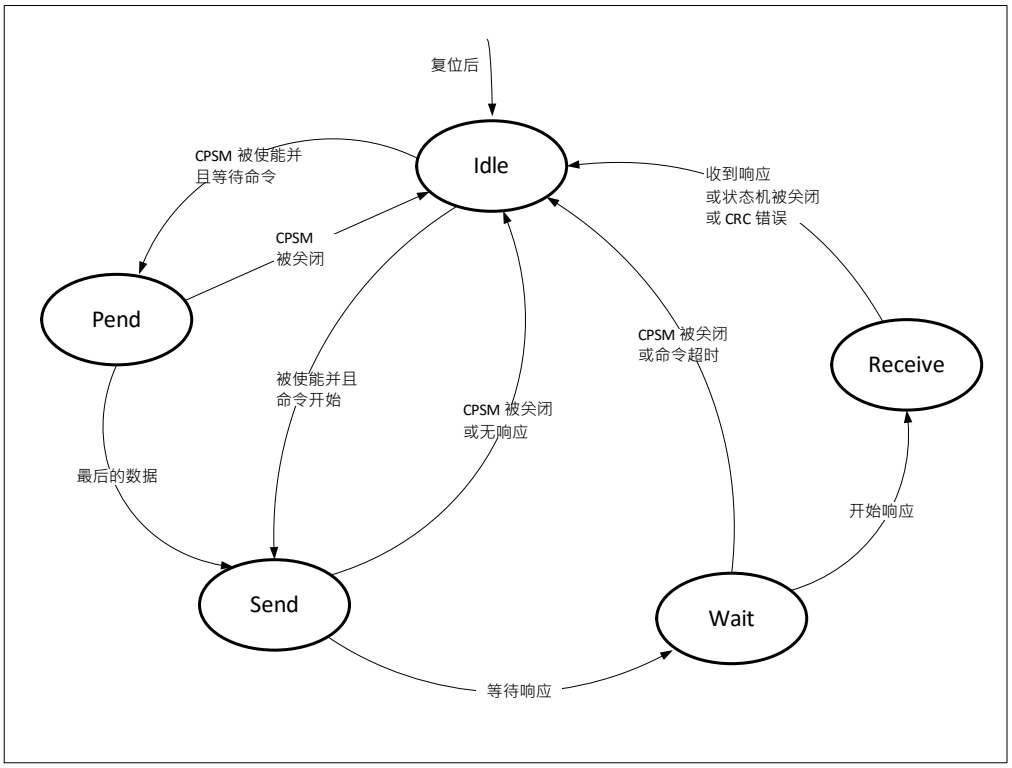


- 命令通道状态机（CPSM）

  —当写入命令寄存器并设置了使能位，开始发送命令。命令发送完成时，命令通道状态机（CPSM）设置状态标志并在不需要响应时进入空闲状态（见图 20-10）。当收到响应后，接收到的 CRC

码将会与内部产生的 CRC 码比较，然后设置相应的状态标志。

图 20-10 命令通道状态机（CPSM）



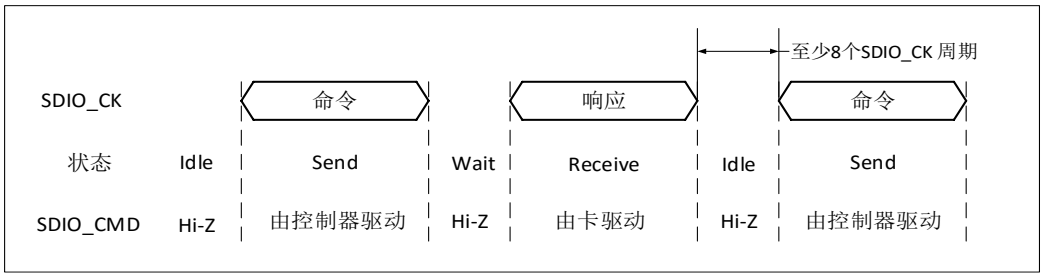
当进入等待（Wait）状态时，命令定时器开始运行；当 CPSM 进入接收（Receive）状态之前，产生了超时，则设置超时标志并进入空闲（Idle）状态。

注意： 命令超时固定为 64 个 SDIO_CK 时钟周期。

如果在命令寄存器设置了中断位，则关闭定时器，CPSM 等待某一个卡发出的中断请求。如果命令寄存器中设置挂起位，CPSM 进入挂起（Pend）状态并等待数据通道子单元发出的 CmdPend 信号，在检测到 CmdPend 信号时，CPSM 进入发送（Send）状态，这将触发数据计数器发送停止命令的功能。

注意： CPSM 保持在空闲状态至少 8 个 SDIO_CK 周期，以满足 N_{CC} 和 N_{RC} 时序限制。N_{CC} 是两个主机命令间的最小间隔；N_{RC} 是主机命令与卡响应之间的最小间隔。

图 20-11 SDIO命令传输



- 命令格式
  - 命令：命令是用于开始一项操作。主机向一个指定的卡或所有的卡发出带地址的命令或广播命令（广播命令只适合于 MMCV3.31 或之前的版本）。命令在 CMD 线上串行传送。所有命令的长度固定为 48 位，下表给出了多媒体卡、SD 存储卡和 SDIO 卡上一般的命令格式。命令通道操作于半双工模式，这样命令和响应可以分别发送和接收。如果 CPSM 不处在发送状态，SDIO_CMD 输出处于高阻状态，如图 20-11 所示。SDIO_CMD 上的数据与 SDIO_CK 的上升沿同步。

表 20-2 命令格式

位	宽度	数值	说明
47	1	0	开始位
46	1	1	传输位
[45：40]	6	-	命令索引
[39：8]	32	-	参数
[7：1]	7	-	CRC7
0	1	1	结束位

— 响应：响应是由一个被指定地址的卡发送到主机，对于 MMCV3.31 或以前版本所有的卡同时发送响应；响应是对先前接收到命令的一个应答。响应在 CMD 线上串行传送。

SDIO 支持 2 种响应类型，2 种类型都有 CRC 错误检测：

- 48 位短响应
- 136 位长响应

注： 如果响应不包含 CRC（如 CMD1 的响应），设备驱动应该忽略 CRC 失败状态。

表 20-3 短响应格式

位	宽度	数值	说明
47	1	0	开始位
46	1	0	传输位
[45：40]	6	-	命令索引
[39：8]	32	-	参数
[7：1]	7	-	CRC7（或 1111111）
0	1	1	结束位

表 20-4 长响应格式

位	宽度	数值	说明
135	1	0	开始位
134	1	0	传输位
[133：128]	6	111111	保留
[127：1]	127	-	CID 或 CSD（包含内部 CRC7）
0	1	1	结束位

命令寄存器包含命令索引（发至卡的 6 位）和命令类型；命令本身决定了是否需要响应和响应的类型是 48 位还是 136 位（见 20.4.4 节）。命令通道中的状态标志示于下表：

表 20-5 命令通道状态标志

标志	说明
CMDRSPCMPL	响应的 CRC 正确
CMDFAIL	响应的 CRC 错误
CMDCMPL	命令（不需要响应的命令）已经送出
CMDTIMEOUT	响应超时
DOCMD	正在发送命令

CRC 发生器计算 CRC 码之前所有位的 CRC 校验和，包括开始位、发送位、命令索引和命令参数（或卡状态）。对于长响应格式，CRC 校验和计算的是 CID 或 CSD 的前 120 位。

注意：长响应格式中的开始位、传输位和 6 个保留位不参与 CRC 计算。

CRC 校验和是一个 7 位的数值：

CRC[6：0]=余数[（M（x）*x^7）/G（x）]

G（x）=x^7+x^3+1

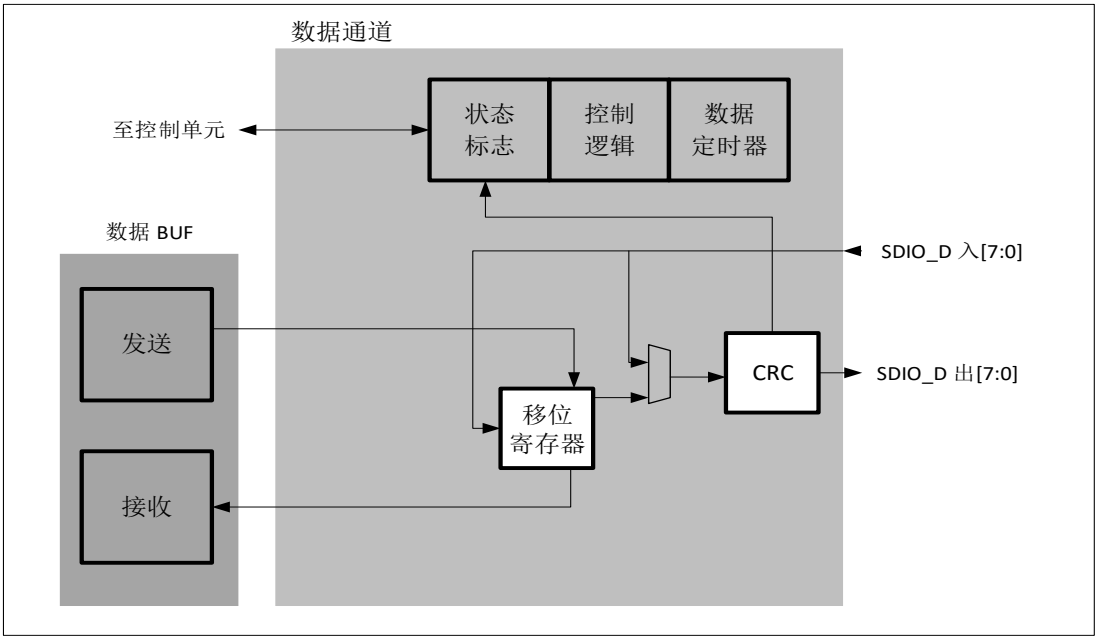
M（x）=（开始位）*x^{39}+…+（CRC 前的最后一位）*x^0,或

M（x）=（开始位）*x^{119}+…+（CRC 前的最后一位）*x^0

数据通道

数据通道子单元在主机与卡之间传输数据。下图是数据通道的框图。

图 20-12 数据通道



在时钟控制寄存器中可以配置卡的数据总线宽度。如果选择了 4 位总线模式，则每个时钟周期四条数据信号线（SDIO_D[3：0]）上将传输 4 位数据；如果选择了 8 位总线模式，则每个时钟周期八条数据信号线（SDIO_D[7：0]）上将传输 8 位数据；如果没有选择宽总线模式，则每个时钟周期只在 SDIO_D0 上传输 1 位数据。

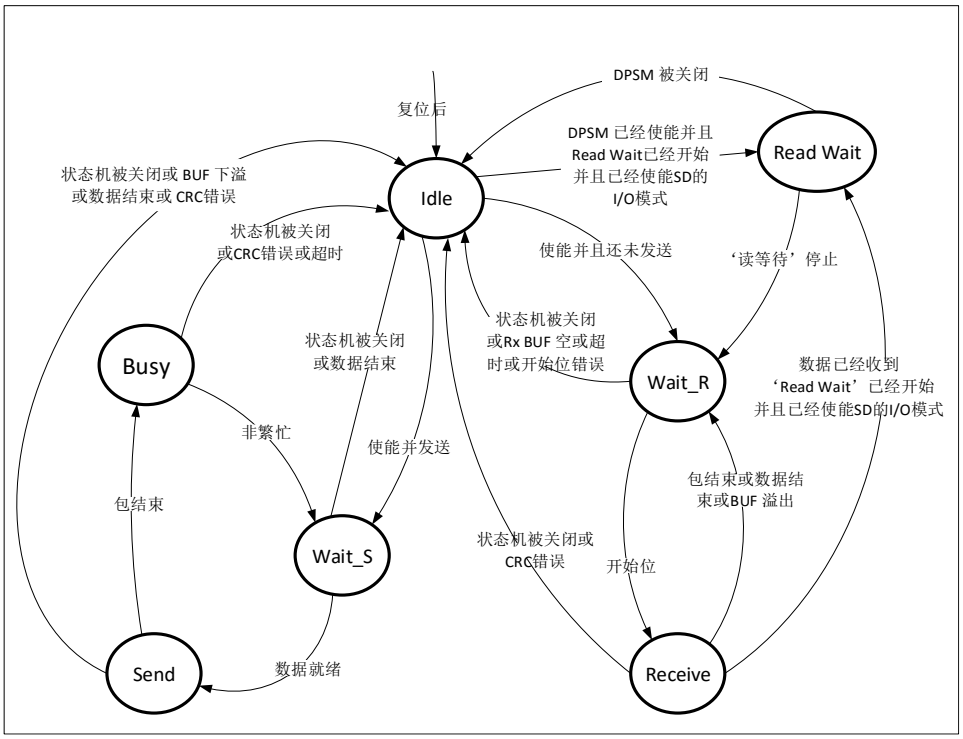
根据传输的方向（发送或接收），使能时数据通道状态机（DPSM）将进入 Wait_S 或 Wait_R 状态：

- 发送：DPSM 进入 Wait_S 状态。如果发送 BUF 中有数据，则 DPSM 进入发送状态，同时数据通道子单元开始向卡发送数据
- 接收：DPSM 进入 Wait_R 状态并等待开始位；当收到开始位时，DPSM 进入接收状态，同时数据通道子单元开始从卡接收数据

数据通道状态机（DPSM）

DPSM 工作在 SDIO_CK 频率，卡总线信号与 SDIO_CK 的上升沿同步。DPSM 有 6 个状态，如下图所示：

图 20-13 数据通道状态机（DPSM）



- 空闲（Idle）：数据通道不工作，SDIO_D[7：0]输出处于高阻状态。当写入数据控制寄存器并设置使能位时，DPSM 为数据计数器加载新的数值，并依据数据方向位进入 Wait_S 或 Wait_R 状态。
- Wait_R：如果数据计数器等于 0，当接收 BUF 为空时 DPSM 进入到空闲（Idle）状态。如果数据计数器不等于 0，DPSM 等待 SDIO_D 上的开始位。如果 DPSM 在超时之前接收到一个开始位，它会进入接收（Receive）状态并加载数据块计数器。如果 DPSM 在检测到一个开始位前出现超时，或发生开始位错误，DPSM 将进入空闲状态并设置超时状态标志。
- 接收（Receive）：接收到的串行数据被组合为字节并写入数据 BUF。根据数据控制寄存器中传输模式位的设置，数据传输模式可以是块传输或流传输：
  - 在块模式下，当数据块计数器达到 0 时，DPSM 等待接收 CRC 码，如果接收到的代码与内部产生的 CRC 码匹配，则 DPSM 进入 Wait_R 状态，否则设置 CRC 失败状态标志同时 DPSM 进入到空闲状态。
  - 在流模式下，当数据计数器不为 0 时，DPSM 接收数据；当计数器为 0 时，将移位寄存器中的剩余数据写入数据 BUF，同时 DPSM 进入 Wait_R 状态。

  如果产生了 BUF 上溢错误，DPSM 设置 BUF 的错误标志并进入 Wait_R 状态。
- Wait_S：如果数据计数器为 0，DPSM 进入空闲状态；否则 DPSM 等待数据 BUF 空标志消失后，进入发送状态。

注： *DPSM 会在 Wait_S 状态保持至少 2 个时钟周期，以满足 N_{WR} 的时序要求，N_{WR} 是接收到卡的响应至主机开始数据传输的间隔。*

- 发送（Send）：DPSM 开始发送数据到卡设备。根据数据控制寄存器中传输模式位的设置，数据传输模式可以是块传输或流传输：
  - 在块模式下，当数据块计数器达到 0 时，DPSM 发送内部产生的 CRC 码，然后是结束位，并进入繁忙状态。
  - 在流模式下，当使能位为高同时数据计数器不为 0 时，DPSM 向卡设备发送数据，然后进入空闲状态。

  如果产生了 BUF 下溢错误，DPSM 设置 BUF 的错误标志并进入空闲状态。

- 繁忙（Busy）：DPSM 等待 CRC 状态标志：
  - 如果没有接收到正确的 CRC 状态，则 DPSM 进入空闲状态并设置 CRC 失败状态标志。
  - 如果接收到正确的 CRC 状态，则当 SDIO_D0 不为低时（卡不繁忙）DPSM 进入 Wait_S 状态。

  当 DPSM 处于繁忙状态时发生了超时，DPSM 则设置数据超时标志并进入空闲状态。

  当 DPSM 处于 Wait_R 或繁忙状态时，数据定时器被使能，并能够产生数据超时错误：
  - 发送数据时，如果 DPSM 处于繁忙状态超过程序设置的超时间隔，则产生超时。
  - 接收数据时，如果未收完所有数据，并且 DPSM 处于 Wait_R 状态超过程序设置的超时间隔，则产生超时。
- 数据：数据可以从主机传送到卡，也可以反向传输。数据在数据线上传输。数据存储在一个 32 字的 BUF 中，每个字为 32 位宽。

表 20-6 数据令牌格式

说明	开始位	数据	CRC16	结束位
块数据	0	-	有	1
流数据	0	-	无	1

数据 BUF

数据 BUF（先进先出）子单元是一个具有发送和接收单元的数据缓冲区。

BUF 包含一个每字 32 位宽、共 32 个字的数据缓冲区，和发送与接收电路。因为数据 BUF 工作在 AHB 时钟区域（HCLK），所有与 SDIO 时钟区域（SDIOCLK）连接的信号都进行了重新同步。

依据 DOTX 和 DORX 标志，可以关闭 BUF、使能发送或使能接收。DOTX 和 DORX 由数据通道子单元设置而且是互斥的：

- 当 DOTX 有效时，发送 BUF 代表发送电路和数据缓冲区
- 当 DORX 有效时，接收 BUF 代表接收电路和数据缓冲区

- 发送 BUF：当使能了 SDIO 的发送功能，数据可以通过 AHB 接口写入发送 BUF。

  发送 BUF 有 32 个连续的地址。发送 BUF 中有一个数据输出寄存器，包含读指针指向的数据字。当数据通道子单元装填了移位寄存器后，它移动读指针至下个数据并传输出数据。

  如果未使能发送 BUF，所有的状态标志均处于无效状态。当发送数据时，数据通道子单元设置 DOTX 为有效。

表 20-7 发送BUF状态标志

标志	说明
TXBUF_F	当所有 32 个发送 BUF 字都有有效的数据时，该标志为高。
TXBUF_E	当所有 32 个发送 BUF 字都没有有效的数据时，该标志为高。
TXBUF_H	当 8 个或更多发送 BUF 字为空时，该标志为高。该标志可以作为 DMA 请求。
TXBUF	当发送 BUF 包含有效数据时，该标志为高。该标志的意思刚好与 TXBUF_E 相反。
TXERRU	当发生下溢错误时，该标志为高。写入 SDIO 清除寄存器时清除该标志。

- 接收 BUF：当数据通道子单元接收到一个数据字，它会把数据写入 BUF，写操作结束后，写指针自动加一；在另一端，有一个读指针始终指向 BUF 中的当前数据。如果关闭了接收 BUF，所有的状态标志会被清除，读写指针也被复位。在接收到数据时数据通道子单元设置 DORX。下表列出了接收 BUF 的状态标志。通过 32 个连续的地址可以访问接收 BUF。

表 20-8 接收BUF状态标志

标志	说明
RXBUF_F	当所有 32 个接收 BUF 字都有有效的数据时，该标志为高。

RXBUF_E	当所有 32 个接收 BUF 字都没有有效的数据时，该标志为高。
RXBUF_H	当 8 个或更多接收 BUF 字有有效的数据时，该标志为高。该标志可以作为 DMA 请求。
RXBUF	当接收 BUF 包含有效数据时，该标志为高。该标志的意思刚好与 RXBUF_E 相反。
RXERRO	当发生上溢错误时，该标志为高。写入 SDIO 清除寄存器时清除该标志。

20.3.1.2 SDIO AHB接口

AHB 接口产生中断和 DMA 请求，并访问 SDIO 接口寄存器和数据 BUF。它包含一个数据通道、寄存器译码器和中断/DMA 控制逻辑。

SDIO 中断

当至少有一个选中的状态标志为高时，中断控制逻辑产生中断请求。有一个屏蔽寄存器用于选择可以产生中断的条件，如果设置了相应的屏蔽标志，则对应的状态标志可以产生中断。

SDIO/DMA 接口：在 SDIO 和存储器之间数据传输的过程

在下面的例子中，主机控制器使用 CMD24（WRITE_BLOCK）从主机传送 512 字节到 MMC 卡，DMA 控制器用于从存储器向 SDIO 的 BUF 填充数据。

1. 执行卡识别过程
2. 提高SDIO_CK频率
3. 发送CMD7命令选择卡
4. 按下述步骤配置DMA2：
   a）使能 DMA2 控制器并清除所有的中断标志位
   b）设置 DMA2 通道 4 的源地址寄存器为存储器缓冲区的基地址，DMA2 通道 4 的目标地址寄存器为 SDIO_BUF 寄存器的地址
   c）设置 DMA2 通道 4 控制寄存器（存储器递增，非外设递增，外设和源的数据宽度为字宽度）
   d）使能 DMA2 通道 4
5. 发送CMD24（WRITE_BLOCK），操作如下：
   a）设置 SDIO 数据长度寄存器（SDIO 数据时钟寄存器应该在执行卡识别过程之前设置好）
   b）设置 SDIO 参数寄存器为卡中需要传送数据的地址
   c）设置 SDIO 命令寄存器：CmdIndex 置为 24（WRITE_BLOCK）；RSPWT 置为 1（SDIO 卡主机等待响应）；CMDMEN 置为 1（使能 SDIO 卡主机发送命令），保持其它域为他们的复位值。
   d）等待 SDIO_STS[6]=CMDRSPCMPL 中断，然后设置 SDIO 数据控制寄存器：TFREN 置为 1（使能 SDIO 卡主机发送数据）；TFRDIR 置为 0（控制器至卡方向）；TFRMODE 置为 0（块数据传送）；DMAEN 置为 1（DMA 使能）；BLKSIZE 置为 9（512 字节）；其它域不用设置
   e）等待 SDIO_STS[10]=DTBLKCMPL
6. 查询DMA通道的使能状态寄存器，确认没有通道仍处于使能状态

20.3.2 卡功能描述

20.3.2.1 卡识别模式

在卡识别模式，主机复位所有的卡、检测操作电压范围、识别卡并为总线上每个卡设置相对地址（RCA）。在卡识别模式下，所有数据通信只使用命令信号线（CMD）。

20.3.2.2 卡复位

GO_IDLE_STATE 命令（CMD0）是一个软件复位命令，它把多媒体卡和 SD 存储器置于空闲状态。IO_RW_DIRECT 命令（CMD52）复位 SDI/O 卡。上电后或执行 CMD0 后，所有卡的输出端都处于高阻状态，同时所有卡都被初始化至一个默认的相对卡地址（RCA=0x0001）和默认的驱动器寄存器设置（最低的速度，最大的电流驱动能力）。

20.3.2.3 操作电压范围确认

所有的卡都可以使用任何规定范围内的电压与 SDIO 卡主机通信，可支持的最小和最大电压 VDD 数值由卡上的操作条件寄存器（OCR）定义。

内部存储器存储了卡识别号（CID）和卡特定数据（CSD）的卡，仅能在数据传输 VDD 条件下传送这些信息。当 SDIO 卡主机模块与卡的 VDD 范围不一致时，卡将不能完成识别周期，也不能发送 CSD 数据；因此，在 VDD 范围不匹配时，SDIO 卡主机可以用下面几个特殊命令去识别和拒绝卡：SEND_OP_COND（CMD1）、SD_APP_OP_COND（SD 存储卡的 ACMD41）和 IO_SEND_OP_COND （SDI/O 卡的 CMD5）。SDIO 卡主机在执行这几个命令时会产生需要的 VDD 电压。不能在指定的电压范围进行数据传输的卡，将从总线断开并进入非激活状态。

使用这些不包含电压范围作为操作数的命令，SDIO 卡主机能够查询每个卡并在确定公共的电压范围前，把不在此范围内的卡置于非激活状态。当 SDIO 卡主机能够选择公共的电压范围或用户需要知道卡是否能用时，SDIO 卡主机可以进行这样的查询。

20.3.2.4 卡识别过程

多媒体卡和 SD 卡的卡识别过程是有区别的；对于多媒体卡，卡识别过程以时钟频率 F_{od} 开始，所有 SDIO_CMD 输出为开路驱动，允许在这个过程中的卡的并行连接，识别过程如下：

1. 总线被激活
2. SDIO卡主机广播发送SEND_OP_COND（CMD1）命令，并接收操作条件
3. 得到的响应是所有卡的操作条件寄存器内容的“线与”
4. 不兼容的卡会被置于非激活状态
5. SDIO卡主机广播发送ALL_SEND_CID（CMD2）至所有激活的卡
6. 所有激活的卡同时串行地发送他们的CID号，那些检测到输出的CID位与命令线上的数据不相符的卡必须停止发送，并等待下一个识别周期。最终只有一个卡能够成功地传送完整的CID至SDIO卡主机并进入识别状态
7. SDIO卡主机发送SET_RELATIVE_ADDR（CMD3）命令至这个卡，这个新的地址被称为相对卡地址（RCA），它比CID短，用于对卡寻址。至此，这个卡转入待机状态，并不再响应新的识别过程，同时它的输出驱动从开路转变为推挽模式
8. SDIO卡主机重复上述步骤5至7，直到收到超时条件

对于 SD 卡而言，卡识别过程以时钟频率 F_{od} 开始，所有 SDIO_CMD 输出为推挽驱动而不是开路驱动，识别过程如下：

1. 总线被激活
2. SDIO卡主机广播发送SEND_APP_OP_COND（ACMD41）命令
3. 得到的响应是所有卡的操作条件寄存器的内容
4. 不兼容的卡会被置于非激活状态
5. SDIO卡主机广播发送ALL_SEND_CID（CMD2）至所有激活的卡
6. 所有激活的卡发送回他们唯一卡识别号（CID）并进入识别状态
7. SDIO卡主机发送SET_RELATIVE_ADDR（CMD3）命令和一个地址到一个激活的卡，这个新的地址被称为相对卡地址（RCA），它比CID短，用于对卡寻址。至此，这个卡转入待机状态。SDIO卡主机可以再次发送该命令更改RCA，卡的RCA将是最后一次的赋值
8. SDIO卡主机对所有激活的卡重复上述步骤5至7

对于 SD I/O 卡而言，卡识别过程如下：

1. 总线被激活
2. SDIO卡主机发送IO_SEND_OP_COND（CMD5）命令
3. 得到的响应是卡的操作条件寄存器的内容

4. 不兼容的卡会被置于非激活状态
5. SDIO卡主机发送SET_RELATIVE_ADDR（CMD3）命令和一个地址到一个激活的卡，这个新的地址被称为相对卡地址（RCA），它比CID短，用于对卡寻址。至此，这个卡转入待机状态。SDIO卡主机可以再次发送该命令更改RCA，卡的RCA将是最后一次的赋值

20.3.2.5 写数据块

执行写数据块命令（CMD24-27）时，主机把一个或多个数据块从主机传送到卡中，同时在每个数据块的末尾传送一个 CRC 码。一个支持写数据块命令的卡应该始终能够接收由 WRITE_BL_LEN 定义的数据块。如果 CRC 校验错误，卡通过 SDIO_D 信号线指示错误，传送的数据被丢弃而不被写入，所有后续（在多块写模式下）传送的数据块将被忽略。

如果主机传送部分数据，而累计的数据长度未与数据块对齐，当不允许块错位（未设置 CSD 的参数 WRITE_BLK_MISALIGN），卡将在第一个错位的块之前检测到块错位错误（设置状态寄存器中的 ADDRESS_ERROR 错误位）。当主机试图写一个写保护区域时，写操作也会被中止，此时卡会设置 WP_VIOLATION 位。

设置 CID 和 CSD 寄存器不需要事先设置块长度，传送的数据也是通过 CRC 保护的。如果 CSD 或 CID 寄存器的部分是存储在 ROM 中，则这个不能更改的部分必须与接收缓冲区的对应部分相一致，如果有不一致之处，卡将报告一个错误同时不修改任何寄存器的内容。有些卡需要长的甚至不可预计的时间完成写一个数据块，在接收一个数据块并完成 CRC 检验后，卡开始写操作，如果它的写缓冲区已经满并且不能再从新的 WRITE_BLOCK 命令接受新的数据时，它会把 SDIO_D 信号线拉低。主机可以在任何时候使用 SEND_STATUS（CMD13）查询卡的状态，卡将返回当前状态。READY_FOR_DATA 状态位指示卡是否可以接受新的数据或写操作是否还在进行。主机可以使用 CMD7（选择另一个卡）不选中某个卡，而把这个卡置于断开状态，这样可以释放 SDIO_D 信号线而不中断未完成的写操作；当重新选择了一个卡，如果写操作仍然在进行并且写缓冲区仍不能使用，它会重新通过拉低 SDIO_D 信号线指示忙的状态。

20.3.2.6 读数据块

在读数据块模式下，数据传输的基本单元是数据块，它的大小在 CSD 中（READ_BL_LEN）定义。如果设置了 READ_BL_PARTIAL，同样可以传送较小的数据块，较小数据块是指开始和结束地址完全包含在一个物理块中，READ_BL_LEN 定义了物理块的大小。为保证数据传输的正确，每个数据块后都有一个 CRC 校验码。CMD17（READ_SINGLE_BLOCK）启动一次读数据块操作，在传输结束后卡返回到发送状态。

CMD18（READ_MULTIPLE_BLOCK）启动一次连续多个数据块的读操作。

主机可以在多数据块读操作的任何时候中止操作，而不管操作的类型。发送停止传输命令即可中止操作。

如果在多数据块读操作中（任一种类型）卡检测到错误（例如：越界、地址错位或内部错误），它将停止数据传输并仍处于数据状态；此时主机必须发送停止传输命令中止操作。在停止传输命令的响应中报告读错误。

如果主机发送停止传输命令时，卡已经传输完一个确定数目的多个数据块操作中的最后一个数据块，因为此时卡已经不在数据状态，主机会得到一个非法命令的响应。如果主机传输部分数据块，而累计的数据长度不能与物理块对齐同时不允许块错位，卡会在出现第一个未对齐的块时检测出一个块对齐错误，并在状态寄存器中设置 ADDRESS_ERROR 错误标志。

20.3.2.7 数据流操作，数据流写入和数据流读出（只适用于多媒体卡）

在数据流模式，数据按字节传输，同时每个数据块后没有 CRC。

数据流写（只适用于多媒体卡）

WRITE_DAT_UNTIL_STOP（CMD20）开始从 SDIO 卡主机至卡的数据传输，从指定的地址开始连续传输直到 SDIO 卡主机发出一个停止命令。如果允许部分数据块传输（设置了 CSD 参数 WRITE_BL_PARTIAL），则数据流可以在卡的地址空间中的任意地址开始和停止，否则数据流只能在数据块的边界开始和停止。因为传输的数据数目没有事先设定，不能使用 CRC 校验。如果发送数据时达到了存储器的最大地址，即使 SDIO 卡主机没有发送停止命令，随后传输的数据也会被丢弃。

数据流写操作的最大时钟频率可以通过下式计算

$$Max\ Write\ Frequency = Min\left(TRANSPEED, \frac{(8 \times 2^{writebllen}) - 100 * NSAC}{TAAC \times \text{R2WFACTOR}}\right)$$

- Max Write Frequency = 最大写频率
- TRANSPEED = 最大数据传输率
- writebllen = 最大写数据块长度
- NSAC = 以 CLK 周期计算的数据读操作时间 2
- TAAC = 数据读操作时间 1
- R2WFACTOR = 写速度因子

如果主机试图使用更高的频率，卡可能不能处理数据并停止编程，同时在状态寄存器中设置 OVERRUN 错误位，丢弃所有随后传输的数据并（在接收数据状态）等待停止命令。如果主机试图写入一个写保护区域，写操作将被中止，同时卡将设置 WP_VIOLATION 位。

数据流读（只适用于多媒体卡）

READ_DAT_UNTIL_STOP（CMD11）控制数据流数据传输。

这个命令要求卡从指定的地址读出数据，直到 SDIO 卡主机发送 STOP_TRANSMISSION（CMD12）。因为串行命令传输的延迟，停止命令的执行会有延迟，数据传送会在停止命令的结束位后停止。如果发送数据时达到了存储器的最大地址，SDIO 卡主机没有发送停止命令，随后传输的数据将是无效数据。

数据流读操作的最大时钟频率可以通过下式计算

$$Max\ Read\ Frequency = Min\left(TRANSPEED, \frac{(8 \times 2^{readbllen}) - 100 * NSAC}{TAAC \times \text{R2WFACTOR}}\right)$$

- Max Read Frequency = 最大读频率
- TRANSPEED = 最大数据传输率
- readbllen = 最大读数据块长度
- NSAC = 以 CLK 周期计算的数据读操作时间 2
- TAAC = 数据读操作时间 1
- R2WFACTOR = 写速度因子

如果主机试图使用更高的频率，卡将不能处理数据传输，此时卡在状态寄存器中设置 UNDERRUN 错误位，中止数据传输并在数据状态等待停止命令。

20.3.2.8 擦除：成组擦除和扇区擦除

多媒体卡的擦除单位是擦除组，擦除组是以写数据块计算，写数据块是卡的基本写入单位。擦除组的大小是卡的特定参数，在 CSD 中定义。

主机可以擦除一个连续范围的擦除组，开始擦除操作有三个步骤。

首先，主机使用 ERASE_GROUP_START（CMD35）命令定义连续范围的开始地址，然后使用 ERASE_GROUP_END（CMD36）命令定义连续范围的结束地址，最后发送擦除命令 ERASE（CMD38）开始擦除操作。擦除命令的地址域是以字节为单位的擦除组地址。卡会舍弃未与擦除组大小对齐的部分，把地址边界对齐到擦除组的边界。

如果未按照上述步骤收到了擦除命令，卡在状态寄存器中设置 ERASE_SEQ_ERROR 位，并重新等待第一个步骤。

如果收到了除 SEND_STATUS 和擦除命令之外的其它命令，卡在状态寄存器中设置 ERASE_RESET 位，解除擦除序列并执行新的命令。

如果擦除范围包含了写保护数据块，这些块不被擦除，只有未保护的块被擦除，同时卡在状态寄存器中设置 WP_ERASE_SKIP 状态位。

在擦除过程中，卡拉低 SDIO_D 信号。实际的擦除时间可能很长，主机可以使用 CMD7 解除卡的选择。

20.3.2.9 宽总线选择和解除选择

可以通过 SET_BUS_WIDTH（ACMD6）命令选择或不选择宽总线（4 位总线宽度）操作模式，上电后或 GO_IDLE_STATE（CMD0）命令后默认的总线宽度为 1 位。SET_BUS_WIDTH（ACMD6）命令仅在传输状态时有效，即只有在使用 SELECT/DESELECT_CARD（CMD7）命令选择了卡后才能改变总线宽度。

20.3.2.10 保护管理

SDIO 卡主机模块支持三种保护方式：

1. 内部卡保护（卡内管理）
2. 机械写保护开关（仅由SDIO卡主机模块管理）
3. 密码管理的卡锁操作

内部卡的写保护

卡的数据可以被保护不被覆盖或擦除。在 CSD 中永久地或临时地设置写保护位，生产厂商或内容提供商可以永久地对整个卡施行写保护。对于支持在 CSD 中设置 WP_GRP_ENABLE 位从而提供一组扇区写保护的卡，部分数据可以被保护，写保护可以通过程序改变。写保护的基本单位是 CSD 参数 WP_GRP_SIZE 个扇区。SET_WRITE_PROT 和 CLR_WRITE_PROT 命令控制指定组的保护，SEND_WRITE_PROT 命令与单数据块读命令类似，卡送出一个包含 32 个写保护位（代表从指定地址开始的 32 个写保护组）的数据块，跟着一个 16 位的 CRC 码。写保护命令的地址域是一个以字节为单位的组地址。

卡将截断所有组大小以下的地址。

机械写保护开关

在卡的侧面有一个机械的滑动开关，允许用户设置或清除卡的写保护。当滑动开关置于小窗口打开的位置时，卡处于写保护状态，当滑动开关置于小窗口关闭的位置时，可以更改卡中内容。在卡的插槽上的对应部位也有一个开关指示 SDIO 卡主机模块，卡是否处于写保护状态。卡的内部电路不知道写保护开关的位置。

密码保护

密码保护功能允许 SDIO 卡主机模块使用密码对卡实行上锁或解锁。密码存储在 128 位的 PWD 寄存器中，它的长度设置在 8 位的 PWD_LEN 寄存器中。这些寄存器是不可挥发的，即掉电后它们的内容不丢失。已上锁的卡能够响应和执行相应的命令，即允许 SDIO 卡主机模块执行复位、初始化和查询状态等操作，但不允许操作卡中的数据。当设置了密码后（即 PWD_LEN 的数值不为 0），上电后卡自动处于上锁状态。正如 CSD 和 CID 寄存器写命令，上锁/解锁命令仅在传输状态下有效，在这个状态下，命令中没有地址参数，但卡已经被选中。卡的上锁/解锁命令具有单数据块写命令的结构和总线操作类型，传输的数据块包含所有命令所需要的信息（密码设置模式、PWD 内容和上锁/解锁指示）。在发送卡的上锁/解锁命令之前，命令数据块的长度由 SDIO 卡主机模块定义，命令结构示于表 20-22。

位的设置如下：

- ERASE：设置该位将执行强制擦除，所有其它位必须为 0，只发送命令字节
- LOCK_UNLOCK：设置该位锁住卡，LOCK_UNLOCK 与 SET_PWD 可以同时设置，但不能与 CLR_PWD 同时设置
- CLR_PWD：设置该位清除密码数据
- SET_PWD：设置该位将密码数据保存至存储器
- PWD_LEN：以字节为单位定义密码的长度
- PWD：密码（依不同的命令，新的密码或正在使用的密码）

以下几节列出了设置/清除密码、上锁/解锁和强制擦除的命令序列。

设置密码

1. 选择一个卡（SELECT/DESELECT_CARD，CMD7）
2. 定义要在8位的卡上锁/解锁模式下发送的数据块长度（SET_BLOCKLEN，CMD16），8位的PWD_LEN，新密码的字节数目。当更换了密码后，发送命令的数据块长度必须同时考虑新旧密码的长度。

3. 以合适的数据块长度在数据线上发送LOCK/UNLOCK（CMD42）命令，并包含16位的CRC码。数据块包含了操作模式（SET_PWD=1）、长度（PWD_LEN）和密码（PWD）。当更换了密码后，长度数值（PWD_LEN）包含了新旧两个密码的长度，PWD域包含了旧的密码（正在使用的）和新的密码。
4. 当旧的密码匹配后，新的密码和它的长度被分别存储在PWD和PWD_LEN域。如果送出的旧密码与期望的密码（长度或内容）不吻合，则设置状态寄存器中的LOCK_UNLOCK_FAILED错误位，同时密码不变。

密码长度域（PWD_LEN）指示当前是否设置了密码，如果该域为非零，则表示使用了密码，卡在上电时自动上锁。在不断电的情况下，如果设置了密码，可以通过设置 LOCK_UNLOCK 位或发送一个额外的上锁命令，立即锁住卡。

清除密码

1. 选择一个卡（SELECT/DESELECT_CARD，CMD7）
2. 定义要在8位的卡上锁/解锁模式下发送的数据块长度（SET_BLOCKLEN，CMD16），8位的PWD_LEN，当前使用密码的字节数目
3. 当密码匹配后，PWD域被清除同时PWD_LEN被设为0。如果送出的密码与期望的密码（长度或内容）不吻合，则设置状态寄存器中的LOCK_UNLOCK_FAILED错误位，同时密码不变。

卡上锁

1. 选择一个卡（SELECT/DESELECT_CARD，CMD7）
2. 定义要在8位的卡上锁/解锁模式（见表20-22的字节0）下发送的数据块长度（SET_BLOCKLEN，CMD16），8位的PWD_LEN，和当前密码的字节数目
3. 以合适的数据块长度在数据线上发送LOCK/UNLOCK（CMD42）命令，并包含16位的CRC码。数据块包含了操作模式（LOCK_UNLOCK=1）、长度（PWD_LEN）和密码（PWD）。
4. 当密码匹配后，卡被上锁并则设置状态寄存器中的CARD_IS_LOCKED状态位。如果送出的密码与期望的密码（长度或内容）不吻合，则设置状态寄存器中的LOCK_UNLOCK_FAILED错误位，同时上锁操作失败。

设置密码和为卡上锁可以在同一个操作序列中进行，此时 SDIO 卡主机模块按照前述的步骤设置密码，但在发送新密码命令的第 3 步需要设置 LOCK_UNLOCK 位。

如果曾经设置过密码（PWD_LEN 不为 0），卡会在上电复位时自动地上锁。对已经上锁的卡执行上锁操作或对没有密码的卡执行上锁操作会导致失败，并设置状态寄存器中的 LOCK_UNLOCK_FAILED 错误位。

卡解锁

1. 选择一个卡（SELECT/DESELECT_CARD，CMD7）
2. 定义要在8位的卡上锁/解锁模式（见表20-22的字节0）下发送的数据块长度（SET_BLOCKLEN，CMD16），8位的PWD_LEN，和当前密码的字节数目
3. 以合适的数据块长度在数据线上发送LOCK/UNLOCK（CMD42）命令，并包含16位的CRC码。数据块包含了操作模式（LOCK_UNLOCK=0）、长度（PWD_LEN）和密码（PWD）。
4. 当密码匹配后，卡锁被解除，同时状态寄存器中的CARD_IS_LOCKED位被清除。如果送出的密码与期望的密码（长度或内容）不吻合，则设置状态寄存器中的LOCK_UNLOCK_FAILED错误位，同时卡仍保持上锁状态。

解锁状态只在当前的供电过程中有效，只要不清除 PWD 域，下次上电后卡会被自动上锁。

试图对已经解了锁的卡执行解锁操作会导致操作失败，并设置状态寄存器中的 LOCK_UNLOCK_FAILED 错误位。

强制擦除

如果用户忘记了密码（PWD 的内容），可以在清除卡中的所有内容后使用卡。强制擦除操作擦除所有卡中的数据和密码。

1. 选择一个卡（SELECT/DESELECT_CARD，CMD7）
2. 设置发送的数据块长度（SET_BLOCKLEN，CMD16）为1，仅发送8位的卡上锁/解锁字节（见表20-22的字节0）。
3. 以合适的数据块长度在数据线上发送LOCK/UNLOCK（CMD42）命令，并包含16位的CRC码。数据块包含了操作模式（ERASE=1）所有其它位为0。
4. 当ERASE位是数据域中仅有的位时，卡中的所有内容将被擦除，包括PWD和PWD_LEN域，同时卡不再被上锁。如果有任何其它位不为0，则设置状态寄存器中的LOCK_UNLOCK_FAILED错误位，卡中的数据保持不变，同时卡仍保持上锁状态。

试图对已经解了锁的卡执行擦除操作会导致操作失败，并设置状态寄存器中的 LOCK_UNLOCK_FAILED 错误位。

20.3.2.11 卡状态寄存器

响应格式 R1 包含了一个 32 位的卡状态域，这个域是用于向卡主机发送卡的状态信息（这些信息有可能存在本地的状态寄存器中）。除非特别说明，卡返回的状态始终是与之前的命令相关的。

表 20-9 定义了不同的状态信息。表中有关类型和清除条件域的缩写定义如下：

类型：

- E：错误位
- S：状态位
- R：检测位，并依据实际的命令响应而设置
- X：检测位，在命令的执行中设置。SDIO 卡主机通过发送状态命令读出这些位而查询卡的状态

清除条件：

- A：依据卡的当前状态
- B：始终与之前的命令相关。接收到正确的命令即可清除（具有一个命令的延迟）
- C：读即可清除

表 20-9 卡状态

位	名称	类型	数值	说明	清除条件
31	ADDRESS_OUT_OF_RANGE	ERX	‘0’=无错误 ‘1’=错误	命令中的地址参数超出了卡的允许范围。 一个多数据块或数据流读/写操作（即使从一个合法的地址开始）试图读或写超出卡的容量的部分。	C
30	ADDRESS_MISALIGN		‘0’=无错误 ‘1’=错误	命令中的地址参数（与当前的数据块长度对照）定义的第一个数据块未与卡的物理块对齐。 一个多数据块或数据流读/写操作（即使从一个合法的地址开始）试图读或写未与物理块对齐的数据块。	C
29	BLOCK_LEN_ERROR		‘0’=无错误 ‘1’=错误	SET_BLOCKLEN 命令的参数超出了卡的最大允许范围，或先前定义的数据块长度对于当前命令来说是非法的（例如：主机发出一个写命令，当前的块长度小于卡所允许的最小长度，同时又不允许写入部分数据块）。	C
28	ERASE_SEQ_ERROR		‘0’=无错误 ‘1’=错误	发送擦除命令的顺序错误。	C
27	ERASE_PARAM	EX	‘0’=无错误 ‘1’=错误	擦除时选择了非法的擦除组。	C

26	WP_VIOLATION	EX	‘0’=无错误 ‘1’=错误	试图对一个写保护的数据块编程。	C
25	CARD_IS_LOCKED	SR	‘0’=卡未锁 ‘1’=卡已锁	当设置了该位，表示卡已经被锁住。	A
24	LOCK_UNLOCK_FAILED	EX	‘0’=无错误 ‘1’=错误	在上锁/解锁中有命令的顺序错误或检测到密码错误。	C
23	COM_CRC_ERROR	ER	‘0’=无错误 ‘1’=错误	之前的命令中 CRC 校验错误。	B
22	ILLEGAL_COMMAND	ER	‘0’=无错误 ‘1’=错误	对于当前的卡状态，命令非法。	B
21	CARD_ECC_FAILED	EX	‘0’=成功 ‘1’=失败	卡的内部实施了 ECC 校验，但在更正数据时失败。	C
20	CC_ERROR	ER	‘0’=无错误 ‘1’=错误	（标准中未定义）卡内部发生错误，与主机的命令无关。	C
19	ERROR	EX	‘0’=无错误 ‘1’=错误	产生了与执行上一个主机命令相关的（标准中未定义） 卡内部的错误（例如：读或写错误）。	C
18	保留				
17	保留				
16	CID/CSD_OVERWRITE	EX	‘0’=无错误 ‘1’=错误	可以是任何一个下述的错误： ▪已经写入了 CID 寄存器，不能覆盖 ▪CSD 的只读部分与卡的内容不匹配 ▪试图进行拷贝或永久写保护的反向操作，即恢复原状或解除写保护。	C
15	WP_ERASE_SKIP	EX	‘0’=未保护 ‘1’=已保护	遇到已经存在的写保护数据块，仅有部分地址空间被擦除。	C
14	CARD_ECC_DISABLED	SX	‘0’=允许 ‘1’=不允许	执行命令时没有使用内部的 ECC。	A
13	ERASE_RESET		‘0’=清除 ‘1’=设置	因为收到一个擦除顺序之外的命令（非 CMD35、CMD36、CMD38 或 CMD13 命令），进入擦除过程的序列被中止。	C
12：9	CURRENT_STATE	SR	0=空闲 1=就绪 2=识别 3=待机 4=发送 5=数据 6=接收 7=编程 8=断开 9=忙测试 10~15=保留	当收到命令时卡内状态机的状态。如果命令的执行导致状态的变化，这个变化将会在下个命令的响应中反映出来。这四个位按十进制数 0 至 15 解释。	B
8	READY_FOR_DATA	SR	‘0’=未就绪 ‘1’=就绪	与总线上的缓冲器空的信号相对应。	
7	SWITCH_ERROR	EX	‘0’=无错误 ‘1’=转换错	卡没有按照 SWITCH 命令的要求转换到希望的模式。	B

6	保留				
5	APP_CMD	SR	‘0’=不允许 ‘1’=允许	卡期望 ACMD，或指示命令已经被解释为 ACMD 命令。	C
4	保留给 SD I/O 卡				
3	AKE_SEQ_ERROR	ER	‘0’=无错误 ‘1’=错误	验证的顺序有错误。	C
2	保留给与应用相关的命令。				
1,0	保留给生产厂家的测试模式。				

20.3.2.12 SD状态寄存器

SD 状态包含与 SD 存储器卡特定功能相关的状态位和一些与未来应用相关的状态位，SD 状态的长度是一个 512 位的数据块。收到 ACMD13 命令（CMD55，然后是 CMD13）后，这个寄存器的内容被传送到 SDIO 卡主机。只有卡处于传输状态时（卡已被选择）才能发送 ACMD13 命令。

表 20-10 定义了不同的 SD 状态寄存器信息。表中有关类型和清除条件域的缩写定义如下：

类型：

- E：错误位
- S：状态位
- R：检测位，并依据实际的命令响应而设置
- X：检测位，在命令的执行中设置。SDIO 卡主机通过发送状态命令读出这些位而查询卡的状态

清除条件：

- A：依据卡的当前状态
- B：始终与之前的命令相关。接收到正确的命令即可清除（具有一个命令的延迟）
- C：读即可清除

表 20-10 SD状态

位	名称	类型	数值	说明	清除条件
511：510	DAT_BUS_WIDTH	SR	‘00’=1（默认） ‘01’=保留 ‘10’=4 位宽 ‘11’=保留	由 SET_BUS_WIDTH 命令定义的当前数据总线宽度。	A
509	SECURED_MODE	SR	‘0’=未处于保密模式 ‘1’=处于保密模式	卡处于保密操作模式（详见“SD 保密规范”）。	A
508：496	保留				
495：480	SD_CARD_TYPE	SR	‘00xxh’=在物理规范版本 1.01~2.00 的 SD 存储器卡（‘x’表示任意值）。 已定义的卡有： '0000'=通用 SD 读写卡 '0001'=SDROM 卡	这个域的低 8 位可以在未来定义 SD 存储卡的不同变种（每个位可以用于定义不同的 SD 类型）。高 8 位可以用于定义那些不遵守当前的 SD 物理层规范的 SD 卡。	A
479：448	SIZE_OF_PROTECTED_AREA	SR	受保护的区域大小（见以下说明）	（见以下说明）	A
447：440	SPEED_CLASS	SR	卡的速度类型 （见以下说明）	（见以下说明）	A

439：432	PERFORMANCE_MOVE	SR	以 1MB/秒为单位的传输性能（见以下说明）	（见以下说明）	A
431：428	AU_SIZE	SR	AU 的大小（见以下说明）	（见以下说明）	A
427：424	保留				
423：408	ERASE_SIZE	SR	一次可以擦除的 AU 数目	（见以下说明）	A
407：402	ERASE_TIMOUT	SR	擦除 UNIT_OF_ERASE_AU 指定的范围的超时数值	（见以下说明）	A
401：400	ERASE_OFFSET	SR	在擦除时增加的固定偏移数值	（见以下说明）	A
399：312	保留				
311：0	保留给生产厂商				

SIZE_OF_PROTECTED_AREA

标准容量卡和高容量卡设置该位的方式不同。对于标准容量卡，受保护区域的容量由下式计算：

受保护区域 = SIZE_OF_PROTECTED_AREA*MULT*BLOCK_LEN

SIZE_OF_PROTECTED_AREA 的单位是 MULT*BLOCK_LEN。

对于高容量卡，受保护区域的容量由下式计算：

受保护区域=SIZE_OF_PROTECTED_AREA

SIZE_OF_PROTECTED_AREA 的单位是字节。

SPEED_CLASS

这 8 位指示速度的类型和可以通过计算 $P_W/2$ 的数值（PW 是写的性能）。

表 20-11 速度类型代码

SPEED_CLASS	数值定义
00h	类型 0
01h	类型 2
02h	类型 4
03h	类型 6
04h~FFh	保留

PERFORMANCE_MOVE

这 8 位以 1MB/秒为单位指示移动性能（Pm）。如果卡不用 RU（纪录单位）移动数据，应该认为 Pm 是无穷大。设置这个域为 FFh 表示无穷大。

表 20-12 移动性能代码

PERFORMANCE_MOVE	数值定义
00h	未定义
01h	1MB/秒
02h	2MB/秒
……	……
FEh	254MB/秒

FFh	无穷大

AU_SIZE

这 4 位指示 AU 的长度，数值是 16K 字节为单位 2 的幂次的倍数。

表 20-13 AU_SIZE代码

AU_SIZE	数值定义
00h	未定义
01h	16KB
02h	32KB
03h	64KB
04h	128KB
05h	256KB
06h	512KB
07h	1MB
08h	2MB
09h	4MB
Ah~Fh	保留

依据卡的容量，最大的 AU 长度由下表定义。卡可以在 RU 长度和最大的 AU 长度之间设置任意的 AU 长度。

表 20-14 最大的AU长度

容量	16MB~64MB	128MB~256MB	512MB	1GB~32GB
最大的 AU 长度	512KB	1MB	2MB	4MB

ERASE_SIZE

这个 16 位域给出了 N_{ERASE}，当 N_{ERASE} 个 AU 被擦除时，ERASE_TIMOUT 定义了超时时间。主机应该确定适当的一次操作中擦除的 AU 数目，这样主机可以显示擦除操作的进度。如果该域为 0，则不支持擦除的超时计算。

表 20-15 ERASE_SIZE代码

ERASE_SIZE	数值定义
0000h	不支持擦除的超时计算
0001h	1 个 AU
0002h	2 个 AU
0003h	3 个 AU
……	……
FFFFh	65535 个 AU

ERASE_TIMOUT

这 6 位给出了 T_{ERASE}，当 ERASE_SIZE 指示的多个 AU 被擦除时，这个数值给出了从偏移量算起的擦除超时。ERASE_TIMOUT 的范围可以定义到最多 63 秒，卡的生产商可以根据具体实现选择合适的 ERASE_SIZE 与 ERASE_TIMOUT 的组合，先确定 ERASE_TIMOUT 再确定 ERASE_SIZE。

表 20-16 擦除超时代码

ERASE_TIMOUT	数值定义
00	不支持擦除的超时计算
01	1 秒

02	2 秒
03	3 秒
……	……
63	63 秒

ERASE_OFFSET

这 2 位给出了 T_{OFFSET}，当 ERASE_SIZE 和 ERASE_TIMOUT 同为 0 时这个数值没有意义。

表 20-17 擦除偏移代码

ERASE_OFFSET	数值定义
0	0 秒
1	1 秒
2	2 秒
3	3 秒

20.3.2.13 SD的I/O模式

SD 的 I/O 中断

为了让 SD I/O 卡能够中断多媒体卡/SD 模块，在 SD 接口上有一个具有中断功能的引脚——第 8 脚，在 4 位 SD 模式下这个脚是 SDIO_D1，卡用它向多媒体卡/SD 模块提出中断申请。对于每一个卡或卡内的功能，中断功能是可选的。SD I/O 的中断是电平有效，即在被识别并得到多媒体卡/SD 模块的响应之前，中断信号线必须保持有效电平（低），在中断过程结束后保持无效电平（高）。在多媒体卡/SD 模块服务了中断请求后，通过一个 I/O 写操作，写入适当的位到 SD I/O 卡的内部寄存器，即可清除中断状态位。所有 SD I/O 卡的中断输出是低电平有效，多媒体卡/SD 模块在所有数据线（SDIO/D[3：0]）上提供上拉电阻。多媒体卡/SD 模块在中断阶段对第 8 脚（SDIO_D/IRQ）采样并进行中断检测，其它时间该信号线上的数值将被忽略。

存储器操作和 I/O 操作都具有中断阶段，单个数据块操作的中断阶段定义与多个数据块传输操作的中断阶段定义不同。

SD 的 I/O 暂停和恢复

在一个多功能的 SD I/O 卡或同时具有 I/O 和存储器功能的卡中，多个设备（I/O 和存储器）共用 MMC/SD 总线。为了使 MMC/SD 模块中的多个设备能够共用总线，SD I/O 卡和复合卡可以有选择地实现暂停/恢复的概念；如果一个卡支持暂停/恢复，MMC/SD 模块能够暂时地停止一个功能或存储器的数据传输操作（暂停），借此让出总线给具有更高优先级的其它功能或存储器，在这个具有更高优先级的传输完成后，再恢复原先暂停的传输。支持暂停/恢复的操作是可选的。在 MMC/SD 总线上执行暂停/恢复操作有下述步骤：

1. 确定SDIO_D[3：0]信号线的当前功能
2. 请求低优先级或慢的操作暂停
3. 等待暂停操作完成，确认设备已暂停
4. 开始高优先级的传输
5. 等待高优先级的传输结束
6. 恢复暂停的操作

SDI/O 读等待（ReadWait）

可选的读等待（RW）操作只适用于 SD 卡的 1 位或 4 位模式。读等待操作允许 MMC/SD 模块在一个卡正在读多个寄存器（IO_RW_EXTENDED, CMD53）时，要求它暂时停止数据传输，同时允许 MMC/SD 模块发送命令到 SD I/O 设备中的其他功能。判断一个卡是否支持读等待协议，MMC/SD 模块应该检测卡的内部寄存器。读等待的时间与中断阶段相关。

20.3.2.14 命令与响应

应用相关命令和通用命令

SD 卡主机模块系统是用于提供一个适用于多种应用类型的标准接口，但同时又要兼顾特定用户和应用的功能，因此标准中定义了两类通用命令：应用相关命令（ACMD）和通用命令（GEN_CMD）。

当卡收到 APP_CMD（CMD55）命令时，卡期待下一个命令是应用相关命令。应用相关命令（ACMD）具有普通多媒体卡相同的格式结构，并可以使用相同的 CMD 号码，因为它是出现在 APP_CMD（CMD55）后面，所以卡把它识别为 ACMD 命令。如果跟随 APP_CMD（CMD55）之后不是一个已经定义的应用相关命令，则认为它是一个标准命令；例如：有一个 SD_STATUS（ACMD13）应用相关命令，如果在紧随 APP_CMD（CMD55）之后收到 CMD13，它将被解释为 SD_STATUS（ACMD13）；但是如果卡在紧随 APP_CMD（CMD55）之后收到 CMD7，而这个卡没有定义 ACMD7，则它将被解释为一个标准的 CMD7（SELECT/DESELECT_CARD）命令。

如果要使用生产厂商自定义的 ACMD，SD 卡主机需要做以下操作：

1. 发送APP_CMD（CMD55）命令

   卡送回响应给多媒体/SD 卡模块，指示设置了 APP_CMD 位并等待 ACMD 命令。

2. 发送指定的ACMD

   卡送回响应给多媒体/SD 卡模块，指示设置了 APP_CMD 位，收到的命令已经正确地按照 ACMD 命令解析；如果发送了一个非 ACMD 命令，卡将按照普通的多媒体卡命令处理同时清除卡中状态寄存器的 APP_CMD 位。

如果发送了一个非法的命令（不管是 ACMD 还是 CMD），将被按照标准的非法多媒体卡命令进行错误处理。

GEN_CMD 命令的总线操作过程，与单数据块读写命令（WRITE_BLOCK，CMD24 或 READ_SINGLE_BLOCK，CMD17）相同；这时命令的参数表示数据传输的方向而不是地址，数据块具有用户自定义的格式和意义。

发送 GEN_CMD（CMD56）命令之前，卡必须被选中（状态机处于传输状态），数据块的长度由 SET_BLOCKLEN（CMD16）定义。GEN_CMD（CMD56）命令的响应是 R1b 格式。

命令类型

应用相关命令和通用命令有四种不同的类型：

1. 广播命令（BC）：发送到所有卡，没有响应返回。
2. 带响应的广播命令（BCR）：发送到所有卡，同时收到从所有卡返回的响应。
3. 带寻址（点对点）的命令（AC）：发送到选中的卡，在SDIO_D信号线上不包括数据传输。
4. 带寻址（点对点）的数据传输命令（AC）：发送到选中的卡，在SDIO_D信号线上包含数据传输。

命令格式

命令格式参见表 20-2。

多媒体卡/SD 卡模块的命令

表 20-18 基于块传输的写命令

CMD 索引	类型	参数	响应格式	缩写	说明
CMD23	ac	[31：16]=0 [15：0]=数据块数目	R1	SET_BLOCK_COUNT	定义在随后的多块读或写命令中需要传输块的数目。
CMD24	adtc	[31：0]=数据地址	R1	WRITE_BLOCK	按照 SET_BLOCK_LEN 命令选择的长度写一个块。
CMD25	adtc	[31：0]=数据地址	R1	WRITE_MULTIPLE_BLOCK	收到一个 STOP_TRANSMISSION 命令或达到了指定的块数目之前，连续地写数据块。

CMD26	adtc	[31：0]=填充位	R1	PROGRAM_CID	对卡的识别寄存器编程。对于每个卡只能发送一次这个命令。卡中有硬件机制防止多次的编程操作。通常该命令保留给生产厂商。
CMD27	adtc	[31：0]=填充位	R1	PROGRAM_CSD	对卡的 CSD 中可编程的位编程。

表 20-19 基于块传输的写保护命令

CMD 索引	类型	参数	响应格式	缩写	说明
CMD28	ac	[31：0]=数据地址	R1b	SET_WRITE_PROT	如果卡具有写保护的功能，该命令设置指定组的写保护位。写保护的属性设置在卡的特定数据域（WP_GRP_SIZE）。
CMD29	ac	[31：0]=数据地址	R1b	CLR_WRITE_PROT	如果卡具有写保护的功能，该命令清除指定组的写保护位。
CMD30	adtc	[31：0]=写保护数据地址	R1	SEND_WRITE_PROT	如果卡具有写保护的功能，该命令要求卡发送写保护位的状态。
CMD31	保留				

表 20-20 擦除命令

CMD 索引	类型	参数	响应格式	缩写	说明
CMD32 … CMD34	保留。为了与旧版本的对媒体卡协议向后兼容，不能使用这些命令代码。				
CMD35	ac	[31：0]=数据地址	R1	ERASE_GROUP_START	在选择的擦除范围内，设置第一个擦除组的地址。
CMD36	ac	[31：0]=数据地址	R1	ERASE_GROUP_END	在选择的连续擦除范围内，设置最后一个擦除组的地址。
CMD37	保留。为了与旧版本的对媒体卡协议向后兼容，不能使用这个命令代码。				
CMD38	ac	[31：0]=填充位	R1b	ERASE	擦除之前选择的数据块。

表 20-21 I/O模式命令

CMD 索引	类型	参数	响应格式	缩写	说明
CMD39	ac	[31：16]=RCA [15]=寄存器写标志 [14：8]=寄存器地址 [7：0]=寄存器数据	R4	FAST_IO	用于写和读 8 位（寄存器）数据域。该命令指定一个卡和寄存器，如果设置了写标志还提供写入的数据。R4 响应包含从指定寄存器读出的数据。该命令访问未在多媒体卡标准中定义的与应用相关的寄存器。
CMD40	bcr	[31：0]=填充位	R5	GO_IRQ_STATE	置系统于中断模式。
CMD41	保留。				

表 20-22 上锁命令

CMD 索引	类型	参数	响应格式	缩写	说明

CMD42	adtc	[31：0]=填充位	R1	LOCK_ UNLOCK	设置/清除密码或对卡上锁/解锁。数据块的长度由 SET_BLOCK_LEN 命令设置。
CMD43 … CMD54	保留。				

表 20-23 应用相关命令

CMD 索引	类型	参数	响应格式	缩写	说明
CMD55	ac	[31：16]=RCA [15：0]=填充位	R1	APP_CMD	指示卡下一个命令是应用相关命令而不是一个标准命令。
CMD56	adtc	[31：1]=填充位 [0]=RD/WR	R1	GEN_CMD	在通用或应用相关命令中，或者用于向卡中传输一个数据块，或者用于从卡中读取一个数据块。数据块的长度由 SET_BLOCK_LEN 命令设置。
CMD57 … CMD59	保留。				
CMD60 … CMD63	保留给生产厂商。				

20.3.3 响应格式

所有的响应是通过 MCCMD 命令在 SDIO_CMD 信号线上传输。响应的传输总是从对应响应字的位串的最左面开始，响应字的长度与响应的类型相关。

一个响应总是有一个起始位（始终为 0），跟随着传输的方向位（卡=0）。下表中标示为 x 的数值表示一个可变的部分。除了 R3 响应类型，所有的响应都有 CRC 保护。每一个命令码字都有一个结束位（始终为 1）。

共有 5 种响应类型，它们的格式定义如下：

20.3.3.1 R1（普通响应命令）

代码长度=48 位。位 45：40 指示要响应的命令索引，它的数值介于 0 至 63 之间。卡的状态由 32 位进行编码。

表 20-24 R1响应

位	域宽度	数值	说明
47	1	0	开始位
46	1	0	传输位
[45：40]	6	X	命令索引
[39：8]	32	X	卡状态
[7：1]	7	X	CRC7
0	1	1	结束位

20.3.3.2 R1b

与 R1 格式相同，但可以选择在数据线上发送一个繁忙信号。收到这些命令后，依据收到命令之前的状态，卡可能变为繁忙。

20.3.3.3 R2（CID、CSD寄存器）

代码长度=136 位。CID 寄存器的内容将作为 CMD2 和 CMD10 的响应发出。CSD 寄存器的内容将作为 CMD9 的响应发出。卡只送出 CID 和 CSD 的位[127…1]，在接收端这些寄存器的位 0 被响应的结束位所

取代。卡通过拉低 MCDAT 指示它正在进行擦除操作；实际擦除操作的时间可能非常长，主机可以发送 CMD7 命令不选中这个卡。

表 20-25 R2响应

位	域宽度	数值	说明
135	1	0	开始位
134	1	0	传输位
[133：128]	6	‘111111’	命令索引
[127：1]	127	X	卡状态
0	1	1	结束位

20.3.3.4 R3（OCR寄存器）

代码长度=48 位。OCR 寄存器的内容将作为 CMD1 的响应发出。电平代码的定义是：限制的电压窗口=低，卡繁忙=低。

表 20-26 R3响应

位	域宽度	数值	说明
47	1	0	开始位
46	1	0	传输位
[45：40]	6	‘111111’	保留
[39：8]	32	X	OCR 寄存器
[7：1]	7	‘1111111’	保留
0	1	1	结束位

20.3.3.5 R4（快速I/O）

代码长度=48 位。参数域包含指定卡的 RCA、需要读出或写入寄存器的地址、和它的内容。

表 20-27 R4响应

位		域宽度	数值	说明
47		1	0	开始位
46		1	0	传输位
[45：40]		6	‘100111’	CMD39
[39：8] 参数域	[31：16]	16	X	RCA
	[15：8]	8	X	寄存器地址
	[7：0]	8	X	读寄存器的内容
[7：1]		7	‘X’	CRC7
0		1	1	结束位

20.3.3.6 R4b

仅适合 SD I/O 卡：一个 SDIO 卡收到 CMD5 后将返回一个唯一的 SDIO 响应 R4。

表 20-28 R4b响应

位	域宽度	数值	说明
47	1	0	开始位
46	1	0	传输位
[45：40]	6	X	保留

[39：8] 参数域	39	1	X	卡已就绪
	[38：36]	3	X	I/O 功能数目
	35	1	X	当前存储器
	[34：32]	3	X	填充位
	[31：8]	24	X	I/O OCR
[7：1]		7	X	保留
0		1	1	结束位

当一个 SD I/O 卡收到命令 CMD5，卡的 I/O 部分被使能并能够正常地响应所有后续的命令。I/O 卡的使能状态将保持到下一次复位、断电或收到 I/O 复位的 CMD52 命令。注意，一个只包含存储器功能的 SD 卡可以响应 CMD5 命令，它的正确响应可以是：当前存储器=1，I/O 功能数目=0。按照 SD 存储器卡规范版本 1.0 设计的只包含存储器功能的 SD 卡，可以检测到 CMD5 命令为一个非法命令并不响应它。可以处理 I/O 卡的主机将发送 CMD5 命令，如果卡返回响应 R4，则主机会依据 R4 响应中的数据确定卡的配置。

20.3.3.7 R5（中断请求）

仅适用于多媒体卡。代码长度=48 位。如果这个响应由主机产生，则参数中的 RCA 域为 0x0。

表 20-29 R5响应

位		域宽度	数值	说明
47		1	0	开始位
46		1	0	传输位
[45：40]		6	‘101000’	CMD40
[39：8] 参数域	[31：16]	16	X	成功的卡或主机的 RCA[31：16]
	[15：0]	16	X	未定义。 可以作为中断数据。
[7：1]		7	X	CRC7
0		1	1	结束位

20.3.3.8 R6（中断请求）

仅适用于 SD I/O 卡。这是一个存储器设备对 CMD3 命令的正常响应。

表 20-30 R6响应

位		域宽度	数值	说明
47		1	0	开始位
46		1	0	传输位
[45：40]		6	‘000011’	CMD3
[39：8] 参数域	[31：16]	16	X	成功的卡或主机的 RCA[31：16]
	[15：0]	16	X	卡状态
[7：1]		7	X	CRC7
0		1	1	结束位

当发送 CMD3 命令到只有 I/O 功能的卡时，卡的状态位[23：8]会改变；此时，响应中的 16 位将是只有 I/O 功能的 SD 卡中的数值：

- 位 15=COM_CRC_ERROR
- 位 14=ILLEGAL_COMMAND
- 位 13=ERROR
- 位[12：0]=保留

20.3.4 SDIO I/O卡特定的操作

下述功能是 SD I/O 卡特定的操作：

- 由 SDIO_D2 信号线实现的 SDIO 读等待操作。
- 通过停止时钟实现的 SDIO 读等待操作。
- SDIO 暂停/恢复操作（写和读暂停）
- SDIO 中断

只有设置了 SDIO_DTCTRL[11]位时，SDIO 才支持这些操作；但读暂停除外，因为它不需要特殊的硬件操作。

20.3.4.1 使用SDIO_D2信号线的SDIO I/O读等待操作

在收到第一个数据块之前即可以开始读等待过程，使能数据通道（设置 SDIO_DTCTRL[0]位）、使能 SDIO 特定操作（设置 SDIO_DTCTRL[11]位）、开始读等待（SDIO_DTCTRL[10]=0 并且 SDIO_DTCTRL[8]=1），同时数据传输方向是从卡至 SDIO 主机（SDIO_DTCTRL[1]=1），DPSM 将直接从空闲进入读等待状态。在读等待状态时，2 个 SDIO_CK 时钟周期后，DPSM 驱动 SDIO_D2 为'0'，在此状态，如果设置 RDWTSTOP 位（SDIO_DTCTRL[9]），则 DPSM 会在等待状态多停留 2 个 SDIO_CK 时钟周期，（根据 SDIO 规范）并在一个时钟周期中驱动 SDIO_D2 为'1'。然后 DPSM 开始等待从卡里接收数据。在接收数据块时，即使设置了开始读等待，DPSM 也不会进入读等待，读等待过程将在收到 CRC 后开始。必须清除 RDWTSTOP 才能开始新的读等待操作。在读等待期间，SDIO 主机可以在 SDIO_D1 上监测 SDIO 中断。

20.3.4.2 使用停止SDIO_CK的SDIO读等待操作

如果 SDIO 卡不能支持前述的读等待操作，SDIO 可以停止 SDIO_CK 进入读等待（按照 20.3.4.1 节介绍的方式设置 SDIO_DTCTRL，但置 SDIO_DTCTRL[10]=1），在接收当前数据块结束位之后的 2 个 SDIO_CK 周期后，DSPM 停止时钟，在设置了读等待开始位后恢复时钟。

因为 SDIO_CK 停止了，不可以向卡发送任何命令。在读等待期间，SDIO 主机可以在 SDIO_D1 上监测 SDIO 中断。

20.3.4.3 SDIO暂停/恢复操作

在向卡发送数据时，SDIO 可以暂停写操作。设置 SDIO_CMD[11]位，这指示 CPSM 当前的命令是一个暂停命令。CPSM 分析响应，在从卡收到 ACK 时（暂停被接受），它确认在收到当前数据块的 CRC 后进入空闲状态。

硬件不会保存结束暂停操作之后，剩余的发送数据块数目。

可以通过软件暂停写操作：在收到卡对暂停命令的 ACK 时，停止 DPSM（SDIO_DTCTRL[0]=0），DPSM 即可进入空闲状态。

暂停读操作：DPSM 在 Wait_R 状态等待，在停止数据传输进入暂停之前，已经发送完成完整的数据包。随后应用程序继续读出 RxBUF 直到 BUF 变空，最后 DPSM 自动地进入空闲状态。

20.3.4.4 SDIO中断

当设置了 SDIO_DTCTRL[11]位，SDIO 主机在 SDIO_D1 信号线上监测 SDIO 中断。

20.3.5 硬件流控制

使用硬件流控制功能可以避免 BUF 下溢（发送模式）和上溢（接收模式）错误。

操作过程是停止 SDIO_CK 并冻结 SDIO 状态机，在 BUF 不能进行发送和接收数据时，数据传输暂停。只有由 SDIOCLK 驱动的状态机被冻结，AHB 接口还在工作。即使在流控制起作用时，仍然可以读出或

写入 BUF。

必须设置 SDIO_CLKCTRL[14]位为'1'，才能使能硬件流控制。复位后，硬件流控制功能关闭。

20.4 SDIO寄存器

设备可以通过在 AHB 上操作的 32 位控制寄存器与系统通信。

必须以字（32 位）的方式操作这些外设寄存器。

下表是 SDIO 寄存器的总结。

表 20-31 SDIO寄存器映像

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	SDIO_POWER	保留																														PWRCTRL	
	0x00000000																															0	0
0x04	SDIO_CLKCTRL	保留															CLKPSC[9：8]		FLWCTRLEN	CLKEDG	BUSWIDTH	BYPS	PWRSVG	CLKEN	CLKPSC[7：0]								
	0x00000000																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08	SDIO_ARG	ARG[31:0]																															
	0x00000000	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	SDIO_CMD	保留																				SDIOSUSP	CMDMEN	PNDWT	INTWT	RSPWT[1:0]		CMDIDX[5:0]					
	0x00000000																					0	0	0	0	0	0	0	0	0	0	0	0
0x10	SDIO_RSPCMD	保留																										RSPCMD[5:0]					
	0x00000000																											0	0	0	0	0	0
0x14	SDIO_RSP1	CARDSTS1[31:0]																															
	0x00000000	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	SDIO_RSP2	CARDSTS2[31:0]																															
	0x00000000	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1C	SDIO_RSP3	CARDSTS3[31:0]																															
	0x00000000	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20	SDIO_RSP4	CARDSTS4[31:0]																															
	0x00000000	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x24	SDIO_DTTMR	TIMEOUT[31:0]																															
	0x00000000	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x28	SDIO_DTLEN	保留							DTLEN[24:0]																								
	0x00000000								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

| | | |

20.4.2 SDIO时钟控制寄存器（SDIO_CLKCTRL）

地址偏移：0x04

复位值：0x00000000

SDIO_CLKCTRL 寄存器控制 SDIO_CK 输出时钟。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															CLKPSC[9]
res															rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CLKPSC[8]	FLWCTRLEN	CLKEDG	BUSWIDTH		BYPS	PWRSVG	CLKEN	CLKPSC[7：0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31：17	保留，始终读为 0。
位 16：15	**CLKPSC[9：8]：**时钟分频系数（Clock divide factor）高 2 位 这个域定义了输入时钟（SDIOCLK）与输出时钟（SDIO_CK）间的分频系数： SDIO_CK 频率=SDIOCLK/[CLKPSC[9：0]+2]
位 14	**FLWCTRLEN：**硬件流控制使能（HW Flow Control enable） 0：关闭硬件流控制 1：使能硬件流控制 当使能硬件流控制后，关于 TXBUF_E 和 RXBUF_F 中断信号的意义请参考 20.4.11 节的 SDIO 状态寄存器的定义。
位 13	**CLKEDG：**SDIO_CK 相位选择位（SDIO_CK dephasing selection bit） 0：在主时钟 SDIOCLK 上升沿产生 SDIO_CK。 1：在主时钟 SDIOCLK 下降沿产生 SDIO_CK。
位 12：11	**BUSWIDTH：**宽总线模式使能位（Wide bus mode enable bit） 00：默认总线模式，使用 SDIO_D0。 01：4 位总线模式，使用 SDIO_D[3：0]。 10：8 位总线模式，使用 SDIO_D[7：0]。
位 10	**BYPS：**旁路时钟分频器（Clock divider bypass enable bit） 0：关闭旁路：驱动 SDIO_CK 输出信号之前，依据 CLKPSC 数值对 SDIOCLK 分频。 1：使能旁路：SDIOCLK 直接驱动 SDIO_CK 输出信号。
位 9	**PWRSVG：**省电配置位（Power saving configuration bit） 为了省电，当总线为空闲时，设置 PWRSVG 位可以关闭 SDIO_CK 时钟输出。 0：始终输出 SDIO_CK。 1：仅在有总线活动时才输出 SDIO_CK。
位 8	**CLKEN：**时钟使能位（Clock enable bit） 0：SDIO_CK 关闭。 1：SDIO_CK 使能。
位 7：0	**CLKPSC[7：0]：**时钟分频系数（Clock divide factor）低 8 位 这个域定义了输入时钟（SDIOCLK）与输出时钟（SDIO_CK）间的分频系数： SDIO_CK 频率=SDIOCLK/[CLKPSC[9：0]+2]。

注意：　1.当 SD/SDIO 卡或多媒体卡在识别模式，SDIO_CK 的频率必须低于 400kHz。

2.当所有卡都被赋予了相应的地址后，时钟频率可以改变到卡总线允许的最大频率。

3.写数据后的 7 个 HCLK 时钟周期内不能写入这个寄存器。对于 SD I/O 卡，在读等待期间可以停止 SDIO_CK，此时 SDIO_CLKCTRL 寄存器不控制 SDIO_CK。

20.4.3 SDIO参数寄存器（SDIO_ARG）

地址偏移：0x08

复位值：0x00000000

SDIO_ARG 寄存器包含 32 位命令参数，它将作为命令的一部分发送到卡中。

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

位 31：0	**ARG**：命令参数（Command argument） 命令参数是发送到卡中命令的一部分，如果一个命令包含一个参数，必须在写命令到命令寄存器之前加载这个寄存器。

20.4.4 SDIO命令寄存器（SDIO_CMD）

地址偏移：0x0C

复位值：0x00000000

SDIO_CMD 寄存器包含命令索引和命令类型位。命令索引是作为命令的一部分发送到卡中。命令类型位控制命令通道状态机（CPSM）。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															
res															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留				SDIO SUSP	CMD MEN	PND WT	INTW T	RSPWT		CMDIDX					
res				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31：12	保留，始终读为 0
位 11	**SDIOSUSP**：SD I/O 暂停命令（SD I/O suspend command） 如果设置该位，则将要发送的命令是一个暂停命令（只能用于 SDIO 卡）。
位 10	**CMDMEN**：命令通道状态机（CPSM）使能位（Command path state machine （CPSM） Enable bit） 如果设置该位，则使能 CPSM。
位 9	**PNDWT**：CPSM 等待数据传输结束（CmdPend 内部信号） （CPSM Waits for ends of data transfer （CmdPend internal signal）） 如果设置该位，则 CPSM 在开始发送一个命令之前等待数据传输结束。
位 8	**INTWT**：CPSM 等待中断请求（CPSM waits for interrupt request） 如果设置该位，则 CPSM 关闭命令超时控制并等待中断请求。
位 7：6	**RSPWT**：等待响应位（Wait for response bits） 这 2 位指示 CPSM 是否需要等待响应，如果需要等待响应，则指示响应类型。 00：无响应，期待 CMDCMPL 标志 01：短响应，期待 CMDRSPCMPL 或 CMDFAIL 标志 10：无响应，期待 CMDCMPL 标志 11：长响应，期待 CMDRSPCMPL 或 CMDFAIL 标志
位 5：0	**CMDIDX**：命令索引（Command index） 命令索引是作为命令的一部分发送到卡中。

注意： 1.写数据后的 7 个 HCLK 时钟周期内不能写入这个寄存器。

2.多媒体卡可以发送 2 种响应：48 位长的短响应，或 136 位长的长响应。SD 卡和 SD I/O 卡只能发送短响应，参数可以根据响应的类型而变化，软件将根据发送的命令区分响应的类型。

20.4.5 SDIO命令响应寄存器（SDIO_RSPCMD）

地址偏移：0x10

复位值：0x00000000

SDIO_RSPCMD 寄存器包含最后收到的命令响应中的命令索引。如果传输的命令响应不包含命令索引（长响应或 OCR 响应），尽管它应该包含 111111b（响应中的保留域值），但 RSPCMD 域的内容未知。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															
res															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留										RSPCMD					
res										r	r	r	r	r	r

位 31：6	保留，始终读为 0
位 5：0	**RSPCMD：**响应的命令索引（Response command index） 只读位，包含最后收到的命令响应中的命令索引。

20.4.6 SDIO响应1..4寄存器（SDIO_RSPx）

地址偏移：0x14+4*（x-1），其中 x=1..4

复位值：0x00000000

SDIO_RSP1/2/3/4 寄存器包含卡的状态，即收到响应的部分信息。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CARDSTSx															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CARDSTSx															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

位 31：0	**CARDSTSx：** 见下表。

根据响应状态，卡的状态长度是 32 位或 127 位。

表 20-32 响应类型和SDIO_RSPx寄存器

寄存器	短响应	长响应
SDIO_RSP1	卡状态[31：0]	卡状态[127：96]
SDIO_RSP2	不用	卡状态[95：64]
SDIO_RSP3	不用	卡状态[63：32]
SDIO_RSP4	不用	卡状态[31：1]

总是先收到卡状态的最高位，SDIO_RSP4 寄存器的最低位始终为 0。

20.4.7 SDIO数据定时器寄存器（SDIO_DTTMR）

地址偏移：0x24

复位值：0x00000000

SDIO_DTTMR 寄存器包含以卡总线时钟周期为单位的数据超时时间。

一个计数器从 SDIO_DTTMR 寄存器加载数值，并在数据通道状态机（DPSM）进入 Wait_R 或繁忙状态时进行递减计数，当 DPSM 处在这些状态时，如果计数器减为 0，则设置超时标志。

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

位 31：0	**TIMEOUT：** 数据超时时间（Data timeout period） 以卡总线时钟周期为单位的数据超时时间。

注意： *在写入数据控制寄存器进行数据传输之前，必须先写入数据定时器寄存器和数据长度寄存器。*

20.4.8 SDIO数据长度寄存器（SDIO_DTLEN）

地址偏移：0x28

复位值：0x00000000

SDIO_DTLEN 寄存器包含需要传输的数据字节长度。当数据传输开始时，这个数值被加载到数据计数器中。

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

位 31：25	保留，始终读为 0。
位 24：0	**DTLEN：** 数据长度（Data length value） 要传输的数据字节数目。

注意： *对于块数据传输，数据长度寄存器中的数值必须是数据块长度（见 20.4.9SDIO 数据控制寄存器（SDIO_DTCTRL））的倍数。在写入数据控制寄存器进行数据传输之前，必须先写入数据定时器寄存器和数据长度寄存器。*

20.4.9 SDIO数据控制寄存器（SDIO_DTCTRL）

地址偏移：0x2C

复位值：0x00000000

SDIO_DTCTRL 寄存器控制数据通道状态机（DPSM）。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															
res															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留				SDIOEN	RDWTMODE	RDWTSTOP	RDWTSTART	BLKSIZE				DMAEN	TFRMODE	TFRDIR	TFREN
res				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31：12	保留，始终读为 0。
位 11	**SDIOEN：**SD I/O 使能功能（SD I/O enable functions） 如果设置了该位，则 DPSM 执行 SD I/O 卡特定的操作。
位 10	**RDWTMODE：**读等待模式（Read wait mode） 0：使用 SDIO_D2 控制读等待； 1：使用 SDIO_CK 控制读等待。
位 9	**RDWTSTOP：**读等待停止（Read wait stop） 0：如果设置了 RDWTSTART，执行读等待； 1：如果设置了 RDWTSTART，停止读等待。
位 8	**RDWTSTART：**读等待开始（Read wait start） 设置该位开始读等待操作。
位 7：4	**BLKSIZE：**数据块长度（Data block size） 当选择了块数据传输模式，该域定义数据块长度： 0000：块长度=2^0=1 字节； 0001：块长度=2^1=2 字节； 0010：块长度=2^2=4 字节； 0011：块长度=2^3=8 字节； 0100：（十进制 4）块长度=2^4=16 字节； 0101：（十进制 5）块长度=2^5=32 字节； 0110：（十进制 6）块长度=2^6=64 字节； 0111：块长度=2^7=128 字节； 1000：块长度=2^8=256 字节； 1001：块长度=2^9=512 字节； 1010：块长度=2^{10}=1024 字节； 1011：块长度=2^{11}=2048 字节； 1100：块长度=2^{12}=4096 字节； 1101：块长度=2^{13}=8192 字节； 1110：块长度=2^{14}=16384 字节； 1111：保留。
位 3	**DMAEN：**DMA 使能位（DMA enable bit） 0：关闭 DMAEN； 1：使能 DMAEN。
位 2	**TFRMODE：**数据传输模式（Data transfer mode selection） 0：块数据传输； 1：流数据传输。
位 1	**TFRDIR：**数据传输方向（Data transfer direction selection） 0：控制器至卡； 1：卡至控制器。

位 0	**TFREN**：数据传输使能位（Data transfer enabled bit） 如果设置该位为 1，则开始数据传输。根据 TFRDIR 方向位，DPSM 进入 Wait_S 或 Wait_R 状态，如果在传输的一开始就设置了 RDWTSTART 位，则 DPSM 进入读等待状态。不需要在数据传输结束后清除使能位，但必须更改 SDIO_DTCTRL 以允许新的数据传输。

注意： 写数据后的 7 个 HCLK 时钟周期内不能写入这个寄存器。

20.4.10 SDIO数据计数器寄存器（SDIO_DTCNTR）

地址偏移：0x30

复位值：0x00000000

当 DPSM 从空闲状态进入 Wait_R 或 Wait_S 状态时，SDIO_DTCNTR 寄存器从数据长度寄存器加载数值（见 20.4.8 节 SDIO 数据长度寄存器（SDIO_DTLEN）），在数据传输过程中，该计数器的数值递减直到减为 0，然后 DPSM 进入空闲状态并设置数据状态结束标志 DTCMPL。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留							CNT								
res							r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

位 31：25	保留，始终读为 0。
位 24：0	**CNT**：数据计数数值（Data count value） 读这个寄存器时返回待传输的数据字节数，写这个寄存器无作用。

注意： 只能在数据传输结束时读这个寄存器。

20.4.11 SDIO状态寄存器（SDIO_STS）

地址偏移：0x34

复位值：0x00000000

SDIO_STS 是一个只读寄存器，它包含两类标志：

- 静态标志（位[23：22、10：0]）：写入 SDIO 中断清除寄存器（见 20.4.12 节 SDIO 清除中断寄存器（SDIO_INTCLR）），可以清除这些位。
- 动态标志（位[21：11]）：这些位的状态变化根据它们对应的那部分逻辑而变化（例如：BUF 满和空标志变高或变低随 BUF 的数据写入变化）。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留									SDIOIF	RXBUF	TXBUF	RXBUF_E	TXBUF_E	RXBUF_F	TXBUF_F
res									r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXBUF_H	TXBUF_H	DORX	DOTX	DOCMD	DTBLKCMPL	SBITERR	DTCMPL	CMDCMPL	CMDRSPCMPL	RXERRO	TXERRU	DTTIMEOUT	CMDTIMEOUT	DTFAIL	CMDFAIL
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

位 31：23	保留，始终读为 0。
位 22	**SDIOIF**：收到 SDIO 中断（SDIO interrupt received）
位 21	**RXBUF**：在接收 BUF 中的数据可用（Data available in receive BUF）

位 20	**TXBUF：**在发送 BUF 中的数据可用（Data available in transmit BUF）
位 19	**RXBUF_E：**接收 BUF 空（Receive BUF empty）
位 18	**TXBUF_E：**发送 BUF 空（Transmit BUF empty） 若使用了硬件流控制，当 BUF 包含 2 个字时，TXBUF_E 信号变为有效。
位 17	**RXBUF_F：**接收 BUF 满（Receive BUF full） 若使用了硬件流控制，当 BUF 还差 2 个字满时，RXBUF_F 信号变为有效。
位 16	**TXBUF_F：**发送 BUF 满（Transmit BUF full）
位 15	**RXBUF_H：**接收 BUF 半满（Receive BUF half full）：BUF 中至少还有 8 个字。
位 14	**TXBUF_H：**发送 BUF 半空（Transmit BUF half empty）：BUF 中至少还可以写入 8 个字。
位 13	**DORX：**正在接收数据（Data receive in progress）
位 12	**DOTX：**正在发送数据（Data transmit in progress）
位 11	**DOCMD：**正在传输命令（Command transfer in progress）
位 10	**DTBLKCMPL：**已发送/接收数据块（CRC 检测成功） （Data block sent/received （CRC check passed））
位 9	**SBITERR：**在宽总线模式，没有在所有数据信号上检测到起始位（Start bit not detected on all data signals in wide bus mode）
位 8	**DTCMPL：**数据结束（数据计数器，SDIO_DTCNTR=0） （Data end （data counter, SDIO CNT, is zero））
位 7	**CMDCMPL：**命令已发送（不需要响应）（Command sent（no response required））
位 6	**CMDRSPCMPL：**已接收到响应（CRC 检测成功）（Command response）
位 5	**RXERRO：**接收 BUF 上溢错误（Received BUF overrun error）
位 4	**TXERRU：**发送 BUF 下溢错误（Transmit BUF underrun error）
位 3	**DTTIMEOUT：**数据超时（Data timeout）
位 2	**CMDTIMEOUT：**命令响应超时（Command response timeout） 命令超时时间是一个固定的值，为 64 个 SDIO_CK 时钟周期。
位 1	**DTFAIL：**已发送/接收数据块（CRC 检测失败）（Data block sent/received）
位 0	**CMDFAIL：**已收到命令响应（CRC 检测失败）（Command response received）

20.4.12 SDIO清除中断寄存器（SDIO_INTCLR）

地址偏移：0x38

复位值：0x00000000

SDIO_INTCLR 是一个只写寄存器，在对应寄存器位写'1'将清除 SDIO_STS 状态寄存器中的对应位。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留									SDIOIF	保留					
res									rw	res					

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留					DTBLKCMPL	SBITERR	DTCMPL	CMDCMPL	CMDRSPCMPL	RXERRO	TXERRU	DTTIMEOUT	CMDTIMEOUT	DTFAIL	CMDFAIL
res					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31：23	保留，始终读为 0。
位 22	**SDIOIF**：SDIOIF 标志清除位（SDIOIF flag clear bit） 软件设置该位以清除 SDIOIF 标志。
位 21：11	保留，始终读为 0。
位 10	**DTBLKCMPL**：DTBLKCMPL 标志清除位（DTBLKCMPL flag clear bit） 软件设置该位以清除 DTBLKCMPL 标志。
位 9	**SBITERR**：SBITERR 标志清除位（SBITERR flag clear bit） 软件设置该位以清除 SBITERR 标志。
位 8	**DTCMPL**：DTCMPL 标志清除位（DTCMPL flag clear bit） 软件设置该位以清除 DTCMPL 标志。
位 7	**CMDCMPL**：CMDCMPL 标志清除位（CMDCMPL flag clear bit） 软件设置该位以清除 CMDCMPL 标志。
位 6	**CMDRSPCMPL**：CMDRSPCMPL 标志清除位（CMDRSPCMPL flag clear bit） 软件设置该位以清除 CMDRSPCMPL 标志。
位 5	**RXERRO**：RXERRO 标志清除位（RXERRO flag clear bit） 软件设置该位以清除 RXERRO 标志。
位 4	**TXERRU**：TXERRU 标志清除位（TXERRU flag clear bit） 软件设置该位以清除 TXERRU 标志。
位 3	**DTTIMEOUT**：DTTIMEOUT 标志清除位（DTTIMEOUT flag clear bit） 软件设置该位以清除 DTTIMEOUT 标志。
位 2	**CMDTIMEOUT**：CMDTIMEOUT 标志清除位（CMDTIMEOUT flag clear bit） 软件设置该位以清除 CMDTIMEOUT 标志。
位 1	**DTFAIL**：DTFAIL 标志清除位（DTFAIL flag clear bit） 软件设置该位以清除 DTFAIL 标志。
位 0	**CMDFAIL**：CMDFAIL 标志清除位。（CMDFAIL flag clear bit） 软件设置该位以清除 CMDFAIL 标志。

20.4.13 SDIO中断屏蔽寄存器（SDIO_INTEN）

地址偏移：0x3C

复位值：0x00000000

在对应位置'1'，SDIO_INTEN 中断屏蔽寄存器决定哪一个状态位产生中断。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留									SDIOIF	RXBUF	TXBUF	RXBUF_E	TXBUF_E	RXBUF_F	TXBUF_F
res									rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXBUF_H	TXBUF_H	DORX	DOTX	DOCMD	DTBLKCMPL	SBITERR	DTCMPL	CMDCMPL	CMDRSPCMPL	RXERRO	TXERRU	DTTIMEOUT	CMDTIMEOUT	DTFAIL	CMDFAIL
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31：23	保留，始终读为 0。

位 22	**SDIOIF：** 允许 SDIO 模式中断已接收中断（SDIO mode interrupt received interrupt enable） 由软件设置/清除该位，允许/关闭 SDIO 模式中断已接收中断功能。 1：SDIO 模式中断已接收不产生中断 0：SDIO 模式中断已接收产生中断
位 21	**RXBUF：** 接收 BUF 中的数据有效产生中断（Data available in RxBUF interrupt enable） 由软件设置/清除该位，允许/关闭接收 BUF 中的数据有效中断。 0：接收 BUF 中的数据有效不产生中断 1：接收 BUF 中的数据有效产生中断
位 20	**TXBUF：** 发送 BUF 中的数据有效产生中断（Data available in TxBUF interrupt enable） 由软件设置/清除该位，允许/关闭发送 BUF 中的数据有效中断。 0：发送 BUF 中的数据有效不产生中断 1：发送 BUF 中的数据有效产生中断
位 19	**RXBUF_E：** 接收 BUF 空产生中断（RxBUF empty interrupt enable） 由软件设置/清除该位，允许/关闭接收 BUF 空中断。 0：接收 BUF 空不产生中断 1：接收 BUF 空产生中断
位 18	**TXBUF_E：** 发送 BUF 空产生中断（TxBUF empty interrupt enable） 由软件设置/清除该位，允许/关闭发送 BUF 空中断。 0：发送 BUF 空不产生中断 1：发送 BUF 空产生中断
位 17	**RXBUF_F：** 接收 BUF 满产生中断（RxBUF full interrupt enable） 由软件设置/清除该位，允许/关闭接收 BUF 满中断。 0：接收 BUF 满不产生中断 1：接收 BUF 满产生中断
位 16	**TXBUF_F：** 发送 BUF 满产生中断（TxBUF full interrupt enable） 由软件设置/清除该位，允许/关闭发送 BUF 满中断。 0：发送 BUF 满不产生中断 1：发送 BUF 满产生中断
位 15	**RXBUF_H：** 接收 BUF 半满产生中断（RxBUF half full interrupt enable） 由软件设置/清除该位，允许/关闭接收 BUF 半满中断。 0：接收 BUF 半满不产生中断 1：接收 BUF 半满产生中断
位 14	**TXBUF_H：** 发送 BUF 半空产生中断（TxBUF half empty interrupt enable） 由软件设置/清除该位，允许/关闭发送 BUF 半空中断。 0：发送 BUF 半空不产生中断 1：发送 BUF 半空产生中断
位 13	**DORX：** 正在接收数据产生中断（Data receive acting interrupt enable） 由软件设置/清除该位，允许/关闭正在接收数据中断。 0：正在接收数据不产生中断 1：正在接收数据产生中断
位 12	**DOTX：** 正在发送数据产生中断（Data transmit acting interrupt enable） 由软件设置/清除该位，允许/关闭正在发送数据中断。 0：正在发送数据不产生中断 1：正在发送数据产生中断
位 11	**DOCMD：** 正在传输命令产生中断（Command acting interrupt enable） 由软件设置/清除该位，允许/关闭正在传输命令中断。 0：正在传输命令不产生中断 1：正在传输命令产生中断
位 10	**DTBLKCMPL：** 数据块传输结束产生中断（Data block end interrupt enable） 由软件设置/清除该位，允许/关闭数据块传输结束中断。 0：数据块传输结束不产生中断 1：数据块传输结束产生中断

位 9	**SBITERR**：起始位错误产生中断（Start bit error interrupt enable） 由软件设置/清除该位，允许/关闭起始位错误中断。 0：起始位错误不产生中断 1：起始位错误产生中断
位 8	**DTCMPL**：数据传输结束产生中断（Data end interrupt enable） 由软件设置/清除该位，允许/关闭数据传输结束中断。 0：数据传输结束不产生中断 1：数据传输结束产生中断
位 7	**CMDCMPL**：命令已发送产生中断（Command sent interrupt enable） 由软件设置/清除该位，允许/关闭命令已发送中断。 0：命令已发送不产生中断 1：命令已发送产生中断
位 6	**CMDRSPCMPL**：接收到响应产生中断（Command response received interrupt enable） 由软件设置/清除该位，允许/关闭接收到响应中断。 0：接收到响应不产生中断 1：接收到响应产生中断
位 5	**RXERRO**：接收 BUF 上溢错误产生中断（RxBUF overrun error interrupt enable） 由软件设置/清除该位，允许/关闭接收 BUF 上溢错误中断。 0：接收 BUF 上溢错误不产生中断 1：接收 BUF 上溢错误产生中断
位 4	**TXERRU**：发送 BUF 下溢错误产生中断（TxBUF underrun error interrupt enable） 由软件设置/清除该位，允许/关闭发送 BUF 下溢错误中断。 0：发送 BUF 下溢错误不产生中断 1：发送 BUF 下溢错误产生中断
位 3	**DTTIMEOUT**：数据超时产生中断（Data timeout interrupt enable） 由软件设置/清除该位，允许/关闭数据超时中断。 0：数据超时不产生中断 1：数据超时产生中断
位 2	**CMDTIMEOUT**：命令超时产生中断（Command timeout interrupt enable） 由软件设置/清除该位，允许/关闭命令超时中断。 0：命令超时不产生中断 1：命令超时产生中断
位 1	**DTFAIL**：数据块 CRC 检测失败产生中断（Data CRC fail interrupt enable） 由软件设置/清除该位，允许/关闭数据块 CRC 检测失败中断。 0：数据块 CRC 检测失败不产生中断 1：数据块 CRC 检测失败产生中断
位 0	**CMDFAIL**：命令 CRC 检测失败产生中断（Command CRC fail interrupt enable） 由软件设置/清除该位，允许/关闭命令 CRC 检测失败中断。 0：命令 CRC 检测失败不产生中断 1：命令 CRC 检测失败产生中断

20.4.14 SDIOBUF计数器寄存器（SDIO_BUFCNTR）

地址偏移：0x48

复位值：0x00000000

SDIO_BUFCNTR 寄存器包含还未写入 BUF 或还未从 BUF 读出的数据字数目。当在数据控制寄存器（SDIO_DTCTRL）中设置了数据传输使能位 TFREN，并且 DPSM 处于空闲状态时，BUF 计数器从数据长度寄存器（见 20.4.8 节 SDIO 数据长度寄存器（SDIO_DTLEN））加载数值。如果数据长度未与字对齐（4 的倍数），则最后剩下的 1~3 个字节被当成一个字处理。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16

保留	CNT

res								r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

CNT															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

位 31：24	保留，始终读为 0。
位 23：0	**CNT**：将要写入 BUF 或将要从 BUF 读出数据字的数目。

20.4.15 SDIO数据BUF寄存器（SDIO_BUF）

地址偏移：0x80

复位值：0x00000000

接收和发送 BUF 是一组可读或可写的 32 位宽的寄存器，它在连续的 32 个地址上包含 32 个寄存器，CPU 可以使用 BUF 读写多个操作数。

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

位 31：0	**DT**： 接收或发送 BUF 数据（Receive and transmit BUF data） BUF 数据占据 32 个 32 位的字，地址为： （SDIO 基址 + 0x80）至（SDIO 基址 + 0xFC）

21 通用串行总线全速设备接口（USBDEV）

21.1 简介

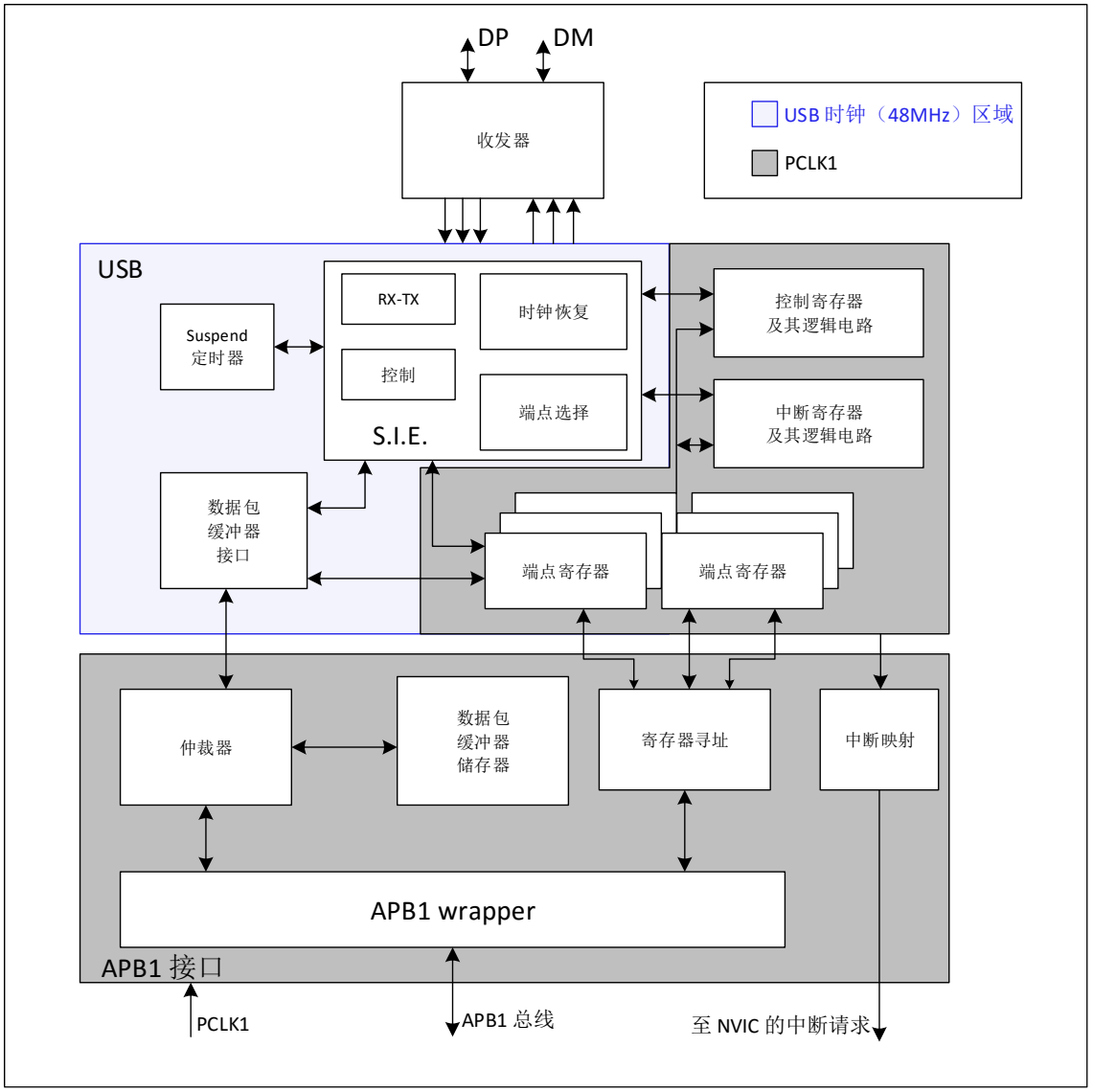
USB 外设实现了 USB2.0 全速总线和 APB1 总线间的接口。USB 外设支持 USB 挂起/恢复操作，可以停止设备时钟实现低功耗。

21.2 USB主要特点

- 符合 USB2.0 全速设备的技术规范
- 可配置 1 到 8 个 USB 端点
- CRC（循环冗余校验）生成/校验，反向不归零（NRZI）编码/解码和位填充
- 支持同步传输
- 支持批量/同步端点的双缓冲区机制
- 支持 USB 挂起/恢复操作
- 帧锁定时钟脉冲生成

下图是 USB 外设的方框图

图 21-1 USB设备框图



21.3 功能描述

USB 模块为 PC 主机和微控制器所实现的功能之间提供了符合 USB 规范的通信连接。PC 主机和微控制器之间的数据传输是通过共享一专用的数据缓冲区来完成的，该数据缓冲区能被 USB 外设直接访问。这块专用数据缓冲区的大小由所使用的端点数目和每个端点最大的数据分组大小所决定，每个端点最大可使用 512/768 字节缓冲区，最多可用于 16 个单向或 8 个双向端点。USB 模块同 PC 主机通信，根据 USB 规范实现令牌分组的检测，数据发送/接收的处理，和握手分组的处理。整个传输的格式由硬件完成，其中包括 CRC 的生成和校验。

每个端点都有一个缓冲区描述块，描述该端点使用的缓冲区地址、大小和需要传输的字节数。当 USB 模块识别出一个有效的功能/端点的令牌分组时，（如果需要传输数据并且端点已配置）随之发生相关的数据传输。USB 模块通过一个内部的 16 位寄存器实现端口与专用缓冲区的数据交换。在所有的数据传输完成后，如果需要，则根据传输的方向，发送或接收适当的握手分组。

在数据传输结束时，USB 模块将触发与端点相关的中断，通过读状态寄存器和/或利用不同的中断处理程序，微控制器可以确定：

- 哪个端点需要得到服务
- 产生如位填充、格式、CRC、协议、缺失 ACK、缓冲区溢出/缓冲区未满等错误时，正在进行的是哪种类型的传输。

USB 模块对同步传输和高吞吐量的批量传输提供了特殊的双缓冲区机制，在微控制器使用一个缓冲区的时候，该机制保证了 USB 外设总是可以使用另一个缓冲区。

在任何不需要使用 USB 模块的时候，通过写控制寄存器总可以使 USB 模块置于低功耗模式（SUSPEND 模式）。在这种模式下，不产生任何静态电流消耗，同时 USB 时钟也会减慢或停止。通过对 USB 线上数据传输的检测，可以在低功耗模式下唤醒 USB 模块。也可以将一特定的中断输入源直接连接到唤醒引脚上，以使系统能立即恢复正常的时钟系统，并支持直接启动或停止时钟系统。

21.3.1 USB功能模块描述

USB 模块实现了标准 USB 接口的所有特性，它由以下部分组成：

- 串行接口控制器（SIE）：该模块包括的功能有：帧头同步域的识别，位填充，CRC 的产生和校验，PID 的验证/产生，和握手分组处理等。它与 USB 收发器交互，利用分组缓冲接口提供的虚拟缓冲区存储局部数据。它也根据 USB 事件，和类似于传输结束或一个包正确接收等与端点相关事件生成信号，例如帧首（Start of Frame），USB 复位，数据错误等等，这些信号用来产生中断。
- 定时器：本模块的功能是产生一个与帧开始报文同步的时钟脉冲，并在 3ms 内没有数据传输的状态，检测出（主机的）全局挂起条件。
- 分组缓冲器接口：此模块管理那些用于发送和接收的临时本地内存单元。它根据 SIE 的要求分配合适的缓冲区，并定位到端点寄存器所指向的存储区地址。它在每个字节传输后，自动递增地址，直到数据分组传输结束。它记录传输的字节数并防止缓冲区溢出。
- 端点相关寄存器：每个端点都有一个与之相关的寄存器，用于描述端点类型和当前状态。对于单向和单缓冲器端点，一个寄存器就可以用于实现两个不同的端点。一共 8 个寄存器，可以用于实现最多 16 个单向/单缓冲的端点或者 7 个双缓冲的端点或者这些端点的组合。例如，可以同时实现 4 个双缓冲端点和 8 个单缓冲/单向端点。
- 控制寄存器：这些寄存器包含整个 USB 模块的状态信息，用来触发诸如恢复，低功耗等 USB 事件。
- 中断寄存器：这些寄存器包含中断屏蔽信息和中断事件的记录信息。配置和访问这些寄存器可以获取中断源，中断状态等信息，并能清除待处理中断的状态标志。

注意： 端点 0 总是作为单缓冲模式下的控制端点。

USB 模块通过 APB1 接口部件与 APB1 总线相连，APB1 接口部件包括以下部分：

- 分组缓冲区：数据分组缓存在分组缓冲区中，它由分组缓冲接口控制并创建数据结构。应用软件可以直接访问该缓冲区。它的大小为 512/（768~1280）字节，由 256/（384~640）个 16 位的字构成。
- 仲裁器：该部件负责处理来自 APB1 总线和 USB 接口的存储器请求。它通过向 APB1 提供较高的访问优先权来解决总线的冲突，并且总是保留一半的存储器带宽供 USB 完成传输。它采用时分复用的策略实现了虚拟的双端口 SRAM，即在 USB 传输的同时，允许应用程序访问存储器。此策略也允许任意长度的多字节 APB1 传输。
- 寄存器映射单元：此部件将 USB 模块的各种字节宽度和位宽度的寄存器映射成能被 APB1 寻址的 16 位宽度的内存集合。
- APB1 封装：此部件为缓冲区和寄存器提供了到 APB1 的接口，并将整个 USB 模块映射到 APB1 地址空间。
- 中断映射单元：将可能产生中断的 USB 事件映射到三个不同的 NVIC 请求线上：
  - USB 低优先级中断（通道 20 或 74）[1]：可由所有 USB 事件触发（正确传输，USB 复位等）。固件在处理中断前应当首先确定中断源。
  - USB 高优先级中断（通道 19 或 73）[1]：仅能由同步和双缓冲批量传输的正确传输事件触发，目的是保证最大的传输速率。
  - USB 唤醒中断（通道 42）：由 USB 挂起模式的唤醒事件触发。

注意 1：USBDEV 模块中断支持重新映射，RCC_INTCTRL 的 USB_INT_CTRL 位置位时，通道 19 可重映射到通道 73，通道 20 可重映射到通道 74。

21.3.2 通用USB设备编程

这一部分描述了实现 USB 设备功能的应用程序需要完成的任务。除了介绍一般的 USB 事件中应该采取的操作外，还着重介绍了双缓冲端点和同步传输的操作。这些相关的操作都是由 USB 模块初始化，并由以下几节所描述的 USB 事件所驱动。

21.3.2.1 系统复位和上电复位

发生系统复位或者上电复位时，应用程序首先需要做的是提供 USB 模块所需要的时钟信号，然后清除复位信号，使程序可以访问 USB 模块的寄存器。复位之后的初始化流程如下所述：

首先，由应用程序激活寄存器单元的时钟，再配置设备时钟管理逻辑单元的相关控制位，清除复位信号。

其次，必须配置 CTRL 寄存器的 PDWN 位用以开启 USB 收发器相关的模拟部分，这点需要特别的处理。此位能打开为端点收发器供电的内部参照电压。由于打开内部电压需要一段启动时间（数据手册中的 $t_{STARTUP}$），在此期间内 USB 收发器处于不确定状态，所以在设置 CTRL 寄存器的 PDWN 后必需等待一段时间之后，才能清除 USB 模块的复位信号（清除 CTRL 寄存器上的 FRST 位），和 INTSTS 寄存器的内容，以便在使能其他任何单元的操作之前清除未处理的假中断标志。

最后，应用程序需要通过配置设备时钟管理逻辑的相应控制位来为 USB 模块提供标准所定义的 48MHz 时钟。

当系统复位时，应用程序应该初始化所有需要的寄存器和分组缓冲区描述表，使 USB 模块能够产生正常的中断和完成数据传输。所有与端点无关的寄存器需要根据应用的需求进行初始化（比如中断使能的选择，分组缓冲区地址的选择等）。接下来按照 USB 复位处理（参见下段）。

USB 复位（RSTF 中断）

发生 USB 复位时，USB 模块进入前面章节中描述过的系统复位状态：所有端点的通信都被禁止（USB 模块不会响应任何分组）。在 USB 复位后，USB 模块被使能，同时地址为 0 的默认控制端点（端点 0）也需要被使能。这可以通过配置 USB_DEVADR 寄存器的 EN 位，EPT0 寄存器和相关的分组缓冲区来实现。在 USB 设备的枚举阶段，主机将分配给设备一个唯一的地址，这个地址必须写入 USB_DEVADR 寄存器的 ADR[6:0]位中，同时配置其他所需的端点。

当复位中断产生时，应用程序必需在中断产生后的 10ms 之内使能端点 0 的传输。

分组缓冲区的结构和用途

每个双向端点都可以接收或发送数据。接收到的数据存储在该端点指定的专用缓冲区内，而另一个缓冲区则用于存放待发送的数据。对这些缓冲区的访问由分组缓冲区接口模块实现，它提出缓冲区访问请求，并等待确认信息后返回。为防止产生微控制器与 USB 模块对缓冲区的访问冲突，缓冲区接口模块使用仲裁机制，使 APB1 总线的一半周期用于微控制器的访问，另一半保证 USB 模块的访问。这样，微控制器和 USB 模块对分组缓冲区的访问如同对一个双端口 SRAM 的访问，即使微控制器连续访问缓冲区，也不会产生访问冲突。

USB 模块使用固定的时钟，此时钟被 USB 标准定义为 48MHz。APB1 总线的时钟可以大于或者小于这个频率。

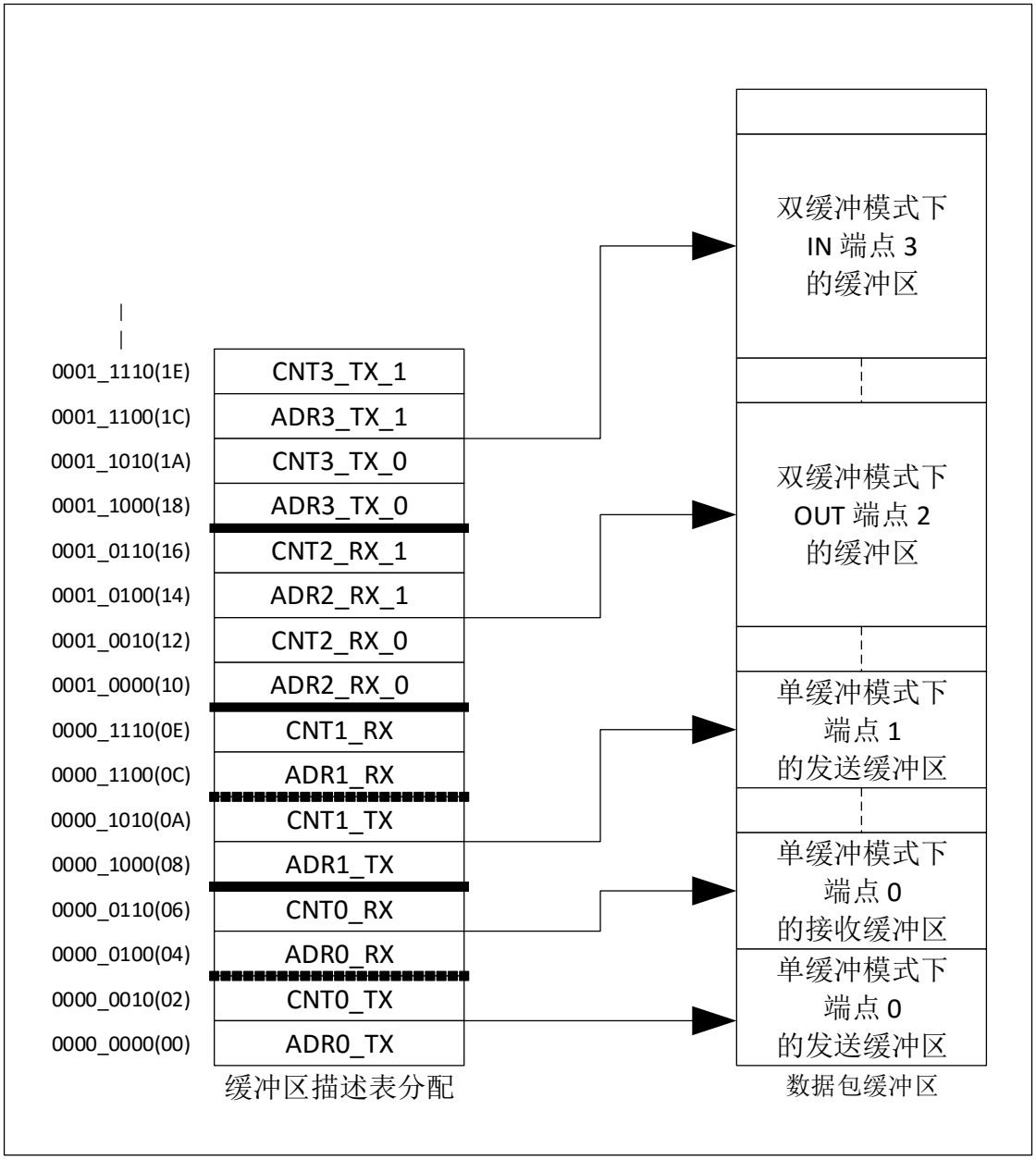
注意： 为满足 USB 数据传输率和分组缓冲区接口的系统需求，APB1 总线时钟的频率必须大于 12MHz，以避免数据缓冲区溢出或不满。

每个端点对应于两个分组缓冲区（一般一个用于发送，另一个用于接收）。这些缓冲区可以位于整个分组存储区的任意位置，因为它们的地址和长度都定义在缓冲区描述表中，而缓冲区描述表也同样位于分组缓冲区中，其地址由寄存器确定。

缓冲区描述表的每个表项都关联到一个端点寄存器，它由 4 个 16 位的字组成，因此缓冲区描述表的起始地址按 8 字节对齐（寄存器的最低 3 位总是'000'）。如果是非同步非双缓冲的单向端点，只需要一个分组缓冲区（即发送方向上的分组缓冲区）。

其他未用到的端点或某个未使用的方向上的缓冲区描述表项可以用于其他用途。同步和双缓冲批量端点有特殊的分组缓冲区处理方法。下图描述了缓冲区描述表项和分组缓冲区区域的关系。

图 21-2 分组缓冲区对应的缓冲区描述表项定位



不管是接收还是发送，分组缓冲区都是从底部开始使用的。USB 模块不会改变超出当前分配到的缓冲区区域以外的其他缓冲区的内容。如果缓冲区收到一个比自己大的数据分组，它只会接收最大为自身大小的数据，其他的丢掉，即发生了所谓的缓冲区溢出异常。

分组缓冲区空间配置

为分组缓冲区的空间分配了两片映射区间，由 RCC_MISC 寄存器的 USB768B 位来选择。当 USB768B 为 0 时，SRAM 的大小固定为 512BYTE，地址范围固定为 0x4000 6000~0x4000 63FF；当 USB768B 为 1 时，SRAM 的大小为 768~1280BYTE，地址范围为 0x4000 7800~0x4000 81FF，缓冲区的大小与 CAN1 和 CAN2 是否使能有关，当 CAN1 和 CAN2 都不使能时，USB 分组缓冲区最大可设定到 1280 字节，当 CAN1 或 CAN2 任意一个使能，USB 分组缓冲区最大可设定到 1024 字节，当 CAN1 和 CAN2 都使能时，USB 分组缓冲区最大可设定到 768 字节。

表 21-1 分组缓冲区配置表

工作条件	USB768B	0	1			
	CAN1 状态	使能/不使能	不使能	不使能	使能	使能

	CAN2 状态	使能/不使能	不使能	使能	不使能	使能
分组缓冲区大小		512BYTE	1280BYTE	1024BYTE	1024BYTE	768BYTE
分组缓冲区地址范围		0x4000 6000~ 0x4000 63FF	0x4000 7800~ 0x4000 81FF	0x4000 7800~ 0x4000 7FFF	0x4000 7800~ 0x4000 7FFF	0x4000 7800~ 0x4000 7DFF

端点初始化

初始化端点的第一步是把适当的值写到 ADRn_TX 或 ADRn_RX 寄存器中，以便 USB 模块能找到要传输的数据或准备好接收数据的缓冲区。USB_EPTx 寄存器的 EPT_TYPE 位确定端点的基本类型，EPT_SUBTYPE 位确定端点的特殊特性。作为发送方，需要设置 USB_EPTx 寄存器的 STS_TX 位来使能端点，并配置 CNTn_TX 位决定发送长度。作为接收方，需要设置 STS_RX 位来使能端点，并且设置 BLKSIZE 和 NUM_BLK 位，确定接收缓冲区的大小，以检测缓冲区溢出的异常。对于非同步非双缓冲批量传输的单向端点，只需要设置一个传输方向上的寄存器。一旦端点被使能，应用程序就不能再修改 USB_EPTx 寄存器的值和 ADRn_TX / ADRn_RX，CNTn_TX / CNTn_RX 所在的位置，因为这些值会被硬件实时修改。当数据传输完成时，CTFRF 中断会产生，此时上述寄存器可以被访问，并重新使能新的传输。

IN 分组（用于数据发送）

当接收到一 IN 令牌分组时，如果接收到的地址和一个配置好的端点地址相符合的话，USB 模块将会根据缓冲区描述表的表项，访问相应的 ADRn_TX 和 CNTn_TX 寄存器，并将这些寄存器中的数值存储到内部的 16 位寄存器 ADDR 和 COUNT（应用程序无法访问）中。此时，USB 模块开始根据 DTOG_TX 位发送 DATA0 或 DATA1 分组，并访问缓冲区。在 IN 分组传输完毕之后，从缓冲区读到的第一个字节将被装载到输出移位寄存器中， 并开始发送。最后一个数据字节发送完成之后，计算好的 CRC 将被发送。如果收到的分组所对应的端点是无效的，将根据 USB_EPTx 寄存器上的 STS_TX 位发送 NAK 或 STALL 握手分组而不发送数据。

ADDR 内部寄存器被用作当前缓冲区的指针，COUNT 寄存器用于记录剩下未传输的字节数。USB 总线使用低字节在先的方式传输从缓冲区中读出的数据。数据从 ADRn_TX 指向的数据分组缓冲区开始读取，长度为 CNTn_TX/2 个字。如果发送的数据分组为奇数个字节，则只使用最后一个字的低 8 位。

在接收到主机响应的 ACK 后，USB_EPTx 寄存器的值有以下更新：DTOG_TX 位被翻转，STS_TX 位为'10'，使端点无效，CTFR_TX 位被置位。应用程序需要通过 USB_INTSTS 寄存器的 EPT_ID 和 DIR 位识别产生中断的 USB 端点。CTFR_TX 事件的中断服务程序需要首先清除中断标志位，然后准备好需要发送的数据缓冲区，更新 CNTn_TX 为下次需要传输的字节数，最后再设置 STS_TX 位为'11'（端点有效），再次使能数据传输。当 STS_TX 位为'10'时（端点为 NAK 状态），任何发送到该端点的 IN 请求都会被 NAK，USB 主机会重发 IN 请求直到该端点确认请求有效。上述操作过程是必需遵守的，以避免丢失紧随上一次 CTFR 中断请求的下一个 IN 传输请求。

OUT 分组和 SETUP 分组（用于数据接收）

USB 模块对这两种分组的处理方式基本相同；对 SETUP 分组的特殊处理将在下面关于控制传输部分详细说明。当接收到一个 OUT 或 SETUP 分组时，如果地址和某个有效端点的地址相匹配，USB 模块将访问缓冲区描述表，找到与该端点相关的 ADRn_RX 和 CNTn_RX 寄存器，并将 ADRn_RX 寄存器的值保存在内部 ADDR 寄存器中。同时， COUNT 会被被复位，从 CNTn_RX 中读出 BLKSIZE 和 NUM_BLK 用于初始化内部 16 位寄存器 BUF_COUNT，该寄存器用于检测缓冲区溢出（所有的内部寄存器都不能被应用程序访问）。USB 模块将随后收到的数据按字方式组织（先收到的为低字节），并存储到 ADDR 指向的分组缓冲区中。同时，BUF_COUNT 值自动递减，COUNT 值自动递增。当检测到数据分组的结束信号时，USB 模块校验收到 CRC 的正确性。如果传输中没有任何错误发生，则发送 ACK 握手分组到主机。即使发生 CRC 错误或者其他类型的错误（位填充，帧错误等），数据还是会被保存到分组缓冲区中，至少会保存到发生错误的数据点，只是不会发送 ACK 分组，并且 USB_INTSTS 寄存器的 ERRF 位将会置位。在这种情况下，应用程序通常不需要干涉处理，USB 模块将从传输错误中自动恢复，并为下一次传输做好准备。如果收到的分组所对应的端点没有准备好，USB 模块将根据 USB_EPTn 寄存器的 STS_RX 位发送 NAK 或 STALL 分组，数据将不会被写入接收缓冲区。

ADRn_RX 的值决定接收缓冲区的起始地址，长度由包含 CRC 的数据分组的长度（即有效数据长度+2）决定，但不能超过 BLKSIZE 和 NUM_BLK 所定义的缓冲区的长度。如果接收到的数据分组的长度超出了

缓冲区的范围，超过范围的数据不会被写入缓冲区，USB 模块将报告缓冲区发生溢出，并向主机发送 STALL 握手分组，通知此次传输失败，也不产生中断。

如果传输正确完成，USB 模块将发送 ACK 握手分组，内部的 COUNT 寄存器的值会被复制到相应的 CNTn_RX 寄存器中，BLKSIZE 和 NUM_BLK 的值保持不变，也不需要重写。USB_EPTn 寄存器按下列方式更新：DTOG_RX 位翻转，STS_RX=10（NAK）使端点无效，CTFR_RX 位置位（如果 CTFRF 中断已使能，将触发中断）。如果传输过程中发生了错误或者缓冲区溢出，前面所列出的动作都不会发生。CTFR 中断发生时，应用程序需要首先根据 USB_INTSTS 寄存器的 EPT_ID 和 DIR 位识别是哪个端点的中断请求。在处理 CTFR_RX 中断事件时，应用程序首先要确定传输的类型（根据 USB_EPTn 寄存器的 SETUP 位），同时清除中断标志位，然后读相关的缓冲区描述表表项指向的 CNTn_RX 寄存器，获得此次传输的总字节数。处理完接收到的数据后，应用程序需要将 USB_EPTn 中的 STS_RX 位置成'11'，使能下一次的的传输。当 STS_RX 位为'10'时（NAK），任何一个发送到端点上的 OUT 请求都会被 NAK，PC 主机将不断重发被 NAK 的分组，直到收到端点的 ACK 握手分组。以上描述的操作次序是必需遵守的，以避免丢失紧随上一个 CTFR 中断的另一个 OUT 分组请求。

控制传输

控制传输由 3 个阶段组成，首先是主机发送 SETUP 分组的 SETUP 阶段，然后是主机发送零个或多个数据的数据阶段，最后是状态阶段，由与数据阶段方向相反的数据分组构成。SETUP 传输只发生在控制端点，它非常类似于 OUT 分组的传输过程。使能 SETUP 传输除了需要分别初始化 DTOG_TX 位为'1'，DTOG_RX 位为'0'外，还需要设置 STS_TX 位和 STS_RX 位为 10（NAK），由应用程序根据 SETUP 分组的相应字段决定后面的传输是 IN 还是 OUT。控制端点在每次发生 CTFR_RX 中断时，都必须检查 USB_EPTn 寄存器的 SETUP 位，以识别是普通的 OUT 分组还是 SETUP 分组。USB 设备应该能够通过 SETUP 分组中的相应数据决定数据阶段传输的字节数和方向，并且能在发生错误的情况下发送 STALL 分组，拒绝数据的传输。因此在数据阶段，未被使用到的方向都应该被设置成 STALL，并且在开始传输数据阶段的最后一个数据分组时，其反方向的传输仍设成 NAK 状态，这样，即使主机立刻改变了传输方向（进入状态阶段），仍然可以保持为等待控制传输结束的状态。在控制传输成功结束后，应用程序可以把 NAK 变为 VALID，如果控制传输出错，就改为 STALL。此时，如果状态分组是由主机发送给设备的，那么 STATUS_OUT 位（USB_EPTn 寄存器中的 EPT_SUBTYPE）应该被置位，只有这样，在状态传输过程中收到了非零长度的数据分组，才会产生传输错误。在完成状态传输阶段后，应用程序应该清除 STATUS_OUT 位，并且将 STS_RX 设为 VALID 表示已准备好接收一个新的命令请求，STS_TX 则设为 NAK， 表示在下一个 SETUP 分组传输完成前，不接受数据传输的请求。

USB 规范定义 SETUP 分组不能以非 ACK 握手分组来响应，如果 SETUP 分组传输失败，则会引发下一个 SETUP 分组。因此，以 NAK 或 STALL 分组响应主机的 SETUP 分组是被禁止的。

当 STS_RX 位被设置为'01'（STALL）或'10'（NAK）时，如果收到 SETUP 分组，USB 模块会接收分组，开始分组所要求的数据传输，并回送 ACK 握手分组。如果应用程序在处理前一个 CTFR_RX 事件时 USB 模块又收到了 SETUP 分组（即 CTFR_RX 仍然保持置位），USB 模块会丢掉收到的 SETUP 分组，并且不回答任何握手分组，以此来模拟一个接收错误，迫使主机再次发送 SETUP 分组。这样做是为了避免丢失紧随一次 CTFR_RX 中断之后的又一个 SETUP 分组传输。

21.3.2.2 双缓冲端点

USB 标准不仅为不同的传输模式定义了不同的端点类型，而且对这些数据传输所需要的系统要求做了描述。其中，批量端点适用于在主机 PC 和 USB 设备之间传输大批量的数据，因为主机可以在一帧内利用尽可能多的带宽批量传输数据，使传输效率得到提高。然而，当 USB 设备处理前一次的数据传输时，又收到新的数据分组，它将回应 NAK 分组，使 PC 主机不断重发同样的数据分组，直到设备在可以处理数据时回应 ACK 分组。这样的重传占用了很多带宽，影响了批量传输的速率，因此引入了批量端点的双缓冲机制，提高数据传输率。

使用双缓冲机制时，单向端点的数据传输将使用到该端点的接收和发送两块数据缓冲区。数据翻转位用来选择当前使用到两块缓冲区中的哪一块，使应用程序可以在 USB 模块访问其中一块缓冲区的同时，对另一块缓冲区进行操作。例如，对一个双缓冲批量端点进行 OUT 分组传输时，USB 模块将来自 PC 主机的数据保存到一个缓冲区，同时应用程序可以对另一个缓冲区中的数据进行处理（对于 IN 分组来说，情况是一样的）。

因为切换缓冲区的管理机制需要用到所有 4 个缓冲区描述表的表项，分别用来表示每个方向上的两个缓冲区的地址指针和缓冲区大小，因此用来实现双缓冲批量端点的 USB_EPTn 寄存器必需配置为单向。所

以只需要设定 STS_RX 位（作为双缓冲批量接收端点）或者 STS_TX 位（作为双缓冲批量发送端点）。如果需要一个双向的双缓冲批量端点，则须使用两个 USB_EPTn 寄存器。

为尽可能利用双缓冲的优势，达到较高的传输速率，双缓冲批量端点的流量控制流程与其他端点的稍有不同。它只在缓冲区发生访问冲突时才会设置端点为 NAK 状态，而不是在每次传输成功后都将端点设为 NAK 状态。

DTOG 位用来标识 USB 模块当前所使用的储存缓冲区。双缓冲批量端点接收方向的缓冲区由 DTOG_RX（USB_EPTn 寄存器的第 14 位）标识，而双缓冲批量端点发送方向的缓冲区由 DTOG_TX（USB_EPTn 寄存器的第 6 位）标识。同时，USB 模块也需要知道当前哪个缓冲区正在被应用程序使用，以避免发生冲突。由于 USB_EPTn 寄存器中有 2 个 DTOG 位，而 USB 模块只使用其中的一位来标识硬件所使用的缓冲区，因此，应用程序可使用另一位来标识当前正在使用哪个缓冲区，这个新的标识被称为 SW_BUF 位。下表列出了双缓冲批量端点在实现发送和接收操作时，USB_EPTn 寄存器的 DTOG 位和 SW_BUF 位之间的关系。

表 21-2 双缓冲批量端点缓冲区标识定义

缓冲区标识位	作为发送端点	作为接收端点
DTOG	DTOG_TX（USB_EPTn 寄存器的第 6 位）	DTOG_RX（USB_EPTn 寄存器的第 14 位）
SW_BUF	USB_EPTn 寄存器的第 14 位	USB_EPTn 寄存器的第 6 位

USB 模块当前使用的缓冲区由 DTOG 位标识，而应用程序所使用的缓冲区由 SW_BUF 位标识，这两个位的标识方式相同，下表描述了这种标识方式。

表 21-3 双缓冲批量端点的缓冲区使用标识

端点类型	DTOG 位	SW_BUF 位	USB 模块使用的缓冲区	应用程序使用的缓冲区
IN 端点	0	1	ADRn_TX_0 / CNTn_TX_0	ADRn_TX_1 / CNTn_TX_1
	1	0	ADRn_TX_1 / CNTn_TX_1	ADRn_TX_0 / CNTn_TX_0
	0	0	无(1)	ADRn_TX_0 / CNTn_TX_0
	1	1	无(1)	ADRn_TX_0 / CNTn_TX_0
OUT 端点	0	1	ADRn_RX_0 / CNTn_RX_0	ADRn_RX_1 / CNTn_RX_1
	1	0	ADRn_RX_1 / CNTn_RX_1	ADRn_RX_0 / CNTn_RX_0
	0	0	无(1)	ADRn_RX_0 / CNTn_RX_0
	1	1	无(1)	ADRn_RX_1 / CNTn_RX_1

注 1：端点处于 NAK 状态

可以通过以下方式设置一个双缓冲批量端点：

- 将 USB_EPTn 寄存器的 EPT_TYPE 位设为’00’，定义端点为批量端点
- 将 USB_EPTn 寄存器的 EPT_SUBTYPE 位设为’1’，定义端点为双缓冲端点

应用程序根据传输开始时用到的缓冲区来初始化 DTOG 和 SW_BUF 位；这需要考虑到这两位的数据翻转特性。设置好 DBL_BUF 位之后，每完成一次传输后，USB 模块将根据双缓冲批量端点的流量控制操作，并且持续到 DBL_BUF 变为无效为止。每次传输结束，根据端点的传输方向，CTFR_RX 位或 CTFR_TX 位将会置为’1’。与此同时，硬件将设置相应的 DTOG 位，完全独立于软件来实现缓冲区交换机制。DBL_BUF 位设置后，每次传输结束时，双缓冲批量端点的 STS 位的取值不会像其他类型端点一样受到传输过程的影响，而是一直保持为’11’（有效）。但是，如果在收到新的数据分组的传输请求时，USB 模块和应用程序发生了缓冲区访问冲突（即 DTOG 和 SW_BUF 为相同的值），状态位将会被置为’10’（NAK）。应用程序响应 CTFR 中断时，首先要清除中断标志，然后再处理传输完成的数据。应用程序访问缓冲区之后，需要翻转 SW_BUF 位，以通知 USB 模块该块缓冲区已变为可用状态。由此，双缓冲批量传输的 NAK 分组的数目只由应用程序处理一次数据传输的快慢所决定：如果数据处理的时间小于 USB 总线上完成一次数据传输的时间，则不会发生重传，此时，数据的传输率仅受限于 USB 主机。应用程序也可以不考虑双缓冲批量端点的特殊控制流程，直接在相应 USB_EPTn 寄存器的 STS 位写入非’11’的任何状态，在这种情况下，USB 模块将按照写入的状态执行流程而忽略缓冲器实际的使用情况。

21.3.2.3 同步传输

USB 标准定义了一种全速的需要保持固定和精确的数据传输率的传输方式：同步传输。同步传输一般用于传输音频流、压缩的视频流等对数据传输率有严格要求的数据。一个端点如果在枚举时被定义为“同步端点”，USB 主机则会为每个帧分配固定的带宽，并且保证每个帧正好传送一个 IN 分组或者 OUT 分组（由端点传输方向确定分组类型）。为了满足带宽要求，同步传输中没有出错重传；这也就意味着，同步传输在发送或接收数据分组之后，无握手协议，即不会发送 ACK 分组。同样，同步传输只传送 PID（分组 ID）为 DATA0 的数据包，而不会用到数据翻转机制。

通过设置 USB_EPTn 寄存器 EPT_TYPE 为’10’，可以使其成为同步端点。同步端点没有握手机制，根据 USB 标准中的说明，USB_EPTn 寄存器的 STS_RX 位和 STS_TX 位分别只能设成’00’（禁止）和’11’（有效）。同步传输通过实现双缓冲机制来简化软件应用程序开发，它同样使用两个缓冲区，以确保在 USB 模块使用其中一块缓冲区时，应用程序可以访问另外一块缓冲区。

USB 模块使用的缓冲区根据不同的传输方向，由不同的 DTOG 位来标识。（同一寄存器中的 DTOG_RX 位用来标识接收同步端点，DTOG_TX 位用来标识发送同步端点），见下表。

表 21-4 同步端点的缓冲区使用标识

端点类型	DTOG 位值	USB 模块使用的缓冲区	应用程序使用的缓冲区
IN 端点	0	ADRn_TX_0 / CNTn_TX_0	ADRn_TX_1 / CNTn_TX_1
	1	ADRn_TX_1 / CNTn_TX_1	ADRn_TX_0 / CNTn_TX_0
OUT 端点	0	ADRn_RX_0 / CNTn_RX_0	ADRn_RX_1 / CNTn_RX_1
	1	ADRn_RX_1 / CNTn_RX_1	ADRn_RX_0 / CNTn_RX_0

与双缓冲批量端点一样，一个 USB_EPTn 寄存器只能处理同步端点单方向的数据传输，如果要求同步端点在两个传输方向上都有效，则需要使用两个 USB_EPTn 寄存器。应用程序需要根据首次传输的数据分组来初始化 DTOG 位；它的取值还需要考虑到 DTOG_RX 或 DTOG_TX 两位的数据翻转特性。每次传输完成时，USB_EPTn 寄存器的 CTFR_RX 位或 CTFR_TX 位置位。与此同时，相关的 DTOG 位由硬件翻转，从而使得交换缓冲区的操作完全独立于应用程序。传输结束时，STS_RX 或 STS_TX 位不会发生变化，因为同步传输没有握手机制，所以不需要任何流量控制，而一直设为’11’（有效）。同步传输中，即使 OUT 分组发生 CRC 错误或者缓冲区溢出，本次传输仍被看作是正确的，并且可以触发 CTFR_RX 中断事件；但是，发生 CRC 错误时硬件会设置 USB_INTSTS 寄存器的 ERRF 位，提醒应用程序数据可能损坏。

21.3.2.4 挂起/恢复事件

USB 标准中定义了一种特殊的设备状态，即挂起状态，在这种状态下 USB 总线上的平均电流消耗不超过 500uA。这种电流限制对于由总线供电的 USB 设备至关重要，而自供电的设备则不需要严格遵守这样的电流消耗限制。USB 主机以 3 毫秒内不发送任何信号标志进入挂起状态。通常情况下 USB 主机每毫秒会发送一个 SOF，当 USB 模块检测到 3 个连续的 SOF 分组丢失事件即可判定主机发出了挂起请求，接着它会置位 USB_INTSTS 寄存器的 SUSPF 位，以触发挂起中断。USB 设备进入挂起状态之后，将由“唤醒”序列唤醒。所谓的“唤醒”序列， 可以由 USB 主机发起，也可以由 USB 设备本身触发；但是，只有 USB 主机可以结束“唤醒”序列。被挂起的 USB 模块必须至少还具备检测 RESET 信号的功能，它会将其当作一次正常的复位操作来执行。

实际的挂起操作过程对于不同的 USB 设备来说是不同的，因为需要不同的操作来降低电源消耗。下面描述了一起典型的挂起操作，重点介绍应用程序如何响应 USB 模块的 SUSPF 信号。

1. 将 USB_CTRL 寄存器的 FSUSP 置为’1’，这将使 USB 模块进入挂起状态。USB 模块一旦进入挂起状态，对 SOF 的检测立刻停止，以避免在 USB 挂起时又发生新的 SUSPF 事件。
2. 消除或减少 USB 模块以外的其他模块的静态电流消耗。
3. 将 USB_CTRL 寄存器的 LPWR 位置为’1’，这将消除模拟 USB 收发器的静态电流消耗，但仍能检测到唤醒信号。
4. 可以选择关闭外部振荡器和设备的 PLL，以停止设备内部的任何活动。当设备处于挂起状态时发生 USB 事件，该设备会被唤醒，并需要调用“唤醒”例程来恢复系统时钟，和 USB 数据传输。如果唤醒设备的是 USB 复位操作，则应该保证唤醒的过程不要超过 10 毫秒（参见“USB 协议规范”）。

USB 模块处于挂起状态时，唤醒或复位事件需要清除 USB_CTRL 寄存器的 LPWR 位。即使唤醒事件可以立刻触发一个 WKUPF 中断事件，但由于恢复系统时钟需要比较长的延迟时间，处理 WKUPF 中断的中断服务程序必须非常小心；为了减短系统唤醒的时间，建议将唤醒代码直接写在挂起代码后面，这样就可以在系统时钟重启后迅速进入唤醒代码中执行。

下面是唤醒操作的过程：启动外部振荡器和设备的 PLL（此项可选）。

1. 清零 USB_CTRL 寄存器的 FSUSP 位。

2. USB_FRNUM 寄存器的 DPSTS 和 DMSTS 位可以用来判断是什么触发了唤醒事件，如表 21-4 所示，它还同时列出了各种情况软件应该采取的操作。如果需要的话，可以通过检测这两位变成'10'（代表空闲总线状态）的时间来知道唤醒或复位事件的结束。此外，在复位事件结束时，USB_INTSTS 寄存器的 RSTF 位被置为'1'，如果 RSTF 中断被使能，就会产生中断。此中断应该按正常的复位操作处理。

表 21-5 唤醒事件检测

[DPSTS，DMSTS]的状态	唤醒事件	应用程序应执行的操作
00	复位	无
10	无（总线干扰）	恢复到挂起状态
01	恢复挂起	无
11	未定义的值（总线干扰）	恢复到挂起状态

设备可能不是被与 USB 模块相关的事件唤醒的（例如一个鼠标的移动可唤醒整个系统）。在这种情况下，先将 USB_CTRL 寄存器的 RESUME 位置为'1'，然后在 1ms－15ms 之间再把它清为 0 可以启动唤醒序列（这个间隔可以用 ESOF 中断来实现，该中断在内核正常运行时每 1ms 发生一次）。RESUME 位被清零后，唤醒过程将由主机 PC 完成，可以利用 USB_FRNUM 寄存器的 DPSTS 和 DMSTS 位来判断唤醒是否完成。

注意：只有在 USB 模块被设置为挂起状态时（设置 USB_CTRL 寄存器的 FSUSP 位为'1'），才可以设置 RESUME 位。

21.3.2.5 SOF输出

设备每次在 USB 线上收到 SOF 时，都会产生帧首标志 SOFF。可以从 USB 帧编号寄存器读出当前帧号。此时，会产生一个宽度为 12 个 APB1 时钟周期的 SOF 脉冲信号，通过使能 USB SOF 控制寄存器的 SOFOUTEN 位，可以将这个 SOF 脉冲信号输出到芯片引脚上。

21.4 USB寄存器

USB 模块的寄存器有以下三类：

- 通用类寄存器：中断寄存器和控制寄存器
- 端点类寄存器：端点配置寄存器和状态寄存器
- 缓冲区描述表类寄存器：用来确定数据分组存放地址的寄存器

缓冲区描述表类寄存器的基地址由 USB_BUFTBL 寄存器指定，所有其他寄存器的基地址则为 USB 模块的基地址 0x4000 5C00。由于 APB1 总线按 32 位寻址，因此所有的 16 位寄存器的地址都是按 32 位字对齐的。同样的地址对齐方式也用于从 0x4000 6000/0x4000 7800 开始的分组缓冲存储区。

可以用半字（16 位）或字（32 位）的方式操作这些外设寄存器。

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	USB_EPT0	保留																CTFR_RX	DTOG_RX	STS_RX[1:0]		SETUP	EPT_TYPE[1:0]		EPT_SUBTYPE	CTFR_TX	DTOG_TX	STS_TX[1:0]		EPTADR[3:0]			
	0x0000																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	USB_EPT1	保留																CTFR_RX	DTOG_RX	STS_RX[1:0]		SETUP	EPT_TYPE[1:0]		EPT_SUBTYPE	CTFR_TX	DTOG_TX	STS_TX[1:0]		EPTADR[3:0]			
	0x0000																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x008	USB_EPT2	保留																CTFR_RX	DTOG_RX	STS_RX[1:0]		SETUP	EPT_TYPE[1:0]		EPT_SUBTYPE	CTFR_TX	DTOG_TX	STS_TX[1:0]		EPTADR[3:0]			
	0x0000																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	USB_EPT3	保留																CTFR_RX	DTOG_RX	STS_RX[1:0]		SETUP	EPT_TYPE[1:0]		EPT_SUBTYPE	CTFR_TX	DTOG_TX	STS_TX[1:0]		EPTADR[3:0]			
	0x0000																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	USB_EPT4	保留																CTFR_RX	DTOG_RX	STS_RX[1:0]		SETUP	EPT_TYPE[1:0]		EPT_SUBTYPE	CTFR_TX	DTOG_TX	STS_TX[1:0]		EPTADR[3:0]			
	0x0000																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14	USB_EPT5	保留																CTFR_RX	DTOG_RX	STS_RX[1:0]		SETUP	EPT_TYPE[1:0]		EPT_SUBTYPE	CTFR_TX	DTOG_TX	STS_TX[1:0]		EPTADR[3:0]			
	0x0000																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	USB_EPT6	保留																CTFR_RX	DTOG_RX	STS_RX[1:0]		SETUP	EPT_TYPE[1:0]		EPT_SUBTYPE	CTFR_TX	DTOG_TX	STS_TX[1:0]		EPTADR[3:0]			
	0x0000																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

0x1C	USB_EPT7	保留	CTFR_RX	DTOG_RX	STS_RX[1:0]		SETUP	EPT_TYPE[1:0]		EPT_SUBTYPE	CTFR_TX	DTOG_TX	STS_TX[1:0]		EPTADR[3:0]																		
	0x0000		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0														
0x40	USB_CTRL	保留	CTFR_IEN	PMOVR_IEN	ERR_IEN	WKUP_IEN	SUSP_IEN	RST_IEN	SOF_IEN	ESOF_IEN	保留								RESUME	FSUSP	LPWR	PDWN	FRST										
	0x0003		0	0	0	0	0	0	0	0									0	0	0	1	1										
0x44	USB_INTSTS	保留	CTFRF	PMOVERF	ERRF	WKUPF	SUSPF	RSTF	SOFF	ESOF	保留								DIR	EPT_ID[3:0]													
	0x0000		0	0	0	0	0	1	0	0									0	0	0	0	0										
0x48	USB_FRNUM	保留	DPSTS	DMSTS	LCK	LSOF[1:0]		FRNUM[10:0]																									
	0x0XXX		0	0	0	0	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X										
0x4C	USB_DEVADDR	保留	EN	ADR[6:0]																													
	0x0000		0	0	0	0	0	0	0	0																							
0x50	USB_BUFTBL	保留	ADR15:3]																保留														
	0x0000		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0															
0x60	USB_SOFEN	保留	PUO	SOFOUTEN																													
	0x0000		0	0																													

21.4.1 通用寄存器

这组寄存器用于定义 USB 模块的工作模式，中断的处理，设备的地址和读取当前帧的编号。

21.4.1.1 USB控制寄存器（USB_CTRL）

地址偏移：0x40

复位值：0x0003

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CTFR_IEN	PMOVR_IEN	ERR_IEN	WKUP_IEN	SUSP_IEN	RST_IEN	SOF_IEN	ESOF_IEN	保留			RESUME	FUSP	LP WR	PD WN	FR ST
rw	rw	rw	rw	rw	rw	rw	rw	res			rw	rw	rw	rw	rw

位 15	**CTFR_IEN：**正确传输（CTFRF）中断屏蔽位 （Correct transfer interrupt mask） 0：正确传输（CTFRF）中断禁止 1：正确传输（CTFRF）中断使能，在中断寄存器的相应位被置 1 时产生中断。
位 14	**PMOVR_IEN：**分组缓冲区溢出中断屏蔽位 （Packet memory area over/underrun interrupt mask） 0：PMOVR 中断禁止 1：PMOVR 中断使能，在中断寄存器的相应位被置 1 时产生中断。
位 13	**ERR_IEN：**出错中断屏蔽位 （Error interrupt mask） 0：出错中断禁止 1：出错中断使能，在中断寄存器的相应位被置 1 时产生中断。
位 12	**WKUP_IEN：**唤醒中断屏蔽位 （Wakeup interrupt mask） 0：唤醒中断禁止 1：唤醒中断使能，在中断寄存器的相应位被置 1 时产生中断。
位 11	**SUSP_IEN：**挂起中断屏蔽位 （Suspend mode interrupt mask） 0：挂起（SUSPF）中断禁止 1：挂起（SUSPF）中断使能，在中断寄存器的相应位被置 1 时产生中断。
位 10	**RST_IEN：**USB 复位中断屏蔽位 （USB reset interrupt mask） 0：USB RSTF 中断禁止 1：USB RSTF 中断使能，在中断寄存器的相应位被置 1 时产生中断。
位 9	**SOF_IEN：**帧首中断屏蔽位 （Start of frame interrupt mask） 0：SOFF 中断禁止 1：SOFF 中断使能，在中断寄存器的相应位被置 1 时产生中断。
位 8	**ESOF_IEN：**期望帧首中断屏蔽位 （Expected start of frame interrupt mask） 0：ESOF 中断禁止 1：ESOF 中断使能，在中断寄存器的相应位被置 1 时产生中断。
位 4	**RESUME：**唤醒请求 （Resume request） 设置此位将向 PC 主机发送唤醒请求。根据 USB 协议，如果此位在 1ms 到 15ms 内保持有效，主机将对 USB 模块实行唤醒操作。
位 3	**FSUSP：**强制挂起 （Force suspend） 当 USB 总线上保持 3ms 没有数据通信时，SUSPF 中断会被触发，此时软件必需设置此位。 0：无效 1：进入挂起模式，USB 模拟收发器的时钟和静态功耗仍然保持。如果需要进入低功耗状态（总线供电类的设备），应用程序需要先置位 FSUSP 再置位 LPWR。
位 2	**LPWR：**低功耗模式 （Low-power mode） 此模式用于在 USB 挂起状态下降低功耗。在此模式下，除了外接上拉电阻的供电，其他的静态功耗都被关闭，系统时钟将会停止或者降低到一定的频率来减少耗电。USB 总线上的活动（唤醒事件）将会复位此位（软件也可以复位此位）。 0：非低功耗模式 1：低功耗模式
位 1	**PDWN：**断电模式 （Power down） 此模式用于彻底关闭 USB 模块。当此位被置位时，不能使用 USB 模块。 0：退出断电模式 1：进入断电模式
位 0	**FRST：**强制 USB 复位 （Force USB Reset） 0：清除 USB 复位信号 1：对 USB 模块强制复位，类似于 USB 总线上的复位信号。USB 模块将一直保持在复位状态下直到软件清除此位。如果 USB 复位中断被使能，将产生一个复位中断。

21.4.1.2 USB中断状态寄存器（USB_INTSTS）

地址偏移：0x44

复位值：0x0400

15	14	13	12	11	10	9	8	7 6 5	4	3 2 1 0
CTFRF	PMOVERF	ERRF	WKUPF	SUSPF	RSTF	SOFF	ESOF	保留	DIR	EPT_ID[3：0]
r	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	res	r	r

此寄存器包含所有中断源的状态信息，以供应用程序确认产生中断请求的事件。

寄存器的高 8 位各表示一个中断源。当相关事件发生时，这些位被硬件置位，如果 USB_CTRL 寄存器上的相应位也被置位，则会产生相应的中断。中断服务程序需要检查每个位，在执行必要的操作后必需清除相应的状态位，不然中断信号线一直保持为高，同样的中断会再次被触发。如果同时多个中断标志被设置，也只会产生一个中断应用程序可以使用不同的方式处理传输完成中断，以减少中断响应的延迟时间。端点在成功完成一次传输后，CTFRF 位会被硬件置起，如果 USB_CTRL 上的相应位也被设置的话，就会产生中断。与端点相关的中断标志和 USB_CTRL 寄存器的 CTFR_IEN 位无关。这两个中断标志位将一直保持有效，直到应用程序清除了 USB_EPTn 寄存器中的相关中断挂起位（CTFRF 位是个只读位）。USB 模块有两路中断请求源：

高优先级的 USB IRQ：用于高优先级的端点（同步和双缓冲批量端点）的中断请求，并且该中断不能被屏蔽。

低优先级 USB IRQ：用于其他中断事件，可以是低优先级的不可屏蔽中断，也可以是由 USB_INTSTS 寄存器的高 8 位标识的可屏蔽中断。对于端点产生的中断，应用程序可以通过 DIR 寄存器和 EPT_ID 只读位来识别中断请求由哪个端点产生，并调用相应的中断服务程序。

用户在处理同时发生的多个中断事件时，可以在中断服务程序里检查 USB_INTSTS 寄存器各个位的顺序来确定这些事件的优先级。在处理完相应位的中断后需要清零该中断标志。完成一次中断服务后，另一中断请求将会产生，用以请求处理剩下的中断事件。

为了避免意外清零某些位，建议使用加载指令，对所有不需改变的位写'1'，对需要清除的位写'0'。对于该寄存器，不建议使用读出－修改－写入的流程，因为在读写操作之间，硬件可能需要设置某些位，而这些位会在写入时被清零。

下面详细描述每个位：

位 15	**CTFRF：**正确的传输 （Correct transfer） 此位在端点正确完成一次数据传输后由硬件置位。应用程序可以通过 DIR 和 EPT_ID 位来识别是哪个端点完成了正确的数据传输。 此位应用程序只读
位 14	**PMOVERF：**分组缓冲区溢出 （Packet memory area over / underrun） 此位在微控制器长时间没有及时响应一个 USB 分组缓冲区访问请求时，由硬件置位。在发生这个事件后，USB 模块采用以下方式进行处理：处于接收过程时，在握手阶段，不发送 ACK 握手分组，或者处于发送过程时，在数据流中强制产生比特填充错误，在以上两种情况下主机都会要求数据重传。在正常的数据传输中不会产生 PMOVERF 中断。由于失败的传输都将由主机发起重传，应用程序就可以在这个中断的服务程序中加速设备的操作，并准备重传。但这个中断不会在同步传输中产生（同步传输不支持重传）因此数据可能会丢失。 此位应用程序可读可写，但只有写 0 有效，写 1 无效。

位 13	**ERRF：**出错 （Error） 在下列错误发生时硬件会置位此位。 NANS：无应答。主机的应答超时。 CRC：循环冗余校验码错误。数据或令牌分组中的 CRC 校验出错。 BST：位填充错误。PID，数据或 CRC 中检测出位填充错误。 FVIO：帧格式错误。收到非标准帧（如 EOP 出现在错误的时刻，错误的令牌等）。 USB 应用程序通常可以忽略这些错误，因为 USB 模块和主机在发生错误时都会启动重传机制。 此位产生的中断可以用于应用程序的开发阶段，可以用来监测 USB 总线的传输质量，标识用户可能发生的错误（连接线松，环境干扰严重，USB 线损坏等）。 此位应用程序可读可写，但只有写 0 有效，写 1 无效。
位 12	**WKUPF：**唤醒请求 （Wakeup） 当 USB 模块处于挂起状态时，如果检测到唤醒信号，此位将由硬件置位。此时 CTRL 寄存器的 LPWR 位将被清零，同时 USB_WAKEUP 被激活，通知设备的其他部分（如唤醒单元）将开始唤醒过程。 此位应用程序可读可写，但只有写 0 有效，写 1 无效。
位 11	**SUSPF：**挂起模块请求 （Suspend mode request） 此位在 USB 线上超过 3ms 没有信号传输时由硬件置位，用以指示一个来自 USB 总线的挂起请求。USB 复位后硬件立即使能对挂起信号的检测，但在挂起模式下（FSUSP＝1）硬件不会再检测挂起信号直到唤醒过程结束。 此位应用程序可读可写，但只有写 0 有效，写 1 无效。
位 10	**RSTF：**USB 复位请求 （USB reset request） 此位在 USB 模块检测到 USB 复位信号输入时由硬件置位。此时 USB 模块将复位内部协议状态机，并在中断使能的情况下触发复位中断来响应复位信号。USB 模块的发送和接收部分将被禁止，直到此位被清除。所有的配置寄存器不会被复位，除非应用程序对他们清零。这用来保证在复位后 USB 传输还可以立即正确执行。但设备的地址和端点寄存器会被 USB 复位所复位。 此位应用程序可读可写，但只有写 0 有效，写 1 无效。
位 9	**SOFF：**帧首标志 （Start of frame） 此位在 USB 模块检测到总线上的 SOF 分组时由硬件置位，标志一个新的 USB 帧的开始。中断服务程序可以通过检测 SOF 事件来完成与主机的 1ms 同步，并正确读出寄存器在收到 SOF 分组时的更新内容（此功能在同步传输时非常有意义）。 此位应用程序可读可写，但只有写 0 有效，写 1 无效。
位 8	**ESOF：**期望帧首标识位 （Expected start of frame） 此位在 USB 模块未收到期望的 SOF 分组时由硬件置位。主机应该每毫秒都发送 SOF 分组，但如果 USB 模块没有收到，挂起定时器将触发此中断。如果连续发生 3 次 ESOF 中断，也就是连续 3 次未收到 SOF 分组，将产生 SUSPF 中断。即使在挂起定时器未被锁定时发生 SOF 分组丢失，此位也会被置位。 此位应用程序可读可写，但只有写 0 有效，写 1 无效。
位 7：5	保留
位 4	**DIR：**传输方向 （Direction of transaction） 此位在完成数据传输产生中断后由硬件根据传输方向写入。 如果 DIR＝0，相应端点的 CTFR_TX 位被置位，标志一个 IN 分组（数据从 USB 模块传输到 PC 主机）的传输完成。 如果 DIR＝1，相应端点的 CTFRF_RX 位被置位，标志一个 OUT 分组（数据从 PC 主机传输到 USB 模块）的传输完成。如果 CTFR_TX 位同时也被置位，就标志同时存在挂起的 OUT 分组和 IN 分组。应用程序可以利用该信息访问 USB_EPTn 位对应的操作，它表示挂起中断传输方向的信息。 该位为只读。
位 3：0	**EPT_ID[3：0]：**端点 ID （Endpoint Identifier） 此位在 USB 模块完成数据传输产生中断后由硬件根据请求中断的端点号写入。如果同时有多个端点的请求中断，硬件写入优先级最高的端点号。端点的优先级按以下方法定义：同步端点和双缓冲批量端点具有高优先级，其他的端点为低优先级。如果多个同优先级的端点请求中断，则根据端点号来确定优先级，即端点 0 具有最高优先级，端点号越小，优先级越高。应用程序可以通过上述的优先级策略顺序处理端点的中断请求。 该位为只读。

21.4.1.3 USB帧编号寄存器（USB_FRNUM）

地址偏移：0x48

复位值：0x0XXX，X 代表未定义数值

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DPSTS	DMSTS	LCK	LSOF[1:0]		FRNUM[10:0]										
r	r	r	r		r										

位 15	**DPSTS：**D+状态位（Receive data + line status） 此位用于观察 USB D+数据线的状态，可在挂起状态下检测唤醒条件的出现。
位 14	**DMSTS：**D-状态位（Receive data - line status） 此位用于观察 USB D-数据线的状态，可在挂起状态下检测唤醒条件的出现。
位 13	**LCK：**锁定位 （Locked） USB 模块在复位或唤醒序列结束后会检测 SOF 分组，如果连续检测到至少 2 个 SOF 分组，则硬件会置位此位。此位一旦锁定，帧计数器将停止计数，一直等到 USB 模块复位或总线挂起时再恢复计数。
位 12：11	**LSOF[1:0]：**帧首丢失标志位（Lost SOF） 当 ESOF 事件发生时，硬件会将丢失的 SOF 分组的数目写入此位。如果再次收到 SOF 分组，引脚会清除此位。
位 10：0	**FRNUM[10:0]：**帧编号（Frame number） 此部分记录了最新收到的 SOF 分组中的 11 位帧编号。主机每发送一个帧，帧编号都会自加，这对于同步传输非常有意义。此部分发生 SOFF 中断时更新。

21.4.1.4 USB设备地址寄存器（USB_DEVADR）

地址偏移：0x4C

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留								EN	ADR[6：0]						
res								rw	rw						

位 7	**EN：**USB 模块使能位 （Enable function） 此位在需要使能 USB 模块时由应用程序置位。如果此位为 0，USB 模块将停止工作，忽略所有寄存器的设置，不响应任何 USB 通信。
位 6：0	**ADR[6:0]：**设备地址 （Device address） 此位记录了 USB 主机在枚举过程中为 USB 设备分配的地址值。该地址值和端点地址（EPTADR）必需和 USB 令牌分组中的地址信息匹配，才能在指定的端点进行正确的 USB 传输。

21.4.1.5 USB分组缓冲区描述表地址寄存器（USB_BUFTBL）

地址偏移：0x50

复位值：0x0000

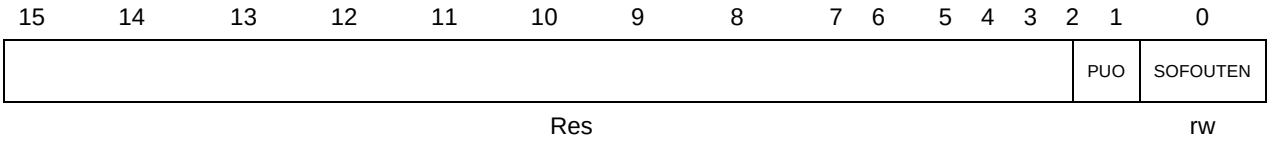
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADR[15:3]													保留		
Rw													res		

位 15：3	**ADR[15:3]：**缓冲表 （Buffer table address） 此位记录分组缓冲区描述表的起始地址。分组缓冲区描述表用来指示每个端点的分组缓冲区地址和大小，按 8 字节对齐（即最低 3 位为 000）。每次传输开始时，USB 模块读取相应端点所对应的分组缓冲区描述表获得缓冲区地址和大小信息。
位 2：0	保留位，由硬件置为 0

21.4.1.6 USB SOF控制寄存器（USB_SOFEN）

地址偏移：0x60

复位值：0x0000



位 15:1	**保留**
位 1	**PUO：**DP 上拉电阻关闭位（DP pullup off） 0：DP 上拉电阻使能。 1：DP 上拉电阻关闭。
位 0	**SOFOUTEN：**SOF 输出使能位 （SOF output enable） 0：不输出 SOF 脉冲 1：输出 SOF 脉冲到引脚上。

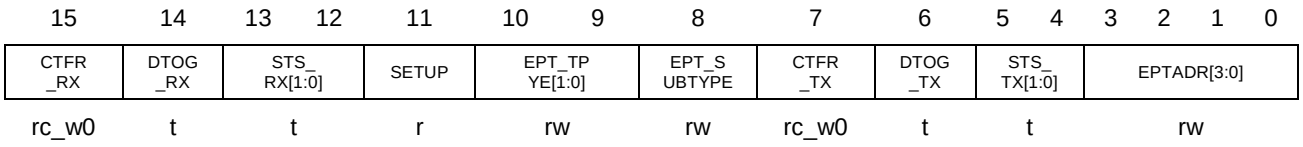
21.4.2 端点寄存器

端点寄存器的数量由 USB 模块所支持的端点数目决定。USB 模块最多支持 8 个双向端点。每个 USB 设备必须支持一个控制端点，控制端点的地址（EPTADR 位）必需为 0。不同的端点必需使用不同的端点号，否则端点的状态不定。每个端点都有与之对应的 USB_EPTn 寄存器，用于存储该端点的各种状态信息。

21.4.2.1 USB端点n寄存器（USB_EPTn），n=[0..7]

地址偏移：0x00 至 0x1C

复位值：0x0000



当 USB 模块收到 USB 总线复位信号，或 CTLR 寄存器的 FRST 位置位时，USB 模块将会复位。该寄存器除了 CTFR_RX 和 CTFR_TX 位保持不变以处理紧随的 USB 传输外，其他位都被复位。每个端点对应一个 USB_EPTn 寄存器，其中 n 为端点地址，即端点 ID 号。

对于此类寄存器应避免执行读出－修改－写入操作，因为在读和写操作之间，硬件可能会设置某些位，而这些位又会在写入时被修改，导致应用程序错过相应的操作。因此，这些位都有一个写入无效的值，建议用 Load 指令修改这些寄存器，以免应用程序修改了不需要修改的位。

位 15	**CTFR_RX：**正确接收标志位 （Correct Transfer for reception） 此位在正确接收到 OUT 或 SETUP 分组时由硬件置位，应用程序只能对此位清零。如果 CTFR_IEN 位已置位，相应的中断会产生。收到的是 OUT 分组还是 SETUP 分组可以通过下面描述的 SETUP 位确定。以 NAK 或 STALL 结束的分组和出错的传输不会导致此位置位，因为没有真正传输数据。 此位应用程序可读可写，但只有写 0 有效，写 1 无效。
位 14	**DTOG_RX：**用于数据接收的数据翻转位 （Data Toggle for reception transfers） 对于非同步端点，此位由硬件设置，用于标记希望接收的下一个数据分组的 Toggle 位（0=DATA0，1=DATA1）。在接收到 PID（分组 ID）正确的数据分组之后，USB 模块发送 ACK 握手分组，并翻转此位。对于控制端点，硬件在收到 SETUP 分组后清除此位。 对于双缓冲端点，此位还用于支持双缓冲区的交换。对于同步端点，由于仅发送 DATA0，因此此位仅用于支持双缓冲区的交换而不需进行翻转。同步传输不需要握手分组，因此硬件在收到数据分组后立即设置此位。 应用程序可以对此位进行初始化（对于非控制端点，初始化是必需的），或者翻转此位用于特殊用途。 此位应用程序可读可写，但写 0 无效，写 1 可以翻转此位。
位 13：12	**STS_RX[1:0]：**用于数据接收的状态位 （Status bits for reception transfers） 此位用于指示端点当前的状态。当一次正确的 OUT 或 SETUP 数据传输完成后（CTFR_RX=1），硬件会自动设置此位为 NAK 状态，使应用程序有足够的时间在处理完当前传输的数据后，响应下一个数据分组。 对于双缓冲批量端点，由于使用特殊的传输流量控制策略，因此根据使用的缓冲区状态控制传输状态。 对于同步端点，由于端点状态只能是有效或禁用，因此硬件不会在正确的传输之后设置此位。 如果应用程序将此位设为 STALL 或者 NAK，USB 模块响应的操作是未定义的。 此位应用程序可读可写，但写 0 无效，写 1 翻转此位。
位 11	**SETUP：**SETUP 分组传输完成标志位 （Setup transaction completed） 此位在 USB 模块收到一个正确的 SETUP 分组后由硬件置位，只有控制端点才使用此位。在接收完成后（CTFR_RX=1），应用程序需要检测此位以判断完成的传输是否是 SETUP 分组。为了防止中断服务程序在处理 SETUP 分组时下一个令牌分组修改了此位，只有 CTFR_RX 为 0 时，此位才可以被修改，CTFR_RX 为 1 时不能修改。 此位应用程序只读。
位 10：9	**EPT_TYPE[1:0]：**端点类型位 （Endpoint type） 此位用于指示端点当前的类型。所有的 USB 设备都必需包含一个地址为 0 的控制端点，如果需要可以有其他地址的控制端点。只有控制端点才会有 SETUP 传输，其他类型的端点无视此类传输。SETUP 传输不能以 NAK 或 STALL 分组响应，如果控制端点在收到 SETUP 分组时处于 NAK 状态，USB 模块将不响应分组，就会出现接收错误。如果控制端点处于 STALL 状态，SETUP 分组会被正确接收，数据会被正确传输，并产生一个正确传输完成的中断。控制端点的 OUT 分组按照普通端点的方式处理。 批量端点和中断端点的处理方式非常类似，仅在对 EPT_SUBTYPE 位的处理上有差别。
位 8	**EPT_SUBTYPE：**端点特殊类型位 （Endpoint kind） 此位需要和 EPT_TYPE 位配合使用。 DBL_BUF：应用程序设置此位能使能批量端点的双缓冲功能。 STATUS_OUT：应用程序设置此位表示 USB 设备期望主机发送一个状态数据分组，此时，设备对于任何长度不为 0 的数据分组都响应 STALL 分组。此功能仅用于控制端点，有利于提供应用程序对于协议层错误的检测。如果 STATUS_OUT 位被清除，OUT 分组可以包含任意长度的数据。
位 7	**CTFR_TX：**正确发送标志位 （Correct transfer for transmission） 此位由硬件在一个正确的 IN 分组传输完成后置位。如果 CTFR_IEN 位已被置位，会产生相应的中断。应用程序需要在处理完该事件后清除此位。在 IN 分组结束时，如果主机响应 NAK 或 STALL 则此位不会被置位，因为数据传输没有成功。 此位应用程序可读可写，但写 0 有效，写 1 无效。

位 6	**DTOG_TX**：发送数据翻转位 （Data Toggle for transmission transfers） 对于非同步端点，此位用于指示下一个要传输的数据分组的 Toggle 位（0＝DATA0，1=DATA1）。在一个成功传输的数据分组后，如果 USB 模块接收到主机发送的 ACK 分组，就会翻转此位。对于控制端点，USB 模块会在收到正确的 SETUP PID 后置位此位。 对于双缓冲端点，此位还可用于支持分组缓冲区交换。 对于同步端点，由于只传送 DATA0，因此该位只用于支持分组缓冲区交换。由于同步传输不需要握手分组，因此硬件在接收到数据分组后即设置该位。 应用程序可以初始化该位（对于非控制端点，初始化此位时必需的），也可以设置该位用于特殊用途。 此位应用程序可读可写，但写 0 无效，写 1 翻转此位。
位 5：4	**STS_TX[1:0]**：用于发送数据的状态位 （Status bits for transmission transfers） 此位用于标识端点的当前状态，表 20-8 列出了所有的状态。应用程序可以翻转这些位来初始化状态信息。在正确完成一次 IN 分组的传输后（CTFR_TX=1），硬件会自动设置此位为 NAK 状态，保证应用程序有足够的时间准备好数据响应后续的数据传输。 对于双缓冲批量端点，由于使用特殊的传输流量控制策略，是根据缓冲区的状态控制传输的状态的。 对于同步端点，由于端点的状态只能是有效或禁用，因此硬件不会在数据传输结束时改变端点的状态。如果应用程序将此位设为 STALL 或者 NAK，则 USB 模块后续的操作是未定义的。 此位应用程序可读可写，但写 0 无效，写 1 翻转此位。
位 3：0	**EPTADR[3:0]**：端点地址 （Endpoint address） 应用程序必需设置此 4 位，在使能一个端点前为它定义一个地址。

表 21-6 接收状态编码

STS_RX[1:0]	描述
00	DISABLED：端点忽略所有的接收请求。
01	STALL：端点以 STALL 分组响应所有的接收请求。
10	NAK：端点以 NAK 分组响应所有的接收请求。
11	VALID：端点可用于接收。

表 21-7 端点类型编码

EPT_TYPE[1:0]	描述
00	BULK：批量端点
01	CTRL：控制端点
10	ISO：同步端点
11	INT：中断端点

表 21-8 端点特殊类型定义

EPT_TYPE[1:0]		EPT_SUBTYPE 意义
00	BULK	DBL_BUF：双缓冲端点
01	CTRL	STATUS_OUT
10	ISO	未使用
11	INT	未使用

表 21-9 发送状态编码

STS_TX[1:0]	描述
00	DISABLED：端点忽略所有的发送请求。
01	STALL：端点以 STALL 分组响应所有的发送请求。
10	NAK：端点以 NAK 分组响应所有的发送请求。
11	VALID：端点可用于发送。

21.4.3 缓冲区描述表

虽然缓冲区描述表位于分组缓冲区内，但仍可将它看作是特殊的寄存器，用以配置 USB 模块和微控制器内核共享的分组缓冲区的地址和大小。由于 APB1 总线按 32 位寻址，所以所有的分组缓冲区地址都使用 32 位对齐的地址，而不是 USB_BUFTBL 寄存器和缓冲区描述表所使用的地址。

以下介绍两种地址表示方式：一种是应用程序访问分组缓冲区时使用的，另一种是相对于 USB 模块的本地地址。供应用程序使用的分组缓冲区地址需要乘以 2 才能得到缓冲区在微控制器中的真正地址。分组缓冲区的首地址为 0x4000 6000/0x4000 7800。下面将描述与 USB_EPTn 寄存器相关的缓冲区描述表。

21.4.3.1 发送缓冲区地址寄存器 n（USB_ADRn_TX）

地址偏移：[USB_BUFTBL] x 2 + n x 16

USB 本地地址：[USB_BUFTBL] + n x 8

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADRn_TX[15：1]															保留
rw															res

位 15：1	**ADRn_TX[15：1]：**发送缓冲区地址 （Transmission buffer address） 此位记录了收到下一个 IN 分组时，需要发送的数据所在的缓冲区起始地址。
位 0	因为分组缓冲区的地址必须按字对齐，所以此位必须为’0’。

21.4.3.2 发送数据字节数寄存器 n（USB_CNTn_TX）

地址偏移：[USB_BUFTBL] x 2 + n x 16 + 4

USB 本地地址：[USB_BUFTBL] + n x 8 + 2

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留						CNTn_TX[9：0]									
res						rw									

位 15：10	由于 USB 模块支持的最大数据分组为 1023 个字节，所以 USB 模块忽略这些位。
位 9：0	**CNTn_TX[9：0]：** 发送数据字节数 （Transmission byte count） 此位记录了收到下一个 IN 分组时要传输的数据字节数。

注意：双缓冲区和同步 IN 端点有两个 USB_CNTn_TX 寄存器：分别为 USB_CNTn _TX_1 和 USB_CNTn_TX_0。

21.4.3.3 接收缓冲区地址寄存器 n（USB_ADRn_RX）

地址偏移：[USB_BUFTBL] x 2 + n x 16 + 8

USB 本地地址：[USB_BUFTBL] + n x 8 + 4

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADRn_RX[15：1]															保留
rw															res

位 15：1	**ADRn_RX[15：1]：**接收缓冲区地址 （Reception buffer address） 此位记录了收到下一个 OUT 或者 SETUP 分组时，用于保存数据的缓冲区起始地址。
位 0	因为分组缓冲区的地址按字对齐，所以此位必需为’0’。

21.4.3.4 接收数据字节数寄存器 n（USB_CNTn_RX）

地址偏移：[USB_BTABLE] x 2 + n x 16 + 12

USB 本地地址：[USB_BTABLE] + n x 8 + 6

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BLKSIZE	NUM_BLK[4:0]					CNTn_RX[9:0]									
rw	rw					rw									

该寄存器用于存放接收分组时需要使用到的两个参数。高 6 位定义了接收分组缓冲区的大小，以便 USB 模块检测缓冲区的溢出。低 10 位则用于 USB 模块记录实际接收到的字节数。由于有效位数的限制，缓冲区的大小由分配到的存储区块数表示，而存储区块的大小则由所需的缓冲区大小决定。缓冲区的大小在设备枚举过程中定义，由端点描述符的参数 maxPacketSize 表述。（具体信息请参考"USB 2.0 协议规范"）

位 15	**BLKSIZE**：存储区块的大小 （Block size） 此位用于定义决定缓冲区大小的存储区块的大小。 如果 BLKSIZE=0，存储区块的大小为 2 字节，因此能分配的分组缓冲区的大小范围为 2－62 个字节。 如果 BLKSIZE=1，存储区块的大小为 32 字节，因此能分配的分组缓冲区的大小范围为 32－512/768 字节，符合 USB 协议定义的最大分组长度限制。
位 14：10	**NUM_BLK[4：0]**：存储区块的数目 （Number of blocks） 此位用以记录分配的存储区块的数目，从而决定最终使用的分组缓冲区的大小。
位 9：0	**CNTn_RX[9：0]**：接收到的字节数 （Reception byte count） 此位由 USB 模块写入，用以记录端点收到的最新的 OUT 或 SETUP 分组的实际字节数。

注意： 双缓冲区和同步 IN 端点有两个 USB_CNTn_RX 寄存器：分别为 USB_CNTn_RX_1 和 USB_CNTn_TX_0

表 21-10 分组缓冲区大小的定义

NUM_BLK[4：0]的值	BLKSIZE=0 时的 分组缓冲区大小	当 BLKSIZE=1 时的 分组缓冲区大小
00000	不允许使用	32 字节
00001	2 字节	64 字节
00010	4 字节	96 字节
00011	6 字节	128 字节
…	…	…
01111	30 字节	512 字节
10000	32 字节	544 字节 (1)
10001	34 字节	576 字节 (1)
10010	36 字节	608 字节 (1)
…	…	…
10111	46 字节	768 字节 (1)
11000	48 字节	800 字节 (1)
…	…	…
11110	60 字节	992 字节 (1)
11111	62 字节	1024 字节 (1)

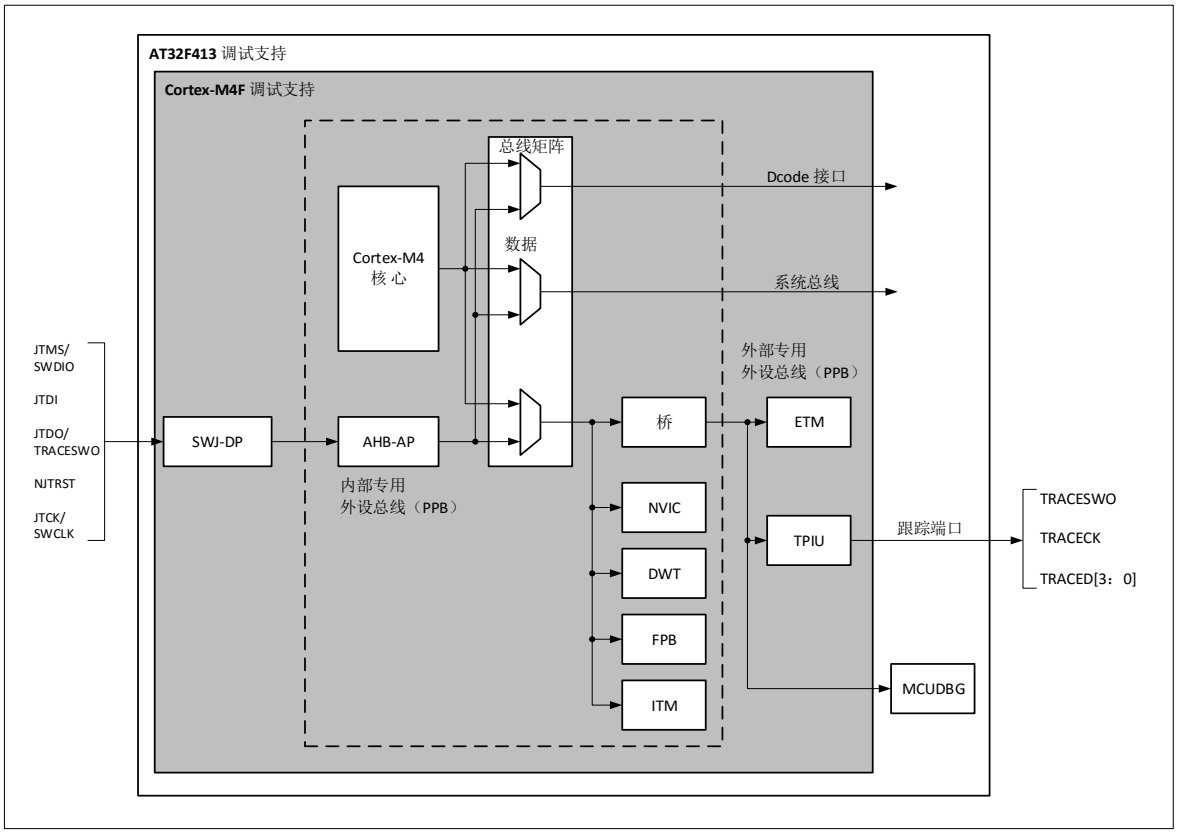
22 MCU调试（MCUDBG）

22.1 简介

AT32F403A/407 使用 Cortex™-M4F 内核，该内核内含硬件调试模块，支持复杂的调试操作。硬件调试模块允许内核在取指(指令断点)或访问数据(数据断点)时停止。内核停止时，内核的内部状态和系统的外部状态都是可以查询的。完成查询后，内核和外设可以被复原，程序将继续执行。
当 AT32F403A/407 微控制器连接到调试器并开始调试时，调试器将使用内核的硬件调试模块进行调试操作。
支持两种调试接口：

- 串行接口
- JTAG调试接口

图 22-1 AT32F403A/407级别和Cortex™-M4F级别的调试框图



ARM Cortex™-M4F 内核提供集成的片上调试功能，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

MCUDBG 模块可帮助调试器调试低功耗模式，定时器，I2C，bxCAN，WWDG 与 IWDG。当相应位置位，在低功耗模式下提供时钟或保持计数器定时器，WWDG，IWDG，CAN 或 I2C 的当前状态。MCU 调试模块协助调试器提供以下功能：

- 低功耗模式的调试支持
- 在断点时提供定时器、看门狗、I2C和bxCAN的时钟控制
- ID代码
- 对跟踪脚分配的控制

22.2 功能描述

22.2.1 低功耗模式的调试支持

使用 WFI 和 WFE 可以进入低功耗模式。MCU 支持多种低功耗模式，分别可以关闭 CPU 时钟，或降低 CPU 的能耗。内核不允许在调试期间关闭 FCLK 或 HCLK。这些时钟对于调试操作是必要的，因此在调试期间，它们必须工作。MCU 使用一种特殊的方式，允许用户在低功耗模式下调试代码。

为实现这一功能，调试器必须先设置一些配置寄存器来改变低功耗模式的特性。

- 在睡眠模式下，调试器必须先置位MCUDBG_CTRL寄存器的DBG_SLEEP位。这将为HCLK提供与FCLK（由代码配置的系统时钟）相同的时钟。
- 在停止模式下，调试器必须先置位DBG_STOP位。这将激活内部RC振荡器，在停止模式下为FCLK和HCLK提供时钟。

22.2.2 支持定时器、看门狗、bxCAN和I2C的调试

在产生断点时，有必要根据定时器和看门狗的不同用途选择计数器的工作模式：

- 在产生断点时，计数器继续计数。这在输出PWM控制电机时常常要用到。
- 在产生断点时，计数器停止计数。这对于看门狗的计数器是必需的。

对于 bxCAN，用户可以选择在断点期间阻止接收寄存器的更新。

对于 I^2C，用户可以选择在断点期间阻止 SMBUS 超时。

22.2.3 ID代码

在 AT32F403A/407 微控制器内部有多个 ID 编码，强烈建议工具设计者使用映射在外部 PPB 存储器上地址为 0xE0042000 的 MCUDEVICEID 来锁定调试器。这个 ID 定义了 MCU 的部件号和硅片版本。它是 MCUDBG 的一个组成部分，并且映射到外部 PPB 总线上。

使用 JTAG 调试口（4~5 个引脚）或 SW 调试口（2 个引脚）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。

22.2.4 SWJ调试端口脚

AT32F403A/407 的 5 个普通 I/O 口可用作 SWJ-DP 接口引脚。这些引脚在所有的封装里都存在。复位(SYSRESETn 或 PORESETn)以后，属于 SWJ-DP 的所有 5 个引脚都立即被初始化为可被调试器使用的专用引脚。

AT32F403A/407 微控制器可以用复用重映射和调试 I/O 配置寄存器(AFIO_MAPR)(见 7.5.11 节)或复用重映射和调试 I/O 配置寄存器 7(AFIO_MAPR7)(见 7.5.21 节)来禁止 SWJ-DP 接口的部分或所有引脚的功能，这些专用引脚将被释放以用作普通 I/O 口，对此寄存器的设置将由用户代码而不是调试器完成，请参考 7.4.4 节 JTAG/SWD 复用功能重映射。

22.2.5 JTAG脚上的内部上拉和下拉

保证 JTAG 的输入引脚不是悬空的是非常必要的，因为他们直接连接到 D 触发器控制着调试模式。必须特别注意 SWCLK/JTCK 引脚，因为他们直接连接到一些 D 触发器的时时钟端。

为了避免任何未受控制的 I/O 电平，JTAG 输入脚上嵌入了内部上拉和下拉。

- JNTRST：内部上拉
- JTDI：内部上拉
- JTMS/SWDIO:内部上拉
- JTCK/SWCLK:内部下拉

一旦 JTAG I/O 被用户代码释放，GPIO 控制器再次取得控制。这些 I/O 口的状态将恢复到复位时的状态。

- JNTRST：带上拉的输入
- JTDI：带上拉的输入
- JTMS/SWDIO：带上拉的输入

- JICK/SWCLK：带下拉的输入
- JTDO：浮动输入

软件可以把这些 I/O 口作为普通的 I/O 口使用。

22.2.6 跟踪脚的分配控制

跟踪引脚在默认状态下不是专用引脚。可以通过设置 MCUDBG_CTRL 寄存器的 TRACE_IOEN 和 TRACE_MODE 位来分配这些引脚。默认时，跟踪脚是不分配的。必需由调试器完成设置。

此寄存器被映射到外部 PPB 并且被 PORESET 所复位（系统复位不复位此寄存器）。调试器可以在系统复位的状态下写该寄存器。

表 22-1 灵活的跟踪引脚分配

DBGMCU_CTRL 寄存器		引脚用途	跟踪引脚分配					
TRACE_IOEN	TRACE_MODE[1:0]		PB3/JTDO/TRACESWO	PE2/TRACECK	PE3/TRACED[0]	PE4/TRACED[1]	PE5/TRACED[2]	PE6/TRACED[3]
0	XX	无跟踪（默认状态）	释放(1)	释放（可用作普通 I/O 口）				
1	00	异步跟踪	TRACESWO					
1	01	同步跟踪 1 位		TRACECK	TRACED[0]	释放（可用作普通 I/O 口）		
1	10	同步跟踪 2 位		TRACECK	TRACED[0]	TRACED[1]	释放（可用作普通 I/O 口）	
1	11	同步跟踪 4 位		TRACECK	TRACED[0]	TRACED[1]	TRACED[2]	TRACED[3]
注释（1）：使用串行调试接口时，此引脚被释放，使用 JTAG 调试接口时，此引脚用作 JTDO。								

22.3 MCUDBG寄存器

下面列出了 MCUDBG 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 22-2 MCUDBG寄存器地址映像和复位值

地址	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xE004 2000	保留																																
0xE004 2004	MCUDBG_CTRL	DBG_I2C3_SMBUS_TIME	DBG_TMRx_STOP						保留			DBG_CAN2_STOP	DBG_TMRx_STOP				DBG_I2C2_SMBUS_TIME	DBG_I2C1_SMBUS_TIME	DBG_CAN1_STOP	DBG_TMRx_STOP					DBG_WWDG_STOP	DBG_IWDG_STOP	TRACE_MODE[1:0]		TRACE_IOEN	保留		DBG_STANDBY	DBG_STOP	DBG_SLEEP
	复位值	0	0	0	0	0	0	0				0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			0	0	0

22.3.1 MCUDBG控制寄存器（MCUDBG_CTRL）

MCUDBG_CTRL 寄存器被映射到外部 PPB 总线，基地址为 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。

地址：0xE0042004（只支持 32 位访问）

POR 复位：0x00000000（不被系统复位所复位）

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DBG_I2C3_SMBUS_TIMEOUT	DBG_TMR11_STOP	DBG_TMR10_STOP	DBG_TMR9_STOP	DBG_TMR14_STOP	DBG_TMR13_STOP	DBG_TMR12_STOP	保留			DBG_CAN2_STOP	DBG_TMR7_STOP	DBG_TMR6_STOP	DBG_TMR5_STOP	DBG_TMR8_STOP	DBG_I2C2_SMBUS_TIMEOUT
res	rw	rw	rw	rw	rw	rw	res			res	res		rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DBG_I2C1_SMBUS_TIMEOUT	DBG_CAN1_STOP	DBG_TMR4_STOP	DBG_TMR3_STOP	DBG_TMR2_STOP	DBG_TMR1_STOP	DBG_WWDG_STOP	DBG_IWDG_STOP	TRACE_MODE[1：0]		TRACE_IOEN	保留		DBG_STANDBY	DBG_STOP	DBG_SLEEP
rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	res		rw	rw	rw

位 31	**DBG_I2C3_SMBUS_TIMEOUT**：当核心停止时停止 SMBUS 超时模式。 0：与正常模式操作相同； 1：冻结 SMBUS 的超时控制。
位 24：22	保留，必须保持为 0。
位 30：25 位 20：17	**DBG_TMRx_STOP**：当核心停止时停止定时器计数器（x=11...8,5） 0：当核心停止时，仍然向相关定时器的计数器提供时钟，定时器输出工作正常； 1：当核心停止时，切断相关定时器的计数器的时钟，同时关闭定时器的输出（就好象对某一暂停事件的紧急响应，停止定时器）。
位 21	**DBG_CAN2_STOP**：当核心停止时，CAN2 停止运行。 0：CAN2 仍然正常运行； 1：CAN2 的接收寄存器不继续接收数据。
位 16	**DBG_I2C2_SMBUS_TIMEOUT**：当核心停止时停止 SMBUS 超时模式。 0：与正常模式操作相同； 1：冻结 SMBUS 的超时控制。
位 15	**DBG_I2C1_SMBUS_TIMEOUT**：当核心停止时停止 SMBUS 超时模式。 0：与正常模式操作相同； 1：冻结 SMBUS 的超时控制。
位 14	**DBG_CAN1_STOP**：当核心停止时，CAN1 停止运行。 0：CAN1 仍然正常运行； 1：CAN1 的接收寄存器不继续接收数据。
位 13：10	**DBG_TMRx_STOP**：当核心停止时，计数器停止工作 x=4..1。 0：选中定时器的计数器仍然正常工作； 1：选中定时器的计数器停止工作。

位 9	**DBG_WWDG_STOP**：当核心停止时，调试窗口看门狗停止工作。 0：窗口看门狗计数器仍然正常工作； 1：窗口看门狗计数器停止工作。
位 8	**DBG_IWDG_STOP**：当核心停止时，看门狗停止工作 0：看门狗计数器仍然正常工作； 1：看门狗计数器停止工作。
位 7：5	**TRACE_MODE[1：0]**和 **TRACE_IOEN**：跟踪引脚分配控制 -当 TRACE_IOEN＝0 时：TRACE_MODE＝xx：不分配跟踪引脚（默认状态）。 -当 TRACE_IOEN=1 时：TRACE_MODE=00：跟踪引脚使用异步模式； TRACE_MODE=01：跟踪引脚使用同步模式，并且数据长度为 1；TRACE_MODE=10：跟踪引脚使用同步模式，并且数据长度为 2； TRACE_MODE=11：跟踪引脚使用同步模式，并且数据长度为 4。
位 4：3	保留，必须保持为 0。
位 2	**DBG_STANDBY**：调试待机模式。 0：（FCLK 关，HCLK 关）整个数字电路部分都断电。从软件的观点看，退出 STANDBY 模式与复位是一样的（除了一些状态位指示了微控制器刚从 STANDBY 状态退出）。 1：（FCLK 开，HCLK 开）数字电路部分不下电，FCLK 和 HCLK 时钟由内部 RLD 振荡器提供时钟。另外，微控制器通过产生系统复位来退出 STANDBY 模式和复位是一样的。
位 1	**DBG_STOP**：调试停止模式。 0：（FCLK 关，HCLK 关）在停止模式时，时钟控制器禁止一切时钟（包括 HCLK 和 FCLK）。当从 STOP 模式退出时，时钟的配置和复位之后的配置一样（微控制器由 48MHz 的内部 RC 振荡器（HSI）提供时钟）。因此，软件必需重新配置时钟控制系统启动 PLL，晶振等。 1：（FCLK 开，HCLK 开）在停止模式时，FCLK 和 HCLK 时钟由内部 RC 振荡器提供。当退出停止模式时，软件必需重新配置时钟系统启动 PLL，晶振等（与配置此比特位为 0 时的操作一样）。
位 0	**DBG_SLEEP**：调试睡眠模式 0：（FCLK 开，HCLK 关）在睡眠模式时，FCLK 由原先已配置好的系统时钟提供，HCLK 则关闭。由于睡眠模式不会复位已配置好的时钟系统，因此从睡眠模式退出时，软件不需要重新配置时钟系统。 1：（FCLK 开，HCLK 开）在睡眠模式时，FCLK 和 HCLK 时钟都由原先配置好的系统时钟提供。

23 HSI自动时钟校准（HSI Auto Clock Calibration）

23.1 ACC介绍

HSI 自动时钟校准器（HSI ACC）利用 USB 模块产生的 SOF 信号（周期为 1 毫秒）作为参考信号，实现对 HSI 时钟的采样和校准。

本模块主要功能就是实现对 USB 设备提供 48MHz±0.25%精度的时钟。

它采取“跨越回归”算法，可以将校准后的频率尽可能地靠近目标频率。

23.2 ACC主要特性

- 可配置的中心频率
- 可配置的触发校准功能的边界频率
- 满足中心频率±0.25%的精度要求
- 状态检测标志
  - 校准就绪标志
- 一个错误检测标志
  - 参考信号丢失错误
- 2 个带标志的中断源
  - 校准就绪标志
  - 参考信号丢失错误
- 两种校验方式：粗校验和精校验。

23.3 ACC功能概述

ACC 模块的功能：利用 USB 模块产生的 SOF 信号（周期为 1 毫秒）作为参考信号，实现对 HSI 时钟的采样和校准。特别的是，可以将 HSI 时钟的频率精度校准到±0.25%以内的精度，从而满足高精度时钟要求的应用场景，例如 USB 应用。

本模块的信号均未外接到芯片管脚，而是和芯片内部的 RCC、HSI 等模块相连。

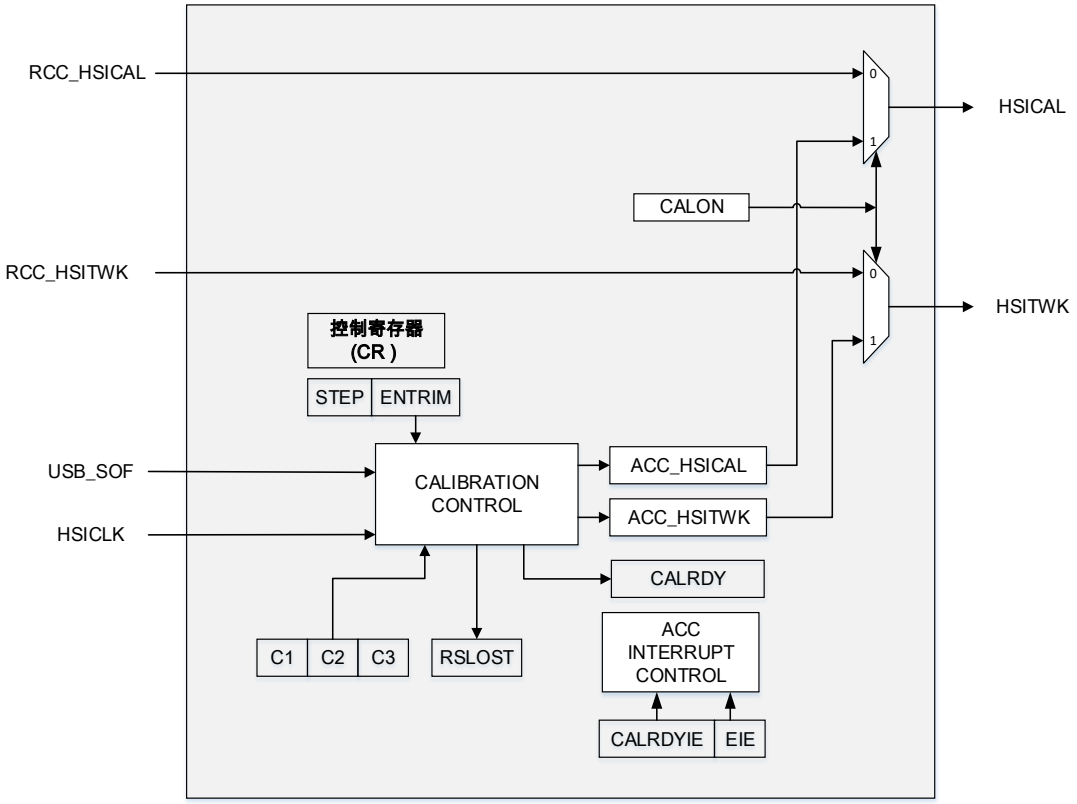
- RCC_HSICAL：复位和时钟控制（RCC）模块之 HSICAL。此信号用于 bypass 模式下对内部高速时钟（HSI 模块）的校准，其值的大小由 RCC_CTRL 中的 HSICAL[7:0]定义。
- RCC_HSITWK：复位和时钟控制（RCC）模块之 HSITWK。此信号用于 bypass 模式下对内部高速时钟校准（HSI）的校准，其值的大小由 RCC_CTRL 中的 HSITWK[5:0]定义。

  默认数值为 32，可以把 HSI 调整到 8MHz±0.25%；每步 RCC_HSITWK 的变化调整 HSI 的频率 20kHz（设计值）。
- USB_SOF：USB 设备解析给出的帧开始信号（USB Start-of-Frame）。其高电平宽度为 12 个系统时钟周期，周期为 1 毫秒的脉冲信号。
- HSICLK：HSI 时钟。本系列的 HSI 模块输出的原始时钟频率为 48MHz，但是 HSI 校准模块使用的采样时钟是除频（1/6）电路输出的时钟，频率约 8MHz。
- HSICAL：HSI 模块的校验信号。对于除频（1/6）后的 HSI 时钟来讲，HSICAL 每改变一步，除频（1/6）后 HSI 时钟频率改变 40KHz（设计值），且为正相关。换句话说，HSICAL 每增加一，除频（1/6）后 HSI 时钟频率会增加 40KHz（设计值）；HSICAL 每减少一，除频（1/6）后 HSI 时钟频率会减少 40KHz。
- HSITWK：HSI 模块的校验信号。对于除频（1/6）后的 HSI 时钟来讲，HSITWK 每改变一步，除

频（1/6）后 HSI 时钟频率改变 20KHz（设计值），且为正相关。

关于以上寄存器中每个位的具体定义，请参考寄存器描述第 23.5 节：ACC 寄存器描述。

图 23-1 ACC框图

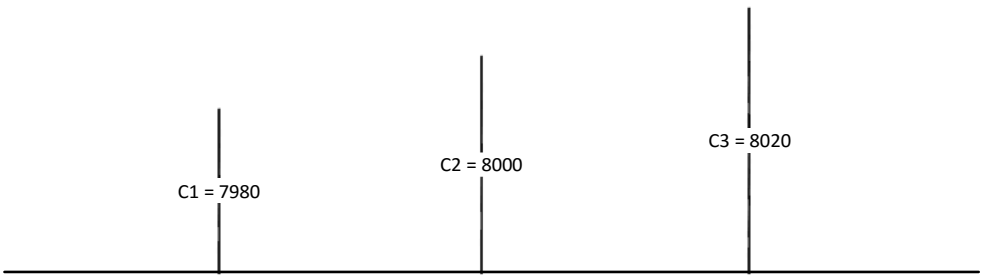


23.3.1 ACC特性描述

USB_SOF 周期信号：1 毫秒的周期性必须是准确的，是自动校准模块能够正常工作的前提条件。

cross-return 策略：以计算出离理论值最近的校准值。从理论上来说，可以将校准后的实际频率调校到离目标频率（8MHz）约 0.5 个 step 的精度范围以内。

图 23-2 cross-return策略



如上图所示，一旦触发自动校准的条件满足，自动校准就会按照 step 所规定的步长调整 HSICAL 或者 HSITWK。

跨越（cross）：

在满足自动校验的条件后的第一个 1 毫秒采样周期内的实际采样值要么小于 C2，要么大于 C2。

当这个值小于 C2，自动校准按照 step 的定义，增加 HSICAL 或者 HSITWK，直到实际采样值比 C2 大，实现实际采样值由小到大对 C2 的跨越。

当这个值大于 C2，自动校准按照 step 的定义，减少 HSICAL 或者 HSITWK，直到实际采样值比 C1 小，实现实际采样值由大到小对 C2 的跨越。

回归（return）：

在跨越完成后，比较在跨越前后的实际采样值和 C2 之间的差值（按绝对值计算），得到离 C2 最近的实际采样值，从而得到最佳的校验值 HSICAL 或者 HSITWK。

若跨越后的实际采样值和 C2 之间的差值小于跨越前的实际采样值和 C2 之间的差值，则以跨越后的校验值为准，并结束校验流程，直到满足下一个满足自动校验的条件。

若跨越后的实际采样值和 C2 之间的差值大于跨越前的实际采样值和 C2 之间的差值，则以跨越前的校验值为准，那么校验值会退回一个 step，并返回到跨越前的那个校验值，并结束校验流程，直到满足下一个满足自动校验的条件。

按照 cross-return 策略，在理论上，可以得到离中心频率约 0.5 个 step 所对应的频率精度。

如下四种情形会启动自动校准：

第一， CALON 的上升沿（从 0 到 1）；

第二， 当 CALON=1 时，参考信号丢失之后又恢复；

第三， 当采样计数器的值小于 C1；

第四， 当采样计数器的值大于 C3。

在 CALON 的上升沿，即便采样计数器的值大于 C1 并小于 C3,也会启动自动校准，其目的在于，在 CALON 之后，能够尽快将 hsi 的频率调整到中心频率的 0.5 个 step 以内。

以上四种情形的自动校准的结果均能将 hsi 的频率调整到中心频率的 0.5 个 step 以内。所以为了获得最佳的校准精度，建议将 step 保持为默认值 1。若将 step 设为 0，则 HSICAL 或者 HSITWK 将无法改变，也即，无法校准。

23.4 ACC中断请求

表 23-1 ACC中断请求

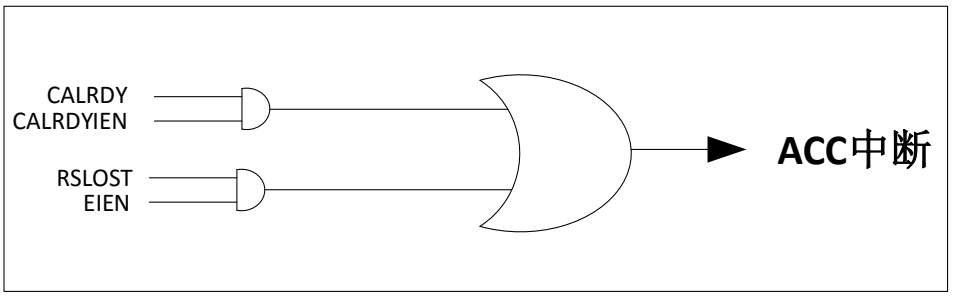
中断事件	事件标志	使能位
校准就绪	CALRDY	CALRDYIEN
参考信号丢失错误	RSLOST	EIEN

ACC 的各种中断事件被连接到同一个中断向量（见下图），有以下各种中断事件：

- 校准期间：当校准就绪和参考信号丢失错误。

如果设置了对应的使能控制位，这些事件就可以产生各自的中断。

图 23-3 ACC中断映像图



23.5 ACC寄存器描述

有关寄存器描述里所使用的缩写，请参考“寄存器描述表中使用的缩写列表”。

可以用半字（16 位）或字（32 位）的方式操作这些外设寄存器。

23.5.1 ACC寄存器地址映象

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	ACC_STS	保留																														RSLOST	CALRDY
	0x00C0																															0	0
0x0C	ACC_CTRL1	保留																				STEP				保留		CALRDYIE	EIEN	保留		ENTRIM	CALON
	0x0100																					0	0	0	1			0	0			0	0
0x0C	ACC_CTRL2	保留																		ACC_HSITWK						ACC_HSICAL							
	0x2080																			1	0	0	0	0	0	1	0	0	0	0	0	0	0
0x0C	ACC_C1	保留																C1															
	0x1F2C																	0	0	0	1	1	1	1	1	0	0	1	0	1	1	0	0
0x10	ACC_C2	保留																C2															
	0x1F40																	0	0	0	1	1	1	1	1	0	1	0	0	0	0	0	0
0x14	ACC_C3	保留																C3															
	0x1F54																	0	0	0	1	1	1	1	1	0	1	0	1	0	1	0	0

23.5.2 状态寄存器（ACC_STS）

地址偏移：0x00

复位值：0x0000

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

位 31：3	保留位，硬件强制为 0
位 1	**RSLOST**：参考信号丢失（Reference Signal Lost） 在校验过程中，当校准模块的采样计数器的值为 C2 的 2 倍时，还未侦测到 SOF 参考信号，则意味着参考信号丢失。内部状态机回归到 idle 状态，除非再次侦测到 SOF 信号，否则内部时钟采样计数器保持为 0。在 CALON 位清零后，或者向 RSLOST 写入 0，则 RSLOST 立即被清零。 0：参考信号未丢失。 1：参考信号丢失。 注意：仅仅在 CALON=1 时，才会检测参考信号。
位 0	**CALRDY**：内部高速时钟就绪（Internal high-speed clock calibration ready） 由硬件置’1’来指示内部 8MHz 振荡器已经校验到离 8MHz 最近的频率上。在 CALON 位清零后，或者向 CALRDY 写入 0，则 CALRDY 立即被清零。 0：内部 8MHz 振荡器校验没有就绪； 1：内部 8MHz 振荡器校验就绪。

23.5.3 控制寄存器1（ACC_CTRL1）

地址偏移：0x04

复位值：0x0100

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															
res															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留				STEP				保留		CALRDYIEN	EIEN	保留		ENTRIM	CALON
res				rw				rw	rw	rw	rw	rw	rw	rw	rw

位 31：12	保留位，硬件强制为 0
位 11：8	**STEP**：校准的步长 这 4 位定义了每次校准改变的值。 备注：为了获得更高的校准精度，建议将 step 设为 1。 当 ENTRIM=0，仅校准 HSICAL，若 step 改变 1，对应的 HSICAL 也改变 1，hsi 频率改变 40KHz（设计值），为正相关关系。 当 ENTRIM=1，仅校准 HSITWK，若 step 改变 1，对应的 HSITWK 也改变 1，hsi 频率改变 20KHz（设计值），为正相关关系。
位 7：6	保留位，硬件强制为 0
位 5	**CALRDYIEN**： CALRDY 中断使能（CALRDY interrupt enable） 该位由软件设置或清除。 0：禁止产生中断； 1：当 ACC_STS 中的 CALRDY 为'1'时，产生 ACC 中断。
位 4	**EIEN**： RSLOST 中断使能（RSLOST error interrupt enable） 该位由软件设置或清除。 0：禁止产生中断； 1：当 ACC_STS 中的 RSLOST 为'1'时，产生 ACC 中断。
位 3：2	保留位，硬件强制为 0
位 1	**ENTRIM**： TWK 使能（Enable trim） 该位由软件设置或清除。 备注：为了获得更高的校准精度，建议将 ENTRIM 设为 1。 0：仅校准 HSICAL； 1：仅校准 HSITWK。
位 0	**CALON**： Calibration 使能（Calibration on） 该位由软件设置或清除。 0：禁止校验； 1：使能校验，并开始搜寻 USB_SOF 上的脉冲。 备注：如果没有 USB_SOF 参考信号，则本模块无法使用。 若对 HSI 时钟的精度没有要求，也无需开启本模块以节省功耗。

23.5.4 控制寄存器2（ACC_CTRL2）

地址偏移：0x08

复位值：0x2080

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															
res															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

保留	ACC_HSITWK	ACC_HSICAL
res	r r r r r r r r r r r r r r r	

位 31：14	保留位，硬件强制为 0
位 13：8	**ACC_HSITWK**：内部高速时钟自动调整（Internal high-speed auto clock trimming） 该位由软件读取，不可写。 由 ACC 自动校准模块来调整内部高速时钟，它们被叠加在 ACC_HSICAL[7：0]数值上。这些位在 ACC_HSICAL[7：0]的基础上，让用户可以输入一个调整数值，根据电压和温度的变化调整内部 HSI RC 振荡器的频率。 默认数值为 32，可以把 HSI 调整到 8MHz±0.25%；每步 ACC_HSITWK 的变化调整 20kHz（设计值）。
位 7：0	**ACC_HSICAL**：内部高速时钟自动校准（Internal high-speed auto clock calibration） 该位由软件读取，不可写。 由 ACC 自动校准模块来调整内部高速时钟，让用户可以输入一个调整数值，根据电压和温度的变化调整内部 HSI RC 振荡器的频率。 默认数值为 128，可以把 HSI 调整到 8MHz±0.25%；每步 ACC_HSICAL 的变化调整 40kHz（设计值）。

23.5.5 比较值1（ACC_C1）

地址偏移：0x0C

复位值：0x1F2C

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															
res															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
C1															
rw															

位 31：16	保留位，硬件强制为 0。
位 15：0	**C1**：比较值 1（Compare 1） 该值是触发校准的下边界，默认值为 7980。当自动校验模块在 1 毫秒的周期内采样的时钟个数小于或等于 C1，则会触发自动校验； 当实际采样值（1 毫秒内的时钟个数）大于 C1，且小于 C3，则不会触发自动校验。

23.5.6 比较值2（ACC_C2）

地址偏移：0x10

复位值：0x1F40

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															
res															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
C2															
rw															

位 31：16	保留位，硬件强制为 0。
位 15：0	**C2**：比较值 2（Compare 2） 该值确定了理想频率（8MHz）时钟在 1 毫秒为采样周期的时钟个数，默认值为 8000，也是其理论值。 该值是 cross-return 策略的中心点，以计算出离理论值最近的校准值。从理论上来说，可以将校准后的实际频率调教到离目标频率（8MHz）约 0.5 个 step 的精度范围以内。

23.5.7 比较值3（ACC_C3）

地址偏移：0x14

复位值：0x1F54

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															
res															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
C3															
rw															

位 31：16	保留位，硬件强制为 0。
位 15：0	**C3**：比较值 3（Compare 3） 该值是触发校准的上边界。当自动校验模块在 1 毫秒的周期内采样的时钟个数大于或等于 C3，则会触发自动校验； 当实际采样值（1 毫秒内的时钟个数）大于 C1，且小于 C3，则不会触发自动校验。

24 以太网（ETH）：具有DMA控制器的介质访问控制（MAC）

仅 AT32F407 支持该模块，AT32F403A 则不支持。

24.1 以太网模块介绍

本部分内容版权归Synopsys Inc所有。已获得使用授权。

AT32F407的以太网模块支持通过以太网收发数据，符合IEEE 802.3-2002标准。

AT32F407以太网模块灵活可调，使之能适应各种不同的客户需求。该模块支持两种标准接口，连接到外接的物理层(PHY)模块：IEEE 802.3协议定义的独立于介质的接口(MII)和简化的独立于介质的接口(RMII)。适用于各类应用，如交换机、网络接口卡等。

以太网模块符合以下标准：

- IEEE 802.3-2002标准的以太网MAC协议
- IEEE 1588-2008的网路精确时钟同步标准
- AMBA2.0标准的AHB主/从端口
- RMII协会定义的RMII标准

24.2 以太网模块主要功能

按不同种类，以太网模块(ETH)主要支持以下功能：

24.2.1 MAC控制器功能

- 通过外接的PHY接口，支持10/100M位/秒的数据传输速率
- 通过兼容IEEE 802.3标准的MII接口，外接高速以太网PHY
- 支持全双工和半双工操作：
  - 支持符合CSMA/CD协议的半双工操作
  - 支持符合IEEE 802.3流控的全双工操作
  - 在全双工模式下，可以选择性地转发接收到的PAUSE控制帧到用户的应用程序
  - 支持背压流控的半双工操作
  - 在全双工模式下当输入流控信号失效时，会自动发送PAUSE帧
- 在发送时插入前导符和帧开始数据(SFD)，在接收时去掉这些域
- 以帧为单位，自动计算CRC和产生可控制的填充位
- 在接收帧时，自动去除填充位/CRC为可选项
- 可对帧长度进行编程，支持最长为16K字节的标准帧
- 可对帧间隙进行编程(40~96位，以8位为单位改变)
- 支持多种灵活的地址过滤模式：
  - 多达4个48位完美的目的地址(DA)过滤器，可在比较时屏蔽任意字节
  - 多达3个48位源地址(SADDR)比较器，可在比较时屏蔽任意字节
  - 64位Hash过滤器(可选的)，用于多播和单播(目的)地址
  - 可选的令所有的多播地址帧通过
  - 混杂模式，支持在做网络监测时不过滤，允许所有的帧直接通过
  - 允许所有接收到的数据包通过，并附带其通过每个过滤器的结果报告

- 对于发送和接收的数据包，返回独立的32位状态信息
- 支持检测接收到帧的IEEE 802.1Q VLAN标签
- 应用程序有独立的发送、接收和控制接口
- 支持使用RMON/MIB计数器(RFC2819/RFC2665)进行强制性的网络统计
- 使用MDIO接口对PHY进行配置和管理
- 检测LAN唤醒帧和AMD的Magic PacketTM帧
- 对IPv4和由以太网帧封装的TCP数据包的接收校验和卸载分流功能
- 对IPv4报头校验和以及对IPv4或IPv6数据格式封装的TCP、UDP或ICMP的校验和进行检查的高级接收功能
- 支持由IEEE 1588-2008标准定义的以太网帧时间戳，在每个帧的接收或发送状态中加上64位的时间戳。
- 两套FIFO：一个2K字节的传输FIFO，带可编程的发送阈值，和一个2K字节的接收FIFO，带可编程的接收阈值(默认值是64字节)
- 在接收FIFO的EOF后插入接收状态信息，使得多个帧可以存储在同一个接收FIFO中，而不需要开辟另一个FIFO来储存这些帧的接收状态信息
- 可以滤掉接收到的错误帧，并在存储-转发模式下，不向应用程序转发错误的帧
- 可以转发“好”的短帧给应用程序
- 支持产生脉冲来统计在接收FIFO中丢失和破坏(由于溢出)的帧数目
- 对于MAC控制器的数据传输，支持存储-转发机制
- 根据接收FIFO的填充程度(阈值可编程)，自动向MAC控制器产生PAUSE帧或背压信号
- 在发送时，如遇到冲突可以自动重发
- 在迟到冲突、冲突过多、顺延过多和欠载(underrun)情况下丢弃帧
- 软件控制清空发送FIFO
- 在存储-转发模式下，在要发送的帧内，计算并插入IPv4的报头校验和及TCP、UDP或ICMP的校验和
- 支持MII接口的内循环，可用于调试

24.2.2 DMA功能

- 在AHB从接口下，支持所有类型的AHB突发传输
- 在AHB主接口下，软件可以选择AHB突发传输的类型(固定的或者不固定长度的突发)
- 可以选择来自AHB主接口的地址对齐的突发传输
- 优化的DMA传输，传输以帧分隔符为界的数据帧
- 支持以字节对齐的方式对数据缓存区寻址
- 双缓存区(环)或链表形式的描述符列表
- 描述符的架构，使得大量的数据传输仅需要最小量的CPU介入
- 每个描述符可以传输高达8K字节的数据
- 无论正常传输还是错误传输都有完整的状态信息报告
- 可配置的发送与接收DMA突发传输长度，优化总线使用
- 可以设置以不同的操作条件产生对应的中断
- 每个帧发送/接收完成时产生中断
- 用轮换或固定优先级方式，仲裁DMA发送和接收控制器的优先级
- 开始/停止模式
- 状态寄存器指向当前发送/接收缓存区

- 状态寄存器指向当前发送/接收描述符

24.2.3 PTP功能

- 设置接收和发送帧的时间戳
- 粗调和细调的校正方法
- 当系统时间比目标时间大时，触发中断
- (通过MCU的复用功能I/O)输出秒脉冲

24.3 以太网模块引脚和内部信号

下表列出了MAC模块的信号以及相应的MII/RMII默认或是重映射的信号，同时也指出了输入或输出这些信号的引脚和引脚的配置。

表 24-1 以太网模块引脚配置

MAC 信号	MII 默认	MII 重映射	RMII 默认	RMII 重映射	引脚	引脚配置
ETH_MDC	MDC	-	MDC	-	PC1	推挽复用输出，高速(50MHz)
ETH_MII_TXD2	TXD2	-	-	-	PC2	推挽复用输出，高速(50MHz)
ETH_MII_TX_CLK	TX_CLK	-	-	-	PC3	浮空输入(复位状态)
ETH_MII_CRS	CRS	-	-	-	PA0	浮空输入(复位状态)
ETH_MII_RX_CLK ETH_RMII_REF_CLK	RX_CLK	-	REF_CLK	-	PA1	浮空输入(复位状态)
ETH_MDIO	MDIO	-	MDIO	-	PA2	推挽复用输出，高速(50MHz)
ETH_MII_COL	COL	-	-	-	PA3	浮空输入(复位状态)
ETH_MII_RX_DV ETH_RMII_CRS_DV	RX_DV	-	CRS_DV	-	PA7	浮空输入(复位状态)
ETH_MII_RXD0 ETH_RMII_RXD0	RXD0	-	RXD0	-	PC4	浮空输入(复位状态)
ETH_MII_RXD1 ETH_RMII_RXD1	RXD1	-	RXD1	-	PC5	浮空输入(复位状态)
ETH_MII_RXD2	RXD2	-	-	-	PB0	浮空输入(复位状态)
ETH_MII_RXD3	RXD3	-	-	-	PB1	浮空输入(复位状态)
ETH_MII_RX_ER	RX_ER	-	-	-	PB10	浮空输入(复位状态)
ETH_MII_TX_EN ETH_RMII_TX_EN	TX_EN	-	TX_EN	-	PB11	推挽复用输出，高速(50MHz)
ETH_MII_TXD0 ETH_RMII_TXD0	TXD0	-	TXD0	-	PB12	推挽复用输出，高速(50MHz)
ETH_MII_TXD1 ETH_RMII_TXD1	TXD1	-	TXD1	-	PB13	推挽复用输出，高速(50MHz)
ETH_PPS_OUT	PPS_OUT	-	PPS_OUT	-	PB5	推挽复用输出，高速(50MHz)
ETH_MII_TXD3	TXD3	-	-	-	PB8	推挽复用输出，高速(50MHz)
ETH_RMII_CRS_DV	-	RX_DV	-	CRS_DV	PD8	浮空输入(复位状态)
ETH_MII_RXD0 ETH_RMII_RXD0	-	RXD0	-	RXD0	PD9	浮空输入(复位状态)
ETH_MII_RXD1 ETH_RMII_RXD1	-	RXD1	-	RXD1	PD10	浮空输入(复位状态)
ETH_MII_RXD2	-	RXD2	-	-	PD11	浮空输入(复位状态)
ETH_MII_RXD3	-	RXD3	-	-	PD12	浮空输入(复位状态)

24.4 以太网模块功能描述：SMI、MII和RMII

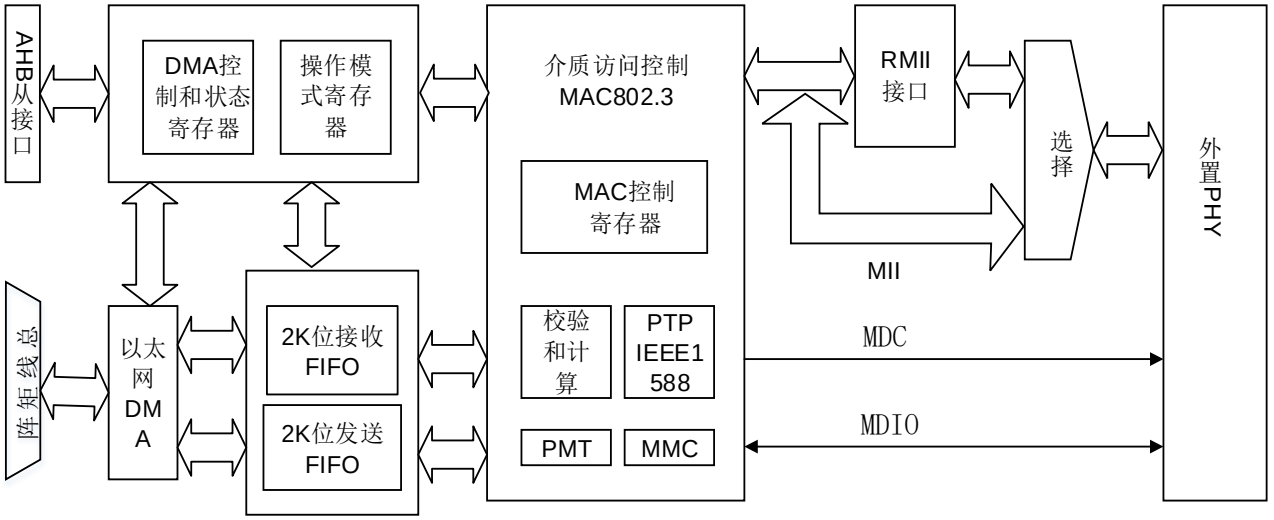
以太网模块包括一个符合802.3协议的MAC(介质访问控制器)和专用的DMA控制器。该模块支持默认的独立于介质的接口(MII)和精简的独立于介质的接口(RMII)，通过AFIO_MAPR寄存器的选择位，可以选择使用哪个接口。

DMA控制器通过AHB主和从接口，分别访问MAC控制器和存储器。AHB主接口用于控制数据传输，AHB从接口则用于访问控制和状态寄存器(CSR)区域。

在MAC控制器发送数据前，DMA会从系统存储区读出数据并储存到发送FIFO中。同样地，从总线上收到的以太网帧会储存在接收FIFO中，并由DMA传送到系统存储区。

以太网模块还包括一个SMI接口，用于和外接的PHY通信。一组配置寄存器则允许用户配置MAC和DMA控制器，以实现所需要的功能。

图 24-1 ETH框图



关于AHB的连接请查阅图 1-2 AT32F407系列微控制器系统架构

24.4.1 站点管理接口(SMI)

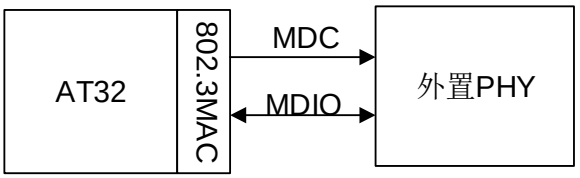
站点管理接口(SMI)允许应用程序通过时钟和数据两根线来访问任何的PHY寄存器。这个接口可以支持多达32个PHY。

应用程序可以选择32个PHY中的任意一个，并访问PHY的32个寄存器中的任意一个。但在任意时刻，只能访问一个PHY的一个寄存器。

在控制器内部，MDC时钟线和MDIO数据线都是作为复用(AF)功能的I/O端口：

- MDC：一个周期性的时钟信号，为数据的传输提供时钟，最高频率为2.5MHz。MDC信号的高电平和低电平的最小维持时间为160ns，MDC信号的最小周期为400ns。在空闲状态下，SMI接口将驱动MDC时钟信号保持在低电平状态
- MDIO：数据的输入/输出线，在MDC时钟信号的驱动下，向PHY设备传递状态信息

图 24-2 SMI接口信号



SMI帧格式

下表列出了SMI接口进行读写操作的帧的数据结构，必需遵循从左至右的发送顺序。

表 24-2 管理帧格式

	帧内容							
	前导符(32 位)	起始符	操作符	PADDR	RADDR	TA	数据(16 位)	空闲位
读	1…1	01	10	ppppp	rrrrr	Z0	dddddddddddddddd	Z
写	1…1	01	01	ppppp	rrrrr	10	dddddddddddddddd	Z

管理帧包括以下8个部分：

- 前导符：每个传输(读或者写)都必需以前导符开始，前导符是MDIO线上连续的32个逻辑’1’信号，和对应MDC线上的32个时钟信号。这部分信号用于和PHY设备建立同步
- 起始符：帧的起始符定义为’01’，也就是MDIO线从逻辑’1’降到’0’再回到’1’，以标记传输的开始

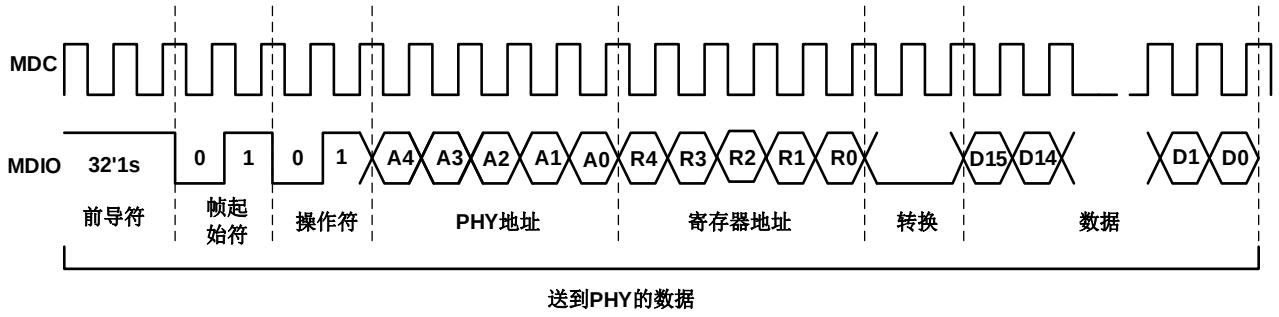
- 操作符：用于定义操作的类型：读或者写
- PADDR：PHY的地址有5位，可以区分32个PHY。高位先被发送和接收
- RADDR：寄存器的地址有5位，可以寻址32个独立的寄存器。高位先被发送和接收
- TA：2位的转向符，插在RADDR和数据(DATA)之间，用于避免读操作时发生冲突。读操作时，在TA的这2位时间内，MAC控制器保持MDIO线的高阻状态，PHY设备则先保持1位的高阻状态在第2位时输出’0’信号。写操作时，在TA的这2位时间内，MAC控制器驱动MDIO线输出’10’信号,而PHY设置则保持高阻状态
- DATA(数据)：16位的数据域。最先发送和接收的是ETH_MIID寄存器的第15位
- 空闲位：MDIO线保持在高阻状态。取消所有的三态驱动，由PHY的上拉电阻保证MDIO线处于逻辑’1’

SMI写操作

当应用程序设置了MII写和忙位(以太网MAC MII地址寄存器(ETH_MACMIIADDR))，SMI接口会向PHY传送PHY地址和PHY寄存器地址，然后传输数据(以太网MAC MII数据寄存器(ETH_MACMIIDT))。在SMI接口传输数据的过程中，不能修改MII地址寄存器和MII数据寄存器的内容；在此过程中(忙位为高)，对MII地址寄存器或MII数据寄存器的写操作将被忽视，并且不影响整个传输的正确完成。当完成写操作时，SMI接口将清除忙位，告知应用程序。

下图描述了写操作时的帧格式

图 24-3 MDIO时序和帧格式图－写操作

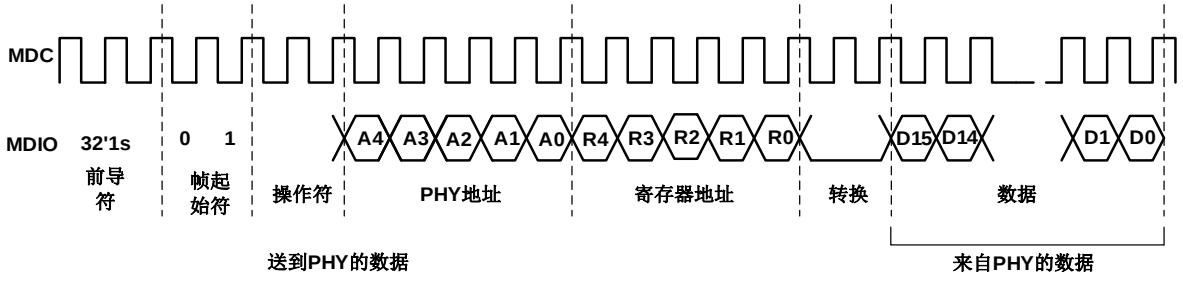


SMI读操作

当程序把以太网MAC MII地址寄存器(ETH_MACMIIADDR)的MII忙位置为’1’，而保持MII写位为’0’，SMI接口则发送PHY地址和PHY寄存器地址，执行读PHY寄存器的操作。在整个传输过程中，应用程序不能修改MII地址寄存器和MII数据寄存器的内容。在传输过程中(忙位为高)，对MII地址寄存器或者MII数据寄存器的写操作将被忽视，并且不影响整个传输的正确完成。在读操作完成后，SMI接口将清除忙位，并把从PHY读回的数据更新到MII数据寄存器中。

下图描述了读操作的帧格式

图 24-4 MDIO时序和帧格式图－读操作



SMI时钟选择

MAC控制器发起SMI的读/写操作，SMI接口的时钟源由AHB时钟分频得到。根据MII地址寄存器中设置的不同时钟频率范围，需要选择不同的分频因子。

下表列出了这个时钟范围。

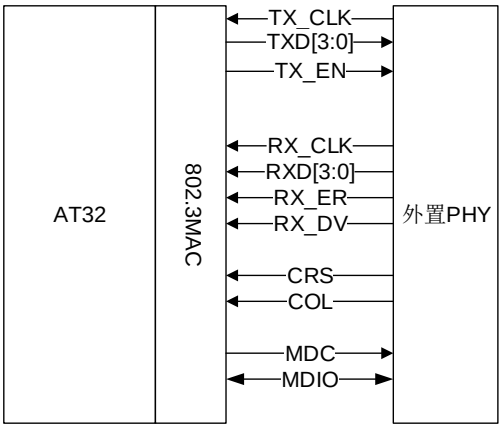
表 24-3 时钟范围

选择位	AHB 时钟	MDC 时钟
0000	60~100MHz	AHB 时钟/42
0001	100~150MHz	AHB 时钟/62
0010	20~35MHz	AHB 时钟/16
0011	35~60MHz	AHB 时钟/26
0100	150~240MHz	AHB 时钟/102
0101,0110,0111	保留	—

24.4.2 独立于介质的接口：MII

独立于介质的接口(MII)用于MAC子层和PHY之间的互联，允许10M位/s和100M位/s数据传输。

图 24-5 独立于介质的接口(MII)信号线



- MII_TX_CLK：为传输发送数据而提供连续的时钟信号，对于10M位/s的数据传输，此时钟为2.5MHz，对于100M位/s的数据传输，此时钟为25MHz
- MII_RX_CLK：为传输接收数据而提供连续的时钟信号，对于10M位/s的数据传输，此时钟为2.5MHz，对于100M位/s的数据传输，此时钟为25MHz
- MII_TX_EN：传输使能信号，表示MAC正在输出要求MII接口传输的数据。此使能信号必需与数据前导符的起始位同步(MII_TX_CLK)出现，并且必需一直保持到所有需要传输的位都传输完毕为止
- MII_TXD[3:0]：由MAC子层控制，每次同步地传输4位数据，数据在MII_TX_EN信号有效时有效。MII_TXD[0]是数据的最低位，MII_TXD[3]是最高位。当MII_TX_EN信号无效时，传输的数据对于PHY无效
- MII_CRS：载波侦听信号，由PHY控制，当发送或接收的介质非空闲时，使能此信号。当传送和接收的介质都空闲时，PHY会撤消此信号。PHY必需保证MII_CS信号在发生冲突的整个时间段内都保持有效。不需要此信号与发送/接收的时钟同步。在全双工模式下，此信号的状态对于MAC子层无意义
- MII_COL：冲突检测信号，由PHY控制，当检测到介质发生冲突时，使能此信号，并且在整个冲突的持续时间内，保持此信号有效。此信号不需要和发送/接收的时钟同步。在全双工模式下，此信号的状态对于MAC子层无意义
- MII_RXD[3:0]：由PHY控制，每次同步地发送4位需要接收的数据，数据在MII_RX_DV信号有效时有效。MII_RXD[0]是数据的最低位，MII_RXD[3]是最高位。当MII_RX_DV无效，而MII_RX_ER有效时，PHY会传送一组特殊的MII_RXD[3:0]数据(请参考表194)，来告知一些特殊的信息
- MII_RX_DV：接收数据使能信号，由PHY控制，当PHY准备好卸载和解码数据供MII接收时，使能该信号。此信号必需和卸载好的帧数据的首位同步(MII_RX_CLK)出现，并在数据完全传输完毕之前，都保持有效。在传送最后4位数据后的第一个时钟之前，此信号必需变为无效状态。为了正确的接收一个帧，MII_RX_DV信号必需在整个帧传输期间内都保持有效，有效电平不能晚于数据线上的SFD

位

- MII_RX_ER：接收出错信号，保持一个或多个时钟周期(MII_RX_CLK)的有效状态，指示MAC子层在帧内检测到了错误。错误情况必需配合MII_RX_DV的状态

表 24-4 发送接口信号编码

MII_TX_EN	MII_TXD[3:0]	描述
0	0000 到 1111	正常的帧间隔
1	0000 到 1111	正常的数据传输

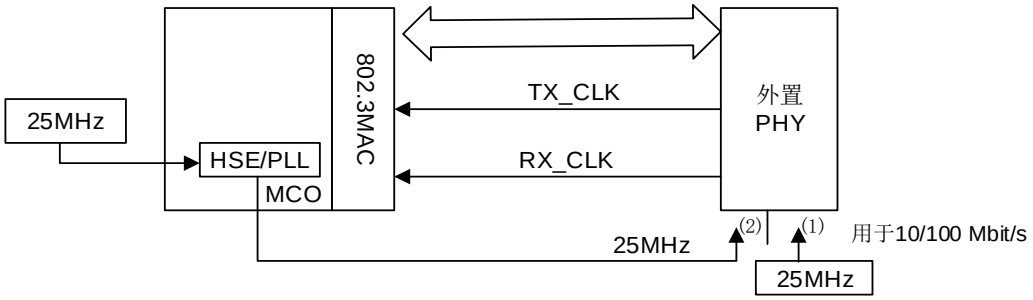
表 24-5 接收接口信号编码

MII_RX_DV	MII_RX_ERR	MII_RXD[3:0]	描述
0	0	0000 到 1111	正常的帧间隔
0	1	0000	正常的帧间隔
0	1	0001 到 1101	保留
0	1	1110	失败的载波指示
0	1	1111	保留
1	0	0000 到 1111	正常的数据接收
1	1	0000 到 1111	数据接收出错

MII时钟源

如下图所示，为了产生TX_CLK 和RX_CLK 时钟信号，外接的PHY模块必需有来自外部的25MHz时钟驱动。除了使用外部的25MHz晶体提供这一时钟外(1)，AT32F407xx微控制器也可以通过MCO引脚来提供这一时钟(2)；此时需要合适地配置PLL或者直接将来自外部25MHz晶体的MCU时钟从MCO引脚输出出去。

图 24-6 MII时钟源



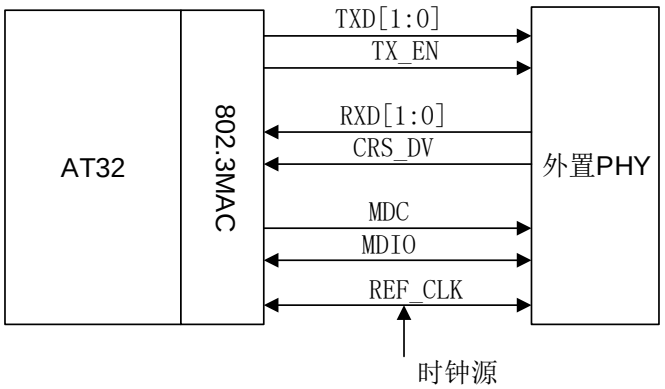
24.4.3 精简的独立于介质的接口：RMII

精简的独立于介质接口(RMII)规范减少了与10/100M位/s通信时，AT32F407xx以太网模块和外部以太网之间的引脚数。根据IEEE802.3u标准，MII接口需要16个数据和控制信号引脚，而RMII标准则将引脚数减少到了7个(减少了62.5%的引脚数目)。

RMII模块用于连接MAC和PHY，该模块将MAC的MII信号转换到RMII接口上。RMII模块具有以下特性：

- 支持10M位/s和100M位/s的通信速率
- 时钟信号需要提高到50MHz
- MAC和外部的以太网PHY需要使用同样的时钟源
- 使用2位宽度的数据收发

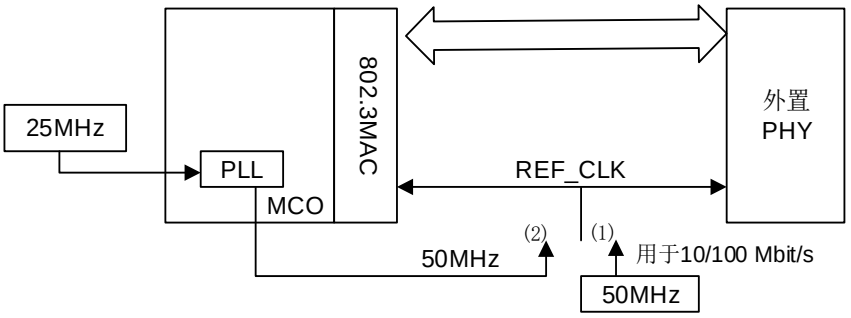
图 24-7 精简的独立于介质的接口信号



RMII时钟源

如下图所示，可以使用外部晶体振荡器提供50MHZ参考时钟(1)，也可以通过AT32F407xx控制器从MCO引脚提供50MHz时钟信号(2)，当然用户需要配置PLL来产生这一时钟。

图 24-8 RMII时钟源



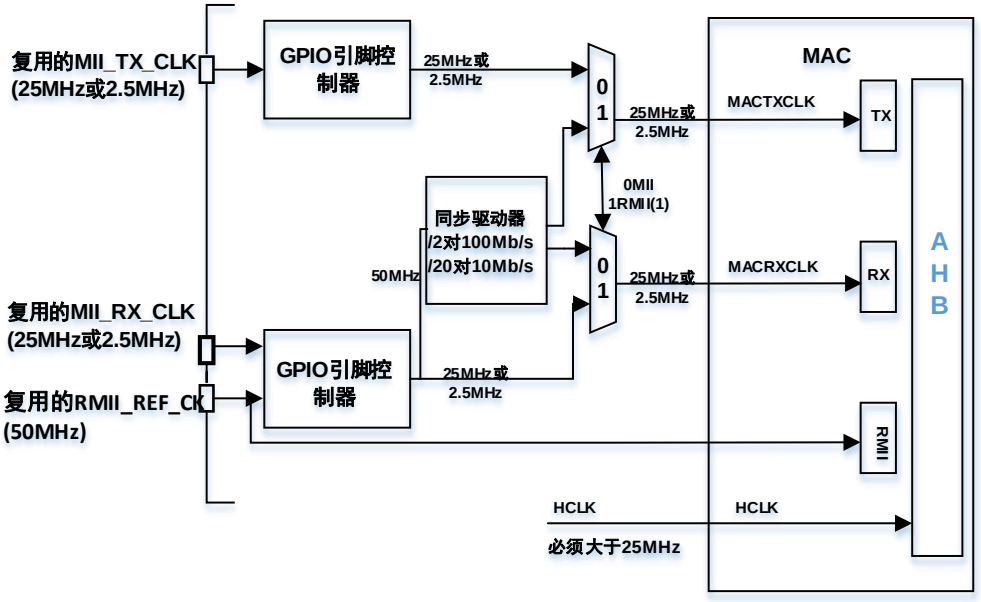
24.4.4 MII/RMII的选择

通过AFIO_MAPR寄存器的第23位MII_RMII_SEL位，可以选择使用MII或者RMII模式。必需在以太网控制器处于复位状态时、或者在使能时钟前，选择好MII/RMII模式。

MII/RMII内部时钟结构

如下图所示，内部的时钟结构应能支持MII和RMII模式，并且支持10和100M位/s的通信速率。

图 24-9 内部时钟结构



由AFIO_MAPR寄存器的第23位MII_RMII_SEL位决定选择使用MII/RMII接口。

为了节省引脚，RMII_REF_CK和MII_RX_CLK这两个时钟信号复用同一个GPIO端口。

24.5 以太网模块功能描述：MAC 802.3

IEEE802.3国际标准为局域网(LANs)定义了CSMA/CD(带冲突检测的载波侦听多路访问)的访问方式。

以太网模块由MAC 802.3(介质访问控制器)、独立于介质的接口(MII)和一个专用的DMA控制器组成。

其中，MAC模块实现了LAN CSMA/CD的子层，适用于以下系统：10M位/s和100M位/s数据传输率的基带和宽带系统。支持全双工和半双工的操作模式。冲突检测的访问方式，仅适用于半双工模式。支持MAC控制帧子层。

MAC子层配合数据链路控制程序，实现以下功能：

- 数据封装(传送和接收)
  - 帧的组装(帧间隔和帧同步)
  - 寻址(管理源地址和目的地址)
  - 错误检测
- 介质访问管理
  - 介质分配(防止冲突)
  - 冲突解决(处理冲突)

通常有两种模式可以操作 MAC 子层：

- 半双工模式：站点通过CSMA/CD算法来抢占对物理介质的访问。
- 全双工模式：满足以下条件时，同时进行收发而不处理冲突(不使用CSMA/CD)：
  - 物理介质支持同时进行收发操作
  - 只有两个站点接入LAN
  - 两个站点都配置为全双工模式

24.5.1 MAC 802.3 帧格式

MAC模块实现了IEEE 802.3-2002标准定义的MAC子层和可选的MAC控制子层(10/100M位/s)的 功能。

有两种帧格式使用CSMA/CD MAC的数据通信系统：

- 基本的MAC帧格式
- 带标签的MAC帧格式(对基本的MAC帧格式的扩展)

图 24-11 MAC帧格式和图 24-12 带标签的MAC帧格式描述了帧结构(带标签的和不带标签的)，包括以下部分：

- 前导符：7个字节，用于同步(PLS电路)。

  十六进制的值：55-55-55-55-55-55-55

  二进制：01010101 01010101 01010101 01010101 01010101 01010101 01010101 (从右至左的传输)
- 帧首符(SFD)：1个字节，用于指示帧首。

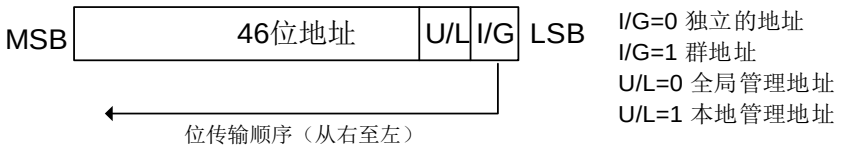
  十六进制的值：D5

  二进制：11010101 (从右至左的传输)
- 目的和源地址：6个字节，用以下方式指示目的地址和源地址(请参考图24-10)：
  - 每个地址都是48位
  - 目的地址的LSB位(I/G)用于区分单独的地址(I/G=0)或是群地址(I/G=1)。一个群地址可以指向0个、1个或多个、或LAN中所有的站点。源地址中的第一位(LSB)位为保留位，固定为’0’
  - 地址的第二位(U/L)用于区分单个(U/L=1)地址或是全球(U/L=0)地址。对于广播地址，此位固定为1
  - 地址位的每个字节都必需遵循低位先发的传输规则

地址的分配有以下的类型：

- 单独的地址：这是指向网络里的某个特定站点的物理地址
- 群地址：这是指向给定网络里的一个或多个站点的群体地址。有以下两种多发地址：
  - 多播群地址：指向一群逻辑上关联的站点的地址
  - 广播地址：是一个唯一的、预先约定好的多发地址(目的地址位全为1)，总是指向给定LAN中的所有站点

图 24-10 地址域格式



- QTag前缀：位于源地址域和MAC”客户端长度/类型”域之间的4字节标识符。此标识符属于带标签的MAC帧，是基本帧(不带标签)的扩展。基本帧不含此标识符。此标识符包括：
  - 作为常数的2个字节的”长度/类型”域，表示协议类型(大于0x0600)，等于802.1Q标签协议类型(0x8100)的值。这个常数域用于区分带标签和不带标签的MAC帧
  - 2个字节的域，包括标签的控制信息。包含以下几部分：3位的用户优先级，1位的规范格式指示位(CFI)和12位的VLAN标识符。带标签的MAC帧比基本帧多了QTag前缀的4字节
- MAC客户端长度/类型：2个字节，根据其值的不同可分为两种含义(互斥的)：
  - 如果2个字节的值等于或者小于maxValidFrame(1500)，那么这两个字节代表后续的802.3帧的数据域MAC客户数据字节数(代表长度)
  - 如果2个字节的值等于或者大于MinTypeValue(1536或0x0600)，那么这两个字节代表与以太网帧相关的MAC客户协议类型(代表类型)

  如果数据域的长度小于协议规定的最小可接收长度，则忽略长度/类型域的数据，在数据域之后、FCS(帧校验序列)域之前，添加填充(PAD)域。数据在传送和接收长度/类型域时，都是高位在前。

对于介于maxValidLength和minTypeValue(范围不相交)值之间的长度/类型值，协议没有定义MAC子层如何处理。这样的数据可能被MAC子层接受也可能不被接受

- 数据和填充域：n个字节的数据。数据字段完全透明化，也就是说任意序列的数值都可以出现在数据段中。如果存在填充域，那么填充的长度由数据的长度决定。数据和填充域的最大最小长度如下：
  - 最大长度为1500个字节
  - 不带标签的MAC帧的最短长度为46字节
  - 带标签的MAC帧的最短长度为42字节

  如果数据的长度小于标准要求的最短长度(对于标签帧为 42 字节，对于非标签帧为 46 字节)，那么就添加填充域，以符合标准要求的长度
- 帧校验序列：4个字节的循环冗余校验值(CRC)。CRC计算包含以下几个域的数值：源地址、目的地址、QTag前缀、长度/类型、LLC数据和填充(即计算除了前导和SFD域以外的所有数据)。计算多项式如下：

  $G(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$

一个帧的CRC值计算如下：

- 帧的前两位互补
- 帧的n位作为(n-1)次多项式M(x)的系数。目的地址的首位作为x^{n-1}项的系数，数据域的末位作为x^0项的系数
- M(x)多项式乘以x^{32}，再除以G(x)，产生一个次数≤31的余数R(x)
- R(x)多项式的系数就作为32位序列
- 对此序列取补就是CRC校验值
- 帧校验序列传输的就是32位的CRC校验值。x^{32}的系数最先传输，x^0的系数最后传输

图 24-11 MAC帧格式

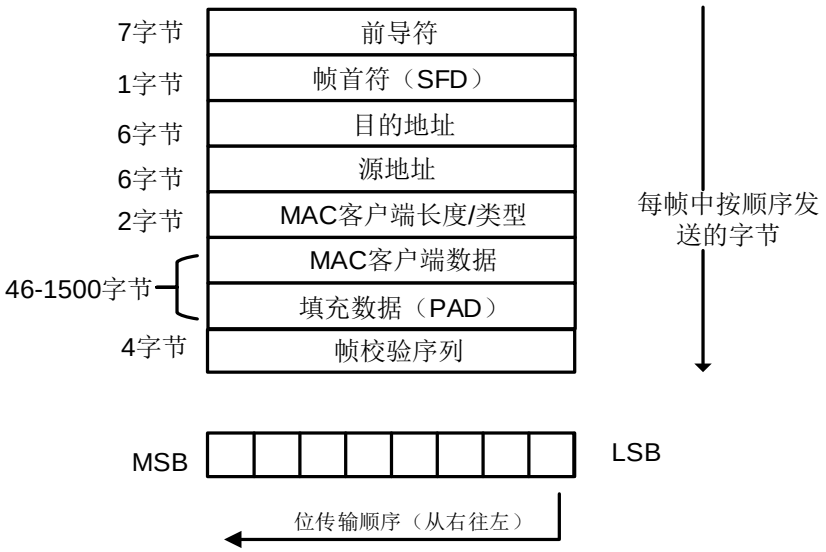
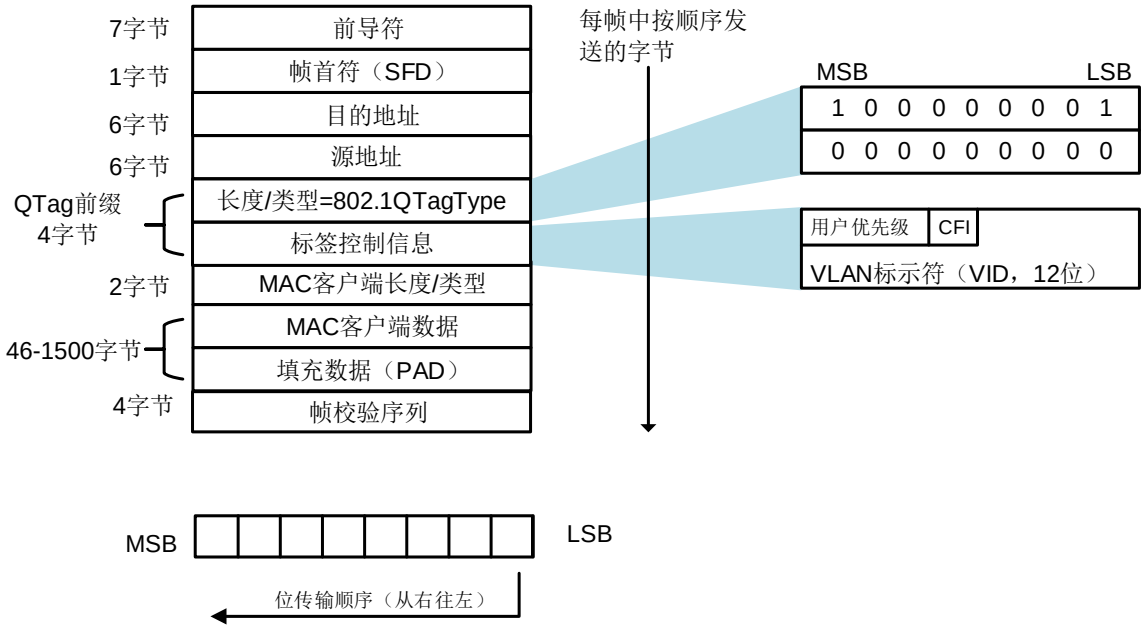


图 24-12 带标签的MAC帧格式



除了FCS域，MAC帧的序列都按照低位先出的次序进行传输。

发生以下情况，就认定MAC帧为无效：

- 帧长度与定义在长度/类型域的值不符。如果长度/类型域的数据为类型值，那么就默认帧长度与定义值一致(没有无效帧)
- 帧长度不是字节的整数倍(有多余的位)
- FCS域中的CRC值与实际帧计算出的CRC值不匹配

24.5.2 MAC帧的传输

DMA控制器管理所有的传输。DMA将以太网帧从系统存储区读出并存入FIFO中，帧再弹出并传入MAC控制器。当帧的最末位传输完毕，传输状态会由MAC控制器返回到DMA控制器中。发送的FIFO深度为2K字节。FIFO的填充程度会反馈到DMA控制器，DMA通过AHB接口发出从系统存储区获取数据的请求。数据会通过AHB主接口存入FIFO中。

当检测到SOF信号后，MAC接收数据，并开始传输数据至MII。由于各种延迟因素，比如FG延迟、发送前导/SFD域的时间、和半双工模式下的退避等待等，从应用程序启动传输，到数据帧发送到MII的时间是不定的。当EOF信号传输到MAC控制器后，控制器完成正常传输，并返还传输状态到DMA控制器。当传送中发生了冲突(在半双工模式下)，MAC控制器会标注在传输状态信息字中，然后接收并抛弃之后的数据，直到收到下一个SOF信号。在检测到来自MAC的(状态信息字中的)重试请求时，应当从SOF开始重发同样的帧。在传输时，如果不能及时连续地提供数据，MAC会标示一个数据下溢状态。在一个帧的传输过程中，如果MAC收到了SOF信号却没有收到EOF信号，MAC会忽略收到的SOF，并把下一个帧作为前一个帧的延续。

从发送FIFO弹出数据到MAC控制器的操作有两种模式：

- 在门限模式下，当FIFO中的数据达到了设置好的阈值时(或者在达到阈值之前写入了EOF)，数据会弹出FIFO并送入到MAC控制器中。这个阈值可以通过ETH_DMOPM寄存器的TXTC位来设置
- 在存储－转发模式下，只有当一个完整的帧写入FIFO之后，数据才会被送入MAC控制器。如果发送FIFO的长度小于要发送的以太网帧，那么在发送FIFO即将全满时，数据会被送入到MAC控制器

可以通过设置 FTXF 位(ETH_DMAOPM 寄存器的第 20 位)来清空发送 FIFO 中的内容。此位由硬件清零，并将 FIFO 的指针初始化至默认值。如果在帧数据传送到 MAC 控制器的过程中设置了 FTXF 位，FIFO 会被认为为空，传送停止。这将导致 MAC 发送器的数据下溢事件，并传送相应的状态信息字到 DMA 控制器

自动计算CRC和填充字节

如果提请传送的数据少于60(DA+SA+LT+数据)，控制器会添加一串0到帧中，以符合IEEE802.3 规范定义的最少数据长度为46字节的标准。当然也可以配置MAC控制器不自动添加填充字节。在帧发送时，循环冗余校验值(CRC)会自动添加到帧的校验序列(FCS)域。如果配置MAC控制器不在以太网帧末自动添加CRC值，计算好的CRC值不会被传送。有一种特例，即当MAC控制器配置为当帧(DA+SA+LT+数据)少于60字节，自动添加填充字节时，不管是否配置了自动添加CRC，都会添加CRC值到填充过的帧结尾。

CRC生成器会为以太网帧的FCS域计算32位的CRC值。计算时使用如下的多项式：

$G(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$

发送协议

MAC控制器管理以太网帧的发送。控制器实现了以下功能来符合IEEE802.3/802.3z规范：

- 生成前导符和SFD
- 在半双工模式下产生拥塞信号
- 控制啰嗦(Jabber)超时
- 控制半双工模式下的流控(背压)
- 产生所传输帧的状态信息
- 包含符合IEEE1588规范的时间戳逻辑

当需要发送一个新的帧，MAC 控制器会先发送前导符和 SFD，然后发送数据。前导是 7 个字节的”10101010”，SFD 是 1 个字节的”10101011”。碰撞窗口定义为 1 个间隙时间(对于 10/100M 位/s 的传输速率来说是 512 位的时间)。堵塞信号仅仅在半双工模式下才产生，在全双工模式下不会产生堵塞信号。

在MII模式下，如果在帧传送的任意时刻(从帧头到CRC的末位)发生了冲突，MAC控制器会发送一个0x5555 5555的32位堵塞信号，通知所有其他站点发生了冲突。如果冲突发生在传送前导符期间，MAC控制器会完成前导符和SFD域的发送，紧跟着发送堵塞信号。

啰嗦(Jabber)定时器用来在发送了超过2048字节(默认值)的数据时把以太网帧切断。MAC控制器在半双工模式下使用顺延(Deferral)机制的流控(背压)。当应用程序要求停止接收帧时，只要使能了传输流控，MAC控制器无论是否在接收帧，都会发送32字节的堵塞信号；这将导致传输冲突和远程站点的后撤。应用程序可以通过设置ETH_MACFCTRL寄存器的BPA位(位0)来使能传输流控。如果应用程序提出了发送帧的请求，控制器会马上安排并传送帧，无论是否发生了背压。需要注意的是，如果背压持续了一段时间(有超过16个的连续冲突事件发生)，远程站点会由于过多的冲突而放弃传输。如果在发送时使能了IEEE 1588时间戳功能，则在向MII总线传送SFD域时，会添加当时的系统时间。

传输调度

MAC负责调度帧在MII上的传输。MAC管理两个帧之间的间隔，并遵循半双工模式下的截断二进制指数退让算法。MAC会在符合FG和退让延迟的情况下启动传输。在两个传输的帧之间， MAC在一定时间内会保持空闲状态，这段时间就是配置好的帧间隔(通过ETH_MACCTRL寄存器的FG位)。如果一个帧早于配置好的FG时间到达，则MII需要等待来自MAC的使能信号，才能开始传输下一个帧。一旦MII的载波信号消失，MAC即启动它的FG计数器。全双工模式下，配置好的FG时间到时后，MAC则使能传输。在半双工模式下，如果FG时间配置为96个比特的时间，MAC将遵循IEEE 802.3规范的4.2.3.2.1章节定义的规则。如果在整个FG时间间隔的前三分之二(64个比特时间)的时间段里检测到了载波，MAC将复位FG计数器。如果在FG时间间隔的后三分之一的时间段里检测到了载波，MAC将继续FG计数，并在达到FG间隔时间后使能传输。在半双工模式下，MAC遵循截断二进制指数退让算法。

传输流控

如果设置了传输流控位(ETH_MACFCTRL寄存器的TFE位)，在全双工模式下，MAC会在需要的时候产生并传输Pause帧。Pause帧带有计算好的CRC值。可以有两种方法来启动产生Pause帧。

当应用程序把ETH_MACFCTRL寄存器的FCB位置’1’，或当接收FIFO全满(包缓存)时，会把Pause帧发送出去。

- 当设置ETH_MACFCTRL寄存器的FCB位为’1’来请求进行流控时，MAC会产生并传输一个单独的Pause帧。这个Pause帧指定的暂停时间为ETH_MACFCTRL寄存器中配置好的暂停时间值。如果想延长暂停时间，或者中止前一个Pause帧指示的暂停，需要重新配置寄存器中的暂停时间

(ETH_MACFCTRL寄存器的PTM域)，并请求传输一个新的Pause帧

- 如果使能了流控，当接收FIFO满时，MAC会产生并传输一个Pause帧。这个Pause帧指定的暂停时间为ETH_MACFCTRL寄存器中配置好的暂停时间值。如果在达到配置好的暂停时间极限(ETH_MACFCTRL寄存器的PLT域)时，接收FIFO仍然为满，MAC会再传输一个Pause帧。如果接收FIFO一直为满，则将一直重复这个过程。当接收FIFO不为满而暂停时间仍未到时，MAC会传输一个暂停时间为0的Pause帧，指示远程站点结束暂停，本地缓存区已经准备好接收新的数据帧

单包的传输过程

下面列出了传输过程中的一系列事件：

1. 如果有数据需要传输，DMA控制器通过AHB主接口从存储区读出数据，并转发到FIFO中。FIFO将持续接收数据，直到整个帧都传输完毕。
2. 当达到FIFO的阈值，或是整个数据包都已经存储在FIFO中，数据帧将从FIFO中弹出并送达MAC控制器。DMA将持续从FIFO中弹出数据直到整个数据包都被传送到MAC。在帧传输完毕之后，DMA控制器会收到MAC送出的状态信息。

传输过程——缓存区中有两个数据包

1. 由于DMA控制器必须先更新描述符的状态信息，再将描述符释放给应用程序(CPU)，因此在发送FIFO中至多只能有2个帧。只有在OPSF(Operation on second frame第二帧操作)位为’1’时，DMA才能读取内存里的第二帧数据，并把它们送进FIFO。如果该位不为’1’，那么只有在MAC发送完第一帧数据，且DMA释放描述符后，DMA才能读取第二帧的数据。
2. 如果OPSF位为’1’，DMA在把第一帧数据传输到FIFO后立刻开始取第二帧的数据，而不等待更新第一帧的描述符状态信息。与此同时，在MAC控制器发送第一帧数据的时候，FIFO接收第二帧数据。一旦MAC把第一帧数据发送完毕，就会传送更新的状态信息给 DMA控制器。如果DMA已经把第二帧的全部数据传送到FIFO，MAC必须等到更新完第一帧的状态信息才能发送下一帧数据。

遇到冲突时重新发送

半双工模式下，在帧发往MAC的时候有可能在MAC线路上发生冲突事件。这样MAC有可能在FIFO接收完这帧数据前就尝试重新发送。这时，如果重新发送开始，就需要再次把数据从FIFO中弹出。但是在FIFO控制器把96字节的数据弹出供MAC发送以后，就会释放这些数据的空间，允许DMA往那里推进新的数据。这意味着如果帧发出的数据数超过该门限(96字节)或者MAC控制器提示迟到冲突事件时，不能进行重新发送。

清空发送FIFO操作

MAC允许软件使用ETH_DMAOPM寄存器的FTXF位(位20)来清空发送FIFO。即使发送FIFO正在把一帧数据发送给MAC控制器，也会立即执行清空操作：清空发送FIFO，复位相应的指针为初始的状态。这个操作会导致MAC发送端发生数据下溢事件，中止发送当前帧，并在该帧的状态信息里同时标记数据下溢位和清空位(TDES0的位1和位13)。在执行清空操作时，发送FIFO不再从应用程序(即从DMA)处接收数据，并把发送状态信息字返回应用程序，通知受影响的帧的数目(包括只有部分被清空的帧)。对于完全被清空的帧，设置了帧清空标志位(TDES0的13位)。在应用程序(DMA)接收到全部被清空帧的状态信息字以后，清空操作完成。随后ETH_DMAOPM寄存器的FTXF位被清’0’。这时，允许FIFO从应用程序(DMA)接收新帧。清空操作后，FIFO会忽略所有不以SOF起始符开头的数据。

发送状态信息字

在以太网帧传输到MAC控制器，并且MAC控制器把该帧发送出去以后，会把发送状态信息返回应用程序。发送状态信息的具体含义与TDES0位[23:0]的描述一致。如果使能了IEEE 1588时间戳功能，会和发送状态信息一起返回一个特别的64位时间戳。

发送校验和模块

诸如TCP和UDP的通讯协议都包含校验和域，用来检测数据在网络上传输的完整性。以太网最广泛的应用是收发封装有TCP或者UDP数据的IP数据包，因此以太网控制器具有发送校验和的功能，支持计算校验和，并在发送时插入计算结果，以及在接收时侦测校验和错误。本节描述了发送帧的校验和模块的操作功能。

注意： *1. 计算 TCP、UDP 和 ICMP 的校验和都是针对整个帧的，然后插入相应的数据报头的校验和域。因此，*

只有在发送 FIFO 设置成存储-转发模式的时候(ETH_DMAOPM 寄存器的 TXSF 位为’1’)才能使能此功能。如果 MAC 控制器设置为门限模式，发送校验和模块的功能无效。

2. 在数据帧发送到MAC控制器发送端前，必须保证发送FIFO的深度足够容纳一个完整的帧。如果FIFO的深度小于输入的以太网帧长度，即便在存储-转发模式下，有效荷载数据(TCP/UDP/ICMP)的校验和功能无效，而只会计算和修改IPv4报头的校验和域。

发送校验和模块支持2种校验和计算和插入的方法。可以通过帧描述符的CIC位(TDES1的位28:27，请参考后续的“TDES1：发送描述符字1”)。

欲了解IPv4、TCP、UDP、ICMP、IPv6和ICMPv6报头的规范，请分别查阅IETF规范RFC791、RFC 793、RFC 768、RFC 792、RFC 2460和RFC 4443。

- IP头校验和

  在IPv4数据包里，16位的报头校验和域(IPv4数据包的第11和12字节)指示了报头域数据的 完整性。如果以太网帧的类型域值为0x0800并且IP数据包的版本域值为0x4，校验和模块就 检测到IPv4数据包。于是MAC不管输入帧的校验和域内容，直接以计算结果取而代之。IPv6的报头不包含校验和域，因此校验和模块不会改变IPv6报头的值，而是通过发送状态信息的IP报头错误状态标志位(TDES0位16)反映这个报头的校验和计算结果；在以太网类型 域以及IP报头版本域取值不匹配，或者以太网帧数据数目不够IP报头的数据长度域时，该标 志位被置为’1’。换而言之，该位在IP报头出现下列错误时被置为’1’：

  a) 对于IPv4数据包

  — 接收到的以太网类型域0x0800，但IP报头的版本域不等于0x4

  — IPv4报头长度域取值小于0x5(20字节)

  — 帧的总长度小于IPv4报头长度域的值

  b) 对于IPv6数据包

  — 接收到的以太网类型域为0x86DD，但IP报头的版本域不等于0x6

  — 帧在完整接收IPv6报头(40字节)或者扩展报头(扩展报头包含报头长度域)之前结束。即使校验和模块检测到这样的报头错误，如果以太网类型域提示载荷数据为IPv4类型，仍然会插入一个IPv4的报头校验和

- TCP/UDP/ICMP校验和

  TCP/UDP/ICMP校验和通过分析IPv4或IPv6报头(包括扩展报头)来判断封装的数据是TCP还是UDP或者ICMP。

注意： a) 对于非 TCP、UDP 或者 ICMP/ICMPv6 的有效数据，校验和功能无效，不改动帧的数据

b) 对于不完整的IP帧(IPv4或者IPv6)，包含安全功能的IP帧(如验证报头或者封装有安全数据)，以及带路由报头的IPv6帧，校验和功能无效。

校验和模块会对TCP、UDP或者ICMP的数据进行计算，并插入报头的相应域。它有以下2种工作模式 ：

— 第一种模式：校验和计算不包括TCP、UDP或者ICMPv6的伪报头，假设报头的校验和值就是输入帧的校验和域的数值。校验和域的内容会被包含在校验和的计算中，并最终被计算的结果取代

— 第二种模式：不管输入帧的校验和域的内容，校验和的计算包括TCP、UDP或者ICMPv6的伪报头数据，最终的计算结果将取代校验和域的原有内容

注意： 无论采用哪种模式，对于 IPv4 上的 ICMP 数据包，ICMP 数据包的校验和域内容必须 为 0x0000；该类数据包没有定义伪报头。如果数据包的校验和域不是 0x0000，可能会在数据包中插入不正确的校验和数值。

发送状态信息向量的数据校验和错误状态位(位12)标示了校验和操作的结果。如果以下任意一种情况发生，那么有效载荷数据校验和错误状态位置’1’：

— 在存储-转发模式下，在帧结尾没有写入FIFO的情况下，帧就被转发给MAC发射端

— 接收到的数据包少于IP报头数据长度域指示的字节数目

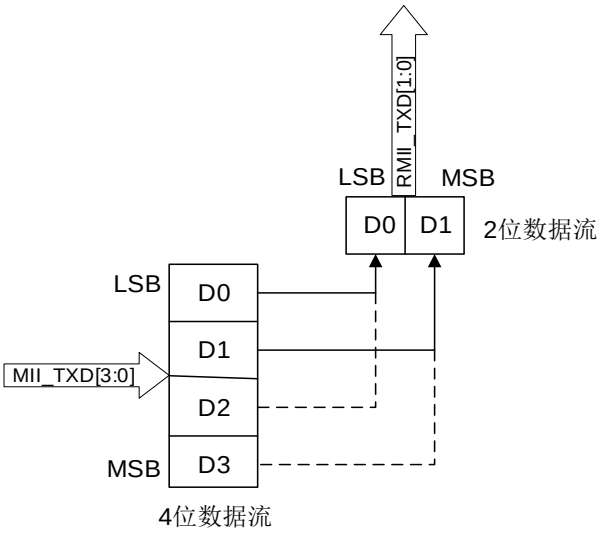
如果数据包长度大于给出的数据长度，多余的数据会被当成充填字节而丢弃，而不会报告错误。如果检测到第一类错误，则不改变TCP、UDP或者ICMP报头。如果检测到第二类错误，仍然会把校验和计

算结果插入报头的相应域。

MII/RMII位传输顺序

下图显示了MII上的4位数据，以2个2位数据的形式发送到RMII的顺序。先发送低2位(D0和D1)，再发送高2位(D2和D3)。

图 24-13 位传输顺序



MII/RMII传输时序图

图 24-14 无冲突时的传输

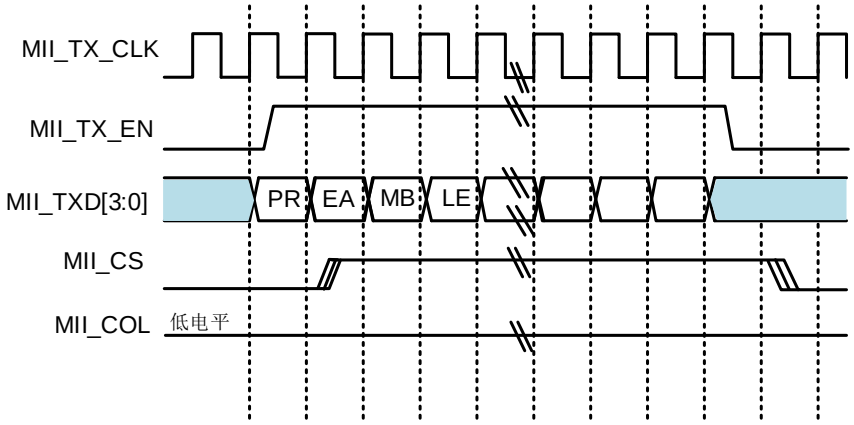
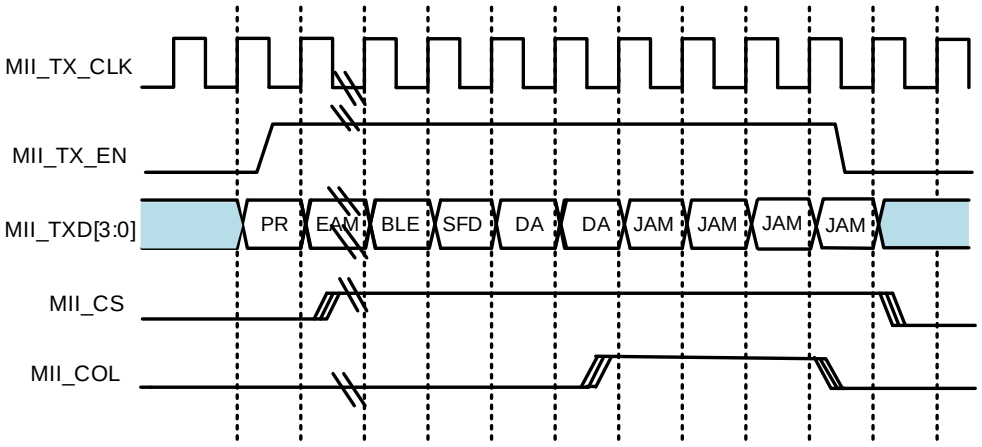
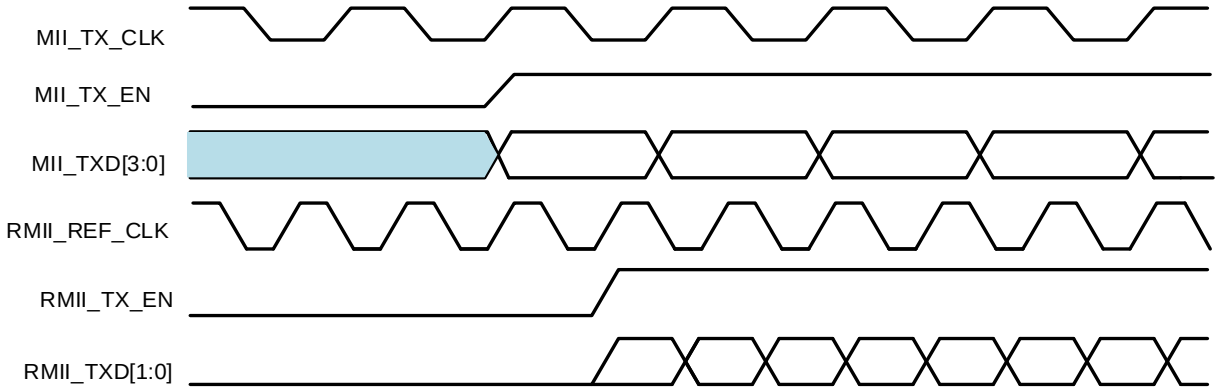


图 24-15 有冲突时的传输



下图显示在MII和RMII上帧的传输

图 24-16 MII和RMII模式下的帧传输



24.5.3 MAC帧的接收

MAC接收到的帧都会被送入接收FIFO中。一旦接收到的数据数目超过了预先设定的阈值(由ETH_DMAOPM寄存器的RXTC域设定)，控制器就会把FIFO的状态通知DMA，由DMA启动的预设传输，把数据通过AHB接口送出。

在直通模式(默认模式)下，如果FIFO接收到64字节(可通过ETH_DMAOPM寄存器的RXTC设置) 数据或者完整的帧，就开始从FIFO中弹出数据，并通知DMA接收。一旦DMA开始向AHB接口传送数据，就会持续从FIFO中弹出数据，直到完成整个数据包的传输。FIFO转发完EOF后，就会弹出接收状态信息字并将其发送给DMA控制器。

在接收FIFO存储-转发模式(通过ETH_DMAOPM寄存器的RSFD位设置)下，只有在接收FIFO完整地收到一个帧后，DMA才能将其读出。此模式下，如果MAC设置成丢弃所有错误帧，那么DMA只会读出合法的帧，并转发给应用程序。而在直通模式下，一部分错误的帧并没有被丢弃，这是因为在帧结尾才会接收到错误状态信息，而这时帧的起始部分已经被从FIFO中读出来了。

一旦MAC在MII上检测到SFD就会启动接收过程。MAC控制器在处理帧之前会剥离前导符和SFD。会通过过滤器检查帧的报头，并用FCS域核对帧的CRC值。如果帧没能通过地址滤波器，MAC控制器就会丢弃该帧。

接收协议

MAC会剥离接收到帧的前导符和SFD。MAC一旦检测到SFD，就从SFD后的第一个字节(目的地址)为开始向FIFO发送以太网帧数据。如果使能了IEEE1588时间戳，会在检测到任意帧的SFD的时候记录下系统的当前时间。除非这个帧没有通过地址过滤器而被丢弃，MAC还会把这个时间戳转交给应用程序。

如果接收到的”帧长度/类型”域的值小于0x600，并且把MAC配置成CRC/充填自动剥离，那么MAC会在向接收FIFO发送完”帧长度/类型”域规定数目的帧数据后，开始丢弃剩下的字节(包括FCS域)。如果长度/类型域的值大于或等于0x600，不管自动CRC剥离选项怎么设置，MAC都会把所有接收到的以太网帧数据发送到接收FIFO。默认地，MAC打开看门狗定时器，即会切断长度大于2048字节的帧(DA + SA + LT + 数据 + 填充 + FCS)。可以通过MAC配置寄存器的看门狗失能位(WD)来关闭这项功能。然而，即使关闭了看门狗定时器，MAC仍然会切断长度大于16KB的帧，并报告一个看门狗超时事件。

接收CRC：自动CRC和填充剥离

MAC会检查接收到帧的CRC错误。MAC会计算从接收到帧目的地址域到FCS域的32位CRC值。计算使用下面的多项式：

$G(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$

无论如何设置”自动充填/CRC剥离” 选项，MAC都会把完整的帧接收下来，并计算收到帧的CRC。

接收校验和模块

不管是IPv4还是IPv6帧，都需要检测接收到的以太网帧数据的完整性。设置ETH_MACCTRL寄存器的IP4CO位，可以使能接收校验和模块。MAC检验以太网帧的类型域是0x0800还是0x86DD，来判别是IPv4帧还是IPv6帧。这个方法也用于带VLAN标签的帧识别。接收校验和模块会计算IPv4报头的校验和，检查它是否与IPv4报头的校验和域的内容匹配。在以太网类型域值指示的数据类型与IP报头版本域不匹配，或者接收到的帧长少于IPv4报头长度域指示的长度，或者IPv4或IPv6报头少于20字节时，设置IP报头错

误位为’1’。接收校验和模块还能识别IP数据包的数据类型是TCP、UDP还是ICMP，并按照TCP、UDP或ICMP的规范计算它们的校验和。计算包括了TCP/UDP/ICMPv6伪报头的数据，接收到的校验和数据与计算的结果比较，比较的结果反映在接收状态信息字的数据校验和错误位；该位也会在收到的TCP、UDP或者ICMP数据长度与IP报头给出的长度不符时置’1’。正如前述”TCP/UDP/ICMP校验和”段落描述的那样，接收校验和模块不计算下列情况：不完整的IP数据包、带安全功能的IP数据包、IPv6路由报头以及数据类型不是TCP、UDP或者ICMP的数据包。如后续的”RDES0：接收描述符字0”段落所述，接收状态信息标示了校验和功能是否生效。MAC控制器不会对接收到的以太网帧添加任何校验字节。

如后续的”RDES0：接收描述符字0”段落所述，寄存器位的含义示于下表：

表 24-6 帧的状态

位 **18**：以太网帧	位 **27**：报头校验和错误	位 **28**：校验和错误	帧状态信息
0	0	0	该帧是一个 802.3 帧(长度域值小于 0x0600)
1	0	0	IPv4/IPv6 帧，未发现校验和错误
1	0	1	IPv4/IPv6 帧，发现校验和错误(PCE 位为 1)
1	1	0	IPv4/IPv6 帧，发现 IP 报头校验和错误(IP4CO HCE 位)
1	1	1	IPv4/IPv6 帧，PCE 位和 IP4CO HCE 位都为’1’
0	0	1	IPv4/IPv6 帧，不带 IPHCE 位，不支持这种类型，检验功能无效
0	1	1	帧类型既不是 IPv4 也不是 IPv6，校验和模块功能失效
0	1	0	保留

接收帧控制器

如果MAC FRMF过滤器寄存器的RALL位为’0’，MAC会根据帧的目的/源地址进行过滤(如果不希望接收诸如过短帧、CRC错误等坏帧，则需要进行另一个层次的过滤)。所有不能通过过滤器的帧，会被丢弃而不会被转给应用程序。如果在接收过程中动态地修改了过滤器参数，并且发生未通过目的地址/源地址过滤器的情况，那么该帧的其余部分会被丢弃，同时更新接收状态信息字(帧长度标志为0，CRC错误标志位，帧过短错误标志位置’1’)，表示帧未通过过滤器。在以太网掉电模式下，丢弃所有帧，而不转发给应用程序。

接收流控

仅在全双工模式下，MAC能够检测PAUSE帧，并按照PAUSE帧中的参数，在一定时间内暂停 发送数据。设置ETH_MACFCTRL寄存器的RFCEN位，可以使能或者取消PAUSE帧检测功能。一旦使能接收流控功能，就会监视所有接收到帧的目的地址，看该地址是否等于控制帧的多播地址(0x0180 C200 0001；如果相符(收到帧的目的地址匹配预留的控制帧目的地址)，MAC会根据ETH_MACFRMF寄存器的PCF位的值，来决定是否将接收到的控制帧转发给应用程序。

MAC也会对接收到帧的类型域、操作数域和PAUSE时间域进行解码。如果状态信息指示帧长为64字节，并且没有侦测到CRC错误，那么MAC发送端会暂停一段时间发送数据，这段时间的长度为得到的PAUSE时间值乘以间隙时间(对10M位/s和100M位/s模式，间隙时间都是64字节)。

在暂停期间，如果接收到另一个PAUSE时间为0的PAUSE帧，MAC会清除PAUSE的时间，重新开始发送数据。

如果接收到的控制帧，类型域不匹配(0x8808)、操作数不匹配(0x00001)、帧长度不符(64字节)、或者检测到CRC错误，那么MAC不会暂停发送数据。

如果PAUSE帧的目的地址是多播地址，那么MAC基于地址匹配对帧进行过滤。如果PAUSE帧的目的地址是单播地址，那么MAC对帧的过滤就是检查目的地址是否匹配MAC地

址0寄存器，同时检查ETH_MACFCTRL寄存器的PCF位是否为’1’(检测目的地址是单点发送地址的PAUSE帧)。PCF位(ETH_MACFRMF的位[7:6])的功能是在地址过滤的基础上进一步控制对控制帧的过滤。

多个帧的接收处理

由于帧的状态信息紧随在帧数据之后。因此只要FIFO未满，就可以存放任意数量的帧。

错误处理

如果在从MAC接收到EOF之前，接收FIFO就满了，会报告溢出错误并丢弃整个帧，同时溢出计数器(位于ETH_DMAMFBOCNT寄存器)加1。状态信息会指出由于溢出，造成某个帧不完整。根据ETH_DMAOPM寄存器的FEF和FUGF位的设置，接收FIFO可以过滤掉错误帧和长度低于最小帧长的帧。

如果接收FIFO设置成存储-转发模式，那么可以过滤并丢弃所有的错误帧。在接收FIFO设置成直通模式时，如果DMA从接收FIFO读出帧的SOF时，FIFO已经给出该帧的状态信息和长度，那么可以丢弃掉整个出错的帧。设置接收帧清空位为’1’，DMA可以清空正从FIFO中读出的错误帧，这时FIFO中止向应用程序(DMA)弹出数据，帧余下的部分会从内部读出 并丢弃。随后就可以开始下一帧的传输。

接收状态信息字

在接收以太网帧结束时，MAC会把接收状态信息发给应用程序(DMA)。接收状态信息的具体含义同RDES0的位[31:0]描述，参见后续的”RDES0：接收描述符字0”段落。

帧长度接口

对于交换机的应用，应用程序与MAC之间数据的收发都是以完整的帧为单位的。应用层软件需要掌握从输入端口接收到的帧的长度，才能把帧转发给输出端口。MAC控制器在接收帧结束时生成的接收状态信息包含了每一个接收到帧的长度。

注意： *帧长度值为0 意味着由于接收FIFO 溢出，写入 FIFO 的帧不完整。*

MII/RMII位接收顺序

下图显示了，从RMII上2个2位的数据转化为传送到MII上的4位数据接收顺序。先接收低2位(D0和D1)，再接收高2位(D2和D3)。

图 24-17 接收位顺序

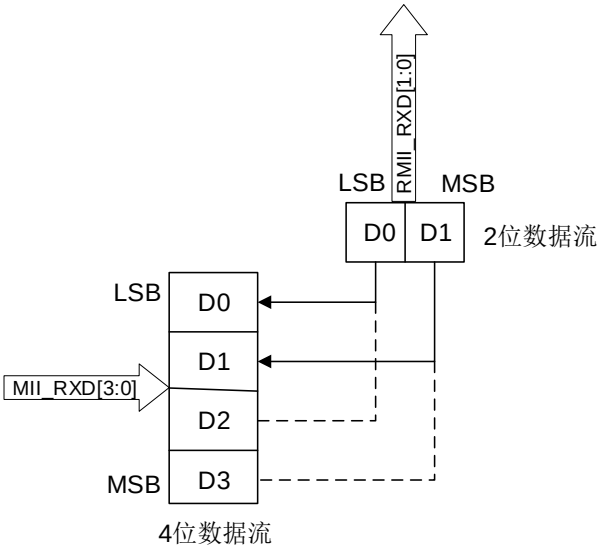


图 24-18 接收时无错误

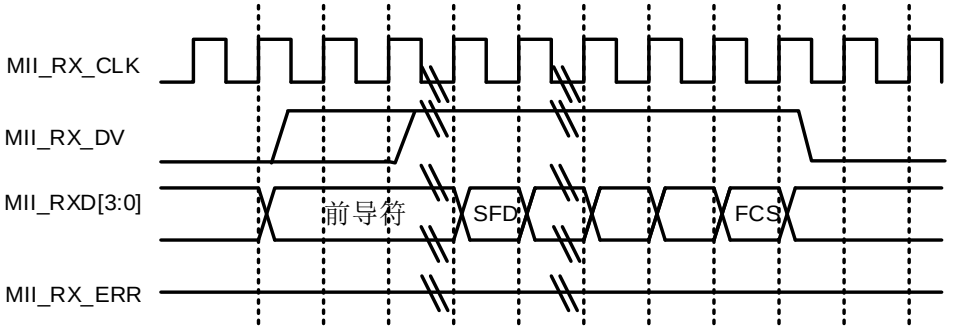


图 24-19 接收时有错误

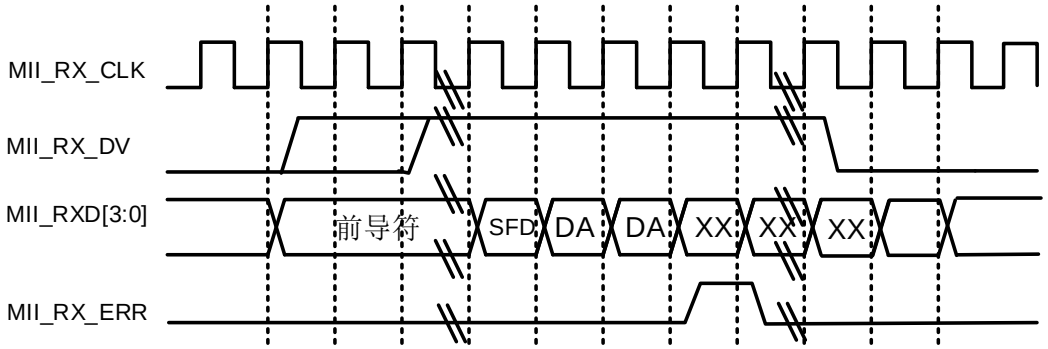
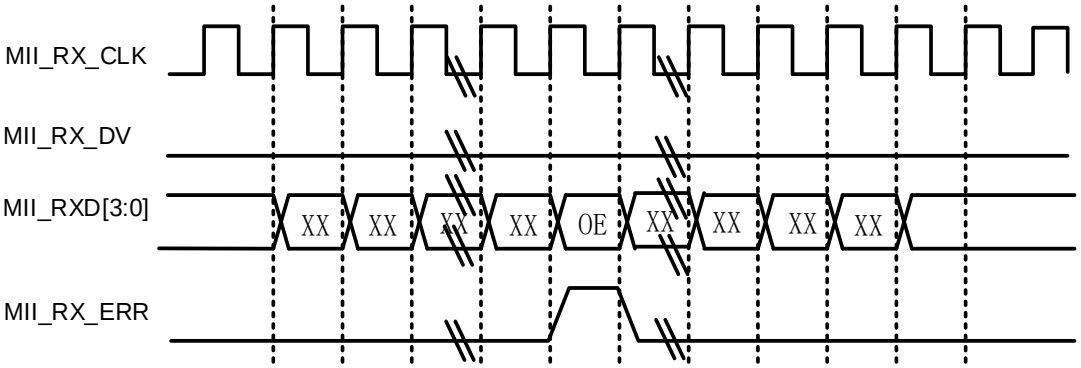


图 24-20 接收时有假载波指示



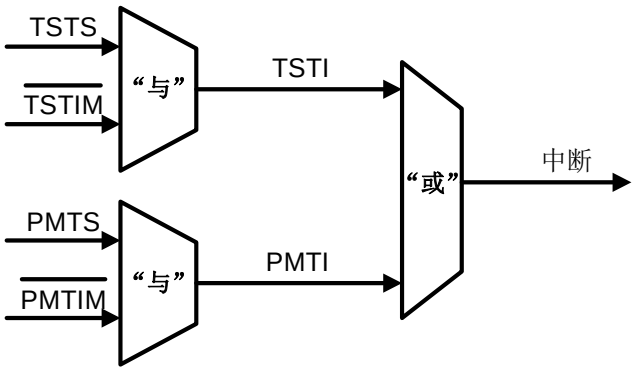
24.5.4 MAC中断

MAC控制器可由于多种事件的发生而产生中断。

ETH_MACISTS寄存器描述了所有可以使MAC控制器产生中断的事件。用户可以通过在中断屏蔽寄存器里把相应的屏蔽位置’1’来防止某一事件引发中断。

中断寄存器的位只表示中断由哪个模块引发。用户需要读相应的状态寄存器位及其他寄存器来清除中断。例如，中断寄存器的位3为’1’，表示MAC在掉电模式下接收到Magic数据包或者远程局域网唤醒帧。必须读ETH_MACPMTCTRLSTS寄存器来清除这个中断事件。

图 24-21 MAC控制器中断屏蔽示意图



24.5.5 MAC过滤

地址过滤

地址过滤机制检查所有接收到帧的目的地址和源地址，并报告相应的地址过滤结果。地址检查根据应用程序设定的参数(帧过滤器寄存器)进行。帧经过过滤还可以识别是多播帧或者广播帧。地址过滤利用静态物理地址(MAC地址)和多播HASH列表来进行地址检验。

单播目的地址滤波器

MAC支持多达4个MAC地址进行单播过滤，如果选择这种方式(帧过滤寄存器的HU位为’0’)， MAC会把接收到帧的48位单播地址与设好的MAC地址逐位比较，检查是否相符。默认情况下，始终使能

MacAddr0。其他几个地址MacAddr1-MacAddr3分别有对应的使能位。在与接收到帧的目的地址时，这几个地址(MacAddr1-MacAddr3)的每一个字节都可以通过设置寄存器的屏蔽字节控制位，来设置不与接收到帧目的地址的相应字节比较。利用这个功能就可以实现帧目的地址群过滤。在HASH过滤模式下(HU位为’1’)，MAC利用64位的HASH列表对单播地址进行不完美过滤。HASH过滤时，MAC计算接收到帧的目的地址的CRC值(见下面的注释)，并取高6位作为索引检索HASH列表。CRC值为’000000’对应HASH列表寄存器的位0，CRC值为’111111’对应HASH列表寄存器的位63。如果CRC值对应的HASH列表上的相应位为’1’，说明该帧能通过HASH过滤器，否则该帧不能通过HASH过滤器。

注释： CRC 是指根据以下多项式计算出的 32 位 CRC 值，更多细节请参考 24.5.3 节。

$G(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$

多播目的地址滤波器

设置帧过滤寄存器的PALLM位为’1’，可以配置MAC接收所有的多播帧。如果PALLM位为’0’，MAC根据帧过滤寄存器HM位的取值对多播地址进行过滤。在完美过滤模式下，把多播地址与设置好的

MAC目的地址寄存器(1-3)比较；完美过滤也支持群地址过滤。在HASH过滤模式下，利用64位HASH列表进行不完美过滤；MAC使用接收到多播地址CRC值(见下面的注释)的高6位检索 HASH列表。CRC值为’000000’对应HASH列表寄存器的位0，CRC值为’111111’对应HASH列表寄存器的位63。如果CRC值对应的HASH列表上的相应位为’1’，说明该帧能通过HASH过滤器，否则该帧不能通过HASH过滤器。

注释： CRC是指根据以下多项式计算出的32位CRC值，更多细节请参考24.5.3节。

$G(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$

HASH或者完美地址滤波器

通过设置帧过滤器寄存器的HOPF位为’1’并设置相应的HU位(对单播帧)或者HM位(对多播帧)，可 以把目的地址(DA)过滤器配置成在只要帧的目的地址匹配HASH滤波器或者完美滤波器之一，就令帧通过。这种设置即对单播帧生效，也对多播帧生效。如果HOPF位为’0’，那么只有一种过滤器(HASH过滤器或者完美过滤器)会对接收到的帧起作用。

广播地址滤波器

MAC默认不过滤任何广播帧。但是，如果设置帧过滤寄存器的BFD位为’1’，MAC将拒收所有的 广播帧并丢弃接收到的广播帧。

单播源地址过滤器

MAC也可以根据接收到帧的源地址进行完美过滤。MAC默认把帧的源地址(SA)域和源地址寄存器的设定值进行比较。设置MAC地址寄存器[1:3]的位30为’1’，则过滤器在过滤时，不再比较帧的目的地址而是比较其源地址。MAC也支持对源地址的成组过滤。如果帧过滤寄存器的SADDRF位为’1’，MAC会丢弃没能通过源地址(SA)过滤器的帧；否则源地址(SA)过滤的结果会通过接收状态信息字的一个标志位反映出来(参见后续的”RDES0：接收描述符字0”段)。

在SADDRF位为’1’时，以源地址滤波器结果和目的地址滤波器结果逻辑“与”的形式判定帧是否转发给 应用程序。这意味着，只要帧没能通过其中一个过滤器，就会被丢弃。MAC只会把通过全部过滤器的帧转发给应用程序。

颠倒过滤操作

无论是目的地址过滤还是源地址过滤，都能在过滤器输出端颠倒过滤结果。颠倒可以分别通过设置帧过滤寄存器的DADDRIF位和SADDRIF位实现。DADDRIF位作用于单播和多播帧的目的地址，在该位 为’1’时，颠倒单播/多播帧的目的地址过滤器结果。同样的，在SADDRIF位为’1’时，颠倒单播帧的(源地址)过滤器结果。下面2个表总结了目的地址和源地址过滤器在不同设置下，接收到不同种类帧时的结果。

表 24-7 目的地址过滤器结果列表

帧类型	PM	HOPF	HU	DADDRIF	HM	PALLM	DB	目的地址(DA)过滤器操作
广播帧	1	X	X	X	X	X	X	通过
	0	X	X	X	X	X	0	通过
	0	X	X	X	X	X	1	不通过
单播帧	1	X	X	X	X	X	X	所有帧通过

	0	X	0	0	X	X	X	完美/组滤波器匹配时通过
	0	X	0	1	X	X	X	完美/组滤波器匹配时不通过
	0	0	1	0	X	X	X	HASH 滤波器匹配时通过
	0	0	1	1	X	X	X	HASH 滤波器匹配时不通过
	0	1	1	0	X	X	X	HASH 或者完美/组滤波器匹配时通过
	0	1	1	1	X	X	X	HASH 或者完美/组滤波器匹配时不通过
多播帧	1	X	X	X	X	X	X	所有帧通过
	X	X	X	X	X	1	X	所有帧通过
	0	X	X	0	0	0	X	完美/ 组滤波器匹配时通过，如果 PCF=0x，丢弃 PAUSE 控制帧
	0	0	X	0	1	0	X	HASH 滤波器匹配时通过，如果 PCF = 0x，丢弃 PAUSE 控制帧
	0	1	X	0	1	0	X	HASH 或者完美/组滤波器匹配时通过，如果 PCF = 0x，丢弃 PAUSE 控制帧
	0	X	X	1	0	0	X	完美/组滤波器匹配时不通过，如果 PCF = 0x，丢弃 PAUSE 控制帧
	0	0	X	1	1	0	X	HASH 滤波器匹配时不通过，如果 PCF = 0x，丢弃 PAUSE 控制帧
	0	1	X	1	1	0	X	HASH 或者完美/组滤波器匹配时不通过，如果 PCF = 0x，丢弃 PAUSE 控制帧

表 24-8 源地址过滤器结果列表

帧类型	RTPR	SADDRIF	SADDRF	源(SA)过滤器操作
单播帧	1	X	X	所有帧通过
	0	0	0	传送完美/组滤波器匹配状态，但不丢弃不通过的帧
	0	1	0	传送完美/组滤波器不匹配状态，但不丢弃帧
	0	0	1	完美/组滤波器匹配时通过，丢弃不通过的帧
	0	1	1	完美/组滤波器匹配时不通过，丢弃不通过的帧

24.5.6 MAC自循环模式

MAC支持自循环模式：发射端把帧发送到自己的接收端上。MAC的自循环模式默认是关闭的，可以通过把MAC的ETH_MACCTRL寄存器的Loopback位置'1'来打开这个功能。

24.5.7 MAC管理计数器：MMC

MAC管理计数器(MMC)是一整套收集发送和接收到帧的统计信息寄存器。这些寄存器包括：1个操控其他寄存器行为的控制寄存器，2个包含生成的中断(接收和发送)的32位寄存器和2个包含中断屏蔽的32位寄存器。应用程序可以访问这些寄存器，每个寄存器长度都是32位。

第24.8节"以太网寄存器描述"说明了这些寄存器的功能，并列出了每个统计信息计数器的地址。使用这个地址可以对需要的发送/接收计数器进行读/写访问。

MAC按照通过地址过滤器的帧数目更新接收MMC计数器，而不会更新被丢弃帧的统计信息，除非被丢弃的帧是长度小于6字节的过短帧(没有完整接收到目的地址)。

"好"的发送和接收帧

发送帧如果被成功地发送出去，就会被认为是"好"的。换而言之，如果一个帧的发送过程没有因为以下错误而中止，就可以被认为是"好"的

— 啰嗦(Jabber)超时

- 没有载波/载波丢失
- 迟到冲突
- 帧溢出
- 顺延(Deferral)过多
- 冲突过多

如果不发生下列错误，一个接收帧就可以被认为是“好”的：

- CRC错误
- 过短帧(少于64字节)
- 对齐错误(仅对10/100M位/s)
- 长度错误(仅对无类型帧)
- 超出范围(仅对无类型帧，帧长超过上限)
- MII_RXER输入错误
- 一个帧的最大帧长由帧的类型决定
- 无标签帧，最大帧长 = 1518
- VLAN帧，最大帧长 = 1522

24.5.8 电源管理：PMT

本节描述了MAC支持的电源管理(PMT)机制。PMT支持接收(远程)网络唤醒帧和Magic Packet 帧。PMT能在MAC接收到唤醒帧和Magic Packets时产生中断。设置远程唤醒帧使能位和Magic Packet使能位即可使能PMT模块，这2个使能位位于ETH_MACPMTCTRLSTS寄存器。在使能了PMT的掉电模式时，MAC会丢弃所有的帧，而不把它们转发给应用程序。只有在接收到Magic Packet或者远程唤醒帧，并且使能了相应的检测功能时，MAC才会退出掉电模式。

远程唤醒帧过滤器寄存器

唤醒帧寄存器一共有8个，需要逐一配置帧过滤器寄存器。连续写8次唤醒帧过滤器寄存器，就可以把需要的值分别写入唤醒帧过滤器寄存器1-8。对这些寄存器的读操作流程和写操作流程一致，需要连续读8次唤醒帧过滤器寄存器，才能读出唤醒帧过滤器寄存器1-8的值。

图 24-22 唤醒帧过滤寄存器

唤醒帧过滤器寄存器0	过滤器0字节屏蔽							
唤醒帧过滤器寄存器1	过滤器1字节屏蔽							
唤醒帧过滤器寄存器2	过滤器2字节屏蔽							
唤醒帧过滤器寄存器3	过滤器3字节屏蔽							
唤醒帧过滤器寄存器4	保留	过滤器3命令	保留	过滤器2命令	保留	过滤器1命令	保留	过滤器0命令
唤醒帧过滤器寄存器5	过滤器3偏移		过滤器2偏移		过滤器1偏移		过滤器0偏移	
唤醒帧过滤器寄存器6	过滤器1 CRC-16				过滤器0 CRC-16			
唤醒帧过滤器寄存器7	过滤器3 CRC-16				过滤器2 CRC-16			

- 过滤器i字节屏蔽

  该寄存器定义了在判断是否为唤醒帧时，使用过滤器 i(i=0~3)的哪些字节检查帧。第 31 位必须为’0’；位 j[30:0]是字节屏蔽位，如果位 j 为’1’，则 CRC 模块会处理输入帧的第[过滤器 i 偏移+ j]字节，否则忽略之

- 过滤器i命令

  4 位命令控制了过滤器 i 的工作。位 3 选择地址类型，定义了过滤器的检测模板对哪一类目的地址有效；如果该位为’1’，则检测模板只对多播地址有效；如果该位为’0’，那么检测模只对单播地址有效。位 2 和位 1 是保留位。位 0 是过滤器的使能位，如果为’0’，则不使用该过滤器

- 过滤器i偏移

  该寄存器定义了过滤器 i 要检查的字节在帧内的起始偏移。这个 8 位的过滤器偏移是指过滤器要检查的第一个字节对帧首的偏移量。最小允许取值是 12，代表了帧的第 13 个字节(偏移值 为 0 表示帧的第 1 个字节)

- 过滤器i CRC-16

  该寄存器包含了过滤器计算出的 CRC_16 值，也包含了唤醒帧过滤器寄存器模块预先写好的 字节屏蔽值

远程唤醒帧检测

在MAC处于睡眠模式，且ETH_MACPMTCTRLSTS寄存器的远程唤醒使能位为’1’时，在接收到远程唤醒帧后，MAC即恢复正常工作。对唤醒帧过滤器寄存器的连续写操作，可以设置所有8个唤醒过滤器寄存器。设置ETH_MACPMTCTRLSTS寄存器的位2为’1’，使能远程唤醒。PMT支持4个可编程的过滤器，可以提供不同的接收帧检测模板。如果输入帧通过了过滤器命令的地址过滤，而且过滤器CRC_16与被检查的输入帧匹配，则认为接收到唤醒帧。过滤器偏移(最小允许取值是12，代表了帧的第13个字节)定义了从帧的哪个字节开始检查。过滤器字节屏蔽定义了帧的哪些字节会被检查。字节屏蔽的位31必须为’0’。PMT只会检查唤醒帧是否有长度错误、FCS错误、 Dribble位错误、MII错误、冲突，并确保唤醒帧不是过短帧。即便唤醒帧的长度超过了512字 节，只要该帧有正确的CRC值，它仍然被认为是有效的。唤醒帧检测在接收到远程唤醒帧时都 会更新ETH_MACPMTCTRLSTS寄存器的相应标志位。如果使能了远程唤醒中断，那么PMT在接收 到远程唤醒帧时会产生中断。

Magic Packet检测

Magic Packet帧是指使用AMD公司的Magic Packet来启动网络上的睡眠设备的技术。MAC接收到包含特殊信息的数据包，这些数据包发向网络上的节点，称为Magic Packet。MAC只会检查那些发往设备地址或者广播地址的Magic Packet，来判断它们是否满足唤醒设备的条件。那些通过地址过滤的Magic Packet，MAC会进一步检查它们是否符合远程Wake-on-LAN的数据格式：6字节全1，紧接着重复16次的MAC地址。设置ETH_MACPMTCTRLSTS寄存器的位1，可以使能Magic Packet唤醒。PMT模块会持续监视每一个发向本节点的帧，检测其是否符合Magic Packet的格式，即：检查0xFFFF FFFF FFFF，接着是目的和源地址域；随后PMT模块检查帧是否在没有任何中断和暂停的情况下有16个重复的MAC地址；如果这16次重复间有任何的间断，则需要重新在输入帧里检测0xFFFF FFFF FFFF。这16次重复的MAC地址可以位于帧的任何位置，但必须位于同步流(0xFFFF FFFF FFFF)之后。只要能够检测到16个重复的MAC地址，设备也接受多播帧作为Magic Packet帧。例如，某设备的MAC地址是0x0011 2233 4455，那么MAC会检测以下数据序列：

目的地址 源地址 ……FFFF FFFF FFFF

0011 2233 4455 0011 2233 4455 0011 2233 4455 0011 2233 4455

0011 2233 4455 0011 2233 4455 0011 2233 4455 0011 2233 4455

0011 2233 4455 0011 2233 4455 0011 2233 4455 0011 2233 4455

0011 2233 4455 0011 2233 4455 0011 2233 4455 0011 2233 4455

… CRC

Magic Packet检测在接收到Magic Packet时会更新ETH_MACPMTCTRLSTS寄存器的相应标志位。如果使能了Magic Packet中断，则PMT在接收到Magic Packet时会产生中断。

系统在掉电期间注意事项

在MCU处于停机模式时，使能外部中断线19，以太网的PMT模块仍能够检测帧。 因为MAC的接收状态机参与了Magic Packet/局域网唤醒帧检测，因此需要在掉电模式下维持它的工作，即ETH_MACCTRL寄存器的RE位需要保持为’1’。在掉电时需要把ETH_MACCTRL寄存器的TE位清’0’来关闭发送状态机。此外，由于不需要把Magic Packet/局域网唤醒帧转发到SRAM 中，因此在掉电时也要关闭DMA。设置ETH_DMAOPM寄存器的STXE位和SRXE位(分别对应发送DMA和接收DMA)为’0’来关闭以太网

DMA。

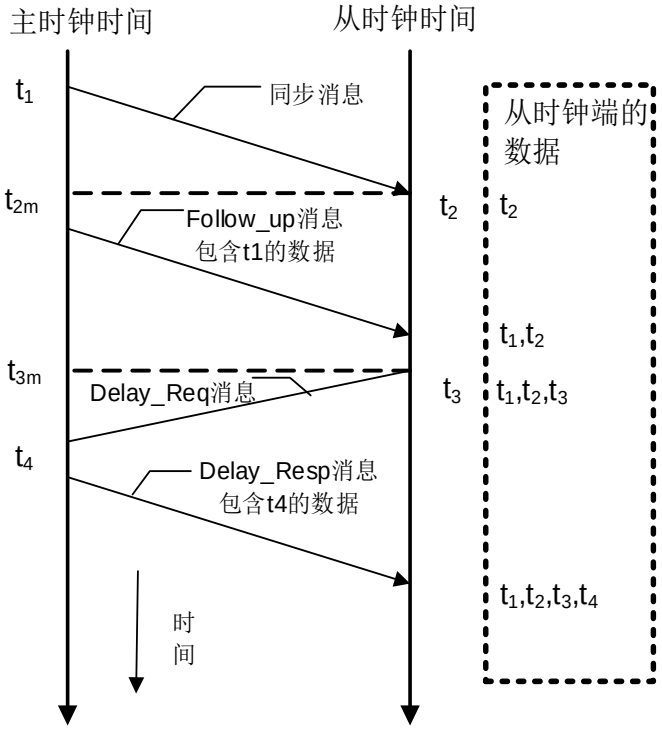
推荐的掉电和唤醒顺序如下：

1. 关闭发送DMA并等待之前所有的帧发送完毕。可以通过观察ETH_DMASTS寄存器位0来检查发送是否完成。
2. 把ETH_MACCTRL寄存器的TE位和RE位清'0'来关闭MAC发射器和MAC接收器。
3. 等待接收DMA把接收FIFO里的所有帧读出。
4. 关闭接收DMA。
5. 配置并使能外部中断线19，使其或者能产生事件或者能产生中断。
6. 如果配置了外部中断线19产生中断，则还需要编写中断处理程序ETH_WKUP_IRQ，在其中清除外部中断线19的中断待处理标志位。
7. 设置ETH_MACPMTCTRLSTS寄存器的MFE/WFEN位为'1'，使能Magic Packet/局域网唤醒帧检测。
8. 设置ETH_MACPMTCTRLSTS寄存器的PD位为'1'，使能掉电模式。
9. 设置ETH_MACCTRL寄存器的RE位为'1'，打开MAC接收器。
10. MCU进入停机(STOP)模式，更多细节请参阅2.3.2.2：停止模式。
11. 在接收到有效的唤醒帧后，以太网模块退出掉电模式。
12. 读取ETH_MACPMTCTRLSTS寄存器来清除电源管理事件标志位，打开MAC发送状态机，以及发送和接收DMA。
13. 设置系统时钟：使能HSE并配置时钟参数。

24.5.9 精确时间协议(IEEE1588 PTP)

IEEE1588标准定义了一种协议，该协议允许通过网络通讯，本地计算和分离设备技术实现测量和控制系统精确的时钟同步。协议可以应用于通过本地局域网通讯的系统，网络要求支持多播通信，这个局域网可以是(但不限于)以太网。协议可以用来同步拥有不同精度、分辨率和稳定度时钟的系统。协议支持在占用最少网络和计算资源的情况下，系统时钟同步精度达到亚微秒级。协议名称是PTP(精确时间协议)，协议的基础是以UDP/IP为载体的消息传递。整个系统或者网络的各个节点，按照发起和接收时间/时钟信息的角色，可以分为主节点和从节点。协议通过在主从节点之间交换PTP消息来把从节点时钟同步到主节点时钟，该技术如下图。

图 24-23 网络时钟同步



1. 主节点向所有的从节点广播PTP Sync消息。Sync消息包含主节点的时间信息。该消息从主节点发出的时间记为t1。对于以太网端口，应当在MII获取这个时间。
2. 从节点接收到Sync消息，并根据自己的时钟，记录下接收到Sync消息的时间t2。
3. 主节点随后向从节点发送follow_up消息，包含t1的信息，以备后用。
4. 从节点向主节点发送Delay_Req消息，并记录下该消息从MII发出的时间t3。
5. 主节点接收到Delay_Req消息，并记录下接收到的时间t4。
6. 主节点向从节点发送Delay_Resp消息，消息包含了t4的信息。
7. 从节点利用t1、t2、t3和t4，把它的本地时钟与主节点的时钟同步

协议的大部分是通过UDP层之上的软件实现的。不过，由上文所述，需要硬件支持记录PTP包从以太网端口MII发出和收到的准确时间。硬件需要记下该时间信息并将其返回给软件，这样才能正确，高精度地实现PTP时间同步。

基准时钟源

IEEE1588协议规定，系统需要64位格式的基准时间来获得当前时间记录，这64位被分为2个32位通道，其中高32位提供以秒为单位的时间信息，低32位提供以纳秒为单位的时间信息。

PTP基准时钟输入用来在内部生成基准时间(也称为系统时间)和记录时间戳。这个基准时钟的频率必须大于或等于时间戳计数器的分辨率。主节点和从节点之间的时间同步精度在100ns左右。

后续的“系统时间校准方法**系统时间校准方法系统时间校准方法系统时间校准方法**”段描述了生成、更新和更改系统时间的方法。

时间同步的精度取决于PTP基准时钟输入的周期，所用晶体振荡器的特性(频漂)和同步流程的执行频度。

由于发送和接收时钟输入域需要和PTP基准时钟域同步，因此时间戳记录的误差是1个基准时钟周期。如果我们考虑由时间戳计数器分辨率引起的误差，需要再加上半个周期的误差。

发送带PTP功能的帧

在帧的SFD从MII输出的时候，就会记录下时间戳。可以以帧为单位选择是否需要记录某个帧的时间邮戳。即每一个等待发送的帧都有一个标志，指示是否需要记录这个帧的时间戳。MAC并不通过处理帧本身的方式来识别PTP帧，而是通过帧发送描述符里的控制位来控制(见图24-31)。记录下来的时间戳会按照与

帧的发送状态信息相同的方式，返回给应用程序。时间戳是和帧的发送状态信息一起，存放在相应的发送描述符里送回的，这样就自动地把时间戳和PTP帧联系起来。64位的时间戳分别写入TDES2和TDES3域，TDES2含有时间戳的低32位数据，具体描述见后续的“带IEEE1588时间戳的发送DMA描述符格式”。

接收带PTP功能的帧

在使能了IEEE1588时间戳功能时，MAC会在MII端记录下所有收到帧的时间戳。MAC不通过处理帧本身来识别PTP帧。一旦接收完毕一帧，MAC就会提供帧的时间戳。记录下来的时间戳会按照与帧的接收状态信息相同的方式，返回给应用程序。时间戳是和帧的接收状态信息一起，存放在相应的发送描述符里送回的。64位的时间戳分别写入RDES2和RDES3域，RDES2含有时间戳的低32位数据，具体描述见后续的“带IEEE1588时间戳的接收DMA描述符格式”。

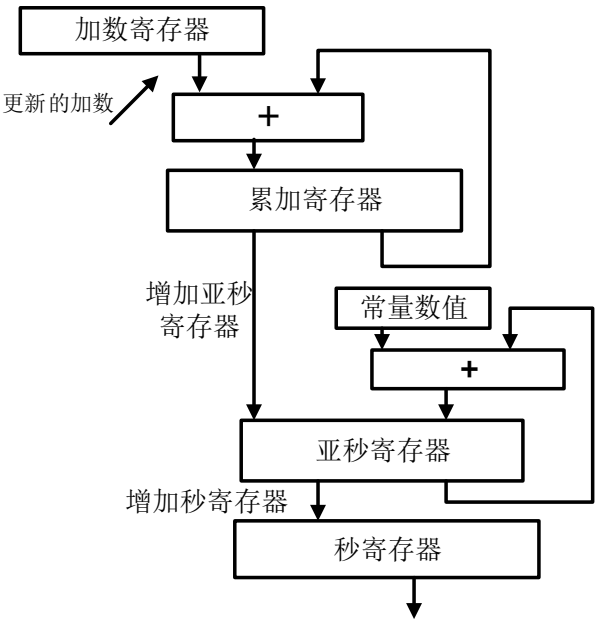
系统时间校准方法

利用PTP输入基准时钟HCLK来更新64位PTP时间。这个PTP时间用来作为记录以太网帧在MII上发送和接收时刻(即时间戳)的依据。这个系统时间的初始化或者校准，既可以用粗调的方式来进行，也可以用精调的方式来进行。

粗调的方式是指，把初始值和偏移值写入时间戳更新寄存器(见第24.8.3 IEEE 1588 时间戳寄存器)。初始化时，用时间戳更新寄存器的值写入系统时钟计数器。在校准时，把时间戳更新寄存器值作为偏移值，系统时间加上或者从中减去这个偏移值。

精调的方式是指，在一段时间里纠正从时钟(基准时钟)相对主时钟(如IEEE1588定义)的频率偏移，不像粗调方式那样能够在一个时钟周期内就完成校正，精调需要一段时间才能完成；采用相对较长的校准时间带来的好处是可以维持时间的线性度，避免了基准时间在PTP Sync消息的间隙内发生突变(或者说大的抖动)。采用这种方法，如下图所示，把加数寄存器的值逐渐加入累加器。累加器的算术进位产生的脉冲令系统时间计数器值增加。累加器和加数寄存器都是32位寄存器。这里累加器起到了高精度频率乘法器或者除法器的作用。下图演示了整套算法。

图 24-24 用精调的方式更新系统时间



系统时钟更新电路需要50MHz的时钟频率来使精度达到20ns。频率的除数是基准时钟频率和要求时钟频率的比。因此，假设基准时钟HCLK是66MHz，计算频率比得66 MHz / 50MHz =1.32。因此，写入加数寄存器的默认加数是2^{32}/1.32，等于0xC1F0 7C1F。

如果基准时钟频率漂低，假设降至65MHz，此时频率比变成65/50 = 1.3，写入加数寄存器的值应当是2^{32}/1.30 = 0xC4EC 4EC4。如果基准时钟漂高，假设升到67MHz，加数寄存器的值必须为0xBF0B 7672。如果频率漂移为0，那么写入加数寄存器的默认加数应当是0xC1F0 7C1F(2^{32}/1.32)。

上图中，累加到亚秒寄存器上的常数是43，这使得系统时间的精度为20ns(也就是说，系统时间累加的步长是20ns)。

软件需要利用Sync消息计算出频率的漂移，并相应地更新累加寄存器的值。开始时，设从时钟加数寄存器值为FreqCompensationValue0：

FreqCompensationValue0 = 2^{32}/频率比 假设对于连续的Sync消息，主从节点之间的时延MasterToSlaveDelay是恒定的，那么使用下文

描述的算法，在若干个 Sync 周期后，就能锁定频率。从时钟就可以确定精确的MasterToSlaveDelay值，并用新的值把从时钟与主时钟同步。算法如下：

- 在主时钟为MasterSyncTime(n)时，主节点向从节点发Sync消息。从节点在它的时钟为SlaveClockTime(n)时，收到Sync消息，并计算出MasterClockTime(n)为

  MasterClockTime(n) = MasterSyncTime(n) + MasterToSlaveDelay(n)
- 当前Sync周期的主时钟计数数目MasterClockCount(n)为

  MasterClockCount(n) = MasterClockTime(n) - MasterClockTime(n-1)，假设 MasterToSlaveDelay 的数值在 Sync 周期 n 与 Sync 周期 n-1 是相同的
- 当前Sync周期的从时钟计数数目SlaveClockCount(n)为

  SlaveClockCount (n) = SlaveClockTime (n) - SlaveClockTime (n-1)
- 当前Sync周期，主时钟计数数目和从时钟计数数目的差别，ClockDiffCount(n)为

  ClockDiffCount(n) = MasterClockCount(n) - SlaveClockCount (n)
- 从时钟的频率比系数，FreqScaleFactor(n)为

  FreqScaleFactor(n) = (MasterClockCount(n) + ClockDiffCount(n)) / SlaveClockCount (n)
- 加数寄存器的频率补偿值，FreqCompensationValue(n)为

  FreqCompensationValue(n) = FreqScaleFactor(n) ×FreqCompensationValue(n-1)

理论上，该算法可以在一个Sync周期内锁定频率。然而，由于网络传播延时和操作环境的改变，实际可能需要几个Sync周期。

该算法具有自校准功能，如果由于种种原因，从时钟初始设定值相对主时钟是不准确的，该算法可以在几个Sync周期内把误差纠正过来。

系统时间生成初始化的编程步骤

设置时间戳控制寄存器(ETH_PTPTSCTRL)的位0为’1’，可以使能时间戳功能。不过在把该位置’1’以后，需要正确地初始化时间戳计数器来开始时间戳操作。正确的初始化流程如下：

1. 设置MACIMR寄存器的位9为’1’，屏蔽时间戳触发中断。
2. 设置时间戳寄存器位0为’1’，使能时间戳。
3. 按照PTP时钟频率编写亚秒累加寄存器。
4. 如果使用精调校准方式，则设置时间戳加数寄存器，并设置时间戳控制寄存器的位5为’1’(更新加数寄存器)。
5. 轮询时间戳控制寄存器直到位5变为’0’。
6. 如果使用精调校准方式，设置时间戳控制寄存器的位1为’1’。
7. 把正确的时间值写入时间戳更新高寄存器和时间戳更新低寄存器。
8. 设置时间戳控制寄存器的位2为’1’(初始化时间戳)。
9. 一旦时间戳计数器初始化为时间戳更新寄存器的值，它就开始工作。
10. 使能MAC的接收端和发送端，即可以正确地记录时间戳。

注意： *如果寄存器ETH_PTPTSCTRL的位0为’0’，关闭了时间戳操作，那么在重新打开此功能时，必须重复以上步骤。*

用粗调方式更新系统时间的编程步骤

采用以下的步骤一次(用粗调方式)同步并更新系统时间：

1. 在时间戳更新高和时间戳更新低寄存器里写入偏移值(正值或者负值)。
2. 设置时间戳控制寄存器的位3(TSSTU)为’1’。

3. 在TSSTU位清’0’时，系统时间就会加上或者从中减去时间戳更新寄存器的值

用精调方式更新系统时间的编程步骤

采用以下的步骤，用减小系统时间抖动的方式(精调方式)同步并更新系统时间：

1. 采用前述” 系统时间校准方法”介绍的算法，计算出希望调快或调慢系统时钟的步率。
2. 更新时间戳。
3. 等待需要的加数寄存器的新值被激活。为此，可以在系统时间达到目标值后打开时间戳触发中断。
4. 把要求的目标时间写入目标时间高和目标时间低寄存器，并设置ETH_MACIM寄存器的位9为’0’来允许时间戳中断。
5. 设置时间戳控制寄存器的位4(TSARU)为’1’。
6. 在这个事件产生中断时，读出ETH_MACIM寄存器。
7. 重新用旧值编写时间戳加数寄存器并设置ETH_TPTSCR寄存器位5为’1’。

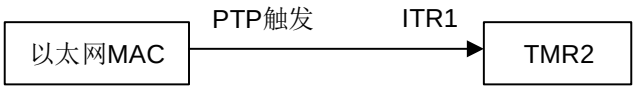
PTP触发与TMR2的内部连接

MAC可以在系统时间大于目标值的时候提供触发中断。使用中断会引入一段已知的中断时延再加上不确定的命令执行时间。

为了避免这部分不确定性，在系统时间大于目标值的时候，PTP会设置一个输出信号为高。这个信号从内部连接到TIM2输入触发。利用这个信号，TIM2由同步后的PTP系统时间触发，可以使用定时器的输入捕获功能，输出比较功能和波形。由于定时器的时钟(PCLK1:TIM2 APB1时钟)与PTP基准时钟(HCLK)是同步的，因此不再有任何不确定的误差。

用户可以通过软件选择把PTP触发信号连接到TIM2 ITR1输入。设置AFIO_MAPR寄存器的位29为’1’可以使能这个连接。下图显示了这个连接。

图 24-25 PTP触发输出连接到TMR2的ITR1



PTP秒脉冲输出信号

PTP脉冲输出详见ETH_PTPPPSCR寄存器描述，下面只介绍emac_pps_sel位（RCC_MISC2寄存器位15）为0时的情况。

PTP脉冲输出用来检查网络全部节点之间的同步。为了能够测试本地从时钟和主时钟之间的差别，由于具有输出每秒脉冲(PPS)信号的功能，在必要时可以把这个输出连接到示波器。这样就可以测量2个时钟之间的差别。PPS输出的脉冲宽度是125ms。

PPS 输出的默认频率为 1 Hz。可使用 PPSFREQ[3:0]（位于 ETH_PTPPPSCR 中）将 PPS 输出的频率设置为 2PPSFREQ Hz。

如果设置为 1 Hz，则使用二进制翻转（TSSSR=0，ETH_PTPTSCTRL 寄存器中的位 9）时， PPS 脉冲宽度为 125 ms；使用数字翻转 (TSSSR=1) 时，PPS 脉冲宽度为 100 ms。如果设 置为 2 Hz 或更高频率，则使用二进制翻转时，PPS 输出的占空比为 50%。

设置AFIO_MAPR寄存器的位30为’1’，可以使能PPS输出功能。

图 24-26 PPS输出



24.6 以太网功能描述：DMA控制器操作

DMA具有独立的发送和接收控制器，还有1个控制和状态寄存器(CSR)空间。发送控制器负责把数据从系统存储器转送至发送FIFO，而接收控制器负责把数据从接收FIFO读出到系统存储器。为了把CPU的操作减到最小，DMA控制器利用描述符来实现数据从源头到目的之间的移动。DMA的设计是以数据包为单位传输，如以太网的帧传输。设置DMA控制器，可以在发送完一个帧、接收到一个帧或者其他正常/错误

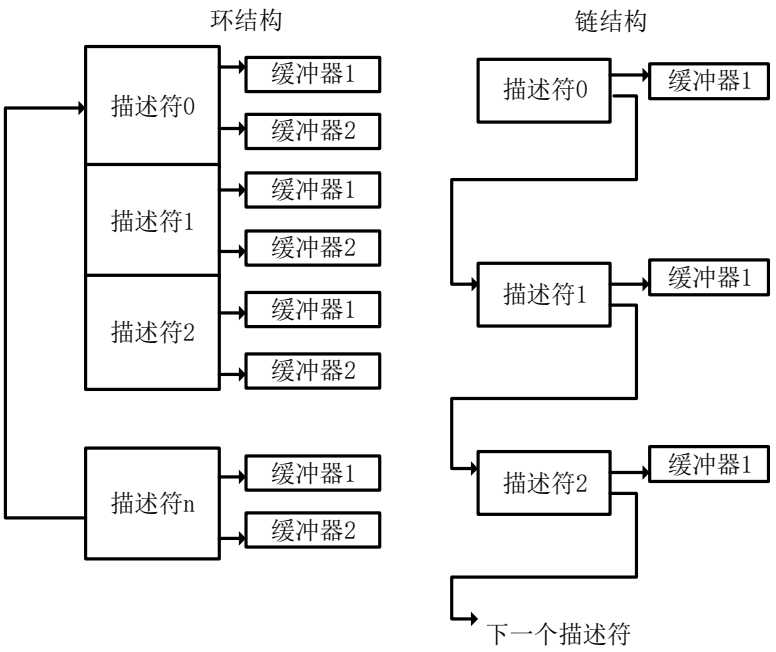
操作的时候产生中断。DMA和CPU之间的通讯通过2种数据结构实现：

- 控制和状态寄存器(CSR)
- 描述符列表和数据缓存

第24.8节描述了控制和状态寄存器的细节。“发送DMA描述符”和“接收DMA描述符”段介绍了描述符的具体情况。

DMA负责把接收到的帧数据传送给AT32F407xx的接收缓存，把AT32F407xx发送缓存里的数据发送出去。在AT32F407xx的存储器里，描述符是以缓存指针的形式存放。有2个描述符队列，一个用作发送，另一个用作接收。两个队列的基地址分别存放在ETH_DMATDLADDR寄存器和ETH_DMARDLADDR寄存器中。描述符队列是(显性或者隐性)前向连接的，最后一个描述符可以指向第一个描述符，而形成环形结构。通过设置接收描述符的RDES1位14和发送描述符的TDES0位20为’1’，可以实现描述符的显性连接。描述符列表存放在MCU的物理内存里，每个描述符可以指向最多2个缓存。这样就允许使用2个不同物理地址的缓存，而不是2个在存储器里连续的缓存。数据缓存存放在MCU的物理内存里，可以存放一个帧的全部或者部分，但是不允许存放一个不属于同一个帧的数据。缓存里只存放数据，缓存的状态信息存放在描述符里。数据链接是指帧的数据分散存放在多个缓存里。但是单个描述符不能用于多个帧。DMA一旦检测到帧结束就会跳到下一个缓存。数据链接功能可以自由地打开和关闭。下图显示了描述符的环结构和链结构

图 24-27 描述符的环结构和链结构



24.6.1 使用DMA发送的初始化步骤

初始化MAC如下：

1. 在ETH_DMABM寄存器中设置AT32F407xx总线访问参数。
2. 在ETH_DMAIE寄存器中屏蔽不需要的中断源。
3. 软件程序生成发送和接收描述符列表，然后写入ETH_DMARDLADDR寄存器和ETH_DMATDLADDR寄存器，把列表的起始地址提供给DMA。
4. 在MAC寄存器1、2、3来中选择期望的过滤器选项。
5. 在MAC的ETH_MACCTRL寄存器中设置并使能发送和接收操作。根据(从PHY读出的)自 商(auto-negotiation)的结果，设置PS位和DM位的值。
6. 设置ETH_DMAOPM寄存器的位13和位1为’1’，开始发送和接收。
7. 发送和接收控制器进入运行模式，并开始从对应的描述符列表里读取描述符。随后接收和发送控制

器开始处理接收和发送操作。发送和接收操作相互独立，可以分别开启和关闭。

24.6.2 主机总线突发访问

如果设置ETH_DMABM寄存器的FB位为’1’，则DMA在AHB主接口上执行固定长度的突发访问。最大突发长度由PBL域(ETH_DMABM寄存器位[13:8])定义。对接收和发送描述符这16字节数据的访问总是以PBL限定的最大可能突发长度进行。

只有在发送FIFO里有足够的空间来容纳预设的突发长度，或者在帧结束之前的数据字节数小于突发长度且发送FIFO能容纳这些字节时，发送DMA才会开始一次数据传输。DMA会向AHB主接口提供起始地址和要传送的字节数量。如果AHB接口设置成固定长度突发传输，那么就会利用 INCR4、INCR8、INCR16和SINGLE传送的最佳组合来传输数据。否则利用INCR(未定义长度) 和SINGLE传送来传输数据。

只有在接收FIFO里的可用数据多于预设的突发长度时，或者在帧结束之前的数据字节数小于突发长度且在接收FIFO里检测到帧结束时，接收DMA才会发起一次数据传输。DMA会向AHB主接口提供起始地址和要传送的字节数量。如果AHB接口设置成固定长度突发传输，那么就会利用 INCR4、INCR8、INCR16和SINGLE传送的最佳组合来传输数据。如果已经传输到帧尾而长度固定的突发传输还没有结束，那么就会进行虚拟传输来完成剩下的传输。如果没有设定成固定长度突发(ETH_DMABM寄存器FB位为’0’)，那么会利用INCR(未定义长度)和SINGLE传送来传输数据。

如果把AHB接口配置成地址对齐，DMA接收和发送控制器会保证AHB发起的第一次突发传输小于或者等于PBL设定的长度。这样，随后的传输都会从与设好的PBL对齐的地址开始。如果PBL>16，DMA只能支持传输地址最多对齐到16，因为AHB接口不支持INCR16以上的传输。

24.6.3 主机数据缓存对齐

发送和接收数据缓存在起始地址上没有限制。在32位存储器的系统中，缓存的起始地址可以对齐到4个字节中的任意一个。不过，DMA发起传输的时候，总是从与总线宽度对齐的地址开始，对于不用到的字节则用虚拟字节代替。这种情况常常在传输以太网帧的起始和结尾时发生。

- 读缓存示例

  如果发送缓存的地址为 0x0000 0FF2，并需要传输 15 字节。DMA 实际会从地址 0x0000 0FF0 开始读 5 个字(32 位)，但是在往 FIFO 发送数据的时候，会忽略或者丢弃多余的字节，即头 2 个字节。同理，最后 3 个字节也会被忽略。除非 DMA 传输的是帧的结尾，否则 DMA 总是保证向发送 FIFO 传输 32 位的数据

- 写缓存示例

  如果接收缓存的地址为 0x0000 0FF2，并需要传输 16 字节长的帧。DMA 实际会从地址 0x0000 0FF0 开始写 5 个 32 位数据。但是传输的头 2 个字节和末尾的 2 个字节是虚拟字节

24.6.4 缓冲区大小计算

DMA不会更改发送和接收描述符的缓存大小域。DMA只会更新描述符的状态信息域(RDES0和 TDES0)。软件需要计算缓存的大小。发送DMA会传输与TDES1缓存大小域相等数目的字节给 MAC控制器。如果描述符标记为帧的第一个部分(TDES0的FS位为’1’)，那么DMA就标记从这个缓存的第一次发送是帧首。如果描述符标记为帧的最后一个部分(TDES0的LS位为’1’)，则DMA 就标记从这个缓存的最后一次发送是帧的结束。接收DMA在缓冲区满或收到帧结束之前，传送数据至缓冲区。如果一个描述符没有标记为帧的最后一个部分(RDES0的LS位)，那么在描述符对应的缓存为满时，有效数据的数目等于缓存长度域减去描述符FS位为’1’时数据缓存指针偏移。在数据缓存指针与数据总线宽度对齐的时候，偏移为0。如果描述符标记为帧的最后一个部分，根据RDES1的缓存长度可以发现缓存有可能不满。为了算出最后一个缓存的有效数据数目，软件需要读出帧长度(FL域，在RDES0的位[29:16])，再减去这个帧存放在之前的缓存内数 据的总长度。接收DMA总是用新的描述符来开始下一帧的传输。

注意： *即便在接收缓存的起始地址和系统数据总线宽度不对齐的时候，系统仍然要求接收缓存的大小和系统数据总线宽度对齐。例如，系统需要一个从0x1000开始，大小为1024字节(1KB)的接收缓存。软件可以把缓存的初始地址偏移到0x1002。这样，DMA就会对前2个地址(0x1000和0x1002)写虚拟字节。帧实际从0x1002开始，由于实际的地址偏移，即使缓存大小被设为1024字节，缓存实际可用的大小变成1022字节。*

24.6.5 DMA仲裁器

DMA的仲裁器负责在发送和接收通道访问AHB主接口时进行仲裁。有两种仲裁方式：轮换和固定优先级。选择轮换仲裁时(ETH_DMABM寄存器DA位为’0’)，在发送和接收DMA同时要求访问数据总线的时候，按照ETH_DMABM寄存器RTPR位设定的比例分配总线。在DA位为’1’ 时，接收DMA总是比发送DMA对总线拥有更高的访问优先级。

24.6.6 DMA错误响应

对于DMA通道发起的数据传输，如果接收到错误响应，那么DMA会停止所有操作，并更新状态寄存器(ETH_DMASTS寄存器)的错误位和总线致命错误位。只有在软件或者硬件复位以太网外设并重新初始化DMA以后，DMA才能恢复操作。

24.6.7 发送DMA设置

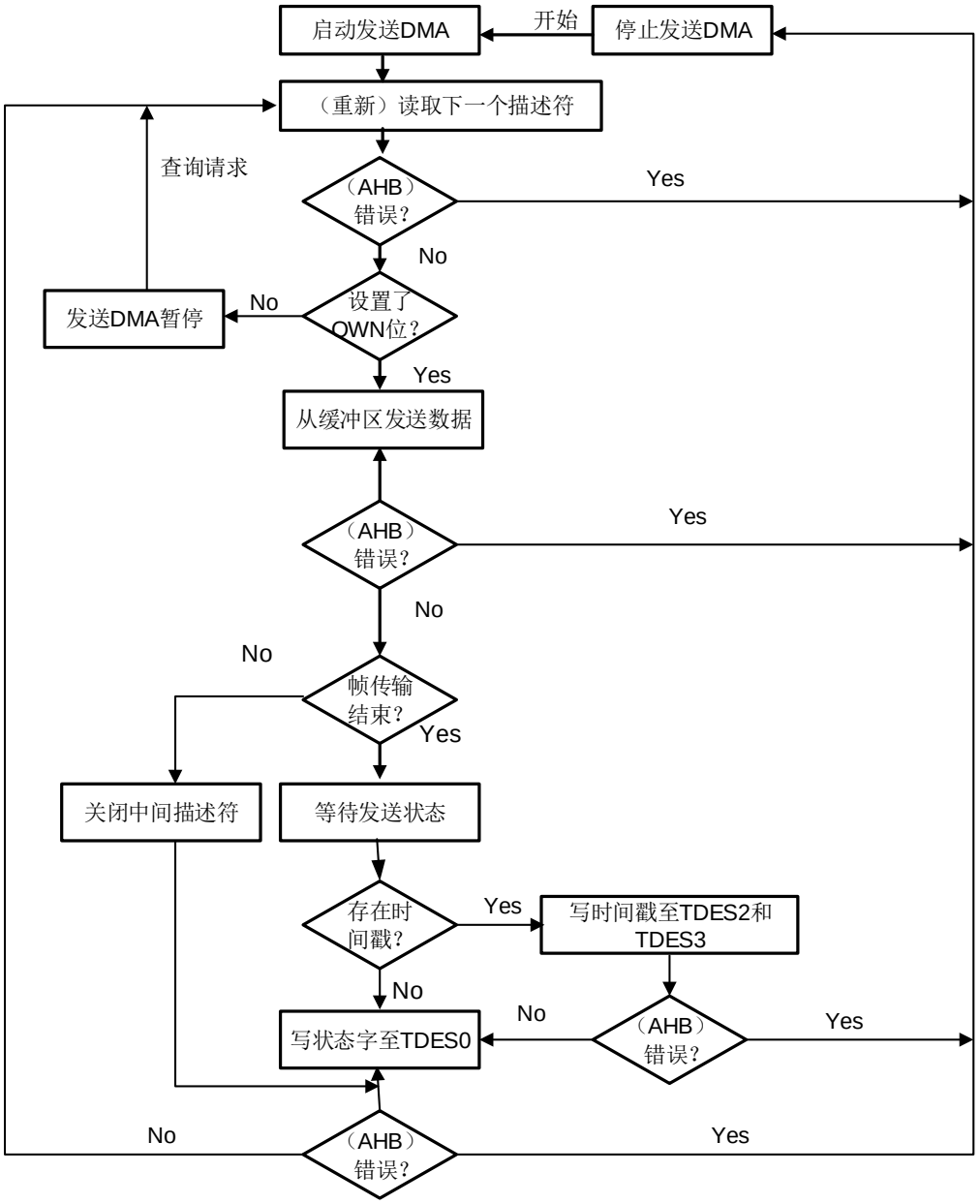
发送DMA操作：默认(非OPSF)模式

在默认模式下，DMA发送控制器的工作流程如下：

1. 在软件建立好包含以太网帧数据的缓存以后，建立发送描述符(TDES0-TDES3)，并置OWN位(TDES[31])为’1’。
2. 设置STXE位(ETH_DMAOPM寄存器位[13])为’1’后，DMA进入运行模式。
3. 在运行模式下，DMA查询发送描述符来获取待发送的帧。查询开始以后，DMA会按照环式或者链式的顺序连续查询描述符。如果DMA检测到标志位提示CPU正在操作描述符或者发生了错误，就会终止传输，并设置发送缓存不可用位(ETH_DMASTS寄存器位2)和正常中断总结位(ETH_DMASTS寄存器位16)为’1’。发送控制器操作跳至步骤9
4. 如果取到的描述符标志位显示该描述符由DMA占有(TDES[31]为’1’)，那么DMA从描述符中解析发送数据缓存的地址。
5. DMA从AT32F407xx存储器里取数据，并把数据发送出去。
6. 如果以太网帧的数据存放在属于不同描述符的不同缓存里，DMA会关闭存放中间数据的描述符，并取下一个缓存对应的描述符，重复步骤3、4、5直到发送完帧结尾的数据
7. 在帧发送完成以后，如果发送状态信息显示这个帧使能了IEEE1588时间戳功能，时间戳的值会写入存放帧尾的缓存对应的发送描述符(TDES2和TDES3)。然后，写状态信息到描述符TDES0。由于OWN位在这时清’0’，CPU自此占有描述符。如果这个帧没有打开时间戳功能，那么DMA不会更改TDES2和TDES3的内容。
8. 在帧发送完成以后，如果描述符的完成中断位(TDES1[31])为’1’，则设置发送中断位(ETH_DMASTS寄存器位[0])为’1’。然后DMA控制器返回步骤3。
9. 在暂停状态下，如果DMA收到发送查询的请求并且溢出中断标志位为’0’，DMA就会尝试 重新获取描述符(因此会回到步骤3)。

下图显示了默认模式下DMA的发送流程

图 24-28 默认模式下发送DMA的操作



发送DMA操作：OPSF模式

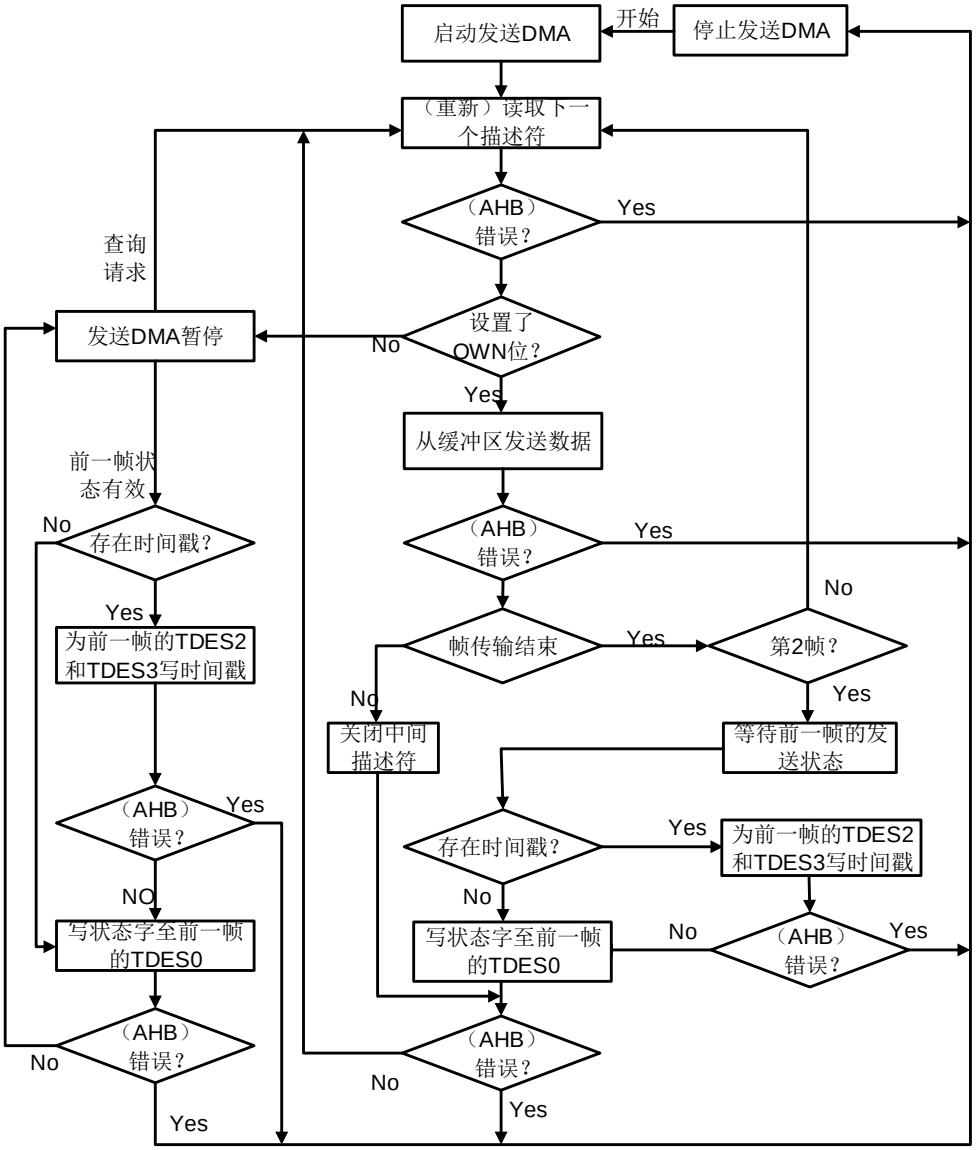
此模式中(OPSF位为’1’时，即ETH_DMAOPM寄存器位2)，在运行状态下，发送DMA可以不必等 到处理完第一个帧的状态信息描述符，就同时取第二个帧。DMA在发送完第一帧数据以后，可以立即查询第二帧的发送描述符。如果第二帧有效，那么DMA在写第一帧的状态信息前就发送第二帧。在OPSF模式下，DMA的操作流程如下：

1. 按照发送DMA默认模式的步骤1-6操作。
2. DMA在没有完成前一帧最后一个描述符状态信息时，直接取下一个描述符。
3. 如果DMA占有取到的描述符，那么就解析发送缓存的地址。如果DMA不占有这个描述符，则进入挂起状态并跳到步骤7。
4. DMA从AT32F407xx的存储器中取出发送帧的数据并发送直到帧尾发送完毕，如果帧数据分散在多个缓存中，DMA关闭存放在中间数据缓存对应的描述符。
5. DMA等待前一帧的发送状态信息和时间戳，在接到状态信息后，如果标志位提示记录了帧的时间戳，DMA把时间戳写入TDES2和TDES3。DMA再把OWN位为’0’的状态信息写入TDES0，这样就关闭了该描述符。如果这个帧没有打开时间戳功能，那么DMA不会更改TDES2和TDES3的内容。

6. 如果使能中断，发送中断位置’1’，DMA在状态信息正常的情况下继续取下一个描述符，并跳到步骤3。如果前一个发送状态信息显示有数据下溢错误，DMA进入暂停状态，并跳到步骤7。
7. 在暂停状态下，如果DMA收到一个待处理的状态信息和时间戳，那么如果这个帧使能了时间戳，DMA就把时间戳写入TDES2和TDES3，然后把状态信息写入TDES0。随后，设置相关的中断标志位并回到暂停状态。
8. 只有在收到发送查询请求(ETH_DMATPD寄存器)后，DMA才会退出暂停状态并进入运行状态，并根据是否有待处理的状态信息跳到步骤1或者步骤2。

下图显示了OPSF模式下DMA的发送流程框图

图 24-29 OPSF模式下发送DMA的操作



发送帧处理

发送DMA要求数据缓存里包含除前导符、填充字节和校验域之外的整个以太网帧数据，目的地址(DA)、源地址(SA)和类型/长度域的值正确有效。如果描述符指示这个帧，要求MAC控制器关闭CRC和插入填充功能，那么缓存里应当包含除前导符以外，包括CRC域在内的完整以太网 帧。一个帧可以分散在不同缓存里，用数据链的形式连接起来；此时应该在第一个缓存的描述符(TDES0[28])和最后一个缓存的描述符(TDES0[29])标记出帧头和帧尾。传输开始时，第一个描述符的TDES0位28应为’1’，DMA则会把数据从内存送到发送FIFO中。同样的，如果当前帧描述符的TDES0位29是’0’，发送DMA就会尝试取下一个描述符，在下一个描述符里，TDES0位28应当是’0’，如果TDES0位29也是’0’，表示这个缓存存放的是帧的中间数据。如果TDES0位29是’1’，表示这个缓存存放的是帧结尾的数据，是这个帧的最后

一个缓存。在这最后一个缓存的数据发送完毕以后，DMA 会返回整个帧的发送状态信息，写入最后一个的发送描述符0 (TDES0)。如果发送完成中断位(TDES0[30])为’1’，则设置发送中断位(ETH_DMASTS寄存器位0) 为’1’。DMA随后取下一个描述符，重复上述步骤。实际的帧发送需要等到在发送FIFO里的数据超过阈值(可通过ETH_DMAOPM寄存器位[16:14]设置)，或者整个帧都位于发送FIFO里时才会开始。设置ETH_DMAOPM寄存器位21为’1’，选择使用存放-转发模式。在DMA传输完帧以后，则会释放描述符(TDES0[31]清’0’)。

发送查询暂停

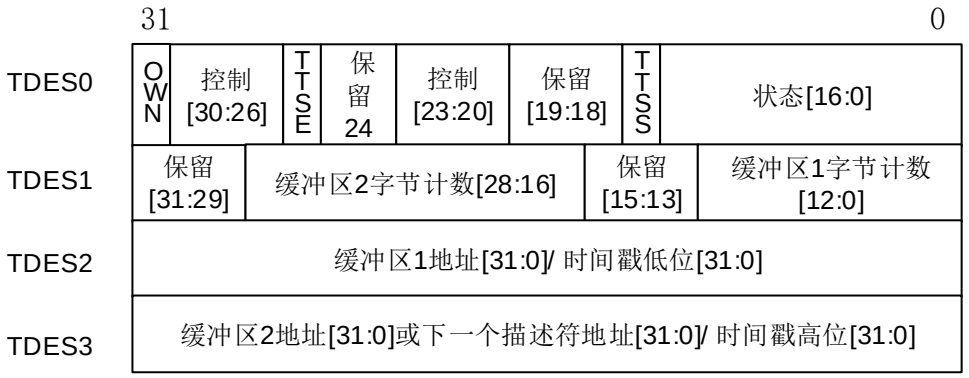
在下列情况下，可以暂停DMA查询描述符：

- DMA检测到CPU占有描述符(TDES0[31]=0)，并且发送缓存不可用标志位(ETH_DMASTS寄 存器位2)为’1’。软件需要把描述符释放给DMA，并向DMA发送查询请求，才能恢复DMA的 发送查询。
- 在检测到数据下溢错误时，DMA会中止帧的传输，并设置TDES0里的相应错误位为’1’。如 果还发生其它错误，则设置异常中断总结位(ETH_DMASTS寄存器位15)和发送数据下溢位(ETH_DMASTS寄存器位5)为’1’，并把该信息写入发送描述符0，并进入暂停。如果DMA因 为第一种情况进入暂停模式，则设置正常中断总结位(ETH_DMASTS寄存器位16)和发送缓存 不可用位(ETH_DMASTS寄存器位2)为’1’。无论发生哪种情况，会保持不变描述符列表的位 置，固定在DMA关闭的最后一个描述符的下一个。软件必须在排除暂停成因之后，明确的 发送查询命令请求，来恢复DMA操作。

发送DMA描述符

描述符结构体包含4个32位字，TDES0、TDES1、TDES2和TDES3的位定义如下图：

图 24-30 传输描述符



- TDES0：发送描述符字0

在初始化描述符的时候需要编写控制位[30:26]+位[23:20]和 OWN 位。在 DMA 更新，或者说写回描述符的时候，会清除所有的控制位和 OWM 位，只报告状态信息标志位。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OWN	IC	LS	FS	DC	DP	TTSE	保留	CIC		TER	TCH	保留		TTSS	IHE
rw	rw	rw	rw	rw	rw	rw	res	rw	rw	rw	rw	res		rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ES	JT	FF	IPE	LCA	NC	LCO	EC	VF	CC				ED	UF	DB
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31	**OWN**：占有位 (Own bit) 1：表示 DMA 占有描述符。0：表示 CPU 占有描述符。 DMA 在传输完整个帧或者这个缓存里的数据全部读出以后把该位清’0’。每个帧的第一个缓存描 述符的占有位，必须在后面缓存描述符的占有位全部置’1’以后，才能置’1’。
位 30	**IC**：完成时中断 (Interrupt on completion) 置’1’时，在当前帧发送完成以后，会把传输中断位(ETH_DMASTS 寄存器位[0])置’1’。
位 29	**LS**：最后分块 (Last segment) 置’1’时，表示缓存存放着帧的最后一个分块。

位 28	**FS**：第一分块 (First segment) 置'1'时，表示缓存存放着帧的第一个分块。
位 27	**DC**：不计算 CRC (Disable CRC) 置'1'时，MAC 不再在发送帧结束时插入循环冗余检测(CRC)域。该位只有在 FS 位(TDES0[28])为'1'时才有效。
位 26	**DP**：不填充 (Disable pad) 1：MAC 不对帧长不足 64 字节的帧自动填充字节。 0：MAC 对帧长不足 64 字节的帧自动添加填充字节，并且不管 DC 位(TDES0[27])的取值而插入 CRC 数值。该位只在 FS 位(TDES0[28])为'1'时有效。
位 25	**TTSE**：使能发送时间戳 (Transmit time stamp enable) 置'1'时，并且 TSE 位(ETH_PTPTSCTRL[0])也为'1'时，这个描述符对应的帧会打开 IEEE1588 硬件 时间戳功能。该位只在 FS 位(TDES0[28])为'1'时有效。
位 24	保留
位 23:22	**CIC**：校验和插入控制 (Checksum insertion control) 这 2 位控制校验和的计算和插入： 00：不插入校验和； 01：只使能 IP 报头的校验和计算和插入； 10：使能 IP 报头和数据域的校验和计算和插入，但是硬件不计算伪报头的校验和； 11：使能 IP 报头和数据域的校验和计算和插入，硬件也计算伪报头的校验和。
位 21	**TER**：环形发送结束 (Transmit end of ring) 置'1'时，表示到达描述符队列的最后一个描述符。DMA 返回队列的起始地址，形成描述符环。
位 20	**TCH**：第二地址链表 (Second address chained) 置'1'时，表示描述符里的第二个地址是下一个描述符的地址，而不是第二个缓存的地址。该位为'1'时，则 TBS2(TDES1[28:16])的值不起作用。TDES0[21]的功能优先于本位(TDES0[20])。 该位只在 FS 位(TDES0[28])为'1'时才有效。
位 19:18	保留
位 17	**TTSS**：发送时间戳状态 (Transmit time stamp status) 作为标志位，置'1'时表示记录下了描述符对应的帧时间戳，记录的时间戳放在 TDES2 和 TDES3 处。该位只在 LS 位(TDES0[29])为'1'时才有效。
位 16	**IHE**：IP 报头错误 (IP header error) 置'1'时，表示 MAC 发送端发现了 IP 数据包报头的错误。对 IPv4 数据包，MAC 把报头的长度域和收到的字节数目比较，不符合则报错。对 IPv6 数据包，如果主报头长度不是 40 字节时则报错。另外，IPv4 或者 IPv6 帧的长度/类型域值和报头的版本信息必须匹配。对 IPv4 帧，如果报头长度域值小于 0x5，该位也置' 1' 报错。
位 15	**ES**：错误汇总 (Error summary) 该位为下列位的逻辑"或" TDES0[14]：啰嗦(Jabber)超时； TDES0[13]：帧清空； TDES0[11]：载波丢失； TDES0[10]：无载波； TDES0[9]：迟到冲突； TDES0[8]：冲突过多； TDES0[2]：顺延(Deferral)过多； TDES0[1]：数据下溢错误； TDES0[16]：IP 报头错误； TDES0[12]：IP 数据错误。
位 14	**JT**：啰嗦超时 (Jabber timeout) 置'1'时，表示 MAC 发送端发生了啰嗦超时。该位只在 MAC 设置寄存器的 JAD 位不为'1'时才会被置'1'。
位 13	**FF**：帧清空 (Frame flushed) 置'1'时，表示由于 CPU 发出命令，DMA/MTL 把帧从 FIFO 中清空。
位 12	**IPE**：IP 数据错误 (IP payload error) 置'1'时，表示 MAC 发送端发现了 IP 数据包的 TCP、UDP 或者 ICMP 的 IP 数据错误。发送端会核对 IPv4 或者 IPv6 显示的数据长度与实际收到的 TCP、UDP 和 ICMP 数据数目，不符合就置'1'报错。
位 11	**LCA**：载波丢失 (Loss of carrier) 置'1'时，表示帧发送的时候发生了载波丢失(在发送时，MII_CSR 信号在一个或一个以上发送时 钟周期中为无效状态)。该位只有在半双工模式下，发送帧没有冲突时有效。
位 10	**NC**：无载波 (No carrier)

	置'1'时，表示帧发送的时候 PHY 的载波侦听信号无效。
位 9	**LCO**：迟到冲突 (Late collision) 置'1'时，表示帧因为发送时在冲突窗口(MII 模式下，包括前导符的 64 个字节时间)之后出现冲突 而中止发送。如果溢出错误位置'1'，该位无效。
位 8	**EC**：冲突过多 (Excessive collision) 置'1'时，表示帧因为发送时连续出现 16 次冲突而中止发送。如果 MAC 设置寄存器的 RD(不进行重试)位为'1'，那么在发生一次冲突后，该位就置'1'，并中止发送。
位 7	**VF**：VLAN 帧 (VLAN frame) 置'1'时，表示发送的帧是 VLAN 帧。
位 6:3	**CC**：冲突计数 (Collision count) 该 4 位值记录了帧发送出去前出现的冲突次数。在 EC 位(TDES0[8])为'1'时，该位无效。
位 2	**EC**：顺延过多 (Excessive deferral) 置'1'时，表示在 MAC 设置寄存器的顺延位为'1'时，因为顺延超过 24288 位的时间而结束发送。
位 1	**UF**：数据下溢错误 (Underflow error) 置'1'时，表示从 RAM 送到 MAC 的数据过迟，导致 MAC 中止发送帧。数据下溢错误表示 DMA 在发送帧的时候遇到了空的缓存。发送过程进入暂停状态，并设置发送数据下溢位(ETH_DMASTS 寄存器位 5)和发送状态位(ETH_DMASTS 寄存器位 0)置' 1' 。
位 0	**DB**：顺延位 (Deferred bit) 置'1'时，表示 MAC 因为载波被占用而推迟发送。该位只在半双工模式下有效。

- TDES1：发送描述符字1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留			TBS2												
		res													rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留			TBS1												
		res													rw

位 31:29	保留
位 28:16	**TBS2**：发送缓存 2 大小 (Transmit buffer 2 size) 这些位给出了第二个数据缓存的大小(以字节记)，如果 TDES0[20]位为'1' 时，这些位无效。
位 15:13	保留
位 12:0	**TBS1**：发送缓存 1 大小 (Transmit buffer 1 size) 这些位给出了第一个数据缓存的大小(以字节记)，如果它的值是 0，那么 DMA 跳过这个缓存，根据 TDES0[20]位使用缓存 2 或者下一个缓存。

- TDES2：发送描述符字2

TDES2包含描述符的第一个缓冲区的地址指针，或时间戳数据。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TBAP1/TTSL															
rw															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TBAP1/TTSL															
rw															

位 31:0	**TBAP1/TTSL**：发送缓存 1 地址指针/发送帧时间戳低位 (Transmit buffer 1 address pointer / Transmit frame time stamp low) 位 31:0 这些位有 2 个功能：应用程序用它们指示 DMA 数据在内存中的位置。等到数据发送完后，DMA 可以用它们存放帧的时间戳 **TBAP1**：当程序把描述符释放给 DMA 时(TDES[31]为'1')，这些位表示缓存 1 的物理地址。对缓存的地址对齐不做限制。 **TTSL**：在 DMA 把 TDES[31]位清'0'之前，DMA 会把对应发送帧的时间戳低 32 位写入这些位，覆盖原来的 TBAP1 内容。只有在该帧的时间戳使能(TDES0 位 25 的 TTSE 位为'1')和帧的最后部分 LS 位为'1'时，DMA 才会把时间戳写入这些位

- TDES3：发送描述符字3

TDES3包含描述符的第二个缓冲区的地址指针或下一个描述符，或时间戳数据。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BAP2/TTSH															
rw															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TBAP2/TTSH															
rw															

位 31:0	**TBAP2/TTSH**：发送缓存 2 地址指针(下个描述符地址)/发送帧时间戳高位 (Transmit buffer 2 address pointer (Next descriptor address) / Transmit frame time stamp high) 这些位有 2 个功能：程序用它们指示 DMA 数据在内存中的位置。等到数据发送完后，DMA 可以用它们存放帧的时间戳 **TBAP2**：当程序把描述符释放给 DMA 的时候(TDES[31]为'1')，如果描述符队列为环式时，这些位表示缓存 2 的物理地址。如果第二地址链位(TDES0[20])为'1'，这些位表示下一个描述符的物理地址。在 TDES0[20]位为'1'时，指针地址必须和总线宽度对齐 **TTSH**：发送帧时间戳高位 (Transmit frame time stamp high) DMA 会把对应帧的时间戳高 32 位写入这些位。只有在 LS 位(帧的最后一部分)为'1'时，DMA 才会把时间戳写入这些位

带IEEE1588时间戳的发送DMA描述符格式

在创建描述符(TDES0的OWN位为'1')时，描述符的格式(如前面所述)和域的意义保持不变。然而，如果使能了IEEE1588 功能，则DMA 关闭描述符(TDES0 的OWN 位清'0') 时，TDES2 和TDES3有不同的意义。

如下图所示，发送描述符会添加额外的时间戳控制和状态位(TTSE和TTSS)。如果在OWN位 为'1'时，设置TTSE位为'1'，则MAC控制器为描述符对应的以太网帧帧生成时间戳。在DMA关 闭描述符的时候(OWN位清'0')，DMA时间戳写入TDES2和TDES3时，会设置TTSS位为'1'。

图 24-31 IEEE1588时间戳使能时的发送描述符格式

	31						0
TDES0	OWN	控制 [30:26]	TTSE	保留 24	控制 [23:20]	保留 [19:18]	TTSS	状态[16:0]
TDES1	保留 [31:29]	缓冲区2字节计数[28:16]			保留 [15:13]	缓冲区1字节计数 [12:0]	
TDES2	缓冲区1地址[31:0]/ 时间戳低位[31:0](1)						
TDES3	缓冲区2地址[31:0]或下一个描述符地址[31:0]/ 时间戳高位[31:0](1)						

DMA在把TDES0的OWN位清'0'之前，会用时间戳的值更新TDES2和TDES3：TDES2为时间戳的低32位(在后 续"接收描述符字2"段落中，该域被称为TTSL)，TDES3为时间戳的高32位(在后续"接收描述符字3"段落中，该域被称为TTSH)。

- TDES0：发送描述符字0：发送时间戳控制和状态

这个域的数值应该留给DMA在关闭描述符时操作。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OWN	IC	LS	FS	DC	DP	TTSE	保留	CIC		TErw	TCH	保留		TTSS	IHE
rw	rw	rw	rw	rw	rw	rw	res	rw	rw	rw	rw	res		rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ES	JT	FF	IPE	LCA	NC	LCO	EC	VF	CC				ED	UF	DB
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31	**OWN**: 占有位 (Own bit) 1：表示 DMA 占有描述符。0：表示 CPU 占有描述符。 DMA 在传输完整个帧或者这个缓存里的数据全部读出以后把该位清'0'。每个帧的第一个缓存描 述符的占有位，必须在后面缓存描述符的占有位全部为'1'以后，才能置'1'。
位 30	**IC**：完成时中断 (Interrupt on completion) 置'1'时，在当前帧发送完成以后，会把传输中断位(ETH_DMASTS 寄存器位[0])置'1'。
位 29	**LS**：最后分块 (Last segment) 置'1'时，表示缓存存放着帧的最后一个分块。
位 28	**FS**：第一分块 (First segment) 置'1'时，表示缓存存放着帧的第一个分块。
位 27	**DC**：不计算 CRC (Disable CRC) 置'1'时，MAC 不再在发送帧结束时插入循环冗余检测(CRC)域。该位只在 FS 位(TDES0[28])为'1'时才有效。
位 26	**DP**：不填充 (Disable pad) 1：MAC 不对帧长不足 64 字节的帧自动填充字节。 0：MAC 对帧长不足 64 字节的帧自动添加填充字节，并且不管 DC 位(TDES0[27])的取值而插入 CRC 数值。该位只在 FS 位(TDES0[28])为'1'时才有效。
位 25	**TTSE**：使能发送时间戳 (Transmit time stamp enable) 置'1'时，并且 TSE 位(ETH_PTPTSCTRL[0])也为'1'时，这个描述符对应的帧会打开 IEEE1588 硬件 时间戳功能。该位只在 FS 位(TDES0[28])为'1'时才有效。
位 24	保留
位 23:22	**CIC**：校验和插入控制 (Checksum insertion control) 这 2 位控制校验和的计算和插入： 00：不插入校验和； 01：只使能 IP 报头的校验和计算和插入； 10：使能 IP 报头和数据的校验和计算和插入，但是硬件不计算伪报头的校验和； 11：使能 IP 报头和数据的校验和计算和插入，硬件也计算伪报头的校验和。
位 21	**TER**：环形发送结束 (Transmit end of ring) 置'1'时，表示到达描述符队列的最后一个描述符。DMA 返回队列的起始地址，形成描述符环。
位 20	**TCH**：第二地址链表 (Second address chained) 置'1'时，表示描述符里的第二地址是下一个描述符的地址，而不是第二个缓存的地址。该位 为'1'时，则 TBS2(TDES1[28:16])的值不起作用。TDES0[21]的功能优先于本位(TDES0[20])。 该位只在 FS 位(TDES0[28])为'1'时才有效。
位 19:18	保留
位 17	**TTSS**：发送时间戳状态 (Transmit time stamp status) 作为标志位，为'1'时表示记录下了描述符对应帧的时间戳，记录的时间戳放在 TDES2 和 TDES3 处。该位只在 LS 位(TDES0[29])为'1'时才有效。
位 16	**IHE**：IP 报头错误 (IP header error) 置'1'时，表示 MAC 发送端发现了 IP 数据包报头的错误。对 IPv4 数据包，MAC 把报头的长度域和 从收到的字节数目比较，不符合就报错。对 IPv6 数据包，如果主报头长度不是 40 字节就报错。 另外，IPv4 或者 IPv6 帧的长度/类型域值和报头的版本信息必须匹配。对 IPv4 帧，如果报头长度 域值小于 0x5，该位也置'1'报错。
位 15	**ES**：错误汇总 (Error summary) 该位为下列位的逻辑"或" TDES0[14]：啰嗦超时； TDES0[13]：帧清空； TDES0[11]：载波丢失； TDES0[10]：无载波； TDES0[9]：迟到冲突； TDES0[8]：冲突过多； TDES0[2]：顺延过多； TDES0[1]：数据下溢错误； TDES0[16]：IP 报头错误； TDES0[12]：IP 数据错误。
位 14	JT：啰嗦超时 (Jabber timeout)置'1'时，表示 MAC 发送端发生了啰嗦超时。该位只在 MAC 设置寄存器的 JAD 位不为'1'时才会被 置'1'。

位 13	**FF**：帧清空 (Frame flushed) 置'1'时，表示由于 CPU 发出命令，DMA/MTL 把帧从 FIFO 中清空。
位 12	**IPE**：IP 数据错误 (IP payload error) 置'1'时，表示 MAC 发送端发现了 IP 数据包的 TCP、UDP 或者 ICMP 的 IP 数据错误。发送端会核 对 IPv4 或者 IPv6 显示的数据长度与实际收到的 TCP、UDP 和 ICMP 数据数目，不符合就置'1'报 错。
位 11	**LCA**：载波丢失 (Loss of carrier) 置'1'时，表示帧发送的时候发生了载波丢失(在发送时，MII_CSR 信号在一个或一个以上发送时 钟周期中为无效状态)。该位只有在半双工模式下，发送帧没有冲突时有效。
位 10	**NC**：无载波 (No carrier) 置'1'时，表示帧发送的时候 PHY 的载波侦听信号无效。
位 9	**LCO**：迟到冲突 (Late collision) 置'1'时，表示帧因为发送时在冲突窗口(MII 模式下，包括前导符的 64 个字节时间)之后出现冲突 而中止发送。如果溢出错误位为'1'，该位无效。
位 8	**EC**：冲突过多 (Excessive collision) 置'1'时，表示帧因为发送时连续出现 16 次冲突而中止发送。如果 MAC 设置寄存器的 RD(不进行 重试)位为'1'，那么在一次冲突发生后，该位就置'1'，并中止发送。
位 7	**VF**：VLAN 帧 (VLAN frame) 置'1'时，表示发送的帧是 VLAN 帧。
位 6:3	**CC**：冲突计数 (Collision count) 该 4 位值记录了帧发送出去前出现的冲突次数。在冲突过多位(TDES0[8])为'1'时，该位无效。
位 2	**EC**：顺延过多 (Excessive deferral) 置'1'时，表示在 MAC 设置寄存器的顺延位为'1'时，因为顺延超过 24288 位时间而结束发送。
位 1	**UF**：数据下溢错误 (Underflow error) 置'1'时，表示从 RAM 送到 MAC 的数据过迟，导致 MAC 中止发送帧。数据下溢错误表示 DMA 在 发送帧的时候遇到了空的缓存。发送过程进入暂停状态，并设置发送数据下溢位(ETH_DMASTS 寄存器位 5)和发送状态位(ETH_DMASTS 寄存器位 0)置'1'。
位 0	**DB**：顺延位 (Deferred bit) 置'1'时，表示 MAC 因为载波被占用而推迟发送。该位只在半双工模式下有效。

- TDES1：发送描述符字1

参见前述"TDES1：发送描述符字1"。

- TDES2：发送描述符字2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TBAP1/TTSL															
rw															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TBAP1/TTSL															
rw															

位 31:0	**TTSL**：发送帧时间戳低位 (Transmit frame time stamp low) DMA 会把对应帧的时间戳低 32 位写入这些位。只有在 LS 位(帧的最后一部分)为'1'时，DMA 才会把时间戳写入这些位

- TDES3：发送描述符字3

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BAP2/TTSH															
rw															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TBAP2/TTSH															
rw															

位 31:0	TTSH：发送帧时间戳高位 (Transmit frame time stamp high) DMA 会把对应帧的时间戳高 32 位写入这些位。只有在 LS 位(帧的最后一部分)为'1'时，DMA 才会 把时间戳写入这些位

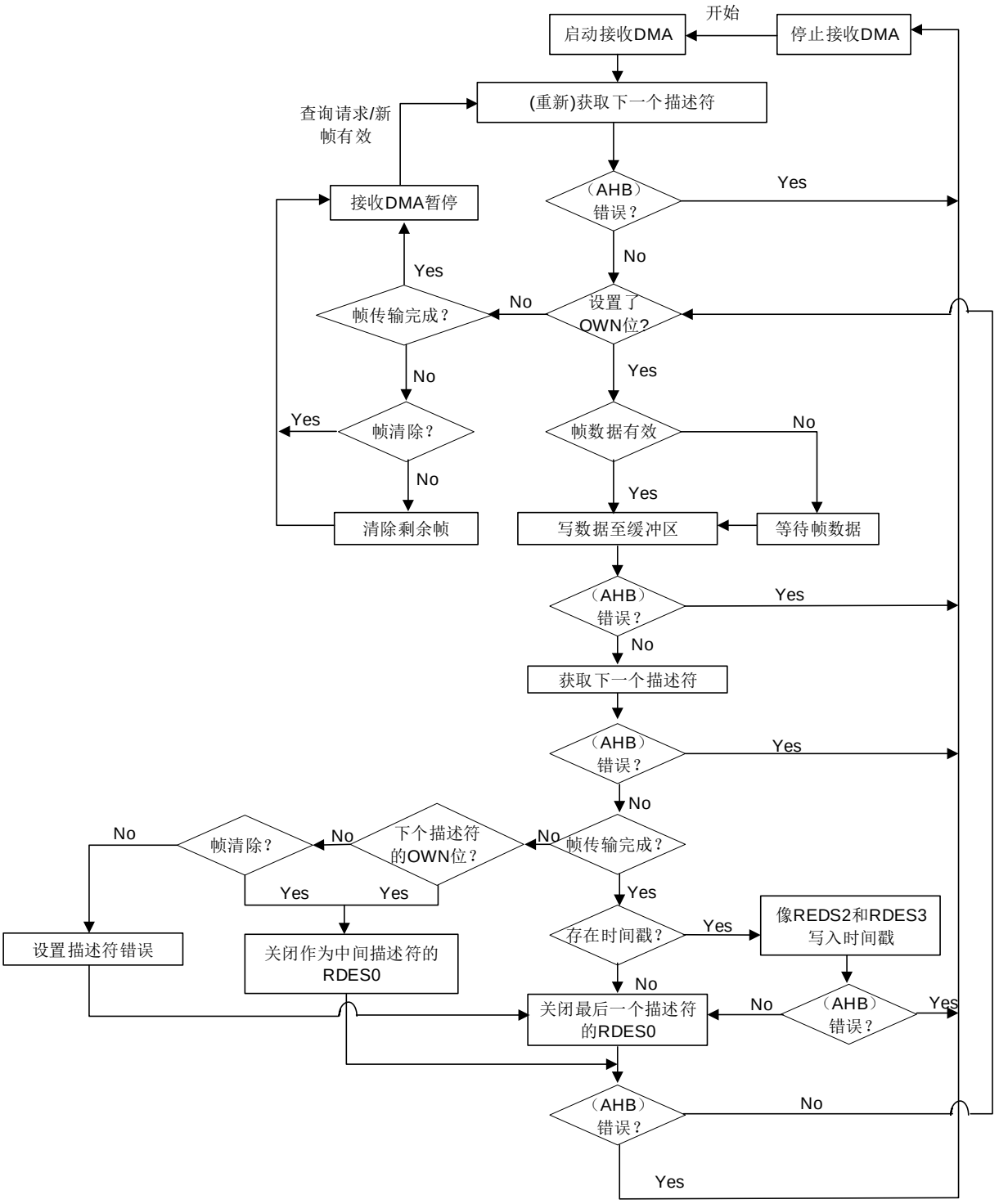
24.6.8 接收DMA设置

下图显示了接收DMA控制器的接收过程，详述如下：

1. CPU建立接收描述符(RDES0~RDES3)，并置OWN位为’1’。
2. 当SRXE位(ETH_DMAOPM[1])为’1’，DMA进入运行状态，在此状态下，DMA尝试获取接收描述符，如果取到的描述符不可用(被CPU占有)，则DMA进入暂停状态，并跳到步骤9。
3. DMA从取到的描述符解析出接收缓存地址。
4. 处理接收到的帧，并将其发送到接收缓存。
5. 如果缓存被填满或者帧传输结束，接收控制器会取描述符队列中的下一个接收描述符。
6. 如果当前帧传输结束，DMA跳到步骤7。如果当前帧传输没有结束(未接收到帧尾EOF)而DMA又没能占有下一个描述符，这时，如果没有使能帧清空功能，则DMA设置RDES0的描述符错误位。DMA关闭当前描述符(OWN位清’0’)，并清除RDES1的LS位为’0’，把描述符标记为存放帧的中间描述符(如果没有使能帧清空功能，则标记为最后帧)，然后跳到步骤8。如果当前帧传输没有结束，而DMA成功占有下一个描述符，那么DMA关闭当前描述符，标记其为中间描述符，然后退回步骤4。
7. 如果使能了IEEE1588时间戳功能，DMA把时间戳写入当前描述符的RDES2和RDES3。然后再把接收到帧的状态信息写入RDES0，把OWN位清’0’，把LS位置’1’。
8. 接收控制器检查对列当前位置描述符的OWN位，如果CPU占有这个描述符(OWN=0)，且接收缓存不可用位(ETH_DMASTS[7])为’1’，DMA进入暂停状态，跳到步骤9。如果DMA占有这个描述符，那么接收控制器回到步骤4，等待接收下一帧。
9. 在进入暂停状态前，DMA清空接收FIFO里不完整的帧(可以通过ETH_DMAOPM位24控制使能这个清空帧功能)。
10. DMA收到接收查询命令或者接收FIFO里有下一个接收到帧的帧头时，DMA退出暂停状态，接收控制器跳到步骤2，并取下一个描述符。

DMA直到完成时间戳写回并准备写回接收状态信息之前，才会应答接收状态信息。如果应用程序通过CSR打开了时间戳功能，但是又没有收到有效的时间戳值(比如接收FIFO在写入时间戳前就满了)，DMA会在RDES2和RDES3写入全’1’。如果没有打开时间戳功能，则保持RDES2和RDES3不变。

图 24-32 接收DMA操作



获取接收描述符

为接收下一帧数据做准备，接收控制器总是会尝试获取另一个描述符。只要满足下列条件中的任意一个，DMA就会尝试获取接收描述符：

- 在DMA进入运行状态后，接收开始/停止位(ETH_DMAOPM[1])立刻被置为’1’
- 在接收到当前帧的结尾之前，当前描述符的缓存就被填满
- 控制器已经完成当前帧的接收，但还没有关闭当前的接收描述符
- 由于接收到新的帧，而接收流程由于描述符被CPU占有(RDES[31]=0)而暂停
- 接到接收查询命令

接收帧处理

只有在接收帧通过地址过滤器，且接收FIFO直通模式下帧长大于等于设好的接收阈值，或者接收FIFO存储- 转发模式下FIFO里写入了完整的帧时，MAC才会把接收到的帧数据转发进 AT32F407xx的内存。如果帧没能通过地址过滤器，而ETH_MACCFFR寄存器位31(接收所有帧)不为’1’，则MAC模块直接把帧丢弃。接收FIFO会清空由于冲突，或者过早停止接收而造成的短于64字节的帧。在接收FIFO接收到64字节(设好的阈值)后，DMA开始把数据转发给当前描述符对应的接收缓存。在DMA的AHB接口数据接收准备就绪后(发送DMA未通过AHB接口取发送帧数据时)，DMA把RDES0的FS位(RDES0[9])置’1’，表示缓存里是帧的第一部分。在DMA填满接受缓存，或者帧接收完毕后，DMA把OWN位(RDES[31])清’0’，释放描述符。如果完整的接收帧包含在单一的描述符中，则LS位(RDES[8])和FS位(RDES[9])都为’1’。DMA取下一个描述符，把前一个描述符的LS位置为’1’，写回接收状态信息到前一个描述符，再将其释放。然后，DMA把接收中断位(ETH_DMASTS[6])置为’1’。上面的步骤会一直重复直到DMA发现描述符被CPU占有，此时接收流程会把接收缓存不可用位(ETH_DMASTS[7])置为’1’，并进入暂停状态。当前接收描述符列表的位置不变。

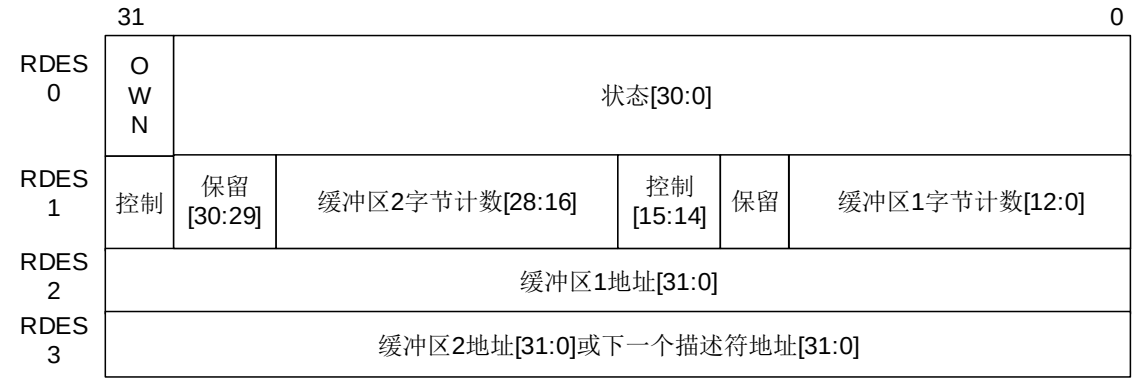
接收流程暂停

在接收流程暂停的时候，如果接收到新的帧，DMA会重新从AT32F407xx内存里取当前描述符。如果这时DMA能够占有描述符，接收流程就会重新进入运行模式。如果CPU仍然占有描述符，那么DMA默认会丢弃在接收FIFO顶部的这个帧，并且丢失帧计数器加1。如果接收FIFO里有一个以上的帧，则重复上述的步骤。把DMA操作模式寄存器(ETH_DMAOPM)的位24(DFRFD位)置为’1’可以阻止清空或者丢弃接收FIFO顶部的帧；这时，接收流程会把接收缓存不可用位置为’1’，并回到暂停状态。

接收DMA描述符

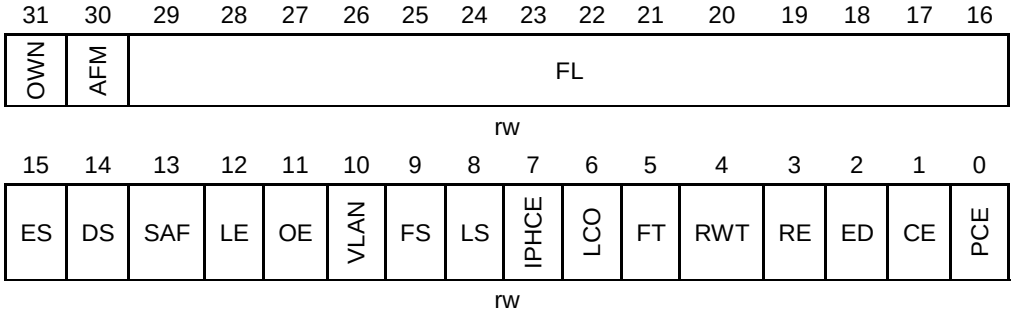
如下图，描述符结构体包含4个32位字，RDES0、RDES1、RDES2和RDES3。

图 24-33 接收DMA描述符



- RDES0：接收描述符字0

RDES0包含了接收帧的状态信息，帧长和描述符的占有信息。



位 31	**OWN**：占有位 (Own bit) 1：表示 DMA 占有描述符。 0：表示 CPU 占有描述符。 DMA 在传输完整个帧或者填满描述符对应的缓存以后把该位清’0’。
位 30	**AFM**：目的地址过滤器未通过 (Destination address filter fail)置’1’时，该位表示帧没有通过 MAC 控制器的目的地址(DA)过滤器。

位 29:16	**FL**：帧长 (Frame length) 这些位表示了发往内存包括 CRC 的接收帧长度，以字节为单位。只有在 LS 位(RESD0[8])为'1'且 描述符错误位(RDES0[14])为'0'时，这些位才有效。 只有在 LS 位(RESD0[8])为'1'时，这些位的值才表示接收帧的长度。如果 LS 位和错误汇总位(ES 位，RDES0[15])都为'0'，则这些位的值表示帧当前传送到内存里的累计字节数目。
位 15	**ES**：错误汇总 (Error summary) 该位为下列位的逻辑"或"： RDES0[1]：CRC 错误； RDES0[3]：接收错误； RDES0[4]：看门狗超时； RDES0[6]：迟到冲突； RDES0[7]：巨人帧(如果 RDES0[7]位提示 IPv4 报头校验和错误，该位无效)； RDES0[11]：溢出错误； RDES0[14]：描述符错误。 该位在 LS 位(RDES0[8])为'1'时才有效。
位 14	**DE**：描述符错误 (Descriptor error) 置'1'时，该位表示由于描述符的缓存装不下当前帧，因而帧被切断，而 DMA 又不占有下一个描 述符。该位在 LS 位(RDES0[8])为'1'时才有效。
位 13	**SAF**：源地址过滤器未通过 (Source address filter fail) 置'1'时，该位表示帧没有通过 MAC 控制器的源地址(SA)过滤器。
位 12	**LE**：长度错误 (Length error) 置'1'时，表示接收到帧的实际长度与以太网帧头长度/类型域的值不相符。该位在 RDES0[5]位 为'0'时才有效。
位 11	**OE**：溢出错误 (Overflow error) 置'1'时，表示由于接收 FIFO 溢出，接收到的帧被破坏。
位 10	**VLAN**：VLAN 标签 (VLAN tag) 置'1'时，表示描述符指向的帧被 MAC 控制器标记为 VLAN 帧。
位 9	**FS**：第一个描述符 (First descriptor) 置'1'时，表示这个描述符包含帧的第一个缓冲区，如果缓冲区的长度为 0，则第 2 个缓冲区包含 帧的开头；如果第 2 个缓冲区的长度也是 0，则下一个缓冲区包含帧的开头。
位 8	**LS**：最后一个描述符 (Last descriptor) 置'1'时，表示这个描述符指向的缓存区是帧的最后一个缓冲区。
位 7	**IPHCE**：IPv 报头校验和错误 (IPv header checksum error) 置'1'时，表示 IPv4 或者 IPv6 的报头错误。错误可能是由于以太网类型域和 IP 版本域值不匹配，IPv4 报头校验和不对或者以太网帧的 IP 报头字节数不足造成的。
位 6	**LCO**：迟到冲突 (Late collision) 置'1'时，表示在半双工模式下接收帧的时候发生了迟到冲突。
位 5	**FT**：帧类型 (Frame type) 置 '1' 时表示接 收到的帧是以太网帧 (LT 域 大于等于 0x0600) 。置 '0' 时 表示接收到的帧是 IEEE802.3 帧。在接收到帧是小于 14 字节的过短帧时，该位无效。
位 4	**RWT**：接收看门狗超时 (Receive watchdog timeout) 置'1'时表示在接收当前帧的时候，看门狗超时，之后，当前接收帧被截断。
位 3	**RE**：接收错误 (Receive error) 置'1'时表示在接收帧过程中 RX_DV 有效时，RX_ERR 信号有效。
位 2	**DE**：Dribble 位错误 (Dribble bit error) 置'1'时表示接收到的帧长度不是字节的整数倍(奇数个 4 位数据)。该位只在 MII 模式下有效。
位 1	**CE**：CRC 错误 (CRC error) 置'1'时表示接收到帧发生循环冗余检测(CRC)错误。该位只在 LS 位(RDES0[8])为'1'时有效。

位 0	**PCE**：数据校验和错误 (Payload checksum error) 置'1'时，表示 MAC 控制器计算的 TCP、UDP 或 ICMP 校验与接收到帧的 TCP、UDP 或 ICMP 的校 验和域值不相符。在接收到以太网帧的数据长度和 IPv4 或 IPv6 数据包长度域的值不符时，该位 也会置'1'。

下表显示了位5、7、0取值的含义

表 24-9 接收描述符0

位 5： 帧类型	位 7： 校验和错误	位 0： 数据校验和错误	帧状态
0	0	0	IEEE802.3 类型帧(长度域值小于 0x0600)
1	0	0	IPv4/IPv6 类型帧，未检测到校验和错误
1	0	1	IPv4/IPv6 类型帧，检测到数据校验和错误(PCE 位)
1	1	0	IPv4/IPv6 类型帧，检测到 IP 报头校验和错误(IPC CE 位)
1	1	1	IPv4/IPv6 类型帧，检测到 IP 报头校验和错误和数据校验和错误
0	0	1	IPv4/IPv6 类型帧，未检测到 IP 报头校验和错误；由于不支持的数据格式，未执行数据校验和检测
0	1	1	帧类型即不是 IPv4 也不是 IPv6(校验和模块不执行校验和检验)
0	1	0	保留

- RDES1：接收描述符字1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DIC	保留		RBS2												
rw	res		rw												

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RER	RCH	保留	RBS1												
rw	res		rw												

位 31	**DIC**：关闭接收完成中断 (Disable interrupt on completion) 置'1'时，在帧接收完成的时候，对于当前描述符指示的帧，不设置 ETH_DMASTS 寄存器的 RECV 位(位 6)。这样也就不会发生 RECV 位触发的中断。
位 30:29	保留
位 28:16	**RBS2**：接收缓存 2 大小 (Receive buffer 2 size) 这些位的值给出接收缓存 2 以字节为单位的大小。即使 RDES3 的值(接收缓存 2 地址指针)没有与 总线宽度对齐，根据总线宽度是 32、64 或者 128 位，缓存大小必须分别是 4、8 或者 16 的倍数， 否则缓存操作会造成不可预料的后果。如果 RDES1[14]位为'1'，这些位取值无意义。
位 15	**RER**：接收描述符环形结构结尾 (Receive end of ring) 置'1'时，该位表示当前描述符是描述符队列中的最后一个。DMA 会回到描述符队列基地址去取 下一个描述符，使描述符队列形成环形结构。
位 14	**RCH**：第二地址链表 (Second address chained) 置'1'时，该位表示描述符里的第二地址是下一个描述符的地址，而不是第二个缓存的地址。该 位为'1'时，RBS2(TDES1[28:16])的值可以忽略。RDES0[15]的作用优先于本位(RDES0[14])。
位 13	保留
位 12:0	**RBS1**：接收缓存 1 大小 (Receive buffer 1 size) 这些位的值表示接收缓存 1 以字节为单位的大小。即使 RDES3 的值(接收缓存 2 地址指针)没有与 总线宽度对齐，根据总线宽度是 32、64 或者 128 位，缓存大小必须分别是 4、8 或者 16 的倍数， 否则缓存操作会造成不可预料的后果。如果这些位取值为 0 ，DMA 忽略接收缓存 1 ，按照 RDES[14]的取值，直接取接收缓存 2 或者下一个描述符。

- RDES2：接收描述符字2

RDES2包含描述符第一个数据缓存的地址指针，或者时间戳数据。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RBAP1/RTSL															
rw															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RBAP1/RTSL															
rw															

位 31:0	**RBAP1/RTSL**：接收缓存 1 地址指针/接收帧时间戳低位 (Receive buffer 1 address pointer /Receive frame time stamp low) 这些位有 2 个功能：应用程序用它们表示 DMA 数据在内存中的位置。等到数据接收完后，DMA 可以用它们存放帧的时间戳。 **RBAP1**：当程序把描述符传递给 DMA 的时候(RDES0 的 OWN 位为'1'时)，这些位表示缓存 1 的 物理地址。不限制缓存区地址的对齐方式，除非发生下列情况：RDES2 指向的缓存用来存放帧 的第一个分块。因为 DMA 在接收帧起始的时候，会对 RDES2 的[3/2/1:0]位(总线宽度 128，64，32 位对应 RDES2[3:0]，[2:0]，[1:0])写 0，但帧数据还是移位存放到指针实际指向的地址。如果 RDES2 指向的缓存存放的是帧中间或者结尾的分块。DMA 会忽略 RDES2[3/2/1:0]的值。 **RTSL**：在 DMA 把 RDES0 的 OWN 位清'0'之前，DMA 会把帧的时间戳低 32 位写入这些位，覆盖掉原来的 RBAP1 值。只有在使能时间戳功能和 LS 位为'1'(表示缓存存放的是帧的最后分块)时，DMA 才会在这些位写入时间戳。

- RDES3：接收描述符字3

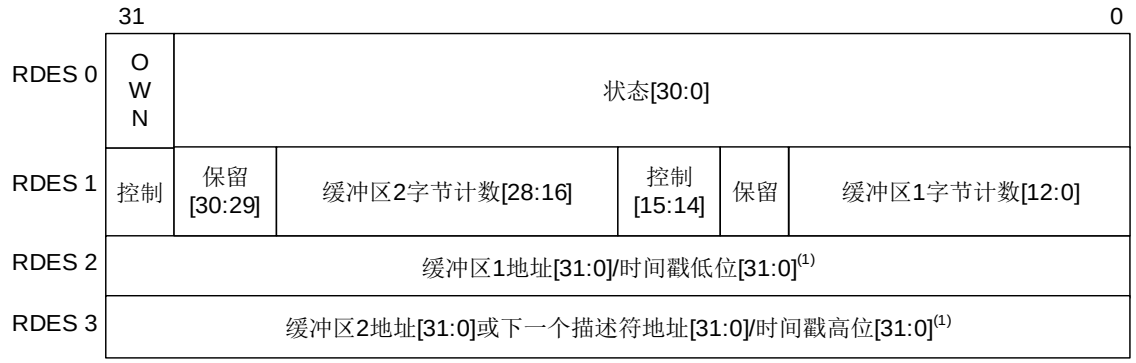
RDES3包含描述符第二个数据缓存的地址指针或者指向下一个描述符，或者时间戳数据。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BAP2/RTSH															
rw															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RBAP2/RTSH															
rw															

位 31:0	**RBAP2/RTSH**：接收缓存 2 地址指针(或者下一个描述符地址指针)/接收帧时间戳高位 (Receive buffer 2 address pointer (next descriptor address) / Receive frame time stamp high) 位 31:0 这些位有 2 个功能：程序用它们表示 DMA 数据在内存中的位置。等到数据接收完后，DMA 可以用它们存放帧的时间戳 **RBAP2**：当应用程序把描述符传递给 DMA 的时候(RDES0 的 OWN 位为'1'时)，在描述符队列为环形结构的时候，这些位表示缓存 2 的物理地址。如果第二地址链表位(RDES0[14])为'1'，并且下一个描述符存在，则这些位指向下一个描述符的物理地址。如果 RDES0[14]位为'1'，描述符地址指针必须和总线宽度对齐(总线宽度分别为 128、64 或者 32 位时，RDES3[3,2 或者 1:0]=0)。在 RDES0[14]为'0'时，RDES3 的取值没有限制，除非发生下列情况：RDES3 指向的缓存用来存放帧的第一个分块，而 DMA 把 RDES3 的值作为缓存的地址。如果 RDES3 指向的缓存存放的是帧中间或者结尾的分块，DMA 会忽略 RDES2[3/2/1:0]的值(依照总线宽度分别为 128、64 或 32 位时)。 **RTSH**：在 DMA 把 RDES0 的 OWN 位清'0'之前，DMA 会把帧的时间戳高 32 位写入这些位，覆盖掉原来的 RBAP2 值。只有在时间戳使能和 LS 位为'1'(表示缓存存放的是帧的最后分块)时，DMA 才会在这些位写入时间戳。

带IEEE1588时间戳的接收DMA描述符格式

图 24-34 IEEE1588时间戳使能时的接收描述符格式



DMA在把RDES0的OWN位清’0’之前，会用时间戳的值更新RDES2和RDES3：RDES2更新为时间戳的低32位(该域在”RDES2：接收描述符字2”中称为RTSL)，RDES3更新为时间戳的高32位(该域在”RDES3：接收描述符字3”中称为RTSH)。

- RDES0：接收描述符字0

  参见”RDES0：接收描述符字0”。

- RDES1：接收描述符字1

  参见”RDES1：接收描述符字1”。

- RDES2：接收描述符字2

  下表描述了在接收帧时间戳功能使能时，DMA关闭接收描述符后，RDES2域的不同含义。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RTSL															
rw															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RTSL															
rw															

位 31	**RTSL**：接收帧时间戳低位 (Receive frame time stamp low) DMA 会把帧的时间戳低 32 位写入这些位。只有在 LS 位(RDES0[8])为’1’(表示缓存存放的是帧的最后一部分)时，DMA 才会把时间戳写入这些位。如果本域和 RDES3 的 RTSH 域值为全”1”，则表示时间戳信息损坏。

- RDES3：发送描述符字3

  下表描述了在接收帧时间戳功能使能时，DMA关闭接收描述符后，RDES3域的不同含义。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RTSH															
rw															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RTSH															
rw															

位 31	**RTSH**：发送帧时间戳高位 (Receive frame time stamp high) DMA 会把帧的时间戳高 32 位写入这些位。只有在 LS 位(RDES0[8])为’1’(表示缓存存放的是帧的最后一部分)时，DMA 才会把时间戳写入这些位。如果本域和 RDES2 的 RTSL 域值为全”1”，则表示时间戳信息损坏。

24.6.9 DMA中断

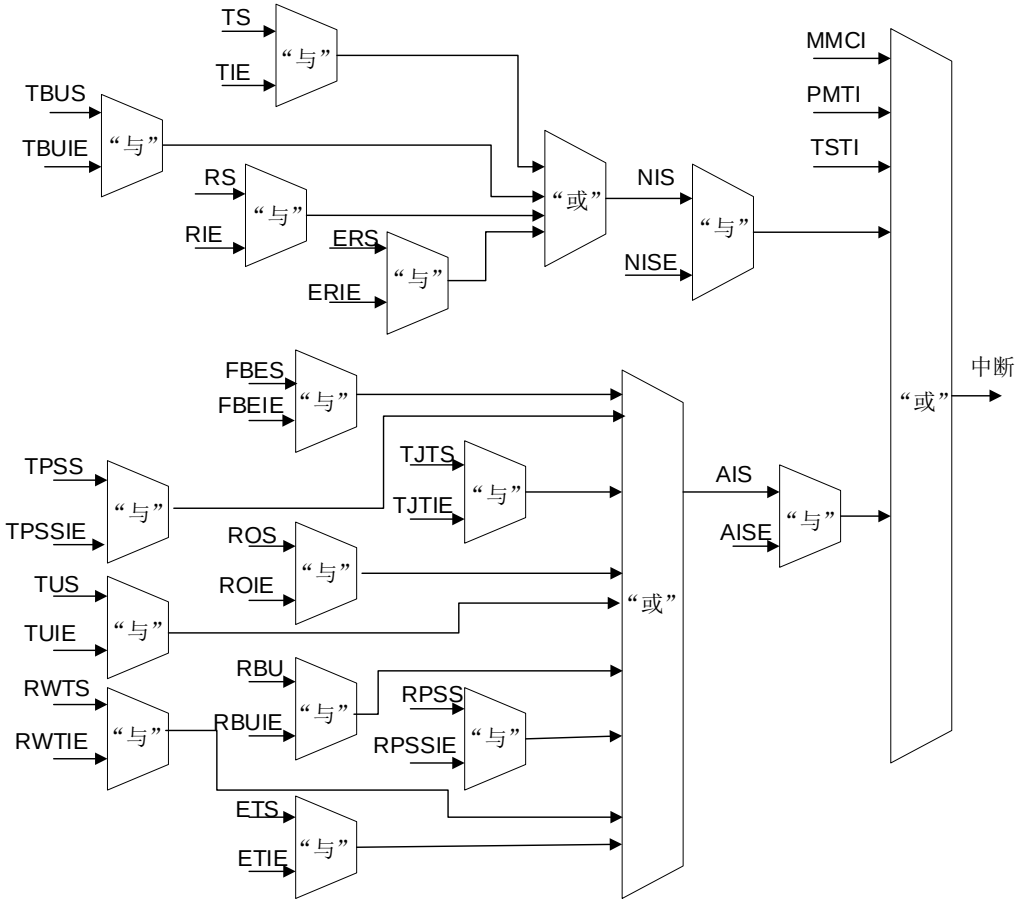
有多种事件可以产生中断。ETH_DMASTS包含了所有能引发中断的位，而ETH_DMAIE寄存器包含了每一个能产生中断事件的中断使能位。

中断有2组，见ETH_DMASTS寄存器描述，可分为正常组和异常组。通过对相应位写’1’，可以清 除中断。如果清除了一个组里所有使能的中断，相应地也清除了这个组的汇总位。如果中断是由MAC控制器引发的，那么ETH_DMASTS的TST位或者PMT位会置高。

中断事件不会排队，即如果发生某一中断，在响应它之前不会再发生另一个中断。例如，接收中断位(ETH_DMASTS[6])表示一个或者多个帧已经接收到AT32F407xx的缓存里，中断处理程序需要从头到尾检索所有DMA占有的描述符。

同一时间多个能引发中断的事件只能产生一个中断。中断处理程序需要检查ETH_DMASTS寄存器来判断中断的来源。在正确地清除掉ETH_DMASTS寄存器的相应位以后，除非再发生一个中断事件，否则不会再产生中断。例如控制器产生接收中断(ETH_DMASTS[6]为’1’)，中断处理程序读出ETH_DMASTS寄存器后，又发生了接收缓存不可用(ETH_DMASTS[7]为’1’)事件；在首先清除了接收中断后，由于接收缓存不可用中断依然有效并需要处理，会再产生一个新的中断。

图 24-35 DMA中断示意图



24.7 以太网中断

以太网控制器有2个中断向量，一个用于以太网正常操作，另一个用于映射到EXTI线19的以太网唤醒事件(检测唤醒帧或者Magic Packet)。

第一个中断向量用于由MAC和DMA产生的中断，详见第24.5.4 MAC中断和24.6.9 DMA中断中断”。

第二个中断向量用于PMT模块在唤醒事件时产生的中断。唤醒事件映射到EXTI线19上，它可以使AT32F407xx微控制器退出低功耗模式，并产生中断。

在以太网唤醒事件映射到EXTI线19时，一旦发生唤醒事件，如果使能了MAC的PMT中断，又使能了EXTI线19检测到上升沿时产生中断，那么2个中断会同时发生。

以太网外设有一个看门狗定时器（见ETH_DMARSWTR寄存器），可用于灵活控制RECV位(ETH_DMASTS[6])的置位。如果对看门狗写入非0的值，而接收到帧的描述符里没有使能接收中断(RDES1[31]=0)，在接收DMA把帧接收到系统内存里的时候，看门狗定时器就被激活，同时不设置RECV位为’1’。在看门狗定时器递减为0时，才设置RECV位为’1’，如果使能了相应的RXIE中断，则发生中断。看门狗定时器递减为0之前，如果接收到一个帧，帧描述符使能了接收中断，RECV位被置为’1’

同时关闭看门狗定时器。

注意： *读PMT控制和状态寄存器会自动清除唤醒帧接收和Magic Packet接收中断标志位。然而，由于这些标志位所在的寄存器位于CLK_RX域，在应用程序看到这些标志位被清除之前可能会有可观的延迟。如果RX时钟比较慢的话(如10M位模式下)而AHB总线时钟频率比较高的情况下，这个延迟会特别长。*

由于PMT向CPU产生的中断请求也在RX_CLK域的同一个寄存器里，因此在读出PMT_CSR寄存器后，CPU有可能会错误地再调用一次中断程序。因此，需要应用程序在中断里查询唤醒帧接收和Magic Packet接收标志位，直到它们变成0再退出中断服务程序。

24.8 以太网寄存器

可以以字节(8位)、半字(16位)和字(32位)的形式对本外设的寄存器进行访问。下表给出了以太网寄存器映像及其复位值。

表 24-10 以太网寄存器映像及其复位值

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	ETH_MACCTRL	保留								WDOG	JAD	保留		FG			CSD	保留	FESM	ROD	LBM	DM	IP4C	RO	保留	AUPC	BOL		DC	TE	RE	保留	
	复位值									0	0			0	0	0	0		0	0	0	0	0	0		0	0	0	0	0	0		
0x04	ETH_MACFRMF	RALL	保留																				HOPF	SADDR	SADDR	PCF			BFD	PALLM	DADDR	HM	HU	PM
	复位值	0																					0	0	0	0	0	0	0	0	0	0	0	0
0x08	ETH_MACHTH	HTH[31:0]																															
	复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	ETH_MACHTL	HTL[31:0]																															
	复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	ETH_MACMIIADDR	保留																PA					MII					保留	CLKR			MIIW	MIIB
	复位值																	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0
0x14	ETH_MACMIIDT	保留																MIID															
	复位值																	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
0x18	ETH_MACFCTRL	PTM																保留								ZQPD	保留	PLT		UPFD	RFCEN	TFCEN	FCB/BPA
	复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0									0		0	0	0	0	0	0
0x1C	ETH_MACVLT	保留															VLTC	VLAN															
	复位值																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x28	ETH_MACRWFF	帧过滤寄存器 0\帧过滤寄存器 1\……\帧过滤寄存器 7																															
	复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x2C	ETH_ MACPMTCTRLSTS	WFFPR	保留																					GLBU	保留		WFR	MPR	保留			WFEN	MPEN	PD
	复位值	0																						0			0	0				0	0	0
0x38	ETH_MACISTS	不可用																保留						TSTS	保留		MMCTS	MMCRS	MMCS	PMTS	保留		
	复位值																							0			0	0	0	0			
0x3C	ETH_MACIM	不可用																保留						TSTINTM	保留					PMTINTM	保留		
	复位值																							0						0			
0x40	ETH_MACA0H	MO	保留															MACADDR0H															
	复位值	1																1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x44	ETH_MACA0L	MACADDR0L																															
	复位值	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x48	ETH_MACA1H	AEN	SAD	MBCTRL[6:0]					保留									MACADDR1H															
	复位值	0	0	0	0	0	0	0										1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x4C	ETH_MACA1L	MACADDR1L																															
	复位值	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x50	ETH_MACA2H	AE	SA	MBCTRL[6:0]					保留									MACADDR2H															
	复位值	0	0	0	0	0	0	0										1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x54	ETH_MACA2L	MACADDR2L																															
	复位值	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x58	ETH_MACA3H	AEN	SAD	MBCTRL[6:0]					保留									MACADDR3H															
	复位值	0	0	0	0	0	0	0										1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x5C	ETH_MACA3L	MACADDR3L																															
	复位值	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x100	ETH_MMCCTRL	保留																												MCN	RST	CSR	CNT
	复位值																													0	0	0	0
0x104	ETH_MMCRI	保留														RGUF	保留											RFAE	RFCE	保留			
	复位值															0												0	0				
0x108	ETH_MMCTI	保留							TGF	保留								TGFMSC	TGFSC	保留													
	复位值								0									0	0														
0x10C	ETH_MMCRIM	保留														RGUFM	保留											RFAEM	RFCEM	保留			
	复位值															0												0	0				
0x110	ETH_MMCTIM	保留							TGFM	保留								TGFMSCM	TGFSCM	保留													
	复位值								0									0	0														
0x14C	ETH_MMCTFSCC	TGFSCC																															
	复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x150	ETH_MMCTFMSCC	TGFMSCC																															
	复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x168	ETH_MMCTFCNT	TGFC																															
	复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x194	ETH_MMCRFCECNT	RFCEC																															
	复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x198	ETH_MMCRFAECNT	RFAEC																															
	复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1C4	ETH_MMCRGUFCNT	RGUFC																															
	复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x700	ETH_PTPTSCTRL	保留													TSPFFMAE	TSCNT	TSSMRME	TSSEME	TSSIPV4FE	TSSIPV6FE	TSSPTPOEFE	TSPTPPSV2E	TSSSR	TSSARFE	保留	TTSARU	TSITE	TSSTU	TSSTI	TSFCU	TSE			
	复位值														0	0	0	0	1	0	0	0	0	0		0	0	0	0	0	0			
0x704	ETH_PTPSSINC	保留																								STSINC							
	复位值																									0	0	0	0	0	0	0	0
0x708	ETH_PTPTSH	STS[31:0]																															
	复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

0x70C	ETH_PTPTSL	STPENS	STSS																													
	复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x708	ETH_PTPTSH	STS[31:0]																														
	复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x70C	ETH_PTPTSL	STSS																														
	复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x710	ETH_PTPTSHUD	TSUS																														
	复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x714	ETH_PTPTSLUD	TSUPNS	TSUSS																													
	复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x718	ETH_PTPTSAD	TSA																														
	复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x71C	ETH_PTPTTH	TTSH																														
	复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x720	ETH_PTPTTL	TTSL																														
	复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x728	ETH_PTPTSSR	保留																													TSTTR	TSSO
	复位值																														0	0
0x72c	ETH_PTPPPSCR	保留																													PPSFREQ	
	复位值																														0	
0x1000	ETH_DMABM	保留							AAB	FPB	USP	RDPBL						FB	RTPR	PBL						EDF	DSL				DA	SWR
	复位值								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1004	ETH_DMATPD	TPD																														
	复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1008	ETH_DMARPD	RPD																														
	复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x100C	ETH_DMARDLADDR	SRL																														
	复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1010	ETH_DMATDLADDR	STL																														
	复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1014	ETH_DMASTS	保留	TST	PMT	MMC	保留	EB		TP			RP			NIS	AIS	ER	FBE	保留	ET	RWT	RPS	RBU	RECV	TU	RO	TJT	TBU	TPS	TX		
	复位值		0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1018	ETH_DMAOPM	保留				DTCEFD	RSFD	DFRFD	保留	TXSF	FTXF	保留			TXTC			STXE	保留				FEF	FUGF	保留	RXTC			OPSF	SRXE	保留	
	复位值					0	0	0		0	0				0	0	0	0					0	0		0	0	0	0	0			
0x101C	ETH_DMAIE	保留													NISE	AISE	ERIE	FBEIE	保留	ETIE	RXWTIE	RXPSIE	RXBUIE	RXIE	TUIE	ROIE	TXJTIE	TXBUIE	TXPSIE	TXIE		
	复位值														0	0	0	0		0	0	0	0	0	0	0	0	0	0	0			
0x1020	ETH_DMAMFBOCNT	保留		OFOC	MFA										OMFC	MFC																
	复位值			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

0x1048	ETH_DMACTD	HTDAP																														
	复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x104C	ETH_DMACRD	HRDAP																														
	复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1050	ETH_DMACTBADDR	HTBAP																														
	复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1054	ETH_DMACRBADDR	HRBAP																														
	复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

24.8.1 MAC寄存器描述

以太网MAC设置寄存器(ETH_MACCTRL)

地址偏移：0x0000

复位值：0x0000 8000

MAC设置寄存器是MAC的工作模式寄存器。它定义了接收和发送的工作模式。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留								WDOGD	JAD	保留		FG			CSD
res								rw	rw	res		rw			rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留	FESM	ROD	LBM	DM	IP4CO	RD	保留	AUPCS	BOL		DC	TE	RE	保留	
res	rw	rw	rw	rw	rw	rw	res	rw	rw		rw	rw	rw	res	

位 31:24	保留
位 23	**WDOGD**：关闭看门狗 (Watchdog disable) 1：MAC 关闭接收端上的看门狗定时器，并且能够接收最多达 16384 字节的帧。 0：MAC 允许接收不超过 2048 字节的帧，超过 2048 字节的帧会被切断。
位 22	**JAD**：不检测啰嗦 (Jabber disable) 1：MAC 关闭发送端上的啰嗦定时器，并且能够发送最多达 16384 字节的帧。 0：如果应用程序试图发送超过 2048 字节的帧，MAC 会关断发射器。
位 21:20	保留
位 19:17	**FG**：帧间间隙 (interframe gap) 这些位控制了发送 2 个帧之间的最短间隙 000：96 位时间； 001：88 位时间； 010：80 位时间； … 111：40 位时间。 *注意：在半双工模式下，FG 可设定的最小值是 64 位时间(FG = 100)，不允许取更小的值。*
位 16	**CSD**：关闭载波侦听功能 (Carrier sense disable) 1：在半双工模式下，MAC 的发送器在发送帧过程中忽略 MII 的 CSR 信号，发送过程中载波丢失或者没有载波都不会报错。 0：MAC 在发送过程中如果发生上述情况会报错，甚至放弃发送。
位 15	保留
位 14	**FESM**：快速以太网 (Fast Ethernet speed) 该位表示快速以太网(MII)模式的速度： 0：10 Mbit/s 1：100 Mbit/s
位 13	**ROD**：关闭自接收功能 (Receive own disable) 1：MAC 在半双工模式下不接受帧。 0：MAC 在发送时接收所有来自 PHY 的数据包。 该位在全双工模式下无意义。

位 12	**LBM**：自循环模式 (Loopback mode) 置'1'时，MAC 的 MII 端工作在自循环模式。自循环模式需要接收时钟输入(RX_CLK)来正常工作，这是因为在内部，发送时钟没有循环返回。
位 11	**DM**：双工模式 (Duplex mode) 置'1'时，MAC 工作在全双工模式，可以同时进行收和发。
位 10	**IP4CO**：IPv4 校验和机制 (IPv4 checksum offload) 1：使能接收到 IPv4 帧的数据 TCP/UDP/ICMP 报头的校验和检验。)。 0：关闭接收端的校验和检测功能，相应的 PCE 和 IPHCE 标志位值总是'0'
位 9	**RD**：不尝试重试 (Retry disable) 1：MAC 只会尝试发送 1 次。如果在 MII 上发生冲突，MAC 会放弃发送，并在发送状态信息里报告冲突过多错误。 0：MAC 会在发生冲突后按照 BOL 位的设定，在一定时间后重发。 注：该位只在半双工模式下有效。
位 8	保留
位 7	**AUPCS**：自动填充/CRC 剥离 (Automatic pad/CRC stripping) 1：只有在接收到帧的长度小于等于 1500 字节时，MAC 会去除帧的填充字节和 CRC 域。所有长度大于等于 1501 字节的帧，MAC 都会保留帧的填充字节和 CRC 域，并转发给应用程序。 0：MAC 会转发所有接收到的帧，而不改变帧的内容。
位 6:5	**BOL**：退后限制 (Back-off limit) 退后限制定义了 MAC 在发送发生冲突后，重发前等待的随机时间间隙数目(在 10/100M 位/s 模式下为 512 位时间)。 *注意：这些位只在半双工模式下有效。* 00：k = min(n, 10)； 01：k = min(n, 8)； 10：k = min(n, 4)； 11：k = min(n, 1)。 其中 n = 重发等待时间间隙数目，r 是从 0≤ r ≤2^k之间的随机数。
位 4	**DC**：顺延检验 (Deferral check) 1：MAC 使能顺延检验功能。在 10/100M 位/s 模式下发送延迟 24288 位时间后，MAC 中止发送，并在发送状态信息里置顺延过久错误标志位。在帧准备发送时就开始顺延检验，但是如果检测到有效的 CRS(载波侦听)信号则不会开始。顺延时间不会累计，假设顺延计时到 10000 位时，然后开始发送，但是发现冲突，退步重发，在重发完成以后会开始重新顺延计时，这时顺延计数器重置为 0，重新启动顺延计时。 0：MAC 关闭顺延检验功能。MAC 会延迟发送直到 CRS 信号失效。 该位只在半双工模式下有效。
位 3	**TE**：使能发送器 (Transmitter enable) 1：MAC 使能在 MII 上的发送状态机。 0：MAC 在完成发送当前帧后，关闭发送状态机，不再发送任何帧。
位 2	**RE**：使能接收器 (Receiver enable) 1：MAC 使能在 MII 上的接收状态机。 0：MAC 在接收完当前正在接收的帧后，关闭接收状态机，不再接收任何帧。
位 1:0	保留

以太网MAC帧过滤器寄存器(ETH_MACFRMF)

地址偏移：0x0004

复位值：0x0000 0000

MAC帧过滤器寄存器包含了接收帧的过滤器控制位。一些控制位控制了MAC的地址检验模块，形成了第一层次的地址过滤。另一些控制位形成的第二层次过滤是对将要输入的帧而言的，进行其他过滤选项如允许坏帧通过、允许控制帧通过等。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RALL	保留														
rw	res														

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留					HOPFD	SADDDF	SADDDF	PCF		BFD	PALLM	DADDDF	HM	HU	PM

	res	rw
位 31	**RALL**：接收全部 (Receive all) 1：MAC 会把所有接收到的帧转发给应用程序，不管它们是否通过了地址过滤器。源/目的地址 过滤器的结果(通过或者未通过)，会反映在更新接收状态信息的相应标志位。 0：MAC 只会把接收到的、通过了源/目的地址过滤器的帧转发给应用程序。	
位 30:11	保留	
位 10	**HOPF**：HASH 或者完美过滤器 (Hash or perfect filter) 1：地址过滤器会根据 HM 位和 HU 位的取值，来判定帧通过的条件：是需要符合 HASH 过滤器， 还是需要符合完美过滤器。 0：如果 HM 位或者 HU 位置 1，只要帧符合 HASH 过滤器，就能通过地址过滤器。	
位 9	**SADDRF**：源地址过滤器 (Source address filter) MAC 会把接收到帧的源地址域与使能的源地址寄存器值比较。如果符合，会设置接收状态信息里的相应标志位。 1：如果帧不能通过源地址过滤器，MAC 会丢弃该帧。 0：MAC 转发所有接收到的帧到应用程序，过滤结果会反映在接收状态信息里的相应标志位。	
位 8	**SADDRIF**：源地址过滤结果颠倒 (Source address inverse filtering) 1：地址检验模块工作在源地址过滤结果颠倒模式。所有源地址符合源地址寄存器的帧会被标记为未通过。 0：所有源地址不符合源地址寄存器的帧会被标记为未通过。	
位 7:6	**PCF**：通过控制帧 (Pass control frames) 这些位设置了转发所有控制帧(包括单播和多播 PAUSE 帧)给应用程序的选项。注意是否处理 PAUSE 控制帧取决于 RFCEN 位(ETH_MACFCTRL[2])的值。 00 或者 01：MAC 不转发任何控制帧给应用程序； 10：MAC 转发所有的控制帧给应用程序，包括那些没能通过地址过滤器的控制帧； 11：MAC 转发通过地址过滤器的控制帧给应用程序。	
位 5	**BFD**：不接收广播帧 (Broadcast frames disable) 1：地址过滤器过滤掉所有收到的广播帧； 0：所有收到的广播帧都能通过地址过滤器。	
位 4	**PALLM**：通过全部多播帧 (Pass all multicast) 1：所有的带多播目的地址的帧(地址第一位为'1')都能通过过滤器； 0：多播帧过滤与否取决于 HM 位的值。	
位 3	**DADDRIF**：目的地址过滤结果颠倒 (Destination address inverse filtering) 1：对多播帧和单播帧，地址检验模块工作在目的地址过滤结果颠倒模式； 0：过滤器正常工作。	
位 2	**HM**：多播 HASH (Hash multicast) 1：MAC 根据 HASH 列表对接收到的多播帧进行目的地址过滤； 0：MAC 对接收到的多播帧进行目的地址完美过滤，即把帧的目的地址域和目的地址寄存器的 设定值比较。	
位 1	**HU**：单播 HASH (Hash unicast) 1：MAC 根据 HASH 列表对接收到的单播帧进行目的地址过滤； 0：MAC 对接收到的单播帧进行目的地址完美过滤，即把帧的目的地址域和目的地址寄存器的 设定值比较。	
位 0	**PM**：混杂模式 (Promiscuous mode) 置'1'时，无论接收到帧的目的地址和源地址为何，所有的帧都能通过地址过滤器。此时，接收 状态信息的目的地址/源地址错误位总是为'0'。	

以太网MAC Hash列表高寄存器(ETH_MACHTH)

地址偏移：0x0008

复位值：0x0000 0000

64位的HASH列表可以用来进行成组的地址过滤。进行HASH过滤时，帧的目的地址输入CRC逻辑，并取CRC值的高6位检索HASH列表。32位的CRC计算多项式如下，详见24.5.3 MAC帧的接收。

$G(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$

这个CRC值的最高位(MSB)决定使用哪个寄存器(Hash列表高寄存器或者Hash列表低寄存器)，剩下的5位决定检查寄存器里的哪一位。HASH值为0b00000的对应选中寄存器的位0，HASH值为0b11111的对应选中寄存器的位31。

例如，输入帧的目的地址是0x1F52 419C B6AF(MII接口最先接收到的字节是0x1F)，则计算出的CRC值高6位是0x2C，因此用HTH的位12进行HASH过滤。如果输入帧的目的地址是0xA00A 9800 0045，则算出的HASH值是0x07，用HTL的位7来进行HASH过滤。

如果HASH 寄存器对应的位是’1’ ，则MAC 就接受这个帧；否则，MAC就丢弃这个帧。如果ETH_MACFRMF寄存器的PALLM位(通过全部多播帧)为’1’，那么无论HASH过滤结果如何，MAC接 受所有的多播帧。

Hash列表高寄存器包含了多播HASH列表的高32位。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
HTH															
rw															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HTH															
rw															

位 31	**HTL**：HASH 列表低 (Hash table low) 这些位是 HASH 列表的低 32 位。

以太网MAC Hash列表低寄存器(ETH_MACHTL)

地址偏移：0x000C

复位值：0x0000 0000

Hash列表低寄存器包含了多播HASH列表的低32位。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
HTL															
rw															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HTL															
rw															

位 31	**HTL**：HASH 列表低 (Hash table low) 这些位是 HASH 列表的低 32 位。

以太网MAC MII地址寄存器(ETH_MACMIIADDR)

地址偏移：0x0010

复位值：0x0000 0000

MII地址寄存器通过接口控制外部PHY的管理信号。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															
res															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PA					MII					CLKR				MIIW	MIIB
rw					rw					rw				rw	rw

位 31:16	保留
位 15:11	**PA**：PHY 地址 (PHY address) 这些位的值表示了 32 个可能的 PHY 地址中，MII 想要访问的 PHY 地址
位 10:6	**MII**：MII 寄存器 (MII register) 这些位的值选择了要访问 PHY 的哪个 MII 寄存器。

位 5:2	**CLKR**：时钟范围 (Clock range) CLKR 值根据 HCLK 的频率来决定 MDC 的时钟频率 取值 HCLK MDC 时钟 0000 60~100 MHz HCLK/42 0001 100~150 MHz HCLK/62 0010 20~35 MHz HCLK/16 0011 35~60 MHz HCLK/26 0100 150~240 MHz HCLK/102 0101, 0110, 0111 保留 -
位 1	**MIIW**：MII 写 (MII write) 1：表示将要使用 MII 的数据寄存器对 PHY 进行写操作； 0：表示对 PHY 进行读操作，数据在 MII 的数据寄存器中。
位 0	**MIIB**：MII 忙 (MII busy) 在写 ETH_MACMIIADDR 寄存器和 ETH_MACMIIDT 寄存 器之前，该 位应当读出为 ’0’ 。在写 ETH_MACMIIADDR 寄存器时，该位也应为’0’。在访问 PHY 的时候，该位由应用程序置’1’，表示正在对 PHY 的进行读或者写操作。对 PHY 写操作时，必须保持 ETH_MACMIIDT 寄存器的值(MII 数据)，直到将该位 MAC 清’0’。对 PHY 读操作时，在 MAC 清’0’该位后，ETH_MACMIIDT 寄存器的值才是有效的。只有该位为’0’时，才能写 ETH_MACMIIADDR 寄存器(MII 地址)。

以太网MAC MII数据寄存器(ETH_MACMIIDT)

地址偏移：0x0014

复位值：0x0000 0000

MII数据寄存器存放要写入PHY寄存器的值，待写PHY寄存器的位置由ETH_MACMIIADDR寄存器设置。它同时也保存从PHY的寄存器读出的值，被读PHY寄存器的位置由ETH_MACMIIADDR寄存器设置。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															
res															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MIID															
rw															

位 31:16	保留
位 15:0	**MIID**：MII 数据 (MII data) 这些位包含了对 PHY 进行一次读操作后，读出的 16 位数据。或者在对 PHY 写操作前，将要写入 的 16 位数据。

以太网MAC流控寄存器(ETH_MACFCTRL)

地址偏移：0x0018

复位值：0x0000 0000

流控寄存器控制MAC生成和接收控制(PAUSE命令)帧。对寄存器的忙位写’1’会使MAC发送 PAUSE控制帧。控制帧的格式遵守802.3x规范的定义，寄存器的PTM域(Pause时间)用来作为控制帧Pause时间域的值。在PAUSE帧发送到电缆上前，忙位会保持为’1’。应用程序在写寄存器之前需要保证忙位已经清’0’。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PTM															
rw															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留								ZQPD	保留	PLT		UPFD	RFCEA	TFCEA	FCB/BPA
res								rw	res	rw		rw	rw	rw	rc_w1/rw

位 31:16	**PTM**：PAUSE 时间 (Pause time) 这些位的值用来作为控制帧 Pause 时间域的值。如果 Pause 时间设置为两次同步于 MII 时钟域， 那么对寄存器的连续 2 次写操作之间间隔至少要有 4 个目标域时钟周期。

位 15:8	保留
位 7	**ZQPD**：关闭零值 PAUSE 功能 (Zero-quanta pause disable) 1：在撤销 FIFO 层流控信号时，关闭自动零值 Pause 控制帧的自动生成； 0：正常操作，打开零值 Pause 控制帧自动生成功能。
位 6	保留
位 5:4	**PLT**：PAUSE 低阈值 (Pause low threshold) 这些位设置了自动重发 Pause 帧的定时器阈值。这个阈值应当小于位[31:16]定义的 Pause 时 间。例如，PTM = 100H(256 个时间间隙)，PLT = 01，那么如果在第一个 Pause 帧发出 228(256-28)个时间间隙后，自动重发第二个 Pause 帧。 取值 阈值 00 Pause 时间 – 4 时间间隙 01 Pause 时间 – 28 时间间隙 10 Pause 时间 – 144 时间间隙 11 Pause 时间 – 256 时间间隙 时间间隙是指 MII 接口发送 512 位(64 字节)数据所需要的时间
位 3	**UPFD**：单播 Pause 帧检测 (Unicast pause frame detect) 1：MAC 在检测带唯一多播地址的 Pause 帧的基础上，还会检测单播地址符合 ETH_MACA0H 寄存器和 ETH_MACA0L 寄存器设定值的 Pause 帧； 0：MAC 只接收带 802.3x 规范定义的唯一地址的 Pause 帧。
位 2	**RFCEN**：接收流控使能 (Receive flow control enable) 1：MAC 解析接收到的 Pause 帧，并关闭发送器一段特定的时间； 0：MAC 不解析 Pause 帧。
位 1	**TFCEN**：发送流控使能 (Transmit flow control enable) 在全双工模式下—— 1：MAC 使能发送流控，可以发送 Pause 帧；0：MAC 关闭发送流控，不发送 Pause 帧。 在半双工模式下—— 1：MAC 使能背压功能；0：MAC 关闭背压功能。
位 0	**FCB/BPA**：流控忙/背压激活 (Flow control busy/back pressure activate) 在全双工模式下，设置该位为'1'可以生成 Pause 控制帧。在半双工模式下，如果 TFCEN 位为'1'， 则设置该位为'1'可以激活背压功能。 在全双工模式下，写流控寄存器之前需要确认该位读出为'0'。应用程序要对该位写'0'，来生成 Pause 控制帧。在发送控制帧的过程中，该位始终为'1'，表示正在进行发送。在 Pause 控制帧发送完成以后，MAC 将该位重置为' 0' 。在 MAC 把该位清' 0' 之前，不允许写流控寄存器。 在半双工模式下，设置该位为' 1' 时(并且 TFCEN 位也为' 1')，MAC 激活背压功能。在背压功能有效 时，如果 MAC 接收到新的帧，就会在发送端发送阻塞信号，导致冲突发生。MAC 设置成全双工模式时，背压(BPA)功能自动关闭。

以太网MAC VLAN标签寄存器(ETH_MACVLT)

地址偏移：0x001C

复位值：0x0000 0000

VLAN标签寄存器包含了用来识别VLAN帧的IEEE802.1Q VLAN标签。MAC把接收到帧的第13，14字节(长度/类型域)与0x8100比较，再把之后的2个字节和VLAN标签比较。如果比较相符，就把这个帧的接收状态信息的VLAN位置为' 1'。VLAN帧的最大合法帧长由基本帧的1518字节增加到1522字节。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															VLANT
res															r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VLTI															
rw															

位 31:17	保留

位 16	**VLTC**：12 位 VLAN 标签比较 (12-bit VLAN tag comparison) 1：使用 12 位而不是 16 位的 VLAN 标识符来进行过滤和比对。用 VLAN 标签域的位[11:0]和接收到 VLAN 标签帧的相应域比对。 0：接收到的 VLAN 帧第 15、16 字节的全部 16 位数据都用来与 VLTI 位比对。
位 15:0	**VLTI**：(接收到帧的)VLAN 标签标识符 (VLAN tag identifier (for receive frames)) 这些位包含了用来识别 VLAN 帧的 802.1Q VLAN 标签，用来和接收到 VLAN 帧的第 15、16 字节 进行比对。位[15:13]是用户优先级，位[12]是规范格式指示符(CFI)，位[11:0]是 VLAN 标签的 VLAN 标识符(VID)域。VLTC 位为’1’时，只用 VID(位[11:0])进行比对。 如果 VLTI 位的值是全’0’，MAC 不再比对检验 VLAN 帧的第 15、16 字节，只要接收到帧的类 型域值是 0x8100，都直接视为 VLAN 帧。

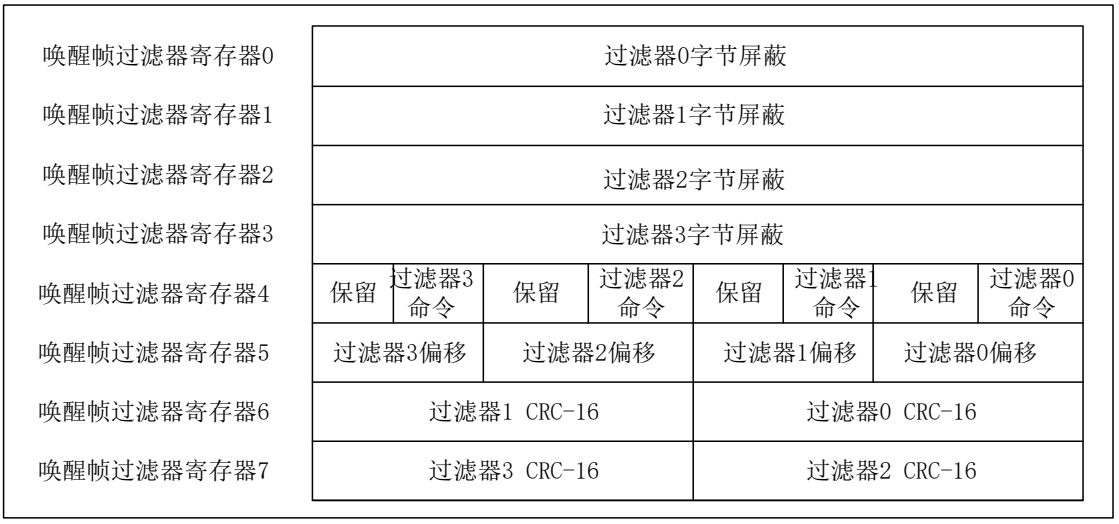
以太网MAC远程唤醒帧过滤器寄存器(ETH_MACRWFF)

地址偏移：0x0028

复位值：0x0000 0000

应用程序可用本地址对远程唤醒帧过滤器寄存器进行读写。该寄存器实质上是指向8个不透明的唤醒帧过滤器寄存器的指针。对该寄存器地址(偏移为0x0028)的8次连续写操作，可以写入全部8个唤醒帧过滤器寄存器；对该寄存器地址(偏移为0x0028)的8次连续读操作，可以读出全部8个唤醒帧过滤器寄存器。该寄存器包含第7个MAC地址的高16位。详见前述”远程唤醒帧过滤器寄存器”段落。

图 24-36 以太网MAC唤醒帧过滤器寄存器(ETH_MACRWFF)



以太网MAC PMT控制和状态寄存器(ETH_MACPMTCTRLSTS)

地址偏移：0x002C

复位值：0x0000 0000

ETH_MACPMTCTRLSTS寄存器设置并监控唤醒事件。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
WFFRP	保留														
rs	res														

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留						GLBU	保留		WFR	MPR	保留		WFE	MPE	PD
res						rw	res		rc_r	rc_r	res		rw	rw	rs

位 31	**WFFPR**：唤醒帧过滤器寄存器指针复位 (Wakeup frame filter register pointer reset) 置’1’时，会把远程唤醒帧过滤器寄存器指针复位为 0，该位在 1 个时钟周期后自动清’0’。
位 30:10	保留

位 9	**GLBU**：全局单播 (Global unicast) 置'1'时， 所有能通过 MAC 地址过滤器的单播帧，都被认为是唤醒帧。
位 8:7	保留
位 6	**WFR**：接收到唤醒帧 (Wakeup frame received) 置'1'时，表示由于接收到唤醒帧，产生了电源管理(PMT)事件。读本寄存器可以清'0'该位。
位 5	**MPR**：接收到 Magic Packet (Magic packet received) 置'1'时，表示由于接收到 Magic Packet 而产生了电源管理(PMT)事件。读本寄存器可以清'0'该 位。
位 4:3	保留
位 2	**WFEN**：唤醒帧使能 (Wakeup frame enable) 置'1'时，表示接收到唤醒帧时，允许产生电源管理(PMT)事件。
位 1	**MPEN**：Magic Packet 使能 (Magic Packet enable)置'1'时，表示接收到 Magic Packet 时，允许产生电源管理(PMT)事件。
位 0	**PD**：掉电 (Power down) 置'1'时，MAC 丢弃所有接收到的帧。在接收到 Magic Packet 或者唤醒帧时，该位自动清'0'，同 时关闭掉电模式。在该位清'0'后，MAC 向应用程序转发接收到的帧。只有在 WFEN 位(唤醒帧使 能)或者 MPEN 位(Magic Packet 使能)为'1'时，才能把该位置'1'。

以太网MAC中断状态寄存器(ETH_MACISTS)

地址偏移：0x0038

复位值：0x0000 0000

ETH_MACISTS寄存器能用来确定MAC中产生中断的事件。

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留						TSTS	保留		MMCTS	MMCRS	MMCS	PMTS	保留		
res						rc_r	res		r	r	r	r	res		

位 15:10	保留
位 9	**TSTS**：时间戳触发状态 (Time stamp trigger status) 在系统时间值等于或者超过目标时间高、低寄存器设定的值时，该位置'1'。 读这个寄存器则清'0'该位。
位 8:7	保留
位 6	**MMCTS**：MMC 发送状态 (MMC transmit status) 在 ETH_MMCTI 寄存器产生任一中断时，该位置'1'。ETH_MMCTI 寄存器里全部位清'0'时， 该位清'0'。
位 5	**MMCRS**：MMC 接收状态 (MMC receive status) 在 ETH_MMCRI 寄存器产生任一中断时，该位置'1'。ETH_MMCRI 寄存器里全部位清'0'时， 该位清'0'。
位 4	**MMCS**：MMC 状态 (MMC status) 位[6:5](MMCTS 位和 MMCRS)中任一位为'1'时，该位置'1'。位 6 和位 5 都为'0'时，该位置'0'。
位 3	**PMTS**：PMT 状态 (PMT status) 在掉电模式下，接收到唤醒帧或者 Magic Packet 时(见"以太网 MAC PMT 控制和状态寄存器"，位 5 和位 6 的描述)，该位置' 1' 。通过读 ETH_MACPMTCTRLSTS 寄存器把 WFR 位和 MPR 位清' 0' 以 后，该位也被清' 0' 。
位 2:0	保留

以太网MAC中断屏蔽寄存器(ETH_MAIMR)

地址偏移：0x003C

复位值：0x0000 0000

ETH_MAIMR寄存器可以用来屏蔽由于ETH_MACISTS中对应事件引起的中断。

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

保留	TSTINTM	保留	PMTINTM	保留
res	rw	res	rw	res

位 15:10	保留
位 9	**TSTINTM**：时间戳触发中断屏蔽 (Time stampe trigger interrupt mask) 置'1'时，禁止产生时间戳中断。
位 8:3	保留
位 3	**PMTINTM**：PMT 中断屏蔽 (PMT interrupt mask) 置'1'时，禁止产生由于 ETH_MACISTS 寄存器的 PMT 状态位置'1'而引发的中断。
位 2:0	保留

以太网MAC地址0高寄存器(ETH_MACA0H)

地址偏移：0x0040

复位值：0x0010 FFFF

MAC地址0高寄存器包含了6字节设备的第一个MAC地址的高16位。注意，MII接收到的帧的目的地址域的第一个字节，对应的是MAC地址低寄存器的低字节(位[7:0])。例如，MII接收到的目的地址是0x1122 3344 5566(0x11是第一个字节)，那么和MAC地址0寄存器位[47:0]比较的其实是0x6655 4433 2211。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MO	保留														
1	res														

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MACADDR0H															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31	**MO**：总是 1
位 30:16	保留
位 15:0	**MACADDR0H**：MAC 地址 0 高[47:32] (MAC address0 high [47:32])这些位包含了 6 字节 MAC 地址 0 的高 16 位。MAC 用这些位对接收到的帧进行过滤，也用来在发送流控(Pause)帧的时候作为帧的源地址。

以太网MAC地址0低寄存器(ETH_MACA0L)

地址偏移：0x0044

复位值：0xFFFF FFFF

MAC地址0低寄存器包含了6字节设备的第一个MAC地址的低32位。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MACADDR0L															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MACADDR0L															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:0	**MACADDR0L**：MAC 地址 0 低[31:0] (MAC addresss0 low [31:0]) 这些位包含了 6 字节 MAC 地址 0 的低 32 位。MAC 用这些位对接收到的帧进行过滤，也用来在发送流控(Pause)帧的时候作为帧的源地址。

以太网MAC地址1高寄存器(ETH_MACA1H)

地址偏移：0x0048

复位值：0x0000 FFFF

MAC地址1高寄存器包含了6字节的设备第二个MAC地址的高16位。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AEN	SADDR	MBCTRL									保留				

rw	rw			rw								res			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MACADDR1H															
rw															

位 31	**AEN**：地址使能 (Address enable) 1：地址过滤器使用 MAC 地址 1 来进行完美过滤。 0：地址过滤器进行过滤时忽略 MAC 地址 1。
位 30	**SADDR**：源地址 (Source address) 1：MAC 地址 1[47:0]用来和接收到帧的源地址比对。 0：MAC 地址 1[47:0]用来和接收到帧的目的地址比对。
位 29:24 位 23:16	**MBCTRL**：屏蔽字节控制 (Mask byte control)这些位是 MAC 地址 1 各字节比对的屏蔽控制位。当某个位置'1'时，MAC 不再把接收到帧目的地址或者源地址的对应字节和 MAC 地址 1 的同位置字节比较。每个控制位对应的 MAC 地址字节如下： 位 29：ETH_MACA1H[15:8] 位 28：ETH_MACA1H[7:0] 位 27：ETH_MACA1L[31:24] … 位 24：ETH_MACA1L[7:0] 保留
位 15:0	**MACADDR1H**：MAC 地址 1 高[47:32] (MAC address1 high [47:32]) 这些位包含了 6 字节的第二个 MAC 地址的高 16 位。

以太网MAC地址1低寄存器(ETH_MACA1L)

地址偏移：0x004C

复位值：0xFFFF FFFF

MAC地址1低寄存器包含了6字节设备第二MAC地址的低32位。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MACADDR1L															
rw															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MACADDR1L															
rw															

位 31:0	**MACADDR1L**：MAC 地址 1 低[31:0] (MAC address1 low [31:0]) 这些位包含了 6 字节 MAC 地址 1 的低 32 位。需要应用程序在初始化流程后，写本寄存器来定义这些位的值。

以太网MAC地址2高寄存器(ETH_MACA2H)

地址偏移：0x0050

复位值：0x0000 FFFF MAC地址2高寄存器包含了6字节的设备第二MAC地址的高16位。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AEN	SADDR	MBCTRL						保留							
rw	rw	rw						res							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MACADDR2H															
rw															

位 31	**AEN**：地址使能 (Address enable) 1：地址过滤器使用 MAC 地址 2 来进行完美过滤。 0：地址过滤器进行过滤时忽略 MAC 地址 2。
位 30	**SADDR**：源地址 (Source address) 1：MAC 地址 2[47:0]用来和接收到帧的源地址比对。 0：MAC 地址 2[47:0]用来和接收到帧的目的地址比对。

位 29:24	**MBCTRL**：屏蔽字节控制 (Mask byte control) 这些位是 MAC 地址 2 各字节比对的屏蔽控制位。当某个位置'1'时，MAC 不再把接收到帧目的地址或者源地址的对应字节和 MAC 地址 2 的同位置字节比较。每个控制位对应的 MAC 地址字节如下： 位 29：ETH_MACA2H[15:8] 位 28：ETH_MACA2H[7:0] 位 27：ETH_MACA2L[31:24] … 位 24：ETH_MACA2L[7:0]
位 23:16	保留
位 15:0	**MACADDR2H**：MAC 地址 2 高[47:32] (MAC address2 high [47:32]) 这些位包含了 6 字节 MAC 地址 2 的高 16 位。

以太网MAC地址2低寄存器(ETH_MACA2L)

地址偏移：0x0054

复位值：0xFFFF FFFF MAC地址2低寄存器包含了6字节的设备第二MAC地址的低32位。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MACADDR2L															
rw															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MACADDR2L															
rw															

位 31:0	**MACADDR2L**：MAC 地址 2 低[31:0] (MAC address2 low [31:0]) 这些位包含了 6 字节 MAC 地址 2 的低 32 位。需要应用程序在初始化流程后，写本寄存器来定义这些位的值。

以太网MAC地址3高寄存器(ETH_MACA3H)

地址偏移：0x0058

复位值：0x0000 FFFF MAC地址1高寄存器包含了6字节的设备第二MAC地址的高16位。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AEN	SADDR	MBCTRL						保留							
rw	rw	rw						res							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MACADDR3H															
rw															

位 31	**AEN**：地址使能 (Address enable) 1：地址过滤器使用 MAC 地址 3 来进行完美过滤。 0：地址过滤器进行过滤时忽略 MAC 地址 3。
位 30	**SADDR**：源地址 (Source address) 1：MAC 地址 3[47:0]用来和接收到帧的源地址比对。 0：MAC 地址 3[47:0]用来和接收到帧的目的地址比对。
位 29:24	**MBCTRL**：屏蔽字节控制 (Mask byte control) 这些位是 MAC 地址 3 各字节比对的屏蔽控制位。当某个位置'1'时，MAC 不再把接收到帧目的地址或者源地址的对应字节和 MAC 地址 3 的同位置字节比较。每个控制位对应的 MAC 地址字节如下： 位 29：ETH_MACA3H[15:8] 位 28：ETH_MACA3H[7:0] 位 27：ETH_MACA3L[31:24] … 位 24：ETH_MACA3L[7:0]
位 23:16	保留
位 15:0	**MACADDR3H**：MAC 地址 3 高[47:32] (MAC address3 high [47:32]) 这些位包含了 6 字节 MAC 地址 3 的高 16 位。

以太网MAC地址3低寄存器(ETH_MACA3L)

地址偏移：0x005C

复位值：0xFFFF FFFF MAC地址3低寄存器包含了6字节的设备第二MAC地址的低32位。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MACADDR3L															
rw															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MACADDR3L															
rw															

位 31:0	**MACADDR3L**：MAC 地址 3 低[31:0] (MAC address3 low [31:0]) 这些位包含了 6 字节 MAC 地址 3 的低 32 位。需要应用程序在初始化流程后，写本寄存器来定义这 些位的值。

24.8.2 MMC寄存器描述

以太网MMC控制寄存器(ETH_MMCCTRL)

地址偏移：0x0100

复位值：0x0000 0000

以太网MMC控制寄存器设定了各管理计数器的工作模式。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															
res															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留												MCNTF	RSTOR	CSR	CNTR
res											rw	rw	rw	rw	rw

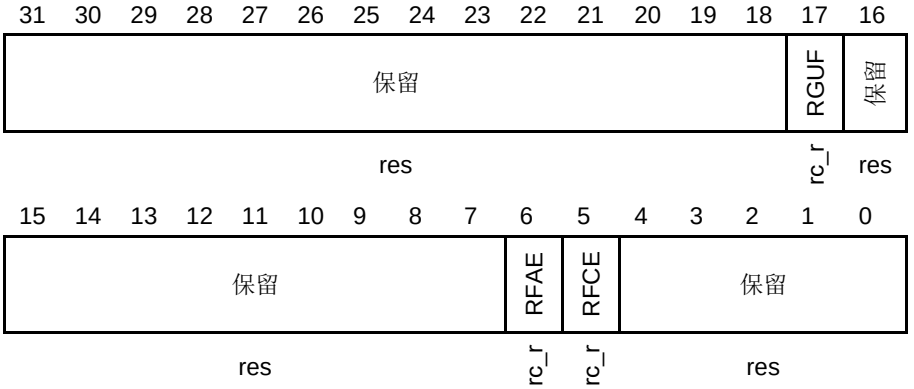
位 31:4	保留
位 3	**MCNTF**：MMC 计数器冻结 (MMC counter freeze) 置'1'时，所有的 MMC 计数器冻结，保持它们的当前值。(直到该位清'0'，接收或者发送帧都不 会更新任何 MMC 计数器。但在此模式下，如果把 RSTOR 位置'1'并读任一计数器，那么该计数器 会清'0')。
位 2	**RSTOR**：读时复位 (Reset on read) 置'1'时，在读 MMC 计数器后，该计数器复位 w 为'0'(复位后自清'0')。读计数器的低字节(位[7:0]) 之后，计数器就会清'0'。
位 1	**CSR**：计数器停止回转 (Counter stop rollover) 置'1'时，计数器在计数到最大值后，不会重新从 0 开始计数。
位 0	**CNTR**：计数器复位 (Counter reset) 置'1'时，复位全部计数器。在 1 个时钟周期后该位自动清'0'。

以太网MMC接收中断寄存器(ETH_MMCRI)

地址偏移：0x0104

复位值：0x0000 0000

以太网MMC接收中断寄存器记录：在接收统计数据寄存器计数到最大值的一半时(计数器的最高位置'1')所产生的中断。它是一个32位寄存器。读产生中断的MMC计数器可以清除对应的中断位。必须读相应计数器的低字节(位[7:0])来清除中断位。



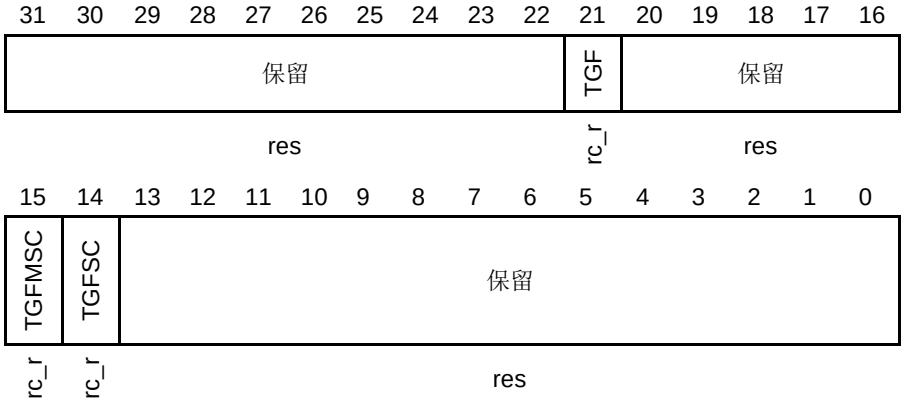
位 31:18	保留
位 17	**RGUF**：接收到”好”的单播帧状态 (Received Good Unicast Frames Status) 接收“好”单播帧计数器达到最大值一半时，该位置’1’。
位 16:7	保留
位 6	**RFAE**：接收到帧对齐错误状态 (Received frames alignment error status) 对齐错误接收帧计数器达到最大值一半时，该位置’1’。
位 5	**RFCE**：接收到帧 CRC 错误状态 (Received frames CRC error status) CRC 错误接收帧计数器达到最大值一半时，该位置’1’。

以太网MMC发送中断寄存器(ETH_MMCTI)

地址偏移：0x0108

复位值：0x0000 0000

以太网MMC发送中断寄存器记录：在发送统计数据寄存器计数到最大值的一半时(计数器的最高位置’1’)所产生的中断。它是一个32位寄存器。读产生中断的MMC计数器可以清除对应的中断位。必须读相应计数器的低字节(位[7:0])来清除中断位。



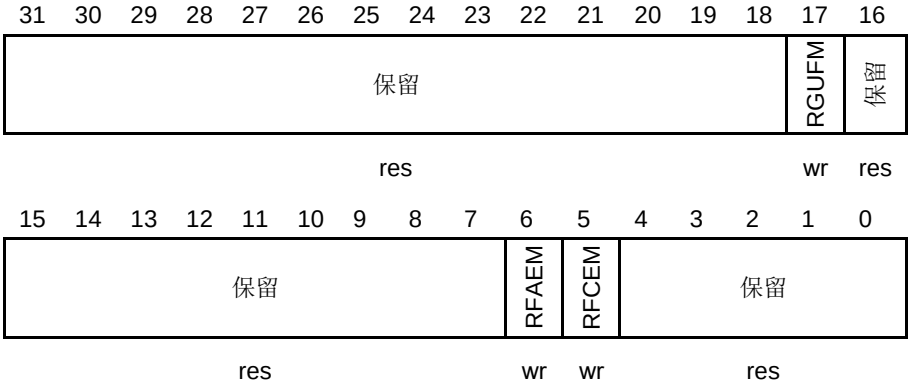
位 31:22	保留
位 21	**TGF**：发送”好”的帧状态 (Transmitted good frames status) 发送“好”单播帧计数器达到最大值一半时，该位置’1’。
位 20:16	保留
位 15	**TGFMSC**：发送”好”的帧时遇到 1 个以上冲突状态 (Transmitted good frames more single collision status) 发送时 1 次以上冲突后”好”帧计数器达到最大值一半时，该位置’1’。
位 14	**TGFSC** ：发送” 好” 的帧时仅遇到 1 个冲突状态 (Transmitted good frames single collisionstatus) 发送时 1 次冲突后的”好:帧计数器达到最大值一半时，该位置’1’。
位 13:0	保留

以太网MMC接收中断屏蔽寄存器(ETH_MMCRIM)

地址偏移：0x010C

复位值：0x0000 0000

以太网MMC接收中断屏蔽寄存器包含：对接收统计数据寄存器计数到最大值的一半时(计数器的最高位置’1’)所产生中断的屏蔽位。它是一个32位寄存器。



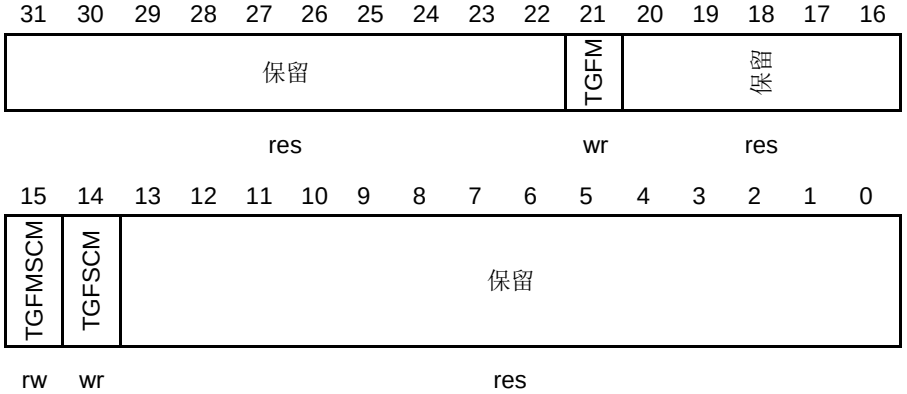
位31:18	保留
位17	**RGUFM**：接收到”好”的单播帧屏蔽（Received good unicast frames mask） 置’1’时，屏蔽接收“好”单播帧计数器达到最大值一半时发生的中断。
位16:15	保留
位6	**RFAEM**：接收到帧对齐错误屏蔽(Received frames alignment error mask) 置’1’时，屏蔽对齐错误接收帧计数器达到最大值一半时发生的中断。
位5	**RFCEM**：接收到帧CRC错误屏蔽(Received frame CRC error mask) 置’1’时，屏蔽CRC错误接收帧计数器达到最大值一半时发生的中断。
位4:0	保留

以太网MMC发送中断屏蔽寄存器(ETH_MMCTIM)

地址偏移：0x0110

复位值：0x0000 0000

以太网MMC发送中断屏蔽寄存器包含：对发送统计数据寄存器计数到最大值的一半时(计数器的最高位置’1’)所产生中断的屏蔽位。它是一个32位寄存器。



位 31:22	保留
位 21	**TGFM**：发送”好”的帧屏蔽（Transmitted good frames mask） 置’1’时，屏蔽发送”好”帧计数器达到最大值一半时发生的中断。
位 20:16	保留
位 15	**TGFMSCM**：发送”好”的帧时遇到 1 个以上冲突屏蔽（Transmitted good frames more single collision mask) 置’1’时，屏蔽发送时 1 次以上冲突后的”好”帧计数器达到最大值一半时发生的中断。
位 14	**TGFSCM**：发送”好”的帧时仅遇到 1 个冲突屏蔽（Transmitted good frames single collision mask)

	置'1'时，屏蔽发送时 1 次冲突后"好"帧计数器达到最大值一半时发生的中断。
位 13:0	保留

以太网MMC 1次冲突后发送”好”帧的计数器寄存器(ETH_MMCTFSCC)

地址偏移：0x014C

复位值：0x0000 0000

该寄存器统计在半双工模式下，发送帧成功时只遇到一次冲突的帧的数目。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TGFSCC															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TGFSCC															
r															

位31:0	**TGFSCC**：发送时1次冲突好帧计数器 (Transmitted good frames single collision counter)

以太网MMC 1次以上冲突后发送”好”帧的计数器寄存器(ETH_ MMCTFSCC)

地址偏移：0x0150

复位值：0x0000 0000

该寄存器统计在半双工模式下，发送帧成功时遇到一次以上冲突的帧的数目。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TGFMSCC															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TGFMSCC															
r															

位31:0	**TGFMSCC**：发送时1次以上冲突好帧计数器(Transmitted good frame more single collision counter) 这些位是1次以上冲突后发送"好"帧的计数器。

以太网MMC发送”好”帧计数器寄存器(ETH_ MMCTFCNT)

地址偏移：0x0168

复位值：0x0000 0000

该寄存器统计发送”好”帧的数目。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TGFC															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TGFC															
r															

位31:0	**TGFC**：发送好帧计数器(Transmitted good frames counter)

以太网MMC CRC错误接收帧计数器寄存器(ETH_ MMCRFCECNT)

地址偏移：0x0194

复位值：0x0000 0000

该寄存器统计接收到有CRC错误帧的数目。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RFCEC															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RFCEC															
r															

位31:0	**RFCEC**：CRC错误接收帧计数器(Received frames CRC error counter)

以太网MMC对齐错误接收帧计数器寄存器(ETH_ MMCRFAECNT)

地址偏移：0x0198

复位值：0x0000 0000

该寄存器统计接收到有对齐错误帧的数目。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RFAEC															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RFAEC															
r															

位31:0	**RFAEC**：CRC错误接收帧计数器 (Received frames CRC error counter)

以太网MMC 接收帧”好”单播帧计数器寄存器(ETH_ MMCRGUFCNT)

地址偏移：0x01C4

复位值：0x0000 0000

该寄存器统计接收到”好”单播帧的数目。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RGUFC															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RGUFC															
r															

位31:0	**RGUFC**：接收“好”单播帧计数器 (Received good unicast frames counter)

24.8.3 IEEE 1588 时间戳寄存器

本节描述的寄存器用来支持IEEE 1588规范定义的精确网络时钟同步功能。

以太网PTP时间戳控制寄存器(ETH_PTPTSCTRL)

地址偏移：0x0700

复位值：0x0000 2000

该寄存器控制了时间戳生成和更新逻辑。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
res													TSPFFMAE	TSCNT	
													rw	rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TSSMRME	TSSEME	TSSIPV4FE	TSSIPV6FE	TSSPTPOEFE	TSPTPPSV2E	TSSSR	TSSARFE	保留		TSARU	TSETE	TSSTU	ITSST	TSFCU	TSE

rw	rw	rw	rw	rw	rw	rw	rw	res	rw	rw	rw	rw	rw	rw

位 31:19	保留
位 18	**TSPFFMAE**：时间戳 PTP 帧过滤 MAC 地址使能 (Time stamp PTP frame filtering MAC address enable) 如果置 1，则当 PTP 直接通过以太网发送时，此位将使用 MAC 地址（除 MAC 地址 0）来过 滤 PTP 帧。
位 17:16	**TSCNT**：时间戳时钟节点类型 (Time stamp clock node type) 可用的时钟节点类型如下： 00：普通时钟 01：边界时钟 10：端对端透明时钟 11：点对点透明时钟
位 15	**TSSMRME**：主节点相关消息时间戳快照使能 (Time stamp snapshot for message relevant to master enable) 当此位置 1 时，仅拍摄主节点相关消息的快照。当此位清零时，仅拍摄从节点相关消息的快 照。这仅适用于普通时钟和边界时钟节点。
位 14	**TSSEME**：事件消息时间戳快照使能 (Time stamp snapshot for event message enable) 当此位置 1 时，仅拍摄事件消息的时间戳快照（SYNC、Delay_Re、Pdelay_Req 或 Pdelay_Resp）。当此位清零时，将拍摄除 Announce、Management 和 Signaling 以外其它 所有消息的快照。
位 13	**TSSIPV4FE**：IPv4 帧时间戳快照使能 (Time stamp snapshot for IPv4 frames enable) 当此位置 1 时，将拍摄 IPv4 帧的时间戳快照。
位 12	**TSSIPV6FE**：IPv6 帧时间戳快照使能 (Time stamp snapshot for IPv6 frames enable) 当此位置 1 时，将拍摄 IPv6 帧的时间戳快照。
位 11	**TSSPTPOEFE**：PTP over ethernet 帧时间戳快照使能 (Time stamp snapshot for PTP over ethernet frames enable) 当此位置 1 时，将拍摄在以太网帧中也含有 PTP 消息 (PTP over Ethernet) 的帧的时间戳快 照。默认情况下，将拍摄 UDP-IPEthernet PTP 数据包的快照。
位 10	**TSPTPPSV2E**：监听版本 2 格式时间戳 PTP 数据包使能 (Time stamp PTP packet snooping for version2 format enable) 当此位置 1 时，将使用版本 2 格式对 PTP 数据包进行监听。当此位清零时，将使用版本 1 格 式对 PTP 数据包进行监听。 注意： *IEEE 1588* 版本 *1* 和版本 *2* 格式符合 *IEEE* 标准 *1588-2008*（版本为 *IEEE STD.1588- 2002).*
位 9	**TSSSR**：时间戳亚秒翻转：数字或二进制翻转控制 (Time stamp subsecond rollover: digital or binary rollover control) 当此位置 1 时，如果亚秒计数器达到值 0x3B9A C9FF（十进制 999 999 999）并递增了时间 戳（高位）秒数，则时间戳低位寄存器将翻转。 当此位清零时，亚秒寄存器的翻转值将达到 0x7FFF FFFF。亚秒增量必须根据 PTP 参考时 钟频率和此位的值正确进行编程。
位 8	**TSSARFE**：所有接收帧的时间戳快照使能 (Time stamp snapshot for all received frames enable) 当此位置 1 时，时间戳快照会针对内核所接收的全部帧处于使能状态。
位 7:6	保留
位 5	**TSARU**：时间戳加数寄存器更新 (Time stamp addend register update) 置'1'时，时间戳加数寄存器的值会更新到 PTP 模块，用做系统时间细调。该位在更新完成后 清'0'。必须确保读出该位为'0'时，才能在把该位置'1'。
位 4	**TSITE**：时间戳中断触发使能 (Time stamp interrupt trigger enable) 置'1'时，如果系统时间大于目标时间寄存器的值，就产生时间戳中断。在时间戳中断产生以 后，该位清'0'。
位 3	**TSSTU**：时间戳系统时间更新 (Time stamp system time update) 置'1'时，系统时间会更新，在原有系统时间上加上或者减去时间戳高和低 2 个更新寄存器的值。 必须确保 TSSTU 位和 TSSTI 位读出为'0'时，才能把该位置'1'。硬件完成更新后，清除该位。
位 2	**TSSTI**：时间戳系统时间初始化 (Time stamp system time initialize)

	置'1'时，系统时间会更新，原有系统时间替换为时间戳高和低 2 个更新寄存器的值。必须确保该 位读出为'0'时，才能把该位置'1'。在硬件初始化完成后，清除该位。
位 1	**TSFCU**：时间戳更新粗调或者细调 (Time stamp fine or coarse update) 1：表示用细调的方式更新系统时间戳； 0：表示用粗调的方式更新系统时间戳。
位 0	**TSE**：时间戳使能 (Time stamp enable) 1：使能接收和发送帧的时间戳功能。 0：不再往接收和发送的帧上添加时间戳。 由于置'0'后，原有的系统时间丢失，因此每次设置该位为'1'后，都需要重新初始化系统时间。

时间戳快照对寄存器位的相依性

TSCNT 位17:16	TSSMRME 位15	TSSEME 位14	拍摄快照时的消息
00 或 01	X	0	SYNC、Follow_Up、Delay_Req、Delay_Resp
00 或 01	1	1	Delay_Req
00 或 01	0	1	SYNC
10	N/A	0	SYNC、Follow_Up、Delay_Req、Delay_Resp
10	N/A	1	SYNC、Follow_Up
11	N/A	0	SYNC、Follow_Up、Delay_Req、Delay_Resp、 Pdelay_Req、Pdelay_Resp
11	N/A	1	SYNC、Pdelay_Req、Pdelay_Resp

1 ：N/A=不适用

2 ：X=无关

以太网PTP亚秒递增寄存器(ETH_PTPSSINC)

地址偏移：0x0704

复位值：0x0000 0000

该寄存器包含亚秒寄存器每次递增的8位数值。在粗调模式下(ETH_PTPTSCTRL寄存器TSFCU位为' 0'时)，每个HCLK时钟周期，系统时间就加一次这个寄存器的值。在细调模式下，在累加器溢出时，系统时间才加一次这个寄存器的值。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															
res															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留								STSINC							
res								rw							

位31:8	保留
位7:0	**STSINC**：系统时间亚秒递增(System time subsecond increment) 在每次更新系统时间时，把这些位的值加到系统时间亚秒寄存器上。 例如，为了系统时间精度达到20ns，这些位的值应当是20 / 0.467 = ~43(或者0x2A)。

以太网PTP时间戳高寄存器(ETH_PTPTSH)

地址偏移：0x0708

复位值：0x0000 0000

该寄存器包含时间信息的高32位。该寄存器为只读，包含了系统时间的秒值。系统时间高和低2个寄存器包含MAC的当前系统时间值，这个值会持续更新。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16

STS

r

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

STS

r

位31:0	**STS**：系统时间秒(System time second) 这些位表示了MAC当前系统时间的秒值。

以太网PTP时间戳低寄存器(ETH_PTPTSL)

地址偏移：0x070C

复位值：0x0000 0000

该寄存器包含时间信息的低32位。该寄存器为只读，包含了系统时间的亚秒值。

31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16

STPNS	STSS

r r

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

STSS

r

位31	**STPNS**：系统时间正或者负标志位(System time positive or negative sign) 该位表示时间值是正值还是负值。 1：表示时间值是负值。 0：表示时间值是正值。由于系统时间总是正的，所以该位一般为'0'。
位30:0	**STSS**：系统时间亚秒(System time subseconds) 这些位表示了MAC当前系统时间的亚秒值，精度为0.46ns。

以太网PTP时间戳高更新寄存器(ETH_PTPTSHUD)

地址偏移：0x0710

复位值：0x0000 0000

该寄存器包含了系统时间更新值的高32位，使用这个更新值对当前系统时间替换、相加、或者相减。系统时间高和低更新2个寄存器可以用来初始化或更新MAC的当前系统时间。应当先写这2个寄存器，再把时间戳控制寄存器的TSSTI位或TSSTU位置为'1'。

31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16

TSUS

rw

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

TSUS

rw

位31:0	**TSUS**：系统时间秒(System time second) 这些位表示了MAC当前系统时间的秒值。

以太网PTP时间戳低更新寄存器(ETH_ PTPTSLUD)

地址偏移：0x0714

复位值：0x0000 0000

该寄存器包含了系统时间更新值的低32位，使用这个更新值对当前系统时间替换、相加、或者相减。

31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16

TSUPNS	TSUSS

rw rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TSUSS															

rw

位31	**TSUPNS**：时间戳更新正或者负标志位 (Time stamp update positive or negative sign)) 该位表示时间值是正值还是负值。 1：表示时间值是负值； 0：表示时间值是正值。TSSTI位置'1'时，该位应当为'0'。如果TSSTU位置'1'，该位为'1'表示从系统时间里减去更新值，该位为'0'表示在系统时间上加上更新值。
位30:0	**TSUSS**：时间戳更新亚秒(Time stamp update subseconds) 这些位表示了将要替换系统时间或者在系统时间上加上的时间亚秒值。这个值的精度是0.46ns(即数值0x0000 0001表示0.46ns)。

以太网PTP时间戳加数寄存器(ETH_PTPTSAD)

地址偏移：0x0718

复位值：0x0000 0000

软件使用该寄存器线性地校准时钟频率到主时钟。该寄存器只用于系统时间更新方式为细调的情况下(ETH_PTPTSCTRL寄存器的TSFCU位为' 1')。该寄存器的值每个时钟周期都会累加到32位累加器上，一旦该累加器溢出就更新系统时间。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TSA															

rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TSA															

rw

位31:0	**TSA**：时间戳加数(Time stamp addend) 这些位表示了用来加到累加器上实现时间同步的32位加数值。

以太网PTP目标时间高寄存器(ETH_PTPTTH)

地址偏移：0x071C

复位值：0x0000 0000

该寄存器包含要与用于中断事件生成的系统时间进行比较的高 32 位时间。目标时间高位寄 存器与目标时间低位寄存器一起，用于在系统时间超过这些寄存器中编程的值时规划中断事件（ETH_PTPTSCTRL 中的 TSARU 位）。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TTSH															

rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TTSH															

rw

位31:0	TTSH：目标时间戳高 (Target time stamp high) 这些位表示了目标时间的秒值。如果时间戳值等于或者超过目标时间，且使能了相应的中断， MAC就会产生中断。

以太网PTP目标时间低寄存器(ETH_PTPTTL)

地址偏移：0x0720

复位值：0x0000 0000

该寄存器包含要与用于中断事件生成的系统时间进行比较的低 32 位时间。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TTSL															
rw															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TTSL															
rw															

位31:0	**TTSL**：目标时间戳低(Target time stamp low) 这些位表示了目标时间的纳秒值。如果时间戳值等于或者超过目标时间，且使能了相应的中断，MAC就会产生中断。

以太网 PTP 时间戳状态寄存器 (ETH_PTPTSSR)

Ethernet PTP time stamp status register

偏移地址：0x0728

复位值：0x0000 0000

该寄存器包含时间戳状态寄存器。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															
res															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留														TSTTR	TSSO
res														ro	ro

位31:2	保留
位1	**TSTTR**：达到时间戳目标时间（Time stamp target time reached） 置 1 时，该位指示系统时间值大于或等于在目标时间高位和低位寄存器中指定的值。
位0	**TSSO**：时间戳秒上溢 (Time stamp second overflow) 置 1 时，该位指示时间戳的秒值已上溢，超过 0xFFFF FFFF。

以太网 PTP PPS 控制寄存器 (ETH_PTPPPSCR)

Ethernet PTP PPS control register

偏移地址：0x072C

复位值：0x0000 0000

该寄存器控制 PPS 输出的频率。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															
res															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

保留	PPSFREQ
res	rw

位31:4	保留
位3:0	**PPSFREQ**：PPS 频率选择（PPS frequency selection） 此位输出受emac_pps_sel位控制（RCC_MISC2寄存器位15） Emac_pps_sel=0: 0000：1 Hz，使用二进制翻转时，脉冲宽度为 125 ms，使用数字翻转时，脉冲宽度为 100 ms 0001：2 hz，使用二进制翻转时，占空比为 50%（不建议使用数字翻转） 0010：4 hz，使用二进制翻转时，占空比为 50%（不建议使用数字翻转） 0011：8 hz，使用二进制翻转时，占空比为 50%（不建议使用数字翻转） 0100：16 hz，使用二进制翻转时，占空比为 50%（不建议使用数字翻转） ... 1111：32.768 khz，使用二进制翻转时，占空比为 50%（不建议使用数字翻转） Emac_pps_sel=1: 0000：1 Hz，脉冲宽度为一个clk_ptp周期 0001：二进制翻转是2hz，占空比50%，数字翻转是1hz（不建议使用数字翻转） 0010：二进制翻转是4hz，占空比50%，数字翻转是2hz（不建议使用数字翻转） 0011：二进制翻转是8hz，占空比50%，数字翻转是4hz（不建议使用数字翻转） ... 1111：二进制翻转是32.768khz，占空比50%，数字翻转是16.384khz（不建议使用数字翻转） 在PPS为非0值时，不建议使用数字翻转，因为这些情况下PPS输出的波形会不规则（尽管其平均频率在任何一秒窗口内始终正确）

24.8.4 DMA寄存器描述

本节定义了DMA寄存器中每一位的意义。只要地址是32位对齐的，允许非32位访问。

以太网DMA总线模式寄存器(ETH_DMABM)

地址偏移：0x1000

复位值：0x0002 0101

总线模式寄存器设定了DMA的总线工作模式。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留						AAB	FPBLM	USP	RDPBL						FB
res						rw	rw	rw	rw						rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RTPR		PBL						保留	DSL					DA	SWRST

rw	rw	res	rw	rw	rw

位 31:26	保留
位 25	**AAB**：传输地址对齐 (Address-aligned beats) 置'1'时，如果 FB 位为'1'，AHB 接口对齐所有突发传输至起始地址的 LS 位。如果 FB 位为'0'，不对齐第一次的突发传输(即对数据缓存起始的访问)，但接下来的传输与地址对齐。
位 24	**FPBLM**：8×PBL 模式 (8xPBL mode) 置'1'时，把设好的 PBL 值(位[22:17]和位[13:8])乘以 8。这样，根据 PBL 的值，DMA 最大传输 8、16、32、64、128 和 256 次数据。
位 23	**USP**：使用分散 PBL (Use separate PBL) 1：把 RDPBL 位(位[22:17])的值作为接收 DMA 的 PBL 值，同时 PBL 位(位[13:8])的值只作为发送 DMA 的 PBL 值。 0：位[13:8]设定的 PBL 值对 DMA 接收和发送控制器都有效。
位 22:17	**RDPBL**：接收 DMA PBL (Rx DMA PBL) 这些位定义了一次 DMA 转发的最大数据传输次数。这个最大值用于单个模块的读写操作。接收 DMA 总是试图在总线上开始突发传输时，连续传输 RDPBL 位定义的次数。RDPBL 位允许设的值有 1、2、4、8、16 和 32。设为其他数值会导致不可预料的后果。 只有在 USP 位置'1'时，这些位才有效。
位 16	**FB**：固定的突发 (Fixed burst) 该位控制 AHB 主接口是否进行固定突发长度的传输。 1：AHB 在开始常规突发传输时，利用 SINGLE，INCR4，INCR8 和 INCR16。 0：AHB 在突发传输时，只用 SINGLE 和 INCR。
位 15:14	**RTPR**：接收 发送优先级比率 (Rx Tx priority ratio) 按照以下比例给予接收 DMA 高于发送 DMA 的优先级： 0:01:01 1:02:01 10:03:01 11:04:01 该位只在 DA 位为'0'时有效。
位 13:8	**PBL**：可编程的突发长度 (Programmable burst length) 这些位定义了一次 DMA 转发的最大数据传输次数。这个最大值用于单个模块的读写操作。 DMA 总是试图在总线上开始突发传输时，连续传输 PBL 域定义的次数。PBL 域允许设的值有 1、2、4、8、16 和 32。设为其他值的话，会导致不可预料的后果。如果 USP 位为' 1' ，PBL 域的值 只对发送 DMA 有效。 PBL 的取值有以下限制： - 可能的最大传输次数(PBL)受发送 FIFO 和接收 FIFO 大小的限制； - FIFO 支持的最大传输次数是 FIFO 深度的一半； - 如果为接收 DMA 和发送 DMA 设定相同的 PBL，则需要考虑接收和发送 FIFO 的最小深度； - 不要取允许取值以外的 PBL 值，不然会引起系统工作不正常。
位 7	保留
位 6:2	**DSL**：描述符跳跃长度 (Descriptor skip length) 这些位定义了 2 个不以链式结构连接的描述符之间的跳跃距离，单位为字(32 位)。地址跳跃是指 从当前描述符的结尾到下一个描述符开头的地址差值。当 DSL 域为 0 时，在环形结构下，DMA 认为描述符是相邻地连续排列的。
位 1	**DA**：DMA 仲裁 (DMA Arbitration) 0：根据位[15:14]定义的发送和接收优先级比，以循环方式仲裁； 1：接收的优先级高于发送。
位 0	**SWRST**：软件复位 (Software reset)

	置'1'时，MAC 的 DMA 控制器复位 MAC 所有子系统的内部寄存器和逻辑电路。在 MAC 内部不同 时钟域模块完成复位操作后，自动清除该位。在重新写 MAC 的寄存器前，应当确保该位为'0'。

以太网DMA发送查询请求寄存器(ETH_DMATPD)

地址偏移：0x1004

复位值：0x0000 0000

程序用该寄存器来要求DMA查询发送描述符队列。发送查询请求可以使发送DMA检查DMA是否占有当前的描述符。发送查询命令可以唤醒处于暂停状态的发送DMA。发送DMA通常因为发送帧的数据下溢错误或者因为不能占有想要的描述符而进入暂停状态。可以在任一时刻通过写寄存器发起发送查询请求，在发送DMA开始重新取描述符时，会把这些位置'0'。

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

位31:0	TPD：发送查询要求 (Transmit poll demand) 对这些位写任意值，DMA 即读 ETH_DMACTD 寄存器指向的当前描述符。如果该描述符不可用(由 CPU 占有)，则发送 DMA 回到暂停状态，并把 ETH_DMASTS 的位 2 置'1'。如果描述符可用，则恢复发送流程。

以太网DMA接收查询请求寄存器(ETH_DMARPD)

地址偏移：0x1008

复位值：0x0000 0000

程序用该寄存器来要求DMA查询接收描述符队列。接收查询请求可以使接收DMA检查新描述符。接收查询命令可以唤醒处于暂停状态的接收DMA。接收DMA通常因为不能占有想要的描述符而进入暂停状态。

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

位31:0	**RPD**：接收查询要求 (Receive poll demand) 对这些位写任意值，DMA 即读 ETH_DMACRD 寄存器指向的当前描述符。如果该描述符不可用(由 CPU 占有)，则接收 DMA 回到暂停状态，并把 ETH_DMASTS 的位 7 置'0'。如果描述符可 用，则接收 DMA 回到运行状态。

以太网DMA接收描述符列表地址寄存器(ETH_DMARDLADDR)

地址偏移：0x100C

复位值：0x0000 0000

接收描述符列表寄存器指向接收描述符队列的开始。描述符队列位于AT32F407xx的物理内存，并且其地址必须以字对齐。DMA会在内部把描述符地址的最低位置'0'来使地址对齐总线宽度。只有在接收停止的时候，才允许写ETH_DMARDLADDR寄存器。在接收停止后，应当先写ETH_DMARDLADDR寄存器，再发接收开始命令。

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

位31:0	**SRL**：接收队列基址 (Start of receive list) 这些位包含了接收描述符队列第一个描述符的地址。DMA 忽略最低的若干位[1/2/3:0](对应总线 宽度为 32/64/128 位的系统)的取值，默认它们为'0'。因此，这几个最低位是只读的。

以太网DMA发送描述符列表地址寄存器(ETH_DMATDLADDR)

地址偏移：0x1010

复位值：0x0000 0000

发送描述符列表寄存器指向发送描述符队列的起始。描述符队列位于AT32F407xx的物理内存，并且其地址必须以字对齐。DMA会在内部把描述符地址的最低位置'0'来使地址对齐总线宽度。只有在发送停止的时候，才允许写ETH_DMATDLADDR寄存器。在发送停止后，应当先写ETH_DMATDLADDR寄存器，再发发送开始命令。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
STL															
rw															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
STL															
rw															

位31:0	**STL**：发送队列基址(Start of transmit list) 这些位包含了发送描述符列表第一个描述符的地址。DMA 忽略最低的若干位[1/2/3:0](对应总线 宽度为 32/64/128 位的系统)的取值，默认它们为'0'。因此，这几个最低位是只读的。

以太网DMA状态寄存器(ETH_DMASTS)

地址偏移：0x1014

复位值：0x0000 0000

状态寄存器包含了所有DMA反馈给应用程序的状态位。软件通常在中断里或在轮询的过程中读该寄存器。该寄存器里大多数位都能触发中断。读ETH_DMASTS寄存器并不能清除其中的标志位。ETH_DMASTS寄存器位[16:0](除保留位)需要写'1'才能清除，而写'0'是无效的。通过设置ETH_DMAIE寄存器里的相应位，可以屏蔽位[16:0]触发的中断。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留		TST	PMT	MMC	保留	EB			TP			RP			NIS
res		r	r	r	res	r				r			r		rc_w1

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AIS	ER	FBE	保留		ET	RWT	RPS	RSBU	RECV	TU	RO	TJT	TBU	TPS	TX
rc_w1			res							rc_w1					

位31:30	保留
位29	**TST**：时间戳触发状态 (Time stamp trigger status) 表示MAC控制器的时间戳生成模块发生了中断事件。软件必须读MAC控制器的状态寄存器，清除中断源(位9)才能清除该位。该位置'1'时，如果使能了相应中断，则产生中断。
位28	**PMT**：PMT状态(PMT status) 表示 MAC 控制器的 PMT 模块发生了中断事件。软件必须读 MAC 控制器的相应寄存器，找到中断来源并清除，才能清除该位。该位置'1'时，如果使能了相应中断，则产生中断。
位27	**MMC**：MMC状态(MMC status) 表示MAC控制器的MMC模块发生了中断事件。软件必须读MAC控制器的相应寄存器，找到中断来源并清除，才能清除该位。该位置'1'时，如果使能了相应中断，则产生中断。
位26	保留

位25:23	**EB**：错误位状态(Error bits status) 该域表示了造成总线错误(AHB接口端)的错误类型。仅在总线致命错误位(ETH_DMASTS[13])为'1'时，这些位才有效。这些位不触发中断。 位23 0 发送DMA转发数据时出错 1 接收DMA转发数据时出错 位24 0 读数据转发时出错 1 写数据转发时出错 位25 0 访问描述符时出错
位22:20	**TP**：发送流程状态(Transmit process state) 这些位表示发送DMA的FSM状态。这些位不触发中断。 000：停止，接到复位或者停止发送命令； 001：运行，正在取发送描述符； 010：运行，正在等待状态信息 011：运行，正在读取内存中数据并压入发送FIFO中； 100，101：保留； 110：暂停，发送描述符不可用或发送缓存数据下溢； 111：运行，正在关闭发送描述符。
位 19:17	**RP**：接收流程状态 (Receive process state) 这些位表示接收 DMA 的 FSM 状态。这些位不触发中断。 000：停止，接到复位或者停止接收命令； 001：运行，正在取接收描述符； 010：保留； 011：运行，正在等待接收数据包； 100：暂停，接收描述符不可用； 101：运行，正在关闭接收描述符； 110：保留； 111：运行，正在把接收到数据包从接收 FIFO 转发到内存中。
位 16	**NIS**：正常中断汇总 (Normal interrupt summary) 在 ETH_DMAIE 寄存器使能了相应中断的情况下，正常中断汇总位是下列位取值的逻辑或 - ETH_DMASTS[0]：发送中断 - ETH_DMASTS[2]：发送缓存不可用 - ETH_DMASTS[6]：接收中断 - ETH_DMASTS[14]：早接收中断 只有未被屏蔽的中断才会影响到本位。 NIS 位的取值和以上位绑定，置'1'后，只有把造成该位置'1'的位清'0'(写'1')，才能把该位清'0'。
位 15	**AIS**：异常中断汇总 (Abnormal interrupt summary) 在 ETH_DMAIE 寄存器使能了相应中断的情况下，异常中断汇总位是下列位取值的逻辑或 - ETH_DMASTS[1]：发送流程停止 - ETH_DMASTS[3]：发送啰嗦超时 - ETH_DMASTS[4]：接收 FIFO 溢出 - ETH_DMASTS[5]：发送数据下溢 - ETH_DMASTS[7]：接收缓存不可用 - ETH_DMASTS[8]：接收流程停止 - ETH_DMASTS[9]：接收看门狗超时 - ETH_DMASTS[10]：早发送中断 - ETH_DMASTS[13]：总线致命错误 只有未被屏蔽的中断才会影响到本位。 AIS 位的取值和以上位绑定，置'1'后，只有把造成该位置'1'的位清'0'(写'1')，才能把该位清'0'。
位 14	**ER**：早接收状态 (Early receive status) 置'1'时，该位表示接收到数据包时，DMA 填满了第一个缓存。在接收中断位(ETH_DMASTS[6])置'1'时，该位自动清'0'。
位 13	**FBE**：总线致命错误状态 (Fatal bus error status) 置'1'时，表示发生了总线错误，错误的具体情况见位[25:23]。在该位置'1'时，相应的 DMA 控制 器关闭全部总线访问。

位 12:11	保留
位 10	**ET**：早发送状态 (Early transmit status) 置'1'时，该位表示发送的帧已经完全位于发送 FIFO 中。
位 9	**RWT**：接收看门狗超时状态 (Receive watchdog timeout status) 置'1'时，表示接收到的帧长度超过 2048 字节。
位 8	**RPS**：接收流程停止状态 (Receive process stopped status) 在接收流程进入停止状态时，该位置'1'。
位 7	**RBU**：接收缓存不可用状态 (Receive buffer unavailable status) 表示接收描述符队列中的下一个描述符由 CPU 占有，DMA 无法取得。因此接收流程暂停。程序需要释放描述符，并发送"接收查询命令"来恢复接收流程。如果没有收到任何接收查询命令，那么 DMA 在接收到下一个帧的时候，会恢复接收流程。只有在 DMA 占有前一个描述符时，该位置'1'。
位 6	**RECV**：接收状态 (Receive status) 该位表示帧接收完成。帧的接收状态信息也更新到描述符中。接收流程仍处于运行状态。
位 5	**TU**：发送数据下溢状态 (Transmit underflow status) 该位表示发送缓存在发送帧的过程中发生数据下溢。这时，发送进程暂停并把数据下溢错误位(TDES[1])置'1'。
位 4	**RO**：接收溢出状态 (Receive overflow status) 该位表示接收缓存在接收帧的过程中发生溢出。如果不完整的帧已经转发给应用程序，则设置溢出错误位(RDES[11])为'1'。
位 3	**TJT**：发送啰嗦超时状态 (Transmit jabber timeout status) 该位表示发送啰嗦定时器超时，说明发送端过于繁忙。这时，中止发送进程并进入停止状态， 同时设置啰嗦超时位(TDES[14])为'1'。
位 2	**TBU**：发送缓存不可用状态 (Transmit buffer unavailable status) 该位表示发送描述符队列中的下一个描述符由 CPU 占有，DMA 无法取得。因此发送流程暂停。 位[22:20]指示了发送流程的当前状态。应用程序需要释放描述符，并发出"发送查询命令"来恢复发送流程。
位 1	**TPS**：发送流程停止状态 (Transmit process stopped status) 置'1'时，表示发送停止。
位 0	**TX**：发送状态 (Transmit status) 该位表示帧发送完成，帧的第一个描述符的 TDES[31]位已置'1'。

以太网DMA工作模式寄存器(ETH_DMAOPM)

地址偏移：0x1018

复位值：0x0000 0000

工作模式寄存器设定了FIFO接收和发送的工作模式和命令。在整个DMA的初始化流程中，应当最后写该寄存器。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留					DTCEFD	RSFD	DFRFD	保留		TXSF	FTXF	保留			TXTC
res					rw	rw	rw	res		rw	rw	res			rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXTC		STXE	保留					FEF	FUGF	保留	RXTC		OPSF	SRXE	保留
rw	rw	rw	res					rw	rw	res	rw	rw	rw	rw	res

位 31:27	保留

位 26	**DTCEFD**：不丢弃 TCP/IP 校验和错误帧 (Dropping of TCP/IP checksum error frames disable) 1：表示如果接收到帧仅有校验和检测模块发现的校验和错误时，MAC 不会丢弃该帧。这类帧仅在接收到的以太网帧的封装数据里有错误，而没有其他错误(包括校验错误)。 0：如果 FEF 位为'0'，MAC 丢弃所有有错的帧。
位 25	**RSFD**：接收存储-转发 (Receive store and forward) 1：表示一个帧只有在完全写入接收 FIFO 后，接收 DMA 才会把它转发给应用程序。并且此时 RXTC 位的取值会被忽略。 0：接收 FIFO 工作在直通模式，DMA 在接收帧写入接收 FIFO 的数据数目超过 RXTC 位设定的阈值 后，向应用程序转发帧数据。
位 24	**DFRFD**：不清空接收帧 (Disable flushing of received frames) 1：表示接收 DMA 不会因为接收描述符或者接收缓存不可用而清空接收 FIFO 里的接收到帧。 0：发生上述情况，接收 DMA 就清空接收 FIFO 里的接收到帧。(详见：接收流程暂停)
位 23:22	保留
位 21	**TXSF**：发送存储-转发 (Transmit store and forward) 1 ： 表示只有在一个帧完 全写入 发送 FIFO 后 ， MAC 才会把发送出去。并且此时 TXTC 位(ETH_DMAOPM[16:14])的取值会被忽略。 0：在待发送帧写入发送 FIFO 的数据数目超过 TXTC 位 (ETH_DMAOPM[16:14])设定的阈值后，MAC 开始发送帧。 只有在发送停止时，才能改变这位的设置状态。
位 22:20	**FTXF**：清空发送 FIFO (Flush transmit FIFO) 置'1'时，发送 FIFO 控制逻辑电路被复位到初始状态，这样发送 FIFO 里所有的数据被清空/丢 失。在清空操作完成后该位被自动清'0'。在该位为'0'之前，不允许写入工作模式寄存器。
位 19:17	保留
位 16:14	**TXTC**：发送阈值控制 (Transmit threshold control) 这三位控制发送 FIFO 的阈值。位于发送 FIFO 中待发送帧的数据数目超过该阈值时，发送开始。 另外，帧长小于该阈值的帧也会被发送出去。只有在 TXSF 位(位 21)为'0'时，这些位才有效。 000：64 100：40 001：128 101：32 010：192 110：24 011：256 111：16
位 13	**STXE**：开始/停止发送 (Start/stop transmission) 1：把发送进程置为运行状态，DMA 检查发送描述符队列的当前位置，搜索待发送的帧。DMA 要么从描述符队列的起始位置(由 ETH_DMATDLADDR 寄存器设定)获取描述符，要么从描述符队列里上次发送中止的位置获取描述符。如果 DMA 不占有描述符，则发送进程进入暂停状态，设 置发送缓存不可用位 (ETH_DMASTS[2])为'1'。只有在发送停止时，发送开始命令才有效。如果在未设置 ETH_DMATDLADDR 寄存器的情况下发出发送开始命令，会引起不可预料的后果。 0：在发送完当前帧后，发送进程进入停止模式。保存发送描述符队列里下一发送描述符的位 置，在传输重新开始时，这个描述符就变成当前描述符。只有在当前帧发送完成，或者发送进 程处于暂停状态时，停止发送命令才生效。
位 12:8	保留
位 7	**FEF**：转发错误帧 (Forward error frames) 1：表示除了过短帧以外，所有的帧都会转发给 DMA。 0：接收 FIFO 会丢弃有错误的帧(CRC 错误、冲突错误、巨人帧、看门狗超时、溢出)。然而，如 果在阈值模式下，已经把帧的起始字节指针转发给应用程序，则就不会丢弃该帧。接收 FIFO 会丢弃那些帧的起始字节还没有发送到 AHB 总线的出错帧。

位 6	**FUGF**：转发长度不够的”好”帧 (Forward undersized good frames) 1：接收 FIFO 把长度不够的”好”帧(帧长小于 64 字节但没有错误)，包括充填字节和 CRC 在内转发 给应用程序。 0：接收 FIFO 丢弃所有长度小于 64 字节的帧，除非这个帧由于小于接收阈值而已经被转发给应用程序了。
位 5	保留
位 4:3	**RXTC**：接收阈值控制 (Receive threshold control) 这两位设置了接收 FIFO 的阈值。在位于接收 FIFO 里，接收到帧的数据数目超过该阈值时，DMA 开始转发数据。另外，帧长小于该阈值的帧也会被自动转发。 *注意：如果接收 FIFO 的长度设为 128 字节，这些位的值不能取”11”。* *注意：只有在 RSFD 位(位 21)为’0’时，这些位才有效。* 00：64 01：32 10：96 11：128
位 2	**OPSF**：操作第二帧 (Operate on second frame) 置’1’时，发送 DMA 在接收到前一个帧的发送状态信息前，就开始发送下一个帧的数据。
位 1	**SRXE**：开始/停止接收 (Start/stop receive) 1：把接收进程置为运行状态，DMA 检查接收描述符队列的当前位置，用来处理下一个收到的帧。要么从描述符队列的起始位置(由 ETH_DMARDLADDR 寄存器设定)获取描述符，要么从描述符队列里上次接收中止的位置获取描述符。如果 DMA 没能占有描述符，那么接收进程进入暂停状态，设置接收缓存不可用位(ETH_DMASTS[7])为’1’。只有在接收停止时，”开始接收”命令才有效。在未设置 ETH_DMARDLADDR 寄存器之前发出”开始接收”命令，会引起不可预料的后果。 0：在转发完当前接收到帧后，接收 DMA 进入停止模式。保存接收描述符队列里下一接收描述符的位置，在传输重新开始时，这个描述符就变成当前描述符。只有在接收进程处于运行或者暂停状态时，”停止接收”命令才有效。
位 0	保留

以太网DMA中断使能寄存器(ETH_DMAIE)

地址偏移：0x101C

复位值：0x0000 0000

中断使能寄存器可以使能ETH_DMASTS寄存器反映的中断。把相应位置’1’可以使能相应的中断。在软件或者硬件复位后，将屏蔽所有的中断。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															NISE
res															rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AISE	ERIE	FBEIE	保留		ETIE	BWTIE	RXPSIE	RXBUIE	RXIE	TUIE	ROIE	TXJTIE	TXBUIE	TXPSIE	TXIE
rw	rw	rw	res		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位31:17	保留
位16	**NISE**：正常中断汇总使能(Normal interrupt summary enable) 1：使能正常中断。0：屏蔽正常中断。该位使能下列位： ETH_DMASTS[0]：发送中断； ETH_DMASTS[2]：发送缓存不可用； ETH_DMASTS[6]：接收中断； ETH_DMASTS[14]：早接收中断。

位15	**AISE**：异常中断汇总使能(Abnormal interrupt summary enable) 1：使能异常中断。 0：屏蔽异常中断。该位使能下列位： ETH_DMASTS[1]：发送流程停止； ETH_DMASTS[3]：发送啰嗦超时； ETH_DMASTS[4]：接收溢出； ETH_DMASTS[5]：发送下溢； ETH_DMASTS[7]：接收缓存不可用； ETH_DMASTS[8]：不可用； ETH_DMASTS[8]：接收流程停止； ETH_DMASTS[9]：接收看门狗超时； ETH_DMASTS[10]：早发送中断； ETH_DMASTS[13]：总线致命错误。
位14	**ERS**：早接收中断使能(Early receive interrupt enable) 1：正常中断汇总使能位(ETH_DMAIE[16])为'1'时，使能早接收中断； 0：屏蔽早接收中断。
位13	**FBES**：总线致命错误中断使能(Fatal bus error interrupt enable) 1：异常中断汇总使能位(ETH_DMAIE[15])为'1'时，使能总线致命错误中断； 0：屏蔽总线致命错误中断。
位12:11	保留
位10	**ETIE**：早发送中断使能(Early transmit interrupt enable) 1：异常中断汇总使能位(ETH_DMAIE[15])为'1'时，使能早发送中断； 0：屏蔽早发送中断。
位9	**RXWTIE**：接收看门狗超时中断使能(Receive watchdog timeout interrupt enable) 1：异常中断汇总使能位(ETH_DMAIE[15])为'1'时，使能接收看门狗超时中断。 0：屏蔽接收看门狗超时中断。
位8	**RXPSIE**：接收流程停止中断使能(Receive process stopped interrupt enable) 1：异常中断汇总使能位(ETH_DMAIE[15])为'1'时，使能接收流程停止中断。 0：屏蔽接收流程停止中断。
位7	**RXBUIE**：接收缓存不可用中断使能(Receive buffer unavailable interrupt enable) 1：异常中断汇总使能位(ETH_DMAIE[15])为'1'时，使能接收缓存不可用中断。 0：屏蔽接收缓存不可用中断。
位6	**RXIE**：接收中断使能(Receive interrupt enable) 1：正常中断汇总使能位(ETH_DMAIE[16])为'1'时，使能接收中断。 0：屏蔽接收中断。
位5	**TUIE**：发送下溢中断使能(Underflow interrupt enable) 1：异常中断汇总使能位(ETH_DMAIE[15])为'1'时，使能发送下溢中断。 0：屏蔽发送数据下溢中断。
位4	**ROIE**：接收溢出中断使能(Overflow interrupt enable) 1：异常中断汇总使能位(ETH_DMAIE[15])为'1'时，使能接收溢出中断。 0：屏蔽接收溢出中断。
位3	**TXJTIE**：发送啰嗦超时中断使能(Transmit jabber timeout interrupt enable) 1：异常中断汇总使能位(ETH_DMAIE[15])为'1'时，使能发送啰嗦超时中断。 0：屏蔽发送啰嗦超时中断。
位2	**TXBUIE**：发送缓存不可用中断使能(Transmit buffer unavailable interrupt enable) 1：正常中断汇总使能位(ETH_DMAIE[16])为'1'时，使能发送缓存不可用中断。 0：屏蔽发送缓存不可用中断。
位1	**TXPSIE**：发送流程停止中断使能(Transmit process stopped interrupt enable) 1：异常中断汇总使能位(ETH_DMAIE[15])为'1'时，使能发送流程停止中断。 0：屏蔽发送流程停止中断。

位0	**TXIE**：发送中断使能(Transmit interrupt enable) 1：正常中断汇总使能位(ETH_DMAIE[16])为'1'时，使能发送中断。 0：屏蔽发送中断。

对于以太网中断，只有在DMA状态寄存器的TST位或者PMT位为'1'，并且相应中断没有被屏 蔽时，或者在NIS/AIS位置'1'，且相应中断被使能时，中断才会发生。

以太网DMA丢失帧和缓存溢出计数器寄存器(ETH_DMAMFBOCNT)

地址偏移：0x1020

复位值：0x0000 0000

DMA有2个计数器，用来统计丢失帧的数目。本寄存器包含了计数器的当前值。这个计数器通常用作故障诊断。位[15:0]是由于AT32F407xx的缓存不可用(接收描述符不可用)而丢失帧的数目。位[27:17]是由于接收FIFO溢出及帧过短(正确但是小于64字节的帧)而丢失帧的数目。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留			OFOC	MFA											OMFC
res				rc_r											

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MFC															
rc-r															

位31:29	保留
位28	**OFOC**：FIFO溢出计数器溢出位(Overflow bit for FIFO overflow counter)
位27:17	**MFA**：应用程序丢失的帧(Missed frames by the application) 这些位表示了应用程序丢失帧的数目。
位16	**OMFC**：丢失帧计数器溢出位(Overflow bit for missed frame counter)
位15:0	**MFC**：控制器丢失的帧(Missed frames by the controller) 这些位表示了由于MCU的接收缓存不可用而丢失帧的数目。每当DMA丢弃一个输入帧时，这个数器加1。

以太网DMA当前发送描述符寄存器(ETH_DMACTD)

地址偏移：0x1048

复位值：0x0000 0000

当前发送描述符寄存器指向DMA正在读取的发送描述符起始地址。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
HTDAP															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HTDAP															
r															

位31:0	**HTDAP**：发送描述符地址指针(Host transmit descriptor address pointer) 这些位在复位时清'0'，由DMA在工作过程中更新。

以太网DMA当前接收描述符寄存器(ETH_DMACRD)

地址偏移：0x104C

复位值：0x0000 0000

当前接收描述符寄存器指向DMA正在读取的接收描述符起始地址。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
HRDAP															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

HRDAP

r

位31:0	**HRDAP**：接收描述符地址指针(Host receive descriptor address pointer) 这些位在复位时清'0'，由DMA在工作过程中更新。

以太网DMA当前发送缓存地址寄存器(ETH_DMACTBADDR)

地址偏移：0x1050

复位值：0x0000 0000

当前发送缓存地址寄存器指向DMA正在读取的发送缓存的地址。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
HTBAP															

r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HTBAP															

r

位31:0	**HTBAP**：发送缓存地址指针(Host transmit buffer address pointer) 这些位在复位时清'0'，由DMA在工作过程中更新。

以太网DMA当前接收缓存地址寄存器(ETH_ DMACRBADDR)

地址偏移：0x1054

复位值：0x0000 0000

当前接收缓存寄存器地址指向DMA正在读取的接收缓存地址。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
HRBAP															

r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HRBAP															

r

位31:0	**HRBAP**：接收缓存地址指针(Host receive buffer address pointer) 这些位在复位时清'0'，由DMA在工作过程中更新。

25 版本历史

文档版本历史

日期	版本	变更
2020.01.13	1.00	最初版本
2020.02.15	1.01	更新系统时钟最高频率到 240M，更新 APB 最高时钟频率到 120M

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